ELEKTROTEHNIŠKI VESTNIK 80(5): 266-270, 2013
ORIGINAL SCIENTIFIC PAPER

A Practical Test Patterns Generation Technique for Hardware Trojan Detection

Lei Fang, Lei Li, Zhen Li

Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, Chengdu, China
Email: 275993425@qq.com

Abstract. Due to the globalization of the integrated circuit manufacturing industry, hardware Trojans constitutes an increasingly probable threat to both military and commercial applications. The Trojans being stealthy in nature, the Trojan hardware is hard to be triggered under random patterns. In this paper, we propose practical test patterns generation technique based on rare logic conditions at internal nodes. The technique increases the probability of the inserted Trojans getting triggered and detected by logic testing. Simulation results for benchmarks show that the proposed test pattern generation technique can make hardware Trojan more frequently triggered than random patterns.

Keywords: hardware Trojan, test patterns, integrated circuit, benchmarks

Generacija testnih vzorcev za odkrivanje trojanskega vezja

Vedno večja globalizacija na področju izdelave integriranih vezij nudi tudi večjo možnost namernega vnosa napak. Eden izmed takšnih namernih vnosov napačnih funkcij v vezje je dodatek trojanskega vezja. Trojansko vezje je zelo težko odkriti z naključnim naborom testnih vzorcev. V tem članku predlagamo generacijo testnih vzorcev, ki temelji na analizi prehodov med logičnimi vrednostmi na posameznih notranjih vozliščih v vezju. S predlagano metodo povečamo verjetnost odkritja trojanskega vezja med postopkom testiranja. Rezultati, dobljeni med postopkom simulacije, potrjujejo, da na ta način lažje odkrijemo trojansko vezje v primerjavi z naključno generiranimi testnimi vzorci.

1 INTRODUCTION

The hardware Trojan is an emerging problem in semiconductor integrated circuit (IC) security [1]. This issue has been become prominent recently due to outsourcing of the chip manufacturing processes to reduce cost. The hardware Trojan is a tiny malicious circuit which can disable or even destroy a system at some future time, and it may leak confidential information covertly to the adversary. This may create a major risk for semi-conductor systems in critical applications such as military, communications, nuclear and space facilities.

Various post-silicon techniques have been proposed in the past few years to tackle the detection of hardware Trojan manufactured abroad. In [2], [3], [4] authors have used power as the side-channel signal to detect the hardware Trojan in the design. In [5] voltage switching on supply rails has been proposed to change the circuit logic thereby making the hardware Trojan more observable. In [6], [7] the authors have exploited additional gate delay introduced by the hardware Trojan to alter the delay signature of the path on which it embedded.

Hardware Trojan detection makes efficient pattern generation necessary to disclose Trojan impact on design characteristics beyond environmental and process variations. Trojan detection methods using power as the side-channel signal [2]-[4] require patterns that increase hardware Trojan activity whereas keep circuit activity low to magnify hardware Trojan contribution into the circuit power consumption. Methods that are based on additional gate delay introduced by the hardware Trojan [6] require patterns that generate transition on nets that supply Trojan inputs to reveal wiring and input gate resistance and capacitance impact of hardware Trojan on the circuit delay characteristic. From authentication standpoint, it is critical to analyze time to generate a transition at hardware Trojan input and in Trojan circuit and reduce detection time.

Hardware Trojan detection is an extremely challenging problem that traditional functional and structural tests can't effectively locate it. Trojan circuits have stealthy nature and are triggered in rare conditions. Trojans are designed such that they are silent most of their life time and may have tiny size relative to their design, with featuring limited contribution into design characteristics [8]. If a portion of the Trojan circuit is activated, it will consume more dynamic power and thus

Received 29 October 2013
Accepted 29 November 2013

make it easier to differentiate the waveform of a Trojan-inserted circuit from that of a Trojan-free circuit [9]. Random patterns active method has been implemented to attempt to activate Trojans, regardless of where in the circuits they might be located. However, the hardware Trojan is hard to be triggered under random patterns. In this paper, we propose practical test patterns generation technique based on rare logic conditions at internal nodes. This technique increases the probability of inserted Trojans getting triggered. This method can accelerate the Trojan detection process and have been combined with power analysis during implementation in some cases.

This paper is organized as follows. In section 2 we will introduce previous work. In section 3 we will introduce the practicality test patterns generation technique, followed by the experimental evaluations in section 4. The conclusions are given in section 5.

2 Previous Work

In [10], the authors propose test patterns generation technique based on multiple excitation of rare logic conditions at internal nodes. Through their mathematical analysis, we can acquire conclusion that the expected number of times the Trojan trigger condition is satisfied increase with the number of times the trigger nodes have been individually excited to their rare values. Then they design the reduced test pattern generation process for hardware Trojan detection. The process as follows:

1) For each random pattern, count the number of nodes whose rare value is satisfied.
2) In the following, considering each vector and modify it by perturbing one bit at a time. If a modified test pattern increases the number of nodes satisfying their rare values, the patterns will be accepted.
3) The process is repeated until each node satisfies its value at least N times. The output of the test generation process is a minimal test set that improves the coverage for hardware Trojans compared to random patterns.

Simulation results for a set of ISCAS benchmarks show that the proposed test generation approach can achieve comparable Trojan detection coverage with about 85% reduction in test length on average over random patterns. However, simulation has to be implemented once for each vector is perturbed one bit at a time in order to acquire the number of nodes satisfying their values. If a chip have 10 inputs and the length of test vector is 100. This process need be executed a long time due to the simulation has to be implemented 1000 times. When the circuit scale is large, this test pattern generation technique will be almost can't implement. So we have to design a practicality test patterns generation technique that does not depend on simulation.

In [8], the authors proposed a novel technique for improving hardware Trojan detection and reducing Trojan activation time. This novel technique estimates the transition probability that modeled by geometric distribution (GD) rather than the simulation. Test patterns generation technique based on GD will be analyzed in next section.

3 Test Patterns Generation Technique for Hardware Trojan Detection

3.1 Test Generation Technique Analysis

The GD is a discrete distribution for n=0,1,2,... with the probability function $p(n) = P \times (1-P)^n$. The probability function states that after n clock cycles, finally in the $(n+1)th$ clock cycle, there will be a transition, i.e., $(n+1)th$ trial is the first success. The average number of experiments is $(P^{-1}-1)$ which indicates the number of required clock cycles to generate a transition.

For the Trojan circuit shown in Fig.1, the calculation based on GD shows that every 256 clocks will generate a transition. The simulation results show that 250 transitions needed after applying 1000 random vectors. Considering random vectors are applied, we could convince that the calculated probability based on GD is close to the simulation results.

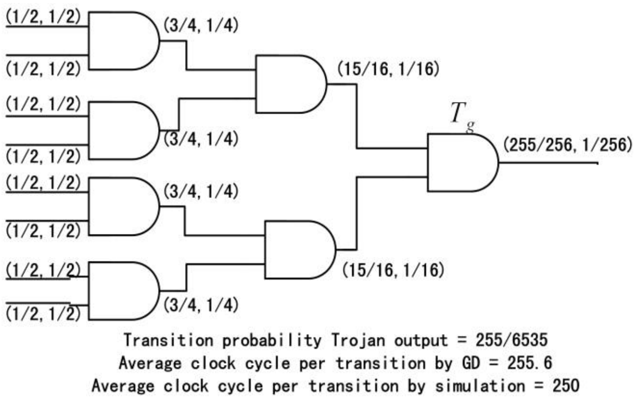

Figure 1. Comparing mathematical and simulation results

Fig.2 presents the same Trojan circuit, but test vectors are generated with the probability of 9/10 for "1". The calculation based on GD shows that four clocks are required to generate a transition at the output. On average, five clock cycles are required to generate a transition at the output after applying 1000 test vectors. In contrast to Fig 1, the T_g gate can be triggered more times. So we propose test patterns generation technique based on GD.

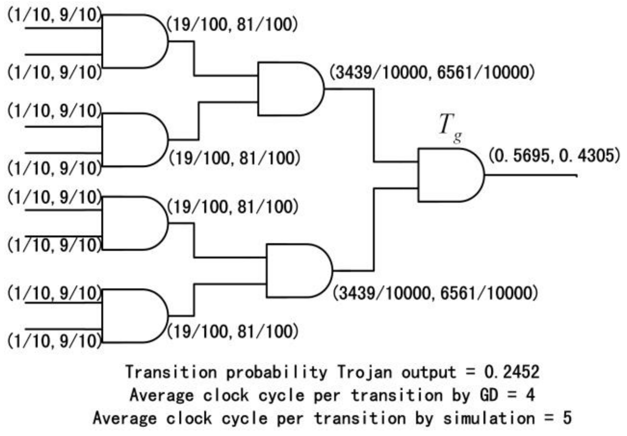
(1/10, 9/10)
(19/100, 81/100)
(3439/10000, 6561/10000)
T_g
(0. 5695, 0. 4305)
Transition probability Trojan output = 0.2452
Average clock cycle per transition by GD = 4
Average clock cycle per transition by simulation = 5

Figure 2. Test patterns generation technique example

Considering the circuit is shown in Fig.3, the calculation based on GD shows that transition probability on T_g output is 127/16384. The average clock cycle per transition by GD is 129 times. However, the simulation result is in each 44 clock cycles a transition was generated. This calculation error is due to the correlation of inputs of T_g gate. In the following, correction method will be proposed.

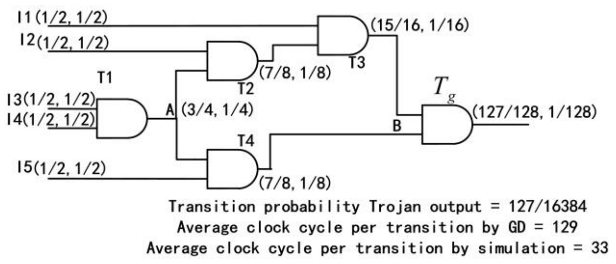
I1 (1/2, 1/2)
I2(1/2, 1/2)
T1
I3(1/2, 1/2)
I4(1/2, 1/2)
A (3/4, 1/4)
T2
(7/8, 1/8)
(15/16, 1/16)
T3
T_g
B
(127/128, 1/128)
T4
I5(1/2, 1/2)
(7/8, 1/8)
Transition probability Trojan output = 127/16384
Average clock cycle per transition by GD = 129
Average clock cycle per transition by simulation = 33

Figure 3. Modified Transition probability calculate example

First, seek the node A which will produce relevant input. Second, locate the node B that has correlation inputs. Then find out the path between A and B. Finally we derive correction formula according to the type of gates in the path and conditional probability formula.

$$\begin{aligned} P1_{Tg} &= P1_{T3} \times P1_{T4} \\ &= \mathrm{P}1_{I1} \times P1_{T2} \times P1_{T4} \\ &= \mathrm{P}1_{I1} \times \mathrm{P}1_{I2} \times \mathrm{P}1_{A} \times \mathrm{P}1_{I5} \\ &= \mathrm{P}1_{I1} \times \mathrm{P}1_{I2} \times \mathrm{P}1_{I3} \times \mathrm{P}1_{I4} \times \mathrm{P}1_{I5} \\ &= \frac{1}{32} \ldots\ldots \end{aligned} \tag{1}$$

$$PT_{Tg} = P1_{Tg} \times P0_{Tg} = \frac{31}{1024} \tag{2}$$

We can get corrected T_g gate transition probability is 31/1024. The result calculated by the correction formula is that 33.03 clocks are required to generate a transition at the output, which is consistent with the simulation result.

In the following, two algorithms are proposed. One realizes transition probability calculation correction formula, another generates test patterns for hardware Trojan detection.

3.2 Test Generation Algorithm

Algorithm 1 shows the calculation procedure of modified transition probability. First, the circuit netlist is read and mapped to a special data type that could reflect the relationship of cells and nets. In the following, we will find the node A which will make transition probability error. Then the transition probability calculation is initialized and the path in the design is acquired. After that, acquire the node B which is related to the node A and find out the path between A and B. Finally we derive modified formula according to the type of gates in the path and calculate correct transition probability.

Table 1. Calculation procedure of modified transition probability

Algorithm 1

```
01: CurrentDesign = SetDesign(Design)
02: CellNet = ElementGet(CurrentDesign)
03: Node = CellNetCreate(CellNet)
04: ProbabilityInit = ProInitialize(CurrentDesign)
05: for Node in CurrentDesign  do
06:   Chain = NetChain(Node)
07:   RelateNode = RelateFind(Chain)
08:   RelateChain = RelateChain(RelateNode)
09:   Modify = ModifyPara(RelateChain)
10: end for
11: Probability = NetsTransitionProbability(ProbabilityInit;Modify)
Output: Transition Probability Correction Formula
```

Algorithm 2 shows the major steps in the proposed test patterns generation process for Trojan detection. We start with the golden circuit without any Trojan, transition probability threshold Pth and list of rare nodes. First, the circuit netlist is read and mapped to a special data type. We calculate transition probability for random patterns and acquire list of rare nodes (LRN) according to transition probability threshold Pth. In the next step, we consider every input in the design and modify probability of “1” and “0” of test vectors. If a modified test patterns increases nodes of LRN transition probability, we accept the modified pattern. The process is repeated until transition probability of each node in LRN is increased. The output of the test generation process is test vectors that improve transition probability of rare nodes compared to random patterns.

Table 2.Test patterns generation process for Trojan detection

```
Algorithm 2
01: CurrentDesign = SetDesign(Design)
02: Probability = NetsTransitionProModify(Vector)
03: LRN = Frequency(0; Pth)
04: for all input do
05:   change input and acquire CurrentVector
06:   ProbabilityCur=NetsTransitionProModify(CurrentVector)
07:   if ProbabilityCur(LRN) > Probability(LRN) then
08:     Vector = CurrentVector
09:   end if
10:end for
Output: Test Patterns
```

4 Experiments

In our experiments, we insert two different Trojan circuits into different benchmarks. The Trojan circuits are triggered conditionally and looking for rare activated conditions. The outputs of hardware Trojan pass to the circuit and cause functional failures.

4.1 Experimental Setup

In general, a functional Trojan consists of two parts: Trigger and Payload [11]. A portion of the trigger mechanism is to activate hardware Trojan, and the payload portion is the Trojan functioning circuit after triggered. Once triggered, Trojan will send one or more signals to the payload; the payload portion begins working, thus playing a role in the realization of offensive purpose, such as leak information or the destruction of the chip.

Three programs are developed to carry out experiments. The first program acquires correction formula which will be used in the second program. The second program generates test patterns through calculating transition probabilities of nets. The third program enumerates transition in the circuit after applying test patterns or random patterns. The first and second programs are written using Perl. They read design and calculate probabilities “1” and “0” of each net. Each net is either primary input or output of a gate. Probability of “1” and “0” for primary inputs are changed according to probabilities of internal nets and for the gates output they are calculated based on the functionality of gates. Finally transition probability of a net is the product of probabilities “1” and “0” of the net. The third program uses Cadence NC-Verilog and applies random patterns or test patterns to monitor and record any transitions on any net of circuit.

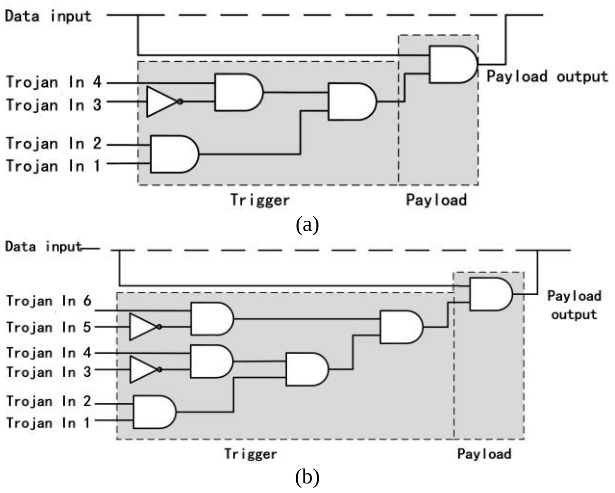

Figure 4. Trojan circuits (a) Trojan 1. (b) Trojan 2.

As in Fig.4, Payload inputs come from Trigger output and data input which part of the original circuit. The comparators look for rare combinations of Trojan inputs based on their “1” and “0” probabilities. Payload gates are selected based on Trojan circuit output’s dominant values. Dash lines above Trojan circuit in the figure represent the connection in the original circuit which is assumed to be restitched through Trojan’s payload by adversary.

Table 3. Test patterns generation process for Trojan detection

Benchmarks	Trojan	Random patterns		Test patterns	
		Total number of Transitions	Transitions inside Trojan circuit	Total number of Transitions	Transitions inside Trojan circuit
C432	Trojan1	7511215	942	6159196	1173
	Trojan2	7511184	1058	6159320	1375
C499	Trojan1	9776744	613	8212464	858
	Trojan2	9776802	704	8212479	1025
C880	Trojan1	14947157	478	12705083	662
	Trojan2	14947269	584	12705284	798
C1355	Trojan1	22821326	247	18485274	322
	Trojan2	22821378	356	18485431	467
C1980	Trojan1	25369089	125	21817416	175
	Trojan2	25369125	164	21817638	211

4.2 Experimental Results

Table 3 compares the triggered times in hardware Trojan between random patterns and test patterns with 100000 vectors. The experimental results are acquired from some ISCAS85 benchmarks. The benchmark c432 is the smallest circuit which contains 160 gates with 36 inputs, and the benchmark c1908 is the biggest circuit which contains 880 gates with 33 inputs. The results show that the transitions inside Trojan circuit decrease with the scale of circuit increasing due to the nodes in circuit is harder to trigger. Total number of transitions inside circuit generated by test patterns is less than random patterns, and the reason is probability of "1" and "0" for inputs are changed.

Furthermore, we can see that test patterns can significantly increase Trojan circuit switching activity about 30% than random patterns. Although Trojan circuits could not be fully activated and cause functional failures in simulation after applying test patterns, Trojan circuits more contribute into side-channel signals due to Trojan circuit switching activity increasing and the total number of transitions decreasing.

5 Conclusion

In this paper, we propose a practical test pattern generation technique based on rare logic conditions at internal nodes. In the following, test patterns are generated through change the probability "0" and "1" for primary inputs. Simulation results show that the proposed test generation approach makes hardware Trojan more triggered than random patterns. Although test patterns can't full active Trojan so that it impacts the circuit output and causes malfunction, it can generate more transitions inside Trojan circuit so that it improves the effectiveness of transient power-based methods. Future work will involve improving the test patterns quality which will help in increasing more transitions inside Trojan circuit.

References

[1] Dapra baa06-04, "Trust for Integrated Circuits", http://www.darpa.mil/BAA/

[2] D. Agarwal, S. Baktir, D. Karakoyunlu, P.Rohtagi and B. Sunar, "Trojan Deteciton using IC Fingerprinting", Symp. on Security & Privacy'07, pp. 296-310.

[3] M. Banga and M. Hsiao, "A Region Based Approach for the Detection of Hardware Trojans", HOST'08, pp. 43-50.

[4] M. Banga and M. Hsiao, "A Novel Sustained Vector Tech. for the Detection of Hardware Trojans", VLSI Design'09, pp. 327-332.

[5] M. Banga and M. Hsiao, "VITAMIN: Voltage Inversion Tech. to Ascertain Malicious Insertions in ICs", HOST'09, PP. 104-107.

[6] Y. Jin and Y. Makris, "Hardware trojan Detection using Path Delay fingerprint", HOST'08, pp. 51-57.

[7] J. Li and J. Lach, "At-speed delay characterization for IC auth. and Trojan Horse detection", HOST'08, 99. 8-14.

[8] Hassan Salmani, Mohammad Tehranipoor, and Jim Plusquellic. "A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time", IEEE Transactions on (VLSI) Systems, Vol. 20, No. 1, January 2012.

[9] Mohammad Tehranipoor, Hassan Salmani, Xuehui Zhang, and Xiaoxiao Wang, "Hardware Trojan Detection Solutions and Design-for-Trust Challenges", Computer, July 2011, pp. 64-72.

[10] Rajat Subhra Chakraborty, Francis Wolff, Somnath Paul, Christos Papachristou, and Swarup Bhunia, "MERO:A Statistical Approach for Hardware Trojan Detection", CHES 2009, LNCS 5747, pp. 396-410.

[11] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraaborty, "Towards Trojan-free trusted ICs: Problem analysis and detection scheme", in Proc. Des. Autom. Test Eur. (DATE), 2008, pp. 1362-1365.

Lei Fang graduated from the Yan Tai University of China in 2007. He is currently working toward the Master degree in Research Institute of Electronic Science and Technology from the University of Electronic Science and Technology of China. His research interests include hardware security.

Lei Li received the M.Sc. And Ph.D. Degree in State Key Laboratory of Electronic Thin Films and Integrated Devices and National Key Laboratory of Science and Technology on Communications from the University of Electronic Science and Technology of China, in 2007 and 2010, respectively. He is currently an Associate Professor of Research Institute of Electronic Science and Technology at the University of Electronic Science and Technology of China. He has published over 20 journal articles and refereed conference papers.

Zhen Li graduated from the China University of Mining and Technology in 2006. He is currently working toward the Master degree in Research Institute of Electronic Science and Technology from the University of Electronic Science and Technology of China. He is currently continuing hardware Trojans research and has published several papers in this field.