Ring oscillator based hardware Trojan detection

Tamzidul Hoque
University of Toledo

Follow this and additional works at: http://utdr.utoledo.edu/theses-dissertations

Recommended Citation
Hoque, Tamzidul, "Ring oscillator based hardware Trojan detection" (2015). Theses and Dissertations. 1882.
http://utdr.utoledo.edu/theses-dissertations/1882

This Thesis is brought to you for free and open access by The University of Toledo Digital Repository. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of The University of Toledo Digital Repository. For more information, please see the repository's About page.
A Thesis

entitled

Ring Oscillator Based Hardware Trojan Detection

by

Tamzidul Hoque

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in

Electrical Engineering

Dr. Mohammed Niamat, Committee Chair

Dr. Mansoor Alam, Committee Member

Dr. Weiqing Sun, Committee Member

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

May 2015
Hardware Trojans are malicious circuits which can be secretly implanted in integrated circuits by unscrupulous third party manufacturers for the purpose of spying or stealing information from the circuit. This has become a matter of concern with the increase in outsourcing of semiconductors which are used both in military and commercial sectors. It has been observed that due to the presence of process variation, environmental variation, and measurement noise; a stealthy Trojan may go undetected. In the first part of this thesis, we study the NOT and NAND based ring oscillators (ROs) as power monitors for detecting these Trojans. A network comprising of 7 ROs is implemented using the ISCAS’85 c2670 benchmark on several Xilinx Spartan-3E FPGAs. The results demonstrate that the impact of Trojans on the frequency of nearby ROs is noticeably larger for NAND based structure compared to the NOT one, thus making the NAND based design more attractive for the detection of Trojans.
In the later part of our work, a circuit partitioning based approach is proposed which facilitates the detection of Trojans. The ratio of the power consumed by the Trojan to the power consumed by the host circuit plays a vital role in detection of Trojans using any power based side channel analysis method. Partitioning the circuit allows us to control the switching activity of the divided areas independently. Therefore, for a chip with uniform switching activity across the chip area, overall dynamic power consumption can be reduced to almost $\frac{1}{n}$ of the total dynamic power, if the chip is divided into $n$ number of partitions. In this work, the circuit under authentication (CUA) is split into two sub circuits and the ring oscillator’s frequency values are observed while keeping one of the two sub circuits inactive. Experimental results show a higher percentage of change in the ring oscillator’s frequencies during the partial activation of CUA which magnifies the discrepancy between the Trojan free and Trojan inserted circuits.
This thesis is dedicated to my family, teachers and friends for making me who I am today.
Acknowledgements

I am thankful to Almighty Allah, most Gracious, who in His infinite mercy has guided me to complete this thesis work. I would like to thank my advisor Dr. Mohammed Niamat for advising me throughout my Master's research under him. His continued support and encouragement has helped me to overcome all the difficulties. My sincere thanks to Dr. Mansoor Alam and Dr. Weiqing Sun for being a part of my thesis committee. Special appreciation goes to Dr. Mansoor Alam for considering me for financial support throughout my Master’s program.

I would like to thank my lab mates Muslim Mustapa, Fathi Amsad, and Majed Albogomi for spending their precious time to help me out in solving intricate issues I faced. The knowledge earned through debate and conversation with my lab mates is priceless.

Sincere thanks to my parents, my entire family and friends for their continuous love, support, understanding and motivation, that made this thesis possible.
Table of Contents

Abstract ........................................................................................................................................ iii
Acknowledgements ................................................................................................................ vi
Table of Contents .................................................................................................................. vi
List of Tables ......................................................................................................................... xi
List of Figures ........................................................................................................................ xii
List of Abbreviations .............................................................................................................. xiv
List of Symbols ....................................................................................................................... xv
1 Introduction and Research Overview ................................................................................. 1
  1.1 Introduction ..................................................................................................................... 1
    1.1.1 Hardware Trojan ........................................................................................................ 3
  1.1.2 Practicality of the Threat ........................................................................................... 3
  1.2 Goals of the Research ..................................................................................................... 4
  1.2.1 Assessment of NAND Based Ring Oscillator ............................................................ 5
  1.2.2 Trojan Detection using Circuit Partitioning Technique ............................................ 5
  1.3 Thesis Organization ......................................................................................................... 6
2 Background ............................................................................................................................ 8
  2.1 Hardware Trojan Classification ....................................................................................... 8
    2.1.1 Insertion Phase .......................................................................................................... 8
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1.1</td>
<td>Specification Phase</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1.2</td>
<td>Design Phase</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1.3</td>
<td>Fabrication Phase</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1.4</td>
<td>Assembly Phase</td>
<td>10</td>
</tr>
<tr>
<td>2.1.1.5</td>
<td>Testing Phase</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2.1</td>
<td>System level</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2.2</td>
<td>Development Environment</td>
<td>12</td>
</tr>
<tr>
<td>2.1.2.3</td>
<td>Register Transfer Level</td>
<td>12</td>
</tr>
<tr>
<td>2.1.2.4</td>
<td>Gate Level</td>
<td>13</td>
</tr>
<tr>
<td>2.1.2.5</td>
<td>Transistor Level</td>
<td>13</td>
</tr>
<tr>
<td>2.1.2.6</td>
<td>Layout Level</td>
<td>13</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Activation Mechanism</td>
<td>14</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Effects</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Class of Protections</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Trojan Detection Approaches</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1.1</td>
<td>Logic Testing</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1.2</td>
<td>Side-Channel Analysis</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1.3</td>
<td>Design for Security (DFS) Approach</td>
<td>20</td>
</tr>
<tr>
<td>2.2.1.3.1</td>
<td>DFS to Prevent Trojan Insertion</td>
<td>20</td>
</tr>
<tr>
<td>2.2.1.3.2</td>
<td>DFS to Facilitate Trojan Detection</td>
<td>22</td>
</tr>
<tr>
<td>2.2.1.4</td>
<td>Run Time Monitoring</td>
<td>24</td>
</tr>
<tr>
<td>2.2.1.4.1</td>
<td>Configurable Security Monitors</td>
<td>24</td>
</tr>
</tbody>
</table>
2.2.1.4.2 Variant Based Parallel Execution .................24

2.2.1.4.3 Hardware-Software Approach .......................25

Assessment of NAND Based Ring Oscillator for Hardware Trojan Detection....26

3.1 Background and Motivation .................................................................26

3.1.1 Power Distribution Network and Impact of Power Supply Noise ....27

3.1.2 Ring Oscillator as Power Monitor......................................................28

3.2 Proposed Approach..................................................................................30

3.2.1 Ring Oscillator Network Structure.......................................................31

3.3 Implementation .......................................................................................33

3.3.1 Implementation on FPGA ..................................................................33

3.3.1.1 Architecture of the Xilinx Spartan-3E FPGA.................................33

3.3.1.2 NOT Gate Based RO Implementation ..........................................34

3.3.1.3 NAND Gate Based RO Implementation......................................36

3.3.2 Trojan Design......................................................................................40

3.4 Measurement Flow..................................................................................40

3.5 Experimental Results and Analysis .......................................................41

3.5.1 Trojan Impact Analysis .......................................................................41

3.6 Conclusion and Future Work ..................................................................44

Hardware Trojan Detection using Circuit Partitioning Technique ............45

4.1 Motivation ..............................................................................................46

4.2 Overview of Power Dissipation on ASICs and FPGAs ..........................47

4.2.1 Power dissipation in CMOS ...............................................................47

4.2.1.1 Dynamic Power Dissipation due to Switching of Gates ....47
References ........................................................................................................................................72

A Source Codes .....................................................................................................................................81

A.1 VHDL Code for a Five-Stage NAND Gate based Ring Oscillator..........................81
A.2 VHDL Code for a Five-Stage NOT Gate based Ring Oscillator.........................84
A.3 VHDL Code for LUT Instantiation to Implementing the Benchmark..........86
## List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Truth table of a NOT gate with input A and output Y</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>Truth table of a NAND gate with input A &amp; B and output Y</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Percentage of increase (positive values) or decrease (negative value)</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Area overhead for inserting MUXs and DE-MUXs</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Increase in the percentage of variation after partitioning in terms of ratio</td>
<td>66</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Dependability attributes</td>
<td>2</td>
</tr>
<tr>
<td>2-1</td>
<td>Taxonomy of hardware Trojans</td>
<td>11</td>
</tr>
<tr>
<td>2-2</td>
<td>(a) Unmodified inverter gate (b) Altered inverter gate at the layout level</td>
<td>14</td>
</tr>
<tr>
<td>2-3</td>
<td>Various protection mechanism against hardware Trojan attacks</td>
<td>17</td>
</tr>
<tr>
<td>2-4</td>
<td>Trigger coverage, Trojan coverage, and test length</td>
<td>18</td>
</tr>
<tr>
<td>2-5</td>
<td>On-chip current monitors attached to power supply bumps</td>
<td>22</td>
</tr>
<tr>
<td>2-6</td>
<td>Comparison between on-chip and off-chip current measurements in [36]</td>
<td>23</td>
</tr>
<tr>
<td>2-7</td>
<td>Runtime monitoring of Trojan effects using a reconfigurable infrastructure</td>
<td>25</td>
</tr>
<tr>
<td>3-1</td>
<td>RLC model of a simple power line in a power distribution network</td>
<td>27</td>
</tr>
<tr>
<td>3-2</td>
<td>NOT gate based ring oscillator studied in [35]</td>
<td>30</td>
</tr>
<tr>
<td>3-3</td>
<td>Proposed NAND gate based ring oscillator</td>
<td>30</td>
</tr>
<tr>
<td>3-4</td>
<td>Proposed NAND gate based ring oscillator network</td>
<td>32</td>
</tr>
<tr>
<td>3-5</td>
<td>Xilinx Spartan-3E FPGA architecture</td>
<td>34</td>
</tr>
<tr>
<td>3-6</td>
<td>NOT gate based ring oscillator implementation</td>
<td>35</td>
</tr>
<tr>
<td>3-7</td>
<td>Post-route simulation of the NOT gate based ring oscillator</td>
<td>36</td>
</tr>
<tr>
<td>3-8</td>
<td>NAND gate based ring oscillator implementation</td>
<td>37</td>
</tr>
<tr>
<td>3-9</td>
<td>Post-route simulation of the NAND gate based ring oscillator</td>
<td>38</td>
</tr>
<tr>
<td>3-10</td>
<td>Position of the benchmark, ROs and Trojan in the FPGA editor</td>
<td>39</td>
</tr>
</tbody>
</table>
3-11 Design of the Trojan stage used .................................................................40
3-12 Trojan impact on NOT gate based ring oscillator frequency ..................42
3-13 Trojan impact on NAND gate based ring oscillator frequency ..................43
3-14 Comparison of Trojan impact on NAND and NOT gate based RO ............43
4-1 The operation of a CMOS inverter ...............................................................48
4-2 Formation of reverse-biased diodes in CMOS .............................................51
4-3 The circuit partitioning technique .................................................................54
4-4 Placement and routing of Sub Circuit 1, mapped on FPGA .......................58
4-5 Placement and routing of Sub Circuit 2, mapped on FPGA .......................59
4-6 Ring oscillators (marked in red) and the Trojans ........................................60
4-7 Single block of the Trojan circuit ...............................................................61
4-8 Frequency counter and RO selector (marked in red) .................................62
4-9 Comparison of Trojan1’s impact over frequencies of all the ROs ..............65
4-10 Comparison of Trojan2’s impact over frequencies of all the ROs ..............66
List of Abbreviations

ASIC ....................... Application Specific Integrated Circuit
CB .......................... Connection Block
CLB .......................... Configurable Logic Block
CLK .......................... Clock
CMOS ........................ Complementary Metal Oxide Semiconductor
CRP .......................... Challenge-Response Pair

FF .......................... Flip-Flop
FPGA ........................ Field Programmable Gate Array

IC .......................... Integrated Circuit
I/O .......................... Input/ Output
IOB .......................... Input Output Block
IP .......................... Intellectual Property

LFSR ........................ Linear Feedback Shift Register
LUT .......................... Look-Up Table

MOSFET ........................ Metal Oxide Semiconductor Field Effect Transistor
MUX .......................... Multiplexer

PUF .......................... Physical Unclonable Function

RO .......................... Ring Oscillator
RON .......................... Ring Oscillator Network

UCF .......................... User Constraint File

VHDL ........................ VHDL Hardware Description Language
VLSI .......................... Very Large Scale Integration
List of Symbols

$\rho_{li}$ ...................... Voltage division coefficient
$\alpha$ ........................... Velocity saturation index
$\mu_g$ ............................ Carrier mobility
$k_g$ .............................. Gate-dependent constant
Chapter 1

Introduction and Research Overview

1.1 Introduction

Computing systems have become an integral part of the human society. Integrated circuits are the core elements of all the computing system that we use in our daily life. These ICs are basically a collection of various electronic components implanted into a piece of semiconductor material, commonly silicon. In spite of accommodating up to several billions of transistors and other electronic components, the dimension of an IC is usually as small as a fingernail. Some of the numerous fields where ICs have vast applications are computing, automotive, defense, aerospace, medical, telecommunication, networking, consumer appliances, portable devices, and wireless applications.

Dependability is considered as a key feature of these integrated circuits when applied to any specific field. Dependability reflects how closely the service being delivered by the device aligns with the functional specification. The functional specification is expressed in terms of the functionality and performance, whereas service
delivered by the device is measured by its behavior in terms of the user’s observation. Dependability constitutes certain sub features such as availability, reliability, safety, integrity, and maintainability as illustrated in Figure 1-1 [1].

Figure 1-1: Dependability attributes

Among the attributes of dependability, features such as availability, integrity, and confidentiality constitute a specific field called security. In computing, software security is a mature field of study. However, recently, a number of occurrences have been reported mentioning infected and counterfeited ICs intruding military systems [2] - [4].
Such incidents indicate that the security of computing systems handling confidential data is at high risk. To confront this threat, hardware security has emerged as an important field of research.

1.1.1 Hardware Trojan

Electronic systems available in today’s commercial, industrial, and military sectors are massive networks of ICs. Proliferation of these devices in everyday life has created a competitive industry where outsourcing of design and fabrication process is considered cost-effective. Thus, a part or whole of the IC supply chain is situated in foreign land, which may not be under surveillance. This provides an opportunity for an adversary to embed functionality not stated in the specification of the device [2]. Such functionality may leak secret information to the enemy, or even disable the device at a specific time in the future. This malicious alteration has been addressed as hardware Trojan in the literature. Incidence of malicious inclusion in hardware has also been reported by the US Department of Defense [3].

1.1.2 Practicality of the Threat

Even though the investigation of the threat of Trojans is relatively recent, the concept of secretly injecting malicious functionality into a device has been prevalent for years. One of the publications of the US National Security Agency History Program states the presence of an electromechanical bug in 16 IBM Selectric typewriters located in the U.S. Embassy in Moscow and the U.S. Mission in Leningrad in the late 1970s [3], [4]. The report states that the Soviet agents secretly installed tiny sensing devices in the
typewriters which captured the contents of official papers typed by embassy secretaries. The contents were then transmitted through the antennas hidden in the embassy walls to the outside world [4].

In modern IC technology where billions of transistors are stacked into one single chip, it is far easier for an adversary to furtively implement additional circuitry or to modify existing design to pursue any sort of malign interest. The Government and industry is already investing in research dedicated to finding a sustainable solution for such a threat. Wally Rhines, CEO of Mentor Graphics, identifies the Trojan attack as the biggest security threat for future semiconductor industry. According to his prediction, most of the future ICs will contain co-processors to monitor the activity within the chip [5]. Bernard Murphy, chief technology officer at Atrenta, dreads that the issue of hardware Trojan is viewed as a theoretical problem, due to the lack of smoking gun evidence and the industry does not have the ultimate solution to detect it right now [6]. The National Science Foundation (NSF) and the Semiconductor Research Corporation (SRC) are funding projects which aim to make the IC resistant to exploitation by hackers who take advantage of hidden Trojan horses and backdoors, implanted into the IC by intellectual property (IP) designed or fabricated from foreign sources. These projects are likely to spend $10 million over several years [7].

1.2 Goals of the Research

The major goal of this research is to improve the ring oscillator structure based side channel analysis technique for Trojan detection. To facilitate the detection of Trojans
using power based analysis, an experiment on NOT gate based ring oscillator network models was done in the past. We aim to contribute to the study in improving this technique in two ways, which are discussed in the following sections.

1.2.1 Assessment of NAND Gate Based Ring Oscillator

It is assumed that the NAND gate based ring oscillator should be more sensitive towards supply voltage fluctuation compared to the NOT gate. This motivates us to pursue experiment over the NAND gate based RO to obtain improved sensitivity for Trojan detection. A ring oscillator network structure composed of seven, five stage NAND gate based ROs is placed on the ISCAS’85 c2670 benchmark, which is the circuit under authentication (CUA). A linear feedback shift register (LFSR) is used as the test pattern generator for the CUA. A similar test bed using NOT gate based RO is designed for comparison purposes. As the hardware validation platform, 10 Xilinx Spartan-3E FPGAs (90nm) are used. This research analyzes the Trojan’s impact on the NAND based ROs placed in different positions across the CUA. It also demonstrates how NAND gate based ROs placed near the Trojan experience a higher percentage of changes due to additional circuit activity compared to NOT gate based RO.

1.2.2 Trojan Detection using Circuit Partitioning Technique

Effectiveness of the ring oscillator network based technique can be improved by obtaining a higher Trojan to circuit power consumption ratio. During the test mode, power consumption is several times higher compared to the normal operation due to the switching activity of the circuit [8]. Thus, a small Trojan circuit’s contribution in power consumption is supposed to be very small. This offers a very small TCP which indicates
that the Trojan might affect the power monitors in an insignificant manner. Thus, detection of Trojans becomes difficult. To overcome this issue, a circuit partitioning technique has been used. This has previously been deployed for facilitating low power testing [9]. A circuit partitioning method divides the circuit into a number of sub circuits. This allows for the activation of one individual sub circuit in the circuit under authentication at a time. Thus, a Trojan hidden in any particular sub circuit obtains a higher TCP as the rest of the CUA is inactive and does not consume dynamic power due to switching. To demonstrate the effectiveness of this method, two ISCAS’85 c1355 benchmarks have been combined into one circuit under verification. This eliminates the need of using a circuit partitioning algorithm. The same LFSR, which supplies test patterns to the combined circuit, is connected to individual sub circuits, where each sub circuit can be supplied test patterns independently, keeping the other portion silent. Ring oscillator frequency values are obtained keeping both sub circuits activated simultaneously and also distinctly. Later, the values are compared to demonstrate how the Trojan’s effect on the ring oscillator’s frequency values become more dominant during the partitioned mode.

1.3 Thesis Organization

In Chapter 2, we present the classification of hardware Trojans and various Trojan detection techniques from previous research.
In Chapter 3, the implementation of NAND and NOT gate based ring oscillator network is described. Later in the chapter, we analyze the results by comparing the data obtained from NAND and NOT gate based implementations.

Chapter 4 presents the implementation of the circuit partitioning technique with the existing ring oscillator based Trojan detection method on the FPGA platform. Analysis of the results is presented, demonstrating how this technique amplifies the Trojan’s effect.

Chapter 5 concludes the thesis. A summary of the proposed research is presented. Major contributions and future work is also discussed.


Chapter 2

Background

In this chapter, classification of various hardware Trojans and available detection approaches are discussed.

2.1 Hardware Trojan Classification

A taxonomy of hardware Trojans is presented in [10]. According to the taxonomy, Trojans have been classified based on five attributes:

1. Phase of insertion,
2. Abstraction level,
3. Activation mechanism,
4. Functionality, and
5. Location.

2.1.1 Insertion Phase

A chip is manufactured in multiple steps from specification to fabrication. Trojans can be classified based on the manufacturing phase during which they are inserted in the chip.
2.1.1.1 Specification Phase

A Trojan can be inserted by manipulating the specification of the chip where all the characteristics of the chip or the system are defined. Characteristics such as delay, area, or power reflect individual fingerprints of the chip. In many Trojan detection approaches, chips are investigated by comparing with the Golden chip’s characteristics identical to the specification. If the specification itself is altered, the detection mechanism will never work. Intended functionality and expected environment (e.g., operating temperature) can also be altered to evade the detection approach.

2.1.1.2 Design Phase

During the design phase the design is mapped onto the intended technology. Various design constraints such as functional, logical, timing, and physical are considered during this phase. In many cases, designers may use third-party intellectual property (IP) blocks and standard cells during design phase to expedite the process. This trend is another source of hardware Trojan threat as pre-silicon verification or simulation might not always be available for such imported cores. However, to detect any undesired modification, pre-silicon verification/simulation and post-silicon testing of all the blocks used in the chip are required [11].

2.1.1.3 Fabrication Phase

During fabrication phase wafers are produced using the mask set derived from the design phase. A very small change in the mask can introduce stealthy functionality.
Similar efforts to design such Trojans are presented in [12] where the authors modify specific parts within the gate’s active area by applying a different dopant polarity. Trojans below the gate level can also be introduced by altering the chemical compositions during fabrication to accelerate failures of power supply and clock grids by increasing the electro-migration in critical circuit components [10].

2.1.1.4 Assembly Phase

A printed circuit board (PCB) is assembled in this phase by putting together various components such as ICs and input/output blocks and other electrical and electronic components. During the assembly phase, it is possible to introduce unshielded wire or even I/O pin with high parasitic capacitance, both of which can lead to information leakage [10]. Therefore, in spite of having trusted ICs in a system, malicious assembly can produce security flaws in the system.

2.1.1.5 Testing Phase

Testing phase is the opportunity to ensure the trustworthiness of the chip or system being manufactured. There is no scope to insert Trojans in this phase, but an unfaithful testing facility can help Trojan inserted chips go undetected. An adversary inserting a Trojan in the design or fabrication phase would try to manipulate the test vectors in a way which allows the Trojan inserted chip remain undetected. Ideally, test vectors are expected to be applied faithfully followed by specified action which accepts, only fault free chips and categorizes (binning) accordingly.
2.1.2 Abstraction Level of Description

Trojans can be described at various levels of hardware abstraction as shown in Figure 2-1. Different abstraction level introduces a different level of furtiveness and varied functionality of Trojans.
2.1.2.1 System level

System level is the highest level of abstraction. The system is defined in terms of the largest building blocks possible. At this level, the architecture of the system is broken down into different components, hardware modules, interfaces, communication protocols and data [10], [13]. A hardware Trojan inserted at system level may be triggered by a module in the compromised hardware. For example, the data input given by a certain interface (e.g., keyboard) can be changed inside the system, prior to the application of the input.

2.1.2.2 Development Environment

Modern IC design often involves electronic design automation (EDA) software tools required for synthesis, simulation, verification, and validation supplied by different third-party vendors. If the EDA tool is deliberately designed to hide a certain functionality of the circuit under test during verification, it will be impossible to detect certain Trojans. Hence, development environment can play a vital role in the act of Trojan insertion.

2.1.2.3 Register Transfer Level

In register transfer level (RTL), the system is described in terms of registers, signals, and combinational logic. If the attacker has access to manipulate the design at RTL level, it may cause a serious threat because of the endless scope of modification and
greater control of the hardware. For example, [16] demonstrates eight diverse Trojan insertion technique at RTL level of an Alpha encryption module.

2.1.2.4 Gate Level

Gate level description of an IC depicts the circuit in terms of the logic gates and interconnects. In most of the research work related to Trojan detection, the Trojan circuit is inserted by modifying the gate level netlist. Compared to behavioral RTL description, it is easier to control the size of the Trojan when modifying the gate level netlist.

2.1.2.5 Transistor Level

Gates can be broken down into transistors. Gate level netlist can modify the functionality but Trojans inserted at the transistor level have higher level of flexibility as they can control power consumption, delay, along with transistor parameters such as threshold voltage, channel length, and oxide thickness [10]. In [35], on-chip trust structures are used as sensors to detect noise (voltage drop) induced by Trojans. High threshold voltage gates have been used in on-chip trust structures to improve their noise sensitivity. If an adversary is able to comprehend the presence of these sensors, he might replace the transistors with the ones with low threshold voltage parameters. This can be an example of a transistor level modification to insert Trojans.

2.1.2.5 Layout Level

In Section 2.1.1.3 it was mentioned how Trojans can be inserted during the fabrication phase by modifying the layout of the IC. Such modification is shown in Figure 2-2, which was done using a layout editor in [12]. At the layout level, it is
possible to insert Trojans by modifying the size of the wires, distances between circuit elements, and changing the distribution of the metal layer [10]. Empty spaces in the layout design can be utilized to insert additional circuitry.

![Diagram of inverter gates](image)

(a) Unmodified inverter gate (b) Altered inverter gate at the layout level [12]

### 2.1.3 Activation Mechanism

Trojans can be classified in terms of their activation mechanism. Some Trojans are activated only under certain conditions. For example, in [14], Trojans are assumed to be activated by multiple triggering of low probability nodes. Activation conditions can be both internal and external. Internal activation of Trojans can be further classified into time based activation (e.g. using counter) and physical condition based (electromagnetic interference, humidity, altitude, atmospheric pressure, etc.) activation. Trojans activated by input from an external switch or any external data input falls under external activation.
based Trojans [10]. It is one of the parameters which demonstrate how difficult it might be to detect the Trojan. Some Trojans are designed to be always active. Such Trojans can affect the system at any time. Always active Trojans can be designed by tempering physical parameters of transistors as discussed in Section 2.1.2.4.

2.1.4 Effects

The adversary inserts the Trojan with a goal to cause a certain undesirable effect. Therefore, Trojans can be classified according to these effects in the following manner [10]:

1. Change of functionality: Trojan effects might lead to functionality which is not stated in the specification of the device. For example, a Trojan might lead a random number generator to generate a certain number desirable to the adversary. A Trojan inserted in a global positioning system (GPS) might modify the position data generated by the GPS. This can lead to several catastrophic conditions.

2. Reduce reliability: Trojans which downgrade the performance fall under this category. It might cause the device to perform poorly after a certain number of operations. For example, introducing a high parasitic capacitor near a high transition probability node can cause significant power consumption which can drain the battery of a portable device.

3. Leak information: We have already discussed in Section 2.1.1.3 about leaking of sensitive information by a malicious inclusion in the PCB. Computers have built-in thermal sensors to detect heat fluctuations. The sensors sends signal to the internal fan to cool down the system to avoid possible damage to the
motherboard. It has been demonstrated that if the computer can be compromised then using this temperature sensor it is possible to transmit eight bits of data to an adjacent computer (which is under the attackers control) using only heat emission [15]. Data can also be leaked by radio frequency transmission and also via interfaces like RS-232 and JTAG [10].

4. Denial-of-service: Denial-of-service (DoS) attacks are usually done to make a system or resource unavailable to its intended user. For instance, in [16], a DoS Trojan is inserted with only 0.024% area overhead on FPGA. When the “F12” key is pressed in the keyboard, the triggered Trojan locks the device and ignores all the inputs unless the FPGA is reprogrammed.

2.2 Class of Protections

According to one of the latest surveys of the state-of-the-art countermeasures against Trojan attacks in [17], research work on Trojan detection methodologies is classified into three broad categories:

1) Trojan detection approaches,

2) Design for security (DFS) approaches, and

3) Runtime monitoring approaches.

Figure 2-3 shows a broad classification of the countermeasures against hardware Trojans.

2.2.1 Trojan Detection Approaches

These techniques can be dedicated to detect Trojans at pre-silicon stage or post-silicon stage. Detection mechanism can be done with logic testing where dedicated test patterns are generated and applied to detect Trojans. Another mechanism is the side-
channel analysis approach, where detection is done by comparing parameters such as power, delay, temperature, or electromagnetic (EM) radiation between Trojan infected and Golden (Trojan free) ICs [17].

Figure 2-3: Various protection mechanism against hardware Trojan attacks

2.2.1.1 Logic Testing

A successful logic testing approach requires the Trojan’s trigger nodes to be activated with appropriate test patterns which will cause observable change at the output. Due to large number of inputs, exhaustive testing is not a viable option for most of the chips. Thus, researchers are focusing on developing test patterns considering the nodes
which might be more vulnerable to Trojan attacks. One of the research methods entitled “multiple excitation of rare occurrence: (MERO)” is described in [14], where a compact set of test patterns is developed which minimizes time and cost but maximizes Trojan detection coverage. By detecting low probability conditions at the internal nodes, optimal set of vectors are developed which can trigger the chosen low probability nodes individually to their rare logic values multiple times (N times, where “N” is a user-specified parameter). Such activation of rare nodes is not possible via random patterns. Thus, it improves the possibility of triggering Trojans compared to random patterns. Trigger coverage and Trojan coverage are the two parameters demonstrating the effectiveness of this test generation approach. Figure 2-4 shows that the Trojan detection coverage increases for higher values of “N,” with the adjustment of increased test length [14].

Figure 2-4: Trigger coverage, Trojan coverage and test length for different values of “N,” using the MERO approach [14]

Another test pattern generation approach involves developing guided test patterns which focus on small but vulnerable areas of the chip where the patterns reveal unusual
or unjustifiable activity [20]. In [21], a tool called FANCI detects such vulnerable nets in terms of their covertness through Boolean functional analysis.

2.2.1.2 Side-Channel Analysis

Developing a testing scheme considering the threat of an unknown functionality is challenging. Furthermore, test patterns generated by using the Trojan-free circuit netlist cannot focus on triggering the nodes connected to the Trojan circuit [22]. An intelligent adversary tries to design the Trojan in a way that is only triggered under rare conditions. Thus, post fabrication functional and structural testing conducted using a limited number of test patterns is usually not reliable to define the trustworthiness of a fabricated IC received from an external foundry. In the previous section, it has already been discussed that, exhaustive testing covering all possible input patterns is also not a practical solution for most chips because of large time requirements. Therefore, it is expected that full activation of Trojans followed by any observable change at the output should be a very rare case under usual testing schemes.

Nonetheless, partial activation of Trojan is possible. During testing or normal operation, for a very brief period of time, the Trojan circuit may receive input patterns which activate some of its gates. Occurrence of signal transition at the input of the Trojan gates is very likely to cause power or delay variation. Many of the Trojan detection techniques are based on observing the possible change in the IC’s side channel behavior due to Trojan’s partial activation. This procedure provides an effective workaround, eliminating the need of exhaustive testing. Side channel analysis using current, [28] delay, [26], [27] and transient power [23]-[25] has proven to be an effective Trojan detection approach. A golden or Trojan free IC signature is required for comparison
purposes in many of these side channel analysis methods. Such signatures might be obtained by destructive reverse engineering approaches or from the software simulation of the original design [10].

One of the critical issues regarding the side channel analysis method is the effect of process and environmental variation and measurement noise. These variations make it difficult to isolate the deviation of side channel parameters caused by the Trojan, which is usually smaller compared to the process and environmental variations. Various techniques to increase the probability of partial activation of Trojan gates have been developed [14], [29]. Facilitating localized switching to magnify the Trojan’s contribution is also proven to improve Trojan detection rates through the side channel method [30].

2.2.1.3 Design for Security (DFS) Approach

The aim of the DFS methodology is to tackle the threat of Trojans by introducing changes to the design. This is similar to the built in self-test (BIST) technique used to detect faults in ICs. The DFS technique can be implemented for two different objectives. One motivation is to prevent Trojan insertion, and another motivation is facilitate Trojan detection [17].

2.2.1.3.1 DFS to Prevent Trojan Insertion

Prevention is done using two different approaches: 1) obfuscation-based approaches; and 2) layout-filler approaches [17]. Obfuscation-based approaches involves designing the system in a way which makes it difficult for the attacker to comprehend the function and structure of the IC. Thus, it becomes hard for the opponent to insert a Trojan keeping the behavior and functionality of the IC identical [31], [32]. In [32] obfuscation
technique is implemented by employing two different modes of circuit operation: normal mode and the obfuscated mode. During normal operational mode, the IC performs desired activity and provides expected outputs. Obscure mode provides incorrect functionality for some patterns. This kind of modification makes it difficult for an attacker to detect the rare nodes inside the IC. This leads to easy-to-detect Trojan insertion by the enemy side. Furthermore, it also makes some Trojans active only during obfuscation mode. Such Trojans are stated as “benign Trojans” in [32]. Switching from the obfuscated to the normal mode depends on a specific input sequence, which is stated as the key.

The layout-filler approaches fill all the vacant space by adding additional circuitry which prevents the attacker from inserting additional circuit components. Filled spaces must be implemented in a crafty way so that attackers are unable to distinguish the filled spaces in a layout and substitute Trojan circuits. Built-in-self-repair (BISA) techniques, demonstrated in [33], is a layout filler technique where unused spaces in an IC are filled with functional standard cells instead of filler cells. This technique makes it very difficult for an opponent to find any space on the device for Trojan insertion. Moreover, BISA is temper resistant, which means attempts to manipulate BISA architecture is easy to detect. If one or more of the filler standard cells are replaced with the Trojan cells, an incorrect signature is generated during the authentication process. Nevertheless, similar filler approaches, including BISA, will not be able to prevent the modification, if modification in the layout does not require extra space.

Authors of [34] have proposed a prevention mechanism against Trojan insertions. It involves a split-manufacturing process. The aim is to make Trojan insertion difficult by
hiding the circuit’s intent. It has been proposed that, the front-end-of-the-line (FEOL) and bank-end-of-the-line (BEOL) process steps will be performed in separate fabrication facilities. By obstructing the exchange of design information between BEOL and FEOL facilities, attackers are prohibited from understanding the design. Therefore, causing a malicious alteration becomes difficult, as it requires realizing the design and manipulating the gates and interconnects in a shrewd manner.

2.2.1.3.2 DFS to Facilitate Trojan Detection

These methods are classified into three major categories which are discussed below.

1) On-Chip Security Monitors: Due to the small size, detection of Trojans using side channel analysis is difficult. Furthermore, process and environmental variation and measurement noise makes it more difficult by obfuscating the fluctuations caused by the Trojan. In [36], integration of transient current sensors in a chip provides significantly higher detection sensitivity than conventional off-chip current monitoring. In Figure 2-5, the on-chip monitoring scheme is shown. Effectiveness of the on-chip current monitors is shown in Figure 2-6, where the red bars denote current values from the chip with the Trojan, and the blue bars denote current values of the Trojan free chip, considering the effect of process variations.

![Figure 2-5: On-chip current monitors attached to power supply bumps [36].](image)
Techniques configuring circuit paths into ring oscillators to detect Trojans in terms of delay variations are proposed by researchers in [37]. Ring oscillator networks have been used as power monitors in [35]. This thesis is focused on improving the detection resolution of this method. A detailed discussion, elaborating this method, is presented in the next chapter.

2) Removing Rare-Triggered Nets: In [29], a dummy scan flip-flop insertion procedure is proposed. This increases the transition probability of nets beyond a specific threshold. Thus, an adversary finds it difficult to select nodes, which will not cause transitions to Trojan gate inputs too often.

3) Trojan-to-circuit ratio (TCR): To magnify the Trojan’s effect on side channel parameters, Trojan-to-circuit ratio in terms of switching and power is very important. In [38], a layout aware scan-cell reordering technique has been proposed, which provides
localize switching activity to one region, while limiting it in other regions of the circuit under authentication.

2.2.1.4 Run Time Monitoring

It’s recommended that, Trojans should be exposed before ICs are deployed in the field. Contemporary approaches cannot promise complete coverage of all the diverse types of Trojans. Therefore, online tracking of computations can considerably reduce the possible disastrous consequences of these Trojan attacks. When implemented, these methods would either ensure disabling the chip containing malicious logic, or continue normal operation, bypassing the infected portions. The notable types of tracking procedures are the followings:

2.2.1.4.1 Configurable Security Monitors

This procedure is based on incorporating reconfigurable logics in a System on Chip (SoC). It observes the operation of the system in real time with the help of security monitors (SMs) [40], [41]. These SMs can configure finite state machines (FSMs) in the system. FSMs are capable of observing vulnerable signals. Furthermore, creation of FSMs doesn’t affect the normal operation of the circuit. Thus, inspections for Trojans are carried out simultaneously with the regular circuit operation. If a deviation is detected, SMs can be set up to perform security checks that will prevent access to memory space, or entering test mode in usual operation. Figure 2-7 shows a SoC designed with SMs.

2.2.1.4.2 Variant Based Parallel Execution

This approach uses multicore hardware with distributed software scheduling process to test the authenticity of the hardware when in operation [42]. The method includes scheduling and implementing functionally similar variants on various processing
elements (PEs). In the case of a mismatch in processing by any of the PE, a new PE takes over the job till the infected PE is marked. The success of this procedure depends on efficient initiation of variants.

Figure 2-7: Runtime monitoring of Trojan effects using a reconfigurable infrastructure on a System on Chip [39].

2.2.1.4.3 Hardware-Software Approach

In [43], a verifiable “hardware guard” unit which is external to the processor is used for runtime monitoring. Another similar hardware/software approach consists of monitoring both the design process and normal circuit operation [44]. It checks the unused circuitry during design process and marks it as suspicious.
Chapter 3

Assessment of NAND Based Ring Oscillator for Hardware Trojan Detection

In this chapter, a NAND based ring oscillator structure for the detection of hardware Trojans is presented.

3.1 Background and Motivation

One of the major challenges in Trojan detection using the side channel analysis is the small size of the Trojan. This leads to an insignificant contribution to side channel parameters. Even with the best available measuring instruments and reasonable input patterns, Trojan’s effect is hard to distinguish from the deviations caused by the process and environmental variation, and the measurement noise. Therefore, measurement techniques with a higher sensitivity are required for the purpose of Trojan detection. On-chip measurement techniques are considered promising, and various research works have already been done on these topics [35]-[37]. The concept of using an on-chip ring oscillator structure as a power monitor to detect Trojans was first proposed in [45]. Based on the size of the circuit under authentication (CUA), several ring oscillators constituting a ring oscillator network (RON), may be required. Power supply noise is one of the
parameters which effects the frequency of a ring oscillator. Therefore, the frequency of ring oscillators in Trojan inserted and Trojan free ICs should be different.

### 3.1.1 Power Distribution Network and Impact of Power Supply Noise

Due to the compactness of the power supply distribution network of modern ICs, transition in some logic gates can have an impact on the power supply of the nearby gates. If two gates share the same VDD line, transition induced noise in one gate impacts the supply voltage of the other gates [46]. Figure 3-1 shows a simple power line model. This represents one row in standard cell design, supplied by VDD.

![RLC model of a simple power line in a power distribution network.](image)

The power rail to the upper metal layer in the power distribution network is connected through VDD. The adjacent cells are represented as current source. Nodes G1, G2, and G3 connects these cells. Each current source/cell has its individual contribution to the overall noise. This contribution is described by Eqn. (1-1). Here \( V_1, V_2, V_3 \) are the voltages at nodes G1, G2, and G3 respectively. From the equation it can be observed that the contribution to power supply noise by cell 1, which is \( V_1 \), is not only dependent on power noise at node 1 (\( V_{11} \)); it is also dependent on the cell 1’s effect over the other
nodes represented via voltage division coefficient $\rho_{ii}$ and the voltage of the neighboring node ($V_{22}, V_{33}$).

$$
V_1 = V_{11} + \rho_{21}(\omega) * V_{22} + \rho_{31}(\omega) * V_{33}
$$

$$
V_2 = \rho_{12}(\omega) * V_{11} + V_{22} + \rho_{32}(\omega) * V_{33}
$$

$$
V_3 = \rho_{13}(\omega) * V_{11} + \rho_{23}(\omega) * V_{22} + V_{33}
$$

(1-1)

Therefore, it can be said that, any particular gate’s transition has impact on nearby gates connected to the same VDD line. Similarly, in the case of the Trojan inserted chip, switching gates in the Trojan is supposed to cause small voltage drop on the VDD line and the ground bounce on the VSS line. If we apply the same input pattern to the Trojan-free (Golden) and Trojan-inserted (infected) IC, power supply noise will differ.

### 3.1.2 Ring Oscillator as Power Monitor

The frequency of this RO is determined by the total delay of all the inverters, in the presence of supply voltage and process variations. The total delay of a $n$-stage RO is approximated as $2 * n * t_d$, where the oscillation frequency is modelled as [45],

$$
f = \frac{1}{2 * n * t_d}
$$

(1 - 2)

Here, $t_d$ is the delay of a single inverter. The delay of a circuit can be affected due to power supply noise (or voltage drop). The more the voltage drops, the more the increase in delay of the gates. The frequency of an $n$-stage RO can also be determined from Eqn. (1-3) [64]

$$
f = \frac{\mu_g \times (V_{DD} - V_{TH})^\infty}{2n \times k_g}
$$

(1 - 3)
In Eqn. (1-3), $\alpha$ is the expression of velocity saturation index, $V_{DD}$ is the supply voltage, $V_{TH}$ represents the threshold voltage, $\mu_g$ is the carrier mobility, and $k_g$ is defined as a gate-dependent constant [45]. Under the presence of a malicious inclusion, there is a rise in load along with a voltage drop of $\Delta V_{TROJ}$, which is introduced by changing the expression of the frequency of the RO to,

$$f = \frac{\mu_g \times (V_{DD} - \Delta V_{TROJ} - V_{TH})}{2n \times k_g}$$

(1-4)

Thus, if there is a change in the supply voltage of any inverters in the RO, the frequency of the RO is affected. Consequently, changes in the frequency of a RO could be an indication of additional voltage drop or malicious inclusion [45].

Taking advantage of this behavior, it was surmised that any addition or removal of gates should impact the nearby RO which is sourced by the VDD line connected to the tempered area of the IC [35]. This methodology was implemented as a solution to the hardware Trojan problem in application specific integrated circuits (ASICs). The effectiveness of the technique was demonstrated on FPGA and ASIC platforms [45]. Furthermore, Trojans of various sizes and switching activities were detected. One of the key observations of the obtained results revealed that even for a large Trojan with adequate switching activity, Trojan’s effect on the nearby RO frequency is obfuscated due to large process variation, environmental variation and measurement noise. When tested on ASICs, a Trojan of 0.87% area of the CUA causes a maximum of 2.5% variation to the nearest RO’s frequency. However, frequency values fluctuated around 8.05% due to the intra-die variation, and 16.67% due to the inter-die variation [35]. Still
Trojan detection was possible with the help of spatial locality analysis where percentage of variation for RO frequency for some ROs is noticeably higher than the ROs placed far from the Trojan. Thus, a RO structure which is more sensitive to the voltage drop is highly desirable.

3.2 Proposed Approach

The ring oscillator used in the previous study [45] was composed of one NAND gate and four NOT gates as shown in Figure 3-2. However, five NAND gate based RO structure as shown in Figure 3-3 should be more sensitive as one of the inputs of each stage is connected to the VDD line. However, no experiment was done to evaluate the performance of the NAND based RO. Table 3.1 and Table 3.2 shows how a NAND gate with input B connected to VDD, acts as an inverter.

![Figure 3-2: NOT gate based ring oscillator studied in [35].](image)

![Figure 3-3: Proposed NAND gate based ring oscillator.](image)
Table 3.1: Truth table of a NOT gate with input A and output Y

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.2: Truth table of a NAND gate with input A & B (connects to VDD) and output Y

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

In this work, a ring oscillator network structure composed of seven NAND gate based ROs are placed on the ISCAS’85 c2670 benchmark [63] which is the circuit under authentication (CUA). A linear feedback shift register (LFSR) is used as the test pattern generator for the CUA. A similar test bed using NOT gate based RO is designed for comparison purpose. 10 Xilinx Spartan-3E FPGAs (90nm) are used as the hardware validation platform. Our research analyzes the Trojan’s impact on the NAND based ROs placed in different position across the CUA. It also demonstrates how NAND gate based ROs placed near the Trojan experience higher percentage of change due to additional circuit activity compared to NOT gate based RO.

### 3.2.1 Ring Oscillator Network Structure

One ring oscillator is not enough to observe the whole CUA. Hence, a ring oscillator network (RON) topology was suggested in [45]. Along with the ROs, the RON structure also consists of a counter, multiplexer and a de-multiplexer. During the authentication mode, the multiplexer selects a particular RO and the RO output is carried to the counter using the de-multiplexer. After that, the frequency of each RO can be
calculated from the cycle count obtained from the counter. Similar RON topology is used in this experiment as shown in Figure 3-4.

Figure. 3-4: Proposed NAND gate based ring oscillator network.

To compare the NAND and NOT gate based RO performance, separate identical RON structure using the NAND and NOT based RO is designed. In this work, seven ROs were used considering the size of the CUA. As the aim is to observe the sensitivity of the two RON scheme under Trojan inserted and Trojan free conditions, the area overhead is
not crucial at this point. Even though it should be mentioned that for a five stage RO, NOT based structure consumes 40% less area compared to the NAND based RO.

It should be noted that the RON based Trojan detection solution is developed for ASIC. However, the assessment platform used in this study is the FPGA. There is a noteworthy difference in the implementation of the RON structure in these two different platforms. In case of ASICs, each RO stage can be implemented between the VDD and VSS line. For NAND based RO, this arrangement should provide better sensitivity as one of the input of each NAND gate is connected to the adjacent VDD. But while implementing this RO in the FPGA, similar distribution of RO stages could not be followed. It is due to the lack of flexibility to connect RO stages to desired VDD nets and the lack of information regarding the internal power distribution network for FPGA. So our experiment result might be of lower bound. Extending this experiment for NAND based RO on ASICs should give better results.

3.3 Implementation

3.3.1 Implementation on FPGA

Xilinx Spartan-3E XC3S100E boards are used in this research to implement the NAND and NOT gate based ring oscillator network structure.

3.3.1.1 Architecture of the Xilinx Spartan-3E FPGA

Layout of the Xilinx Spartan-3E FPGA XC3S100 chip is shown in Figure 3-5. This FPGA contains 100,000 gates in 240 configurable logic blocks (CLBs). The CLBs
are distributed across 22 rows and 16 columns as shown in Figure 3-5. This FPGA uses 90 nm technology [48]. Each CLB in this FPGA consists of four identical slices. Thus, a total of 960 slices are available. Each slice comprises of two look up tables (LUTs), flip-flops, and multiplexers. Hence there are eight 4-input LUTs in each CLB along with multiplexers and arithmetic carry logic.

![Figure 3-5: Xilinx Spartan-3E FPGA architecture](image)

### 3.3.1.2 NOT Gate Based RO Implementation

A five stage NOT gate based ring oscillator, similar to Figure 3-2, is implemented on the FPGA by instantiating individual LUTs. Each LUT can accommodate one inverter
stage. Thus, the ring oscillator occupies three slices as shown in red color in Figure 3-6. All the ring oscillators used in the study are identical. This is done by making the hard macro of the ring oscillator. Later, the same hard macro is instantiated for all the ROs of the ring oscillator network. Post-route simulation is done to observe the functionality of the RO. The output of the post-route simulation is shown in Figure 3-7.

Figure 3-6: NOT gate based ring oscillator implementation
3.3.1.3 NAND Gate Based RO Implementation

A five stage NAND gate based ring oscillator, similar to Figure 3-3, is implemented on the FPGA in a similar way. The ring oscillator occupies three slices just like the NOT gate based one. The NAND gate based RO is instantiated with a different reference slice as shown in Figure 3-8. This should not cause any difference in the performance for the two kinds of ROs. While creating the hard macro, the Xilinx FPGA Editor removes all the VCC and GND nets. As the NAND gates of each stages of the inverter are connected to the logic ‘1’, we had to replace the logic’1’ signals with input ports. Later, after making the hard macro, we supply a constant logic ‘1’ to those input ports to activate the RO. The same hard macro is instantiated for all the ROs of the ring oscillator network. The output of the post-route simulation for the NAND gate based RO is shown in Figure 3-9. From Figures 3-6 and 3-8, it can be seen that the NAND gate based RO stages have multiple connections to the power net due to the constant logic ‘1’ fed to each NAND gate. It is difficult to comment, if such connection to the power net
will make the NAND based RO more sensitive, as the platform is FPGA. The information about the internal power distribution network of the FPGA is not available.

Figure 3-8: NAND gate based ring oscillator implementation
In order to compare the effectiveness of the NAND and NOT based RON structures, ISCAS’85 c2670 benchmark is manually placed using LOC (placement) constraint [47] to maintain identical placement of CUA for both NAND and NOT based RON. To observe if the ring oscillator frequency values are consistent even under process variations, 10 Digilent Basys-2 board having Xilinx Spartan-3E FPGA are used.

The CUA c2670 benchmark has 233 inputs and 140 outputs. To map it on the Digilent Basys-2 board a multiplexer is used with the CUA to reduce the number of outputs. As mentioned earlier, hard macro of both the NAND and NOT based RO is designed to get identical placement and routing. Each RO takes three of the four slices of one configurable logic block (CLB). A total of seven hard macros of the RO are instantiated for covering the whole CUA and numbered as RO1 through RO7 as shown in Figure 3-10. The CUA takes 180 CLBs out of the 240 available CLBs on the chip. Due to very tight placement of the benchmark, the LFSR, counter, multiplexer and de-multiplexer are placed separately in the rest of the empty CLBs.

Figure 3-9: Post-route simulation of the NAND gate based ring oscillator
Figure 3-10: Position of the benchmark, ROs and Trojan in the FPGA editor

Inside the CUA eight slices are kept empty for the Trojan. By doing this, the Trojan free version of the CUA is created. The Trojan is later placed manually on those empty slices to obtain the Trojan inserted chip.
3.3.2 Trojan Design

The Trojan that is used is similar to the one studied in [35]. Figure 3-11 shows one stage of the single Trojan that is used. 4 similar stages constitute the Trojan design. These 20 Trojan gates are manually placed in the 10 empty slices that were kept empty in the Trojan Free version of CUA. The first stage of the Trojan obtains input from the LFSR and the rest of the stages are supplied by the output of the previous stage. Such design of Trojan guarantees partial activation during the circuit operation which should impact the nearby ROs. The Trojan also has an Enable signal which can be used to block any transition that might occur due to different inputs from the LFSR.

![Diagram](image)

Figure 3-11: Design of the Trojan stage used

3.4 Measurement Flow

At first, for NAND based RON, several frequency measurements are taken for each of the seven ROs for all the 10 Trojan free FPGAs. The average of the measurements is used to eliminate measurement noise. Similar measurements are repeated for all ROs of all Trojan inserted FPGAs. Similar procedures are repeated for the NOT based RON.
3.5 Experimental Results and Analysis

3.5.1 Trojan Impact Analysis

The impact of the hardware Trojan on a particular RO of an FPGA is analyzed by comparing it with the same reference RO of the Trojan free version on the same FPGA. By comparing with the same RO between Trojan and Trojan free version of FPGA, we are evading the intra die process variation. By evaluating the Trojan and Trojan free CUA on the same FPGA, we are evading inter die process variation. For each RO, the average difference of all Trojan free and Trojan inserted FPGAs is taken to compute the mean percentage change. The mean impact of the Trojan on a particular RO is calculated by using the following equation [35],

\[
TRO_{R_{pj},T} = \left( \frac{1}{10} \right) \sum_{k=1}^{10} \frac{|RO_{j,k,T}^{free} - RO_{j,k,T}^{T}|}{RO_{j,k,T}^{free}}
\]

Here \(TRO_{R_{pj},T}\) is the mean impact of the Trojan on the \(j^{th}\) RO across all FPGAs compared to the Trojan free case. \(RO_{j,k,T}^{free}\) is the Trojan-free frequency of the \(j^{th}\) RO in the \(k^{th}\) FPGA. Using (4) we have calculated the Trojan impact for both NAND and NOT gate based ring oscillators.

The results of the NOT-RON are summarized in Figure 3-12. We can see that the Trojan impact on RO3 and RO5 is higher compared to the rest. Such anomaly in particular ROs is very useful in detecting Trojan using the spatial locality analysis method described in [35]. But the difference is not very high compared to other RO impacts. If the impact is very large or very small for some of the ROs, it can be considered as a sign of malicious inclusion. Similarly, Figure 3-13 shows the percentage...
of mean impact on RO frequency for NAND based RO structure. It is obvious that Trojan impact on RO3, RO5 and RO6 is very high compared to the other ROs of the FPGA. This indicates that ROs placed closer to the Trojan are affected the most if compared to the other ROs which are distant from the Trojan. In Figure 3-14, obtained results from both NAND and NOT based ROs are compared. It can be inferred that NAND based ROs go through a higher percentage of variation compared to the NOT gate based ROs when placed near the Trojan. ROs placed far from NAND based ROs follow the same trend of NOT based ROs with relatively lower percentage of variation. This might be due to the fact that the NAND based RO frequency is more influenced by the nearby logics compared to the NOT gate, which makes it hard for a distant Trojan to effect the frequency.

![Bar chart showing Trojan impact on NOT gate based ring oscillator frequency.](image)

Figure 3-12: Trojan impact on NOT gate based ring oscillator frequency.
After calculating the Trojan’s impact, which is expressed in terms of the percentage of change in frequencies of ROs, the percentage of increase or decrease among these Trojan impacts are calculated and shown in Table 3.3.
Table 3.3: Percentage of increase (positive values) or decrease (negative value) in Trojan impacts between NOT gate based and NAND gate based RO

<table>
<thead>
<tr>
<th>Ring Oscillator</th>
<th>percentage of increase or decrease (from NOT to NAND) in Trojan impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-17.113518919</td>
</tr>
<tr>
<td>2</td>
<td>-5.570567979</td>
</tr>
<tr>
<td>3</td>
<td>36.273787897</td>
</tr>
<tr>
<td>4</td>
<td>-9.051798722</td>
</tr>
<tr>
<td>5</td>
<td>0.018182372</td>
</tr>
<tr>
<td>6</td>
<td>43.528327665</td>
</tr>
<tr>
<td>7</td>
<td>-0.958966508</td>
</tr>
</tbody>
</table>

3.6 Conclusion and Future Work

In this chapter, we have demonstrated the effectiveness of the NAND gate based RO network. The results indicate that the ring oscillators located closer to the Trojan undergo a higher percentage of variation in frequency, compared to the NOT gate based RO. Further research on ASIC based implementation where the input of each of the NANDs of each stage is connected to power strap can be more helpful in demonstrating better results.
Chapter 4

Hardware Trojan Detection using Circuit Partitioning Technique

It has been shown in [35] that using a power based side channel analysis method; it is possible to detect a Trojan even under measurement noise, ambient noise, and process variation. In order to avoid detection, adversaries often design Trojans which have minimal impact on the overall circuit power. For understanding the Trojan’s impact on a circuit’s power fluctuation, following definitions are useful [38]:

- **Trojan to circuit power consumption (TCP):** It is defined as the ratio of the Trojan power consumption to the circuit power consumption.
- **Trojan to circuit switching activity (TCA):** It is defined as the ratio of the number of transitions inside the Trojan circuit to the number of transitions over the entire circuit.

TCA is measured in logic level and TCP is measured in transistor level. Nevertheless, they hold a relation. A higher amount of switching in Trojan gates is one of the causes of
power consumption by the Trojan circuit. Thus, higher TCA is one possible reason for a higher TCP.

4.1 Motivation

In [35], it has been observed that variations to the side channel parameters caused by the Trojan is below the process variation level. This is also true for our study in Chapter 3. Thus, it is important to investigate techniques, which can increase the Trojan’s impact and make it observable in the presence of process and environmental variation, and measurement noise. A technique to magnify the Trojan’s activity by limiting the overall switching activity of the circuit using scan cell re-ordering method has been proposed in [38]. It is very difficult to generate such test patterns which can achieve high switching activity in a target region, while keeping the overall circuit activity low.

In our study, a higher TCP is supposed to cause a higher percentage of change in the frequency of the ring oscillators placed near the Trojan. But, the use of LFSR as the test pattern generator keeps the overall circuit activity very high. Furthermore, in the worst case scenario, where the patterns generated from the LFSR are unable to cause a partial activation inside the Trojan, the TCP is further reduced. Therefore, a technique, which can improve the ratio of the Trojan to circuit power dissipation, is desirable to make the power based monitoring technique more fruitful. In this chapter, we aim to present a technique, that can improve the TCP by reducing the overall power consumption of the circuit under authentication.
A brief review of various kinds of power consumption is presented in this section. The solution presented in this chapter is dedicated for ASICs. As our implementation platform is FPGA, it is also essential to know the differences between these two platforms in terms of the power dissipation. We will briefly discuss these topics in later sections.

4.2 Overview of Power Dissipation in ASICs and FPGAs

4.2.1 Power Dissipation in CMOS

Power dissipation in CMOS circuits consists of two major components [49]:

1. Dynamic power dissipation due to
   a. Charging and discharging load capacitances as gates switch,
   b. “Short-circuit” current while both pMOS and nMOS stacks are partially ON.

2. Static dissipation due to
   a. Subthreshold leakage through OFF transistors,
   b. Tunnelling current through gate oxide,
   c. Leakage current through reverse-biased diodes, and
   d. Contention current in ratioed circuits.

4.2.1.1 Dynamic Power Dissipation due to Switching of Gates

One of the sources of the dynamic power dissipation is charging and discharging of capacitance in the circuit. This capacitance is mostly gate and wire capacitance along
with some drain and source capacitance. To illustrate the dynamic power consumption due to switching, a CMOS inverter example is shown in Figure 4-1.

![CMOS Inverter Diagram](image)

Figure 4-1: Operation of a CMOS inverter: (a) Charging phase, (b) Discharging phase.

In Figure 4-1, the total capacitance $C_L$ represents the parasitic capacitances of the nMOS and pMOS transistors (source and drain-diffusion to bulk), the capacitance related to the internal and external wires of the inverter cell, and the input capacitance (gate to bulk) of the circuits.

In Figure 4-1 (a), the circuit is provided with a logic value ‘0’ as the input. Due to the logic ‘0’ at the input the pMOS is on and nMOS is off. These states create a path from power supply to the load capacitor. The current drawn from the power supply charges the capacitor $C_L$ up to $V_{DD}$. Thus, the output becomes ‘1’. In Figure 4-1 (b), when the input changes to logic ‘0’, the pMOS is off and nMOS is on, creating a path from $C_L$ to ground through which $C_L$ discharges and eventually the output turns to logic ‘0’.
During the charging period, the energy drawn from the power supply is $C_L V_{DD}^2$. Half of this energy is stored in capacitor $C_L$ and other half is dissipated in the parasitic capacitances of pMOS transistor and the interconnect [50]. During the discharge phase, the stored energy in the capacitor $C_L$ dissipates into the nMOS transistor and interconnects. Thus, the dynamic power dissipation for a single rising transition is computed as:

$$P_{dynamic} = C_L V_{DD}^2$$  \hspace{1cm} (4-1)

Thus, if the inverter operates in a frequency $f$, and that the average number of transition per clock cycle at the inverter output is $N$, then Eqn. (4-1) can be rewritten as:

$$P_{dynamic} = C_L V_{DD}^2 N f$$  \hspace{1cm} (4-2)

### 4.2.1.2 Dynamic Power Dissipation due to Short-Circuit Current

Power consumption due to short circuit current occurs during the transition phase. When the input signal changes from ‘0’ to ‘1’ or vice versa, for a brief period of time, both the nMOS and pMOS transistors are in ON state. Thus, there exists a short circuit current from the power supply to the ground which leads to the consumption of power. Short-circuit power is sensitive to $\nu$ which is the ratio of the threshold voltage and the supply voltage. Short-circuit power is usually around 2% and 10% of switching power for $\nu$ of 0.3 and 0.2 respectively [49]. In nanometer processes, short-circuit current has become just about negligible.
4.2.1.3 Static Power Dissipation due to Subthreshold Leakage

At the 90nm technology node, leakage power may constitute about 42% of total power due to low threshold voltage and thin gate oxide [51]. When the gate-to-source voltage is below the threshold voltage, the transistor is in subthreshold region, or weak-inversion region. At this region, subthreshold leakage current occurs due to carrier diffusion between the source and the drain of the MOSFET.

In recent years, supply voltage has been scaled down to reduce the dynamic power consumption and to lower the electric fields inside small devices, which improves device reliability. Subthreshold conduction is controlled by the threshold voltage, which is between ground and the supply voltage. Thus, threshold voltage also has to be reduced along with the supply voltage which offers less gate voltage swing below threshold to turn the device off. Furthermore, subthreshold conduction varies exponentially with gate voltage. Therefore, it is becoming more significant with shrinking size of MOSFET [53]. Leakage can exceed 50% of total power consumption for certain technology nodes with threshold voltage of 0.2 V [52].

4.2.1.4 Static Power Dissipation due to Tunnelling Current

Due to very small thickness level, insulation (gate oxide) has become weak, which facilitates tunneling of electrons across the thin insulation. This is particularly important for transistors below 130 nm technology which have gate oxide thickness of 20 Å or less [55]
4.2.1.5 Static Power Dissipation due to Contention Current

Contention current exists only in certain alternative circuits (e.g., pseudo-nMOS gates). Conventional CMOS circuits have no contention current in static mode [49]. Thus, we do not discuss static power consumption due to contention current.

4.2.1.6 Static Power Dissipation due to Leakage Current through Reverse-Biased Diodes

The P-N junctions between diffusion, substrate and well are all junction diodes as shown in Figure 4-2. These junctions are reversed biased as the substrate is connected to GND and the well is connected to $V_{DD}$. Small reverse-bias leakage currents are formed due to formation of this reverse bias. However, in current technology, this leakage current is very small compared to sub threshold and tunnelling currents, so these may be neglected.

Figure 4-2: Formation of reverse-biased diodes in CMOS [54]
4.2.2 Power Consumption in FPGAs

Due to availability, our implementation platform is an FPGA, though the proposed Trojan detection methodology is specifically designed for ASICs. Hence, we discuss the difference between the ASIC and the FPGA in terms of power consumption.

4.2.2.1 Dynamic Power Consumption in FPGA

FPGAs provide flexibility and re-programmability which come at the cost of higher power consumption compared to the ASIC. If we compare in terms of logic, FPGAs consume 12 times more dynamic power than ASICs. There are many other components inside the FPGA other than the logic, such as DSP and Memory. These components are also found to consume nine times greater power, when implemented on FPGAs rather than ASICs [60]. FPGA interconnects are one of the main reasons for higher dynamic power consumption [56]. The large interconnect power inherit from large loads. To facilitate programmability feature to interconnect of FPGAs; interconnect structure with significantly larger loading than ASICs is required.

Components such as signal buffers, pass transistors and other programmable switching structures increase the capacitive load of signal nets significantly compared to dedicated metal wires used in ASICs. It has been mentioned in Section 4.2.1.1 that, capacitive switching is one of the key reasons of dynamic power consumption. Therefore, it is clear how FPGAs consume a greater amount of dynamic power. Besides power consumption, amount of delay in interconnects is also high due to this large capacitive load.
Another reason contributing to higher dynamic power consumption in FPGAs is the extra wire-length. A significant portion of an FPGA’s total area is consists of SRAM configuration cells and circuitry. For instance, according to [57], more than 40% of an FPGA’s logic block area is occupied by SRAM configuration cells. Due to this area overhead, wire-lengths in FPGAs are longer than wire-lengths in ASICs. Therefore, a higher capacitive load in the FPGA interconnects is introduced, making it the primary source of the dynamic power consumption.

4.2.2.2 Static Power Consumption in FPGA

A design implemented on an FPGA uses only a portion of the FPGA. In case of ASICs, there is no such unused portion. However, during operation, static power is constantly drawn from transistors occupying the whole FPGA, irrespective of whether they are used in the design. Thus FPGAs consume relatively more static power than ASICs. One study regarding leakage power analysis of 90nm FPGA shows that, even if 100% of all the CLBs are used, 35% of leakage power comes from unused interconnect which are required to provide programmability to the switches [58]. Thus, for a smaller design on FPGA with large number of unused CLBs, the share of leakage power from the unused portion should be higher.

4.3 Proposed Methodology

A circuit partitioning technique has been used to improve reliability of the circuit by reducing the peak power and the average power consumption during built-in self-test (BIST) [9]. In this chapter, we investigate a method which reduces the overall switching
activity of the circuit by activating a portion of it during verification, and isolating the rest. The original circuit is partitioned into two different structural sub circuits of nearly equal area. Thus, each sub circuit is tested for Trojans in two different sessions. Circuit partitioning comes with an area overhead, but this can be minimized by keeping the number of connections between the sub circuits low. Similar approach was first used in pseudo-exhaustive testing, where test length is minimized by reducing the number of input signals through circuit partitioning [59]. The partitioning scheme is shown in Figure 4-3. A circuit is partitioned into two sub circuits $C_1$ and $C_2$ as shown in Figure 4-3 (a). There are many different partitioning techniques available for VLSI systems. Most partitioning algorithms divide the circuit in a way that the sub circuits have some common primary inputs and some embedded inputs going from one sub circuit to another. After partitioning the circuit using an algorithm, multiplexers are inserted, which allow us to isolate $C_1$ from $C_2$ during the authentication process as depicted in Figure 4-3 (b). De-multiplexers are used to distribute the common primary inputs to both sub circuits.

![Figure 4-3: The circuit partitioning technique [9]](image)
The embedded inputs D and E are multiplexed with A’ and C’ respectively. A’ and C’ indicate subsets of the primary input A and C. During the authentication mode, A’ and C’ are chosen over D and E, as the sub circuits will not exchange any signals from one another. Figure 4-3 (c) shows how the multiplexers and the de-multiplexers can be used to test $C_1$ while isolating $C_2$ [9], [59].

To demonstrate the effectiveness of the circuit partitioning approach, ring oscillator based power monitoring technique is used in this work. However, other power based side channel analysis can benefit, if integrated with this method. During the authentication mode, the sub circuit which is under authentication receives the test patterns from the LFSR. Conversely, the other sub circuit does not receive any switching pattern either from the LFSR or from the embedded inputs coming from the other sub circuit. Both sub circuits will have ring oscillators distributed across the circuit area and all of them activate regardless of which sub circuit is under authentication. The measurement process is similar to the one in Chapter 3. While testing each sub circuit, we consider the frequency values from all the ROs. In our experiment, we also observe RO values by keeping the sub circuits together to compare the Trojan’s impact for both with and without partitioning. This allows us to compare the results.

### 4.4 Circuit Partitioning Technique

In this section, circuit partitioning technique which can be used to obtain the sub circuits is discussed. It should be noted that, in our implementation we have eliminated the need of a partitioning algorithm by using the same benchmark as the two sub circuits
and then connecting them with multiplexers and de-multiplexers. This emulates the equivalent behavior of a circuit partitioned using an algorithm.

The low power BIST technique in [9] uses the partition tool, named pMETIS. This tool has been developed at the University of Minnesota [61]. The tool implements a multilevel hypergraph partition algorithm, where the quality of the partitioning is measured in terms of the cut size and computation time required. This tool is free and available for download at the University of Minnesota website. In [61] the author of pMETIS reports that their algorithm usually produces 6% to 23% better partition than existing algorithms, in terms of the cut size. Furthermore, it takes 4 to 10 times less time to compute, compared to other algorithms available. Table 4.1 shows the area overhead required to include the MUXs to implement the low power BIST, which is similar to our implementation for Trojan detection [9].

Table 4.1: Area overhead for inserting MUXs and DE-MUXs for low power BIST [9]

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of gates</th>
<th>Cut size</th>
<th># of add. gates</th>
<th>Area overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2670</td>
<td>1145</td>
<td>47</td>
<td>94</td>
<td>8.2%</td>
</tr>
<tr>
<td>c3540</td>
<td>1653</td>
<td>81</td>
<td>162</td>
<td>9.8%</td>
</tr>
<tr>
<td>c5315</td>
<td>2502.5</td>
<td>85</td>
<td>170</td>
<td>6.8%</td>
</tr>
<tr>
<td>c7552</td>
<td>3315</td>
<td>51</td>
<td>102</td>
<td>3.1%</td>
</tr>
<tr>
<td>s420</td>
<td>230</td>
<td>8</td>
<td>16</td>
<td>6.9%</td>
</tr>
<tr>
<td>s1423</td>
<td>749</td>
<td>27</td>
<td>54</td>
<td>7.2%</td>
</tr>
<tr>
<td>s9234</td>
<td>4678.5</td>
<td>81</td>
<td>162</td>
<td>3.5%</td>
</tr>
<tr>
<td>s13207</td>
<td>6395.5</td>
<td>57</td>
<td>114</td>
<td>1.7%</td>
</tr>
<tr>
<td>s15850</td>
<td>7987</td>
<td>82</td>
<td>164</td>
<td>2.0%</td>
</tr>
<tr>
<td>s35001</td>
<td>8999.5</td>
<td>41</td>
<td>82</td>
<td>0.4%</td>
</tr>
<tr>
<td>s38417</td>
<td>18204</td>
<td>61</td>
<td>122</td>
<td>0.7%</td>
</tr>
<tr>
<td>s38584</td>
<td>20446.5</td>
<td>132</td>
<td>264</td>
<td>1.3%</td>
</tr>
</tbody>
</table>

From the table it can be seen that the area overhead is very small for large benchmarks like s35001 and s38584. For Trojan detection using the RO based technique
another set of multiplexers is required at the primary inputs if the LFSR is embedded inside the IC. pMETIS also allows users to edge-cut on lines (i.e., MUX insertion) which belong to critical paths. This is achieved by setting a lower value to the probability to be cut prior running the algorithm. Even though partitioning of the circuit into more than two partitions is possible with this algorithm, higher number of partitions will lead to larger overhead and verification time. Moreover, a Trojan’s affect distributed across multiple sub circuits may lead to obfuscation of the Trojan.

4.5 Experimental Setup

4.5.1 Implementation Platform

In this experiment, Spartan-3E (XC3S100E) FPGA is chosen as the implementation platform. At first, we tried to implement the system on Artix-7 XCA100T FPGA. But this chip contains 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops [62]. To cover the whole FPGA, a very large benchmark is required. Even though there are many large benchmarks available, we had to manually place the benchmark on the FPGA. This is essential because the two subsections should be placed into distinct places on the FPGA. Therefore, switching in one sub circuit does not affect the other one, during authentication. Due to this requirement, such a big benchmark was not used. Furthermore, if we place a small benchmark and a proportionally smaller Trojan on large capacity FPGA, the huge static power of the FPGA covers the small variations in power consumption caused by the small Trojan. Even with moderate utilization of the whole FPGA fabric around one third of the static power comes from the
programmable interconnects over which; the design placed on the FPGA has no contribution [58].

4.5.2 Benchmark Used

We chose the ISCAS’85 c1355 benchmark as a sub circuit, which has 560 gates. The same benchmark is mapped in left and right portions of the FPGA together. Figure 4-4 shows the benchmark placed on the left half of the FPGA. It is named as ‘Sub Circuit 1’ or P1.

![Figure 4-4: Placement and routing of Sub Circuit 1(P1), mapped on FPGA](image)
Figure 4-5 shows the same benchmark placed on the right half of the FPGA. It is designated as ‘Sub Circuit 2’ or P2. The two sub circuits placed in separate portions of the same FPGA are connected with the MUXs according to the method shown in Figure 4-3 (b). Number of inputs of P2 is less than the number of outputs of P1. Therefore, the number of outputs of P1 has been reduced and made equal to the number of inputs of P2 by connecting the outputs to additional gates. Similarly, number of outputs of P2 has been reduced, as it exceeds the number of available output ports on the FPGA board.

When choosing the benchmark, capacity of the FPGA has to be considered. The intended platform, Spartan-3E (XC3S100E) can accommodate 100,000 gates. However,
when the design is mapped manually on to the FPGA, each LUT only takes one logic gate. Thus, 960 slices of the Spartan-3E only take a total of 1920 gates. Besides the benchmark, space for RO, frequency counter, RO selector MUX, partitioning MUX, and LFSR is required.

4.5.3 Ring Oscillator Network

Considering the conclusion drawn from Chapter 3, NAND gate based ring oscillators are used. A ring oscillator network consisting 10 ROs is deployed as shown in Figure 4-6.

![Figure 4-6: Ring oscillators (marked in red) and the Trojans](image)
4.5.4 Trojan Circuit

Figure 4-7 shows one of the blocks of Trojans that we have used. Two different Trojans have been used in the experiment. The basic building block of the Trojan block is the NAND based ring oscillator itself. For designing a single Trojan, multiple number of this single block has been used. The Trojans are named Trojan1 and Trojan2. Trojan1 contains three blocks, thus it has 15 gates, and Trojan2 contains seven blocks, therefore it has 35 gates.

![Diagram of Trojan circuit](image)

Figure 4-7: Single block of the Trojan circuit

This design of Trojan is used to make sure that the Trojan experiences adequate signal transitions throughout the experiment. A Trojan design, which experiences few transitions, might not cause significant changes to the power monitor’s output, because of the implementation platform, which is a FPGA. However, routing delay is introduced between the inverter stages of the Trojan. Thus, number of signal transitions per clock cycle is less for the Trojan block, compared to the standard RO used as a power monitor. Even though the experiment is done using Trojans with high number of transitions, conclusive results will motivate researchers to continue this experiment on the ASIC platform.

We have mapped the Trojans along with the Trojan free benchmark and used the ‘Enable’ signal to turn the Trojan on and off. Therefore, the ‘Enable’ signal can switch
the FPGA from the Trojan free to the Trojan inserted FPGA. Using the above mentioned mechanism to create a Trojan free and Trojan inserted chip, identical placement and routing between Trojan free and Trojan inserted FPGAs is guaranteed.

4.5.5 Frequency Counter and RO Selector

The frequency counter receives the signal from the ring oscillator’s outputs. It also incorporates multiplexer and de-multiplexer to select a certain RO’s output to record. The design is similar to the one used in Chapter 3. The placement is shown in Figure 4-8.

![Figure 4-8: Frequency counter and RO selector (marked in red)](image)

Placement of this network is particularly important, because it will be activated irrespective of the sub circuit. Even if we stop the switching activities of the sub circuit
containing the counter, the counter has to be activated and this activity has to be considered when analyzing the data.

4.6 Measurement Flow

According to our method, when an IC will return from the foundry, for each sub circuit, ring oscillator’s frequency values will be obtained individually keeping the other sub circuits silent. Later, ring oscillator frequency values will be recorded while keeping all the sub circuits active simultaneously. Thus, if the CUA contains two sub circuits, data from RO network will be collected in following three phases: 1) Keeping both the sub circuits active, 2) Keeping only Sub Circuit 1 active, and 3) Keeping only Sub Circuit 2 active. After collecting the data, the data is analyzed to find any anomaly. If there is a Trojan in one of the partitions, and if the Trojan experiences partial activation, then there will be some discrepancy in the RO’s frequency values. If the Trojan free sub circuit is switched off and the Trojan inserted sub circuit is switched on, then this discrepancy will be magnified due to the reduction of the overall circuit activity. It is technically not possible that the Trojan will receive partial activation, when the host sub circuit is kept silent. In this work, there are two sub circuits, Sub Circuit 1 or P1, and Sub Circuit 2 or P2. At first, data is taken when both P1 and P2 are activated as a combined CUA. In this mode of operation, ring oscillator frequency values are collected in three ways. First, by keeping the Trojan disabled, which is the Trojan free state. After that, Trojan1 is enabled and data is taken. Similar procedure is followed for Trojan2. The next stage of data collection is done by turning off one of the sub circuits.
Since it is known that both the Trojans are placed in P2, the data is collected while keeping P1 silent. However, in a real environment, data will be collected in the ideal way, as mentioned earlier in this section.

4.7 Results and Analysis

The procedure of calculating Trojan’s impact is similar to the one in Section 3.5.1. 10 ring oscillators are used and data is taken from 5 FPGAs. Trojan impact is calculated for both Trojan1 and Trojan2 in two different modes. In the first mode, both P1 and P2 are on. This is the normal mode of operation for a circuit. The percentage of changes in frequencies of ROs due to the switching activity of Trojan1 is plotted in Figure 4-9. In the second mode, P1 is detached from operation, and only P2 receives input patterns from the LFSR. Trojan1 resides in P2. Due to the reduced overall circuit activity, a rise in the Trojan’s impact can be seen over all the RO’s frequencies in Figure 4-9. For some of the RO’s, the change is almost doubled. It should be noted that RO5 is located far from Trojan1, compared to RO7. Still, during the partitioned mode, it experiences slightly higher percentage of change in frequency than RO7. One possible reason is that RO5 is exposed to less circuit activity during the time P1 is off. At the same time RO7 is exposed to circuit activity, as P2 is still receiving test patterns. Therefore, the noise (voltage drop) by Trojan1 caused higher amount of change to the frequency of RO5. Similar, RO1, RO2, and RO3 are located further from Trojan1 compared to RO8, RO9, and RO10 but the first three experience lesser change during the normal mode.
However, when P1 is turned off, these three are exposed to slightly higher percentage of change in frequency compared to the closer ones.

Figure 4-9: Comparison of Trojan1’s impact over frequencies of all the ROs, with (Red) and without (Blue) partitioning.

Trojan2’s impacts over the frequencies of the ROs are plotted on Figure 4-10. It is obvious that Trojan2 will cause a greater fraction of change in frequency compared to Trojan1, due to the larger size. This has been held true for both modes of operation. However, it should be noted that the difference between the two percentages of change has increased. It can be concluded that, with increased Trojan size, the partitioning mode will become more effective. All other trends in the graph are similar to Trojan1.
Figure 4-10: Comparison of Trojan2’s impact over frequencies of all the ROs, with (Red) and without (Blue) partitioning.

The increase in percentage of variation in frequencies of ROs after partitioning is summarized in Table 4.2.

Table 4.2: Increase in the percentage of variation after partitioning in terms of ratio

<table>
<thead>
<tr>
<th>Ring Oscillator</th>
<th>ratio of the % of variation with partitioning to without partitioning for Trojan1</th>
<th>ratio of the % of variation with partitioning to without partitioning for Trojan2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.74</td>
<td>1.75</td>
</tr>
<tr>
<td>2</td>
<td>1.65</td>
<td>1.85</td>
</tr>
<tr>
<td>3</td>
<td>1.82</td>
<td>2.09</td>
</tr>
<tr>
<td>4</td>
<td>1.93</td>
<td>2.1</td>
</tr>
<tr>
<td>5</td>
<td>1.83</td>
<td>2.09</td>
</tr>
<tr>
<td>6</td>
<td>1.82</td>
<td>2.1</td>
</tr>
<tr>
<td>7</td>
<td>1.8</td>
<td>2.02</td>
</tr>
<tr>
<td>8</td>
<td>1.8</td>
<td>2.22</td>
</tr>
<tr>
<td>9</td>
<td>1.5</td>
<td>1.87</td>
</tr>
<tr>
<td>10</td>
<td>1.54</td>
<td>1.77</td>
</tr>
</tbody>
</table>
4.8 Summary

Our proposed partition technique has been proven to provide improvement over the ring oscillator based technique. The results are still below the process variation, but ASIC implementation of our technique should provide better results. Other power based side channel analysis methods can also benefit from this partitioning technique.
Chapter 5

Conclusion and Future Work

In this work, one of the most recent and intricate problems faced by the semiconductor industry is addressed. There is a threat to the industry that the trustworthiness of integrated circuits deployed in military, industry, and even in biomedical sectors may be compromised. The complexity of the problem has increased with the discovery of hardware Trojans that are undetectable even by the latest technologies [12]. The lack of trust arises due to outsourcing of the chip fabrication process to untrusted third parties, which are not under surveillance. The industry is moving towards finding a comprehensive solution to this problem.

In this work, a side channel analysis based Trojan detection technique is implemented using the NAND based ring oscillators. A ring oscillator network consisting of 7 ROs is mapped along with the ISCAS’85 c2670 benchmark on 10 Xilinx Spartan-3E FPGAs. A Trojan is inserted to create the Trojan inserted version of the benchmark. Frequency values of the ring oscillators are obtained during the authentication process from ring oscillator networks of both Trojan free and Trojan inserted benchmarks placed
on FPGAs. Similar procedure is followed to collect data using the NOT gate based ring oscillator network. Later, frequencies from both NAND and NOT gate based ring oscillator networks are analyzed. The results demonstrate that the impact of Trojans on the frequency of nearby ROs is larger for NAND based structure compared to the NOT gate based design.

A technique is also proposed in this work to improve the ring oscillator based Trojan detection. A circuit partitioning technique is used to improve the ratio of the power consumed by the Trojan to the power consumed by the host circuit. This ratio plays an important role in detection of Trojans using power based side channel analysis method. By creating two sub circuits, switching activity of individual sub circuits are controlled during the authentication process. A test bed is created by implementing two sub circuits which constitute the circuit under authentication. NAND gate based ring oscillator network is integrated with the CUA. Trojan free and Trojan inserted version of the CUA are created and implemented on 5 Xilinx Spartan-3E FPGAs. Ring oscillator frequency values are obtained while keeping one of the sub circuits silent, and compared with the data collected while both the sub circuits are active. The results show a higher percentage of change in the ring oscillator’s frequencies when the CUA is partially activated, which is helpful in detecting the Trojan.

### 5.1 Contributions

The major contributions of this research are summarized below:
1) A five-stage NAND gate based ring oscillator is designed. The design is implemented as a Look-Up-Table based synchronous RO on Spartan-3E FPGAs.

2) Using the hard macro of the ROs, a ring oscillator network is designed and implemented on FPGAs. The network constitutes multiplexer, de-multiplexer, and a counter to collect frequency values of individual ROs during the authentication period.

3) The Verilog code of the ISCAS’85 c2670 benchmark is converted for Look-Up-Table based implementation on the Spartan-3E FPGA. Identical placement of the benchmark is maintained between NAND and NOT gate based ring oscillator network using this technique.

4) The percentage of change in RO frequencies due to the noise (voltage drop) caused by the Trojan’s switching activity is calculated to compare the effectiveness between the NAND gate based ring oscillator and NOT gate based ring oscillator. For the two ring oscillators located near the Trojan, percentage of change has improved by 43.5% and 36.2% for NAND gate based RO.

5) A circuit partitioning based approach is proposed to magnify the Trojan’s activity limiting the switching activity of the circuit under authentication.

6) Look-Up-Table based implementation of the ISCAS’85 c1355 benchmark is done by translating the Verilog code of the benchmark.

7) The percentage of variation in RO frequencies due to switching activities of Trojan1 and Trojan2 is calculated. The average increase in the
percentage of variation over all the RO’s frequencies is found to be 1.74 times for the small Trojan and 1.98 times for the large Trojan.

5.2 Future Work

For further research, following works are suggested:

- Implement the NAND gate based ring oscillator architecture on AISCs. In ASICs, this structure can be implemented by keeping one stage of the NAND based RO between each VSS and VDD line. Similar implementation was done with the NOT gate based RO in the past. However, a VDD line connects to each stage of the NAND gate based RO, which makes it more susceptible to voltage drops due to gates sharing the same VDD line.

- Use the partitioning technique with other side channel measurements using ASICs, as the high amount of static power consumption on FPGA makes it difficult to observe the true performance of such techniques.
References


15. Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations Mordechai Guri, Matan Monitz, Yisroel Mirski, Yuval Elovici


34. F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, “Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation,” in Proc. 22nd USENIX Con


50. “SOURCES OF POWER DISSIPATION IN CMOS CIRCUITS” [online] Available:

http://www.it.uom.gr/teaching/embedded/material/support.inf.uth.gr_courses_CE536/Related_Documents/3%20Synthesis,%20estimation%20and%20power%20optimization%20of%20embedded%20systems/Low_power_circuit_design.pdf

52. Kaushik Roy, Kiat Seng Yeo (2004). Low Voltage, Low Power VLSI Subsystems. McGraw-Hill Professional. Fig. 2.1, p. 44.


Appendix A

Source Codes

A.1 VHDL Code for a Five-Stage NAND Gate based Ring Oscillator

library IEEE;
Library UNISIM;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use UNISIM.vcomponents.all;

entity NAND_RO is
port ( input : in std_logic;
      VDD_in: in std_logic;
      output : out std_logic
architecture Behavioral of NAND_RO is
signal s1, s2, s3, s4, s5: std_logic;
attribute keep: boolean;
attribute keep of s1, s2, s3, s4, s5: signal is true;
attribute loc : string;
attribute loc of LUT1_inst: label is "SLICE_X45Y105";
attribute loc of LUT2_inst: label is "SLICE_X45Y105";
attribute loc of LUT3_inst: label is "SLICE_X45Y105";
attribute loc of LUT4_inst: label is "SLICE_X45Y105";
attribute loc of LUT5_inst: label is "SLICE_X44Y105";
begn

LUT1_inst : LUT2

    generic map (INIT => X"7")
    port map (   
        O => s1, -- LUT general output
        I0 => input, -- LUT input
        I1 => s5 -- LUT input);

LUT2_inst : LUT2

    generic map (INIT => X"7")
    port map (   
        O => s2,
I0 => s1,
I1 => VDD_in);

LUT3_inst : LUT2
generic map (INIT => X"7")
port map (O => s3,
        I0 => s2 ,
        I1 => VDD_in );

LUT4_inst : LUT2
generic map (INIT => X"7")
port map (O => s4,
        I0 => s3 ,
        I1 => VDD_in);

LUT5_inst : LUT2
generic map (INIT => X"7")
port map (O => s5,
        I0 => s4 ,
        I1 => VDD_in);

output <= s5;
end Behavioral;
A.2    VHDL Code for a Five-Stage NOT Gate based Ring Oscillator

library IEEE;
Library UNISIM;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use UNISIM.vcomponents.all;
entity NOT_RO is
port( input : in std_logic;
            output : out std_logic
                   );
end NOT_RO;
architecture Behavioral of NOT_RO is
signal s1,s2,s3,s4,s5: std_logic;
attribute keep: boolean;
attribute keep of s1, s2, s3, s4, s5: signal is true;
attribute loc : string;
attribute loc of LUT1_inst: label is "SLICE_X1Y43";
attribute loc of LUT2_inst: label is "SLICE_X1Y43";
attribute loc of LUT3_inst: label is "SLICE_X1Y42"
attribute loc of LUT4_inst: label is "SLICE_X1Y42"
attribute loc of LUT5_inst: label is "SLICE_X0Y43"

begin

LUT1_inst : LUT2

    generic map (INIT => X"7")

    port map ( 
        O => s1, -- LUT general output
        I0 => input, -- LUT input
        I1 => s5 -- LUT input);

LUT2_inst : LUT1

    generic map (INIT => "01")

    port map ( 
        O => s2, -- LUT local output
        I0 => s1 -- LUT input);

LUT3_inst : LUT1

    generic map (INIT => "01")

    port map ( 
        O => s3, -- LUT local output
        I0 => s2 -- LUT input);

LUT4_inst : LUT1

    generic map (INIT => "01")

     85
port map (  
    O => s4, -- LUT local output  
    I0 => s3  -- LUT input);  

LUT5_inst : LUT1  
    generic map (INIT => "01")  
    port map (  
      O => s5, -- LUT local output  
      I0 => s4  -- LUT input);  

    output <= s5;  
end Behavioral;  

A.3 VHDL Code for LUT Instantiation to Implementing the Benchmark

---------------------Buffer----------------------
LUT_Inst_Buffer :LUT1  
    generic map (INIT=> X"10" )  
    port map ( O=> Out1 ,  
                I0=> In1);  

------------------NOT gate----------------------
LUT_Inst_NOT :LUT1  
    generic map (INIT=> X"01" )
port map (O=> Out1, I0=> In1);

----------------------2 input AND gate-----------------------
LUT_Inst_AND2 :LUT2
generic map (INIT=> X"8" )
port map (O=> Out1, I0=> In1, I1=> In2);

----------------------2 input OR gate-----------------------
LUT_Inst_OR2 :LUT2
generic map (INIT=> X"E" )
port map (O=> Out1, I0=> In1, I1=> In2);

----------------------2 input NAND gate----------------------
LUT_Inst_NAND2 :LUT2
generic map (INIT=> X"7" )
port map (O=> Out1, I0=> In1, I1=> In2);

----------------------2 input NOR gate----------------------
LUT_Inst_NOR2 :LUT2
generic map (INIT=> X"1" )
port map (O=> Out1, I0=> In1, I1=> In2);

----------------------3 input AND gate----------------------
LUT_Inst_AND3 : LUT3

    generic map (INIT=> X"80" )
    port map ( O=> Out1 ,
               I0=> In1, I1=> In2, I2=> In3);

----------------------3 input OR gate-----------------------

LUT_Inst_OR3 : LUT3

    generic map (INIT=> X"FE" )
    port map ( O=> Out1 ,
               I0=> In1, I1=> In2, I2=> In3);

----------------------3 input NAND gate-----------------------

LUT_Inst_NAND3 : LUT3

    generic map (INIT=> X"7F" )
    port map ( O=> Out1 ,
               I0=> In1, I1=> In2, I2=> In3);

    port map ( O=> Out1 ,
               I0=> In1, I1=> In2, I2=> In3);

----------------------3 input NOR gate-----------------------

LUT_Inst_NOR3 : LUT3

    generic map (INIT=> X"01" )
    port map ( O=> Out1 ,
               I0=> In1, I1=> In2, I2=> In3);

    port map ( O=> Out1 ,
               I0=> In1, I1=> In2, I2=> In3);
----------------------4 input AND gate-----------------------

LUT_Inst_AND4 : LUT4

generic map (INIT=> X"8000" )

port map ( O=> Out1 ,
           I0=> In1, I1=> In2, I2=> In3, I3=> In4);

----------------------4 input OR gate-----------------------

LUT_Inst_OR4 : LUT4

generic map (INIT=> X"FFFE" )

port map ( O=> Out1 ,
           I0=> In1, I1=> In2, I2=> In3, I3=> In4);

----------------------4 input NAND gate----------------------

LUT_Inst_NAND4 : LUT4

generic map (INIT=> X"777F" )

port map ( O=> Out1 ,
           I0=> In1, I1=> In2, I2=> In3, I3=> In4);

----------------------4 input NOR gate-----------------------

LUT_Inst_NOR4 : LUT4

generic map (INIT=> X"0001" )

port map ( O=> Out1 ,
           I0=> In1, I1=> In2, I2=> In3, I3=> In4);