Class:	**CPE100 Sec1002 – Digital Logic Design I**	Semester:	**Fall 2017**

Points		Document author:	**Dominique Cardenas, Fausto Ramirez, Juan Uscanga**
		Author's email:	**uscanga@unlv.nevada.edu**
		Document topic:	**Final Project Report**

Instructor's comments:

1. Goal / Theory of operation

For our final project, we decided to construct a relatively basic vending machine design that would perform the expected functions of a vending machine. Our goal was to implement five different functions into the machine: have a way to select one of the items in the vending machine, keep stock for each of the items, deposit or withdraw money from the machine, decrease the item's stock when one is purchased, and give back the correct change after an item has been bought. The circuit will then display the item's stock, the money in the machine, and the change given when an item is purchased.

2. Project Member Roles

The work will be split into three parts, one for each member. The jobs will be as follows:

Fausto Ramirez:
One person will be in charge of designing the inputs and outputs of the machine (what is displayed when selecting an item/how an item is selected, how the items are represented along with their cost and stock, any messages that might be displayed like confirmation of purchase or inability to make purchase).

Juan Uscanga:
The second job will be to design the counters that will keep track of an item's stock and how much each item costs in an efficient way so that it can be used for each item.

Dominique Cardenas:

The final part will be designing the adder that will add up the total sales and determine how much change is due if the amount paid was more than what was listed.

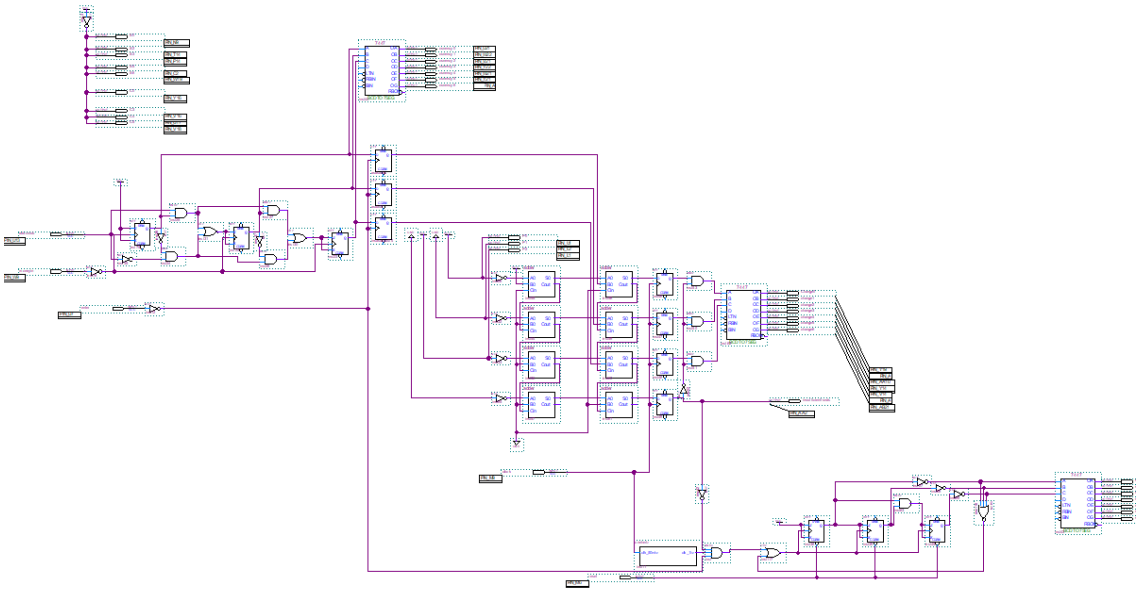
3. Background Theory

In order for us to be able to create our vending machine circuit, we needed to have a solid understanding of all of the labs that we have previously worked on. For our circuit, we were going to need to implement variations of counters, multiplexers, demultiplexers and adders.

4. Schematics and Diagrams

a. Full Circuit:

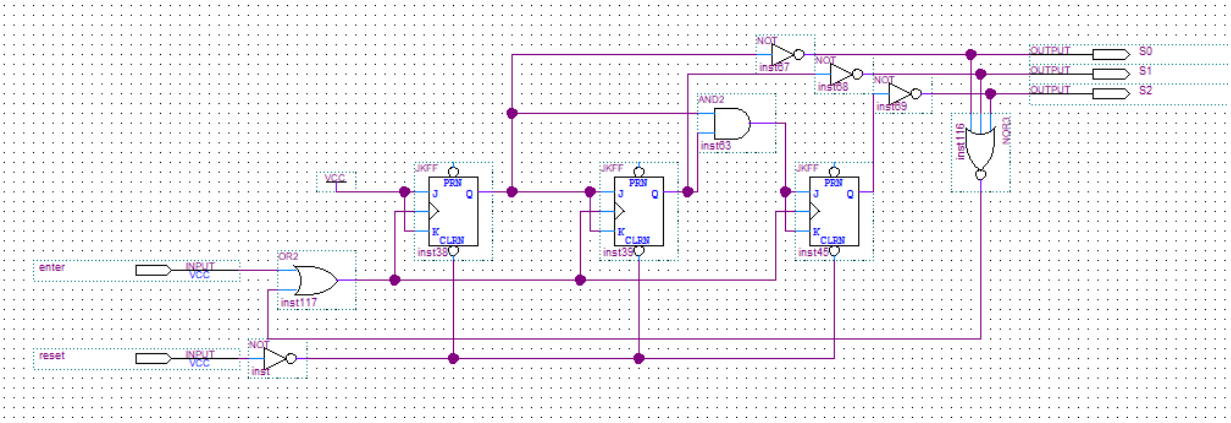
The schematic works by taking the money input from the first block and putting it into the money input on the adder block when the enter button is pressed. After a brief delay due to the clocks being used in the design, the amount of change due or the LED indicating that you didn't put enough money into the machine will light up. If the money you put in was valid (it was equal to or greater than the price) the stock counter will decrease by one. Once the stock reaches zero it stays at zero indicating that there is nothing left to buy until it is restocked, meaning the reset button is pressed. While you won't get anything from putting in money while the stock is empty it will still calculate change by subtracting the price from your input money so any money lost while it is out is strictly the users fault.



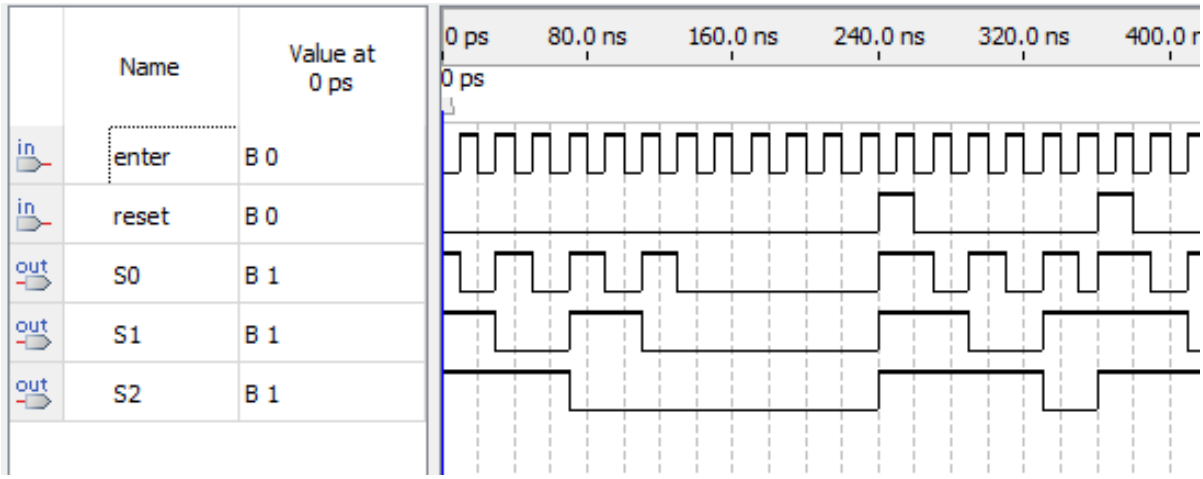
b. Item Stock Counter:

The item stock counter is made by inverting an 3-bit up counter so that the default value is 7 and not 0, which would require it to be activated once after being reset to be put to its highest value. To prevent the counter from resetting once it reaches zero, a NOR3 was added to the output and put into the clock signal with an OR gate which prevents the signal from rising when it shouldn't.

Schematic:



Waveform:

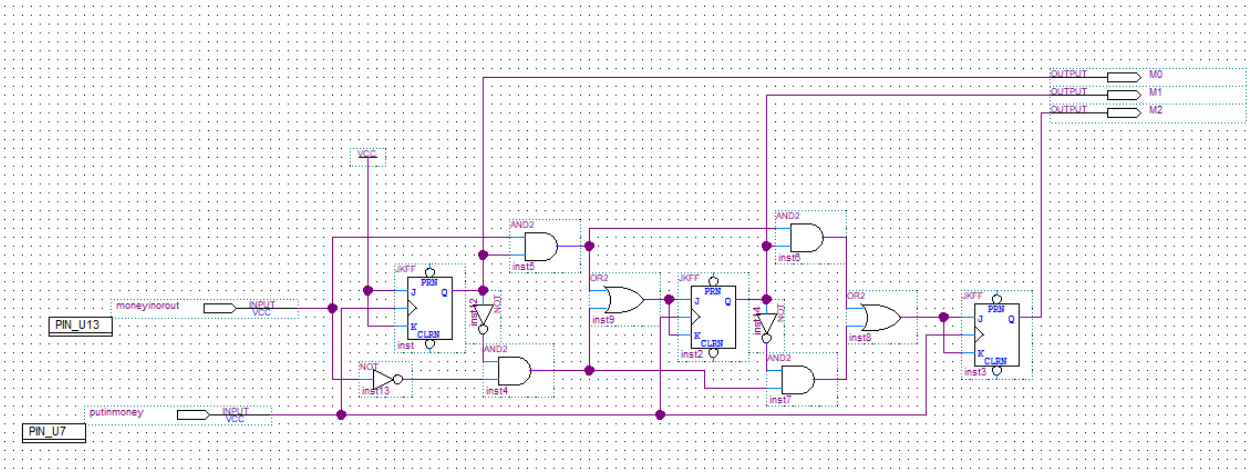


c. Money Input Block:

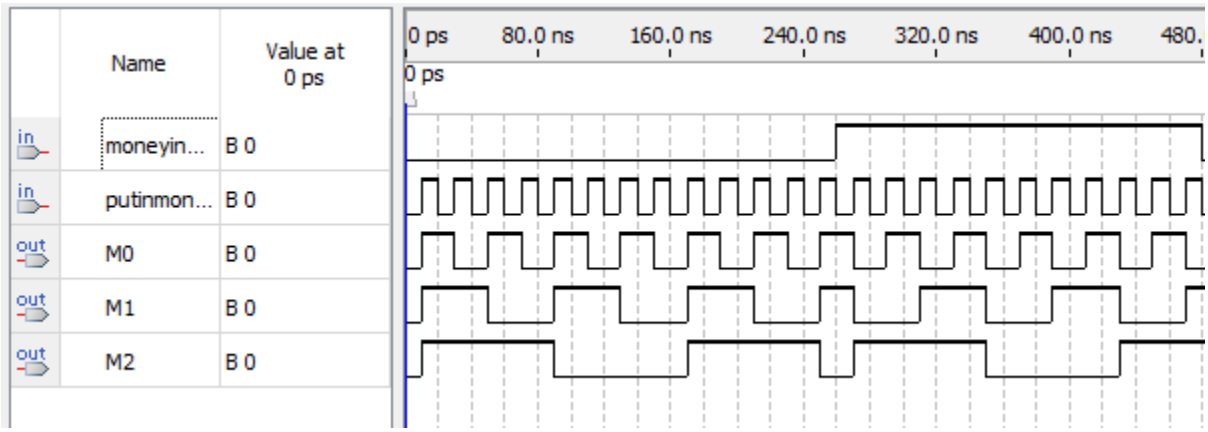
This block was made for the purpose of inputting an amount of money into the vending machine. In order to input the money you would choose a number with the counter and once ready to enact

the transaction you would press the enter button which would allow the signal to pass through the D-flip flops, however they aren't shown here as they were implemented when we combined all of the blocks. A switch would toggle if the counter increases or decreases and a button increments it by one.

Schematic:



Waveform:

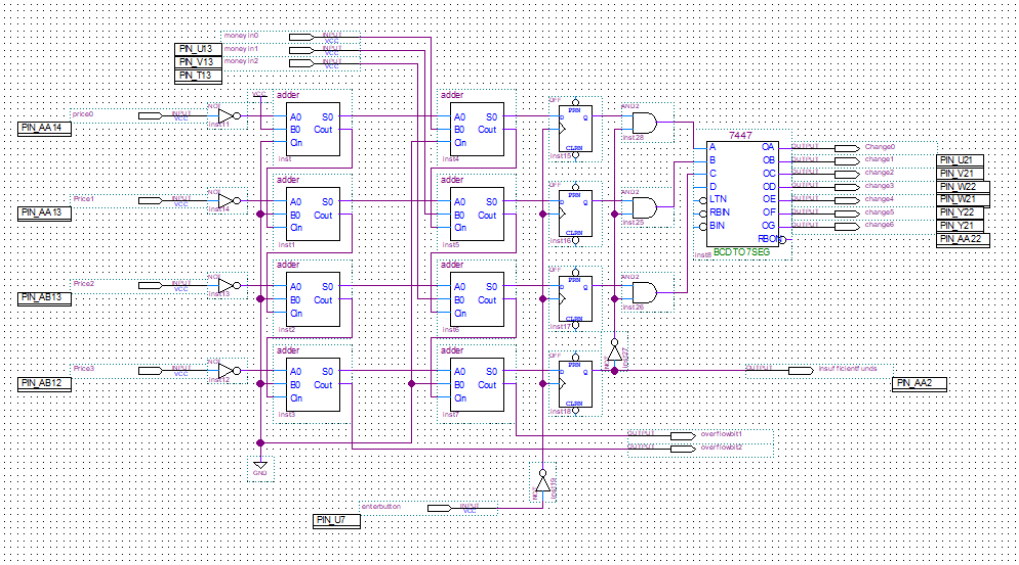


d. Change Calculation Block:

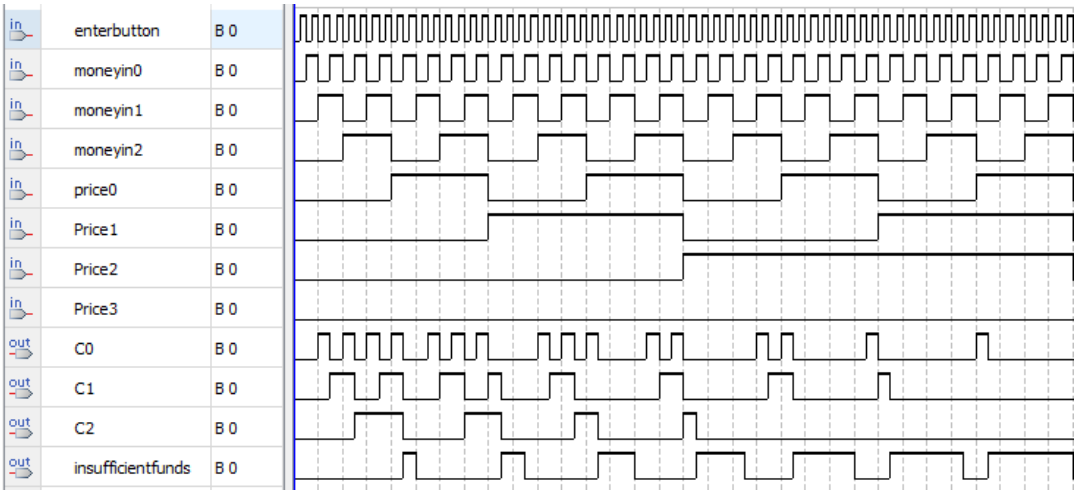
This block is the most important part of the circuit, it is the brains of the machine, without it it wouldn't be possible to easily know if the amount of input money was correct and, if so, whether change would be required or not. it works by receiving a binary number that represents the price of the item selected, it then inverts it with the NOT gates and with the first string of adders, adds

one to create the two's complement of the number in order for subtraction to work. Then the amount of money input and the now negative number are added in the second set of adders, if the leading bit is a one it means that the number is negative and that the amount of money put into the machine wasn't greater than or equal to the price so the insufficient funds output lights up and a NOT gate connected to three AND gates prevents any other output from displaying.

Schematic:



Waveform:



5. Conclusions

During our time working on the final project, we were given the opportunity to put everything we have learned to the test. Since we had to come up with an idea for the project on our own, and we did not get any instructions on how to go about creating our circuit, we had to really think about how we were going to implement each of the functions we had initially drafted. Additionally, we

had to figure out how we were going to divide up the work in order to complete the project in time. In the end, we were able to improve our understanding of logic design.

a. Problems Encountered:

Over the weeks it took us to complete this project, we ran into several problems. One of the difficulties we had was with trying to output different functions onto the 7-segment display. This was a goal that we were asked to try and figure out in one of the previous labs (Lab 11), but we were never able to fully figure out how to display multiple outputs using the same pin assignments in Quartus. In order to get around this problem, we tried using multiplexers and demultiplexers to select which output to be displayed depending on our set of inputs.

Another problem that we encountered throughout this project was that we had difficulty trying to implement a working 3-bit down counter for the items' stocks. With this function implementation, we came across 2 issues:

The first issue that we had with our stock counters was that they kept resetting when the item ran out and an attempt to purchase another was made. This made the counter restock itself, which was not what we intended, unless a master reset button was pressed. In order to fix this, we had to tweak our counters so that the output was input back into it with a NOR3 gate so that the clock wouldn't rise when it wasn't supposed to.

The second issue that we ran into while working with the stock counters was that they were decreasing despite the vending machine not having enough money in it to cover the cost of the selected item. With that complication, the circuit would also stop working as intended until there was an attempt to purchase an item with insufficient money first, and then trying to purchase it with enough money. To get rid of this issue, we had to find a way of preventing the counter from decreasing when there was insufficient money in the system. We did this by making an AND3 that was connected to the clock and the button press so that it would only increment when it was supposed to.