ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	6
EXPERIMENT DATE	:	28.05.2020
LAB SESSION	:	FRIDAY - 13.30
GROUP NO	:	G11

GROUP MEMBERS:

150180064 : MELİS GÜNŞEBER

150170043 : EBRAR ÖMER

150140902 : PERİT JAN AYDEMİR

SPRING 2020

Contents

FRONT COVER

CONTENTS

1 INTRODUCTION **1**

1.1 AND Gate 1

1.2 OR Gate 1

1.3 NOT Gate 1

2 MATERIALS AND METHODS **2**

2.1 Experiment I – SR Latch 2

2.2 Experiment II – SR Latch with and Enable Input 3

2.3 Experiment III – DD Flip Flop from D Latches 4

2.3.1 D Latch 4

2.3.2 Negative Triggered D Flip Flop 4

2.4 Part 4 5

3 RESULTS **6**

3.1 SR Latch Simulation 6

3.2 SR Latch with Enable Simulation 8

3.3 Negative Triggered D Flip-Flop Simulation 9

3.4 Part 4 Simulation 10

3.4.1 1/2 Frequency Simulation 12

3.4.2 1/4 Frequency Simulation 13

3.4.3 1/8 Frequency Simulation 13

3.4.4 1/3 Pulse-Gap Simulation 14

3.4.5 1/7 Pulse-Gap Simulation 14

4 DISCUSSION **14**

5 CONCLUSION **15**

6 REFERENCES **15**

1 INTRODUCTION

The main goal of this experiment is to implement latches and flip-flops using the Verilog software. This is an introduction experiment for us to use and learn the usage and functions of the Verilog. We have imported our basic modules from the previous experiment as can seen below.

1.1 AND Gate

```
module and_gate(
    input A,
    input B,
    output C
    );
    assign C = A & B;
endmodule
```

1.2 OR Gate

```
module or_gate(
    input  A,
    input  B,
    output  C
    );
    assign C=A|B;
endmodule
```

1.3 NOT Gate

```
module not_gate(
    input A,
    output B
    );
    assign B=~A;
endmodule
```

2 MATERIALS AND METHODS

2.1 Experiment I – SR Latch

In the first part, we implemented a SR latch using only NOR gates. The S and R are our inputs. Q and Q n are our outputs. There is not any enable input in this latch design.

Verilog Code

```
module sr_latch(
    input S,
    input R,
    output Q,
    output Qn
    );
    wire q1,qn1;
    or_gate or1(S,Q,qn1);
    not_gate not1(qn1,Qn);
    or_gate or2(R, Qn, q1);
    not_gate not2(q1,Q);
endmodule
```

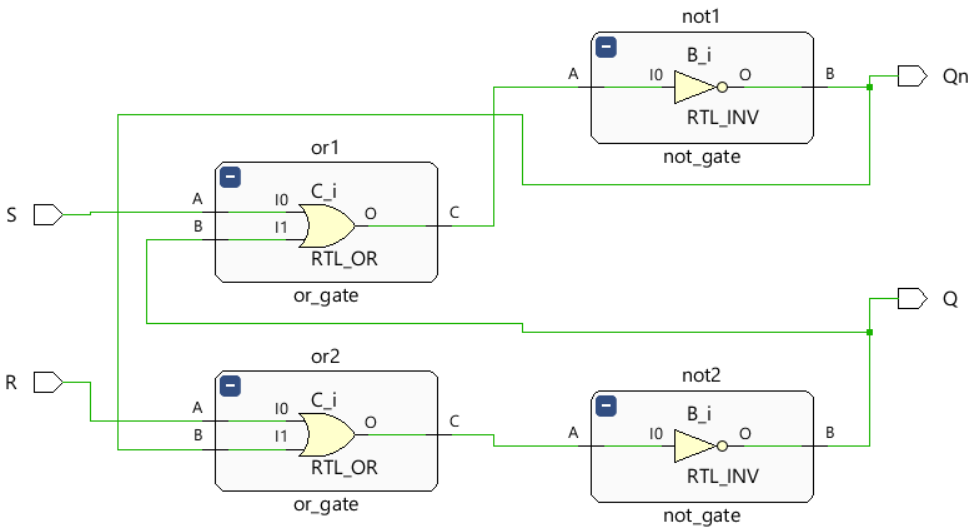

Figure 1: SR Latch

We have defined the inputs S, R and the output as Q and Qn. And after that, we have send S and Q to an OR gate, Q and Qn to NOT gates and R and Qn to an Or gate.

2.2 Experiment II – SR Latch with and Enable Input

In this part, similarly like the previous part we implemented a SR latch but with an enable input with only using NAND gates. The S and R are our inputs. Q and Qn are our outputs.

Verilog Code

```
module sr_latch_enabled(
    input A,
    input B,
    input C,
    output Q,
    output Qn
    );

    wire S, R;
    wire s,r,q,qn;

    and_gate s1(A,C,s); not_gate s2(s,S);
    and_gate r1(B,C,r); not_gate r2(r,R);
    and_gate out1(S,Qn,q); not_gate out2(q,Q);
    and_gate comp1(R,Q,qn); not_gate comp2(qn,Qn)
```

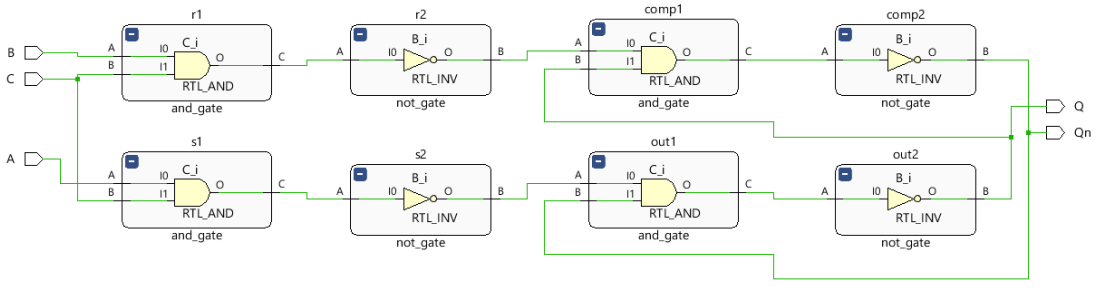

Figure 2: SR Latch with Enable Input

We have defined the inputs A, B and C. C is our enable input and we implemented outputs as Q and Qn. This time we used S and R as wires. Wires only carry A and B's values if the C is enabled.

2.3 Experiment III – DD Flip Flop from D Latches

In this part, we implemented a negative edge triggered D Flip Flop from D latches. The D is our input. Q and Qn are our outputs. We have implemented D latches with only NAND and NOT gates with enable inputs.

2.3.1 D Latch

```
module D_latch(
    input D,
    input En,
    output Q,
    output Qn
    );
    wire R;
    not_gate n1(D,R);
    sr_latch_enabled s(D,R,En,Q,Qn);

endmodule
```

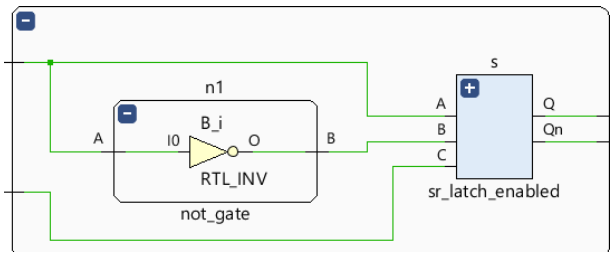

Figure 3: D Latch

As seen above we have used SR latch within the D latch.

2.3.2 Negative Triggered D Flip Flop

```
module Triggered_D(input D,input CLK, output Q, output Qn);
    wire Clk_new, Qm, Qmn;
    not_gate clkn(CLK,Clk_new);
    D_latch d1(D,CLK,Qm, QMn);
    D_latch d2(Qm,Clk_new,Q,Qn);
endmodule
```

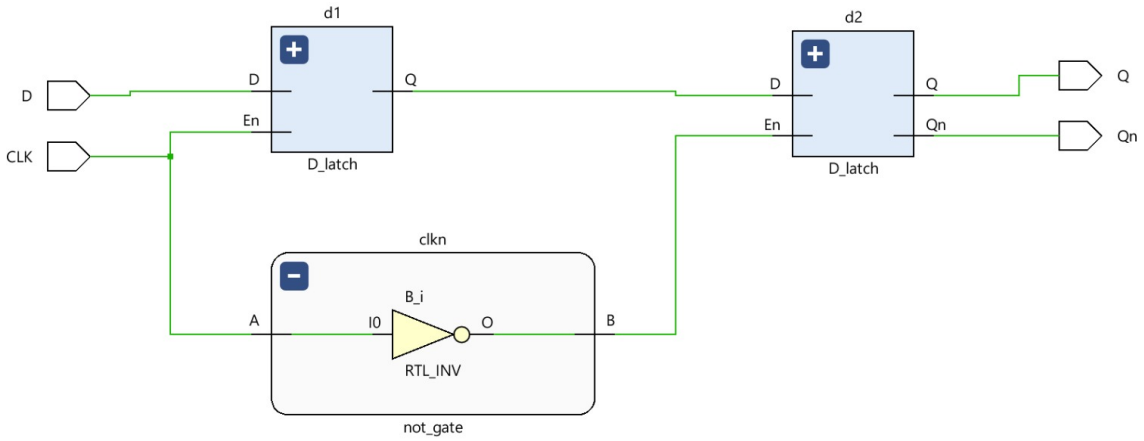

Figure 4: Negative Triggered D Flip-Flop

2.4 Part 4

In this part, we implemented a pulse generator with circular shift register module. The implementation is built to support all the variable pulse frequencies and durations. The D is our input. Q and Qn are our outputs. We have implemented D latches with only NAND and NOT gates with enable inputs. We have a 8-bit input, a 1-bit load input, a CLK input and a 1-bit output. First, we load our 8 bit input and we record the most significant bit, shift all the rest bits to 1-bit left and then add the most significant bit to the end.

```
module loadshift (Inp,C,L, SO);
input [7:0] Inp;
input C,L;
output SO;
reg [7:0] tmp;
reg hold;
  always @(posedge C)
    if (L) begin
         tmp=Inp;
      //   hold=tmp[0];
    end
    else begin
       hold=tmp[7];
       tmp = tmp << 1;
       tmp[0]=hold;
    end
    assign SO=hold;
endmodule
```

3 RESULTS

3.1 SR Latch Simulation

```
module sr_test();
    reg S;
    reg R;
    wire Q;
    wire Qn;

    sr_latch uut(S, R,Q,Qn);
    initial begin
        S =1;   R=0;    #200;
        S =0;   R=0;    #200;
        S =0;   R=1;    #200;
        S =0;   R=0;    #200;
        S =1;   R=1;    #200; //forbidden
     end
endmodule
```

Truth Table of SR Latch

S	R	Q	Qn
0	0	1	0
0	1	0	1
1	0	1	0
1	1	0	0

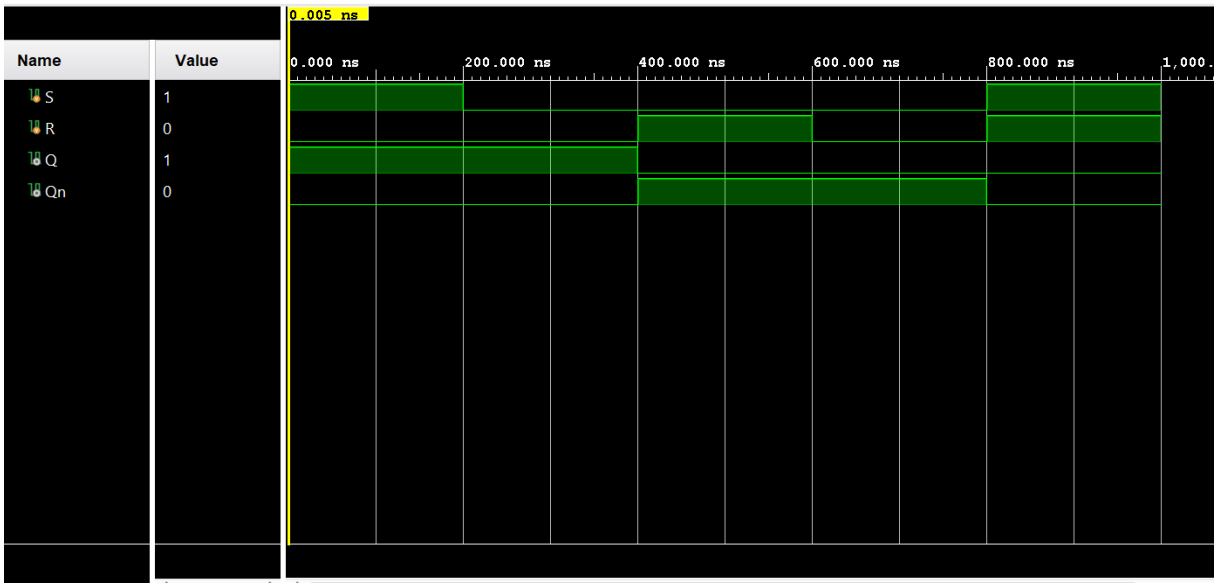

Figure 5: SR Latch Simulation

3.2 SR Latch with Enable Simulation

```
module sr_enable_test();
    reg S;
    reg R;
    reg C;
    wire Q;
    wire Qn;

    sr_latch_enabled uut(S, R,C,Q,Qn);
    initial begin
        S =0;   R=0;  C=0;   #150; //memory
        S =1;   R=0;  C=1;   #150;
        S =0;   R=1;  C=1;   #150;
        S =0;   R=0;  C=1;   #150;
        S =0;   R=0;  C=0;   #150; //memory
        S =1;   R=1;  C=1;   #150; //forbidden
     end
endmodule
```

Truth Table of SR Latch with Enable

S	R	En	Q	Qn
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

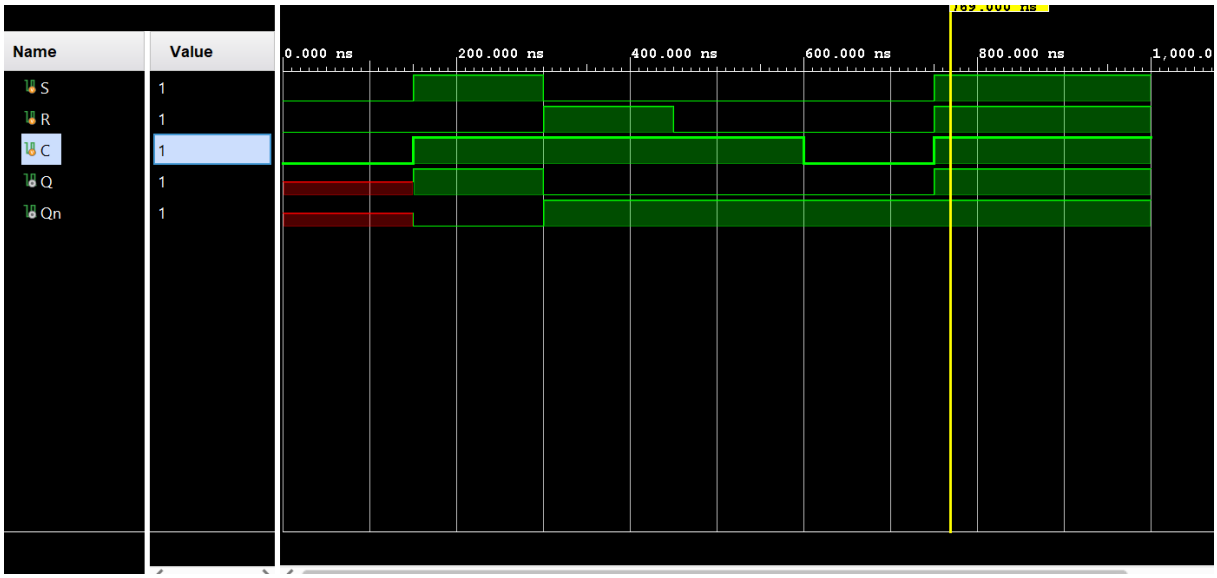

Figure 6: SR Latch with Enable Simulation

3.3 Negative Triggered D Flip-Flop Simulation

```
module D_latch(
    input D,
    input En,
    output Q,
    output Qn
    );
    wire R;
    not_gate n1(D,R);
    sr_latch_enabled s(D,R,En,Q,Qn);

endmodule

module Triggered_D(input D,input CLK, output Q, output Qn);
    wire Clk_new, Qm, Qmn;
    not_gate clkn(CLK,Clk_new);
    D_latch d1(D,Clk_new,Qm, QMn);
    D_latch d2(Qm,CLK,Q,Qn);
endmodule
```

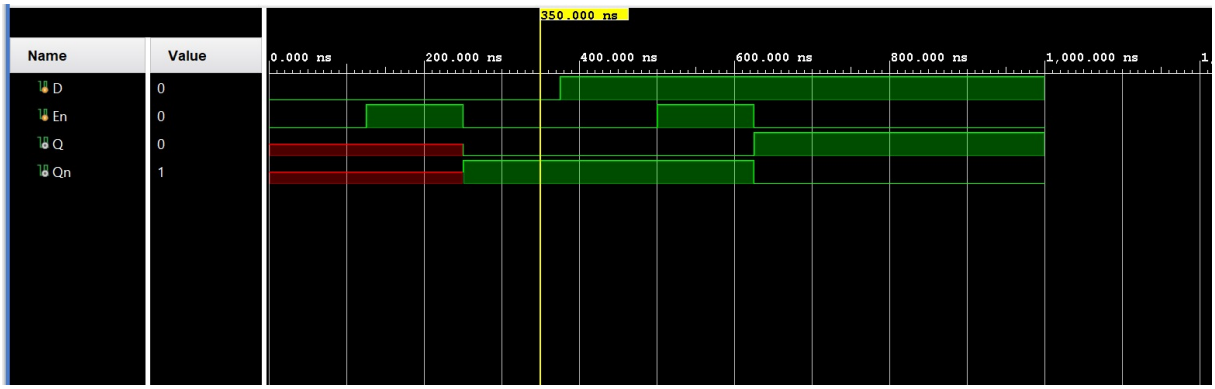

Figure 7: Negative Triggered D Flip-Flop

3.4 Part 4 Simulation

For each signal, observe both input and output. When Load=0 (that is, shift=1), circular shift operation is done, when Load=1, input value is loaded. in the shift operation or the value been loaded in the load operation. This circuit is positive edge triggered. Hint: You can analyze the internal structure of 74XX165 IC for pulse-generator.

```
module ls_test();
    reg [7:0] signal;
    reg clock;
    reg load;
    wire outp;
    loadshift deneme(signal,clock,load,outp);
    initial begin
        signal=8'b10101010 ;
        clock=1; load=1; #50;
        clock =0;#50;
        clock=1; load=0;#50;
        clock=0; #50; clock=1; #50;
        clock=0; #50; clock=1; #50;
        clock=0; #50; clock=1; #50;
        clock=0; #50; clock=1; #50;
        clock=0; #50; clock=1; #50;
        clock=0; #50; clock=1; #50;
        clock=0; #50; clock=1; #50;
        clock=0; #150;
    end
     initial begin
        signal=8'b11001100 ;
        clock=1; load=1; #50;
```

```
24          clock =0;#50;
25          clock=1; load=0;#50;
26          clock=0; #50; clock=1; #50;
27          clock=0; #50; clock=1; #50;
28          clock=0; #50; clock=1; #50;
29          clock=0; #50; clock=1; #50;
30          clock=0; #50; clock=1; #50;
31          clock=0; #50; clock=1; #50;
32          clock=0; #50; clock=1; #50;
33          clock=0; #150;
34      end
35       initial begin
36          signal=8'b11110000 ;
37          clock=1; load=1; #50;
38          clock =0;#50;
39          clock=1; load=0;#50;
40          clock=0; #50; clock=1; #50;
41          clock=0; #50; clock=1; #50;
42          clock=0; #50; clock=1; #50;
43          clock=0; #50; clock=1; #50;
44          clock=0; #50; clock=1; #50;
45          clock=0; #50; clock=1; #50;
46          clock=0; #50; clock=1; #50;
47          clock=0; #150;
48      end
49       initial begin
50          signal=8'b10001000 ;
51          clock=1; load=1; #50;
52          clock =0;#50;
53          clock=1; load=0;#50;
54          clock=0; #50; clock=1; #50;
55          clock=0; #50; clock=1; #50;
56          clock=0; #50; clock=1; #50;
57          clock=0; #50; clock=1; #50;
58          clock=0; #50; clock=1; #50;
59          clock=0; #50; clock=1; #50;
60          clock=0; #50; clock=1; #50;
61          clock=0; #150;
62      end
63       initial begin
```

```
64          signal=8'b10000000 ;
65          clock=1; load=1; #50;
66          clock =0;#50;
67          clock=1; load=0;#50;
68          clock=0; #50; clock=1; #50;
69          clock=0; #50; clock=1; #50;
70          clock=0; #50; clock=1; #50;
71          clock=0; #50; clock=1; #50;
72          clock=0; #50; clock=1; #50;
73          clock=0; #50; clock=1; #50;
74          clock=0; #50; clock=1; #50;
75          clock=0; #150;
76      end
77
78  endmodule
```

3.4.1 1/2 Frequency Simulation

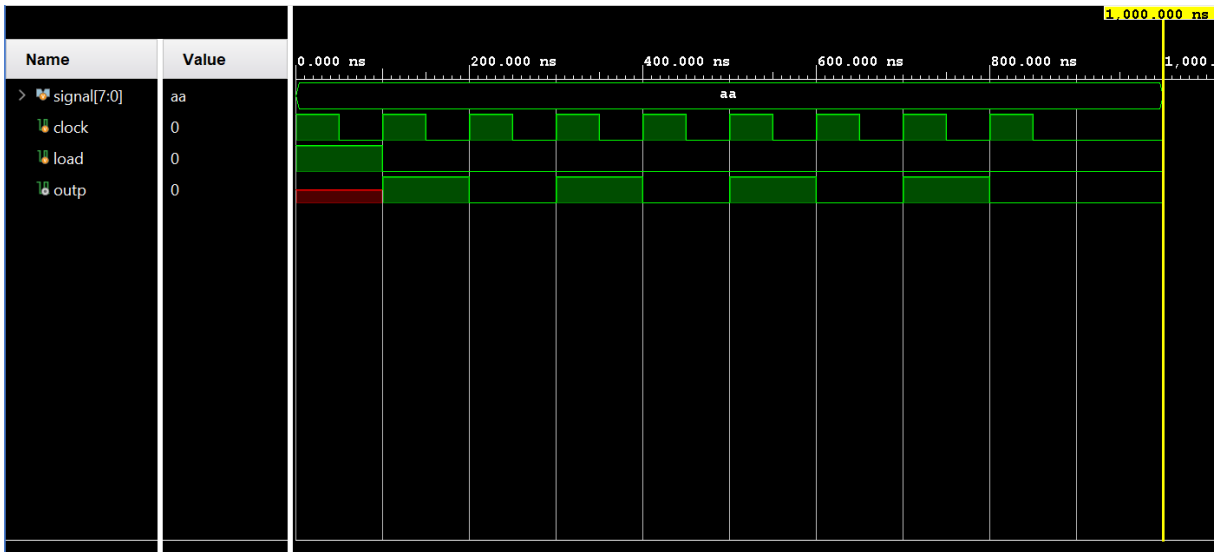

Figure 8: 1/2 Frequency Simulation

3.4.2 1/4 Frequency Simulation

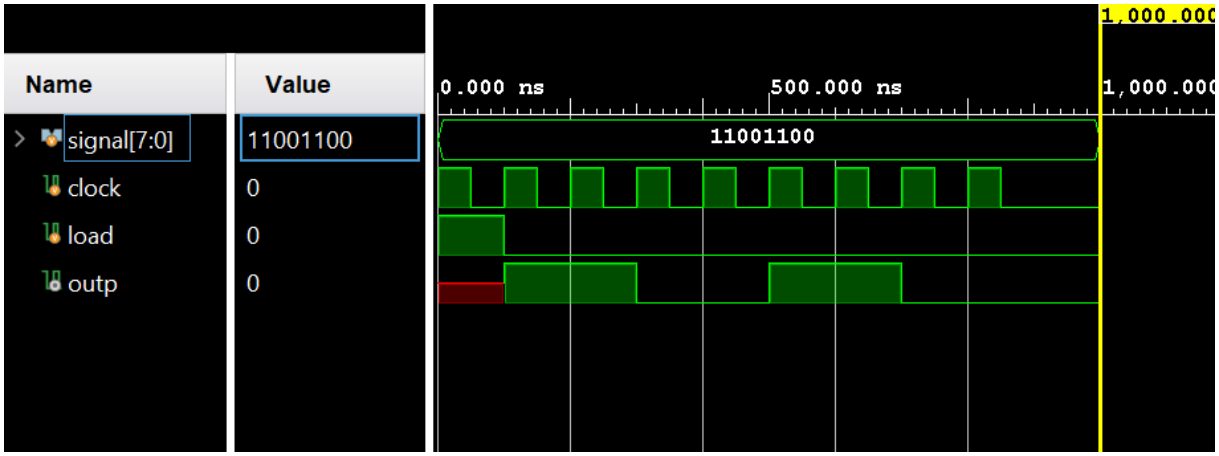

Figure 9: 1/4 Frequency Simulation

3.4.3 1/8 Frequency Simulation

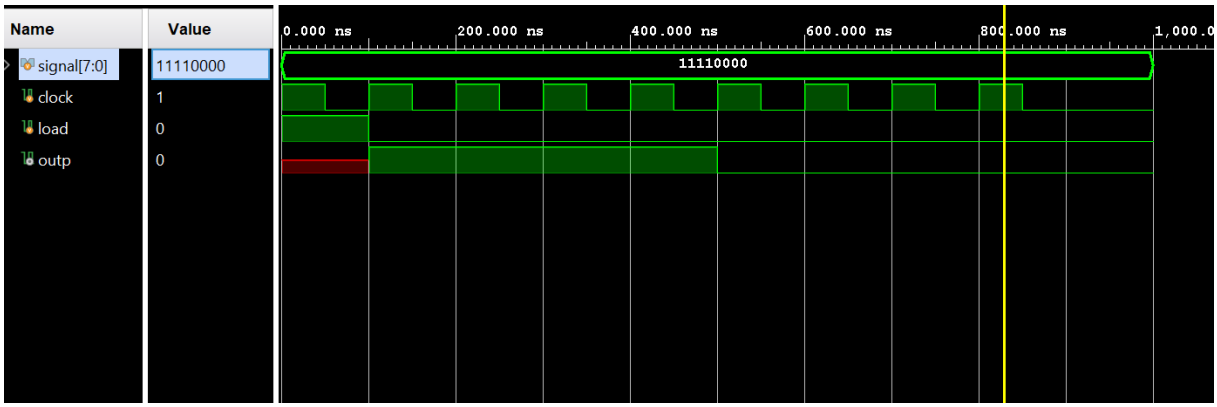

Figure 10: 1/8 Frequency Simulation

3.4.4 1/3 Pulse-Gap Simulation

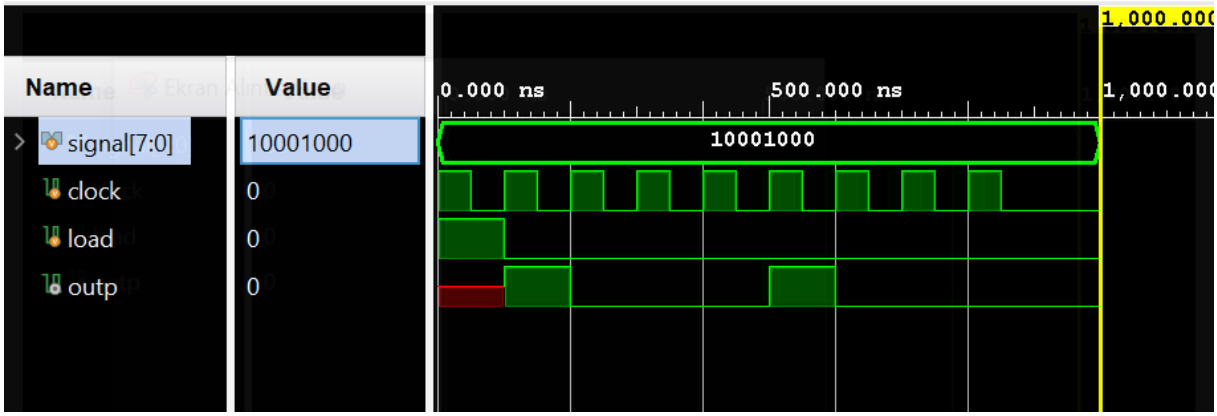

Figure 11: 1/3 Pulse-Gap Simulation

3.4.5 1/7 Pulse-Gap Simulation

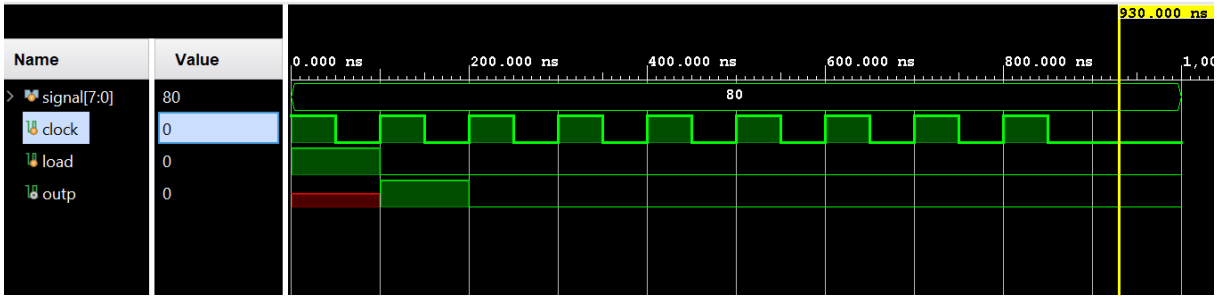

Figure 12: 1/7 Pulse-Gap Simulation

4 DISCUSSION

First of all, we have imported our simple logical units (AND Gate, OR Gate, NOT Gate) from our previous experiment after that we have designed simple SR memory unit using only NOR gates. After that, we have added an enable button to our SR latch. And then, we have implemented a slightly complex D latch module using SR latch and a NOT Gate. After that, we have designed a Negative Triggered D Flip-Flop using two D latches. Now we have all the requirements to build a 8 bit circular shifter module with pulse generator. The last complex module consists all our previously implemented modules.

5 CONCLUSION

We imported our simple modules from previous experiment and by using them we have created larger modules. We implemented inner circuity of memory units and observed their behaviour. Building larger modules we have observed that, many complex design is made by simple elements. We used enable input and CLK signal and interpreted them and in the simulations we observed that how crucial they are.

6 REFERENCES