Name: Ahmed Essam Abdel-Aleem
ID: 900193476

Vending Water Report

I. The project discerption

The project is building a water vending machine which is responsible for taking a specific amount of money to pour water for the customer. For more details, the user can increase the amount of money by three types of incrementations: one, five, or ten pounds. When the money reaches 20 pounds (The water price), the water is pouring. Consequently, there are three conditions that can stop the water:

1- The user presses the push button while pouring

2- Thirty seconds have passed without any interaction

3- The user adds more amount of money

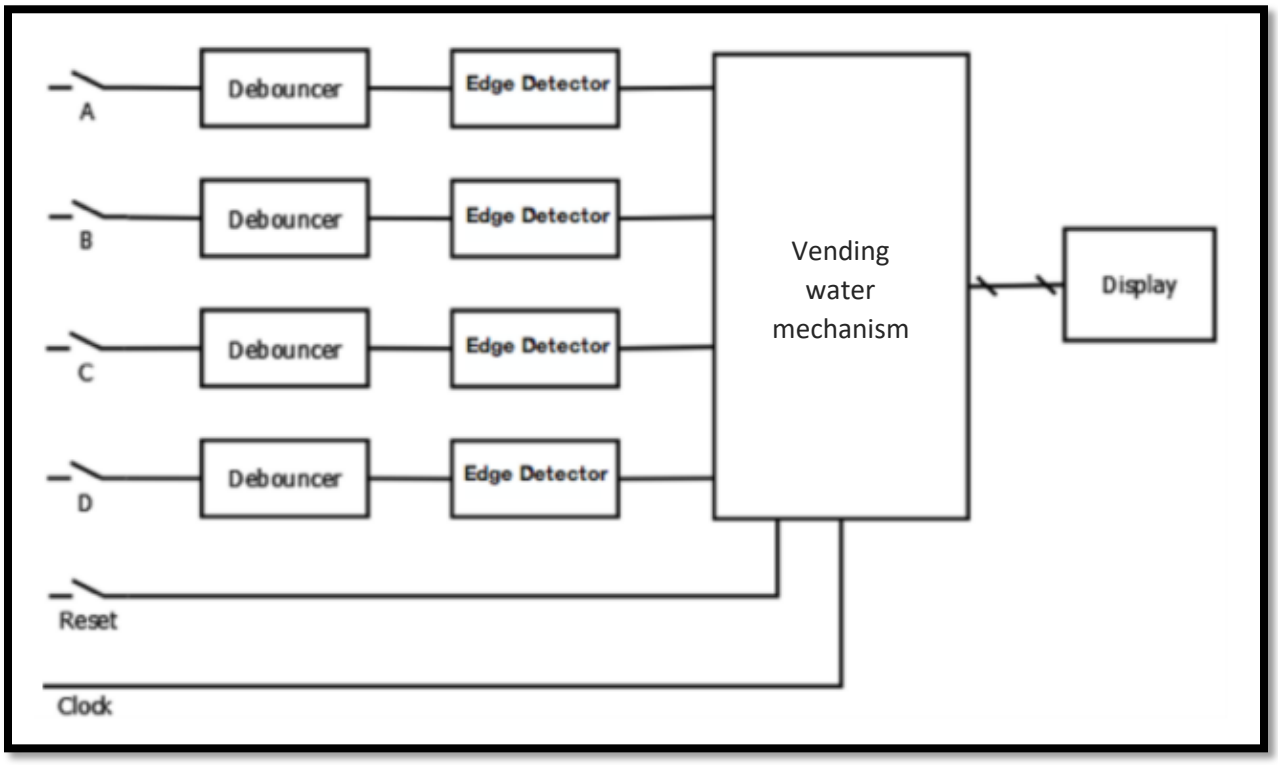
According to the input, there are four main buttons to take the interactions from the user. The first three buttons are responsible for increasing the money by one, five, or ten pounds respectively. The fourth button is for stopping the water while pouring. In addition, there is an extra button (made for simplicity) which is resetting the whole system to its initial position.

According to the output, the amount of money will be displayed on the seven-segment display. Once the machine detects twenty pounds in the given money, it takes them from the available amount and turn on the water. In other words, the displayed amount will be the available amount minus 20. As for the water, it will be presented by a led below the display to be ON while water is pouring and OFF in the inverse position.

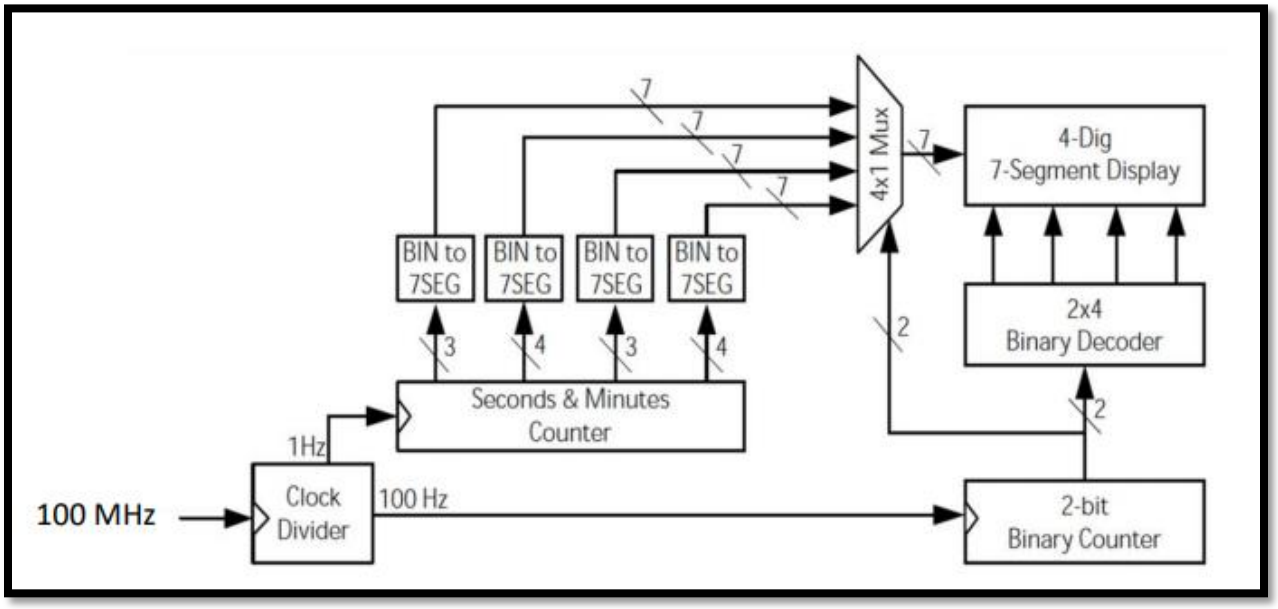
- **What happened after water is pouring**
  - If the user presses the stop button, the water will stop pouring (LED is OFF), but with the same amount of displayed money.
  - If the user adds more money, the displayed amount will increase by the input amount, and the water will automatically stop pouring.
  - If 30 seconds have passed without any interaction from the user, the system will return to its initial position (Money = 0 & LED = OFF)

II. The block diagrams

The project modules can be illustrated by the following block diagram:



The display part can be shown in the following block diagram:



- **Explanation of the block diagram:**

The input will be taken from the user remotely by the keyboard buttons using UART, the one, five, ten, stop, and reset buttons will be controlled by ‘a’, ‘b’, ‘c’, ‘d’, and ‘r’ respectively. It will go into a synchronous, debouncer, and rising edge detector (Each module’s function will be explained in the next section). After that, the inputs will go into the vending water mechanism which depends on the Moore Finite State Machines. The current state will determine the displayed amount of money and the led mood. Consequently, the two numbers of the amount will go into a collection of BCDs, MUXes, decoder, and clock dividers to execute the final function which is displaying the amount of money on the seven-segment display.

III. Explanation of the code

In this section, we will introduce the functionality of each module in the project and how the input is processing until it gives a proper output.

a) UART

The UART is responsible for remotely controlling the inputs by the keyboard buttons instead of taking them from the FBGA. The module takes the ASCII code of the specified keyboard button and compares it to any input from the keyboard. For example, when the user enters ‘a’, every bit of the 8 ASCII bits of ‘a’ are communicated and stored in an array of 8 bits through the pin B18 to the FBGA through the USB connecting the FBGA to the PC. Then, The HDL code compares the received value to the parameter given to it and if they are the same, the part of the code where ‘a’ is stimulated is executed. Similarly, the rest of the push buttons are handled. Hence, there are five instantiations of the UART for the five input buttons.

b) Synchronizer

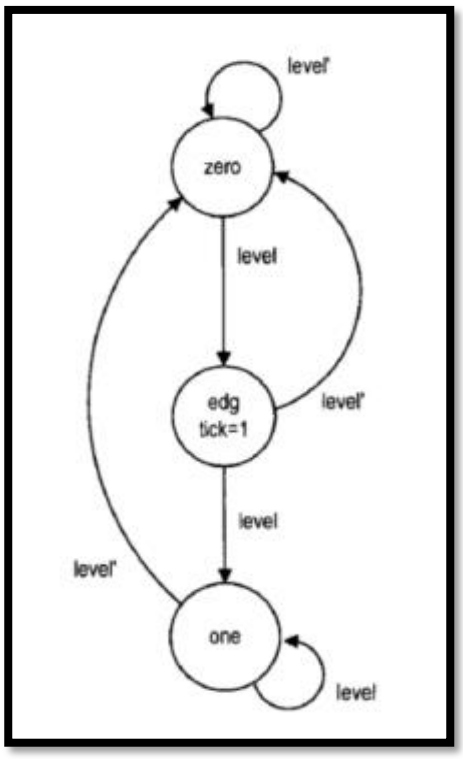
As known in the previous labs, there are two types of input which are synchronous and asynchronous. If asynchronous inputs are handled carelessly, these inputs can lead to static voltage within the system, causing an accident system failure that is extremely difficult to track and handle. Since, it’s better for guaranteeing good logical levels to pass the asynchronous inputs through the synchronizer. However, the functionality of this module will not appear in

our case since we are using the UART buttons to control the output. There are four instantiations of this module to adjust the four main inputs.

c) Debouncer

If we are trying to take the input from the FPGA, the pushbutton switches are considered mechanical devices. Therefore, when any of them are pressed, the switch may bounce back and forth a few times before settling down on the final output. The bounces lead to glitches in the signal, which could be translated into many presses. The bounces usually settle within 20ms. Hence, the purpose of a de-bouncing circuit is to filter out the glitches associated with switch transitions through taking the signal in three different points with this range and decide the output according to the signal's stability.

d) Rising Edge Detector



This module is the third one through which the inputs pass before entering the finite state machine. Unlike the synchronizer and debouncer, the rising edge detector is necessarily used to guarantee to generate a short one-clock-cycle when the input changes from 0 to 1. In other words, this module is responsible for considering any press from the user (by the UART or from the FBGA) as one positive edge whatever the time it takes. Therefore, it works by building a finite state machine and detect a positive output when the current state is the tick between zero and one. Similarly, there are also four instantiations in the main module.

e) Finite State Machine

The Finite State machine is simply a programming model which presents all the possible states that the system can be in at any point of time. Our project has 30 different states; since it's so complicated to draw a state diagram presenting the connections between those states, it will be explained now in details.

- The system has possibilities to print from zero amount of money to 29 pounds as maximum. The maximum case is generated when the user

reaches 19 pounds, then presses another ten pounds. Hence, we will define 30 states. Each of them will determine the displayed amount of money (equal to the state number), and the state of the water (LED is on in the states more than or equal 20).

- The system at any point of time can be interpreted by a press from the four main buttons: one, five, ten, and stop (noted that the stop button will not appear until the current state reaches 20 or exceeds it). Therefore, there will be four if statements in every state block below 20, while five if statements, to determine what the next state will be.

- In all states below the state “20”, the four if statement are:
  - If “one” is pressed, system will go to the next state.
  - If “five” is pressed, system will go to the advanced state with five digits.
  - If “ten” is pressed, system will go to the advanced state with ten digits.
  - If nothing is pressed, the system will remain on the current state.

- For the states which are more than 20, it has an extra if statement for the stop button. In this case, it will go the same version of the state but without the LED opened. For example, if the system is in the state “26”, which displays 6 pounds on the screen while pouring water, the system will go to the state “6” if the user presses the stop button. Note that the only difference between the two states is the LED mood.

- For handling returning the system to the initial state when 30 seconds is passed. We declare a 5-bit counter and use a second always block to increment this counter by one every single second while initializing it to zero when the user presses any of the input buttons. Therefore, there is another if statement in each state block to check whether the counter reaches 30 or not. If yes, then the system will go the state “0”.

- There is a third always statement that assign the next state, declared in the previous always block, to the current state, besides taking into consideration the reset button to make the same functionality of the thirty seconds.

- The Display Components

f) Binary-Coded-Decimal (BCD)

The system now has two different types of outputs. The first one is the LED variable, and this output will be connected to one of the FPGA's LEDs through the constraints file. The second output is the amount of money, which is stored in two 4-bit variables (can store the range from 0 to 9). Now, the BCD module is responsible to translate these numbers to their representations in the seven-segment display.

g) 1-bit counter

The counter module takes an adjusted clock, differentiated from the FBGA clock, at frequency of 100 Hz. This frequency is very suitable to choose which segment will be ON without realizing that there is swapping between the segments. With every positive edge of this adjusted clock, the output of the counter toggles between 0 and 1. This toggling will help us, as an input to the decoder, to determine which of the seven segment displays will be enabled.

*h) 1*2 decoder*

The decoder decides which of the seven-segment display is enabled using the output generated from the counter. Since we need only the first two segments to be displayed, we deactivate the other two segments by assigning en[3] and en[2] to 1 (OFF, since the decoder is active low). On the other hand, for en[1] and en[0], it depends on the counter's output such that if it equals zero, then the decoder will enable en[0] and disable en[1], and vice versa.

i) MUX

The mux module is responsible for choosing between the 1st and 2nd segment to decide which one will be displayed on the seven-segment display.

- As mentioned in the previous section, the combination of the MUXes, the decoder, and the counter helped display different numbers on the seven-segment display at the same time. It chooses different number each time, but it happens very fast (100 Hz) To the extent that it is not seen with the naked eye and is noticed as if it is working all the time.