# Hardware Trojan Detection Technique Using Frequency Characteristic Analysis of Path Delay in Application Specific Integrated Circuits

**Van Phuc Hoang, Thai Ha Tran, Ngoc Tuan Do, Hai Duong Nguyen**

***Abstract*— Since the last decade, hardware Trojan (HT) have become a serious problem for hardware security because of outsourcing trends in Integrated Circuit (IC) manufacturing. As the fabrication of IC is becoming very complex and costly, more and more chipmakers outsource their designs or parts of the fabrication process. This trend opens a loophole in hardware security, as an untrusted company could perform malicious modifications to the golden circuit at design or fabrication stages. Therefore, assessing risks and proposing solutions to detect HT are very important tasks. This paper presents a technique for detecting HT using frequency characteristic analysis of path delay. The results show that measuring with the frequency step of 0.016 MHz can detect a HT having the size of 0.2% of the original design.**

***Tóm tắt*— Từ thập niên 2010, Trojan phần cứng (HT) đã trở thành một vấn đề nghiêm trọng đối với bảo mật phần cứng, do xu hướng thuê sản xuất mạch tích hợp (Integrated Circuit - IC). Khi quá trình chế tạo IC trở nên phức tạp và tốn kém, ngày càng nhiều nhà sản xuất chip lựa chọn phương án thuê lại một phần hoặc toàn bộ thiết kế IC. Xu hướng này tạo ra lỗ hổng trong bảo mật phần cứng, vì một công ty không đáng tin cậy có thể thực hiện các sửa đổi độc hại vào trong mạch nguyên bản ở giai đoạn thiết kế hoặc chế tạo. Do đó, đánh giá rủi ro và đề xuất giải pháp phát hiện HT là một trong những nhiệm vụ hết sức quan trọng. Bài báo này trình bày một giải pháp phát hiện HT sử dụng phân tích đặc tính tần số của độ trễ đường truyền tín hiệu. Kết quả cho thấy, thực hiện khảo sát với bước tần số 0,016 MHz có thể phát hiện được HT có kích thước 0,2% so với thiết kế ban đầu.**

***Keywords*— Hardware Trojan; path delay, side-channel analysis, hardware security.**

***Từ khóa*— Trojan phần cứng, trễ đường truyền, phân tích kênh kề, bảo mật phần cứng.**

## I. INTRODUCTION

HT is a malicious module inserted in the Integrated Circuits during design or fabrication processes. An HT consists of two parts by common, namely Trigger and Payload. The Trigger is the condition that HT changes from the inactive state to the active state. Payload executes the HT’s function. Once inserted, HT can perform dangerous tasks such as Denial of Service, extract secret information or change behavior of the circuit... Detection and prevention are two main categories to protect embedded systems from the risk of HT [1, 2]. Prevention consists of modifying the original circuit during the conception phase to make a secure design, to assist another detection technique or to create a trusted production chain. On the other hand, detection includes techniques to determine whether or not HT is in the design. Classification of the existing HT detection techniques is shown in Fig.1:

This manuscript is received September 7, 2019. It is commented on October 18, 2019 and is accepted on October 21, 2019 by the first reviewer. It is commented on November 2, 2019 and is accepted on November 6, 2019 by the second reviewer.

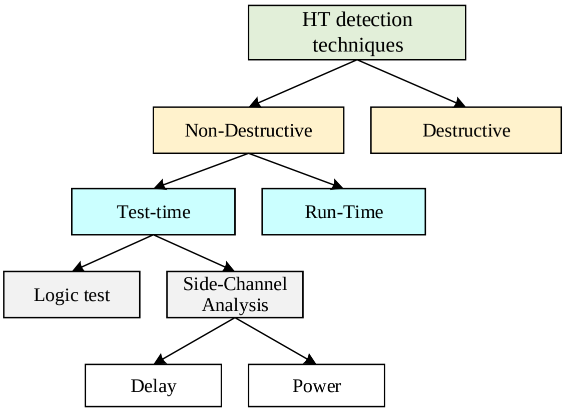


Fig.1. HT detection techniques

Side-channel analysis (SCA) is considered as one of the most effective technique to detect HT. In these methods, side-channel signals such as power, current, electromagnetic and delay are used for HT detecting. Typically, HT insertion results in the change of physical characteristic of circuits in some parameters. Hence, in SCA methods, these parameters can be used to detect HT by comparing with the golden circuit. The most common parameters used for detecting HT are power and delay. Also, in most of the techniques based on power, HT activation is necessary but it is not necessary when using delay [3]. Various delay-based detection methods are proposed as follows:

- A fingerprint is generated by measuring the delay and comparing it with a golden circuit fingerprint [4]. This method tries to generate the test vectors covering maximum outputs and uses them to measure the path delays. There is no hardware overhead, however, in complex circuits with a large number of inputs and outputs, measuring all path delays is difficult and takes a lot of time. Also, generating test vectors for all paths is complicated and it may not be able to cover all desired states.
- In the method using shadow register, some registers are placed beside the circuit registers with the same input as circuit registers and different clocks by different phases and use them to measure delay [5].
- Another method is proposed to use path delays to detect HT [6]. In this method, path delays in the *k* shortest paths are measured and compared to the corresponding path in the golden circuit. Detection probability in this method depends on two factors: the number of measured paths and delay measurement precision. The results show that measuring the delays on 20 paths with an accuracy of 0.01 ns can detect more than 80 % of Trojans. However, the main problem of this method is not flexible because it uses ISE reports (Timing Analyzer tool) to get delay paths [7]. Also, these reports only include information about paths from input to output signals.

These above mentioned methods focus on timing characteristics. In this paper, we propose a new approach to detect HT using frequency characteristic analysis of path delay. This method will evaluate the difference in distance between points in during signal propagation. Normally, the clock frequency of the system is chosen so as not to generate errors during the working process. However, when the clock is being adjusted in increasing direction, a critical value will be obtained at which the error occurs. Comparing this critical value with the original reference that was tested and stored in the database, if any difference is observed, HT will be detected.

The remainder of this research is organized as follows. Section II introduces a proposed design for HT detection based on path delay. The structure of the database is illustrated in section III. Evaluation of the proposed method is done in section IV. Then, section V concludes the research.

## II. PROPOSED DESIGN FOR HT DETECTION BASED ON PATH DELAY

### *A. Frequency characteristic of path delay*

Fig.2 presents the voltage waveforms that explain the differences in path delays leading to differences in the critical frequency at the survey points. At $t = T_0$, three points (*i*, *j*, *k*) have logic level of “0”. Then, the internal states will be changed according to the function of the design. Suppose that at $t = T_1$, these points have to be stable at logic level of “1”. Due to the path delays, however, we have only $k-$th point that satisfies the requirement because $T_k < T_1$. With $i-$th point, it has logic "0". In this proposed method, we aim to determine frequency corresponding to the point on rising edge with a half of amplitude ( $t = T_j$, as shown in the third waveform).

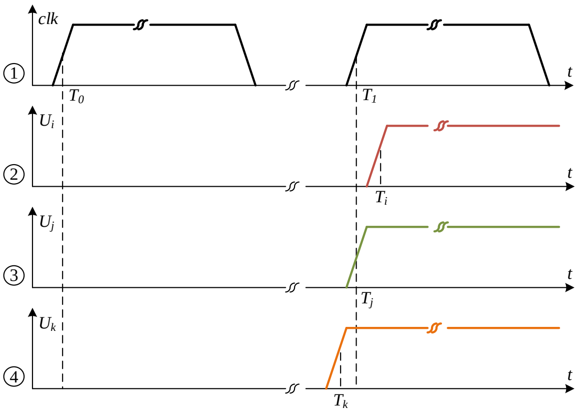


Fig.2. Frequency characteristic of path delay

### *B. Block diagram of the proposed HT detection method*

As one of the ChipScope Pro cores, ILA (*Integrated Logic Analyzer* component) can be used to monitor any internal signal of a design. The ILA core includes many advanced features of modern logic analyzers, including Boolean trigger equations, trigger sequences, and storage qualification. There is a problem when using ILA with a script because not all the Chipscope Analyzer GUI behavior can be done with Tcl script. ChipScope Engine Tcl Interface provides Tcl scripting access to JTAG download cables using the communication library in the ChipScoppe logic analyzer engine. The purpose of the CSE/Tcl interface is to provide a simple scripting system to access basic JTAG, FPGA, and VIO (virtual input/output) core functions. The Tcl script can perform detecting the cable, downloading the .bit file, submitting instructions through JTAG interface and VIO core function. But it cannot perform ILA function such as trigger condition setup, data capturing or exporting data [8]. The aim of this subsection is to design a new ILA called ILA_tiny with UART interface by VHDL language. ILA_tiny has simple features than the original ILA on Xilinx's ChipScope.

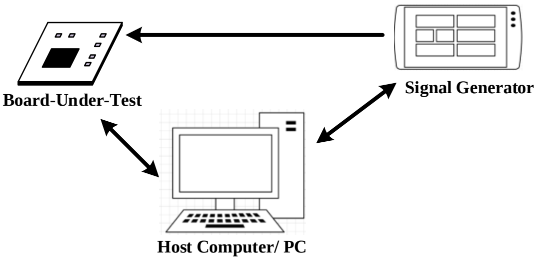


Fig.3. Connections between devices.

Fig.3 shows the connections between devices in this method. Here, the PC changes output signal of Signal Generator (*clk_ext* for design in Board-Under-Test) and receives the desired data from Board-Under-Test whose block diagram is illustrated in Fig.4 Signals in the FPGA design are connected to the inputs of ILA_tiny, and those signals can be captured at design speeds. Before the design is implemented, the parameters of the core are selected, including how many signals to capture and how many samples can be captured. Required input signals of ILA_tiny include.

- $\mathit{Conditions}$: $n-1$ bits;
- $\mathit{TriggerPorts}$: $n-1$ bits;
- *DataPorts*: $m-1$bits ( $m, n = 0, 1, \ldots, 127$ ).

The *TriggerPorts* input is compared against a set of expected values known as match units in *Conditions*. If the match equations evaluate to true, then a trigger event occurs and data is collected and stored into trace memory.

$$\mathit{TriggerPorts} = \mathit{Conditions} \qquad (1)$$

Because of the difference in clocks between UART_control (*clk_int* is constant frequency) and ILA_tiny (*clk_ext* is a changeable frequency, it is used in Main_Design), the signals which connect of these components have to extend. Data and control bytes from PC are sequentially transmitted in the individual bits by the *rx_in* signal. They will be processed in UART_TX before sent to UART_control. When being transmited, the desired data (*capture_data*) is divided into bytes, then pass through UART_TX and *tx_out* to PC. Note that the condition in Eq.(1) is only checked when the input signal from UART (*enable*) is high level. Also, *capture_done* will be a high level as an indicating signal to start sending in UART_control. At the end of the transmission, based on the *clear* signal, ILA_tiny is returned to the initialization state for the next cycle.

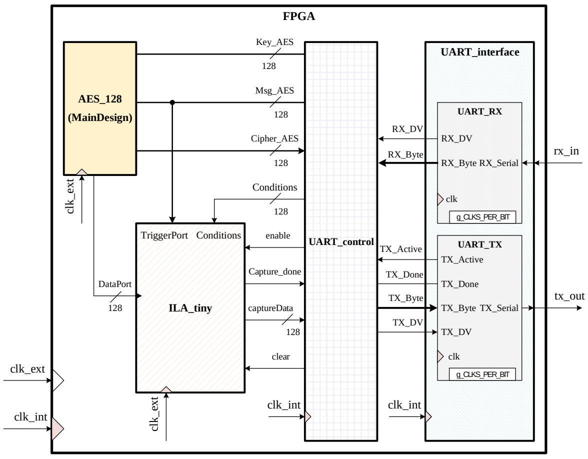


Fig.4. Block diagram of the proposed design

## *C. Algorithm of the proposed program*

Algorithm of the main program is illustrated in Fig.5, it is divided into three subprograms, where:

$m$: total number of bits (or points) to check, in this research $m = 128$;

$i$ : number of checked bits, default $i = 0$;

$j$: number of bits is being checked, default $j = 0$;

$f_0$: initial frequency;

$\Delta f_0$ : maximum of step frequency, default value: $\Delta f_0 = 4.096$ MHz;

$f$ : instantaneous frequency;

$\Delta f$ : instantaneous step frequency;

$\delta f$ : minimum of step frequency, default value $\delta f = 0.016$ MHz.

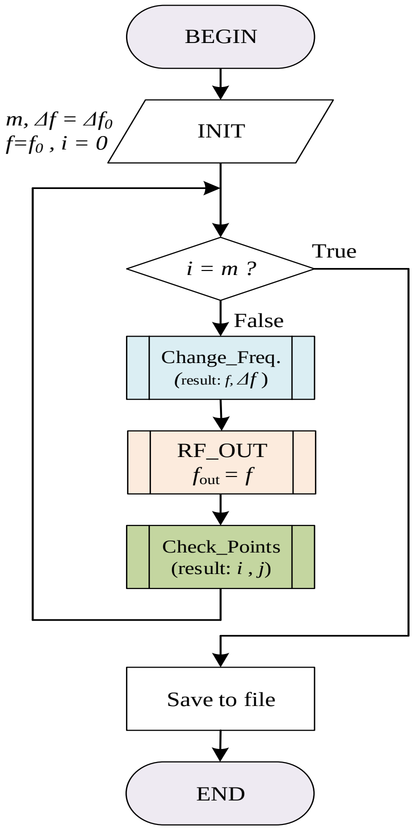


Fig.5. Algorithm of the proposed program

- *Change_Freq* is a subprogram to change the frequency of signal generator, determine the pair of values $(f, \Delta f)$. At the previous loop, assuming that the pair values of frequency and its step are $(f_{old}, \Delta f_{old})$. Choosing *Coarse_step* or *Fine_step* process will depend on $j$ - the number of bits is being checked. Then, $(f, \Delta f)$ is sent to the next subprogram called RF_OUT.
- In coarse_step process:

+ if $j = 0$: step frequency will get previous value:

$$\Delta f = \Delta f_{old} \tag{2}$$

+ if $j > 0$: the new step value will be less than the old value four times:

$$\Delta f = \frac{\Delta f_{old}}{4} \tag{3}$$

and

$$f = f_{old} + \Delta f \tag{4}$$

- *Fine_step* process: step frequency will be changed based on bisection method:

$$\Delta f = \frac{\Delta f_{old}}{2} \tag{5}$$

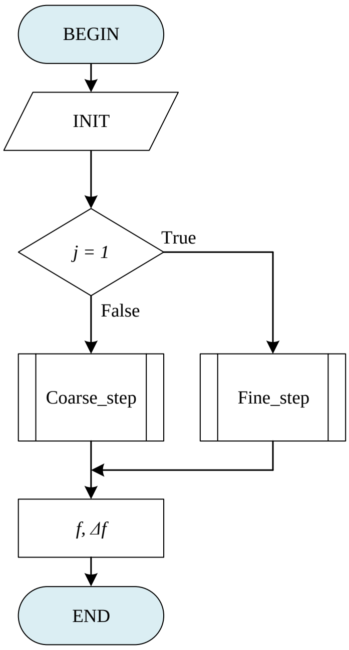


Fig.6. Flowchart of *Change_Freq* subprogram

- *RF_OUT*: this is a program to connect and control parameters on the signal generator. When the connection is successful, the required parameters from the PC will be sent, such as frequency, state, signal level, and so on.
- *Check_Points*: at each frequency, PC sends *capture_en* command to Board_Under_Test, then receives 128 bits of the desired data. This operation is repeated 20 times. Then, it compares each bit of *capture_data* with reference data that was tested and stored in the database, if there are more than 10 different values and the process in *Change_Freq* is *Fine_step*, the number of checked bits will increment. When *m* bits are checked, the measurement results are saved to the database that will be used for evaluation.

## III. STRUCTURE OF DATABASE

The block diagram of AES_128 is shown in Fig.7. This is a program that was written for Trojan benchmarks [9] and its architecture is the pipeline. The survey process will evaluate the difference in distance between points in one of the rounds. The selected round is random and can be changed. In this research, the first round is evaluated, so input and output signals are S0 and S1, respectively.

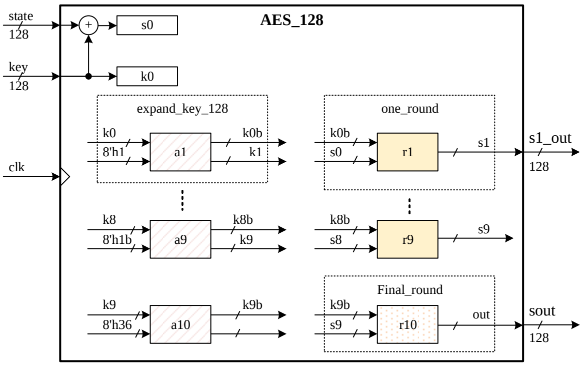


Fig.7. Block diagram of 128-bit AES core

*Msg* is selected as the pair of values *Msg_*0 and *Msg_*1 corresponding to the output of S1 contains all of bits 0 or all of bits 1 (Table 1). *Msg_*0 is used to set an initial value for registers and signals inside AES. For ILA_tiny, the *Conditions* input has a value equal *Msg_*1. Thus, when changing *Msg*, the condition in Eq.(1) is satisfied. After two periods of the clock, S1 will contain all of the bits to 1 which is the desired data *capture_data*. The selected inputs of AES as follows:

```
Key  = "00112233445566778899aabbccddeeff"
Msg_0= "5aa6044e28ec2d1596cae34557eac82c"
Msg_1= "f8a89d615fe23b9a3ca0223df0615106"
```

At each measurement, the corresponding critical values are saved. With a mathematical model, this result is represented in the form of a row vector, each element is the frequency corresponding to each bit of S1. To ensure the statistical properties, the survey process was carried out in *N* trials. Finally, the data set of measurement results is presented in the form of a matrix with a size of *N*×128.

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \\ \cdots \\ \mathbf{f}_{N-1} \end{bmatrix} = \begin{bmatrix} f_{0.0} & f_{0.1} & \cdots & f_{0.127} \\ f_{1.0} & f_{1.1} & \cdots & f_{1.127} \\ \cdots & \cdots & \cdots & \cdots \\ f_{\mathrm{N}-1.0} & f_{\mathrm{N}-1.1} & \cdots & f_{\mathrm{N}-1.127} \end{bmatrix} \tag{6}$$

where:

$\mathbf{f}_i$ : Row vector, its size is $1\times128$ resulted in *i*-th trial;

$f_{i.j}$: Element in row *i*, column *j*, it is presented critical frequency corresponding to *j-th* bit of S1 in the *i*-th trial.

From (6), the HT can be detected based on the pair of values $(\mu_j, \sigma_j)$ for each bit, where:

- Mean value:

$$\boldsymbol{\mu} = [\mu_0 \quad \mu_1 \quad \cdots \quad \mu_{127}] \tag{7}$$

$$\mu_j = \frac{1}{N}\sum_{i=0}^{N-1} f_{i.j} \tag{8}$$

- Variance:

$$\boldsymbol{\sigma}^2 = [\sigma_0^2 \quad \sigma_1^2 \quad \cdots \quad \sigma_{127}^2] \tag{9}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=0}^{N-1}\left(f_{i.j} - \mu_j\right)^2 \tag{10}$$

TABLE 1. VALUE OF EACH TRANSFORMATION IN ROUND 1

| State | Use Msg_0 | Use Msg_0 |
|---|---|---|
| **Msg**<br>(Initial state) | 5a 28 96 57<br>a6 ec ca ea<br>04 2d e3 c8<br>4e 15 45 2c | f8 5f 3c f0<br>a8 e2 a0 61<br>9d 3b 22 51<br>61 9a 3d 06 |
| **Key**<br>(Initial round key) | 00 44 88 cc<br>11 55 99 dd<br>22 66 aa ee<br>33 77 bb ff | 00 44 88 cc<br>11 55 99 dd<br>22 66 aa ee<br>33 77 bb ff |
| **S0**<br>(State at start of Round 1) | 5a 6c 1e 9b<br>b7 b9 53 37<br>26 4b 49 26<br>7d 62 fe d3 | f8 1b b4 3c<br>b9 b7 39 bc<br>bf 5d 88 bf<br>52 ed 86 f9 |
| After SubBytes | be 50 72 14<br>a9 56 ed 9a<br>f7 b3 3b f7<br>ff aa bb 66 | 41 af 8d eb<br>56 a9 12 65<br>08 4c c4 08<br>00 55 44 99 |
| After ShiftRows | be 50 72 14<br>56 ed 9a a9<br>3b f7 f7 b3<br>66 ff aa bb | 41 af 8d eb<br>a9 12 65 56<br>c4 08 08 4c<br>99 00 55 44 |
| After MixColumns | c0 84 0c c0<br>39 6c f5 28<br>34 52 f8 16<br>78 0f b4 4b | 3f 7b f3 3f<br>c6 93 0a d7<br>cb ad 07 e9<br>87 f0 4b b4 |
| AddRoundkey | c0 84 0c c0<br>39 6c f5 28<br>34 52 f8 16<br>78 0f b4 4b | c0 84 0c c0<br>39 6c f5 28<br>34 52 f8 16<br>78 0f b4 4b |
| **S1**<br>(State at start of Round 2) | 00 00 00 00<br>00 00 00 00<br>00 00 00 00<br>00 00 00 00 | ff ff ff ff<br>ff ff ff ff<br>ff ff ff ff<br>ff ff ff ff |

## IV. HT DETECTION RESULTS

In order to evaluate the impact of HT in FPGAs, we need to keep the same placement and routing between the golden and HT infected circuits. Hence, the only difference between them is the logic utilized for implementing the HT logic. Chip Planner in Altera Quartus II and Xilinx FPGA Editor in Xilinx ISE/Vivado Suites are two basic tools that can insert HTs without modifying the designed routing. There are four main steps to implement HT with Xilinx FPGA Editor tool [10]:

1) Perform Synthesize, Translate, Map, Place & Route steps for the original circuit.

2) Extract the Native Circuit Description (NCD) file which contains the logic, placement & routing information of the original circuit as the golden model.

3) Using the FPGA Editor to insert HT in unused LUTs and slices of FPGA with the NCD file, manually or by a script.

4) Generate bit files for both original and HT infected designs with FPGA Editor.

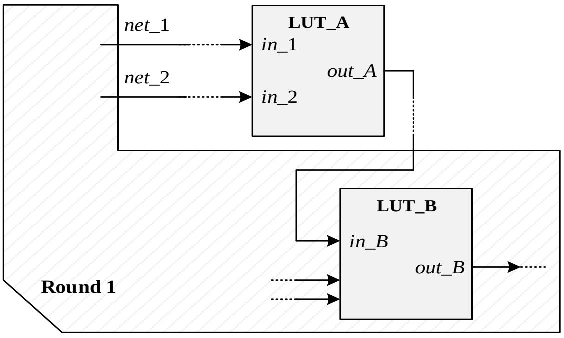


Fig.8. Algorithm of the proposed program

With this method, we can ensure that the placement and routing of the original circuit are the same in both golden and HT infected circuit. We explain how to add HT in the third step as follows:

*Create Trigger component of HT:*

- Randomly select an unused LUT, denoted by LUT_A;
- Select signals related to Round 1, assume that two selected signals are *net*_1 and *net*_2. These nets are routed to *in*_1 and *in*_2 of LUT_A;
- Change the function of LUT_A so that HT is not activated.

*Create Payload component of HT:*

- Randomly select a used LUT in Round 1, denoted by LUT_B. Note that LUT_B has at least a free pin.
- Connect *out_A* to *in_B*, then changing LUT_B's function.

In this work, two selected nets are S0[126] và S0[125]. There is only an OR gate in *LUT_A*. From Table 1, *in_B* is always "True" when MSG is either *Msg*_0 or *Msg*_1. LUT_B's function is given by:

$$out\_B = f(B). \qquad (11a)$$

When adding the *in_B* into LUT_B'pin, its function is modified so that the value of output is not changed. Here, an AND gate is used:

$$out\_B = f(B)\ AND\ in\_B. \qquad (11b)$$

TABLE 2. CRITICAL FREQUENCIES OF S1[0:1] (MHz)

| Trials | S1[0] | | S1[1] | |
|---|---|---|---|---|
| | Without HT | With HT | Without HT | With HT |
| 1 | 416.970 | 417.513 | 418.438 | 418.902 |
| 2 | 417.225 | 417.587 | 418.311 | 418.960 |
| 3 | 417.102 | 417.442 | 418.444 | 418.991 |
| 4 | 417.098 | 417.472 | 418.183 | 419.115 |
| 5 | 417.095 | 418.066 | 418.433 | 419.329 |
| 6 | 416.960 | 417.882 | 418.492 | 419.320 |
| 7 | 417.630 | 418.002 | 419.035 | 419.376 |
| 8 | 417.789 | 417.834 | 419.068 | 419.110 |
| 9 | 416.971 | 417.852 | 418.265 | 419.081 |
| 10 | 417.500 | 417.404 | 419.107 | 418.760 |
| $\mu_j$ | 417.234 | 417.705 | 418.577 | 419.094 |
| $\sigma_j$ | 0.282 | 0.234 | 0.334 | 0.189 |

TABLE 3. CRITICAL FREQUENCIES OF S1[126:127]

| Trials | S1[126] | | S1[127] | |
|---|---|---|---|---|
| | Without HT | With HT | Without HT | With HT |
| 1 | 356.569 | 357.119 | 358.808 | 359.357 |
| 2 | 356.319 | 357.097 | 358.619 | 359.365 |
| 3 | 357.156 | 357.100 | 359.267 | 359.433 |
| 4 | 356.513 | 357.150 | 358.813 | 359.390 |
| 5 | 356.514 | 357.482 | 358.813 | 359.717 |
| 6 | 356.568 | 357.409 | 358.742 | 359.582 |
| 7 | 357.409 | 357.381 | 359.615 | 359.760 |
| 8 | 357.281 | 357.378 | 359.487 | 359.645 |
| 9 | 357.005 | 357.474 | 359.162 | 359.618 |
| 10 | 356.622 | 357.059 | 358.972 | 359.248 |
| $\mu_j$ | 356.795 | 357.264 | 359.029 | 359.511 |
| $\sigma_j$ | 0.360 | 0.164 | 0.319 | 0.164 |

In this research, the Board_Under_Test is Sakura-G board and the signal generator is Rohde&Schwarz SMBV100A [11, 12]. In our implementation, the size of the genuine and infected circuit is 626 and 627 slices, respectively. This information is presented in Xilinx's reports or the number of slices in FPGA Editor. So, we have an infected circuit with HT of size 0.2% of the original one. Fig.9 is the normal distributions of the critical frequencies corresponding to the benchmark circuits S1[0], S1[1], S1[126] and S1[127].

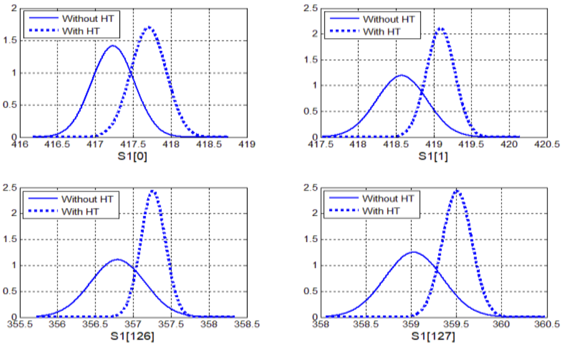
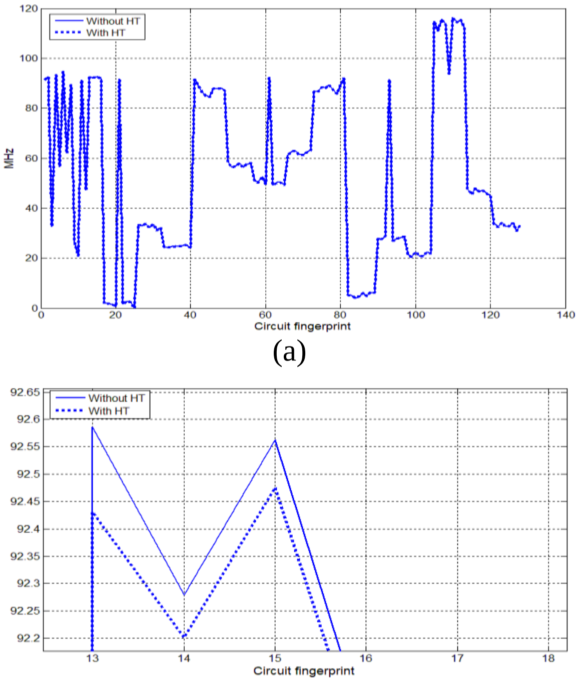


Fig.9. Distributions of the critical frequencies corresponding to path delays



(a)

(b)

Fig.10. Using frequency charateristic combined with fingerprint

In addition, combining fingerprints can be a solution to determine whether or not HT is in the design. Firstly, this method finds the smallest critical frequency. Then, fingerprint is a set of differences between the remaining points and this frequency. We can see that the fingerprints of the two circuits are nearly overlapping (Fig.10a), the difference is more evident with the segment in Fig.10b.

## IV. CONCLUSION

This paper presented the new technique to detect HT using frequency characteristic analysis of path delay. The preliminary hardware implementation results in the FPGA platform have clarified the feasibility of the proposed method. Similar to other SCA based detection methods, the experiment's conditions are constant or negligibly changed, such as temperature, the accuracy of frequency, and so on. In future work, we will improve the proposed method to achieve better results with more detail analysis.

## ACKNOWLEDGMENT

This work is funded by the research project under grant number HNQT/TKCG/04.20.

## REFERENCES

[1]. Swarup Bhunia, Mark M. Tehranipoor, "The Hardware Trojan War: Attacks, Myths, and Defenses," Springer, pp. 15-51, 2018.

[2]. Xuan Thuy Ngo, Van Phuc Hoang and Han Le Duc, "Hardware Trojan threat and its countermeasures," NAFOSTED Conference on Information and Computer Science, pp. 36-51, 2018.

[3]. Hao Xue, Saiyu Ren, "Hardware Trojan detection by timing measurement theory and implementation," Microelectronics Journal, vol. 77, pp. 16-25, 2018.

[4]. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," IEEE Int. Workshop Hardware-Oriented Security and Trust, 2008, pp. 51-57, IEEE, 2008.

[5]. L. Jie, J. Lach, "At-speed delay characterization for IC authentication and Trojan Horse detection," IEEE Int. Workshop Hardware-Oriented Security and Trust, 2008, pp. 8-14, IEEE, 2008.

[6]. A. Amelian and S.E. Borujeni, "A Side-Channel Analysis for Hardware Trojan detection based on Path Delay Measurement," Journal of Circuits, Systems, and Computers Vol. 27, No. 9, (2018).

[7]. Xilinx, "Timing Closure User guide," UG612 (v13.3) October 19, 2011.

[8]. Xilinx, LogiCORE IP ChipScope Pro Integrated Logic Analyzer (ILA) (v1.04a), DS299, June 2011.

[9]. Trojan Benchmarks, AES-T1500, https://www.trusthub.org/resource/benchmarks/AES/AES-T1500.zip.

[10].Xuan Thuy Ngo, Prevention and Detection of Hardware Trojan in Integrated Circuits, PhD Thesis, Telecom ParisTech, 2016.

[11]. Sakura-G specification ver 1.0, http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G_Spec_Ver1.0_English.pdf

[12].Rohde&Schwarz, R&S SMBV100A Vector Signal Generator Operating Manual, 2017.Bertoni, G., et al. Sponge functions. in ECRYPT hash workshop. 2007. Citeseer.

## ABOUT THE AUTHORS

**PhD. Associate Professor**

**Van Phuc Hoang**

Workplace: Deputy Head, Department of Microelectronics & Microprocessing, Le Quy Don Technical University.

Email: phuchv@lqdtu.edu.vn

The education process: Received B.S. degree and M.S. degree from Le Quy Don Technical University. Ph.D. degree in Electronic Engineering from The University of Electro-Communications, Tokyo, Japan in 2012.

Research today: Hardware security, Embedded system design for Internet of Things (IoT); Digital VLSI/ASIC design and FPGA-based system hardware design.

**MSc. Thai Ha Tran**

Workplace: Le Quy Don Technical University.

Email: hathaitran@lqdtu.edu.vn

The education process: received B.S. degree and M.Sc. degree from Faculty of Radio & Electronic Engineering, Le Quy Don Technical University.

Research today: Micro-electronics and hardware security; Digital Signal processing

**MSc. Ngoc Tuan Do**

Workplace: Le Quy Don Technical University.

Email: ngoctuansqtt@gmail.com

The education process: Received B.S. degree from

Telecommunications University and M.S. degree from Le QuyDon Technical University.

Research today: Hardware security and embedded system.

**PhD. Hai Duong Nguyen**

Workplace: Le Quy Don Technical University

Email:mta.haiduongnguyen@gmail.com

The education process: B.S. degree, M.S. degree from Le Quy Don Technical University, and Ph.D. degree from Bauman Moscow State Technical University, Russia.

Research today: Embedded system, hardware security and parallel system.