IOWA STATE UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SENIOR DESIGN PROJECT

TEAM DEC 1515

FINAL DESIGN DOCUMENT

PROJECT TITLE: Trusted Electronic Systems

TEAM MEMBERS:

1. Fatema Aftab – Team leader
2. Colton Bachman – Team Webmaster
3. Xin Hu – Team Webmaster
4. Yitao (Tyler) Liu – Key concept holder
5. Qian Chang – Team communication
6. Chao Huang – Team communication

Group email: dec1515@iastate.edu

Client/ Company/ Organization: Dr. Randy Geiger and Dr. Degang Chen

Client Contacts/Advisors:

1. Dr. Randy Geiger

Advisors Email: rlgeiger@iastate.edu

2. Dr. Degang Chen

Advisors Email: djchen@iastate.edu

Website link:

http://dec1515.sd.ece.iastate.edu/index.html

Table of Contents

GLOSSARY

ALU Arithmetic Logic Unit

ASIC Application Specific Integrated Circuit

CAD Computer Aided Design

CMOS Complementary Metal-Oxide-Semiconductor

CPU Central Processing Unit

DoS Denial of Service

EDA Electronic Design Automation

EM Electron Migration

FPGA Field Programmable Gate Array

HDL Hardware Description Language

IC Integrated Circuit

IP Intellectual Property

IO Input/Output

JTAG Joint Test Action Group specified IC test interface

RF Radio Frequency

RTL Register Transfer Level

SoC System on Chip

TCB Trusted Computing Base

VHDL VHSIC Hardware Description Language

PREFACE

Our senior design project is about hardware Trojans. A hardware Trojan is a malicious, dangerous and an intentional attempt of a designer towards an electronic circuit design resulting in nasty consequences. It is basically a backdoor that can be inserted into a system which can benefit the bad guy in controlling the entire hardware.

Hardware Trojans are capable of defeating nearly all of the security mechanisms these days because almost every hardware system has millions of transistors installed in them. The Trojan might leak important information to a third party, modify how the system functions or it can also be a simple Denial of Service (DoS).

PURPOSE OF THIS DOCUMENT

In this document we demonstrate the design of the vending machine, the Trojan inside it, and the design for its detection.

USE OF THIS DOCUMENT

This document can help anyone who is interested in contributing towards the detection and prevention of hardware Trojans. It helps us understand the threats and also shows the design of how it is implemented. This document does not state that there are Trojans in almost every IC out there, most ICs are reliable and trustworthy. This is also the case with the commercial vending machines, most of them are very good and they do not cheat the customers. This is just an explanation of how Trojans can be affecting our everyday lives without even us realizing it. Therefore, it is important to pay attention towards this topic and find creative ways to detect hardware Trojans. In order for us to illustrate this concept it was important to make a Trojan ourselves first and then find a way to detect it.

SYSTEM DESIGN OVERVIEW

The system design is divided into three main components:

1. Vending machine design
2. Trojan Design
3. Trojan Detection design

DESIGN METHODS AND STANDARDS

VENDING MACHINE DESIGN

Before we can create a hardware Trojan and insert it into a system, we need a Trojan-free system at the very beginning. After discussions, we decided to create a virtual FPGA vending machine as our platform for the Trojan. Although we have a design about FPGA vending machine in CPRE 281, it is too simple and there is not enough room to insert a series of Trojans in it. Besides, we are going to create a new one with a certain level of complexity.

Figure 1 shows the basic block diagram of this system. We used a 50MHz clock to make sure calculations can be done as soon as possible. In addition, we decide to use 3 bits input to represent money input. The table of money input bits shows below. We also decided to make a drink machine with push buttons so we have our push buttons input in the system. In commercial vending machines the owner can change the price of product without reprograming the whole FPGA board. This means there is a memory in the system so that the owner can just change the value of the price of products. Meanwhile, commercial vending machines can print out the sales information when the owner needs it to make sure they get the right amount of money back.

Figures 2 and 3 shows the system of money input. We did a little research on the coin/bill operators and because they are all analog devices, the outputs of these devices are pulses and the pulse width. For example 0.01±5% seconds, is much longer than few clock cycles, which means, we need a counter to count out 0.01-5% seconds and then generate a digital signal after the counter reach the value that represent 0.01-5% seconds. Since we are using a 50MHz clock, there will be too many JK flip-flops if we need to use this clock. Our group designed another counter in Verilog in order to make the "clock speed" of money input counter to 10 kHz so that we can use only 10 JK flip-flops in the counter. And once the counter reaches certain value, in our example 896 which represents 8.96ms, there is only one digital signal in one input will be generating output based on the analog input pulse.

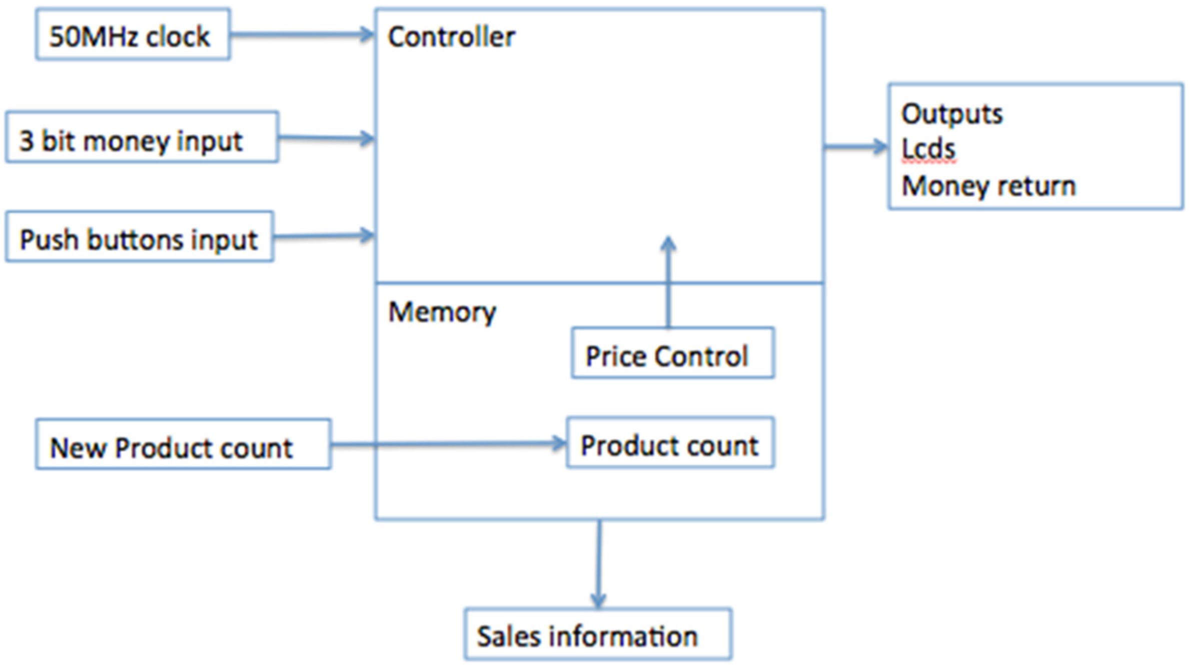


Figure 1: System block diagram

Signal	Money
001	5 cents
010	10 cents
011	25 cents
100	1 dollar
101	5 dollars

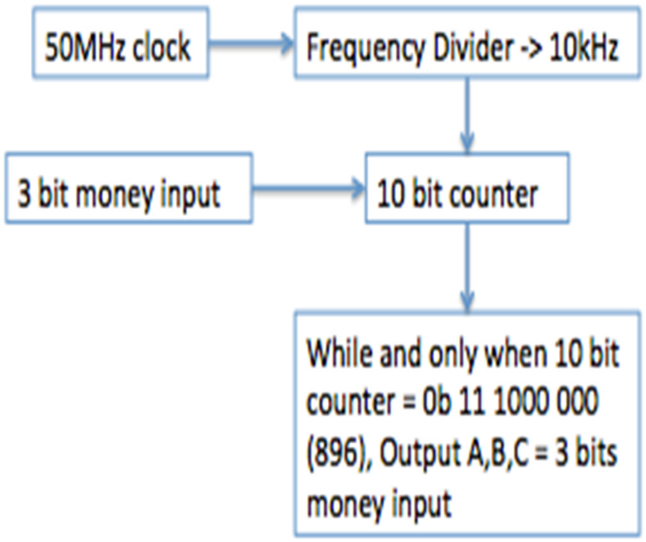

50MHz clock
Frequency Divider -> 10kHz
3 bit money input
10 bit counter
While and only when 10 bit counter = 0b 11 1000 000 (896), Output A,B,C = 3 bits money input

Figure 2: Counter Design

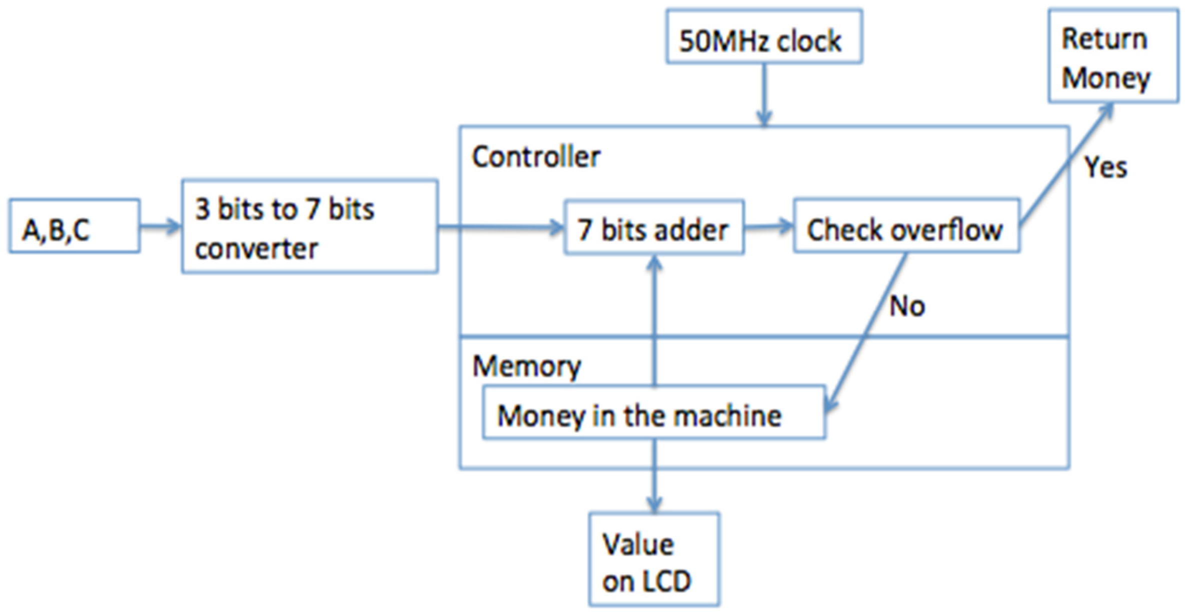


Figure 3: Money Input System

After we get the digital signal of money input, we convert the 3 bits input to 7 bits data. The first two bits represent the dollar, following by 4 bits data represent 10 cents, and finished by the last one bit of data represent 5 cents, the tables show the relation between 7-bit data and the money input and some example of the relation between money in the machine and 7-bit data shows below:

Money Input	3 bit	7 bits
5 cents	001	00 0000 1
10 cents	010	00 0001 0
25 cents	011	00 0010 1
1 dollar	100	01 0000 0

Table 2: Money input to 7-bit data

Money in the machine	7 bits
15 cents	00 0001 1
45 cents	00 0100 1
95 cents	00 1001 1
1.05 dollar	01 0000 1
1.5 dollar	01 0101 0

Table 3: Money in the machine to 7-bit data

After we get the 7-bit data, it goes into a 7-bit adder. However, there is a special section at the end of the adder to carry the number in cents to dollar, for example, change the data from 00 1010 0 to 01 0000 0. In addition, because most of vending machine have a maximum amount of money input, while the total amount of money in the machine is over 3.95, there will be an overflow in the adder. When this happens, the system will return the money back to the customers and would not change the 7-bit data.

When the customer is purchasing the product, after inserting the money, they need to push the button for the selection. Because most of the push button circuit has a de-bounce circuit in the push button so there would not be multiple readings in one push of the push button. Figure 4 shows how the purchasing system works.

SYSTEM ARCHITECTURE

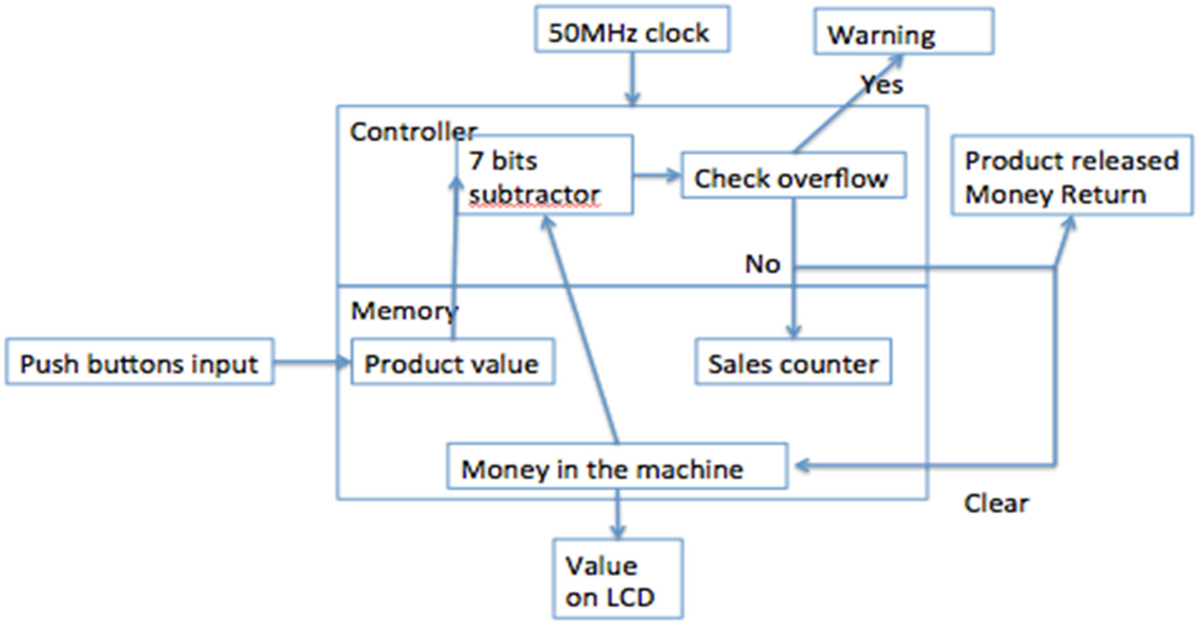


Figure 4: Purchasing System

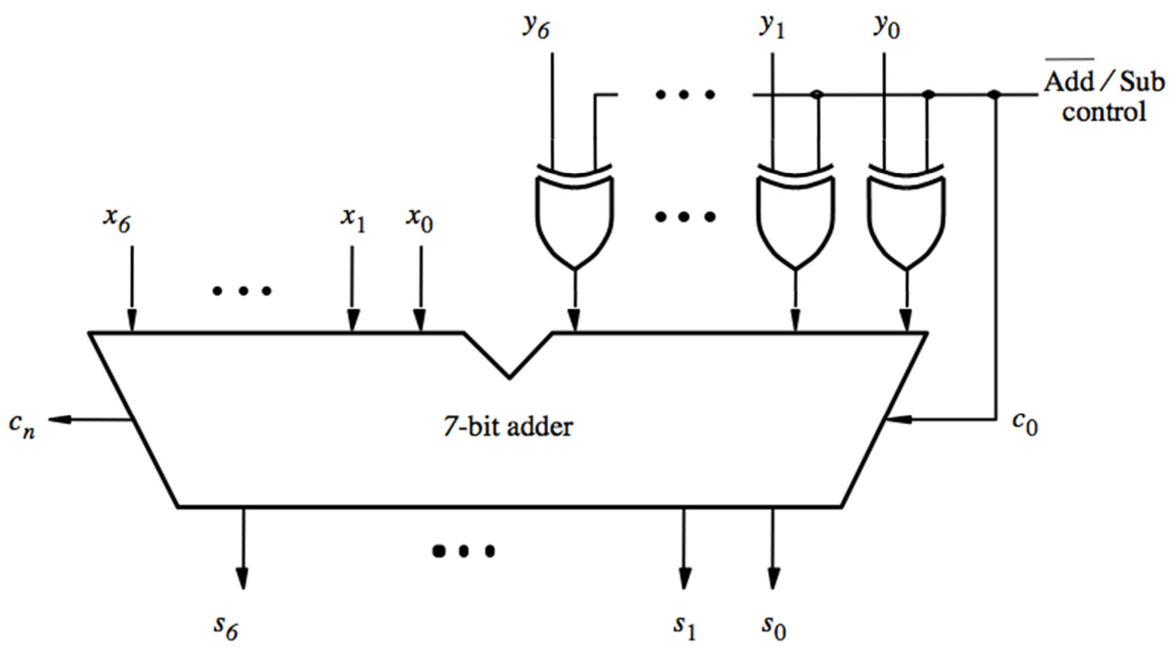


Figure 6: adder/subtractor design

When a customer attempts to purchase a product without having enough money, the current amount of money in the machine will not change. If there is not an overflow, the purchase is successful.

Figure 7 shows the basic idea about how to change the price of the product and how the system gives the owner the sales information. When the owner needs to change the value of the product, it will change the value in the memory. In the block diagram below, the sales counter is a 17-bit counter, with the maximum of record of 999.95 dollars. Once a product is sold, the sales counter will read the price of the product in 7-bit data formation and add the 7-bit value to the lowest 7 bits of the sales counter. Once the owner needs this information, the system will read the data from this counter.

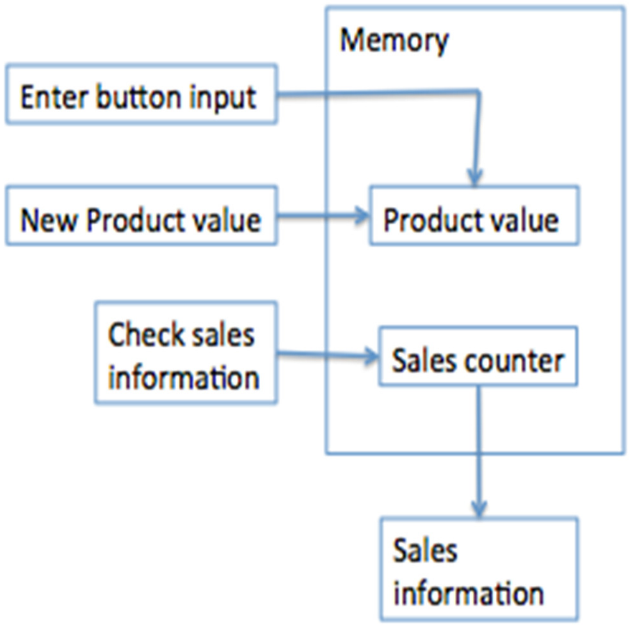


Figure 7: Memory system

In the Fall semester, we added more inputs into the vending machine. This made the vending machine a little bit more complex. Later in the semester we added junk code to the vending machine which gave more room for the Trojan design team to hide the Trojans. This junk code does not affect the system at all it does not do anything but just adds more code.

TROJAN DESIGN

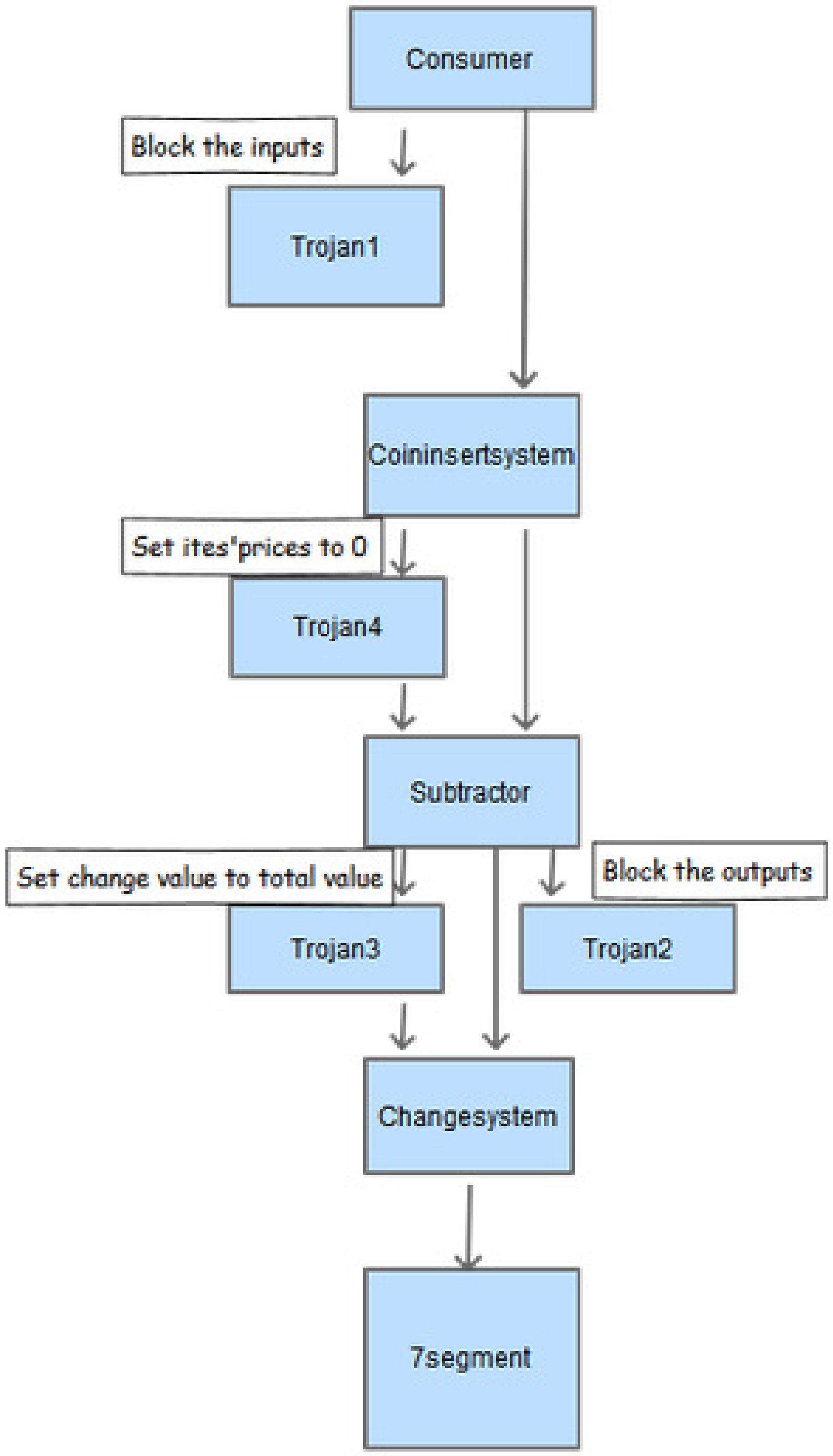
In our project we are doing two types of vending machine Trojan:

1. Trojans that help the owner of the vending machine and hurts the customer
2. Trojans that help the customer and harm the owner.

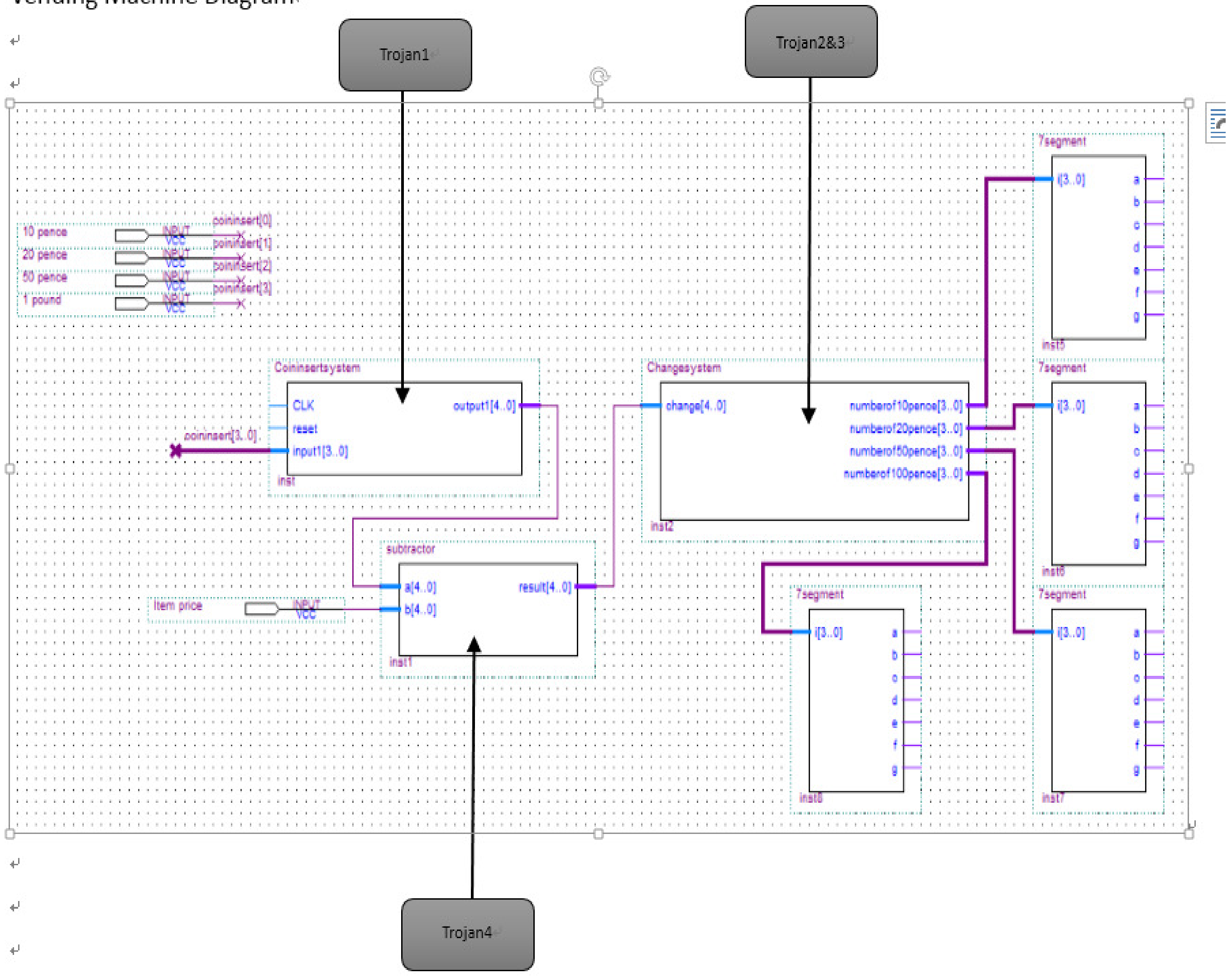
A Trojan that would help the owner would work by getting more money from a customer. The customer would put money into the machine, but the machine could randomly "eat" the customer's money. Or, the user could input the code for the item he or she wants, and then the vending machine would deduct money but not dispense the item. These Trojans are harder to detect, because many users would not call the owner and demand money back. The customers could be none the wiser while the owner is getting extra money without dispensing products.

Trojans that help the customers but hurt the owner involves the customer getting something out of the machine, usually for free. The users could insert a special code or sequence of coins and the machine could give out money or free products. This type of Trojan would be easier to detect because vending machines keep records of how many products have been sold and how much money is in the machine. If the owner sees that there is less money than there is supposed to be, then they will become suspicious and possibly replace the parts of the vending machine with a Trojan in it.

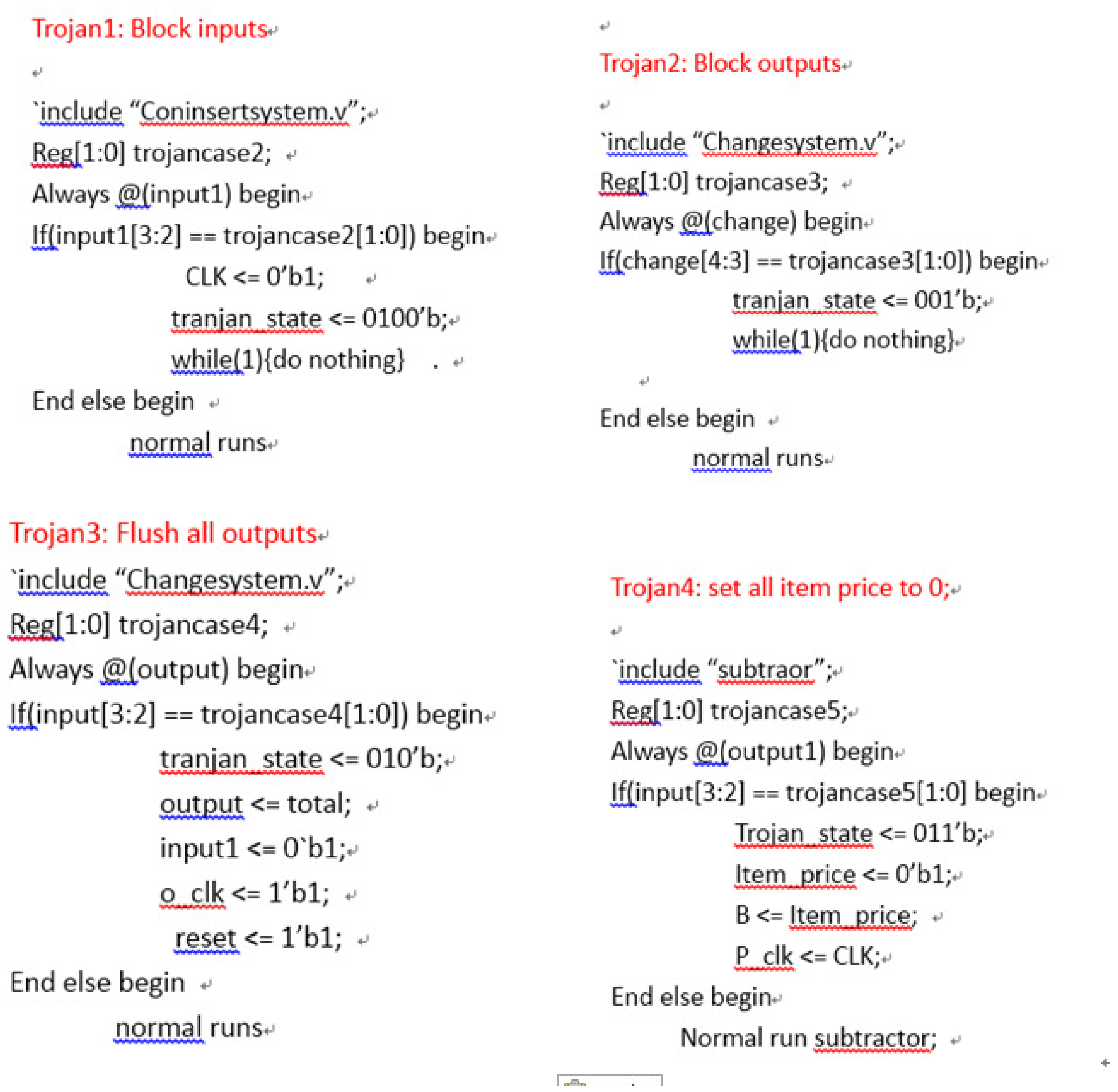
This is the diagram of vending machine with our Hardware Trojan:


Consumer
Block the inputs
Trojan1
Coininsertsystem
Set ites'prices to 0
Trojan4
Subtractor
Set change value to total value
Block the outputs
Trojan3
Trojan2
Changesystem
7segment

Vending Machine Diagram



There is the circuit diagram to explain where we installed our hardware Trojan. In the vending machine, there are three major blocks we can set our hardware Trojans.



Trojan1: Block inputs

```
`include "Coninsertsystem.v";
Reg[1:0] trojancase2;
Always @(input1) begin
If(input1[3:2] == trojancase2[1:0]) begin
            CLK <= 0'b1;
          tranjan_state <= 0100'b;
          while(1){do nothing}    .
End else begin
       normal runs
```

Trojan2: Block outputs

```
`include "Changesystem.v";
Reg[1:0] trojancase3;
Always @(change) begin
If(change[4:3] == trojancase3[1:0]) begin
          tranjan_state <= 001'b;
          while(1){do nothing}

End else begin
       normal runs
```

Trojan3: Flush all outputs

```
`include "Changesystem.v";
Reg[1:0] trojancase4;
Always @(output) begin
If(input[3:2] == trojancase4[1:0]) begin
          tranjan_state <= 010'b;
          output <= total;
          input1 <= 0`b1;
          o_clk <= 1'b1;
           reset <= 1'b1;
End else begin
       normal runs
```

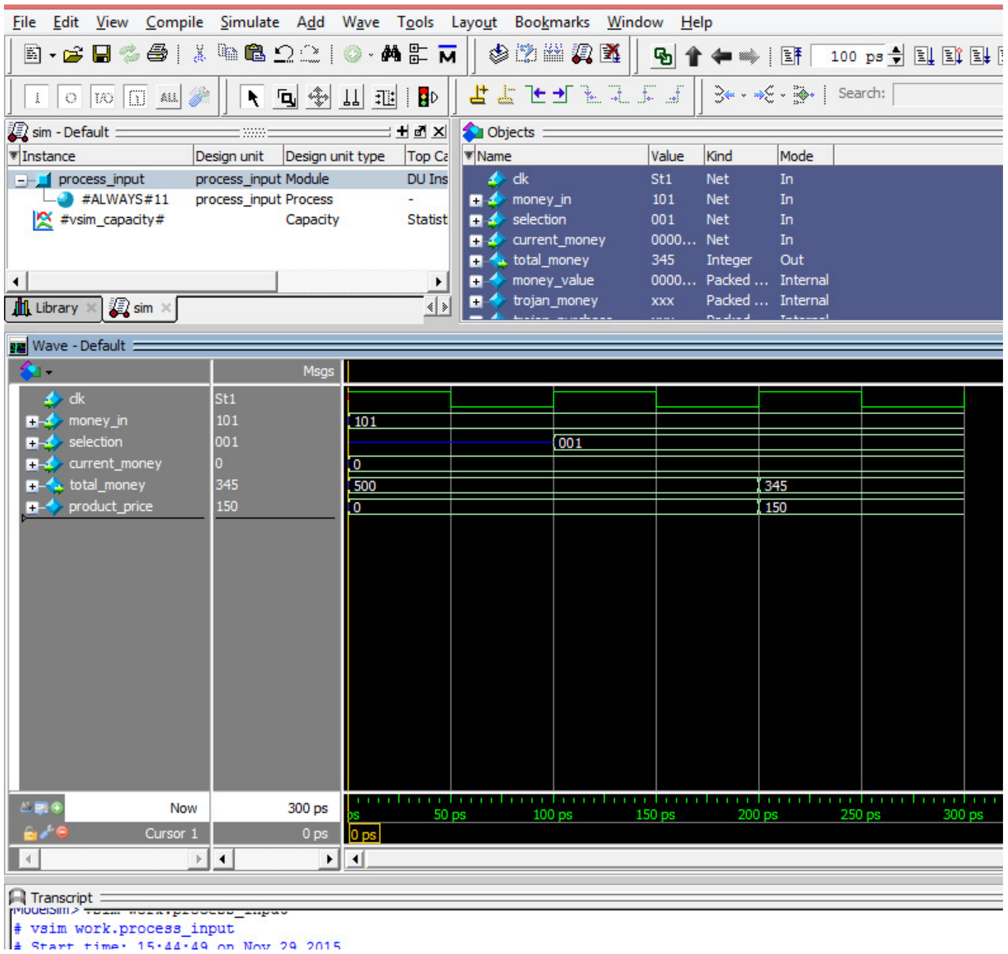
Trojan4: set all item price to 0;

```
`include "subtraor";
Reg[1:0] trojancase5;
Always @(output1) begin
If(input[3:2] == trojancase5[1:0] begin
          Trojan_state <= 011'b;
          Item_price <= 0'b1;
          B <= Item_price;
          P_clk <= CLK;
End else begin
     Normal run subtractor;
```

These are some hardware Trojans we have implemented into our vending machine. Trojan 1 is used for blocking all coins that customer insert. If this Trojan is activated, it will set the clock variable to 0 on the state block so the vending machine will not go to the next state, and there is another polling loop for keep Trojan 1 running. Trojan 2 is for blocking the output, so customers will not receive change or products. This has similar functionality as Trojan 1, the only difference is changing from an active statement to an output variable. Trojan 3 is works by flushing all coins out when activated. Finally, Trojan 4 sets all item prices to 0.

ALWAYS ON TROJAN DESIGN

In always on Trojan the Trojan does not need any external interference to activate it. This always on Trojan always gives $0.05 less than what it is supposed to. For instance, if the customer put $2.00 into the machine and bought an item for $1.25, the customer will receive $0.70



Instance	Design unit	Design unit type	Top Ca
process_input	process_input	Module	DU Ins
#ALWAYS#11	process_input	Process	-
#vsim_capacity#		Capacity	Statist

Name	Value	Kind	Mode
clk	St1	Net	In
money_in	101	Net	In
selection	001	Net	In
current_money	0000...	Net	In
total_money	345	Integer	Out
money_value	0000...	Packed ...	Internal
trojan_money	xxx	Packed ...	Internal

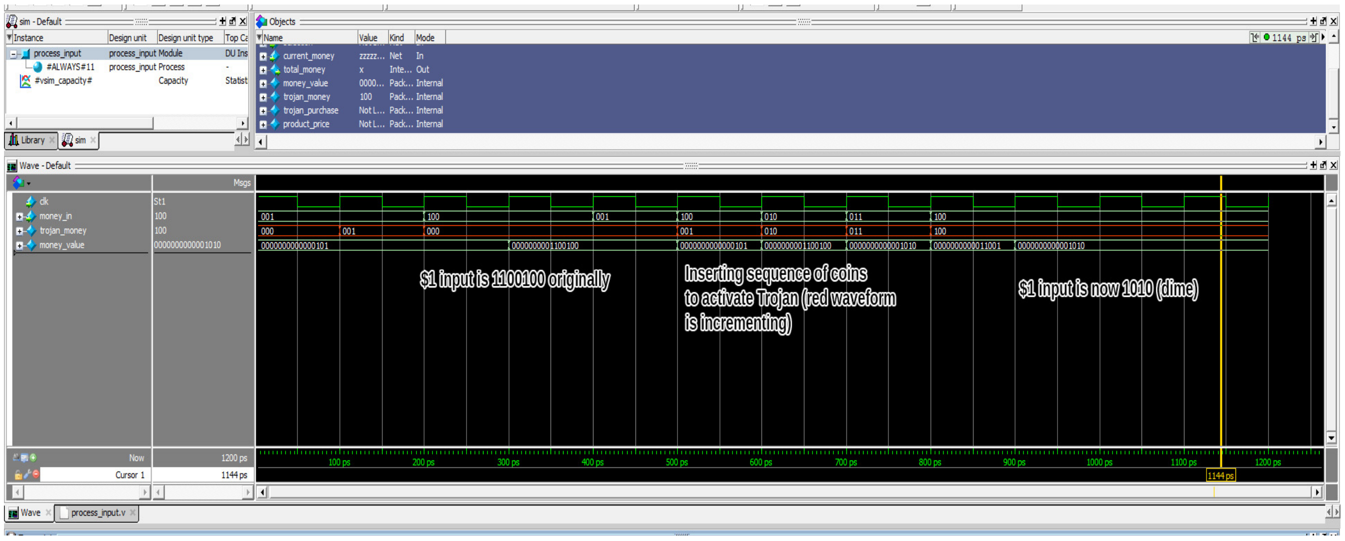
	Msgs
clk	St1
money_in	101
selection	001
current_money	0
total_money	345
product_price	150

```
Now 300 ps
Cursor 1 0 ps
ps 50 ps 100 ps 150 ps 200 ps 250 ps 300 ps
```

```
# vsim work.process_input
# Start time: 15:44:49 on Nov 29 2015
```

SEQUENTIAL TROJAN DESIGN

This sequential Trojan activates when a customer puts a particular sequence of coins; the sequence is one dollar, nickel, dime and a quarter. When the Trojan activates it changes the next dollar into a dime.


sim - Default
Objects
Wave - Default
clk
money_in
trojan_money
money_value
$1 input is 1100100 originally
Inserting sequence of coins to activate Trojan (red waveform is incrementing)
$1 input is now 1010 (dime)

TROJAN DESIGN ISSUES

These Trojans need to be small and do simple tasks; otherwise they will be easily detected by manufacturers. If a Trojan is too big and has too many transistors, it could use too much power or it could be easy to identify on a circuit. We want to hide the Trojan within the circuit so it is undetectable with current detection methods, and the best way to do this is to design a Trojan that is as small as possible. Or our Trojan could be hidden in different parts of a circuit to make detecting the Trojan quite complicated.

3. TROJAN DETECTION METHODS

I. Using current integration and localized current analysis method

We need two FPGA boards. One is golden chip without Trojan, the other one is the chip with Trojan. For each board, we need to choose enough power port, which can cover the whole board, to measure each port's current. Then, we will input some patterns into each board. A large number of transitions is generated when apply these patterns to chips. By measuring the local current through each power ports, we can integrate the measurement current and compare the test chip with the golden chip. Because the amount of current drawn by the Trojan can make the integrated current on each port significant with the Trojan free chip.

II. Using path delay fingerprint method

Trojans will insert extra delay in some passing signals. Therefore, we collect the path delay to generate the fingerprint of the nominal (golden) chip. The test pattern should cover as many parts of the whole chip as possible. Moreover, we will input these same patterns into the Trojan inserted chip to collect its path delay information. Compare the Trojan inserted chip output delay information with the fingerprint, then we can get a convex hull set of points. Chips whose test points are in or near the convex hull in the whole fingerprint spaces will be considered golden chip. Otherwise, it will be considered as Trojan inserted chips.

IMPROVE DETECTION PROBABILITY

Imagine we already have a method to detect the Trojan, if we want more accurate result then we need to come up with some ideas to improve our method.

Here are the steps for this method that we can use to improve detection method:

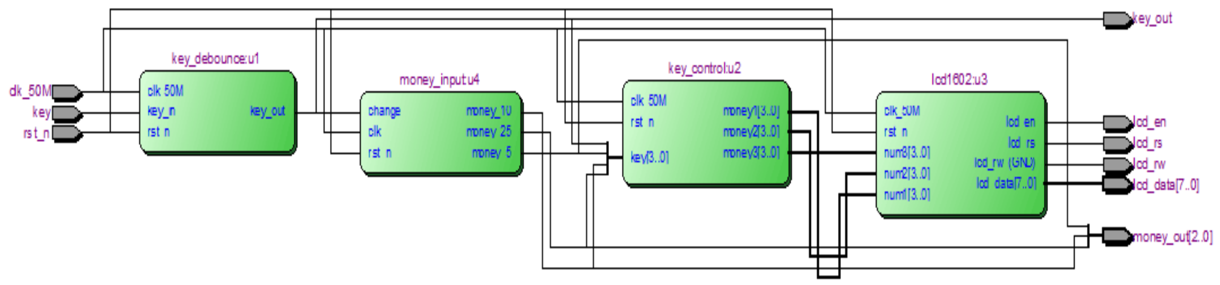
We find out the weakness node, and set up a special mode to control the certain node that are important. Our Trojan runs in that condition, we can call that safe mode. We only control the main node at that time in each single module; we just run them, and record current, power, frequency. Then we run vending machine in normal mode, and record those data again, we compare those with "safe mode", we can know if there is a real Trojan in our vending machine.

Then we made an assumption, if we push the button hard, to achieve special pulse in vending machine, that is a trigger for Trojans. The way to detect this Trojan is try to record and monitor the pulse and frequency change. We can connect the circuits with oscilloscope and push the button normally, then we keep monitor the pulse change, try other kind of pulse input, then we can measure the current or voltage change, if there is a change under the special pulse, that means there is a Trojan.

If the Trojan based on an algorithm, then we need to double check the code, and we make an algorithm checking code to check the algorithm in all possible ways. That may need someone who is the expert of coding to finish this code.

These two methods are just a rough plan, we still need to get the blueprint for Trojan, and then we can start to find out the way to detect Trojans.

TROJAN DETECTION SYSTEM



Key Debounce:

To avoid multiple signals being sent when you press the key or button

Money Input:

5 inputs for money. Money from 5 cents to 5 dollars.

Key Control:

Keeps track of the current amount of money in the machine.

LCD Display:

Display the total money by LCD

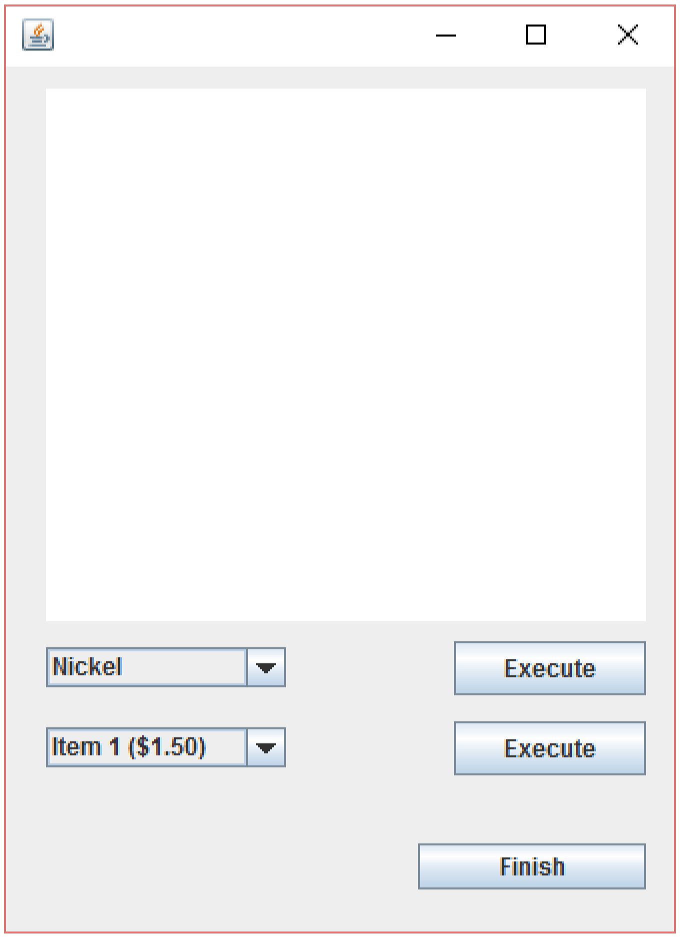
CONCLUSION

Hardware Trojans are a continuing threat to our society. They compromise the security and functionality of our electronic systems. The threats can be as dangerous as a backdoor or as mild as a Denial of Service. They are easy to activate and hard to catch. So it is important for us to think of something which would stop this from spreading and come up with some counter measures to avoid them.

APPENDIX I: Operation Manual

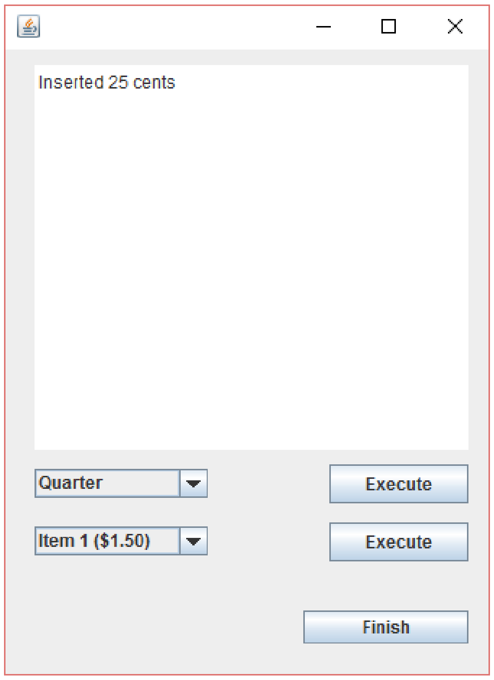
Instructions to set up the system

First connect the FPGA from the RS232 port and then program the FPGA using Quartus. Next, open the GUI, shown below:

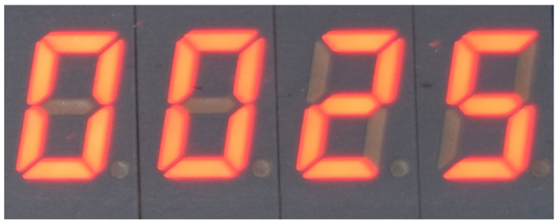


The upper ComboBox contains the different denominations of money to insert (nickel, dime, quarter, one dollar, and five dollars), while the lower ComboBox contains the purchasable items (Items 1-3, each with a different price). Selecting an item in the ComboBox and clicking on the appropriate "Execute" button will send the command to the FPGA. The money added and purchases are logged in the white box in the middle. Current money in the machine is shown on the FPGA's 7-segment display.

Next we will add money and purchase an item. First add 25 cents:

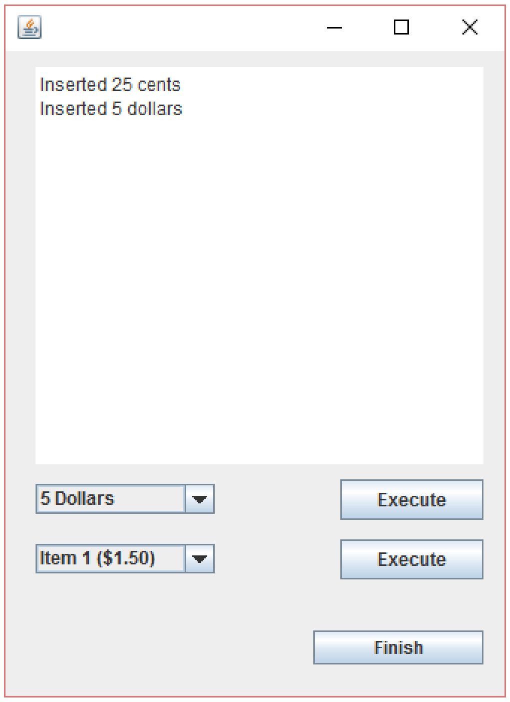


```
Inserted 25 cents
```

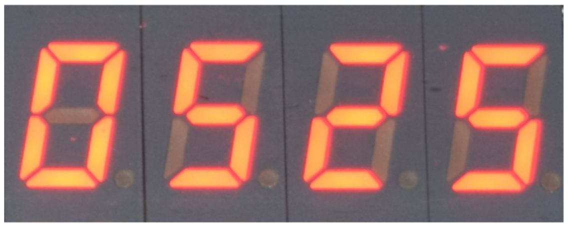


After adding 25 cents, the LCD indicates that I have 0025, which translates to $00.25.

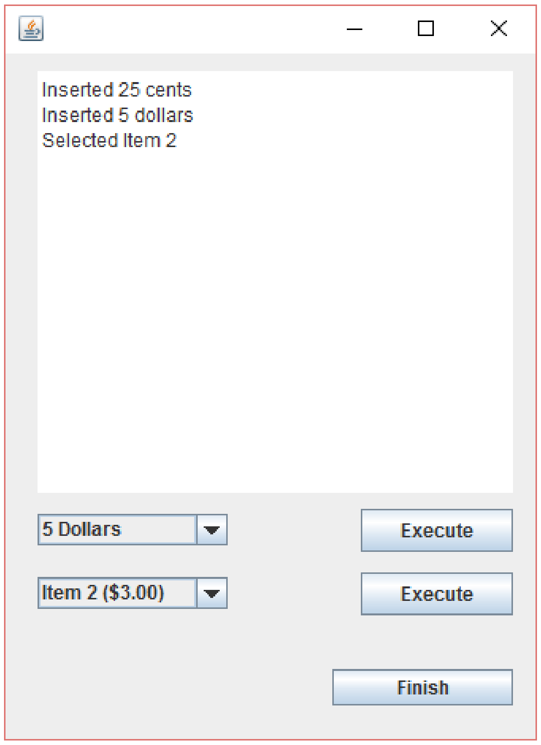
Next we add 5 dollars so we have enough to purchase item 2 ($3.00).



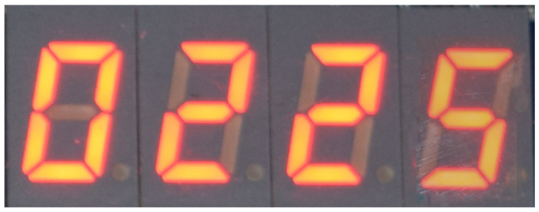
```
Inserted 25 cents
Inserted 5 dollars
```



Finally item 2 will be purchased:



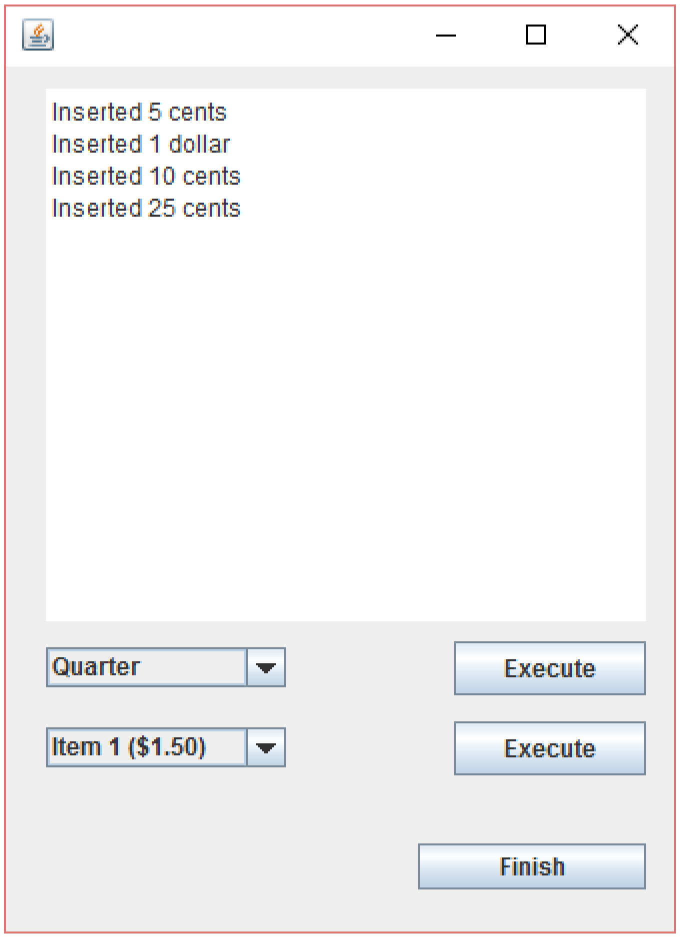
```
Inserted 25 cents
Inserted 5 dollars
Selected Item 2
```



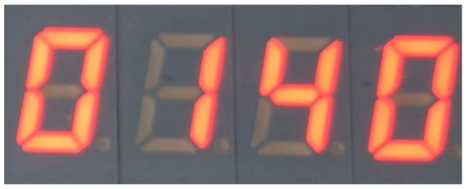
The LCD displays there is 0225, or $2.25 left in the machine, as intended.

Now a Trojan will be demonstrated. This Trojan requires the user to input the sequence “nickel, 1 dollar, dime, quarter.” After the Trojan is active, if the next amount of money input is 1 dollar, then the user will only receive $0.10.

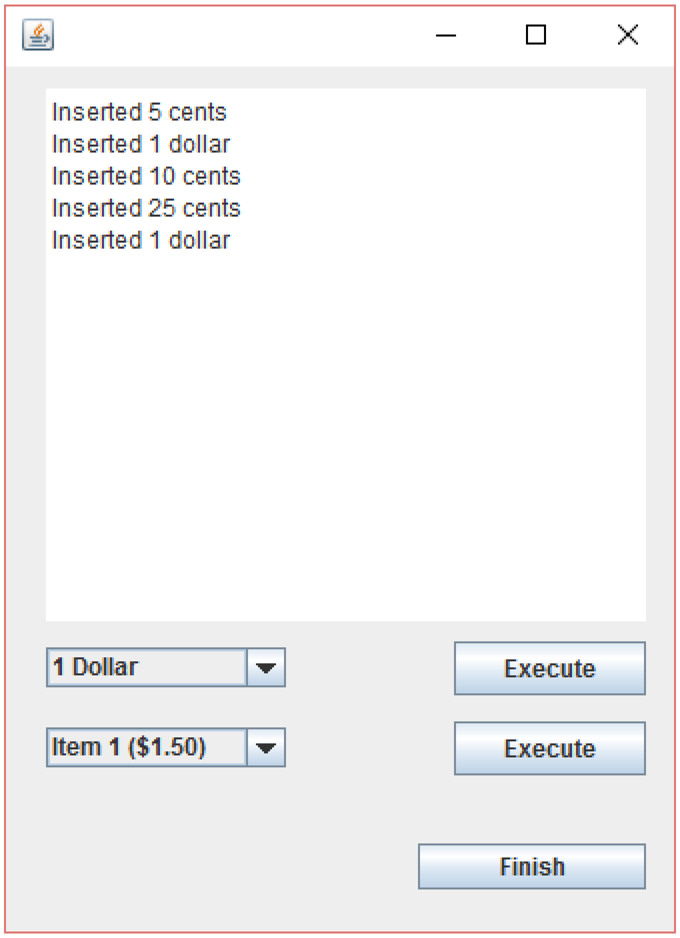
Here’s a picture of after the Trojan is active, but before inserting another dollar:



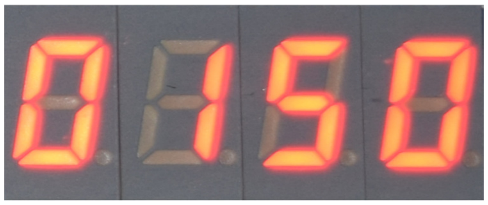
```
Inserted 5 cents
Inserted 1 dollar
Inserted 10 cents
Inserted 25 cents
```



And here's a picture after inserting 1 dollar. The amount should be $2.40, but is actually $1.50.



```
Inserted 5 cents
Inserted 1 dollar
Inserted 10 cents
Inserted 25 cents
Inserted 1 dollar
```



APPENDIX 2

IOWA STATE UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SENIOR DESIGN PROJECT
TEAM DEC 1515

UPDATED PROJECT PLAN

Project title: Trusted Electronic Systems

Abstract

Our senior design project is about hardware Trojans. A hardware Trojan is a malicious, dangerous and an intentional attempt of a designer towards an electronic circuit design resulting in nasty consequences. It is basically a backdoor that can be inserted into a system which can benefit the bad guy in controlling the entire hardware.

Hardware Trojans are capable of defeating nearly all of the security mechanisms these days because almost every hardware system has millions of transistors installed in them. The Trojan might leak important information to a third party, modify how the system functions, or it can also be a simple Denial of Service (DoS).

Executive Summary

Scientists have been working on hardware Trojans since last few decades because our world is overwhelmed with technology. Everything we use these days has control over us whether it is just an automation device or a computer. The ability for a common man to trust all these devices is becoming a concern. The security of our devices is making the integrated Circuits (IC) main topic of research for a lot of engineers. This is because ICs are used in procuring and maintain cyber, military, financial, and other critical systems. Therefore, this project mostly focuses on the hardware security, most importantly on identifying methods that can be used to introduce the Trojans into the electronic systems. Moreover, this project also emphasizes on methods for detecting these counterfeited electronic components and devices. We will also study the Trojans that reside in the software which helps trigger the Trojan in the hardware. More and more industries are getting aware of this vulnerability and millions of dollars are being invested to find the way out this problem.

Hardware Trojan insertion into an IC is very easy and can occur at any point of design. We know that there are different steps in making of an IC namely, specification, design, verification and manufacturing. A hardware Trojan can be built in any time and any of these steps. Maintaining a tight eye onto all these processes is very time and money consuming. These days most industries are separating all these different processes and increasing the use of Electronic Design Automation (EDA).

Hardware Trojans are very difficult to spot. They are hard to observe during the primitive IC verification and testing phases. It's tough to find the payloads and trigger states. Most ASIC designs are too huge to do exhaustive testing and verification on them so it is very likely that a lot of Trojans go undetected every day.

For making next generation defensive mechanisms for the development of electronic circuits we need to understand hardware Trojans. Even after a bunch of research, engineers still do not have a system that can ensure the device safety. This makes our project challenging yet very interesting for us because in near future it will be necessary for hardware systems to perform securely in presence of Trojans.

Introduction

In our project we want to introduce a hardware Trojan in a vending machine design. We are going to use a commercial vending machine which works on FPGA and introduce a hardware Trojan. We chose a vending machine design because it is a very applicable and a practical example of a hardware Trojan. This gives us capability of making Trojan as bad as possible and also come up with its detection and prevention methods. The reason we want a vending machine on FPGA is because we all are familiar with the operation of FPGA board after taking CPR E 281 (Digital Logic). The program is Verilog which uses Hardware Descriptive Language (HDL) and this is a programming language we all know.

Process Details

With this project we had to keep in mind three most important aspects of hardware Trojan:
Creating a hardware Trojan
Detection of Trojan
Steps to prevent the Trojan
For all these parts to come together and completing the project we had to understand the classification and trigger mechanisms of the Trojans.

Threats

Hardware Trojans are very obstinate once they are installed in a system they are always there. They are active when they system is turned on. They are able to change any and almost all electronic systems by infecting and changing the behavior of the integrated circuit. Considering the threats, understanding the insertion and activation mechanisms becomes very important. So therefore we researched classification of Trojans and how it affects the system. For our vending machine design we are thinking about making a sequential, always on and externally triggered Trojan.

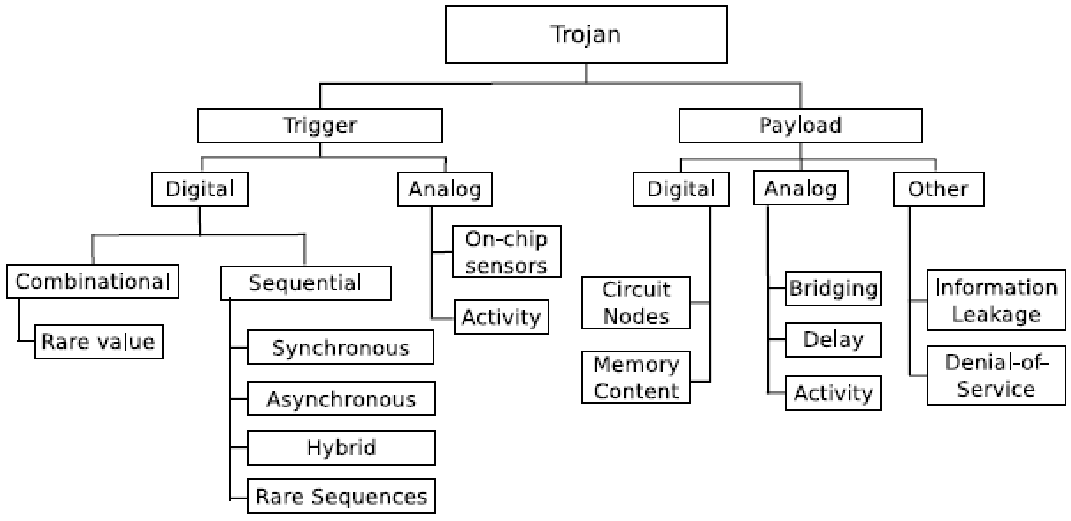


Figure 1: Hardware Trojan Taxonomy: Chakraborty, Narasimhan and Bhunia (2010)

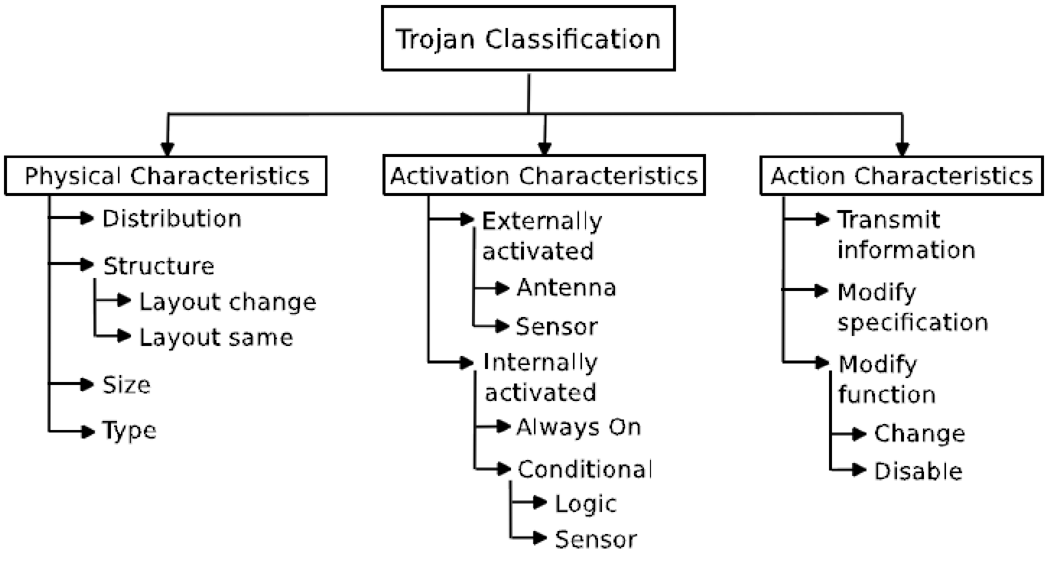


Figure 2: Hardware Trojan Taxonomy: Wang, Tehranipoor and Plusquellic (2008)

Trigger Mechanisms

A trigger mechanism means the activation of a Trojan once it is inserted in to the software/hardware. There are several types of trigger mechanisms:

- Internally Triggered
- Combinational Activation
- Sequential Activation
- Externally Triggered
- Always on

Sequential Activation

This Trojan relies on the sequence of events occurring for activation. Mostly sequential activation helps with turning on a certain number of clock cycles.

Externally Triggered

Externally triggered Trojans depend on some interaction of the outside environment with the Trojan. It can be any sort of human interaction with the IC. The target is immediately activated if a button or switch is pressed. A trigger can also come from another component, such as a light sensor, which is connected to the target.

Always on

This is a Trojan which is always on and does not need any activation from any other source. This is never turned off which means that this Trojan is always working once it is inserted in the system. This Trojan is mostly used in leaking data through a circuit which has a side channel inside an IC.

Combinatorial Activation

The Trojan is activated by using a combination of buttons or switches. For example, if the user held down the first, third, and sixth buttons on the vending machine.

Test Plan

Since we have decided to do always on trigger mechanism our Trojan is going to be always on in the vending machine schematic. That means that there is no outside interference needed for the Trojan to get activated. There will be two kinds of Trojans in the vending machine. One Trojan will trick the customers into putting extra quarters/ dimes more than what the product costs. Another Trojan will be activated which will help us take the money out of the vending machine. The first Trojan could be classified into "always on" and the other Trojan will be "externally triggered" Trojan.

Once these two Trojans are inserted into the design we want to make the Trojan more insidious by having the Trojan trick only random customers so it is less obvious to the owner that something is wrong with the machine. We have to trick the owner into believing that the machine can sometimes do Denial of Service by taking extra money or not returning change. The externally triggered Trojan will give us control of how much money we want to take out of the vending machine. It will be activated by entering a sequence of coins which is specifically entered in the machine. For example, putting 5 pennies in a row will activate the sequential Trojan and gives us access to the money in the vending machine.

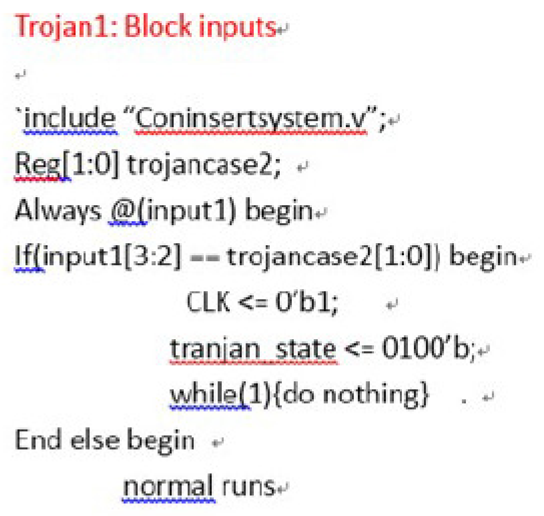
Our Trojan will be activated using sequential and combinatorial logic. For example, the Trojan is activated if you insert 2 dimes, 1 quarter, and 5 nickels while holding the coin return button.

Updates for Fall 2015:

1. We will install two completely functional Trojans in the vending machine. The Trojan detection team is not allowed to ask the Trojan design team about the details of the design. This gives us a real life like situation in which we do not know where exactly the virus is located.
2. We want to make a fully functional vending machine which can detect the Trojan inserted.
3. Expanding the vending machine by adding junk code and more inputs in order to give more room to Trojan design team for hiding Trojans.

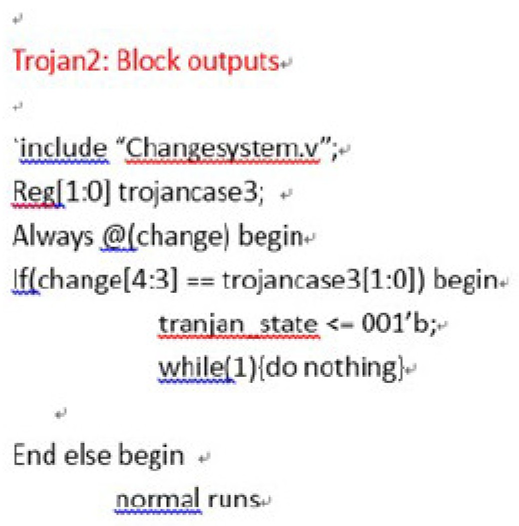
Trojan Design

We have created four different types of Trojans:
Block inputs
Block outputs
Flush all outputs
Set all items price to zero



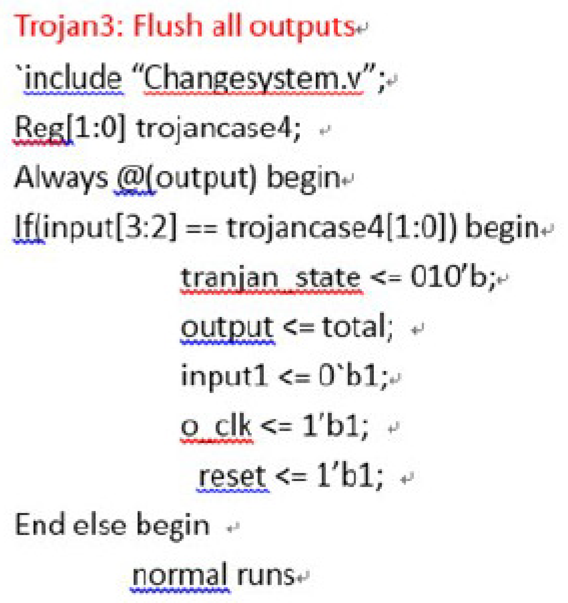
Trojan1: Block inputs

```
`include "Coninsertsystem.v";
Reg[1:0] trojancase2;
Always @(input1) begin
If(input1[3:2] == trojancase2[1:0]) begin
                CLK <= 0'b1;
              tranjan_state <= 0100'b;
              while(1){do nothing}  .
End else begin
          normal runs
```



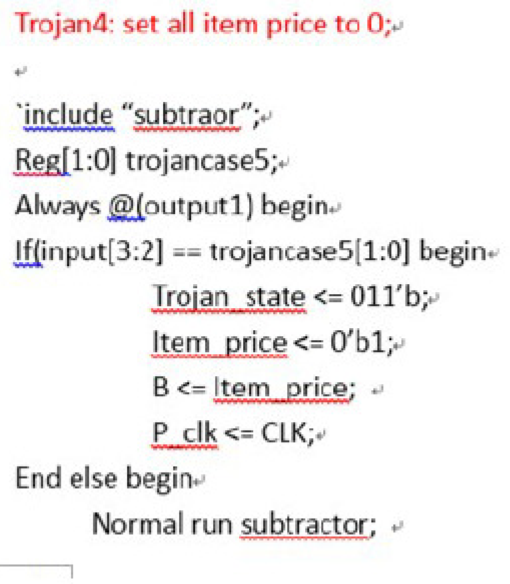
Trojan2: Block outputs

```
`include "Changesystem.v";
Reg[1:0] trojancase3;
Always @(change) begin
If(change[4:3] == trojancase3[1:0]) begin
             tranjan_state <= 001'b;
             while(1){do nothing}

End else begin
          normal runs
```



Trojan3: Flush all outputs

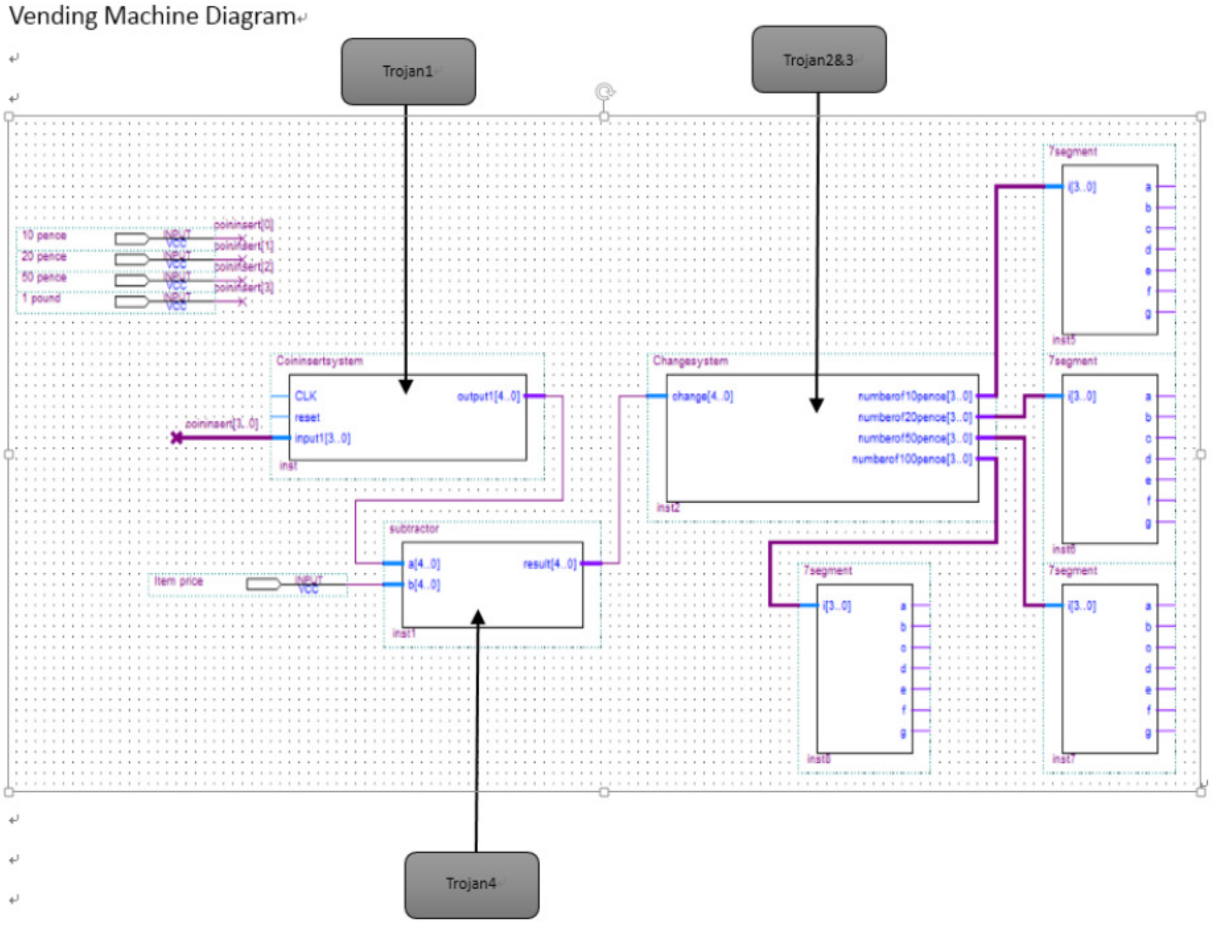
```
`include "Changesystem.v";
Reg[1:0] trojancase4;
Always @(output) begin
If(input[3:2] == trojancase4[1:0]) begin
             tranjan_state <= 010'b;
             output <= total;
             input1 <= 0`b1;
             o_clk <= 1'b1;
              reset <= 1'b1;
End else begin
          normal runs
```

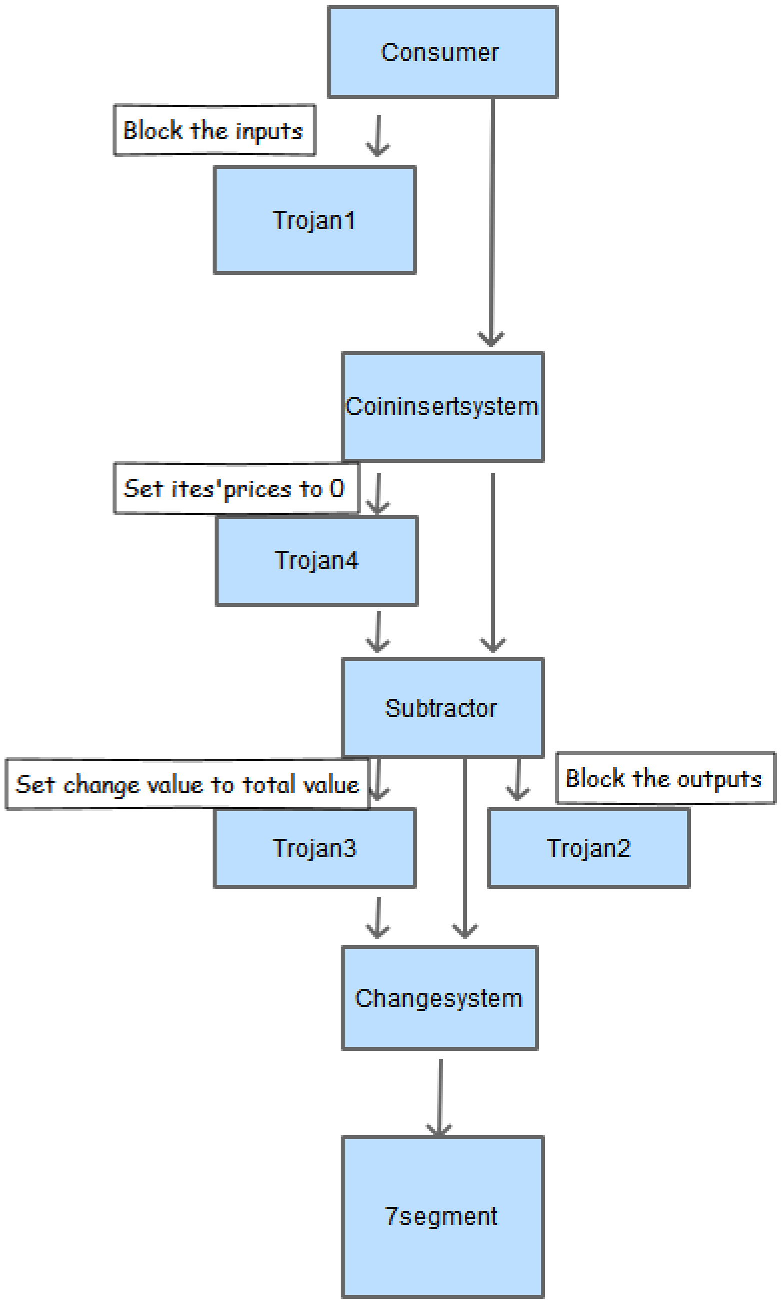


Trojan4: set all item price to 0;

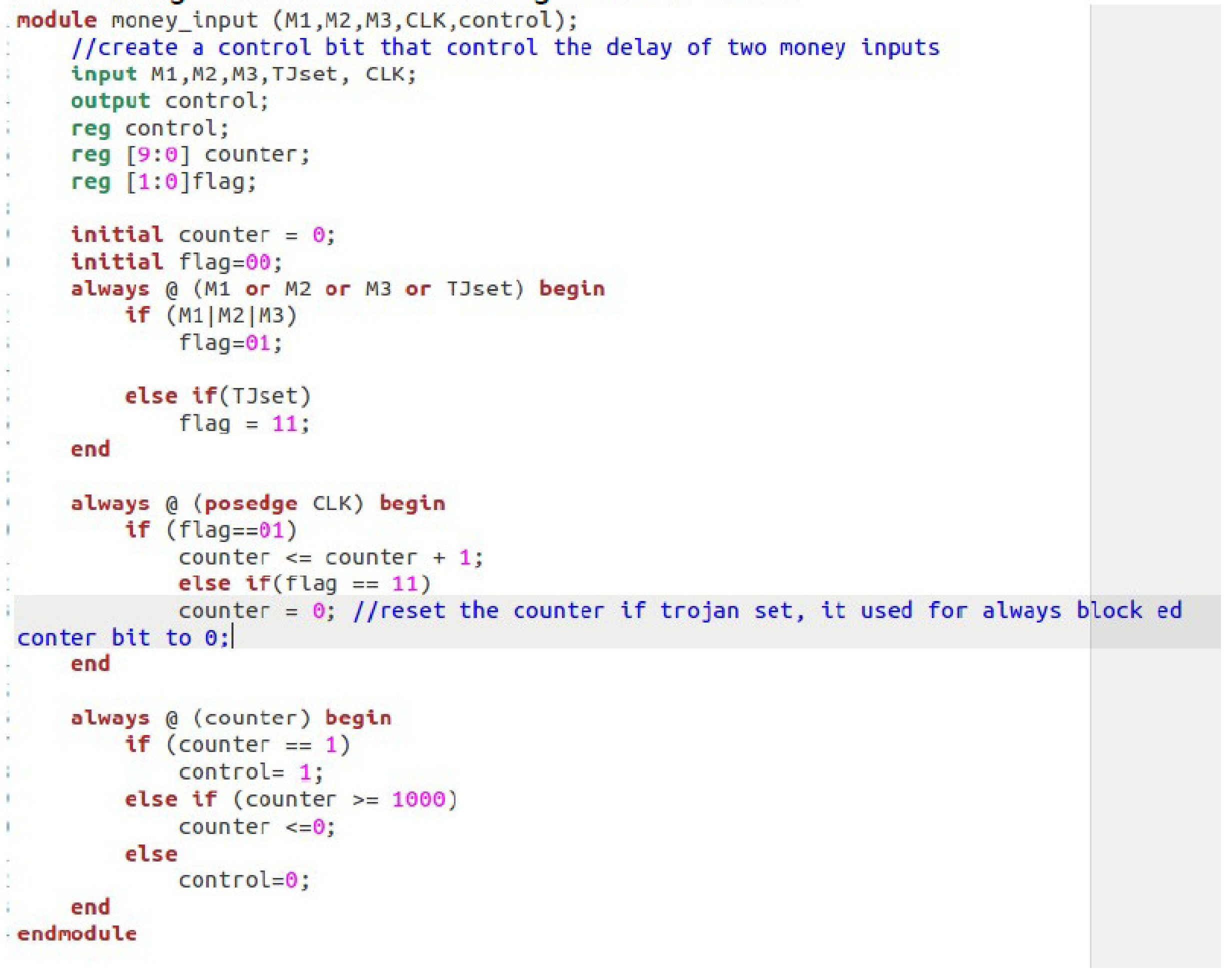
```
`include "subtraor";
Reg[1:0] trojancase5;
Always @(output1) begin
If(input[3:2] == trojancase5[1:0] begin
             Trojan_state <= 011'b;
             Item_price <= 0'b1;
             B <= Item_price;
             P_clk <= CLK;
End else begin
       Normal run subtractor;
```

Vending Machine Diagram

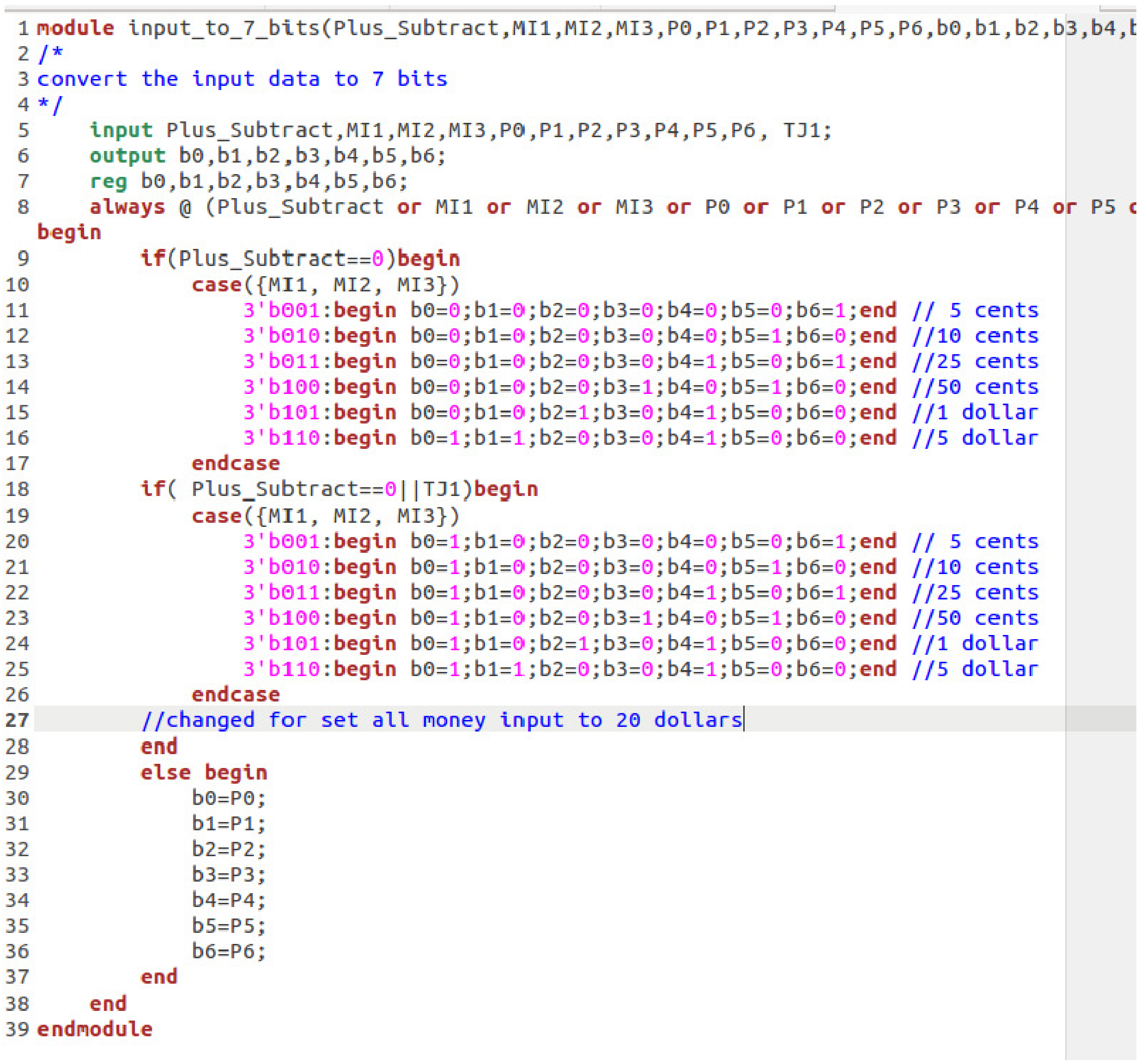



Consumer
Block the inputs
Trojan1
Coininsertsystem
Set ites'prices to 0
Trojan4
Subtractor
Set change value to total value
Block the outputs
Trojan3
Trojan2
Changesystem
7segment

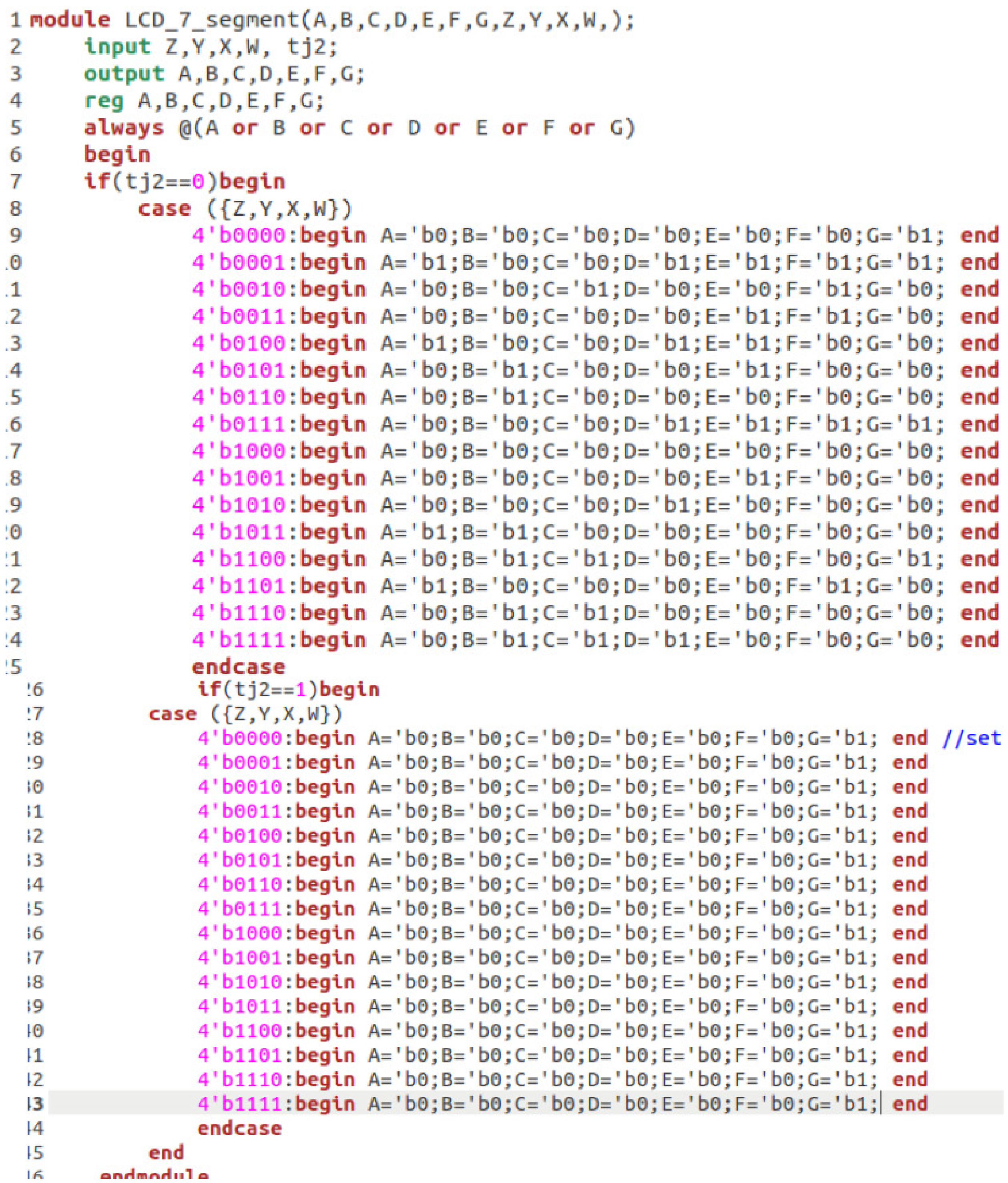
Some design drafts for our Vending machine codes:



```
module money_input (M1,M2,M3,CLK,control);
    //create a control bit that control the delay of two money inputs
    input M1,M2,M3,TJset, CLK;
    output control;
    reg control;
    reg [9:0] counter;
    reg [1:0]flag;

    initial counter = 0;
    initial flag=00;
    always @ (M1 or M2 or M3 or TJset) begin
        if (M1|M2|M3)
            flag=01;

        else if(TJset)
            flag = 11;
    end

    always @ (posedge CLK) begin
        if (flag==01)
            counter <= counter + 1;
            else if(flag == 11)
            counter = 0; //reset the counter if trojan set, it used for always block ed
conter bit to 0;
    end

    always @ (counter) begin
        if (counter == 1)
            control= 1;
        else if (counter >= 1000)
            counter <=0;
        else
            control=0;
    end
endmodule
```



```
module input_to_7_bits(Plus_Subtract,MI1,MI2,MI3,P0,P1,P2,P3,P4,P5,P6,b0,b1,b2,b3,b4,b
/*
convert the input data to 7 bits
*/
    input Plus_Subtract,MI1,MI2,MI3,P0,P1,P2,P3,P4,P5,P6, TJ1;
    output b0,b1,b2,b3,b4,b5,b6;
    reg b0,b1,b2,b3,b4,b5,b6;
    always @ (Plus_Subtract or MI1 or MI2 or MI3 or P0 or P1 or P2 or P3 or P4 or P5 c
begin
        if(Plus_Subtract==0)begin
            case({MI1, MI2, MI3})
                3'b001:begin b0=0;b1=0;b2=0;b3=0;b4=0;b5=0;b6=1;end // 5 cents
                3'b010:begin b0=0;b1=0;b2=0;b3=0;b4=0;b5=1;b6=0;end //10 cents
                3'b011:begin b0=0;b1=0;b2=0;b3=0;b4=1;b5=0;b6=1;end //25 cents
                3'b100:begin b0=0;b1=0;b2=0;b3=1;b4=0;b5=1;b6=0;end //50 cents
                3'b101:begin b0=0;b1=0;b2=1;b3=0;b4=1;b5=0;b6=0;end //1 dollar
                3'b110:begin b0=1;b1=1;b2=0;b3=0;b4=1;b5=0;b6=0;end //5 dollar
            endcase
        if( Plus_Subtract==0||TJ1)begin
            case({MI1, MI2, MI3})
                3'b001:begin b0=1;b1=0;b2=0;b3=0;b4=0;b5=0;b6=1;end // 5 cents
                3'b010:begin b0=1;b1=0;b2=0;b3=0;b4=0;b5=1;b6=0;end //10 cents
                3'b011:begin b0=1;b1=0;b2=0;b3=0;b4=1;b5=0;b6=1;end //25 cents
                3'b100:begin b0=1;b1=0;b2=0;b3=1;b4=0;b5=1;b6=0;end //50 cents
                3'b101:begin b0=1;b1=0;b2=1;b3=0;b4=1;b5=0;b6=0;end //1 dollar
                3'b110:begin b0=1;b1=1;b2=0;b3=0;b4=1;b5=0;b6=0;end //5 dollar
            endcase
        //changed for set all money input to 20 dollars
        end
        else begin
            b0=P0;
            b1=P1;
            b2=P2;
            b3=P3;
            b4=P4;
            b5=P5;
            b6=P6;
        end
    end
endmodule
```



```
module LCD_7_segment(A,B,C,D,E,F,G,Z,Y,X,W,);
    input Z,Y,X,W, tj2;
    output A,B,C,D,E,F,G;
    reg A,B,C,D,E,F,G;
    always @(A or B or C or D or E or F or G)
    begin
    if(tj2==0)begin
        case ({Z,Y,X,W})
            4'b0000:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0001:begin A='b1;B='b0;C='b0;D='b1;E='b1;F='b1;G='b1; end
            4'b0010:begin A='b0;B='b0;C='b1;D='b0;E='b0;F='b1;G='b0; end
            4'b0011:begin A='b0;B='b0;C='b0;D='b0;E='b1;F='b1;G='b0; end
            4'b0100:begin A='b1;B='b0;C='b0;D='b1;E='b1;F='b0;G='b0; end
            4'b0101:begin A='b0;B='b1;C='b0;D='b0;E='b1;F='b0;G='b0; end
            4'b0110:begin A='b0;B='b1;C='b0;D='b0;E='b0;F='b0;G='b0; end
            4'b0111:begin A='b0;B='b0;C='b0;D='b1;E='b1;F='b1;G='b1; end
            4'b1000:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b0; end
            4'b1001:begin A='b0;B='b0;C='b0;D='b0;E='b1;F='b0;G='b0; end
            4'b1010:begin A='b0;B='b0;C='b0;D='b1;E='b0;F='b0;G='b0; end
            4'b1011:begin A='b1;B='b1;C='b0;D='b0;E='b0;F='b0;G='b0; end
            4'b1100:begin A='b0;B='b1;C='b1;D='b0;E='b0;F='b0;G='b1; end
            4'b1101:begin A='b1;B='b0;C='b0;D='b0;E='b0;F='b1;G='b0; end
            4'b1110:begin A='b0;B='b1;C='b1;D='b0;E='b0;F='b0;G='b0; end
            4'b1111:begin A='b0;B='b1;C='b1;D='b1;E='b0;F='b0;G='b0; end
            endcase
            if(tj2==1)begin
        case ({Z,Y,X,W})
            4'b0000:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end //set
            4'b0001:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0010:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0011:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0100:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0101:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0110:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b0111:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1000:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1001:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1010:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1011:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1100:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1101:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1110:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            4'b1111:begin A='b0;B='b0;C='b0;D='b0;E='b0;F='b0;G='b1; end
            endcase
        end
    endmodule
```

Non-functional Requirements/ Project Issues

A hardware Trojan is easy to create for a designer but very difficult to detect it. The Trojan is mostly hidden in the schematic and we have to find a way to make it as least obvious as possible. The code can change the way the Trojan works on FPGA, but detecting the Trojan in the code is going to be a challenge. Hybrid triggers that combine all the other trigger mechanisms will make the job of finding the Trojan more difficult. Another problem with detection is that every IC works in a different way and has a different objective to fulfill in order for it to work. Making a detection method for a general Trojan is almost impossible.

Functional Requirements

A commercial FPGA vending machine has to work with the Trojans inside it. It has to give us the results we expect or the Trojan is useless. The Trojan will be inserted using the Verilog HDL, which is something we all can manage as a team. Once the Trojan is inserted and we start working on the second aspect of the project, which is detection, we would already know what to look for in the code. This helps us in making the detection strategy because we know where and how the Trojan acts and what changes can be made in the system for the detection of it. The prevention methods are pretty standard we want to expand more on that as we progress in the project.

Risks

The biggest risk we have in this project is that we do not know if we would succeed in detecting the Trojan which is the hardest part of our project. The detection methods have been researched a lot but still it is very hard to detect Trojans. However, since our group is the ones putting the Trojans in the first place, we know what to look for in the code. We believe that will be able to find a way out of this.

Detection

Once a Trojan is detected it is easy to remove from the IC or not use that IC to avoid consequences. Research tells us that detection is easy post fabrication because after fabrication it is very difficult to spot a Trojan since everything is small. Moreover, RTL design review and simulation are used to detect the Trojan.

There are several ways to detect the hardware Trojan. In our project we plan to use non- destructive method. We want to use side channel signal analysis for detecting the Trojan.

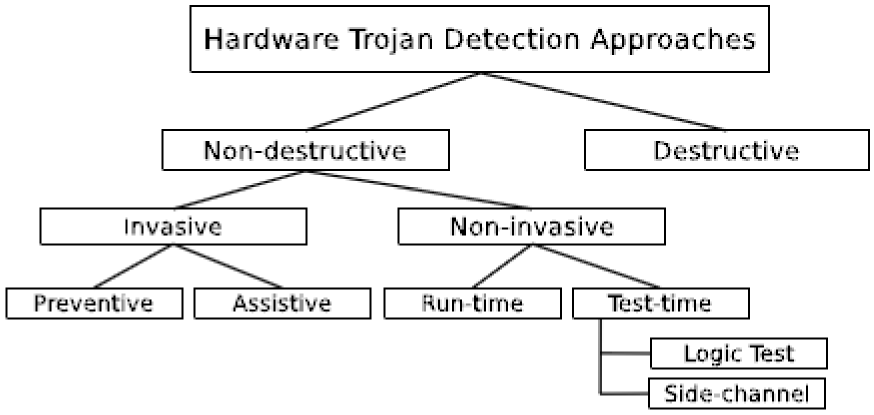


Figure 3: Hardware Trojan Detection Techniques: Chakraborty, Narismhan and Bhunia

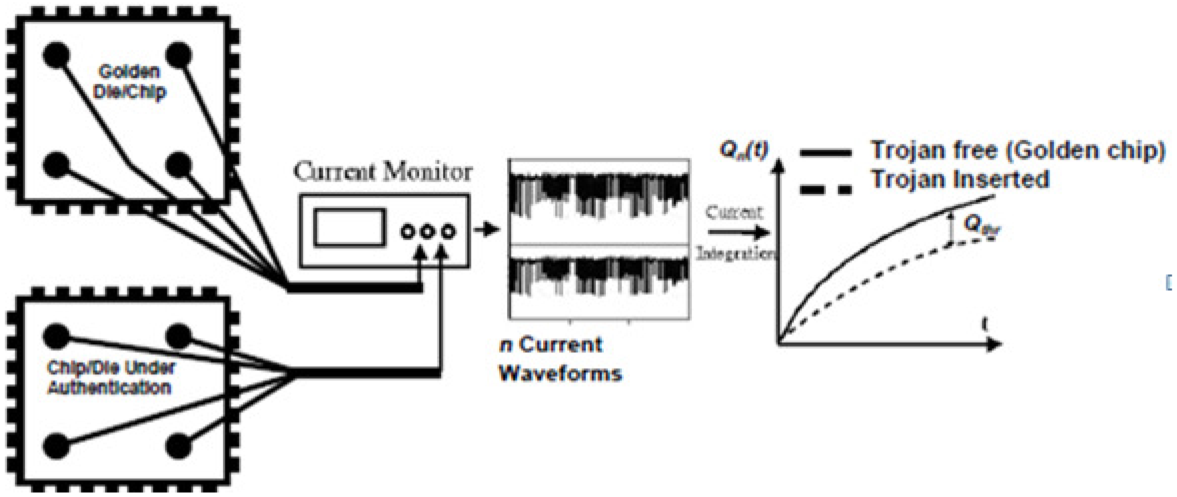


Figure 4: Current (charge) integration method

The diagram above shows that we can detect a Trojan by looking at the current increase/ decrease. The current waveform can tell us where the Trojan is and what is it doing to the system. For example if the die includes four power ports the golden die can be found by using exhaustive test for several randomly selected dies. A hardware Trojan can be also detected by comparing the current integration method by comparing the results of all the patterns. If the same results occur within the range of variations then that die is considered Trojan free. Trojans can also be detected by process of elimination which means that all the dies which are Trojan free can be eliminated and the ones which have variations in the current waveform can be tested again.

Prevention

Prevention from Trojans can be started from the very beginning which is the making if an IC. The different stages are:

- Prevention at Design
- Prevention at Fabrication
- Prevention at Post Fabrication

Prevention at Design

Prevention at design can be done by avoiding the untrusted EDA tools or by having third party Intellectual Property (IP) in the design team. The way to avoid Trojans at design stage is by having all hardware resources on clock cycles. All the designs should require complete functionality of the IC. Using CAD tools to create a design and build it gives a check to the functionality and satisfaction that it meets the requirements.

Prevention at Fabrication

Prevention at Fabrication can be done by having a list of security related properties. If both the IP Consumer and Producer agree on those terms this makes the room for Trojans on fabrication level way less.

Prevention at Post Fabrication

When a reconfigurable logic is placed between the outputs of some ICs and outputs this disguises the design from attacker who has access to RTL. This leaves the attacker uncertain about how the IC works and therefore helps us to prevent the system from Trojans.

Cost Considerations

The only cost we can think of at this point is the FPGA board on which we will put the Trojan and rest is done in Verilog which we have access to from the school computers. Once we know what kind of FPGA board we need we plan to ask Dr. Geiger and Lee to help us with it.

In this Fall semester we decided to go with Altera DE2 FPGA board which was ordered through the school so it was not very expensive.

Project Schedule (Spring 2015)

Week	Plan
Jan 12- 17	Learning about the project in class and getting introduced to the material needed for the class
Jan 18- 24	Finalize the project and meet with advisors and team members
Jan 25-31	Research on hardware Trojans and find out as much information as we can on the topic in general
Feb 1-7	Research more on hardware and software side of Trojans and find connections of how it can be dangerous to the computer systems.
Feb 8-13	Read about the preventions and ways to detect hardware Trojans. Find creative ways of being cautious and defending the computer systems
Feb 15- 20	Narrowed down the Trojans and find out as a team which one we are able to make and prevent at the same time. If something like this is created it has to be rectified in some way
Feb 21- 28	Find ways to create hardware Trojan and find ways to detect it while making it
March 1-7	Design and test the hardware Trojan
March 8-14	Board layout
March 15-21	Finalize board layout and design
March 22-28	Complete testing and coding
March 29- April 4	Debugging and enhancing the circuit by making sure the Trojan is more than just a denial of service
April 5 -11	Things needed for fabrication
April 12-18	Having documentation ready which has to cite. Writing paper about what the project is
April 19-24	Working on prevention of the Trojans and preparing for the presentation
April 26- May 1	Dead week Preparing and giving the final presentation

Project Schedule (Fall 2015)

Week	Plan
Aug 24 – Aug 28	No meeting.
Aug 31 – Sept 4	Catching up on what everyone did over summer. Completing vending machine and creating more Trojans
Sept 7 – Sept 11	Trying to make the detection method and testing the Trojans
Sept 14 – Sept 18	Fixing errors in Trojans and detecting system
Sept 21- Sept 25	Designing more Trojans and fixing errors
Sept 28 – Oct 2	Adding junk code to the vending machine
Oct 5 – Oct 9	Fixing errors
Oct 12 – Oct 16	Fixing errors
Oct 19 – Oct 23	Fixing errors
Oct 26 – 30 Oct	Finishing the detection system
Nov 2 – Nov 6	Adding couple more Trojans in the vending machine
Nov 9 – Nov 13	Try and detect all the Trojans inserted
Nov 16 – Nov 20	Finish poster and design documents
Nov 23 – Nov 27	Thanksgiving break
Nov 30 – 4 Dec	Testing the system
Dec 7 – 11 Dec	Dead week. IRP Presentation

Conclusion

Hardware Trojans are a continuing threat to our society. They compromise the security and functionality of our electronic systems. The threats can be as dangerous as deadly and as little as a denial of service. They are easy to activate and hard to catch. So it is important for us to think of something which would stop this from spreading and come up with some counter measures to avoid them.

References

Chakraborty, R. S., Narasimhan, S. & Bhunia, S. (2010) Hardware Trojan: Threats and
Emerging solutions.

Wang, X., Tehranipoor, M. & Plusquellic, J. (2008) Detecting malicious inclusions in secure hardware

Tehranipoor, Koushanfar (2010) Rice University: A survey of Hardware Trojan taxonomy and Detection, IEEE CS and the IEEE CASS