HARDWARE TROJAN INSERTION AND DETECTION USING SIDE CHANNEL ANALYSIS AND IMPLEMENTATION OF BPUF

A Project report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

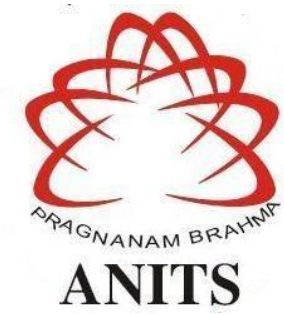
E. Yaswanth Raj (317126512018) **Y. Malathi (317126512060)**

Ch SVSS Upendra (318126512L05) **R. Saroja Sneha (318126512L11)**

Under the guidance of

Mr. P. Devi Pradeep

Assistant Professor



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(*Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade*)

Sangivalasa, Bheemili Mandal, Visakhapatnam Dist. (A.P)-531162

2020-2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)
Sangivalasa, Bheemili Mandal, Visakhapatnam dist. (A.P)-531162

PRAGNANAM BRAHMA

ANITS

CERTIFICATE

This is to certify that the project report entitled **" Hardware Trojan Insertion And Detection Using Side Channel Analysis And Implementation Of BPUF "** submitted by **E. Yaswanth Raj (317126512018), Y. Malathi (317126512060) ,Ch SVSS Upendra (318126512L05), R. Saroja Sneha (318126512L11)** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** of Andhra University, Visakhapatnam is a record of bona fide work carried out under my guidance and supervision.

P. Devi pradeep

Project Guide
Mr. P. Devi Pradeep
Asst. Professor
Department of E.C.E
ANITS

Assistant Professor
Department of E.C.E
Anil Neerukonda
Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

Head of the Department
Dr. V. Rajyalakshmi
Department of E.C.E
ANITS

Head of the Department
Department of E C E
Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Mr. P. Devi Pradeep Asst. Professor**, Department of Electronics and Communication Engineering, ANITS, for his guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. V. Rajyalakshmi**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa,** for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project duration. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT BATCH STUDENTS
E. Yaswanth Raj(317126512018)
Y. Malathi (317126512060)
Ch SVSS Upendra (318126512L05)
R. Saroja Sneha (318126512L11)

ABSTRACT

Hardware based security is meant by carrying out of reliability protection in ICs rather than the logic installed on the IC. Many ICs are produced by some manufacturers outside the network. Under such cases there are chances of introducing the trojan in hardware system causing the fatal damage. Security of any system has been related to the security of the information being processed. The hardware used for information processing has been considered trusted. This information should not get interfered by the third parties. Trojans can be employed by cyber-thieves and hackers trying to gain access to user's systems. A Hardware Trojan (HT) is a malicious modification of the circuitry of an integrated circuit. A hardware trojan is completely characterized by its physical representation and its behaviour. Unlike software viruses and software trojans, hardware trojans cannot be easily eliminated through firmware by updating, therefore are more harmful to working systems.

By and large, the function which can't be duplicated or recovered is called as unclonable function. Instead of storing secrets in digital memory, PUFs derive a secret from the physical characteristics of the integrated circuit (IC). So, attacker won't be permitted to deliver duplicate of the IC/device even with actual access. PUFs can deliver different responses for different difficulties at different occasions. Butterfly PUF is a cross-coupled bistable circuit in which the circuit can be forced to unsteady state before it settles in any of the possible steady states.

This project shows a solution to prevent hardware trojans by detecting them using side channel analysis and also the implementation of Butterfly PUF by using a simulation software called Vivado Xilinx 2019. Hardware trojan detection methods try to detect the existence of a trojan which is having a high probability by studying output waveforms and the power side channel. However, these methods mainly depend on comparing the un-trusted chip with a trusted chip. This project is implemented using side channel analysis as it is a compromise of all other methods in parameters like time taken, success rate, infrastructure needed, implementation ability, coverage scope. In addition to these this method has high performance. Also, in this project we had implemented Butterfly PUF which can be used to in circuits in order to have a unique signature avoiding the problem of copying the original circuit. we had seen the hardware trojan scientific classification, showing of combinational circuit trojan, and thus most natural strategy for trojan recognition i.e., side channel analysis. Finally, we discussed about the PUF and its implementation. At long last, we examined about the PUF and its execution. PUF which has the singular to extricate the interesting mark from the devices.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

HT	Hardware Trojan
HTT	Hardware Trojan Threat
FPGA	Field Programmable Gate Array
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
CLB	Configurable Logic Blocks
EMB	Embedded Memory Blocks
HDL	Hardware Description Language
PUF	Physically Unclonable Function
BPUF	Butterfly Physically Unclonable Function
RO	Ring Oscillator
CUT	Circuit Under Test
IP	Intellectual Property
EDA	Electronic Design Automation

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Nowadays, hardware trojan protection became a hot topic especially after the horizontal silicon industry business model. Third party IP is the building block of many critical systems and that arise a question of confidentiality and reliability of these blocks. In this work, we present novel methods for system protection and Trojan detection that alleviate the need for a golden chip. In this project side-channel analysis-based approach were used to detect Hardware Trojans. Also, Butterfly and Ring Oscillator PUF's has been implemented. Dynamic Trojan detection is done using multiple variant voting. Different methodologies can be implemented for different trojans.

1.1 Project Objective

The main objective of the project is to insert a hardware trojan and detect it using various methods and thereby compare which type of trojan is detected efficiently by which type of methods. The hardware trojan is nothing but a malicious circuit which is placed in the integrated chip without effecting its normal functionality. All these detection methods are followed by a flow of process specified as an algorithm in chapter 5. Every time the trojan inserted circuit parameters are compared with the golden circuit and then judge if trojan is present or not. Butterfly PUF and Ring oscillator PUF were also been implemented through simulation and results had been obtained.

Simulation results are obtained for different circuits which are in Chapter 5.

1.2 Background

Malicious modifications of integrated circuits, referred to as Hardware Trojans, have emerged as a major security threat due to widespread outsourcing of IC manufacturing to un-trusted foundries. An adversary can potentially tamper with a design in these fabrication facilities by inserting malicious circuitry, leading to potentially catastrophic malfunctions in security-critical application domains, such as the military, government, communications, space, and medicine. Conventional post-manufacturing testing, test generation algorithms, and test coverage metrics often fail to detect Hardware Trojans due to their diversity, complexity, and rare triggering conditions. An intelligent adversary can design a Trojan to only trigger under very rare conditions on an internal node, which is unlikely to arise during post-manufacturing test, but can be triggered during long hours of in-field operation.

The detection of Trojans by employing side-channel parameters, such as power trace or delay overhead, is limited due to the large process variations in nano scale IC technologies, detection sensitivities of small Trojans, and measurement noise. Often these issues mask the effect of Trojan circuits, especially for ultra-small Trojans. From an adversary's perspective, the desired features for a successful Trojan are as follows:

rarely activated to evade logic-based testing, low overhead to evade side-channel based detection approach, and low side-channel signature to evade Design for Security (DfS) hardening mechanisms. The condition of Trojan activation is referred to as the trigger, and the node affected by the Trojan is referred to as its payload. Trojans can be classified based on their triggering conditions or payload mechanisms. The trigger mechanism can be either digital or analog. Digitally triggered Trojans can be classified into combinational and sequential Trojans. Trojan can also be classified into digital and analog based on the payload mechanisms. Digital Trojans invert the logic values at internal nodes or modify the contents of memory locations, while the analog payload Trojans may affect circuit parameters, such as performance, power, and noise margin.

A combinational Trojan is activated on the simultaneous occurrences of a particular condition at certain internal nodes, while a sequential Trojan act as a time-bomb, exhibiting its malicious effect due to a sequence of rare events after a long period of operation. Fig 1.1(a) illustrates the general scenario of a Trojan attack in a design, where a Trojan is realized through the malicious modification of the circuit with a trigger condition and payload. Fig 1.1 (b) shows an example of combinational Trojan which does not contain any sequential elements, and depends only on the simultaneous occurrence of a set of rare node conditions. Conversely, the sequential Trojans shown in Fig 1.1 (c) undergo a sequence of state transitions before triggering a malfunction. The 3-bit counter causes a malfunction at the node S on reaching a particular count, and the count is increased only when the condition a = 1, b = 0 is satisfied at the positive clock-edge. Protection against hardware Trojan has been widely explored by researchers. These approaches are based on the following three approaches:

(1) Specialized functional testing that rely on triggering an unknown Trojan and observing its effect in output ports of a design;

(2) side-channel analysis that rely on observing a Trojan effect in physical parameters, such as supply current or path delay and

(3) design/integration approaches that either prevent a Trojan insertion or facilitate detection during production test.

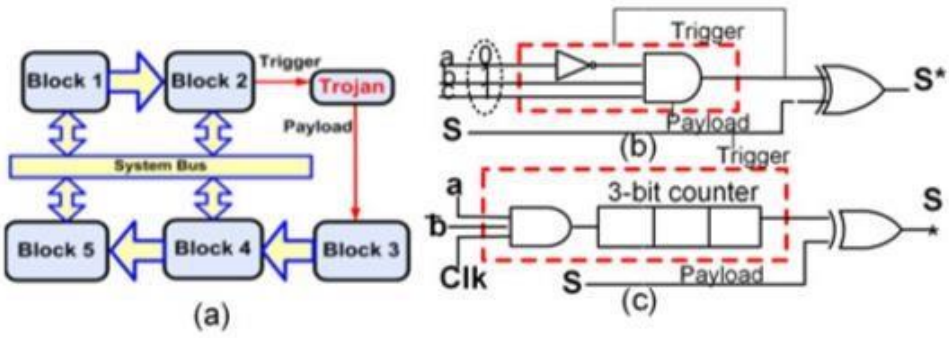


Fig1.1 (a) General model of a hardware Trojan circuit realized through malicious modification of a hardware. (b) An example of combinational Trojan. (c) An example of sequential Trojan.

1.3 Motivation

The main idea for working on hardware trojan detection is to control the cyber-crimes to the maximum extent. For example, in the year 2007 the backdoor built into a Syrian radar system was responsible for the system's failure. There are also reports of Trojans being used by the USSR to intercept American communications during the cold war. Time to activate a hardware Trojan circuit is a major concern from the authentication standpoint.

It is also a direct threat to the already vulnerable Internet of Things, meaning that wireless-enabled household devices also become potential targets. The problem is such that even previously 'reputable' factories are vulnerable to attacks, since all that is required is one employee to alter the existing code to include a trojan. As most IC designs are extremely large and contain a huge amount of hardware description, these inclusions are difficult to detect and the sheer size of the code can require many people having access to the code at production level.

Regarding military grade products utilizing ICs, the problem of hardware trojans is critical with the threat level of the trojan being such that it could potentially be catastrophic. Malicious inclusions of code could cause lifesaving equipment to fail, missiles to lose control, and cryptography keys to be leaked. While incidents of hardware trojans, are not openly discussed there have been a few noted. In 2007, it was assumed that a backdoor built into a Syrian radar system was responsible for the system's failure. There are also reports of trojans being used by the USSR to intercept American communications during the cold war. The problem is aggravated further still when considered in relation to the growth in production of counterfeit goods. Such goods may be produced in less than reputable factories, so the inclusion of malicious code in the production process is far from unrealistic. As counterfeit goods are not generally sold through trustworthy channels, it is impossible to recall products found to be unsafe or indeed to produce updated firmware to deal with emerging threats. This can expose consumers to a plethora of malicious attacks by hackers. For example, a trojan leaking cryptography keys in counterfeit IoT devices could potentially give hackers access to a network of devices that can be utilized in 'Mirai' like attacks and cannot be recalled or patched In this paper, a hardware trojan is created and emulated on a consumer FPGA board. The experiments to detect the trojan in a dormant and active state are made using off-the-shelf technologies which rely on thermal imaging, power monitoring, and side-channel analysis.

Unfortunately, those theories are not in nature groundless or out-with the realms of the possible. In fact, several of them have already been instantiated. One such rumour of 'kill switches' being hidden in commercial processors was confirmed by an anonymous U.S. defence contractor who indicated the culprit to be a 'European Chip Maker'. The potential consequence of the existence of such a switch could be catastrophic. Indeed, as previously highlighted, this particular hardware trojan was blamed for the failure of a Syrian radar to detect an incoming air strike. The Threat of the Hardware Trojan. At Design Level The complexity and cost

of the design of ICs has grown exponentially over the last decade as the semiconductor industry has scaled to sub-micron levels. A typical IC board will go through a rigorous process consisting of several stages. Firstly, the specifications must be translated into a behavioural description, usually in a hardware description language such as Verilog or VHDL. Once this has been completed, the next phase is to perform synthesis to transform the behavioural description into a design implementation using logic gates such as a net-list. Once the synthesis has been completed, the net-list is implemented as a layout design and the digital files are passed to the foundry for fabrication. As well as outsourcing the production of ICs, many companies are also purchasing third party Intellectual Property (IP) cores, and utilizing third party Electronic Design Automation (EDA) tools. Each use of third-party software presents a new opportunity for attacks such as hardware trojan insertion, IP piracy, IC tampering, and IC cloning. Although these attacks are all of importance, the hardware trojan is by far the most dangerous attack, and, as such, has garnered much attention. At Foundry Level As semiconductor technology has advanced, the cost of owning foundry has increased dramatically. In 2015, the cost was estimated to be in the region of 5 billion USD. As a direct result of this, many companies can no longer afford to fund the production process from start to finish, and are outsourcing their production to cheaper foreign foundries. Whilst undesirable modifications to ICs should ideally be detectable by pre-silicon verification and simulation, this would require a specific model of the entire IC design and this is not always readily available particularly where third-party IP cores or EDA tools have been used. In addition, large multi module designs are rarely compliant with exhaustive verification.

Post silicone approaches to design verification include destructive de-packaging and reverse engineering of the IC. However, current techniques do not allow destructive verification of ICs to be scalable. It is also possible for an attacker to infect only a portion of the produced ICs, making these tests futile. Most post silicone logical testing techniques are also unsuitable for detecting hardware trojans. This is attributed to the stealthy nature of the hardware trojan and to the large numbers of differing taxonomy's that can be employed by the attackers. Most hardware trojans are programmed to activate under a specific set of conditions, and a skilled attacker would ensure that these conditions were undetectable by the testing routine. This is particularly true of trojans targeting sequential finite state machines. Industries Affected Military Hardware trojans are a huge threat to many industries. However, security conscious industries, such as the military, are in a particularly high-risk bracket and defence departments are very aware of this. The U.S.
Department of Defense (DoD) has created a "Trusted Foundry Program" to ensure its military equipment remains free of hardware trojans by using only accredited foundries. This means that only American foundries which are located on the Americal soil and which underwent the strictest vetting process are allowed to work on the chips for the U.S. DoD. In addition to vetting the foundries, close attention is being paid to the other links in the design and supply chain. While this approach may seem effective, it has its limitations. The majority of western foundries are woefully behind their foreign counterparts when it comes to the level of

technology they can provide. This seriously limits access to more advanced chips which are required for modern avionics and weapons systems.

If the attacker creates the trojan through the modification of the existing code, then it will be classified as a parametric. Typically, this can be achieved by thinning wires or weakening transistors and flip flops. This type of trojan is notoriously hard to detect as the alteration can be minuscule. The next physical characteristic the attacker would have to consider would be the size of the hardware trojan. In this context, the size refers to the physical extension of the hardware trojan or the number of components it consists of. In case of a large trojan consisting of many components, an attacker can distribute these across the IC, placing components where they are necessary to execute their payload in accordance with the functions of the hardware trojan. This is known as loose distribution. In contrast, a smaller hardware trojan consisting of only a few components allows for the components to be placed together as they will occupy only a small part of the layout of the IC. This is known as tight distribution.

On rare occasions, a determined attacker could regenerate the layout to encompass the hardware trojan, moving the components of the IC to accommodate the components of the hardware trojan. This is referred to as a structural alteration Activation Characteristics Typically, a hardware trojan will be condition based, meaning that its activation will be dependent on a trigger defined by the attacker. The trigger itself will generally consist of either a predefined input pattern, or specific internal logic state, or counter value, and can be triggered both internally and externally. An externally triggered hardware trojan will usually consist of malicious logic within the IC that utilizes an external sensor such as a radio antenna. The attacker will then communicate via the compromised component enabling them to trigger the antenna. It is easy to see why this can be extremely dangerous when it comes to security conscious industries such as the military. It is not out-with the realms of the believable to postulate that an attacker could feasibly re-route or switch off a missile via a radio signal as suggested.

Conversely, an internally triggered hardware trojan will look within the circuitry for the set of conditions that will cause it to activate. A typical example of this would be countdown logic. In contrast to the condition-based trojan that will only activate when its trigger conditions are met, the “always-on” trojan is active from the moment of insertion, and relies on internal signals. This type of hardware trojan is generally split into two categories; combinational and sequential. A combinational trojan will activate upon detection of a specific set of circumstances within the internal signals of the IC. Sequential trojans will also monitor the internal signals of the IC. However, instead of looking for a specific condition, they activate when a specific sequence of events occurs.

1.4 Project Outline

The project Hardware Insertion and Detection deals with the hardware security domain where securing information is vital in this sophisticated contemporary world. Day to day technology has been improving along with this the loop holes for hackers are becoming more and causing cyber security problems. In order to prevent and control these problems this project gives a scope to control the problems. By continuous analysis of the circuit the parameters like path delay can be found changed if trojan is present.

A circuit is said to be trojan effected if its parameters show a significant change with the parameters of trojan free circuit. The trojan free circuit is called the golden circuit. Different trojans can be detected using different methodologies. Most popular and effective method is side channel analysis. BPUF and RO PUF circuits implementation and their importance is discussed. All these circuits are simulated using Xilinx Vivado and reports were generated and the parameter were compared for trojan free and trojan effected circuit.

CHAPTER 2

HARDWARE SECURITY

2.1 Introduction

Hardware is a collection of physical elements that constitutes a computer system. Hardware is used by everyone even if they are not aware of it. Hardware in this context might be:

a. Computer Hardware: Some computer hardware are Processors, firmware, memory etc.

i. Processors: is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions

ii. Firmware: is the combination of a hardware device, e.g., an integrated circuit, and computer instructions and data that reside as read only software on that device

iii. Memory: Memory refers to the device used to store information for use in a computer.

b. Mobile Hardware: Sim Card, RFID/Smart Card, Chip and Pin

i. Sim Card: is an integrated circuit that is intended to securely store the international mobile subscriber identity (IMSI) and the related key used to identify and authenticate subscribers on mobile telephony devices (such as mobile phones and computers).

ii. RFID: is the wireless use of electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. For example, an RFID tag attached to an automobile during production can be used to track its progress through the assembly line.

iii. Smart Card: is any pocket-sized card with embedded integrated circuits. Smart cards are made of plastic, and can provide strong security authentication for single sign-on (SSO) with large organizations

iv. Chip & Pin: "Chip" refers to a computer chip embedded in the smartcard, and "PIN" refers to a personal identification number that the customer must supply. "Chip and PIN" is also used in a generic sense to mean any EMV smart card technology that relies on an embedded chip and a PIN.

c. Future Hardware: PUFs (Physically Unclonable Functions) PUFs have a unique fingerprint in a physical object that means if you have an object with one fingerprint; another object with the same fingerprint cannot be created. It uses challenge/response for its operations. The challenge/response explains that if a message is

sent to a physical object and the physical object is changed to another physical object, and same message is sent to the second physical object, the two physical objects react differently because of their unique fingerprint.

2.2 Attacks on Hardware

1.Physical Attacks: The main thing that differentiates hardware attacks from software attacks is the physicality of the attack done with hardware tools. This raises the bar for hardware attacks because any attacker that wants to perform an attack on the hardware needs to have extensive knowledge of the hardware, unlike the software attacks that can be done by just downloading a vulnerability tool on the internet to perform attacks.

2. Secret Key: Generally, the hardware wants to protect a secret so the secret is embedded in a physical object. For example, the bank card wants to protect your pin, the pin is encoded in the card and if the attacker can probe the chip of the card and read the pin then the card is useless. The secret in the hardware should not be writable even though it provides the information on the card when placed on a terminal. If we consider STRIDE, we have to consider two main points: The Information Disclosure (Confidentiality) which means something is hidden, and Tampering (Integrity) which means it is not writable.

3. Attack Vectors: The hardware that would be used to protect the secret would be fabricated by someone in a factory. The factory will either program the secret onto the hardware or send the hardware to the company with memory for read access in one component and write access in another component so that the company can program the hardware and destroy the write component to avoid rewriting. This approach can't be used for all hardware. For example, subway cards need to be rewritten to every month so in that case, the terminals have written access to the card but the user doesn't have write access so it can't be overwritten by the user. To fabricate the hardware, the laboratory/factory needs to be trusted. And to prove they are trustworthy; the laboratory gets certified. The certification body sends people in, to audit them, check out the employees and their procedures and conclude whether the laboratory is trusted or not.

4. Supply Chain: After the hardware has been made, it will be shipped to stores that will sell it or it is shipped to the consumers from the store. During shipping it could be intercepted by the attackers and tampered with, then re-packaged without the knowledge of the store or the consumers. Also, some attacks can be performed on point-of-sale terminals when some insider (employees) have access to the terminals and tampered with it in the warehouse

5. Accidents: there are a lot of memory-based devices (e.g., USB Keys, Digital picture frames) which may contain malware in them accidentally, which could affect the system of the user. The company that created this hardware may not be aware of the malware on the device. Examples of companies that had this kind of accidents are IBM, Dell, Samsung, HP, Apple, etc. All the attacks stated above are the main reasons for why the user might get a bad hardware.

2.3 Attack Circuits

RFID, Smart Card, Micro- Controller, ASICS, FPGAs RFID is passive, a signal can be sent to it and it responds. It can't be programmed; it can't perform any computations based on the signal sent to it. Smart card, on the other hand, performs computations based on what was sent to it. Micro-controller is like an ID with no chip, ASCIS are fabricated circuits that are custom made to do implementations so all your processors and memories are on ASICS. ASICS are expensive because they are custom made, and before the design is committed to ASICS, a lot of testing will be done with FPGAs to make sure that circuit actually works. FPGAs are programmable chips. When the chip is bought, its blank and it can be programmed with software to do whatever you want. They are not as fast as ASCIS because they are general purpose. FPGAs are faster than software but slower than ASCIS. Why are we attacking Circuits? The main reason for attacking circuits is to recover a secret that had been encoded on a piece of hardware or for the attacker to program a certain value to the circuit. The secret could be the actual algorithm itself. Some attackers reverse engineer the algorithm of the RFID or Smart Card to find flaws in the algorithm itself. This is because some developer wants to keep the algorithm used to protect the circuit a secret due to the fact that the developers are not using a standard algorithm. The algorithm used should be a standard algorithm so as to know how to better protect it. The Circuit attacks are:

1. **Black Box Testing:** To perform this attack, the attacker sends an input to the circuit and receives an output. Based on the input and output behavior, the attacker will decide what kind of algorithm to used. An example is Speed Gas RFID which are proprietary stream cipher. The attackers found the documentation and modified it to discover the cipher used and break the circuit. This type of attack is non-invasive, meaning that the card/chip will not be destroyed when probed so it can be used another time. Another method that can be used in black boxing is fuzzing in software security which allows large random inputs to the circuit and get strange responses like undocumented features, factory testing, etc.

2. **Physical Probing:** To perform this attack the attacker sticks a probe unto the chip itself and reads data off the chip. Within a circuit, there is a wire that connects components to each other called the bus and the bus is where the information would be read as the data is moving around in the bus. The data can also be read off the memory location in the circuit. The probe can have a submission precision and it's an invasive method. A lot of circuits are driven by a clock, and if the attacker can slow down the clock it gives a lot of time to the attacker to read the voltage of the circuit.

3. **Reverse Engineering:** To perform this attacker, the attacker must acquire the smart card and physically expose the circuit. The smart card is manufactured with different layers and each layer is removed until the physical circuit is exposed. The attacker then takes high resolution photographs of the circuit and uploads it to a computer and uses a code and machine learning application to figure out what the actual circuit does. Once

the circuit is figured out, then the algorithm used in the smart card also is exposed and the algorithm can be broken. An example of a smart card where the reverse engineering attack was performed on is the Mi fare (Subway card).

4. **Fault Generation:** Some technologies fail. E.g., A TV provider sends a message to the client asking if the clients want to renew the subscription. If TV providers don't receive a message NO from the clients, they don't disconnect, so the clients are granted access. So, some clients can just cut the power at the right time to prevent the response to the TV providers, since they won't disconnect with no response. This is a non-invasive attack. Other things that can be done are modifying the memory contents (non-invasive), glitch (rapid change) the power or clock (non-invasive), heating up components e.g., with a user (semi- invasive), modify chip e.g., cutting wires (invasive) etc.

5. **Side Channel Analysis:** To perform this attack, the attackers make use of the hardware normally but makes sensitive measure of certain things and based on the measurements done, the attacker can infer secrets. An example of things to measure is power (the amount of voltage in an ATM), timing analysis in cryptography (software effect), electromagnetic emission, acoustic sounds (performed on RSA). These are called side channel because they are outside the normal channels. They are non-invasive. It is slower than the normal attacks.

2.4 Counter Measures

1. Obfuscate data (Scramble, encrypt) on buses

2. Obfuscate the ASICS layout, 3D stacking

3. Metal mesh on top of the circuit (if the circuit is probed, it causes a short and the memory resets)

4. Side Channel: physical shields, asynchronous circuits.

Also, a decrease in the signals from the circuits of the hardware like the noise or add artificial noise or low the circuit's power. METHODOLOGIES-There is no good methodology for hardware (that means no static analysis or dynamic analysis of hardware). It is an open question that needs to be researched on. Most of it has to do with domain specific knowledge and it is advisable to follow the requirement engineering process. Common criteria/NIST has protection profiles which provide the properties not how to achieve them.

2.5 Hardware Trojan:

A hardware trojan can be described as a malicious alteration or inclusion to an integrated circuit (IC) that will either alter its intended function or cause it to perform an additional malicious function.

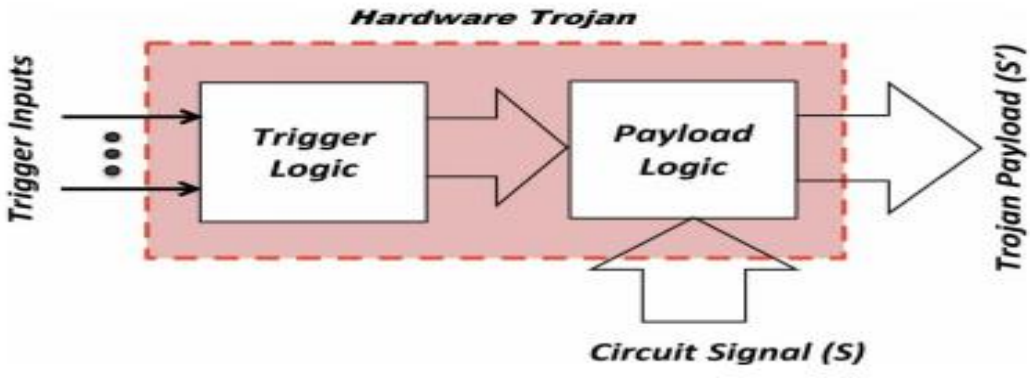


Fig 2.1: Block Diagram of Hardware Trojan Inserted Circuit

These malicious inclusions or alterations are generally programmed to activate only under a specific set of circumstances created by an attacker and are extremely hard to detect when in their dormant state.

Block diagram of typical hardware trojan inserted circuit is shown in fig2. 1.

As technology advances, so does the demand for IC boards leaving many technology companies without the resources to produce secure enough ICs to meet current demands.

This has pushed companies into the 'fabless' trend prevalent in today's semi-conductor industry, where companies are no longer attempting to produce the goods in their own factories, but instead are outsourcing the process to cheaper factories abroad. This growth brings with it a significant rise in the level of threat posed by hardware trojans; a threat that directly affects all companies concerned with products that utilize ICs. This encompasses many different industries, including the military and telecommunications companies, and can potentially affect billions of devices from mobile phones and computers to military grade aviation and detection devices, particularly at a time when wireless devices are being introduced as links in critical infrastructure, compounding trust and security issues even further.

2.5.1 Action Characteristics:

The action characteristics of a hardware trojan refer to the effect the trojan will have on the execution of its payload. Hardware trojans will typically fall into one of two categories: implicit or explicit. Implicit trojans will not change the board's circuitry of the IC; instead, they will perform their malicious function in tandem with the intended function of the board. This makes these trojans easier to detect as they tend to cause small path delays on activation and consume more power whilst active.

In contrast, an explicit trojan will change the function of the board's circuitry on activation. This can come in the form of a signal alteration or even leaking of information via predefined board pins. These trojans tend to cause distinct path delays as well as large changes in circuit's capacity Hardware Trojan Detection requires overcoming numerous challenges. Namely:

1. Handling large architectures.

2. Being non-destructive to the IC.

3. Being cost effective.

4. Ability to detect trojans of all sizes.

5. Authenticating chips in as small a time frame as possible.

6. Dealing with variations in manufacturing processes.

7. Detecting all trojan classifications.

8. Detecting trojans in a reasonable time frame.

There is no single method capable of detecting all types of hardware trojans, nor overcoming all the challenges described here-above. Over the years, several methods have been developed to detect different types of trojans. These methods are described here-after.

Physical Inspection One of the most obvious method of detection is physical inspection of the board itself. This method is sometimes classified as a failure analysis-based technique. Those techniques usually comprise two steps:

(1) Cutting and lifting the moulding coat to expose the circuitry; and

(2) Performing various scans

Functional Testing Often referred to as Automatic Test Pattern Generation (ATPG) this technique is more commonly used to locate manufacturing faults; it has been shown to be effective in detecting hardware trojans. ATPG involves inputs of ports are stimulated and then the output ports are monitored for variations that may indicate a hardware trojan has been activated. Functional testing techniques can also be useful when attempting to determine the trigger patterns of conditional trojans. Built-In-Self-Test (BIST) techniques are commonly used to detect manufacturing faults and are present in many chips. If unknown or malicious logic is detected during these tests a bad checksum result is given, although designed to detect manufacturing faults on some occasions these tests can detect hardware trojans.

Side channel analysis techniques are some of the most commonly used procedures in hardware trojan detection. These techniques generally measure signals such as power and path delay, looking for fluctuations potentially caused by trojans. Side channel analysis can have a high success rate as even in a dormant state the trojans trigger signal will cause some current leak

2.5.2 Payload Characteristics

Hardware Trojans can also be classified based on their intended behaviour. Trojans can be inserted for causing malfunction or for leaking sensitive information. In the former case, Trojans alter the functionality of the design in some way, while Trojans designed for leaking sensitive information may do so without modifying the logic functionality of the design.

Trojans for Malfunction can be further classified into two subcategories based on whether they cause logical malfunction or physical malfunction. Trojans presented in the previous sections cause logic malfunction by modifying the values in the LUTs, causing undesired routing between two logic modules, etc. Fig.2.2 shows additional examples of payloads affected by Trojans.

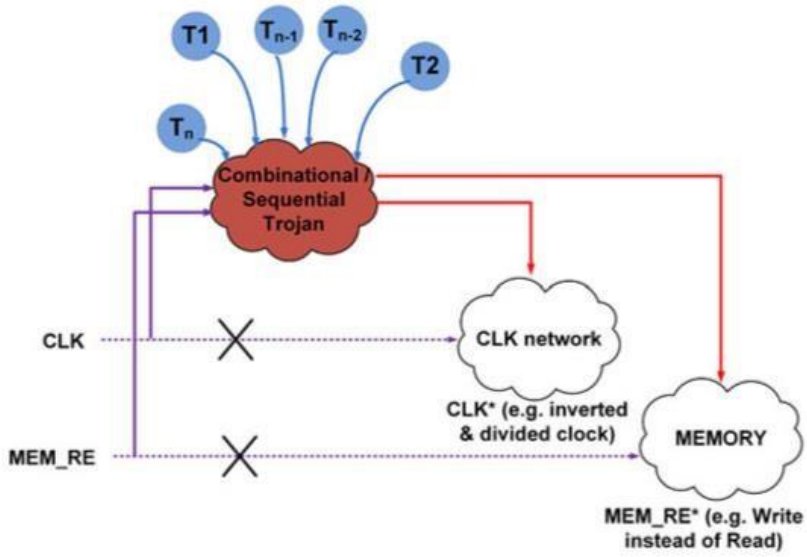


Fig: 2.2Diagram showing examples of payloads that can be altered by an implanted Trojan circuit.

Trojans intended to cause physical damage can create electrical conflicts at the I/O ports or at the programmable interconnects. Consider the programmable I/O block in Fig. 2.5. When an I/O port is configured to be an input by a design, the configuration cells in the I/O block should disable the output block to prevent internal conflicts. A counter-based Trojan can be inserted in the foundry which detects the state of the I/O port and begins counting. When the counter counts to the final value, the Trojan may enable the output logic when the port is configured as an input. This would cause damaging the system. These Trojans are similar to the MELT viruses described in except that Trojans causing physical destruction may also be inserted in the foundry.

The device also contains a decryption module for decrypting the bit-stream using a key stored in a non-volatile memory. Security measures in the device

(1) Prevent the key from being read and sent to a port by clearing the configuration data and keys when a read attempt is made,

(2) Prevent read-back of the configuration data, and

(3) Restrict decryption access after configuration.

However, all of these measures only prevent malicious code in an IP from accessing the key or configuration data. Hardware Trojans can leak the IP in two ways: by leaking the decryption key, or by leaking the design itself. An attacker in the foundry can insert an extraneous circuit to tap the wires connecting the non-volatile memory and decryption module. Even if the decryption module is implemented in the logic array by using a decryption bit-stream as mentioned in, such an instantiated module must have access to the non-volatile key for decryption. A copy of the key can be stored in the Trojan, which may then leak it through side-channels or covert-channels. Using side-channels, a Trojan can hide the key in the power traces or by emitting electromagnetic radiation containing the information and an attacker can observe these signals to steal the key. For example, the MOLES Trojan presented in uses a spread-spectrum technique to leak the key in the power traces over several clock cycles. Alternatively, a Trojan may also multiplex the JTAG port, USB port, or any other programming port to leak the key through covert channels when the ports are not being used. The Trojan MUXes the bit-stream and rather than sending it to the decryption, it may store blocks of the bit-stream at any given time and leak them through side-channels or covert-channels.

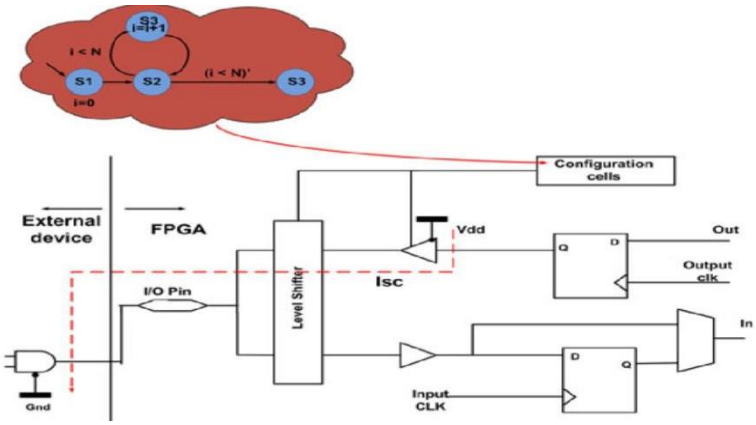


Fig 2.3 Programmable I/O block containing hardware Trojans to cause logical and electrical malfunction.

2.5.3 Trojans in Cryptographic Engines

A possible Trojan attack in a crypto engine can try to subvert the security mechanisms. The payload could range from a mechanism that presents dummy keys, predefined by the attacker, instead of the actual cryptographic keys used for sensitive encryption or signature verification operations, to leaking the secret hardware keys via covert side channels, e.g., information leaked through a power trace. Fig. 2.6 provides an example of such a Trojan that attempts to leak a secret key from inside a cryptographic IC through power side-channels using a technique called malicious off-chip leakage enabled by side channels (MOLES). Even if the IC has been designed to minimize side-channel information leakage, a hardware modification could help overcome the protection under specific circumstances where the attacker is in possession of the system or physically near the system to extract the secret information. Other targets could be a random

number generator used for deriving random session keys for a particular operation or the debug passwords used for unlocking test-mode access to security-sensitive signals. Researchers have also proposed leaking such secret information over wireless channels by using low-bandwidth modulation of the transmitted signal.

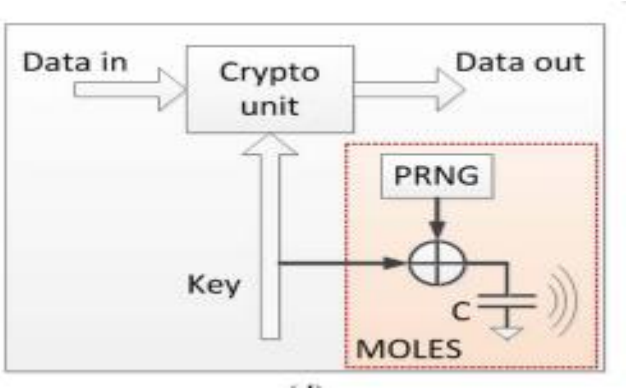


Fig 2.4 Trojans with capability of leaking secret information from inside a crypto chip.

2.5.4 Trojans in General Purpose Processors

In general- purpose processors, an attacker at the fabrication facility can implement a backdoor, which can be exploited in the field by a software adversary. For example, modern processors implement a hardware chain of trust to ensure that malware cannot compromise the hardware assets such as secret keys and memory range protections. By using different stages of firmware and boot code authentication, one can ensure that the operating system (OS) kernel and lower levels (such as hypervisor) are not corrupted. However, in such systems, the attacker at an untrusted fabrication facility could implement a backdoor which disables the secure booting mechanism under certain rare conditions or when presented with a unique rare input condition in the hands of an end-user adversary. Similarly, other objectives which could be realized with the help of hardware Trojans would be to bypass memory range protections using buffer overflow attacks or to gain access to privileged assets by evading the access control protection mechanisms implemented in the hardware.

2.6 Types of Hardware Trojans

There are different types of hardware Trojans like Combinational, Sequential and Hybrid Trojans as shown in fig2.5

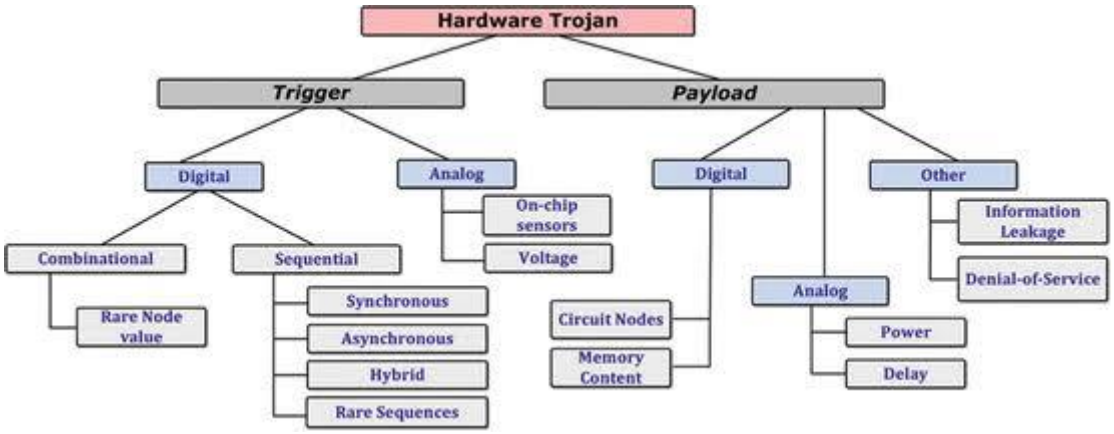


Fig2.5 Types of Hardware Trojans

Trojans are classified based on their trigger and payload mechanisms, as shown in Fig. 2.5. Hardware Trojan attacks in different forms: Combinational (whose activation depends on the occurrence of a particular condition at certain internal nodes of the circuit) and sequential (whose activation depends on the occurrence of a specific sequence of rare logic values at internal nodes). Trojans are based on three attributes: physical, activation and action. Based on the activation mechanisms (referred as Trojan trigger) and the part of the circuit or the functionality affected by the activation of the Trojan (referred as Trojan payload). The trigger mechanisms can be of two types: digital and analog. Digitally triggered Trojans can again be classified into combinational and sequential types. Sequentially triggered Trojans (the so-called time bombs), on the other hand, are activated by the occurrence of a sequence, or a period of continuous operation. The simplest sequential Trojans are synchronous stand-alone counters, which trigger a malfunction on reaching a particular count. The trigger mechanism can also be hybrid, where the counts of both a synchronous and an asynchronous counter simultaneously determine the Trojan trigger condition. Note that more complex state machines of different types and sizes can be used to generate the trigger condition based on a sequence of rare events. In general, it is more challenging to detect sequential Trojans using conventional test generation and application, because it requires satisfying a sequence of rare conditions at internal circuit nodes to activate them. The number of such sequential trigger conditions for arbitrary Trojan instances can be unmanageably large for a deterministic logic testing approach.

2.7 Combinational Trojan

This trojan is combinational sort of trojan. In this, we will interface the combinational circuit as the hardware trojan alongside the Trusted IC's unique circuit i.e.,256:1 mux.

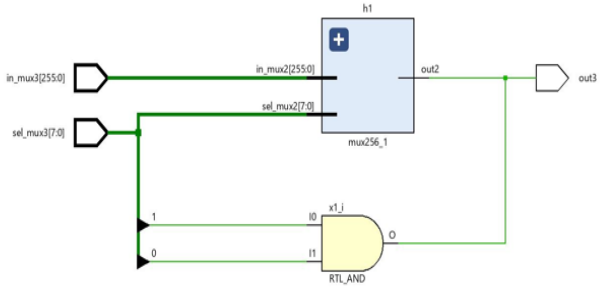


Fig2.6 Combinational trojan inserted in 256:1 MUX

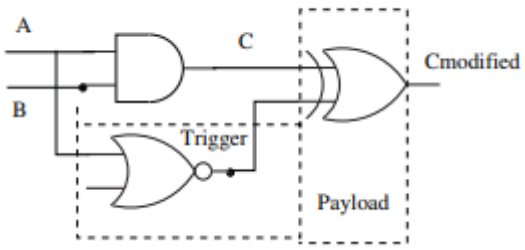


Fig .2.7 Combinationally triggered trojan

A	B	$C = A \cap B$	$\overline{A \cup B}$	C_{mo}
0	0	**0**	1	**1**
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

TABLE I: Truth table of circuit shown in Fig. 2.7

In Fig. 2.7, the trigger and payload are shown for the original circuit $C = A.B$. The input 00 of AND gate is taken as the rare condition. That is, in general case, 00 input pattern will not appear. When an outsider chooses this input pattern, the trigger logic gets activated. From the columns 3^{rd} and 5^{th} of Table I, it is clear that for input pattern 00, the trigger (NOR gate) is activated and the trigger action goes through an XOR gate to activate

the payload logic. As a result of this, it is observable that the outcome of an AND gate is 1 when inputs are 00, which is not true for normal operation.

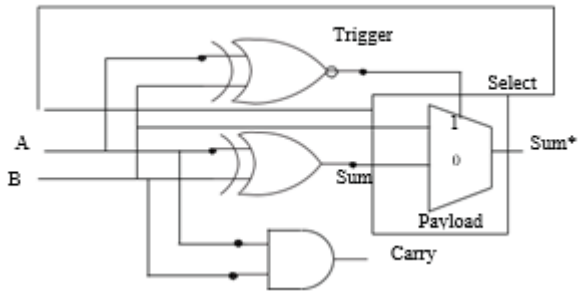
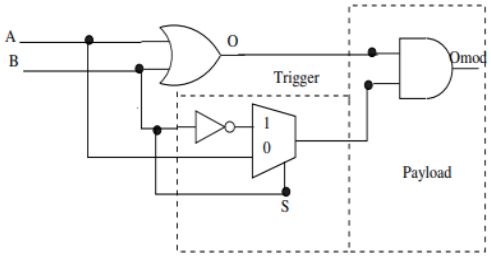


Fig. 2.8: Combinationally triggered trojan in half-adder

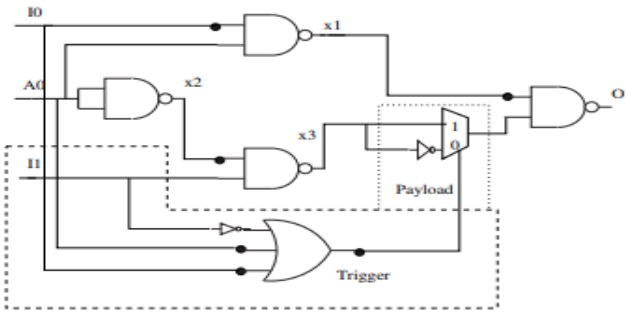
A half adder of Fig. 2.8 is embedded with a Hardware Trojan (HT) as reported. In original half adder circuit, the outputs (carry and sum) for the 4 input sequences are 00, 01, 10, 11 respectively. Here, 11 input patterns is considered as the extreme rare condition. That is, in general case, 11 input patterns will not appear. But if forcefully the malicious adversary generates the 11-input sequence, then the trigger logic becomes active and the original outcome (Carry = 1, Sum = 0) is altered, say 11. Here, the XNOR gate works as the trigger logic and the multiplexer acts as the payload logic. When the rare input 11 is passed, then the output of the XNOR gate selects input 1 of the multiplexers - that is, $B = 1$. The generated outcome is dissimilar with the normal output. This class of hardware trojan is referred to as the HT with explicit behaviour. On the other hand, for the other input combinations, the Sum outputs (of Fig. 2) are same as that of trojan free circuit. For such input patterns, it is not possible to expose the existence of trigger and payload circuit. In that cases, the HT shows implicit malicious behaviour.



A	B	$O = A \cup B$	$\overline{B}$	Omod $S = 0 \Rightarrow A \cdot O, S = 1 \Rightarrow \overline{B} \cdot O$
0	0	0	1	0
0	1	1	0	**0**
1	0	1	1	1
1	1	1	0	**0**

Fig. 2.9: Two-input based combinationally triggered trojan TABLE II: Truth table of circuit shown in Fig.2.9

Fig. 2.9 shows the trigger and payload logic with the original logic circuit (OR gate). Here, any one of the 00, 01, 10 and 11 input patterns can be taken as the rare pattern. The select line $S = 1$ denotes that the output *Omod* is *B.O* and $S = 0$, sets *Omod* = *A.O*. *O* is the output of original circuit. The truth table for the circuit with HT is shown in Table II. Column 5 notes the values of *Omod* when $S = 0$ and $S = 1$.



A	I1	I0	Q	Qmod
0	0	0	0	0
0	0	1	0	0
0	1	0	**1**	**0**
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Fig. 2.10: Three-input based combinatioanlly triggered trojan TABLE III: Truth table of circuit shown in Fig. 2.10

In Fig. 2.10, 010 input pattern is the rare condition. The trigger logic is designed in such a way that for this particular pattern (010), the original output is flipped (as in row 3, of Table IV). That is, its detection probability is- 1/8 (best possible case)

2.8 Delay Trojan

Trojan circuit is created by introducing some delay in 256:1 MUX. In this trojan implementation, until count 252 we will get the same 256:1 mux output, yet, after check 252 we will get defective output. Since a trojan block is added to the 256:1 mux block. The variations between the brilliant circuit and trojan circuit have been investigated.

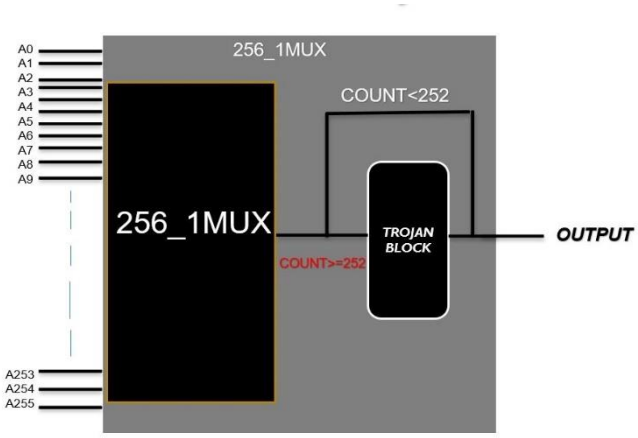


Fig.2.11 Delay Trojan Inserted in 256:1 MUX

Fundamentals of Delay-Based HT Detection Methods. This section introduces the three fundamental technical domains that need to be considered by path delay-based methodologies: 1) the test vector generation strategy, 2) the technique employed for measuring path delays, and 3) the statistical detection method for distinguishing between process variation effects and HT anomalies. A commercially viable HT detection method must address each of these in a cost-effective manner. We investigate the challenges associated with each of these domains and describe proposed solutions in this section. Many of the methods surveyed in Section 5 address only a subset of these technical domains and therefore must be combined with other techniques to be fully operational in practice.

Hardware Trojan (HT) detection method is presented that is based on measuring and detecting small systematic changes in path delays introduced by capacitive loading effects or series inserted gates of HTs. The path delays are measured using a high resolution on-chip embedded test structure called a time-to-digital converter (TDC)

that provides approx. 25 ps of timing resolution. A calibration method for the TDC as well as a chip-averaging technique are demonstrated to nearly eliminate chip-to-chip and within-die process variation effects on the measured path delays across chips.

CHAPTER 3

PHYSICALLY UNCLONABLE FUNCTIONS

3.1 Introduction

Mobile and embedded devices are becoming ubiquitous, interconnected platforms for everyday tasks. Many such tasks require the mobile device to securely authenticate and be authenticated by another party and securely handle private information. Physical unclonable functions (PUFs) are a promising innovative primitive that are used for authentication and secret key storage. PUF, is a physical object that for a given input and conditions (challenge), provides a physically defined "digital fingerprint" output (response) that serves as a unique identifier, most often for a semiconductor device. Today, PUFs are usually implemented in integrated circuits and are typically used in applications with high security requirements, more specifically cryptography.

3.1.1 Why Physical Unclonable Function (PUF)?

20 years ago, digital security was implemented only in dedicated electronic devices such as banking cards or payment terminals. Today everyone connects to its bank using a secure internet connection signalled by "https://" and we all expect that the information we manipulate with our smartphone is protected. Cryptographic techniques such as encryption or digital signature have been deployed to meet these requirements. As a consequence, a growing number of ASICs, microcontrollers and SoCs embed hardware cryptographic accelerators or software cryptographic libraries. The emergence of the Internet of Things (IoT) will call for an even faster adoption. We now can talk about cryptography pervasion.

Such pervasion has been made possible because in modern cryptography algorithms are public and standardized. The immediate consequence of algorithms being publicly known is that keys become the most valuable assets, hence they must be strongly protected.

Historically, the first Integrated Circuits (IC) designed to strongly protect keys were the smartcard ones. With the growing need for digital security, cryptography has been implemented in more and more ICs such as generic microcontrollers but the protection of keys is always a challenge. The main drawback of the obfuscation methods is that they also require highly specialized know how, mastered by only a few IC designers. Such solutions are not available of the shelves and are thus inapplicable in many cases. We will see that Physical Unclonable Functions (PUF) delivered as IPs enable the highest levels of security even for non-security experts. A fundamental difference between the traditional techniques and PUFs is that PUFs are, by nature, immune to reverse engineering techniques.

PUFs implement challenge–response authentication. The applied stimulus is called the challenge, and the reaction of the PUF is called the response. A specific challenge and its corresponding response together form a challenge–response pair.

3.2 Physically Unclonable Function

In general, the function which cannot be copied or regenerated is called as unclonable function. So, attacker cannot produce copy of the device even with physical access. PUFs can produce various responses for various challenges at various times. A PUF is based on the idea that even though the mask and manufacturing process is the same among different ICs, each IC is actually slightly different due to normal manufacturing variability. Physical unclonable functions (PUFs) are increasingly used for authentication and identification applications as well as the cryptographic key generation. An important feature of a PUF is the reliance on minute random variations in the fabricated hardware to derive a trusted random key. The characteristics of a PUF are to be robust (stable over time), unique (so no two PUFs are the same), easy to evaluate (to be feasibly implemented),

difficult to replicate (so the PUF cannot be copied) and very difficult or impossible to predict (so the responses cannot be guessed).

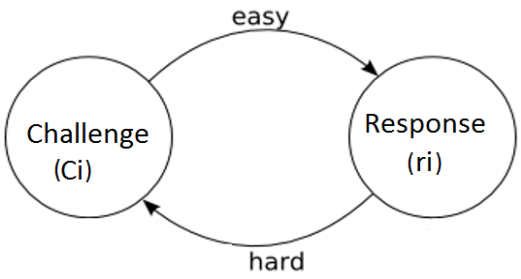


Fig 3.1 Cycle of Challenge and response

3.2.1 Principle used in PUF

A turnkey solution for implementing secure storage while providing a higher level of protection than traditional techniques - which involves custom design - may sound like the security Graal. We will see that robust and easy to integrate PUF is now a reality.

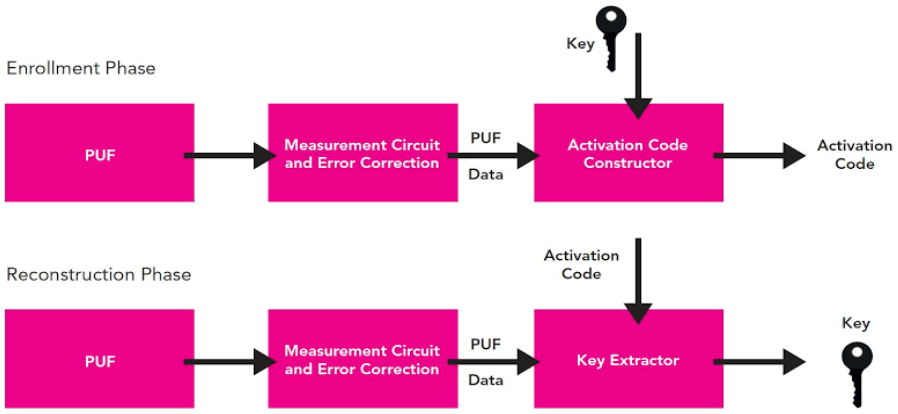


Fig 3.2 Block diagram of PUF Construction

PUFs rely on minuscule manufacturing variations. The manufacturing variations result in devices mismatch. The idea is that two (or more) devices that are identical by design will actually have different electrical characteristics. The difference in the electrical characteristics is unpredictable and cannot be estimated through observation, neither optical, nor SEM.

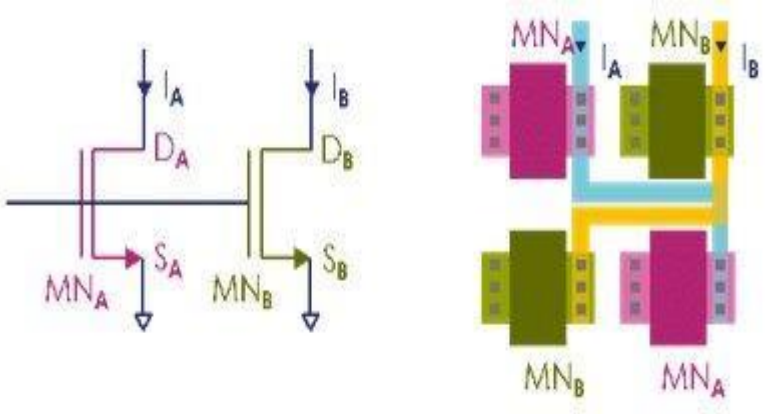


Figure 3.3 Schematics and layout views of a transistor pair as a PUF element

In the above schematic, although the two transistors A and B are identical by design, they always have in practice slightly different physical characteristics. Parameters such as threshold voltage (VT), drain-source current (IDS) or drain-source resistance (RDSON) are different. Designers may choose different parameters to build their PUF. In order to stay generic in this paper we will refer to "parameter" PA and "parameter" PB, keeping in mind that it could be any transistor parameter or a combination of them.

As transistors A and B are identical by design, it is impossible neither by simulation nor reverse engineering to predict for each structure whether we will have PA > PB or PA < PB. If we arbitrarily decide that PA > PB generates a “0” and PA < PB a “1”, it is then impossible to guess whether the pair will generate a “0” or “1” when sensed. By repeating our structure N times we can generate an unpredictable stream of N bits. We have just designed a Physical Unclonable Function.

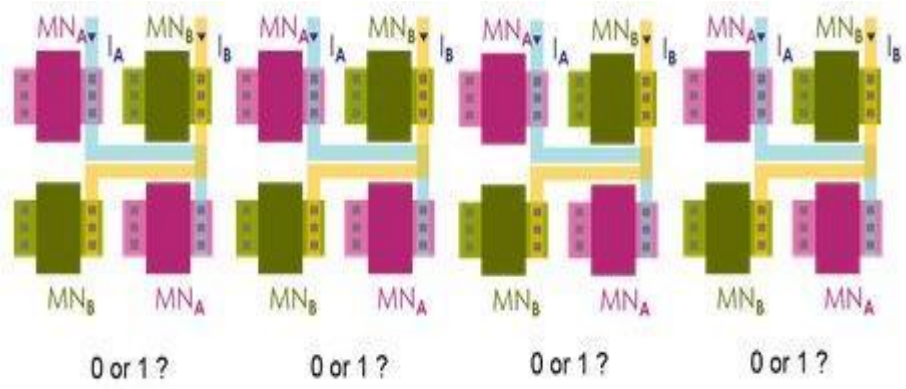


Fig 3.4 Multiple instances of the transistor pair create an un predictable bit stream

3.3 Types of PUF

PUFs are categorized as follows:

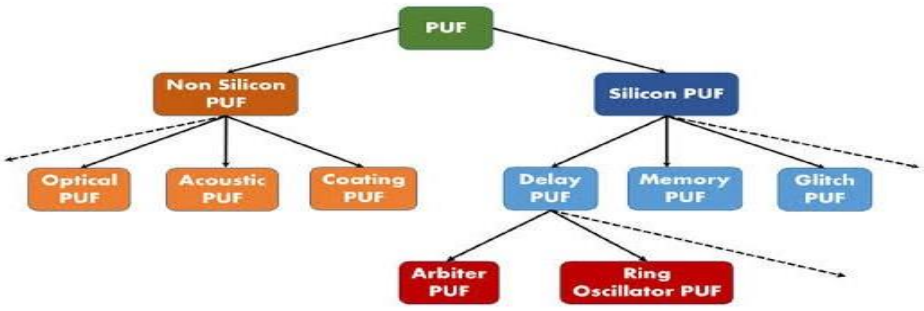


Fig 3.5 Types of PUF

3.3.1 Based on fabrication:

a) Silicon PUFs: These types of PUFs are interfaced with other ICs and are fabricated on the same die as part of the circuit.

b) Non-Silicon PUFs: PUFs that are not classified as silicon PUFs are referred as non-silicon PUFs. They are fabricated in silicon systems but require special fabrication techniques which are not part of generic CMOS fabrication technology.

Types of silicon and non-silicon PUFs

Physical gradation:

- PUF on the optical elements-

Any PUF in which the reaction of the system is based on any optical phenomenon called optical. It is generally used non-uniform transparency object (like reflective bubbles in the example above).

- PUF on integrated microcircuits-

Despite the fact that the microcircuits are made by the same manufacturing process, each of them is unique enough to allow PUF work correctly. It can be used in information security systems as a unique identifier of the device. Integrated microcircuit is covered with a protective substance with dielectric inclusions. These inclusions have random size and shape.

- PUF on field-effect transistors-

The basis of such PUFs is delay of the signal passing through it at unpredictable times, depending on the physical properties of the material of the transistor. The system response to a request. That reaction is unique to this device.

- PUF on magnetic elements-

Practical application - unique identifier of the magnetic stripe of a credit card. Magnetic strip also have a random size and shape. Unclonability based on the physical imperfection of the manufacturing process.

Operation gradation:

- Arbiter-based PUF-

Arbiter-based physical unclonable function is a type of delay-based PUF. The main idea is to introduce race condition between two digital paths. Both paths end in an arbiter element of a chip, which determines which of paths was faster than another and then outputs a corresponding binary value.

- Ring Oscillator PUF-

Physical unclonable functions based on ring oscillators also use uncontrollable process delay variations of digital components as a source of randomness. Frequencies of ring oscillator outputs differ and that effect is used to form binary response.

- Glitch-based PUF-

This type PUFs base on combinatorial logic circuit behavior glitches. Perfectly, combinatorial circuit has no internal state, meaning that steady-state output is completely determined by its input signals. However, when logical input value changes, it takes some time for output to assume steady-state value. Glitch occurrence is influenced by different logical circuit delays differences from the inputs to an output signal.

- SRAM PUF-

Static Random-Access Memory (SRAM) physical unclonable function operation principle uses state randomness of some SRAM cells right after it is powered up.

- Butterfly PUF-

Butterfly physical unclonable function imitates SRAM cell behavior, forming a bistable circuit. Circuit is forced to an unstable state, then it goes into one of the two stable states, which depends on random delay difference between two cross-coupling data latches and input signal line.

3.3.2 Based on security:

a) Strong PUFs- Strong PUFs are typically used for authentication. Strong PUFs, on the other hand, scale in a manner as to support a much larger set of CRPs. The number of these pairs is so large, in fact, that even if an attacker has access to the PUF they cannot feasibly record them all. If in the manufacturing stage a sample of these CRPs is randomly taken, the chances that the attacker also recorded the response to the same challenge can be negligible. This results in a system where even if the attacker had access to the PUF at a certain point, only the user with physical access to the PUF at the time of the challenge can give the correct response and be authenticated.

b) weak PUFs- Weak PUFs are used for key storage. Weak PUFs support a relatively small number of CRPs, typically as a consequence of a low-order rate of scaling. This means that the full set of these pairs can be read from the device should an attacker gain physical access to the PUF for any given time. While it would not be possible to copy the physical PUF itself, with knowledge of the PUF's

The strength of the PUF depends on the number of challenge response pairs (CRPs) that can be generated from a single device. This, in turn, typically corresponds to how the number of CRPs increases with the increasing device size. This rate of scaling tends to act as the metric that determines the strength of a PUF.

3.4 Butterfly PUF:

A cross-coupled circuit is a basic building block used in all types of storage elements in electronic circuits such as latches, flip-flops and SRAM memories. A cross-coupled circuit is constructed such that it provides a positive-feedback loop to store the required bit value within the loop. An example of such a circuit is a simple latch built using two cross-coupled inverters as shown in Figure 3.6.

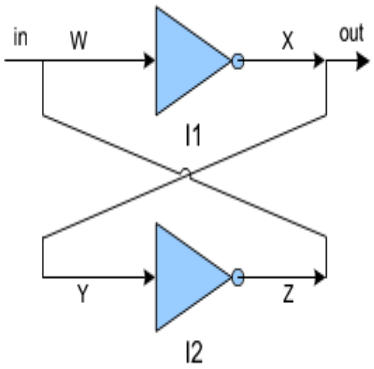


Figure 3.6. Cross coupled inverter

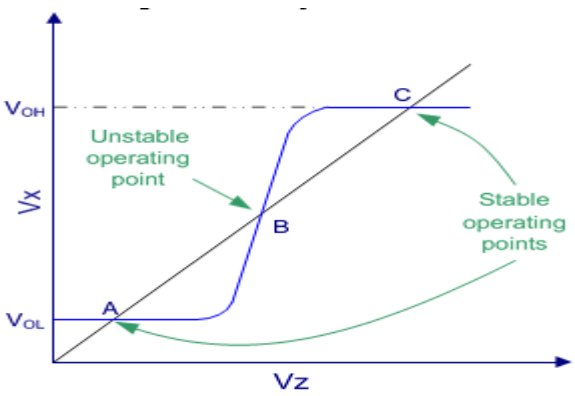


Figure 3.7. cross coupled inverter stable states

Notice that such cross-coupled circuits have two different stable operating points (to store the bit value) and an unstable operating point as shown in Figure 3.7. The circuit can be easily driven from the unstable state to a stable state by an external signal on the input or due to slight differences in the elements used to build the circuit (here inverters). We use this fact to build a PUF where the circuit is initially at the unstable operating point and left to attain one of the two stable operating points without any external excitation. We find that with high probability the circuit goes more often to one of the stable states. This behavior is due to small differences in the wire delays and cross-coupled element's (here inverter) voltage transfer characteristics. It is important to note that these circuits are constructed as symmetrically as possible and all variations are due to randomness in the circuit which is beyond the control of the designer. Different cross-coupled devices can be built using different elements like NOR gates or NAND gates.

The concept of the Butterfly PUF (BPUF) is based on the idea of creating structures within the FPGA matrix which behave similarly to a SRAM cell during the startup phase. A BPUF cell is a cross-coupled circuit which can be brought to a floating/unstable state before allowing it to settle to one of the two stable states that are possible. Implementing a cross- coupled element using combinational logic is not

straightforward due to the inability to create combinational loops. To overcome this problem, we simulate a cross-coupled combinational loop using latches. We create a cross-coupled structure using latches. This allows for an unstable state set by an excite signal, which then settles down to one of two possible stable states after some time.

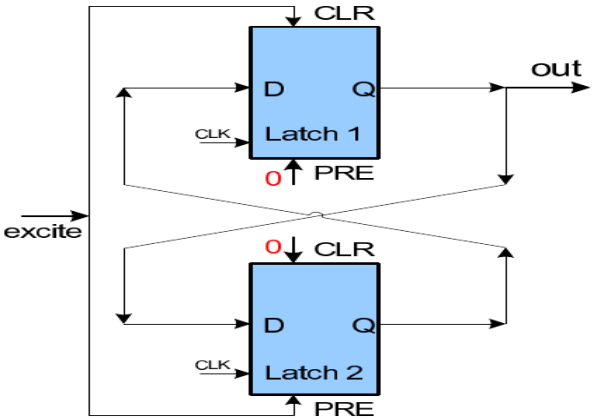


Fig3.8: Butterfly PUF

The structure of the BPUF cell is as shown in Figure 3.8 constructed *as symmetrically as possible* by manual routing of the signal wires. It consists of two latches, each with a preset (*PRE*) signal (which turns output *Q* to 1 on high) and a clear (*CLR*) signal (which turns output *Q* to 0 on high). The data *D* is transferred to the output *Q* when the *CLK* is high. In the construction, the *PRE* of *Latch* 1 and *CLR* of *Latch* 2 are always set to low. The *excite* signal is connected to *CLR* of Latch 1 and *PRE* of Latch 2. The outputs of the latch are cross-coupled. We set *CLK* in both latches to always high, effectively simulating a combinational loop. To start the PUF operation, the *excite* signal is set to high. This brings the BPUF circuit to an unstable operating point (as both latches have opposite signals on their inputs and outputs). After a few clocks cycles the excite signal is set to low. This starts the process of the PUF circuit to attain either one of the two possible stable states, 0 or 1, on the *out* signal. The stable state depends on the slight differences in the delays of the connecting wires which are designed using symmetrical path.

3.5 Ring oscillator PUF

One of the most common types of Physical Unclonable Functions (PUFs) is the ring oscillator PUF (RO-PUF), in which the output bits are obtained by comparing the oscillation frequencies of different ring oscillators. Ring Oscillator PUFs (RO-PUF) were introduced in 2007 and exploit the differences between the delay characteristics of wires and transistors. The output bits of a ROPUF are determined by comparing the oscillation frequencies of ring oscillators. RO-PUFs have a high reliability and are easier to implement compared to previously proposed designs such as butterfly PUFs. Since 2007, many researches were conducted on RO-PUFs.

Ring Oscillator PUFs (RO-PUFs) have a simple architecture made of two n-bit multiplexer, 2 counters,1 comparator and n ring oscillators (ROs). Each ring oscillator contains an odd number of inverters connected in a loop; each ring oscillates with a unique frequency depending on the characteristics of each of its inverters, which variate unpredictably from cell to cell due to manufacturing variations, even within the same chip, and are impossible to imitate. If the frequencies at which the ring oscillators oscillate are too high, the counters may not be able to count oscillations; therefore, there is a minimal number of inverters in every ring oscillator necessary to ensure a suitable oscillating frequency. The two multiplexers select two ROs which are compared together (pair). The two counter blocks count the number of oscillations of each of the two ROs in a fixed time interval (comparison time). At the end of the interval, the outputs of the two counters are compared together. Depending on which of the two counters has the highest value, the output of the PUF is set to 0 or 1. The output of the PUF is set to 0 if the first ring oscillator in the pair is faster than the second (the value of the first counter is higher than that of the second),and to 1 if it is slower (the value of the first counter is lower than that of the second). If the two frequencies are very close to each

other, the output of the PUF may variate unpredictably from run to run.

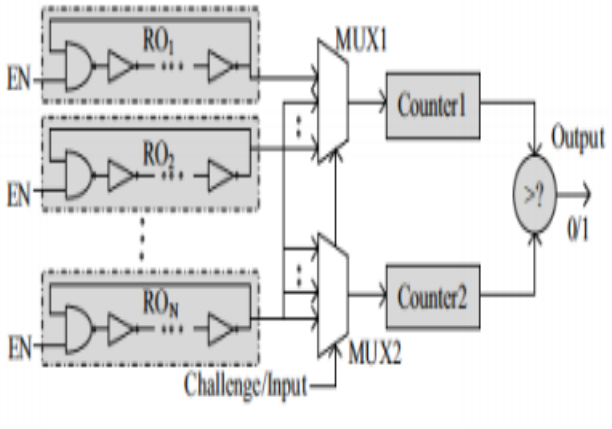


Fig3.9 conventional RO-PUF

RO-PUF can support a high number of challenge/response pairs without impacting excessively the area of the PUF. It can also be used to implement a temperature-aware RO-PUF with a 100% hardware utilization. Since all the components of RO go through significant usage, it suffers from runtime power and temperature variations and aging. This makes the RO-PUF prone to generating erroneous output.

3.6 Common metrics of the PUF:

Common metrics of the PUFs are the following:

1) Uniqueness: It is a measure of the average inter-chip. Hamming Distance THD of two strings of bits is simply the number of bits in which the strings differ. It quantifies how different is one chip from another. An ideal PUF has a uniqueness value of 50%.

2) Reliability: It is a measure of how much reliable is the CRP under noise and environmental variations. For the given challenge, the PUF should give the same response under varying operating conditions. The ideal value for reliability is 100%.

3) Randomness: It is a measure of balance between “0”s and “1”s in the response bits of the PUF and measures the randomness. The ideal value is 100% (i.e.., perfect balance).

4) Correctness: It is a measure of correctness of the response under different operating conditions. The ideal value is 100%.

5) Bit Aliasing: It is a measure of biasness of a particular response bit across several chips. The ideal value is 50%.

6) Uniformity: It is a measure of how random is the CRP. For a response to be random, the number of “0”s and “1”s in the response should occur with equal probability (i.e. 50%).

7) Steadiness: The measure of biasness of a response bit for a given number of “0”s and “1”s over a total number of samples gives the steadiness. The ideal value is 100%.

3.6.1 PUF Challenges

As illustrated, implementing a series of multiple instances of our transistor pair, or any other device, is trivial. Thus, it might seem very easy to build a PUF based on this principle. Actually, it is not!

As said in the introduction, PUF is based on the *minuscule* variations in silicon manufacturing. In our example this translates into PA > PB or PA < PB. However, because the manufacturing variations are *minuscule*, so is the difference ΔP = PA - PB. Because ΔP is small, it has to be measured with high accuracy. If not, a "0" could easily flip to "1" or vice-versa and the PUF becomes unusable for key generation. Measurement accuracy is thus a major challenge. Even worse, ΔP is generally sensitive to aging, as well as temperature, process and power-supply variations. ΔP is by nature small but also randomly distributed, hence cells having the lowest ΔP have a tendency to flip when used at different temperatures. We can consider these cells as "weak" while those having higher ΔP as "strong", the latter being less sensitive to variations. Adding extra or redundant cells as this is done in memory designs is a possible path to replace the weak cells by strong ones.

While implementing the PUF elements is relatively straightforward, getting stability over said parameters is a real challenge. There are several techniques to build stable PUFs:

- Choose the parameters (VT, IDS, RDSON) so that there easily measurable with high-accuracy;
- Redundancy: design more PUF elements than needed and eliminate the "weak" instances. Here again the number of weak cells need to be thoroughly estimated. Having not enough cells would create yield issues while adding too many redundant cells can make the PUF too large in terms of silicon area. Both would increase the actual die cost.
- Error correction: assuming the percentage of unstable cells is low enough, implementing a proper error correction mechanism such as Hamming coding would "repair" the key. The limitation is that one needs to have a pretty strong estimate of the potential defective PUF units

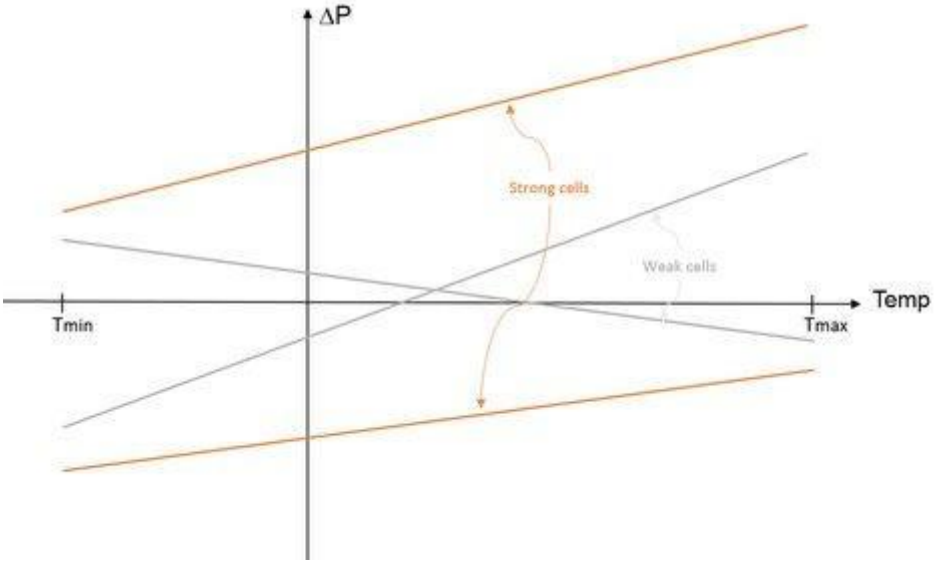


Fig3.10 Behaviour of "weak" and "strong" cells

On top of the measurement accuracy, what makes a PUF solution value are indeed the cost of efficiency of the error correction or redundancy schemes.

Reliability is essential but also as for other key generation processes, one expects for the PUF unpredictability and uniqueness. Unpredictability means that on a given die, even knowing the PUF response to a set of challenges, one cannot guess the response to the next challenge. Uniqueness is the capability for a given PUF design to generate a unique response for each die and for the same challenge F.

3.7 Applications of PUF-

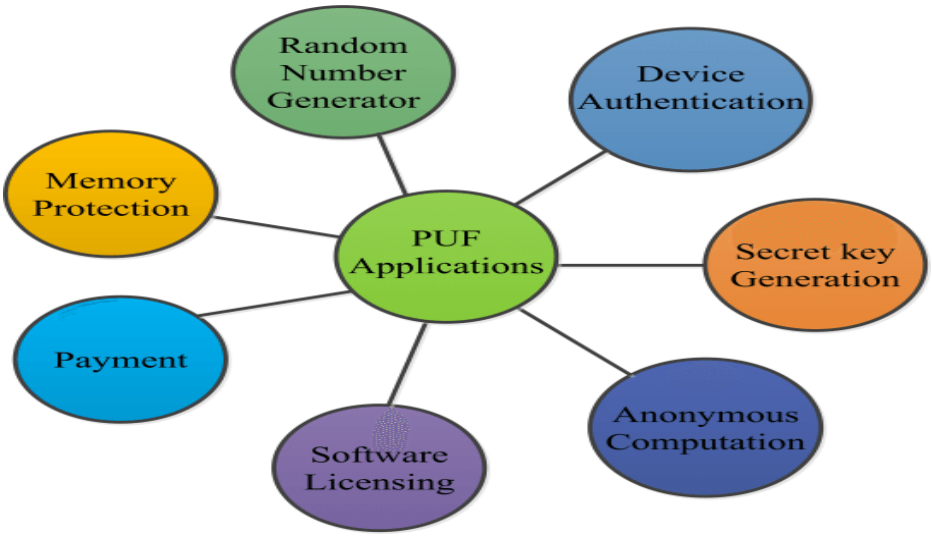


Fig 3.11 Applications of PUF

- PUFs can be used in applications that require some sort of randomness during their operation.
- PUFs seem to be an elegant solution in applications such as random number generators, Radio Frequency Identification (RFID) tags, secret key generation,and in device authentication where the required randomness property is obtained from process variation. PUFs have also been used in consumer devices for low-cost authentication purposes.
- Private and Secret Keys Storage-

The key storage is often the primary concern. The PUF generated key is used to build a secure vault within the on-chip non-volatile memory such as EEPROM, Flash or OTP.

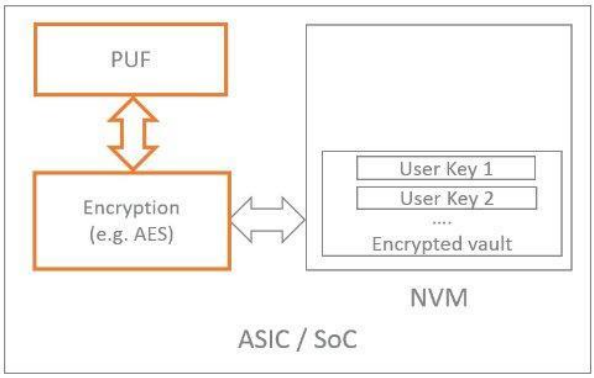


Figure 3.12 Implementing a highly secure key vault with a PUF

- Software IP protection

Some algorithms, such as those applied for medical diagnosis or vital signs measurement, are the results of years or research and development. Hence, they are extremely valuable assets deserving strong protection. PUF generated keys can protect these software IPs through encryption.

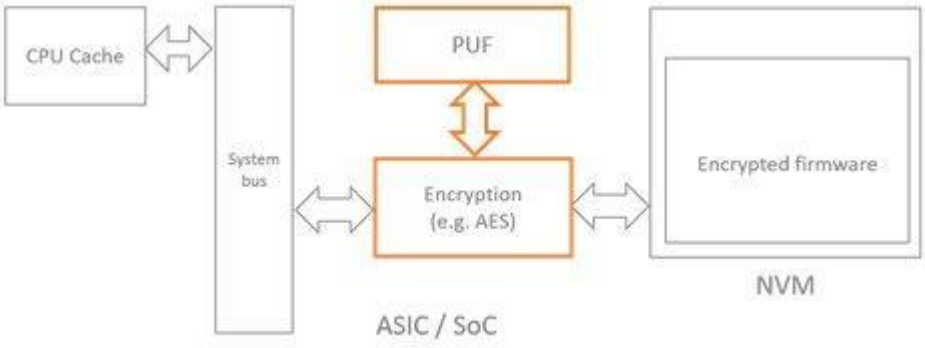


Fig 3.13 PUF based software IP protection

- Device authentication

One of the very first security requirements for connected devices is authentication, that is to say making sure the unit is genuine. The most secure way is to perform challenge – response authentication. In this scheme a random number – the challenge – is sent to the device to be authenticated and the said device signs the challenge with its private key. Here again, the private key must be strongly protected.

CHAPTER 4

VERILOG HARDWARE DESCRIPTION LANGUAGE

4.1 What is HDL?

Hardware description language (HDL) is a specialized computer language used to program electronic and digital logic circuits. The structure, operation and design of the circuits are programmable using HDL. HDL includes a textual description consisting of operators, expressions, statements, inputs and outputs. Instead of generating a computer executable file, the HDL compilers provide a gate map. The gate map obtained is then downloaded to the programming device to check the operations of the desired circuit. The language helps to describe any digital circuit in the form of structural, behavioural and gate level and it is found to be an excellent programming language for FPGAs and CPLDs. The three common HDLs are Verilog, VHDL, and SystemC.

4.2 Importance of HDLs

HDLs have many advantages compared to traditional schematic-based design.

1. Designs can be described at a very abstract level by use of HDLs. Designers can write their RTL description without choosing a specific fabrication technology. Logic synthesis tools can automatically convert the design to any fabrication technology. If a new technology emerges, designers do not need to redesign their circuit. They simply input the RTL description to the logic synthesis tool and create a new gate-level net-list, using the new fabrication technology. The logic synthesis tool will optimize the circuit in area and timing for the new technology.

2. By describing designs in HDLs, functional verification of the design can be done early in the design cycle. Since designers work at the RTL level, they can optimize and modify the RTL description until it meets the desired functionality. Most design bugs are eliminated at this point. This cuts down design cycle time significantly because the probability of hitting a functional bug at a later time in the gate-level netlist or physical layout is minimized.

3. Designing with HDLs is analogous to computer programming. A textual description with comments is an easier way to develop and debug circuits. This also provides a concise representation of the design, compared to gate-level schematics. Gate-level schematics are almost incomprehensible for very complex designs.

4.3 Introduction to Verilog HDL

Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by integrated circuit (IC) designers. The other one is VHDL. HDL's allows the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs described in HDL are technology-independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits.

Verilog can be used to describe designs at four levels of abstraction:

(i) Algorithmic level (much like c code with if, case and loop statements).
(ii) Register transfer level (RTL uses registers connected by Boolean equations).
(iii) Gate level (interconnected AND, NOR etc.).

(iv)Switch level (the switches are MOS transistors inside gates).
The language also defines constructs that can be used to control the input and output of simulation.

Verilog has a variety of constructs as part of it. All are aimed at providing a functionally tested and a verified design description for the target FPGA or ASIC. The language has a dual function – one fulfilling the need for a design description and the other fulfilling the need for verifying the design for functionality and timing constraints like propagation delay, critical path delay, slack, setup, and hold times.

Verilog as an HDL has been introduced here and its overall structure explained. A widely used development tool for simulation and synthesis has been introduced; the brief procedural explanation provided suffices to try out the Examples and Exercises in the text.

4.4 Module

Any Verilog program begins with a keyword– called a "module." A module is the name given to any system considering it as a black box with input and output terminals as shown in Figure 4.1. The terminals of the module are referred to as 'ports'. The ports attached to a module can be of three types:

- **input ports** through which one gets entry into the module; they signify the input signal terminals of the module.
- **output ports** through which one exits the module; these signify the output signal terminals of the module.
- **inout ports:** These represent ports through which one gets entry into the module or exits the module; These are terminals through which signals are input to the module sometimes; at some other times signals are output from the module through these.

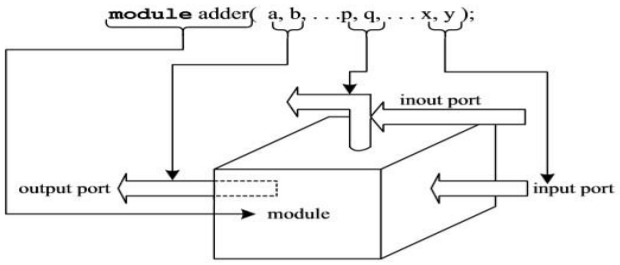


Fig 4.1: Representation of a module as black box with its ports.

4.5 Tokens of Verilog

The basic lexical conventions used by Verilog HDL are similar to those in the C programming language. Verilog contains a stream of tokens. Tokens can be comments, delimiters, numbers, strings, identifiers, and keywords.

4.5.1 Case Sensitivity

Verilog is a case-sensitive language like C. Thus sense, Sense, SENSE, sENse, etc., are all treated as different entities / quantities in Verilog.

4.5.2 Keywords

The keywords define the language constructs. A keyword signifies an activity to be carried out, initiated, or terminated. As such, a programmer cannot use a keyword for any purpose other than that it is intended for.

All keywords in Verilog are in small letters and require to be used as such (since Verilog is a case-sensitive language). All keywords appear in the text in New Courier Bold-type letters.

Examples:

module <- signifies the beginning of a module definition.

endmodule <- signifies the end of a module definition.

begin<- signifies the beginning of a block of statements.

end<- signifies the end of a block of statements.

if <- signifies a conditional activity to be checked

while<- signifies a conditional activity to be carried out.

4.5.3 Operators

Operators are of three types: unary, binary, and ternary. Unary operators precede the operand. Binary operators appear between two operands. Ternary operators have two separate operators that separate three operands.

Examples:

a = ~ b; // ~ is a unary operator. b is the operand

a = b && c; // && is a binary operator. b and c are operands

a = b ? c : d; // ?: is a ternary operator. b, c and d are operands

4.5.4 Data Types

There are two groups of types, "**net data types**" and "**variable data types**."

An identifier of "**net data type**" means that it must be driven. The value changes when the driver changes value. These identifiers basically represent wires and are used to connect components.
"net data types" are: **wire**, **supply0**, **supply1**, **tri**, **triand**, **trior**,**tri0**, **tri1**, **wand**, **wor** "net data types" can have strength modifiers: **supply0**, **supply1**,**strong0**, **strong1**, **pull0**, **pull1**, **weak0**, **weak1**,**highz0**, **highz1**, **small**, **medium**, **large**.
Some "net data types" can take modifiers: **signed**, **vectored**, scalar.

An identifier of "**variable data type**" means that it changes value upon assignment and holds its value until another assignment. This is a traditional programming language variable and is used in sequential statements.

"Variable data types" are: **integer**, **real**, **realtime**, **reg**, **time**.

integer is typically a 32-bit twos complement integer.

real is typically a 64-bit IEEE floating point number.

real time is of type **real** used for storing time as a floating-point value.

reg is by default a one-bit unsigned value.
The **reg** variable data type may have a modifier **signed**, and may have may bits by using the vector modifier msb: lsb].

Scalars and Vectors

Entities representing single bits — whether the bit is stored, changed, or transferred — are called "scalars." Often multiple lines carry signals in a cluster – like data bus, address bus, and so on. Similarly, a group of regs stores a value, which may be assigned, changed, and handled together. The collection here is treated as a "vector." Figure 4.2 illustrates the difference between a scalar and a vector. wr and rd are two scalar nets connecting two circuit blocks circuit1 and circuit2. b is a 4-bit-wide vector net connecting the same two blocks. [b0], [b1], [b2], and [b3] are the individual bits of vector b. They are "part vectors."

A vector reg or net is declared at the outset in a Verilog program and hence treated as such. The range of a vector is specified by a set of 2 digits (or expressions evaluating to a digit) with a colon in between the two. The combination is enclosed [within square brackets.

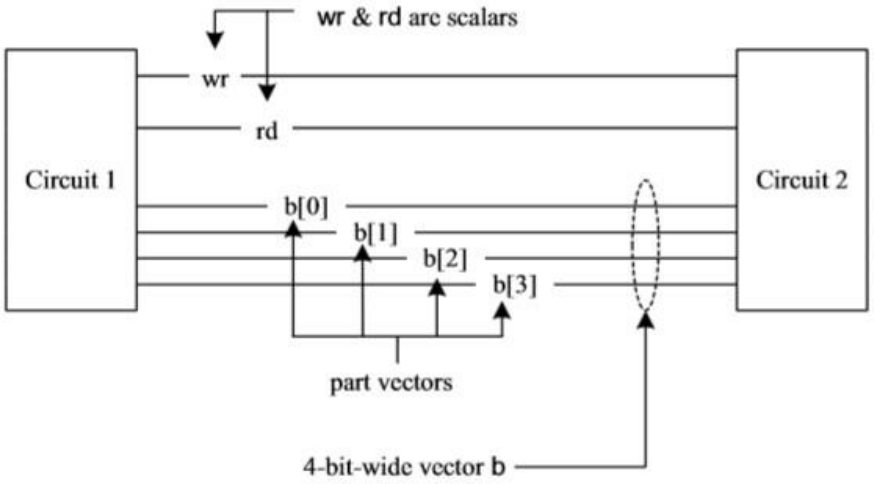


Fig 4.2: Illustration of Scalars and Vectors

Examples:

wire [3:0] a; */* a is a four bit vector of net type; the bits are designated as [a3], [a2], [a1] and [a0]. */*

reg [2:0] b; */* b is a three bit vector of reg type; the bits are designated as [b2], [b1] and [b0]. */*

reg [4:2] c; */* c is a three bit vector of reg type; the bits are designated as [c4], [c3] and [c2]. */*

wire [-2:2] d ; */* d is a 5 bit vector with individual bits designated as [d-2], [d-1], [d0], [d1] and [d2]. */*

Whenever a range is not specified for a net or a reg, the same is treated as a scalar – a single bit quantity. In the range specification of a vector the most significant bit and the least significant bit can be assigned specific integer values. These can also be expressions evaluating to integer constants – positive or negative. Normally

vectors – nets or regs – are treated as unsigned quantities. They have to be specifically declared as “signed” if so desired.

Examples:

wire signed [4:0] num; *// num is a vector in the range -16 to +15.*

reg signed [3:0] num_1; *// num_1 is a vector in the range -8 to +7.*

4.5.5 Comments

Comments can be inserted in the code for readability and documentation. There are two ways to write comments. A one-line comment starts with "//". Verilog skips from that point to the end of line. A multiple-line comment starts with "/*" and ends with "*/". Multiple-line comments cannot be nested. However, one-line comments can be embedded in multiple-line comments.

```
a = b && c; // This is a one-line comment

/* This is a multiple line     comment */

/* This is /* an illegal */ comment */

/* This is //a legal comment */
```

4.5.6 Number Specification

There are two types of number specification in Verilog they are sized and unsized.

Sized numbers

Sized numbers are represented as <size> '<base format> <number>.

<size> is written only in decimal and specifies the number of bits in the number. Legal base formats are decimal ('d or 'D), hexadecimal ('h or 'H), binary ('b or 'B) and octal ('o or 'O). The number is specified as consecutive digits from 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f. Only a subset of these digits is legal for a particular base. Uppercase letters are legal for number specification.

```
4'b1111 // This is a 4-bit   binary number

12'habc // This is a 12-bit hexadecimal number

16'd255 // This is a 16-bit decimal number.
```

Un-sized numbers

Numbers that are specified without a <base format> specification are decimal numbers by default. Numbers that are written without a <size> specification have a default number of bits that is simulator- and machine-specific (must be at least 32).

```
 23456 // This is a 32-bit   decimal number by default

'hc3 // This is a 32-bit   hexadecimal number
```

'o21 // This is a 32-bit octal number

X or Z values

Verilog has two symbols for unknown and high impedance values. These values are very important for modelling real circuits. An unknown value is denoted by an x. A high impedance value is denoted by z. 12'h13x // This is a 12-bit hex number; 4 least significant bits unknown

6'hx // This is a 6-bit hex number

32'bz // This is a 32-bit high impedance number

An X or Z sets four bits for a number in the hexadecimal base, three bits for a number in the octal base, and one bit for a number in the binary base. If the most significant bit of a number is 0, X, or Z, the number is automatically extended to fill the most significant bits, respectively, with 0, X, or Z. This makes it easy to assign X or Z to whole vector. If the most significant digit is 1, then it is also zero extended.

Negative numbers

Negative numbers can be specified by putting a minus sign before the size for a constant number. Size constants are always positive. It is illegal to have a minus sign between <base format> and <number>. An optional signed specifier can be added for signed arithmetic.

-6'd3 // 8-bit negative number stored as 2's complement of 3

-6'sd3 // Used for performing signed integer math

4'd-2 // Illegal specification

4.6 Module Declaration:

Modules are the building blocks of Verilog designs. A module can be an element or a collection of lower-level design blocks. A module provides the necessary functionality to the higher-level block through its port interface (inputs and outputs), but hides the internal implementation. Module interface refers, how module communicates with external world. This communication is possible through different ports such as input, output and bi-directional (inout) ports. Design functionality is implemented inside module, after port declaration. In Verilog, a module is declared by the keyword module. A corresponding keyword endmodule must appear at the end of the module definition. Each module must have a module_name, which is the identifier for the module, and a port list, which describes the input and output terminals of the module. Design functionality is implemented inside module, after port declaration. The design functionality implementation part is represented as “body” here.

Syntax

module

```
module name(port-list);
input [msb:lsb] input_port_list;
```

output [msb:lsb] output_port_list;
inout [msb:lsb] inout_port_list;
……statements…… **endmodule**

NOTE: All module declarations must begin with the **module** (or **macro-module**) keyword and end with the **endmodule** keyword. After the module declaration, an identifier is required. A ports list is an option. After that, ports declaration is given with declarations of the direction of ports and the optionally type. The body of module can be any of the following:

- Any declaration including parameter, function, task, event or any variable declaration.
- Continuous assignment.
- Gate, UDP or module instantiation.
- Specify block.
- Initial block
- Always block.

If there is no instantiation inside the module, it will be treated as a top-level module.

Example:

```
module module_1(a,b,c) ;
parameter size = 3 ;
input size : 0] a, b ;
output size : 0] c;
assign c = a &b;
endmodule
```

```
module top;
reg data, clock;
wire q_out, net_1;
dff inst_1 (.d(data), .q(net_1), .clk(clock));
dff inst_2 (.clk(clock), .d(net_1), .q(q_out));

 endmodule
```

In the top module there are two instantiations of the 'dff' module. In both cases port connections are done by name, so the port order is insignificant. The first port is input port 'd', the second is output 'q' and the last is the clock in the 'inst_1'. In the dff module the order of ports is different than either of the two instantiations.

Example 2

```
module dff (clk, d, q);
input clk, d; output q;
reg q;
always @(posedge clk) q = d;
endmodule
```

```
module top;
reg  data,  clock;
wire q_out, net_1;
dff inst_1 (clock, data, net_1);
dff inst_2 (clock, net_1, q_out);
```

endmodule

Example 3

dff inst_1 (clock, , net_1);

Second port is unconnected and has the value Z because it is of the net type.

Example 4

```
module my_module (a, b, c);
input     a,     b;
output  c;  assign
c  =  a  &  b  ;
endmodule

module  top  (a,  b,  c)  ;
input    [3:0]    a,    b;
output[3:0] c;
  my_module inst 3:0] (a, b, c);

endmodule
```

4.7 Flowchart of Verilog Code

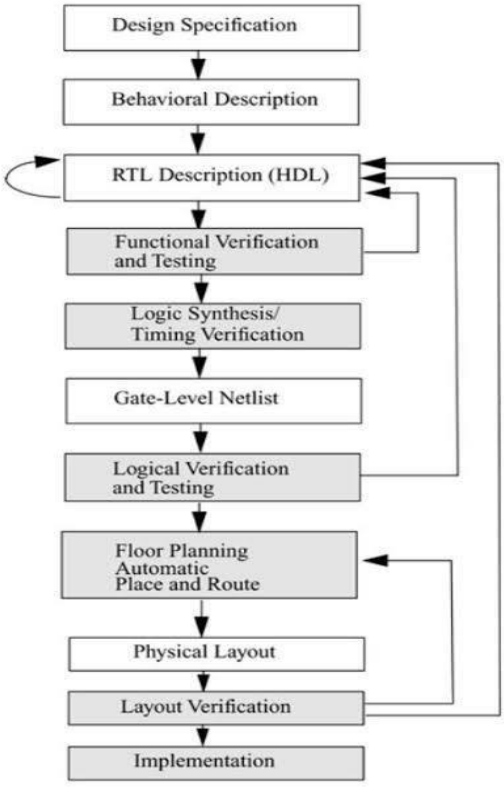


Fig 4.3: Flowchart representation of Verilog Code

Verilog has four levels of Modeling:

1) The switch level Modeling.
2) Gate-level Modeling.
3) The Data-Flow level.
4) The Behavioural Procedure

1. Switch level Modeling: A circuit is defined by explicitly showing how to construct it using transistors like pmos and nmos, pre-defined modules.

Example:

```
module inverter (out, in);

output out;

input in;
supply0 gnd;
supply1 vdd;
nmosx1 (out, in,gnd);
pmosx2(out, in, vdd);
endmodule
```

2. Gate level modeling: A circuit is defined by explicitly showing how to correct it using logic gates, predefined modules, and the connections between them. In this first we think of our circuit as a box or module which is encapsulated from its outer environment, in such a way that its only communication with the outer environment is through input and output ports. We then set out to describe the structure within the module by explicitly describing its gates and sub modules, and how they connect with one another as well as to the module ports. In other words, structural modeling is used to draw a schematic diagram for the circuit. As an example, consider the fulladder below.

Example:

```
module fulladder (a, b, sum, C out);

Input a, b;
output sum, C out;
xor x1(a, b, y);
xor x2(a, b, y);
endmodule
```

3. Data-flow modeling: Dataflow modelling uses Boolean expressions and operators. In this we use assign statement.

Example :

```
module fulladder (a, b, sum, C out);
input a, b;
output sum, C out;
assign sum=a^b;
assign Cout =a^b;
endmodule
```

4. Behavioural modeling: It is higher level of modeling where behaviour of logic is modelled. Verilog behavioural code is inside procedure blocks, but there is an exception: some behavioural code also exist outside procedure blocks.

There are two types of procedural blocks in Verilog :

Initial: initial blocks execute only once at time zero(start execution at time zero)

Always: always blocks loop to execute over and over again; in other words, as the name suggests, it executes always.

An always statement executes repeatedly, it starts and its execution at 0ns

Syntax:

```
always@ (sensitivitylist)
begin
--Procedural statements-- end
```

Example:

```
module fulladder (a, b, clk,sum);
input a, b, clk;
output sum;
always@ (posedgeclk)
begin
sum =a+b;
end
endmodule
```

4.8 Software Tools Used

Xilinx Vivado 2019.2

Few important terminologies are tasks and functions. They are described below.

Tasks:

Tasks are used in all programming languages, generally known as procedures or subroutines. The lines of code are enclosed in task....end task brackets. Data is passed to the task, the processing done, and the result returned. They have to be specifically called, with data ins and outs, rather than just wired in to the general netlist. Included in the main body of code, they can be called many times, reducing code repetition.

- tasks are defined in the module in which they are used. It is possible to define a task in a separate file and use the compile directive 'include to include the task in the file which instantiates the task.
- tasks can include timing delays, like posedge, negedge, # delay and wait.
- tasks can have any number of inputs and outputs.
- The variables declared within the task are local to that task. The order of declaration within the task defines how the variables passed to the task by the caller are used.

- tasks can take, drive and source global variables, when no local variables are used. When local variables are used, basically output is assigned only at the end of task execution.
- tasks can call another task or function.
- tasks can be used for modeling both combinational and sequential logic.
- A task must be specifically called with a statement; it cannot be used within an expression as a function can.

Functions:

A Verilog HDL function is the same as a task, with very little differences, like function cannot drive more than one output, cannot contain delays.

- functions are defined in the module in which they are used. It is possible to define functions in separate files and use compile directive 'include to include the function in the file which instantiates the task.
- functions cannot include timing delays, like posedge, negedge, # delay, which means that functions should be executed in "zero" time delay.
- functions can have any number of inputs but only one output.
- The variables declared within the function are local to that function. The order of declaration within the function defines how the variables passed to the function by the caller are used.
- functions can take, drive, and source global variables, when no local variables are used. When local variables are used, basically output is assigned only at the end of function execution.
- functions can be used for modeling combinational logic.
- functions can call other functions, but cannot call tasks.

Steps to be followed to Create a Project in Xilinx Vivado 2019.2

- First open Xilinx Vivado 2019.2 then the Fig 4.4 appears on the screen.
- Click on **Create Project** to create a new project.

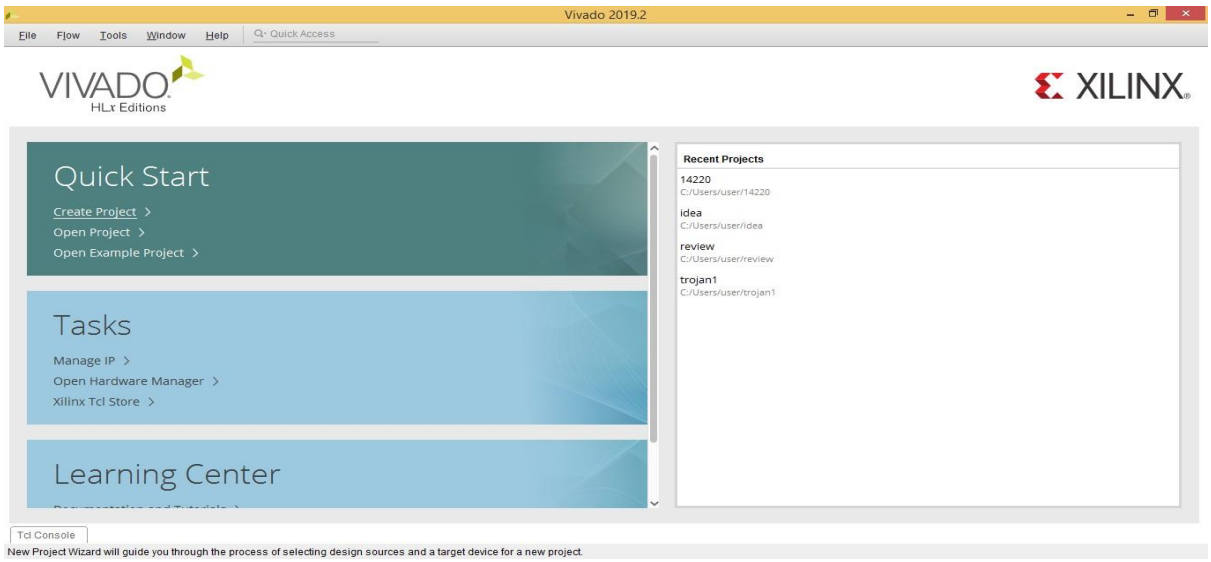


Fig 4.4 Opening window of Xilinx 2019.2

- Now a new window appears as shown in Fig 4.5

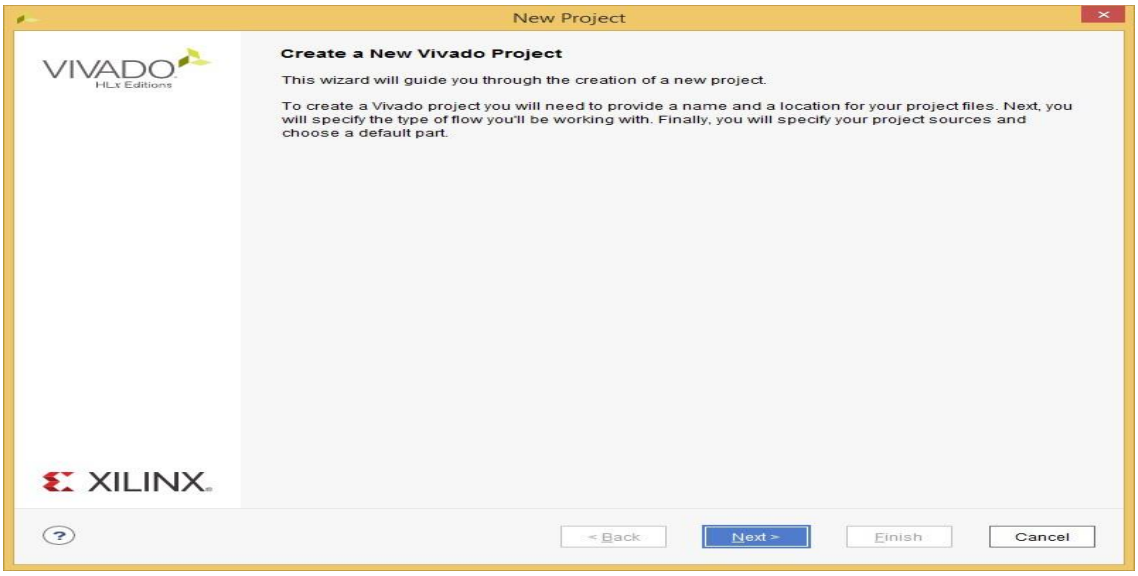


Fig:4.5: Create project window

- Click on **NEXT** to move further fig 4.6 shows the name and location of the project.

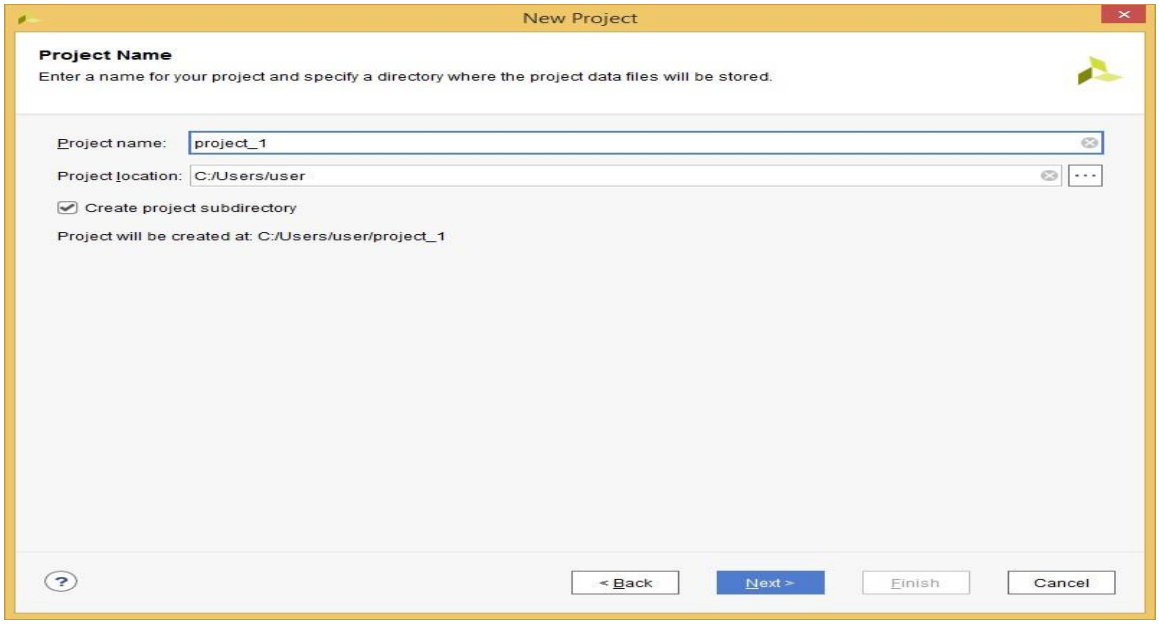


Fig 4.6: Name and Location entry for project

- Select Type of the project fig 4.7 shows type of project.

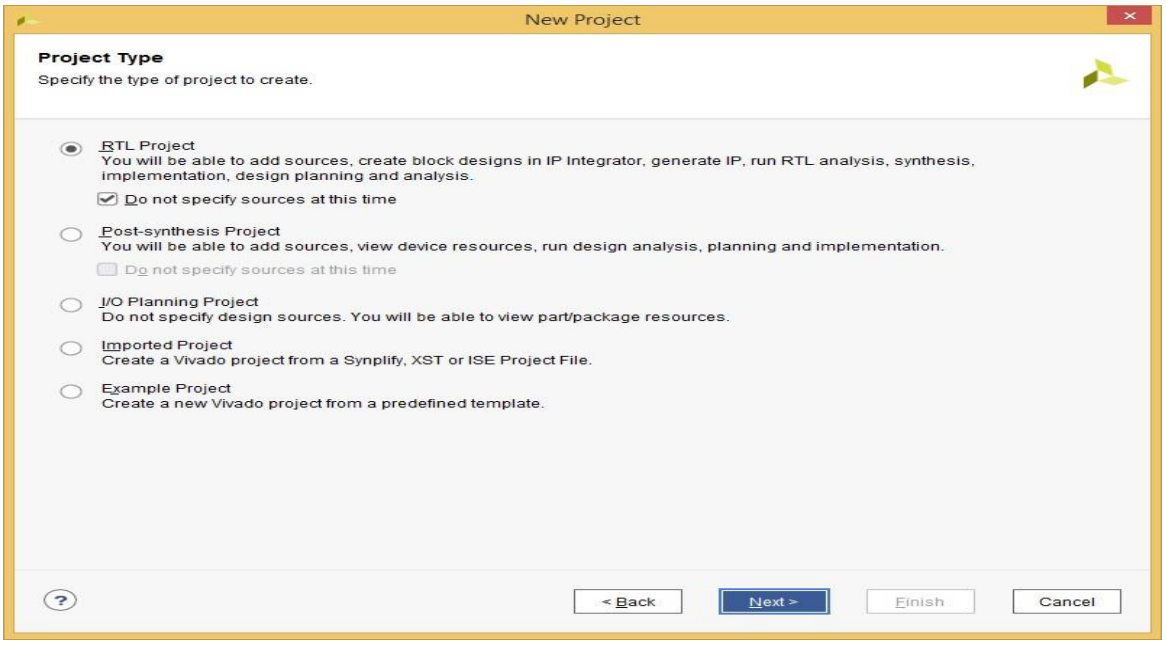


Fig 4.7 Selecting type of the project

• Now select specifications for the required project. Fig 4.8 shows Specifications of the project.

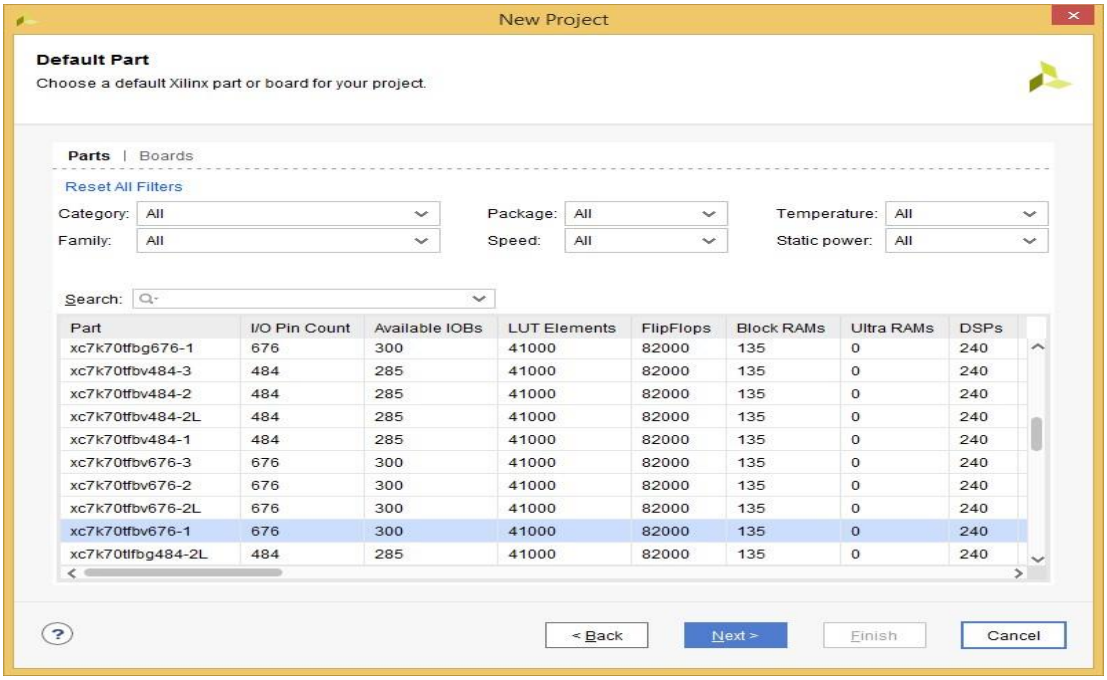


Fig 4.8.: Specifications window for the project

• Fig 4.9 shows summary of the project.

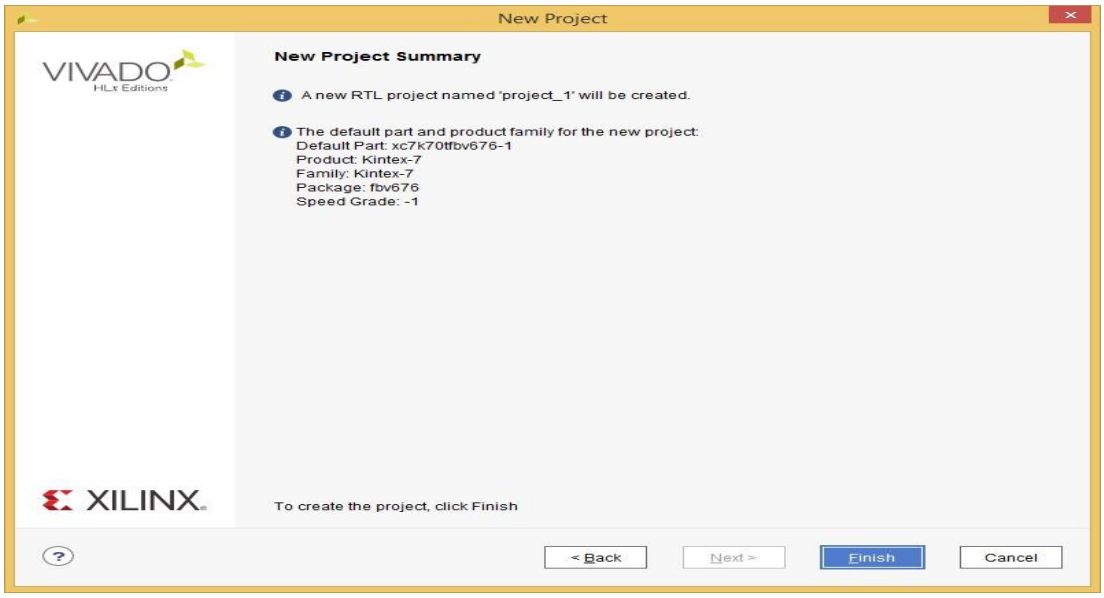


Fig 4.9: Summary window of the project

• Sources can be added upon clicking Add Sources. Fig 4.10 appears after following the above steps.

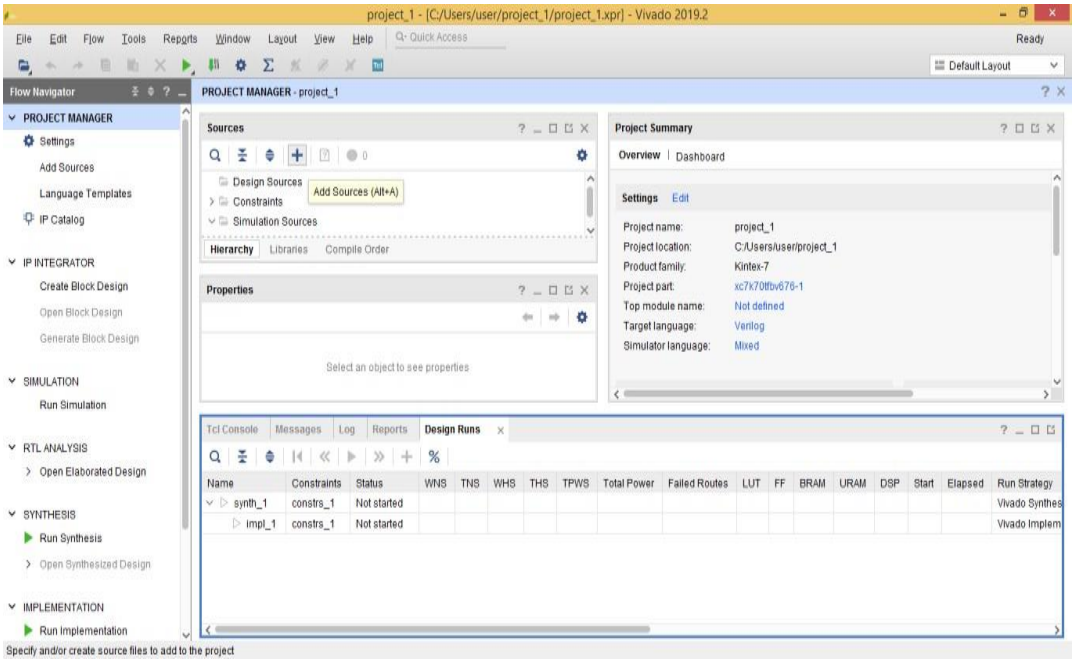


Fig 4.10: Add sources

• Sources are added to the project by clicking on add or create design and a Verilog file name.

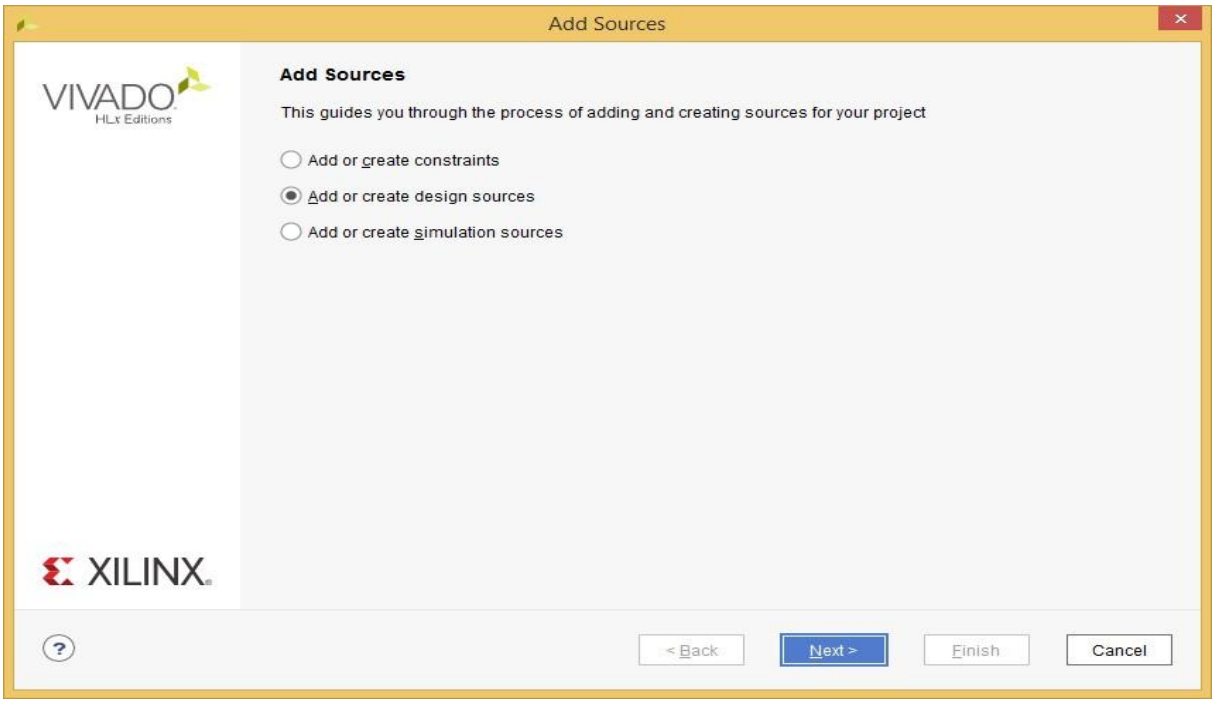


Fig4.11: Creating source

• Create Verilog files to write code. Fig 4.8.9 and 4.8.10 deals with creating a Verilog file for writing the code.

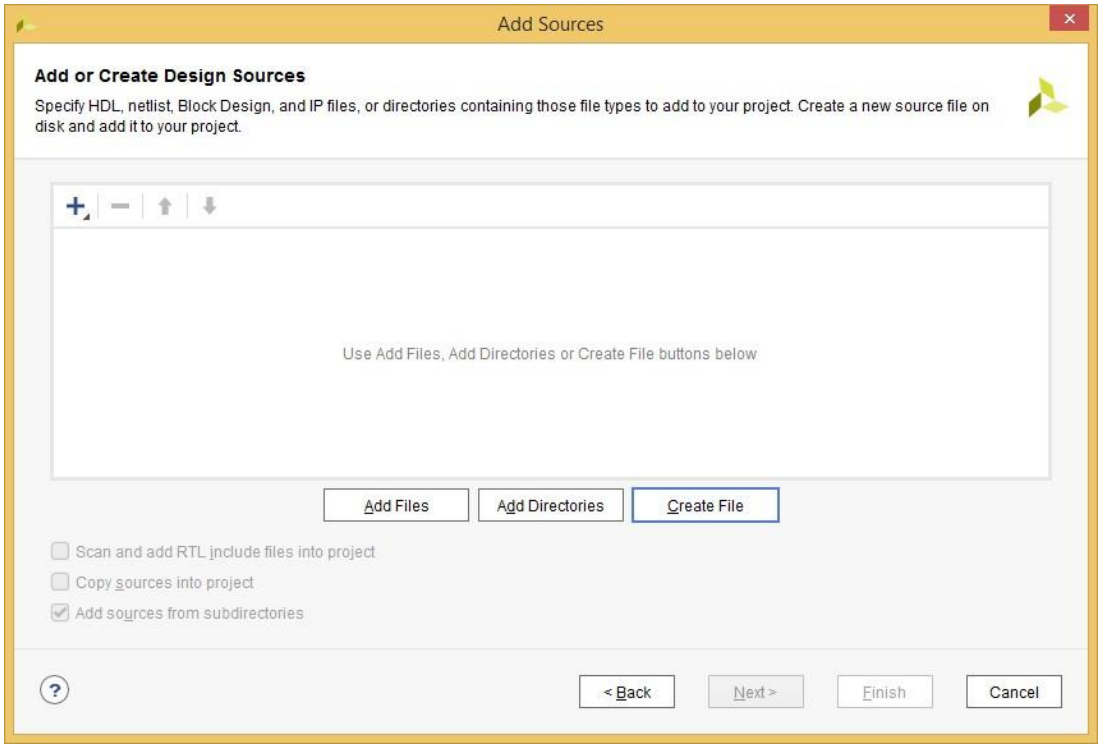


Fig 4.12: Creating file window

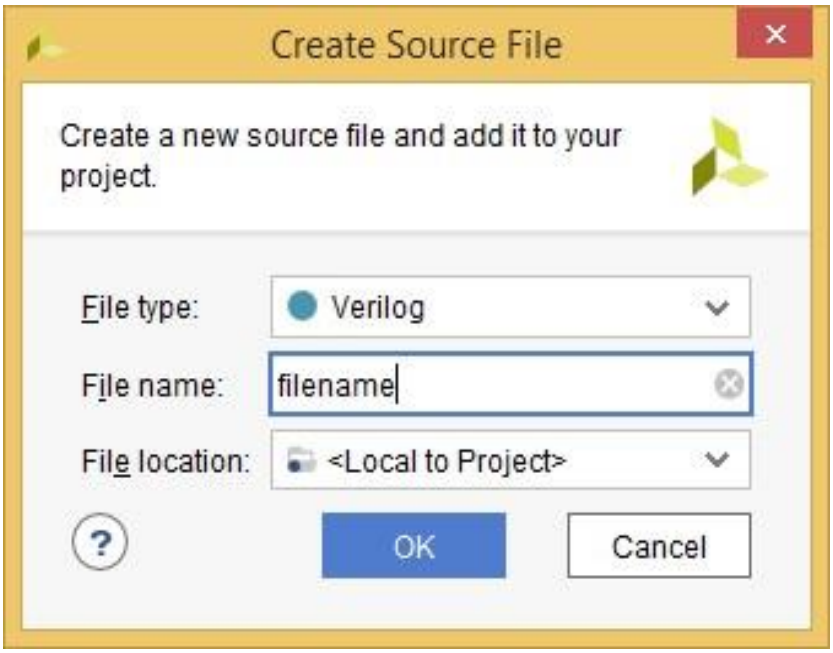


Fig 4.13: Filename creation window

•Sources will be shown in a new window as in fig 4.14

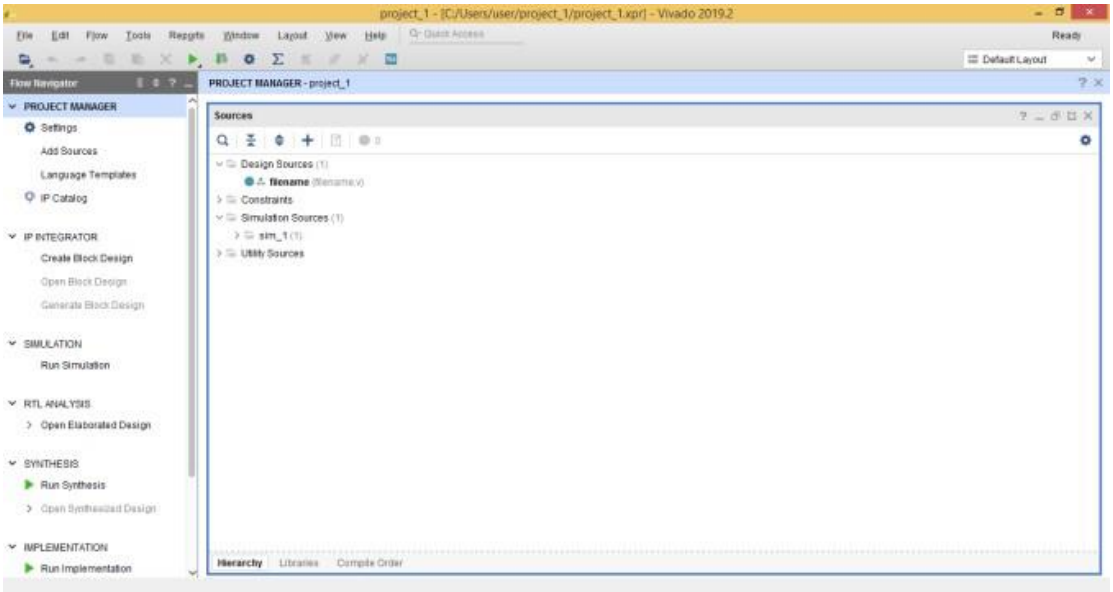


Fig 4.14: Sources window

• Simulation, Synthesis and implementation can be done by using the side bar options as shown in fig 4.15

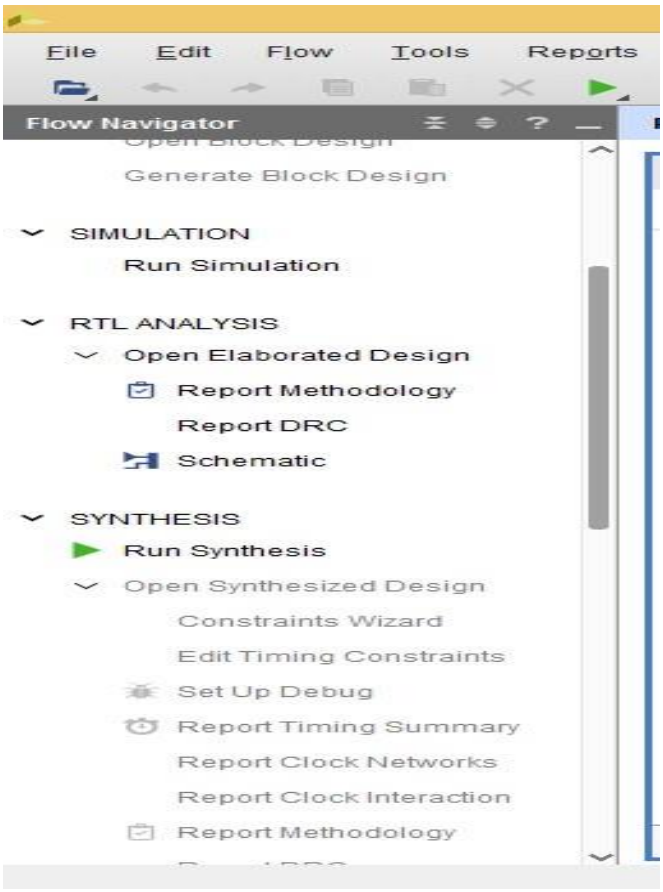


Fig 4.15 Sidebar for performing required process.

CHAPTER 5

HARDWARE TROJAN INSERTION AND DETECTION

5.1 Hardware Trojan

A hardware trojan can be described as a malicious alteration or inclusion to an integrated circuit (IC) that will either alter its intended function or cause it to perform an additional malicious function. These malicious inclusions or alterations are generally programmed to activate only under a specific set of circumstances created by an attacker and are extremely hard to detect when in their dormant state. As technology advances, so does the demand for IC boards leaving many technology companies without the resources to produce secure enough ICs to meet current demands. This has pushed companies into the 'fabless' trend prevalent in today's semi-conductor industry, where companies are no longer attempting to produce the goods in their own factories, but instead are outsourcing the process to cheaper factories abroad. This growth brings with it a significant rise in the level of threat posed by hardware trojans, a threat that directly affects all companies concerned with products that utilise ICs. This encompasses many different industries, including the military and telecommunications companies, and can potentially affect billions of devices from mobile phones and computers to military grade aviation and detection devices, particularly at a time when wireless devices are being introduced as links in critical infrastructure, compounding trust and security issues even further.

5.2 Algorithm for Trojan Detection

The following steps are to be followed for detecting an hardware trojan.

1. Get the circuit under test
2. Retrieve the path delays and other parameters
3. i=1
4. Retrieve ith path
5. Give test vector to circuit under test
6. Measure the path delay and other parameters
7. If delay equal to golden circuit's delay then go to '**a'** else go to **'b'.**

a. i=i+1

if i=number of available paths then go to **'f1**' else go to **step-4**.

f1. **Circuit is trojan free.**

b. ***Circuit is affected.***

5.3. Side Channel Analysis

Logic-based testing may not be effective for activating large combinational or sequential Trojans due to the extremely large number of possible trigger nodes. Side-channel analysis involves the measurement and analysis of information obtained from an IC's side-channels. The information could be based on timing, power

consumption, or electromagnetic radiation. Side-channel analysis has been proposed previously as a powerful technique to detect malicious insertions in an IC. In this section, we specifically concentrate on side-channel information obtained from power consumption in the device. Static power contains comprehensive information regarding all the gates in the chip (including malicious gates/ transistors). Trojans causing physical damage by creating electrical conflicts can also be detected using side-channel analysis since these Trojans result in a large current flow through the power supply. A simple design file can be loaded that configures I/O ports as inputs and then measures the supply current. If these Trojans simultaneously try to configure the port as an output, then a very large current can be detected by current sensors in the device, indicating a malicious modification. Since on-chip current sensors may also be tampered in the foundry during production, they must be tested thoroughly to identify any tampering. An alternative and secure strategy would be to use an on-board current sensor to detect short-circuit conditions. Trojans which do not cause physical damage and only cause logical malfunction may be extremely difficult to detect by analysing static power. This is due to the difficulty in isolating the contribution of the malicious insertions from the measured power traces in ICs containing many millions of transistors. On the other hand, transient or dynamic power can be controlled by applying input vectors to reveal information about a few gates which are switching at any given time. The advantage of this type of analysis is that, unlike logic-based testing, a Trojan does not have to become active for detection; it merely needs to cause switching in the Trojan to consume dynamic power. For the IP-independent Trojans presented in Section 3, transient power analysis can be an effective detection method. For example, a counter-based Trojan inserted in the clock manager module can be detected by applying a clock signal to the FPGA and applying constant inputs to prevent logic blocks from switching. An extraneous counter or any sequential circuit will consume transient power as it transitions from one state to another. This contribution to dynamic power can be identified and associated with malicious insertions after accounting for process noise and clock coupling power.

5.4 Trojan Implementation :

Among those 4 types of trojans, we have implemented hardware trojan procedure in three ways:

(1) Trojan by short circuiting input line: Trojan is inserted/created by short circuiting the inputs of the IC. In this way, that for certain test vectors/inputs the output will be faulty. At that point the circuit is broke down and contrasted and the golden circuit to distinguish the trojan.

(2) Combinational trojan: This trojan is combinational sort of trojan. In this, we will interface the combinational circuit as the hardware trojan alongside the Trusted IC's unique circuit i.e.,256:1 mux.

(3) Delay type of trojan: Trojan circuit is created by introducing some delay in 256:1 MUX. In this trojan implementation, until count 252 we will get the same 256:1 mux output, yet, after check 252 we will get defective output. Since a trojan block is added to the 256:1 mux block. The variations between the brilliant circuit and trojan circuit have been investigated.

5.5 Simulation Results

5.5.1 Trojan Free Circuit

Here A 256:1 MUX is taken as the trojan free circuit and then the circuit is implemented by writing a VHDL code in Vivado Xilinx 2019. The algorithm stated earlier in chapter 4 should be followed in achieving this.

Schematic of the Trojan free circuit i.e.,256:1 MUX shown in fig 5.1 is obtained after synthesis is done. The LUTs and other components can be observed in the schematic shown in fig 5.1.

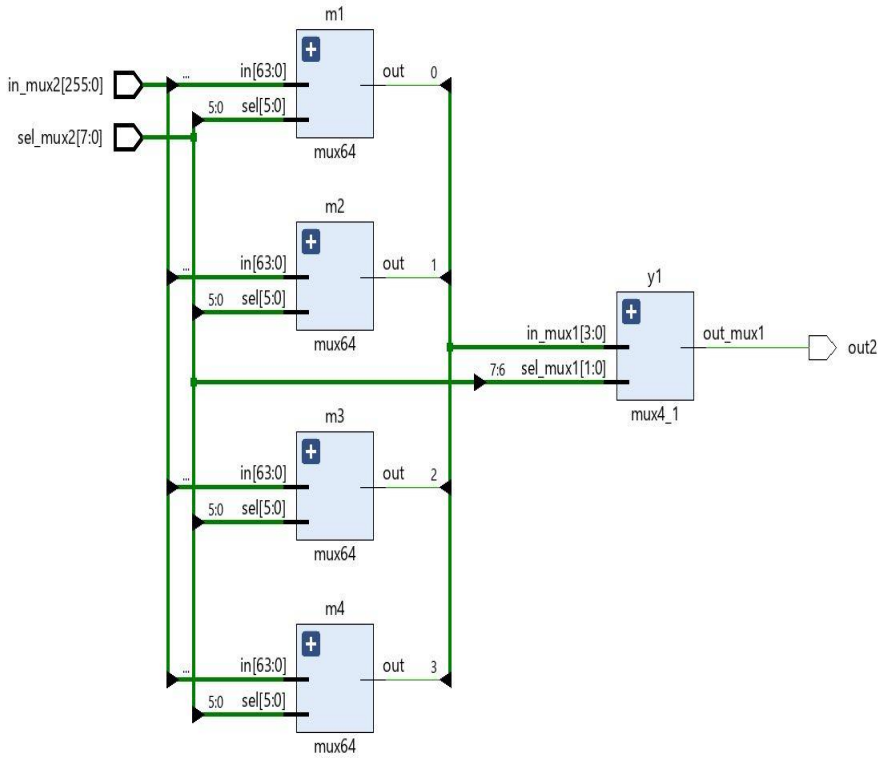


Fig 5.1: Schematic diagram for the Trojan Free Circuit

The floor planning for the Trojan free circuit i.e.,256:1 MUX is shown in fig 5.2

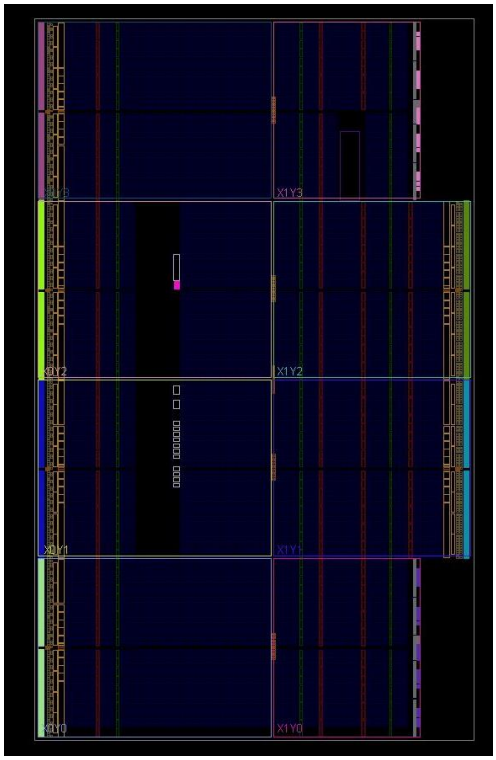


Fig 5.2: Floor planning for trojan free circuit

The simulation results in fig 5.3 shows the normal functioning circuit which is trojan free circuit.

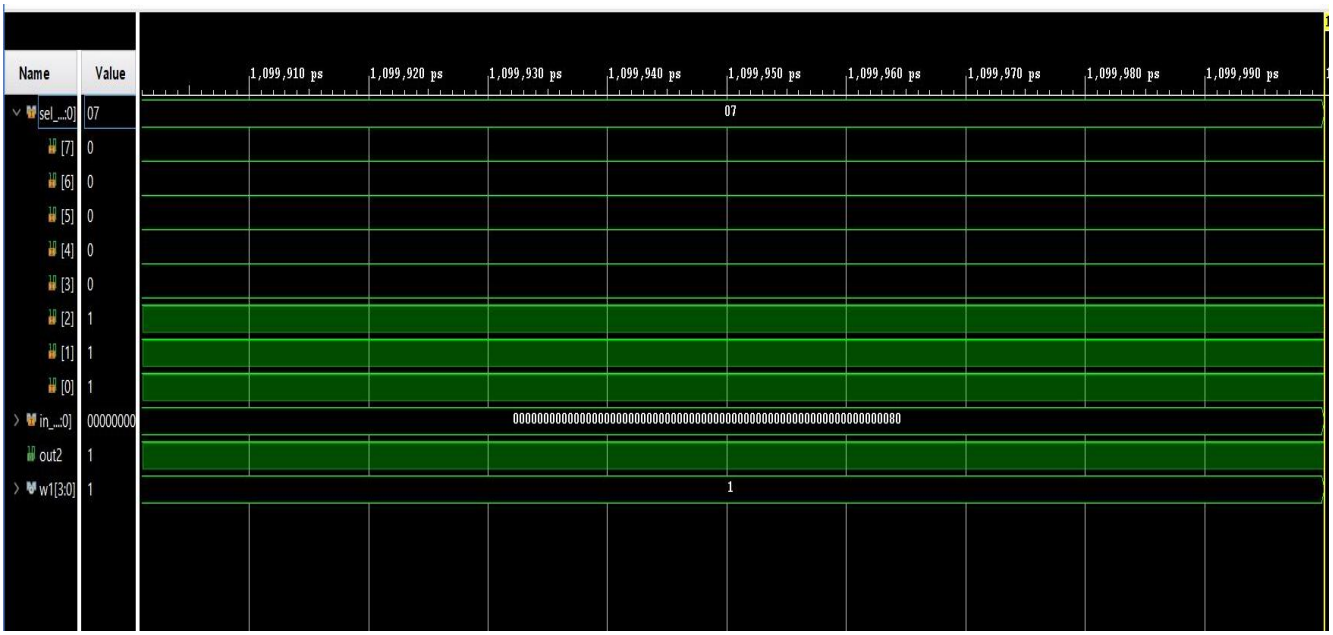


Fig 5.3 Simulation Results of trojan free circuit

In this Trojan free output, we are having 8 selection lines i.e. [0:7]. Here we have selected selection line as 00000111 i.e., 7 unsigned decimal, by giving the input as 255 decimal. So the 7th bit of the input is checked because selection lines input is 7. Hence output is HIGH since 7th bit is HIGH.

5.5.2 Delay Type of Trojan Inserted Circuit

A 256:1 MUX is taken as the trojan free circuit. The following trojan free circuit is taken and without changing the propagation delay the Delay Trojan was inserted in lowest path delay thereby we can detect the Trojan using path delays then the circuit is implemented by writing a VHDL code in Vivado Xilinx 2019. The algorithm stated earlier in chapter 4 should be followed in achieving this.

Here is how a Delay Trojan is inserted in Trojan free circuit i.e.,256:1 MUX shown in fig 5.1 is shown as in fig 5.4

1. The implementation of 64:1 multiplexer using 8:1 mux has been done.
2. The implementation of 256:1 multiplexer using 64:1 mux and 4:1 mux has been done.
3. We have analyzed the variations in golden circuit and trojan circuit. Trojan circuit is created by inserting the delay in 256 MUX. In this trojan implementation, until count < 252 we will get the same 256_1 mux output, but after count >= 252 we will get faulty output. Because a trojan block is added to the 256_1 mux block.

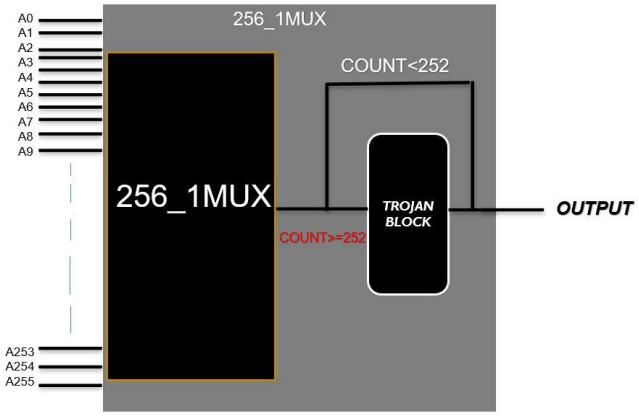


Fig 5.4 : Delay Trojan inserted circuit i.e., Circuit Under Test

The simulation results in fig 5.5 shows the normal functioning circuit which is trojan free circuit.

In this Trojan output, we are having 8 selection lines i.e. [0:7]. Here we have selected selection line as 00000111 i.e., 7 unsigned decimal, by giving the input as 255 decimal. But while comparing to trojan free output here output should be HIGH actually but the output is not HIGH due to the trojan circuit is added to it. Hence, we get the faulty output. Due to the reason that a net variable cannot be given non-net variable there had been some problems in the writing the code and synthesizing the code. So since the circuit cannot be synthesized the schematic diagram and the floor planning of the circuit cannot be obtained. So we had done another type of hardware trojan insertion which is Combinational type of Hardware Trojan.

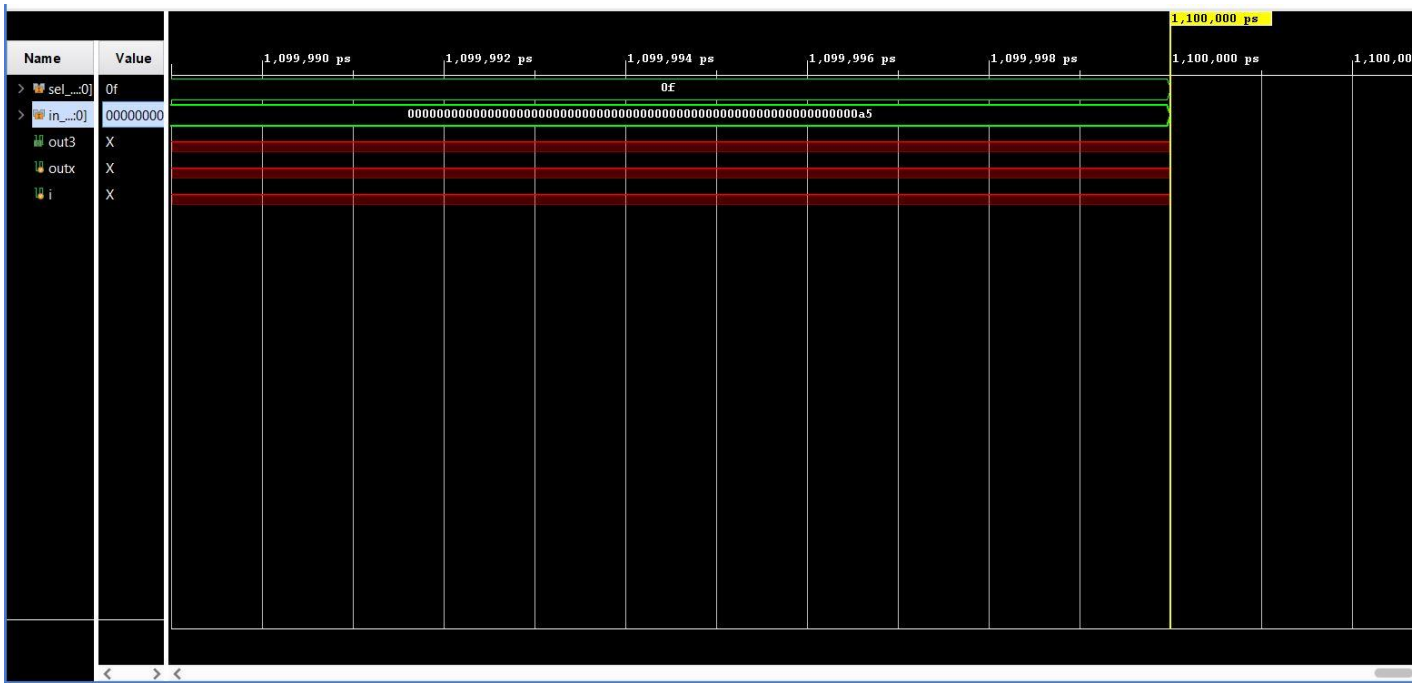


Fig 5.5 Simulation Results of trojan free circuit

5.5.3 Combinational Trojan Inserted Circuit

A 256:1 MUX is taken as the trojan free circuit. The following trojan free circuit is taken and without changing the propagation delay the Delay Trojan was inserted in lowest path delay thereby we can detect the Trojan using path delays then the circuit is implemented by writing a VHDL code in Vivado Xilinx 2019. The algorithm stated earlier in chapter 4 should be followed in achieving this.

Schematic of the Combinational Trojan inserted circuit shown in fig 5.6 is obtained after synthesis is done. The LUTs and other components can be observed in the schematic shown in fig 5.6.

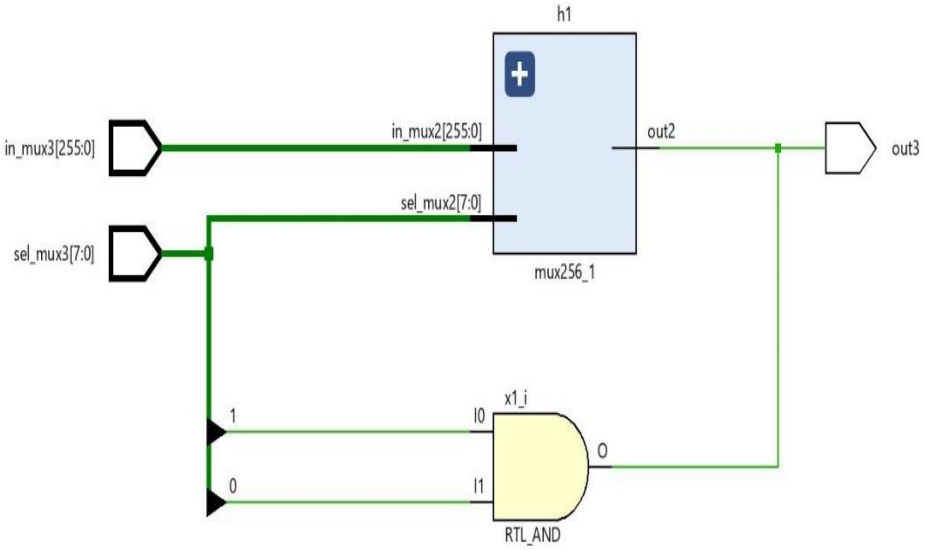


Fig 5.6: Schematic diagram for the Combinational Trojan Inserted Circuit

The floor planning for the the Combinational Trojan Inserted Circuit is shown in fig 5.7

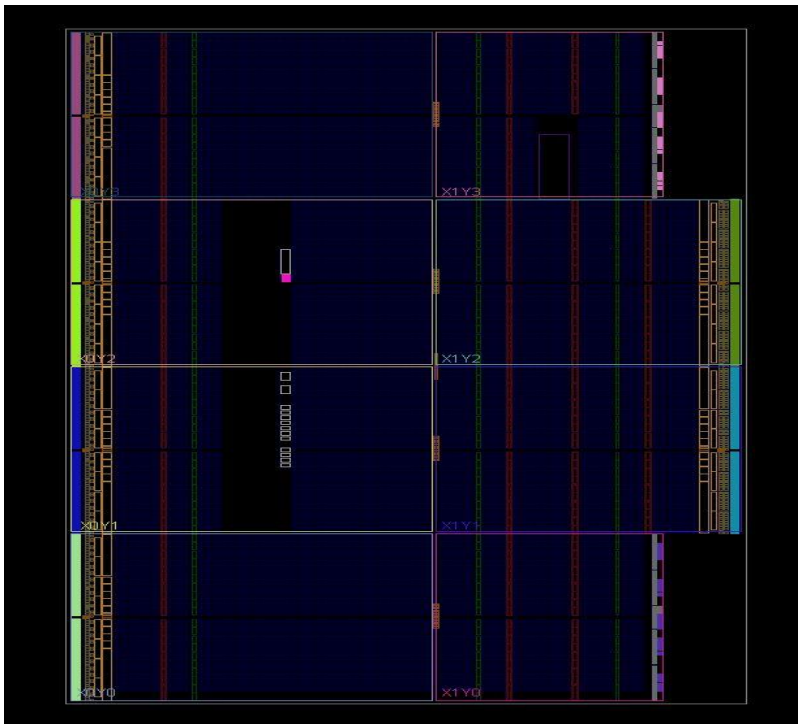


Fig 5.7: Floor planning for the Combinational Trojan Inserted Circuit

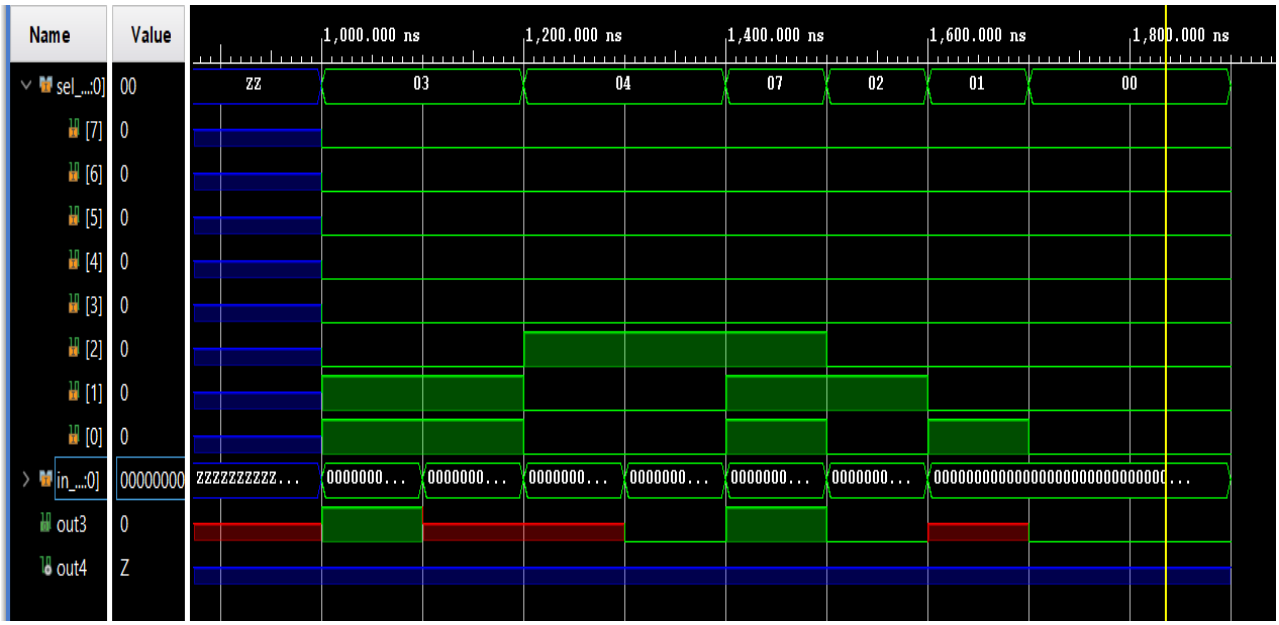


Fig 5.8: Simulation Results of trojan free circuit

The simulation results in fig 5.8 shows the normal functioning circuit which is trojan free circuit.

Here we had taken the selection line inputs sel_mux3[0], sel_mux3[1], these lines are given to the input of AND gate whose output is given to the final output of the IC. So, if the actual output of the 256:1 MUX and the output of the AND gate are equal then there is no contention but if the output of the AND gate and the 256:1 MUX are equal then there occurs a contention. This contention can also be resolved using wand or wor logic. Hence for some test vectors the output will be faulty which results in the IC which cannot be trusted i.e., trojan inserted circuit.

Table IV: Parameters comparison of Trojan Free and Trojan Effected Circuit

Trojan Free Circuit	**Trojan Effected Circuit**
LUTs utilized are 68 LUTS AS LOGIC: 68 LUT6 : 66 LUT2 : 1 LUT3 :1	LUTs utilized are: 70 LUTs AS LOGIC: 70 LUT 6 : 68 LUT 2 : 1 LUT 3 :1
Slice Registers utilized are 45	Slice Registers utilized are 45
Registers as flip-flops are 40	Registers as flip-flops are 40
BUFGCTRL are 15	BUFGCTRL are 20
Total Onchip Power: 2.939 W	Total Onchip Power: 3.098W
Junction Temperature: 38.2	Junction Temperature: 39.1
F7 MUXES : 34 F8 MUXES : 17 Bonded IOB :265	F7 MUXES : 38 F8 MUXES : 16 Bonded IOB :267

From the table 5.1 we can observe the performance of both the trojan affected and trojan free circuits in the parameters like LUTs, Slice registers, IOBs, Registers as flip-flops, BUFGCTRL, Onchip temperature, F7, F8 MUXes and Junction Temperature

The components like LUTs, IOBs, Onchip temperature, F7 MUXes, Junction Temperature and BUFGCTRLs are increased in trojan inserted circuit.

Few components like Slice Registers and Registers as flip flops remained same. From this observation we can extract that the number of memory elements utilized are same in both trojan free and trojan affected circuits.

5.5.4 Butterfly PUF :

A Butterfly PUF cell is a cross-coupled bistable circuit, as shown in fig 5.9 which can be brought to an unstable state before it settles to one of the two stable states that are possible. It is made of two flipflops whose outputs are cross-coupled. Forcing the excite signal high brings the system to unsteady state. When the excite signal is set low, after certain clock cycles BPUF arrives at any of the consistent states i.e., 1 or 0. The interfacing wires defers drives the output. As the attributes of the circuit relies upon inborn properties, so the output is unpredictable. The main property of this circuit to construct a BPUF is its unpredictable state of output.

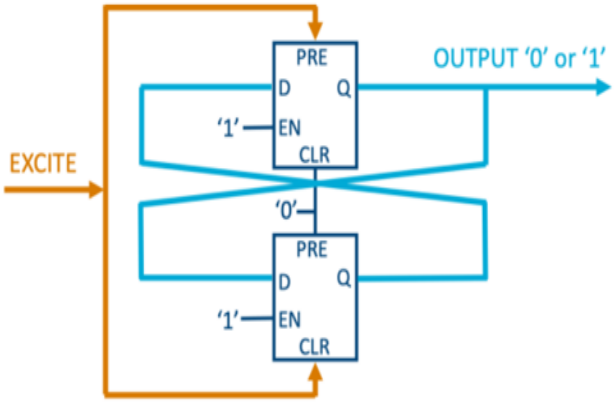


Fig 5.9 : Butterfly PUF

Schematic of the Combinational Trojan inserted circuit shown in fig 5.6 is obtained after synthesis is done. The LUTs and other components can be observed in the schematic shown in fig 5.6.

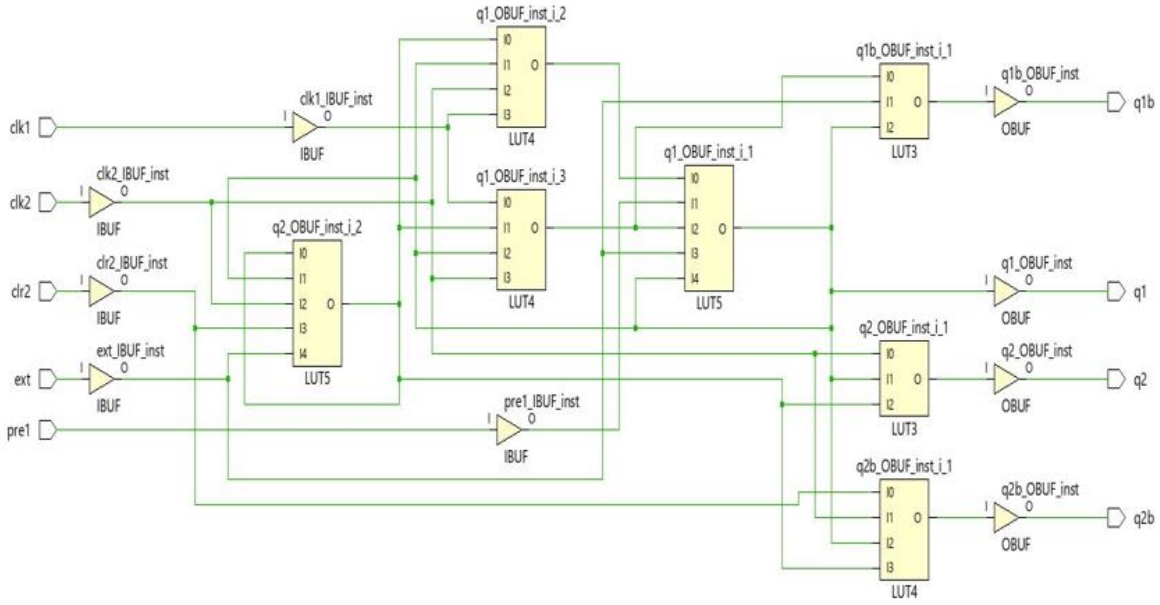


Fig 5.10 : Schematic Diagram of Butterfly PUF

The floor planning for the Butterfly PUF circuit is shown in fig 5.11

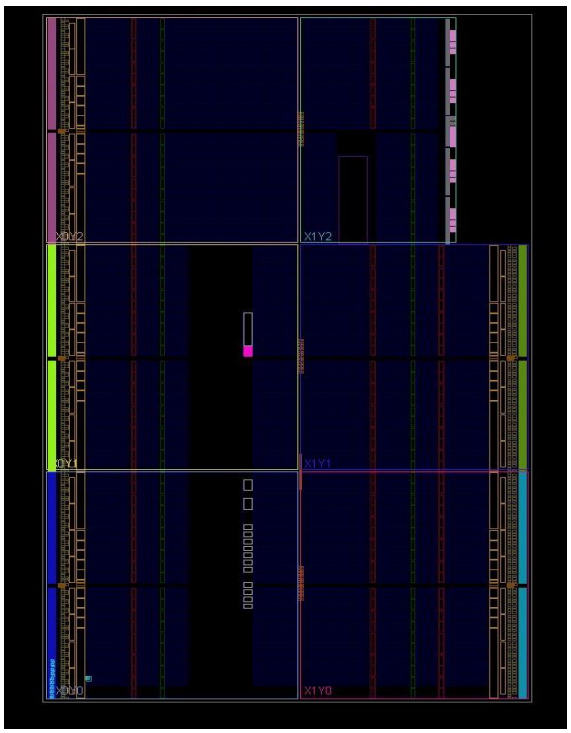


Fig 5.11 : Floor planning for the Butterfly PUF

The simulation results in fig 5.12 shows the normal functioning circuit of Butterfly PUF.

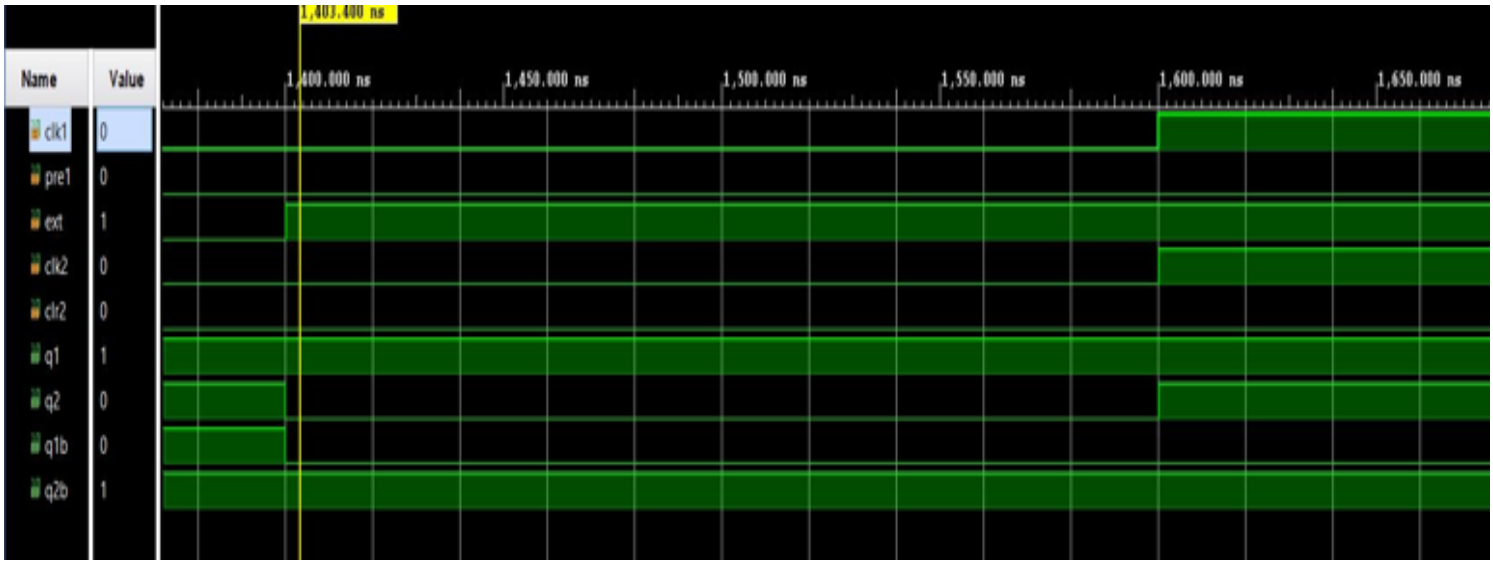


Fig 5.12: Simulation Result of Butterfly PUF

Simulation results:

NAME	VALUE
LUTs used	5
On Chip Power	1.298 W
Junction Temperature	26.2 `C

Table V : Simulation Results of Butterfly PUF

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Formal methods are great tool in order to prove that the implementation of an electronic design behaves as specified. Hence, we identify equivalence checking as adequate measure in order to reveal manipulations of the bit-stream configuration, which is well-known state of the art. These type of trojans inserted into the ICs are known as the hardware trojans, which are very hard to detect and there is more significance in discovering these hardware trojans and eliminating them as early as possible. This project shows that it is easily possible to inject malicious behaviour into electronic designs using compromised design tools without being noticed by neither the designer nor the state-of-the-art tools targeted at Trojan detection. However, once the design complexity increases of both the design and malicious functionality, more sophisticated approaches to equivalence checking are needed. Here we had introduced a delay hardware trojan and a combinational hardware trojan inside the circuit and through simulation using Vivado Xilinix software and we had listed down all the parameters of those trojan inserted circuit and compared them with trojan free circuit which is called as the golden circuit here. This method is called as Side Channel Analysis which is a most common method for hardware trojan detection.

This work provides sharper bounds for the case of detection of hardware trojans using off-the-shelf devices, allowing reducing the costs associated with trojan detection. In this manuscript, we highlighted the dilemma of finding a one fits all solution to the problem finding hardware trojans fitting different taxonomies. To this end, we presented the corner stone for the detection of hardware trojans using off-the-shelf devices. We successfully demonstrated the ability of off-the-shelf devices to detect trojans in different settings, namely: sleeping and active. We believe that our practical work has the enormous potential in the successful detection of hardware trojans. We also believe that our research regarding the Physically Unclonable Functions tells that future of our world is completely based on the data, privacy and its security where these PUFs play a crucial role. These PUFs are more used where the output of the system to be unique which defines its level of security.

In the future we will aim at developing techniques to insert more types of complex hardware trojans into the circuit and also exploit more detection methods and explore other trojan taxonomies in more intricate designs and with advanced malicious purposes. We also believe that these Physically unclonable functions can be of more use in situations where a key used for the encryption is to be unique and secure. However this work is completely based on the insertion of a combinational and delay trojan into a circuit and then detecting it, so this is completely based on the simulation software Vivado Xilinix. But this work can be implemented in the hardware/real-time applications. Also the trojan used here in combinational trojan can be minimalised using contention such that it is hard to detect the faulty output making the detection of the trojans even more tougher. Future work will compare the technique proposed against smaller known trojans and the process variation and manufacturing variation will be taken into account. Furthermore, the number of test vectors for Vivado power estimator will be increased in order to increase its accuracy.

REFERENCES

1. He, J.; Zhao, Y.; Guo, X.; Jimmie. Hardware Trojan Detection Through Chip free Electromagnetic Side-Channel Statistical Analysis. IEEE Trans. Very Large-Scale Integration (VLSI) Syst. 2017, 25, 2939– 2948.

2. TetraMAX ATPG: Automatic Test Pattern Generation.” Synopsys, Inc., 2017.

3. Shende, R.; Amba wade, D.D. A Side channel-based power analysis technique for hardware trojan detection using statistical learning approach. In Proceedings of the 2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN), Hyderabad, India, 21–23 July 2016; pp. 1–4

4. Bao, C.; Forte, D.; Srivastava, A. Temperature tracking: Toward robust run-time detection of hardware Trojans. IEEE Trans. Computation Aided Design Integration Circuits Syst. 2015, 34, 1577–1585

5. S. Moein, J. Subramanian, T. A. Gulliver, F. Gebali and M. W. El-Kharashi, "Classification of hardware Trojan detection techniques," *2015 Tenth International Conference on Computer Engineering & Systems (ICCES)*, Cairo, 2015, pp. 357-362.

6. Nowruz, A.N.; Hu, K.; Koushanfar, F.; Reda, S. Novel Techniques for High-Sensitivity Hardware Trojan Detection Using Thermal and Power Maps. IEEE Trans. Computation Aided Design Integration Circuits Syst. 2014, 33, 1792–1805

7. S. Bhunia, M. Hsiao, M. Banga and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures", Proceedings of the IEEE, vol. 102, no. 8, pp. 1229-1247, Aug 2014.

8. N. Yoshimizu, “Hardware Trojan detection by symmetry breaking in path delays,” in Proc. IEEE Int. Symp. Hardware Oriented Security Trust, 2014, pp. 107 111

9. Charles Herder, Meng-Day Yu, Farinaz Koushanfar. "Physical Unclonable Functions and Applications: A Tutorial "Volume: 102 Issue: 8. 2014

10. Swarup Bhunia, Seetharam Narasimhan, Rajat Subhra Chakraborty "Hardware Trojan Detection by Multiple-Parameter-Side-Channel-Analysis" IEEE Transactions, Year:2013, Volume: 62, Issue: 11.

11. Charles Herder, Meng-Day Yu, Farinaz Koushanfar. "Physical Unclonable Functions and Applications: A Tutorial "Volume: 102 Issue: 8. 2014 Swarup Bhunia, Seetharam Narasimhan, Rajat ubhra Chakraborty "Hardware Trojan Detection by Multiple-Parameter Side Channel Analysis "IEEE Transactions, Year: 2013, Volume: 62, Issue: 11.

12. Clemens Helfmeier, Christian Boit "Cloning Physically Unclonable Functions"2013 IEEE,4th edition.

13. J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, “Design and analysis of ring oscillator-based design for trust technique,” in Proc. 29th VLSI Test Symp., 2011, pp. 105 110.

14. H. Salmani, M. Tehrani poor, and J. Plusquellic, “A novel technique for improving hardware Trojan detection and reducing Trojan activation time,” IEEE Trans. Very Large-Scale Integration Syst., vol. 20, no. 1, pp. 112 125, Jan. 2011.

15. Zheng Gong1, and Marc X. Makkes21, Guangzhou "Hardware Trojan Side-Channels Based on Physical Unclonable Functions",2011.

16. Merli, D., Stumpf, F., Eckert, C.: Improving the quality of ring oscillator pufs on fpgas. In: WESS, p.ACM, New York (2010) Morozov, S., Maiti, A., Schaumont, P.: An Analysis of Delay Based PUF Implementations on FPGA. In: Sirisuk, P., Morgan, F., El-Ghazawi, T., Amano, H.(eds.) ARC 2010. LNCS, vol. 5992, pp. 382–387. Springer, Heidelberg (2010)

17. Morozov, S., Maiti, A., Schaumont, P.: An Analysis of Delay Based PUF Implementations on FPGA. In: Sirisuk, P., Morgan, F., El- Ghazawi, T., Amano, H.(eds.) ARC 2010. LNCS, vol. 5992, pp. 382–387. Springer, Heidelberg (2010)

18. R¨ uhrmair, U., Sehnke, F., S¨olter, J., Dror, G., Devadas, S., Schmid Huber, J.: Modeling attacks on physical unclonable functions. In: Al-Shaer, E., Keromytis, A.D. Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 237–249. ACM, New York (2010)

19. L. Lin, W. Burleson, C. Paar, "MOLES: Malicious off chip leakage enabled by side channels," in Proc. Intl. Conf. Comput. Aided Des., 2009, pp. 117 122.

20. DARPA, TRUST in Integrated Circuits (TRUST),2008. [Online]. Available: http://www.darpa.mil/mto/programs/trust/index.html

21. Jin, Y.; Makris, Y. Hardware Trojan detection using path delay fingerprint. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 9 June 2008; pp. 51–57.

22. Banga, M.; Hsiao, M.S. A region-based approach for the identification of hardware Trojans. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 9 June 2008; pp. 40–47.

PAPER PUBLICATION DETAILS

Paper submitted to a conference “INTERNATIONAL CONFERENCE ON EMERGING TRENDS IN INFORMATION AND COMMUNICATION TECHNOLOGY”, -ICETICT 2021.