CCF CFTC | 2023

第二十届全国容错计算学术会议

论文集

目 录

1. Efficient DfT Architecture and Test Vector Reordering Scheme for 3D-Stacked Chiplets · 1
2. Identifying Reliability High-Correlated Gates of Logic Circuits with Pearson Correlation Coefficient ····· 16
3. Fine-grained Pruning for RRAM-based Accelerator with Redundant Weight Layer-wise Searching ····· 21
4. 通用处理器性能测评基准负载设计与分析 ····· 29
5. 目标检测算法失效模式的抽取与验证 ····· 36
6. Test Cost Reduction for VLSI Adaptive Test with K-Nearest Neighbor Classification Algorithm ····· 46
7. Low-latency and Reconfigurable Hardware Accelerator for Complex Functions Using Approximate Rotation CORDIC-Based Parallel Method ····· 53
8. 面向自动驾驶的复杂场景建模及测试用例生成方法 ····· 65
9. 目标遥感数据检测与识别模型 ····· 77
10. FusionFL: A statement-level feature fusion based fault localization approach ····· 85
11. 一种基于 Selenium 复杂多层 web 系统自动化测试方法 ····· 95
12. Vivado 综合器安全状态机精确行为分析 ····· 101
13. 基于边界度和局部密度的不平衡数据软件缺陷预测过采样方法 ····· 113
14. A Novel High Throughput RO-Based True Random Number Generation on FPGA ····· 128
15. 基于低测试逃逸的晶圆级适应性测试方法 ····· 133
16. LRHOP：一种基于时空特征的 DeepFake 集成检测方法 ····· 141
17. 图结构深层语义捕获软件缺陷预测 ····· 151
18. 采用低精度信号源和低精度 ADC 测量低失真放大器的频谱性能 ····· 162
19. 基于区块链和可追溯 CP-ABE 的电子病历存储和共享方案 ····· 171
20. MRFI: An Open Source Multi-Resolution Fault Injection Framework for Neural Network Processing ····· 181

21. 基于最大延迟调度的 Android 应用并发相关不稳定测试检测 …… 190

22. 低密度奇偶校验（LDPC）码中陷阱集相关研究进展 …… 204

23. 基于振动传感器的预测性维护系统安全优化 …… 247

24. Test Primitive: A Unify Feature of Minimal Test Sequence for Memory Faults …… 262

25. 面向目标检测模型的高效能安全可重构处理器 …… 271

26. Secure Conversion Circuit from Arithmetic to Boolean Masking …… 280

27. 基于 FTSTM 的工控协议可信性验证 …… 285

28. 机器人路径规划算法的高层综合设计研究 …… 296

29. 一种模型驱动的装备嵌入式软件自动化测试方法 …… 305

30. 基于混沌映射的抗机器学习攻击强 PUF 设计 …… 314

31. SI PUF: An SRAM-and-Inverter-based PUF with Bit Error Rate 0.0053% and 0.073 …… 324

32. DeDfTr: Defeating Design-for-Trust by Hardware Trojan-based Cross Computing Attacks …… 329

33. 航天嵌入式软件典型缺陷建模 …… 341

34. 一种基于权重范围变换的忆阻神经网络容错方法 …… 349

35. An Efficient Hybrid Approximate Adder Tree Based on Mismatch Propagation Analysis for CNN Deployment …… 365

36. 面向区块链数字资产的智能合约缺陷分析 …… 370

Efficient DfT Architecture and Test Vector Reordering Scheme for 3D-Stacked Chiplets

Wenfa Zhan[1,2], Qingping Zhang[1*], Jiangyun Zheng[1], Xueyuan Cai[1], Pan Pan[1], Xiaoqing Wen[3]

1. School of Electronic Engineering and Intelligent Manufacturing, Anqing Normal University, Anqing 246133,

2. China RF Integration and Microassembly Technology National and Local Joint Engineering Laboratory, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

3. Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Fukuoka 804-8550, Japan

* Corresponding author, Email: ttzgyrsflh@aliyun.com.

Abstract: With the rapid development of 3D-stacked ICs, test issue is becoming increasingly prominent. Flexible access to targeted chiplets or IP-cores and optimized test vector sequences to improve test efficiency are highly required. In this work, based on the IEEE 1838 protocol, an enhanced DfT architecture, named QS-PTAP, is developed for 3D chip testing. It provides a convenient and efficient way to chiplet and IP-core selection, and features inter-die scan path automatic configuration that is unavailable in the existing DfT architectures. A test access mechanism, which comprises of the proposed QS-PTAP and SIBs in IEEE 1687, is proposed to achieve feasible modular testing. Furthermore, a test vector reordering scheme dedicated to the modular testing of 3D-stacked chips is designed to improve the efficiency of detecting faulty chips. Circuit simulation demonstrates that the scan chain configuration is significantly simplified with a time reduction of about 55%. For the studied cases, the proposed test vector reordering scheme can reduce the simulated time for faulty chip detection by more than 45%.

Key words: Chiplet test; Design for test; Test access mechanism; Test pattern reordering.

1. Introduction

3D-stacked packaging is a promising technology in the post-Moore era. Multiple chiplets are integrated into one chip via internal interconnections. Fabrication cost can be significantly reduced by improved yields for smaller chiplets and by adopting a proper process for each chiplet flexibly [1-3].

Chiplet testing is conducted throughout the whole manufacturing and packaging stages. The IEEE 1838 protocol (2019) provides systematic solutions for the chiplet testing of 3D/5.5D stacked ICs [4]. The test access port (TAP) in IEEE 1149 [5] is reused, and a new feature of inter-die connectivity is introduced. In recent years, other advanced 3D test methods have been explored. A design-for-test (DfT) architecture based on the IEEE 1500 wrapper [6] was proposed [7]. An architecture for testing the active mediation layer of the 3D chip was proposed [8] based on the IEEE 1687 [9]. A configurable multi-die parallel test architecture and a test vector reuse method were proposed [10]. A new test approach to transmitting test vectors via the existing

high-speed interfaces in the chiplet was also proposed, and USB and PCIe ports were employed for 3D testing [11].

In 3D chip testing, each test vector usually functions on only a few test block in order to shorten the scan chain, resulting in considerable inter-die scan path reconfiguration operations. IEEE 1838 provides a solution for scan path configuration by introducing a 3DConf register to the primary TAP (PTAP) to control whether the chiplets associated with the secondary TAPs (STAPs) are loaded onto the JTAG daisychain [4]. When configuring the daisychain, the STAPs of each chiplet in the stacked IC are opened or closed layer by layer, and thus the configuration codes should be shifted into the 3DConf registers multiple times. Therefore, the extra time for inter-die scan path configuration is long, leading to an increased test time.

Recently, a modified PTAP controller for 3D IC test was put forward [12]. New states for the direct access to the 3DConf register are introduced to the PTAP finite state machine (FSM), and time reduction of the JTAG daisychain configuration was achieved. Yet similar to IEEE 1838, the inter-die daisychain has to be configured layer by layer, and its operation becomes fussy especially for 5.5D chips with more complicated stacking. Moreover, for both available schemes in IEEE 1838 and Ref. [12], the daisychain cannot be established automatically according to the selected chiplets, and its configuration depends on the previous scan path, which makes it difficult to maintain and understand the test plan. Hence, it is highly important to simplify the operation of scan path configuration for test cost reduction.

Test vector reordering is another approach to test cost deduction. Dynamic test vector reordering methods were put forward based on k-nearest neighbors [13], improved SVMRANK algorithm [14], gamma distribution [15] and stochastic heuristic approach [16], resulting in higher test efficiency. Yet these previous methods were designed for a chip with a single die inside, rather than 3D-stacked multiple dies with frequent inter-die scan path alteration. Once the scan path switches, time needed for scan chain configuration and an additional test vector for shifting out the previous test response should be considered. This is unique to 3D-stacked chip testing.

In this work, a new test controller, named QS-PTAP, is proposed for 3D-stacked IC testing, where QS means quick selection. Compared with the existing schemes for inter-die scan chain configuration, this flexible and efficient DfT architecture provides a simplified and concise operation for chiplet quick access, and features scan chain automatic configuration and test time reduction. In addition, each scan path configuration is independent from previous scan paths, which is intuitive and convenient for operation. The proposed architecture is compatible with the standard JTAG interface and thus any existing IC test machine. Based on the QS-PTAP and IEEE 1687 SIB infrastructure, a feasible modular test flow is further developed. Moreover, by considering the time cost of scan path configuration and additional test vectors introduced from

the scan path switching, as well as cooperating with the QS-PTAP, a new test vector reordering scheme dedicated to the modular testing of 3D-stacked chip, is also proposed to increase the faulty chip detection efficiency.

2. DfT Architecture Description

The DfT architecture in each chiplet consists of an enhanced PSTAP mentioned above, several STAP interfaces for enabling or disabling this chiplet to plug into the PTAP of the next chiplet, and several SIB infrastructures defined in IEEE 1687.

An illustrative conceptual overview of the DfT architecture is depicted in Fig.1. This example 5.5D-stacked chip contains 6 chiplets, where chiplets 0 and 1 are side-by-side stacked on top of chiplet 3. Each chiplet contains a QS-PTAP to implement the traditional PTAP controller logic, along with new features of chiplet quick selection and JTAG daisychain automatic configuration. In chiplets 5 and 3, several SIBs are introduced to support modular tests. If an SIB is 1, the scan chain segment of the IP-core is included in the JTAG daisychain. All SIBs are 0 at power-on reset, and thus these cores are excluded by default. All external I/O pins are in the bottom chiplet 5. In this design, SIB bit is after the corresponding test pattern, as illustrated in Section F.3.1.1 of Ref. [9]. JTAG ports in the chiplet are compatible with IEEE 1149.

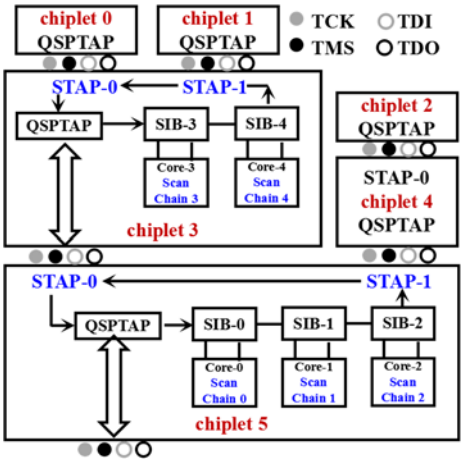


Fig. 1 An illustrative conceptual overview of the DfT architecture.

This DfT architecture is based on two test access mechanisms (TAMs):

(1) A newly proposed QS-PTAP controller that automatically configures multiple STAPs and establishes the *inter-die* active JTAG daisychain.

(2) IEEE 1687 SIBs concatenated on an *intra-die* scan chain that controls whether or not a bypassable IP-core is included in the test.

In the following sections, the number of chiplets in the stacked chip is N_c. These chiplets are numbered from 0 to N_c-1 in sequence, starting from the top to the bottom of the stacked layers. For chiplets at the same layer, the numbering order has no special requirement. Fig. 1 shows a numbering example.

Denote the number of chiplets on top of the back side of the *k*-th chiplet by M_k. It is also the number of the STAPs in the *k*-th chiplet. In Fig. 1, M_5 and M_3 are 2 and the others are 1.

2.1 QS-PTAP implementation

The QS-PTAP consists of the following main components: A conventional PTAP FSM, a counter from 0 to N_c-1 (denoted by qs3d_cnt below), and a group of dedicated automatic configuration registers (denoted by qs3d_reg below).

Based on the FSM in IEEE 1149, a new state, named quick-select, is added to the proposed QS-PTAP FSM, as shown in Fig. 2. This new state features chiplet quick selection and scan chain automatic configuration. Modifications to the original FSM are marked in red in the figure.

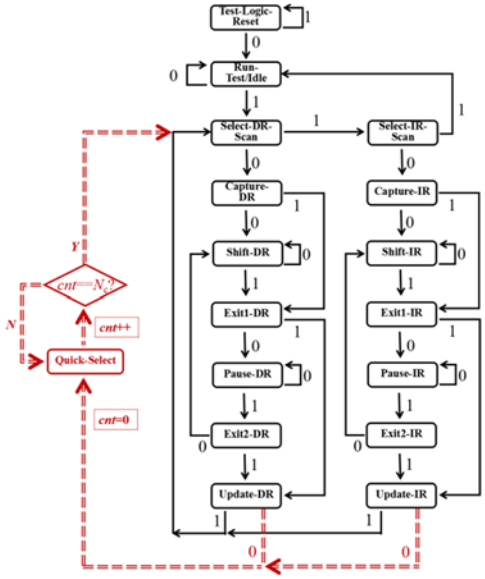


Fig. 2 FSM of the proposed QS-PTAP.

Once the FSM is under the quick-select state, qs3d_cnt counts from 0 to N_c-1, and adds 1 in each TCK cycle. For the *k*-th chiplet, when its qs3d_cnt is *k*, the input TMS signal in this TCK cycle indicates the *test enabling bit* of this chiplet. The test enabling bit is 1 if this chiplet or some IP-cores in the chiplet need testing in the subsequent test cycle.

The qs3d_reg register in the *k*-th chiplet contains M_k+1 bits. The 1st bit (qs3d_reg[0]) stores the *test enabling bit* of this chiplet, and the remaining bits (qs3d_reg[1:M_k]) store the *load enabling bits* of the 1st to the M_k-th STAPs in this chiplet, respectively. Each load enabling bit refers to the connectivity information of an STAP in the chiplet.

An STAP controls whether its TDO port connects to the TDI port of the PTAP in the chiplet on top of it. The load enabling bit is asserted if these two ports are connected and therefore the next chiplet above this STAP is plugged into the active JTAG daisychain, and vice versa. Note that an asserted load enabling bit does not indicate that the chiplet above this STAP needs testing in the next cycle. This is because chiplet may only provide an access to another chiplet above it in the

active scan path.

In each chiplet, its test enabling bit is evaluated by the TMS input as described above, and the load enabling bits are automatically evaluated via the TDO port of the chiplet above this STAP under the quick-select state, using the schemes proposed in Section 2.2.

Fig. 3 depicts the QS-PTAP and its associated STAPs. The number of STAPs and load enabling bits are 3 in this figure. Modify this number in DfT design to adjust the actual chiplet stack. Under quick-select state where qs3d signal asserts, the load enabling bits for automatic configuration are transferred across the chiplet through the multiplexers and the reused TDI and TDO ports. Under other conventional states defined in the IEEE 1149 FSM, these multiplexers are switched to form the multi-die JTAG daisychain for shifting the serial test vectors.

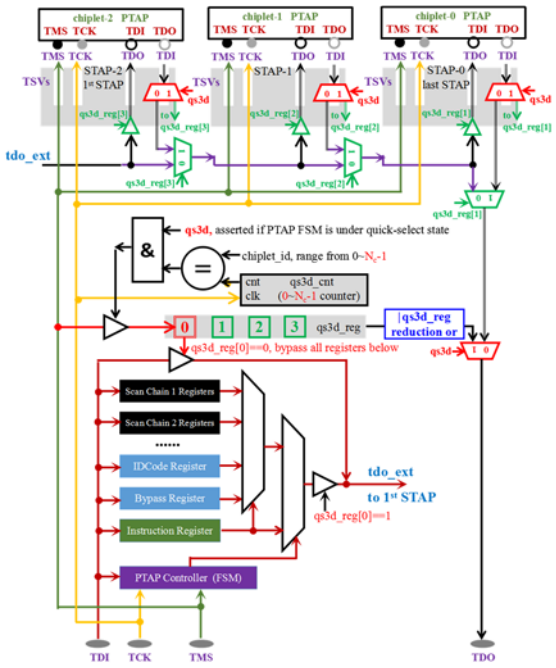


Fig. 3 Infrastructure of the QS-PTAP and the associated STAPs.

2.2 Scan chain automatic configuration

Chiplet quick selection and active daisychain automatic configuration are processed under the above-mentioned *quick-select* state. Denote the *k*-th chiplet and the chiplet below it by chiplet-A and chiplet-B respectively in this section. Automatic configuration scheme under this state is as follows.

Chiplet-A gains its test enabling bit from the TMS input at the *k*-th TCK cycle, and stores this value to qs3d_reg[0]. Before this TCK cycle, all load enabling bits are ready and have been stored to qs3d_reg[1:M_k]. They are given by the TDO ports of the chiplets above chiplet-A, and captured by chiplet-A from its TDI ports of the corresponding STAPs. Subsequently, at the next TCK cycle, all M_k+1 bits in the qs3d_reg register of chiplet-A are reduction-ORed. The reduction-OR result is immediately sent to the TDO port of the QS-PTAP in chiplet-A, then obtained by the TDI port of

the STAP in chiplet-B, and finally written to the qs3d_reg register in chiplet-B as a load enabling bit. The corresponding STAP in chiplet-B is automatically configured simultaneously, according to this load enabling bit.

Repeat above operation M_k times under the quick-select state, starting from the first chiplet numbered 0 at the top layer, and ending with the last chiplet numbered M_k-1 at the bottom layer, After these M_k TCK cycles, all STAPs are configured as the expected on or off state, and consequently the whole active JTAG daisychain across multi-chiplets is established automatically.

Under the shift-IR and shift-DR states, the shift operation for the instruction register (IR) and all scan registers should be disabled if the test enabling bit of this chiplet is 0. In a chiplet with asserted test enabling bit, these registers shift as usual under the shift-IR and shift-DR states.

Taking Fig. 1 as example. If chiplet-1 and chiplet-4 need testing later, the serial input at the TMS pin of the chip is 6'b010010 *under quick-select state*. The value of qs3d_reg register in each chiplet at every TCK cycle is illustrated in Fig. 4. Note that the qs3d_reg registers in chiplets 0~2 have no load enabling bits, since these chiplets are at the top layer.

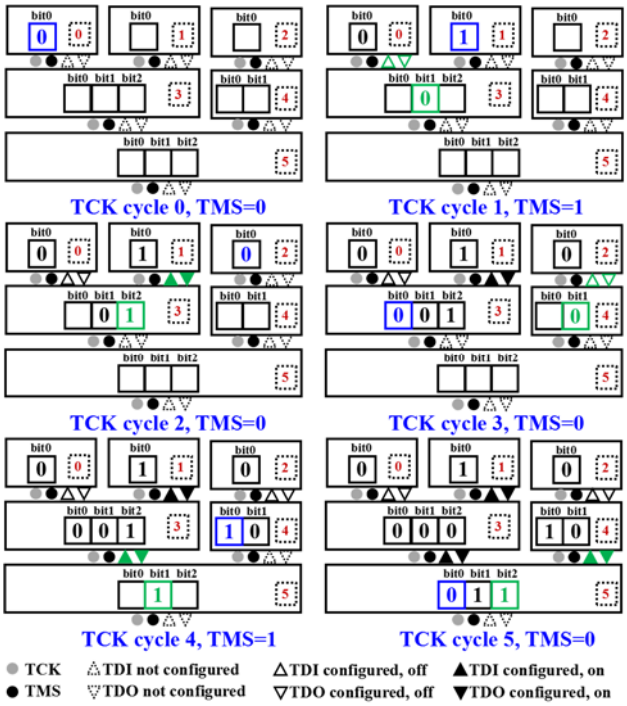


Fig. 4 The value of qs3d_reg register at every TCK cycle *under quick-select state*.

In this figure, square boxes with solid lines represent qs3d_reg register. The blank box means that the register has not been configured by far, the blue box highlights the test enabling bit evaluated via TMS input at the current TCK cycle, and the green box highlights the load enabling bit evaluated via reduction-OR result of the qs3d_reg in the chiplet above it at the current TCK cycle. The upward (downward) triangle represents the TDI (TDO) port of the STAP in the lower chiplet; and it is also the TDO (TDI) port of the QS-PTAP in the upper chiplet. The triangle with dashed lines means that this path has not been configured by far. The hollow triangle with solid

lines means that this path is configured as off/disconnected state. The filled triangle means that it is configured as on/connected state. The green triangles highlight the TDI and TDO ports configured at current TCK cycle.

Clearly, under quick-select state, the automatic configuration is conducted from the top layer down to the bottom layer, in the ascending sort of chiplet numbering. During TCK cycles 1~5, all 5 STAPs in these chiplets are setup one by one according to the corresponding load enabling bits.

After N_c=6 TCK cycles, all QS-PTAP FSMs in these chiplets exit the quick-select state and enter the select-dr-scan state simultaneously. At this moment, as shown in Fig. 3, the qs3d signal changes to 0, and the multiplexers controlled by qs3d are switched. As a result, the multi-die data paths for transferring load enabling bits turn off, and the JTAG path across the specified chiplets is connected automatically.

2.3 Modular test flow

Based on the proposed DfT hardware architecture for the 3D-stacked ICs, the modular testing of chiplets or IP-cores in the chiplet is flexible. This is achieved by combining two TAMs: Inter-die chiplet access based on the proposed QS-PTAP and intra-die IP-core insertion based on the IEEE 1687 SIB infrastructure. The general modular testing flow is as follows.

First, select chiplets to be tested under the quick-select state, where N_c bits are shifted in from the TMS pin in series. Details are described in the above section. An active scan chain across the selected chiplets is available after automatic configuration.

Then, choose IP-cores in the selected chiplets by asserting the corresponding SIBs. Under *shift-DR state*, the SIBs with IP-cores to be tested (or not) in the next test cycle are set as 1 (or 0), and test vectors for the previously selected cores are shifted in via serial TDI input. These new SIB settings will take effect under the succedent *update-DR* state, and the selected IP-cores will be inserted into the JTAG daisychain and ready for the next test cycle.

2.4 Test vector reordering scheme for 3D-stacked IC

For 3D ICs, modular testing is employed to separately check each test block, namely a chiplet or an IP-core. The test vector length and the scan path for each test block are different. For the common test sequence, test vectors of the same test block are grouped together and ranked in an order according to their fault ratios, and the next test block is examined after its previous block. Hence, some test vectors with a high fault ratio in the last few test blocks are tested quite late. These vectors can move forward in the test sequence to improve the fault detection efficiency. However, after this operation, the scan path of a forward-adjusted test vector may differ from those of its new neighbored test vectors. This results in a succedent additional test vector with the

same length and scan path in order to shift out or compact the test response of that forward-adjusted test vector, and one more scan path configuration, both of which increase the test time.

To take advantage of the forward adjustment of those test vectors with a high fault ratio, and reduce scan chain alteration as much as possible, a test vector reordering scheme is designed, cooperating with the QS-PTAP architecture to simplify the scan path configuration. This scheme is especially suitable for the situation where the yield is not very high and the fault ratios of some test vectors are significantly larger than others, which are always the case for 3D-stacked chip testing.

For a test vector sequence, the expectation of TCK cycles for testing a 3D chip (L_{exp}) is:

$$\begin{aligned}
&L_{exp,fault} = L_0 + \sum_{i=1}^{N_{tv}-1} Q_{i-1} \cdot S_i + Q_{N_{tv}-1} \cdot L_{N_{tv}-1} \\
&Q_0 = 1,\ Q_i = (1-P_{i-1})Q_{i-1}\ (i = 1 \sim N_{tv} - 1) \\
&S_i = \begin{cases} L_i & (if\ T_i = T_{i-1}) \\ L_{i-1} + L_{conf} + L_i & (if\ T_i \neq T_{i-1}) \end{cases} (i = 1 \sim N_{tv} - 1) \\
&L_{pass} = L_0 + \sum_{i=1}^{N_{tv}-1} S_i + L_{N_{tv}-1} \\
&L_{exp} = (1-Y) \cdot L_{exp,fault} + Y \cdot L_{pass}
\end{aligned} \tag{1}$$

where $L_{exp,fault}$ is the expectation of TCK cycles for a *faulty chip* to find out its defect; N_{tv} is the number of all test vectors; Y is the yield; L_{pass} is the TCK cycles to test a normal chip; L_i and P_i are the length and the fault ratio of the *i*-th test vector in the test sequence respectively; and L_{conf} is the TCK cycles to configure a scan path. The *i*-th test vector belongs to the T_i-th test block. Once a test vector fails, the followed test vectors are skipped.

The common test vector sequence is as follows:

$$\begin{matrix}
tv_{0,0} & tv_{0,1} & \cdots & tv_{0,Ltv_0-1} \\
tv_{1,0} & tv_{1,1} & \cdots & tv_{1,Ltv_1-1} \\
\cdots & \cdots & \cdots & \cdots \\
tv_{N-1,0} & tv_{N-1,1} & \cdots & tv_{N-1,Ltv_{N-1}-1}
\end{matrix}, \tag{2}$$

where $tv_{i,j}$ is the test vector with the *j*-th largest fault ratio in the *i*-th test block, *N* is the number of test blocks, and Ltv_i is the number of test vectors for the *i*-th test block. All subscripts start from 0 in this section.

Starting from this common sequence where test vectors are grouped by test block, a test vector reordering scheme to reduce the time of distinguishing a faulty 3D-stacked chip is as follows.

(I) Read common test vector sequence tv0 described in Eq. 2, and the fault ratio of each test vector from input file. The fault ratio P_i of the *i*-th test vector in the test sequence is defined as the number of faulty chips that are detected by this test vector over the total number of *faulty chip*s.

The sum of all fault ratios is normalized as 1. In practice, the fault ratios are gathered by employing all test vectors to each faulty chip and counting large amount of faulty chips.

(II) Rank tv0 according to the fault ratio of each test vector in descending order, and generate a new sequence tv2.

(III) Accumulate fault ratios in tv2 sequence from the 1st to the *k*-th test vector, until:

$$\sum_{i=0}^{k-1} P_i > P_{th} \cdot \tag{3}$$

where P_{th} is the accumulated threshold for defining test vectors with a high fault ratio (HFR), and k is the *initial* number of HFR vectors. These vectors are tested the most forward.

For a test block, if the number of HFR vectors belong to it is one, and the fault ratio of this HFR vector ranks in the last quarter of all HFR vectors, then this HFR vector is removed from the HFR list since the additional time that stems from more scan path alterations probably exceeds the time reduction effect of the test priority promotion. Denote the *actual* number of HFR vectors as N_{hfr}.

(IV) Allocate a new test sequence tv. Its first N_{hfr} test vectors are HFR ones in tv2, ranking in descending order. Remove these HFR vectors in tv0 sequence, and append its remaining part to tv. Thus the initial tv sequence begins with N_{hfr} HFR vectors, followed by the remaining part of tv0 which is grouped by test block. This method promotes the test priority of HFR vectors without introducing too many scan path alterations, and hence is expected to reduce test time.

(V) Reorder above tv sequence to further decrease L_{exp} determined by Eq. 1, as described by the pseudo code below.

Algorithm 1 Reordering of test sequence tv.

```
calculate L_exp for original tv via Eq. 1, the result is L_exp
tv2[:] <= tv[:]      //copy tv as tv2
for (id=1; id<=N_hfr-1; id++):
   if test vectors tv2[id] and tv2[id-1] belong to the same test block: continue
   else:
      j <= tv2[id-1].test_block_type ;    //tv2[id-1] belongs to the j-th test block
      search next test vector that belongs to j-th test block between tv2[id+1] and tv2[N_tfr-1]
      if matched test vector tv2[i] is found:
         //move tv2[i] forward to tv2[id] so that tv2[id-1] and tv2[id] belong to the same
         //test block to avoid frequent scan chain alteration; and accordingly, shift
         //original tv2[id]~tv2[i-1] backward by one in the test sequence
         {tv2[id], tv2[id+1]~tv2[i]} <= {tv2[i], tv2[id]~tv2[i-1]} //exchanging order
      else: continue    //if no test vector matches
   calculate L_exp for tv2 via Eq. 1, the result is L_tv2
   if L_tv2<L_exp:    //if L_exp is further reduced
      tv[:] <= tv2[:]       //apply reordered tv2 to tv
      L_exp <= L_tv2
   else:
      tv2[:] <= tv[:]      //restore tv2 as tv before exchanging order
return the optimized expectation L_exp, and the final reordering test sequence tv
```

3. Experimental Results

To evaluate the proposed QS-PTAP, DfT TAMs and test vector reordering scheme, a 3D-stacked IC with 6 chiplets shown in Fig. 1 was established as an illustrative demonstration. The circuits under test (CUTs) of chiplets 0, 1, 2, and 4 are b13 from ITC99 benchmark. These chiplets contain no SIBs. CUTs of all 5 SIB IP-cores in chiplets 3 and 5 are b11 from ITC99 benchmark. The width of IR in PTAP is 4 bits.

The QS-PTAP controller, SIB infrastructure and other involved components were implemented by RTL-level verilog codes. The Icarus verilog (iverilog) simulator was applied for circuit validation. The GTKwave viewer was adopted to visualize the waveform of simulation results. The test vector reordering scheme was implemented by C++ codes. The g++ compiler was used for compilation. All toolchains involved in the experiments are free software under the GNU General Public License.

3.1 DfT architecture simulation

Three test cycles were simulated. Each test cycle had different active JTAG daisychain, and contained two test vectors. The latter vector was to shift out the test response of the former one and configure a new JTAG daisychain for the next test cycle. In the 1st cycle, only chiplets 0 and 1 were selected to test. In the 2nd cycle, cores associated with SIB3, SIB2 and SIB0, and chiplet 2 were tested. In the 3rd cycle, cores associated with SIB4 and SIB1, and chiplet 4 were tested.

The simulated waveform is shown in Fig. 5. Time fragments during the quick-select state where the qs3d signal is asserted are elongated to show more details.

Fig. 5 Simulated waveform of the QS-PTAP design.

The modular testing flow for test cycle 1 is as follows. Suppose that the starting state of the

FSM is update-DR, which is the final state of the previous test cycle.

Step 1: Chiplet Selection. De-assert TMS to enter the *quick-select state*. Then serial input 6’b110101 is shifted in from the TMS pin to select the chiplet to be tested, i.e., chiplets 0, 1 , 3 and 5. Chiplets 3 and 5 are additionally selected because they need testing in the next cycle 2, and thus their SIBs should be configured previously in this test cycle. During the quick-select state, the qs3d_reg[0] bit of a selected chiplet *i* becomes 1 at the *i*-th TCK cycle where the qs3d_cnt counter is *i*, and the reduction-or of all qs3d_reg bits are exported via the reused TDO port of this chiplet and then transmitted to the chiplet below it. After 6 TCK cycles, the quick-select state is done and qs3d_cnt becomes 0. Note that STAP-1 of chiplet 5 automatically turns off since the chiplets 4 and 2 above this STAP are not tested.

Step 2: Inter-Die Scan Path Setup. Under the *shift-IR state*, four INTEST instructions are input in series from the TDI pin to fill the IRs of the previously selected chiplets. These instructions will load the internal scan chains of the selected chiplets onto the JTAG daisychain.

Step 3: Intra-Die Scan Path Setup. Under the *shift-DR state*, the serial TDI input contains test vectors for the tested chiplets and all 5 SIB bits to enable or disable the scan chain segments of the corresponding cores. For the former test pattern, the serial input stream is {test vector of chiplet 0, test vector of chiplet 1, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0}, where all SIB bits are de-asserted. For the latter test pattern, it is {test vector of chiplet 0, test vector of chiplet 1, 1'b0, 1'b1, 1'b1, 1'b0, 1'b1}, where SIB3, SIB2 and SIB0 bits are asserted to choose their related IP-cores for next cycle 2. The shift-DR operation is forbidden for those unselected chiplets or cores. The response of the former test vector is shifted out in series via the TDO pin when the latter vector is shifted in from the TDI pin. The SIB bits become effective and the related IP-cores are inserted into the JTAG daisychain under the *update-DR state* that followed.

The modular testing flow for test cycle 2 is similar to that for cycle 1 above.

(I) Under *quick-select state*, serial input 6’b001101 is input from the TMS pin to select chiplets 2, 3 and 5. Note that although chiplet 4 is not tested, it is still automatically loaded onto the JTAG daisychain and bypassed, since it provides a data path to access chiplet 2 which needs testing.

(II) Under *shift-DR state*, there are two test patterns in this test cycle. For the former one, the TDI serial input is {1'b0 for SIB4, test vector of SIB3 core, 1'b1 for SIB3, test vector of chiplet 2, test vector of SIB2 core, 1'b1 for SIB2, 1'b0 for SIB1, test vector of SIB0 core, 1'b1 for SIB0}. For the latter one, the TDI serial input is {1'b1 for SIB4, test vector of SIB3 core, 1'b0 for SIB3, test vector of chiplet 2, test vector of SIB2 core, 1'b0 for SIB2, 1'b1 for SIB1, test vector of SIB0 core, 1'b0 for SIB0}. Note that SIB bits in the former test pattern remain the previous values assigned in cycle 1 to maintain the same JTAG daisychain for the latter test pattern. SIB bits for

the next test cycle 3 are assigned in the latter pattern, and then cores associated with SIB4 and SIB1 are ready for next test cycle after *update-DR state*.

In test cycle 3, serial input 6'b000111 is shifted in from the TMS pin during *quick-select state* to select chplets 5, 4 and 3. Then STAP0 of chiplet 4 turns off, scan registers in chiplet 4 are loaded, and cores with previously asserted SIBs are inserted, so as to alter the JTAG daisychain from the configuration of cycle 2 to that of cycle 3.

This test architecture features configuration time saving. For instance, to configure the inter-die scan path in test cycle 1, the number of TCK cycles in steps 1 and 2 above is 28. while it is 63 via the IEEE 1838 approach. Test time is significantly reduced by 55%.

3.2 Test vector reordering simulation

The reordering scheme was applied to four situations where the fault ratios of test vectors were randomly generated with different random seeds. For each situation, 10000 faulty chips with randomly specified failure test vectors that are proportional to their fault ratios were simulated, and the TCK cycles needed to detect their faults using both the common sequence (L_{f0}) and the proposed reordered sequence (L_f) were gathered. Parameters Y and P_{th} in section 2.4 were 0.8 and 0.6, respectively. Test blocks were SIB cores (SIB0~SIB4) and chiplets (0~2 and 4) in the 3D-stacked configuration. Here the length of a test vector equals the number of the internal scan registers of b11 or b13 circuit. For b11 and b13, their test vector numbers were 31 and 53, respectively. These test vectors were independent and non-redundant, which means that each test vector was able to detect at least one unique fault that other test vectors cannot detect.

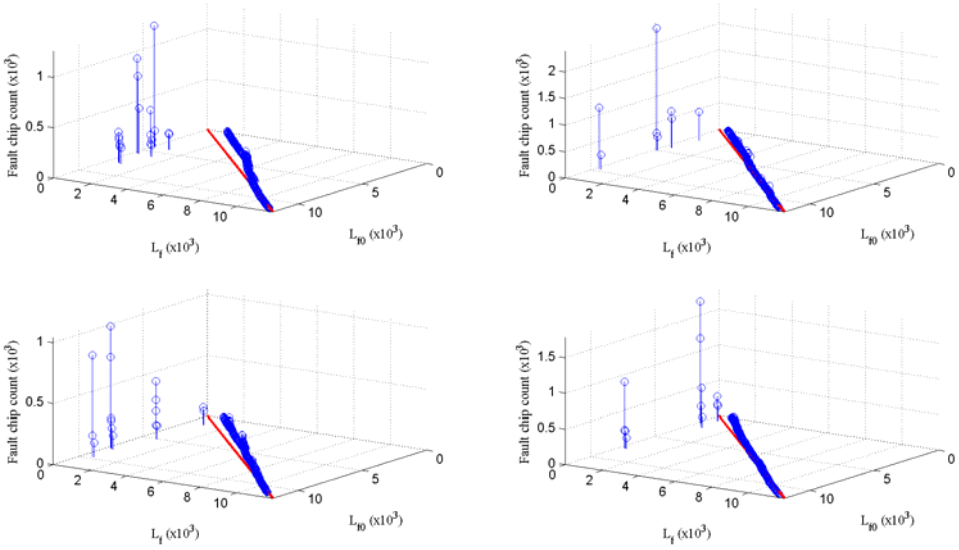


Fig. 6 Simulation results for the proposed test vector reordering scheme.

Fig. 6 shows the simulation results. In the figure, each stem stands for a kind of faulty chips

detected by the same test vector, and its height represents the number of this kind of faulty chips. The X and Y coordinates of a stem are L_{f0} and L_f for a kind of faulty chips. The bold red line is a division of left and right regions with $L_f<L_{f0}$ and $L_f>L_{f0}$ respectively. As presented in the figures, chips with high fault ratios all fall into the left region properly, indicating that this reordering scheme is more efficient for fault detection. By applying the reordered test sequence, the averaged number of TCK cycles for detecting faulty chip decreases by 52.1%, 56.2%, 53.7% and 47.6% for the four situations respectively, compared with the common test sequence grouped by test blocks.

4 Conclusions

An enhanced DfT architecture QS-PTAP has been proposed for the chiplet test in 3D-stacked chips, featuring chiplet quick selection, inter-die scan chain automatic configuration and IEEE 1149 compatibility. It makes modular testing of selected chiplets or IP-cores in the chiplet more flexible. Inter-die chiplet selection via QS-PTAP and intra-die IP-core selection via the IEEE 1687 SIB infrastructure are combined for efficient access to the test blocks. Moreover, a test vector reordering scheme has been proposed for 3D chip test, aiming at improving the faulty chip detection efficiency. Then a 3D-stacked chip is adopted to validate these designs. Simulation results show that the chiplet access and scan chain configuration are remarkably simplified by introducing the QS-PTAP architecture; and for the studied cases, the time for faulty chip detection is reduced by at least 45% when the reordering scheme is applied.

Acknowledgments

This work is funded by the open foundation of the RF Integration and Microassembly Technology National and Local Joint Engineering Laboratory under Grant No. KFJJ20230101, and open foundation of the ASIC System-Level Packaging Engineering Research Center of Anhui Province under Grant No. 2022GCZXZD03, 2022GCZXZD04 and 2022GCZXZD05.

We are grateful to the authors of Icarus verilog, GTKwave and GCC for their efforts on the free and open-source software.

References

[1] M. Hutner, R. Sethuram, B. Vinnakota, Special Session: Test Challenges in a Chiplet Marketplace, *IEEE 38th VLSI Test Symposium*, Apr. 2020, doi: 10.1109/VTS48691.2020.9107636.

[2] Y. Fkih, P. Vivet et al. 3D DFT Challenges And Solutions, *ISVLSI Conference*, Jul. 2015, doi: 10.1109/ISVLSI.2015.11.

[3] E.J. Marinissen, T. McLaurin, H.L. Jiao, IEEE Std P1838: DfT Standard under Development for 2.5D-/3D- and 5.5D-SICs, *21th IEEE European Test Symposium*, May. 2016, doi: 10.1109/ETS.2016.7519330.

[4] IEEE Std 1838™-2019, IEEE Standard for Test Access Architecture for Three-Dimensional Stacked Integrated Circuits, *IEEE Computer Society*, Nov. 2019, doi: 10.1109/IEEESTD.2020.9036129.

[5] IEEE Std 1149.1™-2013, IEEE Standard for Test Access Port and Boundary-Scan Architecture, *IEEE Computer Society*, May.

2013, doi: 10.1109/IEEESTD.2013.6515989.

[6] IEEE Std 1500™-2005, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits, *IEEE Computer Society*, Aug. 2005, doi: 10.1109/IEEESTD.2005.96465.

[7] S.K. Roy, C. Giri, H. Rahaman, Optimization of Test Wrapper for TSV Based 3D SOCs, *J.Electronic Testing*, 2016 32: 511-529, doi: 10.1007/s10836-016-5610-4.

[8] J. Durupt, P. Vivet, J. Schloeffel, IJTAG supported 3D DFT using Chiplet-Footprints for testing Multi-Chips Active Interposer System, *21st IEEE European Test Symposium*, May. 2016, doi: 10.1109/ETS.2016.7519310.

[9] IEEE Std 1687™-2014, "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," *IEEE Computer Society*, Dec. 2014, doi: 10.1109/IEEESTD.2014.6974961.

[10] A. Sinha, Y. Cho, J. Easter, M. Rojas, Multi-die Parallel Test Fabric for Scalability and Pattern Reusability, *2022 IEEE International Test Conference*, Nov. 2022, doi: 10.1109/ITC50671.2022.00033.

[11] A. Pandey, B. Tully, A. Samudra, A. Nagarandal, Novel Technique for Manufacturing & In-system Testing of Large Scale SoC using Functional Protocol Based High-Speed I/O, *2022 IEEE 40th VLSI Test Symposium*, May. 2022, doi: 10.1109/VTS52500.2021.9794234.

[12] Z.K. Cai, G.P. Zhou, J. Song, Z.X. Wang, Y.F. Guo, A Universal Test Access Port Controller Circuit Design for Chiplet Testing, *Journal of Electronics & Information Technology*, 2022, doi: 10.11999/JEIT220854.

[13] C.L. Fang, Q.C. Huang, R.D. Blanton, Adaptive Test Pattern Reordering for Diagnosis using k-Nearest Neighbors, *2020 IEEE International Test Conference in Asia*, Sup. 2020, doi: 10.1109/ITC-Asia51099.2020.00022.

[14] T. Song, H.G. Liang, T.M. Ni et al. Pattern Reorder for Test Cost Reduction Through Improved SVMRANK Algorithm, *IEEE Access* 8:147965-147972, Aug. 2020, doi: 10.1109/ACCESS.2020.3016039.

[15] W.F. Zhan, Z.W. Shao, Test patterns reordering method based on Gamma distribution, *Integration the VLSI Journal* 62:66-71, May. 2020, doi: doi.org/10.1016/j.vlsi.2019.12.008.

[16] S. Mitra, D. Das, A Stochastic Heuristic based Approach to Test Vector Reordering for Dynamic Test Power Reduction, *2016 2nd International Conference on Contemporary Computing and Informatics*, May 2017, doi: 10.1109/IC3I.2016.7918020.

Identifying Reliability High-Correlated Gates of Logic Circuits with Pearson Correlation Coefficient

Zhanhui Shi, Jie Xiao, Jianhui Jiang*, Ying Zhang, and Yuhao Zhou

***Abstract*—Identifying reliability high-correlated gates (HRCGs) is vital for fault location and exclusion, especially for cascading faults. By executing a linear fit based on the results of the circuit's reliability evaluation and calibrating the fit function using regression residual analysis, this study first proves the existence of HRCGs. A time-series-oriented PCC model is then introduced to quantify gates' reliability correlation (GRC) and identify all the HRCGs in the circuit. Circuit-correlated primary outputs and sequential circuit-correlated flip-flops were further identified based on this approach. Experimental results on benchmark circuits show that the average accuracy of this approach is 0.9972 with the Monte Carlo (MC) method, and it is 2591 times faster than the MC method. On larger circuits, the identification rate and stability are 2.73 times and 18 times greater than the reference method, respectively.**

***Index Terms*—Pearson correlation coefficient, high-correlated gates, gates' reliability correlation, correlated gates identification.**

I. Introduction

NANOCIRCUIT integration is constantly developing with the use of new materials and scaling technologies, while the reliability of circuits will unavoidably decrease due to extreme oscillations in static and dynamic parameters.

Reliability analysis provides improvement measures and the effects of improvements on system reliability by investigating the internal and external reasons that result in weak segments. Circuit reliability estimation techniques can broadly be divided into simulation-based and model analytical-based methods. The most prevalent simulation-based methods is the Monte Carlo (MC) method. Although this method's high accuracy is widely acknowledged in the industry, it is not suitable for large-scale circuits due to its time overhead. There are already many tools and algorithms for the model analytical-based methods, and some researchers have tried to improve these traditional techniques. For example, examining the effect of logical susceptibility and constancy of gates on PTM calculations [1]. A uniform and normal noise distribution is added to the PGM [2]. Utilizing various structure learning and transfer learning algorithms to determine the most appropriate BN model [3]. In addition, some researchers combined novel models to estimate circuit reliability, such as stochastic computation [4], hybrid estimation [5], and approximate arithmetic [6], to speed up circuit reliability evaluation.

However, these methods calculate the reliability of each gate by traversing from the primary inputs to the primary outputs. The front-to-back connection relationships of the gates are considered, while the correlation between non-adjacent gates is easily ignored. As a result, the accuracy of these methods is affected, especially as the circuit size increases. For instance, the PTM method considers the effects of specific gate failures repeatedly. This results in reliability values that become smaller and smaller and eventually approach zero in very large circuits.

In fact, there is a general correlation between gates in the circuit due to the abundance of fan-out branches and re-convergence events. Some distant gates even show significant correlations. Fig. 1 illustrates the reliability variation of benchmark circuit C17's six gates with 100 continuous input vectors. In terms of the general tendency, gates closer to the primary inputs are less influenced by faults in other gates. Conversely, gates closer to the primary outputs are more vulnerable to faults caused by correlated gates passing backward. In addition, gates at similar-level locations are also influenced by the association in a similar way (refer to the colors in Fig. 1).

Identifying reliability high-correlated gates (HRCGs) not only improves reliability assessment accuracy but also provides an effective reference basis for failure locating, particularly forcascade failures. It can also be excluded in order according to the gates' reliability correlation (GRC). Furthermore, for large-scale circuits with numerous primary

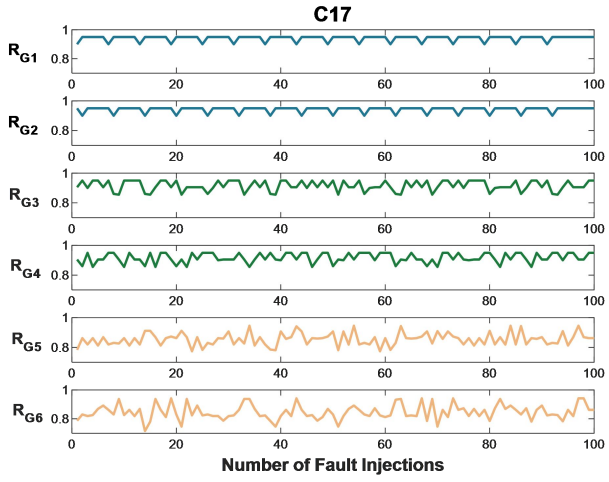


Fig. 1. The reliability of C17's six gates with 100 random fault injections.

This work is supported by the National Natural Science Foundation of China, Nos. 61974105 and 61972354 and the Natural Science Foundation of Zhejiang Province of China, No. LZ22F020011.

Zhanhui Shi, Jianhui Jiang, Ying Zhang and Yuhao Zhou are with the College of Software Engineering, Tongji University, Shanghai, China.(e-mails: 2010849@tongji.edu.cn, jhjiang@tongji.edu.cn, yingzhang@tongji.edu.cn, and zhouyuhao@tongji.edu.cn).

Jie Xiao is with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. (e-mails: xiaojiexqj@foxmail.com).

outputs, it is possible to precisely determine the individual reliability or overall reliability of the circuit after splitting or merging by calculating the GRC of the primary outputs.

Similarly, voltage correlation-based analysis [7], power correlation-based analysis [8], and time correlation-based analysis [9] at the circuit level and system level are similar in identifying HRCGs to assist in fault diagnosis. There are few studies to estimate gate correlation in terms of reliability and identify HRCGs. Therefore, we calculated the logic circuit's GRC using the Pearson correlation coefficient (PCC) and then identified all of the HRCGs in the circuit (PCC-HRCGI).

Our approach contributes to the following:

- We fitted the linear function and utilized logistic residual analysis to validate HRCGs existence based on the gates' reliability data;
- We introduced the PCC model to quantify the GRC and employed the F-significance test and correlation coefficient ρ test to further examine the correctness;
- This approach was used to identify HRCGs as well as high-correlated primary outputs and sequential circuits with high-correlated flip-flops.

II. Validate the Existence of HRCGs

A. Acquiring Gates' Reliability

In order to avoid pseudo-randomness and high repetition rates, we initially generated N uniform random input vectors based on UNBS [10]. To simulate more fault scenarios, we then randomly inject faults for all gates under each input vector. Since the SCA model [11] is an input vector-oriented reliability calculation method, the reliability values of all gates are then determined utilizing the SCA model with various input vectors. Finally, we counted the reliability value variations for each gate.

B. Fitting Data with Distinct Linearity

To demonstrate the GRC more intuitively, we conduct further statistics and analysis on the reliability dataset generated in the previous step. By comparing the reliability values of any two gates, the two basic gates with the most comparable trends are selected. As in Fig. 2(a)'s 74148 circuit, G_{14} and G_{31}. Fig. 2(a) shows that x=R_{G14} and y=R_{G31} are linear. Thus, the regression model was configured as follows:

$$\hat{y}=\widehat{\beta_0}+\widehat{\beta_1}x_1 \tag{1}$$

The regression coefficients in Equation (1) are $\widehat{\beta_0}$ and $\widehat{\beta_1}$, which are usually solved using the least squares method as in Equation (2).

$$\min Q(\beta_0,\beta_1)=\sum_{i=1}^{N}\left[Y_i-\left(\widehat{\beta_0}+\widehat{\beta_1}x_1\right)\right]^2 \tag{2}$$

Calculations reveal that R_{G14} and R_{G31} in Fig. 2 have correlation coefficients of 0.0113 and 0.9881, respectively. After the initial fit of R_{G14} and R_{G31}, the correlation function is R_{G31}=0.9881×R_{G14}+0.0113.

C. Validating the Reliability Fit Function

Taking into account that certain data are impacted by objective factors during the sampling process, such as fluctuations caused by the input vector with the worst reliability. Due to the regression residual analysis model could accurately identify outliers in the dataset. In this subsection, we use this model to verify and adjust the reliability fit function. Typically, outliers are assessed using standard residual values (r), which are calculated in Equation (3), where s is the standard deviation.

$$r_{i,\text{stand}}=\frac{y_i-\hat{y_i}}{s_{i,res}},s_{i,res}=s\sqrt{1-h_i},h_i=\frac{1}{N}+\frac{\left(x_i-\bar{x}\right)^2}{\sum_i\left(x_i-\bar{x}\right)^2} \tag{3}$$

Fig. 2(b)'s residual distribution reveals eight outliers, and their confidence intervals do not include zero. After excluding these outliers, use step 2 to obtain the most accurate relational function of R_{G14} and R_{G31} as: R_{G31}=0.9974×R_{G14}+0.0023. R_{G14} and R_{G31} are confirmed to be linearly correlated, hence G14 and G31 of 74148 are HRCGs.

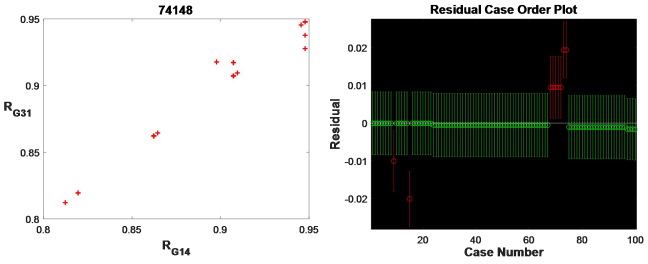


Fig. 2. Procedure for fitting and verifying the reliability correlation of 74148 circuit's G_{14} and G_{31}.

III. Identifying HRCGs

A. Reliability Analysis for basic gates

The PCC-HRCGI approach first creates N uniform and random input vectors to prevent high repetition rates and pseudo-randomness. To reflect variances in the way faulty gates affect other gates, the PCC-HRCGI method randomly injects N faults under N input vectors. In contrast to the traditional overall circuit reliability analysis, the GRC computation requires the probability of each gate's correct output from each fault injection. Lastly, method PCC-HRCGI computes and stores the reliability values of each gate under N fault injections using the SCA model.

B. PCC-based GRC Calculation

Although logistic regression analysis (LRA) can accurately estimate the linear connection between any two gates, its temporal complexity of $O(G^2)$ makes it inapplicable to large-scale circuits, where G is the number of basic gates in the circuit. Different from Chapter II's Part A, PCC-HRCGI employs the PCC model to quantify GRC. The PCC's superior computing performance and support for matrix calculation enable the PCC-HRCGI method to complete the GRC computation with the temporal complexity of $O(G)$.

The Pearson correlation coefficient, established by Pearson in 1895, measures the significance and direction of a correlation between variables [12]. The formula for deriving the PCC between two samples x_i and y is shown in Equation (4).

The range of PCC is [-1,1], the correlation grows stronger the larger the absolute value of r. While r=1 denotes a completely positive linear correlation, r=-1 means a completely negative

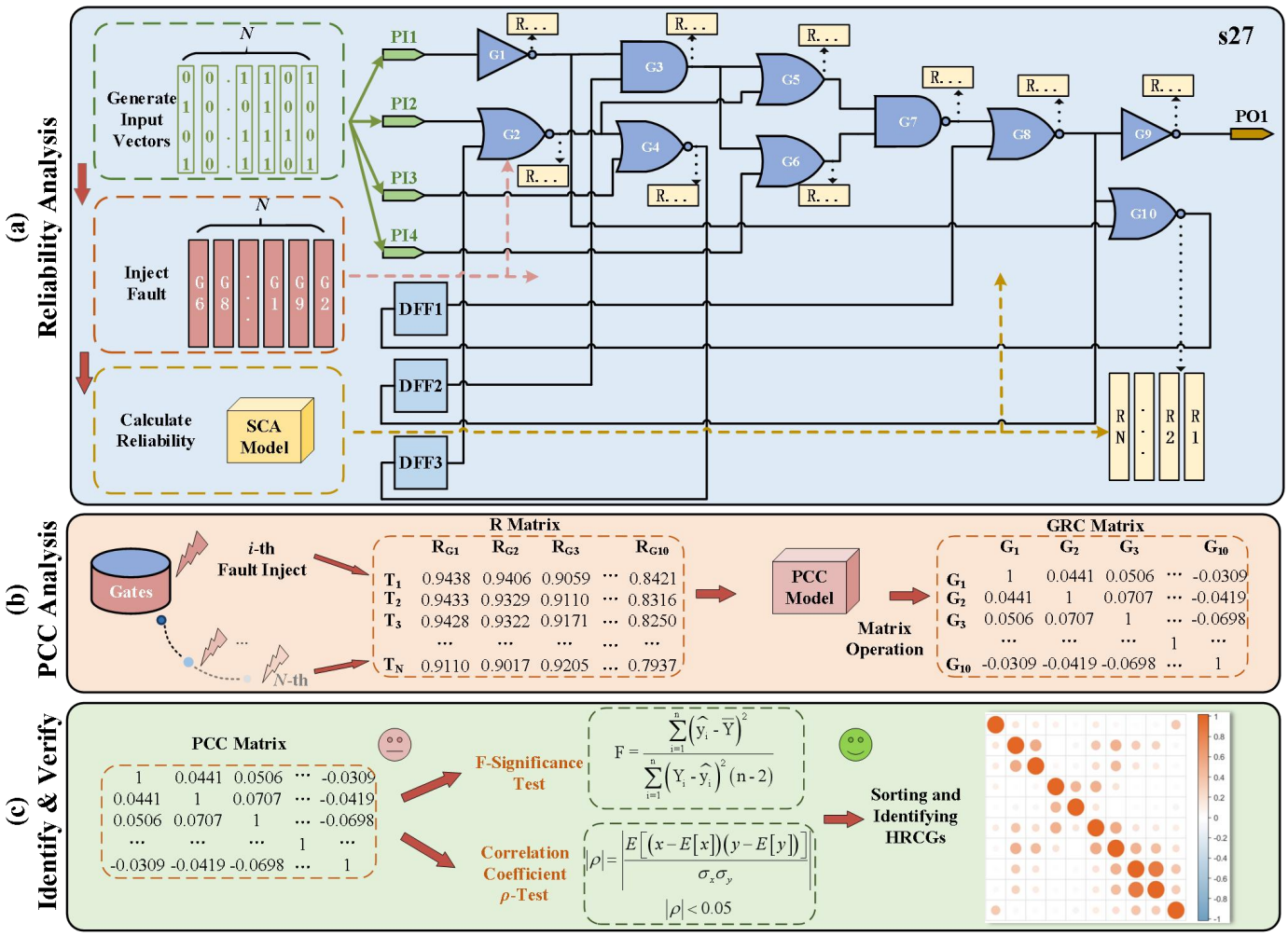


Fig. 3. An illustration of the PCC-HRCGI method on the s27 circuit.

linear correlation, and r=0, stands for no linear correlation.

$$r_{x,y} = \frac{\sum_{i=1}^{N}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2}\sqrt{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2}} \tag{4}$$

Because the fault injection sequence has a time-series relationship with each gate reliability, the PCC-HRCGI method employs time-series PCC to accelerate GRC calculation. Thus, PCC-HRCGI creates a reliability value matrix of size $N \times G$ (***R*** matrix, as in Fig. 3(b)). Each column in the ***R*** matrix represents the reliable value of the gate in a time series. The PCC-HRCGI method uses the overall correlation coefficient calculation formula, Equation (5):

$$\rho_{R_{Gi},R_G} = \frac{\mathrm{cov}\left(R_{Gi}, R_G\right)}{\sigma_{R_{Gi}}\sigma_{R_G}} \tag{5}$$

$\boldsymbol{R_{Gi}}$ and $\boldsymbol{R_G}$ are the time series of reliability values of the i-th gate and other gates. The standard deviation is represented by σ, and the function cov(x,y) is used to find covariance. Similar to r, ρ>0 means positive correlation and ρ<0 means negative correlation, with the difference that ρ<0.05 means significant correlation. The PCC-HRCGI method calculates GRC utilizing Equation (5) and generates a GRC matrix of size $G \times G$ based on the matrix operation of ***R***. ***GRC***(i,j) represents the correlation coefficient of gate i and gate j.

C. Identifying and Verifying HRCGs

Considering that not all gates are HRCGs, even though we have calculated the GRC of all gates, further identification of HRCGs from the ***GRC*** is required. It is common to check the correlation coefficient ρ following PCC calculation. The PCC-HRCGI method identifies all terms in the ***GRC*** that have a strong correlation ($|\rho|$<0.05). For instance, ***GRC***(1,10) in Fig. 3(c) shows that G_1 and G_{10} have a high reliability correlation degree.

Taking into account the possibility that the dataset contains outliers that cause inaccurate HRCGs identification, the F-significance check is widely used as a significance test. Usually when the test parameter α = 0.05 if F > 2 then the correlation is significant. The dataset of all identified HRCGs will also undergo an F-significance check by the PCC-HRCGI method. Equation (6) calculated the significance value F.

$$F = \frac{\sum_{i=1}^{n}\left(\hat{y}_i - \bar{Y}\right)^2}{\sum_{i=1}^{n}\left(Y_i - \hat{y}_i\right)^2 (n-2)} \tag{6}$$

The PCC-HRCGI method returns the final HRCGs after finishing identification and verification. In addition, to depict the strength of gate correlation, we display the gate correlation in the circuit using a heat map. As shown in Fig. 3 (c), there are 9 HRCGs with strong correlation, 12 with medium correlation, and the rest with weak or no correlation in the s27 circuit.

TABLE I
PERFORMANCE OF METHOD PCC-HRCGI ON DIFFERENT CIRCUITS

Circuits		Gates	PCC-HRCGI								MC-based LRA			
			1st-HRCGs	Time	*r*	*p*	*Acc*	*Ir*	*St*	$F_{.05}$	1st-HRCGs	β_0	β_1	Time
Combinational Logic Circuits														
74-serial	74148	31	G22&G26	0.15s	0.9725	0.0004	1	1	0.001	>2	G22&G26	0.0419	0.9532	0.78s
	74185	33	G9&G21	0.16s	0.8813	0.0015	1	1	0.001	>2	G9&G21	0.1072	0.8841	0.92s
	74283	36	G27&G36	0.19s	0.9343	0.0009	1	1	0.001	>2	G27&G36	0.1460	0.9130	1.34s
ISCAS'85	C432	160	G86&G97	1.05s	0.9997	0.0002	1	1	0.002	>2	G86&G97	0.0506	0.9890	23.12s
	C5315	2,307	G121&G456	9.16s	1	0	1	1	0.003	>2	G121&G456	1.49e-13	1	3,530.71s
	C7552	3,512	G48&G173	11.47s	1	0	1	1	0.003	>2	G48&G173	7.91e-14	1	7,090.42s
EPFL	sin_sin23	11,247	G14&G1803	126.21s	1	0	0.97	1	0.005	>2	G97&G127	0.0500	0.9000	440,711.53s
	square_70	24,136	G41&G46	491.22s	1	0	null	1	0.007	>2	null	3.04e-11	0.9946	null
	voter_maj	34,103	G33945&G34090	484.90s	1	0	null	0.98	0.012	>2	null	6.17e-13	1	null
Sequential Logic Circuits														
ISCAS'89	s298	119	G103&G104	2.01s	0.9009	1.01e-73	1	1	0.002	>2	G103&G104	0.0370	0.9176	109.95s
	s713	393	G274&G284	5.40s	0.9156	1.20e-80	1	1	0.002	>2	G274&G284	0.0288	0.9029	430.47s
	s1238	508	G311&G318	6.54s	0.8850	1.16e-67	1	1	0.003	>2	G311&G318	0.0342	0.9301	480.82s
ITC'99	b05	927	G400&G402	12.66s	0.9169	6.73e-81	1	1	0.002	>2	G400&G402	0.0468	0.9036	1,190.11s
	b20	19,682	G12773&G12750	223.94s	0.9829	1.09e-14	null	1	0.011	>2	null	0.0155	0.9407	null
	b21	20,027	G1247&G1342	302.16s	0.9756	2.68e-13	null	0.99	0.014	>2	null	0.0205	0.9391	null

IV. SIMULATING

We performed simulation experiments on benchmark circuits to demonstrate the advantages of the PCC-HRCGI method in terms of performance, efficiency, and applicability. To facilitate comparison and validation between different approaches, we employed benchmark circuits widely used in [6], [10], [11], and [12]. All experiments were conducted on a personal PC with a Core i5-7550 processor (3.4 GHz) and 32GB of RAM.

A. Evaluation Metrics

1) **Accuracy**

Considering that the HRCGs identified by different methods may have close GRC but completely different specific gates, we construct accuracy metrics mainly based on the GRC value, as in Equation (7).

$$Acc_i = \frac{\overline{GRC_i}}{\overline{GRC_{MC}}} = \frac{\sum_{k=1}^{N} GRC_{i,k} / N}{\sum_{k=1}^{N} GRC_{MC,k} / N} \tag{7}$$

1) **Identification rate**

As the number of HRCGs that a method can identify is an essential metric of its efficiency, Equation (8) measures the size of the MC-supported circuits, while Equation (9) measures larger-scale circuits.

$$Ir_i = \frac{\sum HRCGs_i}{\sum HRCGs_{MC}} \tag{8} \qquad Ir_i = \frac{\sum HRCGs_i}{\sum HRCGs_i + \sum HRCGs_j} \tag{9}$$

2) **Stability**

In light of the fact that experimental data are influenced by other factors, Equation (10) measures identification of HRCGs on the same circuit with T times.

$$St_i = \sqrt{\frac{\sum_{t=1}^{T} \left(GRC_{i,t} - \overline{GRC_i} \right)^2}{T}} \tag{10}$$

B. Performance Analysis

On combinational and sequential benchmark circuits of various sizes, we examined the PCC-HRCGI method's performance. As shown in Table 1, compared to the MC method, the average accuracy of the PCC-HRCGI method is about 0.9972, and it is 2591 times faster than MC on average. The standard deviation and identification rate on average are 99.8% and 0.0046, respectively. From the fitting coefficients of the MC method and the F-validation results, the identified top HRCGs are fully linearly correlated.

In addition, we selected the Rand-based LRA method as a reference for larger-scale circuits due to the MC method's limitations. Fig. 4 displays the four evaluation metrics of the two methods for identifying HRCGs on various types of circuits. As can be observed, the PCC-HRCGI method is superior to the Rand-LRA method in terms of *Acc*, *Ir*, and *St*, and its T (T=log$_{10}t$) is shorter by two orders of magnitude.

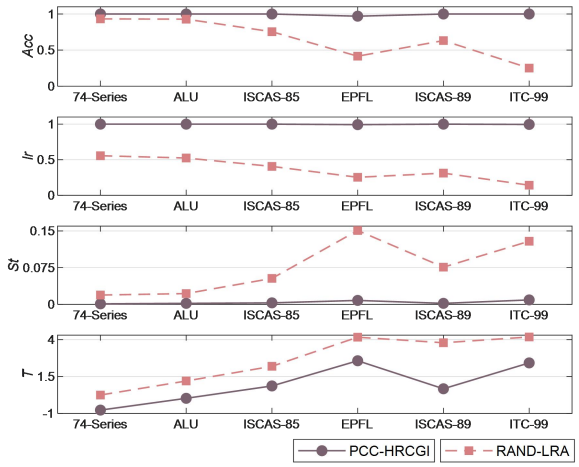


Fig. 4. Comparison of *Acc*, *Ir*, *St*, and T corresponding to HRCGs identified by different methods.

C. Efficiency Analysis

As shown in Fig. 5, we counted the number of HRCGs identified by the method PCC-HRCGI on six distinct types of benchmark circuits to validate the method's efficiency. For moderately correlated RCGs (0.6<GRC<0.8), we use yellow bars, and for highly correlated RCGs (0.8<GRC<1), red bars. Fig. 5 demonstrates that as circuit size increases, the method PCC-HRCGI identifies exponentially more HRCGs on combinational and sequential circuits. The EPFL and ITC'99 benchmark circuits, in particular, contain more than 10^6 and

10^7 identified HRCGs, respectively. It is evident that there is a regular correlation among gates, and the PCC-HRCGI method is quite efficient within 10^4 gates circuits.

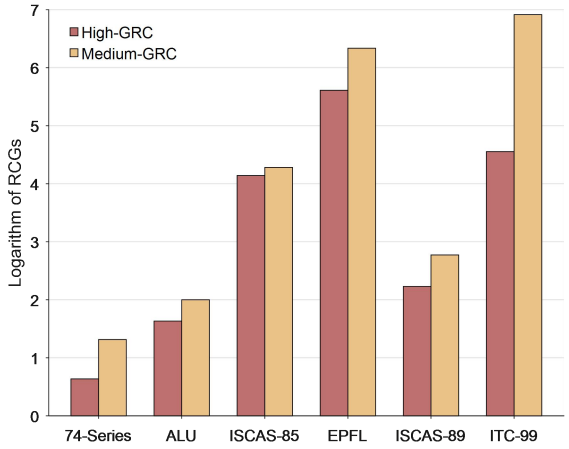


Fig. 5. The number of two types of RCGs identified by the PCC-HRCGI method on circuits of various sizes.

D. Applicability Analysis

Based on the PCC-HRCGI method, the GRC between different gates can be accurately quantified, and highly correlated HRCGs can be quickly and efficiently identified, as shown in Fig. 6. This approach could provide an effective reference basis for fault location and is especially advantageous for cascading faults.

From Fig. 7, the PCC-HRCGI method can be used to find HRCGs as well as correlations between the primary outputs or flip-flops to increase the precision of circuit reliability analysis and provide an explanation for dividing and merging multiple output circuits. Additionally, for very large-scale integration circuits, this approach can effectively calculate the correlation coefficients of any two gates, even if the identification of all HRCGs is constrained.

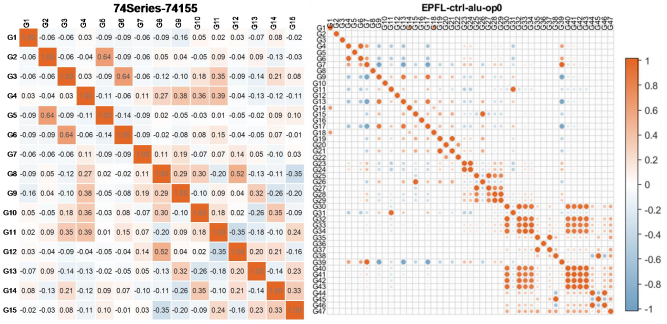


Fig. 6. GRC quantification and HRCGs identification.

V. Conclusion

In this brief, we first fit the reliability evaluation results of the circuit's gates to create the linear correlation function of HRCGs. We then used regression residual analysis to calibrate the function and demonstrate HRCGs' existence. Then, we applied a time-series-based PCC model to quantify the GRC and identified all the HRCGs in the circuit by matrix operations, called PCC-HRCGI. Finally, we applied it to identify circuit-correlated primary outputs and sequential circuits' correlated flip-flops. Experimental results on benchmark circuits show that the average accuracy of the PCC-HRCGI method is about 0.99 with the MC-based LRA approach, and the time is roughly 2591 times faster than the MC method. On larger circuits, the identification rate and stability are 2.73 times and 18 times superior to the Rand-based LRA method.

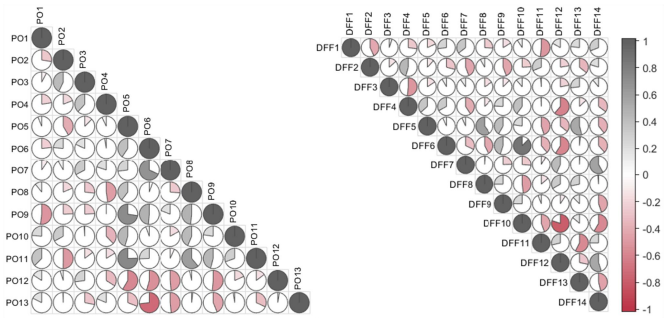


Fig. 7. Identification of correlated primary outputs (PO) and D-flip-flops with the PCC-HRCGI method on s1238 and s298.

References

[1] M. F. Pontes, I. F. V. Oliveira, et al, "The Impact of Logic Gates Susceptibility in Overall Circuit Reliability Analysis," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Austin, TX, USA, 2022, pp. 1610-1614.

[2] M. A. K. Abuelala, A. Wassal, A. Khattab and H. A. H. Fahmy, "A Novel Model for Injecting Error in Probabilistic Gates," *31st International Conference on Microelectronics (ICM)*, Cairo, Egypt, 2019, pp. 276-279.

[3] R. Pan, Z. Zhang, X. Li, K. Chakrabarty and X. Gu, "Black-Box Test-Cost Reduction Based on Bayesian Network Models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 386-399, Feb. 2021.

[4] J. Xiao, Q. Ji, Q. Shen, J Jiang, et al, "Accelerating stochastic-based reliability estimation for combinational circuits at RTL using GPU parallel computing," *International Journal of Intelligent Systems*, vol. 37, no. 11, pp. 8309-8326, June. 2022, doi: 10.1002/int.22940.

[5] C. Chen and S. Zhan, "A Hybrid Method for Signal Probability Estimation with Combinational Circuits," *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Shenzhen, China, 2022, pp. 472-475.

[6] Z. Wang, G. Zhang, P. Liu, J. Ye and J. Jiang, "Accurate Reliability Boundary Evaluation of Approximate Arithmetic Circuit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 10, pp. 1507-1518, Oct. 2022, doi: 10.1109/TVLSI.2022.3193897.

[7] G. Wang, J. Yang and J. Jiao, "Voltage Correlation-Based Principal Component Analysis Method for Short Circuit Fault Diagnosis of Series Battery Pack," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 9, pp. 9025-9034, Sept. 2023, doi: 10.1109/TIE.2022.3210588.

[8] H. Huang, L. Liu, Q. Huang, Y. Chen, S. Yin and S. Wei, "Low Area-Overhead Low-Entropy Masking Scheme (LEMS) Against Correlation Power Analysis Attack," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 208-219, Feb. 2019, doi: 10.1109/TCAD.2018.2802867.

[9] M. Ni, X. Wang, F. Li, W. Rhee and Z. Wang, "A 13-Bit 2-GS/s Time-Interleaved ADC With Improved Correlation-Based Timing Skew Calibration Strategy," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 2, pp. 481-494, Feb. 2022.

[10] J. Xiao, Z. Shi, X. Yang and J. Lou, "BM-RCGL: Benchmarking Approach for Localization of Reliability-Critical Gates in Combinational Logic Blocks," *IEEE Transactions on Computers*, vol. 71, no. 5, pp. 1063-1076, 1 May 2022, doi: 10.1109/TC.2021.3071253.

[11] J. Xiao, W. Chen, J. Lou, J. Jiang and Q. Zhou, "Identifying Reliability-Critical Primary Inputs of Combinational Circuits Based on the Model of Gate-Sensitive Attributes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4708-4720, Nov. 2022, doi: 10.1109/TCAD.2022.3142194.

[12] K. Pearson, "Contributions to the Mathematical Theory of Evolution," *Philosophical Transactions of the Royal Society of London*. vol. 185, pp. 71–110, 1894, doi: org/90667.

Zhanhui Shi received his MS degree in computer science and technology from Zhejiang University of Technology, China, in 2020. He is currently a Ph.D. student in software engineering at Tongji University, Shanghai, China. His current research interests include reliability evaluation and fault-tolerant design and combinatorial optimization computation. He has published 4 technical papers in refereed international journals such as IEEE TC and JCST.

Jie Xiao received his Ph.D. degree in computer system architecture from Tongji University, Shanghai, China, in 2013. He is currently an associate professor with the department of computer science and technology, at Zhejiang University of Technology, Hangzhou, China. His current research interests include reliability evaluation, and fault-tolerant design, AI Security Testing and Assessment, Combinatorial Optimization-Computation, Machine Learning. He has co-authored one book and published more than 50 technical papers in refereed international journals including IEEE TR, IEEE TC, and IEEE T-CAD.

Jianhui Jiang received his BE, ME and PhD degrees in traffic information engineering and control from Shanghai Tiedao University (in April 2000, it was merged to Tongji University) in 1985, 1988, and 1999, respectively. From 1994-2000, he was an Associate Professor of Computer Science and Technology at Shanghai Tiedao University. Since 2000, he has been a full professor of Computer Science and Technology at Tongji University. From 2007-2011, he was chair of the Department of Computer Science and Technology at Tongji University. Since 2011, he is Associate Dean of the School of Software Engineering at Tongji University. He is Vice Director of Technical Committee on Fault-tolerant Computing, Chinese Computer Federation (CCF). He has served on several program committees of national or international symposiums or workshops including IEEE Pacific Rim International Symposium on Dependable Computing, IEEE Asian Test Symposium, IEEE Workshop on RTL and High-Level Testing. He has coauthored two books and published more than 200 technical papers. His current research interests include dependable systems and net-works, software reliability engineering, VLSI/SoC testing and fault-tolerance. He is a senior member of CCF.

Ying Zhang received the B.S. degree in computer science from Harbin Engineering University, Harbin, China, in 2006, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2011. He completed one-year postdoc in the Embedded Systems Laboratory, Linkoping University, Sweden. He also worked as a Visiting Scholar with the ECE Department, Duke University, USA, in 2016. He is currently an Associate Professor with the School of Software Engineering, Tongji University. He has served on several program committees of international conferences and symposiums, including IEEE Asia and South Pacific Design Automation Conference and IEEE Asian Test Symposium. His research interests include signal integrity, reliable design of network-on-chip,and wireless sensor networks. He is a member of CCF.

Yuhao Zhou is a PhD candidate in the School of Software Engineering, Tongji University, China. He has authored some academic articles which have been published in refereed international journals, such as IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, ACM Transactions on Design Automation of Electronic Systems, Applied Soft Computing, Chinese Journal of Electronic, etc. His research interests include Reed-Muller circuits area optimization based on XNOR/OR, power optimization, low power integrated circuits design, computer aided design and intelligent optimization algorithm. He is a member of ACM.

Fine-grained Pruning for RRAM-based Accelerator with Redundant Weight Layer-wise Searching

Bo Zhou, Hui Chen*, Peng Liu, Jigang Wu
School of Computer Science and Technology
Guangdong University of Technology
Guangzhou, China
chenhui02@gdut.edu.cn

Abstract— Eliminating redundancy to reduce ineffectual computation is a common approach to achieving energy and computation efficiency for accelerating neural networks. The Weight pruning is an effective model compression method by deleting redundant weights. However, existing unstructured pruning methods do not consider the RRAM crossbar structure and the structured pruning methods have a great impact on the network accuracy because of the coarse granularity. In this paper, a fine-grained pruning method is proposed, which can effectively reduce the hardware overhead of an RRAM-based accelerator. It limits the amount of pruned weight for each column in submatrices to match the crossbar structure. Furthermore, in order to achieve the best compression rate, a pruning optimization algorithm is proposed. A layer-by-layer searching method is designed to find the optimal pruning rate in each layer according to the redundant weight distribution in different layers. Considering there are correlations between the adjacent layers, a global strategy is proposed for pruning and fine-tuning. We evaluate the method by using two datasets of CIFAR10 and MNIST and three CNN models of AlexNet, VGG16, and Plain20, and achieve a compression rate of 12∼24x and an energy efficiency of 10∼24x.

Index Terms—RRAM, crossbar, CNN, pruning

I. INTRODUCTION

In recent years, with the rapid development of artificial intelligence, deep neural networks are widely used in various fields, such as natural language processing [1], computer vision [2], object detection [3] and so on. Even in many areas, Convolution Neural Network (CNN) models achieve better accuracy than humans. However, CNNs also bring high power consumption.

Many dedicated neural network hardware accelerators have been proposed to solve the problem of high calculation of neural networks. The RRAM-based accelerator has attracted much attention due to its unique advantages in fast matrix calculation and Processing-In-Memory(PIM) structure. The main structure of RRAM is the crossbar composed of memristors. RRAM crossbar structure can perform arithmetic operations as well as data storage [4]. This structure can achieve high-performance speedup and energy efficiency because it can perform matrix-vector multiplication with a high degree of parallelism and high efficiency.

*Corresponding author: Hui Chen

However, the RRAM-based structure has an inherent weakness, which needs too many crossbars to store the whole CNN weights in such a PIM architecture [5]. Compression techniques have been proposed to solve this problem, including pruning [4]–[10], quantization [7], [10]–[12] and matrix factorization [13]. CNN pruning schemes are effective methods for compressing models, where the conventional unstructured pruning results in zero weights randomly distributed in the weight, which cannot significantly improve performance for the RRAM-based accelerator [14]. The structured pruning methods are more hardware-friendly because of the structurally pruned weight. But the structured pruning method is easier to delete the important weights, affecting the inference accuracy of the CNN model.

In this paper, we propose a fine-grained pruning method to overcome the shortcomings of unstructured and structured pruning, which prunes a fixed number of weights in each column of the submatrix, and then screens out the specified number of pruning shapes to ensure that a complete matrix can still be formed after pruning. Our contributions can be summarized as follows:

- A fine-grained pruning method is proposed to adapt the crossbar structure in fine granularity. In this method, a fixed number of weights in each column are pruned, and the pruning shapes can be combined into a complete matrix. This method can be applied to the convolution layer and the fully connected layer and effectively reduce the crossbar area.
- A pruning optimization algorithm is proposed. Considering the distribution of weight redundancy in CNN, we propose a layer-by-layer searching method to find the optimal pruning rate in each layer. And according to the association information between layers in CNN, a global pruning strategy is proposed based on the Alternating Direction Method of Multipliers (ADMM) after layer-by-layer searching. The algorithm can maximize the pruning rate while ensuring high accuracy.
- We evaluate the method with three CNN models including AlexNet, VGG16, and Plain20 on two datasets of CIFAR10 and MNIST and achieve a compression rate of 12∼24x and an energy efficiency

of 10∼24x.

The rest of this paper is organized as follows. In Section II, we briefly introduce the related work about CNN weight mapping in RRAM-based accelerators, and pruning methods. Section III presents the main contributions of the article, including the fine-grained pruning method and the Pruning Optimization Algorithm. The evaluations of the proposed method are presented in Section IV, and Section V concludes this paper.

II. BACKGROUND

A. Mapping Weight in RRAM Crossbar

Memristors have attracted extensive attention in recent years because of their characteristics of non-volatility, low power consumption, high integration density, and high scalability. Memristors have the property of variable conductance that can be used to store different values [15]. Generally, each vertical bit line (BL) is connected to each horizontal word line (WL) through a memristor in a crossbar array. [8]. The architecture of RRAM-based CNN accelerators consists of crossbar arrays and peripheral components. When a convolution layer is mapped in the crossbar, 1D column vectors transformed from 3D filters are mapped into the crossbar columns, and the input activations corresponding to the sliding windows are also reshaped into the column vectors [16]. The multiple filters are mapped into different columns, as shown in Fig. 1. Similar to the convolution layer, the fully connected layer can also be mapped into the crossbar in the same way. In the calculation process, analog voltage inputs converted by digital to analog converter(DAC) are multiplied by Ohm's law with the weights which are mapped as the conductance value. Current signals are obtained and accumulated from each BL by Kirchhoff's law and then converted to the output result of the digital signal by analog to digital converter(ADC) in the peripheral circuit. This method reduces the time complexity of matrix calculation to O(1) and significantly improves the computational efficiency [10]. However, due to the limitations of peripheral circuits such as ADC and DAC, only a limited number of word lines and bit lines can be activated in one cycle. Therefore, it takes multiple cycles to complete matrix calculation in non-ideal conditions [17].

B. Weight Pruning on Crossbar Structure

Weight pruning is a method to compress the neural network model by removing the redundant weight, which can be divided into unstructured pruning and structured pruning. The unstructured pruning method using a single weight as granularity removes specific weights in the filter, as shown in Fig. 2(a). However, such an unstructured pruning method causes the unstructured sparsity in the network, and the RRAM-based accelerator scheme is difficult to use the unstructured sparsity [9]. The crossbar

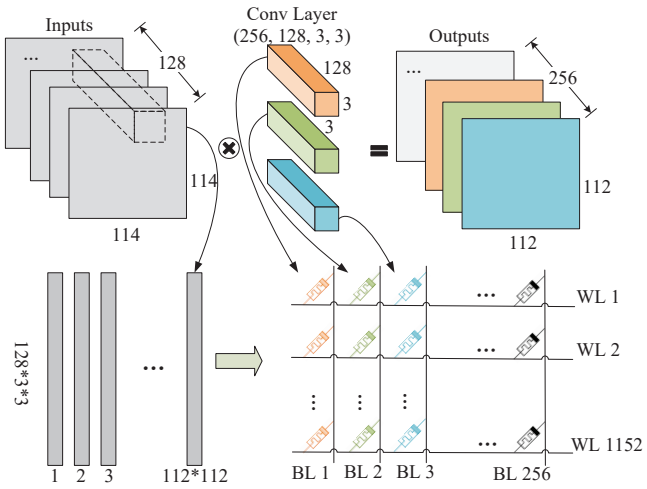
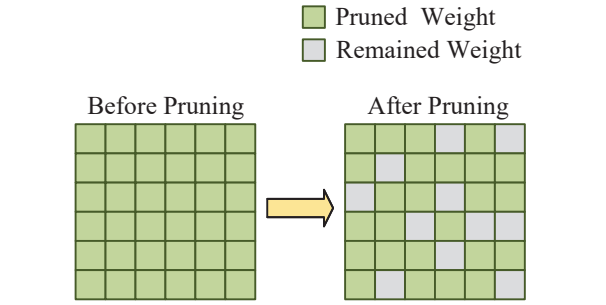


Fig. 1: RRAM Crossbar Mapping

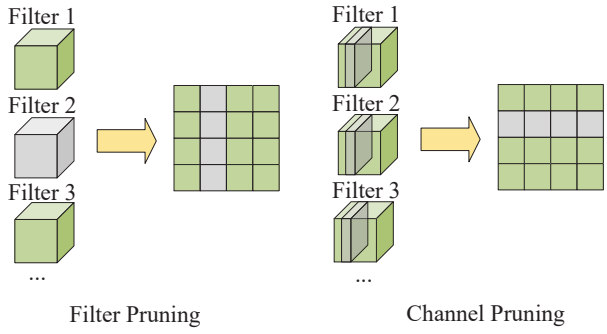
has a tightly coupled structure that the memristors in a WL share the same input and the dot product within the same BL should be summed. Due to the tightly coupled structure of the RRAM crossbar, the single zero-value weight cannot be skipped [5].

The structured pruning method deletes the entire convolution filter, which is fit with the crossbar structure, as shown in Fig. 2(b). By removing unimportant filters, a compact network with less overhead can be obtained. It is equivalent to deleting an entire column of the crossbar array in the RRAM-based accelerators. Similarly, deleting channels at the same position of each filter in the convolution layers corresponds to the deletion of elements in the same row of the crossbar. However, the structured pruning method has an inevitable disadvantage in that all the rows and columns in a crossbar are coupled together and there is no guarantee that all the weights in an entire row or column are unimportant. It is inevitable that some important weights will be pruned when the non-important weights are pruned. That means the model performance is more damaged than unstructured pruning. To solve this problem, Siling Yang et al. proposed an automated DNN pruning and mapping framework named AUTO-PRUNE in a finer granularity based on the OU mechanism [8]. It prunes on a finer granularity of column-vector, which makes better trade-offs between compression rate and accuracy. Yu et al. used a new pruning algorithm named pattern pruning in which the pattern is defined as a boolean mask that indicates whether the weights are nonzero in each position [4]. It can be considered as an intermediate type between unstructured and structured pruning, but it can only be used in the convolution layer.

Identifying redundant weights is a crucial aspect of the CNN pruning. The redundancy varies across different layers, with deeper layers generally exhibiting higher redundancy than shallower ones. Meanwhile, the number of weights in deeper layers is larger than that in shallower ones, and it is easier to achieve a higher pruning rate by prioritizing the layers with more weights for pruning



(a) Unstructured Pruning in Crossbar



(b) Structured Pruning in Crossbar

Fig. 2: The Pruning Methods in Crossbar

[18]. It indicates that layer-by-layer pruning would be a good pruning search strategy. Furthermore, in CNNs, the weight information between layers is interdependent. Ma et al. purposed a concept named Unused Data Path that if a whole filter is pruned, the generated feature maps will be "blank"(all zeros). And if the "blank" filters are input to the next layer, the corresponding weights in the next layer will become useless [6]. This emphasizes the need to consider inter-layer information to avoid accuracy degradation.

C. Basics of ADMM

The ADMM is a powerful optimization technique that has gained considerable popularity in the field of machine learning due to its ability to rapidly and effectively converge. In particular, the ADMM is well-suited to solving non-convex optimization problems such as the pruning problems [19]. This approach works by converting the original problem into two subproblems, and then solving them iteratively until convergence is achieved. By leveraging this decomposition strategy, the ADMM provides an efficient means of tackling complex, nonlinear optimization challenges. The general form of the ADMM function is

$$\text{minimize}_{\{x,z\}}\, f(x) + g(z) \qquad \text{s.t. } Ax + Bz = C, \tag{1}$$

where A, B, and C are constant. Through the formation of the augmented Lagrangian, the ADMM decomposes the problem into two subproblems and solves them iteratively until convergence:

$$\begin{gathered} L(x,z,\lambda) = f(x) + g(z) + \lambda^T(W) + \rho/2\,\|W\|^2, \\ where \quad W = Ax + Bz - C, \end{gathered} \tag{2}$$

The iterative solution formulas are as follows:

$$step1 : x^{k+1} = \text{argmin}_x\, L\left(x, z^k, \lambda^k\right), \tag{3}$$

$$step2 : z^{k+1} = \text{argmin}_z\, L\left(x^{k+1}, z, \lambda^k\right), \tag{4}$$

$$step3 : \lambda^{k+1} = \lambda^k + \rho\left(Ax + Bz - C\right). \tag{5}$$

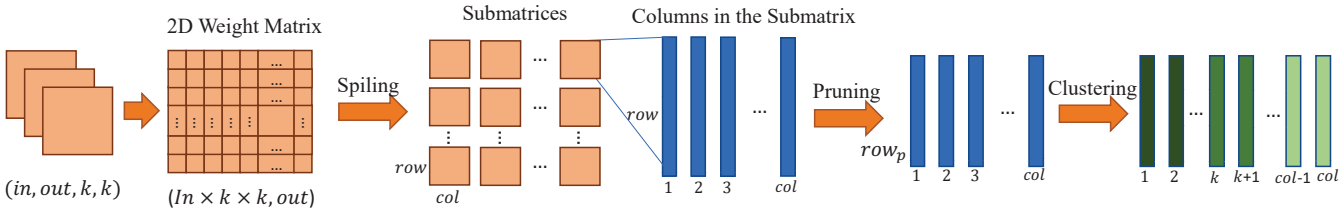
The ADMM applied in neural networks can be regarded as the use of an intelligent dynamic L2 regularization method, whose regularization targets change judiciously and analytically in each iteration and performs better than the traditional fixed regularization method [19].

III. CONSTRUCTION

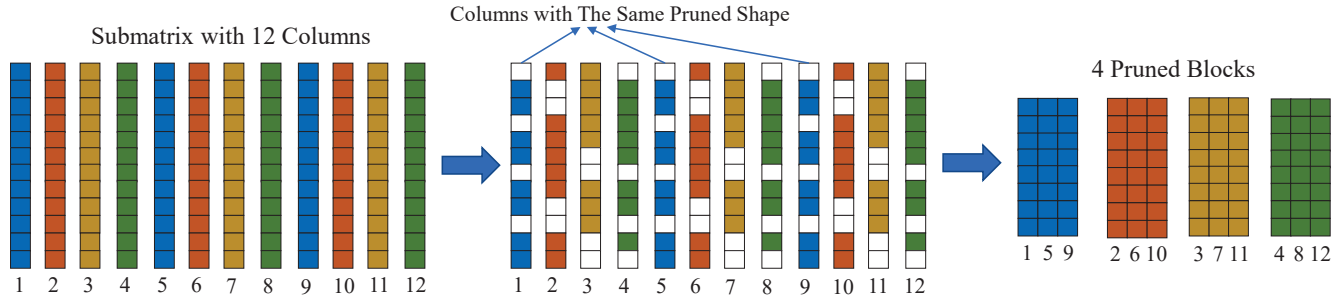
In this section, a fine-grained pruning method is introduced, which can prune both convolution and fully connected layers. By pruning a fixed number of weights in each column of the weight submatrices and limiting the mask shape of each column, the newly compact weight matrix with the smaller size can be well-mapped into the RRAM crossbars. The weight pruning problem is a difficult non-convex optimization problem, and finding the position and the number of pruned weights is not a simple problem. To obtain the optimal pruning rate, a pruning optimization algorithm based on the ADMM is developed to search for the optimal pruning rate of a single layer from the deep to the shallow of CNN, and then prune the weights globally.

A. Fine-Grained Pruning Method

There are three steps in the proposed fine-grained pruning method. First, the weight matrices are split into several submatrices. These submatrices are the basic units for pruning, and the pruning operation will be performed on each column of the submatrix. Second, in the submatrices, each column vector is pruned by deleting the same number of weights. This step is crucial for the pruning method, through which the size of the submatrix will be reduced. Finally, the columns with the same pruned shape are clustered as a block, and they are mapped into the adjacent position in the crossbar. Through this step, the weight columns in each block still share the same input and can complete matrix operations within a single cycle. In the example of Fig. 3(a), the weight matrix is split into several submatrices with the size of $row \times col$, and each column in the submatrices after pruning remains row_p weights. The pruning rate can be obtained as $1 - row_p/row$. The



(a) Process of Splitting and Pruning Weight Matrix



(b) The Method for Pruning Submatrix by Columns Pruned Shapes

Fig. 3: The Fine-Grained Pruning Method

three colors represent the three clustering results. the number of pruned weights after pruning(the value of row_p) is obtained by the layer-by-layer searching method to be discussed later. During the submatrix pruning step, the number of remaining weights, and the pruning shape remain constant throughout the process. The same pruned shape indicates that the positions of the pruned weights in the column vectors are the same. In each column vector with the same color, weights that are removed occupy the same positions, as depicted in Fig. 3(b). After the pruning, the column vectors with the same pruned shape are clustered together and mapped to the adjacent positions of the crossbar.

In the proposed pruning method, the greedy strategy is used to select the optimal pruned shapes for the column vectors, as shown in Algorithm 1. Firstly, the specified number of weights is removed from each column without considering the pruned shape. Secondly, the corresponding column vector masks are generated according to the column vector processed in the previous step. The masks of the positions corresponding to the pruned weights are set as 0, and the others are set as 1. Thirdly, the column vector masks are sorted by the occurrence number, and the num_p weight masks with the most occurrence times are selected by the defined hyperparameters. After pruning and clustering the column vectors, each cluster's result will correspond to a pruned shape. Finally, every column vector will be compared with each mask shape, and the mask that incurs the minimum loss will be chosen as the pruning pattern for that particular column.

Algorithm 1: Pseudo Code for Getting the Masks

Data: mtx: Weight submatrix, col_p: The remaining weight of each column after pruning, num_p: Number of mask shapes in the column vector;
Result: $mask$

```
mtx_p ← Each column in the mtx after cutting with a num_p weight matrix;
mtx_m ← mtx_p corresponding mask matrix;
Arrange the column vectors in mtx in descending order of occurrence as list_m;
list_m ← list_m[0:lnum_p];
masks ← [ ];
for col in mtx do
    Add the mask vector with the smallest col loss in list_m to masks;
end
```

B. Pruning Optimization Algorithm

A pruning optimization algorithm is proposed to search the optimal pruning rate and redundant weights, which is based on the ADMM and includes two parts: layer-by-layer searching and the global pruning strategy.

1) ADMM-based Optimization Algorithm: In the proposed ADMM-based algorithm, the pruning problem is described as

$$\text{minimize}_{W_i} \; f\left(\{W_i\}_{i=1}^{N}\right) \quad \text{s.t. } W_i \in S_i, \tag{6}$$

where f represents the loss function, and $S_i = \{W_i \mid pruned(W_i) = l_i\}$, indicating that the number of pruned weights in each column in W_i should be equal to l_i. Formula (6) represents the minimum value of the loss function under the pruning constraint.

For the constraint set S_i, we define the indicator

function as

$$g_i\left(\{W_i\}_{i=1}^N\right) = \begin{cases} 0 & \text{if pruned}\,(W_i) = l_i \\ +\infty & \text{otherwise} \end{cases}. \quad (7)$$

Further, auxiliary variables $Z_i, i = 1, ..., M$ are introduced which satisfies $Z_i = W_i$, and combine it Formula (6), then we can get the general form o ADMM function that

$$\text{minimize}_{\{W_i\}}\, f\left(\{W_i\}_{i=1}^N\right) + \sum_{i=1}^{M} g_i\left(Z_i\right) \quad \text{s.t. } W_i =$$

where the optimal solution problem with two indepe variables satisfies the equality constraint. The AI solves the problem by decomposing the main pro into two subproblems and solving them iteratively b augmented Lagrangian function until the convergen completed. The two subproblems decomposed by For (8) are

$$\text{minimize}_{\{W_i\}}\, f\left(\{W_i\}_{i=1}^N\right) + \sum_{i=1}^{M} \frac{\rho_i}{2} \left\|W_i + Z_i^k - U_i^k\right\|_F^2, \quad (9)$$

$$\text{minimize}_{\{Z_i\}} \sum_{i=1}^{M} g_i\left(Z_i\right) + \sum_{i=1}^{M} \frac{\rho_i}{2} \left\|W_i + Z_i^k - U_i^k\right\|_F^2, \quad (10)$$

where $U_i^k = U_i^{k-1} + W_i^k - Z_i^k$, and k is the number of iterations. The second term is the quadratic regularization term of W_i, which is differentiable and convex. The stochastic gradient descent (SGD) method can solve the two subproblems. For $i = 1, ..., M$, the optimal solution is

$$Z_i^{k+1} = \prod_{S_i} \left(U_i^k + W_i^{k+1}\right), \quad (11)$$

where $\prod_{S_i}$ represents the Euclidean projection of $U_i^k + W_i^{k+1}$ onto S_i. In this case, it represents that the weights satisfy the constraint S_i.

2) Layer-by-Layer Searching method: Based on the ADMM, a layer-by-layer searching and global pruning strategy are designed, which consider the distribution of redundant weights in each layer and the inter-layer information in the pruning search process, respectively. As the number of layers of CNN increases, the number of the weights and redundancy also increase [18]. For example, in the VGG16 model, except for the first layer, the shape of the front convolution layer is $(64, 64, 3, 3)$, while the shape of the last convolution layer is $(512, 512, 3, 3)$, larger than that of the front convolution layer. And the redundancy of the former may only be $30\% \sim 40\%$, while the latter may even exceed 95%. Therefore, the layer-by-layer pruning is started from the deep layers.

The flow chart of the layer-by-layer searching method is shown in Fig. 4. To obtain a dynamic trade-off between the pruning rate and the model accuracy, we define a fitness function as

$$f = \alpha * acc + \beta * rate_p \quad (12)$$

where α and β are the weighted values of the model accuracy after pruning and the current pruning rate, respectively. The ratio of α and β is not fixed, since the

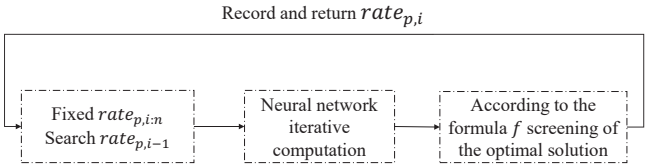


Fig. 4: Pruning Rate Searching

3) Global Pruning strategy: In the process of CNN inference, the data is transferred from the shallow layer to the deep layer, and there are associations between adjacent layers. And there is some information dependence between the adjacent layers. Simply considering the information of the single layer will cause accumulative errors, which has a tremendous impact on the performance of CNN. According to this characteristic, we propose a global pruning strategy on the basis of a layer-by-layer searching method, as shown in Algorithm 2. In the process of searching the optimal pruning strategy layer by layer, only the pruning rate is recorded, while the specific pruned weight masks of this layer are discarded, because the pruning results obtained at this time do not take into account the information from other layers. After traversing all the layers, the global ADMM iterative pruning is performed based on the pruning rate, and the CNN is trained to restore the network's accuracy.

Algorithm 2: Global Pruning Method

Data: *model*: CNN model, N: the Layer Number of model
Result: $model_p$: the pruned CNN model

```
train model;
list_rate ← [ ];
for i in 1, ..., N do
    Search the pruning rate of i layer as rate_p;
    add rate_p into list_rate;
end
model_p ← pruned model by using the ADMM and list_rate;
retrain model_p to restore accuracy;
return model_p;
```

TABLE I: The Structure of CNNs

Network	Structure
AlexNet	Conv3-64, Conv3-384, Conv3-256, 2×FC-4096, FC-10
VGG16	2×Conv3-64, 2×Conv3-128, 3×Conv-256, 3×Conv3-512, 3×Conv-512, FC-4096, FC-1000, FC-10
Plain20	7×Conv3-16, 6×Conv3-32, 6×Conv3-64, FC-10

IV. EVALUATION

We use an open-source simulator named MNSIM [20] for RRAM-based accelerator simulation experiments, which is a behavior-level modeling tool for NVM-based CNN accelerators developed in Python.

In our simulation setting, 1-bit ADC and DAC are used., and each weight of the DNN models uses 8-bit quantization. The size of the crossbar is 128×128, and each memristor stores 1 bit. Other hardware parameters are default values of MNSIM. In terms of the pruning method, the number of columns of each subarray is set as 32, and for the convolution layer, the number of rows is set as 32×9, and it is 1024 for the fully connected layer. The number of mask shapes of each subarray after pruning is 5, i.e., each submatrix will be clustered into 5 parts after pruning.

We evaluate Auto-prune with three DNN models, including AlexNet [21], VGG16 [22], and Plain20 [23]. The structures of them are shown in Table I, where Conv3-64 indicates that the convolution layer has 3×3 kernels and 64 output channels, and FC-4096 means that the fully connected layer has 4096 neurons. We execute the inference of the models on two datasets, i.e., CIFAR10 [24] and MNIST [25]. The CIFAR10 dataset consists of 60,000 color images, with 50,000 images for training and 10,000 images for testing. Each image has a size of 32×32 and contains 3 channels of RGB. MNIST dataset is composed of 70,000 grayscale images with the size of $28 \times 28 \times 1$, including 60,000 images for training and 10,000 images for testing. The proposed method is compared with the three models of Auto-Prune [8], PIM-Prune [5], and Pattern-Prune [4], and we get the comparison data of others in [8].

As mentioned above, the proposed method searches the pruning rate from the deep layer to the shallow layer. We can take the redundancy degree of three CNN models on MNIST as an example to observe the distribution of redundancy between each layer. Table II shows the parameter num_p about the number of weights in each column after pruning in each layer of different CNN models on the MNIST dataset. The smaller the num_p, the greater the pruning rate of the layer. In the pruning process, the first input layer and the last output layer are ignored. we can see that in each layer, the num_p in the deep layer is smaller than that in the shallow layer, which proves that the redundant weights in the deep layer are more than in the shallow layer. In other words, the pruning rate of the deep layer is higher.

TABLE II: Redundancy Degree in Each Layer of CNNs on MNIST

Network	num_p in Each Layer
AlexNet	0, 63, 27, 14, 14, 27, 14, 0
VGG16	0, 35, 31, 27, 27, 18, 4, 4, 4, 4, 4, 4, 4, 14, 12, 0
Plain20	0, 99, 90, 90, 18, 18, 18, 9, 9, 9, 9, 9, 9, 4, 4, 4, 4, 4, 4, 0

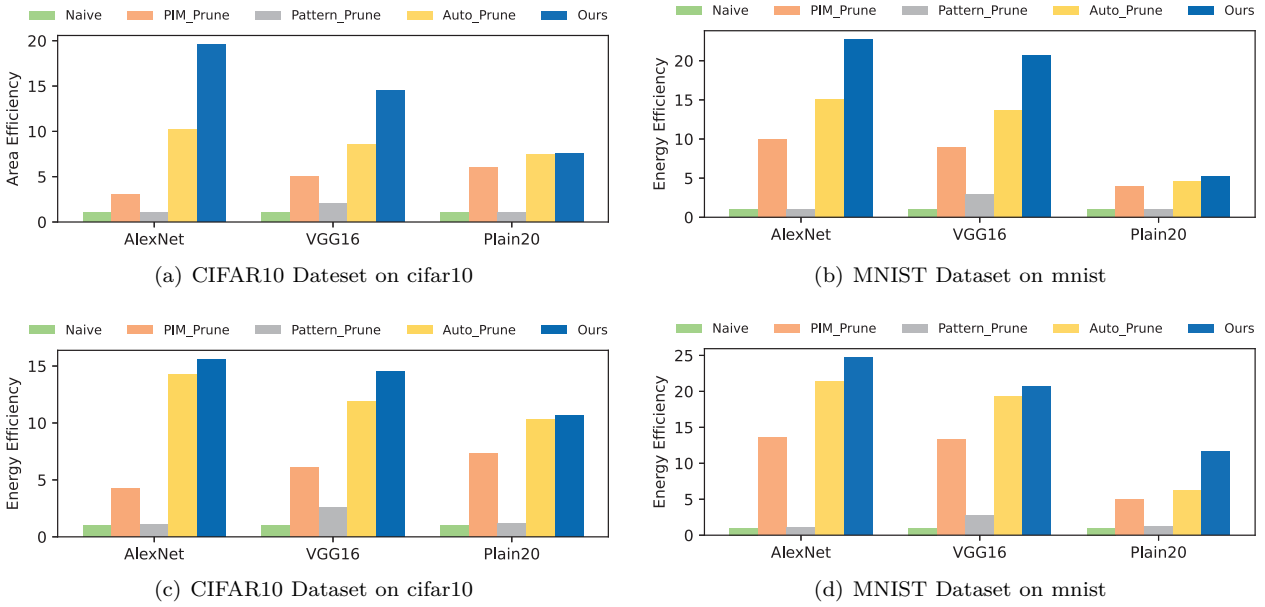
TABLE III: Compression Rate and Accuracy Drop on CIFAR10

Network	Method	CR	Acc5	Acc Drop
AlexNet	Naive	1	99.36%	-
	PIM-Prune	4.3	98.81%	0.55%
	Pattern-Prune	1.1	96.48%	2.88%
	Auto-Prune	14.3	99.10%	0.26%
	Ours	**20.5**	**99.11%**	**0.25%**
VGG16	Naive	1	99.29%	-
	PIM-Prune	6.1	98.62%	0.67%
	Pattern-Prune	2.6	98.43%	0.86%
	Auto-Prune	11.9	98.62%	0.67%
	Ours	**14.5**	**98.89%**	**0.40%**
Plain20	Naive	1	98.14%	-
	PIM-Prune	7.3	98.19%	-0.05%
	Pattern-Prune	1.2	98.24%	-0.10%
	Auto-Prune	10.3	98.29%	-0.15%
	Ours	**12.6**	**98.28%**	**-0.15%**

A. Compression Rate

Table III shows the comparison of compression rate obtained by four pruning schemes using AlexNet, VGG16, and Plain20 CNN models on CIFAR10 dataset, where Naive refers to the initial situation before pruning. The compression rate of our method is 20.5x, 14.5x, and 12.6x, respectively, and the accuracy drop is 0.25%, 0.40%, and -0.14%, respectively. It can be observed that our method achieves the highest compression rate of these pruning methods while maintaining the lowest accuracy reduction. Among the three CNN models, we observe that AlexNet has the highest compression rate. The reason is that AlexNet has higher weight sparsity than VGG16 and Plain20.

Table IV shows the results of the above methods in the MNIST dataset. Our method achieved the best compression rate of 24.7x, 20.7x, and 12.6x with the 0.40%, 0.02%, and 0.10% accuracy drop. We can see from the results of the two datasets that the compression rate is greater in the MNIST dataset. This suggests that CNNs trained on the MNIST dataset can achieve similar



(a) CIFAR10 Dateset on cifar10

(b) MNIST Dataset on mnist

(c) CIFAR10 Dataset on cifar10

(d) MNIST Dataset on mnist

Fig. 5: The Result of Area and Energy Efficiency

TABLE IV: Compression Rate and Accuracy Drop on MNIST

Network	Method	CR	Acc1	Acc Drop
AlexNet	Naive	1	98.89%	-
	PIM-Prune	13.6	98.41%	0.48%
	Pattern-Prune	1.1	97.78%	1.11%
	Auto-Prune	21.4	98.49%	0.40%
	Ours	**24.7**	**98.49%**	**0.40%**
VGG16	Naive	1	98.67%	-
	PIM-Prune	13.3	98.56%	0.11%
	Pattern-Prune	2.8	98.34%	0.33%
	Auto-Prune	19.3	98.63%	0.04%
	Ours	**20.7**	**98.65%**	**0.02%**
Plain20	Naive	1	98.13%	-
	PIM-Prune	5.0	97.91%	0.22%
	Pattern-Prune	1.2	97.42%	0.71%
	Auto-Prune	6.2	98.00%	0.13%
	Ours	**12.6**	**98.03%**	**0.10%**

accuracy with fewer parameters compared to the other dataset. Under the same circumstances, it is easier to obtain ideal results in the MNIST dataset.

B. Area and Energy Efficiency

Fig. 5(a) shows the area efficiency of different methods in the CIFAR10 dataset. For the CIFAR10 dataset, the proposed method achieves an improvement of 19.6x, 14.5x, and 7.6x, respectively. We observe that the area efficiency and compression rate of the three CNN models is equivalent, except Plain20. The reason is that the convolution layer of the Plain20 model is smaller than that of other models, and it brings the problem of low utilization after mapping to the crossbar. Fig. 5(b) shows the results on the MNIST dataset. Our method achieved improvements of 22.7x, 20.7x, and 5.2x, respectively. Except for Plain20, which has been explained earlier, the degree of elevation corresponds to the compression rate.

Fig. 5(c) shows the power efficiency of four pruning methods among three models on the CIFAR10 dataset. The energy efficiency is that of Naive. Our method achieves up to 15.6x, 14.5x, and 10.7x higher efficiency, which is superior to other methods. The results of these four methods on the MNIST dataset are shown in Fig. 5(d), where our method obtains 24.1x, 19.9x, and 10.1x improvement on the three CNN models, respectively, which are also superior to those of other methods. We observe that the energy saving of pruning methods on the three neural network models and the two datasets are positively correlated with the compression rate of the models because pruning not only reduces the area occupancy of the crossbar but also saves the corresponding peripheral circuit overhead.

V. CONCLUSION

In this paper, we propose a new pruning method for an RRAM-based neural network accelerator, which uses a fine-grained pruning method like a structured pruning method, and form regular matrices to match RRAM crossbar structures. And we propose a pruning optimization algorithm, including layer-by-layer searching and the global pruning strategy, to obtain the optimal pruning rate and search the redundant weights. Experimental results show that our method can achieve a higher model compression rate while maintaining a similar or even

higher accuracy, and reduce the power consumption and area cost of RRAM-based accelerators.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," Advances in neural information processing systems, vol. 33, pp. 1877–1901, 2020.

[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, "Advances in neural information processing systems," 2012.

[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," arXiv preprint arXiv:1312.6229, 2013.

[4] S. Yu, Y. Liu, L. Zhang, J. Wang, J. Yue, Z. Yuan, X. Li, and H. Yang, "High area/energy efficiency rram cnn accelerator with kernel-reordering weight mapping scheme based on pattern pruning," arXiv preprint arXiv:2010.06156, 2020.

[5] C. Chu, Y. Wang, Y. Zhao, X. Ma, S. Ye, Y. Hong, X. Liang, Y. Han, and L. Jiang, "Pim-prune: Fine-grain dcnn pruning for crossbar-based process-in-memory architecture," in 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020, pp. 1–6.

[6] X. Ma, G. Yuan, S. Lin, C. Ding, F. Yu, T. Liu, W. Wen, X. Chen, and Y. Wang, "Tiny but accurate: A pruned, quantized and optimized memristor crossbar framework for ultra efficient dnn implementation," in 2020 25th Asia and South Pacific design automation conference (ASP-DAC). IEEE, 2020, pp. 301–306.

[7] J. Meng, L. Yang, X. Peng, S. Yu, D. Fan, and J.-S. Seo, "Structured pruning of rram crossbars for efficient in-memory computing acceleration of deep neural networks," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 5, pp. 1576–1580, 2021.

[8] S. Yang, W. Chen, X. Zhang, S. He, Y. Yin, and X.-H. Sun, "Auto-prune: automated dnn pruning and mapping for reram-based accelerator," in Proceedings of the ACM International Conference on Supercomputing, 2021, pp. 304–315.

[9] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14 913–14 922.

[10] G. Yuan, X. Ma, C. Ding, S. Lin, T. Zhang, Z. S. Jalali, Y. Zhao, L. Jiang, S. Soundarajan, and Y. Wang, "An ultra-efficient memristor-based dnn framework with structured weight pruning and quantization using admm," in 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2019, pp. 1–6.

[11] F. Liu, W. Zhao, Z. He, Z. Wang, Y. Zhao, Y. Chen, and L. Jiang, "Bit-transformer: Transforming bit-level sparsity into higher preformance in reram-based accelerator," in 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE, 2021, pp. 1–9.

[12] Z. Song, B. Fu, F. Wu, Z. Jiang, L. Jiang, N. Jing, and X. Liang, "Drq: Dynamic region-based quantization for deep neural network acceleration," in 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020, pp. 1010–1021.

[13] Y. Wang, W. Wen, B. Liu, D. Chiarulli, and H. Li, "Group scissor: Scaling neuromorphic computing design to large neural networks," in 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2017, pp. 1–6.

[14] J. Wang, S. Yu, J. Yue, Z. Yuan, Z. Yuan, H. Yang, X. Li, and Y. Liu, "High pe utilization cnn accelerator with channel fusion supporting pattern-compressed sparse neural networks," in 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020, pp. 1–6.

[15] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," nature, vol. 453, no. 7191, pp. 80–83, 2008.

[16] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in 2017 IEEE international symposium on high performance computer architecture (HPCA). IEEE, 2017, pp. 541–552.

[17] T.-H. Yang, H.-Y. Cheng, C.-L. Yang, I.-C. Tseng, H.-W. Hu, H.-S. Chang, and H.-P. Li, "Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks," in Proceedings of the 46th International Symposium on Computer Architecture, 2019, pp. 236–249.

[18] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5687–5695.

[19] T. Zhang, K. Zhang, S. Ye, J. Li, J. Tang, W. Wen, X. Lin, M. Fardad, and Y. Wang, "Adam-admm: A unified, systematic framework of structured weight pruning for dnns," arXiv preprint arXiv:1807.11091, vol. 2, no. 3, 2018.

[20] Z. Zhu, H. Sun, K. Qiu, L. Xia, G. Krishnan, G. Dai, D. Niu, X. Chen, X. S. Hu, Y. Cao et al., "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in Proceedings of the 2020 on Great Lakes Symposium on VLSI, 2020, pp. 83–88.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[23] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 784–800.

[24] A. Krizhevsky, G. Hinton et al., "Learning multiple layers of features from tiny images," 2009.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

通用处理器性能测评基准负载设计与分析

钟伟军 逯海涛 任翔 尹航

中国电子技术标准化研究院 北京 100176

摘 要：针对通用处理器性能测评的基准负载设计进行了研究，以一套通用处理器的性能测评基准工具CPUBench 为例，讨论了基准负载选用原则和设计剪裁方法，并从指令并行度、指令混合比、架构无关性及相似度等维度对基准负载技术特性进行了分析。对 CPUBench 和 SPEC CPU2017 的基准负载进行了对比实验，实验数据表明 CPUBench 具备了与 SPEC CPU2017 同等的评价能力，可用于指导高性能通用处理器性能测评工作。

关键词：通用处理器；性能测评；基准负载；CPUBench；SPEC CPU2017

Design and analysis of benchmark load for performance evaluation of general-purpose processors

ZHONG Weijun, LU Haitao, REN xiang and YIN Hang

China Electronics Standardization Institute, Beijing 100176

Abstract: A study was conducted on the design of benchmark loads for performance evaluation of general-purpose processors. Taking a set of benchmark tools for performance evaluation of general-purpose processors, CPUBench, as an example, the selection principles and design clipping methods of benchmark loads were discussed. The technical characteristics of benchmark loads were analyzed from the dimensions of instruction parallelism, instruction mix ratio, architecture independence, and similarity. Comparative experiments were conducted on the benchmark load of CPUEnch and SPEC CPU2017, and the experimental data showed that CPUEnch has the same evaluation ability as SPEC CPU2017, which can be used to guide the performance evaluation of high-performance general-purpose processors.

Key words: general computing CPU； performance evaluation； benchmark workload；CPUBench； SPEC CPU2017

摘 要: 本文针对通用处理器性能测评基准负载设计进行了研究，讨论了基准负载设计原则及选用要求。以一套高性能能处理器的性能测评基准工具 CPUBench 为例，介绍了 CPUBench 基准负载的设计，从指令并行度、指令混合比、架构无关性及相似度等维度分析了负载的技术特性。对 CPUBench 和 SPEC CPU2017 的测试负载进行了对比实验，实验数据表面，CPUBench 具备了与 SPEC CPU2017 同等的评价能力，可用于指导高性能通用处理器性能测评工作。

关键词: 通用处理器 性能测评 基准 负载 SPEC CPU2017

0 引言

经过多年技术攻关和产品升级迭代，国产处理器性能和技术成熟度有了大幅提升，已步入规模化部署应用的阶段。现阶段，推动国产处理器产业良性发展，需要客观准确地认识当前产品技术水平以采取适合的技术发展路径实现产品持续演进。无论在学术界还是在工业界，一般都会借助性能测评基准对处理器性能做测试评价，发现性能瓶颈，指导下一代产品的设计改进[1]。

性能测评基是按照性能评估理论方法构建性能测评工具，它由一组经过筛选、剪裁形成的规格化、标准化的具有典型特征和行为的应用程序组成[2]。性能基准测试工具采用量化数值定量地展现处理器性能，在不同类型处理器之间、同一类型处理器的不同年代产品之间建立了一个具有性能可对比的模型，使得处理器产品间量化的性能比较成为可能。因此，性能基准测评结果对处理器研发和产业发展具有“风向标”的引导作用，是指导产品发展的“指挥棒”。

与国外相比，国产处理器研制起步相对较晚，技术状态一直处于追赶状态，企业将有限的研发力量都有投入在产品技术升级中，对性能测评技术的研究投入较少，更未能形成具有影响力的测试工具。国内普遍选择直接采用国外测评标准与性能基准测评工具进行产品的选型与评价，如 SPEC CPU2017 等。从长远眼光看，这种“拿来”方式不利于建立面向国内实际应用需求为导向的性能测评体系，难以助力国产处理器产品的高质量发展。

针对国内通用处理器测试评价领域缺少面向国内实际需求的性能基准测试工具的现实问题，开展了高性能通用处理器性能测评方法研究，研发了一款与当前工业界性能测评“事实标准”——SPEC CPU2017 对标的测评工具 CPUBench。实验数据表明 CPUBench 基本达到了与 SPEC CPU2017 同等的评价能力，且在应用需求覆盖性、平台兼容性、易用性方面有创新和突破[3]。评价能力和一致性。性能测评基准最关键、最核心的是用于评价处理器性能的测试负载，其直接关乎最终的评价结论。性能测评设计也是性能测评基准研制中耗时最长、难度最大的工作。本文针对 CPUBench 测试负载的选型、设计及其技术特点和实验验证情况进行分析阐述。实现效果进行分析。

1 CPUBench 基准负载设计原则与实现

处理器性能基准测评有两个关键技术，一是计算性能基准测评框架设计；二是计算性能基准测评测试负载的选用。前者是用于构建通用处理器计算性能测试基准运行框架，保证测评规则公平公正、测评流程公开透明、测试步骤标准一致、测试操作自动便捷。后者是选取测试负载的规则，需规定基准负载选择原则、负载裁剪要求、输入输出数据规定等，通过处理器运行测试负载的性能表现标定其性能。因测试负载对性能测评结果至关重要，测试负载的取舍对于处理器设计有很强的牵引作用，故测试负载选用是计算性能基准测评最最核心的技术。

1.1 测试负载设计遵循原则

计算性能基准测评目的是通过被测高性能处理器完成指定的测试负载表现出的性能来表征其计算性能。测试负载是构建基准测评最核心内容，在测试负载选用时应遵循五项基本原则[4]。

（1）源自代表性的应用程序，具有较高的业界认可度和影响力，满足应用领域的前沿技术真实需求，经过了足够多的用户验证。

（2）核心代码属于计算密集型，体现对运算性能的考核。

（3）采用高级编程语言实现，不含硬件专用指令，具有良好的跨平台性，支持不同硬件架构和操作系统移植。

（4）解题规模可定义、通过改变输入数据规模可构造成不同规模量级的计算任务。

（5）算法应为确定性计算、代码执行路径应可预测、可重复。

以上是计算性能基准测试负载的选取应遵循的基本原则，确保测试负载能够按照使用者的需求对被测试的通用处理器计算性能进行评价。其中，选用原则中的第一条强调的是测试负载的来源应符合实际应用需求，达到“源于实践、指导设计”的目的；第二条是强化测评基准设计要收敛到计算相关的评价；第三条是要求基准测评具有普适应和广泛性、且不偏向任何处理器指令体系硬件特征，确保测评公正性；第四条是便于优化调试；第五条是保证测试结果的一致性性、可比较和可追溯，即便不同平台之间浮点精度存在微小差异，但不应导致执行完成测试负载得出的结果存在较大差异。

1.2 测试负载设计实现

1.2.1 测试负载选用

除了遵循 1.1 节的设计原则外，在测试负选用方面，CPUBench 还考虑了工程实现方面的因素，包括综合评估各负载于多架构平台的移植成本、计算密集程度、编程语言、稳定性、公平性、知识产权等。此外，为了提升测评的全面性，测试负载选用中注重了测试基准对处理器体系结构整体的微架构特征覆盖度、业务领域覆盖度的考虑，同时也考虑不同负载之间的差异性，确保测评指标的均衡性。

CPUBench 选用了 12 个整型负载和 10 个浮点负载，如表 1 和表 2。这些测试负载均来源于实际的应用场景，整型测试负载除了来自编译器、视频编解码、数据压缩、加解密等常见的应用外，还选取了云计算、大数据领域的新兴应用；浮点负载除了来自计算辅助制造 CAD/CAE、基因工程、分子动力学等传统应用，还选取了电磁学、流体力学、气象预报等科学计算场景。其中，为了激励国内处理器设计更多贴近国内行业应用，CPUBench 在负载选取时考察范围扩展到国内自主研发的行业软件，选取了由中国空气动力研究与发展中心（CARDC）研发的面向流体工程的混合 CFD 平台软件风雷等国内开源应用。

为了引导处理器设计面向解决新兴应用，在测试负载选用中，CPUBench 还创新性地引入了采用 Java 和 Scala 语言编写的应用程序：kmeans 和 wordcount。实际上，Java 等解释型编程语言应用非常广泛，特别是对性能有较高要求的服务端应用场景中 Java 是主要的语言，如 Hadoop 大数据生态、Web 中间件 Tomcat 等。

表 1 整型测试负载

负载名称	应用领域	编程语言
x264	视频编解码	C
gcc	编译器	C++
gzip	数据压缩	C
tpcc	事务处理	C,C++
tpch	在线分析	C,C++
kmeans	机器学习	Java,Scala
wordcount	大数据-统计	Java,Scala
velvet	基因拼接	C
openssl	加解密	C
rapidjson	JSON 解析/生成	C++
python	Python 解释器	C
xz	数据压缩	C

表 2 浮点测试负载

负载名称	应用领域	编程语言
lightgbm	机器学习	C++
nektar++	气动声学	C++,Fortran
phenglei	流体动力学	C++
phyml	基因分析	C
gromacs	分子动力学	C++
povray	光线追踪	C++
openfoam	电磁学	C++
lammps	分子动力学	C++
cube	天体物理	C
wrf	天气预报	C,Fortran

1.2.2 测试负载剪裁要求

尽管需要秉持测试负载来源于线上业务应用，但并不意味着测试负载是对原生应用的简单截取集成。为了能够满足 1.2.2 节中对计算性能基准测评测试负载选用要求，特别是确保测试负载跨平台兼容、测评的公正性，需要对选取的测试负载进行剪裁、规格化。因此，研制一个可用的测试负载是一个复杂的过程，需要按照如下要求进行剪裁和规格化修调：

（1）裁剪业务核心计算逻辑之外的无关代码，剔除影响性能测试的因素。裁剪的内容包括代码中涉及网络数据传输、图形处理、I/O 接口操作及依赖于特定架构的汇编代码等；

（2）修调并重新制定测试输入数据集，并确每个测试负载在规定的时长范围内执行；

（3）应在各种软硬件环境下进行跨平台兼容性测验证，确保裁剪修调后的测试负载运行的兼容性、稳定性、可重复性，且对软硬件资源的占用及使用率符合大多数应用场景、覆盖大多数处理器架构。

2 CPUBench 基准负载特性分析

在本节中，对 CPUBench 选用的 12 个整型和 10 个浮点测试负载的特性进行分析，重点是分析对处理器微架构中影响计算性能的特征参数的覆盖性，确保测试负载对计算性能测评的多样性与完备性。本节从指令执行并行度和指令占比（Intstruction Mix）分析测试负载对计算性能特性的覆盖，然后运用 Top-down 微架构分析法（TMA，Top-down Microarchitecture Analysis）[11]分析测试负载对架构性能测试的覆盖性，最后运用主成分分析法（PCA，Principal Components Analysis）[5]分析负载测试的多样性。

2.1 测试负载指令执行并行度分析

指令执行并行度是处理器主要性能指标之一，不仅与微架构设计相关，如流水线深度、乱序执行、分支预测器的准确性、Cache 配置等，同时与应用程序特征密切相关，如数据访问模式，指令序列的依赖距离等，还有编译器的优化能力。指令执行并行度通常采用 IPC（Instruction Per Cycle，平均每时钟周期完成的指令数）表征。IPC 通过单位时间内处理器执行指令吞吐能力来客观反映指令并行度。

本文以 ARM V8 指令架构的处理器为例，使用 perf 工具采集 CPUBench 各测试载的 IPC。在单进程运行模式下，整型负载 IPC 范围是 0.76~2.73，浮点 IPC 范围是 0.36~2.76。其中，整型测试负载中 IPC 较高的是 openssl、x264，浮点是 nektar；整型测试负载中 IPC 最低的是 velvet，浮点是 lightgbm。经分析，IPC 高的原因是 openssl 的访存操作少，数据局部性好，指令执行效率高；x264、nektar 等程序，包含较多的矩阵向量运，数据访问规律性强，指令易于向量化。IPC 低的原因是 velvet、lightgbm 等数据局部性差，访存操作频繁，使得指令并行度较低。

对比 SPEC CPU2017 整型与浮点测试负载的 IPC 范围[6][7][8]，CPUBench IPC 覆盖区间与 SPEC

CPU2017 基本相似。经上述指令执行并行度分析可知 CPUBench 的负载具有不同指令并行度特征，覆盖范围广，可用于评估 CPU 微架构执行复杂指令流的效率。

2.2 测试负载指令混合比分析

指令占比是程序的主要微架构无关特征之，不同的微架构具有不同的计算单元数量，执行不同指令所需时钟周期也不相同。因此，性能基准工具涵盖各种指令比例的测试负载至关重要。通过采集 CPUBench 各个负载的指令占比数据可知整型和浮点测试负载涵盖了较为丰富的指令类型，且指令占比呈现显著的差异化。

CPUBench 测试负载指令类型覆盖广的特征为：整型测试负载平均指令分布为 Integer 46.7%，FP 0.1%，SIMD 1.4%，Control 18.8%，Load 23.5%，Store 9.4%；浮点测试负载平均指令分布为 Integer 30.02%，FP 22.9%，SIMD 2.8%， Control 11.8%，Load 25.8%，Store 6.6%。

CPUBench 测试负载指令占比呈现显著的差异化特征为：基于 mysql 的两个测试负载 tpcc 和 tpch，代码规模大，代码控制流复杂，具有较高的控制质量/分支指令频率，适合测评复杂场景下分支预测器的执行效率；视频编码负载 x264，包含矩阵运算，其数据访问模式有较强的规律性，故可向量化，具有较高的 SIMD 指令占比，适合测评编译器的向量优化能力及处理器向量化执行效率；各测试负载中包含不同比例的 Load 和 Store 指令，可真实反映实际应用差异化的数据访问读写模式[16]。

2.3 测试负载微架构覆盖性与相似度分析

采用 Top-down 微架构分析方法可以分析测试负载对微架构性能的覆盖性。主要从测试负载对处理器流水线前端依赖（Front Bound）、后端依赖（Backend Bound）和错误预测（Bad Speculative）三个微架构特性入手。经分析 CPUBench 的整型测试负载覆盖了上述三种特性，浮点测试负载主要覆盖了后端依赖特性，测试数据如图 1 和图 2。

整型测试负载覆盖较全面，测试负载 tpcc 和 tpch 因其代码规模庞大，控制流复杂，指令 footprint 比其他应用大，指令 Cache miss 率高，具有非常典型的前端依赖特性；测试负载 velvet、rapidjson 等处理数据量大，访存操作密集，具有典型的后端依赖特性；其他的测试负载在错误预测特性方面也有覆盖；浮点测试负载则整体上偏重于后端依赖特性。

根据 Top-down 验证数据的分析，总体上 CPUBench 对微架构的特征覆盖较全面。

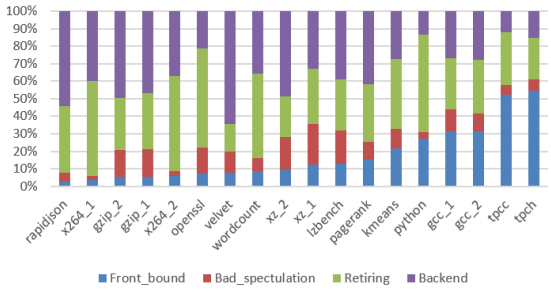


图 1 整型测试负载微架构特性测评覆盖性分析

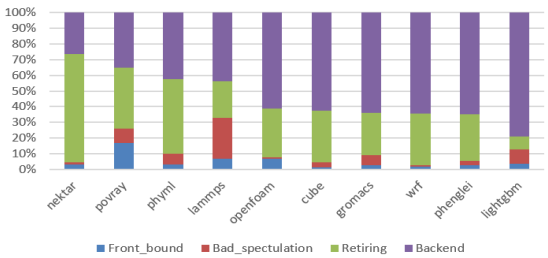


图 2 浮点测试负载微架构特性测评覆盖性分析

采用主成分分析法分析 CPUBench 测试负载特征的相似度，同时也针对 SPEC CPU2017 测试负载进行分析[17]，并对比两个性能基准测评工具的情况。主成分包含 26 种特征或 pmu 事件，包括 IPC、L1/L2/L3 Cache miss 率、TLB miss、条件分支跳转、直接跳转、间接跳转、指令占比、访存带宽、working size set、footprint 等。使用聚类工具进行相关性分析，得到相似度数据。

通过聚类分析，按照统计欧氏距离大于 10 的分类，将整型负载降维成 7 类新特征，浮点聚类成 5 类新特征。

针对整型负载，SPEC CPU2017 覆盖了 7 类新特征中的 5 类，CPUBench 覆盖 6 类；浮点负载，SPEC CPU2017 覆盖了 5 类新特征中 4 类，CPUBench 覆盖了 3 类，具体情况如图 8 和图 9。在整型方面，CPUBench 比 SPEC CPU2017 覆盖度高是因为引入了基于 Java 语言的大数据相关负载，单独覆盖了 2 类新特征。

总体来说，在测试负载特征差异化方面，CPUBench 与 SPEC CPU2017 基本相当，各形成了独有的应用特征。其中，CPUBench 引入的大数据应用及 Java 语言负载对反映当前流行的服务端应用特征有很强的参考价值。

4 验证试验与对比分析

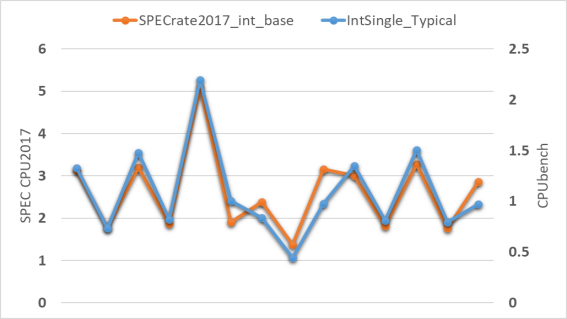
采用 CPUBench、SPEC CPU2017 对 10 款基于国内外服务器处理器的计算系统进行了实测，表 3 为处理器产品型号。

表 4 处理器信息描述

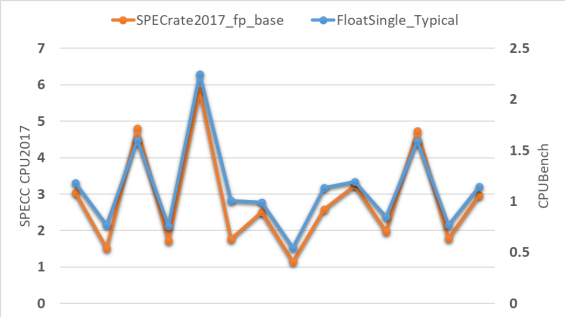
架构	型号
x86_64	ygon C86 7285 KH-37800D Intel Xeon Gold 6248 Intel Xeon Bronze 3106
aarch64	Kupeng920 5250 Kunpeng920 5251K FT-S2500
loongarch	LS3C5000L
ppc64le	POWER8
sw_64	SW3231

为使得测试数据具有可比性，同一被测平台的软硬件配置保持一致。

图 3 是测试结果的对比图。从对比结果可以看出，针对本次测试的 10 款产品，其性能基准的性能分值曲线趋势非常相似。从实验验证结果分析，CPUBench 的测试负载设计合理、有效，也证明本文提出的负载设计方法学是正确的，能够实现正确、合理地评估不同被测对象的计算性能的目标。



a. 整型测试负载性能测评分值



b. 浮点测试负载性能测评分值

图 3 CPUBench 和 SPEC CPU2017 单任务测评结果趋势对比

5 结语

本文展示了一款通用计算 CPU 性能基准工具 CPUBench 测试负载的研制实践。实验验证情况证明了 CPUBench 测评结果可用性与合理性，具有与对标基准程序 SPEC CPU2017 较为一致的评价结论。

本文提出的负载设计方法可用于指导通用计算 CPU、服务器等计算产品的设计、优化和规格选型等工作。从技术成熟度看，本项工作还需要持续开展技术迭代，还有很多需要进一步探索和改进之处。未来我们将继续对 CPUBench 工具展开更深入细致的分析，并公开收集不同业务领域的应用负载，尽可能贴近实际应用需求，为行业内用户提供除 SPEC CPU 之外的测评工具的选择，为国内行业服务。

参考文献:

[1] Samuel Kounev, Klaus-Dieter Lange and Jóakim von Kistowski, Systems Benchmarking[M],1st ed. Springer International Publishing, 2020.

[2] 吕超,戴晨,张为华.计算机体系结构基准测试程序集的研究[J].计算机应用 与软件,2013,30(10):189-194.

[3] 逯海涛，任翔，钟伟军，等．CPUBench: 一款开放的通用计算 CPU 性能基准工具[J]. 微电子学与计算机，2023，40（5）：75-83.

[4] SPEC CPU [EB/OL].[2022-06-16].https://www.spec.org/cpuv8.

[5] Top-down Microarchitecture Analysis Method [EB/OL]. https:// www.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/methodologies/top-down-microarchitecture-analysis-method.html

[6] Maćkiewicz, Andrzej, and Waldemar Ratajczak. "Principal components analysis (PCA)." Computers & Geosciences 19.3 (1993): 303-342.

[7] Shuang Song, Qinzhe Wu, Steven Flolid, Joseph Dean, Reena Panda, Junyong Deng, Lizy K. John, “Experiments with SPEC CPU 2017 Similarity, Balance, Phase Behavior and SimPoints”

[8] Ankur Limaye and Tosiron Adegbija，“A Workload Characterization of the SPEC CPU2017 Benchmark Suite”

[9] A. Limaye and T. Adegbija, "A Workload Characterization of the SPEC CPU2017 Benchmark Suite," 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2018, pp. 149-158, doi: 10.1109/ISPASS.2018.00028.

[10] Navarro-Torres A , J Alastruey-Benedé, P Ibáez-Marí n, et al. Memory hierarchy characterization of SPEC CPU2006 and SPEC CPU2017 on the Intel Xeon Skylake-SP[J]. PLOS ONE, 2019, 14.

[11] Sarabjeet Singh，Manu Awasthi，“Memory Centric Characterization and Analysis of SPEC CPU2017 Suite”.

作者简介部分

钟伟军：男，高级工程师，研究方向为集成电路测试与评价.

目标检测算法失效模式的抽取与验证

谢妮珊[①②] 李瑞[②] 董乾[②] 孟令中[*②]

[①](中国科学院大学 北京 100190)

[②](中国科学院软件研究所 北京 100190)

摘 要：在目标检测领域，理解和预测目标检测算法可能出现的失效模式对于算法的稳健性和鲁棒性有着重要的意义。针对这个需求，本文明确定义了目标检测算法的失效模式，并从众多常见失效现象中抽取出了几种符合定义的失效模式。为了证明这些提出的失效模式是正确的，我们对失效模式的普适性这一具有代表性的性质进行了验证。本文利用故障注入方式在 VOC2007 数据集上模拟这些失效模式所对应的失效场景，在这些场景下观察分析 Single Shot MultiBox Detector(SSD)模型的表现。实验结果说明在不同算法模型下同样会产生该文所抽取出的失效模式，这验证了失效模式的普适性，并进一步证实了本文提出的失效模式的正确性。这个过程对于深入理解算法在面对不同环境因素时的性能表现，以及为算法的进一步优化，提供了宝贵的参考。

关键词：失效模式；目标检测算法；故障注入

Extraction and Verification of Failure Modes in Object Detection Algorithms

XIE Nishan[①②] LI Rui[②] DONG Qian[②] MENG Lingzhong[*②]

[①](*University of Chinese Academy of Sciences, BEI'jing 100049, China)*

[①](*Institute of Software Chinese Academy of Sciences, BEI'jing 100190, China*)

Abstract: In the field of object detection, understanding and predicting the possible failure modes of object detection algorithms are of significant importance for the robustness and stability of the algorithms. To address this need, this paper clearly defines the failure modes of object detection algorithms and extracts several failure modes that conform to this definition from a variety of common failure phenomena. In order to validate the correctness of these proposed failure modes, we tested the universality, a representative property of failure modes. Using fault injection, we simulated the failure scenarios corresponding to these failure modes on the VOC2007 dataset and analyzed the performance of the Single Shot MultiBox Detector (SSD) model in these scenarios. Experimental results indicate that the failure modes extracted in this paper also emerge in different algorithm models, thus confirming the universality of the failure modes and further validating the correctness of the failure modes proposed in this paper. This process provides valuable insights for understanding the performance of algorithms when facing different environmental factors and offers valuable references for further optimization of algorithms.

Key words: Failure Mode; Object Detection Algorithms; Failure Injection

1. 引言

近年来，人工智能与视觉感知技术的融合逐渐深入，引发了各领域的变革，使得目标检测算法的应用范围越来越广泛，如自动驾驶[1]、医学影像以及环境监控等。随着目标检测算法日益深入到人们的生活中，可以发现目标检测算法在生活的各方面都发挥了重要作用。然而，正是因为它与人们的生活紧密相连，其失效的发生所带来的危害也越来越严重。具体来说，对于自动驾驶技术，若目标检测算法无法准确识别物体，会导致车辆行人误判，对人们的生命安全构成巨大威胁；对于医学影像，噪声干扰或是图片质

[*]通信作者：孟令中 lingzhong@iscas.ac.cn

基金项目：本研究中部分得到了中国科学院青年创新促进会的资助

量不足可能会导致漏诊或误诊，为病患的治疗带来重大风险[2]；在环境监控方面，对行人身份识别错误可能引发泄密或权限错误赋予等安全问题[3]。

正是这些现象引发的问题，警示了我们目标检测算法的失效问题的严重性，需要引起足够的重视。为此，分析并研究目标检测算法的失效模式成为了迫在眉睫的任务。通过大量的案例研究，观察到了一些相似的问题现象，例如，在自动驾驶领域，面对复杂动态环境中的遮挡物体，或是照明变化等引发剧烈光线变化的场景时，算法的表现较差，性能明显下降。另外，在图像质量至关重要的场景，如医学影像领域或卫星遥感领域，由于图像存在噪声干扰和图像清晰度不够导致算法出现失效的这一现象频频出现。还有，在环境监控领域，行人戴口罩或是出现雨雪等天气时，由于存在遮挡和环境噪声引发的算法失效也屡见不鲜[4]。

对于这些失效现象，需要从中抽取出失效模式，使其更为规范且具有代表性。失效模式的存在反映了目标检测算法的局限性和挑战。分析出正确的失效模式可以减少算法的内在问题，为算法的改进提供有益的指导和帮助。因此，这篇论文的主要目的就是抽取出一些常见的失效模式并验证这些提出的失效模式的正确性，以期能够提升对目标检测算法失效问题的理解，从而进一步优化算法，降低其在实际应用中的失效风险。

2. 目标检测算法的失效

在本章节中，本文将会探讨目标检测算法在实际应用中面临各种应用场景时可能会发生的一些失效现象。这些失效常常来自于环境条件的变化和特定的场景。

首先目标检测算法的失效可能源于环境光照的剧烈变化，例如，当车辆从明亮环境进入阴暗的隧道，或在日夜交替的过程中，这种情况可能导致图像的曝光程度异常，算法的性能会受到一定程度的影响，影响算法的识别能力，出现部分失效现象。此外，摄像设备的硬件环节或图像传输与存储过程中可能会导致图像出现噪声，这会导致目标检测算法在处理这样被噪声干扰的图像时面临挑战。当出现设备移动、相机焦距不准确，或者由于环境条件如雨雪、雾天，照明条件的变化等情况时，可能会导致算法的输入图像变得模糊，从而导致算法的性能下降[5]。再比如，当图像中包含遮挡物体时，例如行人、车辆或建筑物等任何可能阻碍传感器清晰观察目标物体的元素时，都会一定程度上对目标检测算法构成困扰。最后，物体自身的旋转、姿态变化，或者观察视角的变化，也会导致算法的识别和定位能力受到影响。

这些失效现象普遍存在于各类目标检测应用场景中，了解和研究这些现象对深入理解目标检测算法的失效模式具有重要意义。在下一章中，会对这些失效现象进行理解和分析，从这些失效现象中抽取出普遍的失效模式，从而为目标检测算法的改进和优化提供有益的参考。

3. 目标检测算法失效模式

本章节中对目标检测算法的失效模式进行了定义，规定了失效模式需满足的性质，随后根据失效模式的定义来从上文中的失效现象中抽取出一些常见的目标检测算法的失效模式。

3.1 目标检测算法失效模式定义

在对目标检测算法的一系列失效现象进行理解分析中，本文引入了一个关键概念——失效模式，在特定应用场景和数据集下，目标检测算法表现出的具有普适性、可预见性和可重现性的预期外的性能下降情况。它与传统硬件或软件的失效模式的区别在于，传统的失效模式通常涉及到硬件或软件的功能完全丧失[6]，而目标检测算法的失效模式则更偏重于算法性能的下降，例如在关键的评估指标如平均精确率的平均值（Mean Average Precision, mAP）、准确率、召回率等方面的表现不佳。

对于目标检测算法的失效模式而言，普适性，即它们并不仅仅在某一特定的算法模型上出现，而是在大多数或所有的算法模型中都会出现。这也意味着，即使对算法进行修改或优化，这些失效模式也有可能产生。因此，当处理这些失效模式时，我们需要从更宽广的视角来进行，而不仅仅是针对某一特定的算法。其次，失效模式的可重现性，即在相同或类似的条件下，同样的失效模式可以再次出现。这使得我们有可能通过分析和研究这些失效模式，找到它们产生的原因，并在未来的算法设计和优化中避免同样的问题。最后，可预测性意味着在给定的条件或场景下，算法是否出现对应的失效模式是可以预测到的。这种可预测性有助于我们提前做好准备，防止可能出现的问题，提升算法的稳定性和可靠性。

在所有这些特性中，普适性是失效模式最为重要的性质，确保了失效模式的广泛应用性和研究价值。这是因为，如果一个失效模式只存在于某一个特定的目标检测算法中，那么这样的模式影响范围较小，很可能只是该算法特有的问题，对于大多数其他的算法并无影响。相反，如果一个失效模式普遍存在于多种目标检测算法中，那么它就可能暴露了视觉感知领域的一种普遍的缺陷，对于提升整个领域的鲁棒性和性能具有显著的指导意义。因此，在本文后面的章节中会通过对抽取出的失效模式的普适性进行验证，从而说明本文所抽取出的失效模式的正确性。

3.2 目标检测算法常见失效模式抽取

基于上文中失效模式的定义以及失效现象，我们将提炼出目标检测系统中出现的主要失效模式。通过分析，我们抽取出了以下几种失效模式：图像曝光异常失效、图像噪声干扰失效、模糊图像处理失效、遮挡物体识别失效和视角变化失效。这些失效模式满足本文定义的失效模式的性质：普适性、可重现性和可预测性的失效模式。下面是对这些失效模式的详细解释。

1）图像曝光异常失效：对于环境光照条件的剧烈变化，一些目标检测算法可能不能很好地适应，导致识别性能下降。这一失效模式在多种算法中普遍存在，同时可以通过模拟光线变化、光线异常的场景重现，并且如果预先知道即将进入曝光异常的环境，能够预测到这一失效模式的出现。

2）图像噪声干扰失效：多种原因可能导致图像出现噪声，从而影响目标检测算法的性能。此失效模式在许多算法中普遍存在，并且可以通过在图像中注入条纹噪声、高斯噪声等方式模拟失效场景，在噪声条件预知的情况下，该失效模式的出现可预测。

3）模糊图像处理失效：若图像因各种原因，例如设备移动、焦距不准确、环境条件等变得模糊，可能导致算法性能下降。此失效模式在任何模型中均有可能出现，通过对图像进行模糊处理的方式可以重现失效场景，若已知图像模糊，失效的发生可以预测。

4）遮挡物体识别失效：当目标物体被其他物体遮挡时，算法可能面临识别困难。这是一个广泛存在于不同算法中的失效模式，对物体加入部分遮挡物能够重现这一失效模式，且当已知有遮挡物存在时，这一失效模式的出现可预测。

5）视角变化失效：物体的旋转、姿态变化，或观察视角的改变，可能影响算法的识别和定位能力。此失效模式在不同算法中普遍存在，可以通过改变观察物体的视角来重现，在已知将发生视角变化的情况下，其失效的出现可预测。

表 1 常见目标检测算法失效模式

失效模式	失效说明	失效场景
图像曝光异常失效	目标检测算法在处理过曝或欠曝图像时出现性能下降的失效现象	日夜交替、明暗变化、直射阳光、阴影处等
图像噪声干扰失效	目标检测算法在处理噪声干扰图像时出现性能下降的失效现象	图像传输、图像存储等
模糊图像处理失效	目标检测算法在处理模糊图像时出现性能下降的失效现象	运动模糊、设备焦距错误、恶劣天气等
遮挡物体识别失效	目标检测算法在处理含有遮挡物的图像时出现性能下降的失效现象	运动模糊、设备焦距错误、恶劣天气等
视角变化失效	目标检测算法在处理视角变化的图像时出现性能下降的失效现象	拍摄角度变化、相机移动、物体旋转等

4. 验证方法

本研究中提出的失效模式是基于多种来源的数据，包括失效案例和学术文献等，例如其中一种失效模式——遮挡物体识别失效，我们曾经在人脸识别[7]中见过类似的失效现象，当时所使用的算法模型是ArcFace。我们需要观测在不同的算法模型上，是否会出现该失效模式，证明本文所抽取出的失效模式的普适性，进而说明它的正确性，而进行验证的核心实验方法就是故障注入。

传统的故障注入方法，主要通过将选定的故障模型有意识地施加于特定的目标系统中，目的是加速该系统的错误和失效的发生，进而收集和分析系统对故障的反应信息[8]。尽管这种方法在各种系统的鲁棒性测试中被广泛应用，但若想应用在目标检测算法领域，需要对这种方法进行创新和扩展。

本文考虑到目标检测算法对输入数据的敏感性，数据集内的图像对目标检测算法性能有着举重若轻的影响，于是将故障注入点设定在输入数据集上。在实验环节中，根据实际的失效模式，设计了一系列故障注入策略，包括调整图像质量（如改变亮度、加入噪声、模糊处理等）、添加遮挡物、变换视角等。这些策略能够模拟真实环境中可能出现的故障情况，而且每一种策略都对应到目标检测算法的一种特定失效模式。

此外，为了能更直观细致地观察和理解失效模式对算法性能的影响，本方法引入了“故障注入等级”这一分级概念，将失效影响分为了轻度、中度和严重三个等级，对应这三种等级来进行不同程度的故障注入。这种分级可以更清晰地了解不同失效模式以及不同程度的故障对算法性能的影响，为后续分析提供更详尽的数据。

故障注入的流程如图 1 所示，首先选定要进行普适性验证的失效模式，选择合适的测试算法和数据集。然后，设定故障注入参数，包括故障注入点和故障注入强度，以模拟可能发生的各种失效现象。接下来根据参数实施故障注入，并收集实验数据进行分析。数据分析主要包括评价指标分析和观测状态分析，目的是从多角度理解和判断失效模式是否已经出现。最后，我们基于实验数据和分析结果，判断失效模式的出现情况，并评估其故障等级，这样可以更深入地理解其在实际环境中可能对目标检测算法的影响程度。通过这种流程化的方法，我们得以有效地验证和理解目标检测算法中的失效模式。

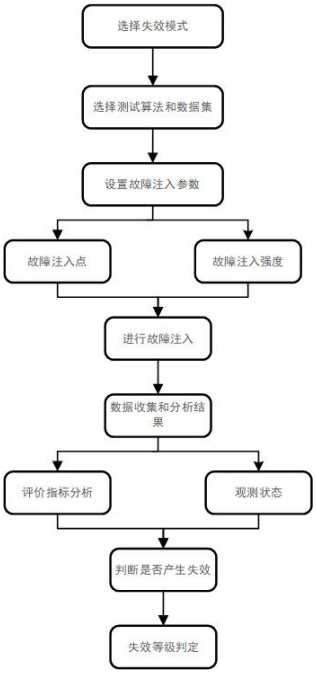


图 1 故障注入流程图

5. 实验验证与结果分析

在本章节中，首先说明实验的设置与准备工作，随后对每条失效模式通过故障注入的方式生成对应的失效场景，观察算法是否会发生失效，测试失效模式的普适性，从而验证本文提出的失效模式的正确性。

5.1 实验设置与准备

本章所涉及到的实验均在 PyTorch 1.13 和 Python 3.7 的开发环境下进行。我们选择了在 VOC2007 数据集上对目标检测模型——SSD 模型[9]进行验证。

SSD 模型在目标检测领域有着广泛的应用，它通过在一次前向传播过程中完成目标检测，显著提升了检测的速度。与此同时，其多尺度特征图和默认框的使用使得模型在处理不同大小目标的能力上展现出优异的性能。因此，SSD 模型的广泛应用以及其在各种场景中的优秀表现，使得其成为进行失效模式验证实验的理想选择。

在 PASCAL VOC 2007 数据集上，本研究对 SSD 模型进行了训练和评估。VOC 2007 数据集是视觉目标类别识别和检测领域中最知名的标准数据集之一，它包括了飞机、汽车、鸟类、猫等 20 个类别的物体，以及各种场景的图片，总共有 9963 张图片，目标的大小、形状和位置各异，目标之间可能存在重叠，对模型的鲁棒性提出了高要求。这样的多样性使得它能很好地模拟目标检测算法在真实环境中可能遇到的各种情况[10]。在未进行故障注入的情况下，SSD 模型在 VOC2007 数据集上获得的 mAP 为 0.7153，这个结果将作为后续实验的基准。

5.2 图像曝光异常失效

为了验证图像曝光异常失效这一失效模式，本文在测试集上对图像进行了故障注入，模拟图像曝光程度的变化。本文设计了轻度、中度和重度三个故障注入等级，模拟不同程度的曝光异常，来观察其对算法性能的影响。具体的说明见表 2，其中故障参数 factor 为图像的曝光参数，其值越高，则图像过曝光的程度越严重[11]。

表 2 VOC2007 数据集过曝光故障注入等级描述

故障注入等级	故障参数	数据集失效表现
轻度	factor=2.0	曝光略有异常，图像质量受到一定影响
中度	factor=3.0	曝光中度异常，图像质量明显下降
重度	factor=4.0	曝光严重异常，图像信息丢失

通过遍历测试集图像中的像素点，对其亮度值增加一个常数的方式来实现图像亮度的增强[12]，轻度故障注入情况下的结果显示，SSD 模型的识别性能略有下降，其 mAP 从 0.7153 下降到了 0.6450，这与表 2 中轻度失效表现的描述一致。在中度故障注入实验时，进一步增强了图像的亮度。此时，图像质量明显下降，导致 SSD 模型的识别性能进一步下滑，mAP 下降到了 0.5375。同时，部分目标开始丢失，这也符合表 2 中对中度故障情况下失效表现的描述。最后，在重度故障注入情况下，图像亮度被增强到极致，模拟了过曝光的极端情况。此时，图像质量严重受损，SSD 模型的识别性能大幅度下降，mAP 仅为 0.4384。识别置信度严重下降，大量目标无法被识别，这正好验证了表 2 中对重度失效的描述。这一系列的实验结果验证了本文所提出的图像曝光异常失效模式的存在，并进一步展示了不同程度的过曝光对算法性能的影响。

表 3 图像曝光异常失效影响

失效等级	mAP	mAP变化比(%)	失效影响
轻度	0.6450	-10.20%	目标检测置信度略有下降，算法性能略受影响
中度	0.5375	-25.17%	目标检测置信度大幅下降，算法性能明显降低
重度	0.4384	-38.97%	大量目标检测失败，算法性能几乎完全丧失

(a)原图-猫 (b)轻度曝光-猫 (c)中度曝光-猫 (d)重度曝光-猫

(e)原图-人，摩托车 (f)轻度曝光-人，摩托车 (g)中度曝光-人，摩托车 (h)重度曝光-人，摩托车

(i)原图-人，自行车 (j)轻度曝光-人，自行车 (k)中度曝光-人，自行车 (l)重度曝光-人，自行车

图 2 过曝光对 SSD 模型识别结果的影响

在对图像进行过曝光故障注入的实验中，出现了一个较为有趣的现象：部分图片在提高曝光度后，模型的识别置信度反而会上升。对于这一现象进行深入分析后，本文提出了以下几种可能的解释。首先，这反映出了数据集的偏差。在实验使用的 VOC2007 数据集中，存在一部分图片本身曝光程度偏低，即这些

图片相对偏暗。在这种情况下，提升曝光度可能有助于模型更清晰地识别图像中的目标，从而提高识别准确率。其次，这个现象也可能是模型过拟合训练数据的结果。换言之，模型可能过度学习了训练数据中的特定噪声或其他非目标特征，而忽视了真正需要识别的目标[13]。当实验过程中提升图像的曝光度时，这些噪声或非目标特征相对被抑制，使得模型更能聚焦于目标的识别。最后，提升曝光度可能起到了图像增强的效果，如提升了图像的对比度，从而增大了目标与背景的区别，这也可能导致了模型识别准确率的提高。

(a)原图-人，马 (b)轻度噪声-人，马 (c)中度噪声-人，马 (d)重度噪声-人，马

(e)原图-椅子 (f)轻度噪声-椅子 (g)中度噪声-椅子 (h)重度噪声-椅子

(i)原图-羊 (j)轻度噪声-羊 (k)中度噪声-羊 (l)重度噪声-羊

图 3 条纹噪声对 SSD 模型识别结果的影响

5.3 噪声干扰失效

本小节中将针对图像噪声干扰失效模式使用条纹噪声对测试数据集进行三种程度的故障注入。条纹噪声是一种易于直观观察到的噪声[14]，实验通过控制条纹的宽度和间隔来模拟不同等级的故障。故障等级说明见表 4，其中参数 width 为条纹的宽度，gap 为条纹之间的间隔。

表 4 VOC2007 数据集噪声故障注入等级描述

故障注入等级	故障参数	数据集失效表现
轻度	width=3, gap=40	噪声水平较低，图像质量受到一定影响
中度	width=2, gap=20	噪声水平中等，图像质量明显下降
重度	width=2, gap=10	噪声水平较高，图像质量严重下降

实验在原始图像上添加了横向随机颜色的轻度、中度和重度的条纹噪声，以模拟在实际场景中可能遇到的不同程度的噪声干扰。注入不同程度的噪声后，将这些图像分别输入到 SSD 模型中，以评估模型在处理含噪图像时的性能。结果显示，随着噪声水平的增加，模型的识别性能显著下降。对于轻度的噪声干扰，识别的精度略有下降，mAP 为 0.6362；对于中度的噪声干扰，识别结果受到一定影响，部分图像出现识别错误的结果，mAP 下降到 0.5218；对于高度的噪声干扰，模型的识别能力几乎完全丧失，mAP 大幅

下降到 0.3355，出现大量识别错误或无法识别的情况。这验证了图像噪声干扰失效模式的存在，并且明确了噪声故障等级和算法识别性能之间的负相关关系。

表 5 噪声干扰失效影响

失效等级	mAP	mAP变化比(%)	失效影响
轻度	0.6362	-11.43%	目标检测置信度略有下降，算法性能略受影响
中度	0.5218	-23.36%	目标检测置信度大幅下降，部分目标识别失败，算法性能明显下降
重度	0.3355	-53.29%	大量目标检测失败，算法性能几乎完全丧失

5.4 模糊图像处理失效

本小节的实验会在图片中注入模糊效果，然后观察算法在处理前后的图像测试集上的性能表现。实验对 VOC2007 测试集中的每一张图片都进行了高斯模糊处理[15]。通过调整高斯模糊滤波器大小(kernel_size)和标准差(sigma)来控制图片的模糊程度[16]。然后，在处理后的图像测试集上部署 SSD 模型，观察模型的表现。

表 6 VOC2007 数据集高斯模糊故障注入等级描述

故障注入等级	故障参数	数据集失效表现
轻度	kernel_size=(5,5), sigma=1.5	图像模糊程度较低，图像质量轻微受损
中度	kernel_size=(9,9), sigma=2.5	图像模糊程度中等，图像质量明显受损
重度	kernel_size=(11,11), sigma=3.75	图像模糊程度较高，图像质量严重受损

结果显示，随着故障程度的加重，模型的 mAP 也相应地降低。当进行轻度故障注入时，mAP 的值从原始的 0.7153 下降到 0.6506，模型性能略有下降；当故障注入为中度时，mAP 的值下降到 0.5069，目标检测置信度明显降低；当进行重度故障注入时，mAP 的值下降到 0.3814，模型几乎完全失效，无法成功检测目标物体。这些数据充分证明了模糊图像处理失效这一失效模式是真实存在的，且随着模糊程度的增大，识别精度受到的影响越大。

表 7 模糊图像处理失效影响

失效等级	mAP	mAP变化比(%)	失效影响
轻度	0.6506	-9.42%	目标检测置信度略有下降，算法性能略受影响，基本能够正常运行
中度	0.5069	-29.43%	目标检测置信度大幅下降，算法性能明显降低，部分目标检测失败
重度	0.3814	-46.90%	大量目标检测失败，算法性能几乎完全丧失

(a)原图-飞机 (b)轻度模糊-飞机 (c)中度模糊-飞机 (d)重度模糊-飞机

(e)原图-沙发 (f)轻度模糊-沙发 (g)中度模糊-沙发 (h)重度模糊-沙发

(i)原图-汽车 (j)轻度模糊-汽车 (k)中度模糊-汽车 (l)重度模糊-汽车

图 4 图像模糊对 SSD 模型识别结果的影响

(a)原图-人 (b)轻度遮挡-人 (c)中度遮挡-人 (d)重度遮挡-人

(e)原图-猫 (f)轻度遮挡-猫 (g)中度遮挡-猫 (h)重度遮挡-猫

(i)原图-飞机 (j)轻度遮挡-飞机 (k)中度遮挡-飞机 (l)重度遮挡-飞机

图 5 遮挡图片对识别结果的影响

在进行实验时，本文发现了与进行过曝光故障注入时一样的现象：在向图像中注入了轻度高斯模糊时，SSD 模型进行目标识别的置信度反而会上升，如图 3(j)所示。在本文经过分析模型图像处理步骤后得出结论：这是由于高斯模糊处理实际上是对图像进行了平滑处理[17]，它可以减少图片上的噪点和失真，等于给图像进行了滤波，过滤掉了部分高频噪声，使得算法性能表现更好。

5.5 遮挡物体识别失效

在此部分，本研究将针对遮挡物体识别失效这一失效模式进行验证。通过对 VOC2007 测试集图像添加遮挡物体以进行故障注入，观察目标检测算法在处理这些被遮挡图像时的表现。具体来说，通过读取图片并在图片上添加一个黑色矩形区域作为遮挡物，然后使用 SSD 模型来对进行遮挡处理后的测试集进行评估。表 5 为故障注入等级说明，其中 obstacle 参数为遮挡矩形的面积大小。

表 8 VOC2007 数据集图像遮挡故障注入等级描述

故障注入等级	故障参数	数据集失效表现
轻度	obstacle=150x70	遮挡面积较小，图像特征信息较为完整
中度	obstacle=180x90	遮挡面积中等，部分图像特征信息丢失
重度	obstacle=220x120	遮挡面积较大，大量图像特征信息丢失

表 9 遮挡物体识别失效影响

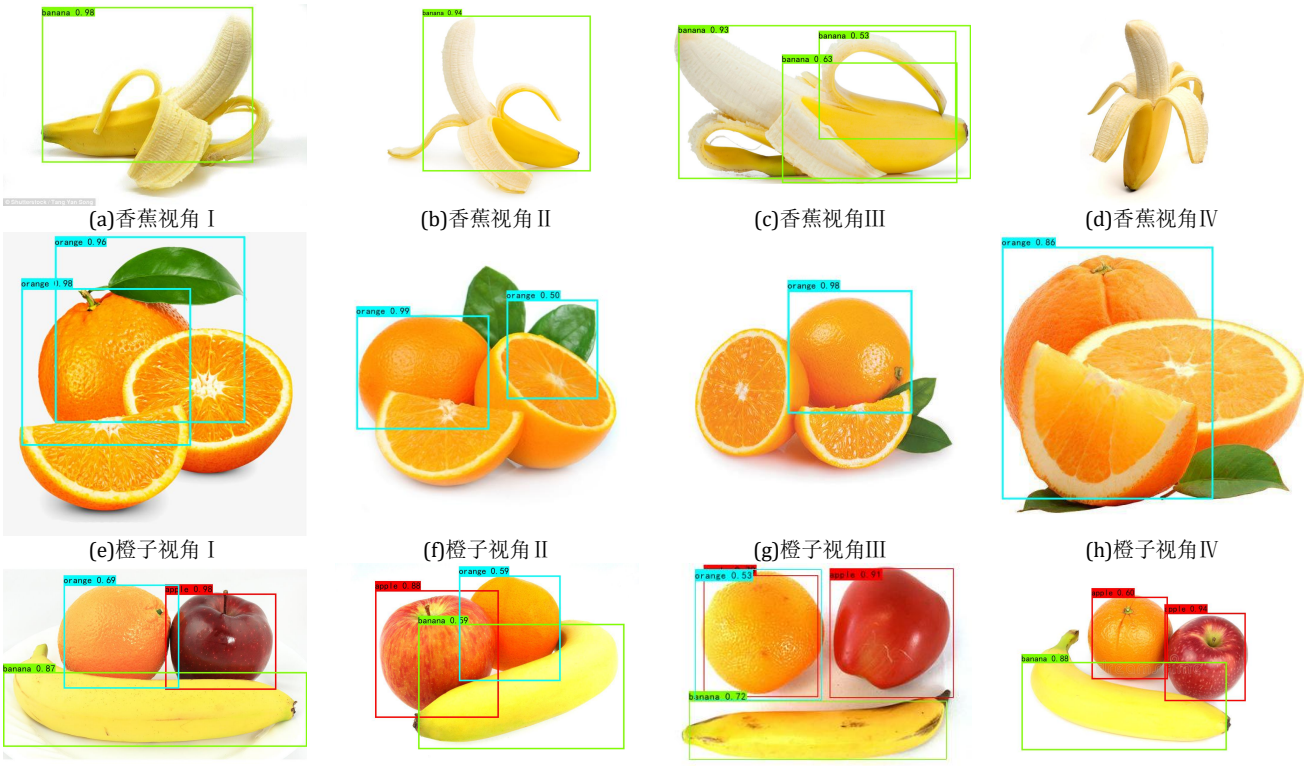
失效等级	mAP	mAP变化比例	失效影响
轻度	0.6181	-13.59%	目标检测置信度略有下降，算法性能略受影响
中度	0.5589	-22.19%	目标检测置信度明显下降，算法性能明显降低
重度	0.4511	-37.94%	目标检测失败，算法性能几乎完全丧失

实验对 VOC2007 测试集图像进行了三种程度的故障注入，即轻度、中度和重度遮挡，发现随着遮挡程度的加重，算法的表现出现了阶梯性的下降。在轻度遮挡下，mAP 的值从原始的 0.7153 下降到了

0.6181；在中度遮挡下，图像丢失部分特征，mAP 的值下降到了 0.5589；在重度遮挡下，图像丢失关键特征信息，mAP 的值下降到了 0.4511。这一结果清晰地说明了遮挡物体对目标检测系统识别能力的影响。

5.6 视角变化失效

在此小节中，本文将针对视角变化失效模式进行验证，实验选用了开源的 VOC 水果数据集，通过观察算法在多组同一物体不同视角的图像上的表现，明确了视角变化的存在会影响算法的性能。实验结果显示，当视角发生变化时，算法的识别结果出现了置信度下降、识别错误、无法识别等问题。



(a)香蕉视角 I　(b)香蕉视角 II　(c)香蕉视角III　(d)香蕉视角IV

(e)橙子视角 I　(f)橙子视角 II　(g)橙子视角III　(h)橙子视角IV

(i)混合水果视角 I　(j)混合水果视角 II　(k)混合水果视角III　(l)混合水果视角IV

图 6　视角变化对 SSD 模型性能影响

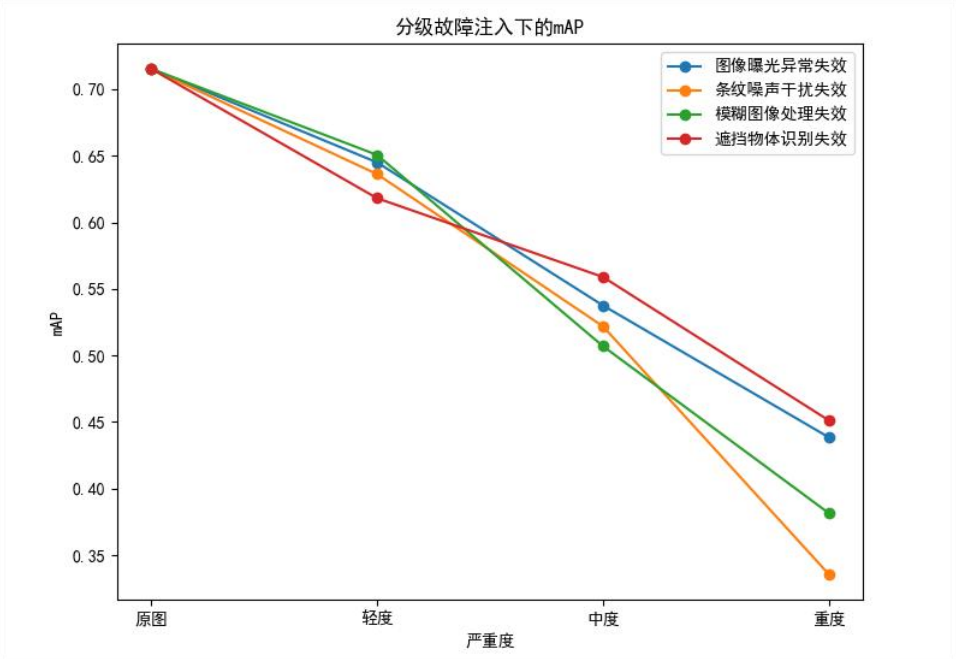


图 7　不同失效模式分级故障注入下的 mAP

5.7 实验结果展示

在对这几条失效模式进行验证后，为了更直观地表现失效严重程度和算法性能的负相关关系，绘制了根据不同失效模式和不同故障注入程度下的 SSD 算法 mAP 值变化的折线图，如图 7 所示。可以看出，随着失效程度的加深，mAP 的下降呈现一个很明显的下降趋势，这说明随着故障程度的加深，算法的性能越来越差，识别准确率也越来越低。

6. 结论

本论文验证了图像曝光异常失效等多种目标检测算法的失效模式的真实性和有效性。通过不同程度的故障注入的方式来模拟失效场景和失效严重程度，直观地观察了这些失效模式对模型性能的影响。实验结果表明，算法的失效程度和故障注入程度呈正相关关系，论文中提出的失效模式是具有普适性的，这说明本文所提出的失效模式定义是合理的，

这些验证结果揭示了目标检测算法的局限性，深化了对目标检测算法的运行失效的了解，为实际应用中更好地应对和处理这些失效场景提供了宝贵的参考。在未来的工作中，我们将进一步研究目标检测算法的失效模式，并对其进行验证。在此基础上，还可以通过深入研究和改进算法来提升其在复杂现实环境中的表现，为提升算法的鲁棒性铺平道路，从而进一步提升目标检测系统的整体性能。

参 考 文 献

[1] He S. Who is liable for the UBER self-driving crash? Analysis of the liability allocation and the regulatory model for autonomous vehicles[J]. Autonomous Vehicles: Business, Technology and Law, 2021: 93-111.

[2] Preim B, Botha C P. Visual computing for medicine: theory, algorithms, and applications[M]. Newnes, 2013.

[3] Andronie M, Lăzăroiu G, Karabolevski O L, et al. Remote Big Data Management Tools, Sensing and Computing Technologies, and Visual Perception and Environment Mapping Algorithms in the Internet of Robotic Things[J]. Electronics, 2023, 12(1): 22.

[4] Dubey A K, Jain V. A review of face recognition methods using deep learning network[J]. Journal of Information and optimization sciences, 2019, 40(2): 547-558.

[5] Secci F, Ceccarelli A. On failures of RGB cameras and their effects in autonomous driving applications[C]//2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2020: 13-24.

[6] Liu H C, Liu L, Liu N. Risk evaluation approaches in failure mode and effects analysis: A literature review[J]. Expert systems with applications, 2013, 40(2): 828-838.

[7] Damer N, Boutros F, Süßmilch M, et al. Masked face recognition: Human versus machine[J]. IET Biometrics, 2022, 11(5): 512-528.

[8] Zhang Yan-hong, Yi Zhi-hu. Research of Fault Injection in Missile Fault Diagnosis System[D]. , 2007.

[9] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11 – 14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016: 21-37.

[10] Everingham M, Winn J. The pascal visual object classes challenge 2007 (voc2007) development kit[J]. University of Leeds, Tech. Rep, 2007.

[11] Ibrahim H, Kong N S P. Brightness preserving dynamic histogram equalization for image contrast enhancement[J]. IEEE Transactions on Consumer Electronics, 2007, 53(4): 1752-1758.

[12] Wang Y, Jiang Z, Liu C, et al. Shedding light on images: Multi-level image brightness enhancement guided by arbitrary references[J]. Pattern Recognition, 2022, 131: 108867.

[13] Ping Tian D. A review on image feature extraction and representation techniques[J]. International Journal of Multimedia and Ubiquitous Engineering, 2013, 8(4): 385-396.

[14] Chang Y, Yan L, Wu T, et al. Remote sensing image stripe noise removal: From image decomposition perspective[J]. IEEE Transactions on Geoscience and Remote Sensing, 2016, 54(12): 7018-7031.

[15] Gedraite E S, Hadad M. Investigation on the effect of a Gaussian Blur in image filtering and segmentation[C]//Proceedings ELMAR-2011. IEEE, 2011: 393-396.

[16] Flusser J, Farokhi S, Höschl C, et al. Recognition of images degraded by Gaussian blur[J]. IEEE transactions on Image Processing, 2015, 25(2): 790-806.

[17] Lee J S. Digital image smoothing and the sigma filter[J]. Computer vision, graphics, and image processing, 1983, 24(2): 255-269.

Test Cost Reduction for VLSI Adaptive Test with K-Nearest Neighbor Classification Algorithm

Tai Song, Zhengfeng Huang, Qi Hong, Li Zhang, MILOS KRSTIC

***Abstract*—Testing of integrated circuits (ICs) is essential for weeding out defects before the products are shipped to customers. With the growing complexity of advanced nanoscale circuits, high-cost specification tests are becoming a bottleneck due to long test time. Reducing test time while maintaining product quality with low defect escape is crucial to optimize the test cost. Aiming at the problems of test time taking too long in the traditional test, *i.e.*, a test with full patterns, a valid test pattern (one that can make the test fail) selection method, called stepwise k-nearest neighbors (KNN), is presented in this paper. The test pattern selection approach utilizes a classification model to select valid patterns only, so as to detect defects quickly to save test time. Experimental results demonstrate a 11.2% improvement in the accuracy of the presented approach compared with regular KNN. Furthermore, the optimal compromise between test time and test escape (when faulty circuits are undetected) can be achieved by means of a polynomial regression function and can consequently convert the optimization problem into a mathematical function to achieve broader applicability and generality.**

***Index Terms*—VLSI test, machine learning (ML),test pattern, test escape, test cost.**

I. Introduction

CHIP test is an important task in production flows for ensuring ICs quality by minimizing defective parts per million (DPPM) [1]. A chip is considered good only if it passes all test patterns [2]. The number of transistors fabricated on a single chip is still increasing exponentially, and the continuous downscaling of transistors has improved the density and performance of chip, leading to tremendous growth in test time [3].

The number of test patterns determines the test time [4]. In traditional test approaches, for the number of test patterns the test response is compared to the expected one and the pass/fail result for each pattern is recorded. In order to reduce the test time the order of pattern execution plays significant role.

Fig. 1 gives an illustration of the pattern reordering method, in which each line consists of test pattern number when failure occurred, and the name of the output port at which the failure is detected. In traditional test, if the order of the patterns is not optimized the original sequence is applied to each circuit until it fails one test (in the example in the figure 4th, 8th or 9th). In this case, first three patterns are applied always regardless of low probability that circuit will fail. On the other hand,

Manuscript created July, 2023; revised December June, 2023. This work was supported by China Scholarship Council under the CSC No. 202206505003 and in part by the Cooperative Education Project of the Ministry of Education No.220803303162437 and in part by the National Natural Science Foundation of China under the Granted No. 62274052. This brief was recommended by Associate Editor *.*. (Corresponding author: Tai Song.

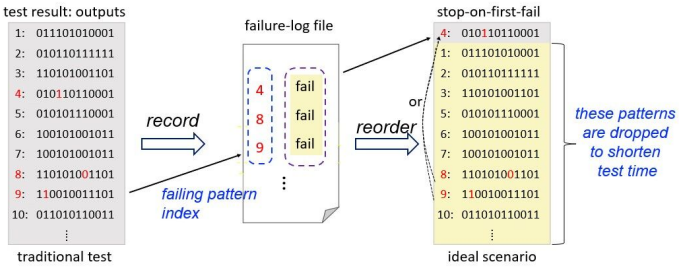


Fig. 1. Red denotes the ports whose output polarities are opposite to the expected values. The corresponding failing test-pattern indices are also marked in red.

if we reorder the patterns, and put expected failing patterns upfront, the test time could be dramatically reduced.

Consequently, traditional test approach is simple but resulting in increased test time due to execution of unordered test patterns [5]. There are several trends which can cause the test cost increase: (*i*) increase in transistor count that outstrip compression technology will increase the amount of test data which must be supplied to the Circuit Under Test (CUT, also called chips). Coupled with some irrelevant or redundant patterns exist in the test list, the overall effect will be longer test times [6]; (*ii*) Device configuration and one-time programming during test is causing more time to be spent during test to perform initial device calibrations or to reconfigure devices based on defects or electrical performance. As silicon geometries shrink and defect densities drive circuit redundancy, repair functions will also add to test costs; (*iii*) traditional test methods are largely manual and lack of automation in collecting failing data, and are only occasionally updated based on expert experience with the time scale of typically every few months; (*iv*) site count increases at probe test are not able to increase owing to the attendant increase in the cost of consumable material and the limitations of Touch-Down Efficiency; (*v*) the continuing increase of silicon content in automotive applications, especially for safety systems, drives additional test pattern insertions resulting more test time [7].

Adaptive test strategies are now being advocated. In [8-9], costly specification tests are performed only on sampling data and, consequently, are used to build spatial correlation models to predict invalid patterns so as to achieve test time reduction. P. Maxwell *et al*. [10] dynamically and cost-effectively tailored an IC test to discriminately manage variations in the manufacturing process. M. Grady *et al*. [11] presented two applications of pattern sampling for logic test. However, the above studies

require a careful selection of sample data, which may not be easily adopted by chip experts due to inaccuracy in prediction, which results in increased test escape. To address this problem, B. Xiao *et al*. [12] proposed a novel deep learning method to build a more accurate model, while M. Liu and M. Agrawal *et al*. [13-14] proposed a quality prediction model, and K. Wang *et al*. [15] proposed a wafer-level chip-scale package method. These methods pursued high accuracy in prediction and obtained low test escape but failed to achieve significant test time benefits.

In [16], C. Xue *et al*. proposed a test-reordering method for achieving a better diagnostic resolution for each failing chip. S. Inuyama *et al*. [17] proposed a pattern reordering method for the residual faults of the first small pattern set in order to create a one-pass approach to shorten CPU time and reduce demands on memory. However, these methods were not scalable due to the lack of a mathematical model, and the test patterns required human selection and domain knowledge from chip experts.

Motivated by the aforementioned apprehensions, a classification solution is presented herein for pattern selection. We determine whether any of the patterns should be skipped during the flow of the test program. Accordingly, the failing patterns are put forward and applied first, while those passing patterns are scheduled at the end of the test list. In this way, the test time can be reduced while test efficiency can be improved. The main contributions of this paper are shown as follows:

1) The unique classification strategies of ML provide effective automated approaches for handling complex and intensive data in very-large-scale integration (VLSI) testing.

2) Optimal patterns can be selected by means of polynomial regression curve and surface, and the optimization problem is converted into a mathematical function to give it broader applicability and generality.

The rest of the paper is structured as follows. In Section II, we review the background. In Section III, the proposed improved KNN algorithm is explored in detail. In Section IV, we describe the experimental setup and results, and the conclusions are drawn in last Section.

II. Background

A. Traditional testing process flow

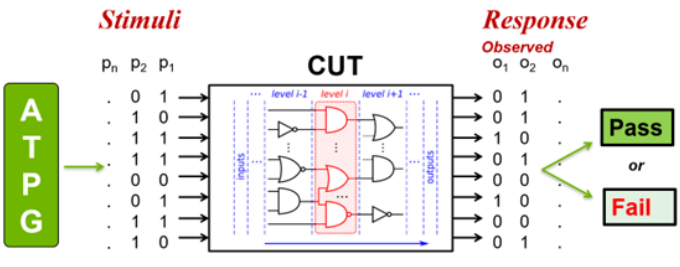


Fig. 2. Example of digital-based test process flow. The CUT can be determined as pass (good chip) when the corresponding ports provide output values are the same as expected values, or fail (bad chip) when the corresponding ports provide output values are different from the expected values.

Typically, the test result varies with the test pattern. Fig. 2 shows the different test results with the Automatic Test Pattern Generation (ATPG).

Fig. 2 provides a simplified overview of traditional test process in digital test. It can be seen that all patterns (*i.e*., from P_1 to P_n) are applied to the CUT and the output responses are recorded (*i.e*., from O_1 to O_n). The pass/fail outcomes of the CUT can be determined by the comparison results of expected value and observed value. Note: CUT will be determined as good chip only if all output responses (from O_1 to O_n) are identical to the expected values. On the contrary, the first failing pattern will be determined defect pattern.

B. Dynamics of Pattern Growth

As the industry moves towards smaller technology nodes, the standard logical fault models (*i.e*., stuck-at and transition) are no longer sufficient to detect some of the subtle defects that occur during manufacturing. In addition, the push towards zero defective parts-per-million, driven by some mission-critical applications such as automotive and medical, is driving faster adoption of new defect-oriented fault models (such as cell-aware model), across the industry. Cell-aware fault models are based on an abstraction of the defects that happen inside the cell boundaries at the transistor level rather than focusing on the defects that can be modeled on the logical representation of the library cell. Since the number of faults based on an abstraction of defects at the transistor level is much larger compared to the logic level, the pattern count required to detect such defects is usually higher. The combination of different fault models along with higher test coverage requirements is expected to continue pushing pattern counts higher. Increased pattern count and test time have been outpacing improvements in scan compression and increases in the scan shift rate. This problem will worsen as new fault models are introduced to reach enhanced quality goals, and the data volume may increase even beyond what is projected in our model.

III. Proposed Classification Method

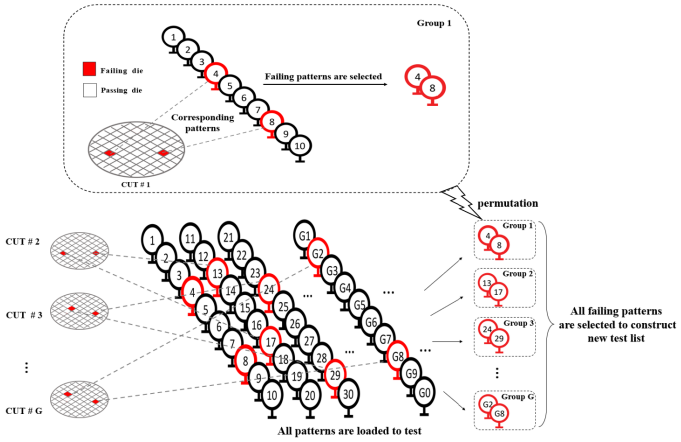


Fig. 3. Method to select the effective patterns, where the red grids in the CUTs represent the failing dies. The corresponding failing pattern numbers are also marked in red.

As the industry is grappling with how to reduce test time, new test strategies are being deployed for modeling cutting-edge neural networks for ML algorithms targeted for data centers, end-point applications. These methods pose new opportunities for testing of logic devices. To perform optimized pattern identification at a lower test cost, failing patterns that lead to fast defect could be selected based on the results of ML algorithm. In this paper we will focus on the use of KNN that could be the good choice.

Fig. 3 gives an illustration of the method to construct new test list from failing patterns, in which each group consists of all failing patterns when failure occurred, and the test pattern number at which the failure is detected. When a CUT generates multiple erroneous outputs, each of failing pattern is selected. For example, in Group *1*, all patterns are applied to each die in CUT#1 until it fails one test, in this case, the corresponding failing patterns (*i.e.*, 4th or 8th) are recorded. Similarly, all CUTs (*i.e.*, from #2 to #G) are tested by all patterns and the failing patterns are recorded to construct new test list, indicates the permutation is composed from Group *1* to Group *G*.

A. Data Processing: binary-to-decimal

KNN has the disadvantage to deal with binary pattern directly, due to the patterns are only composed of 0 and 1, leading it difficult to distinguish between defective and defect-free patterns. To select the pattern features so as to classification, the binary-to-decimal transformation for binary pattern is performed as shown in Fig. 4.

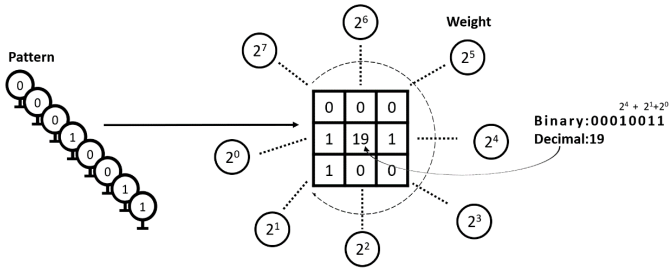


Fig. 4. Data encoding method: A 33 matrix is used to explain binary-to-decimal transformation. The filling of binary numbers is done from the top left corner, clockwise by sequentially arranging the binary values, to obtain an 8-bit long encoding. According to the corresponding weight, calculating the decimal value and filling it in the center of matrix.

For the sake of simplicity, only one 8-bit pattern is chosen to demonstrate in our application. An example of the pattern is composed of the sequence with 0,0,0,1,0,0,1,1. It can be seen that the content of pattern is only composed of 1 or 0, and this is not intuitive for kNN classification. Therefore, a 33 matrix (only for 8-bit data) is employed. The encoding is done from the top left corner, clockwise by sequentially to obtain an 8-bit long encoding. According to the fundamental conversion operation for 8421Binary Code Decimal (BCD) code, the decimal data can be obtained and filled in the center of matrix.

B. Description of KNN classification method

In order to clarify the proposed idea and how KNN is applied, all patterns are loaded and thus the test results (pass or fail) for each pattern are collected. Passing patterns are denoted by black while failing ones are in red, respectively.

Algorithm Description: In Fig. 5, the sample X is determined by its nearest label, via a majority vote. For clarity, passing patterns are denoted by black while failing ones are in red, respectively. The numbers inside each square indicates their decimal values. Given an unlabeled X, indicating untested pattern, is going to predict its class label by kNN algorithm. The algorithm searches the nearest k distances from X, indicating this distance-based approach can determine the class label of X. More specifically, X can be attributed to passing patterns due to the passing number $>$ the failing number (4 black squares (16,19,27,35) are more than 2 red squares (4,5) in the training circle).

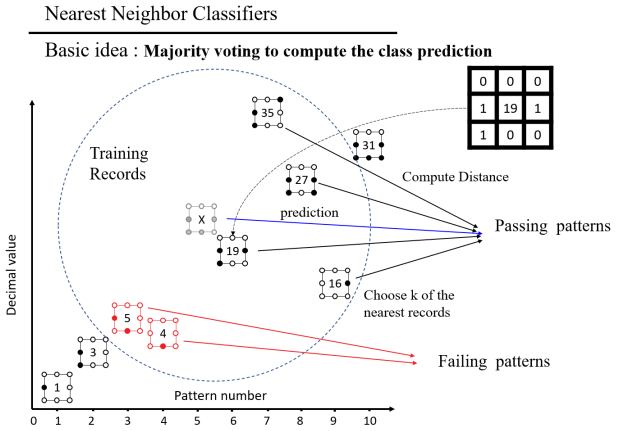


Fig. 5. Filled and empty circles correspond to bit values of 1 and 0 in 8-bit operator. The numbers inside each square correspond to their decimal values. Passing patterns are denoted by black while failing ones are in red, respectively.

IV. Experimental Results

A. Characteristics of the designs

We investigate test performance of proposed method on various benchmarks, CUT designs include IS-CAS85 [18], MCNC [19], and ISCAS89 [20]. To apply test on sequential circuits in the ISCAS89 benchmark, we unroll the circuit for two-time frames and convert it to a combinational one. Note that the unrolling process duplicates the combinational logic blocks, thus increasing the effective circuit size. To compare test performance of proposed method with other logic testing-based methods, we use accuracy as the metrics to quantify predict model. To characterize detection efficiency, we use the number of test patterns and the detection runtime as the metrics. We empirically show that the proposed method achieves a higher fault coverage with shorter runtime overhead compared to the counterparts in the rest of this section. To fully characterize the performance of our proposed method, we devise various defects for each circuit and insert defects to each benchmark.

B. Defects injection

In this section, we define a list of probable defects, *i.e.*, a defect dictionary, and report the number of defects that are

detected, given the selected test stimuli and test response, as shown in Fig. 6. To better mimic reality, a pool of defective circuits was generated by randomly injecting the defect dictionary into the CUT.

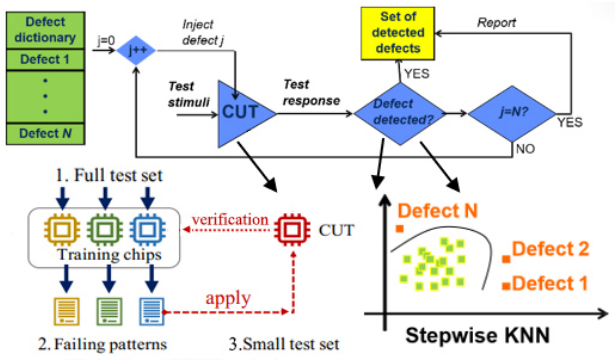


Fig. 6. Flowchart of the experiment.

C. Competing Methods

In this paper, we selected three supervised ML classification methods as the competing methods: traditional KNN [21], support vector machine (SVM) [22] and linear discriminant analysis (LDA) [23]:

1. K-Nearest-Neighbor: KNN is a classical classification method. Following the literature in [21], we set **K** from 1 to 20, and reported the best result.

2. SVM based on feature extraction [22]: This method maps the original t-dimension feature space to the new f-dimension feature space by using the transformation matrix W to obtain the optimal function so as to improve classification accuracy. Following the literature [22], we set the penalty parameter C =0.001 and gamma=0.5.

3. LDA with time labels [23]: The T-LDA model makes the parameter training more accurate and improves the performance of driving pattern recognition. Following the literature [23], we set the parameter alpha=0.5 and beta= 0.1.

D. Indices for Evaluation of Classification Performance

We use two indices to evaluate the performance of the classifiers.

1) Accuracy (*acc*): In this 2-class classification problem, let the test set contain m number of points. Suppose that the classifier correctly classifies m' of the test set members. Then, the accuracy is given by $acc = (m'/m)$. *acc* is used to evaluate performance in our experiments.

2) Running cost (RC): The number of test patterns influences the test duration because the number of patterns changes the test time. Accordingly, reducing the number can shorten the duration of the test, as shown in Fig. 7.

In this figure, the bars arranged on the horizontal axis are used to represent the durations of 8 tests, while the clock signs arranged there above are used to indicate the time slots corresponding to the test execution or skip. Thus, the T1 (1st test) clock indicates that the execution was done in the 1st time slot. In the case of the 1st device, the procedure begins

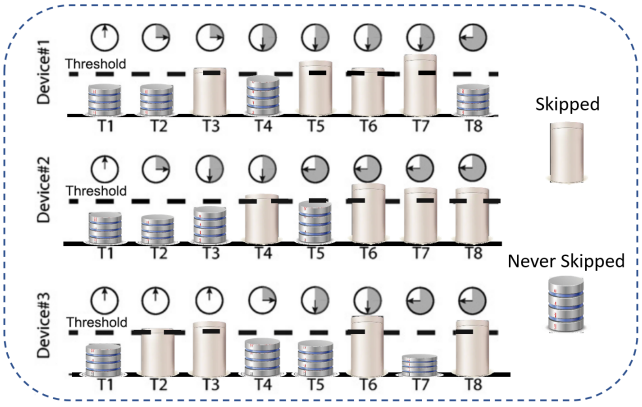


Fig. 7. Test time elimination illustration. Elimination process is demonstrated for three devices. Applied tests are selected based on the performance of the devices being tested.

with the execution of T1. The next step is the comparison of the T2 (2nd test) with the threshold level, and where the threshold is exceeded, this test is skipped. Note that the threshold depends on the mean failure rate of the test pattern. Where T2s likelihood falls below the threshold, the test is executed in the 2nd time slot. Thereafter, T3 shows a likelihood of going beyond a threshold, which is thus skipped. Subsequently, the succeeding tests are sought until a test kept below the threshold is found. In case T4 is less than the threshold, it is run in the 3rd time slot. Later on, the T5, T6, and T7 are skipped since they go beyond the threshold. The procedure continues until the test list is completed either by execution or skipping. For the example extension purpose, the test skipping is diagrammatic for devices #2 and #3 as well. Two crucial features of the algorithm are provided by this diagram. The first feature is the application of tests depending on the device's performance traits. The test is skipped when sufficient confidence has been attained for it. The second feature is that the pattern of test skipping varies from device to device because of diversity. Thus, with the help of AT higher-efficiency device adaptation and testing execution can be achieved.

E. Performance on Different Algorithms

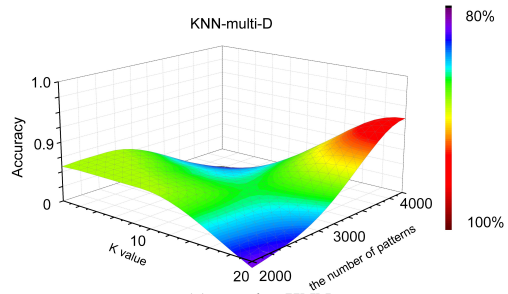
Classification tasks were conducted with four ML methods on the two types of datasets, which are defined below.

1) An Multi-dimensional dataset contains more than 2000 data patterns and has a data dimension greater than 45.

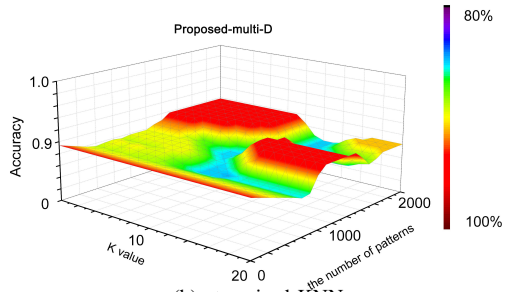
2) Imbalance datasets can be quantified in the form of an imbalance ratio (IR). IR is defined for a two-class dataset as the ratio of the number of points in the majority class to that of the minority class. Based on its IR value, the imbalanced data can be defined, where IR $>$ 3.5.

We conducted classification tasks with the regular KNN classifier, the SVM, the LDA and the stepwise KNN on the two datasets, and then reported the classification accuracy of regular KNN and stepwise KNN methods in Figs. 8 and 9, where the vertical axis indicates the classificaiton accuracy. We also listed the running cost (in time) of all methods in Fig 10, where the horizontal axis indicates the number of iterations

and the vertical axis represents the running cost. Moreover, the value of accuracy and the runing cost are listed in Table I.

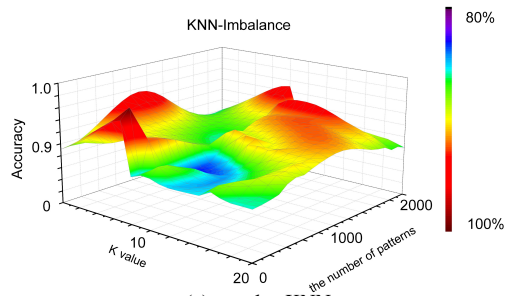


(a) regular KNN

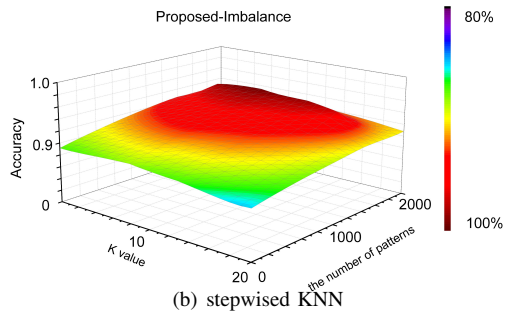


(b) stepwised KNN

Fig. 8. Accuracy of Multi-dimensional dataset.



(a) regular KNN



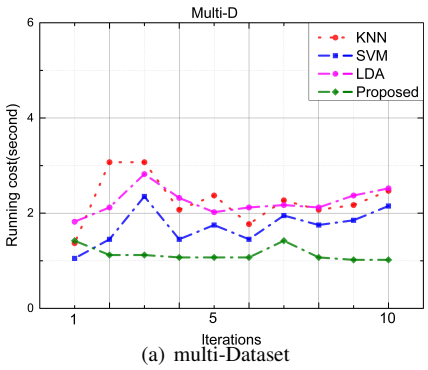
(b) stepwised KNN

Fig. 9. Accuracy of imbalance dataset.

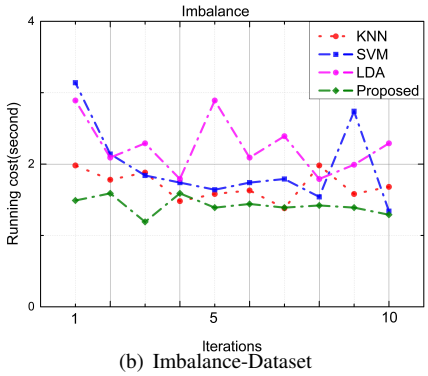
From Figs.8-10 (Note: only regular and stepwise KNN are shown due to the space limitation) and Table I, we can make two observations.

1) Comparison of algorithms in terms of accuracy

The proposed stepwise KNN method improved the classification accuracy by an average of 11.2% when compared to regular KNN, by 14% when compared to SVM and by



(a) multi-Dataset



(b) Imbalance-Dataset

Fig. 10. Running time of all methods on two dataset.

Table I. Comparison of Algorithms in Terms of Accuracy /RC

Dataset	KNN	SVM	LDA	**Proposed**
Multi-D	0.75/1.37	0.83/1.05	0.76/1.82	**0.89/1.02**
Imbalance	0.85/1.38	0.74/1.14	0.88/1.79	**0.89/1.19**
Average	**0.8/1.37**	**0.78/1.1**	**0.82/1.8**	**0.89/1.1**

8.5% when compared to LDA, on multi-D and imbalance datasets. In addition, stepwise KNN, in terms of the learning optimal-k-values for different samples, outperformed the other methods with a fixed predefined value, such as regular KNN. The results on the imbalanced datasets show that the proposed method achieved the best mean *acc*. The LDA algorithm had the second-best performance in terms of average *acc*. However, the regular KNN method exhibits lower *acc* on the multi-dimensional datasets, as is evident from the low *acc*. In addition, regular KNN failed to achieve good performance on the imbalanced datasets, as is evident from the fluctuations in accuracy, possibly due to insufficient learning of the **K**-computation, leading to erroneous prediction of the **K** values. These findings imply that the proposed classifiers outperform the state of the art on multi-D datasets due to the adaptive choice of **K** values. Stepwise KNN is also able to retain the good performance on the imbalance data sets, as is evident that the data processing method proposed in Section 3.1 is necessary.

2) Comparison of algorithms in terms of RC

We summarize the results of stepwise KNN and the other three competing algorithms over two datasets in terms of RC. The proposed stepwise method achieved the lowest RC, same as SVM, followed by regular KNN and LDA. An inspection of the results on the both datasets reveal that stepwise KNN

and SVM achieved the same mean RC. In the case of the imbalance datasets, stepwise KNN achieved the second best RC slightly lower than that of SVM. The reason is that the **K** computation method set a suitable **K** for a different training dataset to conduct KNN classification, leading to the time consumption. On the other hand, SVM found the optimal hyperplane to divide the entire dataset into several parts, *i.e.*, scanning only a subset of the training dataset. It is noteworthy that the regular KNN failed to achieve good performance on the multi-D datasets, possibly due to sufficient learning by the underlying feature, leading to a large computational burden.

The comparison between *acc* and RC (over 2 datasets) is summarized in average of the last row of Table II, and indicates that proposed method achieved the best compromise between accuracy and running cost. For example, although SVM and stepwise KNN both achieved the same RC, the stepwise KNN attaining the higher accuracy.

F. Searching Optimal Patterns for Multiple Sets of Data

This work further explores the characteristics for large amounts of data to enhance predictive performance. Accordingly, 100 sets of data that were composed of test time, test escape and test patterns were selected to generate a 3-D space, as shown in Fig 11. Similarly, the corresponding function:

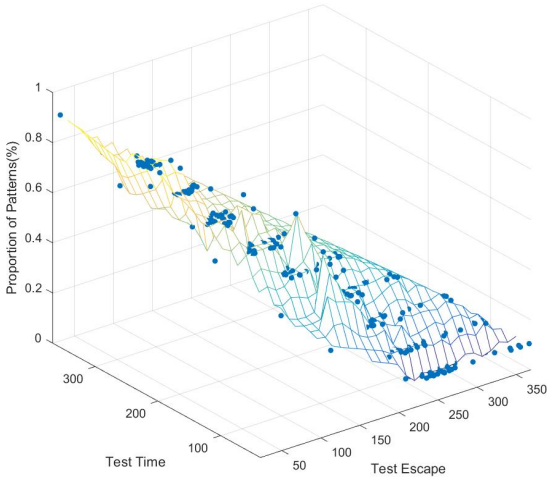


Fig. 11. Fitted surface generated from 100 sets of data.

$$Z = 0.011x^2 + 0.006xy + 0.001y^2 \\ -1.756x - 0.917y + 156.6 \quad (1)$$

where *Z*-axis denotes the ratio of patterns, and *X*-axis and *Y*-axis denotes the number of test escape and test time, respectively. In this way, the contributions of this work can determine the optimal test patterns according to mathematical function.

G. Limitation and Future Work

Note that our method is suitable for digital circuit testing. When it comes to deal with analog circuits and mixed-signal circuits, adjust threshold value method is necessary. More specifically, the Lower Spec. limit (LSL) and Upper Spec. limit (USL) as the measured criteria *(i.e.*, voltage or current) is greater than the USL or smaller than the LSL, which can make the test fail; otherwise, when the measured parameter value is between in the USL and the LSL, which can make the test pass. The threshold adjustment method will be explored in future work.

V. Conclusions

This work proposes an innovative method for reducing test cost, which selects the most effective test patterns with a stepwise KNN algorithm. The main advantage is that it selects the most effective test patterns to make the test stop quickly, and results show that the proposed method significantly improves the performance of classification compared with other methods. In addition, the proposed method can update the prediction model according to the statistical data and update the test patterns simultaneously, which confirms the method's adaptive ability.Finally, since AdaTest is a generic test pattern generation framework, we plan to investigate its performance on other hardware security problems such as logic verification and built-in self-test in our future work.

References

[1] S. Letchumanan, T. H. H. Tan, Y. P. Gan and S. L. Wong. Adaptive test method on production system-level testing (SLT) to optimize test cost, resources and defect parts per million (DPPM). In *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT).*, 2018, pp.1-3.

[2] Liang Huang, et al. VLSI test through an improved LDA classification algorithm for test cost reduction. In *Microelectronics Journal.*, 2022, PP.26-29.

[3] T. S. Valid test pattern identification for VLSI adaptive test. In *Integration,the VLSI.*, May 2021.

[4] H. Stratigopoulos and C. Streitwieser. Adaptive Test with Test Escape Estimation for Mixed-Signal ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, 37(10): 2125-2138.

[5] Song, T.et al. Chip test pattern reordering method using adaptive test to reduce cost for testing of ics. IEICE Electronics Express, 18(2) (2020), DOI:10.1587/elex.17.20200420.

[6] T Song, et al Machine learning classification algorithm for VLSI test cost reduction,In *Integration,the VLSI.*, september,2022.

[7] Song et al., "Pattern Reorder for Test Cost Reduction Through Improved SVMRANK Algorithm," in IEEE Access, vol. 8, pp. 147965-147972, 2020, doi: 10.1109/ACCESS.2020.3016039.

[8] W. Zhang et al. Virtual probe: A statistical framework for low cost silicon characterization of nanoscale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2011, 30(12): 1814-1827.

[9] N. Kupp, K. Huang, J. M. Carulli, and Y. Makris. Spatial correlation modeling for probe test cost reduction in RF devices. In *in Proc. IEEE/ACM Int. Conf.*, 2012, pp.23-29.

[10] P. Maxwell. Adaptive Testing: Dealing with Process Variability. *IEEE Design & Test of Computers*, 2011, 28(6): 41-49.

[11] M. Grady, B. Pepper, J. Patch, M. Degregorio and P. Nigh. Adaptive testing - Cost reduction through test pattern sampling. In *IEEE International Test Conference (ITC).*, 2013, pp.1-8.

[12] B. Xiao, J. Xiong and Y. Shi. Novel applications of deep learning hidden features for adaptive testing. In *21st Asia and South Pacific Design Automation Conference (ASP-DAC).*, 2016, pp.743-748.

[13] M. Liu, R. Pan, F. Ye, X. Li, K. Chakrabarty and X. Gu. Fine-Grained Adaptive Testing Based on Quality Prediction. In *IEEE International Test Conference (ITC).*, 2018, pp.1-10.

[14] M. Agrawal and K. Chakrabarty. Test-Cost Modeling and Optimal Test-Flow Selection of 3-D-Stacked ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, 34(9): 1523-1536.

[15] K. Wang et al. Test Cost Reduction Methodology for InFO Wafer-Level Chip-Scale Package. in *IEEE Design & Test*, 2017, 34(3): 50-58.

[16] C. Xue and R. D. Blanton. Test-set reordering for improving diagnosability. In *IEEE 35th VLSI Test Symposium (VTS).*, 2017, pp.1-6.

[17] S. Inuyama, M. Arai and K. Iwasaki. Critical-Area-Aware Test Pattern Generation and Reordering. In *IEEE 25th Asian Test Symposium (ATS).*, 2016, pp.191-196.

[18] Mark C Hansen, Hakan Yalcin, and John P Hayes. 1999. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering.In *IEEE Design Test of Computers 16*, 3 (1999), 72-80.

[19] Theodore W. Manikas. June 28, 2012. MCNC Benchmark Netlists.

[20] Franc Brglez, David Bryan, and Krzysztof Kozminski. September 22, 2006. ISCAS89 Benchmark Netlists.

[21] U. Lall and A. Sharma. A nearest neighbor bootstrap for resampling hydrologic time series. In *Water Resour. Res*, 1996, 32(3): PP.679-693.

[22] J. Li, Y. Zhao, X. Gao and W. Zhao. Feature Extraction and SVM Parameters Optimization Based on ICQPSO. In *2020 International Conference on Computer Network, Electronic and Automation (ICCNEA).*, 2020, pp.329-332.

[23] L. Xie, Y. Shi and Z. Li. Driving Pattern Recognition Based on Improved LDA Model. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS).*, 2018, pp.320-324.

Low-latency and Reconfigurable Hardware Accelerator for Complex Functions Using Approximate Rotation CORDIC-Based Parallel Method

Yi Wang , Ming-Da Zhang , Yu-Hang Ji , Yao-Hua Xu , Na Bai ,
Chen-Xiao Mao and Ding Zhou

***Abstract*—Advanced SoC chip designs need to accelerate many complex functions such as multiplication, division, and trigonometric functions. Designing a separate DSP processor for each function would result in a large footprint, inflexibility, and high latency. Therefore it would make sense to aggregate the various functions in one hardware accelerator. In order to address the lack of capability of traditional embedded processors for specific application scenarios, this paper proposes a hardware accelerator for on-chip SoC systems that provides performance, efficiency and latency advantages for specific tasks. By analyzing the mathematical principles, we propose an improvement on the traditional cordic algorithm and propose the ARC (Approximate Rotation Cordic) algorithm to solve the difficulties that may exist in the original algorithm such as high latency or inflexible architecture that is difficult to scale. A reconfigurable architecture is then designed to implement them. Then we perform reconfigurable architecture design to implement them. The designed reconfigurable hardware accelerator implements a variety of complex function operations with small area and low power consumption. In TSMC 28nm process, it is reported that the area is 0.1511 mm^2 , power consumption is 68.2mW, and frequency is 2Ghz, which is suitable for integration in SoC design.**

***Index Terms*—SoC, accelerator, trigonometric functions, reconfigurable architecture**

I. INTRODUCTION

Hardware accelerators take advantage of the fast computing characteristics of hardware by replacing software algorithms with hardware modules that do not require intensive instruction access and can greatly ease the storage bandwidth of on-chip soc systems. At the same time, accelerators can be dynamically algorithmically adapted to the application or specific task. An example is the SPARCNet (SPARse Convolutional NETwork) a DSP-oriented accelerator[1] suitable for accelerating FFT (Fast Fourier Transform) algorithms, which involve multiplication. Another example is the RNN (Recurrent Neural Network)[2], DNN (Deep Neural Network)[3] and CNN (Convolutional Neural Network)[4] accelerators for artificial intelligence algorithms, suitable for accelerating neural network operations, which include activation functions.

The mixture of DSP and AI algorithms has led to the development of artificial intelligence and the birth of many new applications in various fields, such as infrared spectroscopy systems in the medical field[5]. This requires a combination of "FFT+RNN", including multiplication, open-square operations and division. Another example is inverse kinematics for robotic arm control systems in robotics[6], which requires a combination of FIR (Finite Impulse Response) and CNN algorithms, including trigonometric functions, inverse trigonometric functions, etc. Complex functions need to be adapted to specific architectures. In order to reduce the design difficulty and integration density of emerging applications, this paper proposes a hardware accelerator that can be embedded in soc systems as an IP core.

So far, there are many methods used to implement complex functions, mainly (I) ROM look-up table method, (II) polynomial fitting, (III) polynomial fitting based on the look-up table method, and (IV) Digit-by-Digit algorithm[7]. There are many problems with the above methods, such as (III) being suitable for small data widths and not for double precision numbers; (II) using the Taylor series requires multiplication operations and consumes many resources for hardware implementation[8]; the look-up table based polynomial fitting algorithm implements a complex circuit for the chip and is too costly in terms of area; and (IV) the main drawback is the slow linear convergence rate[9].

The CORDIC algorithm requires no hardware multiplier, only shift and addition operations, and does not cover much space in hardware, so it has the advantage of high speed. The algorithm requires several iterations and therefore also has a

The manuscript was submitted on June 6, 2023. This research was Supported by the Open Research Fund of National Engineering Research Center for Agro-Ecological Big Data Analysis & Application, Anhui University (No. AE202214). (Corresponding author: Yao-Hua Xu.)

Na Bai and Yao-Hua Xu are with the Information Materials and Intelligent Sensing Laboratory of Anhui Province, Anhui University, Hefei, Anhui 230601, China(e-mail: 11074@ahu.edu.cn; xyh@ahu.edu.cn)

Yi Wang, Ming-da Zhang, Yu-Hang Ji , Chen-Xiao Mao and Ding Zhou are with the Information Materials and Intelligent Sensing Laboratory of Anhui Province and National Engineering Research Center for Agro-Ecological Big Data Analysis & Application, Anhui University, Hefei, Anhui 230601, China(e-mail:yiwang@ahu.edu.cn;wb22301062@stu.ahu.edu.cn;p20301119@stu.ahu.edu.cn;p20301196@stu.ahu.edu.cn;wb22301109@stu.ahu.edu.cn)

long delay. To solve this problem, this paper improves the original algorithm and proposes an ARC algorithm that can effectively reduce the number of iterations. Our hardware accelerator is designed with a reconfigurable architecture to implement a variety of complex functions using the ARC algorithm. The main contributions made in this paper are as follows.

(1) In Section 3 we propose an ARC algorithm based on the basic features of the traditional CORDIC algorithm, with improvements. This algorithm effectively reduces the number of iterations of the original algorithm and reduces the latency. Then, several complex functions, such as division, and atan2, are implemented according to the newly proposed algorithm. The advantages of the new algorithm are demonstrated.

(2) We design a new reconfigurable hardware accelerator based on the newly proposed ARC algorithm, with high performance and more flexibility.

(3) We demonstrate the feasibility of the method by giving the relationship between the accuracy distribution and the number of iterations through circuit simulation and comparison. We then complete the hardware implementation using the TSMC 28nm process and report that our proposed reconfigurable hardware accelerator has low latency, small area and low power consumption.

Section II of this paper reviews the principles of the traditional CORDIC algorithm and areas for improvement; Section III details the optimised ARC algorithm used in the designed hardware accelerator and the implementation of various functions using the optimised algorithm; Section IV focuses on the composition and internal structure of the designed hardware architecture; Section V analyses and evaluates the simulation results and the synthesis results of the hardware accelerator; and Section VI is a summary and outlook.

II. Overview of The Cordic Algorithm

The two modes of operation of the traditional CORDIC algorithm are as follows.

(1) Rotation: in this mode, the new vector coordinates are in three modes running the CORDIC algorithm. linear, circular, and hyperbolic, with each of the three running in vector mode or rotation mode [10]. The vector mode enables conversion from Cartesian to Polar coordinates, the rotation of the input vector to the X-axis, and the storage of the rotation angle to be accomplished. In the rotation mode it is possible to realize the conversion from polar to right-angle coordinates (x_{end}, y_{end}) is obtained by rotating the vector with coordinates (x_0, y_0) by θ_i angle.

(2) Vector: This paper describes the vector mode in detail. The coordinates P_1 are rotated until the vertical coordinates converge to 0, i.e., the starting and final coordinates exist, and then the angular difference between the vectors is determined.

The most basic rotation and transformation equations:

$$X_{i+1} = X_i \cos\theta_i - Y_i \sin\theta_i \quad (1)$$

$$Y_{i+1} = Y_i \cos\theta_i + X_i \sin\theta_i \quad (2)$$

Extracting $\cos\theta_i$ from Eq(1),(2). such that $K_i = \cos\theta_i$, the equation changes to

$$X_{i+1} = K_i(X_i - Y_i \tan\theta_i) \quad (3)$$

$$Y_{i+1} = K_i(Y_i + X_i \tan\theta_i) \quad (4)$$

θ_i denotes the angle of each rotation. To simplify the above equation, take $\theta_i = \tan^{-1} 2^{-i}$, i is the iteration index indicating each iteration. The advantage of this assumption is that the angle θ_i of each rotation can be varied according to the law of $\tan^{-1} 2^{-i}$,each variation is $1/2 = 2^{-1}$, $1/4 = 2^{-2}$, $1/8 = 2^{-3}$, $1/16 = 2^{-4}$ ....., called micro-rotation [11], which can be achieved by shifting right in binary.

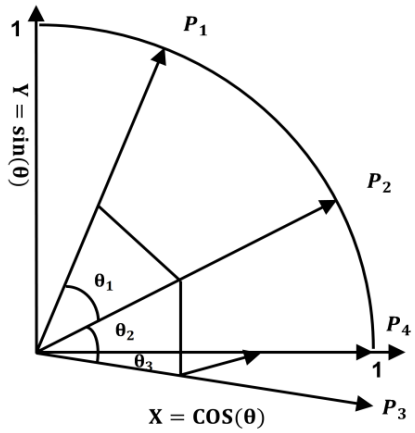


Fig. 1. Illustration of vectoring mode

The equation is changed by extracting K_i separately as follows.

$$X_{i+1} = X_i - \delta_i Y_i 2^{-i} \quad (5)$$

$$Y_{i+1} = Y_i + \delta_i X_i 2^{-i} \quad (6)$$

$$\theta' = \theta - \delta_i \tan^{-1} 2^{-i} \quad (7)$$

Here δ_i is the direction of rotation, θ is the sum of angles already rotated, and θ' is the sum of angles after adding the current rotation angle. δ_i takes positive or negative values to determine the direction of rotation clockwise and counterclockwise, which can be consulted in Table I. The magnitude of each rotation angle will get the value of $\tan^{-1} 2^{-i}$ from the LUT table, as shown in Table II. If Y_i is greater than 0, then δ_i is negative (-1), theθvalue queried in the LUT table should be added to θ_i. Assuming that Y_i is less than 0, then δ_i is positive (+1), the corresponding θ value should be subtracted from z_i [12].

K_i denotes the i-th rotation modulus length compensation factor, which is given by $\theta_i = \tan^{-1} 2^{-i}$.

$$K_i = \cos\theta_i = \cos\arctan\left(\frac{1}{2^i}\right) = \frac{1}{\sqrt{1+2^{-2i}}} \quad (8)$$

The above equation simplifies K_i by taking values that are only related to the index value i and are independent of the rotation direction and angle values. If the total number of rotations is N, the total mode length compensation factor K can be expressed as: $K = \prod_{i=0}^{N-1}\left(\frac{1}{\sqrt{1+2^{-2i}}}\right) = (0.7071) \times (0.8944) \times (0.9701) \times (0.9923) \times (0.9981) \times (1) \times (1) \times \cdots \approx 0.6073$

The traditional CORDIC algorithm angle iteration range is shown in Table II.

$$\begin{cases} \theta_{max} = \sum_{i=0}^{\infty} \tan^{-1} 2^{-i} \approx 99.8829° \\ \theta_{min} = \sum_{i=0}^{\infty} -\tan^{-1} 2^{-i} \approx -99.8829° \end{cases} \quad (9)$$

TABLE I
ITERATIVE FORMULA OF CORDIC ALGORITHM

Mode	Vectoring $Y_i \to 0$ $d_i = \begin{cases} +1, Y_i \le 0 \\ -1, Y_i > 0 \end{cases}$	Rotation $\theta' \to 0$ $d_i = \begin{cases} +1, \theta' < 0 \\ -1, \theta' \ge 0 \end{cases}$
Hyperbolic $\theta_i = \tanh^{-1} 2^{-i}$ (i=1,2,3,4,4, ……,N)	$X_n = (X_0^2 - Y_0^2)^{\frac{1}{2}}/K$ $\theta' = \theta + \tanh^{-1}(Y_0/X_0)$	$X_n = [X_0 \cosh\theta_0 + Y_0 \sinh\theta_0]/K$ $Y_n = [Y_0 \cosh\theta_0 + X_0 \sinh\theta_0]/K$
Linear $\theta_i = 2^{-i}$ (i=1,2,3,……,N)	$X_n = X_0$ $\theta' = \theta + Y_0/X_0$	$X_n = X_0$ $Y_n = Y_0 + X_0\theta$
Circular $\theta_i = \tan^{-1} 2^{-i}$ (i=1,2,3,……,N)	$X_n = (X_0^2 + Y_0^2)^{\frac{1}{2}}/K$ $\theta' = \theta + \tan^{-1}(Y_0/X_0)$	$X_n = [X_0 \cos\theta_0 - Y_0 \sin\theta_0]/K$ $Y_n = [Y_0 \cos\theta_0 + X_0 \sin\theta_0]/K$

Note: In Hyperbolic mode, i={4,13,40,…,k,3k+1}repeat

Therefore, the traditional CORDIC algorithm has disadvantages such as limited convergence range and too many iterative layers while implementing a variety of functions. This increases the resource consumption of hardware and limits the data processing capability. To address the shortcomings of the traditional CORDIC algorithm, the coprocessor designed in this paper proposes and adopts an ARC (Approximate Rotation Cordic) algorithm. First, the rotation range is converged in the range of $[-54.827°, 54.827°]$ by using numerical transformation; next, partial rotation is completed by using the traditional CORDIC algorithm; then, the final level of rotation direction is determined by using the approximate rotation method; finally, the phase rotation completes the remaining iterations.

TABLE II
LOOK UP TABLE(LUT)

Iteration number i	Inverse tangent 2^{-i}	Rotation angle $\tan^{-1} 2^{-i}$
0	2^{-0}	45.0°
1	2^{-1}	26.5651°
2	2^{-2}	14.0362°
3	2^{-3}	7.1250°
4	2^{-4}	3.5763°
5	2^{-5}	1.7899°
6	2^{-6}	0.8952°
7	2^{-7}	0.4476°
8	2^{-8}	0.2238°
9	2^{-9}	0.1119°
…	…	…

III. CONSTRUCTION OF AN NOVEL CORDIC-BASED METHOD

A. Digital Transformation Method

Two common methods used to address the problem of limited angle coverage are iterative iteration and mathematical conversion. Repeat iteration is the process of iterating through special iteration units to achieve the most effective angle coverage. The traditional CORDIC algorithm is extended to cover the entire circumference of the circle by adding two iterations of 45°. Additionally, the iterative rotation sequence is extended from i=0,1,2,...,N-1 to i=0,0,0,1,2,...,N-1. In the pipeline structure, adding one iteration requires three adders and two subtractors. Since the pipeline structure is not suitable for repeated iterations, we opt for a more appropriate digital transformation method.

The mathematical transformation uses mathematical transformation relations between functions to uniquely map the input data to the angle coverage of the traditional CORDIC algorithm $[-99.8829°, 99.8829°]$, indirectly extending the range of convergence. To this end, as shown in Table II, this paper divides the entire circumference of four regions, each spanning a consistent area, and transform the input from the entire circumference of quadrants of the convergence range by means of trigonometric transformation relations. In each domain, the angle of the corresponding region can be redefined by another angle. For example, let us consider the angle $\frac{2\pi}{3}$, which ranges in the interval $\left(\frac{\pi}{4}, \frac{3\pi}{4}\right)$. This implies that the angle falls within region II. To proceed with the CORDIC rotation step, we apply a preprocessing transformation $\varphi = \theta - \frac{\pi}{2}$. As

a result, $\varphi = \frac{\pi}{6}$, which falls within the required convergence range of CORDIC algorithm, $[-54.827°,\ 54.827°]$.

TABLE III
THE PROPOSED ZONING METHODOLOGY

region	Range	Angle redefinition
I	$(-\frac{\pi}{4}, \frac{\pi}{4}]$	$\varphi = \theta$
II	$(\frac{\pi}{4}, \frac{3\pi}{4}]$	$\varphi = \theta - \frac{\pi}{2}$
III	$(-\frac{3\pi}{4}, -\frac{\pi}{4}]$	$\varphi = \theta - \pi$
IV	$[-\pi, -\frac{3\pi}{4}] \cup [\frac{3\pi}{4}, \pi]$	$\varphi = \theta - \frac{3\pi}{2}$

The proposed regional division method Table III shows the division method of the four regions. Based on Table II angle redefinition, the data onto regions II, III and IV are converted to region I. For the redefined angle can be expressed as:

$$\begin{bmatrix} x_{\mathrm{I}} \\ y_{\mathrm{I}} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = Q \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$
$$\begin{bmatrix} x_{\mathrm{II}} \\ y_{\mathrm{II}} \end{bmatrix} = \begin{bmatrix} -\sin\varphi & -\cos\varphi \\ \cos\varphi & -\sin\varphi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = Q \begin{bmatrix} x_0 \\ -y_0 \end{bmatrix}^{\mathrm{T}}$$
$$\begin{bmatrix} x_{\mathrm{III}} \\ y_{\mathrm{III}} \end{bmatrix} = \begin{bmatrix} -\cos\varphi & \sin\varphi \\ -\sin\varphi & -\cos\varphi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = Q \begin{bmatrix} -x_0 \\ -y_0 \end{bmatrix}$$
$$\begin{bmatrix} x_{\mathrm{IV}} \\ y_{\mathrm{IV}} \end{bmatrix} = \begin{bmatrix} \sin\varphi & \cos\varphi \\ -\cos\varphi & \sin\varphi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = Q \begin{bmatrix} -x_0 \\ y_0 \end{bmatrix}^{\mathrm{T}} \quad (10)$$

The rotation mode is to convert the input angle to region I, which is $[-45°,\ 45°]$; the vector mode is to convert the input vector to within region I. Obviously, the I region is within the convergence range of CORDIC algorithm.

In order to further reduce the resource consumption, this design cancels the first level iteration, which is 45° rotation. This step transformation changes the convergence domain of CORDIC algorithm to $[-54.827°,\ 54.827°]$, which also contains I quadrant range. After the first level iteration is discarded, the iteration sequence becomes i=1,2,3,...,N-1.

As the input data is transformed, the output data has to be recovered according to the corresponding regional interest and restored to the corresponding region. The data is recovered using the same inverse of the mathematical transformation. The mathematical transformation reduces one level of iterations, and the overall reduction of 3 levels of iterations compared to repeated iterations makes the mathematical transformation more applicable to the pipeline structure.

B. Approximate Rotation Method

To address the problem of excessive number of iterations, the Hybrid CORDIC algorithm is proposed in [13], which combines a hybrid structure of two-step selection to effectively reduce the number of iterations. However, this approach comes with the disadvantage of serious loss of computational accuracy. Another optimized algorithm is the ARC algorithm, which reduces the number of iterations without significantly increasing the resource consumption. The details of the optimized ARC algorithm will be described below.

When the initial value is (x_0, y_0, z_0) after m (m<n) iterations, the coordinate value to become (x_m, y_m, z_m). The phase of rotation of the m+1st iteration is $\theta_m = \tan^{-1} 2^{-m}$. Using the Taylor series expansion θ_m, we have.

$$\theta_m = \tan^{-1}\frac{y_m}{x_m} = \frac{y_m}{x_m} - \frac{1}{3}\left(\frac{y_m}{x_m}\right)^3 + \cdots \quad (11)$$

Also ensuring a phase accuracy of 2^{-n}, according to Eq. (11) yields

$$\frac{1}{3}\left(\frac{y_m}{x_m}\right)^3 < 2^{-n} \quad (12)$$

Since $\tan^{-1} 2^{-m} < \tan^{-1} 2^{-m+1}$,then there is that

$$\theta_m < \tan^{-1} 2^{-m+1} \quad (13)$$

Then, from equation (13), we get that

$$\frac{y_m}{x_m} < 2^{-m+1} \quad (14)$$

According to the formula (14), then we have

$$m > \frac{n}{3} - \log_2\left(3^{\frac{1}{3}} + 1\right) \approx \frac{n}{3} + 0.47 \quad (15)$$

At this point the remaining rotational phase values can be approximated as $\frac{y_m}{x_m}$. The vector pattern of the CORDIC algorithm can be rewritten as

$$\begin{cases} x_n = x_m[1 + \left(\frac{y_m}{x_m}\right)^2]^{1/2} \\ z_n = z_m + \tan^{-1}(\frac{y_m}{x_m}) \end{cases} \quad (16)$$

Since when $\tan^{-1}(\frac{y_m}{x_m})$ is sufficiently small, there is $\tan^{-1}(\frac{y_m}{x_m}) \approx \frac{y_m}{x_m}$. Thus, after the mth iteration then we have

$$\begin{cases} x_n = x_m[1 + \left(\frac{y_m}{x_m}\right)^2]^{1/2} \\ z_n = z_m + \frac{y_m}{x_m} \end{cases} \quad (17)$$

Since the expression of Eq. (17) still requires high complexity of the hardware, if the inequality $\frac{1}{2}\left(\frac{y_m}{x_m}\right)^3 < 2^{-n}$ holds, to guarantee the phase accuracy of n bits, when $m > \frac{n+1}{2}$ holds, $x_m \approx x_n$.

Based on the above conclusions, the one-approximation rotation CORDIC algorithm can streamline the set of rotational phases $\{\tan^{-1} 2^0,\ \tan^{-1} 2^{-1},\ \ldots \tan^{-1} 2^n\}$ to $\left\{\tan^{-1} 2^0,\ \tan^{-1} 2^{-1},\ \ldots \tan^{-1} 2^{-\left\lceil\frac{n+1}{2}\right\rceil},\ y_m\right\}$, where $\lceil\blacksquare\rceil$ stands for upward rounding. y_m stands for the last level of rotation, from which the direction of rotation is directly obtained. Compared to the original CORDIC algorithm, the number of rotations is reduced by $n - \left\lceil\frac{n+1}{2}\right\rceil - 1$.

C. Complex Function Implementation

Based on the introduction in the previous subsection, the approximate rotation method by his nature can obtain a specific output from a specific input. Table III lists some functions that can be obtained using only one ARC algorithm, such as assigning θ_0 to X, and X_0 and Y_0 to the values of K and 0, respectively. The delayed values of X_n and Y_n are the desired sine and cosine values.

TABLE IV
IMPLEMENTATION OF MULTIPLE FUNCTION FUNCTIONS

Type	ARC used	Configuration
sinX	Circular Rotation	$X_0 = K, Y_0 = 0, \theta_0 = X$
cosX	Circular Rotation	$X_0 = K, Y_0 = 0, \theta_0 = X$
$\tan^{-1}(\frac{Y}{X})$	Circular Vector	$X_0 = X, Y_0 = Y, \theta_0 = 0$
X÷Y	Linear Vector	$X_0 = X, Y_0 = Y, \theta_0 = 0$
$\sqrt{X}$	Hyperbolic	$X_0 = X + 1, Y_0 = Y - 1,$
	Vector	$\theta_0 = 0$
quad(Y, X)	Circular Vector Linear	$X_0 = X, Y_0 = Y, \theta_0 = 0$

When using the circumferential system for iterative operations in vector mode it is possible to find the arctangent value. However, according to Table II, the angle iteration range is $[-99.8829°, 99.8829°]$, i.e. not all angles can be covered. Therefore, the key to solving this problem is to induce an auxiliary function (Q,R)=quad(Y,X) to indirectly implement the atan2(Y,X) function.

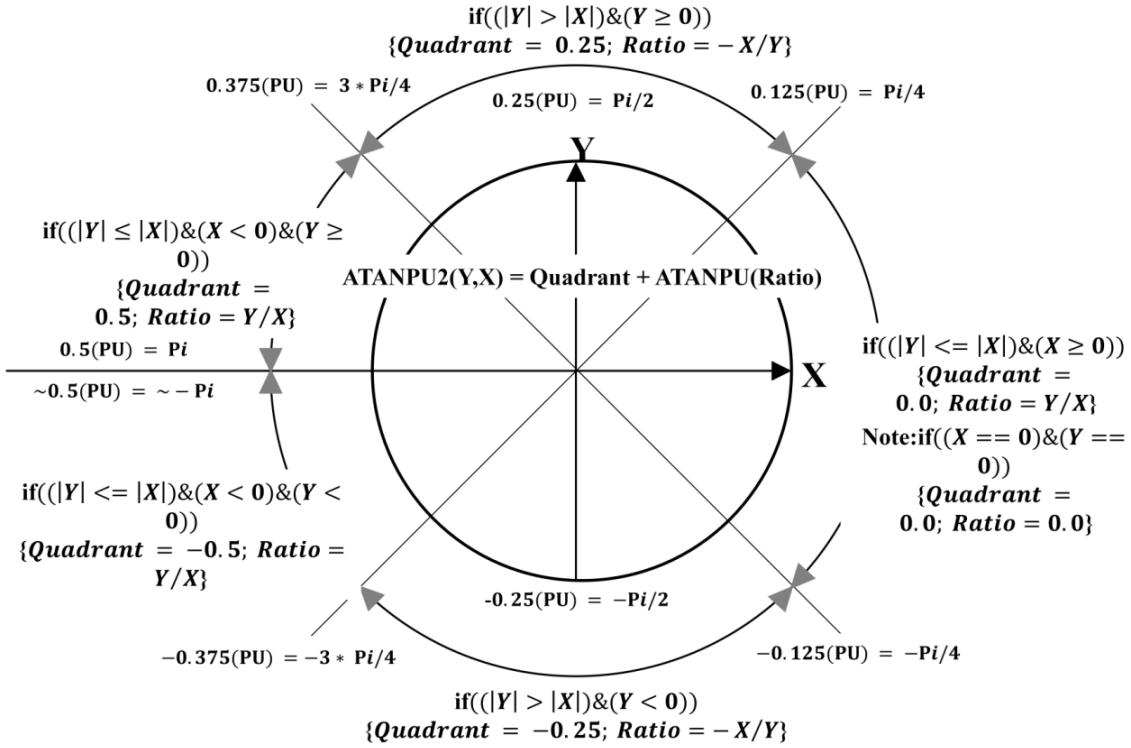


Fig.2. Quad Function Schematic

The Quad(X,Y) function inputs coordinates for a point in the coordinate system. Based on the basic trigonometric formula, the positive and negative signs of X and Y and the relationship between |X|, |Y|, we can map the result of atan(R) to atan2(Y,X). quad(X,Y) has two main functions, the first is to calculate the quadrant offset Q based on the position of the input coordinates (X,Y) (shown in the principle reference figure 2), and the second is to iterate through the vector pattern of the linear system The operation calculates the ratio R.

$$R = \begin{cases} \frac{Y}{X}, \text{if } |Y| \le |X| \\ \frac{X}{Y}, \text{if } |Y| > |X| \end{cases} \quad (18)$$

Thus far, the iterative formula for atan2(Y,X) can be obtained as follows:

$$\text{atan2}(Y, X) = \text{Quadrant} + \text{atan (Ratio)} \quad (19)$$

IV. HARDWARE REALIZATION

A. Top-level Design

This section focuses on the hardware architecture of the designed hardware accelerator. The hardware accelerator can be composed of three sub-modules, namely the pre-processing module, the RC (Reconfigurable Controller) module, the execution module and the post-processing module. The top-level module structure diagram is shown in Figure 3.

As can be seen from the diagram, the pre-processing module performs a distribution operation on the incoming data onto the module, first checks the legality of the input. If the input data is illegal, the exception signals Data_in_vld = 1 is entered and the final output result data_out is selected in preference to the exception signal output. If the input data is legal, then the data continues to be converted from floating point of fixed point. The result of the pre-processing is passed into the execution module for iterative operation according to the operation

selected by the RC module. The post-processing section to perform the fixed-point to floating-point conversion to the results passed in by the execution module to ensure consistency between input and output.

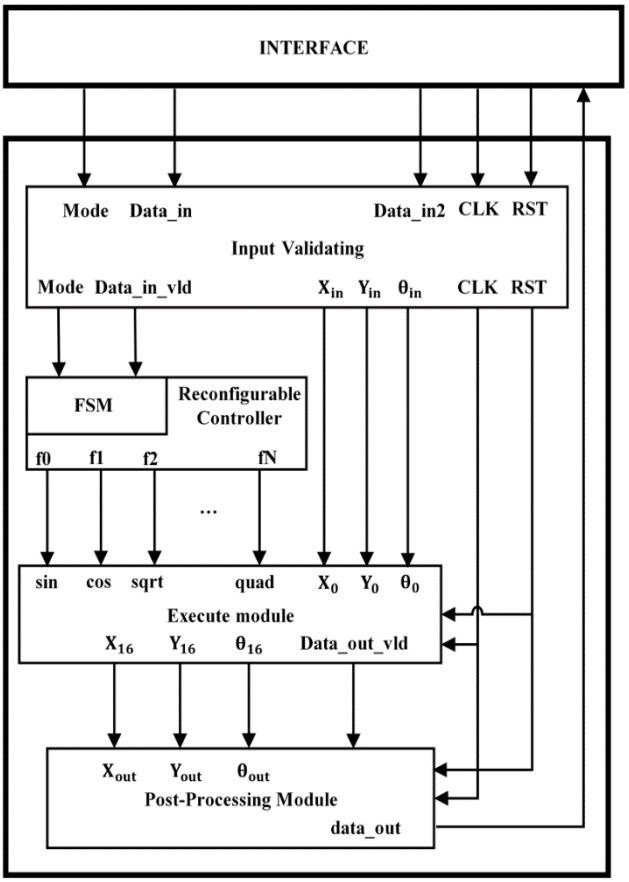


Fig.3. Top floor plans

B. Pre-processing Module

i. Detection Blocks

The pre-processing module firstly sets the input values within the range that the CORDIC algorithm can calculate, and secondly converts the input values of floating point of fixed point form, as shown in Figure 4. The inputs and outputs of this design are designed as 32-bit numbers.

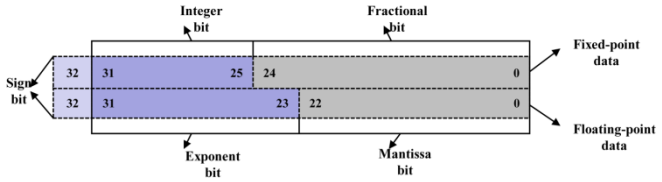


Fig.4. The composition of floating-point and fixed-point data

Sign indicates the sign bit of a floating point number, Exponent indicates the exponent bit of a floating point number, and Mantissa indicates the mantissa bit of a floating point number. The detection of a floating point number will output an exception value if the following conditions occur.

When the floating point number is in the correct format, if $1 \le E \le 8'hff$ and $M \ne 0$, the floating point number can be converted to a fixed point number and passed into the core module as a valid input. In addition to this, exception values are still output when.

•If mode=2 is set and Data_in<0, the exception value 2 is output.

•If mode=3 is set and |Data_in|>1, an exception value of 4 is output.

That is, determine whether the input value is within the range of the CORDIC algorithm operation. If mode=2 is set for open root operation, the input cannot be negative. If mode=3 is set, the input range of the atan function operation is [-1, 1], and if the range is exceeded, mode=4 should be set, i.e. the atan2 function operation is performed.

ii. Conversion Blocks

The digital conversion module is divided into the angle determination and angle conversion functions, with the flow shown in Figure 5. The first step is to determine the zone to which the input angle belongs, generating the control logic; comparing the input angle with the boundary value of each zone would consume a large number of comparators and the latency would increase. According to Table V, each interval boundary value has a different high three digit, so the high three digits can be used to determine the interval to which it belongs. In flowchart 5, MSB(θ) denote the top three bits of the θ angle and MSB(m_n) denotes the top three bits of the boundary value.

The second step are to calculate the value of the corresponding angle φ in the region I where θ is located; this sequential execution strategy saves hardware resources, but increases the delay of the one-level full adder. With algorithm execution times as a priority, this design takes two steps of parallel execution to pre-compute the value of angle φ at the cost of an additional subtractor, followed by data selection, which reduces the delay of the one-stage subtractor.

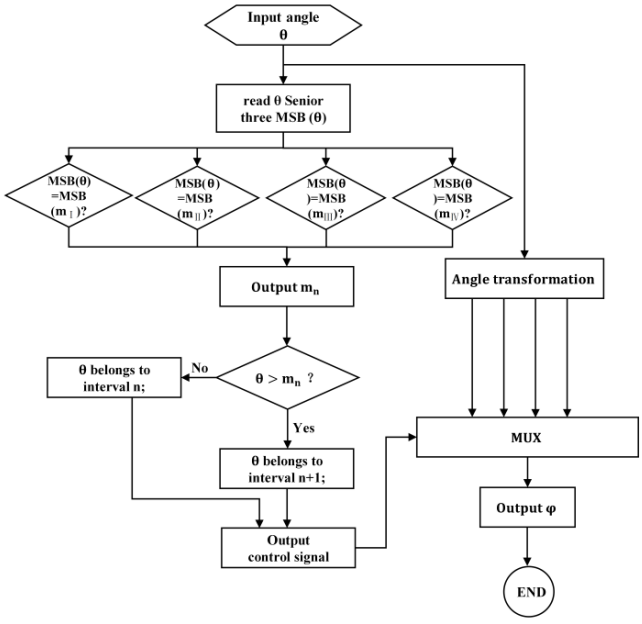


Fig.5. Conversion flow chart

TABLE V
HIGH THREE INTERVAL JUDGEMENT IN THE CONVERSION MODULE

region	Boundaries (M_I, M_{II}, M_{III}, M_{IV})	Binary high three digits
I	$\pi/4$	000
II	$3\pi/4$	010
III	$5\pi/4$	011
IV	$7\pi/4$	101

C. Reconfigurable Controller Module

The RC (Reconfigurable Controller) module is divided into an FSM (Finite State Machine) and various operator controllers. The FSM specifies an operator controller to process the incoming data, and according to the predetermined configuration, determines how the input data should be processed. The FSM also distributes the output data that has been processed by the previous level. Each operator controller has a different control process, which is determined by the algorithm described in section III-C above.

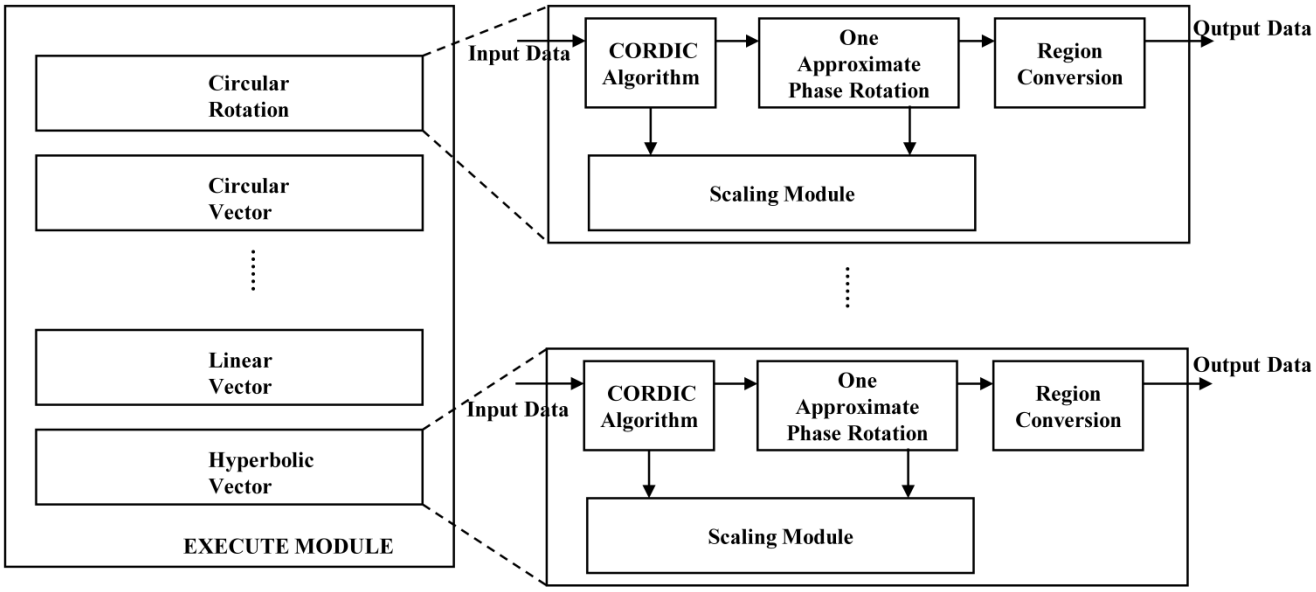


Fig.6. Schematic representation of the data transfer of the execution module

D. Implementation Module

The schematic diagram of the execution module is shown in Figure 7. The data is pre-processed and turned into data that can be operated by the execution module; the execution module then performs the operation and finally, the calculated values are passed out in a logical order by means of area conversion.

The ARC algorithm first performs N rotations of the CORDIC algorithm and then performs an approximate phase rotation to complete the remaining angle iterations. The newly proposed algorithm is therefore a combination of a traditional architecture and a new architecture. The existing architectures of the CORDIC algorithm include:

- Serial architecture: The serial iterative architecture is implemented using a micro architecture combining adders, some dedicated registers, variable shifters and ALUs (Arithmetic Logic Units) [14]. Only one level of CORDIC arithmetic iteration unit is designed, and the system clock is a time-division multiplex. This approach takes up the least amount of resources, but the timing control is cumbersome and not suitable for high speed processing of data.
- Parallel architecture: The parallel architecture uses n identical arithmetic units to allow each iteration to be performed simultaneously, thus eliminate the need for control circuitry and allowing shift, add and subtract operations to be performed via shift registers and add/subtractors [15]. For the same number of iterations, parallel architectures operate more frequently than serial architectures. The significant advantage of the parallel architecture is the high throughput, which is more advantageous when processing cascaded inputs [16].

Considering the pursuit of timeliness, this design chooses a parallel architecture of flowing registers to increase the speed while saving resources. The hardware designs block diagram of the algorithm is shown in Figure 7.

The data output for the completed iteration requires a region recovery as shown in Table VI, where x and y are the final level outputs of the CORDIC iterative operation.

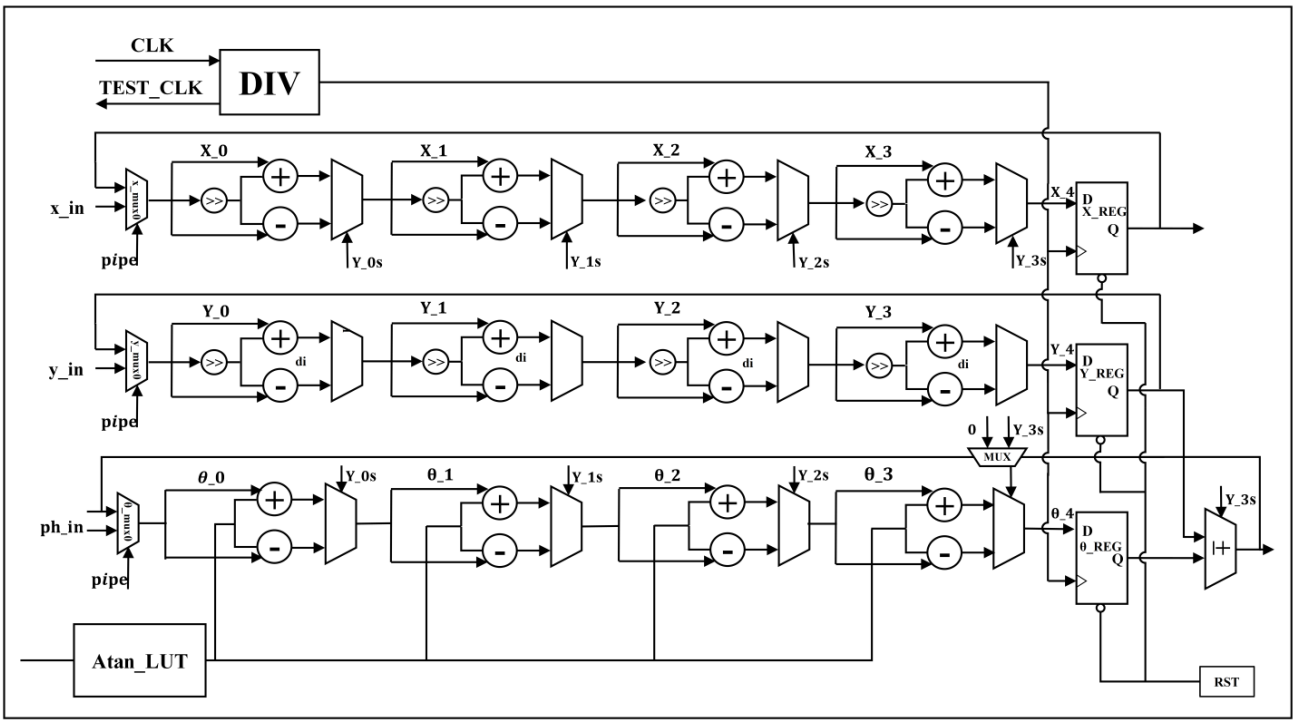


Fig.7. Execution module for streamlined structures

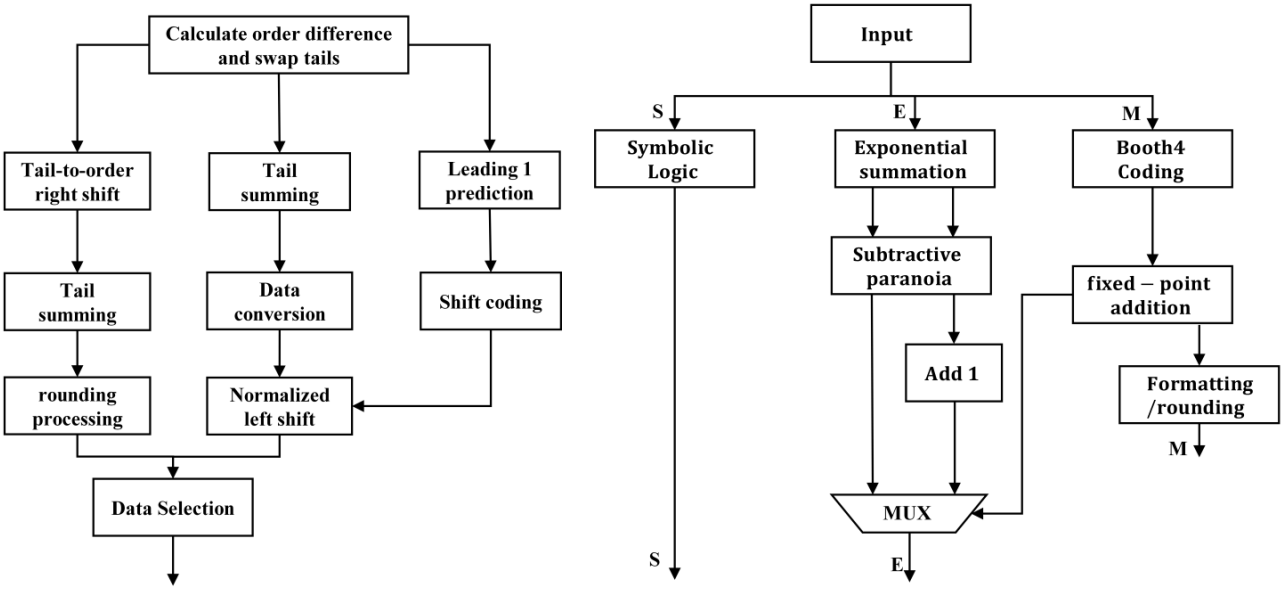


Fig .8. Stages of Floating Multiplier and Addition

E. Post-processing Module

The post-processing module contains a floating point adder, a floating point multiplier and a floating point normalisation module. The data to be converged is passed to the post-processing module, where the normalisation module performs floating point leading zero detection and splicing of sign, exponent and trailing bits to normalise the result.

All multipliers and adders adhere to the IEEE 754 single precision floating point standard. The steps involved in the adder and multiplier are presented in Figure 8. To reduce latency, parallelism and complex multiplex are used to introduce longer combinatorial paths into the normalisation phase.

V. Results and Comparisons

A. Functional simulation and error analysis

In this section, we verified the authenticity of the algorithm from the perspective of software simulation by implementing the hardware structure shown in Figure 8. In this paper, some random simulation results of each function are extracted as shown in Table VII. The corresponding function matlab scripts are written to compare the experimental results and error judgments, and the simulation results are as expected. Combined with Table VII, it can be found that the error average is accurate to five decimal places when performing Atan2 operation, and the accuracy reaches nine decimal places when square root operation, which is because the error is eliminated under the hyperbolic system of ARC algorithm. Figure 9 shows the accuracy distribution (average error) of the ARC algorithm when the iterations are varied from 14 to 20, so we conclude that the hardware design is essentially the same as the software results.

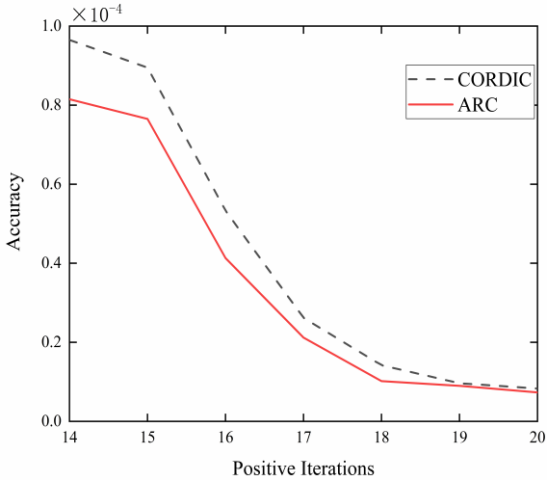


Fig.9. Accuracy distribution corresponding to different positive iterations

B. Modularity and Time-Series Analysis

When we did the power analysis using Prime-Timing and RedHawk , we did path, power and latency detection for the input module, legality detection module, core iteration module and post-processing module. The results in Table VIII show that the core iteration module has the longest critical path, the highest path latency, and the highest power consumption. The high latency of the core module is mainly due to the fact that the clock cycles necessary to query the Atan_LUT table in the x and y paths occupy some resources. The high power consumption is due to the fact that the floating-point normalization module contains numerous complex combinational logic circuits.

TABLE VII
Phase Error Comparison

MODE	Input data	ARC	CORDIC
0:sin	0.125	0.707106781	0.707106781
	0.1875	0.923879533	0.923879533
1:cos	0.125	0.707106781	0.707106781
	0.1875	0.382683432	0.382683432
2:arctan	0.25	0.244978663	0.244978663
	0.375	0.35877067	0.35877067
3:sqrt	0.0000001	0.000316228	0.000316228
	0.06	0.244948974	0.244948974
4: quad	$\binom{Y}{X}$ 1.23	$\binom{Q}{R}$ 0.25	-0.8419416
	$\binom{Y}{X}$ -1.12	$\binom{Q}{R}$ 0.910552979	
	$\binom{Y}{X}$ -1.3	$\binom{Q}{R}$ 0	0.982793723
	$\binom{Y}{X}$ 1.5	$\binom{Q}{R}$ -0.866668701	

TABLE VIII
Module Analysis of Coprocessors

Block	Path(ns)	Power(mW)	Latency
Input	2.303	9.653	3
Check	2.965	14.325	3
core	3.325	84.002	8
Post-Processing	3.035	46.332	5

The timing analysis is divided into two parts: the maximum operation frequency and the latency of the instructions. The maximum operation frequency is determined by the longest critical path, as shown in Table VII, the longest critical path of the core module is 3.3325 ns, which corresponds to a maximum frequency of 400 MHz. Table IX presents the latency and power consumption of each instruction in the core module in different modes. The latency cycle varies between 21 and 26, since most of the computations require four iterations, each of which takes four cycles, plus the latency of the rotation compensation factor.

TABLE IX
Time Allowance and Power Analysis of Different Modes

Instruction	Power(mW)	Latency
Sin and cos	1.3698	21
Arctan	1.3741	23
Square root	1.3652	23
quad	1.4562	26

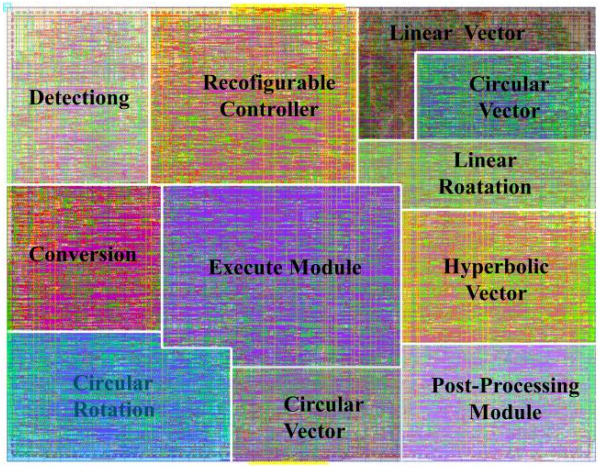
C. Physical design

In this section, we have used Synopsys Design Compiler synthesis and IC Compiler II for the physical design, synthesized using TSMC's 28nm process, and the chip specifications and module distribution are shown in Figure 10. The synthesis results are compared with the hardware accelerator from [9] and the traditional floating-point arithmetic hardware accelerator designed using the FP32-CORDIC algorithm. The hardware accelerator described in reference [9] is reported to have a power consumption of 97mW and occupies a chip area of 0.1876 mm^2. In contrast, the proposed design achieves a lower power consumption of 68.2 mW and a reduced core area of 0.1511 mm^2. Comparing the proposed design with the Nth root hardware accelerator [9],

significant improvements are observed, including a 50% reduction in area consumption and a 15% reduction in power consumption. These results demonstrate the superior performance and efficiency of the proposed design in comparison to the referenced hardware accelerator. Table X and Table XI show the area and power consumption of this design compared to the hardware accelerator designed by the FP32-CORDIC algorithm. The results show that the ARC algorithm-based hardware accelerator saves at least 40% of the basic logic resources and triggers, and the area and power consumption are 3.5 times and 4 times better than the hardware accelerator designed by the FP32-CORDIC algorithm, respectively, compared to the hardware accelerator designed by the FP32-CORDIC algorithm. The hardware accelerator described in reference [17] is a 16-bit fully parallel approximate CORDIC design. The proposed design, with its longer bit length, offers higher accuracy, while the incorporation of the reconfigurable design provides additional functionality.

TABLE X
COMPARISON OF HARDWARE RESOURCE USAGE

Type	logical resources	trigger	shift register	Iterations	Normalized delay time
ARC	7266	1214	3	16	0.6659
FP32 CORDIC	12748	2035	3	23	1



Technology	**TSMC 28nm**
Supply Voltage	**0.81V**
Frequency	**2Ghz**
Core Size	**0.1511mm^2**
Power	**68.20mW**

Fig.10. Layout design and chip specifications

TABLE XI
PERFORMANCE COMPARISON OF DIFFERENT ARCHITECTURES

Floating-point representation		ARC algorithm	FP32 CORDIC
Area(mm^2)	Combinational Area	0.2826	1.543
	Non-combinational area	0.0495	0.368
	Total	0.3321	1.911
Power(mW)	Static	99.55	324.40
	Dynamic	36.10	169.76
	Total	135.65	494.16

D. Outlook

The multiplicity of application scenarios has led to the gradual replacement of single-function algorithm accelerators by accelerators with complex functions. Our hardware accelerators are designed to cope with complex application scenarios and are suitable for algorithm acceleration in multiple domains. Examples include FFT, FIR and other DSP algorithms.

Nowadays, large-scale integration is rapidly driving hardware-based digital signal processing. The hardware accelerator technology may be applied to ultrasound diagnostic systems [18] to enable the transition from analog to digital ultrasound diagnostic instruments, where digital processing effectively avoids the introduction of noise in later processing. In addition to being applied to medical ultrasound images, it can also be applied, for example, to direct frequency synthesis techniques [19] for digital modulation and coding of

speech/music synthesis and communication, using trigonometric operations; for example, kinematic forward and inverse operations for robot manipulation, using the arctan2 algorithm; and for example, the phase discriminator in navigation SoC chips, the frequency discriminator function [20], using arctan, division, multiplication, sign excitation functions, etc.

In addition to the application level, the research in the algorithm itself is also a major feature. the Cordic algorithm, which first appeared in 1959, got rid of the multiplier and progressed step by step toward lower latency, higher accuracy, and fewer devices. We have reduced the number of iterations and the latency time without reducing functionality or accuracy. This gives the algorithm the flexibility to change the input range, which is not possible with many approaches. We also illustrate with real-world data that the designed accelerator has a small hardware overhead compared to accelerators based on common architectures. In addition to this, we can add more complex functions such as activation functions, exponential functions, and logarithmic functions as required by the application. All these are within the range of ARC algorithm open execution, only the design of RC module and execution module need to be changed, but the architecture will not be changed, which is the advantage of this accelerator.

VI. Conclusion

In this paper, we propose a hardware accelerator implemented using an improved CORDIC algorithm. In the implementation, we optimize the original algorithm using the advantages and disadvantages of the CORDIC algorithm and propose the ARC algorithm, where adding one level of approximate rotation to the original algorithm can effectively reduce the number of iterations and delay time. The proposed reconfigurable architecture has the advantages of flexible scalability, tunable input range and accuracy.

With the hardware implementation, we evaluate the required area and power consumption. Combining multiple sets of random test analysis and multi-functional testing, experiments show that sine, cosine, inverse tangent, open root sign, and quad function operations can be implemented. The input data is represented using 32-bit numbers and the angles are assigned accordingly, since the combination 2^32 = 4294967295 is possible, so a resolution of 360°/4294967295 = 8.3819e-8° is available (as in TABLE III). The iterations can be started by assigning the values of X, Y, and Z. The algorithm adheres to the floating point standard IEEE-754 in order to provide the correct sign for the values in each quadrant. The proposed custom algorithm involves only shift registers, adders and lookup tables (ROM). Minimal use of gate devices in the hardware implementation is ensured. The above features offer various advantages in terms of cost, circuit design complexity, power consumption and area compared to processors such as DSP multipliers.

In the future, we will continue to build on the work completed so far by integrating more complex functions with similar features and continuously optimizing algorithms to improve comprehensive performance.

References

[1] Leon V, Paparouni T, Petrongonas E, et al. Improving power of DSP and CNN hardware accelerators using approximate floating-point multipliers[J]. ACM Transactions on Embedded Computing Systems (TECS), 2021, 20(5): 1-21.

[2] Mittal S, Umesh S. A survey on hardware accelerators and optimization techniques for RNNs[J]. Journal of Systems Architecture, 2021, 112: 101839.

[3] Latotzke C, Gemmeke T. Efficiency versus accuracy: a review of design techniques for DNN hardware accelerators[J]. IEEE Access, 2021, 9: 9785-9799.

[4] Li J, Jiang S, Gong S, et al. SqueezeFlow: A sparse CNN accelerator exploiting concise convolution rules[J]. IEEE Transactions on Computers, 2019, 68(11): 1663-1677.

[5] Chen C, Ma Z, Liu Z, et al. An Energy-Efficient Wearable Functional Near-infrared Spectroscopy System Employing Dual-level Adaptive Sampling Technique[J]. IEEE Transactions on Biomedical Circuits and Systems, 2022, 16(1): 119-128.

[6] De Dinechin F, Istoan M. Hardware implementations of fixed-point Atan2[C]//2015 IEEE 22nd Symposium on Computer Arithmetic. IEEE, 2015: 34-41.

[7] Volder J E. The CORDIC trigonometric computing technique[J]. IRE Transactions on electronic computers, 1959 (3): 330-334.

[8] Volder J. The CORDIC computing technique[C]//Papers presented at the the March 3-5, 1959, western joint computer conference. 1959: 257-261.

[9] Juang T B, Hsiao S F, Tsai M Y. Para-CORDIC: Parallel CORDIC rotation algorithm[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2004, 51(8): 1515-1524.

[10] Xue Y, Ma Z. Design and implementation of an efficient modified CORDIC algorithm[C]//2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP). IEEE, 2019: 480-484.

[11] Tiwari V, Khare N. Hardware implementation of neural network with Sigmoidal activation functions using CORDIC[J]. Microprocessors and Microsystems, 2015, 39(6): 373-381.

[12] Heidarpour M, Ahmadi A, Rashidzadeh R. A CORDIC based digital hardware for adaptive exponential integrate and fire neuron[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2016, 63(11): 1986-1996.

[13] Shukla R, Ray K C. Low latency hybrid CORDIC algorithm[J]. IEEE Transactions on Computers, 2013, 63(12): 3066-3078.

[14] Anas M M, Padiyar R S, Boban A S. Implementation of cordic algorithm and design of high speed cordic algorithm[C]//2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). IEEE, 2017: 1278-1281.

[15] I. Nascimento, R. Jardim, F.Morgado-Dias. A new solution to the hyperbolic tangent implementation in hardware: polynomial modeling of the fractional exponential part[C]. NEURAL COMPUTING & APPLICATIONS, 2013, 2(23):363-369.

[16] Kumar P S, Vatsalkumar P, Dolui S, et al. Design of digital architecture for custom implementation of cordic algorithm[C]//2021 International Conference on System, Computation, Automation and Networking (ICSCAN). IEEE, 2021: 1-6.

[17] Chen L, Lombardi F, Han J, et al. A fully parallel approximate CORDIC design[C]//2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH). IEEE, 2016: 197-202.

[18] C. Chen et al., "An Energy-Efficient Wearable Functional Near-infrared Spectroscopy System Employing Dual-level Adaptive Sampling Technique," in IEEE Transactions on Biomedical Circuits and Systems, vol. 16, no. 1, pp. 119-128, Feb. 2022, doi: 10.1109/TBCAS.2022.3149766.

[19] Y. SU, Z. KUANG, G. ZHANG and L. SUN, "A 14-bit DDS Circuit Design Based on CORDIC Algorithm," 2019 IEEE International Conference of Intelligent Applied Systems on Engineering (ICIASE), 2019, pp. 321-323, doi: 10.1109/ICIASE45644.2019.9074119.

[20] V. Vijay et al., "Implementation of Low Complexity Signal Tracking Loop of a GPS Receiver Using CORDIC Algorithm," 2021 International Conference on Advances in Computing and Communications (ICACC), 2021, pp. 1-6, doi: 10.1109/ICACC-202152719.2021.9708191.

Yi Wang is currently working in the Anhui University. His research interests include communication integrated circuit system and mobile communication.

Mingda Zhang is currently a graduate student in the School of Integrated Circuits at Anhui University. His research interests include digital integrated circuits.

Yuhang Ji received the M.S. degree in the School of Integrated Circuits at Anhui University, in 2023. His research interests include digital integrated circuits and SOC chip design.

Yaohua Xu is currently an Associate Professor with the School of Integrated Circuits, Anhui University. His research interests include communication integrated circuit system and the industrial Internet of Things.

Na Bai received the Ph.D. degree in microelectronics from the Southeast university & National University of Singapore, in 2008. She is currently an Associate Professor with Anhui University. Her research interests include memory design, radiation hardened circuit design, Novel RF tag chip design and development, and Universal SOC chip development.

Chenxiao Mao received the M.S. degree in the School of Electronics at Anhui University, in 2023. His research interests include communication systems and digital-analog hybrid ADC design.

Ding Zhou s currently a graduate student in the School of Integrated Circuits at Anhui University. His research interests include communication protocols and reconfigurable design.

面向自动驾驶的复杂场景建模及测试用例生成方法

夏春艳[1,2] 黄松[1] 郑长友[1] 姚永明[1] 孙乐乐[1] 胡哲[1] 阳真[1]

1 中国人民解放军陆军工程大学 南京 210007
2 牡丹江师范学院 黑龙江 牡丹江 157011

摘 要 自动驾驶汽车在缓解交通拥堵和消除交通事故方面发挥着重要作用。在自动驾驶汽车部署到公共道路之前，必须进行广泛的测试，而满足交通规则的测试是自动驾驶系统最基本的要求。现有的基于交规的测试方法主要关注简单的场景，从交通事故和违法场景手动设计生成测试用例，并不能很好的适应需要复杂的周围环境意识的驾驶情况。为了解决这些问题，本文提出了面向自动驾驶的复杂场景建模及测试用例生成方法，从交通法规出发，给出自动驾驶场景的形式化定义，提出复杂场景的组合关系，构建复杂场景的 Petri 网模型；基于适合 Petri 网模型的覆盖标准，采用搜索策略生成表征交规特征的模型路径；以发生碰撞为目标，采用组合参数的方法生成覆盖模型路径的测试用例；通过对百度 Apollo 的仿真实验结果表明，本文方法生成的测试用例能够发现自动驾驶系统 5 种典型场景类型的交通违规行为，从而验证了本文方法的有效性和合理性。

关键词：软件测试；自动驾驶系统；测试用例；交通法规

Modeling and Test Case Generation for Complex Scenarios of Autonomous Driving System

XIA Chunyan[1,2], HUANG Song[1], ZHENG Changyou[1], YAO Yongming[1], SUN Lele[1], HU Zhe[1] and YANG Zhen[1]

1 College of Command and Control Engineering, Army Engineering University of PLA, 210007, China
2 College of Computer and Information Technology, Mudanjiang Normal University, 157012, China

Abstract Autonomous vehicles play a crucial role in alleviating traffic congestion and eliminating traffic accidents. Before deploying autonomous vehicles on public roads, extensive testing must be conducted, with compliance to traffic regulations being the fundamental requirement. Existing rule-based testing methods primarily focus on simple scenarios, manually designing and generating test cases for traffic accidents and violations, which do not adapt well to driving situations that require complex environmental awareness. To address these issues, this paper proposes a modeling and test case generation for complex scenarios in autonomous driving system. Starting from traffic regulations, a formal definition of autonomous driving scenarios is provided, and a Petri net model of complex scenarios is constructed based on the combination relationships of these scenarios. Using coverage criteria suitable for Petri net models, a search strategy is employed to generate model paths that represent regulatory features. By targeting collision occurrences, a combination parameter approach is used to generate test cases that cover the model paths. Simulation experiments conducted on Baidu Apollo demonstrate that the test cases generated by this method can detect traffic violations in five typical scenario types of autonomous driving systems, thereby validating the effectiveness and rationality of this approach.

Keywords Software testing; Autonomous driving system; Test case; Traffic regulation

基金项目：牡丹江师范学院学科建设项目(MSYSYL2022008)；黑龙江省省属高等学校基本科研业务费项目(1452ZD010)；黑龙江省高等教育教学改革重点委托项目(SJGZ20200175)；黑龙省自然科学基金项目(LH2020F038)
This work was supported by Discipline Construction Project of Mudanjiang Normal University (No.MSYSYL2022008); Basic Research Fund Project of Provincial Universities in Heilongjiang Province (No.1452ZD010) and Key Commissioned Project for Higher Education Teaching Reform in Heilongjiang Province (No.SJGZ20200175); Natural Science Fund Project of Heilongjiang Province(LH2020F038)
通信作者：黄松（E-mail:huangsong@aeu.edu.cn）

1 引言

随着人工智能的深入发展，自动驾驶已成为人工智能技术的典型应用，近十年来取得了长足的发展。作为一类非确定性系统，自动驾驶汽车的安全性受到越来越多的关注，而避免碰撞是自动驾驶系统的主要安全要求之一[1]。2018 年，一辆 Uber 的自动驾驶汽车在公共道路上行驶时，由于没有礼让行人而发生交通事故，导致行人死忙[2]。因此，作为安全关键系统，自动驾驶汽车必须在实际道路部署之前进行充分测试，以确保其安全性。

研究人员针对自动驾驶系统的测试提出了许多关键场景生成技术[3-11]和测试用例生成方法[12-15]。在自动驾驶领域，场景被理解为一定时间和空间范围内行驶环境与汽车驾驶行为的综合反映，以评估自动驾驶汽车的安全性和可靠性。依据不同的抽象级别，场景可分为功能场景、逻辑场景和具体场景。测试用例被定义为包含“通过/失败”标准的具体场景，以评估自动驾驶汽车的执行情况，比如自动驾驶汽车和障碍物的碰撞。现有的大多数自动驾驶测试方法没有充分考虑交通安全法律法规的限制，这导致了自动驾驶汽车在公共道路行驶中存在严重的交通安全隐患。因此，基于交通安全法规的约束，构建和重组复杂场景，以及罕见的边缘测试用例来充分测试自动驾驶汽车的安全性是一个极具挑战的问题。

本文的主要工作如下：

（1）从交规出发，给出自动驾驶场景的形式化定义，提出复杂场景的组合关系，构建复杂场景的 Petri 网模型，并通过形式化方法验证模型设计与系统需求的一致性。

（2）基于复杂场景模型，提出适合 Petri 网的覆盖标准，采用搜索策略生成表征交规特征的场景模型路径；以发生碰撞为目标，采用组合参数的方法迭代生成覆盖模型路径的测试用例。

（3）通过在百度 Apollo 平台上的仿真实验评价本文方法，实验结果表明本文方法生成的测试用例能够发现自动驾驶汽车违反交规行为的 5 种不同场景类型，从而验证了本文方法的有效性和合理性。

2 相关工作

随着自动驾驶技术的快速发展，对自动驾驶系统的测试已经成为学术界和工业界的研究热点。目前，许多公司都致力于这个领域，如谷歌 Waymo、百度 Apollo 和 Autoware[16]。

当前较为流行的自动驾驶测试生成方法是基于数据驱动的场景生成方法和基于模型驱动的场景生成方法，以及基于搜索的测试用例生成方法。在数据驱动的方法中，场景是由现有的数据集生成，如碰撞报告[3-5]和真实世界的驾驶记录[6-8]。在模型驱动的方法中，场景是由基于系统模型的评价结果生成[9-11]。基于搜索的进化方法，通过迭代的结果来指导测试用例的生成[12-15]。此外，Petri 网理论在自动驾驶测试中也得到了广泛的应用。Tang 等人[17]提出了一种通过地图建模进行自动驾驶汽车路线覆盖的测试方法，该方法首先将地图建模为一个 Petri 网，然后基于 Petri 网模型定义了交叉路口的拓扑特征和路线特征，实验表明该方法生成的测试用例实现了更高的路线覆盖率。

通过实验验证，在测试自动驾驶系统安全性方面，上述场景生成方法均取得了较好的效果，但这些方法没有充分考虑交通安全法律法规的约束，使得生成的测试用例存在忽视交通法规的潜在可能性。如果可以生成表征交规特征的测试用例，进而测试自动驾驶系统是否存在违反交规的行为，则对于自动驾驶系统的安全性测试具有重要意义。鉴于此，本

文提出了一种面向自动驾驶系统的复杂场景建模及测试用例生成方法，通过在工业级平台百度 Apollo 上的仿真实验，结果显示本文方法生成的测试用例能够发现自动驾驶系统中 5 种不同场景类型的交通违规行为，进而验证了本文方法的有效性和合理性。

3 复杂场景建模

自动驾驶复杂场景可以被认为是通过基本场景按照某种关系组合而成的。在本节中，首先描述复杂场景的组合关系，然后阐述复杂场景的建模与验证。

3.1 复杂场景的组合关系

本文中的基本场景主要包括巡航、变道、超车、路口、跟车和停车。其中，基本场景又可进一步划分为元场景。比如，变道场景分为向左变道和向右变道两个元场景；路口场景分为交叉路口和环形路口两个场景，而交叉路口场景还可以更进一步细分为左转、直行、右转和掉头四个元场景，环形路口场景也可以更进一步细分为驶入环形路口和驶出环形路口两个元场景。最后，这些元场景通过参数实例化，可以获得用于测试的具体场景。本文中，基本场景和元场景的关系如表 1 所示。

表 1　基本场景和元场景的关系

巡航	cruise	静止障碍物	obstacle_static	跟车	following	车辆切入	cut_in
		动态障碍物	obstacle_dynamic			车辆切出	cut_out
		正常巡航	cruise_normal			突发情况	accident_traffic
变道	change	向左变道	change_left	超车	overtake	跟随目标车辆	lane_follow
		向右变道	change_right			并入相邻车道	lane_merge
停车	stop	应急车道	lane_emergency			超越目标车辆	lane_pass
		右侧车道	lane_right			返回原来车道	lane_return
路口	crossing	交叉路口	intersection	左转	cross_left	直行	straight_ahead
				右转	cross_right	掉头	turn_around
		环形路口	roundabout	驶入环形路口	roundabout_in	驶出环形路口	roundabout_out

基于交规构建的场景模型使用的是自然语言对交通规则进行的描述。为了解决自然语言与形式化语言的鸿沟问题，本文给出了场景的形式化定义。场景被形式化的描述为一个三元组<π，＜，s>，π代表的是一组场景，s 为π中的一个场景，即 s∈π，＜表示优先关系，是π上的严格顺序关系，例如，$s_1 < s_2$ 表示 s_1 排在 s_2 的前面，即 s_1 优先 s_2 的发生。由于任何场景的出现都可以看成是事件的发生，所以场景的关系可以使用事件之间的关系来描述。这里场景的元素描述的仍然是场景。因此，基于场景的形式化定义，可以描述场景之间的三种常见关系，如表 2 所示。

表 2　场景关系的描述

定义	含义	示例
$s_1 < s_2$	s_1 的发生严格先于 s_2 的发生	s_1=“跟车”，s_2=“超车”，则 $s_1 < s_2$
$s=s_1 \cup s_2$	如果 s_1 或 s_2 发生，则 s 发生	s=“变道”，s_1=“向左变道”，s_2=“向右变道”，则 $s=s_1 \cup s_2$
$s=s_1 \cap s_2$	如果 s_1 与 s_2 发生，则 s 发生	s=“环形路口”，s_1=“驶入环形路口”，s_2=“驶出环形路口”，则=$s_1 \cap s_2$

基于上述场景的形式化定义，以及表 2 中场景的三种常见关系的描述，将表 1 中的基本场景和元场景之间的关系详细描述如下。

Relation1：cruise = obstacle_static ∪ obstacle_dynamic ∪ cruise_normal

Relation2：lane change = change_left ∪ change_right

Relation3：following = cut_in ∪ cut_out ∪ accident_traffic

Relation4：crossing = intersection ∪ roundabout，

Relation4.1：roundabout = roundabout_in ∩ roundabout_out,

Relation4.2：intersection = cross_left ∪ straight_ahead ∪ cross_right ∪ turn_around，

Relation5：overtake = lane_follow ∩ lane_merge ∩ lane_pass ∩ lane_return

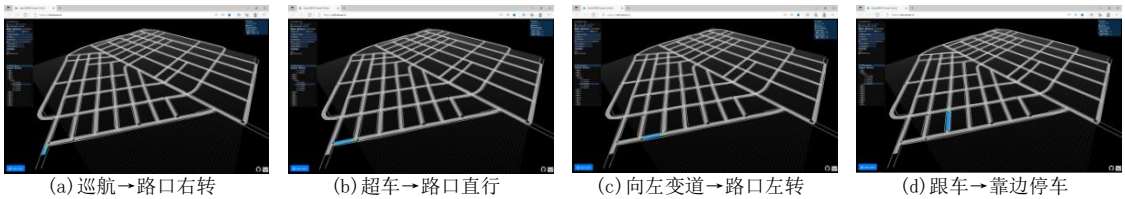
Relation6：stop = lane_emergency ∪ lane_right

其中，Relation4 描述路口场景由交叉路口或者环形路口构成，当且仅当二者其一被触发，路口场景才会出现。进一步，Relation4.1 描述环形路口场景由驶入环形路口和驶出环形路口两个元场景构成，当且仅当二者都被触发，环形路口场景才会出现；Relation4.2 描述交叉路口场景由左转、直行、右转和掉头四个元场景构成，其中一个元场景的发生即可导致交叉路口场景的发生。由此可知，基于场景关系构建复杂场景时，“或”关系描述的元场景只能出现一个，“与”关系描述的元场景每个都要出现，更为重要的是所有场景必须严格遵守“优先”关系。

3.2 复杂场景的建模与验证

为了建模复杂场景并验证模型设计与系统需求的一致性，本文考虑基本场景交叉融合为复杂场景的情况。实际上，基本场景可能只有一名行人或一个障碍车辆，相对简单，但由于复杂场景是由多个基本场景组合而成，同时考虑多条交规的约束，并增加了自动驾驶汽车和障碍车辆或行人的交互行为，所以增加了场景交互环境的复杂性。

基于交规建模的基本场景，选择地图上的一段区域构建自动驾驶系统的复杂场景，包括巡航、路口、变道、跟车、停车等基本场景。本文以图 1 所示的自动驾驶汽车运行轨迹路线为例，来说明复杂场景的建模及验证过程。



(a) 巡航→路口右转　(b) 超车→路口直行　(c) 向左变道→路口左转　(d) 跟车→靠边停车

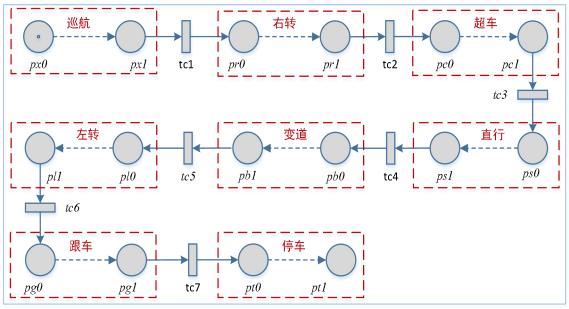
图 1　自动驾驶汽车的道路运行轨迹路线

该段道路的场景组合关系如下：

cruise< cross_right< overtake< cross_straight< change_left< cross_left< following< stop

其中，overtake = lane_follow∩lane_merge∩lane_pass∩lane_return。

在基本场景中，设有接收指令，相当于其它场景模型调用该场景模型的接口。容易理解，可以通过调用变迁来发送指令，进而将基本场景模型组合成复杂场景模型。依据图 1 自动驾驶汽车道路运行轨迹路线，以及场景组合关系，构建复杂场景的 Petri 网模型，便于理解，图 2 给出了复杂场景的 Petri 网模型简图。



（a）Petri 网模型简图

pl0~pl1	路口左转场景
ps0~ps1	路口直行场景
pr0~pr1	路口右转场景
px0~px1	巡航场景
pc0~pc1	超车场景
pb0~pb1	变道场景
pg0~pg1	跟车场景
pt0~pt1	停车场景

tc1	发送指令，交叉路口右转
tc2	发送指令，超越前方车辆
tc3	发送指令，交叉路口直行
tc4	发送指令，向左变更车道
tc5	发送指令，交叉路口左转
tc6	发送指令，跟车行驶
tc7	发送指令，路边车道停车

（b）Petri 网模型的库所和变迁

图 2　Petri 网模型简图

在前期研究成果中，详细的给出了基于 Petri 网的自动驾驶交叉路口场景的建模及验证方法[18]。在此基础上，本文首先构建巡航、超车、变道、跟车和停车等基本场景的 Petri 网模型，并通过形式化方法验证模型设计与系统需求的一致性。然后，基于复杂场景的 Petri 网模型简图和组合关系，通过调用变迁将基本场景的 Petri 网模型进行组合，从而构建复杂场景的 Petri 网模型，如图 3 所示。https://github.com/xiachunyan/test2，复杂场景的形式化验证结果可以访问上述链接。

基于交规构建复杂场景的 Petri 网模型，通过模型设计与系统需求的形式化验证，实验结果表明模型路径能够表征交通规则。例如，交叉路口场景模型通过了“路口礼让行人”的规则验证，说明模型的对应路径能够表征“路口礼让行人”的交通规则。因此，基于复杂场景的 Petri 网模型，生成覆盖模型路径的表征交规特征的测试用例，能够测试自动驾驶系统是否存在的违反交通规则行为，从而提高自动驾驶系统的安全性和可靠性。

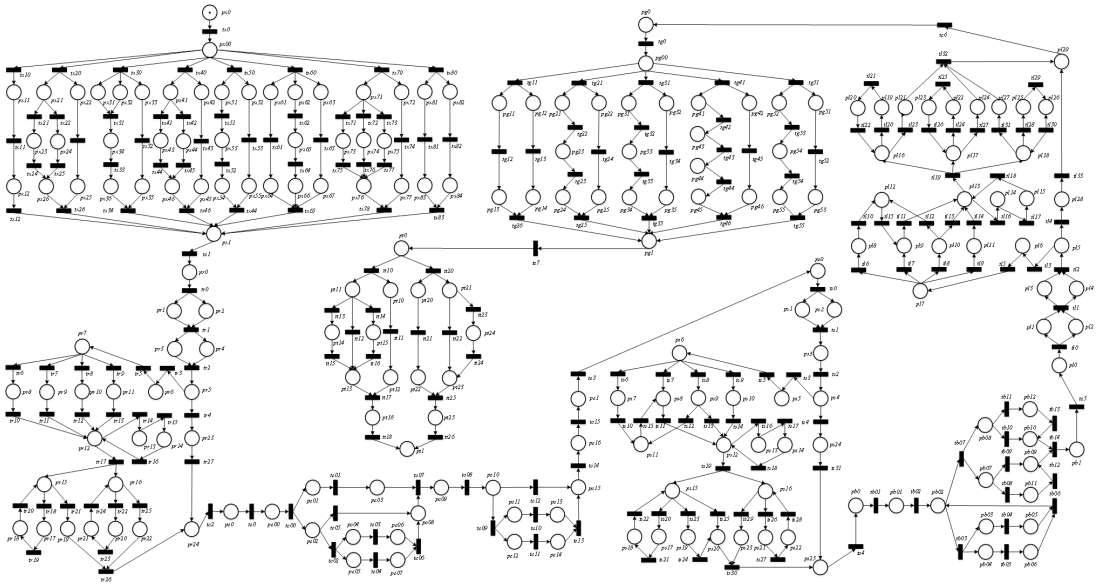


图 3　复杂场景的 Petri 网模型

4　测试用例生成

在本节中，首先采用遍历模型的方法生成无测试用例的状态调用序列，即 Petri 网模型的测试路径；然后采用组合参数的方法生成覆盖模型路径的可执行状态调用序列，即覆盖模型路径的测试用例。

4.1　场景模型的测试路径生成策略

目前，Petri 网没有规范的覆盖标准，本文依据状态图的覆盖标准定义 Petri 网模型的库所覆盖准则、变迁覆盖准则和库所变迁覆盖准则。

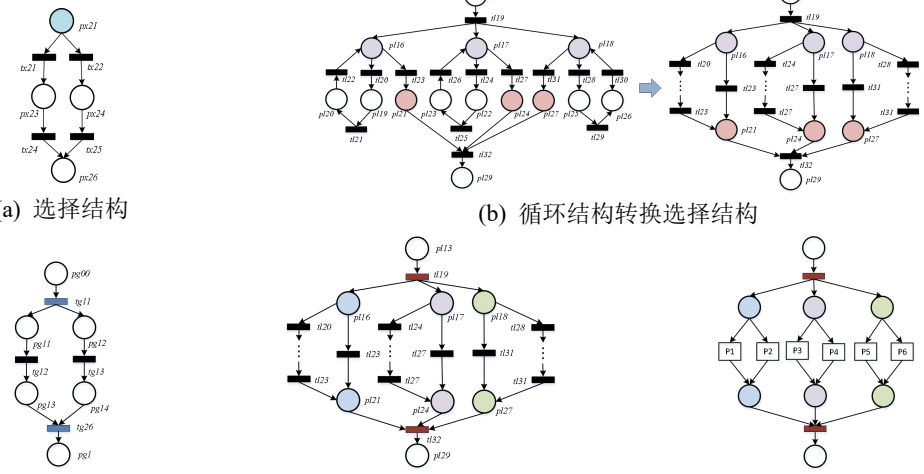
（1）库所覆盖准则：测试用例集应确保 Petri 网模型中的所有库所至少被访问一次；

（2）变迁覆盖准则：测试用例集应确保 Petri 网模型中的所有变迁至少被激活一次；

（3）库所变迁覆盖准则：测试用例集应确保 Petri 网模型中的所有库所至少被访问一次和所有变迁至少被激活一次。

基于交规构建的自动驾驶复杂场景的 Petri 网模型，其基本结构包括五种：顺序结构、选择结构、循环结构、同步结构和并发结构。顺序结构，按照库所和变迁的顺序关系依次

覆盖测试路径。选择结构，覆盖每一条测试路径，即各个分支至少被遍历一次，而且选择结构的互斥分支不能在同一条测试路径中同时出现，选择结构如图 4(a)所示。循环结构，为了保证测试的相对完备性，可考虑循环体被执行 0 次和 1 次两种情况。因此，可将循环结构转化为选择结构，如图 4(b)所示。



(a) 选择结构　　(b) 循环结构转换选择结构

(c) 并发-同步简单结构　　(d) 并发-同步复杂结构　　(e) 并发-同步复杂结构简图

图 4　Petri 网模型的结构图

对于 Petri 网模型中存在的并发结构和同步结构，由于可能存在互斥分支，所以增大了测试的难度。下面分别对并发-同步结构是否存在互斥分支的两种情况进行分析。

（1）不存在互斥分支：在并发-同步结构中，交互变迁的路径不只一条，但不存在互斥分支，如图 4(c)所示。这种不存在互斥分支的简单结构中，各个并发-同步路径的选择方式很多，利用随机生成方法任意选择路径依次进行测试，其结果是一样的，还可避免组合爆炸和多次重复测试等问题。

（2）存在互斥分支：在并发-同步结构中，交互变迁的路径不只一条，但存在互斥分支，如图 4(d)所示。在其简图 4(e)中，P_1 与 P_2、P_3 与 P_4，以及 P_5 与 P_6 为互斥分支。通过分析选择结构可知，互斥路径是不能被同时执行的，所以不能直接选择这些互斥分支作为并发-同步路径。在并发-同步结构中，如果交互库所的组数为m，对应库所的互斥分支数为$n_1, n_2, \ldots n_m$个，那么m组交互库所的组合数为$n_1 \times n_2 \times \ldots \times n_m$，而且每个组合都是一条并发-同步路径。在并发-同步结构中，交互库所有多条分支时，由此而产生的并发-同步路径的数目就会更多，进而产生组合爆炸。针对这一情况，本文采用两两组合的测试覆盖方法。

两两组合覆盖准则：在并发-同步结构中，对于有m组交互库所的复杂结构，对应库所的互斥分支数为$n_1, n_2, \ldots n_m$，那么要求这些交互库所集合中的任何两个分支的所有可能执行路径的组合至少被一个测试用例覆盖。这个覆盖准则既能够保证测试的完整性和充分性，还能防止组合爆炸问题。在简图 4(e)中，P_1...P_6 表示该模型的 6 条路径，采用普通的组合方法产生并发-同步路径的数目为 $2 \times 2 \times 2 = 8$ 条，而采用两两组合的方法产生并发-同步路径的数目为 4 条，即{P_1，P_3，P_5}、{P_1，P_4，P_6}、{P_2，P_3，P_6}和{P_2，P_4，P_5}。并发-同步路径{P_1，P_3，P_5}覆盖了{P_1，P_3}、{P_1，P_5}和{P_3，P_5}；并发-同步路径{P_1，P_4，P_6}覆盖了{P_1，P_4}、{P_1，P_6}和{P_4，P_6}；并发-同步路径{P_2，P_3，P_6}覆盖了{P_2，P_3}、{P_2，P_6}和{P_3，P_6}；并发-同步路径{P_2，P_4，P_5}覆盖了{P_2，P_4}、{P_2，P_5}和{P_4，P_5}。由此可见，

两两组合的方法能够覆盖任意两个分支的所有可能执行路径的组合，既能保证测试的完整性和充分性，又能防止组合爆炸。

在基于 Petri 网模型的测试中，每条测试路径表示为 Petri 网模型的初始状态到终止状态的完整路径，新生成的测试路径应该至少含有一个未在其他路径中出现的库所或者变迁。为了达到测试的有效性和减少冗余，在搜索模型的测试路径时，原则是所有路径的集合需要覆盖模型中全部的库所和变迁，选择分支至少执行一次和循环路径最多执行一次，并发和同步的复杂结构采用两两组合覆盖准则，而且测试路径集应该包含 Petri 网模型中从初始状态开始到终止状态结束的全部路径。

4.2 路径覆盖的测试用例生成方法

在本文中，通过定义抽象场景，指定一些固定的特征，比如自动驾驶车辆(EGO)的初始位置，应用组合参数的迭代方法搜索潜在的碰撞测试用例。在抽象场景中，每个场景都有不同的配置参数，通过在特征阈上定义的变量进行参数化。在搜索中，测试用例是这些变量的赋值，可以通过使用分配变量值的实例化抽象场景来派生具体场景。例如，首先生成基本测试用例，即没有任何交通参与者(NPC)的测试用例，然后在测试场景中通过添加 NPC 的方式来搜索复杂场景的测试用例。容易理解，车辆是否发生碰撞受到 EGO 和 NPC 之间的相对距离和速度的影响。在确定了 NPC 的初始速度后，每个测试用例都可以通过 EGO 的初始位置和初始速度，以及 EGO 和 NPC 的相对距离来参数化。由此可知，每个测试用例都可以被描述为一个向量 $t = (s, v, d)$，其中，s为 EGO 的初始位置，v为 EGO 的初始速度，d为 EGO 和 NPC 的相对距离。

基于复杂场景的 Petri 网模型，以发生碰撞为目标，采用组合参数的方法生成测试用例。首先，生成覆盖模型路径的基本测试用例，即没有任何 NPC 的测试用例，以确保生成测试用例的有效性。对于场景模型的测试路径P^j，$\forall P^j \in PC$，其中 PC 为测试路径集。基本测试用例为$t_{i0}^j = (s_{i0}^j, v_{i0}^j, \emptyset)$，$s_{i0}^j$为 EGO 的初始位置，$v_{i0}^j$为 EGO 的初始速度，这里$\emptyset$意味着没有 NPC。$\forall t_{i0}^j \in T_i^j$，$T_i^j$为$P^j$中的第$i$个元场景的基本测试用例集。$\forall T_i^j \in T^j$，$T^j$表示覆盖$P^j$的测试用例集，$\forall T^j \in TC$，而$TC$为覆盖所有测试路径的测试用例集。

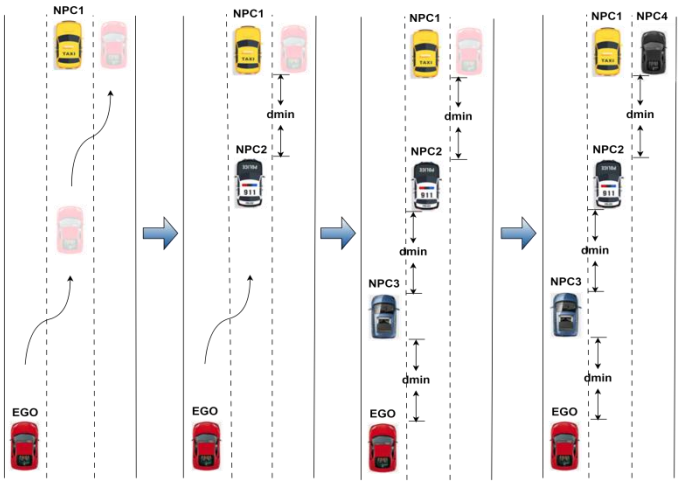


图 5　测试用例生成的工作流程

其次，通过添加 NPC 来搜索碰撞测试用例，如图 5 所示。给定一个元场景的基本测试用例，其覆盖的模型路径$P^j = Cov(t_{i0}^j)$，通过添加 NPC 生成新的测试用例$t_{ik}^j =$

$(s_{ik}^j, v_{ik}^j, d_{ik}^j)$，$s_{ik}^j$为新生成测试用例中 EGO 的位置，$v_{ik}^j$为新生成测试用例中 EGO 的速度，$d_{ik}^j$为新生成测试用例中 EGO 与 NPC 的相对距离，依据交规特征设定变量的阈值，以发生碰撞为目标，调整变量的参数值，得到覆盖模型路径的元场景测试用例集T_i^j，$\forall t_{ik}^j \in T_i^j$。

然后，选择元场景变量的不同参数进行组合，生成组合参数的可执行状态调用序列，也就是覆盖模型路径的复杂场景的测试用例集，$P^j = Cov(T^j)$，$\forall T^j \in TC$。为了保证测试用例的多样性，避免产生过多的冗余测试用例，当同一基本场景多次出现在复杂场景时，原则是从基本场景中选择不同的元场景组合复杂场景。因此，根据基本场景组合的实际场景也极其复杂，能够满足自动驾驶系统对测试场景复杂性的需求。

算法 1 描述了模型路径覆盖的测试用例生成方法。首先，生成每个元场景的基本测试用例，并检查生成的测试用例是否有效。如果生成的基本测试用例覆盖模型路径，将其加入元场景的测试用例集(第 2-6 行)。否则，重新生成基本测试用例，直到生成有效的测试用例为止(第 7-13 行)。然后，生成元场景的碰撞测试用例，并检查新生成的测试用例是否覆盖模型路径。如果覆盖模型路径，将其加入元场景的测试用例集，并删除该元场景的基本测试用例(第 14-18 行)。否则，不会修改元场景的测试用例集，直到生成有效的碰撞测试用例为止(第 19-20 行)。最后，将生成的元场景测试用例集添加到覆盖模型路径的测试用例集(第 21 行)，从而得到覆盖模型所有路径的测试用例集(第 22 行)。

算法 1：路径覆盖的测试用例生成

输入：Petri 网模型的测试路径集 *PC*.
输出：覆盖模型路径的测试用例集 *TC*.

```
1   初始化：TC = ∅;
2   for ∀P^j ∈ PC do
3       for i = 0:n do
4           t_i0^j = (s_i0^j, v_i0^j, ∅);   //生成元场景无障碍的基本测试用例
5           if Cov(t_i0^j) ∈ P^j then
6               T_i^j = T_i^j ∪ {t_i0^j};
7           else
8               while t_i0^j = (s_i0^j, v_i0^j, ∅) do
9                   if Cov(t_i0^j) ∈ P^j then
10                      T_i^j = T_i^j ∪ {t_i0^j};
11                      break;
12                  else
13                      continue;
14          while t_ik^j = (s_ik^j, v_ik^j, d_ik^j) do   //调整变量参数值，组合生成元场景的碰撞测试用例
15              if Cov(t_ik^j) ∈ P^j then
16                  T_i^j = T_i^j ∪ {t_ik^j};
17                  T_i^j = T_i^j \ {t_i0^j};
18                  break;
19              else
20                  continue;
21          T^j = T^j ∪ {T_i^j};
22      TC = TC ∪ {T^j};   //生成覆盖路径的测试用例集
```

5 仿真实验

为了评估本文方法的有效性和合理性，在百度 Apollo 6.0 平台上进行了仿真实验，这是研究界和行业中最受欢迎的自动驾驶系统之一。仿真环境为 LGSVL 2021.3，使用台式计算机在 Ubuntu 18.04 上运行实验，其配置为：Inter Core i9-10900K，32GB RAM 和 NVIDIA RTX 3090Ti 24G GPU。

本文主要研究以下两个问题：

（1）有效性：生成的测试用例是否能够有效地发现自动驾驶系统的安全违规行为？

（2）合理性：发现的安全违规行为是否为自动驾驶汽车违反交规导致的安全问题？

为了确保仿真实验是从有效且有意义的状态开始，对仿真环境的初始状态有下面的约束：当所有车辆启动时，EGO 与 NPC 之间存在安全距离。此外，本文方法并不局限于特定的模拟器，选择 LGSVL 是由于其支持与 Apollo 的链接，也可以迁移到其它模拟器，比如 Calar，只要它们提供 API 接口链接到自动驾驶系统即可。

5.1 有效性分析

通过在区域地图上的仿真测试，对自动驾驶复杂场景的建模及测试用例生成方法进行评估。观察分析实验中的所有安全违规行为，根据组成复杂场景的基本场景类型，将发现的安全违规行为分为 5 种不同的场景类型，并对其中的典型例子进行描述分析。

（1）路口场景，在示例 1 中，当通过交叉路口时，EGO 与 NPC 发生碰撞，如图 6(a)所示。当接近路口时，EGO 的速度为每小时 36 公里，从左边准备通过路口的 NPC1 相对较慢，EGO 决策通过路口而不是让行。当到达路口时，EGO 的速度为每小时 32 公里，左边 NPC2 也接近路口，险些发生碰撞，此时已经相当危险。当通过路口时，EGO 的速度为每小时 22 公里，采取制动让行 NPC3，然而为时已晚，EGO 速度太快，无法避免与 NPC3 发生碰撞。如果 EGO 不超速行驶，并且让行 NPC，那么碰撞是可以避免的。

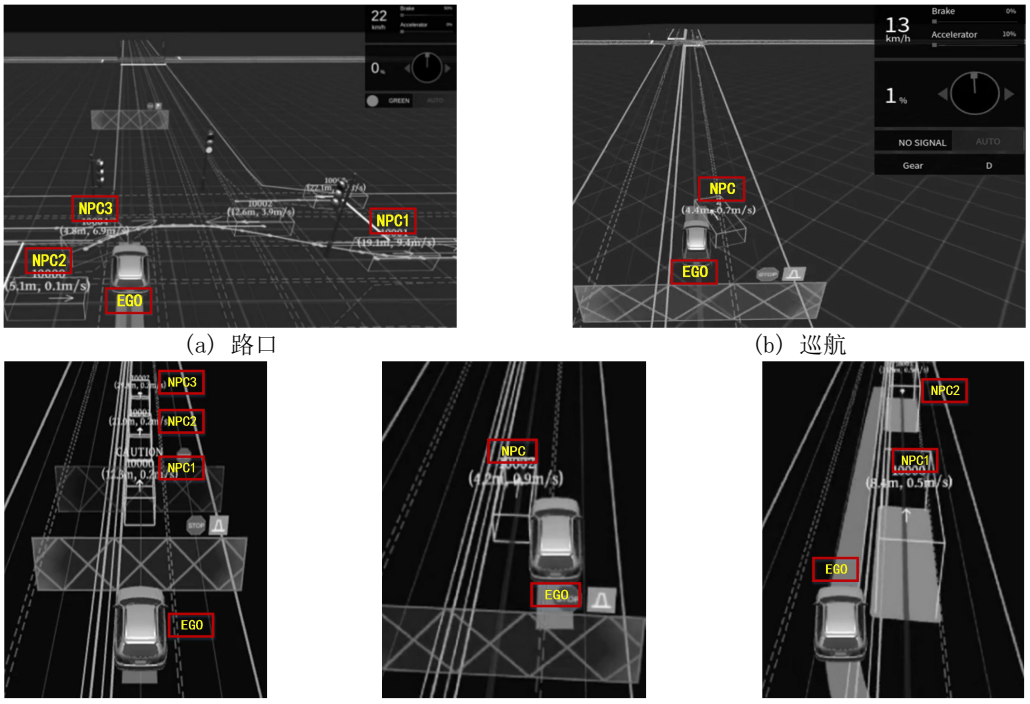
（2）巡航场景，在示例 2 中，当 NPC 切入巡航车道时，EGO 与 NPC 发生碰撞，如图 6(b)所示。当 EGO 在道路上巡航行驶时，NPC 从右侧车道切入，EGO 与 NPC 没有保持必要的安全距离，未能及时避让 NPC，导致 EGO 与 NPC 发生碰撞。如果 EGO 与 NPC 保持足够的安全距离，并及时刹车避让，那么碰撞是可以避免的。

（3）跟车场景，在示例 3 中，当跟车缓慢行驶时，EGO 未选择变更车道或者超车行驶，并且与前方 NPC1 未能保持安全距离，险些发生追尾事故，如图 6(c)所示。当 EGO 在道路上跟车行驶时，前方 NPC1 行驶缓慢，此时 EGO 并未选择向右变更车道进而超车行驶，最后导致 EGO 与 NPC1 非常接近，险些与其发生碰撞。在这种情况下，如果 EGO 选择右侧超车行驶，则能迅速到达目的地。

（4）变道场景，在示例 4 中，当没有充足的安全距离时，EGO 变更车道发生碰撞，如图 6(d)所示。在向左变更车道时，EGO 生成了一个非常接近 NPC 的轨迹来执行变道。然后，当 EGO 沿着轨迹移动时，它与 NPC 发生碰撞。在这种情况下，EGO 应该在 NPC 的更前面执行变道。

（5）超车场景，在示例 5 中，当同向双车道时，EGO 借用对向车道超车，如图 6(e)所示。在前方 NPC1 速度较慢时，EGO 生成了一个向左变更车道的轨迹来执行超车，通过左侧车道行驶强制超越 NPC1 和 NPC2。然而，左侧车道属于对向车道，右侧车道属于同向车道。此时，EGO 应该通过向右变更车道行驶来完成超车。

通过实验结果描述分析可知，依据交规建模自动驾驶复杂场景，并基于模型生成表征交规特征的测试用例，能够有效的发现自动驾驶系统存在的安全违规行为，对于测试自动驾驶系统的安全性和可靠性具有重要的意义。



(a) 路口　(b) 巡航

(c) 跟车　(d) 变道　(e) 超车

图 6 实验仿真场景图

5.2 合理性分析

本文针对交通法规建模测试场景，进而生成表征交规特征的测试用例，其主要目的就是从交通法规出发，测试自动驾驶汽车违反交规的行为，以提高自动驾驶系统的安全性。因此，针对本文方法发现的自动驾驶系统存在的 5 种不同场景类型的安全问题，分析导致问题出现的最可能原因。

（1）路口场景，在示例 1 中，违反的交通法规为“当机动车通过交叉路口时，最高行驶速度不得超过每小时 30 公里，并且少数车辆让行多数车辆”。在通过交叉路口时，EGO 没有减速至每小时 30 公里以下，而且没有遵守少数车辆让行多数车辆的原则，由于速度过快和没有让行发生了碰撞。由此可知，发生碰撞的最可能原因是 EGO 违反了交通规则，超速行驶并且没有让行，从而导致的交通事故。

（2）巡航场景，在示例 2 中，违反的交通法规为“当机动车在道路上行驶时，应与其他车辆保持必要的安全距离”。在道路上行驶时，EGO 没有与右侧车道的 NPC 保持必要的安全距离，当 NPC 切入巡航车道时与其发生碰撞。由此可知，发生碰撞的最可能原因是 EGO 违反了交通规则，没有与其他车辆保持必要的安全距离，从而导致的交通事故。

（3）跟车场景，在示例 3 中，违反的交通法规为“当机动车在道路上跟车行驶时，应当保持足够的安全距离，当前方车辆行驶缓慢时，可以选择变更车道或者超车行驶”。在跟车行驶时，EGO 没有与前车保持足够的安全距离，并且在前车行驶缓慢的情况下，未能选择变更车道或者超车行驶，险些发生碰撞事故。由此可知，虽然 EGO 和 NPC 没有发生追尾事故，但可以预测自动驾驶系统存在非常严重的交通安全风险，很可能导致交通事故。

（4）变道场景，在示例 4 中，违反的交通法规为“当机动车变道时，应该在确认有充足的安全距离后，变更车道”。在向左变更车道时，EGO 没有与 NPC 拉开必要的安全距离后再变道，出现了车辆碰撞场景。由此可知，发生碰撞的最可能原因还是 EGO 违反了

交通规则，安全距离不足而变更车道，从而导致的交通事故。

（5）超车场景，在示例 5 中，违反的交通法规为“在同向双车道情况下，机动车超越前车时应该借用同向车道行驶”。在超车时，EGO 没有借用同向车道行驶，而是借用对向车道行驶，违反了交通规则。在这种情况下，虽然 EGO 完成了超车动作而没有发生碰撞，但是可以预测到自动驾驶系统存在非常严重的交通安全风险，如果对向车道有车行驶，则随时可能发生碰撞，从而导致交通事故。

从上述分析可知，本文方法不仅可以发现自动驾驶汽车发生碰撞的问题，还可以发现更多的自动驾驶汽车违反交规的行为。比如，本文方法发现了自动驾驶汽车左侧超车的违规行为，而自动驾驶汽车的这种违规行为也是导致交通事故的最大安全隐患。综上所述，本文方法生成的表征交规特征的测试用例能够发现自动驾驶汽车违反交通法规的行为，并且这些违反交规的行为正是自动驾驶汽车发生交通事故的主要原因，从而也说明了应用本文方法测试自动驾驶汽车安全违规行为的合理性。

6 总结与展望

本文提出了一种面向自动驾驶的复杂场景建模及测试用例生成方法，旨在从交通法规出发，依据复杂场景的组合关系，构建自动驾驶系统复杂场景的 Petri 网模型，并提出适合 Petri 网模型的覆盖标准，采用搜索策略生成场景模型的测试路径，以发生碰撞为目标，采用组合参数方法生成覆盖模型路径的测试用例，用于测试自动驾驶系统的安全性。通过仿真实验结果表明，本文方法生成的测试用例能够有效的发现自动驾驶系统存在的安全问题，而导致这些问题的最可能原因是自动驾驶汽车存在违反交通法规的行为。

后续工作，将研究基于交规建模自动驾驶复杂场景的更多测试用例生成方法，如进化方法。进一步，将研究测试用例的优化方法，以提高自动驾驶系统的测试效率。

参考文献

[1] Tang Y, Zhou Y, Liu Y,et al. Collision Avoidance Testing for Autonomous Driving Systems on Complete Maps[C]. 2021 IEEE Intelligent Vehicles Symposium (IV). Nagoya, Japan: IEEE, 2021: 179-185.

[2] An DD, Liu J, Chen XH, et al. Formal modeling and dynamic verification for human cyber physical systems under uncertain environment. Ruan Jian Xue Bao/Journal of Software, 2021, 32(7): 1999-2015 (in Chinese).

安冬冬,刘静,陈小红,等.不确定环境下 hCPS 系统的形式化建模与动态验证.软件学报, 2021, 32(7): 1999-2015.

[3] Toma S. Depiction of priority light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications[J]. National Highway Traffific Safety Administration. U.S. Department of Transportation, DOT HS 811 732, Apr. 2013.

[4] Nitsche P, Thomas P, Stuetz R, et al. Pre-crash scenarios at road junctions: A clustering method for car crash data[J]. Accident Analysis & Prevention, 2017, 107: 137-151,.

[5] Hauer F, Schmidt T, Holzmuller B, et al. Did we test all scenarios for automated and autonomous driving systems?[C]. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019: 2950-2955.

[6] Ding W, Xu M, Zhao D. Cmts: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios[C]. 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020: 4314-4321.

[7] Paardekooper J, Montfort S, Manders J, et al. Automatic identifification of critical scenarios in a public dataset of 6000 km of public-road driving[C]. 26th International Technical Conference on the Enhanced Safety of Vehicles (ESV), 2019.

[8] Zhao D, Lam H, Peng H, et al. Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques[J]. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(3): 595-607.

[9] Beglerovic H, Stolz M, Horn M. Testing of autonomous vehicles using surrogate models and stochastic optimization[C]. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017: 1-6.

[10] Althoff M, Lutz S. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles[C]. 2018 IEEE Intelligent Vehicles Symposium (IV), 2018: 1326-1333.

[11] Klischat M, Althoff M. Generating critical test scenarios for automated vehicles with evolutionary algorithms[C]. 2019 IEEE Intelligent Vehicles Symposium (IV), 2019: 2352-2358.

[12] Abdessalem B, Panichella A, Nejati S, et al. Testing autonomous cars for feature interaction failures using many-objective search[C]. Proc. of the 33rd IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE), 2018: 143-154.

[13] Abdessalem R, Nejati S, Briand L, et al. Testing vision-based control systems using learnable evolutionary algorithms[C]. Proc. of the 40th IEEE/ACM Int'l Conf. on Software Engineering (ICSE), 2018: 1016-1026.

[14] Gambi A, Mueller M, Fraser G. Automatically testing self-driving cars with search-based procedural content generation[C]. Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis, 2019: 318-328.

[15] Zhong Z, Kaiser G, Ray B. Neural Network Guided Evolutionary Fuzzing for Finding Traffic Violations of Autonomous Vehicles[J]. arXiv, 2022.

[16] Tang Y, Zhou Y, Zhang T, et al. Systematic Testing of Autonomous Driving Systems Using Map Topology-Based Scenario Classification[C]. 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). Melbourne, Australia: IEEE, 2021: 1342–1346.

[17] Tang Y, Zhou Y, Wu F, et al. Route coverage testing for autonomous vehicles via map modeling. 2021 IEEE International Conference on Robotics Automation(ICRA), 2021.11450-11456.

[18] Xia CY, Huang S, Zheng CY, et al. Modeling and verification method of intersection test scenario for automatic driving[J]. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3002–3021. (in Chinese).

夏春艳,黄松,郑长友,等.自动驾驶交叉路口测试场景建模及验证方法[J].软件学报, 2023, 34(7): 3002–3021.

目标遥感数据检测与识别模型

张东源，余川江，田治贵，李研

（中国电子科技集团公司第二十研究所软件测评中心，西安 710068）

摘要：人工智能软件测试中数据集的质量是保证模型预测准确性和可靠性的关键，其中由卫星、飞艇、无人机等设备获取的目标遥感数据在军事领域有广泛的应用。针对遥感数据中目标的检测与识别需要使用专业的数据处理模型和算法，并结合有效的检测指标进行定量分析，以保证人工智能软件在遥感数据应用场景中的性能和预测能力。确保目标数据集中的数据完整、准确、具有代表性和可靠性，从而确保模型在实际应用场景中的预测质量。

关键词：人工智能软件测试；数据集质量；遥感数据；目标检测

1 概述

目标遥感数据检测与识别主要可以分为单阶段法和两阶段法两种方法，其中单阶段法是一种比较新的方法，相比于传统两阶段目标检测方法，有着更快的检测速度的优势，单阶段法可以直接对特征图进行处理，大大减少了计算量，因此通常比两阶段法具有更快的检测速度。另外，单阶段法在小目标和重叠目标方面的表现通常比两阶段法更好,因为单阶段法的检测框是直接对全图像进行处理，而不是仅在感兴趣的区域内进行处理，更适合对于目标分布稀疏的图像进行处理。

本文采用单阶段法对目标遥感数据进行检测与识别，其基本流程包括以下几个步骤：

1. 数据增强：将待检测的图像进行数据增强，然后将增强后的数据输入到模型中进行处理；
2. 模型结构：模型的网络结构采用 DarkNet 结构，是比较轻量级的卷积神经网络，并且拥有很好的特征提取能力；
3. 注意力机制：通过动态地调整每个区域的重要性，使模型更关注有意义的特征，从而增强目标的区分度，减少噪声的影响；
4. 数据训练：利用卷积神经网络对输入图像进行特征提取和处理。这个过程

中，模型采用了全卷积的方式，即将全连接层替换成卷积层，将最后一层卷积层的输出结果作为特征图；

5. 输出结果：针对每个特征图，采用先验框对目标的位置以及尺寸进行估计，同时算出置信度分数，输出预测框的坐标和预测框的置信度分数，置信度越大表示识别到相应的对象的可能性越大。

通过目标增强、添加注意力机制、进行数据训练等过程，能够具有高速率和较为准确的对象检测结果，利用检测结果来筛选数据集，从而提高目标数据集的质量。

2 目标遥感数据检测与识别过程

2.1 目标数据增强

对输入的普通光学遥感影像进行数据增强，使用的数据增强方法有：图像扰动，改变亮度、对比度、饱和度、色调，加噪声，随机缩放，随机裁剪，翻转，旋转，随机擦除，Mosaic 等方式。

2.2 检测模型结构

目标检测模型，整个模型的结构如下图所示。

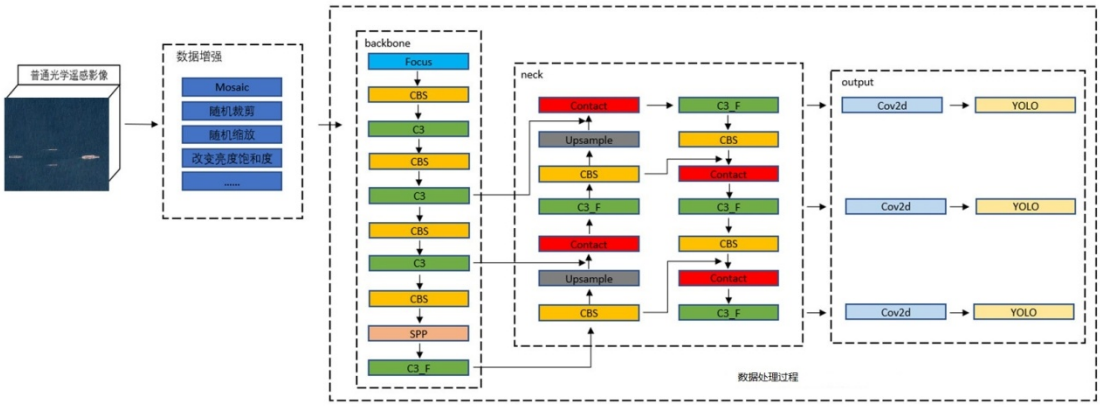


图 1 目标检测模型结构

将增强数据集输入到神经网络中，模型的网络主要由三个部分组成：

1）Backbone:利用卷积神经网络处理输入图像，生成深层的特征图；

2）Neck:对特征进行组合与混合，并把这些特征传递给预测层；

3）Output:进行最后的预测和输出

2.3 注意力机制

在目标检测任务中，添加注意力机制可以提高模型的性能。通过增强目标的区分度，注意力机制可以提高模型的检测精度。在目标检测任务中，模型需要快速有效地定位对象，注意力机制可以帮助模型更准确地区分目标所在的区域。注意力机制可以使模型更聚焦于关键信息，减少对背景干扰，并且在缺少目标信息的情况下，注意力机制可以增强一些替代区域的关注度，减少可能的误差。

Transformer 注意力机制模块作为编码器－解码器架构的一个实例，其整体架构图如图 2 所示。

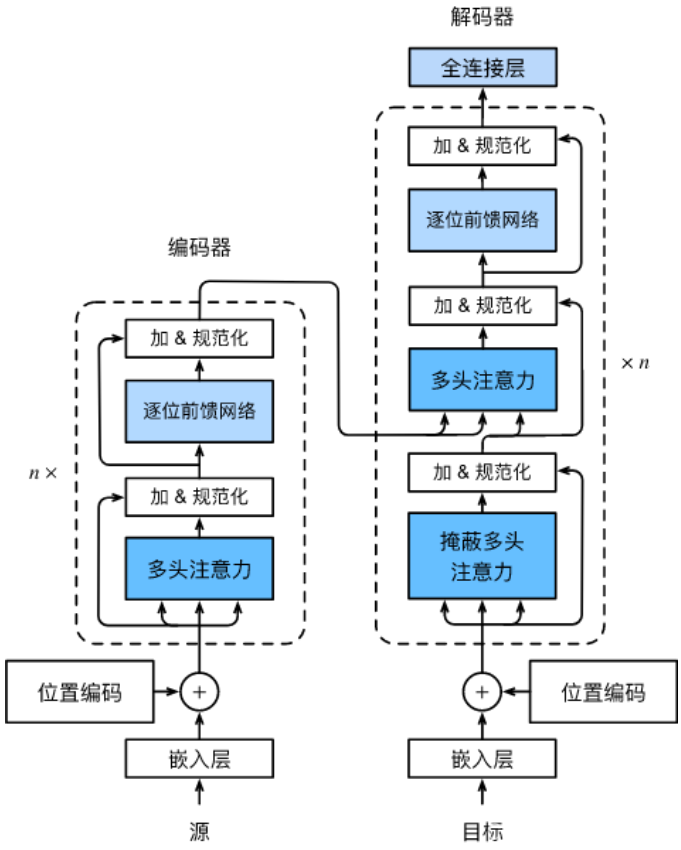


图 2 Transformer 注意力机制模块

Transformer 是由编码器和解码器组成的。Transformer 的编码器和解码器是基于自注意力的模块叠加而成的，源（输入）序列和目标（输出）序列的嵌入表示将加上位置编码，再分别输入到编码器和解码器中。

Transformer 的编码器是由多个相同的层叠加而成的，每个层都有两个子层。第一个子层是多头自注意力汇聚；第二个子层是基于位置的前馈网络 Transformer 解码器也是由多个相同的层叠加而成的，并且层中使用了残差连接和层规范化。除了编码器中描述的两个子层之外，解码器还在这两个子层之间插

入了第三个子层，称为编码器－解码器注意力层。在编码器－解码器注意力中，查询来自前一个解码器层的输出，而键和值来自整个编码器的输出。在解码器自注意力中，查询、键和值都来自上一个解码器层的输出。但是，解码器中的每个位置只能考虑该位置之前的所有位置。这种掩蔽注意力保留了自回归属性，确保预测仅依赖于已生成的输出词元。

2.4 训练过程

将数据集按照训练、验证、测试集为 8：1：1 的比例进行划分。得到训练样本为 3052 个，验证样本为 382 个，测试样本为 381 个。具体初始参数如图 3 所示。

```
lr0（初始学习率）为0.0032
lrf(循环学习率)为0.12
momentum（SGD算法的动量）为0.843
weight decay(权重衰减系数)为0.00036
warmup_epoch（预热学习)为2
warmup momentum(预热学习动量）为0.5
warmup bias lr（预热初始学习率）为0.05
box(iou损失系数）为0.0296
cls(标签损失系数）为0.243
obj(置信度损失系数）为0.301
batchsize（批数量)为 16
pretrain_model（预训练模型）为 YOLOv5s
epoch（迭代次数）为 200
```

图 3 模型初始参数

在训练(train)、测试(val)数据集上的三种损失如图 4、图 5 所示：

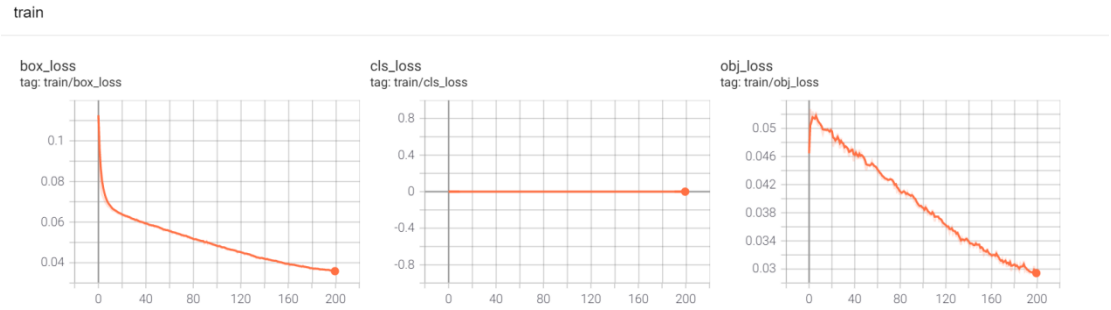


图 4 训练数据集上的损失变化图

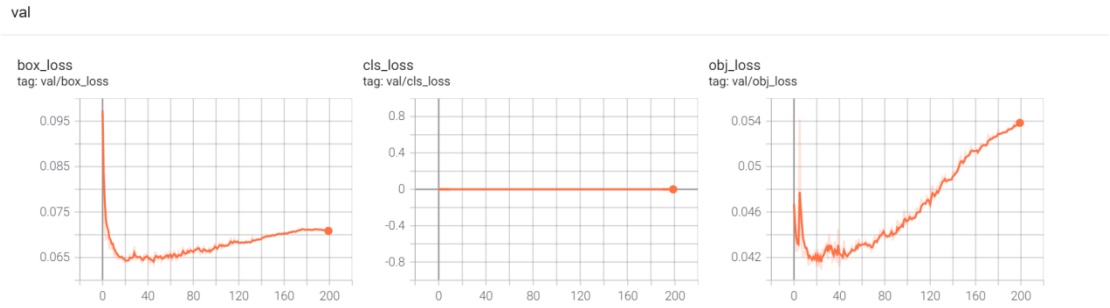


图 5 测试数据集上的损失变化图

其中，box_loss 为边框损失，表示模型预测得到的预测框与标记的真实框之间位置的偏差；cls_loss 为分类损失，表示模型预测得到的类别是否与真实类别一致，本文只有特定目标这一类数据，所以没有分类损失；obj_loss 为置信度损失，置信度代表当前预测框内是否有对象的概率。这三种损失越小，代表检测效果越好。

当 epoch 非常大的时候，训练集的损失下降有所减缓，而验证集的损失，尤其是置信度损失，增长的非常迅速，这标志着模型的过拟合，对于这种问题，采取早停止的策略，综合训练集和测试集的损失，找到并保存最优的迭代次数和迭代模型权重。

2.5 评价指标与结果

1.分类结果

True Positive(TP)：预测和真实结果都为目标；

False Positive(FP)：预测结果为目标，真实结果为非目标；

True Negative(TN)：预测和真实结果都为非目标；

False Negative(FN)：预测结果为非目标，真实结果为目标。

2.精度指标

1）精确率(Precision)：正确预测为正的占全部预测为正的比例：

$$P = \frac{TP}{TP + FP} \tag{1}$$

2）召回率(Recall)：正确预测为正的占全部实际为正的比例：

$$R = \frac{TP}{TP + FN} \tag{2}$$

3）F1 分数(F1 -score)：用来衡量二分类模型精确度的一种指标,它同时兼顾了分类模型的精确率和召回率：

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3}$$

精确率和召回率两个指标存在一定的矛盾，因为提高精确率有可能会降低召回率。F1 分数是解决这个矛盾的综合指标，其取值范围从 0 到 1，F1 分数越接近于 1，则表示模型的精确率和召回率都越高，表明模型的性能相对较好；如果 F1 分数较低，则表示模型的性能相对较差，需要对模型进行进一步的优化。

在测试集上进行检测后，可以看出训练的模型在军事目标的检测方面取得了良好的结果，其在召回率为 31%的情况下，精确率可以达到 58%。在 90%的置信度上，F1 分数达到 0.67。图 6 所示为详细的精确率曲线。

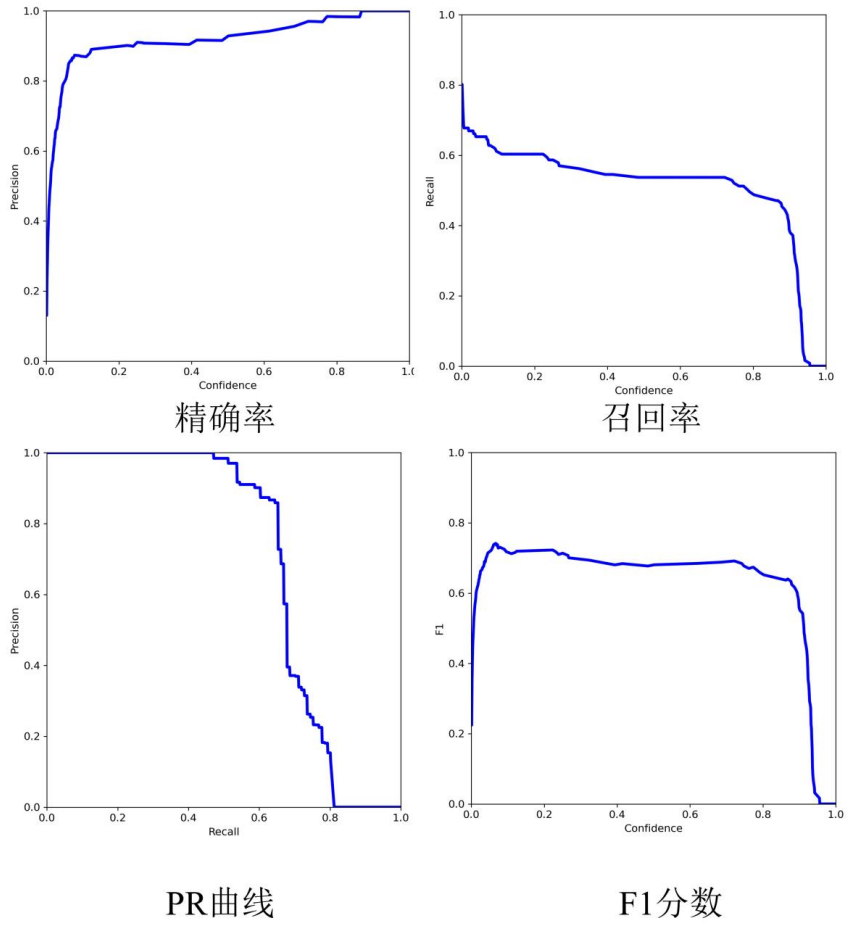


图 6 测试结果精确率曲线

我们在检测集上进行军事目标的检测，其结果的示例如图 7 所示，其中左侧图像为标签样本，右侧图像为预测样本。

图 7 目标遥感数据检测与识别结果

3 结论

通过对目标遥感数据的检测与识别，获得目标的检测结果，利用检测结果来筛选数据集，从而提高目标数据集的质量。数据集中的数据决定了模型学习到的信息，同时也决定了模型预测结果中包含的信息，保证数据集质量可以帮助解释模型的预测结果。

为了继续提升模型的准确率和性能，接下来工作主要有两点：

1）训练过程中出现模型过拟合的问题，除了采用早停止的方法外，需要对数据进行增强，增加训练数据的数量和多样性，可以帮助模型学习到更多的特征和模式，从而减少过拟合的风险。

2）对于目前使用的模型，可以对其进行压缩，量化，剪枝，在不影响精度的情况下削减模型的参数，使其更轻量化，部署在受硬件计算能力限制的移动端。并且可以提升模型的检测速度，实现实时检测。

参考文献：

[1] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Advances in Neural Information Processing Systems, pp. 91-99.

[2] JAEGER P F, KOHL S A A, BICKELHAUPT S, et al. Retina U-Net: Embarrassingly simple exploitation of segmentation supervision for medical object detection[C]//Machine Learning for Health Workshop. [S. l.]: PMLR, 2020: 171-183.

[3] 郭磊，王邱龙，薛伟，郭济.基于改进 YOLOv5 的小目标检测算法[J].电子科技大学学报,2022,第 51 卷(2): 251-258.

[4]付苗苗，邓淼磊，张德贤.基于深度学习和 Transformer 的目标检测算法[J].计算机工程

与应用,2023,第 59 卷(1): 37-48.

[5]XIONG R, YANG Y, HE D, et al. On layer normalization in the transformer architecture[EB/OL]. [2021-10-12]. https://arxiv.org/abs/2002.04745v1.

FusionFL: A statement-level feature fusion based fault localization approach

Yanbo Zhang*, Yawen Wang* †[1], Wenjing Liu*, Dongming Zhu*
*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
†Guangxi Key Laboratory of Cryptography and Information Security, Guilin, Guangxi, China

***Abstract*—Manual fault localization of programs is typically a time-consuming and tedious task. Many automated fault localization approaches have been proposed to reduce development efforts, such as spectrum-based and mutation-based fault localization. However, these methods rarely consider deep semantics embedded in the source code. In this paper, we propose FusionFL, a statement-level feature fusion-based fault localization approach. Specifically, FusionFL generates a separate vector representation for each code statement. Then, FusionFL provides attention weights for each statement by combining spectrum and mutation suspicious scores into a suspicious score matrix. Finally, FusionFL generates suspicious scores by associating each suspicious statement with its corresponding context statements. We evaluate FusionFL on widely used benchmark Defects4j. Experimental results show that FusionFL can effectively fuse multiple types of features and achieves a 45.3% improvement in Top-1 average over baselines. Meanwhile, we conduct extensive experiments on FusionFL to verify its effectiveness in feature fusion.**

***Index Terms*—Fault Localization, Deep Learning, Code Representation**

I. Introduction

Locating and fixing faults is an important step to ensure competent and high-quality software development. The manual fault localization is a lengthy affair that significantly prolongs the development cycle [1]. Researchers propose multiple fault localization (FL) methods to reduce time squandering. The core component of FL is the code coverage matrix, which reflects the execution situation of code lines in different test cases and the pass/fail results of test cases. Based on the code coverage matrix, FL tools calculate the suspiciousness of each code element, i.e., the probability that the code element is faulty. Then FL methods prioritize the most suspicious elements so developers can inspect code in a targeted manner without retrieving entire code segments.

Spectrum-based fault localization (SBFL), which only utilizes the code coverage matrix, is widely used due to its conciseness. The core idea of SBFL is that a code element covered by more failed test cases is more likely to have faults. Based on this assumption, many SBFL techniques have been proposed [2]–[5]. These methods hold the same idea but different formulas for calculating the suspiciousness of code elements. The limitation of SBFL is that the code elements covered by the same failing test cases usually have the same suspicion scores, such as adjacent code executed in sequence or code elements located in the same branch.

To solve the problem of SBFL, mutation-based fault localization (MBFL) is proposed [6]–[8], [11]. MBFL alters the code with mutation operators and compares the code coverage results of the original and mutated code. MBFL approaches utilize formulas analogously to SBFL to calculate the suspiciousness of code elements. Although addressing the issue of SBFL, MBFL suffers from the limitation that mutation operators cannot capture unique mutation contents specific in different code snippets. Only simple mutations can be conducted under the influence of general mutation operators, which lead to inferior localization ability for complex faults.

Multiple approaches concentrate on applying machine learning and deep learning to FL [9]–[23]. Among them, [18]–[20], [22], [23] apply deep learning techniques only on the code coverage matrix, ignoring other relevant information in the code. DeepFL [13] collects multiple features, including spectrum-based suspicious score, mutation-based suspicious score, fault propensity, code complexity, and code similarity. These features are fed into a neural network and operated for fault localization at the method level. In recent years, inspired by code comprehension tasks [24]–[31], researchers have utilized source code semantics for fault localization. Deeprl4fl [15] incorporates code coverage matrix, code semantics, and data dependencies to locate faults. However, Deeprl4fl only applies a fully connected layer for statement-level FL to obtain the code semantics and loses the deep semantics. Transfer [16] pre-trains 11 binary classifiers to extract deep semantics of code. Nevertheless, it does not fully incorporate spectral-based scores, mutation-based scores, and semantic features. The three features are only fed into a multilayer perceptron to obtain the suspicion score. This results in inadequate representation learning. Meanwhile, the contextual information on spectrum-based and mutation-based scores is ignored.

Code elements in FL research can be classified into three granularities: class, method, and statement. Class-level and method-level fault localization results still need developers to locate faulty components for a lengthy time. Moreover, the research on extracting contextual information at class and method granularity is still an active area of investigation. In this paper, we focus on statement-level fault localization.

We propose FusionFL, a statement-level fault localization approach based on the fusion of code semantics, SBFL, and

[1]Corresponding author, E-mail:wangyawen@bupt.edu.cn

MBFL features. FusionFL is inspired by the multimodal idea and fuses code semantics with spectrum-based and mutation-based suspicious scores in one neural network. The fusion can fully explore the impact between the three types of features. Specifically, FusionFL concatenates each statement's spectrum-based and mutation-based scores into a suspicious score matrix, serving as a priori knowledge for code semantics. Meanwhile, source code is sliced into statements to assemble a statement matrix. The hidden representation of each statement is learned using long short-term memory (LSTM) [32]. Then, the attention mechanism [33] captures the correlations within the suspicious score matrix and assigns weights to the statement representation vectors. Finally, the second LSTM network is applied to learn the deep semantics within the contextual statements and provide the suspicion score.

We conduct extensive experiments on 395 real software faults from the widely used Defects4J benchmark (V1.2.0) [34]. The experimental results show that FusionFL can significantly outperform the baselines (e.g., localizing 55 faults within Top-1, 16 more than DeepFL). The study also investigates the impact of different configurations of input and model, including (1) the length of contextual statements substantially affecting the localization results; (2) hinge loss is more influential than binary cross entropy loss for FusionFL.

The main contributions of this paper are as follows:

- We offer a new fusion scheme incorporating code semantics with spectrum-based and mutation-based suspicious scores. The scheme allows a fine-grained combination of features at the statement level.
- We propose a new deep learning-based fault localization method FusionFL. We tackle the inadequate feature fusion problem and introduce a detailed implementation of our method.
- We compare FusionFL with the most recent approaches on the widely used Defects4j benchmark. The experimental results indicate that our approach is effective. We also investigate performance of FusionFL under different model configurations.

The rest of this paper is organized as follows. In the next section, we present our methodology. Section III tells the experiment setup, explains the evaluation results, and discusses the performance of the model. Section IV stresses the related work to our approach.In Section V, we conclude this paper.

II. METHODOLOGY

A. Overview

The general structure of FusionFL is shown in Fig 1. It contains two main components: The first part is feature extraction (Section II-B), which is designed to execute the test suites, mutate source code and generate suspicious score matrix for the code. The second part is representation learning (Section II-C), which preprocesses the code blocks selected from source code and uses two LSTM networks at statement/block level to learn the deep semantics. The representation learning model fuses the suspicious score matrix with code semantics and provides suspicious scores. In this section, we also discuss the loss function of FusionFL (Section II-D).

B. Feature Extraction

Spectrum-based Features. Spectrum-based fault localization (SBFL) is one of the most widely used fault localization methods. The core of SBFL is the code coverage matrix. However, since the code coverage matrix is sparse, deep learning methods have difficulty capturing the information in the code coverage matrix. Recent studies have found that the suspicious scores derived from SBFL tools can be fed as partial input to train learning-to-rank (L2R) models [35]. Such incorporation can promote the localization results of L2R approaches [11], [13], [19]. In FusionFL, we also utilize the suspicion scores given by SBFL tools as the input to the representation model. SBFL needs to count: T_p/T_f, the number of pass/fail test cases; $T_p(e)/T_f(e)$, the number of pass/fail test cases executing code element e; $T_p(\overline{e})/T_f(\overline{e})$, the number of pass/fail test cases not executing e. The SBFL tools use the above elements to calculate the suspiciousness of each code element according to the suspiciousness formulas. We employ three widely used SBFL formulas in FusionFL: Ochiai, DStar, and Tarantula. The three suspiciousness calculation formulas are as follows:

$$Ochiai : Sus(e) = \frac{T_f(e)}{\sqrt{T_f(T_f(e) + T_p(e))}} \tag{1}$$

$$DStar : Sus(e) = \frac{T_f(e)^*}{T_p(e) + T_f(\overline{e})} \tag{2}$$

$$Tarantula : Sus(e) = \frac{T_f(e)/T_f}{T_f(e)/T_f + T_p(e)/T_p} \tag{3}$$

Mutation-based Features. Mutation-based fault localization (MBFL) alters the code by mutants and locates faults by comparing code coverage matrix changes before and after the mutations. Similar to SBFL, the code element suspicious scores obtained by the MBFL tools can also be fed to L2R models as input [11], [19]. Since Metallaxis-FL performs better than Muse at statement-level fault localization [15], we refer to Metallaxis-FL implementation in Transfer and apply the SBFL formula to calculate the suspiciousness of each statement. We select the most frequently used Ochiai for suspiciousness calculation.

$$Sus(e) = \max_{m \in M(e)} \frac{T_f^{(m)}(e)}{\sqrt{T_f^{(m)} \cdot (T_f^{(m)}(e) + T_p^{(m)}(e))}} \tag{4}$$

where $M(e)$ represents all the mutation factors in code element e, $T_f^{(m)}$ represents the number of originally failed tests that affect by mutant m, $T_f^{(m)}(e)/T_p^{(m)}(e)$ represents the number of originally failed/passed tests which are affected by m and cover e. TraPt demonstrates that different granularity of error message changes can impact localization performance. Inspired by this, we utilize four granularities of error message changes in FusionFL: (1) Type1: pass/fail information, (2) Type2: exception type information, (3) Type3: exception type,

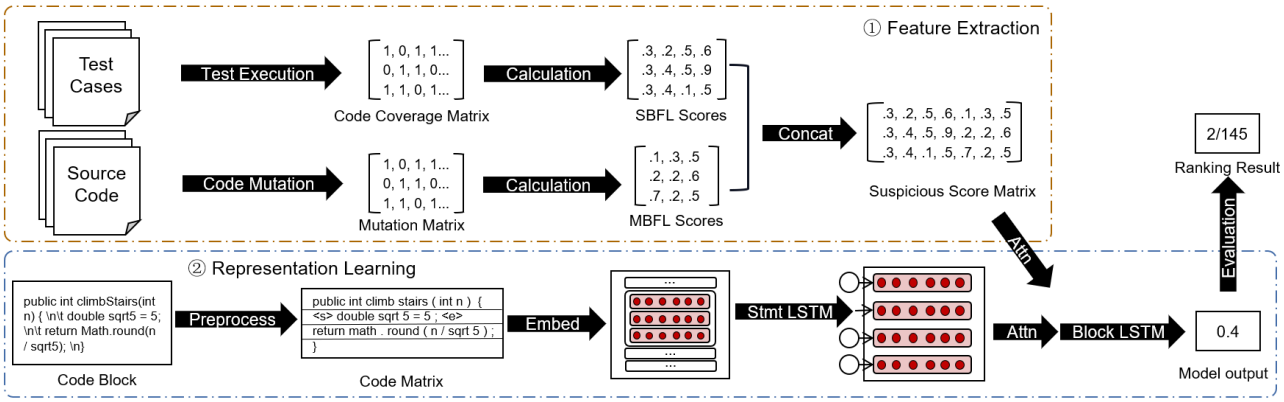


Fig. 1. Structure of FusionFL.

and (4) Type4: exception type, message and the full stack trace of the exception. Mutation-based suspicious score at each granularity is computed using Eq 4.

Suspicious Score Matrix. After the spectrum-based and mutation-based suspicious scores are obtained, the scores are concatenated to acquire the suspicious score matrix: $S \in \mathbb{R}^{n\times 7}$. where n represents the length of the context containing the suspicious statement, 7 means seven scores for each statement (four SBFL scores and three MBFL scores).

TABLE I
AN EXAMPLE PAIR OF POSITIVE AND NEGATIVE SAMPLES

	Positive Sample	**Negative Sample**
1	boolean Eq(String s1,String s2){	boolean Eq(String s1,String s2){
2	⟨***s***⟩ if(s1==s2){⟨**e**⟩	if(s1==s2){
3	return true;	⟨***s***⟩ return true; ⟨***e***⟩
4	}	}
5	return false;	return false;
6	}	}

C. Representation Learning

Source Code Matrix. Real-world projects contain an average of 30 thousand lines [36]. The presence of uncovered statements in the training set can increase sparsity, detrimentally impacting model performance. Therefore, our dataset consists of only suspicious statements covered by test cases. To ensure the surrounding semantics of the code statements are fully considered, we add contextual code to the suspicious statements, i.e., the method that the suspicious statement belongs. We called the combination of a suspicious statement and its contextual code a **code block**. To indicate the location of the suspicious statement in the code block, we add start and stop symbols ⟨start⟩ and ⟨end⟩ to the front and the end of the suspicious statement. Similarly, the negative samples are constructed from all the covered non-faulty statements and their contexts. An example of a positive and negative sample is given in Table I. In the positive sample, there is a fault in the *if* statement: comparing strings for equality should use the *equals* method rather than ==.

In Deeprl4fl, each token is placed in word2vec [37] without tokenization. Source code includes extensive words, such as custom variable names and string values, which cause the out-of-vocabulary (OOV) problem and reduce the generalization ability of the model [38]. We employ the following normalization process to alleviate the OOV problem. For user-defined variable names, we split each variable name according to the camel naming and underscore naming rules, keep only letters, and use the split word as a token. For example, "toString" is split into "to" and "string". For the content of a string, we employ a flag *stringData* to replace the content. We concatenate the processed tokens as a token sequence, which represents the unprocessed statement.

After the process, all the token sequences of the code block are concatenated to generate a statement sequence representing the code block. LSTM requires the same shape inputs for different time steps. Therefore, the length of different processed token sequences and the length of contextual codes for different suspicious statements are normalized. Therefore, we normalize the length of the token sequence and statement sequence. Assume l_s be the length of the statement sequence and l_t be the length of the token sequence. If the length does not reach the threshold, paddings are added to fill the sequence. Finally, we obtain input source code matrix $C \in \mathbb{R}^{l_s\times l_t}$. Algorithm 1 gives the pseudo-code for processing.

Word Embedding. Pre-trained word2vec is used for token vectorization. We consider a statement as a sentence and feed all statements into word2vec to obtain a word bag $W_e \in \mathbb{R}^{V\times d}$, where V represents the number of words and d represents the embedding dimension. For a token sequence $< t_1, t_2, ..., t_{l_t} >$, the vector representation $< e_1, e_2, ..., e_{l_t} >$ is obtained after word embedding. The transformation rule is:

$$e_i = W_e^T o_i \tag{5}$$

where o_i represents the one-hot vector of t_i.

Statement-level LSTM. Fold2Vec [24] mentioned that treating a method as a sentence input to the model is inadequate for capturing hidden information. Instead, slicing

Algorithm 1 Input Preprocessing

Require: the project P to be preprocessed, l_s, l_t.
Ensure: preprocessed data set D

```
for data ∈ P do
    code, matrix ← split(data)
    for line ∈ code do
        tokens ← tokenize(line)
        for token ∈ tokens do
            if isString(token) then
                newToken ←' stringData'
            else
                newTokens ← splitToken(newToken)
            end if
            tokenSeq ← concat(newToken)
        end for
        stmtSeq ← concat(tokenSeq)
    end for
    codeMT, scoreMT ← cutOrPad(stmtSeq, matrix, l_s, l_t)
end for
D ← concat(codeMT, scoreMT)
return D
```

the method into statements and combining the statement-level hidden states can sufficiently consider the impact of different statements on the output. Inspired by Fold2Vec, we employ two long short-term memory (LSTM) models to extract statement-level and block-level semantics. LSTM has better robustness and more substantial generalization capability than other RNN models in software engineering [27]. To enhance the semantics of the code and to capture more information about code dependencies, we employ a bidirectional LSTM network. the output h_t^s of the Bi-LSTM network at time t is a concatenating of hidden states in two directions:

$$\begin{gathered} \overrightarrow{h_t^s} = \overrightarrow{LSTM^s}(e_t, \overrightarrow{h_{t-1}^s}) \quad \overleftarrow{h_t^s} = \overleftarrow{LSTM^s}(e_t, \overleftarrow{h_{t+1}^s}) \\ h_t^s = [\overrightarrow{h_t^s}, \overleftarrow{h_t^s}] \end{gathered} \tag{6}$$

The output of the statement-level LSTM is $H^s = [h_1^s, h_2^s ..... h_{l_t}^s], H \in \mathbb{R}^{l_t \times 2k}$, k is the hidden state dimension of the LSTM. To obtain the most valuable feature of the statement, we perform max pooling on H^s. Specifically, for each hidden state dimension $j \in 2k$, maximal feature r_j^s is selected from all time steps and splice all maximal features to obtain the semantic representation of the statement:

$$r_j^s = \max_{1 \le t \le l_t} h_{tj}^s \tag{7}$$

$$R^s = [r_1^s, r_2^s, ..., r_{2k}^s] \tag{8}$$

The code block can be represented by splicing all the statement semantics: $B^s = [R_1^s, R_2^s, ..., R_{l_s}^s]$.

Attention Mechanism. It is unattainable to capture the statement-level features at a fine-grained precision by code semantics alone (see III-D). As the attention mechanism has shown the capability for feature fusion [39], we leverage the self-attention mechanism to incorporate spectrum-based features, mutation-based features and code semantics. Self-attention consists of three matrices K, Q, V. It uses Q and V to calculate the attention weight between different tokens and then assigns the weight to K. In FusionFL, Q and V are the suspicious score matrix S, and K is the statement representation matrix B^s. Using the attention mechanism to fuse features can: (1) introduce a context for the suspicious score matrix and learn the relevance of the suspicious score thoroughly; (2) assign the suspicious score to the corresponding statement representation separately; (3) introduce the suspicious score as a priori knowledge to code representation for further learning. The equation of attention calculation is as follows:

$$score = \frac{S \cdot S^T}{\sqrt{7}} \tag{9}$$

$$B'^s = softmax(score) \cdot B^s \tag{10}$$

Block-level LSTM. After obtaining weighted statement representation matrix B_{fusion}^s, we employ another LSTM to understand the semantics of the code block. We also utilize max pooling to obtain the maximum representation after obtaining the hidden states from LSTM.

$$\begin{gathered} \overrightarrow{h_t^b} = \overrightarrow{LSTM^b}(R_t'^s, \overrightarrow{h_{t-1}^b}) \quad \overleftarrow{h_t^b} = \overleftarrow{LSTM^b}(R_t'^s, \overleftarrow{h_{t+1}^b}) \\ h_t^b = [\overrightarrow{h_t^b}, \overleftarrow{h_t^b}] \end{gathered} \tag{11}$$

$$r_j^b = \max_{1 \le t \le l_s} h_{tj}^b \tag{12}$$

$$R^b = [r_1^b, r_2^b, ..., r_{2k}^b] \tag{13}$$

Ranking. After the code representation R^b is obtained, it is fed into the fully connected layer and the *softmax* function is employed to get the model output. The output of the model represents the possibility that the labelled statement is faulty. All the statements covered by the test cases under a particular faulty version are fed into the model and the probability of the statements being faulty is obtained. By ranking the probabilities, we can attain the relative order of the faulty statements among all the suspicious statements.

D. Loss Function

The computation of the loss defines the learning objective of the neural network. In FusionFL, we discuss the effect of two loss functions: hinge loss and binary cross entropy loss(BCE).

Hinge. The premise of the fault localization problem is that there must be at least one faulty statement in the code. From this perspective, the target of the FL is to place the faulty statement at the top of the ranking list. i.e., the faulty statement needs a higher suspicious score than the other statements. Therefore, we adapt the *pairwise* function to FusionFL.

The objective of the hinge loss function is to maximize the interval plane of positive and negative samples. The model output for FL is not just ± 1, but the probability of the presence of a fault. Therefore, we need to maximize the distance between positive and negative samples. The hinge loss function is defined under the ranking problem as Eq 14.

$$\mathcal{L}(\theta) = \sum \max(0, \tau - (\hat{y_p} - \hat{y_n})) \tag{14}$$

where τ is a constant, $\hat{y_p}$ represents the predicted value of positive samples and $\hat{y_n}$ is the predicted value of negative samples.

Binary Cross Entropy. In FusionFL, the model inferencing and the suspicion ranking are two independent steps. The

TABLE II
SUMMARY OF DEFECTS4J V1.2.0

Identifier	Project name	# of bugs	LoC(k)	Test
Chart	JFreeChart	26	96	2205
Closure	Closure compiler	133	90	7927
Math	Apache commons-math	106	85	3602
Lang	Apache commons-lang	65	22	2245
Time	Joda-Time	27	28	4130
Mokito	Mokito	38	67	1075
Total	-	395	388	21184

TABLE III
SETUPS FOR EXPERIMENT

hyper-parameter	value	initial cnf.	option-class
no. stmts.	40	30	20,30,40,45
no. token per stmt.	20	20	15,20,25
batch size	64	32	16,32,64,128
loss	hinge	hinge	hinge,BCE
optimizer	Adam [44]	Adam	-
learning rate	0.01	0.001	0.05,0.01,0.005,0.001
embed. size	64	32	32,64,128
τ in hinge loss	0.1	0.1	0.05, 0.1, 0.15, 0.2

suspicion ranking is not involved in the model training phase. Therefore, FL can also be regarded as a classification problem when training the model. Binary cross entropy (BCE) is an ordinary loss function for classification in deep learning. For n samples, BCE loss function is defined as Eq 15.

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{n} y_i \log(p_i) + (1-y_i)\log(1-p_i) \qquad (15)$$

III. EVALUATION

This work focuses on the following four research questions.

RQ1. How does FusionFL perform in statement-level fault localization?

RQ2. How effective are the main components of FusionFL?

RQ3. How do different code block sizes affect FusionFL results?

RQ4. How do different loss functions affect FusionFL results?

A. Benchmark Dataset

We use Defects4j (v.1.2.0) [34] in our experiments, which is a dataset widely used for verification of fault localization tasks [11], [13], [15], [16], [40], [41]. Defects4j can export a separate copy of a faulty version that contains only the test suites relevant to the fault. Faults between different versions are independent in the test suites. An overview of Defects4j is shown in Table II.

B. Experiment Setting

We utilize Gzoltar [42] and PIT [43] to gather spectrum-based and mutation-based features. We apply the same technique as Trapt [11] and DeepFL [13] to adapt PIT. For the hyperparameter selection of the model, grid search is applied to verify the effect of the model under different parameters. To avoid experimental randomness, 10-fold cross-validation is employed. The specific hyperparameter selection is presented in the Table III. All experiments are conducted on a 64-bit, 20-core 2.1 GHz CPU, 502 G RAM, NVIDIA Tesla V100 GPU with 32 GB memory server with Ubuntu 16.04 operating system.

C. Evaluation Metric

We use the following evaluation metrics which are commonly used in fault localization studies [11], [13], [15], [40]:

Top-K: is the number of fault versions with at least one faulty element present in the first K position. According to the finding [45], only 16.99% of developers check the code element that is located after Top-5. Based on the previous work [13], [40], [41], we choose Top-1/3/5 for evaluation.

Mean First Rank (MFR): Some fault versions contain multiple faulty code elements. It is essential to find the most suspicious one because the rest of the fault elements may be found sequentially after this. MFR reflects the average ranking of the most suspicious faulty element in all fault versions.

Mean Average Rank (MAR): We compute the rank average of all fault elements in each faulty version. For a project, MAR represents the mean of the average rank of all its fault versions.

D. Result and Discussion

RQ1. FusionFL's performance in statement-level fault localization. To investigate the effectiveness of FusionFL, the following statement-level fault localization methods are chosen:

- Three SBFL tools: Ochiai [2], Dstar [3] and Tarantula [4]
- One MBFL tool: Metallaxis [8]
- Three deep learning-based tools: RBF [18], DeepFL [13] and Transfer [16]
- Two recent tools: Aeneas (data augmentation) [40] and Feature-FL (feature metrics) [41]

For the ablation study, we modified the three compared methods as follows. Aeneas is a data augmentation approach with different suspiciousness formulas and different formulas may lead to slight variations in the results. In this paper, we select the best result of Aeneas for our analysis. The open-source code of DeepFL is designed for method-level fault localization, and we modify it to meet our requirements. Transfer utilizes an additional dataset for pre-training the defect type classifiers. However, many studies have demonstrated that pre-trained models generally outperform non-pretrained models [38], [46]. Among the compared approaches chosen in this paper, Transfer is the only method that utilizes pre-training. Therefore, to compare only the learning ability of different models, Transfer has been modified to a non-pretrained mode (Transfer'). The transformer model [33] can acquire abundant linguistic knowledge and patterns, exhibiting solid contextual awareness. However, achieving adequate progress in fusing semantics and suspicious scores using the transformer is still challenging. Meanwhile, existing datasets are insufficient to support the training of transformer. Thus, we do not consider it a comparison method.

TABLE IV
RESULTS OF COMPARATIVE STUDY FOR STATEMENTLEVEL FAULT LOCALIZATION.

Approach	Top-1	Top-3	Top-5	MAR	MFR
Ochiai	30	78	95	323.49	634.88
Dstar	29	79	94	366.73	674.05
Tarantula	25	69	89	371.56	698.16
Metallaxis	13	36	63	512.28	649.41
RBF	12	37	52	135.24	344.28
DeepFL	39	114	129	128.02	170.46
Aeneas	46	99	131	332.54	594.88
Feature-FL	37	90	112	-	-
Transfer'	**61**	112	139	130.76	167.7
FusionFL	55	**116**	**141**	**114.88**	**162.27**

TABLE V
CONTRIBUTION OF EACH FUSIONFL COMPONENT TO POSITIONING ACCURACY.

Variants	Top-1	Top-3	Top-5	MAR	MFR
FusionFL	55	116	141	114.88	162.27
w/o statement-LSTM	47	76	99	149.25	219.24
w/o block-LSTM	29	65	76	170.73	239.63
w/o SBFL scores	49	84	109	135.19	192.93
w/o MBFL scores	34	71	89	180.78	244.26
w/o metric	27	58	72	256.09	355.62

The detailed results of the experiment are presented in Table IV. The table shows that except Top-1, FusionFL outperforms all compared fault localization methods. Compared to the three SBFL tools, the increase in Top-1, 3, and 5 is 25+, 36+, and 46+, respectively. Meanwhile, the improvements in MAR and MFR are 64.48%+ and 74.4%+. Compared with the MBFL tool Metallaxis, the increased values of Top-1, 3, 5 are 42, 79, and 78, and the boosts of MAR and MFR are 77.5% and 75.01%, respectively. Compared to SBFL and MBFL approaches, FusionFL's improvement is significant. SBFL and MBFL approaches only consider the dynamic information of the code without considering the code semantics.

Compared to the two deep learning-based approaches RBF and DeepFL, Top-1, 3, 5 have an increase of 16+, 2+, and 12+, and MAR and MFR have an improvement of 10.26%+ and 4.8%+, respectively. Compared to Aeneas and Feature-FL, the increases for Top-1, 3, 5 are 9+, 17+, and 10+, with boosts of 65.4%+ and 72.72%+ for MAR and MFR. These methods do not consider code semantics. Meanwhile, they use SBFL and MBFL in a straightforward manner, which leads to underutilization of the features.

Compared to the unpretrained Transfer, the increase in Top-3 and 5 is 4 and 2, respectively, and the improvements in MAR and MFR are 12.1% and 3.4%. Compared to FusionFL, Transfer achieves weaker performance by learning semantics and fuse features in two stages. By incorporating code semantics with suspicious score matrix and scoring the fused features, FusionFL can significantly increase effectiveness of FL.

RQ2. Contribution of each FusionFL component to positioning accuracy. In this RQ, we aim to explore the effectiveness of each main component in FusionFL. The central of FusionFL's architecture consists of two parts: code semantics and suspicious score matrix. The code semantics part can be divided into statement semantic learning and block semantic learning. The suspicious score matrix includes two types of scores: SBFL and MBFL scores. Therefore, we design the following five experimental combinations: (1) replacing statement-level LSTM with averaging token vectors in each statement (w/o statement-LSTM); (2) replacing block-level LSTM with averaging each statement vector (w/o block-LSTM); (3) removing three SBFL scores (w/o SBFL scores); (4) removing four MBFL scores (w/o MBFL scores); (5) eliminating the entire suspicious score matrix (w/o metric).

The results in Table V demonstrate that both the code representation learning and score fusion mechanisms of the model provide positive contributions. Specifically, for representation learning, statement-level LSTM helps FusionFL locate 8 (17.8%) more Top-1 faults, and improves MAR by 29.91% and MFR by 25.98%. Using block-level LSTM helps locate 26 (47.2%) more Top-1 faults, and improves MAR by 32.71% and MFR by 32.28%. For the suspicious score matrix, involved SBFL scores increase the number of Top-1 faults from 49 to 55. Involved MBFL scores contribute to 21 (38.18%) more Top-1 faults and improve MAR by 36.45% and MFR by 33.5%. Compared to the model without using the suspicious score matrix, FusionFL locates 28 (50.91%) more Top-1 faults and improves MAR and MFR by 55.14% and 54.36%, respectively.

RQ3. Impact of different code block sizes on FusionFL. In this RQ, we discuss the impact of different number of tokens per line and different number of lines on the results of FusionFL. Specifically, we select token quantities per line (l_t) of 15, 20, and 25. And we select line numbers (l_s) of 20, 30, 40, and 45. We evaluate the performance of FusionFL under different l_t and l_s. The results are as in Table VI.

As can be observed from Table VI, there is a gradual improvement as l_s increases from 20 to 40. However, the trend reverses and worsens at 45 code lines. This phenomenon can be attributed to the fact that increasing l_s from 20 to 40 results in the model's absorption of more contextual code semantics. As l_s continues to increase, more non-relevant code is introduced into the context around the marked lines, thereby affecting the model's comprehension ability. Therefore, when l_s is 45, all metrics exhibit an overall decreasing trend.

Similar trends can also be observed by increasing l_t. Specifically, increasing l_t from 15 to 20 leads to an upward trend in all metrics, while changing l_t from 20 to 25 results in a downward trend. This phenomenon is caused by excessive padding in each line of code, as most lines of code do not exceed 20 tokens after splitting.

RQ4. Impact of different Loss functions on FusionFL results. We compared the performance of FusionFL with hinge loss and binary cross-entropy loss (BCE) on all metrics. In Fig 2, lines with different colors and shapes represent different losses. Each subplot shows the trend of different metrics under different epochs and loss functions. The figure shows that

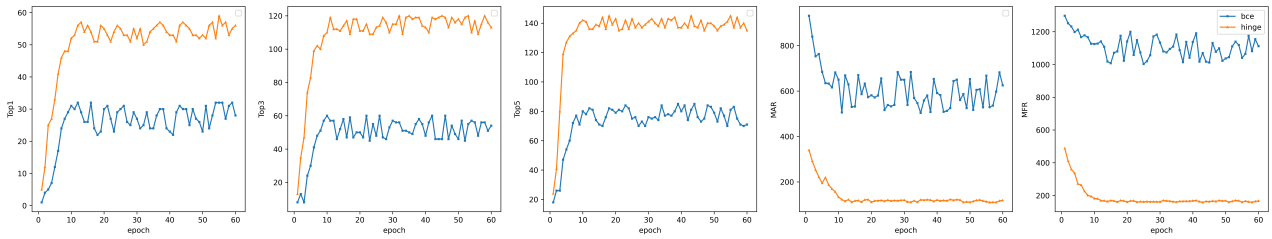


Fig. 2. Impact of different Loss functions.

TABLE VI
IMPACT OF DIFFERENT CODE BLOCK SIZES ON FUSIONFL.

	l_t \ l_s	20	30	40	45
Top-1	15	50	53	53	48
	20	59	51	55	49
	25	51	46	53	49
Top-3	15	94	95	101	87
	20	95	106	115	102
	25	88	89	97	94
Top-5	15	118	121	133	108
	20	114	131	141	134
	25	112	119	123	121
MFR	15	175.27	158.32	121.71	170.72
	20	165.67	124.52	114.88	140.87
	25	142.32	141.15	127.49	146.12
MAR	15	222.22	181.31	174.82	195.49
	20	211.83	175.39	162.27	180.39
	25	203.27	189.52	173.87	195.81

the Top-1, 3, 5 metrics have similar trends with both hinge loss and BCE loss, and they stabilize around the 10th epoch. However, the validation results with BCE loss are less than half of those with hinge loss. Regarding the MAR and MFR metrics, using BCE loss significantly reduces the average effectiveness of fault localization. One possible reason is that the learning objective of BCE loss is to accurately judge whether a code element is faulty, whereas fault localization research requires distinguishing the boundaries between faulty and non-faulty elements. Therefore, using BCE loss may refine the characteristics of specific faulty codes while ignoring the features of others.

Why it works. The previous fault localization methods considering semantics either simply concatenated word embeddings or only combined the suspicious scores obtained from semantic models with SBFL and MBFL scores. In contrast, our approach employs two LSTM networks to extract the semantics of each statement and the code segment separately, which can comprehensively learn fine-grained semantic information within the code. Furthermore, we introduce the self-attention mechanism for the first time to capture the relationship between each statement's SBFL and MBFL scores. By applying the learned relationship to the statement semantics, the attention mechanism enhances the dependency between code and suspicious scores, as well as the modelling ability of contextual information. Compared to approaches that employ CNN and RNN to learn code coverage matrix, the attention mechanism focus on global correlations and addresses the problem of long-distance information loss.

Threats to validity. The external validity of our research is threatened by programming languages, as we have only verified the effectiveness of our methods in Java rather than in other programming languages. However, the methodology used in our data processing can be applied to other languages. Another threat to external validity is the quality of the test code. To address this issue, we chose Defects4j as a validation dataset, a widely used database constructed from actual errors. The main internal validity threat involves potential errors in data processing and model construction. We employed state-of-the-art tools and frameworks such as PIT, Pytorch, and ASM to mitigate these threats. A threat to construct validity is that the measurement tools may need to reflect real-world situations. We reduced this threat by using widely adopted metrics such as Top-K, MAR, and MFR, which are widely used by other works [11], [13], [15], [16], [41].

IV. RELATED WORK

A. Fault Localization

Traditional fault localization methods are mainly classified as spectrum-based and mutation-based. Spectrum-based fault localization (SBFL) [2]–[5], [47]–[52] is based on a code coverage matrix in which each row represents a code statement and each column represents a test case. SBFL requires statistics on the coverage of each code element by different test cases and its pass/fail results under different test cases. These data are represented through the code coverage matrix. SBFL tools analyze the code coverage matrix to generate a ranked list in descending order to reflect the suspiciousness level of different code elements.

Mutation-based fault localization (MBFL) [6]–[8], [53]–[55] locates faults based on mutants. The first proposed usable mutation technique for fault localization is Metallaxis-FL [8], which assumes that mutants of the same program element in

a program always have similar behavior and treat faults as mutants. It captures the complete pass/fail output message for code running in test cases and considers that the code element at the location where the mutant impacts the failed test case is more likely to cause the fault. On top of this, Metallaxis-FL considers the mutants that affect the test results as covered elements and the rest as uncovered elements and calculates the suspicion using the SBFL formula. Muse [6], conversely, considers mutated incorrect code elements would cause more failed test cases to turn into pass than mutated correct ones. Muse also proposes a new suspiciousness formula based on the change in the pass/fail test case ratio. Trapt [11] argues that Metallaxis-FL and Muse are based on incomplete change information and proposes using four granularities of failure information. Based on this, Trapt combined SBFL scores and used Learning to Rank [35] technique to rank the code elements.

Recently, machine learning and deep learning-based methods have been proposed [9], [10], [13]–[23], [56]. However, many of these works need to exploit the ability of the model to extract deep information. For example, CNN-FL only applies convolutional neural networks to the code coverage matrix to learn the information in the matrix. DeepFL incorporates more than 200 features, but many do not serve as critical fault factors (e.g. the number of annotations.) Furthermore, it ignores the deep semantics in the code.

Many innovative pieces of researches have been explored. Variable-fl [57] proposed a fault localization method with variable granularity through enhanced decision trees. Aeneas [40] augmented dataset by conditional variational autoencoder (CVAE) [58] to issue the category imbalance problem. [59] models code semantics by probabilistic information, using both static and dynamic information to collect features. Feature-fl [41] evaluates suspicious code elements by abstracting program behavior into feature values and associating features with failures.

B. Code Representation

Recent achievements in natural language processing have led to the application of deep learning techniques to software engineering and program analysis tasks, such as code cloning [60], code summarization [61]–[63], code classification [64], and bug detection [65]. Reflecting the structure of the code in the model is a vital issue. CodeBert [46], CodeT5 [38], CodeXGLUE [67], PLBART [66] all input the source code directly into the model without considering its unique structure. U Alon et al. successively proposed Code2Seq [27] and Code2Vec [25], both of which use random walking abstract syntax tree (AST) to generate inputs. Code2Seq uses LSTM to take different AST path representations at different time steps as inputs. Code2Vec applies path attention to consider all AST path representations simultaneously. [68] converts the AST to linear while using the position matrix to represent the relationships of the nodes in the AST. Fold2vec [24] considers the unique features of each statement, first learns the representation of each statement of terminal tokens and non-terminal tokens using global attention and LSTM networks, and then captures the correlation of different rows by self-attention to obtain the final code representation.

V. Conclusion

In this paper, we propose FusionFL, a new deep learning-based statement-level fault localization method with feature fusion. We provides a new feature fusion schema, which leverages an attention mechanism to integrate code semantics with SBFL and MBFL scores, thereby enhancing the efficiency of fault localization. Furthermore, we investigate the impact of different numbers of code lines and tokens on FusionFL. Experimental results on 395 real faults from the widely-used Defects4J benchmark demonstrate that FusionFL can outperform all the fault localization baselines.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. U1736110), the Basic Scientific Research Expenses Program of Universities directly under Inner Mongolia Autonomous Region (Grant No. JY20220273), the Natural Science Foundation of Inner Mongolia Autonomous Region(Grant No. 2023LHMS06023), and Guangxi Key Laboratory of Cryptography and Information Security (Grant No.GCIS202103).

References

[1] C. S. Wright and T. A. Zia, "A quantitative analysis into the economics of correcting software bugs," in *Computational Intelligence in Security for Information Systems: 4th International Conference, CISIS 2011, Held at IWANN 2011, Torremolinos-Málaga, Spain, June 8-10, 2011. Proceedings*. Springer, 2011, pp. 198–205.

[2] R. Abreu, P. Zoeteweij, and A. J. Van Gemund, "On the accuracy of spectrum-based fault localization," in *Testing: Academic and industrial conference practice and research techniques-MUTATION (TAICPART-MUTATION 2007)*. IEEE, 2007, pp. 89–98.

[3] W. E. Wong, V. Debroy, R. Gao, and Y. Li, "The dstar method for effective software fault localization," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 290–308, 2013.

[4] J. A. Jones, M. J. Harrold, and J. Stasko, "Visualization of test information to assist fault localization," in *Proceedings of the 24th international conference on Software engineering*, 2002, pp. 467–477.

[5] W. E. Wong, V. Debroy, and B. Choi, "A family of code coverage-based heuristics for effective fault localization," *Journal of Systems and Software*, vol. 83, no. 2, pp. 188–208, 2010.

[6] S. Moon, Y. Kim, M. Kim, and S. Yoo, "Ask the mutants: Mutating faulty programs for fault localization," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*. IEEE, 2014, pp. 153–162.

[7] M. Papadakis and Y. Le Traon, "Using mutants to locate" unknown" faults," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 691–700.

[8] M. Papadakis and Y. Le Traon, "Metallaxis-fl: mutation-based fault localization," *Software Testing, Verification and Reliability*, vol. 25, no. 5-7, pp. 605–628, 2015.

[9] T.-D. B. Le, D. Lo, C. Le Goues, and L. Grunske, "A learning-to-rank based fault localization approach using likely invariants," in *Proceedings of the 25th international symposium on software testing and analysis*, 2016, pp. 177–188.

[10] J. Xuan and M. Monperrus, "Learning to combine multiple ranking metrics for fault localization," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 191–200.

[11] X. Li and L. Zhang, "Transforming programs and tests in tandem for fault localization," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 1–30, 2017.

[12] J. Sohn and S. Yoo, "Fluccs: Using code and change metrics to improve fault localization," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2017, pp. 273–283.

[13] X. Li, W. Li, Y. Zhang, and L. Zhang, "Deepfl: Integrating multiple fault diagnosis dimensions for deep fault localization," in *Proceedings of the 28th ACM SIGSOFT international symposium on software testing and analysis*, 2019, pp. 169–180.

[14] Y. Lou, Q. Zhu, J. Dong, X. Li, Z. Sun, D. Hao, L. Zhang, and L. Zhang, "Boosting coverage-based fault localization via graph-based representation learning," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 664–676.'

[15] Y. Li, S. Wang, and T. Nguyen, "Fault localization with code coverage representation learning," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 661–673.

[16] X. Meng, X. Wang, H. Zhang, H. Sun, and X. Liu, "Improving fault localization and program repair with deep semantic features and transferred knowledge," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1169–1180.

[17] H. Liang, D. Hang, and X. Li, "Modeling function-level interactions for file-level bug localization," *Empirical Softw. Engg.*, vol. 27, no. 7, dec 2022.

[18] W. E. Wong, V. Debroy, R. Golden, X. Xu, and B. Thuraisingham, "Effective software fault localization using an rbf neural network," *IEEE Transactions on Reliability*, vol. 61, no. 1, pp. 149–169, 2011.

[19] A. Dutta, S. S. Srivastava, S. Godboley, and D. P. Mohapatra, "Combifl: Neural network and sbfl based fault localization using mutation analysis," *Journal of Computer Languages*, vol. 66, p. 101064, 2021.

[20] W. Zheng, D. Hu, and J. Wang, "Fault localization analysis based on deep neural network," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[21] T. Wang, H. Yu, K. Wang, and X. Su, "Fault localization based on wide & deep learning model by mining software behavior," *Future Generation Computer Systems*, vol. 127, pp. 309–319, 2022.

[22] W. E. Wong and Y. Qi, "Bp neural network-based effective fault localization," *International Journal of Software Engineering and Knowledge Engineering*, vol. 19, no. 04, pp. 573–597, 2009.

[23] A. Dutta, N. Pant, P. Mitra, and R. Mall, "Effective fault localization using an ensemble classifier," in *2019 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*, 2019, pp. 847–855.

[24] F. Bertolotti and W. Cazzola, "Fold2vec: Towards a statement-based representation of code for code comprehension," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 1, feb 2023.

[25] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, "code2vec: Learning distributed representations of code," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–29, 2019.

[26] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified pre-training for program understanding and generation," *arXiv preprint arXiv:2103.06333*, 2021.

[27] U. Alon, S. Brody, O. Levy, and E. Yahav, "code2seq: Generating sequences from structured representations of code," *arXiv preprint arXiv:1808.01400*, 2018.

[28] T. Lutellier, H. V. Pham, L. Pang, Y. Li, M. Wei, and L. Tan, "Coconut: Combining context-aware neural translation models using ensemble for program repair," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 101–114.

[29] J. Cambronero, H. Li, S. Kim, K. Sen, and S. Chandra, "When deep learning met code search," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 964–974.

[30] T. Hoang, H. J. Kang, D. Lo, and J. Lawall, "Cc2vec: Distributed representations of code changes," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 518–529.

[31] T. Ben-Nun, A. S. Jakobovits, and T. Hoefler, "Neural code comprehension: A learnable representation of code semantics," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] "Defects4j," https://github.com/rjust/defects4j.

[35] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[36] "The "Typical Software Project" Over Time," https://www.qsm.com/blog/2014/typical-software-project-over-time.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[38] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *arXiv preprint arXiv:2109.00859*, 2021.

[39] Y. Dai, F. Gieseke, S. Oehmcke, Y. Wu, and K. Barnard, "Attentional feature fusion," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2021, pp. 3559–3568.

[40] H. Xie, Y. Lei, M. Yan, Y. Yu, X. Xia, and X. Mao, "A universal data augmentation approach for fault localization," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 48–60.

[41] Y. Lei, H. Xie, T. Zhang, M. Yan, Z. Xu, and C. Sun, "Feature-fl: Feature-based fault localization," *IEEE Transactions on Reliability*, vol. 71, no. 1, pp. 264–283, 2022.

[42] "Gzoltar," https://github.com/GZoltar/gzoltar.

[43] "Pit," https://pitest.org/.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] P. S. Kochhar, X. Xia, D. Lo, and S. Li, "Practitioners' expectations on automated fault localization," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, 2016, pp. 165–176.

[46] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang *et al.*, "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.

[47] F. Keller, L. Grunske, S. Heiden, A. Filieri, A. van Hoorn, and D. Lo, "A critical evaluation of spectrum-based fault localization techniques on a large-scale software system," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2017, pp. 114–125.

[48] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan, "Scalable statistical bug isolation," *Acm Sigplan Notices*, vol. 40, no. 6, pp. 15–26, 2005.

[49] L. Lucia, D. Lo, L. Jiang, F. Thung, and A. Budi, "Extended comprehensive study of association measures for fault localization," *Journal of software: Evolution and Process*, vol. 26, no. 2, pp. 172–219, 2014.

[50] L. Naish, H. J. Lee, and K. Ramamohanarao, "A model for spectra-based software diagnosis," *ACM Transactions on software engineering and methodology (TOSEM)*, vol. 20, no. 3, pp. 1–32, 2011.

[51] W. E. Wong, Y. Qi, L. Zhao, and K.-Y. Cai, "Effective fault localization using code coverage," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1. IEEE, 2007, pp. 449–456.

[52] L. Zhang, M. Kim, and S. Khurshid, "Localizing failure-inducing program edits based on spectrum information," in *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2011, pp. 23–32.

[53] V. Musco, M. Monperrus, and P. Preux, "A large-scale study of call graph-based impact prediction using mutation testing," *Software Quality Journal*, vol. 25, pp. 921–950, 2017.

[54] L. Zhang, L. Zhang, and S. Khurshid, "Injecting mechanical faults to localize developer faults for evolving software," *ACM SIGPLAN Notices*, vol. 48, no. 10, pp. 765–784, 2013.

[55] L. Zhang, T. Xie, L. Zhang, N. Tillmann, J. De Halleux, and H. Mei, "Test generation via dynamic symbolic execution for mutation testing," in *2010 IEEE international conference on software maintenance*. IEEE, 2010, pp. 1–10.

[56] Z. Zhang, Y. Lei, X. Mao, and P. Li, "Cnn-fl: An effective approach for localizing faults using convolutional neural networks," in *2019 IEEE*

26th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2019, pp. 445–455.

[57] J. Jiang, Y. Wang, J. Chen, D. Lv, and M. Liu, "Variable-based fault localization via enhanced decision tree," *arXiv preprint arXiv:2211.11526*, 2022.

[58] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[59] M. Zeng, Y. Wu, Z. Ye, Y. Xiong, X. Zhang, and L. Zhang, "Fault localization via efficient probabilistic modeling of program semantics," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 958–969.

[60] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, 2016, pp. 87–98.

[61] A. LeClair, S. Haque, L. Wu, and C. McMillan, "Improved code summarization via a graph neural network," in *Proceedings of the 28th International Conference on Program Comprehension*, ser. ICPC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 184–195.

[62] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "A transformer-based approach for source code summarization," *arXiv preprint arXiv:2005.00653*, 2020.

[63] Y. Wan, Z. Zhao, M. Yang, G. Xu, H. Ying, J. Wu, and P. S. Yu, "Improving automatic source code summarization via deep reinforcement learning," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 397–407.

[64] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 783–794.

[65] Y. Li, S. Wang, T. N. Nguyen, and S. Van Nguyen, "Improving bug detection via context-based code representation learning and attention-based neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, oct 2019.

[66] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified pre-training for program understanding and generation," *arXiv preprint arXiv:2103.06333*, 2021.

[67] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, "Codexglue: A machine learning benchmark dataset for code understanding and generation," *arXiv preprint arXiv:2102.04664*, 2021.

[68] Z. Tang, X. Shen, C. Li, J. Ge, L. Huang, Z. Zhu, and B. Luo, "Ast-trans: code summarization with efficient tree-structured attention," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 150–162.

[69] X. Li, Y. Qu, and H. Yin, "PalmTree: Learning an assembly language model for instruction embedding," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. ACM, nov 2021.

[70] F. Zuo, X. Li, P. Young, L. Luo, Q. Zeng, and Z. Zhang, "Neural machine translation inspired binary code similarity comparison beyond function pairs," in *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019.

一种基于 Selenium 复杂多层 Web 系统自动化测试方法

仵林博[1]，尚小虎[1]，邓霏[1]，聂长海[2]，马思友[1]
（1．中国工程物理研究院 计算机应用研究所，绵阳市，621999；2.南京大学计算机科学与技术系，南京市，210023)

摘 要: 为降低 Web 应用的异构性和动态性所带来的测试复杂性，提高复杂多层 Web 业务组合测试覆盖充分性和测试效率，本文基于 Selenium 提出了一种关键字驱动的 Web 应用自动化测试方法，通过测试建模、基于模型的用例自动生成准则构建，解决了不同测试覆盖强度的测试用例设计的充分性要求。通过自动化测试执行框架、关键字驱动以及脚本执行引擎设计，实现了测试用例的自动执行、测试过程实时显示以及测试报告自动生成，提高了测试效率。最后，以电子会议系统为例，在研制的 Web 应用自动化测试平台中验证了该方法的可行性和应用价值。

关键词: Web 应用； 基于模型的测试； 关键字驱动； 测试用例自动生成； 自动化测试

An Antomated Testing Method for Complex Multi-layer Web System Based on Selenium

Wu Linbo[1], Shang Xiaohu[1], Deng Fei[1], Nie Changhai[2], Ma Siyou[1]
(1. Institute of Computer Application, China Academy of Engineering Physics, Mianyang 621999, China;
2. Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

Abstract: In order to reduce the testing complexity caused by the heterogeneity and dynamism of web applications, and improve the adequacy and efficiency of testing coverage for complex multi-layer web business combinations, this paper proposes a keyword driven web application automation testing method based on Selenium. By constructing test modeling and model based use case automatic generation criteria, the adequacy requirements for test case design with different test coverage intensities are addressed. Through the design of an automated test execution framework, keyword driven, and script execution engine, automatic execution of test cases, real-time display of test processes, and automatic generation of test reports have been achieved, improving testing efficiency. Finally, taking the electronic conference system as an example, the feasibility and application value of this method were verified in the developed web application automation testing platform.

Key words: Web application; Model based testing; Keyword driven; Automatic test case generation; Automated testing

0 引言

随着综合办公、科研生产管理信息系统等信息化、数字化建设的深入推进，政府及企业投资的政务、金融、医疗、征信、民生等领域以及自主可控 Web 应用系统逐步向大规模、多集成、一体化的复杂应用形式演进，Web 应用的层次越多，其结构越复杂、测试的难度越大、安全性和可靠性要求也越高。伴随信息技术的进步，分布式系统的发展，软件系统越来越重视交互协作，多个模块服务的交叉调用，通常一个简单的常用功能插件可能就会有上千个测试用例，并且随着新版本的发布，一次完整的回归测试需要很大的人力物力，这些都给 Web 应用的自动化测试研究工作提出了更高更迫切的要求。

目前，大多数的测试工具主要关注自动化的测试执行，较少考虑有效的内建的测试技术与方法，无法满足测试需求分析及策划、测试用例设计与生成、测试执行与测试结果确认以及测试覆盖率分析等测试过程中不同阶段的需要，没有提供有效的自动化测试方案达到测试集合更新和复用，且测试分析设计过程中未能充分考虑复杂多层 Web 业务组合，为了保证 Web 应用测试覆盖的科学性和充分性，本文基于 Selenium 提出了一种模型驱动的自动化测试框架，通过遍历被测系统模型及其派生模型来产生测试用例，测试模型和基于测试模型生成的测试用例是抽象的，并且独立于平台，可重用。测试执行时通过对测试执行平台的动态配置自动产生实例化的可执行的测试用例集合。实现了将测试模型、测试用例平台和自动化测试执行平台分离实现，通过适配模块连接自动化测试平台和执行终端，降低 Web 应用的异构性和动态性所带来的测试复杂性。

本文的主要贡献如下：

1）提出了一种基于模型-关键字驱动的 Web 应用自动化测试方法，通过对被测对象测试建模、关键字设计，实现了测试用例的自动化生成与执行；

2）构建了测试用例生成的状态全覆盖、主辅功能优先覆盖等准则，解决了不同测试覆盖强度的测试用例设计的充分性要求；

3）根据复杂多层 Web 应用软件技术特征，研制一套高适应性模型驱动的 Web 应用自动化测试平台，并应用到具体实际项目中，验证了该方法的可行性和应用价值。

1 相关研究工作

模型驱动的软件开发与测试是当前软件工程领域的一个研究热点，许多软件开发者、学者、公司都投入到了 Web 应用的模型测试与自动化测试。基于模型的系统工程 MBSE（Model-Based Systems Engineering）是在系统工程的整个生命过程中，通过应用系统模型来支持需求、设计、分析、验证和确认等活动[1]。系统建模语言 SysML(Systems Modeling Language) 是国际系统工程学会 INCOSE(International Council on Systems Engineering)和对象管理组织 OMG 为了更好推动基于模型的系统工程，在对 UML2.0 的子集进行重用和扩展的基础上，提出一种新的系统建模语言[2]。SysML 定义了 3 类共 9 种模型图，每种模型图都针对某种特定的目的，说明系统的一个方面的特定信息。通过采用 SysML 可视化建模技术，能够从不同角度、不同粒度进行软件系统的描述，精化系统不同层面、不同层次的内容[3]，加强软件利益相关团队间的沟通，通过模型模

基金项目：国家重点研发计划（2021YFB3101805）；国家自然科学基金项目（61903348）；四川省科技计划重点研发项目（2021YFG0174）

作者简介：仵林博(1983－)，男，硕士，高级工程师，主要研究方向软件建模与测试。

Email: wulbcaep@163.com

块在不同的系统模型中重用，提高开发效率，增强软件质量。模型测试技术是一种依据模型规定的预期行为来检查被测对象运行结果正确性的软件测试方法，自动化测试技术是依据自动化策略来实现被测对象测试用例的自动执行，能够有效提升测试效率。基于模型驱动的自动化测试技术在不同的软件开发模型、不同业务领域中均可应用。2014 年，RobertV.Binder 的调查表明[4]：一般情况下，模型测试技术能够降低的 bug 遗漏、测试成本、测试周期分别达到 29%、35%、30%。相关学者针对测试用例的生成方法进行了大量研究，如基于模型、基于场景、面向目标、面向路径等方法[5]。Samuel[6]针对 UML 时序图中的很多关键特征不能被覆盖的问题，通过考虑循环、分支和终止等 UML 时序图特征，提出了一种基于遍历算法的自测试序列自动生成方法。Ranjit Swain[7]等人通过边界测试获取最少的测试用例数量，并实现了迁移路径的全覆盖。李吟等针对 Web 应用自动化生成的测试用例无法覆盖全部测试需求，提出了基于接口契约的测试方法，有效满足了测试需求[8,9]。侯俊[10]等人认为 Web 测试主要依靠人工完成，耗时且具有盲目性和倾向性，并提出了一种基于 WSDL 文档和形式化模型树 Web 服务操作测试用例的自动生成方法，能够简化测试用例的生成过程。Choi 等人[11]将分组遗传算法用于迁移路径的生成，通过对不同系统的实验取得了较高的迁移覆盖率。张毅伟等人[12]针对状态图中迁移之间的依赖关系会使得迁移路径不能按照输入序列来执行，从而导致覆盖率的下降，影响测试效果这一问题，提出了一种改进的分组遗传算法来实现可执行迁移路径的自动生成，通过合理设计编码方式和适应度函数来解决问题。

2 总体技术方案设计

自动化测试的主要目标是扩大覆盖，降低测试成本。但是自动化测试从自动化测试方案到自动化测试实施的过程中，也有固有限制、额外开发以及持续维护成本，要选择合适的测试策略，尤其要有好的架构和合适的自动化测试框架，ISTQB（国际软件测试资质认证委员会）体系的通用测试自动化架构 GTAA 针对 Web 服务应用场景，通过控制浏览器访问服务、控制路由与网络、模拟网路行为、控制内容业务逻辑注入以及控制数据库注入等，可实现各个环节的自动化测试，通用测试自动化架构 GTAA 设计如下图 1 所示，分为 4 层，包括测试生成层，测试定义层，测试执行层以及测试适配层。

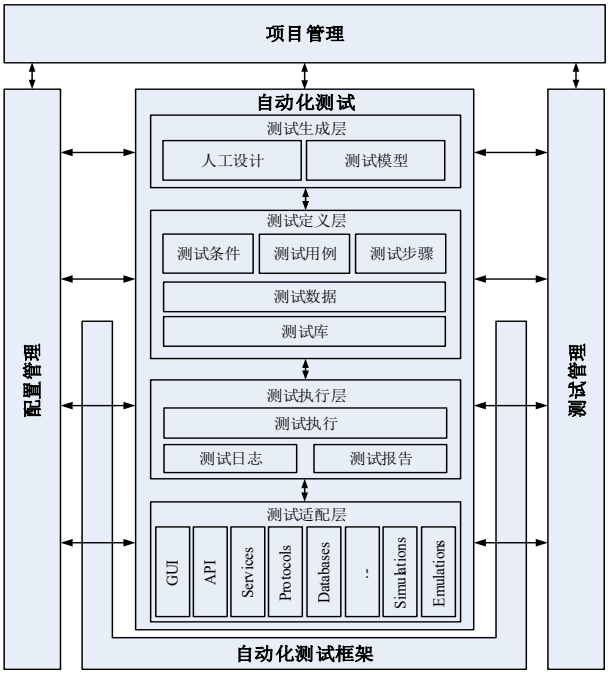


图 1 通用测试自动化架构 GTAA 设计

测试生成层提供手工或者自动设计测试用例的方法。测试定义层定义测试集合，测试用例，与被测试对象及驱动被测试对象的具体技术方法隔离，可以定义高层抽象测试，底层具体业务测试，定义测试数据以及支持测试的库。测试执行层提供用例执行的方法，Log 分析和测试报告的基本能力，并提供错误检测和恢复机制。测试适配层提供与被测试对象交互的办法/途径，以及操作被测试对象的接口。GTAA 每个模块仅负责一个职责，模块都是特殊目的，避免多功能/目的的模块，完全封装该职责消除依赖。同时，对扩展开放，每个模块可替换，对既有功能修改不影响依赖于系统其他部分。GTAA 具体实现的层次可选（如测试生成层、测试定义层），基于 GTAA 测试架构，为实现复杂多层 Web 业务自动化测试，设计了一种面向 Web 应用的自动化测试架构，架构决策设计与 GTAA 的对应关系如下图 2 所示。

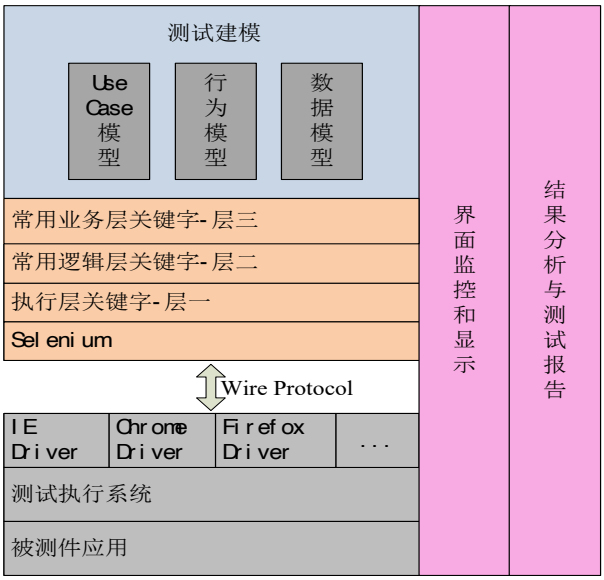


图 2 架构决策与 GTAA 的对应关系

在测试生成层，对系统功能进行抽象的测试模型包括 UseCase 模型、行为模型和数据模型，UseCase 模型又称为需求模型，用于对用户的需求，描述被测对象的测试项、功能点等，建立该模型主要为了清晰直观地捕捉用户需求，使得测试结果覆盖的内容能够直观的与需求联系在一起。行为模型主要用于对被测对象期待的行为进行描述，数据模型也用于描述被测对象的期待行为，这些行为包括输入和输出，通过描述相应的数据和状态来表达被测试对象的行为。利用该模型能够直接表达出期望的被测对象的动作和状态，绑定测试脚本，使得动作、状态与行为能够对应，可以自动生成可被自动化测试执行系统执行的测试用例，替代了人工的测试用例设计和编写，改善了工作效率。

在测试定义层和测试执行层，为了实现测试生成层的测试逻辑，需要设计人员编写脚本实现业务逻辑封装，即将人对于软件行为的操作、对界面元素的操作、对硬件的操作，通过编写测试脚本的形式设计为不同层次的关键字，并将关键字脚本绑定在状态图中各个节点，这样生成层生成的测试用例/测试逻辑的执行，就转化为关键字脚本序列的运行，这些动作序列通过测试脚本调用 windows API 操作，实现测试用例的自动执行。底层关键字驱动执行基于 Selenium 库，Selenium 是个强大的工具集，支持快速开发 Web 应用的测试自动化，并提供了丰富的测试函数集，这些函数操作非常灵活，允许多种选择定位 UI 元素和比较测试的期望结果与应用程序的实际行为。

在测试适配层，适配特定的控制，监控与交互技术。实现不同的与被测试对象的控制，监控与交互，将测试动作合理地分布在多个测试环境/设备上，即将基于测试用例或业务的操作/动作序列（测试逻辑）和执行每个动作的方法（执

行逻辑）进行隔离，对逻辑隔离模块独立封装，通过跨进程通信 RPC 机制，实现测试逻辑与执行逻辑可以分布在不同的宿主机上。对 Web 业务来说，需要适配不同的浏览器，除了被测系统外，还需要适配相应的陪测环境、测试工具、自动化仿真测试环境等，保证被测系统能够流畅运行的环境是实现整个测试用例的自动执行的前提。

3 Web 应用测试建模

3.1 UseCase建模

参照 SysML 建模中的 UseCase 模型，UseCase 模型主要用于对用户需求，如界面迁移，测试策略，用例说明，原型系统等进行表述，建立该模型主要为了：1）清晰直观地捕捉用户需求；2）迅速暴露测试计划制定时可能的缺漏点；3）建立从需求到结果的双向可追溯关系，使测试活动呈现的更加明确；4）建立追溯信息，使得测试结果覆盖的内容能够直观的与需求联系在一起；5）抓住用户的测试重点，形成大致的测试策略，确认 UseCase 模型的抽象层次。UseCase 模型的抽象层次可按功能抽象，即按照功能所需要的最上层的测试条件进行抽象，不细化到内部，按照功能下测试条件所分属的数据类和动作类的标准将二者分开，画出不同功能下数据类和动作类的 UseCase 图，对动作类测试条件和数据类测试条件的方法进行区分。

1）数据类：在测试条件中主要为测试范围，具体表现为该测试条件在测试时需要遍历需求内大部分甚至整个测试需求，或者它的改变会影响到整个测试内容的结果但几乎每一条需求对它都有要求，或者该测试条件无法通过被测对象的界面进行更改的都可以作为数据类。

2）动作类：在需求的测试条件中主要为测试内容，具体表现为每一轮测试可能因为数据改变而多次测试的相同的动作部分。需要对测试对象做出某种改变来检查测试结果的，通常归类于动作类。

3.2 状态图建模

状态图模型是行为模型的一种，主要用于对被测对象期待的行为进行描述，通过描述相应的状态来表达被测试对象的行为，建立该模型主要为了：1）直接表达出期望的被测对象的动作及状态；2）绑定测试脚本，使得动作、状态与行为能够对应；3）可以自动生成可被自动化测试执行系统执行的测试用例，替代了人工的测试用例设计和编写，改善了工作效率。状态图模型的抽象层次有以下几种抽象方式：

1) 按照测试执行系统的最小步骤来进行抽象。测试执行系统即为该自动化测试项目的测试对象。这种抽象方法要求将测试执行系统内每一步动作都封装成迁移，做成的结果都在状态中进行检查。优点是在做测试执行系统时工作量小，测试简单，缺点是生成的 case 量过大。

2) 按照用例进行抽象。这种抽象方法要求将用例中的动作或数据封装在几个迁移里，只有用例结果需要检查。这种抽象方法的优点是在建立状态图模型时工作量小，生成的 case 数量少，缺点是做测试执行系统内的工作量大，测试复杂容易出错。

3) 按照需求中的测试条件和测试重点，对 UseCase 模型的用例进行抽象。这种抽象方法要求将从一个条件到另一个条件的迁移过程封装成迁移，需要检查的测试条件及测试重点封装成状态，只检查状态中的值。这种抽象方法的优点是测试条件明确，建立状态图模型时工作量一般，做测试执行系统时的工作量一般，缺点是因为迁移和状态中可能都封装有判定条件，误判程度可能高于第一种方法但是要小于第二种方法。

3.3 数据建模

数据模型用以描述被测对象的期待行为，这些行为包括输入和输出，通过描述相应的数据来表达被测对象的行为。数据模型的抽象相对简单，其遵循的规则为简单类型可以通过结构化（明确的代码或者业务上的关联性），抽象为复合类型。例如姓名，年龄，薪资这些数据参数的取值为简单类型，其可以结构化抽象为复合类型“员工”，复合类型的数据可以进一步结构化进行抽象，比如员工，技术经验，管理经验这些复合类型的数据可以进一步抽象为“技术专家”复合类数据。数据模型的建立过程可分为以下两个步骤：

1）根据对状态图模型分析得到的数据，将同一功能中属于相同的数据进行列出并按照需要的位置不同进行分类。数据在该功能中可能做到某一步骤时才会用到，而在这个步骤中可能因为数据不同产生的结果也不一样。因此，我们需要对每一项用到的数据进行分类。

2）按照划分的数据分类，建立数据模型。在建立模型时，数据作为状态，迁移只作为指向作用，按照同一步骤中用到的不同数据放在同一横向的要求进行建立，如果是不同步骤则放在纵向上表达。

4 关键字设计

4.1 关键字定义

要实现自动化执行，本文采用关键字驱动和数据驱动的模式，关键字设计方式可以通过关键字录制或编写关键字来实现，关键字录制支持用户在主流编程语言和技术构建的 Web 页面上，不需要任何编程逻辑来编写测试脚本，只需要记录与浏览器的交互，工具自动记录相关操作并录制相应的测试脚本。另一种方式就是测试设计人员自主编写关键字，实现相应的业务逻辑与操作。数据驱动的模式主要通过引入关键字“变量”概念来支持“数据驱动”模式的自动化测试，通过从数据模型生成的测试脚本修改数据文件，然后由脚本读取数据文件，根据数据文件的内容替换脚本内指定的“变量”来实现，当批量执行脚本时，可以在执行集合中首先执行一个由数据模型生成的测试用例，该用例将写入数据文件（.ini 文件），该文件的写入为将一些变量的值改变为数据模型里脚本指定的值，当执行由行为模型生成的关键字序列对应的自动化测试脚本时，将自动读取数据文件中变量的值并将其赋值给脚本中的“变量”。由此将多个数据模型生成的用例与关键字脚本用例组合后就形成了关键字驱动+数据驱动的自动化测试脚本框架。

4.2 关键字使用

建立完状态图模型之后，为了使从模型生成的测试用例能够自动执行，需将状态图模型与实际执行系统相连接，即需要在模型中绑定相应关键字脚本，绑定关键字支持在模型中直接输入所关联的关键字脚本，也可以选择插入关键字脚本方式从关键字库（采用录制回放创建关键字）中选择对应关键字插入执行脚本中。一般情况下，在状态图模型中状态框处绑定状态检测和结果检查、确认类关键字脚本，在迁移线处绑定触发或输入类关键字脚本，状态或者迁移对应的测试脚本可以输入多条脚本命令，这些脚本命令代表了状体或迁移实际代表的动作，一般要求状态对应脚本至少包含一条与状态含义匹配的检查脚本命令，迁移至少包含一条与迁移含义匹配的动作脚本命令。

5 用例自动生成

用例自动生成主要是将 UseCase 模型与状态图模型关联，通过配置测试覆盖生成策略，自动化生成满足覆盖目标的测试用例集。基于状态图模型的测试用例自动生成的核心是测试覆盖准则的制定，本文基于状态图模型的测试用例设计准则为以下三个准则：

测试覆盖准则 1：状态全覆盖准则。状态全覆盖准则本质上是对状态图模型所定义的软件行为围绕需求对应的功能进行最全面的覆盖。在此覆盖下，将枚举所有符合参数定义规约的路径，所以也将产生最为庞大的测试用例集合。当需要对应高风险的需求或测试执行的早期轮次中应选择此

覆盖来确保足够的覆盖。

测试覆盖准则 2：主辅功能优先覆盖准则。主辅功能优先覆盖准则本质上是对状态图模型所定义的软件行为围绕需求对应的功能进行完整的覆盖。在此覆盖下，将枚举与需求有明确对应关系的路径和已知可能与需求发生交互影响的路径，最后再添加上影响不大或影响未知的行为路径。当功能开发完毕，需要进行完整的功能测试等较完整的测试时可选择此覆盖以取得测试执行代价与测试覆盖（质量可信度水平）之间的平衡。

测试覆盖准则 3：测试数据覆盖准则。测试数据覆盖准则针对数据模型图中的数据生成覆盖要求，数据图覆盖本质上是对行为模型所定义的软件行为围绕需求对应的功能进行基础的覆盖。在此覆盖下，将仅枚举与需求有明确对应关系的路径和已知可能与需求发生交互影响的路径。此生成算法是最小生成算法。对功能直接对应的和有相关影响的行为进行基础的覆盖，适合功能开发完毕或开发中生成少量的冒烟测试用例来快速对功能进行验证。另外，对于数据模型，推荐采用此算法来避免产生大量不必要的组合。

在基于状态图模型和数据模型图覆盖准则生成测试用例集合的同时，为便于控制测试用例数量，我们需要配置 Case Size（最大用例数），Loop Size（状态图环大小），Loop Times（环次数）以及 Max Steps（用例的最大步长）等参数，来限制测试用例的生成数量，各个参数的定义如下：

Case Size：用例最大数量设置用例集合的用例最大数量，默认为 1000。假设能够生成的用例数量为 20 条，此时将 Case Size 设置成 5，则只会生成 5 条 case，并且生成的顺序和前次生成的相同。

Loop Size：指定要生成的包含特定大小环的测试用例。环大小定义为路径内距离，路径内距离是指最近的两个重复节点之间的节点数（含重复的节点本身）。如路径 a，b，c，d，a；则长度为 5。如路径 a，b，c，b，a；则长度为 3。因为在 a 到 a 的环内存在 b 到 b 的环。由此定义，若环的长度设为 1，则意味着没有环。

Loop Times：环次数。实际上决定了生成的测试用例最多能够走几次环路，即执行相同的操作序列多少次，比如执行一系列操作 20 次。

Max Steps：用例的最大步长。设置用例集合内一条用例可能的最大步长，默认为 100。最大步长是指一条 case 从起点到终点所经过的 transition 的数量。比如 100 条 case 中，步长小于等于 5 的 case 有 30 条，此时把 Max Steps 的值设为 5，在其他选项都为默认的情况下，则此时会生成 30 条步长小于等于 5 的 case。

6 用例集合建立、执行与结果分析

6.1 用例集合建立

基于建立 UseCase 模型、状态图模型、数据模型以及测试覆盖生成策略/准则，通过模型图的遍历算法已经能生成测试用例集合，为了将测试用例与最初的用户需求对应，方便工作人员管理以及方便数据与行为模型的结合使用，需要对测试用例进行管理与执行，即创建测试执行集合（Execution Sets）。Execution Sets 为生成的测试用例集合的一个实例，支持对执行系统执行测试用例后的结果进行记录，具体功能包括：

1）建立 Execution Sets 集合，包括集合名称、测试项目、功能、生成日期等，以便后期查阅。如有需要，可以在名字中添加其他内容，如版本号、需求等，便于后续回归测试时使用。

2）按照用户需求，将测试用例逐条添加到 Execution Sets。涉及到数据与行为测试用例结合的，先加入需要的数据测试用例，再加入建立在该条数据用例上使用的行为用例。

3）按照实际测试要求，如果执行系统终端没有硬件变化或者其他需要工作人员手动调整的变化，可以将所有的 Excecution Sets 集合合并生成一个大的 Excecution Sets 集合，作为实际的测试用集合使用。

6.2 用例集合执行

在完成 Excecution Sets 集合的建立之后，针对建立的 Excecution Sets 集合能实现自动执行，而用例集合的自动执行是通过脚本引擎（script engine service，简称 SES）来实现，脚本引擎 SES 是自动化平台和执行系统的中间件，负责自动化平台与执行系统的通信，主要功能包括：

1) 脚本的传输，自动化平台将服务端的测试脚本通过脚本引擎传输到执行系统；

2) 脚本的执行，脚本引擎将接受到的测试脚本，通过调用执行系统顺序解析并执行；

3) 结果的返回，脚本引擎将执行系统的测试结果返回到自动化平台。

不同的脚本引擎中包含的执行器会不一样，所以可选择与测试项目对应的脚本引擎，点击运行测试执行集合后，执行系统顺序执行完成所有的测试用例并返回执行结果。脚本引擎 SES 的设计采用跨进程通信 RPC 机制，实现将测试逻辑与执行逻辑隔离，脚本执行系统与自动化平台控制端可以分布在不同的宿主机上。

6.3 测试结果分析

在完成 Execution Sets 集合自动化测试执行后，自动化平台要对结果进行反馈，测试反馈结果包括整个测试结合的执行结果/Run Result 和测试用例覆盖率统计/Strategy Coverage。Run Result 是指测试用例执行结束后，执行结果以 linebar 或者 dashboard 的形式展示。Strategy Coverage 是指不同 UseCase 的测试用例以不同颜色区别开来，并显示占总数的百分比。

依据执行结果，针对执行 Fail 测试用例，由于可能会出现误判或者由于执行系统本身的原因导致出错，因此需要对结果进行分析来找到真正的问题点，需针对具体的问题进一步分析与确认，自动化平台提供 Fail 问题用例执行过程的整个 log 信息，便于用户追踪具体的问题出错点。

7 应用验证

基于本文提出的自动化测试方法，设计开发了相应的软件自动化测试工具平台，并选择电子会议系统进行应用验证。电子会议系统是一个典型复杂 Web 应用信息系统，其代码规模超过 20 万行，测试过程中存在功能业务的各种组合，系统 iframe 框架嵌套超过了 3 层。应用本文提出的自动化测试方法，对电子会议系统的功能业务行为进行描述，建立如下图 3 所示状态图模型。

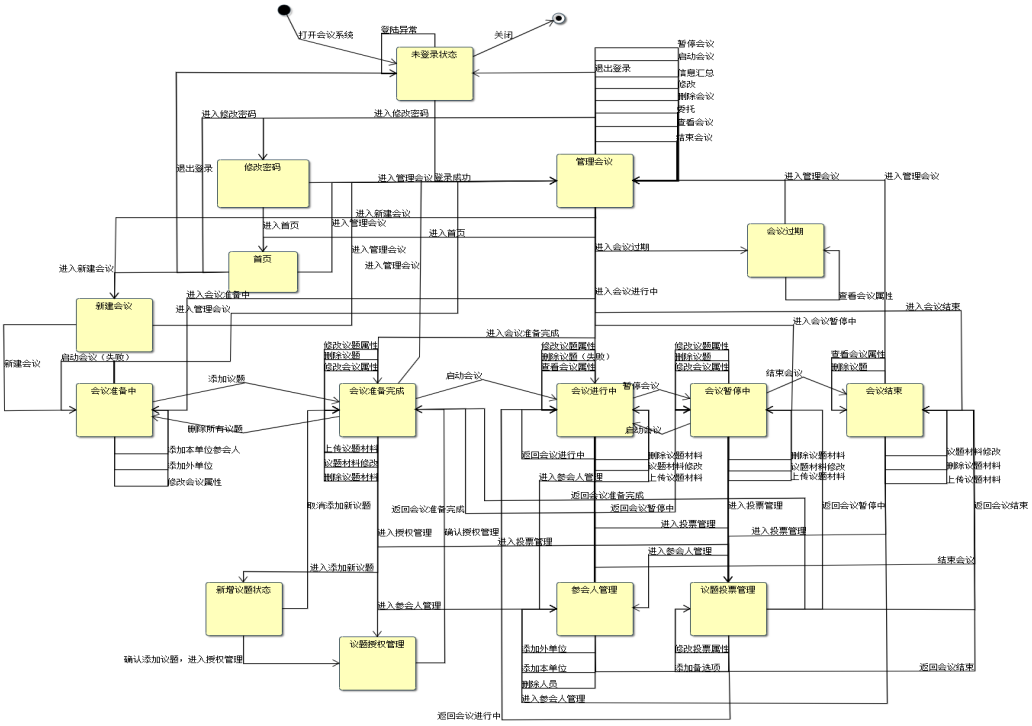


图 3 电子会议系统状态图建模

对状态图模型的每个状态和迁移动作分别设计关键字，其中状态迁移处绑定输入操作类关键字，状态框处绑定确认类/Check 类关键字，针对电子会议系统，共设计实现关键字 93 个，部分关键字脚本如下图 4 所示。

```
class sems(baseOperation):
    # 1 打开网站（传入网站地址）
    def open(self, url):
        try:
            url = self.findValueInFile(url)
            self.driver.get(url)
            time.sleep(1)
            self.driver.maximize_window()
            if not self.waitelementExist(By.CSS_SELECTOR, ".admin"):
                logger.info("open fail! can't find element!")
                return False
            logger.info("open success!")
            return True
        except Exception as e:
            logger.error("open Fail! \nError Message:\n" + str(e))
            return False
    # 2 登录成功（传入账号，传入密码）
    def loginSuccess(self, user, psw):
        try:
            user = self.findValueInFile(user)
            psw = self.findValueInFile(psw)
            self.driver.switch_to.default_content()
            self.driver.refresh()
            time.sleep(1)
            if not self.waitelementExist(By.CSS_SELECTOR, ".admin"):
                logger.info("loginSuccess fail! can't find element!")
                return False
            self.driver.find_element(By.CSS_SELECTOR, ".admin").click()
            time.sleep(1)
            self.driver.find_element(By.ID, "name").click()
            time.sleep(1)
            self.driver.find_element(By.ID, "name").send_keys(user)
            time.sleep(1)
            self.driver.find_element(By.ID, "password").send_keys(psw)
            time.sleep(1)
            self.driver.find_element(By.XPATH, "//div[@id=\'form\']/table/tbody/tr/td[2]/div").click()
            time.sleep(1)
            logger.info("loginSuccess success!")
            return True
        except Exception as e:
            logger.error("loginSuccess Fail! \nError Message:\n" + str(e))
            return False
```

图 4 电子会议系统关键字设计

配置测试覆盖策略/准则，其中 Case Size 为 1000，Loop Size 为 0，Loop Times 为 1，Max Steps 为 100，状态图覆盖策略为状态全覆盖准则，配置完成后点击 Case Generate 一键自动生成，平台依据覆盖策略自动化生成满足要求的 1000 条测试用例，这些用例覆盖了电子会议系统各个主要功能业务之间的组合覆盖情况，包括数据组合，这些组合覆盖的复杂性超出了人工设计的极限，满足系统测试充分性设计要求。通过关键字驱动的测试执行，实现了测试用例中每个步骤的精确重复，整个测试执行过程，也实现了可视化显示，每个测试用例的执行过程都保存有完整的 log 执行日志，平台执行界面也显示了每个测试用例的执行结果（成功/失败/未执行），以及当前正在执行的步骤，见下图 6 所示。

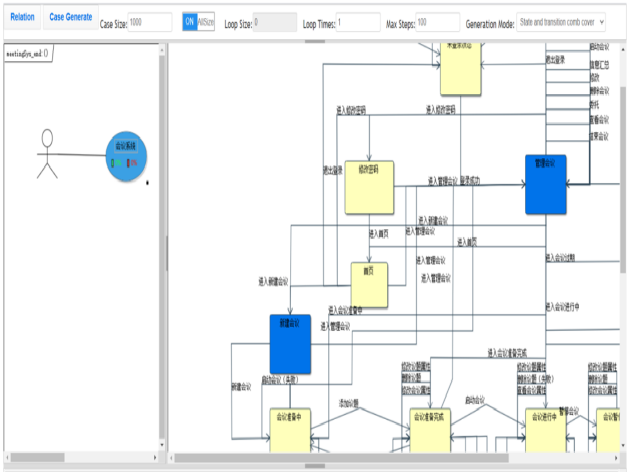


图 5 覆盖策略配置及测试用例自动生成

创建测试执行集合 Execution Sets，并运行测试执行集合后，执行系统顺序执行完成所有的测试用例并返回执行结果，如下图 6 所示，左栏为执行结果显示。中间为当前正在执行测试用例的动态实时路径覆盖情况，绿色显示已经覆盖过路径，蓝色显示正在执行路径。右栏为整个用例集合的执行情况（1000 条用例顺序执行情况），包括已/未执行用例，正在执行用例以及执行结果。

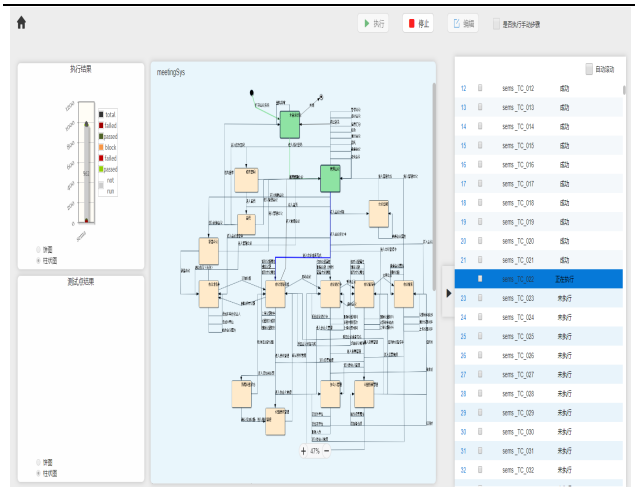


图 6 用例集合建立及测试用例自动执行

8 结束语

本文提出的基于Selenium的Web系统自动化测试方法，是模型驱动的自动化测试方法的扩展与应用。通过采用关键字驱动和数据驱动的测试设计方法，实现了测试用例的7X24 小时自动执行，达到了测试资源的充分利用。由于采用全流程自动化执行，比人工执行更快，较少的人工错误，缩短项目执行时间，改善了测试效率。自动化测试可执行人工无法执行的用例，包括实现复杂的业务组合，精确的步骤重复，执行更多测试用例，扩大了测试覆盖。同时，本文提出的自动化测试方法也对其他类型系统（比如嵌入式系统）自动化测试具有重要借鉴意义。下一步的工作方向是异构模型之间嵌套测试设计方法，包括异构模型之间嵌套覆盖策略/准则设计，测试用例生成算法等是其中的重要研究内容。

参考文献:

[1] Friedenthal S, Moore A, Steiner R. A Practical Guide to SysML[M]. Germany:Morgan Kaufmann. 2014. (ISBN: 978-0-12-385206-9).

[2] Gauthier J M. Test Generation for RTES from SysML Models:Context, Motivations and Research Proposal[C]. IEEE 6th International Conference on Software Testing, Verification and Validation, 2013, pp. 503–504.

[3] Lenny Delligatti, Baiwei Hou, Yanlan Zhu. SysML Distilled:A Brief Guide to the Systems Modeling Language[M]. Beijing: Machinery Industry Press, 2014.

[4] Binder RV, Kramer A, Legeard B. 2014 Model-based testing user survey: Results. Robert V. Binder's Technology and Businiss Blog. 2014

[5] Kaur P, Gupta G. Automated model-based test path generation from UML diagrams via graph coverage techniques. IJCSMC, 2013, 2, 7: 302-311.

[6] Samuel P, Joseph AT. Test sequence generation from UML sequence diagrams. Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. 2008. 879-887.

[7] Swain R, Panthi V, Behera PK, Mahapatra DP. Automatic test case generation based on state machine diagram. International Journal of Computer Information Systems, 2012, 4(2): 99-124.

[8] 李吟. 基于接口契约的有状态 Web 服务用例集生成[J]. 计算机研究与发展, 2017, 54(3):609-622.

[9] 李吟, 方建勇, 江梦. 面向需求覆盖的 Web 服务自动化测试框架[J]. 计算机科学与探索, 2017, 11(11):1747-1763.

[10] 侯俊, 周红, 马春燕, 等. 面向 WEB 服务的测试用例自动化生成方法[J]. 西北工业大学学报, 2018, 36(1):149-155.

[11] Choi Y M, Lim D J, Choi Y M, et al. Automatic feasible transition path generation from UML state chart diagrams using grouping genetic algorithms[J]. Information & Software Technology, 2018, 94: 38-58.

[12] 张毅伟, 贲可荣. 基于状态图测试的迁移路径生成方法[J]. 计算机科学与探索, 2019, 13(6): 961-972.

Vivado 综合器安全状态机精确行为分析

于志杰*[①②] 田 甜[①②] 朱 倩[①②] 童宗挺[①]
[①](北京轩宇信息技术有限公司 北京 100190)
[②](北京控制工程研究所 北京 100190)

摘 要：针对 Vivado 综合器对状态机安全实现后，难于直观掌握门级电路精确行为的问题，在逻辑等价性检查形式化方法的基础上，提出一种对综合后状态机全状态空间精确行为进行分析的流程，建立与门级电路满足全状态空间严格等价的寄存器传输级（RTL）代码，更直接易懂地描述了安全状态机的综合结果。选择 IWSL2005 样例集中的代码，对 Vivado 的 4 种安全模式综合结果进行分析，验证了该方法的正确性和有效性。并给出了不同安全实现模式对应状态编码，状态寄存器纠检错、状态转移及优化，状态输出函数的实现结果，提供了 Vivado 综合器状态机不同安全模式注意事项和应用参考。

关键词：现场可编程门阵列；有限状态机；安全实现；Vivado

Analysis of Accurate Behavior of Vivado Synthesis for Safe State Machines

YU Zhijie[①②] TIAN Tian[①②] ZHU Qian[①②] TONG Zongting[①]
[①](*Beijing Sunwise Information Technology Ltd., Beijing* 100190, *China*)
[②](*Beijing Institute of Control Engineering, Beijing* 100190, *China*)

Abstract: To solve the problem that it is difficult to intuitively understand the accurate behavior of gate-level circuits after the state machine is safely implemented by Vivado Synthesis, based on the formal method of logical equivalence checking, a process for analyzing the accurate behavior of the synthesized state machine in the whole state space is proposed, the Register Transfer Level(RTL) code is provided which is strict logical equivalence with gate-level circuit in full state space, and the synthesized results of safety state machines are described more directly and understandably. The code from the IWSL2005 benchmark is selected to analyze the results for four kinds of safe modes implemented by Vivado, the correctness of this process is proved. The results of state encoding, the error detection and correction of state registers, the state transition and optimization, and the state output functions for the different safe modes are presented, the cautions and application references are proposed for the different safe modes of the finite state machine by Vivado.

Key words: Field Programmable Gate Array(FPGA) ;Finite State Machine(FSM); Safe implement; Vivado

1 引 言

在数字逻辑电路设计中，有限状态机（Finite State Machine，FSM）是一种广泛应用的设计方法，可以高效地进行同步时序逻辑处理，是实现复杂处理流程的典型设计方式，在现场可编程门阵列（Field Programmable Gate Array，FPGA）等可编程逻辑设计中经常采用[1]。

Vivado 是 Xilinx 公司针对 7 系列 FPGA 推出的新一代集成开发环境，Vivado 中的综合器较传统 ISE 环境下的 XST 综合器进行了较大升级。随着电子设备功能复杂度提升、FPGA 器件换代以及国产化兼容器件的推广应用，在航天、航空、军工等领域，FPGA 器件也逐步从 Virtex4/5 系列更新到 7 系列。特别是在商用航天等领域，出于低成本、小型化、低功耗等考虑，属于货架产品（Commercial-off-the-shelf，

*通信作者：于志杰 yuzhijie@sunwiseinfo.com

COTS）的 Zynq-7 系列器件也开始采用。伴随着 FPGA 器件升级，Vivado 综合器的应用也愈发广泛。

传统认为只要航天器的轨道高度在白天超过 60 公里或夜间超过 80 公里[2]，均需要考虑空间辐射效应的影响，对于航空机载电子设备，在大气中也需要考虑因大气中子引起的单粒子效应[3]，文献[4]的试验数据表明，即使是采用专门抗辐射加固工艺设计的国产 FPGA，也属于单粒子敏感元器件。因此在使用状态机时，需要避免因辐射环境等外部因素导致状态机出现错误或死锁的情况[5]。

传统的防护设计方法是采用三模冗余的方式，但会带来较大的资源开销和时序性能下降[6]，Frigerio 等人[7]提出一种基于内部 RAM 实现的状态机设计方法，并通过纠检错编码对 RAM 内容进行保护。李列文等人[8]采用双冗余 RAM 块的方式，对状态机错误进行检测。严健生等人[9]提出增加编码冗余的方式对状态机进行加固。针对 one-hot 编码状态机，Cassel 等人[10]提出了一种自检测和三模冗余相结合的设计方法，王鹏等人[11]提出一种将状态位复制 M 次的编码加固方法。Sooraj 等人[12]提出了采用海明码对状态机编码进行错误检测，Lupi 等人[13]设计了具备三表决器的三模冗余状态机。一种回滚恢复电路[14]也应用于有限状态机的设计中。若时间延迟可以接受，还可以采用信息冗余和时间冗余相结合的方式[15]，对状态机电路进行防护。但上述研究工作主要还是集中在 RTL 代码设计中进行状态机的防护。FPGA 综合器在对 RTL 代码进行综合时，会进行一系列的转换和优化[16]，并引入状态机安全实现等功能[17]。

Liszewski 等人[18]提出了采用约束求解器对综合后门级网表进行逆向分析的流程，但该方法较为繁琐，不能直观反应综合后的状态机行为，文献[19]采用状态转移图（State Transition Graph，STG）对 Vivado 缺省设置下状态机的综合行为进行了分析，未考虑综合器引入安全实现后的结果，并且对于状态数量多、转移条件复杂的状态机行为，STG 难于进行精确表达。朱倩等人[20]提出一种针对有限状态机进行逻辑等价性分析的方法，但该方法未能覆盖状态机进行安全实现后的情况。

本文提出一种通过形式化的逻辑等价性方法，采用 RTL 行为级代码形式，对 Vivado 综合实现的安全状态机进行精确行为分析，得到全状态空间下有限状态机行为的直观描述。

本文第一节给出状态机安全实现的形式化描述。第二节介绍 Vivado 综合器中 4 种状态机安全实现模式。第三节提出一种基于逻辑等价性形式化方法的门级状态机精确行为分析流程。第四节选择 IWLS 2005 Benchmarks 中的样本代码，对 Vivado 综合器 4 种状态机安全模式进行实验和分析。最后给出高安全性要求场景下 Vivado 安全状态机的应用要点。

2 状态机安全实现

有限状态机可以由如下的形式定义给出。

定义：有限状态机可以表示为五元组 $M=\{Q,I,O,\delta,\lambda\}$，其中：

(1) $Q=\{q_0,q_1...q_{n-1}\}$是有限状态集合，在任一确定时刻，有限状态机只能处于其中一个确定的状态 $q_i\in Q$。其中 $q_0\in Q$是初始状态，有限状态机由此开始工作。

(2) $I=\{i_0,i_1...i_{m-1}\}$是有限状态机的输入集合，在任一确定时刻，有限状态机只接收一个确定的输入 $i_i\in I$。

(3) $\delta:Q\times I\to Q$是状态转移函数，如果在某一确定时刻，有限状态机处于某一状态 $q_i\in Q$，,并接收输入 $i_i\in I$，下一个时钟周期有限状态机将进入新的状态 $q'=\delta(q_i,i_i)\in Q$。

(4) $O=\{o_0,o_1...o_{r-1}\}$是有限状态机的输出集合，在任一确定时刻，有限状态机有一个确定的输出 $o_k\in O$。

(5) λ是输出函数，如果 λ是状态 q_i和输入 i_j的函数 $\lambda:Q\times I\to O$，状态机被称为 Mealy 机，输出 $o=\lambda(q_i,i_i)\in O$；如果 λ只是状态 q_i的函数 $\lambda:Q\to O$，状态机被称为 Moore 机，输出 $o=\lambda(q_i)\in O$。

通过 VHDL 或 Verilog 等硬件描述语言完成的状态机设计，称之为状态机的行为描述。经过综合布局布线，状态机被转化为门级电路形式，称之为状态机的门级实现。门级实现一般由组合逻辑和寄存器时序逻辑构成，其中的寄存器用于存储状态，组合逻辑用于状态译码、输入输出处理等。

在状态机的门级实现中，根据选择的状态编码方式，将会使用一定数量的寄存器保存状态，如果使用了 k 个寄存器，Q 中状态的数量为 n，显然需要满足 $2^k\geq n$。门级 k 个寄存器对应的全空间共有 2^k个状态，通常情况会大于 Q 集合中所有合法状态的总数量 n。设 k 个寄存器对应的全空间为 U，$\bar{Q}$为 Q 在 U 上的补集，即 $\bar{Q}=\{x\mid x\in U且x\notin Q\}$。

在航天，航空等高安全性应用领域，必须要考虑因异常情况门级状态出现 $q \in \bar{Q}$ 的情况。例如针对卫星星上 FPGA 产品，正常情况下状态机都在 Q 集合范围内进行转移，当空间辐射环境出现单粒子事件，导致门级状态机的某一位寄存器被打翻，门级的状态机寄存器就可能会进入 $\bar{Q}$ 集合。除因单粒子翻转导致的状态异常外，还存在因外部输入时序的异步关系，导致状态机进入异常状态的情况。

对于状态转移函数 δ 和输出函数 λ，在状态机的行为描述中只约束了 $q \in Q$ 情况下的状态转移和输出处理，对于 $q \in \bar{Q}$ 的行为，将取决于综合布局布线后的实现结果，有时其处理结果可能会导致状态机出现死锁，即 $q_i = \delta(q_i, i_i)$。显然针对高安全性要求的应用中不允许出现这种情况。

为了解决上述问题，各综合工具中均提供了状态机安全实现（safe_implementation）的功能，状态机综合过程中，增加了对状态机的安全实现处理，即状态转移函数中，增加针对 $q \in \bar{Q}$ 的处理，新的状态转移函数更改为如下形式，避免状态转移出现死锁或未知的处理结果。

$$\delta: \begin{cases} Q \times I \to Q, \text{当} q \in Q \\ \bar{Q} \times I \to Q, \text{当} q \in \bar{Q} \end{cases}$$

3 Vivado 综合器的状态机安全实现

在 Vivado 中，综合器针对状态机的安全实现，相对 ISE 有了非常大的变化，变为了 4 种可选择的状态机安全模式[21]，安全状态机的设置方式也发生很大变化。

在 Vivado 环境中，采用“FSM_SAFE_STATE”综合属性，用于实现安全状态机。该综合属性可以设置在 VHDL 或 Verilog 的 RTL 文件中，也可以设置在 XDC 约束文件中。FSM_SAFE_STATE 属性共有如下 4 种设置选择：

default_state：强制状态机进入 RTL 代码中指定的 default 状态。即状态转移 case 语句的 default/others（Verilog/VHDL）分支中指定的状态；

reset_state：使用 Hamming-2 编码方式，可以实现单 bit 状态的错误检测，强制状态机进入复位状态；

power_on_state：使用 Hamming-2 编码方式，可以实现单 bit 状态的错误检测，强制状态机进入上电状态；

auto_safe_state：使用 Hamming-3 编码方式，可以实现单 bit 状态错误的自动纠正。

上述四种设置中，所谓的复位状态（reset state）是指状态机寄存器进行复位时进入的状态，上电状态（power on state）是指 SRAM 型 FPGA，在完成配置加载后的初始状态，对应于状态机寄存器定义时设置的初始值。例如如下的 VHDL 代码中，S_RESET 对应为复位状态，S_CALC_RETTA 为上电状态。

```
process(reset,clock)
  variable stato : integer range 6 downto 0 := S_CALC_RETTA;
  variable cont,mar,x,y,t :num8bit;
begin
  if reset = '1' then
    stato := S_RESET;
    ......
  elsif clock'event and clock = '1' then
    case stato is
      when S_RESET =>
        ......
    end case;
  end if;
end process;
```

图 1 复位状态与上电状态

Vivado 综合器所增加的多种状态机安全实现选择，其具体的实现方式，包括如何判断状态 $q \in \bar{Q}$、状态转移函数 δ 中增加了哪些附加逻辑，特别是状态输出函数 λ 增加了哪些处理，Xilinx 提供的文档中并未提供明确说明，对于用户来说都属于黑盒状态。因此，为了将状态机安全实现结果的进行白盒化，对门级状态机的精确行为进行分析，可以更好的指导状态机使用。

4 门级状态机行为的分析方法

对于完成综合布局布线的门级网表，由于其可读性差、规模巨大，难于通过阅读网表的方式分析其具体功能，传统上可以采用门级仿真的方式，设计 TestBench，输出测试激励，观察门级网表的输出结果，了解门级网表的具体功能。但这种方式对测试激励的依赖性较大，如果输入的测试激励未能触发其功能处理场景，通过输出也无法观测到对应的处理行为。

逻辑等价性检查是一种基于形式化技术对门级网表进行功能分析的方法。通过逻辑等价性检查，比较状态机 RTL 代码中的行为描述，与门级网表电路的逻辑一致性，在满足逻辑一致的前提下，通过 RTL 代码的行为描述，就可以了解状态机的具体行为。但对于安全实现后的状态机，由于综合工具在安全实现过程中引入的附加逻辑，在 RTL 行为描述中并没有体现，在进行逻辑等价性检查的过程中，文献[20]中使用状态机重编码约束（guide_fsm_reencoding）的方法，将逻辑锥比较的输入限定在状态机的合法空间 Q 集合内，因此无法反映出安全状态机的全部行为。

本文提出一种改进的方式，不使用状态机重编码等约束，实现针对安全状态机全部状态空间的比较，从而精确得到安全状态机的全部行为，包括进行安全实现后的状态转移函数和输出函数。具体的流程如下：

Step1. 加载完成综合布局布线后的门级网表，作为实现设计（IMPL），IMPL 中的状态机已经完成状态机安全实现。

Step2. 根据综合约束设置，以及 Vivado 输出的综合报告 vds 文件，确定综合后的状态机编码方式，以及门级状态机的寄存器位数。

Step3. 修改 RTL 代码，根据门级状态机情况，修改状态寄存器的定义，以及编码设置。

Step4. 加载 RTL 代码，执行逻辑等价性验证过程，包括如下的子步骤。

Step4.1. 加载 RTL 代码，作为参考设计（REF）。

Step4.2. 进行匹配（Match）操作，实现 REF 和 IMPL 中所有的驱动点和比较点的对应，重点解决状态机寄存器的对应。如果未能实现一一对应，则修改 RTL 代码，重新回到 Step4 执行。

Step4.3. 进行验证（Verify）操作，遍历驱动点构成的状态空间，检查各比较点是否满足逻辑等价。

Step4.4. 针对不等价的比较点，进行调试（Debug）操作，分析驱动点的图样（pattern），并结合门级仿真等方法，推测门级电路的真实行为，修改 RTL 代码，重新回到 Step4 执行。直至验证成功。

由于在比较过程中，没有使用约减状态空间的设置，此时得到的 RTL 在全状态空间上都与综合后的门级网表等价，RTL 代码有具备良好的可读性，分析 RTL 代码，就可以实现对综合后的安全状态机行为的精确分析。

5 实验及结果分析

依据本文提出的分析方法，选择包含状态机的 VHDL 和 Verilog 的代码样本，进行状态机安全实现后的精确行为分析。

5.1 实验方案

代码样本从 IWLS 2005 Benchmarks 进行选取，IWLS 2005 Benchmarks 中共有 84 套设计代码，分为了 ICT99、ISCAS 85 and 89、OpenCores、Gaisler Research 和 Faraday Technology Corporation 几套代码样本集，从其中挑选包含状态机的设计代码作为样本，选择时覆盖 VHDL 和 Verilog 两种语言，状态机数的状态数量覆盖非 2 整数次幂情况和 2 整数次幂情况，以便体现综合后的不同结果。

为了对比安全状态机不同选项，在不考虑代码功能的情况下，对代码进行修改。修改的过程中主要考虑增加如下方面的内容：

(1) 为状态机设置不同的复位状态和上电状态，以体现安全状态机不同选项的区别。

(2) 在状态转移 case 语句的 default/others（Verilog/VHDL）分支中，增加状态转移处理，转移后的新状态不同于复位状态和上电状态，用来实验综合器识别出状态机后，对 default/others 的优化处理情况。

(3) 在状态转移 case 语句的 default/others 分支中，增加其他处理逻辑，用来实验综合器识别出状态机后，对 default/others 的优化处理情况。实验中通过在 default/others 中增加对输出端口操作的方式，以

免其他原因造成的逻辑优化。

(4) 在状态转移 case 语句之外，增加根据当前状态，进行逻辑控制的语句。主要原因是，代码样本中，部分 RTL 代码将状态机的输出函数与状态转移的 case 语句耦合在了一起，不便于直观体现安全实现对状态机输出函数的影响。新增的状态输出函数包括判断状态“等于”和状态“不等于”的情况。同样为了避免出现其他原因导致的优化，增加逻辑控制为直接操作端口。

以 b10 样本代码为例，修改前后的对比如下图所示，其中左侧为原始代码，右侧为修改后的代码。

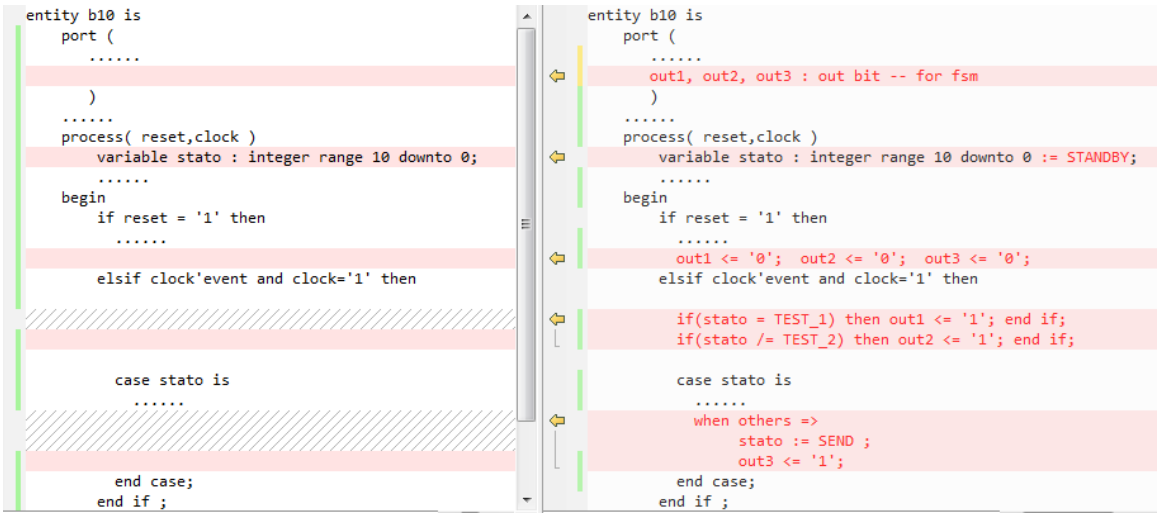


图 2 综合前代码修改对比

针对选定的样例代码，使用 Vivado 2019.1 版本进行综合布局布线。每个样本的综合过程中覆盖全部 4 种状态机安全选项设置，同时增加未设置安全状态机的情况作为对照。完成综合布局布线后，输出 Verilog 格式的门级网表。针对所得到的门级网表，采用本文提出的分析方法，对所有的安全状态机进行精确行为分析，总结不同方式对应的结果。

5.2 未设置安全状态机的结果分析

通过分析可以看出，当 Vivado 综合器识别出状态机后，根据优化的不同情况，会出现综合器对 default/others 分支中进行优化的不同情况。

例如 b10 样例代码中，针对 out3 端口的输出驱动逻辑，由于代码中只在复位处理中驱动 out3 端口为 0，default/others 分支中对 out3 端口驱动为 1 的逻辑被优化，最终门级电路中，out3 端口被实现为输出为 0 的固定电路。同时针对 default/others 分支中状态机跳转到 SEND 状态的处理，也被综合器经优化为保持状态不变。b10 样例综合前后状态机的行为对比如图 3 所示，其中左侧代码为原始 RTL 代码，右侧代码为与门级状态机行为精确一致的代码。本文中，如无特殊说明，代码对比结果中均采用此种排列方式。

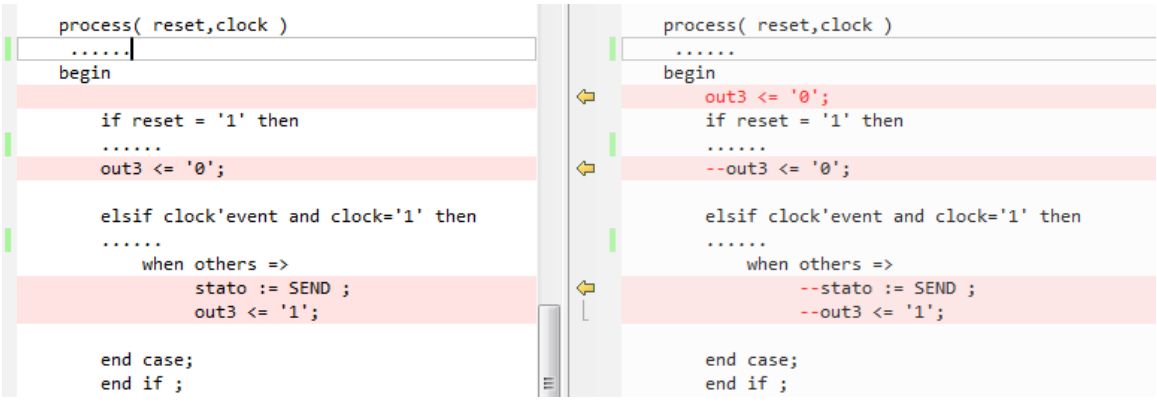
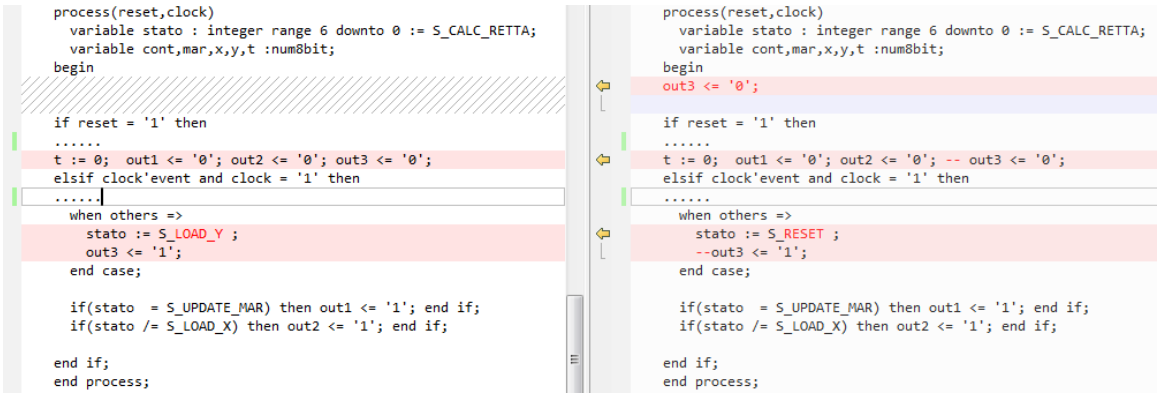


图 3 b10 样本不设置安全实现综合前后功能对别

但是这种综合后的结果，取决于综合工具的优化过程，具有不确定性，例如，在 b07 样本中，综合后

结果除了 out3 被优化为常量外，状态更新操作由综合前的跳转到 S_LOAD_Y 状态，实现为跳转至 S_RESET 状态。b07 样本未设置安全状态机的综合结果对比如下：



```
process(reset,clock)
  variable stato : integer range 6 downto 0 := S_CALC_RETTA;
  variable cont,mar,x,y,t :num8bit;
begin


if reset = '1' then
......
t := 0;  out1 <= '0'; out2 <= '0'; out3 <= '0';
elsif clock'event and clock = '1' then
.....
  when others =>
    stato := S_LOAD_Y ;
    out3 <= '1';
  end case;

  if(stato  = S_UPDATE_MAR) then out1 <= '1'; end if;
  if(stato /= S_LOAD_X) then out2 <= '1'; end if;

end if;
end process;
```

```
process(reset,clock)
  variable stato : integer range 6 downto 0 := S_CALC_RETTA;
  variable cont,mar,x,y,t :num8bit;
begin
out3 <= '0';

if reset = '1' then
......
t := 0;  out1 <= '0'; out2 <= '0'; -- out3 <= '0';
elsif clock'event and clock = '1' then
......
  when others =>
    stato := S_RESET ;
    --out3 <= '1';
  end case;

  if(stato  = S_UPDATE_MAR) then out1 <= '1'; end if;
  if(stato /= S_LOAD_X) then out2 <= '1'; end if;

end if;
end process;
```

图 4 b07 样本不设置安全状态机综合前后代码对比

在 usb_tx_phy 样例，usb_tx_phy.v 中 state 状态机，未设置状态实现时，综合器针对状态机采用 1-hot 方式编码，并且 default 分支中的 out3 驱动操作和状态跳转为 EOP1 状态的操作，均被保留了下来。门级状态机的处理结果与 RTL 代码描述一致。

由此可以看出，当未设置安全状态机选项时，default/others 分支的综合实现结果具有较大的不确定性，最终门级电路的行为取决于综合器根据上下文综合优化的结果。因此针对高安全性要求的 FPGA 代码，建议通过设置状态机安全选项，避免综合实现结果的不确定性。

5.3 default_state 方式的结果分析

设置安全状态机选项为 default_state 方式，对各样本进行综合布局布线。通过分析可以看出，首先状态机编码方式与 5.2 节不同，不再根据状态机编码选项选择不同的编码方式，而是采取固定的二进制编码方式。例如 usb_tx_phy 样本综合报告中给出的状态机编码结果如下：

```
                State |              New Encoding |             Previous Encoding
-----------------------------------------------------------------------------------------------
*
                 IDLE |                       000 |                       000001
                  SOP |                       001 |                       000010
                 DATA |                       010 |                       000100
                 EOP1 |                       011 |                       001000
                 EOP2 |                       100 |                       010000
                 WAIT |                       101 |                       100000
-----------------------------------------------------------------------------------------------
```

图 5 usb_tx_phy 样本 default_state 方式综合后编码

可以看出，usb_tx_phy 样本中状态机的状态数量为 6，不满足 2 的整数次幂，综合报告给出状态编码时，用*号表示还有空余状态。default/others 分支中的处理逻辑被完成保留，门级状态机完全按照 RTL 代码中的 default/others 分支实现。

当状态机的状态数量为 2 整数次幂时，此时没有空余状态，综合报告中也不再使用*号，default/others 分支中也变成不可达分支，其中的代码对综合结果没有影响，会被综合器直接忽略。

此外，门级状态机的二进制编码顺序由综合器根据综合过程决定，未必与 RTL 代码中的状态编码顺序保持一致。

b10 样本 default_sate 方式综合前后的代码对比结果如下：

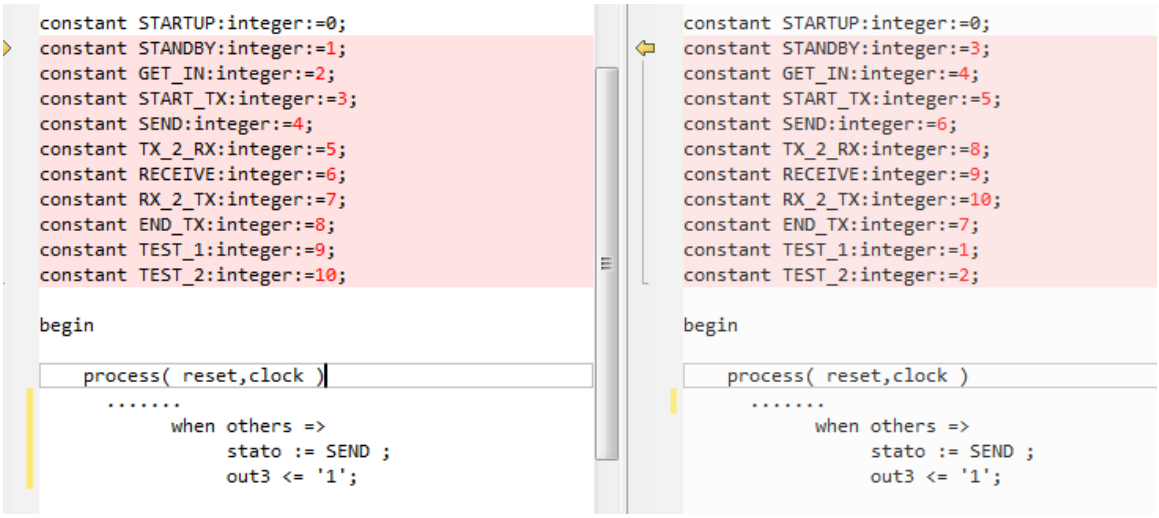


图 6 b10 样本 default_state 方式综合后编码

5.3 reset_state 方式的结果分析

设置安全状态机选项为 reset_state 方式，进行同样类似的分析。首先对于状态机的编码，根据 Xilinx 的手册，reset_state 方式下，状态机编码采用 Hamming-2 编码方式，分析综合报告可以看出，所谓 Hamming-2 方式，是在二进制顺序编码的基础上，增加了一位偶校验位。通过引入校验位，实现门级不同状态的码距最小为 2。b10 样本 reset_state 方式，综合报告中给出的状态机编码如下所示：

```
INFO: [Synth 8-5552] Implemented safe state 'startup' for state register 'stato_reg' in module 'b10'
-----------------------------------------------------------------------------------------------------
                   State |                     New Encoding |               Previous Encoding
-----------------------------------------------------------------------------------------------------
                 startup |                            00000 |                                 0000
                  test_1 |                            10010 |                                 1001
                  test_2 |                            10100 |                                 1010
                 standby |                            00011 |                                 0001
                  get_in |                            00101 |                                 0010
                start_tx |                            00110 |                                 0011
                    send |                            01001 |                                 0100
                  end_tx |                            10001 |                                 1000
                tx_2_rx |                            01010 |                                 0101
                receive |                            01100 |                                 0110
                rx_2_tx |                            01111 |                                 0111
-----------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'stato_reg' using encoding 'Hamming 2' in module 'b10'
```

图 7 b10 样本 reset_state 方式综合后编码

可以看出，门级状态机状态向量的最低位为新增加的校验位，从最高位到校验位之前为二进制顺序编码。校验位采用偶校验的算法，即将所有状态位进行逻辑异或，得到校验位。

对于 Verilog，校验位可以采用“归约异或”操作符实现，例如：state_p <= ^next_state。对于 VHDL，为了方便，采用函数的方式进行校验位生成和校验位检查，设计了 function mk_pbits(state: integer)函数用于校验位生成，function chk_state(state: integer; p_bit: bit)用于校验位检查，其中 state 为输入状态，p_bit 为输入校验位，返回值为'1'代表校验正确，'0'代表校验位错误。

通过分析可以看出，reset_state 方式综合安全状态机时，进行的处理主要有：

(1) 状态机编码中增加一位状态校验位，本文中将新增加的状态校验位定义为 fsm_p。

(2) 状态机复位时，增加针对 fsm_p 的复位处理。

(3) RTL 代码 case 语句对应的状态转移处理，在门级状态机中，改为仅在状态寄存器校验正确时才进行，当校验错误时，状态机切换到 reset 状态，并更新 fsm_p。

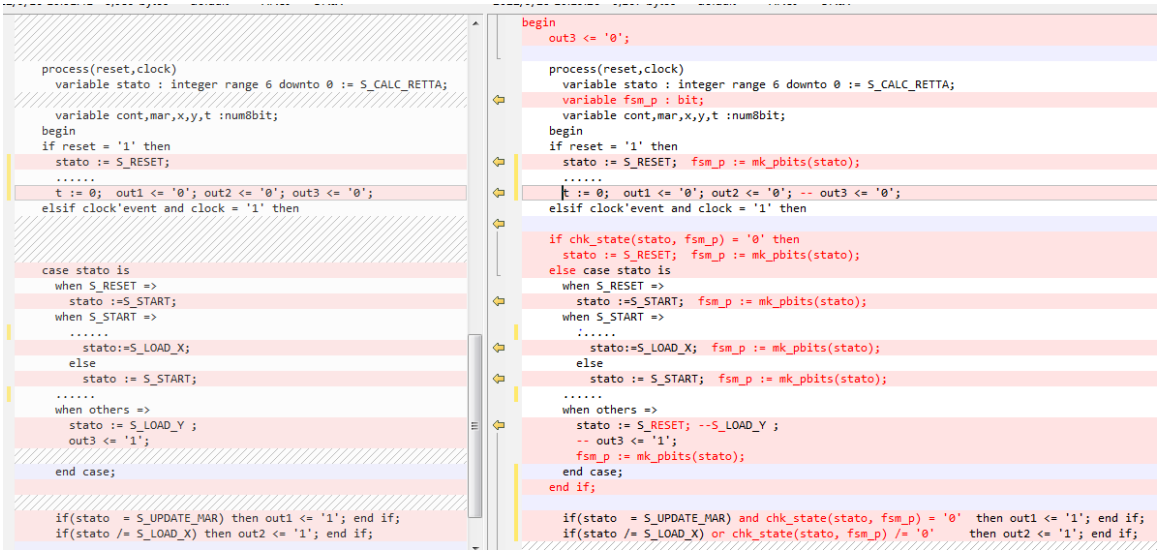
(4) 对于原 RTL 代码中每一个状态转移语句，均增加更新 fsm_p 的处理。

(5) 状态机的输出函数中，增加针对状态寄存器的校验判断，当输出函数为“判断状态等于”时，增加的校验判断为“且状态校验正确”，当输出函数为“判断状态不等于”时，增加的校验判断为“或校验错误”。

(6) 由于门级状态机中，状态转移的 case 语句位于状态校验正确的分支，default/others 分支对应于校验正确的前提下，状态机进入未定义的状态空间。此种情况对应为出现两位以上的状态机寄存器翻转场景，在只有一位翻转的情况下不会进入 default/others 分支。对于 default/others 分支中的状态转移语句，门级状态机中实现的结果，取决于综合工具根据上下文优化的情况，门级电路的行为存在不确定性，例如 b07 样本实现为切换到 reset 状态，b10 样本实现为维持此前状态保持不变。

(7) case 语句 default/others 分支中的其他输出控制逻辑，被综合器优化。在本文中对应为 out3 寄存器被优化实现为常量电路。

b07 样本 reset_sate 方式综合前后的代码对比结果如下：



```
process(reset,clock)
  variable stato : integer range 6 downto 0 := S_CALC_RETTA;
  variable cont,mar,x,y,t :num8bit;
begin
if reset = '1' then
  stato := S_RESET;
  ......
  t := 0;  out1 <= '0'; out2 <= '0'; out3 <= '0';
elsif clock'event and clock = '1' then

case stato is
  when S_RESET =>
    stato :=S_START;
  when S_START =>
    ......
      stato:=S_LOAD_X;
    else
      stato := S_START;
  ......
  when others =>
    stato := S_LOAD_Y ;
    out3 <= '1';
  end case;

  if(stato  = S_UPDATE_MAR) then out1 <= '1'; end if;
  if(stato /= S_LOAD_X) then out2 <= '1'; end if;
```

```
begin
  out3 <= '0';

  process(reset,clock)
    variable stato : integer range 6 downto 0 := S_CALC_RETTA;
    variable fsm_p : bit;
    variable cont,mar,x,y,t :num8bit;
  begin
  if reset = '1' then
    stato := S_RESET;  fsm_p := mk_pbits(stato);
    ......
    t := 0;  out1 <= '0'; out2 <= '0'; -- out3 <= '0';
  elsif clock'event and clock = '1' then

  if chk_state(stato, fsm_p) = '0' then
    stato := S_RESET;  fsm_p := mk_pbits(stato);
  else case stato is
    when S_RESET =>
      stato :=S_START;  fsm_p := mk_pbits(stato);
    when S_START =>
      ......
        stato:=S_LOAD_X;  fsm_p := mk_pbits(stato);
      else
        stato := S_START;  fsm_p := mk_pbits(stato);
    ......
    when others =>
      stato := S_RESET; --S_LOAD_Y ;
      -- out3 <= '1';
      fsm_p := mk_pbits(stato);
    end case;
  end if;

  if(stato  = S_UPDATE_MAR) and chk_state(stato, fsm_p) = '0'  then out1 <= '1'; end if;
  if(stato /= S_LOAD_X) or chk_state(stato, fsm_p) /= '0'     then out2 <= '1'; end if;
```

图 8 b07 样本 reset_state 方式实现结果

当综合前代码中针对状态机状态没有进行 reset 处理时，综合报告中将会给出警告，Vivado 将会按常规方式进行综合，无法进行状态机安全实现。

5.4 power_on_state 方式的结果分析

设置安全状态机选项为 power_on_state 方式，进行同样类似的分析。可以看出，power_on_state 方式下，与 reset_state 方式处理的结果非常类似，状态机编码也是采用 Hamming-2 编码方式，增加 1 位偶校验。门级状态机所进行的处理与 reset_state 方式的区别，仅在校验位错误时，状态机转移切换至 power_on_state，其余处理结果一致。

当综合前代码中针对状态机状态没有显式的定义上电状态时，与 reset_state 方式不同，SRAM 型的 FPGA 完成配置加载后总会有一个上电初始状态，Vivado 将会根据优化结果，选择某状态作为上电状态，进行 power_on_state 方式的状态机安全实现。

5.5 auto_safe_state 方式的结果分析

设置安全状态机选项为 auto_safe_state 方式，进行同样类似的分析，可以看出 auto_safe_state 方式下，门级状态机的处理最为复杂。

首先 auto_safe_state 方式下，状态机编码采用 Hamming-3 编码方式，所谓 Hamming-3 编码，即采用海明码[22]的方式增加校验位，实现对状态机编码的检错和纠错。相比之下，reset_state 和 power_on_state 模式的偶检验编码，不能纠正错误位，并且只能检测奇数个错误发生。auto_safe_state 模式的编码，可以实现检测两 bits 错误，纠正单 bit 错误。

```
INFO: [Synth 8-5552] Implemented safe state 'auto_safe_state' for state register 'stato_reg' in module 'b07'
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |               Previous Encoding
---------------------------------------------------------------------------------------------------
                 s_reset |                           000000 |                              000
                 s_start |                           010101 |                              001
                 s_load_x |                           100110 |                              010
              s_update_mar |                           110011 |                              011
                 s_load_y |                           111000 |                              100
              s_calc_retta |                           101101 |                              101
              s_incrementa |                           011110 |                              110
---------------------------------------------------------------------------------------------------
```

图 9 b07 样本 auto_safe_state 方式状态机编码

设 RTL 中的状态机经过二进制编码后的原始状态编码为 $I=b_0b_1b_2...b_{k-1}$，共 k 位，校验位 $P=p_0p_1p_2...p_{p-1}$，共 p 位，k、p 满足 $k+p\le 2^n-1$。对于不同的 p，对应 k 的范围以及可支持最大原始状态数量如下表所示：

表 1 p 对应最大状态数量

校验位位数p	对应k的范围	可支持最大原始状态数量
1	-	-
2	1	2
3	2~4	16
4	5~11	2048

可以看出，p 值为 2~4，就可以满足大多数 RTL 代码的设计需求。

校验位 P 中各校验位的生成算法以及校验算法如下表所示：

表 2 海明编码及校验算法

位置	1	2	3	4	5	6	7
原始状态			b0		b1	b2	b3
校验位	p0	p1		p2			
海明编码	p0	p1	b0	p2	b1	b2	b3
编码算法	p0 = b0 xor b1 xor b3 p2 = b0 xor b2 xor b3 P2 = b1 xor b2 xor b3						
校验算法	s0 = p0 xor b0 xor b1 xor b3 s1 = p1 xor b0 xor b2 xor b3 s2 = p2 xor b1 xor b2 xor b3						

首先按 1、2、3…n 的序号对位置进行编排，校验位 p_i 填入序号为 2 整数幂的位置 2^i，原始状态的编码 b_n 按从小到大填入剩余的空位。对于位置序号为 2^i 的校验位，其生成算法为选择位置序号中第 i 位为 1 的信息位进行异或，就可以得到最终编码。

当进行 EDAC 校验时，对于 $i=0,1,2...$，依次对位置序号第 i 位为 1 的信息位和校验位进行异或，得到 $S=s_is_{i-1}...s_1s_0$，当 S 为 0 时，表示没有发生错误，当 S 不为 0 时，表示位置序号为 S 的状态位或校验位发生错误。若出现 S 大于位序号最大值，表示发生了不可纠正的错误，即发生 2 位以上的错误。

设计 mk_pbits(state: integer)函数，用于生成所需的校验位；chk_state(state: integer; p_bits: bit_vector)函数用于进行编码校验，返回值 1 代表未发生错误或发生的错误可纠正，返回 0 表示发生不可纠正错误；edac_state (state: integer; p_bits: bit_vector)函数用于返回 EDAC 后的状态编码，当发生不可纠正错误时，维持当前状态编码不变。

基于上述设计，对 auto_safe_sate 方式的安全状态机进行分析，可以得到如下结果：

(1) 对状态机进行二进制顺序编码，并根据 RTL 代码中状态数量的不同，增加一定数量的状态校验位，本文中将新增加的状态校验位定义为向量 p。

(2) 状态机复位时，增加针对校验向量 p 的复位处理。

(3) case 语句对应的状态转移处理，仅在状态校验没有错误或发生可纠正错误时进行。当发生不可纠正错误时，门级状态机的转移情况取决于综合器的优化结果，具有不确定性。

(4) 所有的状态判断语句，包括状态转移 case 语句中的状态判断，由 RTL 代码中的原始状态，改为判断 EDAC 后的状态（本文中用 aux_state 表示），以避免单 bit 错误造成的错误状态判断。

(5) 针对原 RTL 代码中的每一处状态更新操作，增加针对校验向量 p 的更新操作。

(6) 状态机的输出函数中，增加针对状态寄存器的校验判断，当输出函数为“判断状态等于”时，增加的校验判断为“且未发生不可纠正错误”，当输出函数为“判断状态不等于”时，增加的校验判断为“或发生不可纠正错误”。

(7) 由于门级状态机中，状态转移的 case 语句的 default/others 分支，对应于未发生错误或错误可纠正的前提下，状态机进入未定义的状态空间，此种场景在只有一位翻转的情况下不会出现。对于 default/others 分支中的状态转移语句，门级状态机中实现的结果，取决于综合工具根据上下文优化的情况，存在不确定性。

(7) case 语句中的其他输出控制逻辑，被综合器优化。在本文中对应为 out3 寄存器被优化实现为常量电路。

b07 样本 auto_safe_sate 方式综合前后的代码对比结果如下：

```
    process(reset,clock)
      variable stato : integer range 6 downto 0 := S_CALC_RETTA;

      variable cont,mar,x,y,t :num8bit;
    begin
    if reset = '1' then
      stato := S_RESET;
      ......
      t := 0;  out1 <= '0'; out2 <= '0'; out3 <= '0';
    elsif clock'event and clock = '1' then

    if(stato  = S_UPDATE_MAR) then out1 <= '1'; end if;
    if(stato /= S_LOAD_X)     then out2 <= '1'; end if;

    case stato is
      when S_RESET =>
        stato :=S_START;
      ......
      when others =>
        stato := S_LOAD_Y ;
        out3 <= '1';

      end case;

    end if;
```

```
begin
    out3 <= '0';

    process(reset,clock)
      variable stato : integer range 6 downto 0 := S_CALC_RETTA;
      variable aux_state : integer range 6 downto 0;
      variable p : bit_vector(2 downto 0);
      variable cont,mar,x,y,t :num8bit;
    begin
    if reset = '1' then
      stato := S_RESET;   p := mk_pbits(stato);
      ......
      t := 0;  out1 <= '0'; out2 <= '0'; -- out3 <= '0';
    elsif clock'event and clock = '1' then

    aux_state := edac_state(stato, p);
    if(aux_state  = S_UPDATE_MAR) and chk_state(stato, p)  = '1' then out1 <= '1'; end if;
    if(aux_state /= S_LOAD_X)     or  chk_state(stato, p) /= '1' then out2 <= '1'; end if;
    if(chk_state(stato, p)='0') then  -- can't edac,
      stato := S_RESET;  p := mk_pbits(stato);
    else case aux_state is
      when S_RESET =>
        stato :=S_START;  p := mk_pbits(stato);
      ......
      when others =>
        stato := S_RESET; --S_LOAD_Y ;
        -- out3 <= '1';
        p := mk_pbits(stato);
      end case;
      end if;
    end if;
```

图 10 b07 样本 auto_safe_state 方式实现结果

此外，auto_safe_mode 模式下，还有一点特殊之处，由于状态编码具有 EDAC 的能力，当发生单 bit 错误时，可以对错误编码进行纠正更新。更新操作仅在原 RTL 代码中存在状态更新语句时，门级状态机才进行状态编码和校验向量的更新操作。即使 RTL 代码中存在 state<=state 这样不产生实际作用的语句描述，在门级状态机的行为中，也会替换为下列两条语句：

state <= edac(state,p);

p <= mk_pbits(edac(state,p));

这样可以避免状态机状态长时间不更新，导致单 bit 错误累计造成无法纠正错误。

5 结 论

本文提出了一种针对状态机门级实现行为的精确分析方法，并据此针对 Vivado 综合器的中所提供的 4 种安全状态机实现方式，分析了所实现安全状态机的精确行为。基于分析结果，对高安全应用场景下的状态机设计，总结了如下需要重点关注的设计要点：

(1) Vivado 中提供的 4 种状态机安全模式，具有不同的特点和侧重点：default_state 方式严格按照 RTL 代码描述的行为实现状态机运行；reset_sate 和 power_state 方式当状态机状态出现错误时，转入指定的状态重现开始运行；auto_safe_state 方式侧重在保证状态机的连续运行。

(2) 对于 auto_safe_state 方式，在 RTL 代码中确保每个分支中都有状态更新语句，即使当状态没有变化，也需要增加类似 state<=state 的语句，确保门级状态机可以在每个时钟周期对状态机寄存器进行 EDAC 刷新。

(3) 状态转移 case 语句的 others/default 分支中的逻辑代码，综合器往往会进行优化，综合结果具有不确定性。应避免在 others/default 分支中增加除状态跳转以外的其他语句，可以通过单独的状态输出函

数实现相应的逻辑操作。

(4)对于 auto_safe_state、reset_sate 和 power_state 方式，当状态机寄存器出现多 bits 错误导致不可纠正错误或无法校验错误时，门级状态机所实现的行为取决于综合器优化，具有不确定性。如果 FPGA 产品的安全性要求较高，需要通过精确行为分析的方法确认门级实现的最终结果是否可以接受。如无法接受则需要通过调整 RTL 代码进行反复迭代尝试，直至可接受。

(5)对于 default_state 方式，门级状态机的行为与 RTL 代码描述保持一致，不会出现综合过程引入的不确定性。但状态机状态编码不具备纠错和检错能力，发生单 bit 错误就能够导致状态机跳转到未定义状态或其他已定义状态，可能会造成正常工作过程出现中断，需要结合具体应用场景进行分析和使用。

本文提出的门级状态机精确行为分析方法，后续还可以进一步扩展到 Synplify、Precision 等其他常用的 FPGA 综合器，并总结其他综合器中安全状态机设计的关注要点。

参 考 文 献

[1]. Rafla N I, Davis L V, et al. A Study of Finite State Machine Coding Styles for Implementation in FPGAs[J]. Midwest Symposium on Circuits & Systems, 2006 ,1:337-341.doi: 10.1109/MWSCAS.2006.382066.

[2]. 王会斌, 呼延奇, 郑悦, 等.航天器空间辐射效应分析技术现状与思考[J]. 航天器环境工程, 2022, 39 (4):427-435.doi: 10.12126/see.2022.04.015.
WANG H B, HU Y Q, ZHENG Y, et al.The present situation of technologies for analysis of space radiation effects to spacecraft and some retrospection[J]. Spacecraft Environment Engineering, 2022, 39 (4):427-435. doi: 10.12126/see.2022.04.015.

[3]. 马英起,朱翔,李悦,韩建伟.机载电子设备大气中子单粒子效应脉冲激光实验方法研究[J].航天器环境工程,2021,38(3):364-369.doi: 10.12126/see.2021.03.018
MA Y Q, ZHU X, LI Y, et al. Pulsed laser test method for atmospheric neutron induced single event effects on avionics[J].Spacecraft Environment Engineering, 2021, 38(3): 364-369

[4]. 余永涛, 陈煜海, 余俊杰, 等. SRAM 型 FPGA 单粒子效应测试方法及试验验证[J]. 航天器环境工程, 2021, 38 (5):534-540.doi: 10.12126/see.2021.05.007
YU Y T, CHEN Y H, YU J J, et al. Testing methods and experimental verification for single event effects of SRAM-based FPGA[J].Spacecraft Environment Engineering, 2021, 38 (5): 534-540. doi: 10.12126/see.2021.05.007

[5]. Hadjicostis C N, Verghese G C, et al. Coding approaches to fault tolerance in linear dynamic systems[J]. IEEE Transactions on Information Theory,2005 ,51 (1): 210-228.doi: 10.1109/TIT.2004.839491.

[6]. 陈雷, 张瑶伟, 王硕, 等. FPGA 三模冗余工具的关键技术与发展[J]. 电子与信息学报, 2022, 44(6): 2230-2244.doi: 10.11999/JEIT210330
CHEN Lei, ZHANG Yaowei, WANG Shuo, et al. Key Technology and Development of Triple Modular Redundancy Tool for FPGA[J]. Journal of Electronics & Information Technology, 2022, 44(6): 2230-2244. doi: 10.11999/JEIT210330

[7]. Frigerio L, Salice F, et al. RAM-based fault tolerant state machines for FPGAs[C], IEEE International Symposium on Defect & Fault-tolerance in Vlsi Systems, Rome, Italy, 2007, 312-320. doi: 10.1109/dft.2007.1

[8]. 李列文, 桂卫华, 胡小龙. 一种基于 FPGA 的低功耗、容错状态机设计方法[J]. 湖南大学学报：自然科学版, 2010 ,37 (6): 77-82.doi: CNKI:SUN:HNDX.0.2010-06-017.
LI L W, Gui W H, HU X L. A FPGA-based Design Method of Low Power Fault-tolerance Finite State Machine[J]. Journal of Hunan University(Natural Sciences), 2010, 31(6):77-82. doi: CNKI:SUN:HNDX.0.2010-06-017.

[9]. 严健生, 杨柳青. 卫星用 SRAM 型 FPGA 抗单粒子翻转可靠性设计研究[J]. 科技创新与应用, 2021(9):48-50.
YAN J;YANG L Q, Reliability Design of Anti-Single Event Upset(SEU)of SRAM-FPGA for Satellites[J],Technology Innovation and Application, 2021(9): 48-50

[10]. Cassel M, Kastensmidt F L. Evaluating One-Hot Encoding Finite State Machines for SEU Reliability in SRAM-based FPGAs[C], IEEE International On-line Testing Symposium, Lake Como, Italy, 2006, 139-144.

[11]. 王鹏,邓智,范毓洋. 面向独热编码的有限状态机抗单粒子翻转设计[J]. 电讯技术, 2022, 62(8):1178-1183. doi： 10.3969/j.issn.1001-893x.2022.08.025
WANG P, DENG Z, FAN Y Y. Anti-SEU Design for Finite State Machine with One-hot Encoding[J], Telecommunication Engineering, 2022, 62 (8):1178-1183. doi: 10.3969/j.issn.1001-893x.2022.08.025

[12]. Sooraj S, Bhakthavatchalu R. Hamming 3 algorithm for improving the reliability of SRAM based FPGAs[C], International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2017, 0938-0942. doi: 10.1109/ICCSP.2017.8286508.

[13]. Lupi M, Giubilato P, Bonora M, et al. Design of Finite State Machines for SRAM-based FPGAs operated in radiation field[C], Topical Workshop on Electronics for Particle Physics, Santiago De Compostela, Spain, 2019, 370, 129. doi: 10.22323/1.370.0129.

[14]. Huang Z, Liang H. The Impact of MBUs on the Reliability of Rollback Recovery Circuits[C], EEE Circuits and Systems International Conference on Testing and Diagnosis(ICTD), Chengdu, China, 2009, 1-4. doi: 10.1109/CAS-ICTD.2009.4960892.

[15]. Ejlali A, Al-Hashimi B M, Schmitz M T, et al. ombined time and information redundancy for SEU-tolerance in energy-efficient real-time systems[J]. IEEE Transactions on Very Large Scale Integration Systems, 2006 ,14 (4): 323-335.doi: 10.1109/TVLSI.2006.874355.

[16]. Uma R, Dhavachelvan P,. Finite state machine optimization in FPGAs[J]. International Journal of Vlsi Design & Communication Systems, 2013 ,3 (6): 79-89.doi: 10.1145/2393216.2393251.

[17]. Nidhin T S, Bhattacharyya A, Behera R P, et al. Verification of fault tolerant techniques in finite state machines using simulation based fault injection targeted at FPGAs for SEU mitigation[C], 2017 4th International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 2017, 153-157. doi: 10.1109/ECS.2017.8067859.

[18]. Liszewski K, Mcdonley T, Delozier J, et al. Netlist Decompilation Workflow for Recovered Design Verification, Validation, and Assurance[C/OL], International Cryptology Conference, 2021, 780.

[19]. Liszewski K, McDonley T. Understanding Tool Synthesis Behavior and Safe Finite State Machine Design[J]. arXiv preprint,2021.doi: 10.48550/arXiv.2108.04042

[20]. 朱倩, 田甜, 赵欢. 基于 FPGA 中状态机的逻辑等价性验证方法[J]. 南通大学学报：自然科学版, 2016 ,15 (3): 45-49.doi: CNKI:SUN:NGZK.0.2016-03-008.

ZHU Q, TIAN T, ZHAO H. Logic equivalence checking based on the state machine in FPGA [J]. Journal of Nantong University (Natural Science Edition), 2016, 15 (3):45-49. doi: CNKI:SUN:NGZK.0.2016-03-008.

[21]. Xilinx. Vivado Design Suite User Guide: Synthesis(UG901)[OL]. https://docs.xilinx.com/r/en-US/ug901-vivado-synthesis, 2022-11-16.

[22]. 蔡永涛, 张国华, 曹靓,等. FPGA 嵌入式多位宽 SRAM 的加固设计与实现[J]. 中国集成电路, 2021, 30 (3):24-28.doi: 10.3969/j.issn.1681-5289.2021.03.005

CAI Yong-tao, ZHANG Guo-hua, CAL liang, et al. Rugged design and implementation of FPGA embedded multi-bit wide SRAM[J]. China Integrated Circuit, 2021, 30 (3): 24-28. doi: 10.3969/j.issn.1681-5289.2021.03.005.

作者简介部分

于志杰：男，研究员，研究方向为 FPGA验证、软件测试。.

田甜：女，高工，研究方向为 FPGA验证、图像系统仿真。.

朱倩：女，高工，研究方向为 FPGA验证、通信系统仿真。.

童宗挺：男，工程师，研究方向为航天器 FPGA产品验证、FPGA设计。.

基于边界度和局部密度的不平衡数据软件缺陷预测过采样方法

陶雪梦[①②] 赵康[①②] 范文磊[①②] 楼俊钢[*①②]

[①](湖州师范学院 长三角(湖州)智慧交通研究院 浙江 湖州 313000)

[②](湖州师范学院 浙江省现代农业资源智慧管理与应用研究重点实验室 浙江 湖州 313000)

摘 要：软件缺陷预测用于分类故障模块和非故障模块，软件缺陷数据集的类不平衡问题容易影响预测模型的性能。过采样方法是当前数据层面解决类不平衡问题的主流技术之一，然而现有的过采样方法大多倾向于在特定区域合成样本而没有考虑不平衡分布的复杂性。此外，在合成样本时大都存在引入噪声样本的风险。为此，提出一种基于边界度和局部密度的不平衡数据软件缺陷预测过采样方法(BDOS)。针对软件缺陷预测不平衡数据分类问题，在数据预处理阶段，使用 BDOS方法，基于边界度和局部密度对少数类样本进行加权，同时增强边界区域和安全区域，得到一个平衡的数据分布，然后使用 IPF 对过采样后的数据分布进行噪声过滤处理。在模型构建阶段，将前一阶段得到的数据分布送入LightGBM 训练，从而得到最终的不平衡数据缺陷预测模型。在 PROMISE库的 11个常用数据集上与10个经典的对比方法进行的数值实验验证了BDOS 方法的有效性。具体来说，11 个数据集中， BDOS 在 F-measure，G-mean，AUC与 Balance四个指标上的性能均值皆为最优，并且与排名第二方法相比均提高了 10% 左右的性能，尤其在 F-measure指标上的提升效果最为明显。

关键词：软件缺陷预测；不平衡数据；加权过采样；集成学习

中图分类号：TP311 **文献标识码**：A **文章编号**：

Imbalanced Oversampling based on Boundary Degree and Local Density for Software Defect Prediction

TAO Xuemeng[①②] ZHAO Kang[①②] FAN Wenlei[①②] LOU Jungang[①②]

[①](*Yangtze River Delta (Huzhou) Intelligent Transportation Research Institute, Huzhou University, Huzhou* 313000*, China*)

[②](*Zhejiang Provincial Key Laboratory of Intelligent Management and Application of Modern Agricultural Resources, Huzhou University, Huzhou* 313000*, China*)

Abstract:Software defect prediction can be applied to classify between faulty modules and non-faulty modules, while the performance of prediction models are likely to be influenced by the class imbalance of software defect datasets. The oversa mpling method is one of the mainstream techniques to solve the class imbalance at the datalevel, most of which, however, t end to synthesizing samples in a specific area without consideration of the complexity of imbalanced distribution. In additi on, the risk of introducing noise samples always exists in time of synthesis of samples. To this end, an imbalanced oversamp ling based on boundary degree and local density for software prediction(BDOS) is proposed. For purpose of solving the pro blem of imbalanced data classification in software defect prediction, BDOS is used to weight minority-class samples based o n the boundary degree and local density in the data pre-processing phase, simultaneously enhances the boundary area and security area to obtain a balanced data distribution, and then IPF is used to filter the noise of the sampled data distributio n.In the model construction stage, the data distribution obtained in the previous stage is sent to LightGBM for training,so as to obtain the final defect prediction model.The effectiveness of the BDOS was validated by numerical experiments with 10 classical comparison methods on 11 common data sets of the PROMISE database.Specifically, the average performance

[*]通信作者: 楼俊钢 loujungang0210@hotmail.com

基金项目: 浙江省杰出青年自然科学基金资助项目(LR20F020002)，湖州师范学院研究生科研创新项目(2023KYCX41)

Foundation Items: Zhejiang Provincial Outstanding Youth Natural Science Funded Project(LR20F020002), Graduate Research and Innovation Project of Huzhou Normal University(2023KYCX41)

of BDOS on F-measure, G-mean, AUC and Balance in the 11 datasets is the best, and the performance of BDOS is improved by about 10% compared with the second ranking approach, especially on F-measure.
Key words: Software defect prediction;Imbalanced data;Weighted oversampling;Ensemble learning

1 引言

不平衡样本分类在现实生活中具有广泛的应用，例如软件缺陷预测[1]、疾病检测[2]、欺诈检测[3]、故障诊断[4]、网络入侵检测[5]等。传统分类器大都以平衡的数据分布为前提，由高度不平衡的数据集训练得到的预测模型往往会偏向于多数类，出现学习偏置问题，而少数类在现实生活中往往有着更重要的意义，错误分类少数类样本可能会导致严重的后果。例如将一个癌症患者分类为健康状态，或者一个安全临界系统的错误操作被分类为正常状态。因此如何提升不平衡学习中少数类的分类精度逐渐成为研究热点。沿着智能化时代的发展趋势，各行各业对于软件系统的依赖逐渐加深，软件规模不断增长、功能日益复杂，软件缺陷漏洞也随之增加，为了保障软件质量，软件开发过程中一个非常重要的环节就是软件测试。软件缺陷预测是在测试阶段开始之前预测哪些模块更容易出现故障，通过将测试工作集中在那些预计有故障的模块上，减少测试时间、控制软件测试阶段不必要的资源消耗，减少软件测试的工作量，从而降低软件的开发成本。数据的类不平衡性是影响预测性能的一个重要问题，引入不平衡学习算法来处理数据集中故障类(少数类)和非故障类(多数类)的偏态分布，以期最大限度地提高软件缺陷预测模型的分类性能是研究重点。

为了缓解软件缺陷预测中的类不平衡问题，现有研究方法大致可以分为三类：(1)数据层面的方法[6-8]，通过重采样技术调整数据空间中多数类和少数类样本的数量来平衡分布。(2)算法层面的方法[9-11]，通过修改算法的训练过程,使得学习算法可以针对少数类取得更好的预测精度,典型方法包括代价敏感学习(cost-sensitive learning)方法。代价敏感学习考虑了多数类和少数类的不相等的错误分类代价，从而使少数类的重要性在学习阶段得到增强。然而，将代价敏感学习方法应用于软件缺陷预测时，其错误分类代价难以确定。(3)集成学习方法[12-16]，通过组合多个分类器来提升分类性能，采样技术与集成学习结合也是处理类不平衡数据的常用方法，采样方法改善少数类样本的劣势，使其得到更多的学习机会，集成学习则更关注容易被误分的实例以提高模型的分类精度。

数据层面的方法因其简单有效，且独立于后续训练分类模型的优势，逐渐成为不平衡学习的主流，因此本文侧重研究重采样方法。两种常见的采样技术是过采样和欠采样，过采样技术通过合成新的少数类样本来平衡数据集，欠采样通过删除数据集中的部分多数类样本,，缩小多数类的数据规模使得数据分布达到平衡。欠采样在采样过程中丢弃的实例可能包含对预测模型有用或重要的信息，相较于欠采样，过采样在不平衡学习中往往有更好的表现，因为过采样可以提供更高比例的安全样本。过采样方法中最为经典的是 Chawla 等人[6]提出的 SMOTE，该方法在两个少数类样本间通过线性插值的方式合成新样本。然而 SMOTE 在生成过程中没有考虑数据集中多数类样本和潜在噪声的基本分布，可能会导致类重叠的增加，并会引入额外的噪声样本。

现有的基于 SMOTE 的改进过采样方法按照采样区域的不同可以分为增强边界区域采样和增强安全区域采样。前者认为边界附近的少数类样本更容易被误分类，对提高分类性能更有用，因此倾向于找寻边界附近的难学样本以增强少数类的决策边界。后者因为考虑到 SMOTE 可能会引入重叠或噪声实例使类边界更难学习进而降低后续分类器的性能，倾向于在一个相对安全的区域合成新样本。本文考虑到不平衡数据分布的复杂性，结合以上两种采样策略的优势，提出一种基于边界度和局部密度的加权过采样方法 BDOS，同时增强边界区域和安全区域。

传统的过采样方法主要关注的是如何生成少数类样本，而忽略了将生成的样本直接标注为少数类是否正确。主流的过采样方法将所有合成样本视为少数类而不加过滤，这可能是不合理的，特别是对于那些接近分类边界的合成样本。这种简单的标注方法会扭曲原始数据集的分布，改变分类边界，导致大量的误分类。因此本文将迭代分区过滤器 IPF 引入到所提的加权过采样 BDOS 中。

此外，采样方法结合集成学习用来提高分类精度也是解决类不平衡问题的一类常用方法[17-23]。Galar 等人[17]综述了此类方法，论证了它们在处理类不平衡问题方面的有效性。相较于现有的集成算法，LightGBM 具有更快的训练效率、低内存占用、更高的准确率等优点。本文在模型构建阶段，将 BDOS 与

集成算法 LightGBM 结合形成最终的不平衡数据缺陷预测模型。

本文提出的基于加权过采样和集成学习的不平衡数据软件缺陷预测模型分为三个阶段：

1)不平衡数据预处理：该阶段首先计算每个少数类样本的边界度和局部密度对其进行加权，根据权重值自适应地确定每个少数类样本合成新样本的数量，并以 SMOTE 线性插值的方法合成样本从而得到一个平衡的数据集。

2)噪声过滤处理：使用 IPF 进行平衡数据集的噪声过滤清理。

3)LightGBM 缺陷预测：将过采样预处理阶段中得到的数据分布送入 LightGBM 中训练，从而得到最终的不平衡数据缺陷预测模型。

本文的其余部分组织如下：第 2 节简要回顾了相关工作，第 3 节对 BDOS 进行了详细描述，第 4 节介绍了实验设计与分析。最后，第 5 节给出了研究结论。

2 相关工作

近年来，大量学者对不平衡数据分类问题展开研究。SMOTE[6]是不平衡分类领域的热点技术之一，在两个少数类样本间通过线性插值的方式合成新样本使类分布平衡。SMOTE 首先对每一个少数类样本x_i，依据欧几里得距离确定其 K个少数类近邻(设 K=5)，随机选择其中一个近邻，执行如下公式(1)进行线性插值合成新样本。

$$x_{new} = x_i + rand(0,1) \times (x_m - x_i) \tag{1}$$

其中，rand(0,1)为介于 0 和 1 之间的随机数，x_m表示第 m 个近邻(m=1,2,...,M)。SMOTE 在样本合成过程中没有考虑数据集中多数类样本和潜在噪声的基本分布，可能会导致类重叠的增加，并会引入额外的噪声。

研究者们对 SMOTE 存在的上述不足展开研究，从不同的角度出发提出了许多基于 SMOTE 的改进方法。Han 等人[24]提出的 Borderline-SMOTE,将少数类样本划分为"Safe"、"Danger"和"Noise"三类，其中"Danger"类指靠近分类边界的样本，该算法只针对边界样本进行近邻线性插值，以强化少数类的决策边界，有效提高少数类样本的分类精度。Safe-level-SMOTE[25]关注的是远离决策边界的少数类实例，以避免引入新的重叠或噪声实例。给每个少数类样本分配一个安全等级,让新生成的样本更靠近安全等级高的样本，从而保证新样本分布在安全区域内。ADASYN[26]使用权重分布作为标准，根据每个少数样本学习的难易程度，自适应的决定需要生成的合成样本的数量。MWMOTE[27]首先识别难以学习的信息量丰富的少数类样本，根据它们与最近的多数类样本的欧几里得距离为它们分配权重，然后利用平均链接聚类算法对少数类样本进行聚类，从同一簇内的加权信息丰富的少数类样本中生成合成样本。WKMeans-SMOTE[28]首先使用特征加权的聚类算法对数据集聚类，然后使用“聚类一致性系数”筛选出少数类边界样本，并使用这个样本作为种子样本进行 SMOTE 过采样。BN-SMOTE[29]针对 Borderline-SMOTE 无法有效识别噪声样本且在某些情况下无法有效识别边界位置的少数类样本问题进行改进，BN-SMOTE 对原始少数类样本进行噪声过滤，然后以少数类样本近邻信息识别较难学习的少数类样本并对其进行过采样。

考虑到欧几里得距离在高维环境中的适用性不高并且现有的过采样方法大都将所有的特征视为同样的权重下对数据进行处理，FW-SMOTE[30]使用 Minkowski 距离定义少数类样本的邻域并结合特征加权策略，在处理类重叠以及高维数据集上拥有较大的优势。NaNSMOTE[31]通过自然最近邻自适应的确定每个少数类样本的邻域，从而保证类中心的少数类样本有更多的邻域来提升合成样本的泛化能力，边界样本则会有更少的邻居从而减少合成样本的误差。KNNOR[32]首先按照少数类样本到其第 k 个近邻样本的距离进行排序，从而忽略噪声数据，然后在种子样本及其 k 个近邻样本之间合成人工候选样本，最后使用 k 近邻分类算法测试人工候选样本，保留安全的人工候选样本。KNNOR 主要识别关键和安全的增强区域，从而更可靠的对少数类样本过采样，并保持对噪声的弹性。总的来说，这些改进方法的关键是如何选择信息丰富的少数类样本作为种子样本，或重新权衡少数类样本以自适应的生成数据。

此外，SMOTE 通常与一些清洗或过滤技术相结合(ADPCHFO[33],GSMOTE-NFM[34]，SMOTE-Tomek[35])来去除重叠或有噪声的实例，以减轻此类实例对后续分类性能的影响。SMOTE-IPF[36]结合了 SMOTE 和基于迭代集成的噪声滤波器(迭代分割滤波器)，在平衡不平衡数据分布的同时使决策边界更加规则。SMOTE-WENN[37]将 SMOTE 与加权编辑近邻规则(WENN)相结合，首先使用 SMOTE 线性插值合成少数类样本，然后 WENN 使用距离加权函数和 k 近邻规则删除危险的多数类样本和少数类样本，从而

在重叠区域尽可能多的保留少数类样本，来提升少数类的局部密度。KNSMOTE[38]结合 k-means 聚类和 SMOTE，通过分析原始样本类以及 k-means 聚类后的聚类样本类的差异，筛选出类别未发生变化的安全样本，并删除容易误分类的边界样本和噪声样本，然后对安全样本进行线性插值过采样。

与 SMOTE 不同，MAHAKIL[39]方法计算少数类样本的马氏距离，根据该值对实例进行降序排列，排序后的实例根据中心实例被划分为两个组，然后取每组中两个对应实例的平均值以生成合成实例，MAHAKIL 合成的数据有更好的多样性。COSTE[40]使用差分进化将复杂性分配给实例的每个特征，然后根据复杂度度量将所有实例按照升序排列，使预测模型更多地关注不那么复杂的实例，对相邻排列的实例对进行平均，以生成合成实例，从而实现平衡的数据集。WACIL[41]首先找到边界实例，通过加权平均质心的概念生成伪数据，并通过过滤过程过滤掉不合适的噪声数据，在处理软件缺陷预测不平衡数据分类问题上取得了不错的效果。

另一种可以应用于过采样的方法是集成策略。集成学习的思想经常被用于分类器中，通常指通过融合多个基分类器组成一个强分类器，来有效的提高整体预测性能的方法。常用的集成学习的思想有：Bagging、Boosting。对于非平衡数据集，boosting 通常比 bagging 表现得更好[42]。集成学习通常结合重采样方法一起，通过减少每个基分类器的类不平衡来提高各分类器的性能。RUSBoost[43]把 RUS 算法和 Boost 算法结合，在每一轮分类学习过程中增加对少数类的样本的权重，使得基学习器 (base learner) 能够更好地关注到少数类样本。RAMOBoost[44]借助 ADASYN 思想根据少数类样本的分布比自适应地调整抽样权值，对少数类生成新样本进行排序。再通过 AdaBoost 集成，迭代提高了边界难学实例的权重。SMOTENDEL[45]将过采样和集成学习应用到软件缺陷预测中，SMOTENDEL 首先根据不平衡率自适应确定过采样比例，并通过多次过采样得到多个平衡的训练集，再使用经典的回归算法在得到的平衡数据集上进行训练，最终便可得到一组预测模型。

3 本文所提方法

3.1 不平衡数据预处理

在不平衡数据预处理阶段提出一种基于边界度和局部密度的过采样方法(BDOS)，该算法的流程主要分为三个步骤：(1)少数类样本边界度估计(2)少数类样本局部密度估计(3)基于边界度和局部密度的加权过采样。

3.1.1 少数类样本边界度估计

一般认为，分类器的性能与其能正确分类靠近边界的样本的能力密切相关，因为这些样本携带了更多的任务信息。对边界样本加大采样权重，突出靠近边界的样本的主要影响是有必要的。已有文献通过 K 近邻的方式选择边界样本[46]，然而以 K 近邻中同类样本与异类样本数量的占比来选择边界样本仍存在一定的缺陷，因为这种方式忽略了样本局部邻域内近邻样本的分布，将近邻中同类样本和异类样本数量占比相同的样本等价为一类，然而实际上这些样本仍存在一定的差异，以图 1 为例更清晰的说明该差异。图 1(a)中对近邻样本中包含异类样本的少数类样本进行标注，计算其 K(K=5)近邻中异类样本数量的占比，以该比值作为该样本的标注值。图 1(b)中则是 Logistic Regression(LR)模型对当前数据分布训练后对边界处的少数类样本给出的预测值，预测该样本为少数类的概率。

如图 1 所示，样本 A 和样本 B 的 $K(K=5)$近邻中均含有 3 个多数类样本和 2 个少数类样本，以 K近邻的方式选择边界样本则会将这两个样本定义为一类样本，如图 1(a)中样本 A 和样本 B 的 K近邻中异类样本数量占比均为 0.6。但是可以发现，样本 A 明显比样本 B 更适合作为种子样本进行过采样，因为选取样本 B 作为种子进行线性插值合成时，存在引入低质量样本的风险，继而加重当前数据分布的类重叠，影响到后续分类模型的性能。为了能够有效区分样本 A 和样本 B 的差异，本文引入 LR 模型，使用当前数据分布训练得到一个 LR 分类模型再对少数类样本预测其分类正确的概率，从而得到更精准的边界度量。图 1(b)中可以发现 LR 模型预测样本 A 和 B 为少数类样本的概率分别为 0.4 和 0.25，相较于样本 B，样本 A 是更安全的。综上，BDOS 引入 LR 模型来获取更精准的边界样本，并更细致地刻画少数类样本，以弥补采用 K近邻方式寻找边界样本的不足。

靠近决策边界的少数类样本对噪声更敏感，这就降低了它们分类的确定性。预测概率接近 0.5 的样本大多分布在决策边界周围，BDOS 试图增强此类样本，从而适当扩大少数类的决策区域，以期提升少数类

的识别准确率。已有研究证明，基尼不纯度在取值上和熵近似，但其计算更加简单[47]，因此基尼不纯度也可以衡量数据的不确定性。基尼不纯度越高，说明数据的混乱程度越大。反之，基尼不纯度越低，说明数据的混乱程度越小，数据越有序。利用基尼不纯度，可以评估训练样本的类别确定性。基尼不纯度的定义如下：在分类问题中，假设序列 Z有 N个类，则序列 Z的基尼不纯度为：

$$Gini(z) = \sum_{n=1}^{N} p_n(1-p_n) = 1 - \sum_{n=1}^{N} p_n^2 \tag{2}$$

其中，n是某个类别，N是类别数量，p_n表示选中样本属于 n类别的概率，$(1-p_n)$为该样本被分错的概率。

本文采用加权过采样的方法，以边界度作为加权依据之一，对于每一个少数类样本$x_i \in D_{min}$，将其边界度BD_i定义如下：

$$BD_i = 1 - {P_i}^2 - (1-P_i)^2 \tag{3}$$

其中P_i为 LR 模型预测x_i属于少数类的概率。图 2 展示了P_i与BD_i的对应关系。对于一个少数类样本，当样本越接近决策边界时，其边界度值也越高。

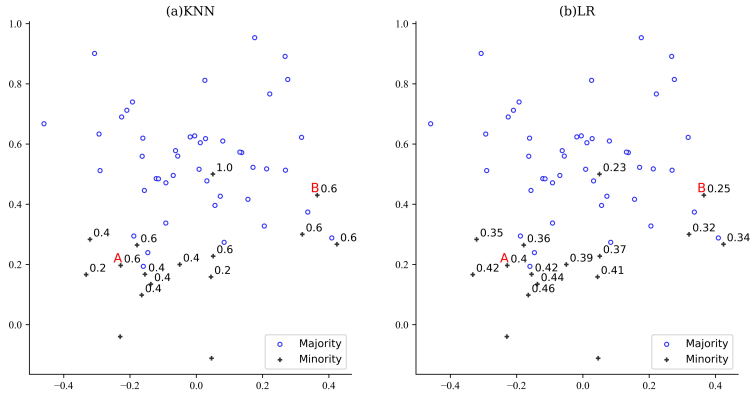


图 1 KNN 与 LR 方法选择边界样本的差异图

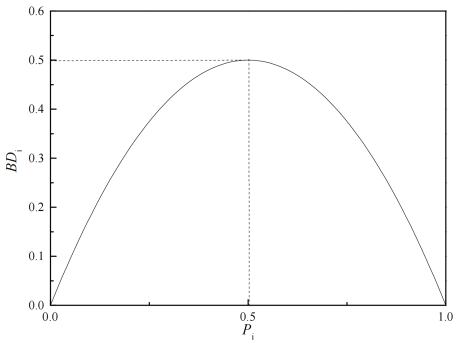


图 2 P_i与BD_i的对应关系

3.1.2 少数类样本局部密度估计

局部密度，表示局部范围内样本分布的密集程度。样本的局部密度可以在一定程度上反映该样本的重要性，并为识别关键的少数类样本提供帮助。幂指数核可以用于估计数据密度，定义如下：

$$\mathrm{K}(\mathrm{x},\mathrm{x}') = \exp^{-\frac{\|x-x'\|}{2\sigma^2}} \tag{4}$$

其中 x 是核函数的中心，$\|x-x'\|$表示任意给定向量 x 到空间中x'的欧式距离。随着两个向量之间距离的增加，幂指数核函数单调减小，说明核函数的作用域是局部的。σ为宽度参数，用于控制幂指数核函数的径向作用范围。σ的值越大，幂指数核函数的局部影响范围越大。

设$\mathrm{D} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{N}]$为给定的包含$D_{maj}$和$D_{min}$的训练集，即$D = D_{maj} \cup D_{min}$，其中 N是训练集 T中的样本数。D_{maj}和D_{min}和分别代表多数类和少数类。m是D_{maj}中的样本数，n是D_{min}中的样本数。对于每

一个x_i，使用幂指数核函数定义其局部密度，如下所示：

$$DS_i = \sum_{j \neq i}^{n} e^{-\frac{d(x_i,x_j)}{{d_c}^2}} \tag{5}$$

其中，$d(x_i, x_j)$表示少数类x_i与x_j之间的欧氏距离，参数d_c为截断距离，用百分比表示，将所有数据点两两之间的距离升序排列，d_c取这个距离集合中 1~2%所对应的距离值，本文将d_c设置为距离集合中 2%对应的距离值。由式(5)可以看出，一个少数类实例周围有更多的少数类实例，其局部密度就更高。为了避免它们大小值的影响，将其做归一化处理。

3.1.3 基于边界度和局部密度的加权过采样

由于 SMOTE 盲目地对每个少数类进行过采样，没有考虑到它们之间的区别。在 BDOS 中，同时考虑少数类样本距离决策边界的接近程度和该样本所在区域对应少数类的局部密度。考虑到上述两个因素(密度和边界度)均可用于区分少数类样本，BDOS 使用两者的线性加权和作为对应少数类样本的采样权重，ω_i定义如下：

$$\omega_i = \lambda BD_i + (1-\lambda)DS_i \tag{6}$$

由式(6)可知，一个给定的少数类样本加权过采样时的权重是由该样本的边界度和局部密度同时决定的。由于不平衡数据分布错综复杂，不同类型的数据分布需要增强的范围也会有所不同，因此本文引入参数λ进一步讨论，为了适应复杂的不平衡数据分布，故将参数λ设置为$\lambda \in [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]$，并对数据集进行寻优，通过控制$\lambda$的大小调整对数据分布的增强权重，随着$\lambda$数值的增大，边界区域权重也会增大，而安全区域的权重则会减小，以此为不同的数据分布找到较优的增强范围。在本文所提出的加权方法中，密度越高或边界度越高的样本权重也就越大，也就是说，这些样本在过采样合成样本时的采样率越大。特别是，同时具有较高边界度和较高密度的样本将被赋予较大的权重，而具有较小边界度和较小密度的样本的权重将小得多。边界样本的合成不仅可以强化少数类的决策边界，而且可以降低安全区域中基于局部密度的样本的合成率，从而可以缓解后续模型在特定区域生成时潜在的过拟合问题。将第 i 个少数类样本归一化后的权重定义如下：

$$\omega_i = \frac{\omega_i}{\sum_i^n \omega_i} \tag{7}$$

最后，根据$G = \left|\mathrm{D_{maj}}\right| - \left|\mathrm{D_{min}}\right|$计算为获得平衡数据需要生成的合成少数样本的数量 G。各个少数类样本需要合成的样本数量定义为 g_i：

$$\mathrm{g}_i = \omega_i \times G \tag{8}$$

得到每个少数类样本的合成样本数量后，使用 SMOTE 线性插值的方式合成少数类样本。

BDOS 的整体算法流程如 Algorithm.1 所示。

Algorithm.1：BDOS

Input：Training Set $D = D_{maj} \cup D_{min}$,$k = 5$

Output：New Training Set $D_{balance}$;

Initialize $D_s = []$;
for x_i in D_{min}:
    Calculate BD_i using Eq.(3);
    Calculate DS_i using Eq.(5);
end for
for x_i in D_{min}:
    Calculate ω_i using Eq.(6);
end for
Normalize each ω_i using Eq.(7);
for x_i in D_{min}:
    Calculate g_i using Eq.(8);
    while $g_i \neq 0$:

Randomly choose a minority sample from k nearest neighbors of x_i;
Generate synthetic examples using Eq.(1);
$D_s \leftarrow D_s \cup x_{new}$;
$g_i = g_i - 1$;
end while
end for
$D_{balance} = D_{maj} \cup D_{min} \cup D_s$;

3.2 噪声过滤处理

已有文献表明[48]，类重叠是影响不平衡分类性能的重要因素。研究者们提出了很多清洗方法来清洗重叠区域的样本，使得决策边界更加明确，并且取得较好的效果。其中 IPF 能够更有效地过滤掉噪声样本以及过采样过程中引入的低质量样本得到了广泛的应用。由于 BDOS 不仅加强安全区域的少数类样本，其加强边界区域的少数类样本时仍存在较小引入低质量样本的风险，并且一些深入到少数类样本区域的多数类样本会大大影响到模型的分类性能。因此，本文选取 IPF 清洗由 BDOS 过采样后得到的平衡数据集 $D_{balance}$，使得决策边界更加清晰，从而提升模型的分类性能。

IPF(Iterative-Partitioning Filters) 是一种噪声过滤技术，它通过多次迭代去除噪声样本，直到满足终止条件[36]。IPF 的主要思想是通过投票从多个分类器中去除噪声实例。IPF 的整体算法流程如 Algorithm.2 所示。$D_{balance} = D_{maj} \cup D_{min}$为 IPF 的输入数据集。$n'$是分割$D_{maj}$和$D_{min}$所需的子集数。η是投票比率，用于确定样本是否为噪声。

Algorithm.2：IPF

Input： $D_{balance} = D_{maj} + D_{min}$, n' and η;
Output：New training set D_{new};

1. Set Missing Matrix $\mathbb{C}$ to a $\mathtt{m}\times 1$ zero matrix, where $\mathbb{C}$ is used to reserve the number of wrong judgments for every sample, $m = |D|$;
2. Split D_{maj} and D_{min} into n' subset, respectively. Then n' sub-minority sets $(D_{min1}, D_{min2}, \dots, D_{minn})$ and n' sub-majority sets $(D_{maj1}, D_{maj2}, \dots, D_{majn})$ can be obtained;
3. For $i = 1{:}\,n'$
4. Use Classification and Regression Tree (CART) to train classifier C_i based on D_{mini} and D_{maji};
5. Test the current trainset $(D_{min} \cup D_{maj})$ by using classifier i,and compare the result with the target. If the result of sample j is not equal to its target, then $\mathbb{C}(j,1)=\mathbb{C}(j,1)+1$;
6. End For
7. Delete the samples whose wrong judgments is greater than or equal to $n' * \eta$, and get the new positive set D'_{min} and new negative set D'_{maj}, $D_{balance} = D'_{maj} \cup D'_{min}$.

3.3 LightGBM 缺陷预测

在不平衡数据分类问题中，由于采样方法具有随机性，分类器可能产生偏差，已有研究将集成学习思想运用于采样中，以获得更高效的解决类不平衡问题的软件缺陷预测模型。

LightGBM[49]是微软于 2017 年提出一个新的 boosting 框架模型，和 XGBoost 一样是对 GDBT 的高效实现，具体算法流程如图 3 所示，分为 8 个步骤。LightGBM 是 XGBoost 的一个改进版本，针对四个方面进行了改进。首先，LightGBM 引入了基于梯度的单边采样算法(Gradient-based One-Side Sampling，GOSS)。GOSS 算法从减少样本的角度出发，排除大部分小梯度的样本，仅用剩余的样本计算信息增益，在减少数据量和保证精度上取得了很好的平衡。在训练过程中，梯度越大的样本受到的关注越多，对增益

的影响也越大。其次，LightGBM 使用直方图算法来识别最优分裂点，其基本思想是：先把连续的浮点特征离散化成 K 个整数，同时构造一个宽度为 K 的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了需要的统计量，然后根据直方图的离散值，遍历寻找最优的分裂点。直方图取代了传统的预排序，所以在某种意义上，它牺牲了准确性来换取速度。再者，LightGBM 通过互斥特征捆绑算法(Exclusive Feature Bundling，EFB)在一定程度上降低了特征维数。对于稀疏的特征空间(例如 onehot 编码)，通过将互斥的特征捆绑在一起，来减少特征数目。最后，LightGBM 采用了具有深度限制的叶子生长 leaf-wise 算法，而不是传统的按层生长 level-wise 算法，提高了精度并防止过拟合。

因集成算法 LightGBM 具有更快的训练速度、更低的内存使用、更好的准确率以及能处理大规模数据等优点，本文采用 LightGBM 模型训练经过 BDOS 和 IPF 预处理后的数据分布，形成最终的软件缺陷预测模型，整体框架如图 4 所示。

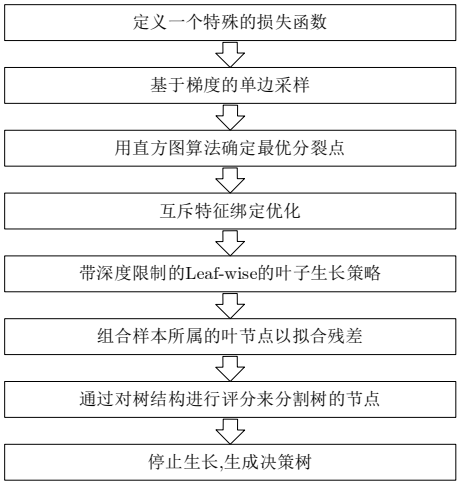


图 3 LightGBM 的算法流程

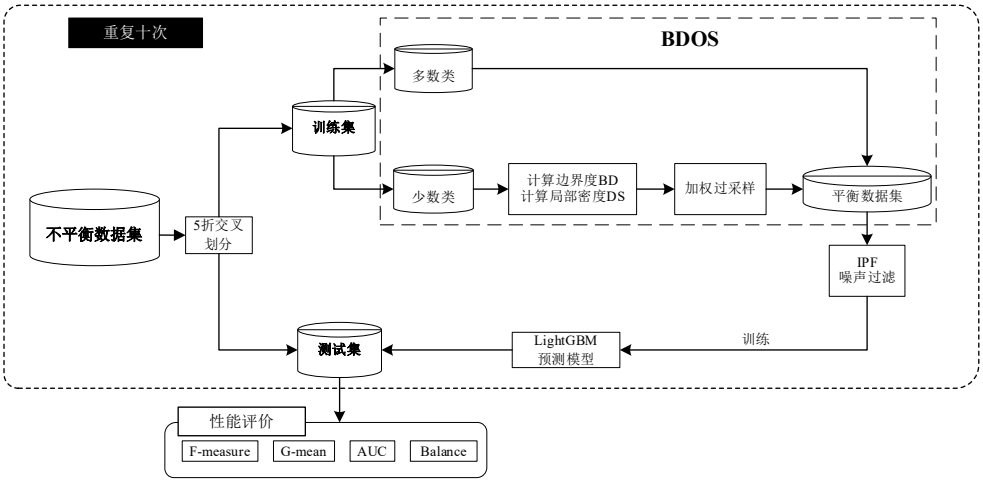


图 4 软件缺陷预测模型架构图

4 实验设计与分析

为了验证本文提出的 BDOS 算法的有效性，本文采用 PROMISE[50]数据库中的不平衡数据集进行实证研究。

4.1 数据集

本文从 PROMISE 数据库中选取了 11 个不平衡数据集进行了数值实验。表 1 展示了这 11 个数据集的详细信息，包括数据集的名称(Datasets)、样本总数(Modules)、有缺陷样本的数量(Minority)及缺陷率(Defects)。

表 1 数据集详细信息

Datasets	Modules	Minority	Defects(%)
ivy-2.0	352	40	11%
jedit-4.2	367	48	13%
ar6	101	15	15%
xalan-2.4	723	110	15%
xerces-1.3	453	69	15%
ant-1.3	125	20	16%
log4j-1.0	135	34	25%
zxing	399	118	30%
synapse-1.2	256	86	34%
velocity-1.6	229	78	34%
poi-2.5	385	137	36%

4.2 评价指标

软件缺陷预测模型的性能通常是通过分析混淆矩阵的计算结果来评估，混淆矩阵的定义如表 2 所示。其中 TP(True Positive)代表将实际是有缺陷的样本正确地预测为有缺陷的样本的数量，TN(True Negative)代表将实际是无缺陷的样本正确地预测为无缺陷的样本的数量，FP(False Positive)代表将实际是无缺陷的样本错误地预测为有缺陷的样本的数量，FN(False Negative)代表将实际是有缺陷的样本错误地预测为无缺陷的样本的数量。

表 2 混淆矩阵表格

True class	Hypothesis output	
	Minority class	Majority class
Minority class	TP	FN
Majority class	FP	TN

传统的分类算法一般以式(9)定义的准确率作为分类器性能的评价指标。

$$\mathrm{ACC}=\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}} \tag{9}$$

但是，在类不平衡情况下，无缺陷的实例的数量大大超过有缺陷实例的数量，即使所有实例都被预测为无缺陷而不考虑它们的实际属性，其值仍然很高，因此，ACC 不是一个合适的性能度量。为了评估分类模型的性能，本文采用软件缺陷预测中常用的四种性能指标,分别为 F-measure、G-mean、AUC 和 Balance，具体定义如下。

F-measure 是查全率(Recall)与查准率(Precision)的加权调和平均值，F-measure 的计算公式如式(10)所示。

$$\mathrm{F-measure}=\frac{(1+\beta^2)\times\mathrm{Precision}\times\mathrm{Recall}}{\beta^2\times\mathrm{Precision}\times\mathrm{Recall}} \tag{10}$$

其中：

$$\mathrm{Recall}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}=\mathrm{TPR} \tag{11}$$

$$\mathrm{Precision}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{12}$$

G-mean 是查全率(Recall)与查准率(Precision)的几何平均值，相关定义如下。

$$\mathrm{G-mean}=\sqrt{\mathrm{Recall}\times\mathrm{Precision}} \tag{13}$$

AUC 指 ROC 曲线下与坐标轴所围的面积。AUC 值越接近 1，表示预测模型性能越好。ROC 曲线是以查全率 TPR 为纵轴，误报率 FPR 为横轴的二维曲线。

在 ROC 曲线上，左上角坐标为(FPR=0,TPR=1)的点是最理想的分类状态，即在所有的缺陷都被发现

的同时没有产生误报，Balance 衡量的是模型预测性能在 ROC 曲线图中点(FPR,TPR)与点(FPR=0,TPR=1)的欧氏距离。Balance 的计算公式如式(14)所示。

$$\text{Balance} = 1 - \frac{\sqrt{(0-\text{FPR})^2 + (1-\text{TPR})^2}}{\sqrt{2}} \tag{14}$$

其中：

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{15}$$

4.3 实验设置

为了验证 BDOS 方法的有效性，选择了 10 个对比方法，其中包括：4 个经典的过采样方法，即 SMOTE[6]、Borderline-SMOTE(BSMOTE)[24]、ADASYN[26]和 MWMOTE[27]；2 个经典的用于软件缺陷预测的过采样方法，即 MAHAKIL[39]和 WACIL[41]；2 个经典的采样结合过滤的方法，即 SMOTE-IPF(SMO+IPF)[36]和 SMOTE-TomekLink(SMO+TL)[35]；2 个经典的基于集成的方法 RUSBoost(RUSB)[43]和 RAMOBoost(RAMOB)[44]。

实验中所涉及的近邻参数均按照文献设置为 5。此外，对于非集成学习方法，统一规定在过采样预处理后，采用 C4.5 决策树作为分类器对预处理之后的数据集进行训练和测试，参数设置都默认为原始参数。对于集成学习方法中的参数均设置为原论文推荐的默认值。为了保证实验结果的稳定性与有效性，采用 5 折交叉验证，将数据集的 80%作为训练集，20%作为测试集，训练过程重复 10 次，性能取 10 次 5 折交叉验证平均值。

4.4 合成样本的可视化对比

为了以更直观的方式说明数据层面的采样方法的差异，本文使用 sklearn 库中的 make_blobs 方法人工合成了一个数据分布，其中多数类样本数量为 400，少数类样本数量为 80。图 5 展示了非集成学习方法的可视化采样结果，其中图 5(b)展示的是 BDOS 方法在送至 lightgbm 训练之前的数据分布。

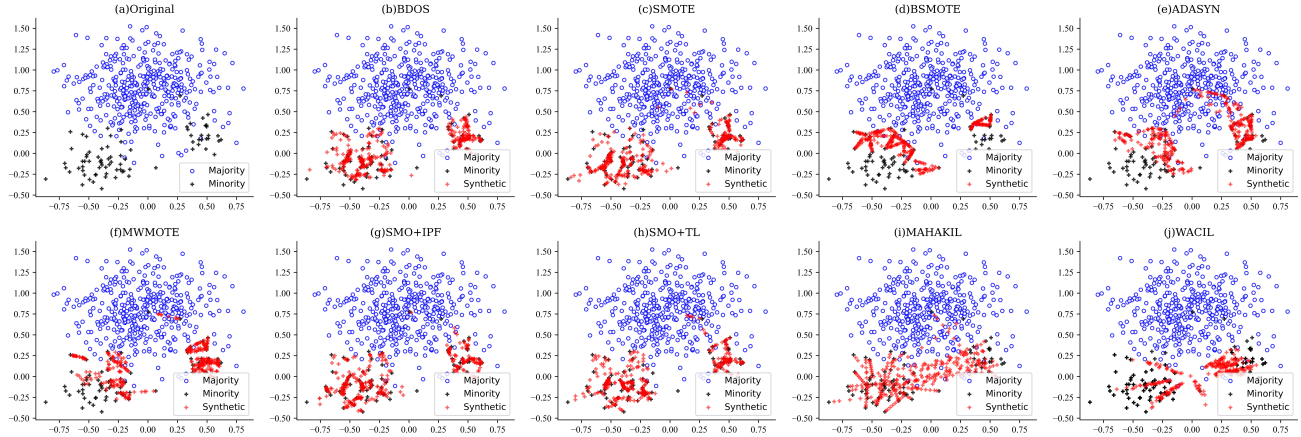


图 5 不同采样方法的合成数据分布图

为了更清晰的展现不同的数据层面方法对危险的少数类样本的敏感性，图 5(a)Original 图中可以发现在多数类密集区域有两个少数类样本。如图 5 所示，除了 BDOS，BSMOTE 和 WACIL 方法没有在多数类较为密集区域合成少数类样本外，其余方法均在多数类样本高密度区域引入了噪声样本，这无疑会加重数据分布的类重叠问题，在多数类密集区域合成少数类样本会模糊分类边界进而影响后续分类性能。然而 BSMOTE 的合成样本大量聚集在边界区域，大大提升了边界区域过拟合的风险，而 WACIL 方法在不存在少数类样本的区域中合成了大量样本，并且合成样本较为密集，容易引发过拟合的风险。

值得注意的是，BDOS 方法与 SMO+IPF 方法中均使用了 IPF 噪声滤波器，但是后者仍引入低质量合成样本。由此可以说明 BDOS 的加权方法是有效的，BDOS 既增强边界区域又增强安全区域的加权策略均会对危险的少数类样本赋予较小的权重，继而保证危险的少数类样本采样权重比较小，因此对于比较危险的少数类样本 BDOS 并不会选其作为种子进行合成，从而降低类重叠问题带来的风险。并且可以发现 BDOS 方法，SMOTE 方法与 SMO+IPF 方法的合成结果示意图比较相似，但其中 BDOS 方法明显合成的样

本分布较为松散，这同样得益于 BDOS 增强边界区域和安全区域的思想，从两个方面考虑少数类样本的采样权重避免了在某一特定区域合成过多，由图 5 可以发现 BDOS 在不加重类重叠问题的前提下，为少数类样本占据了更多的潜在空间，这不仅可以降低过拟合的风险，还能表明 BDOS 拥有较好的泛化能力，并且 BDOS 没有引入低质量样本。因此从整体来看 BDOS 方法取得了较优的合成数据分布图。

4.5 对比方法分组比较

为了更细致的展示 BDOS 与 10 种对比方法之间的性能优劣，对 11 种方法两两分组进行更详细的对比。图 6 展示了不同的方法在 11 个数据集上的详细性能对比。该图中单元格中的数字表示该单元格所在行代表的方法与该单元格所在列代表的方法相比，取得更好的实验效果的数据集的个数。以图 6(b)G-mean 中的第一行数据为例，11 个数据集中，BDOS 在 11 个数据集上的性能优于 SMOTE，BDOS 在 11 个数据集上的性能优于 BSMOTE，……，BDOS 在 9 个数据集上的性能优于 RAMOB。除了从单元格中的数值观察不同方法之间的优劣外，还可以通过颜色的深浅，颜色越深表示该单元格所在行的方法性能越好。由图 6 可以看出两两对比时，BDOS 在大多数数据集上均优于对比方法，从而更详细的展现出 BDOS 的优势。

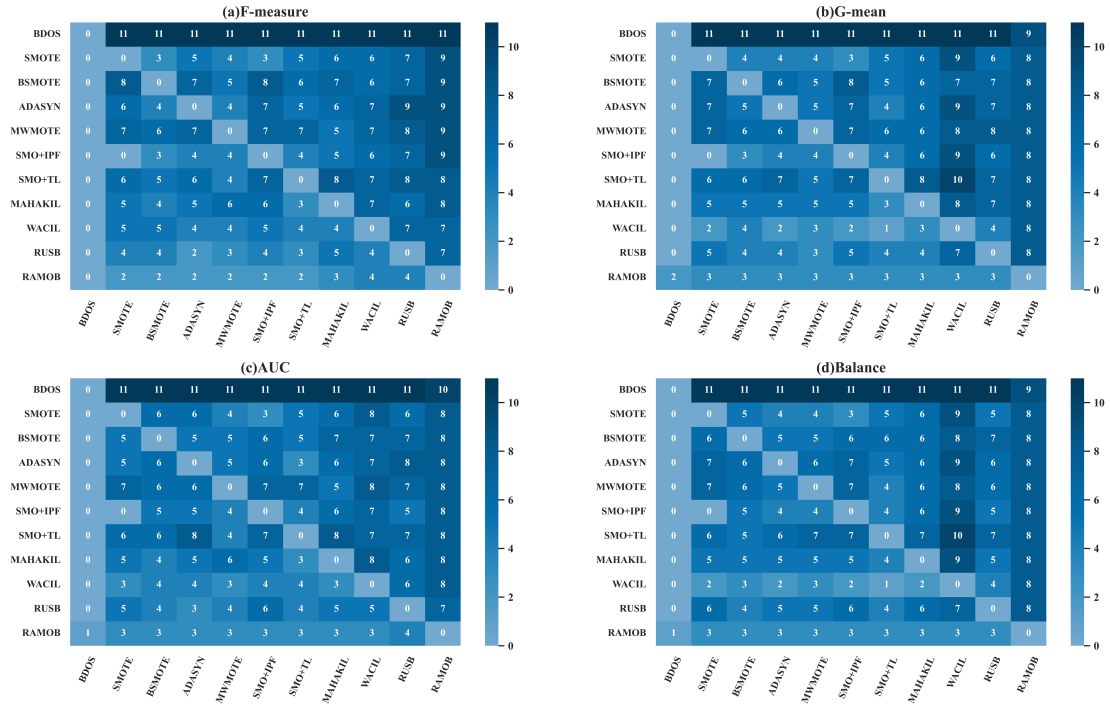


图 6 BDOS 与 10 种方法性能对比图

4.6 实验结果分析

在本小节中，将所提出的 BDOS 与 10 种经典的方法在 4 个评价指标上进行了比较。F-measure、G-mean、AUC 和 Balance 的结果见表 3-6，其中加粗表示为该值为所有方法中的最优值。

表 3 BDOS 与比较算法的 F-measure 值

	BDOS	SMOTE	BSMOTE	ADASYN	MWMOTE	SMO+IPF	SMO+TL	MAHAKIL	WACIL	RUSB	RAMOB
ant-1.3	**0.4442**	0.3061	0.3822	0.3204	0.3505	0.3061	0.2923	0.2837	0.2715	0.2439	0.3014
ar6	**0.4299**	0.3032	0.3100	0.3566	0.3430	0.3032	0.2859	0.2324	0.2370	0.3407	0.2574
ivy-2.0	**0.4132**	0.3122	0.2816	0.3135	0.2591	0.3122	0.3100	0.3062	0.3130	0.3094	0.2860
jedit-4.2	**0.5157**	0.4100	0.3696	0.3643	0.3844	0.4100	0.3737	0.3925	0.3894	0.3179	0.3622
log4j-1.0	**0.5609**	0.4985	0.5020	0.4962	0.5312	0.4985	0.5259	0.4570	0.4415	0.4906	0.4402
poi-2.5	**0.7552**	0.6719	0.6989	0.6875	0.7005	0.6654	0.6855	0.6748	0.6799	0.6761	0.5341
synapse-1.2	**0.6492**	0.5459	0.5583	0.5524	0.5527	0.5459	0.5494	0.5796	0.5785	0.5511	0.5241
velocity-1.6	**0.6392**	0.5474	0.5520	0.5528	0.5465	0.5474	0.5537	0.5508	0.5330	0.5909	0.5214
xalan-2.4	**0.4656**	0.3498	0.3443	0.3471	0.3763	0.3498	0.3810	0.3804	0.3603	0.3472	0.3828

xerces-1.3	**0.5133**	0.4291	0.4323	0.4074	0.3858	0.4181	0.4279	0.4254	0.4394	0.3889	0.3095
zxing	**0.5087**	0.4538	0.4667	0.4526	0.4927	0.4448	0.4582	0.4633	0.4127	0.4413	0.4582
Avg	**0.5359**	0.4389	0.4453	0.4410	0.4475	0.4365	0.4403	0.4315	0.4233	0.4271	0.3979

表 4　BDOS 与比较算法的 G-mean 值

	BDOS	SMOTE	BSMOTE	ADASYN	MWMOTE	SMO+IPF	SMO+TL	MAHAKIL	WACIL	RUSB	RAMOB
ant-1.3	**0.6194**	0.4850	0.5608	0.4996	0.5283	0.4850	0.4740	0.4605	0.4102	0.4146	0.3692
ar6	**0.5861**	0.4401	0.4925	0.5402	0.5020	0.4401	0.4182	0.3778	0.3512	0.4965	0.2405
ivy-2.0	0.6177	0.5720	0.5000	0.5611	0.4907	0.5720	0.5669	0.5465	0.5425	0.5670	**0.6237**
jedit-4.2	**0.7127**	0.6442	0.5824	0.6126	0.6132	0.6442	0.6178	0.6232	0.6060	0.5620	0.6885
log4j-1.0	**0.6947**	0.6406	0.6416	0.6412	0.6617	0.6406	0.6604	0.6045	0.5828	0.6363	0.4015
poi-2.5	**0.8095**	0.7410	0.7630	0.7537	0.7657	0.7364	0.7532	0.7444	0.7479	0.7430	0.2159
synapse-1.2	**0.7277**	0.6430	0.6560	0.6505	0.6502	0.6430	0.6489	0.6721	0.6731	0.6496	0.2833
velocity-1.6	**0.7201**	0.6437	0.6478	0.6484	0.6433	0.6437	0.6500	0.6480	0.6319	0.6798	0.2780
xalan-2.4	0.6567	0.5921	0.5700	0.5844	0.6076	0.5921	0.6132	0.6014	0.5731	0.5984	**0.6682**
xerces-1.3	**0.6916**	0.6445	0.6371	0.6242	0.5995	0.6351	0.6393	0.6275	0.6338	0.6256	0.4502
zxing	**0.6348**	0.5898	0.6019	0.5906	0.6226	0.5825	0.5945	0.5997	0.5564	0.5799	0.0694
Avg	**0.6792**	0.6033	0.6048	0.6097	0.6077	0.6013	0.6033	0.5914	0.5735	0.5957	0.3899

表 5　BDOS 与比较算法的 AUC 值

	BDOS	SMOTE	BSMOTE	ADASYN	MWMOTE	SMO+IPF	SMO+TL	MAHAKIL	WACIL	RUSB	RAMOB
ant-1.3	**0.6850**	0.5838	0.6374	0.6024	0.6217	0.5838	0.5767	0.5855	0.5826	0.5574	0.5590
ar6	**0.6733**	0.6045	0.5998	0.6314	0.6069	0.6045	0.6036	0.5610	0.5585	0.6196	0.5067
ivy-2.0	**0.6758**	0.6295	0.6003	0.6221	0.5907	0.6295	0.6280	0.6196	0.6174	0.6287	0.6639
jedit-4.2	**0.7372**	0.6778	0.6452	0.6511	0.6582	0.6778	0.6544	0.6681	0.6548	0.6145	0.7203
log4j-1.0	**0.7153**	0.6666	0.6673	0.6683	0.6890	0.6666	0.6826	0.6422	0.6307	0.6642	0.5740
poi-2.5	**0.8116**	0.7475	0.7668	0.7569	0.7670	0.7424	0.7573	0.7481	0.7530	0.7487	0.5231
synapse-1.2	**0.7382**	0.6549	0.6631	0.6614	0.6602	0.6549	0.6565	0.6802	0.6810	0.6584	0.5418
velocity-1.6	**0.7270**	0.6596	0.6565	0.6578	0.6549	0.6596	0.6605	0.6560	0.6470	0.6866	0.5328
xalan-2.4	0.6899	0.6240	0.6197	0.6225	0.6416	0.6240	0.6448	0.6421	0.6262	0.6252	**0.6962**
xerces-1.3	**0.7181**	0.6774	0.6752	0.6596	0.6465	0.6699	0.6726	0.6651	0.6746	0.6468	0.5898
zxing	**0.6496**	0.6055	0.6139	0.6035	0.6364	0.5963	0.6136	0.6125	0.5809	0.5966	0.5048
Avg	**0.7110**	0.6483	0.6496	0.6488	0.6521	0.6463	0.6501	0.6437	0.6370	0.6406	0.5829

表 6　BDOS 与比较算法的 Balance 值

	BDOS	SMOTE	BSMOTE	ADASYN	MWMOTE	SMO+IPF	SMO+TL	MAHAKIL	WACIL	RUSB	RAMOB
ant-1.3	**0.6193**	0.5222	0.5676	0.5366	0.5607	0.5222	0.5114	0.5101	0.4939	0.4780	0.4002
ar6	**0.6015**	0.5144	0.5222	0.5580	0.5246	0.5144	0.5039	0.4688	0.4515	0.5528	0.3447
ivy-2.0	**0.5949**	0.5620	0.5119	0.5467	0.5093	0.5620	0.5586	0.5463	0.5290	0.5681	**0.5949**
jedit-4.2	**0.6874**	0.6264	0.5694	0.5996	0.5957	0.6264	0.5997	0.6071	0.5834	0.5530	0.6512
log4j-1.0	**0.6794**	0.6320	0.6291	0.6335	0.6465	0.6320	0.6484	0.5984	0.5817	0.6233	0.4410
poi-2.5	**0.8036**	0.7321	0.7561	0.7477	0.7624	0.7288	0.7465	0.7388	0.7410	0.7365	0.3416
synapse-1.2	**0.7151**	0.6345	0.6512	0.6444	0.6430	0.6345	0.6434	0.6664	0.6659	0.6460	0.3584
velocity-1.6	**0.7108**	0.6396	0.6426	0.6429	0.6362	0.6396	0.6425	0.6435	0.6249	0.6749	0.3597
xalan-2.4	0.6286	0.5806	0.5573	0.5716	0.5934	0.5806	0.6001	0.5843	0.5548	0.5901	**0.6379**
xerces-1.3	**0.6629**	0.6300	0.6200	0.6069	0.5838	0.6204	0.6209	0.6048	0.6101	0.6125	0.4605
zxing	**0.6248**	0.5843	0.5970	0.5863	0.6160	0.5782	0.5890	0.5950	0.5521	0.5763	0.3044
Avg	**0.6662**	0.6053	0.6022	0.6067	0.6065	0.6036	0.6059	0.5967	0.5808	0.6010	0.4450

总体来看，从表 3-6 中可以看出，11 个数据集中，BDOS 在 F-measure，AUC，G-mean 与 Balance 四

个指标上的性能均值均是最优，并且 BDOS 在各个指标中，在大多数数据集上均取得了最优结果，具体分析如下。

从表 3 中可以看到 BDOS 在 11 个数据集上的 F-measure 值均取得最优，并且 BDOS 在 F-measure 指标上的平均性能为 0.5359，比排名第二的 MWMOTE 方法高出 19.75%，可以发现 BDOS 方法在 F-measure 指标上具有较大的优势。

从表 4 中可以发现 BDOS 在 G-mean 指标上有 9 个数据集上的排名均为第一，在 ivy-2.0，xalan-2.4 两个数据集上均取得排名第二的结果。BDOS 的 G-mean 指标均值与排名第二的 ADASYN 的 G-mean 指标均值相比，性能提高了 11.4%。

如表 5 所示，BDOS 在 10 个数据集上的 AUC 指标均排名第一，仅在 xalan-2.4 数据集上 AUC 指标排名第二。BDOS 的 AUC 均值为 0.7110 比排名第二的 MWMOTE 方法相比性能提高了 9.03%。

如表 6 所示，BDOS 的 Balance 指标中，在 xalan-2.4 数据集上排名第二，在其余 10 个数据集上均取得了最优的结果。可以发现在 xalan-2.4 数据集上 BDOS 的 Balance 值为 0.6286，与排名第一的 RAMOBoost 方法的 Balance 值 0.6379 差距较小。整体来看 BDOS 的 Balance 均值为 0.6662，与排名第二的 ADASYN 方法相比性能提高了 9.8%。

综合表 3-表 6 的性能均值，可以发现 SMOTE、BSMOTE、ADASYN、MWMOTE、SMO+IPF 和 SMO+TL 在 F-measure、G-mean、AUC 和 Balance 指标上的 11 个数据集的性能均值均相差不大，但在 ant1.3、ar6、ivy-2.0 等数据集上却有较大的差别，这一现象表明这些方法均能在某些特定的数据集上取得较好的结果，进一步说明增强边界区域的样本以及有针对性的选择种子样本的均是有效的，而 BDOS 在绝大部分数据集上均可以获得较好的效果，从实验结果方面可以发现 BDOS 同时增强边界区域和安全区域的少数类样本的策略是有效的，同时可以表明以边界度和局部密度对少数类样本进行加权过采样方法能够有效提升 BDOS 的泛化能力。BDOS 方法在 4 个指标中均取得最佳性能，并且与排名第二方法相比均提高了 10%左右的性能，尤其在 F-measure 指标上的提升效果最为明显。因此由表 3-6 可以发现，BDOS 方法与其余 10 种对比方法相比具有较大的优势。

4.7 统计假设检验

为了比较 BDOS 算法和 10 个对比方法的显著性差异，表 7 展示了 BDOS 和其余方法进行 t 检验时的 p 值统计量结果。在显著性水平$\alpha = 0.05$的水平下，从表中可以看出 BDOS 算法与其他 10 个方法在 F-measure、AUC、G-mean 和 balance 四个指标上的 p 值结果均小于 0.05。由此可以得出结论：在 0.05 的水平上应该拒绝原假设，即这 11 种方法的实验效果有显著性差异。由此可以说明 BDOS 显著性的比其余方法好。

表 7　BDOS 与对比方法在 t 检验下的 p 值对照表

	SMOTE	BSMOTE	ADASYN	MWMOTE	SMO+IPF	SMO+TL	MAHAKIL	WACIL	RUSB	RAMOB
F-measure	1.69216E-07	5.78447E-06	7.51614E-07	3.78872E-05	5.77443E-08	8.14087E-06	1.06801E-05	2.53942E-06	2.80909E-05	3.17646E-06
AUC	4.43085E-07	5.32693E-07	9.33616E-08	7.20779E-06	1.43022E-07	5.54662E-06	1.37783E-06	4.17553E-07	1.21789E-05	0.000804535
G-mean	2.53508E-05	9.88627E-06	1.64049E-06	3.31603E-05	1.42653E-05	0.000191455	0.000197583	0.00017805	0.000245174	0.001292888
Balancce	3.76432E-06	5.59802E-06	2.33794E-07	2.09583E-05	1.60305E-06	3.37365E-05	1.71423E-05	3.99517E-06	0.000204925	0.000811094

5 结语

本文提出了一种改进过采样的不平衡数据缺陷预测模型。在数据预处理阶段针对缺陷数据类不平衡问题提出一种基于边界度和局部密度的加权过采样方法，同时增强边界区域和安全区域，考虑利用尽可能多的少数类实例进行样本合成，避免仅在特定区域合成少数类，从而缓解下游模型学习可能出现的过拟合问题。对于局部密度和数据分布不确定性较大的少数实例，BDOS 会加大其采样权重。在模型构建阶段，与集成算法 LightGBM 结合形成最终的软件缺陷预测模型。BDOS 与 10 个经典方法的对比实验验证了该方法的有效性。

BDOS 是基于软件缺陷场景下的二分类问题，未来可以拓展至其他领域下的多分类问题，多分类的分布情况更加复杂，如何改进方法以应用于多分类任务是一个研究方向。由于在许多现实缺陷预测数据集中，类别分布不平衡和标记样本数量过少的情况经常同时存在，因此，采样方法结合半监督学习以解决缺陷数据标记样本不足的问题也是今后的一个研究方向。

参 考 文 献

[1] Folino G,Pisani F S,Sabatino P.An Incremental Ensemble Evolved by using Genetic Programming to Efficiently Detect Drifts in Cyber Security Datasets:[C]// On Genetic & Evolutionary Computation Conference Companion. ACM, 2016:1103-1110.

[2] Vasighizaker A,Jalili S.C-PUGP:A cluster-based positive unlabeled learning method for disease gene prediction and prioritization-Scien ceDirect[J]. Computational Biology and Chemistry, 2018, 76:23-31.

[3] Jurgovsky J, Granitzer M, Ziegler K, et al.Sequence Classification for Credit-Card Fraud Detection[J].Expert Systems with Application s, 2018, 100(JUN.):234-245.

[4] KIM J H. Time frequency image and artificial neural network based classification of impact noise for machine fault diagnosis[J]. Intern ational journal of precision engineering and manufacturing,2018,19(6): 821–827.

[5] D Rodríguez, Herraiz I, Harrison R, et al. Preliminary comparison of techniques for dealing with imbalance in software defect predictio n.[C]// International Conference on Evaluation & Assessment in Software Engineering. ACM,2014.

[6] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of Artificial Intel ligence Research16 (2002) 321–357.

[7] Hui H, Wang W, Mao B. Borderline-SMOTE: A New Over-sampling Method in Imbalanced Data Sets Learning[C]// Advances in Intel ligent Computing, International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I. 2 005.

[8] SáezJosé A, LuengoJulián, StefanowskiJerzy, et al. SMOTE-IPF[J]. Information SciencesInformatics and Computer Science, Intelligent Systems, Applications: An International Journal, 2015.

[9] G Bin, VS Sheng, KY Tay,et al. Cross Validation Through Two-Dimensional Solution Surface for Cost-Sensitive SVM[J]. IEEE Transac tions on Pattern Analysis and Machine Intelligence, 2017.

[10] Liu Y, Lu H, Yan K, et al. Applying Cost-Sensitive Extreme Learning Machine and Dissimilarity Integration to Gene Expression Data Classification[J]. Computational Intelligence & Neuroscience, 2016, 2016:8056253.

[11] Tapkan P, Ozbakir L ,Kulluk S , et al. A cost-sensitive classification algorithm: BEE-Miner[J]. Knowledge-Based Systems, 2016, 95(M ar.1):99-113.

[12] Fernandez-Baldera, Antonio, Buenaposada, et al. BAdaCost: Multi-class Boosting with Costs[J]. Pattern Recognition: The Journal of the Pattern Recognition Society, 2018, 79:467-479.

[13] Radtke P ,Granger E ,Sabourin R ,et al. Skew-Sensitive Boolean Combination for Adaptive Ensembles – An Application to Face Reco gnition in Video Surveillance[J]. Information Fusion, 2014, 20:31-48.

[14] Bhardwaj M ,Sharma K ,Sharma K .Cost-effectiveness of classification ensembles[J]. Pattern Recognition, 2016, 57:84-96.

[15] SUN Z, SONG Q, ZHU X, et al. A novel ensemble method for classifying imbalanced data[J/OL]. Pattern Recognition, 2014: 1623-163 7. http://dx.doi.org/10.1016/j.patcog.2014.11.014. DOI:10.1016/j.patcog.2014.11.014.

[16] Diez-Pastor, Jose, F, et al. Diversity techniques improve the performance of the best imbalance learning ensembles[J]. Information Sci ences An International Journal, 2015.

[17] M.Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F.Herrera,"A review on ensembles for the class imbalance problem: Bagging -,boosting-, and hybrid-based approaches," IEEE Trans. Syst. ManCybern. Syst. Part C, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[18] H. Cao, X. L. Li, D. Y. K. Woon, and S. K. Ng, "Integrated oversampling for imbalanced time series classification," IEEE Trans. Kno wl. Data Eng., vol. 25, no. 12, pp. 2809–2822,Dec. 2013.

[19] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," IEEE Trans. Knowl. Data Eng., vol. 28. no. 12,p p. 3353–3366, Dec. 2016.

[20] S. Barua, M. M. Islam, X. Yao, and K.Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," IEEE Trans. Knowl. Data Eng.,vol.26, no.2,pp. 405–425,Feb. 2014.

[21] S. Wang, L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," IEEE Trans. Knowl.Data Eng., vol. 27, no. 5, pp. 1356–1368, May 2015.

[22] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of datastreams with gradually evolved classes," IEEE Trans. Knowl. Data Eng., vol. 28, no. 6, pp. 1532–1545, Jun. 2016.

[23] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," IEEE Tran s.Knowl. Data Eng., vol. 25, no. 1, pp. 206–219,Jan. 2013.

[24] Han H,Wang W Y,Mao B H.Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning[J]. Lecture Notes i n Computer Science, 2005.

[25] Bunkhumpornpat C,Sinapiromsaran K,Lursinsap C.Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem[C]// Pacific-asia Conference on Advances in Knowledge Discovery & Data Mining. Springer-Verla g, 2009.

[26] He H,Yang B,Garcia E A,et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning[J]. IEEE, 2008.

[27] Barua S,Islam M M,Yao X,et al. MWMOTE--Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning [J]. IEEE Transactions on Knowledge &Data Engineering, 2013, 26(2):405-425.

[28] 陈俊丰,郑中团.WKMeans 与 SMOTE 结合的不平衡数据过采样方法[J].计算机工程与应用,2021,57(23):106-112.

[29] 杨赛华,周从华,蒋跃明等.一种改进的不平衡数据过采样算法 BN-SMOTE[J].计算机与数字工程,2020,48(09):2108-2113.

[30] Maldonado S, Vairetti C, Fernandez A, et al. FW-SMOTE: A feature-weighted oversampling approach for imbalanced classification[J]. Pattern Recognition, 2022, 124: 108511.

[31] Li J, Zhu Q, Wu Q, et al. A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbors[J]. In formation Sciences, 2021, 565: 438-455.

[32] Islam A, Belhaouari S B, Rehman A U, et al. KNNOR: An oversampling technique for imbalanced datasets[J]. Applied Soft Computin g, 2022, 115: 108288.

[33] Tao, X., Li, Q., Guo, W., Ren, C., He, Q., Liu, R., & Zou, J. (2020). Adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristicfiltering. Information Sciences, 519, 43–73.

[34] Cheng K , Zhang C , Yu H , et al. Grouped SMOTE With Noise Filtering Mechanism for Classifying Imbalanced Data[J]. IEEE Acc ess, 2019, 7:170668-170681.

[35] Batista G E, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data[J]. AC M SIGKDD explorations newsletter, 2004,6(1): 20-29.

[36] Sáez, J. A., Luengo, J., Stefanowski, J., & Herrera, F. (2015). SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Information Sciences, 291, 184–203.

[37] Guan H, Zhang Y, Xian M, et al. SMOTE-WENN: Solving class imbalance and small sample problems by oversampling and distance scaling[J]. Applied Intelligence, 2021, 51: 1394-1409.
[38] Xu Z, Shen D, Nie T, et al. A cluster-based oversampling algorithm combining SMOTE and k-means for imbalanced medical data[J]. Information Sciences, 2021, 572: 574-589.
[39] K.E. Bennin, J. Keung, P. Phannachitta, A. Monden, S. Mensah, Mahakil:Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction, IEEE Trans. Softw. Eng. 44 (6) (2017)534–550.
[40] S. Feng, J. Keung, X. Yu, Y. Xiao, K.E. Bennin, M.A. Kabir, M. Zhang, COSTE:Complexity-based OverSampling technique to allevia te the class imbalance problem in software defect prediction, Inf. Softw. Technol. 129 (2021)106432.
[41] Manchala P,Bisi M.Diversity based imbalance learning approach for software fault prediction using machine learning models[J]. Applie d Soft Computing, 2022:124.
[42] Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2008). Building usefulmodels from imbalanced data with sampling and boosting. In Proceedings of the 21th international Florida artificial intelligence research society conference (pp. 306–311).AAAI.
[43] Seiffert C, Khoshgoftaar T M, Van Hulse J, et al. RUSBoost: A hybrid approach to alleviating class imbalance[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2009, 40(1): 185-197.Chen S,He H,Garcia E A.RAMOBoost:ranked mino rity oversampling in boosting[J].IEEE Transactions on Neural Networks,2010,21(10):1624-1642.
[44] Sheng Chen, Haibo He, and Edwardo A Garcia. 2010. RAMOBoost: ranked minority oversampling in boosting. IEEE Transactions on Neural Networks 21, 10 (2010), 1624–1642.
[45] 简艺恒,余啸.基于数据过采样和集成学习的软件缺陷数目预测方法[J].计算机应用,2018,38(09):2637-2643+2659.
[46] Mostafaei S, Ahmadi A, Shahrabi J. USWAVG-BS: Under-Sampled Weighted AVeraGed BorderlineSMOTE to handle data intrinsic di fficulties[J]. Expert Systems with Applications, 2023:120379.
[47] 钱晨. 基于遥测数据的卫星姿态控制系统故障检测方法 [D]. 杭州: 浙江大学, 2010: 30-32.(Qian C. Satellite attitude control system fault detection method based on telemetry data[D]. Hangzhou: Zhejiang University, 2021: 30-32.)
[48] Stefanowski, Jerzy. "Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data." Emerging parad igms in machine learning (2013): 277-306.
[49] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[J]. Advances in neural information process ing systems, 2017, 30.
[50] Menzies T, Caglayan B, Kocaguneli E, Krall J, Peters F, Turhan B. The promise repository of empirical software engineering data[J]. West Virginia University, Department of Computer Science. 2012.

A Novel High Throughput RO-Based True Random Number Generation on FPGA

Junjun Wang, Jinhui Liu, Zhao Huang* and Quan Wang, *Member, IEEE*

***Abstract*—True random numbers are generated by true random number generator (TRNG), which are important cryptographic primitives for random secret key generation. In this brief, we propose a high throughput low resource overhead TRNG based on ring oscillator (RO): NAND-XOR gates ring oscillator (NXRO). By replacing NAND gates with XOR gates in traditional NAND RO, NXRO solves the problem that only even-stages inverters are available in NAND ROs. In addition, we use the output of self-feedback NAND gate as the enable signals to control the oscillation states of XOR RO, which leads to different oscillation states. These ROs influence each other and resulting in more random jitter, which improves the quality of the entropy source and improves the throughput rate of TRNG. We implement NXRO on Xilinx Spartan-6 XC6SLX16 FPGA and Xilinx Atrix-7 XC7A35T FPGA, respectively. The experimental results show that our proposed NXRO only using 11 slice achieves 350Mbps and 575Mbps throughput rates, respectively. It also passed the NIST 800-22, NIST 800-90B, and AIS-31 tests. Compared with the existing RO-based TRNGs, our proposed NXRO achieve high throughput rate and low hardware resource overhead.**

***Index Terms*—TRNG, FPGA, XOR, Ring oscillator.**

I. Introduction

True Random numbers is an important primitives for random secret key generation, which can generate unpredictable random bits. It is an important part to improve the security of devices. True random number generators (TRNG) extract noise sources from physical processes to generate true random numbers, which are unpredictable and irreducible. A complete TRNG includes entropy sources and post-processing units. The circuit structure that converts an uncertain noise source into random numbers in a circuit is called the entropy source. The entropy source with very low resource overhead and power consumption to achieve high throughput rate is recognized as an excellent TRNG. Ideally, the probability of both 0 and 1 occurring in a set of consecutive random sequences should be 50%. However, the output of the TRNG may be biased by factors such as voltage, temperature, noise etc. [1], the output of the TRNG may be biased and deviate from the ideal random sequence, which can be exploited by attackers.

Programmable gate arrays (FPGAs) are an important platform for implementing TRNG, the traditional entropy sources based on FPGA are ring oscillators (RO) [2],[5],[6],[9], self-timed rings (STR) [5], digital clock management units (DCM) [11], Metastability [4], etc. Among these entropy sources, STR has good robustness but also has problems such as operational complexity; DCM is a clock management module embedded in the FPGA, which is easy to handle but also has additional resource overhead. And Metastability requires difficult manual wiring to generate enough randomness. RO uses phase noise and clock jitter of oscillators as entropy sources, with low power consumption and few hardware elements (usually only basic gate circuits are needed). However, Some RO-based TRNG circuits have high design cost, low throughput rate and occupy a lot of hardware resources. Therefore, our design target is propose a high throughput low resource overhead TRNG based on RO.

In this brief, we propose a lightweight and high throughput TRNG structure, namely, NAND-XOR ring oscillators (NXRO). Our NXRO modify the traditional RO structure, the new RO structure consists of a NAND gate and a RO consisting of a XOR gate and an inverter (we call it XOR RO), and fusing the all outputs of ROs with XOR tree. This not only improves the quality of the TRNG entropy source without post-processing, but also achieves a very high throughput rate with very low resource overhead and power consumption. In particular, the main contributions of this brief are as follows.

- We improve the structure of the existing RO and propose a lightweight, resource efficient and high throughput TRNG named NXRO.
- By using the output of the self-feedback NAND gate as the enable signal of the XOR RO to control its oscillation state, our proposed RO structure not only outputs two random signals simultaneously, but also improves the quality of the TRNG entropy source.
- The experimental results show that our proposed NXRO achieves the highest throughput rate, the lowest resource overhead and operational power consumption, performs well in process, voltage and temperature (PVT) variation experiments.

The rest of this brief is organized as follows: Section II introduces the related research. Section III provides the proposed NXRO. Section IV analyzes the experimental results, and finally we conclude the brief in Section V.

II. Related Research

A. RO clock jitter

[10] gives the relationship of RO stage and phase noise. The phase noise in the frequency domain and the average jitter in the time domain are related as shown in Equation 1. K is the Boltzmann constant, T is the absolute temperature, η, V_{DD}, V, I and R are constants, Δf is the offset frequency, f_0 is the

Junjun Wang, Jinhui Liu, Zhao Huang* and Quan Wang are with School of Computer Science and Technology, Xidian University, Xi'an, China (e-mail: junjunwang@stu.xidian.edu.cn; jhliu@mail.xidian.edu.cn; z_huang@xidian.edu.cn; qwang@xidian.edu.cn; *(*Corresponding Author))*

frequency of the RO, P is the power consumption of the ring oscillator and N is the RO stage. Increasing the stage N will decrease the frequency f_0, which optimizes the phase noise and reduces the throughput.

$$L_{min}\{\Delta f\} = \frac{8NKT}{3\eta P}\left(\frac{V_{DD}}{V} + \frac{V_{DD}}{IR}\right)\left(\frac{f_0}{f}\right)^2 \quad (1)$$

Equation 2 shows that the average jitter σ_{RMS} is positively correlated with the phase noise and negatively correlated with the signal frequency f_0. When the signal frequency f_0 increases, the average jitter σ_{RMS} becomes smaller, the clock jitter is optimized, and the entropy source randomness become worse.

$$\sigma_{RMS} = \frac{\sqrt{2\int_{f_1}^{f_2} L\{\Delta f\}\, df}}{2\pi f_0} \quad (2)$$

B. D type flip-flop Metastability

The D type flip-flop (DFF) has specified setup time and hold time. As shown in Figure 1(a), when the input signal $Input$ violates the timing requirement, it will trigger a metastability state. Therefore, the DFF can be used in TRNG for sampling random signals. [8] proves that increase the number of DFFs can increase the quality of the entropy source.

C. XOR gate glitches

As shown in Figure 1(b), when the time difference between input X and Y does not match the response time of the XOR gate, the XOR gate will output approximately random short pulses (a.k.a. glitches). [13] has experimentally demonstrated the narrow jump suppression effect of the XOR gate, which means that the XOR gate produces glitches when dealing with fast transitions in RO.

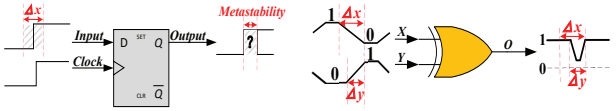


Fig. 1: (a) DFF Metastability, (b) XOR gate glitches.

III. XOR-based Ring Oscillator True Random Number Generator

Our proposed NXRO architecture consists of a TRNG circuit, a Control Unit, a First Input First Output (FIFO), and a phase-locked loop (PLL). The Control Unit output signal En controls the oscillation state of the ROs. The outputs random bits of the TRNG are transmitted through the FIFO and then through the USB-UART to the host computer. The TRNG circuit consists of ROs, DFFs and a XOR gate tree. The RO consist of a self-feedback NAND gate and a RO consisting of a XOR gate and an inverter; The DFFs sample all the outputs of the ROs circuit. The high-speed clock $Clock$ from the PLL is used as the sampling signal for each DFF. The output of the $RO1 \sim ROn$ circuit is sampled by the DFFs and connected to a XOR gate tree, and the output of the XOR gate is also

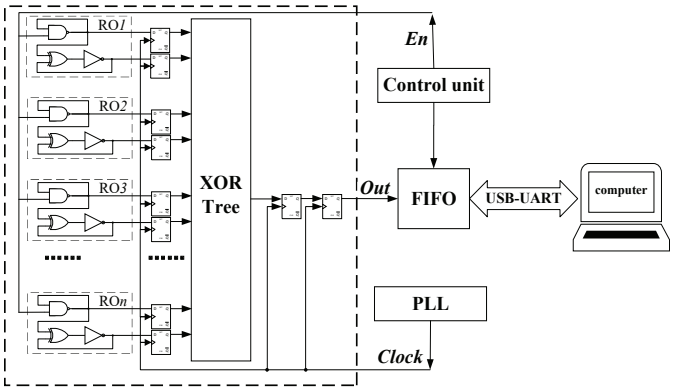


Fig. 2: The proposed TRNG design.

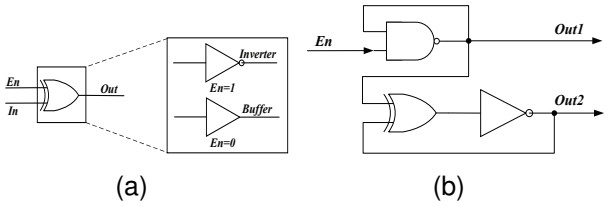


Fig. 3: (a) The XOR gate, and the modified RO (b).

sampled by the double-serial DFFs and output as a TRNG for one random bit. The circuit of NXRO is shown in Figure 2.

The NAND RO is only oscillated by the even-stage inverter. We replace the NAND gate with the XOR gate, and the inverter in RO is no longer limited to even-stage, and the odd-stage inverter is also able to oscillate RO. As shown in Figure 3(a), the XOR gate is functionally equivalent to an inverter when the enable signal En is 1, otherwise, it is a buffer. As shown in Figure 3(b), the self-feedback NAND gate starts to oscillate, and the output of the NAND gate is not only used as an enable signal to control the oscillation state of the XOR RO, but also as a random output signal. Similarly, the output of XOR RO is also output to DFF as a random signal.

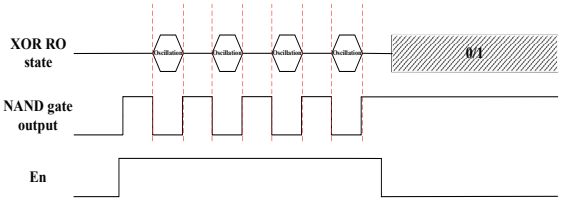


Fig. 4: The RO state transition waveform.

As shown in Figure 4, when En is 1, Self feedback NAND gate began to oscillate, when the output of the NAND gate is 0, the XOR gate is configured as buffer, which means the XOR RO has a 1-stage inverter and the XOR RO starts oscillating. Otherwise, the XOR gate is configured as an inverter, which means increasing the stages of the XOR RO, that is, it has two inverters, and the XOR RO stops oscillating. When $En = 0$, the self-feedback NAND stops oscillating first, and the output of NAND is constant to 1 at this time, which means the enable signal of XOR RO is also 1. At this time, XOR RO stops oscillating and then the whole TRNG stop output. We use ROs instead of external quartz oscillators to control the oscillation

state of XOR RO, which not only significantly r educes the effect of the determined global jitter source in the RO, but also increases the randomness of the TRNG. In addition, during the input signal high-speed variation, the XOR gate generate glitches, and the Metastability generated by the DFF time violation. All of these can affect the output of the RO and increase the randomness of the entropy source.

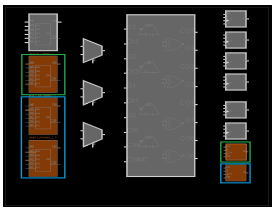


Fig. 5: The NXRO layout.

To reduce the Slice overhead, we lay out the LUTs in the ROs and the corresponding sampled DFFs in the same Slice, and the ROs are placed sequentially among each other. As shown in Figure 5, the orange part shows the LUTs and DFFs in the placed ROs, and the green frame shows the NAND gates and the corresponding sampling DFFs; the blue frame shows the XOR ROs and the corresponding sampling DFFs, such placement can avoid the noise of line delay. Our NXRO uses only 11 Slice, including 10 Slices used by 10 ROs, and 1 Slice used to place the XOR Tree.

IV. Experimental Results

To avoid the contingency of experimental data from individual development boards, we select two Xilinx Spartan-6 XC6SLX16 FPGA microboards (XC6SLX16 chip using 45nm process) and two Xilinx Atrix-7 XC7A35T FPGA microboards (XC7A35T chip using 28nm process) as the target FPGA devices to implement the NXRO TRNG. The standard operating conditions is 20°C, core voltage 1.2V (Spartan-6), 1.0V (Atrix-7), respectively. Figure 6 shows the complete experimental platform, which consists of a high/low temperature chamber, a DC power supply and a host computer. The data output from the TRNG is sent to the host computer through the USB-UART. Finally, the host computer calculates randomness of received data.

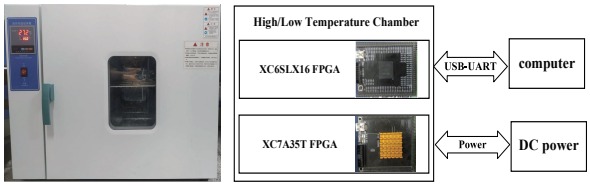


Fig. 6: Experimental System.

In this brief, we measure the performance of NXRO TRNG by seven different evaluation metrics, such as NIST Test (include NIST 800-22 test and NIST 800-90B test), AIS31 Test, Deviation Test, Autocorrelation Test, Restart Test, Voltages and Temperatures Test. Finally, we compare the performance and resource overhead of the proposed NXRO TRNG with the advanced FPGA-based TRNG architectures of recent years and present our conclusion.

TABLE I: NIST Test Results of Spartan-6 FPGAs and Artix-7 FPGAs

	Spartan-6				Artix-7			
	Chip1		Chip2		Chip1		Chip2	
NIST SP 800-22	P-value	Prop.	P-value	Prop.	P-value	Prop.	P-value	Prop.
Frequency	0.534146	10/10	0.911413	10/10	0.534146	10/10	0.991468	10/10
BlockFrequency	0.534146	10/10	0.350485	10/10	0.911413	10/10	0.739918	10/10
CumulativeSums*	0.534146	10/10	0.637032	10/10	0.545202	10/10	0.637032	10/10
Runs	0.739918	10/10	0.213309	10/10	0.350485	10/10	0.534146	10/10
LongestRun	0.739918	10/10	0.350485	10/10	0.534146	10/10	0.350485	10/10
Rank	0.350485	10/10	0.534146	10/10	0.739918	10/10	0.534146	10/10
FFT	0.122325	9/10	0.911413	9/10	0.534146	10/10	0.350485	10/10
NonOverlappingTemplate*	0.486995	10/10	0.485061	10/10	0.489674	10/10	0.550849	10/10
OverlappingTemplate	0.534146	10/10	0.066882	10/10	0.911413	10/10	0.171867	10/10
Universal	0.739918	9/10	0.350485	10/10	0.739918	9/10	0.350485	10/10
ApproximateEntropy	0.350485	10/10	0.739918	10/10	0.911413	9/10	0.455937	10/10
Serial*	0.476614	50/50	0.115611	10/10	0.516869	10/10	0.192829	10/10
LinearComplexity	0.534146	10/10	0.350485	10/10	0.739918	10/10	0.350485	10/10
NIST SP 800-90B	p-max	h-min	p-max	h-min	p-max	h-min	p-max	h-min
MCV	0.500032	0.998788	0.500072	0.998671	0.500121	0.998530	0.500163	0.998410
Collision	0.499999	0.935669	0.500000	0.951989	0.499999	0.965102	0.500000	0.946539
Markov	2.97E-39	0.999895	3.14E-39	0.999238	3.18E-39	0.935367	3.22E-39	0.998940
Compression	0.5	1	0.5	1	0.5	1	0.5	1
t-Tuple	0.521091	0.939317	0.521693	0.937653	0.522265	0.936073	0.523389	0.932974
LRS	0.500414	0.998803	0.512011	0.964658	0.500139	0.998478	0.500012	0.998843
Multi-MCW	0.500165	0.999523	0.500466	0.998654	0.500149	0.999570	0.500433	0.998749
Lag	0.501242	0.996418	0.500502	0.998552	0.500508	0.998535	0.500890	0.997433
Multi-MMC	0.502364	0.993194	0.500358	0.998967	0.500244	0.999294	0.500555	0.998398
LZ78Y	0.500350	0.998989	0.500422	0.998783	0.500220	0.999363	0.500514	0.998518

* the p-value is the average of the p-values of all subtests.

A. NIST Test

For each microboard, we collected 10 sets of random data with a length of 1 million bits under standard operating conditions. The NIST SP 800-22 evaluates the performance of TRNG by calculating the *P-value* of a random sequence with 15 tests, where '*P-value*' is an indicator of the randomness of the sequence and '*Prop.*' indicates the proportion of passing tests. If the *P-value* is lower than 0.01, it indicates a poor randomness of the random sequence.

The NIST SP 800-90B contains tests for non-independent identical distribution to evaluate the randomness of the data by minimum entropy estimates, where the final estimate is the smallest of the ten smallest entropy values. The p-max is the probability of observing the most common sample and the h-min is the minimum entropy estimate. The NIST test experimental results are shown in Table I.

As can be seen from Table I, the *P-value* of random sequences generated by different FPGA microboards are all greater than 0.01, and the average *Prop.* is above 96%. The lower *P-value* in the table may be influenced by the chip differences. The minimum entropy of our proposed TRNG is 0.939317 and 0.937653 (Spartan-6), 0.935367 and 0.932974 (Atrix-7), respectively. The above results shows that our proposed NXRO TRNG generate random sequence bit streams can all pass the randomness test, and it also means that the random sequence has good randomness.

B. AIS31 Test

AIS31 consists of three functional stages: P1 (T0∼T5), P2 (T6∼T8), P2 is compatible with P1 but is more strictly than P1. Each stage requires 3,145,728 bits, 5,880,096 bits and 7,200,000 bits respectively. Therefore, a total of 17 million bits of data was used for the tests. As can be seen in Table II, all the FPGA microboards passed the tests.

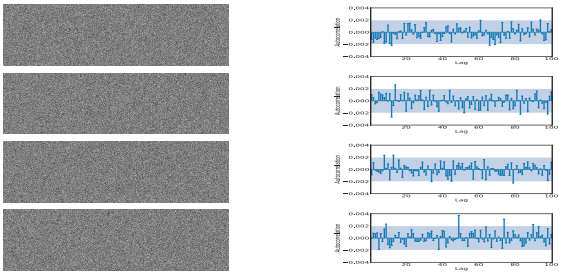
C. Deviation Test

We generated one million consecutive bits separately using the FPGA microboard under standard operating conditions and generated a bitstream image from the collected data. Figure 7(a) shows the generated bitstream image (image size 2000

TABLE II: AIS31 Test Results of Spartan-6 FPGAs and Artix-7 FPGAs

AIS31	Spartan-6		Artix-7	
	Chip1	Chip2	Chip1	Chip2
disjointness test(T0)	Pass	Pass	Pass	Pass
monobit tests(T1)*	100	100	100	100
poker tests(T2)*	100	100	100	100
run tests(T3)*	100	100	100	100
long run test(T4)*	100	100	100	100
autocorrelation test(T5)*	100	100	100	100
uniform distribution test(T6)	Pass	Pass	Pass	Pass
multinomial distributions(T7)	Pass	Pass	Pass	Pass
entropy test(T8)	Pass	Pass	Pass	Pass

* this is the pass rate.



(a) (b)

Fig. 7: Test Results of Spartan-6 FPGAs and Artix-7 FPGAs: (a) Deviation Test bitstream images (pixel size 2000*500), (b) Autocorrelation Test.

× 500), where the black pixels represent 1 and the white pixels represent 0. It can be seen that the distribution of black and white pixel points in the bitstream image is uniform, which means that the number of 0 and 1 in the generated random sequence is approximately equal. In addtion, Equation 3 calculate the statistical bias of our proposed TRNG on different microboards. addition

$$Bias = \frac{N_0 - N_1}{1000000} \times 100\% \tag{3}$$

In Equation 3, N_0 and N_1 represent the number of occurrences of '0' and '1' in the random sequence, respectively. As shown in Table III. The statistical deviations of the four microboards generated TRNG random sequences are 0.006409% and 0.014479%(Spartan-6), 0.032616% and -0.024304%(Atrix-7), respectively. This can proves that the proposed TRNG can generate random number sequences with low deviations.

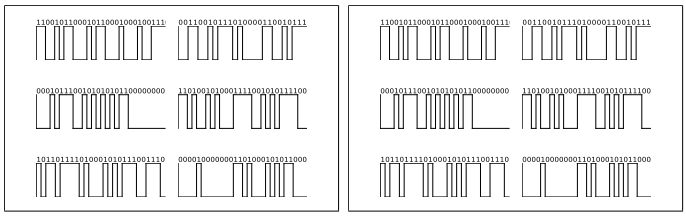
D. Autocorrelation Test

The correlation at different times in a random sequence can be calculated by the autocorrelation coefficient. Therefore, we collected one million consecutive bits on Spartan-6 and Atrix-7 FPGA microboards, respectively, and calculated the autocorrelation coefficient in Python. The lag values in the autocorrelation function were set from 1 to 100, and analyze the autocorrelation factor (ACF) between variables with lags of 1 to 100 bits. According to the statistical index designed by Karl Pearson, correlation can be considered irrelevant when the correlation coefficient is less than 0.3.

The results of the autocorrelation test are shown in Figure 7(b), where the ACF of all data sets fluctuates between -0.004 to 0.004, which means that all correlation coefficients are below 0.3. Therefore, it can be concluded that the random sequences generated by the proposed TRNG are not autocorrelated.

TABLE III: Bias Test Results of Spartan-6 FPGAs and Artix-7 FPGAs

	Spartan-6		Artix-7	
	Chip1	Chip2	Chip1	Chip2
Deviation	0.006409%	0.014479%	0.032616%	-0.024304%



(a) (b)

Fig. 8: Restart Test Results of (a) Spartan-6 FPGAs and Artix-7 FPGAs (b).

E. Restart Test

Ideally, the random sequences generated by the TRNG should be inconsistent at each power-up. Therefore, we collected six sets of 30 consecutive bits generated on the Spartan-6 and Atrix-7 FPGA microboards under standard operating conditions, respectively. As shown in Figures 8, all random sequences were different, which means that after each power-up, the TRNG generated a different random bit stream. Therefore, it can be concluded that the random sequences generated by the TRNG are not similar to each other and has strong randomness.

F. Voltages and Temperatures Test

The output quality of the TRNG can be affected by voltage, temperature, and noise. Therefore, we evaluated the performance of NXRO at different temperatures and voltages, we based on standard operating conditions: 20°C at 1.2V for Spartan-6 and 20°C at 1.0V for Atrix-7; temperature change from 0°C to 80°C, with 20°C change per step; and voltage operation from 1.0V to 1.4V (Spartan-6), 1 V to 1.2V (Atrix-7), with 0.2V change per step. 0.8 V to 1.2 V (Atrix-7), changing 0.2 V for each step. For each step, we collected 10 sets of 1 million bits of random bit streams. Finally, we selected the NIST SP 800-22 test suite to evaluate the percentage of passing tests (The following is referred to as the pass rate), and the results are shown in Figure 9.

As can be seen in Figure 9, both Spartan-6 and Atrix-7 generate random bitstream sequences with the highest pass rate in the environment of 20°C with standard core voltage, and the best randomness is achieved at this time. However, with the increase of temperature, the pass rate of NXRO decreases on different microboards. Besides, the change of voltage also affects the pass rate of NXRO. Lower voltages

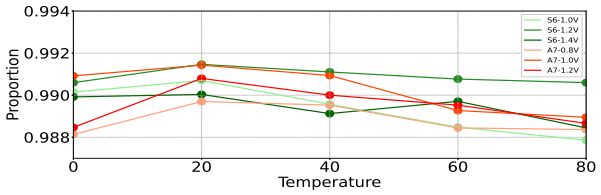


Fig. 9: Voltages and temperatures test Results of Spartan-6 FPGAs and Artix-7 FPGAs.

result in lower pass rate for the generated bitstreams, and decrease more rapidly at increasing temperatures. Although the drop in NXRO pass rate at higher voltages is not as fast as at lower voltages, both are less stable than at standard voltages.

G. Comparison With Other TRNGs Based on FPGA

We compare the performance difference between our proposed NXRO TRNG and advanced TRNG by seven dimensions (columns 3 to 9 in Table IV): (1) experimental platform of TRNG; (2) chip process; (3) entropy source of TRNG; (4) hardware overhead (include Look-up table (LUT) and D flip-flop (DFF)); (5) TRNG throughput rate (Mbps)); (6) throughput rate to slice ratio (Mbps/Slice); (7) power of TRNG. The comparison results are shown in Table IV.

From Table IV, it can be seen that these TRNG implementations are on advanced process FPGAs and most of them have good performance. [2] uses RO with multi-stage feedback to improve the quality of the entropy sources, and although only 24 LUTs and 2 DFFs are used, their throughput rate still can be improved. [5] proposes an ultra-lightweight RO structure, each RO only consists of one D-Latches and one inverter, and the whole TRNG uses only 4LUTs and 3DFFs, but their throughput rate was only 0.76Mbps. [6] uses Galois RO to sample the high-speed clock from the PLL, and although it achieves a throughput rate of 280Mbps, its resource overhead still needs to be reduced. [7] proposed the delay chain feedback loop (DCFL) structure, which consists of an XNOR gate and several inverters, and achieved a throughput rate of 150Mbps but used high resource overhead. [9] use one multiphase sampler with eight sampling points to sample four 2-LUT ROs, which achieves low resource overhead but the throughput rate needs to be enhanced. [8] also uses XOR gate to mix the output of RO, but different with NXRO, it XOR gate is used to nest the output of the previous RO.

Therefore, our NXRO achieves 350 Mbps (Spartan-6) and 575 Mbps (Atrix-7) throughput rates by using 35 LUTs and 22 DFFs, and achieves 31.82 and 52.27 in the ratio of throughput rate to resource overhead ($\frac{Throughput}{Slice}$), and the total logic power obtained from ISE and Vivado power analysis results is only 0.068W and 0.135W, respectively. This proves our TRNG is low resource overhead and high throughput rate.

V. Conclusion

In this brief, we use self-feedback NAND gate as the enable signal of the XOR RO to control its oscillation state. The output of the self-feedback NAND gate and the output of the XOR RO are both used as the output of the RO and then sampled by the DFFs. Our NXRO improves the randomness and throughput rate of the entropy source and has simple structure. Experimental analysis shows that our proposed NXRO uses 35 LUTs and 22 DFFs, which achieves throughput rates of 350 Mbps on Spartan-6 FPGA and 575 Mbps on Atrix-7 FPGA, respectively. Compared with existing RO-based TRNGs, our proposed NXRO is more compact, lower resource overhead and higher throughput rate.

TABLE IV: Resource Cost and Throughput Comparison With Other State-of-the-Art FPGA TRNG Designs

Design	Platform	Process (nm)	Area		Throughput ($Mbps$)	Throughput/Slice ($\frac{Mbps}{Slice}$)	Power (W)
			LUTs	DFFs			
[2] 2022	Virtex-6 Spartan-6	40 45	24	2	290 150	-	3.687
[3] 2021	Spartan-6 Virtex-6	45 40	56	19	100	-	0.00115
[4] 2023	Zynq XC7Z020	28	38	121	300	7.89	-
[5] 2022	Spartan-6	45	4	3	0.76	0.76	-
[6] 2020	Aritix-7	28	50	79	280	-	-
[7] 2017	Cyclone IV Cyclone II	60 90	298	-	150 133	-	-
[8] 2018	Artix-7	28	25	37	160	-	-
[9] 2023	Artix-7	28	24	33	275.8	21.21	0.049
[12] 2020	Virtex UltraScale+	16	791	559	2.43	0.013	-
This work	**Spartan-6** **Artix-7**	**45** **28**	**35**	**22**	**350** **575**	**31.82** **52.27**	**0.068** **0.135**

References

[1] R. Serrano, C. Duran, M. Sarmiento, T.-T. Hoang, A. Tsukamoto, K. Suzaki, and C.-K. Pham, "A robust and healthy against PVT variations TRNG based on frequency collapse," *IEEE Access*, vol. 10, pp. 41852–41862, 2022.

[2] J. Cui et al., "Design of True Random Number Generator Based on Multi-Stage Feedback Ring Oscillator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1752-1756, March 2022.

[3] X. Wang et al., "High-Throughput Portable True Random Number Generator Based on Jitter-Latch Structure," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 2, pp. 741-750, Feb. 2021.

[4] F. Frustaci, F. Spagnolo, S. Perri and P. Corsonello, "A High-Speed FPGA-Based True Random Number Generator Using Metastability With Clock Managers," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 2, pp. 756-760, Feb. 2023.

[5] R. Della Sala, D. Bellizia and G. Scotti, "A Novel Ultra-Compact FPGA-Compatible TRNG Architecture Exploiting Latched Ring Oscillators," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1672-1676, March 2022.

[6] J. Lin, Y. Wang, Z. Zhao, C. Hui and Z. Song, "A New Method of True Random Number Generation based on Galois Ring Oscillator with Event Sampling Architecture in FPGA," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, Dubrovnik, Croatia, 2020, pp. 1-6.

[7] X. Wu and S. Li, "A new digital true random number generator based on delay chain feedback loop," in *Proc. IEEE Int. Symp. Circuits Syst (ISCAS)*, Baltimore, MD, USA, 2017, pp. 1-4.

[8] F. Mei, L. Zhang, C. Gu, Y. Cao, C. Wang and W. Liu, "A Highly Flexible Lightweight and High Speed True Random Number Generator on FPGA," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Hong Kong, China, 2018, pp. 399-404.

[9] Z. Lu, H. Qidiao, Q. Chen, Z. Liu, and J. Zhang, "An FPGA-Compatible TRNG with Ultra-High Throughput and Energy Efficiency," in *Proc. DAC*, San Francisco, USA, 2023, pp. 1-6.

[10] A. Hajimiri, S. Limotyrakis, and T. H. Lee, "Jitter and phase noise in ring oscillators," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 790–804, Jun. 1999.

[11] N. Fujieda, M. Takeda, and S. Ichikawa, "An analysis of DCM-based true random number generator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 6, pp. 1109–1113, Jun. 2020.

[12] X. Li, P. Stanwicks, G. Provelengios, R. Tessier, and D. Holcomb, "Jitter-based adaptive true random number generation for FPGAs in the cloud," in *Proc. Int. Conf. Field-Programmable Technol. (ICFPT)*, Maui, HI, USA, 2020, pp. 112–119.

[13] V. Rozic and I. Verbauwhede, "Random numbers generation: Investigation of narrowtransitions suppression on FPGA," in *Proc. Int. Conf. Field Program. Logic Appl. (ICFPLA)*, Prague, Czech Republic, 2009, pp. 699-702.

基于低测试逃逸的晶圆级适应性测试方法

曲金星　梁华国*　潘宇琦　易茂祥　鲁迎春　章宏
（合肥工业大学 微电子学院，合肥 230009）

摘　要：为了降低集成电路中测试成本，提高测试质量。该文提出一种低测试逃逸的晶圆级适应性测试方法。该方法根据历史测试数据中测试项检测故障晶粒的有效性筛选测试集，降低待测晶圆的测试成本。同时，分析晶粒邻域参数波动程度，将存在波动晶粒的参数差异进行放大并建模，提高该类晶粒质量预测模型的分类准确率；无波动的晶粒使用有效测试集建模的方法进行质量预测，减少测试逃逸的风险。根据实际晶圆生产数据的实验结果表明，该方法可以明显降低晶圆的测试项成本40.7%，并保持较低的测试逃逸率0.0091%。

关键词：晶圆级适应性测试；邻域参数波动；参数差异；质量预测

Low Test Escape Based on Wafer-Level Adaptive Testing Method

QU Jinxing　LIANG Huaguo*　PAN Yuqi　YI Maoxiang　LU Yingchun　ZHANG Hong

(School of Microelectronics, Hefei University of Technology, Hefei 230009)

Abstract: In order to reduce the test cost and improve the test quality in ICs. The paper proposes a wafer-level adaptive test method with low test escapes. The method reduces the test cost of wafers to be tested by filtering the test set based on the effectiveness of the test item to detect faulty die in historical test data. At the same time, the degree of fluctuation of the parameters in the neighborhood of the die is analyzed, and the parameter differences of the die with fluctuations are amplified and modeled to improve the classification accuracy of the quality prediction model for this type of dies; the dies without fluctuations are used for quality prediction using the valid test set modeling method to reduce the risk of test escapes. Experimental results based on actual wafer production data show that the method can significantly reduce the test item cost of wafers by 40.7% and maintain a low test escape rate of 0.0091%.

Keywords: wafer-level adaptive testing; neighborhood parameter fluctuation; parameter variance; quality prediction

1　引言

随着集成电路技术的发展，晶体管的尺寸不断地减小，其制造成本也在随之减少。但是每个晶体管的测试成本并没有随着工艺的进步而降低。随着电路板变得越来越复杂，测试成本占总制造成本的很大一部分，有时甚至占主导地位[1]。因此提出了适应性测试的概念：通过对历史测试数据的分析，对后续的测试条件、测试内容等进行优化，在保证测试质量的同时，又可以降低测试成本[2]。在晶圆适应性测试过程中，为了降低测试成本需要对晶粒的测试项数量或者顺序进行调整，删减测试项数量可能会导致故障晶粒产生测试逃逸。在晶圆测试过程中，测试项指的是要测试的电气指标，如电阻、电容、电流等，而参数则是这些测试项的具体数值，如静态漏电流，反向击穿电压和输出电压等参数。在晶圆制造过程中，即使制造工艺完全相同，由于生产设备和环境等不可控因素会导致同一批次之间不同的晶圆或者同一块晶圆中相邻位置的晶粒出现参数差异的情况[3]。因此如何从邻域参数差异方面权衡测试成本和测试逃逸成为了集成电路测试领域的重大挑战。

为了应对这一挑战，一部分学者对晶粒测试项相关性进行研究。该研究主要是分析测试项之间存

*通信作者：梁华国　huagulg@hfut.edu.cn

基金项目：国家重大科研仪器研制项目(62027815)；国家自然科学基金重点项目(61834006)
Foundation Item：The National Major Research Instrument Development Project (62027815); The National Natural Science Foundation of China Key Project (61834006)

在的线性或非线性关系，通过统计或机器学习的方法对原测试内容进行优化。卡内基梅隆大学相关学者使用贪心算法从原有测试项中进行循环迭代，找到高故障覆盖率的测试集[4]。但是这种方法可能会导致过拟合现象、模型泛化能力下降、产生过多的测试逃逸和产量损失。杜克大学相关学者提出使用贝叶斯网络模型，基于测试项之间的互信息和马尔科夫毯的原理选择测试子集，以此来降低测试成本[5]-[6]。东华大学相关学者通过分析使用最大相关性最小冗余项算法分析测试项和测试结果之间的关系，并通过深度信念网络模型准确率选择最终的测试集[7]。该类研究往往为了降低测试成本，会对测试内容进行删减，从而导致晶粒出现测试逃逸。

另一部分学者对相邻位置晶粒进行研究。这类研究主要是分析待测晶粒与邻域晶粒存在的差异性以此找到存在测试逃逸的晶粒。加州大学相关学者提出使用晶粒测量值和邻域晶粒的参数均值得出的残差向量作为待测晶粒的特征、邻域晶粒的参数值中位数作为新的特征，使用双边滤波的非线性滤波方法检测存在可能测试逃逸的晶粒[8]-[10]。奈良科学技术大学相关学者对晶圆中不同位置晶粒之间的变化进行研究，提出使用高斯线性模型分析相邻晶粒之间差异性，分析可能存在测试逃逸的晶粒[11]。新加坡国际大学相关学者分析批次间以及批次内晶圆由工艺波动导致的参数差异，提出使用高斯混合模型对晶粒进行分类[3]。西安电子科技大学相关学者提出相邻晶粒的参数偏差和邻域参数波动结合的方法找到可能存在测逃逸的晶粒[12]。相邻位置晶粒的研究更多是根据相邻晶粒的测试结果或者参数值分析晶粒是否存在测试逃逸。

综上所述，为了降低晶圆的测试成本，本文提出有效测试集筛选的测试方法。为了提高测试质量，本文提出邻域参数波动程度的指标，对有邻域参数波动的晶粒采用邻域参数差异放大化的处理方法，将合格晶粒和故障晶粒之间存在的微小差异放大化，从而提高机器学习模型的分类准确率，减少测试逃逸晶粒的数量。无邻域参数波动的晶粒采用机器学习对有效测试集建模的方法，降低测试逃逸和产量损失的风险。本文方法的平均测试项减少率为40.7%，测试逃逸率为 0.0091%。

2　研究基础

2.1 参数波动

在晶圆生产制造过程中，无法避免工艺波动的产生，从而导致了晶圆中参数之间出现不同程度的波动情况[3]。通常相关学者会对晶圆整体参数波动进行分析，找到晶圆中存在晶粒参数波动异常的大致位置，以此找到存在测试逃逸的晶粒，并对晶圆制造过程进行优化和改进。无论是同一块晶圆或者同一个批次，相同参数的测量值也是有变化。如图 1 所示为工艺波动导致三个批次的某一参数均值对比和批次内三块晶圆参数均值对比。从图中可以看出，三个批次间的参数均值是不同的，在批次内部不同晶圆的参数均值也是不一样的。邻域参数表示待测晶粒相邻位置晶粒的参数。全局参数波动的分析往往会掩盖邻域参数波动的存在，无法利用其邻域参数之间的微小差异对晶粒进行分析，从而导致了待测晶粒可能会出现测试逃逸的风险。

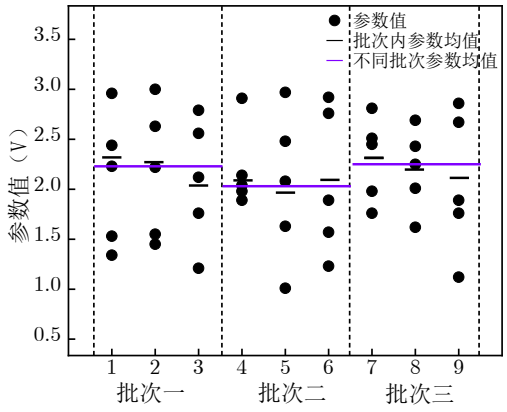


图 1 工艺波动影响参数均值差异图

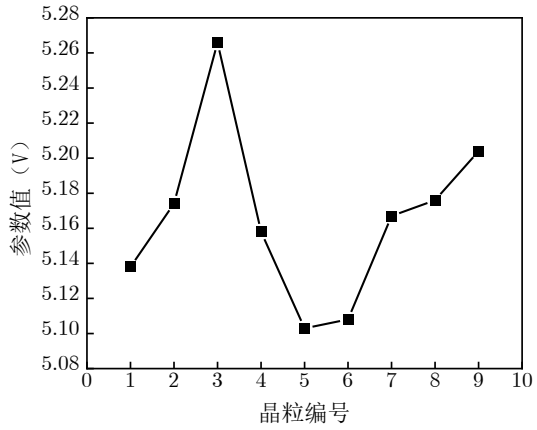


图 2 邻域参数波动图

邻域参数波动是基于晶圆中相邻位置晶粒的空间相关性在参数方面的表现形式。如图 2 所示，邻域参数波动表示在相邻位置范围内晶粒的参数呈现出波动的程度。通过分析参数波动可以掌握晶粒质量变化规律，并依据晶粒不同邻域参数波动程度使用不同的预测模型，可以减少晶粒产生测试逃逸的

风险。

2.2 参数差异放大化

参数差异性放大化是将晶粒原有参数值转换为差异大的特征值来表示。原有参数值之间的差异性很小，有时无法准确地识别出合格晶粒和故障晶粒之间的不同，容易在测试过程中产生测试逃逸和产量损失。对于存在邻域参数波动的晶粒，存在不同的波动情况和邻域参数之间的差异性。如果使用统计学的方法对邻域参数波动进行异常波动筛选，会导致晶粒出现严重的测试逃逸和产量损失。比如：合格晶粒邻域参数波动很大会被误判故障晶粒；故障晶粒邻域参数波动很小，会被误判为合格晶粒。因此需要结合邻域参数之间存在的微小差异对晶粒进行建模，而机器学习对于这种数据之间存在微小的差异无法进行准确的建模，所以对邻域参数进行差异放大化处理，则可以有效解决这种问题。

待测晶粒参数与邻域参数之间的差异性可以通过残差向量进行表示，通过计算待测晶粒的测量值和邻域晶粒参数的均值之间的差值，并将其作为晶粒的特征进行建模，可以有效找到存在测试逃逸的晶粒[8]。但是这种计算方法准确度容易被邻域晶粒参数中存在的异常值所影响，而中位数既可以表示邻域晶粒参数的分布特征，又可以避免邻域参数中异常值对于量化过程的影响。

2.3 质量预测模型

质量预测模型主要是使用历史晶圆的晶粒参数信息或者坐标信息和测试结果使用机器学习训练出的预测模型[13]-[14]。本文方法中既使用了历史晶粒的测试项信息训练的特征参数质量预测模型，又使用了其空间坐标信息训练空间波动质量预测模型。如图 3 所示为使用历史晶粒进行随机森林建模的过程。本文采用随机森林算法对晶粒集进行自举采样并对每个样本集选取不同的有效测试项进行建模，从而避免模型过拟合，且每个子模型之间并行进行，模型泛化能力强。特征参数质量预测模型是针对于没有邻域参数波动的晶粒训练的模型。将历史测试数据中的有效测试集作为特征，测试结果作为标签，使用随机森林训练模型。空间波动质量预测模型是针对于有邻域参数波动的晶粒训练的模型。将经过邻域参数差异放大化的有效测试集作为新特征，测试结果作为标签，使用随机森林训练模型。

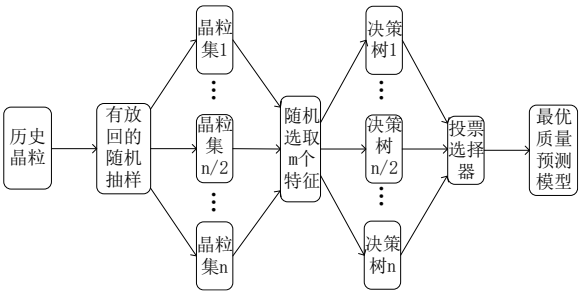


图 3 随机森林建模过程

3 低测试逃逸的晶圆级适应性测试方法

本节将详细描述所提出的低测试逃逸的晶圆级适应性方法建模流程。如图 4 所示，该方法主要分为三个部分：（1）测试集选择；（2）邻域参数波动性分析；（3）邻域参数差异放大化。该方法是基于历史数据进行分析处理并建模测试的晶圆离线测试方法。

在第（1）部分中，根据批次级晶圆历史测试数据进行有效测试集筛选，选择可以检测故障晶粒的测试集。这些有效测试集在邻域空间内存在有无参数波动两种情况。因此需要在（2）中对邻域参数波动进行分析，并将待测晶粒进行分类。对于存在邻域参数波动的晶粒，其参数之间存在大量的信息，需要在（3）中对其进行邻域参数差异放大化处理，然后使用新特征建模进行质量预测；而对于无邻域参数波动的晶粒，通过特征参数模型对其进行质量预测。

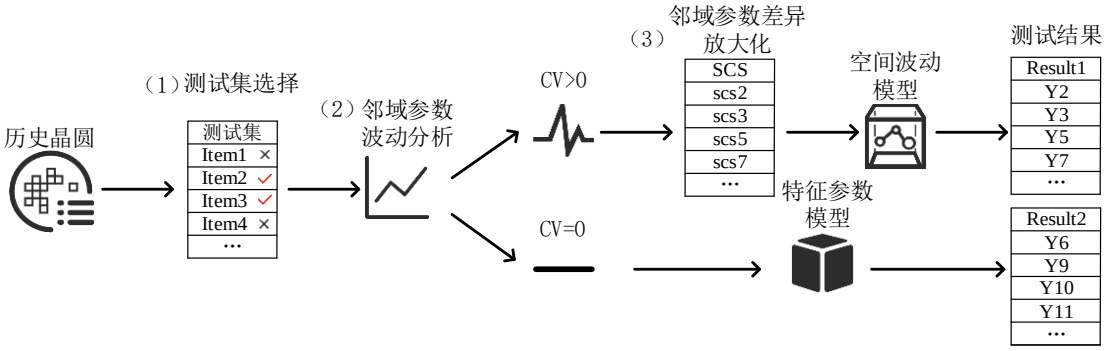


图 4 基于低测试逃逸的晶圆级适应性测试方法建模流程

3.1 测试集选择

A 数据预处理

由于晶圆历史测试数据存在大量缺失值，对于邻域参数波动分析时和训练模型时会产生不良影响，需要对于历史测试数据的缺失值进行处理。在 ATE 测试晶圆过程中，遵循“首败即停”的原则，即在某一个测试项检测到故障晶粒时，后续的测试内容就不需要进行，所以导致测试数据表中会出现大量的缺失值。对于这些缺失值，采用同一测试项下无故障晶粒的参数数值的中位数进行填充，从而可以让历史数据集完整以便进行下一步的邻域参数波动分析和邻域参数差异放大化处理。

B 有效测试集筛选

有效测试集筛选的目的是获得一个可以降低测试成本同时减少测试逃逸风险的测试集。有效测试集将用于下一步的邻域参数波动性分析和邻域参数差异放大化处理。本文有效测试集选择主要是通过对历史晶圆测试数据进行分析，选择出可以检测到历史晶圆中所有故障晶粒的有效测试集。然后使用该测试集对待测晶圆进行测试。

如图 5 所示，对于批次历史晶圆测试数据，需要检测每块历史晶圆其测试项t_i测试到故障晶粒的数量λ是否为零，若是$\lambda \neq 0$，则证明该测试项t_i为有效测试集的子集，反之则说明该测试项t_i不能检测到故障晶粒。当测试项t_i循环迭代至所有历史晶圆时，可以得到最终的有效测试集。本文中的测试项对应为参数测试中某一具体测试内容（如：漏电流，反向击穿电压等等），且每个测试项并不针对于某一具体故障类型。

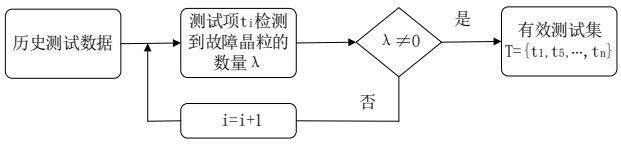


图 5 有效测试集筛选流程

3.2 邻域参数波动性分析

本文提出了以变异系数 CV（coefficient of variation）来表示晶圆中存在的邻域参数波动的情况，变异系数越小，表示其邻域参数波动越小，参数趋于稳定；反之，变异系数越大，表明其邻域参数波动越大，待测晶粒的参数越有可能是异常。如图 6 所示，本文中邻域参数的计算都是以 3×3 的窗口进行的，关于晶粒空间相关性研究的邻域窗口的选择是基于文献[15]中关于晶圆空间特征与质量之间的关系。CV 是邻域晶粒参数的标准差SD_{ij}和均值μ_{ij}的比值，它既消除了不同参数之间量纲的差异，又代表待测晶粒邻域参数的波动程度。其中邻域参数均值μ_{ij}由公式（1）可得：

$$\mu_{ij} = \frac{\sum P_{ab}}{n} \tag{1}$$

其中P_{ab}表示的是邻域晶粒的参数值，$i-1 \leq a \leq i+1$，$j-1 \leq b \leq j+1$，n表示邻域晶粒的个数。i表示晶粒在晶圆中 X 轴的坐标，j表示晶粒在晶圆中 Y 轴的坐标。邻域参数标准差SD_{ij}由公式（2）可得：

$$SD_{ij} = \sqrt{\frac{\sum(P_{ab} - \mu_{ij})}{n-1}} \tag{2}$$

最后 CV 的计算方法由公式（3）得到：

$$CV_{ij} = \left|\frac{SD_{ij}}{\mu_{ij}}\right| \tag{3}$$

其中CV_{ij}表示的是待测晶粒t的邻域参数波动大小，需要注意的是，在计算过程中邻域晶粒的参数值可能会出现异常值的情况，为了保证邻域参数波动量化的稳定性，需要在计算过程中去掉最大值最小值。

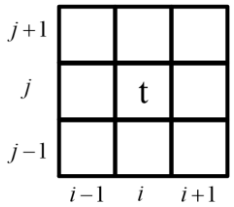


图 6 晶粒t的邻域晶粒分布

通过对于晶粒邻域参数波动的分析，可以将晶粒分为有无邻域参数波动的晶粒：有邻域参数波动晶粒，这类晶粒和邻域晶粒之间存在参数差异性，需要通过残差向量和邻域参数差异放大化将其特征进行放大分析；无邻域参数波动晶粒，其制造工艺相同，可以通过特征参数模型对其进行质量预测。

3.3 空间相关性强度

本文提出了以空间相关性强度（SCS）表示待测晶粒的邻域空间相关性的强度，如果待测晶粒参数和邻域晶粒参数之间差异性越大，晶粒的邻域空间相关性越弱，该晶粒是测试逃逸可能性就越大。在晶圆的生产制造过程中，缺陷会导致重要电压，电流等参数“阶跃”，而不是像正常晶粒参数那样逐渐变化。这种突然的变化会表现为负尖峰（晶粒参数低于邻域）或正尖峰（晶粒参数高于邻域）等异常现象[16]。空间相关性强度可以将晶粒中存在这种异常现象进行量化，并将存在测试逃逸的晶粒其特征放大化以便机器学习可以更加准确的获取存在测试逃逸晶粒的特征。

残差向量RV_{ij}的计算方法由公式（4）表示：

$$RV_{ij} = P_{ij} - M_{ij} \tag{4}$$

其中P_{ij}表示为待测晶粒的参数值，M_{ij}表示为邻域晶粒参数的中位数。这里采用以中位数表示领域参数的数据分布特点，而不是采用均值。因为实际生产过程中工艺波动对晶粒产生的影响，每个晶粒周围无法避免异常值的出现。若采用均值的方法，异常值会造成邻域参数计算偏差过大，而中位数既可以减少异常值对于邻域参数数据分布的影响，又可以表示邻域参数的数据分布特点。

空间相关性强度（SCS）的计算方法由公式（5）表示：

$$SCS_{ij} = \frac{RV_{ij}}{CV_{ij}} \tag{5}$$

其中RV_{ij}表示为待测晶粒参数值与邻域晶粒参数中位数的差值，CV_{ij}表示的是待测晶粒邻域晶粒参数波动的大小。

通过邻域参数波动分析，以 CV 的值为分类标准可以将历史晶圆中和待测晶圆中的晶粒分为有无邻域参数波动的晶粒。有邻域参数波动晶粒通过邻域参数差异放大化的过程将其参数之间的差异放大化后，以空间相关性强度SCS_{ij}替代原有的晶粒参数值P_{ij}，将历史晶圆的测试结果作为标签，空间相关性强度SCS_{ij}作为每个晶粒的特征，使用随机森林建立空间波动模型。

对于无邻域参数波动晶粒，以历史晶圆中无邻域参数波动（CV=0）晶粒的参数值P_{ij}作为该类晶粒的特征，以测试结果为标签，使用随机森林建立特征参数模型。通过该模型可以有效解决空间波动模型分析对于这类晶粒测试逃逸和产量损坏过高的问题。

在测试过程中，待测晶圆通过邻域参数波动分析，将里面的晶粒分为有无邻域参数波动的晶粒，有邻域参数波动晶粒使用空间波动模型对其进行测试得到测试结果 Result1；无邻域参数波动晶粒使用特征参数模型进行测试得到测试结果 Result2。

4　实验结果

4.1 实验平台设置以及数据描述

本次实验运行环境为 Intel i7-7700 CPU，16G 运行内存，Windows 10 系统。实验所用仿真平台为 python3.8，主要用到的库为 numpy 库，pandas 库和 scikit-learn 库。实验数据选取了来自实际工业生产过程中 3 个不同批次的模拟芯片测试结果，每个批次 12 块晶圆，总计 1485228 块晶粒。本次实验中随机抽样的每个批次的 5 块晶圆数据进行建模，剩余的 7 块晶圆数据用于验证模型。实验数据具体信息如表 1 所示。

表 1 实验数据分布

批次	晶粒总数	平均良率	测试项数
批次 1	383640	98.55%	67
批次 2	312312	98.96%	39
批次 3	789276	99.76%	25

4.2 邻域参数波动性分析结果

本文以变异系数（CV）来衡量晶粒的邻域参数波动大小。如图 7 所示为批次 1 中某块晶圆部分晶粒的邻域参数波动图。从图中可以看出，相邻位置晶粒参数之间的差异性很小，使用邻域参数波动更能反映出晶粒之间不同。有邻域参数波动晶粒邻域参数波动大小不同，再结合测量值和邻域晶粒参数中位数的差值，可以将其作为晶粒的新的特征，这类特征将更好区分合格晶粒和故障晶粒之间的不同。无邻域参数波动晶粒的特点是邻域晶粒参数稳定，但是因为不是全集测试，无法保证未测试的参数是否影响其最终的预测结果。如果使用邻域晶粒故障率（BNR）等空间相关性[17]去判定，会导致合格晶粒被判定为故障晶粒，故障晶粒被判定为合格晶粒。通过历史数据中没有邻域参数波动晶粒的参数和测试结果建模，可以减少测试逃逸和产量损失。

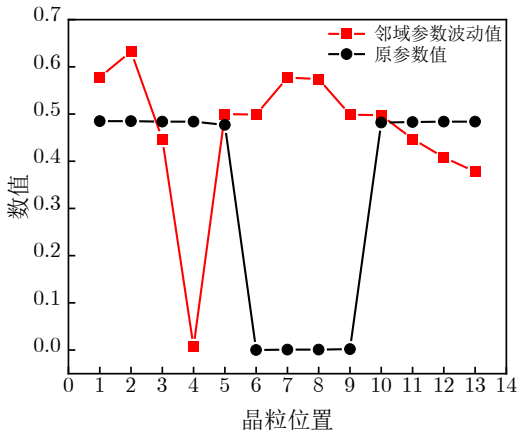


图 7 晶粒邻域参数波动

4.3 邻域参数差异放大化结果

对于有效测试集，相同测试项之间的参数差异性不同，其里面隐藏着很多有效的信息。对于每个批次中有邻域参数波动晶粒使用邻域参数差异放大化前后实验结果对比，如图 8 所示，在邻域参数差异放大化前晶粒参数值之间的差异性很小，无法区分合格晶粒和故障晶粒的不同，而在邻域参数差异放大化后，晶粒参数值之间的差异性明显变大，合

格晶粒和故障晶粒之间的特征变得更加明显，更容易找到存在测试逃逸的晶粒。

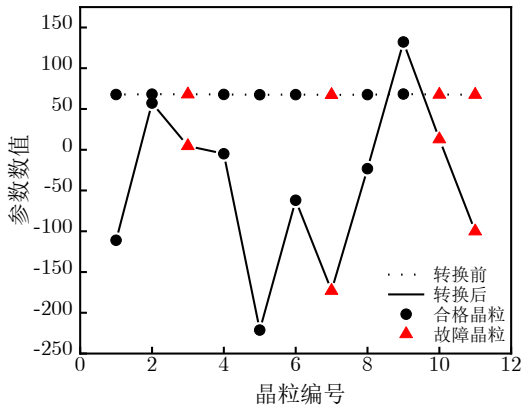


图 8 参数差异放大前后数值对比

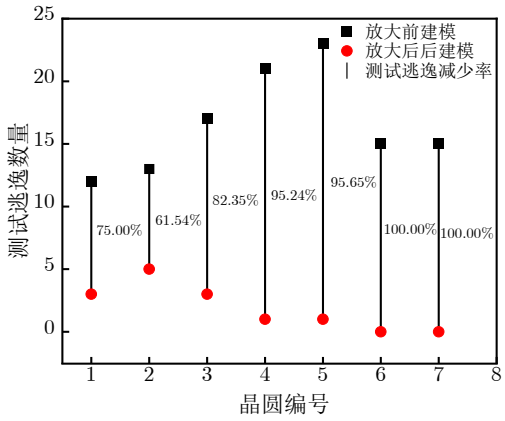


图 9 参数差异放大前后建模测试逃逸数量对比

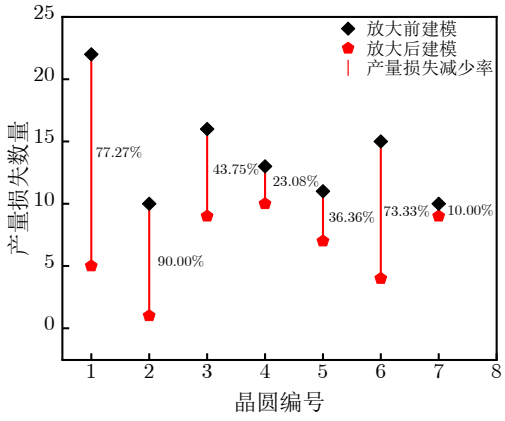


图 10 参数差异放大前后建模产量损失数量对比

如图 9 所示为批次 1 中七块晶圆邻域参数差异放大化前后建模的测试逃逸晶粒数量对比和测试逃逸减少率。从图中可以看出，在每块晶圆上进行邻域参数差异放大化后，存在测试逃逸的晶粒数量都明显少于邻域参数差异放大化前建模的数量。经过邻域参数差异放大化处理之后，存在测试逃逸的晶粒参数特征被明显放大了，这样更能捕捉到合格晶粒和故障晶粒之间的不同差异。

如图 10 所示为邻域参数差异放大化前后建模的产量损失数量对比和产量损失减少率。从图中可以看出，在每块晶圆上进行邻域参数差异放大化后，存在产量损失的晶粒数量都明显少于邻域参数差异放大化前建模的数量。因此本文方案在减少晶粒测试逃逸的数量，同时并没有增加其产量损失的数量。

4.4 方案对比

本次实验与 2020 年提出的过滤式和封装式相结合的测试方法[7]（方法 1）和 2022 年提出的位置相关性筛选方法[12]（方法 2）进行比较，实验主要比较了测试项减少率（TIRR）、测试逃逸率（TER）、产量损失率（YLR）三个指标。测试项减少率表示与标准测试（标准测试指的是使用原测试计划中所有的测试项）相比节省测试项的比例。该指标的计算公式如下：

$$\mathrm{TIRR} = 1 - \frac{N \times T_{Select}}{N \times T_{All}} \times 100\% \qquad (8)$$

其中N表示的是晶圆中待测晶粒的数量，T_{Select}表示有效测试集的测试项数量，T_{All}表示测试全集中测试项的数量。TIRR 的大小只与有效测试集筛选中测试项的数量有关，而与晶圆中晶粒数量无关。

测试逃逸率表示故障晶粒被预测为合格晶粒的数量占待测晶圆中总晶粒数量的比例，其计算公式如下：

$$\mathrm{TER} = \frac{N_{escape}}{N} \times 100\% \qquad (9)$$

其中N_{escape}表示测试逃逸晶粒的数量，N表示待测晶圆中晶粒的数量。测试逃逸率越低表示测试完成的晶圆中故障晶粒数量越少，但并不能保证测试质量越高，测试质量与产量损失率也有关。

产量损失率表示合格晶粒被预测为故障晶粒的数量占待测晶圆中晶粒数量的比例，其计算公式如下：

$$\mathrm{YLR} = \frac{N_{loss}}{N} \times 100\% \qquad (10)$$

其中N_{loss}表示产量损失晶粒的数量，N表示待测晶圆中晶粒的数量。

从表 2 可以看出，本文方法和方法 1 相比 TIRR 虽然增加了 11.28%，但是测试逃逸率比方法 1 降低了 84.04%，产量损失率降低了 59.35%。本文方法和方法 2 相比 TIRR 降低了了 11.94%，测试逃逸率比方法 2 降低了 51.39%，产量损失率降低了 50.13%。因此本文提出方法在降低晶圆测试成本同时保持较

低的测试逃逸率。在实际的晶圆制造过程，无法避免工艺波动的产生，从而也无法避免参数波动的发生，因此保证了本文所提出的方法的适用性。

表 2 对比实验结果

方案	TIRR	TER	YLR
本文方法	40.13%	0.0091%	0.0137%
方法 1	45.23%	0.0570%	0.0337%
方法 2	35.85%	0.0188%	0.0274%

5 结论

针对集成电路越来越复杂，测试成本日益增多，测试质量要求越来越高的问题。本文提出了一种基于低测试逃逸的晶圆级适应性测试方法用于降低晶圆测试成本，减少晶粒测试逃逸的数量。通过有效测试集选择可以降低待测晶圆测试成本。同时对其进行邻域参数波动分析区分有无邻域参数波动晶粒，对有邻域参数波动晶粒进行邻域参数放大化处理，放大合格晶粒和故障晶粒之间参数差异性，提高模型的准确率，减少测试逃逸和产量损失。对无邻域参数波动晶粒采用有效测试集建模方法保证其测试质量。本文方法的平均测试项减少率为 40.7%，测试逃逸率为 0.0091%。

参考文献

[1] Liu M, Pan R, Ye F, et al. Fine-Grained Adaptive Testing Based on Quality Prediction[J]. ACM Transactions on Design Automation of Electronic Systems, 2020, 25(5). doi: 10.1145/3385261

[2] Yilmaz E, Ozev S, Sinanoglu O, et al. Adaptive testing: Conquering process variations[C]. 2012 17th IEEE European Test Symposium (ETS). IEEE, 2012:1-6.

[3] R. Wang, L. Zhang and N. Chen, Spatial Correlated Data Monitoring in Semiconductor Manufacturing Using Gaussian Process Model[J]. IEEE Transactions on Semiconductor Manufacturing, 2019, 32(1):104-111, doi: 10.1109/TSM.2018.2883763.

[4] Xue C, Shawn R D. A one-pass test-selection method for maximizing test coverage[C]. IEEE International Conference on Computer Design. IEEE, 2015: 621-628, doi: 10.1109/ICCD.2015.7357173.

[5] Pan R , Zhang Z , Li X , et al. Black-Box Test-Cost Reduction Based on Bayesian Network Models[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(2):386-399, doi: 10.1109/TCAD.2020.2994257.

[6] Liu M, Li X, Chakrabarty K, et al. Knowledge Transfer in Board-Level Functional Fault Diagnosis Enabled by Domain Adaptation[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021, 41(3):762-775, doi: 10.1109/TCAD.2021.3065919..

[7] H. Xu, J. Zhang, Y. Lv and P. Zheng, Hybrid Feature Selection for Wafer Acceptance Test Parameters in Semiconductor Manufacturing[J]. IEEE Access, 2020,.8: 17320-17330, doi: 10.1109/ACCESS.2020.2966520.

[8] F. Lin, C. -K. Hsu and K. -T. Cheng, AdaTest: An efficient statistical test framework for test escape screening[C]. 2015 IEEE International Test Conference (ITC), Anaheim, CA, USA, 2015:1-8, doi: 10.1109/TEST.2015.7342391.

[9] F. Lin, C. -K. Hsu and K. -T. Cheng, Learning from Production Test Data: Correlation Exploration and Feature Engineering[C]. 2014 IEEE 23rd Asian Test Symposium, Hangzhou, China, 2014: 236-241, doi: 10.1109/ATS.2014.51.

[10] F. Lin, C. -K. Hsu and K. -T. Cheng, Feature engineering with canonical analysis for effective statistical tests screening test escapes[C]. 2014 International Test Conference, Seattle, WA, USA, 2014:1-10, doi: 10.1109/TEST.2014.7035344.

[11] M. Shintani, R. -U. -H. Mian, M. Inoue, T. Nakamura, M. Kajiyama and M. Eiki, Wafer-level Variation Modeling for Multi-site RF IC Testing via Hierarchical Gaussian Process[C]. 2021 IEEE International Test Conference (ITC), Anaheim, CA, USA, 2021:103-112, doi: 10.1109/ITC50571.2021.00018.

[12] J. Zhang, H. You, R. Jia and X. Wang, The Research on Screening Method to Reduce Chip Test Escapes by Using Multi-Correlation Analysis of Parameters[J]. IEEE Transactions on Semiconductor Manufacturing, 2022, 35(2):266-271, doi: 10.1109/TSM.2022.3144283.

[13] Katragadda V, Muthee M, Gasasira A, et al. Algorithm based adaptive parametric testing for outlier detection and test time reduction[C]. 2018 IEEE International Conference on Microelectronic Test Structures (ICMTS). IEEE, 2018: 142-146.

[14] Kuo Y T, Lin W C, Chen C, et al. Minimum Operating Voltage Prediction in Production Test

Using Accumulative Learning[C]. 2021 IEEE International Test Conference (ITC). IEEE, 2021: 47-52.

[15] Daasch W R , Cota K , Mcnames J , et al. Neighbor selection for variance reduction in IDDQ and other parametric data[C]. Test Conference, 2002. Proceedings. International. 2002.

[16] J. Zhang, H. You and R. Jia, Reliability hazard characterization of wafer-level spatial metrology parameters based on LOF-KNN method[C]. 2019 IEEE 26th International Symposium on Physical and Failure Analysis of Integrated Circuits (IPFA), Hangzhou, China, 2019:1-4, doi: 10.1109/IPFA47161.2019.8984814.

[17] C. -H. Yang et al., Identifying Good-Dice-in-Bad-Neighborhoods Using Artificial Neural Networks[C]. 2021 IEEE 39th VLSI Test Symposium (VTS), San Diego, CA, USA, 2021:1-7, doi: 10.1109/VTS50974.2021.9441055

曲金星：男，硕士生，研究方向为集成电路测试.

梁华国：男，教授，研究方向为容错计算与硬件安全.

潘宇琦：男，博士生，研究方向为集成电路测试.

易茂祥：男，教授，研究方向为 VLSI 可靠性及安全性设计.

鲁迎春：男，副教授，研究方向为集成电路硬件安全.

章　宏：女，博士生，研究方向为集成电路测试.

LRHOP：一种基于时空特征的 DeepFake 集成检测方法

李元昕[①]，李晓龙[②]，马钰锡[①]，孟令中*[①]

[①](中国科学院软件研究所　北京　100190)

[②](北京交通大学信息科学研究所　北京　100190)

摘　要：随着人工智能的迅速发展，多媒体信息安全得到越来越多的关注，深度伪造视频(DeepFake)的大量传播让公众无法判断信息的真假，对公民的名誉安全和国家安全都有着重大影响，DeepFake 检测成为重点研究的问题之一。目前大多数基于图像特征的检测方法虽然性能表现良好，但对于篡改痕迹的挖掘仍然考虑得不够充分。为了更大程度地利用伪造过程在不同层次留下的线索，本文提出了一种时空特征结合的集成检测方法，在基于帧内图像检测的模型上结合时序检测模型，通过对帧间运动模式差异的检测来弥补基于图像特征检测的不足，融合图像特征和时序特征，提高检测性能。集成后的方法与原模型 DefakeHOP 对比在 FF++和 Celeb-DF-V2 数据集上的 AUC 值分别提升了 0.15%和 0.86%，其中在检测难度较大的 Celeb-DF-V2 数据集上准确率和精确率分别提高了 13.39%和 15.13%。体现了时空特征集成对于 DeepFake 检测的有效性。

关键词：深度伪造检测；多媒体安全；时空结合；集成方法

LRHOP: A DeepFake ensemble detection method based on spatiotemporal features

LI Yuanxin[①], LI Xiaolong[②], MA Yuxi[①], MENG Lingzhong[①]

[①] (*Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*)

[②] (*Institute of Information science, Beijing Jiaotong University, Beijing 100190, China*)

Abstract: With the rapid development of artificial intelligence, multimedia information security has received increasing attention. The widespread dissemination of DeepFake videos makes it difficult for the public to judge the authenticity of information, which has a significant impact on the reputation and national security of citizens. DeepFake detection has become one of the key research issues. At present, although most image feature based detection methods perform well, they still do not fully consider the mining of tamper marks. In order to make greater use of the clues left by the forgery process at different levels, this paper proposes an integrated detection method that combines spatiotemporal features. By combining the temporal detection model with the intra frame image detection model, the shortcomings of image feature detection are compensated by detecting inter frame motion mode differences. The fusion of image features and temporal features improves detection performance. Compared with the original model DefakeHOP, the integrated method improved the AUC values by 0.15% and 0.86% on the FF++and Celeb DF-V2 datasets, respectively. Among them, the accuracy and precision were improved by 13.39% and 15.13% on the Celeb DF-V2 dataset, which is more difficult to detect. This reflects the effectiveness of spatiotemporal feature integration for DeepFake detection.

Keywords:Deep forgery#detection;Multimedia security;Combination of time and space; Integrated methods

*通信作者：孟令中　lingzhong@iscas.ac.cn

1 引言

随着互联网和人工智能的发展，数字媒体的传播和演化形式越来越广泛多样。无论是图像和视频的编辑软件，还是各种新兴的 AI 换脸应用，都为人们的社交娱乐提供了更多方式。基于深度学习的图像生成技术不断发展，随之而来的安全问题也不得不让人关注，媒体内容的真实性越来越无法得到保证。近几年流行的基于深度学习的换脸技术 DeepFake[1]在社会上产生了巨大的影响，使用者能够通过深度学习技术将目标人脸替换到任意图像或者视频中，使大众无法分辨内容的真假。伪造的目的可能为了娱乐（例如美颜 P 图、换脸体验），也可能是恶意传播虚假新闻或者制造伪证。而出于后者目的的行为将对社会造成恶劣影响。面对当前人工智能技术的迅速发展，大数据环境下的网络空间安全受到新的威胁与挑战，传统的取证理论已经不足以应对当前的新兴技术。

DeepFake 是指利用深度学习技术合成的伪造内容，可应用于视频、音频等多个方面。对于伪造视频来说，它的生成原理主要指通过自动编码器(Auto Encoder, AE)[2]或者生成对抗网络(Generative Adversarial Network, GAN)[3]等网络模型将目标人脸替换到原视频的人脸上。在大量训练数据的驱动下，伪造视频的质量越来越高，肉眼难以分辨真假，这些伪造视频的传播混淆视听，误导公众判断。早期的 DeepFake 存在“光线不自然”、“面部出现融合边缘”等较为明显的痕迹，但随着深度学习的技术不断发展进步，深度合成的内容变得越来越真实，对于现有检测技术的抵抗攻击越来越强，伪造效果也越来越好，无论是从图像的融合程度上还是帧与帧之间的连接过渡上都做到近乎完美，传统的基于生物特征的鉴别方式越来越难以发挥作用，对检测技术的挑战也越来越大。

2 相关工作

2.1 时空特征结合

对于 DeepFake 检测的研究，在基于图像特征学习的基础上，越来越多学者开始关注视频流中存在的时间信息。即同时提取帧内图像级特征和帧间时间连续性特征，从中寻找伪造的线索。在图 1 中我们列举了部分可用于识别深度伪造的时空特征，接下来将介绍一些基于时空特征结合的检测方法。

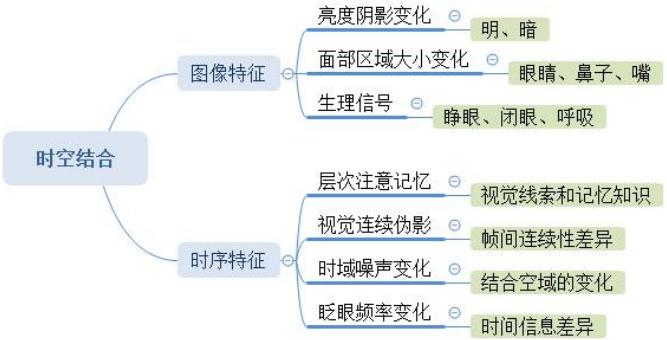


图 1 时空结合检测指标总结

Ekraam Sabir 等人[4]提出了基于循环神经网络(Recurrent Neural Network, RNN)的时空检测网络。首先对视频进行预处理。然后将帧序列输入递归卷积网络，先使用卷积神经网络(Convolutional Neural Networks, CNN)提取图像级特征，随后用 RNN 扩展主干网络，在递归网络中学习帧间特征。同样利用 CNN 结合 RNN 的学习模式，David Guera 等人[5]提出了一种时间感知网络。使用卷积神经网络来提取帧级的特征。然后，将这些特征输入卷积的 LSTM 结构来学习帧间时间不一致性。在时空结合的基础上，Bing Han 等人[6]提出了一个能够处理低质量数据的双流网络。该架构首先将输入视频划分为多个片段，然后将每个片段的选定帧分别输入两个网络：第一个网络以 RGB 信息作为输入，学习语义的不一致性。第二个网络利用由空域富模型(Spatial Rich Model, SRM)[7]提取的噪声特征，建模人脸的篡改部分。将每个片段经过主干深度网络处理得到的二进制分类结果合并，利用时间信息，在最后阶段进行双流融合，得到最终的预测。网络权值在每个段中共享，而不在 RGB 流和 SRM 流之间共享。

此外，不同于单纯的挖掘帧间的时间信息，Yuezun Li 等人[8]发现人类的生理信号在伪造视频与真实视频之间存在差异，其中眼部运动差异较为明显，例如眨眼频率、开合状态等。于是作者将眨眼模式的差异作为检测指标，具体来说，该方法通过循环神经网络学习一段视频内人眼眨眼的概率和运动模式，得到眨眼频率差异，将在一定概率下帧内缺少眨眼作为伪造视频的证据，以此进行检测和分类。

2.2 DeepFake 检测相关原理

在本文提出的集成方法中，所选取的两种检测模型分别基于图像级特征学习和时序特征学习，为了便于理解模型的检测过程，本节将介绍模型中所使用的相关技术原理。

2.2.1 连续子空间学习

连续子空间学习(Successive Subspace Learning, SSL)由 Yueru Chen 等人[9]提出的一种机器学习方法。它由四个关键步骤组成：(1)连续的近远邻域扩展；(2)子空间近似无监督降维；(3)标签辅助回归(Label-assisted regression, LAG)监督降维；(4)特征连接和决策。为了说明 SSL 的原理，文献[9]中基于 SSL 模型提出了一种图像分类方法 PixelHop。实验结果表明，PixelHop 方法在三个基准数据集（MNIST、Fashion MNIST 和 CIFAR-10）上的分类效果均优于模型复杂度相似的经典 CNN 模型。

如图 2 所示为 PixelHop 方法的框架图，该方法由级联的 i 个 PixelHop 单元序列组成，每个 PixelHop 单元的空间位置上的属性以多种形式聚合，然后输入 LAG 单元进行进一步的降维，以生成每个单元的 M 个属性。最后，在模块 3 中将这些属性连接起来，形成维度为 $M \times i$ 的最终特征向量，用于图像分类。其中，模块 1 的作用是通过级联的 PixelHop 单元计算选定像素的近端邻域的属性，连接一个目标像素及其 Ni 邻域像素的属性，形成一个维度为 K_{i-1} 的邻域并集，扩大后的邻域维数等于 $K_{i-1} \times (N_i + 1)$。随后应用子空间近似技术 Saab 变换[10]控制快速增长的维度。模块 2 通过标签辅助回归单元进行聚合和监督降维。

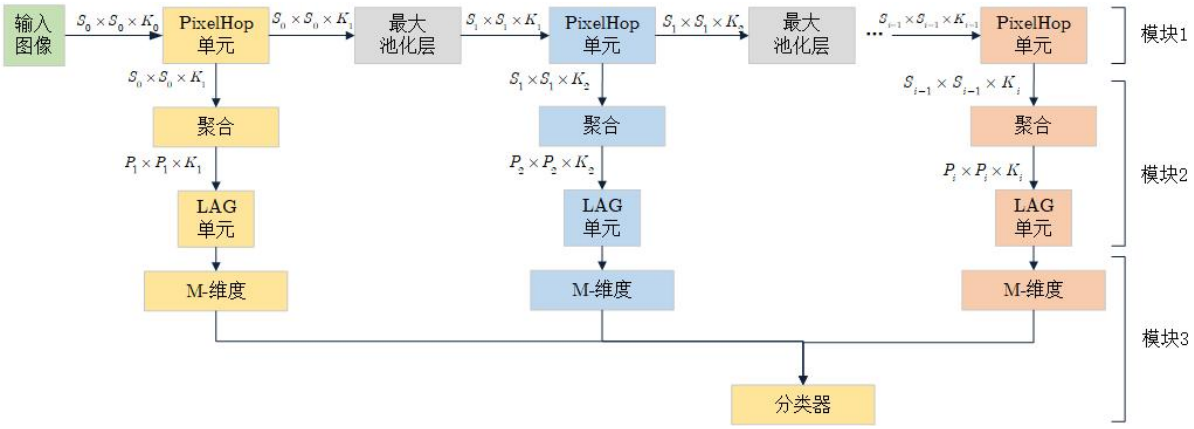


图 2 基于 SSL 的 PixelHop 框架[5]

2.2.2 门控循环单元

门控循环单元(Gate Recurrent Unit, GRU)[11]是循环神经网络的一种，是基于长短期记忆网络(Long Short-Term Memory, LSTM)[12]发展的一种简化变体，通常具有更快的计算速度，且更易训练。它解决了捕捉时间序列里时间距离较大的依赖关系，通过可以学习的门来控制信息流动。GRU 模型的隐藏状态计算模块不引入额外的记忆单元，且将逻辑门简化为重置门(reset gate)和更新门(update gate)，重置门和更新门的输入均为当前时间步输入 X_t 与上一时间步隐藏状态 H_{t-1}，输出由激活函数为 sigmoid 函数的全连接层计算得到。基于 Ekraam Sabir 等人[4]对于伪造视频时序检测特征的研究可以得知，由于伪造操作是逐帧进行的，所以会在图像中留下线索。而将各帧图像中留下的视觉伪影连接起来可能构成帧间连续性的差异，在本文方法中的时序模型便通过 GRU 来学习帧间运动的连续性特征。

3 时空集成检测方法

基于时空特征结合的思想，本节在现有检测模型的基础上设计了一个结合帧内图像特征学习和帧间面

部运动模式提取的集成检测方法，称为 LRHOP(Landmark Recurrent Hop)。基于前期的研究，在本节的方法设计中，以帧级检测模型 DefakeHOP[13]为主干网络，结合帧间连续性检测模型 LRNet[14]作为辅助网络，通过增加面部整体运动模式差异检测的方式来完善基于面部区域特征学习机制的不足，结合时空特征进行分类。

3.1 集成检测框架

DefakeHOP 是基于非深度学习的轻量化模型，该模型的学习机制无需反向传播，具有参数少、易训练的特点。由于其不使用基于深度学习的方法，所以能够很好的抵抗 DeepFake 的对抗攻击。作为帧级检测模型，它通过提取面部区域特征进行学习来实现分类过程，虽然实验表明该模型能够在训练的数据集上具有良好的检测性能，但它没有考虑到图像篡改对面部整体运动的影响。尽管眼睛、嘴巴这些重要区域能够在很大程度上表征人脸的主要特征，但仍不够充分。这是因为在伪造视频的合成中，单帧面部篡改对于视频整体的连续性会产生一定的影响，而这种影响便可能体现在面部整体运动模式的差异上。因此，本文在基于图像特征检测的模型上结合时序模型作为辅助分支，融合两者的预测结果，从而提升检测性能。

整体方法的检测框架如图 3 所示，首先对视频进行数据预处理，提取面部边缘特征点并记录连续坐标，然后分别送入两个分支中；在图像特征检测分支，首先根据面部特征坐标对人脸进行校正对齐，保证头部姿势基本保持一致；然后提取左右眼和嘴巴所在的局部区域，送入特征提取学习模块(PixelHop++)，分别对三个区域进行特征提取、空间降维处理，同时依据处理好的特征进行局部区域的初步信道分类；最后结合同一帧内各个区域的分类结果计算平均值得到该帧预测值，最后连接各帧结果计算整个视频的预测值。在时序特征检测分支，首先通过校准模块对提取的连续面部特征点坐标进行校准，并进行数值的归一化处理；然后根据面部特征坐标分别构建两个向量输入到两个 RNN 网络中，最后将两个网络的分类结果集成，计算平均值。最后融合两个分支的预测结果，根据两个模型训练时的 AUC 值分配预测权重，计算得到最终的预测结果。在后面的小节中我们将对流程中的关键模块进行详细介绍。

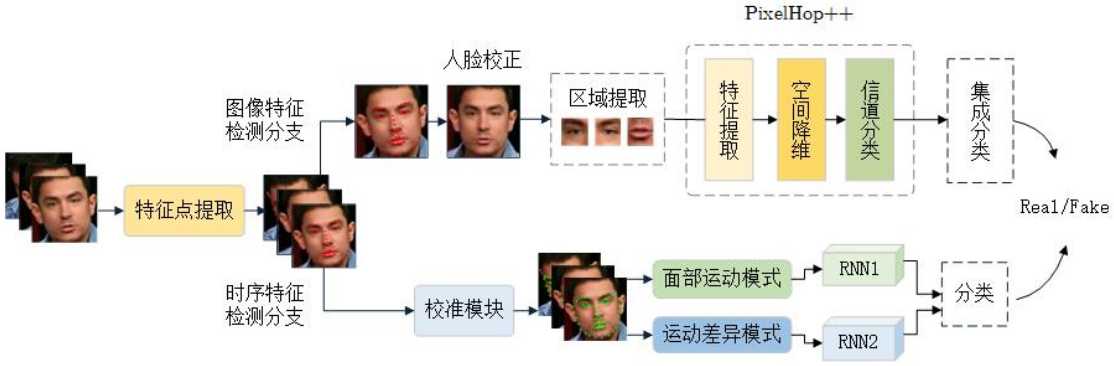


图 3 多指标集成检测框架

3.2 数据预处理

在输入网络之前，首先进行视频的预处理。使用“OpenFace2”开源工具[15]检测人脸并提取包含五官及面部轮廓的最小图像范围，并用圆点定位。于是可以在每张人脸图像上检测到 68 个面部边缘特征点，然后将面部大小调整到 128×128，并旋转到特定的坐标位置，使所有样本的头部姿态和面部大小基本保持一致，以进行对齐和规范化。

接下来，由于两个网络所需的数据不完全相同，所以对数据的处理将分别进行，对于帧内检测，需要进一步从面部的不同区域（包括左眼、右眼和嘴）裁剪出大小为 32×32 的局部图像，作为 PixelHop++模块的输入数据。而基于帧间连续性的检测，需要使用连续的面部特征点位置坐标。然而，目前的面部地标探测器大多是帧级上进行，会存在误差。为此，需要使用一个校准模块来提高地标检测结果的精度。即利用 Lucas-Kanade 光流算法[16]，利用帧与帧之间的连续性来预测每个特征点在下一帧中对应的位置。通过改进的卡尔曼滤波器[17]将预测结果与实际的检测结果合并、去噪，并进行数值归一化处理，以获得更加准确的的地标表示。

3.3 特征提取学习

PixelHop 是一种基于连续子空间学习(Successive Subspace Learning, SSL)的分类学习方法[9]，在此基础上，通过改进的 PixelHop++模块从局部图像中提取特征，该模块由三个级联的 PixelHop++单元组成，每个单元的块大小为 3×3，步长为 1。如图 4 所示，将空间分辨率为 32×32 的面部区域彩色图像作为输入，块的大小和步长可进行更改。第一跳的像素块包含 $3\times3\times K_0=9K_0$ 个变量作为向量，其中 $K_0=3$ 表示 RGB 输入。

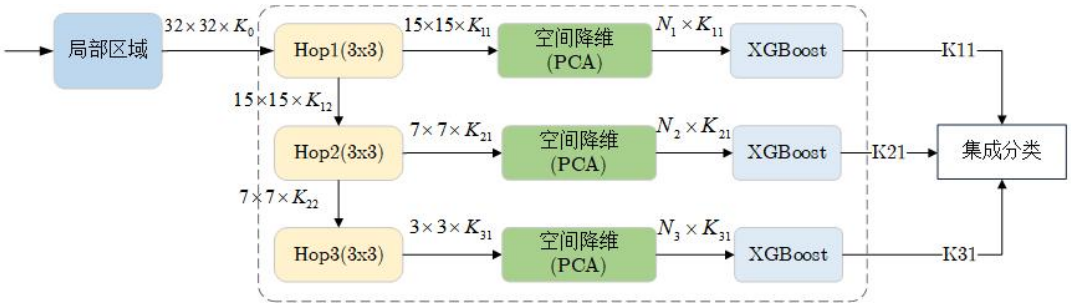


图 4 特征学习模块

在 Hop2 和 Hop3 单元中，利用像素邻域之间的统计相关性，将信道级子空间近似变换(Subspace approximation with adjusted bias, Saab)[10]应用于每个单元的输入向量。Saab 是主成分分析(Principal Component Analysis, PCA)的一个变体，首先将信号空间分解为局部均值和频率分量，然后将主成分分析应用于频率分量，推导出核。每个核代表一个特定的频率选择滤波器。特征值较大的核提取较低的频率分量，而特征值较小的核提取较高的频率分量。通过丢弃特征值非常小的高频分量来实现降维。通过 PixelHop++模块提取特征后，得到一组人脸区域特征集。为了使输出维度更加简洁以便输入 XGBoost 分类器[18]，应用主成分分析进行进一步的空间降维，保留包含输入图像约 90%能量的前 N_1 个 PCA 分量。随后，从第 i 跳中获得了空间维度为 N_i 的 K_{il} 个信道。对每个信道训练分类器学习语义信息，预测每个信道来自伪造视频的概率。

3.4 双流集成分类

在基于帧内图像特征学习的检测分支中，对于每个人脸图像块，结合所有信道的分类预测概率，得到人脸图像块为假的概率。而对于整个视频的预测，首先将多个区域（左眼、右眼、嘴）的预测概率值进行集成。其次进行多帧集成，对于每一帧，将当前帧与其前 3 帧和后 3 帧的预测值结合。

在基于帧间面部运动模式的检测分支中，使用时序分支中经过提取和校准的面部特征点序列构造两种的特征向量，然后输入到一个双流 RNN 中进行时序特征学习与分类。假设每个面部特征点用 $z^{\alpha}=[x^{\alpha},y^{\alpha}]^T$ 表示，因此可以通过特征点 $L_i=[z_i^1,...,z_i^{68}]^T$ 来表示表示面部运动模式的特征向量 $\alpha_i=[x_i^1,y_i^1,...,x_i^{68},y_i^{68}]$，用后一帧的特征点坐标与前一帧的特征点坐标相减得到代表面部运动差异模式的特征向量 $\beta_i=[x_{i+1}^1-x_i^1,...,y_{i+1}^{68}-y_i^{68}]$，用来表示连续帧之间特征点位置的差异。分别将两个向量输入到由 GRU 组成的 RNN 网络中来检测面部运动特征的不连续性。最后计算两个分支的预测平均值作为面部运动模式的预测结果。检测过程可以视作判断视频真假的二分类过程，每个视频的预测得分表示了该视频可能为伪造视频的概率，在训练中真假视频的均匀分布，同时参照一般的二分类模型，实验将 0.5 作为得分阈值。即在每个分支中，预测得分大于等于 0.5 时认为该视频伪造的概率较大，判定其为伪造视频，同理，该值小于 0.5 时认为该视频为真实视频。

为了更加有效的结合两个分支的预测值，根据两个检测方法的检测性能进行权重分配。经过多次实验测试验证，以两个分支单独检测时的 AUC 值占比作为检测权重，得到的融合结果最好。具体实现过程如下，定义图像级检测分支的预测权重为 f_weight，检测 AUC 值为 f_auc；视频级分支的预测权重为 l_weight，检测 AUC 值为 l_auc。根据两个分支单独检测时的 AUC 值占总值的比例进行权重的划分。由于帧内检测分支在各个数据集上的测试 AUC 值均可达到 90%以上，因此为了更好的进行结果对比，在本实验中只对预

测分值在 0.4 - 0.6 区间的视频进行辅助判断，在这个预测分值区间的样本分类结果是存在较大误差可能性的，在这里引入时序分支进行辅助预测，能够帮助帧内检测分支修正预测分值，从而提高检测的准确率。

4 实验讨论与结果分析

4.1 数据集与环境设置

实验使用 FF++和 Celeb-DF-V2 两大 DeepFake 人脸视频公开数据集。其中 FF++[19]包含 1000 个真实人脸视频以及对应的伪造版本。伪造方式包括 Deepfakes、Face2Face、FaceSwap 和 NeuralTextures。每个视频中的人脸大部分以正脸出现，没有遮挡。考虑到不同压缩程度对于检测难度的影响以及在现实应用场景下的普及频率，在本实验中只在 Deepfakes 伪造版本和原始视频的轻度压缩版本(c23)上进行训练和测试。Celeb-DF-V2[20]数据集中的人脸来自 59 位来自不同年龄和种族的明星。相比于 Celeb-DF-V1 和其他伪造数据集，该版本的数据集在数据质量上有了各方面的提升，比如提升人脸分辨率至 256×256、合成边界融合更好、消除抖动使帧与帧之间更加连续。

在原始数据集中伪造样本和真实样本的数量相差悬殊，由概率性质得知，样本分布不均衡可能会导致模型“随机预测”。在实际应用中真假视频的分布情况视具体场景而定，大部分情况下认为真实视频应该多于伪造视频，实验中对样本数量的划分可能会影响对于真实视频的判断，但为了能够均衡地学习到真假视频的特征，便于结果展示，实验中将选取相同数量的真假样本进行训练和测试，视作真假视频为均匀分布。其中在 FF++(DeepFakes)数据集中分别选取 1000 个真实视频和虚假视频作为样本；在 Celeb-DF-V2 数据集中分别选取 590 个真实视频和虚假视频作为样本。在两个数据集上，训练集、验证集与测试集的划分比例均为 8:1:1，具体划分情况如表 1 所示：

表 1 数据集划分情况

数据集	样本名称	真实样本	伪造样本
Celeb-DF-V2	训练集(train)	472	472
	验证集(val)	59	59
	测试集(test)	59	59
FF++	训练集(train)	720	720
	验证集(val)	140	140
	测试集(test)	140	140

实验主要基于深度学习框架 Pytorch 实现，所用显卡为 MEDIA Corporation Device 2206 (rev a1)，系统为 Ubuntu 18.04.1 x86_64，Pytorch 版本为 1.12.1，OpenCV 版本为 4.7.0。

4.2 检测算法评价指标

对于 DeepFake 检测算法的评价，重点需要关注模型的泛化性、检测的准确性和检测效率。在研究各项评价指标前，我们首先将 DeepFake 检测视作一个判断视频真实或伪造的二分类问题，假定真实视频为正类样本(Positive)，伪造视频为负类样本(Negative)，则 TP 表示正确预测真实视频的正样本个数，FN 表示错误预测真实视频的样本个数，FP 表示错误预测伪造视频的样本个数，TN 表示正确预测伪造视频的样本个数。

4.2.1 泛化性指标

准确率(Accuracy, Acc)表示所有预测正确的正样本和负样本占所有样本预测结果的比例，计算准确率如公式(1)所示。对于模型来说，计算得出的准确率越高，表明该模型对样本的分类能力越强。

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

精确率(Precision)表示被正确划分的正样本占所有被预测为正样本的样本比例，计算方式如公式(2)所示。能够体现一个模型对于正样本识别的精确性。

$$precision = \frac{TP}{TP + FP} \tag{2}$$

ROC 曲线是反映敏感性和特异性连续变量的综合指标。而 AUC(Area under the Curve of ROC)表示 ROC 曲线下方的面积，是判断二分类预测模型优劣的标准。AUC 的数值越大，表明进行测试的模型的分类效果越好。

F1 值是精确率和召回率的调和平均数，表示精确率和召回率的综合决策，计算方式如公式(3)所示。

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{3}$$

4.2.2 准确性指标

对准确性的评价，考虑从误报率和伪造检测率两个方面进行评价。

误报率(False Alarm Rate, FAR)，也称虚警率或误检率，表示将正样本误报为负样本的正样本占所有正样本的比例，计算方式如公式(4)所示。误报率值越小，检测的准确性越好。

$$FAR = \frac{FP}{FP + TN} \tag{4}$$

伪造检测率，可以理解为将虚假视频正确识别出来的比例，也就是真阴性率(True Negative Rate, TNR)，又称特异度(Specificity, SPE)，计算方式如公式(5)所示。伪造检测率的值越大越好。

$$TNR = \frac{TN}{TN + FP} \tag{5}$$

4.2.3 效率指标

检测效率(Detection Efficiency, DE)不只是关注检测所花的时间，还要关注检测的准确率，需要在保证准确率的前提下提升速度。因此对检测效率的定义为在单位时间内正确识别视频类别的比例，也就是准确率与检测时间的比值，如公式(6)所示：

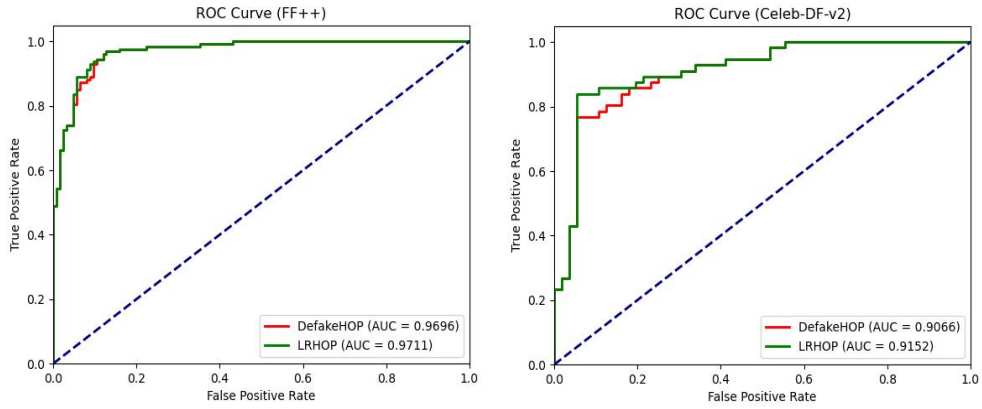
$$DE = \frac{Acc}{t} \tag{6}$$

4.3 实验结果分析

为了检验集成效果，将结合图像特征学习与时序特征提取后的检测方法 LRHOP 与原模型 DefakeHOP 进行对比。并使用 4.2 节的算法评价指标进行计算，从泛化性、准确性和效率三个方面综合评价集成前后的检测效果。

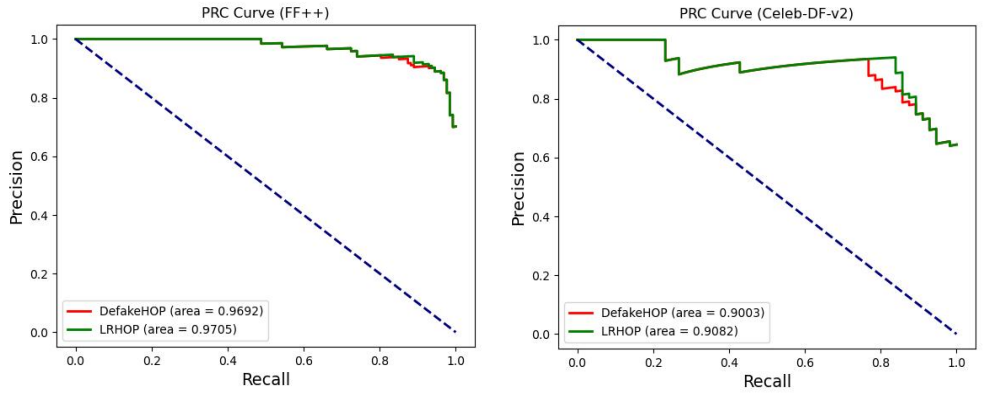
DefakeHOP 模型只考虑到 DeepFake 生成在各帧图像上引入的变化，通过提取面部区域进行特征学习的方式进行分类预测，但却忽略了面部整体运动在帧与帧之间的连续性差异。仅仅依靠人脸中左右眼及嘴部这些关键区域来挖掘篡改痕迹仍然不够充分。而本文提出的方法针对这一问题，在 DefakeHOP 模型的基础上拓展了时序检测模型，将面部整体运动模式差异作为补充指标，完善了基于区域特征的检测方法。

通过对模型集成前后的实验数据对比，表明了增加时序模型检测整体面部运动模式的差异，能够很好的弥补基于区域特征学习检测方法的不足。首先来看泛化性指标，集成前后的 ROC 曲线和 PRC 曲线如图 5 和图 6 所示，其中红色折线表示原模型的 ROC 曲线，绿色折线表示指标集成后的模型的 ROC 曲线。从 ROC 曲线中可以看到集成后的 AUC 值有所提高，说明通过帧间连续性差异的检测结果可以作为帧内图像检测的补充，帮助原模型修正分类结果。



(a) FF++数据集测试结果　　(b) Celeb-DF-v2 数据集测试结果

图 5 RUC 曲线测试对比图



(a) FF++数据集测试结果　　(b) Celeb-DF-v2 数据集测试结果

图 6 PRC 曲线测试对比图

此外，通过计算准确率等相关指标，也可以看到集成后的算法在各项泛化性指标上均有较大的提升，其中准确率和精确率分别提高了 7.42%和 10.09%。进一步体现了时空特征结合的集成方法优于单纯基于图像静态特征学习分类的检测方法，增加时序特征可以帮助对伪造痕迹进行更加全面的挖掘，从而提升检测准确率。

表 2 泛化性指标评价结果

数据集	方法	准确率(ACC)	精确率(Precision)	召回率(Recall)	F1 值
FF++ (DeepFakes)	DefakeHOP	89.68%	89.91%	89.68%	89.67%
	LRHOP	97.10%	100.00%	100.00%	100.00%
Celeb-DF-v2	DefakeHOP	83.04%	83.30%	83.04%	83.00%
	LRHOP	96.43%	98.43%	96.43%	96.43%

在 FF++的测试样本中误报率降低了 0.71%，伪造检测率提升了 2.85%。而在 Celeb-DF-v2 的测试样本中误报率降低了 8.47%，伪造检测率提高了 16.95%。可以看到，在较难检测的 Celeb-DF-v2 数据集中两个指标的提升都较为明显，该比例这也与视频的数量有关。

表 3 准确性指标评价结果

数据集	方法	误报率(FAR)	伪造检测率(TNR)
FF++ (DeepFakes)	DefakeHOP	5.71%	94.29%
	LRHOP	5.00%	97.14%
Celeb-DF-v2	DefakeHOP	11.86%	79.66%

	LRHOP	3.39%	96.61%

通过检测效率评估也可以看到，集成后虽然在检测时间上稍有增加，但检测效率依然提升，这主要是因为检测准确率的提升。不过由于测试样本的数量相对较少，所以在测试时间上的变化并不明显。

表 4 效率指标评价结果

数据集	方法	准确率(ACC)	测试时间(t)	检测效率
FF++ (DeepFakes)	DefakeHOP	89.68%	76.1332s	1.18
	LRHOP	97.10%	76.1372s	1.28
Celeb-DF-v2	DefakeHOP	83.04%	34.6034s	2.40
	LRHOP	96.43%	34.6069s	2.78

5 结论

本文提出了一种时空结合的集成检测方法，通过结合时序特征检测的方式来解决基于区域图像特征检测方法的不足。具体实现为在现有模型的基础上进行拓展，构建基于图像特征和时序特征的双流网络。首先进行视频预处理，然后分别输入基于连续子空间学习的图像级检测分支和基于面部运行模式差异的帧间连续性检测分支，再根据模型性能权重计算最终的融合预测结果。通过集成前后在 FF++和 Celeb-DF-v2 数据集上的实验对比，指标集成后的方法比原模型 DefakeHOP[7]在准确率、精确率、AUC 值等各项评价指标中的数值均有所提升，证明了时空结合集成检测方法的有效性。

参 考 文 献

[1] Mirsky Y, Lee W. The creation and detection of deepfakes: A survey[J]. ACM Computing Surveys (CSUR), 2021, 54(1): 1-41.

[2] Pu Y, Gan Z, Henao R, et al. Variational autoencoder for deep learning of images, labels and captions[J]. Advances in neural information processing systems, 2016, 29.

[3] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks[J]. Communications of the ACM, 2020, 63(11): 139-144.

[4] Sabir E, Cheng J, Jaiswal A, et al. Recurrent convolutional strategies for face manipulation detection in videos[J]. Interfaces (GUI), 2019, 3(1): 80-87.

[5] Güera D, Delp E J. Deepfake video detection using recurrent neural networks[C]//2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS). IEEE, 2018: 1-6.

[6] Han B, Han X, Zhang H, et al. Fighting fake news: two stream network for deepfake detection via learnable SRM[J]. IEEE Transactions on Biometrics, Behavior, and Identity Science, 2021, 3(3): 320-331.

[7] Fridrich J , Kodovsky J .1 Rich Models for Steganalysis of Digital Images[J].IEEE Transactions on Information Forensics and Security, 2012, 7(3):868-882.DOI:10.1109/tifs.2012.2190402.

[8] Xin Yang, Yuezun Li, and Siwei Lyu. 2019. Exposing deep fakes using inconsistent head poses. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 8261 – 8265.

[9] Chen Y, Kuo C C J. Pixelhop: A successive subspace learning (ssl) method for object recognition[J]. Journal of Visual Communication and Image Representation, 2020, 70: 102749.

[10] Kuo C C J, Zhang M, Li S, et al. Interpretable convolutional neural networks via feedforward design[J]. Journal of Visual Communication and Image Representation, 2019, 60: 346-359.

[11] Chung J , Gulcehre C , Cho K H ,et al.Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[J].Eprint Arxiv, 2014.DOI:10.48550/arXiv.1412.3555.

[12] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[13] Chen H S, Rouhsedaghat M, Ghani H, et al. Defakehop: A light-weight high-performance deepfake detector[C]//2021 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2021: 1-6.

[14] Sun Z, Han Y, Hua Z, et al. Improving the efficiency and robustness of deepfakes detection through precise geometric features[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 3609-3618.

[15] Baltrusaitis T, Zadeh A, Lim Y C, et al. Openface 2.0: Facial behavior analysis toolkit[C]//2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018). IEEE, 2018: 59-66.

[16] Lucas B D, Kanade T. An iterative image registration technique with an application to stereo vision[C]//IJCAI'81: 7th international joint conference on Artificial intelligence. 1981, 2: 674-679.

[17] Kalman R E. A new approach to linear filtering and prediction problems[J]. 1960.

[18] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016: 785-794.

[19] Rossler A, Cozzolino D, Verdoliva L, et al. Faceforensics++: Learning to detect manipulated facial images[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1-11.

[20] Li Y, Yang X, Sun P, et al. Celeb-df (v2): a new dataset for deepfake forensics[J]. arXiv preprint arXiv:1909.12962, 2019, 4.

图结构深层语义捕获软件缺陷预测

范文磊[①②] 申情[①②] 陶雪梦[①②] 楼俊钢*[①②]

[①](湖州师范学院 长三角(湖州)智慧交通研究院 浙江 湖州 313000)

[②](湖州师范学院 浙江省现代农业资源智慧管理与应用研究重点实验室 浙江 湖州 313000)

摘 要: 针对目前现有软件缺陷中对源代码表征不充分，缺陷特征难以提取导致预测准确率低的问题。本文提出一种对源文件进行图表示的新方法，称为流增强抽象语法图(Flow-enhanced Abstract Syntax Diagram, FEASD),首先在不破坏语法结构的基础上过滤掉 AST 中重复的节点和边，减少了树中与缺陷无关的冗余信息，并且在裁剪的过程中将树形结构转变成了图形结构，使源文件中不同的代码片段和语法单元可通过相同的语法边建立依赖性，然后通过加入显式控制和数据流边来丰富图中节点的语义和语法信息。因此，FEASD 充分表征了源码中可反映缺陷的隐藏信息。另外在提取缺陷特征时使用图神经网络模型，在得到每个节点向量后，通过引入二阶图池化层聚合节点特征捕获整图信息。为了验证方法的有效性，与目前最新的缺陷预测方法相比，在常用的 PROMISE 数据集上表明，就 F1 分数和 AUC 分别提升了 10.5%和11.4%。

关键词: 软件缺陷预测；源代码表示；抽象语法树；图神经网络；二阶图池化

Deep semantics of graph structure capture software defect prediction

FAN Wenlei[①②] SHEN Qing[①②] TAO Xuemeng[①②] LOU Jungang[①②]

[①](*Yangtze River Delta (Huzhou) Intelligent Transportation Research Institute, Huzhou University, Huzhou* 313000, *China*)

[②](*Zhejiang Provincial Key Laboratory of Intelligent Management and Application of Modern Agricultural Resources, Huzhou University, Huzhou* 313000, *China*)

Abstract: Aiming at the problem that the source code is not sufficiently characterized in the existing software defects, and the defect features are difficult to extract and are insufficient, resulting in low prediction accuracy. This paper proposes a new method for graphical representation of source files, called Flow-enhanced Abstract Syntax Diagram (FEASD), which first filters out duplicate nodes and edges in AST without destroying the syntax structure, reduces redundant information in the tree that is not related to defects, and converts the tree structure into a graph structure in the process of clipping. By enabling different code snippets and syntax units in the source file to establish dependencies through the same syntax edge, and then enriching the semantic and syntactic information of nodes in the graph by adding explicit control and data flow edges, FEASD fully characterizes the hidden information in the source code that can reflect defects. Further, the graph neural network model is used when extracting defect features, and after obtaining each node vector, the whole graph information is captured by introducing the second-order graph pooling layer to aggregate the node features. In order to verify the effectiveness of the method, compared with the latest defect prediction method, the F1 score and AUC are improved by 10.5% and 11.4%, respectively, on the commonly used PROMISE dataset.

Keywords: software defect prediction; **s**ource code representation; abstract syntax tree; **g**raph neural networks; second-order graph pooling

0 引言

在过去的十年中发现软件缺陷预测(Software Defect Prediction, SDP)一直是软件可靠性研究的热点，因为准确的识别出缺陷文件不仅可以合理的分配硬件资源，也可以节约人力资源[1, 2]。近年来随着大代码概念的被提出，软件的规模和复杂度随之增加,这也对软件的高可靠性提出了新的挑战。根据度量元和预测模型的不同可以分为传统的缺陷预测和现有的缺陷预测[3]。其中，现有的缺陷预测不仅解决了人工设

*通信作者: 楼俊钢

loujungang0210@hotmail.com

基金项目: 浙江省杰出青年自然科学基金资助项目(LR20F020002)，湖州师范学院研究生科研创新项目(2023KYCX41)

Foundation Items: Zhejiang Provincial Outstanding Youth Natural Science Funded Project(LR20F020002), Graduate Research and Innovation Project of Huzhou Normal University(2023KYCX41)

计缺陷度量元肤浅的认知,而且引入了更多的信息[4]。目前，据徐等人[5]表明在现有的 SDP 中面临两个核心问题:一是如何结合源码加强缺陷的深度特征表征能力的问题，二是如何构建二元分类神经网络模型解决复杂和多样的缺陷预测问题。

对于问题一用抽象语法树(Abstract Syntax Tree，AST)表征源码最近吸引了众多研究者的兴趣由于独立于编程语言，反映的是源码的语义和语法结构信息[6, 7]，其中王等[8]首次通过 AST 学习源码的语义特征 ，然后李等人[9]获取 AST 向量通过神经网络自动学习缺陷特征。虽然他们的研究表明使用 AST 一定程度可以增强源码缺陷表示[10]，提高缺陷预测的准确率,然而通过以上研究我们发现目前使用 AST 有以下问题:

1) 完整的 AST 具有复杂的树形结构，反映在更多的节点和更深的尺寸上，并且很容易受到代码表示学习中消失的梯度问题的影响。

2) 抽象语法树虽然捕获了源码的语义信息，但无法捕获控制代码行和数据之间的联系。

针对 AST 疲弱的源码表示能力一方面我们在不改变语法结构的规则下过滤掉重复的节点和语法边，并且在裁剪的过程中已改变原有的树形结构转变成图型结构，该图反映的是整个源文件的源码结构，即不同的方法和代码片段可通过相同的语法边建立过程间的依赖性。另一方面我们在新构建的图中加入控制和数据流图的边，最终我们构造的源码表示图不仅包含了语法信息还包括控制流和数据流这 3 种与漏洞密切相关的信息[11]，相比于王等人[12]流增强的语法树本文构造的图在保证源码语法信息的同时保留了更少的节点和边，因此称为源码流增强语法结构图(FEASD)。

另外得益于图神经网络(GNNS)近期成功研究的启发，我们构造的流增强语法结构图是有向无环图结构，本质可以利用 GNNS 学习图的表示，但识别缺陷文件不同于其他任务，源码对位置信息比较敏感比如循环语句中的跳转指令，过早或者过晚跳出循环都将影响循环体内的语句执行导致缺陷。因此我们考虑使用 transformer 的位置编码对图中节点向量化表示， 进一步的为了学习正确的缺陷特征，保留元素拓扑信息，有效地聚合代码元素信息，我们还在 GNNS 中引入二阶图池化达到对不同图层的缺陷和非缺陷表示，定义了用于建立 SDP 模型的图分类任务。

本文的主要贡献可以总结如下:

1. 针对 Java 语言提出一种新的源码表示方法 FEASD，一种图形结构的源码表示，捕获源代码的语法、控制和数据结构，从而解决 SDP 数据表示疲弱的问题。

2. 提出了结合二阶图池化和 GNN 的 GAT_BP 深度神经网络模型进行文件级别的缺陷预测。

3.所提方法在 PROMISE 数据集中的 F1 分数和 AUC 值相比于最新的基于图型结构的方法分别提升了 10.5%和 11.4%。

1 问题定义&背景

1.1 软件缺陷预测问题定义

本文是文件级别的倾向性预测本质上是二分类问题。给定一个源文件 x_i，设置对应的常数标签 y_i，在缺陷预测的问题中 $y_i \in \{0,1\}$，当 $y_i = 1$ 表明该源文件样本中包含一个或多个缺陷， $y_i = 0$ 代表该源文件中没有缺陷。整个的训练数据集为 $D = \{(x_i, y_i)\}_{i=1}^{P}$, P 是总样本数量。另外 $x_i \in R^M$ 表示根据缺陷预测模型 M 度量出的缺陷文件，目标数据集定义为 $T = \{u_i\}_{i=1}^{Q}$， Q 是目标样本的的数量，样本 u_i 也以相同缺陷模型 M 度量。缺陷预测的目标是将 u_i 映射到目标 $\{0,1\}$ 中。因此,SDP 可以看作是一个发现的过程最优映射函数 $f_{SDP}(x)$。

1.2 抽象语法树

AST 通常是由编译器对源码解析出的树状表示，也是生成许多其他代码表达式的基础,被应用在许多软件工程任务中,如代码生成[13]、代码完成[14]和代码克隆检测[12]。对 java 语言本文用 python 第三方工具库提取总共包括 78 种语法节点类型，再结合源代码分词后的 token，树中可以有几十到数千个节点[4]。尤其在源码中存在多个重复的语法结构时，如图 1 所示给出一段简单源码示例，对应的 AST 如图 2 所示。为了凸显树中存在大量的重复结构我们从树的深度和广度两个角度来说明，图中黄色边表示在同一深度下重复比如在源码示例中第三、四行定义的两个变量对应的语法树为 *LocalVariableDeclaration* 节点下部分，可以看出两行代码对应的语法树中只有两个变量名不同其它节点完全相同导致树的广度冗余。同样的冗余还反映在第 8 和第 11 行的 *if...else...* 语句中对两个变量进行加法操作对应在抽象语法树中也只有变量节点不同。图中的绿色边表示在不同深度下对应的出现相同的语法节点，源码中第 5,6 行分别是 *for* 语和 *if* 语句，其中 *for* 循环结构中包括 *if* 判断语句，*for* 循环语句在抽象语法树中的语法节点为 *ForStatement* 他的孩子节点包括 *BlockStatement* 而该节点又包括循环体中的 *if* 语句，*if* 语句中再次出现 *BlockStatement* 节点，对应的语法树如绿色边所示，*IfStatement* 节点的父节点和孩子节点重复，给语法树中不同深度造成冗余。 根据徐[4]等人的发现软件缺陷主要集中在少数语法边上，与缺陷无关的冗余信息占据了训练参数的主要部分，大大增加了模型学习缺陷特征的难度，导致模型的整体偏好发生转变。

```
1:public class Demo {
2: public static void main(String[] args) {
3:  int a=0;
4:  int b=0;
5:  for (int i = 0; i <=100; i++) {
6:   if(i%2==0)
7:   {
8:    a = a+i;
9:   }
10:   else{
11:   b = b+i;
12:   }}}}
```

图 1 源代码示例

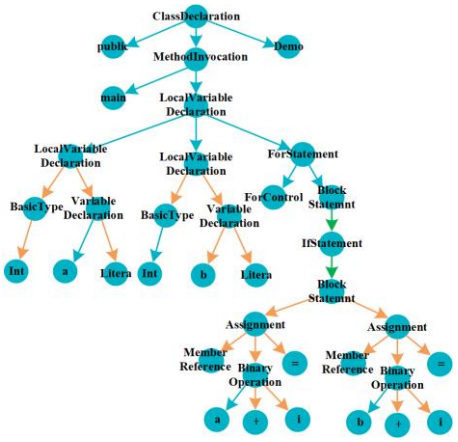


图 2 对应的抽象语法树

1.3 基于图的软件缺陷预测

目前源代码的表示主要来自于从文件粒度或类粒度的统计指标中获得的度量，这要求研究人员离散软件程序片段。这些代码表示被认为是认知上的浅层特征。然而，在许多实际情况下，很难从程序代码中挖掘出更深层次的信息，特别是程序中编码的语义信息，而且数据缺乏明确的线性特征。因此，传统的数据表示在实际的软件项目中并不有效。近年来，随着图神经网络的发展也为深层次的源码表征提供了可能，为了利用源代码丰富的上下文和语法结构，深层捕获程序的语义信息在 SDP 最早提出图结构的是越等人[15]构造的程序执行流程图，马等人[16]提出使用源码中的变量结合数据流向对源程序建模，王等人[12]提出了上下文流图由数据流图和控制流图组成建模，李等人[17]将程序表示成了一个有向的多重图，捕获源码的控制、数据和调用关系并且利用程序依赖图[18]，周等人[19]依据抽象语法树构建了程序的类依赖网络来获取源码的类间的依赖关系。利用图结构提取源码的语义和结构信息近年来取得了很大进展，目前将源程序表示成图形结构可以分为四种类型：控制流图、数据流图、程序依赖图以及基于 AST 和其他图模型的混合[20]。

1.4 图注意力机制

本文中我们用 GAT 来学习语法结构图中不同边的权重值。首先计算计算注意力系数假设节点特征向量 h_i 首先对节点向量做一个增维操作，用可训练的共享参数 W 对 h_i 进行线性变换，把变换后的节点特征拼接在一起然后将高维特征映射到一个实数上公式如下:

$$e_{ij} = \alpha([W\vec{h}_i \| W\vec{h}_j]), j \in \mathcal{N}_i \tag{1}$$

接下来对注意力系数做归一化，就是 Softmax，其中使用 LeakyReLU 作为激活函数具体公式:

$$\alpha_{ij} = \frac{exp(LeakyReLU(e_{ij}))}{\sum_{k\in\mathcal{N}_i} exp(LeakyReLU(e_{ik}))} \tag{2}$$

最后对该节点的邻居节点进行加权求和得到新的节点节点特征向量 $h_i^{'}$ 更新节点特征公式如下:

$$\vec{h}_i^{'} = \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{ij} W\vec{h}_j) \tag{3}$$

另外，为了使模型有更好的拟合能力我们使用多头注意力机制，最后将每个 head 捕捉的特征串联起来最终得到的节点特征表示为:

$$\vec{h}_i^{'} = \|_{k=1}^{K} \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j) \tag{4}$$

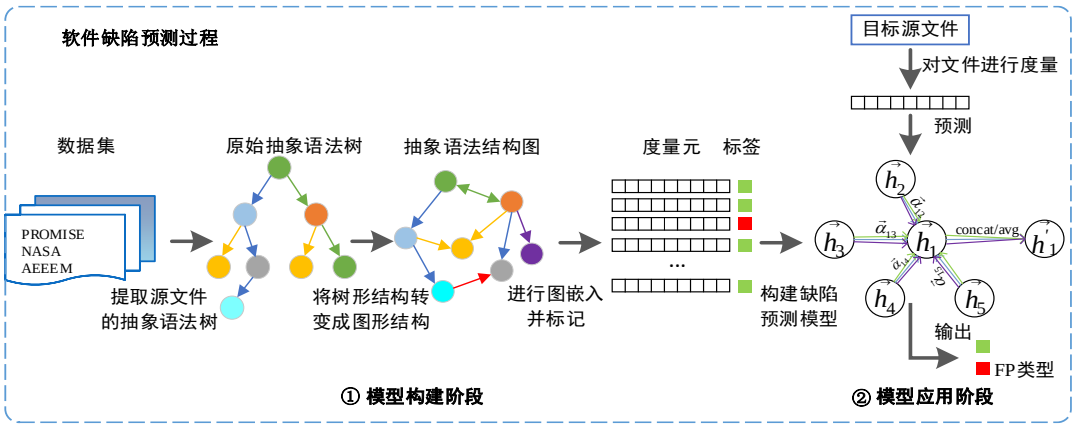


图 3 软件缺陷预测过程

2 方法概述

在本节中，我们将介绍源代码的图结构表示和具体的构图细节，然后是缺陷预测模型的概述和构建过程。整体的缺陷预测过程如图 3 所示。

2.1 AST 的重构过程

目的:在不破坏原语法结构的基础上，过滤掉重复的节点和边提高了每条边的利用率，既一条边可以作为多条链路公用的边。为了验证我们提出的图并没有破坏源码的语法结构信息，我们对比了原始 AST 中包含的路径和图中所有路径发现图中路径集包含了数中的链路集。从图 4 上半部分可以看出定义变量的两行代码生成了 12 个节点和 10 条边，其中只有两个节点和两条边不同，就这两行代码而言生成对应的 AST 节点的重复率达到了 83%,边的重复率 80%。类似的在同一深度中第 8 行和第 11 行代码两个变量的加法运算,对应的语法树如图 4 下半部分面箭头左边所示。它们有着共同的

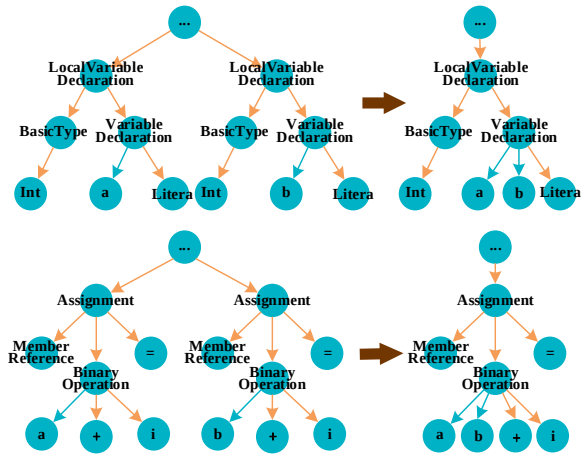


图 4 同一深度下相同的语法结构

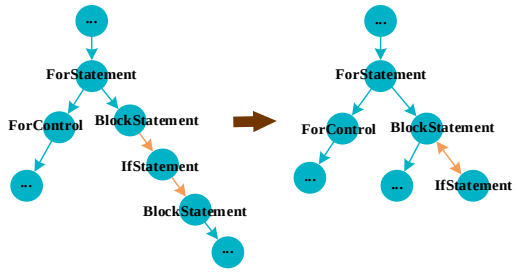


图 5 不同深度下相同的语法结构

父节点 *BlockStatement*，对应节点的重复率 85%边的重复率 83%。为了减少语法重复，我们裁剪掉重复的边，其中箭头右边分别是我们裁剪过后的语法结构形式，可以看到我们不仅过滤掉重复的边，并且保留了不同的节点。第 5,6 行分别是 *for* 语和 *if* 语句，其中 *for* 循环结构中包括 *if* 判断语句，*for* 循环语句在抽象语法树中的语法节点为 *ForStatement* 他的孩子节点包括 *BlockStatement* 而该节点又包括循环体中的 *if* 语句，*if* 语句中再次出现 *BlockStatement* 节点，对应的语法树如图 5 箭头左边所示，黄色的边代表着在不同深度中出现的重复边。同样为了减少节点重复，我们只保留一个 *BlockStatement* 节点，如箭头右边所示我们用一条双向边代替，这样一来就打破了原来的树形结构，变成图形结构。

显示的控制流图和数据流图：关于控制流我们在王等人提出显式控制流图边的基础上又加补充了其他 Java 中出现的所有控制结构。主要包括顺序执行语句、*If* 语句、*While* 语句、*For* 语句、*DoWhile* 语句和 *Switch* 语句。分析这些语句包含的子节点和满足条件的执行过程，具体加入的控制流边如图 5 所示。

为了更加详细的描述构图过程，我们定义以下基本变量和符号。

定义 1(抽象语法树): 本文使用 python 语言的第三方 javalang 库提取输入的整个源文件的原始抽象语法树。源码的 AST 作为 $T=\{t_1,t_2...t_n\}$,其中 n 表示树中的节点数。

定义 2(语法节点集合): 定义语法节点集合为 $S=\{s_{CompilationUnit},s_{ClassDeclaration},...,s_i\}(i\in Node)$，javalang 把源代码的各种元素比如类、属性、方法、代码块、注解等封装成相应的节点对象 *Node*，总共包括 78 个语法节点类型。

定义 3(变量节点集合): 定义变量节点集合 $R=\{r_{VariableDeclaration},r_{MenberReference}\}(R\subseteq S)$，据我们观察变量节点只在语法树中的叶子节点中出现，并且作为变量声明 *VariableDeclaration* 和变量引用 *MemberReference* 的孩子节点。

定义 4(控制节点集合): 控制语法节点集合为 $C=\{c_{IfStatement},c_{WhileStatement},...,c_{DoStatement}\}(C\subseteq S)$ 包括五类控制语法节点，通过控制节点集在语法结构图中加入控制流图边，并且该类语法节点有多个孩子节点我们将选择条件节点 $node_{condition}$ 作为要加入的控制流节点。

定义 5(语法结构图):语法结构图代表了整个源文件的语法结构，由 $G=(V,E)$ 表示 $V=\{v_i\}$ 包含了 AST 中的所有节点的节点。$E=\{e_{(i,j)}|i\neq j\}$ 包含了整个 AST 中不同的边，并且还包括我们额外加入的控制流图边和数据流图边。

生成 ASG 的伪代码如算法一所示。该算法的输入是源文件，输出 ASG 文件。主要包括两个步骤，第一步提取源码的抽象语法树，第二步遍历树中节点并判断所属类型依次加入到图中。具体来说，第一行定义一个空图，第 2，3 行解析源码的 AST 并将树的根节点压入栈中，第 4-20 行是遍历树中节点加入到图中，10-14 行是对控制语法节点和变量节点做出判断分别接入对应的控制流和数据流的边。

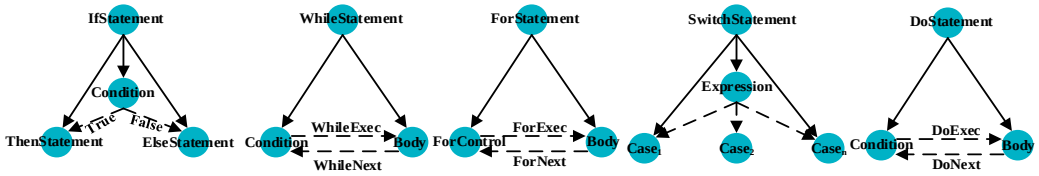


图 6 控制流图边

控制流图的边我们按照王等人[12]提出的流增强抽象语法树(FA_AST)的方法，在此基础上我们又扩展了更多的控制语法包括 *DoStatemen* 和 *SwitchStatement* 并且将这种方法应用到软件缺陷预测任务中，加入的显式控制流图的边主要包括图 6 虚线所示显示了几种基本控制流类型。

算法 1: 源文件的图表示算法

输入: 目标源代码文件: Source_file

输出: 源文件的图形表示

```
G = DiGraph();
T← parse(Source_file);
Stack.append(T.root);
while ! stack.isEmpty() do
    node = Stack.pop();
    for i ← node.child do
        if i ∈ S then
            G.add_edgs(node,i);
            if i ∈ C then
                G.add_edgs(i,i.condition);
            if i ∈ R then
                e = next_edge(i);
                G.add_edgs(i, e);
            end
        else if isinstance(i, list) then
            Stack.append(i);
        else
            G.add_edges(node,i);
    end
end
return G
```

对于数据流图我们根据马等人[16]提出的方法考虑变量间的远程依赖。在遍历 AST 时保留出现在变量集 R 中的孩子节点存入到变量序列中，为了区分不同语法结构中名字相同的变量，我们给每个变量加入上标。在构图过程遇到新的变量节点通过 *find* 函数在变量序列中查找相同变量名且下标靠后的节点，$node_{data_flow}$ 为最终要加入数据流图边的节点。其中每次在边加入到图之前都会判断图中是否存在该边，如果有则跳过。最终我们的语法结构图中包括三种类型的边既原始 AST 中的边以及源码的控制流图和数据流图边。我们打破了以代码块为单位的语法单元而是通过语法边跨代码块连接，提供了更细的缺陷粒度。

2.2 节点嵌入

在得到源码的 FEASD 后，为了使图中节点变成模型可以接收的向量格式，我们用 transformer 的编码器对图中节点进行编码。在图嵌入过程中每个节点向量应保留图的结构和单个节点之间的联系。本文首先对图进行深度优先遍历得到节点序列，然后通过 transformer 的编码器对节点序列编码。相比于其他图嵌入算法（随机游走[21, 22]等，Word2Vec[23]）transformer 考虑了位置信息，这对软件缺陷预测是非常重要的，因为源码对位置信息敏感，代码处在位置不同可能引发缺陷如图 7 所示，左右两边源码中 break 语句所在位置不同，左边 break 后面的语句永远无法执行，虽然他们在 FEASD 中对应的边相同，但生成的节点序列位置不同。因此使用 transformer 编码可以使图中缺陷信息表达更完整,我们也尝试了其他的算法，但就结果来说 transformer 编码效果最好。

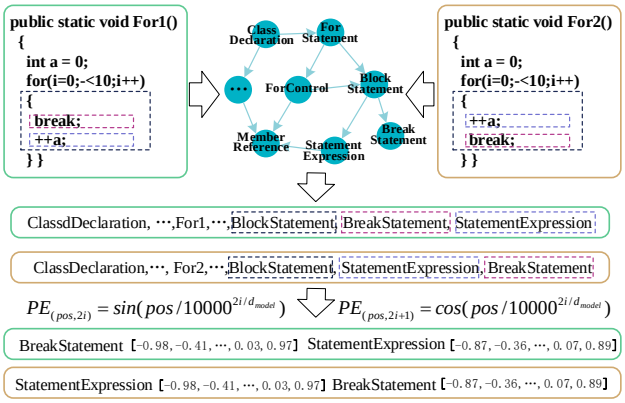


图 7 节点序列位置编码

2.3 模型构建

为了聚合整个文件的缺陷特征，我们在 GAT 的基础上引入二阶图池化。GAT 在聚合了每个节点的邻居信息后，利用二阶图池化从节点表示中学习图的表示。具体来说经过节点嵌入后得节点的特征向量 FEASD 为 G=(A,X),其中 A 和 X 分别表示邻接矩阵和特征矩阵，假设图的节点数为 n，节点特征维度为 d，那么，图注意力机制通过从图的拓扑结构和特征矩阵，为每个节点学习一个 f 维的表示。

$$H = [h_1, h_2, ..., h_n]^T = GAT(A, X) \in R^{n \times f} \quad (5)$$

而图池化，则是通过表示矩阵 H，为全图学习一个 C 维的特征向量。

$$h_G = g(A, H) \in R^C \quad (6)$$

其中 g 是图池化函数。在本工作中，我们提出使用二阶池，也称为双线性池，作为图池。给定 $H = [h_1, h_2, ..., h_n]^T \in R^{n \times f}$ 二阶池化定义为:

$$\mathrm{SOPoo\,L}(H)=\sum_{i=1}^{n}h_i h_i^{T}=H^{T}H\in R^{f\times f} \tag{7}$$

将 $\mathrm{SOPooL}(H)$ 看作一个 f^2 维的图表示向量，只需对矩阵进行展平即可。为了防止参数量过大我们采用双线性映射二阶池的图池化具体来说定义为:

$$SOPooL_{bimap}(H)=SOPooL(HW)=W^{T}H^{T}HW\in R^{f'\times f} \tag{8}$$

其中 $f'<f$ 和 $W\in R^{f\times f'}$ 是表示线性映射的可训练矩阵。之后，我们按照同样的过程对矩阵进行展平，得到一个 f^2 维图表示向量:

$$h_G=Flatten(SOPooL_{bimap}(GAT(A,X)))\in R^{f'^2} \tag{9}$$

最终的缺陷预测模型如图 7 所示主要包括两个模块，第一部分图注意力模块用来学习节点的向量，最终得到节点向量，第二部分是表示双线性映射二阶池化，从节点向量中学习整图的表示，最后得到整图向量。

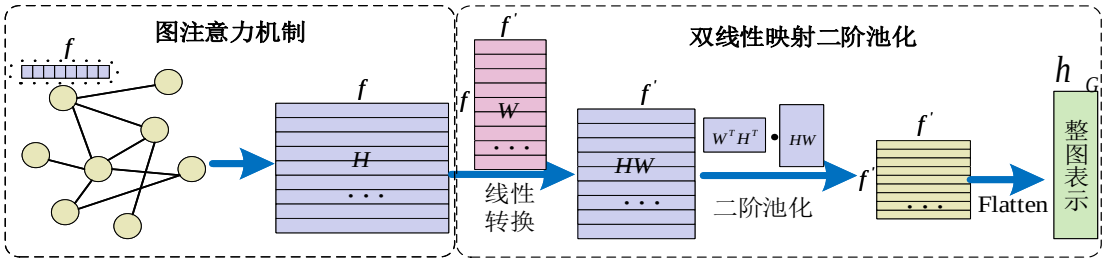


图 8 缺陷预测模型

具体的模型训练过程如算法 2 所示。

算法 2: GAT_BP模型训练过程

输入: 数据样本集 $\mathcal{D}$(X,Y),迭代次数 K,批量大小 B,学习率 α

输出: 缺陷预测模型 $f_{SDP}(x)$

```
初始化模型参数Θ= ( vec(•), W, b )
for k=1 to K do
    for i=1 to B do
        H ← GAT(X_i^A, X_i^F);
        Z ← Softmax( Flatten(W^T H^T HW));
        loss = Loss_function(Z, Y);
        反向传播计算偏导数 ∂J(Θ)/∂Θ;
        更新参数 Θ ← Θ − α ∂J(Θ)/∂Θ;
    end
end
return f_SDP(x)
```

我们的预测模型表示为函数，样本数据集和超参数 K、B、α 以及 Θ 作为输入，首先初始化模型训练参数算法 2 中的第 1 行，3-8 行利用前向和反向传播一次更新 Θ，在经过 K 次迭代后输出最终的缺陷预测模型 $f_{SDP}(x)$。

3 实验与分析

3.1 实验细节

我们进行了一系列的实验来评估本文所提方法，并将其与一些现有的先进方法进行了比较。所使用的操作系统为 CentOS 8.5.2111 64 位，CPU 型号为 Intel(R) Core(TM) i9-10980XE @ 3.00GHz，内存为 96GB，模型的加载以及梯度计算任务由一张 3090(24GB)显卡执行。

3.2 数据集

本研究采用的数据集是从 PROMISE 库中选择了 8 个真实项目来评估我们方法的先进性，这些数据集在软件缺预测中已被广泛应用[6],这些项目属于不同的应用程序域，可以从 Apache 公开获得。表一显示了八个项目的统计信息。为了为缺陷预测收集足够的数据，我们将每个项目的所有不同版本作为一个单独的数据集连接起来。

表 1 实验中使用的 PROMISE 数据集

项目名称	项目版本	平均文件数	平均缺陷文件数	缺陷率	总节点数	总边数
Ant	1.0,2.0,3.0	450	95	21.2	40500	84000
Camel	1.6,1.7	528	124	23.4	46464	90752
Velocity	1.4,1.5	298	103	34.7	30694	77548
jEdit	3.2,4.0	197	84	24.5	16548	57580
Lucene	2.0,2.2,2.4	332	158	52.6	29880	61538
Poi	1.5,2.0,3.0	401	209	61.7	30401	67144
Xalan	2.4,2.5,2.6	630	252	40.1	56070	126940
Synapse	1.1,1.2	227	72	31.6	21565	48375

表 1 列出了每个项目的版本号、每个版本的平均源文件数量，以及每个项目的平均缺陷率。另外项目的缺陷率最小的为 21.2%，最大值为 61.7%，为了解决有些项目的缺陷率过低引起的类不平衡问题，本文采用了 SMOTETomek 抽样来解决数据不平衡的问题，其中周等人[19]已证实用该方法提升了软件缺陷预测的准确率。

3.3 评价指标

在本实验中我们选择的准确性(accuracy)和 F1 度量(*F1-measure*)[24]评估测试集上的预测结果。假设任务 K 类准确率的定义是:

$$accuracy = \frac{\sum_{i=1}^{K} TP_i}{N} \tag{9}$$

由于软件缺陷预测本质上是二分类问题，因此 K 的值为 2，TP_i 指的是预测样本为真的情形，N 是样本的总数。

对于二分类任务 F1 值的定义如下:

$$F1-score = \frac{2 * precision * recall}{precision + recall} \tag{10}$$

$$precision = \frac{TP}{TP + FP} \tag{12}$$

$$recall = \frac{TP}{TP + FN} \tag{13}$$

TP 是真阳性，FP 是假阳性，FN 是假阴性。

3.4 研究问题

在本研究中，我们设计了两个研究问题来评估我们提出的源码图结构表示方法在软件缺陷预测中的有效性，和图结构深层语义捕获缺陷预测的有效性。

1. 在现有的图神经网络中哪一种模型更适合提取缺陷特征，引入的二阶图池化是否对结果有影响？
2. 提出的方法相比于树形结构和最近的图型结构总体性能如何？

为了回答问题 1：该实验的目的旨在找出哪种图神经网络能更好的提取缺陷特征进行预测，虽然在整体框架图中我们用 GAT 作为预测的图神经网络，但为了证明该图模型具有最好的预测性能，我们选择四种典型的图神经网络模型包括常见的图卷积网络(Graph Convolutional Network, GCN),图同构网络(Graph Isomorphism Network, GIN)、简化图卷积网络(Simple Graph Convolution, SGC)、和 GAT 做对比，这四种 GNNs 有着相同的结构，但不同的邻域聚合方法。在每次的节点特征更新后，我们添加了二阶线性映射图池化层。为了简化我们的建模，每个隐藏层都有相同数量的神经元。根据实验结果确定了分层数和神经元数。一旦邻域聚合完成，将使用池化层和 Softmax 函数来预测样本的标签，并使用 Adam 优化算法相应地最小化交叉熵损失来训练学习模型。所有 GNN 模型都基于 Python3.8 和 PyTorch geometry 中实现。

回答问题 2 我们将本模型与当前典型的缺陷预测方法进行了比较，根据其采用的源代码模型可以分为三种类型。

第一种使用静态度量指标表征缺陷特征，使用 PROMISE 的给出的静态指标[25, 26]，使用机器学习的方法构建缺陷模型，包括 Adaboost 算法、多层感知机(MLP)。之前的研究已经证明，这些分类器往往会取得更好的结果[27, 28]。

第二种基于源码 AST 的语义和语法特征，使用序列神经网络模型来提取缺陷特征进行预测，在本实验中我们选择经典的 DBN[29]和 TreeLSTM[30, 31] 的网络。具体来说，以上两种神经网络模型都是基于抽象语法树的树形结构来提取源码的缺陷特征。并且以上两种研究中和我们使用相同的数据集这样我们可以直接将以上工作和本研究比较，为了公平起见我们在对比的时候完全按照别人工作中的实验要求进行。

第三种在 AST 的基础上将树形结构转变成图型结构，在最近的一些软件缺陷预测中集中于将源程序表示为图型结构，并采用基于图的学习方法从图中提取缺陷特征。具体来说我们选择赵等人[32]提出的源代码模型嵌入层次依赖关系通过将 AST 和 CFG 集成在一起来表示嵌入层次依赖关系的源码图型结构本文我们称作 CFG_AST。此外，我们还选择 FA_AST 表示显式流扩充的源码图型结构，以上两种图结构都是在源码 AST 的基础上加入控制流图和数据流图的边构建源码深层模型。这和本实验将源码表征为语法结构图十分吻合，以此做为基准实验更具比较性。其中我们根据论文中公开的源码复现了 CFG_AST 的实验结果，另外将软件缺陷预测作为 FA_AST 下游任务。

3.5 结果分析

3.5.1 回答问题 1

为了回答问题 1 我们统计了不同图神经网络的预测性能。实验结果见图 8，对图同构神经网络、图卷积网络、简化图卷积网络、GAT 的 F1 分数评价指标进行了比较来评估模型的性能，对于 F1 分数，这是精度和召回之间的权衡。同时我们将实验超参数学习率为 0.001，卷积层数为 3，隐藏层大小为 200，dropout 为 0.5,epoch 为 200。从图 9 右边中可以清楚看出 4 个 gnn 的 F1 整体上没有太大的变化。但在八个数据集中，GAT 的 F1 分数在 *Ant(0.69)*、*Camel(0.71)*、*jEdit(0.69)*、*poi(0.79)*、*Xalan(0.72)*五个数据集中高于其它三种图神经网络。图 9 左边的箱型图更是直观反映了不同 CNN 的 F1 分数分布情况，跟基准模型相比 GAT 的平均 F1 值高于性能最差的 SGC4%，从各个箱形图的中位数和上下四位数的间距也可以看出，GAT 的 F1 分数分布是比较的对称，SGC、GCN 和 GIN 有明显的不平衡。 分析原因注意力机制让模型更加关注与当前节点相关的邻居节点，这在软件缺陷预测中是非常关键的，因为编程语言对语法严格要求更加注重节点间的连接性，不仅如此还可以更充分地处理局部信息比如某变量的上下文依赖，从而提高模型的准确性。因此最

终将 GAT 作为本方法中缺陷特征提取和分类的神经网络模型。最终实验表明，我们提出的源码语法结构图可以用来对源码进行深层表示，并在不同的图神经网络中取得了令人满意的结果。

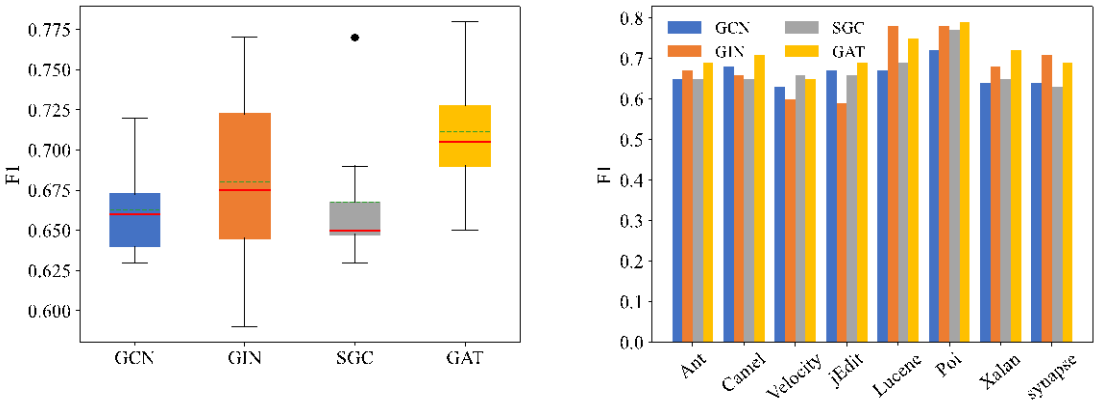


图 9 不同 CNN 的 F1 值比较(左)整个分布(右)八个项目的详细分布

为了验证我们引入的二阶图池化对实验结果的影响，即当用图神经网络学习得到节点向量后，用哪种方式可以更好的表示图向量。在本实验中我们选择了四种方式进行图层池化方法包括对图中节点求最大值(MAX)、求平均值(MEAN)以及使用全连接层(Fully Connected Layer，FC),图 10(右)是八个数据集在不同图池化的整体性能，可以看出使用二阶图池化有最高的中值 0.69,max 为 0.63,mean 为 0.65,FC 为 0.67,说明我们加入的二阶图池化模型性能确实优于基准模型。从图 9 左边发现，虽然二阶图池化并不是在所有的数据集中表现最佳例如在数据集 *Velocity*、*Poi*、*Synapse* 的 *F1* 值低于 FC，但不得不承认二阶图池化的总体性能要优于其他几种方法。我们将这一实验结果归结为两点一方面在进行最大池化或平均池化操作时，会丢失一些原始特征的信息尤其是在进行最大池化时，只会保留区域内最大的特征值，其他特征值都会被舍弃，可能会影响模型的性能。另一方面，最大池化和平均池化都是对区域内的特征进行操作，对于不同的位置和尺度可能会得到相同的输出结果，从而丧失了对位置和尺度的敏感性，全连接层中的权重参数很容易过拟合，从而导致模型在测试数据上的性能下降。最终结果表明二阶图池化在缺陷预测中可以更好的聚合图中缺陷特征，提高了模型的性能和鲁棒性。

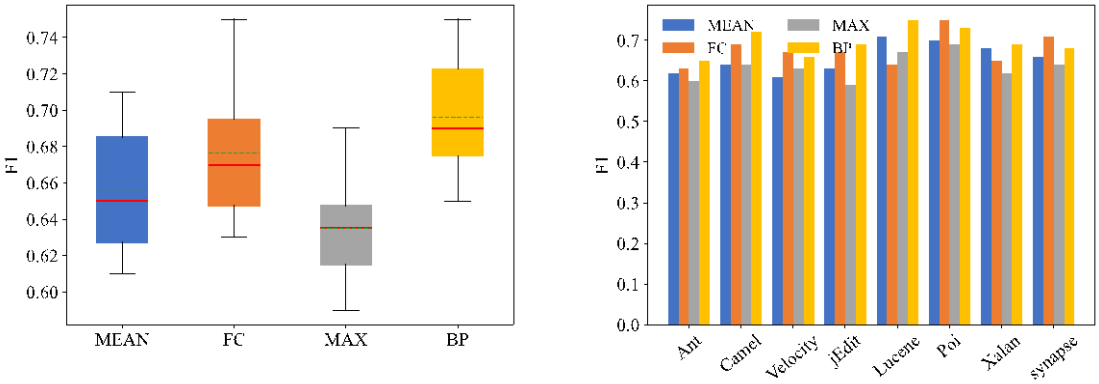


图 10 不同 CNN 的 F1 值比较(左)整个分布(右)八个项目的详细分布

3.5.2 回答问题 2

表 2 显示了不同源码度量方法在缺陷预测中的结果，我们用 AUC 和 F1 两个指标评价预测性能，其中最好的结果加粗突出显示。在得到每个数据集后我们根据 PROMISE 提供的缺陷指标，把所有 bug 数大于 1 的源文件标记为有缺陷，将 bug 数为 0 的标记为无缺陷，因此虽然我们和选择的基准实验里的项目相同，但可能导致实际使用的数据集分布和之前的研究不同。为了保证实验的公平性我们重新实验了上述方法，并且每种方法都运行多次，保持最好的结果。从表 2 的平均度量可以看出通过对源码表征提取缺陷特征整体性能要比传统的使用静态度量好，在基于源码静态度量中 AUC 和 F1 的平均值较高的是 Ad 算法分别为 57.35%、47.78%。结合源码深层语义和语法的 AUC 值平均值较高的为 TreeLstm 网络 64.88%，平均 F1 较高的为 DBN 网络 59.93%，具体来说，较高的 AUC 往往意味着模型对预测结果有更大的置信度。

而在对源码的图形表征中，我们提出的方法与 CFG_AST 和 FA_AST 相比就平均 F1 的值分别提升了 10.5%和 8.7%，在 AUC 上分别提高了 11.4%和 8.03%。具体来说，这种显著的改善可以归结对源码结构的表示差异，这可以分为三个方面。首先，是对整个源文件语法结构更细粒度的表示。根据

CFG_AST 的开源实现，首先将 AST 分成单独的语法块，然后通过 CFG 来连接每个语法块。第二，对图中节点编码的过程。FA_AST 首先建立节点词汇表，然后使用 Word2vec 对节点进行嵌入，而在本实验中我们通过 transformer 来学习特征。第三，学习图的过程，CFG_AST 使用 AGN4D 来捕获图中的依赖特性，但并没有很好的聚合整图信息，而我们通过引入二阶图池化的方法从节点向量中学习整图的表示弥补了以上缺陷。综上所述，本实验的方法在软件缺陷预测任务中具有较好的性能。

表 2 不同方法在 PROMISE 的实验结果

项目名称	源码的静态度量指标				基于树形结构的表征				基于图形结构的表征					
	Ad		MLP		DBN		TreeLstm		CFG_AST		FA_AST		Our	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
Ant	48.1	42.6	52.8	48.4	28.6	32.7	66.8	67.8	51.8	53.5	69.6	58.6	**70.1**	**69.7**
Camel	58.9	38.2	43.4	35.8	59.3	55.0	60.1	58.2	60.1	56.0	66.1	66.7	**68.6**	**67.1**
Velocity	73.9	**66.1**	39.8	46.3	49.2	47.5	68.7	60.0	72.9	67.7	70.8	63.2	**75.6**	64.6
jEdit	52.9	57.3	36.5	24.2	66.0	63.3	71.3	46.1	67.2	65.3	60.0	55.4	**72.0**	**68.9**
Lucene	47.2	33.6	66.8	64.0	68.0	69.1	65.0	62.2	70.6	72.9	68.9	67.0	**79.3**	**75.3**
Poi	61.8	66.5	63.0	71.4	77.4	76.0	59.2	61.7	**79.0**	**83.2**	61.3	62.8	78.8	76.0
Xalan	59.3	37.6	48.1	49.4	70.2	68.3	60.0	58.8	45.7	33.9	72.2	67.0	**77.8**	**73.6**
Synapse	56.7	40.3	55.2	39.6	**73.4**	67.5	67.9	63.4	59.2	50.0	63.1	56.4	70.0	**69.2**
AVG	57.35	47.78	50.7	47.39	61.51	59.93	64.88	59.78	63.19	60.31	66.5	62.14	**74.53**	**70.8**

另外为了评估 GAT_BP 分类模型性能我们画出来不同数据集中测试集的 ROC 曲线。ROC 曲线被广泛应用于二分类模型性评估中，以真阳性率(True Positive Rate)为纵坐标，假阳性率(False Positive Rate)为横坐标绘制的曲线。其中，曲线下的面积代表 AUC 值,越大代表了分类器的性能越好，由图 11 可以看出在八个数据集中 auc 最大的 Lucene0.82，最小的 Camel0.67，最好的缺陷预测阈值在 0.3-0.5 之间，并且整体的 ROC 曲线波动范围较小，进一步的说明模型的泛化性较好。

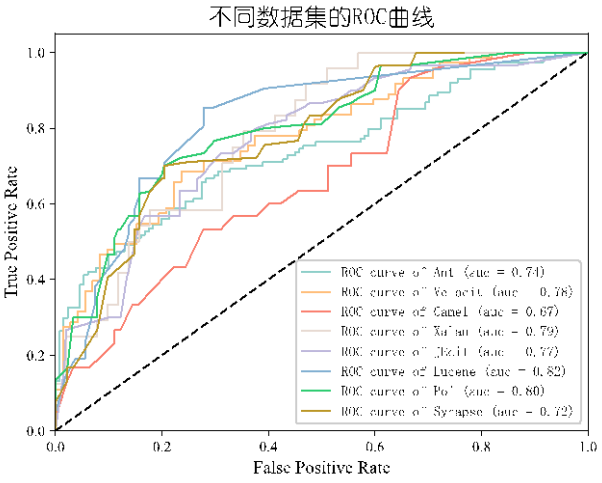


图 11 不同数据集的 ROC 曲线

4 结语

在本文中我们提出了一种新的方法来表征源码深层语义和结构特征，通过图神经网络学习源码中潜在的缺陷表示。我们提出的方法与现有方法相比最大的不同是将源码的树形语法结构转换成了图形结构。相比于规模大复杂度高的树形结构，我们构造的语法结构图更加简洁，表现在有更少的节点和边，在不破坏原有语法结构的基础上提高了每条边的利用率。我们通过 GAT 聚合了每条边的邻居信息，并用双线性二阶图池化从节点向量中学习整图的表示。最后在目前被广泛使用的 PROMISE 数据集上实验的结果表明，本方法提高了软件缺陷预测的准确率和 F1 分数。

为了验证本方法面对大规模源码文件的有效性，我们通过模型训练时间对本文所提方法进行量化分析结果如图 12 所示，按照数据量的大小对八个数据进行排序，在相同的量级下我们的方法相比于 CFG_AST 时间成本减少了 6~8 倍，相比于 FA_AST 节省了 3~4 倍，并且随着文件数增加时间增幅较小。说明在面对更大量级的数据时，本方法时间成本较低。

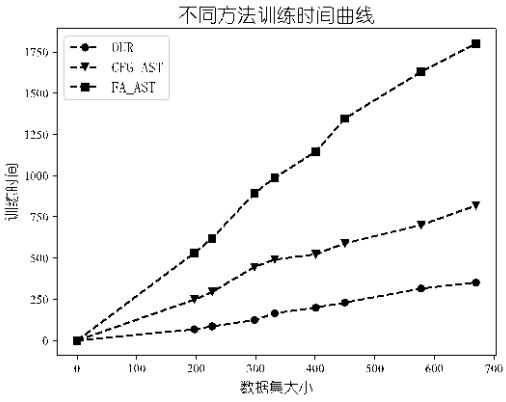


图 12 不同方法训练时间曲线

在未来，我们计划将本方法应用到跨项目缺陷预测中，探索在不同的程序之间表达源代码的语义和结构特征，这样可以捕获更多的缺陷特征。另外我们也将会从第三方版本控

制平台中搜集缺陷数据集，将本方法适用用于不同环境和情形的缺陷预测。

参考文献

[1] WONG W E, DEBROY V, SURAMPUDI A, *et al*. Recent Catastrophic Accidents: Investigating How Software was Responsible [C]. 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement, Singapore, 2010: 14-22. doi: 10.1109/SSIRI.2010.38.

[2] WONG W E, LI X, and LAPLANTE P A. Be more familiar with our enemies and pave the way forward: A review of the roles bugs played in software failures[J]. *Journal of Systems and Software*, 2017, 133(6): 68-94. doi:10.1016/j.jss.2017.06.069.

[3] 宫丽娜, 姜淑娟, 姜丽. 软件缺陷预测技术研究进展[J]. 软件学报. 2019, 30(10): 3090-3114. doi: 10.13328/j.cnki.jos.005790.
GONG Lina, JIANG Shuli, JIANG li. Research Progress of Software Defect Prediction[J]. *Journal of Software*, 2019, 30(10): 3090-3114. doi: 10.13328/j.cnki.jos.005790.

[4] XU Jiaxi, WANG Fei, AI Jun. Defect Prediction With Semantics and Context Features of Codes Based on Graph Representation Learning[J]. *IEEE Transactions on Reliability*, 2021, v 70(2): 613-625. doi: 10.1109/TR.2020.3040191.

[5] XU Jiaxi, AI Jun, LIU Jingyu, *et al*. ACGDP: An Augmented Code Graph-Based System for Software Defect Prediction[J]. *IEEE Transactions on Reliability*,2022, 71(2): 850-864. doi: 10.1109/TR.2022.3161581.

[6] CHEN Deyu, CHEN, Xiang, LI Hao, *et al*. DeepCPDP: Deep Learning Based Cross-Project Defect Prediction[J].*IEEE Access*, 2019, 7(7): 184832-184848. doi: 10.1109/ACCESS.2019.2961129.

[7] GUAN Zhibin, WANG Xiaomeng, XIN Wei, *et al*. A Survey on Deep Learning-Based Source Code Defect Analysis[C]. 2020 5th International Conference on Computer and Communication Systems (ICCCS), Shanghai, China, 2020: 167-171. doi: 10.1109/ICCCS49078.2020.9118556.

[8] WANG Song, LIU Taiyue, NAM Jaechang. Deep Semantic Feature Learning for Software Defect Prediction[J]. *IEEE Transactions on Software Engineering*, 2020, 46(12): 1267-1293. doi: 10.1109/TSE.2018.2877612.

[9] LI Jian, HE Pinjia, ZHU Jieming, *et al.* Software Defect Prediction via Convolutional Neural Network[C]. 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, 2017: 318-328. doi: 10.1109/QRS.2017.42.

[10] LIU Wangshu, ZHU Yongteng, CHEN Xiang, *et al*. S2 LMMD: Cross-Project Software Defect Prediction via Statement Semantic Learning and Maximum Mean Discrepancy[C]. 2021 28th Asia-Pacific Software Engineering Conference (APSEC), Taiwan, China 2021: 369-379. doi: 10.1109/APSEC53868.2021.00044.

[11] 段旭, 吴敬征, 罗天悦，等. 基于代码属性图 及注意力双向 LSTM 的漏洞挖掘方法[J]. 软件学报. 2020, 31(11): 3404-3420. doi: 10.13328/j.cnki.jos.006061.
DUAN Xu, WU Jingzheng, LUO Tianyue, et al. Vulnerability Mining Method Based on Code Property Graph and Attention BiLSTM[J]. *Journal of Software*, 2020, 31(11): 3404-3420. doi: 10.13328/j.cnki.jos.006061.

[12] WANG Wenhan, LI Ge, MA Bo. Detecting Code Clones with Graph Neural Network and Flow-Augmented Abstract Syntax Tree[C]. 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, Canada, 2020: 261-271. doi: 10.1109/SANER48275.2020.9054857.

[13] RIGONI D, NAVARIN N, SPERDUTI A. Conditional Constrained Graph Variational Autoencoders for Molecule Design[C]. 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 2020: 729-736. doi: 10.1109/SSCI47803.2020.9308554.

[14] KIM Seohyun, ZHAO Jinman, TIAN Yuchi, *et al*. Code Prediction by Feeding Trees to Transformers[C]. 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, Spain，2021: 150-162. doi: 10.1109/ICSE43902.2021.00026.

[15] KARTCHNER David, CHRISTENSEN Tanner, HUMPHERYS Jeffrey. Code2Vec: Embedding and Clustering Medical Diagnosis Data[C]. 2017 IEEE International Conference on Healthcare Informatics (ICHI), Park City, USA, 2017: 386-390. doi: 10.1109/ICHI.2017.94.

[16] ALLAMANIS M, BROCKSCHMIDT M, KHADEMI J.Learning to Represent Programs with Graphs[J]. *arXiv preprint arXiv*, 2017. doi: 10.48550/arXiv.1711.00740

[17] LI Yi, WANG Shaohua, NGUYEN Tien, *et al*. Improving bug detection via context-based code representation learning and attention-based neural networks[J]. *Proceedings of the ACM on Programming Languages*, 2019, 3(10): 0-30. doi.org/10.1145/3360588.

[18] HOTTA Keisuke, HIGO Yoshiki, KUSUMOTO Shinji. Identifying, Tailoring, and Suggesting Form Template Method Refactoring Opportunities with Program Dependence Graph[C]. 2012 16th European Conference on Software Maintenance and Reengineering, Szeged, Hungary, 2012: 53-62. doi: 10.1109/CSMR.2012.16.

[19] ZENG Cheng, ZHON ChunYing, LV ShengKai, *et al*. GCN2defect : Graph Convolutional Networks for SMOTETomek-based Software Defect Prediction[C]. 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), Wuhan, China,2021: 69-79. doi: 10.1109/ISSRE52982.2021.00020.

[20] URI Alon, MEITAL Zilberstein, OMER Levy, *et al*. A General Path-Based Representation for Predicting Program Properties[C]. Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, Philadelphia, USA, 2018: 404-419. doi: 10.48550/arXiv.1803.09544.

[21] ADITYA Grover, JURE Leskovec. node2vec: Scalable Feature Learning for Networks[C]. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA,2016: 855-864. doi: 10.48550/arXiv.1607.00653.

[22] BRYAN Perozzi,RAMI Al-Rfou, STEVEN Skiena. DeepWalk: Online Learning of Social Representations[C]. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2014: 701-710. doi: 10.48550/arXiv.1403.6652.

[23] MIKOLOV Tomas,CHEN Kai. Efficient Estimation of Word Representations in Vector Space[C]. 2013 International Conference on Learning Representations, London, Canada, 2013. doi: 10.48550/arXiv.1301.3781.

[24] LIU Ying,LON HanTong,SUN Aixin. Imbalanced text classification: A term weighting approach[J]. *Expert Systems with Applications: An International Journal*, 2009,36(1): 690-701. doi: 10.1016/j.eswa.2007.10.042.

[25] MENZIES Tim, GREENWALD Jeremy, FRANK Art. Data Mining Static Code Attributes to Learn Defect Predictors[J]. *IEEE Transactions on Software Engineering*, 2007, 33(1): 2-13. doi: 10.1109/TSE.2007.256941.

[26] TIM Menzies, ZACH Milton, BURAK Turhan. Defect prediction from static code features Current results, limitations, new approaches[J]. *Automated Software Engineering*, 2010, 17(4): 375-407. doi: 10.1007/s10515-010-0069-5.

[27] GHOTRA Baljinder, MCINTOSH Shane, HASSAN Ahmed. Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models[C]. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy, 2015:789-800, doi: 10.1109/ICSE.2015.91.

[28] TANTITHAMTHAVORN Chakkrit,MCINTOSH Shane, HASSAN Ahmed. An Empirical Comparison of Model Validation Techniques for Defect Prediction Models[J]. *IEEE Transactions on Software Engineering*, 2017, 43(1): 1-18, doi: 10.1109/TSE.2016.2584050.

[29] WANG Song, LIU Taiyue, TAN Lin. Automatically learning semantic features for defect prediction[C]. Proceedings of the 38th International Conference on Software Engineering, Austin, USA, 2016: 297-308. doi: 10.1145/2884781.2884804.

[30] DAM HoaKhanh, PHAM Trang, NG Shien Wee, *et al*. Lessons Learned from Using a Deep Tree-Based Model for Software Defect Prediction in Practice[C]. 2019 IEEE/ACM 16th Internation

al Conference on Mining Software Repositories (MSR), Montreal, Canada, 2019: 46-57. doi: 10.1109/MSR.2019.00017.

[31] HUANG Jingdong, GUAN Xin, LI Siyu. Software Defect Prediction Model Based on Attention Mechanism[C]. 2021 International Conference on Computer Engineering and Application (ICCEA), Kunming, China, 2021: 338-345. doi: 10.1109/ICCEA53728.2021.00073.

[32] ZHAO Zhehao, YANG Bo, LI Ge, et al. Precise Learning of Source Code Contextual Semantics via Hierarchical Dependence Structure and Graph Attention Networks[J]. *Journal of Systems and Software*, 2022, 184(1): 0-13. doi: 10.1016/j.jss.2021.111108.

采用低精度信号源和低精度 ADC 测量低失真放大器的频谱性能

班诚[①]，伍民顺*[①]，张晓鹏[②]，许江涛[①]，耿莉[①]
[①]（西安交通大学微电子学院 西安 710049）
[②]（西北核技术研究所 西安 710024）

摘 要：本文提出了一种高精度、低成本的测试方法，用于对低失真放大器的频谱性能进行测试。该方法不同于传统方法中竭力满足“精确激励和精确测量”的做法，而是采用精度相对较低的信号源和模数转换器（ADC）测试放大器的频谱特性。在该方法中，多路滤波技术被用来产生多个频谱相关的信号，利用这些信号的频谱相关性，结合提出的算法，可以从被低精度测试仪器恶化的测试数据中获取被测放大器的频谱信息。仿真结果表明，总谐波失真（THD）值为-79.76 dB 的信号源和精度为 20 位的 ADC 可以准确地测量 THD 值为-121.42 dB 的放大器的频谱性能。与传统方法相比，所提出的方法分别降低了放大器测试对信号源的精度要求约 60 dB 和对 ADC 的精度要求约 4 位。

关键词：放大器；频谱测试；多路滤波技术；信号源；模数转换器

Testing the Spectral Performance of Low Distortion Amplifier Using Impure Signal Source and Imprecise ADC

BAN Cheng[①] WU Minshun*[①] ZHANG Xiaopeng[②] XU Jiangtao[①] GENG Li[①]
[①](School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China)
[②](Northwest Institute of Nuclear Technology, Xi'an 710024, China)

Abstract: An accurate and low-cost method is proposed for testing the spectral performance of low-distortion amplifier. Different from the paradigm of "accurate excitation and measurement", the proposed method uses an impure signal source and an imprecise Analog-to-Digital Converter (ADC) to test the spectral performance of an amplifier. Multi-path filtering technique is used to generate multiple spectral related signals. Combining the spectral correlation of these signals and the proposed algorithm, the spectral information can be obtained from the test data deteriorated by the imprecise instruments. Simulation results show that a -79.76 dB pure signal source and a 20-bit ADC can be used to accurately test an amplifier with targeted -121.42 dB Total Harmonic Distortion (THD) performance. The proposed method can relax the requirements on signal source and ADC by 60 dB and 4 bits respectively, compared with traditional methods.

Key words: Amplifier; Spectral test; Multi-path filtering technique; Signal source; Analog-to-digital converter

1 引言

放大器是世界上体量最大的模拟集成电路模块之一，也是几乎所有片上系统的关键组成部分。测试低失真放大器的频谱特性不仅非常重要，也极具挑战性。为了确保测试精确，IEEE 标准以及工业界先进的测试技术要求，激励信号和测量仪器的精度必须比被测放大器的精度至少高 20 dB 或 3~4 位[1-3]。

当今，最先进的放大器其总谐波失真（Total Harmonic Distortion, THD）已达到-120 dB 或甚至更低的水平[4,5]。为了测试如此低失真的放大器，激励信号的失真成分应在-140 dB 级别或更低，测量仪器的分辨率至少为 24 位。但是，想要获得如此高精度的激励信号和测量仪器，即使对当今性能

最先进的仪器而言，这个要求也是极具挑战性的，在内建自测试（Built-In Self-Test, BIST）环境中几乎是不可能的。据作者所知，大部分商用信号发生器只能产生 THD 值约为-90 dB 的信号，当前性能最佳的音频信号发生器 APX555 可以产生-120 dB 的正弦信号。即使采用价格昂贵的高品质带通滤波器对信号进行滤波以后，所得信号的精度达到-130 dB 左右，尚不满足 IEEE 标准要求的、测试 THD 值为-120 dB 的放大器的精度要求。由于放大器的输出是模拟信号，因此需要模数转换器(Analog-to-Digital Converter, ADC)对其进行量化。而大多数高分辨率的 ADC 是基于 Σ-Δ 调制技术，此类 ADC 的转换速率比较慢，有时甚至无法满足放大器测试要求的采样速度。对于 BIST 环境，在有限的芯片面积芯片上设计这种高性能的信号发生电路和采样电路更加艰难。

为应对现有仪器设备无法满足超低失真放大器测试需求的难题，产业界和学术界进行了广泛的研究。Vasan 等人提出了一种谐波消除技术，可以产生 THD 值为-100 dB 的低失真正弦波[6]。德州仪器公司提出了一个特殊的测试电路，用于扩展失真分析仪的测试能力。但该方法易受限于加在该标准同相放大器结构的电阻的精度[7]。Yanagida 等人提出了一种产生低失真正弦波的方法，该方法通过修改任意波形发生器的程序和使用滤波器来降低失真成分。实验结果表明，谐波被抑制超过-90 dBFS[8,9]。Zhuang 等人采用数模转换器(Digital – to-Analog Converter, DAC)非线性估计与校准方法研制超高精度信号源，具备测试-120 dB 的 ADC 的能力[10]。Magstadt 等人提出了一种应用于 ADC 频谱测试的算法，该算法可以降低数据转换器测试对仪器的精度要求[11]。然而，这些算法的例子仅见于 ADC 或 DAC 测试中。Thanapitak 等人提出了一种测试生物电信号放大器性能的测试台，该测试台可精确测量放大器的增益、输入阻抗、共模抑制比等测试参数，但该方法并未放松对测试仪器的要求[12]。日本 Gunma 大学和 ROHM 半导体团队提出了一种无需音频分析仪的运放交流特性测试方法，该方法采用 summing node(SN)测试方法，可根据运放输出和反馈电路求和点处的频率分量之间的差值，计算出运放的谐波失真成分[13,14]。实验结果表明，该方法的测量精度可达到-130 dBc。另有一些研究工作则聚焦在提高运放测试中的缺陷覆盖率、搭建 5G 功率运放测试系统[15,16]。在解决因测量仪器精度不足导致的低失真放大器测试难题上，仍缺乏切实可行的技术方案。

为此，本文提出了一种高精度、低成本的低失真放大器频谱测试方法。与传统方法不同的是，该方法采用低精度信号源激励被测放大器，采用低精度 ADC 对放大器的输出进行采样。此外，该方法采用多路滤波技术来产生多组频谱相关的信号，这些信号间的频谱相关性可以用于从被低精度信号源和测量仪器恶化的测试数据中提取被测放大器的谐波失真，从而计算出被测放大器的频谱信息。详细的理论分析和仿真结果表明，THD 值为-79.76 dB 的正弦信号和精度为 20 位的 ADC 可以精确测量出 THD 值为-121.42 dB 的放大器的频谱特性。与传统方法相比，所提出的方法分别降低了放大器测试对信号源和 ADC 的精度要求约 60 dB 和 4 位。

2 传统的放大器频谱测试方法及其不足

2.1 传统的放大器频谱测试方法

图 1 展示了传统的放大器频谱测试方法。为了精确测试放大器的频谱特性，首先使用高精度信号源产生高纯度正弦信号，用于激励被测放大器，接着放大器的输出信号由高精度 ADC 采样，并将 ADC 的输出数据传输至电脑，通过对数据进行 FFT 处理，最终计算出被测放大器的频谱参数。

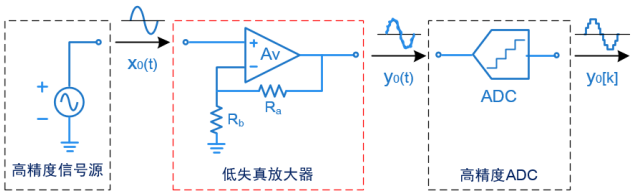


图 1 传统的放大器频谱测试方法的框图

在模拟与混合信号集成电路测试中，放大器通常被视为弱非线性系统。当给被测放大器输入一组正弦信号 $x_0(t)=\cos(\omega t)$（幅值和相位已归一化）时，放大器的单位增益输出响应 $y_0(t)$是一组低失真的正弦信号，该信号除了包含输入的正弦信号，还包含因放大器的非线性引入的高阶谐波失真项。因此，该信号的傅里叶级数形式表示为：

$$y_0(t)=\frac{1}{2}\left(e^{j\omega t}+e^{-j\omega t}\right)+\sum_{k=2}^{K}\left(h_{uk}\cdot e^{jk\omega t}+\overline{h_{uk}}\cdot e^{-jk\omega t}\right) \quad (1)$$

式（1）中，ω 表示正弦信号 $x_0(t)$的角频率，t 表示时间，h_{uk}表示被测放大器输出信号的第 k 次谐波的傅里叶系数，K 表示谐波的最高阶数，h_{uk} 与 $\overline{h_{uk}}$ 互为共轭关系。对于低失真放大器而言，这些系数远小于 1。在该式中，被测放大器的增益误差和初始相位偏移假设已被校正过。

被测放大器的 THD 和无杂散动态范围（Spurious-Free Dynamic Range, SFDR）可以根据系数 h_{uk} 计算出来，计算式如下所示：

$$THD = 10 \cdot \log_{10}\left\{\sum_{k=2}^{K}\left|2h_{uk}\right|^2\right\} \tag{2}$$

$$SFDR = -20 \cdot \log_{10}\left(\max_{k\geq 2}\left|2h_{uk}\right|\right) \tag{3}$$

2.2 传统测试方法的不足

根据 IEEE 标准及工业界先进的测试技术，激励信号和测量仪器（主要是仪器内置的 ADC）的非线性失真必须远小于放大器测试所能容忍的非线性失真。当被测放大器的失真度非常低时，如前文所述，该项要求对现有的仪器设备极具挑战性。当激励信号或测量仪器的失真相对于被测放大器的失真较大时，仪器的失真会恶化放大器的测试结果。

当信号源的精度不足时，所产生的低精度正弦信号 $x(t)$包含将非线性失真成分，而绝大部分非线性失真成分主要是各次谐波，因此，$x(t)$可以表示为基波频率处的正弦信号和高阶谐波失真项的组合。不失一般性，在归一化信号幅值、同步相位后，$x(t)$有如下的傅里叶级数形式：

$$x(t) = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right) + \sum_{k=2}^{K}\left(h_{xk}\cdot e^{jk\omega t} + \overline{h_{xk}}\cdot e^{-jk\omega t}\right) \tag{4}$$

式中，其中 h_{xk} 是输入信号第 k 次谐波的傅里叶系数。在实际生活中，由于失真度约为 0.01%的信号相对容易产生，所以 h_{xk}可以认为远小于 1。

使用该信号 $x(t)$激励被测放大器，并定义被测放大器的单位增益输出响应为 $y(t)$，则 $y(t)$同时受到输入信号的失真和放大器自身固有的失真的影响，因此，$y(t)$的失真项是两者的组合效应[10,11]，其表达式为：

$$y(t) = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right) + \sum_{k=2}^{K}\left(h_{xk}\cdot e^{jk\omega t} + \overline{h_{xk}}\cdot e^{-jk\omega t}\right) + \sum_{k=2}^{K}\left(h_{uk}\cdot e^{jk\omega t} + \overline{h_{uk}}\cdot e^{-jk\omega t}\right) \tag{5}$$

由于放大器的输出同时包含了输入信号的失真和被测放大器的失真，所以不能利用式（2）和式（3）直接计算被测放大器的非线性失真。

同理，当测量仪器内置的 ADC 包含较大的非线性时，使用高纯度正弦信号 $x_0(t)$激励 ADC，ADC 的输出将同时包含输入信号和 ADC 本身的非线性导致的高阶谐波失真项，其傅里叶级数形式可以表示为：

$$s(t) = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right) + \sum_{k=2}^{K}\left(h_{sk}\cdot e^{jk\omega t} + \overline{h_{sk}}\cdot e^{-jk\omega t}\right) \tag{6}$$

式中，其中 h_{sk}是 ADC 第 k次谐波的傅里叶系数。对于分辨率高于 16 位的 ADC 而言，h_{sk}显然远小于 1。

当使用低精度信号源和低精度 ADC 测试低失真放大器时，如果不进行额外的处理，所获得的输出信号将同时包含信号源、放大器和 ADC 的非线性失真，该信号 $z(t)$包含的失真项为三者失真项的组合效应，如式（7）所示：

$$z(t) = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right) + \sum_{k=2}^{K}\left(h_{xk}\cdot e^{jk\omega t} + \overline{h_{xk}}\cdot e^{-jk\omega t}\right) + \sum_{k=2}^{K}\left(h_{uk}\cdot e^{jk\omega t} + \overline{h_{uk}}\cdot e^{-jk\omega t}\right) + \sum_{k=2}^{K}\left(h_{sk}\cdot e^{jk\omega t} + \overline{h_{sk}}\cdot e^{-jk\omega t}\right) \tag{7}$$

显然，通过式（7）无法计算出被测放大器的频谱参数信息。因此，本文将采用多路滤波技术来识别和提取被测放大器的失真项。

3 使用低精度信号源和低精度 ADC 对放大器进行精确地频谱测试

本节将介绍所提出的放大器频谱测试方法，该方法允许采用精度低于 IEEE 标准要求的信号源和 ADC 测量低失真放大器的频谱特性，从而极大地降低了放大器测试对信号源和测量仪器的精度要求。需注意，本文所述的低精度，特指低于 IEEE 标准对测试仪器（信号源和 ADC）要求的精度。

3.1 多路滤波技术

上一节描述了当信号源和测量仪器精度不足时，被测放大器的输出被严重恶化，导致无法获得准确的频谱测试结果。为了从被恶化的测试数据中分离出被测放大器的失真项，我们使用多路滤波技术[10,11]。

所使用的多路滤波技术包含至少两个串联的滤波器，这些滤波器可以对信号进行处理，从而产生多组频谱相关的信号。利用信号间的频谱相关性，可以将电路的非线性失真从多组频谱相关的信号中分离出来。此处的频谱相关是指两组信号所包含的基波及各次谐波的频率相同，尽管幅值和相位不一定相同，但可以通过滤波器的传递函数计算出来。

所使用的多路滤波技术需满足以下三项要求。第一项要求是多路滤波器组线性度要高，不能在电路中引入额外的误差。第二项要求是传递函数要已知，从而可以计算出滤波后信号的各次谐波的幅值

和相位信息。第三项要求是为了确保后级电路引入相同量级的误差，对于滤波后的信号，它们基波的幅值要相等。对于第三项要求，在实际测试中，可以通过选择参数值合适的电阻和电容来调整滤波器的转折频率，使得信号幅值在基波频率处的衰减程度相等。

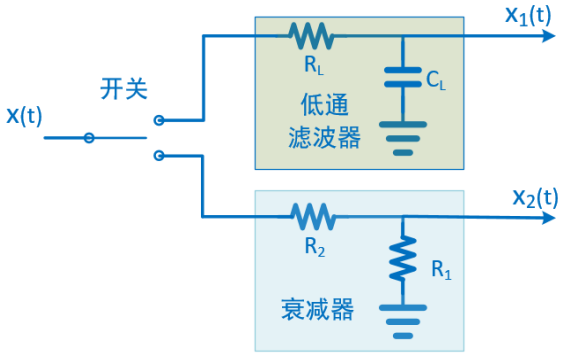


图 2 一个可供选择的用于实现多路滤波技术的电路

图 2 展示了一个可供选择的用于实现多路滤波技术的电路。该滤波器组是由一阶低通滤波器和电阻衰减器并联构成，由开关控制滤波通路。这两个滤波器的传递函数如下所示：

$$H_1(j\omega)=\left(j\frac{\omega}{\omega_0}+1\right)^{-1} \tag{8}$$

$$H_2(j\omega)=\frac{R_1}{R_1+R_2} \tag{9}$$

在式（8）中，H_1 是低通滤波器的传递函数，ω_0 是低通滤波器的转折频率，它由电阻 R_L 和电容 C_L 决定。在式（9）中，H_2 是电阻衰减器的传递函数，R_1 和 R_2 是衰减器中的电阻。

当式（4）所示的低精度正弦信号 $x(t)$ 通过多路滤波器组，可以得到两组滤波后的信号，其表达式分别为：

$$\begin{aligned}x_1(t)&=\frac{1}{2}\left(H_1(j\omega)\cdot e^{j\omega t}+\overline{H}_1(j\omega)\cdot e^{-j\omega t}\right)\\&+\sum_{k=2}^{K}\left(h_{xk}\cdot H_1(jk\omega)\cdot e^{jk\omega t}+\overline{h}_{xk}\cdot\overline{H}_1(jk\omega)\cdot e^{-jk\omega t}\right)\end{aligned} \tag{10}$$

$$\begin{aligned}x_2(t)&=\frac{1}{2}\left(H_2(j\omega)\cdot e^{j\omega t}+\overline{H}_2(j\omega)\cdot e^{-j\omega t}\right)\\&+\sum_{k=2}^{K}\left(h_{xk}\cdot H_2(jk\omega)\cdot e^{jk\omega t}+\overline{h}_{xk}\cdot\overline{H}_2(jk\omega)\cdot e^{-jk\omega t}\right)\end{aligned} \tag{11}$$

式中，$x_1(t)$ 和 $x_2(t)$ 的各次谐波经滤波处理后，呈现频谱相关性，这些谐波间的频谱相关性，在本文方法中将被用于提取被测放大器的谐波失真项。

3.2 运用多路滤波技术、采用精度相对较低的信号源和 ADC 对放大器进行频谱测试

本文提出的、采用精度相对较低的信号源和 ADC 对放大器进行频谱测试的方法，其框图如图 3 所示。

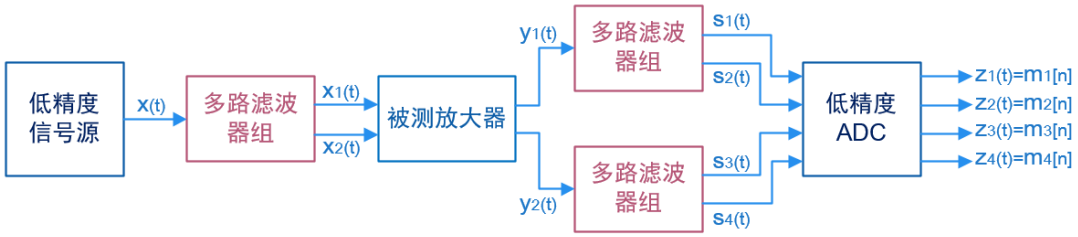


图 3 采用精度相对较低的信号源和 ADC 对低失真放大器进行频谱测试的框图

首先，采用低精度信号源产生的正弦信号 $x(t)$，$x(t)$ 分别通过多路滤波器组的低通滤波器和衰减器，得到两组频谱相关的信号 $x_1(t)$ 和 $x_2(t)$。然后，让 $x_1(t)$ 和 $x_2(t)$ 分别通过被测放大器，得到两组相应的输出 $y_1(t)$ 和 $y_2(t)$。接着，$y_1(t)$ 和 $y_2(t)$ 再分别通过多路滤波器组，得到四组信号 $s_1(t)$，$s_2(t)$，$s_3(t)$ 和 $s_4(t)$。四组信号经过低精度 ADC 采样，得到四组 ADC 的输出数据，分别为 $m_1[n]$，$m_2[n]$，$m_3[n]$ 和 $m_4[n]$。

为了确保四组信号通过 ADC 时引入相同量级的失真，这四组信号的基波成分的幅值必须相等。为区别被测放大器前后的滤波器组，将被测放大器后的滤波器组的传递函数分别用 H_3 和 H_4 表示。为了简便，在本文中，被测放大器前后的滤波器组设置为相同的，即具有相同的传递函数和电阻电容值，也就是 $H_1=H_3$，$H_2=H_4$。若后组滤波器组的设置和传递函数与前组滤波器组不同，只需将 H_3 和 H_4 修改为当前的、后组滤波器组的传递函数即可，修改后的 H_3 和 H_4 不影响后续的公式推导。

在图 3 中，低精度信号源产生的正弦信号 $x(t)$ 如式（4）所示。其经多路滤波器组后得到的 $x_1(t)$ 和 $x_2(t)$ 如式（10）和式（11）所示。当 $x_1(t)$ 和 $x_2(t)$ 通过被测放大器时，结合式（2），得到的两组被测放大器的输出 $y_1(t)$ 和 $y_2(t)$，其表达式分别为：

$$y_1(t)=\frac{1}{2}\left(H_1(j\omega)\cdot e^{j\omega t}+\overline{H}_1(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_1(jk\omega)+h_{uk}\right)\cdot e^{jk\omega t}\right.\\ \left.+\left(\overline{h}_{xk}\cdot\overline{H}_1(jk\omega)+\overline{h}_{uk}\right)\cdot e^{-jk\omega t}\right)\end{pmatrix} \tag{12}$$

$$y_2(t)=\frac{1}{2}\left(H_2(j\omega)\cdot e^{j\omega t}+\overline{H}_2(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_2(jk\omega)+h_{uk}\right)\cdot e^{jk\omega t}\right.\\ \left.+\left(\overline{h}_{xk}\cdot\overline{H}_2(jk\omega)+\overline{h}_{uk}\right)\cdot e^{-jk\omega t}\right)\end{pmatrix} \tag{13}$$

$y_1(t)$和 $y_2(t)$再分别通过多路滤波器组，得到四组信号 $s_1(t)$，$s_2(t)$，$s_3(t)$和 $s_4(t)$，其表达式分别为：

$$s_1(t)=\frac{1}{2}\left(H_1(j\omega)\cdot H_3(j\omega)\cdot e^{j\omega t}+\overline{H}_1(j\omega)\cdot\overline{H}_3(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_1(jk\omega)+h_{uk}\right)\cdot H_3(jk\omega)\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_1(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_3(jk\omega)\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{14}$$

$$s_2(t)=\frac{1}{2}\left(H_1(j\omega)\cdot H_4(j\omega)\cdot e^{j\omega t}+\overline{H}_1(j\omega)\cdot\overline{H}_4(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_1(jk\omega)+h_{uk}\right)\cdot H_4(jk\omega)\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_1(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_4(jk\omega)\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{15}$$

$$s_3(t)=\frac{1}{2}\left(H_2(j\omega)\cdot H_3(j\omega)\cdot e^{j\omega t}+\overline{H}_2(j\omega)\cdot\overline{H}_3(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_2(jk\omega)+h_{uk}\right)\cdot H_3(jk\omega)\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_2(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_3(jk\omega)\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{16}$$

$$s_4(t)=\frac{1}{2}\left(H_2(j\omega)\cdot H_4(j\omega)\cdot e^{j\omega t}+\overline{H}_2(j\omega)\cdot\overline{H}_4(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=1}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_2(jk\omega)+h_{uk}\right)\cdot H_4(jk\omega)\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_2(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_4(jk\omega)\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{17}$$

式中，H_3表示放大器后级的多路滤波器组中低通滤波器的传递函数，H_4表示放大器后级的多路滤波器组中电阻衰减器的传递函数.

从式（14）至式（17）中可看出，这些信号依旧是频谱相关的。该四组信号经过低精度 ADC 采样，得到的信号的表达式分别为：

$$z_1(t)=\frac{1}{2}\left(H_1(j\omega)\cdot H_3(j\omega)\cdot e^{j\omega t}+\overline{H}_1(j\omega)\cdot\overline{H}_3(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_1(jk\omega)+h_{uk}\right)\cdot H_3(jk\omega)+h_{sk}\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_1(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_3(jk\omega)+\overline{h}_{sk}\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{18}$$

$$z_2(t)=\frac{1}{2}\left(H_1(j\omega)\cdot H_4(j\omega)\cdot e^{j\omega t}+\overline{H}_1(j\omega)\cdot\overline{H}_4(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_1(jk\omega)+h_{uk}\right)\cdot H_4(jk\omega)+h_{sk}\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_1(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_4(jk\omega)+\overline{h}_{sk}\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{19}$$

$$z_3(t)=\frac{1}{2}\left(H_2(j\omega)\cdot H_3(j\omega)\cdot e^{j\omega t}+\overline{H}_2(j\omega)\cdot\overline{H}_3(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_2(jk\omega)+h_{uk}\right)\cdot H_3(jk\omega)+h_{sk}\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_2(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_3(jk\omega)+\overline{h}_{sk}\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{20}$$

$$z_4(t)=\frac{1}{2}\left(H_2(j\omega)\cdot H_4(j\omega)\cdot e^{j\omega t}+\overline{H}_2(j\omega)\cdot\overline{H}_4(j\omega)\cdot e^{-j\omega t}\right)+\sum_{k=2}^{K}\begin{pmatrix}\left(\left(h_{xk}\cdot H_2(jk\omega)+h_{uk}\right)\cdot H_4(jk\omega)+h_{sk}\right)\cdot e^{jk\omega t}\\ +\left(\left(\overline{h}_{xk}\cdot\overline{H}_2(jk\omega)+\overline{h}_{uk}\right)\cdot\overline{H}_4(jk\omega)+\overline{h}_{sk}\right)\cdot e^{-jk\omega t}\end{pmatrix} \tag{21}$$

$z_1(t)$，$z_2(t)$，$z_3(t)$和 $z_4(t)$所描述的是依据测试信号流向推导出来的四组 ADC 输出数据，因为式中包含未知参数 h_{xk}、h_{uk}和 h_{sk}，所以 $z_1(t)$，$z_2(t)$，$z_3(t)$和 $z_4(t)$无法根据式（18）至式（21）直接计算出来。但是，由于 ADC 的输出数据可以被采集并进行分析，且所采集的数据也是周期信号和各次谐波的形式，因此，采集到的数据可以表示为：

$$m_1[n]=\sum_{k=1}^{K}\left(h_{m1k}\cdot e^{jk\omega t}+\overline{h}_{m1k}\cdot e^{-jk\omega t}\right) \tag{22}$$

$$m_2[n]=\sum_{k=1}^{K}\left(h_{m2k}\cdot e^{jk\omega t}+\overline{h}_{m2k}\cdot e^{-jk\omega t}\right) \tag{23}$$

$$m_3[n]=\sum_{k=1}^{K}\left(h_{m3k}\cdot e^{jk\omega t}+\overline{h}_{m3k}\cdot e^{-jk\omega t}\right) \tag{24}$$

$$m_4[n]=\sum_{k=1}^{K}\left(h_{m4k}\cdot e^{jk\omega t}+\overline{h}_{m4k}\cdot e^{-jk\omega t}\right) \tag{25}$$

式中，$m_1[n]$，$m_2[n]$，$m_3[n]$和 $m_4[n]$表示采集到的 ADC 的输出数据，分别与 $z_1(t)$，$z_2(t)$，$z_3(t)$和 $z_4(t)$对应，且分别表示同一组 ADC 输出数据，因此，它们分别相等，如下所示：

$$z_1(t)=m_1[n] \tag{26}$$

$$z_2(t)=m_2[n] \tag{27}$$

$$z_3\left(t\right)=m_3\left[n\right] \tag{28}$$

$$z_4\left(t\right)=m_4\left[n\right] \tag{29}$$

结合上述表达式，可以获得被测放大器各次谐波的系数的计算式，其表达式为：

$$h_{uk}=\frac{\left(h_{m1k}-h_{m2k}\right)}{H_3\left(jk\omega\right)-H_4\left(jk\omega\right)} -\frac{h_{m1k}-h_{m3k}}{H_3\left(jk\omega\right)\cdot\left(H_1\left(jk\omega\right)-H_2\left(jk\omega\right)\right)}\cdot H_1\left(jk\omega\right),\ k\geq 2 \tag{30}$$

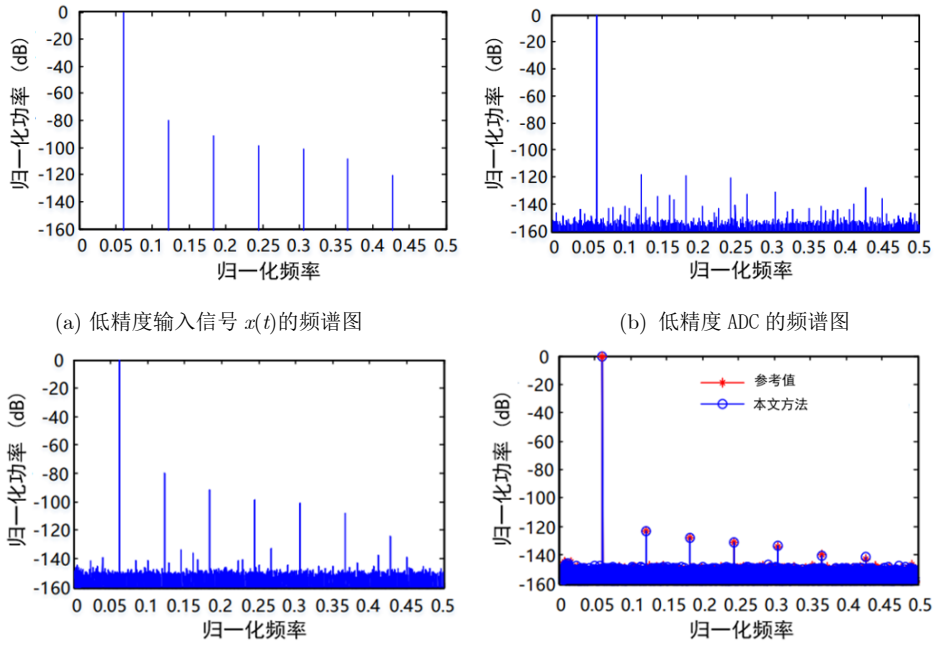
将式（30）获得的 h_{uk} 带入式（2）和式（3），即可计算出被测放大器的频谱参数 THD 和 SFDR。

4 仿真结果

为了验证所提出的低失真放大器频谱测试方法，本节将展示仿真结果。

在仿真中，被测放大器的单位增益输出响应可以表征为一个七阶多项式。为了获得被测放大器准确的频谱特性，在对照组中，使用满足 IEEE 标准要求的高精度信号源和高精度 ADC 作为被测放大器的激励和采集，使用传统测试方法测量被测放大器的频谱参数值，作为验证本文方法的参考值。在仿真中，使用理想的、不含非线性失真的正弦信号作为输入，使用理想的、24 位的 ADC 采集被测放大器的输出，使用 FFT 和频谱参数计算公式获得被测放大器的参数值。在单次仿真中，测得的放大器 THD 值为-121.42 dB，得到的频谱如图 4（d）中的红色频谱所示。

为了仿真本文提出的方法，首先对低精度输入信号和低精度 ADC 进行建模。低精度输入信号是由理想正弦信号和随机加入的谐波失真成分组成。低精度 ADC 用一组转换电平进行建模，转换电平随机加入均值为零、给定方差的高斯随机变量，以构成 ADC 的非线性误差。多路滤波器组中的无源器件根据高精度电阻电容的实际误差值设置误差百分比，并在误差范围内随机生成。外部测量噪声在放大器的输入端引入，且噪声的量级与放大器实际的噪声量级相当。所有仿真均是在相干采样下进行。



(a) 低精度输入信号 $x(t)$的频谱图

(b) 低精度 ADC 的频谱图

(c) 基于传统方法、使用低精度信号和低精度 ADC 测试放大器的得到频谱图

(d) 本文方法得到的放大器频谱图（蓝色）与基于传统方法、使用理想正弦信号和理想 ADC 测试放大器得到的频谱图（红色）对比

图 4 单次仿真中本文方法与传统方法的仿真结果对比

图 4(a)给出了在单次仿真中低精度输入信号 $x(t)$的频谱图。可以看到 $x(t)$包含了各次谐波失真成分，其 THD 值仅有-79.76 dB。图 4(b)给出了低精度 ADC 的频谱图，其 THD 约为-118.13 dB。对于 THD 值低于-120 dB 的放大器而言，根据 IEEE 标准，激励信号和测量仪器的 THD 值应至少为-

140 dB。对于图 3 所示的低精度输入信号和低精度 ADC，其失真度远大于传统方法要求的数值。因此，当使用低精度输入信号和低精度 ADC 测试放大器时，若仍采用传统的测试方法和数据分析方式，将会得到错误的结果，如图 4(c)所示，该条件下计算出来的被测放大器的 THD 值仅-79.73 dB，几乎等于低精度信号源的失真。因此，我们已经验证了在传统方法中，低精度的正弦信号和低精度的 ADC 不能用来测试低失真放大器的频谱特性。

在我们的算法中，低精度信号源和低精度 ADC 同时用来测试放大器的频谱特性。根据图 3 的测试框图，多路滤波技术被用来产生频谱相关的两组信号，并激励被测放大器。然后放大器的两组输出将通过后级的多路滤波器组，得到四组频谱相关的信号。最后四组频谱相关的信号被低精度 ADC 采集，获得四组相干的测试数据。利用这四组信号间的频谱相关性和提出来的算法，可以精确计算出被测放大器的频谱特性。

图 4(d)展示了采用传统测试方法、采用高精度信号源和高精度 ADC 测得的放大器频谱图（红色）和采用本文方法、采用精度较低的信号源和 ADC 测得的放大器频谱图（蓝色）。使用理想正弦信号和理想 ADC 测得的放大器 THD 值为-121.42 dB。使用所提出的方法仿真得到的放大器 THD 值为-121.02 dB，很接近参考值。值得强调的是，本文方法是在输入信号 THD 值仅为-79.76 dB，ADC 仅为 20 位的条件下，获得了精确的测试结果。与传统方法相比，要获得相同量级的测试精度，则需要 THD 值至少为-140 dB 的输入信号和精度为 24 位或更高的 ADC。

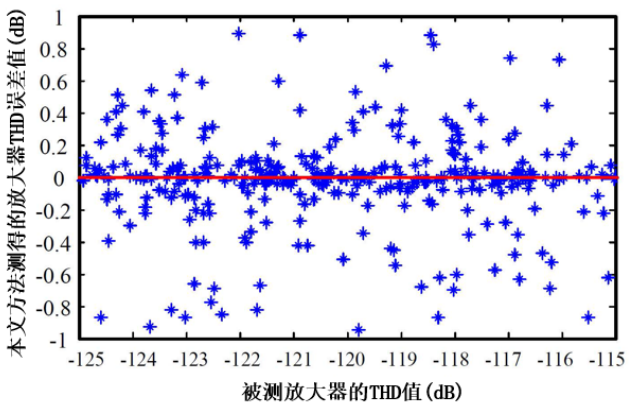


图 5 仿真的放大器 THD 误差

图 5 给出了使用所提出的方法进行 500 次随机运行后得到的仿真的放大器 THD 值。在仿真中，放大器的谐波失真成分在-125 dB 到-115 dB 的范围内随机产生。可以看出，仿真得到的 THD 值约等于它们的真实值，最大的仿真误差也仅为 0.97 dB。

5 方法比较

为与相关方法进行比较，本节将现有文献中具备降低测试仪器精度的能力、且测量精度在-120 dB 及以下的放大器测试方法列出来，并将测试方法降低信号源和测量仪器精度要求的程度作为指标进行对比，得到表格如表 1 所示。需要说明的是，指标所指的精度降低程度是与 IEEE 标准测试方法所需的仪器精度相比较，以测试 THD 值为-120 dB 的放大器为例，IEEE 标准测试方法所需的信号源的 THD 值为-140 dB（约 24 位），所需的测量仪器的 THD 值为-140 dB（约 24 位）。

表 1 不同方法对信号源和测量仪器的精度要求的降低程度对比

方法	信号源	测量仪器
文献[7]	40 dB	40 dB
文献[11]	8位	0 dB
文献[14]	未给出	8位
本文方法	60 dB	4位

在文献[7]中，德州仪器公司提出了一种特殊的测试电路，可测量失真程度低于市售失真分析仪测量本底值的运算放大器。该电路可将运算放大器的失真比正常产生的失真高出 101 倍（约 40 dB）。这意味着与 IEEE 标准对测试仪器的精度要求相比，该方法将放大器测试对信号源和失真分析仪的精度要求分别降低 40 dB。

在文献[11]中，Zhuang 等人设计了一款超高精度正弦信号发生器，该信号发生器采用 16 位 DAC 作为正弦信号源，通过对该 DAC 的输出进行非线性识别与校准，可以大幅降低该 DAC 输出信号的失真程度，使其具备测试 THD 为-120 dB 的 ADC 的能力。尽管文献以测试 ADC 为例，但由于其出色的超低失真性能，也具备测试失真程度相近的放大器的能力。为便于与本文方法进行比较，将 DAC 非线性失真的降低程度作为比较值。与 IEEE 标准对信号源的精度要求相比，该方法将测试对信号源的精度要求降低了 8 位。

在文献[14]中，Iimori 等人提出了一种无需音频分析仪的运放交流特性测试方法，实验结果表明，该方法可以在无需高精度音频分析仪的情况下，仅采用内置 16 位 ADC 的 myDAQ 采集系统，可对 THD 值超过-120 dB 的放大器进行测试。由于该方法没有给出所使用的信号源的具体型号，也没有针对信号源精度的降低程度做具体说明，因此无法评估该方法降低信号源精度的性能。

本文提出了一种基于多路滤波技术的放大器频谱测试方法。该方法通过产生多组频谱相关的信号，并利用信号间的频谱相关性，结合文中给出的放大器谐波失真计算式，可以在信号源和测量器件精度不足的情况下，精确计算出被测放大器的频谱性能参数。仿真结果表明，本文方法分别降低了放大器测试对信号源和 ADC 精度的要求约 60 dB、4 位。

6 结论

本文提出了一种高精度、低成本的放大器测试方法，该方法采用低精度信号源和低精度测量仪器对低失真放大器进行频谱测试。在该方法中，多路滤波技术被用来产生多路频谱相关的信号，利用信号间的频谱相关性，结合本文给出的算法，可以从被信号源和 ADC 的非线性失真恶化的测试数据中提取出被测放大器的失真成分。仿真结果表明，即采用 THD 值为-79.76 dB 正弦信号和分辨率为 20 位的 ADC，即可精确测量 THD 低于-120 dB 的放大器的频谱性能。与传统方法相比较，本文方法分别降低了放大器测试对信号源和 ADC 精度的要求约 60 dB、4 位。为更好的验证本文方法，我们在后续工作中将搭建电路实验平台，用实际的信号源和测量仪器对多款商用低失真放大器进行测试。

更进一步的，该方法可以同时计算出低精度信号源和低精度 ADC 的频谱性能参数，为放大器及其相关模拟与混合信号集成电路的协同测试提供解决方案。该方法还可拓展为低精度信号源或低精度 ADC 对放大器及其相关电路的单独测试，从而降低测试对仪器设备的精度要求。相关研究工作已在进行中。

由于所提出的方法可以极大地降低放大器测试对信号源和测量仪器的精度的要求，因此该方法可以大幅降低放大器的测试成本。所提出的方法还能为性能已超过现有商用测试设备测试能力的放大器及其相关集成电路产品提供切实可行的测试解决方案。由于该方法的算法简洁，对硬件要求较低，因此该方法还将适用于内建自测试。

参 考 文 献

[1] IEEE Std, 2008-1057. Waveform Generation, Measurements and Analysis Committee (TC-10). IEEE standard for digitizing waveform recorders[S]. 2008. doi: 10.1109/IEEESTD.2008.4494996.

[2] IEEE Std, 2011-1241. Waveform Generation Measurement and Analysis Technical Committee. IEEE standard for terminology and test methods for analog to digital converters[S]. 2011. doi: 10.1109/IEEESTD.2011.5692956.

[3] 伯恩斯, 罗伯茨, 冯建华, 肖钢. 混合信号集成电路测试与测量[M]. 电子工业出版社, 2009.

[4] Analog Devices, Inc.: ADA4807-1/ADA4807-2/ADA4807-4[OL]. http://www.analog.com/media/en/technical-documentation/data-sheets/ADA4807-1_4807-2_4807-4.pdf,2015.

[5] Analog Devices, Inc.: AD8597/AD8599[OL]. http://www.analog.com/media/en/technical-documentation/data-sheets/AD8597_8599.pdf,2013.

[6] Bharath Karthik Vasan, Siva Kumar Sudani, Degang J. Chen, *et al.* Low-distortion sine wave generation using a novel harmonic cancellation technique[C]. IEEE TCAS-I,2013,1122-1134. doi: 10.1109/TCSI.2013.2249178.

[7] Texas Instruments Inc.: OPA1611[OL]. http://www.ti.com/lit/ds/symlink/opa1611.pdf,2011.

[8] Tomonori Yanagida, Shohei Shibuya, Haruo Kobayashi, *et al.* High-frequency low-distortion one-tone and two-tone signal generation using arbitrary waveform generator[C]. 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Hangzhou, China, 2016: 741-743.doi: 10.1109/ICSICT.2016.7999029.

[9] Shohei Shibuya, Yutaro Kobayashi and Haruo Kobayashi. High-frequency low-distortion signal generation algorithm with arbitrary waveform generator[C]. 2015 IEEE 11th International Conference on ASIC (ASICON), Chengdu, China, 2016: 1-4.

[10] Benjamin Magstadt, Yuming Zhuang and Degang Chen. Accurate spectral testing with impure source and noncoherent sampling[J]. IEEE Transactions on Instrumentation and Measurement, November 2016: Vol. 65, No. 11.

[11] Yuming Zhuang, Akhilesh Kesavan Unnithan, Arun Joseph, *et al.* Low Cost Ultra-Pure Sine Wave Generation with Self Calibration. 2016 IEEE International Test Conference (ITC). Fort Worth, TX, USA, 2016: 1-9. doi: 10.1109/test.2016.7805843.

[12] Surachoke Thanapitak, Pongsatorn Sedtheetorn, Pornchai Chanyagorn, *et al.* Test Bench for Biopotential Instrumentation Amplifier using Single-Ended to Differential Amplifiers[C]. 2023 35th International Conference on Microelectronic Test Structure (ICMTS), Tokyo, Japan, 2023: 1-4. doi: 10.1109/ICMTS55420.2023.10094130.

[13] Gaku Ogihara, Takayuki Nakatani, Akemi Hatta, *et al.* Summing Node Test Method: Simultaneous Multiple AC Characteristics Testing of Multiple Operational Amplifiers[C]. 2020 IEEE 29th Asian Test Symposium (ATS), Penang, Malaysia, 2020, 1-6. doi: 10.1109/ATS49688.2020.9301550.

[14] Daisuke Iimori, Takayuki Nakatani, Shogo Katayama, *et al.* Summing Node and False Summing Node Methods: Accurate Operational Amplifier AC Characteristics Testing without A

udio Analyzer [C]. 2021 IEEE International Test Conference (ITC), CA, USA, 2021: 364-373. doi: 10.1109/ITC50571.2021.00052.

[15] Michael Sekyere; Marampally Saikiran; Degang Chen. All Digital Low-Cost Built-in Defect Testing Strategy for Operational Amplifiers with High Coverage [C]. 2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS), Torino, Italy, 2022: 1-5.doi: 10.1109/IOLTS56730.2022.9897224.

[16] Zongqi Cai, Sha Tang, Jun Luo, *et al*. High efficiency testing system for 5G power amplifier [C]. 2021 22nd International Conference on Electronic Packaging Technology (ICEPT), Xiamen, China, 2021:1-4. doi: 10.1109/ICEPT52650.2021.9568110.

班　诚：男，博士生，研究方向为数模混合集成电路测试.

伍民顺：男，副教授，研究方向为模拟与混合信号集成电路的设计与测试.

张晓鹏：男，博士生，研究方向为地震传感器、数据采集系统和接口电路.

许江涛：男，副教授，研究方向为射频、模拟电路设计和超低功耗生物医学电路设计.

耿　莉：女，教授，研究方向为电源管理集成电路，低压低功耗模拟与混合信号集成电路，射频集成电路和生物植入系统.

基于区块链和可追溯 CP-ABE 的电子病历存储和共享方案

杨琴[①] 王国梁[①] 张华君[*①] 石林[①] 徐守坤[*①]

(常州大学计算机与人工智能学院 江苏 213164)

摘 要：电子病历记录的存储需要保证数据的机密性、完整性和不可篡改。如果这些方面没有得到保证，会影响诊断结果和医疗质量，不利于构建良好的就医环境。本文结合星际文件系统(IPFS)，区块链和可追踪的密文策略属性基加密(T-CP-ABE)构造了一种安全的电子病历数据存储与共享方案。首先，患者将病历数据加密存储在 IPFS 上，减轻患者和区块链上的存储压力。链上存储用户身份、哈希后的密文数据在 IPFS 的地址和密钥密文等信息，避免中心化存储带来的单点故障问题，实现数据的可信存储与不可篡改。T-CP-ABE 方案不仅可验证来访者的身份，还可抵制来访者共谋导致的密钥泄露危机。现有的研究将访问控制编写在区块链上，会使得策略不可更改。遇到复杂的策略会影响节点的运行效率。本文将访问控制放在链外，实现可追踪的访问控制。经分析发现，本文的方案安全并且切实可行，满足现实需求。

关键词：访问控制；区块链；不可篡改；隐私保护；电子病历

Electronic Medical Record Storage and Sharing Scheme Based on Blockchain and Traceable CP-ABE

YANG Qin[①] WANG Guoliang[①] ZHANG Huajun[*①] SHI Lin[①] XU Shoukun[*①]

(①*School of Computer Science and Artificial Intelligence, Changzhou University, Jiangsu* 213164, *China*)

Abstract: The storage of electronic medical records needs to ensure the confidentiality, integrity and non-tampering of data. If these aspects are not guaranteed, it will affect the diagnostic results and medical quality, which is not conducive to building a good medical environment. This paper combines the Interplanetary File System (IPFS), blockchain and traceable ciphertext policy attribute-based encryption (T-CP-ABE) to construct a secure electronic medical record data storage and sharing scheme. First, patients encrypt and store medical record data on IPFS, reducing the storage pressure on patients and the blockchain. Information such as user identity, hashed ciphertext data in IPFS address and key ciphertext are stored on the chain to avoid the single point of failure problem caused by centralized storage, and to realize trusted storage and Immutability of the data. The T-CP-ABE scheme can not only verify the identity of the visitor, but also resist the key leakage crisis caused by the collusion of the visitor. Existing research writes access control on the blockchain, which makes the policy unchangeable. Encountering complex strategies will affect the operating efficiency of nodes. This article places access control outside the chain to achieve traceable access control. After analysis, it is found that the scheme in this paper is safe and feasible, and meets the actual needs.

Key words: Access Control; Blockchain; Immutability; Privacy Protection; Electronic Medical Record

1 引言

数字化时代的发展使得数据成为最有价值的生产要素之一。电子病历是医疗机构重要的数字化管理内容，它使得医生能够有效的共享患者信息，提高诊断和治疗的准确性和效率。在这个时代浪潮下，推动医疗管理网格化是至关重要的，也引起了各地有关部门的重视。

但电子病历的数据泄露事件频频发生。2022 年 1 月，田纳西州的电子病历厂商 QRS 未采取有效的监测手段，导致患者门户被攻击，数据被泄露。同年 CommonSpirit 医疗集团作为全美第二大非营利性连锁医院遭受网络攻击后，患者无法访问病历数据和登录患者门户，患者病历数据泄露。11 月，凯撒医疗集团的一名员工非法访问了位于美国东中部几个州单位的患者医疗记录，导致患者信息和医疗信息泄露。以上案件[1]的发生导致患者医疗健康隐私暴露，使得用户对第三方的数据访问控制能力和存储能力存疑。

基金项目：江苏省石化过程关键设备数字孪生技术工程研究中心（苏发高高新发〔2019〕1125 号）

This work was supported in part by Jiangsu Province Petrochemical Process Key Equipment Digital Twin Technology Engineering Research Center (Su Fagagao Hi-Tech Development [2019] No. 1125)

通信作者：张华君(zhang.huajun@cczu.edu.cn) 徐守坤（xsk@cczu,edu,cn）

保障电子病历存储的机密性、不可篡改性和完整性对于医疗安全和服务质量至关重要。首先，电子病历存储的机密性可确保患者的就医隐私，维护患者权益，避免敏感信息泄露。其次，在保障患者权益的前提下，又要防止患者和他人对电子病历的恶意篡改，电子病历存储不完整和易篡改会影响医生的诊断和患者的治疗，导致医闹事件的发生，造成不良的就医环境。因此，如何保障电子病历在存储和共享过程中的安全成为了学者们争相研究的重点。

考虑到存储资源有限，医院往往会将电子病历数据存储在第三方云设备中。文献[2]基于云的框架并利用 Hadoop 进行离线数据存储和索引构建，对数据进行分析帮助用户自主判断患病状况。文献[3]在医疗云计算背景下提出了两种动态可搜索对称加密方案，实现医疗数据共享。但中心化的云存储存在许多安全和隐私隐患，诸如参与方信任问题，单点存储的故障问题，数据注入攻击，sybil 攻击漏洞等[4]。

而区块链是一种具备去中心化特性的分布式数据库，具有不可篡改、可验证和可追溯等多项良好的性质，可也解决单点故障和参与方的信任问题。文献[5,6]基于区块链采用联盟链和私有链实现电子病历安全存储和共享。文献[7,8]也通过构造患者和医生的双链实现病历数据和诊断数据的上传下载。文献[9,10]采取“链上+链下”的存储方式减轻患者或用户的存储压力，但以上方案没有考虑到恶意访问的状况。

访问控制是实现系统安全和隐私保护的重要手段之一，旨在限制未经过授权的系统访问行为。密文策略的访问控制(Ciphertext-Policy Attribute-Based Encryption，CP-ABE)被视为一种细粒度访问控制机制[11]。数据所有者可以制定访问策略，只有满足要求的访问者才能使用其私钥解密。但授权用户者有着和非法访问者的共谋活动的可能，为此 CP-ABE 增加了追踪和撤销机制，进一步加强敏感数据的安全性和可控性。文献[12]按照敏感度对医疗数据分级，采用区块链和 CP-ABE，解决数据的安全存储与共享问题。文献[13]为防止用户共谋的问题提出可追溯的 CP-ABE 方案，将私钥发布在区块链上进行交易。文献[14]利用可追踪的 CP-ABE 方案，追踪恶意泄露密钥的用户。文献[15]利用区块链存储电子病历并实现细粒度的访问控制，通过密钥生成中心可撤销恶意用户的访问权限。文献[16]利用区块链保证数据存储安全，采用基于属性的访问控制模型并用智能合约定义系统流程和访问策略。文献[17]利用以太坊技术提出了一种新的具有访问控制的安全云存储框架。文献[18]利用智能合约来执行基于角色的访问控制策略。文献[19]将基于属性的访问控制策略写进智能合约并将它们部署在区块链上，从而将访问策略评估过程转变为完全分布式的智能合约执行。

但访问控制写进区块链的智能合约中会使得访问控制的策略不可更改和难以扩展，无法满足灵活变化的场景。访问策略过于复杂时会降低区块链中节点的运行效率。

针对目前电子病历数据在存储和共享中存在的问题和挑战，本文在总结和借鉴前人研究成果的基础上，提出了新的解决方案和思路，更好地促进电子病历数据的安全访问和保护，从而维系患者与医生的良好关系，避免可能导致医疗纠纷的情况发生。本文的贡献主要有以下几个方面：

本文结合星际文件系统(Inter Planetary File System, IPFS)，区块链和可追踪的密文策略的属性基加密(Traceable Ciphertext-Policy Attribute-Based Encryption, T-CP-ABE)构造了一种安全的电子病历数据存储与共享方案。患者将病历数据加密存储在分布式数据库 IPFS 上，减轻患者和区块链上的存储压力。链上存储有关信息，实现数据的可信存储与不可篡改，同时避免中心化存储带来的单点故障问题，

本文将访问控制与区块链进行分离，使得访问策略更易维护和更新，设置了 T-CP-ABE 方案，防止授权访问者与未授权者的共谋问题，以便及时更新密钥。经过理论和实验分析，本方案是安全和切实可行的。

表 1 相关工作

	区块链	智能合约	访问控制方式	存储方式
[9]	√	√	×	IPFS
[10]	√	×	×	IPFS
[17]	√	√	CP-ABE	云
[18]	√	√	RBAC	区块链
[19]	√	√	CP-ABE	×
[20]	√	×	CP-ABE	IPFS
[21]	√	√	CP-ABE	区块链
本文	√	×	T-CP-ABE	IPFS

2 相关知识

2.1 区块链

区块链作为一个分布式记录、存储和更新交易的公共账本，有着去中心化、公开透明、无法篡改和安全可信等特点。通过哈希指针连接前后区块，形成不可篡改的链。事务按照时间顺序附加在矿工创建的区块中。区块链解决了权力集中的问题，确保所有参与者在无需第三方建立信任的前提下分布式执行事务。区块链由两部分组成分别是块头和块身。块头存放上一区块的哈希值，随机值，时间戳和 Merkle 树的根哈希值等信息。Merkle 树的数据结构作为一种特殊的二叉树，具有高效的验证性能，可快速验证某个特定数据信息是否存在，并且保证存放数据的不可篡改。这种构造节省了区块链的存储空间和开发新区块的成本，提高了区块链的运行效率和扩展性。

2.2 相关定义

定义 1．双线性映射． G,G_T为 q阶的乘法循环群， g是群 G的生成元。双线性映射 $e:G\times G\to G_T$，满足以下性质：

双线性：对任意 $u,v\in G$， $a,b\in Z_p$，有满足 $e(u^a,v^b)=e(u,v)^{ab}$；

非退化性：有 $u,v\in G$满足 $e(u,v)\neq 1$;

可计算性：对任意 $u,v\in G$，存在有效算法计算 $e(u,v)$。

定义 2．访问结构[22]。 访问结构 有属性集 $U=\{att_1,...,att_n\}$和一组属性值 $S_i=\{A_{i,1},A_{i,2},...,A_{i,n_i}\}$，其中 n_i是满足 $att_i\in U$ 的属性值的数目。 $L=[L_1,L_2,...,L_n]$是满足 $L_i=A_{i,t_i}\in S_i$， $t_i\in\{1,2,...,n_i\}$。有访问策略 $W=[W_1,W_2,...,W_n]$，其中 $W_i\subset S_i$。当属性列表 L满足访问策略 W则用 $L_i\in W_i(\forall i=1,2,..,n)$表示。

2.3 CP-ABE 算法

在 CP-ABE 算法中，用户的密钥对应一组属性的集合，而密文中包含访问结构或策略。只有当用户的属性集合满足访问结构时，才能正确解密密文。CP-ABE 算法一般可抽象概括为四个步骤：

1. $Setup(\lambda)$：给出安全参数生成公钥 *PK* 和主密钥 *MSK*;
2. $KeyGen(PK,MSK,L)$：给出公钥、主密钥以及数据请求者的属性集合 *L*，生成私钥 *SK*;
3. $Encrypt(PK,M,W)$：给出公钥、明文 *M* 和访问策略 *W*，得到密文 *CT*;
4. $Decrypt(PK,CT,SK)$：数据请求者拥有私钥、公钥和密文，对密文进行解密。只有满足访问策略的请求者可获得明文 *M*。

3 系统模型

本文基于区块链和可追踪的 CP-ABE 医疗数据共享方案分为访问控制和数据存储两个部分，方案构造如图所示，其中参与方有：上传病历的患者，访问病历并上传诊断决策的医生，属性授权中心，星际文件系统 IPFS 和区块链。患者：患者拥有自身病历的控制权。因此可以通过设置访问控制权限来限制病历的访问者。当医生满足访问策略后，才能获取患者的个人信息，病历存储地址，以及解密病历和加密诊断信息的密钥。

医生：不同的医生拥有不同的权限，当医生仅访问患者的病历不提供诊断服务时，患者不会为医生分配用于加密诊断结果的密钥。只有满足访问策略的医生会获得一组密钥，用于验证医生的身份以及保证诊断结果的机密性和完整性。然而，满足访问策略的医生仍存在泄露访问密钥的风险，针对该问题将医生的身份信息嵌入密钥中以跟踪医生访问行为。

属性授权中心：在访问控制阶段，属性授权中心作为一个可信第三方，是用户和医生沟通的桥梁。由属性授权中心生成系统所需的系统参数和公钥，用于患者加密个人信息。此外，属性授权中心依据医生的属性集分配私钥。只有满足患者定义的访问策略的医生可以解密用户信息。

星际文件系统：用于存储患者加密后的病历数据以及医生加密后的诊断结果，减轻用户和医生的存储压力以及避免单点故障的问题

区块链：患者病历信息以及医生诊断结果上链，对密钥进行管理，追溯患者病历的访问和诊断记录，避免医闹纠纷的发生。

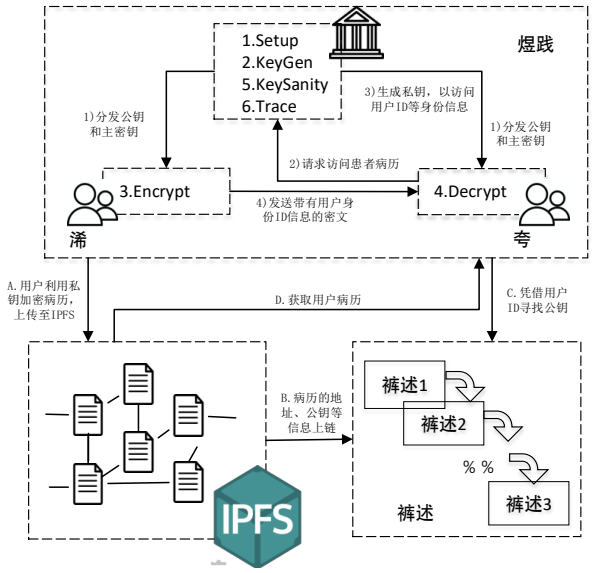


图 1 系统架构

4 安全模型

4.1 安全性

本文定义了一个安全模型，证明在选择明文条件下，方案是安全的。通过挑战者 C 和敌手 A 之间的游戏来表示，其中非法的数据访问者表示为敌手 A。具体过程如下：

初始化：挑战者 C 运行初始化算法后有系统参数 PK 和 MSK，把 PK 发送给 A。A 提交访问策略 W^* 给 C，A 规定了访问电子病历的访问策略。

阶段一：A 不断发送 $(did_1, L_1), ..., (did_q, L_q)$ 给 C，C 运行 $KeyGen$ 算法，发送私钥 $SK_{did_i, Li}$ 给 A。其中 L_i 不满足访问策略 W^*。

挑战：提交两个长度相等的明文 M_0 和 M_1。挑战者 C 随机选择 $b \in \{0,1\}$，加密明文 M_b，将密文 CT^* 交给 A。

阶段二：A 的执行过程同阶段一。

猜测：敌手输出 $b' \in \{0,1\}$，若 $b' = b$，则挑战成功。

若敌手在此次游戏获胜的概率定义为 $\Pr[b' = b] - 1/2$，而这个优势是可忽略的，那么该访问控制策略是安全的。

4.2 可追踪性

可追踪性是为了追查伪造或串通密钥的访问者，其中非法访问者是敌手 A。通过挑战者 C 和敌手 A 之间的游戏来证明方案具有可追踪性，具体如下：

初始化：挑战者 C 运行初始化算法后有系统参数 PK 和 MSK，把 PK 发送给 A。

查询：A 不断发送 $(did_1, L_1), ..., (did_q, L_q)$ 给 C，其中 (did_i, L_i) 与 $\{c_i, k_i\}_{i \in [q]}$ 是有关的。C 运行 $KeyGen$ 算法，发送私钥 SK_i 给 A。

密钥伪造：A 将输出一个解密密钥，该密钥与 (did, c, k) 有关。而 c 与 k 与 did 绑定。如果 $Trace(SK_*) \neq \perp$，即私钥 SK 形式完美且 (did, c) 在 $(did_1, c_1), (did_2, c_2), ..., (did_q, c_q)$ 中，而 $k \neq k_i$，那么 A 就赢得了这次游戏。A 赢得这场游侠的优势定义为 $\Pr(Trace(SK_*) \notin \{\perp, did_1, did_, , ..., did_a\}, k \neq k_i)$。

5 方案设计

5.1 访问控制

系统初始化 $Setup(\lambda)\to(PK,MK)$：已知 G_T和 G 为 p 阶的乘法循环群， g 是 G_0的生成元，选择随机数 $u,h,w,v\in G,(\alpha,x,y)\in Z_n^*$，有双线性映射 $e:G\times G\to G_T$。计算 $X=g^x,Y=g^y$。输出公钥和主密钥：

$$PK=((p,g,G,G_T,e),u,h,w,v,X,Y,e(g,g)^{\alpha}) \qquad MSK=(\alpha,x,y) \tag{1}$$

密钥生成 $KeyGen(PK,MSK,did,L)\to SK$：在该算法中，输入为属性授权中心提供的公钥 PK 和主密钥 MSK，医生自身的身份信息 did 和属性集 L，输出为属性授权中心为其分配的私钥 SK，用于解密患者的个人信息 M。过程如下：用户随机选择 $k\in Z_p^*$，计算承诺值 $R=w^k$，它将 R 发送给属性授权中心。属性授权中心随机选择 $d,r,r_1,r_2,...,r_n\in Z_n$并计算

$$\{K_{i,2}=g^{r_i}\}_{i\in[n]},\{K_{i,3}=(u^{A_{i,a_i}}h)^{r_i}v^{-r(x+did+yd)}\}_{i\in[n]}$$
$$K_0=g^{\alpha/(x+did+yd)}w^{rk},K_1=g^r,K_2=g^{xr},K_3=g^{yr},T_1=did,T_3=d \tag{2}$$

如果 $x+did+yd \bmod p=0$，则另外选择 d。输出私钥 $SK=(L,K_1,K_2,K_3,T_1,T_3,\{K_{i,2},K_{i,3}\}_{i\in[n]})$，其中 $L=\{A_{1,a_1},A_{2,a_2},...,A_{n,a_n}\}_{a_i\in[n_i]}$。用户将 $T_2=k$加入私钥中得到

$$SK=(L,K_1,K_2,K_3,T_1,T_2,T_3,\{K_{i,2},K_{i,3}\}_{i\in[n]}) \tag{3}$$

数据加密 $Encrypt(PK,M,W)\to CT$：该算法的输入为属性授权中心分配的公钥 PK，访问结构 $W=[W_1,W_2,...,W_n]$，其中 $W_i\subset S_i$。明文数据 $M=\{uid,address,(sk_D,pk_D),(Rk,Wk)\}$。其中 uid 为患者信息， $address$为病历存储在 IPFS 的文件地址， (sk_D,pk_D)为授权医生所需的密钥对，访问用户病历数据的密钥 Rk 和加密病历诊断结果的密钥 Wk，授权的医生都拥有密钥 Rk，但秘钥 Wk 只交付给负责实际诊断的医生。具体的加密过程如下：

选择 $s,s_1,s_2,...,s_{n-1}\in Z_n$计算 $s_n=s-\sum_{i=1}^{n-1}s_i$， s为要分享的秘密值，随机选择 $t_1,t_2,...,t_n\in Z_n$并计算

$$C=Me(g,g)^{\alpha s},C_0=g^s,C_1=g^s,C_2=g^{xs},C_3=g^{ys}\ \{C_{i,1}=w^{s_i}v^{t_i},C_{j,3}=g^{t_i}\}_{1\le i\le n} \tag{4}$$

当 $A_{i,a_i}\in W_i$， $\{C_{i,a_i,2}=(u^{A_{i,a_i}}h)^{t_i}\}_{1<i<n,1<a_i<n_i}$。否则 $C_{i,a_i,2}\in G$，输出密文：

$$CT=(C,C_1,C_2,C_3,\{C_{i,1},C_{j,3}\}_{1\le i\le n},\{C_{i,a_i,2}\}_{1\le i\le n,1\le a_i\le n_i}) \tag{5}$$

数据解密 $Decrypt(PK,CT,SK)\to M$：医生输入公钥 PK，患者发送的密文 CT 和自身私钥 SK，获取患者病历数据。计算明文：

$$B=\frac{e(C_1^{T_1}C_2C_3^{T3},K_0)}{(\prod_{i=1}^{n}e(C_{i,1},K_1^{T_1}K_2K_3^{T3})e(C_{i,2},K_{i,2})e(C_{i,3},K_{i,3}))^k}=e(g,g)^{\alpha s} \tag{6}$$

其中 $C_{i,2}=C_{i,a_i,2},M=C/B$。

跟踪算法 $Trace(SK)\to did/\perp$：若待检查的私钥 SK 不是一个形式完美的解密密钥，则输出 $\perp$。否则输出 $T_1=did,T_2=k$。属性授权中心检查 k 的值是否与 $T_1=did$的医生的一致。若属性授权机构输出 did，则表示该密钥是不合法的。

5.2 数据存储

考虑到用户存储资源有限，便采用“链上+链下”的存储方式。

在链下方面，本文采用 IPFS 分布式存储的方式，防止单点故障的问题。考虑到患者直接将病历数据以明文的

形式传输到 IPFS 上是不安全的，IPFS 并不保证传输路径上的可靠性与机密性。因此采用 AES 对称加密方式，生成对称密钥 Rk，对患者的病历进行加密 $Enc(Rk,data)=C$后再上传分布式数据库 IPFS 中，以确保数据安全。实行 T-CP-ABE 方案后，若有医生泄露密钥，患者可以重新在 IPFS 上建立新的加密病历数据，及时更新对称密钥 Rk。撤销了不良医生的访问权限，保证病历数据的后向安全。

IPFS 采用内容寻址的方式，其节点采用 Merkle 有向无环图的格式分块存储，并用根哈希来表示文件。因此电子病历以密文形式生成的哈希值 $address$会作为电子病历数据的唯一标识符。患者将该值作为明文的一部分交付给访问的医生，以便查看患者过往的就诊信息。每一位授权的医生都拥有访问的权限，获得访问密钥 Rk。但只有实际对患者病历进行诊断或就医的医生会获得由用户生成的密钥 Wk，用于加密诊断记录，而加密后的诊断信息也会存入 IPFS 中。

在链上部分,存放信息如下；由于区块链上的每个区块分为块头和块身两部分，块头记录了上一区块的哈希值，随机值，时间戳和 Merkle 树的根哈希值等信息。按照链式结构连接前后的区块。该部分由轻节点负责，以促进区块之间的运转。全节点负责维护链上信息，利用共识机制对区块进行验证。

在块身部分，加密的病历数据的 IPFS 地址 *address* 经 SHA256 哈希，得到

$$HIPFS = Hash_{SHA256}(address) \tag{7}$$

$HIPFS$此外，用户会对每次诊断或访问过的医生所生成的密钥进行管理和上链，过程如下，；

$$DEnc = Enc(pk_{D},(Rk,Wk),did) \quad UEnc = Enc(pk_{u},(RK,Wk)) \quad S = Sig(sk_{u},H(DEnc,UEnc)) \tag{8}$$

患者的 uid，访问时间，用户对密钥的签名 S， $HIPFS$发送至全节点将以上信息经 SHA256 哈希后得到的哈希值按照 Merkle 树的数据结构的方式形成根哈希存放在块头上。

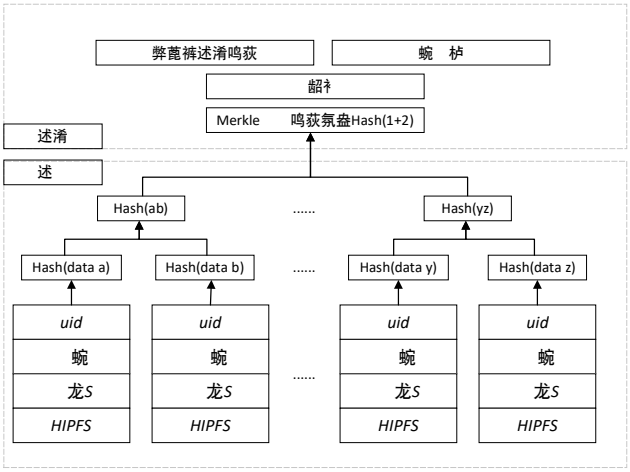


图 2 区块链数据存储模型

6 协议分析

6.1 正确性分析

解密阶段，根据私钥 SK 和密文 CT

$$B = \frac{e(C_1^{T_1}C_2C_3^{T3},K_0)}{(\prod_{i=1}^{n}e(C_{i,1},K_1^{T_1}K_2K_3^{T3})e(C_{i,2},K_{i,2})e(C_{i,3},K_{i,3}))^k}$$

$$= \frac{e(g^{sdid}g^{xs}g^{ysd},g^{\alpha/(x+did+yd)}w^{rk})}{(\prod_{i=1}^{n}e(w^{s_i}v^{t_i},g^{rdid}g^{xs}g^{yrd})e((u^{A_{i,a_i}}h)^{-t_i},g^{r_i})e(g^{t_i},(u^{A_{i,a_i}}h)^{r_i}v^{-r(x+did+yd)}))^k}$$

$$= \frac{e(g,g)^{\alpha s} e(g,w)^{(x+did+yd)srk}}{(\prod_{i=1}^{n} e(w^{s_i}, g^{rdid} g^{rs} g^{yrd}))^{k}} = \frac{e(g,g)^{\alpha s} e(g,w)^{(x+did+yd)srk}}{e(g,w)^{(x+did+yd)rk\sum_{i=1}^{n} s_i}}$$

$$= e(g,g)^{\alpha s}$$

$$C / B = \frac{Me(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} = M$$

因此，协议是正确的。

6.2 安全性分析

在系统模型部分简单刻画了患者和医生各自的行为，现对他们可能产生的恶意行为进行分析；

情况 1　不可篡改性

若患者篡改电子病历，满足访问策略的医生获取病历数据在 IPFS 的存放地址并读取患者的病历数据。医生利用患者发放的对称密钥对病历数据再次加密后，比较该地址是否与获得的明文中的 *address* 一致。因为 IPFS 是采用内容寻址的方式对病历数据进行分布式存储，因此数据一旦被篡改，则会导致 *address* 的改变。同样的 *HIPFS* 也会发生改变，导致与区块链存储数据地址不一致的情况出现。而医生只有读取病历的权限和编写诊断结果的权限，没有篡改电子病历的权限。

只要医生或合谋的非法访问者访问患者的病历，就会记录在区块链上。而区块链不可篡改的特性，使得他们无法删除浏览记录。因此方案为患者的就医和医生的诊断提供了客观公正的环境。

情况 2 抗合谋攻击

医生获取了患者的密钥后，为谋取更多利益将密钥泄露给他人，造成数据泄露。该种情况可以利用协议的可追踪性对医生分配的私钥进行追踪。以便患者后期更新访问密钥 Rk 和撤销医生后续的访问资格。协议的选择明文下的安全性和可追踪性在文献[23]中已给出。

情况 3 抗单点故障

本文采用区块链和 IPFS 技术，避免了对云等第三方存储机构的依赖，增强了方案的鲁棒性。以上两个分布式技术将数据存储在多个节点上，即使某个节点发生故障，仍旧可以从其他节点获取传输数据，防止单个节点故障带来的经济损失。

6.3 性能分析

本文使用的实验器材 PC 参数如下：Intel(R) Core(TM) i7-1165G7 CPU @ 2.80GHz 2.80GHz and 16GB of RAM、64 位 Windows 10 操作系统。首先，实验对不同大小的文件进行上传和下载的效率测试。在 PC 上安装了 IPFS 的安装包，在 Eclipse 配置 IPFS 所需的环境依赖后进行测试。通过对单一文件多次上传/下载的时间取平均值，再对文件进行扩容得到图的结果。实验表明，IPFS 上传下载的时间是符合实际生活的要求，且保证了数据的机密性和抗单点故障问题。

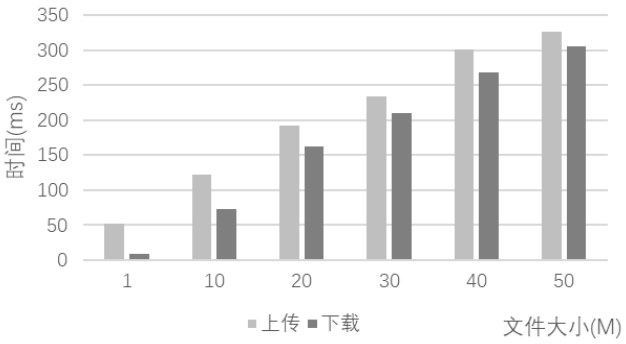


图 3 IPFS 上传下载文件的时间消耗

接下来对 T-CP-ABE 的性能进行比较。首先对下表所需符号进行说明：S 表示访问者所对应的属性大小，K 表示访问控制策略的大小，I 表示满足访问策略的属性的数量，l 表示生成矩阵的行数，n 表示生成矩阵的列数，t_e 表示指数运算所需时间，t_m 表示乘法运算所需时间，t_p 表示配对运算所需时间。文献[24]针对访问控

制策略的设计出发基于 TypeIII 配对群的标准假设，设计了 CP-ABE 方案。但该方案的局限性在于没有可追踪的特性。而文献[13]提出可追踪的 CP-ABE 方案，解决了一组不诚实用户在满足访问结构的情况下相互串通的问题。表二所示，文献[13,24]和本文相比，在公钥和私钥所需的密钥长度方面，随着属性和访问策略中属性的增加线性增长。但与本文相比，文献[13]在密文和密钥的大小上增长更快，即通信开销更大。相比文献[24]本文在私钥上的通信开销要小。

表 2 通信开销与计算开销比较

	通信开销			计算开销		
	公钥	私钥	密文	Key generation	Ciphertext generation	Decryption
[24]	3	6+3S	4+3K	(15+9S)t_e+(9+6S)t_m	(5+9|l×n|+6l)t_e+(2+6l+3|l×n|)t_m	6It_e+(2+6I)t_m+6t_p
[13]	5	2+3S	1+4K	(4+2S)t_e+(1+S)t_m	(2+5l)t_e+(1+2l)t_m	7It_e+(1+5I)t_m+4It_p
本文	8	7+2S	4+3K	(6+3S)t_e+2St_m	(6+2l)t_e+(1+2l)t_m	(3+4I)t_e+(2+2I)t_m+(1+2I)t_p

计算开销上，在解密算法中本文所需的配对计算只需 $2I$ 次。由于加密和解密是在用户本地计算，因此如图 4 和图 5 展示了随着访问策略和访问者属性的增加，方案在加密和解密算法下时间的变化。在加密算法中本文方案优于文献[13,24]，而在解密算法中本文所花费的时间是在可接受的范围内。因此本文的性能符合现实需要，是切实可行的。

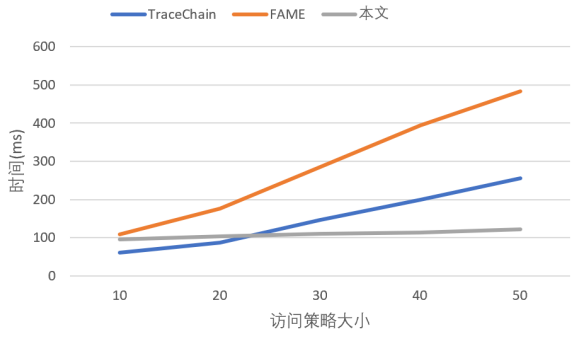


图 4 加密算法时间消耗

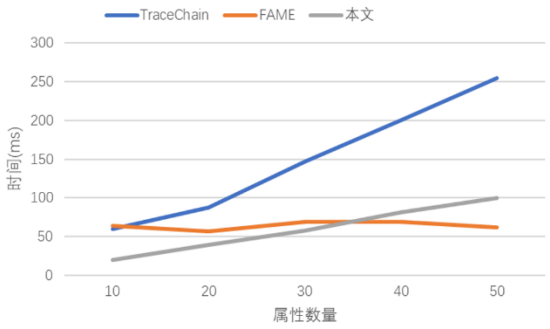


图 5 解密算法时间消耗

7 结语

本文基于区块链和可追踪的 CP-ABE 技术，为医疗数据建立隐私保护机制，防止用户和医生随意篡改已有的病历数据，从而影响医疗事故的判定。该方案的优势在于将访问控制策略放在链下，便于策略的修改与扩展。采用可追踪的 CP-ABE 确保数据的可追溯性和访问权限的设定，及时撤销合谋访问者的密钥。将有关的数据信息加密存储在 IPFS 上，避免用户存储资源不足和单点故障问题。将加密的病历数据的摘要存放在区块链上，便于患者和医生验证数据的准确性和完整性。

8 参考文献

[1]Top 10 privacy and cybersecurity stories of 2022[EB/OL]//Healthcare IT News. (2022-12-29). https://www.healthcareitnews.com/news/top-10-cybersecurity-stories-2022.

[2]LIN W, DOU W, ZHOU Z, et al. A cloud-based framework for Home-diagnosis service over big medical data[J/OL]. Journal of Systems and Software, 2015, 102: 192-206. DOI:10.1016/j.jss.2014.05.068.

[3]LI H, YANG Y, DAI Y, et al. Achieving Secure and Efficient Dynamic Searchable Symmetric Encryption over Medical Cloud Data[J/OL]. IEEE Transactions on Cloud Computing, 2020, 8(2): 484-494. DOI:10.1109/TCC.2017.2769645.

[4]GRAMMATIKIS P I R, SARIGIANNIDIS P G, MOSCHOLIOS I D. Securing the Internet of Things: Challenges, threats and solutions[J]. Internet of Things, 2019, 5: 41-70.

[5]WU G, WANG Y. The security and privacy of blockchain-enabled EMR storage management scheme[C/OL]//2020 16th International Conference on Computational Intelligence and Security (CIS). IEEE Computer Society, 2020: 283-287. https://www.computer.org/csdl/proceedings-article/cis/2020/044500a283/1t90uPa1Yuk. DOI:10.1109/CIS52066.2020.00067.

[6]ZHANG Y, LIN S, ZHAO J. SV-DEMR: An Electronic Medical Record Data Sharing Scheme Based on Searchable and Verifiable Encryption via Consortium Blockchain[C/OL]//2022 International Conference on Blockchain Technology and Information Security (ICBCTIS). IEEE Computer Society, 2022: 123-126. https://www.computer.org/csdl/proceedings-article/icbctis/2022/969100a123/1GqgYfWDyZG. DOI:10.1109/ICBCTIS55569.2022.00038.
[7]XU J, XUE K, LI S, et al. Healthchain: A Blockchain-Based Privacy Preserving Scheme for Large-Scale Health Data[J/OL]. IEEE Internet of Things Journal, 2019, 6(5): 8770-8781. DOI:10.1109/JIOT.2019.2923525.
[8]赵剑, 董文华, 安天博, 等. 面向医疗隐私保护的双区块链模型研究[J]. 信息安全研究, 2021, 7(2): 136-144.
ZHAO Jian, DONG Wenhua, AN Tianbo, et al. Research on Dual Blockchain Model for Medical Privacy Protection[J]. Information Security Research, 2021, 7 (2): 136-144.
[9]RIFI N, RACHKIDI E, AGOULMINE N, et al. Towards using blockchain technology for eHealth data access management[C/OL]//2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME). 2017: 1-4. DOI:10.1109/ICABME.2017.8167555.
[10]AZBEG K, OUCHETTO O, ANDALOUSSI S J. Access Control and Privacy-Preserving Blockchain-Based System for Diseases Management[J/OL]. IEEE Transactions on Computational Social Systems, 2022: 1-13. DOI:10.1109/TCSS.2022.3186945.
[11]BETHENCOURT J, SAHAI A, WATERS B. Ciphertext-Policy Attribute-Based Encryption[C/OL]//2007 IEEE Symposium on Security and Privacy (SP '07). Berkeley, CA: IEEE, 2007: 321-334. http://ieeexplore.ieee.org/document/4223236/. DOI:10.1109/SP.2007.11.
[12]曹萌, 余孙婕, 曾辉, 等. 基于区块链的医疗数据分级访问控制与共享系统[J]. 计算机应用, 2023, 43(5): 1518-1526.
CAO Meng, YU Sunjie, ZENG Hui, et al. Hierarchical access control and sharing system of medical data based on blockchain[J]. Computer Applications, 2023, 43 (5): 1518-1526.
[13]FAN Y, LIN X, LIANG W, et al. TraceChain: A blockchain-based scheme to protect data confidentiality and traceability[J]. Software: Practice and Experience, 2022, 52(1): 115-129.
[14]曾志兵, 吴晓鸽, 凌捷. 结合区块链和可追踪 CP-ABE 的数据存储与共享研究[J/OL]. 小型微型计算机系统: 1-8. DOI:10.20009/j.cnki.21-1106/TP.2021-0623.
ZENG Zhibing, WU Xiaoling, LING Jie. Research on data storage and sharing combining blockchain and traceable CP-ABE[J]. Small Micro Computer Systems: 1-8.
[15]XU S, NING J, LI Y, et al. A Secure EMR Sharing System With Tamper Resistance and Expressive Access Control[J/OL]. IEEE Transactions on Dependable and Secure Computing, 2023, 20(01): 53-67. DOI:10.1109/TDSC.2021.3126532.
[16]LI D, HAN D, CRESPI N, et al. A blockchain-based secure storage and access control scheme for supply chain finance[J/OL]. The Journal of Supercomputing, 2023, 79(1): 109-138. DOI:10.1007/s11227-022-04655-5.
[17]WANG S, WANG X, ZHANG Y. A secure cloud storage framework with access control based on blockchain[J]. IEEE access, 2019, 7: 112713-112725.
[18]SAIDI H, LABRAOUI N, ARI A A A, et al. DSMAC: Privacy-aware Decentralized Self-Management of data Access Control based on blockchain for health data[J/OL]. IEEE Access, 2022, 10: 101011-101028. DOI:10.1109/ACCESS.2022.3207803.
[19]DI FRANCESCO MAESA D, MORI P, RICCI L. A blockchain based approach for the definition of auditable Access Control systems[J/OL]. Computers & Security, 2019, 84: 93-119. DOI:10.1016/j.cose.2019.03.016.
[20]SUN J, YAO X, WANG S, et al. Blockchain-Based Secure Storage and Access Scheme For Electronic Medical Records in IPFS[J/OL]. IEEE Access, 2020, 8: 59389-59401. DOI:10.1109/ACCESS.2020.2982964.
[21]GAO S, PIAO G, ZHU J, et al. TrustAccess: A Trustworthy Secure Ciphertext-Policy and Attribute Hiding Access Control Scheme Based on Blockchain[J/OL]. IEEE Transactions on Vehicular Technology, 2020, 69(6): 5784-5798. DOI:10.1109/TVT.2020.2967099.
[22]EMURA K, MIYAJI A, NOMURA A, et al. A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length[M/OL]//Bao F, Li H, Wang G. Information Security Practice and Experience: Vol. 5451. Berlin, Heidelberg:

Springer Berlin Heidelberg, 2009: 13-23. http://link.springer.com/10.1007/978-3-642-00843-6_2. DOI:10.1007/978-3-642-00843-6_2.

[23]LI J, ZHANG Y, NING J, et al. Attribute Based Encryption with Privacy Protection and Accountability for CloudIoT[J/OL]. IEEE Transactions on Cloud Computing, 2022, 10(2): 762-773. DOI:10.1109/TCC.2020.2975184.

[24]AGRAWAL S, CHASE M. FAME: fast attribute-based message encryption[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 665-682.

杨琴：女，硕士研究生，研究方向大数据安全与隐私保护.

张华君：男，博士，硕士生导师，研究方向信息安全、密码学、区块链、隐私保护的大数据技术.

石林：男，教授，硕士生导师，研究方向大数据、物联网、深度学习.

徐守坤：男，教授、博士生导师，研究方向深度学习、自然语言处理、数字孪生技术.

MRFI: An Open Source Multi-Resolution Fault Injection Framework for Neural Network Processing

Haitong Huang*†, Cheng Liu*, Xinghua Xue*†, Ying Wang*, Huawei Li*, Xiaowei Li*,
* Institute of Computing Technology, Chinese Academy of Sciences
† University of Chinese Academy of Sciences
{huanghaitong21s, liucheng,xuexinghua,wangying2009}@ict.ac.cn

Abstract—To ensure resilient neural network processing on even unreliable hardware, comprehensive reliability analysis against various hardware faults is generally required before the deep neural network models are deployed, and efficient error injection tools are highly demanded. However, most existing fault injection tools remain rather limited to basic fault injection to neurons and fail to provide fine-grained vulnerability analysis capability. In addition, many of the fault injection tools still need to change the neural network models and make the fault injection closely coupled with normal neural network processing, which further complicates the use of the fault injection tools and slows down the fault simulation. In this work, we propose MRFI, a highly configurable multi-resolution fault injection tool for deep neural networks. It enables users to modify an independent fault configuration file rather than neural network models for the fault injection and vulnerability analysis. Particularly, it integrates extensive fault analysis functionalities from different perspectives and enables multi-resolution investigation of the vulnerability of neural networks. In addition, it does not modify the major neural network computing framework of PyTorch. Hence, it allows parallel processing on GPUs naturally and exhibits fast fault simulation according to our experiments.

I. Introduction

Deep neural network models are increasingly used in safety-critical applications like autonomous driving and avionics, but computational errors caused by imperfect hardware such as process variations, defects and impurities in materials, and noise can lead to unexpected inference and cause severe consequences in these safety-critical applications. To guarantee resilient deep learning, it is crucial to thoroughly evaluate the reliability of neural network models before their deployment. While it is prohibitively expensive to perform fault injection to realistic chips, simulation based error injection tools such as PyTorchFI [1], Ares [2], TensorFI [3], TorchFI [4], and MindFI [5] have been extensively developed. These tools typically provide basic functionalities such as error injection and reliability analysis to general neural network processing and significantly alleviate the design efforts required for neural network reliability analysis in presence of various hardware errors.

Nevertheless, it has become evident that existing fault injection tools are insufficient to address complex neural network reliability analysis for the following aspects. First, they generally utilize hardware fault models such as bit flip model to simulate the influence of hardware errors on neural network processing, but they do not model any specific hardware architectures and the correlation between the neuron-wise error injection and realistic hardware-based fault injection remains unclear. The accuracy of the fault simulation needs to be calibrated such that it can be utilized as a common metric to reliability study of neural network processing. Second, there is a lack of standard error injection metrics and neural network evaluation metrics that can be utilized to align the fault simulation results and distinct fault simulation results may be obtained from these different fault injection tools, which dramatically affects the reliability study of neural network processing. For instance, many prior fault injection tools fail to clarify the model quantization during reliability evaluation [6] and biased evaluation results are obtained because of the quantization setups. Third, many of the fault injection tools are closely coupled with the neural network frameworks such as PyTorch. The error model and error injection may vary with the different neural network layers [7], which also complicates the fault simulation. In addition, fine-grained selective fault-tolerant design [8] has also been explored and it poses great challenges to the fault simulation and evaluation of existing fault injection tools. These unique fault simulation and evaluation requirements further make the fault injection frameworks difficult to scale and port to GPUs or other parallel computing systems.

To address the above problems, we present MRFI, a highly configurable open-sourced fault injection tool for reliability investigation and evaluation of neural network processing. On the one hand, it provides unified error models, neural network quantization setups, and error injection mechanisms to ensure consistent metrics of error injection and reliability evaluation. Moreover, it is compared to typical fault injection platforms with more detailed architectural details and calibrated for more fair comparison. On the other hand, it provides a multi-resolution vulnerability investigation of neural network processing from distinct angles and granularities, which can be utilized to guide selective fault-tolerant neural network design conveniently. In addition, MRFI utilizes a tree structure for flexible error injection configurations and

reliability evaluation of different neural network modules without modifying the neural network processing engines, which will not affect the underlying parallel computing engines on GPUs and can achieve the state-of-the-art fault simulation.

The contributions of this paper can be summarized as follows:

- We present an open-sourced multi-resolution fault injection framework MRFI and have it calibrated with architecture-specific fault injection frameworks.
- MRFI provides unified error injection and reliability evaluation with flexible simulation configurations. Particularly, it allows fine-grained vulnerability analysis from different angles and can be utilized for selective protection against various hardware errors.
- MRFI is built on top of PyTorch and does not affect the computing engine of PyTorch. Hence, the fault injection and reliability evaluation can be scaled to parallel computing systems like GPUs conveniently and achieve competitive simulation speed.

II. Related Work

Research on the reliability and fault tolerance of deep learning revolves around error simulation, fault evaluation, fault-tolerant model design, and fault-tolerant hardware design. Error simulation methods can be divided into hardware-level simulation and model-level simulation. Hardware-level simulation is generally more accurate but has limited generality and performance due to the need to simulate the underlying logic of specific hardware architectures. Notable works in this area include NVIDIA SASSIFI for fault injection simulation on GPUs [9] and fault simulation on FPGAs [8] [10] [11] [12]. These error simulation methods often require tens to thousands of times the normal execution time.

On the other hand, model-level evaluation and simulation have better performance, making them widely studied in previous works. Some notable works in this area include [1] [3] [2] [13] [5] [4]. Some works have shown that accurate hardware-level reliability results can be obtained through model-level simulation. For example, in Are [2], the authors obtained consistent measurements with silicon measurement by injecting errors at the model level. Yi He et al. [7] mapped hardware unit error rates to model neurons to obtain accurate model-level evaluation results. Xinghua Xue et al. [14] improved the accuracy of computational evaluation by focusing on operations. Some works aim to improve the performance of error simulation. For example, PyTorchFI [1] optimized the implementation of fault injection simulation, reducing the additional time overhead of single-point fault injection to a negligible level. The closest state-of-the-art tools to our work are PyTorchFI and Ares, which offer more comprehensive functionality compared to other works. Table I shows a summary of their features.

TABLE I
Comparison of different fault injection frameworks including MRFI, PyTorchFI [1], and Ares [2]

	MRFI	PFI	Ares
activaion injection			
permanent fault	✓	✓	×
transient fault	✓	✓	✓
weight injection			
permanent fault	✓	✓	✓
transient fault	✓	×	×
number of faults			
fixed number fault	✓	✓	×
random error rate	✓	×	✓
fault injection on quantized data			
quantized fault	✓	✓	✓
full quantization	✓	×	✓
quantization method selection	✓	×	×
quantization parameter settings	✓	limited	limited
fault injection on floating point data			
fixed value inject	✓	✓	×
random perturbation	✓	×	✓
random bit flip	✓	×	×
internal observers			
clean activations	✓	×	×
fault impact	✓	×	×
functional flexibility			
custom error model	✓	✓	×
multi-resolution evaluation	B,N,C,L[1]	N,L	L
fine-grained simulation configuration			
by python program	✓	limited	×
by configuration file	✓	×	×

[1] B: bit, N: neuron, C: channel, L: layer

For fault evaluation, recent works have proposed improved statistical metrics and mathematical modeling analysis to further reduce the number of error simulations required. In HarDNN [15], the authors explored various surrogate test metrics to replace simple model accuracy measurements, resulting in a 10-times reduction in simulation count under the same simulation accuracy. BinFI [16] models error propagation and introduces binary search to reduce the number of simulations. [17] achieved a two-order-of-magnitude speedup in fault evaluation by using statistical modeling. Previous fault injection frameworks did not consider these optimizations, whereas our framework's flexibility allows researchers to conveniently perform these evaluations.

After fault evaluation, safety-critical systems often require fault-tolerant design. Fault-tolerant design can be applied at both the software and hardware levels, such

as adjusting models for fault tolerance or introducing additional hardware redundancy computations [18] [19] [20] [21] [22]. These solutions include Selective Hardening [8], Algorithm-Based Fault Tolerance [23], adjustment of quantization parameter settings [6], adjustment of activation layer values to limit numerical magnitude [24], fault-tolerant retraining [25], and more. Evaluating these solutions requires a more flexible fault injection framework.

III. MRFI Framework

A. MRFI Overview

The overall structure of MRFI is illustrated in Figure 1. The framework as a whole can be divided into two parts: the configuration part and the execution modules, which are further divided into several functional modules.

MRFI provides two main approaches to configure different experimental requirements: EasyConfig for simple and preliminary experiments, and a detailed error injection configuration tree for more complex needs. For simple experiments, EasyConfig allows users to specify the layers and fault injection method through a config file. This configuration applies uniformly to the specified set of layers, enabling users to conduct fault injection experiments by loading the corresponding error injection configuration, similar to working with a typical PyTorch model. To address complex and precise error injection requirements without modifying the experimental model, MRFI employs a detailed error injection configuration tree. This tree structure corresponds to the nested structure of neural network modules in PyTorch, recording the error injection configuration of each module and layer. Users can easily adjust the parameters within the configuration tree to meet their specific needs. MRFI's error injection module then dynamically performs error injection based on the configured settings.

The execution modules of MRFI offer multiple commonly used injection methods that encompass the majority of error injection requirements for experiments. For enhanced flexibility, users can also customize the execution module according to their specific needs.

The structured configuration approach of MRFI offers several advantages over previous manual error injection implementations. In the traditional manual approach, experimenters typically need to modify both the neural network model code and the error injection parameters to accommodate different error injection requirements. This process is hard especially for modern large-scale models with numerous layers. Additionally, manually adjusting error injection parameters for multiple fine-grained settings in dozens of model layers is cumbersome, making experimental management challenging. MRFI overcomes these issues by decoupling the error injection configuration from the model definition and organizing it into separate modules, facilitating error injection experiments with specific configurations.

B. Integration with PyTorch

MRFI is built upon PyTorch, the widely adopted neural network training framework. Error injection in MRFI is achieved through PyTorch hook, allowing for seamless integration with the model. PyTorch serves as the foundation for MRFI due to its extensive user base and well-established software ecosystem. Additionally, we have found that PyTorch's dynamic graph approach aligns well with error injection scenarios that necessitate dynamic modification and configuration. The hook function in PyTorch enables dynamic observation and modification of intermediate data within network modules, making it highly suitable for fine-grained error injection experiment configuration. An alternative implementation approach involves adding or replacing layers in the model with fault injection operations. However, this method requires prior knowledge of the original layer implementations, which is impractical given the diverse range of user-defined modules and layers. It is worth noting that error injection at the model level only requires access to the model's weights and feature map information for modification. By leveraging hooks, rewriting of the underlying computation implementation is not necessary. Our execution module also mainly uses PyTorch tensors, which allows the error injection process to be naturally used on the CPU and GPU to maximize performance.

C. Major MRFI Execution Modules

As depicted in Figure 1, MRFI incorporates three key components for fault injection: quantizer, selector, and error model. Additionally, an observer feature is available for monitoring numerical values and their error impact within the model.

1) Quantizer: In MRFI, we recognize the significance of quantization methods in evaluating fault tolerance. Quantization is commonly employed during inference for fixed-point or integer computations, differing from the floating-point storage and computation used during model training. Previous studies highlight the varying fault-tolerance characteristics of different numerical representations and the influence of quantization choices on model accuracy and fault tolerance. To facilitate accurate fault simulation for quantized models, MRFI offers multiple configurable quantization options. These simulated quantizations encompass the quantization and de-quantization steps, converting tensors from floating-point to integer format and vice versa. The quantizer serves as a wrapper for the fault injection component, enabling the provision of integer tensors to the selector and error model for simulating integer computation and error occurrence on real hardware. It is worth noting that the quantizer is optional, and users can omit it if they solely require simulations with floating-point models.

2) Selector: The selector determines the locations within the tensor where errors are injected. By defining the selector, users can specify the manner in which errors

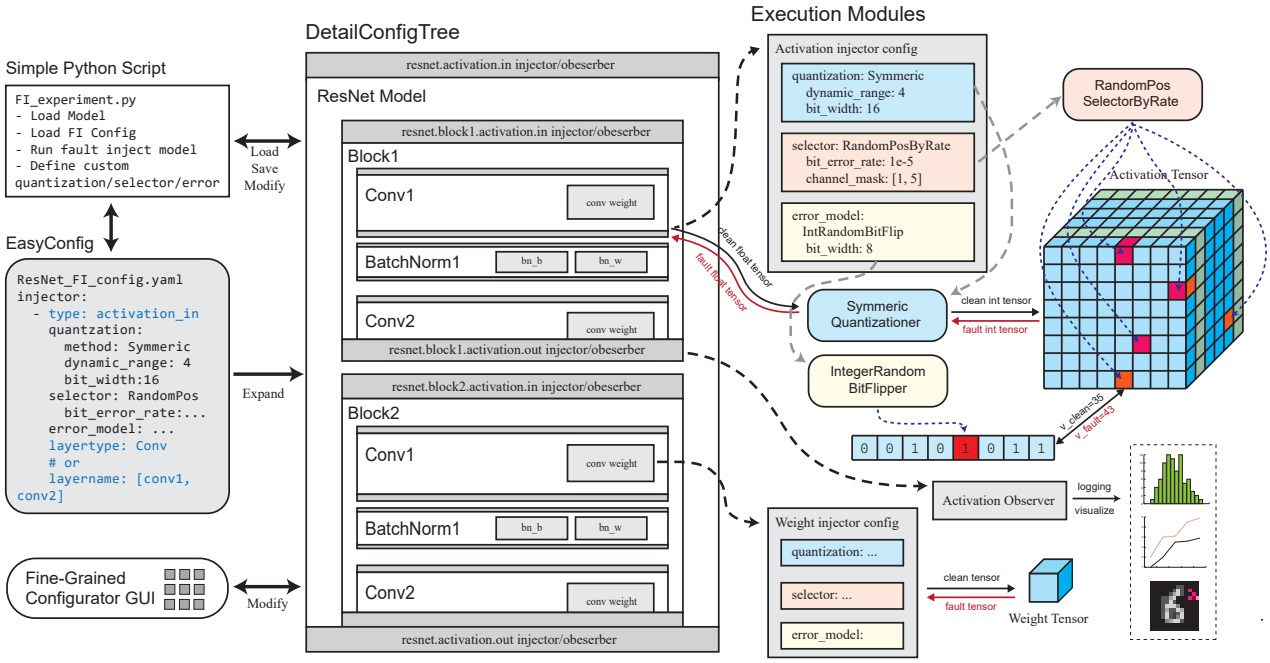


Fig. 1. The MRFI tool integrates multiple modules. The configuration section can be used for simple or detailed experiments. EasyConfig provides a unified annotation error configuration for multiple layers, while DetailConfigTree corresponds to each neural network module and can be modified separately. The execution module is decomposed into quantizers, selectors, and error models according to the requirements of the fault injection, and provides various predefined functions.

manifest. For example, a fixed position selector can be employed to evaluate the impact of permanent errors. Conversely, a random position selector serves as a random engine to simulate the occurrence of soft errors. Soft errors are often associated with a bit error rate, and the random position selector efficiently generates erroneous positions based on this rate. Users also have the flexibility to customize filtering masks at the selector level or implement custom selectors to simulate fine-grained selective fault-tolerant protection.

3) Error model: The error model dictates how errors are injected into the numerical values at the selected error positions. The error model may rely on the original numerical values, such as random bit flips for integers or floating-point numbers, fixed bit flips, or the addition of perturbations following a normal distribution. Alternatively, the error model may be independent of the original numerical values, involving stuck-at 0 faults, faults with specific fixed numerical values, or random uniform integers or floating-point numbers. MRFI incorporates these common error models, while also allowing users to define custom error models conveniently.

4) Observer: MRFI provides internal observers to enhance the visibility of numerical values and their error impact within the model. Observers that do not require fault injection can be employed to monitor the distribution of internal activation values. For example, an observer can track the maximum and minimum values to determine the dynamic range of activations, aiding in the determination of model quantization parameters. Another type of observer enables the comparison between fault injection and non-fault injection scenarios. Before initiating fault injection, this observer conducts a golden run comparison analysis. It can utilize a counter to tally the number of numerical values affected by fault injection or employ an RMSE observer to gauge the extent of error propagation.

D. Implementation details of MRFI

In this section, we explain several key points in the implementation of MRFI, which make it simple to use and powerful in functionality.

1) Human-readable configuration files: MRFI employs DetailConfigTree as its internal configuration for execution, which corresponds to each PyTorch module. EasyConfig, on the other hand, can match multiple layers and is expanded into DetailConfigTree before usage. Both EasyConfig and DetailConfigTree can be loaded, saved, and modified using human-readable YAML files. Figure 1 presents an example of EasyConfig YAML configuration, specifying the injection method, selected layer, and execution module parameters. DetailConfigTree uses similar configuration parameters but adopts a recursive structure, with each child node corresponding to a specific module or layer. For large deep models, creating and modifying DetailConfigTree manually can be challenging. Therefore, MRFI simplifies the initial stage by starting with a simple EasyConfig, parsing the PyTorch model, and performing

fuzzy or precise matching on the user-selected layers. The parameters from EasyConfig are then expanded onto the DetailConfigTree to facilitate the execution.

2) Dynamic fault injection execution: As mentioned above, MRFI utilizes PyTorch hooks to implement runtime fault injection of model variables. During the loading phase, hooks are inserted into the target model's modules and layers. During the experimental stage, when a hook with fault injection settings is executed, MRFI selects the corresponding execution module based on the DetailConfigTree configuration and invokes them in order. Notably, MRFI adopts a fully dynamic approach to execute configurations, allowing for dynamic modification of fault injection parameters during the experiment. This design maximizes flexibility, enabling users to easily conduct experiments with various parameters by dynamically adjusting the internal configuration within a loop. This eliminates the need for repetitive manual modifications of the text configuration file. By offering flexible and diverse methods, MRFI surpasses previous tools that offer limited or fixed injection error settings, effectively meeting diverse experimental needs.

3) Extensible execution modules: MRFI divides error injection into simple parts, it is easy to customize and add new functionality. For instance, a selector function receives the shape of a target tensor and generates fault injection positions, while the error model function processes the original tensor data and returns the error results based on the specified error model. As long as a function adheres to the defined interfaces, it can be written in the configuration and executed by MRFI. This extensibility enables researchers to tailor MRFI to their specific requirements and incorporate new error injection techniques with ease.

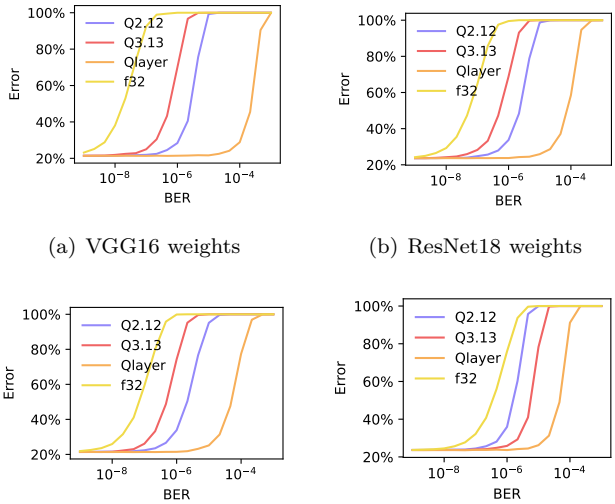
IV. Experiment Results

In this section, we present the results of various error injection experiments conducted using MRFI. Our focus is on demonstrating experimental approaches not covered by previous error injection tools, highlighting the necessity of utilizing a more fine-grained error injection framework.

For our experiments, we primarily employ VGG and ResNet models evaluated on the ImageNet dataset. In subsection IV-B, we also utilize a smaller dataset, cifar-10, to visualize error propagation.

A. Basic BER-Accuracy fault inject experiment

We begin by testing and evaluating MRFI's performance under simple configurations. Specifically, we conduct random bit flipping error injection on the quantization weight of the VGG16 model using the fixed-point quantization settings of 2.12 and 3.13. Additionally, MRFI supports different data types and quantization methods. For comparison, we include the original 32-bit floating point (f32) and layer-wise range quantization (Qlayer).



(a) VGG16 weights (b) ResNet18 weights

(c) VGG16 activations (d) ResNet18 activations

Fig. 2. Relation between bit error rates and classification error rate of different data type.

As illustrated in Figure 2(a)(b), our results of fixed-point quantization with settings 2.12 and 3.13 align with those obtained from Ares [2], exhibiting a significant increase in bit error rates at $BER = 10^{-6}$. In contrast, floating-point types exhibit lower error tolerance, while layered quantization demonstrates superior fault tolerance, corroborating the conclusions from SNR [6]. In Figure 2(c)(d), the injection of activation values exhibits a similar trend, but not entirely the same.

While the above experiments could be conducted separately using different fault injection tools in the past, the advantage of MRFI lies in its integration of diverse functionalities within a unified framework.

B. Observing error propagation

To gain deeper insights into error propagation within the network, MRFI provides a straightforward operation to apply error observers to all network modules. These observers enhance transparency and offer diverse perspectives for in-depth research on error impact.

Firstly, we employ a simple equation comparison observer to assess the number of neurons affected by a single injection. We perform one-bit flipping on the activation values and weights of the first convolution layer (conv1) of VGG11, and then calculate the average number of errors across 10,000 input images. The golden run accuracy and accuracy after activation injection both yield 76.49%, while the accuracy after weight injection slightly decreases to 76.34%. As shown in Figure 3, the number of affected neurons increases rapidly along the layers, with nearly all neurons in the fully-connected layers displaying differences from their original values. This indicates the network's resilience to errors, while also highlighting that the number

of affected neurons is not a good indicator of the severity of error effects.

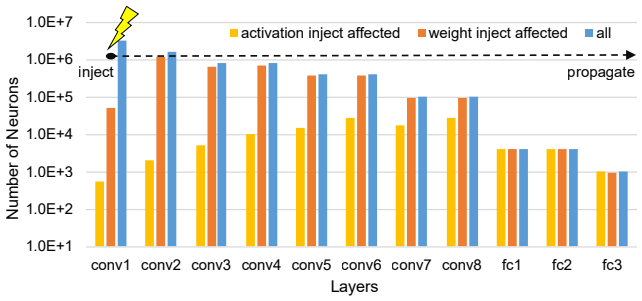
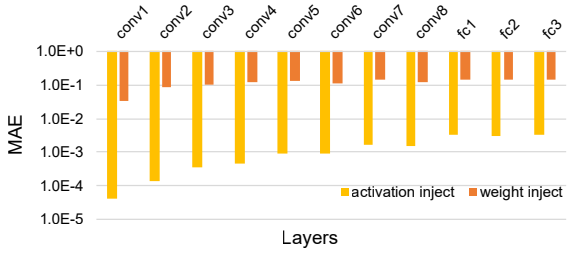
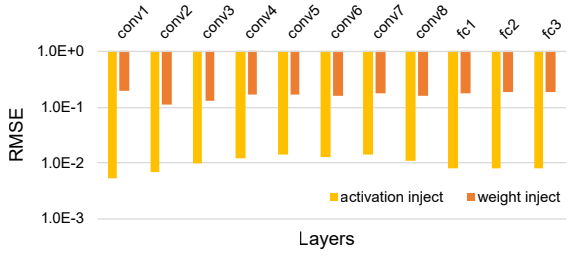


Fig. 3. Number of affected neurons of one bit flip on first layer in VGG-11.



(a) MAE



(b) RMSE

Fig. 4. Evaluate error propagation by better metrics.

Next, we evaluate error propagation using more robust metrics. Inspired by [15] [17], MRFI provides mean absolute error (MAE) and rooted mean square error (RMSE) observers. In Figure 4, both MAE and RMSE reveal that weight injection has a significantly greater influence than activation value injection when performing single-point injection on the first convolution layer. This finding aligns with the accuracy results. Additionally, although the number of affected neurons rapidly increases during the propagation process, these error measures remain relatively stable.

MRFI also includes observers for recording activation and error values directly, facilitating the visualization of internal feature maps in neural networks. Figure 5 showcases the raw neuron values input by LeNet on a single image from the cifar-10 dataset, along with the errors caused by a single activation injection in the first layer. We observe that errors occur in fixed positions across all channels in the convolution layer's feature map, while nearly all neurons in the fully connected layers experience irregular disturbances.

C. Quantization Settings Resilience Study

Recent studies have highlighted the close relationship between quantization methods and neural network fault tolerance. Therefore, our framework integrates quantization settings into its configuration. In subsection IV-A, we compared the differences between two quantization methods using a simple global quantization setting. With MRFI, we can now quantize each submodule of the network separately to study these differences at a fine-grained level.

Figure 6 illustrates the differences in RMSE metrics for various error injections on different layers and blocks of ResNet18. A larger RMSE value indicates greater sensitivity to errors. It is evident that global quantization settings exhibit significant variations in fault tolerance and perform poorly at each layer, resulting in lower fault tolerance than layered quantization. Conversely, layer-wise quantization demonstrates stable error resilience. Additionally, we observe that weights are more sensitive to errors during global quantization, whereas weights and activation values exhibit similar sensitivities during layerwise quantization.

The quantization parameters of the model also impact fault tolerance [6], and MRFI can easily test these settings. To demonstrate it, we employ layerwise quantization and set different quantization scaling factors for each layer of the model. Here we use a typical symmetric quantization, that is, $x_Q = clamp(round(x \times scale), Q_{min}, Q_{max})$. We evaluate accuracy and RMSE metric in ResNet as an example. As shown in Figure 7, larger scale factors generally lead to worse fault tolerance.

In practical deployment scenarios, models are typically quantized after training, and the quantization parameters depend on both model accuracy and target hardware requirements. The experimental differences outlined above emphasize the need for a thorough assessment of the quantization methods' impact when deploying security-critical neural network models.

D. Evaluate Fault-Tolerant Resilience at Different Level

MRFI's flexible configuration capabilities enable error injection analysis at various levels, providing robust support for selective fault-tolerant design. In this section, we demonstrate the differences in fault tolerance based on bit, channel and pixel.

1) Bit sensitivity evaluation: We inject random errors into each bit of the float16 floating-point type and fixed-point 3.13 quantization type of the VGG-11 model, observing the model's accuracy at different injection rates.

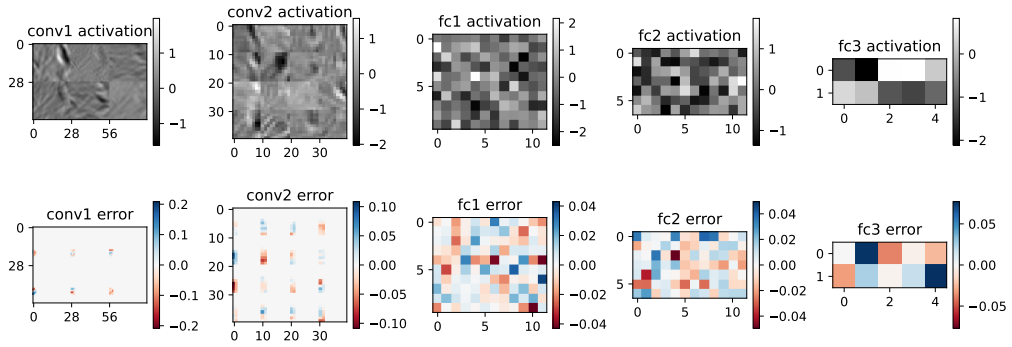


Fig. 5. Visualize of LeNet internal layers and error propagation.

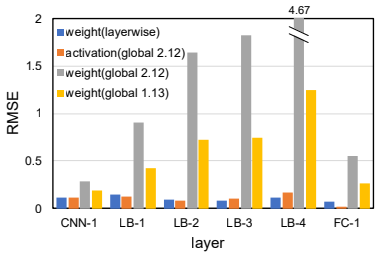
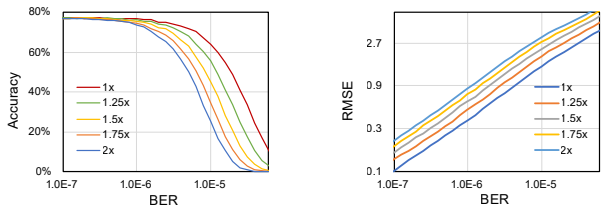


Fig. 6. The difference in error sensitivity of quantization methods at different network modules.



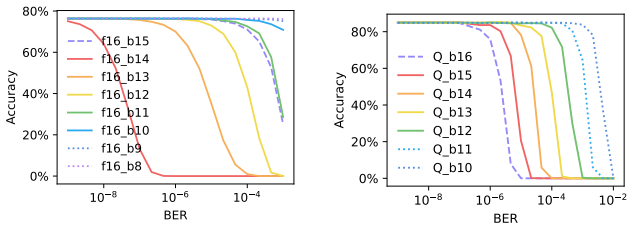
(a) Accuracy (b) RMSE

Fig. 7. Fault tolerance under different quantization scale parameter.

Figure 8 illustrates the model accuracy for the high-order bits, while the low-order bits demonstrate greater insensitivity to error injection. In float16, the highest bit represents the sign, followed by 5 exponent bits, and finally 10 mantissa bits. From Figure 8, we observe that high-order exponent bits are highly sensitive to errors, whereas error injection into the mantissa bits does not significantly affect the model's accuracy. The sign bit exhibits a sensitivity level lower than that of high exponent bits but higher than that of low exponent and mantissa bits. For fixed-point 3.13 quantization, a regular pattern emerges in the fault tolerance of each bit, with lower bits consistently displaying better fault tolerance than higher bits.

2) Channel sensitivity evaluation: Figure 9 showcases the differences in sensitivity indicators for each channel in the activation values of the first three layers of VGG-11. The three convolutional layers contain 64, 128, and 256 feature map channels, respectively. By employing layerwise quantization parameters, the differences in fault tolerance among different layers are small. After sorting the RMSE indicators for each channel in ascending order, we observe that the variance in channel sensitivity within a layer surpasses the variance in sensitivity between layers. This suggests that selective fault-tolerant design at the channel level can yield enhanced fault tolerance outcomes.

3) Pixel sensitivity evaluation: It is also possible to evaluate the pixel sensitivity differences of feature maps at the spatial level. We use MRFI to randomly inject errors into each channel of different pixels with different activation values in the ResNet two layers, resulting in the output RMSE as shown in Figure 10. We can see that pixels in the center position are usually more sensitive to errors, which may be related to the dataset features of ImageNet, as most targets are located at the center of the image.



(a) float16 (b) fixed 3.13 quantization

Fig. 8. Bit sensitivity evaluation of VGG-11 among all layers.

E. Performance

In addition to accuracy, the performance of error injection and simulation in neural networks is a crucial factor for conducting experiments. MRFI has implemented optimizations in error injection, resulting in comparable time costs between error injection and the golden run. Below, we will refer the state of the art tools PyTorchFI and Ares to conduct two similar experiments and measure the time consumption. PyTorchFI has been optimized for

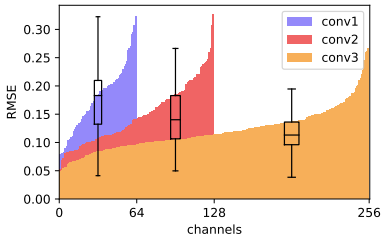


Fig. 9. Channel sensitivity evaluation on three convolution layers of VGG-11. We use the RMSE metric to test the fault injection sensitivity of each channel and sort them in ascending order. The box plot shows the median and quantile of the data, indicating significant differences between channels.

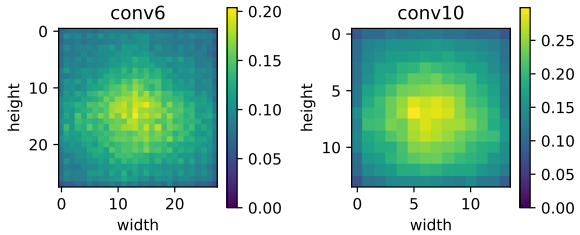


Fig. 10. Pixel sensitivity evaluation on two convolution layers of ResNet18 using RMSE metric. The data for each pixel comes from random bit injection along all channels of the feature map.

single point error injection, making its overhead negligible compared to forward propagation. However, PyTorchFI does not provide random error generation based on injection rate, so we compared it with the random sampling method used by Ares as the baseline.

1) Single point injection: Table II presents the evaluation time for single point injection errors on GPUs and CPUs. It demonstrates that the execution time of the MRFI framework can be disregarded relative to the model's computation. This indicates that although our tool is designed for complex injection configurations, it has also achieved target performance close to PyTorchFI.

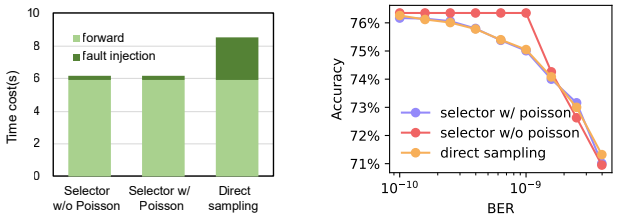
2) Injection according to bit error rate: For random multi-point injection, Ares conducts acceptance-rejection sampling on each bit, while MRFI uses selectors to choose injection error positions based on the expected number calculated using Equation 1. This reduces the number of random number generations required. However, a discrepancy arises when the injection error rate is low, as n_{inject} is a discrete integer. To address this issue, MRFI adopts the Poisson sampling strategy described in Equation 2, ensuring consistent results in theory. Figure 11 compares the performance and accuracy between direct probability sampling and optimization methods. While using a selector can significantly reduce the cost of sampling calculations, it may introduce jumps at low injection rates. On the other hand, Poisson sampling maintains consistent results with the baseline sampling methodology.

$$n_{inject} = \lfloor N_{neurons} \times BER \rceil \tag{1}$$

$$N_{inject} \sim Poisson(N_{neurons} \times BER) \tag{2}$$

TABLE II
Time cost comparison on single point injection

	Acc.	GPU time	CPU time
golden run	68.87%	86.83±0.06s	2603±2s
1p fault inject	68.33%	87.94±0.04s	2607±3s



(a) Time cost (b) Accuracy

Fig. 11. Comparison of performance and accuracy of different random sampling methods for random multi-point injection.

V. Conclusion

In this paper, we have introduced MRFI, an open-source multi-resolution fault injection framework for neural network processing. MRFI enables refined error assessments, considering the impact of model quantization and supporting fine-grained error injection configurations. To simplify configuration, MRFI decompose complex configuration requirements into multiple modules and offers a user-friendly configuration interface. Our experiments have shown that MRFI can evaluate errors from multiple aspects with practical needs.

References

[1] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, and S. K. S. Hari, "PyTorchFI: A Runtime Perturbation Tool for DNNs," Proceedings - 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN-W 2020, pp. 25–31, 2020.

[2] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G. Y. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in Proceedings - Design Automation Conference, vol. Part F1377, 2018.

[3] Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman, and N. DeBardeleben, "Tensorfi: A flexible fault injection framework for tensorflow applications," Proceedings - International Symposium on Software Reliability Engineering, ISSRE, vol. 2020-Octob, pp. 426–435, 2020.

[4] B. Goldstein, S. Srinivasan, N. K. Mellempudi, D. Das, L. Santiago, V. C. Ferreira, N. Solon, S. Kundu, and F. M. G. França, "Reliability evaluation of compressed deep learning models," in 2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS), 2020.

[5] Y. Zheng, Z. Feng, Z. Hu, and K. Pei, "Mindfi: A fault injection tool for reliability assessment of mindspore applicacions," in 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2021, pp. 235–238.
[6] E. Ozen and A. Orailoglu, "SNR: Squeezing Numerical Range Defuses Bit Error Vulnerability Surface in Deep Neural Networks," ACM Trans. Embed. Comput. Syst., vol. 20, no. 5s, 2021. [Online]. Available: https://doi.org/10.1145/3477007
[7] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," Proceedings of the Annual International Symposium on Microarchitecture, MICRO, vol. 2020-Octob, pp. 270–281, 2020.
[8] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective hardening for neural networks in fpgas," IEEE Transactions on Nuclear Science, vol. 66, no. 1, pp. 216–222, 2018.
[9] S. K. S. Hari, T. Tsai, M. Stephenson, S. W. Keckler, and J. Emer, "SASSIFI: An architecture-level fault injection tool for GPU application resilience evaluation," ISPASS 2017 - IEEE International Symposium on Performance Analysis of Systems and Software, vol. 1, no. 1, pp. 249–258, 2017.
[10] G. Gambardella, J. Kappauf, M. Blott, C. Doehring, M. Kumm, P. Zipf, and K. Vissers, "Efficient error-tolerant quantized neural network accelerators," in 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). IEEE, 2019, pp. 1–6.
[11] D. Xu, Z. Zhu, C. Liu, Y. Wang, H. Li, L. Zhang, and K.-T. Cheng, "Persistent fault analysis of neural networks on fpga-based acceleration system," in 2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2020, pp. 85–92.
[12] D. Xu, Z. Zhu, C. Liu, Y. Wang, S. Zhao, L. Zhang, H. Liang, H. Li, and K.-T. Cheng, "Reliability evaluation and analysis of fpga-based neural network acceleration system," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 3, pp. 472–484, 2021.
[13] G. Li, S. K. S. Hari, M. B. Sullivan, T. Tsai, K. Pattabiraman, J. S. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2017, Denver, CO, USA, November 12 - 17, 2017, B. Mohr and P. Raghavan, Eds. ACM, 2017, pp. 8:1–8:12. [Online]. Available: https://doi.org/10.1145/3126908.3126964
[14] X. Xue, H. Huang, C. Liu, T. Luo, L. Zhang, and Y. Wang, "Winograd convolution: A perspective from fault tolerance," in Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022, pp. 853–858.
[15] A. Mahmoud, S. K. S. Hari, C. W. Fletcher, S. V. Adve, C. Sakr, N. R. Shanbhag, P. Molchanov, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Hardnn: Feature map vulnerability evaluation in cnns," ArXiv, vol. abs/2002.09786, 2020.
[16] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, "Binfi: An efficient fault injector for safety-critical machine learning systems," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356177
[17] H. Huang, X. Xue, C. Liu, Y. Wang, T. Luo, L. Cheng, H. Li, and X. Li, "Statistical modeling of soft error influence on neural networks," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1–1, 2023.
[18] C. Liu, Z. Gao, S. Liu, X. Ning, H. Li, and X. Li, "Special session: Fault-tolerant deep learning: A hierarchical perspective," in 2022 IEEE 40th VLSI Test Symposium (VTS). IEEE, 2022, pp. 1–12.
[19] W. Li, X. Ning, G. Ge, X. Chen, Y. Wang, and H. Yang, "Ftt-nas: Discovering fault-tolerant neural architecture," in 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2020, pp. 211–216.
[20] X. Ning, G. Ge, W. Li, Z. Zhu, Y. Zheng, X. Chen, Z. Gao, Y. Wang, and H. Yang, "Ftt-nas: Discovering fault-tolerant convolutional neural architecture," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 26, no. 6, pp. 1–24, 2021.
[21] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in 2018 IEEE 36th VLSI Test Symposium (VTS). IEEE, 2018, pp. 1–6.
[22] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). IEEE, 2016, pp. 267–278.
[23] S. K. S. Hari, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Making convolutions resilient via algorithm-based error detection techniques," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 2546–2558, 2022.
[24] B. Ghavami, M. Sadati, Z. Fang, and L. Shannon, "Fitact: Error resilient deep neural networks via fine-grained post-trainable activation functions," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022, pp. 1239–1244.
[25] D. Xu, M. He, C. Liu, Y. Wang, L. Cheng, H. Li, X. Li, and K.-T. Cheng, "R2f: A remote retraining framework for aiot processors with computing errors," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 11, pp. 1955–1966, 2021.

基于最大延迟调度的 Android 应用并发相关不稳定测试检测

张未晞 华铖钰 王微微* 赵瑞莲

（北京化工大学信息科学与技术学院 北京 100029）

摘　要：Android 系统的事件驱动编程模式和多线程并发机制是造成 Android 应用不稳定测试的主要原因。现有研究大多通过随机探索异步事件的执行顺序来改变测试执行的事件顺序，以检测不稳定测试（Flaky Test），但对异步事件可调度位置的探索并不准确，检测效果不高。因此，本文提出一种基于最大延迟调度的 Android 应用并发相关不稳定测试方法，并开发相应的测试检测工具 FlakyDroider，对测试执行中的异步事件进行最大延迟调度位置识别，并采用自适应的调度策略对所有异步事件进行最大延迟调度，以期通过尽可能少的测试执行次数，使测试结果发生变化，提高不稳定测试的检测效率。本文还对 Android 应用并发相关不稳定测试数据集进行了扩充，增加了 18 个应用程序及相应测试用例，并在扩充测试数据集上，利用开发的 FlakyDroider，进行不稳定测试检测，取得了较好的检测效果。

关键词：Android 应用；并发；不稳定测试检测；异步事件最大延迟调度位置；自适应调度

Android application concurrency-related flaky test detection based on maximum delay scheduling

ZHANG Weixi　HUA Chengyu　WANG Weiwei　ZHAO Ruilian

(Institute of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: The main reason for existing flaky tests in Android applications is the event-driven programming model and the multi-threading concurrency mechanism of the Android system. Existing research mostly focuses on altering the execution order of asynchronous events through random exploration to change the event sequence during testing, aiming to detect flaky tests. However, the exploration of the schedulable positions of asynchronous events is inaccurate, leading to ineffective detection. Therefore, this paper proposes a method for detecting flaky tests related to Android application concurrency based on maximum delay scheduling. A corresponding testing detection tool called FlakyDroider is developed. It identifies the positions for maximum delay scheduling of asynchronous events during test execution and applies an adaptive scheduling strategy to schedule all asynchronous events with maximum delay. The goal is to achieve changes in test results through as few test executions as possible, thereby improving the efficiency of flaky test detection. Additionally, this paper expands the dataset for flaky tests related to Android application concurrency by adding 18 applications and their corresponding test cases. Using the developed FlakyDroider on the expanded test dataset, favorable detection results have been achieved.

Key words: Android application; Concurrency; Flaky test detection; Maximum delay scheduling position for asynchronous events; Adaptive scheduling

1 引言

软件测试是验证软件功能，提高软件质量的主要手段。然而，近几年不稳定测试（Flaky Test）的出现，影响了软件测试的可信性。不稳定测试是指在没有改变代码和配置的情况下，重复执行一个测试用例，可

*通信作者：王微微 wangww@mail.buct.edu.cn

基金项目：国家自然科学基金 61872026

Foundation Item：National Natural Science Foundation of China 61872026

能会得到不同的测试结果（执行通过或失败）[1-6]。这意味着，测试的成功与否不能反映软件是否有错误，也就是说，人们无法根据测试结果来判断软件中是否存在错误，而且还可能因调试虚假失败而造成大量的时间资源浪费[7]，削弱了人们对软件测试的信赖。此外，不稳定测试对任何与测试结果相关的研究如故障定位[8]、变异测试[9]、测试套件自动生成[10]等都有极为严重的负面影响。因此，不稳定测试目前已成为学术界和工业界广泛关注和高度重视的研究课题。

Android 系统的事件驱动编程模式以及多线程并发机制是造成 Android 应用测试不稳定的主要原因[2, 11, 12]。Android 系统只有一个主线程，也称为 UI 线程，是负责处理与用户界面相关的所有事件和操作的线程。如果 UI 线程被阻塞，应用程序将无法响应用户的操作。为了保证应用程序的性能和 UI 线程的流畅性，Android 系统将耗时的操作任务放在后台线程中执行。当任务完成后，后台线程通过 Handler、Looper 和 AsyncTask 等机制，向 UI 线程返回一个事件来更新 GUI，这种后台线程发送的事件称为异步事件。然而 Android 系统的多线程并发机制，可能会使后台线程处理任务的时间不确定，导致异步事件返回顺序不确定，从而产生不确定的执行结果[15]。据 Luo 等人[1]对不稳定测试的实证研究发现，他们收集的 161 个提交中有 32 个属于并发相关的不稳定测试，占比达 20%。Eck 等人[13]对 200 个不稳定测试的分类研究表明，26%的不稳定测试是由于并发导致的。Alan 等人[14]收集了流程网站和 Android 平台的 235 个不稳定的 UI 测试，其中由异步等待造成的不稳定测试占比达 45%，例如测试试图与尚未完全加载的元素进行交互，这种不正确的事件顺序导致了操作无效并引发了异常。由此可见，不稳定测试是 Android 测试中经常遇到的问题，其中很大一部分都是与并发相关的。而在并发相关的不稳定测试中，则可能因异步事件执行顺序的不同，导致测试结果发生改变，从而产生不稳定测试。

由上可知，异步事件执行顺序的不同可能导致不一样的测试结果，因此，不稳定测试检测应探索异步事件的执行顺序[15]，这也是 Android 并发相关的不稳定测试检测的主要出发点。目前，Android 并发相关的不稳定测试检测方法主要包括三种：（1）重复执行。王潇[16]、Zhen 等[15]通过多次执行测试用例，判断在一定次数内（比如 50 次），测试结果是否发生了改变。如果测试结果既有成功又有失败，则该测试为不稳定测试。这类方法检测不稳定测试简单明了，只需要重复执行测试，但检测效果及效率不佳。（2）添加环境噪音。Silva[17]、Adamsen[18]、Shi[19]、Zhang[20]等人通过在环境中添加噪音，比如增加 CPU 负载或减少内存资源等，影响测试过程中异步事件的执行顺序，进而检测测试结果是否发生变化。该方法随机性大，无法有效检测并发相关的不稳定测试。（3）事件执行序探索。Zhen 等人[15]通过探索测试执行过程中异步事件的调度空间，以确定异步事件的调度位置并在调度空间内多次探索事件的执行顺序，进而观察测试结果是否改变。这种方法要求识别各个异步事件的调度空间，但异步事件调度空间的准确识别是一件困难的事，因此，该方法检测不稳定测试的准确性及效率较低。

由上述分析可知，现有研究大多通过随机探索异步事件的执行顺序，观察测试结果的改变，从而检测 Android 应用的并发相关不稳定测试。然而，Zhen 等人[21]对 Android 并发相关不稳定测试的实证研究表明，导致 Android 应用测试不稳定的根因主要包括（1）不确定的执行环境导致后台服务或资源没有得到及时响应，（2）事件执行顺序的改变引起执行操作的改变，导致不同的应用程序行为，比如软键盘消失较晚导致测试失败以及（3）线程间的数据竞争，如数据延迟更新导致测试检测“旧”数据。也就是说，后台线程执行任务过长，没有及时向主线程返回执行结果是不稳定测试产生的根源。显然，异步事件延迟执行的时间越长，其测试结果越有可能发生改变，越可能导致测试不稳定。

因此，本文提出一种基于最大延迟调度的 Android 应用并发相关不稳定测试检测方法。通过识别测试用例执行过程中异步事件的最大延迟调度位置，并采用相应的自适应调度策略，实现对异步事件的最大延迟调度，以尽可能少的测试执行，导致测试结果的不一致，达到检测不稳定测试的目的。本文的贡献如下：

- 提出了一种基于最大延迟调度的 Android 应用并发相关不稳定测试检测方法，通过识别异步事件的最大延迟调度位置，进行基于最大延迟的异步事件自适应调度，以产生更有可能导致测试不稳定的事件执行顺序，提高了并发相关的 Android 不稳定测试检测效果。
- 在现有 FlakyAppRepo 数据集中增加了 18 个 Android 应用程序，扩充了 Android 应用并发相关不稳定测试的数据集。
- 本文在 6 个不同开源应用程序的 13 个已知不稳定测试中对本文方法进行了实验验证，在 13 个不稳定

测试中检测到 12 个，与目前最先进的检测工具 FlakeScanner 相比，检测率提高了 15.4%。

2 相关背景

2.1 Android 线程间消息传递机制

Android 为了确保 UI 操作的线程安全，规定所有的 UI 操作都必须在 UI 线程中执行。每个 Android 应用程序都只有一个 UI 线程，也称为主线程，负责处理这些用户事件以及系统事件。为了系统的流畅性以及用户的体验，会把耗时的任务，比如网络、IO 等放在后台线程中执行，后台线程执行完成后，需要给 UI 线程发送消息来对 GUI 进行更新。UI 线程中有一个消息队列和一个循环器（Looper），Looper 负责不断地从消息队列（MessageQueue）中获取消息，并发送给各个线程中的 Handler 处理。在图 1 中，Looper 从消息队列中取出消息后，通过 dispatchMessage 方法发送给后台线程中的处理器（Handler）进行处理。Handler 处理完消息后，使用 sendMessage 方法发送一个消息 m_4（本文称之为异步事件）给 UI 线程，并使用 enqueueMessage 方法使消息进入消息队列中。由于后台线程处理任务的时间不确定，因此 m_4 进入消息队列的顺序的也不确定，从而导致不确定的执行顺序，造成测试的不稳定。本文把线程之间传递的消息对象定义为事件。

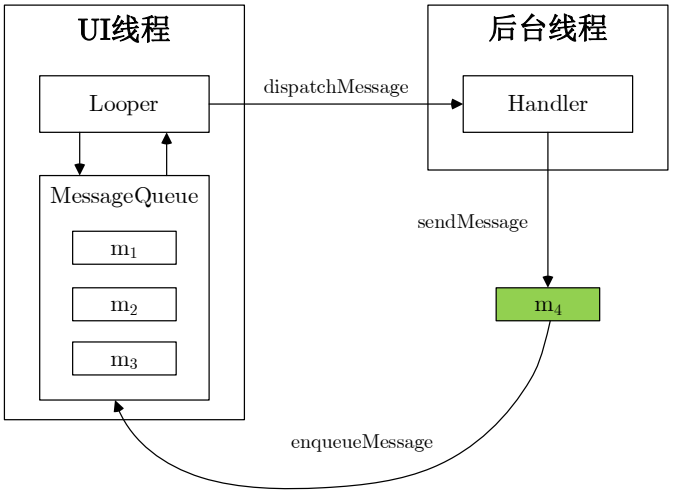


图 1 Android 线程间消息传递

2.2 Android 应用测试用例构成及其动态执行过程

测试用例构成：在 Android 应用的开发测试过程中，测试用例通过模拟应用程序的不同行为和测试场景，检查结果是否符合预期，以验证 Android 应用功能的正确性。测试用例由一系列测试语句构成，可表述为 tc= $< s_1,s_2,\cdots,s_n >$，其中 tc 为测试用例，s 为测试语句。测试语句通常包含对 Android 组件的操作、调用应用程序的方法、断言等。

测试用例的动态执行过程：为了使测试用例能够自动化执行，需借助自动化测试框架与 Android 应用实现自动交互。测试框架的测试线程与 Android 应用的 UI 线程及后台线程之间共同协作，实现自动化测试过程。测试线程负责执行测试用例代码。假设在测试用例 tc = $< s_1,s_2,\cdots, s_n >$中，s_1 语句是对用户界面进行的操作，例如模拟用户输入、点击按钮等。测试线程执行 s_1 语句后，给 UI 线程发送消息，使相应的应用程序函数执行，触发事件$<e_i,\cdots,e_j>$。若在此过程中，涉及到了耗时操作，将在后台线程中执行响应函数，触发事件$<e_u,\cdots,e_v>$，后台线程处理结束后，向 UI 线程发送消息返回执行结果。最终，UI 线程处理完任务，发送消息到测试线程，测试线程接收测试语句 s_1 的执行结果，并驱动下一语句 s_2 的执行，重复上述过程，直至所有测试语句执行完成，测试用例执行结束。

2.3 FlakeScanner 检测原理

为检测 Android 应用的测试用例是否是不稳定测试，FlakeScanner 通过调度异步事件改变事件执行序列，观察测试结果是否发生改变。如图 2 所示，s 表示测试语句，该测试用例为 tc = $<s_1,s_2,\ldots,s_i,\ldots,s_n>$；e 为事件，序号表示测试执行过程中事件的执行顺序。对于给定的测试，FlakeScanner 执行测试用例以跟踪所有生成的事件。接下来，创建测试语句与其对应事件之间的映射。多次执行测试以探索异步事件的调度

空间。 最后，在调度空间内探索异步事件执行顺序。 如果在探索过程中检测到测试结果改变，则该测试被识别为不稳定测试。

具体来讲，FlakeScanner 首先生成测试语句与事件之间的映射，如图 2 中测试语句 s_1 与 e_1、e_2 映射，表示在执行 s_1 时生成了 e_1、e_2 两个事件，其中 e_2 为异步事件。接着工具探索测试执行时异步事件的调度空间，即异步事件可以在空间内任意位置执行，调度空间由一系列测试语句构成，假设异步事件 e_2 的可执行空间为该事件生成到 e_i 事件后，则 e_2 的调度空间为$<s_1,s_2,\ldots,s_i>$。FlakeScanner 在调度空间内每一个测试语句前调度异步事件，从而生成不同的测试执行序列，对于 e_2 来说，则是分别调度 e_2 在 s_i、s_{i-1}、…、s_2 前执行，直到发现测试结果改变或所有异步事件都在调度空间内探索结束，即异步事件在其调度空间内每个测试语句前都完成调度。

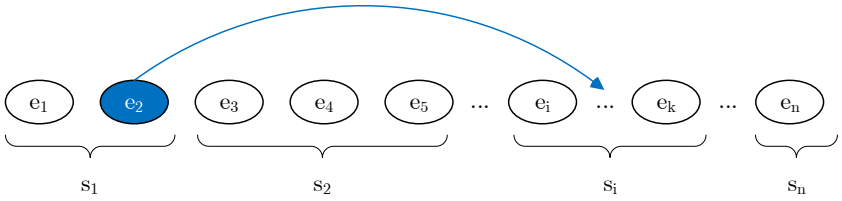


图 2 FlakeScanner 检测原理

3 研究动机

本节以一个 Android 应用及其相应的测试用例为例，说明本文的研究动机。该 Android 应用的一个常用模块是下载并保存图片功能，即根据用户提供的图片网址 URL，通过点击“下载图片”按钮，从网络中下载并显示图片，用户也可点击图片另存至本地文件中。测试用例 tc 用来检验其下载与保存图片的功能是否正确。该测试用例由四条测试语句组成，即 tc = $<s_1,s_2,s_3,s_4>$，语句 s_1 提供两个图片网址 URL 并点击下载按钮，应用程序下载图片并在 ImageView1 控件和 ImageView2 控件中显示；语句 s_2 点击第二张图片（ImageView2），保存图片到本地；语句 s_3 和 s_4 分别检查 ImageView1 和 ImageView2 的图片是否为空，若至少一个为空则测试失败。

该测试用例的一次执行过程如图 3 所示。测试语句在测试线程中执行，驱动 Android 应用 UI 线程及后台线程的执行。具体而言，测试线程首先执行测试语句 s_1，点击下载按钮 DownloadButton（第 1 步）；由于 s_1 语句中提供了两个图片 URL 地址，Android 应用 UI 线程执行按钮响应函数 onClick()（第 2 步），会分别创建了两个后台线程子线程 1 和子线程 2，用以执行图片下载任务，函数 DownloadImage1 和 DowmloadImage2 通过 URL 获得位图，将其传入到 message，并通过 sendMessage 方法将位图传给 UI 线程（第 3 和 4 步），两个后台线程发送的 message1 和 message2 分别为异步事件 e_1 和 e_2；UI 线程接收到两个异步事件 e_1 和 e_2 并处理，将下载的图片分别存放在 ImageView1 和 ImageView2 控件（第 5 和 6 步），至此，测试语句 s_1 执行完毕；测试线程执行语句 s_2 点击 ImageView2 控件（第 7 步），将该控件的图片保存到本地中（第 8 步）；然后测试线程对 ImageView1 以及 ImageView2 控件进行断言，检查两个控件中是否传入了图片（第 9 步）。

该测试用例执行过程中，第 7 步将用到第 4 步的执行结果，即异步事件 e_2 的执行结果，若 UI 线程没有得到 e_2 的结果，程序会停滞，测试线程一直等待下去。但异步事件 e_1 则不同，后续程序执行步骤并不依赖于 e_1 的结果，也就是说，即使 e_1 没有返回到 UI 线程，测试也可以继续执行。但若异步事件 e_1 在第 9 步断言前未返回给 UI 线程结果，即第一张图片不能及时传入到 ImageView1 中，则会导致断言失败，触发测试不稳定。

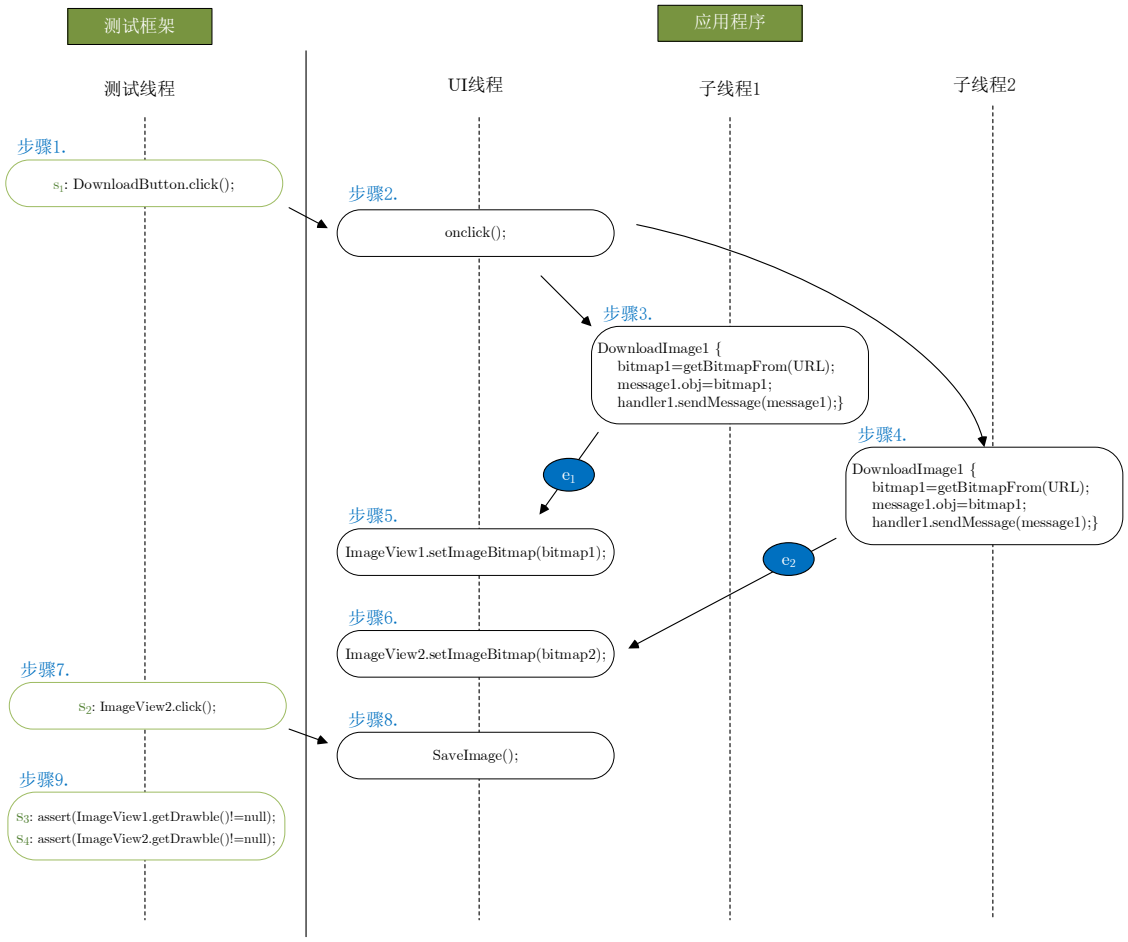


图 3 测试用例的一次执行流程

FlakeScanner[15]对该测试用例进行不稳定检测时，首先挂起子线程 1 以及子线程 2，延迟 e_1，e_2 的返回，当程序执行到第 7 步，点击 ImageView2 控件时，由于程序停滞等待 e_2 的返回结果，FlakeScanner 按照 FIFO 策略开始释放挂起的线程，按照挂起顺序先释放子线程 1 返回 e_1，释放后依然观测到程序停滞，接着释放子线程 2 返回 e_2。而由于 e_1 的提前返回，即使 FlakeScanner 多次执行测试用例探索异步事件的调度空间，第 9 步断言一直可以通过，检测到该测试用例是稳定的（NotFlaky）。

事实上，在上述测试执行过程中，子线程 1 依然可以被挂起一段时间直至测试执行结束，也就是说，若在第 7 步时直接释放子线程 2 返回 e_2，子线程 1 依然保持挂起状态，e_1 事件仍然被延迟执行，直到第 9 步检查断言，则可观察到测试用例失败，检测出该测试用例是不稳定的。分析可知，若子线程 1 返回的异步事件 e_1 达到其最大延迟位置，则可以触发不稳定测试。因此，本文基于上述考虑，提出一种基于最大延迟调度的 Android 应用并发相关不稳定测试检测方法，具体如下。

4 基于最大延迟调度的不稳定测试检测方法

对于一个给定的 Android 应用及其测试用例，若能在测试用例执行过程中，动态改变事件的执行顺序，则可能改变测试的执行结果，进而判断出测试是否是不稳定的。上述分析可知，测试执行过程中，对异步事件进行最大延迟调度，有助于不稳定测试的检测。因此，本文提出一种基于最大延迟调度的 Android 应用并发相关不稳定测试检测方法。

4.1 不稳定测试检测总体框架

本文不稳定测试检测方法主要包括异步事件最大延迟调度位置识别及其自适应调度两部分，总体框架如图 4 所示。针对给定的 Android 应用及其测试用例，首先执行测试用例，获取其事件执行轨迹；并在测试执行过程中，通过动态观测程序状态（如是执行状态还是停滞状态），识别测试执行中异步事件的最大延迟调度位置；在此基础上，自适应的地对异步事件进行调度，以尽可能少的测试执行，实现所有异步事件

的最大延迟调度，并依据测试结果的异同，检测测试是否为不稳定测试。

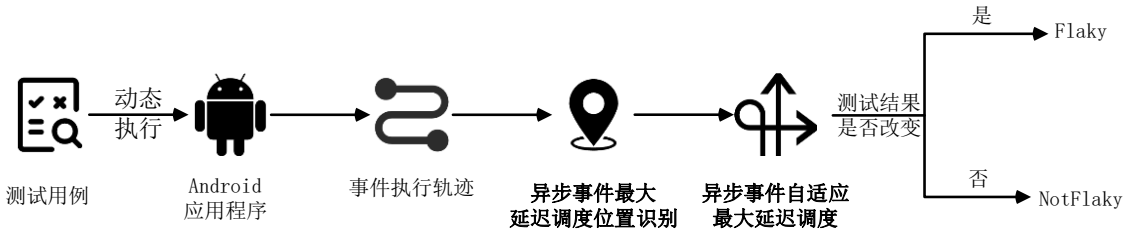


图 4 总体框架

4.2 异步事件最大延迟调度位置识别

异步事件的最大延迟调度位置识别涉及异步事件识别和基于挂起释放的异步事件最大延迟调度位置识别，具体如下。

4.2.1 异步事件识别

本文首先给出事件以及异步事件的定义：

事件（e）：本文中所指事件 e 为主线程（UI 线程）、后台线程以及测试线程生成的 Message 对象。

异步事件（e_{asyn}）：后台线程生成的 Message 对象。

由于测试执行环境的不确定性，每次测试运行时事件执行顺序都有可能不同，为了识别不同测试轨迹中相同的事件，本文使用事件标识（eID）对事件进行表示，以保证每次测试执行时，事件都有一个唯一的标识。

事件标识（eID）：本文对测试执行过程中生成的事件 e 都进行了唯一的标识，即事件 ID。事件 ID 由一个三元组表示<messageID，sID，thread>。其中，messageID 是当前事件，即 message 对象的标识，由生成当前事件时的线程堆栈中一系列方法签名来表示；sID 是测试语句的标识，由生成该异步事件的测试语句执行时测试线程堆栈中一系列方法的签名来表示；thread 是生成当前事件所在的线程名，通过 thread 来判断当前事件是否是异步的。

为识别事件轨迹中的异步事件，本文将测试执行过程中的线程分为测试线程、UI 线程以及后台线程，并基于事件所在的线程识别该事件是否为异步事件，即当在测试执行过程中检测到事件时，获取当前正在执行的线程名，若线程名包含“main”，表明该线程为主线程，即 UI 线程。若线程名与当前应用程序中 testRunnerThread 的值相同，表明该线程为测试线程。若事件的线程既不是 UI 线程也不是测试线程，认定该事件为异步事件。

4.2.2 基于挂起释放的最大延迟调度位置识别方法

（1）最大延迟调度位置分析

后台线程处理任务的时间与当前执行环境有关。由于执行环境的不确定性，导致后台线程处理任务的时间不确定，其返回异步事件 e_{asyn} 的时间也就不确定。但后续测试执行有可能依赖于 e_{asyn} 的结果，换言之，后续测试执行中生成的某一事件 e_x 有可能需要等待 e_{asyn} 的执行结果才能执行。如果 e_{asyn} 没有及时返回执行结果，程序会停滞，直到获得所需执行结果。因此，异步事件 e_{asyn} 执行结果返回越晚，测试的不稳定性则越大。则把程序停滞等待异步事件返回执行结果的位置称为该异步事件的最大延迟调度位置。

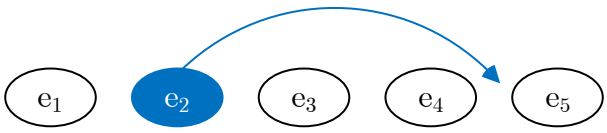


图 5 异步事件调度位置

例如，图 5 所示是一个测试用例的事件执行轨迹，trace = <e_1,e_2,e_3,e_4,e_5>，其中 e_2 为异步事件。假设 e_5 的执行依赖于 e_2 的执行结果，那么挂起 e_2 所在线程后，程序执行完 e_4 事件后会停滞，等待 e_2 事件的执行结果。则事件 e_4 与 e_5 之间，是异步事件 e_2 的最大延迟调度位置。

对于某些异步事件来说，后续程序的执行不依赖于此类异步事件的执行结果。对此类异步事件而言，程序能够一直等待其执行结果直至测试结束，不会导致程序停滞。则此类异步事件的最大延迟调度位置是

测试用例结束执行的位置。

（2）挂起释放策略

为了识别异步事件的最大延迟位置，本文通过对异步事件所在线程进行挂起和释放，以动态地观测程序状态（是执行状态还是停滞状态），从而判定异步事件是否处于最大延迟位置。具体的挂起和释放策略如下：

顺序挂起策略：挂起异步事件所在的线程旨在延后异步事件的执行，改变事件的执行顺序。为了保证所有异步事件在动态执行过程中均可延后执行，以调整其执行顺序，本文采用顺序挂起策略，即一旦检测到测试执行触发的异步事件，便将其所在线程挂起，也就是说，异步事件线程的挂起顺序取决于其在测试执行中的出现顺序。

当程序停滞时，即当前测试执行需要异步事件的执行结果才能继续，说明有一个或多个异步事件处于挂起状态，相应的释放策略有：

顺序释放策略：顺序释放策略即为 FIFO 释放策略，即按照线程挂起的顺序，依次释放线程，直到程序可以再次执行为止，并记录下该停滞位置以及当前线程释放后能够使测试再次执行的异步事件，并将该异步事件与停滞位置关联。

直接关联释放策略：直接关联释放策略是指已知当前程序停滞是由哪一个或哪几个异步事件造成的条件下，直接释放造成程序停滞的异步事件线程。

（3）最大延迟调度位置识别

本文对最大延迟调度位置的识别基于程序的动态执行状态，其基本思想是：当有一个或多个异步事件线程挂起时，若程序停滞，在释放一个异步事件 e_{asyn} 后程序开始继续执行，则认为当前的停滞是由于 e_{asyn} 的执行结果没有及时返回引起的，则认为当前测试执行停滞的位置即为 e_{asyn} 的最大延迟调度位置。

具体而言，当程序停滞时，当前线程堆栈中的一系列方法签名可作为停滞位置的标识。在释放后台线程执行一个异步事件后，若测试执行能够继续进行，则表明该异步事件是在最大延迟调度位置处释放，对该异步事件的最大延迟调度位置识别成功，并记录其异步事件 ID（eID）以及停滞位置标识。若在释放后台线程执行一个异步事件后，程序依然为停滞状态，则表明释放的异步事件并没有达到最大延迟调度位置，那么在下一次测试执行时，依据不同的释放策略继续识别未达到最大延迟调度位置的异步事件。

4.3 异步事件自适应最大延迟调度策略

Android 应用程序以某种顺序执行异步事件，可能会导致测试结果的改变。通过改变异步事件的执行位置，可以改变测试执行中事件的执行顺序。前面分析已表明，依据异步事件的最大延迟执行位置进行调度，有助于提高不稳定测试检测的效率。因此，本文在一次测试执行时，尽可能识别多个异步事件的最大延迟调度位置。但当测试执行程序停滞时，对应有多个异步事件挂起的情况下，无法判断是由某一个或者某几个异步事件导致的，尤其是在测试第一次执行时，此时采用顺序释放策略，即按照线程挂起的顺序，依次释放线程，直到测试可以再次执行为止，并记录下该停滞位置以及当前线程释放后能够使测试再次执行的异步事件，并将该异步事件与停滞位置关联。若已知道当前测试执行停滞是由哪一个或哪几个异步事件造成的，则采用直接关联释放策略，直接释放造成程序停滞的异步事件的线程。

本文提出的自适应最大延迟调度策略基本思想是：构建一个达到最大延迟调度位置的异步事件及程序停滞位置的列表 isFindList，用来指导异步事件调度；构建一个未达到最大延迟位置的异步事件列表 isNotFindList，用来判断测试执行是否需要进行下一次。每进行一次测试执行，修订上述两个列表。每次测试执行结束后，判断 isNotFindList 列表是否为空，为空则结束不稳定测试的检测。具体而言，每次测试执行时，将没有识别出最大延迟调度位置的异步事件放入 isNotFindList 列表中，将识别出最大延迟调度位置的异步事件以及程序停滞位置放入 isFindList 列表中，并查找 isNotFindList 列表中是否存在该异步事件，若有则表明当前测试执行中识别出了之前测试执行时未能识别出最大延迟调度位置的异步事件，在 isNotFindList 列表中删除该异步事件。每次测试执行时，在测试执行程序停滞处根据 isFindList（如果列表中存在该位置）直接调度在该位置达到最大延迟的异步事件执行，没有则按照 FIFO 释放策略识别在该位置达到最大延迟调度的异步事件，并记录在 isFindList 列表中。每次测试执行结束后，判断 isNotFindList 列表是否为空，不为空表明仍有异步事件需要在最大延迟调度位置处执行，需要启动下一次测试执行，否则

结束不稳定测试的检测。通过两个列表的信息，每次测试执行时可以快速有效地找到异步事件的最大延迟调度位置，进行异步事件的最大延迟调度，以最少的测试执行次数，实现所有异步事件的最大延迟调度。

4.4 实现算法

测试用例执行中，首先识别异步事件，然后通过线程的挂起和释放对每一个异步事件进行最大延迟调度位置的识别，并在测试用例下次执行时，依据最大延迟调度位置进行异步事件调度。为减少检测不稳定测试的成本，可在一次测试执行，对多个异步事件进行调度。对于上一次测试执行没有识别出最大延迟调度位置的异步事件，将在下一次测试执行时进行处识别，具体算法如算法 1、2、3 所示。

算法1 不稳定测试检测算法

```
1   Procedure DetectFlaky（App A，Test T，Android ART)
2     isFindList ← Ø;    //达到最大延迟调度位置的异步事件及位置
3     isNotFindList ← Ø;  //未达到最大延迟调度位置的异步事件
4     eventTrace = RunTest ( A , ART ,T ); //eventTrace为测试执行得到的事件执行
    序列
5     identifyLoc ( isFindList , isNotFindList ,eventTrace);
6     testResult = AdaptiveDispatchAsynEvent ( isFindList , isNotFindList );
7     if isChange(testResult) then //若测试结果改变
8       return flaky;
9     else
10      return notFlaky;
11    end
```

算法 1 是本文不稳定测试检测方法的具体实现，其输入为 Android 的应用程序及其测试用例，输出为检测结果——该测试是否稳定。算法 1 主要包括两个方法，方法 identifyLoc 用于异步事件最大延迟调度位置的识别，方法 AdaptiveDispatchAsynEvent 用于异步事件自适应最大延迟调度。算法 1 首先初始化，置 isFindList 列表和 isNotFindList 列表为空（第 2、3 行）；执行测试用例（第 4 行），得到测试执行的事件轨迹，并记录该次测试执行的结果；调用 identifyLoc 方法（第 5 行），得到这次测试执行中识别出的异步事件最大延迟位置；调用 AdaptiveDispatchAsynEvent 方法（第 6 行），对异步事件进行自适应的最大延迟调度，当所有异步事件都进行最大延迟调度后，返回测试执行结果，若测试执行结果发生变化，则判定该测试为不稳定的，否则为稳定测试。下面具体介绍两个方法的实现过程，分别如算法 2 和 3 所示。

算法2 异步事件最大延迟调度位置识别算法

```
1   Func identifyLoc(List isFindList,List isNotFindList,List eventTrace)
2     List ← Ø;  //被挂起线程的事件
3     RunTest ( A , ART ,T );
4     for e in eventTrace then
5       if isAsync(e) then   //如果当前事件为异步事件
6         suspend(e.thread);
7         List ← List ∪ e.eID;
8         if isWating(THtest) && isEmpty(eventQueue) then   //程序停滞
9           while True do
10            unsuspend(e_earlier.thread);
11            remove e_earlier.eID from List;
12            if isWating（THtest） && isEmpty(eventQueue) then
13              isNotFindList ← isNotFindList ∪ e.eID;
14              break;
15            else
```

算法2 异步事件最大延迟调度位置识别算法

```
16                  isFindList ← isFindList ∪ <e.eID, getCurrentLocation()>;
17                  break;
18                end
19              end
20            end
21          end
22        end
23        return isFindList, isNotFindList;
```

如算法 3 所示，对于 AdaptiveDispatchAsynEvent 方法，输入为上一次测试执行结束后的 isFindList 列表以及 isNotFindList 列表，输出为测试执行结果。算法 3 在测试执行过程中对检测到的异步事件的线程都挂起（第 5、6 行）。如果测试执行过程中检查到程序停滞（第 8 行），判断该位置是否在 isFindList 列表中，如果在则根据直接关联释放策略直接释放在该位置达到最大延迟调度的异步事件线程（第 11 行）。如果不在则按照识别异步事件最大延迟调度位置的方法继续寻找在该位置达到最大延迟调度的异步事件（第 13-24 行）。如果对一个异步事件确定了最大延迟调度位置后，将其从 isNotFindList 列表中移除（第 18 行）。通过执行几次测试用例后，对每一个异步事件都达到了最大延迟调度，并返回测试执行结果（第 29 行）。

算法3 异步事件自适应最大延迟调度算法

```
1     Func AdaptiveDispatchAsynEvent(List isFindList , List isNotFindList)
2       while isNotFindList != null do
3         RunTest ( A , ART ,T );   //重新执行测试用例
4         List ← Ø;   //挂起的事件
5         if isAsync(e) then
6           suspend(e.thread);
7           List ← List ∪ e.eID;
8           if isWating(THtest) && isEmpty(eventQueue) then
9             currentLocation = getCurrentLocation();
10            if currentLocation in isFindList then
11              unsuspend(e.thread);
12              else
13                while True do
14                  unsuspend(e_earlier.thread);
15                  remove e_earlier.eID from List;
16                  if isWating (THtest) && isEmpty(eventQueue) then
17                    isFindList ← isFindList ∪ <e.eID, getCurrentLocation()>;
18                    remove e.eID from isNotFindList;
19                    break;
20                  else
21                    isNotFindList ← isNotFindList ∪ e.eID;
22                    break;
23                  end
24                end
25              end
26            end
```

算法3 异步事件自适应最大延迟调度算法	
27	**end**
28	**end**
29	**return** testResult;

5 评估

本文在 Android 项目中评估 FlakyDroider 在检测不稳定测试方面的有效性。FlakyDroider 可以在 Android 的模拟器和物理设备上运行，并支持多个测试框架，如 Espresso[22]，Robotium[23]，UiAutomator[24] 等测试框架。

5.1 数据集

本文工具的输入是 Android 应用程序的 APK 文件，目前对于 Android 并发相关的不稳定测试数据集只有 FlakeScanner[15]作者收集的 FlakyReppo，共 33 个 GitHub 上 Android 项目。我们扩展了该数据集，在 GitHub 上 以“FlakyTest”作为关键字搜索提交，并人工筛选出合适的 Android 项目。本文在 FlakyReppo 的基础上新增了 18 个 Android 项目，如表一所示，其中版本号为 0 表明当前只有一个版本。

表 1 数据集

应用名	版本	#Stars	类别
armeria	1.23.1	4.3k	Framework
Quality-Tools-for-Android	0	1.3k	Tools
PdfBox-Android	2.0.22	864	Tools
kontalk	4.4.0	565	Communication
centraldogma	0.61.1	531	Service Configuration Repository
AndProx	2.0.5	209	Tools
SweetBlue	0	22	Productivity
AndroidRTC	0	16	Demo
PowerAct	1.7.1	16	Libraries
FreeHamburger	1.5.1	10	News
android-monorepo	0	10	Libraries
fdutil	1.6	5	Libraries
jetpack	0	5	Tools
SpeedoXtream	0	4	Tools
Breadcrumbs492	0	2	Tools
assignment2	0	2	Tools
Screen_Mirroring	0	1	Tools
Cellar	0	0	Download

5.2 研究问题

本文的评估旨在解决一下研究问题。

RQ1：不同释放策略对释放位置的影响，本文采取的释放策略能否使异步事件探索更延迟执行的位置？

RQ2：FlakyDroider 在检测不稳定测试效果方面与现有技术相比如何？

5.2.1 RQ1：异步事件执行位置对比

本文首先评估FlakyDroider采用FIFO释放策略加直接关联释放策略能否比FlakeScanner只采用FIFO释放策略使异步事件能够在更加延后的位置处执行。

在已知是不稳定的 Android 测试上进行检测，在同一个测试用例中，对比本文的工具 FlakyDroider 与 FlakeScanner 工具释放异步事件执行的位置。本文对三个不同应用程序的测试用例进行对比，根据测试执行轨迹，判断异步事件分别是在哪个同步事件（除异步事件以外的事件）前执行。用表 2，3，4 来说明，

其中 e 表示异步事件，p 表示同步事件，事件序号表示当前测试执行时事件的执行顺序。

表 2 FlexBox 应用 testJustifyContent_center 测试异步事件释放位置对比

异步事件	FlakeScanner	FlakyDroider
e_1	p_2	p_2
e_2	p_{10}	p_{10}
e_3	p_{10}	p_{10}
e_4	p_{10}	p_{10}
e_5	p_{11}	p_{12}
e_6	p_{11}	p_{11}
e_7	p_{12}	p_{12}

表 3 FlexBox 应用 testScrollToPosition_scrollToNewItem_direction_row 测试异步事件释放位置对比

异步事件	FlakeScanner	FlakyDroider
e_1	p_2	p_2
e_2	p_{10}	p_{11}
e_3	p_{10}	p_{10}
e_4	p_{10}	p_{10}
e_5	p_{11}	p_{11}
e_6	p_{11}	p_{11}
e_7	p_{12}	p_{12}

表 4 Barista 应用 someFlakyTest 测试异步事件释放位置对比

异步事件	FlakeScanner	FlakyDroider
e_1	p_2	p_2
e_2	p_{18}	p_{18}
e_3	p_{19}	p_{19}
e_4	p_{19}	p_{19}
e_5	p_{19}	p_{20}
e_6	p_{20}	p_{20}
e_7	p_{20}	p_{20}

其中，如表 2 中标红部分所示。对于 e_5 异步事件，在 FlakeScanner 工具中，e_5 在同步事件 p_{11} 前释放执行，而本文的工具表示，e_5 异步事件依然可以延迟在 p_{11} 后执行，在 FlakyDroider 中，e_5 事件在同步事件 p_{12} 前释放执行。表 3 和表 4 同样如此。

由上述分析可知：FlakyDroider 相比于 FlakeScanner 能够使部分异步事件在测试执行时更加延后的位置处释放执行，即 FlakyDroider 采用的释放策略能够使测试执行过程中所有异步事件达到最大延迟调度位置处释放执行。

5.2.2 RQ2：FlakyDroider 检测效果分析

对于 RQ2 的问题，我们检测了数据集中已知为不稳定测试的测试用例，并与 FlakeScanner 进行对比，实验结果如表三所示。其中运行次数指在检测测试用例是否为不稳定时测试用例的运行次数。检测结果中为“√”的表示工具能够检测出该测试为不稳定测试，“×”表示无法检测出该测试为不稳定测试。其中，FlakeScanner 在检测 AntennaPod 应用程序的 testAddFeed 测试方法时报错，用“-”表示无法检测。

如表 5 所示，FlakyDroider 可以检测出 FlakeScanner 无法检测出的不稳定测试，如 ImageSave 应用的 ButtonClick_SearchPic 测试用例以及 AntennaPod 应用程序的 testAddFeed 测试用例。在检测效率上，除去 AntennaPod 应用程序的 testAddFeed 测试方法，FlakyDroider 在图 5 所示的所有测试方法中检测不稳定测试的总运行次数为 25 次，FlakeScanner 的总运行次数为 29 次，FlakyDroider 的检测效率较高。

表 5 已知为不稳定测试上 FlakyDroider 与 FlakeScanner 的实验结果对比

应用程序名	测试方法名	FlakyDroider		FlakeScanner	
		运行次数	检测结果	运行次数	检测结果
FlexBox	testJustifyContent_center	2	√	2	√
	FtestScroll...column	2	√	2	√
Omni-Notes	GeocodeHelperTest	2	√	2	√
Android-RTC	testCaptureFormatChange	2	×	3	×
	testVideoSourceRestart	2	√	2	√
ImageSave	ButtonClick_SearchPic	3	√	3	×
Barista	overflowMenuClick_byId	2	√	2	√
	openOverflowMenu...Option	2	√	2	√
	overflowMenuClick_byTitle	2	√	2	√
AntennaPod	testClickNavDrawer	2	√	3	√
	testAddFeed	2	√	-	-
	testReplay…PlaybackOff	2	√	3	√
	testReplay…PlaybackOn	2	√	3	√

由上述分析可知：FlakyDroider 在并发相关 Android 不稳定测试检测效果较好，能够检测出大部分的不稳定测试，相比于现在最先进的检测工具 FlakeScanner 来说，能够检测到其检测不出的不稳定测试，且检测效率比较高。

6 结论

不稳定测试降低了测试可靠性，造成严重的后果。在本文中，提出了一种基于最大延迟调度的并发相关 Android 不稳定测试检测方法，并给出 FlakyDroider 工具，通过识别异步事件最大延迟调度位置，并调度异步事件对事件执行序进行探索，以触发不稳定测试。此外本文扩充了并发相关 Android 不稳定测试检测数据集——FlakyAppRepo，增加了 18 个应用程序。并且本文在数据集上进行了不稳定测试检测，得到了较好的结果。在未来的研究工作中，将继续对数据集进行扩充，并分析未能检测到不稳定测试的原因，以改进 FlakyDroider 增强检测并发相关的不稳定测试的效果。

参 考 文 献

[1] Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, et al. An empirical analysis of flaky tests[C], The 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, USA , 2014: 643–653.

[2] Swapna Thorve, Chandani Sreshtha, Na Meng. An Empirical Study of Flaky Tests in Android Apps[C], International Conference on Software Maintenance and Evolution.ICSME,Madrid, Spain, 2018:Sep 23-29.

[3] Owain Parry, Gregory M. Kapfhammer, Michael Hilton, et al. A Survey of Flaky Tests[J]. ACM Transactions on Software Engineering and Methodology, 2021 ,31 (1): 1-74. https://doi.org/10.1145/3476105.

[4] Martin Gruber, Stephan Lukasczyk, Florian Kroiß, et al. An Empirical Study of Flaky Tests in Python[C], 14th IEEE Conference on Software Testing, Verification and Validation. ICST, London, UK, 2021:37-38.

[5] Negar Hashemi, Amjed Tahir, Shawn Rasheed. An Empirical Study of Flaky Tests in JavaScript[C], 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), Limassol, Cyprus, 2022:24-34.

[6] Wing Lam, Patrice Godefroid, Suman Nath, et al. Root causing flaky tests in a large-scale industrial setting[C], The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019), New York, USA, 2019: 101–111.

[7] Azeem Ahmad, Ola Leifler, Kristian Sandahl. Empirical analysis of practitioners' perceptions of test flakiness factors [J]. Software Testing Verification and Reliability,2021 ,31 (8): e1791. https://doi.org/10.1002/stvr.1791.

[8] Béla Vancsics,Tamás Gergely,rpád Beszédes. Simulating the Effect of Test Flakiness on Fault Localization Effectiveness [C], IEEE Workshop on Validation, Analysis and Evolution of Software Tests (VST), London, Canada, 2020:28-35.

[9] August Shi, Jonathan Bell, Darko Marinov. Mitigating the effects of flaky tests on mutation testing [C], The 28th ACM SIGSOFT International Symposium.ACM, Beijing, China, 2019:112-122.

[10] Samad Paydar, Aidin Azamnouri. An Experimental Study on Flakiness and Fragility of Randoop Regression Test Suites [J], Fundamentals of Software Engineering. FSEN,2019, 11761:111-126. https://doi.org/10.1007/978-3-030-31517-7_8.

[11] Mattia Fazzini, Alessandra Gorla, Alessandro Orso. A framework for automated test mocking of mobile apps [C], The 35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20), New York, USA, 2021:1204–1208.

[12] Zebao Gao. Making System User Interactive Tests Repeatable: When and What Should we Control? [C], IEEE International Conference on Software Testing, Verification and Validation (ICST), Chicago, USA, 2016: 379-380.

[13] Moritz Eck, Fabio Palomba, Marco Castelluccio, et al. Understanding flaky tests: the developer's perspective [J]. The 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019), 2019 , 830–840.doi: 10.1145/3338906.3338945.

[14] Alan Romano, Zihe Song, Sampath Grandhi, et al. An Empirical Analysis of UI-Based Flaky Tests [C], IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, ES, 2021:1585-1597.

[15] Zhen Dong, Abhishek Tiwari, Xiao Liang Yu, et al. Flaky test detection in Android via event order exploration [J]. The 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021), 2021 , 367–378.doi: 10.1145/3468264.3468584.

[16] 王潇, 杨秋辉, 刘芳, 等. Randoop 和 Evosuite 生成的测试用例的质量分析[J]. 计算机应用研究, 2020(S01):3.
Wang Xiao, Yang Qiuhui, Liu Fang, 等. Quality Analysis of Test Cases Generated by Randoop and Evosuite[J]. Application Research of Computers, 2020(S01):3.

[17] Denini Silva, Leopoldo Teixeira, Marcelo d'Amorim. Shake It! Detecting Flaky Tests Caused by Concurrency with Shaker [C], IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, Australia, 2020: 1063-6773.

[18] Christoffer Quist Adamsen, Gianluca Mezzetti, Anders Møller. Systematic execution of Android test suites in adverse conditions [C], The 2015 International Symposium on Software Testing and Analysis (ISSTA 2015), New York, USA, 2015:83–93.

[19] August Shi, Alex Gyori, Owolabi Legunsen, et al. Detecting Assumptions on Deterministic Implementations of Non-deterministic Specifications [C], IEEE International Conference on Software Testing, Verification and Validation (ICST), Chicago, USA, 2016: 80-90.

[20] Sai Zhang, Darioush Jalali, Jochen Wuttke, et al. Empirically revisiting the test independence assumption [C], The 2014 International Symposium on Software Testing and Analysis (ISSTA 2014), New York, USA, 2014:385–396.

[21] Zhen Dong, Abhishek Tiwari, Xiao Liang Yu, et al. Concurrency-related Flaky Test Detection in Android apps [J].doi: 10.48550/arXiv.2005.10762.

[22] Android developers. Espresso [OL]. https://developer.android.com/training/testing/espresso, 2023.

[23] renas. robotium [OL]. https://github.com/RobotiumTech/robotium, 2023.

[24] Android developers. androidx.test.uiautomator [OL]. https://developer.android.com/reference/kotlin/androidx/test/uiautomator, 2023.

低密度奇偶校验（LDPC）码中陷阱集相关研究进展

崔子浩[①] 刘奇浩[②*] 钟戟[③] 郭鹏[②]

[①](山东云海国创云计算装备产业创新中心有限公司 济南 250101)

[②](华芯半导体有限公司 济南 250101)

[③](山东浪潮人工智能研究院有限公司 济南 250101)

摘要:

低密度奇偶校验（LDPC）码由于其理论性能逼近香农限的优点成为了目前流行的向前纠错方案，吸引了众多研究者的广泛关注。特别地，其在基于置信传播基础上的迭代译码方案如最小和算法等的条件下具有良好的性能。然而，对于大部分迭代解码方案，LDPC 码在高信噪比的区域内都会出现错误平层（error floor）现象，即误码率的下降速率会明显减慢。这一弱点阻碍了 LDPC 码在纠错领域中的进一步应用。对于 AWGN 信道而言，LDPC 码的错误平层主要源于其对应 Tanner 图中会导致译码失败的特殊拓扑结构，这种特殊的结构被称为陷阱集（trapping set）。陷阱集的存在使得当陷阱集中包含的节点发生数据错误时，译码器很难检测出错误的发生，大大增加了译码的失败概率，即使在一定次数的迭代之后也无法得到正确的结果。这一特性限制了 LDPC 码在要求具有低目标错误率的应用（如存储设备）中的使用。因此，对陷阱集进行深入的了解和研究，对于解决错误平层现象有着很大的意义。本文综述了近年来与 LDPC 码中与陷阱集有关的研究，对相关的陷阱集的基本定义、检索方法、消除手段等进行了系统的介绍，最后对目前改善陷阱集和错误平层研究中尚未解决的问题、面临的挑战和未来的发展前景进行了进一步展望。

关键词: 低密度奇偶校验码 错误平层 陷阱集 算法

Research Progress on Trapping Sets in Low-Density Parity-Check (LDPC) Codes

Zihao Cui[①], Qihao Liu[②*], Ji Zhong[③], and Peng Guo[②]

[①](Shandong Yunhai Guochuang Innovative Technology Co., Ltd, Jinan 250101, China)

[②](Sinochip Semiconductors Co., Ltd, Jinan 250101, China)

[③](Shandong Inspur Artificial Intelligence Research Institute Co., Ltd Jinan 250101, China)

Abstract: Low-density parit-check (LDPC) codes has become a popular forward error correction scheme owing to their theoretical performance approaches the Shannon Limit, which has attracted extensive attention of many researchers. In particular, they have good performance under the conditions of iterative decoding schemes based on belief propagation, such as Min Sum algorithm. However, LDPC codes will have an error floor in the area of high signal-to-noise ratio for most iterative decoding schemes, where the decline rate of bit error rate will be significantly slowed down. This weakness hinders the further application of LDPC code in the field of error correction. For AWGN (Additive White Gaussian Noise) channels, the error floor of LDPC codes is mainly caused by some special topologies that lead to decoding failure in the corresponding Tanner graph, which is called trapping sets. The existence of the trapping sets makes it difficult for the decoder to detect the occurrence of data errors when the nodes in the

trapping sets have data errors, which greatly enhances the probability of decoding failure, thus the correct result cannot be obtained even after a certain number of iterations. This feature limits the use of LDPC codes in applications that require low target error rates, such as storage devices. Therefore, the deep understanding and research of trapping sets is of great significance to solve the problem of error floor. In this paper, the recent researches on trapping sets in LDPC codes are reviewed, and the basic definition, searching methods, elimination methods are systematically introduced. Finally, some unsolved problems, challenges and future development prospects in the current researches on trapping sets and error leveling are further discussed.

Keywords: LDPC codes, Error floor, Trapping sets, Algorithm

1. 引言:

作为一类特殊的具有稀疏校验矩阵的线性分组码，低密度奇偶校验（LDPC）码由于其通用的纠错特性和接近香农限的优良性能站在了数字时代信道编码的前沿[1]，在无线通信[2]、集成电路[3, 4]、深空通信[5]和数字水印[6]等领域吸引了众多研究者的关注和研究。特别地，LDPC 码已经成为了国际 5G 标准中的大容量数据信道编码技术方案[7]。LDPC 码最早由 Gallager 于 1962 年发明[8]，此后在 1981 年，Tanner 利用二分图表示法对这类线性分组码进行了描述[9]。随后在上世纪 90 年代，MacKay 等人重新“发明”了 LDPC 码，他们发现当使用 SPA 算法进行译码时，码长较长的 LDPC 码可以逼近信道容量[10, 11]。此外，Luby 等人将度分布的概念引入 LDPC 码中，并开始通过非规则的二分图来对 LDPC 码进行设计和研究[12]，而 Richardson 等人则将 LDPC 码的分析方法拓展到了更宽

泛的信道（加性高斯白噪声信道）和更多的解码算法中，并引入了新的工具来进一步对这些解码系统进行分析[13]。至此，有关 LDPC 码的研究如雨后春笋般浮现出来，在接下来的二十多年中保持着持久的竞争力。

LDPC 码本质上属于线性分组码，其可以由一个维度大小为 $m \times n$ 的校验矩阵 H 来确定，其中校验矩阵中的每一行代表一个对应的校验方程，每一列代表一个码字。对于一个具体的向量 $\mathbf{c} = [c(1)\ c(2)\ c(3)\ \cdots\ c(m)]$而言，当且仅当该向量满足 $\mathbf{c} \cdot H^T = \mathbf{0}$ 时，其才为一个有效码字。由于校验矩阵具有稀疏性，因此矩阵中的非零元素个数很少，又可以根据每一行和每一列中非零元素的个数是否相同分为规则 LDPC 码和不规则 LDPC 码。此外，还可以用 Tanner 图来更直观地表示 LDPC 码中各个变量节点和校验节点之间的连接关系。在 Tanner 图中，通常用方形来表示校验节点（对应校验矩阵中的 m 行），用圆圈来表示变量节点（对应校验矩阵中的 n 列），当对应校验矩阵中出现非零元素“1”时，即将对应的校验节点和变量节点相连。为了更进一步地分析从 Tanner 图中得到的信息，研究者引入了“度”和“环”的概念。对于 Tanner 图中任意一个节点，与之相连的其他节点的数目称为该节点的“度”；而对于一个变量节点，从该变量节点出发，交替经过若干步从校验节点到变量节点的传递后再次回到该起始变量节点所形成的闭合环路称为 LDPC 码中的“环”，Tanner 图的围长即为其最小环的长度[14-16]。

目前 LDPC 码的译码方案多采用迭代译码方式，若要使得构造好的 LDPC 码具有良好的纠错性能，就要求所构造的码具有低的复杂度、良好的瀑布区域下降速率和低的错误平层[17, 18]。对于改善 LDPC 码的复杂程度上，已经有了一系列的有关构造的研究，主要分为随机构造法（包括：Gallager 随机构造法[8]、PEG 构造法[19]、以及 ACE 构造法[20]等）和结构化构造法（包括有限几何法[21]以

及循环置换矩阵法[22]等）两大类。随机构造法通过对节点的度的分布状况最小环的长度等设置相应的约束条件来对所需要的校验矩阵进行搜寻；而结构化构造法则通过特定的代数和集合等理论来构造出结构确定的校验矩阵。结构化构造法构造出的 LDPC 校验矩阵具有规律的非零元素分布和较低的硬件实现复杂度，因此更容易被研究者关注[23]。结构性码字按照具体的结构特征又可以进一步分为准循环码（QC）[24]、重复累计码（RA 和 IRA 等[25, 26]）和 Raptor-like LDPC 码[27]等。其中，QC-LDPC 码在硬件实现上可以通过移位寄存器实现编码过程，可以节约储存空间、有很高的数据吞吐量且具有接近随机 LDPC 码的优良纠错性能，因此得到了更广泛地的研究，被众多通信标准和系统采纳[28-30]。

对于改善 LDPC 纠错性能图中的瀑布区和错误平层的角度来讲，环和陷阱集是影响 LDPC 码在这两个区域内表现的主要因素。在迭代译码过程中，信息会在节点间交互传递，如果环长越短，信息经过一定传递返回原始节点的路径就会短[31]。环的存在会影响节点的独立性，使得译码的失败效率大大增加。一般来说，LDPC 码中环的存在会影响译码的性能，故而要尽可能地避免短环的存在，尤其是四环和六环，增大 LDPC 码的围长能够提高对应码字的纠错能力。当 LDPC 码中没有短环存在时，围长接近无穷大，其各个节点会相互独立，对应的 Tanner 图会退化为树状图，理论上可以通过和积 SP（Sum-Product）译码算法来使译码器快速收敛，实现最佳译码[21]。然而，环的存在也有一定的好处，并不是围长越大的 LDPC 码性能就越好。这是因为环的存在本身也有一定的好处，即可以改善码的最小汉明距离，进一步提升码的性能[32]。一般地，当 LDPC 码的列重大于等于 3 时，已经基本可以很好地解决码距问题。合理地调控环的大小和分布是影响 LDPC 在瀑布区下降速率的重要因素。此外，很多 LDPC 码会在高信噪比区域

内出现错误平层现象，在这一区域内纠错性能下降速率趋于平缓，并且在较长区域内都无法改变，这是因为 LDPC 码中往往存在着各种各样的陷阱集[33, 34]。在 AWGN 信道下在高信噪比区域中 LDPC 码更容易受到陷阱集的影响。各类陷阱集制约了译码过程中信息在校验节点和信息节点之间的传递，导致在译码过程中，信息的传播陷入闭塞。在此阶段的译码过程中，往往会有一部分错误的信息位无法被纠正。由于目前存储和通信领域的飞速发展对错误平层区域内误比特率的要求越来越高（10^{-12}-10^{-15}），进一步认识和降低错误平层就显得尤为重要，成为 LDPC 编码理论和实际应用中一类亟待解决的问题[35]。由于陷阱集的大小和分布则是导致高信噪比区域内错误平层出现和影响其高低的重要原因，这种特殊的拓扑结构使得当陷阱集中包含的节点发生数据错误时，译码器很在迭代译码的过程中很难检测出错误的发生，从而大大增加了译码的失败概率，因此深入研究陷阱集的形成原因、作用机理以及消除方式，对于优化 LDPC 码在错误平层的性能有着深远的意义[36]。

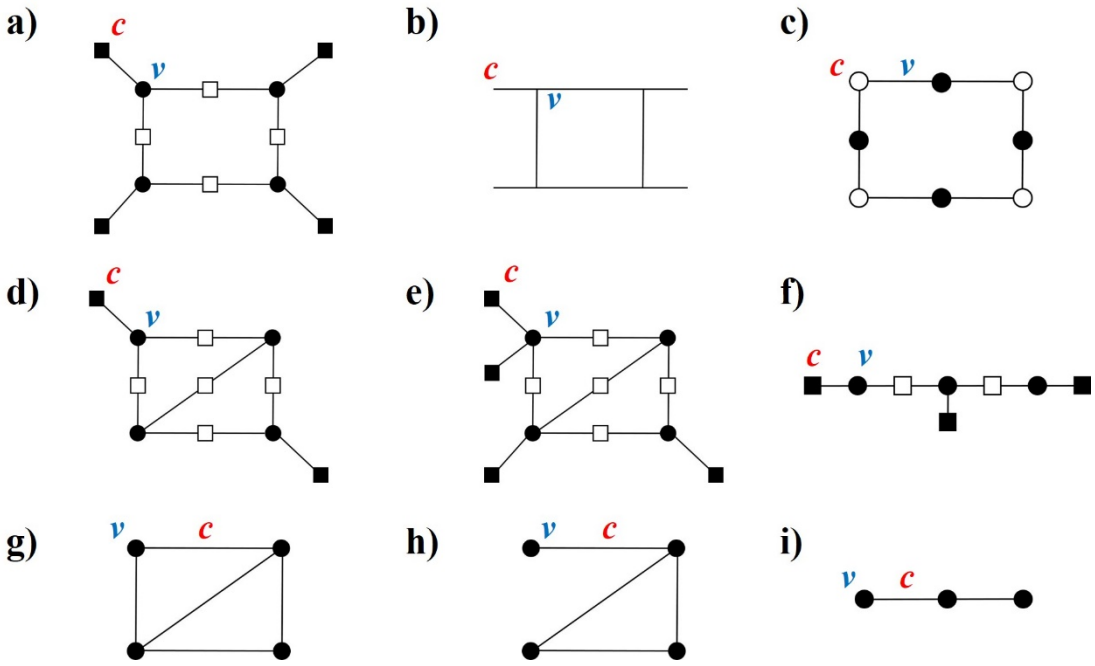
陷阱集最早由 Richardson 等人提出[37]，在此后关于 LDPC 码的研究中，关于陷阱集的研究也成为了极其重要的一部分。本文综述了 LDPC 码中有关陷阱集研究和分析的最新进展和研究成果。包括其基本的定义、分类、检索方式以及改善方法。本文的主要结构如下：在第二节中介绍了陷阱集的基础定义，具体分类方法，以及其对于 LDPC 码的作用机理。第三节中介绍了 LDPC 码中陷阱集的检索方式。在第四节中介绍了改善陷阱集的方法，包括矩阵的构造的方法以及特点。第五节提出了相应译码算法和译码方法改进的具体措施。第六节是结束语和对未来研究的展望。我们希望我们的综述可以帮助读者更好地了解 LDPC 码中陷阱集的存在方式和基本运作特点，并为读者指出有关 LDPC 码陷阱集的文献。

进一步地，据我们所知，这篇综述是目前为数不多的有关 LDPC 陷阱集的综述，可以为相关 LDPC 在陷阱集中新的研究带来启发。

2. 陷阱集基础

2.1 陷阱集的基础定义和分类

假设 T(y)是一个最终会造成译码失败的所有输出码字的比特集合，如果 T(y)不为空集，则其子图中变量节点和校验节点所构成的集合即为陷阱集。一个(a, b)类型的陷阱集包括 a 个变量节点和与这些变量节点相连的所有校验节点。其中有 b 个校验节点的度为奇数[38]。如果 a 和 b 的值越小，则其对错误平层的影响也就越大。如果校验节点最多只连接两个变量节点，则称该陷阱集为基本陷阱集（Elementary Trapping Sets）[39]。基本陷阱集可以按照变量节点连接的度数为偶数的校验节点个数分为无叶基本陷阱集（Leafless Elementary Trapping Sets, LETS）和有叶基本陷阱集（Elementary Trapping Sets with Leaf, ETSL）。而有叶基本陷阱集又可以进一步按照是否有环的存在来分为 $ETSL_1$ 和 $ESTL_2$ 两类[40]。陷阱集通常可以用其对应的 Tanner 图来表示，进一步地，其可以简化为用顶点代表变量节点，用游离的边代表其奇校验节点的形式。此外，用点和线分别代表校验节点和变量节点的点线法也经常被使用。进一步地，移除度数为奇数的校验节点及其连接线，并将度数为偶数的校验节点用直线代替的正态图表示法也被报道[41]。图 1 表示了上述不同类型的陷阱集以及陷阱集的几种表示方式。



(a) (4, 4)陷阱集的 Tanner 图；(b) (4, 4)陷阱集的简化形式；(c) (4, 4)陷阱集的点线图；(d, g) (4, 2)无叶基本陷阱集及其正态图；(e, h) (4, 4)有叶基本陷阱集及其正态图 (ETSL)；(f, i) (3, 3)有叶基本陷阱集及其正态图

图 1. 不同陷阱集的不同表示方式（其中 c 为校验节点，v 为变量节点）

2.2 陷阱集对 LDPC 译码过程的影响

陷阱集形成的原因有多种因素，主要包括校验矩阵 H 的结构以及解码的方式。其中校验矩阵中存在的环结构是陷阱集形成的最基本原因[42]。这是因为校验矩阵中存在的环是影响陷阱集中变量节点和校验节点之间拓扑结构的重要原因，校验矩阵中存在的环可以经过演变形成不同的陷阱集[43]。因此改善校验矩阵的构造方法就成为了克服陷阱集影响的一种重要方式。此外，解码方式的不同也是陷阱集形成的主要因素。解码的过程往往包含初始信息以及信息传递过程中的量化等，不同的解码方式往往会形成不同种类的陷阱集。我们将在后续章节的叙述中详细给出相关的解决方法。

在 LDPC 译码过程中，陷阱集中的变量节点往往会同时连接度数为偶数的校验节点和度数为奇数的校验节点。在信息传递的过程中，度数为偶数的校验节点会给该变量节点传递错误的信息，度数为奇数的校验节点会给该变量节点传递正确的信息。在这个过程中，错误信息在解码过程中往往会带来比正确信息更大的影响，所以所涉及的节点会在信息的传递过程中一直处于错误的状态。这就是陷阱集在译码过程的作用机理[40-43]。

陷阱集也会根据译码的方式不同而有不同的表现方式。当陷阱集在译码后期处于稳定状态时，各个节点中传递的信息在每次迭代中基本不会发生任何变化，和上一次迭代的结果基本相同（图 2a），或者在译码过程中会出现误码率随迭代次数逐渐增加然后趋于稳定（图 2b），此外，在译码的后期陷阱集有时也会处于一种动态循环的状态，即每次迭代的信息会根据迭代次数的增加发生周期性的变化，造成误码率也随之发生波动性变化。这是由于陷阱集和正确信息的节点会随迭代次数的增加存在着此消彼长的现象，两者都有向外传递信息的趋势，但是两者之间并未达到平衡，并随迭代次数的增加产生周期性的波动（图 2c）[44]。陷阱集在这个过程中会周期性地增大缩小，但值得注意的是，基本陷阱集在这个过程中不发生任何变化，这也是基本陷阱集对 LDPC 译码危害更大的原因。

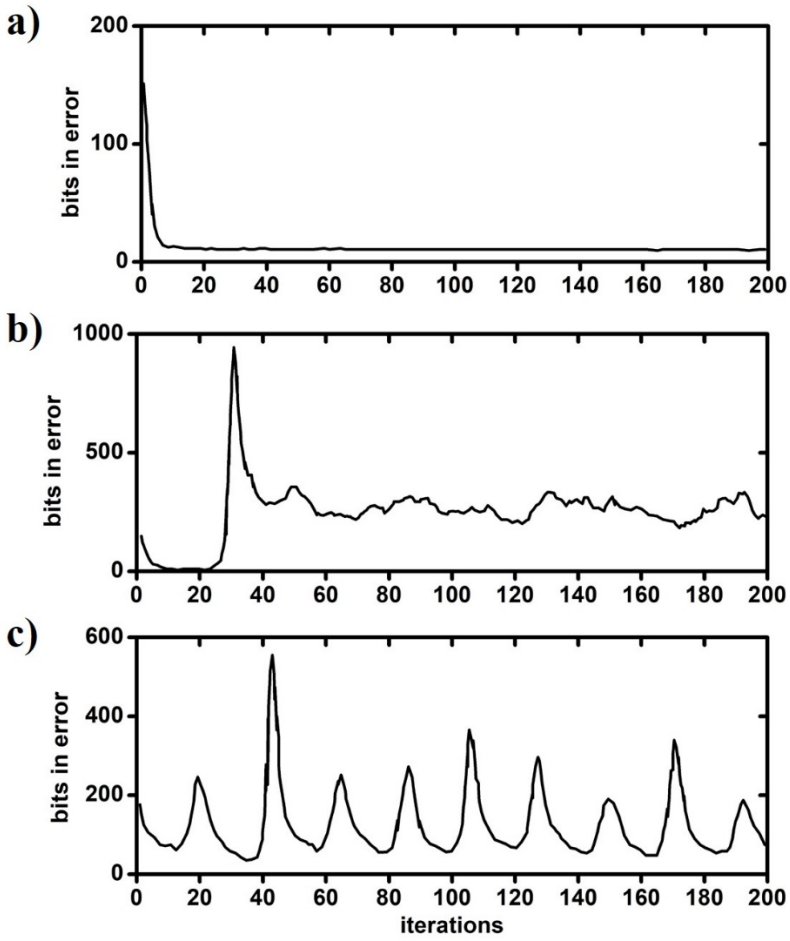


图 2 三种不同陷阱集的表现形式[44]

2.3 基本陷阱集的分析

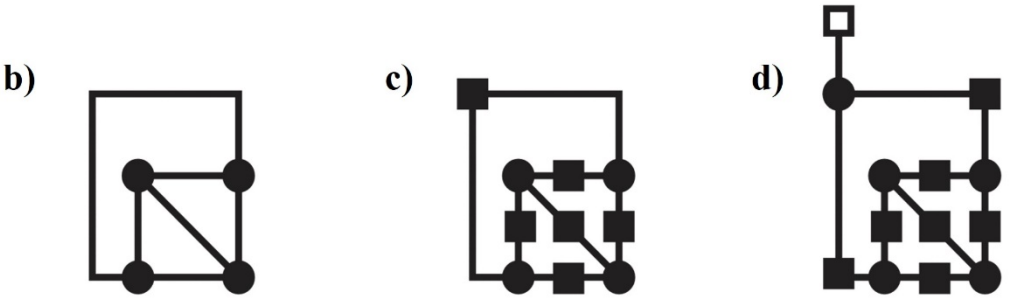
陷阱集所连接的变量节点和校验节点个数对错误平层有着很大的影响，这是因为在错误平层区域，信道错误的可能信很低，在译码过程中译码器被困在陷阱集中的概率会随着陷阱集大小的增大而迅速降低。Hashemi 等人推导了非规则 LDPC 码中具有给定 b 的最小可能非基本陷阱集的大小的下界，并证明了具有与相同 b 值的基本陷阱集相比，其下界要更大，进一步为证明基本陷阱集的危害更大提供了理论依据[45]。Dehghan 等人则建立了笼（给定围长的具有最小节点数的规则图）与陷阱集之间的联系，并推导出了在给定变量节点度数和围长的规则 Tanner 图中当 b 已知时，(a, b)陷阱集中 a 的最小值的严格上界。不同类型的陷阱

集的边界是确定的。当 b 等于 0 时，即可得到 LDPC 码最小距离的上界。这一工作将陷阱集的最小大小（或 LDPC 码的最小码距）同码的围长和度分布联系起来，对识别陷阱集和最小大小的非零码字结构有很大的意义[46]。Karimi 等人则基于 LDPC 码对应的 Tanner 图中的短环研究了规则 LDPC 码中基本陷阱集的图形结构（图 4）。经过对任意一类基本陷阱集的非同构结构的表征和确认，他们证明了任意一类陷阱集都存在一个嵌套的基本陷阱集序列（即分层超集 LLS），该序列从环开始，每次增长一个变量节点，从而生成最后的陷阱集[47]。随后，他们还证明了 $ETSL_2$ 对非规则 QC-LDPC 码的错误平层没有影响，且非规则 QC-LDPC 的 Tanner 图中若不含度数为奇数校验节点，则其基本陷阱集中的 b 不为奇数[48]，这为构造校验矩阵，搜索和消除其中的陷阱集提供了新的思路。

a)

$NETS$ / $\overline{ETS}^{\lceil LETS \rceil}_{(ETSL)}$		$d_v = 3$			$d_v = 4$			$d_v = 5$			$d_v = 6$		
		$g=6$	$g=8$	$g=10$	$g=6$	$g=8$	$g=10$	$g=6$	$g=8$	$g=10$	$g=6$	$g=8$	$g=10$
	$b=0$	$8^{-}_{4^{(4)}_{(-)}}$	$12^{-}_{6^{(6)}_{(-)}}$	$18^{-}_{10^{\lceil 10 \rceil}_{(-)}}$	$7^{-}_{5^{(5)}_{(-)}}$	$16^{-}_{8^{(8)}_{(-)}}$	$25^{-}_{17^{\lceil 19 \rceil}_{(-)}}$	$8^{-}_{6^{(6)}_{(-)}}$	$20^{-}_{10^{\lceil 10 \rceil}_{(-)}}$	$36^{-}_{26^{\lceil 30 \rceil}_{(-)}}$	$9^{-}_{7^{(7)}_{(-)}}$	$24^{-}_{12^{\lceil 12 \rceil}_{(-)}}$	$49^{-}_{37^{\lceil ? \rceil}_{(-)}}$
	$b=1$	$5^{-}_{5^{(5)}_{(-)}}$	$9^{-}_{5^{(7)}_{(-)}}$	$13^{-}_{9^{\lceil 11 \rceil}_{(-)}}$	no TS			$7^{-}_{7^{(7)}_{(-)}}$	$15^{-}_{9^{\lceil 13 \rceil}_{(-)}}$	$31^{-}_{25^{\lceil ? \rceil}_{(-)}}$	no TS		
	$b=2$	$6^{-}_{4^{\lceil 4 \rceil}_{(6)}}$	$8^{-}_{4^{\lceil 6 \rceil}_{(8)}}$	$12^{-}_{6^{\lceil 10 \rceil}_{(12)}}$	$6^{-}_{4^{(5)}_{(-)}}$	$11^{-}_{7^{(8)}_{(-)}}$	$20^{-}_{12^{\lceil 19 \rceil}_{(-)}}$	$8^{-}_{5^{(6)}_{(-)}}$	$14^{-}_{8^{\lceil 10 \rceil}_{(-)}}$	$30^{-}_{20^{\lceil ? \rceil}_{(-)}}$	$8^{-}_{7^{(7)}_{(-)}}$	$17^{-}_{10^{\lceil 12 \rceil}_{(-)}}$	$42^{-}_{30^{\lceil ? \rceil}_{(-)}}$
	$b=3$	$5^{-}_{3^{\lceil 3 \rceil}_{(5)}}$	$7^{-}_{3^{\lceil 5 \rceil}_{(7)}}$	$11^{-}_{5^{(9)}_{(11)}}$	no TS			$7^{-}_{5^{(7)}_{(-)}}$	$13^{-}_{7^{\lceil 11 \rceil}_{(-)}}$	$29^{-}_{19^{\lceil ? \rceil}_{(-)}}$	no TS		
	$b=4$	$4^{-}_{-^{\lceil 4 \rceil}_{(4)}}$	$6^{-}_{-^{\lceil 4 \rceil}_{(6)}}$	$8^{-}_{-^{(8)}_{(10)}}$	$5^{-}_{3^{(4)}_{(6)}}$	$9^{-}_{4^{(7)}_{(9)}}$	$15^{-}_{7^{\lceil 18 \rceil}_{(20)}}$	$6^{-}_{4^{(6)}_{(8)}}$	$12^{-}_{6^{\lceil 10 \rceil}_{(14)}}$	$24^{-}_{14^{\lceil ? \rceil}_{(?)}}$	$7^{-}_{5^{(7)}_{(-)}}$	$15^{-}_{8^{\lceil 12 \rceil}_{(-)}}$	$35^{-}_{23^{\lceil ? \rceil}_{(-)}}$
	$b=5$	$5^{-}_{-^{\lceil 5 \rceil}_{(5)}}$	$5^{-}_{-^{\lceil 5 \rceil}_{(5)}}$	$7^{-}_{-^{\lceil 5 \rceil}_{(9)}}$	no TS			$7^{-}_{5^{\lceil 5 \rceil}_{(7)}}$	$11^{-}_{5^{(9)}_{(11)}}$	$23^{-}_{13^{\lceil ? \rceil}_{(?)}}$	no TS		

b) **c)** **d)**



(a) 当 Tanner 图变量节点度数为 3、4、5、6，围长为 6、8、10，且 b≤5 时，基本陷阱集（有叶和无叶基本陷阱集）的最小大小及文章中推论关

于基本陷阱集和非基本陷阱集的大小的下界[45]；(b-d) (3; 3)-笼 G、$G^{1/2}$ 的 Tanner 图（dv=3 和 g=6 时变正则 LDPC 码的 a (4, 0) LETS 结构）以及将 $G^{1/2}$ 修改为 a (5,1) LETS 结构[46]。

图 3. 陷阱集最小规模及笼结构图。

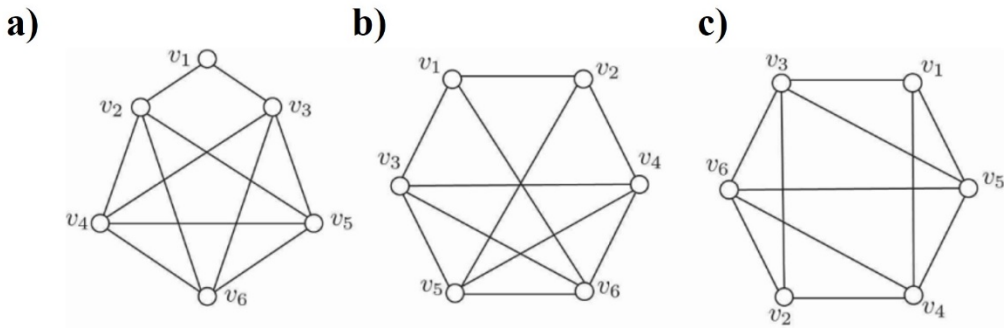


图 4. (6,2) 基本陷阱集在 d_l=4, g = 6 时的左-规则 LDPC 码中的所有可能的非同构正态图。

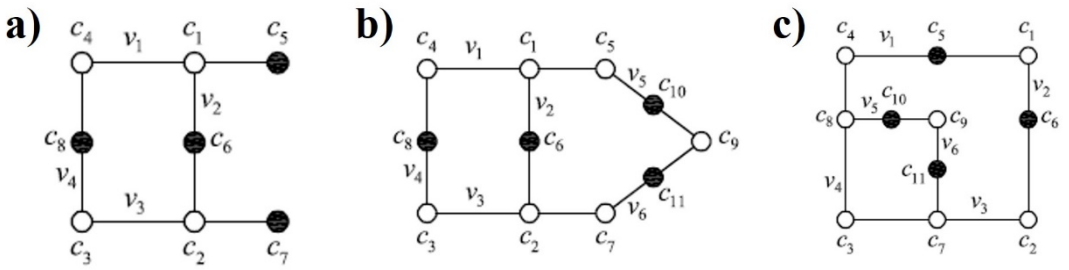
此外，Dehghan 等人计算了随机 LDPC 码集成中基本陷阱集的渐近期望多重性，计算的渐近期望值与随机选择的有限长度 LDPC 码中的基本陷阱集的多重性相匹配，即使在相对较短的块长度下也是如此，为分析 LDPC 码的错误平层提供了新的见解[49]。基本陷阱集是陷阱集的主要类别，对于许多 LDPC 码而言，由多个环组成的基本陷阱集主导了其译码过程中的错误平层特性。因此关于基本陷阱集的研究对于正确认识陷阱集的作用、降低 LDPC 码的错误平层有着很大的意义。

3 陷阱集搜索和测量

3.1 陷阱集的搜索方法。

基本陷阱集的存在导致了 LDPC 码在译码过程中的错误平层现象，导致了译码性能的下降。因此，发现 LDPC 码中的陷阱集并对它进行适当的调控就成为了研究者研究的重点。然而，在 LDPC 码中找到陷阱集是一个复杂程度为 NP-Hard 问题，需要消耗大量的资源[50]。Dehghan 等人也证明了对于一个一般的 LDPC 码，寻找一个给定大小为 a 且具有最少奇数次校验节点 b 的无叶基本陷阱集在任何近似因子内的难易程度都是 NP-Hard[51]。但是不可避免地，寻找适当的方法来搜索陷阱集，对于 LDPC 码的分析和设计有着很重要的作用。Wang 等人提出了一种高效的算法来详尽地枚举不超过一定大小的陷阱集。使用搜索结果作为代价函数，可以进一步优化任意有限长度代码的错误平层。然而，这一算法仅适用于码长在 500 左右的 LDPC 码，如果代码长度远远大于 500，则该方法将变得过于复杂[52]。

Vasić 等人推导出了一种系统的方法来识别最相关的陷阱集。他们总结了陷阱集之间的拓扑关系、阐述了陷阱集临界数的概念并建立了基本陷阱集数据库。在具体的操作中，以(4, 4)陷阱集为例，可以在图 5a 看到，代表校验节点的黑色阴影点 c5 和 c7 分别位于代表变量节点 v1 和 v3 的直线上。通过增加 v5 和 v6 两个变量节点，使 v5 连接到 c5，v6 连接到 c7，c9 是变量节点 v5 和 v6 的共同的相邻节点，于是得到如图 5b 所示的(6, 4)陷阱集。这个陷阱集是 8 环和 10 环的并集。如果增加两个变量节点 v5 和 v6，使 v5 和 v6 分别连接到 c8 和 c7, c9 仍然是 v5 和 v6 的共同的相邻节点，也得可以到如图 5c 所示的(6, 4)陷阱集。然而，这个(6, 4)陷阱集是两个 8 环的并集。通过上述方法构建的陷阱集本体论可以对陷阱集的危害进行评估，并且对 LDPC 码的设计有一定的指导作用[53]。



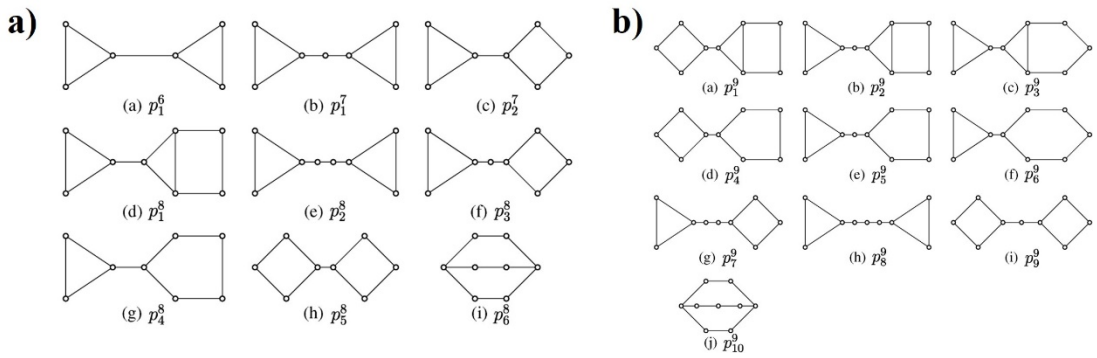
(a) (4, 4)陷阱集的另一种形式；(b) (6, 4) 8 环和 10 环并集形成的陷阱集；(c) (6,4)两个 8 环并集形成的陷阱集。

图 5. 从(4, 4)陷阱集得到(6, 4)陷阱集[53]

Milenkovic 等人基于枚举技术和大偏差理论提出了一种提出了一种新方法来分析陷阱集的渐近行为，并得到了关于随机、规则和不规则 LDPC 码中陷阱集集分布的综合结果[54]。Falsafain 等人则提出了一种有效的分支定界算法来穷举给定非规则 LDPC 码中基本陷阱集的方法。他们通过引入了一个新的 0-1 整数线性规划公式，即寻找 Tanner 图中存在一个基本陷阱集(ω，ν)的最小 ω，然后基于此公式的 LP 松弛进行分支定界过程。这个公式可以在搜索过程中产生强（紧）松弛，从而使搜索空间可以得到有效地修正。这种方法不要求输入 Tanner 图的形式，具有一定的普适性[55]。

Karimi 等人提出了一个可以保证从短环开始找到所有具有 LSS 结构的基本陷阱集的搜索算法[47]，在此基础上，如图 6 所示，Hashemi 等人通过证明基本陷阱集的其余结构都是非短环结构的其他图形结构的 LSS 来进一步对这个理论进行了补充。他们证明了对于变量节点度数为 3，4，5 和 6；围长为 6 和 8；a，b 均小于 10 的(a, b)陷阱集来说，其非环 LLS-prime 结构都是由两个短环组成的。它们均符合以下两个条件中的一个，即两个环共享一个变量节点或多个变量和校

验节点（及其连接边）或者两个环之间由一个链连接（交错的变量和校验节点）。这一搜索算法可以确保找到任意规则 LDPC 码中任意(a, b)基本陷阱集，大大改善了穷举搜索技术的复杂性[42]。

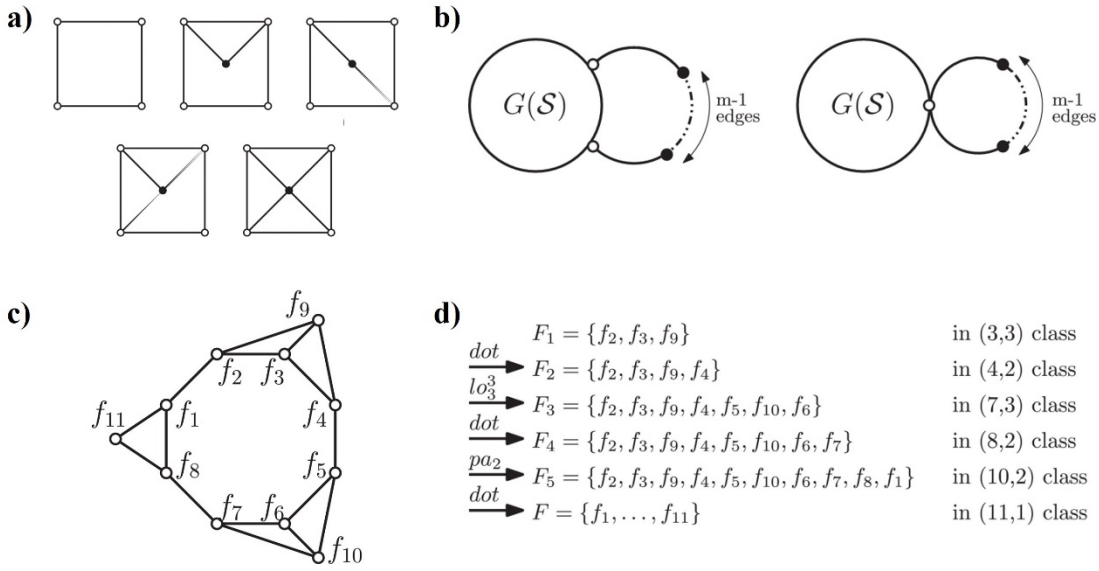


(a) 变量节点度数为 3，围长为 6 的规则 Tanner 图中，大小为 6，7，8 的所有非同构 LLS-prime 基本陷阱集结构的正态图；(b) 在变量节点度数为 3，围长为 6 的规则 Tanner 图中，大小为 9 的非同构 LLS-prime 基本陷阱集结构的正态图。

图 6. 不同大小的非同构 LLS-prime 基本陷阱集结构的正态图[42]

尽管基于 LLS 的陷阱集搜索算法较为简单，但是其仍然面临着两种问题：首先对于无叶陷阱集来讲，并非每个无叶陷阱集结构都是环组成的 LLS；其次，基于 LLS 的搜索算法通常需要枚举长度远远大于 Tanner 图围长的环，且环的多样性会随着环长度的增加而迅速增大。这两个问题往往会导致环的枚举、存储和处理成为实现搜索算法的实际障碍。特别对于校验节点和变量节点有较大的度数，以及具有较大 a 和 b 值的无叶陷阱集时，这一情况变得尤为明显。鉴于此，Hashemi 等人继续报道了规则 LDPC 码中无叶基本陷阱集的新表征方案，该表征方案基于一个根据分层图的扩展方法，涉及 LSS 扩展和两种额外的扩展技术，通过使

用嵌入的无叶基本陷阱集结构序列来表征每个无叶基本陷阱集结构，这一序列从一个简单的环开始，并在每一步使用三个展开（即 Dots，Path 以及 Lollipop，简称 DPL 表征）中的一个扩展到更大的无叶基本陷阱集，直到达到关注的无叶基本陷阱集结构（如图 7）。这一表征还对应着一种高效的无叶陷阱集的穷举搜索算法，该算法只需要 Tanner 图中简单短环作为输入，且输入的最大环长也通常比基于 LLS 的搜索算法更短[41]。进一步地，他们又将这种表征方法推广到了非规则 LDPC 码的结构中。通过利用规则 LDPC 码中无叶基本陷阱集结构的 DPL 表征来进一步详尽地表征在不规则 LDPC 码中给定 a 和 b 值的范围内的无叶基本陷阱集结构。表征的方法仍然基于基本陷阱集的分层图形表示，从简单的环或者单一变量节点开始，在 a、b 均在最大值的范围内，对无叶基本陷阱集结构中大小为 a'的子结构中可能出现的不满足校验节点 b 的最大数目使用递归推导得到上界$b_{max}^{a'}$序列，进而得到基本陷阱集的所有实例。这一方法也可以应用在有叶基本陷阱集的搜索中。这一算法同传统的方法相比，效率更高，而且平均复杂度是恒定的，具有一定的通用性[56]。



(a) 从(4, 8)无叶基本陷阱集到(5, 4)无叶基本陷阱集的 Dots 扩展 (b) 用长度为 m+1 的开放（左）和闭合（右）Path 拓展无叶基本陷阱集结构 S；(c) 变量节点度数为 3，围长为 6 的规则 Tanner 图中的一类(11,1)无叶基本陷阱集结构；(d) 从 S3 生成图(c)中所示的(11, 1) 无叶基本陷阱集的步骤[41]

图 7. DPL 拓展示意图

除了基本陷阱集，非基本陷阱集的搜索方法也被研究者关注。在 Tanner 图中，由停止集引起的子图只包含度数为 2 或 2 以上的校验节点。停止集一般根据子图中的校验节点度的数分为两类，若所有的校验节点的度数为 2，则该停止集为初等集（ESS），否则为非初等集（NESS）。非初等集是非基本陷阱集的一个子集。Hashemi 等人提出了 LDPC 码中非基本陷阱集的特征，他们通过将非基本陷阱集视为从基本陷阱集开始的嵌入式图形层次结构，推导出了 LDPC 码的停止距离 Smin 的上下界。同他们之前提出的搜索方法效果类似，通过大量规则和不规则的 LDPC 码，证明了这一方法具有很好的精度和效率。然而值得注意的是，当码长较长时，大尺寸的停止集的穷举搜索仍然可能过于复杂而无法执行[57]。

对于 QC-LDPC 码而言，Karimi 等人基于 DPL 拓展方法对其中的基本陷阱集结构进行了表征，并且证明了随机构造的 LDPC 码中可能存在的部分基本陷阱集结构在 QC-LDPC 码中是不存在的。考虑到集合中陷阱集之间的父子关系，他们提出了一种最小父结构的搜索算法，将目标由找到某个基本陷阱集结构集合的所有实例调整为验证 LDPC 码中是否存在至少一个目标结构的实例，与 Hashemi 提出的搜索穷举算法相比，其复杂度大大降低[48]。Yacoub 等人则利用生成函数提出了一种有效的方法来获得（初等）完全吸收集的归一化对数渐近分

布，并用该方法推导了基于二进制原型的 LDPC 码的平均有限长（初等）全吸收集（陷阱集的一种特殊形式）计数器[58]。进一步地，在此基础上他们又考虑了有约束和无约束的边缘标记方法得到了基于非二进制原型的 LDPC 码的有限长度陷阱集和（初等）吸收集计数器。通过求解方程组的有效方法，他们进一步得到了非二进制原型的 LDPC 码中陷阱集和（初等）吸收集的归一化对数渐近分布，这些分部可以进一步用来评估 LDPC 码的错误平层性能[59]。

3.2 从环到陷阱集

搜索陷阱集的更有效方法目前还在被研究者不断地探索。考虑到其难度较大，在实际构造 LDPC 码字的过程中，更多方法则是通过考虑到 LDPC 码和环之间的关系来进行间接的研究。由于大部分陷阱集中都会包括环结构，因此可以通过研究 LDPC 码字中的短环和围长来对相关的陷阱集进行进一步搜索。Amirzade 等人给出了任意围长为 g 的规则 LDPC 码和围长为 8 的不规则 LDPC 码的基本陷阱集(a, b)大小的下界。这一结果主要取决于码的围长，或者 a，b 和码的列重值 γ。对于围长为 8 的规则 LDPC 码而言，其不含有满足$b < a\gamma - \frac{a^2}{2}$的(a, b)基本陷阱集，对于满足$\frac{b}{a} < 1$性质的基本陷阱集，有 $a \geq 2\gamma - 1$。若一个围长为 8 的 Tanner 图中的基本陷阱集具有最小的尺寸 $a = 2\gamma - 1$，则其度数为 1 的校验节点的个数满足 $b \geq y$。对于围长为 8 的规则 LDPC 码中的满足$\frac{b}{a} < 1$性质的基本陷阱集，所得到的 a、b 的下界和所有列重的值 γ 都是密切相关的，这一方法同样适用于列重为 2-6 的不规则 LDPC 码（如图 8）；对于围长为 10 的 LDPC 码来讲，其不含有满足$b \leq a\gamma - \frac{a^{3/2}}{2}$的(a, b)基本陷阱集，而且围长为 10 的规则 LDPC 码中不包含$\frac{b}{a} < 1$且 $a \leq (\gamma\text{-}1)^2$ 的基本陷阱集[60]。这一方法相比之前文献报道的更为严格。

除此之外，在 QC-LDPC 码中，Fossorier 根据索引矩阵 P 提出了校验矩阵中存在长度为 2i 环的充要条件：$\sum_{k=0}^{i-1}\left(P_{i_k,\ j_k}-P_{i_{k+1},\ j_k}\right)=0 \bmod \mathrm{L}$。其中 L 为子矩阵维度，P 为索引矩阵中的元素，通过该公式来对矩阵中的短环进行判断和消除，进而改善其中的陷阱集[61]。

$a=4$	(4,1) ETS with $d(v)\in\{2,3\}$, (4,2) ETS with $d(v)\in\{2,4\}$ (4,3) ETS with $d(v)\in\{2,5\}$, (4,3) ETS with $d(v)\in\{2,3,4\}$
$a=5$	(5,4) ETS with $d(v)\in\{2,6\}$, (5,4) ETS with $d(v)\in\{3,4\}$ (5,2) ETS with $d(v)\in\{2,3,5\}$, (5,3) ETS with $d(v)\in\{2,3,6\}$ (5,3) ETS with $d(v)\in\{2,4,5\}$, (5,4) ETS with $d(v)\in\{2,4,6\}$ (5,4) ETS with $d(v)\in\{2,3,4,5\}$
$a=6$	(6,2) ETS with $d(v)\in\{3,5\}$, (6,3) ETS with $d(v)\in\{3,6\}$ (6,3) ETS with $d(v)\in\{2,5,6\}$, (6,3) ETS with $d(v)\in\{3,4,5\}$ (6,4) ETS with $d(v)\in\{3,4,6\}$, (6,5) ETS with $d(v)\in\{3,5,6\}$ (6,5) ETS with $d(v)\in\{2,4,5,6\}$, (6,4) with $d(v)\in\{2,3,5,6\}$
$a=7$	(7,5) ETS with $d(v)\in\{4,5\}$, (7,6) ETS with $d(v)\in\{4,6\}$ (7,3) ETS with $d(v)\in\{3,4,5,6\}$, (7,4) ETS with $d(v)\in\{2,3,4,5,6\}$
$a=8$	(8,3) ETS with $d(v)\in\{4,5,6\}$
$a=9$	(9,6) ETS with $d(v)\in\{5,6\}$

图 8. 围长为 8 且列重为{2,3,4,5,6}子集的不规则 LDPC 码的最小基本陷阱集大小。

4 基于矩阵构造的陷阱集改善方法

陷阱集形成的重要原因之一是校验矩阵 H 的结构，因此在构造矩阵时消除部分有害的陷阱集就成为了看研究者研究的主要方向。上文的叙述中我们可以发现，陷阱集和 LDPC 码字中的环是密切相关的，大部分陷阱集都会包括一个或者多个短环。因此，增加矩阵中环的长度或者对校验矩阵中部分特定的陷阱集进行消除，对于改进陷阱集有着很大的意义。

4.1 基于 PEG 算法的矩阵构造

在校验矩阵的构造方法中，PEG 法是经常用到的一种。PEG 算法是一种贪心算法，它是为了使 LDPC 码 Tanner 图的局部围长最大化而提出的，在利用该方法构建矩阵时，每一个节点扩展步骤都尽可能地选取最长的节点路径，即可确保矩阵有较大的围长[19, 62-64]。在 PEG 算法的基础上，也有部分研究者为避免矩阵中的陷阱集对错误平层的影响做出了努力。Khazraie 等人在原有的 PEG 算法中加入了一种高效的递归搜索算法，他们通过添加每个变量节点及其相邻边来搜索构建 LDPC 码时在 Tanner 图中形成的基本陷阱集，从而在利用 PEG 算法构建矩阵的每一次选边步骤中都运行一次基本陷阱集搜索算法，避开可以产生基本陷阱集的边，这一方法度分布规则和不规则的变量节点均适用，所构建的代码具有优越的基本陷阱集分布和错误平层性能（图 9a）[65]；Diouf 等人基于 PEG 算法提出了一种构造围长为 8、列重为 3 的规则 LDPC 码的预测方法，其对应的 Tanner 图包含最小数量的小陷阱集。他们提供了如何在 PEG 算法里使用的计算树中检测陷阱集的方法，并预测了哪些候选边可以形成陷阱集。然后他们提出包含一个代价函数的 Randomized-PEG 算法，使得在构造过程中（5，3）陷阱集可以被完全消除，并使得（6，4）陷阱集的数量最小化。该算法对小型 LDPC 码的性能有着较大提升（图 9b）[66]。Zheng 等人则在构造短码长不规则 LDPC 码的过程中引入了参数 e，用其来识别由基本陷阱集引起的连通子图中可区分循环的数量，并提出了一种新的构造算法 PEG-ACSE（Approximate-minimum-Cycle-Set-Extrinsic-message-degree）。该算法构造出的 LDPC 码可以有效避免小型基础陷阱集，特别是有害基础陷阱集，也有着较好的纠错性能（图 9c）[67]。

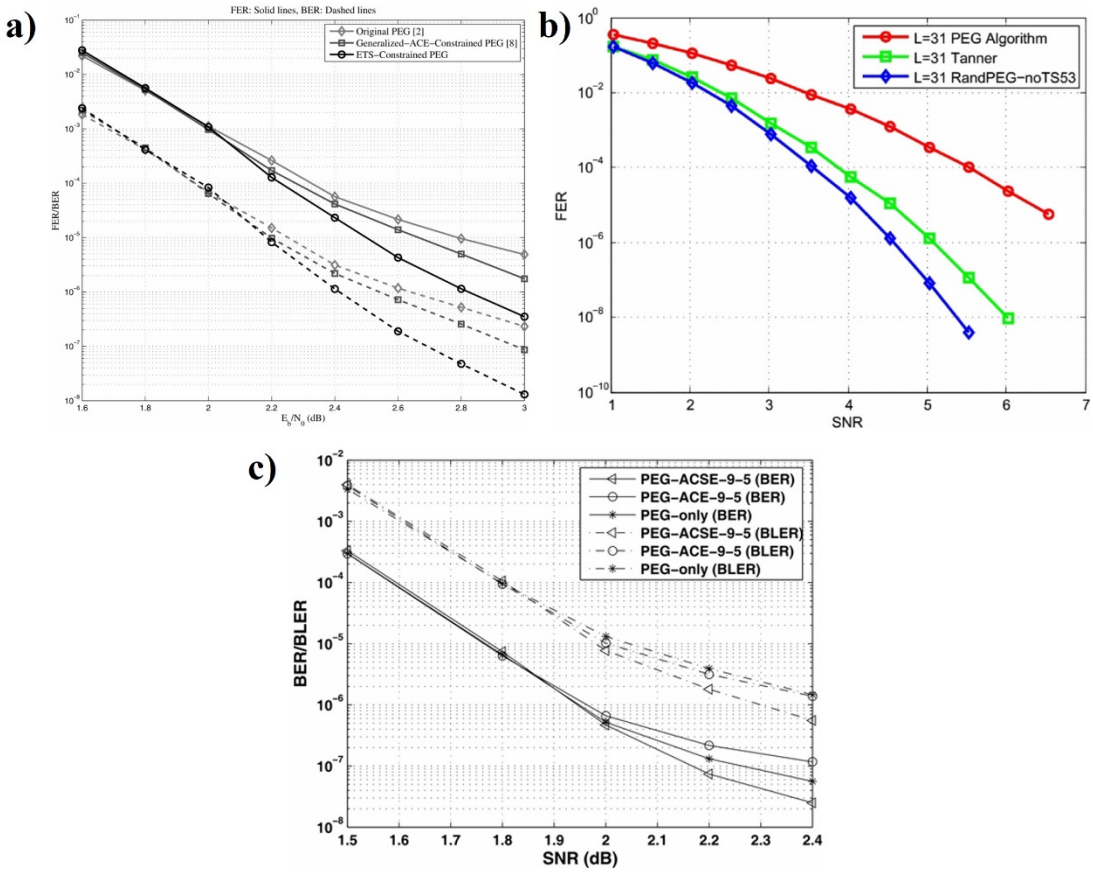


图 9. 不同 PEG 方法构造的 LDPC 码的性能。

值得注意的是，虽然利用 PEG 算法可以使整个矩阵的围长和环的长度变大，但是其往往也带来了较为复杂的计算难度。不过目前基于 PEG 算法构造 LDPC 码的方法仍有不少在使用和研究。

4.2 基于有限几何与有限域算法的矩阵构造

Kou 等人基于有限域上欧几里德几何和射影几何的线和点，构造了四类 LDPC 码，提出了利用有限几何构造 LDPC 码的方法。有限几何由若干个点构成，并且满足以下条件：任意两点之间均存在一条直线使其相互连接；每条直线上都有 P 个点，而每个点又最多由 Q 条线相互交叉形成；每两条线都不会相互重合，

即平行或者有且只有一个交点[68]。利用有限几何法构造矩阵是一种经典的结构化构造方法，通过将有限几何中的点线同校验矩阵中的行列相互对应，即可以得到不存在短环的校验矩阵。同这种方法也可以对 LDPC 码中存在的陷阱集进行调控[69-71]。

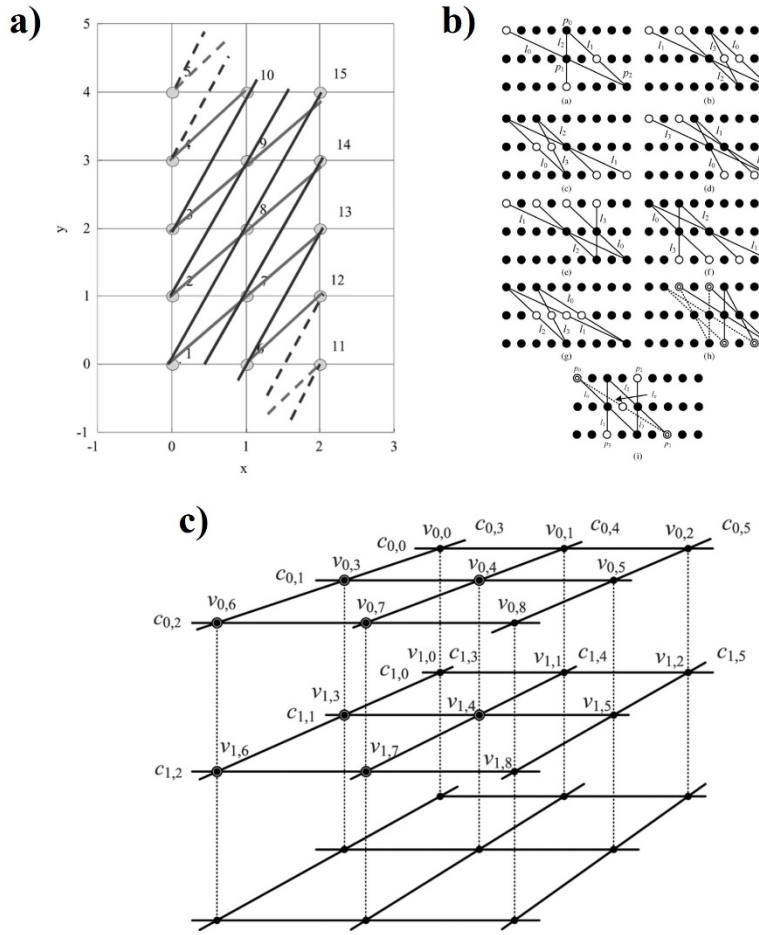
除了有限几何，基于有限域的 LDPC 码构造也是一种可以避免短环和陷阱集对错误平层影响的重要方法[23, 72, 73]。Zhang 等人提出了一种基于有限域上的拉丁方阵构造 QC-LDPC 码的方法。通过分析这些码的校验矩阵，推导出它们的秩和维数的组合表达式。这类码相比代数构造的 QC-LDPC 码要大得多，且有着优秀的译码性能[74]。进一步地，Nguyen 等人提出了一种在 BSC 信道上构建低错误平层的 LDPC 码的方法。他们利用不同陷阱集之间的拓扑关系，通过分析目标陷阱集在是否存在其他陷阱集的情况下的译码失败情况来评估不同陷阱集对和积算法的相对危害性。该方法可以应用于结构化构造 LDPC 码并具有较好的错误平层性能[75]。

4.3 QC-LDPC 码的构造

在现有的结构化 LDPC 码构造中，QC-LDPC 码的设计引起了研究人员的极大兴趣，这主要是因为 QC-LDPC 码在编解码实现上具有显著的优势，并且只需少量的存储单元即可存储编码矩阵。在设计 QC-LDPC 码构造的方法中，上述已经介绍的 PEG 法、有限几何和有限域的方法均有所涉及。我们在这一小节中将重点介绍下面几种。

Vasić 等人介绍了在整数格子上构造围长为 8 的高码率 LDPC 码的方法，他们将校验矩阵定义为基于矩形整数格上 1 位型的点线关联矩阵（图 10a），并通

过位型中包含的平行线集来实现围长为 8 的性质，得到了长度和列重范围都较宽的一类 LDPC 码[76]。在此基础上，Tao 等人进一步分析了小的基本陷阱集和矩形格中连线斜率的关系，选择合适的斜率并避免了基础陷阱集的存在，用这种方法构建的 QC-LDPC 码在 AWGN 信道上具有良好的误码率性能[77]。此外，Tao 等人继续提出了一种通过去除 Tanner 图中的小陷阱集来构造围长为 8 的低错误平层的 QC-LDPC 码的方法，他们发现如果 Tanner 图中没有某些八环，则 a≤8 和 b≤3 范围内的任意（a，b）陷阱集都会被自然消除。该方法在 AWGN 信道上具有较低的误码率（图 10b）[78]。在 2D 矩形格的基础上，Tao 等人进一步将该方法扩展至 3D 矩形格（图 10c），通过分析层内多边形和层间多边形的控制条件以及分层 LDPC 码的短环和小陷阱集之间的关系，并进一步对层内和层间线的连线斜率进行筛选，最终构造了两类 QC-LDPC 码，第一类码的的围长为 8，且没有任何 a≤8 和 b≤3 的（a，b）陷阱集，第二类码中无八环，围长为 10，这两类码在 AWGN 信道上均有着优良的错误平层性能[79]。最近，他们又提出了一种列重为 3，围长至少为 8 的整数序列 QC-LDPC 码。如果序列中的数字不相同，则可以构造不存在 a≤10 和 b≤3 的（a，b）基本陷阱集的 QC-LDPC 码。这个校验矩阵的行重可以是任意的，此外，他们还给出了循环置换矩阵的下界。仿真结果表明生成的 LDPC 码具有良好的纠错性能[80]。



(a) 矩形网格；(b)矩形网格中陷阱集的表示；(c)基于立方体形式的框架。

图 10. 不同维度的矩形格表现形式

除了上面介绍的基于矩形格的构造方法，Sadeghi 和 Amirzade 等人还提出了一种基于边缘着色（Edge-Coloring Technique）的方法来表征和避免 Tanner 图中的基本陷阱集。这种边缘着色技术有助于明确得到指数矩阵的充分条件，从而避免在 Tanner 图中出现任何的基本陷阱集[81]。在这一理论的基础上，他们给出了一类基于代数的围长为 6 的 QC-LDPC 码的指数矩阵的充分条件，该码不存在 a ≤ 5 且 b ≤ 2 的（a，b）基本陷阱集，随后，他们还给出了一种基于有限域 F_q 的乘法循环子群的 QC-LDPC 码的新构造方法，其中 q-1 可以被分解为三个相对素数整数。测试结果表明，去除小尺寸的基本陷阱集有助于在错误平层获得更好的

性能曲线[82]。最近，他们又将这一理论应用在列重较大且围长大于 8 的 LDPC 码中。他们利用边缘着色技术和图论中的部分概念如彩虹环来证明消去除了“彩虹八环”之外的八环会导致小尺寸陷阱集的消除。当列重为 4（或 5）时，如果 Tanner 图中的所有八环均为彩虹环，则其不存在（a，b）基础陷阱集，其中 $b < a$，$0 \leq b \leq 8$ 且 $a < 12$（或 $0 \leq b \leq 12$ 且 $a < 15$）。这一方法极大地简化了从基于阵列的 LDPC 码的 Tanner 图中除去除彩虹环外所有 8 环的条件，为构造围长和列重都较大的 QC-LDPC 码提拱了新的选择[83]。

(m,n)	$(4,5)$	$(4,6)$	$(4,7)$	$(4,8)$	$(4,9)$	$(4,10)$	$(4,11)$
QC-LDPC	12420	26730	50652	87696	141912	217890	320760
Array-based	1400	3000	5712	9968	16272	25200	37400
(m,n)	$(5,6)$	$(5,7)$	$(5,8)$	$(5,9)$	$(5,10)$	$(5,11)$	$(5,12)$
QC-LDPC	59400	111510	191520	307800	469800	688050	974160
Array-based	7230	13902	24416	40032	62190	92510	132792

图 11. 去掉除了彩虹环外的八环的(m, n)规则 QC-LDPC 码与基于阵列的 LDPC 码的运算次数比较

此外，Naseri 等人报道了一种设计围长为 8、变量节点度数为 3 和 4 的 QC-LDPC 的新技术。在设计的代码中，通过对代码 Tanner 图中长度为 8 的环施加简单的约束，从而避免了 LDPC 码中小的陷阱集的存在。他们又推导了满足这些约束的码字长度的下界。仿真结果表明，所设计的编码与具有一定的优越性[84]。Sariduman 等人提出了一种“模拟退火”（SA）算法来构建具有良好陷阱集分布的 QC-LDPC 码。他们基于 Karimi 给出的算法粗略穷举了 QC-LDPC 码中主要的基本陷阱集，随后又根据其基本陷阱集分布计算对应 QC-LDPC 码的代价值。利用该 SA 算法构建的 QC-LDPC 码不含小的基本陷阱集，且有最佳的陷阱集分布和

较好的纠错性能[85]。Zheng 等人则基于 PEG 算法和改进的消除基本陷阱集的算法来消除基矩阵中的基本陷阱集，然后利用 Zig-Zag 法来构建循环移位矩阵。该方法可以设计码长和码率均可变的 QC-LDPC 码，且在高信噪比区域的纠错性能大大提升[86]。

改进校验矩阵构造的方法，对其对应 Tanner 图的围长进行适当地调控已经成为了优化 LDPC 码特别是 QC-LDPC 码中的陷阱集的一个重要手段。增加矩阵的围长、优化环的分布是构造矩阵过程中消除对应码中陷阱集的重要手段。我们上文中介绍的也只是其中的一部分。目前关于矩阵构造的方法还在不断被发现和提出，这对进一步认识陷阱集对译码过程中错误平层的作用将有着很大的意义。

5. 基于译码过程的陷阱集改善方法

5.1 基于 BP 译码算法

译码算法对陷阱集的形成也会产生一定的影响，这是因为在译码过程中的信息迭代方式也会影响其错误平层的性能。目前来讲，置信度传播（BP）算法是 LDPC 码中最经典的算法，这种方法通过原始信息与码字之间进行概率运算，从而使译码收敛，得到最正确的信息[87, 88]。

Landner 等人基于数据融合原理和专门的信息平均方法对 BP 传播算法进行了改进。平均的方式主要有代数平均、几何平均还有加权平均等。他们通过平均传递信息来对陷阱集中错误信息新出的速度进行有效地调控，从而抑制了变量节点信息的震荡[89]。但是值得注意的是，这种方法尽管降低了译码过程的错误平层性能，但是也大大增加了译码过程中的迭代次数。为了解决这一问题，Zhang 等人提出了一种两段译码方式，他们在第一阶段引入了一种有效地停止标准，从

而避免不必要的迭代，如果达到停止标准，将终止第一阶段解码，并开始第二阶段解码。在下一阶段，选择两种类型的变量节点，并通过操纵这两种特定的变量节点的对数似然比（LLR）来重新对之前没成功译码的码字进行译码过程。仿真结果证明这种方法能有效降低 LDPC 码的误码率，同时保持较低的译码复杂度（图 12a）[90]。类似的，Yang 等人也提出了一种两阶段译码方式，他们将传统的置信传播译码算法作为第一阶段的译码方法，结合提出的停止标准，确定了变量节点的集合。这一停止标准有利于在较早时间内监测到译码失败。并进一步跟踪导致译码失败的小陷阱集。在第二阶段的译码过程中，他们提出了一种不可靠路径搜索算法来为选中的所有变量节点打分，并在上述变量节点集合中按得分升序排序选择变量节点。通过修改排序后的集合中每个变量节点的 LLR，每次重译码试验由 BP 译码算法进行，并且对集合中选择的变量节点的 LLR 进行操纵，每次重新译码都采用第二个提前停止标准。实验表明该方法可以纠正大部分未被成功译码的码字（图 12b）[91]。

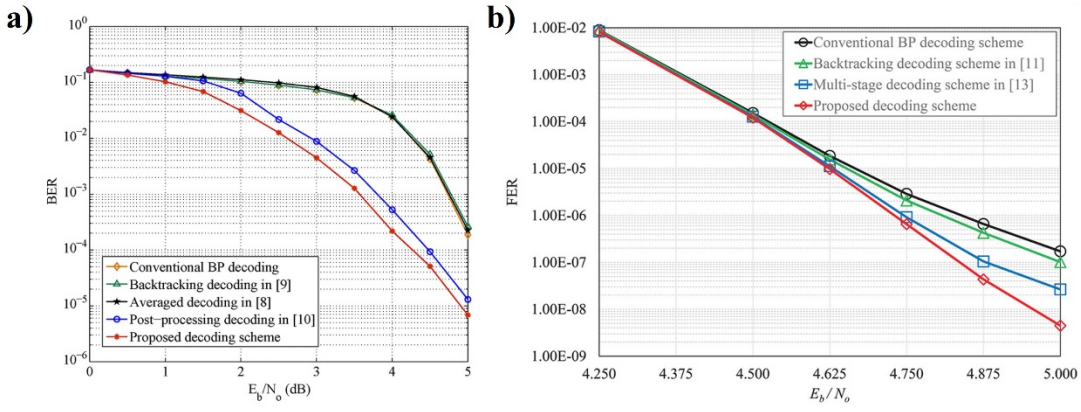


图 12. 提出的两段译码方案的性能

5.2 基于有限字母迭代译码算法

BP 译码源于一个广泛的贝叶斯推论问题，已经在 LDPC 码中有了较为广泛的应用。但是公认的是，即使在具有较强限制的拓扑结构下，精确地贝叶斯置信网络推论也较为论难[92]，因此亟需开展一种新的译码方式。有限字母表迭代译码算法（Finite Alphabet Iterative Decoders，即 FAIDs 算法）就是基于此提出的。这种算法中通过一个有限的字母表来进行消息的传递。与传统的 BP 算法相比，该算法在节点处的更新函数是一个简单的布尔对照表。此外，FAIDs 算法的复杂程度往往很小，且性能要更优异。

Planjery 等人首先提出了这一信息传递译码方法，其更新规则源于陷阱集基本论，他们发现在传递信息的比特位较小的情况下，这一规则仍可以纠正 BP 算法无法纠正的某些错误码字[93]。在此基础上，他们又以平面分区的形式给出了 FAIDs 的代替表示，从而方面枚举可能的译码数量，并提出了良好的 3 比特精度的信息传递译码器，此外，他们还表明了仅基于密度演化设计的译码器并不是对应码字的最佳方法[94]。随后，Vasic 等人继续对 FAIDs 算法进行了扩展，他们发现通过改变 FAIDs 变量节点更新的布尔对照表，可以够纠正单个 FAIDs 无法纠正的各种不同的错误码字。他们考虑了能够形成陷阱集的代码中存在的有害拓扑相关的错误码字，然后在 LDPC 码中搜索所有这样的拓扑结构，直到达到一定的权重，最后找到可以纠正的所有这类错误码字的 FAIDs 组合。结果表明，这类译码方案可以显著提高短长度 LDPC 码的纠错能力（图 13a）[95]。Cai 等人进一步对 FAIDs 进行了较为详细的分析和仿真，并利用 FAIDs 的特性提出了一种低复杂度的实现体系结构。他们表明了在二进制对称信道（BSC）的错误平层区域，具有 7 级字母的 FAIDs 也可以优于具有 6 比特的偏移最小和译码器，并为 FAIDs 开发了一种位串行 CNU 架构，只需占用 45%的面积即可达到比具有更长比特的

偏移最小和译码方案更高的吞吐量，同时在错误平层区域具有更好的性能，极大地提高了硬件利用率（图 13b）[96]。

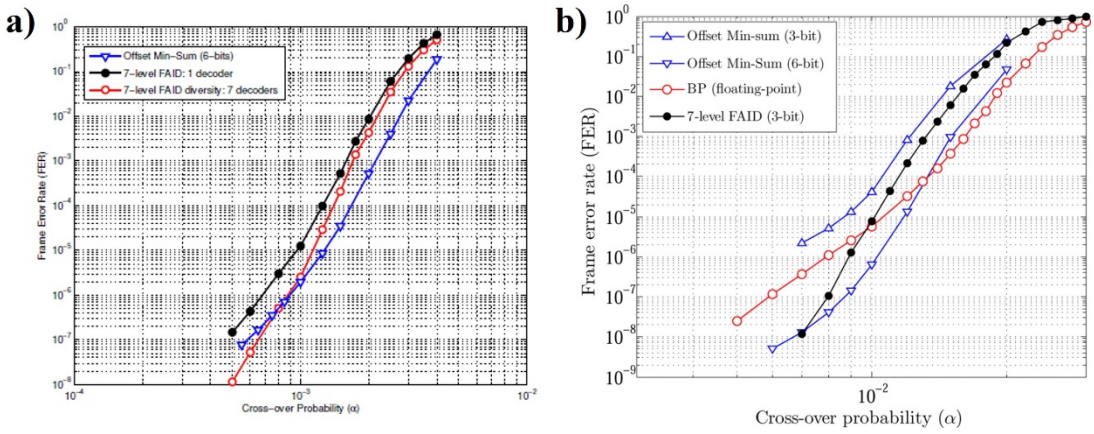


图 13. FAIDs 算法的译码性能。

最近，Xiao 等人提出了一种模型驱动的神经网络方案来设计基于 BSC 上 LDPC 码的有限字母迭代译码方案。他们训练一个循环量化神经网络（RQNN）来学习和设计 FAIDs。由于 RQNN 可以对广泛的 FAIDs 建模，因此它们能够学习任意的 FAIDs，他们依靠子图展开/收缩，在不同的错误模式上训练 RQNN 以顺序方式设计 FAIDs。这种方法生成的译码方法可以保证瀑布区性能的同时降低错误平层。他们证明了机器学习译码可以在性能上超越相同复杂性的人工译码，具有很强的战略意义[97]。

除了上述方法之外，Ouyang 等人提出了一种基于串行可靠性的迭代最小和解码（RBI-MSD）算法，同时也提出了一种基于 LLR 分布的 RBI-MSD 算法非均匀量化方法[98]。而 Zhang 等人也提出了一种（q+1）位准均匀量化方法，从而有效地增加了消息的动态范围，从而克服了传统量化方案的局限性。这两种方法均利用了 LLR 值的分布特性，具有一定的普适性[99]。

5.3 基于后处理方法

在LDPC译码器后联级其他的译码器也消除陷阱集影响并进一步改善LDPC码译码过程中错误平层性能的重要方式。这些译码器可以是不同类型的译码器如联级 BCH 译码器和 RS 码译码器等，也可以是另外一类 LDPC 码，此外，通过引入扰动来进行后处理也是一种简易的后处理方式[100-102]。

6. 总结和展望

在过去的几十年里，基于 LDPC 编解码的发展已经成为一个活跃的研究领域，并为世界上很多杰出的研究者提供了灵感。对于其译码过程出现错误平层现象的原因和改善方式的研究，也被众多研究者涉及。陷阱集作为造成这一现象的主要因素，其形成原因和改进办法也日渐得到了研究者的关注。本文对 LDPC 码中陷阱集有关的发展进行了综述。对其形成原因、搜索方法以及改进方法进行了分类和分组。目前随着 AI 时代的到来，传递的数据和信息量不断指数增加，对 LDPC 码的长度和其中陷阱集的要求也随之更为严苛。在设计易存储、高围长、有合适陷阱集分布的 LDPC 码的同时，研究下一代通信系统和存储中需要的复杂而强大的译码算法来促进陷阱集改善的算法也刻不容缓。因此，LDPC 码陷阱集相关研究工作的未来发展过程应该围绕大码长的、高信息量的码字，评估在不同技术中的使用效果，从而激发出更有前景的解决方案。

参考文献

[1] E. Ben Yacoub, Trapping and Absorbing Set Enumerators for Irregular Generalized

Low-Density Parity-Check Code Ensembles,[J] IEEE Transactions on Information 2023, (69)6, 3637-3662. doi: 10.1109/TIT.2023.3240843.

[2] Y. Fang, P. Chen, G. Cai, F. C. M. Lau, S. C. Liew and G. Han, Outage-Limit-Approaching Channel Coding for Future Wireless Communications: Root-Protograph Low-Density Parity-Check Codes,[J] IEEE Vehicular Technology Magazine 2019,14(2), 85-93. doi: 10.1109/MVT.2019.2903343.

[3] Z. Li, L. Chen, L. Zeng, S. Lin and W. H. Fong, Efficient encoding of quasi-cyclic low-density parity-check codes,[J] IEEE Transactions on Communications, 2006, 54(1), 71-81. doi: 10.1109/TCOMM.2005.861667.

[4] Lei Yang, Hui Liu and C. J. R. Shi, Code construction and FPGA implementation of a low-error-floor multi-rate low-density Parity-check code decoder, IEEE Transactions on Circuits and Systems I: Regular Papers, 2006, 53(4), 892-904. doi: 10.1109/TCSI.2005.862074.

[5] L. Costantini, B. Matuz, G. Liva, E. Paolini, and M. Chiani, Non-binary proto-graph low-density parity-check codes for space communications,[J] International Journal of Satellite Communications and Networking, 2012, 30(2), 43-51. doi: org/10.1002/sat.1004

[6] A. Bastug and B. Sankur, Improving the payload of watermarking channels via LDPC coding, IEEE Signal Processing Letters, 2004 11(2) 90-92, doi: 10.1109/LSP.2003.819350.

[7] T. Richardson and S. Kudekar, Design of Low-Density Parity Check Codes for 5G New Radio, IEEE Communications Magazine, 2018 56(3) 28-34. doi:

10.1109/MCOM.2018.1700839.

[8] R. Gallager, Low-density parity-check codes, IRE Transactions on Information, 1962, 8(1) 21-28. doi: 10.1109/TIT.1962.1057683.

[9] R. Tanner, A recursive approach to low complexity codes, in IEEE Transactions on Information Theory, 1981 27(5) 533-547. doi: 10.1109/TIT.1981.1056404.

[10] D. J. C. MacKay and R. M. Neal, Near Shannon limit performance of low density parity check codes, Electronics Letters,1996,32(18), 1645–1646.

[11] D. J. C. MacKay, Good error-correcting codes based on very sparse matrices, IEEE Transactions on Information 1999, 45(2), 399-431. doi: 10.1109/18.748992.

[12] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi and D. A. Spielman, Improved low-density parity-check codes using irregular graphs, IEEE Transactions on Information Theory, 2001 47(2) 585-598. doi: 10.1109/18.910576.

[13] T. J. Richardson and R. L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, IEEE Transactions on Information Theory, 2001, 47(2), 599-618. doi: 10.1109/18.910577.

[14] Liva G, Song S, Lan L, et al. Design of LDPC codes: A survey and new results[J]. Journal of Communications Software and Systems, 2006, 2(3): 191-211.

[15] Fang Y, Bi G, Guan Y L, et al. A survey on protograph LDPC codes and their applications[J]. IEEE Communications Surveys & Tutorials, 2015, 17(4): 1989-2016.

[16] Myung S, Yang K, Kim J. Quasi-cyclic LDPC codes for fast encoding[J]. IEEE Transactions on Information Theory, 2005, 51(8): 2894-2901. doi: 10.1109/TIT.2005.851753.

[17] Chen J, Dholakia A, Eleftheriou E, et al. Reduced-complexity decoding of LDPC codes[J]. IEEE transactions on communications, 2005, 53(8): 1288-1299. doi: 10.1109/TCOMM.2005.852852.

[18] Roberts M K, Anguraj P. A comparative review of recent advances in decoding algorithms for Low-Density Parity-Check (LDPC) codes and their applications[J]. Archives of Computational Methods in Engineering, 2021, 28(4): 2225-2251.

[19] Hu X Y, Eleftheriou E, Arnold D M. Regular and irregular progressive edge-growth tanner graphs[J]. IEEE transactions on information theory, 2005, 51(1): 386-398. doi: 10.1109/TIT.2004.839541.

[20] Tian T, Jones C R, Villasenor J D, et al. Selective avoidance of cycles in irregular LDPC code construction[J]. IEEE Transactions on Communications, 2004, 52(8): 1242-1247. doi: 10.1109/TCOMM.2004.833048.

[21] Lan L, Zeng L, Tai Y Y, et al. Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach[J]. IEEE Transactions on Information Theory, 2007, 53(7): 2429-2458. doi: 10.1109/TIT.2007.899516.

[22] Diao Q, Huang Q, Lin S, et al. A matrix-theoretic approach for analyzing quasi-cyclic low-density parity-check codes[J]. IEEE Transactions on Information Theory, 2012, 58(6): 4030-4048.

[23] Zhang L, Lin S, Abdel-Ghaffar K, et al. Quasi-cyclic LDPC codes on cyclic subgroups of finite fields[J]. IEEE Transactions on communications, 2011, 59(9): 2330-2336. doi: 10.1109/TCOMM.2011.060911.100208.

[24] Wang Y, Draper S C, Yedidia J S. Hierarchical and high-girth QC LDPC codes[J].

IEEE transactions on information theory, 2013, 59(7): 4553-4583.

[25] Abbasfar A, Divsalar D, Yao K. Accumulate-repeat-accumulate codes[J]. IEEE Transactions on Communications, 2007, 55(4): 692-702.

[26] Yang M, Ryan W E, Li Y. Design of efficiently encodable moderate-length high-rate irregular LDPC codes[J]. IEEE Transactions on Communications, 2004, 52(4): 564-571.

[27] Chen T Y, Vakilinia K, Divsalar D, et al. Protograph-based raptor-like LDPC codes[J]. IEEE Transactions on Communications, 2015, 63(5): 1522-1532.

[28] Tasdighi A, Banihashemi A H, Sadeghi M R. Efficient search of girth-optimal QC-LDPC codes[J]. IEEE Transactions on Information Theory, 2016, 62(4): 1552-1564.

[29] Kim I, Song H Y. Some new constructions of girth-8 QC-LDPC codes for future GNSS[J]. IEEE Communications Letters, 2021, 25(12): 3780-3784.

[30] Thi Bao Nguyen T, Nguyen Tan T, Lee H. Low-complexity high-throughput QC-LDPC decoder for 5G new radio wireless communication[J]. Electronics, 2021, 10(4): 516.

[31] Zhang G, Sun R, Wang X. Construction of girth-eight QC-LDPC codes from greatest common divisor[J]. IEEE Communications Letters, 2013, 17(2): 369-372.

[32] Sadeghi M R, Amirzade F. Analytical lower bound on the lifting degree of multiple-edge QC-LDPC codes with girth 6[J]. IEEE Communications Letters, 2018, 22(8): 1528-1531.

[33] Xiao H, Banihashemi A H. Improved progressive-edge-growth (PEG) construction of irregular LDPC codes[J]. IEEE Communications Letters, 2004, 8(12): 715-717.

[34] Farsiabi A, Banihashemi A H. Error floor analysis of LDPC column layered decoders[J]. IEEE Communications Letters, 2021, 26(3): 485-489.

[35] Wang C C, Kulkarni S R, Poor H V. Finding all small error-prone substructures in LDPC codes[J]. IEEE Transactions on Information Theory, 2009, 55(5): 1976-1999.

[36] Milenkovic O, Soljanin E, Whiting P. Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles[J]. IEEE Transactions on Information Theory, 2006, 53(1): 39-55.

[37] Richardson T. Error floors of LDPC codes[C] Proceedings of the annual Allerton conference on communication control and computing. The University; 1998, 2003, 41(3): 1426-1435.

[38] Ivkovic M, Chilappagari S K, Vasic B. Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers[J]. IEEE transactions on information theory, 2008, 54(8): 3763-3768.

[39] Han S, Oh J, Oh K, et al. Deep-learning for breaking the trapping sets in low-density parity-check codes[J]. IEEE Transactions on Communications, 2022, 70(5): 2909-2923.

[40] Dehghan A, Banihashemi A H. Hardness results on finding leafless elementary trapping sets and elementary absorbing sets of LDPC codes[J]. IEEE Transactions on Information Theory, 2019, 65(7): 4307-4315.

[41] Hashemi Y, Banihashemi A H. New characterization and efficient exhaustive search algorithm for leafless elementary trapping sets of variable-regular LDPC codes[J]. IEEE Transactions on Information Theory, 2016, 62(12): 6713-6736.

[42] Hashemi Y, Banihashemi A H. On characterization and efficient exhaustive search of elementary trapping sets of variable-regular LDPC codes[J]. IEEE Communications Letters, 2015, 19(3): 323-326.

[43] Raveendran N, Vasić B. Trapping sets of quantum LDPC codes[J]. Quantum, 2021, 5: 562.

[44] Landner S, Milenkovic O. Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes[C] 2005 International Conference on Wireless Networks, Communications and Mobile Computing. IEEE, 2005, 1: 630-635.

[45] Hashemi Y, Banihashemi A H. Lower bounds on the size of smallest elementary and non-elementary trapping sets in variable-regular LDPC codes[J]. IEEE Communications Letters, 2017, 21(9): 1905-1908.

[46] Dehghan A, Banihashemi A H. From cages to trapping sets and codewords: A technique to derive tight upper bounds on the minimum size of trapping sets and minimum distance of LDPC codes[J]. IEEE Transactions on Information Theory, 2018, 65(4): 2062-2074.

[47] Karimi M, Banihashemi A H. On characterization of elementary trapping sets of variable-regular LDPC codes[J]. IEEE transactions on information theory, 2014, 60(9): 5188-5203.

[48] Karimi B, Banihashemi A H. Construction of irregular protograph-based QC-LDPC codes with low error floor[J]. IEEE Transactions on Communications, 2020, 69(1): 3-18.

[49] Dehghan A, Banihashemi A H. Computing the Asymptotic Expected Multiplicity

of Elementary Trapping Sets (ETSs) in Random LDPC Code Ensembles[C] 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC). IEEE, 2018: 1-5.

[50] McGregor A, Milenkovic O. On the hardness of approximating stopping and trapping sets[J]. IEEE Transactions on Information Theory, 2010, 56(4): 1640-1650.

[51] Dehghan A, Banihashemi A H. Hardness results on finding leafless elementary trapping sets and elementary absorbing sets of LDPC codes[J]. IEEE Transactions on Information Theory, 2019, 65(7): 4307-4315.

[52] Wang C C, Kulkarni S R, Poor H V. Finding all small error-prone substructures in LDPC codes[J]. IEEE Transactions on Information Theory, 2009, 55(5): 1976-1999.

[53] Vasić B, Chilappagari S K, Nguyen D V, et al. Trapping set ontology[C]//2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2009: 1-7.

[54] Milenkovic O, Soljanin E, Whiting P. Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles[J]. IEEE Transactions on Information Theory, 2006, 53(1): 39-55.

[55] Falsafain H, Mousavi S R. Exhaustive enumeration of elementary trapping sets of an arbitrary Tanner graph[J]. IEEE Communications Letters, 2016, 20(9): 1713-1716.

[56] Hashemi Y, Banihashemi A H. Characterization of elementary trapping sets in irregular LDPC codes and the corresponding efficient exhaustive search algorithms[J]. IEEE Transactions on Information Theory, 2018, 64(5): 3411-3430.

[57] Hashemi Y, Banihashemi A H. Characterization and efficient search of non-

elementary trapping sets of LDPC codes with applications to stopping sets[J]. IEEE Transactions on Information Theory, 2018, 65(2): 1017-1033.

[58] Yacoub E B, Liva G. Fully Absorbing Set Enumerators for Protograph-Based LDPC Code Ensembles[J]. IEEE Communications Letters, 2022, 26(12): 2831-2835.

[59] Yacoub E B, Liva G. Trapping and Absorbing Set Enumerators for Nonbinary Protograph-Based Low-Density Parity-Check Code Ensembles[J]. IEEE Transactions on Communications, 2023, 71(4): 1847-1862.

[60] Amirzade F, Sadeghi M R. Analytical lower bounds on the size of elementary trapping sets of variable-regular LDPC codes with any girth and irregular ones with girth 8[J]. IEEE Transactions on Communications, 2018, 66(6): 2313-2321.

[61] Fossorier M P C. Quasicyclic low-density parity-check codes from circulant permutation matrices[J]. IEEE transactions on information theory, 2004, 50(8): 1788-1793. doi: 10.1109/TIT.2004.831841.

[62] Richter G, Hof A. On a construction method of irregular LDPC codes without small stopping sets[C] 2006 IEEE International Conference on Communications. IEEE, 2006, 3: 1119-1124.

[63] Vukobratovic D, Senk V. Generalized ACE constrained progressive edge-growth LDPC code design[J]. IEEE Communications Letters, 2008, 12(1): 32-34.

[64] Xiao H, Banihashemi A H. Improved progressive-edge-growth (PEG) construction of irregular LDPC codes[J]. IEEE Communications Letters, 2004, 8(12): 715-717.

[65] Khazraie S, Asvadi R, Banihashemi A H. A PEG construction of finite-length LDPC codes with low error floor[J]. IEEE Communications Letters, 2012, 16(8): 1288-

1291.

[66] Diouf M, Declercq D, Ouya S, et al. A PEG-like LDPC code design avoiding short trapping sets[C] 2015 IEEE International Symposium on Information Theory (ISIT). IEEE, 2015: 1079-1083.

[67] Zheng X, Lau F C M, Chi K T. Constructing short-length irregular LDPC codes with low error floor[J]. IEEE Transactions on Communications, 2010, 58(10): 2823-2834.

[68] Kou Y, Lin S, Fossorier M P C. Low-density parity-check codes based on finite geometries: a rediscovery and new results[J]. IEEE Transactions on Information theory, 2001, 47(7): 2711-2736.

[69] Tang H, Xu J, Kou Y, et al. On algebraic construction of Gallager and circulant low-density parity-check codes[J]. IEEE Transactions on Information Theory, 2004, 50(6): 1269-1279.

[70] Xu J, Chen L, Zeng L, et al. Construction of low-density parity-check codes by superposition[J]. IEEE transactions on communications, 2005, 53(2): 243-251.

[71] Diao Q, Tai Y Y, Lin S, et al. LDPC codes on partial geometries: construction, trapping set structure, and puncturing[J]. IEEE transactions on information theory, 2013, 59(12): 7898-7914.

[72] Kang J, Huang Q, Zhang L, et al. Quasi-cyclic LDPC codes: An algebraic construction[J]. IEEE Transactions on communications, 2010, 58(5): 1383-1396.

[73] Zhao S, Ma X. Construction of high-performance array-based non-binary LDPC codes with moderate rates[J]. IEEE Communications letters, 2015, 20(1): 13-16.

[74] Zhang L, Huang Q, Lin S, et al. Quasi-cyclic LDPC codes: an algebraic construction, rank analysis, and codes on Latin squares[J]. IEEE Transactions on Communications, 2010, 58(11): 3126-3139.

[75] Nguyen D V, Chilappagari S K, Marcellin M W, et al. On the construction of structured LDPC codes free of small trapping sets[J]. IEEE Transactions on Information Theory, 2012, 58(4): 2280-2302.

[76] Vasic B, Pedagani K, Ivkovic M. High-rate girth-eight low-density parity-check codes on rectangular integer lattices[J]. IEEE Transactions on Communications, 2004, 52(8): 1248-1252.

[77] Tao X, Liu P, Feng Z, et al. On the construction of low error floor LDPC codes on rectangular lattices[J]. IEEE Communications Letters, 2014, 18(12): 2073-2076.

[78] Tao X, Li Y, Liu Y, et al. On the construction of LDPC codes free of small trapping sets by controlling cycles[J]. IEEE Communications Letters, 2017, 22(1): 9-12.

[79] Tao X, Xin Y, Wang B, et al. Layered construction of quasi-cyclic LDPC codes[J]. IEEE Communications Letters, 2020, 24(5): 946-950.

[80] Tao X, Chen X, Wang B. On the Construction of QC-LDPC Codes Based on Integer Sequence With Low Error Floor[J]. IEEE Communications Letters, 2022, 26(10): 2267-2271.

[81] Sadeghi M R, Amirzade F. Edge-coloring technique to analyze elementary trapping sets of spatially-coupled LDPC convolutional codes[J]. IEEE Communications Letters, 2019, 24(4): 711-715.

[82] Amirzade F, Sadeghi M R, Panario D. QC-LDPC construction free of small size

elementary trapping sets based on multiplicative subgroups of a finite field[J]. Advances in Mathematics of Communications, 2020, 14(3).

[83] Amirzade F, Sadeghi M R, Panario D. QC-LDPC Codes With Large Column Weight and Free of Small Size ETSs[J]. IEEE Communications Letters, 2021, 26(3): 500-504.

[84] Naseri S, Banihashemi A H. Construction of girth-8 QC-LDPC codes free of small trapping sets[J]. IEEE Communications Letters, 2019, 23(11): 1904-1908.

[85] Sarıduman A, Pusane A E, Taşkın Z C. On the construction of regular QC-LDPC codes with low error floor[J]. IEEE Communications Letters, 2019, 24(1): 25-28.

[86] Zheng D, Yuan J, Wang H, et al. A new construction method of QC-LDPC codes with low error floor based on EETS and Zig-Zag[J]. Optoelectronics Letters, 2019, 15(4): 292-296.

[87] Casado A I V, Griot M, Wesel R D. Informed dynamic scheduling for belief-propagation decoding of LDPC codes[C] 2007 IEEE International Conference on Communications. IEEE, 2007: 932-937.

[88] Liu J, de Lamare R C. Low-latency reweighted belief propagation decoding for LDPC codes[J]. IEEE Communications Letters, 2012, 16(10): 1660-1663.

[89] Landner S, Hehn T, Milenkovic O, et al. Two methods for reducing the error-floor of LDPC codes[J]. arXiv preprint cs/0701006, 2007.

[90] Zhang X, Chen S. A two-stage decoding algorithm to lower the error-floors for LDPC codes[J]. IEEE Communications Letters, 2015, 19(4): 517-520.

[91] Yang P, Jun B, No J S, et al. A new two‐stage decoding scheme with unreliable

path search to lower the error‐floor for low‐density parity‐check codes[J]. IET Communications, 2017, 11(14): 2173-2180.

[92] Cooper G F. The computational complexity of probabilistic inference using Bayesian belief networks[J]. Artificial intelligence, 1990, 42(2-3): 393-405.

[93] Planjery S K, Declercq D, Chilappagari S K, et al. Multilevel decoders surpassing belief propagation on the binary symmetric channel[C] 2010 IEEE International Symposium on Information Theory. IEEE, 2010: 769-773.

[94] Planjery S K, Declercq D, Danjean L, et al. Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders[J]. Electronics Letters, 2011, 47(16): 919-921.

[95] Vasic B, Declercq D, Planjery S K. Decoder diversity architectures for finite alphabet iterative decoders[C] 2014 48th Asilomar Conference on Signals, Systems and Computers. IEEE, 2014: 131-135.

[96] Cai F, Zhang X, Declercq D, et al. Finite alphabet iterative decoders for LDPC codes: Optimization, architecture and analysis[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2014, 61(5): 1366-1375.

[97] Xiao X, Raveendran N, Vasić B, et al. FAID diversity via neural networks[C] 2021 11th International Symposium on Topics in Coding (ISTC). IEEE, 2021: 1-5.

[98] Ouyang S, Han G, Fang Y, et al. LLR-distribution-based non-uniform quantization for RBI-MSD algorithm in MLC flash memory[J]. IEEE Communications Letters, 2017, 22(1): 45-48.

[99] Zhang X, Siegel P H. Quantized iterative message passing decoders with low error

floor for LDPC codes[J]. IEEE Transactions on Communications, 2013, 62(1): 1-14.

[100] Spourlis G, Tsatsaragkos I, Kanistras N, et al. Error floor compensation for LDPC codes using concatenated schemes[C]//2012 IEEE Workshop on Signal Processing Systems. IEEE, 2012: 155-160.

[101] Shieh S L. Concatenated BCH and LDPC coding scheme with iterative decoding algorithm for flash memory[J]. IEEE communications letters, 2015, 19(3): 327-330.

[102] Lechner G, Sayir J. On the convergence of log-likelihood values in iterative decoding[C] Mini-Workshop on Topics in Information Theory. 2002: 12-13.

基于振动传感器的预测性维护系统安全优化

黄道旗[①] 张华君[*①] 徐守坤[*①] 庄丽华[①]

[①](常州大学计算机与人工智能学院 常州 213164)

摘 要：针对振动传感器在信息采集阶段受到人为或自然环境的干扰问题，本文提出四种保护措施以及一种集成式预检测诊断方案。本方案是在原有的基础上改进了相似性计算方法，通过采用多维度距离计算振动传感器连续采集数据的相似度，并与阈值比较，以提高预测性维护系统的可靠性。同时，其它保护措施旨在有效地为预测性维护系统提供预警，使预测性维护系统能够对被干扰的数据进行二次诊断，从而提高故障检测的准确性。并且本文对所有措施和方案进行安全分析，证明其具备安全性。通过两种场景的实验下，对比了使用和未使用集成式预检测诊断方案的情况，结果表明方案能够显著提高故障检测的准确性。本文也与未嵌入该方案的情况进行多方面比较，结果表明本方案的有效性和可行性。

关键词：设备安全；预测性维护；相似性检测

Security Optimization of Predictive Maintenance Systems

HUANG Daoqi[①] ZHANG Huajun[*①] XU Shoukun[*①] ZHUANG Lihua[①]

[①]（School of Computer Science and Artificial Intelligence, Changzhou University, Jiangsu 213164, China）

Abstract: The article presents four protection measures and an integrated pre-detection diagnostic approach to address the issue of interference in the information acquisition phase of vibration sensors caused by human or natural environmental factors. This approach improves the similarity calculation method by using multidimensional distance similarity to calculate the continuous data collected by vibration sensors and compare it with a threshold, thereby enhancing the reliability of the predictive maintenance system. Additionally, other protection measures aim to effectively provide warnings to the predictive maintenance system, enabling it to perform secondary diagnosis on the interfered data and improve the accuracy of fault detection. Furthermore, the paper conducts a safety analysis of all measures and approaches, demonstrating their safety. Through experiments conducted in two different scenarios, a comparison is made between the use and absence of the integrated pre-detection diagnostic approach. The results indicate a significant improvement in fault detection accuracy with the implementation of the proposed approach. The paper also presents a comprehensive comparison with scenarios where the approach is not embedded, demonstrating the effectiveness and feasibility of the proposed approach.

Key words: Equipment Security; Predictive Maintenance Systems; Similarity Detection

1 引言

近年来，随着物联网、人工智能和大数据等相关技术的迅速发展，预测性维护系统在工业设备的故障检测和预测性维护方面取得了显著进展，可以准确地分析和预测设备故障和维护需求，并为企业提供有效的维护建议和计划。预测性维护系统[1–3]的技术包括数据采集与处理、状态监测、健康评估与剩余使用寿命预测及维修决策等模块组成。其中，数据采集的核心组件是感知层的振动传感器，但这些传感器容易受到现实环境的恶意攻击，从而威胁整个系统的安全性。

根据国际自动化学会的数据显示，全球制造商每年因设备停机所遭受的损失高达 6470 亿美元[4]。然而，我国制造行业中仅有 16%的企业实施预测性维护，这表明预测性维护潜在市场巨大。已知石化企业作为预测性维护的主要应用领域之一，而且石化企业设备设施规模庞大，生产连续性强且易燃易爆。并且计划外停工是造成油气公司亏损的主要原因之一，在人为或自然环境情况下造成的计划外停机，都会对石化行业的安全生产和经济效益都会带来重大影响。每年仅 3.65 天的计划外停机时间就可能造成约 503.7 万美元的损失[5]。因此，预测性维护对于石化行业的安全生产至关重要。近年来的研究中，学者们提出了一些改进预测性维护系统的方法。在 2021 年, 周章宁等人[6]提出基于深度学习的石油钻机微电网故障预测与维护研究，在 2022 年，Alsuwian 等人[7]提出了一种先进的主动容错控制系统，通过故障检测、隔离单元和虚拟冗余值的应用，能够在传感器故障情况下保持系统的稳定性。这些研究说明预测性维护系统在国内外的石化行业迈向新的一步。

然而曾在 2010 年，美国对伊朗投放“震网”病毒，其毁坏了伊朗近 1/5 的离心机，感染了 20 多万台计算机，并使得伊朗核计划倒退了两年。在 2015 年，乌克兰电网因遭受“BlackEnergy”（黑暗力量）网络入侵而

收稿日期：2023-05-30；修回日期：2023-07-08

*通信作者：张华君(zhang.huajun@cczu.edu.cn) 徐守坤(xsk@cczu.edu.cn)

造成的大规模停电事故。在 2020 年，工业物联网芯片制造商研华科技遭到 Conti 勒索软件攻击，导致数据被窃取。以上事件表明了工业控制系统的安全性受到严峻的挑战。

另外，现实生活中的各类传感器也面临着各种人为或环境影响，导致采集的数据不准确。这是因为大多数传感器并没有设置或加装对干扰源防护，而且传感器的生产过程中没有考虑这种安全隐患。在处理传感器采集的数据，通过机器学习、深度学习和退化模型等可以用于建立对正常状况的模型，但是对于异常状况的样本，可能存在样本不足或无法准确建模。

针对上述问题，Attoui 等人[8]提出一种不需要任何机器学习算法的程序，是一种新型的基于信号的快速自动故障诊断程序，通过测量收集的数据与少量标记参考信号之间的频域相似性判断故障；Sulaiman 等人[9] 设计出一种基于有限标记数据的滚动轴承预测性维护工具（SB-PdM），其通过连续地测量所生成的振动段与所标记的参考振动段之间的特征空间中的相似性来实现对过程完整性的监测。还有其他学者相继提出的计算测量收集的数据与标记参考信号相似性的方法[10–13]。

在上述的研究的启发下，本文针对基于振动传感器的预测性维护系统在人为或环境的干扰下提出一系列相应的保护措施和集成式预检测诊断方案。首先通过物理层面隔离振动传感器，减少干扰信号对传感器的影响。其次，利用短时间内连续采集到的振动信号数据，采用信号相似检测法作为预测性维护系统的预过滤环节，可以有效识别并剔除干扰信号。此外，考虑增加额外的振动传感器从多位置检测设备是否存在潜在故障。最后，引入随机关闭振动传感器的策略，增加攻击者的难度和成本，提高系统的安全性。本文的主要贡献如下：

(1)提出的这几种措施容易实现且可以单独使用，还可以组合使用，提高系统的安全性。

(2)提出的集成式预检测方案可以嵌入其他模型的预测性维护系统，可以通过集成不同模型的数据分析和判断能力，提高对异常和干扰的检测准确性，提高系统对安全攻击的抵御能力。

2 相关工作

2.1 攻击传感器的策略

目前攻击传感器的策略主要可分为物理域和数字域两大类。物理域的攻击策略与数字域的攻击策略不同，物理域的攻击策略主要侧重于利用实际物体的特性和外部环境条件，以产生对抗性干扰。而数字域的攻击策略主要利用软件漏洞、网络协议弱点或恶意软件等手段来攻击传感器。

目前最容易实现对传感器的物理域攻击是声波攻击。Trippel 等人[14]和 Tu 等人[15]通过调整声音信号的频率、幅度和相位控制混叠结果，可以精准控制实现了对 MEMS（Micro-Electro-Mechanical System，微机电系统）加速度计和陀螺仪的输出。Roy 等人[16]曾提到可以利用麦克风中的硬件非线性，使得听不见的超声波信号注入到 MEMS 麦克风，从而控制电子设备。

除了声波攻击外，还有其他类型传感器可以被其他方式攻击。例如，在 2015 年的黑帽欧洲大会上，Petit 等人[17]通过使用廉价的商用硬件成功地攻击摄像机和激光雷达系统，展示了 PEAS（Performance, Environment, Actuators and Sensors，性能，环境，执行器和传感器）所构成的潜在危险。Yan 等人[18]证明，特斯拉 Model S 上的传感器可能会受到攻击，会导致自动驾驶系统无法检测到某些对象。

2.2 预测性维护系统的方法

目前在预测性维护系统中，主要采用以下两种主流方法：

(1)退化模型：此类方法利用退化模型[19,20](如伽马退化过程或维纳退化过程)来描述设备的退化过程进行寿命预测，基于预测结果再采用优化方法寻找最优的设备维修方案。其中，在文献[21]采用非线性退化过程，即基于 KWP(Kernel function-Wiener Process，核函数-Wiener 过程)退化模型，同时在贝叶斯框架下，推导出了参数估计的解析表达式，并构建了衡量轧辊可工作转动量的健康指标。此外，其他数学模型[22]也可以预测任何机器的运行，但预测结果取决于从这些机器中收集的数据的数量和准确性。

(2)机器学习或深度学习：利用机器学习或深度学习方法实现设备寿命预测。在文献[23] 中利用 LSTM（Long Short-Term Memory，长短期记忆神经网络）进行寿命预测，获得了比周期性预测策略和理想预测维护策略更好的结果。在文献[24]和文献[25]表明，基于 LSTM 的网络可以从传感器信号中提取有效特征，并在刀具磨损预测任务上进行了验证。也发现除了时间上的信息外，不同维度传感器数据之间包含的空间关系和数据中的噪音会使得仅使用 LSTM 变得不够鲁棒，所以选择采取使用 CNN（Convolutional Neural Network，卷积神经网络）

提取局部特征，然后进行寿命预测。除了前面提到的基于 LSTM 和 CNN 的方法，还有基于其它机器学习和深度学习的方法[26–30]构建预测性维护系统。

除了上述的两个主流的方法，还有先后出现其他探索性方法，例如袁烨等人[31]分别从模型方法、数据方法以及融合方法分析了设备的寿命预测技术和维护决策理论；Berghout 等人[32] 用知识驱动的方法构建知识图谱预测滚动承剩余使用寿命；王岩等人[33] 出了面向采煤机智能运维的知识图谱构建方法，实现实时、快速地进行预测性维护目的。

2.3 预测性维护系统的局限性

振动传感器在预测性维护系统中起着关键作用，也是目前广泛应用的报警检测传感器之一，它能够收集到设备最关键的振动数据。通过采集振动数据，可以发现设备的潜在故障，从而降低生产过程中的安全风险。在实际的工业场景中，除了旋转设备外，还存在多种可能产生振动的干扰源，如设备的周边环境、其他机器设备的振动以及人为的因素等。这些干扰振动源可能会导致检测结果波动，影响预测性维护系统的准确性。

目前主流方法有基于深度学习的预测性维护系统，尽管深度学习已被成功应用于早期故障特征的自动提取和识别，但该系统在应对干扰振动源的场景下仍存在局限性，如局部特征提取可能导致整体时序信息丢失。这些不足之处主要归因于卷积神经网络在处理时序数据方面的固有限制。

3 系统设定与攻击假设

本节将介绍一个通用且灵活的系统设定，适用于大多数工业预测性维护系统。同时，还提出了一个较为全面的攻击假设，包括多种攻击类型和攻击者的能力。

3.1 系统设定

本文研究的系统架构中，如图 1 所示，振动传感器作为一个单自由度振荡系统，用来采集工业设备的振动加速度数据。该传感器包含弹簧、阻尼器和惯性质量块，安装在工厂中使用的电机设备上。传感器在电机设备上采集振动加速度数据，数据通过振动传感器内置的蓝牙模块传输至手机。手机作为一个智能终端，接收振动传感器发送的数据，并通过网络将数据上传至云端。云端作为数据处理和存储平台，负责接收和处理来自多个手机上传的数据，并运行预测性维护系统进行数据分析和故障预测。当振动加速度数据超过预设阈值时，系统会发出预警信号并通知相关人员进行维护或修理。同时，系统记录设备的振动数据，为将来的故障预测和维护提供参考。

在实际的工业环境中，这种预测性维护系统具有很高的可扩展性。通过在单个设备安装多个振动传感器，可以构建智能集群接入点，以满足不同设备位置和连接类型的需求。

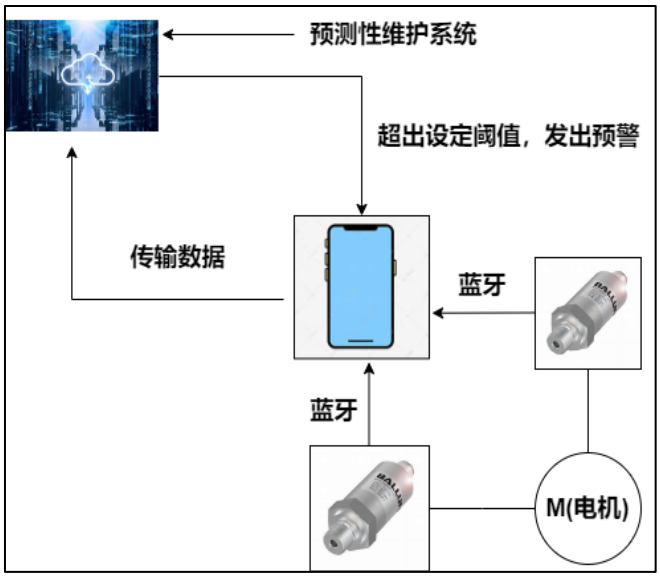


图 1 系统模型架构图

3.2 攻击假设

本文提出以下假设来评估基于振动传感器的预测性维护系统的安全性：

(1)攻击者的能力：攻击者具备制造干扰或振动源的能力。攻击者可以利用手机产生的振动对小型设备干扰；或者利用环境中的刚性材料对大型设备干扰。因为工业设备通常由刚性材料构成，刚性材料之间的碰撞可能导致剧烈的应力和形变，进而在短时间内改变设备的振动特性，对传感器数据造成影响。

(2)攻击者的访问级别：攻击者可能以不同方式接近振动传感器，如恶意员工、客人或维护人员可以接近电机设备并将振动源放置在传感器附近。虽然传感器只允许授权人员使用，但攻击者可以伪装成客人或维护人员，短暂地在传感器附近进行一次性访问，并放置干扰源，造成影响。

(3)攻击工具和成本：对于小型设备，一部售价几百元的手机，能够在特定情况下可产生足够的振动。反之大型设备，攻击者可使用低成本的刚性材料进行干扰。

(4)攻击结果：首先是误报警告（误导性预测），攻击者可能会制造异常干扰来欺骗预测性维护系统，使其误报警告，从而导致设备停机或不必要的维修。这会增加维修和停机时间的成本，甚至可能会导致生产中断和生产损失。再者就是数据篡改，攻击者可以修改传感器收集的数据，以欺骗预测性维护系统，使其不能准确预测设备的健康状况。这可能会导致设备在未经充分维修或更换之前失效，从而增加了设备维修和更换的成本。最后就是最初的影响不明显，但是长此以往会让预测性维护系统的预测精度会有所下降。

基于上述的系统设定与攻击假设，本文设计出相应的保护措施和集成式预检测方案，以确保基于振动传感器的预测性维护系统的安全性和可靠性。同时，预测性维护系统的初始状态是可信的，设备上运行的实时传感器采集模块也是可信，并假设工业生产或实验室的所有流程是安全可靠的。

4 集成式预检测诊断方案

4.1 单独保护措施

4.1.1 物理隔离

物理隔离是其中一种安全措施，旨在确保设备之间的隔离程度，以保证设备的正常运行。物理隔离的效果采用公式(1)表示：

$$I = \Phi\left(A_i, A_d, A_c\right) \tag{1}$$

其中 I 表示设备之间的物理隔离程度，这是一个综合性的安全指标，用于评估设备之间信息、数据和计算过程的隔离性。

Φ 表示一种将所有的物理隔离参数进行综合的函数，具体形式可取决于这四个参数的重要性。

A_i 是设备之间的信息流隔离，是指设备之间的信息传输是否能够被其他设备或系统获取。A_d 是设备之间的数据隔离，即确保从一个设备上收集的数据不会被其他设备获取，保证数据隔离。 A_c 是设备之间的控制隔离，确保设备的控制命令不会被未授权的设备或人员获取，防止潜在的安全威胁。

4.1.2 相似性检测方法

相似性检测法是一种基于振动信号频谱特征分析的机械故障诊断方法。其核心原理是通过计算其中一个传感器在一段时间内多组连续的振动加速度数据之间的相似度指标来识别机械设备是否存在潜在故障。

首先采集 N 组振动加速度信号 x_i，其中 $i = 1,2,3,\cdots N$。对每个初始振动信号 x_i，通过信号处理技术（如小波分析、时频分析等）以获得优化后的信号 v_i。然后需要先计算每组数据之间时域距离、频域距离、皮尔逊相关系数和综合距离，最后计算相似度，其计算公式如(2)下：

$$\mathrm{sim}\left(v_i, v_j\right) = \frac{1}{1 + \sqrt{\left(t_w \cdot \mathrm{distance1}\right)^2 + \left(\left(1 - t_w\right) \cdot \left(1 - \mathrm{freq}_{\mathrm{pearson}}\right)\right)^2 + \mathrm{freq}_{\mathrm{error}}}} \tag{2}$$

其中，$\mathrm{distance1}$ 表示两组信号数据的低频信号的欧氏距离，$\mathrm{freq}_{\mathrm{pearson}}$ 表示两组信号数据的振幅值间的皮尔逊相关系数，$\mathrm{freq}_{\mathrm{error}}$ 表示两组信号数据在频域上的误差，t_w 表示时间权重，用于平衡时域距离和频域距离

的影响。最终的 $\mathrm{sim}\left(v_i,v_j\right)$表示两组信号数据的振动加速度信号的相似度。 v_i和 v_j 分别表示第 i和 j组数据的振动加速度向量，其中振动加速度向量里有 m个数据。如下所示：

$$v_i=\left[a_{i1},a_{i2},\cdots,a_{im}\right]^T$$

$$v_j=\left[a_{j1},a_{j2},\cdots,a_{jm}\right]^T$$

在计算完每组数据之间的相似度指标后，将这些单独指标 $\mathrm{sim}\left(v_i,v_j\right)$结合为相似度矩阵 S，这是因为相似度矩阵 S相比于单个相似度指标 $\mathrm{sim}\left(v_i,v_j\right)$，包含了更多组数据之间的相似度信息，如公式(3)表示，其中第 i行第 j列的元素表示第 i组数据和第 j组数据之间的相似度指标：

$$S=\begin{bmatrix}\mathrm{sim}(v_1,v_1) & \mathrm{sim}(v_1,v_2) & \cdots & \mathrm{sim}(v_1,v_n)\\ \mathrm{sim}(v_2,v_1) & \mathrm{sim}(v_2,v_2) & \cdots & \mathrm{sim}(v_2,v_n)\\ \vdots & \vdots & \ddots & \vdots\\ \mathrm{sim}(v_n,v_1) & \mathrm{sim}(v_n,v_2) & \cdots & \mathrm{sim}(v_n,v_n)\end{bmatrix} \tag{3}$$

矩阵 S的平均值用 $\overline{S}$表示，本文需设定相似度阈值 T（这个阈值是需要每经过一段时间后选取连续采集的数据进行计算更新），根据相似度矩阵 S的值范围在 0 到 1 之间，其中 1 表示完全相似，0 表示完全相反，但在这里小于阈值 T，就可以认为机械设备存在振动信号异常或者振动信号变化趋势异常，从而发送给后端让其注意此时段数据问题。

此阶段可以采用算法 1 表示：

算法 1：集成式预检测诊断算法

输入：采集的振动加速度数据 x_i,相似度阈值 T

输出：异常警报或正常状态

1. 初始化相似度矩阵 S为一个 $n\times n$的零矩阵
2. 对采集的数据 (x_i,x_j)优化
3. 得到优化后的信号 v_i,v_j
4. 分别计算优化后的信号之间的时域距离、频域距离、皮尔逊相关系数和综合距离
5. 计算 $\mathrm{sim}\left(v_i,v_j\right)$
6. S[i][j] 和 S[j][i] 为 $\mathrm{sim}\left(v_i,v_j\right)$的值
7. 返回 $\overline{S}$
8. 如果 $\overline{S}\geq T$
9. 处于正常状态
10. 如果 $\overline{S}<T$
11. 需异常警报

4.1.3 添加振动传感器

上一节相似性检测方法是在一段时间连续采集，而本小节措施考虑的是在工业设备周围增加额外的振动传感器，这样会使组合方案有时间和空间两个维度对设备或系统振动状况进行监测。

在设备外部周围增加 m个振动传感器，其中编号为 1 振动传感器在某个时间点内采集的 n组数据向量为 $V_1\in R^m$，其中 $V_1=\left[w_{11},w_{12},\cdots,w_{1n}\right]$。而其他传感器在某个时间点内采集的 n组数据向量表示 $V_i=\left[w_{i1},w_{i2},\cdots,w_{in}\right]$，$(i=1,2\cdots m)$。

为了提高故障预测和检测的准确性，各个传感器将各组数据进行相似性检测方法进行检测，计算出各个传感器在某一时间段内的相似矩阵平均值 $S_i\ (i=1,2\cdots m)$，然后使用公式（4）计算出 $S_{总}$。

$$S_{总}=\frac{S_1+S_2+\cdots+S_m}{\mathrm{m}} \tag{4}$$

然后将 $S_{总}$ 与前面规定的阈值 T 进行比较，如果 $S_{总} \geq T$，认为是处于正常状态，反之认为是处于故障状态。

4.1.4 随机关闭传感器

目前的工业设备对于干扰信号的监测方法主要基于信号分析和深度学习技术。攻击者的目的是干扰数据，使数据不准确，本小节措施重点在单个或多个传感器的开关状态上，通过控制开关状态，使攻击者无法准确干扰传感器，从而提高设备的安全性和稳定性。

假设有 n（$n \geq 2$）个传感器安装在工业设备，第 i 个传感器的开关状态用二进制变量 $s_i \in (0,1)$ 表示，其中 $s_i = 1$ 表示第 i 个传感器是开启状态，$s_i = 0$ 表示关闭状态。则 n 个传感器的状态向量 $s = [s_1, s_2, \cdots, s_n]^T$。

在设备正常运行阶段，随机选择 m（$0 \leq m \leq n$）个传感器进行开关，将这 m 个传感器的状态设为 1，其余传感器状态设为 0，对 m 个传感器进行编号，编号表示为 $I = (i_1, i_2, \cdots, i_m)$，则有 $\forall i \in I, s_i = 1, \forall i \notin I, s_i = 0$。

此时，采集到的振动数据向量为 $x = [x_1, x_2, \cdots, x_m]^T$，其中 x_i 表示第 i 个开启状态的传感器采集到的振动信号。因此，我们可以定义一个信号异常指标 E，用于判断是否存在攻击信号，即：

$$\mathrm{E} = \frac{\|x - \mu\|}{\sigma} \tag{5}$$

其中 $\mu = [\mu_1, \mu_2, \cdots, \mu_m]^T$ 表示开启状态的 m 个传感器采集到的平均振动信号向量，σ 表示开启状态的 m 个传感器对应之前正常运行状态下采集到的振动信号向量的标准差，如公式（6）所示：

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - u_i)^2} \tag{6}$$

如果 E 超过了预设的标准差阈值 K（正常状态下的所有传感器采集到的振动信号向量的平均标准差），则也会判断为存在干扰信号，此时需要采取相应的措施来保证工业设备的稳定运行。

此阶段可以采用算法 2 表示：

算法 2：随机选择传感器

输入：采集的振动加速度数据

输出：矩阵 S 的值

1. 传感器数量 ← 计算电机传感器的数量
2. 随机选择数量 ← 在 1 到传感器数量之间随机选择一个整数
3. 随机选择索引 ← 从范围为传感器数量的序列中随机抽取随机选择数量个索引
4. 从 i-=0 到对于所有选项的的传感器
5. 采集对应传感器的数据
6. 对数据与前面正常数据计算出标准差
7. 与标准差进行比较，大于等于属于正常状态，反之，存在潜在故障。

4.2 组合方案

工业设备在正常运转期间经常面临着人为干扰或自然环境的影响，这会严重影响设备的稳定运行和使用寿命。本节将前面几种保护措施进行组合，形成一个集成式预检测诊断方案。流程图如图 2 所示。

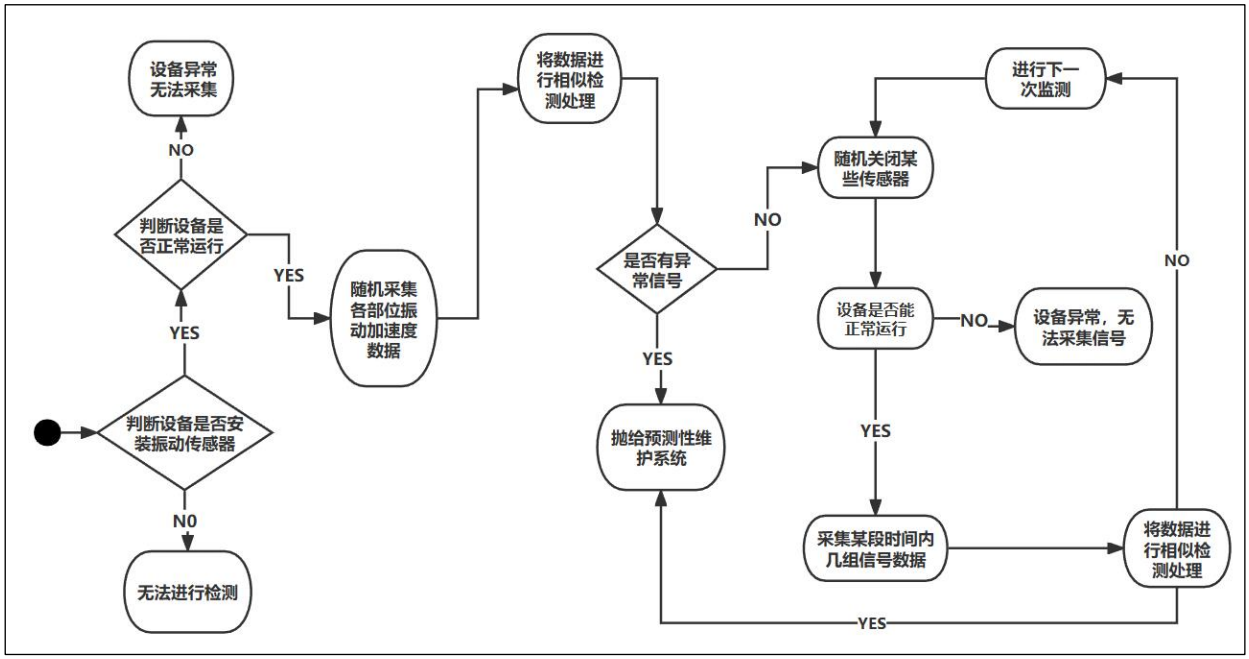


图 2 组合方案预检测诊断流程图

首先，为了精确监测设备的一些关键位置的振动情况，会在关键部位安装多个振动传感器。可以定义传感器在设备在运转情况下采集的振动信号为 x，其振动加速度信号在不同部位的取值可以表示为 $x=[x_1,x_2,\cdots x_n]$，其中 n 表示传感器的数量。

其次，为防止恶意人员对设备造成的人为干扰。用隔离距离 d 表示物理隔离程度。在设备运行阶段，通过相似性检测方法，比较各个测点连续几组振动加速度信号的相似程度，以判断设备是否受到了干扰信号。同时定义阈值 T 用于评估传感器采集的振动信号是否处于正常范围。同时还采用 4.1.4 所提的策略，判断是否存在攻击信号，用公式（5）计算的信号异常指标 E 判断。如果 E 超过了预设的 K，则判断为存在攻击信号。此时，需要采取相应的措施来保证电机设备的稳定运行，例如立即停机或通知相关人员进行检修。

最后，在采集完所有数据后，再次使用公式（2）计算完每组振动数据之间的相似度，然后使用公式（3）表示相似度矩阵 S，计算矩阵 S 的平均值，如果 $\overline{S}$ 的值接近于 1，则说明两个部位的振动加速度信号相似，数据准确性较高。

需要注意的是，为了避免环境变化对传感器响应的影响，应定期重新计算 μ 和 σ 的值，并更新阈值。在检测到异常信号时，可以将其直接传递给预测性维护系统进行处理。

集成式预检测方案可以采用算法 3 表示：

算法 3：集成式预检测方案

输入：采集的振动加速度数据

输出：矩阵 S 的值

1. 传感器数量 ← m
2. 传感器 ← 初始化包含传感器数量个空元素的列表
3. 将所有传感器初始化
4. 电机 ← 初始化一个电机设备(传感器，假设未受攻击)
5. 正常运行时间 ← t_{normal} 和攻击持续时间 ← t_{attack}
6. 使用算法 2 随机选择 n 个传感器
7. 使用算法 1 计算选中的传感器的振动相似度
8. 相似度矩阵←
9. 计算并输出矩阵平均值 $\overline{S}$
10. 将 $\overline{S}$ 与相似度阈值 T 进行比较，低于阈值，将此数据传输给预测性维护系统

5 安全分析

5.1 措施的安全性

5.1.1 相似性检测方法

本小节通过振动信号处理与相似性检测方法，使原有数据更加具有可靠性，特别是那些受到干扰的数据。此措施的安全性的考量来自以下方面：信噪比优化、故障检测性能、抗干扰以及系统的安全性。

（1）信噪比优化

信噪比是信号功率（S）与噪声功率（N）的比值，设只经过预测性维护系统信号处理的信噪比为SNR_1，经过本方案信号处理后的信噪比为SNR_2。信噪比可以用以下公式（7）表示：

$$\mathrm{SNR}=10\times\lg\left(\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}\right) \tag{7}$$

其中P_{signal}为信号功率，P_{noise}为噪声功率。对原始振动信号进行信号处理后，会有效地降低了噪声对信号的干扰。其中信号处理与预测性维护系统的信号处理方法不太相同，二者都可以有效降低噪声对信号的影响，从而提高信噪比。

但这两种处理方法在处理流程的先后顺序上存在区别。在相似性检测法中，优先对原始振动信号进行了优化处理，然后再将处理后的信号输入到预测性维护系统中。这种方式可以最大限度地减少预测性维护系统需要处理的噪声数据量，从而提高其处理效率。相比之下，如果直接将原始信号输入到预测性维护系统中，可能会由于噪声的干扰而影响系统的预测准确性，使得$\mathrm{SNR}_2>\mathrm{SNR}_1$。

（2）故障检测性能

使用故障检测的准确率（accuracy）和查全率（recall）来评估故障检测性能。设真实故障数为 TP，预测正确的非故障数为 TN，预测错误的故障数为 FP，未检测到的故障数为 FN。

准确率表示预测正确的比例，计算公式如下：

$$\mathrm{accuracy}=\frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \tag{8}$$

查全率表示正确检测到故障的比例，计算公式如下：

$$\mathrm{recall}=\frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{9}$$

这两个指标都在0到1之间，准确率反映了预测结果的正确比例，而查全率则表示正确检测到故障的比例。在没有干扰情况下，相似性检测方法的准确率与查全率会与预测性维护系统的相近，后面第六章实验与分析可以证明。但是在干扰情况下，因为未检测到的故障数会上升，预测性维护系统的准确率和查全率会降低。

（3）抗干扰

相似性检测法在计算相似度时采用了时域距离、频域距离、皮尔逊相关系数和综合距离，这种多维度的相似性计算方法对信号的幅度不敏感，更多地关注信号的方向或者说趋势，更能抵抗噪声和其他外部干扰。它能够捕捉微小的信号变化，对设备故障初期的微妙变化敏感，从而能够及早发现潜在故障。而基于深度学习的预测性维护系统的抗干扰能力主要是来自数据源，如果数据集中的干扰源较少，会导致抗干扰能力较低。

（4）系统安全性

那么此系统安全性可以通过信噪比优化、相似性检测方法的准确率与查全率和具有一定的抗干扰能力来评估。当$\mathrm{SNR}_2>\ \mathrm{SNR}_1$以及相似性检测方法的准确率和查全率较高时，可以认为这个措施在预测性维护系统中具有较高的安全性。

5.1.2 添加振动传感器

首先分别定义三个函数来描述信号处理、预测模型和安全性指标。设$f(v)$表示处理振动信号的函数，如卡尔曼滤波器等；$P(v)$为系统故障的概率预测；而$S(v)$表示系统安全性指标。

其中，$f(v)$表示一个处理振动信号的函数，其中$v=[v_1;v_2]$，v_1表示一个振动传感器采集的数据向量，v_2表示其他增加的传感器采集的数据向量。$P(v)$为系统故障的概率预测，其中v为振动数据向量。$S(v)$表示系统的安全性指标，其值在 0 到 1 之间，1 表示完全安全，0 表示不安全。假设$f(v)$、$P(v)$和$S(v)$可以在多项式时间内计算并能有效地检测潜在故障。

在本文中提出了一种通过添加外部振动传感器以提高工业设备振动监测安全性的策略。证明此措施的安全性，从以下方面分析：

考虑以下两种运行情况，分别是仅使用原设备的振动传感器采集的数据，即 $v=v_1$。在这种情况下，设系统的安全性指标为 $S_1=S(v_1)$。另外一个是同时使用原设备的振动传感器和外部振动传感器采集的数据，即 $v=[v_1;v_2]$。在这种情况下，设系统的安全性指标为 $S_2=S([v_1;v_2])$。

因为 $f(v)$可以有效地检测潜在故障，所以在包含更多振动信号的情况下（即 $v=[v_1;v_2]$），故障预测和检测的准确性会更高，所以得出 $P([v_1;v_2])<P(v_1)$。

那么假定系统安全性指标 $S(v)$与故障概率预测 $P(v)$呈反比关系，即 $S(v)=\dfrac{1}{1+P(v)}$。当故障概率降低时，系统安全性指标会相应地提高。那么可以推导出公式（10）和公式（11）：

$$S_2=S([v_1;v_2])=\frac{1}{1+P([v_1;v_2])} \tag{10}$$

$$S_1=S(v_1)=\frac{1}{1+P(v_1)} \tag{11}$$

由于 $P([v_1;v_2])<P(v_1)$，得出：$S_2=\dfrac{1}{1+P([v_1;v_2])}>\dfrac{1}{1+P(v_1)}=S_1$。

因此，在添加外部振动传感器的情况下，系统的安全性指标 S_2大于仅使用一台设备振动传感器的安全性指标 S_1。这表明，在电机设备周围增加外部振动传感器可以提高对设备或系统振动状况的准确监测。

5.1.3 随机关闭传感器

本方案通过随机关闭传感器降低数据不被干扰的概率。此外攻击者要想成功攻击系统，他需要完全准确地猜中被关闭的所有传感器。但是每一个传感器都有可能被随机关闭，攻击者想要准确地预测出所有的可能性，这个任务难度非常大。

由于被开关的传感器是随机选择的，攻击者需要准确猜中所有 m 个被开关的传感器，那攻击者猜中被开关的传感器的概率 P_{guess}如公式(12)所示:

$$P_{\text{guess}}=\frac{C_m^k}{C_n^m} \tag{12}$$

其中，C_n^m表示从 n 个传感器中选择 m 个传感器的组合数，C_m^k表示表示从 k 个干扰传感器中选择 m 个传感器的组合数。

攻击者成功发动攻击的条件是他准确猜中了所有被开关的传感器。因此，攻击者成功发动攻击的概率（P_{attack}）必须要满足 $P_{\text{attack}}=P_{\text{guess}}$，显然当 m 较小，且与 k 差值变大，这会使攻击者成功发动攻击的概率变低。

选择合适的 m 值将使攻击者的期望被监测传感器数量降至较低水平，从而降低被攻击的概率。因此通过随机开关传感器来监测攻击信号的方法在概率上具有一定的安全性，可以降低被攻击的概率。

5.2 组合方案的安全性

本文提出了一种将多种策略相互组合的综合方案。首先，与单一策略相比，整体方案综合了多个层面的保护策略。可以从安全角度设计一个整体数学模型，用来评估工业设备在恶意攻击和自然环境下的安全性。

定义 1：单一策略的安全性指标为 S_{single}，整体预检测方案的安全性指标为 S_{total}，$S_{\text{total}}>S_{\text{single}}$。安全指标参数包括振动信号相似度 $\text{sim}(v_i,v_j)$、物理隔离因子 P、异常信号检测率 R、响应时间 T 等多个因素，以全面反映设备的安全状况。

振动信号相似度 $\mathrm{sim}\left(v_i, v_j\right)$反映了设备各部位振动信号的相似程度，采用 4.1.2 节提出公式（2）进行计算。

其中物理隔离因子 P用公式（13）表示，其指设备与人的实际隔离距离 d和安全隔离距离 d_s之比。

$$P=\frac{d}{d_s} \tag{13}$$

异常信号检测率 R用公式（14）表示，其指在一定时间内成功检测到的异常信号数量与实际异常信号数量之比。

$$R=\frac{n_{\text{detected}}}{n_{\text{actaul}}} \tag{14}$$

响应时间 T是指从检测到异常信号到预测性维护系统处理完成所花费的时间。

证明：构建一个安全指标 S_{total}，那么安全指标 S_{total}如公式（15）表示:

$$S_{\text{total}}=q_1\times\mathrm{sim}\left(v_i, v_j\right)+q_2\times P+q_3\times R+q_4\times T \tag{15}$$

其中，q_1-q_4是各因素的权重系数，它们反映了这些因素在整体安全性评估中的相对重要性。通过调整权重系数 q_i，可以根据不同设备和场景的需求，突出某些因素在安全性评估中的重要性。

假设存在一个极端情况，即只有振动信号相似度对安全性有影响，其他因素 P、R和 T对安全性没有影响，即 $P=R=T=0$，这时用 S_{single}表示安全性指标，用公式（16）表示：

$$S_{\text{single}}=q_1\times\mathrm{sim}\left(v_i, v_j\right) \tag{16}$$

由于 q_1-q_4均为非负数，那么假设至少存在一个非零的因素对安全性有影响，那么只需要 $P>0$，那么可以得出 $S_{\text{total}}=q_1\times\mathrm{sim}\left(v_i, v_i\right)+q_2\times P>q_1\times\mathrm{sim}\left(v_i, v_i\right)=S_{\text{single}}$。

由此可得出在综合考虑多个影响设备安全性的因素时，整体预检测方案的安全性指标（ S_{total}）优于单一策略的安全性指标（ S_{single}）。此外，只要至少有一个因素（物理隔离、异常信号检测率或响应时间）对安全性评估产生积极影响（即它们的值大于 0），那么 S_{total} 将大于 S_{single}。同理可证其他情况。

6 实验及分析

6.1 效率评估

对于本文方案的时间复杂度，主要计算三个算法的时间复杂度：

（1）算法 1 的时间复杂度主要取决于计算相关系数矩阵的复杂度，对于 n个传感器，时间复杂度为 $O\left(n^2\right)$，其次是传感器振动数据的长度 m。因此该算法的总时间复杂度为 $O\left(m\times n^2\right)$。

（2）算法 2 的时间复杂度是 $O\left(k\right)$，在这个过程中，本文随机选择一些传感器进行数据采集。随机选择操作的时间复杂度是 $O\left(k\right)$，其中 k是所选传感器数量。

（3）算法 3 的时间复杂度是 $O\left(m\times n^2\right)+O\left(\text{duration}\times k\right)+O\left(k\right)$，算法 3 主要是算法 1 和 2 组合，还有执行持续时间（duration）。执行持续时间与切换过程有关，在最坏情况下，每次迭代都会切换所有传感器，因此时间复杂度为 $O\left(\text{duration}\times k\right)$。

所以，整体方案的时间复杂度是算法 3 的时间复杂度，其主要取决于 n(传感器数量)、duration(持续时间)和 k（切换次数)。算法 1 和算法 2 的时间复杂度相对较高，而算法 3 的时间复杂度相对较低。尽管算法 3 中的时间复杂度可能随着切换次数和持续时间的增加而增加，但在实际操作中仍然是可接受的。考虑到算法 2 中的随机选择和数据处理策略，以及算法 1 中计算相关系数矩阵的时间复杂度实际较低，该方案在运行效率上是合理的。

6.2 实验设置

本文在实验室的试验台所验证集成式预检测诊断方案性能。

(1)设备准备：设备是使用旋转设备振动故障模拟试验台，传感器是使用的频率探索智能科技江苏有限公司提供的振动传感器，并与试验台的设备进行连接。在试验台安装传感器和联轴器，同时确保试验台的设备和传感器工作正常。

(2)实验设定：设定伺服电机转速为 1200rpm，并让设备运转 1-2 小时，以模拟设备长时间运转，使预测性维护系统的设备分数达到相对稳定。

(3)数据采集：使用指定的软件连接振动传感器，并在采集数据界面设置 2K 采样频率和 Z 轴，并进行实时数据采集。

(4)数据下载和分析：从预测性维护系统的官方网站，下载所采集数据的原始波形文件。然后将数据进行相似性检测方法进行检测。

(5)非对抗场景的实验条件是在设备有一定磨损情况下，使用提供的传感器采集 67 组数据，然后计算出 65 组相似度。对抗场景下的实验条件也是设备存在一定磨损的情况下注入了随机干扰，并继续使用同一个传感器采集 77 组数据，然后计算出 75 组相似度。其中实验中所用阈值是经过几百组未被干扰的数据的平均值。

6.3 非对抗场景中的组合方案性能

在非对抗场景下，本研究提出的整集成式预检测诊断方案表现出较高的性能。通过利用多传感器数据互相验证和随机选择传感器策略，使本方案提高了系统的鲁棒性。

其次是高准确率和查全率，本文所采用的方案经过实验验证，如图 3 所示，其中有 10 个连续三组数据的相似度低于阈值，但是根据设备的预测性维护系统的评判，仅有 1 组数据是存在潜在故障（这组数据是第 22 组）。采用 5.1.1 小节的准确率和查全率评估，计算得出准确率为 100%和查全率为 100%。尽管具有一定的误报情况，但依然能够在非对抗场景下将潜在故障预警出来。

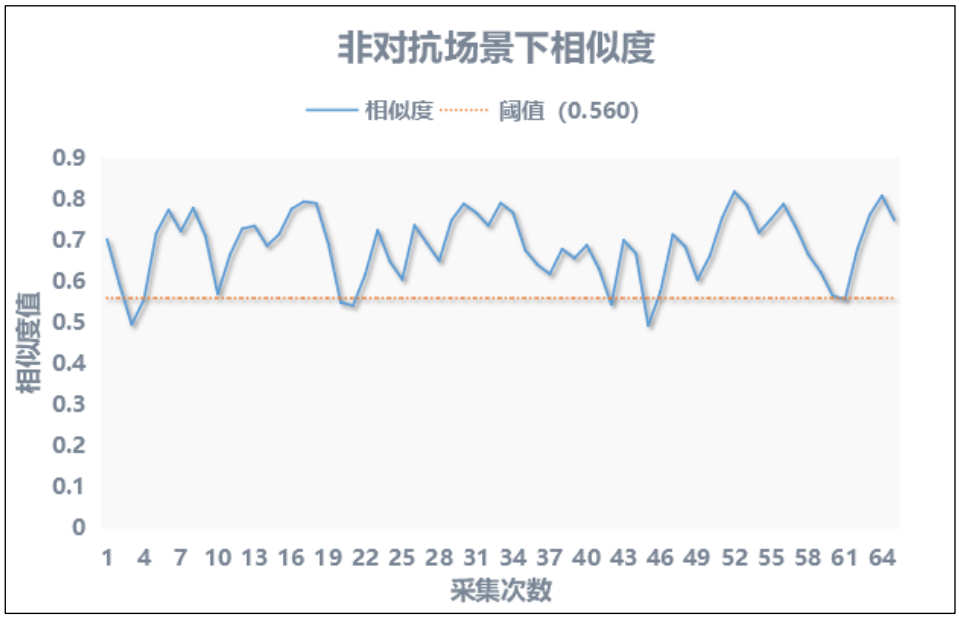


图 3 非对抗场景下相似度

而且该预检测诊断方案具有较高的可扩展性，时间复杂度为 $O(n)$，其中 n 为传感器数量。在非对抗场景下，可以根据需要增加更多传感器来进一步提高检测精度和覆盖范围。通过对振动加速度数据进行相似性检测和验证，使本方案确保数据的可靠性，同时有助于减少误报概率。

6.4 对抗场景中的组合方案性能

在对抗场景中，本文提出的集成式预检测诊断方案也同样表现出了显著的优势。针对对抗性攻击可能引发的振动信号异常、物理隔离破坏和恶意信号注入等问题，该方案通过准确率和召回率的综合评估，在确保系统性能的同时提高设备的安全性。

本文所采用的方案在另一种实验条件下也经过了验证，如图 4 所示，其中有 17 个连续三组数据的相似度低于阈值，但是根据设备的预测性维护系统的评判，有 14 组数据是具有潜在故障。

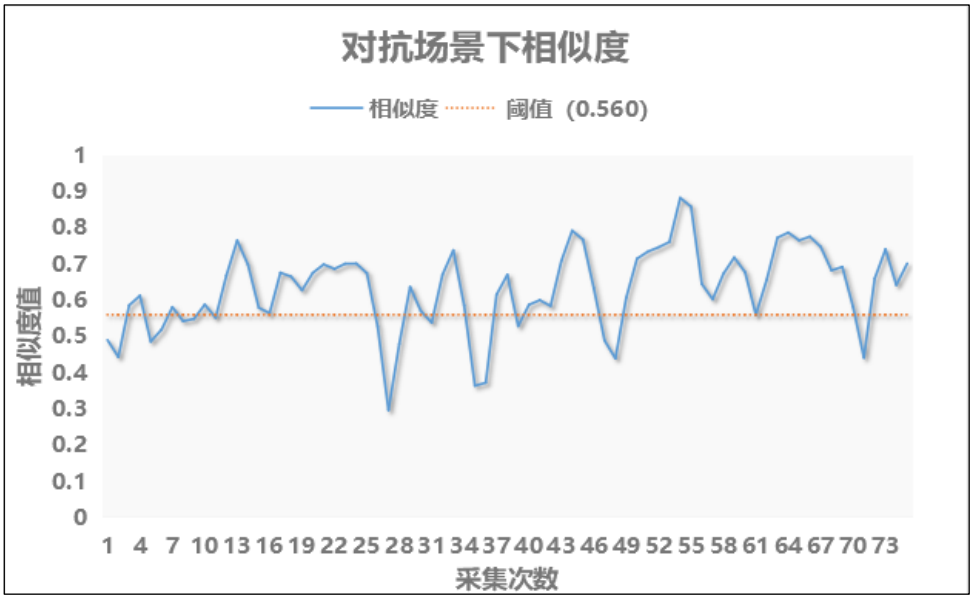


图 4 对抗场景下相似度

然而，在基于上一小节的实验条件并注入了随机干扰，实际上应该只有一到两组数据存在故障。然而，本文方案能够提前检测并预警出 17 组数据。而其中的 3 组数据虽然受到了干扰,但预测性维护系统并未检测到，而本文方案却成功进行了预警。同样进行计算得出方案的准确率为 96%和查全率为 100%。所以，本文提出的集成式预检测诊断方案在对抗场景中展现出了良好的性能表现。其可行性和可扩展性使得方案能够在保持较高效率的前提下，提升设备的安全性。

6.5 与其他方案比较

本文提出的集成式预检测诊断方案是可以嵌入任何基于其他模型的预测性维护系统。在前面文献中提到过许多近些年还在使用的方法，本节将对本文使用的集成式预检测诊断方案与一些常用模型的预测性维护系统进行对比。如表 2 所示，尽管本文方案只具有异常检测功能，而其他模型具有故障分类，故障检测与诊断和剩余寿命估计，但本文提出的集成式预检测诊断方案是可以嵌入任何基于其他模型（方法）的预测性维护系统。

相较于其他模型的预测性维护系统，本文方案在总体安全性、物理层防护和系统外围过滤几个指标，这是其它方案不具备。在对抗与非对抗场景下的性能的指标下，本文方案的表现接近或优于其他预测性维护系统。这意味着该方案在不同环境和攻击情景下都能提供可靠的故障检测和诊断能力，确保工业设备的正常运行。为工业物联网的安全提供了新的解决方案（表 2 中 1 级到 10 级代表从低到高）。

表 2 不同方法在安全指标上进行比较

方法	所用模型	应用	总体安全性	实时性	物理层防护	系统外围过滤	在对抗场景的性能	在非对抗场景的性能
[23]	退化模型	剩余寿命估计	×	×	×	×	4 级	4 级
[27]	CNN	故障诊断与检测/剩余寿命估计	×	×	×	×	5 级	4 级
[29]	SVM	异常检测/故障分类	×	×	×	×	4 级	3 级
特征类型相似性度量	无	故障诊断与检测/剩余寿命估计	×	√	×	×	5 级	5 级
集成式预检测诊断方案	无	异常检测	√	√	√	√	5 级	6 级

7 结束语

集成式预检测诊断方案虽然在实验室阶段具有一定的效果，但是集成式方案需要在实时或接近实时的情况下对数据进行处理和分析，以便及时做出预检测和诊断。其次，处理大量数据和实时响应的要求可能对计算资源和算法效率提出挑战。其次采集和分析大量的振动数据可能涉及到敏感的设备信息和操作数据。因此，在实施方案时，必须确保适当的数据隐私和安全措施，以防止未经授权的访问和数据泄露。

在进一步的研究中，计划通过更多实验来全面评估集成式预检测诊断方案的性能。这些实验将采用综合性的方法，从多个角度对方案进行论证和评估。通过实验、数据分析和理论推导等手段，关注准确率、查全率、误报率等性能指标，以及方案的鲁棒性、可扩展性和可靠性等方面。同时，还将与其他代表性的故障诊断方法进行对比，选取几种常用的方法，如基于统计特征、机器学习和深度学习等方法，以验证集成式方案在性能上的优势，并进行准确率、查全率等方面的比较，以量化集成式方案的有效性和适用性。为了提高实验的可信度和适用性，计划扩大实验规模，收集更多的振动数据，并模拟不同的工作场景和故障情况。通过在大规模实验中验证集成式方案的适用性和鲁棒性，这样能够更好地观察其在复杂环境下的性能表现。

尽管我们的集成式预检测诊断方案在实验室阶段已经显示出一定的效果，但在实际应用中还存在一些挑战。例如，方案需要在实时或接近实时的情况下对数据进行处理和分析，这对计算资源和算法效率提出了要求。此外，采集和分析大量振动数据可能涉及敏感的设备信息和操作数据，因此必须确保适当的数据隐私和安全措施。

当然，希望未来会采用多模态数据融合去解决对振动传感器的干扰问题，通过结合数据分析和神经网络算法，以构建更为丰富和全面的设备状态特征，从而在不同场景下实现更高的异常行为识别准确率。

在此基础上，我们还将进一步拓展研究内容，会考虑基于对抗网络的传感器故障下设备剩余寿命预测方法，或者通过多模态数据的深度学习获取泛化的设备健康状态特征以提高系统的自适应能力。具体工作将在未来的研究中进一步展开。

参考文献

[1]JARDINE A, LIN D, BANJEVIC D. A review on machinery diagnostics and prognostics implementing condition-based maintenance[J]. Mechanical Systems & Signal Processing, 2006(7): 20.

[2]王庆锋, 高金吉. 过程工业动态的以可靠性为中心的维修研究及应用[J]. 机械工程学报, 2012, 48(08): 135-143.
WANG Qingfeng, GAO Jinji. Research and application of reliability-centered maintenance for process industry dynamics[J]. Journal of Mechanical Engineering, 2012, 48(08): 135-143.

[3]周东华, 魏慕恒, 司小胜. 工业过程异常检测、寿命预测与维修决策的研究进展[J]. 自动化学报, 2013, 39(06): 711-722.
ZHOU Donghua, WEI Muheng, SI Xiaosheng. Research progress on industrial process anomaly detection, life prediction and maintenance decision[J]. Journal of Automation, 2013, 39(06): 711-722.

[4]FERNANDO B. Predictive Maintenance Market: The Evolution from Niche Topic to High ROI Application[EB/OL]. (2021-05). https://iot-analytics.com/predictive-maintenance-market-evolution-from-niche-topic-to-high-roi-application/.

[5]东吴证券. 工业智能运维百亿蓝海市场，容知日新：第三方运维龙头崛起[EB/OL]. (2022-07). http://stock.finance.sina.com.cn/stock/go.php/vReport_Show/kind/lastest/rptid/710576165248/index.phtml.

[6]周章宁, 李波. 基于深度学习的石油钻机微电网故障预测与维护研究[D]. 电子科技大学, 2021.

[7]ALSUWIAN T, AMIN A A, MAQSOOD M T, et al. Advanced Fault-Tolerant Anti-Surge Control System of Centrifugal Compressors for Sensor and Actuator Faults[J/OL]. Sensors, 2022, 22(10): 3864. DOI:10.3390/s22103864.

[8]ATTOUI I. Novel Fast and Automatic Condition Monitoring Strategy Based on Small Amount of Labeled Data[J/OL]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(2): 1266-1275. DOI:10.1109/TSMC.2020.3018102.

[9]ABURAKHIA S, SHAMI A. SB-PdM: A tool for predictive maintenance of rolling bearings based on limited labeled data[J/OL]. Software Impacts, 2023, 16: 100503. DOI:10.1016/j.simpa.2023.100503.

[10]ATTOUI I, BOUTASSETA N, FERGANI N. Novel Machinery Monitoring Strategy Based on Time–Frequency Domain Similarity Measurement With Limited Labeled Data[J/OL]. IEEE Transactions on Instrumentation and Measurement, 2021, 70: 1-8. DOI:10.1109/TIM.2020.3011874.

[11]ARUN P, LINCON S A, PRABHAKARAN N. An Automated method for the analysis of bearing vibration based on spectrogram pattern matching[J/OL]. Journal of Applied Research and Technology, 2019(2). DOI:10.22201/ICAT.16656423.2019.17.2.805.

[12]ABURAKHIA S, TAYEH T, MYERS R, et al. A Transfer Learning Framework for Anomaly Detection Using Model of Normality[C/OL]//2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). 2020: 0055-0061. DOI:10.1109/IEMCON51383.2020.9284916.

[13]ABURAKHIA S, TAYEH T, MYERS R, et al. Similarity-Based Predictive Maintenance Framework for Rotating Machinery[C/OL]//2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA). 2022: 1-6. DOI:10.1109/ICCSPA55860.2022.10019121.

[14]TRIPPEL T, WEISSE O, XU W, et al. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks[C/OL]//2017 IEEE European Symposium on Security and Privacy (EuroS&P). 2017: 3-18. DOI:10.1109/EuroSP.2017.42.

[15]TU Y, LIN Z, LEE I, et al. Injected and delivered: fabricating implicit control over actuation systems by spoofing inertial sensors[C]//Proceedings of the 27th USENIX Conference on Security Symposium. USA: USENIX Association, 2018: 1545-1562.

[16]ROY N, SHEN S, HASSANIEH H, et al. Inaudible voice commands: the long-range attack and defense[C]//Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation. USA: USENIX Association, 2018: 547-560.

[17]PETIT B J, STOTTELAAR B, FEIRI M, et al. Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR Black Hat Europe[Z]. 2015.

[18]YAN C. Can You Trust Autonomous Vehicles : Contactless Attacks against Sensors of Self-driving Vehicle[C/OL]. 2016. DOI:10.1145/1235.

[19]HUYNH K T, GRALL A, BÉRENGUER C. A Parametric Predictive Maintenance Decision-Making Framework Considering Improved System Health Prognosis Precision[J/OL]. IEEE Transactions on Reliability, 2019, 68(1): 375-396. DOI:10.1109/TR.2018.2829771.

[20]HU J, CHEN P. Predictive maintenance of systems subject to hard failure based on proportional hazards model[J/OL]. Reliability Engineering & System Safety, 2020, 196: 106707. DOI:https://doi.org/10.1016/j.ress.2019.106707.

[21]王汉禹, 陈震, 周笛, 等. 基于核函数-Wiener 过程的轧辊非线性退化建模与剩余寿命预测[J/OL]. 上海交通大学学报: 1-9. DOI:10.16183/j.cnki.jsjtu.2022.004.

WANG Hanyu, CHEN Zhen, ZHOU Di, et al. Nonlinear degradation modeling and remaining life prediction of rolls based on kernel function-Wiener process[J/OL]. Journal of Shanghai Jiaotong University: 1-9. DOI:10.16183/j.cnki.jsjtu.2022.004.

[22]冯吉, 万强, 王艺潼, 等. 基于大数据的总装设备的预测性维护系统[J]. 上海汽车, 2020(10): 12-15.

FENG Ji, Wan Qang, WANG Yitong, et al. Big data-based predictive maintenance system for final assembly equipment[J]. Shanghai Automotive, 2020(10): 12-15.

[23]NGUYEN K T P, MEDJAHER K. A new dynamic predictive maintenance framework using deep learning for failure prognostics[J/OL]. Reliability Engineering & System Safety, 2019, 188: 251-262. DOI:https://doi.org/10.1016/j.ress.2019.03.018.

[24]ZHAO R, WANG J, YAN R, et al. Machine health monitoring with LSTM networks[C/OL]//2016 10th International Conference on Sensing Technology (ICST). 2016: 1-6. DOI:10.1109/ICSensT.2016.7796266.

[25]ZHAO R, YAN R, WANG J, et al. Learning to Monitor Machine Health with Convolutional Bi-Directional LSTM Networks[J/OL]. Sensors, 2017, 17(2). DOI:10.3390/s17020273.

[26]LIU B, DO P, IUNG B, et al. Stochastic Filtering Approach for Condition-Based Maintenance Considering Sensor Degradation[J/OL]. IEEE Transactions on Automation Science and Engineering, 2020, 17(1): 177-190. DOI:10.1109/TASE.2019.2918734.

[27]SUSTO G A, SCHIRRU A, PAMPURI S, et al. Machine Learning for Predictive Maintenance: A Multiple Classifier Approach[J/OL]. IEEE Transactions on Industrial Informatics, 2015, 11(3): 812-820. DOI:10.1109/TII.2014.2349359.

[28]CARVALHO T P, SOARES F A A M N, VITA R, et al. A systematic literature review of machine learning methods applied to predictive maintenance[J/OL]. Computers & Industrial Engineering, 2019, 137: 106024. DOI:10.1016/j.cie.2019.106024.

[29]LI C, SANCHEZ R V, ZURITA G, et al. Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals[J/OL]. Mechanical Systems and Signal Processing, 2016, 76-77: 283-293. DOI:10.1016/j.ymssp.2016.02.007.

[30]SHI Q, ZHANG H. Fault Diagnosis of an Autonomous Vehicle With an Improved SVM Algorithm Subject to Unbalanced Datasets[J/OL]. IEEE Transactions on Industrial Electronics, 2021, 68(7): 6248-6256. DOI:10.1109/TIE.2020.2994868.

[31]袁烨, 张永, 丁汉. 工业人工智能的关键技术及其在预测性维护中的应用现状[J]. 自动化学报: 1-18.

YUAN Ye, ZHANG Yong, DING Han. Key technologies of industrial artificial intelligence and its application status in predictive maintenance[J]. Journal of Automation: 1-18.

[32]BERGHOUT T, BENBOUZID M, MOUSS L H. Leveraging Label Information in a Knowledge-Driven Approach for Rolling-Element Bearings Remaining Useful Life Prediction[J/OL]. Energies, 2021, 14(8): 2163. DOI:10.3390/en14082163.

[33]王岩, 曹现刚, 张旭辉, 等. 基于知识图谱的采煤机智能维护知识库构建[J/OL]. 工矿自动化: 1-9. DOI:10.13272/j.issn.1671-251x.17786.
WANG Yan, CAO Xiangang, ZHANG Xuhui, et al. Construction of knowledge base for intelligent maintenance of coal mining machine based on knowledge graph[J/OL]. Industrial and Mining Automation: 1-9. doi:10.13272/j.issn.1671-251x.17786.

黄道旗：男，硕士在读研究生，研究方向为信息系统安全.
张华君：男，博士，硕士生导师，研究方向信息安全、密码学、区块链、隐私保护的大数据技术.
徐守坤：男，教授，博士生导师，研究方向深度学习、自然语言处理、数字孪生技术.
庄丽华：女，讲师，研究方向计算机应用.

Test Primitive:A Unify Feature of Minimal Test Sequence for Memory Faults

Yindong Xiao, Shanshan Lu, Ensheng Wang,Zhijian Dai

***Abstract*—The academic community has made outstanding achievements in researching the March algorithm. However, the current fault modeling method, which centers on fault primitives, cannot be directly applied to analyzing the March algorithm. This paper proposes a new test primitive.The test primitives, which decouple the cell states from sensitization and detection operations, describe the common features that must be possessed for the March algorithm to detect corresponding faults, forming a highly flexible and scalable March algorithm analysis unit. The theoretical analysis proves that the test primitives demonstrate completeness, uniqueness, and conciseness. On this foundation, the utilization of test primitives within the March analysis procedure is elucidated.**

***Index Terms*—Memory Test,March Algorithm,Fault Primitive,Test Primitive.**

I. Introduction

MARCH algorithms, as a class of test vector generation algorithms, are widely used in the field of memory testing. Due to their excellent test efficiency and high fault coverage , they have been extensively studied by scholars at home and abroad [1]. Understanding of March algorithms in the industry is continuously deepening and becoming more abstract. Each new perspective and high-level theory has brought progress at the application level.In recent years, Jidin et al. proposed the cell trend theory based on address direction and the state of the previous memory cell [2]. They applied this theory to non-connectivity static fault coverage analysis and automatic generation of March algorithms, achieving significant progress [3], [4]. However, dealing with dynamic faults brought by technologies such as FinFET remains challenging.

On the other hand, changes in application-level demands have also guided the direction of theoretical research on March algorithms. With the increasing focus of industry on hierarchical memory diagnosis techniques [5], the need for scalable and flexible test primitives has arisen. In 2000, Van De Goor et al. [6]proposed fault primitives to represent various behavioral-level faults. This model has been widely used to describe memory faults. Building on this, Al-Ars et al. proposed the concept of test primitives with the goal of developing a scalable and flexible model dedicated to fault detection. This was proposed to address the issue that fault primitive models cannot be directly used for test diagnosis.However, the test primitive theory proposed by Al-Ars et al. [7] primarily focuses on ensuring uniqueness and conciseness of primitives. The feature of coupling memory cells states with sensitization and detection operations makes proving completeness difficult. Therefore, this theory cannot be applied to algorithm analysis and generation tasks.

Based on fault primitives, this paper defines new test primitives to describe the characteristics of memory access operations required to detect corresponding faults. The proposed test primitives demonstrate completeness, uniqueness, and conciseness. This paper examines the construction method for a test primitive model and establishes a test primitive library that covers non-connectivity static faults. The functional-level fault model of memory can be linked with the test primitives, simplifying analysis of the fault coverage of March algorithms.

Completeness: The test primitives encompass all characteristics of March algorithms that can detect corresponding faults.

Uniqueness: The test primitives do not include descriptions that cannot test corresponding faults, ensuring one fault primitive corresponds to only one test primitive.

Conciseness: When the test primitives cover corresponding faults, the number of detection operations is minimal without introducing redundant operations.

The content of this paper is organized as follows: Section 2 mainly introduces the target fault types and their test primitives. Section 3 proves the properties of the proposed test primitives. Section 4 elucidates the utilization of test primitives within the March analytical procedure.Finally, the full text is summarized.

II. Fault Model and Its Test Primitives

A. Fault Model

The fault model of memory can be divided into static and dynamic faults based on the number of operations required to sensitize the fault. Static faults fall into simple faults and connectivity faults depending on whether faults affect one another. While primarily considering non-connectivity static faults encompassing single-cell faults as well as coupled faults, which implicate multiple memory cells, this paper exclusively examines the scenario of two memory cells, namely double-cell coupled faults that represent a type of memory fault residing autonomously in a single memory cell without influencing other cells, as opposed to double-cell faults designating two coupled memory cells commonly termed aggressor cells (a) and victim cells (v). Common single-cell faults and double-cell coupled faults appear in Table II.The fault primitive expression of a single-cell fault is $\langle S/F/R\rangle$, where S represents the operation or state that triggers the memory fault behavior, F represents the state value of the faulty cell, and R represents the logic output level of the

read operation.The fault primitive expression of a double-cell fault is $\langle Sa; Sv/F/R\rangle$, Where Sa and Sv represent the state or operation of the aggressor cell and the victim cell that trigger the memory fault, respectively.Sa and Sv cannot refer to operations simultaneously [8].

B. Test Primitive Definition

With the continuous improvement of memory manufacturing technology, many new types of faults have emerged, requiring the March algorithm to continuously adapt to new types of faults. The high test complexity of the test algorithm makes the analysis, verification, and generation of the algorithm difficult. Based on fault primitives, this paper decouples the cell state in the March algorithm from fault sensitization and detection, establishes a test primitive , and simplifies the analysis, verification, and generation process of the March algorithm.

The test primitive is defined to describe the common characteristics of the March algorithm that can detect the corresponding fault primitive. It can be obtained directly from the fault primitives, and each fault primitive corresponds to a test primitive. The definition of the test primitive is based on the fault primitive, described in the form of$\langle S\rangle\, D$, which describes the characteristics of the operations required to detect the corresponding fault, where $\langle S\rangle$ is called the sensitization operation set, and D is the detection operation feature descriptor, used to describe the detection operation set features that can verify the cell state after sensitization. Similarly, the double cell fault $\langle Sa; Sv/F/R\rangle$ can be detected using the test primitive in the form of $\langle Sa; Sv\rangle\, D$.

Test primitives and fault primitives have the same sensitization operation set. Therefore, the key to deriving test primitives is studying the applicability of detection operation feature descriptors. The detection operation feature descriptor includes the read operation for detecting faults, hereinafter referred to as the detection operation, and special symbols describing the position of the detection operation. Symbols similar to regular expressions are used to indicate the position of the detection operation. Common symbols and their meanings are shown in Table I.The detection descriptor is divided into two cases according to the address change direction. "a⇒v" indicates that the address change direction is from the aggressor cell to the victim cell. "v⇒a" indicates the opposite. Before proposing this representation method in this paper, the forms "a>v" and "a<v" were used to indicate the relative address relationship between the aggressor cell and the victim cell.When generating the test primitive of the fault primitive, the relative address relationship between the aggressor cell and the victim cell needs to be considered in both the increasing address direction and the decreasing address direction. Using the "a⇒v" and "v⇒a" methods proposed in this paper combines the address change direction and the relative address relationship between the victim cell and the aggressor cell, making the analysis process more convenient.

According to the following method to derive the test primitive, where the fault cell of the double cell coupled fault refers to the victim cell:

TABLE I
COMMON SYMBOLS AND THEIR MEANINGS IN TEST PRIMITIVES

Symbol	Meaning
*	Indicates that 0 or more arbitrary memory access operations can be added.
#	Indicates that the sensitization and detection of faults can be split into two March elements at the current position, but this split is not required.If a split is not chosen, the address traversal order before and after "#" remains consistent.
;	Indicates that fault detection and fault sensitization be split into two March elements at the current position.
^	Indicates that the memory access operation following it must be in the first position of the March element,and fault sensitization is after the detection operation

1. If the fault cell executes a read operation during sensitization, and the expected state of this read operation differs from R in the fault primitive, $D = \phi$. Otherwise, there must be a detection operation in the detection operation descriptor.
2. The expected state of the detection operation is determined by the state achieved by sensitizing the fault cell, i.e. the state after the sensitization operation or the sensitization state itself.
3. For a single cell fault,"#" is added before the detection operation, regardless of whether the sensitization of the fault cell involves a state or an operation.
4. Double cell faults are divided into the following three cases according to the different methods of sensitizing the aggressor cell and the victim cell:
   a. If the aggressor cell executes an operation during sensitization, add "^" before the detection operation in the a⇒v case. Add "*" after the detection operation. Add ";" before the detection operation in the v⇒a case. Add "*" after the sensitization operation.
   b. If the victim cell executes an operation during sensitization, the detection operations come after the sensitization operation in both the a⇒v and v⇒a cases. "#" is added before the detection operation.
   c. If both the aggressor cell and the victim cell are in the state during sensitization, two cases must be distinguished. The aggressor cell and the victim cell can separately occupy the current address cell (CAC). When the aggressor cell occupies the CAC, insert "^" before the detection operation and "*" after it. When the victim cell occupies the CAC, the detection operation follows sensitization.

Here, it needs explanation that if the aggressor cell executes an operation during sensitization, the different positions of the detection operation in the a⇒v and v⇒a directions ensure that the detection operation lags behind fault sensitization in time. The above test primitive derive method also applies to dynamic faults because detection of dynamic faults is also completed by read operations after ensuring the fault sensitization conditions are met. As long as its detection operation meets the conditions for generating the above rules, its test primitive can also be easily express.

For example, the fault primitive of Incorrect Read Fault is

TABLE II
TEST PRIMITIVE LIBRARY

Fault Model	Fault Primitive	Test Primitive
SF	$\langle x/\overline{x}/-\rangle$	$\langle x\rangle \#Rx$
SAF	$\langle \forall/x/-\rangle$	$\langle x\rangle \#Rx$
TF	$\langle xW\overline{x}/x/-\rangle$	$\langle xW\overline{x}\rangle \#R\overline{x}$
WDF	$\langle xWx/\overline{x}/-\rangle$	$\langle xWx\rangle \#Rx$
RDF	$\langle Rx/\overline{x}/\overline{x}\rangle$	$\langle Rx\rangle$
DRDF	$\langle Rx/\overline{x}/x\rangle$	$\langle Rx\rangle \#Rx$
IRF	$\langle Rx/x/\overline{x}\rangle$	$\langle Rx\rangle$
CFst	$\langle x;y/\overline{y}/-\rangle$	$\langle x;y\rangle \begin{cases} Ry & (v\ in\ CAC) \\ \hat{}Ry & (a\ in\ CAC) \end{cases}$
CFdsrx	$\langle Rx;y/\overline{y}/-\rangle$	$\langle RX;y\rangle \begin{cases} \hat{}Ry* & (a\Rightarrow v) \\ *;Ry & (v\Rightarrow a) \end{cases}$
CFdsxwx	$\langle xWx;y/\overline{y}/-\rangle$	$\langle xWx;y\rangle \begin{cases} \hat{}Ry* & (a\Rightarrow v) \\ *;Ry & (v\Rightarrow a) \end{cases}$
CFdsxw!x	$\langle xW\overline{x};y/\overline{y}/-\rangle$	$\langle xW\overline{x};y\rangle \begin{cases} \hat{}Ry* & (a\Rightarrow v) \\ *;Ry & (v\Rightarrow a) \end{cases}$
CFtr	$\langle x;yW\overline{y}/y/-\rangle$	$\langle x;yW\overline{y}\rangle \#R\overline{y}$
CFwd	$\langle x;yWy/\overline{y}/-\rangle$	$\langle x;yWy\rangle \#Ry$
CFrd	$\langle x;Ry/\overline{y}/\overline{y}\rangle$	$\langle x;Ry\rangle$
CFdrd	$\langle x;Ry/\overline{y}/y\rangle$	$\langle x;Ry\rangle \#Ry$
CFir	$\langle x;Ry/y/\overline{y}\rangle$	$\langle x;Ry\rangle$

Please note:
1) For the CFdsrx fault, when $x=y$, detection operation is empty. However, the sensitized read operation must be in the first position S of the March element. According to the above description, its test primitive can be expressed as $\langle Rx;y\rangle\,(x=y)$.
2) For the CFst fault, when the victim cell occupies the CAC, another test primitive for $x=y$ is $\langle x;y\rangle\,;Ry$.

$\langle Rx/x/\overline{x}\rangle$, which shows that the sensitization of the fault cell is a read operation. The state after the read operation differs from R in the fault primitive, so $D=\phi$. Its sensitization operation set is $\langle Rx\rangle$. Therefore, the test primitive of the fault primitive $\langle Rx/x/\overline{x}\rangle$ is $\langle Rx\rangle$.

For cases where the aggressor cell executes an operation during sensitization, generating the test primitive is more complex. The fault primitive$\langle Rx;y/\overline{y}/-\rangle$ is used as an example of Disturb Coupling Fault to illustrate. The sensitization condition set of $\langle Rx;y/\overline{y}/-\rangle$ is $\langle Rx;y\rangle$. The sensitization of the aggressor cell is the operation Rx. The sensitization of the victim cell is the state y. Therefore, the detection operation is Ry. Because the aggressor cell executes an operation during sensitization, after the sensitization operation, 0 or more arbitrary memory access operations can be added without affecting fault detection. Add "*". It should be noted here that the final state after adding multiple memory access operations should remain the same as the state before sensitizing the victim cell. According to the above rules, the expression of D can be obtained.

$$D=\begin{cases} \hat{}Ry* & (a\Rightarrow v) \\ *;Ry & (v\Rightarrow a) \end{cases} \tag{1}$$

Its test primitives are shown in Table II. According to the above rules, the test primitive library of the fault set involved in this paper is recorded in Table II.

III. TEST PRIMITIVE NATURE PROOF

The test primitives proposed in this paper demonstrate completeness, uniqueness, and conciseness for the target fault primitives.If the test primitives lacked completeness, they would not guarantee the ability to analyze fault coverage for any March algorithm. Uniqueness ensures that when a certain March algorithm satisfies the test primitive description, it must detect the corresponding fault. Conciseness ensures that the test primitive can directly optimize test sequence generation.

The test primitive adds at most one read operation based on the sensitization operation, so conciseness is easier to guarantee. To address the completeness problem of the test primitive, this paper adopts proof by contradiction to prove the assumption that a certain test primitive cannot cover the March algorithm, but the algorithm can detect the fault corresponding to the original language. The existence of this kind of algorithm is analyzed. If it does not exist, the completeness of the test primitive can be proved.The uniqueness of the test primitive only needs to ensure the following three conditions simultaneously to prove its uniqueness:

1) The sensitization operation set of test primitive is the same as the sensitization operation set of the fault primitive.
2) The detection operation lags behind the sensitization operation in time.
3) After the sensitization operation completes and before the detection operation, no operation can change the state of the fault cell.

When deriving the test primitive, its uniqueness is proved; thus, it will not be repeated here.

The March algorithm comprises multiple March elements M. Each March element contains one of {⇑,⇓,⇕}representing address change direction and the memory access operation set$O=\{op0,op1,...\}$, $opn\in\{W0,W1,R0,R1\}$.

As defined by the detection operation feature descriptor, when $D\neq\phi$, the relationship between operation set Od in D and sensitization operation set Os may include:

$$;[O1]Od[O2]Os[O3] \tag{2}$$

$$Od[O1];[O2]Os \tag{3}$$

$$Os[O1]Od[O2] \tag{4}$$

$$Os[O1];[O2]Od \tag{5}$$

In the formula,";" signifies the end of one march in the March algorithm, and "[...]" indicates optional content within brackets. The above four formulas describe 20 possible positional relationships. In addition to $D=\phi$, only 21 cases require discussion.

Among the above 21 relationships between detection and sensitization operations, assume that $O1$ exists in formula 2—that is, $;O1Od[O2]Os[O3]$—if $O1$ is a read operation and the expected state from the read operation differs from the fault cell state, such a read-write sequence does not comply with March rules. If $O1$ is a read operation and its expected state matches the fault cell state, $O1$ is Od. If $O1$ contains write operations, the write operation will change the fault cell state, preventing fault detection. Therefore, formulas containing $O1$

do not satisfy test primitive conditions. Similarly for 5 when $O2$ exists. No formula in 3 satisfies the condition of detection operations lagging behind sensitization operations in time; thus, they do not meet test primitive conditions. In summary, we simplify the positional relationship to:

$$;Od[O2]Os[O3] \tag{6}$$

$$Os[O1]Od[O2] \tag{7}$$

$$Os[O1];Od \tag{8}$$

Through the above formula, adding $D = \phi$ leaves only 11 remaining combinations to consider. Next, the completeness of the test primitives for the faults addressed in this paper is demonstrated through proof by contradiction.

A. Proof of Completeness for Single Cell Fault Testing Primitives

The fault primitive for State Fault is $\langle x/\overline{x}/-\rangle$. The sensitization operation set is $\langle x\rangle$, and the detection operation is Rx. According to Table II, its test primitive is $\langle x\rangle \#Rx$. Assume that in addition to $\langle x\rangle \#Rx$, other test primitives can detect $\langle x/\overline{x}/-\rangle$. The positional relationship between sensitization operation and detection operation contains:

1. Formula 6 contains four cases: (1) $O2$ and $O3$ exist; (2) $O2$ exists, $O3$ does not; (3) $O2$ does not exist, $O3$ exists; (4) Neither $O2$ nor $O3$ exist. The positional relationships for these cases are: (1) $;RxO2\,\langle x\rangle\,O3$, (2); $RxO2\,\langle x\rangle$,(3); $Rx\,\langle x\rangle\,O3$,(4); $Rx\,\langle x\rangle\,O3$. These cannot guarantee detection operation after sensitization operation, so they cannot detect State Fault.
2. Formula 7 contains four positional relationships: (1) $O1$ and $O2$ exist, $\langle x\rangle\,O1RxO2$, (2) $O1$ exists, $\langle x\rangle\,O1Rx$, (3) $O2$ exists, $\langle x\rangle\,RxO2$, (4) Neither exists, $\langle x\rangle\,Rx$. When $O1$ exists, multiple memory accesses operation after sensitization may change the fault cell state, so (1) and (2) cannot detect State Fault. However, (3) and (4) can. Because memory accesses operation after detection operation do not affect detection, (3) and (4) merge into $\langle x\rangle\,Rx$.
3. Formula 8, When $O1$ exists, the positional relationship is $\langle x\rangle\,O1;Rx$. Memory accesses operation after sensitization may change the fault cell state, preventing detection. When $O1$ does not exist, the positional relationship is $\langle x\rangle\,;Rx$. The test primitive formed by this relationship can detect State Fault.Analysis shows two test primitives, $\langle x\rangle\,Rx$ and $\langle x\rangle\,;Rx$, detect State Fault. Merging these yields $\langle x\rangle\,\#Rx$. Other test primitives cannot detect state faults. This proves the completeness of the State Fault test primitive.

The Read Destructive Fault occurs when a read operation inverts the fault cell state, reading the incorrect state. The read operation for sensitization detects this directly. The test primitive is $\langle Rx\rangle$, where the D is ϕ. Ultimately, there is only one Rx operation. Its position and surrounding memory accesses operation are irrelevant. The final test primitive is $\langle Rx\rangle$, proving its completeness. The analysis of Incorrect Read Fault test primitive's completeness is similar and omitted here.

B. Proof of Completeness for Double-cell Fault Coupled Testing Primitives

To prove the completeness of double-cell coupled fault test primitives, this paper demonstrates three different scenarios:

a. The aggressor cell and victim cell are both in the state during sensitization.
b. The aggressor cell is in the operation during sensitization.
c. The victim cell is in the operation during sensitization.

1) The aggressor cell and victim cell are both in the state during sensitization: State Coupling Faults have the aggressor cell and victim cell both in the state during sensitization. The fault primitive is $\langle x;y/\overline{y}/-\rangle$, and the test primitive is shown in Table II. Special cases have been introduced previously. Assume there are other forms of test primitives that can detect State Coupling Faults:

1. According to formula 6,When $O2$ exists,the positional relationships $;RyO2\,\langle x;y\rangle\,O3$ and $;RyO2\,\langle x;y\rangle$ form test primitives that can detect State Coupling Faults when the aggressor cell occupies the current address cell if x≠y.When $O2$ does not exist,the positional relationships $;Ry\,\langle x;y\rangle\,O3$ and $;Ry\,\langle x;y\rangle$ form test primitives that can detect State Coupling Faults when the aggressor cell is in the CAC if $x = y$.Because the memory access operation following the detection operation does not impact fault detection,the test primitives formed by the above positional relationships can be summarized as follows:$\langle x;y\rangle\ \hat{}\,Ry*$.
2. According to formula 7,When $O1$ exists,the positional relationships $\langle x;y\rangle\,O1RyO2$ and $\langle x;y\rangle\,O1Ry$ introduce multiple memory access operations following fault sensitization. If these operations alter the state of the fault cell, detection will fail.The positional relationships $\langle x;y\rangle\,RyO2$ and $\langle x;y\rangle\,Ry$ form test primitives that can detect State Coupling Faults when the victim cell occupies the CAC.Through analysis and synthesis,the test primitive $\langle x;y\rangle\,Ry$ is obtained.
3. According to formula 8,When $O1$ exists, the positional relationship is $\langle x;y\rangle\,O1;Ry$. The test primitive formed based on this positional relationship adds multiple memory access operations after fault sensitization. When $O1$ does not exist, the positional relationship is $\langle x;y\rangle\,;Ry$. The test primitive it forms can detect State Coupling Faults when the victim cell occupies the CAC if $x = y$. When x≠y, the sensitization condition cannot be guaranteed because the memory array will be written to the same state before a March element starts.Through the above analysis, the formed positional relationships $\langle x;y\rangle\,Ry$ and $\langle x;y\rangle\ \hat{}\,Ry*$ are merged to obtain the test primitive:

$$\langle x;y\rangle \begin{cases} Ry & (v\ in\ CAC) \\ \hat{}Ry* & (a\ in\ CAC) \end{cases} \tag{9}$$

It can be seen that except for the form of the test primitive we have given, there are no other forms of test primitives that can detect State Coupling Faults, proving its completeness.

2) The aggressor cell is in the operation during sensitization: Disturb Coupling Fault CFdsrx has the aggressor cell sensitization as the operation during sensitization. Its fault primitive is $\langle Rx; y/\overline{y}/-\rangle$, and the test primitive can be obtained in Table II. Assume there are other forms of test primitives that can detect CFdsrx. The analysis process is as follows:

1. According to formula 6, when $O2$ exists, the positional relationships are $; RyO2\,\langle Rx; y\rangle\, O3$ and $; RyO2\,\langle Rx; y\rangle$. The test primitives formed based on these can detect CFdsrx in the a⇒v direction when x≠y. The positional relationships when $O2$ does not exist are $; Ry\,\langle Rx; y\rangle\, O3$ and $; Ry\,\langle Rx; y\rangle$. The test primitives formed based on these can detect CFdsrx in the a⇒v direction when $x = y$.Organizing the test primitives formed by the above positional relationships, it is found that adding memory access operations after Ry can detect CFdsrx in the a⇒v direction when x≠ y, whereas not adding memory access operations can detect CFdsrx in the a⇒v direction when $x = y$. The "*" can be add and obtain the test primitive.
2. According to formula 7, four positional relationships can be obtained, but within these positional relationships, the sensitive operation and detection operation are in the same March element. In the current address cell, both the sensitive operation and detection operation act on the aggressor cell. However, when performing fault detection, it should be ensured that the detection operation acts on the victim cell. Therefore, the test primitives formed by these four positional relationships cannot detect CFdsrx faults.
3. According to formula 8,When $O1$ exists, the positional relationship at this time is $\langle Rx; y\rangle\, O1; Ry$. The test primitive formed based on this can detect CFdsrx faults in the v⇒a direction when x≠y. When $O1$ does not exist, the positional relationship at this time is $\langle Rx; y\rangle\, ; Ry$. The test primitive formed based on this can detect CFdsrx faults in the v⇒a direction when $x = y$. The two positional relationships are merged together as $\langle Rx; y\rangle *; Ry$(v⇒a).Through the above analysis, the test primitives that can detect CFdsrx faults are obtained:

$$\langle Rx; y\rangle \begin{cases} \hat{}Ry* & (a \Rightarrow v) \\ *; Ry & (v \Rightarrow a) \end{cases} \tag{10}$$

It can be concluded that there are no other forms of test primitives that can detect this fault, proving the completeness of its test primitives. The completeness proofs of the two Disturb Coupling Fault test primitives CFdsxwx and CFdsxw!x are the same as the above type and will not be further proved in this paper.

3) The victim cell is in the operation during sensitization: The victim cell of Transition Coupling Fault is the operation during sensitization. Its fault primitive is $\langle x; yW\overline{y}/y/-\rangle$, and the test primitive is $\langle x; yW\overline{y}\rangle\, \#R\overline{y}$. Assume there are other forms of test primitives that can detect Transition Coupling Fault. Discuss according to the following conditions:

1. According to formula 6, four positional relationships can be obtained. Because it cannot be guaranteed that fault detection temporally lags fault sensitization, the test primitives formed by these four positional relationships cannot detect Transition Coupling Fault.
2. According to formula 7, when $O1$ exists, the positional relationships are $\langle x; yW\overline{y}\rangle\, O1R\overline{y}O2$ and $\langle x; yW\overline{y}\rangle\, O1R\overline{y}$. After sensitization, there are multiple memory access operations that may change the state of the fault cells and cannot detect Transition Coupling Fault. The test primitives formed by the positional relationships $\langle x; yW\overline{y}\rangle\, R\overline{y}O2$ and $\langle x; yW\overline{y}\rangle\, R\overline{y}$ when $O1$ does not exist can detect Transition Coupling Fault. Because the memory access operations after the detection operation do not affect fault detection, the above positional relationships $\langle x; yW\overline{y}\rangle\, R\overline{y}O2$ and $\langle x; yW\overline{y}\rangle\, R\overline{y}$can be merged into $\langle x; yW\overline{y}\rangle\, R\overline{y}$.
3. According to formula 8,When $O1$ exists, the positional relationship at this time is $\langle x; yW\overline{y}\rangle\, O1; R\overline{y}$. After sensitization, there are multiple memory access operations that may change the state of the fault cells and cannot detect Transition Coupling Fault. When $O1$ does not exist, the positional relationship at this time is $\langle x; yW\overline{y}\rangle\, ; R\overline{y}$. The test primitive formed based on this can detect Transition Coupling Fault.

Through the above analysis, the test primitives formed by the two positional relationships $\langle x; yW\overline{y}\rangle\, R\overline{y}$ and $\langle x; yW\overline{y}\rangle\, ; R\overline{y}$ respectively can detect Transition Coupling Fault. Merge them into the test primitive $\langle x; yW\overline{y}\rangle\, \#R\overline{y}$. It can be seen that except for the test primitive given in Table II, there are no other forms of test primitives that can detect Transition Coupling Fault, proving its completeness. The completeness proofs of Write Destructive Coupling Fault, Read Destructive Coupling Fault, Deceptive Read Destructive Coupling Fault and Incorrect Reading Coupling Fault test primitives are similar to that of Transition Coupling Fault and will not be repeated here.

IV. The Application of Test Primitives

The purpose of test primitives is to serve as a bridge between fault primitives and March algorithms to simplify the study of March algorithms. Test primitives describe the minimum test sequence corresponding to the fault primitives.In order to decouple cell status and fault sensitivity/detection, state tuples are generated, based on the test primitives, to represent the state requirements for the minimum detection sequence to detect faults. At the same time, we also generate state tuples for March algorithms during their progress to describe the state of the entire memory array when reaching a certain memory access operation. On this basis, the test primitives and March algorithms are matched to complete the analysis of March algorithms.

A. Definition and Generation of State Tuples

The state tuple of a test primitive represents the state conditions required for fault detection and the state tuple of a March algorithm represents the state of the memory array when the

March algorithm proceeds to a given memory access operation. The expression of a state tuple is $\langle LAS/CAS/HAS\rangle$, where LAS refers to the status of the low address cell, CAS refers to the current address cell status, and HAS refers to the high address cell status. $LAS, CAS, HAS \in \{0, 1, x\}$, where 0 indicates that the memory cell state is 0, 1 indicates that the memory cell state is 1, and x indicates that the memory cell state does not matter.

1) Generation of State Tuples in Test Primitive: The generation of test primitive state tuples primarily depends on sensitizing the aggressor cell and victim cell. The steps to derive test primitives for coupled faults can be summarized as follows:

a. Obtain the sensitization operations set from the test primitive;
b. For sensitive operation sets that contain operations, the required status for those operations is placed in the current address cell's status . For sensitive operation sets without operations, any status is placed in the current address cell. Remove any sensitive conditions that have already been used in the current sensitive operation set.
c. According to the relative address positions of the aggressor cell and the victim cell, the remaining sensitization conditions in the sensitization operation set are used to determine the state of the high address cell or the low address cell, and the other element is set to x.
d. Generate the test primitive with state tuple by placing the state tuple before the sensitization operations and including the D detection operation.If only states are required, include only the state tuple and D detection operation in the final test primitive.

For the state tuple of a single cell fault test primitive, simply change the current address cell status in the state tuple to the state required for fault sensitization. Set the high and low address cell status to x. The state required for the sensitization operation can be discussed in the following cases:

1. When the sensitization operation is a read operation, the state required for the sensitization operation is the expected state to be read out by this read operation, such as state 0 in r0.
2. When the sensitization operation is a write operation, the state required for the sensitization operation is the state before the write operation is performed, such as state 0 in 0w1.
3. When the sensitization operation is a state, the state required for the sensitization operation is the current state, such as state 0 in state 0.

For the Transition Coupling Fault, when the aggressor cell is 0 and the victim cell is $1W0$ during fault sensitization, the test primitive is $\langle 0; 1W0\rangle\, \#R0$. Its fault sensitization set can be obtained as $\langle 0; 1W0\rangle$. During fault sensitization, operations are required. Therefore, set CAS in the state tuple to 1, representing the state required to sensitize the victim cell.Assuming the aggressor cell address is higher than the victim cell address, set HAS in the state tuple to 1, representing the state required to sensitize the aggressor cell. Set LAS to x . Therefore, the state tuple is $\langle x, 1, 1\rangle$. The obtained test primitive with state tuple is $\langle x, 1, 1\rangle\, W0\#R0$.

2) Generation of State Tuples in March Algorithms : The generation of the entire state tuple of the March algorithm is done with March elements as cells. Generating the state tuple before the first memory access operation in each March element is relatively complex. From the second memory access operation onwards, this state tuple equals the state tuple following the previous operation. For conciseness, only the state tuple preceding each memory access operation is written. The state tuple following an operation depends on the operation type. A read operation does not alter the memory cell state, so the state tuple following equals that preceding. A write operation only modifies the current address, so the high and low address states in the tuple following remain unchanged. Only the current address state is rewritten to that following the write. For example, after w0 the current address state is 0.Here, two March element concepts are introduced: initial state and end state. In the March algorithm, from the second element onward the initial state equals the end state of the previous element. The initial state of the first element is either preset or x. The end state equals that following the final memory access operation. For example, there is a March element $\Uparrow (r0, r0)$,and the final operation is r0, the end state is 0.

Deriving the state tuple for the first memory access operation in a March element requires defining the initial and end states of that element. The following conclusions apply:

a. For $\Uparrow$ address order, the state tuple is $\langle end\ state, initial\ state, initial\ state\rangle$.
b. For $\Downarrow$ address order, the state tuple is $\langle initial\ state, initial\ state, end\ state\rangle$.
c. For $\Updownarrow$ address order, either tuple in a or b can apply.

The expression for March LR is: $\Updownarrow$(w0); $\Downarrow$(r0,w1); $\Uparrow$(r1,w0,r0,w1); $\Uparrow$(r1,w0); $\Uparrow$(r0,w1,r1,w0); $\Uparrow$(r0). For the first element, the address order is $\Updownarrow$, the initial state is x, and the end state is 0. The state tuple is $\langle 0, x, x\rangle$. For the second element, the address order is $\Downarrow$. The initial state is the end state (0) of the first element. The end state is the state after the final operation w1, so is 1. The initial state tuple is $\langle 0, 0, 1\rangle$.The state tuple generation schematic for the first memory access operation appears in Figure 1. Boxes above the March expression denote end states of the March element. Lines below the expression, arrowed, indicate state tuple state sources. Subsequent tuples is following state tuple after the previous operation. Combining the operations and tuples for the second element: $\langle 0, 0, 1\rangle \Downarrow r0\, \langle 0, 0, 1\rangle \Downarrow w1$.Repeating this for each element in March LR yields: $\langle 0, x, x\rangle \Updownarrow w0; \langle 0, 0, 1\rangle \Downarrow r0\, \langle 0, 0, 1\rangle \Downarrow w1; \langle 1, 1, 1\rangle \Uparrow r1\, \langle 1, 1, 1\rangle \Uparrow w0\, \langle 1, 0, 1\rangle \Uparrow r0\, \langle 1, 0, 1\rangle \Uparrow w1; \langle 0, 1, 1\rangle \Uparrow r1\, \langle 0, 1, 1\rangle \Uparrow w0; \langle 0, 0, 0\rangle \Uparrow r0\, \langle 0, 0, 0\rangle \Uparrow w1\, \langle 0, 1, 0\rangle \Uparrow r1\, \langle 0, 1, 0\rangle \Uparrow w0; \langle 0, 0, 0\rangle \Uparrow r0$.

B. The Analysis Process of the March Algorithm

The March analysis process involves matching a test primitive with state tuples to a March expression with state tuples. If a match is found, then all faults detectable by the test primitive can also be detected by March. Matching a test primitive and

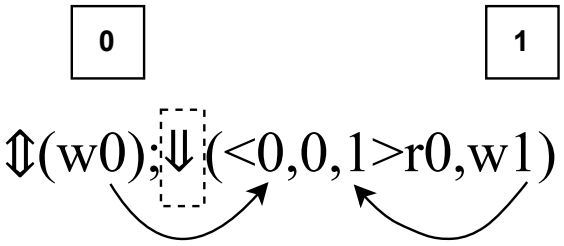


Fig. 1. State Tuple Generation Schematic

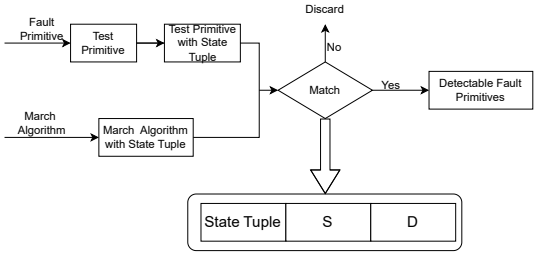


Fig. 2. Analysis Process Block Diagram

March requires ensuring their state tuples, sensitized operation sets, and detection operation feature descriptor are compatible. The block diagram of the analysis process is shown in Figure 2.

For test primitive state tuples to match, non-x state values must remain unchanged in one March algorithm state tuple, while x-state values can be any state. To match the sensitized operation set, required operations and order must be the same. For the detection operation feature descriptor to match, the detection operation and special symbols must be matched. As shown in Table I, these symbols "^", "#", ";" denote the relative positional relationship between detection and sensitization operations. Matching symbols requires matching these relationships. Adding a memory access operation at "*" that does not change the sensitized state is a match. Matching usually proceeds by first matching sensitized and detection operations, then state tuples, and finally special symbols.

For example, one of the fault primitives in Transition Coupling Fault (CFtr) mentioned above is $\langle 1; 1W0/1/-\rangle$. Its test primitive is $\langle 1; 1W0\rangle\, \#R0$. Assuming the aggressor cell address is lower than the victim cell address, the test primitive with state tuples is $\langle 1, 1, x\rangle\, W0\#R0$. For descriptive convenience, March LR memory access operations 1 through 14 are numbered.As Figure 3 shows, the 14 access operations are placed in the grid in the order from left to right and from top to bottom. The green box in the figure indicates the matching sensitization operation, the purple box indicates the matching detection operation, the red box represents the mismatched state, and the blue box represents the matched state. It can be seen that access operation 1, access operation 9 and access operation 13 are the same, all of them are w0, which can match the sensitization operation; the three have access operation 2, access operation 10 and access operation 14 after them, respectively, as detection operation r0; but the state tuples of the three do not match the test primitive, so they

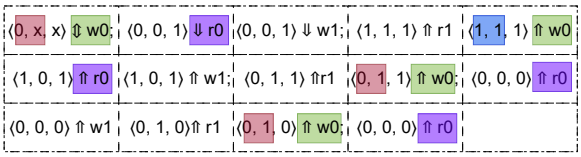


Fig. 3. State Primitive-March Algorithm Matching Schematic

cannot detect the above-mentioned fault with fault primitive $\langle 1; 1W0\rangle\, \#R0$. The access operation 5 is w0, which can match the sensitization operation; it is followed by the access operation 6 as the detection operation r0; and its state tuple can match the state tuple of the test primitive. In terms of symbols, the test primitive of CFtr uses "#" to indicate the position relationship between the sensitization operation and the detection operation, indicating that the two can be split in different March elements or not in the same March element. And the access operation 5 and access operation 6 are in the same third March element in March LR, which satisfies the requirement of the test primitive, and this test primitive is matched in March LR, which means the above fault can be detected.

1) Analysis of Fault Coverage for March LR and March LR2: The March LR expression and state tuple expression were provided above. According to Table II,all non-connectivity static fault primitives and test primitives were obtained. Applying the method for obtaining state tuples yielded 86 test primitives with state tuples. These were matched with March LR using state tuples and the matching method described previously. Fault primitives detectable by March LR are summarized in Table III, where each row denotes one fault model. Entries indicate March LR fault primitives that can detect that model. For coupling faults,"a>v" signifies the aggressor cell address exceeds the victim cell address,"a<v" signifies the aggressor cell address is below the victim cell address, and "↕" signifies fault primitives in both directions are detectable.The rightmost column characterizes the fault coverage of March algorithm for that fault type. March LR detects 58 of the 86 fault primitives.

According to literature [3], the authors have obtained many new March algorithms by modifying the data background sequence, etc. This paper uses one of the March LR2 algorithms for specific analysis.March LR2 remains the same in the address order and the number of access operations for each March element compared to March LR, but the access operations inside the March element are changed. The expression is: ⇳(w0); ⇓(w0,r0); ⇑(r0,w1,r1,w0); ⇑(r0,w1); ⇑(w1,r1,r1,w0); ⇑(r0).The expressions of March LR2 with state tuples are obtained and recorded in Table IV, where each row represents a March element.

Upon generating state-tuple test primitives from Table II, these were matched with the March LR2 expression with state tuples. The resulting fault primitives detectable by March LR2 were obtained and statistically summarized in Table V, whereby March LR2 detects 55 of the 86 fault primitives. It can be seen that March LR2 detects two fewer fault primitives than March LR, but increases the number of fault types that

TABLE III
FAULT PRIMITIVES DETECTED BY MARCH LR

Fault Model	Detectable Fault Primitives	Fault Coverage
SF	$\langle 1/0/- \rangle$,$\langle 0/1/- \rangle$	2/2
SAF	$\langle \forall/0/- \rangle$,$\langle \forall/1/- \rangle$	2/2
RDF	$\langle R1/0/0 \rangle$,$\langle R0/1/1 \rangle$	2/2
IRF	$\langle R1/1/0 \rangle$,$\langle R0/0/1 \rangle$	2/2
TF	$\langle 1W0/1/- \rangle$,$\langle 0W1/0/- \rangle$	2/2
WDF	unable to detect	0/2
DRDF	unable to detect	0/2
CFst	$\langle 0;0/1/- \rangle \updownarrow$, $\langle 1;1/0/- \rangle \updownarrow$, $\langle 0;1/0/- \rangle \updownarrow$, $\langle 1;0/1/- \rangle \updownarrow$	8/8
CFrd	$\langle 0;R0/1/1 \rangle \updownarrow$, $\langle 1;R1/0/0 \rangle \updownarrow$, $\langle 0;R1/0/0 \rangle \updownarrow$,$\langle 1;R0/1/1 \rangle \updownarrow$	8/8
CFir	$\langle 0;R0/0/1 \rangle \updownarrow$,$\langle 1;R1/1/0 \rangle \updownarrow$, $\langle 0;R1/1/0 \rangle \updownarrow$,$\langle 1;R0/0/1 \rangle \updownarrow$	8/8
CFdsrx	$\langle R0;1/0/- \rangle \updownarrow$, $\langle R1;0/1/- \rangle \updownarrow$, $\langle R0;0/1/- \rangle \updownarrow$,$\langle R1;1/0/- \rangle \updownarrow$	8/8
CFdsxw!x	$\langle 0W1;0/1/- \rangle \updownarrow$, $\langle 0W1;1/0/- \rangle \updownarrow$, $\langle 1W0;0/1/- \rangle \updownarrow$, $\langle 1W0;1/0/- \rangle \updownarrow$	8/8
CFdsxwx	unable to detect	0/8
CFtr	$\langle 0;0W1/0/- \rangle \updownarrow$,$\langle 0;1W0/1/- \rangle \updownarrow$, $\langle 1;0W1/0/- \rangle \updownarrow$,$\langle 1;1W0/1/- \rangle \updownarrow$	8/8
CFwd	unable to detect	0/8
CFdrd	unable to detect	0/8
Total	/	58/86

TABLE IV
MARCH LR2 EXPRESSION WITH STATE TUPLES

Number	Expression
1	$\langle 0,x,x \rangle \Updownarrow w0$
2	$\langle 0,0,0 \rangle \Downarrow w0 \langle 0,0,0 \rangle \Downarrow r0$
3	$\langle 0,0,0 \rangle \Uparrow r0 \langle 0,0,0 \rangle \Uparrow w1 \langle 0,1,0 \rangle \Uparrow r1 \langle 0,1,0 \rangle \Uparrow w0$
4	$\langle 1,0,0 \rangle \Uparrow r0 \langle 1,0,0 \rangle \Uparrow w1$
5	$\langle 0,1,1 \rangle \Uparrow w1 \langle 0,1,1 \rangle \Uparrow r1 \langle 0,1,1 \rangle \Uparrow r1 \langle 0,1,1 \rangle \Uparrow w0$
6	$\langle 0,0,0 \rangle \Uparrow r0$

can be detected, which are DRDF, WDF, CFdrd, CFwd and CFdsxwx. It is worth noting that in the literature [3], the March LR2 algorithm can detect 6 of the 8 CFtr fault primitives. However, by the analysis of this paper, it can only detect 5, where $\langle 1;0W1/0/- \rangle a < v$ is not detectable.The reason for this is that the literature [3] considers that other access operations can be added between the sensitization operation and the detection operation, which can be interpreted as the detection operation feature descriptor containing "*" using the notation of this paper. However, the detection feature descriptor "#Rx" of the test primitive of CFtr given in this paper does not contain "*", because in this fault type, once there is another access operation between the sensitization operation and the detection operation it will inevitably refresh the fault cell state, resulting in fault behavior is masked and thus undetectable.

2) Analysis of Fault Coverage for March SR: The expression for March SR is: $\Updownarrow (w0); \Uparrow (r0,w1,r1,w0); \Uparrow (r0,r0); \Uparrow (w1); \Downarrow (r1,w0,r0,w1); \Downarrow (r1,r1)$. Its state-tuple representation is: $\langle 0,x,x \rangle \Updownarrow (w0); \langle 0,0,0 \rangle \Uparrow r0 \langle 0,0,0 \rangle \Uparrow w1 \langle 0,1,0 \rangle \Uparrow r1 \langle 0,1,0 \rangle \Uparrow w0; \langle 0,0,0 \rangle \Uparrow r0 \langle 0,0,0 \rangle \Uparrow r0; \langle 1,0,0 \rangle \Uparrow w1; \langle 1,1,1 \rangle \Downarrow r1 \langle 1,1,1 \rangle \Downarrow w0 \langle 1,0,1 \rangle \Downarrow r0 \langle 1,0,1 \rangle \Downarrow w1; \langle 1,1,1 \rangle \Downarrow r1 \langle 1,1,1 \rangle \Downarrow r1$. Upon matching these with state-tuple test primitives, the detectable fault primitives were obtained and summarized in Table VI, whereby March SR detects 64 of the 86 fault primitives.

TABLE V
FAULT PRIMITIVES DETECTED BY MARCH LR2

Fault Model	Detectable Fault Primitives	Fault Coverage
SF	$\langle 1/0/- \rangle$,$\langle 0/1/- \rangle$	2/2
SAF	$\langle \forall/0/- \rangle$,$\langle \forall/1/- \rangle$	2/2
RDF	$\langle R1/0/0 \rangle$,$\langle R0/1/1 \rangle$	2/2
IRF	$\langle R1/1/0 \rangle$,$\langle R0/0/1 \rangle$	2/2
TF	$\langle 1W0/1/- \rangle$,$\langle 0W1/0/- \rangle$	2/2
WDF	$\langle 1W1/0/- \rangle$,$\langle 0W0/1/- \rangle$	2/2
DRDF	$\langle R1/0/1 \rangle$,$\langle R0/1/0 \rangle$	2/2
CFst	$\langle 0;0/1/- \rangle \updownarrow$, $\langle 1;1/0/- \rangle a > v$, $\langle 0;1/0/- \rangle \updownarrow$, $\langle 1;0/1/- \rangle a < v$	6/8
CFrd	$\langle 0;R0/1/1 \rangle \updownarrow$, $\langle 1;R1/0/0 \rangle a > v$, $\langle 0;R1/0/0 \rangle \updownarrow$,$\langle 1;R0/1/1 \rangle a < v$	6/8
CFir	$\langle 0;R0/0/1 \rangle \updownarrow$,$\langle 1;R1/1/0 \rangle a > v$, $\langle 0;R1/1/0 \rangle \updownarrow$,$\langle 1;R0/0/1 \rangle a < v$	6/8
CFdsrx	$\langle R1;0/1/- \rangle \updownarrow$,$\langle R0;0/1/- \rangle \updownarrow$	4/8
CFdsxw!x	$\langle 0W1;0/1/- \rangle \updownarrow$,$\langle 1W0;0/1/- \rangle \updownarrow$	4/8
CFdsxwx	$\langle 0W0;0/1/- \rangle a < v$, $\langle 1W1;0/1/- \rangle a > v$	2/8
CFtr	$\langle 0;0W1/0/- \rangle \updownarrow$,$\langle 0;1W0/1/- \rangle \updownarrow$, $\langle 1;1W0/1/- \rangle a > v$	5/8
CFwd	$\langle 0;0W0/1/- \rangle \updownarrow$, $\langle 0;1W1/0/- \rangle a < v$, $\langle 1;1W1/0/- \rangle a > v$	4/8
CFdrd	$\langle 0;R0/1/0 \rangle \updownarrow$,$\langle 0;R1/0/1 \rangle a < v$, $\langle 1;R1/0/1 \rangle a > v$	4/8
Total	/	55/86

According to literature [9], March SR can detect Write Destructive Faults but cannot detect Deceptive Read Destructive Fault. Write Destructive Faults requires writing without changing state, i.e., 0w0 or 1w1. However, March SR lacks such operations; thus, Write Destructive Faults cannot be sensitized and remain undetectable. In contrast, Deceptive Read Destructive Fault necessitates two consecutive reads post-sensitization, which March SR can detect via dedicated test sequences. The test primitive for Write Destructive Faults is $\langle xWx \rangle \#Rx$. When x=0, the test primitive becomes $\langle 0W0 \rangle \#R0$ with state-tuple $\langle x,0,x \rangle W0\#R0$. When x=1, the test primitive is $\langle 1W1 \rangle \#R1$ with state-tuple $\langle x,1,x \rangle W1\#R1$. Numbering March SR's memory access operations from 1 to 14 reveals operations 1, 5 and 10 meet sensitization for x=0, yet their preceding state-tuples do not match. Operations 3, 8 and 12 satisfy sensitization for x=1 but also lack state-tuple matches, preventing Write Destructive Fault detection. The fault primitive for Deceptive Read Destructive Fault is $\langle Rx \rangle \#Rx$. When x=0 or x=1, the corresponding state-tuple test primitives are $\langle x,0,x \rangle R0\#R0$ and $\langle x,1,x \rangle R1\#R1$, respectively. Specifically, operations 6 and 7 enable Deceptive Read Destructive Fault detection for x=0 by satisfying sensitization and detection requirements with guaranteed state-tuples and symbol matching. Operations 13 and 14 similarly match the test primitive for x=1 Deceptive Read Destructive Fault. In summary, March SR cannot detect Write Destructive Faults but detects Deceptive Read Destructive Fault.

V. CONCLUSION

This paper proposes a novel test primitive . The test primitive functions as a bridge between the fault primitive and the test algorithm, decoupling the cell state, fault sensitization, and detection and forming a highly flexible analysis unit. It is

TABLE VI
FAULT PRIMITIVES DETECTED BY MARCH SR

Fault Model	Detectable Fault Primitives	Fault Coverage
SF	$\langle 1/0/-\rangle$,$\langle 0/1/-\rangle$	2/2
SAF	$\langle \forall/0/-\rangle$,$\langle \forall/1/-\rangle$	2/2
RDF	$\langle R1/0/0\rangle$,$\langle R0/1/1\rangle$	2/2
IRF	$\langle R1/1/0\rangle$,$\langle R0/0/1\rangle$	2/2
TF	$\langle 1W0/1/-\rangle$,$\langle 0W1/0/-\rangle$	2/2
WDF	unable to detect	0/2
DRDF	$\langle R1/0/1\rangle$,$\langle R0/1/0\rangle$	2/2
CFst	$\langle 0;0/1/-\rangle \updownarrow$, $\langle 1;1/0/-\rangle \updownarrow$, $\langle 0;1/0/-\rangle \updownarrow$, $\langle 1;0/1/-\rangle \updownarrow$	8/8
CFrd	$\langle 0;R0/1/1\rangle \updownarrow$, $\langle 1;R1/0/0\rangle \updownarrow$, $\langle 0;R1/0/0\rangle \updownarrow$,$\langle 1;R0/1/1\rangle \updownarrow$	8/8
CFir	$\langle 0;R0/0/1\rangle \updownarrow$,$\langle 1;R1/1/0\rangle \updownarrow$, $\langle 0;R1/1/0\rangle \updownarrow$,$\langle 1;R0/0/1\rangle \updownarrow$	8/8
CFdsrx	$\langle R0;1/0/-\rangle \updownarrow$, $\langle R1;0/1/-\rangle \updownarrow$, $\langle R0;0/1/-\rangle \updownarrow$,$\langle R1;1/0/-\rangle \updownarrow$	8/8
CFdsxw!x	$\langle 0W1;0/1/-\rangle \updownarrow$, $\langle 0W1;1/0/-\rangle \updownarrow$, $\langle 1W0;0/1/-\rangle \updownarrow$, $\langle 1W0;1/0/-\rangle \updownarrow$	8/8
CFdsxwx	unable to detect	0/8
CFtr	$\langle 0;0W1/0/-\rangle \updownarrow$,$\langle 0;1W0/1/-\rangle \updownarrow$, $\langle 1;0W1/1/-\rangle \updownarrow$,$\langle 1;1W0/1/-\rangle \updownarrow$	8/8
CFwd	unable to detect	0/8
CFdrd	$\langle 0;R0/1/0\rangle \updownarrow$, $\langle 1;R1/0/1\rangle \updownarrow$	4/8
Total	/	64/86

anticipated to streamline the analysis process of the March algorithm. For the set of faults addressed in this paper, a test primitive library is provided, and the completeness of the test primitives is demonstrated through proof by contradiction, thus ensuring completeness. The uniqueness and conciseness of the test primitives can also be guaranteed.The test primitives involved in this paper are applied in the analysis process of the March algorithm by expanding the state tuples. The algorithms such as March LR, March LR2, and March SR have been analyzed. In the future, the test primitives will be used in automatically generating the March algorithm to achieve the purpose of simplifying this process.

REFERENCES

[1] X. Du, N. Mukherjee, W.-T. Cheng, and S. Reddy, "Full-speed field-programmable memory bist architecture," in *IEEE International Conference on Test, 2005.*, 2005, pp. 9 pp.–1173.

[2] A. Z. Jidin, R. Hussin, L. W. Fook, and M. S. Mispan, "An automation program for march algorithm fault detection analysis," in *2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS)*, 2021, pp. 149–152.

[3] A. Z. Jidin, R. Hussin, L. W. Fook, M. S. Mispan, N. A. Zakaria, L. W. Ying, and N. Zamin, "Generation of new low-complexity march algorithms for optimum faults detection in sram," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2022.

[4] A. Z. Jidin, R. Hussin, M. S. Mispan, L. W. Fook, and L. W. Ying, "Reduced march sr algorithm for deep-submicron sram testing," in *2022 IEEE International Conference on Semiconductor Electronics (ICSE)*, 2022, pp. 93–96.

[5] G. Harutyunyan, S. Martirosyan, S. Shoukourian, and Y. Zorian, "Memory physical aware multi-level fault diagnosis flow," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 3, pp. 700–711, 2020.

[6] A. van de Goor and Z. Al-Ars, "Functional memory faults: a formal notation and a taxonomy," in *Proceedings 18th IEEE VLSI Test Symposium*, 2000, pp. 281–289.

[7] Z. Al-Ars and S. Hamdioui, "Fault diagnosis using test primitives in random access memories," in *2009 Asian Test Symposium*, 2009, pp. 403–408.

[8] Z. Al-Ars, S. Hamdioui, and A. van de Goor, "A fault primitive based analysis of linked faults in rams," in *Records of the 2003 International Workshop on Memory Technology, Design and Testing*, 2003, pp. 33–39.

[9] N. A. Zakaria, W. Hasan, I. Abdul Halin, R. Sidek, and X. Wen, "Testing Static Single Cell Faults using static and dynamic data background," in *2011 IEEE Student Conference on Research and Development*, 2011, pp. 1–6.

面向目标检测模型的高效能安全可重构处理器

常立博[①②] 范毓洋[①] 张嘉诚[①]

[①]中国民航大学民航航空适航审定技术重点实验室，天津 300300

[②]西安邮电大学电子工程学院，西安 710121

摘 要：针对应用于终端设备中基于神经网络的目标检测处理器，需要同时满足高效能和计算结果可信的要求，本文提出一种可高效加速轻量化神经网络，且可保证推理结果可信和防御参数泄露的可重构同构处理器。首先，采用粗粒度可重构技术，设计了可满足“安全白片”要求的全同构处理器架构；然后，提出一种针对深度可分离卷积运算的流水式可重构多核计算阵列，以提高计算效能；接着，提出基于拟态安全策略的动态数据缓存策略，和基于冗余运算的自动检测运算错误的方法；最后，在 Ultra96-V2 可重构平台上分别测试了处理器的安全性和计算效率。实验结果表明，本文提出的可重构处理器即可自动检测结果是否可信，又可防御已有最新防御参数截取攻击。并且，吞吐率和计算效率可分别达到 215.8 GOPS 和 0.62 GOPS/DSP，相较于其它基于 FPGA 平台的目标检测加速器可达到更高的计算效率和检测精度。

关键词：目标检测；可重构加速器；容错计算；神经网络加速器

An Efficient and Secure Reconfigurable Processor for Object Detection

CHANG Libo[①②] FAN Yuyang[①] ZHANG Jiacheng[①]

[①]Key Laboratory of Civil Aircraft Airworthiness Technology, Civil Aviation University of China, Tianjing, 300300, China

[②]School of Electronic Engineering, Xi'an University of Posts and Telecommunication Xi'an, 710121, China

Abstract: To meet the requirements of high efficiency and reliable computing results, a reconfigurable processor is proposed which can accelerate lightweight neural networks, ensure reliable inference results, and protect against parameter leakage. Firstly, the processor architecture based on CGRAs is used to design a secure processor that can meet the "security white chip". Then, a pipelined reconfigurable multi-core computing array for depth-separable convolution operation is proposed to improve computing efficiency. Then, a dynamic data caching strategy based on mimicry security strategy is proposed, and an automatic detection method based on redundancy operation is proposed. Finally, the security and computational efficiency of the processor are tested on Xilinx Ultra96-V2. The experimental results show that the reconfigurable processor proposed in this paper can self-detect whether the results are reliable or not, and can also defend against the latest defense parameter interception attack. Moreover, the throughput rate and computational efficiency can reach 215.8 GOPS and 0.62 GOPS/DSP respectively, which can achieve higher computational efficiency and detection accuracy compared with other object detection accelerators based on FPGAs.

Key words: Object Detection; Reconfigurable Accelerator; Fault-tolerant Computing; Neural Network Accelerator

1. 引言

近年来，随着以基于深度神经网络(Deep Neural Networks, DNNs)模型的智能算法在图像分类等应用领域取得巨大进展[1]，基于 DNNs 的目标

*通信作者：常立博 changlibo@xupt.edu.cn

基金项目：国家重点研发计划（2019YFB1803600），中国民航适航中心开放基金（SH2021111903），陕西省教育厅重点科学研究计划(22JY059)

Foundation Item：National key research and development program（2019YFB1803600），Open Fund of China Civil Aviation Airworthiness Center （SH2021111903），Shaanxi Provincial Department of Education Key Scientific Research Program(22JY059)

检测算法的精度也获得了极大提高[2]。当前基于DNNs 的经典目标检测算法依赖计算复杂度高、且模型参数量巨大的深度卷积网络模型（如VGGNet、ResNet 等）以实现高效特征提取的功能[3]。然而，面向终端应用的嵌入式系统可提供的计算和存储资源非常有限，并要求系统功耗限制在数瓦之内。因此，当面对基于嵌入式终端设备的应用场景时，已有基于 DNNs 的目标检测算法和通用计算平台则无法同时满足高精度、实时性并且低功耗的要求。与此同时，如应用于无人机等终端系统对智能计算平台安全性也有较高的要求，需要确保计算结果可信并且可防御模型参数泄露。因此，如何在有限的计算资源和功耗约束下，提高智能目标检测算法计算能效和安全性，正成为人们越来越关注的问题[4-6]。

目前，结合应用场景的需求，采用软件/硬件协同(software/hardware co-design)的方法，设计对应的计算系统架构是提高深度神经网络模型加速器计算效率的重要途径[7]：首先采用参数量化、模型稀疏以及轻量化模型等方法降低模型的计算复杂度，然后设计可适应其特定计算模式的加速器[8]。然而其通用性受限，往往只能支持固定的一种计算模式，很难根据神经网络结构和执行目标不同来改变其计算架构，因此无法适应终端系统场景多样化的要求。相比于传统的仅仅采用指令流驱动或者数据流驱动的计算技术，可重构计算则通过将两者相结合的策略，因此既可以利用指令流驱动方法以提高处理器的功能灵活性，又可以利用数据流驱动方法以提高处理器的能量。基于可重构器件实现的深度神经网络模型加速器对通用处理器平台和基于ASIC 的专用处理器进行了折中，因此其在性能、功耗和功能灵活性等芯片的关键指标之间可以达到很好的平衡[9]。

与此同时，理论分析和实验结果都表明基于深度学习算法的处理器存在安全问题[10]，并且可通过利用深度神经网络模型加速器特定的体系结构和工作模式，可以很容易植入木马并可成功攻击 DNNs 加速器[11]。虽然，已有大量学者设计了相对应的防御机制[6]，然而，目前的方法仍采用针对通用硬件芯片的被动防御策略，因此不但需要消耗不必要的计算和存在资源，而且会因为攻击者掌握了其防御策略，从而成功攻击此类加速器。采用可重构计算技术实现的处理器，因其硬件可随软件变化而变化，即芯片的功能与电路结构非直接相关，所以此类处理器芯片可达到“安全白片”的效果，从而增加了植入硬件木马的难度；其次，通过设计不同的算法到计算引擎的映射方法，从而在计算过程中可随机改变算法处理流程，所以基于可重构计算的智能动态映射算法可实现拟态安全的功能[12]。本文针对以无人机应用为代表的嵌入式系统对高效能和计算结果可信的目标检测系统的应用需求，设计了一种可加速基于深度可离卷积的轻量化目标检测模型，并通过增加冗余计算单元和动态机制的高效且结果可信的处理器，本文的主要工作和创新点：

1. 为了解决专用加速器和异构处理器容易受到硬件木马等攻击，及细粒度同构处理器计算效率低的问题，本文针对深度可分离卷积的目标检测模型计算特征，设计粗粒度可重构处理器架构，从而即可通过满足“安全白片”的要求以提高处理器的安全性，又可通过设计专用计算单元以提高处理器的效率；

2. 为了提高处理器的计算效率，设计可以流水模式完成深度可分离卷积运算的计算单元。通过设计可以流水完成卷积、池化、激活，点卷积等运算的宏运算单元，以及可根据模型结构的可重构多核计算阵列，以提高加速轻量化目标检测模型的计算效率；

3. 为了提高处理器的安全性，提出基于“Tile”的流处理器架构及基于拟态安全策略的动态数据缓存策略，和基于冗余运算的自动检测方法。通过在计算阵列中增加同构冗余计算单元，从而可动态重计算部分结果以提高计算结果的可信度；通过在访存单元中增加随机地址映射策略，从而防御模型参考窃取攻击。

2. 研究背景和动机

2.1 轻量化目标检测算法

目前，已有采用针对物体检测而设计轻量化网络，如与 YOLO 算法对应的轻量化网络模型 YOLOv2-tiny[13]，YOLOv3-tiny[14]，YOLO Nano[15]，基于 MobileNet 和 ShuffleNet 的的轻量化网络模型 ShuffleNet-V2[16]，以及基于 SSD[17]目标检测框架的轻量化的物体检测算法--MobileNetV2+SSDLite[18]。其中，采用深度可分离卷积代替普通卷积运算，可极大降低模型规模和计算复杂度，然而，已有轻量化网络设计仍存在精度不高或压缩比例较低的问题。目前，可采用知识蒸馏训练方法将基于复杂网络的教师网络提取特征的能力迁移到轻量级的学生网络[19, 20]，从而通过深度学习网络训练的方法提高轻量化网络模型的性能且不增加物体检测算法计算复杂度。

Libo Chang 等基于深度可分离卷积的设计了一种轻量化目标检测模型，并采用知识蒸馏训练方法提高检测精度[21]，与主流目标检测算法--YOLOv3 相比，针对 COCO 数据集，在检测精度下降 7.4 mmAP 的代价下，实现了将其模型规模压缩到标准 YOLOv3 模型的 1/11，其目标检测模型如图 1 所示，轻量化目标检测网络分为基础卷积神经网络和分支预测网络这两个部分，轻量化的基础卷积神经网络中卷积模块的 表示步长为 1， 表示输出特征映射的通道数为 N：基础卷积神经网络中红色的卷积模块的步长为 1，输出通道数与输入通道数相同；蓝色的卷积模块的步长为 2，输出通道数是输入通道数的 2 倍；绿色的卷积模块，其步长为 1，输出通道数为输入通道数的 2 倍。主要包括步长为 1 并且通道数不变的轻量化卷积模块，以及步长为 2 并且通道数增加一倍的轻量化卷积模块。其中，步长为 1 的轻量化卷积模块由卷积核尺寸为 3x3 的深度卷积层与一个 1x1 的卷积层以及旁路组成，因此需要设计可支持深度可分离卷积运算的计算单元，才能提高目标检测系统的计算效率。

2.2 高性能安全 CNNs 处理器研究

虽然深度神经网络模型的计算模式相对规则和固定，但是不同的模型之间甚至同一模型中不同层之间，计算模式和数据访存模式方面也有较大的差异。因此可利用 CGRA 平台具备的可重构特性，设计可适应不同模型计算特性的处理器。L.Song 等提出一种可实现针对 CNN 网络模型中不同类型数据级并行的加速器[22]，其可以自适应地切换不同类型的并行性和相应的数据拼接方法，并且可以动态匹配不同卷积计算任务，与固定计算模式的深度神经网络模型加速器相比，可以实现 4 到 8.3 倍的加速，并且片上缓存访问的能耗平均下降了 90.3%。W.Lu 等提出了一种可支持不同并行计算模式的灵活数据流架构--FlexFlow[23]，从而缓解因计算模式与被加速的 CNN 网络模型负载不匹配引起的计算性能下降的问题，与同时期三种最先进的加速器架构相比，获得了 2 到 10 倍的性能提升和 2.5 倍到 10 倍的能效提升，并且其具有较好的可扩展性。为了适应不同模型对片上缓存器的差异化需求，并提高数据重用率以降低数据访存开销，A.Azizimazreah 等提出了一种可根据不同数据缓存模式的特性重构片上存储器结构和访存策略的可重构深度神经网络模型处理器[24, 25]。针对面向目标检测等算法的特定深度神经网络模型，已有学者设计了基于 FPGA 平台的专用加速器。如基于 YOLOv2 的目标检测算法，H. Nakahara 等[26]首先通过算法轻量化技术压缩模型，然后针对算法特性设计了对应的加速器，实现了对 Lightweight YOLOv2 加速。C. Ding 等[8]首先对目标检测算法进行量化以降低计算复杂度，然后基于 FPGA 平台设计了可适应不同神经网络层的计算任务的可配置加速器。

为了进一步提高针对神经网络模型的加速系统计算效率，已有学者采用软件/硬件协同的方法，首先采用参数量化、模型稀疏以及轻量化模型等方法降低模型的计算复杂度，然后设计可适应其特定

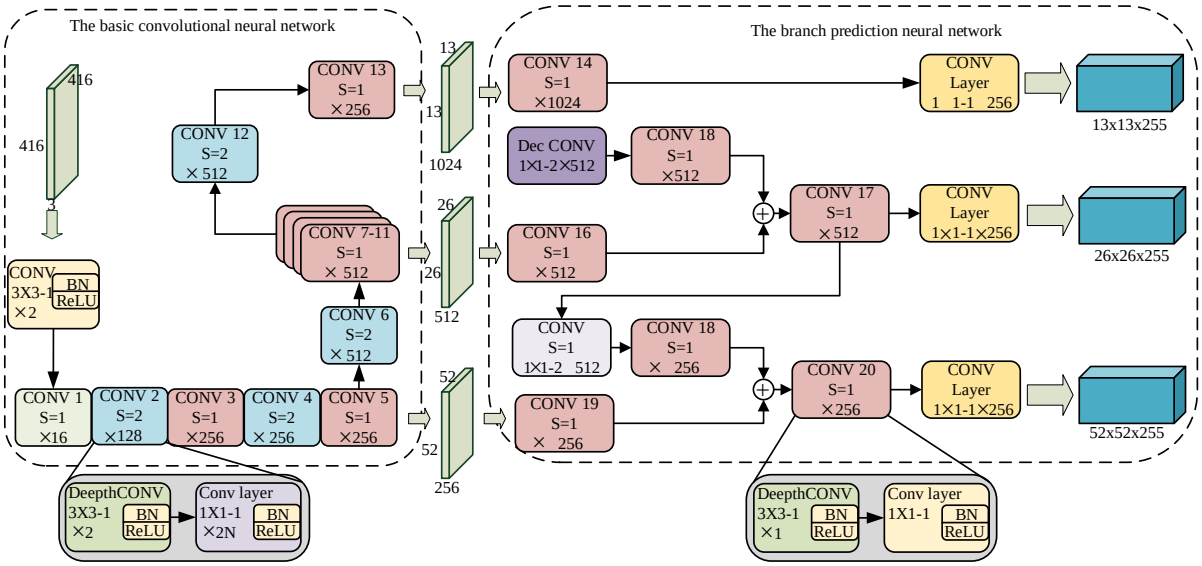


图 1 轻量化目标检测网络结构图

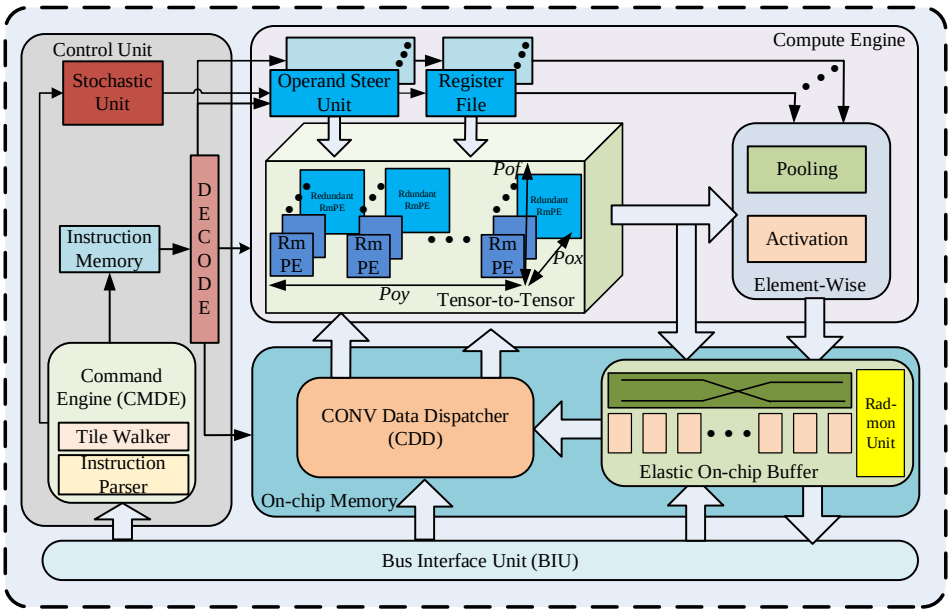


图 2 流水线多核单元整体架构

计算模式的可重构处理器。如 S.Yin 等[27]基于 CGRA 结构实现了一种可重构的混合神经网络模型处理器，通过支持 8bit/16bit 数据位宽自适应运算单元、多端口片上缓存以及同时支持多种计算流程，其性能可达到 5.09 TOPS/W。同时，Libo Chang 等[28]将模型量化与可重构处理器处理设计相结合，采用基于强化学习智能模型，以权衡量化精度、模型规模和加速器计算延时的多目标优化算法，从而提高处理器性能。与此同时，为了提高面向神经网络模型的处理器的安全性，Cheng Liu 等[6]基于重计算的容错策略和异构容错处理器架构，通过设计专用的重计算单元以完成所有故障 PE 的重计算任务，然而其必须通过自己检测程序以识别故障 PE，因此很难应对特定人为攻击。

3 面向目标检测的高效安全处理器

为了提高面向卷积等计算密集型运算处理器的安全性，本文在已有可支持混合精度运算的流水式高效能多核处理器[29]的基础上，通过采用基于同构运算单元的处理器结构，以实现“安全白片”，从而增加处理器芯片制造过程中植入木马的难度。与此同时，引入空间随机性并随机改变冗余计算映射和片上数据缓存地址，从而防御故障攻击。如图 2 所示，处理器由支持可变精度的卷积、激活、池化等操作的计算引擎，可支持弹性划分与动态重组的缓存单元以及可支持多指令乱序发射及执行的控制单元组成，并且计算和控制数据通过片上总线接口单元与外部完成数据交换。然而，本文在控制单元中增加随机单元—Stochastic Unit，以控制计算和数据映射；在计算阵列中增加冗余计算单元—Redundant RmPE，以检测故障运算单元；在弹性片上缓存器中增加地址随机映射单元—Radnom Unit，以动态改变相同数据的物理存储地址。

4. 面向深度可分离卷积运算的容错计算单元

本文采用流水式多核加速器单元，将卷积、点卷积、池化和激活等运算以流水的方式实现，以提高基于深度可分离卷积的轻量化目标检测模型计算效率。并且采用簇状分层结构进行互联，以提高系统的可扩展性，其结构和计算数据流如图 3 所示：多个可完成混合精度乘法运算的可重构处理部件(reconfigurable micro processing element，RmPE)组成一个可重构微处理单元(reconfigurable micro processing units，RmPU)，多个 RmPU 组成一个计算引擎并通过在不同 RmPU 上映射不同卷积层以实现层间流水。为了提高乘法运算单元的利用率，本文利用乘法分割原理并结合 FPGA 器件特性，设计了支持 2-8bit 内任意精度并行乘加运算的 RmPE，并且利用乘法运算结果的符号位与计算无关的特性，通过分离符号位的方法提高乘法运算的并行度[17][23]。

根据轻量化目标检测模型的结构特点和可重构计算平台的资源约束，本文首先从 Pox 和 Poy 两个维度扩展如图 3 所示的 RmPU 单元，然后将多个 RmPU 单元通过在 Pof 维度进行扩展，以适应不同层的卷积计算量和卷积计算模式。在执行卷积等运算中，RmPU 首先从输入缓存和权重缓存中读取数据，以并行计算的模式完成卷积计算，然后将计算

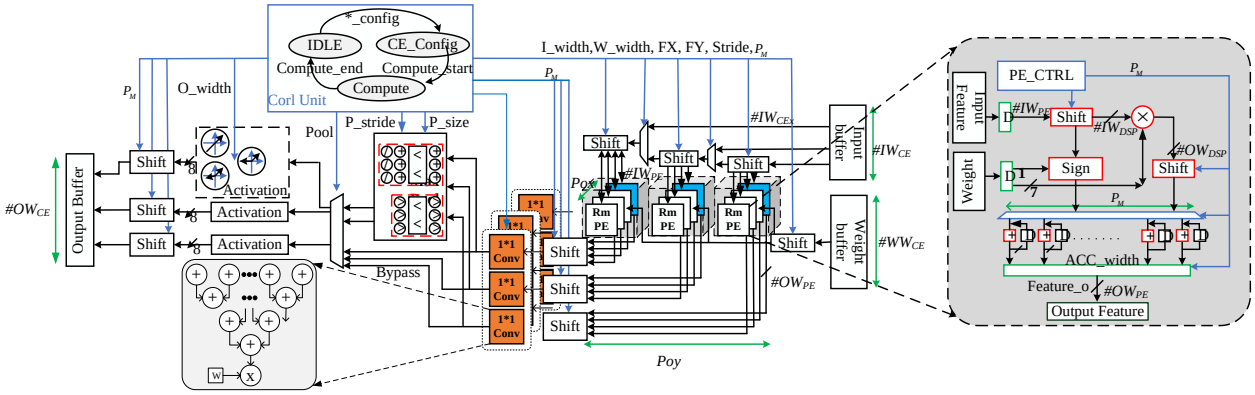


图 3 流式运算单元结构

结果重组并以流水模式完成激活运算，最后将运算结果以并行方式存入输出缓存，计算流程中的数据通路选择由控制单元管理。与此同时，为了适应轻量化目标检测模型中不同层的计算特性及不同数据位宽，本文通过定义 RmPE 中对应的可重构参数，从而可根据不同混合精度量化网络模型定制计算单元的规模，并提升计算性能的并提高计算资源利用率，图 3 中各参数的定义如文献[29]所示。为了提高如图 1 所示的深度分离卷积运算中的点卷积操作的计算效率，本文则增加了专用于实现点卷积的运算单元。由于点卷积完成一个 3*3 卷积运算数据累加及与权重相乘的操作，因此本文采用分层树累加结构，共设置 9 个累加单元及一个乘法单元。并且，为了满足基于同构体系结构以实现“安全白片”的需求，将所有加法器的运算数据位宽设定为固定值(如 16 bit)。

5. 基于随机地址的拟态安全数据访问策略

由于神经网络模型的计算模式固定，所以其参数的存储地址和访问方式通常是确定的，从而攻击者可通过监听地址以分析访问外存的数据，以获得模型参考等。因此，本文设计了一种动态存储资源分配和基于随机的动态数据访问方法，以防御基于地址监听的攻击。

5.1 动态存储资源分配策略

本文从 RmPE 阵列的 *Pox* 、 *Poy* 以及 *Pof* 三个维度进行重构，以提供计算资源的利用率，因此构成输入特征存储区域的存储块在位面方向的数目需要匹配 *Pox* ，从而为 RmPE 阵列提供 *Poy* 行的并行输入；构成权重以及输出特征存储区域的存储块在位面方向的数目需要匹配 *Pof* ，从而满足 RmPE 阵列在输出通道方向上的并行度。为了简化存储单元的重构逻辑，片上数据存储数据位宽采用 4 bit 和 8 bit 两种模式，即混合精度 CNNs 中小于 4 bit 的数据会按照 4 bit 编码，大于 4 bit 小于 8 bit 的数据会按照 8 bit 编码。由于特征映射尺寸的变化会影响“Tile”的划分，因此“Tile”尺寸以及数据位宽共同决定了弹性存储架构中存储块的重构方式。在基于 FPGA 等已有可重构平台中，片上 SRAM 通常由固定大小的 RAM 块组成(如在 Xilinx 的 FPGA 芯片中，片上的 SRAM 资源由 36kb 大小的 Block RAM 组成)，因此硬件加速核的弹性存储架构中，一个存储块即为一个 Block RAM，则片上存储块总数 B 可以表示如式(1)所示：

$$B \geq B_{if} + B_{w} + B_{of} \quad (1)$$

其中，Bif、Bw 以及 Bof 分别为输入特征数据、权重数据以及输出特征数据使用的存储块数目，根据上述的划分原则，三种数据各自的存储块计算方法如式(2)、式(3)和式(4)，由于存储块数目是整数，因此各部分的计算结果均向上取整。

$$B_{if} = \lceil C_{pif} \times \#IW_f / \#W_b \rceil \times \lceil \lceil C_{if} / C_{pif} \rceil \times \lceil R_{if} / n_{PEy} \rceil \times N / \#H_b \rceil \times n_{PEy} \times 2 \quad (2)$$

$$B_{of} = \lceil C_{pof} \times \#OW_f / \#W_b \rceil \times \lceil \lceil C_{of} / C_{pof} \rceil \times R_{of} / \#H_b \rceil \times n_{PEz} \times 2 \quad (3)$$

$$B_{w} = \lceil K_x \times K_y \times \#IW_w / \#W_b \rceil \times \lceil N / \#H_b \rceil \times n_{PEz} \times 2 \quad (4)$$

其中，C_{if} 和 C_{of} 分别为输入特征映射以及输出特征映射中对应 Tile 的列方向尺寸；C_{pif} 和 C_{pof} 分别为卷积运算单元在特征映射列方向上的并行计算尺寸；R_{if} 和 R_{of} 分别为 Tile 块的在输入特征映射以及输出特征映射中行方向尺寸；$\#IW_f$ 、$\#IW_w$ 、$\#OW_f$ 分别表示输入特征映射中激活值，权重参数，以及输出特征映射中激活值的数据位宽；$\#W_b$ 和 $\#H_b$ 分别表示存储块的宽度和深度。

在获得各运算单元需要的存储容量之后，则可以将其动态映射到某些存储块，从而无法通过监听访存地址来获得模型数据。

5.2 基于随机数据访问方法

本文将片上存储单元划分为多个独立的“bank”块以提高数据访问的并行度，控制器根据各“bank”块的状态和“bank”划分策略，以动态重组的方式将“bank”块指定为特定的 RmPU 缓存单元。与此同时，为了提高计算的并行度，本文采用文献[30]所示的基于“Tile”块的细粒度任务并行策略，因此为每一个特征映射的 Tile 块分配若干个“bank”块的缓存空间；在计算过程中，计算引擎根据 Tile 的索引信息完成特征块的读出或写入。

然而，由于神经网络模型计算模式固定，在模型推理运算过程中，其通常采用固定模式读写相同数据，所以攻击者则可以通过监测 CNN 处理器访问数据总线以推测其计算模式，并可精确激活硬件木马[11]，从而影响模型的计算结果。因此，本文采用逻辑地址和数据物理地址分离的数据访问策略，以隐藏数据访问模式。并且，在地址映射的过程中，增加基于伪随机的地址生成器，使得神经网络模型的数据访问模式具有了随机性和动态性，从而防御基于地址监测的模型攻击。

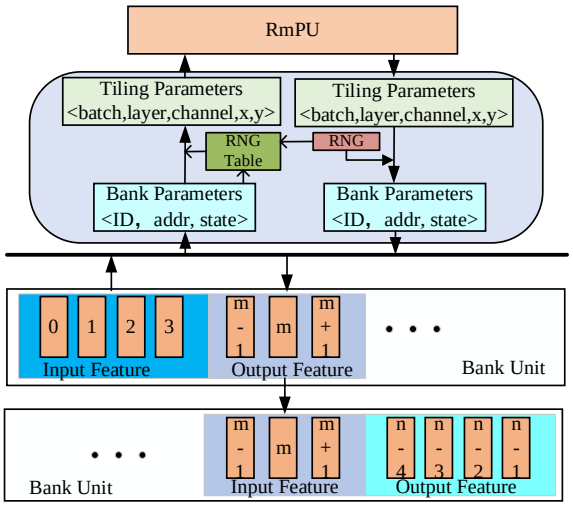


图 4 基于随机地址映射的数据访问策略

如图 4 所示，卷积计算单元以“Tile”为基本处理单元，因此激活值(Activation)以“Tile”块为单元整体存储。每一个 RmPU 读入或写出的特征值均以“Tile”为整体块存储到片上“Bank”中，每一个特征映射“Tile”块 ID 将由 Batch 号、Layer 号、特征层中 Channel 号、“Tile”的坐标以及尺寸组成。同时，每一个存储“bank”块由确定的索引号标识，从而可根据控制模块产生的片上缓存划分策略，本文通过动态重组的方法将多个“bank”块组成特定的缓存区（如输入缓存区，输出缓存区和权重缓存区）。

在进行卷积等运算时，根据如图 2 所示的控制单元产生的片上缓冲区划分策略，将多个“bank”被动态分组为一组以形成缓冲区（例如，输入缓冲区、输出缓冲区缓冲区，或权重缓冲区）。与此同时，在将基于“Tile”编号的逻辑地址映射到基于存储“Bank”块号的物理地址过程中，将基于随机数发生器(random number generator, RNG)产生的随机数加入到地址映射中，并将对应的随机地址记录到 RNG Table 单元中，从而计算单元读取对应“Bank”块的数据时，可以反向自动去除随机数，从而正确读取到缓存数据。

6. 评估与分析

6.1 实验设置

本文采用文献[28]提出的自动搜索最优参数位宽的混合精度量化策略，将文献[21]所示的轻量化目标检测模型—PaLiDet-D 进行参数量化，以适应终端平台对计算复杂度和参数量的限制。表 1 展示了混合精度量化后轻量化目标检测模型的参数量相较于原始网络模型参数规模的变化，以及基于 COCO 标准测试数据集下的量化精度损失。由表 1 可知，在将权重和特征值分别压缩到 3-8bit 和 6-8bit，将精度控制在 44.0 mAP 的情况下，针对 PaLiDet-D 网络的压缩比为 5.3。

表 1　轻量化模型的量化结果

类目	PaLiDet-D
原始模型(MB)	22.3
压缩模型(MB)	4.2
压缩比	5.3
精度(mAP)	44.0
权重位宽(bit)	3-8 MP 5.31
特征值位宽(bit)	6-8 MP 6.8

针对混合精度 PaLiDet-D 模型的结构特点，我们通过重构如图 2 所示的计算引擎中 RmPU 的阵列结构以及片上缓存单元包含的存储块数量，将本文的加速器架构分别映射到针对嵌入式端的 FPGA 平台--Ultra96-V2。从如表 2 所示的各类资源使用情况可以看出，为了减少布局布线的难度，将 LUTs 和 FFs 资源使用量控制在 75%以内，同时为了提高计算性能，加速器中 DSP 的使用率控制在 90%以上。加速器在 Xilinx Vivado 2022.1 环境下实现，其中流水式多核加速器单元在异构 FPGA 的可编程逻辑(programable logic，PL)上完成，而片外数据传输和任务调度等在异构 FPGA 的处理系统(processing system，PS)上完成，片上互联采用基于 AXI 4.0 的

表 2 不同平台加速器性能对比

	[29]	[31]	[32]	[8]	本文
FPGA 平台	Xilinx ZCU102	Zedboard	Pynq-z2	Virtex-7 690t	Ultra96-V2
频率(MHz)	200	--	125	200	200
目标检测模型	YOLOv3	Tiny YOLOv3	YOLOv2	Tiny YOLOv2	PaLiDet-D
参数位宽(W, A)	(MP5.98，MP7.8)	(16，16)	(16，16)	(6，32)	（MP 5.31，MP 6.8）
DSP	2315	160	153	3348	347
BRAMs(36kb)	680	185	440	1323	202
FFs(K)	308K	60K	36	494K	98.56
LUTs(K)	185K	25.9K	38	430K	51.22
GOPS	809	10.45	--	1449	215.8
GOPS/DSP	0.36	0.06	--	0.43	0.62

总线实现。

本文以一定的概率随机将某些计算单元—RmPE 结果强制赋值为零，以测试可重构处理器能否自动检测到因硬件木马等攻击，如图 5 所示，本文逐步增加错误单元比例(0%~100%)，并且分别采用不同的错误单元分布的方差，即三组测试案例的方差分别为 0.03、0.1 和 0.15。与此同时，本文采用文献[11]所示的硬件木马触发及总线数据监测方法，并假设攻击者已知使用的 PaLiDet-D 模型结构，即攻击者已知模型所有参考的逻辑地址，攻击者则通过监测数据总线以获得模型的参考。然而由于本文提出的随机地址映射方法，我们将测试其能否防御此类攻击，具体测试结果及如 6.2 节所示。

6.2. 容错代价和能力评估

本文通过在计算单元中增加冗余计算单元—Redundant RmPE，从而在推理运算过程中，采用此单元随机重计算某个 RmPE 的运算结果，以检测故障运算单元，因此共需要增加 Poy*Pof 个 RmPE 单元。如基于 Ultra96-V2 平台，本文的可重构计算阵列中的 Pox、Poy、Pof 分别为 4、7、12，因此共增加 48 个 DSP 运算单元。在弹性片上缓存器中增加地址随机映射单元—Radnom Unit，以动态改变相同数据的物理存储地址。因此共增加 Pof 个 RNG 单元，其成本约为 9.7K LUT（占总 LUT 的 19%）和 12 K FFs（占总 FFs 的 12%）。

本文随机的将错误注入一部分 RmPE 单元中，以验证冗余单元可以检测到结果不可信的概率。如图 5 所示，随着注入的错误的增加，即故障 RmPE 单元的比率提高，通过冗余比较以自动发现错误的概率则以线性趋势增长，当故障 RmPE 单元的比例大于 80%时，不管计算错误单元的分布情况如何，自动发现错误的概率均接近 100%。因此，可以通过重复推理运算的方式，以增加计算结果的可信度。与 HyCA[6]相比，本文通过设计的随机冗余计算的方法来检测运算单元是否发生错误，而不用运算标准测试案例，从而增加了系统的适应性。

由于本文以随机的模式访问片上和片外存储，因此在运行约 12 个小时，通过数据总线共截取了约 4.21GB 的数据（即约 1000 倍模型规模），但未得到模型的参数，从而证明了本文设计的随机地址访问策略，可防御如文献[11]所示的攻击。

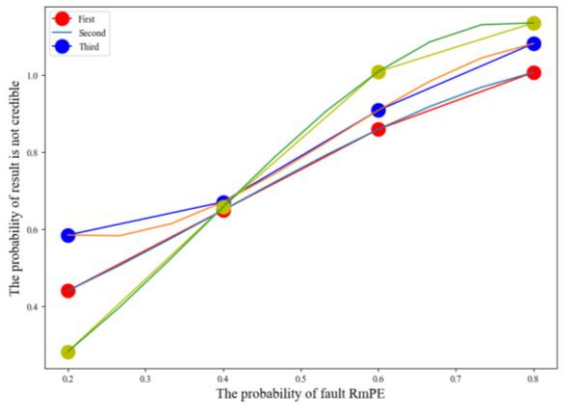


图 5 基于随机冗余计算发现计算结果不可信的概率

6.3 计算性能分析及与相关研究对比

将本文设计的目标检测加速器与其它基于 FPGA 平台的目标检测加速器进行性能对比，各加速器均通过量化 CNN 模型的参数提高系统实时性。为了对比的公平性，把加速器计算效率 GOPS/DSP 作为加速器性能的统一衡量标准。对比结果如表 2 所示，由于片上缓存资源的限制，相较于文献[29]和[8]计算效率较低，但是计算效率分别提高了 72%和 44%；并且，针对复杂测试集 COCO 的精度可达到 44 mAP，可满足现代嵌入式系统对智能算法的性能要求；由于本文针对轻量化 CNNs 模型的计算特性，增加了特定的计算和缓存单元，因此，虽然本文为了提高计算结果的可信度，增加了冗余计算单元，但是仍可达到较高的计算效率。与文献[31]采用片上资源更少实现的加速器相比，虽然使用的 DSP 和 LUT 资源分别为文献[31]的 2.16 倍和 1.98 倍，但本文的计算性能及计算效率可分别了 20.65 倍和 10.33 倍。可以看出，为了提高处理器的安全性，本文在处理器中增加了冗余计算

和动态地址映射电路，但是仍可以满足嵌入式系统对计算性能的需求。

7 总结

本文针对嵌入式平台要求处理器满足计算结果可信并计算效率高的要求，设计了一种可重构加速器，采用流水式众核体系架构，以增加计算的数据级并行度。与此同时，提出基于拟态安全策略的动态数据缓存策略，和基于冗余运算的自动检测方法。从实验结果可以看出，本文提出的可重构处理器既可自检测结果是否可信，又可防御已有最新防御参数截取攻击。并且，吞吐率和计算效率可分别达到 215.8 GOPS 和 0.62 GOPS/DSP，可基本满足无人机等终端系统对智能处理器在安全性和计算效率等方面的要求。

参 考 文 献

[1] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. nature, 2015, 521(7553): 436-444.

[2] ZOU Z, SHI Z, GUO Y, *et al*. Object detection in 20 years: A survey[J]. 2019, arXiv:1905.05055.

[3] REDMON J, FARHADI A. Yolov3: An incremental improvement[J]. 2018, arXiv:1804.02767.

[4] SHRIVASTAVA N, SARANGI R. Securator: A Fast and Secure Neural Processing Unit, 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Montreal, QC, Canada, 2023, pp. 1127-1139, doi: 10.1109/HPCA56546.2023.10071091.

[5] SADI M, GUIN U. Test and Yield Loss Reduction of AI and Deep Learning Accelerators, in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 1, pp. 104-115, Jan. 2022, doi: 10.1109/TCAD.2021.3051841.

[6] LIU C, *et al*. HyCA: A Hybrid Computing Architecture for Fault-Tolerant Deep Learning[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 10, pp. 3400-3413, Oct. 2022, doi: 10.1109/TCAD.2021.3124763.

[7] LEISERSON E, THOMPSON C., EMER S, *et al*. There's plenty of room at the Top: What will drive computer performance after Moore's law?[J]. Science, 2020, 368(6495): eaam9744.

[8] DING C, WANG S, LIU N, *et al*. REQ-YOLO: A resource-aware, efficient quantization framework for object detection on FPGAs[C]. The 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Seaside, CA, USA, 2019: 33-42.

[9] 魏少军, 刘雷波, 尹首一. 可重构计算处理器技术[J]. 中国科学: 信息科学, 2013, 42(12): 1559-1576.
WEI SHAOJUN, LIU LEIBO, YIN SHOUYI. Key techniques of reconfigurable computing processor[J]. Scientia Sinica(Informationis) , 2013, 42(12): 1559-1576 (in Chinese).

[10] XU D. *et al*. Reliability Evaluation and Analysis of FPGA-Based Neural Network Acceleration System[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 3, pp. 472-484, March 2021, doi: 10.1109/TVLSI.2020.3046075.

[11] PENG L and RUI H. Int-Monitor: a model triggered hardware trojan in deep learning accelerators[J]. J. Supercomput. 79, 3 (Feb 2023), 3095–3111. https://doi.org/10.1007/s11227-022-04759-y

[12] LIU L, ZHOU Z, WEI S, ZHU M, *et al*. DRMaSV: Enhanced Capability Against Hardware Trojans in Coarse Grained Reconfigurable Architectures[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 4, pp. 782-795, April 2018, doi: 10.1109/TCAD.2017.2729340.

[13] REDMON J, FARHADI A. YOLO9000: better, faster, stronger[C]. The 2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Honolulu, HI, USA, 2017: 7263-7271.

[14] REDMON J, FARHADI A.. Yolov3: An incremental improvement[J]. 2018, arXiv:1804.02767.

[15] WONG A, FAMUORI M, SHAFIEE J., *et al*. YOLO Nano: a Highly Compact You Only Look Once Convolutional Neural Network for Object Detection[C]. 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition, Vancouver, British Columbia, Canada, 2019: 22-25.

[16] MA N, ZHANG X., ZHENG T, *et al*. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]. The 15th European Conference on Computer Vision. Munich, Germany, 2018: 116-131.

[17] LIU W, ANGUELOV D, ERHAN D, *et al*.. Ssd: Single shot multibox detector[C]. The 14th European Conference on Computer Vision. Amsterdam, Netherlands, 2016: 21-37.

[18] SANDLER M, HOWARD A., ZHU M, *et al*. Mobilenetv2: Inverted residuals and linear bottlenecks[C]. The 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA, 2018: 4510-4520.

[19] HINTON G, VINYALS O, J. Dean. Distilling the knowledge in a neural network[J]. 2015, arXiv:1503.02531.

[20] ROMERO A, BALLAS N, KAHOU E, *et al*. Fitnets: Hints for thin deep nets[J]. 2014, arXiv:1412.6550.

[21] CHANG L, ZHANG S, DU H, *et al*. Position-aware lightweight

object detectors with depthwise separable convolutions[J]. Journal of Real-Time Image Processing, 2021, 18(3): 857-871.

[22] SONG L, WANG Y, HAN Y, *et al*. C-Brain: A deep learning accelerator that tames the diversity of CNNs through adaptive data-level parallelization[C]. The 53rd Annual Design Automation Conference, San Francisco, CA, USA, 2016: 1-6.

[23] LU W, YAN G., LI J, *et al*. Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks[C]. The 23th IEEE International Symposium on High Performance Computer Architecture. Austin, Texas, USA, 2017: 553-564.

[24] AZIZIMAZREAH A, CHEN L. Shortcut mining: Exploiting cross-layer shortcut reuse in dcnn accelerators[C]. The 25th IEEE International Symposium on High Performance Computer Architecture. Washington, DC, USA, 2019: 94-105.

[25] AZIZIMAZREAH A, CHEN L, Polymorphic Accelerators for Deep Neural Networks[J]. IEEE Transactions on Computers, 2022, 71(3): 534-546.

[26] NAKAHARA H，YONEKAWA H, FUJII T., *et al*. A lightweight YOLOv2: A binarized CNN with a parallel support vector regression for an FPGA[C]. The 2018 ACM/SIGDA International Symposium on field-programmable gate arrays, Monterey, CALIFORNIA, USA, 2018: 31-40.

[27] YIN S, OUYANG P, TANG S, *et al*. A high energy efficient reconfigurable hybrid neural network processor for deep learning applications[J]. IEEE Journal of Solid-State Circuits, 2017, 53(4): 968-982.

[28] CHANG L, ZHANG S., DU H, *et al*. Accuracy vs. Efficiency: Achieving both Through Hardware-Aware Quantization and Reconfigurable Architecture with Mixed Precision[C]. The 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, 2021: 151-158.

[29] CHANG L, ZHANG S, DU H, *et al*. A Reconfigurable Neural Network Processor With Tile-Grained Multicore Pipeline for Object Detection on FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(11): 1967-1980.

[30] JIANG W., ZHANG X., SHA M., *et al*. Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search[C]. The 56th ACM/IEEE Design Automation Conference, Las Vegas, Nevada, USA, 2019: 1-6.

[31] YU Z, BOUGANIS S. A parameterisable FPGA-tailored architecture for YOLOv3-tiny[C]. International Symposium on Applied Reconfigurable Computing (ARC). Toledo, Spain: Springer, 2020: 330-344.

[32] BAO C, XIE T, FENG W, *et al*. A power-efficient optimizing framework FPGA accelerator based on winograd for YOLO[J]. IEEE Access, 2020, 8: 94307-94317.

常立博：男，博士，讲师，CCF 容错专委通信委员，研究方向为微处理器体系结构及容错深度学习加速器设计

范毓洋：男，硕士，讲师，CCF 会员，研究方向为芯片容错设计和芯片容错设计

张嘉诚：男，硕士生，硕士生，研究方向为机载电子适航设计验证、机载电子软错误防护

Secure Conversion Circuit from Arithmetic to Boolean Masking

Jiajie Zhu, Yanbin Li, Ming Tang

***Abstract*—Masking is a widely-used technique for protecting cryptographic circuits from side-channel attacks (SCAs). When implementing lattice-based KEMs, arithmetic masking is often applied to arithmetic operations, while Boolean masking is used for symmetric cryptography primitives. However, the conversion circuit costs between the two types of masking have been a significant factor in masked implementations. To address this issue, we propose a novel A2B (arithmetic to Boolean masking) conversion circuit that can efficiently convert between the two types of masking. We have optimized the ripple-carry adder circuit to solve the secure carry propagation. Our hardware implementation of the A2B conversion circuit has been integrated into the compress operator in Kyber, and we have compared our results with the existing works. The results illustrate that our conversion circuit has advantages compared to other existing circuits.**

***Index Terms*—Masking, Conversion circuit, Lattice-based cryptographic circuit.**

I. Introduction

THE advent of quantum computers has prompted concerns about the security of traditional public key cryptographic techniques based on factoring or the discrete logarithm problem. In response, the National Institute of Standards and Technology (NIST) launched the Post-Quantum Cryptography (PQC) Standardization Project [1] to safeguard against quantum attacks. It is crucial for cryptographic circuits, including PQC, to be resistant to physical attacks such as side-channel attacks (SCAs) to protect sensitive information [2]. SCAs exploit leakage information like power, electromagnetic radiation, and timing to recover secret keys or messages. Masking is an effective countermeasure that divides sensitive variables into multiple shares, preventing attackers from obtaining complete information. In lattice-based KEMs such as Kyber, algebraic operations use arithmetic masking for efficiency, while bit-level operations utilize Boolean masking. Conversion circuits are necessary to switch between two types of masking [3]–[7].

J. Zhu is with the College of Artificial Intelligence, Nanjing Agricultural University, Nanjing, China, 210095, and with the State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, China, 100878. E-mail: 15650175290@163.com.

Y. Li is with the College of Artificial Intelligence, Nanjing Agricultural University, Nanjing, China, 210095, with the State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, China, 100878, and with Henan Key Laboratory of Network Cryptography Technology. E-mail: yanbinli@njau.edu.cn. (Corresponding author: Yanbin Li)

M. Tang is with the School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, China, 430072, and with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University, China. E-mail: m.tang@126.com.

The first masking conversion circuit of any order was proposed by Coron et al., which utilized the idea of ripple-carry adder to construct the addition circuit for Boolean sharing [8]. The efficiency of the masking conversion circuit is hindered as the function SecAnd with strong non-interference (SNI) [9] property is called three times within the loop. To adapt to constrained devices, Won and Han employed the concept of the kogge-stone adder to develop an A2B conversion circuit [10]. Barthe et al. presented a novel A2B conversion circuit for sampling in lattice-based schemes [11]. This method was further enhanced in [12], exhibiting a complexity of $O(n^2 \cdot \log k)$, where n and k represent the number of shares and bits, respectively.

In this paper, we aim to present a efficient bitwise conversion circuit combined with the idea outlined in [12]. We utilize the optimization algorithm of ripple-carry adder circuit to deal with the carry propagation in the conversion process effectively. It requires two secure AND circuits instead of three, which improves the performance of the masking conversion. In addition, we compare the hardware resource costs with the existing works.

II. New Bitwise A2B Conversion Circuit

In this section, we propose a new bitwise conversion circuit from arithmetic masking to Boolean masking. How to ensure the security of carry propagation is an important issue of arithmetic masking. In our algorithms, we utilize the mathematical law of distribution to optimize the ripple-carry Adder to resist high-order SCAs. We provide the proof of correctness for the conversion circuit. In the last subsection, we analyze the security property.

A. Notions

Let n denote the number of shares and k denote the bit size of shares. Let $x^{(j)}$ denote the j-th bit of x. Obtaining a random number uniformly and randomly from a finite field is represented as $r \xleftarrow{\$} \mathbb{F}_q$.

B. Bitwise Construction

We start by presenting the one-bit construction. For one-bit arithmetic addition considering the carry calculation, it satisfies that arithmetic masking $A_1, A_2 \in \mathbb{F}_2$ follows Boolean operation $A_1 + A_2 = (A_1 \oplus A_2) + ((A_1 \wedge A_2) \ll 1)$. Let x denote the addition with $x = A_1 + A_2$, y denote the XOR of two variables with $y = A_1 \oplus A_2$, and c denote the carry from Boolean to arithmetic operation with $c = (A_1 \wedge A_2) \ll 1$.

Therefore, it can be simplified as $x = y+c$. Apparently, x may have two bits, and y has only one bit. For the carry variable c, it always has two bits with $c^{(1)} = 0$. There exists

$$\begin{aligned} x^{(1)} &= y^{(1)} + c^{(1)} \\ x^{(2)} &= c^{(2)} \end{aligned} \tag{1}$$

The arithmetic masking $(A_1, A_2), A_i \in \mathbb{F}_2$ with $A_1 + A_2 (\text{mod } 2) = A$ can be converted to Boolean shares (B_1, B_2) with $B_1 \oplus B_2 = A$, where $B_i \in \mathbb{F}_2$. It can encode arithmetic shares by $A_1 = x_1 \oplus x_2$ and $A_2 = y_1 \oplus y_2$. There exists

$$\begin{aligned} A &= A_1 + A_2 \ (mod\ 2) \\ &= (x_1 \oplus x_2) + (y_1 \oplus y_2) \ (mod\ 2) \\ &= (x_1 \oplus x_2 \oplus y_1 \oplus y_2) + c^{(1)} \ (mod\ 2) \\ &= x_1 \oplus x_2 \oplus y_1 \oplus y_2 \end{aligned} \tag{2}$$

Since $c^{(1)} = 0$, Equation (2) involves XOR operations solely. It utilizes a random number to protect them against first-order SCAs, as shown in Algorithm 1.

Algorithm 1 $\mathsf{SecA2B}^f_{Bit}$

Input: $\mathbf{A} = (A_i)_{1\le i\le 2} \in \mathbb{F}_2$
Output: $\mathbf{B} = (B_i)_{1\le i\le 2} \in \mathbb{F}_2$ with $B_1 \oplus B_2 = A_1 + A_2$

1: $B_2 \xleftarrow{\$} \mathbb{F}_2$
2: $B_1 \leftarrow B_2 \oplus A_1$
3: $B_1 \leftarrow B_1 \oplus A_2$
4: $r \xleftarrow{\$} \mathbb{F}_2$
5: $B_1 \leftarrow B_1 \oplus r$
6: $B_2 \leftarrow B_2 \oplus r$

To extend the conversion circuit to arbitrary bit sizes, it first applies Algorithm 1 to each input bit of shares separately, as shown in Algorithm 3. It requires to deal with the carry from low-order significant bit to high significant bit. Based on Equation (1), $c^{(2)}$ is the carry to be propagated. However, the cumulative effect of the carry is non negligible. It cannot implement a correct conversion from arithmetic to Boolean shares if only $c^{(2)}$ is considered. We introduce a new $\mathsf{SecCarry}^f_{Bit}$ algorithm to perform carry propagation for one-bit shares based on RCA, as shown in Algorithm 3. For $(A_1, A_2), A_i \in \mathbb{F}_{2^k}$, it can obtain that $c^{(1)} = 0$, and where $j \in [2, k]$

$$c^{(j)} = (A_1^{(j-1)} \wedge A_2^{(j-1)}) \oplus (A_1^{(j-1)} \wedge c^{(j-1)}) \oplus (c^{(j-1)} \wedge A_2^{(j-1)}) \tag{3}$$

It has four cases, namely $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$. If $c^{(j)} = 0$, $c^{(j+1)}$ obtains the value $0,0,0,1$. If $c^{(j)} = 1$, the values are $0,1,1,1$. Apparently, the result fits for the carry propagation of two shares. There exists an theorem with $(x \wedge z) \oplus (z \wedge y) = z \wedge (x \oplus y)$ in field $\mathbb{GF}_2$, which reduces one AND operation. Therefore, Equation (3) can be simplified as

$$c^{(j)} = (A_1^{(j-1)} \wedge A_2^{(j-1)}) \oplus (c^{(j-1)} \wedge (A_1^{(j-1)} \oplus A_2^{(j-1)})) \tag{4}$$

The recursive Equation (4) can be completed only by AND and XOR. Trivial linear functions are more susceptible to being transformed into private gadgets, whereas non-linear operations typically require additional random numbers. We use the SecAnd with SNI security property to protect logical And operation, as shown in Algorithm 2. The Expand circuit is from [8]. In Algorithm 3, the conversion circuit is initialized with no carry. Thus, it sets the first carry to zero. In the loop of 2 to k, it exploits the formula given in Equation (4) to compute the carries for the high significant bit. It adds the carry to the shares after the conversion of one-bit, such that

$$\begin{aligned} A^{(j)} &= A_1^{(j)} + A_2^{(j)} \\ &= B_1^{(j)} \oplus B_2^{(j)} \oplus c^{(j)} \end{aligned} \tag{5}$$

Algorithm 2 $\mathsf{SecCarry}^f_{Bit}$

Input: $\mathbf{x} = (x_i)_{1\le i\le 2} \in \mathbb{F}_2$, $\mathbf{y} = (y_i)_{1\le i\le 2} \in \mathbb{F}_2$, and $\mathbf{z} = (z_i)_{1\le i\le 2} \in \mathbb{F}_2$
Output: $\mathbf{w} = (w_i)_{1\le i\le 2} \in \mathbb{F}_2$

1: $(w_i)_{1\le i\le 2} \leftarrow (z_i)_{1\le i\le 2}$
2: $(u_i)_{1\le i\le 2} \leftarrow \mathsf{SecAnd}((x_i)_{1\le i\le 2}, (y_i)_{1\le i\le 2})$
3: $(v_i)_{1\le i\le 2} \leftarrow (x_i)_{1\le i\le 2} \oplus (y_i)_{1\le i\le 2}$
4: $(s_i)_{1\le i\le 2} \leftarrow \mathsf{SecAnd}((w_i)_{1\le i\le 2}, (v_i)_{1\le i\le 2})$
5: $(w_i)_{1\le i\le 2} \leftarrow (u_i)_{1\le i\le 2} \oplus (s_i)_{1\le i\le 2}$

Algorithm 3 Bitwise $\mathsf{SecA2B}^f$

Input: $\mathbf{A} = (A_i)_{1\le i\le 2} \in \mathbb{F}_{2^k}$
Output: $\mathbf{B} = (B_i)_{1\le i\le 2} \in \mathbb{F}_{2^k}$ with $B_1 \oplus B_2 = A_1 + A_2 (\text{mod } 2^k)$

1: $(B_i)_{1\le i\le 2}, (c_i)^{(1)}_{1\le i\le 2} \leftarrow 0$
2: $(x_i)_{1\le i\le 2} \leftarrow (A_i)_{1\le i\le 2} \& 1$
3: $(y_i)_{1\le i\le 2} \leftarrow \mathsf{SecA2B}^f_{Bit}((x_i)_{1\le i\le 2})$
4: $(B_i)_{1\le i\le 2} \leftarrow (B_i)_{1\le i\le 2} \oplus (y_i)_{1\le i\le 2}$
5: $(u_i)_{1\le i\le 2}, (v_i)_{1\le i\le 2} \leftarrow \mathsf{Expand}((y_i)_{1\le i\le 2})$
6: **for** $j = 2$ to k **do**
7: $\quad (c_i)^{(j)}_{1\le i\le 2} \leftarrow \mathsf{SecCarry}^f_{Bit}(\mathbf{u}, \mathbf{v}, (c_i)^{(j-1)}_{1\le i\le 2})$
8: $\quad (x_i)_{1\le i\le 2} \leftarrow ((A_i)_{1\le i\le 2} \gg (j-1)) \& 1$
9: $\quad (y_i)_{1\le i\le 2} \leftarrow \mathsf{SecA2B}^f_{Bit}((x_i)_{1\le i\le 2})$
10: $\quad (u_i)_{1\le i\le 2}, (v_i)_{1\le i\le 2} \leftarrow \mathsf{Expand}((y_i)_{1\le i\le 2})$
11: $\quad (B_i)_{1\le i\le 2} \leftarrow (B_i)_{1\le i\le 2} \oplus (((y_i)_{1\le i\le 2} \oplus (c_i)^{(j)}_{1\le i\le 2}) \ll (j-1))$
12: **end for**

Algorithm 2 can be directly extended to n shares to resist higher-order side-channel attacks, as shown in Algorithm 4. Let $(x_i)_{1\le i\le n}$ denote the extension of $(A_i)_{1\le i\le n/2}$, and $(y_i)_{1\le i\le n}$ denote the extension of $(A_i)_{n/2+1\le i\le n}$. It has

$$\begin{aligned} (c_i)^{(j)} = ((x_i)^{(j-1)} \wedge (y_i)^{(j-1)}) \oplus ((c_i)^{(j-1)} \\ \wedge ((x_i)^{(j-1)} \oplus (y_i)^{(j-1)})),\ 1 \le i \le n \end{aligned} \tag{6}$$

In order to reduce the random numbers, the new higher-order bitwise conversion circuit uses the SecAndPINI circuit with the probe isolating non-interference (PINI) property instead of SNI [13]. Compared with SecAnd, the computational complexity is similar. The complete conversion circuit is shown in Algorithm 5, which uses the idea of divide and conquer. Fig. 1 shows the complete process of A2B schemes.

Algorithm 4 SecCarry

Input: $\mathbf{x} = (x_i)_{1\le i\le n}$, $\mathbf{y} = (y_i)_{1\le i\le n}$
Output: $\mathbf{z} = (z_i)_{1\le i\le n}$ with $\bigoplus_{i=1}^{n} z_i = \bigoplus_{i=1}^{n} x_i + \bigoplus_{i=1}^{n} y_i$

$(u_i)_{1\le i\le n} \leftarrow (x_i)_{1\le i\le n} \oplus (y_i)_{1\le i\le n}$
$(c_i)^{(1)}_{1\le i\le n} \leftarrow 0$
$(v_i)_{1\le i\le n} \leftarrow$SecAndPINI$((x_i)_{1\le i\le n}, (y_i)_{1\le i\le n})$
for $j = 2$ to k **do**
    $(c_i)^{(j)}_{1\le i\le n} \leftarrow$SecAndPINI$((c_i)^{(j-1)}_{1\le i\le n}, (u_i)^{(j-1)}_{1\le i\le n})$
    $(c_i)^{(j)}_{1\le i\le n} \leftarrow (c_i)^{(j)}_{1\le i\le n} \oplus (v_i)^{(j-1)}_{1\le i\le n}$
end for
$(z_i)_{1\le i\le n} \leftarrow (u_i)_{1\le i\le n} \oplus (c_i)_{1\le i\le n}$

Algorithm 5 Bitwise SecA2B

Input: $\mathbf{A} = (A_i)_{1\le i\le n} \in \mathbb{F}_{2^k}$
Output: $\mathbf{B} = (B_i)_{1\le i\le n} \in \mathbb{F}_{2^k}$ with $\bigoplus_{i=1}^{n} B_i = \sum_{i=1}^{n} A_i$

if $n = 1$ **then**
    return A_1
end if
$(x_i)_{1\le i\le n/2} \leftarrow$Bitwise SecA2B$((a_i)_{1\le i\le n/2})$
$(x'_i)_{1\le i\le n} \leftarrow ((x_i)_{1\le i\le n/2}, (0,\cdots,0))$
$(y_i)_{1\le i\le n/2} \leftarrow$Bitwise SecA2B$((a_i)_{n/2+1\le i\le n})$
$(y'_i)_{1\le i\le n} \leftarrow ((0,\cdots,0), (y_i)_{1\le i\le n/2})$
$(B_i)_{1\le i\le n} \leftarrow$SecCarry$((x'_i)_{1\le i\le n}, (y'_i)_{1\le i\le n})$

C. Correctness

Based on Equation (2), Algorithm 1 is correct. It proves the correctness of Algorithm 3 based on the property of Algorithm 1. For two input shares, it can obtain that

$$A = A_1 + A_2 (mod\ 2^k),\ A_i \in \mathbb{F}_{2^k} \tag{7}$$

The output shares are

$$B_i = C_i \oplus c_i,\ B_i \in \mathbb{F}_{2^k} \tag{8}$$

where C_i denotes the results of conversion without carry and therefore $\bigoplus_{i=1}^{2} B_i = A$, combined with Equation (3) to (5), it can obtain the correctness of Algorithm 3.

The correctness of Algorithm 4 is determined. For Algorithm 5 with n input shares, it can obtain that

$$A = A_1 + A_2 + \cdots + A_n (mod\ 2^k),\ A_i \in \mathbb{F}_{2^k} \tag{9}$$

The output shares get

$$B_i^{(j)} = 2^0 (A_1^{(j)} \oplus c_1^{(j)}) \oplus \cdots \oplus 2^{n-1} (A_n^{(j)} \oplus c_n^{(j)}),\ B_i \in \mathbb{F}_{2^k} \tag{10}$$

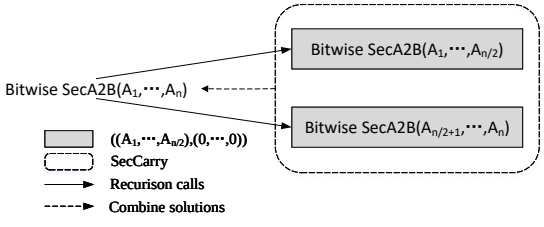


Fig. 1. The complete process of Bitwise SecA2B schemes.

and therefore $\bigoplus_{i=1}^{n} B_i = A$. Algorithm 5 can perform the conversion of n shares correctly.

D. Security

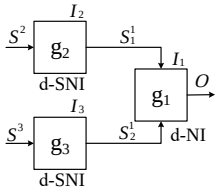


Fig. 2. Structure of gadget portfolio 1 for Lemma 1.

Lemma 1: Given a gadget portfolio circuit C as in Figure 4, where g_2, g_3 are d-SNI gadgets and g_1 is d-NI, the circuit C is d-SNI.

Proof: Let $\Omega = (I, O)$ denote the collection of antagonistic observations on C with $|I| + |O| \le d$, where I are the probes on the internal and O the ones on the output values. In particular, let I_1 denote the probe on g_1, I_2 on g_2 and I_3 on g_3, with $|I_1 \cup I_2 \cup I_3| \le |I|$. Firstly, g_1 is d-NI and $|I_1 \cup O| \le d$ with two observation sets $|S_1^1| \le |I_1 \cup O|$, $|S_2^1| \le |I_1 \cup O|$. It can be simulated by at most $|S_1^1| + |S_2^1|$ input. Secondly, g_2 is d-SNI and $|I_2 \cup S_1^1| \le d$ with an observation set $|S^2| \le |I_2|$. It can be simulated by at most $|S^2|$ input. Thirdly, g_3 is d-SNI and $|I_3 \cup S_2^1| \le d$ with an observation set $|S^3| \le |I_3|$. It can be simulated by at most $|S^3|$ input. Therefore, the circuit C can be simulated by at most $|S^2 \cup S^3| \le |I_2| + |I_3| \le |I|$ input with d-SNI property. ■

Proposition 1: The masking conversion circuit of Algorithm 1 is 1-SNI.

Proof: Let assume that the adversary has one probe d for Algorithm 1. The following possible groups exist:

(1) A_1, A_2
(2) $B_2 = s$
(3) $B_1 = s \oplus A_1$
(4) $B_1 = B_1 \oplus A_2 = s \oplus A_1 \oplus A_2$
(5) r
(6) $B_1 = s \oplus A_1 \oplus A_2 \oplus r, B_2 = s \oplus r$

If one probe d is in group (1), it can be simulated by A_1 or A_2. If d is in one of the groups from (2) to (6), it is simulated by assigning it to a independent random value in $\mathbb{F}_2$ in the computation of d. If it probes the output $(s \oplus A_1 \oplus A_2 \oplus r)$ and the internal value r or s, it cannot obtain the information of the input, following the definition of 1-SNI. ■

Proposition 2: The masking conversion circuit of Algorithm 3 is 1-SNI.

Proof: Since the SecAnd has the d-SNI property, the property also holds for Algorithm 2. The 1-SNI security of Algorithm 3 easily follows from the fact that Algorithm 1 and Algorithm 2 are d-SNI, and Expand circuit is d-NI. Based on Lemma 1 corresponding to [12], C_1 is 1-SNI. Similarly, C_2 and C_2 are 1-SNI. Therefore, it can be proved that C_n is 1-SNI. ■

Proposition 3: The masking conversion circuit of Algorithm 5 is $(n-1)$-PINI.

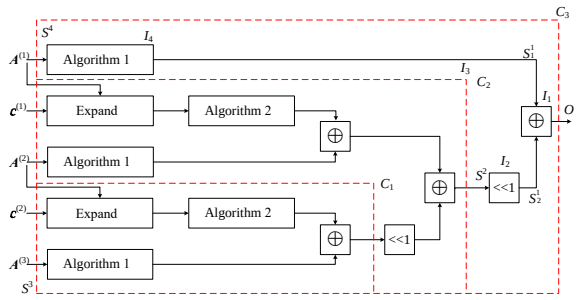


Fig. 3. Structure of Algorithm 3 for $k = 3$.

Proof: The masking conversion circuit is the composition of PINI gadgets, therefore Algorithm 5 is PINI. ■

III. Implementation Result

In this section, we start by outlining the implementation of our novel A2B conversion circuits, which have been executed on an Intel (R) Core (TM) i7-12700HQ CPU and RTX 3050Ti Laptop GPU, featuring a clock speed of 2.70 GHz and 16 GB of memory within a 64-bit Windows environment. Subsequently, we present comparisons between our new algorithms and existing methods, taking into account resource consumption, as well as the complexity of implementations.

A. Operation Count of Boolean Circuit

We analyze the number of required operations of the bitwise-based masking conversion circuit. It is compared to the existing works for the conversion from arithmetic masking to Boolean masking. Referring to [14], we assume that the time consumed for the randomness generation circuit is one unit of time. In the following, it does not compared with [14] because of the different directions of masking conversion circuit. We estimate the operation counts in several cases including different numbers of share and bit. The results are shown in Table I.

Coron et al. used the idea of ripple-carry adder to implement the masking conversion circuit with the worst performance called the basic circuit [8]. The demand for the secure AND circuit affects the efficiency of the transformation between two types of masking. Our conversion circuit utilizes the law of distribution among AND and XOR to reduce this demand, which improved the performance of the conversion circuit significantly. Based on the recursion from Goubin's theorem, Coron et al. proposed a new conversion circuit with the time complexity $O(n^2 \cdot k)$ [8]. Since the number of calls to the secure AND function inside the loop is required only one time, the masking conversion circuit of Algorithm 5 performs similarly to this circuit. For the masking conversion circuit based on the kogge-stone adder, the algorithm of [12] is compared instead of the circuit in [11] because of the lower overall complexity. Compared with [12], Algorithm 5 outperforms for the case of small bits.

B. Hardware Resource

The proposed conversion circuit has been implemented with Vivado HLS 2019.2 design suite on virtex7 xc7vx485t-ffg1157-1 device, with all building blocks implemented in hardware. The utilization of look up table (LUT), flip flop (FF), digital signal processing (DSP) blocks, and BRAM resources determines the FPGA resource consumption of a design. All designs are synthesized to achieve a target clock rate of 121.34 MHz. We first compare the new conversion circuit in terms of resource consumption implemented on FPGA with the existing works. All our findings are summarized in Table II. Compared with [12] based on the kogge-stone adder, Algorithm 5 outperforms for the resource consumption corresponding to LUTs and FFs. Our conversion circuit required more resources than [8].

In lattice-based cryptographic circuits such as Kyber, they require both arithmetic masking and Boolean masking to protect the circuit. Algebraic operations are typically well-suited for arithmetic masking, while Boolean masking is commonly used for bit-level operations. Based on the masking idea of [12], we compare hardware design resources with different conversion circuits for the masked compress operator. The results are similar to the performance of the conversion circuit itself, as shown in Table III. In addition, we utilize the Test Vector Leakage Assessment (TVLA) [15], [16] to provide the practical security of the implementation of the masked compress operator of Kyber, as shown in Fig. 4.

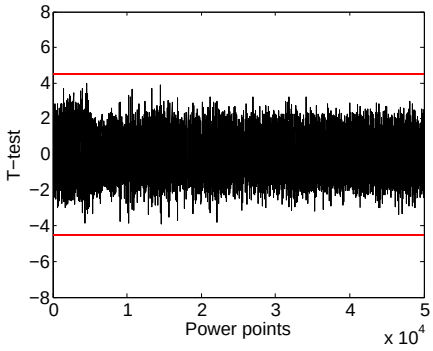


Fig. 4. The t-test results for the masked compress operator implementation with Algorithm 5.

IV. Conclusion

In this work, we propose a higher-order A2B conversion circuit. Starting by implementing the bitwise conversion from arithmetic to Boolean shares, we use the law of distribution to optimize the addition circuit for Boolean shares. The proposed circuit is proven to be $(n-1)$-PINI enabling composition with n shares. In the hardware implementation, the bitwise A2B circuit performs better since the bitwise operations and the overhead has been reduced by about 10% compared with kogge-stone adder-based conversion circuits.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant(NO.62072247,

TABLE I
OPERATION COUNT FOR DIFFERENT HIGHER-ORDER A2B CIRCUITS.

Counts		$n=2$	$n=4$	$n=8$	$n=16$	$n=32$
$k=2$	Basic circuit [8]	43	256	1188	5072	20912
	Coron et al. [8]	36	200	880	3616	14528
	Schneider et al. [12]	36	224	1072	4672	19520
	Algorithm 5	29	172	796	3392	13968
$k=4$	Basic circuit [8]	105	672	3276	14448	60816
	Coron et al. [8]	62	360	1640	6912	28256
	Schneider et al. [12]	94	592	2856	12512	52448
	Algorithm 5	49	308	1484	6496	27216
$k=8$	Basic circuit [8]	229	1504	7452	33200	140624
	Coron et al. [8]	114	680	3160	13504	55712
	Schneider et al. [12]	123	776	3748	16432	68912
	Algorithm 5	89	580	2860	12704	53712
$k=12$	Basic circuit [8]	353	2336	11628	51952	220432
	Coron et al. [8]	166	1000	4680	20096	83168
	Schneider et al. [12]	152	960	4640	20352	85376
	Algorithm 5	129	852	4236	18912	80208

TABLE II
HARDWARE RESOURCE COSTS FOR DIFFERENT HIGHER-ORDER A2B CIRCUITS.

Design	LUTs			FFs			DSP			BRAM		
	n=2	n=4	n=8	n=2	n=4	n=8	n=2	n=4	n=8	n=2	n=4	n=8
Coron et al. [8]	2059	5135	7198	1094	2186	2940	9	18	24	0	8	16
Schneider et al. [12]	2473	7366	9377	1380	6434	3954	6	15	24	0	0	28
Algorithm 5	2401	5354	8093	826	1758	2888	6	12	18	0	8	16

TABLE III
HARDWARE RESOURCE COSTS FOR THE COMPRESS OPERATOR WITH DIFFERENT HIGHER-ORDER A2B CIRCUITS.

Design	LUTs			FFs			DSP			BRAM		
	n=2	n=4	n=8	n=2	n=4	n=8	n=2	n=4	n=8	n=2	n=4	n=8
Coron et al. [8]	4833	8205	10389	3629	4943	5799	9	18	24	0	8	16
Schneider et al. [12]	6283	9340	12441	4041	5418	6528	9	18	27	0	7	16
Algorithm 5	5181	8440	11271	3298	4451	5714	6	12	18	0	8	16

NO.61972295), and by Henan Key Laboratory of Network Cryptography Technology (LNCT2022-A07).

REFERENCES

[1] NIST, "Post-quantum cryptography standardization," 2017, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[2] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the second round of the nist post-quantum cryptography standardization process," *US Department of Commerce, NIST*, Jul. 2020.

[3] T. Fritzmann, M. Van Beirendonck, D. Basu Roy, P. Karl, T. Schamberger, I. Verbauwhede, and G. Sigl, "Masked accelerators and instruction set extensions for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 1, pp. 414–460, Nov. 2021.

[4] D. Heinz, M. J. Kannwischer, G. Land, T. Pöppelmann, P. Schwabe, and D. Sprenkels, "First-order masked kyber on arm cortex-m4," *Cryptology ePrint Archive*, 2022.

[5] R. Jevtic and M. G. Otero, "Methodology for complete decorrelation of power supply em side-channel signal and sensitive data," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 4, pp. 2256–2260, 2022.

[6] Y. Xiang, Y. Xu, Y. Li, W. Ma, Q. Xuan, and Y. Liu, "Side-channel gray-box attack for dnns," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 501–505, 2020.

[7] S. Sun, H. Zhang, X. Cui, L. Dong, and X. Fang, "Electromagnetic side-channel hardware trojan detection based on transfer learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1742–1746, 2021.

[8] J.-S. Coron, J. Großschädl, and P. K. Vadnala, "Secure conversion between boolean and arithmetic masking of any order," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg, 2014, pp. 188–205.

[9] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini, "Strong non-interference and type-directed higher-order masking," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 2016, pp. 116–129.

[10] Y.-S. Won and D.-G. Han, "Efficient conversion method from arithmetic to boolean masking in constrained devices," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, Paris, France, 2017, pp. 120–137.

[11] G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi, "Masking the glp lattice-based signature scheme at any order," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Tel Aviv, Israel, 2018, pp. 354–384.

[12] T. Schneider, C. Paglialonga, T. Oder, and T. Güneysu, "Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto," in *IACR International Workshop on Public Key Cryptography*. Springer, Beijing, China, 2019, pp. 534–564.

[13] G. Cassiers and F.-X. Standaert, "Trivially and efficiently composing masked gadgets with probe isolating non-interference," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2542–2555, 2020.

[14] J.-S. Coron, "High-order conversion from boolean to arithmetic masking," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, Taipei, Taiwan, 2017, pp. 93–114.

[15] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136.

[16] T. Schneider and A. Moradi, "Leakage assessment methodology," *Journal of Cryptographic Engineering*, vol. 6, no. 2, pp. 85–99, Feb. 2016.

基于 FTSTM 的工控协议可信性验证

王洁*[①②]　吴鑫涛[①②]　安然[①②]　范搏栋[①②]

[①](大连理工大学软件学院　大连　11620)

[②](辽宁省泛在网络与服务软件重点实验室　大连　11620)

摘　要：在工业控制网络系统中，偶发的故障或者系统失效常常伴随着危及生命、财产的重大损失。因此对于工业控制系统网络协议设计的可行性要求，不仅需要保证系统在正常情况下行为的可用性、正确性，还需要考虑在可能的系统功能失效和故障发生时，保证系统也能正常运行。为了验证工业控制网络协议的可信性，结合形式化技术与故障注入技术对现有的时间状态迁移矩阵进行扩展，提出一种基于故障时间状态迁移矩阵（Fault Time State Transition Matrix，FTSTM）的形式化建模方法。并设计一个基于 FTSTM 的可信验证框架，通过结合预设或自定义的故障模式，对原始协议模型进行故障注入并生成 FTSTM。而后结合协议需求规约与不同的 FTSTM 故障激活模型，通过限界模型检测技术来验证协议模型的正确性、可用性与鲁棒性，最后基于二元决策图技术自动生成故障树，通过对故障树进行分析，为提高协议设计的容错性提供了优化思路。

关键词：模型检测；故障注入；故障时间状态迁移矩阵；故障树

Trustworthiness verification of industrial control protocols based on FTSTM

WANG Jie[①②]　WU Xintao[①②]　AN Ran[①②]　FAN Bodong[①②]

[①](School of Software, Dalian University of Technology, Dalian 11620, *China*)

[②](*Liaoning Key Laboratory of Ubiquitous Network and Service Software, Dalian* 11620, *China*)

Abstract: In the industrial control network system, accidental failures or system failures are often accompanied by serious losses of life and property. Therefore, for the feasibility requirements of industrial control system network protocol design, it is not only necessary to ensure the availability and correctness of system behavior under normal circumstances, but also need to consider the possibility of system function failure and failure, to ensure that the system can run normally. In order to verify the trustworthiness of industrial control network protocols, this paper proposes a formal modeling method based on Fault Time State Transition Matrix (FTSTM) by combining formal technology and fault injection technology to extend the existing time state transition matrix. A trusted verification framework based on FTSTM is designed to inject faults into the original protocol model and generate FTSTM by combining preset or custom fault modes. Then, combined with the protocol requirements specification and different FTSTM fault activation models, the bounded model checking technology is used to verify the correctness, availability and robustness of the protocol model. Finally, the fault tree is automatically generated based on the binary decision diagram technology, and the fault tree is analyzed to provide an optimization idea for improving the fault tolerance of the protocol design.

Key words: Model checking; Fault injection; Fault time state transition matrix; Fault tree

*通信作者：王洁　wang_jie@dlut.edu.cn

基金项目：工控系统安全可信关键技术及应用（NO.2020YFB2009500）

1 引言

1.1 研究背景及意义

工业控制系统[1-2]（Industrial Control System，ICS）是由计算机与工业过程控制部件组成的自动控制系统，通常由控制器、传感器、传送器、执行器和输入/输出接口等部分组成。工业控制系统广泛应用于电力、水利、污水处理、石油化工、冶金、汽车、航空航天等诸多现代工业领域，其中超八成涉及关键基础设施。随着工业化与信息化的深度融合，以及“互联网+”等国家战略的推出，云计算、大数据、物联网等新兴技术的大规模推广，工业控制系统逐渐由封闭独立走向开发、由单机走向互联、由自动化走向智能化。伴随着这一趋势，传统工业控制系统网络安全的隐患日益凸显。现有信息安全防护手段难以应付层出不穷的未知新型攻击，工业控制系统设备缺乏内生安全可信机制。工业控制系统的设计是以工业系统的高可靠性、高抗干扰性、高实时性和高控制性为目标的，其底层通讯协议以高速、短帧、抗干扰为特点的异步数据通讯方式，目前以各种现场总线的应用方式为主，与之相关的各种通讯协议在设计之初基本上没有过多信息安全方面的风险考虑，如攻击者熟知设备的通讯协议和通讯地址，系统极易受到危害性攻击。

为了对工业控制网络协议的可信性验证，本文展开了一种基于形式化方法与故障注入技术的分析研究，结合形式化技术与故障注入技术对现有的时间状态迁移矩阵进行扩展，提出一种故障时间状态迁移矩阵的形式化建模方法。并设计一个基于故障时间状态迁移矩阵（Fault Time State Transition Matrix，FTSTM）的可信验证框架，该框架从工业控制网络协议设计出发，抽象出原型模型，用于故障扩展与模型迭代，围绕该框架设计一套故障扩展工具包（FTSTM-based Fault extension Toolkit，FFET）。工具包自动对基于时间状态矩阵的原始协议模型进行故障注入并生成 FTSTM，结合协议需求规约与不同的 FTSTM 故障激活模型，通过限界模型检测技术来验证协议模型的正确性、可用性与鲁棒性，而后使用二元决策图（Binary Decision Diagram，BDD）技术自动生成故障树，通过对故障树分析，为协议设计的容错性提高提供了优化思路。最后以真实的工业控制网络协议 PowerLink 为例，应用该可信验证框架进行建模与设计优化，例证方法的有效性。

1.2 国内外研究现状

目前针对工业控制系统的可信性保证方法主要包括访问控制和认证技术、安全风险评估技术和形式化验证方法等[3-5]。Shazad 等人[6]提出了一种基于密码的解决方案保护监控和数据采集（SCADA）系统协议和通信。在测试平台上实现时，创建了一个攻击场景，并根据安全级别进行了测量和比较攻击检测级别和攻击影响级别。Ani 等人[7]提出了一种以人为中心的安全能力和脆弱性评估方法，用于评估工业控制系统环境中人类代理的安全能力。在实际应用中，上述方法虽然也表现良好，但在实际应用中却不如正规的验证方法确保对系统的某些关键组件的高度信任。赖英旭等人[8]介绍了一种基于模糊测试的工控网络协议漏洞挖掘方法，使用工控网络协议测试用例变异因子生成协议特征值，结合协议特征生成不同的测试用例。旨在提高私有协议的测试用例接收率，以提高工业控制系统的安全性。

协议的可信性保证方法是计算机科学中的一个重要研究领域，其目的是确保协议在运行过程中能够保持一定的安全性和可靠性，避免遭受攻击或出现意外情况。目前协议可信性保证方法主要包括：形式化验证、安全协议分析、代码审计、模拟器测试、模拟攻击。

常见的形式验证方法主要由自动定理证明和模型检验。自动定理证明是一种将自动化技术与人工指导相结合来证明的技术一个系统。Bertolotti 等人[9]提出了一种形式化技术和一种工具，可用于支持加密协议的开发，并通过案例研究说明如何使用这种技术来检测可能的逻辑标准安全协议中的错误。Bugliesi 等人[10]依靠 Coq（一个交互式定理证明工具）来验证本机 cookie 保护机制第一次基于 HttpOnly 和 Secure 标志。而自动化定理证明技术减少了人工推理的工作量，对这些工具的使用仍有较高的要求来自相关人员的专业知识。模型校验在工业控制网络协议的验证中得到了广泛的应用 Chan 等人[11]提出了一种生成操作技术级别安全约束的方法，并通过模型检查对这些约束进行形式化验证。Zhang 等[12]测试了快速 UDP 互联网连接握手的安全性协议的符号模型，并在协议设计中发现缺陷。Kunnappilly 等[13]使用了 UPPAAL（一种形式化的模型检查集成工具环境，用于建模和验证）对动态服务的 5G 编排进行建模和验证。

Yamane 和 Uemura[14]比较了 SPIN（一种允许动态创建并发进程的集成工具）模型通过随机或迭代模拟系统操作和可满足模理论为基础 BMC 软件中嵌入式装配程序的模型检查。

此外，还有混合的方法来验证系统，例如将模型检查与其他方法相结合。Villani 等[15]提出了 UPPAAL 模型检查和故障注入的结合应用。将一种较新的系统理论分析方法与模型检验相结合，对飞行安全进行评价模拟器。

上述模型检查方法大多适用于工业控制网工作协议的具体模块分析（如认证或加密），而不适用于协议基本功能的验证设计和发现基本的潜在冲突和漏洞。相比之下，基于 STM 的建模方法具有简单的模型结构和建模设计中缺失部分的清晰表示的优点。它是否更容易建立一个完整的沟通过程模型，并发现设计中的遗漏。普通的 STM 的描述能力有限，为了扩展 STM 的表达能力，侯刚等人[16]在原有的 STM 基础上增加了时间语义与约束，提出时间状态迁移矩阵（Time State Transition Matrix，TSTM）。

故障注入和故障扩展技术是指在工业控制系统中故意引入故障，旨在故意引入故障以测试系统的可靠性和容错能力。这些技术对于确保系统的安全性和可靠性起着重要的作用。该研究最早可以追溯到二十世纪七十年代初，IBM 实验室率先采用硬件注入方法对计算机系统进行故障注入，以评估经过故障注入后的系统能自行修复及规避多少故障，这是对系统容错进行评估的一种最早的验证方式。随着测试和容错研究的深入以及集成电路的广泛应用，故障注入技术也得到了不断改进和突破。李润泽等人 [17] 提出了一种基于统计模型和改进宽度学习系统（BLS）的故障诊断方法，该方法利用多元数据统计分析表征无人机集群系统在正常和不同故障模式下的行为特征，采用改进的 BLS 模型实现准确、快速的故障诊断。根据研究目标的不同，故障注入方法逐渐呈现出多样性和复杂性的实现方式。故障注入方法在整个电路的设计、实现及应用等开发过程的各个阶段中起到了重要作用，主要用于评估系统测试容错的有效性、可靠性验证以及算法实行效率的评估。常见的应用包括故障字典生成、故障传播率评估、系统或算法故障覆盖率评估、延迟时间计算以及软件测试等。

1.3 本文主要工作

结合形式化技术与故障注入技术提出一种面向工业控制网络协议的基于故障扩展的可信验证框架。对原有的时间状态迁移矩阵（TSTM）进行了故障扩展，提出一种故障时间状态迁移矩阵（FTSTM）的形式化建模方法。使用该形式化建模方法完成从协议设计描述到协议原型模型的转化，同时使用线性时序逻辑（LTL）对协议需求进行属性规约。首先对原型模型进行故障扩展，而后将故障扩展模型输入模型检测器进行验证，验证结果用于优化协议设计与迭代原型模型。同时，使用二元决策图（BDD）技术生成故障树（Fault Tree，FT），用于对模型的故障发生过程进行研究分析，从另一个角度为协议设计优化提供了思路。最后，针对常用的工控网络协议 PowerLink 应用该可信分析框架进行 FTSTM 建模与验证，例证了方法的有效性。

2 基于 FTSTM 的可信性验证

2.1 故障时间状态迁移矩阵（FTSTM）

基于模型的故障扩展是一种基于系统模型进行故障注入的方法，通过建立系统的形式化模型，对系统进行故障注入和评估，以提高系统的可靠性和容错能力。故障时间状态迁移矩阵（FTSTM）是对 TSTM 进行了故障扩展，保留了 TSTM 原有的形式与语义的同时增加了故障切片与故障注入机制。在选择故障注入点时，针对模型特性选用状态转移时注入故障的方式，可以高效的配置需要注入的故障并保持原型模型的独立性。

一个 FTSTM 表格可以用一个五元组＜S，E，C，T，FE＞表示，其中S，E，C，T的语义与 TSTM 相同。FE 为故障扩展，由受影响符号子集 A，故障注入函数集Ψ，故障集 F，以及故障触发函数ϕ构成。

$$FE:=<A，\Psi，F，\phi>$$

A为受影响符号子集，为原型模型中全体符号变量的子集，为受到故障直接影响的变量集合。

Ψ为故障注入函数集，故障注入函数 $\psi \in \Psi$，用于对受控变量的值进行特定的修改（故障处理方式），受控变量输出值 varout 与变量输入值 varin 和参数列表Params之间的关系有：

$$varout = \psi\ (varin，Params)$$

F为故障的集合，分为基本故障F_{base}与派生故障F_{derive}。约定当故障名称以“Fault_”开头时表示基本故障（下文简称为故障），若故障全称过长有时也简写成 Fault 加序号，当故障名称以“E_”开头时表示派生故障。

（1）F_{base}

$f_b \in F_{base}$表示基本故障，可以表示为四元组

$$< a, v, e, d > \in \{\text{TRUE}, \text{FALSE}\} \times \text{A} \times \Psi \times \Psi$$

其中：

$a = \text{available}(f_b)$表示基本故障$f_b$当前的触发状态；

$v = \text{variable}(f_b)$表示受基本故障f_b影响的符号子集；

$e = \text{entering}(f_b)$表示基本故障f_b发生时的故障注入函数；

$d = \text{during}(f_b)$表示基本故障f_b保持期间的故障注入函数；

表 1 为一个基本故障集样例，varin 为进入故障模式时受控变量的初始值，varout为进入故障模式的注入值，varout'为故障保持期间的注入值。“Fault_frozen”表示受控变量处于卡死状态，即数值不再发生改变。“Fault_invert”表示该受控变量数值发生反转。“Fault_delay”与“Fault_randomDelay”可用于模拟通信网络中发生了延时，前者为固定的时延，后者为模拟网络状态不稳定而产生的随机时延。

表 1 基本故障集合样例

故障名称	参数	Entering函数	During函数
Fault_frozen	-	varout=varin	varout'=varout
Fault_invert	-	varout=!varin	varout'=varout
Fault_delay	Delay	varout=varin+Delay	varout'=varout+Delay
Fault_randomDelay	MaxDelay	varout=varin+ Random(0,MaxDelay)	varout'=varin+Random(0,MaxDelay)

（2） F_{derive}

$f_d \in F_{derive}$表示派生故障，可以表示为二元组

$$< a, c > \in \{\text{TRUE}, \text{FALSE}\} \times (\Gamma)$$

其中：

$a = \text{available}(f_d)$表示派生故障$f_d$当前的触发状态；

$c = \text{constraint}(f_d)$表示派生故障$f_d$的约束判定语句，判定约束符合 LTL 语法，$\Gamma$ 为 LTL 的非空子集；表 2 为一个派生故障集样例，派生故障的判定语句符合 LTL 语法。其名称可以具有实际意义如“E_Network_Unavailable”表示当断线重连次数大于 10 后，即标记网络失败，标记为一个派生故障。派生故障的名称也可以不具有实际的意义，用于描述基本故障传递过程中间事件。如 E_1or2，E_2and3 表示基本故障形成的几种逻辑组合。

Φ为故障触发函数，用于原型模型与故障扩展模型之间的切换。在原型模型中单元格末尾增加该函数表示当该单元格执行完毕，将要进行状态跳转之前，进入故障模型，触发相应的故障模式。而在故障模型中Φ表示故障注入结束，返回原型模型继续执行。

表 2 派生故障集合样例

故障名称	约束判定
E_Network_Unavailable	Reconnect_Num>10
E_1or2	Fault1||Fault2
E_2and3	Fault2&&Fault3

2.2 FTSTM 构造

故障扩展器接收故障扩展指令，将配置的故障模式注入原型模型。故障扩展器接收 xml 格式的故障扩展指令文件，可以通过配置故障指令文件实现不更改原型模型动态构造扩展模型的功能。

一个故障扩展指令文件中包含一个故障扩展标签（<Fault_Extension>），一个故障扩展标签对应多个故障切片标签（<Fault_Slice>）。一个故障切片中使用<TSTM>标签与模型表格进行绑定，<TSTM>标签中包含一个故障模式组标签（<Fault_Mode_List>），故障模式组标签中又可嵌套多个故障模式标签（<Fault_Mode>），并使用<position>标签标记故障注入位置，<position>标签中包含<position_row>、<position_column>分别标记注入表格的行号与列号，<Affected_variable>标签表示故障作用的变量名。<Fault_Library_path>为待读取故障模式库文件路径。<Default_State>表示该故障模式的初始状态，分为 occurred（已触发态）、wait（等待触发态。一个完整故障扩展指令的标签结构如下：

通过故障扩展指令文件的配置故障模式注入到原型模型中对应位置。实现方法首先在指令模型中指定位置插入 ϕ 函数，其次将配置中的故障模式与所绑定的受控变量结合，生成故障模型中的函数体，最后用生成的故障名称标记故障模型中的事件，填入矩阵生成完整的故障模型。

使用故障扩展器自动扩展，可以动态的生成各种所需的扩展模型。根据配置文件指定位置中插入故障触发函数 ϕ，将特定的故障注入到指定的位置。同时自动生成故障扩展矩阵，注入故障行为，每配置一个故障模式，Fault_Mode 矩阵中增加一行。如图 1 所示，故障扩展矩阵中的事件名称为故障名加受控变量名拼接而成。单元格中的行为函数与故障函数一一对应。如 Frozen_ent 函数对应故障模式“Fault_frozen”的 Entering 函数，Frozen_dur 函数对应故障模式中“Fault_frozen”的 During 函数。同时，每一个单元格中插入故障触发函数 ϕ，用于完成故障扩展矩阵与模型矩阵之间的跳转。

Fault_Mode	S	Entering	During
E		0	1
Normal_Mode	0	ϕ	ϕ
		1	1
Fault_frozen_var	1	Frozen_ent(var); ϕ	Frozen_dur(var); ϕ
		1	1
Fault_invert_var2	2	Invert_ent(var2); ϕ	Invert_dur(var2); ϕ
		1	1

图 1 故障扩展矩阵样例

2. 3 FTSTM 编码

相较于 STM 的编码方式[17]，FTSTM 在每一步迁移之后增加了故障影响。对于一个 FTSTM 设计 *D*，在初始状态为 *step*[0]，则有：

$$step[0] \equiv \wedge_{i=1}^{n} (\text{index}(active(M_i)[0]) = 0) \wedge (value(X(D)[0]) \wedge (timer = 0)$$
$$\wedge\wedge_{i=1}^{m} occur(F_i)[0] = \text{Default_State}(F_i)$$

其中，设计 D 中包含的矩阵总数为 n，注入的故障总数为 m，$value(X(D)[0]$表示为全体变量集 X(D)赋初始值。$active(M_i)[k]$表示在第 k 步时矩阵M_i中激活的状态。$occur(F_i)[k]$表示在第 k 步时故障F_i的触发状态。Default_State(F_i)表示故障模式F_i在故障扩展指令中配置的默认触发状态。$timer$为全局时钟变量。

对于普通单元格c_N，$c_N \in C_{normal}$，当单元格被触发时，单元格中的动作、跳转事件、以及对应的故障影响都被触发，故障影响变量将继续受到 During 函数的控制，而未被触发单元格影响的其他变量保持不变。

$$c_N[k] \equiv (enabled(c_N)[k] \wedge effects(c_N)[k])$$
$$\vee \left(\neg enabled(c_N)[k] \wedge\wedge_{x \in X(D)/\text{variable}(F_{occur})} \text{x[k]} = \text{x[k} - 1]\right)$$
$$\vee \left(\wedge_{x \in \text{variable}(F_{occur})} \text{x[k]} = During(\text{x[k} - 1])\right)$$

x$[k]$表示第 k 步的变量取值。enabled$(c_N)[k]$表示第 k 步时单元格c_N的使能条件，$effects(c_N)[k]$表示第 k 步时触发该单元格的所有影响，F_{occur}表示已经发生的故障模式，variable(F)表示受到故障模式的 F 影响的符号子集。

单元格c_N的使能条件有上一步的触发事件与跳转状态共同决定，即为：

$$enabled(c_N)[\text{k}] \equiv guard(c_N)[k-1] \wedge event(c_N)[k-1] \wedge active(M_i)[k-1] == source(c_N)$$

$guard(c_N)[\text{k} - 1]$表示在 k-1 步时c_N的守卫条件，$event(c_N)[k-1]$表示第 k-1 步能够触发c_N执行的事件，$active(M_i)[k-1]$表示第 k-1 步执行完成，矩阵M_i所激活的状态。

单元格c_N的影响包括单元格动作影响、状态跳转影响、时间影响以及故障触发影响。

$$effect(c_N)[k] \equiv actions(c_N) \wedge active(M_i)[k] = target(c_N) \wedge timer = timer + \Delta(x_N) \wedge Entering(F_\Phi)$$

$actions(c_N)$表示单元格c_N的中的执行操作，x_N是对应于单元格c_N的时间变量集，$\Delta(x_N)$为该时间变量对应事件的估算时间。F_Φ为当前单元格触发的故障模式，$Entering(F_\Phi)$表示当该单元格存在故障触发函数Φ，则执行故障模式中相应$Entering$函数。

模型状态转换需要由事件触发，事件分为内部事件和外部事件：内部事件由c_N的操作触发，外部事件需要由与模型交互的环境触发。在故障模型中，故障本质上也可以视为一类事件，但为了完成原型模型故障模型的隔离，故障事件仅存在于故障模型中。以下是外部事件 e 触发时，模型需要进行的处理。$e[k]$是个布尔变量，表示事件 e 在第 k 步的事件状态。

$$ext(e)[k] \equiv \left(\neg e[k-1] \wedge e[k] = true \wedge\wedge_{x \in X(D)/e} x[k] = x[k-1]\right)$$
$$\vee \left(e[k-1] \wedge\wedge_{x \in X(D)} x[k] = x[k-1]\right)$$

于是，得到第 k 步转移关系的符号编码为：

$$step[k] \equiv \wedge_{i=1}^{n} \left(\left(\vee_{c \in M_i.C_{normal}} c_N[k]\right) \vee \left(\vee_{e \in M_i.e_{external}} ext(e)[k]\right)\right)$$

2.4 基于 FTSTM 的可信验证框架

工业控制网络协议的可信性验证通常是指对工业控制网络协议的安全性、可靠性和正确性等方面进行评估和验证，以确保工业控制系统的安全和稳定性。本文主要关注工业控制网络协议本身的正确性与可用性，同时在一系列故障发生时，能否保持系统正常运行或不会出现重大的影响，具有一定的容错性与鲁棒性。为此设计了一套基于故障时间状态迁移矩阵（FTSTM）的可信性验证框架。

该可信性验证框架主要分为四个主要步骤：协议建模与属性提取、故障模式定义与故障扩展、属性验证与故障树分析（Prototype Model，PM），（所谓原型模型，是指根据协议描述建立的未经验证完善的初始模型）。框架中使用到了课题组自研的工具包（FTSTM-based Fault extension Toolkit，FFET），可用于实现故障扩展与故障树生成。

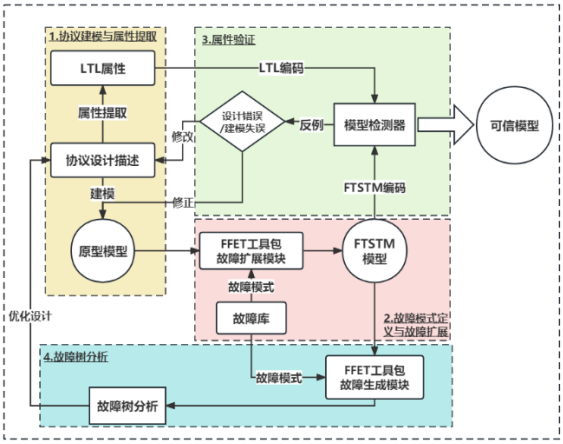


图 2　基于 FTSTM 的可信验证框架

为验证工业控制网络协议的可信性设计验证框架如图 2 所示。其主要包含以下四个关键步骤：

（1） 协议建模与属性提取：通过从协议设计描述中抽象出原型模型[18]，同时提取相应的可信性属性约束，并使用 LTL 语句进行描述。其中，由于整个模型建模迭代过程都是以属性约束为参照，因此属性约束的提取质量直接决定了最终可信模型的质量。

（2） 故障模式定义与故障扩展：使用预定义或自定义的故障模式构建系统故障模式，结合所建立的原型模型，生成 FTSTM 模型。

（3） 属性验证：通过配置将所需验证的模型与协议的 LTL 规约属性同时输入到模型检测器[19]中。模型检测器同一时间只能针对一个模型验证一条语句。将待检验的模型与 LTL 语句同时进行编码，输入到模型检测器，如若输入的某个模型符合某条 LTL 语句，则模型检测器返回 TRUE，否则返回违背该语句的反例。协议设计人员可通过跟踪反例路径，辨别出现该反例的原因，若是建模失误就修正原始模型，若是协议设计缺陷则改进协议设计并迭代原型模型。

（4） 故障树分析：为给定的顶层事件构造出故障树。通过对故障树进行分析，进一步优化模型设计。优化模型的目的并不要求模型设计能对所有的故障进行处理，而是通过故障树分析一些重要影响事件的形成过程，使协议设计中有意识的增加阻断该过程的机制。

协议设计与协议模型经过了上述过程的反复迭代更新之后，得到最终的可信模型。

2.5 基于 FTSTM 的故障树生成

二元决策图技术可以用来生成故障树[20]。首先定义系统或过程的故障模式，即确定故障的顶事件（Top Event）和基本事件。顶事件是系统最不愿意发生的事件，基本事件是可以直接测量或检测到的故障事件，即不能再进一步分解的最小事件，本文中也就是基本故障事件，将这些基本事件作为故障树的叶节点。故障树的顶节点是顶事件，底部是基本事件的叶子节点。构造 BDD 的简单方法是先构造一个二元决策树（Binary Decision Tree，BDT），然后逐步消除冗余并标识相同的子树，然而直接构建完整的二元决策树容易出现组合爆炸，另一种方法是在构建 BDD 的过程中同时进行 BDD 的归约化简过程。构造过程如算法 1 所示，为了提升故障树构建的效率，首先将顶事件划分为若干简单子事件，记为派生故障，与派生故障相关的基本故障应尽量少，但不为一。而后利用模型检测工具验证基本故障组合与派生故障的关系，构建出二元决策树。将二元决策树进行化简规约得到二元决策图。最后使用模块连接法[37]将派生故障的二元决策图合并化简，得到最终的二元决策图，并据此绘制故障树。

算法 1： 故障树绘制

输入： FTSTM 模型 M；顶事件 T；基本故障集合 BaseFaultSet

输出： 故障树

```
eventSet = T.splitByLogic(); //将顶层事件按照逻辑门划分为若干子事件
For event in eventSet:
    For bf in BaseFaultSet:
        If event. Affected(bf):     //基本故障 bf 与子事件 event 相关
            affectSet.add(bf);
    bdt=Check(affectSet,event); //通过模型检测器生成关于 event 的 BDT
    bdd=BDD(bdt);                //将 BDT 转为相应的 BDD
    bddSet.add(bdd);
 For bdd in bddSet:
     Bdd.merge(bdd);        // 通过逻辑门将子 BDD 合并连接并化简
  DrawFaultTree(Bdd);       //根据最终的 BDD 绘制故障树
```

3 实验分析

3.1 协议建模

本章将应用基于 FTSTM 进行故障扩展的可信验证框架进行案例研究，对照 GB/T 27960-2011 标准的以太网 PowerLink 通信行规规范，对协议的数据链路层进行可信性分析。

PowerLink 是一种实时通信协议，常用于工业控制数据的采集和传输[21]。协议的一个通信周期可以划分为三个阶段：同步阶段、异步阶段和空闲阶段。这三个阶段构成了协议的一个循环。

协议的节点类型是管理节点（Managing Node，MN）和受控节点（Controlled Node，CN）。通常，一个 MN 控制多个 CN。PowerLink 三个通讯阶段的主要功能分别是：

（1） 等时同步阶段（SYN）：在等时阶段开始时，MN 首先广播循环 SoC 帧的开始，并将时钟与所有 CN 同步，然后依次向每个节点发送轮询请求 PReq 帧。CN 将要发送的所有信息封装到轮询响应 PRes 帧中，并通过广播将其发送到 MN 或其他 CN。当 MN 完成与所有 CN 的交互时：成功接收 PRes 帧或等待超时，然后进入下一阶段。

（2） 异步阶段（ASND）：MN 广播异步 SoA 帧的开始，以标记异步阶段的开始。CN 根据需要向 MN 发送异步数据，并通过异步发送 ASnd 帧报告请求。MN 根据接收到的请求的顺序进行响应。

（3） 空闲阶段（IDLE）：由于每个周期中异步阶段的时间是不确定的，因此在每个周期的间隔中填充一定的空闲时间，以确保协议每个周期长度的一致性，以此控制各节点的周期同步。

下面对两个通讯实体分别建模，如图 3-4，其中 CE_SOC 表示 CN 收到了 SOC 帧，其他事件命名规则相同，isochr 表示 MN 需要轮训的次数，SoAProcess 表示该节点时候获得授权，CEV_LOSS_表示相应的数据帧丢失，X_TRIG 为外部变量，通过触发该变量进入通讯过程。

DLL_CS	S	IDLE	SYN		ASND	
E		0	1		2	
CE_SOC	0	/	CEV_LOSS_SOA=true;		CEV_LOSS_SOA=true;	
		1	1		1	
CE_PREQ	1	ME_PRES=true; presNum=presNum+1; CEV_LOSS_SOC=true;	presNum!=1	else	presNum!=1	else
			ME_PRES=true; presNum=presNum+1;	ME_PRES_TIMEOUT=true; presNum=0;	ME_PRES=true; CEV_LOSS_SOC=true; CEV_LOSS_SOA=true; presNum=presNum+1;	ME_PRES_TIMEOUT=true; CEV_LOSS_SOC=true; CEV_LOSS_SOA=true; presNum=0;
		0	2	2	0	0
CE_PRES	2	/	presProcess=1;		presProcess=1;	
		0	1		2	
CE_SOA	3	CEV_LOSS_SOC=true; SoAProcess=1;	SoAAuthorization==1	else	SoAAuthorization==1	else
			ME_ASND=true; SoAProcess=1; Δ(ME_ASND);	SoAProcess=1;	ME_ASND=true; SoAProcess=1;	SoAProcess=1;
		0	0	0	0	0
CE_ASND	4	asndProcess=1;	CEV_LOSS_SOA=true;		CEV_LOSS_SOA=true;	
		0	1		2	

图 3　CN 通讯过程模型

DLL_MS	S	IDLE	SYN		ASND	
E		0	1		2	
ME_PRES	0	×	isochr!=0 &&CE_PRES!=0	else	/	
			Δ(time_PRES); Rec_PRES++; presNum--;	ME_ASND=true; CE_ASND=true; SOA_Process=true; ME_PRES=false;		
		0	1	2	0	
ME_ASND	2	/	×		isochr!=0 &&CE_PREQ	else
					Δ(time_ASND); presNum--;	CE_SOA=true; DLL_ME_PRES=false;
		2			2	0
X_TRIG		ME_PRES=true;	/		/	
		1	1		2	

图 4　MN 通讯过程模型

Fault_Mode	S	Entering	During
E		0	1
Normal_Mode	0	ϕ	ϕ
		1	1
Fault_ frozen_isochr	1	Frozen_ent(isochr); ϕ	Frozen_dur(isochr); ϕ
		1	1
Fault_ randomDelay	2	randomDelay_ent (timer);	randomDelay_dur(timer); ϕ
		1	1

图 5　故障模型

结合 PowerLink 在实际的应用场景，选择出现概率较高的 Fault_frozen、Fault_randomDelay 故障注入。

Fault_randomDelay：在矩阵 DLL_CS(0，0)中插入 Fault_randomDelay 故障模式，设置 MaxDelay 参数为 5。该故障表示 MN 与 CN 通讯过程中发生了随机 0 到 5 个时间单位的网络延迟，故障可能的发生原因是因遇到强干扰或网络拓扑结构发生变化而导致的通讯链路不稳定。

Fault_frozen：在矩阵 DLL_MS(1，0)中插入 Fault_stuckByValue 故障模式，使得 isochr 卡死值为 2。该

故障表示需要轮询的 CN 数为 2，而此时的 CN 数为 1，即需要轮询的 CN 数大于实际的 CN 数，出现该故障的可能原因是有 CN 发生故障致使丧失功能。结合生成的故障模型如图 5 所示。

3.2 属性验证

为了对模型进行可信性验证，列出 5 条 LTL 语句，如表 3 所示：

表 3 针对 PowerLink 的 LTL 约束

属性名称	LTL语句
Unreachable	[G]!((active(DLL_MS)==1)&&(DLL_ME_ASND==true))
Dynamatic1	[G]((active(DLL_MS)==1&&[N](active(DLL_MS)==2))⇒active(DLL_CS==2))
Logic1	[G](DLL_CE_PREQ==true⇒[F]DLL_ME_PRES==true)
Logic2	[F]([G]SoAAuthorization>0)
noDeadLock	![F]([G](X_TRIG)||[G](! X_TRIG))

Unreachable 表示当 MN 处于周期的开始状态时，前一个周期的所有通信任务应该已经完成。如果 MN 此时接收到 ASnd 帧，则发生协议规范中未定义且无法处理的异常。异常定义为模型中不允许到达的错误单元格。Dynamaic1 指当 MN 从同步相位转移到异步阶段时，CN 也必须处于异步阶段。Logic1表示在 CN 发送请求之后，MN 必须响应。Logic2 表示 MN 在向 CN 授予授权之后授权节点记录的编号总是有效的。noDeadLock属性表示外部事件“X_TRIG”不再发生改变。为了检验模型的容错能力，分别激活故障并对上述 LTL 语句进行验证，结果如表 4 所示，PowerLink 协议设计对于网络延时有较高的容错性，而对于卡值等故障则较容易导致失败。

表 4 故障模型验证结果

属性	Fault_frozen激活	Fault_randomDelay激活
Unreachable	√	√
Logic1	√	√
Logic2	×	√
Dynamic1	×	√
no Deadlock	√	√

4 结语

本文结合形式化技术与故障注入技术提出，提出一种故障时间状态迁移矩阵（FTSTM）的形式化建模方法。并设计一种基于 FTSTM 的可信验证框架。通过使用线性时序逻辑对协议需求进行属性规约，并对原型模型进行故障扩展，而后将故障模型输入模型检测器进行验证，验证结果用于优化协议设计与迭代原型模型。同时，使用二元决策图技术生成故障树，用于对模型的故障发生过程进行研究分析，从另一个角度为协议设计优化提供了思路。最后，针对常用的工控网络协议 PowerLink 进行 FTSTM 建模与验证，验证协议的容错能力，例证了方法的有效性。

参 考 文 献

[1] Krotofil M, Gollmann D. Industrial control systems security: What is happening?[C]//2013 11th IEEE International Conference on Industrial Informatics (INDIN).IEEE,2013:670-675.

[2] Mo Y, Kim H J, Brancik K., et al. Cyber–Physical Security of a Smart Grid Infrastructure[J]. Proceedings of the IEEE,2011,100(1):195-209.

[3] Krotofil M, Gollmann D. Industrial control systems security: What is happening?[C]//2013 11th IEEE International Conference on Industrial Informatics (INDIN).IEEE,2013:670-675.

[4] 何之栋,裘坤,钟晨,王海凤.工业控制系统的信息安全问题研究[J].工业控制计算机,2013,26(10):1-4.
[5] Tuptuk N, Hailes S. Security of smart manufacturing systems[J]. Journal of manufacturing systems,2018,47:93-106.
[6] Shahzad A A, Musa S, Aborujilah A, et al. Industrial control systems (ICSs) vulnerabilities analysis and SCADA security enhancement using testbed encryption[C]//Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication.2014:1-6.
[7] Ani U D, He H, Tiwari A. Human factor security: evaluating the cybersecurity capacity of the industrial workforce[J]. Journal of Systems and Information Technology,2019,21(1):1-35.
[8] 赖英旭,杨凯翔,刘静,刘增辉.基于模糊测试的工控网络协议漏洞挖掘方法[J].计算机集成制造系统,2019,25(9):2265-2279.
[9] Bertolotti I C, Durante L, Maggi P, et al. Improving the security of industrial networks by means of formal verification[J]. Computer Standards & Interfaces,2007,29(3):387-397.
[10] Bugliesi M, Calzavara S, Focardi R, et al. CookiExt: Patching the browser against session hijacking attacks[J]. Journal of Computer Security,2015,23(4):509-537.
[11] Chan C C, Yang C Z, Fan C F. Security verification for cyber-physical systems using model checking[J]. IEEE Access,2021,9(1):75169-75186.
[12] Zhang J, Yang L, Gao X, et al. Formal analysis of QUIC handshake protocol using symbolic model checking[J]. IEEE Access,2021,9:14836-14848.
[13] Kunnappilly A, Backeman P, Seceleanu C. From UML modeling to UPPAAL model checking of 5G dynamic service orchestration[C]//7th Conference on the Engineering of Computer Based Systems,2021:1-10.
[14] Yamane S, Uemura K. Comparative experiment of SPIN and SMT in model checking of embedded assembly program[C]//2020 IEEE 9th Global Conference on Consumer Electronics (GCCE),IEEE,2020:54-57.
[15] Villani E, Pontes R P, Coracini G K, et al. Integrating model checking and model based testing for industrial software development[J]. Computers in Industry, 2019,104:88-102.
[16] 侯刚, 周宽久, 常军旺, 等. 基于时间 STM 的软件形式化建模与验证方法[J]. Journal of Software,2015,26(2):223-238.
[17] 李润泽,姜斌,余自权,等.基于数据驱动的无人机集群故障检测与诊断[J].北京航空航天大学学报，2022,48(1):1-9.
[18] Bertolotti I C, Durante L, Maggi P, et al. Improving the security of industrial networks by means of formal verification[J]. Computer Standards & Interfaces, 2007,29(3):387-397.
[19] Kong W, Katahira N, Watanabe M, et al. Formal verification of software designs in hierarchical state transition matrix with SMT-based bounded model checking[C]//2011 18th Asia-Pacific Software Engineering Conference. IEEE,2011: 81-88.
[20]郎荣玲.故障树转化为二元决策树的算法研究[J].计算机工程与应用,2008,622(27):69-71.
[21]胡书立,王清理.POWERLINK 总线技术的研究与实现[J].计算机工程与设计,2012,33(10):3821-3827.

作者简介部分

王洁：男，副教授，研究方向为并行体系结构、异构计算、演化硬件与容错计算、网络空间安全.
吴鑫涛：男，硕士生，研究方向为形式化验证、模型检测
安然：女，硕士生，研究方向为形式化验证、机器学习.
范搏栋：男，硕士生，研究方向为模型检测、容错计算.

机器人路径规划算法的高层综合设计研究

赖李洋[①*]，郑锆骏[①]，梁海成[①]，李华伟[②]

[1](汕头大学电子系　汕头　515000)

[2](中国科学院计算技术研究所计算机体系结构国家重点实验室　北京　100190)

摘　要: 随着机器人自动导航技术的快速发展，基于软件实现的路径规划算法在实时性上已无法满足许多应用场景的需求，这就要求对算法进行快速高效的硬件定制，从而获得低延时的性能加速。本研究以机器人路径规划中的经典 A*算法为对象，通过构建面向硬件设计的 C/C++数据结构和函数流程优化，采用高层综合（High Level Synthesis）实现快速的硬件架构探索和选取较优的设计方案，并完成硬件 FPGA 综合。实验数据表明，相较于传统 RTL 开发模式，基于HLS 开发模式的路径规划算法在 FPGA 实现上在开发效率、硬件性能和资源占用率上都有显著提升，验证了高层综合在硬件定制中的可行性和成本优势。

关键词: 机器人自动导航；路径规划算法；高层综合；算法硬件加速

Case Study of High Level Synthesis on Robotic Path Planning Algorithm

Lai Liyang[1], Zheng Peijun [1], Liang Haicheng[1], and Li Huawei[2]

[1)] (*Department of Electronics and Information Engineering, Shantou University, Shantou*　515000)

[2)] (*State key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing*　515000)

Abstract: With the advancement of robot automatic navigation technology, software-based path planning algorithms can no longer satisfy the needs in scenarios of many real-time applications. Fast and efficient hardware customization of the algorithm is required to achieve low-latency performance acceleration. In this work, high level synthesis (HLS) of classic A* algorithm is studied. Hardware-oriented data structure and function optimization, varying design constraints are explored to pick the right architecture, which is then followed by FPGA synthesis. Experimental results show that, compared to the conventional RTL method, the HLS-based FPGA implementation of the A* algorithm can achieve better productivity, improved hardware performance and resource utilization efficiency, which demonstrates the advantages of high level synthesis in hardware customization in algorithm-centric applications.

Key words: Automatic navigation; mapping algorithm; path panning algorithm; high level synthesis; hardware acceleration

1　引言

机器人自动导航技术在现代工业中有着广泛的应用。随着工业互联网和人工智能技术的快速发展，人们对机器人自动导航[10]的性能要求越来越高，希望它们面对现实环境的变化做出更加实时智能的反应。传统的基于软件实现的导航算法由于需要较长的运行时间，在实时性上已无法满足许多应用场景的需要。因此导航算法的性能加速是需要解决的一个关键问题[1]。

将算法综合成硬件，一方面可以有效解决在纯软件导航算法的性能不足的问题，另一方面可以大幅降低系统的整体功耗，在许多功耗受限的应用场景中有显著的优势。传统上算法设计和硬件设计属于不同的领域，是分别由软件工程师和硬件设计工程师来实现的，采用不同的编程语言。算法通常是由 C/C++/SystemC 实现，而硬件则由硬件描述语言 HDL 来定义。软件工程师通常没有硬件设计的概念，在算法的硬件定制中需要有专门的设计工程师详细分析算法的数据结构和函数实现，手动设计合适的数据通路模块和控制逻辑，以实现特定的算法功能。但这往往面临开发周期长，人力资源开销大，设计结果验证

调试困难等问题，不符合市场的经济效益[2]。

高层综合（HLS）[3]，即为提高系统设计从软件算法到硬件实现的转换效率而应运而生。它有着几十年的历史，高层综合的定义也随着它的发展而不断演化[9]。在今天，高层综合指的是将高级编程语言实现的算法功能，自动转换成寄存器传输级（RTL 级）的硬件模型的过程。

使用高层综合进行硬件开发[16]，可以大大提高硬件设计的效率。例如，一个拥有一百万逻辑门的芯片设计通常需要编写数十（比如 30）万行的 RTL 代码，而使用 C/C++等语言对算法进行高层次的建模可以将代码的密度压缩 7-10 倍，同时可以去掉使用 RTL 代码进行编程过程中繁琐而且耗时的调试过程。基于 RTL 实现的 IP 核在重用时，通常需要重新定义固定的架构和接口标准，这不仅需要花费大量的时间进行系统互联和端口验证，同时也消耗大量的人力。而高层次综合则可以提高对 IP 核的复用率[4]。在进行设计时，更多关注顶层算法的设计与硬件映射关系，因此，可以促进算法工程师和软件工程师主动参与甚至主导芯片设计的开发设计。

当然，在一些对芯片性能、 面积和功耗有严格要求的场景中（比如通用 CPU），仍然会直接使用硬件描述语言（如 Verilog、VHDL 等）进行定制设计。但是如果结果要求较为宽松，对设计后得到的结果可能存在的风险较为不敏感，使用高层综合则有其独特的效率和成本优势。同时，若顶层算法设计合理，高层综合得到的结果可以接近甚至超过使用定制硬件描述语言设计得到的结果[5]。

本研究对路径规划算法中的 A*算法进行面向硬件设计的数据结构和函数流程优化，采用高层综合产生 RTL 代码，并运行 FPGA 综合生成硬件原型，同传统 RTL 开发模式实现的 A*算法相比，所获得的 FPGA 硬件不论是在算法性能和资源占用率上，还是在开发效率上，都有显著的提升。

2 高层综合的流程简介

虽然高层综合可以将基于高层语言（比如 C/C++）的算法自动转换成基于 RTL 的硬件模型，但这并不意味着任意的 C/C++语言特性都适合综合，这点同逻辑综合只适用于特定的 HDL 子集类似[11]。高层综合需要对算法的数据结构和函数流程进行优化，并满足必要的编程规范。

以西门子公司的工具 Catapult[8]为例，高层综合主要采取如图 1 所示的流程，首先在 Linux 系统下或者其他系统中完成 C/C++算法的正确编写，然后使用面向高层综合的位精确数据类型（如 ac_int/ac_fixed）去代替 C/C++算法中的常用内建数据类型（如 int，double 或 float 等）[19]；接着设计顶层模块，并且优化函数实现，以及改写高层综合不支持的函数类型，如指针函数，常规的递归函数和使用了动态内存分配的库函数等；函数优化和改写完成后，还需要编写测试函数和设定工具的约束命令脚本；然后使用高层综合工具进行综合，观察结果是否通过，如果不通过则检查问题出现的原因并将之解决，如果综合通过则查看综合参数是否理想，若不理想，则对相关函数进行再优化或是修改综合约束条件等，直到得到一个理想的结果。

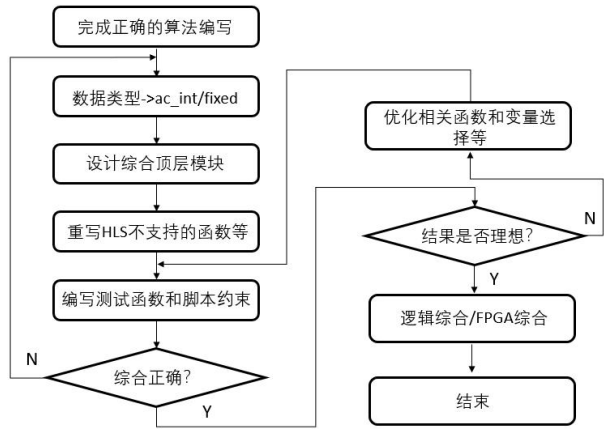


图 1 实现高层综合的基本流程

需要明确的一点是，高层综合时存在很多的规则和限制条件，不是所有的 C/C++算法描述都可以进行硬件综合并得到理想的综合硬件结果。这一点同逻辑综合类似，在逻辑综合中也不是所有的 HDL 都可以综合成网表，同时，为了产生好的综合结果，需要 HDL 满足特定的编写规范。

到目前为止有两种位精确数据类型（Bit Accurate Data Types）的行业标准，分别是 SystemC 和西门子公司（原 Mentor 公司）的 Algorithmic C 数据类型[15]。

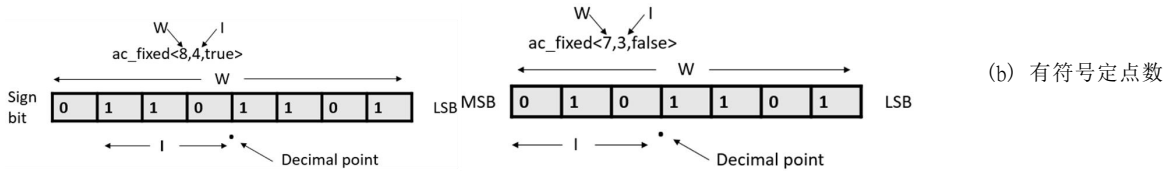
位精确数据类型已经被算法工程师和硬件设计工程师使用了多年，但是 C/C++中的基本内建数据类型（如 char，short, int，double）的宽度是 8 位、16 位、32 位、或 64 位等。因此在高层综合中，为了更好的与硬件进行匹配，需要将 C/C++的内建数据类型替换成位精确数据类型。在 Catapult 中，Algorithmic C 中的位精确数据类型能更好地将 C/C++与 RTL 语言的硬件综合对应起来，得到比传统的位精确数据类型更好的硬

件结果。举个例子，公式（1）和公式（2），是位精确整数变量的声明方式。其中 W 表示的是位宽，false 代表变量是无符号整数，true 则表明变量是有符号整数。

$$ac_int<w,false>x \quad (1)$$

$$ac_int<w,true>y \quad (2)$$

如图 2 所示 ac_fixed<W, I, true | false>可定义位宽 W 且整数为 I 位的定点数，true 或者 false 表示有符号或是无符号。图 2(a)表示的是 7 位无符号定点数，3 表示整数部分的位数，即头 3 位是定点数的整数，后 4 位是定点数的小数部分；图 2(b)表示的是一个 8 位有符号定点数，头 4 位用于表示整数位（包含最高位的符号位），余下的低 4 位是小数位。高层综合中，还有一个重要的



（a）无符号定点数　　（b）有符号定点数

图 2 定点数类型

切片函数（英文全称为 slice）。

该函数可对整数的指定位或位域进行读写操作，用于替代算法中常用的移位和位与等操作。图 3 是一个读操作的例子。

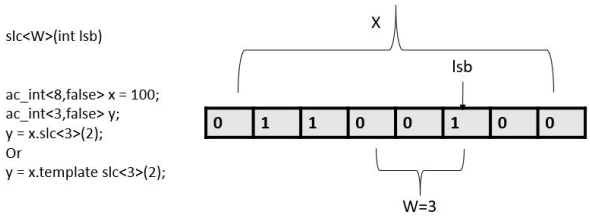


图 3 切片函数的读操作

图 3 中，左边代码的第一行是切片函数的读操作的基本使用方式，其中 W 代表插入数据的位宽，lsb 代表将要插入数值的最低位。图中代码给出了一个例子，其对应于图中右边的情况。给定一个八位的无符号整数 x，其值是 100，读取 x 中的第 2 至第 4 位的值并赋值给位宽为 3 的无符号整数 y。例子中切片函数的读操作写法有两种：x.slc<3>(2)和 x.template slc<3>(2)。区别在于有无 template 关键字。在某些编译器平台上，使用模板函数需要加入 template 关键字，用户可以根据自己的实际情况选取适合的写法。

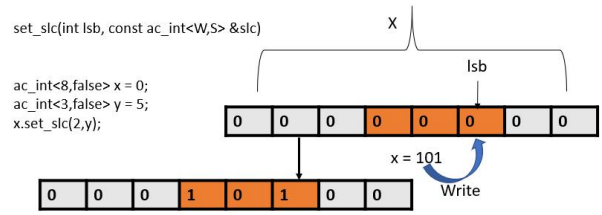


图 4，切片函数的写操作

图 4 中左上角的代码定义了切片函数写操作的语法，其中 lsb 代表的是写入位域的最低位。而要被写入的数值必须是一个常量。后三行的例子中，x 是一个 8 位的无符号整数，初始值为 0，y 是一个位宽为 3 的无符号整数，其值是 5，y 将被写入到 x 的 2-4 位中，这将使得 x 值将变更为 20。

在进行高层次综合时，普通递归函数方式是不支持的。符合要求的递归函数需要具备以下条件，对顶层递归函数进行模板化，同时要有一个专用模板函数，用作函数退出条件。

动态内存分配是从操作系统中获取未被使用的内存的一个过程，尽管内存不是无限的，但是不用担心内存不够而导致分配失败。而高层综合不支持动态内存的使用。所以，当算法程序中出现动态内存分配时，我们需要改写代码，以固定的预分配内存去替代。函数指针也是不支持的，因此需要用普通的函数调用去替代。

另外，完成高层综合后，还要进行逻辑综合或 FPGA 综合等步骤，以完成最终的硬件实现。

3 路径规划 A*算法的优化与综合

本研究以机器人路径规划的经典算法 A*算法作为研究对象[20]。A*算法是一个计算点对点路径的快速启发式算法，从某种程度上来说，它是经典 Dijkstra 算法[6]（计算单点到其余多点的最短路径）一个改良型算法。在介绍 A*算法之前，

我们先定义一下算法描述中一些基本术语。

- 栅格地图：由格点构成的地图，路径是沿着格点行进的。
- 起点：路径的起始格点
- 终点：路径的目标格点。
- 路障格点：地图中不能通行的格点。
- F：路径搜索中格点的代价函数。
- 开启列表：路径搜索中下一个可能格点的集合，每一次取出 F 值最小的格点。
- 关闭列表：路径搜索中已走过的格点集合。

```
1 初始化开启列表和关闭列表为空。
2 将起点插入开启列表中。
3 while( 开启列表不为空 )
4   从开启列表中取出最小 F 值的格点作为当前格点，
    并将之加入关闭列表。
5   foreach( 当前格点的相邻格点 )
6     if（该格点在关闭列表中或是路障格点）then (跳过
      该格点)
7     elsif（该格点在开启列表中）then (比较通过当前格点
      计算得出的 F 值是否更小，若更小，则更新该格点
      的 F 值，并设其父节点为当前节点)
8     elsif（该格点不在开启列表中）then (通过当前节点计
      算该节点的 F 值，设其父节点为当前节点，并将之
      插入开启列表)
9     elsif（该格点为终点）then（设其父节点为当前节点，
      并通过追踪父节点链条至起点，输出起点到终点的
      路径，算法结束)
9   end_foreach
10 end_while
11 if（开启列表为空）then (路径搜索失败)
```

图 5 A*算法的基本流程

A*算法的输入为含路障格点信息的栅格地图、起点和终点，输出为从起点到终点的一条可通行路径（或是路径不存在），其伪码形式的基本算法流程如图 5 所示。首先初始化开启列表和关闭列表为空，并将起点添加到开启列表中。当开启列表不为空时，取出 F 值最小的格点，放入关闭列表中，并作为当前格点。接下来对当前格点的所有相邻格点逐个进行判断和数据更新。若该格点已在关闭列表中或是路障格点，则跳过该格点；若该格点已在开启列表中，则比较通过当前格点计算出的 F 值是否更小，若更小，则更新该格点的 F 值并将其父格点设成当前格点；若该格点既不在开启列表中，又不在关闭列表中，则将其插入开启列表中，并通过当前格点计算其 F 值和将其父格点设成当前格点；若该格点为终点，则设置其父格点为当前格点，并通过追踪父格点链条关系输出生成路径，算法结束。如果开启列表最终为空，则路径不存在。

代价函数 F 对于 A*算法效率有关键的影响，它是 G 和 H 两部分组成，即 F = G + H。其中 G 代表的是从起点到待处理格点的代价；H 是指从待处理格点到终点的代价。在本文的实现中，G 采用的是从起点到待处理格点累计走过的距离；H 使用的是曼哈顿距离，即栅格地图中待处理格点和终点坐标在 x 轴和 y 轴方向的距离（绝对值）之和。

图 6 是一幅栅格地图，其中绿色格点是起点，红色格点为终点，棕色格点是路障格点。定义上下左右相邻格点的距离为 10，斜向相邻格点距离为 14。起初开启列表中只有起点，将其从开启列表中取出，并且将其加到关闭列表中，由于其八个相邻格点都符合图 5 中算法流程的条件 8，所以八个相邻格点均加入到了开启列表中，并且 F 值和父格点（箭头所示）都进行了更新。以图 6 中黄色格点为例，其 G 值是 14（由于它是当前格点的斜向相邻格点）；H 值是 40（此格点与终点在 x 轴方向的距离是 30，在 y 轴方向的距离是 10）；F 值是 G+H，为 54；其父格点为当前格点，箭头指向左下。

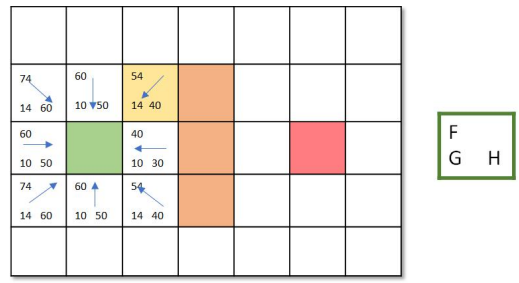


图 6 格点数据更新过程示例

运行图 5 所示的算法流程，最后可以得到图 7 所示的结果，根据终点的父格点链接关系依次索引回去，即从终点开始的格点内箭头开始，依次追踪下一个格点的箭头方向，可得到一条合法的由终点到起点的路径，如黄色格点所示。当然，最终的起点到终点的路径只是把上述路径的方向反转过来。需要注意的一点是，并不是所有的格点在路径搜索过程中都会被访问到，

图中绿色粗线所覆盖的区域，是这个例子中路径搜索所访问过的格点。

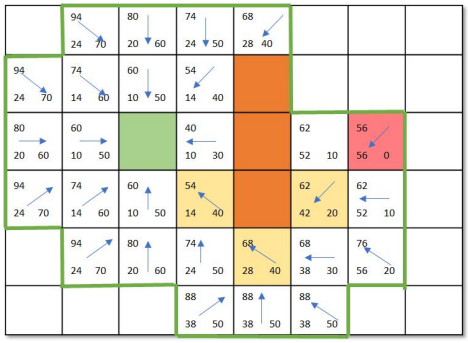
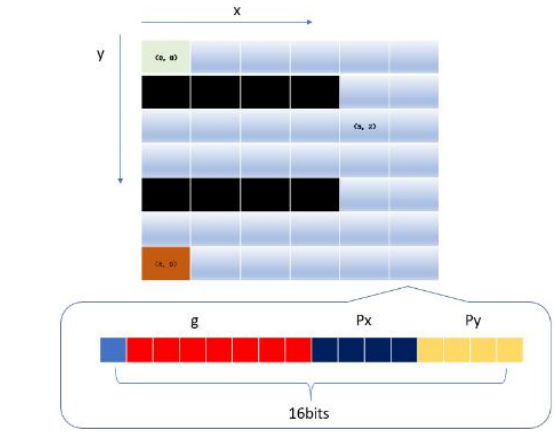


图 7 路径搜索结果

3.1 数据结构优化

A*算法的 C/C++实现通常是以结构体或者类的方式存储相关数据，这些信息一般采用的是内建数据类型，比如字符、短整数、和长整数等。由于算法中处理的地图规模有限，并不需要使用到内建数据类型的所有比特位，这对于定制硬件来说是一种浪费。另外，现有的高层综合工具不支持 C++标准模板库（STL）（比如 vector、map 等），虽然它们使用广泛，但实际上分配了远超硬件运行所需要的内存空间，所以为了降低算法的定制硬件开销，必须对数据结构进行面向硬件设计的优化。

举个例子，地图的每个格点需要存储的信息有父格点坐标，路障信息，以及 F、G 和 H 值。为了降低硬件内存，我们使用了一种位精确的紧凑数据结构去存储格点的相关信息。一个大小为 n*m 的二维数组用来表示一个 n*m 的栅格地图，其单个数据的宽度是由地图规模的上限来确定的。如图 8(a)，当地图规模为 16*16 时，可用 16 位无符号整数的二维数组来表示，其低 8 位用于存器格点的父节点坐标；紧接着的高 7 位用于存储格点 G 值，因为 F=G+H，且 H 值很容易通过当前格点和终点的坐标来计算，故只需存储 G 值便可得到 F 值；最高位则用于标识格点是否可通行，1 标识为可通行，0 标识为障碍物。图 8(b)是采用这种位精确类型的两个具体例子，第一个标注格点为可通行格点，其 G 值初始为 127，父格点初始为（16，16）；第二个标注格点为路障格点。对于格点坐标等信息可通过切片函数 slc()和 set_slc()直接进行读写访问，这比通常 C 实现中的移位和位与操作要有效的多。



(a) 格点的信息存储

(b) 格点的信息存储示例

图 8 格点信息的数据结构

A*算法中另外一个重要的数据结构是开启列表，需要存储路径搜索中的候选格点，并每次取出 F 值最小的格点。开启列表是由一维数组来实现的，每个元素包含该格点的 F 值和坐标。对于一个最大为 16*16 的地图，位精确优化后的数据类型是 16 位的无符号整数，如图 9，其低 8 位是格点坐标，高 8 位用于存储格点的 F 值。开启列表初始化所有元素为最大值（即全 1）。

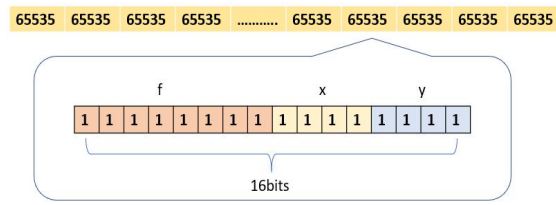
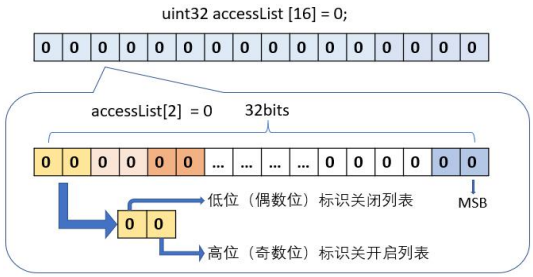
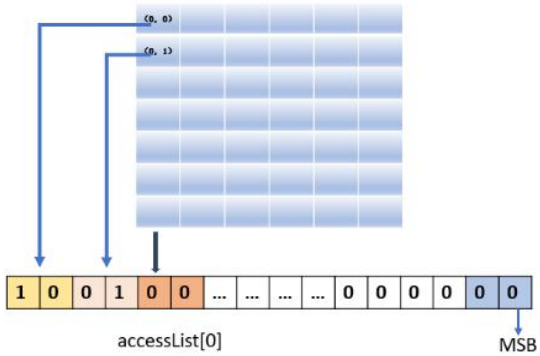


图 9 开启列表数组

除了开启列表，另外一个访问频率很高的数据结构是标识数组 accessList[]，该数组是用来标识一个格点是否在关闭列表中或在开启列表中，位精确优化后的标识数组的实现如图 10a 所示。数组元素中每两位用来标识一个格点，高位用于标识格点是否在开启列表中，低位用于标识格点是否在关闭列表中。如图 10(b)，对于最大为 16*16 的地图，数组第零个元素可以标识地图第一行的所有元素，假定（row, column）表示格点的行与列，前两位“10”表示格点（0，0）已经在关闭列表中，后两位“01”代表格点（0，1）在开启列表中。“00”则代表格点还未被访问到；“11”为非法状态，因为一个格点不可能同时出现在关闭列表和开启列表中。



(a) 标识数组结构



(b) 实例

图 10，标识数组

3.2 函数优化

A*算法中一个重要函数是从开启列表中取出 F 值最小的格点。研究中使用的 C 开源版本[7]A*算法使用一个 N*M（即栅格地图的大小）的一维数组来存储开启列表。因为一个格点最多有 8 个相邻格点，开启列表数组尾部预留了 8 个位置作为新插入格点的位置。每次新格点会依据它同当前格点的相邻关系插入到其中的一个位置。

每次从开启列表中弹出一个格点时，算法会对开启列表做一次快速排序[6]（Quick Sort），这样开启列表的第 0 个元素即是最小 F 值的格点，如图 11 所示。随后该格点被弹出，其位置上的值用全 1 所覆盖，这样下一次排序时该位置会自动被最小 F 值格点所占用。由于快速排序是递归算法，预期的时间复杂度是 O(n lg n)，硬件复杂度大，综合后得到的结果不理想。并且当栅格地图规模扩大时，会出现综合时间过长或综合失败的问题。

图 11 原始算法中的开启列表操作

为了降低原始算法的复杂度，我们在图 11 版本的基础上，使用优先队列排序函数[8]替代快速排序函数对开启列表里面的元素进行排序。排序完成后，开启列表里面的元素将以最小顶堆的顺序在列表中呈现，其中表头为最小顶堆的顶点，即 F 值最小的格点 F 值。当需要从开启列表中取出 F 值最小的格点时，只需在列表表头取出对应的元素即可。同时，使用一个寻找“空元素”（即值为全 1 的元素）的函数找到最后一个非空元素，并将它列表置于表头。当需要插入新的格点时，先找到第一个空元素在列表中的位置，然后将新的格点置于第一个空元素的位置上。上述操作的具体流程如图 12 所示。这样做的好处是让开启列表中所有的有效格点都集中在列表前端，使得每次排序操作只需对列表前端的有效元素进行排序即可，而不是对整个列表的所有元素完成排序操作，大幅度地提升了排序效率。并且使用优先队列方式设计的开启列表，其有效格点数一般在（2~5）*ROW 区间，说明区间性的排序操作更加合理。

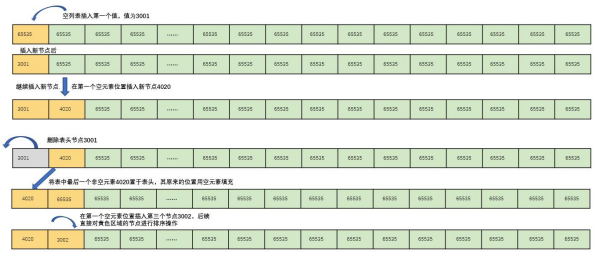


图 12 开启列表操作优化

4 实验结果

本研究的所有实验都是在一台 Linux 服务器上运行的。服务器的配置为 CentOS 7.4，单 Intel i9-10900K CPU，64GB DDR4 内存。CPU 主频 3.7GHz，CPU 为 10 核 20 线程。服务器上运行的商用工具来自西门子公司（原 Mentor 公司），包括高层综合工具 Catapult Ultra，逻辑仿真工具 QuestaSim。赛灵思公司（Xilinx）的 FPGA 综合工具 Vivado[18]。本文使用的 FPGA 开发板的主芯片是由赛灵思公司生产的 XCZU4EV-SFVC784-2I 芯片[13]。表 1 展示的是该芯片的 PL 资源情况。

表 1 SOC FPGA 芯片 PL 资源

资源名称	总数
逻辑单元 LUT	87840
LUTRAM	57600
触发器 Flip-flop（FF）	175680
BRAM	128
I/O 引脚	256
全局时钟缓冲器 BUFG	352

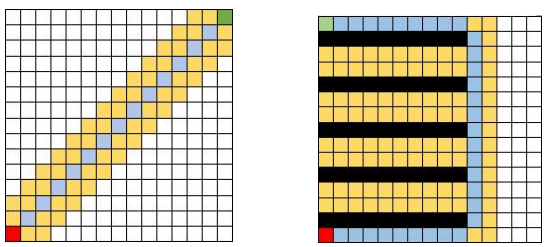
4.1 数据结构对比

在 A*算法中，使用不同的数据结构实现开启列表，所得的结果在路径搜索效率和硬件资源开销上都会有不同的表现。本节引入了优先队列（Priority queue）、队列（Queue）和堆栈（Stack）三种不同的数据结构来实现开启列表。其中，优先队列方案使用优先队列排序来搜寻列表中 F 值最小的节点，地图搜索时每次从列表中取出 F 值最小的点；而队列和堆栈方案中则没有使用任何排序手段，地图搜索时队列和堆栈分别采用先进先出和后进先出的方式取出列表中的节点。表 3 列出了在相同的综合约束条件和地图规模为 15*15 的环境下，三种不同的数据结构所得到路径搜索效率和相应的 FPGA 硬件资源开销情况[14]。其中表 2，运行时长表示的是路径搜索时间。

表 2，15*15 地图下不同开启列表优化比较

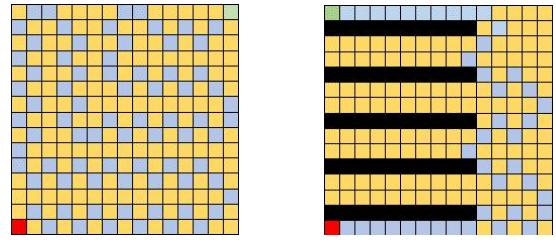
数据结构	LUT 个数	FF 个数	BRAM 个数	运行时长（ns）
优先队列	2485	2807	3	18,598
队列	2137	2755	2.5	44,325
堆栈	2148	2739	2.5	38,779

由表 3 中可知，不同的数据结构实现方式对最终的硬件实现结果有影响。从运行时长来看，优先级队列明显优于其余两者，是最快找到路径的。而其余两种数据结构在硬件资源上都有不同程度的优化，但这种程度的优化需要在更大的地图上才能体现，在后续的篇章中我们还有更大的地图实验，到时就可以看出堆栈和队列两种函数对硬件资源的优化程度。图 13、图 14、图 15 分别展示的是优先级队列、堆栈、队列三种数据结构所实现的地图路径。



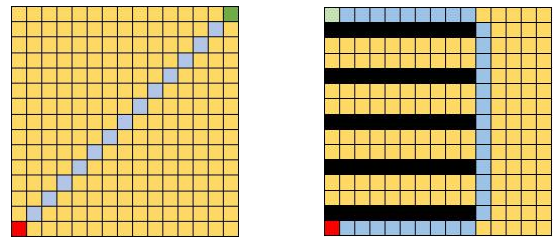
(a)地图一搜索路径　　(b)地图二搜索路径

图 13 优先级队列实现的地图路径



a.地图一搜索路径　　b.地图二搜索路径

图 14 堆栈实现的地图路径



a.地图一搜索路径　　b.地图二搜索路径

图 15 队列实现的地图路径

在以上三幅图中，红色是起点，绿色是终点，浅蓝色是我们最终的路径，黄色代表的是得出最终路径开启列表访问过的格点。可以明显的看到，优先队列得到最终路径的过程中所需要访问的格点数明显少于另外两种数据结构，这也侧面印证了优先队列搜索效率要优于另外两种数据结构。堆栈得到的地图搜索路径较为复杂，在访问格点的数量以及最终路径效果这两方面表现不佳。队列得到的搜索路径与优先队列的搜索路径相同，但是搜索的过程中需要遍历所有的地图格点，大大增加了路径搜索时长。

综上所述，与队列和堆栈两种方案相比，优先队列拥有排序功能，在地图搜索的过程中，能够在遍历更少的地图格点情况下获得最优搜索路径，从而降低了算法运行时长。优先队列方案的性能优化需要更多的硬件资源来实现，不过这种资源置换都是在可接受范围内，因此我们选用优先队列数据结构来实现 A*算法。

4.2 HLS 与传统 RTL 开发模式实验结果比较

为了能够直观地突出 HLS 开发模式中开发效率的优点，本文同时实现了同样设计需求下的人工手写的 RTL 代码。从算法实现的代码量来看，传统 RTL 开发模式的 RTL 代码大约为 2000 行，而 HLS 开发模式下手写的 C++代码约为 550 行，由此可以看出 HLS 开发模式降低了工作量和后续代码维护的难度，提高了开发效率。两种开发模式的算法代码结构选择是相同的，均为状

态机结构，在具体的实现上有细微差别，虽然 HLS 开发模式综合的 RTL 代码与手写的 RTL 代码均为状态机，但是 HLS 开发模式综合出来的代码可读性较低。从算法优化的难度来看，在 HLS 开发模式中，把算法从非流水线结构优化为流水线结构较为简易，例如对一个 for-loop 进行流水线化仅需一条 HLS 软件指令即可完成[17]，而传统 RTL 开发模式实现流水线化则需要对 RTL 代码进行大量改写，这需要花费更长的设计时间。如果传统 RTL 开发模式需要达到 HLS 开发模式相近的实验结果，则需要对代码结构进行重新评估，甚至需要推翻现有的代码结构进行重构来达到相似的效果。但是在对照实验中，两种开发模式相应的数据结构和最终的算法行为都是保持一致的，实验结果的对比都是基于以上代码结构差异得出。

在对照实验中，两种开发模式的实验结果都是在 16 种不同的地图样式中选定相同的地图尺寸、起点终点求出的平均值。图 16、图 17、图 18、图 19 分别展示了 16 个地图中的 4 个具有代表性的地图样式。

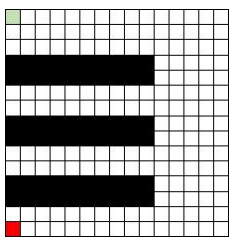


图 16.地图样式一

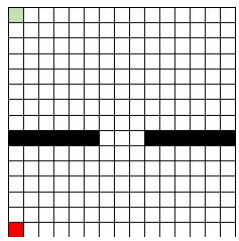


图 17.地图样式二

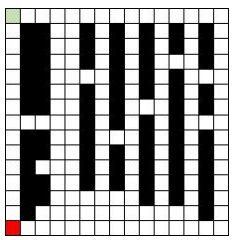


图 18.地图样式三

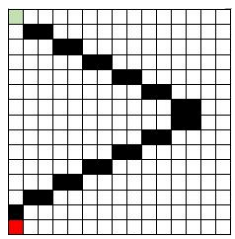


图 19.地图样式四

此外，本实验还设置了两个隔绝起点终点的地图，算法无法从这两个地图样式中获取搜索路径，从另一个角度验证了算法的正确性。图 20、图 21 分别是测试的地图样式。

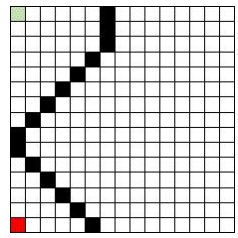


图 20.地图样式五

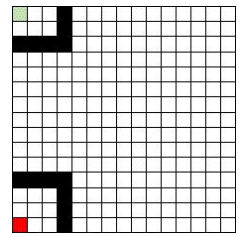


图 21.地图样式六

表 3、表 4 和表 5 分别列举了 HLS 开发模式和传统 RTL 开发模式的优先队列、堆栈和队列三种不同数据结构实现的 A*算法下硬件和延时表现。

表 3 同为优先队列数据结构下 HLS 和传统 FPGA 开发模式比较

FPGA 开发模式	运行时间(ns)	LUT 个数	FF 个数	BRAM 个数
HLS	483,445	3208	3303	34.5
RTL	523,046	7253	7161	59.5

表 4 同为堆栈数据结构下 HLS 和传统 FPGA 开始模式比较

FPGA 开发模式	运行时间(ns)	LUT 个数	FF 个数	BRAM 个数
HLS	3,305,830	1669	1523	25
RTL	5,987,348	7313	7065	56

表 5 同为队列数据结构下 HLS 和传统 FPGA 开发模式比较

FPGA 开发模式	运行时间(ns)	LUT 个数	FF 个数	BRAM 个数
HLS	3,389,425	1756	1555	25
RTL	5,598,080	7317	7053	56

由以上表格可知，在相同的开发周期内，HLS 开发模式的实验结果都是要优于传统 RTL 开发模式的。从运行时间上看，优先级队列数据结构的 HLS 开发模式的运行时间比传统 RTL 开发模式要短 10%左右，而对于另外两种数据结构，HLS 开发模式对运行时间的优化更是可以达到 40%左右，这意味着 HLS 开发模式实现的算法可以更快找出最优路径。由此可以体现出 HLS 工具有优秀的硬件资源分配，以及 HLS 工具所生成的 RTL 代码风格更加有利于算法的运算。从硬件资源看，随着地图规模的增大，两种开发模式对硬件的资源占用都是越来越多的，但是两种开发模式的硬件占用增长幅度是不同的。经过计算，从 50*50 地图扩展到 180*180 的地图，

HLS 开发模式的硬件资源占用增长率为 52%，而传统 RTL 开发模式的硬件资源占用增长率为 69%。由此可以看出，在相同的开发周期中 HLS 开发模式在硬件资源管理上是要比传统 RTL 开发模式做的更好。

综上所述，在相同的开发周期内，HLS 开发模式所得到的 RTL 设计可以用更少的硬件资源来更快的实现 A*算法完成路径搜索任务。因此可以借助 HLS 工具进行硬件算法辅助开发，从而可以缩短整体硬件项目的开发周期，也可以移植更多的软件算法在硬件上进行实现。

5 结 语

本文针对机器人导航中的实时性需求，提出采用高层综合对路径规划中经典的 A*算法实现快速高效的硬件定制。文中详细分析了算法中优先队列、堆栈和队列三种不同的数据结构，以及两种不同的开发模式之间的性能比较；实验数据分析表明，高层综合可以有效缩短算法硬件的开发周期，并获得不亚于传统 RTL 开发模式的硬件效能。

参 考 文 献

[1] 郑岩. 改进势场蚁群算法的机器人自主导航应用研究[D]. 重庆三峡学院, 2020.

[2] 郭炜, 魏继增, 郭筝, 谢憬. SoC 设计方法与实现（第 3 版）. 电子工业出版社, 2017 年 8 月.

[3] Johansson A, Johansson J. High-Level Synthesis for Efficient Design and Verification[J]. 2020.

[4] 吴家飞. 基于 HLS 的 MobileNet 加速方法研究[D]. 福建师范大学, 2019.

[5] 张壮. 基于 Vivado HLS 图像预处理 IP 核设计[J]. 电子世界, 2018, 18.

[6] （美)Thomas H.Cormen; Charles E.Leiserson; Ronald L.Rivest; Clifford Stein，《算法导论(原书第 3 版)》，机械工业出版社，2013

[7] https://github.com/abdokaseb/AStar-C/

[8] Mentor, a Siemens Business,“HLS Bluebook”，Software Version v10.5b，June 2020

[9] W. Altoyan and J. J. Alonso, "Investigating Performance Losses in High-Level Synthesis for Stencil Computations," presented at the 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2020.

[10] T. Ohkawa, Y. Sugata, H. Watanabe, N. Ogura, K. Ootsu and T. Yokota, "High Level Synthesis of ROS Protocol Interpretation and Communication Circuit for FPGA," 2019 IEEE/ACM 2nd International Workshop on Robotics Software Engineering (RoSE), Montreal, QC, Canada, 2019.

[11] M. Tatsuoka and M. Kaneko, "Wire congestion aware high level synthesis flow with source code compiler," 2018 International Conference on IC Design & Technology (ICICDT), Otranto, Italy, 2018.

[12] G. Martin and G. Smith, "High-Level Synthesis: Past, Present, and Future," in IEEE Design & Test of Computers, vol. 26, no. 4, pp. 18-25, July-Aug. 2009.

[13] 潘妍,程岳,高雅濛.面向 FPGA 的高层次综合技术综述[J].信息技术与信息化,2022(03):96-99.

[14] 石添介,刘飞阳,田径等.基于高层次综合的 FPGA 循环神经网络加速器设计[J].信息技术与信息化,2022(01):151-153.

[15] 叶海雄,陶宁蓉,匡兴红等.基于 Catapult C 高层次综合工具平台优化运动检测算法的研究[J].电子设计工程,2017,25(14):1-4.DOI:10.14022/j.cnki.dzsjgc.2017.14.001.

[16] 徐瑞帆,肖有为,罗进等.高层次综合综述[J].微纳电子与智制造 ,2021,3(02):74-79.DOI:10.19816/j.cnki.10-1594/tn.2021.02.074.

[17] P. R. Panda, N. Sharma, S. Kurra, K. A. Bhartia and N. K. Singh, "Exploration of Loop Unroll Factors in High Level Synthesis," 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, India, 2018.

[18] 谢晓燕,张玉婷,刘镇弢.高层次综合特征检测算法的 FPGA 实现[J].实验室研究与探索,2018,37(01):93-97+117.

[19] 游余新.利用 Mentor 高层次综合技术(Catapult Synthesis)快速实现复杂 DSP 算法[J].中国集成电路,2007(05):35-41.

[20] 周成瑞,杨玲玲.基于 A 星算法的全向移动机器人仿真研究[J].电脑与信息技术,2023,31(03):29-31.

一种模型驱动的装备嵌入式软件自动化测试方法

尚小虎[1]，仵林博[1]，邓霏[1]，聂长海[2]，马思友[1]
（1.中国工程物理研究院 计算机应用研究所，绵阳市，621900）
（2.南京大学 计算机科学与技术系，南京市，210023）

摘要：为了提高装备嵌入式软件测试过程中的覆盖率和效率，减少人为错误，本文提出了一种模型驱动的装备嵌入式软件自动化测试方法。通过行为模型和数据模型的建立，采用算法遍历路径得到测试用例集，保障了测试的充分性和覆盖率；通过关键字、自动执行框架的设计，实现了测试用例集的自动执行及测试过程的实时显示和记录，提高了测试效率；此外，执行脚本生成模块保证了嵌入式软件测试的实时性。将本文设计的平台应用于某监控软件测试中，与人工测试的实验结果表明，该平台能够实现装备嵌入式软件的自动化测试，有效保障了测试的覆盖率和效率。

关键词：装备嵌入式软件；模型驱动；测试用例生成；自动化测试

A Model Driven Automated Testing Method for Equipment Embedded Software

Xiaohu Shang[1], Linbo Wu[1], Deng Fei[1], Nie Changhai[2], Ma Siyou[1]
(1. Institute of Computer Application, China Academy of Engineering Physics, Mianyang 621999, China;
2. Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

【Abstract】In order to improve the test coverage and efficiency and reduce human error in the process of equipment embedded software testing, this paper proposes a model-driven equipment embedded software automatic testing method. Through the establishment of the behavior model and the data model, the test case set is obtained by using the algorithm traversal path, which ensures the adequacy and coverage of the test. Through the design of keywords and automatic execution frameworks, the automatic execution of test case sets and real-time display and recording of test processes are realized, which improves test efficiency. In addition, the Execute Script Generation Module guarantees real-time testing of embedded software. In a monitoring software test experiment, comparing the platform designed in this paper and manual test, the experimental results show that the platform can realize the automatic test of equipped embedded software, which effectively ensures the coverage and efficiency of the test.

【Key words】Equipment embedded software; Model driven; Test case generation; Automated testing

伴随着武器装备自动化程度的大幅度提高，现代武器装备软件的复杂程度越来越高，如何保障日益复杂的装备软件的质量与可靠性也变得极其重要[1]。基于仿真的软件测试是提升装备软件质量与可靠性的重要手段[2-6]。尽管装备软件的测试工作在“尽早测试”、“全面测试”、“全过程测试”和“独立、迭代的测试”等测试理念的影响

基金项目：国家重点研发计划（2021YFB3101805）;国家自然科学基金青年基金项目（61903348）;四川省科技计划重点研发项目（2021YFG0174）

作者简介：尚小虎(1992－)，男，硕士，工程师，主要研究方向软件建模与测试。
E-mail：jlusxh@qq.com

下不断前延，但装备软件仍面临内部测试、三方测试、鉴定测试等多轮迭代测试，加之在回归测试中存在测试用例有效选取的困难，测试机构往往完全复用全部测试用例，给装备软件测试带来了巨大的工作量[7]。因此，研究装备嵌入式软件自动化测试方法，可以有效减少测试的时间成本和手动测试的错误，同时能够高质量地完成测试任务，提高装备嵌入式软件的质量与可靠性。

目前，装备嵌入式软件的测试主要以手工测试为主，现有仿真系统也较少考虑测试用例设计、测试用例自动化执行等方面的功能，很难满足测试策划、测试设计、测试执行、结果确认等不同测试阶段的需要，且很少考虑回归测试阶段测试用例的选取与复用。上述原因可能导致装备软件的测试不够充分、测试效率低、软件延迟交付等一系列问题。

本文研究了一种模型驱动的装备嵌入式软件自动化测试方法。最终构建一套基于模型的测试用例自动生成并自动执行的测试平台，实现装备嵌入式软件的自动化测试。主要包括业务建模、关键字设计与实现、测试用例自动生成、测试用例解析、数据执行脚本自动生成、嵌入式软件测试实时执行引擎等关键技术。以 UML 状态机作为被测装备嵌入式软件的测试模型，集成关键字库设计模块、测试用例自动生成模块、数据执行脚本自动生成模块、自动测试执行引擎模块及测试结果统计模块等。测试用例生成主要采用算法对模型的状态路径进行遍历，藉此来生成测试用例，一个完整测试用例由一个数据用例和一个行为用例组合而成。最后通过具体实验介绍了平台主要功能。

本文的主要贡献如下：

1）研究了一种模型驱动的装备嵌入式软件自动化测试方法，通过对被测装备软件业务建模、关键字设计、测试用例与数据驱动脚本自动生成，实现了测试的自动化；

2）基于行为模型和数据模型的设计，能够实现业务逻辑和测试数据的分离，便于测试模型的复用；

3）模型生成的测试用例自动转化成装备嵌入式软件测试的执行脚本，能够实现较强实时性的测试；

4）设计实现模型驱动的测试平台，并应用到具体测试项目中，验证了该平台的可行性。

1 相关工作

国内外许多软件开发者、学者、测评机构针对装备嵌入式软件的模型驱动测试与自动化测试进行了许多有益的探索实践。

1.1 模型驱动测试

模型驱动测试技术是一种依据模型规定的预期行为来检查被测对象运行结果正确性的软件测试方法[8]。RobertV.Binder 的调查表明[9]：一般情况下，模型驱动的测试能够降低 29%的 bug 遗漏、35%的测试成本、30%的测试周期。

模型驱动测试是基于被测对象进行需求分析，并结合测试策略构建一个能够清晰、全面描述被测对象的测试模型，最终基于该模型实现测试用例的自动生成，从而提高软件测试的覆盖强度和自动化水平[10]。因此，模型驱动的测试主要包含两方面的技术，一是测试模型的构建，二是测试用例的生成算法。

目前常用的建模语言有 UML、Petri 网、马尔科夫链、AADL 等[11-14]。其中，UML 具有良好的面向对象性以及丰富的建模表达能力，能够对系统的动态行为进行较为准确的描述。考虑到 UML 在实时性方面的表达能力不够，我们设计了嵌入式软件测试实时执行引擎，能够将模型生成的测试用例转化成半实物仿真平台执行的测试执行脚本，满足装备嵌入式软件测试的实时性要求。

模型驱动测试中测试用例的生成是基于测试模型实现的，测试模型大都有状态和路径可循，如 FSM、有向图等，测试用例生成主要是采用算法对模型的状态、路径进行遍历来实现。Samuel[15]针对 UML 时序图中的很多关键特征不能被覆盖的问题，通过考虑循环、分支和终止等 UML 时序图特征，提出了一种基于遍历算法的自测试序列自动生成方法。黄晨[16]为了消除 UML 状态图多并发性和多层次的影响，将 UML 状态图转换为扩展有限状态机（EFSM），并采用

广度优先搜索覆盖路径、状态满足、迁移覆盖的准则，通过测试序列建立查找表的方式生成测试用例集合，一条路径对应一个行为序列，一个行为序列对应一条测试用例。基于状态图生成测试用例的本质是遍历有限状态机，也就等同于遍历状态机生成的有向图，因此，测试用例的生成算法也可采用有向图的遍历算法。

1.2 装备软件自动化测试

软件自动化测试的方法主要有模块化测试、测试库测试、数据驱动测试和关键字驱动测试[17]。其中关键字驱动测试将相关测试逻辑抽象为简单易懂的关键字，实现了测试逻辑、测试数据、测试脚本的三者的分离[18]。最终，每个测试用例均通过关键字序列组合而成。

国内外学者对装备软件的自动化测试进行了大量研究。其中，Aamir Mehmood Mirza 等基于关键字的思想，提出了一种端到端的功能测试自动化测试框架 ContextDrive，实现了基于场景的测试技术来测试环境感知应用[19]。Maxwell J. Cobar 等基于仿真环境设计了一种开源可配置软件，能够模拟飞行器飞行时的真实环境[20]。B.Srinivas Reddy 等针对军用航空电子设备，提出了一种测试任务计算机关键软件的自动化测试框架[21]。此外，Yang Shuaishuai 等通过对嵌入式软件自动化测试框架的研究，设计了一种测试脚本生成的平台[22]。王子豪在测试用例自动生成的基础上，将测试用例转化为了可供测试执行的测试脚本[23]。然而，上述研究大都面向某一专业领域，技术门槛较高，且测试自动执行的难题均未涉及。

研究者们针对模型驱动的测试用例自动生成和软件自动化测试都做了很大探索和尝试，但较少涉及两者的结合及平台的实现，本文提出的一种模型驱动的装备嵌入式软件自动化测试方法，将基于模型的测试用例自动生成与测试用例自动执行集成到同一个测试平台中，实现了装备嵌入式软件的自动化测试，在保障装备软件测试覆盖、减少人为错误的同时也降低了测试成本、提高了测试效率。

2 测试模型构建和测试用例生成

模型驱动的装备嵌入式软件自动化测试的首要任务是建立装备嵌入式软件的抽象测试模型，用于对测试需求，如状态迁移、测试策略、系统结构等进行描述。本文采用 UML 状态图对装备嵌入式软件的行为和数据进行建模，主要针对装备嵌入式软件的业务流程。采用算法对行为模型和数据模型进行路径遍历得到行为用例和数据用例，再将两者进行组合就可以得到测试所需的测试用例集合。

如图 1 所示为行为模型示例，规定每个模型图只能有一个起点 I 和终点 F；S 是能够稳定存在的软件状态抽象，如自检成功状态；T 是状态之间的迁移过程的抽象，由事件触发；每条路径均是从起点迁移到终点。

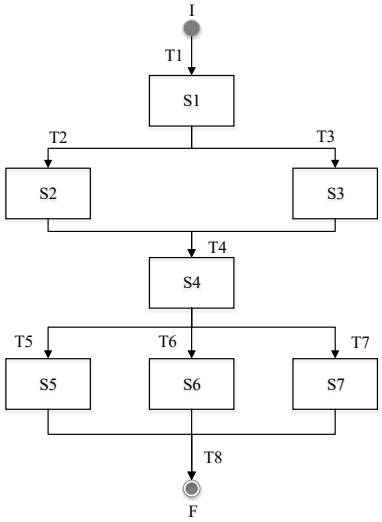


图 1 行为模型示例

采用算法对上述模型图进行路径遍历可以得到表 1 的行为用例表。

表 1 示例遍历的行为用例

标识	行为用例
TC1	I,T1,S1,T2,S2,T4,S4,T5,S5,T8,F
TC2	I,T1,S1,T2,S2,T4,S4,T6,S6,T8,F
TC3	I,T1,S1,T2,S2,T4,S4,T6,S7,T8,F
TC4	I,T1,S1,T3,S3,T4,S4,T5,S5,T8,F
TC5	I,T1,S1,T2,S2,T4,S4,T6,S6,T8,F
TC6	I,T1,S1,T2,S2,T4,S4,T7,S7,T8,F

如图 2 所示为数据模型示例，与行为模型不同的是，D1、D2 为行为模型需要用到的数据变量名，M1、M2、N1、N2、N3 为变量的具体取值。采用算法对数据模型进行

路径遍历可以得到表 2 的数据用例。

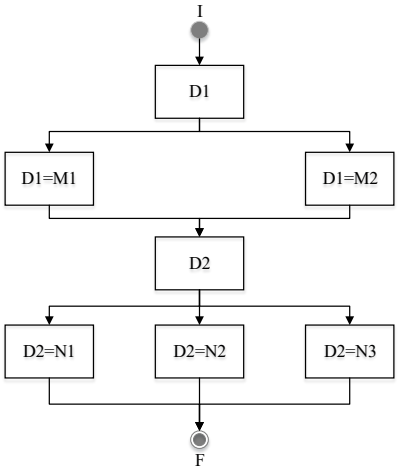


图 2 数据模型示例

表 2 示例遍历的数据用例

标识	数据用例
data1	D1=M1,D2=N1
data2	D1=M1,D2=N2
data3	D1=M1,D2=N3
data4	D1=M2,D2=N1
data5	D1=M2,D2=N2
data6	D1=M2,D2=N3

假设每条行为用例均用到 D1 和 D2 这两个数据，且每组数据的取值在任意一条行为用例均能执行，那么由行为用例和数据组成的最终测试集合的数量就是 6*6=36 条。

上述测试用例生成方法已集成在模型驱动的装备嵌入式软件自动化测试平台中。

3 模型驱动的装备嵌入式软件自动化测试平台实现

3.1 平台框架

本文研究模型驱动的装备嵌入式软件自动化测试方法的实现，最终研发一套基于模型自动生成测试用例并能自动执行测试用例的平台。如图 3 所示为平台实现的功能架构，主要包含测试建模及用例生成、自动执行两部分，其中测试建模及用例生成集成了行为模型、数据模型、关键字库设计、测试用例集合建立等模块，自动执行包括测试用例解析模块、执行脚本生成模块、测试执行引擎、总线接口底层驱动等模块。

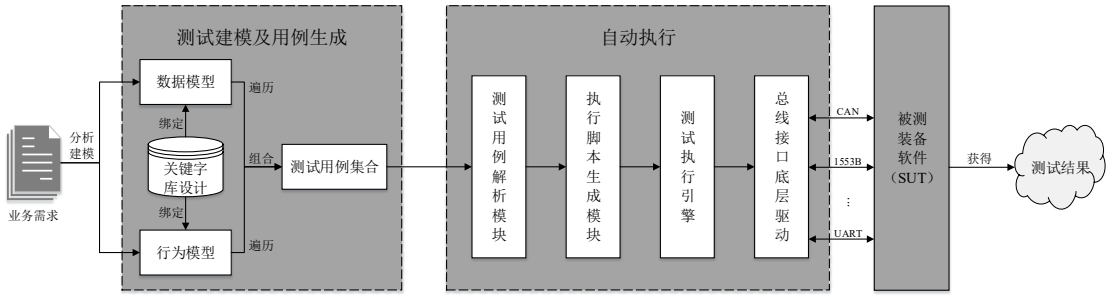


图 3 平台功能架构示意图

3.2 测试建模及用例生成

对于给定被测软件，测试人员对业务需求进行分析，建立行为模型和数据模型。行为模型的建立能够直接表达出期望的被测对象的动作及状态；能够与测试脚本绑定，使得动作、状态与行为能够对应；可以自动生成可被自动化测试执行系统执行的测试用例，替代了人工的测试用例设计和编写，提高工作效率及用例复用率。数据模型主要用来构建测试用例所需要的测试数据，驱动测试的执行。

为了使模型能够被执行，我们采用在模型上绑定关键字的方式与自动执行系统建立连接，每个状态和迁移均能绑定一个或多个关键字。在状态和迁移的拓扑图中遍历出的路径就是生成的测试用例，每条用例也就是一组关键字执行序列。

在平台框架中，根据业务需要将关键字设计分为三层：

（1）常用业务层关键字：面向装备软件用户以及测试执行人员，对应装备软件工作流程的核心步骤，位于关键字最上层，如自检功能；

（2）常用逻辑层关键字：面向脚本开发人员，对应核心步骤的主要功能模块，位于常用关键字下层，如 CAN 接口自检、UART

接口自检；

（3）执行层关键字：面向测试开发人员，对应总线接口底层驱动，位于关键字最下层，如 CAN 总线的发送与接收。

3.3 自动执行

测试用例解析模块对从模型得到的测试用例集合进行解析，并逐条获取测试用例传递给执行脚本生成模块，转化成可供仿真平台执行的测试执行脚本，然后通过测试执行引擎驱动测试的自动执行，其流程如图 4 所示。

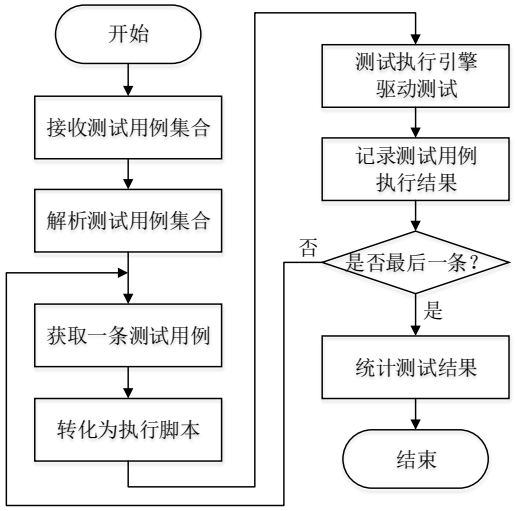


图 4 自动执行流程

执行脚本是仿真系统进行测试的输入，本文基于测试用例（数据用例和行为用例的关键字序列）转化得到。执行脚本的格式包括两种，一是主动发送脚本，脚本结构如图 5 所示，一是被动响应脚本，脚本结构如图 6 所示。

下面对两种脚本进行详细说明。

主动发送脚本：图 5 示例配置了四组数据。第一组总线接口类型为 NET，发送通道为 0，起始发送时间为 1000ms，发送次数为 10 次，发送间隔时间为 10ms，发送数据为 11 22 33 44 55 66 77；第二组、第三组、第四组数据各字段含义与第一组相同。四组数据按照配置的参数含义并行发送。

```
//接口类型 发送通道 起始发送时间...
//发送次数 发送间隔时间 发送数据
NET 0 1000 10 10 11223344556677
CAN 0 10 20 5 55AA55AA55AA55AA
UART 1 50 200 2 AABBCCDDEEFF
UART 0 50 200 2 010203040506
```

图 5 主动发送脚本示例

被动响应脚本：图 6 示例配置了一组触发数据，两组响应数据。CAN 总线 1 通道收到指令“AA BB CC DD”后，CAN 总线 1 通道延时 50ms，返回应答数据“01 02 03 04”1 次，CAN 总线 2 通道延时 100ms，返回应答数据“01 02 03 04”2 次，周期时间为 10ms。

```
//触发数据
[AA BB CC DD]
TriggerType=CAN //触发总线类型
TriggerChannel=1 //触发总线通道
Event_ID=自检命令

Response_0_Type=CAN //响应总线类型
Response_0_Channel=1 //响应总线通道
Response_0_Data=00 01 02 03 //响应数据
Response_0_ID=自检回令-1
Response_0_Delay=50
Response_0_RepeatNum=1
Response_0_RepeatPeriod=0

Response_1_Type=CAN //响应总线类型
Response_1_Channel=2 //响应总线通道
Response_1_Data=10 11 12 13 //响应数据
Response_1_ID=自检回令-2
Response_1_Delay=100
Response_1_RepeatNum=2
Response_1_RepeatPeriod=10

Event_ResponseNum=2
```

图 6 被动响应脚本示例

脚本中还可配置发送数据校验、发送帧计数、回令数据从接收指令任意选取字节、屏蔽接收指令中不关注的字节等规则。

单条测试用例执行完成后，平台会自动生成一个对应的日志文件，记录该条测试用例的执行情况，包括每个关键字的执行通过与否。同时，还会将该条用例对应的总线上的交互数据记录下来，便于事后分析。

4 工程应用

本文提出的模型驱动的装备嵌入式自动化测试方法已应用到许多装备软件测试中，在保证装备软件测试充分性的同时，也提高了测试的效率，对装备软件的高质量、按时交付起到了重要作用。下面，本文以某监控软件为测试对象进行实验，验证本文方法的可行性。

下面对模型驱动的装备嵌入式软件自动化测试平台的运行进行简单描述。被测对象为某监控软件，图 7 是应用实例的行为状态状模型，图 8 为其数据模型。

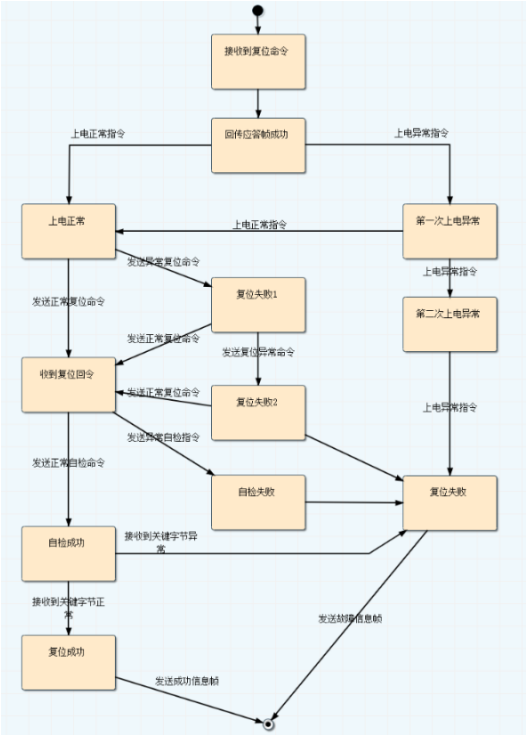


图 7 某监控软件行为模型

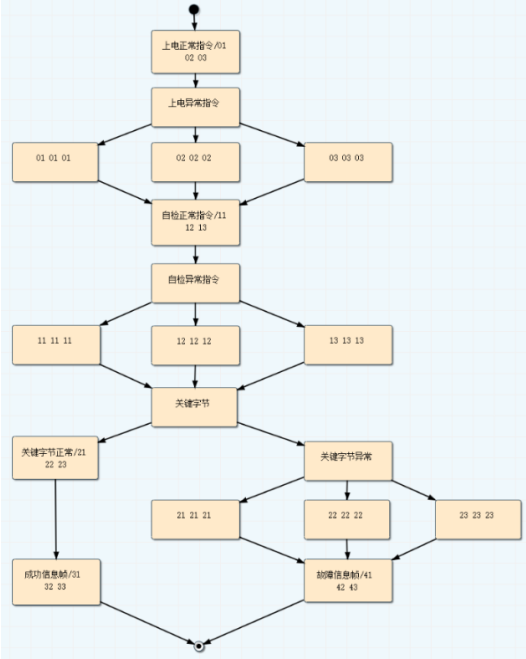


图 8 某监控软件数据模型

根据监控软件的业务流程建立行为模型后，基于模型设计实现用于驱动模型执行所需的关键字，共设计关键字 51 个，如图 9 为发送正常复位关键字的代码示例，并将关键字与模型进行绑定，如图 10 为绑定过程，每个状态或迁移均可绑定一个或多个关键字。模型生成的测试用例即为模型对应路径的关键字序列，自动执行平台将根据关键字脚本将关键字序列转化成可供仿真系统执行的执行脚本。

```
except Exception as e:
    logger.error("resetfail Fail! \nError Message:\n" + str(e))
    return False

def sendnormalreset(self,normalresetcommand)
    try:
        normalresetcommand = self.findValueInFile(normalresetcommand)
        #写主动发送事件
        script.sendtxt("CAN", "0", "10", "1", "0", str(normalresetcommand))
        #(总线类型、通道号、起始时间，发送次数，间隔时间，发送数据)
        if script.readfile("snd", "CAN", "0", str(normalresetcommand))
            logger.info("sendnormalreset success!")
            return True
        logger.error("sendnormalreset Fail!")
        return False
    except Exception as e:
        logger.error("sendnormalreset Fail! \nError Message:\n" + str(e))
        return False
```

图 9 部分关键字设计

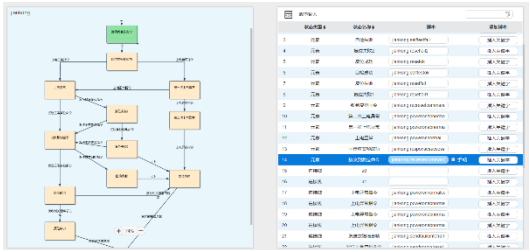


图 10 关键字与模型绑定

采用算法遍历行为模型共得到行为用例 15 条，其中复位成功用例 5 条，复位失败用例 10 条；遍历数据模型共得到数据用例 36 条，其中复位成功的数据用例 9 条，复位失败的数据用例 27 条。将数据用例和行为用例进行组合得到的测试集合包含 9*5+27*10=315 条测试用例。

测试集合建立后，启动自动化测试框架，测试集合会按照顺序将测试用例发送给自动执行平台，并解析转化为执行脚本供仿真系统执行，并在界面上实时显示执行进度与结果，生成测试报告，如图 11 所示。

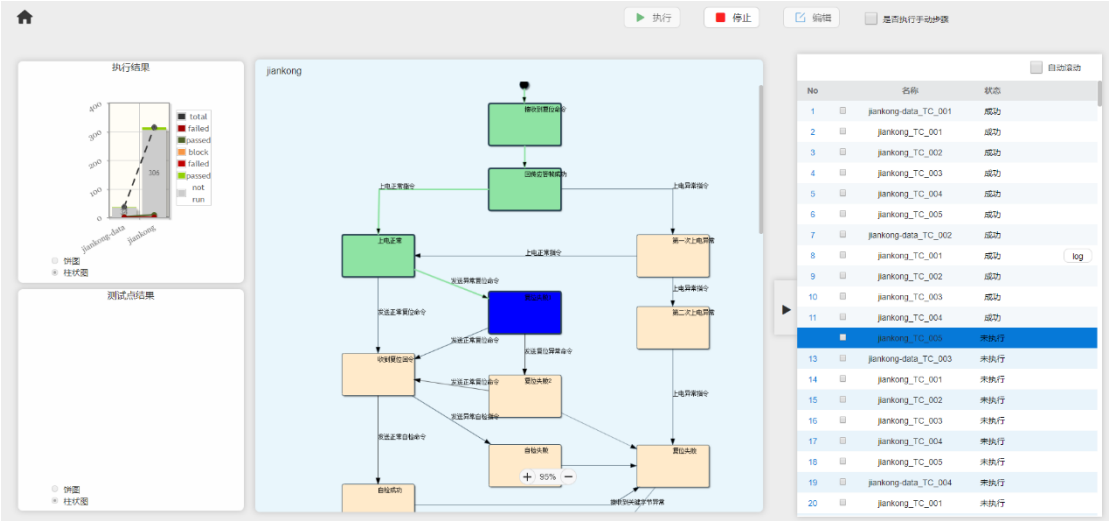


图 11 测试用例自动执行与实时显示

针对上述示例，从测试设计和测试执行两个方面与人工测试进行分析比较。

按照人工测试的流程对上述示例进行测试设计，由于人工大都只考虑单一环节的分析，较少考虑相邻环节乃至整个流程的测试组合，因此人工分析设计得到测试用例 33 个，约为模型生成用例的十分之一。从测试设计方面可以看出，基于模型的测试有效提高了测试覆盖的场景，能更好的保证软件的质量。

在测试执行方面，模型共生成用例 315 条，按照每人每小时执行用例 20 个进行统计，共需 15.75 小时执行完全部用例，而自动执行平台关键字脚本编写需用时 12 小时，执行仅需 3 小时。示例软件完成代码升级后，测试人员复用全部测试过程，从表 3 测试结果看，自动化提高了测试效率，特别是对多轮次的迭代测试，其提升的效率更加明显。

表 3 人工测试和自动化测试结果对比

分类	人工测试	自动化测试
首轮测试	15.75h	15h
回归测试	15.75h	3h
总计	31.5h	18h

结论

装备软件作为一种关键软件，对充分性、安全性和可靠性的要求也较高，相应地测试难度较大，通过手工测试较易产生人为的错误。此外，人工设计的测试用例充分性难以得到保障，人工测试的效率也较低。因此，对模型驱动的装备软件自动化测试方法和工具的需求越来越迫切。基于这一问题，本文设计并实现了一种模型驱动的装备嵌入式软件自动化测试平台，使用 UML 对被测软件业务流程和数据进行建模，采用算法分别对行为模型和数据模型进行遍历得到行为用例和数据用例，并将两者进行组合得到测试所需的集合；通过关键字的设计实现并与模型绑定，实现了模型的驱动执行；基于自动执行框架实现了测试用例到执行脚本的自动转化与执行、测试过程实时显示及测试报告自动生成。与人工测试的实验分析表明本方法可以有效提升测试的充分性和测试的效率，保证了软件的质量。下一步的工作方向将考虑将数据嵌入到行为模型中，以实现行为用例生成时，该路径对应所需的数据已完成自动组合，减少人工组合可能产生的错误。

参考文献

[1] Zhang J , Yang H , Sun Z ,et al.Automated Testing of Multi-UDP Interface Equipment Software based on Scenario and Data Driven[C]//2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC).IEEE, 2021..

[2] Zhan, Guang; Gong, Zheng; Lv Quanhui; et al. Flight Test of Autonomous

Formation Management for Multiple Fixed-Wing UAVs Based on Missile Parallel Method[J]. Drones(S2504-446x),2022,6(5):99.

[3] Wenxing Li, Yu Fang M.D., Xiaojun He, et al. Analysis and design of testing supporting system of flight track processing software[C].Society of Photo-Optical Instrumentation Engineers Proceedings(SPIE Proceedings)：2022.

[4] 吴枫, 刘秀罗, 王佳,等. 航天发射可视化仿真分析技术与应用研究[J]. 系统仿真学报, 2022, 34(4):856-869.

[5] 冯洋,夏志龙,郭安等.自动驾驶软件测试技术研究综述[J].中国图象图形学报,2021,26(01):13-27.

[6] 孙剑,黄润涵,李霖等.智能汽车环境感知与规划决策一体化仿真测试平台[J].系统仿真学报,2020,32(02):236-246.

[7] 张玉晓, 孙伟, 张军. 基于数据驱动的装备嵌入式软件自动化测试的研究与应用[J]. 测试技术学报, 2015(3) :219-223.

[8] Pretschner A , Prenninger W , Wagner S , et al. One evaluation of model-based testing and its automation[J]. DBLP, 2017.

[9] Binder RV, Kramer A, Legeard B. 2014 Model-based testing user survey: Results. Robert V. Binder's Technology and Businiss Blog. 2014.

[10] Wang L Z, Xuan-Dong L I. Research on Model-driven Software Testing[J]. Computer Science , 2005 , 32(10):230-235.

[11] Nianhua Yang, Huiqun Yu , Hua Sun, et al. Modeling UML sequence diagrams using extended Petri nets[J].Telecommunication Systems,2012，51（2）：147-158.

[12] Guo H , Yang X . Automatic creation of Markov models for reliability assessment of safety instrumented systems[J]. Reliability Engineering & System Safety, 2008, 93(6):829-837.

[13] 冯冰,杨志义,董云卫等.一种面向 AADL 架构的模型测试方法[J].计算机测量与控制,2010,18(04):778-781.

[14] 王瀚博,周兴社,董云卫等.结构分析和设计语言 AADL 研究[J].计算机工程与应用,2009,45(16):1-4.

[15] Samuel P , Joseph A T . Test Sequence Generation from UML Sequence Diagrams[C]// International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. 2008.

[16] 黄晨,段永颢.基于 UML 模型的嵌入式软件测试用例生成方法[J].空间控制技术与应用,2013,39(01):57-62.

[17] 程海南. 视讯系统的软件自动化测试应用与研究[D]. 南京：东南大学，2016.

[18] 章鸽鸽. 基于关键字驱动的自动化测试工具的设计与实现[D].合肥，安徽大学，2018.

[19] Mirza A M , Khan M , Wagan R A , et al. ContextDrive: Towards a functional scenario-based testing framework for context-aware applications[J]. IEEE Access, 2021, PP(99):1-1.

[20] Maxwell J, Cobar, Carlos J, Montalvo. The Facility for Aerospace Systems and Technology Simulation: FASTSim - An Open-Source Configurable Software in the Loop Simulation Environment[C]//. AIAA AVIATION 2022 Forum. Chicago, USA: American Institute of Aeronautics and Astronautics, 2022.

[21] Reddy B S , Padmavathi, Maruthamma P , et al. Automation of Testing Military Avionics Mission Critical Software[C]// 2018 Second International conference on Electronics, Communication and Aerospace Technology (ICECA). 2018.

[22] Yang S , Yu Z , Liu B , et al. An automatic testing framework for embedded software[C]// 2017 12th International Conference on Computer

Science and Education (ICCSE). IEEE, 2017.
[23] 王子豪. 基于需求模型的航天软件测试自动化方法研究[D].中国运载火箭技术研究院,2019.

基于混沌映射的抗机器学习攻击强 PUF 设计

方皓冉[①]　　李刚[①②]　　汪鹏君[*①]

[①](温州大学电气与电子工程学院 温州 325035)

[②](南京邮电大学集成电路科学与工程学院 南京 210023)

摘　要：物理不可克隆函数(Physical Unclonable Function，PUF)在信息安全领域具有极其重要的应用前景，然而也存在易受到机器学习攻击等方面的不足。该文通过对 PUF 电路、响应反馈和混沌算法的研究，提出一种基于混沌映射的抗机器学习攻击 PUF 电路设计方案，该方案将激励作为混沌映射初值，利用 PUF 运行时间与混沌映射深度的内在关系，产生不可预测的混沌映射，降低激励与响应之间的线性关系；对该 PUF 在 FPGA 中提取的 100 万组激励响应对(Challenge Response Pairs, CRPs)进行安全性测试和分析，结果表明，逻辑回归、人工神经网络和支持向量机的攻击预测率接近 50%的理想值，且具有良好的随机性、唯一性和可靠性。

关键词：物理不可克隆函数；机器学习；混沌算法；响应反馈

Design of Strong PUF Circuit Against Machine Learning Attacks Based on Chaos Mapping

FANG Haoran[①]　　LI Gang[①②]　　WANG Pengjun[*①]

[①](*College of Electrical and Electronic Engineering, Wenzhou University, Wenzhou 325035, China*)

[②](*College of Integrated Circuit Science and Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*)

Abstract: Physical Unclonable Function (PUF) has extremely important application prospects in the field of information security, but there are also shortcomings such as susceptibility to machine learning attacks. This article proposes a design scheme for anti machine learning attack PUF circuit based on chaotic mapping by studying PUF circuits, response feedback, and chaotic algorithms. The scheme takes excitation as the initial value of chaotic mapping, and utilizes the inherent relationship between PUF runtime and chaotic mapping depth to generate unpredictable chaotic mapping, reducing the linear relationship between excitation and response; The security test and analysis of 1 million groups of Challenge Response Pairs (CRPs) extracted from the PUF in FPGA shows that the attack prediction rate of Logistic regression, artificial neural network and support vector machine is close to the ideal value of 50% and has good randomness, uniqueness and reliability.

Key words: Physical Unclonable function; Machine learning; Chaos algorithm; Response feedback

1 引言

随着物联网的发展，网络安全问题日益受到关注。安全芯片正被广泛应用于日常生活的方方面面，涉及军事、金融、医疗、物联网、移动通信和航空航天等领域。这些应用的基本安全需求是认证和证明，即确定底层嵌入式系统的唯一标识和数据的完整性[1]。物理不可克隆函数(Physical Unclonable Function，PUF)作为一种特殊的“芯片指纹”提取技术，通过捕获集成电路(IC)制造过程中无法避免引入的工艺偏差，产生随机性、唯一性以及物理不可克隆的特征密钥，可广泛应用于知识产权保护、设备认证、物联网防护、密钥安全

[*]通信作者：汪鹏君　wangpengjun@wzu.edu.cn

基金项目：国家自然科学基金(62234008, 62174121)，温州市基础性科研项目(G20220005, G20210023)

Foundation Item : The National Natural Science Foundation of China (62234008, 62174121), The Wenzhou Basic Scientific Research Projects (G20220005, G20210023)

存储等信息安全领域[2]。

PUF 根据其激励响应对(CRP)数量不同分为弱 PUF 和强 PUF。弱 PUF 仅能产生有限数量 CRPs，且一个 PUF 单元通常只产生一位输出响应，因此输出响应间相互独立，用于密钥产生和存储。典型的弱 PUF 包括 SRAM-PUF，DRAM-PUF 和蝴蝶 PUF 等。强 PUF 主要包括仲裁器 PUF(Arbiter PUF，APUF)、环形振荡器 PUF(RO-PUF)和算术逻辑单元 PUF 等，用于设备认证和目标识别。强 PUF 通过硬件资源重构产生大量 CRPs，输出响应之间不可避免地存在相关性，因此容易受到逻辑回归(Logistic Regression，LR)、人工神经网络(Artificial Neural Network，ANN)和支持向量机(Support Vector Machine，SVM)等多种机器学习(Machine Learning，ML)算法[3]的攻击，攻击者可通过对强 PUF 的部分 CRPs 进行分析，一旦 PUF 电路的激励响应对被精确建模，便可根据输入激励C预测输出响应R。

为抵抗 ML 攻击，近年来提出了许多抗攻击技术。Zhang 等人[4]提出一种可配置的三态 PUF (CT PUF)，它可以灵活地作为仲裁器 PUF、环形振荡器 PUF(RO-PUF)或双稳态环 PUF(BR-PUF)，采用基于按位异或的机制来混淆激励和响应之间的关系，从而抵制 ML 攻击。Wu 等人[5]提出一种基于响应-反馈的轻量级抗 ML 攻击 PUF(FLAM-PUF)，由一个 APUF 和一个 Galois 线性反馈移位寄存器(LFSR)以及一些基本的逻辑门组成，有着很好的抗 ML 攻击能力。Nassar 等人[6]提出一种级联 PUF(CaPUF)，对典型机器学习攻击，如 LR 和 SVM 具有抵抗力。Zhang 等人[7]提出一种基于放大器链的 XOR PUF(AC-XOR PUF)，以两个相同的放大器链的和/或偏置相关的电压和放大信息作为熵源，可以有效地抵御 ML 建模攻击。Ahmed 等人[8]提出一种独特的混合 PUF 结构，该结构使用来自 APUF 的 CRPs，并将其馈送到环形振荡器中生成响应，从而使 PUF 不易受到机器学习建模攻击；Avvaru 等人[9]提出一种同构和异构的前馈异或 PUF(FFXOR PUF)。同构 PUF 结构中的异或门具有相同的物理特性，简化了设计和测试过程，而异构 PUF 结构中的异或门具有不同的物理特性，提高了安全性和可靠性但也增加了系统的复杂性；Shah 等人[10]提出一种基于递归的 PUF(Rec-PUF)，它将反馈和异或函数结合起来，在不显著降低可靠性的情况下，可显著提高 ML 攻击的抵抗能力。然而，非线性结构将在一定程度上降低输出响应的稳定性。此外，采用多层人工神经网络算法也能实现对非线性 PUF 电路的成功建模。鉴此，通过对 PUF 电路结构和抗攻击方法的研究，利用响应反馈技术，结合混沌算法，提出一种基于混沌映射的抗机器学习攻击 PUF(CM-PUF)。利用 FPGA 对该 PUF 进行实验验证，采用典型的 ML 攻击算法分析其安全性，并对随机性、唯一性及硬件资源开销等关键技术指标展开评估。

2 理论基础

2.1 APUF 电路结构与工作原理

在芯片生产过程中，由于环境影响会产生偏差，导致不同芯片上相同电路结构的物理特性不同。PUF 通过提取差异并将其表示为可识别信息来实现芯片身份认证的功能。强 PUF 通过激励重新配置硬件资源来产生差异化响应，并且随着激励位数量的增加，CRP 的数量呈指数级增加。APUF 作为一种典型的强 PUF，其结构简单、易于实现，已广泛应用于芯片认证、知识产权保护等多种安全场景[11]。APUF 电路结构如图 1 所示，由上下两条完全对称的延时路径和交叉耦合与非门组成的仲裁器构成。在激励信号$C = C_1, C_2, \cdots, C_n$的作用下，路径信号平行或交叉通过延时单元，仲裁器通过比较两条路径信号到达的先后顺序，判决输出响应 0 或 1。理论上，两个信号应同时到达仲裁器，但延时路径不可避免地存在工艺偏差，导致信号延时不同，因此产生的响应 0 或 1 具有不可预测性。通常用下式线性延时模型表示，总延时为每一级延时单元传播延时的累加和：

$$\Delta = \omega^{\mathrm{T}} \Phi \tag{1}$$

其中，ω表示 APUF 中每一个多路选择器传播延时的特征向量。n位激励C的函数表达式用Φ表示：

$$\Phi(C) = \left(\Phi_1(C), \Phi_2(C), \cdots, \Phi_n(C), 1\right)^{\mathrm{T}} \tag{2}$$

其中，$\Phi_j(C) = \prod_{i=j}^{n}(1 - C_i), j = 1,2,\cdots,k$。若$\Delta > 0$，APUF 的输出响应为 1，否则为 0。输出响应t表示为：

$$t = \mathrm{sgn}(\Delta) = \mathrm{sgn}\left(\omega^{\mathrm{T}} \Phi\right) \tag{3}$$

为简化计算，式(3)中的t可用$t=2r-1(t\in\{-1,1\})$代替，其中$r\in\{0,1\}$。

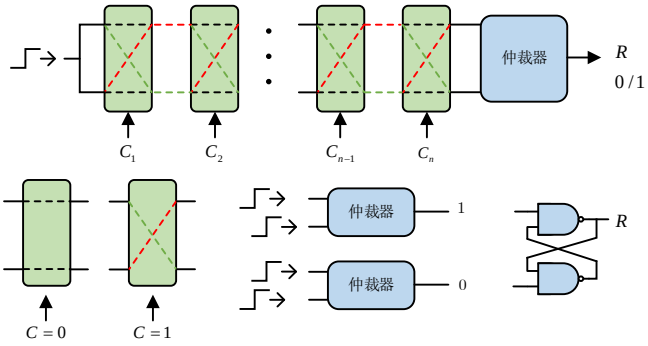


图 1 仲裁器 PUF 结构图

2.2 机器学习攻击方法

机器学习攻击依赖大量数据，在无法提前知道的情况下，利用已经获取的相关数据，学习数据规律进行分析，用来构建 PUF 激励响应模型，从而达到在新的输入数据下，精准预测输出响应的功能[12]。机器学习算法根据训练集样本的不同，可分为监督学习算法、无监督学习算法和强化学习算法[13]。监督学习算法，训练集中的样本均有标签，在训练过程中使用后标签进行模型构建的调整，可使预测的精度不断提升，最终正确预测未知数据集；无监督学习正好与监督学习相反，训练集中未设置标签，可根据解决问题的不同分为关联分析、聚类问题以及维度约减；强化学习的样本需要通过和环境不断进行交互而获得，区别于监督学习算法需要通过人工收集并标注。

常见的机器学习算法包括逻辑回归(Logistic Regression，LR)、人工神经网络(Artificial Neural Networks，ANN)和支持向量机(Support Vector Machines，SVM)等。LR 是一种广泛应用于分类问题的机器学习算法，可以用于二分类和多分类任务。LR 的基本思想是建立一个逻辑回归模型，通过对样本特征进行线性组合得到预测结果，然后通过 Sigmoid 函数将预测结果映射到 0 和 1 之间的概率值：

$$h(x)=\frac{1}{1+e^{-y}} \tag{4}$$

其中，y表示线性回归函数值。SVM 是一种常用于分类和回归问题的机器学习算法，通过找到一个最优的超平面来将不同类别的数据分开，使两个不同类别的数据点到超平面的距离最大化，样本到超平面的距离可表示为：

$$\gamma^{(n)}=\frac{y^{(n)}\left(\omega^{\mathrm{T}}\cdot x+b\right)}{\|\omega\|} \tag{5}$$

其中，法向量ω表示超平面的方向，x表示样本点，b表示超平面到原点的距离，n表示样本的组数。借助超平面，SVM 可以处理高维数据，并且可以使用核函数将非线性问题转化为线性问题。ANN 是一种模仿人脑神经网络结构和工作原理的计算模型。ANN 由大量神经元组成，这些神经元通过加权连接构成一个网络。通过对网络进行训练，ANN 可以学习输入数据之间的复杂关系，从而进行分类或回归任务。ANN 通常被分为多层神经网络(MLP)和循环神经网络(RNN)。

PUF 电路的输出响应为一串二进制序列，仅包含 0 和 1，通常采取分类算法以构建 PUF 电路输入输出模型。由于仲裁器 PUF 上下两条路径信号延时不同，且延时差呈线性，其产生的激励响应对可通过线性的方式进行分类，因此仲裁器 PUF 极易受到 ML 攻击。

2.3 混沌算法

2.3.1 混沌映射发生器

混沌来自于任意随时间变化的非线性动力系统。混沌映射是一种确定性的随机现象，具有非线性、不可预测等特性，而且在相空间中存在奇异吸引子，因此，其初始条件的微小变化能导致系统演化轨迹产生巨大差异[14]。混沌映射发生器是一种产生随机数序列的电路，利用混沌映射的非线性动力学性质产生一系列伪随

机的数值，通常用于密码学、模拟物理系统、随机数生成等领域[15]。

混沌映射发生器可以通过选择不同的混沌映射方程和参数组合来产生不同的随机数序列。常用的混沌映射方程包括 Logistic 映射、Henon 映射、Tent 映射等。其中，Logistic 映射是最常用的混沌映射方程之一，其形式为：

$$x_{n+1} = \mu x_n \left(1 - x_n\right) \tag{6}$$

其中μ表示映射的参数。图 2 为 Logistic 映射，也称虫口模型，当$3.569945 < \mu \leq 4$时，Logistic 映射处于混沌状态，即由初始条件x_0在 Logistic 映射的作用下所产生的序列$\{x_k\}$是非周期、不收敛的，并对初始值非常敏感；当$\mu = 4$时，该映射是满射，产生的混沌序列在区间$[0,1]$上具有遍历性。由于 Logistic 映射具有与白噪声相似的特性、简单和初始值敏感性的特点，因此很多混沌图像加密算法都是基于 Logistic 映射的。

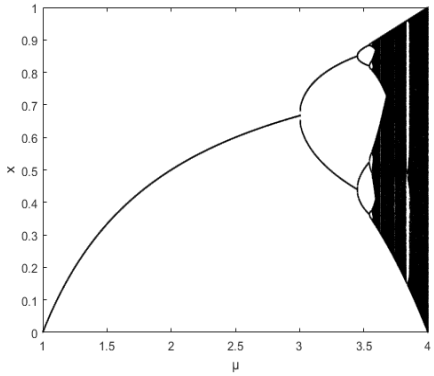


图 2 Logistic 映射(虫口模型)

3 CM-PUF 电路设计

3.1 CM-PUF 抗 ML 攻击原理

为产生与原始激励位数相同的混淆值，提出一种基于激励的 Logistic 混沌映射。以 64 位的激励为例，当式(6)Logistic 映射μ取 4 时，若x_n的取值范围为$[0,1)$，则x_{n+1}的取值范围也为$[0,1)$，$2^{64} \cdot x_n$的取值范围为$[0, 2^{64})$，则将式(7)等式两边同时乘2^{64}得：

$$2^{64} \cdot x_{n+1} = 4 \cdot 2^{64} \cdot x_n \left(2^{64} - 2^{64} \cdot x_n\right) / 2^{64} \tag{7}$$

令$2^{64} \cdot x_n = X_n$，则：

$$X_{n+1} = 4X_n \left(2^{64} - X_n\right) / 2^{64} \tag{8}$$

其中若$X_n \in [0, 2^{64})$，则$X_{n+1} \in [0, 2^{64})$。将混沌映射发生器里对应的混沌映射替换为基于 64 位激励的 Logistic 映射：

$$X_{n+1} = 4X_n \left(2^{64} - X_n\right) / 2^{64} \tag{8}$$

其中，混沌初值X_0为原始激励，由于激励为 64 位，则X_0取值范围为$0 \leq X_0 \leq 2^{64} - 1$，则$X_n$取值范围为$0 \leq X_n \leq 2^{64} - 1, n = 1,2,3,\cdots$，可以遍历 64 位的所有值。将激励作为混沌映射初值，利用 PUF 运行时间与混沌映射深度的内在关系，产生不可预测的混沌映射，降低激励与响应之间的线性关系。

为进一步降低激励与响应之间的线性关系，提出一种抗机器学习攻击 PUF 设计，即 CM-PUF，使用响应反馈结合混淆后的混沌映射值作循环移位变换，将变换后的值作为新的激励。由于混淆后的混沌映射值为 64 位，因此需要使用 6 位响应信号作为循环移位的位移控制量，以产生 0 到 63 位的循环移位范围。为尽可能减小 6 位响应值之间的相关性，在 APUF 的每 10 个选择单元之间插入一个仲裁器，每一个仲裁器产生一位响应信号，通过插入 6 个仲裁器，可以产生 6 位响应。这种方案的改进可以有效地提高 PUF 的安全性和可靠性。

CM-PUF 的原理框图如图 3 所示，APUF 与混沌映射发生器相结合可以形成一个闭环结构来混淆激励响应对。该 PUF 的特性如下，首先在原始 64 位激励输入 APUF 的同时，将混沌映射发生器的初值更新为当前输入的激励从而产生混沌映射，每一个时钟信号上升沿产生一个混沌映射，在 PUF 响应生成之前不断产生混沌映射，直到产生响应，此时产生的中间响应不会直接输出，其中时钟信号的周期为 10ns，而 64 位

PUF 产生响应时间为 μs 级；然后通过中间响应的值对当前混沌值作循环移位变换，将变换后的 64 位信号值作为新的激励；最后由反馈模块将原始激励替换为新的激励输入 PUF，产生最终响应。

基本步骤如下：

步骤(1)：原始 64 位激励C同时输入 APUF 和混沌映射发生器中，混沌映射发生器的初值为 64 位的X_0，其中$X_0 = C$，混沌映射发生器中使用基于 64 位激励的 Logistic 映射：

$$\begin{cases} X_0 = C, t = 0 \\ X_{n+1} = 4X_n\left(2^{64} - X_n\right)/2^{64}, t = 1,2,3,\cdots \end{cases} \tag{9}$$

其中t为时钟信号，每个时钟信号上升沿产生一个混沌值，时钟信号的频率为 100MHz。

步骤(2)：在 APUF 每 10 个选择单元之间插入一个仲裁器产生 1 位响应信号，分别为$R_{14}, R_{24}, R_{34}, R_{44}, R_{54}, R_{64}$，组成一个 6 位的控制信号$S = \{R_{14}, R_{24}, R_{34}, R_{44}, R_{54}, R_{64}\}$，在最后一个仲裁器产生响应时，选择当前 64 位的混沌值X_n，将X_n向右循环移位S位，组成新的激励C^*则：

$$C^* = \left\{X_n[S-1:0], X_n[63:S]\right\} \tag{10}$$

步骤(3)：将原始激励替换为新激励C^*输入 PUF。

步骤(4)：PUF 产生的响应作为最终响应R。

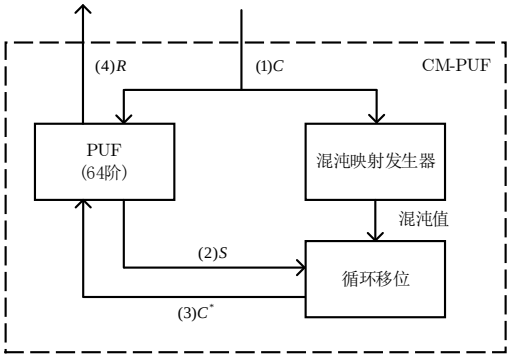


图 3 CM-PUF 原理框图

3.2 CM-PUF 电路结构

CM-PUF 电路结构如图 4 所示，以 64 位的 PUF 单元为例，每 10 个单元之间插入 1 个仲裁器，共 6 个仲裁器，信号经过每个仲裁器时都会产生 1 位响应，则可产生 6 位中间响应S。

输入信号先经过激励选择模块，当输入信号为原始激励C时，状态值变为 1，原始激励C将输入 PUF 产生中间响应S；当输入信号为循环移位变换后的激励C^*时，状态值变为 0，从而使C^*输入 PUF 产生响应R。而 PUF 产生的响应先经过响应选择模块，若当前状态值为 1，则中间响应S与当前混沌映射发生器中的混沌值作循环移位变换产生新激励C^*；若当前状态值为 0，则直接输出响应R。

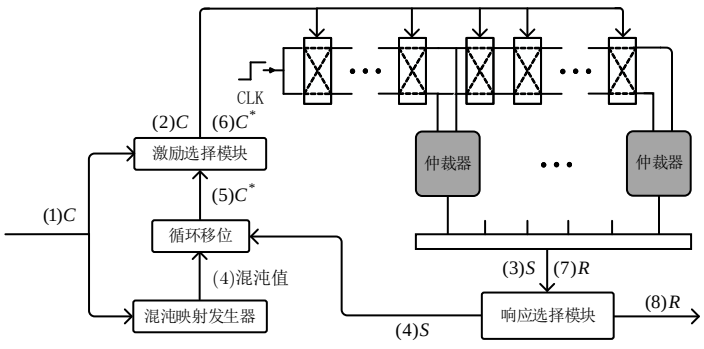
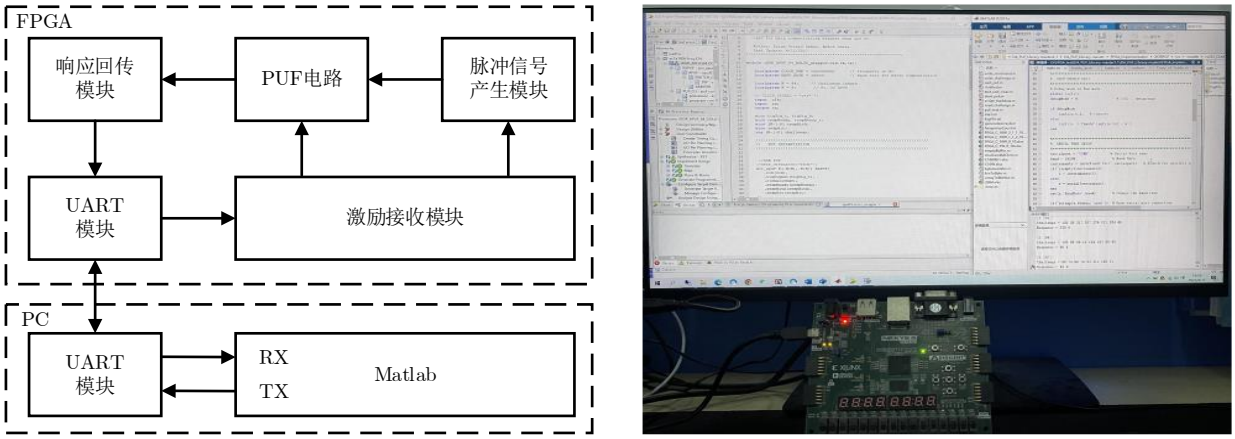


图 4 CM-PUF 电路结构

4 实验结果与分析

4.1 实验平台

CM-PUF 采用 Xilinx Artix-7 FPGA (XC7A100T)开发板实现，并利用 Xilinx ISE 14.7 和 Xilinx PlanAhead 14.7 工具对 PUF 电路分别进行逻辑综合和布局布线。测试系统如图 5 所示，首先 PC 端利用 Matlab 并结合 FPGA 开发板自带的 MicroBlaze 嵌入式微控制器实现测试平台的搭建，通过 PC 端通讯接口集成的通用异步收发传输器(Universal Asynchronous Receiver/Transmitter，UART)模块发送生成的随机激励至 FPGA；其次通过 FPGA 的 UART 模块接收来自 PC 端的激励数据，由 MicroBlaze 完成对数据的处理，驱动 FPGA 各硬件电路实现周期性方波信号的发生，PUF 电路响应生成以及输出响应的回传；UART 模块在接收到输出响应后，将其发送至 PC 端；最后 PC 端利用接收到的响应信号，结合 Matlab 实现对信号的分析处理。

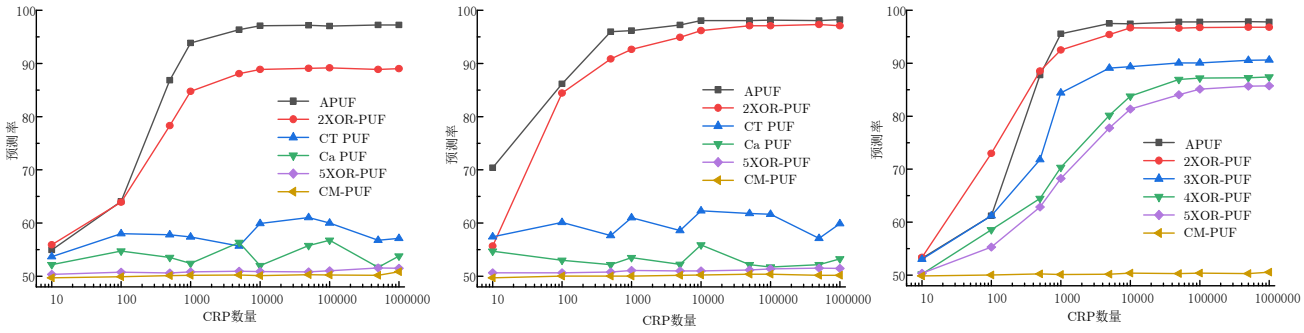


(a)测试系统结构　　(b) 测试平台

图 5 测试平台与测试系统结构

4.2 抗攻击能力

为评估 CM-PUF 的安全性，基于 LR，SVM 和 ANN 3 种常见的 ML 攻击方法对 FPGA 硬件实现的 CM-PUF 电路分析抗攻击能力，攻击预测率与实验所用 CRPs 数量之间的关系如图 6 所示，其中图 6(a)为 LR 攻击的预测率，图 6(b)为 SVM 攻击的预测率，图 6(c)为 ANN 攻击的预测率。可以发现，CM-PUF 的 LR 和 SVM 攻击预测率均接近 50%，相当于随机猜测，与 APUF 和 2XOR-PUF 相比，有更好的抗 LR 和 SVM 攻击能力，相较 CT PUF 和 CaPUF 也有一定优势，抗 LR 和 SVM 攻击能力与 5XOR-PUF 相当；另外，CM-PUF 的 ANN 攻击预测率也接近 50%，相较 APUF、2XOR-PUF、3XOR-PUF、4XOR-PUF 和 5XOR-PUF，有更好的抗 ANN 攻击能力。因此，CM-PUF 具有良好的抗 ML 攻击能力。



(a) LR 攻击　　(b) SVM 攻击　　(c)ANN 攻击

图 6 机器学习攻击预测率

4.3 统计特性分析

4.3.1 随机性

随机性可以通过美国国家标准与技术研究院（National Institute of Standards and Technology，NIST）提出的 NIST SpecialPublication800-22 套件进行测试。该测试主要包括：频率、场内频数、游程、块内最

长游程、二元矩阵秩、离散傅里叶变换、非重叠模块匹配、重叠模块匹配、Maurer 的通用统计、伦佩尔-齐夫压缩、线性复杂度、序列、近似熵、累积和、随机游动等 15 项检验[16]。在 NIST 测试中，测试数据与一个或多个已知分布进行比较，如均匀分布或正态分布等。通过比较，可以计算出一个 P-value，用于表示观察到的数据在相应分布中出现的概率，并将 P-value 与预先设定的显著性水平α比较，如果 P-value 大于α时，则接受原假设，判断该待测序列通过了该项随机性测试。相反，当 P-value 小于α时，则拒绝原假设。通常，显著性水平设置为 0.01。对 CM-PUF 和 APUF 进行 NIST 随机性测试。为既能满足使 PUF 所有位翻转，又能减小激励对随机性的影响，实验采用$[0, 2^{64}-1]$的 100 万组等差数列作为激励，将每个 PUF 所产生的响应数据分成 50 组，并对每组数据进行 NIST 随机性测试。测试结果如表 1 所示，结果表明，CM-PUF 在 50 组测试数据中通过了 7 项 NIST 检测项，相较 APUF，有较好的随机性。为反映出输出与响应间相关程度，利用自相关函数(Autocorrelation Function，ACF)对 PUF 响应进行测试，测试结果如图 7 所示。结果显示，在 95%置信区间内，CM-PUF 响应的 ACF 值为 0.0045，接近理想值 0，反映出响应具有良好的空间独立性。

表 1 CM-PUF 的随机性 NIST 测试结果

NIST 检测项	CM-PUF		APUF	
	P-value	Proportion	P-value	Proportion
频率检验	0.000000*	39/50	0.000000*	4/50
场内频数检验	**0.026948**	**50/50**	0.000000*	0/50
游程检验	0.000000*	28/50	0.000000*	0/50
块内最长游程检验	0.000000*	36/50	0.000000*	0/50
二元矩阵秩检验	**0.236810**	**50/50**	**0.574903**	**47/50**
离散傅里叶变换检验	**0.108791**	**50/50**	0.000000*	0/50
非重叠模块匹配检验	**0.991468**	**50/50**	0.000000*	37/50
重叠模块匹配检验	**0.383827**	**49/50**	0.000000*	11/50
线性复杂度检验	**0.779188**	**48/50**	**0.236810**	**49/50**
累积和检验	0.000000*	40/50	0.000000*	0/50
通用统计检验	0.000000*	50/50	0.000000*	0/50
近似熵检验	**0.013569**	**46/50**	0.000000*	0/50
序列检验	0.000000*	26/50	0.000000*	0/50

*表示 P-value 的值非常小，接近于零

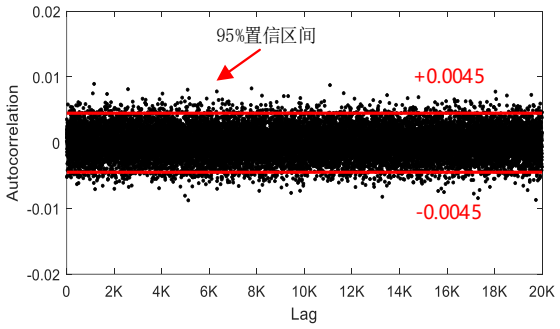


图 7 自相关性测试结果

4.3.2 唯一性

唯一性用于评估不同 PUF 输出响应之间的差异，用平均片间汉明距离(Inter-Hamming Distance，Inter-HD)表示。理想情况下，相同激励作用于两个不同 PUF 时，输出响应间应有一半比特位不同。数值上，通常采用相同激励作用于K个不同 PUF，输出响应间不同位数的统计值来表示唯一性，其计算公式为:

$$\text{Uniqueness} = \frac{2}{K(K-1)} \sum_{i=1}^{k-1} \sum_{j=j+1}^{k} \frac{HD(R(i)R(j))}{n} \times 100\% \tag{11}$$

其中R_i和R_j分别表示第i和第j个 PUF 在相同激励作用下生成的n比特响应。唯一性可采用平均 Inter-HD 的统计直方图形象化表示。实验测得 CM-PUF 平均 Inter-HD 的统计直方图，经归一化处理和高斯拟合后所得曲线如图右半部分所示，拟合曲线如图 9 所示(红线部分)，归一化平均片间汉明距离为 0.50078，唯一性为 50.078%，接近理想值 50%，具有良好的唯一性。

4.3.3 稳定性

稳定性表示 PUF 电路在任何情况下给定相同的激励都能输出相同的响应。然而，受到环境温度变化、电源电压波动以及器件老化等因素影响，响应中不可避免的会混入噪声。因此，有必要引入稳定性指标来衡量 PUF 电路复现响应的能力。稳定性指标通常采用平均片内汉明距离(Intra-Hamming Distance，Intra-HD)来衡量，即通过比较相同激励作用下存在于不同环境或电压条件下相同 PUF 个体输出响应间不同位数，其计算公式为:

$$\text{Reliability} = \left(1 - \frac{1}{m} \sum_{i=1}^{m} \frac{HD(R, R')}{n}\right) \times 100\% \tag{12}$$

其中R和R'分别表示同一个 PUF 在相同激励作用下生成的n比特响应。将 CM-PUF 置于常温常压环境下对输出响应进行循环读取，所得高斯拟合曲线如图左半部分所示，拟合曲线如图 8 所示(蓝线部分)，归一化片内汉明距离为 0.0126，对应的稳定性为 98. 74%，接近理想值 100%，具有良好的稳定性。

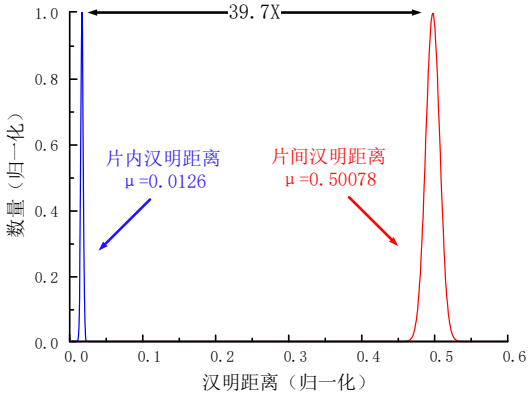


图 8 片间汉明距离和片内汉明距离拟合曲线

4.4 综合性能对比

FPGA 开发板由可编辑逻辑单元、可编程 I/O 单元和布线资源 3 部分构成，其中可编程逻辑单元由 2 个 Slice 构成，一个 Slice 包含 4 个 6 输入查找表(Look Up Table，LUT)、3 个数据选择器、1 个进位链和 8 个触发器。CM-PUF 结构使用 240 个 Slice，其中包含 491 个 LUTs 和 546 个 Flip Flops。

表 2 列出了 CM-PUF 与 APUF、2XOR-PUF、5XOR-PUF、CaPUF 和 CT PUF 的电路结构硬件资源消耗数目，可以发现 CM-PUF 消耗的 LUTs 数量较 5XOR-PUF、CaPUF、CT PUF 有所减少，消耗的触发器数量较 CT PUF 有所增加，占用 Slice 数量与 CT PUF 相当。

实验采用 LR、ANN 和 SVM 三种 ML 算法分别对 CM-PUF 进行攻击，攻击预测率统计表数据如表 2 所示。表中最终预测率均在训练集数量为 10 万时获得，分析表格数据可知 CM-PUF 相较 APUF、2XOR-PUF、5XOR-PUR、CaPUF 和 CT PUF，具备更强的抗攻击能力，且 CM-PUF 的随机性、唯一性和稳定性也较为优秀。

表 2 各类 PUF 电路统计特性、抗攻击能力与硬件开销对比

PUF 类型	随机性	唯一性	稳定性	机器学习攻击			硬件开销		
				LR	ANN	SVM	LUTs	FFs	SLICEs
APUF[1]	50.73%	49.56%	99.02%	99.30%	99.12%	98.98%	266	278	109
2XOR-PUF[1]	50.13%	49.35%	98.36%	96.91%	96.63%	96.74%	469	279	177
5XOR-PUF[1]	50.07%	50.59%	92.54%	53.89%	85.47%	54.31%	786	282	440
CT PUF[4]	49%-50%	49%-50%	92%	57%	-	55%	731	486	244
CaPUF[6]	46.38%	47.03%	88.47%	50%-56%	-	50%-56%	5723	-	-
CM-PUF	50.533%	50.078%	98.74%	50.332%	50.33%	50.239%	491	546	240

5 结论

通过对混沌算法的研究，提出一种针对强 PUF 的抗 ML 攻击电路，结合响应反馈、混沌映射混淆等多种抗攻击方法，防止攻击者获取真正作用于原始 PUF 上的 CRPs 进行 ML 攻击，能显著降低激励与响应之间的线性关系，具有抗攻击能力强和易于 FPGA 实现等特点。实验结果表明，即便用于建模的 CRPs 数高达 100 万组，LR，ANN 和 SVM 的攻击预测率均接近 50%。所以，CM-PUF 可有效抵抗 ML 攻击。此外，CM-PUF 的随机性、唯一性和可靠性等关键特性也较为优秀，可广泛用于物联网认证设备等领域。

参 考 文 献

[1] 汪鹏君，连佳娜，陈博. 基于序列密码的强 PUF 抗机器学习攻击方法[J]. 电子与信息学报, 2021, 43(09):2474-2481.

[2] Zhou Z, Li G, Wang P. A challenge-screening strategy for enhancing the stability of strong PUF based on machine learning[J]. Microelectronics Journal, 2023, 131: 105667.

[3] Ali R, Ma H, Hou Z, et al. A Reconfigurable Arbiter MPUF With High Resistance Against Machine Learning Attack[J]. IEEE Transactions on Magnetics, 2021, 57(10): 1-7.

[4] Zhang J, Shen C, Guo Z, et al. CT PUF: Configurable Tristate PUF Against Machine Learning Attacks for IoT Security[J]. IEEE Internet of Things Journal, 2022, 9(16): 14452-14462.

[5] Wu L, Hu Y, Zhang K, et al. FLAM-PUF: A Response–Feedback-Based Lightweight Anti-Machine-Learning-Attack PUF[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(11): 4433-4444.

[6] Nassar H, Bauer L, Henkel J. CaPUF: Cascaded PUF Structure for Machine Learning Resiliency[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(11): 4349-4360.

[7] Zhang J, Xu C, Law M K, et al. A 4T/Cell Amplifier-Chain-Based XOR PUF With Strong Machine Learning Attack Resilience[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(1): 366-377.

[8] Oun A, Niamat M. Design of a Delay-Based FPGA PUF Resistant to Machine Learning Attacks[C]. 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2021: 865-868.

[9] Avvaru S V S, Zeng Z, Parhi K K. Homogeneous and Heterogeneous Feed-Forward XOR Physical Unclonable Functions[J]. IEEE Transactions on Information Forensics and Security, 2020, 15:2485-2498.

[10] Shah N, Chatterjee D, Sapui B, et al. Introducing Recurrence in Strong PUFs for Enhanced Machine Learning Attack Resistance[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2021, 11(2):319-332.

[11] Xu Y, Lao Y, Liu W, et al. Mathematical modeling analysis of strong physical unclonable functions[J]. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 2020, 39(12): 4426–4438.

[12] Pan P, Wang H, Shen L, et al. Equivalence of Joint ML-Decoding and Separate MMSE-ML Decoding for Training-Based MIMO Systems[J]. IEEE Access, 2019, 7:178862-178869.

[13] 杨剑锋，乔佩蕊，李永梅，等. 机器学习分类问题及算法研究综述[J]. 统计与决策, 2019, 35(6): 36-40.

[14] 魏旭晖, 王辉. 异构网络下 TCP 拥塞控制的混沌特性分析[J]. 系统仿真学报, 2015, 27(07):1541-1547.

[15] Wannaboon C, Ketthong P. A Simple Random-Bit Generator Implemented on FPGA Based on Signum Chaotic Map[C]. 2022 International Conference on Digital Government Technology and Innovation (DGTi-CON), Bangkok, Thailand, 2022: 101-104.

[16] Lee C Y, Bharathi K, Lansford J, et al. NIST-Lite: Randomness Testing of RNGs on an Energy-Constrained Platform[C]. IEEE 39th International Conference on Computer Design (ICCD), 2021: 41-48.

方皓冉：男，1998 年生，研究生，研究方向为物理不可克隆函数的攻击与防御.
李刚：男，1988 年生，副教授，研究方向为集成电路设计、集成电路硬件安全、存内计算.
汪鹏君：男，1966 年生，教授，研究方向为集成电路设计、信息安全等技术及其相关理论.

SI PUF: An SRAM-and-Inverter-based PUF with Bit Error Rate 0.0053% and 0.073/0.042 pJ/bit

Li Ni, Pengjun Wang, Yuejun Zhang, Xiangyu Li, Gang Li, Lin Ding, and Jiliang Zhang

***Abstract*—The physical unclonable function (PUF) can generate a unique identifier for each chip, ideal for key generation and chip anti-counterfeiting. The reliability of PUF is paramount and thus is one of the significant challenges to PUF design. This paper proposes a novel SRAM-and-Inverter-based PUF (SI PUF) which can work as either an SRAM PUF or an inverter PUF, depending on the input configuration signal. A zero-overhead bit configuration strategy (BCS) is proposed to enhance the reliability of PUF. Moreover, the working voltage of the sub-threshold level and the well-designed discharge stage ensure that our SI PUF can work with low power consumption. The tested results of chips fabricated in 40-nm CMOS show that our SI PUF has only 0.0053% of the worst bit error rate (BER) in the working conditions of -50 to 125°C and 0.75 to 1.5V, with 0.073/0.042 pJ/bit of power consumption. The low BER and energy overhead illustrate that our SI PUF is more suitable for resource-limited devices, compared to DAC'22, ISSCC'17, and JSSC'20.**

***Index Terms*—Physical Unclonable Function, Hardware Security, Reliable, Key Generation.**

I. Introduction

THE Internet of Things (IoT) enables smart devices to connect to the Internet, facilitating information exchange and communication. However, with the increasing number of connected intelligent devices in the IoT, security and privacy risks have become a significant concern. Moreover, due to the severe limitations in resources such as memory, CPU, and power consumption within the IoT, providing a secure and lightweight solution for key storage becomes paramount.

Physical unclonable function (PUF) [1] stands as a highly secure and low-overhead hardware security primitive that generates a series of unclonable responses by harnessing the variation in chip fabrication processes. Based on the number of challenge-response pairs (CRPs), PUFs can be classified as either strong PUF or weak PUF. While weak PUFs yield fewer responses than strong PUF, they demonstrate better overall performance and are extensively leveraged for key generation.

Manuscript received July 30, 2023. This work is supported by the National Natural Science Foundation of China (No. 62234008, No. 62174121, and No. U20A20202), the National Key Research and Development Program of China (2018YFB100), Wenzhou Basic Scientific Research Projects (G20210023). (*Corresponding author: Pengjun Wang and Yuejun Zhang*)

L. Ni, P. J. Wang and G. Li are with College of Electrical and Electronic Engineering, Wenzhou University, Wenzhou, 325035, China. (e-mails: wangpengjun@wzu.edu.cn)

X. Y. Li and Y. J. Zhang are with Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, 315211, China. (e-mails: zhangyuejun@nbu.edu.cn)

L. Ding and J. L. Zhang are with College of Semiconductors (College of Integrated Circuits), Hunan University, Changsha 410082, China. (e-mails: zhangjiliang@hnu.edu.cn)

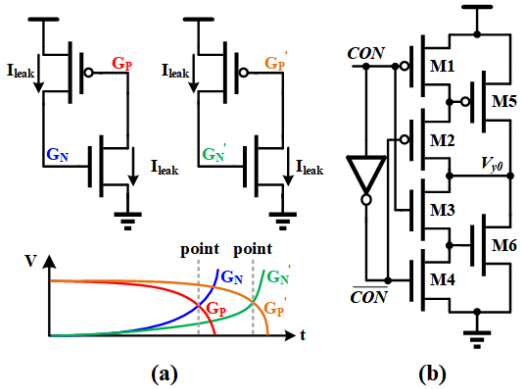


Fig. 1: (a) Entropy source of leakage PUF [7]; (b) Entropy source of DA PUF[6].

Nonetheless, weak PUFs encounter reliability concerns in the face of voltage and temperature fluctuations in the operating environment.

Typically, techniques for improving reliability, such as temporary majority voting (TMV) [2] and error correcting code (ECC) [3], are employed to enhance reliability of PUFs . However, these schemes often require extensive testing time or result in hardware overhead. To substantially enhance PUF reliability while minimizing cost, this brief proposes a novel high-reliability SI PUF.

The primary innovation of the SI PUF lies in its circuit structure, known as the SI PUF cell. This cell can be configured to function as either an SRAM PUF [4] or an inverter PUF [5]. By biasing the supply voltage of the PUF cells to the sub-threshold level and increasing the discharge stage, our SI PUF generates much lower power consumption compared to state-of-the-art, such as DAC'22 [6] and JSSC'20 [7]. Moreover, we propose a simple but useful bit configuration strategy (BCS) to further enhance the reliability of PUF without any extra hardware overhead.

The proposed SI PUF is fabricated with 40nm CMOS process, and the experimental results demonstrate the SI PUF achieves 0.0053% native bit error rate (BER). Furthermore, the PUF's temperature sensitivity was found to be 0.018%/10°C within the range of -50-125°C, while its voltage sensitivity was measured to be 0.014%/0.1V within the range of 0.75-1.5V.

The remainder of this paper is structured as follows. In section II related works are discussed. The proposed SI PUF scheme is presented in section III. Section IV presents the experimental results and compares them with prior works. Finally, Section V concludes the entire work.

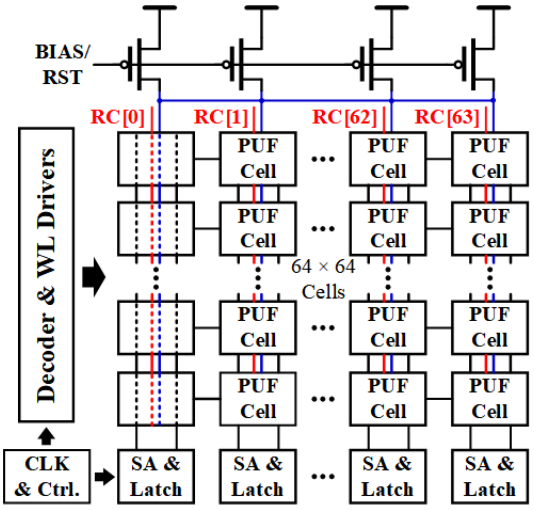


Fig. 2: Top-level architecture of the PUF.

II. Related Works

PUF, as a distinctive identifier of a chip, harnesses process variations inherent in chip manufacturing to generate a unique and immutable fingerprint. Thus, PUF can be efficiently utilized for chip anti-counterfeiting, key generation, and other related domains. Currently, numerous scholars proposed a significant number of high-reliability PUF works, but most of them do not achieve both reliability and low power consumption simultaneously. Lee et al.[7] proposed a leakage PUF cell design, depicted in Fig. 1(a). The voltage of G_P gradually decreases while G_N rises. The voltage developed on G_N and G_P creates positive feedback, accelerating the increase in leakage current. When the accumulated voltage approaches the threshold voltage (V_{th}) of PMOS or NMOS transistors, the drain current of PMOS/NMOS transistors experiences a rapid increase, leading to a sharp voltage transition of G_N to VDD and G_P to ground, resulting in a latching event. However, the need for an extended PUF response generation process results in excessive power consumption. Zhang et al.[6] proposed a dual-state analog PUF to improve reliability, as depicted in Fig. 1(b). When *CON* = 1, the PUF cell operates as an inverter PUF, and when *CON* = 0, it is configured as a leakage PUF. However, when the PUF cell is inactive, it still experiences electricity leakage. Due to the absence of circuitry designed to prevent leakage, the circuit incurs significant power loss.

In this paper, a novel SI PUF structure is proposed, which show high reliability and low power consumption. Moreover we proposes a simple but useful approach for improving the reliability of PUFs through the use of a SI PUF circuit and a bit configuration strategy, without incurring any additional sacrifices in terms of area or power consumption.

III. The Proposed SRAM-and-Inverter-based PUF

In this section, we first presents an overview of the top-level architecture of the SI PUF. Then introduce the structure of the SI PUF cell, explaining its functional principles and properties in detail. Finally, we will propose a bit configuration strategy for the SI PUF.

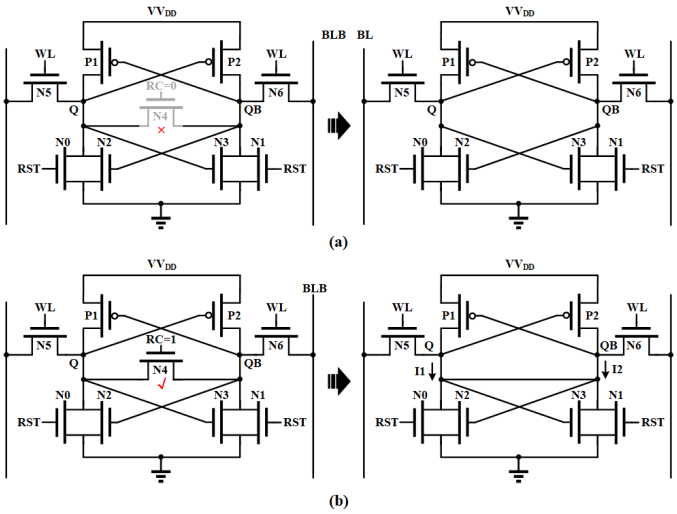


Fig. 3: (a) when *RC* = 0, cell works in the SRAM mode and (b) when *RC* = 1, cell works in the inverter mode.

A. The top architecture of SI PUF

Fig. 2 depicts the top-level architecture of the proposed SI PUF, which comprises a PUF cell array, decoder (with WL drive unit), timing control circuit, sense amplifier (SA), and latch array. The PUF unit array consists of 64×64 PUF cells, each of which is configured to a high or low state based on the *RC* signal. Then, the decoder is activated to enable a row of PUF cells, and the resulting signal is amplified by SA and latched. To save the area of the PUF array, the 64 PUF cells in each column share a sense amplifier. The peripheral circuit is SRAM-style to achieve high-speed and parallel readout of PUF.

B. SI PUF Cell

Fig. 3 is the entropy source circuit of a SI PUF cell. The P1, P2, N2, and N3 form a pair of cross-coupled inverters. Additionally, a configurable transistor, N4, is included which can alter the operating mode of the PUF cell. The operation of the PUF cell can be divided into two stages, namely pre-discharge and operation:

- When RST and WL are at the high level, P0 is turned off, and N0 and N1 are turned on. Thus, BL/BLB discharged to a low level (discharge stage). Then, Q/QB is in the metastable state of ”0/0”
- Subsequently, when RST and WL change from high level to low level simultaneously, P0 is turned on, and N0 and N1 are turned off. Then, Q/QB immediately transitions from the metastable state ”0/0” to the stable state (operation stage).

Due to the greater process variation and lower power consumption exhibited by MOS transistors operating in the sub-threshold range, they are an ideal choice to serve as entropy sources for PUFs. The proposed PUF is designed to function in two independent modes, namely the SRAM mode and the inverter mode, which correspond to input signal values of *RC* = 0 and 1, respectively. As illustrated in Fig. 3(a), when the *RC* input is connected to a low-level voltage, the PUF operates in the SRAM mode. With the configurable transistor

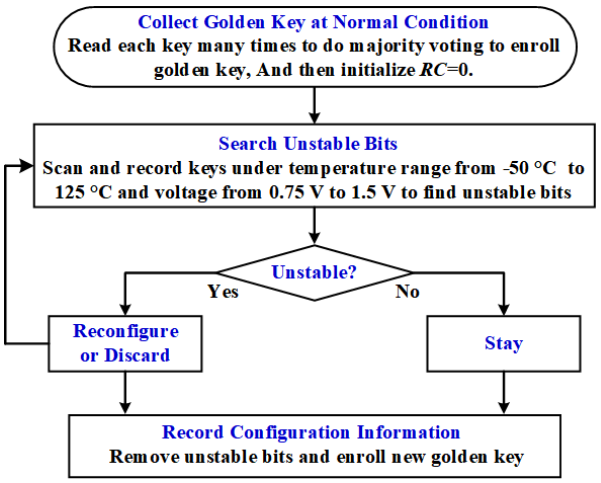


Fig. 4: Bit configuration strategy.

N4 turned off, the P1, P2, N2, and N3 transistors combine to form a conventional SRAM PUF. During the charging phase, when VV_{DD} is applied, the Q and QB nodes become charged simultaneously. As a result, the gate-source voltage difference between transistors P1 and P2 decreases gradually, resulting in a corresponding decrease in the current flowing through P1 and P2. Ideally, the voltages at the Q and QB nodes should remain at mid-level voltage. However, due to the inevitable process deviation of the transistors, this is reflected in the difference in the rate of charging of the Q and QB node voltages. The final state of the voltage at V_Q can be expressed as follows:

$$V_Q = \begin{cases} 1, \text{if} D_{P1} + D_{N2} > D_{P2} + D_{N3}, \\ 0, \text{if} D_{P1} + D_{N2} < D_{P2} + D_{N3}, \\ \text{uncertain, otherwise}, \end{cases} \quad (1)$$

where D is the driving capability of the MOS transistor. If the sum of the drive capabilities of transistors P1 and N2 is greater than the sum of the drive capabilities of P2 and N3, the voltage V_Q is at the high level. On the contrary, V_Q is at the low level. However, if the driving capabilities of both sides are similar, the level of V_Q is uncertain which determined by the noise, and the PUF response is unstable.

As shown in Fig. 3(b), when *RC* is connected to a high voltage level, the PUF cell operates in inverter mode. At this point, the configurable transistor N4 is activated. The currents I_1 and I_2 flowing through the two branches can be expressed as follows:

$$I_1 = I_s \exp(\frac{V_Q - V_{thN2}}{nV_T}), \quad (2)$$

$$I_2 = I_s \exp(\frac{V_Q - V_{thN3}}{nV_T}), \quad (3)$$

where V_{thN2} and V_{thN3} indicate the threshold voltages of N2 and N3, respectively. V_T denotes the thermal voltage *kT/q*, and I_s represents the characteristic current. The output voltage V_Q can be expressed as follows:

$$I_1 + I_2 = I_{s1} \exp(\frac{V_{DD} - V_{BIAS} - |V_{thP0}|}{nV_T}), \quad (4)$$

$$V_Q = nV_T \ln(\tfrac{I_{s1}}{I_s}) + V_{DD} + V_{thN2} - V_{BIAS} - |V_{thP0}| - nV_T \ln\left[1 + \exp\left(\tfrac{V_{thN2} - V_{thN3}}{nV_T}\right)\right], \quad (5)$$

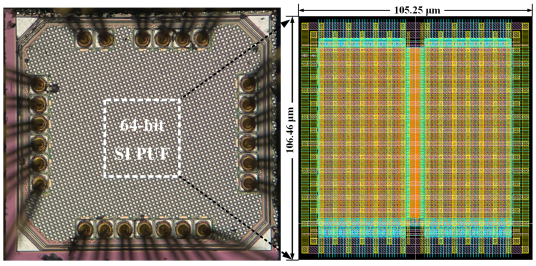


Fig. 5: PUF chip micrograph and PUF array layout.

where $|V_{thp0}|$ is the threshold voltage of PMOS transistor. I_s and I_{s1} represent the characteristic current of PMOS and NMOS, respectively. Due to the significant variation of process deviation, the threshold voltage exhibits a substantial change. Consequently, the PUF cell in inverter mode exhibits a high level of entropy. As stated previously, when *RC*=0, the PUF cell is similar to an SRAM PUF. However, when *RC*=1, the PUF cell transforms into an inverter PUF. Equations (1) and (5) clearly demonstrate that the responses produced by the two PUF modes are mutually independent. In the SRAM mode, the PUF response is influenced by the process variation of the drive capability of the two transistor pairs(P1, N2 and P2, N3). Conversely, in the inverter mode, the PUF response is collectively influenced by the process variation of all transistors within the PUF cell.

C. Bit configuration strategy

To enhance reliability, we propose a reliability improvement approach, which combines online and offline processing in an independent two-state mode. The principle of this technique involves conducting multiple tests to find unreliable cells and performing reconstruction to improve overall reliability

As illustrated in Fig. 4, the first step involves setting *RC* to ”0”, configuring all PUF cells in SRAM mode, extracting PUF responses multiple times, and utilizing TMV to obtain a stable response to complete the registration of the gold key. Next, we search for unstable PUF responses by scanning the PUF cell and recording its keys across a broad range of temperatures and voltages, from -50°C to 125°C and 0.75V to 1.5V, respectively. We deem a PUF cell stable if it does not flip during the test and decide to retain it. Conversely, we reconfigure the cell, set up the PUF cell in inverter mode, and retest its reliability if it is unreliable. We keep the cell if it is dependable; otherwise, we discard it. The leakage of configuration information does not jeopardize the security of PUF. This is because the configuration information *RC* represents the reliability of the PUF cell, and it cannot deduce the value of the PUF response. Therefore, the adversary cannot deduce the output of the PUF from the stored *RC*.

IV. Measurement Results

To validate the proposed SI PUF design, the chip is fabricated using a 40-nm CMOS process. The die micrograph and the layout of the PUF array are presented in Fig. 5. The complete 64 × 64 PUF array, consisting of the readout circuitry, occupies a silicon area of approximately 300,000 μm^2 (311 μm × 965 μm). The area per bit is 0.96 μm × 2.475 μm.

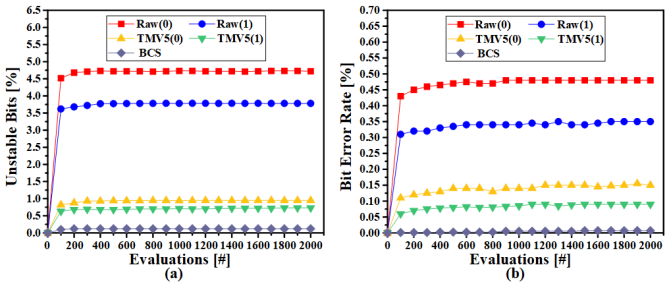


Fig. 6: (a) Unstable Bits and (b) BER evaluated at the nominal condition.

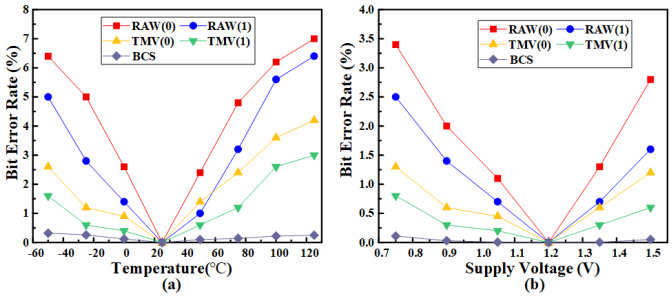


Fig. 7: (a) BER versus temperature variation; (b) BER bits versus supply voltage.

A. Correlation between two modes

To examine the independence of the responses generated by the two modes of the PUF, we employed the McNimar test to assess the correlation between these two states. A significance level (P) of ≤ 0.05 indicates a negligible correlation between the two sets of test data. We calculated correlations independently for the responses generated by the SRAM mode and the INV mode. As depicted in Table I, the difference between the two modes is statistically significant (P=0.000), indicating a lack of correlation between them.

TABLE I: Correlation test.

Program	Value	Exact Sig. (2-sided)
McNemar Test	/	0.000
Num. of test bits	20480	/

B. Native PUF reliability

The reliability of the PUF is evaluated using two metrics, BER and the percentage of unstable bits. The BER represents the probability of a bit flip compared to the golden key, while the percentage of unstable bits indicates the percentage of bit flips under multiple measurements. The reliability tests were conducted on 20 chips (20k bits) at 1.2V/25°C. Fig. 6 shows the BER and the percentage of unstable bits for the PUF at low and high levels. The unstable bits increase with the number of measurements, but after a certain number of evaluations, the local BER stabilizes. Without any stabilization method, the worst native unstable bit of SI PUF's low (high) state is 4.72% (3.78%). By using TMV with 1 output out of 5 samples, the unstable bits are reduced to 0.94% (0.73%). TMV5 reduces the percentage by 80.1%/80.7%, respectively. By adopting the proposed BCS method, the unstable bit positions are reduced to 0.12%, while the worst original bit error rate is reduced from 1.89% (1.3%) to 0.0053%. BCS reduces the percentage by 99.6%.

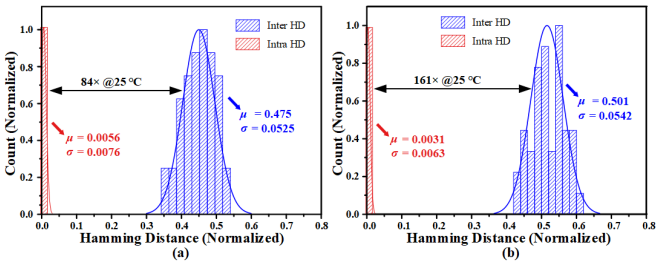


Fig. 8: Measured normalized HD: (a) high state; (b) low state.

TABLE II: NIST 800-22 randomness test.

Test	Stream length	No. of runs	Low-state P-value	High-state P-value	Pass?
Frequency	1024	20	0.793	0.521	Yes
BF	1024	20	0.124	0.482	Yes
Runs	1024	20	0.638	0.437	Yes
LRO	1024	20	0.579	0.524	Yes
FFT	1024	20	0.460	0.443	Yes
NOT	1024	20	0.153	0.392	Yes
Serial	1024	20	0.722	0.427	Yes
AppEn	1024	20	0.834	0.562	Yes
CumSums	1024	20	0.052	0.127	Yes

C. PUF reliability Over Voltage and Temperature Variations

The PUF response is dictated by the disparity between the two subthreshold inverters. However, variations in voltage and temperature influence the Vth of the transistor. We conducted an evaluation to assess the reliability of the PUF response across a temperature range sweeping from -50°C to 125°C. The baseline is the outputs tested with the golden index (1.2V/25°C). Fig. 7(a) illustrates the BER in relation to temperature, considering different methods of stabilization. The average native BER is 6.4% (4.9%) at -50°C and 6.9% (6.3%) at 125°C. Through the utilization of TMV, the worst BER decreased to 4.2% (2.9%). By implementing the proposed BCS approach, the worst BER decreased to 0.32%. This signifies a temperature sensitivity of 0.018%/10°C. As shown in Fig. 7(b), it presents the BER with respect to voltage, employing different stabilization methods. The average native BER of the PUF stands at 3.4% (2.5%) at 0.75V and 2.8% (1.6%) at 1.5V. Following the adoption of BCS, there is a notable decrease in the BER, with the worst recorded rate decreased to 0.11%. This results in 0.014% increment of BER per 0.1V supply voltage variation.

D. Uniqueness and Randomness

Uniqueness and randomness are fundamental evaluation criteria for PUFs. Uniqueness measures how distinct a PUF is from another PUF and is determined by the Hamming distance (HD) between PUFs. Fig. 8. illustrates the measured HD of 20 PUFs in the high (low) state configuration. The normalized inter-die HD shows a mean of 0.475 (0.501), and the average intra-die HD is 0.0056 (0.0031), achieving 84 (161) times separation between inter-die and intra-die. The autocorrelation of 5k PUF bits with 95% white noise confidence level at 0.0177 (0.0141) is shown in Fig. 9. The nearly ideal HD and autocorrelation results confirm the uniqueness of the SI PUF.

To further evaluate the randomness of PUF responses, NIST 800-22 randomness test suites are performed on 20480 bits collected from 10 chips. With the limited number of bits, a subset of 9 out of 15 from the 800-22 subtests is available.

TABLE III: Comparison with the state-of-the-art PUFs.

	DAC'22[6]	TCAD'20[8]	JSSC'20[7]	ISSCC'17[5]	This Work
Technology (nm)	55	180	180	180	40
Normalized Bitcell Area (F^2/bit)	1071.5	3730	354	553	982
Core Energy(pJ/bit)	0.91/0.27	0.163	465	0.0135	0.073/0.042
Temp. Range ($^\circ C$)	-40-125	-40-120	0-80	-40-120	-50-125
VDD Range (V)	0.96-1.44	1.2-1.8	1.0-1.8	0.8-1.8	0.75-1.5
Native Unstable Bits (%)	-	-	3.87	1.73	4.72/3.78
Native BER (%)	0.54/0.50	-	0.43	0.18	0.48/0.35
BER per 10 $^\circ C$(%e)	0.530/1.598	0.39	0.472	0.21	0.39/0.36
BER per 0.1 V (%e)	1.319/0.69	0.26	0.386	0.29	0.45/0.33
BER after Stabilization (%)	0.0061/0.0009	-	0.0034/0.0047	0.08	0.0053

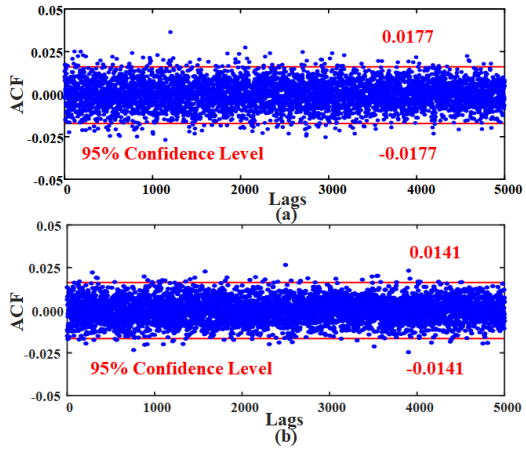


Fig. 9: Measured autocorrelation of bit cells: (a) high state; (b) low state.

From Table II, the PUF outputs from both high and low states pass all tests, showing high-quality randomness.

E. Performance Comparison

Finally, Table III summarizes the main metrics evaluated for the SI PUF solution and the most advanced analog PUFs. Compared to exiting work, Our PUF achieves only 0.0053% native BER and 0.073/0.042pJ per bit energy efficiency without the use of redundancy-based ideal masking and ECC. On the one hand, despite the excellent reliability of DAC'22 [6], it exhibits significantly higher power consumption compared to TCAD'20 [8], ISSCC'17 [5], and our work. On the other hand, ISSCC'17 [5] shows low power consumption but has the lowest reliability among the compared works. Thus, Table III clearly indicates that our SI PUF is the optimal choice when a balance between power consumption and reliability is desired.

V. Conclusion

This paper proposes a SI PUF with high reliability and low power consumption. The PUF chip fabricated in 40nm CMOS technology with a total area of 12580μm^2. Experimental results show that the PUF response passes the randomness test of NIST 800-22. The uniqueness and autocorrelation of high (low) state are 0.475 (0.501) and ±0.0177 (±0.0141), respectively. The PUF output can reduce the bit error rate to 0.0053% after using the BCS strategy, which is a very advantageous performance. In addition, the PUF raw response has a sensitivity of 0.39%/10°C (0.36%/10°C) and 0.45%/0.1V (0.33%/0.1 V). The characteristics of SI PUF, including its low BER and remarkable resilience to a wide array of temperature and voltage fluctuations, render it an optimal choice for lightweight security applications in the realm of IoT and other devices with limited resources.

References

[1] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*. IEEE, 2004, pp. 176–179.

[2] S. K. Mathew, S. K. Satpathy, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, G. K. Chen, R. J. Parker, R. K. Krishnamurthy, and V. De, "16.2 a 0.19 pj/b pvt-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm cmos," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 278–279.

[3] M.-D. Yu, D. M'Raihi, R. Sowell, and S. Devadas, "Lightweight and secure puf key storage using limits of machine learning," in *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*. Springer, 2011, pp. 358–373.

[4] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2008.

[5] K. Yang, Q. Dong, D. Blaauw, and D. Sylvester, "8.3 a 553f 2 2-transistor amplifier-based physically unclonable function (puf) with 1.67% native instability," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 146–147.

[6] J. Zhang, L. Ding, Z. Chen, W. Li, and G. Qu, "Da puf: dual-state analog puf," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 73–78.

[7] J. Lee, D. Lee, Y. Lee, and Y. Lee, "A 354f 2 leakage-based physically unclonable function with lossless stabilization through remapping for low-cost iot security," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 2, pp. 648–657, 2020.

[8] P. Gan, X. Zhao, and Y. Cao, "An all-mosfet voltage reference-based puf featuring low ber sensitivity to vt variations and 163 fj/bit in 180-nm cmos," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1172–1182, 2020.

DeDfTr: Defeating Design-for-Trust by Hardware Trojan-based Cross Computing Attacks

Xiaotong Cui, Hanzhou Lv, Xiaoke Du, and Kefei Cheng

***Abstract*—The outsourcing for fabrication introduces security threats, namely hardware Trojans (HTs). Many design-for-trust (DfTr) techniques have been proposed to address such threats. As a subset of DfTr techniques, the diversified redundancy establishes trustworthy systems with untrusted components by detecting and recovering from errors incurred by hardware Trojans (HTs). However, the technique is found to have vulnerabilities that can be exploited by attackers to conduct cross computing attacks (XCAs). An XCA can be launched by activating HTs with a special part referred to as "sequence extraction". This paper presents the detail of XCAs and proposes a checkpoint-based security scheme to defend against such attacks. The overhead of the proposed technique is evaluated in terms of timing, area and power overhead.**

***Index Terms*—hardware Trojan, sequence extraction, diversified redundancy, cross computing attacks.**

I. Introduction

A system-on-chip (SoC) integrates all necessary components that form a system [1]. Part of these components are digital, analog or mixed-signal intellectual property (IP) cores. The IP-based SoC design flow has become a global trend due to the benefits of performance, cost and efficiency brought by IP reuse.

It is observed in Fig. 1 that the IP-based SoC life cycle involves several stages[2]. First, the SoC integrator/designer creates design specifications where target functionalities and performance are defined. Then he/she identifies a list of IP blocks to implement the predefined specifications. These IPs could be divided into two categories according to their sources, i.e., IPs developed in-house and IPs purchased from third party IP (3PIP) vendors, which are further classified into soft, firm and hard IPs according to the phases they are applied. After integrating all these IPs, the SoC integrator sends the final GDSII format layout to foundry for fabrication.

However, the above SoC design flow incurs security issues in recent years, where 3PIP vendors may not be trusted and they could insert hardware Trojans (HTs) into their IPs [3], [4], [5], [6], [7]. Hardware Trojans are malicious design modifications, hidden backdoors and time bombs hidden in ICs. Once exploited by attackers, they will cause unpredictable consequences such as information leakage [8], [9], performance degradation [10], [11], logic errors [12], [13] and system crashes [14], [15]. What is worse, due to the stealth of HTs and the absence of "golden models" [16], detecting HTs within 3PIPs has always been a challenging task.

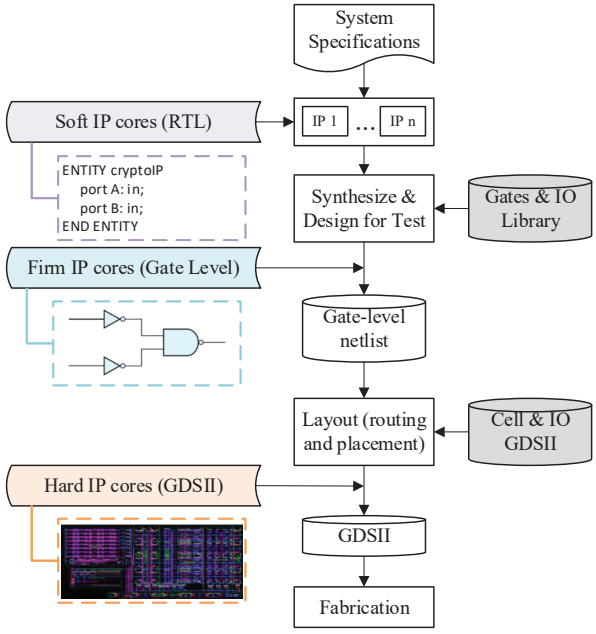


Fig. 1. The IP-based SoC design flow.

As a subset of design-for-trust techniques for HT detection in 3PIPs, run-time monitoring methods can achieve trustworthy computing with untrusted components when cooperated with recovery mechanisms. Compared to HT-detection methods working in the test phase, the run-time methods do not try to detect all possible Trojan circuits, but only focus on the consequences caused by them. Therefore, they are more suitable for HT detection in SoC with untrusted 3PIPs. Such kind of techniques are usually referred to as diversified redundancy mechanisms, where a diverse set of 3PIP vendors are provided. Operations are then bound to 3PIPs from different vendors according to predefined security rules. These security rules are defined to detect malicious outputs incurred by HTs.

A. Related Work

Related work on the diversified redundancy is presented as follows.

Based on the majority voting theory, Amin *et al.* proposed to use an odd number of IP cores from multiple vendors and then selected the same result of most IP cores as the correct result,

X. Cui, K. Cheng are with the School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing, 400065 China e-mail:xiaotong.sd@gmail.com

H. Lv is with the College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, 400065 China.

X. Du is with the China Automotive Engineering Research Institute Co., Ltd., Chongqing, 401122 China.

so that the system can resist against functional disruptions and availability attacks [17]. This technique requires at least 3 IP cores from different vendors to perform each operation simultaneously, thus resulting in a large computational overhead.

Rajendran *et al.* proposed a hardware Trojan detection technology at runtime based on the diversity of IP vendors and a redundant computing paradigm, where two task-to-IP core binding rules are defined to detect HT attacks and prevent collusion among IP cores from the same vendor [18], [19]. The disadvantage of this technology is that it only provides the detection of HTs such that not applicable to mission-critical systems with fault-tolerance requirements.

Cui *et al.* extended their work by adding a recovery phase with extra recovery rules. All these security rules were mapped to an integer linear programming (ILP) model with the goal of minimizing purchasing overhead of IP diversity [20]. Since the problem is NP-hard, the ILP scheme cannot obtain the optimal solution for large-scale circuits in an acceptable time. They further mapped the optimization problem to fast graph-theoretic models and proposed to locate and replace HT-infected IP cores [21].

To prevent accesses to confidential data from malicious IP cores, Gundabolu and Wang proposed a data protection technique based on access control mechanism. But it is unable to detect logic errors incurred by HTs [22].

Sayed-Ahmed *et al.* designed a subsystem to monitor a series of continuous states. Once behaviors of a IP core violate the security rules, then the network originated from the IP core will be disconnected to isolate it [23]. Obviously, this method cannot guarantee business continuity of the underlying system.

Chen *et al.* proposed an IP-level tag runtime Monitoring technology to prevent malicious IP from illegally accessing confidential data, but this technology is powerless to hardware Trojans that cause logic errors [24].

Although the diversified redundancy-based techniques are promising in achieving trustworthy computing, this paper shows that the diversified redundancy has vulnerabilities that could be exploited to conduct cross computing attacks(XCAs).

Upon an XCA, the underlying system could generate logic errors due to HTs implanted in IP cores from the same vendor distributed in different computings. To defeat the diversified redundancy, HTs within XCAs are equipped with a "sequence extraction" structure such that they can be activated under different inputs. To defend against XCAs, we propose a checkpoint-based scheme to enhance the security of the diversified redundancy. Then we evaluated the proposed technique in terms of timing, area and power overhead on a set of high-level synthesis benchmarks.

The main contributions of this paper are as follows:

- We explore and present vulnerabilities of the diversified redundancy for the first time.
- We show how such vulnerabilities can be exploited to conduct a XCA, where HTs with a special "sequence extraction" part are distributed in different computings.
- To enhance the security of the diversified redundancy, we propose a checkpoint-based scheme to defend against XCAs.

The rest of the paper is organized as follows: Section II introduces the diversified redundancy as a background. Section III is the threat model. Section IV explores vulnerabilities of the diversified redundancy mechanism and shows how HTs can be designed to defeat it. Section V presents possible security schemes that could mitigate impact of XCAs. Section VI and Section VII presents experimental analysis of the proposed defending technique with regard to timing, area and power consumption and shows how to mitigate the system overhead. Section VIII concludes the paper.

II. Review on the Diversified Redundancy

Untrusted 3PIP cores may contain HTs. As SoC designers/integrators, our goal is to use these untrusted components to build a trustworthy system, where the application is partitioned into a series of tasks and these tasks are mapped to 3PIP cores from untrustworthy vendors. Mapping mechanisms (including task scheduling and binding) need to be designed to ensure appropriate security and cost of the system. Once untrusted 3PIP cores are integrated into SoCs, HTs (if any) in these IP cores may destroy the functionality of the system during run time.

Although an activated HT causes logic errors as well, it cannot be modeled by traditional faults because its mechanism and impact are different from that of environment-induced soft/temperary or hard/permanent errors of hardware [25]. An environment-induced soft error of hardware, such as a single event upset that makes a bit reversed temporally, will disappear after a period of time. Hence the affected unit remains healthy [26]. The soft error of hardware can either be recovered by re-executing the task in the same hardware with time redundancy or using another hardware to execute the task in parallel with space redundancy [27], [28], [29]. An environment-induced hard error, such as a single event latch-up that makes a bit line of the hardware stuck-at-0, does not disappear [30], [31]. Hence the affected unit is deemed faulty thereafter. The hard error of hardware can only be recovered by using another hardware unit to execute the task in parallel with space redundancy [32], [33]. It is observed that with time and/or space redundancies, it is easy to recover environment-induced soft or hard errors.

A logic error caused by a HT, on the other hand, will persist if its trigger condition remains true and will disappear (deactivate) if its trigger condition becomes invalid [34], [35]. *This means that re-executing the task in the same hardware with time redundancy will not work if the HT's trigger condition holds true. Using another hardware to execute the task in parallel with space redundancy may not work either since these hardware units from the same vendor will contain the same HT.*

To address the above problem, the diversified redundancy mechanism is deployed such that the underlying systems could function properly and securely. It takes advantage of IP cores from diverse vendors. Although no "golden" IP core is available for comparison to ensure trust, the outputs of two IP cores of the same functionality from different vendors can be compared to discover abnormal behavior. This is based on

the assumption that "IP cores from different vendors tend to have different implementations. It is unlikely that they are both Trojan-infected. Even if they are, the chance that their Trojans are activated in the same way is assumed to be extremely rare" [17], [19], [21].

A typical diversified redundancy scheme consists of a detection phase and a recovery phase. In the detection phase, there is a normal computing and a recomputing. Normal computing is defined as a continuous execution of subtasks that will eventually achieve goals of an application. For example, to obtain the final results of $f = a_1x_1 + a_2x_2$, we need a series of computations. Recomputing is defined as a re-execution of corresponding subtasks of the normal computing on the same or different hardware units. For mission-critical applications, normal computing and recomputing are necessary parts of fault-tolerant schemes. Within a HT detection scenario, normal computing and recomputing are adapted according to security policies – outputs of normal computing and recomputing are compared to check whether there exist HTs. If the outputs are not consistent, the underlying system switches to the recovery phase to execute the recovery computing.

Taking the 5-node application in Fig. 2(a) as an example, the security policies are summarized by the following two operation-to-IP core diversity binding rules:

- *Rule 1:* The corresponding task nodes in normal computing, recomputing and recovery computing are bound to IP cores from different vendors.
- *Rule 2:* Task nodes that have parent-child relations, provide inputs to the same child nodes, or have closely related inputs[1] should be bound to IP cores from different vendors.

Rule 2 is used to prevents collusion between HTs from different IP cores of the same vendor. In the end, the resulting security solution is shown in Fig. 2(b), where task nodes in different colors means that they are bound to IP cores from different vendors. It is observed that 3 IP core vendors are required.

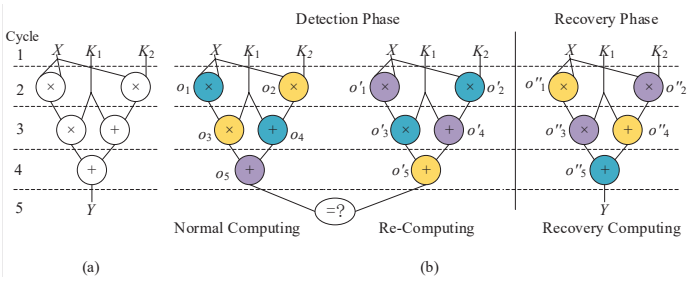


Fig. 2. The deployment of security polices. (a) The data flow graph (DFG) of the application; (b) The HT detection and recovery scheme according to security policies, where operations in different colors will be allocated to IP cores from different vendors.

Let us look into the example in Fig. 2 to check how security rules work. The function denoted by Fig. 2(a) is $f = K_1X^2 + K_2X + K_1$. Assume that $K_1 = 2$, $K_2 = 3$, $X = 5$ and there are no HTs in all IP cores, then the output of normal computing will be 62 as well as that of recomputing. In such a case, the recovery computing will not be executed. However, if a HT in IP core of o_3 is activated under input pattern (25,2) during function mode and it changes o_3's output to 30. Consequently, the output of normal computing is different from that of recomputing and hence the recovery computing will be executed. Values of all intermediate outputs could be find in Table I, where values in bold are wrong outputs due to the activation of the HT.

Although IP cores of o_2, o_1'' and o_3 may contain the same HT as they are in the same type and from the same vendor, HTs in IP cores of o_2 and o_1'' will not be activated due to different input conditions. The output of recovery computing will be correct as no HTs will be activated, on which the system depends to tell which computing from the detection phase is wrong through majority voting. This is how the diversified redundancy works.

TABLE I
THE INTERMEDIATE RESULTS WHEN ONE HT IS ACTIVATED.

Cycle	Normal computing					Recomputing				
	o_1	o_2	o_3	o_4	o_5	o_1'	o_2'	o_3'	o_4'	o_5'
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	25	15	N/A	N/A	N/A	25	15	N/A	N/A	N/A
3	25	15	**30**	17	N/A	25	15	50	17	N/A
4	25	15	**30**	17	**47**	25	15	50	17	67
5	$o_5 \neq o_5'$ followed by recovery computing									
	o_1''	o_2''	o_3''	o_4''	o_5''					
6	25	15	N/A	N/A	N/A					
7	25	15	50	17	N/A					
8	25	15	50	17	67					

III. THE THREAT MODEL

An SoC integrates all necessary IP cores to implement expected functions. These IP cores may be from untrusted third party IP vendors.

In the threat model, attackers from the untrusted vendors insert HTs into their IP cores. HTs can be designed in either combinational or sequential.

HTs hidden in 3PIPs works either in a individual way or in a collusion way [36]. In the former case, a HT is activated while the rest(if exist) remain silent. In the latter case, more than one HTs distributed in different IP instances are activated during runtime[37]. Once triggered, the payload of the HT incurs logic errors.

The attacker is assumed to have the full controllability of the IP development process within one IP vendor. That is, he can design HTs and insert them into any type of IP cores. By exploring and exploiting vulnerabilities of the diversified redundancy with the designed HTs, the goal of an attacker is to bypass the defense mechanism.

IV. USING XCA TO DEFEAT TRUST

In this section, we first present two vulnerabilities of the diversified redundancy, then we show how they can be exploited by attackers to conduct XCAs through HTs with a "sequence extraction" structure.

[1]Task nodes with close-related inputs typically have similar or the same inputs. In computing paradigms like FFT and FIR filters, they usually have symmetric structures.

A. Vulnerabilities of the Diversified Redundancy

From the basic assumption on which diversified redundancy works, we note that the security rules have a certain limitation, i.e., only *ONE* wrong computing can be tolerated at a time. For example, if IP cores of o_3 and o_3' in Fig. 2 contain the same HT and their HTs are activated by the same input pattern, then outputs of normal computing and recomputing will be the same wrong results, which hence are treated as correct results. This is one of sufficient conditions that violates security policies.

As an another example, if IP cores of o_3 and o_3' contain different HTs and they are activated, then outputs of normal computing and recomputing are different wrong results. This makes outputs of all 3 computings will be different such that we cannot tell which output is correct. This is another sufficient condition that violates security policies.

We summarize the above situations as two vulnerabilities of the diversified redundancy.

- *Vulnerability 1*: Take the wrong output as the correct one.
- *Vulnerability 2*: Cannot tell the correct result from three different outputs.

Thus, we can use $\sum_{i=1}^{3} A_i \leq 1$ to describe the situation where security rules are still effective, where A_i represents whether there are *activated* HTs in the i^{th} computing (of normal, re- and recovery computings). $A_i = 1$ if there exist, otherwise 0.

An attacker can defeat the diversified redundancy by launching a XCA, which will result in $\sum_{i=1}^{3} A_i > 1$. In an XCA, the attacker enables multiple HTs to be activated at a time across more than one computings. As it is difficult for 3PIPs from different vendors to collude, an attacker is assumed to have the capability of controlling 3PIPs from only one vendor. In the following, we will analyze how to achieve $\sum_{i=1}^{3} A_i > 1$ to defeat security policies.

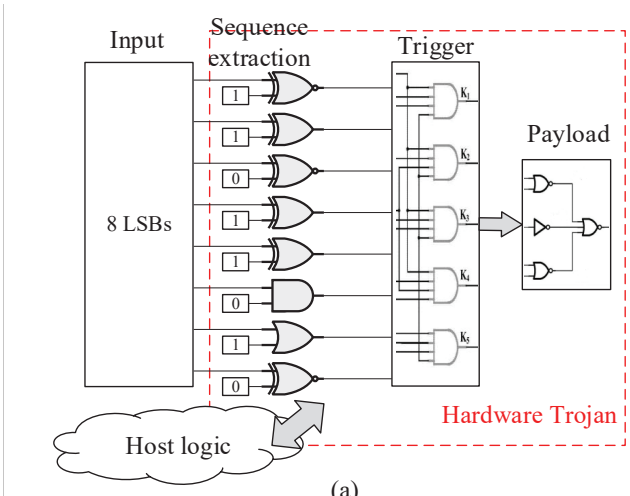
B. Defeating Trust within Detection Phase

To make $\sum_{i=1}^{3} A_i > 1$ within only detection phase indicates that $A_1 = 1$, $A_2 = 1$ and $A_3 = 0$. Take Fig. 2(b) as an example, all operations are allocated to IP cores of three vendors, which are colored in blue, purple and yellow, respectively. The attacker from one 3PIP vendor can hide a HT in IP cores that carry out multiplications, e.g., IP cores of o_1, o_2' and o_3'. These IP cores contain the same HT as they are multiple copies of the original IP core.

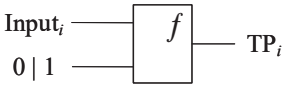
When HTs hidden in them are activated such that $A_1 = 1$ and $A_2 = 1$, there are two possible situations according to the outputs of normal computing and recomputing. One is that their outputs are the same wrong results, which exploits *Vulnerability 1* to break security policies. As IP cores of o_1, o_2' and o_3' receive inputs from and provide outputs to different operations, the other more common situation is that their outputs are different wrong results. In this case, it exploits *Vulnerability 2* to break security policies.

The key to exploit *Vulnerability 1* or *Vulnerability 2* is how to design HTs such that $A_1 = 1$ and $A_2 = 1$. It is noted the original security rules listed in Section II have already excluded situations that the same/close-related inputs of IP cores activate their same HTs. As a result, the same HTs within different IP cores need to be activated under different input patterns. We find that such a goal can be achieved by embedding the HT with a "sequence extraction" structure.

The sequence extraction makes HTs hidden in different IP cores receive the same triggering patterns. As an example, assume inputs of o_1 and o_3' are 677 (1010100101 in binary form) and 163 (10100011 in binary form), respectively and the triggering pattern of the HT is 11011010. One possible structure of the sequence extraction is shown in Fig. 3(a). It is observed that the sequence extracting structure takes 8 least significant bits (LSBs) of the primary input as input. It transfers both 10100101 and 10100011 to 11011010 such that HTs in both IP cores of o_1 and o_3' are activated. Actually, the sequence extracting structure can be designed to extract any triggering pattern bit (TP_i) from any arbitrary input bit ($Input_i$) by selecting proper gate f and its another input. Besides, the input of sequence extraction can be either from primary inputs or internal signals of host logic. Fig. 3(b) lists all possible combinations of input bit and gate f for a single triggering pattern bit TP_i. For example, to extract a 0 of triggering bit from any arbitrary input bit, an AND gate can be used by setting its another input to be 0.



(a)



TP_i \ Gate f \ $Input_i$		0	1
0	AND	0|1	0
	NAND	*none*	1
	OR	0	*none*
	NOR	1	0|1
	XOR	0	1
	XNOR	1	0
1	AND	*none*	1
	NAND	0|1	0
	OR	1	0|1
	NOR	0	*none*
	XOR	1	0
	XNOR	0	1

(b)

Fig. 3. The structure of sequence extraction within a HT. (a) A hardware Trojan with a "sequence extraction" structure (b) The design of i^{th} bit of triggering pattern.

With such a HT, it is necessary for the attacker to evaluate the probability of the HT being activated. The attacker would like the probability as small as possible to avoid accidental activation. But the probability is positively correlated with the number of different bits of two input patterns. For example, there are 2 different bits in 8 LSBs of 677 and 163 in binary form. For each different bit, both 0 and 1 need to be transferred to a certain *TP* bit by sequence extraction. As a result, the probability of the HT being activated is $\frac{2^2}{2^8}$.

There are 2 distinct ways to reduce the probability of HT being activated accidently: 1) Use more XOR or XNOR gates to implement the sequence extraction, such that fewer bits of input patterns that are used to extract the triggering pattern are different; 2) Embed the trigger of a HT with sequential logic, e.g., a clock-based counter.

C. Defeating Trust Across Recovery Phase

Defeating security policies across recovery phase indicates that $\sum_{i=1}^{2} A_i \geq 1$ in the detection phase and $A_3 = 1$ in the recovery phase.

1) $\sum_{i=1}^{2} A_i = 1$*:* Let us first consider $\sum_{i=1}^{2} A_i = 1$ and $A_3 = 1$. To be more specific here, we take $A_1 = 1$ and $A_3 = 1$, where there exist the same activated HTs in both normal computing and recovery computing.

Still use Fig. 2(b) as an illustrative example, the attacker from one 3PIP vendor hides a HT in IP cores that carry out additions, e.g., IP cores of o_4 and o_5''. When HTs within them are activated such that $A_1 = 1$ and $A_3 = 1$, there are two possible situations according to the outputs of normal computing and recovery computing.

One is that their outputs are the same wrong results. Under this situation, the outputs of normal computing and recomputing will be different first in the detection phase before the underlying system switches to the recovery phase. As we cannot tell which output is correct in the detection phase, we need to rely on the output of recovery computing to identify the correct one. However, as outputs of normal computing and recovery computing are the same wrong results, the underlying system will take the wrong result as the correct one, which exploits *Vulnerability 1* to break security policies.

The other situation is that the outputs of normal computing and recovery computing are different wrong results. Under this situation, the outputs of normal computing and recomputing are different as well in the detection phase. Then the underlying system switches to the recovery phase, where the recovery computing is counted on to identify the correct output. However, it will be found that outputs of all 3 computings will be different, which exploits *Vulnerability 2* to break security policies.

2) $\sum_{i=1}^{2} A_i = 2$*:* Now let us consider $\sum_{i=1}^{2} A_i = 2$, i.e., $A_1 = 1$, $A_2 = 1$ and $A_3 = 1$. As an example, the attacker hides HTs in IP cores of o_1, o_3', o_5'' in Fig. 2(b). It is observed that these IP cores implement different operations such as additions and multiplications. This is because *a HT can be implanted in different types of IP cores to carry out an XCA when embedded with a sequence extracting structure* shown in Fig. 3.

When HTs in these IP cores are activated, outputs of all 3 computings will be erroneous. There are 2 situations based on whether these erroneous outputs are the same. One is that outputs of normal computing and recomputing are the same, then it exploits *Vulnerability 1* as analyzed in Section IV-B. The other is that outputs of normal computing and recomputing are different, then the system switches to the recovery phase. If the output of the recovery computing is the same with that of one computing in the detection phase, then such an output will be treated as the correct one, which exploits *Vulnerability 1*. Otherwise, it exploits *Vulnerability 2* as outputs of all 3 computings are different.

In summary, HTs can be embedded with a sequence extracting structure similar to Fig. 3(a). Such kind of HTs could then be used to conduct an XCA by an attacker, which makes the diversified redundancy ineffective.

V. Defending Against XCA

In an XCA, the security policies are broken by HTs implanted in IP cores from the same vendor distributed in different computings. In this section, we show how to defend against such attacks.

A. Introducing new IP vendors?

As IP cores from the same vendor across different computings may be used to conduct XCAs, one intuitive defending method is to make these IP cores from different vendors. This strengthened policy need to be deployed based on the original security solution shown in Fig. 2(b) such that the set of IP cores from the same vendor and distributed in different computings are clear.

In XCAs analyzed above, HTs can be either implanted in IP cores in the same type or in different types. To avoid XCAs on the same type of IP cores, it is necessary to ensure any pair of operations in the same type across different computings are bound to IP cores from different vendors. To avoid XCAs on different types of IP cores, it is necessary to ensure any pair of operations across different computings are bound to IP cores from different vendors.

However, these strengthened binding rules incur large overhead in terms of the number of IP vendors. As operation-to-IP core diversity binding problems can be mapped to graph coloring problems [19][21], the comparison of preprocessing a graph between the strengthened binding rules and original security rules is illustrated on a 3-node application shown in Fig. 4, where a line between a pair of nodes means that they should be bound to IP cores from different vendors. It is observed that when strengthened security rules are applied, more operations cannot be bound to IP cores from the same vendor, which indicates that more IP vendors are required.

When strengthened security rules are applied to the example in Fig. 2, the resulting solution is shown in Fig. 5. Compared to 3 IP vendors required in Fig. 2(b) under original security rules, it is observed in Fig. 5(a) and (b) that 6 IP vendors are required to defend against XCAs utilizing the same types of IP cores and 9 IP vendors are required to defend against XCAs utilizing different types of IP cores, respectively.

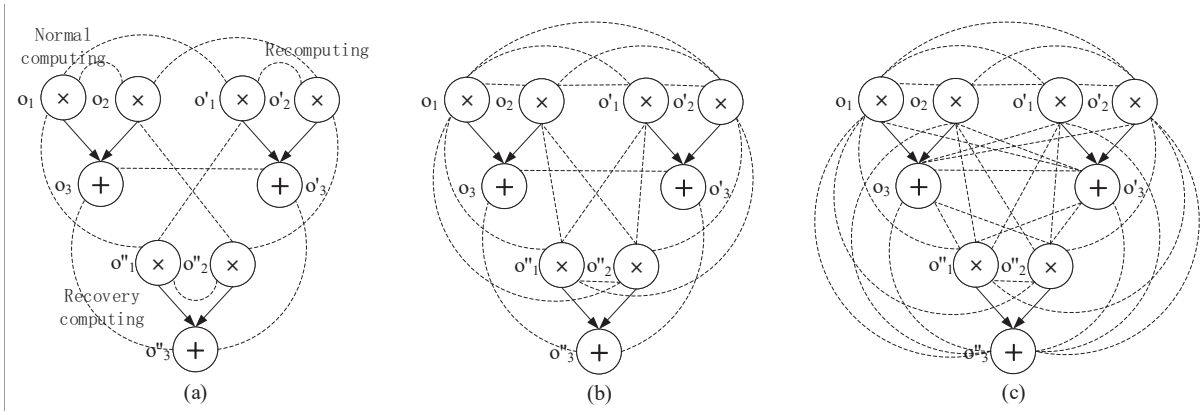


Fig. 4. A graph-coloring preprocess comparison on a 3-node application between original security rules and strengthened security rules of introducing new IP vendors. (a) Original security rules; (b) Strengthened security rules against XCAs on the same type of IP cores; (c) Strengthened security rules against XCAs on different types of IP cores. Note, a line between a pair of operations means that they should be bound to IP cores from different vendors.

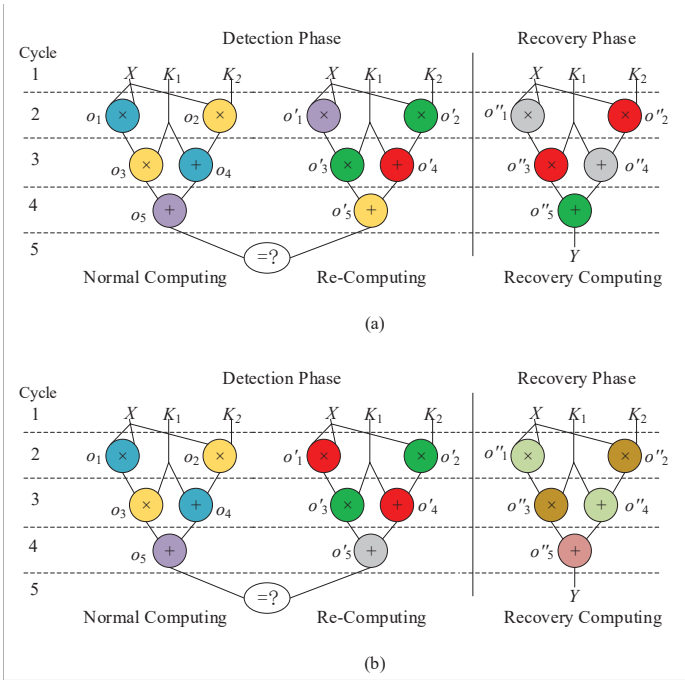


Fig. 5. The strengthened security rules by introducing new IP vendors. (a) Defending against XCAs on the same-type IP cores; (b) Defending against XCAs on different-type IP cores.

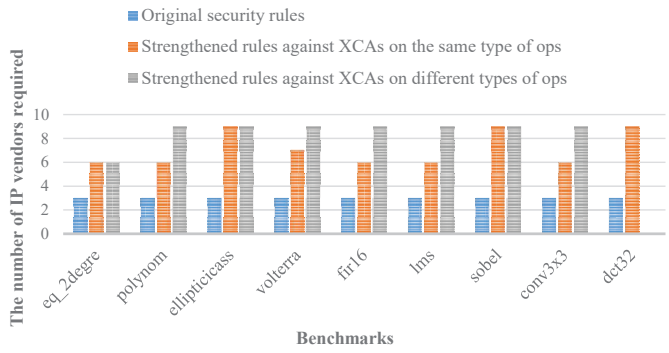


Fig. 6. The number of IP vendors required under strengthened security rules by introducing new IP vendors.

Finally, we evaluate the strengthened security rules in terms of introducing new IP vendors on a set of high-level synthesis benchmarks generated by Gaut[38]. The experimental results are shown in Fig. 6. We find that the deployment of strengthened security rules requires many more IP vendors compared to that of original security rules. Therefore, it may be impractical to deploy strengthened rules within an SoC due to the limited number of IP vendors in the market.

B. Inserting checking points?

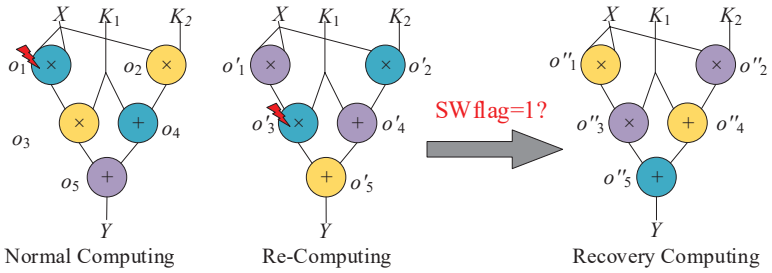
Let us consider the threat again, where HTs of IP cores from the same vendor are distributed and activated in different computings. Without loss of generality, we consider HTs in IP cores of o_1 and o'_3 are activated shown in Fig. 7. Then the outputs of normal computing and recomputing are both wrong results, which may exploit *Vulnerability 1* or *Vulnerability 2* such that we cannot identify which computing is correct even after recovery computing.

But when we look into the computing process, we find that by comparing corresponding outputs in 3 computings, wrong and correct outputs could be identified through majority voting. To achieve this, it is necessary to store intermediate results of the detection phase in memory. When the underlying system switches to the recovery phase, the intermediate results obtained in the detection phase are fetched out for comparison. That is, the intermediate results obtained in normal computing and recomputing would serve as checkpoints in recovery computing. We refer to such a mechanism as a checkpoint-based defending technique.

There are 2 points worth particular attention in the checkpoint-based security scheme. First, in the original diversified redundancy scheme, the system switches to the recovery phase when the outputs of normal computing and recomputing are not consistent. While in XCAs, the outputs of normal computing and recomputing could be the same wrong results, which prevents the system from switching to the recovery phase. Therefore, we need to adapt the original scheme to

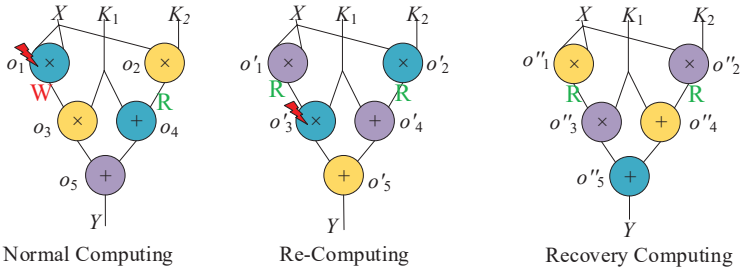
Steps in the Detection phase

(a)



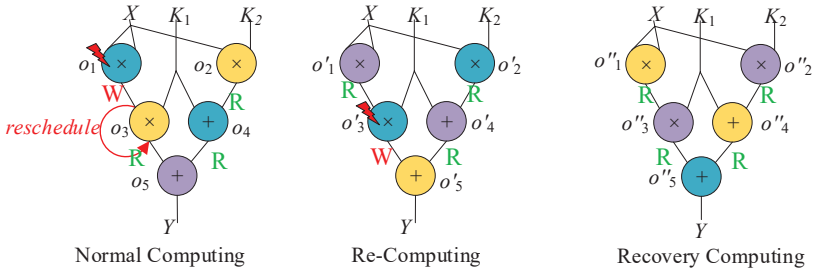
Steps: Initialize the flag of switching to recovery phase to 0 (SWflag=0). Compute corresponding outputs of each pair of operations and compare them before they are stored in memory. Once there exist inconsistences, set SWflag=1. When normal computing and recomputing are over, the system will switches to the recovery phase if SWflag=1.

Steps in the recovery phase

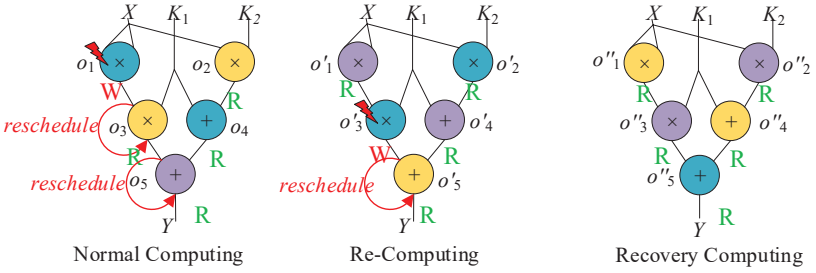


1st step: Compute outputs of o"$_1$ and o"$_2$ and compare them with previously stored intermediate results of <o_1, o'$_1$> <o_2, o'$_2$>, respectively. Then observed that the output of o'$_1$ is wrong based on majority voting. (Note that all previously stored intermediate results that rely on o'$_1$ will not be used in the following steps and all following operations must be rescheduled.)

(b)



2st step: Reschedule operation o_3 with correct inputs generated from previous step and compute outputs of o"$_3$ and o"$_4$. Similarly, compare corresponding outputs and find that the output of o'$_3$ is wrong based on majority voting.



3st step: Reschedule operation o_5 and o'$_5$ with correct inputs generated from previous step and compute outputs of o"$_5$. Similarly, compare corresponding outputs and find that all final outputs are the same correct results.

Fig. 7. Defending against XCA by a checkpoint-based security scheme. (a) Steps and hardware Trojan overview in the detection phase, where hardware Trojans in o_1 and o_3' will be activated during the computing process. (b) Steps in the recovery phase, where the errors incurred by hardware Trojans can be recovered.

defending against XCAs. The adapted solution is to compare intermediate results of normal computing and recomputing before they are stored in memory. Once a pair of intermediate results are found inconsistent, the system will switches to the recovery computing no matter the final outputs of normal computing and recomputing are consistent or not.

Second, in the comparing process, the rescheduling of computing resources is necessary to establish a complete, correct computing. This is because errors can be passed from parent nodes to child nodes. Therefore, erroneous outputs should be fixed once found in the comparing process of the recovery phase. After that, correct outputs are rescheduled as inputs of followed operations on the same type of IP cores to eliminate wrong datapaths.

Take Fig. 7 as an illustrative example, in the 1^{st} step of the recovery phase, the output of o_1 is marked as "W(rong)" based on majority voting. Then this erroneous output is replaced by correct outputs of o_1' or o_1'' and rescheduled on the IP core of o_3 such that the output of o_3 is correct. By this way, we can identify that the output of o_3' is wrong in the 2^{nd} cycle. Following the above steps, we will finally obtain the correct output. It should be noted that, once an erroneous intermediate output is identified, all operations that depend on this output must be rescheduled such that the corresponding previously stored results will not be used in the following comparison. It is observed that by identifying wrong outputs and reschedule followed operations, we can fix wrong datapaths and reconstruct a correct computing.

In the end, to make the checkpoint-based defending scheme more concrete and clear, we use Algorithm 1 to summarize its procedure.

Algorithm 1: Defending against XCA

Input: Normal computing *NC*, recomputing *RC* and recovery computing *RecC* with diversified redundancy; a set of operations with wrong outputs *WS*; the flag of switching to *RecC SWflag*; the array of correct output $CO[n]$, where n is the number of operations in *NC*.
Output: Correct output of the system O_f.

```
1  SWflag = 0 ;
2  WS = ∅;
3  while !NC.isEnd() do
4      foreach pair of operations o_i and o_i' in NC and RC in
       each clock cycle do
5          Calculate, compare and store their outputs;
6          if !isEqual(o_i,o_i') then
7              SWflag = 1;
8          end
9      end
10 end
11 if SWflag == 1 then
12     while !RecC.isEnd() do
13         foreach operation o_i'' in RecC do
14             Calculate output of o_i'';
15             if !({o_i,o_i'} ∩ WS).isEmpty() then
16                 Obtain outputs of {o_i,o_i'} ∩ WS by
                   rescheduling them with correct inputs in CO;
17             else
18                 Fetch outputs of {o_i,o_i'}\ ({o_i,o_i'} ∩ WS)
                   from memory;
19             end
20             Compare outputs of o_i, o_i' and o_i'';
21             if {o_i, o_i', o_i''}.MajorityNum()!=3 then
22                 foreach operation o_j that is no earlier than
                   {o_i, o_i', o_i''}.Minority() within the same
                   datapath do
23                     WS = WS ∪ {o_j};
24                 end
25             end
26             CO_i = {o_i, o_i', o_i''}.MajorityOutput();
27         end
28     end
29     O_f = CO_last;
30 else
31     O_f = NC.Output();
32 end
33 return O_f
```

VI. Experiments

In this section, we evaluate the checkpoint-based security scheme in terms of timing, area and power overhead.

A. Experimental Setup

The experiments are conducted on a set of high-level synthesis benchmarks generated by Gaut[38]. The operations used in these benchmarks include addition (add), subtraction (sub), comparison (comp), multiplication (mul), division (div), shift-right arithmetic (sra), the shift-left logical operator (sll) and absolute value (abs). Each operation is assigned an IP core in the experiments.

The timing, area and power setups of the IP cores that carry out these operations are shown in Table II.

TABLE II
THE TIMING, AREA AND POWER SETUP OF THE IP CORES USED IN THE BENCHMARKS

Unit type	$add/sub/comp$	$mul/div/sra$	sll	$comp_mux/abs$
timing/unit	1	4	3	2
area/unit	8	83	44	18
power/unit	1	8	5	2

B. Timing Overhead Evaluation

As each operation node is allocated with an IP core in the experimental setup, they can be scheduled as early as possible. In the checkpoint-based security scheme, corresponding outputs are compared and stored in memory in the detection phase. These comparisons can be done simultaneously with followed task nodes. As a result, the timing overhead in the detection phase of the proposed technique is the same or almost consistent with that of the original security scheme. When it goes to the recovery phase, the system relies on the comparison results to decide whether to reschedule task nodes. Thus there will be a comparison between each pair of parent-child operations.

Two related techniques are used for comparison with this paper. One is the original diversified redundancy scheme, where there exist a detection phase and a recovery phase. The original diversified redundancy scheme is referred to as *BaseLine* thereafter. The other provides the localization of the HT-infected IP core based on the *BaseLine* [21], which is

referred to as *Baseline+Loc* thereafter.

The experimental results are shown in Table III. The timing overhead of *BaseLine* is represented by clock cycles. Timing overhead of *BaseLine+Loc* and that of *This paper* are represented by the percentage of extra overhead compared to *BaseLine*. It is observed that the timing overhead of *Baseline+Loc* is the same with that of *BaseLine*. This is because the localization process is not a real-time required task such that it can be done simultaneously with followed task nodes. To defend against XCAs, the proposed checkpoint-based security technique incurs 47.5% extra timing overhead under the setup of Table II compared to *BaseLine* and *BaseLine+Loc* on average.

TABLE III
EXTRA TIMING OVERHEAD OF THE PROPOSED TECHNIQUE.

Benchmarks	*BaseLine*	*BaseLine+Loc*(%)	*This paper*(%)
polynom	17	0.0	47.1
ellipticicass	19	0.0	47.4
fir16	39	0.0	48.7
volterra	39	0.0	48.7
lms	59	0.0	49.2
sobel	29	0.0	48.3
*conv3*3*	31	0.0	48.4
dct32	107	0.0	49.5

C. Area Overhead Evaluation

In the *BaseLine+Loc* architecture, corresponding outputs of normal computing and recomputing are stored in memory in the detection phase. When the system switches to the recovery phase, previously stored results are fetched out to compare with that of recovery computing. The number of comparators required is given by $2 * N_c$ in [21], where N_c is the number of IP cores in one clock cycle. As comparators in different clock cycles can be multiplexed, we only need to care about the clock cycle with the maximum number of IP cores. In the proposed technique, although both detection phase and recovery phase require comparators for comparison, these comparison tasks are finished in different phases such that the comparators can be multiplexed, resulting in the same number of comparators compared to *BaseLine+Loc*. It should be noted that the proposed technique can locate the infected IP core earlier than *BaseLine+Loc*. This because the comparison tasks of the recovery phase in the proposed technique must be finished in time before followed executions of task nodes, which is also the source of extra timing overhead in Table III.

The experimental results of area overhead are shown in Table IV. The area overhead of *BaseLine* is represented by unit area. Area overhead of *BaseLine+Loc* and that of *This paper* are represented by the percentage of extra overhead compared to *BaseLine*. It is observed that the proposed technique has the same area overehead with that of *BaseLine+Loc*. On average, they incurs 19.1% extra area overhead compared to *BaseLine*. Besides, it is found that the extra overhead is related with the scale and unit types of benchmark circuits. When the scale is large, the comparators required will only account for a small part, e.g., *dct32* with 3.2% extra area overhead in Table IV. For benchmark circuits with small-area operators, the comparators will account for a large part of overhead, e.g., *ellipticicass* with 68% extra area overhead in Table IV.

TABLE IV
EXTRA AREA OVERHEAD OF THE PROPOSED TECHNIQUE.

Benchmarks	*BaseLine*	*BaseLine+Loc*(%)	*This paper*(%)
polynom	686	18.7	18.7
ellipticicass	704	68.2	68.2
fir16	4352	17.7	17.7
volterra	7412	11.2	11.2
lms	9017	13.5	13.5
sobel	13388	13.4	13.4
*conv3*3*	64952	6.4	6.4
dct32	58088	3.2	3.2

D. Power Overhead Evaluation

The power consumption of the proposed technique can be divided into several parts, including normal functions, comparing process and rescheduling process. The power consumption of former two parts is fixed while that of the latter one depends on the location of IP core where the hardware Trojan is activated in an XCA. This is because when an error of an operation is incurred by a HT, all followed operations that (directly/indirectly) depends on it will be rescheduled on their IP cores.

In this experiment, we evaluate the power consumption of the proposed technique on the occasion of an XCA taking place in the detection phase, e.g., Fig. 7(a). As we cannot know which IP core will malfunction in advance during run time, the power consumption of rescheduling process is calculated by its expectation.

Assume there are n task nodes in normal computing and the probability of occurring error due to HTs for each IP core is equal. Then the expectation of power consumption of rescheduling in normal computing can be calculated by

$$POW_{norm_resch} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j\in Decendant(i)} pow(j)$$

The other error of IP core k takes place in recomputing and depends on the first error of IP core i in normal computing. That is, these IP cores are from the same vendor. Assume there are m IP cores from $Vendor(i)$ in recomputing. Then the expectation of power consumption of rescheduling in recovery computing based on the first error can be calculated by

$$POW_{re_resch} = \frac{1}{m}\sum_{k=1}^{m}\sum_{h\in Decendant(k)} pow(h)$$

As a result, the overall power consumption of the rescheduling when two errors occur is calculated as follows.

$$POW_{all_resch} =$$

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j\in Decendant(i)}\left[pow(j) + \frac{1}{m}\sum_{k=1}^{m}\sum_{h\in Decendant(k)} pow(h)\right]$$

s.t. $Vendor(i) = Vendor(k)$

$i \in normal\ computing$

$k \in recomputing$

The experimental results of power overhead are shown in Table V. The power overhead of *BaseLine* is represented by unit power. Power overhead of *BaseLine+Loc* and that of *This paper* are represented by the percentage of extra overhead compared to *BaseLine*. It is observed that the proposed technique incurs extra 46.2% and 16.6% power overhead compared to *BaseLine* and *BaseLine+Loc*, respectively on average when an XCA happens.

TABLE V
EXTRA POWER OVERHEAD OF THE PROPOSED TECHNIQUE.

Benchmarks	*BaseLine*	*BaseLine+Loc*(%)	*This paper*(%)	
			One Error	Two Errors
polynom	70	14.3	30.9	34.3
ellipticicass	88	65.9	107.1	116.7
fir16	430	14.4	24.5	27.4
volterra	727	12.7	21.1	23.0
lms	892	14.8	31.8	45.8
sobel	1594	43.8	68.1	70.7
*conv3*3*	6409	15.2	21.6	21.8
dct32	5842	16.0	27.2	29.7

VII. MITIGATING SYSTEM OVERHEAD

The underlying mechanism incurs large overhead due to its redundant computings. As a result, it is necessary to optimize system overhead for practical applications. In this section, we propose an adaptive optimization algorithm by controlling the degree of multiplexing IP cores.

The proposed algorithm considers trade-offs between time (i.e., the schedule length) and area overhead. By relaxing the schedule length, more IP cores can be multiplexed such that the area overhead is reduced. On the contrary, more IP cores are necessary to meet the tight schedule length. By allocating each operation with an IP core, we will get a solution with the minimum timing overhead (referred to as solution *MinT*). By multiplexing all IP cores that are in the same type from the same vendor, we will get a solution with the minimum area overhead (referred to as solution *MinA*).

The comparison of the timing and area overhead of *MinT* and *MinA* are shown in Fig. 8. Take *ellipticicass* as an example, *MinA* is 5.5 times the timing overhead of *MinT* and *MinT* is 29 times the area overhead of *MinA*. It is observed in Fig. 8 that there exists a large solution space to trade off between area and timing overhead, indicating the security mechanism can be adapted to different system budgets.

The detail of the proposed algorithm is shown in Algorithm 2. It starts from a basic solution where each operation is allocated with an IP core such that it has the minimum time overhead. The algorithm then multiplexes IP cores gradually by increasing the time overhead until the area overhead satisfies the requirement before the time overhead exceeds the constraint. As a result, the algorithm can output different scheduling and binding solutions under various system budgets.

VIII. DISCUSSION AND CONCLUSION

As IP cores to be integrated may be HT-infected, security schemes deployed thereafter reach a high-level security since potential attacks from HT-infected IP cores are already taken into consideration. Therefore, these kind of design-for-trust techniques are promising in detecting or prevent HTs within 3PIPs. As a subset of design-for-trust techniques, the diversified redundancy is dedicated to building trustworthy systems with untrusted components. However, it is shown that the diversified redundancy is vulnerable to XCAs, where HTs are embedded with a sequence extracting structure. To defend against such attacks, we propose a checkpoint-based method that can reconstruct a correct computing from erroneous computings.

Algorithm 2: Trading off between area and time.

Input: System budgets in terms of time constraint T and area constraint A; The basic coloring solution generated by the adapted diversified redundancy with the number of types *VT* (Vendors $\times$ Types).
Output: The scheduling and binding solution under system budgets.

```
Allocate each task node with an IP core;
Calculate current time and area overhead t_c and a_c;
Initialize MultiplexFlag to be 0;
Initialize MultiplexValue of all operations to be 0;
Initialize MultiplexType[VT] to be 1;
if t_c > T then
    return failed;
else
    while sum(MultiplexType[])>=1 do
        foreach type of IP core h do
            Choose only one operation o_i that
            MultiplexValue[i] equal to 0 at each clock cycle;
            if more than one was chosen then
                MultiplexFlag +=1 ;
                Assign each MultiplexValue[i] =
                MultiplexFlag;
                Update a_c;
            else
                MultiplexType[h]=0;
                continue;
            end
        end
    end
    while a_c > A do
        foreach type of IP core h do
            Find the Set of minimum number of operations for
            each MultiplexValue;
            foreach operation o_i in the Set do
                Assign result = the maximum MultiplexValue[]
                of type h;
                Pre-schedule the Set and relevant operations to
                the set with maximum value set;
                if t_c > T then
                    return failed;
                else
                    Assign each MultiplexValue[i]= result;
                    Update a_c and t_c;
                end
            end
        end
    end
    return the solution;
end
```

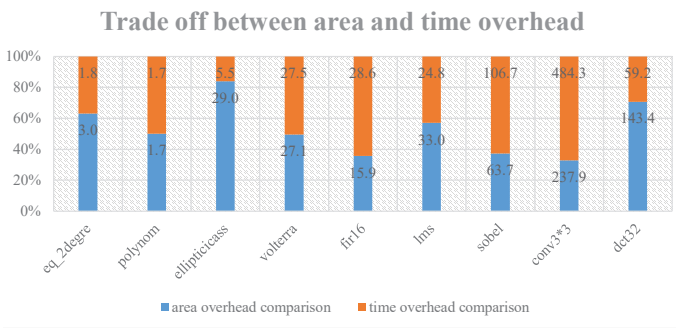


Fig. 8. Trade-off between the area and schedule length. Take *ellipticicass* as an example, the result 5.5 means that by multiplexing IP cores as many as possible (represented by design *A*), the schedule length of design *A* is 5.5 times longer than a design that schedules all operations as early as possible (represented by design *B*). Meanwhile, the area overhead of design *B* is 29 times bigger than design *A*.

REFERENCES

[1] K. Z. Azar, M. M. Hossain, A. Vafaei, H. Al Shaikh, N. N. Mondol, F. Rahman, M. Tehranipoor, and F. Farahmandi, "Fuzz, penetration, and ai testing for soc security verification: Challenges and solutions," *Cryptology ePrint Archive*, 2022.

[2] S. Bhunia and M. Tehranipoor, *Hardware Security: A Hands-on Learning Approach*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2018.

[3] R. JS, K. Chakraborty, and S. Roy, "Hardware trojan attacks in soc and noc," in *The Hardware Trojan War*. Springer, 2018, pp. 55–74.

[4] S. Sidhu, B. J. Mohd, and T. Hayajneh, "Hardware security in iot devices with emphasis on hardware trojans," *Journal of Sensor and Actuator Networks*, vol. 8, no. 3, p. 42, 2019.

[5] M. Shayan, K. Basu, and R. Karri, "Hardware trojans inspired ip watermarks," *IEEE Design & Test*, vol. 36, no. 6, pp. 72–79, 2019.

[6] A. P. D. Nath, S. Boddupalli, S. Bhunia, and S. Ray, "Resilient system-on-chip designs with noc fabrics," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2808–2823, 2020.

[7] M. Elshamy, G. Di Natale, A. Sayed, A. Pavlidis, M.-M. Louërat, H. Aboushady, and H.-G. Stratigopoulos, "Digital-to-analog hardware trojan attacks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[8] A. De, M. N. I. Khan, K. Nagarajan, and S. Ghosh, "Hartbleed: Using hardware trojans for data leakage exploits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 968–979, 2020.

[9] T. Perez and S. Pagliarini, "A side-channel hardware trojan in 65nm cmos with $2\mu W$ precision and multi-bit leakage capability," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2022, pp. 9–10.

[10] K. S. Subraman, A. Antonopoulos, A. A. Abotabl, A. Nosratinia, and Y. Makris, "Demonstrating and mitigating the risk of an fec-based hardware trojan in wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2720–2734, 2019.

[11] V. Y. Raparti and S. Pasricha, "Lightweight mitigation of hardware trojan attacks in noc-based manycore computing," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[12] H. Martin, L. Entrena, S. Dupuis, and G. Di Natale, "A novel use of approximate circuits to thwart hardware trojan insertion and provide obfuscation," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 41–42.

[13] V. R. Surabhi, P. Krishnamurthy, H. Amrouch, K. Basu, J. Henkel, R. Karri, and F. Khorrami, "Hardware trojan detection using controlled circuit aging," *IEEE Access*, vol. 8, pp. 77 415–77 434, 2020.

[14] A. Palumbo, L. Cassano, P. Reviriego, G. Bianchi, and M. Ottavi, "A lightweight security checking module to protect microprocessors against hardware trojan horses," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2021, pp. 1–6.

[15] S. Faezi, R. Yasaei, A. Barua, and M. A. Al Faruque, "Brain-inspired golden chip free hardware trojan detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2697–2708, 2021.

[16] R. Lu, H. Shen, Z. Feng, H. Li, W. Zhao, and X. Li, "Htdet: A clustering method using information entropy for hardware trojan detection," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 48–61, 2021.

[17] H. A. Amin, Y. Alkabani, and G. M. Selim, "System-level protection and hardware trojan detection using weighted voting," *Journal of Advanced Research*, vol. 5, no. 4, pp. 499 – 505, 2014, cyber Security.

[18] J. Rajendran, H. Zhang, O. Sinanoglu, and R. Karri, "High-level synthesis for security and trust," in *2013 IEEE 19th International On-Line Testing Symposium (IOLTS)*, July 2013, pp. 232–233.

[19] J. Rajendran, O. Sinanoglu, and R. Karri, "Building trustworthy systems using untrusted components: A high-level synthesis approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 9, pp. 2946–2959, Sep. 2016.

[20] X. Cui, K. Ma, L. Shi, and K. Wu, "High-level synthesis for run-time hardware trojan detection and recovery," in *Proceedings of the 51st Annual Design Automation Conference*, 2014, pp. 157:1–157:6.

[21] X. Cui, X. Zhang, H. Yan, L. Zhang, K. Cheng, Y. Wu, and K. Wu, "Towards building and optimizing trustworthy systems using untrusted components: A graph-theoretic perspective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021, doi:10.1109/TCAD.2021.3086765.

[22] S. Gundabolu and X. Wang, "On-chip data security against untrustworthy software and hardware ips in embedded systems," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 644–649.

[23] A. Sayed-Ahmed, J. Haj-Yahya, and A. Chattopadhyay, "Socint: Resilient system-on-chip via dynamic intrusion detection," in *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, 2019, pp. 359–364.

[24] K. Chen, O. Arias, X. Guo, Q. Deng, and Y. Jin, "Ip-tag: Tag-based runtime 3pip hardware trojan detection in soc platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 1, pp. 68–81, 2023.

[25] X. Cui, S. M. Saeed, A. Zulehner, R. Wille, K. Wu, R. Drechsler, and R. Karri, "On the difficulty of inserting trojans in reversible computing architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 4, pp. 960–972, 2020.

[26] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Transactions on nuclear Science*, vol. 50, no. 3, pp. 583–602, 2003.

[27] A. M. Cheng, G. Dai, P. K. Paluri, M. Ansari, D. Knape, and Y. Li, "Fault-tolerant regularity-based real-time virtual resources," in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2019, pp. 1–12.

[28] H. Liang, Z. Wang, R. Jiao, and Q. Zhu, "Leveraging weakly-hard constraints for improving system fault tolerance with functional and timing guarantees," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[29] M. Biasielli, C. Bolchini, L. Cassano, A. Mazzeo, and A. Miele, "Approximation-based fault tolerance in image processing applications," *IEEE Transactions on Emerging Topics in Computing*, 2021.

[30] J. Karp, M. J. Hart, P. Maillard, G. Hellings, and D. Linten, "Single-event latch-up: Increased sensitivity from planar to finfet," *IEEE Transactions on Nuclear Science*, vol. 65, no. 1, pp. 217–222, 2018.

[31] H. So, M. Didehban, A. Shrivastava, and K. Lee, "A software-level redundant multithreading for soft/hard error detection and recovery," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1559–1562.

[32] F. Mireshghallah, M. Bakhshalipour, M. Sadrosadati, and H. Sarbazi-Azad, "Energy-efficient permanent fault tolerance in hard real-time systems," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1539–1545, 2019.

[33] H. J. Mohammed, W. N. Flayyih, and F. Z. Rokhani, "Tolerating permanent faults in the input port of the network on chip router," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, p. 11, 2019.

[34] T. Hoque, X. Wang, A. Basak, R. Karam, and S. Bhunia, "Hardware trojan attacks in embedded memory," in *2018 IEEE 36th VLSI Test Symposium (VTS)*. IEEE, 2018, pp. 1–6.

[35] Z. Liu, J. Ye, X. Hu, H. Li, X. Li, and Y. Hu, "Sequence triggered hardware trojan in neural network accelerator," in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–6.

[36] J. Vosatka, "Introduction to hardware trojans," in *The Hardware Trojan War*. Springer, 2018, pp. 15–51.

[37] C. Liu and C. Yang, "Defense against hardware trojan collusion in mpsocs," in *Behavioral Synthesis for Hardware Security*. Springer, 2022, pp. 233–264.

[38] E. Martin, O. Sentieys, H. Dubois, and J.-L. Philippe, "Gaut: An architectural synthesis tool for dedicated signal processors," in *Design Automation Conference, 1993, with EURO-VHDL'93. Proceedings EURO-DAC'93., European*. IEEE, 1993, pp. 14–19.

航天嵌入式软件典型缺陷建模

左万娟[1,2] 王小丽[1,2] 黄晨[1,2] 董燕[1,2]
北京轩宇信息技术有限公司、北京控制工程研究所

摘要：以提高静态分析工具的软件缺陷自动化检测能力为目标，以工具查什么作为研究核心问题，从航天嵌入式软件海量原始缺陷数据入手，研究构建适用于工具自动化检测的航天嵌入式软件典型缺陷模式。从建模过程、模式要素、建模思路、模式实例等方面，对缺陷模式构建进行了阐述，提出了缺陷模式四要素，以及确定型、猜错型、提示型、拓展型等四种建模思路。实践数据表明，所构建的典型缺陷模式在提高静态分析工具的软件缺陷自动化检测能力方面，发挥了重要的作用，有效促进了航天软件质量的提升。

关键词：静态分析;缺陷模式;模式要素;建模思路;自动化检测

引言

随着软件定义航天器设计理念的提出、以及各种复杂控制及自主规划算法的大量应用，航天嵌入式软件的规模和复杂度不断攀升，软件潜在缺陷越来越多。对于追求零缺陷的航天嵌入式软件而言，综合运用各种测试技术手段、提升软件质量势在必行。

静态分析技术作为一种重要的软件检测手段，依托工具可以实现软件缺陷的静态自动化检测，无需执行程序，对检测环境要求低、覆盖率高，因而在工业届和学术界受到了更为广泛的关注[1]。

但是，根据文献[2]对静态工具的调研结果，66%的项目配置了静态检测工具，仅仅 37%的项目要求代码提交必须通过静态工具检测，相比于 100%的测试通过率要求，这一比例仍然很小。 无论是开源社区还是工业界开发者，都对静态检测工具缺乏信心。很多缺陷分析工具只关注那些普遍存在于各软件中的基础性、常见性缺陷，例如空指针引用、声明的变量未被使用等等，以确保工具可以被广泛使用[3]。总之，静态分析工具的检测能力和检测精度抑制了用户的使用欲望。

为提高静态分析工具的检测能力和检测精度，业内针对静态分析算法开展了很多研究[4-6]。但是，算法层面上的改进难以很好地在工程中实施，基于算法改进的精度提升技术已面临一定瓶颈[7]。

基于缺陷模式的静态软件缺陷检测技术是近年来发展较快的一种静态源代码分析技术，可以实现软件缺陷自动和半自动检测。此类缺陷检测工具由于效率高和使用简单等特点在高可信软件测试中得到了广泛应用。基于缺陷的测试技术主要包括两大方面：1)软件的缺陷模式；2)面向该模式的自动测试方法。此类测试首先提出缺陷模式，然后通过检测算法进行检测，如果检测算法是完全的，则能够从软件中排除该类缺陷，缺陷模式越多，检测能力就越强[8]。

目前，针对基于缺陷模式的软件测试的研究，更多地集中在自动测试方法研究[8-11]、以及基于特定工具的缺陷模式自定制方面的研究[12]，针对缺陷模式构建的研究很少。

与静态分析算法研究不同，缺陷模式的构建，需要长期、丰富的软件测试领域工程实践经验、以及大量的工程实际缺陷数据支持，目前业内针对适用于工具自动化检测的典型缺陷模式的公开发表的研究成果很少。

本文针对航天嵌入式软件典型缺陷模式构建开展研究，充分发挥软件测试领域工程实践经验，通过研究航天嵌入式软件实际缺陷数据，构建适用于自动化检测的、免配置的典型缺

陷模式，实现航天嵌入式软件典型缺陷的工具静态自动化检测，从而与其它软件测试技术手段形成良好的互补，实现提升航天软件质量的终极目标。

1. 概述

所谓缺陷模式是指程序中经常出现的错误或者导致程序出现异常的缺陷，这些缺陷有同样的产生原因，同样的表示形式，按照这些缺陷发生的规律能够提取出相同的缺陷模式[13]。

针对航天嵌入式软件典型缺陷建模的研究，目标是解决如下两方面核心问题：

a. 通过构建缺陷模式，解决工具查什么的问题，从而通过静态分析工具的研发实现软件典型缺陷自动化检测。

b. 通过缺陷报告的精心设计，指导软件编码及测试人员确认缺陷。

2. 建模过程

从大量的航天嵌入式软件工程实际缺陷数据出发，通过对原始缺陷数据的预处理及特征提取等，构建典型缺陷模式，形成缺陷模式集，提供给静态分析工具实现缺陷自动化检测。

下图用于说明航天嵌入式软件缺陷建模的过程：

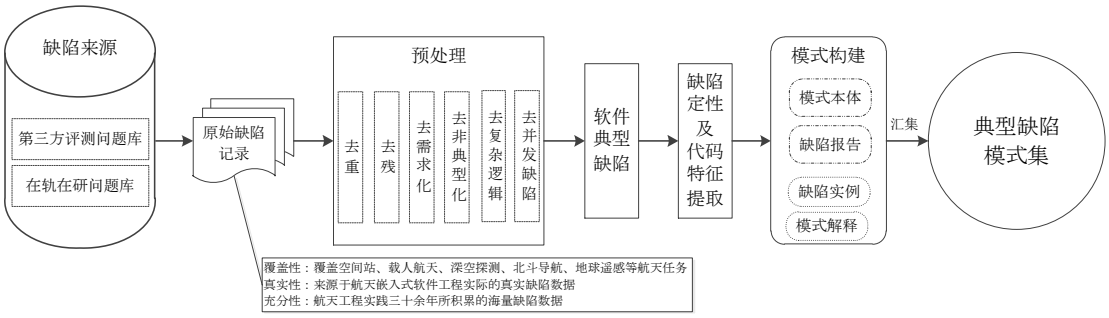


图 1 典型缺陷模式构建过程框图

1）缺陷数据来源

作为研究出发点的全部原始缺陷数据，均来源于航天器软件第三方评测缺陷数据库、以及航天器软件在轨/在研质量问题数据库，全部是软件工程实践中的真实缺陷数据，覆盖了航天各大重点工程项目。

2）缺陷数据预处理

针对原始缺陷记录，需要进行如下几方面的预处理：

a. 去重。来源于工程实践的原始缺陷数据中，不可避免地存在一些类似、或重复的缺陷数据，需要予以剔除。

b. 去残。原始缺陷数据中，势必存在一些信息残缺的缺陷数据，不具有研究价值，需要予以剔除。

c. 去需求化。需求相关的软件缺陷，工具是不具备检测能力的，因此，无论是以修改文档方式处理的软件缺陷、还是因程序实现不满足需求而做出修改程序处理的软件缺陷，均要予以剔除。

d. 去非典型化。有些软件缺陷，不具有典型意义，研究价值小，需要予以剔除。

e. 去复杂逻辑。针对复杂代码逻辑，工具分析能力是非常受限的，因此，需要剔除复杂逻辑缺陷数据。

f. 去并发缺陷。并发缺陷检测，有其独立研究方向，不在本研究范畴，因此，需要予以剔除。

3）缺陷定性及代码特征提取

经过预处理得到软件典型缺陷后，需要分析并确定缺陷的核心属性，比如，if 判断缺陷、强制类型转换缺陷、赋值缺陷等等，以便进行典型缺陷及缺陷模式的归类。

然后，在缺陷定性的基础之上，根据缺陷描述提取或构造特征代码，实现模式构建。

3. 模式要素

通过研究所构建的缺陷模式，既要给工具研发人员提供检测信息及说明，又要给软件相关人员提供缺陷确认所需要的指导信息。缺陷模式的构成要素如表 1 所示：

表 1 缺陷模式要素表

模式要素	重要性	主要服务对象	其它说明
模式本体	核心要素	工具研发人员	解决工具查什么的核心问题
缺陷报告	核心要素	编码人员、测试人员	指导软件相关人员确认缺陷
缺陷实例	辅助要素	工具研发人员、编码人员、测试人员	辅助工具研发人员理解缺陷模式 辅助软件相关人员（主要是编码及测试人员）理解缺陷报告
模式解释	辅助要素	工具研发人员、编码人员、测试人员	

其中，核心要素是必备要素，辅助要素是可选要素。

辅助要素通常仅适用于复杂模式，这些模式，仅靠核心要素，可能无法使相关人员予以充分理解。一方面，绝大多数静态分析工具研发人员不具备嵌入式软件开发及测试经验，不了解嵌入式软件典型功能及设计，不熟悉嵌入式软件常用处理器及接口芯片，这些都阻碍了其对缺陷模式的理解，从而影响了缺陷自动化检测工具的研发。另一方面，软件相关人员能力及经验参差不齐，也需要一些辅助信息，帮助其理解缺陷模式，从而根据缺陷报告做出正确的缺陷认定。

下面分别予以说明：

3.1 模式本体

模式本体是典型缺陷的精炼表达形式，解决工具查什么的核心问题。通常以特征代码、自然语言的形式予以表述。模式本体要力求精准，否则会直接影响自动化检测的精度。

3.2 缺陷报告

缺陷报告是工具自动化检测的输出结果。缺陷报告的优劣，直接影响到软件相关人员对工具报告缺陷的最终确认结果。

好的缺陷报告设计，是一个好的缺陷模式的关键环节。好的缺陷报告，应满足以下几个条件：第一，应短小精悍、切忌长篇大论；第二，应包含明确的缺陷定性信息、且与缺陷代码紧密相关；第三，应包含充分的指导性信息，能够帮助、引导软件相关人员迅速做出正确的判断。

3.3 缺陷实例

适用于复杂模式，辅助理解模式。通常是实际缺陷代码片段的拷贝。

3.4 模式解释

适用于复杂模式，对缺陷模式的机理作出必要的解释，辅助理解模式。

4. 建模思路及实例

4.1 建模思路

构建缺陷模式时，不仅要关注无条件成立的确定型缺陷，也要关注有条件成立的非确定型缺陷，同时，针对一些需要通过进一步分析方可确认的缺陷苗头也不能放过，尽可能地让工具给出充分数量的缺陷报告，供软件相关人员进行确认。对建模思路具体说明如下：

a. 确定型建模

有一些特征代码形式，无论出现在何种场合，都一定属于软件缺陷。确定型建模，就是要尽可能地挖掘此类具有确定缺陷属性的特征代码形式，构建缺陷模式。

b. 猜错型建模

有一些特征代码形式，如果出现在特定场合，就一定属于软件缺陷。猜错型建模，就是要尽可能地挖掘此类具有特定场合下的确定缺陷属性的特征代码形式，构建缺陷模式。

c. 提示型建模

有一些特征代码形式，在满足某些综合条件下，可能属于软件缺陷。提示型建模，就是针对此类情况构建缺陷模式，提示软件相关人员结合代码实际应用场景做重点确认。

d. 拓展型建模

有些缺陷模式，本身具备一定的可拓展性，可以通过拓展得到多个缺陷模式。

拓展型建模的目的，是为了获得多个具有类似缺陷属性的缺陷模式，而不是将拓展而得的类似缺陷属性全部集中在一个缺陷模式中，这一方面可以丰富完善缺陷模式、从而提升缺陷检测能力，另一方面通过细化缺陷模式、从一定程度上避免工具针对某特定缺陷模式误报过多而导致的缺陷漏判。

误报是任何静态分析工具都无法避免的，误报越多、使用者对工具的信任度就越低，将缺陷模式最小化、避免缺陷模式集成化，是规避单个缺陷模式存在过多误报的一种有效方式。

4.2 模式实例

针对每种建模思路，分别列举了若干缺陷模式实例，如表 2 所示。每个实例中，仅给出两个核心要素的说明。

表 2 模式实例

建模思路	模式本体	缺陷报告
确定型建模	软件调用了 abs()库函数，但是未 include stdlib.h 头文件	abs()库函数头文件缺失，应 include stdlib.h 头文件（注：本条主要针对 DSP、SPARC 类芯片）
	x=(float/double)(y/z); 注：x 为 float/double 型变量，y、z 为整型（有符号、或无符号的 char、int、short 型）变量/立即数	运算精度受损，应先进行强制类型转换、再进行除法运算
	for(k=M;k>=0;k--) 其中，k 为无符号整型变量，M 为正整数	无符号整型变量 k 递减后始终大于等于 0，形成死循环
猜错型建模	x = (x&0x0F)+1;	如果高低 4 位分别表示不同物理意义，则运算结果破坏了高 4 位
	x=y; 其中，x 为 64 位无符号整型变量，y 为 double 型变量。	如果 double 型变量 y 取值为负，则赋值给 64 位无符号整型变量将导致赋值结果为 0
	if((x>=A)&&(y<=B)) 其中，A<B	如果 x、y 分别为起始、结束地址，则判断设计不完善，无法规避起始地址

		x>B、及结束地址 y<A 的非法情况，可能导致非法地址访问
提示型建模	中断函数中存在多处 return 设计	中断中多处 return，请确认退中断前的必需处理是否完备
	变量赋值后未使用再次赋值	变量赋值后未使用再次赋值，请确认是否存在变量使用上的疏漏
拓展型建模	变量只有读操作	变量操作不完整，仅读未写
	变量只有写操作	变量操作不完整，仅写未读
	y=x<<M; 其中，x、y 为 unsigned char 型变量，M>=8	移位溢出，导致赋值后的结果为 0
	y=x>>M; 其中，x、y 为 unsigned char 型变量，M>=8	移位溢出，导致赋值后的结果为 0

4.3 小结

4.3.1 缺陷模式特点分析

与建模思路相对应，可将缺陷模式分为三类：确定型模式、猜错型模式、提示型模式。

需要说明的是，拓展型建模是基于确定型/猜错型/提示型模式基础上拓展得到新的、类似的缺陷模式，它不会产生新的模式类别。

各类模式的特点见表 3 说明：

表 3 模式特点

模式类别	缺陷概率	建模难度	软件相关人员确认缺陷所需要的投入	缺陷是否成立
确定型	最大	最小	最小	缺陷无条件成立
猜错型	居中	居中	居中	缺陷有条件成立
提示型	最小	最大	最大	缺陷可能成立 通常需要综合考虑多种因素

确定型模式的缺陷报告，只要确认不是工具误报，基本就可以确认为缺陷。

猜错型模式的缺陷报告，需要确认其所在的场景，只要场景（条件）符合，基本就可以确认为缺陷。

提示型模式的缺陷报告，通常需要分析的相关内容会比较多、比较离散，因此需要的分析投入最大。

4.3.2 缺陷报告设计准则

除了给出充分的指导性信息之外，每类缺陷模式的缺陷报告设计，也需要与模式类别相适应，具体见表 4 说明：

表 4 缺陷报告设计准则

模式类别	缺陷报告设计准则
确定型	给出明确的缺陷说明
猜错型	给出缺陷条件及缺陷说明
提示型	提示潜在缺陷

5. 技术优势分析

本项研究，通过构建典型缺陷模式，与自动检测算法研究形成有效互补，共同提升静态分析工具的缺陷检测能力。

相比业内其他研究成果，本项研究在如下几个方面具有较为独特的技术优势：

1）基础数据来源于航天工程实践三十余年所积累的海量缺陷数据，在此基础之上所构建的缺陷模式，在航天工程上具有良好的适用性。

2）提出了缺陷模式四要素，为工具研发人员及软件相关人员提供了充分的缺陷模式说明信息，有效促进相关人员对缺陷模式的理解，为工具研发及工程应用提供了充分的依据和指导信息。

3）建模思路有所创新，除了基于代码自身机理的确定型建模思路之外，增加了对应用场景、物理意义等综合背景因素的考量，形成了猜错型建模、提示型建模思路，使缺陷模式更加丰富、覆盖面更广，工程实用意义更为突出。

4）缺陷模式最小化及拓展型建模思路的提出，对抑制工具误报率、提升工具可信度、促进工具的推广应用，起到一定的积极作用。

5）将嵌入式软件所特有的芯片操作特征纳入研究范围，构建了相应的缺陷模式，进一步拓展了工具自动化检测的范畴。

6）检测过程完全自动化，无需人工配置。

6. 应用情况

依托于全自主研发的静态分析工具 SpecChecker，对近两年的缺陷模式的应用情况进行了统计，统计数据见表 5：

表 5 近两年缺陷模式应用情况统计数据

序号	缺陷归类	检出缺陷情况	缺陷模式归类		
			确定型	猜错型	提示型
1	if 判断条件设计不当	33	12	17	4
2	变量使用缺陷	30	9	12	9
3	赋值缺陷	18	13	4	1
4	浮点数判 0 缺陷	16	16	0	0
5	强制类型转换错误	16	11	3	2
6	Hook 函数的实参冲突	7	7	0	0
7	switch 设计缺陷	7	7	0	0
8	数组使用缺陷	6	5	1	0
9	传参缺陷	6	5	1	0
10	While 死循环	5	2	3	0
11	按位与/或表达式/变量	4	0	2	2
12	函数返回值未处理	4	4	0	0
13	for 循环设计缺陷	4	3	1	0
14	运算符相关缺陷	4	2	1	1
15	逻辑与/或立即数	4	1	1	2
16	abs()头文件缺失	3	3	0	0
17	立即数“f”后缀检查	3	0	3	0
18	精度受损	3	3	0	0
19	运算溢出	2	1	0	1
20	连续开关中断操作失效	2	2	0	0
总计	-	177	-		

数据来源说明：

1)统计数据取自中国空间技术研究院软件产保中心第三方评测数据库。

2)应用项目均为航天器嵌入式软件第三方评测项目、且提交第三方评测之前均已完成开发方自测试。

3)仅统计已做修改程序处理的缺陷数。

通过应用数据可以看出，缺陷模式的应用，对提升航天软件质量起到了很好的辅助作用。

结束语

实践表明，基于缺陷模式的源代码静态分析，在工程上具有很高的实用价值。适用于自动化检测的缺陷模式的构建，虽然不易，但其本质上与软件缺陷是类似的，具有无限可扩展性，因此，构建缺陷模式是一项可持续性的研究工作。同时，通过工程应用，缺陷模式的精准度也可以得到进一步改进，从一定程度上解决静态分析工具的误报和漏报问题。

目前，原始缺陷记录的预处理、缺陷特征提取和模式构建的整个过程都是人工完成的，效率偏低，未来考虑引入数据挖掘技术，实现典型缺陷自动提取、加速缺陷建模，通过持续构建适用于自动化检测的缺陷模式，不断提升静态分析工具的缺陷自动化检测能力和检测精度，从而与其它测试手段形成良好的互补，共同促进软件质量的提升。

参考文献

[1] 李广威，袁挺， 李炼. 开源 C/C++静态软件缺陷检测工具实证研究[J]. 软件学报， 2022, 33(6): 2061–2081.http://www.jos.org.cn/1000-9825/6569.htm. [doi: 10.13328/j.cnki.jos.006569]

Li GW, Yuan T, Li L. Study of state-of-the-art open-source C/C++ static analysis tools[J]. Ruan Jian Xue Bao/Journal of Software, 2022, 33(6): 2061−2081 (in Chinese). http://www.jos.org.cn/1000-9825/6569.htm. [doi: 10.13328/j.cnki.jos.006569]

[2] Vassallo C, Panichella S, Palomba F, Proksch S, Gall HC, Zaidman A. How developers engage with static analysis tools in different contexts[J]. Empirical Software Engineering, 2020, 25(2): 1419−1457.

[3] 梁广泰，孟娜，李进辉等. 一个可半自动化扩展的静态代码缺陷分析工具[J]. 计算机学报，2011,34(6):1114-1125.[doi: 10.3724/SP.J.1016.2011.01114]

Liang GT, Meng N, Li JH, etc. A semi-automatic extensible static defect analysis tool[J]. Chinese Journal of Computers, 2011,34(6):1114-1125 (in Chinese).[doi: 10.3724/SP.J.1016.2011.01114]

[4] 董玉坤，金大海，宫云战，邢颖．基于区域内存模型的 C 程序静态分析[J]．软件学报，2014,25(2)：357-372．http：//www.jos.org.cn/1000-9825/4532.htm. [doi: 10.13328/j.cnki.jos.004532]

Dong YK，Jin DH，Gong YZ，Xing Y．Static analysis of C programs via region-based memory model[J]．Ruan Jian Xue Bao/Journal of Software，2014，25(2)：357-372(in Chinese)．http：//www.jos.org.cn/1000-9825/4532.htm. [doi: 10.13328/j.cnki.jos.004532]

[5] 王眉林，张旖旎，李明月等. 基于符号执行提高缺陷分析的准确性研究[J]. 北京理工大学学报，2020,40(4):382-385. [doi: 10.15918/j.tbit 1001-0645.2017.383]

WANG ML，ZHANG YN，LING MY. etc. Improving the accuracy of static defect analysis based on symbolic execution[J]. Transactions of Beijing Institute of Technology，2020,40(4):382-385(in Chinese). [doi: 10.15918/j.tbit 1001-0645.2017.383]

[6] 王雅文，姚欣洪，宫云战，杨朝红. 一种基于代码静态分析的缓冲区溢出检测算法[J]. 计算机研究与发展，2012,49(4):839-845.

WANG YW，YAO XH，GONG YZ，YANG ZH. A method of buffer overflow detection based on static code analysis. Journal of Computer Research and Development, 2012,49(4):839-845(in Chinese).

[7] 张丽. 代码静态分析工具的能力评估与集成优化技术研究[D]. 长沙：国防科学技术大学，2017.

ZHANG L. Research on evaluation and integrated optimization of code static analysis tools[D]. ChangSha：National University of Defense Technology,2017
[8] 王雅文. 基于缺陷模式的软件测试技术研究[D]. 北京：北京邮电大学，2009.
WANG YW. Research on software testing technology based on defect pattern[D]. BeiJing：Beijing University of Posts and Telecommunications，2009.
[9] 张仕金. 基于 Cppcheck 软件缺陷模式的研究与定位[D]. 重庆：重庆大学，2014
ZHANG SJ. Software defect pattern analysis and location based on Cppcheck[D]. CHONGQING：Chongqing University，2014
[10] 高猛，滕俊元，陈睿等. 航天器软件典型缺陷模式的自动检测技术[J]. 空间控制技术与应用，2019,45(5):72-78. [doi: 10.3969/j.issn.1674-1579.2019.05.011]
GAO M，TENG JY，CHEN R，etc. Automatic detection technology of typical defect pattern of spacecraft software[J]. Aerospace Control and Application，2019,45(5):72-78(in Chinese). [doi: 10.3969/j.issn.1674-1579.2019.05.011]
[11] 杨朝红，宫云战，肖庆等. 基于缺陷模式的软件测试中的区间运算应用[J]. 计算机辅助设计与图形学学报，2008,20(12):1630-1635.
YANG ZH，GONG YZ，XIAO Q，etc. The application of interval computation in software testing based on defect patterns[J]. Journal of Computer-aided Design & Computer Graphics，2008,20(12):1630-1635(in Chinese).
[12] 王德朋，王前，薛伟. 一种软件缺陷模式自定制方法[J]. 软件，2013,34(12):68-72. [doi: 10.3969/j.issn.1003-6970.2013.12.016]
WANG DP，WANG Q，XUE W. A method for software defect patterns customization[J]. Software，2013,34(12):68-72(in Chinese). [doi: 10.3969/j.issn.1003-6970.2013.12.016]
[13] Rutar N， Almazan C， Foster J. S. A comparison of bug finding tools for java[C]. In proceedings of the 15th IEEE International Symposium on Software Reliability Engineering， Saint-Malo， France， 2004 : 269 - 270.

作者简介：[通信作者]左万娟（1971-），女，汉族，硕士，研究员，主要研究方向为嵌入式软件测试；王小丽（1982-），女，硕士，高级工程师，主要研究方向为嵌入式软件测试；黄晨（1982-），女，博士，高级工程师，主要研究方向为嵌入式软件测试；董燕（1972-），女，硕士，研究员，主要研究方向为嵌入式软件测试.

一种基于权重范围变换的忆阻神经网络容错方法

钟志伟[①] 尤志强*[①] 刘 鹏[②] 杨 静[①] 邝继顺[①]

[①](湖南大学信息科学与工程学院 长沙 410082)

[②](广东工业大学计算机学院 广州 510060)

摘 要：针对带故障忆阻神经网络（MNN）存在训练时间长、硬件冗余、结构复杂等问题，本文提出了一种基于权重范围变换的 MNN 容错方法（WRT），在无需硬件冗余、神经元排列、重训练的前提下，只修改权重变化范围，就能提高带故障 MNN 的识别精度。由于 MNN 的层数多、参数量大，获得最优精度时权重变化范围的搜索空间异常巨大。为了降低搜索的时间复杂度，本文提出了一种利用权重量化和树形结构的权重变化范围搜索方法（WQTS），时间复杂度由穷举法的 $2N_L$ 阶降到只有对数阶，并没有增加空间复杂度。实验结果验证了本文提出方法的有效性，与前人工作相比较，在前馈神经网络（FF NN）和卷积神经网络（CNN）中 WRT 均取得最好或相当的识别精度。对于 4 层 FF NN 中存在 20%故障时，平均恢复精度提高了 218.9%；在 CNN 存在 10%的故障时，平均恢复精度约提高了 3 倍。WQTS 在搜索时间和识别精度之间取得很好的均衡。由 WQTS 获得的识别精度与时间复杂度很高的穷举法识别精度相当。与本文设计的逐层搜索与梯度下降方法相比，WQTS 获得的识别精度都有提高。

关键词：忆阻器；神经网络；权重范围变换；固定故障；权重量化

Fault Tolerant of Memristor Neural Networks with Weight Range Transformation

ZHONG Zhiwei[①] YOU Zhiqiang*[①] LIU Peng[②] YANG Jing[①] KUANG Jishun[①]

[①](*College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China*)

[②](*School of Computer, Guangdong University of Technology, Guangzhou 510006, China*)

Abstract: Aiming at the problems of long training time, hardware redundancy, and complex structure of Memristive Neural Networks (MNNs) with faults, this paper proposes a fault-tolerant MNN method based on Weight Range Transformation (WRT). Without the need for hardware redundancy, neuron arrangement, or retraining, this method improves the recognition accuracy of faulty MNNs simply by modifying the range of weight changes. Due to the many layers and parameters in MNN, the search space for the weight variation range is huge when obtaining optimal accuracy. In order to reduce the time complexity of the search, this paper proposes a Weight Quantization Transformation Search (WQTS) method using weight quantization and tree structure. The time complexity is reduced from the $2N_L$ order of the ergodic method to only the logarithmic order without increasing space complexity. The experimental results verify the effectiveness of the proposed method. Compared with previous work, WRT has achieved the best or comparable recognition accuracy in Feedforward Neural Networks (FF NNs) and convolutional neural networks (CNNs). When there are 20% faults in the 4-layer FF NNs, the average recovery accuracy increases by 218.9%. When there are 10% faults in the CNN, the average recovery accuracy increases by about 3 times. WQTS achieves a good balance between search time and recognition accuracy. The recognition accuracy obtained from WQTS is comparable to the high time complexity of the ergodic method. Compared with the layer-by-layer search and gradient-descent methods designed in this article, WQTS achieves improved recognition accuracy.

Key words: Memristor; Neural network; Weight range transformation; Stuck-at-Fault; Weight quantization.

*通信作者：尤志强 you@hnu.edu.cn

基金项目：国家自然科学基金（62074055、62174038），广东省基础与应用基础研究基金（2021A1515011962）

Foundation Item：The National Natural Science Foundation of China (62074055, 62174038), Guangdong Basic and Applied Basic Research Foundation (2021A1515011962)

1 引言

卷积神经网络（CNN）在推理过程中执行许多矩阵矢量乘法操作，因而需要内存与计算单元的交互，消耗大量的能量。忆阻器具有面积小、功耗低、电阻可变等特性，非常适合存储神经网络中的权重。忆阻交叉阵列（MCA）由忆阻器组成可用于存储 CNN 中的权重矩阵并执行矩阵矢量乘法运算，降低了能耗和加速了运算。通常忆阻神经网络（MNN）由 MCA 和实现池化等操作的 CMOS 器件组成。MNN 常应用于目标检测[1]、目标识别[6][7]、图像边缘检测[2]、疾病预测[3]、Q 学习的路径搜索[4]、路径规划[5]等应用。然而，由于忆阻器的制造工艺尚不成熟，MCA 中存在许多故障，导致用 MNN 实现 CNN 的精度较低[8]。因此，如何提高带故障 MNN 的识别精度是研究的关键问题。

忆阻器通过两端施加电压脉冲的方式改变阻值。在理想情况下，忆阻器的阻值可以在最大值 R_{off} 和最小值 R_{on} 之间连续变化。MCA 中存在多种故障，常见的故障为固定故障（SAF）和变化的故障（Variance）[9][10]。带有 SAF 忆阻器的阻值无法改变，将始终保持在高电阻状态（HRS R_{off} ）或低电阻状态（LRS R_{on} ），分别对应固定为 0（SA0）和固定为 1（SA1）的故障[11]。SAF 会改变 MCA 中的权重分布，使得 CNN 的权重值映射到 MNN 中存在巨大的差异，导致精度显著下降。当前忆阻器的变化故障主要表现为器件到器件和周期到周期的故障。变化的故障可以使用忆阻器写后验证的方式[14]克服。因此，本文主要研究带 SAF 的 MNN 容错方法。

减少 SAF 影响的方法主要集中在硬件冗余[15][16]、神经元重排列[17][18][23]、权重范围变换[17][19]、数字偏置补偿[20][21]、MNN 重训练[20][22]这几个方面，如图 1 所示。

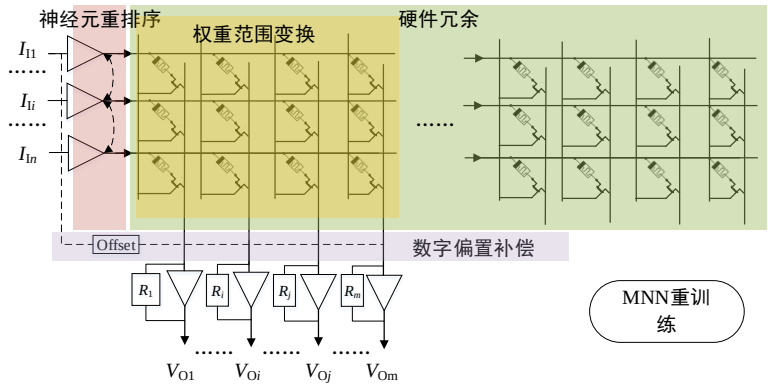


图 1　忆阻神经网络故障容错方法

硬件冗余通常在原有 MCA 的基础上添加冗余行、冗余列或冗余 MCA。Xia 等采用冗余 MCA 和冗余列的方法替换掉故障部件以恢复识别精度[16]。Liu 等采用冗余列的方法，标记出重要的权重，直接替换包含最重要权重差异的小部分列[25]。Chen 等提出了标识关键故障的方法，在 MCA 中添加少数几个备用列，以替换包含关键故障的列[15]。Xu 等采用整数线性规划的方式在包含冗余行列在内的 MCA 中选取权重矩阵大小的行和列来存储权重[24]。硬件冗余适合于故障率低、故障集中的情况，然而当故障率高或故障分散时，需要添加大量的硬件。

神经元排列通常是对 MCA 的行列进行重排序，尽可能将权重矩阵中大的权重分配给 SA1 故障的忆阻器，将小的权重分配给 SA0 故障的忆阻器，以最大程度地掩盖忆阻器硬件中的故障[18][41]。Zhang 等采用行翻转变换和神经元排列[17][19]，以减少实际权重矩阵与故障 MCA 表示的权重矩阵之间的差异。Zhang 等采用激活值方差和权重误差的神经元排序技术[29]，把方差大的权重放入到故障少的列中。神经元排列无需额外的忆阻器，然而当 MCA 中的故障位置改变时，需要重新进行排列。

权重范围变换是修改 CNN 权重映射到 MNN 中的最大值和最小值，以减小故障忆阻器表示的权重偏离真实值的差异。文献[17][19]以分类精度阈值 T_r 为权重范围变换判别依据，当 MNN 精度小于阈值时进入行翻转变换或神经元排列操作。文献[18][41]假设权重列的分布相同，采用每列神经元排列和 MCA 分块的方式，使得单 MCA 块的权重范围缩小，以减少实际权重矩阵与故障 MCA 表示的权重矩阵之间的差异。权重范围变换减少了排序和重训练等的时间，无需额外忆阻器的支持。然而，故障下的 MNN 精度提高的不明显，通常需要与其它的方法结合起来。

数字偏置补偿是指在不改变 MCA 的情况下，在 MCA 周围添加校正量来提高精度，如图 1 中的“Offset”模块，

“Offset”模块可以是软件实现也可以硬件实现。Lin 等通过输入缓冲中添加输入的偏置补偿[20]，减少重映射后仍然存在偏差的问题。Meng 等同时考虑输入和输出的影响，把数字偏置补偿引入到输入信号和 MCA 输出的之间[21]。Yeo 等在 MCA 和 ADC 中间添加电阻，使得即使存在故障时，电压值依然可以保持在合适的范围内[26]。同时，由于每列添加的电阻不一样，可以适当修正权重，故障的影响被进一步降低。Zhang 等针对带故障 MCA 表示权重矩阵的差异，在偏置中加入补偿[29]以减少故障的影响。数字偏置补偿不改变 MCA 的结构，然而需要额外的软件或者硬件支持。

重训练是对映射到 MNN 上的权重进行重新训练，调整没有故障位置点的值，以提高网络精度。Yeo 等首先获取故障权重位置，通过修改反向传播方式对 MNN 全部参数进行训练[26]。Liu 等提取 CNN 权重的重要性，只训练 MNN 对应的重要权重[25]。Lin 等在训练权重的同时训练了输入偏置补偿[20]。重训练 MNN 权重在恢复精度上相对较高，然而当 MCA 改变时，需要再次重训练。

综上所述，MNN 的容错带来了各种额外的开销，软硬件开销从低到高大致为权重范围变换、神经元排序、重新训练、MCA 分块、数字补偿、硬件冗余等。本文主要针对规模化生产过程中需要结构简单、精度高的网络、提出了权重范围变换的 MNN 容错方法（WRT）。本文的主要创新点如下。

1）本文提出了一种只采用权重范围变换的高精度 MNN 容错方法，根据神经网络量化理论和权值误差损失方法，在无需增加硬件冗余、神经元排列、重训练等的前提下，通过权重范围变换的方式提高带故障 MNN 的精度。

2）为了降低获得最优精度时权重范围变换的搜索空间，本文提出了一种利用权重量化和树形结构的权重变化范围搜索方法（WQTS），采用蒙特卡洛法提高本文方法的鲁棒性。通过把连续的权重分布量化为离散的权重分布，建立树形结构的搜索方式，从低比特量化的低精度到高比特量化的高精度转换，以此显著地降低时间复杂度。

3）本文通过不同的实验来验证提出方法的有效性。本文与文献在 3 层、4 层前馈神经网络（FF NN）、卷积神经网络（CNN），以及与其它搜索算法的比较有所提高。此外，本文还分析了在不同卷积核大小，使用或不使用硬件冗余，预先是否获取 MCA 故障位置等情况下，本文方法的可行性。

本文的主要结构安排，第 2 节为带故障 MNN 和训练后量化的相关介绍，介绍带故障 MNN 的权重表示、权重映射、损失函数以及训练后量化的相关知识，第 3 节为介绍基于 WQTS 的忆阻网络 MNN 容错方法以及对应的网络结构，第 4 节为实验部分，介绍本文方法的相关实验。最后对本文的工作进行总结与展望。

2 带故障 MNN 和训练后量化的相关介绍

2.1 忆阻交叉阵列

在 MCA 中，忆阻器位于字线和位线的交叉点上，通过字线电压脉冲对忆阻器进行编程，以改变忆阻器的阻值。在 CNN 中输入矢量被转换为输入电压并被施加到 MCA 的字线上，通过读取沿位线的电流或者电压来获得模拟矩阵矢量乘法的结果，如图 2 所示。忆阻器位线上的电压计算公式可以表示为：

$$V_{Oj}=\left(\sum_{i=1}^{n}\frac{V_{Ii}}{R_{ij}}\right)*R_j=\left(\sum_{i=1}^{n}G_{ij}V_{Ii}\right)*R_j \tag{1}$$

其中，V_{Ii}表示输入电压，V_{Oj}代表输出电压，R_{ij}、G_{ij}是位于第 i 行第 j 列忆阻器的电阻值和电导值，R_j为偏置电阻。V_{j+}和 V_{j-}为晶体管栅极电压，控制忆阻器导通和截止。由于神经网络的权重值可正可负，而电导值非负，通常权重矩阵由正负 MCA 组成[23]。每个权重由两个忆阻器表示，

$$w=w_+-w_- \tag{2}$$

其中，w_+，w_-分别是正负 MCA 中对应忆阻器表示的权重值。在通常情况下，当权重 w 为正数时，w_+为需要表示的权重值，w_-为 0。而当权重 w 为负数时，w_+为 0，w_-为需要表示权重值的绝对值。

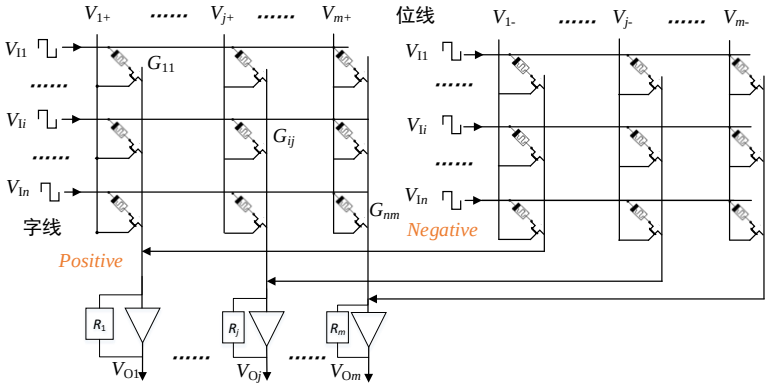


图 2　正负MCA结构

2.2 MNN权重映射

将训练好的权重映射到 MNN 的示意图如图 3 所示。图 3 的上半部分为 CNN 架构，通常由三种类型的层组成：卷积层、池化层和全连接层。图 3 的下半部分为对应的 MNN 结构，它也包括卷积、池化和全连接层。利用 MCA 的存储和计算来实现卷积层和全连接层中的矩阵矢量乘法[27]，激活函数和池化通常由 CMOS 电路实现[28]。卷积层的输入数据是输入特征图，通过数模转换器（DAC）转换为模拟信号。单个卷积核权重相应的电导值被存储在正负 MCA 的一列中。MCA 中的多个列形成 MNN 卷积层的权重矩阵，通过卷积计算在 MCA 中实现矩阵矢量乘法。在全连接层中，输入数据由上一层神经元的输出组成，与卷积层一样，输入数据换成模拟信号的矢量。全连接层的计算也是通过 MCA 中的权重矩阵矢量乘法来实现。

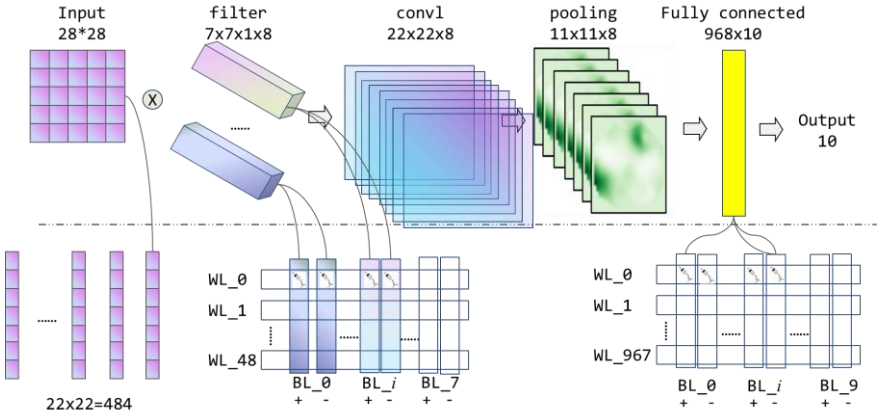


图 3　CNN权重映射到MNN中

如图 3 所示，以 MNIST 数据集为例，输入图像的大小为 28×28，经过 8 个 7×7 的卷积核，输出为 22×22×8 的特征图。然后，经过 2×2 的池化得到 11×11×8 的特征图。把该特征图进行拉平，其全连接层为 968×10。对应 MCA 中的操作为图像中 22×22 个 7×7 的图像块输入到 49×8×2（正负权重）的 MCA 中，得到 22×22×8 的模型信号值，经过 CMOS 池化电路后输出 11×11×8 的模拟信号。最后，这些模拟信号被拉平后输入到 968×10×2 的 MCA 中得到分类结果。

2.3 带故障MCA的权重表示

SAF 一般由忆阻器制造缺陷导致，通过读写操作检测。图 4 是包含 SAF 的忆阻器 MCA，每个忆阻器可能是无故障的、SA1 故障或者 SA0 故障三种状态。G_{11}^{-}，G_{13}^{+} 位置上的忆阻器分别带有 SA1、SA0 故障。本文假设在权重映射时，已经获得 MCA 的故障的类型和位置。

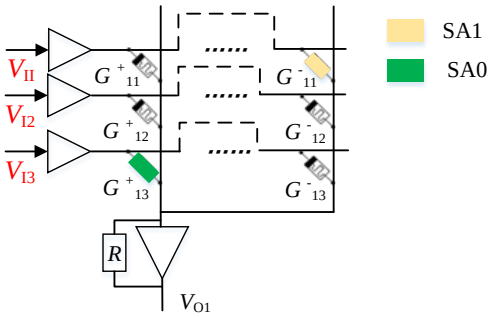


图 4　带故障的MCA

假如训练好后，CNN 权重为 w_i，若存储该权重的忆阻器中有故障，则写入 MCA 中的权重值与 w_i 可能不同，记为 w_{c0}^{eff}。根据公式 2，$w_{c0}^{eff}=w_{c+}^{eff}-w_{c-}^{eff}$，$w_{c+}^{eff}$ 和 w_{c-}^{eff} 分别为对应忆阻器表示的正负权重值。

$$w_{c+}^{eff}=\begin{cases} w_{min+}, & \text{if } m \text{ SA0} \\ w_{c+}, & \text{if } m \text{ nodefective} \\ w_{max+}, & \text{if } m \text{ SA1} \end{cases} \tag{3}$$

$$w_{c-}^{eff}=\begin{cases} w_{min-}, & \text{if } m \text{ SA0} \\ w_{c-}, & \text{if } m \text{ nodefective} \\ w_{max-}, & \text{if } m \text{ SA1} \end{cases} \tag{4}$$

其中，w_{min+}和 w_{max+}分别为正 MCA 中忆阻器表示的最小权重和最大权重，w_{min-}和 w_{max-}分别为负 MCA 中忆阻器表示的最小权重和最大权重。在 CNN 权重中存在很多 0 值，通常 $w_{min+}=w_{min-}=0$。w_{i+}和 w_{i-}分别为需要写入 MCA 的正负权重值。m 为忆阻器的状态，分别为无故障的、SA1 故障或者 SA0 故障。

w_{c0}^{eff} 的取值范围为$[-w_{max-}, w_{max+}]$，在故障位置已知，无硬件冗余的情况下 w_{c+}^{eff} 和 w_{c-}^{eff} 组合有 9 种情况[23]，记为 R=1。本文假设故障位置已知，根据这 9 种情况对忆阻器进行编程，以实现 CNN 权重映射到 MNN 中。类似地，在 1 倍硬件冗余的情况下，也就是一个权重用 4 个忆阻器表示，两个忆阻器位于正 MCA，两个位于负 MCA，记为 R=2，以此容忍更多的故障。

2.4 权重损失

权重损失值 $cost(w_{c0}^{eff}, w_i)$ 是 CNN 离线的权重值 w_i 与 MNN 表示的权重值 w_{c0}^{eff} 的差值的绝对值平均数[19]。

$$cost\left(w_{c0}^{eff}, w_c\right)=\frac{\sum\left|w_{c0}^{eff}-w_c\right|}{N_w} \tag{5}$$

其中 N_w 为 w_i 的权重个数。权重损失值越低，表示写入 MNN 的权重值越接近理想权重值，MNN 的识别精度越高。为了降低 $cost(w_{c0}^{eff}, w_i)$ 的值，许多工作尝试改变 w_{c0}^{eff} 的值，如 MCA 排序[29]，拆分为多个 MCA[18] [41]，复杂的控制单元[30][41]等。主要是由于当 w_i 改变时，神经网络的精度可能降低，但是 w_i 的修改是在 CNN 离线网络上，相比 w_{c0}^{eff} 更加容易，因此本文主要研究 w_i 改变下的忆阻器容错。

CNN 权重值的范围等于$[-w_{max-}, w_{max+}]$。本文所采用的值范围为$[-w_{max-}^{r}, w_{max+}^{r}]$，其中 $-w_{max-}^{r}\geq -w_{max-}$ 和 $w_{max+}^{r}\leq w_{max+}$。通过将 W 中的所有权重设置为大于（小于）CNN 的权重以降低损失函数，提高故障下的精度[19]，其表达式如下所示：

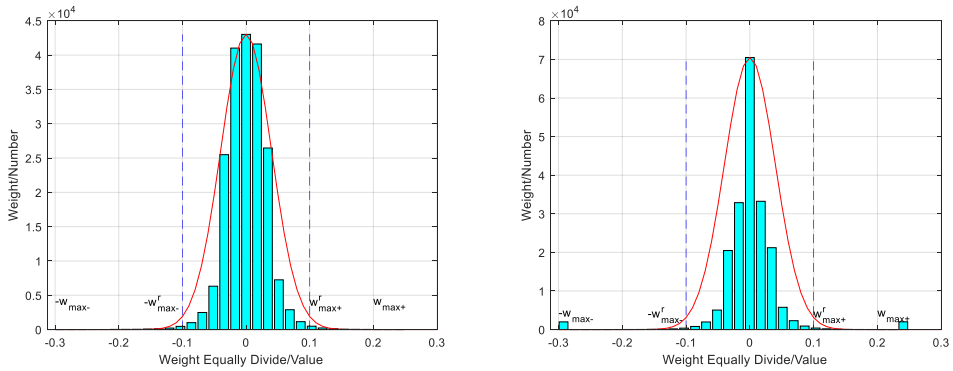
$$w_{c1}^{eff}=\begin{cases}-w^{r}_{max-}, & \text{if} \quad w_c \le -w^{r}_{max-} \\ w_c, \ \text{if} \ w_k \in \left[-w^{r}_{max-}, w^{r}_{max+}\right] \\ w^{r}_{max+}, & \text{if} \quad w_c \ge w^{r}_{max+}\end{cases} \tag{6}$$

因此，其损失函数比公式 5 要小，然而如何快速的确定-w^{r}_{max-}和 w^{r}_{max+}是本文研究的关键问题。

2.5 训练后量化

深度学习量化方法主要分为感知量化训练[31][32]（Quantization-Aware Training）和训练后量化[33][34]（post-training quantization）。感知量化训练是一种参数在线重训练方法，可以在改变全精度参数模型的情况下，以最小化量化误差进行训练使之逼近全精度模型的方式。而训练后量化是在不改变全精度参数模型的情况下，采用标定、修剪、微调等方式。训练后量化处理起来简单些，不需要重训练，节省了微调的时间，可实现快速的部署。

由于 CNN 每层的权重分布通常为正态分布或近似正态分布，呈现钟形分布，越远离中心值的权重越稀少。例如，图 5 为 784×256 全连接层的权重，呈现近似正态分布。如果进行 n 位均匀量化，也就是 $m=2^n$等分量化。图 5 (a)为 w_i该层权重的 m=32 等分量化。w_{max+}和 w_{max-}是权重最大值和最小值，w_{max+}=0.2491，-w_{max-}=-0.3007。假设 w^{r}_{max+}=0.1，-w^{r}_{max-}=-0.1 则[-w^{r}_{max-}, w^{r}_{max+}]中的权重数依然占总数的 99.2%，并且 CNN 权重等分区间数由 32 减少为 11。图 5 (b)为 20%故障下 w^{eff}_{co} 该层权重的 M=32 等分量化。存在故障时会使得 w^{r}_{max-}和 w^{r}_{max+}附近的值增加，中间存在权重很少的情况。假如把这些权重往中心移动，这样故障下的精度可能提升。因此，我们可以考虑使用[-w^{r}_{max-}, w^{r}_{max+}]区域来进行容错。



(a) (b)

图 5 训练后量化方法。（a）为无故障下的权重分布，（b）为 20%故障下的权重分布。

3 基于 WQTS 的 MNN 容错方法

本文在故障位置已知的前提下，提出了一种只修改权重取值范围以提高识别精度的 MNN 容错方法，不需要对 MNN 进行重新训练，根据故障忆阻器的权重损失、CNN 中的权重分布和 MNN 权重映射获取。本文主要是对公式 6 进行优化求解，使得在 CNN 精度降低不多的情况下，提高故障下的 MNN 精度。

3.1 m等分

在后量化方法中，如果量化位数不低于 5 位，精度不会明显降低，而当小于 3 位时，精度快速下降。因此，我们对量化位数不低于 3 位的情况进行研究。图 6 是分别对 784×256 的 FC 层进行（a）m=8，（b）m=10，（c）m=16 和（d）m=32 等分。可以看出，无论 m 是什么，权重值都集中在中心位置，接近零值。

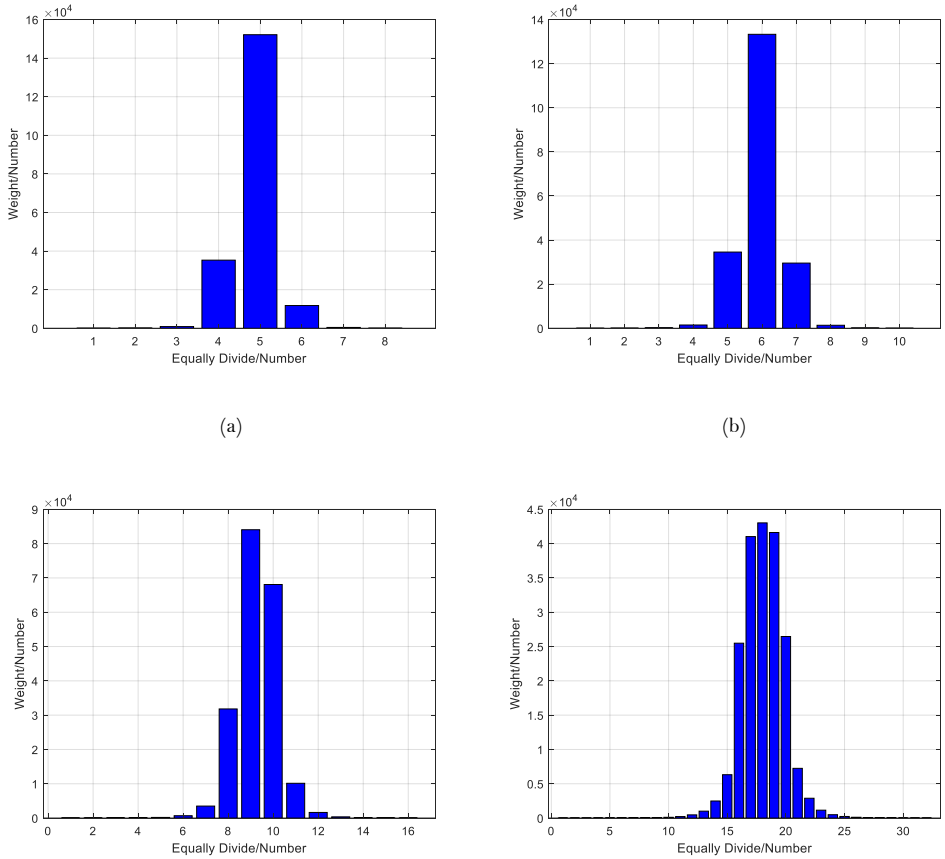


图 6　784×256 的FC层参数m等分。（a）m=8，（b）m=10，（c）m=16，（d）m=32

针对公式 6 中-w^r_{max-}和 w^r_{max+}较难确定的问题，本文将提出求解较好的-w^r_{max-}和 w^r_{max+} 的方法，使得带故障 MNN 取得较高的精度。求解过程采用了量化的方式，可以描述为：

$$w_{c2}^{\text{eff}}=\begin{cases}-w_{\max-}+\dfrac{\Delta}{m}k_1, & w_c\le -w_{\max-}+\dfrac{\Delta}{m}k_1\\ w_c,\ w_k\in\left[-w_{\max-}+\dfrac{\Delta}{m}k_1, w_{\max+}-\dfrac{\Delta}{m}k_2\right] & \\ w_{\max+}-\dfrac{\Delta}{m}k_2, & w_c\ge w_{\max+}-\dfrac{\Delta}{m}k_2\end{cases}\tag{7}$$

其中Δ= w_{max+} -（-w_{max-}），k_1，$k_2\in\{0, 1, \ldots, m/2-1\}$，$k_1$、$k_2$和 m为自然数。k_1和 k_2为左右两边截取掉的区间数，当 k_1 =k_2 =0 时，没有截取任何的等分区间，写入 MNN 中的权重值 w^{eff}_{c2}为原始写入权重值 w^{eff}_{c0}。

3.2 k_1、k_2和m的获取

k_1、k_2 和 m 值的获取可以使用穷举搜索法，k_1 和 k_2 所有可能取值为 $m/2$ 个，运算次数为$(m/2)^{2N_L}$，时间复杂度可以为$O(m^{2N_L})$，其中 N_L是层数。随着 m的增加，穷举算法的运算次数急剧增加。一般 m取 32 时可以获得较好的识别精度。

为了降低搜索算法的复杂度，在搜索的初期，先假定 k_1=k_2，粗分权重区间，取 m=8。之后，再逐步细分权重区间，增加 m的值，并在最优点附近搜索。最后，再使得 $k_1\neq k_2$，进行细调。我们可以采用 Monto-Carlo 的方式，随机注入 10 次相同

概率的故障，用树形结构方式[39][40]。对其进行搜索获取当前 k_1、k_2 和 m 下的精度，如图 7 所示。搜索算法的主要步骤如下：

第一步 对于整个网络每一层的权重，进行 $m=2^3$ 等分，穷举搜索出具有最优精度 a_1 的 $k_1^l=k_2^l$，赋给 i，其中 $l\in\{1, 2,\ldots, N_L\}$，记为 i；

第二步 $m=2\times m$，在 $k_1^l=k_2^l\in\{2\times i-2, 2\times i-1, 2\times i, 2\times i+1, 2\times i+2\}$中进行搜索，获取最大精度 a_2 对应 k_1^l和 k_2^l的取值，赋给 i；

第三步 判断精度的提高是否超过给定的阈值$\boldsymbol{\tau}$。若超过，转第二步，否则继续执行下一步；

第四步 此时，对 $k_1^l\neq k_2^l$进行细调，在 $k_1^l\in\{i-1, i, i+1\}$，$k_2^l\in\{i-1, i, i+1\}$中进行搜索，获取最大精度 a_4，a_4 为最后的精度。

第一步的运算次数为 4^{N_L}，第二步的运算次数为 5^{N_L}，第四步的运算次数为 $(3\times 3)^{N_L}$。第二步要被执行 $log_2 m-3$ 次，因此总运算次数为 $4^{N_L}+(log_2 m-3)\times 5^{N_L}+(9)^{N_L}$，时间复杂度为 $O(log_2 m)$。时间复杂度急剧下降，由穷举法的 $2N_L$ 阶降到只有对数阶。假如 $m=16$，$N_L=3$， 穷举搜索的运算次数为$(8\times 8)^3=262144$。本文提出搜索算法的运算次数仅为 $4^3+5^3+3^6=918$，速度提高了 284 倍。当 $m=32$，$N_L=3$ 时，穷举搜索的运算次数为$(16\times 16)^3=16777216$ 。提出搜索算法的运算次数仅为 $4^3+5^3+5^3+3^6=1043$，速度提高了 16084 倍。

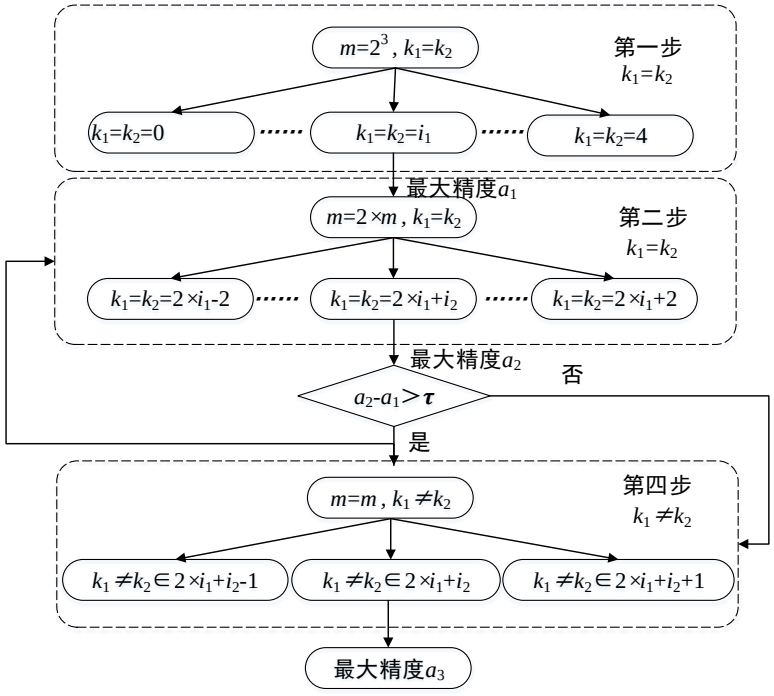


图 7 树形结构方式搜索

k_1 、k_2 和 m 获取算法的伪代码如算法 1 所示，该算法的主要目的是在最佳精度的前提下获得最优等分 m 和每层权重 $k_1^1, k_2^1, \ldots, k_1^l, k_2^l, \ldots, k_1^{N_L}, k_2^{N_L}$ 压缩值，$k_1^1, k_2^1, \ldots, k_1^l, k_2^l, \ldots, k_1^{N_L}, k_2^{N_L}$ 分别为第一层到第 N_L 层的权重压缩值。在本文中，根据量化方法，当 $m<8$ 时，精度下降相对较快，因此本文选择 $m\geq 8$ 用于以下选择 $k_1^1, k_2^1, \ldots, k_1^l, k_2^l, \ldots, k_1^{N_L}, k_2^{N_L}$。

算法 1. 最大精度获取

输入：离线训练 CNN 权重为 Model，MCA 中故障的位置，随机注入概率为 P 的故障，精度的提高阈值$\boldsymbol{\tau}$

输出：新的模型 Model'和最大精度

1. 随机注入概率为 P 的固定故障

2. $m=2^3$

3. for k_1^1 in range(m/2-1), …, k_1^s in range(m/2-1), …, $k_1^{N_1}$ in range(m/2-1) #在搜索初期，$k_1^s=k_2^s$

4. 权重区间$[-w_{max-}, w_{max+}]$ to $[-w_{max-}^r, w_{max+}^r]$，得到最大精度 a_1 对应的 i_1^1, …, i_1^s,…, $i_1^{N_1}$

5. $m=2\times m$

6. for k_1^1 *in range*($2*i_1^1-2, 2*i_1^1+2$)，k_1^s *in range*($2*i_1^s-2, 2*i_1^s+2$)，$k_1^{N_1}$ *in range*($2*i_1^{N_1}-2, 2*i_1^{N_1}+2$)

7. 权重区间$[-w_{max-}, w_{max+}]$ to $[-w_{max-}^r, w_{max+}^r]$，得到最大精度 a_2 对应的 i_1^1, …, i_1^s,…, $i_1^{N_1}$

8. if $a_2 - a_1 > \tau$ $a_1=a_2$, goto 5

9. for k_1^1 *in range*(i_1^1-1, i_1^1+1)，k_2^1 *in range*(i_1^1-1, i_1^1+1)，k_1^s *in range*(i_1^s-1, i_1^s+1) #在搜索末期，$k_1^s \neq k_2^s$

for k_2^s *in range*(i_1^s-1, i_1^s+1)，$k_1^{N_1}$ *in range*($i_1^{N_1}-1, i_1^{N_1}+1$)，$k_2^{N_1}$ *in range*($i_1^{N_1}-1, i_1^{N_1}+1$)

10. 权重区间$[-w_{max-}, w_{max+}]$ to $[-w_{max-}^r, w_{max+}^r]$，得到最大精度 a_3 对应的 $i_1^1, i_2^1, \ldots, i_1^s, i_2^s, \ldots, i_1^{N_1}, i_2^{N_1}$

11.最后输出新的模型 Model'和精度 a_3，结束。

3.3 提出方法对应的网络结构设计

本文不需要控制器单元[30]、选择单元[30]来交换输入缓冲区顺序或 MCA 排序[29]，采用输入和输出缓冲器的形式与 MCA 进行交互，总线执行主机处理，并且与缓冲器进行交互，如图 8 所示。本文方法相对于文献[30]的 MNN 结构简单，在更换 MCA 时也不需要重新设置控制单元（Controller）和 MCA 分块。

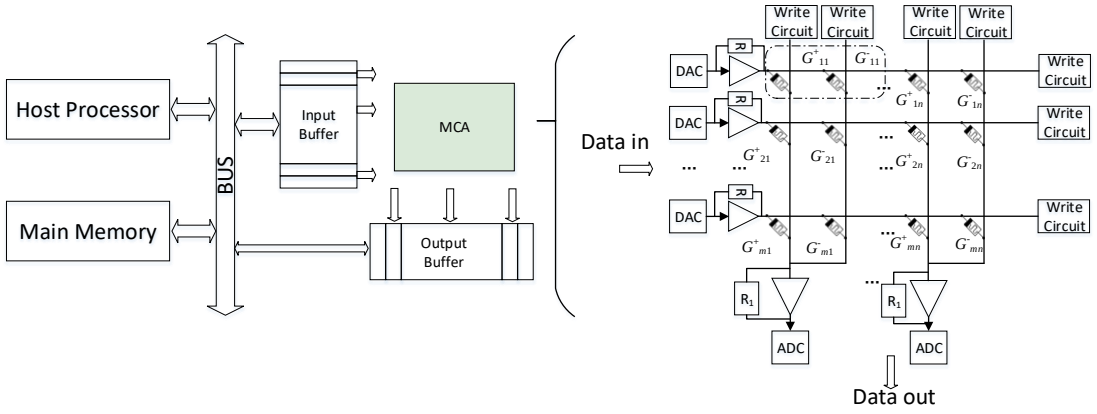


图 8 提出方法对应的网络结构

4 实验结果

本文使用手写数字分类数据集 MNIST 和图像分类数据集 CIFAR-10 进行评估。这两个数据集都由 60000 张图像组成，这些图像被划分为十类，分别随机选择 50000 和 10000 张图像用于训练和评估。CIFAR-10 数据集中的图像有三个颜色通道，而 MNIST 数据集中的图像是单通道的。我们使用三层和四层前馈神经网络（FF NN）[17]-[19] [29]在 MNIST 数据集上进行训练，使用 7 层 CNN[16] [19]在 CIFAR-10 数据集上进行训练，网络设置如表 1，0 所示。本文采用 linux 环境，1080ti 显卡，pytorch 下的仿真实验。

表 1 MNIST 数据集在 FF NN 的实验

网络结构	权重尺寸	原始精度
3 层 FC	784×256×10	97.5%
4 层 FC	784×500×300×10	98.0%

表 2　CIFAR-10 数据集在 CNN 上的实验

层	权重尺寸	原始精度
Conv1	3×3×3×32	77.4%
Conv2	32×3×3×32	
Conv3	32×3×3×64	
Conv4	64×3×3×64	
FC1	2304×512	
FC2	512×10	

本文的实验包括精细或粗略修改无故障下网络的权值，改变卷积核大小，使用或不使用硬件冗余，与其它搜索算法的对比等，本文同时对预先是否获取 MCA 故障位置进行了分析。

4.1 评估FF NN的影响

（1）评估 MNIST 在 FF NN 上的影响

在 MNIST 上与其它方法的对比实验如表 3 所示，采用的网络结构为表 1 中的 3 层和 4 层 FF NN。“*P*”是故障的百分比，“*P*=10%”或“*P*=20%”，表示 MCA 中总共有 10%或 20%的固定故障。通过对不同概率的 SAF 进行 Monto-Carlo 仿真，记录平均恢复精度和最大最小（[min，max]）恢复精度。文献[17] [19]分别采用行翻转变换（F），神经元排序（P），权重范围变换（V），以及它们的组合。文献[29]中的 PMV 采用数字偏置补偿、考虑激活值方差和权重误差的神经元排序技术。文献[18]采用每列神经元排列、标记神经元位置和 MCA 分块的方式。没有给出的结果用“-”表示，“R=1”为无硬件冗余，“R=2”为 1 倍硬件冗余，所有方法运行次数都是 10 次。

表 3　评估 3 层 FF NN 和 4 层 FF NN 的 MNN 在 MNIST 数据集上的影响

P	方法	R	恢复精度（%） 3 层 FF NN		4 层 FF NN	
			[min，max]	平均值	[min，max]	平均值
10%	V[19]	1	[11.5,68.6]	44.7	-	38.0
	P[17]		[59.3,90.5]	78.5	[33.7,51.9]	43.3
	PMV[29]		[92.5,97.3]	96.1	[41.7,71.3]	55.8
	VPF[19]		[54.9, 84.0]	71.7	-	73.7
	[18]		-	98.3	-	-
	Our		[97.9, 99.3]	**98.7**	[98.6, 99.5]	**99.2**
	V[19]	2	[90.5,98.5]	96.2	-	99.1
	P[17]		[99.3,99.9]	99.6	[99.4,99.8]	99.6
	PMV[29]		[99.7,100.0]	**99.8**	[99.5,99.8]	99.6
	VPF[19]		[97.8, 100.1]	99.5	-	99.7
	Our		[98.8, 99.9]	99.6	[99.8, 100.0]	**99.9**
20%	V[19]	1	[9.4, 29.8]	15.6	-	30.3
	P[21]		[26.4,63.8]	43.4	[9.2,37.5]	19.2
	PMV[29]		[74.8,89.2]	84.0	[9.6,30.1]	19.6
	VPF[19]		[9.4, 53.2]	30.1	-	14.9
	[18]		-	**95.9**	-	-
	Our		[88.67, 96.49]	95.0	[94.73, 98.05]	**96.6**
	V[19]	2	[37.4, 80.7]	67.0	-	81.6
	P[21]		[96.7,98.7]	98.1	[86.6,95.6]	92.2
	PMV[29]		[97.8,99.3]	98.8	[89.1,96.1]	93.5

	VPF[19]	[91.9, 99.3]	97.6	-	**99.7**
	Our	[97.0, 99.6]	**98.9**	[99.1,99.7]	99.5

表 3 为数据集 MNIST 在 MNN 上的评估值。从表中可以看出，随着故障率“*P*”的增大，恢复精度提高的百分比更多，主要是由于本文方法通过树形结构的全局和局部搜索，获得更合适的权重范围以减少由大量故障造成的精度损失。在“*P*=10%”、“R=1”、三层和四层 FF NN 中，本文方法分别提高了 0.4%，34.6%。在“*P*=20%”、“R=1”、三层 FF NN 中，本文方法相对于文献[18]降低了 0.9%，相对于其它方法提高了 13.1%。相比文献[18]，本文方法不需要神经元排列、标记神经元位置和 MCA 分块，采用的 WQTS 更容易实现。在四层 FF NN 中，本文方法相对于除文献[18]以外的文献提高了 218.9%，且恢复精度达到 96.6%。四层 FF NN 相对于三层 FF NN 效果好的主要原因可能是由于本文方法采用的是搜索方式，对深的网络，可以采用每层不同的压缩系统进行调节。在“R=2”、三层和四层 FF NN 中，恢复精度都为 99%以上，因此区别不大。

本文采用 4 层 FF NN 在 MNIST 上与权重范围不变时注入故障的恢复精度进行对比，如图 9 所示，“Injection Faults”是 FF NN 权重范围不变时随机注入故障，“Our R=1”是本文在无硬件冗余情况下的方法，而“Our R=2”是本文在 1 倍硬件冗余的方法。恢复精度变化的影响主要集中在故障大于 20%时。当故障“*P*=20%”时，恢复精度变化从 83.9%（权重范围不变）、96.6%（Our R=1）到 99.5%（Our R=2）。当故障“*P*=40%”时，恢复精度从 25.9%、72.5%（Our R=1）变为 91.4%（Our R=2），相对于权重范围不变时注入故障有明显提高。

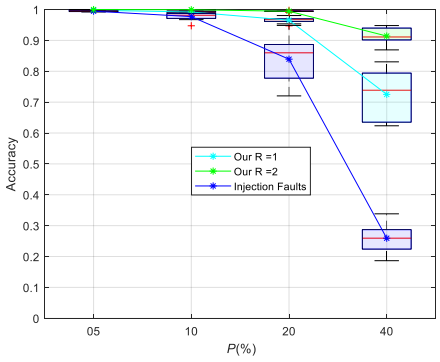


图 9 评估 MNN 在 MNIST 数据集上的 4 层 FF NN 上的差异， “Injection Faults”是 FF NN 权重范围不变时随机注入故障，“Our R=1”是本文在无硬件冗余情况下的方法，而“Our R=2”是本文在 1 倍硬件冗余的方法。

（2）MCA 故障位置分析

本文方法需要提前确定故障的位置，当故障位置不确定的时候，其精度如图 10 所示，“Our + W”为本文预先在确定故障位置时采用 WQTS，“Our + O”为预先不确定故障位置时采用 WQTS，“Injection Faults + O” 为预先不确定故障位置、FF NN 权重范围不变时随机注入故障下的精度。从图可以看出本文提出的树形结构搜索算法在故障已知和未知上相对都有较大精度的提高，然后由于实际生产过程中，MNN 精度较低会影响生产，因此本文还是采用提前确定故障的位置下的 WQTS。后续都是在已知故障位置的前提下获取最大精度。

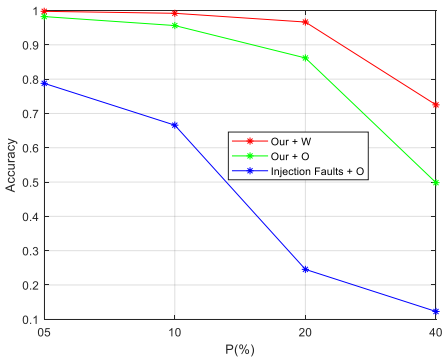


图 10 评估是否预先确认 MNN 故障位置在 MNIST 数据集上的 4 层 FF NN 影响，“Our + W”为本文算法，“Our +O”为预先不确定故障位置时采用本文提出的树形结构搜索算法，“Injection Faults + O” 为预先不确定故障位置、FF NN 权重范围不变时随机注入故障下的精度

（3）不同的搜索算法

本文根据权重分布和量化理论的方式获取故障下的最大精度，采用树形结构方式对其进行搜索，本文首先采用在低的 m 等分进行全局搜索，然后在高的 m 等分进行局部搜索的方式。本文通过实验对比穷举法（Ergodic）、逐层（layer-by-layer[42]）和梯度下降（Gradient descent）的搜索方式。其结果如图 11 所示，从图中可以看出，本文算法相对于 Ergodic 的精度相当或略有下降，在速度上提高很多。本文相对于 layer-by-layer 和 Gradient descent 的搜索算法都有所提高，在故障为 20%的时候精度提高了 2.7%，在故障为 40%的时候精度提高了 18.5%，主要是本文算法结合了全局搜索和局部搜索的方式，由于故障增加时，MNN 的精度可能波动较大，layer-by-layer 和 Gradient descent 的搜索算法容易陷入局部的最优。

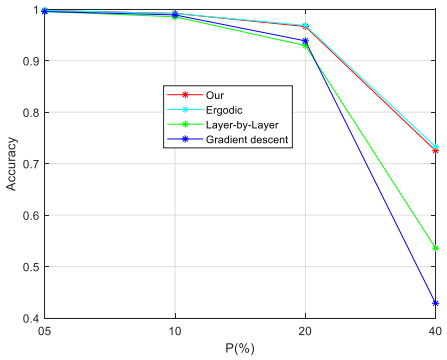


图 11不同的搜索算法

4.2 评估CNN的影响

（1）评估在 CIFAR-10 在 CNN 网络上的影响

7 层 CNN 在 CIFAR-10 上的 MNN 与其它方法的对比实验如表 4 所示，采用的网络结构为 0 中的 7 层 CNN。通过对不同概率的 SAF 进行 Monto-Carlo 仿真，记录平均恢复精度和方差。文献[16]采用冗余 MCA 和冗余列的方法替换掉故障部件以恢复识别精度，文献[19]分别采用权重范围变换（V），神经元排序（P），行翻转变换（F）。没有给出的结果用“-”表示，“R=1”为无硬件冗余，“R=2”为 1 倍硬件冗余，所有方法运行次数都是 10 次。从表 4 可以看出，本文相对文献[16] [19]有明显的提升，提升主要是在“R=1”处，在“P=5%”时提高了 3 倍，在“P=10%”时提高了 5 倍。

表 4 评估七层 CNN 的 MNN 在 CIFAR-10 数据集上的影响

P	方法	R	恢复精度（%）	
			方差（%）	平均值（%）
5%	-[16]	1	1.9	14.7

10%	VPF[19]		3.3	22.9
	Our		0.5	**97.3**
	-[16]		6.5	42.8
	VPF[19]	2	3.1	93.0
	Our		0.1	**99.9**
	-[16]		1.0	13.7
	VPF[19]	1	1.1	14.4
	Our		2.3	**91.6**
	-[16]		3.1	23.3
	VPF[19]	2	7.5	64.2
	Our		0.6	**99.0**

本文采用 7 层 CNN 在 CIFAR-10 上的 MNN 与权重范围不变时注入故障的恢复精度进行对比，如图 12 所示，“Injection Faults”是 CNN 权重范围不变时随机注入故障，“Our R=1”是本文在无硬件冗余情况下的方法， “Our R=2”是本文在 1 倍硬件冗余的方法。恢复精度变化的影响主要集中在故障大于 20%时。当故障“P=20%”时，恢复精度变化从 70.6%（权重范围不变）、75.3%（Our R=1）到 93.1%（Our R=2）。当故障“P=40%”时，恢复精度从 27.6%（权重范围不变）、37.3%（Our R=1）变为 58.3%（Our R=2），相对于权重范围不变时注入故障有了明显提高。

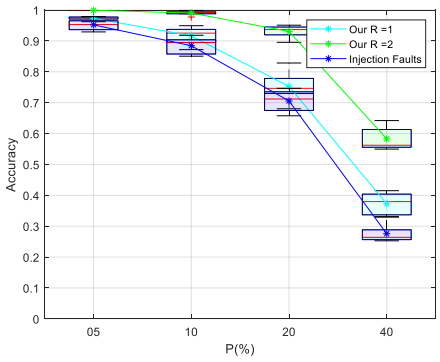


图 12 评估 MNN 在 CIFAR-10 数据集上的 7 层 CNN 上的差异，“Injection Faults”是 CNN 权重范围不变时随机注入故障，“Our R=1”是本文在无硬件冗余情况下的方法，而“Our R=2”是本文在 1 倍硬件冗余的方法。

（2）更改卷积网络内核大小

为了研究本文方法的鲁棒性，本文通过改变卷积网络内核大小进行对比。修改卷积网络主要包括修改网络结构[35]和卷积内核[36][37]。网络结构的自适应容错需要大量时间和资源，卷积内核更改很容易实现。InceptionNets [38]采用级联设计，用于聚合来自“Inception”构建块内的 3×3、5×5、7×7 卷积核的多尺度信息。因此，我们从步长、填充和 FC 层更改卷积网络内核大小，如表 5 所示，精度变化如图 13 所示。曲线是本文方法“R=1”的精度，直方图是本文方法“R=1”的精度直接减去权重范围不变时随机注入故障的精度。

表 5 在数据集 MNIST 上 CNN 改变卷积核尺寸

网络核尺寸	权重尺寸	理想精度
Conv3×3	3×3×1×8 3×3×8×12 192×10	96.97%
Conv5×5	5×5×1×8 5×5×8×12 192×10	97.85%
Conv7×7	7×7×1×8 7×7×8×12 192×10	98.43%
Conv9×9	9×9×1×8 9×9×8×12 108×10	98.52%

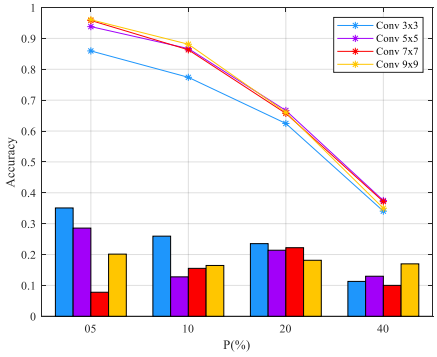
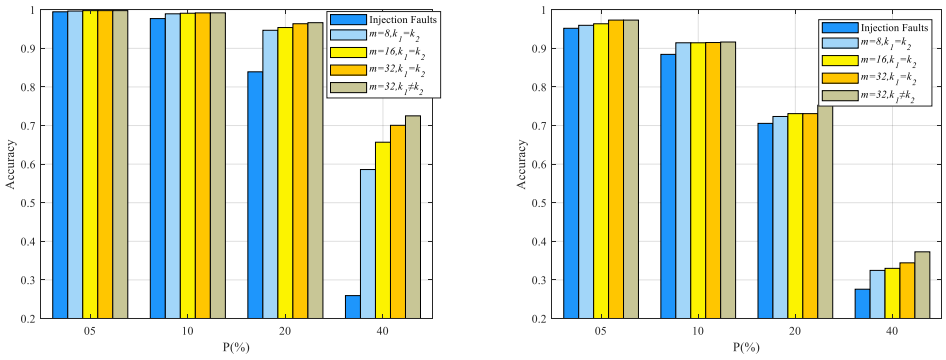


图 13 CNN 改变卷积核尺寸，曲线是本文方法在无硬件冗余“R=1”上的恢复精度，直方图是本文方法的恢复精度直接减去权重范围不变时的精度

从图中可以看出，随着 CNN 卷积核大小的增加恢复精度有所提高，Conv9×9 比 Conv3×3 的恢复精度有所提高，可能是由于随着卷积核的增加，CNN 的参数量也显著增加，参数量多的话，更能容忍故障。当“P（%）”大于 20%时，恢复精度下降的较快，主要是由于某些故障可能聚集在一起。因此，在实际产品化中应该选择故障小于 20%和卷积核大于等于 5×5 的 MCA。

4.3 k_1、k_2和m的差异

m 的值一般选择 8,16,32。本文分别选择(m=8, k_1=k_2), (m=16, k_1=k_2), (m=32, k_1=k_2), (m=32, k_1≠k_2) 4 种情况，分别采用 4 层 FF NN 和 7 层 CNN 进行实验。如图 14 所示，“Injection Faults”是权重范围不变时随机注入故障，其它分别为(m=8, k_1=k_2), (m=16, k_1=k_2), (m=32, k_1=k_2), (m=32, k_1≠k_2) 下的精度，从图中可以看出，本文方法相比权重范围不变时随机注入故障有明显的提升，随着故障率的增加，恢复精度提高的越明显，在 4 层 FF NN 故障率为 20%时候，恢复精度分别提高了 12.83%、13.67%、14.85%、15.17%。在 7 层 CNN 故障率为 20%时候，恢复精度分别提高了 2.53%、3.58%、3.58%、6.67%。因此本文在低故障率的时候选择小的 m 值，在高故障率的时候选择大的 m 值。因此阈值$\boldsymbol{\tau}$的设置可以根据故障率进行设置，“P<20%”时，$\boldsymbol{\tau}$=0.2%，“P≥20%”时，$\boldsymbol{\tau}$=0.5%。



（a） （b）

图 14 k_1、k_2和 m 不同情况下的恢复精度。（a）为 4 层 FF NN 的恢复精度，（b）7 层 CNN 的恢复精度

5 结论

本文提出了一种无需硬件冗余、神经元排列、重训练的前提下，只修改权重变化范围的基于权重范围变换的 MNN 容错方法，采用权重量化和树形结构的权重变化范围搜索方法获得最优精度时权重变化范围的搜索空间。本文在 3 层、4 层 FF NN、CNN 上进行实验都取得了较好的效果，本文同时分析了改变卷积核大小，使用或不使用硬件冗余，与其它搜索算法进

行对比的情况。然而当 CNN 层很大或网络结构非常复杂时，计算复杂度将变得非常高，因此研究复杂网络下的精度提升是后续研究的主要问题。

参 考 文 献

[1] Ravikumar K I. Memristor-Based Deep Learning Classification Model for Object Detection[J]. ECS Transactions, 2022, 107(1): 277.

[2] Yongbin Y U, Chenyu Y, Quanxin D, et al. Memristive network-based genetic algorithm and its application to image edge detection[J]. Journal of Systems Engineering and Electronics, 2021, 32(5): 1062-1070.

[3] Lammie C, Xiang W, Azghadi M R. Towards memristive deep learning systems for real-time mobile epileptic seizure prediction[C]//2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2021: 1-5.

[4] Song W, Zhou Y, Hu X, et al. Memristive neural network based reinforcement learning with reward shaping for path finding[C]//2018 5th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS). IEEE, 2018: 200-205.

[5] Petavratzis E, Volos C, Ouannas A, et al. A 2d discrete chaotic memristive map and its application in robot's path planning[C]//2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST). IEEE, 2021: 1-4.

[6] Guo H, Peng L, Zhang J, et al. ATT: A fault-tolerant ReRAM accelerator for attention-based neural networks[C]//2020 IEEE 38th International Conference on Computer Design (ICCD). IEEE, 2020: 213-221.

[7] Kang M, Shin H, Kim L S. A Framework for Accelerating Transformer-Based Language Model on ReRAM-Based Architecture[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021, 41(9): 3026-3039.

[8] Liu C, Gao Z, Liu S, et al. Special Session: Fault-Tolerant Deep Learning: A Hierarchical Perspective[C]//2022 IEEE 40th VLSI Test Symposium (VTS). IEEE, 2022: 1-12.

[9] Huang L, Yu H, Chen C, et al. A training strategy for improving the robustness of memristor-based binarized convolutional neural networks[J]. Semiconductor Science and Technology, 2021, 37(1): 015013.

[10] Chen Y, Fan Z, Dong S, et al. A Multifault‐Tolerant Training Scheme for Nonideal Memristive Neural Networks[J]. Advanced Intelligent Systems, 2022, 4(5): 2100237.

[11] Oli-Uz-Zaman M, Khan S A, Yuan G, et al. Mapping transformation enabled high-performance and low-energy memristor-based DNNs[J]. Journal of Low Power Electronics and Applications, 2022, 12(1): 10.

[12] Meng Z, Sun Y, Qian W. Write or not: programming scheme optimization for RRAM-based neuromorphic computing[C]//Proceedings of the 59th ACM/IEEE Design Automation Conference. 2022: 985-990.

[13] Shim W, Seo J, Yu S. Two-step write–verify scheme and impact of the read noise in multilevel RRAM-based inference engine[J]. Semiconductor Science and Technology, 2020, 35(11): 115026.

[14] Liu P, You Z, Wu J, et al. Fault modeling and efficient testing of memristor-based memory[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(11): 4444-4455.

[15] Chen C Y, Chakrabarty K. Efficient identification of critical faults in memristor crossbars for deep neural networks[C]//2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021: 1074-1077.

[16] Xia L, Huangfu W, Tang T, et al. Stuck-at fault tolerance in RRAM computing systems[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2017, 8(1): 102-115.

[17] Zhang B, Uysal N, Fan D, et al. Handling stuck-at-faults in memristor crossbar arrays using matrix transformations[C]//Proceedings of the 24th Asia and South Pacific Design Automation Conference. 2019: 438-443.

[18] Ma Y, Zheng L, Zhou P. Tolerating Stuck-at Fault and Variation in Resistive Edge Inference Engine via Weight Mapping[C]//Proceedings of the 2021 on Great Lakes Symposium on VLSI. 2021: 313-318.

[19] Zhang B, Uysal N, Fan D, et al. Handling stuck-at-fault defects using matrix transformation for robust inference of DNNs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(10): 2448-2460.

[20] Lin J, Wen C D, Hu X, et al. Rescuing RRAM-based computing from static and dynamic faults[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(10): 2049-2062.

[21] Meng Z, Oian W, Zhao Y, et al. Digital offset for rram-based neuromorphic computing: A novel solution to conquer cycle-to-cycle variation[C]//2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021: 1078-1083.

[22] Yuan G, Liao Z, Ma X, et al. Improving DNN fault tolerance using weight pruning and differential crossbar mapping for reram-based edge ai[C]//2021 22nd International Symposium on Quality Electronic Design (ISQED). IEEE, 2021: 135-141.

[23] Zhang W, Gao B, Yao P, et al. Array-level boosting method with spatial extended allocation to improve the accuracy of memristor based computing-in-memory chips[J]. Science China Information Sciences, 2021, 64(6): 160406.

[24] Xu Q, Chen S, Geng H, et al. Fault tolerance in memristive crossbar-based neuromorphic computing systems[J]. Integration, 2020, 70: 70-79.

[25] Liu C, Hu M, Strachan J P, et al. Rescuing memristor-based neuromorphic design with high defects[C]//Proceedings of the 54th Annual Design Automation Conference 2017. 2017: 1-6.

[26] Yeo I, Chu M, Gi S G, et al. Stuck-at-fault tolerant schemes for memristor crossbar array-based neural networks[J]. IEEE Transactions on Electron Devices, 2019, 66(7): 2937-2945.

[27] Song Z, Sun Y, Chen L, et al. ITT-RNA: Imperfection tolerable training for RRAM-crossbar-based deep neural-network accelerator[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(1): 129-142.

[28] Yildiz H A, Altun M, Gungordu A D, et al. Analog neural network based on memristor crossbar arrays[C]//2019 11th International Conference on Electrical and Electronics Engineering (ELECO). IEEE, 2019: 358-361.

[29] Zhang B, Uysal N, Ewetz R. Stat: Mean and variance characterization for robust inference of dnns on memristor-based platforms[C]//Proceedings of the 2019 on Great Lakes Symposium on VLSI. 2019: 339-342.

[30] Jain S, Raghunathan A. Cxdnn: Hardware-software compensation methods for deep neural networks on resistive crossbar systems[J]. ACM Transactions on Embedded Computing Systems (TECS), 2019, 18(6): 1-23.

[31] de Lima J P C, Carro L. Quantization-aware in-situ training for reliable and accurate edge AI[C]//2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2022: 1497-1502.

[32] Greenberg-Toledo T, Perach B, Hubara I, et al. Training of quantized deep neural networks using a magnetic tunnel junction-based synapse[J]. Semiconductor Science and Technology, 2021, 36(11): 114003.

[33] Nahshan Y, Chmiel B, Baskin C, et al. Loss aware post-training quantization[J]. Machine Learning, 2021, 110(11-12): 3245-3262.

[34] Gholami A, Kim S, Dong Z, et al. A survey of quantization methods for efficient neural network inference[J]. arXiv preprint arXiv:2103.13630, 2021.

[35] Ning X, Ge G, Li W, et al. FTT-NAS: Discovering fault-tolerant convolutional neural architecture[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2021, 26(6): 1-24.

[36] Li X, Wang W, Hu X, et al. Selective kernel networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 510-519.

[37] Ye Z, Yu J. AKSNet: A novel convolutional neural network with adaptive kernel width and sparse regularization for machinery fault diagnosis[J]. Journal of Manufacturing Systems, 2021, 59: 467-480.

[38] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2017, 31(1).

[39] Huang Y, Lai L, Li W, et al. A differential evolution algorithm with ternary search tree for solving the three-dimensional packing problem[J]. Information Sciences, 2022, 606: 440-452.

[40] Zhu H, Chang D, Xu Z, et al. Joint optimization of tree-based index and deep model for recommender systems[J]. Advances in Neural Information Processing Systems, 2019, 32.

[41] Ma Y, Zheng L, Zhou P. A Mapping Method Tolerating SAF and Variation for Memristor Crossbar Array Based Neural Network Inference on Edge Devices[J]. ACM Journal on Emerging Technologies in Computing Systems, 2023.

[42] Guo J, Ouyang W, Xu D. Channel pruning guided by classification loss and feature importance[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(07): 10885-10892.

An Efficient Hybrid Approximate Adder Tree Based on Mismatch Propagation Analysis for CNN Deployment

1st Lvyang Zhou
School of Electronic Science and Engineering
Southeast University
Nanjing, China
lvyangzhou@seu.edu.cn

2nd Yuanhao Wang
School of Integrated Circuits
Southeast University
Wuxi, China
wangyh21@seu.edu.cn

3rd Bo Liu
School of Electronic Science and Engineering
Southeast University
Nanjing, China
liubo_cnasic@seu.edu.cn

***Abstract*—The approximate calculation of data across different networks may result in a variety of mismatches. In this paper, two typical neural networks are analyzed to explore the accuracy loss caused by different types of mismatches in approximate adder structures. Based on this analysis, a hybrid approximate adder tree structure is proposed. This structure allows for an adaptive balance between accuracy and power consumption in the machine learning accelerator through heterogeneous design and software/hardware coordination. Then, we tested this proposed approximate adder tree structure in a BCNN accelerator. This accelerator was implemented and evaluated with 22 nm technology, resulting in 25% less power consumption with less than 1% precision loss.**

***Index Terms*—approximate computing, binary network, deviation analysis, structure of approximate adders**

I. Introduction

Recently, deep convolutional neural networks have achieved great success in many areas such as AIoT. The spread of edge devices calls for low-power and high efficiency design. [1]. This leads to many applications of approximate calculation, which can achieve comparable accuracy with less power consumption [2].

Several techniques have been thoroughly investigated, including network quantization, model pruning, sparse compression and so on. Among these methods, an extremum quantization policy can be used to quantize the weight or activation to 1 bit [3]. The Binary Convolutional Neural Network (BCNN), as illustrated in Fig. 1, is used to reduce the initial multiplication to bitwise, which greatly increases the energy efficiency of the system.

Taking into account the computational properties of BCNN, multiplication is practically removed from the network. In BCNN, the original convolution operation is simplified from multiplication to summation. Therefore, it is more important to research the approximate adder for deploying BCNN. Recently, the approximation of adders has attracted considerable attention in the field of approximation computing. These works range from designing low-power adders [2], [4]–[6] to designing high-performance adders [7]–[9]. If there is no detailed analysis of the computation quality of the approximation adder, the accuracy loss could be higher than expected, resulting in an unacceptable deterioration in BCNN performance. According to the calculation features of BCNN, it is necessary to analyze the variation of computation quality during the accumulating process.

The mean error distance is an important parameter in the evaluation of approximate computational structure design. Traditionally, the Monte Carlo method is used to measure the statistic error of the adder/multiplier in neural networks . The method has the advantages of simplicity and maturity of the arithmetic. However, this approach has an extreme complexity in computing since it focuses on the constraint functions and the model of the approximate adders [10].

Recently, a compact error model has been developed [11], which can be used to simulate the accuracy loss caused by the introduction of approximation. This concept has been adopted and applied in a number of areas of research. Our research is also based on the same concept.

II. Hybrid Approximate Adder Tree Based on Mismatch Propagation Analysis

A. Method and Network Used

In convolution, the weight and the source data are sensitive to nonmatching [12], [13]. In this paper, the mismatch is added as the random noise into the network. Previous research has also indicated that the mismatch may have different impact on zero and non-zero data in convolutional computing [4]. They could both have strong affection in some specific circumstances.

For a better assessment of the effects in both cases, the zero data and the non-zero data are divided into two groups. The maximum offset in zero data's group is indicated as ω, whereas the maximum offset in nonzero-data's group is indicated as ϵ.

As illustrated in Fig. 2, this thesis adopts two kinds of neural networks:

This work was supported by the National Key R&D Program of China (Grant No. 2018YFB2202102).

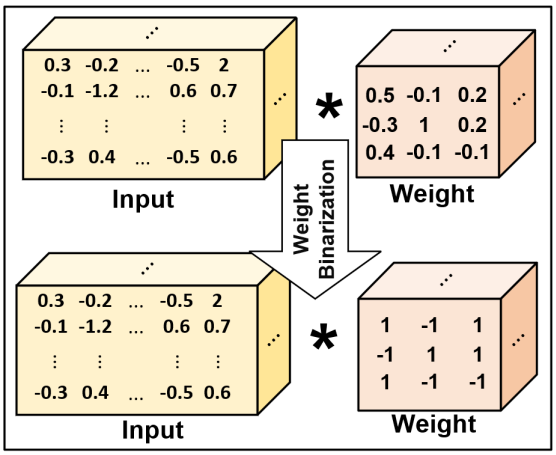


Fig. 1. BWN method used to compress the data input.

Algorithm 1: Deviation added as computing noise

Input: Convolution data (X) and weight (Y)
Output: Convolution result

```
X ⟵ the data from the former process;
while current is not the last data do
    AddDeviationFunction(X, Y);
    α ⟵ noise generated from the random factor;
    β ⟵ restriction factor for non-zero data;
    γ ⟵ restriction factor for zero data;
    if data = 0 then
        X ⟵ α · γ;
    else
        X ⟵ α · β;
    end
end
```

- LeNet-5 is one of the most fundamental and easy to implement for validation of all algorithms and hardware accelerators. It is composed of 2 convolutional layers and 2 max-pooling layers. In this paper, it is used for testing the "MNIST" dataset [14].
- ResNet-10, which is a little more complex. It has four convolutions and is designed for the "Key Word Spotting" data set (KWS) [15], [16].

B. Mismatch propagation analysis for BCNN

The analysis of mismatch propagation is divided into two parts. First is the uniform or nonuniform mismatch. In this paper, a variety of approximation adder is assessed, and they can be classified into three categories: homogeneous, co-directional, and inverse. The offset of the zero-position group ω and the non-zero position group ϵ in the analysis process are multiplied by a difference of 20% as Algorithm 1.The mean square error (MSE) is typically used as the parameter characterizing the noise power in the signal processing area. MSE is also important when calculating the best performance that may be achieved when estimating a parameter of a signal that is corrupted by noise. In general, the mean square error may be expressed as:

$$MSE = E\{e^2[n]\} = \frac{1}{N}\sum_{n=0}^{N-1} e^2[n] = \sum_{i=0}^{k-1} ED_i^2 \times P(ED_i) \tag{1}$$

For constructing heterogeneous adder tree, mean error distance (MED) is more suitable for deployment in BCNN. Because it can exploit different adders' error distribution, thus could guide the inter-compensation to improvement the overall accuracy of adder tree. In general, the mean error distance may be expressed as:

$$MED = \sum_{i=0}^{k-1} ED_i \times P(ED_i) \tag{2}$$

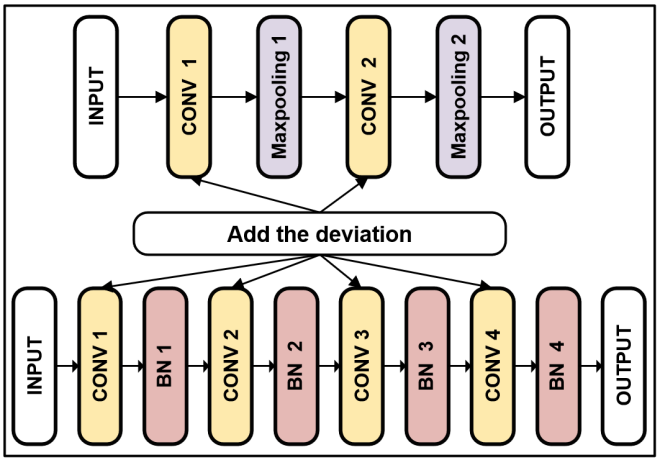


Fig. 2. Core layers of the networks used.

Fig. 3(a) illustrates the difference in precision loss due to homogeneous, co-directional and inverse mismatches. We can draw a conclusion that the mismatch can be reduced if we combine various kinds of adders. Fig. 3(b) shows the relationship between precision and various data sets ("MNIST" and "KWS" for example). In the "MNIST" dataset, the accuracy decreased slightly with the increase of nonzero mismatch, however, the accuracy dropped dramatically with the increase of the zero-position mismatch. In the "KWS" dataset, the results are quite different. The zero-position mismatch almost has no effect on precision, but the precision drops sharply when the non-zero position mismatch is added.

So, it can be concluded from the analysis that the network structure has great influence on the error rate of approximate computation. In previous work, a hybrid adder tree structure was proposed, but it can only be used in a fixed network [17]. In order to expand the application of the hybrid adder tree to more general cases, the codesign of software/hardware is adopted in order to satisfy the reconfigurability of an approximate adder.

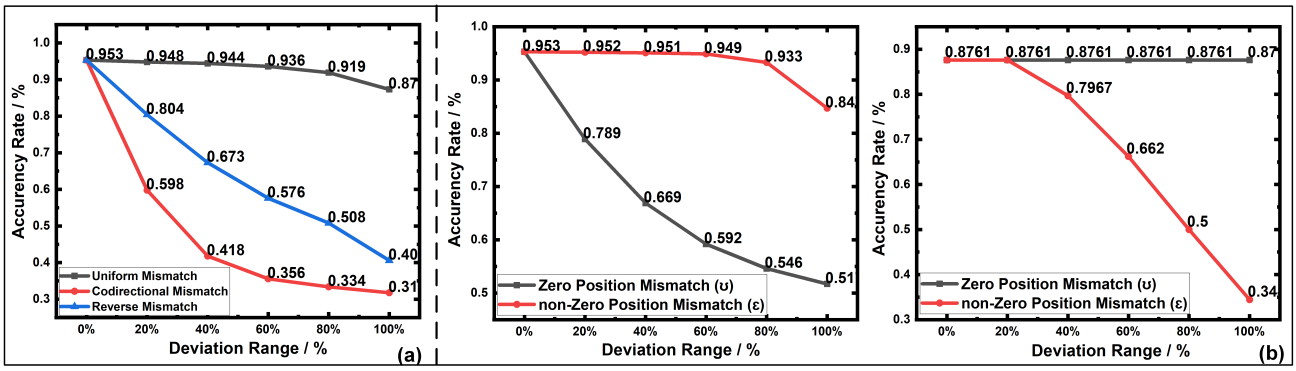


Fig. 3. (a) Accuracy rate at different deviation range by uniform or non-uniform mismatch. (b) by zero or non-zero mismatch.

C. Data Quantification and Software/Hardware Co-Design

Through quantizing, it is possible to reduce the computational load by decreasing the bit-width of the data or the weight, while not changing the initial accelerating policy [18]–[20]. A variety of quantitative strategies are used in this article, including the BCNN, which is one of the extreme cases, all weights are binarized into 1-bit 1 and -1 based on their sign bit as illustrated in Fig. 1 [3], [21]–[24].

The fault tolerance functionality is contained in the Multiply-Accumulate Operation units (MACs). On the basis of pre-analyzing and simulating the object network model, the tolerance of the model to the convolution computation error is determined. A pre-file will be generated at the same time, which determines the internal path of a hybrid adder tree structure.

Especially, for multilayer convolutions, it is possible to transfer or modify the error of each layer by the previous layer. Thus, the benefits of reconfiguration in this structure will be demonstrated. Based on the error transmission in each level of the network, the new algorithm can be used to modify the path on the fly so as to obtain the best energy-efficiency adder tree.

In this paper, the situation of BCNN is first taken into consideration. In this situation the Multiply-Accumulate operations in the networks are converted into a pure addition operation. Under such circumstances, the difference of adder tree is the sole reason for calculating error. Therefore, it is much more convenient to estimate the computation error caused by the configurable adder tree.

D. Proposed Hybrid Approximate Adder Tree

In this paper, two kinds of approximate adder structure and a specific zero-processing element are combined in one hybrid approximate adder cluster, as shown in Fig. 4. Thousands of hybrid approximation adder clusters make up a complete approximate adder. Every node in a cluster has specific links with its neighboring clusters, and which path will be activated is determined by the pre-file given from co-design.

One of the approximate adder structures used in this work is a specially designed carry upward approximate adder with

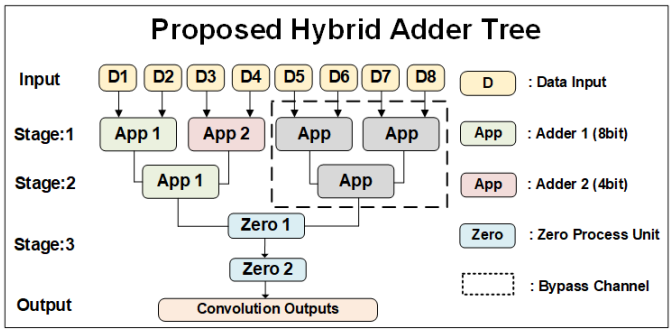


Fig. 4. Overall view of the hybrid approximate adder tree.

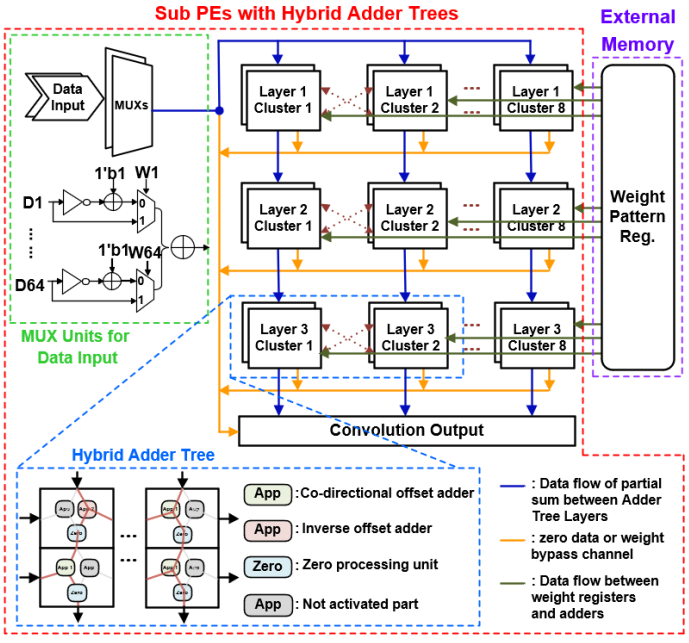


Fig. 5. Sub PEs in accelerator with hybrid adder tree structure.

moderate co-directional offset. Another is a specially designed intercept approximate adder that has some inverse offset. Furthermore, the data width of every approximation adder can

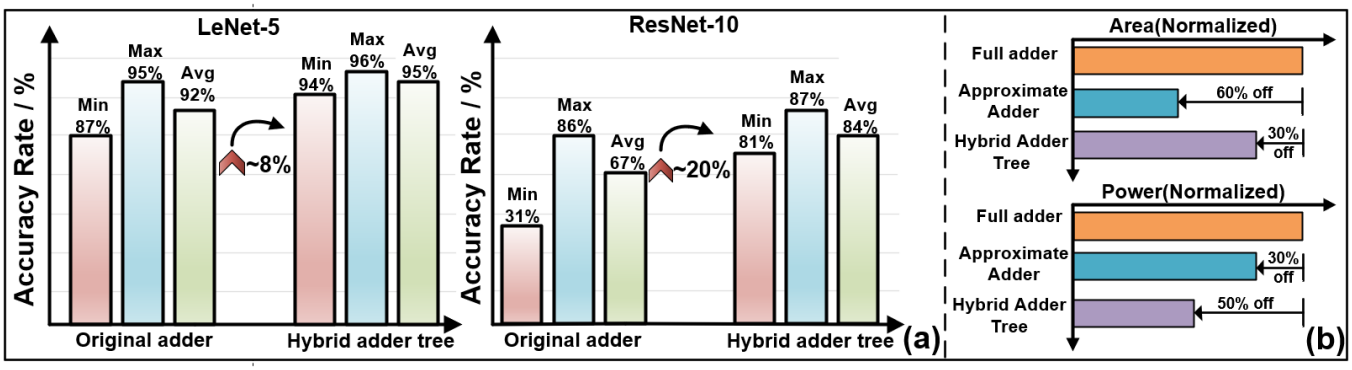


Fig. 6. (a) Accuracyrate in origin adder and hybrid tree. (b) Comparison between different adder structure.

be changed from 4 to 8 bits. Furthermore, a new zero-bypass channel is proposed in order to increase the energy efficiency of the proposed structure, which skips the unnecessary convolutional computation (e.g. the feature or weight is zero).

The variable accuracy allows the adder tree structure to be more flexible in all kinds of computation situations. For example, in the error-sensitive pre-stage convolutions, a high-precision approximate adder tree is adopted to decrease the propagation of the error to the next layers. Moreover, a low-precision approximation adder tree is applied to the post-phase convolutions in order to improve the efficiency and reduce the energy consumption of the system.

Fig. 5 illustrates a PE calculation array consisting of a hybrid approximate adder tree structure. The data is transferred to the cell via multiplexers, which automatically determines the 0-bit data and places it in the bypass channel. The weight data of convolution computation is stored in an outer memory module and distributed to all layers in the adder tree via a parallel bus. Furthermore, the interlace path is established among the neighboring adder clusters, which can be connected and expanded in order to compensate the mismatch from each other.

III. Case Study: Proposed Addition Tree Structure for LeNet-5 and ResNet-10

Based on this approach, the effectiveness and adaptability of the approximate adder tree structure are evaluated. The adder tree is constructed in a hybrid type, and the combination of different precision adders results in a low nonuniform mismatch, in which the discrete noise offsets one another. At the same time, zero bypass channels are activated, so that a number of data with zero or near-zero weight can pass through the computation without having to perform any convolution. In Fig. 6(a), the accuracy is displayed as a maximum, a minim, and an average style.

As shown in Fig. 6(a), by applying the approximate adder tree structure, the minimal accuracy ratio is greatly improved, and the mean accuracy is also improved. The ResNet-10 network improved by about 20%, and the LeNet-5 network improved by about 8%. The ResNet-10 network is slightly more complicated than that of the LeNet-5 network, so it got a more significant improvement.

The power and area were also calculated as shown in Fig. 6(b). Compared with the conventional Full Adder (FA) structure, this hybrid tree structure has a reduction of approximately 30% of the mapping area and approximately 25% of the power consumption. Remarkably, it achieves this with only about 1% precision loss, whereas all using the approximate adder yields a loss of almost 10%.

In this paper, the properties of the original adders and the mixed adder tree have been studied. Critical path delay and power consumption were assessed on a 22 nm commercial ULL process technique with a 0.6V logic source voltage and a clock frequency of 250 kHz.

In comparison to SOTA, we can see that we have a significantly improved performance. As illustrated in Table I, this hybrid structure can improve the precision by about 15%, which is only 3% lower than that of the full adder. When taking the delay and power consumption into consideration, this structure is slightly inferior to all approximate adder structure. In comparison with the full adder structure, the proposed hybrid structure is only one-third of the PDP. Therefore, this configuration has an advantage in PDP, compared to all the AMA3 or the entire InXA2 configuration. In comparison with the other AMA4 structure, the PDP is not superior, but the precision of the whole AMA4 is far superior.

TABLE I
Comparison with SOTA Works

Compare Items	This work	AMA3 [25]	AMA4 [5]	InXA2 [4]	FA
Accuracy(%)	**95.01**	**79.3**	**80.3**	**80.3**	**98.0**
Delays (ns)	2.7	3.2	2.44	2.47	4.53
Power (nW)	48.67	47.96	43.87	62.88	77.53
PDP ($\times 10^{-3} fJ$)	**118.2**	**153.5**	**107.0**	**155.3**	**351.21**

IV. Conclusion

In this paper, we present a novel BCNN deployment approach based on the hybrid approximate adder tree. In order to evaluate the computation quality of convolution operations, an analytic model based on probability was established. The

proposed hybrid adder tree has significantly improved the minimum accuracy and has improved the average accuracy. Finally, the hybrid tree is simulated on the Lenet-5 network and the ResNet-10 network. Since ResNet-10 is slightly more sophisticated than Lenet-5, it provides a more significant benefit. As expected, The ResNet-10 network improved by about 20%, and the LeNet-5 network improved by about 8%. This hybrid architecture can increase the accuracy by about 15% in comparison with the all-approximate adder when it is assessed using a 22 nm commercial ULL process technology with a 0.6 V logic source voltage and 250 kHz clock frequency. When compared with full adder structure, the hybrid adder structure is only 3% less accurate, while the power consumption is only one-third of that in full adder structure.

REFERENCES

[1] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[3] S. Yin, P. Ouyang, S. Zheng, D. Song, X. Li, L. Liu, and S. Wei, "A 141 uw, 2.46 pj/neuron binarized convolutional neural network based self-learning speech recognition processor in 28nm cmos," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 139–140.

[4] H. A. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 660–665.

[5] Z. Yang, J. Han, and F. Lombardi, "Transmission gate-based approximate adders for inexact computing," in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH 15)*. IEEE, 2015, pp. 145–150.

[6] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in *2013 13Th IEEE international conference on nanotechnology (IEEE-NANO 2013)*. IEEE, 2013, pp. 690–693.

[7] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *2013 IEEE/ACM international conference on computer-aided design (ICCAD)*. IEEE, 2013, pp. 48–54.

[8] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[9] M. A. Hanif, R. Hafiz, O. Hasan, and M. Shafique, "Quad: Design and analysis of quality-area optimal low-latency approximate adders," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[10] J. Huang, J. Lach, and G. Robins, "Analytic error modeling for imprecise arithmetic circuits," *Proc. SELSE*, vol. 51, pp. 64–69, 2011.

[11] M. Pashaeifar, M. Kamal, A. Afzali-Kusha, and M. Pedram, "A theoretical framework for quality estimation and optimization of dsp applications using low-power approximate adders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 327–340, 2018.

[12] B. Liu, H. Cai, Z. Zhang, X. Ding, Z. Wang, Y. Gong, W. Liu, J. Yang, Z. Wang, and J. Yang, "More is less: Domain-specific speech recognition microprocessor using one-dimensional convolutional recurrent neural network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1571–1582, 2021.

[13] B. Liu, H. Cai, Z. Wang, Y. Sun, Z. Shen, W. Zhu, Y. Li, Y. Gong, W. Ge, J. Yang *et al.*, "A 22nm, 10.8 μ w/15.1 μ w dual computing modes high power-performance-area efficiency domained background noise aware keyword-spotting processor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4733–4746, 2020.

[14] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[15] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[16] P. P. Bernardo, C. Gerum, A. Frischknecht, K. Lübeck, and O. Bringmann, "Ultratrail: A configurable ultralow-power tc-resnet ai accelerator for efficient keyword spotting," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4240–4251, 2020.

[17] B. Liu, Z. Wang, X. Wang, R. Zhang, A. Xue, Q. Shen, N. Xie, Y. Gong, Z. Wang, J. Yang *et al.*, "An efficient bcnn deployment method using quality-aware approximate computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4217–4228, 2022.

[18] B. Liu, Z. Wang, W. Zhu, Y. Sun, Z. Shen, L. Huang, Y. Li, Y. Gong, and W. Ge, "An ultra-low power always-on keyword spotting accelerator using quantized convolutional neural network and voltage-domain analog switching network-based approximate computing," *IEEE Access*, vol. 7, pp. 186 456–186 469, 2019.

[19] Y. Zhou, X. Hu, L. Wang, G. Zhou, and S. Duan, "Quantbayes: Weight optimization for memristive neural networks via quantization-aware bayesian inference," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 12, pp. 4851–4861, 2021.

[20] J. Zhu, Y. Zhao, and J. Pei, "Progressive kernel pruning based on the information mapping sparse index for cnn compression," *IEEE Access*, vol. 9, pp. 10 974–10 987, 2021.

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[22] Y. Gong, H. Cai, H. Wu, W. Ge, H. Yan, Z. Wang, L. Shi, and B. Liu, "Quality driven systematic approximation for binary-weight neural network deployment," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[23] A. J. Redfern, L. Zhu, and M. K. Newquist, "Bcnn: A binary cnn with all matrix ops quantized to 1 bit precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4604–4612.

[24] H. Zhang, Y. Shu, W. Jiang, Z. Yin, W. Zhao, and Y. Ha, "A 55nm, 0.4 v 5526-tops/w compute-in-memory binarized cnn accelerator for aiot applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 5, pp. 1695–1699, 2021.

[25] J. Huang, T. N. Kumar, H. A. Almurib, and F. Lombardi, "Commutative approximate adders: Analysis and evaluation," in *2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 2021, pp. 1–6.

面向区块链数字资产的智能合约缺陷分析

孙金磊[①] 惠战伟*[①] 王小娟[①] 曹旭[①] 宋爽[①]
(军事科学院 北京 100091)

摘 要：智能合约是运行在区块链上的程序，由于编程语言设计不完善、开发人员编码不规范等问题，智能合约难以避免地存在着缺陷，影响着智能合约的质量和数字资产的安全。深入分析智能合约缺陷的成因和影响，对于检测智能合约缺陷、保障数字资产安全具有重要的指导意义和积极的推动作用。本文首先综述了现有智能合约缺陷分类相关研究，分析了当前研究在区块链数字资产安全方面存在的不足；然后从稀缺性和可控性的角度给出了缺陷的描述和示例，分析了缺陷产生的原因和潜在影响，并对缺陷信息进行了总结；最后，提出了一种面向区块链数字资产的智能合约模糊测试框架，并展望了未来的研究方向。

关键词：区块链；智能合约；数字资产；缺陷检测

Analysis of the Defect of Smart Contracts for Blockchain Digital Assets

Jinlei Sun[1], Zhanwei Hui[1,*], Xiaojuan Wang[1], Xu Cao[1], and Shuang Song[1]
[1]Academy of Military Science, Beijing, China
jinleisun@yeah.net, hzw_1983821@163.com, rgldmy@126.com, 13911761719@139.com, 48796447@qq.com
*corresponding author

Abstract: Smart contracts are programs that run on the blockchain. Due to imperfect programming language design and unstandardized coding by developers, smart contracts are inevitably flawed, affecting the quality of smart contracts and the security of digital assets. In-depth analysis of the causes and effects of smart contract defects is an important guide and positive impetus to detecting smart contract defects and securing digital assets. We first review the existing research related to the classification of smart contract defects and analyse the shortcomings of current research in the security of blockchain digital assets. Then we give descriptions and examples of defects from the perspectives of scarcity and controllability, analyse the causes and potential impacts of defects, and summarise the defect information. Finally, we propose a fuzzy testing framework for smart contracts for blockchain digital assets, and future research directions are foreseen.

Key words: Blockchain; Smart contracts; Digital assets; Defect detection

1 引言

区块链是一种按照时间顺序将数据区块以链式结构组合而成的特定数据结构，具有去中心化、防篡改等特点。随着比特币、以太坊等区块链的迅速兴起，区块链数字资产逐渐被人们关注。区块链数字资产（Blockchain Digital Asset, BDA）是基于区块链技术发行、登记、存储、持有、转让或交易的新型无形资产[1]。在具备传统资产的经济价值属性的同时，BDA 借助区块链技术实现了去中心化、匿名和防篡改等特性，为数字资产的创设、发行、交易和管理等提供了新的范式。智能合约是运行在区块链上的程序，其中包含了预定义的状态和转换规则、执行条件以及执行逻辑等[2]。智能合约为 BDA 的交易和管理提供了一种可编程途径，大大增加了 BDA 的灵活性，进一步促进了 BDA 的发展与应用。以太坊[3]是首个支持智能合约的区块链平台，大量的 BDA 在以太坊区块链上发行，成为了区块链和数字经济的重要组成部分[4]。

然而，由于编程语言设计不完善、开发人员编码不规范等问题，智能合约难以避免地存在着缺陷。根据文献[6]的调查，在其研究的 3759 份智能合约中有 3686 份含有安全缺陷。区块链防篡改的特性使得智能合约一旦在链上部署后就难以修改，导致智能合约中的缺陷无法以打补丁的方式更正，严重威胁着区块链及资产安全。深入分析智能合约缺陷的成因和影响，能够使研究人员和从业人员更充分地了解智能合约缺陷及相关测试技术，对于提高智能合约质量和保障区块链安全均具有指导意义和推动作用。

本文首先对公开发表的智能合约缺陷分析相关研究进行了综述，分析了现有智能合约测试技术和缺陷分析存在的不足；然后针对 BDA 安全需要满足属性进行分析，定义了违反 BDA 稀缺性和可控性的智能合

*通信作者：惠战伟 hzw_1983821@163.com

约缺陷，指明了造成这两类缺陷的原因，并对每种缺陷给出了相应的示例；接着对 BDA 安全相关的稀缺性缺陷和可控性缺陷进行总结，分析了每个缺陷所在层级、潜在影响及严重程度等；最后针对影响数字资产安全的缺陷，本文提出了一种面向数字资产安全的智能合约模糊测试框架，并展望了未来研究方向。

本文的组织如下：第 2 节介绍了以太坊智能合约相关概念；第 3 节综述了智能合约缺陷相关研究；第 4 节从数字资产安全的角度分析了现有智能合约缺陷；第 5 节阐述了一种面向数字资产安全的智能合约模糊测试框架；第 6 节展望了未来研究方向。

2 以太坊智能合约相关概念

2.1 以太坊区块链

区块链是一种按照时间顺序将数据区块以链式结构组合而成的特定数据结构。当前比较流行的区块链系统有比特币 Bitcoin[1]、以太坊 Ethereum[2]、超级账本 Hyperledger[3]等。以太坊是一个开放的区块链平台，支持图灵完备的编程语言，编程人员可以通过部署智能合约来实现不同的应用。

以太坊区块链上包含两类账户：外部账户和合约账户。其中外部账户由个人创建，通过密钥控制；合约账户在合约部署时创建，由合约代码控制。每个账户均有唯一地址与之对应。以太坊上账户之间的以太币转账和合约调用需要通过交易（Transaction）完成。交易是由以太坊外部账户发起的经过加密签名的指令。从内容上看，交易是经过打包的二进制数据包，其中包含交易编号 nonce，交易金额 value，gas 单价 gasPrice，gas 上限 gasLimit，接收方地址 to，交易签名<v,r,s>，交易包含数据 data。可以形式化定义如下：

定义 1 交易（Transaction）：以太坊区块链上的一次交易 T 可以用一个 5 元组表示：

$$T := \langle from, to, value, data, conf \rangle \tag{1}$$

其中，中 *from* 表示交易发送方地址，*to* 表示交易接收方地址，*value* 表示转移的以太币数量，*data* 表示交易所包含的数据，*conf* 中包含 nonce、gasPrice、gasLimit 等交易相关配置参数。

2.2 智能合约

智能合约是运行在区块链上的程序，其中包含了预定义的状态和转换规则、执行条件以及执行逻辑等，当执行条件满足时，执行逻辑会被自动执行。目前以太坊智能合约的编程语言最为流行的是 Solidity[7]，Solidity 具有布尔型、整型、浮点型、字节数组、枚举型、结构体、映射等多种基本的变量类型。此外，Solidity 还有地址类型 address，用于表示以太坊中的地址。

Solidity 中函数定义方式为：function FunctionName(<parameter types>) {public | private | internal | external} [pure|constant|view|payable] [modifier] [returns (<return types>)]。其中 FunctionName 表示函数的 id，<parameter types>为函数输入参数的类型，modifier 为函数的修饰符。此外，对于函数的可见性、函数访问控制等定义了一系列新的关键字，函数的可见性有 public、private、internal、external 等四种关键字，访问控制有 pure、constant、view、payable 等四种关键字。

2.3 区块链数字资产

区块链的出现拓展了数字资产的形式，推动了数字资产化与资产数字化的发展，区块链数字资产（Blockchain Digital Assets, BDA）应运而生。BDA 是基于区块链技术发行、登记、存储、持有、转让或交易的新型无形资产，具有传统资产的经济价值属性。根据是否由区块链直接发行，BDA 可以分为内置货币（Build-in Currency, BIC）和合约定义资产（Contract-defined Asset, CDA）。前者是由基于区块链发行机制产生，如以太坊中的以太币 Ether；后者由链上智能合约定义，如基于 ERC20 标准的代币 Token 等。BIC 和 CDA 均可通过智能合约实现管理和交易等操作。

智能合约对于 BDA 的管理操作（后文简称操作）有发行、存入、取出、转账、销毁等。图 1 的示例展示了通过智能合约管理 BDA 的典型场景。其中，Owner 表示合约的拥有者，User A 和 User B 表示合约的普通用户。Owner 通过智能合约发行 CDA 并在合约中存入一定数量的 CDA，同时能接受合约转账的 BIC；User A 和 User B 等用户可以往合约中存入 BIC，并接收合约提供的 CDA，或者将个人账户中的

[1] https://bitcoin.org

[2] https://ethereum.org

[3] https://www.hyperledger.org

CDA 转账给合约并取回 BIC；此外，User A 和 User B 之间可通过合约互相转账 CDA。

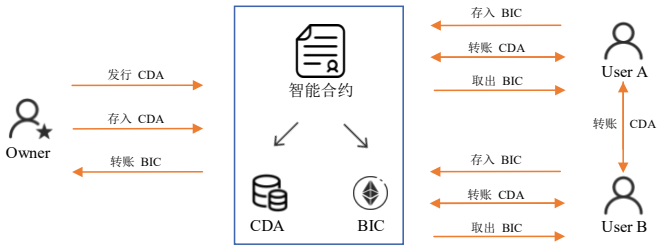


图 1 BDA 交易和管理示例

3 现有以太坊智能合约缺陷分析方法

随着研究的不断发展，研究人员对于智能合约缺陷分析的广度和深度也不断增大。研究人员从不同角度对智能合约缺陷进行了分类。当前的分类方法主要有依据缺陷产生的不同层级划分、依据缺陷通用标准划分、依据缺陷产生的影响划分、依据合约领域知识划分、以及多维度复合分析等方法。这些方法从智能合约缺陷的不同角度展开分析和分类，有助于开发人员更好的了解智能合约缺陷的特征，对智能合约的安全保障起到了一定作用。本节对现有缺陷分析和分类研究进行阐述，总结了现有的缺陷分类方法的不足。

3.1 按缺陷层级分类

智能合约的缺陷往往存在于编程语言、虚拟机或者区块链等不同层级，因此最常见的分类方法是根据引入缺陷的层级进行划分。

Atzei 等人[8]首次系统地阐述了以太坊及其高级编程语言 Solidity 的安全缺陷，将以太坊智能合约缺陷分为 Solidity、EVM 和区块链三类，并将这些缺陷与合约中的缺陷实例进行了关联。Dika 等人[9]在 Atzei 等人的分类方法础上补充了新的缺陷种类。分类得出 13 个 Solidity 层面缺陷、2 个 EVM 层面缺陷和 6 个区块链层面缺陷。同时，将缺陷划分为高严重性、非关键和警告三个严重等级。Mense 等人[10]沿用了 Atzei 和 Dika 等人分类的标准，并进一步阐述了每种缺陷的具体信息，同时分析了 Oyente 等 7 种工具在具体缺陷上的检测能力表现，最后基于 The Dao 攻击，阐述了有效避免重入缺陷发生的策略。Praitheeshan 等[11]人着眼智能合约安全问题，从智能合约的内部机制和软件安全角度出发，总结了 6 个区块链层面的缺陷，19 个软件安全相关的问题和 16 个以太坊及 Solidity 层面的缺陷。并将以太坊 Solidity 层面的缺陷与软件安全问题进行关联，分析了缺陷产生的机制，给出了对应的攻击实例。Zhou 等人[12]研究了智能合约安全的现状、普遍存在的缺陷和安全分析工具支持。该研究沿用了 Atzei 等人的三层结构分类策略，将 13 个缺陷分为 Solidity、EVM 和区块链三个层级，并将这些缺陷与 CWE 中的分类和实际攻击事件进行了关联，针对每个缺陷给出了对策。Kushwaha 等人[13]从 Solidity、EVM 和区块链三个层级进行分类，给出了缺陷的简单示例和检测工具等。Demir 等人[14]阐述了智能合约的生命周期，从合约生命周期的不同阶段分析了智能合约中编码人员需要避免的缺陷，指出了代码中的安全异味。

基于缺陷层级的分类关注于缺陷发生于区块链系统的哪个层次，能够为缺陷提供一定区分，但是缺乏对缺陷原理和影响的分析。

3.2 按通用标准分类

这一类的研究更关注通用的、标准化的分类方法。在传统程序的缺陷相关的数据库中，有一些被研究人员公认的数据库。其中较为知名的 CVE[15]，NVD[16]，CWE[17]，NIST-BF[18]等。在智能合约缺陷分析方面，常用的分类标准为 SWC[19]，SWC 是以太坊改进提案中提出的缺陷分类方案的实现。

Dingman 等人[20]指出了当前分类方法不够结构化的问题，使用 NIST 开发的缺陷框架（BF）对智能合约缺陷进行了首次的正式分类。该研究将 Tikhomirov 等人指出的 21 种智能合约缺陷分为 10 个类别，并与的安全性、功能性、开发性和操作性分类方法进行对应。Samreen 等人[21]同样关注于区块链应用层的缺陷，基于 NIST-BF 将根据已有的缺陷攻击案例分析了重入等 8 个缺陷分为了不可用（NA）、未检查的错误（UCE）、计算或转换缺陷（ARC）等 3 类，同时给出了每个缺陷对应的实例和预防方案。Staderini 等人[22]首次基于 CWE 对以太坊 Solidity 智能合约缺陷进行了系统化的分类，总结了 28 个缺陷，阐述了缺陷的细节，并给出了每个缺陷的示例。Zhang 等人[23]针对了对当前研究没有覆盖所有已知缺陷，同时缺乏详细的

缺陷检测标准的问题，基于 IEEE 软件异常标准分类[24]（IEEE Standard Classification For Software Anomalies）对 49 种缺陷进行了分类。

基于通用标准的分类方法使得智能合约缺陷的分类更加系统化和标准化，有助于研究人员更准确查找缺陷相关信息，但是对智能合约及区块链相关的知识缺少考虑，不能直观展示智能合约缺陷的特性。

3.3 按缺陷影响分类

不同类型的缺陷会对智能合约带来不同的影响。一些研究着重考虑了缺陷产生的影响，并从这一角度对现有智能合约缺陷进行了分类。

Tikhomirov 等人将 21 个缺陷分为安全问题、功能问题、运营问题、开发问题 4 个类别。其中安全缺陷会导致恶意用户攻击或利用合约，功能缺陷导致合约违反预期的功能，运营缺陷导致运行时间问题，开发缺陷会使得代码难以理解和改进。同时，该研究将这些缺陷的严重性分为了 3 个等级。Alharby 等人[25]对筛除出的 24 篇文章进行分析，根据智能合约问题的原因和影响范围将 13 个缺陷分为编码问题、安全问题、隐私问题和性能问题 4 类，并列举了已有研究中针对这些问题的解决方案。Chen 等人[26]从以太坊 StackExchange 收集了智能合约相关的帖子和 587 个现实中的智能合约，并手动分析了这些帖子与合约。总结了 20 种合约缺陷，并从缺陷对安全性、可用性、性能、可维护性和可重用性带来的负面影响对其进行系统性的分类，给出了示例和可行的解决方案。Li 等人[27]从区块链的去中心化金融（DeFi）属性出发，首次系统的分析了区块链上影响的 DeFi 生态系统的安全问题，从数据层、共识层、合约层和应用层阐述了以太坊中的缺陷。调查了现实中与 DeFi 相关的攻击事件，并给出了详细分析。

基于缺陷影响的分类方法从不同的角度分析了智能合约缺陷产生的影响，对于研究人员更好的理解和检测缺陷有一定帮助。但是这些分类方法往往不够明确，对于缺陷的影响分析不够深入，同时类别之间的界限存在模糊。

3.4 按领域知识分类

除了按照缺陷层级、通用标准等系统化的分类方法之外，一些研究从智能合约领域知识的视角对缺陷进行分类。智能合约缺陷的领域知识指的是智能合约中与缺陷相关的特定操作知识。

Chen 等人[28]从首次从智能合约的安全性、架构和可用性 3 个方面系统的分析了智能合约代码异味，为智能合约定义了未检查的外部调用、严格的余额相对、硬编码地址等 20 种代码异味，并给出了每种代码异味的示例。Khan 等人[29]指出了当前缺陷分类方法无法反应特定于智能合约知识的问题，并根据智能合约中错误操作相关的领域知识对当前的智能合约的 20 种缺陷进行分类。Groce 等人[30]关注于智能合约中潜在缺陷的可能性和关键程度，基于 Trail of Bits 执行的 23 次审计得出一共 246 个缺陷，共有 22 个不同的类型。该研究将缺陷潜在影响的严重程度分为高、中、低和信息性四个层级，同时将缺陷的暴露难度分为高、中、低三个层级。

基于智能合约领域知识的分类方法充分的考虑了缺陷中智能合约相关特性，生成的分类结果更加直观，有助于研究人员更清晰的理解这些缺陷。

3.5 多维度的缺陷分类

一些学者综合分析了已有的智能合约缺陷分类方法，指出了单一的分类方法缺乏整体性的问题，从多个维度对智能合约缺陷展开分析和分类。

Chen 等人[31]从缺陷、攻击和防御三个角度展开研究，深入分析了缺陷的根本原因、攻击后果和防御能力。该研究列举了以太坊体系结构各层共 40 种缺陷，并根据这些缺陷的存在层级、产生原因和当前状态（如已解决、可避免、开放等）等多个维度进行描述和分类。Rameder[32]收集了 54 个智能合约缺陷，并将其分为恶意环境/交易/输入、区块链/环境依赖等 10 个类别，针对每种缺陷给出了详细的描述和示例。同时，该研究对现有的 140 个智能合约分析工具进行了系统化的描述和分类，为研究人员提供了有力的支撑。Soud 等人[33]从公共代码平台和流行缺陷数据库中提取了 2143 个缺陷，进一步得出 47 个智能合约的缺陷，然后采用卡片分类的方法进行分类。从智能合约缺陷自身的角度，划分了 11 个描述缺陷错误来源的类别；从缺陷产生的影响角度，划分了 13 个描述潜在影响的类别。

多维度的缺陷分类方法从不同的角度对现有缺陷进行了充分分析，同时对多种分类方法进行关联或映射，为研究人员提供了更加全面的缺陷信息、检测工具分析以及相关数据集。

4 面向区块链数字资产的缺陷分类

智能合约作为区块链上的程序，担负着十分重要的职能——区块链数字资产的定义和管理，智能合约中的缺陷往往对 BDA 产生影响。分析缺陷和 BDA 之间的关系，了解缺陷对 BDA 的影响，对于检测缺陷和保护 BDA 具有积极作用。现有研究从缺陷层级、通用标准、缺陷影响、领域知识、多维度综合等角度给出了智能合约缺陷的不同分类，尽管这些分类方法已经十分丰富，但是仍然存在一定的局限性，主要体现为对于智能合约缺陷与 BDA 之间关系分析不足。当前对于智能合约缺陷的分析和分类方法主要关注于缺陷的代码属性，很少从 BDA 的角度考虑缺陷的原因与影响，导致智能合约的分类和分析结果存在片面性，对于 BDA 的安全保障支撑性不足。因此，需要从 BDA 安全保障的视角出发，分析智能合约在编码和管理 BDA 方面存在的不足，阐述缺陷产生的原因和造成的影响，促进 BDA 的安全保障。

4.1 区块链数字资产稀缺性缺陷

BDA 以数字化的形式存于区块链中，其经济价值的属性要求必须具备稀缺性。在经济学中，稀缺性指现实中人们在某段时间内所拥有的资源数量不能满足人们的欲望时的一种状态。对应于 BDA，稀缺性可以定义为：BDA 的供应应该被严格控制,不能够被随意复制或丢弃，增发应该是一种特权操作。不满足稀缺性的 BDA 的供应量可以被任意增加，导致 BDA 的经济价值稀释或丧失。

以太坊中对 BDA 的管理操作通过智能合约函数定义，违反稀缺性的缺陷也存在于合约函数中，主要由 3 种原因导致：

（1）发行权限管理不当：BDA 初次发行由合约的构造函数（Constructor）完成，后续增发时由特殊的铸币函数实现。构造函数一般只能在合约发布时执行，而铸币函数一般只能由合约的拥有者调用。如果这两种函数的权限管理不当，则可能导致任意用户都能增发 BDA，那么 BDA 的供应将不受控制，从而破坏其稀缺性；

（2）资产修改逻辑错误：智能合约中将 BDA 的编码为状态变量，存储于全局存储中，存入、取出、转账等交易过程都涉及到对 BDA 对应状态变量的修改，如果函数中存在错误的 BDA 修改逻辑，那么可能导致 BDA 数量变化异常，影响其稀缺性。

（3）资产数值计算错误：智能合约中 BDA 的具体数量由算术运算过程计算得出并赋值给其对应的状态变量，如果运算过程或者赋值过程发生溢出等意外错误，那么也会导致 BDA 数量发生异常的变化，影响其稀缺性。

4.1.1 发行权限管理不当

（1）错误的构造函数名（Incorrect Constructor Name, ICN）

错误的构造函数名的含义是智能合约中的构造函数名称不正确。智能合约中的构造函数是特殊函数，仅在合约创建期间调用一次，构造函数通常执行关键的特权操作。在 0.4.22 版本之前，Solidity 允许开发者使用与合约同名的函数作为构造函数，如果开发人员拼错构造函数的名称或者修改了合约名称，导致构造函数与合约名称不匹配，那么构造函数就会变为一个公共的、可以被外部调用的函数。如果构造函数中存在设定 BDA 的数量的代码，那么任意用户均可以调用该函数修改 BDA 的数量。

```
1  contract MyIco {
2      uint public total;
3      mapping ( address => uint ) balance ;
4      function Myico(uint256 initialSupply) {   // bug
5          total = initialSupply * 10;
6          balance[msg.sender] = total;
7      }
8  }
```

图 2 错误构造函数名缺陷代码示例

图 2 了 ICN 缺陷的示例代码片段。其中合约名称为 MyIco，第 4 行的构造函数接收 initialSupply 为输入参数，为合约创建者初始化代币余额为 initialSupply。然而，代码第 4 行的构造函数名称为 Myico，与合约名不一致，且可见性为默认的 public。因此，函数 Myico()可以被任意用户调用，在第 5 行重置发行量 total，并在第 6 行修改自己账户下的余额，造成代币数量可以被任意修改，违反了稀缺性。

（2）错误的函数可见性（Incorrect Function Visibility, IFV）

Solidity 中对于函数的定义有 public、private、internal、external 等 4 种不同的可见性关键字，如不设置

关键字，那么函数的可见性默认为 public，此时函数可以被外部调用。如果函数中有增加 BDA 的供应量代码，则会导致任意用户均可以增发 BDA。

```
1   contract MyIco {
2       uint public total;
3       mapping ( address => uint ) balance ;
4       address public owner;
5       function do_mint(uint256 value) {   // bug
6           require(balance[msg.sender] + value >= value );
7           require(total + value >= value );
8           balance[msg.sender] = balance[msg.sender] + value;
9           total = total + value;
10      }
11      function mint(uint256 value) {
12          require(msg.sender == owner );
13          do_mint(value); }
14  }
```

图 3 错误的函数可见性缺陷代码示例

在图 3 所示的 IFV 缺陷代码片段中，第 5 行的函数 do_mint()接收参数 value 作为输入，合约的拥有者调用第 11 行的 mint()函数时，第 13 行通过调用 do_mint()为其增发代币。do_mint()需要定义为 internal，供 mint 函数内部调用，进行增发操作。然而在第 5 行中，do_mint()函数可见性没有进行明确指定，默认为 public，可以被公开调用。导致任意用户均可给自己的账户进行增发，破坏了代币的稀缺性。

（3）无保护的资产增发（Unprotected Assets Increase, UAI）

无保护的资产增发的含义是智能合约中在 BDA 增发操作前没有进行合理的权限验证。如果没有在执行增加 BDA 的代码之前验证交易发送方的身份，那么可能导致任意用户均可以增发 BDA。

```
1   contract MyIco {
2       uint public total;
3       mapping ( address => uint ) balance ;
4       function do_mint(uint256 value) internal {
5           require(balance[msg.sender] + value >= value );
6           require(total + value >= value );
7           balance[msg.sender] = balance[msg.sender] + value;
8           total = total + value; }
9           function mint(uint256 value) public {
10          // require(msg.sender == owner );  bug
11          do_mint(value);
12      }
13  }
```

图 4 无保护的资产增发缺陷代码示例

图 4 展示了 UAI 缺陷示例代码，其中第 9 行的 mint()函数在第 11 行调用 do_mint()函数进行增发操作。虽然 do_mint()函数被定义为 internal，但是如果缺少第 10 行的权限验证语句，mint()函数将可以被任意用户调用以给自己的账户增发代币，违反了稀缺性。

4.1.2 资产修改逻辑错误

（1）错误的运算符（Incorrect Arithmetic Operator, IAO）

错误的运算符的含义是在 BDA 计算过程中使用了不正确的算术运算符。在交易过程中，BDA 数量的计算和修改依赖于函数代码中的运算符，运算符使用错误可能导致对 BDA 错误的修改。例如，在 Solidity 的 0.4.22 版本之前，在整数运算中使用编译器不会报错。

```
1   contract MyIco {
2       uint public total;
3       mapping ( address => uint ) balance ;
4       function transfer(address rec, uint256 amount) public {
5           require(rec != address(0) );
6           require(balance[msg.sender] >= amount);
7           balance[msg.sender] =- amount;    // bug
8           balance[rec] =+ amount;    // bug
9       }
10  }
```

图 5 错误的运算符缺陷示例代码

在图 5 所示的 IAO 缺陷示例代码中，第 4 行 transfer()函数接收地址 rec 和 uint256 的 amount 为输入参数，用于为调用者 msg.sender 向接收者 rec 转账数量为 amount 的代币。代码中第 7 行和第 8 行计算并修改代币数量时，使用了运算符“=+”和“=-”。导致调用者 msg.sender 和接收者 rec 的代币数量均被重新设置，造成代币总量发生异常变化，违反稀缺性。

（2）意外的资产复制（Unexpected Asset Copy, UAC）

意外的资产复制的意味着智能合约中包含能够复制 BDA 的代码。Solidity 中将资产编码为普通的数据类型，如 ERC20 中的 Token 为无符号整型 uint256，这些数据类型能够被复制，如果在资产转账的逻辑中编码不当则可能使其代表的资产发生复制，导致 BDA 稀缺性受到影响。

图 6 展示了 UAC 缺陷示例代码，其中第 7 行的 transfer()函数接收两个地址_from 和_to 和一个 uint256 类型的参数_value 作为输入，将数量为_value 代币从_from 转移给_to。然而，如果_from 和_to 为同一个账户，在第 15 行 newFromVal 的值将被 newToVal 覆盖。即并未从_from 账户减掉_value，反而加了_value，最终导致了无限增发，破坏了 BDA 的稀缺性。

```
1   contract Token {
2       mapping ( address => uint ) public balances ;
3       constructor() public payable{
4           owner = msg.sender;
5           balances[this] = 300;
6       }
7       function transfer(address _from, address _to, uint256 _value)
8           external returns (bool) {
9           require(_to != address(0));
10          require(_value > 0 && _value <= balances[_from]);
11          uint256 newToVal = balances[_to] + _value;
12          assert(newToVal >= balances[_to]);
13          uint256 newFromVal = balances[_from] - _value;
14          balances[_from] =  newFromVal;
15          balances[_to] = newToVal;          // bug
16          uint256 temp = _value.mul(profit);
17          changeProfit[_from] = changeProfit[_from].add(temp);
18          received[_to] = received[_to].add(temp);
19      return true;
20    }
21  }
```

图 6 意外的资产复制缺陷示例代码

4.1.3 资产数值计算错误

（1）算术溢出（Arithmetic Overflow, AO）

算术溢出是智能合约整数溢出缺陷里面最常见的一类，主要是由进行算术运算时未进行边界检查导致，可以分为上溢、下溢。整数变量的赋值超出其表示范围的上限的溢出称为上溢，造成上溢的主要算术运算操作为加法和乘法。例如，在图 7 中所示算术上溢缺陷示例代码片段中，第三行函数 batchTransfer()接收地址数组 rec 和数量_value 作为输入，为 rec 中的每个地址转账_value 数量的代币。然而第 5 行中计算 amount 的代码中，当输入参数_value 较大时，uint256(cnt)*_value 可能发生溢出，使得实际计算结果 amount 较小，从而绕过第 7 行的检查。第 10 行为每个地址增加_value 的余额，而 msg.sender 只减少较少 amount 数量的余额，造成 BDA 总量发生意外的增发。

```
1   contract Token {
2       mapping(address => uint256) public balances;
3       function batchTransfer(address[] rec, uint256 _value) {
4           uint cnt = rec.length;
5           uint256 amount = uint256(cnt) * _value;   // bug
6           require(cnt > 0 && cnt <= 20);
7           require(_value > 0 && balances[msg.sender] >= amount);
8           balances[msg.sender]=balances[msg.sender].sub(amount);
9           for (uint i = 0; i < cnt; i++) {
10              balances[rec[i]]=balances[rec [i]].add(_value);
11          }
12      }
13  }
```

图 7 算术上溢缺陷示例代码

整数变量的赋值超出其表示范围的下限，从而发生超出下边界的溢出称为下溢，造成下溢的主要算术运算操作为减法。例如，在图 8 所示的算术下溢代码示例片段中，第 7 行 transfer()函数接收地址_to 和整数_value 作为输入，从交易发起方 msg.sender 转账_value 数量的代币到_to 地址下。然而当_value 大于 balances[msg.sender]，第 9 行的 require 语句因发生下溢而被满足，第 10 行发生下溢，使得 balances[msg.sender]变为一个较大的值，即 BDA 总量发生意外的增加。

```
1   contract Token {
2       mapping(address => uint) public balances;
3       uint public totalSupply;
4       function Token(uint _initialSupply) {
5           balances[msg.sender] = totalSupply = _initialSupply;
6       }
7       function transfer(address _to, uint _value)
8           public returns (bool) {
9           require(balances[msg.sender] - _value >= 0);  // bug
10          balances[msg.sender] -= _value;  // bug
11          balances[_to] += _value;
12          return true;
13      }
14  }
```

图 8 算术下溢缺陷示例代码

（2）截断溢出（Truncation Overflow, TO）

程序中将位宽大的整数赋值给位宽小的整数，从而导致的溢出被称为截断溢出。Solidity 中的整型从 8 位到 256 位，以 8 位为间隔递增。将位宽大的整数赋值给位宽小的整数时会发生截断，变成一个较小的值。在图 9 的 TO 缺陷示例代码片段中，第 4 行中将原本是 uint256 位的 msg.value 截断为 32 位赋值给 value 时会发生截断溢出，导致 balance[msg.sender]的值比 msg.value 小很多，造成 BDA 数量的意外减少。

```
1   contract Token {
2       mapping(address => uint32) public balances;
3       function buy() public payable{
4           value = uint32(msg.value);
5           require(balances[msg.sender] + value >= value);
6           balances[msg.sender] += uint32(msg.value) ;  // bug
7       }
8   }
```

图 9 截断溢出缺陷示例代码

（3）符号转化溢出（Signed Overflow, SO）

符号转化溢出是由于目标变量类型和被赋予的值类型不同造成，包括两种情况：一种是有符号整数到无符号整数的转化溢出：有符号整数的最高位为符号位，如果将一个负数赋值给同长度的无符号整型，则会溢出为一个较大的正数；另一种是无符号整数转化为有符号整数时的溢出：如果将一个较大的无符号整数赋值给同长度的有符号整数，则可能会转化为一个负数从而导致缺陷。

```
1   contract Token {
2       mapping(address => uint) public balances;
3       function transfer(address _to, int _value) public{
4           value = uint(_value);  // bug
5           require(balances[msg.sender] - value >= value);
6           balances[msg.sender] -= value;
7           balances[_to] += value;
8   }
```

图 10 符号转化溢出缺陷示例代码

在图 10 所示的 SO 缺陷代码片段中，第 3 行中函数 trasfer()输入参数_value 为 int 类型，在后续使用时转化为 uint 类型，可能导致 balances[_to]加上一个较大的值，导致 BDA 的意外增加。

4.2 区块链数字资产可控性缺陷

上一节中对 BDA 稀缺性及相关缺陷做了阐述。除了稀缺性以外，BDA 经济价值的属性还要求必须具备可控性。BDA 的可控性可以定义为：BDA 的应该由其所有者严格控制。如果可控性被破坏，BDA 能够受其拥有者以外的区块链参与者影响，产生非预期操作，造成意外的经济损失。

以太坊智能合约中违反可控性的缺陷主要由 3 种原因导致：

（1）操作权限管理不当：BDA 作为一种新型资产，具有传统资产的所有权等属性。这要求 BDA 的发布、存入、取出、转账等操作均需要特殊权限才能执行。如果智能合约对于 BDA 操作权限管理不当，则可能导致 BDA 能够被非授权人员操作，影响 BDA 的可控性。

（2）未检查调用返回值：智能合约的本质是计算机程序，基于智能合约的管理 BDA 依赖于合约代码的正常执行。合约代码执行过程中往往出现 call()或者 send()调用，调用的成功与失败通过返回的布尔值体现，不同的返回值结果需要对应于不同的代码执行逻辑。如果智能合约中对返回值进行检查和判断，则可能导致合约执行错误的 BDA 操作逻辑，影响 BDA 的可控性。

（3）非确定性因素：智能合约中的非确定性因素指智能合约中导致非确定执行结果的因素，主要包括未知的函数调用、受外部影响的执行、系统属性依赖和交易顺序依赖等。例如，以太坊区块链上的合约调用交易具有共享存储的特性，这种特性类似于线程访问共享内存中的并发对象，会导致智能合约中状态变量的修改结果依赖于交易的执行顺序；此外，合约代码中可能包含对时间戳、区块号等区块链全局变量的使用，这些全局变量的具体值依赖于区块被都打包的时间。因此，智能合约中的非确定性因素会导致智能合约的执行结果能够被外部干预，从而控制数字资产操作，对 BDA 的可控性产生影响。

4.2.1 操作权限管理不当

（1）使用 tx.origin 授权（Authorize with tx.origin, AWT）

使用 tx.origin 授权的含义是在检测条件中使用 tx.origin 代替 msg.sender 进行权限判断。在以太坊交易过程中，存在着跨合约调用的场景。例如在一次交易中，交易发起方 A 调用智能合约 B 的函数 b()，b()中又调用了合约 C 的函数 c()，即存在调用链“A—>B—>C”。以太坊智能合约编程语言 Solidity 中，msg.sender 用于表示当前调用消息的发送方地址，其类型为 address；tx.origin 用于表示交易的原始发送方地址，其类型也为 address。例如，在“A—>B—>C”的调用链中，对于合约 C 来说，交易的原始发送方 tx.origin 为 A，当前调用消息的发送方 msg.sender 为 B。

如果在 BDA 操作函数中使用 tx.origin 进行访问授权，则可能导致函数的直接调用者利用 tx.origin 通过检查条件，造成对 BDA 操作的越权使用。在图 11 所示的 AWT 缺陷示例代码中，第 9 行 transfer()函数接收地址 to 和无符号整数_value 作为输入，在第 10 行进行权限验证，并在 11-14 行为地址 to 增加_value 数量的余额 balances。然而，第 10 行进行权限验证时，使用了 tx.origin 而非 msg.sender。如果攻击者通过 ower 发起的交易调用 transfer()，则能够通过第 10 行的验证，进而越权转移 owner 拥有的资产，违反了 owner 对 BDA 的可控性。

```
1   contract Token {
2       mapping(address => uint) public balances;
3       uint public totalSupply;
4       address public owner;
5       function Token(uint _initialSupply) {
6           balances[msg.sender] = totalSupply = _initialSupply;
7           owner = msg.sender;
8       }
9       function transfer(address to, uint _value) external{
10          require(tx.origin == owner);   // bug
11          require(balances[to] + _value >= _value);
12          require(balances[owner] - _value <= _value);
13          balances[to] += _value;
14          balances[owner] -= _value;
15      }
16  }
```

图 11 使用 tx.origin 授权缺陷示例代码

（2）无保护自毁（Unprotected Selfdestruct, USD）

无保护自毁的含义是在智能合约的自毁操作之前没有设置严格的权限控制。虽然智能合约在链上部署后无法修改，但是 Solidity 内建的自毁函数 selfdestruct()可以将所在合约销毁。合约被销毁意味着合约中定义和管理的 BAD 会一并被销毁，因此需要十分严格地限制自毁函数 selfdestruct()的调用权限（通常应该限制为只有合约的拥有者可以调用）。如果合约中缺乏对自毁函数权限的调用保护，那么任何用户都能够销毁，导致合约中的 BDA 一并被销毁，影响其可控性。

图 12 展示了 USD 缺陷示例代码，其中第 9 行的 suicide()函数是 external 可见性，可以被外部调用。在第 11 行调用自毁函数 selfdestruct()进行自毁操作。而自毁函数调用前缺乏正确的权限限制，例如验证调用者是否为函数的拥有者（即第 10 行注释内容）。同时，suicide()函数为 external，意味着该函数能够被外部调用，因此任意地址都能够执行自毁函数，销毁 Token 合约及其中的代币 Token。

```
1   contract Token {
2       mapping(address => uint) public balances;
3       uint public totalSupply;
4       address public owner;
5       function Token(uint _initialSupply) {
6           balances[msg.sender] = totalSupply = _initialSupply;
7           owner = msg.sender;
8       }
9       function suicide() external{
10          // require(msg.sender == owner);  bug
11          selfdestruct(msg.sender);
12      }
13  }
```

图 12 无保护自毁缺陷代码示例

（3）无保护的以太币发送（Unprotected Ether Sending, UES）

无保护的以太币发送的含义是合约中在发送以太币的代码前没有设置严格的权限控制。除了自毁函数的权限需要严格保护外，带有以太币取款操作的合约函数（下文简称取款函数）也应该被保护。通常来看，合约会给之前存入以太币的地址或者是合约的所有者发送以太币。如果合约中没有对取款函数的权限进行合理的限制，那么可能造成恶意用户越权给发送以太币，即存在无保护的以太币发送缺陷。

图 13 展示了 UES 缺陷示例代码，其中第 3 行的 payout()接收地址列表 recipients 和无符号整数列表 amounts，在第 5-6 行向每个地址 recipients 发送 amounts 中对应数量的以太币。payout()函数是默认的 public 可见性，因此可以被外部调用。调用者只需调用该函数并传入相应的参数，即可向指定地址发送以太币。破坏了合约拥有者对以太币的可控性。

```
1   contract Token {
2   …
3       function payout ( address [] recipients , uint256 [] amounts ) {
4           require ( recipients.length == amounts.length );
5               for ( uint i = 0; i < recipients.length ; i ++) {
6                   recipients[i].send( amounts[i]);
7               }
8       }
9   }
```

图 13 无保护以太币取款缺陷代码示例

（4）资产锁定（Locked Asset, LA）

该资产锁定也被称为贪婪合约，即智能合约只能存入 BDA（如以太币），而不能取出 BDA。造成这一缺陷的原因是智能合约中只有接受 BDA 的函数而缺少发送 BDA 的函数，或者发送 BDA 的函数权限不可达。如果合约中存在 LA 缺陷，那么用户存入 BDA 后将无法取出。

```
1   contract Token {
2       mapping(address => uint) public balances;
3       uint public totalSupply;
4       address public owner;
5       function Token(uint _initialSupply) {
6           balances[msg.sender] = totalSupply = _initialSupply;
7           owner = msg.sender; }
8       function suicide() external{
9           // require(msg.sender == owner);  bug
10          selfdestruct(msg.sender); }
11  }
```

图 14 资产锁定缺陷代码示例

智能合约在链上部署后无法修改，但是 Solidity 内建的自毁函数 selfdestruct()可以将所在合约销毁。合约被销毁意味着合约中定义和管理的 BAD 会一并被销毁，因此需要十分严格地限制自毁函数 selfdestruct()的调用权限（通常应该限制为只有合约的拥有者可以调用）。如果合约中缺乏对自毁函数权限的调用保

护，那么任何用户都能够销毁，导致合约中的 BDA 一并被销毁，影响其可控性。

图 14 展示了 USD 缺陷示例代码，其中第 8 行的 suicide()函数是 external 可见性，可以被外部调用。在第 10 行调用自毁函数 selfdestruct()进行自毁操作。而自毁函数调用前缺乏正确的权限限制，例如验证调用者是否为函数的拥有者（即第 9 行注释内容）。同时，suicide()函数为 external，意味着该函数能够被外部调用，因此任意地址都能够执行自毁函数，销毁 Token 合约及其中的代币。

（5）错误的构造函数名（Incorrect Constructor Name, ICN）

构造函数中除了规定代币的初始数量之外，也常常对合约的拥有者等特殊权限进行初始化。与对稀缺性的影响类似，错误的构造函数名将导致构造函数能够被外部调用。如果构造函数内部包含对合约拥有者字段的赋值语句，那么调用者可以修改合约的拥有者地址，造成智能合约所有权的意外变化，从而影响 BDA 的可控性。

```
1  contract MyIco {
2      uint public total;
3      mapping ( address => uint ) balance ;
4      address public owner;
5      function Myico() {   // bug
6          owner = msg.sender;
7      }
8  }
```

图 15 错误构造函数名缺陷影响 BDA 可控性的代码示例

图 15 展示了 ICN 缺陷影响可控性的示例代码片段。其中合约名称为 MyIco，第 4 行定义了合约的拥有 owner，并在第 5 行的构造函数中进行初始化。然而，代码第 5 行的构造函数名称为 Myico，与合约名不一致，且没有设置函数可见性，默认为公开。因此，函数 Myico()可以被任意用户调用，在第 6 行重置合约拥有者的地址，造成合约的所有者改变，从而可能影响 BDA 的可控性。

4.2.2 未检查调用返回值

（1）未检查的调用返回值（Unchecked Return Value, URV）

未检查的调用返回值的含义是合约代码中在进行 call/send()等调用时没有检查调用返回值为 True 还是 False。Solidity 中，一个合约可以通过两种途径调用另一个合约：一是直接引用被调合约的实例，二是通过 send、call、delegatecall 等方法进行代码的调用。这两种调用方式针对异常的处理是不同的。第一种调用方式发生异常时，异常会被自动回传至调用者，交易过程回滚；第二种调用方式发生异常时，会有产生一个返回值 False 给调用者。如果调用合约中没有检查调用结果的代码，可能会导致 BDA 的越权操作。

```
1  contract Token {
2      mapping(address => uint) public balances;
3      function withdraw(uint _value) public {
4      require(balances[msg.sender] - _value >= balances[msg.sender]);
5      msg.sender.call.value(_value);   // bug
6      balances[msg.sender] -= _value;
7      }
8  }
```

图 16 未检查的调用返回值缺陷代码示例

如图 16 所示，第 3 行的 withdraw()函数可以接收参数_value 作为输入，在第 5 行为调用者通过 call.value()取款_value 的以太币，同时在第 6 行减去相应的余额。然而，如果第 5 行的 call.value(_value)执行失败，同时代码中没有检查调用的返回值，调用者取款失败的同时被减去了_value 的余额，影响了调用者对 BDA 的可控性。

4.2.3 非确定性因素

（1）未知的外部调用（Call to Unknown, CTU）

未知的外部调用的含义是合约代码中存在着无法确定行为的跨合约函数调用。Solidity 中存在着跨合约函数调用的场景。当执行到跨合约函数调用时，以太坊矿工会暂停当前合约的执行，并切换到被调用合约函数的执行中。然而，被调用合约的状态是非确定的，合约中定义的代码和变量等具体内容受该合约所执行的交易影响，导致该外部调用所产生的结果能够被外部操控。经过精心设计的被调用合约及被调用函数可能存在对 BDA 的越权操作，违反 BDA 的可控性。

```
1   contract SimpleDAO {
2       mapping ( address => uint ) balance;
3       ...
4       function withdraw (uint amount) {
5           if (balance[msg.sender] >= amount) {
6               require(msg.sender.call.value(amount)());  // bug
7               balance[msg.sender] -= amount;
8           }
9       }
10  }
11  contract Attack{
12      SimpleDAO public dao = SimpleDAO(0x354...);
13      ...
14      function() { dao.withdraw(); }
15          ...
16  }
```

图 17 未知的外部调用缺陷代码示例

重入是 CTU 缺陷的一个典型场景，其触发原因是发送以太币时发送方会触发接收方的 fallback 函数，而 fallback 函数的具体内容未知，恶意的用户可以构造能够再次调用发送方转账的 fallback 函数，以实现“重新进入”。例如，在图 17 所示的代码中，SimpleDAO 合约第 4 行定义了取款函数 withdraw()，能够为调用者取出 amount 数量的以太币。然而，第 6 行的 call()函数在转移以太币时会将交易剩余的 gas 全部转发，使得被调函数有较多的 gas 用来执行操作。第 11 行构造了一个恶意的外部合约 Attack，其中第 14 行定义的 fallback 函数中包含对 withdraw()函数的再次调用。攻击者可以先存入一定的以太币，然后通过调用 withdraw()函数取出以太币，withdraw()函数调用在第 6 行时会发生以太币给攻击者，触发攻击者的第 14 行的 fallback 函数，攻击者在第 7 行 balance 值修改前可以再次调用 withdraw()函数，从而通过第 5 行的检查取出以太币。这个调用将会循环多次至被攻击者的以太币被取出至最低、调用栈满或者 gas 耗尽。

（2）受外部影响的执行（Externally Influenced Execution, EIE）

受外部影响的执行意味着函数的执行结果能够被外部因素干扰，主要表现为受 throw/revert（第 2.1.2 节）或 out-of-gas 等原因抛出异常而中断执行。智能合约中存在外部调用的代码，这些调用可能会失败或者抛出异常，导致交易回滚。如果没有对这些异常进行恰当的处理，则可能导致智能合约发生拒绝服务（Denial of service, DoS），从而影响 BDA 的可控性。例如，在 Solidity 中，使用 send 或者 transfer 来发送以太币会导致被调用方合约的 fallback 函数被执行。如果没有包含足够的 gas 来支付高消耗的 fallback 函数，那么交易将失败并导致 out-of-gas 异常，如果处理不当，则无法获取资产操作相关交易的执行结果。

例如，在图 18 所示的 EIE 缺陷代码中，第 5 行 invest()函数能够接收其他地址的以太币转账，并将该地址和转账金额计入地址列表 investors 和无符号整数列表 amounts 中。第 8 行的 payout ()函数能被合约拥有者调用，并在第 11-12 行向 investors 中的每个地址发送 amounts 中对应数量的以太币。然而，如果恶意用户通过 invest()函数注册大量冗余地址，使得 investors 的长度过长，则会导致第 11 行中的 for 循环因 gas 耗尽而执行失败，抛出异常同时回滚交易。因此，后续以太币发送操作均无法被执行，导致合约发生 DoS，影响以太币转移的可控性。

```
1   contract Token {
2       address public owner
3       address[] investors;
4       uint256[] amounts
5       function invest() public payable{
6           investors.push(msg.sender);
7           amounts.push(msg.value); }
8       function payout () public {
9           require ( msg.sender == owner );
10              require (investors.length == amounts.length );
11                  for ( uint i = 0; i < recipients.length ; i ++) {
12                      investors [i].transfer(amounts[i]); }  // bug
13      }
14  }
```

图 18 受外部影响的执行缺陷代码示例

（3）区块链属性依赖（Blockchain Property Dependency, BPD）

区块链属性依赖的含义是智能合约代码在条件检查语句或者随机数生成等代码中使用了时间戳、区块号等区块链属性值作为数据来源。区块链上的交易需要先在交易池中挂起一段时间，才能够被矿工打包和执行。因此，交易的时间戳、区块号等区块链属性值具有不确定性，依赖于矿工打包和执行交易的顺序或时间。如果在 BDA 的转账等操作的权限检查中包含区块链属性的依赖，那么该操作的执行将可能受到外部矿工控制，从而改变代码的执行逻辑，影响 BDA 的可控性。

在图 19 所示的 BPD 缺陷代码第 4 行的 fallback 函数中，参与者可以下注 10 个以太币，并且有均等机会获得所有余额。然而，如果矿工找到一个模 15 为 0 的时间戳，那么就有机会通过第 8 行的检查条件，从而执行第 9 行的转账代码，获得所有剩余的以太币。

```
1   contract Game{
2       uint public pastBlockTime;
3       constructor() public payable{};
4       function () public payable{
5           require(msg.value == 10 ether);
6           require(now != pastBlockTime);
7           pastBlockTime = now;
8           if (now % 15 == 0){
9           msg.sender.transfer(this.balance); }
10      }
11  }
```

图 19 区块链属性依赖缺陷代码示例

（4）交易顺序依赖（Transaction Order Dependent, TOD）

交易顺序依赖的含义是合约中 BDA 操作相关代码的执行结果依赖于其执行顺序。以太坊区块链的状态取决于交易被执行的顺序，而交易在被执行之前会被收集到一个交易池之中，由获得工作量证明的矿工根据交易的 gas 价格进行排序决定上传至区块中的交易。这存在两方面的问题，一方面恶意的外部账户可以通过提高交易的 gas 价格来使得交易更快的被打包上链，另一方面恶意的矿工可以无视 gas 价格决定交易进行的顺序。因此，合约中对于交易顺序有依赖的 BDA 操作代码的执行结果将会受到其他用户或者矿工的影响，即违反 BDA 的可控性。

```
1   contract Game{
2       bytes32 constant hash = 0x419ef0ef5c3f93a715;
3       constructor() public payable{};
4       function solve(string solution) public {
5           require(hash == sha3(solution));
6           msg.sender.transfer(200 ether);
7       }
8   }
```

图 20 交易顺序依赖缺陷代码示例

在图 20 所示的 TOD 缺陷代码示例中，第 4 行 solve()函数接收字符串 solution 作为输入，如果用户输入的 solution 的哈希值等于第 2 行定义的 hash，就可以得到 200 个以太币的奖励。如果交易池被攻击者监视，攻击者在验证答案的正确性之后提交了 gas 价格更高的交易，那么矿工在打包时可能会优先选择攻击者的交易执行，导致奖励的以太币被攻击者“抢走”。而如果攻击者是矿工本身，那么可以直接优先打包自己的交易，从而获得奖励的以太币。

5 总结与展望

5.1 区块链数字资产相关缺陷总结

智能合约中的缺陷存在于合约代码中，对 BDA 的稀缺性和可控性产生不同程度。影响稀缺性的缺陷会导致 BDA 发生意外的数量变化，影响可控性的缺陷会导致 BDA 出现意外的操作。前文阐述了 BDA 稀缺性和可控性缺陷的类型及具体示例，本小节根据缺陷对应的影响 BDA 的属性、所在的层级、对应的 SWC 和 CWE 分类对缺陷进行进一步的总结与分析，所得结果信息如表 1 所示。

（1）缺陷对应的 BDA 安全属性

根据缺陷对应的 BDA 安全属性，分为违反稀缺性和违反可控性。具体含义在前文已经给出，此处不再赘述。

（2）缺陷产生的层级

Atzei 等人首次将以太坊智能合约缺陷分为语言层、虚拟机层和区块链层三个层级，并在后续工作中被多次沿用。本文中也将 BDA 稀缺性和可控性缺陷映射到所在层级，与现有工作形成补充。

（3）CWE 分类

CWE 是社区开发的常见软件安全缺陷列表，可以作为缺陷识别、缓解和预防工作的基准，主要目的是帮助开发和测试人员在软件上限前，使用工具来识别、发现和修复计算机软件中的缺陷。将智能合约的 BDA 稀缺性和可控性缺陷映射到 CWE 分类有助于研究人员更好的理解相应缺陷。

（4）SWC 分类

SWC 是以太坊改进提案中提出的缺陷分类方案的实现，并在智能合约测试相关研究中被广泛认可。SWC 与 CWE 中描述的术语和结构保持一致，每个 SWC 缺陷都有一个标识符、弱点标题、CWE 关联父级和相关的代码示例列表。本章所描述的缺陷也于 SWC 之间存在交叉，因此给出与本章缺陷相关联的 SWC 分类，以便增强缺陷的可理解性。

表 1 BDA 稀缺性和可控性缺陷信息

序号	属性	类别	名称	层级	SWC 分类	CWE 分类
1	稀缺性	发行权限管理不当	错误的构造函数名	语言层	SWC-118	CWE-665：初始化不正确
2			错误的函数可见性	语言层	SWC-100	CWE-284：访问控制不当
3			无保护的资产增发	语言层	-	CWE-284：访问控制不当
4		资产修改逻辑错误	意外的资产复制	语言层	-	CWE-20：输入控制不当
5			错误的运算符	语言层	-	CWE-480：使用不正确的运算
6		资产数值计算错误	算术溢出	语言层	SWC-101	CWE-682：计算错误
7			截断溢出	语言层	SWC-101	CWE-682：计算错误
8			符号转化溢出	语言层	SWC-101	CWE-682：计算错误
9	可控性	操作权限管理不当	使用 tx.origin 授权	语言层	SWC-115	CWE-284：访问控制不当
10			无保护自毁	语言层	SWC-106	CWE-284：访问控制不当
11			无保护以太币发送	语言层	SWC-105	CWE-284：访问控制不当
12			资产锁定	语言层	-	CWE-691：控制流管理不足
13			错误的构造函数名	语言层	SWC-118	CWE-665：初始化不正确
14		未检查调用返回值	未检查的调用返回值	语言层	SWC-104	CWE-703:异常情况检查不当
15		非确定性因素	未知的函数调用	语言层	SWC-107	CWE-691：控制流管理不足
16			受外部影响的执行	语言层	SWC-113、SWC-126	CWE-691：控制流管理不足
17			区块链属性依赖	区块链	SWC-116、SWC-120	CWE-668：资源暴露于错误领
18			交易顺序依赖	区块链	SWC-114	CWE-691：控制流管理不足

在如表 1 中。其中第 1 列为缺陷序号；第 2 列为缺陷影响的 BDA 属性，分为稀缺性和可控性；第 3 列为根据缺陷产生原因划分的缺陷类别；第 4 列为缺陷的具体名称，该名称也可以反应产生出缺陷的详细原因；第 5 列为缺陷产生的层级；第 6 列为缺陷对应的 SWC 分类；第 7 列为缺陷对应的 CWE 分类。从缺陷产生的层级可以看出，影响数字资产的因素主要存在于语言层（18 个中的 16 个），这也印证了开展智能合约测试对于保障 BDA 安全十分重要。

5.1 未来研究展望

（2）基于变异测试的智能合约测试评价方法

智能合约作为区块链上的程序，一旦上链发布后将难以修改，合约中存在的缺陷将无法通过打补丁的方式进行修复，从而对 BDA 安全产生持续性的影响。因此，需要在上链前对智能合约进行充分的测试。当前的方法能够分析测试用例集合对合约整体代码的测试是否充分，但是不能很好的评价测试用例在 BDA 安全缺陷检测方面的效果。因此，有必要研究面向数字资产安全的智能合约测试评价方法，以更好地衡量智能合约中数字资产安全测试的充分性。

本部分可以研究基于变异测试的智能合约测试评价方法。变异测试是一种基于缺陷注入的测试评价方法。在进行智能合约变异测试时，需要先设计变异算子。本部分内容可以根据研究本文中总结的 BDA 安全缺陷，分析缺陷模式并定义变异算子，模拟合约中影响 BDA 稀缺性和可控性的缺陷，从而更充分的评估测试方法在检测 BDA 安全缺陷方面的效果。

（3）面向数字资产安全的智能合约测试判定方法

智能合约测试过程中需要准确的测试预言进行测试判定。智能合约缺陷会影响合约代码的执行，从而对 BDA 安全产生影响，例如造成 BDA 的增发或者越权访问等。当前的测试判定方法主要基于缺陷的代码模式进行测试预言的定义，存在一定的误报。因此，有必要从对 BDA 安全的角度出发构造智能合约测试预言，以更精准地检测 BDA 相关的缺陷，保障 BDA 的安全。

本部分可以研究面向数字资产安全的智能合约测试判定方法。一方面，分析交易执行过程中 BDA 需要满足的不变关系，可以构造能够检测 BDA 安全缺陷的测试预言，以发现引起 BDA 数量异常变化的稀缺性和可控性缺陷。另一方面，对交易执行轨迹进行分析，可以提出缺陷类型对应的轨迹特征作为测试预言，对智能合约测试预言进行补充，以检测未违反不变量的部分可控性缺陷。

（4）基于资产流引导的智能合约模糊测试方法

智能合约的本质是计算机程序代码，由于语言特性、人员疏忽等多种因素的影响，智能合约中难以避免地存在缺陷。因此需要通过完整的测试方法检测其中的缺陷，保障智能合约质量和数字资产安全。现有智能合约测试技术在构造测试用例时主要依据合约的代码特征（如代码结构和代码覆盖率等），缺乏对合约中 BDA 相关逻辑的关注，导致生成的测试用例缺乏针对性，不能优先、快速、准确发现影响 BDA 的缺陷，降低了测试效果。因此，有必要从 BDA 安全需求出发，生成智能合约测试用例、构建智能合约测试框架，以高效地检测智能合约中影响 BDA 安全的缺陷。

本部分可以研究基于资产流引导的智能合约模糊测试方法。智能合约中的数字资产一般由某种数据结构编码，作为区块链的全局变量存于全局存储中。在进行智能合约测试时，首先根据 BDA 的操作码特征识别合约中编码 BDA 的变量，确定合约中对 BDA 进行操作的合约函数；然后通过控制流和数据流等代码分析技术，获取函数之间对 BDA 的控制流关系，构建资产流图，并引导生成初始输入；接着在 EVM 执行环境中执行测试用例，并根据定义的测试预言判断测试是否通过；最后对交易执行轨迹进行分析，得出交易产生的 BDA 数据依赖和代码覆盖等信息，结合遗传算法迭代更新测试用例，直到达到预定的测试目标。

6 结语

智能合约常用于管理区块链上的数字资产，其安全性十分重要。分析智能合约中的缺陷可以为智能合约测试和质量保证提供有效参考。本文针对现有智能合约缺陷分析相关研究进行了梳理，分析了现有缺陷分析和分类相关研究，指出了现有研究中缺乏对数字资产分析的不足。进而，本文从区块链数字资产稀缺性和可控性的角度出发，分析了区块链数字资产安全缺陷的产生原因和潜在影响，给出了相应的分类和示例。最后，本文对智能合约缺陷进行了进一步总结，并提出了 4 点未来值得研究的方向。

参 考 文 献

[1] 司晓. 区块链数字资产物权论[J]. 探索与争鸣, 2021(12):80-90.

[2] 欧阳丽炜，王帅，袁勇，等. 智能合约: 架构及进展[J]. 自动化学报，2019, 45(3): 445-457.

[3] Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1-32.

[4] Ethereum. [EB/OL]. (2018-03-21) [2023-06-01]. https://github.com/ethereum/EIPs

[5] Cryptoslate. Ethereum Market Cap. [EB/OL]. (2018-08-16) [2023-06-01]] https://cryptoslate.com/coins/ethereum.

[6] Nikolić I, Kolluri A, Sergey I, et al. Finding the greedy, prodigal, and suicidal contracts at scale[C]//Proceedings of the 34th AnnualComputerSecurity Applications Conference. 2018: 653-663.

[7] Dannen C. Introducing Ethereum and solidity[M]. Berkeley: Apress, 2017.

[8] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (sok)[C]//Proceedings of International conference on principles of security and trust. 2017: 164-186.

[9] Dika A, Nowostawski M. Security vulnerabilities in ethereum smart contracts[C]//Proceedings of 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data. 2018: 955-

962.

[10] Mense A, Flatscher M. Security vulnerabilities in ethereum smart contracts[C]//Proceedings of the 20th international conference on information integration and web-based applications & services. 2018: 375-380.

[11] Praitheeshan P, Pan L, Yu J, et al. Security analysis methods on ethereum smart con-tract vulnerabilities: a survey[J]. arXiv preprint arXiv:1908.08605, 2019.

[12] Zhou H, Milani Fard A, Makanju A. The State of Ethereum Smart Contracts Security: Vulnerabilities, Countermeasures, and Tool Support[J]. Journal of Cybersecurity and Privacy, 2022, 2(2): 358-378.

[13] Kushwaha S. S, Joshi S, Singh D, et al, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," in IEEE Access, vol. 10, pp. 6605-6621, 2022.

[14] Demir M, Alalfi M, Turetken O, et al. Security smells in smart contracts[C]//Proceedings of 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Com-panion. 2019: 442-449.

[15] Mitre. CVE[EB/OL]. (2020-12-11) [2022-09-16] https://cve.mitre.org/cve/

[16] CISA. NVD[EB/OL]. (2022-08-11) [2022-09-16] http://nvd.nist.gov/

[17] Mitre. CWE[EB/OL]. (2022-09-13) [2022-09-16] http://cwe.mitre.org/

[18] NIST. BF [EB/OL]. (2022-08-14) [2022-09-16] https://samate.nist.gov/BF/

[19] Ethereum. Swcregistry [EB/OL]. (2020-05-19) [2022-09-16] https://swcregistry.io/

[20] Dingman W, Cohen A, Ferrara N, et al. Defects and vulnerabilities in smart contracts, a classification using the NIST bugs framework[J]. International Journal of Networked and Distributed Computing, 2019, 7(3): 121-132.

[21] Samreen N F, Alalfi M H. A survey of security vulnerabilities in ethereum smart con-tracts[J]. arXiv preprint arXiv:2105.06974, 2021.

[22] Staderini M, Palli C, Bondavalli A. Classification of ethereum vulnerabilities and their propagations[C]//Proceedings of 2020 Second International Conference on Blockchain Computing and Applications. 2020: 44-51.

[23] Zhang P, Xiao F, Luo X. A framework and dataset for bugs in ethereum smart con-tracts[C]//Proceedings of 2020 IEEE International Conference on Software Maintenance and Evolution. 2020: 139-150.

[24] ISDW Group. 1044-2009-IEEE Standard Classification for Software Anomalies[J]. IEEE, New York, 2010.

[25] Alharby M, Van Moorsel A. Blockchain-based smart contracts: A systematic mapping study[J]. arXiv preprint arXiv:1710.06372, 2017.

[26] Chen J, Xia X, Lo D, et al. Defining smart contract defects on ethereum[J]. IEEE Transactions on Software Engineering, 2020, 48(1): 327-345.

[27] Li W, Bu J, Li X, et al. Security Analysis of DeFi: Vulnerabilities, Attacks and Ad-vances[J]. arXiv preprint arXiv:2205.09524, 2022.

[28] Chen J, Xia X, Lo J G D, et al. Domain specific code smells in smart contracts[J]. arXiv preprint arXiv: 1905.01467, 2020.

[29] Khan Z A, Namin A S. Ethereum smart contracts: Vulnerabilities and their classifica-tions[C]//Proceedings of 2020 IEEE International Conference on Big Data. 2020: 1-10.

[30] Groce A, Feist J, Grieco G, et al. What are the actual flaws in important smart contracts (and how can we find them)?[C]//Proceedings of International Conference on Financial Cryptography and Data Security. 2020: 634-653.

[31] Chen H, Pendleton M, Njilla L, et al. A survey on ethereum systems security: Vul-nerabilities, attacks, and defenses[J]. ACM Computing Surveys (CSUR), 2020, 53(3): 1-43.

[32] Rameder, H. (2021). "Systematic Review of Ethereum Smart Contract Security Vulnerabilities, Analysis Methods and Tools" [Diploma Thesis, Technische Universität Wien]. reposiTUm. https://doi.org/10.34726/hss.2021.86784

[33] Soud M, Liebel G, Hamdaqa M. A Fly in the Ointment: An Empirical Study on the Characteristics of Ethereum Smart Contracts Code Weaknesses and Vulnerabilities[J]. arXiv preprint arXiv:2203.14850, 2022.