**Câu 1:**

module Cau\_1(SW,LEDG);

input [2:0]SW;

output [1:0]LEDG;

fulladder1bit(SW[2], SW[1], SW[0], LEDG[0], LEDG[1]);

endmodule

module fulladder1bit (cin, a, b, s, cout);

parameter n = 1;

input [n-1:0] a, b;

input cin;

output reg [n-1:0]s;

output reg cout;

reg [n:0]c;

integer k;

always @(a, b, cin)

begin

c[0] = cin;

for (k = 0; k < n; k = k + 1)

begin

s[k] = a[k] ^ b[k] ^ c[k];

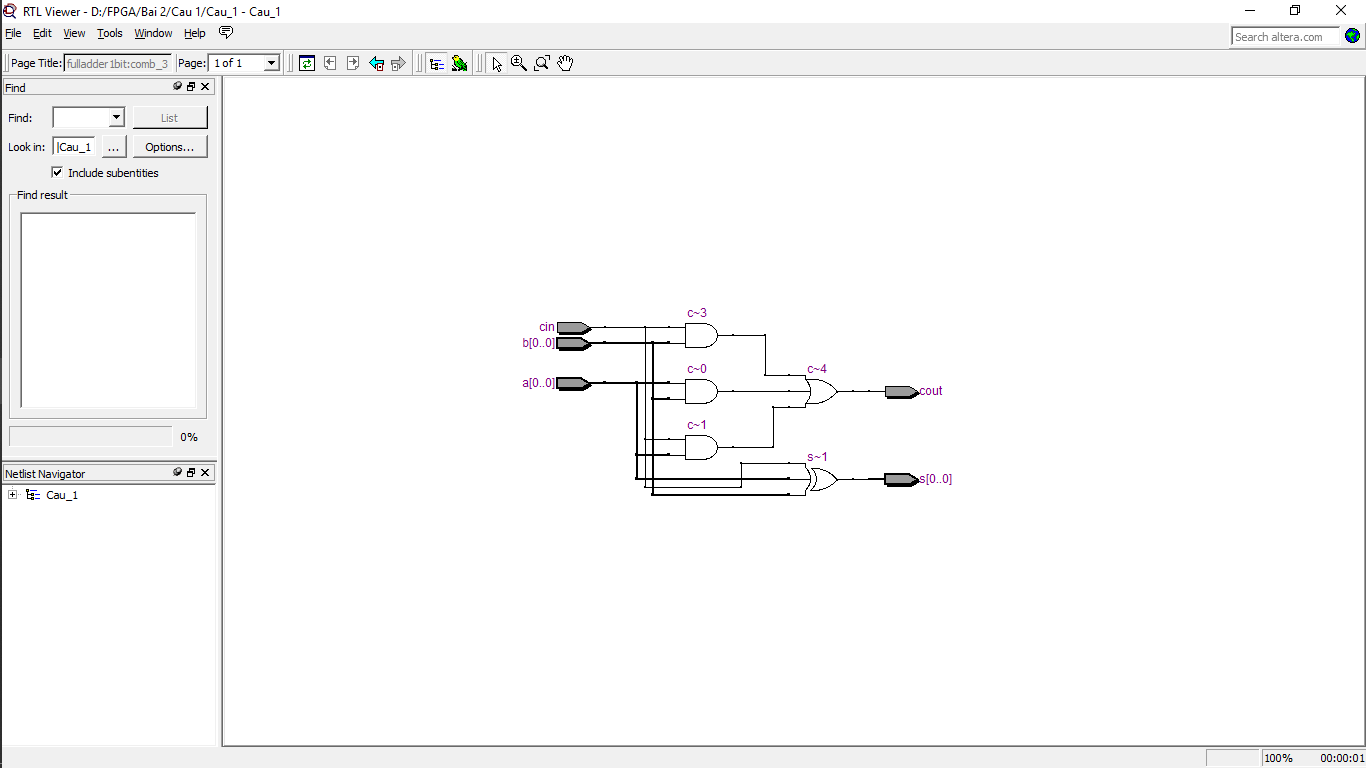
c[k + 1] = (a[k] & b[k])|(a[k] & c[k])|(c[k] & b[k]);

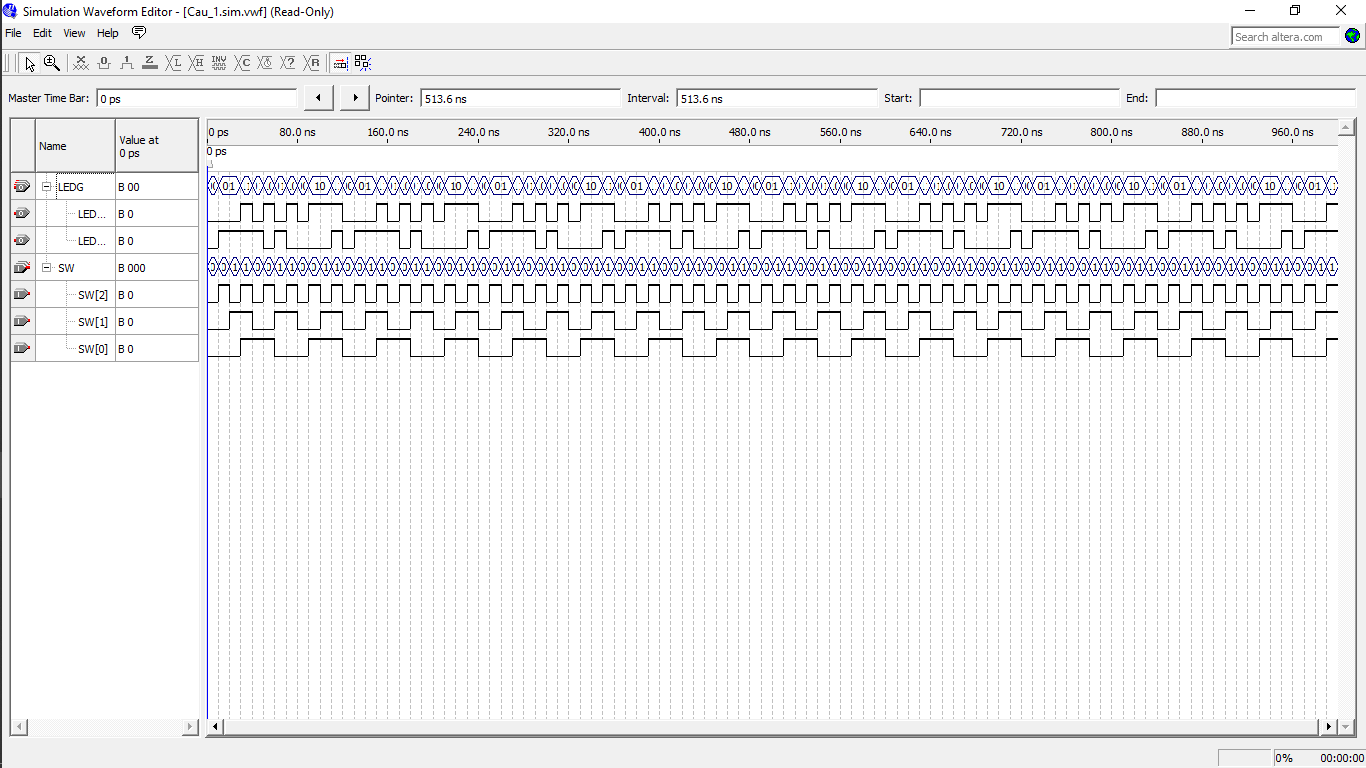
end

cout = c[n];

end

endmodule





**Câu 2:**

module Cau\_2(SW,LEDR);

input [8:0]SW;

output [4:0]LEDR;

fulladder4bit (SW[8], SW[3:0], SW[7:4], LEDR[3:0], LEDR[4]);

endmodule

module fulladd(cin, a, b, s, cout);

input cin, a, b;

output s, cout;

assign s = a ^ b ^ cin;

assign cout = (a & b)|(a & cin)|(b & cin);

endmodule

module fulladder4bit (cin, a, b, s, cout);

input cin;

input [3:0]a,b;

output [3:0]s;

output cout;

wire [3:1]c;

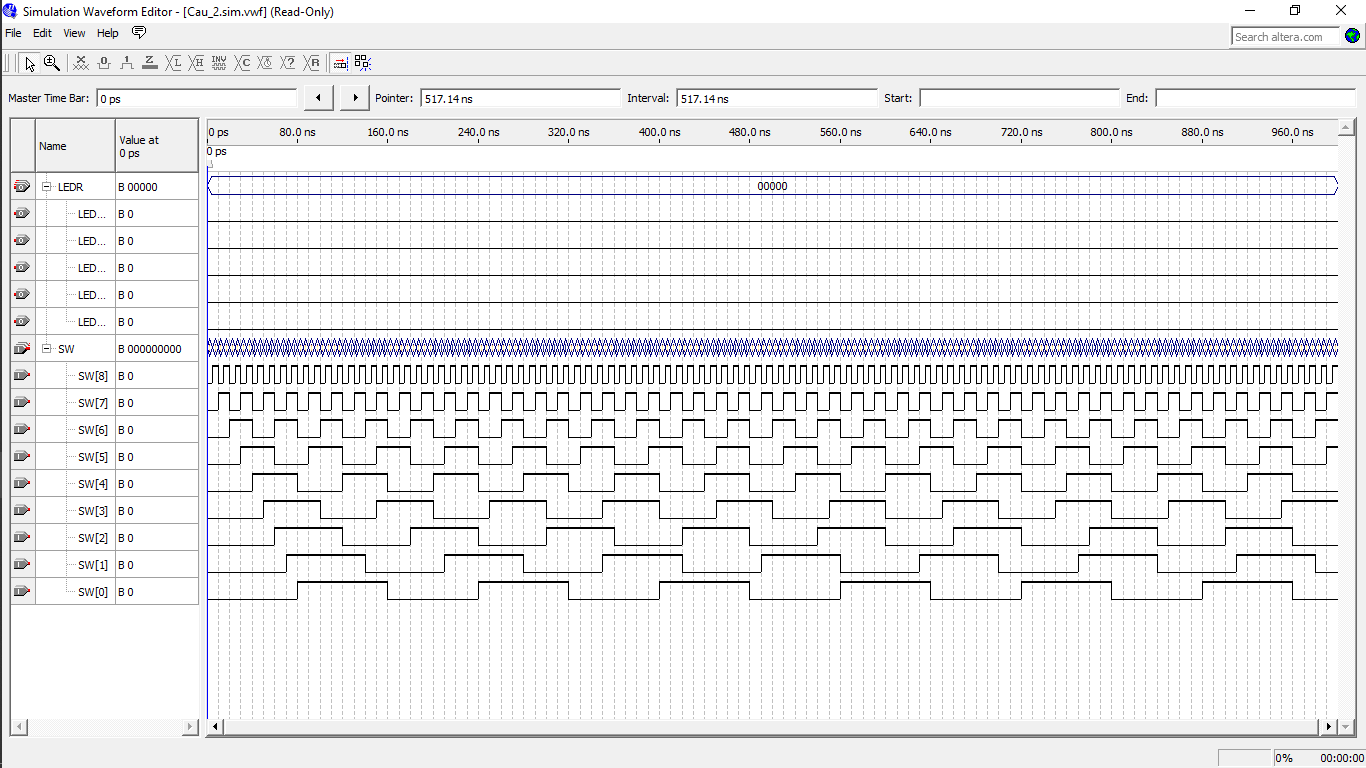
fulladd stage0 (cin, a0, b0, s0, c1);

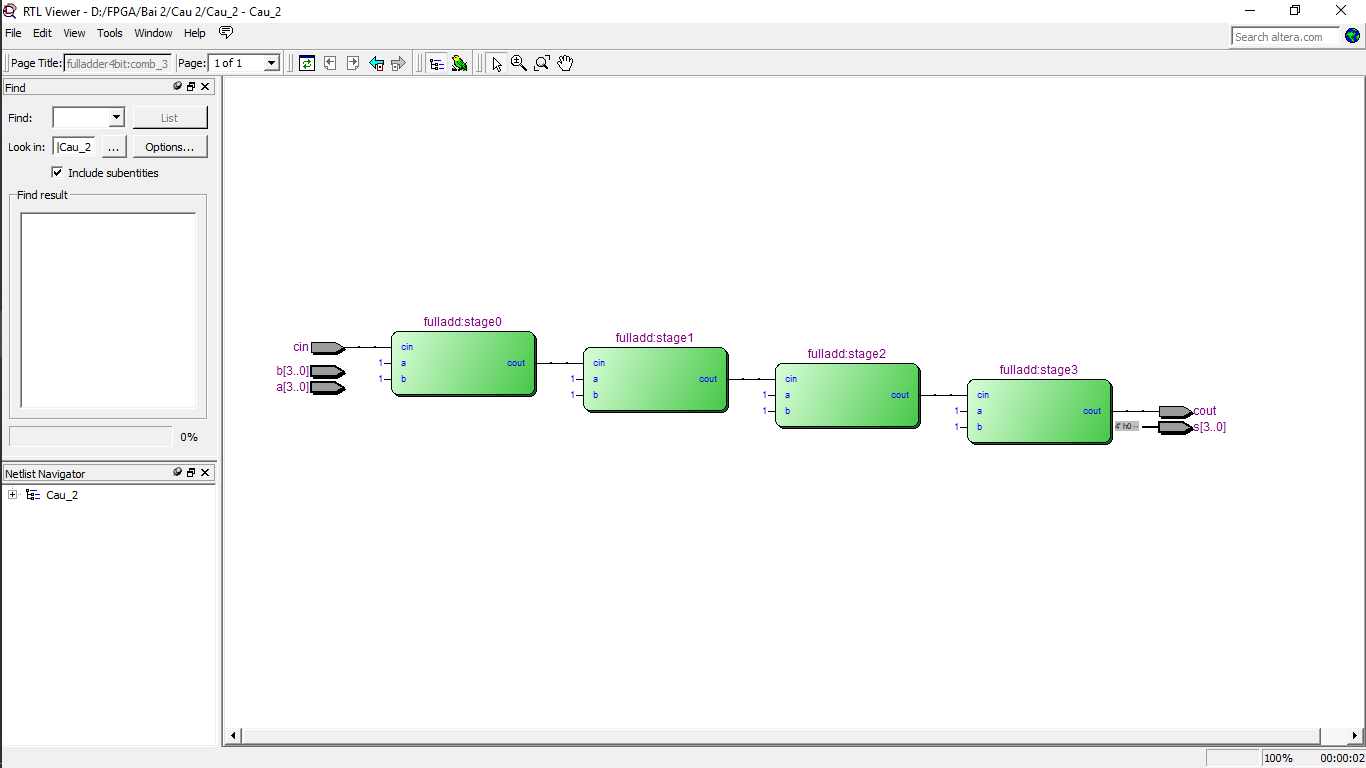
fulladd stage1 (c1, a1, b1, s1, c2);

fulladd stage2 (c2, a2, b2, s2, c3);

fulladd stage3 (c3, a3, b3, s3, cout);

endmodule





**Câu 3:**

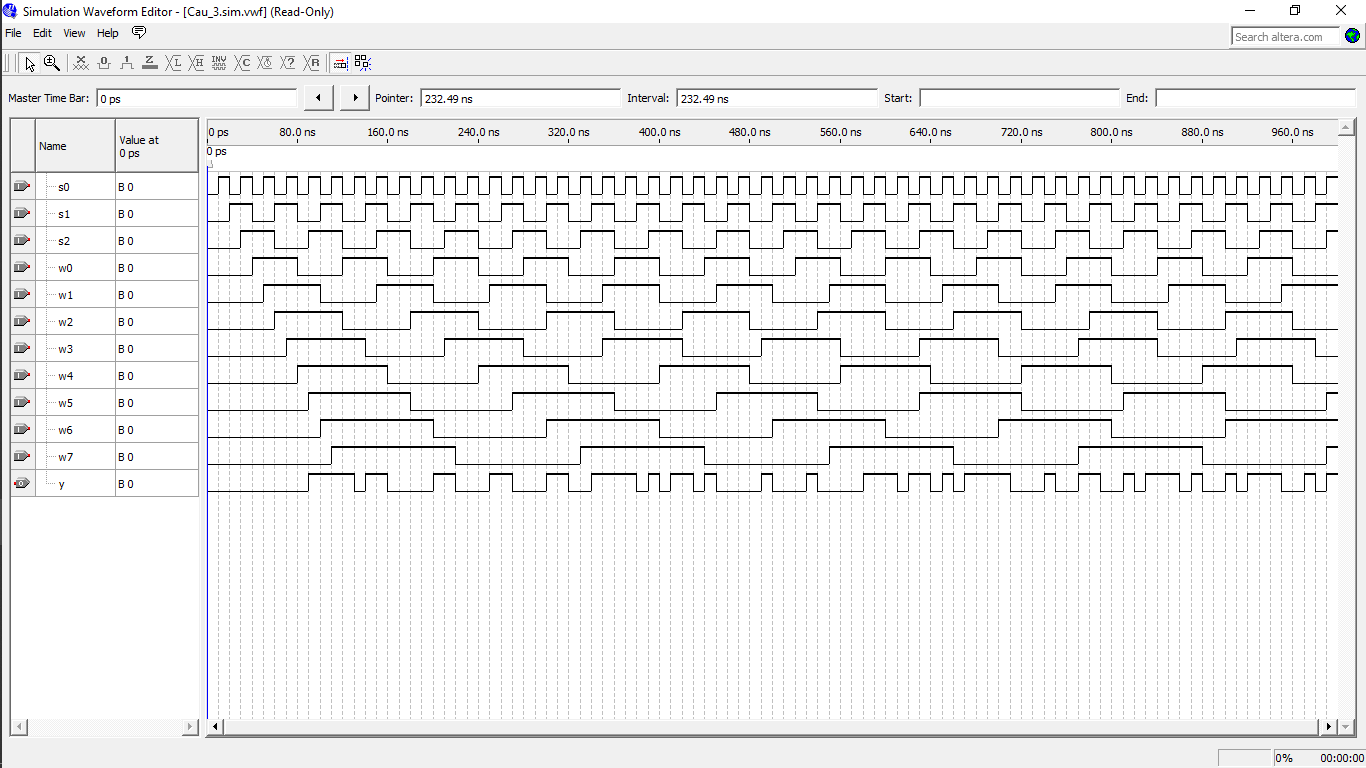
module Cau\_3(w0, w1, w2, w3, w4, w5, w6, w7, s0, s1, s2, y);

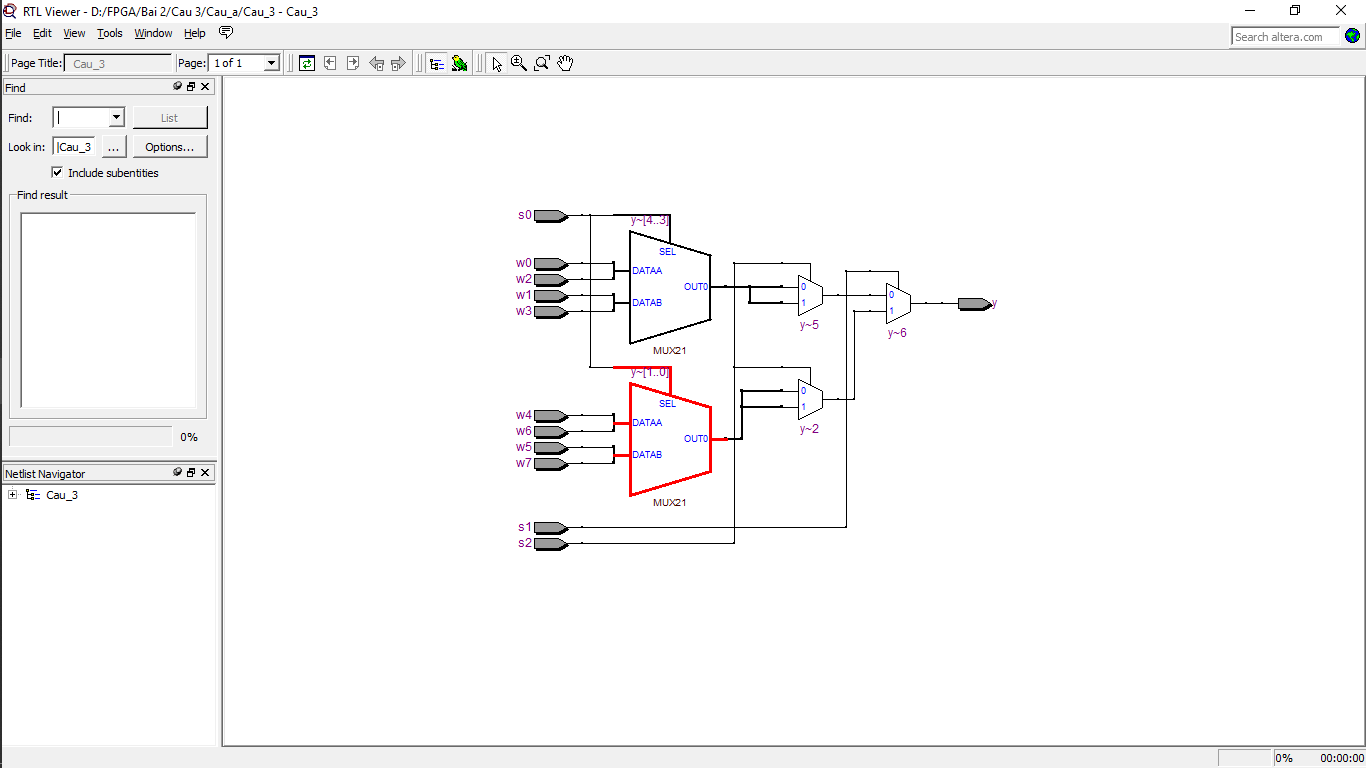
input w0, w1, w2, w3, w4, w5, w6, w7, s0, s1, s2;

output y;

assign y = s1?(s2?(s0?w7:w6):(s0?w5:w4)):(s2?(s0?w3:w2):(s0?w1:w0));

endmodule





module Cau\_b (W, S, y);

input [7:0]W;

input [2:0]S;

output reg y;

always @(W, S)

case (S)

'b000: y = W[0];

'b010: y = W[2];

'b100: y = W[4];

'b110: y = W[6];

'b001: y = W[1];

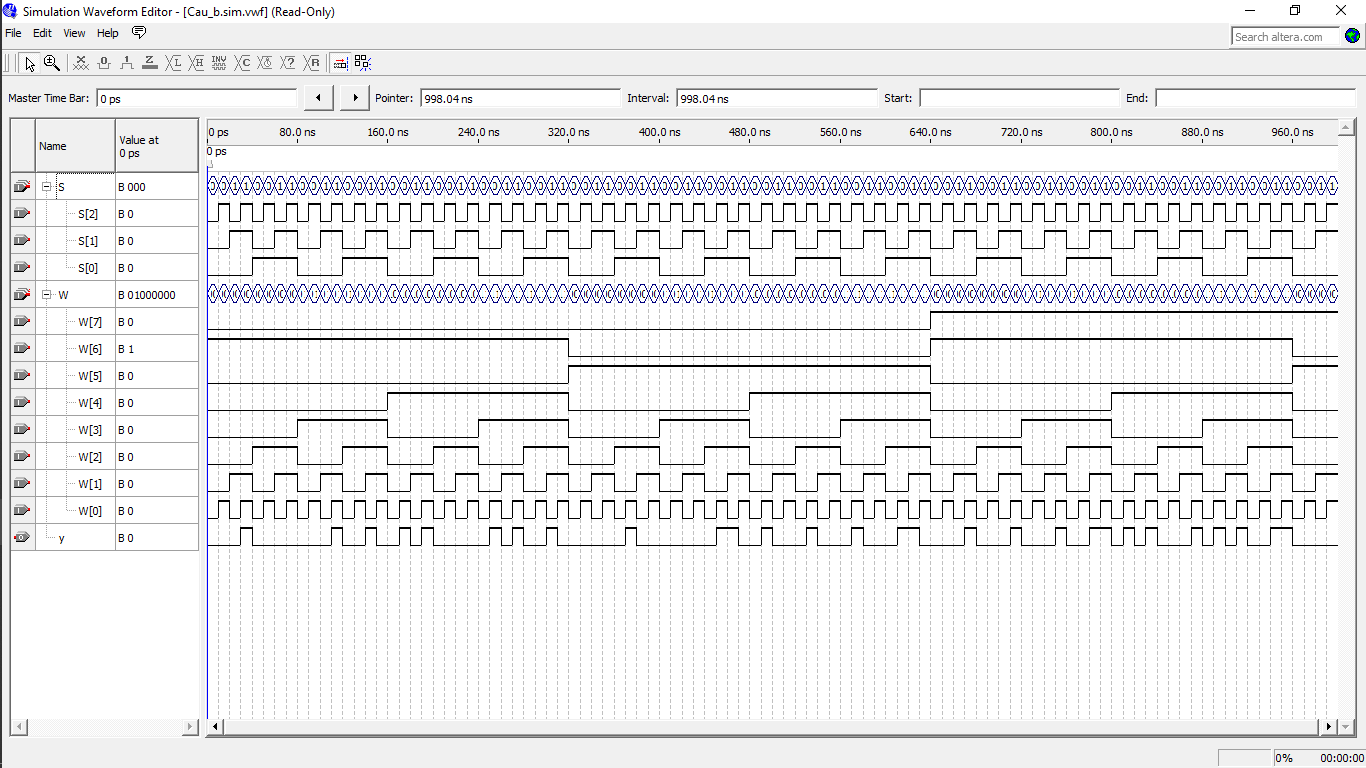
'b011: y = W[3];

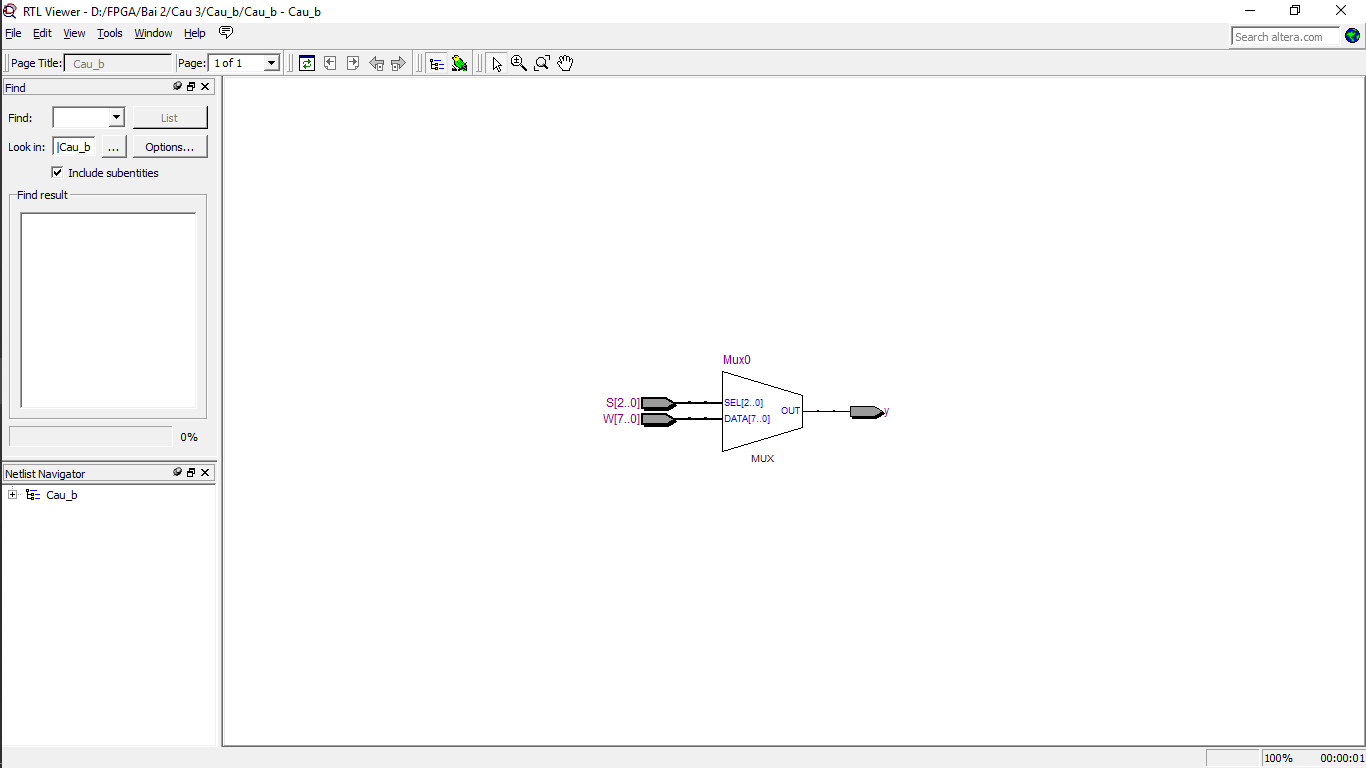
'b101: y = W[5];

'b111: y = W[7];

endcase

endmodule





**Câu 4:**

module Cau\_4(A, B, S, Y0, Y1, Y2, Y3);

input A, B, S;

output Y0, Y1, Y2, Y3;

assign Y0 = (~A & ~B & S);

assign Y1 = (A & ~B & S);

assign Y2 = (~A & B & S);

assign Y3 = (A & B & S);

endmodule

