Final Project Report

Nicholas Bart
Samarah Uriarte
Isaac Yamnitsky
Jordan Remar

EC311
12/11/19

Vending Machine

Overview

Our goal for the project was to design and program a vending machine. Using combinational and sequential logic, we wrote and implemented code onto an FPGA, which served as the machine. The system would first ask the user to input the code of the snack that they want. Next, it would ask for payment. Finally, it would count the deposited money and produce an output dependent on what was inserted.

We each had different jobs to do as part of this project.

- **Nicholas Bart:**
    - Helped with the design and the state machine diagram.
    - Wrote code for the Snack_Select module which iterates through the possible snacks.
    - Created the majority of the powerpoint, including every slide that wasn't a diagram or a photo.
    - Wrote the overview and the conclusion for the report.
- **Samarah Uriarte:**
    - Wrote the code for the top-level module, the money deposit module (containing a debouncer, price selector, money intake submodule, and change calculator).
    - Aided in rewriting the code for the display module and developing the constraint file.

- Contributed to the creation and organization of the state machine diagram.
- Aided with the approach and design on a general and logic-based level.
- Wrote the “System Functionality” subsection to the report.

- **Isaac Yamnitsky:**
  - Designed the top, snack_select, money_deposit, and display_driver modules.
  - Implemented the display_driver and its sub-modules.
  - Implemented enable lines to the snack_select module.
  - Designed block diagrams.
  - Wrote the “Schematic and Block Diagrams” and “Explanation of Design Choices” section of the report.
- **Jordan Remar:**
  - Created the constraints file used for implementation.
  - Helped create the state machine and a diagram explaining the system inputs.
  - Illustrated state machine and block diagrams
  - Aided in conceptually designing the top level module.

Schematics and Block Diagrams

Figure 1 shows a top level view of the vending machine. The top module has inputs inc, reset, clk, sm, and mc. Sm stands for the “snack-money” switch, which holds the system in the snack selection state for sm = 0, and the money deposit state for sm = 1. Likewise, the mc switch holds the system in the money deposit phase for mc = 0, and the pay/receive change phase for

mc = 1. The top module outputs a 7-bit signal for the 7-segment display, an 8-bit digit_select signal, and a 1-bit signal for the decimal point required in our design.

The snack_select module consists of a debouncer, and a 3-bit counter whose output represents the user's snack selection. It is important to note that the 3-bit counter also contains an enable input which is fed the inverse signal of sm. This is because we only want the user to be able to change their snack selection while in the snack selection state (sm = 0).

The money_deposit module consists of a debouncer, a 4-bit counter to keep track of the money a user deposits (in 25 cent increments), a price_finder module to calculate the price a user needs to pay for their selected snack, and then a calculate_change module which subtracts the money_deposited from the calculated price.

The display_driver module consists of a left_text and right_text module, used to calculate the seven segment output for the left and right displays, respectively. It also contains a 3-bit counter that is fed a 1 kHz clock frequency and outputs a select line for two 8-bit muxes. One mux is for the display signals and the other is synchronized for an 8-bit digit_select signal.

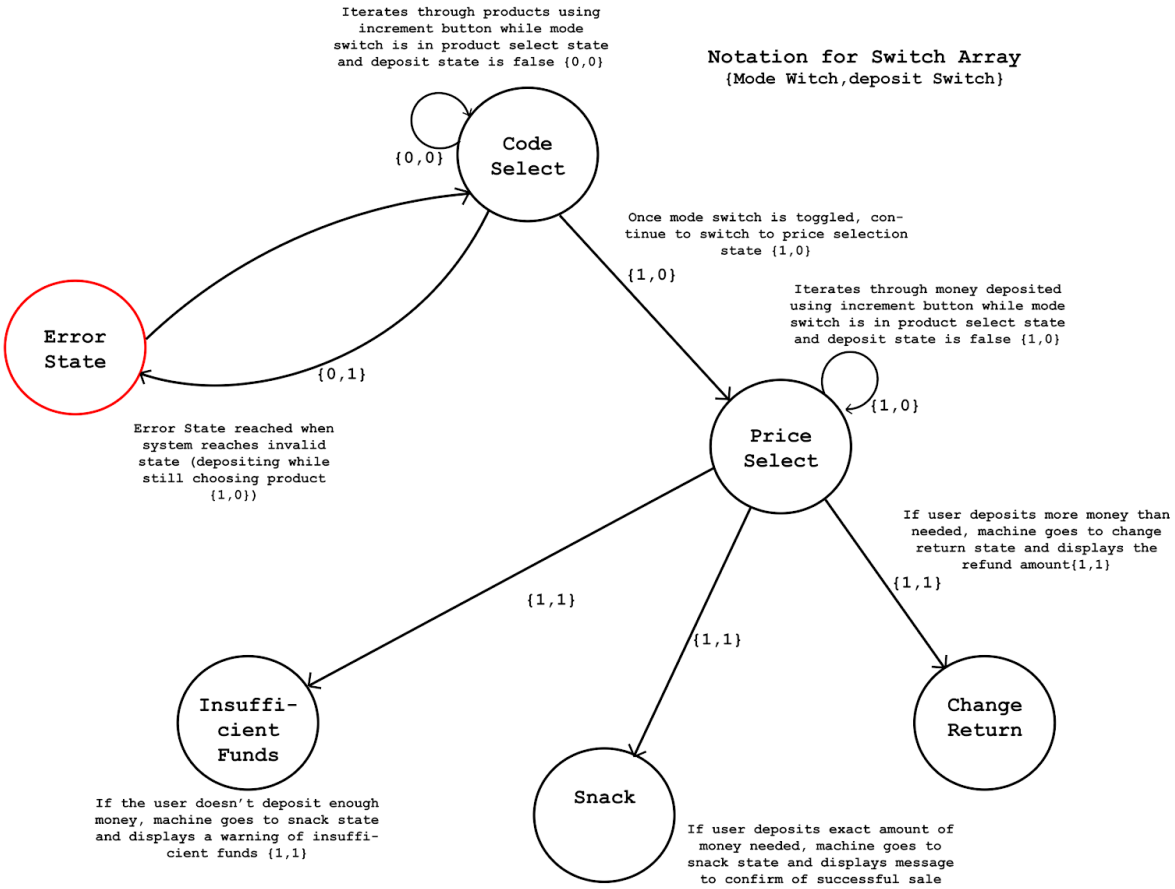
Explanation of Design Choices

The challenge in designing the vending machine stems from the need to transition between different states while minimizing the number of states which require error handling. To accomplish this, enables were implemented to the internal modules so that, for instance, your snack selection would not change while you were depositing money.

We chose to use a 3-bit signal for the snack selection code because it has eight possible values which corresponds nicely to the eight possible snack codes (see Figure 3). The money

deposited and change values are 4 bits large, but the user can only deposit $3.75 maximum. This is in order to avoid a number of error states. If the user does not deposit sufficient funds, then the change given back will be $6.00 which is not possible given the current constraints. This acts as an extra signal to inform the display_driver module that no change should be returned and no snacks should be vended. We were able to remove an input and an output using this design, simplifying the overall vending machine.

System Functionality



This diagram shows the system's state machine in a simplified manner. The inputs displayed represented the switch values, but there are also increment and reset buttons within the first two states (they are generalized for simplicity).

The user interacts with the FPGA vending machine by pressing buttons and pushing switches and the system will produce dynamic outputs that depend on the inputs it is fed by the user through these methods. Additionally, it must be assumed that the user has access to the menu that specifies the snack types and code corresponding to each snack.

Once the FPGA is turned on and all switches are off, it will automatically show the word "code" on the left display using four 7-segment LEDs and the code "A1" on the right display using two 7-segment LEDs.

The user is able to iterate through the snack codes by pressing the code select button on the FPGA, starting from A1 and continuing to C2 to then loop around back to A1. To indicate that the user has chosen the specified snack, they must then push the mode select switch to give it the value of 1'b1, which will then tell the system that the code value must stay constant and equal to the code that is currently displayed. In the vending machine's code, the mode select switch is on code select mode when it is equal to 1'b0 and on price select mode when it is equal to 1'b1.

In price select mode, the user is then prompted to enter some money by the left display, which will read "pay" with three 7-segment LEDs. The initial value the money deposit will exhibit is $0.00, which can then be incremented by 25 cents every time the user presses the button that was used to iterate through the snack codes. This is efficient because rather than

using two buttons for incrementing each counter, one is used, which saves area and reduces the overall cost of implementation (outside of the FPGA). Additionally, the value of 25 cents was used because real-world vending machines usually accept quarters as the smallest form of currency that a user can insert to obtain a snack, so our design attempted to mimic this convention. The maximum amount we allowed the user to enter is $3.75, and this amount is based on the maximum the money deposit counter is able to count to. It is a 4-bit counter, so it can count to 15, which adds up to our money input capacity.

Finally, in order for the user to confirm the money amount they entered, they must push the money select switch on to set it to 1'b1. This switch will move the system into one of three states, depending on how the deposited money relates to the price of the selected snack:

1. If the inputted amount is less than the price, the abbreviation "insf," which indicates the amount is insufficient, will be shown on the left display using four 7-segment LEDs and the right display will output four zero's.

2. If the inputted amount is equal to the price, the left display will be completely off and the right display will show the abbreviation "snac" using four 7-segment LEDs, which indicates the snack is being vended and there is no change to be given back to the user.

3. If the inputted amount is greater than the price, the left display will exhibit the abbreviation "chng" also using four 7-segment LEDs, which indicates that the user entered too much money, and outputs the exact amount of change the user is owed on the right display using three 7-segment LEDs.

At any time during their interaction with the system, the user is able to press the global reset button to reset the machine back into its initial state, with a default code of A1 and a corresponding default price of $1.00.

There are not many possibilities for the user to enter an unexpected input; however, if the user were to push the switch relating to money select on without pushing the snack select switch as well or already having it previously on, an error message will appear on the left display, solely. Specifically, “error” will be exhibited through four 7-segment LEDs to indicate an anomalous input, which the user can then fix by simply turning off the money select switch or by pushing on the snack select switch.

Conclusion

Overall, this project was a success. We successfully implemented a vending machine using the given FPGA. Throughout this project, we learned several things. Probably the most important one, and the one that almost no one thinks about, is making sure to find group members whose schedules at least somewhat lineup. We did not do this, unfortunately, which left us to meetings in the late evenings because that is simply when everyone was free. Secondly, you need to set up a group chat immediately and plan your time and meetings carefully. Thirdly, you should thoroughly plan every aspect of your code before writing it; this includes creating state machines and block diagrams.

In the future, there are a few things we could do better. We could have been more organized from the start. We would also make a few changes to the code, such as using only one debouncer instead of two and implementing it on the top level module. Additionally, we would

change the design so it does not depend on the user's possession of the menu by adding more information about the snack, such as the name and price.