FTEI: A Fault Tolerance Model of FPGA with Endogenous Immunity

Jie Wang, Shuangmin Deng(✉), Junjie Kang, and Gang Hou

School of Software Technology, Dalian University of Technology, Dalian 116023, China
wang_jie@dlut.edu.cn, ku_nan_xi@163.com, 18840837856@163.com, Hg.dut@163.com

Abstract. FPGA emerges as a very promising AI chip and algorithm hardware accelerator. However, the FPGA is susceptible to complex and changeable environment, which leads to circuit configuration information faults. To address this issue, we propose FTEI, a fault tolerance model of FPGA with endogenous immunity. At fault detection phase, we put forward a fault detection models based on optimized logistic regression classification and use it to establish a fault model matching library. During fault recover stage, we use fault configuration library and online evolution to recover faults. In order to improve the success ratio of online evolution, we propose RLAGA, an adaptive genetic algorithm based on reinforcement learning. Experiments on typical functional circuits, 8-bit parity verifier and 2-bit multiplier, demonstrate that the fault detection accuracy rates reach 94.4% and 93.2%, and the fault recover success rates of RLAGA are 100% and 90%, which significantly improves FPGA errors detection and recover effectiveness.

Keywords: FPGA · Fault tolerance · Fault detection · Fault recover

1 Introduction

The extraordinary advantages of FPGA are high concurrency, low delay, and reconfiguration. Taking these advantages FPGA emerges as a very promising AI chip and algorithm hardware accelerator. However, a non-negligible issue of FPGA is that complex and changeable environment, high temperature, high pressure and high radiation, will change the configuration information on FPGA, which causes Single Event Upset (SEU). Once the FPGA chip occurs SEU errors that cannot be ruled out timely, the output results of FPGA will be wrong. More seriously, it will lead to equipment stagnation. Hence, it is a core issue to improve the reliability of FPGA chips.

In literature, flourishing researches have been achieved on fault tolerance technology. In general, approaches can be broadly grouped into three categories:

S. Shi et al. (Eds.): AICON 2020, LNICST 356, pp. 544–557, 2021.
https://doi.org/10.1007/978-3-030-69066-3_48

redundancy technology [1,2], reconfigurable technology [3,4] and evolvable hardware technology [5]. Although much effort has been devoted to redundancy technology because of its simple design ideas, this method increases the complexity of circuit design rapidly and large resource consumption. Reconfigurable technology refreshes the system by reconfiguring configuration information, saving lots of resource. However, the fault detection and location methods [6] are required to highly accurately locate the fault circuit of the system, which greatly increases the difficulty of system design. The recover technology based on evolvable hardware, using the characteristics of self-organization, self-adaption and self-recovery, can recover FPGA faults with less hardware circuit resources and has better fault tolerance performance.

In order to achieve efficient and real-time recover of SEU errors, our comprehensive analysis of the above methods found that the following three challenges need to be solved. The first challenge is (1) how to improve the real-time and accuracy of fault location. The second challenge is (2) how to ensure the normal operation of the circuit system while recovering faults. The third challenge is (3) how to improve the efficiency of fault recover under strict space-time constraints.

To tackle all the challenges mentioned above, in this paper we propose FTEI, a Fault tolerance model of FPGA with endogenous immunity. It endows FPGA fault perception, fault memory, and environment adaptation to improve the reliability of the FPGA platform. We realize fault perception through fault detection and location mechanism. The realization of fault memory and environment adaptation is achieved through fault recover mechanism. In general, the main contributions of this paper are summarized as below:

- To realize real-time fault location, we propose to take the FTRL-optimized logistic regression classification algorithm as the fault detection model.
- We propose pre-setting chromosome of known faults in fault configuration library to save recover time and truth table of circuits in circuit truth table module to guarantee the normally operation of fault circuits.
- To realize faults recover of unknown faults we recover circuit by online evolution and RLAGA algorithm is proposed to raise the success rate of online evolution.

2 Related Work

FPGA is susceptible to complex and changeable environment. Therefore, enhancing the reliability of FPGA has won a lot of research interest. These researches can be classified into two categories: fault-tolerant methods with fault detection capabilities and fault-tolerant methods with fault shielding capabilities.

2.1 Fault-Tolerant Methods with Fault Detection Capabilities

The fault-tolerant methods with fault detection capability achieve troubleshooting by adding additional fault detection resources to the system design. Wang

et al. [7] took deep learning algorithms as a fault detection model to monitor runtime data. Reorda et al. [8] utilized additional logic of carrying chains and hard links to perform error detection to implement fault detection and correct single and two errors that affected FPGA configuration memory due to configuration bit flips. Du et al. [9] exploited the bitstream analysis tool readback bitstream to obtain the current status and absolute addresses of D flip-flops (DFFs) and storage units. By analyzing the above information, the fault location is obtained. Ranjbar et al. [10] devised a method which could transfer the effects of faults occurring in the LUT (Look-Up-Table) to triggers facilitating fault detection. Although the above methods can be used to locate circuit faults, it cannot guarantee the real-time performance of circuit fault detection and increase the difficulty of circuit design and realization.

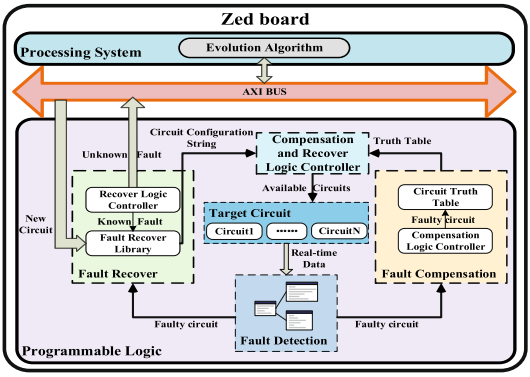

Fig. 1. Chip real-time self-healing system architecture

2.2 Fault-Tolerant Methods with Fault Shielding Capabilities

Triple Modular Redundancy (TMR) can directly shield circuit faults when a single modular circuit fails, which can effectively protect the system from circuit faults and maintain the normal operation of the circuit system [11]. Given the hardware resource consumption problem of TMR, Schweizer et al. [12] proposed a strategy using unused functional units for redundant calculation on a coarse-grained reconfigurable architecture and realized a low-cost TMR strategy. Experiments showed that this strategy reduced the area of hardware resources by 12.8% compared with the traditional TMR method. Burdyshev et al. [13] reduced a large number of extra hardware resources consumed by TMR technology to achieve fine-grained TMR by calculating the combination of channel redundancy and transistor redundancy. Although the above methods can save hardware resource, it makes the circuit more complex and difficult.

To overcome these above shortcomings, we develop FTEI, a fault tolerance model of FPGA with endogenous immunity. Besides real-time fault perception, the fault memory and environment adaptation are realized.

3 The Proposed Model

In this section, we generalize FTEI architecture firstly. Secondly, we introduce fault detection and location mechanism. Finally, the circuit fault recover mechanism are depicted.

3.1 Fault Tolerance Model of FPGA with Endogenous Immunity (FTEI)

The system structure is shown in Fig. 1. These circuits are divided into modules according to their functions in the design stage to facilitate detection and recover. The target circuit module runs on FPGA which needs to be detected and recovered when faults occur to it. The detection module is exploited to supervise the real-time data of the system. Once the detection module detects circuit faults, the circuit needs to be recovered in servery clock time. In order to buy time for fault recover and ensure the normal operation of the circuit system, the compensation module is mentioned. We pre-store truth table data of target circuits in the circuit truth table and the compensation logic controller is utilized to control data transfer of the compensation module. The fault recover module consists of two parts: fault recover library and evolution algorithm. We preset some circuit configuration chromosome in the fault configuration library whose faults are known. For unknown faults, we use evolution algorithm to recover and then store obtained configuration chromosome into the fault configuration library. The process is controlled by recover logic controller.

3.2 Fault Detection and Location Mechanism

In order to realize the precise location of fault circuit, this paper proposes a fault detection model, FTRL-optimized logistic regression, and uses it to establish a fault model matching library.

Fault Detection Mechanism. The fault detection process is divided into two stages: fault detection model acquisition and fault online detection. The fault detection model is obtained through offline training. The offline training of fault detection model needs to obtain a large number of circuit fault data. In this paper, a software tool is utilized to randomly inject faults into the Cartesian Genetic Programming (CGP) [16,17] code string running on the Virtual Reconfigurable Circuit (VRC) [7,15]. Considering that the circuit data can be divided into two categories: with fault and without fault, therefore, circuit fault detection is actually a binary classification problem. To get the fault detections model, this paper proposes FTRL-optimized logistic regression.

FTRL (Follow-the-regularized-Leader) online learning algorithm integrates the advantages of FOBOS algorithm and RDA algorithm, which can better guarantee the accuracy of model parameters and the sparsity of feature items in the training process. At the same time, after adding non smooth regular items, FTRL

can get better sparsity value. So we utilized FTRL to optimize the parameters of the fault circuit logistic regression model [18].

The main iteration formula of FTRL can be expressed as Eq. (1):

$$w_{t+1} = \arg\min_{w}\{g_{1:t}w + \frac{1}{2}\sum_{s=1}^{t}\delta_s||w - w_s||_2^2 + \lambda_1||w||_1\} \tag{1}$$

The above formula is divided into three parts: $(\sum_{r=1}^{t} G^r)w$ is the cumulative gradient sum, which indicates the direction of loss function decline; $\frac{1}{2}\sum_{s=1}^{t}\delta_s||w - w_s||_2^2$ indicates that the new result should not deviate too far from the existing result; $\lambda_1||w||_1$ is the regular term, which is used to generate the sparse solution.

Let $\delta_{1:t} = \frac{1}{\sigma_t}$, $z_{t-1} = \sum_{r=1}^{t-1} G^r - \sum_{s=1}^{t-1}\delta_s - w_s$ we derive the following Eq. (2):

$$z_t = z_{t-1} + g_t - (\frac{1}{\sigma_t} - \frac{1}{\sigma_{t-1}})w_t \tag{2}$$

Substituting Eq. (2) into Eq. (1), the iteration formula can be rewritten as Eq. (3).

$$w_{t-1} = \arg\min_{w}\{(g_{1:t}w - \sum_{s=1}^{t}\delta_s w_s)w\frac{1}{2\sigma_t}||w - w_s||_2^2 + \lambda_1||w||_1 + c\} \tag{3}$$

Thus, Eq. (4) can be described as:

$$w_{t+1,i} = \begin{cases} 0, & |z_{t,i}| < \lambda_1 \\ -\sigma_t(z_{t,i} - \lambda_1 sgn(z_{t,i})), & z_{t,i}| \geq \lambda. \end{cases} \tag{4}$$

To improve the sparsity and accuracy of parameters, L2 regular term is added to the regular term of FTRL in this paper, and the method of mixed regular term is adopted to make the solution result of the fault model smoother, and the accuracy of model prediction is also increased.

The feature weight iterative formula of the optimized FTRL can be expressed as Eq. (5):

$$w_{t+1} = \arg\min_{w}\{g_{1:t}w + \frac{1}{2}\sum_{s=1}^{t}\delta_{1:t}||w - w_s||_2^2 + \lambda_1||w||_1 + \frac{1}{2}\lambda_1||w||_2^2\} \tag{5}$$

Here, g_s is the standard gradient; δ_s is learning an update strategy; $\delta_{1:t} = \frac{1}{\sigma_t}$ and σ_t is described as Eq. (6):

$$\sigma_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{s,t}^2}} \tag{6}$$

The formula (5) is expanded to obtain the optimal solution problem represented by Eq. (7):

$$w_{t+1} = \arg\min_{w}\{(g_{1:t}w - \frac{1}{2}\sum_{s=1}^{t}\delta_s w_s)w + \lambda_1||w||_1 + \frac{1}{2}(\lambda_2 + \sum_{s=1}^{t}\sigma_s)||w||_2^2 + \frac{1}{2}\sum_{s=1}^{t}\sigma_s)||w_s||_2^2\} \tag{7}$$

Where $\frac{1}{2}\sum_{s=1}^{t}\sigma_s)||w_s||_2^2$ for w is a constant, let:

$$z^{(t)} = g_{1:t}w - \sum_{s=1}^{t}\delta_s w_s \tag{8}$$

The Eq. (7) is equivalent to Eq. (9):

$$w_{t+1} = \arg\min_{w}\{z_t w + \lambda_2||w||_1 + \frac{1}{2}(\lambda_2 + \sum_{s=1}^{t}\sigma_s)||w||_2^2\} \tag{9}$$

According to each dimension of feature weight, the above formula is decomposed into n independent scalar minimization problems.

$$w_{t+1} = \arg\min_{w}\{z_{t,i}w_i + \lambda_1||w||_1 + \frac{1}{2}(\lambda_2 + \sum_{s=1}^{t}\sigma_s)w_i^2\} \tag{10}$$

Therefore, the weight formula shown in Eq. (11) can be obtained.

$$w_{t+1,i} = \begin{cases} 0, & |z_{t,i}| < \lambda_1 \\ -(\lambda_2 \sum_{s=1}^{t}\sigma_s)^{-1}(z_{t,i} - \lambda_1 sgn(z_{t,i})), & |z_{t,i}| \geq \lambda_1. \end{cases} \tag{11}$$

The pseudo code of circuit fault detection based on FTRL-optimized logistic regression is as Algorithm 1.

Algorithm 1. OPTIMAL SOLUTION OF LOGISTIC REGRESSION BASED ON FTRL-OPTIMIZED ALGORITHM

Input: $\alpha, \beta, \lambda_1, \lambda_2$;

for $i = 0$ to d **do**
  Initialize $z_i = 0, n_i = 0$;
end for
for $i = 1$ to t **do**
  The eigenvector of fault data set is X_t, Let $I = \{i|x_i \neq 0\}$;
  for $i \in I$ **do**
    $w_{t+1,i} = \begin{cases} 0, & |z_i| < \lambda_1 \\ -(\lambda_2 + \frac{\beta + \sqrt{n_i}}{\alpha})^{-1}(z_i - \lambda_1 sgn(z_i)), & |z_i| \geq \lambda_1. \end{cases}$
  end for
  Calculate estimated probability $p_t = \sigma(X_t \cdot W)$ by $w_{t,i}$ according to Equation 4;
  Tags from training set $y_t \in \{0, 1\}$, indicates if the circuit is faulty;
  for $i \in I$ **do**
    Calculate gradient loss $g_i = (p_t - y_t)x_i$;
    Calculate $\sigma_i = \frac{1}{\alpha}(\sqrt{n_i - g_i^2} - \sqrt{n_i})$;
    Update $z_i = z_i + g_i - \sigma_i W_{t,i}$;
    Update $n_i = n_i + g_i^2$;
  end for
end for

3.3 Fault Recovery Mechanism

RLAGA: Adaptive Genetic Algorithm Based on Reinforcement Learning. When an unknown circuit fault occurs, in order to ensure the recover of the fault, the evolutionary recover algorithm is used to obtain an alternative circuit. Therefore, this paper proposes RLAGA. In this algorithm, the crossover operators and mutation operators of genetic algorithm are dynamically and adaptively adjusted by the reward feedback mechanism of reinforcement learning, so the diversity of population in the iterative process can be maintained which avoids the algorithm falling into the local optimal solution, and improves the evolution efficiency of genetic algorithm. Q-learning algorithm [19] is adopted as the learning algorithm to strengthen the learning agent. Through the crossover operators and mutation operators of the population to reducing the probability of local optimum. Therefore, it is necessary to consider the impact of crossover operators and mutation operations on population diversity, when designs the learning process of reinforcement learning.

The population diversity is measured by the population fitness. The population size is n, the fitness of the i-th individual x_t^i at time t is $fit(x_t^i)$, and the average fitness of the population at time t is $fit_{avg}(x_t)$, if the individual difference of population is d_t^i, it means the number of chromosomes in the population with different fitness from the i-th chromosome, then the population diversity can be expressed as Eq. (12).

$$div(x_t) = \frac{1}{N} \sum_{i=1} N|fit(x_t^i) - fit_{avg}(x_t)|d_t^i \tag{12}$$

The learning mechanism of reinforcement learning for genetic algorithms mainly includes three elements: (1) setting and division of environmental status; (2) Agent's action division; (3) determination of action reward value.

State Setting and Division. As shown in Eq. (12), according to the maximum iterative algebra G of the genetic algorithm, the whole evolutionary iterative process is divided into four stages. According to the value of div of population diversity, the value range is divided into four intervals. In this paper, the environment of genetic algorithm is divided into 16 states by combining iterative algebra and population diversity as state value of environment.

$$S_G = \begin{cases} s_{G1}, \ G \in \left[0, \frac{G}{4}\right) \\ s_{G2}, \ G \in \left[\frac{G}{4}, \frac{G}{2}\right) \\ s_{G3}, \ G \in \left[\frac{G}{2}, \frac{3G}{4}\right) \\ s_{G4}, \ G \in \left[\frac{3G}{4}, G\right) \end{cases} S_{div} = \begin{cases} S_{div1}, \ div \in [0, 0.5) \\ S_{div2}, \ div \in [0.5, 1.0) \\ S_{div3}, \ div \in [1.0, 1.5) \\ S_{div4}, \ div \in [1.5, +\infty) \end{cases} \tag{13}$$

Action Division. After Agent obtains the state and reward value at time t from the genetic algorithm environment, it will dynamically and adaptively adjust the crossover operator P_c and mutation operator P_m of the genetic algorithm according to the feedback value, and then agent will transfer the adjusted operator parameters to genetic algorithm. The adjustment of P_c and P_m according to Agent is shown in Eq. (14) and (15). According to the difference between Pc and Pm, there are 9 kinds of action combinations that Agent can take for genetic algorithm at time t.

$$P_c(t) = P_c(t) + \Delta\alpha, \Delta\alpha = \begin{cases} -k_1 \\ 0, k_1 = 0.05 \\ k_1 \end{cases} \tag{14}$$

$$P_m(t) = P_m(t) + \Delta\beta, \Delta\beta = \begin{cases} -k_1 \\ 0, k_1 = 0.05 \\ k_1 \end{cases} \tag{15}$$

Determination of Reward Value. In this paper, the reward mechanism of the state action is established by comparing the population fitness obtained by the cross mutation of the genetic algorithm after the agent action. The calculation of action reward R is shown in Eq. (16). While the average fitness of the population after iteration is greater than the previous generation, the reward value is positive; while the average fitness of the offspring is equal to the previous generation, the action does not generate revenue, the reward value is 0; while the average fitness of the offspring is less than the previous generation, the reward value is negative.

$$R = \begin{cases} 1, -\Delta fit > 0 \\ 0, -\Delta fit = 0, \Delta fit = fit_{avg}(x_t) - fit_{avg}(x_{t-1}) \\ -1, -\Delta fit < 0 \end{cases} \tag{16}$$

In order to better calculate the cumulative reward value of Agent in the iterative process, this paper adopts the Q-learning algorithm. The optimal action strategy group is obtained by continuously evaluating the value function of the state action pairs, and Eq. (17) is the Q value calculation formula of Q-learning algorithm. Here, Q is the value of the state action pairs; α is the learning step of agent; r_t is the reward value of environmental feedback when action a_t is taken in state s_t at time t; γ is the discount rate; $maxQ(s_{t+1}, a_{t+1)})$ is in the next state $s_(t+1)$ the maximum Q value corresponding to the action taken.

$$Q(s_t, \alpha_t) = Q(s_t, \alpha_t) + \alpha[r_t + \gamma maxQ(s_{t+1}, \alpha_{t+1}) - Q(s_t, \alpha_t)] \tag{17}$$

The pseudo code of the adaptive genetic algorithm based on reinforcement learning is shown as Algorithm 2

Algorithm 2. RLAGA

Input: parameter population size N, chromosome length L, maximum iteration algebra G, initial crossover operator P_c, initial mutation operator P_m, initial Q-value table, initial learning step length, initial discount rate;

1: Initial $P_{(t)}$;
2: $fit(x_t) = AdpCalculateFit(P_{(t)})$;
3: Calculate the current status value S and reward value R;
4: Agent reads status value S and reward value R, selects actions a_t by greedy strategy, and updates the crossover operator P_c and mutation operator P_m;
5: **if** Termination condition **then**
6: break;
7: **end if**
8: **for** $i = 0$ to G **do**
9: $fit^{'}(x_t) = a \cdot fit(x_t) + b$;
10: $P^{'}(t) = Select(P(t), fit^{'}(x_t))$;
11: $P^{''}(t) = Crossover(P^{'}(t), P_c)$;
12: $P^{'''}(t) = Mutation(P^{''}(t), P_m)$;
13: $P(t) = P^{'''}(t)$;
14: $fit(x_t) = AdpCalculateFit(P(t))$;
15: Calculate the current status value S and reward value R, and send them to the agent;
16: Calculate $Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma maxQ(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$; Update Q value table; Selects actions by greedy strategy; Updates the crossover operator P_c and mutation operator P_m;
17: **if** Termination condition **then**
18: break;
19: **end if**
20: $t = t + 1$;
21: **end for**

4 Experiment

In this section, we firstly illustrate the fault injection process, data acquisition and system experimental parameter settings. Secondly, we utilize the 8-bit parity checker and the 2-bit multiplier to verify the effectiveness of the proposed scheme and demonstrate the advantages of our system.

4.1 Fault Injection and Data Acquisition

Existing fault injection methods to simulate the actual fault of the circuit can be roughly divided into two categories: (1) utilize the simulator to simulate the fault of the hardware circuit; (2) directly modify the hardware circuit and inject the fault from the circuit source file. In order to fit the actual fault problem of the circuit more closely, we uses the method mentioned by Wang et al. [7]. Taking the 2-bit multiplier as an example, the chromosome coding string is { 020 131 210 033 055 444 554 591 480 502 765 250 583 4132 1242 18 12 14 10 }. Then we inject errors into CGP code string to obtain fault data.

Table 1. Experimental results of 8-bit parity checker

	Logistic regression	Optimized logistic regression
Off-chip test accuracy	85.1%	86.6%
On-chip test accuracy	92.5%	94.4%
Average time (clock cycle)	8	8

Table 2. Experimental results of 2-bit multiplier

	Logistic regression	Optimized logistic regression
Off-chip test accuracy	80.2%	84.3%
On-chip test accuracy	89.4%	93.2%
Average time (clock cycle)	15	15

4.2 Experiment Setting

Zed board [14] is an SRAM-based FPGA, which is used as the verification platform for the self-recovery experiment of chip circuit. In order to fully utilize the cooperative features of Zed boards software and hardware, the target circuit is mapped to chromosome structure string through CGP and configured on VRC circuit to make it work on Programmable Logic (PL) part. Then the fault detection and location mechanism and the fault recover mechanism are established. In the Processing System (PS) part, an evolutionary hardware recovery module is established. When an unknown fault occurs in the circuit evolution algorithm will generate configuration chromosome of the fault circuit and save in the fault recovery configuration library on PL interacting through AXI bus.

4.3 FTRL-Optimized Logistic Regression

We use the fault data of the 8-bit parity checker and the 2-bit multiplier to perform offline model training on traditional logistic regression and FTRL-optimized logistic regression algorithm. Then we download and configure the above model into FPGA, which is used as the fault model matching library for fault detection and location. After that the actual injection of faults on the chip is used to verify the accuracy of the obtained fault model.

The 8-bit parity checker data set scale of fault training is 1000; the fault test set scale is 500, and the on-chip fault test scale is 100. For 2-bit multiplier, the scale of fault training set is 2000; the scale of fault test set is 500, and the on-chip fault test set is 200. Table 1 and Table 2 show the accuracy of the offline fault test set of the fault model, the fault test set after fault injection of the on-chip circuit, and the average time consumption of fault detection.

According to Table 1 and Table 2, FTRL-optimized logistic regression performs better in accuracy both online and offline tests. At the same time, the average time of fault detection is only related to the circuit itself, and the fault model is not related, which guarantees the real-time performance of fault detection on chip. Due to the complexity of the circuit structure, the average time-consuming of the on-chip fault detection needs 15 clock cycles, and the real-time detection of the on-chip fault still has good performance.

The fault model matching library based on FTRL-optimized logistic regression adopted in this paper can effectively solve the problem of real-time detection of faulty circuits.

4.4 RLAGA Performance Comparison

The modify adaptive genetic algorithm adopts a population chromosome adaptation assessment method which changes with the iteration time. Different evolution iteration coefficients have an influence on the evolution time of each generation of chromosomes. Therefore, different evolutionary iterative coefficients are used to test the adaptive evaluation method, and the most appropriate evolutionary iterative coefficient is determined by evolutionary operation of the target circuit. The experimental results are shown in Table 3. We can known the higher evolution iteration coefficient saves the evaluation time, but leads to the lower efficiency of evolution, which cannot truly reflect the real-time fitness value of each chromosome. Therefore, it is necessary to comprehensively consider the evolution time consumption and the overall efficiency of evolution. According to the results in Table 3, we set the evolution iteration coefficient to 0.3 to ensure the best evolution time and efficiency.

Table 3. Time-consuming results of evolution with different coefficients

	8-bit parity checker		2-bit multiplier	
Iteration coefficient	*Time consuming per generation*	*Total time (s)*	*Time consuming per generation*	*Total time(s)*
0.1	5.09	5.37	5.23	41.21
0.3	4.73	5.23	4.87	34.27
0.5	4.53	6.45	4.61	76.32
0.7	4.46	8.36	4.49	265.74
0.9	4.35	12.83	4.38	403.82

To validate the evolutionary recover efficiency of the adaptive genetic algorithm based on reinforcement learning, in this paper we do experiment and compare with standard genetic algorithm SGA, ant colony algorithm PSO, adaptive genetic algorithm AGA and self-simulation annealing Adaptive genetic algorithm SAGA in the time consumption and evolution success rate [1]. Figure 2 respectively represents the evolutionary recover results of randomly injecting different circuit faults into the 8-bit parity verifier and the 2-bit multiplier for 100 times,

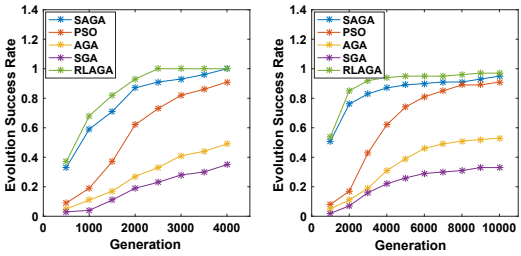

Fig. 2. Fault recovery experiment results, left figure is 8-bit parity checker results and right figure is 2-bit multiplier results

and then adopting 4 algorithms to carry out fault recover operation on the chip circuit at recover stage.

Comparing the results of the five algorithms, we can see that the RLAGA algorithm and SAGA algorithm proposed in this paper have achieved a high evolutionary success rate for 8-bit parity checker at about 1000 generations, and the circuit recover success rate is 100% at about 4000 generations. Among the other three algorithms, only PSO can achieve 90% of the recover success rate in 4000 generations, and the overall recover time consumption is far higher than the algorithm proposed in this paper. The circuit structure of 2-bit multiplier is more complex than that of 8-bit parity check. In the process of circuit evolution, the faults recover success rate of five kinds of algorithms in 1000 generations is not very good. However, RLAGA and SAGA algorithm can still achieve more than 50% of the recovery rate. Because RLAGA algorithm guarantees the population diversity in the evolutionary iteration process, the faults recover rate reaches 90% at about 3000 generations, while SAGA is relatively slow. Other algorithms have defects in the recover time consumption and recovery rate. Therefore, for the circuit with simple structure, SAGA and RLAGA can achieve better results, but when the circuit to be recovered is large, RLAGA can better improve the efficiency of evolutionary recover (Table 4).

Table 4. Comprehensive comparison of fault recover algorithms

	8-bit parity checker		2-bit multiplier	
Algorithm	*Average time(s)*	*Success rate*	*Average time(s)*	*Success rate*
RLAGA	4.2	100%	27.8	97%
SAGA	5.3	100%	34.5	95%
PSO	9.2	91%	89.8	92%
AGA	19.7	56%	163.1	53%
SGA	24.1	35%	178.6	33%

5 Conclusion

In this paper, we design and implement FTEI, an FPGA fault-tolerant model with endogenous immunity. In fault detection phase, we take FTRL-optimized logistic regression as fault detection model to establish a fault model matching library. In fault recover stage, we combine fault configuration library with online evolution to improve the recovery efficiency. Simultaneously, RLAGA is proposed to improve success rate of evolution. Experimental results of typical functional circuits demonstrate that the fault detection accuracy rates are 94.4% and 93.2%, and the fault recovery success rates of RLAGA are 100% and 90%. As part of our future works, we will:

- Further study multi-class algorithms, and improve the accuracy of fault detection;
- Study evolutionary algorithm in higher scale digital integrated circuits and improve success rate of evolution.

References

1. Wang, J., Kang, J., Hou, G.: Real-time fault recovery scheme based on improved genetic algorithm. IEEE Access **7**, 35805–35815 (2019)
2. Anjankar, S., Kolte, M., Pund, A., Kolte, P., Kumare, A., Mankarf, P., Ambhore, K.: FPGA based multiple fault tolerant and recoverable technique using triple modular redundancy (FRTMR). In: ICCCV, pp. 827–834 (2016)
3. Yang, X., Li, Y., Fang, C., Nie, C., Ni, F.: Research on evolution mechanism in different-structure module redundancy fault-tolerant system. In: ISICA, pp. 171–180 (2015)
4. Gong, J., Yang, M.: Evolutionary fault tolerance method based on virtual reconfigurable circuit with neural network architecture. IEEE Trans. Evol. Comput. **22**(6), 949–960 (2018)
5. Wang, J., Liu, J.: Fault-tolerant strategy for real-time system based on evolvable hardware. J. Circ. Syst. Comput. **26**(7), 1–18 (2017)
6. Palchaudhuri, A., Dhar, A.: Design and automation of VLSI architectures for bidirectional scan based fault localization approach in FPGA fabric aware cellular automata topologies. J. Parallel Distrib. Comput. **130**, 110–125 (2019)
7. Wang, J., Deng, S., Kang, J., Hou, G., Zhou, K., Lin, C.: A real-time fault location mechanism combining CGP code and deep learning. In: DSA (2020, in press)
8. Reorda, M.S., Sterpone, L., Ullah, A.: An error-detection and self-recoverying method for dynamically and partially reconfigurable systems. IEEE Trans. Comput. **66**(6), 1022–1033 (2017)
9. Ruan, T., Jie, P.: A bitstream readback based FPGA test and diagnosis system. In: ISIC, pp. 592–595 (2014)
10. Ranjbar, O., Sarmadi, S., Pooyan, F., Asadi, H.: A unified approach to detect and distinguish hardware trojans and faults in SRAM-based FPGAs. J. Electron. Test. **35**(2), 201–214 (2019)
11. Halawa, H., Daoud, R., Amer, H.: FPGA-based reliable TMR controller design for S2A architectures. In: ETFA, pp. 1–8 (2015)

12. Schweizer, T., Schlicker, P., Eisenhardt, S., Kuhn, T., Rosenstiel, W.: Low-cost TMR for fault-tolerance on coarse-grained reconfigurable architectures. In: ReConFig, pp. 135–140 (2011)
13. Burdyshev, I., Tyurin, S.: Fault tolerant FPGAs design method. In: EIConRus, pp. 248–251 (2020)
14. Shanker, S., Bhaskar, B., Kizheppatt, V., Suhaib, A.: Dynamic cognitive radios on the Xilinx Zynq hybrid FPGA. In: CROWNCOM, pp. 427–437 (2015)
15. Gong, J., Yang, M.: Evolutionary fault tolerance method based on virtual reconfigurable circuit with neural network architecture. IEEE Trans. Evol. Comput. **99**, 1–1 (2017)
16. Miller, J., Thomson, P.: Cartesian genetic programming. In: EuroGP, pp. 121–132 (2000)
17. Julian, M., Andrew, T.: Cartesian genetic programming. In: Proc. 2015, pp. 179–198
18. Ruder, S.: An overview of gradient descent optimization algorithms. Computer Research Repository, vol. abs/1609.04747, p. 12 (2016)
19. Indu, J., Chandramouli, K., Shalabh, B.: Generalized speedy Q-learning. IEEE Control Syst. Lett. **4**(3), 524–529 (2019)