EE2016 - Microprocessors (Theory + Lab)

Experiment-2

1 Introduction

This experiment involves (i) simulating a D flip-flop using Xilinx Vivado (ii) completing the Verilog code given below that extends the basic D flip-flop design by adding a reset signal (call it *dflipflop_with_reset*). (iii) designing a 3-bit Johnson counter via instantiations of *dflipflop_with_reset* and (iv) writing clock divider and decoder modulus in Verilog to enable implementation of the Johnson counter design on a seven segment display on the FPGA board.

2 Simulation of D flip-flop in Vivado

Use the Verilog modules for D flip-flop and testbench uploaded on Moodle to perform a simulation in Xilinx Vivado.

3 Extension of D flip-flop via addition of reset signal

Complete the following Verilog module

```
module dflipflop_withreset(q, qbar, d, rst, clk);
output q,qbar;
input d, rst, clk;
reg q,qbar;
always @ (posedge clk)
 begin
   if (~rst)

      ........  // Initialize q and qbar

   else

   ............ // Transfer data to q and qbar

 end
endmodule
```

4 Structure of a 3-bit Johnson counter

A 3-bit Johnson counter is shown in Figure 1. It will have states given by 000, 100, 110, 111, 011 and 001 and this sequence repeats.

5 Writing a Verilog module for clock divider

The FPGA board provides a 50 MHz clock. In order for a viewer to see the count (i.e., distinguish the digits appearing), one needs to "divide the clock" to get a lower frequency. Complete the following clock divider module.

```
module clk_divider (inClk,reset,outClk);
input inClk;
input reset;
```

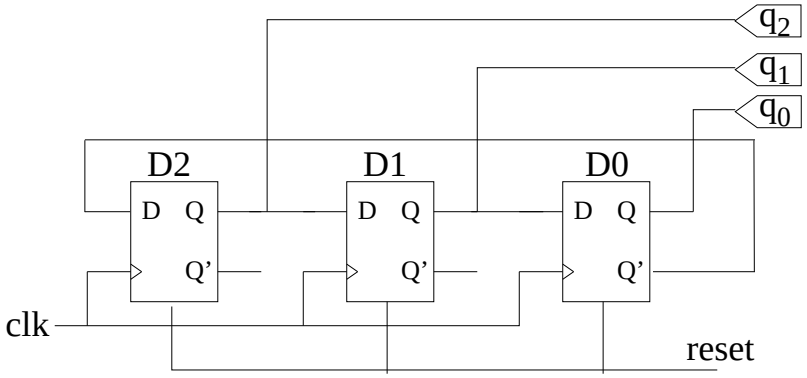

Figure 1: 3-bit Johnson counter

```
output outClk;
reg outClk;
// declare a vector (of type reg) called clockCount of appropriate size

always @(negedge reset or posedge inClk)
begin
if (reset == 1'b0)
        // initialize clockCount, outClk suitably

else
// increment clockCount
// determine an appropriate value v (by hand) and
// check if clockCount has reached v
// to reset and negate outClk
end
endmodule
```

6 Writing a decoder module in Verilog

We want the results of the 3-bit Johnson counter to appear on the seven segment display on the FPGA board. Complete the following Verilog module for the decoder.

```
module decoder(cntr,Seven_Seg);
input [2:0] cntr;
output [7:0] Seven_Seg;
reg [6:0] val;

assign Seven_Seg = {1'b1,~val};  // Think of the reason for this assignment ...

always@(cntr)  // Think of the reason for having cntr in the sensitivity list
  begin
   case(cntr)
   3'd0: val = 7'b0111111;
.........................
.............................
.............................
   endcase
  end

endmodule
```

7 Implementing the Johnson counter design on the FPGA board

A partially-filled Verilog module for a 3-bit Johnson counter is given below that includes instantiations to clock divider and decoder. Note also that the code has a statement that "enables" the display. Complete the module.

```
// Top level module
module johnson3bit(Seven_Seg,in_clk,rst,digit);
input in_clk, rst;
output [7:0] Seven_Seg;
output [3:0] digit;
wire [2:0] cntr;

wire q0,q1,q2;
wire  q2bar, q1bar, q0bar;
wire out_clk;

assign digit = 4'b0001;  // enable seven seg display ...
assign cntr = {..........};


-- Add instantiations to dflipflop_withreset -----

clk_divider cd0(.................);
decoder dec0(................);
```

The final step to get the implementation running on the FPGA board is inclusion of an appropriate Xilinx design constraints (.xdc) file. Complete the partially filled .xdc file (contents) given below.

```
# Clock signal - identify package pin and fill in
set_property -dict { PACKAGE_PIN .....    IOSTANDARD LVCMOS33 } [get_ports { in_clk }];

# Sliding switch -- to start count
set_property -dict { PACKAGE_PIN L5    IOSTANDARD LVCMOS33 } [get_ports { rst }];



#Enable seven segment display
set_property -dict { PACKAGE_PIN F2    IOSTANDARD LVCMOS33 } [get_ports {digit[0]}];
# .... fill three more  lines here ..... use package pins E1, G5 and G4

#locations of the segments on the display
set_property -dict { PACKAGE_PIN G2    IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[0]}];

# fill 7 more lines -- use package pins G1, H5, H4, J5, J4, H2 and H1
```

8 Report

Submit a report on the experiment on Moodle (within a week of this experiment). One report per group (with the names of the group members) is sufficient. The report should contain details of the solution

(including the code) and your observations and experience (in programming, debugging etc.). Please note that reports that closely match those of other groups will be penalized.