T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

HARDWARE TROJAN DETECTION USING DELAY RATIO METHOD ON FPGA

HASAN MUTLU

SUPERVISOR
DR. ALP ARSLAN BAYRAKÇİ

GEBZE
2023

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

HARDWARE TROJAN DETECTION USING DELAY RATIO METHOD ON FPGA

HASAN MUTLU

SUPERVISOR
DR. ALP ARSLAN BAYRAKÇİ

2023
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 16/03/2023 by the following jury.

JURY

Member
(Supervisor) : Dr. Alp Arslan BAYRAKÇİ

Member : Prof. Dr. Erkan ZERGEROĞLU

ABSTRACT

This project explores the detection of hardware trojans using delay-based methods on FPGAs. The study builds upon a previous project and addresses challenges while exploring various scenarios. It investigates the impact of trojan position on delay values, finding minimal effects with one-unit differences being insignificant and two-unit differences notable. The study also examines the influence of different trojan locations on the chip, observing varying in-chip variations across devices and noting the trojan's detectability in most cases. Additionally, the use of ring oscillators to detect trojan hardware among different chips is explored, showing significant variations across devices and confirming the method's effectiveness in identifying trojan hardware. Overall, the study highlights the potential of delay-based methods for trojan detection on FPGAs and suggests avenues for future research.

Keywords: path delay, trojan hardware, FPGA, in-chip variations, .

ÖZET

Bu proje, FPGAlarda gecikmeye dayalı yöntemler kullanarak donanım truva atlarının (trojan) tespitini araştırmaktadır. Çalışma, önceki bir projenin üzerine inşa edilmiş olup, orada karşılaşılan zorluklar ile çeşitli yeni senaryolara değinmektedir. Trojan'ın mantıksal devre üzerindeki pozisyonunun gecikme değerleri üzerindeki etkisini araştırmakta ve bir bir birimlik farkların önemsiz, iki birim farkların ise dikkate değer olduğunu belirlemektedir. Çalışma ayrıca çip üzerinde trojan'ın farklı konumlarının etkisini incelemekte, cihazlar arasında değişen iç çip varyasyonlarını gözlemlemekte ve trojanın çoğu durumda tespit edilebilirliğini vurgulamaktadır. Ayrıca, farklı çiplerde trojan donanımının tespiti için halka osilatörlerinin (ring oscillator) kullanılmakta, cihazlar arasında önemli varyasyonlar saptayarak yöntemin trojan donanımını tanımlama etkinliği doğrulanmaktadır. Genel olarak, bu çalışma, FPGAlarda trojan tespiti için gecikmeye dayalı yöntemlerin potansiyelini vurgulamakta ve gelecekteki araştırmalar için yeni konular sunmaktadır.

Anahtar Kelimeler: FPGA, donanımsal truva atı, mantıksal devre, gecikme değeri.

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
LTH	:	Low-to-high delay
RO	:	Ring oscillator

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

A hardware trojan is a malicious modification or addition to the hardware design or manufacturing process of a computer system or electronic device. It is a form of hardware-based attack where an attacker seeks to compromise the integrity, security, or functionality of the targeted system.

Hardware trojans can be designed to introduce delays or unintentionally cause delays while performing different harmful actions. One way to detect hardware trojans is by looking for these unusual delays in the operation of a device.

The primary aim of this project is detecting hardware trojans using a series of delay based methods on FPGA. (Figure 1.1) The measurement method is based on comparing the delays produced by a logic path with a trojan and a logic path without it. This project is continuation of a previous project[1]. The measurement methods used in the previous project were continued, besides, it aimed to bring solutions to some problems encountered in the previous project as well as experimenting different situations.

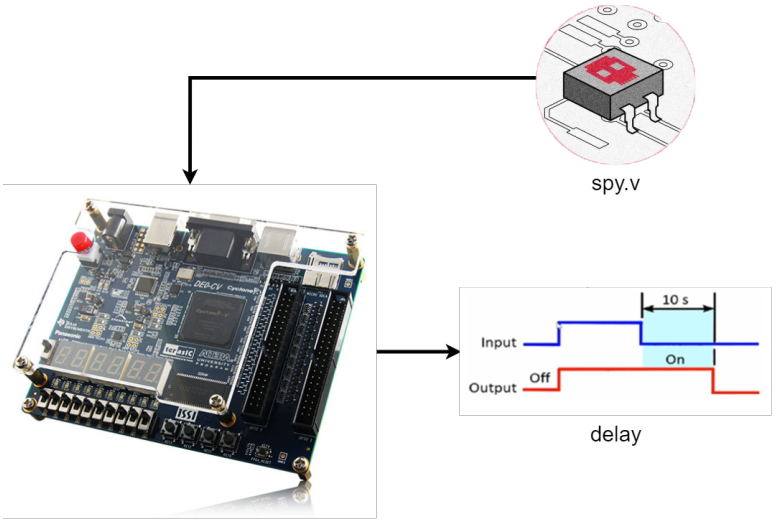

Figure 1.1: An abstraction of delay on FPGA.

FPGA stands for Field-Programmable Gate Array. It is a type of digital integrated circuit that can be configured by the user to implement complex logic circuits, by defining and interconnecting a variety of different logic functions.

FPGAs can be programmed using a hardware description language (HDL) such as VHDL or Verilog, which is used to describe the logic and interconnections of the circuit. The HDL code is then compiled and used to configure the FPGA to perform

the desired function. In this project, he Verilog HDL is used in the implementation.

The cases tested in the project include the effect of in-chip and inter-chip variations of the FPGA on delay values and the detection of hardware trojan despite these variations. In-chip variations, also called as internal variations, are variations that occur within the circuit itself whereas inter-chip variations are variations that occur between the same model of different electronic circuits. These variations typically occur due to manufacturing processes, component tolerances, or environmental factors.

Another case tested in the project is the effect of placing the trojan on different positions on the sample path.

The main component used in these measurements is ring oscillator. It is a device that generates a continuous waveform output. It consists of odd number of inverter gates. It is used as a main reference while comparing clean and infected delay values.

1.1. Path-Delay Definition

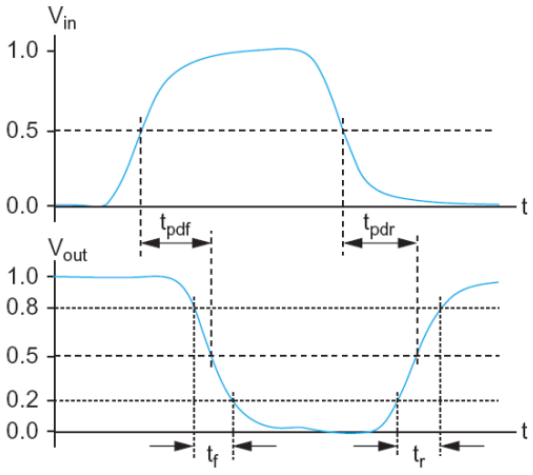

Figure 1.2: Propagation Delay[2]

Path delay, also known as propagation delay, is a measure of the time it takes for a signal to travel through a specific path in a circuit. In other words, it is the time it takes for a signal to propagate from the input of a circuit to the output of that same circuit. The path delay is determined by the number of gates and components in the path, as well as their individual delays. The critical path in a circuit is the path with the longest delay. Identifying the critical path can help to determine the speed and performance of a logic circuit.

Trojan hardwares can cause additional delay by slowing down the system's performance, disrupting communication between components, or causing other forms of

malfunction. Additionally, the existence of a trojan hardware may take time to perform its harmful operations, causing further delay.

Propagation delay, or path delay is the amount of time it takes for a signal to travel through a circuit or device, such as a gate or an amplifier. It is typically measured in nanoseconds (ns) or picoseconds (ps). In digital circuits, it can affect the overall performance of the system and can limit the maximum clock speed at which the circuit can operate. Figure 1.2 gives an example of propagation delay.

1.2. Ring Oscillator Definition

A ring oscillator is a type of electronic oscillator circuit that generates a continuous waveform output without the need for an external input signal. It consists of an odd number of inverting stages (typically composed of digital inverters or analog amplifiers) connected in a closed-loop configuration, forming a ring-like structure. (Figure 1.3)

The frequency of oscillation in a ring oscillator is determined by the delay introduced by each stage and the propagation delay of the inverter or amplifier used. The more stages in the ring, the longer the total delay, and hence the lower the frequency of oscillation. Conversely, reducing the delay in each stage or increasing the number of stages leads to a higher oscillation frequency.

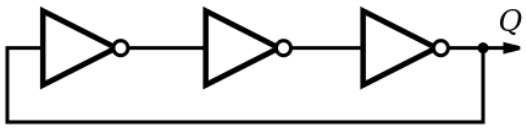

Figure 1.3: Ring Oscillator[3]

In this project, the implemented ring oscillators' periods are measured the same way the other regular paths' delays are measured. Instead of connecting its output back to it as input, its input and output are connected to the measured state machine components. The ring oscillators' period values are used as a universal reference for each location inside a device or between different devices.

1.3. Previous Work

As stated earlier, this project is a continuation of a previous project. Therefore, the works done in the previous project and the methods used are important for understanding this project as well. The most basic delay value measurement method used in the previous project has been continued in this project as well.

In the previous project, the delay values are measured on one sample path. Critical path of this sample path extracted, and a trojan path is produced by introducing one additional gate in the middle of this path.

The delay values on these paths are measured using a simple state machine. Firstly, an input is provided to the path. After the path produced the provided input, the input is complemented and the measurement process starts. Number of clock pulses are counted until the path finally produced the new input as output. The final value of the counter is displayed on the 7-segment display of the device.

The delay of one single path is very low. It is not very differentiable with the 250 MHz PLL clock of the FPGA. In order to make the difference distinguishable, one sample path is chained multiple times, up to 100 times. These methods mentioned so far are also used in this continuation project.

1.3.1. Main Results of the Previous Work

The method is proved to be effective. As it can be seen on Table 1.1, the path with a trojan produced higher delay values and the difference is captured on the same chip. The difference between a clean path and an infected path become more distinguishable as paths are chained more.

Table 1.1: sinlgepath_3 measurement results of the previous project.

No. Chained Paths	Clean Path Delay	Trojan Path Delay
5	10	10
10	19	21
20	37	41
50	95	104

1.3.2. In-Chip Variations Results

The previous project aimed to capture the effect of in-chip variations on the delay. In order to test this case, the same path is duplicated and placed on the FPGA at the same time. However, the Quartus Fitter program mixed the paths across the entire FPGA and this prevented us from measuring different delay values at specific locations. Therefore, a definite conclusion could not be reached. (Figure 1.4)

Due to this situation, it is aimed to investigate new methods that will distribute the paths in a more localized way within the FPGA and to determine the effect of in-chip variations with these methods.

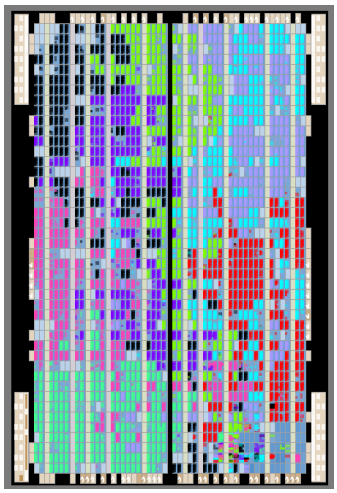

Figure 1.4: 8 duplicate singlepath_3_100 paths on Chip Planner. Every color represents a different path.

1.3.3. Hardware Trojan Detection Among Different Devices

Shortly after the first project, there was the opportunity to make the same measurements on another device of the same model. In this way, the effect of variations between chips on delay values could be tested. As can be seen in Figure 1.5, in the results found, it was seen that the clean path on one device produced a higher delay value than the path with trojan on the other device in all occasions. This revealed that the trojan could be hidden in measurements made between different chips.

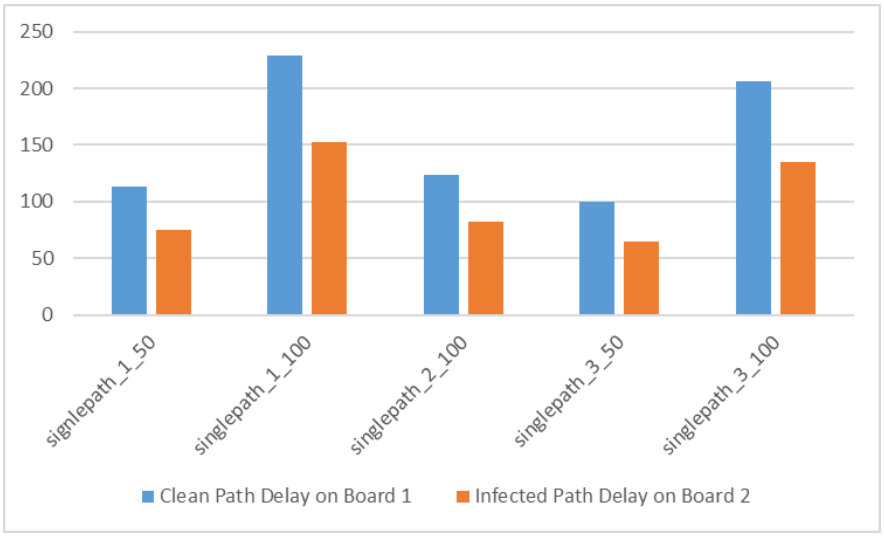

Figure 1.5: Comparison between delay values of several clean and infected paths on different boards.

This situation created a new research topic for this project and new methods aimed to detect this hidden trojan hardware.

2. METHOD

2.1. Preperations

2.1.1. Disabling Delay Optimization

Quartus Prime Lite [4] software is used in implementation of both projects. It provides various optimization options like area optimization, timing optimization, power optimization etc. However, these optimizations minimizes the logic path and therefore minimize the delay.

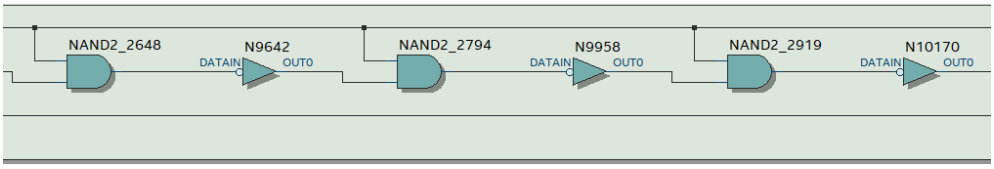

Figure 2.1: A small section of singlepath_3 in RTL Viewer

In order to differentiate between a clean path and a path with trojan, it should cause delay with all of its components. Therefore, the path optimization should be disabled.

One way to make the path cause delay with all of its components is to use one Verilog HDL Synthesis Attribute, *keep* keyword. This keyword is used for the wire components that connect the logic elements to each other. It doesn't prevent the Quartus to logically optimize the path. However, it introduces a buffer on every single wire in between them. Therefore, these buffers cause delay. (Figure 2.1)

2.1.2. Output Display Method

There are several ways to display numerical outputs on an FPGA. One method to display the numerical outputs is to use the 7-segment display on the FPGA device.

After measuring the delay value, a series of operations is performed with this value. The value is divided into decimals and displayed on the 7-segment. (Figure 2.2)

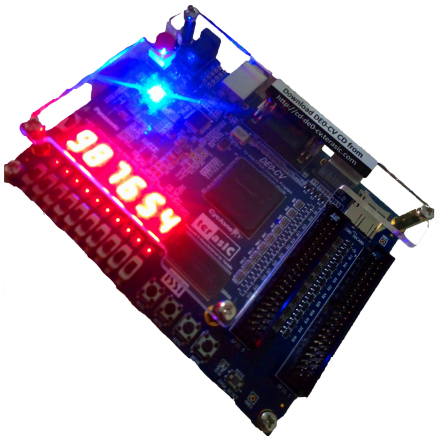

Figure 2.2: DE0-CV device displaying a numerical value.

2.2. Delay Measuring Method

One economical way of measuring the delay value is using dedicated logic designs that can be embedded within the FPGA design. This design can be implemented by designing circuit specific state machines and counting the clock pulses in the given delay duration. Such solution is used in this project.

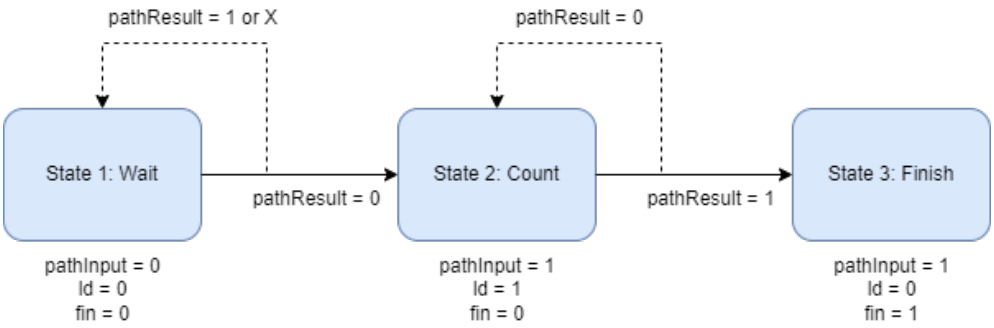

Figure 2.3: Low-to-High State Machine

In order to measure delay, a simple state machine is used. As it can be seen on Figure 2.3, the state machine have only three states. It starts as soon as the FPGA starts working. In the first step, an input is sent to the path. The path has an undefined output at the start. In this first state, it is waited to produce the output that is sent into it.

As soon as it starts producing the initial output, it goes to the second state. In this state, the complement of the first input is sent to the path and it is waited to produce the new output. During this process, a counter in the is incremented in every clock pulse.

When the path finally produces the new output, the counting stops and the

measurement is finished. The delay value is found and the result is displayed on the 7-segment display.

2.2.1. Chaining Paths to Increase Delay Difference

The delay caused by one path is very minimal. It is hard to detect the difference in the delays of two single paths wtih relatively slow clock frequencies of FPGA devices.

In order to make the delay values more distinguishable, the paths are chained multiple times. By this way, the delay is increased by a factor and even a little more than that. The more the path is chained, the more distinguishable delay values a clean path and a path with trojan would produce.

In order to chain these paths, multiple paths are created and connected from start to finish. The path input is sent into the first path and the result is obtained from the last path. Such structures are created by implementing a Python program and is used as follows:

```
$ chainPaths.py <pathName> <isNot> <chainAmount> <isRO>
```

where pathName is path to be chained, isNot is whether it produces a complement of input or not, chainAmount is

2.2.2. Interpreting The Output Value

After all measurements, we finally got some numerical outputs. As mentioned earlier, the counter that measures the amount of time that passed is fed by a signal of 250 MHz. In other words, it is incremented in every 4 nanoseconds. By the nature of the state machines, the changes on the output can only be detected in rising clock edges.

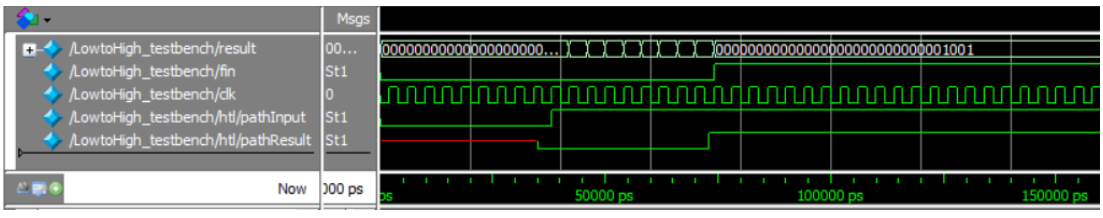

Figure 2.4: An example testbench result.[5]

In the Figure 2.4, there is an example testbench result of a low-to-high delay measurement. It gives an exact representation of how the delay measurement is performed inside the FPGA. All three states of the state machine can easily be understandable.

The path has a 35 ns of delay, and the 250MHz clock has a period of 4 ns. The state machine detects this change on the next rising clock edge. The result is found as 9. This means that the delay is between 32 and 36 nanoseconds, which is accurate. A more precise measurement would need a faster clock.

The previous project showed very little difference in high-to-low and low-to-high delay values. Therefore, only low-to-high delay values are measured in this project. Furthermore, in the results section, direct output values of measurements are used instead of their nanoseconds values.

2.3. Placing Trojan on Different Positions on Path

One of the aims of this project is to capture the effect of placing trojan on different positions on the path. Figure 2.5 demonstrates how the trojan can be placed on different positions. In order to achieve this, a Python program is implemented and used. An example usage of the program is as follows:

```
$ trojanPlacer.py <pathName> <isNot>
$ trojanPlacer.py singlepath_1 y
There are 37 gates in this path. Which position would you like
to put the trojan on?
25
Trojan path is created successfully as singlepath_1_spy_p25n.v
```

It takes a path name as an argument and checks the Verilog file of the specified path. It counts the number of gates in the file and asks user to which position would the trojan be placed on. After the position is specified by the user, it creates a new Verilog file with a trojan hardware.

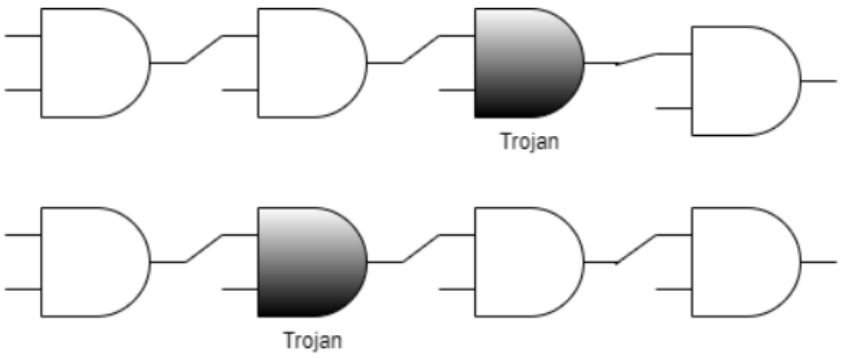

Figure 2.5: A demonstration of trojan on different positions on a path.

The added trojan hardware basically consists of one additional XOR gate and a NAND gate that is out of critical path. This same trojan is used in every creation and usage of infected paths. The results from the previous project showed that even such a minimal additional hardware could be detected with this method.

2.4. Placing Trojan on Different Locations on Chip

As stated earlier in the Section 1.3.2, adefinite conclusion could not be reached in the previous project regarding the effect of in-chip variations.

New research has been carried out in order to place the paths in a more localized way within the device and thus to reach a more precise result. Quartus's Fitter settings were tested with different values. However, no option to automatically place the design on a specific part of the circuit is found.

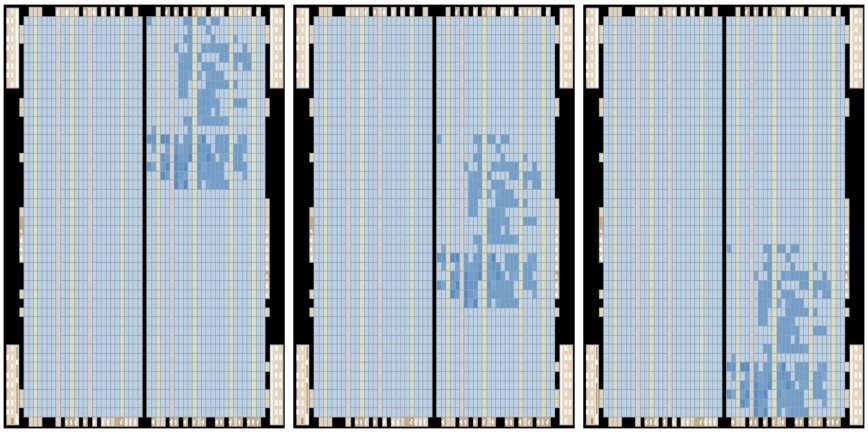

Figure 2.6: A path placed on three different locations on chip.

It has been discovered that the design can be moved to different points in the device by manually drag-and-drop method in the Chip Planner interface of Quartus. However, this feature does not work properly, perhaps because it is not very preferred by users. When the entire design is dragged to a different point at once, the program does not detect it. Pieces of the design have to be moved by human labor, several at a time. This is a time consuming process. Also, the Chip Planner doesn't allow carrying some components to some other specific places. This prevents the user from carrying the entire design without changing its initial form.

Despite all these difficulties, this method was used in this part of the project. Due to the difficulties of the process, only a small version of a single path could be measured.

Since the schematic of the device is not horizontally symmetrical, moving to dif-

ferent points was done only on the vertical axis. Depending on the size of the measured path, the device was divided into three virtual locations. Clean path, infected path and equivalent ring oscillator were moved to these three locations and the measurements of the delay values in these different locations were taken. These three different locations are shown in Figure 2.6.

2.5. Using Ring Oscillator to Detect Trojan Hardware Among Different Chips

The cases described so far were tested to check whether the trojan outputs a lower delay value than the clean path under different conditions and thus hides. As stated earlier in the Section 1.3.3, it was determined that such a situation was possible in the measurements taken between different devices.

It was decided to use ring oscillator as a reference in such situations where the trojan is hidden under different circumstances. In this method, instead of directly comparing the delay values of two measurements, an equivalent ring oscillator is used as a universal reference. An equivalent ring oscillator consists of the same or close number of inverters to the sample path.

The ring oscillator is put under the same two circumstances of the two measurements to be compared. Measurements were taken from a clean and an infected path placed under different conditions. Measurements were also taken from the ring oscillator under the same conditions. The ratios of both measurement results to the ring oscillator measurement results under the same conditions were compared.

$$\text{Clean Ratio} = \frac{\text{Clean Path on Board 1}}{\text{RO on Board 1}}$$

$$\text{Infected Ratio} = \frac{\text{Trojan Path on Board 2}}{\text{RO on Board 2}}$$

For example, while comparing a measurement from different boards, the ring oscillator delay on these two boards are also measured. Then, the results of the two measurements were compared by calculating their ratio to the results of the ring oscillator on the same board. In summary, the ratios can be calculated with the formulas above.

3. RESULTS

There are three different paths there are measured throughout the tests. Regular paths come from ISCAS-85 c7552 (32bit adder/comparator) benchmark.[6] These paths are extracted using a tool named PLODE[7]. They are the extracted critical paths of some bigger circuitry. With all the gates and connections inside, they basically just get an input and either direct it to the output or invert it.

The path names follow a naming convention. singlepath_1, singlepath_2 and singlepath_3 are the names of three example paths. When these path names are followed by another number, as in singlepath_1_100, it means that the path is chained 100 times.

3.1. Trojan on Different Positions on Path Results

While measuring the effect of trojan's position on the path, the trojan is placed on the several positions on the example paths. These positions start from first possible position, several positions on the middle and the last possible place.

The path files contain a number of side gates that doesn't have effect on output. However, the trojan placer program also identifies these positions and enables user to place the trojan there. Results of trojan positions where it isn't located on the critical path are also included in the result tables.

The results are shown in one table below, for each path. The tables specify the position of the trojan, measured delay value and whether the trojan is on the critical path or not; Y stands for yes and N stands for no. The delay value of a clean path is also specified on the top.

Table 3.1: singlepath_1_100 Different Trojan Position Results

path name: singlepath_1_100		
Clean Delay:		229
Position	Delay	On Critical Path
1	236	Y
5	230	N
15	236	Y
25	238	Y
36	237	Y

Table 3.2: singlepath_2_100 Different Trojan Position Results

path name: singlepath_2_100		
Clean Delay:		124
Position	Delay	On Critical Path
1	127	Y
5	124	N
15	125	N
25	129	Y
31	128	Y

Table 3.3: singlepath_3_100 Different Trojan Position Results

path name: singlepath_3_100		
Clean Delay:		206
Position	Delay	On Critical Path
1	206	N
5	211	Y
15	211	Y
25	212	Y
35	206	N
45	206	N
53	210	Y

These tested are conducted several times. The delay values are measured at most one unit differently in each measurements. The final measurement results are as in the tables above.

The results show that the position of the trojan on the path has a very small effect on the delay value. One unit of difference is not too much but two units of difference is remarkable. In all three paths, the trojan path produced one or two units higher delay value on position 25. But these small changes aren't so worrying.

Another thing discovered with these results is that even when the trojan is not in the critical path, it sometimes produces 1 unit higher delay value. Maybe this is due to the load brought by the extra component.

3.2. Trojan on Different Location on Chip Results

As explained earlier in the Section 2.4, 50 times chained version of singlepath_1 is used in testing the effect of placing trojan on different locations on the chip. The measurements are conducted using three boards.

There are several columns on the result tables below. The columns represent location of the design, delays of clean path, trojan path and ring oscillator, ratio of clean delay to ring oscillator delay and ratio of trojan delay toring oscillator delay. Mean values of these ratios are calculated below. The delay value of the default positions on the path, where Quartus places them, is also included on top.

Table 3.4: singlepath_1_50 Different Location on Chip Results on Board 1

Board 1					
path name: singlepath_1_50					
				Default Location Delay:	113
Location	Clean Delay	Trojan Delay	RO Delay	Clean Ratio	Infected Ratio
1	113	116	106	1,066037736	1,094339623
2	113	116	106	1,066037736	1,094339623
3	113	115	106	1,066037736	1,08490566
			Mean:	1,066037736	1,091194969

Table 3.5: singlepath_1_50 Different Location on Chip Results on Board 2

Board 2					
path name: singlepath_1_50					
				Default Location Delay:	70
Location	Clean Delay	Trojan Delay	RO Delay	Clean Ratio	Infected Ratio
1	71	72	70	1,014285714	1,028571429
2	70	72	67	1,029411765	1,058823529
3	71	72	69	1,028985507	1,043478261
			Mean:	1,024227662	1,043624406

Table 3.6: singlepath_1_50 Different Location on Chip Results on Board 3

Board 3					
path name: singlepath_1_50					
				Default Location Delay:	82
Location	Clean Delay	Trojan Delay	RO Delay	Clean Ratio	Infected Ratio
1	82	84	79	1,037974684	1,069620253
2	83	86	79	1,050314465	1,081761006
3	84	85	80	1,05	1,06875
			Mean:	1,046096383	1,073377086

Due to the difficulties of placing the path on different locations, the tests are conducted with 50 times chained paths. It is relatively small, therefore it is not the easiest to distinguish differences.

It is observed that the in-chip variations vary from device to device. In first board, almost no change is observed, whereas on second board, there are more diverse results, but the trojan is still distinguishable. On the third board, however, the variation is so high that the clean path on location 3 produced the same delay with infected path on location 1. Despite that, still, there isn't a pattern observed like best location is this and worst location is that.

Inside one chip, the lowest infected ratio is found to be higher than the higest clean ratio in first and the third boards. However, on the second board, infected ratio on the location 1 is found to be smaller than the clean ratio on the location 3, meaning that the trojan was able to hide when placed on a different location.

This might be due to the path used in the measurements being not the biggest path we have. If same experiments could be conducted with bigger paths, preferably on a bigger FPGA, the method might be proved to be effective. In this case, however, it is failed to detect the trojan on different locations on the second board.

3.3. Using Ring Oscillator to Detect Trojan Hardware Among Different Chips Results

As it mentioned earlier in the section 1.3, variations among different devices was significant. A device with a clean path can cause a delay greater than another device with trojan hardware in it. This enables the trojan to be hidden in certain devices.

In order to detect the trojan hardware among multiple chips, measurements are made on two boards. Couple of versions of paths, their clean and trojan delays and

equivalent ring oscillator delays are measured. Clean and infected ratios are calculated using the measurement results. These results are listed in the two below tables.

Table 3.7: Measurement Results of Several Paths on Board 1

Board 1					
Path Name	Clean Delay	Trojan Delay	RO Delay	Clean Ratio	Infected Ratio
singlepath_1_50	112	114	106	1,056603774	1,075472
singlepath_1_100	229	237	214	1,070093	1,107477
singlepath_2_100	124	129	102	1,215686	1,264706
singlepath_3_50	100	104	102	0,980392	1,019608
singlepath_3_100	206	212	203	1,014778	1,044335

Table 3.8: Measurement Results of Several Paths on Board 2

Board 2					
Path Name	Clean Delay	Trojan Delay	RO Delay	Clean Ratio	Infected Ratio
singlepath_1_50	70	72	67	1,044776	1,074627
singlepath_1_100	143	148	135	1,059259	1,096296
singlepath_2_100	78	84	65	1,2	1,292308
singlepath_3_50	65	67	66	0,984848	1,015152
singlepath_3_100	132	136	131	1,007634	1,038168

Next two tables give a comparison of ratios on both occasions where one device consists of a clean path and the other consists a trojan.

Table 3.9: Comparison of a Clean Board 1 and Infected Board 2

Clean Board 1, Infected Board 2			
Path Name	Board 1 Clean Ratio	Board 2 Infected Ratio	Difference
singlepath_1_50	1,056604	1,074627	0,018023
singlepath_1_100	1,070093	1,096296	0,026203
singlepath_2_100	1,215686	1,292308	0,076621
singlepath_3_50	0,980392	1,015152	0,034759
singlepath_3_100	1,014778	1,038168	0,02339

Table 3.10: Comparison of a Clean Board 2 and Infected Board 1

Clean Board 2, Infected Board 1			
Path Name	Board 2 Clean Ratio	Board 1 Infected Ratio	Difference
singlepath_1_50	1,044776	1,075472	0,030696
singlepath_1_100	1,059259	1,107477	0,048217
singlepath_2_100	1,2	1,264706	0,064706
singlepath_3_50	0,984848	1,019608	0,034759
singlepath_3_100	1,007634	1,044335	0,036701

In both occasions, where one board is clean and the other board is infected, ratio of infected board is always found to be greater than the ratio of clean board. This proves the effectiveness of the method. When comparing the two boards, it allowed us to decide which one was suspected to contain a hardware trojan.

By this method, when comparing two devices, one known device with a clean path and another one with an unknown path, it is guaranteed that, if the unknown device consists of a trojan hardware that causes delay, it will produce a greater ratio than the clean device.

This method was proved to be effective unlike the unclear results on the second part, where the trojan is located on different locations, because the default locations of all the paths are used in this part. Quartus places all the designs on similar positions positions, around the third location on the Figure 2.6. This ensures that the delay values are independent of in-chip variations. When both in-chip and inter-chip variations play a role, however, it became harder to detect the trojan and the method doesn't prove certain effectiveness. Further research can be conducted using better tools and technologies to overcome the difficulty of placing the paths on desired location on FPGA.

4. CONCLUSION

In conclusion, this project aimed to detect hardware trojans using delay-based methods on FPGA. The project built upon a previous study and aimed to address some of the challenges encountered while exploring different scenarios. The effects of in-chip and inter-chip variations on delay values and the detection of hardware trojans despite these variations were investigated. The main measurement component used was a ring oscillator, which generated a continuous waveform output and served as a reference for comparing clean and infected delay values.

The first set of test results focused on the position of the trojan on the path. The findings showed that the trojan's position had a minimal impact on the delay value, with a difference of one unit being insignificant. However, a difference of two units was considered noteworthy. Interestingly, even when the trojan was not located on the critical path, it occasionally resulted in a delay increase of one unit. This could be attributed to the additional load introduced by the extra component.

The second set of test results examined the effect of placing the trojan in different locations on the chip. The study used a 50-times chained version of a single path and tested three boards. The results revealed varying in-chip variations across different devices. While some boards showed minimal changes, others exhibited more diverse results. Nevertheless, the trojan remained detectable. On one particular board, the variation was significant enough that the clean path at one location produced the same delay as the infected path at another location. No specific pattern emerged indicating the best or worst location for hiding the trojan. However, on the second board, the trojan managed to remain undetected when placed in a different location, suggesting that the method's effectiveness could depend on the size of the path and the FPGA used.

The third set of test results focused on using the ring oscillator to detect trojan hardware among different chips. Significant variations were observed among different devices, where a clean path on one device could exhibit a greater delay than a device with trojan hardware. This allowed the trojan to remain hidden in certain devices. To address this, measurements were conducted on two boards with multiple versions of paths, including their clean and trojan delays, as well as equivalent ring oscillator delays. The results consistently showed that the ratio of the infected board was always greater than that of the clean board, confirming the method's effectiveness in detecting trojan hardware. When comparing two devices, one known to have a clean path and another with an unknown path, the method reliably identified the device suspected to contain a hardware trojan based on the greater delay ratio produced by the unknown device.

In summary, this project successfully explored various aspects of detecting hardware trojans using delay-based methods on FPGA. The results demonstrated the method's potential in identifying trojans, even with variations in trojan position, chip location, and different devices. Further research is recommended to validate the method's effectiveness on larger paths and larger FPGAs, which may provide more robust detection capabilities.

BIBLIOGRAPHY

[1] Hasan Mutlu, "Hardware Trojan Detection Using Delay Based Method on FPGA," 2023. [Online]. Available: https://github.com/hasanmutlu26/Hardware-Trojan-Detection-Using-Delay-Based-Method-on-FPGA.

[2] Neil Weste, David Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 2010.

[3] Wikipedia, "Ring oscillator," [Online]. Available: https://en.wikipedia.org/wiki/Ring_oscillator.

[4] Intel Quartus Prime Software, [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html.

[5] Modelsim HDL Simulator, [Online]. Available: https://eda.sw.siemens.com/en-US/ic/modelsim/.

[6] University of Michigan, "ISCAS High-Level Models," [Online]. Available: https://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html.

[7] Gokce Nur Erer, Alp Arslan Bayrakci, "PLODE: Precise Logic and Delay Simulator for Structural Verilog," 2021.