T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

HARDWARE TROJAN DETECTION USING DELAY-BASED METHOD ON FPGA

HASAN MUTLU

SUPERVISOR
DR. ALP ARSLAN BAYRAKÇİ

GEBZE
2022

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

HARDWARE TROJAN DETECTION USING DELAY-BASED METHOD ON FPGA

HASAN MUTLU

SUPERVISOR
DR. ALP ARSLAN BAYRAKÇİ

2022
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 31/08/2021 by the following jury.

JURY

Member
(Supervisor) : Dr. Alp Arslan BAYRAKÇİ

Member : Prof. Dr. Erkan ZERGEROĞLU

ABSTRACT

This project aimed to detect hardware Trojans in FPGA designs by measuring delays on a single example path and its trojan-added versions. The method proved to be effective by showing a significant difference in delay between the clean path and the trojan path. Additionally, the project aimed to investigate the impact of in-chip variations on delay measurements by placing multiple copies of the same design on different parts of the chip. The results showed variations in delay, but the cause of these variations is uncertain and further investigation is needed. The project concludes with recommendations for future studies to improve the method and to test it on different FPGA boards.

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

ÖZET

Bu proje, tek bir örnek mantıksal devre yolu ve onun casus donanım eklenmiş versiyonlarındaki gecikmeleri ölçerek FPGA tasarımlarındaki donanımsal truva atlarını tespit etmeyi amaçlamıştır. Yöntem, temiz yol ile casus donanım eklenmiş versiyonları arasındaki gecikmede önemli bir fark tespit ederek etkili olduğunu kanıtlamıştır. Ek olarak proje, aynı tasarımın birden fazla kopyasını çipin farklı bölümlerine yerleştirerek, çip içi varyasyonların gecikme değerleri üzerindeki etkisini araştırmayı amaçlamıştır. Sonuçlar gecikmede farklılıklar tespit etmiştir. Ancak bu varyasyonların kesin nedeni belirsizdir ve daha fazla araştırmaya ihtiyaç vardır. Proje, yöntemi geliştirmek ve farklı FPGA cihazlarında test etmek için gelecekteki çalışmalar için önerilerle sona ermektedir.

Anahtar Kelimeler: mantıksal devre yolu, gecikme, casus donanım, truva atı donanım, FPGA, çip içi varyasyonlar.

ACKNOWLEDGEMENT

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
LTH	:	Low to High
HTL	:	High to Low
STDEV	:	Standard Deviation

CONTENTS

Abstract iv

Özet v

Acknowledgement vi

List of Symbols and Abbreviations vii

Contents viii

List of Figures ix

List of Tables x

1 Introduction 1

1.1 Related Work 2

1.1.1 Path-Delay Definition 2

1.1.2 Path Delay Measurement on FPGA 3

2 Method 5

2.1 Preperations 5

2.1.1 Disabling Delay Optimization 5

2.1.2 Output Display Method 6

2.1.3 Generating and Accelerating Clock 6

2.2 Delay Measuring Method 7

2.3 Chaining Duplicated Paths to Increase Delay Difference 8

2.4 Filling the FPGA With Separate Duplicate Paths 9

2.5 Interpreting The Output Value 11

3 Results 12

3.1 Single Path Measurement Results 12

3.2 Seperate Duplicate Paths Measurement Results 14

4 Conclusion 26

Bibliography 27

LIST OF FIGURES

1.1 An abstraction of delay on FPGA. 1
1.2 Propagation Delay[1] 2
1.3 Digital Oscilloscope 3
1.4 Logic Analyzer 3

2.1 A small section of singlepath_3 in RTL Viewer 5
2.2 DE0-CV device displaying a numerical value. 6
2.3 Low-to-High State Machine Table 7
2.4 Low-to-High State Machine 7
2.5 A LUT matrix.[5] 9
2.6 Inside of lookup table, there are multiplexers.[5] 9
2.7 8 duplicate singlepath_3_100 paths on Chip Planner. Every color represents a different path. 10
2.8 An example testbench result.[6] 11

3.1 30-32 times duplicated sinlepath_3 and its trojan added versions on the chip planner. 21
3.2 Different duplicate paths on Chip Planner 25

LIST OF TABLES

3.1 Paths and definitions. 12
3.2 sinlgepath_1 measurement results. 13
3.3 sinlgepath_2 measurement results. 13
3.4 sinlgepath_3 measurement results. 13
3.5 spypath_3_1 measurement results. 13
3.6 spypath_3_2 measurement results. 14
3.7 not3_33 measurement results. 14
3.8 8 copies of sinlgepath_1_50 measurement results. 15
3.9 8 copies of sinlgepath_1_100 measurement results. 16
3.10 8 copies of sinlgepath_2_50 measurement results. 16
3.11 8 copies of sinlgepath_2_100 measurement results. 17
3.12 8 copies of sinlgepath_3_50 measurement results. 17
3.13 8 copies of sinlgepath_3_100 measurement results. 18
3.14 8 copies of spypath_3_1_50 measurement results. 18
3.15 8 copies of spypath_3_1_100 measurement results. 19
3.16 8 copies of spypath_3_2_50 measurement results. 19
3.17 8 copies of spypath_3_2_100 measurement results. 20
3.18 8 copies of not3_33_50 measurement results. 20
3.19 8 copies of not3_33_100 measurement results. 21
3.20 32 copies of singlepath_3_20 measurement results. 22
3.21 30 copies of spypath_3_1_20 measurement results. 23
3.22 30 copies of spypath_3_2_20 measurement results. 24

1. INTRODUCTION

A hardware Trojan is a malicious design or modification in a hardware device, such as a computer chip, that can be used to compromise the security of a 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 first aim of this project is detecting how the delay values change when we add a trojan hardware on top of a Verilog design to be placed on an FPGA. (Figure 1.1)

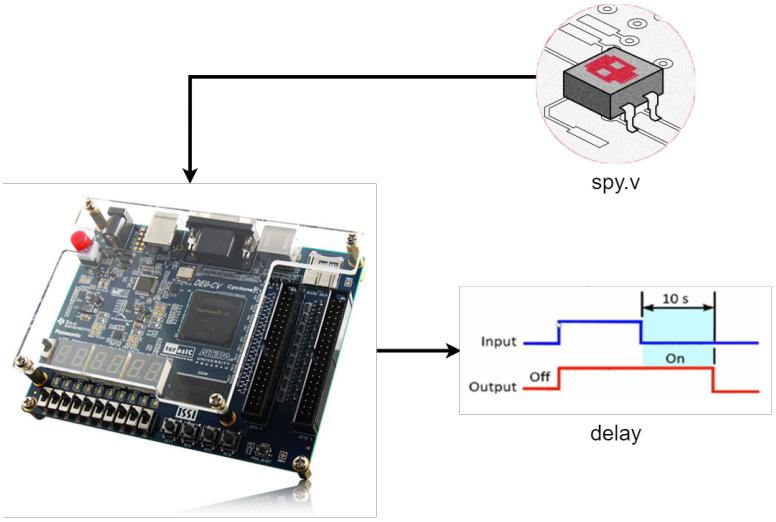

Figure 1.1: An abstraction of delay on FPGA.

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, such as AND, OR, and XOR gates.

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. The Verilog HDL is used in the implementation of this project.

Another key goal of this project is, if the trojan hardware can be placed on different parts on the layout, detecting how much the delay values change due to the internal variations of the chip.

Internal variations in a chip refer to the natural variations in the physical properties of the transistors and other components that make up a chip. These variations can occur

during the manufacturing process and can also cause delay to vary even on the identical logic elements, when they are on different parts of the layout.

1.1. Related Work

1.1.1. Path-Delay Definition

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 and optimizing the critical path can help to improve the speed and performance of a logic circuit.

Trojan hardware 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.

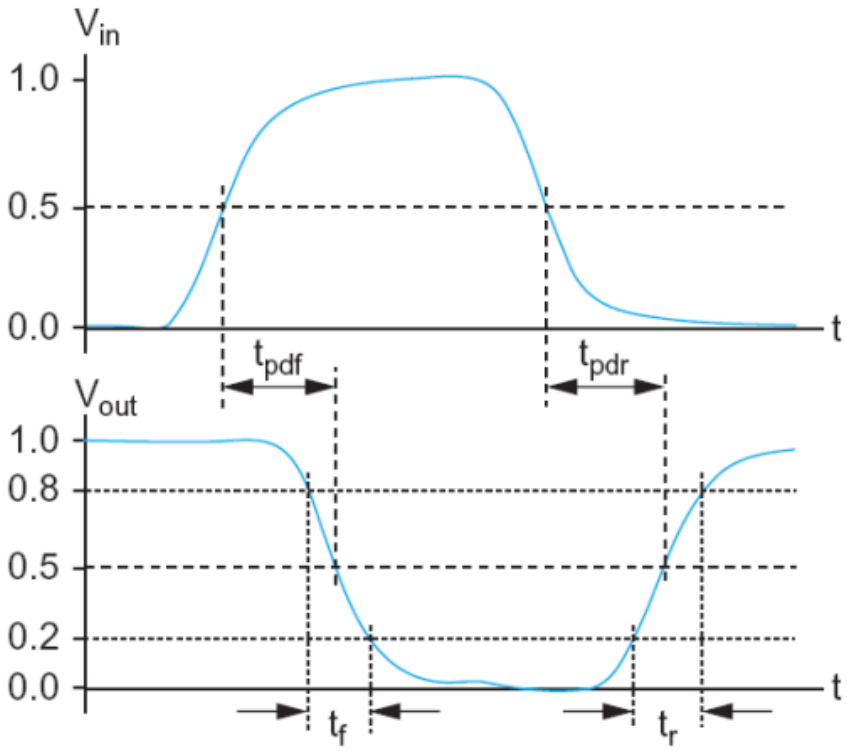

Figure 1.2: Propagation Delay[1]

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.

The path delay can be calculated by hand using the following formula:

Path delay = sum of gate delays + interconnect delays

where the gate delay is the delay caused by the logic gates and the interconnect delay is the delay caused by the wiring connecting the gates.

1.1.2. Path Delay Measurement on FPGA

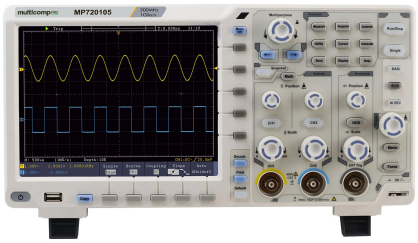

Figure 1.3: Digital Oscilloscope

There are several ways to measure delay in a circuit, some of the most common methods include:

Oscilloscope: An oscilloscope is an instrument that can measure the voltage and time relationship of a signal. By measuring the time between a specific event on the input signal and the corresponding event on the output signal, you can determine the delay. (figure 1.3)

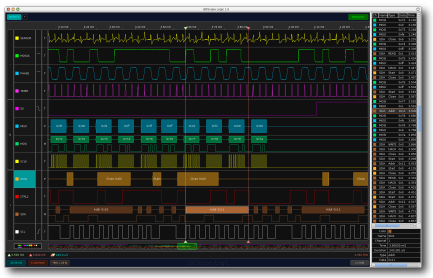

Figure 1.4: Logic Analyzer

Logic Analyzer: A logic analyzer is a specialized test instrument that can

measure the timing of digital signals. By measuring the delay between different digital events, you can determine the delay in a digital circuit. (Figute 1.4)

Signal Generator: A signal generator can be used to generate a test signal, which can then be fed into the circuit. By measuring the delay between the input and output signals, you can determine the delay of the circuit.

Simulation: The delay can also be measured using simulation software. By simulating the circuit using a tool such as SPICE, you can measure the delay between different points in the circuit by analyzing the voltage and current waveforms.

Testers: There are specialized testers for measuring delays in specific types of circuits, such as SerDes(Serializer-Deserializer), DDR(Double data rate) interfaces, and PLL(Phase-Locked Loop) testers.

Measuring delay on an FPGA (Field-Programmable Gate Array) can be done in a similar way as measuring delay in a circuit. However, there are some additional options or considerations when measuring delay on an FPGA.

The delay can also be measured by using dedicated logic designs that can be embedded within the FPGA design. This design can be implemented by designing circuit-specific state machines and making use of the clock pulses in the given delay duration. Such a method is used throughout this project.

2. METHOD

2.1. Preperations

2.1.1. Disabling Delay Optimization

Quartus[2] Prime Lite software is used to implement the project. It provides various optimization options like area optimization, timing optimization, power optimization etc. These optimizations are used in default to improve the performance of the design and reduce its resource usage. However, these optimizations minimizes the logic path and therefore minimize the delay.

In order to differentiate between a clean path and a path with trojan, the path 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.

" keep: A Verilog HDL synthesis attribute that directs Analysis and Synthesis to not minimize or remove a particular net when optimizing combinational logic.[3] "

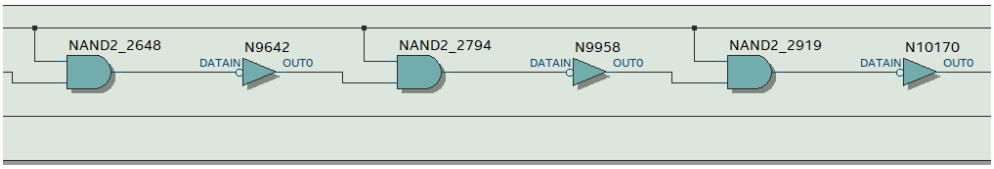

Figure 2.1: A small section of singlepath_3 in RTL Viewer

This *keep* 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.

In the Figure 2.1 above, the NAND gates doesn't occupy a space in the chip. However, the buffers on the wires do. Therefore, the delay is caused by these buffers.

We are looking forward to maximize the delay so that the delay caused by the additional trojan circuitry become more obvious.

2.1.2. Output Display Method

There are several ways to display numerical outputs on an FPGA. The first initial thought solution was to provide UART communication with the computer and display the results on the serial terminal. However, there were some issues with finding the output ports on the specific device worked on so that and alternative solution is implemented.

Another method to display the numerical outputs is to use the 7-segment display on the FPGA device. However, it is not as easy like ”Send the number to 7-segment and it displays it.”. Every segment must be logically adjusted in order to light up when necessary.

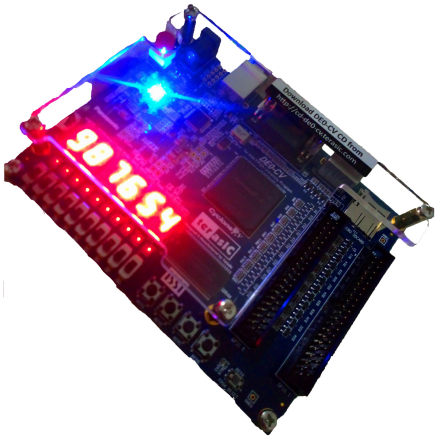

Figure 2.2: DE0-CV device displaying a numerical value.

In order to drive the 7-segment display with the correct logic, divider circuits are used. A simple, straightforward divider circuit is implemented. This divider basically works by subtracting divisor from dividend and counting the number of subtractions.

Six of these dividers are used for the DE0-CV device, which has 6 digits on 7-segment display. All dividends are divided by 10. The result of each divider are fed into the next divider and the remainders are fed to the 7-segment display. With this method, every digit of the output are separately obtained and displayed on the 7-segments as shown in Figure 2.2.

2.1.3. Generating and Accelerating Clock

The DE0-CV device comes with a clock frequency of 50 MHz. Also, it supports a PLL that can go up to 250 MHz. A Phase-Locked Loop (PLL)[4] is a type of feedback control system that is used to generate a stable and precise frequency signal. A PLL

consists of a phase detector, a voltage-controlled oscillator (VCO), and a low-pass filter. The input to the PLL is a reference signal, and the output is a signal that is locked to the phase of the reference signal.

In order to get more precise results, the 50 MHz clock is used as reference signal to generate a 250 MHz PLL signal.

2.2. Delay Measuring Method

As it is mentioned in the introduction, there are several ways to measure the delay in the FPGA. One economical way 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.

In order to measure delay, two simple state machines are used. One is for measuring low-to-high delay and the other is for measuring high-to-low delay. Their states are identical, only the fed and expected path input-outputs differ.

Low to High State Machine								
	Control Inputs			Next State		Control Outputs		
State	R[1]	R[0]	pathResult	N[1]	N[0]	pathInput	ld (counter++)	fin
Wait	0	0	0	0	1	0	0	0
	0	0	1	0	0	0	0	0
Count	0	1	0	0	1	1	1	0
	0	1	1	1	0	1	1	0
Finish	1	0	0	1	0	1	0	1
	1	0	1	1	0	1	0	1

Figure 2.3: Low-to-High State Machine Table

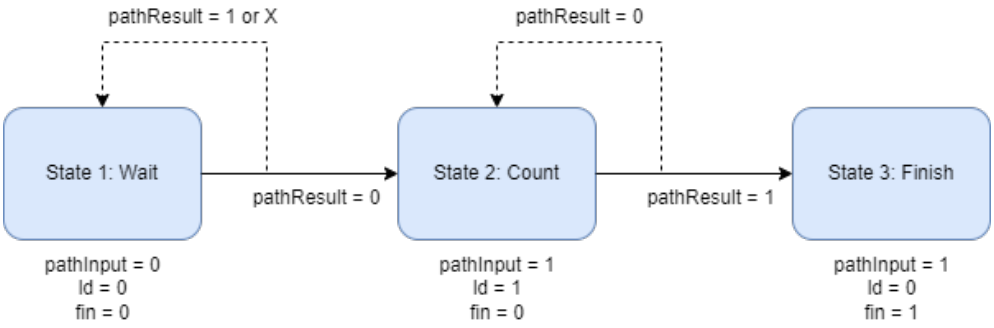

Figure 2.4: Low-to-High State Machine

As it can be seen on Figure 2.3 and Figure 2.4, the state machines have only three states. It starts as soon as the FPGA starts working. In the first step, an input, either 1 or 0, 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 datapath is incremented in every clock pulse.

When the path finally produces the new output, the counting stops and the measurement is finished. The result is displayed on the 7-segment display.

2.3. Chaining Duplicated 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 components, especially when you have a relatively slow clock frequency.

In order to make the delay differences look more obvious, the paths are needed to be duplicated multiple times. By this way, the delay is increased by a factor. The more the path is duplicated, the more divergent delay values a clean path and a path with trojan would produce.

In order to duplicate 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 structure is created by the following procedure:

```
1 f duplicate_paths(pathName, reps, Vcc, Gnd):
2   print(f"{pathName} path0(w0, pathInput, {Vcc}, {Gnd});")
3
4   for i in range (1, reps - 1):
5       print(f"{pathName} path{i}(w{i}, w{i-1}, {Vcc}, {Gnd});")
6
7   print(f"{pathName} path{reps - 1}(pathResult, w{reps - 1}, {Vcc},
8   {Gnd});")
```

In this project, the paths are duplicated by 5, 10, 20, 50 and 100 times.

2.4. Filling the FPGA With Separate Duplicate Paths

This method is related to the secondary aim of this project, which is detecting how the delay values change when we place the design on different parts of the FPGA layout.

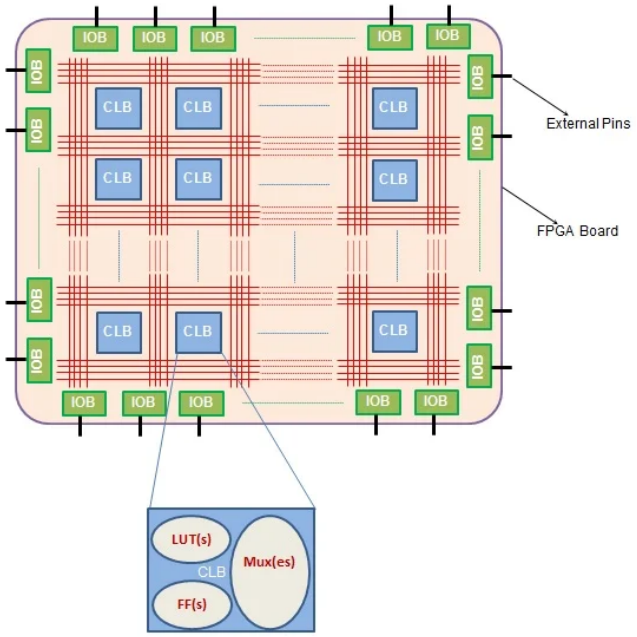

Figure 2.5: A LUT matrix.[5]

A lookup table (LUT) on an FPGA is a small logic block that can be used implement a wide variety of digital functions. As shown in Figure 2.5, FPGA chips are build up from a matrix of LUTs to perform combinational logic operations. LUTs on FPGAs typically consist of multiplexers inside them. (Figure 2.6) They are a key building block in the architecture of modern FPGAs.

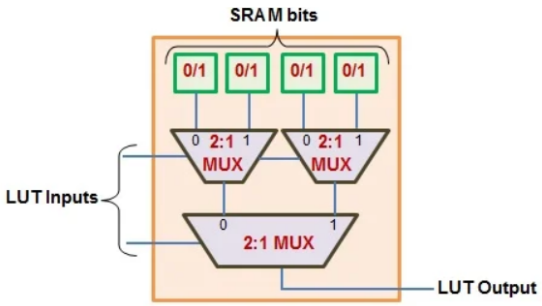

Figure 2.6: Inside of lookup table, there are multiplexers.[5]

It is possible to place the design on different parts of the layout by directly using the Chip Planner in Quartus. In the ECO Editing mode, the design can be assigned to different LUT blocks on the chip by drag and drop operation. But it isn't as easy as it sounds. Not all blocks can be dragged to every other block, and moving multiple blocks at once is very tricky and time-consuming.

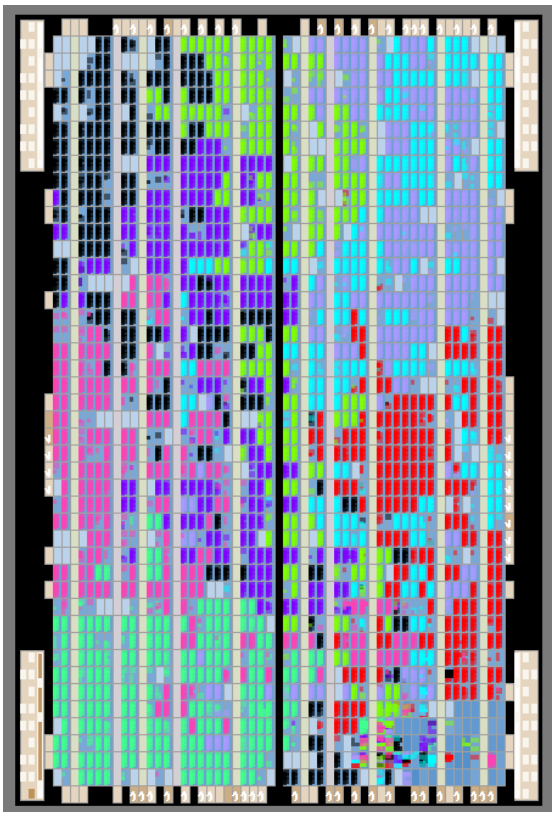

Figure 2.7: 8 duplicate singlepath_3_100 paths on Chip Planner. Every color represents a different path.

More elegant solution to this problem is to fill the FPGA with duplicate separate paths and get measurement results from them one by one. By this way, the paths are placed on different parts of the layout automatically by the Quartus. In this project, the paths are duplicated 8 times, in other words, placed on 8 different parts on the layout. While duplicating the paths, the state machines and counters assigned to each path should also be duplicated. Figure 2.7 shows how these designs are placed on the chip. More than that, 20 times chained versions of paths are also copied 30 to 32 times.

With the intention of getting the measurement results from each of these duplicated paths, a multiplexer-like structure is used with the physical switches on the FPGA. Before programming the FPGA, the first three switches are set to determine which path's measurement results would be displayed. Inside the FPGA, the delays of all 8 paths are measured but only the selected one's result is displayed.

2.5. Interpreting The Output Value

After all measurements, we finally got some numerical outputs. But what do these values mean?

As mentioned earlier, the counter that measures the amount of time that has passed until the path gives the correct output 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.

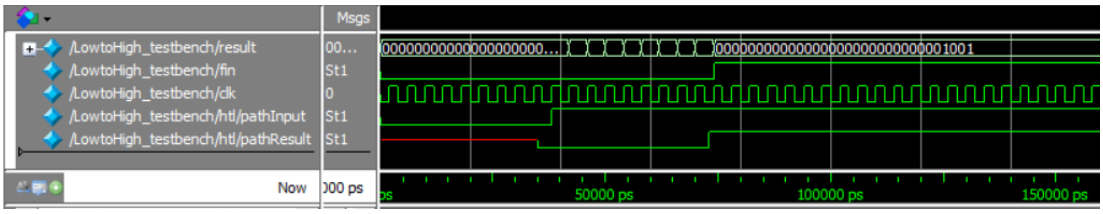

Figure 2.8: An example testbench result.[6]

In the Figure 2.8, there is an example testbench result. 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.

In this testbench example, the path has a 35 ns of delay, and the 250MHz clock has a period of 4 ns. However, as mentioned earlier, the state machine can detect the changes only on rising clock edges. In this example, the result is changed between two rising clock edges. The state machine detects this change on the next rising clock edge and increments the counter for one last time. 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.

For the sake of simplicity, the ceiling value is used throughout this report.

3. RESULTS

There are six different paths that are measured throughout the tests. Regular paths come from ISCAS-85 c7552 (32bit adder/comparator) benchmark.[7] These paths are extracted using a tool named PLODE[8]. You can think of them as 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. Here are some of the paths and their definitions:

Table 3.1: Paths and definitions.

sinlgepath_1	Regular path.
singlepath_2	Regular path.
singlepath_3	Regular path.
spypath_3_1	Trojan added version of singlepath_3
spypath_3_2	Trojan added version of singlepath_3
not3_33	Delay equivalent of singlepath_3. It consists of equal number of (33) not gates to the number of buffers in the singlepath_3. It is expected to give equal or similar delays to the singlepath_3.

In very rare occasions, the outputs are found to be unusually high. These outputs got back to their average window in the next measurements. Such measurement results aren't recorded as they were very rare unusual cases.

3.1. Single Path Measurement Results

In the following tables in this section, the measurement results of chained duplicate paths are given. In this part, the delays of clean paths and paths with additional trojan circuitry can be compared. The first two paths, sinlgepath_1 and sinlgepath_2 don't have any trojan added versions of them but they are added to the report in case there can be future work using these paths.

The most important tables to analyze are tables from 3.4 to 3.6. These tables include the the results of clean sinlgepath_3 and its trojan added versions.

There are one table for each path. In each table, there are five columns. *No.* column specifies how many times one single path is duplicated in a chaining manner. *LTH* and *HTL* stand for low-to-high and high-to-low. These columns show the displayed output on the FPGA device. *LTH Delay* and *HTL Delay* columns show the delay values

in nanoseconds. Since the clock is 250 MHz, which has a period of 4 nanoseconds, this value is four times of the output on the device.

Table 3.2: sinlgepath_1 measurement results.

No.	LTH	LTH Delay (ns)	HTL	HTL Delay (ns)
5	11	44	11	44
10	21	84	21	84
20	42	168	42	168
50	109	436	109	436

Table 3.3: sinlgepath_2 measurement results.

No.	LTH	LTH Delay (ns)	HTL	HTL Delay (ns)
5	6	24	6	24
10	12	48	12	48
20	22	88	23	92
50	58	232	58	232
100	118	472	119	476

Table 3.4: sinlgepath_3 measurement results.

No.	LTH	LTH Delay (ns)	HTL	HTL Delay (ns)
5	10	40	10	40
10	19	76	19	76
20	37	148	38	152
50	95	380	98	392

Table 3.5: spypath_3_1 measurement results.

No.	LTH	LTH Delay (ns)	HTL	HTL Delay (ns)
5	10	40	11	44
10	21	84	21	84
20	41	164	41	164
50	104	416	105	420

Table 3.6: spypath_3_2 measurement results.

No.	LTH	LTH Delay (ns)	HTL	HTL Delay (ns)
5	11	44	11	44
10	21	84	21	84
20	41	164	41	164
50	105	420	105	420

Table 3.7: not3_33 measurement results.

No.	LTH	LTH Delay (ns)	HTL	HTL Delay (ns)
5	10	40	10	40
10	19	76	19	76
20	38	152	38	152
50	98	392	97	388

When the delay differences between sinlgepath_3 and its trojanadded versions are analyzed, it is observed that there are significant differences between the delays. It is not as obvious when smaller number of duplicates are compared. However, as expected, as we duplicate the path even more, the delay difference is multiplied by a factor and become more obvious on the results. It shows that when we have a path that we are sure of its purity, we can use it as a reference to determine if there is any additional delay in the suspected paths with this method.

Furthermore, when the sinlgepath_3 is generated by the Quartus, it includes 33 buffers on its critical path. In the FPGA, all logics are performed by LUT units. Therefore, whatever the logic gate is, we expect every component that occupies one LUT in the FPGA to produce equal delays as they use the same circuitry inside. In order to test this situation, an equivalent circuit to sinlgepath_3, the not3_33 is used. It includes 33 not gates. When their measurement results are compared, they are not exactly equal but very similar. This minimal difference may be caused by different place and routing operations the Quartus has done on them while programming the device.

3.2. Seperate Duplicate Paths Measurement Results

In this section, the delay results of 8 separately duplicated paths that are placed on the FPGA at the same time are displayed. As mentioned before, using high number

of chained paths multiplies the delay difference by a factor. Therefore, 50 and 100 times chained versions of each path are used in this part. Only the low-to-high delays are measured.

In all calculations using higher number of duplicated paths, the output result could change by one unit in consecutive measurements. Therefore, each of them are measured at least three times and the noted output is determined by best of the three manner. In addition, this also ensured that the design is span across the entire FPGA layout, making it being placed on all different parts as possible.

There are three columns in each table. There are 8 chained paths placed on the FPGA. The first column, *ID* specifies which of these 8 paths result that is. In the *LTH* column, the output on the FPGA device is shown. Finally, in the *LTH Delay* column, the delay in microseconds is denoted. At the very bottom of each table, in the *STDEV* rows, the standard deviations of the results are included in order to indicate how much the delay values vary.

Table 3.8: 8 copies of sinlgepath_1_50 measurement results.

ID	LTH	LTH Delay (ns)
1	107	428
2	109	436
3	109	436
4	110	440
5	108	432
6	109	436
7	107	428
8	109	436
STDEV	1	4

Table 3.9: 8 copies of sinlgepath_1_100 measurement results.

ID	LTH	LTH Delay (ns)
1	225	900
2	223	892
3	221	884
4	227	908
5	221	884
6	225	900
7	222	888
8	225	900
STDEV	2,05	8,23

Table 3.10: 8 copies of sinlgepath_2_50 measurement results.

ID	LTH	LTH Delay (ns)
1	59	236
2	58	232
3	59	236
4	57	228
5	57	228
6	59	236
7	57	228
8	58	232
STDEV	0,86	3,46

Table 3.11: 8 copies of sinlgepath_2_100 measurement results.

ID	LTH	LTH Delay (ns)
1	118	472
2	120	480
3	119	476
4	118	472
5	118	472
6	118	472
7	118	472
8	117	468
STDEV	0,82	3,31

Table 3.12: 8 copies of sinlgepath_3_50 measurement results.

ID	LTH	LTH Delay (ns)
1	96	384
2	96	384
3	97	388
4	97	388
5	98	392
6	95	380
7	94	376
8	97	388
STDEV	1,19	4,79

Table 3.13: 8 copies of sinlgepath_3_100 measurement results.

ID	LTH	LTH Delay (ns)
1	200	800
2	196	784
3	201	804
4	197	788
5	196	784
6	198	792
7	195	780
8	199	796
STDEV	1,98	7,93

Table 3.14: 8 copies of spypath_3_1_50 measurement results.

ID	LTH	LTH Delay (ns)
1	106	424
2	106	424
3	103	412
4	108	432
5	105	420
6	106	424
7	106	424
8	104	416
STDEV	1,41	5,65

Table 3.15: 8 copies of spypath_3_1_100 measurement results.

ID	LTH	LTH Delay (ns)
1	217	868
2	216	864
3	213	852
4	219	876
5	213	852
6	218	872
7	214	856
8	219	872
STDEV	2,36	9,47

Table 3.16: 8 copies of spypath_3_2_50 measurement results.

ID	LTH	LTH Delay (ns)
1	104	416
2	104	416
3	106	424
4	104	416
5	107	428
6	103	412
7	103	412
8	105	420
STDEV	1,31	5,26

Table 3.17: 8 copies of spypath_3_2_100 measurement results.

ID	LTH	LTH Delay (ns)
1	213	852
2	218	872
3	213	852
4	216	864
5	217	868
6	281	1124
7	216	864
8	217	868
STDEV	21,65	86,63

Table 3.18: 8 copies of not3_33_50 measurement results.

ID	LTH	LTH Delay (ns)
1	98	392
2	97	388
3	98	392
4	97	388
5	98	392
6	97	388
7	98	392
8	98	392
STDEV	0,48	1,93

Table 3.19: 8 copies of not3_33_100 measurement results.

ID	LTH	LTH Delay (ns)
1	199	796
2	201	804
3	201	804
4	200	800
5	198	792
6	199	796
7	200	800
8	199	796
STDEV	0,99	3,96

Additionally, 20 times chained versions of singlepath_3 and its two trojan added versions are copied 30 to 32 times and their delay variances are measured. Figure 3.1 can be used as reference to which of the 32 samples are placed on which part on the layout.

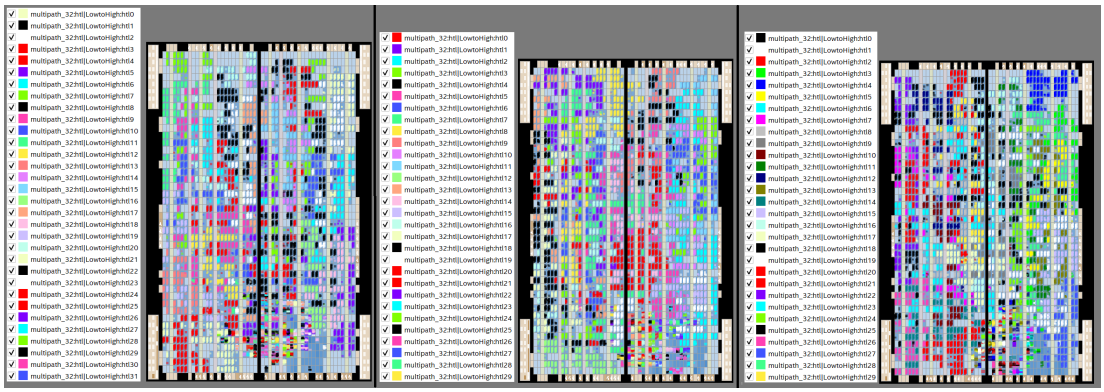

Figure 3.1: 30-32 times duplicated sinlepath_3 and its trojan added versions on the chip planner.

Table 3.20: 32 copies of singlepath_3_20 measurement results.

ID	LTH	LTH Delay (ns)
1	38	152
2	37	148
3	38	152
4	38	152
5	38	152
6	38	152
7	38	152
8	37	148
9	38	152
10	38	152
11	38	152
12	38	152
13	38	152
14	38	152
15	38	152
16	38	152
17	38	152
18	38	152
19	38	152
20	37	148
21	38	152
22	38	152
23	38	152
24	38	152
25	39	156
26	37	148
27	38	152
28	38	152
29	38	152
30	38	152
31	38	152
32	38	152
STDEV	0,38	1,53

Table 3.21: 30 copies of spypath_3_1_20 measurement results.

ID	LTH	LTH Delay (ns)
1	41	164
2	41	164
3	41	164
4	41	164
5	41	164
6	41	164
7	42	168
8	42	168
9	41	164
10	42	168
11	42	168
12	41	164
13	42	168
14	41	164
15	42	168
16	42	168
17	41	164
18	41	164
19	41	164
20	41	164
21	42	168
22	41	164
23	40	160
24	40	160
25	41	164
26	41	164
27	42	168
28	41	164
29	42	168
30	42	168
STDEV	0,58	2,34

Table 3.22: 30 copies of spypath_3_2_20 measurement results.

ID	LTH	LTH Delay (ns)
1	42	168
2	42	168
3	40	160
4	41	164
5	41	164
6	43	172
7	41	164
8	41	164
9	41	164
10	41	164
11	41	164
12	41	164
13	41	164
14	42	168
15	42	168
16	41	164
17	41	164
18	41	164
19	42	168
20	41	164
21	42	168
22	41	164
23	40	160
24	40	160
25	41	164
26	41	164
27	41	164
28	41	164
29	42	168
30	40	160
STDEV	0,68	2,74

In all of the measurements, variances are observed. Also, the measurements are performed multiple times and the results of one single selection didn't vary more than one unit. In other words, the paths that produce high always produced high and vice versa.

In some of the paths, these variances are even more significant. For example, in spypath_3_2_100, the usual outputs vary from 213 to 218 and there is even an unusual spike with the value of 281.

In the measurement results of 20 times chained paths, the variances were very minimal. We think paths are not synthesized using spatially close LUTs, which is the reason.

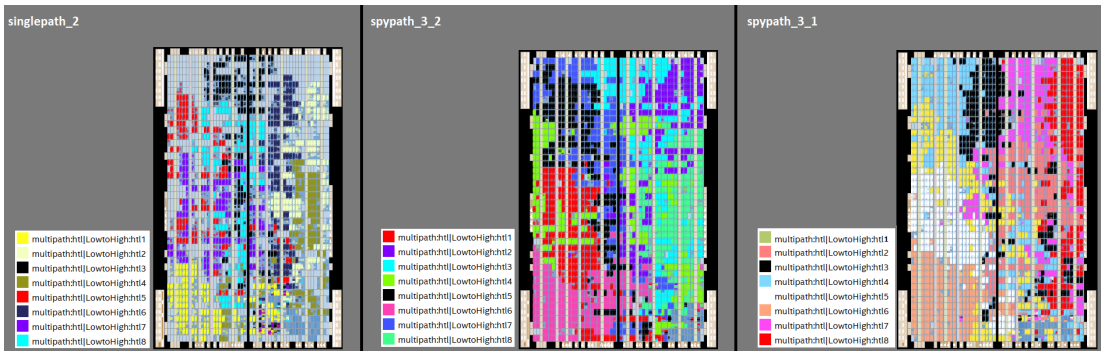

Figure 3.2: Different duplicate paths on Chip Planner

The exact places of each path on the layout isn't taken into account in the measurements. Furthermore, the Quartus doesn't place the paths very uniformly on the chip. As it can be seen in Figure 4.1, some of the paths are placed relatively tightly, for instance, the ones on the bottom left. However, some others are mingled a lot.

There are more or less some variances observed in all of the paths. This proves that placing the design on different parts on the layout does make a difference. However, it doesn't necessarily prove that the cause of such variance is due to the internal variations in the chip. In order to get more accurate and precise insight, further research is needed.

4. CONCLUSION

One way to detect the presence of a trojan hardware is to look for suspicious delays. The first aim of this project was detecting a suspicious delay on an FPGA design.

In first part of the project, the delays of one example path and two trojan added versions of it are measured. Measuring one single units of each path wouldn't create an observable difference. Therefore, these paths are chained multiple times to multiply the delays by a factor. On average, a difference of 0.8 nanoseconds is captured between the clean path and its trojan versions.

The secondary aim of this project was to place the same design on different parts of the layout and detecting how much the delay values vary due to the in-chip variations. For this purpose, copies of the same design are placed on the chip multiple times at the same time. Each of these copies' delays are measured.

The measurements have given results with variations. There were at least some minimal variations on the results on all of the paths, with one giving even an extraordinarily distinct result. In the light of these measurements, we can say that placing the desing on different parts of the layout does make a difference on the delay. However, the placements aren't done so uniformly by the Quartus software. We still can't say the cause of such variations on the results is surely caused by the in-chip variations. It might be caused by stretched routing paths on some of the copies as well. During the measurement process, the places of each copy aren't taken into account.

With the ambition of improving this method further, new measurements that pays attention to the specific locations of each copy of the design can be done. Furthermore, Quartus' Fitter options can be investigated in order to place the designs in a more uniform manner.

Lastly, all measurements in this project are done on one single FPGA board, DE0-CV. The effectiveness of the method can be tested further on multiple different boards, preferably with higher clock frequencies and a greater layout area.

BIBLIOGRAPHY

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

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

[3] Verilog HDL Synthesis Attributes and Directives, [Online]. Available: https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/hdl/vlog/vlog_file_dir.htm.

[4] Altera Phase-Locked Loop, [Online]. Available: https://www.intel.com/content/www/us/en/docs/programmable/683359/17-0/altera-phase-locked-loop-ip-core-user-guide.html.

[5] All About Circuits, "Purpose and Internal Functionality of FPGA Look-Up Tables," [Online]. Available: https://www.allaboutcircuits.com/technical-articles/purpose-and-internal-functionality-of-fpga-look-up-tables/.

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

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

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