

# DIGITAL DESIGN

## LAB-8

### 1. MLS using LFSR:

Logic Code:

```
module mls(A,reset,clk,O);  
input [1:8] A;  
input reset,clk;  
output O;  
reg [1:8] X;  
wire V;  
  
always @(posedge reset)  
begin  
    X[1]<=A[1];  
    X[2]<=A[2];  
    X[3]<=A[3];  
    X[4]<=A[4];  
    X[5]<=A[5];  
    X[6]<=A[6];  
    X[7]<=A[7];  
    X[8]<=A[8];  
end  
  
always @(posedge clk)  
begin  
    X[1]<=V;  
    X[2]<=X[1];  
    X[3]<=X[2];  
    X[4]<=X[3];  
    X[5]<=X[4];  
    X[6]<=X[5];  
    X[7]<=X[6];  
    X[8]<=X[7];  
end  
  
assign V=X[8]^X[6]^X[5]^X[4];  
assign O=X[8];  
endmodule
```

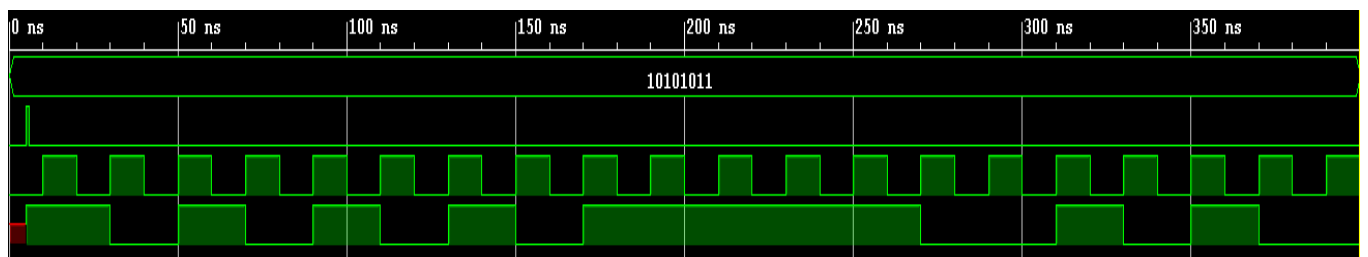
Reset and load an  
initial number to  
begin in Shift Register

Shifting process in SR

Assigning the first bit  
of SR and the output

### Test Bench:

```
module test_mls();  
    reg [1:8] A;  
    reg reset, clk;  
    wire O;  
  
    mls MLS(A, reset, clk, O);  
  
initial  
begin  
    A=8'b10101011; reset=0; clk=0;  
    #5 reset=1; #1 reset=0; #4 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
    #10 clk=0; #10 clk=1;  
end  
initial #400 $finish;  
endmodule
```



## 2.2's Compliment:

### a. Melay:

#### Logic Code:

```
module melay_2s_compliment(X,load,ctrl,clk,Y);
input [3:0]X;
input load,ctrl,clk;
output reg [3:0] Y;
wire D,V;
reg A=0;

always @(posedge clk)
begin
    if(load)
    begin
        Y[0]=X[0];
        Y[1]=X[1];
        Y[2]=X[2];
        Y[3]=X[3];
    end
    if(ctrl)
    begin
        A<=D;
        Y[0]<=Y[1];
        Y[1]<=Y[2];
        Y[2]<=Y[3];
        Y[3]<=V;
    end
end

assign D= A | Y[0];
assign V= Y[0]^A;
endmodule
```

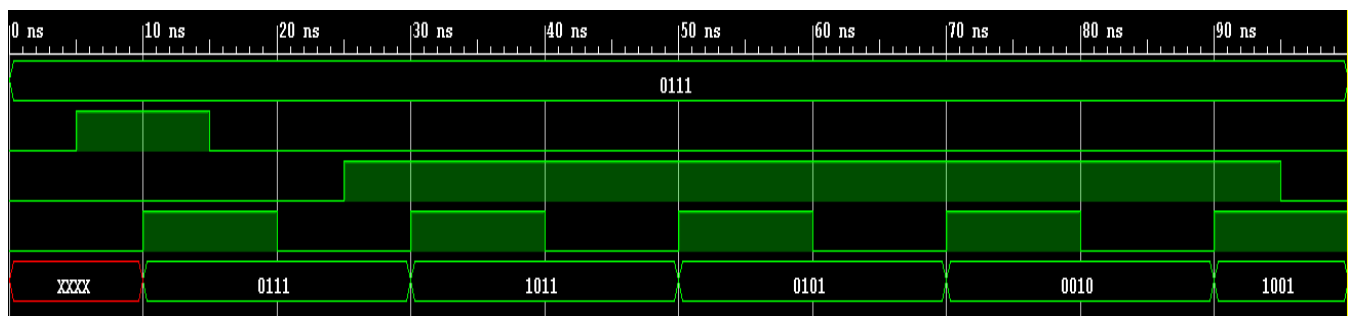
Load the number  
in a Shift Register

Shifting process in SR  
and flipflop process

Assigning D for next  
state and the first bit  
of SR after next clock

## Test Bench:

```
module test_melay();  
  reg [3:0] X;  
  reg load,ctrl,clk;  
  wire [3:0] Y;  
  melay_2s_compliment mB(X,load,ctrl,clk,Y);  
  
  initial  
  begin  
    X=4'b0111; load =0; ctrl=0; clk=0;  
    #5 load=1;  
    #5 clk=1; #5 load=0;  
    #5 clk=0; #5 ctrl=1;  
    #5 clk=1;  
    #10 clk=0;  
    #10 clk=1;  
    #10 clk=0;  
    #10 clk=1;  
    #10 clk=0;  
    #10 clk=1; #5 ctrl=0;  
  end  
  initial #100 $finish;  
endmodule
```



## b. Moore:

### Logic Code:

```
module moore_2s_compliment(X,load,ctrl,clk,Y);
input [3:0]X;
input load,ctrl,clk;
output reg [3:0] Y;
wire D1,D2;
reg A=0,B=0;

always @(posedge clk)
begin
    if(load)
    begin
        Y[0]=X[0];
        Y[1]=X[1];
        Y[2]=X[2];
        Y[3]=X[3];
    end

    if(ctrl)
    begin
        A<=D1;
        B<=D2;
        Y[0]<=Y[1];
        Y[1]<=Y[2];
        Y[2]<=Y[3];
        Y[3]<=B;
    end
end

assign D1=Y[0] & (A | B);
assign D2=Y[0] ^ (A | B);
endmodule
```

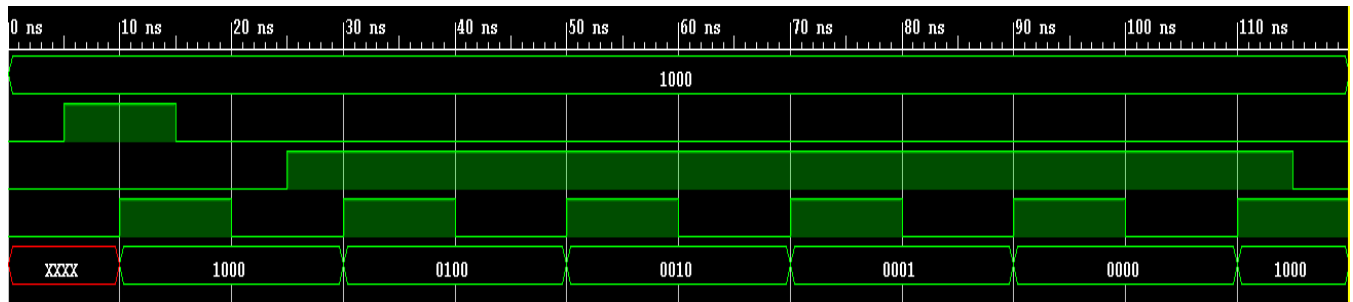
Load the number  
in a Shift Register

Shifting process in SR  
and flipflop process

Assigning D1 & D2  
for next state

## Test Bench:

```
module test_moore();  
  reg [3:0] X;  
  reg load,ctrl,clk;  
  wire [3:0] Y;  
  moore_2s_compliment mA(X,load,ctrl,clk,Y);  
  
  initial  
  begin  
    X=4'd8; load =0; ctrl=0; clk=0;  
    #5 load=1;  
    #5 clk=1; #5 load=0;  
    #5 clk=0; #5 ctrl=1;  
    #5 clk=1;  
    #10 clk=0;  
    #10 clk=1;  
    #10 clk=0;  
    #10 clk=1;  
    #10 clk=0;  
    #10 clk=1;  
    #10 clk=0;  
    #10 clk=1; #5 ctrl=0;  
  end  
  initial #120 $finish;  
endmodule
```



Melay takes 4 clock cycles because its output changes with input while Moore takes 5 clock cycles as the output depends only on the present states so output reaches the SR 1 cycle after it is calculated.