Masaryk University
Faculty of Informatics

DESIGN AND DETECTION OF HARDWARE TROJANS

Master's Thesis

Prashanth Reddy G

Hyderabad, December 2017

Masaryk University
Faculty of Informatics

DESIGN AND DETECTION OF HARDWARE TROJANS

Master's Thesis

Prashanth Reddy G

Hyderabad, December 2017

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this thesis is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Prashanth Reddy G

Advisor: prof. Ing. Václav Přenosil, CSc

Acknowledgements

I take this chance to thank my supervisor prof. Ing. Václav Přenosil, CSc for my thesis supervision right from formulating the thesis statement till the end. Thanks for his warm interaction and patience.

I thank prof RNDr. Václav Matyáš, M.Sc., Ph.D. for his help in fine tuning and finalizing thesis assignment. Special thanks to his quick and accurate replies to my queries.

Finally, I thank my parents, wife Lasya, child Aneesh, fellow course participants, other faculty, staff members of Masaryk University and friends for their support during this course.

Abstract

Trojan hardware is an interesting topic in terms of information systems and computer networks protection. This thesis will explore the possibilities of implementation and then the possibility of detecting the operation of Trojan in Systems-on-Chip (SoC), ASIC and FPGA.

In this thesis a simple model of the transmission system UART serial communication is implemented on SPARTAN 6 Xilinx FPGA. Hardware Trojans with four different types of triggers and two different payloads are implemented in verilog HDL and realized on FPGA board. As a proof of concept Trojan detection techniques Logic equivalence check and side channel analysis with parameters area, powers are applied.

Keywords

Hardware Trojans,Application Specific Integrated circuit(ASIC),Field Programmable Gate Array(FPGA), System-on-Chip(SoC),Universal Asynchronous Receiver Transmitter (UART), Side Channel Analysis, Logic Equivalence Check(LEC), Intellectual Property Core(IP core), Leakage Power, Dynamic Power, Area, Register Transfer Logic(RTL), Hardware Description Language(HDL),Complimentary Metal Oxide Semiconductor(CMOS)

Contents

1 Introduction **1**
1.1 *Contributions* 1
1.2 *Thesis organisation* 2

2 Hardware Trojans:Classification **3**
2.1 *Hardware Trojan* 3
2.2 *Taxonomy of Hardware Trojans* 3
2.3 *Activation Mechanism* 4
2.3.1 Always on Trojans 4
2.3.2 Internally triggered 5
2.4 *Payload: Action of Trojans* 6
2.5 *Chapter Summary* 7

3 Analysis of Trojan Insertion Possibility **9**
3.1 *Hardware Trojans Inside FPGA: Analysis* 9
3.2 *FPGA Architecture* 9
3.2.1 Configurable Logic Blocks (CLB): 10
3.2.2 Configurable Interconnect 10
3.3 *Trojans insertion possibilities in FPGA* 12
3.3.1 Digital Clock Manager: 12
3.3.2 Trojans in interconnect switch matrix: 12
3.3.3 Block memories 13
3.3.4 Bit stream Decryption key: 14
3.3.5 Trojans insertion at I/O: 14
3.4 *Hardware Trojans inside Systems-on-Chip and ASIC: Analysis* 14
3.4.1 System Level Trojans 14
3.4.2 Processor: 15
3.4.3 Device controller: 16
3.4.4 Bus Interconnect: 16
3.5 *Chapter Summary* 16

4 Trojan Detection Techniques **17**
4.1 *Destructive Techniques:* 17
4.2 *Side channel analysis* 18
4.2.1 Path Delay: 19

4.2.2 Power: . . . 20
4.2.3 Area: . . . 21
4.3 *Ring-Oscillator Network(RON):* . . . 22
4.4 *Logic testing:* . . . 24
4.4.1 Full functional testing: . . . 24
4.4.2 LEC: . . . 25
4.4.3 FANCI: Functional analysis of nearly unused circuit identification . . . 25
4.5 *Chapter Summary* . . . 26

5 FPGA Implementation of Trojans and Detection 27
5.1 *FPGA board details:* . . . 27
5.2 *Specifications of the design:* . . . 27
5.3 *FPGA implementation* . . . 27
5.4 *Trojan insertion:* . . . 28
5.4.1 State machine in receive module: . . . 30
5.4.2 Counter based Sequential Trojan: . . . 32
5.4.3 State machine in transmit module: . . . 33
5.4.4 Combinational Trojan: . . . 34
5.5 *Applying Detection Techniques* . . . 36
5.5.1 Side channel parameter analysis: . . . 36
5.5.2 Logic Equivalence Check: . . . 44
5.6 *Chapter Summary:* . . . 46

6 Conclusions 47
6.1 *Future work:* . . . 48

Bibliography 49

1 Introduction

Electronic systems today from simple to large designs almost all are in the form of Systems-on-Chip, Application Specific Integrated Circuit (ASIC) or realized in Field Programmable Gate Array(FPGA). Now in the days of faster time to market to develop these complex ICs or to realise design in FPGA many IP cores such as Processors, Memory controllers, DSP cores, USB controllers, Programmable Bus Interconnects, and Camera controllers etc., are taken from third party IP vendors and integrated [1,11]. Most of the electronic design centers do not have their own fabrication centers. They rely on the outside foundries for manufacturing. When relied on outside agencies for either IP cores or for IC, FPGA fabrication there is possibility for hardware Trojans being inserted or to keep back doors inside design with malicious intent.

ASIC or SoC development has a long cycle of development. From specification to deployment several stages are involved. Due to involvement many teams and organisations in many cases of IC development most of the stages are untrusted as shown in Fig 1-1.

1.1 Contributions

- This thesis analysed the possibilities of Trojan insertion in FPGA, ASIC and System on Chip. Explained Trojan detection mechanism using non-destructive methods.
- Implemented Trojans in simple transmission model UART serial communication, demonstrated the stealthiness of Trojan(s)
- Detected the presence of Trojan applying side channel analysis techniques and logic equivalence check.

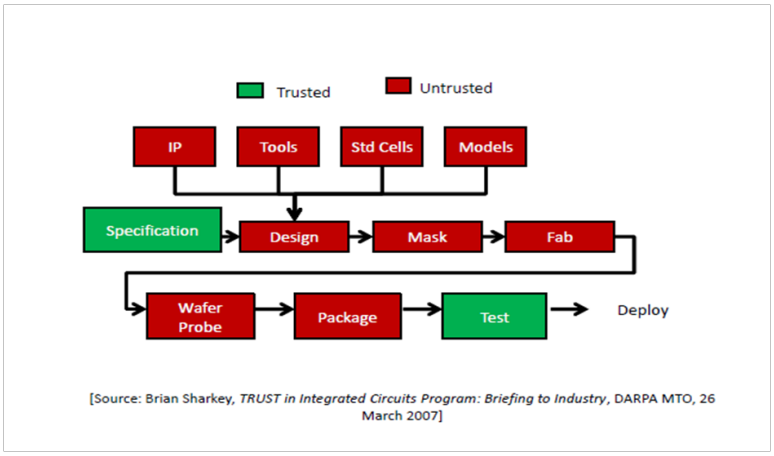

Figure 1.1: SoC development cycle

1.2 Thesis organisation

Rest of the thesis is organised as follows. Chapter 2 gives the introduction to hardware Trojans their classification. Chapter 3 presents potential places of Trojan insertion in Field Programmable Gate Array(FPGA), System-on-Chip(SoC) vis a vis Application Specific Integrated Circuit(ASIC). Chapter 4 presents various Trojan detection techniques classification, mainly focusing on non-destructive testing including Side Channel Analysis, Ring Oscillator based network, Logic Equivalence Check, techniques relating to identifying suspicious code in third party un-trusted IP cores, Functional testing etc. In chapter 5 FPGA implementation of simple-transmission model UART serial transmission, Trojan(s) implementation in this design at register transfer level abstraction and result(s) of applying side channel analysis and LEC methods are presented. Finally in Chapter6 conclusions and future work are presented.

2 Hardware Trojans:Classification

Here in this chapter I will cover the concepts of hardware Trojans, their classification. This chapter presents concepts from available literature on hardware Trojans.

2.1 Hardware Trojan

Hardware Trojan is a malicious modification of the electronic design. This can be a simple stuck at zero or struck at one Trojan to a complex modification. A simple example of Trojan is an two input OR gate inserted between an output port with one input connected to logic-1 when triggered and the other to actual output. One more simple Trojan can be a two input AND gate with one input tied to logic-0 when triggered and the other input to actual output. Trojan consists of two parts Trigger and Payload. Trigger part decides when the Trojan is to be active and when it should be dormant. Payload part is actual modification to a system which does the damage. Trojan is designed by keeping two aspects one is malicious intent which determines extent of damage to be caused and the other is how to evade being detected during standard testing.

2.2 Taxonomy of Hardware Trojans

Here in this section hardware Trojan attributes such as where in the design, in which phase of the design hardware Trojans can be inserted, how they can modify expected behavior of the system are presented. Hardware Trojans can be inserted at any stage of design phase. Design phase of SoC,ASIC and FPGA consists of specification, design,fabrication and assembly and testing.Trojans can be inserted at any of these phases. Design phase consists of various abstraction levels such as System Level,Register-transfer-level, Gate level, Transistor level and Physical level. During transformation from one abstraction level to another at design time, many teams will be working on the design. At any stage of the transformation phase Trojans can be inserted. Hardware in digital designs consists of Processors, Memory,

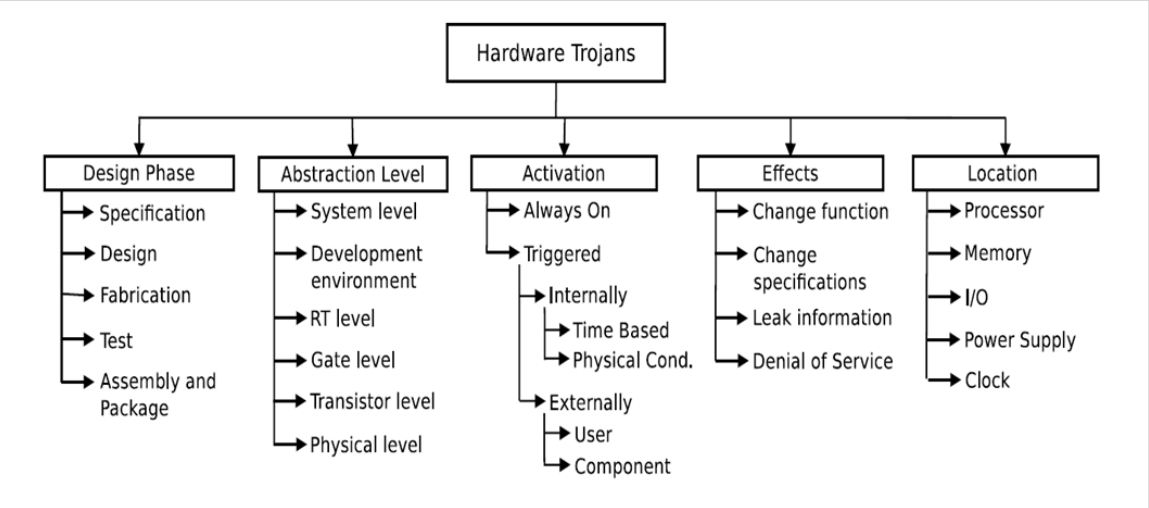

Figure 2.1: Taxonomy of hardware Trojans: Rajendran et al. (2010) [9]

Input/Output ports, Power supply and Clock etc. In any of these Trojans can be inserted. So far where and when Trojans inserted is discussed.When these Trojans becomes active is decided by activation/trigger mechanism and what it does when activated is called as payload.

2.3 Activation Mechanism

Based on activation Trojans are of two types

1. Always on
2. Triggered

2.3.1 Always on Trojans

In Always on category Trojan is active all the time when design is under operation. For example during fabrication some parameters such as thickness of metal layers at certain places can be made very small intentionally. On repeated use of the component permanent failure may happen to device after sometime due to early wear out. Another similar case is during fabrication if low power cells are replaced by

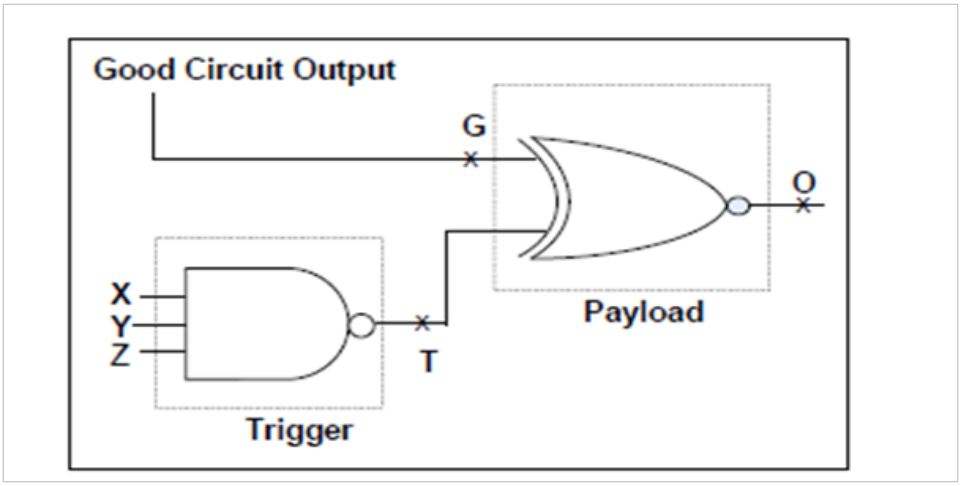

Figure 2.2: Combinational trigger mechanism [3]

high power consuming cells, for battery operated devices where power is a critical parameter, device fabricated does not meet its specification. These types of parametric variations can not be detected during the functional testing but causes permanent deteriorating effect on the system. This may not cause a damage but it certainly gives a loss to the designer in terms of time and reputation.

2.3.2 Internally triggered

Internally triggered hard ware Trojans make use of certain condition inside the device or component during the operation. There are in general two types of internally triggered mechanisms one is combinational and other is sequential.

A. Combinational Trigger Mechanism: In combinational trigger mechanism, only logic gates are used to form trigger circuit.In this memory elements are not involved.These trigger mechanisms makes use of certain internal nodes, when they take some predefined values, trigger is activated. In Fig 2-2 if each of the nodes X, Y,Z has the probability .01 to take certain logic value then all of them to take desired combination has the probability .000001. It means that during normal functional testing the probability of such nodes to get a desired

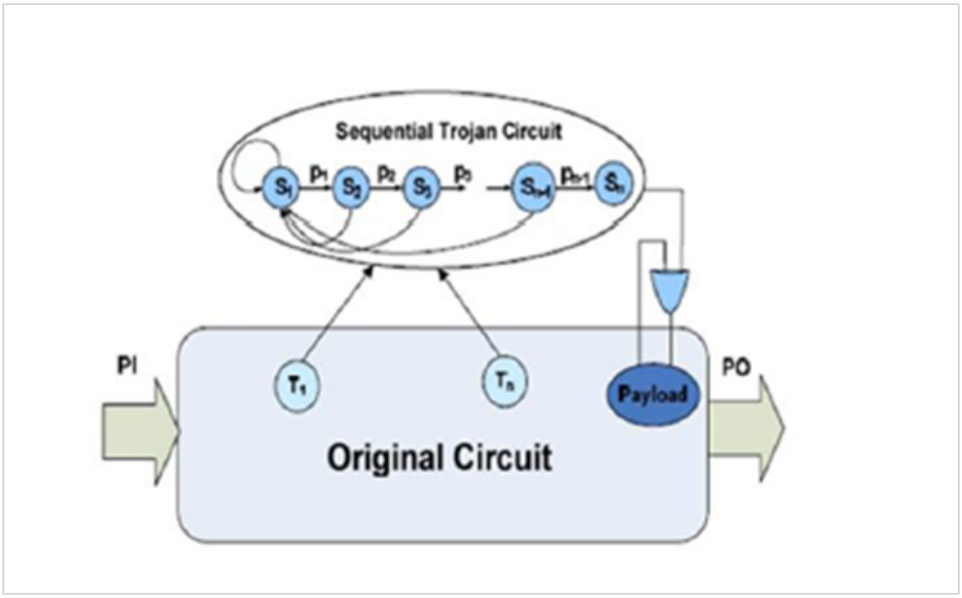

Figure 2.3: Sequential trigger mechanism [3]

value is very low. In general,these signals are taken from part selects of address & data buses.

B.Sequential Trigger Mechanism: Sequential Trojan circuit is enabled on the occurrence of certain events in a sequence. These events can be write to the control registers or read from certain address in memory. As shown in Fig 2-3, Sequential Trojans are implemented using state machines. To hold these states, memory elements such as flip flops and latches are used. During normal functional testing to detect these types of Trojans is very difficult because the Trojan designer can choose any complex type of sequence to trigger the payload.Simple counter based Trojan is an example for sequential Trojan design. After certain number of toggles of a rare node or clock, counter based Trojan gets activated.

2.4 Payload: Action of Trojans

Payload part of the hardware Trojan does the intended job of Trojan designer. It can be of following types.

- Functional modification
- Denial of service
- Specification modification
- Leakage of secret information.

2.5 Chapter Summary

In this chapter, I have described the basic model of the hardware Trojan consisting of trigger part and payload part, taxonomy of hardware Trojans. In the coming chapters where and how they can be inserted in SoC,FPGA are discussed.

3 Analysis of Trojan Insertion Possibility

In this chapter I am reporting the analysis of possibilities for Trojan insertion in FPGA, ASIC and Systems-On-Chip.

3.1 Hardware Trojans Inside FPGA: Analysis

FPGAs are used in many fields such as signal processing, embedded computing, multimedia, and security. FPGA devices in general are bought from different FPGA vendors and configured many times as and when required in the field. These vendors may fabricate these FPGA devices inside their own organizations or may send the designs to IC fabrication centers outside. This creates the possibilities for Trojan insertion in FPGA designs. In next two sections, I am presenting structure of FPGA designs and the analysis of Trojan insertion possibilities in the FPGA hardware designs.

3.2 FPGA Architecture

FPGA devices broadly consists of following major blocks

1. programmable logic blocks or Configurable logic blocks (CLBs)
2. Programmable Interconnects
3. Programmable I/O blocks
4. Digital clock managers
5. Multipliers
6. Block RAMS
7. Embedded processor cores (general purpose processor or DSP)
8. JTAG
9. Non-volatile memory (EEPROM).

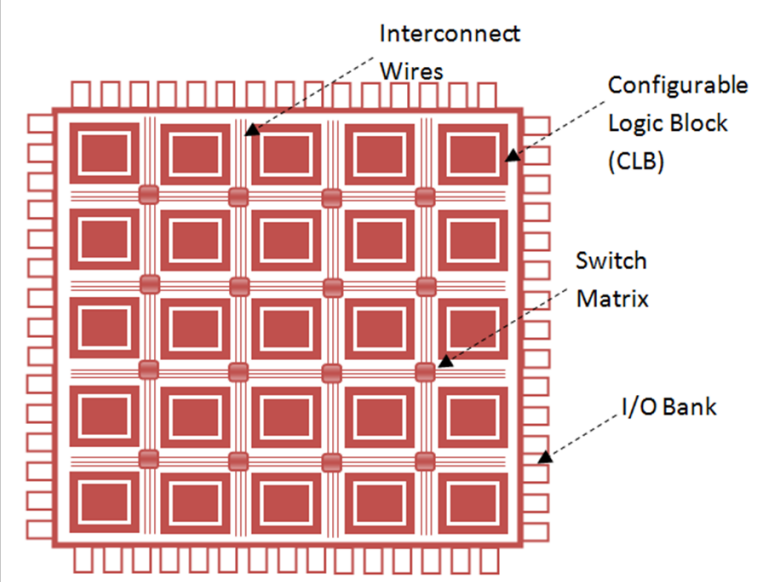

Figure 3.1: FPGA architecture block diagram [8]

3.2.1 Configurable Logic Blocks (CLB):

Configurable logic blocks typically consist of two slices SLICEL and SLICEM. Each slice consists of several logic cells. Logic cell contains lookup table, full adder carry logic, multiplexers and flip-flops. Lookup table implements the functionality by storing function as a truth table. It gives the set output of a function for a given input combination. Lookup tables are used to implement shift registers, combinational logic and distributed RAM.

3.2.2 Configurable Interconnect

Configurable interconnect provides interconnection among logic blocks to realise user defined function. There are connecting lines long and short to connect CLBs. Switch matrix provides connection among these connecting lines in flexible manner. Global clock lines are designed with lower propagation delays connecting clocked flops in CLBs minimising skew.

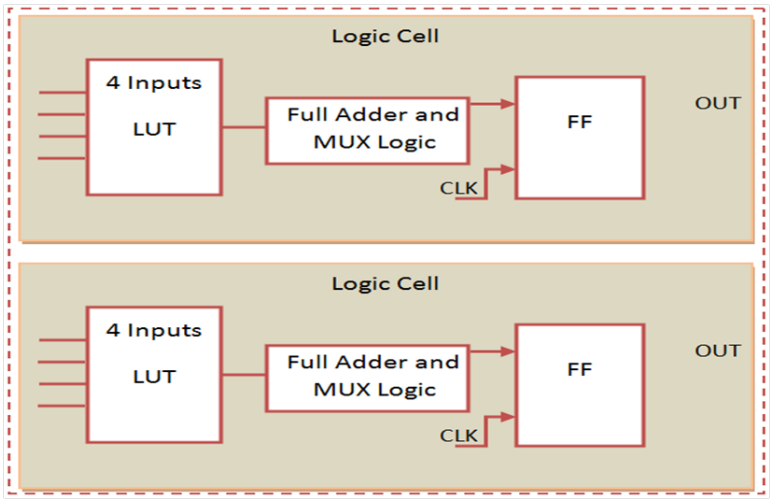

Figure 3.2: Logic Cell internal units [8]

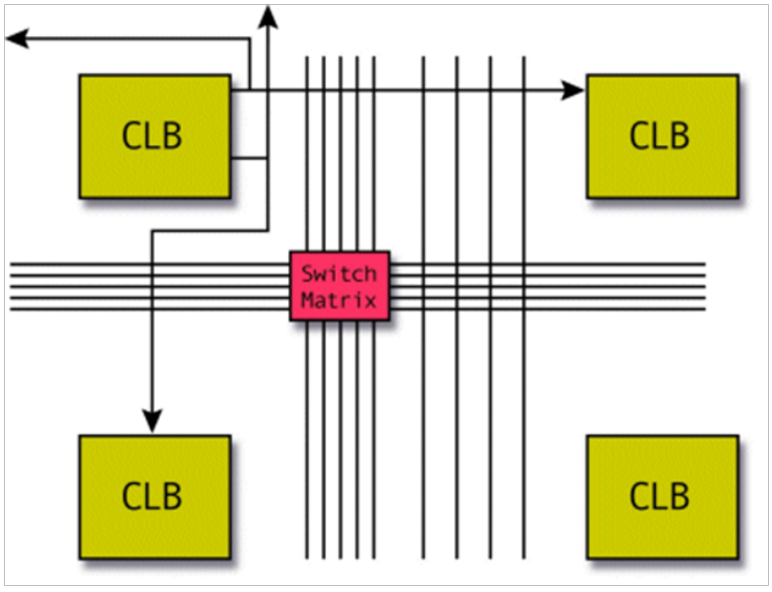

Figure 3.3: Routing in FPGA[8]

Digital Clock Manager: DCM performs frequency Synthesis, clock division, clock phase shift, clock jitter and skew management.

Multiplier block: Implements dedicated multiply operation for both signed and unsigned.

Block RAMS: Dual port Block RAMs are available in FPGAs to implement memory of the designs

3.3 Trojans insertion possibilities in FPGA

FPGA typically consists of regular array of configurable logic cells, other blocks such as digital clock managers, block RAMs and interconnect structure. So, an attacker can identify these regular blocks easily by doing reverse engineering the layout of FPGA to acquire required know how of the design to insert Trojans. Trojans can affect I/O s, memory, clocks and interconnect adversely.

3.3.1 Digital Clock Manager:

For desired frequency synthesis, division factor, multiplication factor and required amount shift for clock phase shifting is stored in configuration cells SRAM memory. These values can be modified after triggering payload which modifies configuration cell values. Trigger can be designed using counter based sequential Trojan as shown in Fig 3-4 [1]. Modification of multiplication factor or division factor if increases the clock frequency or change in the required phase shift causes timing being not met in clock based sequential designs which leads to a potential system failure.

3.3.2 Trojans in interconnect switch matrix:

If Trojans are inserted in switch matrix, which are enabled taking some time after start, then the interconnection among various blocks changes which in turn changes the functionality to some other arbitrary function. These Trojans can be independent of the design being configured in FPGA. Their activation when independent of the design

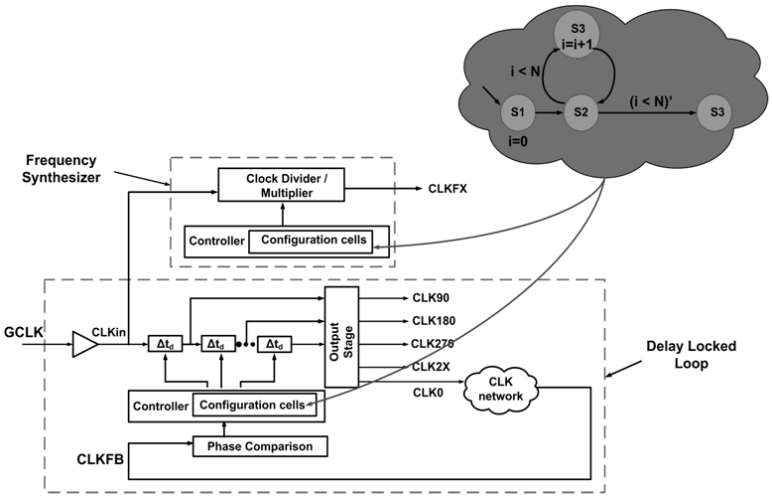

Figure 3.4: Trojan insertion possibilities in DCM[1]

can be rare because it is not known priorly in which part of the FPGA, design is going to be configured. When inserted at different places and corners of the FPGA device their chance of affecting the design becomes more.

3.3.3 Block memories

Block memories are used for table lookup in the designs to make computation faster. Normal flops and latches a.k.a distributed memory can be used for them, but their access times and area requirements are large. Block memories are denser and access times are less. These are vulnerable for Trojan insertion. These memories can be easily identified and what for they are used can be easily guessed if the context of the design is known. For example, if the configured design is symmetric Crypto blocks like DES and AES these memories are used for storing S-Boxes and round key constants. FPGAs being configurable on the fly, these memories can be reconfigured for some other values and from that it becomes easy to crack the key or decrypt the content from these ciphers.

3.3.4 Bit stream Decryption key:

In order to protect bit-stream copying and then reverse engineering the design, FPGA vendors encrypt the bit-stream in high end devices. If Trojan is inserted such a way that it intercepts the bus between decryptor and memory in which key is stored, key can be copied and send to attacker.

3.3.5 Trojans insertion at I/O:

I/Os in FPGA are either configured as input or output and are interfaced to external system. When Trojan is inserted at I/O it can turn on output enable for a port which is configured as input and turnoff output enable for a port which is configured as output. This causes excess current flow into the system connected and can damage the system connected [1].

3.4 Hardware Trojans inside Systems-on-Chip and ASIC: Analysis

In this section potential places and ways of inserting Trojans in a System-on-Chip are explored. Since System-on-Chip and ASIC development stages are same except the presence of more number of modules interconnected in SoC, ASIC is a design for dedicated function. What applies to ASIC applies to SoC as well in specific module. What applies to SoC in a specific module applies to ASIC.

3.4.1 System Level Trojans

In the case of System On chip it is usual practice in the industry to take processor, bus interconnect and peripheral IP cores from third party vendors like ARM, CEVA, and ST microelectronics instead of developing from scratch by themselves. When third party IPs are used then there is possibility to have Trojans in individual IPs or in the collusion of two or more IPs from the same IP vendor. Following four scenarios exists [11].

1. Passive interception

2. Modification
3. Diversion
4. Masquerading

Interception: In this scenario, an IP silently reads the communication between two other IPs and sends it to external system or another IP.

Modification: In this scenario third IP interrupts the communication between two IPs and modifies it maliciously.

Diversion: In this scenario IP diverts the communication between two IPs to other IPs.

Masquerading: In this scenario IP disguises, itself as another IP to get service from other IP or to control the behavior of overall system.

Here below I will describe threat scenarios by taking some example IP cores.

3.4.2 Processor:

When third party processor IP is used it is very difficult to test processor IP exhaustively. Designer must rely on the regression test suit provided by the vendor to verify it is what the vendor is intended to deliver and not modified in between during the delivery process. But when the processor IP has some malicious design following cases are possible. Processor IP can exploit reserved instruction op-codes. Processor can initiate DMA transfer to external peripheral and can send secure content outside of the system. Processor can write information intended for secure memory region into non-secure memory region by using shadow loads and stores if it has malicious memory access logic inside it. subsectionMemory Controller: when system uses memory mapped IO, Memory controller which contains malicious logic may change the contents of the specific IP for example if it effects the setting of device controller which controls the critical section of system then it may cause great damage to the system. These kinds of

Trojans identification at individual IP validation is very difficult. This kind of situations arises in a system level under operation.

3.4.3 Device controller:

Device controller controls devices like Ethernet, Modem, USB, Blue tooth, and different I/O components. An untrusted device controller may modify the data to/from the device from/to processor. When the communication is for authentication of some purpose then modification of few bits in data may cause denial of service.

3.4.4 Bus Interconnect:

Bus interconnect arbitrates the access of bus among peripherals. A malicious bus interconnect may grant access of secure region to a non-secure component. Interconnect may escalate the privileges of certain I/O s to access the secure region or can deny service to I/O with a malicious intent.

3.5 Chapter Summary

In this chapter possible ways of inserting Trojans in FPGA designs, System on chip are discussed. In FPGA while fabricating the FPGA Trojans can be inserted and in the design being configured also Trojans may be present or even during runtime because of on the fly reconfiguration of design Trojans can be inserted. In System-On-Chip Trojans may exist in single IP alone or it is possible for two or more IPs to collude to form a Trojan.

4 Trojan Detection Techniques

In this chapter I am presenting Trojan detection techniques for an integrated circuit. For an ASIC, SoC and FPGA majority of these detection techniques can be applied with small differences. I will point out these differences if any exist for the specific detection technique otherwise I am presenting detection techniques for all of them together. There are many proposed detection techniques available in literature. Detection techniques are applied depending on the targeted design and types of hardware Trojans planning to be detected. For some techniques assistance is required in the design others can be applied directly or at run time. Broadly these detection techniques can be classified into simple taxonomy as shown in Fig 4-1.

4.1 Destructive Techniques:

Here I will present briefly about destructive techniques which are well covered in literature and exist from a long time. In this method IC under test is examined physically layer by layer. Design is extracted by reverse engineering information collected from each layer. For this process IC after opening physically treated with chemicals.Layer information is collected by scanning each layer. There are many scanning techniques are available such as Scanning electron microscopy, Scanning optical microscopy, Light induced voltage alteration etc.

These methods are accurate, but they are quite expensive. They need skilled persons and a dedicated laboratory. At the end of the testing,

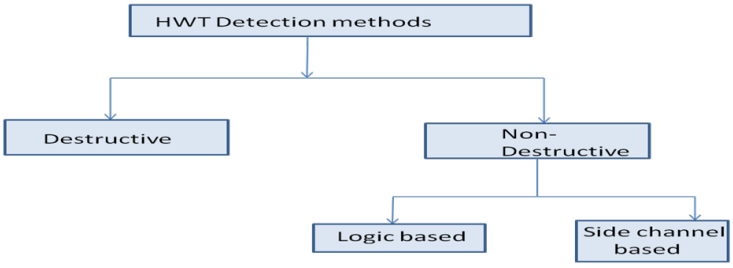

Figure 4.1: Hardware Trojan detection methods [self]

chip becomes destroyed physically. Only on a random chip from a lot these techniques can be applied. These methods can be used for final conformation after initial analysis from other methods.

When FPGA is considered as an IC from the foundry without having any design configured these methods are still valid for it. When design is configured in the FPGA, bit stream can be reverse engineered to get the source design.

Non-Destructive Techniques:

In non-destructive detection techniques design under test is not destroyed physically. Design can be used after applying these detection techniques if found free of Trojan. In this category broadly two types of techniques are present one is Logic-testing and the other is Side channel analysis.

4.2 Side channel analysis

In side-channel-analysis analysis of physical parameters of the design such as power, temperature, radiation pattern, time delay, noise (acoustics) and area etc., reveal some information about the design. Two identical designs have same physical parameter profile. This characteristic is made use of in side channel analysis for detecting hardware Trojans.

In these methods Trojan(s) free reference design parameters are compared with parameters of Design-Under-Test (DUT). If any significant deviation is there in DUT parameters from reference design parameters, then it can be suspected that the design is modified. Introduction of any new circuit changes these parameters. Here it must be considered that in any of the identical design even from the same manufacturing foundry these parameters differ due to process variations. To avoid false positives or false negatives here threshold limit of comparison has to be kept judiciously.

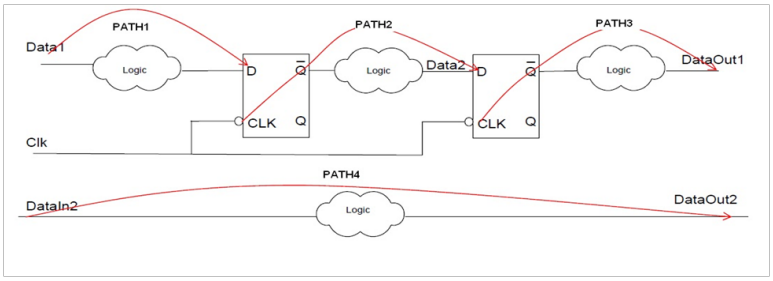

Figure 4.2: Data paths [7]

Here I'm describing few important parameters to be considered for side channel analysis.

4.2.1 Path Delay:

In general, in digital systems, design is modeled as set of flip-flops interconnected with combinational logic in between. Design is treated as set of paths. Each element in the path from starting to end takes certain time and contributes to delay for signal to pass through it. Cumulative sum of the delays of each element in the path from start to end is called as path Delay. Broadly two types of paths are there

1. Clock Path.
2. Data Path.

Data Path:
As shown in Fig 4-2, Data paths can be from

- input to register (PATH1)
- register to register(PATH2)
- register to output(PATH3)
- input to output (PATH4)

Clock Path:

Clock path starts from input port of the design to clock input of register. From starting point to end point there are many buffers to increase

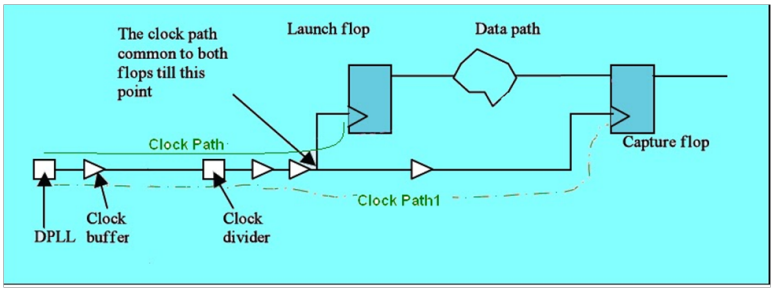

Figure 4.3: Clock path [6]

the drive strength, clock dividers to get the desired frequency. Each of these contributes to delay.

Path delays for set of paths in the reference design are compared with paths of design under test if any significant mismatch is found it indicates the presence of Trojan.This parameter is suitable for comparing designs before fabrication. Once fabricated it becomes difficult to find the path delays between all paths except input to output paths.

4.2.2 Power:

Two kinds of power consumption can be utilized as side channel metrics

- Static power
- Dynamic power

Static Power: Static Power is power consumed by the design when design is powered on without any switching activity i.e. when it is just powered on without any input applied. This is mainly due to leakage power of transistors. Leakage current is due to parasitic diodes in CMOS. These parasitic diodes are reverse biased they contribute reverse saturation current as leakage current. Other sources of leakage are due to many factors such as gate tunneling, sub threshold current, capacitor leakage etc.

$$StaticpowerP_s = VDD * I_{leak}$$

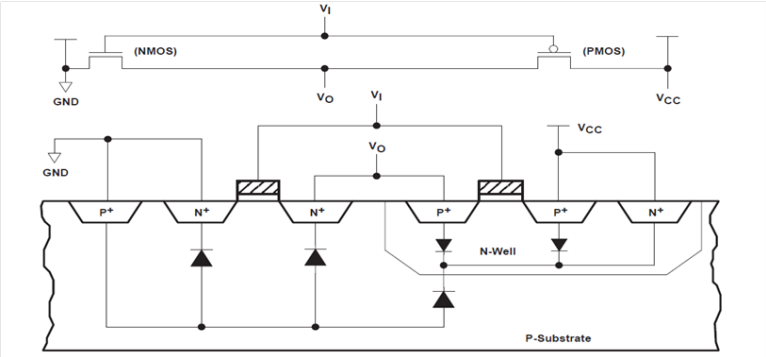

Figure 4.4: Parasitic diodes in CMOS inverter[13]

where VDD is supply voltage
I_{leak} is Total leakage current

Dynamic Power: Dynamic power is due to switching of transistors from one state to another. When design is functional due to data transfers or computations transistors change their state from on to off and off to on. In this process charging and discharging of capacitors take place which causes power dissipation. Ideally in CMOS current flow should be zero, so power consumption should also be zero. But, due to stray capacitances of wires and transistor junctions there exists a path for current flow. There exists a path from supply to ground when switching from one to zero or otherwise for a small fraction of time when both transistors ON. Current that flows during this time is called as short circuit current. This also causes power dissipation.

$$P_{dyn} = \frac{1}{2} * C_L * VDD^2 * f$$

where VDD is supply voltage
c_L is Total load capacitance
f is switching frequency.

4.2.3 Area:

Area is another important parameter that can be used as side channel analysis metric. For System-On-Chip and ASIC area is measured in

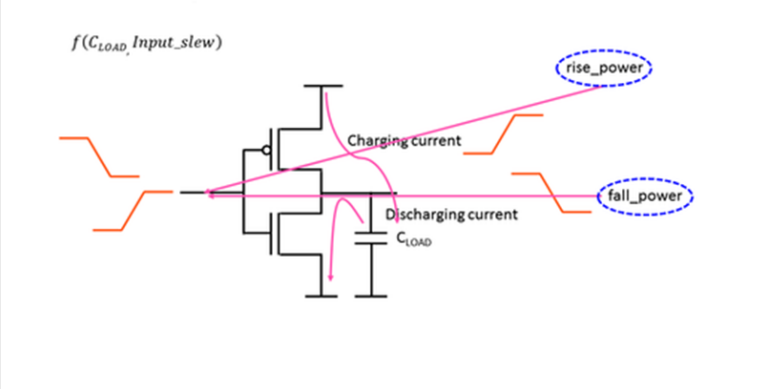

Figure 4.5: dynamic power dissipation in CMOS [13]

terms square nano meters or number of equivalent 2 input NAND gates. If the design is implemented on FPGA it is measured in number of look-up tables (LUTS). If Trojan is present in the design, it does increase area. If the area of Design under test is compared with the area of Trojan free reference design, any significant mismatch indicates the presence of Trojan. Before fabrication to detect Trojans, area is a useful metric, inside-channel-analysis.After fabrication this again can be used as parameter in Destructive methods. Thermal profile, radiation profile of chip are other parameters which can also be used as side channel metrics are not presented in this thesis.

4.3 Ring-Oscillator Network(RON):

This technique makes use of power consumption by Trojan alters the voltage of the Trojan detecting circuit. Odd number of inverters connected serially in a loop forms a Ring Oscillator(RO). Time delay of each inverter contributes to the frequency of oscillator. Power supply noise variation affects this delay. Neighboring gates activity contributes to power supply noise. This feature is utilized in this method to detect the presence of Trojan. Network of ring oscillators placed such a way that ring oscillators are distributed across the entire area of chip. Fig 4-6 shows the distribution of ROs and how they are connected to power metal rails in a Design. This network acts like power monitoring network. Ring Oscillator near Trojan gets affected and its

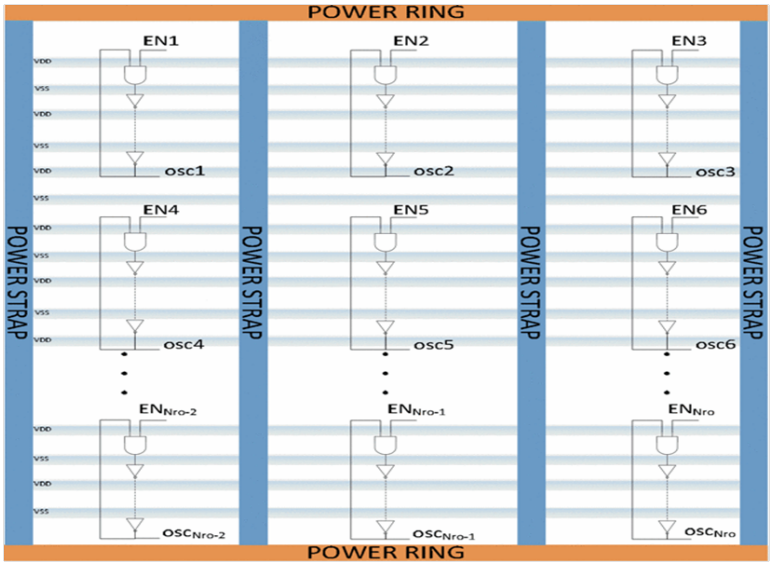

Figure 4.6: Ring oscillator power monitoring network [3]

oscillating frequency gets altered. RO that is far from Trojan gets less affected or doesn't get affected.

The detailed structure of the RON based detection structure is shown in Fig 4-7. Trojan detection system consists of a Phase Locked Loop, Finite State Machine (FSM), Decoder, Multiplexer, RON, Counter, Linear feedback shift register (LFSR) and Serial transmission module.

FSM generates the code as an input to multiplexer and decoder to select RON. During the predefined time interval, counter counts the clock periods from RON. FSM selects RO s one after another and RO's oscillations are counted and recorded. LFSR generates input to the test module, to keep the same environment for all ROs, for reference design and design under test. Since the Trojan can be at any part of entire design area in FPGA, ASIC and SoC, RON network distributes RO to entire area. If RO oscillation count in the DUT and reference design differ then it indicates the Trojan presence. To make noise effects invalid this oscillation count is measured for n number say 1000 times. This method can be used to take finger print or signature of reference Design and can be compared with the signature from Design under test.

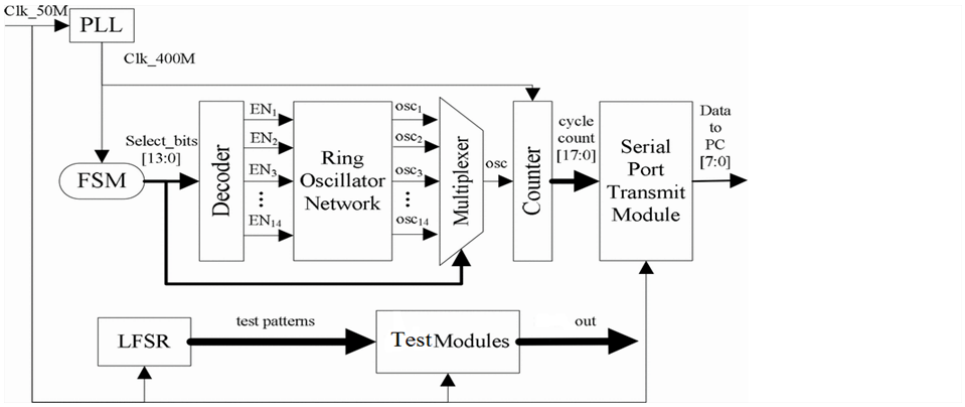

Figure 4.7: RON structural model [3]

4.4 Logic testing:

In the previous section side channel parameters are utilized either directly or in a RON scheme to detect the presence of Trojan. Logic testing of design where logic of the design is tested to check whether it is performing as per the intended behavior or not, manufactured without any faults as intended, there is no extra logic added during design process from one abstraction to other.In logic testing it is also checked for presence of any dormant code present in the design.

4.4.1 Full functional testing:

Functional testing is to verify the design is performing as per its specified functionality. In this method test inputs are applied to design and outputs from the design are collected and compared with the expected output. If the output of design and expected output matches, then there is no modification in the design. In this method all the features and modes of the design are to be tested. Generally,Trojans are designed to evade detection during functional testing of design. If the Trojan is designed such a way that if it gets triggered in the extensive testing for example if the test is run for longer time than the usual practice of testing providing huge resources like emulators, high performance computers then there is a chance that simple counter based sequential Trojans become active and get detected. For completeness

of functional testing code coverage and functional coverage metrics can be utilised. Until desired percentages of these metrics are covered using functional test inputs need to be keep on applied.

4.4.2 LEC:

Logic equivalence check is a formal verification technique where two designs are compared for their equivalence functionally. Compared to exhaustive functional verification where functionality of the design is verified for its intended behavior against the specification, here in equivalence check one design is compared with the reference design, assuming reference design fulfills the specification. In digital design equivalence check is performed by checking combinational logic design connected to key points in the two designs being compared. Key points are basically primary inputs, primary outputs, latches, flip-flops, open ended wires. Key points are matched first in both reference and design for comparison. Once key points are mapped then logic connected to them compared in both designs. If logic connected to all key points are matching, then they are said to be equivalent. If there are extra key points or if logic connected to key points mismatch, then the two designs are not equivalent. This method can be applied for Trojan detection if a reference design is available.

4.4.3 FANCI: Functional analysis of nearly unused circuit identification

This technique is a static analysis tool to detect spurious unused circuit. FANCI is a technique to identify back doors in prefabrication stage of IC design which may be not possible to detect during validation of third party IP in RTL or net list or in hard macro forms. This technique reports suspicious code. It gives a quantifiable metric of stealth from control value.
EXAMPLE: In the table 4.1 the truth table for output OUT as a function of inputs A,B,C is shown. Value of input C changes the output only 4 out of 8 times. So control value of C for output is 0.50. Low control value indicates that the particular input to change the output has less chance, so it has high chance to hide malicious logic. Its stealth value is high. The key insight used in this technique is that the back-doors

Table 4.1: Influence of input c on output OUT

A	B	C	OUT
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

are nearly always dormant and thus rely on nearly unused logic that is this logic never really determines the output [2].The gist of this method is to approximate the truth table for each intermediate output in a design as a function of any wire that can determine that output. Then it computes the influence of each input on the output [2]. If this value is below some threshold that point is designated as unused.

4.5 Chapter Summary

In this chapter Trojan detection methods classification is discussed. Out of these Non-destructive techniques such as Side channel analysis,Ring Oscillator based Network,Logic equivalence check, FANCI are elaborated in detail.

5 FPGA Implementation of Trojans and Detection

In this chapter as a part of proof of concept I am implementing Trojans in Field Programmable Gate Array (FPGA). To implement Trojans, design I have chosen is a serial transmission model UART serial communication. I have chosen UART because I can demonstrate the effect of Trojan using existing equipment PC's hyper terminal and FPGA board. For this I have taken existing reference design in verilog hardware description language at register transfer level abstraction and implementing Trojans in existing verilog code.

5.1 FPGA board details:

FPGA Manufacturer	: Xilinx
Family	: Spartan-6
Device	: XC6SLX100
Package	: FGG676
Speed grade	: -3

5.2 Specifications of the design:

Although the existing design supports many features and programmable, I am choosing UART design with transmit and receive frame of ten bits consisting one start bit, one-stop-bit, eight data bits and parity bit(s)none. Baud rate I am configuring is 9600. Port definitions of the module are given in Table 5.1

5.3 FPGA implementation

I have implemented this design on FPGA in the following steps

1. Verified the functionality of the design using Synopsys VCS simulator. Simulation wave forms where UART is both transmitting and receiving shown in Fig 5-1.

Table 5.1: UART Port list

S.No.	Name	Port Type	Definition
1	Clk	input	External Clock to configure registers
2	uart_ref_clk	input	External clock for data transmission and reception
3	Resetb	input	Design Reset
4	uart_ro	input	Serial data in
5	uart_di	output	Serial dta out
6	regfile_addr	input	Register address
7	regfile_data_in	input	Register data
8	uart_rts	output	Ready to send
9	uart_cts	input	Clearto send

2. Synthesized UART RTL design using Xilinx ISE into FPGA net list.
3. Implemented the floor plan, placement and routing in Xilinx ISE with I/O information in user constraint file (uart.ucf).
4. Generated the bit stream and configured the FPGA.
5. Connected the board to PC using serial port COM1 using cable to DB9 connector of FPGA board.

Design is configured for loop-back mode with baud rate of 9600. Transmit and receive frame are of 10-bits consisting 8-data bits, 1-start and 1-stop bit. Data is sent from hyper terminal, whatever the data received is transmitted back on to hyper terminal. Screen shot of the setup is shown in Fig 5-2.

5.4 Trojan insertion:

I have planned to implement the Trojans in the chosen design at RTL abstraction. Payload of the Trojan is to corrupt the transmitted frame

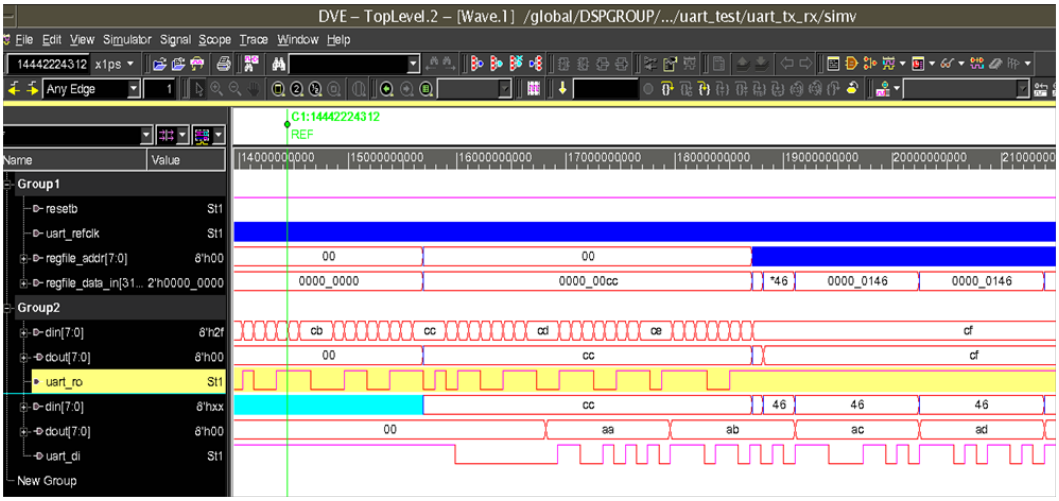

Figure 5.1: UART simulation waveform

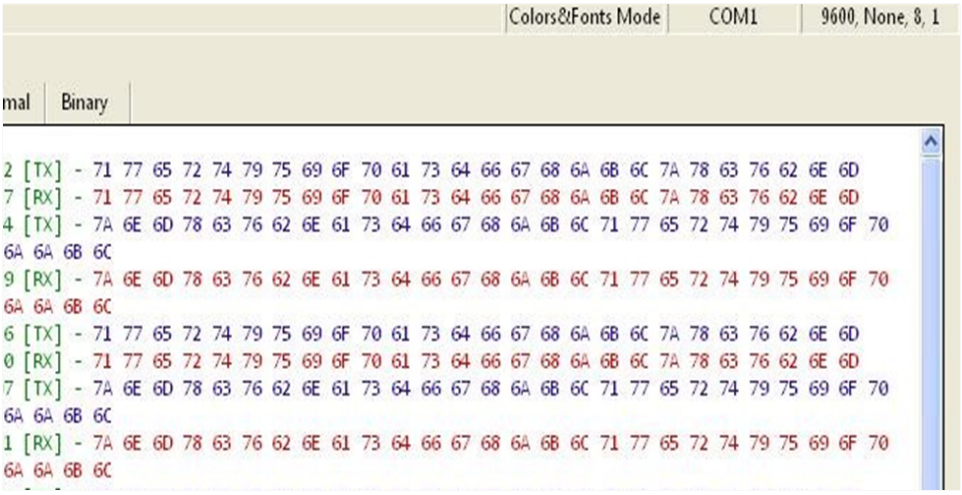

Figure 5.2: UART in loop back [self]

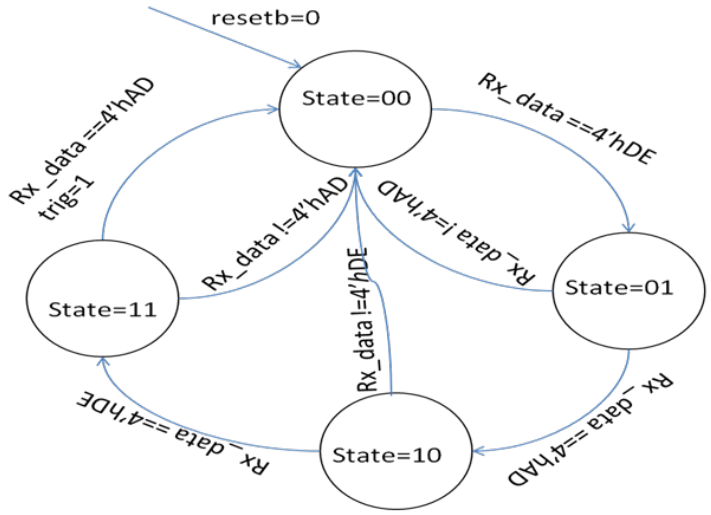

Figure 5.3: Trigger FSM in receiver [self]

after trigger gets activated.Denial of service(DoS) is another Trojan payload variation. In DoS when Trojan implemented is triggered one of the control signals in the transmitter or receiver are modified to permanently high or low then transmitter or receiver can't perform their function of transmitting, receiving. Following scenarios explain various trigger mechanisms used.

5.4.1 State machine in receive module:

In this scenario trigger is implemented in receiving module. When the sequence DE, AD, DE, AD is received trigger gets activated. For this sequence detection state machine as shown in Fig 5.3 is implemented in the receiver part of the design. After trigger activation transmitter replaces the four bits [7:4] in transmitted data bits with 4'hA.

```
reg state =2'b00;
always @ (posedge clk or negedge resetb)
begin
  if (!resetb)
```

```
begin
state <=2'b00;
cor<=1'b0;
trig<=1'b0;
end
else
                begin
                case (state)
                2'b00:
                 begin
                 if(rx_uart_fifo_dout == 8'hde)
                        state<=2'b01;
                 end
                2'b01:
                 begin
                 if(rx_uart_fifo_dout==8'had)
                        state<=2'b10;
                end
                2'b10:
                 begin
                        if(rx_uart_fifo_dout ==8'hde)
                                state<=2'b11;
                        end
                2'b11:
                 begin
                 if(rx_uart_fifo_dout==8'had)
                        begin
                        cor<=1'b1;
                        trig<=cor;
                        end
                 end
        default:
                        begin
                        state<=2'b00;
                        cor<=1'b0;
                        trig<=cor;
```

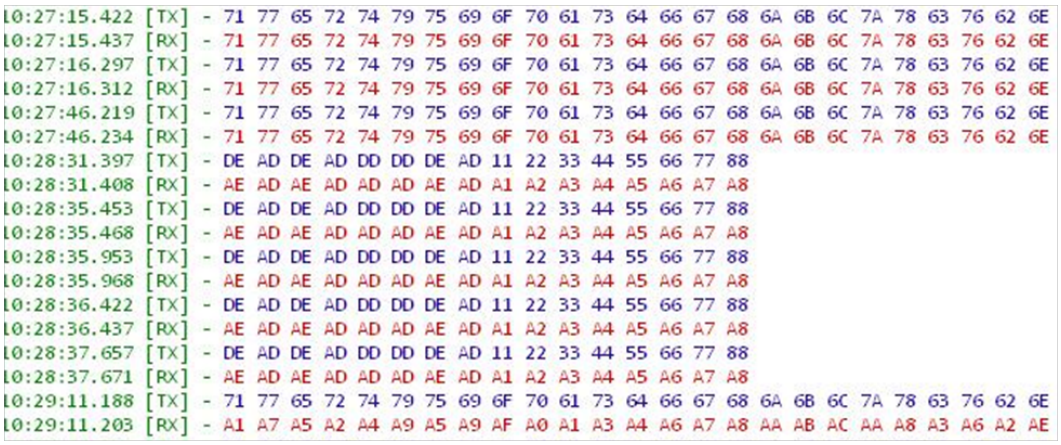

Figure 5.4: Frame corruption [self]

```
                    end
                     endcase
                     end //else
end //always
```

Effect of Trojan is shown in Fig 5-4 here in the setup UART design with Trojan inserted, is implemented on FPGA. UART is configured to work in loop-back i.e. what it receives is transmitted back. Initially design is transmitting and receiving as desired. When the sequence DE, AD, DE, AD is received design is replacing the most significant four bits of data bits in transmitting frame with hexa decimal number 'A'.

5.4.2 Counter based Sequential Trojan:

A large 32-bit counter implemented in the transmitter part of transceiver module which counts the number of transmitted data bytes. Once the count value reaches intended count Trojan gets activated. Trojan payload modifies or corrupts the transmitted data frame. This counter based Trojan in general goes undetected during simulation because to reach large count value takes longer simulation time. Counter is

implemented in the RTL of the design in verilog as shown in below code snippet.

```
reg  [31:0] count;
always @(posedge byte_transmitted or negedge resetb)
begin
if (resetb==1'b0)
count<=32'b0;
        else
        begin
                count<= count+1'b1;
                      if(count == 32'hFFFF_FFFF)
                         trig <=1'b1;
        end
end
```

Counter triggered denial of service pay load which modifies the control signal is demonstrated as shown in Fig 5-5 by implementing on the FPGA board. Here initially the UART design is functioning normally in loop back mode. Once set count which is kept very low for demonstration is reached, control signal responsible for transmission is made logic zero permanently. This causes transmission of UART to stop, though it is configured to transmit continuously. Thus counter based Trojan causing Denial of Service(DoS) is demonstrated.

5.4.3 State machine in transmit module:

This trigger is similar to the state machine Trojan trigger in receiver, after transmitting the DE, AD, DE, AD sequence Trojan trigger is activated. To detect this sequence a state machine is implemented in the transmitter part of the transceiver. State diagram is shown in Fig 5-6. Trojans in the design can be implemented as explained above or in combination of the above either modifying transmitted frame or causing denial of service or degrading the performance. Here the implementation of FSM in verilog is similar to that of implementation in receiver part. Payload can be either modification of transmitted

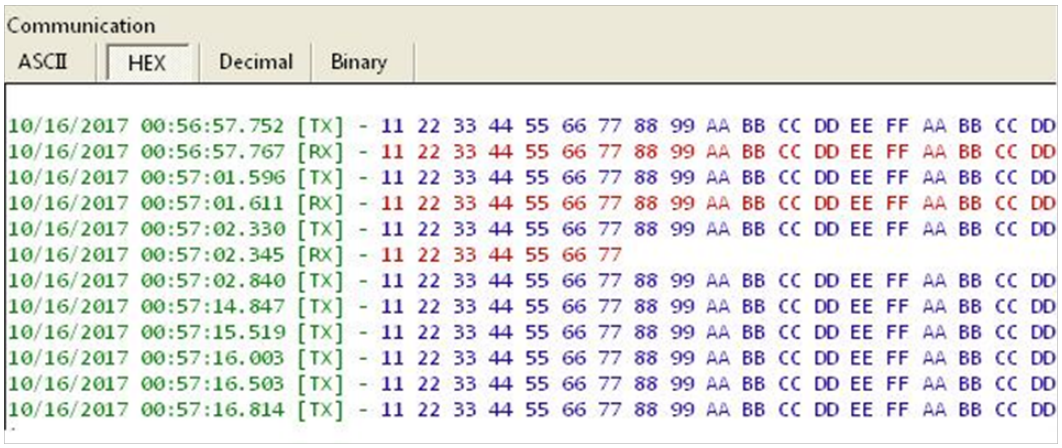

Figure 5.5: Denial of service [self]

frame or denial of service, Similar to the ones demonstrated in FSM in receiver part case and counter based Trojan case respectively.

5.4.4 Combinational Trojan:

In this case the trigger is activated by taking internal signals of the design. Once all these signals reach the targeted values the Trojan is activated. For implementation ten internal signals are chosen. When these ten signals takes certain values AND and XOR combination among them produces logic-1. When such a combinational circuit output becomes high for N- number of times Trojan gets triggered. Counter is implemented by taking output of combinational circuit as clock to counter. Trojan can have different payloads like modification of data bits in UART frame, denial of service, combination of these two or any other malicious modifications. Verilog implementation of trigger part is shown below.

```
wire cor1_en;
 assign cor1_en =( xor_tx_bits & xor_rx_bits &\\
 tx_parity_bit & rx_parity_bit & num_parity_bits\\
 & tx_bit_out)^  (start_byte_transmission & \\
 rx_uart_error & rxsample_count[3] & rxfsm_state[2]);
```

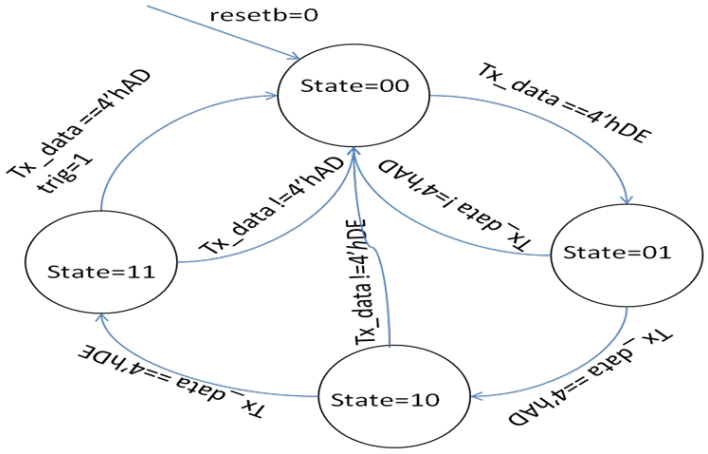

Figure 5.6: Trigger FSM in transmit module [self]

```
reg [15:0] count= 16'h0000;

always @(posedge cor1_en or negedge resetb)
begin
if(!resetb)
                begin
                        count <= 16'h0000;
                        cor<=2'b0;
                        cor1<=1'b0;
                end
else
        begin
                count<=count+1;
                if(count>=16'h002f)
                begin
                 cor <= 1'b1;
                 cor1<=cor;
                end
                else
                begin
```

```
                            cor<=1'b0;
                             cor1<=cor;
                    end
            end
end
```

5.5 Applying Detection Techniques

In the previous section how Trojans are inserted in the UART is presented. To detect these Trojans techniques discussed in third chapter have to be applied. Here I have planned to detect the Trojans using 1.side channel analysis using parameters power, delay and area 2.formal verification technique logic equivalence check.

5.5.1 Side channel parameter analysis:

For side channel parameter extraction from the design I am using industry standard Synopsys EDA tool Design compiler for estimating area, power and delay.

Though the design is implemented on FPGA because of proper tool set non-availability at my work place for FPGA I am using ASIC or SoC design tools. Since in side-channel-analysis metrics of deign under test are compared with the reference design, results will be similar for FPGA and ASIC with slight variation for example static power consumption for FPGA is more than ASIC or SoC for same design. Then percentage increase in static power for FPGA due to Trojan is less than for an ASIC. But if proven for ASIC there is high chance that it will be valid for FPGA in side channel analysis.

Working of the Design compiler is shown in Fig 5-7. It takes RTL design in HDL either verilog or VHDL, standard cell library and user constraints such as area, timing as inputs and translates them in to gate level net list in <design>.v from, synopsis design constraints and generates area, timing, power reports of design.

The standard cell library I am using is from global foundries 40 nm technology. Targeting design to 100 Mega hertz clock frequency, it can be any other frequency since getting optimum timing performance is not the criteria for these experiments.

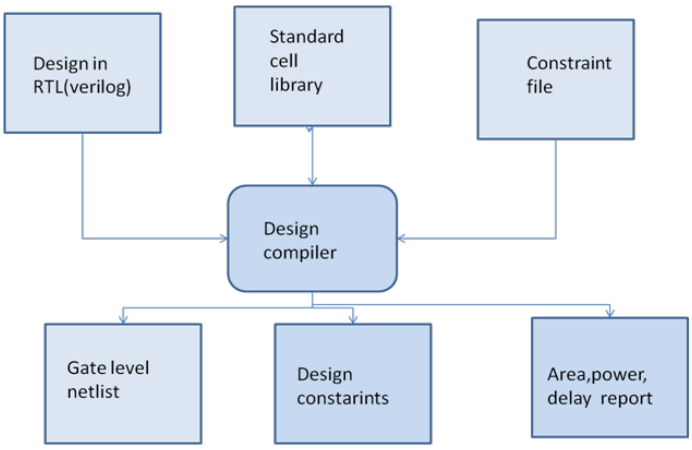

Figure 5.7: Synopsys Design Compiler synthesis flow [self]

Here below are reports generated from DC for Trojan free reference design.

Area Report:

****

Report : area
Design : uart_module
Version: K-2015.06-SP4
Date : Wed Nov 1 11:14:50 2017

****

Library(s) Used:
gf40npkhsst_ss0p99v125c (File: /global/../gf40npkhsst_ss0p99v125c.db)

Number of ports : 411
Number of nets : 2035
Number of cells : 1711
Number of combinational cells : 1209
Number of sequential cells : 496
Number of macros/black boxes : 0
Number of buf/inv : 256
Number of references undefined
Combinational area : 2089.752030
Buf/Inv area : 430.886399
Noncombinational area : 3039.019130
Macro/Black Box area : 0.000000
Net Interconnect area : undefined (Wire load has zero net area)
Total cell area : 5128.771160

Similarly reports are generated for designs with Trojans.

Area comparison:

Area comparison of the Trojan inserted designs with the original Trojan free design is recorded in tables from 5-2 to 5-5.

Table 5.2: Comparison of Rx_FSM Trojan present Design

Area	Original	FSM in receiver	Difference
Combinational	2089.752	2114.68323	24.9312
Buf/Inv area	430.8864	436.766399	5.88
Non combinational	3039.019	3055.24793	16.2288
Total cell	5128.771	5169.931159	41.16

Table 5.3: Comparison of counter Trojan present Design

Area	Original	Counter	Difference
Combinational	2089.752	2153.021	63.2688
Buf/Inv area	430.8864	437.2368	6.3504
Non combinational	3039.019	3136.392	97.3728
Total cell	5128.771	5289.413	160.6416

Table 5.4: Comparison of Tx_FSM Trojan present Design

Area	Original	FSM in transmitter	Difference
Combinational	2089.752	2108.803	19.0512
Buf/Inv area	430.8864	433.2384	2.352
Non combinational	3039.019	3055.248	16.2288
Total cell	5128.771	5164.051	35.28

Table 5.5: Comparison of combinational Trojan present Design

Area	Original	Comb	Difference
Combinational	2089.752	2158.43	68.6784
Buf/Inv area	430.8864	438.648	7.7616
Non combinational	3039.019	3136.392	97.3728
Total cell	5128.771	5294.822	166.0512

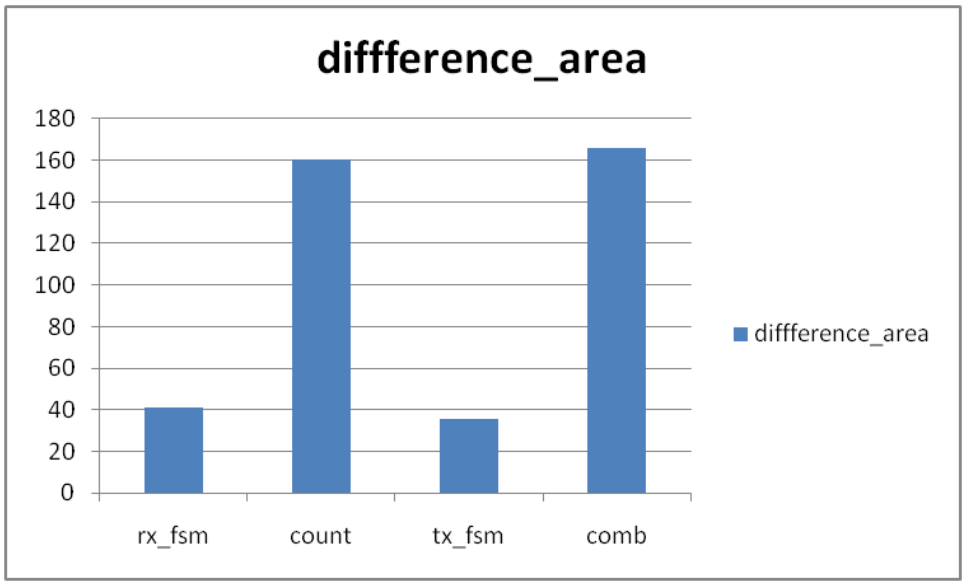

Figure 5.8: Area difference

Power Report: Power report of the original design form design compiler is presented below.

```
****************************************
Report : power
-analysis_effort low
Design : uart_module
Version: K-2015.06-SP4
Date : Wed Nov 1 11:14:58 2017
****************************************
Library(s) Used: gf40npkhsst_ss0p99v125c (File:../gf40npkhsst_ss0p99v125c.db)
Operating Conditions: SS0P99V125C Library: gf40npkhsst_ss0p99v125c
Wire Load Model Mode: top
Design        Wire Load Model       Library
------------------------------------------------
uart_module     B0.3X0.3        gf40npkhsst_ss0p99v125c
------------------------------------------------
Global Operating Voltage    =    0.99
```

Power-specific unit information:

Voltage Units	1V
Capacitance Units	1.000000pf
Time Units	1ns
Dynamic Power Units	1mW (derived from V,C,T units)
Leakage Power Units	1nW

Cell Internal Power	= 512.5878 uW (79%)
Net Switching Power	= 133.5580 uW (21%)
Total Dynamic Power	= 646.1458 uW (100%)
Cell Leakage Power	= 709.2640 nW

Power Group	internal Power	Switching power	Leakage power	Total power
register	0.5057	7.9911e-04	324.0785	0.5068
sequential	1.6724e-03	3.8341e-05	45.4842	1.7562e-03
combinational	5.0077e-03	0.1329	342.5647	0.1382
Total	0.5124 mW	0.1337 mW	712.1274 nW	0.6468 mW

Power Comparison:

Comparison of power profile of designs with Trojan is compared with the original design is presented in tables from 5-7 to 5-10.

Table 5.7: Power comparison of Rx_FSM Trojan present design

Power	Original	Rx_fsm_Trojan	Difference
Dynamic power(uw)	646.1458	650.1465	4.0007
Leakage power(nw)	709.2481	713.7266	4.4785
Total power(uw)	646.9	650.9	4

Side channel Analysis discussion:

For side channel analysis, parameters I have considered are power, area and delay. Of these area and power results from Synopsys Design compiler EDA tool are presented. As a standard practice these parameters of Trojan effected design are compared with the golden (original)

Table 5.8: Power comparison of counter trigger Trojan present design

Power	Original	counter	Difference
Dynamic power(uw)	646.1458	646.748	0.6022
Leakage power(nw)	709.2481	727.7484	18.5003
Total power(uw)	646.9	647.5	6

Table 5.9: power comparison of FSM triggered Trojan in transmit module design

Power	Original	Tx_fsm_Trojan	Difference
Dynamic power(uw)	646.1458	650.1481	4.0023
Leakage power(nw)	709.2481	712.9347	3.6866
Total power(uw)	646.9	650.9	4

Table 5.10: Power comparison of combinational Trojan present design

Power	Original	Comb	Difference
Dynamic power(uw)	646.1458	646.7421	0.5963
Leakage power(nw)	709.2481	731.6564	22.4083
Total power(uw)	646.9	647.5	6

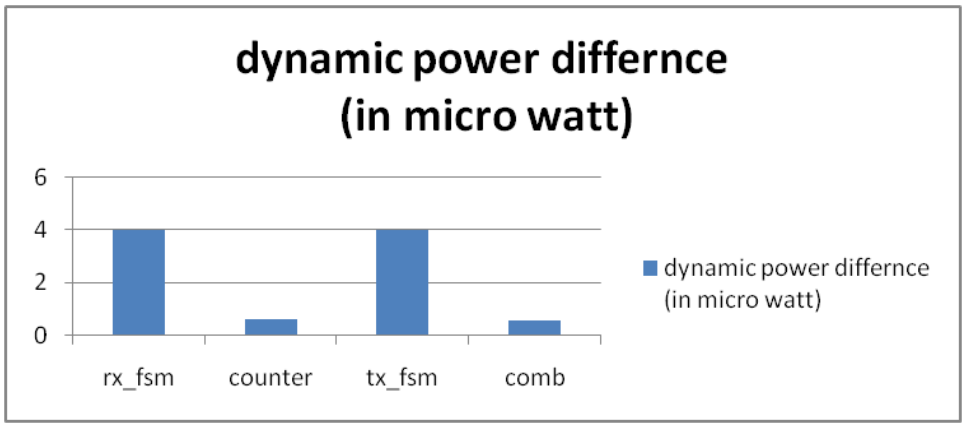

Figure 5.9: Dynamic power difference comparison

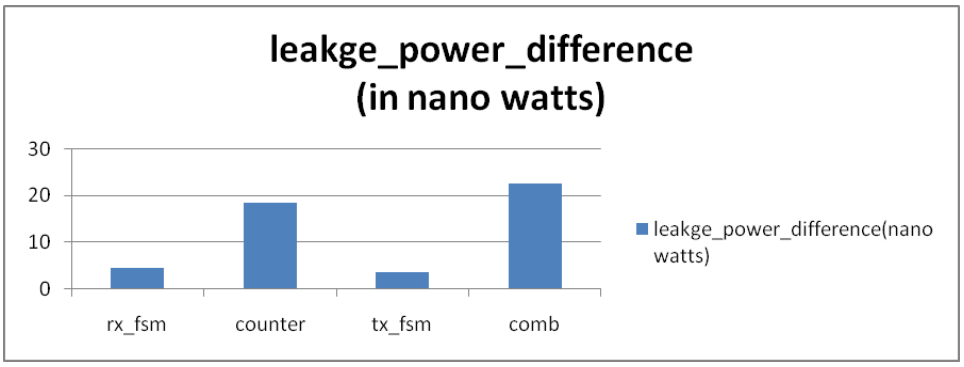

Figure 5.10: Static power difference comparison

reference design parameters. Four different Trojan effected design's side channel parameters, which are explained in the beginning of this chapter are presented. Following observations are made from this analysis:

Area:

1. Area of the Trojan effected design is increased significantly for counter triggered sequential Trojan and combinational Trojan in combination with counter.

2. Area of the Design with state machine based Trojan in transmitter and receiver is also increased slightly.

Power: Total power has two components one is Dynamic power and another is Leakage or Static power. Dynamic power is in the order of mill watt and Static power is in the order of nano watts. To differentiate the effect of Trojans on the Dynamic power and Static power components in Total power these two powers are compared separately.

1. Leakage power of FSM based Trojans increased slightly.

2. Leakage power of counter based and combinational Trojan in combination with counter increased significantly.

3. Dynamic power of FSM based Trojan increased significantly.

4. Dynamic power of counter based and combinational Trojan in combination with counter increased slightly.

Path delay When delay of max path and min path are observed there is no change in the delays of Trojan effected design from reference design. This is due to the fact that Trojan is inserted in a path which is not in the maximum or minimum path. If suspicious delay paths are identified and compared instead of only comparing Maximum delay and Minimum delay paths then Delay comparison may detect the presence of Trojan but that exercise is not taken up here.

Side channel analysis Summary:

Considering area, delay and power as metrics for side channel analysis, analysing results from four Trojan infected designs each Trojan will effect parameters differently. For example FSM based Trojans effect dynamic power more where as counter based and combinational Trojans effect leakage power. So to detect the presence of Trojan multiple parameters have to be considered instead of relying on analysing single parameter. Multiple parameter analysis increases the chance of Trojan detection. But the demerit of these side channel analysis methods is they need a reference Trojan free design parameters.

5.5.2 Logic Equivalence Check:

Logic equivalence check is formal verification method to compare two designs for their equivalence logically. For this industry standard EDA tool from CADENCE Conformal LEC version 14.20-s140 is used.There are two phases in performing LEC. One is setup mode and the other is lec mode.

Setup mode: In this mode Trojan free reference design (Golden) and design with FSM Trojan in receiver(Revised) are read.After reading designs, the environment is set to compare the two designs read into it. Since the two designs are in RTL no library is required to be read in. The screen shot of conformal tool after reading design for comparison is shown in Fig 5-11. Once setup is done mode is changed to lec mode. While moving from setup to lec mode tool models reference design and design,for comparison maps the key points from one to other.

LEC mode: In LEC mode comparison of two designs is done. And

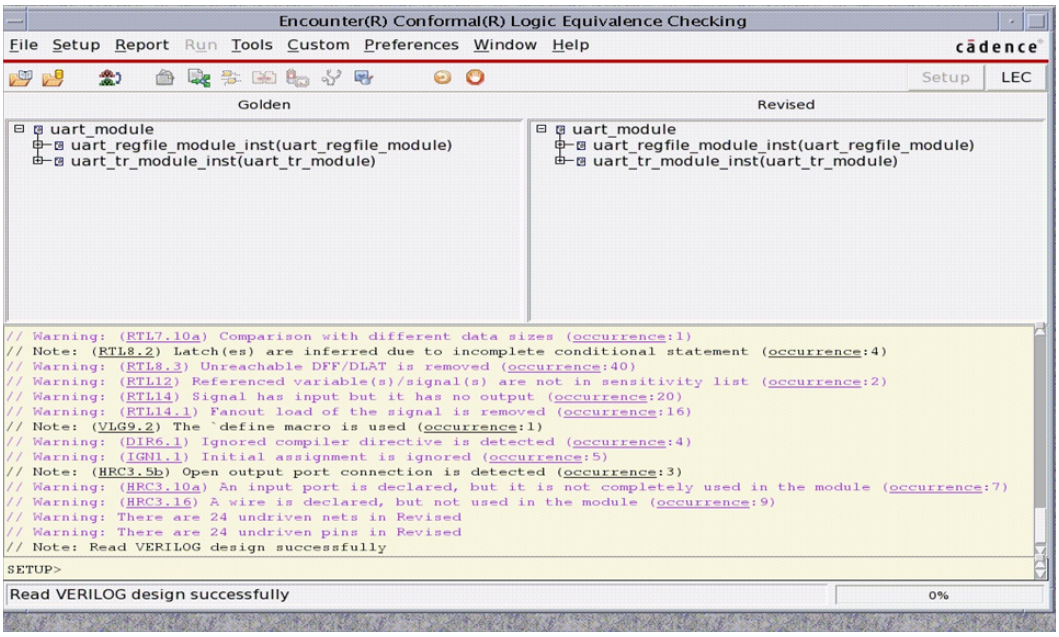

Figure 5.11: Conformal LEC EDA tool setup

report of the comparison is generated in this mode. Comparison can be done hierarchically or in Flat. Here flat comparison is done. Here tool reports all key points are added for comparison. Result of the comparison is shown in Fig 5-12. All compare points are mapped into three categories mapped, unmapped and not equivalent. Unmapped points are either unreachable means they do not impact the output or not mapped means they are not present in one of designs being compared. In unmapped points three not mapped points in revised design are present which are not present in original design. These are added extra in the Trojan infected design. Four non equivalent points are present. Which mean that they are present in both reference and infected designs but they are getting different inputs or connected to different out-puts. If source is inspected for these key points Trojan logic was found in modified design. Thus the Trojan can be detected using LEC. For this to be done reference design free from Trojan is required.

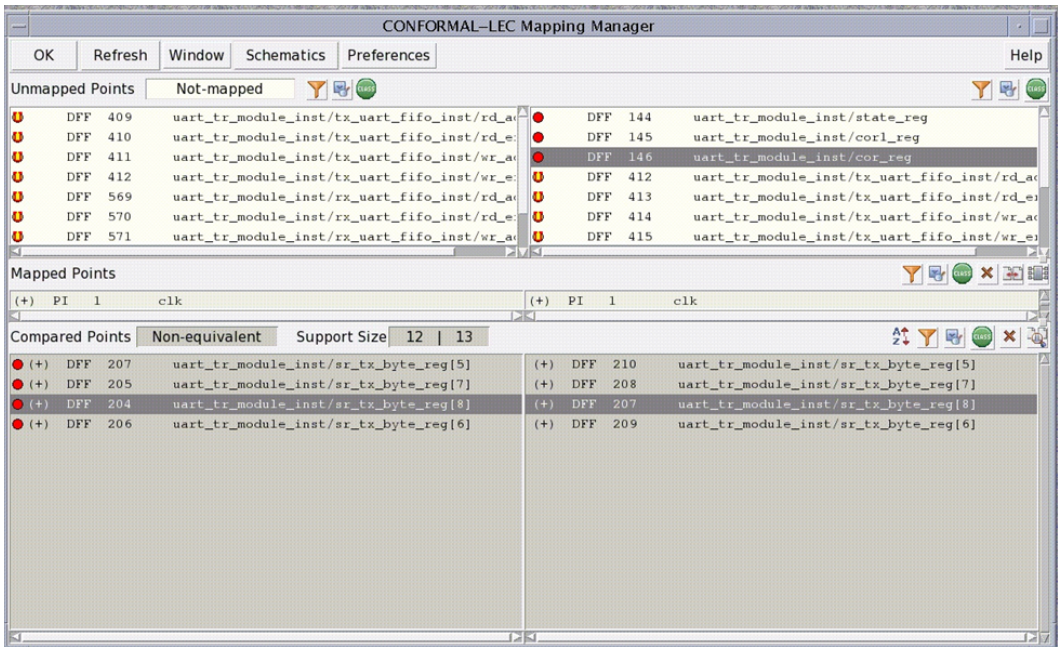

Figure 5.12: LEC comparison result

5.6 Chapter Summary:

In this chapter FPGA implementation of a UART serial communication, four different kinds of trigger part of Trojan and two different types of Trojan payload implementation details are discussed. Effect of these Trojans is demonstrated by sending/receiving data from PC's hyper terminal to/from FPGA board. To detect the presence of Trojan in the design side channel parameters area, power, and path delay parameters are analyzed. Area and power analysis indicated presence of Trojan but path delay analysis of max paths could not detect the Trojan. One more technique logic equivalence check(LEC) a formal verification method is used to detect the presence of Trojan. LEC method could detect the Trojan in an infected design. But for both the side channel analysis and LEC methods Trojan free reference design is required.

6 Conclusions

This thesis explored the possibilities of hardware Trojans presence in System-On-Chip (SoC), Application Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA). ASIC and SoC being similar in design and fabrication except in terms of size, what applies to one is applicable for the other. FPGA being different from ASIC/SoC has different possibilities for Trojan insertion. In SoC Trojan can be present in single IP core or it is because of two or three IP cores collusion. Trojan can modify the functionality during run time, cause performance degradation, leak information or can cause denial of service. Trojans can be inserted at any phase of design while undergoing translation from one abstraction to another abstraction or during fabrication.

This thesis also explored the various detection techniques. Detection techniques broadly classified into two types 1. Destructive 2. Non-Destructive. In this thesis non destructive techniques using side channel analysis using delay, area, power as metrics are discussed.Ring-oscillator network where ring oscillators are distributed across entire design area,which uses the principles of side channel analysis is discussed in detail. If Trojan is present, it does consume some power and changes frequency of ring oscillator in proximity to Trojan. For side channel analysis a reference design is required which is free from Trojan. If Trojan is inserted in the beginning stage of development i.e. in RTL then to get reference Trojan free design is difficult. When source of IP for SoC, ASIC or FPGA is in RTL form then side channel analysis techniques are ineffective. FANCI technique is useful for analyzing un-trusted IP to identify suspicious code.

To show the proof of concept a design UART serial transmission model is chosen for Trojan implementation and detection in FPGA. This design is first implemented on FPGA to show it is functionally working properly. Four different Trojan triggers, two different types of payloads one modifying the data bits in transmitting frame for example replacing 4 most significant bits with 4'hA, second causing Denial of Service(DoS) are implemented. Till the Trojan is triggered it is demonstrated that the design is functioning as intended to be, this makes

Trojan stealthy. Once Trojan is triggered payload shown its malicious effect.

To detect the Trojans two methods are applied one is side channel analysis and the other is a formal verification technique Logic Equivalence Checking (LEC). In side channel analysis to estimate the powers ASIC development EDA tool Synopsys Design compiler is chosen as FPGA tools for estimation of power are not available. Comparison of parameters area and power of Trojan present designs with original design indicated the design modification whereas parameter path delay could not detect as only maximum delay paths are considered. LEC of original and Trojan infected design is carried out using Cadence EDA tool Conformal LEC. In this method extra logic elements present in the infected design and logic elements having modified input are detected.

It is proved that if reference Trojan free design is available, side channel analysis and LEC method can detect or at least give a reason to suspect presence of Trojan. But limitation is, in general getting reference Trojan free design is always not feasible.

6.1 Future work:

To detect the Trojan presence in an un-trusted IP cores techniques like FANCI, unused circuit identification have to be used. Though the concept of these techniques is explained in the thesis EDA tools are not present to apply on the implemented design. If any similar tools become available they can be applied on the Trojan suspected un-trusted IPs.

Bibliography

[1] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: threat analysis and countermeasures," Proceedings of the IEEE 102, no. 8 (2014): 1229-1247

[2] Adam Waksman, Matthew Suozzo, Simha Sethumadhavan "FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis", ACM.CCS'13.

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

[4] Kaige Qu, Liji Wu*, Xiangmin Zhang, "A Novel Detection Algorithm for Ring Oscillator Network Based Hardware Trojan Detection with Tactful FPGA Implementation", 2015 11th International Conference on Computational Intelligence and Security.

[5] AbulSarwar, "CMOS Power Consumption and Cpd calculation", Texas instrument report.
http://www.ti.com/lit/an/scaa035b/scaa035b.pdf

[6] http://www.vlsi-expert.com/2011/03/static-timing-analysis-sta-basic-timing.html

[7] http://www.vlsi-expert.com/2011/04/static-timing-analysis-sta-basic-timing.html

[8] http://allaboutfpga.com/fpga-architecture/

[9] Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor. 2010. Trustworthy hardware: Identify-ing and classifying hardware Trojans.Computer43, 10 (Oct. 2010), 39–46.

[10] https://www.researchgate.net/profile/Michael_Hsiao2/publication/264124590/figure /fig3/AS 295835645562884@1447544122632/Fig-3-Hardware-Trojan-attacks-in-different-forms-a-combinational-and-sequential.png

[11] Abhishek basak,swaroop bhunia,Thomas tkacik," Security Assurance for System-on-Chip Designs With Untrusted IPs", IEEE transactions on information forensics and security, vol. 12, no. 7, july 2017

[12] Nicole Fern, Ismail Sany, Cetin Kaya Koc, and Kwang-Ting (Tim) Cheng, "Hiding Hardware Trojan Communication Channels in Partially Specified SoC Bus Functionality", IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems.

[13] https://www.vlsisystemdesign.com/i-am-hidden-but-i-exist-dynamic-power-dissipation-in-cmos/

[14] Defence Science and Technology Organisation, DSTO–TN–1012 (Australia Government): Hardware Trojans – Prevention, Detection, Countermeasures. Edinburgh, South Australia 5111, Australia.

[15] Yier Jin, Nathan Kupp, Yiorgos Makris: Experiences in Hardware Trojan Design and Implementation. 2009 IEEE International Workshop on Hardware-Oriented Security and Trust.