Danny Dutton
5/7/2015
ECE3544

Project 4: Design and Synthesis of a Synchronous Finite State Machine

Objective: The objective of this project is to design a group of interacting finite state machines that mimic the operation of a vending machine. The vending machine can accept change, dispense a product, and return change. Additionally, it can be put into maintenance mode where it can be preloaded with quarters, dimes, and nickels to be used as change for future purchases.

Design: My design uses eight different state machines. The first one is the synchronous deposit machine. This takes in the SW[1:0] input and sets the state of whether to increment quarters, dimes, or nickels when enable (KEY[1]) is high. Also, it controls when the dispense machine should start dispensing products and change.

There is a synchronous dedicated state machine for each quarter, dime, and nickel count. These are controlled by increment, decrement, and reset (KEY[0]) inputs. This keeps track of how many of each coin there is in the machine at all times.

There is a synchronous count machine that controls the amount of money inserted for a transaction. Its inputs are the increment and decrement inputs that the coin counters use and the current mode of the machine, normal or maintenance (SW[3]). Normal mode increases the change count while maintenance mode does not. During the dispense state, the machine will subtract a certain amount of cents from the count depending on how much money was inserted for the transaction.

The dispense machine is triggered by the deposit machine when the total change inserted is greater than or equal to 60 cents. This machine will output the decrement inputs for the count machine and LED selection machine. Using the count machine's state as an input, the dispense machine will determine how much change should be given. Next, it will use the coin counter machine states to determine which denomination to give and how much it can give. For example, if the machine should return 10 cents, it will check if there are enough dimes and returns one if it has any. If not, the machine will check if there are two nickels to give and returns them if it has enough. If not, the machine will give a single nickel if it has any. If it has no dimes or nickels, the machine returns nothing other than the product. Since there can only be up to 80 cents inserted in to the machine, determining which coins to give back is simply one of nine possibilities.

The synchronous LED selection machine uses the decrement outputs of the dispense machine to determine what the sequence of flashing LEDs should be. When we wish to return a dime and then a nickel, the single dime and single nickel output will be enabled and fed to the LED output machine. This machine also keeps track of a clock cycle counter that serves as an input to the LED output machine as well. The cycle counter is reset when there is no LED activity. These sequences are only asserted high for one clock cycle so there is an LED lock input that is checked on each clock cycle to check if the LEDs are currently lit. If the lock is engaged, the LED sequence outputs are kept at their current state. In addition this machine serves as an input to the count machine. It will output a count reset to make sure that if there is any leftover change that can't be returned because the coin counters don't have enough of the correct denomination, the change count will be reset to zero. For example, 65 cents is inserted but there are no nickels in the vending machine to return so the extra 5 cents is kept and the change count reset. This prevents a scenario later on where a product will dispense when less than 60 cents is inserted.

The LED output machine is triggered by any of the LED sequences and the LED lock. This machine will immediately relock the LED lock and then start checking which LED sequence should be used. Each possible sequence will at first light up LED[2] for a total of 50,000,000 cycles or 1 second using a 50MHz clock. Then, if selected, the dime LED[1] will remain lit for 1 second times however many dimes are to be returned. After, if selected, the nickel LED[0] will remain lit for 1 second times however many nickels are to be returned. The LED lock is then unlocked so that the LED output machine won't run again until there is a new sequence of LEDs to flash during a dispensing.

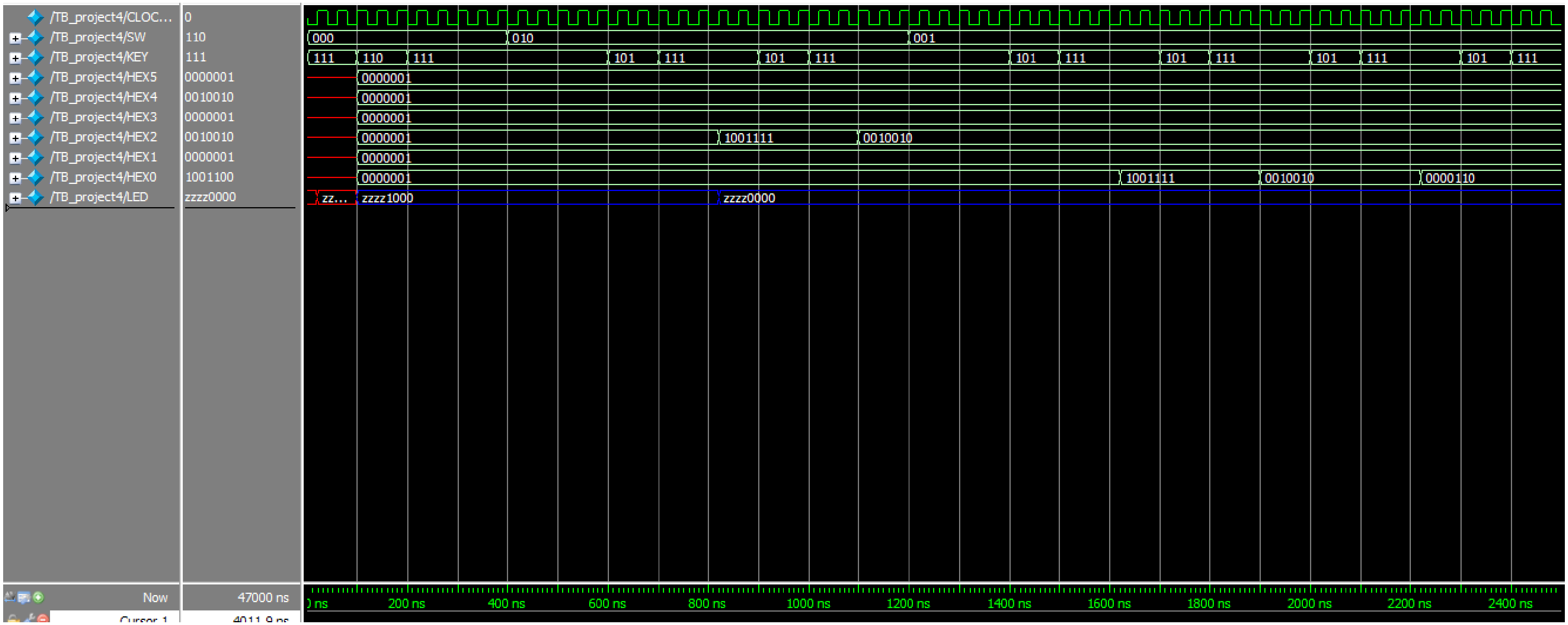
Additionally, there is a continuous assignment to drive LED[3] that is only lit when there are no dimes or nickels in the machine. This is just using the state of the dime and nickel coin counters and seeing if both are zero.

Simulation: On the following pages are waveforms which represent different sections of operation of the vending machine.

Waveform 1 shows maintenance mode while inserting two dimes and then three nickels. It can be observed that the nickel and dime hex displays increment 200 ns after the enable button is pressed.

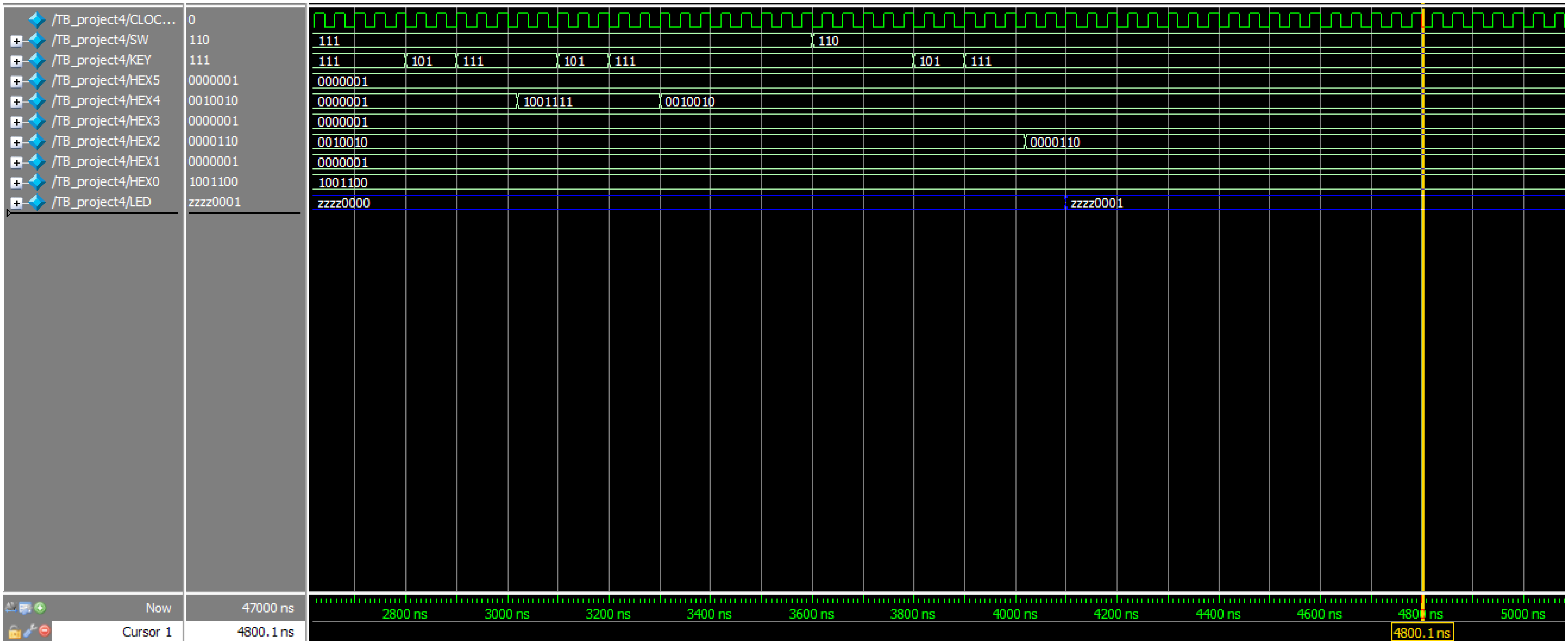
Waveform 2 shows the insertion of two quarters and then a dime. The vend LED comes on to represent the dispensing of the product

1)



```
/TB_project4/CLOC...  0
/TB_project4/SW       110
/TB_project4/KEY      111
/TB_project4/HEX5     0000001
/TB_project4/HEX4     0010010
/TB_project4/HEX3     0000001
/TB_project4/HEX2     0010010
/TB_project4/HEX1     0000001
/TB_project4/HEX0     1001100
/TB_project4/LED      zzzz0000
Now                   47000 ns
Cursor 1              4011.9 ns
```

2)



```
/TB_project4/CLOC...  0
/TB_project4/SW       110
/TB_project4/KEY      111
/TB_project4/HEX5     0000001
/TB_project4/HEX4     0010010
/TB_project4/HEX3     0000001
/TB_project4/HEX2     0000110
/TB_project4/HEX1     0000001
/TB_project4/HEX0     1001100
/TB_project4/LED      zzzz0001

Now       47000 ns
Cursor 1  4800.1 ns
```

Conclusions: This project was one of the most involved pieces of code I have had to write for Digital Design. Looking back at my code, there are places where I could have reduced the number of always blocks. The LED sequence block could have been combined with the dispense block. Additionally, all of the coin counters and change counter could have been combined into a single always block since incrementing these values should happen at the same time. This project really made me consider the number of clock cycles between events as sometimes some state outputs would have already been reset by the time another state machine wished to use it as an input. This has really helped my understanding of FSMs and how they work much better than the smaller machines and examples we have worked on previously in the class and has given me an insight in to how they work that state machines in C cannot.