# DIGITAL DESIGN

## LAB-7 REPORT

## 1. Serial Multiplier:

Logic Code:

```
module full_adder(A,B,C,S,Carry);...


module four_bit_adder(A,B,S,C);...


module serial_multiplier(A,B,clk,ctrl,load,P,Q,M,PQ);
input [3:0] A,B;
input clk,ctrl,load;
output reg [3:0] P,Q,M;
output [7:0] PQ;
wire [3:0] D1,D2,D3,R,S,U;
wire V;

assign D1[3]=load?A[3]:Q[3];
assign D1[2]=load?A[2]:Q[2];
assign D1[1]=load?A[1]:Q[1];
assign D1[0]=load?A[0]:Q[0];

assign D2[3]=load?B[3]:M[3];
assign D2[2]=load?B[2]:M[2];
assign D2[1]=load?B[1]:M[1];
assign D2[0]=load?B[0]:M[0];

assign D3[3]=load?0:V;
assign D3[2]=load?0:U[3];
assign D3[1]=load?0:U[2];
assign D3[0]=load?0:U[1];

always @(posedge clk)
begin
    Q[0]=D1[0];
    Q[1]=D1[1];
    Q[2]=D1[2];
    Q[3]=D1[3];

    M[0]=D2[0];
    M[1]=D2[1];
    M[2]=D2[2];
    M[3]=D2[3];
end
```

Multiplexer for Parallel Loading of Data/Input to Shift Registers

Multiplexer for Parallel Loading of Output of four-bit adder to Shift Register

Parallel Loading of Data/Input to Shift Registers

```
assign R[3]=M[3]&Q[0];
assign R[2]=M[2]&Q[0];
assign R[1]=M[1]&Q[0];
assign R[0]=M[0]&Q[0];
```

AND operation of LSB of one number with all bits of other number

```
four_bit_adder FBA(R,P,U,V);

always @(posedge clk)
begin
    if(ctrl)
    begin
        Q[0]=Q[1];
        Q[1]=Q[2];
        Q[2]=Q[3];
        Q[3]=U[0];
    end

    P[0]=D3[0];
    P[1]=D3[1];
    P[2]=D3[2];
    P[3]=D3[3];
end
```

Shifting Process

```
assign PQ[0]=Q[0];
assign PQ[1]=Q[1];
assign PQ[2]=Q[2];
assign PQ[3]=Q[3];
assign PQ[4]=P[0];
assign PQ[5]=P[1];
assign PQ[6]=P[2];
assign PQ[7]=P[3];
```

Display Final Output

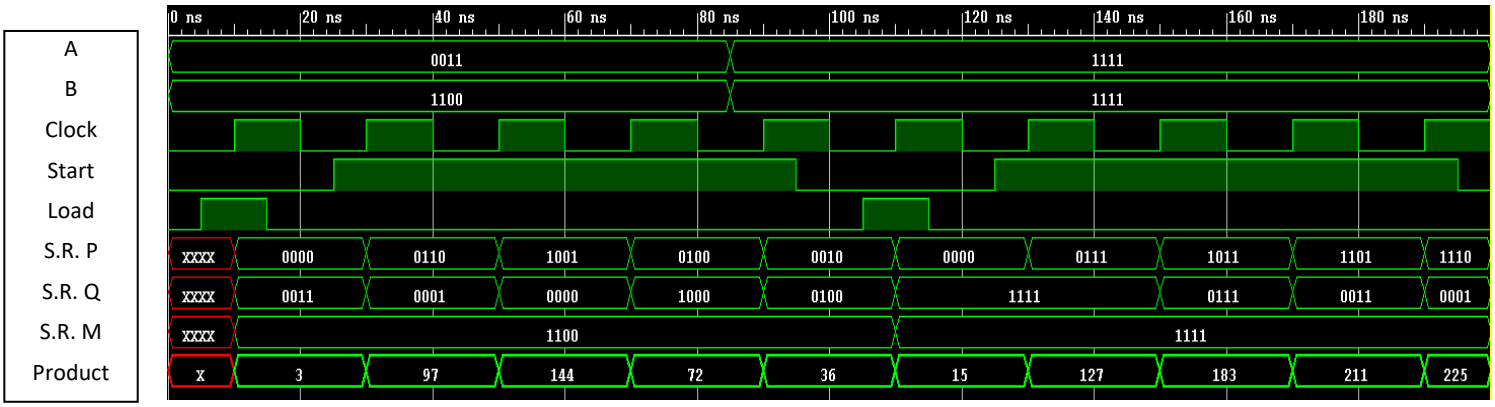
```
endmodule
```

## Test Bench:

```
module test_serial_multiplier();
reg [3:0] A,B;
reg clk,ctrl,load;
wire [3:0] P,Q,M;
wire [7:0] PQ;
serial_multiplier SM(A,B,clk,ctrl,load,P,Q,M,PQ);

initial
fork
    A=4'd3;B=4'd12;clk=0;ctrl=0;load=0;
    #5 load=1;
    #10 clk=1; #15 load=0;
    #20 clk=0; #25 ctrl=1;
    #30 clk=1;
    #40 clk=0;
    #50 clk=1;
    #60 clk=0;
    #70 clk=1;
    #80 clk=0; #85 A=4'd15; #85 B=4'd15;
    #90 clk=1; #95 ctrl=0;
    #100 clk=0; #105 load=1;
    #110 clk=1; #115 load=0;
    #120 clk=0; #125 ctrl=1;
```

```
        #130 clk=1;
        #140 clk=0;
        #150 clk=1;
        #160 clk=0;
        #170 clk=1;
        #180 clk=0;
        #190 clk=1; #195 ctrl=0;
    join
    initial #200 $finish;
    endmodule
```




Load used to load the input to the Shift Registers. As seen the input changes at 85 ns, but gets loaded to the S.R.s when load is high.
Start is used to begin the multiplication process.
The process requires 5 cycles of clock to complete (1 to load and 4 to do multiplication).

# 2. Electronic Voting Machine:

## Logic Code:

```
module evm(admin,C1,C2,reset,led1,led2,led,invalid,X1,Y1,Z1,X2,Y2,Z2);
input admin,C1,C2,reset;
output reg led1=0,led2=0,led=0,invalid=0;
reg ctrl=0;
reg [7:0] CA=0,CB=0;
output [3:0] X1,Y1,Z1,X2,Y2,Z2;

always @(posedge admin)
begin
    ctrl=1;
    invalid=0;
end
```

Admin Control

```
always @(posedge C1 or posedge C2)
begin
    if(C1 & !C2 & ctrl)
    begin
        led1=1;
        ctrl=0;
        led=1;
        invalid=0;
        CA=CA+1;
    end
    else if(C2 & !C1 & ctrl)
    begin
        led2=1;
        ctrl=0;
        led=1;
        invalid=0;
        CB=CB+1;
    end
    else if((C1 & C2 & ctrl) | (!ctrl))
        invalid=1;
end
```

Counting and displaying the vote on pressing the respective button

```
assign Z1=CA%(4'd10);
assign Y1=(CA%7'd100)/(4'd10);
assign X1=CA/(7'd100);
assign Z2=CB%(4'd10);
assign Y2=(CB%7'd100)/(4'd10);
assign X2=CB/(7'd100);
```

Displaying the count in BCD form

XYZ

```
always @(posedge reset)
begin
    led1=0;
    led2=0;
    led=0;
    invalid=0;
    ctrl=0;
end

endmodule
```

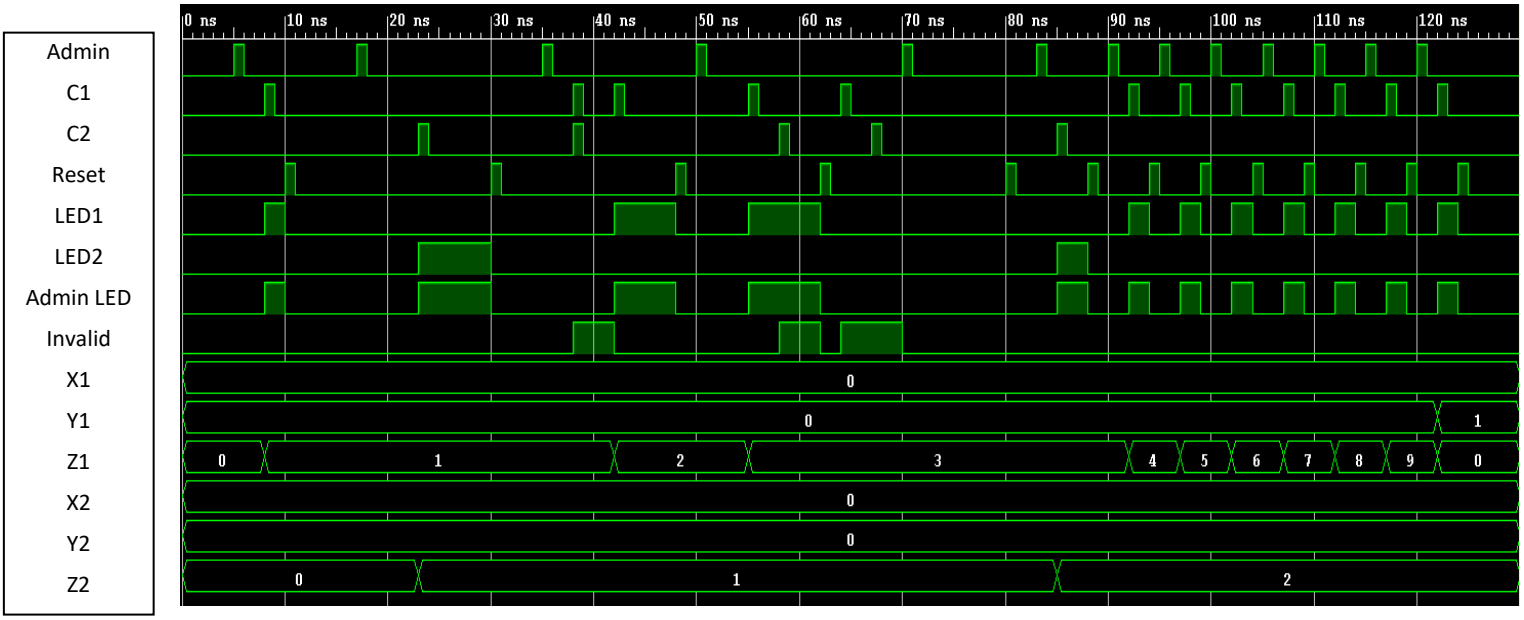
Reset the LEDs, control operation and invalid to 0

## Test Bench:

```
module test_evm();
reg admin,C1,C2,reset;
wire led1,led2,led,invalid;
wire [3:0] X1,Y1,Z1,X2,Y2,Z2;
evm EVM(admin,C1,C2,reset,led1,led2,led,invalid,X1,Y1,Z1,X2,Y2,Z2);

initial
begin
    admin=0; C1=0; C2=0; reset=0; #5 admin=1; #1 admin=0;
    #2 C1=1; #1 C1=0;
    #1 reset=1; #1 reset=0;
    #6 admin=1; #1 admin=0; #5 C2=1; #1 C2=0;
    #6 reset=1; #1 reset=0;
    #4 admin=1; #1 admin=0;
    #2 C1=1; C2=1; #1 C1=0; C2=0;
    #3 C1=1; #1 C1=0;
    #5 reset=1; #1 reset=0;
```

```
        #1 admin=1; #1 admin=0; #4 C1=1; #1 C1=0;
        #2 C2=1; #1 C2=0;
        #3 reset=1; #1 reset=0;
        #1 C1=1; #1 C1=0; #2 C2=1; #1 C2=0;
        #2 admin=1; #1 admin=0; #9 reset=1; #1 reset=0;
        #2 admin=1; #1 admin=0; #1 C2=1; #1 C2=0;
        #2 reset=1; #1 reset=0;
        #1 admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
        admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
        admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
        admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
        admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
        admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
        admin=1; #1 admin=0; #1 C1=1; #1 C1=0; #1 reset=1; #1 reset=0;
    end
    initial #130 $finish;
    endmodule
```




Admin is used to give control after which the vote can be casted.
If voted without Admin control the vote will not be counted and invalid LED will glow which will notify the admin. (as seen around 65 ns)
If one tries to press both the buttons at same time then also the invalid LED will start glowing. (As seen between 35-40 ns)
If after casting vote one tries to vote another time then the second vote will not be counted and the admin will be notified by invalid LED. (As seen between 55-60 ns)
After giving the vote the LED of the corresponding candidate will glow and the admin will be notified with the Admin LED.
Reset is used to turn off the LEDs.
If one does not vote for long time the Admin LED will not glow giving signal to Admin that the voter is not giving the vote. (As seen between 70-80 ns)