ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY
EXPERIMENT REPORT 7

EXPERIMENT NO : 7

EXPERIMENT DATE : 04.06.2020

LAB SESSION : FRIDAY - 13.30

GROUP NO : G11

GROUP MEMBERS:

150180064 : MELİS & GÜNŞEBER

150170043 : EBRAR & ÖMER

SPRING 2020

Contents

1 INTRODUCTION [10 points]

In this experiment we first implemented a sequential circuit with D flip flop then worked on counters. At the beginning we implemented a 2 bit counter which counts to 3 with every clock cycle then a 16 bit counter. At the end we modified part 3 as a circular counter with the given inputs.

2 MATERIALS AND METHODS [40 points]

2.1 PART 1

In the first part we implemented the given circuit with using 2 D flip flops, 2 OR gates, 1 AND gate and a clock signal.The state transition table and state diagram are given below and with the help of these two we observe and compare the results with the simulation. To create the table and diagram first we generated the input and output equations.

$D_A = Q_B + Q'_A$

$D_B = X(Q'_B)$

$Y = Q_A + Q'_B$

The characteristic equation for D flip flop is $D = Q_{n+1}$

Thus, $Q_A^+ = D_A$ and $Q_B^+ = D_B$

Q_A	Q_B	X	Q_A^+	Q_B^+	Y
0	0	0	1	0	1
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	1

Figure 1: State Transition Table

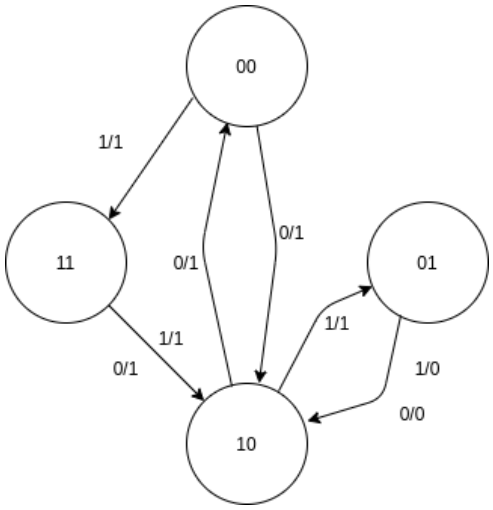

Figure 2: State Diagram of the Circuit

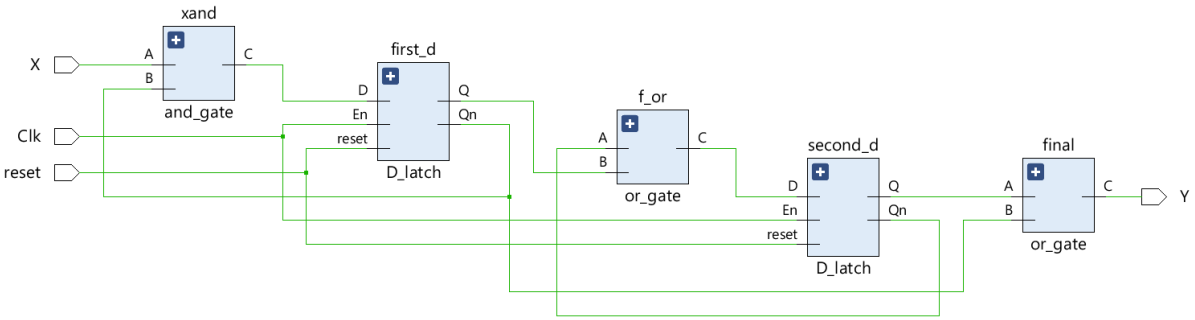

Figure 3: Elaborated design for Sequential Circuit with D Flip Flop

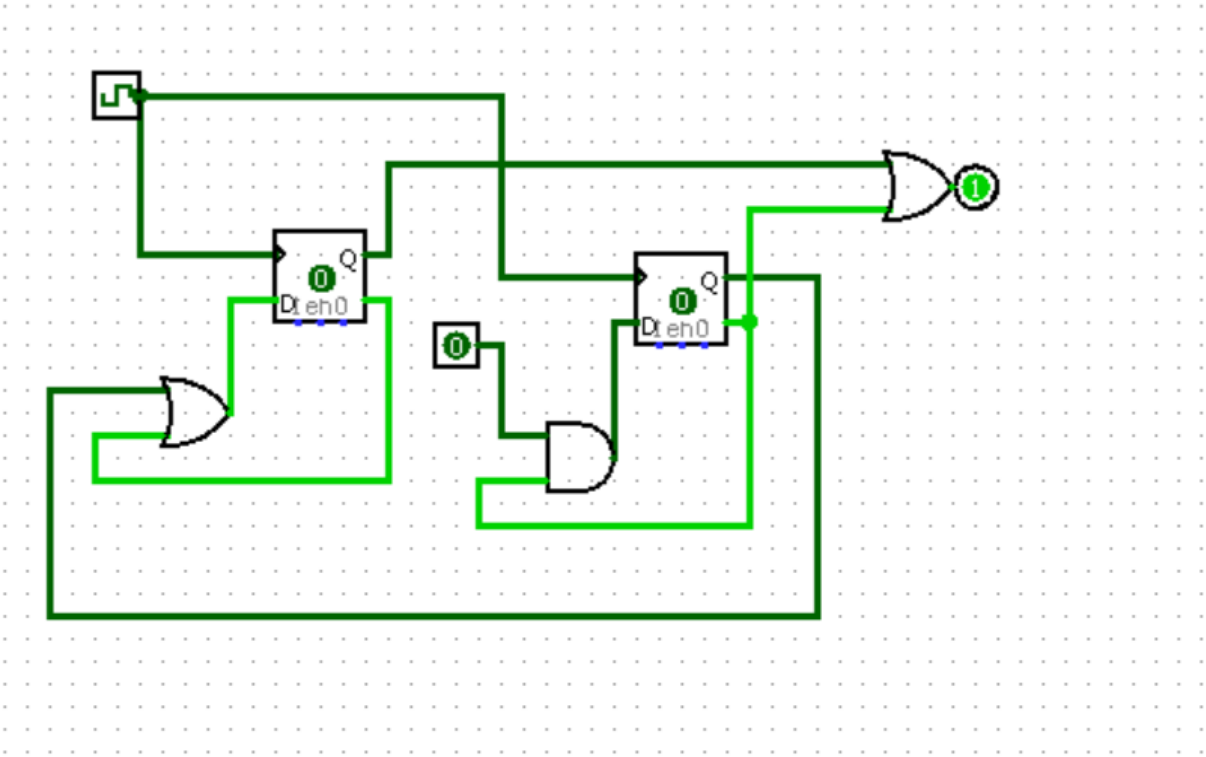

Figure 4: Logism Circuit for Sequential Circuit with D Flip Flop

Design Code

```
module Seq_D(input Clk,input reset,input X,output Y);
    wire x1,x2;
    wire q1;wire q2;
    wire qn1,qn2;

    and_gate xand(X,qn1,x1);
    D_latch first_d(x1,Clk,reset,q1,qn1);z


    or_gate f_or(qn2,q1,x2);
    D_latch second_d(x2,Clk,reset,q2,qn2);

    or_gate final(q2,qn1,Y);

endmodule
```

Testbench Code

```
module d_test();
    reg clk;
    reg x,reset;
    wire y;
    Seq_D uut(clk,reset,x,y);
    initial begin
    reset=0; clk=0; x=0; #50;
    reset=0; clk=1; x=0; #50;
    reset=1; clk=0; x=0; #50;
```

```
10      clk=1; x=0; #50;
11      clk=0; #50;
12      clk=1; x=1; #50;
13      clk=0; #50;
14      clk=1; x=1; #50;
15      clk=0; #50;
16      clk=1; x=0; #50;
17      clk=0; #50;
18      clk=1; x=1; #50;
19      clk=0; #50;
20      clk=1; x=0; #50;
21      clk=0; #50;
22      clk=1; x=1; #50;
23      clk=0; #50;
24      clk=1; x=0; #50;
25      clk=0; #50;
26      end
27  endmodule
```

2.2 PART 2

In tis part we implemented a 2 bit counter with 2 D flip flops, 2 OR gates, 5 AND gates, a NOT gate and a clock signal. The counter counts up or down dependent on x, meaning when x=0 it counts down and when x=1 it counts up until 2. We gave the D latches a reset input so the counter became circular.

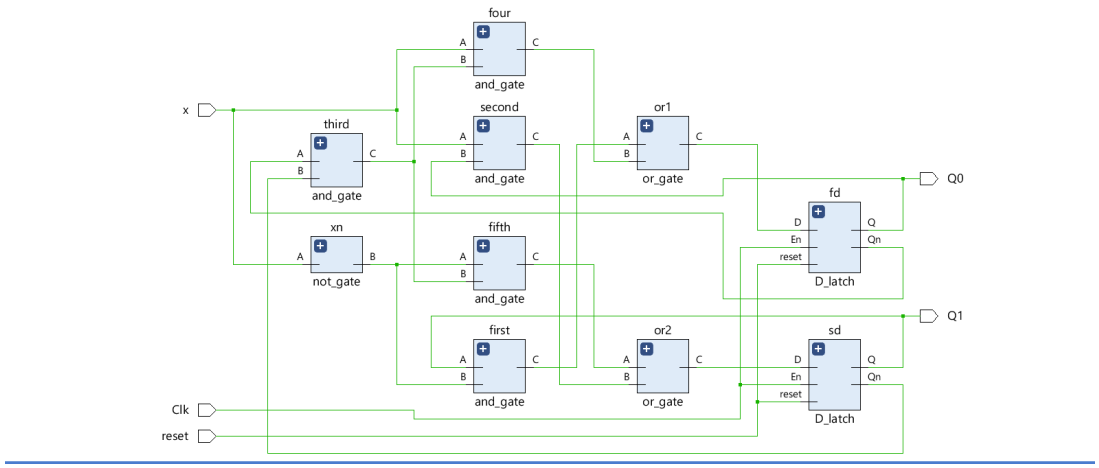

Figure 5: Elaborated design for 2 Bit Counter

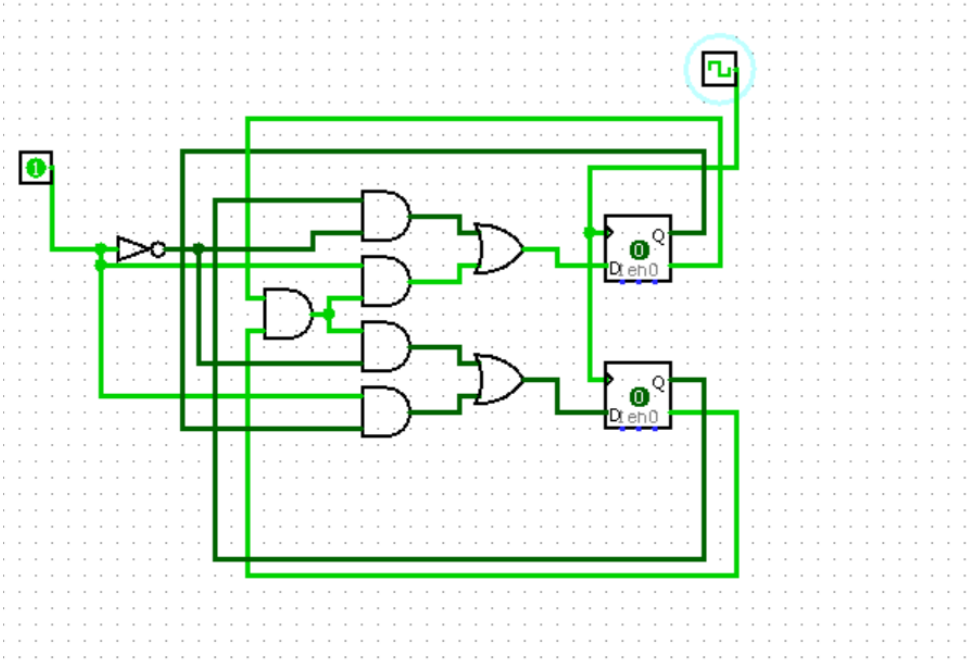

Figure 6: Logism Circuit for 2 Bit Counter

Design Code

```
module counter_2bit( input x, input Clk,input reset,output Q0, output Q1);

    wire q0; wire q1;
    wire xnot;
    wire or11,or22;
    wire q0not, q1not;
    wire new;
    wire or12, or21;
    wire D0,D1;

    not_gate xn(x,xnot);
    and_gate first(Q1,xnot,or11);
    and_gate second(x,Q0,or22);

    and_gate third(q0not,q1not,new);

     and_gate four(x,new,or12);
     and_gate fifth(xnot,new,or21);

    or_gate or1(or11,or12,D0);
    or_gate or2(or21,or22,D1);

    D_latch fd(D0,Clk,reset,Q0,q0not);
    D_latch sd(D1,Clk,reset,Q1,q1not);


endmodule
```

Testbench Code

```
module part2_test();
    reg x,clk,reset;
    wire q0,q1;
   counter_2bit uut(x,clk,reset,q0,q1);
    initial begin
     clk=0;   x=1;  reset=0; #50;
     clk=1;   x=1;  reset=0; #50;
     clk=0;   x=1;  reset=1; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; x=0; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;
     clk=1; #50;
     clk=0; #50;

     end
endmodule
```

2.3 PART 3

In the third part we implemented 16 bit up-down counter using reg type parameter and always block. The counter counts from 0 to the end of the 16 bit numbers. In the simulation the count number is equal to the clock period. The design and testbench codes are also given below.

Design Code

```
module up_down_counter(input[15:0],begpoint,inputclk,load,reset,up_down,output[15:0]counter);
reg [15:0] counter_up_down;
always @(posedge clk or posedge reset)
begin
if(~reset)
 counter_up_down<=begpoint;
else if(~load) begin
    if(up_down)
        counter_up_down <= counter_up_down + 16'd1;
    else
        counter_up_down <= counter_up_down - 16'd1;
    end
end
```

```
14 assign counter = counter_up_down;
15 endmodule
```

Testbench Code

```
1  module updowncounter_testbench();
2  reg clk, reset,up_down,load;
3  reg[15:0] begpoint;
4  wire [15:0] counter;
5
6  up_down_counter dut(begpoint,clk, load,reset,up_down, counter);
7  initial begin
8  clk=0;
9  forever #5 clk=~clk;
10 end
11 initial begin
12 begpoint =16'h0; reset=0;up_down=0;load=1;
13 #20;
14 reset=1; load=0;
15 #200;
16 up_down=1;
17 end
18 endmodule
```

2.4 PART 4

In part 4 we have implemented the same counter but with given number ranges. So we wrote different test codes with giving them beginning points and reset inputs for the counter to stop and count from the beginning again as a circular counter. The test codes are given below.

Testbench Code

```
1  module part4_test();
2      reg clk, reset,up_down,load;
3      reg[15:0] begpoint;
4      wire [15:0] counter;
5      reg [5:0] clocktime;
6      up_down_counter dut(begpoint,clk, load,reset,up_down, counter);
7      initial begin //code for counting 0 to 20
8      clk=1;
9      clocktime=0;
10     begpoint =16'd0; reset=0;up_down=1;load=1;
11     forever begin #5 clk=~clk; clocktime=clocktime+6'd1;
12     if(clocktime<6'd40)begin reset=1; load=0; end
13     else begin reset=0;load=1; clocktime=5'd0; end
14     end
15     end
16     initial begin //code for counting 432 to 454
17     clk=1;
18     clocktime=0;
19     begpoint =16'd432; reset=0;up_down=1;load=1;
20     forever begin #5 clk=~clk; clocktime=clocktime+6'd1;
21     if(clocktime<6'd44)begin reset=1; load=0; end
```

```
22      else begin reset=0;load=1; clocktime=5'd0; end
23      end
24      end
25      initial begin //code for counting 65525 to 65535
26      clk=1;
27      clocktime=0;
28      begpoint =16'd65525; reset=0;up_down=1;load=1;
29      forever begin #5 clk=~clk; clocktime=clocktime+6'd1;
30      if(clocktime<6'd20)begin reset=1; load=0; end
31      else begin reset=0;load=1; clocktime=5'd0; end
32      end
33      end
34
35  endmodule
```

3 RESULTS [15 points]

3.1 Simulation Codes for Part 1

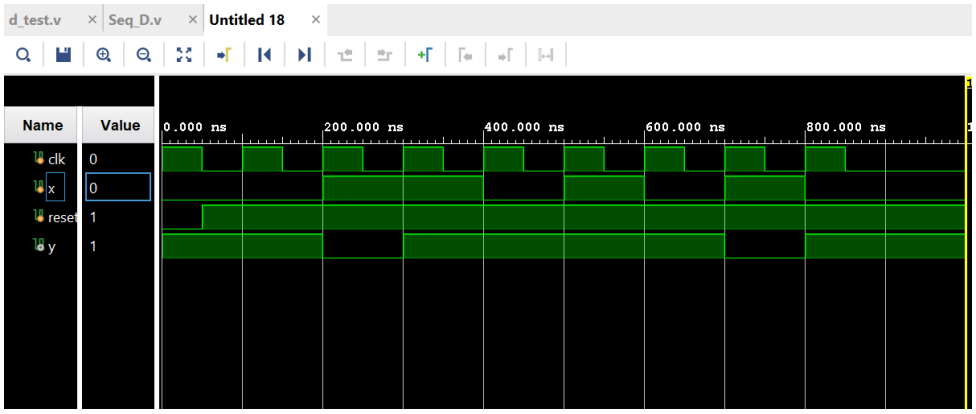

Figure 7: Simulation Code for Sequential Circuit with D Flip Flop

In this part we observed Sequential circuit with D flip flop according to state transition table of this part.

3.2 Simulation Codes for Part 2

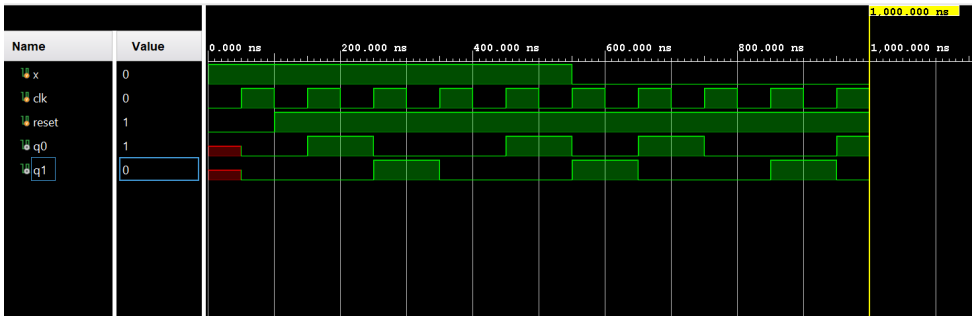

Figure 8: Simulation Code for 2-Bit Counter

In this part, we observed a counter which counts 0 to 2 in circular way. We used q0 for least significant bit and q1 for most significant bit. Beginning of the simulation we count up,then count down as shown in the figure.

3.3 Simulation Codes for Part 3

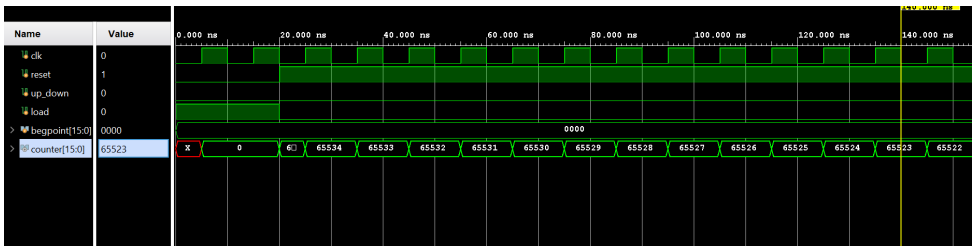

Figure 9: Simulation Code for Counter 16 bit-Counts down

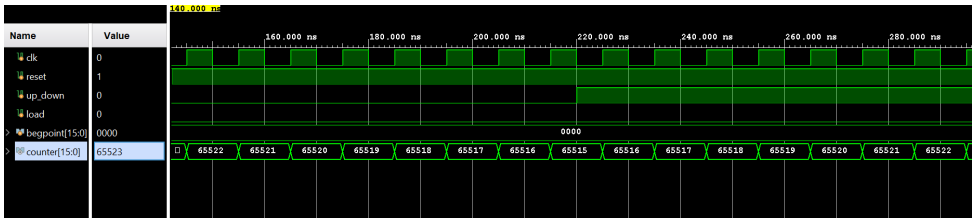

Figure 10: Simulation Code for Counter 16 bit-Counts up

In this part we observed 16 bit counter, simulation starts with down way and after x being 1 it counts up.

3.4 Simulation Codes for Part 4

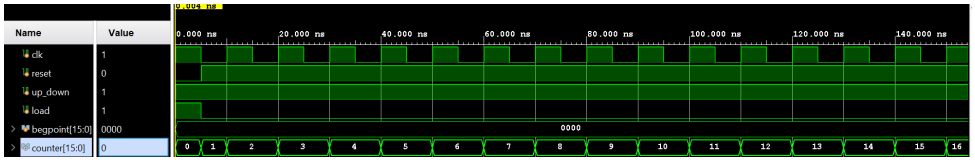

Figure 11: Simulation Code for counting 0 to 20

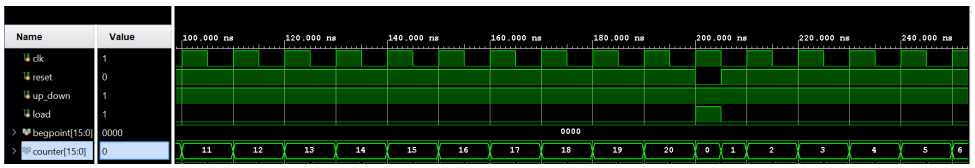

Figure 12: Simulation Code for counting 0 to 20

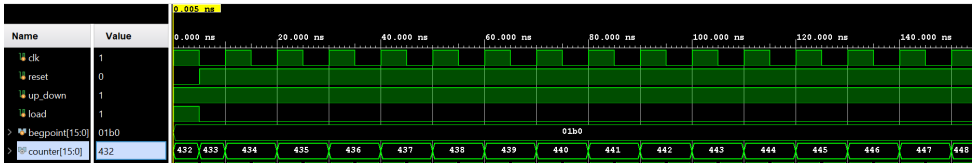

Figure 13: Simulation Code for counting 432 to 454

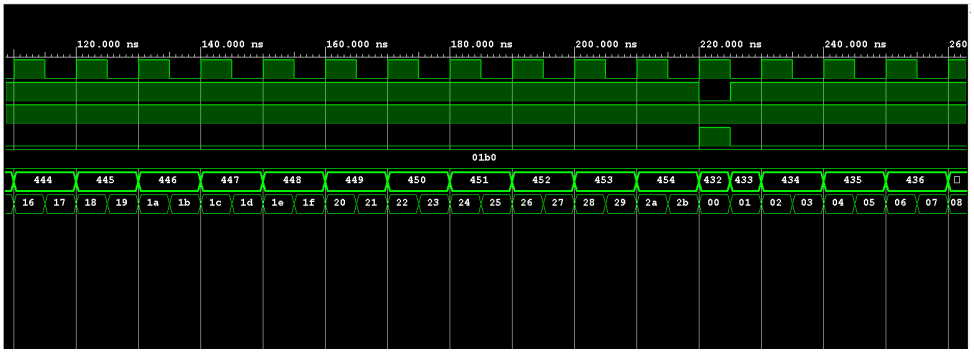

Figure 14: Simulation Code for counting 432 to 454

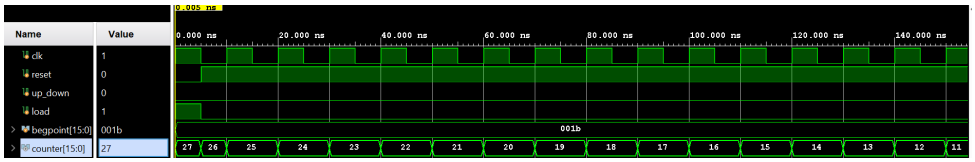

Figure 15: Simulation Code for counting 27 to 5

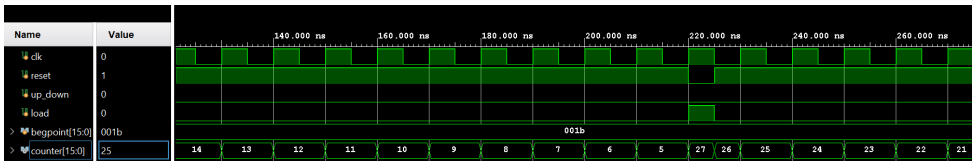

Figure 16: Simulation Code for counting 27 to 5

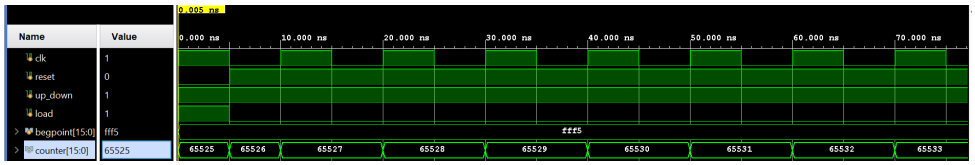

Figure 17: Simulation Code for counting 65525 to 65535

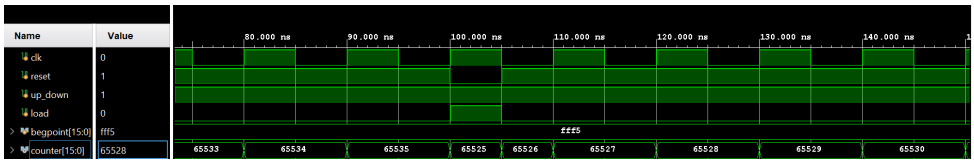

Figure 18: Simulation Code for counting 65525 to 65535

In this part we gave a beginning point for 16-bit counter and stopped the counting with appropriate clock time. For example if we want to count 20 time, we give 40 clock for on and off. We observed the test modules with more than one beginning point.

4 DISCUSSION [25 points]

In this experiment firstly we draw the state transition table and the state diagram of the sequential circuit and compare the results with the simulation in Verilog. Then we built a 2 bit counter with using D Flip Flops.We figured out if we don't give reset inputs to D latchs at first, the simulation fails. Because simulation uses the outputs of Dlatch at the same time. In the change of X the counter switches up and down. Then we implement a 16 bit circuit using reg type parameter and always block for giving the clock frequently.And last we increment a register clocktime which holds the difference between the given two number. So each time when the counter equals to the given final integer we reset the circuit to be circular.

5 CONCLUSION [10 points]

In this experiment we learned to use the reg type parameters more deeply. We had some trouble with the testbenches and also with the simulations but we eventually reinforced how to give the clock signals truly. We learned the counters very detailed and tried to count as we want not just from 0 to the maximum bit.