Andy, Yu-Guang Chen

EE6094 CAD for VLSI Design
Programming Assignment 4 (Due: 23:59:59, 2022/06/09)

Introduction

Hardware security to ensure Trust in ICs has emerged as an important research topic in recent years. Economic reasons dictate that most of the modern ICs are manufactured in off-shore fabrication facilities. Each party associated with the design and manufacture of an IC can be a potential adversary who inserts malicious modifications, referred as **Hardware Trojans**. [1]

Concern about this vulnerability of ICs and the resultant compromise of security has been expressed globally, especially since several unexplained military mishaps are attributed to the presence of malicious Hardware Trojans.

Background

The basic model of a hardware trojan circuit can be broken down into two major components: (a) Triggering and (b) Payload activation logic. These can also be divided into digital and analog. The attacker will design both of them, according to the malicious behavior which negatively affects a circuit or even the overall system to achieve their purposes. Typically, the adversary would design a Hardware Trojan that triggers a malfunction only under very rare circuit conditions to evade detection.

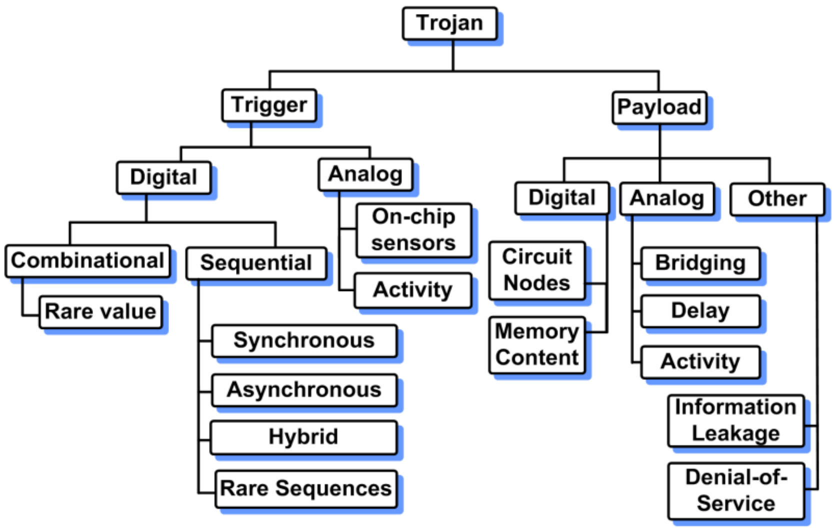


Figure 1. Hardware Trojan taxonomy based on trigger and payload mechanisms. [2]

In this assignment, we focus on digital trigger (combinational & sequential) and payload (circuit nodes). It will contain several related HT designs, such as: Combinationally triggered Trojan, Sequentially Trojan, and Hybrid Trojan.

Andy, Yu-Guang Chen

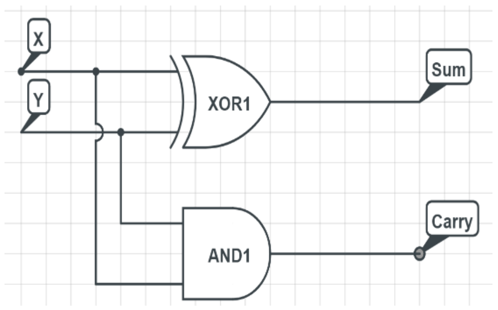
Problem definition

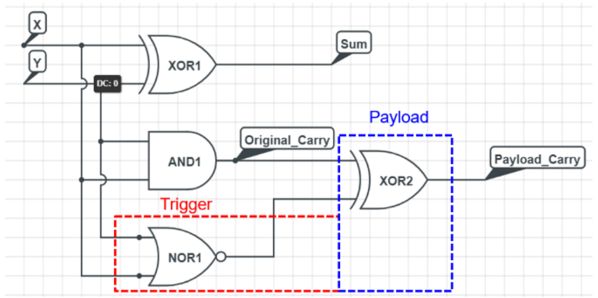
There are a few simple circuits, which are inserted with malicious modifications. These contain:

Combinational Trojan

1. **Half adder**

   Trigger & Payload: For example, half adder, we insert some HTs to interfere with the operation of the circuit. The chance of occurrence is 1/4, triggered by primary inputs.

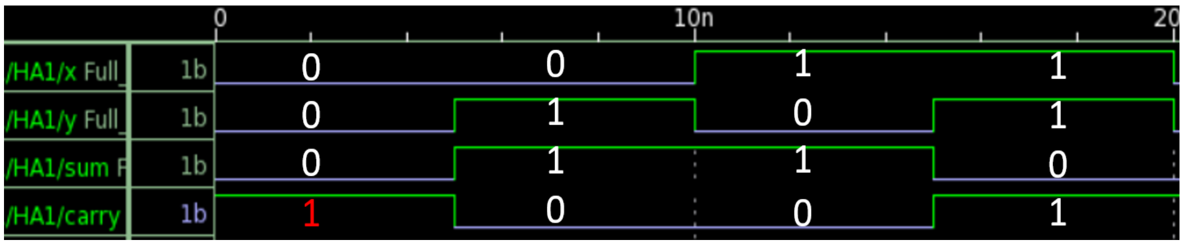




Normal Half Adder

Infected Trojan Half Adder

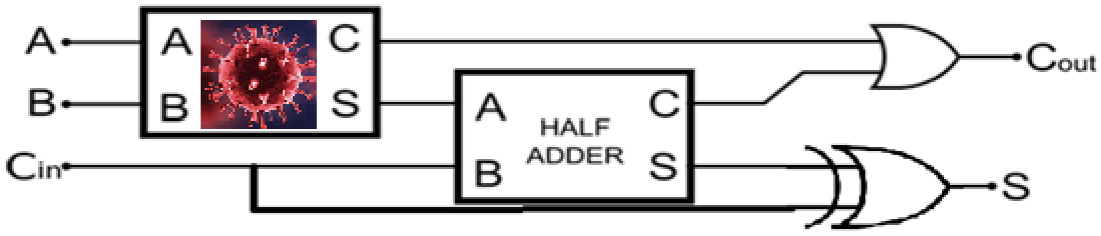
Detection: Our example will utilize the waveform of Verilog simulation.
We can observe that the carry is wrong when input x=0, y=0.



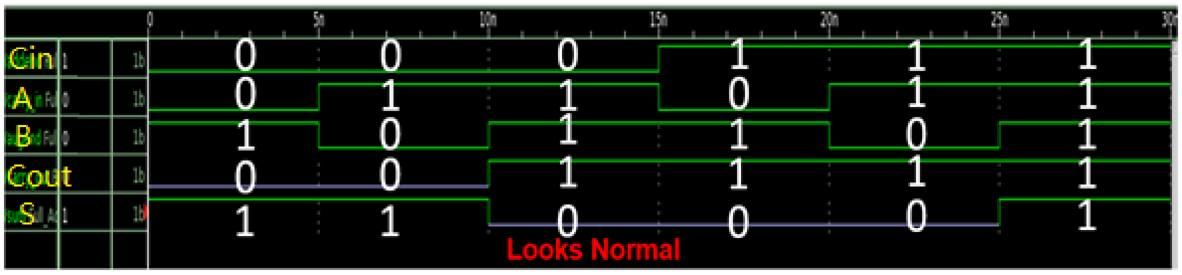
From this example, it can be observed that not all inputs can generate incorrect output results. Therefore, comprehensive or special method testing is required.

2. **Full adder**

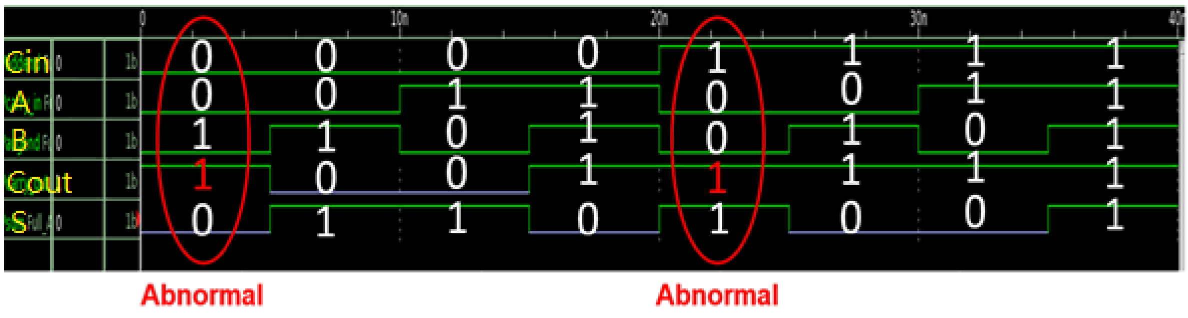
   Trigger & Payload: Classic Trojans also include hierarchical infections. For full adder, it was composed of two half adders so that if half adder is infected, it will also be affected. This example shows that Trojan can exist from the bottom to the top-level.



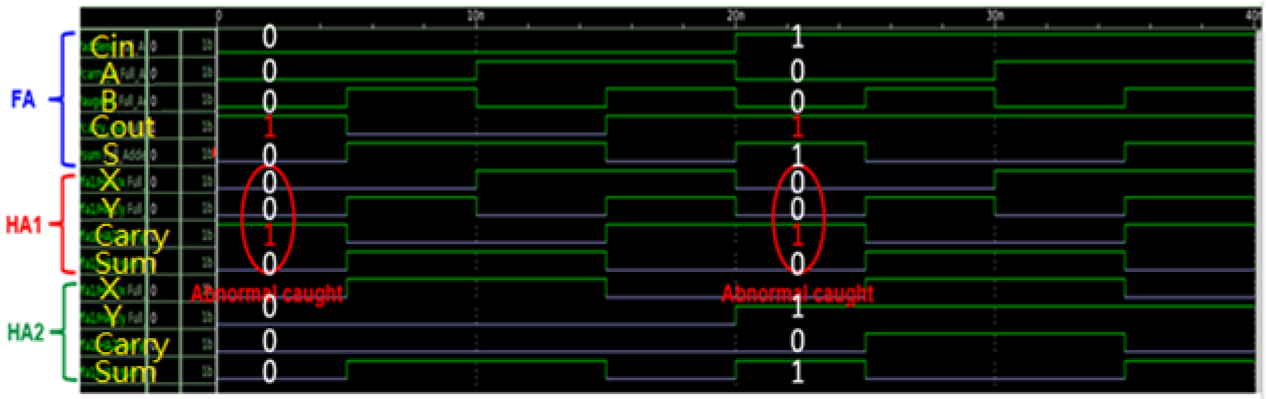
Detection: The traditional HT detection method needs to conduct a comprehensive detection of the input in order to catch the HT with a low probability of occurrence. The example given below, which shows that the possibility of finding HT may be missed if the detection is not comprehensive.



Although the picture above looks normal, the test project is not complete. Only when the conditions are full can we find the hidden abnormal.

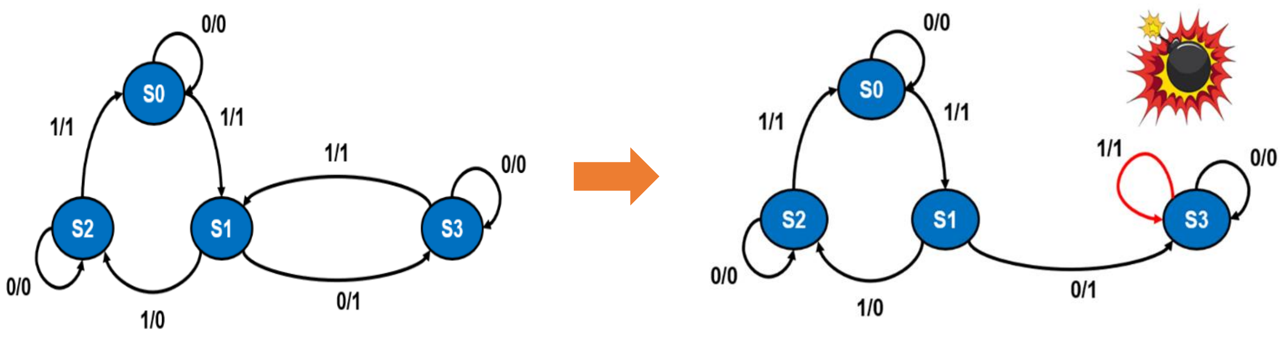


If the abnormal state cannot be observed from the top-level circuit, a hierarchical search can be used. It is easier to find the place where maybe a Trojan exists, and then check the RTL code to fix it after finding it.

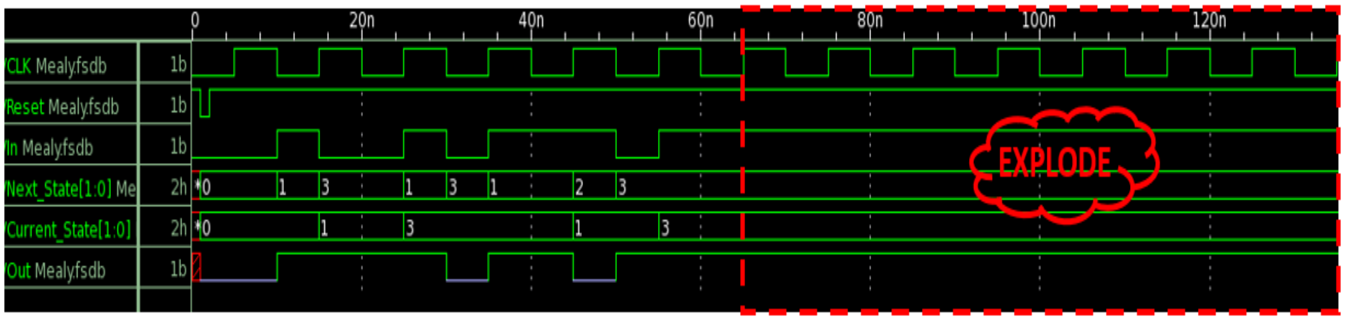


Sequential Trojan – Finite State Machine (Bomb)

Trigger & Payload: We implanted a Trojan into Finite State Machine to generate the error the state in certain time or condition. After the bomb (hidden counter) explodes, the state is stuck at S3 forever.

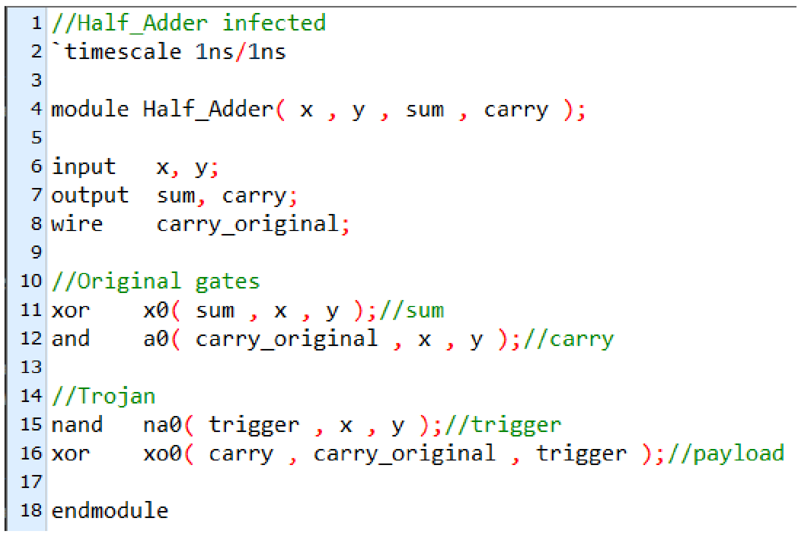


Detection: In general, it can take a long time for the bomb to trigger, so it must take a long time to detect it.



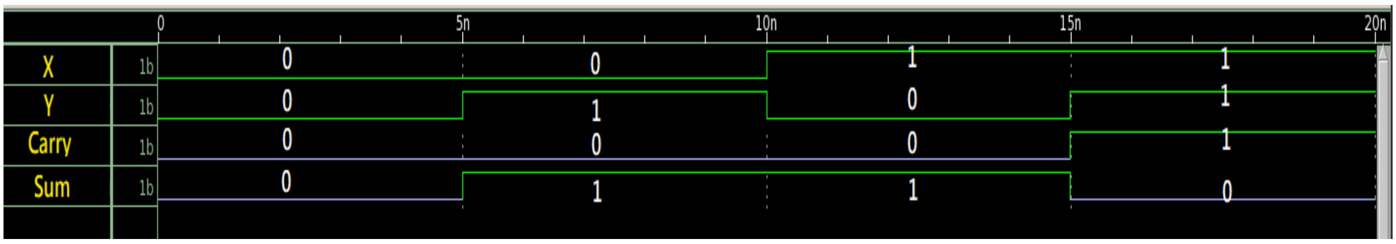
Input file format

The input file contains gate-level netlist (infected) Verilog code and correct behavior (waveform or state diagram).



```
//Half_Adder infected
`timescale 1ns/1ns

module Half_Adder( x , y , sum , carry );

input   x, y;
output  sum, carry;
wire    carry_original;

//Original gates
xor    x0( sum , x , y );//sum
and    a0( carry_original , x , y );//carry

//Trojan
nand   na0( trigger , x , y );//trigger
xor    xo0( carry , carry_original , trigger );//payload

endmodule
```

(a) Gate-Level Verilog (Infected Half Adder)



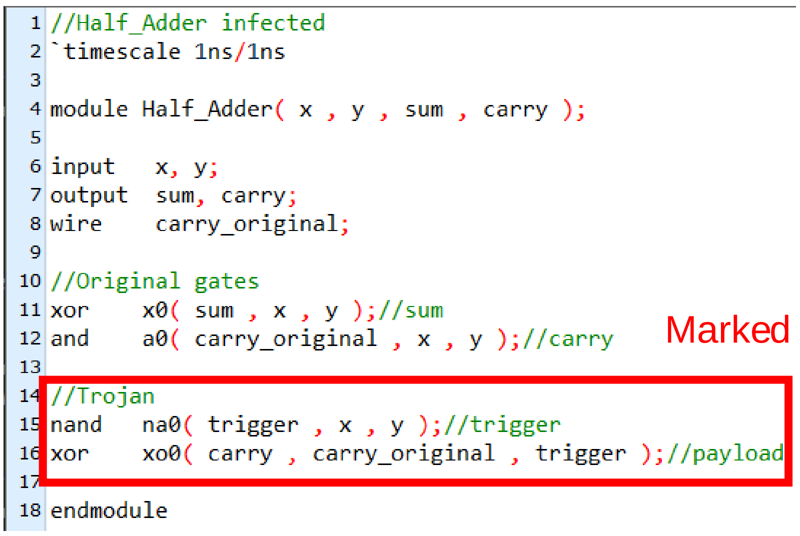
(b) Correct Behavior (Half Adder)

You have to design suitable test pattern through testbench and observe the response. After getting the response, you need to check the overall circuit to find the place where Trojan may be inserted.

Output file format

Required:

1. Report: You must specify how to detect the trojan and where the trojan exists (mark it).



```
//Half_Adder infected
`timescale 1ns/1ns

module Half_Adder( x , y , sum , carry );

input   x, y;
output  sum, carry;
wire    carry_original;

//Original gates
xor     x0( sum , x , y );//sum
and     a0( carry_original , x , y );//carry

//Trojan
nand    na0( trigger , x , y );//trigger
xor     xo0( carry , carry_original , trigger );//payload

endmodule
```

2.Testbench: You need to provide the Testbench named *StudID*_PA4_casen_tb.v (ex: 9862534_PA4_case1_tb.v).

Optional:

1.C/C++ Code: If you use automated way to detect the Trojan, you have to offer your programming code.

2.Golden Circuit: If you use the golden method, please provide the circuit created by yourself.

Requirement

1. You can use the naked eye to observe the waveform or use C/C++ to automatically detect Trojans.
2. The advanced method is to create a golden circuit (correct) and compare it with wire or output of the infected circuit.
3. If you use automated way, you need to provide your code.
4. We don't restrict the report format and length. In your report, you have to at least include:
   (1) You are asked to point out the place where Trojans exist. (You have to take a screenshot of the Verilog code)
   (2) You have to explain the methods you used to detect each circuit.
   (3) The hardness of this assignment and how you overcome it;
5. All files should be submitted through ee-class. You have to submit **a report** named *StudID_Name*_PA4_report.pdf (ex: 9862534_陳聿廣_PA4_report.pdf). Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable. **BE SURE to follow the naming rule mentioned above. Otherwise, your program will be not graded.**

Grading

The grading is as follows:

(1) Correct Trojan's location: 30%
(2) Testbench: 20%
(3) Detection method: 10%
(4) The report: 10%
(5) Demo session: 30%
(6) Bonus: C/C++ Code (5%)、Golden Circuit (5%)

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 24hrs delay: 20% off
- Within 48hrs delay: 40% off
- More than 48hrs: 0 point

Contact

For all questions about PA4, please send E-mail to TA 李宗穎 (lee.zongyi7326@gmail.com)

References

[1] Rajat Subhra Chakraborty, Seetharam Narasimhan and Swarup Bhunia, " Hardware Trojan: Threats and Emerging Solutions" 2009 IEEE International High Level Design Validation and Test Workshop

[2] Francis Wolff, Chris Papachristou, Swarup Bhunia, Rajat S. Chakraborty, " Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme" 2008, DATE

[3] Scott C. Smith; Jia Di, "Detecting Malicious Logic Through Structural Checking"

[4] Julien Francq , Florian Frick , "Introduction to Hardware Trojan Detection Methods"