**CMPE-260 Laboratory Exercise 04**

**Vending Machine**

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students; however, other than code provided by the instructor for this exercise, all code was developed by me.

John Judge

John Judge

Performed 3/16/16

Submitted 4/14/16

Lab Section 04

Instructor: Dr. Lou Beato

TAs: Nick Barlow

Sergio Martin

Lecture Section 01

Professor: Dr. Reza Azarderakhsh

**Table of Contents**

1. Abstract …………………………………………………………………………...1
2. Design Methodology………………………………………………………………1
3. Results……………………………………………………………………………..3
4. Conclusion………………………………………………………………………...5
5. Appendix…………………………………………………………………………..5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Abstract:**

In this exercise multiple VHDL files were created and added to an existing project in order to design, simulate, and synthesize a vending machine. In order to complete the design a Coin Controller, a Seven-Segment Decoder and a Vending Machine Controller needed to be created. Once these units were designed and implemented into the project the project was simulated using ModelSim. After analyzing the waveforms generated in the simulation and verification that the simulation produced the correct results the project was then synthesized. The design was routed to a NEXYS 3 FPGA and the design was tested. After several key test cases were tested the design was verified to be correct. The results produced from the simulation and the hardware implementation confirmed a successful exercise.

**Design Methodology:**

*The Coin Controller*

The Coin Controller was the first design implemented during this exercise. The Coin controller handled the operations that the vending machine would do when coins were inserted, sodas were selected, and sodas were dispensed. In order to handle these operations the Coin Controller needed to be a Moore State Machine. The advantage of using a Moore State Machine instead of an ASM or Mealy State Machine was that in a Moore Machine the state changes on the rising clock edge and the output of the machine is dependent on the current state. The signals of the Coin Controller were a clock (clk), reset (rst), quarter received (Qp), dime received (Dp), nickel received (Np), the 4-bit soda price code (soda\_price), and a soda request signal (soda\_req). The outputs were a amount error signal (amt\_err) that represented an inadequate amount of money deposited, a drop soda signal (drop\_soda) and a 10-bit signal representing the amount of money deposited (amt\_dep).

In order to implement the design in VHDL a series of case statements were used to describe the state machine. Each case statement represented what operation would take place in each state and the condition necessary to move to the next state. The Coin Controller had a total of eleven states. A graphical representation of the controller can be seen in Figure (1.0). As seen in Figure (1.0), the change between states was triggered by a change in input and the outputs of the controller changed depending on the state. The state machine developed essentially had two different operations; handling money and dispensing soda.

Whenever a coin was inserted and a pulse representing a coin was triggered the state moved from Idle to a state representing the addition of that coins nomination to the total amount deposited. In these states (Add5, Add10, and Add25) the coin’s nomination was added to the temporary signal representing the amount collected. From these states the machine would immediately move to the CoinHolder state. When in the CoinHolder state, the temporary signal for the amount collected was assigned to the total amount deposited. The machine would then immediately move to the Hold state. The machine would stay in the hold state until the pulses representing coins being deposited would change to low. The purpose of this was to eliminate the coin controller from adding coins for each rising clock edge the machine experienced when coin signal was being read. Durring a post-route simulation this error occurred and the hold states were added. The initial design of this state machine did not include the hold states. In any situation in which the signal for the coins went to low the machine would move back to the Idle state and wait for the next stimulus.

In order to handle a request to vend a soda the Coin Controller used the states on the right-hand-side of the graph in Figure (1.0). When a request for a soda was triggered the controller would move from the Idle state to the CheckPrice state. In this state the total amount deposited was compared to the signal representing the price code of the desired soda. The table representing the soda price codes can be seen in Table (1.0). If the amount of money deposited was greater than or equal to the price of the soda then the machine would move to the drop soda state. In the drop soda state the amount of money deposited would decrement accordingly to the sale of the desired soda. From here the machine would move to the holddropsoda state where the machine would stay until the soda request signal went low. The purpose of the hold state was to eliminate machine from purchasing a soda on each clock edge while the soda request signal was high. If the soda price had been higher than the amount deposited then the controller would move to the NoSoda state. In this state the machine would trigger the amount error signal and then move to the holdnosoda state until the soda request signal would go low and then the machine would move back to the Idle state.

|  |  |
| --- | --- |
| Price | Code |
| $0.55 | 0000 |
| $0.85 | 0001 |
| $0.95 | 0010 |
| $1.25 | 0011 |
| $1.35 | 0100 |
| $1.50 | 0101 |
| $2.25 | 0110 |
| $2.50 | 0111 |
| $3.00 | 1000 |
| Reserved | 1001-1111 |

Table 1: The Price Codes

The outputs of the coin controller: the amount deposited signal, the drop soda signal, and the amount error signal were assigned following the case statements representing the states. No matter the state, the amount deposited signal was updated whenever the temporary signal for the amount deposited changed. The temporary signal changed when in the Add5, Add10, Add25, and DropSoda states. The drop soda signal was high only when in the holdsoda state. The amount error signal was high only when in the holdnosoda state.

*The Seven-Segment Decoder*

The seven-segment decoder was used to decode a 12-bit vector representing the binary representation of the amount deposited. A block diagram representation of the Seven-Segment Decoder can be seen in Figure (1.1). The output of the decoder was into a 21-bit vector that represented which LED’s of a 3 number seven-segment display would be lit. This was done using three case statements. Using the four most significant bits of the input the decoder would assign which LED’s needed to be lit to represent how many dollars were deposited. This was done using the standard seven-segment cryptography in which LED’s are labeled A-G in the standard way and the binary representation of the LED’s being lit are active low. Using case statements regarding the next four most significant bits and the four least significant bits of the input, which represented the tenth’s place and the hundredth’s place of the amount deposited respectively, their decoded 7-bit signals were derived. To create the 21-bit output the signals were combined with the dollars being the seven most significant bits, the tenth’s place being the next seven most significant bits, and the hundredth’s place being represented by the seven least significant bits.

*The Vending Machine Controller*

The wrapper used to combine the Coin Controller and the Seven-Segment Display was called the Vending Machine Controller. This unit was then connected to the User Interaction wrapper and both of these were bundled within the Vending Machine top level wrapper. Its inputs were the clock (clk), reset (rst), quarter received (Qp), dime received (Dp), nickel received (Np), the 4-bit soda price code (soda\_price), and a soda request signal (soda\_req). The outputs were a amount error signal (amt\_err) that represented an inadequate amount of money deposited, a drop soda signal (drop\_soda) and the 7-bit anode vectors for each of the displays (hund\_disp\_n, tens\_disp\_n, and ones\_disp\_n).

Within the wrapper the correct signals were mapped to the correct inputs and outputs of the Coin Controller and the Seven-Segment Display. Using the BCD package the amount deposited signal from the Coin Controller could be converted to the 12-bit vector necessary for the Seven-Segment display. Once all the connections between the inputs and outputs of the sub-blocks were assigned accordingly the entire Vending Machine module could be tested.

**Results:**

*The Design of the Test Bench*

In order to test the code developed during this exercise a test bench was created to test the components through the Vending Machine Controller. In order to be correctly implemented the clock period was set to 100 ns. This is a standard clock period, however, when the design would be implemented in hardware the testing would occur over billions of clock cycles even if the testing occurred over a few seconds. Since ModelSim operates in a “draw every edge” approach the simulation resulted in heavy lagging even when simulation for several microseconds. As a result the simulations were kept to 200 ms and the inputs, whether it be a request or a coin pulse, changed approximately every 5 ms. Due to the constraint of 200 ms, the test was limited to resetting the machine, inserting all coins, requesting a soda with not enough money deposited, and requesting a soda with the correct amount of money.

*Results of the Behavioral Test Bench*

To begin the testing a behavioral simulation occurred. As seen in Figure (1.2) a waveform was generated by ModelSim to describe the result of the test bench. The signals and temporary signals representing the amount deposited, the price of the request soda, and the result of the purchase were changed to decimal for ease of reading.

It was during this simulation that the need for the hold states was discovered. Originally the design of the Coin Controller did not include the hold states. Once an intial post-route simulation occurred it was discovered the machine was iterating through the loops of the state machine for each clock cycle. In events such as a quarter being inserted over 5 ms resulted in millions of quarters being inserted. The design was changed to include the hold states and then the next post-route simulation occurred and then the results were analyzed.

As seen in Figure (1.2) the program is initially reset but the edge of the reset cannot be seen since it changes at the beginning of the simulation. A quarter pulse is then triggered representing a quarter being deposited. The machine then moves to the add25 state and eventually returns to idle. Then a soda is requested. The operations regarding the amount deposited and the temporary signals used conjunctly updated accordingly. As a result the machine changed to the CheckPriceState, the NoSoda, and Idle state sequentially. During this operation the amount error signal was triggered, as it should. Next the simulation adds the exact amount of coins necessary using quarters, dimes, and nickels. The machine did change state as it should during this process. Finally, the soda request was triggered and the machine responded by correctly outputting a high drop soda signal and returning to the Idle state. This concluded the behavioral test since the Vending Machine responded correctly to the test cases.

*Results of the Post-Route Simulation*

Following the Behavioral testing a Post-Route simulation took place in order to simulate the delays to be expected in the hardware implantation. Durring this simulation ModelSim experienced heavy lagging due to drawing the clock edge. Since the simulation occurred in 200 billion clock periods the delays could not be accurately seen when viewing the entire simulation. As a result the change between the soda request and the drop soda signal was zoomed into and the image can be viewed in Figure (1.3). As seen in the image, the soda request is triggered by the test bench and after the clock periods used to move between the Idle state, CheckPrice, and DropSoda there is a delay of approximately 5 ns between when the drop soda signal is theoretical triggered and when the machine, with the post-route delay, triggers the drop soda signal.

*FPGA Implementation*

Following the post route simulation the design was routed to the NEXYS 3 FPGA and was tested using the board’s switches to represent the reset and soda price code. The buttons were used for the coin pulse signals and the request soda signal. The LEDs were used to show when a drop soda, amount error, and the coin pulse signals were triggered. Three of the four possible seven-segmented displays were used to show the amount deposited. During this testing multiple tests occurred using different soda requests with varying amount of change deposited. In all of these tests the board responded appropriately. The resulting lit and unlit LEDs matched the expected results and the seven-segmented display updated accordingly to the increasing and decreasing amount of money deposited. The design could then be verified to be functioning and the design summary was then consulted.

*Design Summary*

The Design Summary tab of the project was then consulted to check the occupied number of slices, flip flops, and LUTs. A capture of the design summary can be seen in Figure (1.4). Based on the Design summary the project used a total of 201 LUTs and Flip Flops as well as 90 slices

**Conclusion:**

There were some lessons to take from doing the exercise. The most major of which was the implementation use of Moore State Machines within a design. Although some minor bugs due to timing errors occurred following synthesis the problems were eventually rectified. Once rectified the design was properly synthesized and the cause of error, not using hold states, was fully understood. When using a Moore State Machine the state changes on the clock edge and is dependent on a signal. In situations, such as the one encountered in this lab, a hold state is necessary otherwise the machine could cycle through many iterations of the same command while a signal remains steady through many clock edges. Overall, the exercise was conducted successfully, with the Vending Machine resulting in the expected outputs for the given inputs. In the future this exercise can be used as an example of how to correctly implement a state machine in VHDL and hardware.

**Appendix:**

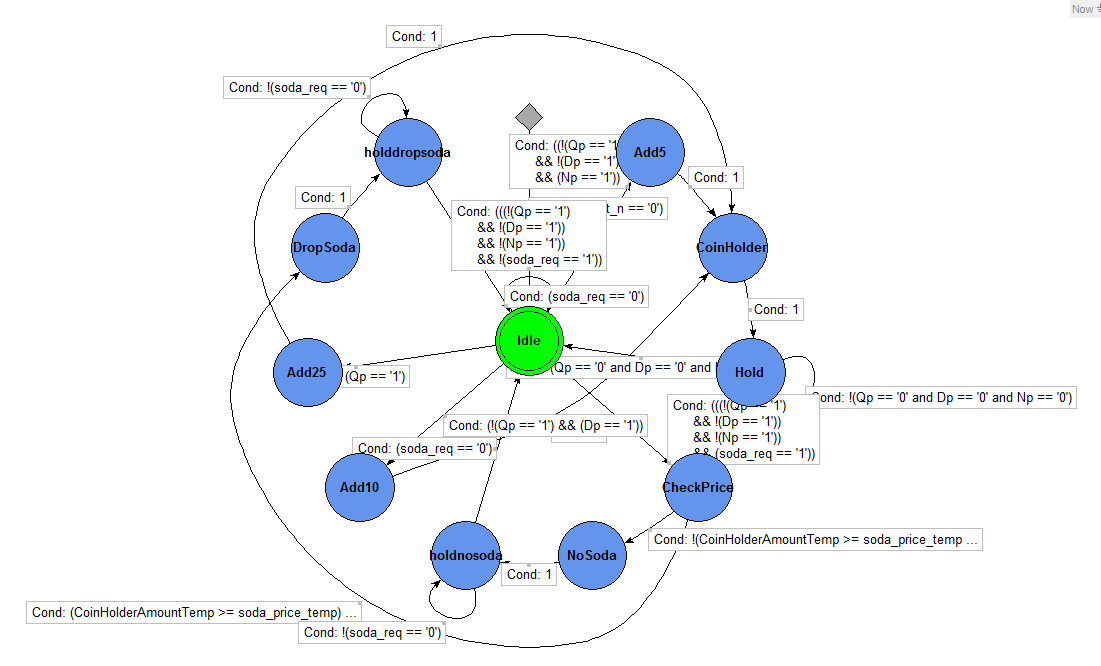


Figure (1.0): The Coin Controller

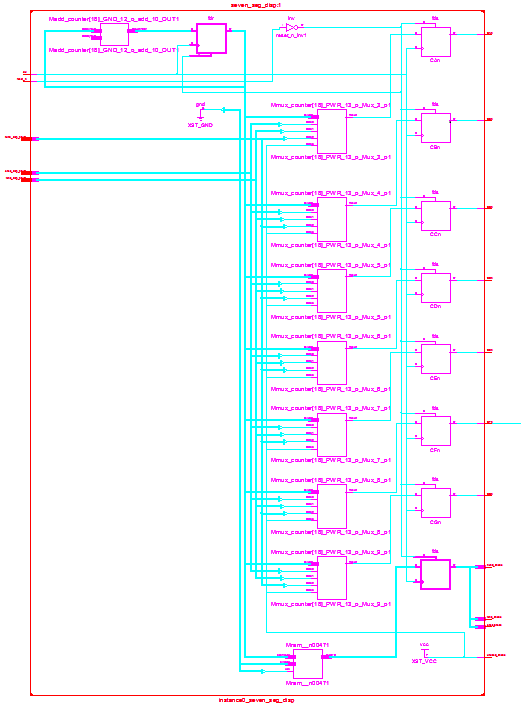


Figure (1.1): The Seven-Segment Decoder

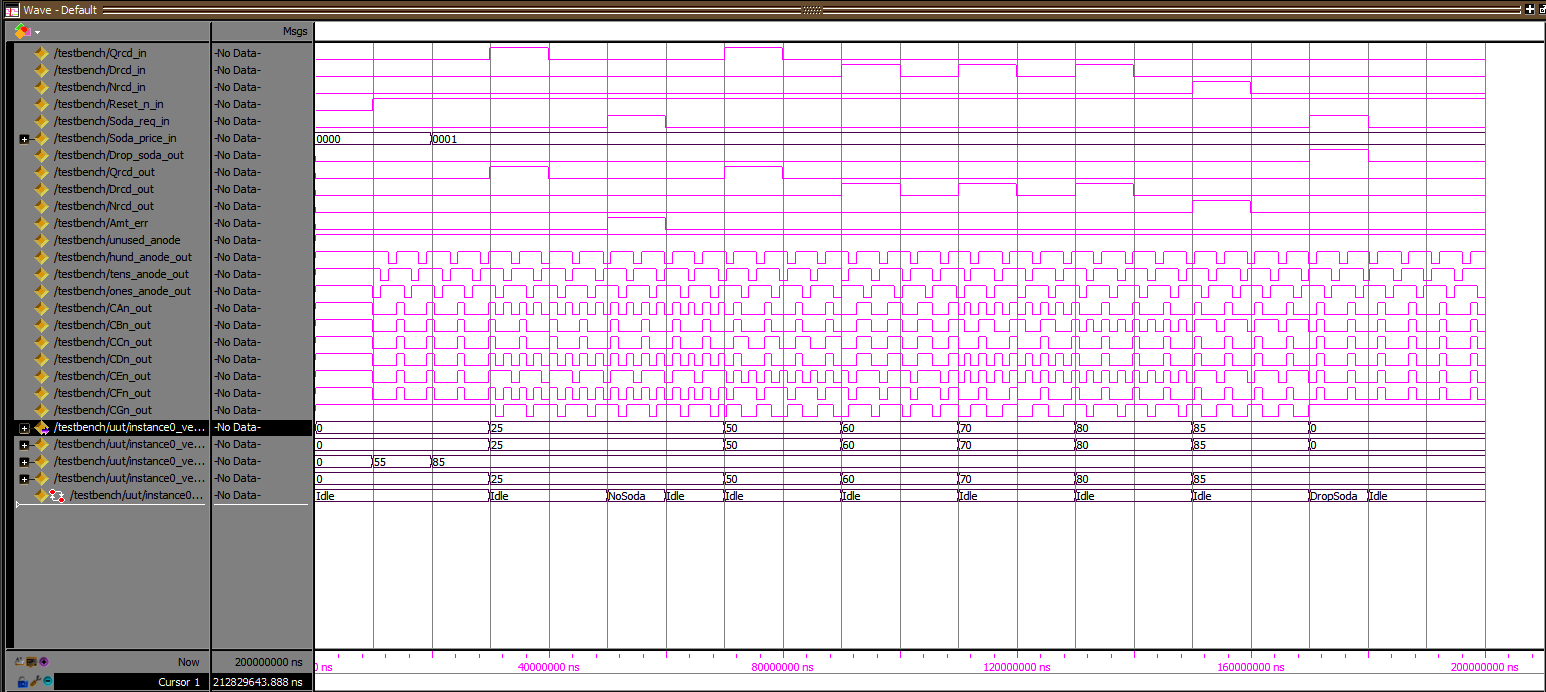
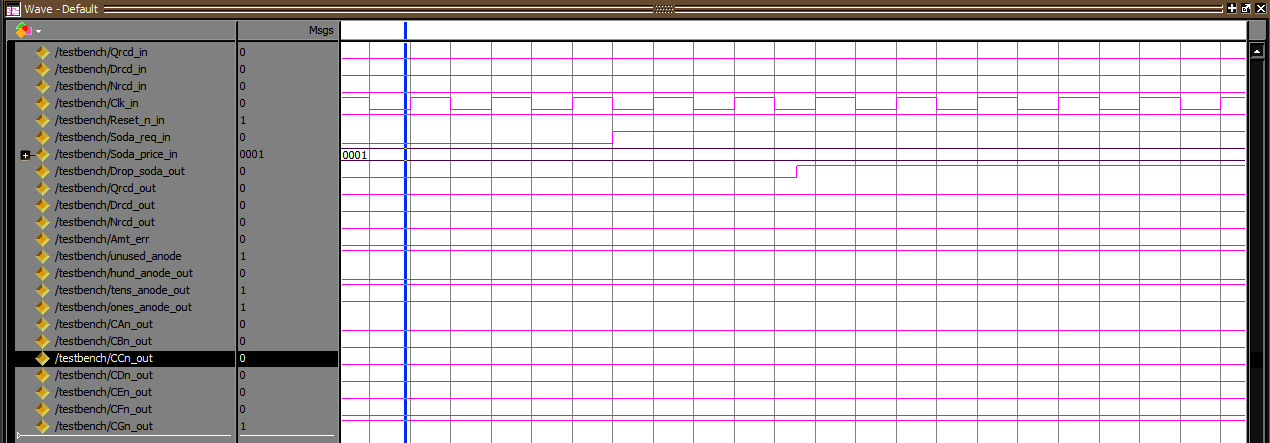


Figure (1.2): The Behavioral Simulation

Figure (1.3): The Post-Route Simulation

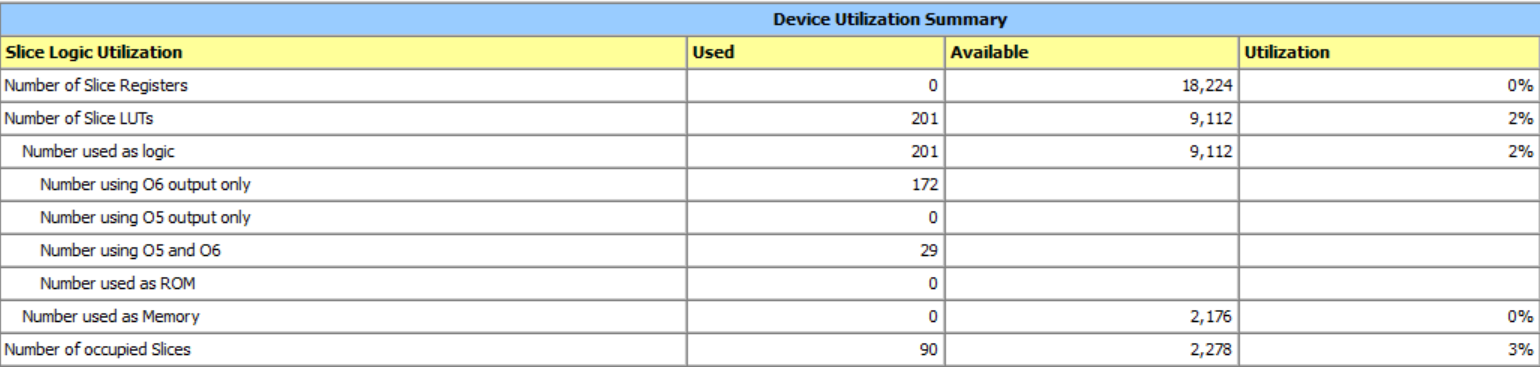
**

Figure (1.4): The Design Summary

