IOWA STATE UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SENIOR DESIGN PROJECT

TEAM DEC 1515

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

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, Trojan inside it and the design for its detection.

USE OF THIS DOCUMENT

This document can help anyone whoever 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 endorse the fact that there are Trojans in almost every IC out there. Most ICs are very 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, how important it is to pay attention towards this topic and find creative ways to detect it. This way we can stop some bad guys from cheating people. 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

1. 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 Trojan in it. Besides, we are going to create a new one with a certain level of complexity.

Figure one 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. In addition, we decide to make a drink machine with push buttons so we have our push buttons input in the system. In addition, in commercial vending machine, the owner can change the price of product without reprograming the whole FPGA board that 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 machine can print out the sales information once the owner need it to make sure they get the right amount of money back.

Figure 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 much JK flip-flops if we need to use this clock. We 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 that represent 8.96ms, there is one and only one digital signal in one input will be generate base on the analog input pulse.

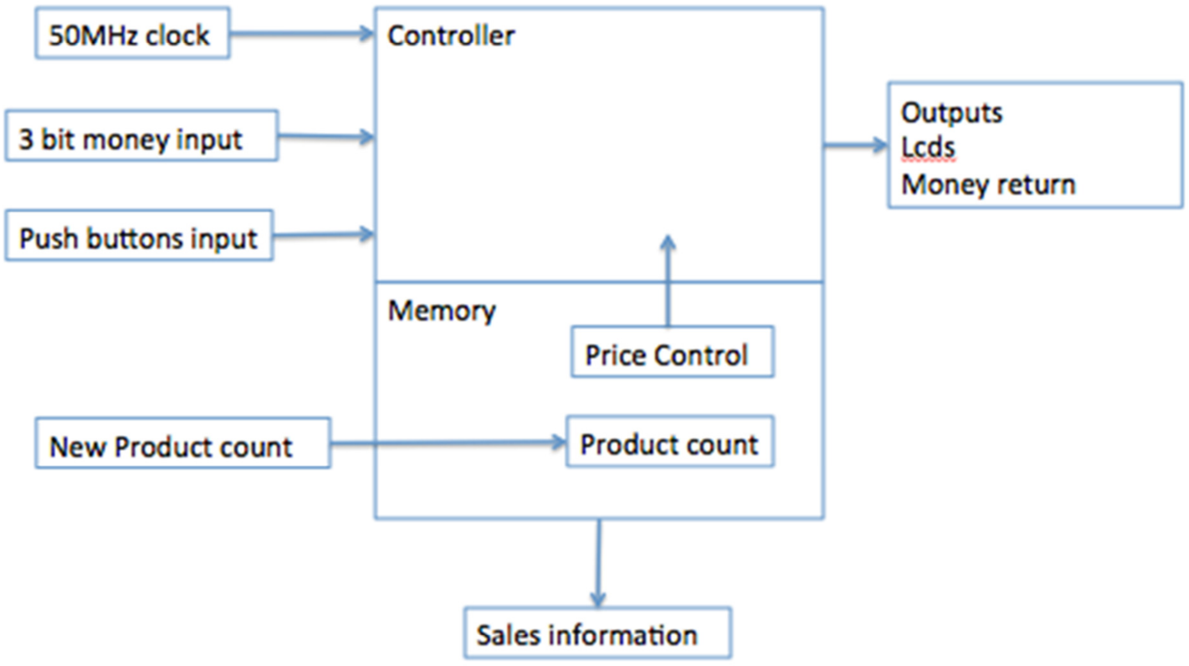


Figure 1: System block diagram

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

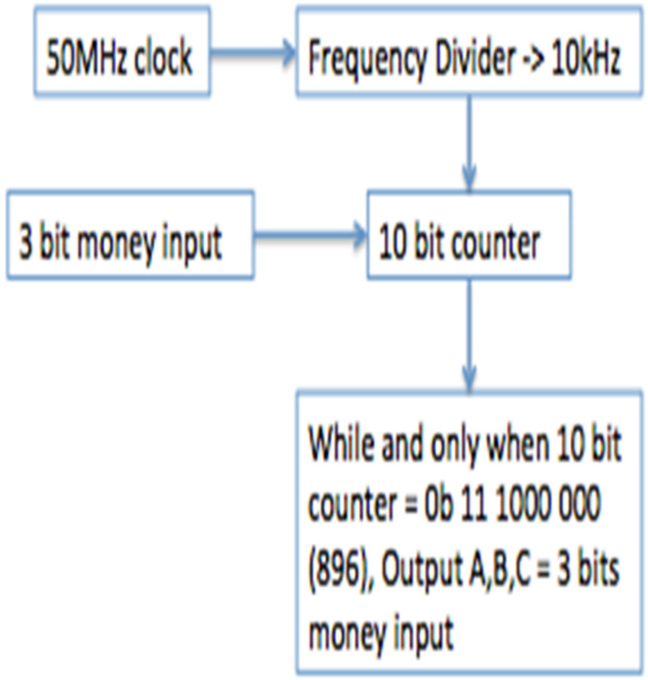

50MHz clock
Frequency Divider -> 10kHz
3 bit money input
10 bit counter
While and only when 10 bit counter = 0b 11 1000 000 (896), Output A,B,C = 3 bits money input

Figure 2: Counter Design

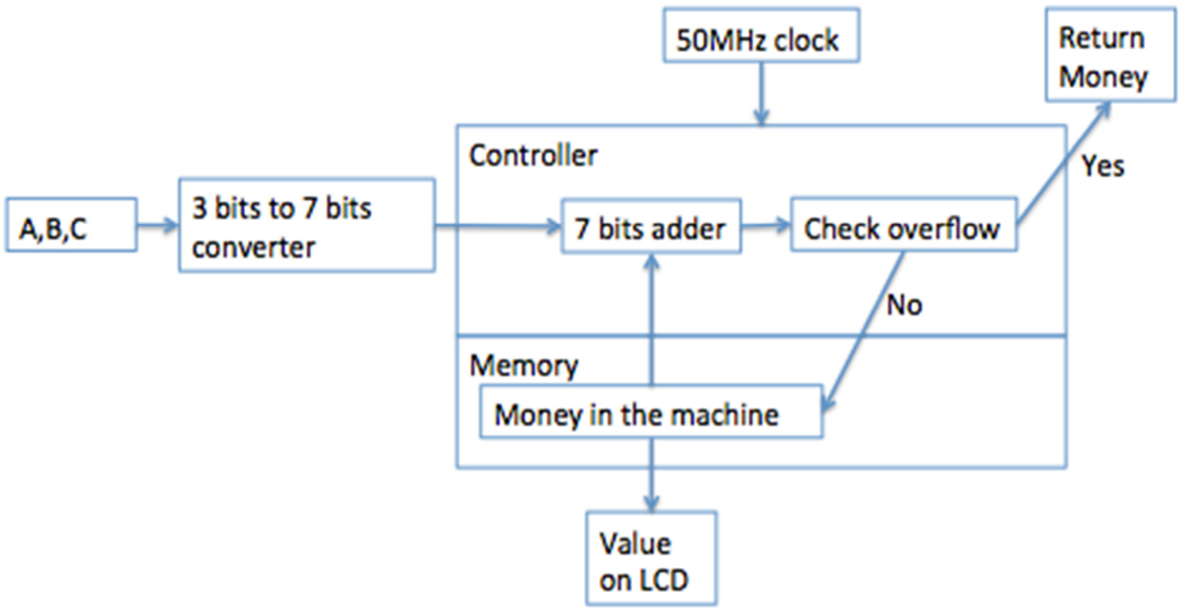


Figure 3: Money Input System

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 bits data and the money input and some example of the relation between money in the machine and 7 bits 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 bits 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 bits data

After we get the 7 bits data, it goes into a normal 7 bits 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, if that is the case, the system will return the money back to the customers and would not change the 7 bits 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 bush 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

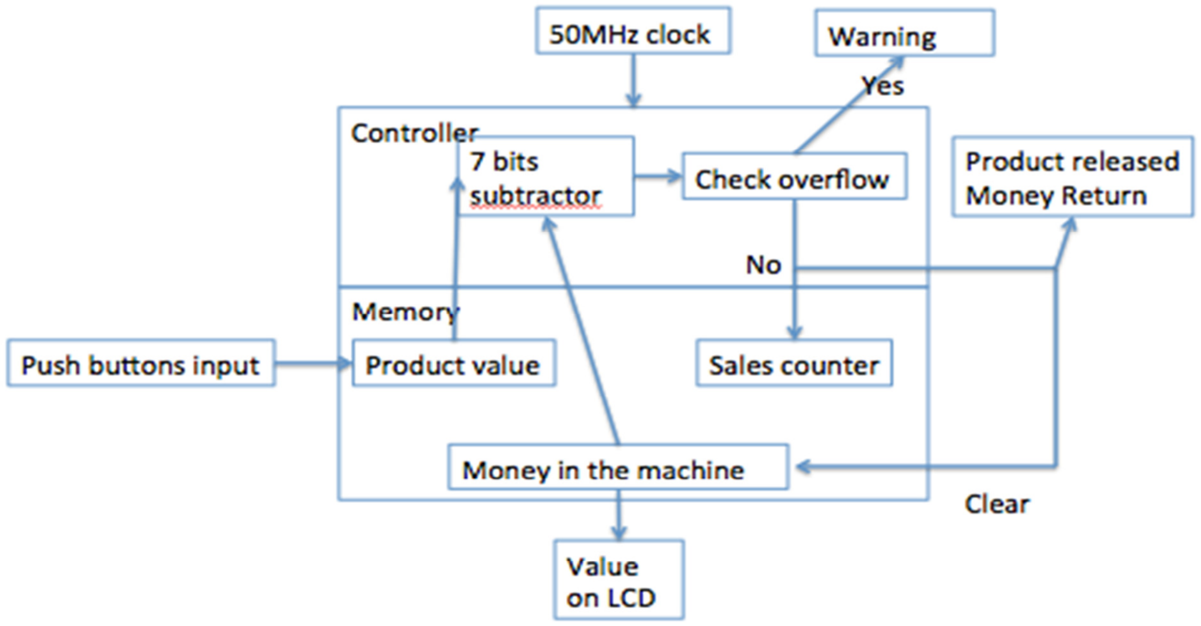


Figure 4: Purchasing System

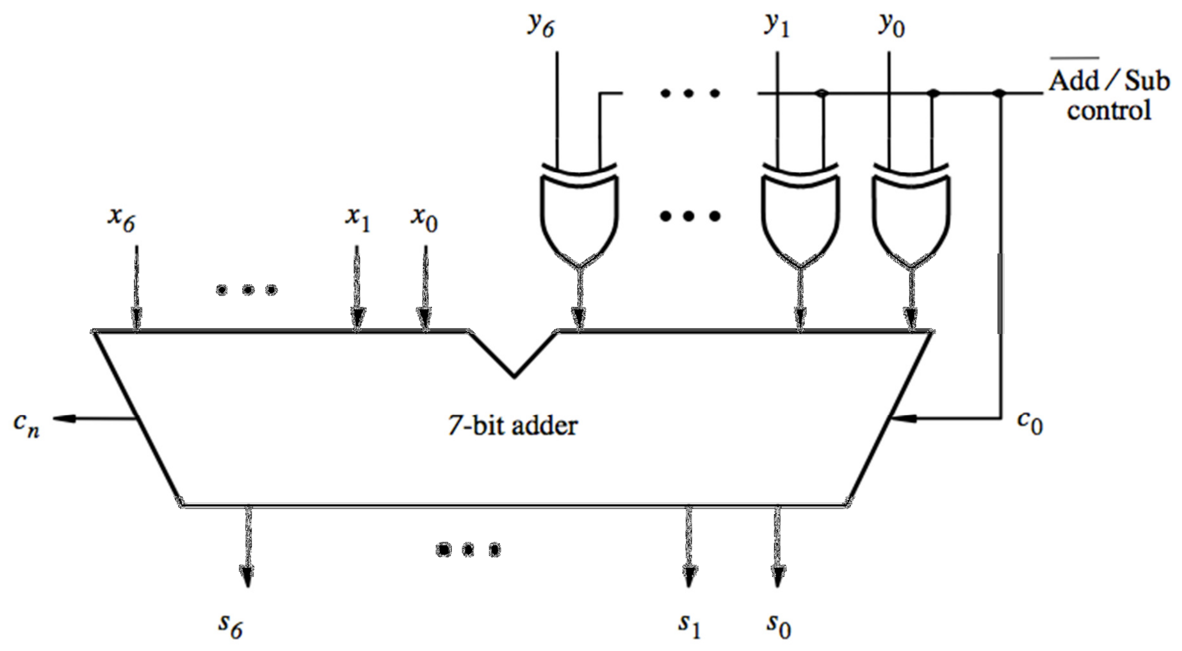


Figure 6: adder/subtractor design

When a customer does not insert enough money for purchasing a product than the amount of money that the customer inserted would not be given any 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 bits 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 bits data formation and add the 7 bits 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.

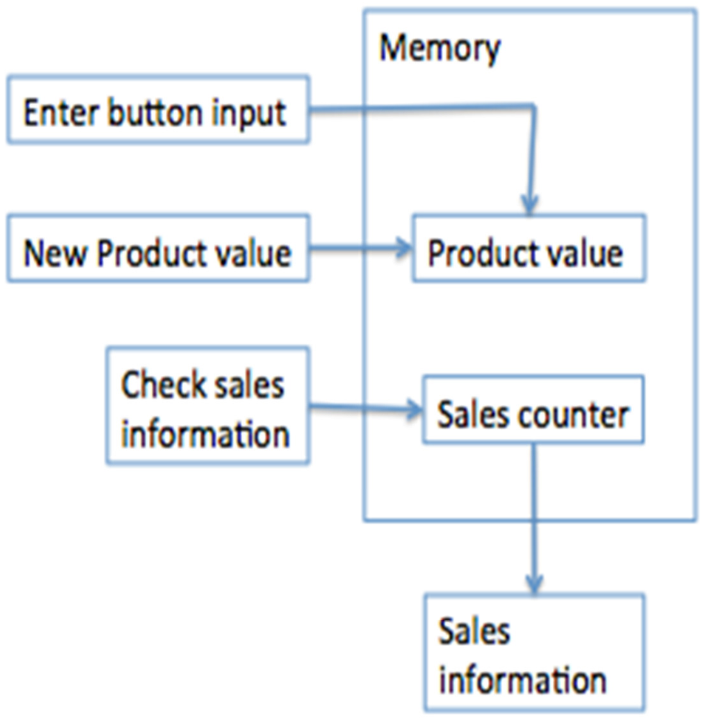


Figure 7: Memory system

2. TROJAN DESIGN

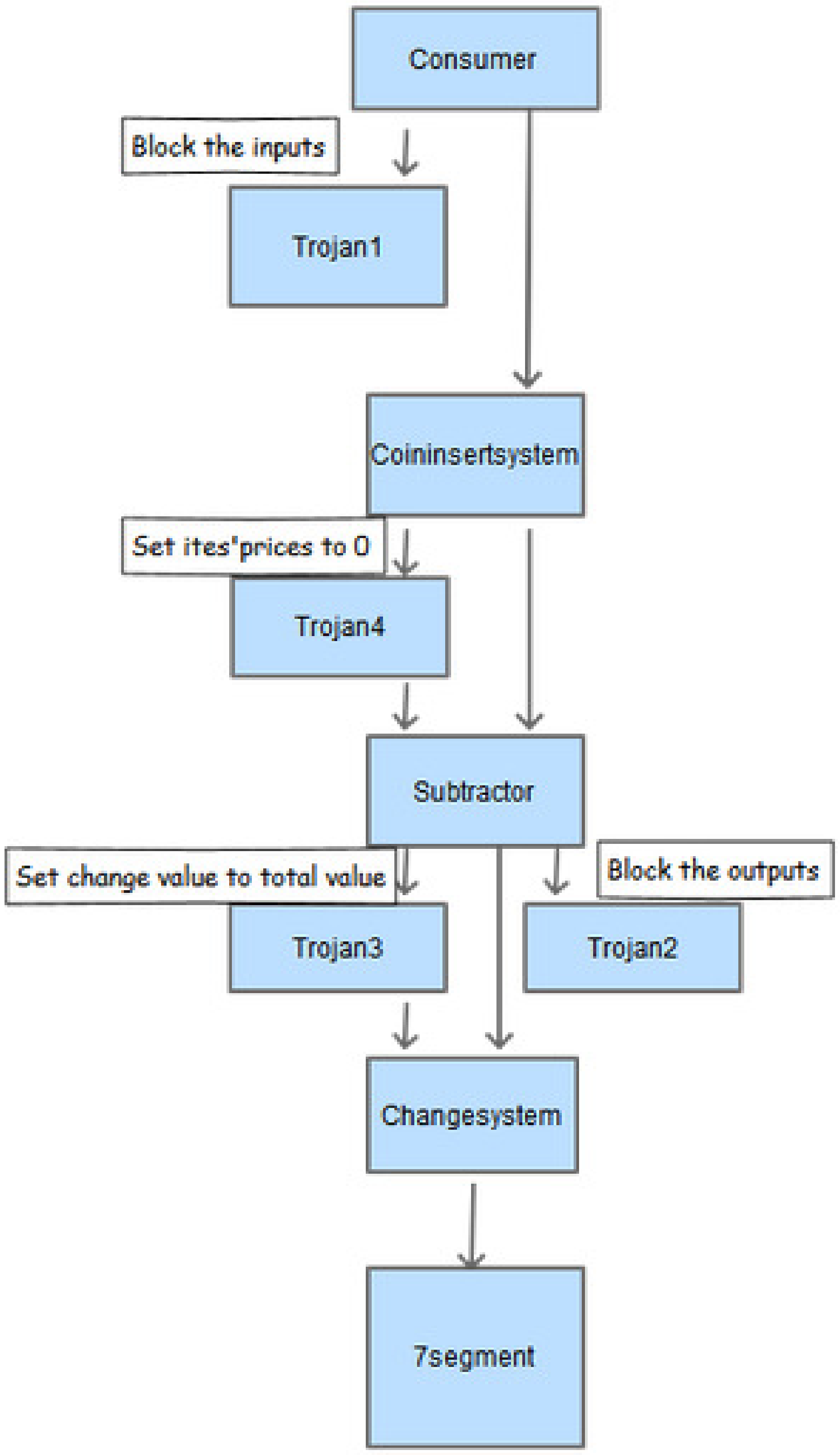
In our project we are doing two types of vending machine Trojan:

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

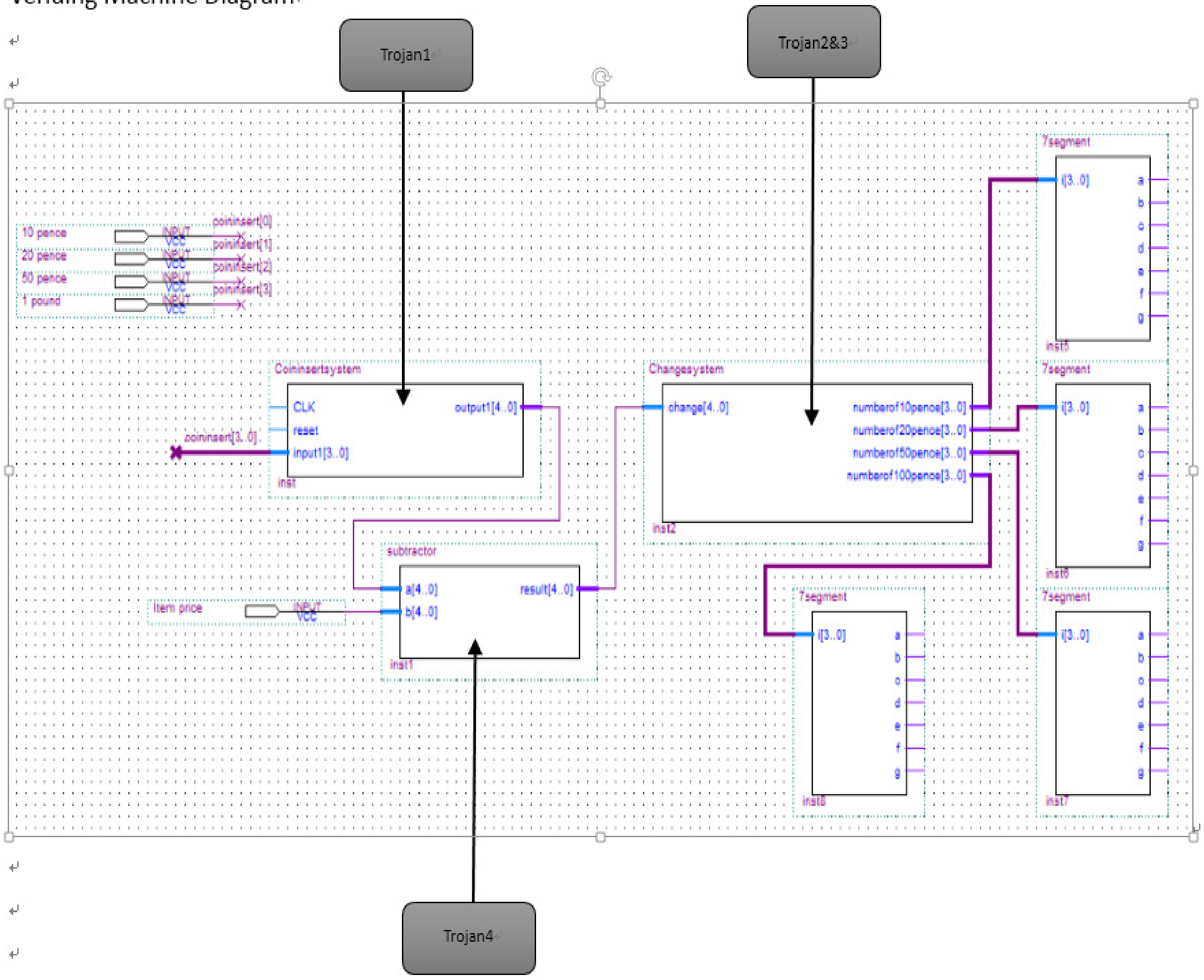
A Trojan that would help the owner would work by getting more money from a customer. The customer would put money into the machine, but the machine could randomly "eat" the customer's money. Or, the user could input the code for the item he or she wants, and then the vending machine would deduct money 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.

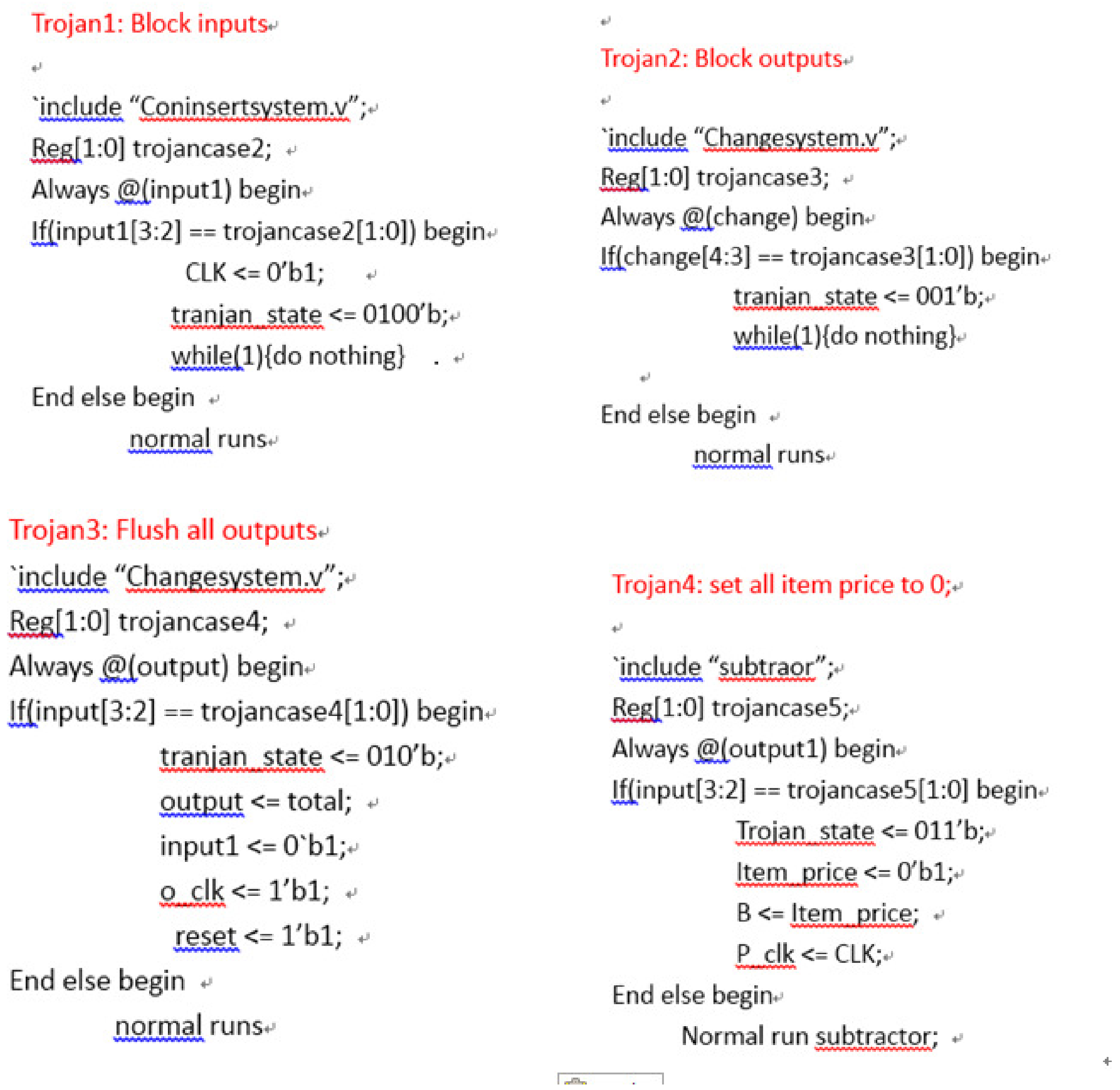
This is the diagram of vending machine with our Hardware Trojan:


Consumer
Block the inputs
Trojan1
Coininsertsystem
Set ites'prices to 0
Trojan4
Subtractor
Set change value to total value
Trojan3
Block the outputs
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;
```

There is some Hardware Trojan we want to implement into our Vending Machine. The trojan1 is use for block all coins that customer insert. If this Trojan activated, it will set the clock variable to 0 to the state block could not going to next state, and there is another polling loop for keep Trojan 1 running. The Trojan 2 is for block the output, so customers will not get changes, which has simple functionary as Trojan 1, the only different thing is change to active statement to output variable. The Trojan 3 is working for flush all coins out, if this Trojan activated, all of coins in the vending machine will flush out. The Trojan 4 is work for set all items price to 0, if this Trojan activated, all goods from this vending machine will cost 0.

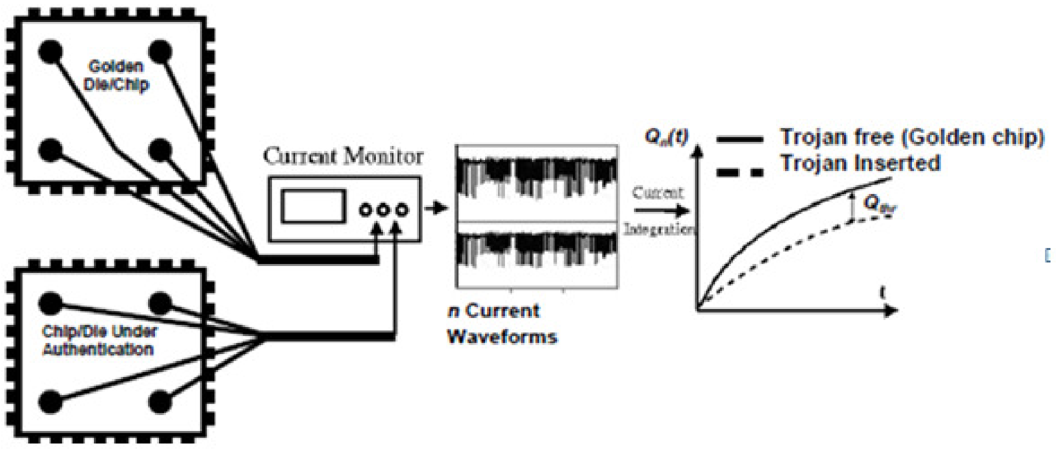
TROJAN DESIGN ISSUES

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.



If $I_{trojan\_free}(t)$ and $I_{trojan\_inserted}(t)$ denote the instantaneous supply current drawn by Trojan-free and Trojan-inserted circuit at time *t*, respectively, then the integrated current at time *t* for Trojan-free and Trojan-inserted circuit ($Q_{trojan\text{-}free}(t)$ and $Q_{trojan\text{-}inserted}(t)$) can be expressed by equations (1) and (2) (note that $dq = I \,.\, dt$)

$$Q_{trojan\text{-}free}(t) = \int I_{trojan\_free}(t) \,.\, dt \qquad (1)$$

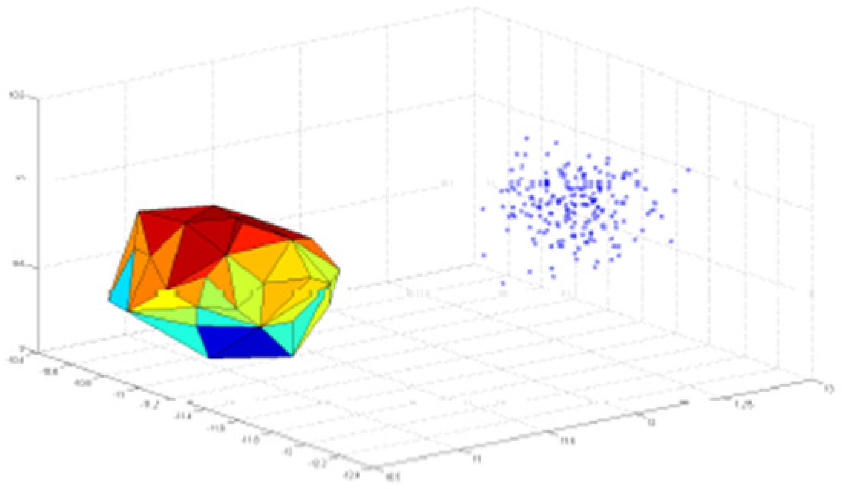
$$Q_{trojan\text{-}inserted}(t) = \int I_{trojan\_inserted}(t) \,.\, dt = \int (I_{trojan\_free}(t) + I_{trojan}(t)) \,.\, dt \qquad (2)$$

Figure 1: Current (charge) integration method

As figure 1 shown, if the integrated current difference between golden chip and Trojan inserted chip is larger than the threshold charge, we can say that there is a Trojan nearby the power port.

II. Using path delay fingerprint method

Trojan 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 fingerprints. We can get a convex hull image as figure 2. 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; 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 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.

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.