EE2016 Experiment 2

Group 3 - EE23B039, EE23B033, EE23B027

1 Aim:

To display the outputs of a 3 bit johnson counter on a 7 segment display on FPGA board

2 D Flip Flop without Reset

Code for Module

```
//circuit D Flipflop

module d_flipflop(q, q_bar, d, clk);
        output reg q, q_bar;
        input d, clk;
        always @(posedge clk)
        begin
                q <= d;
                q_bar = ~q;
        end
endmodule
```

Testbench

```
// testbench for d_flipflop
`timescale 1ns/100ps

module test_dff;
        wire q, q_bar;
        reg d, clk;

        initial begin
                clk = 0;
                forever
                        #10
                        clk = ~clk;
                end

        d_flipflop DUT(q, q_bar, d, clk);

        initial begin

                $dumpfile("d_flip_flop.vcd");
                $dumpvars(0, test_dff);
                d = 0;
                #10
                d = 1;
                #10
                d = 0;
            #100
            $finish;

                end


endmodule
```

3 D Flip Flop with Reset

Code for Module

```
//circuit d_flip_flop_with_reset

module dflipflop_with_reset(q, q_bar, d, rst, clk);
    output q, q_bar;
    input d, rst, clk;
    reg q, q_bar;

    always @ ( posedge clk)
    begin
        if (~rst)
        begin
        q <= 1'b0;
        q_bar <= 1'b1;
    end

    else
        begin
        q <= d;
        q_bar <= ~q;
        end
    //q_bar = ~q;
    end

endmodule
```

Testbench

```
// testbench for d_flipflop
`timescale 1ns/100ps

module test_dff_rst;
        wire q, q_bar;
        reg d, clk, rst;

        initial begin
                clk = 0;
                forever
                        #10
                        clk = ~clk;
                end

        dflipflop_with_reset DUT(q, q_bar, d, rst, clk);

        initial begin

                $dumpfile("d_flip_flop_rst.vcd");
                $dumpvars(0, test_dff_rst);
                // Initial conditions
                rst = 1;
                d = 0;

                // Apply stimulus
                #10 rst = 1; d = 1;
                #10 rst = 0; d = 1;
                #10 rst = 0; d = 0;
                #10 rst = 1; d = 1;

                // End simulation
                #50
```

```
            $finish;
            end

endmodule
```

4 3-bit Johnson Counter

Code for Module

```
//Johnson counter

module johnson_counter(out, clk, rst);
    output reg[2:0] out;
    input clk, rst;
    wire q2, q1, q0, q2b, q1b, q0b;

    dflipflop_with_reset d2(q2, q2b, q0b, rst, clk);
    dflipflop_with_reset d1(q1, q1b, q2, rst, clk);
    dflipflop_with_reset d0(q0, q0b, q1, rst, clk);

    always @(posedge clk)
        begin
            if (~rst)
            begin
                out <= 3'b000;
            end
            else
            begin
                out <= {q2, q1, q0};
            end
        end

endmodule
```

Testbench

```
//testbench johnson_counter

`timescale 1ns/100ps

module test_johnson_counter;
        wire[2:0] out;
        reg clk, rst;
        reg q2, q1, q0, q2b, q1b, q0b;

        initial begin
            clk = 0;
            forever
                #10
                clk = ~clk;
            end
        johnson_counter DUT(out, clk, rst);

        initial begin
            $monitor("At t = %t, out = %b", $time, out);
            rst = 0;
            #10
            rst = 1;
            #10
            rst = 0;
            #10
            rst = 1;
```

```
                #100
                $finish;
                end
endmodule
```

5 Clock Divider

Code for Module

```
module clk_divider(inClk,reset,outClk);
    input inClk;
    input reset;
    output outClk;
    reg outClk;
    reg[32:0] clockCount;

    always @( posedge inClk)
        begin
        if (reset == 1'b0)
        begin
        clockCount <= 33'b0;
        outClk <= 1'b0;
        end

        else
        begin
        if (clockCount == 33'd25000000)
        begin
        clockCount <= 33'b0;
        outClk <= ~outClk;
        end
        else
        begin
        clockCount <= clockCount + 33'd1;
        end
        end

        end
endmodule
```

Testbench

```
`timescale 1ns / 100ps

module tb_clk_divider;

    reg inClk;
    reg reset;

    wire outClk;

    clk_divider uut (
        .inClk(inClk),
        .reset(reset),
        .outClk(outClk)
    );

    initial begin
        inClk = 0;
        forever #10 inClk = ~inClk;
    end
```

```
    initial begin
        reset = 0;
        #100
        reset = 1;
        #10000000000
        $finish;
    end


endmodule
```

6 7-Segment Decoder Module

Code

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

assign Seven_Seg = {1'b1,~val};

always@(cntr)
begin
case(cntr)
3'd0: val = 7'b0111111;
3'd1: val = 7'b0000110;
3'd2: val = 7'b1011011;
3'd3: val = 7'b1001111;
3'd4: val = 7'b1100110;
3'd5: val = 7'b1101101;
3'd6: val = 7'b1111101;
3'd7: val = 7'b0000111;

endcase
end
endmodule
```

7 Top-level Code for 3-bit Johnson Counter Implementation on FPGA Board

Code

```
// 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 out_clk;

assign digit = 4'b0001;


johnson_counter jc0(cntr, out_clk, rst);
clk_divider clk0(in_clk, rst, out_clk);
decoder dec0(cntr,Seven_Seg);
```

endmodule

8 Constraints File

```
# Clock signal
set_property -dict { PACKAGE_PIN N11     IOSTANDARD LVCMOS33 } [get_ports { in_clk }];

# Switches
set_property -dict { PACKAGE_PIN L5     IOSTANDARD LVCMOS33 } [get_ports { rst }];#LSB


#7 segment display
set_property -dict { PACKAGE_PIN F2     IOSTANDARD LVCMOS33 } [get_ports {digit[0]}];
set_property -dict { PACKAGE_PIN E1     IOSTANDARD LVCMOS33 } [get_ports {digit[1]}];
set_property -dict { PACKAGE_PIN G5     IOSTANDARD LVCMOS33 } [get_ports {digit[2]}];
set_property -dict { PACKAGE_PIN G4     IOSTANDARD LVCMOS33 } [get_ports {digit[3]}]; #MSB


set_property -dict { PACKAGE_PIN G2     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[0]}];#A
set_property -dict { PACKAGE_PIN G1     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[1]}];#B
set_property -dict { PACKAGE_PIN H5     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[2]}];#C
set_property -dict { PACKAGE_PIN H4     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[3]}];#D
set_property -dict { PACKAGE_PIN J5     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[4]}];#E
set_property -dict { PACKAGE_PIN J4     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[5]}];#F
set_property -dict { PACKAGE_PIN H2     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[6]}];#G
set_property -dict { PACKAGE_PIN H1     IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[7]}];#DP
```

9 Errors and Corrections Made

1. **Error**: In the clock divider module, we had given active high reset and all others were active low resets, because of which there was no synchronization in the reset and display was not changing from 0.
Correction: Changed all the resets to active low.
2. **Error** (bit stream generation error) In the constraints file, we had given only one input pin for 7 segment display instead of 4 inputs for 4 bit digit.
Correction: Added 3 more package pins for digit.

10 Observations

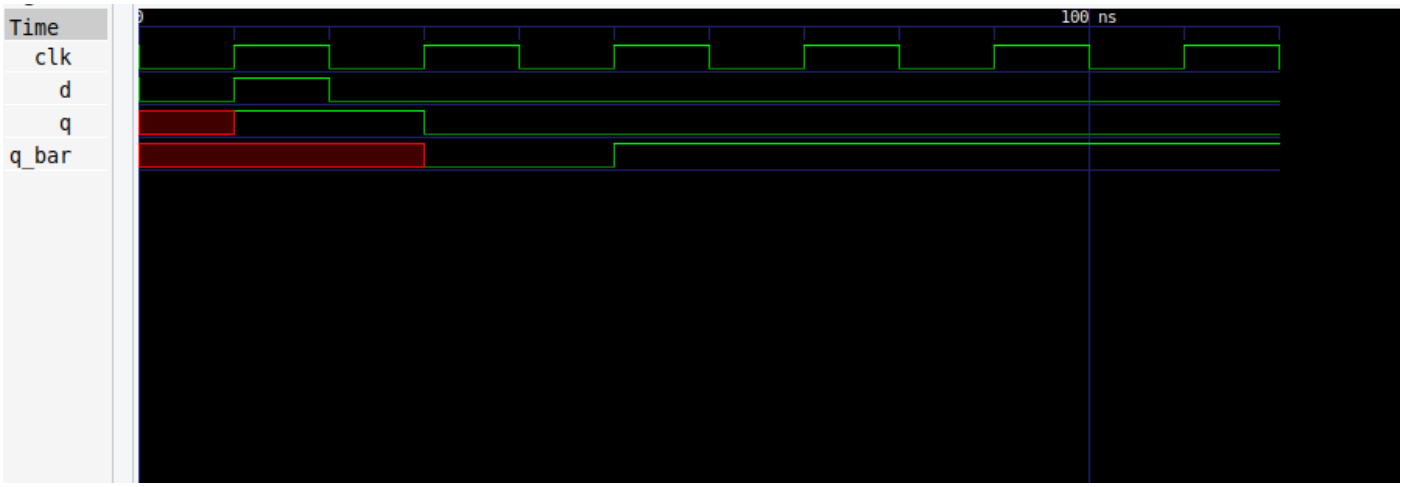

Figure 1: gtkwave of dff

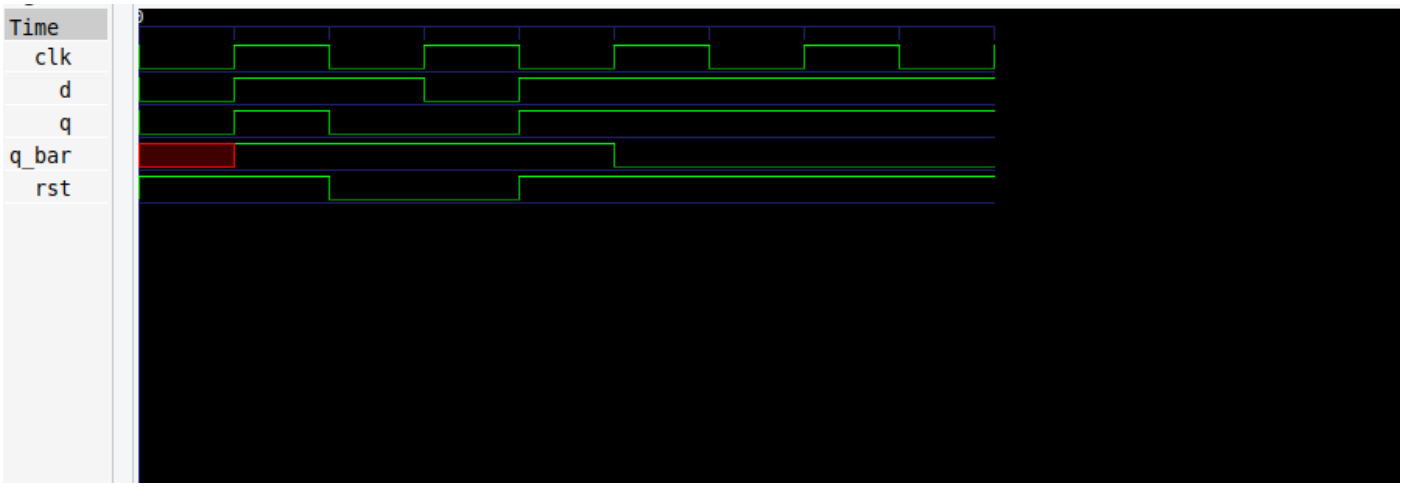

Figure 2: gtkwave of dff with reset

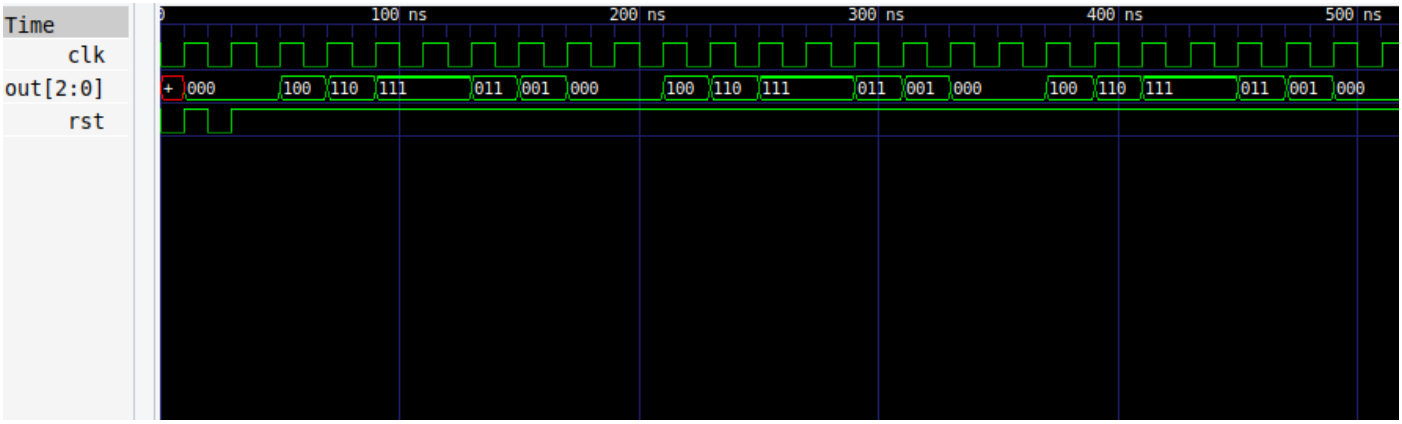

Figure 3: gtkwave of johnson counter