Indian Institute of Engineering Science and Technology,
Shibpur

Hardware Security - Trojan Horses

Anthony Rajiv Francis 510517017

Swagata Kundu 510517018

Supervisor: Prof. Biplab Kumar Sikdar

Final year project report

2020 - 2021

Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. We understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: Swagata Kundu

Name: Anthony Rajiv Francis

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of my project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank our mentor Prof. Biplab Kumar Sikdar, for providing us an opportunity to do the project work and giving us all support and guidance which helped us complete the project duly.

We owe our deepest gratitude to our project guide, Nilanjana Das who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information and assistance whenever we needed.

Contents

List of Figures

Chapter 1

Introduction

Hardware security primitives play an important role in ensuring trust, integrity, and authenticity of integrated circuits (ICs) and electronic systems. Nowadays, Integrated Chips (ICs) have been used in the information infrastructure and systems that contain confidential information, for example, financial information, secret key data. It is necessary for these ICs to have high secrecy. Globalization during the IC manufacturing decreases control of a designer on the fabricated chips. The designer can not entirely handle the whole process of designing as well as manufacturing. Therefore, it is extremely difficult to control the whole process of an IC design. The designing phase is likely to be fabricated by the attackers through the plantation of Hardware Trojans (HTs).

Several untrusted authorities during the manufacturing phase can potentially compromise an ICs functional and/ or parametric behavior such that it can evade conventional IC testing approaches. Such alterations of a design can be introduced at different stages of the IC manufacturing phase, e.g. malicious insertion in an Intellectual Property (IP), modification of netlist or tampering specific files during fabrication. These can have adverse effect during in-field operations, especially in the security-critical applications such as defence, communication and national as well as international infrastructure.

Chapter 2

Microchip supply chain

We cannot and should not trust the hardware we are given, no matter it is a cell phone, it is a laptop, it is a desktop, or maybe it is just a very simple embedded system. There are a couple of reasons behind this. First is the trust issue in microchip supply chain. There might be back doors in the server or in the system. When designers design the system they have to use third party intellectual properties. They have to use design tools which may not come from a trusted source.
The following chip manufacturing process has been collected from DARPA.

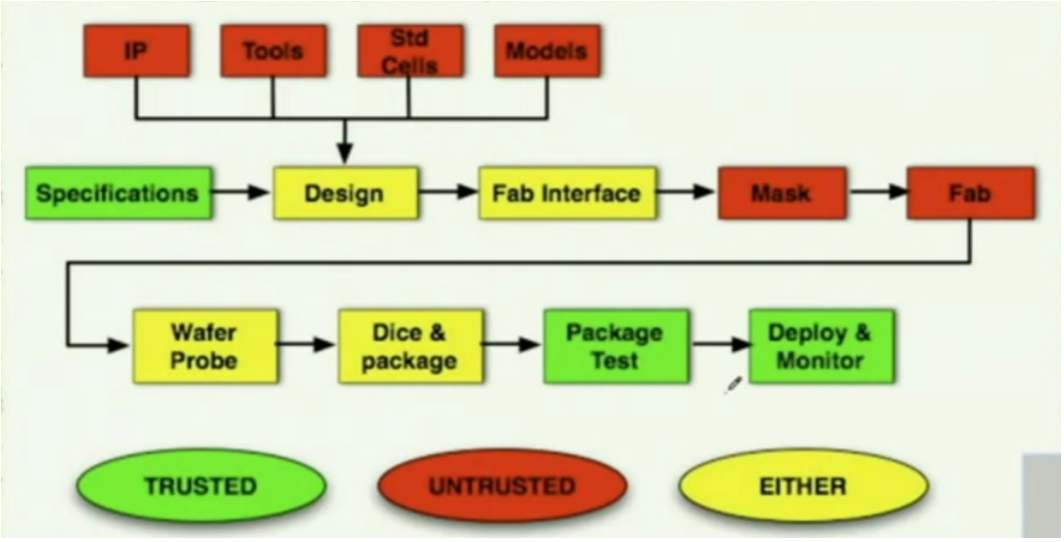

Figure 2.1: *A typical microchip manufacturing chain*

We can see that trust becomes an issue with offshore foundry and design complexity.

Chapter 3

Design vulnerabilities

Let's understand this with examples.

3.1 Design vulnerability in Combinational Circuit

A 3-input encoder that assigns a 2-bit code (as input for the next module) to each of the three different inputs.

x	y	z	a	b
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

Figure 3.1: *Truth table*

An optimal design of the encoder would be: a = z, b = y
But then, with this design several problems may arise.

1. On input 000, the output is 11 which coincides with the output for 100 according to our optimal design. But for the given encoder the input 000 is not an acceptable input. This results in a backdoor to the case of input 100

2. On input 011 or 111, the output is 00 according to our optimal design and this output will be fed to the next module. But as per our encoder, the output 00 is not desired. This results in a fault injection attack to the next module.

3.2 Design vulnerability in Sequential Circuit

We have a three state finite machine with states 00, 01, and 10.

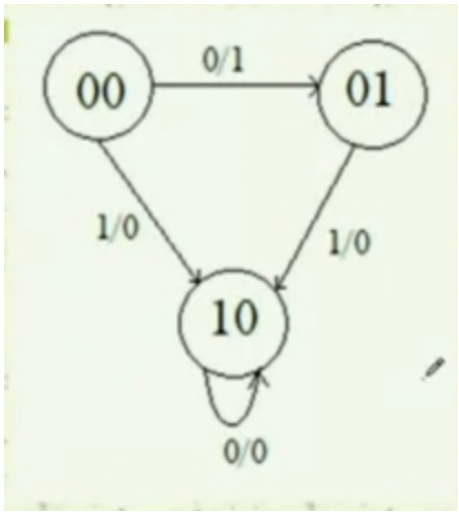

Figure 3.2: *State transition dig*

A	B	x	A'	B'
0	0	0	0	1
0	0	1	1	0
0	1	0	-	-
0	1	1	1	0
1	0	0	1	0
1	0	1	-	-

Figure 3.3: *State transition table*

The circuit needed to implement this finite state machine is something like

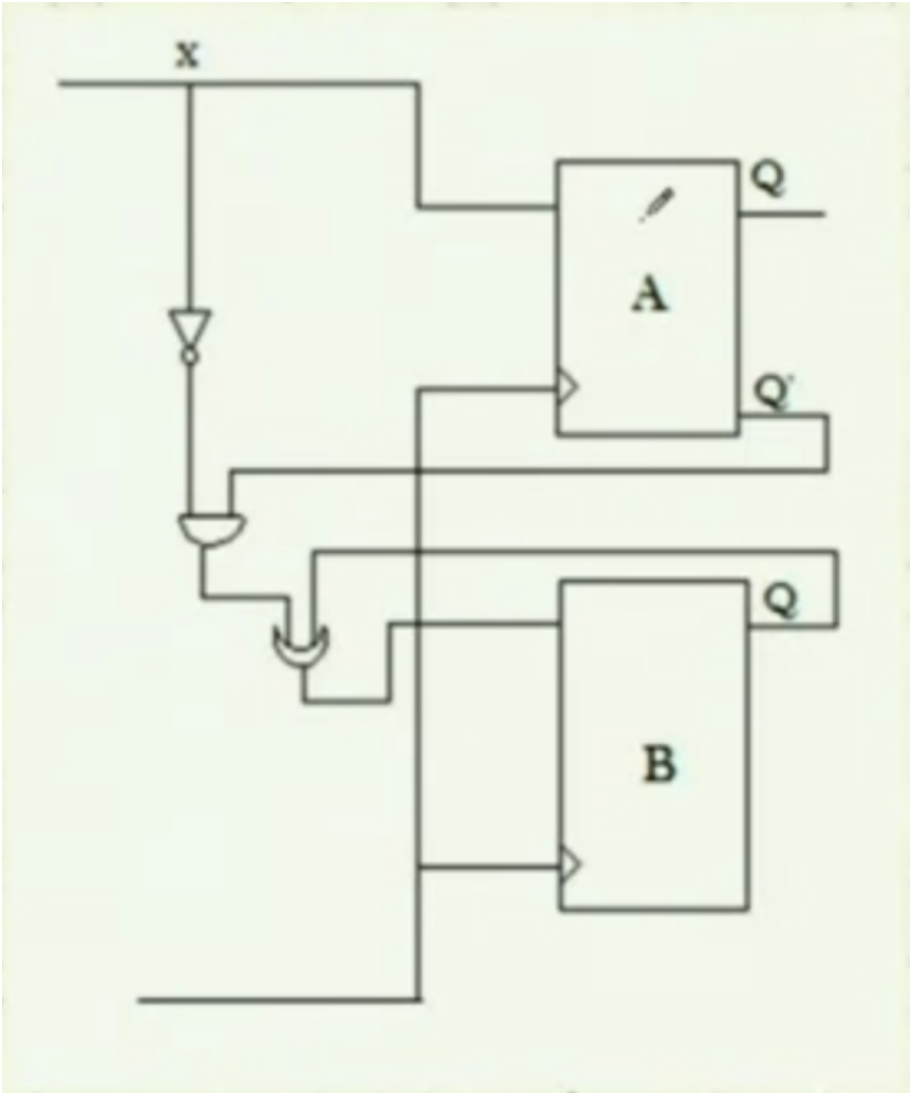

Figure 3.4: *Circuit diagram*

Here we have two flip flops A and B and a couple of logic gates.

And what is interesting here is what happens with the two dashes in the 3rd row of our state transition table. Here, 01 is our current state. Then if input is 0, we don't care what the next state is. But, from the circuit here, we cannot generate don't care. The circuit instead directs us to a deterministic state 00.
And similarly in the other don't care case, output is also 00, which is the next state.

We can have another state, state 11, which is not specified in the original system specification here. The next states of state 11 can be one of the two states shown in the table below.

A	B	x	A'	B'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	0

Figure 3.5: *State transition table of the given circuit*

So this leads us to the following finite state machine model for the system.

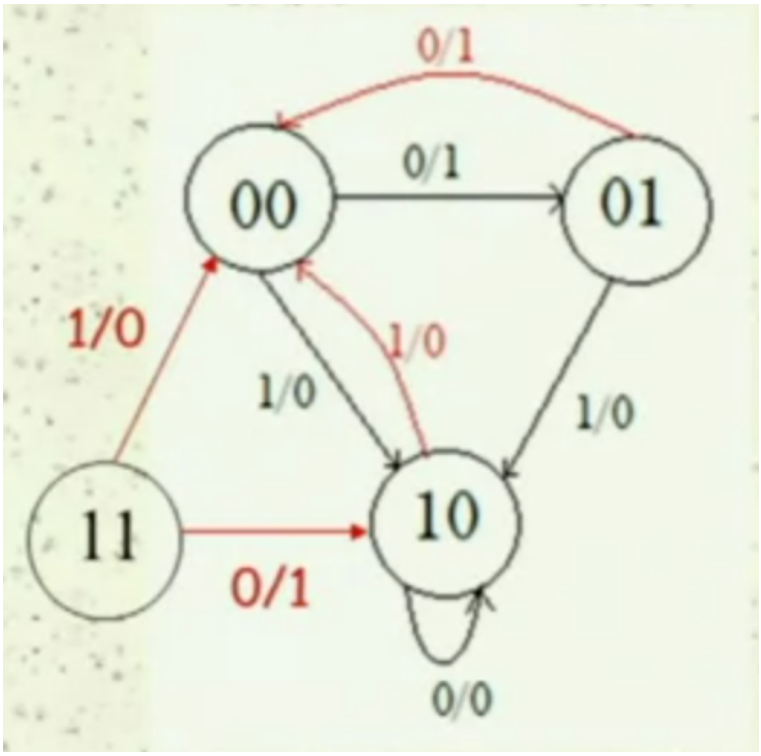

Figure 3.6: *An alternate state transition diagram which seems equivalent to fig 3.2*

But this has got some backdoors. In the original system there's no incoming edge coming into 00. However in this new state machine we can reach state 00 from all the states 00,01,10,11. So this is the potential vulnerability.
We can explore a random walk attack in which the attacker is going to start from a random state, for example, 01. And then he is going to try a random input for example 1 which will move him into state 10. Since its not the desired state, he is going to try another random input, for example 0. And then this one stays at state 10, still not the target at 00. And next time the attacker is going to try input 1, and this will take him directly to the state of 00. Which means that the attacker now has access to the state 00 which he shouldn't have access to.

3.3 A proposed solution

Here's a model that is safe and satisfies our need.

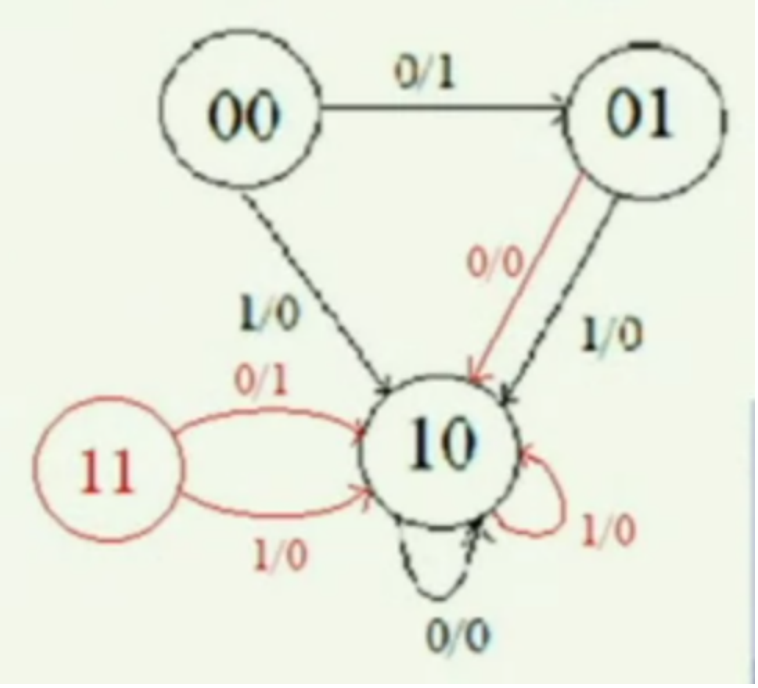

Figure 3.7: *A safe model*

So in this case, we do have 100% trust, where we do have 100% security. However, there is a huge design overhead, which normally we cannot afford.

Chapter 4

Hardware Trojans and trusted IC design

4.1 Hardware Trojans

Hardware Trojans are any additions or modifications to a system or to a circuit for malicious intentions.
The most common malicious goal for Hardware Trojans includes, trying to change the circuit's functionality, trying to control the system, or trying to steal sensitive information such as secret key from the system. Also, some of the hardware Trojans are trying to reduce the circuit's reliability, and by doing this a system will start malfunctioning before it's lifetime expires.
A hardware Trojan must be added intentionally. So this basically tells us that in the traditional designs, we have a lot of don't cares as we have discussed in Chapter 3, so we can introduce a lot of back doors as security vulnerabilities during the design. These are not intentionally introduced into the system, so they are not considered as hardware Trojans.

4.2 Trusted integrated circuits

Trusted integrated circuit is an integrated circuit that does exactly what is asked to do, no more and no less.
If a system fails to deliver certain required functionalities, such system cannot be trusted.
If a system has hardware Trojan inside it, it cannot be trusted either because this system does something more, and these things are malicious hardware Trojans.

Chapter 5

Hardware Trojan Taxonomy

Based on different criteria the hardware Trojan's can be categorized in many different ways. These criteria include how the hardware Trojan will be triggered or activated? And what will the hardware Trojan do once it is activated? Or what is the goal or payload of the hardware Trojan? In which system components the hardware Trojan will be inserted such as the processor, memory unit, or clog networks? Whether the hardware Trojan is functional or non-functional? In the case of non-functional, it is also called parametric hardware Trojan. Are the hardware Trojans big or small? Tight or loose? Tight means that the hardware Trojan is clustered or centralized at one spot on the chip, and loose means that the hardware Trojans are distributed all over the chip. Does the insertion of the hardware Trojan require redesign of the layout or not?

5.1 HT Taxonomy: Type

Based on whether the hardware Trojan changes system's functionality or not, they can be considered as functional hardware Trojan or parametric hardware Trojan.
Functional hardware Trojans are those that will change the system's functionalities. Examples include adding a actual functional unit, removing or modifying systems components.
Parametric Trojans on the other side, on the other hand, normally do not change system's functionalities. Instead, they try to reduce system's reliability to increase the chance of performance failure.

5.2 HT Taxonomy: Activation method

Hardware Trojans can be always on or triggered by some event or signals.
The Trojans such as those based on the modification of physical features of the transistors, wires or other hardware components, are always on.
Hardware Trojans can also be triggered, and most of these cases, triggered by rare event or signals. These triggers can be data collected by the sensors or logical units from the hardware components, such as registers or counters. The triggers can also come from either internally or externally from outside of the chip.

5.3 HT Taxonomy: Effect or Payload

The payload is the action or the damage that a hardware Trojan will do once it is activated. There are three major types of damage or malicious purpose a hardware Trojan may have.
First it may try to change or control the functionality of the system.
Second, a hardware Trojan may try to leak sensitive information. Normally this is done through side channels, such as the power network, the clock network, optical, thermal, or electrical, electrical magnetic emissions.
Finally, some hardware Trojans may try to reduce circuit reliability or the lifetime of the system. These type of Trojans include parametric hardware Trojans or those Trojans that can activate applications or functional units that will drain resource from the system, to make system die earlier.

5.4 HT Taxonomy: Locations

Hardware Trojans can be embedded into many locations of the system.
In the processing units, Hardware Trojans can be hidden to change or control the systems functionality.
In memory structures, hardware Trojan can be embedded to alter or to change the memory's contents, or simply to monitor memory activities and leak information.
The I/O devices are the link between the outside world and under the chip. So hardware Trojan can be embedded there to control, monitor, or modify data communication between the chip and the outside.
Power supply units, Hardware Trojan can be embedded there to change the power or current supply to cause system failure or to leak information through the power side channel.
And the clock grids is another position to hide hardware Trojan. The attacker can change the frequency to cause fault or failure, and to leak information through the timing side channels.

5.5 HT Taxonomy: Physical features

Based on the physical features of hardware Trojans, the hardware Trojan can also be put into several different categories.

5.5.1 Size

So first based on their size, we can have large or big hardware Trojans such as sophisticated time bombs or powerful antennas embedded or hidden in the system, trying to receive signals from outside.
There can also be a lot of very small hardware Trojans, such as some very small temperature sensors.

5.5.2 Layout

Sometimes when hardware Trojans are embedded into the system, we need to redo the layout or placement and examples of this are including adding functional units which are nontrivial

ones, cannot fit into a small extra space on the layout. There are also hardware Trojans that doesn't need to change the layout. For example, the parametric hardware Trojans or some very small hardware Trojans that can utilize the actual white space on the chips layout.

5.5.3 Distribution

Based on the distribution of the hardware Trojans, they can be either centralized or distributed. The centralized hardware Trojans normally are bigger, they can take over bigger space on one spot of the chip. And the distributed hardware Trojans are normally small and are scattered all over the chip.

Chapter 6

Hardware Trojan Detection

6.1 Logical test based HT detection

In the logical test based Hardware Trojan detection, we run different test patterns, which are called the test vectors and we monitor the systems, the output and the behavior. If the system has Hardware Trojan, when the Trojan is activated, it's malicious behavior, or malicious payload will be observed and it can be caught.
However, a full coverage test is impractical for reasonable size of the circuit. For example, if we have a combinational block with n input then there potentially will be the 2n̂ test vectors. For a sequential circuit if we have m flip flops and n input, then the total number of test cases will be 2 to the power n plus m.

6.2 Random test

Random test based Hardware Trojan detection mechanisms will fail because we have learned Hardware Trojan normally are triggered by a rare event. When you do a random selection, these rare test vectors may not get selected.

6.3 Side Channel Analysis

we monitor the side channel information during the execution of the system at the test-time. If the system has Hardware Trojan, then its presence will show on certain physical parameters, and can be observed through side channels. This is pretty good, because they not only capture functional hardware Trojans, they can also capture non-functional Hardware Trojans like the parametric Hardware Trojans.

6.3.1 Power Side Channels

We can measure the supply current at the quiescent stage, when the circuit is not switching and the inputs are also not changing. In this case, if there is there is a Hardware Trojan, the Hardware Trojan circuitry will consume additional leakage power that can be caught. However, this may also have false alarm rate, because nowadays chips have a very high leakage

curve.

Similarly, we can also measure the transient, supply current. When the Hardware Trojan is activated, it has additional switch activity, and this will cause a increase in dynamic power, so it can be caught. For this one to work, we need to find ways to activate the Hardware Trojan, or at least activate a part of the Hardware Trojan.

Limitations: it fails to capture small Hardware Trojans or always-on Hardware Trojans because you cannot distinguish this from the power side channel trace. And also, they are very sensitive to noise to errors and also to fabrication variations.

6.3.2 Delay and other side channels

We know that Hardware Trojan can change delay of a path either gate or wire or both. In the parametric Hardware Trojans, the change of the wire or the change of the transistors may change the delay, as well.

Limitation: So the limitation of path delay based Hardware Trojan detection mechanism is we can't measure delay for all the paths. For us to measure the delay of a path, we need to be able to observe or to see both the starting gate and the ending gate of the path. And also the, just the path delay measurement will be very sensitive to fabrication variation and other noises.

6.4 Run-time monitoring

The basic idea is simply to monitor the execution of the system at real time. If the system has Hardware Trojan and the trojan gets activated, then we can find the malicious behavior. However in this case most likely there will be an interrupt mechanism coupled with the run time monitoring system. Once the Hardware Trojan is detected the interrupted mechanisms will stop the execution to protect the system. As we know that at test-time we cannot detect all the Hardware Trojans with 100% guarantee. Therefore run-time monitoring is a very good complementary approach to test-time approaches.

Chapter 7

Trusted IC design with HT prevention

7.1 Pre-synthesis techniques

When we use third party IPs, we want to ensure they don't have hardware Trojan and they are trusted before we can use them. So typically there are two approach to do this. We can turn this problem into a formal verification problem. We can verify whether the IP meets the specification or not. This can be done by doing property check, model check, or equivalence check. For example, if f(x) and g(x) are two combinational logics

$$f(x)=g(x) \text{ iff } f(x)g'(x) + f'(x)g(x) = 0$$

This can be verified by the Boolean satisfiability solvers trying to verify this entity is unsatisfiable, which means it is always a 0.

And for sequential components we can do the final state machine equivalence check. Two finite state machines they are stated to be equivalent if and only if they give the same output of any input sequence.

7.2 Post-synthesis techniques

There are several post-synthesis techniques to prevent HT after the chip is built or when the layout is given.

1. Removal of dead spaces, when the layout is given, there are white space, the spaces that are not utilized. And those are the potential spots for hardware Trojan insertion. So we can add the dummy logic trying to take away this white space to prevent direct hardware Trojan insertion. This is particularly effective to prevent big hardware Trojans.

2. We can also do circuit obfuscation trying to make reverse engineering harder. So the attackers will not be able to understand the circuit and then it will be hard for them to insert

a functional hardware Trojan.

3. And we can also do shielding of the wires. This will be able to prevent the hardware Trojan based on electrical magnetic radiations.

4. And because most of the hardware Trojans require a trigger signal, we should pay special attention to interface protections. This interface includes the I/O pins of the system, the internal connections between modules, the power in the clock networks and also the scan chains. So once we will be able to control this, we can monitor this interface and then whenever there is a suspicious communication we can check whether they are required or they are the hardware Trojan's triggering signal.

7.3 Popular HT prevention approaches

7.3.1 Rare event removal

This is based on the observation that hardware Trojans use rare event as their triggers. If we can remove rare events it will be easier to detect or activate the hardware Trojans.

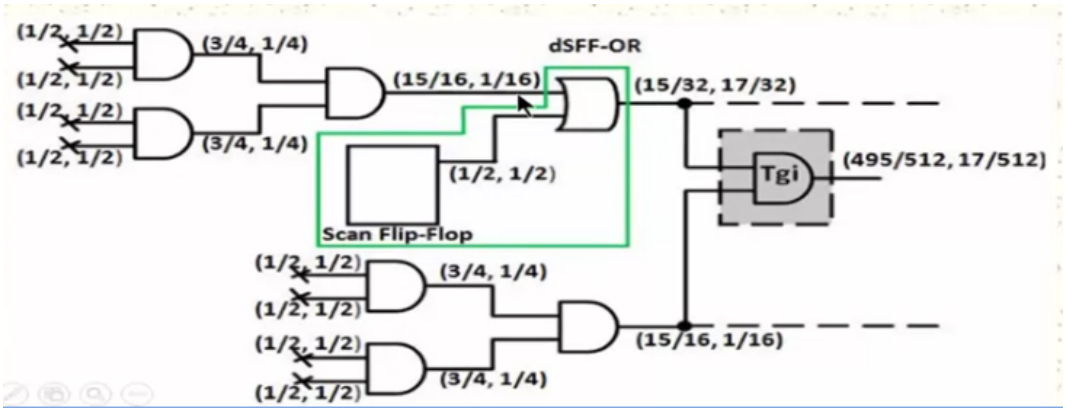

Figure 7.1: *Rare event removal*

7.3.2 Shadow register

Shadow Register approach will greatly facilitate the the measuring of the internal path delay. So we have the source register, we have the destination register. To measure the parts that delay may not be easy because we may not be able to see this signals. So in this approach we we put a shadow register which has the same clock as the system clock. However, the face of the cog might be different. We call the clock 1 as system clock and clock 2 will be called the shadow clock, which is used to drive the shadow registers. So, initially, the shadow register and the destination register will have the same face. They will have the same content, the same value. And we can deliberately change the face of the shadow clock, and then eventually it will become different. This can be monitored through the result bit. When they become different, the comparison result will be different. So this significantly

improve the observability or visibility of this internal pass, and it can be used to facilitate the hardware Trojan detection.

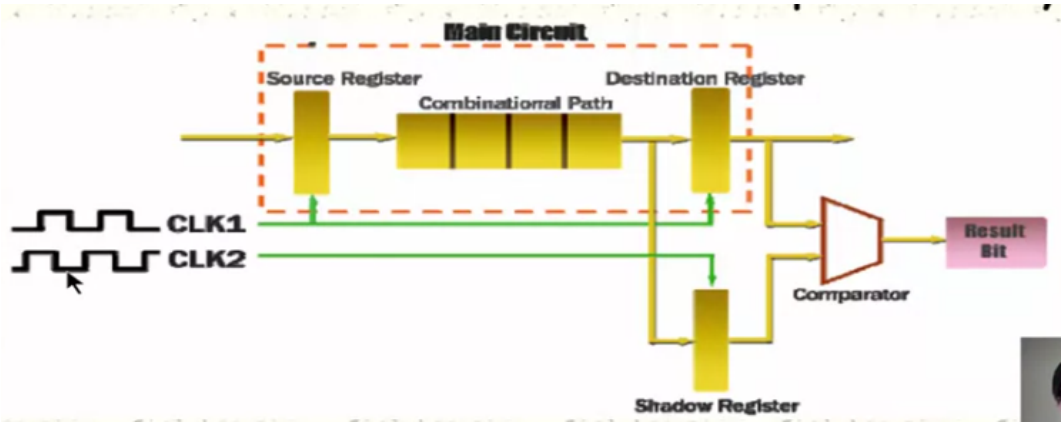

Figure 7.2: *Use of shadow register*

Chapter 8

References

1.Hardware security - University of Maryland

2.Exploring hard to detect sequential hardware trojans

3.Hardware Trojan Exploiting the FiF-FoF Values of Machine States

4.Hardware Trojans - Wikipedia