Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

3C7 DIGITAL SYSTEMS DESIGN LABORATORY

Lab F Report

Department of Electronic and Electrical Engineering

Professor: Dr. Shreejith Shanker

1. Abstract:

The 3C7 Digital Systems Design report involves designing and testing a Sequential Logic Circuits in Vivado, and implementing it on the Basys-3 Board. For an understanding of the development and testing of sequential designs in digital systems, this study synthesizes knowledge from lectures and previous year basic electronics knowledge. The design must generate waveforms showing the various sequential circuit logic like in case of D – Flip Flop taking clock, reset, and d as inputs and generating resulting waveform showing output Q. The report emphasizes the importance of understanding and applying digital design principles in a practical context to implement a suitable FPGA design.

Submitted By:
Shreshtha Kamboj
23364317
Submission Date: 23/03/2024

2. Introduction:

This assignment's objectives were to gather all the knowledge from previous year modules and current lectures to design a Verilog module to implement sequential circuits logic. Sequential circuits are the ones in which the output not only depends on the current input but also previous inputs. This shows that these circuits have a memory feature and store information. In this report we closely look at two types of sequential circuits which are synchronous and asynchronous. We design both testbench and top module to examine these circuits.

3. Implementation:

We implement DFFs and LFSR on the Basys 3 Board for the first-time using clocks. We also use the buttons on the Basys 3 Board in this lab.

4. Sources:

4.1 Lab F part A:

The Design sources consists of testbench_labF module which inherits the property of d_type_ff module by instantiation it as shown below:

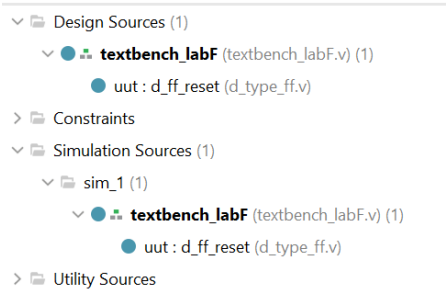


Fig 4.1 Design Sources, Simulation Sources for part A

File directory is as follows:

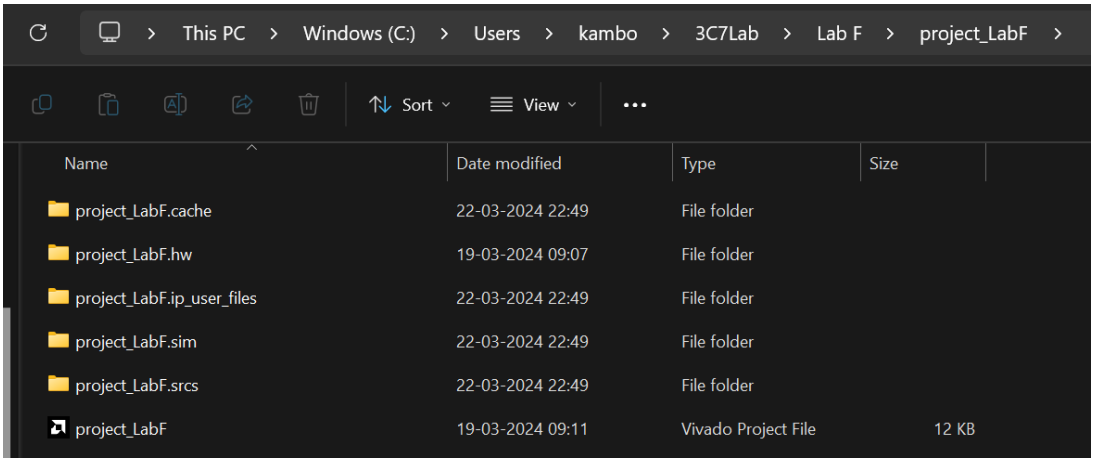


Fig 4.2 File Directory for part A

4.2 Lab F part B:

The Design sources consists of toplevel module which inherits the property of d_type_ff module, sevenseg module, and debouncer module by instantiation it as shown below:

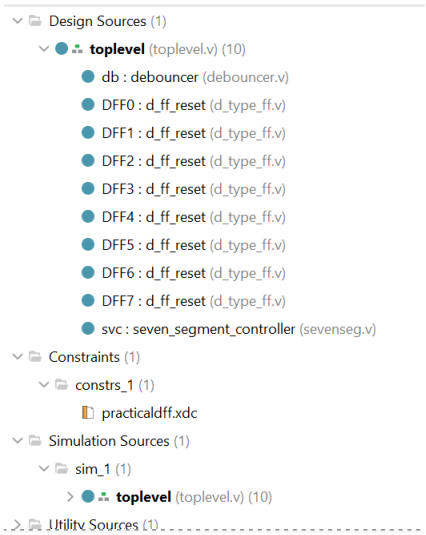


Fig 4.3 Design Sources, Simulation Sources, Constraints for part B

File directory is as follows:

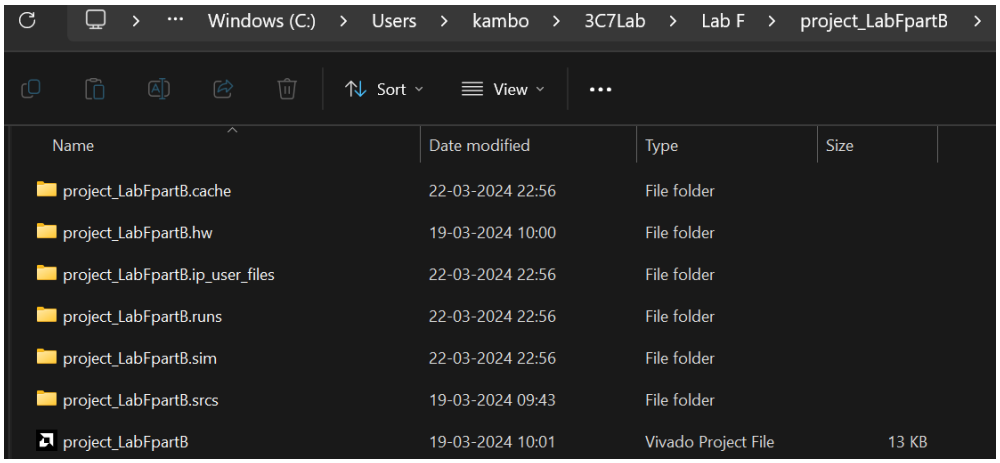


Fig 4.4 File Directory for part B

4.3 Lab F part C:

The Design sources consists of lfsr_13bit_tb module which inherits the property of lfsr_13bit module, and counter module by instantiation it as shown below:

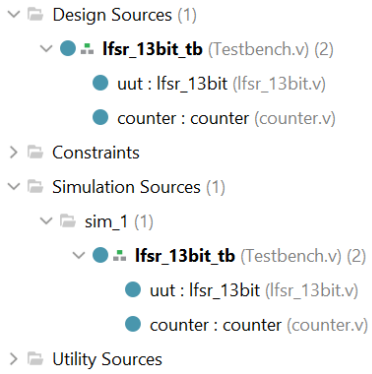


Fig 4.5 Design Sources, Simulation Sources for part C

File directory is as follows:

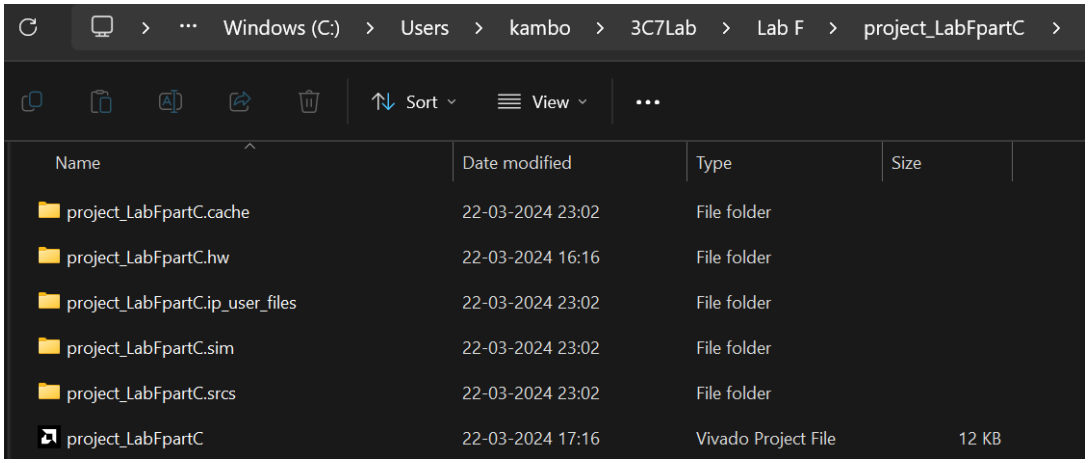


Fig 4.6 File Directory for part C

4.4 Lab F part D:

The Design sources consists of top_module module which inherits the property of clock_divider module, and lfsr_13bit module by instantiation it as shown below:

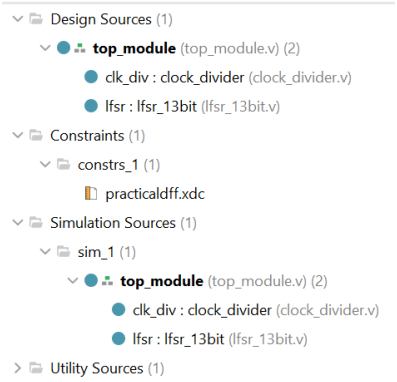


Fig 4.7 Design Sources, Simulation Sources for part D

File directory is as follows:

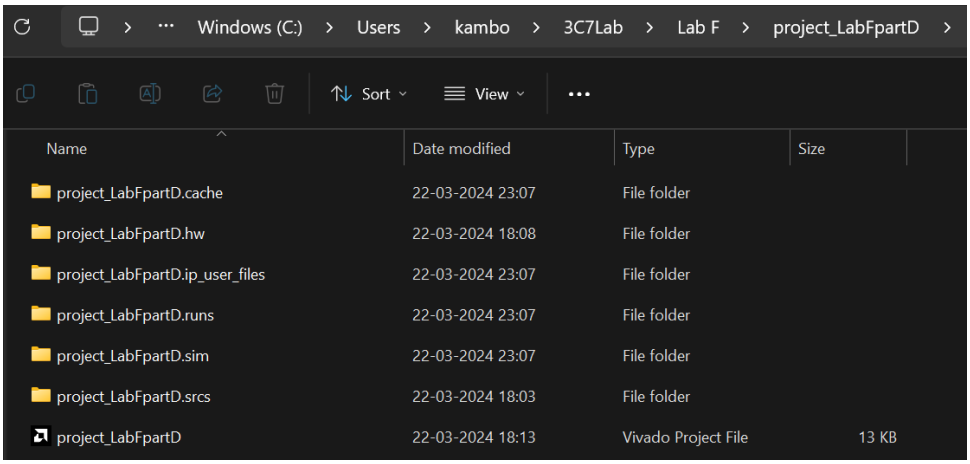


Fig 4.8 File Directory for part D

5. Schematics Generated:

5.1 Lab F part A:

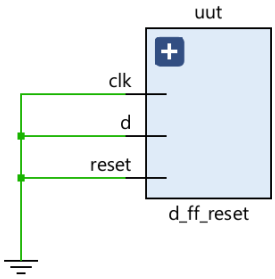


Fig 5.1 Elaborated Design for part A

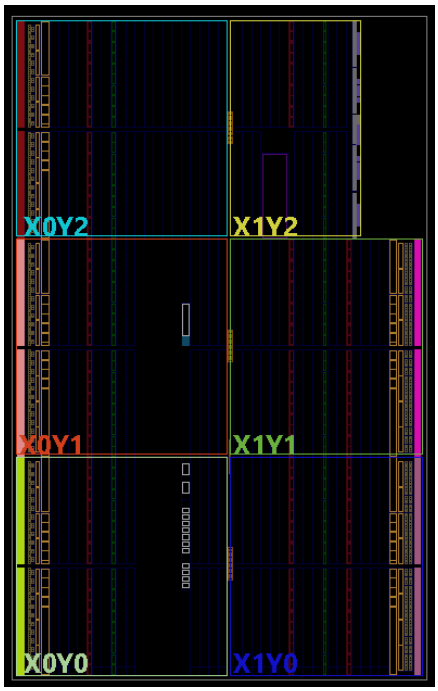


Fig 5.2 Implemented Device for all parts

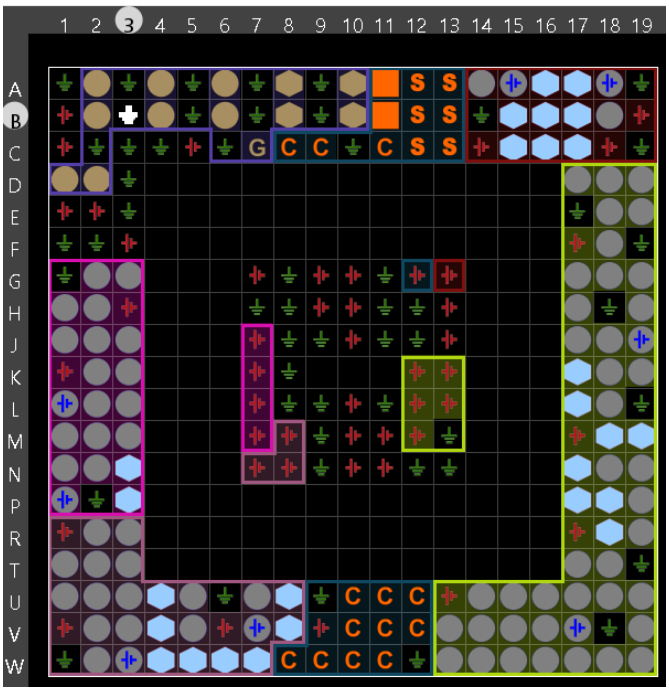


Fig 5.3 Implemented Package for all parts

5.2 Lab F part B:

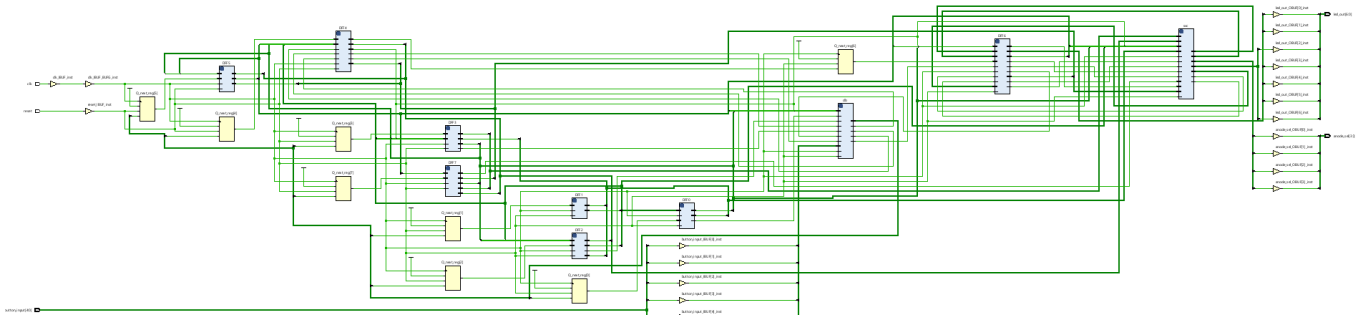


Fig 5.4 Elaborated Design for part B

5.3 Lab F part C:

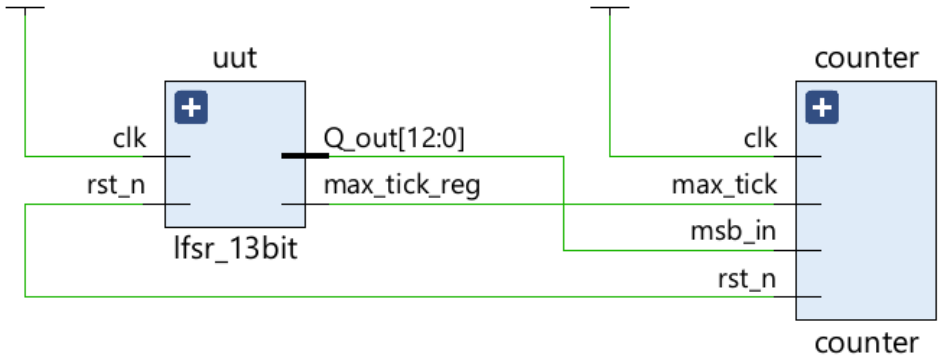


Fig 5.5 Elaborated Design for part C

5.4 Lab F part D:

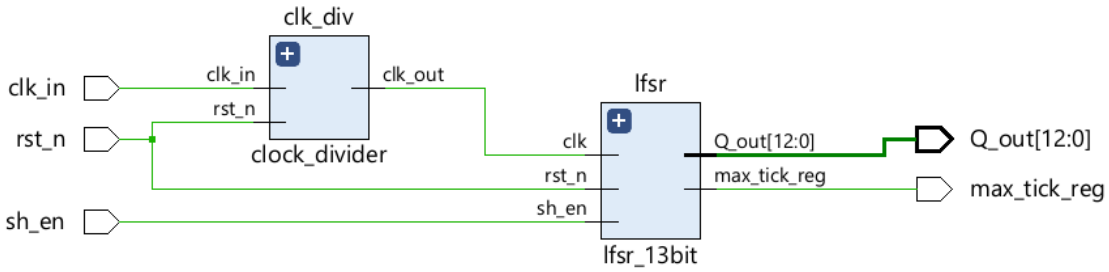


Fig 5.6 Elaborated Design for part D

6. Code Snippets:

We have included the following modules in this report to correctly implement the given task for each part of the Lab:-

6.1 Lab F part A:

C:/Users/kambo/3C7Lab/Lab F/d_type_ff.v

```
module d_ff_reset(
    input wire clk, reset,
    input wire d,
    output reg q);
   // Code for asynchronous reset & clock
   always @(posedge clk, posedge reset)     // For negative edge we can use negedge
      if (reset)
         q <= 1'b0;
      else
         q <= d;
// // Code for synchronous reset & clock
//    always@(posedge clk, posedge reset)
//    begin
//        if(reset)
//        begin
//            q = 1'b0;
//        end
//        else
//        begin
//            q = d;
//        end
//   end
endmodule
```

Fig 6.1 Module having code for a D type Flip Flop

C:/Users/kambo/3C7Lab/Lab F/project_LabF/project_LabF.srcs/sources_1/new/textbench_labF.v

```
`timescale 1ns / 10ps
module textbench_labF();
    reg clk, reset, d;
    wire q;
    parameter T=20;
    d_ff_reset uut (.clk(clk), .reset(reset), .d(d), .q(q));
    // Giving clock the period 10ns
    always
    begin
        clk = 1'b1;
        #(T/2);
        clk = 1'b0;
        #(T/2);
    end
    // Giving reset the period 40ns and 60ns
    always
    begin
        reset = 1'b1;
        #(2*T);
        reset = 1'b0;
        #(3*T);
        reset = 1'b1;
        #(2*T);
        reset = 1'b0;
        #(3*T);
    end
    // Giving d input the period 10ns, 60ns, 40ns, 100ns, and 40ns
    initial
    begin
        d = 1'b0;
        #T;
        d = 1'b1;
        #(3*T);
        d = 1'b0;
        #(2*T);
        d = 1'b1;
        #(5*T);
        d = 1'b0;
        #(2*T);
    end

endmodule
```

Fig 6.2 Testbench Module for part A

6.2 Lab F part B:

C:/Users/kambo/3C7Lab/Lab F/sevenseg.v

```
module seven_segment_controller(
    input clk, // 100 Mhz clock source on Basys 3 FPGA
    input reset, // reset
    input [7:0] temp,
    output reg [3:0] anode_select, // select one of the 4 7-segment modules by choosing one to be activated - note that this is
    output reg [6:0] LED_out);// cathode patterns of the 7-segment LED display
    reg [3:0] LED_BCD;
    reg [19:0] refresh_counter; // 20-bit for creating 10.5ms refresh period or 380Hz refresh rate
            // the first 2 MSB bits for creating 4 LED-activating signals with 2.6ms digit period
    wire [1:0] LED_activating_counter;
                // count     0    ->  1  ->  2  ->  3
            // activates    LED1    LED2    LED3    LED4
    always @(posedge clk or posedge reset)begin
        if(reset==1)
            refresh_counter <= 0;
        else
            refresh_counter <= refresh_counter + 1;
    end
    assign LED_activating_counter = refresh_counter[19:18];
    // anode activating signals for 4 LEDs, digit period of 2.6ms
    // decoder to generate anode signals
    always @(*)
    begin
        case(LED_activating_counter)
        2'b00: begin
            anode_select = 4'b0111;
            // activate LED1 and Deactivate LED2, LED3, LED4
            LED_BCD = temp/100;
             // the first digit of the 8-bit temperature value
             end
        2'b01: begin
            anode_select = 4'b1011;
            // activate LED2 and Deactivate LED1, LED3, LED4
            LED_BCD = (temp%100)/10;
            // the second digit of the 8-bit temperature value
             end
        2'b10: begin
            anode_select = 4'b1101;
            // activate LED3 and Deactivate LED2, LED1, LED4
            LED_BCD = (temp%100)%10;
            // the last digit of the 8-bit temperature value
                end
        2'b11: begin
            anode_select = 4'b1110;
            // activate LED4 and Deactivate LED2, LED3, LED1
            LED_BCD = 4'hF;
            // F symbol to indicate Fahrenheit
                end
        endcase
    end
    // Cathode patterns of the 7-segment LED display
    always @(*)
    begin
        case(LED_BCD)
        4'b0000: LED_out = 7'b0000001; // "0"
        4'b0001: LED_out = 7'b1001111; // "1"
        4'b0010: LED_out = 7'b0010010; // "2"
        4'b0011: LED_out = 7'b0000110; // "3"
        4'b0100: LED_out = 7'b1001100; // "4"
        4'b0101: LED_out = 7'b0100100; // "5"
        4'b0110: LED_out = 7'b0100000; // "6"
        4'b0111: LED_out = 7'b0001111; // "7"
        4'b1000: LED_out = 7'b0000000; // "8"
        4'b1001: LED_out = 7'b0000100; // "9"
        4'b1111: LED_out = 7'b0111000; // "F"
        default: LED_out = 7'b0000001; // "0"
        endcase
    end
endmodule
```

Fig 6.3 Module having code for seven segment

C:/Users/kambo/3C7Lab/Lab F/debouncer.v

```
module debouncer
(input clk, // 100 Mhz clock source on Basys 3 FPGA
    input reset, // reset
    input [4:0] button_in,
    output reg [4:0] button_out
);
localparam threshold = 24'hFFFFFF;
reg [26:0] counter;
reg [4:0] button_d1, button_d2;
always @ (posedge clk)
begin
    button_d1 <= button_in;
    button_d2 <= button_d1;
end
always @(posedge clk or posedge reset)
    begin
        if(reset==1) begin
            counter <= 1;
            button_out <= 0;
        end
        else begin
            button_out <= 0;
            if (|button_d2) begin
                if(~&counter)
                    counter <= counter + 1;
            end
            else begin
                if (|counter)
                    counter <= counter - 1;
            end
            if (counter > threshold) begin
                button_out <= button_d2;
                counter <= 0;
            end
        end
    end
endmodule
```

Fig 6.4 Module having code for debouncer

C:/Users/kambo/3C7Lab/Lab F/project_LabFpartB/project_LabFpartB.srcs/sources_1/new/toplevel.v

```
`timescale 1ns / 1ps
module toplevel(
    input clk, // Clock signal
    input reset, // Reset signal
    input [4:0] button_input, // 5-bit input for buttons
    output [3:0] anode_sel, // 4-bit output for anode selection in a 7-segment display
    output [6:0] led_out); // 7-bit output for LED segments
    wire UP, LEFT, DOWN, RIGHT, CENTRE;// Wire declarations for button states
    wire [7:0] Q;// 8-bit wire for storing the current state
    reg [7:0] Q_next;// 8-bit register for storing the next state
    wire [4:0] buttons;// Wire for debounced button inputs
    // Instantiate a debouncer module for button inputs
    debouncer db(.clk(clk), .reset(reset), .button_in(button_input), .button_out(buttons));
    // Instantiate flip-flops with reset for each bit of Q
    d_ff_reset DFF0(.clk(clk), .reset(reset), .d(Q_next[0]), .q(Q[0]));
    d_ff_reset DFF1(.clk(clk), .reset(reset), .d(Q_next[1]), .q(Q[1]));
    d_ff_reset DFF2(.clk(clk), .reset(reset), .d(Q_next[2]), .q(Q[2]));
    d_ff_reset DFF3(.clk(clk), .reset(reset), .d(Q_next[3]), .q(Q[3]));
    d_ff_reset DFF4(.clk(clk), .reset(reset), .d(Q_next[4]), .q(Q[4]));
    d_ff_reset DFF5(.clk(clk), .reset(reset), .d(Q_next[5]), .q(Q[5]));
    d_ff_reset DFF6(.clk(clk), .reset(reset), .d(Q_next[6]), .q(Q[6]));
    d_ff_reset DFF7(.clk(clk), .reset(reset), .d(Q_next[7]), .q(Q[7]));
    // Assign button states to corresponding wires
    assign UP = buttons[0];
    assign LEFT = buttons[1];
    assign DOWN = buttons[2];
    assign RIGHT = buttons[3];
    assign CENTRE = buttons[4];
    always @(posedge clk or posedge reset) begin// Sequential logic to update Q_next based on button inputs
        if (reset) begin
            Q_next <= 8'b0; // Reset Q_next to 0
        end else begin
            if(UP == 1'b1 || RIGHT == 1'b1) begin
                Q_next <= Q + 8'b00000001; // Increment Q if UP or RIGHT is pressed
            end else if(DOWN == 1'b1 || LEFT == 1'b1) begin
                Q_next <= Q - 8'b00000001; // Decrement Q if DOWN or LEFT is pressed
            end else if(CENTRE == 1'b1) begin
                Q_next <= 8'b00010110; // Set Q to 22 if CENTRE is pressed
            end else begin
                Q_next <= Q; // Keep Q unchanged if no buttons are pressed
            end
        end
    end
    // Instantiate a seven-segment controller module
    seven_segment_controller svc(.clk(clk), .reset(reset), .temp(Q), .anode_select(anode_sel), .LED_out(led_out));
endmodule
```

Fig 6.5 Toplevel Module for part B

6.3 Lab F part C:

C:/Users/kambo/3C7Lab/Lab F/project_LabFpartC/project_LabFpartC.srcs/sources_1/new/counter.v

```
module counter(
        input clk, rst_n, max_tick, msb_in,
        output reg [12:0] ones_count, // Adjusted to 13 bits for the 13-bit LFSR
        output reg [12:0] zeros_count ); // Adjusted to 13 bits for the 13-bit LFSR
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            // Reset the counters when the reset signal is asserted
            ones_count <= 13'b0;
            zeros_count <= 13'b0;
        end else if (max_tick) begin
            // Reset the counters when the LFSR completes a full cycle
            ones_count <= 13'b0;
            zeros_count <= 13'b0;
        end else begin
            // Increment the appropriate counter based on the MSB input
            if (msb_in)
                ones_count <= ones_count + 1;
            else
                zeros_count <= zeros_count + 1;
        end
    end
endmodule
```

Fig 6.6 Module having code for a counter

C:/Users/kambo/3C7Lab/Lab F/project_LabFpartC/project_LabFpartC.srcs/sources_1/new/lfsr_13bit.v

```
`timescale 1ns / 1ps
module lfsr_13bit
    #(parameter seed = 13'h1EE) // Adjusted seed parameter width to match LFSR size
    ( input clk, input rst_n, input sh_en, output reg [12:0] Q_out, output reg max_tick_reg);
    reg [12:0] Q_state;
    wire Q_fb;
    wire [12:0] Q_ns;
    localparam [12:0] max_value = 13'h1FFF; // Maximum value for a 13-bit LFSR
    // Asynchronous active-low reset
    always @ (posedge clk or negedge rst_n) begin // Changed to negedge rst_n
        if (!rst_n) // Active-low reset condition
            Q_state <= seed; // Reset state to seed value
        else if (sh_en)
            Q_state <= Q_ns; // Shift operation

    end
    // Next state logic with feedback function using XNOR
    assign Q_fb = ~(Q_state[12] ^ Q_state[3] ^ Q_state[2] ^ Q_state[0]); // Changed to XNOR
    assign Q_ns = {Q_state[11:0], Q_fb}; // Shift left and insert feedback
    // Output logic
    always @ (posedge clk) begin // Explicitly specify posedge clk
        Q_out <= Q_state; // Update output
        // Check if LFSR reached its maximum value and reset max_tick_reg
        if(Q_state == max_value) begin
            max_tick_reg <= 1'b1; // Set max_tick_reg high when full count is reached
        end else begin
            max_tick_reg <= 1'b0; // Reset max_tick_reg
        end
    end
endmodule
```

Fig 6.7 Module having code for LFSR

C:/Users/kambo/3C7Lab/Lab F/project_LabFpartC/project_LabFpartC.srcs/sources_1/new/Testbench.v

```
`timescale 1ns / 1ps
module lfsr_13bit_tb;
    // Parameters
    parameter CLK_PERIOD = 10;  // Clock period in nanoseconds
    parameter RESET_CYCLES = 10;  // Number of clock cycles for reset
    parameter SIM_DURATION = (2**13)-1; // Duration for one full LFSR cycle
    // Inputs
    reg clk;
    reg rst_n;
    reg sh_en;
    // Outputs
    wire [12:0] Q_out;
    wire max_tick_reg;
    wire [12:0] ones_count;
    wire [12:0] zeros_count;
    // Instantiate the Unit Under Test (UUT)
    lfsr_13bit uut (
        .clk(clk),
        .rst_n(rst_n),
        .sh_en(sh_en),
        .Q_out(Q_out),
        .max_tick_reg(max_tick_reg)
    );
    // Instantiate the bit_counter module
    counter counter (
        .clk(clk),
        .rst_n(rst_n),
        .msb_in(Q_out[12]),
        .max_tick(max_tick_reg),
        .ones_count(ones_count),
        .zeros_count(zeros_count)
    );
    // Clock generation
    always begin
        clk = 1'b0;
        #(CLK_PERIOD/2) clk = 1'b1;
        #(CLK_PERIOD/2);
    end
    // Reset and stimulus generation
    initial begin
        // Initialize Inputs
        rst_n = 1'b0;  // Assert reset
        sh_en = 1'b0;

        // Wait for reset
        repeat(RESET_CYCLES) @(posedge clk);

        rst_n = 1'b1;  // De-assert reset
        sh_en = 1'b1;  // Enable shifting

        // Run for more than one full LFSR cycle to observe the counter restarting
        repeat(2*SIM_DURATION) @(posedge clk);

        // Finish simulation
        $finish;
    end
    // Generate waveform file
    initial begin
        $dumpfile("lfsr_13bit_tb.vcd");
        $dumpvars(0, lfsr_13bit_tb);
    end
    // Monitoring
    initial begin
        $monitor("Time: %0t, Q_out: %b, max_tick_reg: %b, ones_count: %0d, zeros_count: %0d",
                 $time, Q_out, max_tick_reg, ones_count, zeros_count);
    end
endmodule
```

Fig 6.8 Testbench Module for part C

6.4 Lab F part D:

C:/Users/kambo/3C7Lab/Lab F/project_LabFpartD/project_LabFpartD.srcs/sources_1/new/clock_divider.v

```
module clock_divider(
    input clk_in,  // Input clock, assumed to be 50MHz based on the context
    input rst_n,   // Asynchronous reset, active low
    output reg clk_out = 0);  // Output clock, initialized to 0, target is 1Hz after division
// Parameter for defining the division factor. Set to 25,000,000 for a 50MHz clock to achieve 1Hz.
parameter DIVIDE_BY = 25000000; // Division factor to toggle the output clock
// Define a 25-bit counter to count up to 25,000,000
reg [24:0] counter = 0; // Counter variable to store intermediate counts
// Always block triggered on the rising edge of clk_in or the falling edge of rst_n
always @(posedge clk_in or negedge rst_n) begin
    if (!rst_n) begin // If reset is active (low)
        counter <= 0;       // Reset the counter to 0
        clk_out <= 0;       // Reset the output clock to 0
    end else begin
        if (counter == DIVIDE_BY-1) begin // If counter reaches the division value minus 1
            counter <= 0;                 // Reset the counter
            clk_out <= ~clk_out;          // Toggle the output clock
        end else begin
            counter <= counter + 1;       // Increment the counter
        end
    end
end
endmodule
```

Fig 6.9 Module for clock divider

C:/Users/kambo/3C7Lab/Lab F/project_LabFpartD/project_LabFpartD.srcs/sources_1/new/top_module.v

```
module top_module(
    input clk_in,  // Original fast clock (e.g., 50MHz)
    input rst_n,   // Asynchronous reset, active low
    input sh_en,   // Shift enable for LFSR
    output [12:0] Q_out,  // LFSR output
    output max_tick_reg  // Indicates a full LFSR cycle
);
// Instantiate the clock divider
wire slow_clk; // This will be the 1Hz clock
clock_divider clk_div(
    .clk_in(clk_in),
    .rst_n(rst_n),
    .clk_out(slow_clk)
);
// Instantiate the LFSR
lfsr_13bit lfsr(
    .clk(slow_clk),  // Use the 1Hz clock from the clock divider
    .rst_n(rst_n),
    .sh_en(sh_en),
    .Q_out(Q_out),
    .max_tick_reg(max_tick_reg)
);
endmodule
```

Fig 6.10 Toplevel Module for part D

7. Demonstration

Configuration Device: BASYS 3 (xc7a35tcpg236-1)
Target Language: Verilog

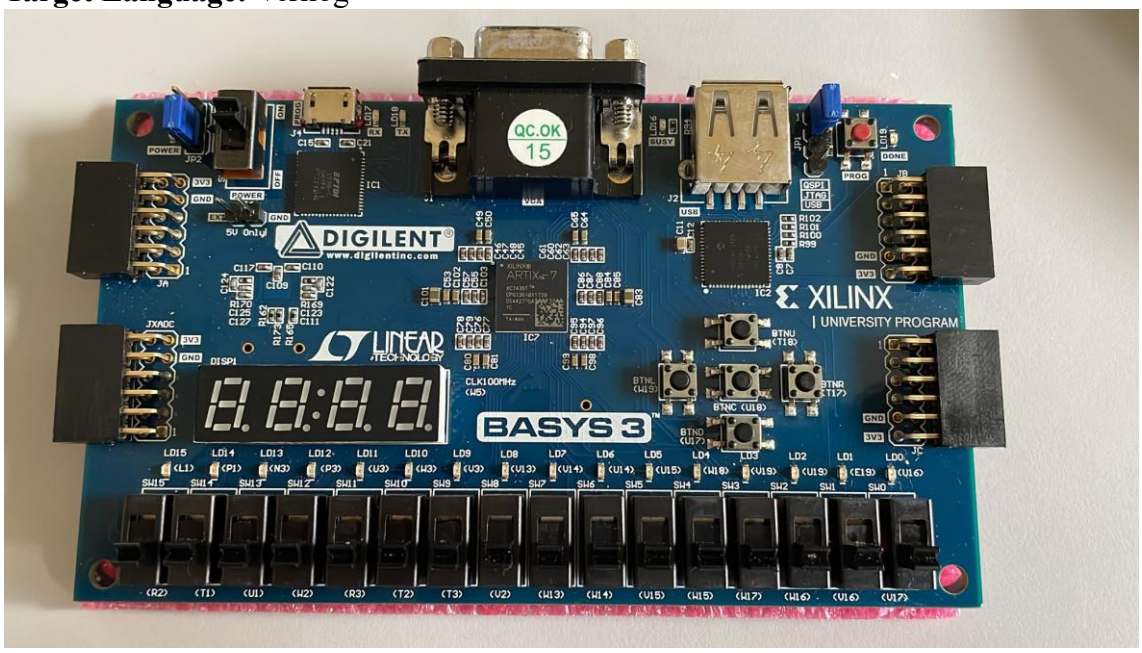


Fig 7.1 Basys 3 Board

V17	reset	W7	led_out[6]	U2	anode_sel [0]
U17	Button_input[0]	W6	led_out[5]	U4	anode_sel [1]
T17	Button_input[1]	U8	led_out[4]	V4	anode_sel [2]
W19	Button_input[2]	V8	led_out[3]	W4	anode_sel [3]
T18	Button_input[3]	U5	led_out[2]	W5	clock
U18	Button_input[4]	V5	led_out[1]	V16	enable
W5	clock	U7	led_out[0]		

Table 7.1 Input switched & Output LEDs

The input is taken using the switches and buttons assigned. State ‘0’ means OFF and state ‘1’ means ON.
The output is generated using LEDs assigned & 7-Segment display. State ‘0’ means OFF and state ‘1’ means ON.

In a synchronous circuit, all behavioural changes occur only on a clock edge. For example, if a button is used to turn on an LED in the circuit, it may only capture a button press on a rising edge.

In an asynchronous circuit, the behaviour of the circuit may change at any time regardless of whether at a clock edge or not.

7.1 Lab F part A:

In this part of lab, we designed to test out the code for D Flip Flop on the Testbench. A D flip-flop is a kind of digital storage element that is part of the sequential logic circuit series. It is also referred to as a data or delay flip-flop. It functions as a key building element in digital electronics for memory and data storage applications and has two stable states. It is used to store binary data. It has 2 inputs which are the clock ‘clk’ and data input ‘d’. It also has an optional input which is ‘reset’. We get only one output which is ‘Q’. Here we also write a testbench to generate waveforms for positive/negative edge in case of asynchronous type and test it for a synchronous type.

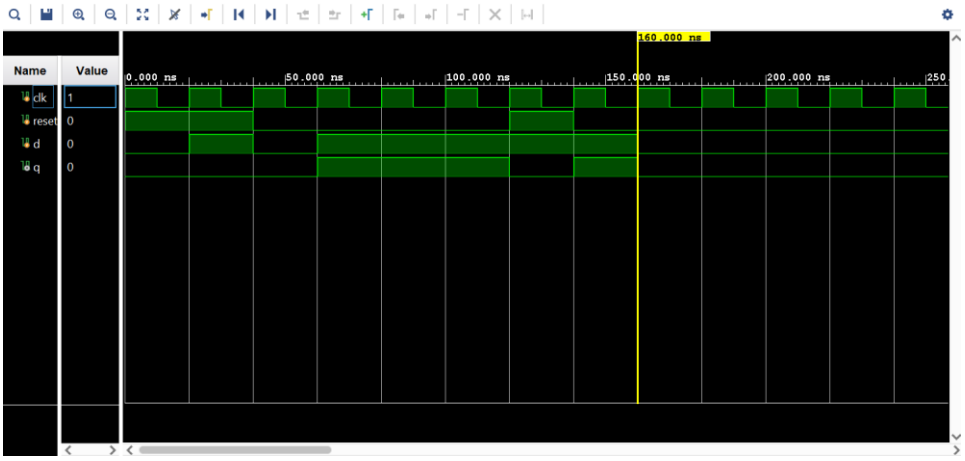


Fig 7.2 Output for positive edge

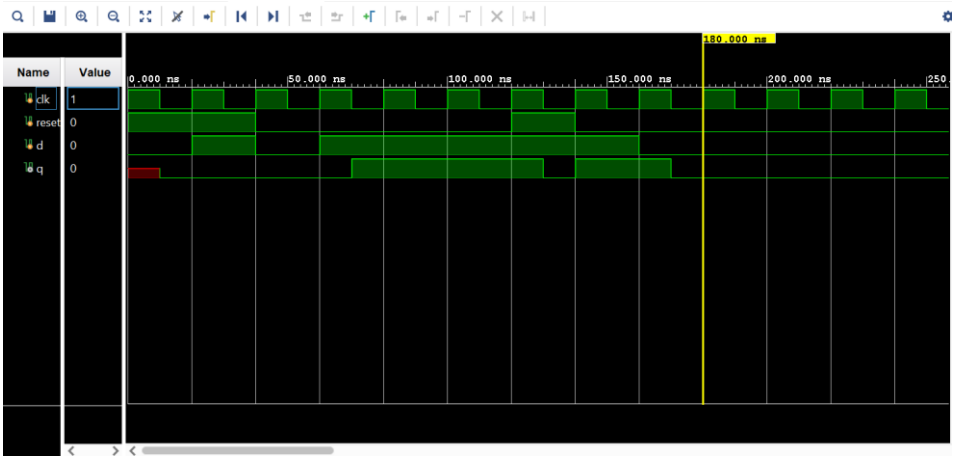


Fig 7.3 Output for negative edge

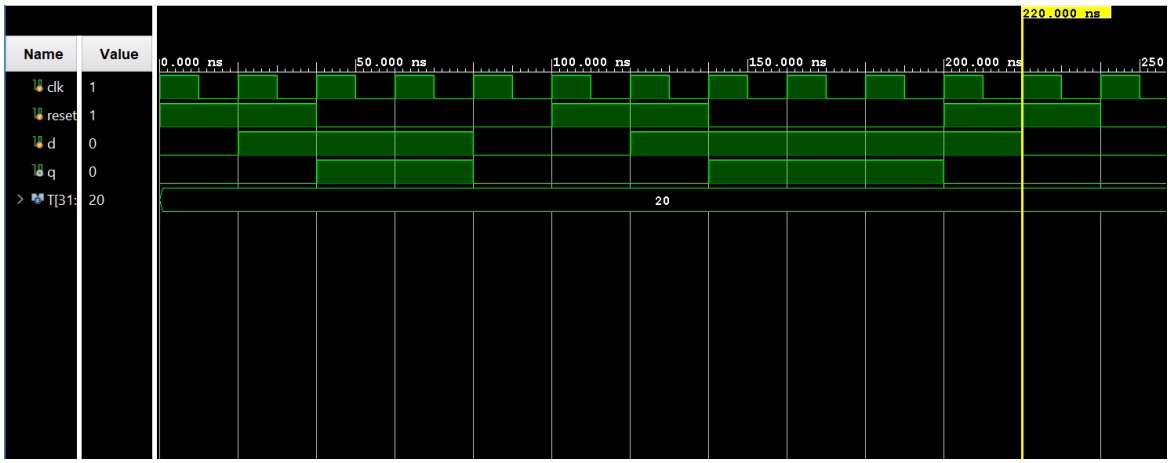


Fig 7.4 Output for synchronous reset

7.2 Lab F part B:

In this part of the lab, we use multiple DFFs to save the current state of a value and use the left and right push buttons to increase or decrease value displayed on the seven-segment display on the board. We also have used a debouncer which is used to stabilize the output of mechanical buttons and switches. When a mechanical switch or button is pressed or released, it frequently makes and breaks contact multiple times in a brief amount of time, a phenomenon called “bouncing.” We have made a topmodule which has functionality to implement the action on board when buttons are pressed by the change of number on 7-segment display. In the xdc file we have all the variables mapped to ports on the board.

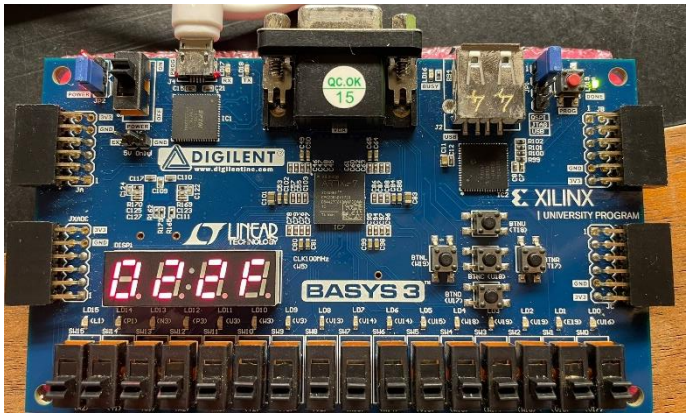


Fig 7.5 Initial output on 7-segment

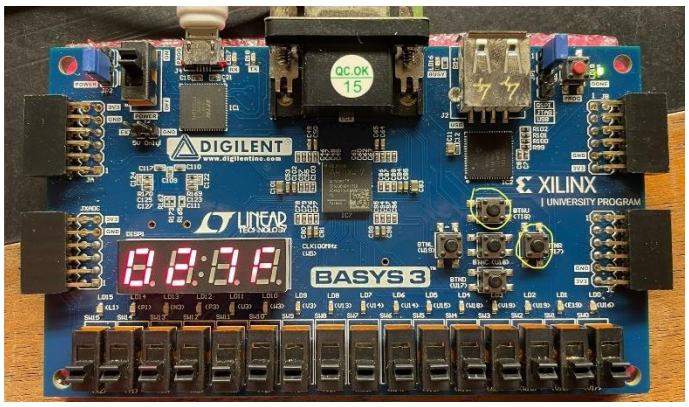


Fig 7.6 Right/Top button increments output on 7-segment

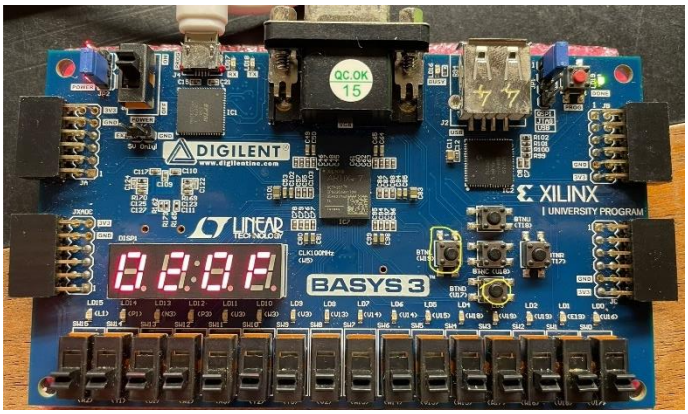


Fig 7.7 Left/Bottom button decrements output on 7-segment

7.3 Lab F part C:

In this part of the lab, we are required to implement the maximal length, i.e. $2^N - 1$, LFSR counter in Verilog. A Linear Feedback Shift Register is a specialised case of synchronous shift-register. The input to shift-register is some combination of its stages and it cycles through sequences of pseudo-random numbers based on the feedbacks chosen. It offers speed, area, and performance advantages. Feedback logic consists of taking XNOR output of selected taps into the input. One thing to note is that XNOR feedback cannot create all '1' state. In our case we have maximal length of 13 bits and according to datasheet provided of Xilinx we must take taps on the bits 13, 4, 3, 1. We also give a seed value which is the XNOR of our board number and the last 3 digits of roll number.

Board Number: 44 (binary - 0000000101100)
Last 3 digits of Roll Number: 317 (binary - 0000100111101)
Seed Value (XNOR): 1111011101110 (hex value - 1EE)

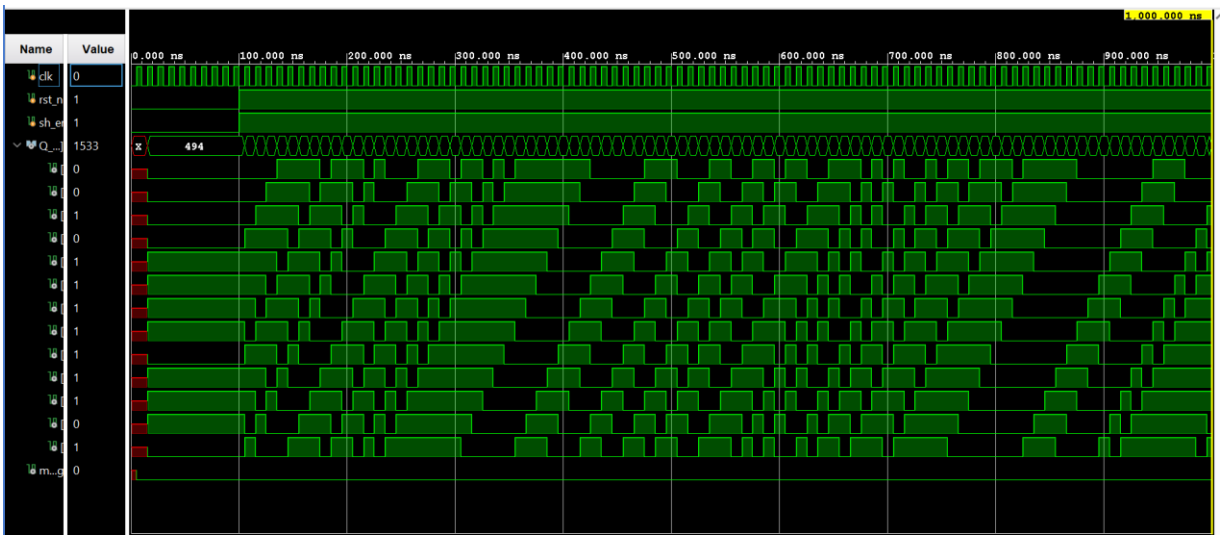


Fig 7.8 Waveform showing the output of each bit

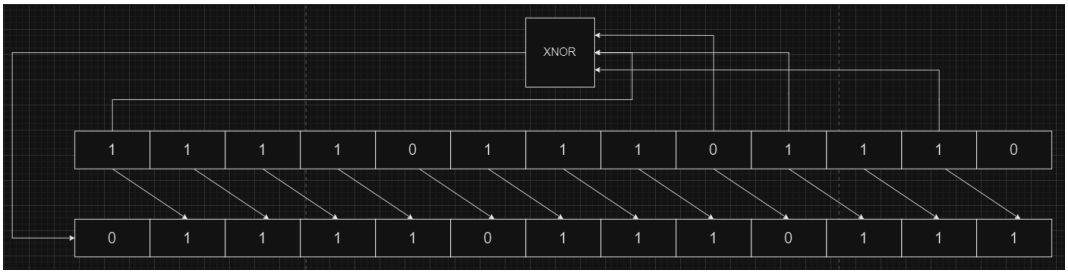


Fig 7.9 Block Diagram showing all 13 bits

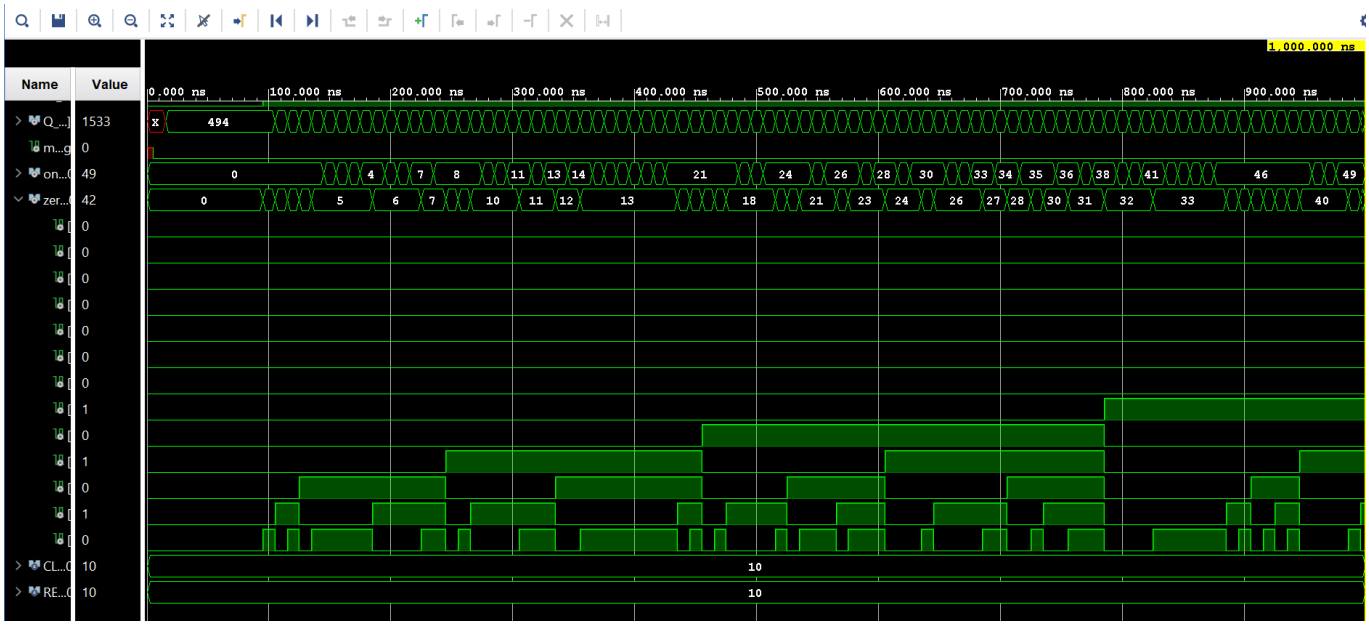


Fig 7.10 Waveform showing the number of zeroes we are counting

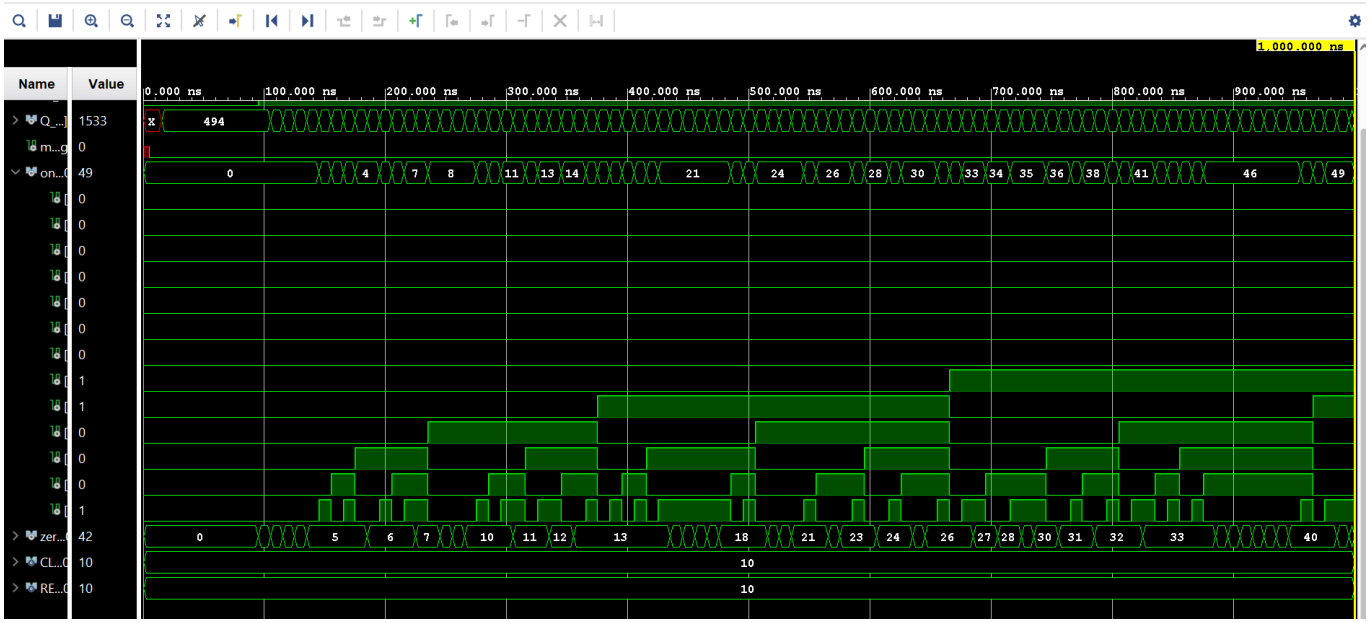
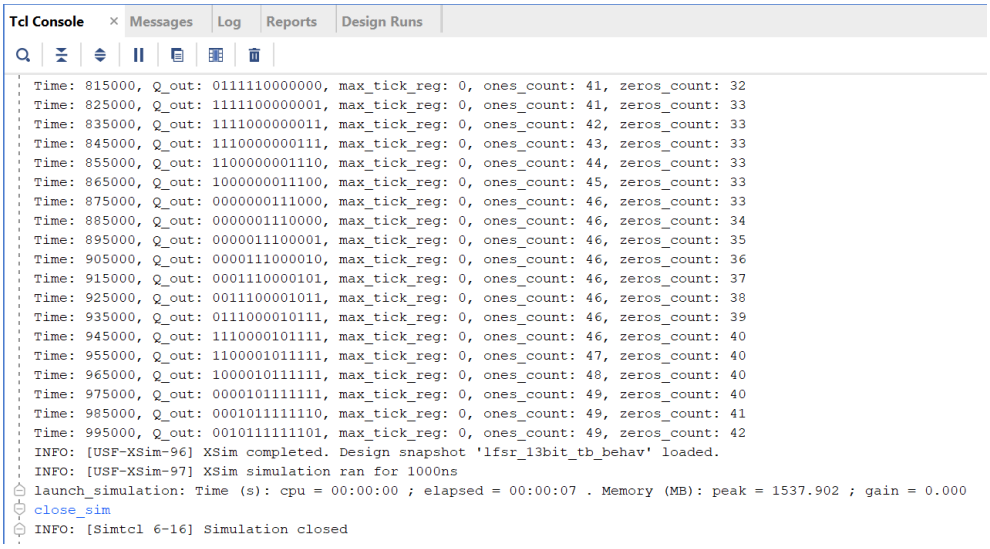


Fig 7.11 Waveform showing the number of ones we are counting



```
Tcl Console   Messages   Log   Reports   Design Runs
Time: 815000, Q_out: 0111110000000, max_tick_reg: 0, ones_count: 41, zeros_count: 32
Time: 825000, Q_out: 1111100000001, max_tick_reg: 0, ones_count: 41, zeros_count: 33
Time: 835000, Q_out: 1111000000011, max_tick_reg: 0, ones_count: 42, zeros_count: 33
Time: 845000, Q_out: 1110000000111, max_tick_reg: 0, ones_count: 43, zeros_count: 33
Time: 855000, Q_out: 1100000001110, max_tick_reg: 0, ones_count: 44, zeros_count: 33
Time: 865000, Q_out: 1000000011100, max_tick_reg: 0, ones_count: 45, zeros_count: 33
Time: 875000, Q_out: 0000000111000, max_tick_reg: 0, ones_count: 46, zeros_count: 33
Time: 885000, Q_out: 0000001110000, max_tick_reg: 0, ones_count: 46, zeros_count: 34
Time: 895000, Q_out: 0000011100001, max_tick_reg: 0, ones_count: 46, zeros_count: 35
Time: 905000, Q_out: 0000111000010, max_tick_reg: 0, ones_count: 46, zeros_count: 36
Time: 915000, Q_out: 0001110000101, max_tick_reg: 0, ones_count: 46, zeros_count: 37
Time: 925000, Q_out: 0011100001011, max_tick_reg: 0, ones_count: 46, zeros_count: 38
Time: 935000, Q_out: 0111000010111, max_tick_reg: 0, ones_count: 46, zeros_count: 39
Time: 945000, Q_out: 1110000101111, max_tick_reg: 0, ones_count: 46, zeros_count: 40
Time: 955000, Q_out: 1100001011111, max_tick_reg: 0, ones_count: 47, zeros_count: 40
Time: 965000, Q_out: 1000010111111, max_tick_reg: 0, ones_count: 48, zeros_count: 40
Time: 975000, Q_out: 0000101111111, max_tick_reg: 0, ones_count: 49, zeros_count: 40
Time: 985000, Q_out: 0001011111110, max_tick_reg: 0, ones_count: 49, zeros_count: 41
Time: 995000, Q_out: 0010111111101, max_tick_reg: 0, ones_count: 49, zeros_count: 42
INFO: [USF-XSim-96] XSim completed. Design snapshot 'lfsr_13bit_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:00 ; elapsed = 00:00:07 . Memory (MB): peak = 1537.902 ; gain = 0.000
close_sim
INFO: [Simtcl 6-16] Simulation closed
```

Fig 7.12 TCL Console output

7.4 Lab F part D:

In this part of the lab, we use clock for the first time. A clock signal is one which varies from a logic high to a logic low periodically (usually a square wave). A transition of the signal from logic low to logic high is referred to as a rising edge. A transition of the signal from logic high to logic low is known as a falling edge. We also have used a clock divider circuit which has numerous uses. It generates a lower-frequency clock by using a scaling factor and the crystal oscillator clock from the FPGA board as inputs. We have created a topmodule file to instantiate this clock divider module and the LFSR module to implement the design on the Basys 3 Board. When we set both reset and enable to 1 the LFSR starts to take feedbacks and shows the intermediate states using the LEDs.

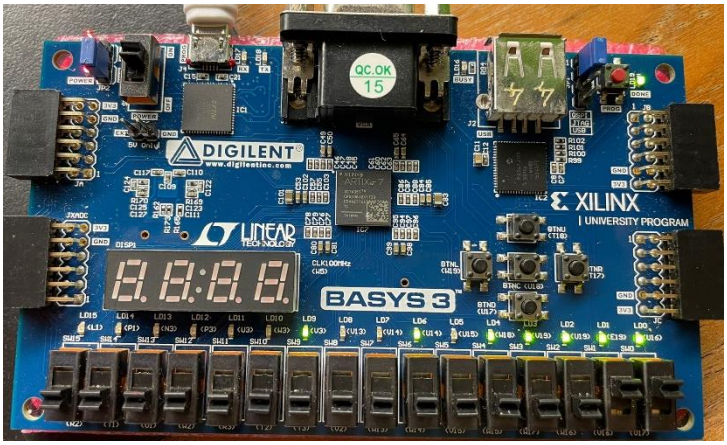


Fig 7.13 One of the intermediate states shown using the LEDs

8. Observations:

8.1 Lab F part B:

Utilization Report, which is produced following the synthesis and implementation phases, provides information on how well our design is utilizing the FPGA's resources. Here we observe that IO resource is utilizing the most percentage of resources, i.e., 17% .

Resource	Utilization	Available	Utilization %
LUT	90	20800	0.43
FF	78	41600	0.19
IO	18	106	16.98

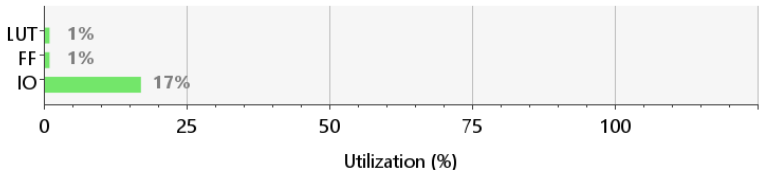


Fig 8.1 Summary of Utilization Report

Power Consumption Report provides information about the target FPGA device's power consumption profile throughout the execution of our design. Here we observe that the total on-chip power is 0.09W and the Junction Temperature is 25.5℃ which tells the designer reliability of their FPGA designs and making thermal management strategies to ensure safe operation of device.

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.09 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (11.8 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

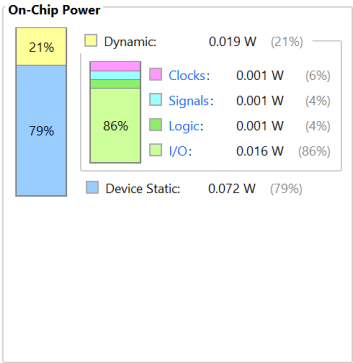


Fig 8.2 Summary of Power Report

Design Timing Report provides insights into our design's timing behaviour, including clock limitations, timing violations, and key routes. For a designer, this is the most crucial report because it is only via analysis of this report that he can determine whether he is fulfilling his timing goals. Here we see that all user specified timing constraints are met.

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	4.628 ns	Worst Hold Slack (WHS):	0.122 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	100	Total Number of Endpoints:	100	Total Number of Endpoints:	79

All user specified timing constraints are met.

Fig 8.3 Summary of Design Timing

Noise Report is a useful tool for evaluating the noise properties of our FPGA design, spotting any noise-induced errors and signal integrity problems, and putting mitigation plans into practice to improve design performance and reliability.

I/O Bank Details

Name	Port	I/O Std	Vcco	Slew	Drive Strength (...	Off-Chip Termina...	Remaining Margin...
I/O Bank 0 (0)							
I/O Bank 14 (0)							
I/O Bank 16 (0)							
I/O Bank 34 (11)		LVCMOS33	3.30	SLOW	12	FP_VTT_50	
U2	anode_sel[0]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	70.53
U4	anode_sel[1]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	73.55
V4	anode_sel[2]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	63.39
W4	anode_sel[3]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	69.49
U7	led_out[0]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	69.76
V5	led_out[1]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	74.80
U5	led_out[2]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	66.65
V8	led_out[3]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	66.92
U8	led_out[4]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	68.51
W6	led_out[5]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	68.28
W7	led_out[6]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	88.53
I/O Bank 35 (0)							

Fig 8.4 Summary of Noise Report

8.2 Lab F part D:

Utilization Report, which is produced following the synthesis and implementation phases, provides information on how well our design is utilizing the FPGA's resources. Here we observe that IO resource is utilizing the most percentage of resources, i.e., 16% .

Resource	Utilization	Available	Utilization %
LUT	37	20800	0.18
FF	53	41600	0.13
IO	17	106	16.04

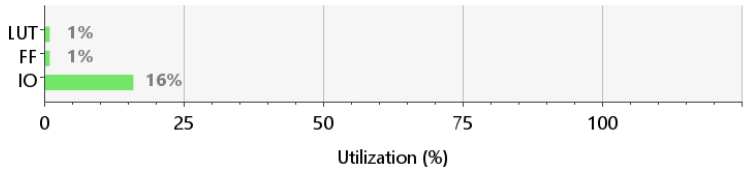


Fig 8.5 Summary of Utilization Report

Power Consumption Report provides information about the target FPGA device's power consumption profile throughout the execution of our design. Here we observe that the total on-chip power is 0.07W and the Junction Temperature is 25.4℃ which tells the designer reliability of their FPGA designs and making thermal management strategies to ensure safe operation of device.

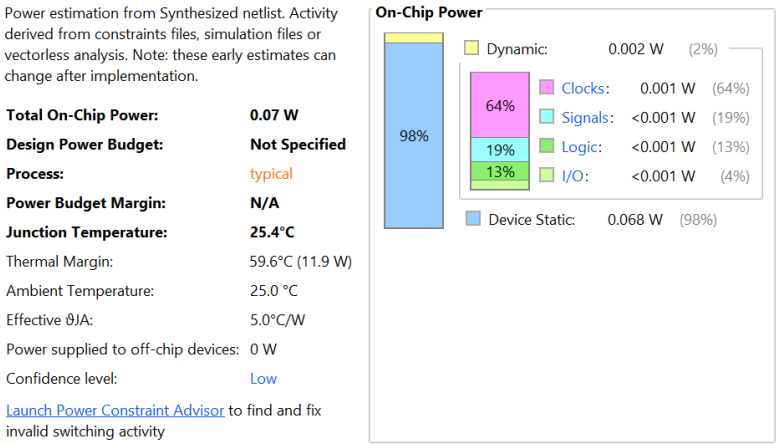


Fig 8.6 Summary of Power Report

Design Timing Report provides insights into our design's timing behaviour, including clock limitations, timing violations, and key routes. For a designer, this is the most crucial report because it is only via analysis of this report that he can determine whether he is fulfilling his timing goals. Here we see that all user specified timing constraints are met.

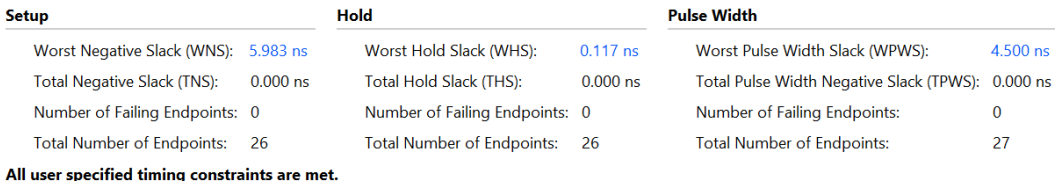


Fig 8.7 Summary of Design Timing

Noise Report is a useful tool for evaluating the noise properties of our FPGA design, spotting any noise-induced errors and signal integrity problems, and putting mitigation plans into practice to improve design performance and reliability. Here we can observe noise margins for all the ports used on the Board.

I/O Bank Details

Name	Port	I/O Std	Vcco	Slew	Drive Strength (...	Off-Chip Termina...	Remaining Margin...
I/O Bank 0 (0)							
I/O Bank 14 (9)		LVCMOS33	3.30	SLOW	12	FP_VTT_50	
U16	Q_out[0]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	73.28
E19	Q_out[1]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	96.38
U19	Q_out[2]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	86.30
V19	Q_out[3]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	87.48
W18	Q_out[4]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	85.00
U15	Q_out[5]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	70.89
U14	Q_out[6]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	73.19
V14	Q_out[7]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	72.99
V13	Q_out[8]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	70.24
I/O Bank 16 (0)							
I/O Bank 34 (3)		LVCMOS33	3.30	SLOW	12	FP_VTT_50	
V3	Q_out[9]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	94.73
W3	Q_out[10]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	95.26
U3	Q_out[11]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	95.01
I/O Bank 35 (2)		LVCMOS33	3.30	SLOW	12	FP_VTT_50	
P3	Q_out[12]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	95.68
N3	max_tick_reg	LVCMOS33	3.30	SLOW	12	FP_VTT_50	96.70

Fig 8.8 Summary of Noise Report

9. Conclusion:

In conclusion, the 3C7 Digital Systems Design Laboratory provided a comprehensive platform to apply theoretical knowledge to practical scenarios, enhancing our understanding of sequential logic circuits. Through the design, implementation, and testing of various components on the Basys-3 Board, we gained valuable insights into the dynamic nature of digital systems.

Our experiments with D Flip-Flops, debouncers, LFSRs, and clock dividers illustrated the critical role of these components in digital electronics. The use of DFFs to maintain state information, debouncers to ensure reliable button and switch interactions, LFSRs for generating pseudo-random sequences, and clock dividers to manage signal frequencies underscored the complex adaptability of elements within digital systems.

The utilization and power consumption reports highlighted the efficient use of FPGA resources, demonstrating the importance of optimization in digital design. Meeting the design timing constraints validated our implementation strategies, ensuring that our designs are not only functional but also robust and reliable.

10. References:

- Lecture Slides
- Lab materials provided on blackboard for this lab
- Basic knowledge of electronics from previous modules
- Doubt clearing from the demonstrators
- Xilinx data sheet provided
- Draw.io to draw block diagram