Digital Design Lab Report

Name	Dev Goel
Roll No.	B20CS090
Experiment Number	**VIII**

Objective

(i) Create the 8bit MLS counter using LFSR.

(ii) Write a program to implement 2's complement of a running bit stream using

a. Melay Machine

b. Moore Machine

Use LFSR for generating input bit stream.

EXPERIMENT (i): Create the 8bit MLS counter using LFSR.

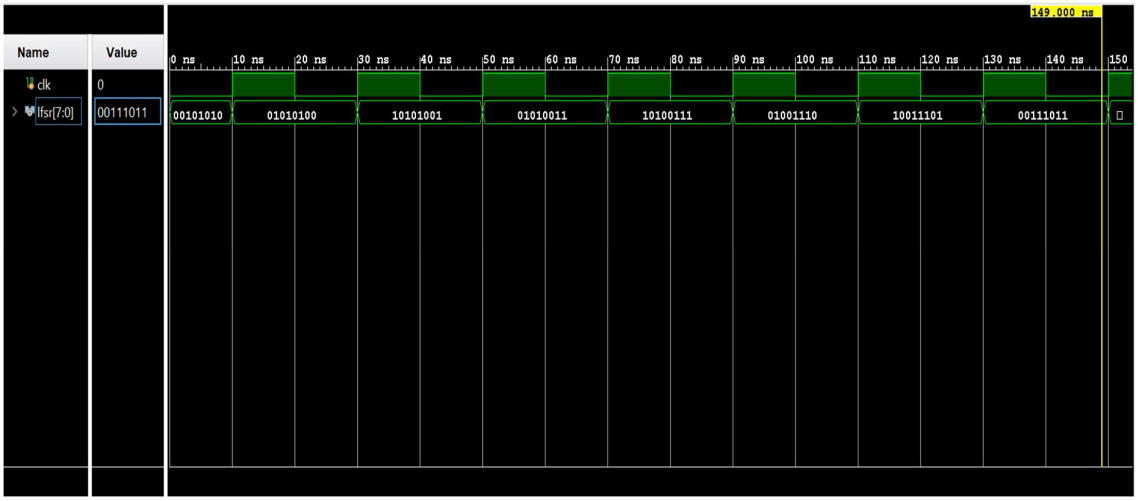
```
`timescale 1ns / 1ps
module MLS_Counter(clk, lfsr);

input clk;
output reg [7:0] lfsr = 42;

wire w1 = lfsr[3];
wire w2 = lfsr[2];
wire w3 = lfsr[1];
wire w4 = lfsr[7];

always@(posedge clk)
   begin
      lfsr[7:1] <= lfsr[6:0];
      lfsr[0] <= lfsr[3]^lfsr[4]^lfsr[5]^lfsr[7];
```

```
    end
endmodule
```

Test Bench:

```
`timescale 1ns / 1ps
module MLS_Counter_test;

reg clk;
wire [7:0] lfsr;

MLS_Counter mls(clk, lfsr);

initial begin
    clk = 0;
end

always #10 clk = ~clk;

initial #180 $finish;
endmodule
```

Waveform:



EXPERIMENT (ii) a: Write a program to implement 2's complement of a running bit stream using Melay Machine.

```
`timescale 1ns / 1ps
module melay_machine_2_complement(clk, rst, a, out, ps, ns);

input clk, rst;
input a;

output reg out;
output reg ps, ns;

parameter s0 = 1'b0;
parameter s1 = 1'b1;

always@(posedge clk, negedge rst)
begin
   if(!rst)
      ps <= s0;
   else
      ps <= ns;
end

always@(ps, a)
begin
   case(ps)
   s0: if(a)
      begin
         ns <= s1;
         out <= 1;
      end
      else
      begin
         ns<=s0;
         out<=0;
      end
   s1: if(a)
```

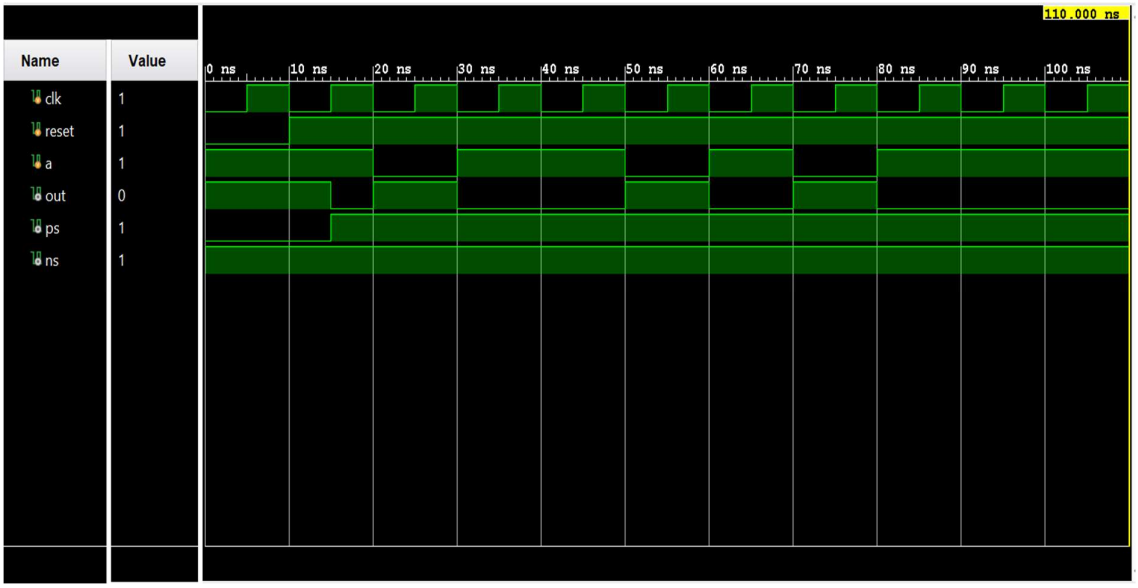
```
        begin
            out <= 0;
            ns <= s1;
        end
        else
        begin
            out <= 1;
            ns <= s1;
        end
    endcase
end
endmodule
```

Test Bench:

```
`timescale 1ns / 1ps
module test_melay_machine_2_complement;

reg clk, reset;
reg a;
wire out;
wire ps, ns;

melay_machine_2_complement mem2c(clk, reset, a, out, ps, ns);

always #5 clk = ~clk;

initial
    begin
        clk = 0;
        reset = 0;
        a = 1;
        #10 reset = 1;
        #10 a = 0;
        #10 a = 1;
        #10 a = 1;
```

```
        #10 a = 0;
        #10 a = 1;
        #10 a = 0;
        #10 a = 1;
        #10 a = 1;
        #10 a = 1;
    end

initial #110 $finish;
endmodule
```

Waveform:



EXPERIMENT (ii) b: Write a program to implement 2's complement of a running bit stream using Moore Machine.

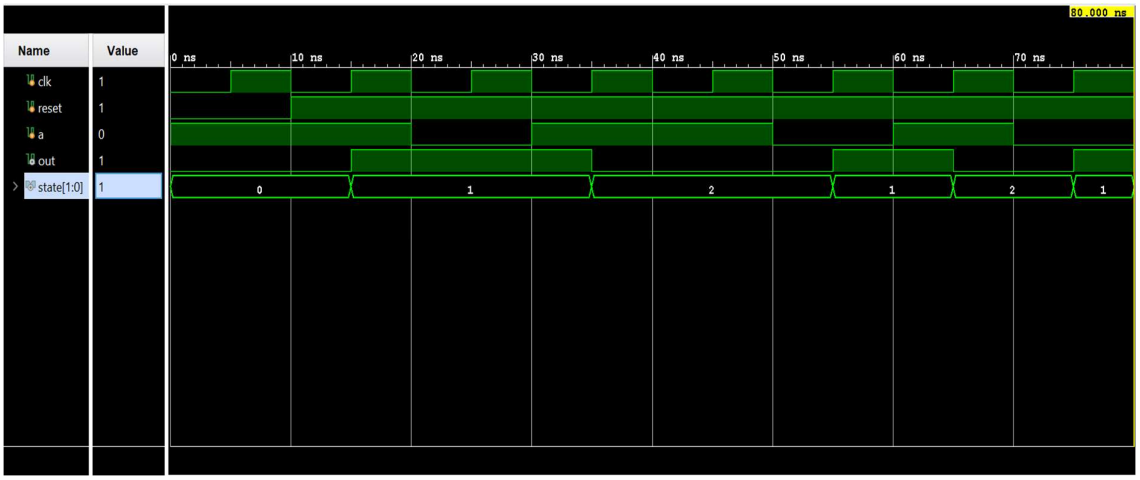
```
`timescale 1ns / 1ps
module moore_machine_2_component(clk, reset, a, out, state);

input clk, reset, a;
output reg out;
output reg [1:0] state;

integer i;
parameter s0 = 2'b00, s1 = 2'b01, s2 = 2'b10;

always @(posedge clk, negedge reset)
begin
   if(!reset)
      state<=s0;
   else
      begin
        case(state)
        s0:if(!a)
          state<=s0;
          else
          state<=s1;
        s1:if(!a)
          state<=s1;
           else
          state<=s2;
        s2:if(!a)
          state<=s1;
           else
          state<=s2;
       endcase
     end
end

always @(state)
```

```
    begin
      case(state)
      s0: out<=0;
      s1: out<=1;
      s2: out<=0;
      endcase
    end

endmodule
```

Test Bench:

```
`timescale 1ns / 1ps
module test_moore_machine_2_complement();

reg clk ,reset;
reg a;
wire out;
wire [1:0] state;
moore_machine_2_component mom2c(clk, reset, a, out, state);

always #5 clk = ~clk;

initial begin
    clk = 0;
    reset = 0;
    a = 1;
    #10 reset = 1;
    #10 a = 0;
    #10 a = 1;
    #10 a = 1;
    #10 a = 0;
    #10 a = 1;
    #10 a = 0;
end
initial #80 $finish;
endmodule
```

Waveform:


80.000 ns
Name
Value
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
clk
1
reset
1
a
0
out
1
state[1:0]
1
0
1
2
1
2
1