WESTERN NEW ENGLAND UNIVERSITY | WNE

Department of Electrical and Computer Engineering

# CPE 462/562 VHDL: Simulation & Synthesis

## Final Project: Vending Machine Design and Implementation

Katelyn Charbonneau
Email: kc325844@wne.edu
Date: 12/8/2017

| **Tasks** | **Grades** |
|---|---|
| Select product, choose to buy the product, and display product name and price (10') | |
| Cancel transaction, display returned amount, and show idle state (20') | |
| Insert coins, display the left amount for purchase on SSD, and reject invalid coins (40') | |
| Deliver product, and display change on SSD (20') | |
| Report (10') | |
| Total (100') | |

# 1 Objective

The objectives of the project are to design and implement a simple vending machine; be able to write VHDL code to design said system using a state machine; and implement the system on the Altera DE-115 FPGA.

# 2 Design Procedure

The first step in designing a system using state machines is to draw a state diagram, which is shown in Figure 1.

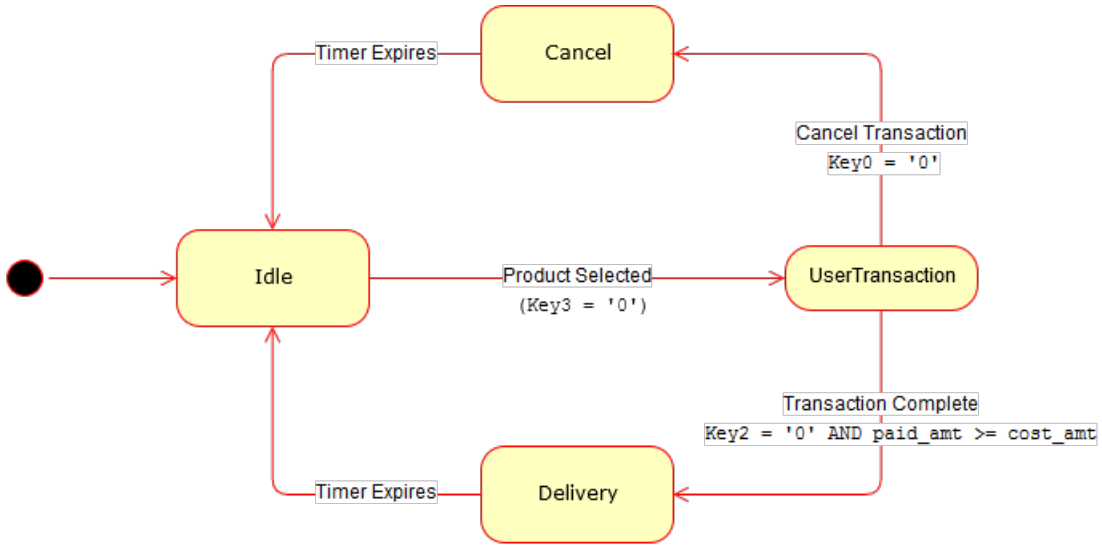


Figure 1: Vending machine state diagram

Development of the VHDL code went state-by-state for simplicity. After the behavior of each state was defined, the code for that state was programmed onto the board and tested.

The first state that was designed was the Idle state. For this state, a record type was declared to contain product information for each of the products. The information included was the product code (two bits standard logic), product abbreviation (character), and price. While records were not covered in class, previous familiarity with similar techniques such as dictionaries from the Python language made using one simple. Using the record allowed the Idle state to be written in much more concise, readable code.

The idle state displays dashes (-) on all of the seven-segment displays (SSDs) used in the project (Hex 5, 2, 1, and 0). When a user presses the product choose button (Key 3), the product abbreviation and price will take on the value of whichever

product was selected with the switches (Switch 17 and 16), then transitions to the UserTransaction state.

The UserTransaction state is the next that was developed. It also contains most of the logic that dictates the behavior of the system. For this system, a number of signals had to be declared to handle the operations required. One signal is simply the price of the selected product. Because it could be any of the four products offered, it had to be able to be changed; although it is not altered within this state (it is set in the Idle state). The second signal is the amount of money that is the customer has inserted. The third is a catch-all signal that is ultimately passed to the output ports of the system; this signal takes on a value of some function of the previous two signals and itself, depending. When the user inserts a coin, the paid amount increases by that coin's value, and the display on the SSDs decreases by that coin's value. When the paid amount is greater or equal to the cost amount, the state transition to Delivery occurs. If the user presses the cancel button (Key 0), the state transition to the Cancel state occurs.

The Cancel state is simple: it displays on the SSDs the amount of money to be returned. This value is equal to the value of the paid amount signal. After 5 seconds, the system will transition to the Idle state. The Delivery state is also simple: it displays on the SSDs the amount of change to be returned. This value is equal to the value of the paid amount minus cost amount. The delivery LED will also blink 5 times, after which the system will transition to the Idle state.

For simplicity, the values of the aforementioned signals involving money are all 3-digit (000 to 999) integer types. A component was written to "index" into these values and take separate them into 3 1-digit integer types. This was so they could easily be used for the SSD function. The component uses the MOD operator and the divide operator (which floors the result) to do this. In addition, the frequently used push button debouncer component was used for the buttons in this project.

# 3 Simulation Results

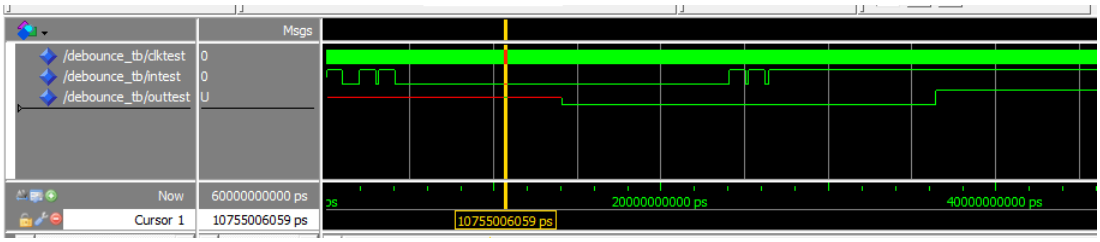


Figure 2: Showing debouncing program works with simulated bouncy input

# 4 Demonstration on FPGA

The design was synthesized onto the Altera Cyclone IV E DE2-115F29C7 FPGA.

- User Controls:
  - Product Select: Switches 17 and 16
  - Product Choose: Key 3
  - Coin Select: Switches 1 and 0
  - Coin Choose: Key 2
  - Cancel: Key 0
- Output Displays:
  - Product Abbreviation: Hex 5
  - Main Display: Hex 2, 1, and 0
  - Delivery Alert: Green LED 8 (LEDG8)


Figure 3: Display in Idle state


Figure 4: Display after choosing Coffee


Figure 5: Inserted a dime


Figure 6: Inserted a quarter


Figure 7: Inserted a dollar


Figure 8: Inserted another dollar; delivery in progress; displaying change


Figure 9: Inserted an invalid coin


Figure 10: Inserted a quarter and dime; canceled; displaying return change

# 5 Conlusion

The project simulated and synthesized correctly. The code required sufficient knowledge of structural, concurrent, and sequential VHDL. A package was designed to compartmentalize the code to make it easier to read and debug. Recently learned functions and procedures were used within this package, as well as components used and user-defined data types. A current issue is that the product abbreviation SSD (Hex 5) currently displayed a dash in the Idle state. A more user-friendly design would be to display the abbreviation before the product is selected; otherwise, the customer needs an external way to know which code goes with which product. Other than that, the project was very cleanly and quickly developed. A strong base of knowledge and practice with VHDL and understanding of hardware constraints has been achieved as a result of not only this project, but the entire course. This knowledge could also serve as a jumping off point if another hardware description language, for example Verilog HDL, needs to be used in the future.