ECE-103 Project: 5-Bit Multiplier

❖design.sv:

```
module half_adder(S, C, A, B);
  output S, C; //S= sum, C= carry
  input A, B;
  wire S, C, A, B;
  assign S = A ^ B;
  assign C = A & B;
endmodule

module full_adder(sum, cout, i1, i2, cin);
  output sum, cout;
  input i1, i2, cin;
  wire sum, cout, i1, i2, cin;
  wire I1, I2, I3; // I1= i1 xor i2, I2= i1+i2, I3= (I1 xor I2)& cin
  half_adder ha1(I1, I2, i1, i2);
  half_adder ha2(sum, I3, I1, cin);
  assign cout = I2 || I3; // cout= final carry generated
endmodule

module
five_bit_adder(S0,S1,S2,S3,S4,Cout,A0,A1,A2,A3,A4,B0,B1,B2,B3,B4);//
this module adds two 5 bit numbers using "full adder" module
  output S0,S1,S2,S3,S4,Cout;
  input A0,A1,A2,A3,A4,B0,B1,B2,B3,B4;
  wire S0,S1,S2,S3,S4,Cout,A0,A1,A2,A3,A4,B0,B1,B2,B3,B4;
  wire c0,c1,c2,c3;
  full_adder fa0(S0,c0,A0,B0,1'b0); // initial carry to five_bit_adder=0
  full_adder fa1(S1,c1,A1,B1,c0);
  full_adder fa2(S2,c2,A2,B2,c1);
  full_adder fa3(S3,c3,A3,B3,c2);
  full_adder fa4(S4,Cout,A4,B4,c3);
```

```
endmodule

module
five_bit_multiplier(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,X0,X1,X2,X3,X4,Y0,Y
1,Y2,Y3,Y4);
 output D0,D1,D2,D3,D4,D5,D6,D7,D8,D9; // final output= D9 D8 D7 D6
D5 D4 D3 D2 D1 D0
 input X0,X1,X2,X3,X4,Y0,Y1,Y2,Y3,Y4;
 wire D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,X0,X1,X2,X3,X4,Y0,Y1,Y2,Y3,Y4;
 wire
M0,M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15,
M16,M17,M18,M19,M20,M21,M22,M23,M24,M25,M26,M27,M28,M29,
M30;

 // generating all the product terms

 assign M0=X0 & Y0;
 assign M1=X1 & Y0;
 assign M2=X2 & Y0;
 assign M3=X3 & Y0;
 assign M4=X4 & Y0;
 assign M5=X0 & Y1;
 assign M6=X1 & Y1;
 assign M7=X2 & Y1;
 assign M8=X3 & Y1;
 assign M9=X4 & Y1;
 assign M10=X0 & Y2;
 assign M11=X1 & Y2;
 assign M12=X2 & Y2;
 assign M13=X3 & Y2;
 assign M14=X4 & Y2;
 assign M15=X0 & Y3;
 assign M16=X1 & Y3;
 assign M17=X2 & Y3;
 assign M18=X3 & Y3;
 assign M19=X4 & Y3;
 assign M20=X0 & Y4;
```

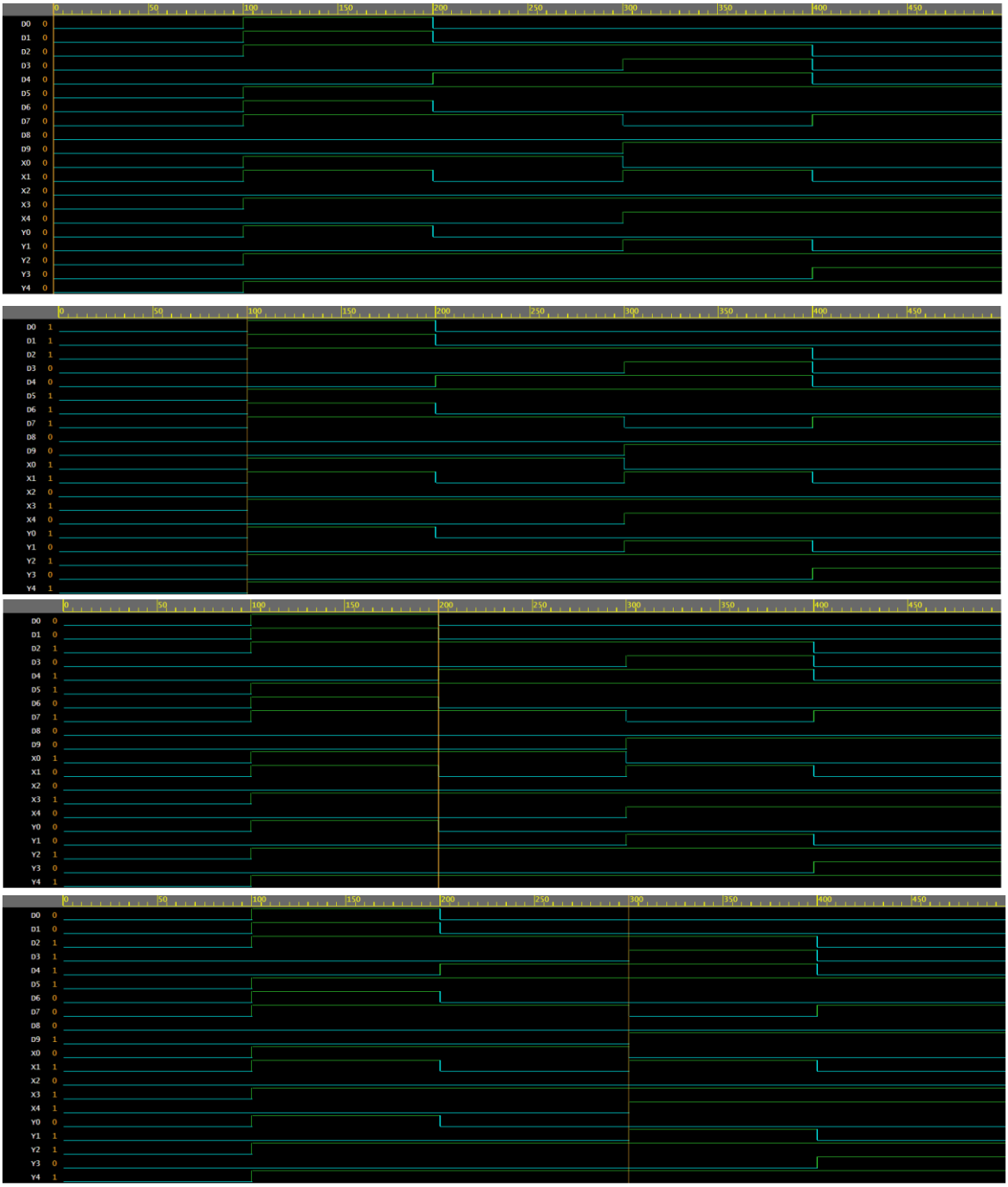
```
  assign M21=X1 & Y4;
  assign M22=X2 & Y4;
  assign M23=X3 & Y4;
  assign M24=X4 & Y4;

  assign D0=M0;
  wire Z00,Z01,Z02,Z03,Z04,C0;// C0 Z04 Z03 Z02 Z01 Z00 = (0 X4Y0 X3Y0
X2Y0 X1Y0)+(X4Y1 X3Y1 X2Y1 X1Y1 X0Y1)
  five_bit_adder
f0(Z00,Z01,Z02,Z03,Z04,C0,M1,M2,M3,M4,1'b0,M5,M6,M7,M8,M9);
  assign D1=Z00;
  wire Z10,Z11,Z12,Z13,Z14,C1;// C1 Z14 Z13 Z12 Z11 Z10 = (C0 Z04 Z03
Z02 Z01)+(X4Y2 X3Y2 X2Y2 X1Y2 X0Y2)
  five_bit_adder
f1(Z10,Z11,Z12,Z13,Z14,C1,M10,M11,M12,M13,M14,Z01,Z02,Z03,Z04,C0
);
  assign D2=Z10;
  wire Z20,Z21,Z22,Z23,Z24,C2;// C2 Z24 Z23 Z22 Z21 Z20 = (C1 Z14 Z13
Z12 Z11)+(X4Y3 X3Y3 X2Y3 X1Y3 X0Y3)
  five_bit_adder
f2(Z20,Z21,Z22,Z23,Z24,C2,Z11,Z12,Z13,Z14,C1,M15,M16,M17,M18,M19
);
  assign D3=Z20;
  // D4,D5 D6 D7 D8 D9 = (C2 Z24 Z23 Z22 Z21)+(X4Y4 X3Y4 X2Y4 X1Y4
X0Y4)
  five_bit_adder
f3(D4,D5,D6,D7,D8,D9,Z21,Z22,Z23,Z24,C2,M20,M21,M22,M23,M24);
endmodule
```

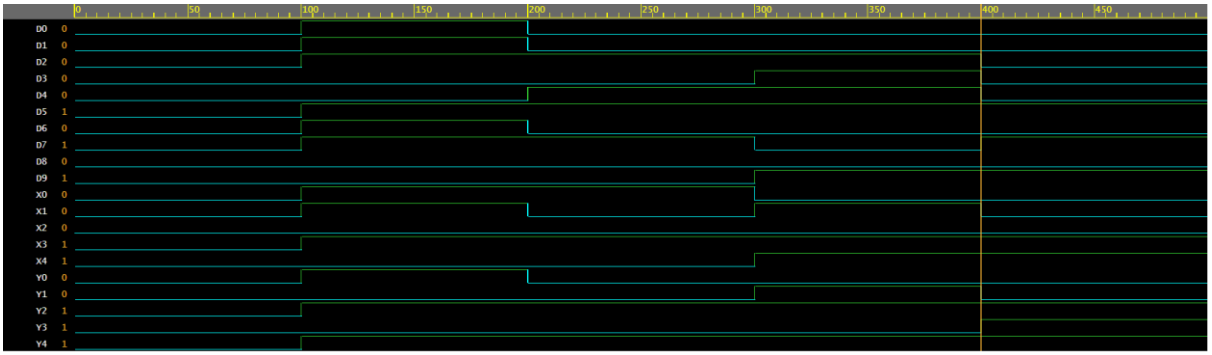
❖testbench.sv:

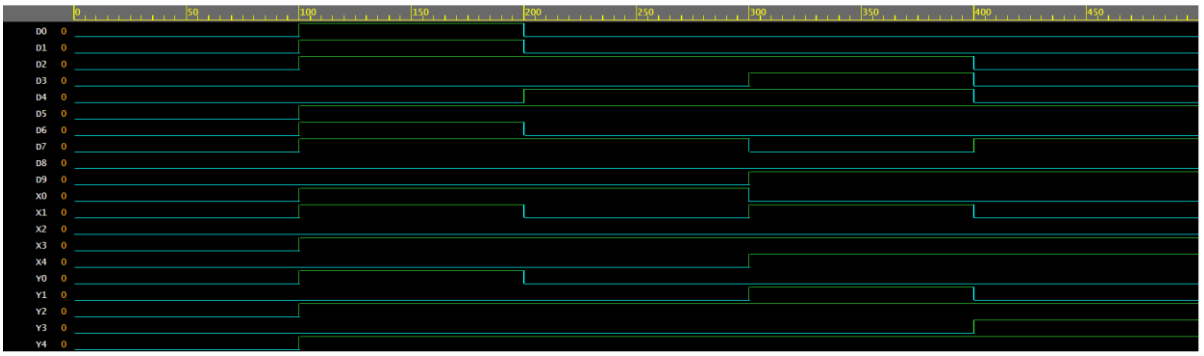
```
module five_bit_multiplier_test;
```

```
  reg X0, X1, X2, X3, X4;
  reg Y0, Y1, Y2, Y3, Y4;
  wire D0, D1, D2, D3, D4, D5, D6, D7, D8, D9;

five_bit_multiplier
five_bit_multiplier_test(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,X0,X1,X2,X3,X
4,Y0,Y1,Y2,Y3,Y4);

initial begin
   $dumpfile("dump.vcd");
   $dumpvars(1);
 end
initial begin

  X4=0;X3=0;X2=0;X1=0;X0=0; // X= 00000
  Y4=0;Y3=0;Y2=0;Y1=0;Y0=0; // Y= 00000
  #100
  X4=0;X3=1;X2=0;X1=1;X0=1; // X= 01011
  Y4=1;Y3=0;Y2=1;Y1=0;Y0=1; // Y= 10101
  #100
  X4=0;X3=1;X2=0;X1=0;X0=1; // X= 01001
  Y4=1;Y3=0;Y2=1;Y1=0;Y0=0; // Y= 10100
  #100
  X4=1;X3=1;X2=0;X1=1;X0=0; // X= 11010
  Y4=1;Y3=0;Y2=1;Y1=1;Y0=0; // Y= 10110
  #100
  X4=1;X3=1;X2=0;X1=0;X0=0; //X= 11000
  Y4=1;Y3=1;Y2=1;Y1=0;Y0=0; //Y= 11100

  #100
  X4=0;X3=0;X2=0;X1=0;X0=0; //X= 00000
  Y4=0;Y3=0;Y2=0;Y1=0;Y0=0; //Y= 00000



end
endmodule
```

❖Output simulation:




0
50
100
150
200
250
300
350
400
450
D0 0
D1 0
D2 0
D3 0
D4 0
D5 1
D6 0
D7 1
D8 0
D9 1
X0 0
X1 0
X2 0
X3 1
X4 1
Y0 0
Y1 0
Y2 1
Y3 1
Y4 1


0
50
100
150
200
250
300
350
400
450
D0 0
D1 0
D2 0
D3 0
D4 0
D5 0
D6 0
D7 0
D8 0
D9 0
X0 0
X1 0
X2 0
X3 0
X4 0
Y0 0
Y1 0
Y2 0
Y3 0
Y4 0