EE2016 - Experiment 2

Nishanth Senthil Kumar(EE23B049)
Arjun Krishnaswamy (EE23B009)
Gopalkrishna Ramanujam (EE23B029)

14 August 2024

1 Aim of Experiment

- The experiment involves designing a D flip flop module, testing it with a test bench, and then designing a 3 bit Johnson counter using 3 D filp flops. The constructing of the Johnson counter also involves the designing of a clock divider and a decoder module to enable the usage of the 7 Segment display that is present in the FPGA.

2 D Flip Flop

2.1 Overview

- A D Flip Flop is a memory component that stores one bit of data. It is driven by a Clock at the positive or negative edge trigger depending on how the flip flop is built.
- It has an input data line, and the D flip flop stores the value that is present in this data line when the clock is at the activating trigger position.
- The D flip flop also has a reset input, which resets the data stored to 0, and the complement of it to 1. The code written uses a synchronous reset pin.

2.2 Code

```
module dflipflop_withreset(q, qbar, d, rst, clk);
output q,qbar;
input d, rst, clk;
reg q,qbar;

always @ (posedge clk)
begin

    if (~rst)

    //assigning the flip flop values
    begin

        q=d;
        qbar=~d;
    end

    else

    //reset conditons when the rest bit is set to 0.
    begin
        q=1'b0;
```

```
        qbar=1'b1;
    end
end
endmodule
```

2.3 Test Bench

```
module dff_testbench;

    reg d,rst,clk;
    wire q,qbar;
    dflipflop_withreset Dut(.q(q),.qbar(qbar),.d(d),.rst(rst),.clk(clk));
    always #5 clk=~clk;
    initial begin
        #0
        d=1'b0;
        rst=1'b0;
        clk=0;
        #10
        d=1'b1;

        #20
        rst=1'b1;
        #10
        $finish;

    end

    //displaying the output onto serial monitor to verify.
    initial begin
        $monitor("Time=%0d, q=%b,qbar=%b",$time,q,qbar);
    end


endmodule
```

2.4 Waveform Output of the D Flip Flop

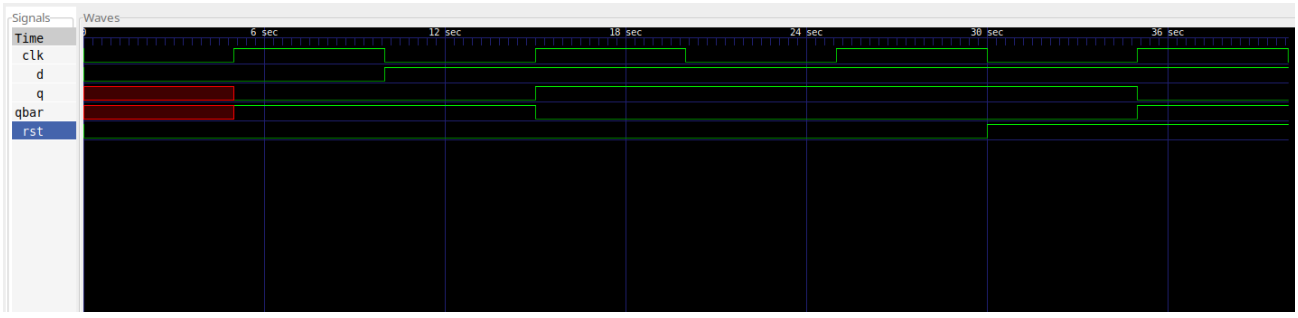

Figure 1: Waveform of Output of D flip flop

3 Johnson Counter

3.1 Overview

- A Johnson counter is a type of digital counter that cycles through a unique sequence of states.
- In Johnson counter, the complemented output of last flip flop is connected to input of first flip flop and to implement n-bit Johnson counter we require n flip-flop.
- It is formed by the feedback of the output to its own input.

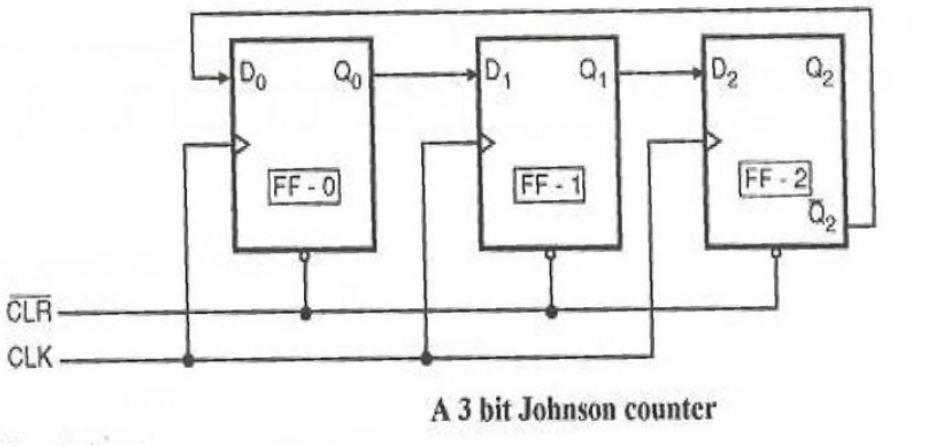

Figure 2: Johnson Counter

- If n flip flops are used, then the number of states in the counter is 2n.
- The counter starts from state 000, then the states are such that it goes from

$$000 \rightarrow 100 \rightarrow 110 \rightarrow 111 \rightarrow 011 \rightarrow 001 \rightarrow 000$$

and the cycle repeats.

3.2 Clock Divider

- We planned to use the FPGA inbuilt clock, that can be accessed through pin N11 to drive our D Flip Flops.
- The FPGA clock frequency works at 50 MHz, and if we used this frequency to drive our clocks, the change of states will be very rapid, faster than we can see the states change.
- Hence, there is a requirement to "slow" down the clock to frequencies of the Hz range, to ensure we can perceive the changes in state and observe it on the 7 Segment Display.
- The clock divider module takes in the FGPA clock as an input, and gives a clock with lower frequency as the output. It also takes a reset bit as an input, which is the same reset bit that is used to reset the D Flip Flops.
- The parameter v can be varied to generate different clock frequencies from the master clock.
- Non-Blocking assignments are used to ensure proper functionality of the system without ambiguities.
- Since our parameter v is set to 25×10^6 MHz and clock frequency is 50×10^6 MHz, the output frequency will be 1 Hz.

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

//counter of 27 bits to store up to the value of v
reg [27:0]clockCount;

//parameter v is the value you want to divide the frequency by
parameter v = 25000000;


always @(negedge reset or posedge inClk)

    begin

    //reset condition, resets everything to 0
    if (reset == 1'b0)
    begin

        clockCount<=0;
        outClk<=1'b0;
        v<=0;
    end

    else

    begin
        clockCount<=clockCount+1;

        //resetting the clock count when it counts upto v
        if (clockCount==v)
        begin
            clockCount<=0;
            outClk <= ~outClk;
        end

    end

end
endmodule
```

3.3 Decoder

- To display the output number onto the 7 segment display, we have to decode the 3 or 4 bit number into a form that can be given as input to the 8 data lines of the 7 segment display (One data line is used to control the decimal point).
- This is similar to the function of the 7447 IC that we used in our Digital Systems (EE2001) course in the second semester.
- The state of the 3 flip flops is computed in the Johnson counter module (read under "Implementing the Johnson Counter" section), and the output of that is sent to this module.
- The FPGA uses common anode LED 7 segment displays, so we have to decode whatever segments we want

to light up for a particular number, and sent the complement of that to the data lines of the 7 segment display.

- The line 'assign Seven_Seg = 1'b1, val;' does the required complememting to cater for the common anode configuration of the display. The 1'b1 that is appended to the front of the vector is used to control the Decimal point of the LED display. We keep the decimal

3.3.1 Code

```
module decoder(cntr,Seven_Seg);
input [3:0] cntr;
output [7:0] Seven_Seg;
reg [6:0] val;
//assigning the decoded and decimal point values
assign Seven_Seg = {1'b1,~val};
//different decoding cases

always@(cntr)
begin
case(cntr)
    4'd0: val =7'b0111111;
    4'd1: val =7'b0000110;
    4'd2: val =7'b1011011;
    4'd3: val =7'b1001111;
    4'd4: val =7'b1100110;
    4'd5: val =7'b1101101;
    4'd6: val =7'b1111101;
    4'd7: val =7'b0000111;
    4'd8: val =7'b1111111;
    4'd9: val =7'b1001111;
    default: val = 7'b0000000;
endcase
end
endmodule
```

3.4 Implementing the Johnson Counter

3.4.1 Code

```
module johnson3bit(Seven_Seg,inClk,rst);

input inClk, rst;
output [7:0] Seven_Seg;
wire [2:0] cntr;
wire q0,q1,q2;
wire q2bar, q1bar, q0bar;
wire out_clk;

//instantiating the clock divider and decoder modules
clk_divider cd0(.inClk(inClk),.reset(rst),.outClk(outClk));
decoder dec0(.cntr(cntr),.Seven_Seg(Seven_Seg));

//d0,d1 and d2 are the 3 d flip flop instantiations
dff d0(q0, q0bar, q1, rst, outClk);
dff d1(q1, q1bar, q2, rst ,outClk);
```

```
dff d2(q2, q2bar, q0bar, rst, outClk);

//assigning the present state to cntr, will be sent to decoder module
assign cntr = {q2, q1 ,q0};
```

3.4.2 Explanation

- We assign 'Seven_Seg' as an output. 'InClk' and 'rst' are our inputs.
- We instantiate our clock divider module and convert 'InClk' to 'OutClk'. The module increments a counter till it reaches a the value that we want to divide 'InClk' by. 'OutClk' is then complemented, giving us a divided clock frquency.
- We define a 3 bit register 'cntr' and proceed to construct a 3 bit counter by instantiating out d flip flops in conjunction.
- Our counter architecture is acccording to the documentation provided with each flip flop receiving its input from the previous flip flop's output.
- 'cntr' stores the ordered output of 'dff2', 'dff1' and 'dff0' and then is sent as an input to our decoder module. This is essentially what we want to display on the Seven Segment module in the decimal form
- The decoder module maps the received 'cntr' value against the decimal equivalent and the corresponding LED sequence for the seven segment display has been coded. Thus we receive an output 'Seven_Seg' which stores the appropriate LED 'HIGH' or 'LOW' states.

  The line 'assign Seven_Seg = 1'b1, val;' is used to turn off the LED that controls the decimal point
- 'digit' is used to configure the set of four seven segment displays on the FPGA and select which display is to be used.

4 Programming and Debugging Experience

- Our compile time error was defining 'cntr' in the Johnson counter module as an output. It is not an output but in fact just an intermediary to program our decoder. We corrected this by declaring it as a wire, instead of a reg.
- We initially did not implement the use of 'digit'. The digit vector is required to activate the 7 Segment Display, we did not do that and had to correct that.
- After recitfying the above two errors, bit-stream generation was successful. However, when operated, our FPGA Seven Segment display statically kept displaying the digit '0'. On inspection we realised that the counter that we had defined in our clock divider module was only 4 bits large. Thus it never reached setpoint divisor value and an incorrect clock signal was supplied to the flip flops.
- The reason why we declared it as a 4 bit vector was that we were testing it locally for small values on our system to check if the module works. We forgot to change it back to a 28 bit vector.
- On correcting these, an accurate Johnson Counter sequence was displayed on our display.

5 References

- Moodle Verilog notes
- Moodle reference material for Experiment 2
- GeeksforGeeks: N-bit Johnson Counter in Digital Logic
- Decoding Logic for 7 segment display