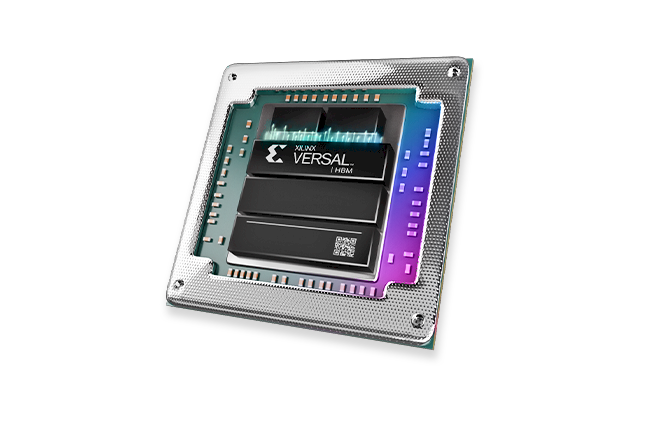
**EC204**

**Digital System Design Lab**

**Lab – 2**



**Utkarsh R Mahajan**

**201EC164, S11**

**1]** **Half adder (a) using gates (b) using dataflow model (using assign statement)**

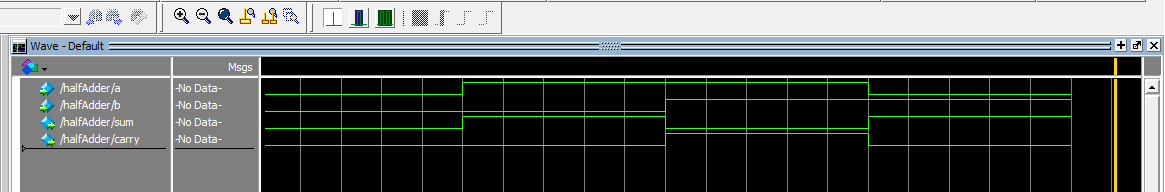
**(a) using gates**

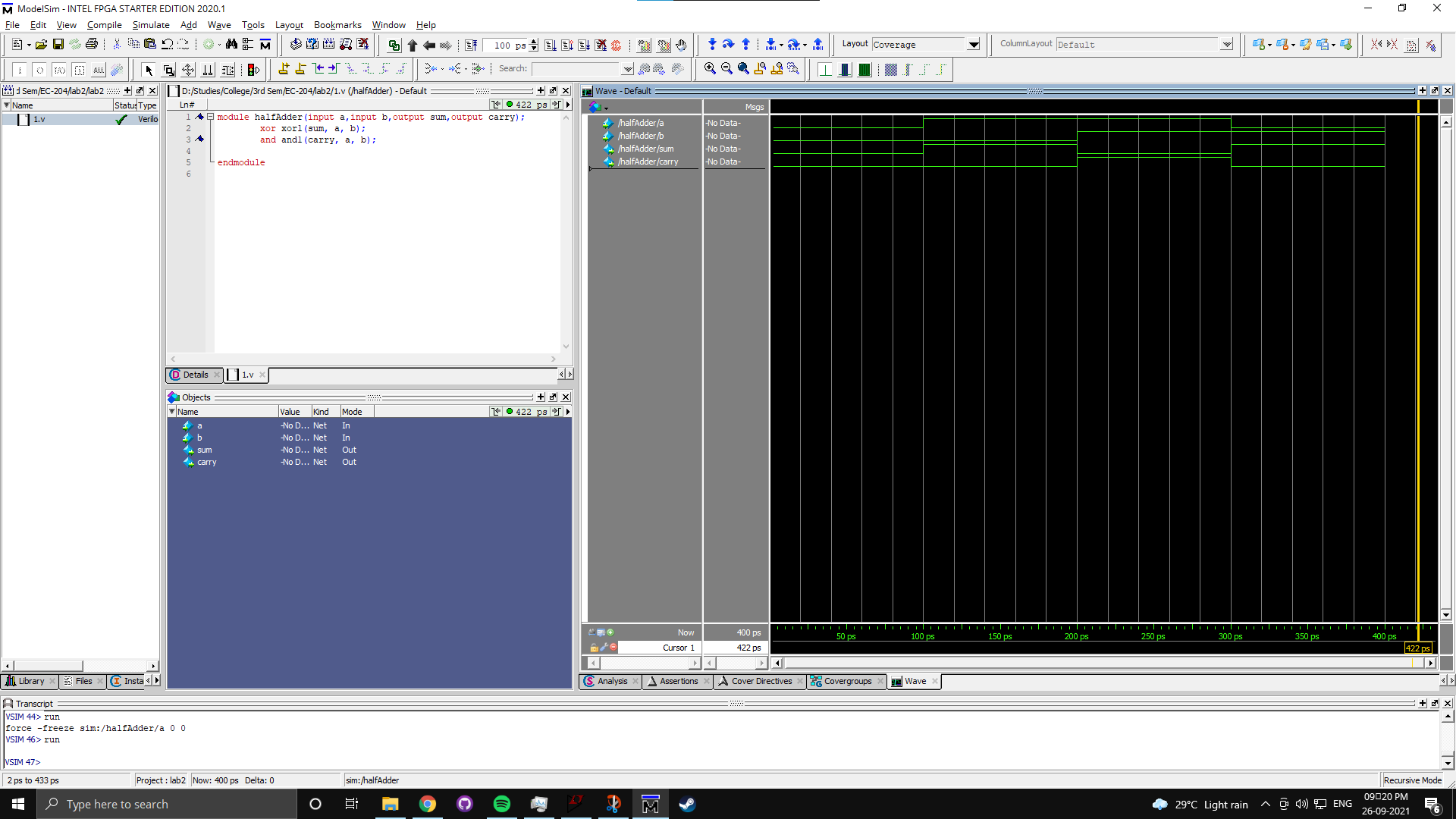
*module* halfAdder(input a,input b,output sum,output carry);

    xor xor1(sum, a, b);

    and and1(carry, a, b);

*endmodule*

****

****

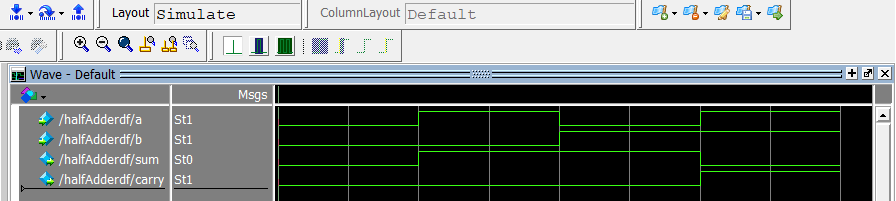
**(b) using dataflow model**

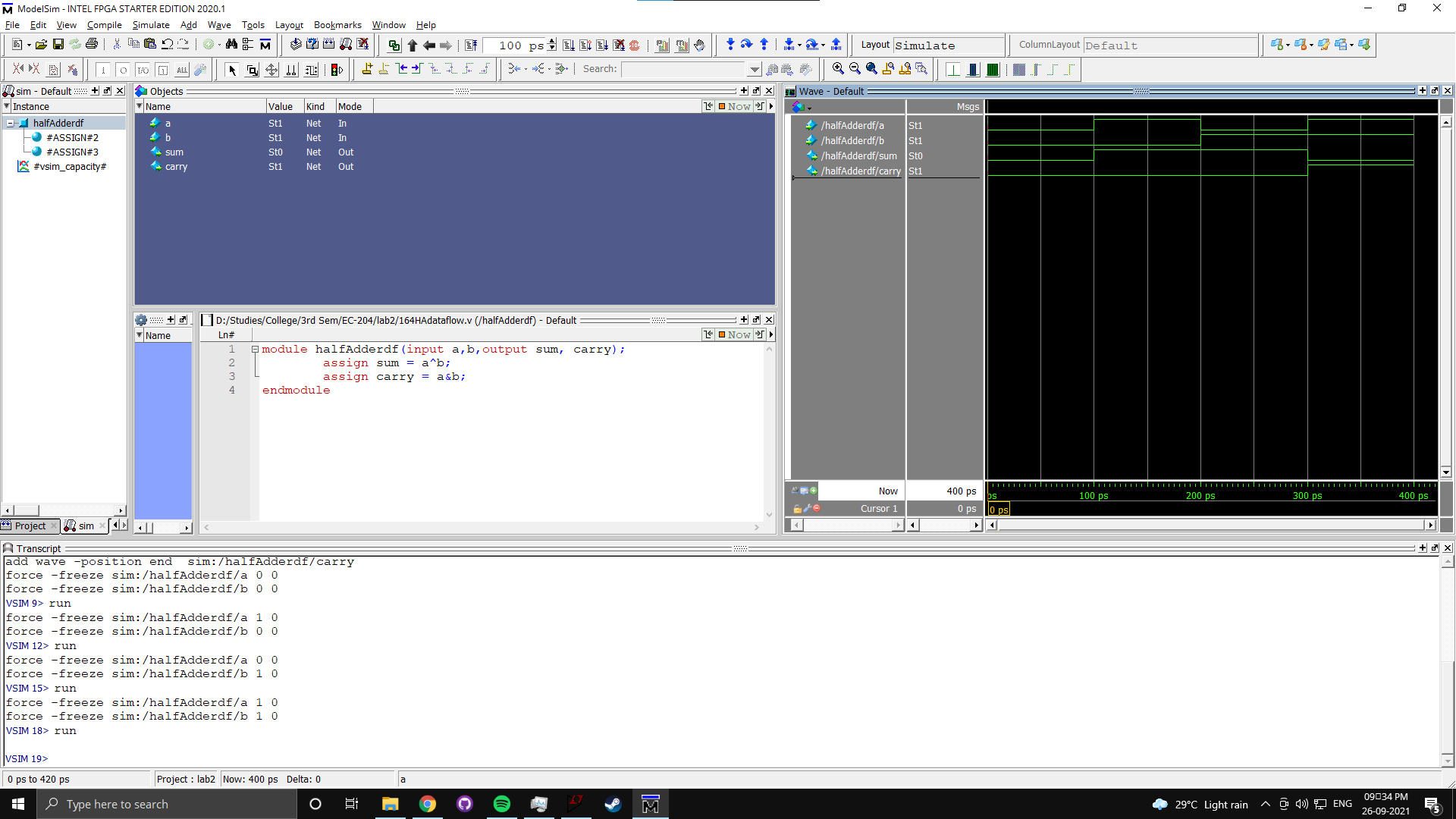
*module* halfAdderdf(input a,b,output sum, carry);

    assign sum = a^b;

    assign carry = a&b;

*endmodule*

****

****

**2]** **Full adder (a) using dataflow model (using assign statement) (b) using two half adders and an OR gate**

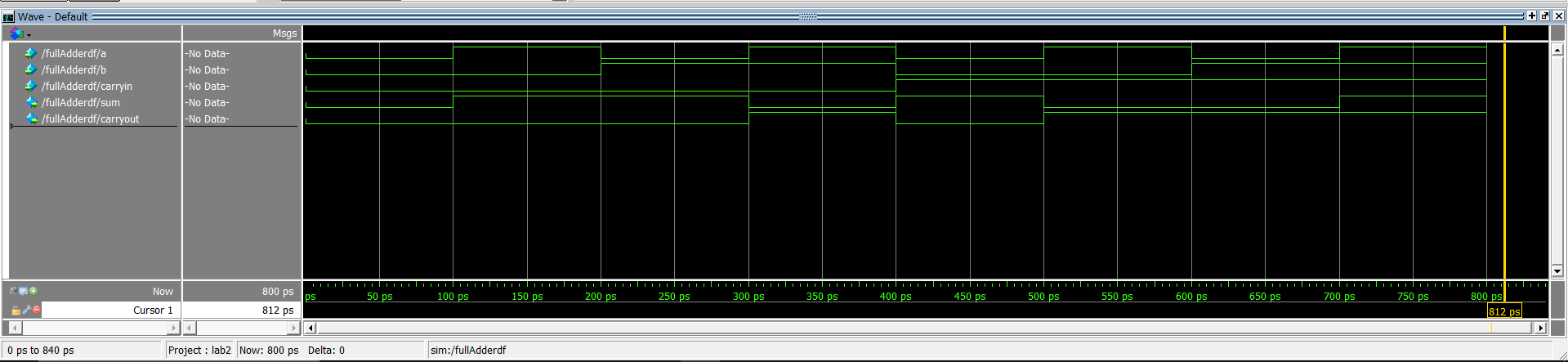
**(a) using dataflow model**

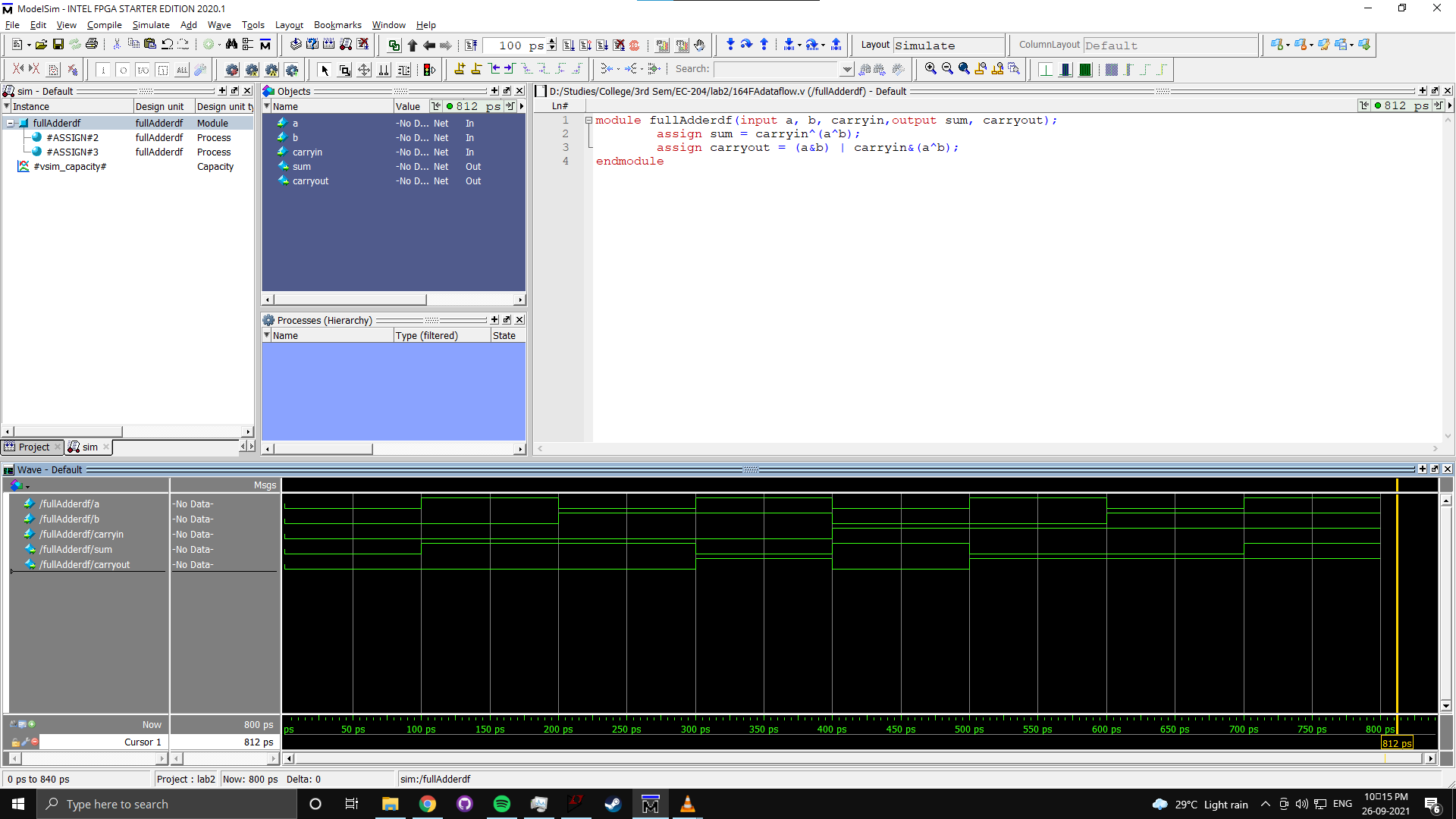
*module* fullAdderdf(input a, b, carryin,output sum, carryout);

    assign sum = carryin^(a^b);

    assign carryout = (a&b) | carryin&(a^b);

*endmodule*





**(b) using two half adders and an OR gate**

*module* HalfAdderdf(input a,b,output sum, carry);

    assign sum = a^b;

    assign carry = a&b;

*endmodule*

*module* FullAdder2haor(input a, b, carryin,output sum, carryout);

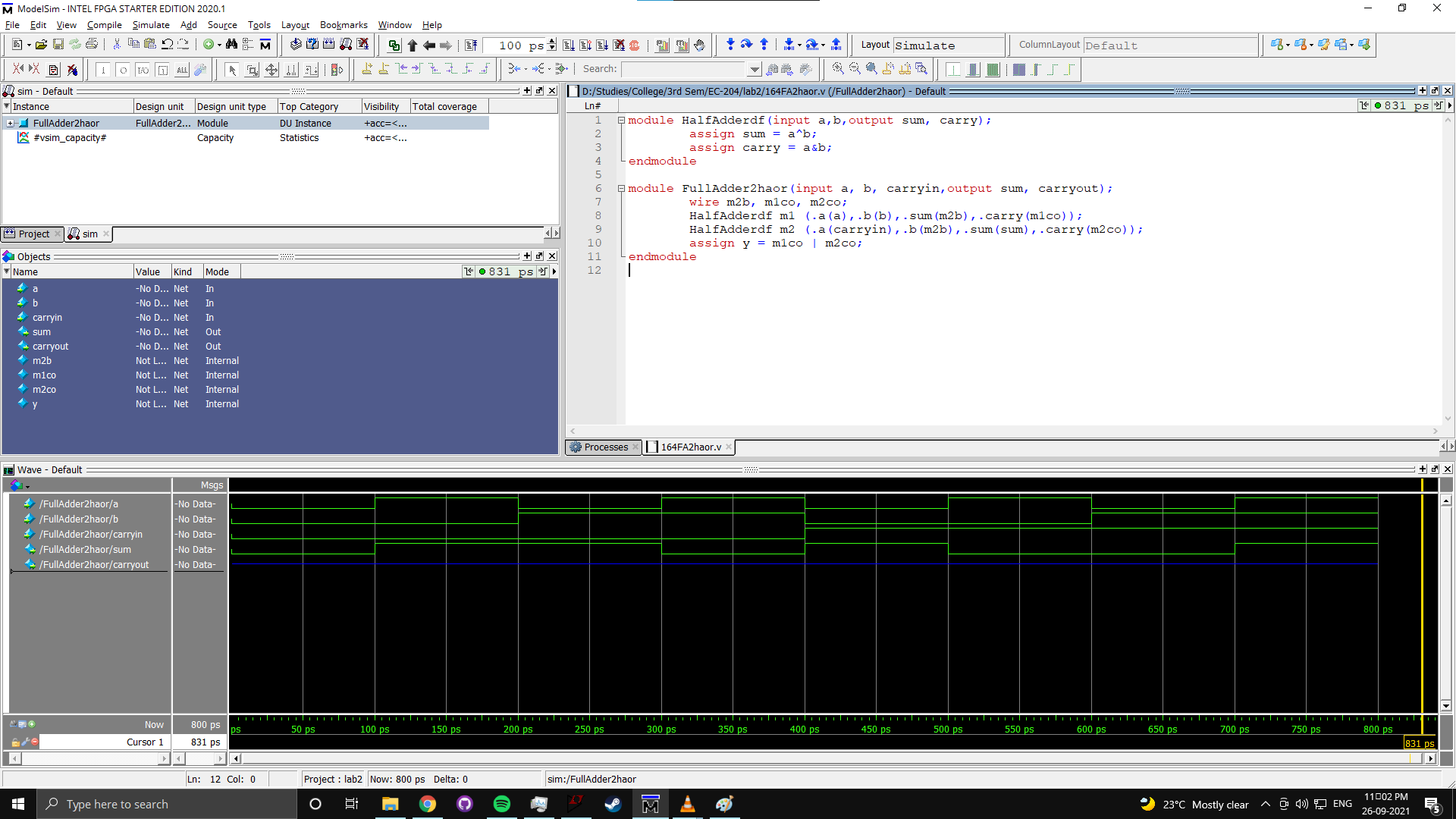
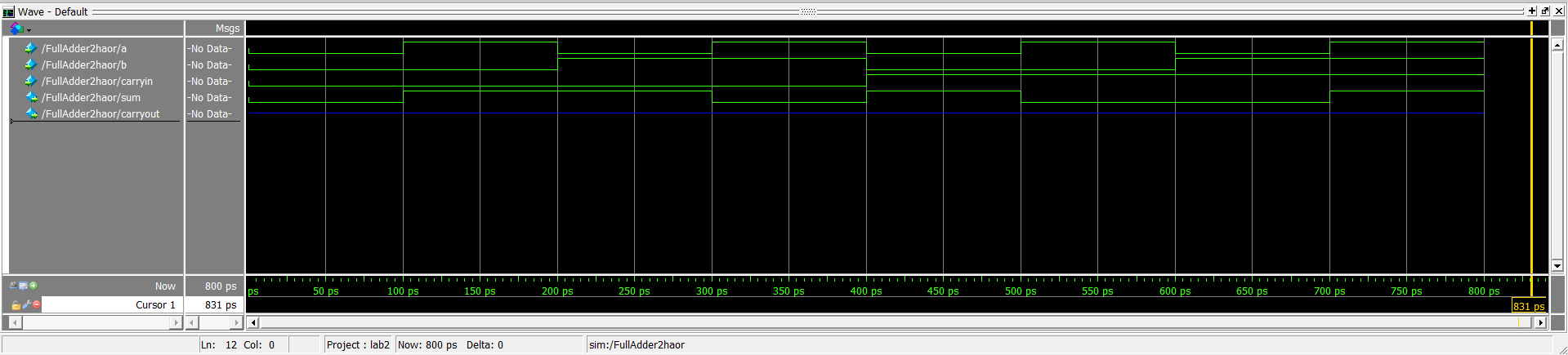
    wire m2b, m1co, m2co;

    HalfAdderdf m1 (.a(a),.b(b),.sum(m2b),.carry(m1co));

    HalfAdderdf m2 (.a(carryin),.b(m2b),.sum(sum),.carry(m2co));

    assign y = m1co | m2co;

*endmodule*



**3] Two to one multiplexer using behavioral model (using if statement)**

*module* Mux\_2\_1(input A, B, S0, output reg Y);

    always @(\*)

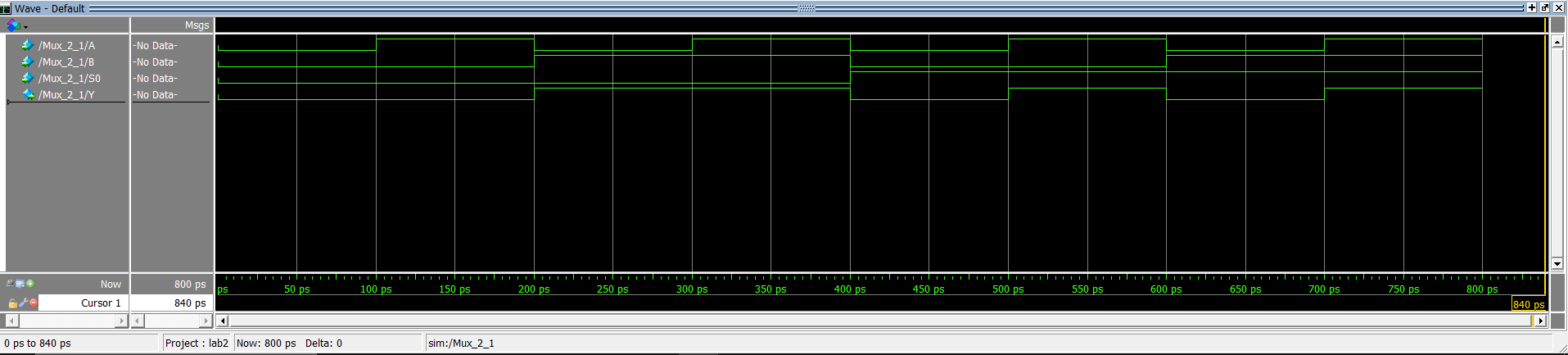
    begin

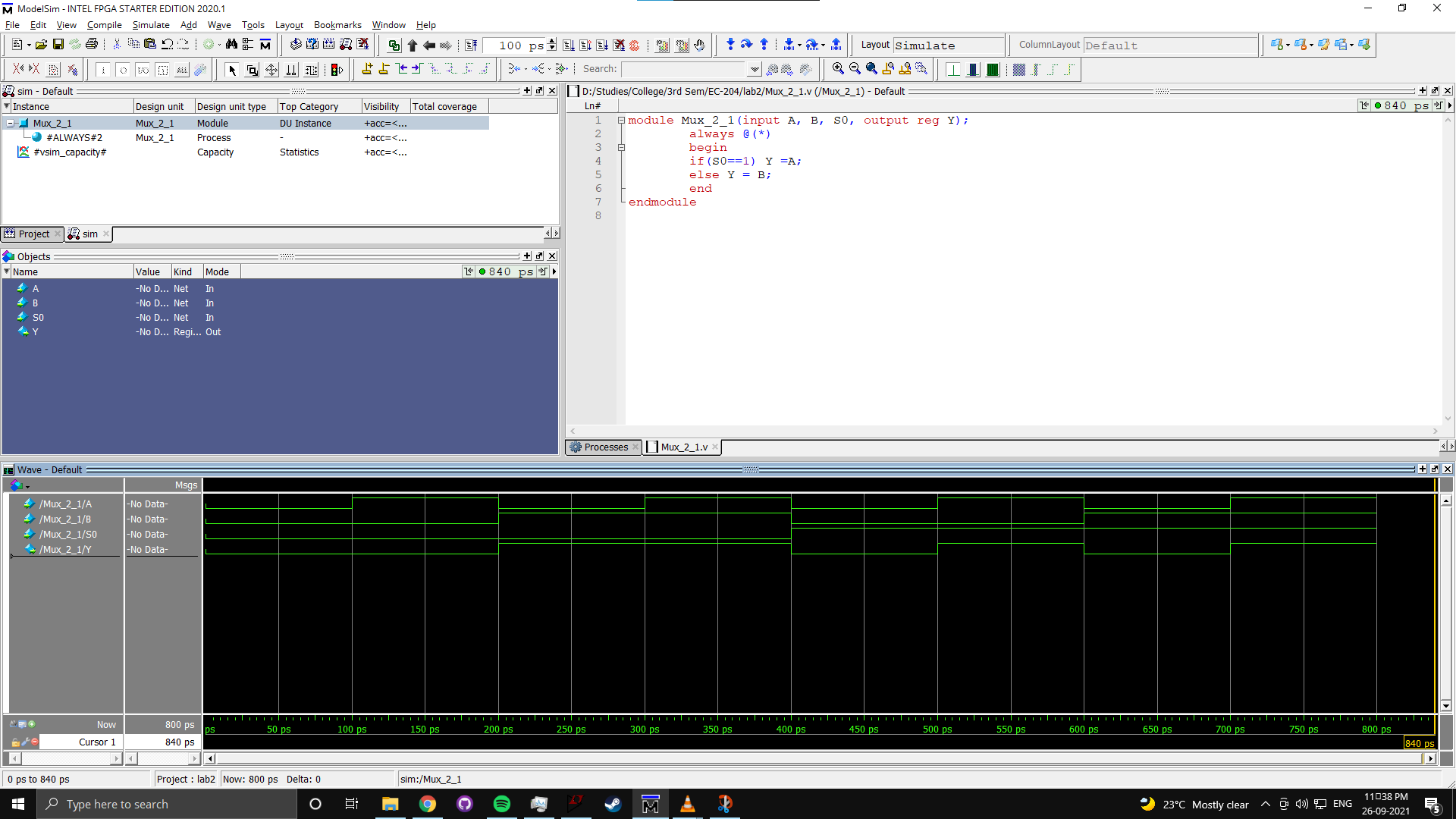
    if(S0==1) Y =A;

    else Y = B;

    end

*endmodule*

****

****

**4]** **4 bit ripple carry adder using full adder modules.**

*module* FullAdder(input A, B, Ci,output S, Co);

    assign S = Ci^(A^B);

    assign Co = (A&B) | Ci&(A^B);

*endmodule*

*module* FourBitAdder(input [3:0]A, B,

            input Ci,

            output [3:0]S,

            output Co);

    wire  C1, C2,C3;

    FullAdder FA1(.A(A[0]),.B(B[0]),.Ci(Ci),.S(S[0]),.Co(C1));

    FullAdder FA2(.A(A[1]),.B(B[1]),.Ci(C1),.S(S[1]),.Co(c2));

    FullAdder FA3(.A(A[2]),.B(B[2]),.Ci(c2),.S(S[2]),.Co(c3));

    FullAdder FA4(.A(A[3]),.B(B[3]),.Ci(c3),.S(S[3]),.Co(Co));

*endmodule*

*module* Test4bitadder();

*/\*201ec164\*/*

    reg  [1:0] ci ;

    integer ca, cb;*/\* ca and cb are counts\*/*

    wire Cout;

    wire [3:0] s;

    reg C;

    reg [3:0] A, B;

    reg [3:0] array[0:15];

    assign array[0] = 4'b0000; assign array[1] = 4'b0001;

    assign array[2] = 4'b0010; assign array[3] = 4'b0011;

    assign array[4] = 4'b0100; assign array[5] = 4'b0101;

    assign array[6] = 4'b0110; assign array[7] = 4'b0111;

    assign array[8] = 4'b1000; assign array[9] = 4'b1001;

    assign array[10] = 4'b1010; assign array[11] = 4'b1011;

    assign array[12] = 4'b1100; assign array[13] = 4'b1101;

    assign array[14] = 4'b1110; assign array[15] = 4'b1111;

    assign C=0;

    FourBitAdder test(.A(A), .B(B), .Ci(C), .S(s), .Co(Cout));

    initial

    begin   for( ca =0; ca<16; ca=ca+1)

            begin   for(cb=0; cb<16; cb=cb+1)

                begin

                A=array[ca];

                B=array[cb];

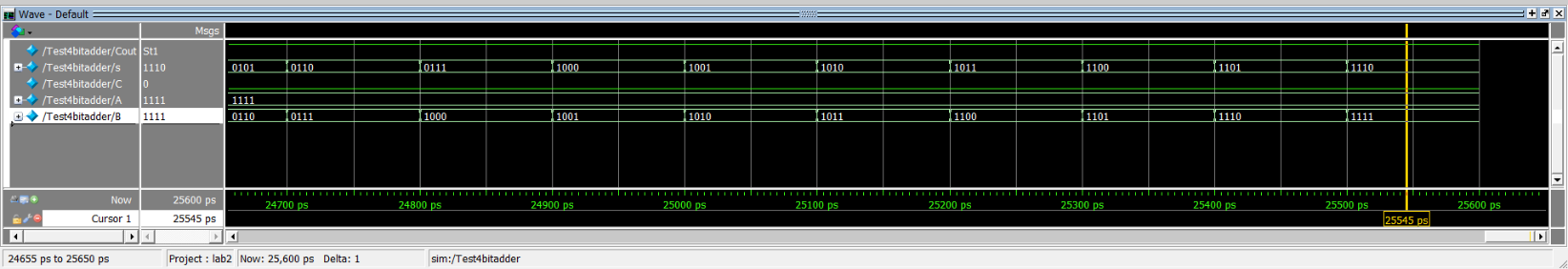
                #100;

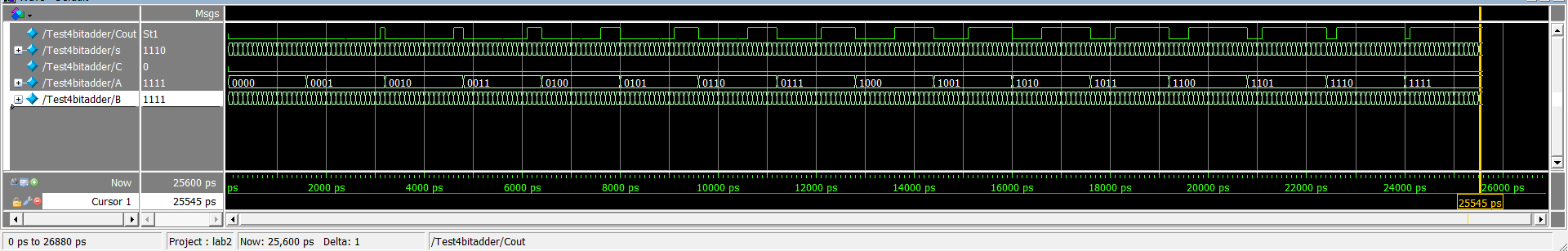
                end

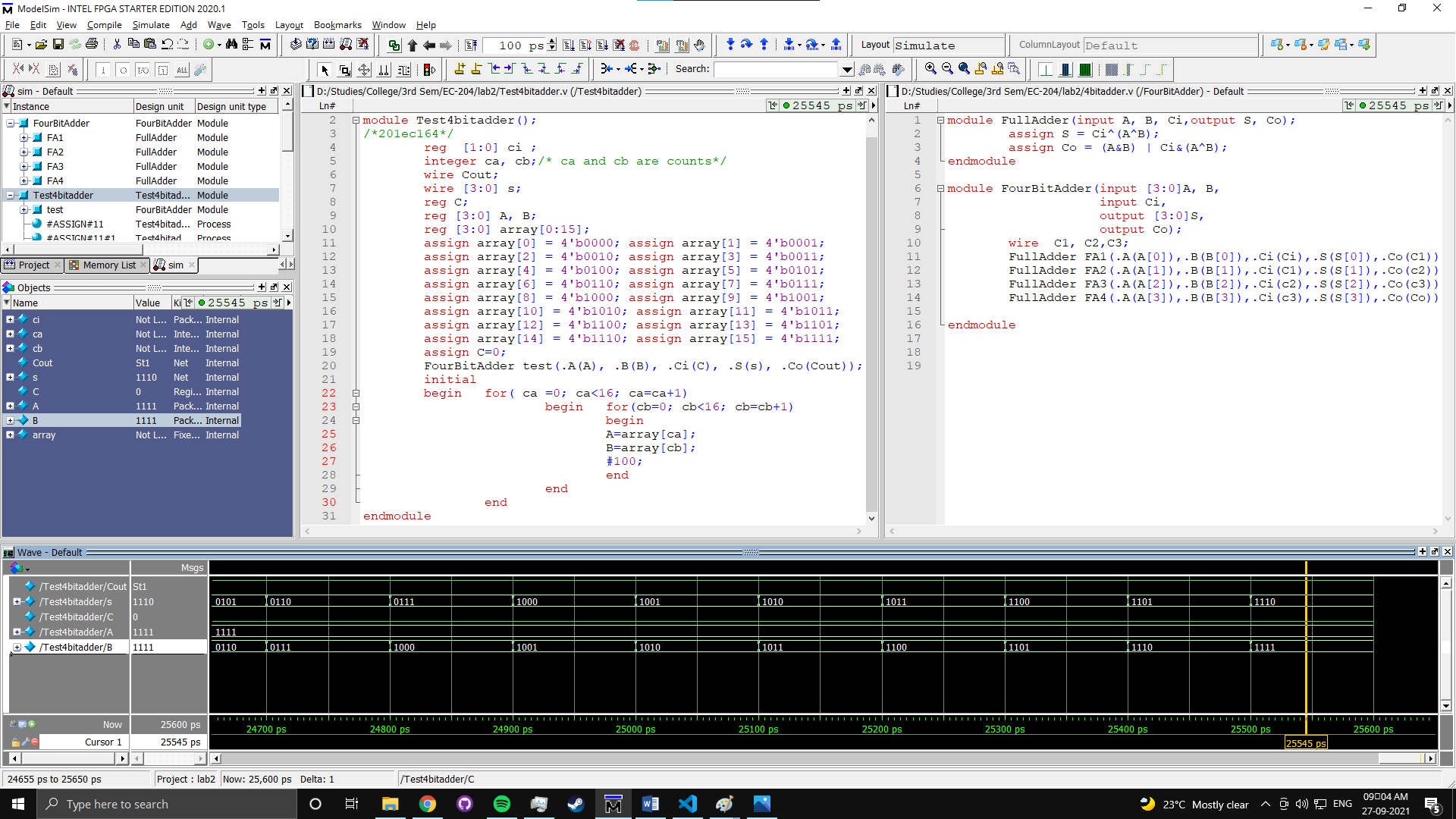
            end

        end

*endmodule*

****

****

****

**5]** **A four variable logic function that is equal to 1 if any three or all four of its variables are equal to 1 is called a majority function. Write a Verilog code that implements this majority function. Use the Boolean equation derived in Lab1 in assign statement.**

*module* Majority4Var(input A, B, C, D, output Y);

assign Y =  ~((~(A|B))|(~(A|C))|(~(A|D))|(~(B|C))|(~(B|D))|(~(C|D)));

*endmodule*

*/\*201EC164\*/*

*module* Majority4VarTest();

    reg A, B, C, D;

    reg [1:0] inp;

    integer a,b,c,d;

    assign inp[0] =0;

    assign inp[1] =1;

    wire Y;

Major4B test(.A(A), .B(B), .C(C), .D(D), .Y(Y));

initial

begin

    for(a=0;a<2;a=a+1)

    begin   for(b=0;b<2;b=b+1)

        begin   for(c=0;c<2;c=c+1)

            begin   for(d=0;d<2;d=d+1)

                begin   A=inp[a]; B=inp[b];

                    C=inp[c]; D=inp[d];

                    #100;

                end

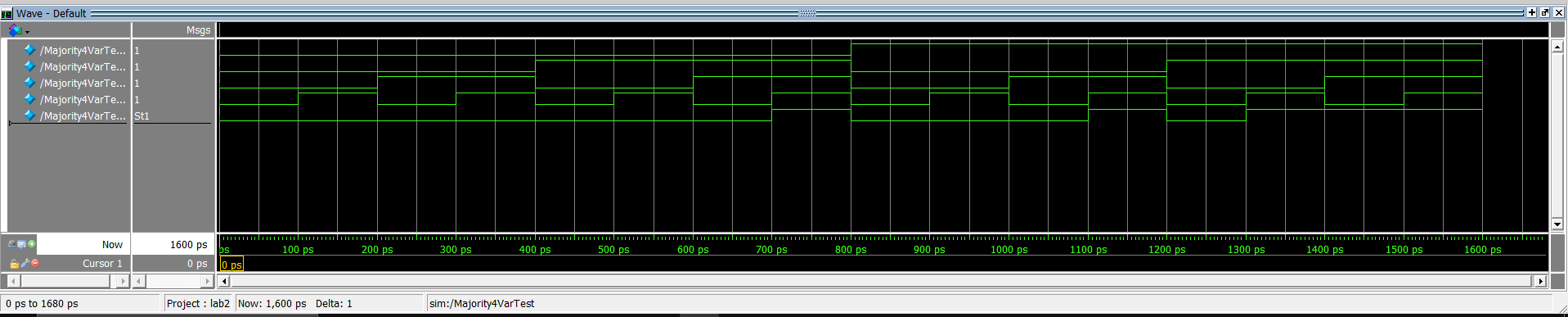
            end

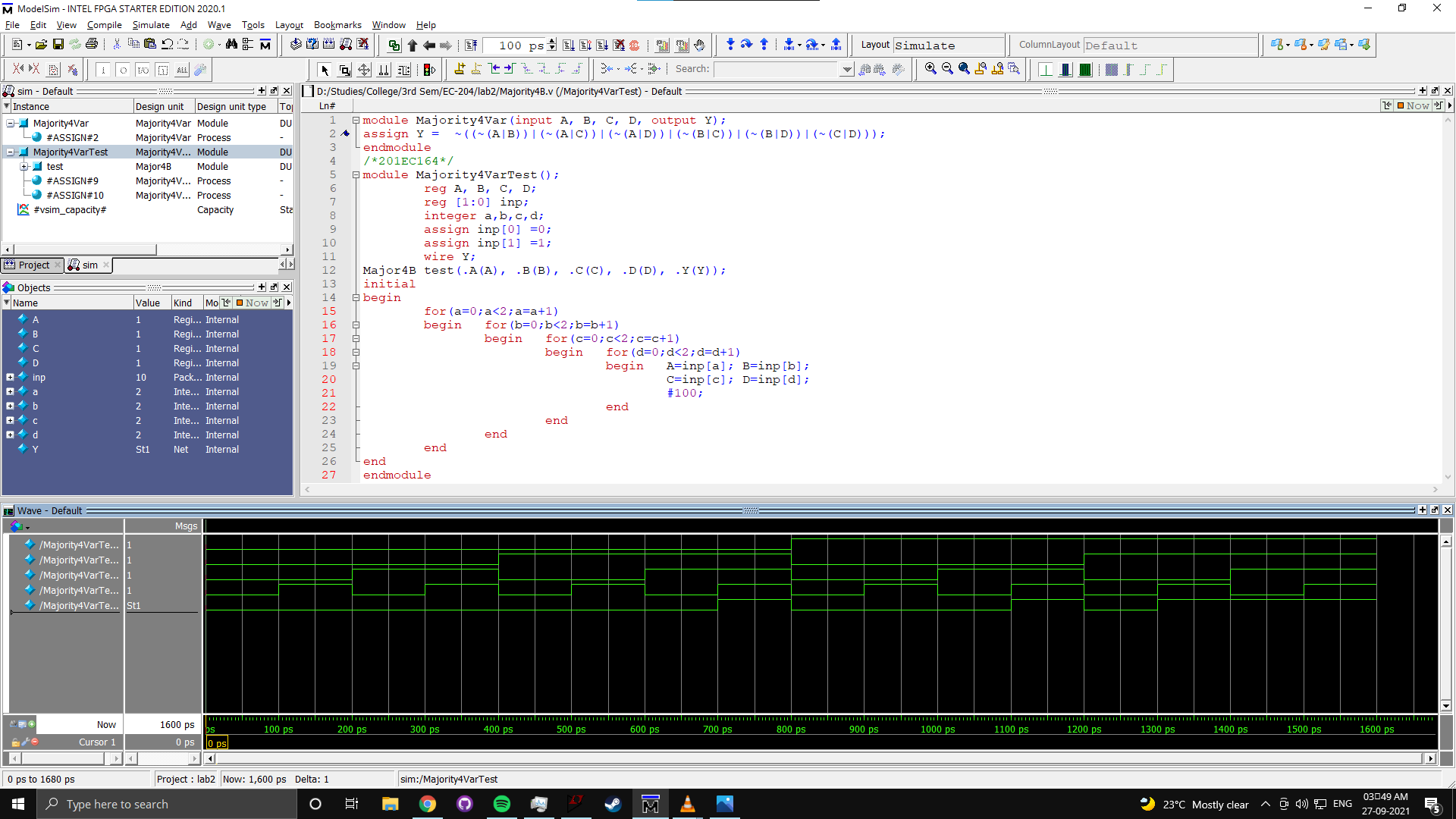
        end

    end

end

*endmodule*

****

****