Birla Institute of Technology and Science – Pilani, Hyderabad Campus
First Semester 2021-22
EEE F348: FPGA based System Design Lab (Open Book)

A1

Time: 11:00AM-12:30 PM **Date: 27-10-2021** **Max. Marks: 60**

Name: Vishwas Vasuki Gautam **ID No:** 2019A3PS0443H

Instructions:

1) The exam duration including implementation and Bitstream generation is 90 Minutes.
2) Examination is open book type. You can use the lab documents for reference. You can also reuse Verilog codes implemented during the regular labs, If needed.
3) The design files (Verilog files) uploaded will be verified against the outputs shown using Bitstream. **Any discrepancies (e.g design files are wrong yet correct Bitstream verified) will automatically result in zero marks. No partial marks will be awarded in such cases.**
4) **The students shall not resort to any unfair means in attempting the exam. The consequences of adopting unfair means will be severe.**

Question:

Design and implement a Moore based control system (on FPGA) for a vending machine. The vending machine has 2 items one which costs Rs. 2 and other which costs Rs. 3 (of course nowadays you don't get much for these prices ☺). The vending machine can accept One-Rupee coin and two-rupee coins only.

If the user inserts the coins of right value and press "Enter" then, the item, corresponding to the value of coins entered, will be released.
If the "Enter" button is pressed without inserting the right coins (if the value of coins inserted is less than Rs. 2) then error signal is raised.
If enter is pressed after the total value of coins inserted is >3 then only one item corresponding to Rs. 3 has to be released.

The control unit of vending machine has three inputs
(a) "**R1**" which is activated (active high pulse) when one-rupee coin is inserted into the machine.
(b) "**R2**" which is activated (active high pulse) when two-rupee coin is inserted into the machine
(c) "**Enter**" which is activated (active high pulse) when "Enter" is pressed.

The control unit of vending machine has two outputs

(a) "Release_2" which becomes 1 when the value of the coins is equal to Rs.2 indicating that item which costs Rs. 2 has to be released by vending machine.
(b) "Release_3" which becomes 1 when the value of the coins is equal to Rs.3 indicating that item which costs Rs. 3 has to be released by vending machine.
(c) "Error" which becomes 1 indicating that the total value of coins entered is less than the value least priced item

For implementation purpose, Assume the following

1. "**R1**" is connected to push button switch: **T18**
2. "**R2**" is connected to push button switch: **P16**
3. "**Enter**" is connected to Push button switch: **R16**
4. "**Release_2**" is connected to LED: **T22**
5. "**Release_3**" is connected to LED: **T21**
6. "**Error**" is connected to LED: **U22**
7. "**Clk**" for main module is connected to internal clock: **Y9**

8. "**Reset**" is connected to push button switch: **R18**

(Note: You can re-use the Verilog code for Clk division and debouncing, which was implemented in regular lab, if required. You need not implement it again here. However, you need to show their instantiation in the main module)

Answer the following questions which are related to the implementation of above design.

1. Copy the image of the Verilog code for the FSM [30M].

Answer:

```
`timescale 1ns / 1ps

//VISHWAS VASUKI GAUTAM
//2019A3PS0443H

module vendingMachineFSM(
    input clk,
    input rst,
    input R1,
    input R2,
    input Enter,
    output reg Release_2,
    output reg Release_3,
    output reg Error
    );

parameter zeroRupees = 2'b00;
parameter oneRupees = 2'b01;
parameter twoRupees = 2'b10;
parameter threeRupees = 2'b11;

reg [1:0] presentState, nextState;

always @(posedge clk, posedge rst)
    begin
        if(rst)
            presentState <= zeroRupees;
        else
            presentState <= nextState;
    end

always @(presentState, R1, R2)
    begin
        case(presentState)
        zeroRupees:
            nextState <= (R1 & ~R2)? oneRupees: (~R1 &
R2)? twoRupees: zeroRupees;

        oneRupees:
            nextState <= (R1 & ~R2)? twoRupees: (~R1 &
R2)? threeRupees: oneRupees;

        twoRupees:
            nextState <= ((R1 & ~R2) | (~R1 & R2)) ?
threeRupees: twoRupees;

        threeRupees:
            nextState <= threeRupees;

        endcase
    end
```

```
always @(presentState, Enter)
    begin
        case(presentState)
            zeroRupees:
                begin
                   if(Enter)
                    begin
                    Error <= 1'b1;
                    Release_2 <= 1'b0;
                    Release_3 <= 1'b0;
                   end
                end

            oneRupees:
                begin
                    if(Enter)
                    begin
                    Error <= 1'b1;
                    Release_2 <= 1'b0;
                    Release_3 <= 1'b0;
                   end
                end

             twoRupees:
                begin
                    if(Enter)
                    begin
                    Error <= 1'b0;
                    Release_2 <= 1'b1;
                    Release_3 <= 1'b0;
                   end
                end

             threeRupees:
                begin
                   if(Enter)
                    begin
                    Error <= 1'b0;
                    Release_2 <= 1'b0;
                    Release_3 <= 1'b1;
                   end
                end
             default:
                begin
                    Error <= 1'b0;
                    Release_2 <= 1'b0;
                    Release_3 <= 1'b0;
                end
          endcase
    end

endmodule
```

2. **Which signal will you use as clock for the above FSM and Why? [5M].**

Answer: The signal used for the clock will be from the Debounce module. Because we need to ensure that the sampling is in the middle of the inputs and the posedge of this signal can be used for the FSM, as the FSM only changes when the inputs are affected. So the inputs are passed through 3 FF and the output clk is taken from these intermediate flip flop values.

3. **Copy the image of the Verilog code for Main module which has instantiations of all the modules and additional statements (if any) used in your design. [10M]**

Answer:

```
`timescale 1ns / 1ps


module main(
    input clk,
    input rst,
    input R1,
    input R2,
    input Enter,
    output Release_2,
    output Release_3,
    output Error
    );

wire Clk190;
wire ClkFSM;
assign keyPress = R1 | R2 | Enter;

clkDiv clk_division(clk, rst, clk190);
Debounce db(clk190, keyPress, rst, clkFSM);
vendingMachineFSM fsm(clkFSM, rst, R1, R2, Enter,
Release_2, Release_3, Error);

endmodule
```

4. **Correct the hardware implementation (using the Bitstream) [15M]**

<u>Upload Instructions:</u>

1. Upload a zipped folder with name Mid_Sem_IDNO_NAME.zip, containing the
   (a) A copy of this document with all the answers (as IDNo_Name.pdf) and
   (b) All the Design Verilog (.v files) of all the modules (7 Sub-modules)
   (c) Bitstream file.
   This .zip/.rar file should be uploaded through the google form link given below:
   https://forms.gle/jTpjk8JzA9tvdtZ97

2. Bitstream file for verification should be uploaded in the google drive folder used for regular labs.