LAB REPORT-B20EE016

Digital Design - EEL2020

Dhruv

B20EE016

Serial Multiplier

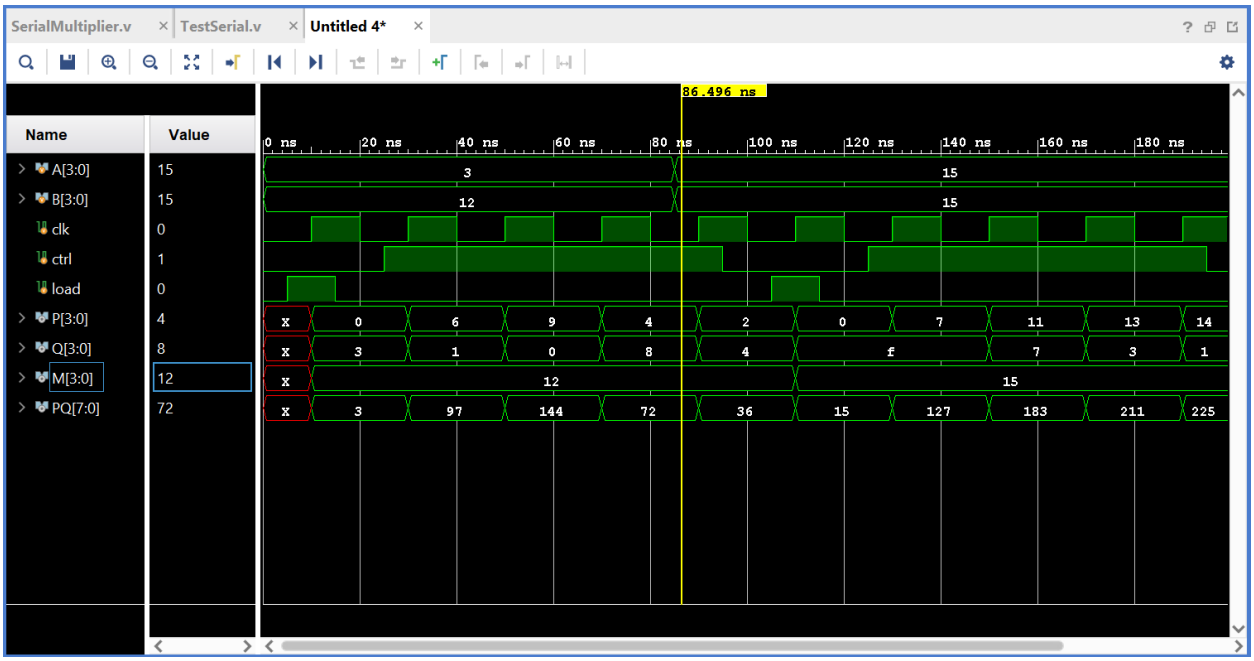
Code

```
`timescale 1ns / 1ps
module Half_Adder(a,b,sum,carry );
input a,b;
output sum ,carry;
assign sum=a^b;
assign carry=a&b;
endmodule

module Full_Adder(a,b,c,sum,carry);
    input a,b,c;
    output sum,carry;
    wire sum1,carry1,carry2;
    Half_Adder H1(b,c,sum1,carry1);
    Half_Adder H2(a,sum1,sum,carry2);
    assign carry=carry1|carry2;
endmodule



module FourBitAdder(A,B,Sum,Carry);
input [3:0] A,B;
output [3:0] Sum;
output Carry;
wire [3:1] D;
Full_Adder F1(A[0],B[0],0,Sum[0],D[1]);
Full_Adder F2(A[1],B[1],D[1],Sum[1],D[2]);
Full_Adder F3(A[2],B[2],D[2],Sum[2],D[3]);
Full_Adder F4(A[3],B[3],D[3],Sum[3],Carry);
endmodule



module SerialMultiplier(A,B,clk,ctrl,load,P,Q,M,Product);
```

```
input clk,ctrl,load;
input [3:0] A,B;
output [7:0] Product;
output reg [3:0] P,Q,M;
wire V;
wire [3:0] temp1,temp2,temp3,R,S,U;


assign temp2[3]=load?B[3]:M[3];
assign temp2[2]=load?B[2]:M[2];
assign temp2[1]=load?B[1]:M[1];
assign temp2[0]=load?B[0]:M[0];

assign temp3[3]=load?0:V;
assign temp3[2]=load?0:U[3];
assign temp3[1]=load?0:U[2];
assign temp3[0]=load?0:U[1];

assign temp1[3]=load?A[3]:Q[3];
assign temp1[2]=load?A[2]:Q[2];
assign temp1[1]=load?A[1]:Q[1];
assign temp1[0]=load?A[0]:Q[0];
always @(posedge clk)
begin
    Q[0]=temp1[0];
    Q[1]=temp1[1];
    Q[2]=temp1[2];
    Q[3]=temp1[3];

    M[0]=temp2[0];
    M[1]=temp2[1];
    M[2]=temp2[2];
    M[3]=temp2[3];
end

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];

FourBitAdder 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]=temp3[0];
```

```
    P[1]=temp3[1];
    P[2]=temp3[2];
    P[3]=temp3[3];
end

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

endmodule
```

Simulation



Test Bench

```
`timescale 1ns / 1ps

module test_serial_multiplier();
reg [3:0] A,B;
reg clk,ctrl,load;
```

```
wire [3:0] P,Q,M;
wire [7:0] PQ;
SerialMultiplier 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
```

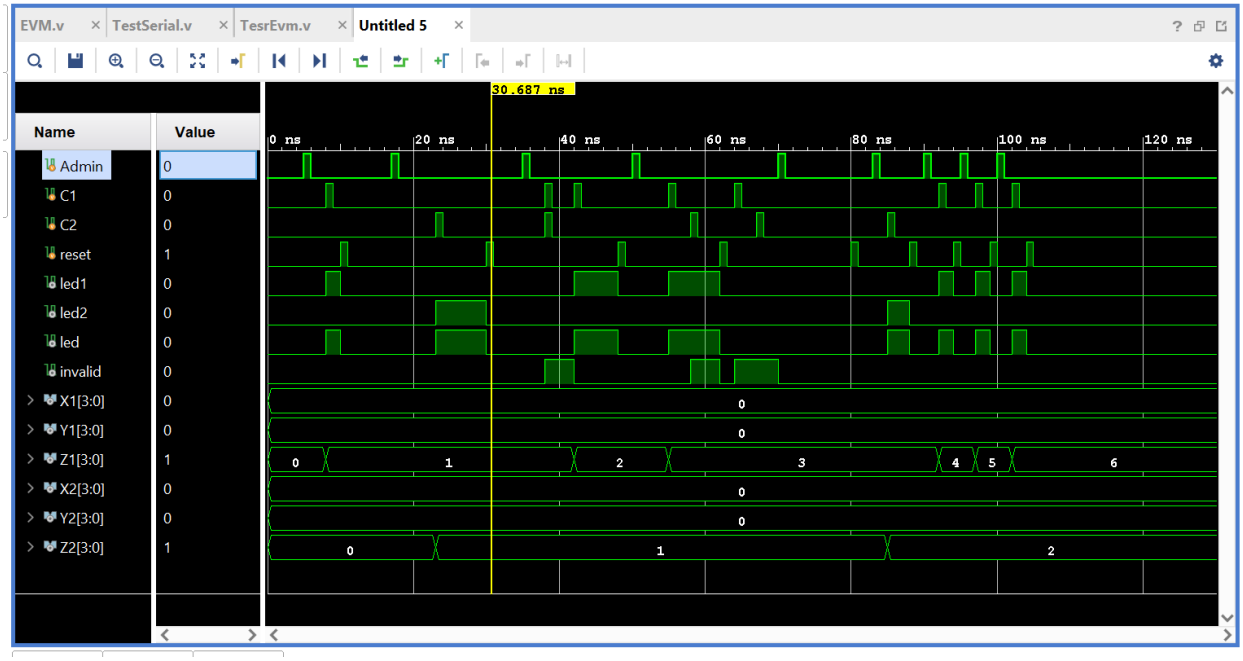
EVM

Code

```
`timescale 1ns / 1ps
module EVM(Admin,Candidate1,Candidate2,reset,led1,led2,led,invalid,X1,Y1,Z1,X2,Y2,Z2);
input Admin,Candidate1,Candidate2,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

always @(posedge Candidate1 or posedge Candidate2)
begin
    if(Candidate1 & !Candidate2 & ctrl)
    begin
        led1=1;
        ctrl=0;
        led=1;
        invalid=0;
        CA=CA+1;
    end
    else if(Candidate2 & !Candidate1 & ctrl)
    begin
        led2=1;
        ctrl=0;
        led=1;
        invalid=0;
        CB=CB+1;
    end
    else if((Candidate1 & Candidate2 & ctrl) | (!ctrl))
        invalid=1;
end

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);

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

endmodule
```

Simulation



Test Bench

```
`timescale 1ns / 1p
module test_evm();
reg Admin,C1,C2,reset;
wire led1,led2,led,invalid;
wire [3:0] X1,Y1,Z1,X2,Y2,Z2;
EVM EM(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;
end
initial #130 $finish;
endmodule
```