Hardware Trojan Detection Method Based on Multi-featured GEP

Huan Zhang[1], Jiliu Zhou[1(✉)], Xi Wu[2], and Yi Zhang[1]

[1] College of Computer Science, Sichuan University, Chengdu 610065, China
zhoujl@cuit.edu.cn
[2] School of Computer Science,
Chengdu University of Information Technology, Chengdu 610225, China

Abstract. In the hardware Trojan detection method, destructive reverse engineering can most precisely restore the original circuit of the chip to be detected, but this method is a huge amount of work, high cost, long life cycle. In this paper, we proposed a multi-featured GEP technology, non-destructive reverse engineering of the chip using various data obtained from bypass detection, in order to restore the actual circuit of the hardware, or at least find out the unknown circuit design.

Keywords: Hardware Trojans · Trojan horse detection · Multi-featured · GEP

1 Introduction

In today's hardware Trojan detection technology [1–7], destructive reverse engineering [1, 3, 8–10] can most precisely restore the original circuit of the chip to be detected, but this technology is a huge amount of work, high cost, long life cycle, and the chip has been scrapped after detection, can only be used for sampling or replication of a class of chips, bypass detection [11–16] technology analyzes the bypass circuit signal, such as timing, power, electromagnetics, heat, etc., determine whether or not they contain Trojans, no damage to the chip, relatively small amount of data is needed, lower cost, is the most important and effective method.

Some of the work is done by evolutionary algorithms to study evolutionary hardware [17–23]. This paper proposes a multi-featured GEP evolutionary algorithm, use a single circuit component or group of circuit structures as a node, use an operator to describe a node's multi-features, use GEP to evolved circuits close to the original circuit, then determine the structure of the original circuit.

2 Brief Introduction of GEP

GEP (Gene Expression Programming) [24] combines the advantages of GA and GP, follow the basic steps of Evolutionary Computation (EC). The basic composition of its genetic material is two kinds of symbols, that is terminators and functions. A gene

X. Wu et al. (Eds.): QShine 2020, LNICST 381, pp. 22–37, 2021.
https://doi.org/10.1007/978-3-030-77569-8_2

consists of a linear, fixed-length string of symbols, code for expression trees with different sizes and shapes, a chromosome can consist of a single gene or multiple genes, each chromosome is decoded to map as a candidate solution for problem response.

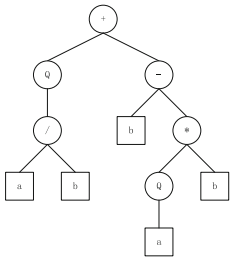
F is the set of functions and T is the set of terminals, the genes of GEP are composed of a head and a tail. The head contains symbols that represent both F and T, whereas the tail contains T. For each problem the length of the head h is chosen, whereas the length of the tail e is a function of h and the number of arguments of the function with more arguments n (also called maximum arity) and is evaluated by the equation:

$$e = h \times (n - 1) + 1 \tag{1}$$

For example, consider a gene for which the set of functions $F = \{+, -, *, /, Q\}$ and the set of terminals $T = \{a, b\}$. In this case $n = 2$, if we chose an $h = 10$, then $e = 11$, thus the length of the gene is 21, one such gene is shown below (the tail is shown in bold):

$$+Q - /b * abQbababababbaaab$$

It codes for the following expression tree (ET):



The algebraic expression:

$$\sqrt{a/b} + b - \sqrt{a} * b$$

Tail excess symbols are discarded directly without use. This allows GEP to use fixed-length encoding to express different sizes and shapes of expression trees.

3 Use GEP to Represent Circuit

GEP has done well in mining association rules, clustering, classification rules, time series predictions, sunspot predictions [25–29].

As can be seen from the data structure of GEP, GEP can solve tree structure problems very well, Other words, circuits that can be represented as a tree-shaped structure with n leaf nodes, can be described directly with GEP. The logic circuit of the 6-in/1 output in Fig. 1(a) can be easily represented as a tree structure in Fig. 1(b), It replaces the logical gate function with the logical symbol, in GEP, Its corresponding effective gene is:

And, And, And, A, Not, And, And, B, C, Not, E, F, D

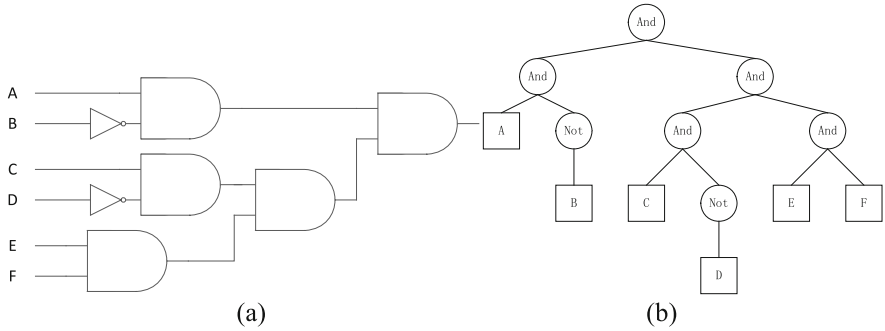


Fig. 1. 6-in/1-out circuit

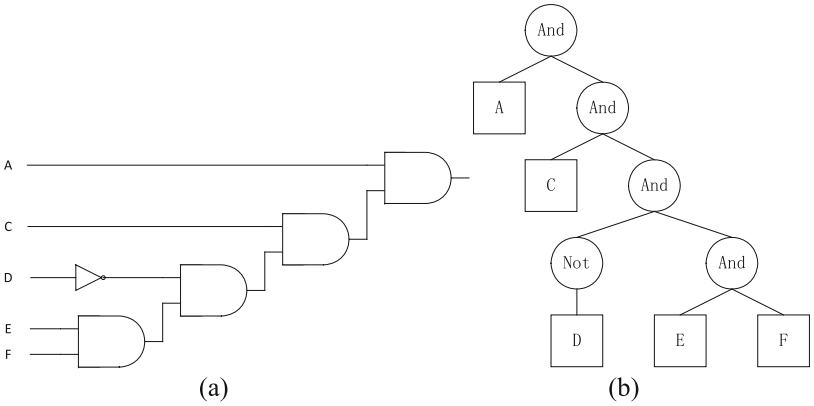


Fig. 2. Isomorphic circuit of Fig. 1

But, if only logical values are used to represent the circuit, there are too many isomorphic situations, for example, the circuit in Fig. 1 can be replaced with the circuit in Fig. 2.

The corresponding valid gene is:

$$\text{And, A, And, C, And, Not, And, D, E, F}$$

The two circuits are fully equivalent in logic values, both is:

$$Y = ACD'EF$$

And you can see, in the second circuit, input B is not used at all. That is, input B in the first circuit does not actually affect the output.

This example is just for the logic value of the circuit, In fact, for the circuit bypass information detection, there is a similar situation. The test results for any single bypass information, we can get a number of different circuit structures. This article refers to this situation as isomorphic.

Thus, only a logical value is used to learn a certain kind of bypass information to describe the circuit, so much Isomorphism that the circuit structure could not be

confirmed. In this paper, a multi-featured GEP algorithm is proposed, use the same structure in GEP to represent multiple circuit features, effectively reduces the number of isomorphic circuits, and use this algorithm to detect hardware trojans.

4 Multi-featured GEP

In this paper, the logic value of the circuit or any kind of bypass information such as voltage, current, etc. is called a characteristic value. Isomorphism for a characteristic value, essentially, the characteristics of the circuit components on this characteristic value are superimposed on each other, allows several different circuit structures to exhibit similar or even identical characteristics after the characteristic values are superimposed on each other, this makes it impossible to represent the corresponding circuit by the result of a particular characteristic value.

While we are doing the test, multiple feature values are detected at the same time, the detection results of each feature value can result in multiple isomorphic circuits, but since the overlay characteristics of different feature values cannot be identical, these homogeneous circuits cannot be exactly the same, and the same part of which is the possible real circuit.

The Multi-featured GEP algorithm can be illustrated by a Not-Gate designed by a triode in Fig. 3.

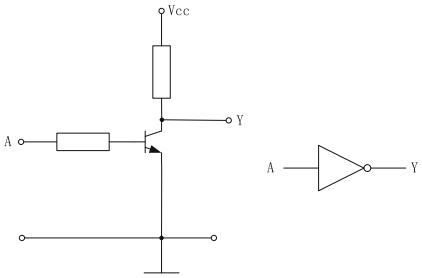


Fig. 3. A Not Gate Designed by Triode

- Feature Value 1:

  In digital logic terms, the description of this circuit is: $Y = !A$.

- Feature Value 2:

  In terms of voltage, the description of this circuit is: $V_y = V_A < V_{SH}\ ?\ V_{cc} : V_{CES}$.

  Among them: VSH represents high-level threshold, that is, the lower potential limit of "1" in the circuit. VCES represents a saturation voltage drop of the tripole.

- Feature Value 3:

In terms of current, the description of this circuit is: Cy = N * CA, where N is the magnification of the current.

In addition, there are a variety of other bypass information detection items, like delay, spectrum, etc.

The above 3 characteristic values, using any one alone can not determine the lower circuit structure, however, if the three cases are combined, the circuit structure can be determined.

In GEP, an operator represents only one calculation, a GEP individual can only represent a description of a feature value, can get an unlimited number of isosome circuits, it is difficult to determine the actual circuit structure.

This paper proposes a method to combine the test results of multiple characteristic values on a basic circuit into a single function representation, combine into a composite function, i.e. have a function represent multiple calculations, use GEP's functional evolutionary ability to evolve representations close to the original circuit.

In detail, we include these multiple detection values in a function, the input of the function is multiple feature values, the output is a vector, like this non-gate circuit. in GEP's expression tree, it is still represented by "Not.", but the implication becomes the calculation of the below vector:

$$\mathrm{Not}(\mathrm{A1},\ \mathrm{A2},\ \ldots,\ \mathrm{An}) = [\mathrm{F1}(\mathrm{A1}),\ \mathrm{F2}(\mathrm{A2}),\ \ldots,\ \mathrm{Fn}(\mathrm{An})]$$

Where Ak (k is 1,…, n) is the input value of some kind of feature detection, Fk (Ak) (k s 1, …, n) is the result of this feature value detection corresponding to the input value Ak.

For example, corresponding to this non-gate circuit, the symbol Not indicates the following meaning:

$$\mathrm{Not}(\mathrm{A},\ \mathrm{V_A},\ \mathrm{C_A}) = [!\mathrm{A},\ \mathrm{V_A} < \mathrm{V_{SH}}?\,\mathrm{Vcc} : \mathrm{V_{CES}},\ \mathrm{N} * \mathrm{C_A}]$$

Where A represents the logical value (1 or 0) represented by the voltage entered at point A, The VA represents the input voltage of point A and the CA represents the input current of point A.

Thus, when using GEP evolution, a single symbol Not can also represent multiple feature value tests that are not associated with each other.

5 Experiments

5.1 Experimental Settings

This paper has designed 4 experiments (Table 1).

1. Parameters
- The circuit parameters are:
  - Output m = 1
  - The number of features is k = 1, 2, 2, 3
  - 3 features are used: feature 1 is logical, feature 2 is voltage value, feature 3 is current value.

- As a comparison experiment, the parameters used are exactly the same:

Table 1. Experiment parameters

Parameter	Value
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 recombination rate	0.1
Input number	4
Output number	1
Function set	Not, And, Or

2. Fitness function

The characteristic data are logical data, voltage data and current data, which have their respective Fitness function:

a. Logical Fitness function:

$$F_1 = 1 - \sum_{i=1}^{N} |y_i - y|/N \tag{2}$$

N is the test data count, yi is the logical value calculated based on the test data after decoding the individual, y is the output logic values for test data. Because of the logic value, so the worst case is that each output after the individual decodes is the opposite of the test value, that is $|y_i - y| = 1$, so the value of the F1 is in [0,1].

b. Voltage Fitness function

$$F_2 = 1 - (\sum_{i=1}^{N} |y_i - y|/N)/(V_{CC} - V_{DD}) \tag{3}$$

N is the test data count, yi is the voltage value calculated based on the test data after decoding the individual, y is the output voltage values for test data. The worst case is, each test output value is either the highest level or the lowest level, and each output that is decoded by an individual is the opposite of the test value that is $|y_i - y| = V_{cc} -$ VDD, so the value of the F2 is in [0,1].

c. Current Fitness function

$$F3 = 1 - SSE/SST \tag{4}$$

$$SSE = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

$$SST = \sum_{i=1}^{m} (y_i - \overline{y}_i)^2$$

N is the test data count, $\hat{y}_i$ is estimate of yi calculated from Ti using formula, $\overline{y}_i$ is the average of the y, SSE is Residual Sum of Squares, SST is Sum of Squares of Deviations, F3 is the square of the multiple correlation coefficient in statistics, its value is also in [0,1].

d. Individual's Fitness

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

$$F = C_1 * F_1 + C_2 * F_2 + C_3 * F_3 \tag{5}$$

C1, C2, C3 is weights of 3 feature values in final fitness.

5.2 Experimental Results

Figure 4 is a circuit that has no Trojan.

Its boolean expression is

$$Y = A + BC + BD \tag{6}$$

The circuit dose the computation

$$Y = \begin{cases} 0, \ (ABCD) < 5 \\ 1, \ else \end{cases} \tag{7}$$

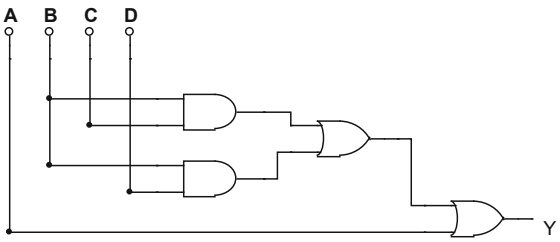


Fig. 4. Circuit has No Trojan

After several logic gates added to the circuit of the Fig. 1, it becomes a circuit has trojan (Fig. 5). In the new circuit that Y will also get a value of 0 when (ABCD) = 7:

$$Y = \begin{cases} 0, \ (ABCD) < 5 \ or \ (ABCD) = 7 \\ 1, \ else \end{cases} \tag{8}$$

Its Boolean expression becomes:

$$Y = A + BCD' + BC'D \tag{9}$$

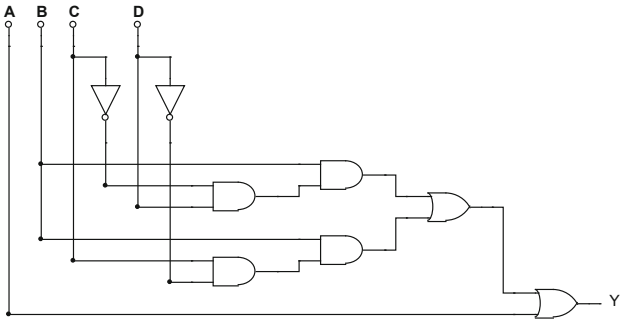


Fig. 5. Circuit with Trojan

The activate gene of the circuit is: Or, A, Or, And, And, B, And, B, And, Not, D, C, Not, C, D

If the pins are so many that we can only test part of the values, the input value to activate the trojan (ABCD) = $(0111)_2$ may be missed at this time. In the following experiment, the input value $(0111)_2$ will not be provided, and the output of this input value will be determined by the evolved circuit.

Using different combinations, we designed 4 sets of experiments. Considering that the logic values are required to be correct in the circuit first, the voltage and current values must be based on the correct logic values in order to make sense. The individual's fitness is a combination of several data, in which the proportion of logical value is larger.

- **Experiment 1** (Table 2)

Table 2. Setting of the Experiment 1

Parameter	Value
Logic Gate	And, or, not
Values Provided	Logic Values
Fitness function	$F = F_1$
Exercise Count	100
Trojan Discovered Count	0

None of the 100 times exercise is able to discover the Trojan circuit if only the logical values were provided. Figure 6 shows some typical results.

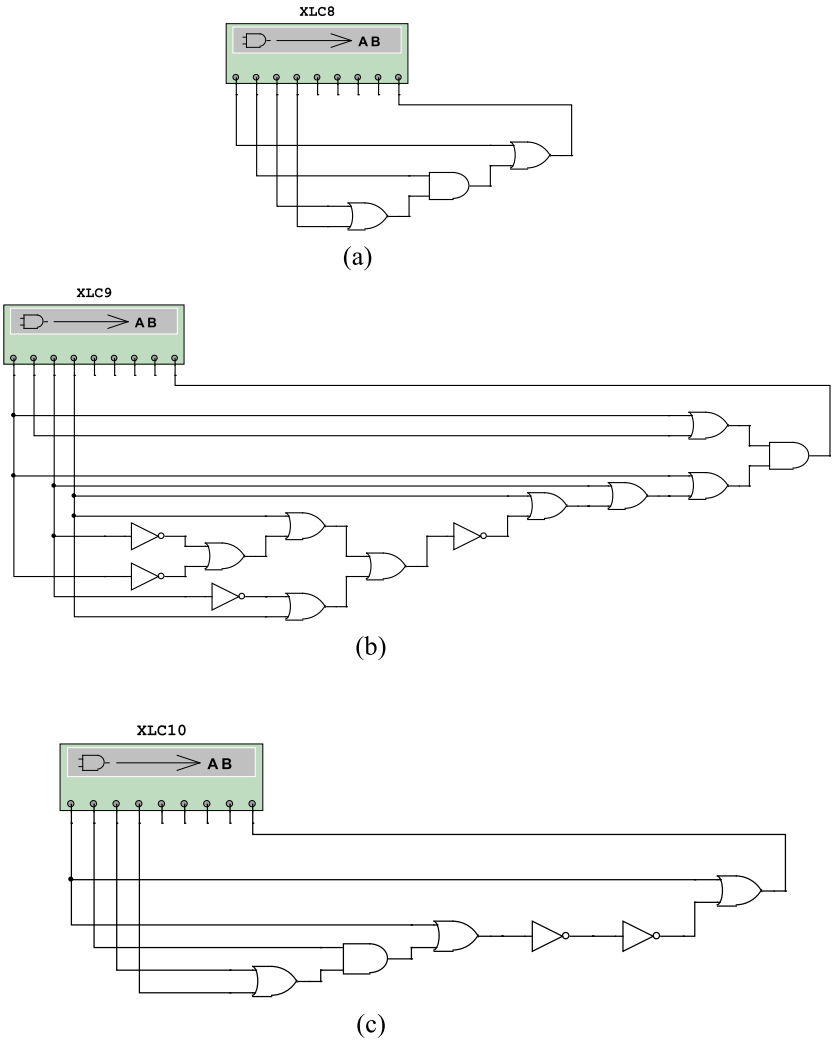


Fig. 6. Some Circuits evolved by Multi-featured GEP in Experiment 1

The simplified Boolean Expression of all the circuits above is:

$$Y = A + BC + BD$$

- **Experiment 2** (Table 3)

Table 3. Setting of the Experiment 2

Parameter	Value
Logic Gate	And, or, not
Values Provided	Logic Values, Voltage Values
Fitness function	F = 0.8 * F1 + 0.2 * F2
Exercise Count	100
Trojan Discovered Count	0

None of the 100 times exercise is able to discover the Trojan circuit if both the logical values and the voltage values were provided. Figure 7 shows some typical results.

The simplified Boolean Expression of all the circuits above is:

$$Y = A + BC + BD$$

The Trojan still cannot be discovered although both the logical value and the voltage value were provided. The reason is in digital circuits, the logic value itself is expressed in terms of voltage values, for example, a voltage value less than 3 V is considered to be 0, a voltage value greater than 3 V is considered to be 1. So the choice of logical value and voltage value as feature values on this issue is as same as only provided the logical value.

- **Experiment 3** (Table 4)

The Trojan circuit was discovered 72 times among the 100 times exercise when the logical values and the voltage values were provided. Figure 8 shows some typical results.

Both circuits' simplified Boolean Expression is

$$Y = A + BCD' + BC'D$$

The equivalent circuit has been discovered although the original circuit has hidden.

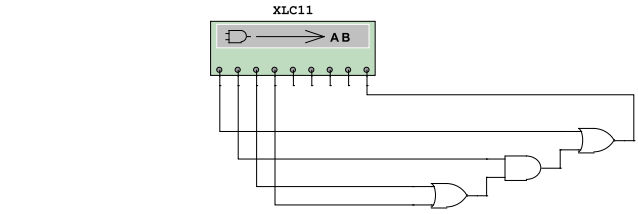
- **Experiment 4** (Table 5)

The Trojan circuit was discovered 67 times among the 100 times exercise when three feature values were all provided. Figure 9 shows a different result.

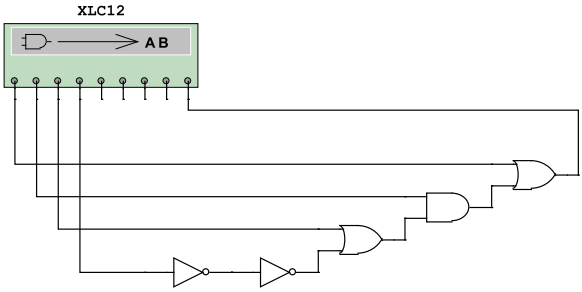
The simplified Boolean Expression is:

$$Y = A + BCD' + BC'D$$

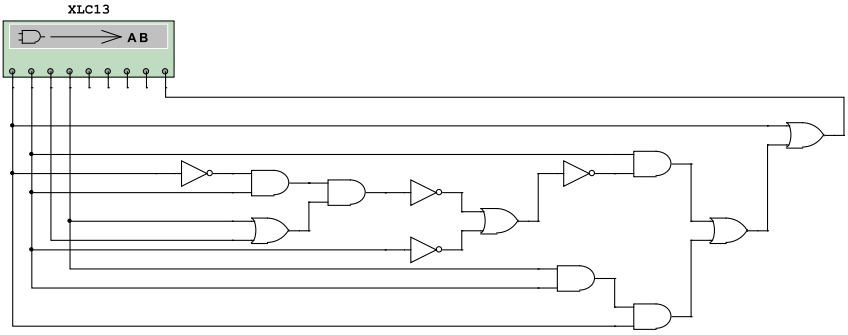
A circuit equivalent to the original circuit has been found.



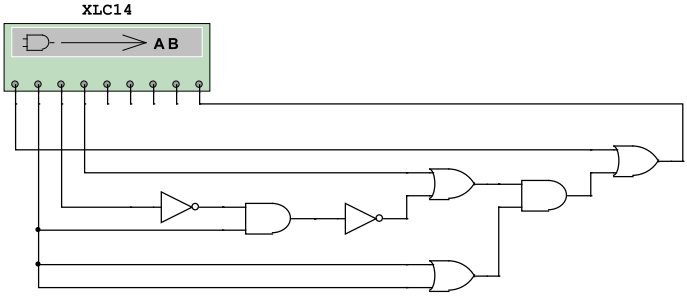
(a)



(b)



(c)



(d)

Fig. 7. Some Circuits evolved by Multi-featured GEP in Experiment 2

Table 4. Setting of the Experiment 3

Parameter	Value
Logic Gate	And, or, not
Values Provided	Logic Values, Current Values
Fitness function	F = 0.8 * F1 + 0.2 * F3
Exercise Count	100
Trojan Discovered Count	72

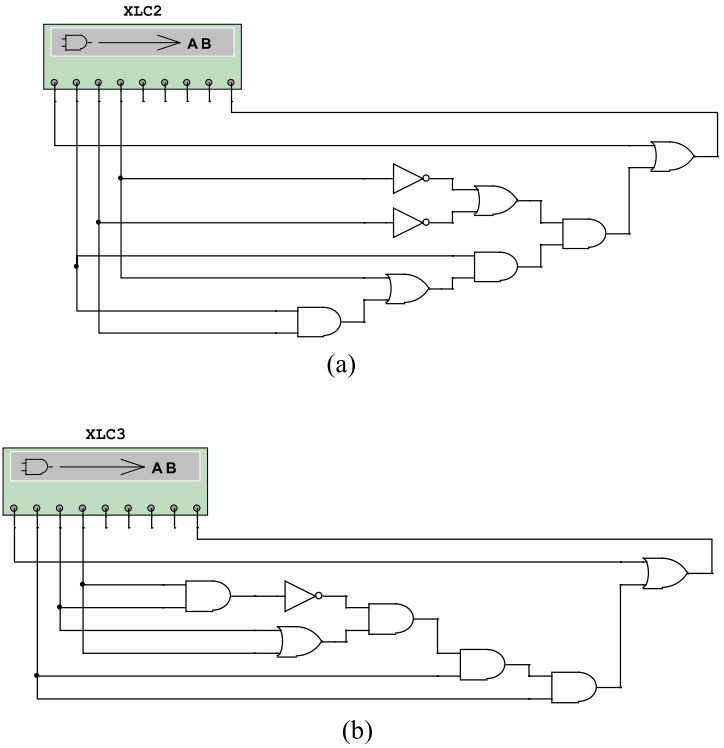


Fig. 8. Some Circuits evolved by Multi-featured GEP in Experiment 3

In this group of experiments, three feature values were used, but the efficiency of discovering the Trojan was similar to experiment 3 which used only two feature values. The reason is since the logical value itself is expressed in terms of voltage values, then the three feature values are equal to the two feature values.

Table 5. Setting of the Experiment 4

Parameter	Value
Logic Gate	And, or, not
Values Provided	Logic Values, Voltage Values Current Values
Fitness function	F = 0.6 * F1 + 0.2 * F2 + 0.2 * F3
Exercise Count	100
Trojan Discovered Count	67

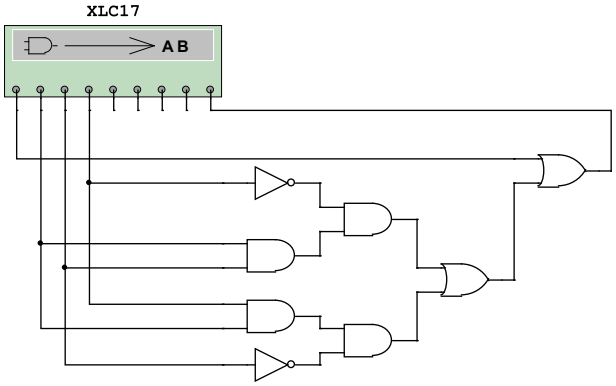


Fig. 9. Typical Circuit evolved by Multi-featured GEP in Experiment 4

6 Conclusions

This paper has proposed multi-featured GEP algorithm, when multiple feature values were fused into the same operator, there is a certain probability that GEP can automatically discover the evolutionary power of mathematical formulas to discover Trojan circuits. The fewer features used, the more efficient GEP evolves, but farther away the conclusion is from the real circuit. On the otherwise, the more features used, the less efficient GEP evolves, but the conclusion is closer to the real circuit. However, if there is a direct conversion relationship between the multiple feature values used, the characteristic values can be considered one, and this situation does not increase the evolutionary accuracy of GEP.

Acknowledgment. This work was supported by the National Key Research and Development Program of China (Grant No. 2018YFB1502803).

References

1. Xiao, K., Forte, D., Jin, Y., et al.: Hardware Trojans: lessons learned after one decade of research

2. Antonopoulos, A., Kapatsori, C., Makris, Y.: Trusted analog/mixed-signal/RF ICs: a survey and a perspective. IEEE Des. Test **34**(6), 63–76 (2017)
3. Bhunia, S., Hsiao, M.S., Banga, M., et al.: Hardware Trojan attacks: threat analysis and countermeasures. Proc. IEEE **102**(8), 1229–1247 (2014)
4. ACM Trans. Des. Autom. Electron. Syst. (TODAES) **22**(1), 6:1–6:23 (2016)
5. Tehranipoor, T., Koushanfar, F.: A survey of hardware Trojan taxonomy and detection. IEEE Des. Test **27**(1), 10–25 (2010)
6. Jian-Feng, Z., Gang, S.: A survey on the studies of hardware Trojan. J. CyberSecur. **2**(1), 74–90 (2017)
7. Bhasin, S., Regazzoni, F.: A survey on hardware Trojan detection techniques. In: Proceedings of the 2015 IEEE International Symposium on Circuits and Systems (ISCAS). Lisbon, Portugal, pp. 2021–2024 (2015)
8. Courbon, F., Loubet-Moundi, P., Fournier, J.J.A., et al.: 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), Trondheim, Norway, pp. 1–4 (2015)
9. Bao, C.X., Forte, D., Srivastava, A.: On application of one-class SVM to reverse engineering-based hardware Trojan detection. In: Proceedings of the 2014 15th International Symposium on Quality Electronic Design (ISQED), Santa Clara, USA, pp. 47–54 (2014)
10. Bao, C.X., Forte, D., Srivastava, A.: On reverse engineering-based hardware Trojan detection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **35**(1), 49–57 (2015)
11. Agrawal, D., Baktir, S., Karakoyunlu, D., et al.: Trojan detection using IC fingerprinting. In: Proceedings of the 2007 IEEE Symposium on Security and Piracy (SP 2007), Berkeley, USA, pp. 296–310 (2007)
12. Xiao, K., Zhang, X.H., Tehranipoor, M.: A clock sweeping technique for detecting hardware Trojans impacting circuits delay. IEEE Des. Test **30**(2), 26–34 (2013)
13. Aarestad, J., Acharyya, D., Rad, R., et al.: Detecting Trojans through leakage current analysis using multiple supply pad IDDQ. IEEE Trans. Inf. Forensics Secur. **5**(4), 893–904 (2010)
14. Nowroz, A.N., Hu, K.Q., Koushanfar, F., et al.: Novel techniques for high-sensitivity hardware Trojan detection using thermal and power maps. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **33**(12), 1792–1805 (2014)
15. Zhou, B.Y., Adato, R., Zangeneh, M., 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 2015), San Francisco, USA, pp. 1–6 (2015)
16. He, J.J., Zhao, Y.Q., Guo, X.L., et al.: Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25**(10), 2939–2948 (2017)
17. Higuchi, T., Murakawa, M., Iwata, M., et al.: Evolvable hardware at function level. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 187–192 (1997)
18. Higuchi, T., Iwata, M., Kajitani, I., Iba, H., Hirao, Y., Furuya, T., Manderick, B.: Evolvable hardware and its application to pattern recognition and fault-tolerant systems. In: Sanchez, E., Tomassini, M. (eds.) TEH 1995. LNCS, vol. 1062, pp. 118–135. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61093-6_6
19. Vassilev, V., Job, D., Miller, J.: Towards the automatic design of more efficient digital circuits. In: Proceedings of the 2nd NASA/DOD Workshop on Evolvable Hardware, pp. 151–160 (2000)
20. Timothy, G.W., Peter, J.B.: Towards development in evolvable hardware. In: Proceedings of the 3rd NASA/DOD Workshop on Evolvable Hardware Pasadena, pp. 241–250 (2002)
21. Erba, M., Rossi, R., Liberali, V., et al.: Digital filter design through simulated evolution. In: Proceedings of the European Conference on Circuit Theory and Design, pp. 389–393 (2001)

22. Hemmi, H., Mizoguchi, J., Shimohara, K.: Development and evolution of hard ware behaviors. In: Proceedings of the Artificial Life IV, pp. 250–265 (1994)
23. Hounsell, B., Arslan, T.: A novel evolvable hardware framework for the evolution of high performance digital circuits. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 525–529 (2000)
24. Ferreira, C.: Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst. **13**(2), 87–129 (2001)
25. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Springer, New York (2002)
26. Zuo, J., Tang, C., Zhang, T.: Mining predicate association rule by gene expression programming. In: Meng, X., Su, J., Wang, Y. (eds.) WAIM 2002. LNCS, vol. 2419, pp. 92–103. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45703-8_9
27. Zuo, J., Tang, C.-J., Li, C., Yuan, C., Chen, A.-L.: Time series prediction based on gene expression programming. In: Li, Q., Wang, G., Feng, L. (eds.) WAIM 2004. LNCS, vol. 3129, pp. 55–64. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27772-9_7
28. Zhou, C., Xiao, W.M., Tirpak, T.M., et al.: Evolution accurate and compact classification rules with gene expression programming. IEEE Trans. Evol. Comput. **7**(6), 519–531 (2003)
29. Lopes, H.S., Weinert, W.R.: EGIPSYS: an enhanced gene expression programming approach for symbolic regression problems. Int. J. Appl. Math. Comput. Sci. **14**(3), 375–384 (2004)