CDA4203L.001 Chris Frazier
Computer System Design Lab John Gangemi
Spring 2015 Bassam Saed

Lab 4 Report

Introduction

A Finite State Machine is a device that stores the status of something at a given time and can operate on input to change the status and/or cause an action or output to take place for any given change. It is possible to synthesize a Finite State Machine onto an FPGA to operate a mechanical device. We will use the state machine model to create a soft drink vending machine controller that will receive coins and disburse soda and change. The system will have the following inputs and outputs:

Inputs:
- quarter
- dime
- nickel
- soda
- diet

Outputs:
- give_soda
- give_diet
- change

Quarter, dime, nickel represent 25,10, and 5 cents respectively. The state machine will keep track of how much money has been deposited. Soda and diet represent the two different types of soft drinks the user can choose. When at least 45 cents have been deposited the user can select either soda or diet to be disbursed, any change input after 65 cents will be ignored. The give_soda and give_diet outputs will pulse once to represent a drink being disbursed. Change will pulse once for each 5 cent increment of change being given back.

However, on the physical FPGA input to the design will come from five push-buttons and output will be displayed on 5 of the 8 on-board LEDs. Two LEDs represent the soft-drink vended to the user and three LEDs show the amount of change returned to the user in 3-bit binary (each bit equals 5 cents).

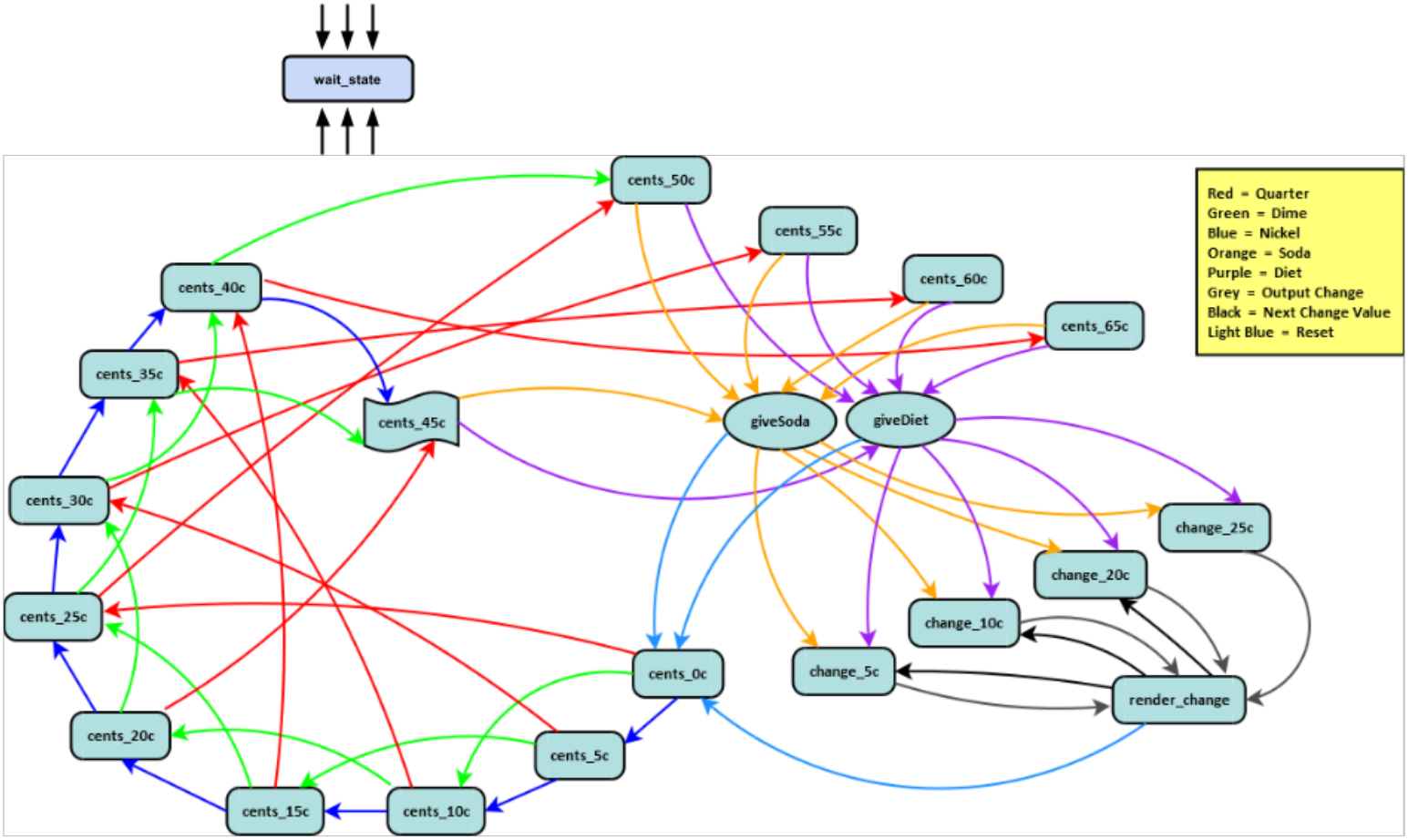
Design

We started our design process by finding every possible combination of coins the machine would be able to hold. Since a nickel is the smallest amount that can be inserted at a time, there is a state for every 5 cent increment up to 45. At 45 cents, the user can choose either diet or soda -- this adds two more states. From here the machine will be reset starting back at 0 cents.

There is also a state for each 5 cent increment from 45 to 65 because overflow could occur if specific combinations of coins are inserted (i.e. four dimes followed by a quarter is 65 cents). These overflow states all lead to the diet or soda states that will disburse whatever the user chooses. However after this state, instead of resetting, the machine will give change in 5 cent increments. We implemented the change model by adding five new states. Four states represent the amount of change still needed to disburse. A fifth state, called “render_change”

will trigger one nickel to be dispensed. The machine will alternate between dispensing a nickel and decrementing the change state until enough change has been output. After the last nickel is dispensed, the machine will reset back to 0 cents. Implementation of a “wait” state prevents registering more than one button press from any given state. A 100MHz clock looks for user input every 10 nanoseconds, this is simply too fast for a human-being to trigger a push-button exactly once. In essence this behavior acts as an un-intentional hold of a push-button.

Finite State Machine Diagram



The FSM (finite state machine) has a total of 22 states

Verilog design of the FSM defines twenty-two 5-bit binary encoded states and declares two 5-bit registers: ‘state’ holds the current state and next state, ‘return_state’ holds the appropriate state in which to return from the “wait” or “render_change” states. An ***always@(*)*** block sets the three FSM outputs based off current state, whereas ***always@(posedge clk)*** block decides the next state to transition too given the current state.

State reduction was made possible using the technique implemented for transitions to the “wait” state. In practice, a second register that can hold a state value makes multiple input and output transitions to/from a single state a possibility.

Example Transition from Cents_0 to Cents_5:

clock cycle 1
current state: cents_0
    input: nickel
        state = wait_state
        return_state = cents_5

clock cycle 2
current state: wait_state
    input: nickel or dime or quarter
        state = wait_state
    input: not (nickel and dime and quarter)
        state = return_state

clock cycle 3
current state: cents_5

Example Transition from Cents_45 to Give_Soda:

clock cycle 1
current state: cents_45
    input: nickel
        state = give_soda
        return_state = cents_0

clock cycle 2
current state: give_soda
    state = return_state

clock cycle 3
current state: cents_0

Example Transition from Cents_55 to Give_Diet:

clock cycle 1
current state: cents_55
    input: nickel
        state = give_diet
        return_state = change_10c

clock cycle 2
current state: give_diet
state = return_state

clock cycle 3
current state: change_10c

Example Transition from Change_5 to Render_Change:
clock cycle 1
current state: change_5
input: nickel
state = render_change
return_state = cents_0

clock cycle 2
current state: render_change
state = return_state

clock cycle 3
current state: cents_0

When debouncing the push-button inputs, the group used a verilog module supplied by the Massachusetts Institute of Technology.

Testing

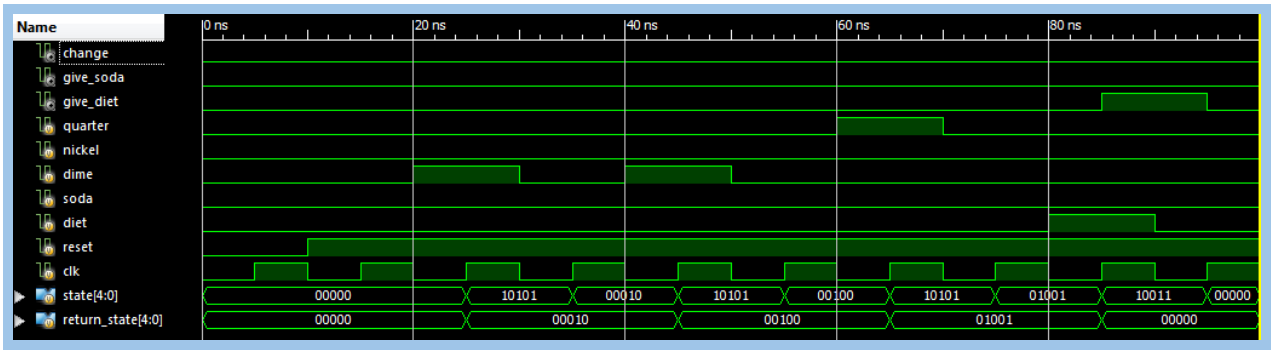
We tested our state machine by choosing three possible input scenarios shown below:

Input	Selection	Expected Output	ChangeState
45 cents (dime, dime, quarter)	Diet	give_diet	no change
50 cents (quarter, quarter)	Soda	give_soda	change_5c
65 cents (dime, dime, dime, dime, quarter)	Soda	give_soda	change_20c

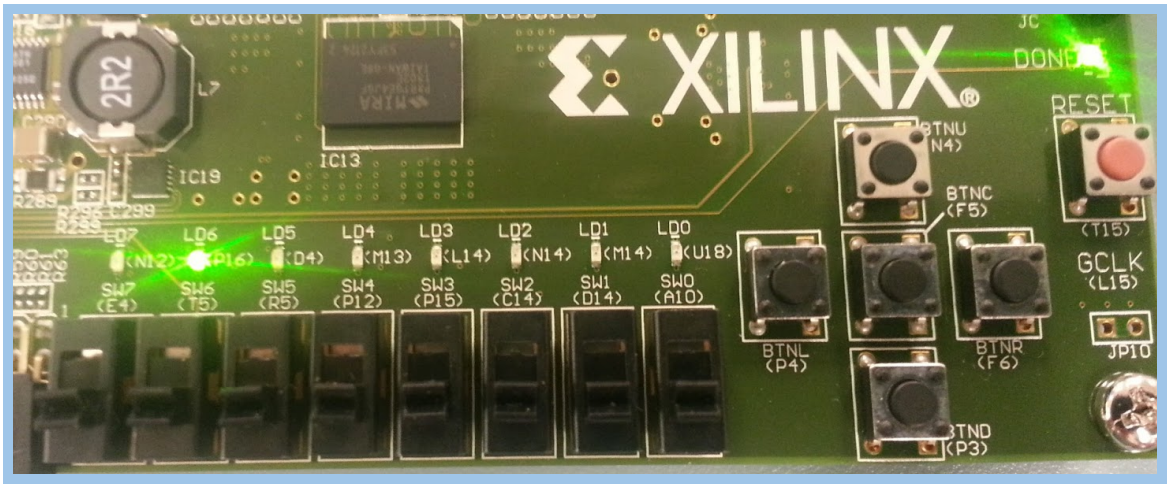
These three sets of inputs simulate all of the possible states and ensure complete working order of the change loop. We also tested the “wait” state by holding some of the inputs high for longer than one clock cycle.

Results

Finite State Machine (No Change)

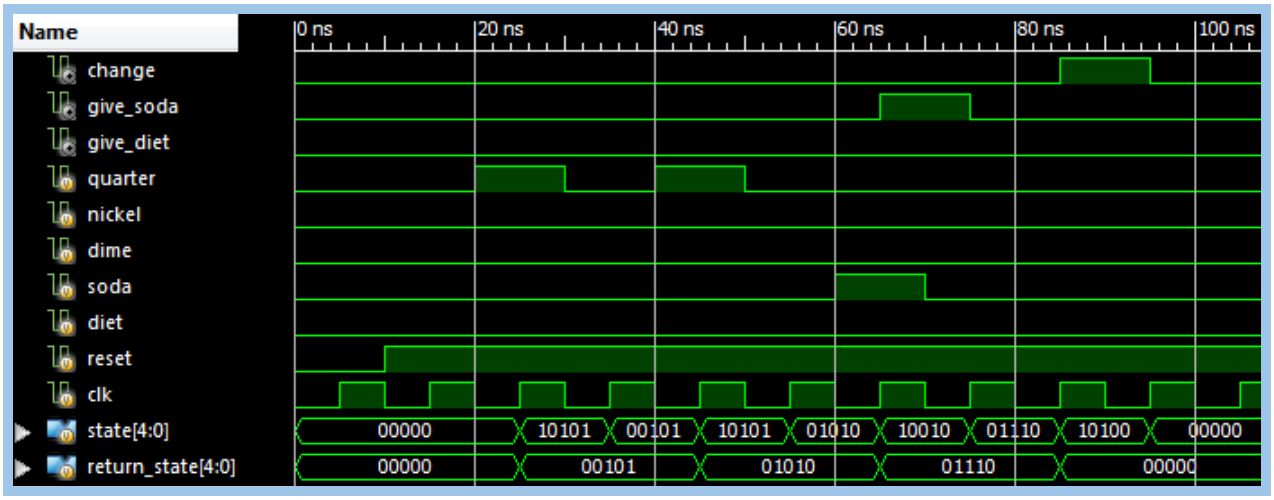


The waveforms show that when 45 cents is received and diet is selected, the give_diet output is set and no change is dispensed.

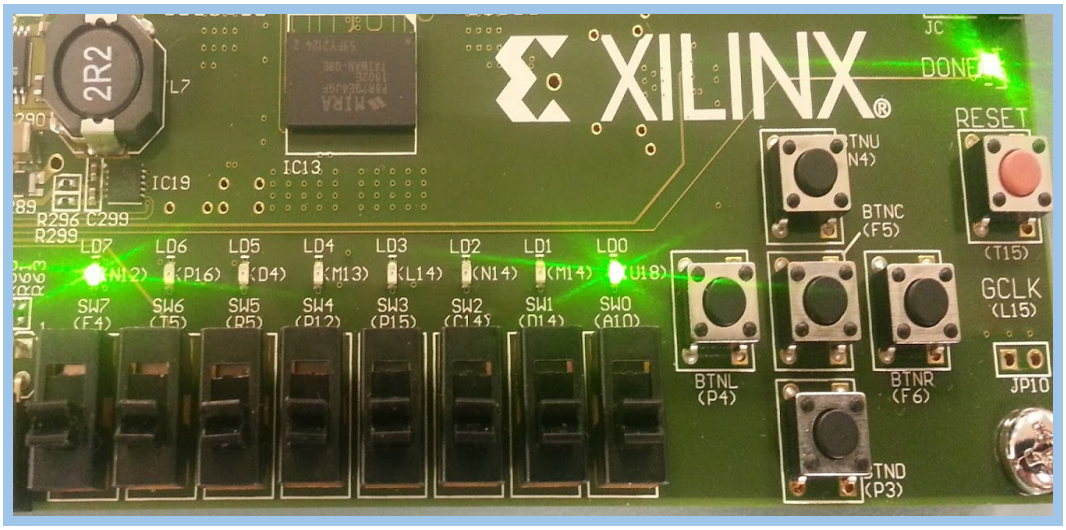


On the FPGA device, only the give_diet LED lights up. This test was successful.

Finite State Machine (5 Cents Change)

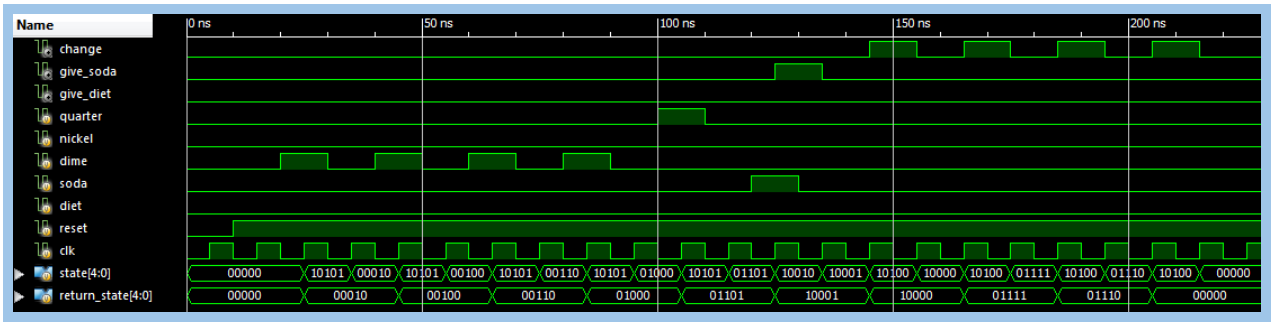


The waveforms show that when 50 cents is received and soda is selected, the give_soda output is set and one nickel is dispensed.

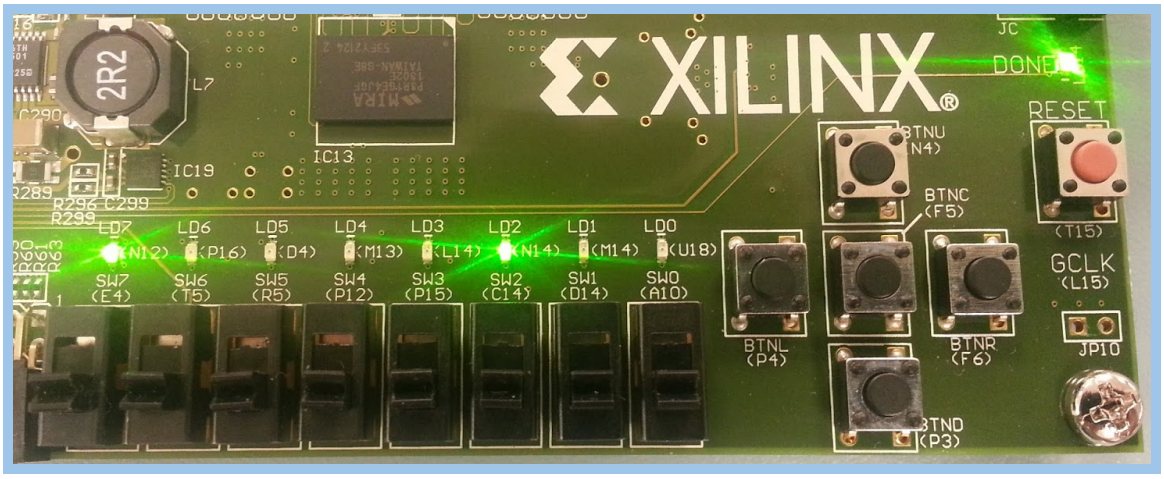


On the FPGA device, the give_soda LED lights up and the “change” LEDs output 001 (one nickel). This test was successful.

Finite State Machine (20 Cents Change)

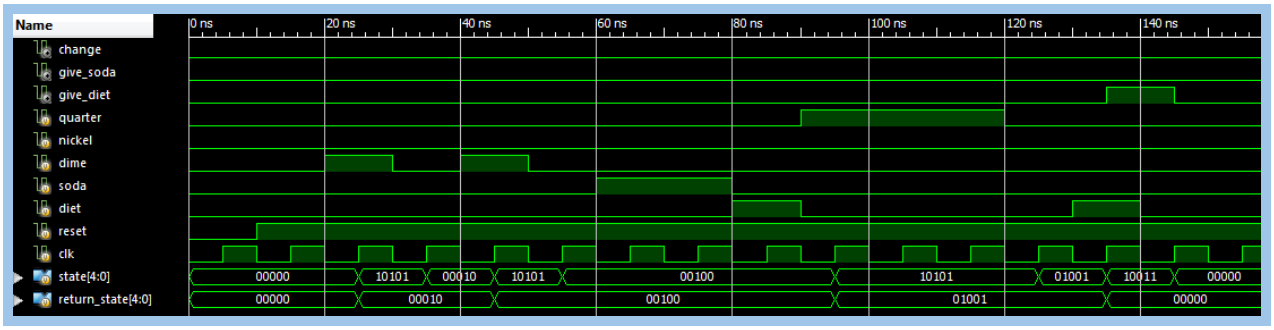


The waveforms show that when 65 cents is received and soda is selected, the give_soda output is set and four nickels are dispensed.



On the FPGA device, the give_soda LED lights up and the “change” LEDs output 100 (four nickels). This test was successful.

Finite State Machine (Testing Inputs)



This test demonstrates how inputs affect state changes when held longer than one clock cycle. This effectively tests the “wait” state and demonstrates what would happen with normal user interaction.

Conclusion

In lab 4 we were able to design a finite state machine for a 45 cent vending machine working with the Moore design we had about 22 states.The only issue encountered dealt with “one-hot” encoding of the states in verilog. During synthesis of the top module, a number of warnings pertaining to latch-up occured and were fixed once changing the state encoding to binary. According to the teaching assistants this type of warning can occur when creating a FSM that uses “one-hot” encoding and has not been designed for optimal efficiency. This lab was the perfect introduction to basic finite state machine structure in verilog.

*All work was distributed evenly amongst all team-members.
*debounce.v is the original work of M.I.T.
http://web.mit.edu/6.111/www/f2005/code/jtag2mem_6111/debounce.v.html