EE2016: Experiment 2

Group:19

Sparsh Gupta EE23B117

Charan Srikkanth EE23B127

Kaushik Iyer EE23B135

August 14, 2024

1 Introduction

This experiment focuses on designing and simulating a D flip-flop using Xilinx Vivado, and then extending its functionality with a reset signal, then using this extended D flip-flop to implement a 3-bit Johnson counter. The counter is further used to drive a seven-segment display on an FPGA board using a clock divider and decoder module. The experiment demonstrates the synthesis and implementation of digital logic circuits using Verilog.

2 Objectives

The primary objectives of this experiment are:

1. Simulate a D flip-flop using Xilinx Vivado.
2. Extend the basic D flip-flop design by adding a reset signal.
3. Design a 3-bit Johnson counter via instantiations of the D flip-flop with reset.
4. Write Verilog modules for a clock divider and decoder modules for displaying the Johnson counter output on a seven-segment display.

3 D Flip-Flop

D flip flop consist of a single input D and two outputs (Q and Q'). The basic working of D Flip Flop is as follows:

1. When the clock signal is low, the flip flop holds its current state and ignores the D input.
2. When the clock signal is high, the flip flop samples and stores D input.

3.1 Verilog Implementation

```
// dff.v
module dff(q, qbar, d, clk);
    input d, clk;

    output qbar;
    output reg q;

    not(qbar, q);
    always @(posedge clk) q <= d;
endmodule
```

3.2 Testbench

```
//dff_tb.v
module dff_tb();
    reg d=1'b1;
    reg clk=1'b0;
    wire q;
    wire qbar;
    dff d1(q, qbar, d, clk);
    initial begin
        $monitor("at t = %t clock is %b q is %b rst is %b",$time,clk,q,qbar);
        repeat(30) #10 clk=~clk;
    end

    initial begin
        #5 d = ~d;
        #20 d = ~d;
        #20 d = ~d;
        #10 d = ~d;
        #10 d = ~d;
        #20 d = ~d;
        #20 d = ~d;
        #20 d = ~d;
    end
endmodule
```

3.3 Output

Given below is the output seen on Xilinx vivado for the above testbench.

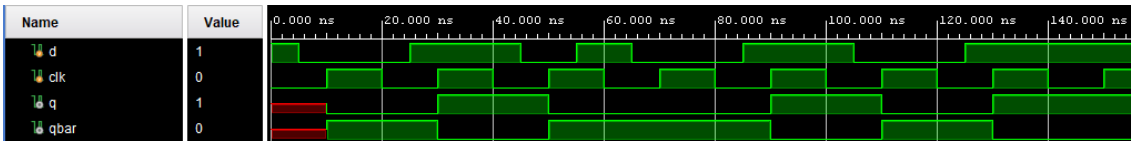

Figure 1: OUTPUT

4 D Flip-Flop with reset

This part adds the functionality of a reset pin to the D-flip flop developed earlier. The reset pin works with active low, which means when reset pin is low,the output is zero. when reset pin is high, implies output is equal to input(d).

```
    rst =0 => q=0;
    rst =1 => q=d;
```

4.1 Verilog Implementation

```
// dff.v
module dff_rst(q, qbar, d, clk, rst); //rst is active low meaning if rst =0 then q=0
    output reg q;
    output wire qbar;
    input d;
    input clk, rst;
```

```
    not(qbar, q);

    initial q = 0;

    always @(posedge clk) begin
        if (~rst) q <= 0;
        else q <= d;
    end
endmodule
```

4.2 Testbench

```
//dff_tb.v
module dff_tb();
    reg d=1'b1;
    reg clk=1'b0;
    wire q;
    wire qbar;
    dff d1(q, qbar, d, clk);
    initial begin
        $monitor("at t = %t clock is %b q is %b rst is %b",$time,clk,q,qbar);
        repeat(30) #10 clk=~clk;
    end

    initial begin
        #5 d = ~d;
        #20 d = ~d;
        #20 d = ~d;
        #10 d = ~d;
        #10 d = ~d;
        #20 d = ~d;
        #20 d = ~d;
        #20 d = ~d;
    end
endmodule
```

4.3 Output

Given below is the output seen on Xilinx vivado for the above testbench.White colour graph is for clock , golden colour graph is to show reset pin.

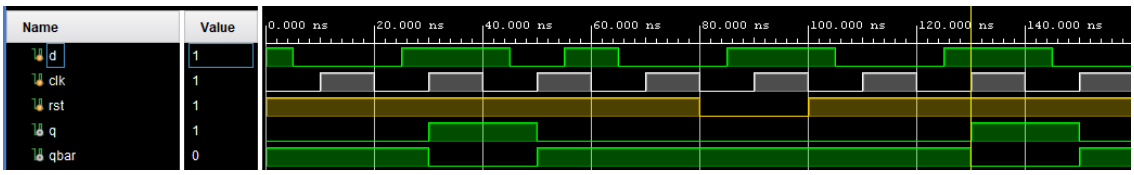

Figure 2: OUTPUT

5 Johnson Counter

The Johnson counter is designed by connecting three instances of the D flip-flop with reset in a sequential manner. The output of the last flip-flop is fed back as input to the first flip-flop, with appropriate connections to form the Johnson counter. It will have states given by 000, 100, 110, 111, 011 and 001 and this sequence repeats. The Verilog code for the counter is given below:

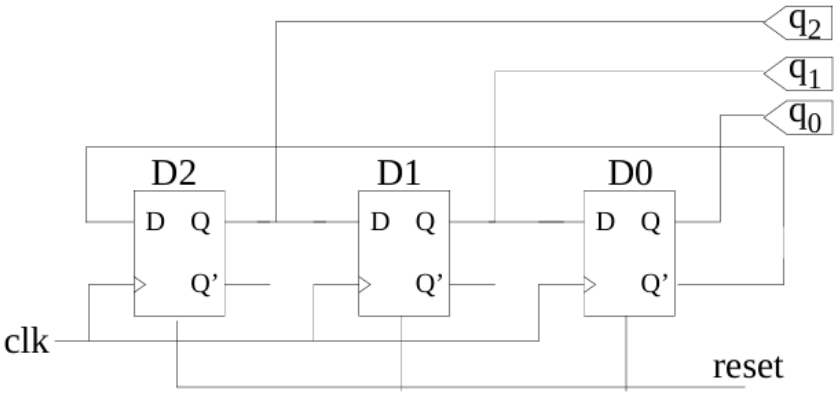

Figure 3: Johnson Counter

5.1 Verilog Implementation

```
// johnson_counter.v
module johnson_counter(q0, q1, q2, qbar0, qbar1, qbar2, clk, rst);
    input clk, rst;
    output q0, q1, q2, qbar0, qbar1, qbar2;

    dff_rst d0(q0, qbar0, q1, clk, rst);
    dff_rst d1(q1, qbar1, q2, clk, rst);
    dff_rst d2(q2, qbar2, qbar0, clk, rst);
endmodule
```

5.2 Output

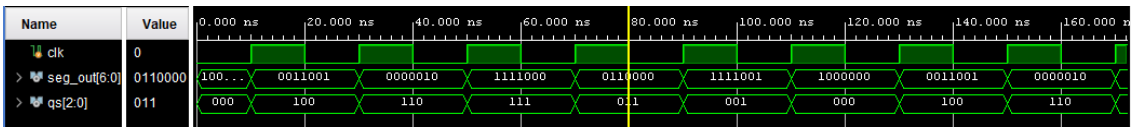

Figure 4: OUTPUT

6 Clock Divider

To display the Johnson counter output on an FPGA board, a clock divider is implemented to slow down the clock signal. In this case we are dividing the clock by 5×10^7 to get the frequency to 1 Hz

6.1 Verilog Implementation

```
// clock_divider.v
module clock_divider(inclk, rst,  outclk);
    input inclk;
    input rst;
    output reg outclk=0;
    reg[32:0] clock_count;
```

```
    initial clock_count=0;

    always @(posedge inclk) begin
        if (rst == 0) begin
            outclk = 0;
            clock_count = 0;
        end
        else begin
        clock_count=clock_count+1;
        if(clock_count==50000000) begin
            outclk = ~outclk;
            clock_count = 0;
        end
    end
end
endmodule
```

7 Seven Segment Display

A decoder module is designed to convert the counter output into a format suitable for driving a seven-segment display.

7.1 Verilog Implementation

```
// sseg.v
module sseg (
    input [2:0] cntr,
    output reg [6:0] seg
    );
    initial  seg[6:0] = 7'b0001110;
    // activate on any input change
    always @*
    begin
        case(cntr)
            3'b000: seg[6:0] <= 7'b1000000;    // digit 0
            3'b001: seg[6:0] <= 7'b1111001;    // digit 1
            3'b010: seg[6:0] <= 7'b0100100;    // digit 2
            3'b011: seg[6:0] <= 7'b0110000;    // digit 3
            3'b100: seg[6:0] <= 7'b0011001;    // digit 4
            3'b101: seg[6:0] <= 7'b0010010;    // digit 5
            3'b110: seg[6:0] <= 7'b0000010;    // digit 6
            3'b111: seg[6:0] <= 7'b1111000;    // digit 7
            default: seg[6:0] <= 7'b0001110; // digit F
        endcase
    end
endmodule
```

8 Xilinx Design Constraints

1. 1 clock pin used to access the internal clock(inclk) of the fpga board
2. 3 Switches were used to set the Johnson reset,clock reset and input clock
3. 3 LEDs were used to show the counter output qs[3] in binary .

4. 4'b0001 supplied as digits[4] to glow only one seven segment display as common anode
5. seg_out[7] used to get output in seven segment display

```
# constraints.xdc

## 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 { johnson_rst }];#LSB
set_property -dict { PACKAGE_PIN L4    IOSTANDARD LVCMOS33 } [get_ports { clock_rst }];
#set_property -dict { PACKAGE_PIN M4    IOSTANDARD LVCMOS33 } [get_ports { in_clk }];


## LEDs
set_property -dict { PACKAGE_PIN J3    IOSTANDARD LVCMOS33 } [get_ports { qs[0] }];#LSB
set_property -dict { PACKAGE_PIN H3    IOSTANDARD LVCMOS33 } [get_ports { qs[1] }];
set_property -dict { PACKAGE_PIN J1    IOSTANDARD LVCMOS33 } [get_ports { qs[2] }];

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

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

9 Comments

9.1 Debugging

9.1.1 Displaying 7-segment display

Initially the seven segment display wasn't lighting up. We soon realized that this was because we didn't give the display common anode voltage high. Once we rectified this we were able to view the 7 segment display properly.

9.1.2 Clock divider issue

We were originally dividing the clock by a very small amount, and were not able to see the output properly, once we figured out the proper ratio for dividing the clock, the output became stable to see by us.