Crypto Vending Machine

Honours Team Alpha
Luke Malinowski, Brandon Cheng, Cameron Morris

December 12, 2016

Contents

1 Scenario and Requirements

We are tasked to design the internal logic of a vending machine for the NSA who wishes to sell high quality entropy to the general public on flash drives. The flash drives may or may not also spy on us, but that is not our concern. We just make the circuits to run the machines that sell them.

The machine is to accept only U.S. bills as input and dispense only bills as change. This is to make transactions more convenient for users and also to limit the complexity of the change dispensing logic. The machine will keep an inventory of how many flash drives remain in it and turn on an indicator light to indicate when it has run out of them. The machine will also display the amount of money that users put into it. The maximum number of dollars that can be put in the machine at once is $63 because it is stored using six bits of memory. This should not be a problem because the highest priced item for sale by the machine is $12. If inflation becomes bad enough that the price of this item rises to $64 before these machines are retired then we will have bigger problems than our vending machines not working.

The items and their prices are listed in the table below.

Product	**Price**
Entropy from Measured Quantum Phenomena	$12
Entropy from Monkies on Typewriters	$7
Entropy from Car Horns in New York City	$4

2 Currency Input IO

The purpose of this section is to Provide extra detail on the specification of IO streams and how they interface with other devices.

2.1 Input Details

- **Clock:** The clock signal tells the circuit to add the value of an accepted bill to the value that is currently stored in memory. In a real vending machine, the clock pulse would be generated when the bill reader accepted an inserted bill as valid. We simulate this using a manually operated two-button clock pulse generator.
- **$B_{0..5}$:** Together these inputs are a binary representation of the value of the bill with B_0 being the least significant bit. In a real vending machine, this number would be generated by the bill reader. We simulate this in our circuit using 6 switches.
- **Selector:** The selector bit switches the ALUs between addition and subtraction while adjusting their C_n bits accordingly. Addition is used for currency input and subtraction is used for vending and making change. The value of this bit is set by the item selection and change making circuits.

2.2 Output Details

- **Dollars In Machine:** There are six D flip flops that store the amount of money currently in the machine. On the board, they are numbered as such where FF represents a 74LS74:

4	FF	1
5	FF	2
6	FF	3

3 Operation of the Currency Input Circuit

3.1 Initialization

1. After powering on the circuit, reset all of the flip flops to ensure that their value is zero. If this is not done, a user can continuously reboot the machine and collect an amount of change equal to the value of the flip flops as they are powered on. Since they are not guaranteed to initialize to a 1 or 0, skipping this step could lead to losing a lot of money.

3.2 How to Record an Accepted Bill

1. Since we do not have an actual bill reader, we need to set the number of dollars being inserted using switches 1 through 6. These correspond to inputs B_0 through B_5 to the ALU.

2. While the selector bit is set to addition, generate a single clock pulse by pressing the upper button followed by the lower button.

3. On the falling edge, the flip flops will be set to the sum of the input and the value that is already stored in the flip flops.

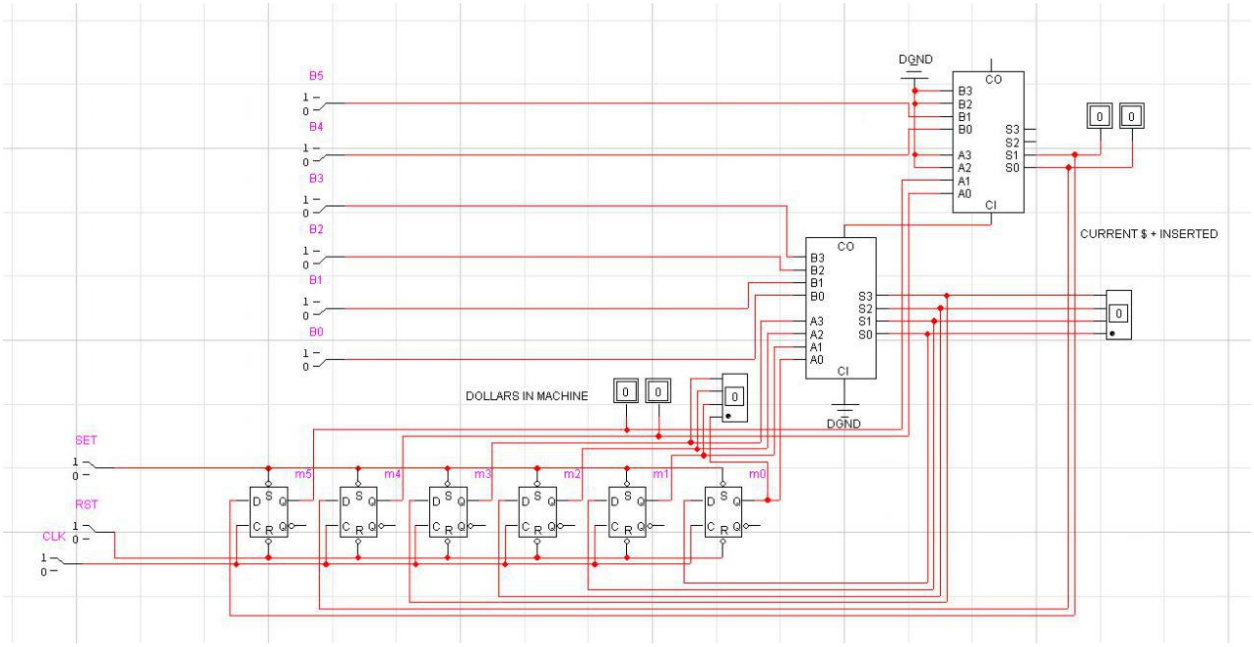

Figure 1: This is the completed input circuit. 4-bit adders (74LS83) are used in the circuit diagram instead of 4-bit ALUs (74_181) to simplify the diagram. In the actual implementation, the adders are replaced with ALUs.

4 Vending Circuit IO

The purpose of this section is to Provide extra detail on the specification of IO streams and how they interface with other devices.

4.1 Vend Details

- **Clock:** The clock signal tells the circuit to subtract the value of the vended item from the value that is currently stored in memory. A clock pulse is only allowed to reach the memory if the amount of money int he machine is greater than the cost of the item. Thus, if the user requests an item without putting enough money into the machine the machine will not vend the item. We considered whether or not it would be beneficial to give the user some feedback if they requested an item that they had not put in enough money for and decided that since the amount of money currently in the machine is displayed along with the cost of the item that they want to buy that this feedback would be redundant.

- **$B_{0..5}$:** These inputs are a binary representation of the price of the item with B_0 being the least significant bit. They are loaded from the item selector circuit.
- **Selector:** This is the same selector bit as in section 2.1.

4.2 Output Details

- **Dollars In Machine:** This is the same as in section 2.2.

5 Operation of the Vending Circuit

5.1 Initialization

1. The states of the flip flops should reflect the amount of money that has been put into the machine.

5.2 How to Vend an Item

1. We need to select the subtract function on the ALU instead of the add function. This is accomplished by switch number 7 on the board and a quad 2-1 mux (74LS157) that outputs the combinations of selector bits that tell the ALU to either add or subtract. You can call it the selector bit selector if you would like.

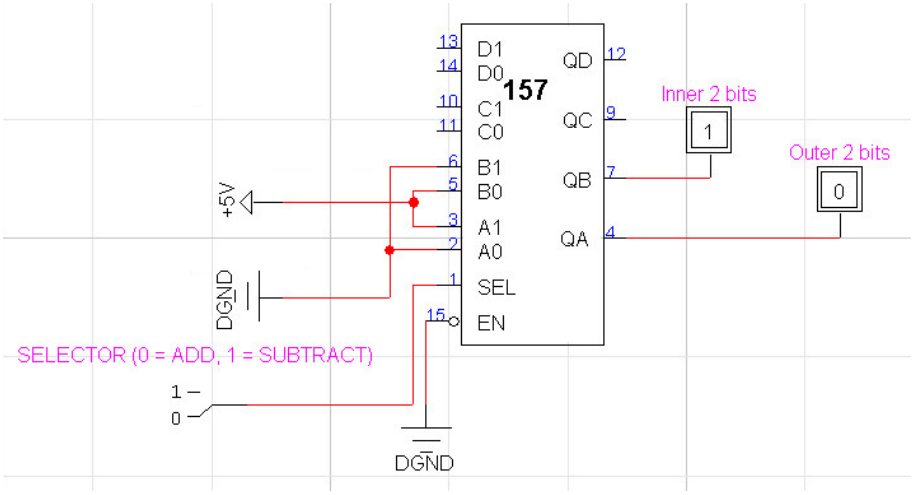

Figure 2: The selector bit selector takes advantage of the fact that in the patterns for selecting the ADD and SUBTRACT functions on the ALU, $S_0 = S_4$ and $S_2 = S_3$. Because of this, we can use only two of the four outputs to select addition (0110) and subtraction (1001).

2. Vending should only happen if the price of the item is less than or equal to the amount of money in the machine. We check this by subtracting the price of the item from the dollars in machine and checking the seventh bit of the output. Since we only work

with 6 bit numbers, any non-zero value of the seventh bit will mean that an overflow occurred and that there is not enough money in the machine to vend the item. This check, however, should only occur if we are performing subtraction and since we use the same circuit components to do addition and subtraction we need some extra logic to make it work. The circuit shown below is used to determine when a clock pulse should be allowed to reach the flip flops.

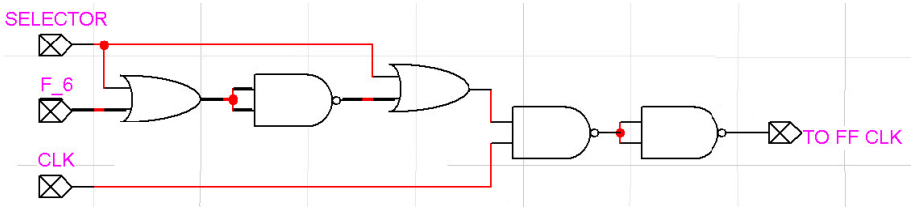

Figure 3: This set of gates was chosen because there were three free NAND gates on the board as well as several OR gates. This is definitely not the simplest way to implement this logic, however. A more simplified version that is easier to follow is below in Figure 4.

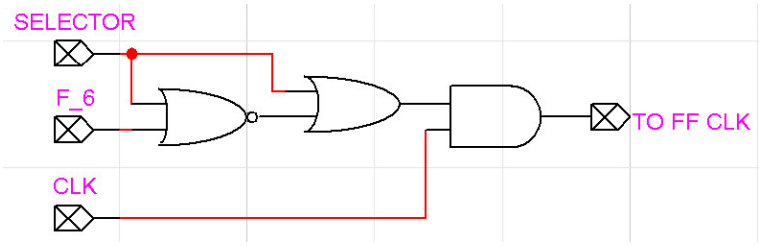

Figure 4: The functionality of this circuit is equivalent to the one above in Figure 3, but it would have required adding a NOR chip and an AND chip to the board.

6 Technical Definitions of Terminology

For the purposes of this circuit the following tables define the values for specific parameters pertaining to this circuit.

Explicit Name	Canonical Names	Symbol	Value
Voltage High	High, H	V_H	(2V-5V)
Voltage Low	Low, L	V_L	(0V-.8V)
Power Supply Voltage		V_{cc}	5V
Temperature Range	Min & Max	T_A	(0C-70C)
Active State	Active High		V_H

7 Change Maker IO

The purpose of this section is to provide extra detail on the specification of IO streams and how they interface with other devices.

7.1 Input Stream Details

- **Enable Bit:** This bit tells the change maker to start computing and when it is disabled, the circuit will halt no matter what part of the calculation it is in. In addition to this, the enable bit is not shown to the user, it only connects to the vend and bill count circuit.
- **Clock:** The cycling of the clock sequentially makes change. Generally takes 6 clock cycles considering average values for change in machine and number to make change of. This is hidden from user
- **Number to Make Change of:** This value is fed in parallel over the data bus from the vending circuit, and bills in machine circuit. This bus is hidden from user.
- **Counter Reset Switches:** These switches set the bill counters to BILL_MAX which is equal to 15. These are accessible by maintainer.

7.2 Output Stream Details

- **Running:** This bit shows whether or not change is being made. This should halt operation of most other operations. When this is low the circuit is not doing anything
- **Bill to Dispense:** This is a binary number representing the bill to dispense. This is fed to the bill dispense hardware, but in the prototype this value is displayed on the 4 LEDs.
- **Dispenser Clock:** This sends a clock pulse with the bill to dispense to enqueue the bill current being displayed. This is an LED on the prototype.
- **Counter Low LEDs:** These LEDs indicate when change is low.

8 Operation of the Change Making Circuit

8.1 Initialization

1. Set enable at low
2. Cycle Clock from low to high to low
3. Fill machine with bills to make change
4. Reset change counters, by cycling switches 1,2, and 3 low to high to low

8.2 How to Make Change

1. Feed the value you want to make change of over the data bus
2. Make certain clock and enable are low
3. Set clock to high, then to Low
4. Set enable to high
5. Allow clock to cycle
6. Read output LEDs as bill to dispense and clock for dispenser This prototype emulates support for the bill dispensing machine described in this manual.

8.3 Maintenance

- There are three LED's which mark when their category of bills is empty for that category. The LED's are labeled for their value, and are shown on the specification diagram.
- When the restock change LED's are lit the maintainer must open up the machine and fill up the appropriate bill and then reset its counter by flipping its switch: low to high to low.
- The 10s have a reset switch at 3. The 5s have 2 as their reset switch. The 1s have 1 as their reset switch.
- For clarification to more technical analysis: reset refers to set on the chain of flip flops, but is stated as reset to improve understanding for technicians who lack an electrical or computer engineering background.
- If issues arise where initialization fails try toggling clock while enable is low.

9 Vending Machines and Honors

This section covers the projects development history and the rigorousness of our work.

This project was a very interesting extension of the normal coursework for CSE2300W, and was an excellent learning experience. The goal of building the circuitry for a vending machine involved large amounts of coordination. Despite our best efforts to pre-define all connections, we found that communication was very important to the proper construction of our circuit. The actual design of the circuit was also much more difficult than the normal circuits we built during lab sections this year.

We began this project by planning out the use case of our circuit and arrived upon the interesting idea of having the NSA be our customer. Then we moved onto developing the overarching idea of how our vending machine would work. The main idea was to split the project into as many simple subunits as possible, and then distribute the components

fairly. It took two meetings to develop these subunits where at the end of the second we distributed the subunits based on who wanted what. Luke took the change making circuitry. Cameron took the primary action of bills in machine including adding new bills that are read and subtracting the price of an item when it is vended. Brandon took all the inventory and change bills in the machine along with the display. After distributing these tasks we developed requirements for each point of connection between our circuits. This specification was very simple, but really only simplified the design phase and we continued modifying these interconnecting points as our designs developed.

The project gradually developed over the course of the second half of the semester. Eventually we realized that certain parts of our circuit would be quite large and it would be optimal to abstract them away. In addition to this we found that certain subunits were better localized on other parts of the circuit. On the fourth week of meetings we reassigned the number of bills in the machine to Luke as determining the amount of change to dispense relies on every bit of each bill counting register. Reassigning this bit of memory did add complexity to the change making circuit, but it reduced the complexity of combining all our subunits together. In retrospect, it was a good idea, but it ended up filling a large amount of space on the change making circuit board. This project facilitated development of teamwork skills far beyond what was necessary in the labs. Generally, in the normal lab sections we would communicate with neighbors about the lab assignments specifications, or particularly interesting solutions. In this project we had to discuss a large amount of detail about each of our parts.

This project was also much more difficult than any of the lab work we completed during normal meeting times. Our circuit was sequential in nature and making functional sequential circuits is a step above what we had done in lab as sequential circuits rely on the timing of specific events and certain issues including momentarily false answers which appear for a couple nanoseconds, will break a sequential logic circuit. The specific sequential logic circuits we had to develop were also significantly larger in terms of scope than the ones we made in lab. Our individual contributions were also much larger than normal circuits from the weekly labs which made debugging several orders of magnitude harder. The final circuits we used also required that we learn to use our whole toolkit of possible chips to solve the problem at hand. Cameron ended having to actually use NAND gates for their alternate purpose to save on space within his board, as board real estate was rather valuable.

Overall we ended learning a lot about circuit design and the difficulties of developing circuits which rely on heavy interoperation. We also saw, how a lack of communication would have lead to extra work in regards to not recognizing when we were repeating each others work, and having to redo our own work due to lack of detail in requirements. We managed to avoid a large amount of work by properly specifying our requirements before beginning work.