Hindawi
Security and Communication Networks
Volume 2021, Article ID 6682674, 9 pages
https://doi.org/10.1155/2021/6682674

Research Article

Hardware Trojan Detection Based on Ordered Mixed Feature GEP

Huan Zhang[1,2] **Jiliu Zhou**[1] **Dongrui Gao**[2,3] **Xinguo Wang**[2] **Zhefan Chen**[4] **and Hongyu Wang**[2]

[1]*College of Computer Science, Sichuan University, Chengdu 610065, China*
[2]*School of Computer Science, Chengdu University of Information Technology, Chengdu 610225, China*
[3]*Center for Information in Biomedicine, School of Life Sciences and Technology, University of Electronic Science and Technology of China, Chengdu 611731, China*
[4]*Faculty of Science, Simon Fraser University, Burnaby V5A 1S6, Canada*

Correspondence should be addressed to Jiliu Zhou; zhoujl@cuit.edu.cn

Received 28 December 2020; Revised 24 January 2021; Accepted 21 February 2021; Published 18 March 2021

Academic Editor: Liguo Zhang

In the hardware Trojan detection field, destructive reverse engineering and bypass detection are both important methods. This paper proposed an evolutionary algorithm called Ordered Mixed Feature GEP (OMF-GEP), trying to restore the circuit structure only by using the bypass information. This algorithm was developed from the basic GEP through three sets of experiments at different stages. To solve the problem, this paper transformed the GEP by introducing mixed features, ordered genes, and superchromosomes. And the experiment results show that the algorithm is effective.

1. Introduction

At all stages of the life cycle of integrated circuits (IC), there are security vulnerabilities for hardware in the global business model of semiconductor supply chain. In the current hardware Trojan detection technology, destructive reverse engineering [1–3] is good but costly bypass detection [4–9] is the technology whose cost is low, which is a development key direction at present.

An evolutionary algorithm called Ordered Mixed Feature GEP (OMF-GEP) is proposed in this paper. This algorithm takes a single-circuit component as a node to form a mixed feature of various logical or physical features of the node. And it can find the original circuit by using the GEP function regression ability.

2. Mixed Features

The logic value of a circuit or any kind of bypass information such as voltage and current can be considered as a manifestation of a characteristic of the circuit. When only one characteristic representation value is used to represent the circuit, if other constraints are not added, there will undoubtedly be a variety of circuits to meet the requirements of this single feature. The two circuits are shown in Figure 1.

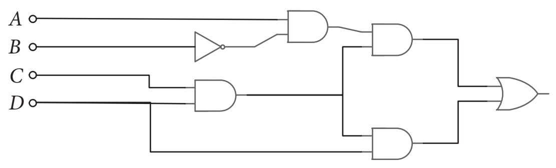
If you look only at the logical values, the two circuits are completely equivalent, and both of which are

$$Y = CD. \tag{1}$$

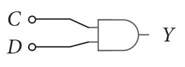
You can also see that in the circuit in Figure 1(b), inputs A and B are used at all; that not, the input s A and B in the first circuit do not actually affect the output.

This example is only the value of the circuit logic value and the circuit bypass information detection, and there is a similar situation. Multiple different circuit structures can be obtained for the detection results of any single bypass information. It can be seen that only using logical values or bypass information to describe the circuit will lead to too much isomorphism to confirm the circuit structure.

The essence of hardware Trojan horse design is to add additional circuit to normal circuits, but the performance of the whole circuit on some features (the most common is the logic value) is the same as that of the normal circuit, to realize hiding. However, this additional circuit will inevitably cause other circuit features to change.



(a)



(b)

Figure 1: Two circuits with the same logical value.

In this paper, the logic value of the circuit or any kind of bypass information such as voltage and current is called a feature. For the isomorphism of a feature, it is essentially due to the superposition of the features of the circuit elements on the feature. The features of multiple different circuit structures with the feature are similar or even the same, so that the corresponding circuit cannot be represented by the result of a feature.

Then, when detecting multiple features at the same time, multiple isomorphic circuits can be obtained from the detection results of each feature, but the superposition features of different features cannot be exactly the same. These isomorphic circuits cannot be the same, where the same part is a possible real circuit.

Figure 2 illustrates the application of the algorithm to a diode-designed And gated circuit.

Its logical meaning is

$$Y = A \cdot B. \tag{2}$$

It has a lot of physical meaning. Here is description of its voltage:

$$V_y = \min(V_A, V_B) + V_{\text{dio}}. \tag{3}$$

Among them, V_y represents voltage of the output position Y; V_A and V_B represent the voltage values of two input locations A and B; V_{dio} represents the diode conduction voltage for silicon tubes, and its value is often 0.7.

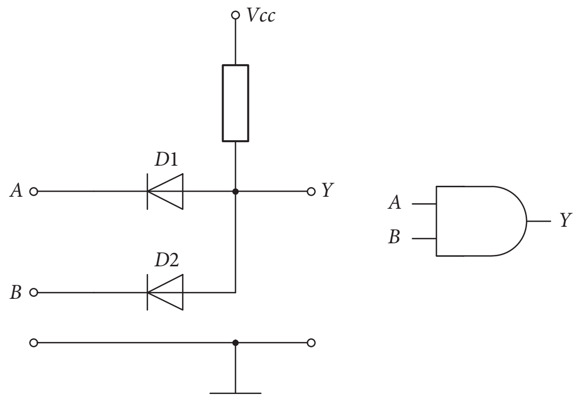


Figure 2: And gate circuits.

Then, the And gate can be expressed as

$$\left(y_{\text{logic}}, y_v\right) = G_{\text{and}}\left(x_{1\_\text{logic}}, x_{2\_\text{logic}}, x_{1\_v}, x_{2\_v}\right). \tag{4}$$

Then, a measurement of k features, n input, and single-output single-gate circuit can be expressed as Expression 1:

$$\left(y_1, y_2, \ldots, y_k\right) = G\left(\left(x_{1,1}, x_{1,2}, \ldots, x_{1,n}\right), \left(x_{2,1}, x_{2,2}, \ldots, x_{2,n}\right), \ldots, \left(x_{k,1}, x_{k,2}, \ldots, x_{k,n}\right)\right). \tag{5}$$

Among them, $y_i\,(i = 1, \ldots k)$ is the output values for the adoption of the kth feature. $x_{i,j}\,(i = 1, \ldots, k, j = 1, \ldots n)$ is the ith input value for the adoption of the feature j.

Without losing generality, let us define

$$\begin{aligned} Y &= \left[y_1, y_2, \ldots y_k\right], \\ X_i &= \left[x_{1,i}, x_{2,i}, \ldots, x_{n,i}\right]^T, \quad i = 1, 2, \ldots, k, \\ X &= \left[X_1, X_2, \ldots, X_k\right]. \end{aligned} \tag{6}$$

Then, Expression 1 can be expressed as

$$Y_{1*k} = G\left(X_{n*k}\right), \tag{7}$$

which is Expression 2.

3. Algorithm 1: Single-Output Circuit

3.1. GEP Representation. In recent years, there have been many studies based on evolutionary algorithms and multisource data such as data fusion of adaptive weighted multisource sensor [10], the research on evolutionary algorithm for symbolic network [11], the application of the genetic algorithm in multiobjective multicast routing [12], and multiplicity problems in genetic association studies [13]. Zhi and Liu [14] proposed a new GA algorithm for mechanical design optimization problems. These studies gave us the inspiration to use the evolutionary algorithms in the hardware Trojan detection.

Gene expression programming (GEP) [15] is an evolutionary computing algorithm that has performed well in the study of evolutionary hardware [16–22]. It can solve the problem of tree structure very well. For the multi-input/single-output tree structure circuit, it can be described as a tree with n leaf nodes, which can be represented directly by GEP. As shown in Figure 3, the 6-input/1-output logic circuit can be easily represented as a tree structure, in which the logic gate function is replaced by the logic symbol, and the corresponding effective gene is

$$\text{And, And, And, } A\text{, Not, And, And, } B, C\text{, Not, } E, F, D. \tag{8}$$

3.2. Algorithm of Mixed Feature GEP. One operator in GEP represents only one kind of calculation, and a GEP individual can only represent one test item, so the idea of algorithm one is to merge the multiple test results of a basic circuit into a function expression. Combined into a compound function, that is, let a function represent multiple calculations and evolve a representation close to the original circuit. Specifically, these multiple detection values are included in a function, the input of the function is multiple values, and the output result is a vector, such as the aforementioned gate circuit, which is still represented as "And" in the GEP expression tree. However, its meaning has become the following vector calculation:

$$\text{And}\left(A_1, A_2, \ldots A_n\right) = \left[F_1\left(A_1\right), F_2\left(A_2\right), \ldots, F_n\left(A_n\right)\right]. \tag{9}$$

The $A_k = (k = 1, \ldots n)$ is the input value of a detection, and $F_k\left(A_k\right)(k = 1, \ldots n)$ is the result of this detection A_k to the input value.

For example, for this gate circuit, the symbol And means the following:

$$\text{And}\left(\left(L_A, L_B\right), \left(V_A, V_B\right), \left(C_A, C_B\right)\right) = \left[L_A L_B, \min\left(V_A, V_B\right) + V_{\text{dio}}, C_A + C_B\right]. \tag{10}$$

Among them, L_A, L_B represent the logical value (1 or 0) of voltage input of the A or B point, V_A, V_B represent the input voltage of the A or B point, and C_A, C_B represent the input current of the A or B point.

Thus, when using GEP evolution, a symbolic value can simultaneously represent multiple unrelated items. This algorithm will be called Mixed Feature GEP (MF-GEP).

3.3. Experiment Setup. The experiment is limited to the use of simple logic gate circuits, does not involve triggers, clocks, etc., and does not consider time effects.

Four groups of experiments were designed.

Output $m = 1$,

The number of features are $k = 1, 2, 2, 3$.

Three features are used: feature 1 is the logical value, feature 2 is the voltage value, and feature 3 is the current value.

As a comparative experiment, the parameters used are identical as Table 1 shows.

3.4. Design of Fitness Function. The feature data are logic data, voltage data, and current data, which have their own fitness.

Logical data fitness is

$$F_{\text{logic}} = 1 - \frac{\sum_{i=1}^{N} \left|y_i - y\right|}{N}, \tag{11}$$

which is Expression 3.

N is the number of test data, y_i is the logic value calculated according to the test data after decoding, and y is the output logic value of the test data. Because it is a logical value, the worst case is that each output decoded by the individual is opposite to the test value, that is, $\left|y_i - y\right| = 1$, so F_{logic} is among the range of $[0, 1]$.

Voltage data fitness is

$$F_{\text{vol}} = 1 - \frac{\left(\sum_{i=1}^{N} \left|y_i - y\right|/N\right)}{\left(V_{\text{CC}} - V_{\text{DD}}\right)}, \tag{12}$$

which is Expression 4.

N is the number of test data, y_i is the voltage value calculated according to the test data after decoding, and y is the output voltage value of the test data. At worst, each test output value is either the highest level or the lowest level, and each output decoded by the individual is opposite to the test value $\left|y_i - y\right| = V_{\text{CC}} - V_{\text{DD}}$; therefore, F_{vol} is among the range of $[0, 1]$.

Current data fitness is

$$F_{\text{cir}} = 1 - \frac{\text{SSE}}{\text{SST}}, \tag{13}$$

which is Expression 5.

Among them,

$$\begin{aligned} \text{SSE} &= \sum_{i=1}^{m} \left(y_i - \widehat{y}_i\right)^2, \\ \text{SST} &= \sum_{i=1}^{m} \left(y_i - \overline{y}_i\right)^2, \end{aligned} \tag{14}$$

where y_i is the data observation, $\widehat{y}_i$ is the estimate value of the y_i which is calculated from the decoding expression, and $\overline{y}_i$ is the average value of the variable y. That is, the SSE is the Sum of Squared Errors and the SST is the Sum of Squares in Total. F_{cir} the square of the multicorrelation coefficient in statistics.

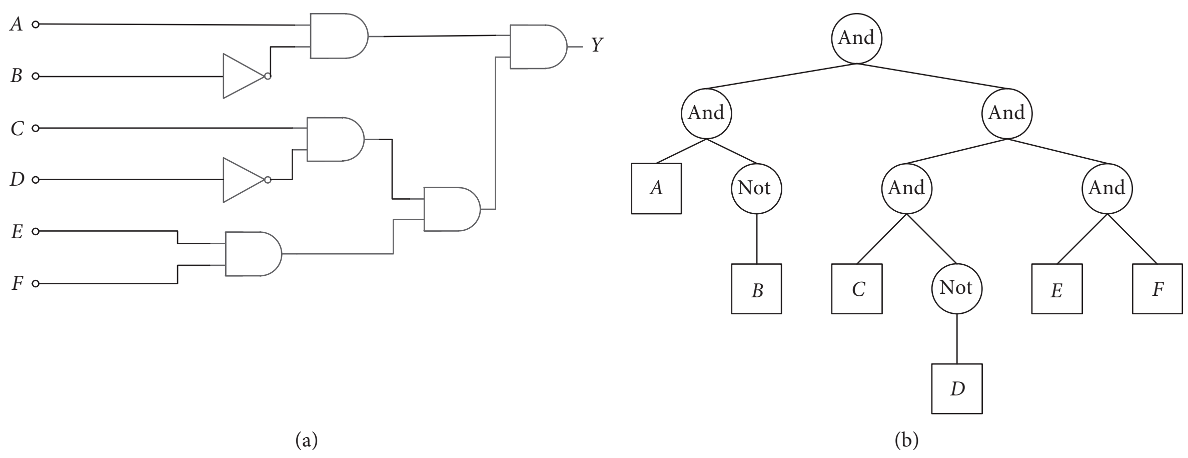


Figure 3: A single-output circuit and its expression tree.

Table 1: Parameter settings for experiment 1.

Parameter	Value
Stop	Fitness = 1
Selection mode	Tournament, size = 3
Population size	10000
Head length	20
Tail length	21
Chromosome length	1
Mutation rate	0.05
Insert rate	0.1
Root insert rate	0.01
One-point cross rate	0.1
Two-point cross rate	0.1
Input number	4
Output number	1
Function set	Not, and, or

According to the previous algorithm description, the individual fitness should be a combination of the three; then, the individual fitness is

$$F = C_1 \cdot F_{\text{logic}} + C_2 \cdot F_{\text{vol}} + C_3 \cdot F_{\text{cir}}, \quad \left(\sum C_k = 1\right), \tag{15}$$

which is Expression 6: individual fitness expression.

C_1, C_2, C_3 are the weight of three features in the final fitness.

3.5. Experiment. Figure 4 shows a circuit. Its Boolean expression is

$$Y = A + BCD' + BC'D. \tag{16}$$

Its calculation is

$$Y = \begin{cases} 0, & (ABCD) < (0101)_2 \text{ or } (ABCD) = (0111)_2. \\ 1, & \text{else.} \end{cases} \tag{17}$$

The input value $(ABCD) = (0111)_2$ can be seen as the Trojan trigger conditions. When there are more pins, only part of the value can be tested, and you may miss the input $(ABCD) = (0111)_2$. In the following experiment, the input

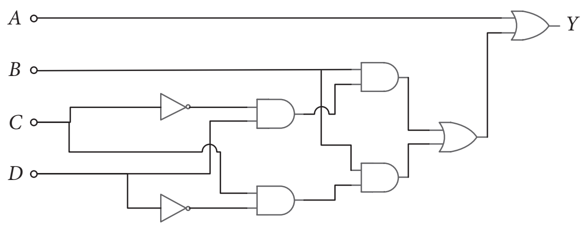


Figure 4: A circuit with a Trojan.

data will not provide the (0111) value to trigger the Trojan, and the output of the input value will be determined by the evolved circuit.

Its effective gene is

$$\text{Or}, A, \text{Or}, \text{And}, \text{And}, B, \text{And}, B, \text{And}, \text{Not}, D, C, \text{Not}, C, D. \tag{18}$$

Using different combination forms, we designed 4 groups of experiments. Considering that the logic value is required to be correct first in the circuit, the voltage value and the current value must be meaningful on the basis of the correct logic value, so the logic value is included in each group of experiments, and the fitness of the individual combines several data; the logical value accounts for a larger proportion. Table 2 shows the results of the experiments.

The experimental results show the following:

(1) Only using a single feature cannot find Trojan circuit.

(2) Using multiple features can effectively discover Trojan circuits.

(3) The features with direct correlation have no effect on the discovery probability of Trojan horse: in experiment 2, two features of logic value and voltage are used at the same time, and the Trojan horse cannot be found; in experiment 4, although three features are used, the probability of finding Trojan horse is not higher than that of real 3. The reason is that in digital circuits, the logical value itself is expressed by the voltage value; for example, the voltage value less than 3 V is considered 0, and the

Table 2: Comparison of results of 4 groups in experiment 1.

Parameter	Exp1Value	Exp2Value	Exp3Value	Exp4Value
Logic gate	And, or, not	And, or, not	And, or, not	And, or, not
Values provided	Logic values	Logic values Voltage values	Logic values Current values	Logic values Voltage values Current values
Fitness function	$F = F_1$	$F = 0.8 \cdot F_1 + 0.2 \cdot F_2$	$F = 0.8 \cdot F_1 + 0.2 \cdot F_3$	$F = 0.6 \cdot F_1 + 0.2 \cdot F_2 + 0.2 \cdot F_3$
Exercise count			100	
Trojan discovered count	0	0	72	67

voltage value greater than 3 V is considered 1. Therefore, there is no difference between the logic value and voltage value.

4. Algorithm 2: Multioutput Circuit

4.1. GEP Representation. One circuit n input/m output can be described as a forest composed of m trees, each with $1{\sim}n$ leaf nodes. The 6-input/2-output circuit in Figure 5 can be decomposed into two tree structured multi-input/single-output circuits.

The circuits shown in Figure 6 can be divided into two independent multi-input/single-output.

The corresponding effective genes are

$$\text{And, And, And}, A, \text{Not, And, And}, B, C, \text{Not}, E, F, D. \quad (19)$$

$$\text{Or, And, And, And, And, And}, F, C, \text{Not}, E, F, E, F, D. \quad (20)$$

The combination of the two genes represents a 6-input/2-output circuit.

4.2. Algorithm of Ordered Mixed Feature GEP. The GEP should be modified as the following to be able to represent this kind of circuit.

4.2.1. Remove the Link Function and Number the Gene. The GEP data structure has its own multigene structure. In formula mining, the basic idea of GEP is to use a polynomial approximation method, so that each independent gene can evolve a part of the final polynomial and then use a connection function (usually "+") to form a complete polynomial. Of course, if the test data are error-free and the cost is sufficient, GEP final expression does not need to be approximated, and it is the expression from which the test data themself come.

The operator such as "+" has a characteristic that there is no sequential difference between the operators. If such an operator is used, it can be considered that there is no sequential difference between the genes in GEP chromosome.

We can also see that GEP can solve a problem similar to $y = f(x_1, x_2, \ldots, x_n)$ function problem; that is, it can deal with the problem of multi-input and single-output. However, circuit combinations are often a multi-input/multi-output problem, that is, a problem as $(y_1, y_2, \ldots, y_m) = f(x_1, x_2, \ldots, x_n)$. This is a situation GEP cannot handle by its own algorithm.

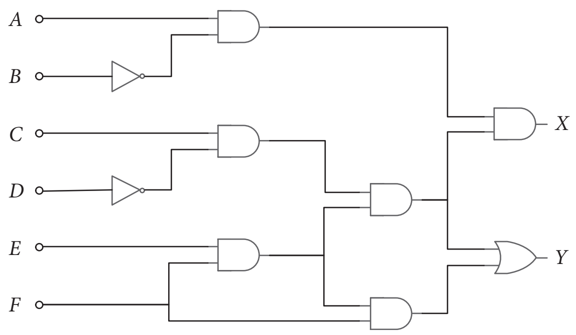


Figure 5: A 6-input/2-output circuit.

For solving the multioutput situation of combinational circuits, the GEP data structure is changed as follows:

(1) The connection function used to connect GEP to multiple genes is removed, so that a gene represents an output, and there is no association between genes; then, a chromosome with k genes represents a circuit with k outputs.

(2) According to the position number of the gene in the chromosome and the position of the gene in the chromosome, the corresponding output pin is represented; that is, the input value in the GEP is the test value of each input pin. The decoding result represents the circuit structure of an output pin. Each gene within a chromosome evolves independently.

4.2.2. Record the Fitness of Each Gene. The fitness is set for each individual in the GEP, which is the basis for the calculation of various evolutionary variations. The fitness represents the approximate degree of the target on the whole of an individual. This fitness is calculated based on the expression tree decoded by an individual.

In the work of this paper, because each gene in an individual is independent of each other, the whole individual decodes not an expression tree, but an expression forest, and the trees in this forest are still orderly. The fitness of an individual depends on each gene. To solve this problem, the fitness is set for each gene of the individual in the work of this paper. The fitness represents the similarity of the gene to the circuit structure of the corresponding pin. Combined with the fitness of all genes, an individual's fitness is formed, indicating the approximation of the individual to the whole circuit structure. Therefore, this paper not only sets the fitness for each individual but also sets the fitness for each gene.

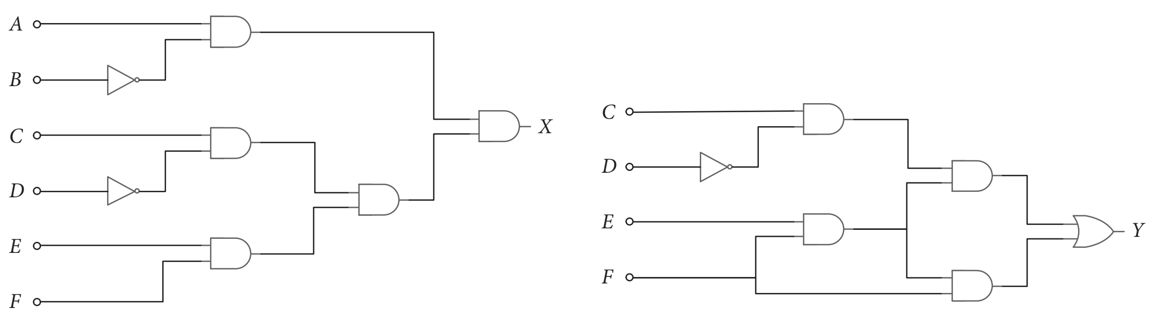


Figure 6: Two circuits divided by the circuit shown in Figure 5.

4.3. Materials and Methods. The relationship between chromosome fitness and gene fitness can be described as

$$\text{fitness}\,(\text{chromosome}) = \frac{\left(\sum \text{Fitness}\,(\text{gene})n\right)}{n}, \quad (21)$$

which is Expression 7: individual fitness of multigene GEP.

This algorithm is Ordered Mixed Feature GEP (OMF-GEP).

4.4. Experiment Setup. The setting of experiment 2 is most consistent with that of experiment 1. The difference is that the number of genes is increased to 2, corresponding to the operation of gene recombination (the probability is 0.01), and the termination condition of the algorithm is changed to 100,000 times.

The fitness of each gene is calculated in the same way as experiment 1. Expression 5 is used to calculate the fitness of the whole chromosome, in which n is 2.

4.5. Experiment. Use the circuit of Figure 5. In this circuit, Boolean expression is

$$X = AB'CD'EF, Y = EF. \quad (22)$$

Among them,

$$X = \begin{cases} 1, & (ABCDEF) < (101011)_2, \\ 0, & \text{else}. \end{cases} \quad (23)$$

During the experiment, we deliberately hide the test cases that allow X to take a value of 1. The setup of the 4 groups of experiments is completely consistent with that of experiment 1. Table 3 shows the results.

It can be seen from the experimental results that no matter what combination of features is used, after evolution begins, the fitness cannot continue to grow after reaching a very low value, and evolution has actually stopped. The overall trend is shown in Figure 7.

5. Algorithm 3: Superchromosome

In experiment 2, it is impossible to evolve continuously when the fitness is not high in the early stage of evolution. By analyzing the reasons, the fitness of the individual represents the approximate degree of the individual to the whole circuit, but in the design of the modified algorithm, each gene evolves alone. That is, the approximation of the circuit is divided into different parts. In the process of evolutionary calculation, such individuals will be considered poor individuals, with less chance of heredity in the next evolution, resulting in the loss of local genes already leading in evolution. This situation will lead to the efficiency of the algorithm evolution being very inefficient, or even unable to converge, because the evolution has entered a situation of almost random evolution.

5.1. Algorithm Description. To solve this situation, this paper introduces the concept of "superchromosome" in its work. Each individual in the population is obtained by genetic variation after initialization, but the superindividual is artificially constructed. Using the fitness of each gene that has been recorded, a superindividual is constructed after an evolution. The method is that the structure of the superindividual and the ordinary chromosome is the same, but the gene at each position is the best one in the same position in the whole population, as shown below. In this way, the superindividual concentrates the last evolutionary optimal gene at each gene location, and there is no doubt that the superindividual is the optimal individual in the population. Then, replacing the worst individuals in the population with such superchromosome to continue the later evolution can effectively avoid the elimination of local excellent genes. Figure 8 shows how to compose the superchromosome.

The introduction of the superchromosome was intended to avoid the elimination of excellent genes in the same individual due to the existence of "low quality" genes, but it brought an additional benefit. Evolutionary individual fitness changes can often reach a high level soon after evolution, as Figure 9 shows. The reason is that the superindividual concentrates the optimal genes in each gene position, so that the whole individual can achieve very high fitness.

5.2. Experimental Setup. The setting of experiment 3 is the same as that of experiment 2, but it increases the generation of superindividuals when each generation evolves. The fitness function designed is exactly as the same as experiment 2.

5.3. Experiment. The content of the experiment is the same as that of experiment 2. Table 4 shows the results.

The experimental results show the following:

Table 3: Comparison of results of 4 groups in experiment 2.

Parameter	Exp1Value	Exp2Value	Exp3Value	Exp4Value
Logic gate	And, or, not	And, or, not Logic values	And, or, not Logic values	And, or, not Logic values
Values provided	Logic values	Voltage values	Current values	Voltage values Current values
Gene fitness function	$F = F_1$	$F = 0.8 \cdot F_1 + 0.2 \cdot F_2$	$F = 0.8 \cdot F_1 + 0.2 \cdot F_3$	$F = 0.6 \cdot F_1 + 0.2 \cdot F_2 + 0.2 \cdot F_3$
Chromosome fitness	$F_{\text{chrom}} = (F_{\text{gene1}} + F_{\text{gene2}})/2$			
Evolution number	100000			
Max chrom-fitness	0.42	0.46	0.44	0.44
Max evolution no. when fitness stopped advance	92	134	112	137

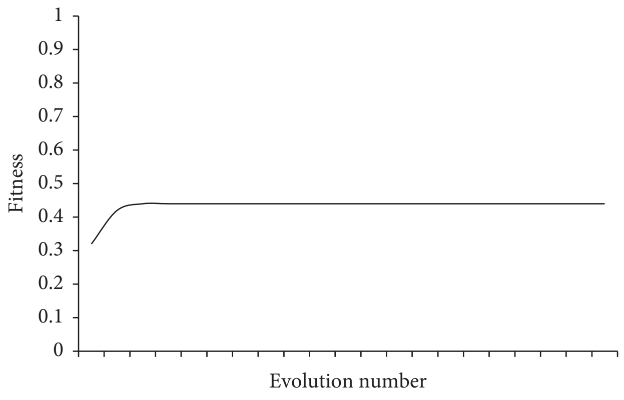


Figure 7: The algorithm cannot converge.

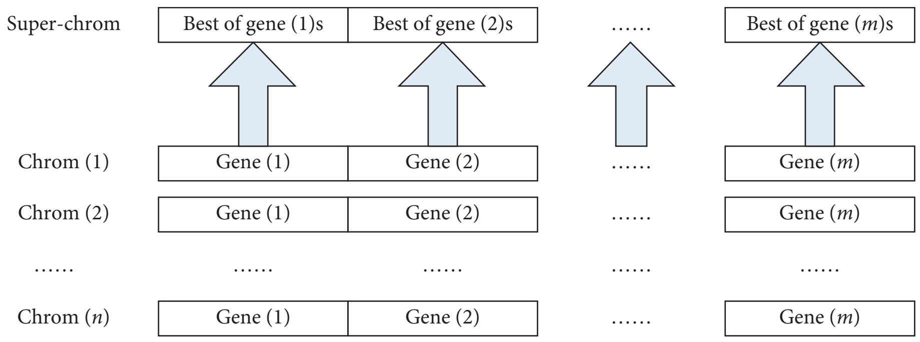


Figure 8: Composition of superchromosome.

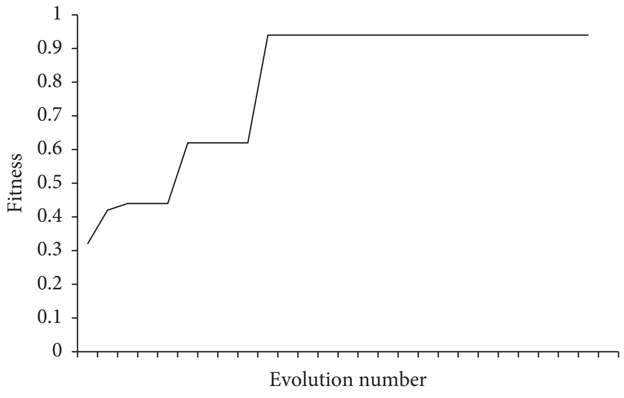


Figure 9: Superchromosome makes the fitness rise rapidly.

TABLE 4: Comparison of results of 4 groups in experiment 3.

Parameter	Exp1Value	Exp2Value	Exp3Value	Exp4Value
Logic gate	And, or, not	And, or, not	And, or, not	And, or, not
Values provided	Logic values	Logic values Voltage values	Logic values Current values	Logic values Voltage values Current values
Gene fitness function	$F = F_1$	$F = 0.8^{\cdot}F_1 + 0.2^{\cdot}F_2$	$F = 0.8^{\cdot}F_1 + 0.2^{\cdot}F_3$	$F = 0.6^{\cdot}F_1 + 0.2^{\cdot}F_2 + 0.2^{\cdot}F_3$
Chromosome fitness	$F_{\text{chrom}} = (F_{\text{gene1}} + F_{\text{gene2}})/2$			
Evolution number	100000			
Max chrom-fitness	0.72	0.76	0.94	0.87
Max evolution no. when fitness stopped advance	324	276	809	1365

(1) After using superindividuals, the evolution can reach a very high level in a very short time, and then the speed of evolution will be significantly reduced.

(2) Using only a single feature or feature with direct correlation, the algorithm is difficult to obtain satisfactory fitness, and the reason has been analyzed in the results of experimental 1.

(3) The use of multiple features that lack direct correlation between each other helps to achieve higher fitness.

6. Conclusion

GEP algorithm based on the mixed features is proposed in this paper, when multiple features with no direct correlation between each other are used, even if some important parameters are missing in the test case, and the abnormal structure in the circuit can be found. For multioutput circuits, if it only simply decomposed into multiple single-output circuits to evolve separately, the algorithm will fall into a complete random search and cannot converge. In this paper, the concept of superindividual is proposed to solve this problem, so that the algorithm can converge smoothly in the circuit structure facing multi-input.

Data Availability

The data used to support the findings of this study can be obtained from https://pan.baidu.com/s/1z29JUVHv8Qx4-uasESnKMw (pwd: 55vd).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the University Research Start-Up Funds of Chengdu University of Information Technology (KYTZ201720), the Open Project of Center for Information in Biomedicine of School of Life Sciences and Technology, University of Electronic Science and Technology of China (SYFD061902K), and Sichuan Science and Technology Program (2019YFG0196).

References

[1] F. Courbon, P. Loubet-Moundi, J. J. A. Fournier, and A. Tria, "SEMBA, a SEM based acquisition technique for fast invasive hardware Trojan detection," in *Proceedings of the 2015 European Conference on Circuit Theory and Design (ECCTD)*, pp. 1–4, Trondheim, Norway, August 2015.

[2] C. X. Bao, D. Forte, and A. Srivastava, "On the application of one-class SVM to reverse engineering-based hardware Trojan detection," in *Proceedings of the 15th International Symposium on Quality Electronic Design (ISQED)*, pp. 47–54, Santa Clara, CA, USA, March 2014.

[3] C. X. Bao, D. Forte, and A. Srivastava, "On reverse engineering-based hardware Trojan detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 49–57, 2015.

[4] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proceedings of the 2007 IEEE Symposium on Security and Piracy (SP'07)*, pp. 296–310, Berkeley, CA, USA, May 2007.

[5] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware Trojans impacting circuits delay," *IEEE Design & Test*, vol. 30, no. 2, pp. 26–34, 2013.

[6] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting Trojans through leakage current analysis using multiple supply pad I_{DDQ}s," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 893–904, 2010.

[7] A. N. Nowroz, K. Hu, F. Koushanfar, and S. Reda, "Novel techniques for high-sensitivity hardware Trojan detection using thermal and power maps," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1792–1805, 2014.

[8] B. Y. Zhou, R. Adato, M. Zangeneh et al., "Detecting hardware Trojans using backside optical imaging of embedded watermarks," in *Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC'15)*, pp. 1–6, San Francisco, CA, USA, June 2015.

[9] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2939–2948, 2017.

[10] D. Li, C. Shen, X. Dai et al., "Research on data fusion of adaptive weighted multi-source sensor," *Computers, Materials & Continua*, vol. 61, no. 3, pp. 1217–1231, 2019.

[11] Y. Jiang, W. Jiang, J. Chen et al., "A new method based on evolutionary algorithm for symbolic network weak unbalance," *Journal on Internet of Things*, vol. 1, no. 2, pp. 41–53, 2019.

[12] A. Y. Hamed, M. H. Alkinani, and M. R. Hassan, “A genetic algorithm optimization for multi-objective multicast routing,” *Intelligent Automation & Soft Computing*, vol. 26, no. 6, pp. 1201–1216, 2020.

[13] F.-I. Chou, W.-H. Ho, and C.-H. Chen, “Niche genetic algorithm for solving multiplicity problems in genetic association studies,” *Intelligent Automation & Soft Computing*, vol. 26, no. 3, pp. 501–512, 2020.

[14] H. Zhi and S. Liua, “A hybrid GABC-GA algorithm for mechanical design optimization problems,” *Intelligent Automation and Soft Computing*, vol. 25, no. 4, pp. 815–825, 2019.

[15] C. Ferreira, “Gene expression programming: a new adaptive algorithm for solving problems,” *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[16] T. Higuchi, M. Murakawa, M. Iwata et al., “Evolvable hardware at function level,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ‘97)*, pp. 187–192, Indianapolis, IN, USA, April 1997.

[17] T. Higuchi, M. Iwata, I. Kajitani et al., “Evolvable hardware and its application to pattern recognition and fault-tolerant systems,” in *Towards Evolvable Hardware*, pp. 118–135, Springer, Berlin, Germany, 1996.

[18] V. Vassilev, D. Job, and J. Miller, “Towards the automatic design of more efficient digital circuits,” in *Proceedings of the 2nd NASA/DOD Workshop on Evolvable Hardware*, pp. 151–160, Palo Alto, CA, USA, July 2000.

[19] G. W. Timothy and J. B. Peter, “Towards development in evolvable hardware,” in *Proceedings of the 3rd NASA/DOD Workshop on Evolvable Hardware Pasadena*, pp. 241–250, Alexandria, VA, USA, July 2002.

[20] M. Erbo, R. Rossi, V. Liberali, and A. G. B. Tettamanzi, “Digital filter design through simulated evolution,” in *Proceedings of the European Conference on Circuit Theory and Design*, pp. 389–393, Espoo, Finland, August 2001.

[21] H. Hemmi, J. Mizoguchi, and K. Shimohara, “Development and evolution of hard ware behaviors,” in *Proceedings of the Artificial Life IV*, pp. 250–265, Cambridge, MA, USA, July 1994.

[22] B. Hounsell and T. Arslan, “A novel evolvable hardware framework for the evolution of high performance digital circuits,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO ‘00)*, pp. 525–529, Las Vegas, NV, USA, July 2000.