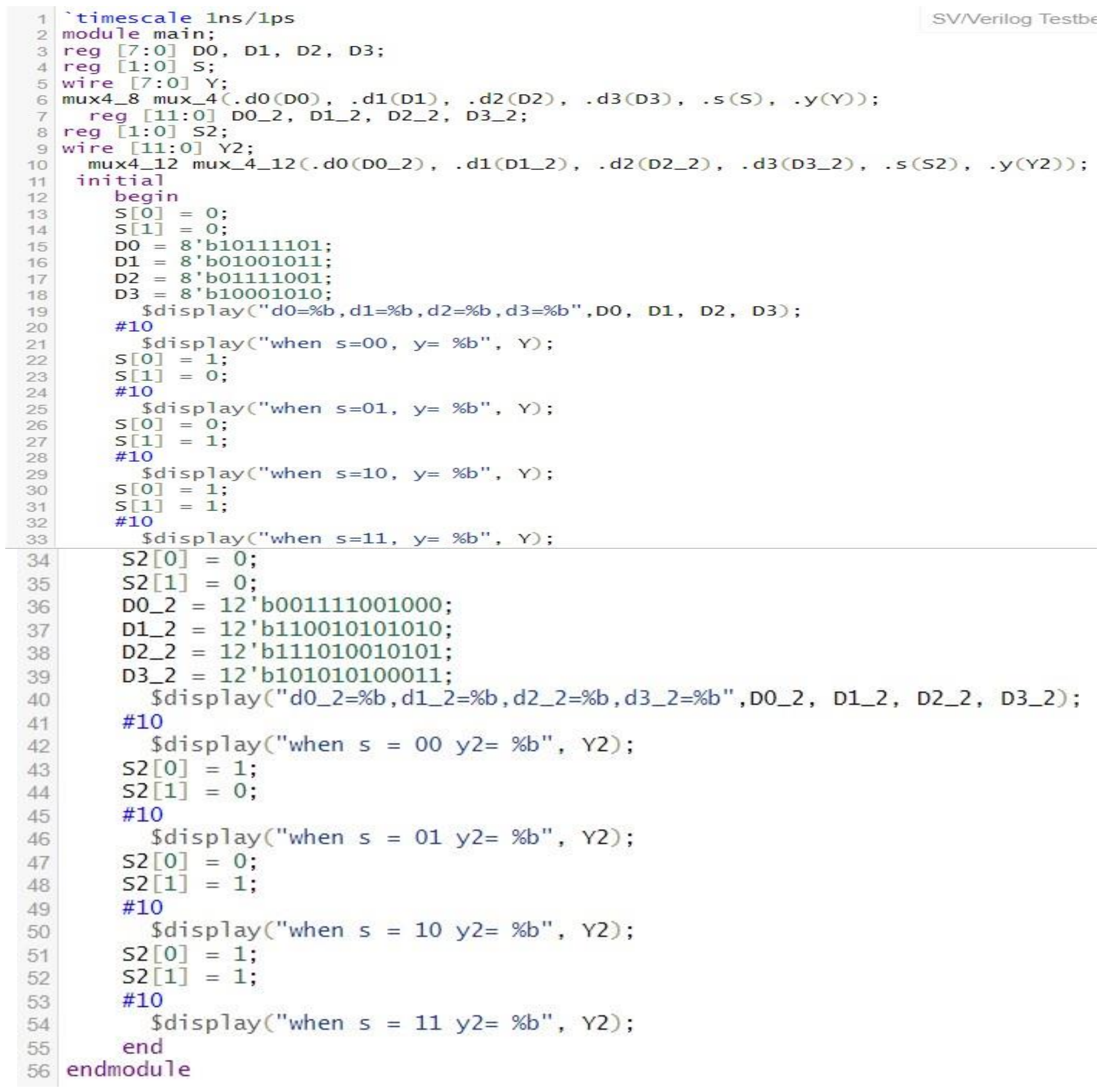
Verilog Assignment #2

Nehal Khaled Ibrahim 7118

Nehal khaled Ibrahim *7118*

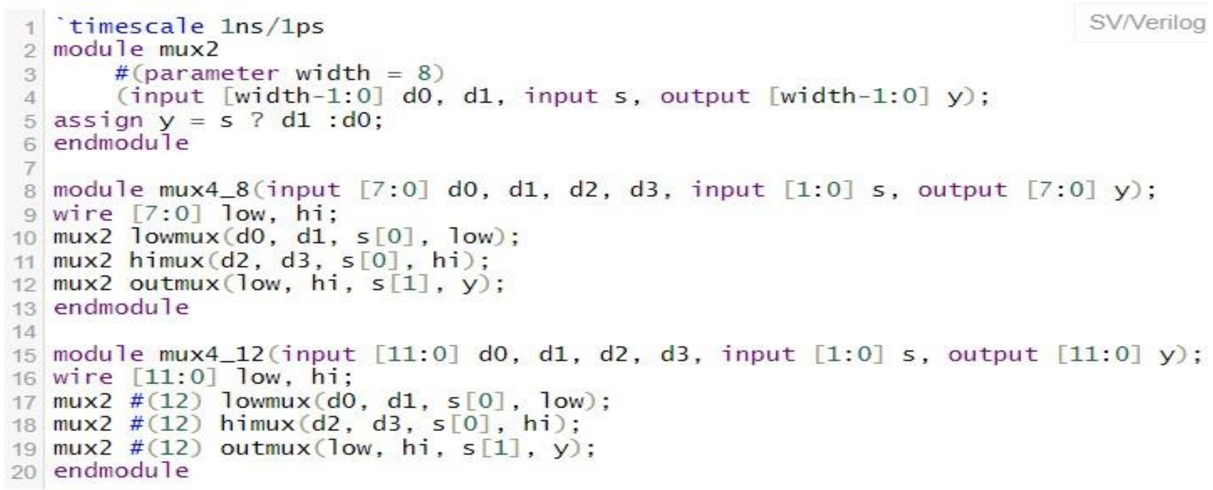
1.N-bit Multiplexers:

TestBench:



```
`timescale 1ns/1ps
module main;
reg [7:0] D0, D1, D2, D3;
reg [1:0] S;
wire [7:0] Y;
mux4_8 mux_4(.d0(D0), .d1(D1), .d2(D2), .d3(D3), .s(S), .y(Y));
  reg [11:0] D0_2, D1_2, D2_2, D3_2;
reg [1:0] S2;
wire [11:0] Y2;
  mux4_12 mux_4_12(.d0(D0_2), .d1(D1_2), .d2(D2_2), .d3(D3_2), .s(S2), .y(Y2));
 initial
    begin
    S[0] = 0;
    S[1] = 0;
    D0 = 8'b10111101;
    D1 = 8'b01001011;
    D2 = 8'b01111001;
    D3 = 8'b10001010;
      $display("d0=%b,d1=%b,d2=%b,d3=%b",D0, D1, D2, D3);
    #10
      $display("when s=00, y= %b", Y);
    S[0] = 1;
    S[1] = 0;
    #10
      $display("when s=01, y= %b", Y);
    S[0] = 0;
    S[1] = 1;
    #10
      $display("when s=10, y= %b", Y);
    S[0] = 1;
    S[1] = 1;
    #10
      $display("when s=11, y= %b", Y);
    S2[0] = 0;
    S2[1] = 0;
    D0_2 = 12'b001111001000;
    D1_2 = 12'b110010101010;
    D2_2 = 12'b111010010101;
    D3_2 = 12'b101010100011;
      $display("d0_2=%b,d1_2=%b,d2_2=%b,d3_2=%b",D0_2, D1_2, D2_2, D3_2);
    #10
      $display("when s = 00 y2= %b", Y2);
    S2[0] = 1;
    S2[1] = 0;
    #10
      $display("when s = 01 y2= %b", Y2);
    S2[0] = 0;
    S2[1] = 1;
    #10
      $display("when s = 10 y2= %b", Y2);
    S2[0] = 1;
    S2[1] = 1;
    #10
      $display("when s = 11 y2= %b", Y2);
    end
endmodule
```

Design:

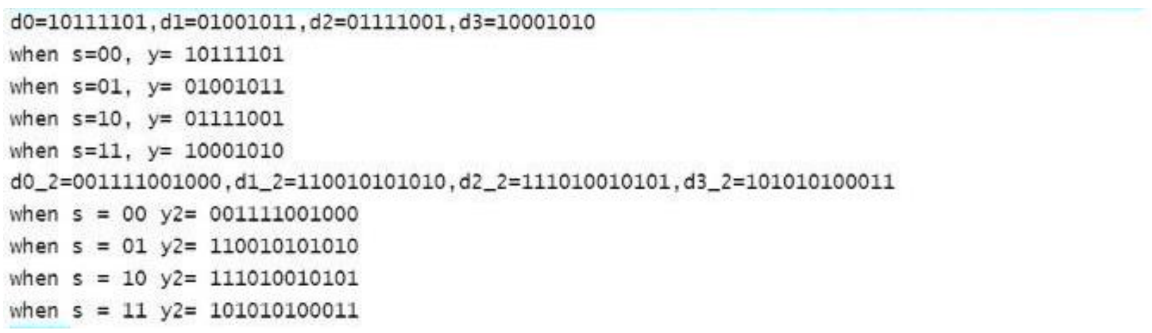


```
`timescale 1ns/1ps
module mux2
    #(parameter width = 8)
    (input [width-1:0] d0, d1, input s, output [width-1:0] y);
assign y = s ? d1 :d0;
endmodule

module mux4_8(input [7:0] d0, d1, d2, d3, input [1:0] s, output [7:0] y);
wire [7:0] low, hi;
mux2 lowmux(d0, d1, s[0], low);
mux2 himux(d2, d3, s[0], hi);
mux2 outmux(low, hi, s[1], y);
endmodule

module mux4_12(input [11:0] d0, d1, d2, d3, input [1:0] s, output [11:0] y);
wire [11:0] low, hi;
mux2 #(12) lowmux(d0, d1, s[0], low);
mux2 #(12) himux(d2, d3, s[0], hi);
mux2 #(12) outmux(low, hi, s[1], y);
endmodule
```

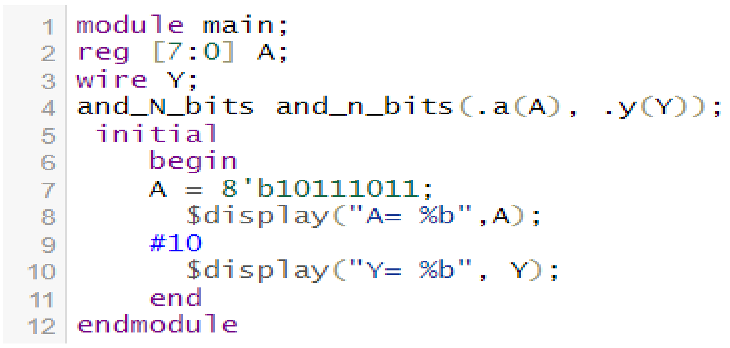
Result:



```
d0=10111101,d1=01001011,d2=01111001,d3=10001010
when s=00, y= 10111101
when s=01, y= 01001011
when s=10, y= 01111001
when s=11, y= 10001010
d0_2=001111001000,d1_2=110010101010,d2_2=111010010101,d3_2=101010100011
when s = 00 y2= 001111001000
when s = 01 y2= 110010101010
when s = 10 y2= 111010010101
when s = 11 y2= 101010100011
```

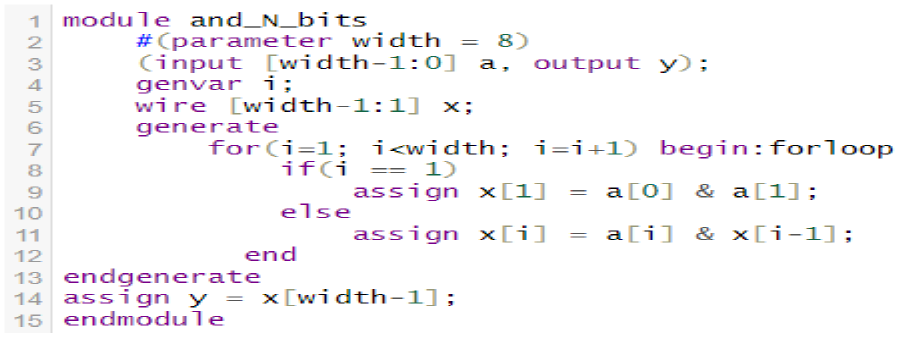
2.N-bit AND gate:

TestBench:



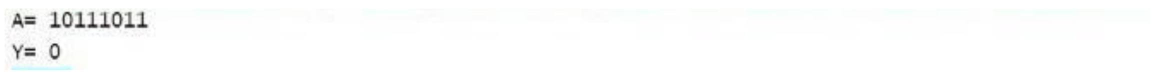
```
module main;
reg [7:0] A;
wire Y;
and_N_bits and_n_bits(.a(A), .y(Y));
 initial
     begin
     A = 8'b10111011;
       $display("A= %b",A);
     #10
       $display("Y= %b", Y);
     end
endmodule
```

Design:



```
module and_N_bits
     #(parameter width = 8)
     (input [width-1:0] a, output y);
     genvar i;
     wire [width-1:1] x;
     generate
          for(i=1; i<width; i=i+1) begin:forloop
               if(i == 1)
                    assign x[1] = a[0] & a[1];
               else
                    assign x[i] = a[i] & x[i-1];
             end
endgenerate
assign y = x[width-1];
endmodule
```

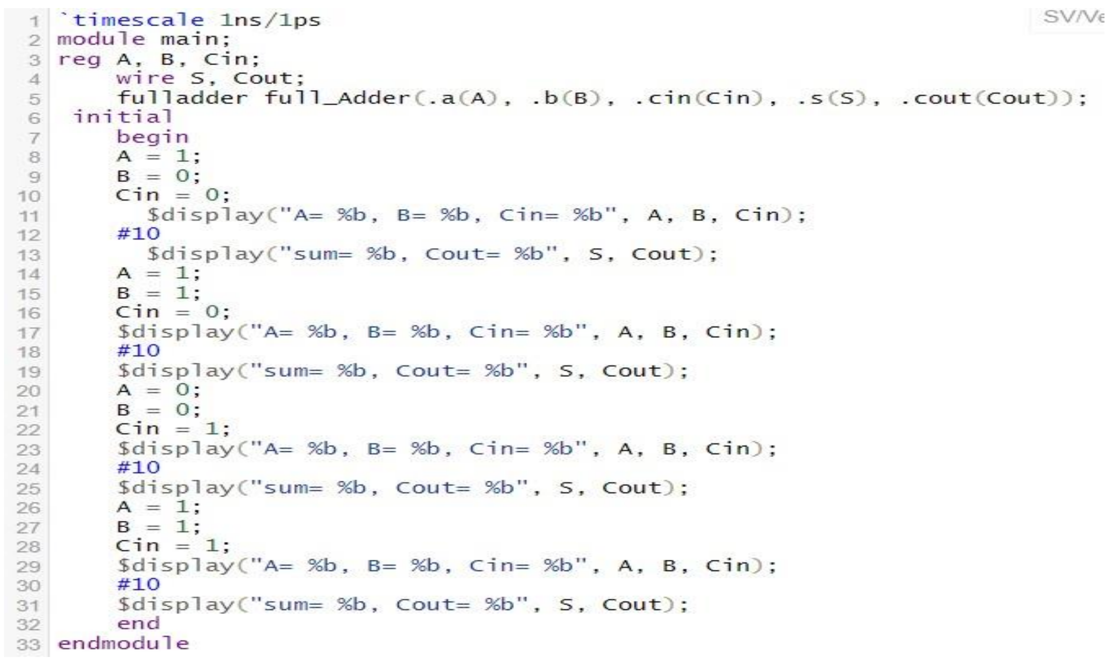
Result:



```
A= 10111011
Y= 0
```

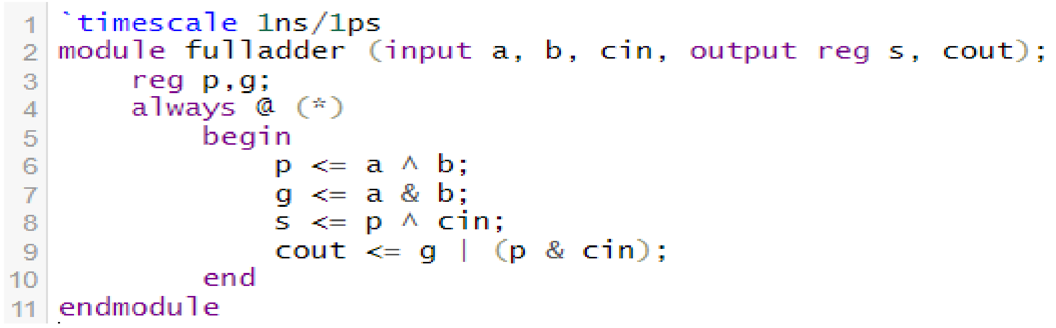
3.Full Adder by nonblocking assignments:

TestBench:



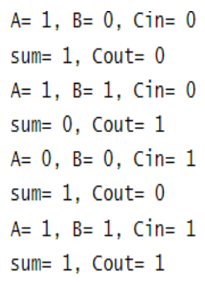
```
`timescale 1ns/1ps
module main;
reg A, B, Cin;
    wire S, Cout;
    fulladder full_Adder(.a(A), .b(B), .cin(Cin), .s(S), .cout(Cout));
 initial
    begin
    A = 1;
    B = 0;
    Cin = 0;
      $display("A= %b, B= %b, Cin= %b", A, B, Cin);
    #10
      $display("sum= %b, Cout= %b", S, Cout);
    A = 1;
    B = 1;
    Cin = 0;
    $display("A= %b, B= %b, Cin= %b", A, B, Cin);
    #10
    $display("sum= %b, Cout= %b", S, Cout);
    A = 0;
    B = 0;
    Cin = 1;
    $display("A= %b, B= %b, Cin= %b", A, B, Cin);
    #10
    $display("sum= %b, Cout= %b", S, Cout);
    A = 1;
    B = 1;
    Cin = 1;
    $display("A= %b, B= %b, Cin= %b", A, B, Cin);
    #10
    $display("sum= %b, Cout= %b", S, Cout);
    end
endmodule
```

Design:



```
`timescale 1ns/1ps
module fulladder (input a, b, cin, output reg s, cout);
     reg p,g;
     always @ (*)
          begin
               p <= a ^ b;
               g <= a & b;
               s <= p ^ cin;
               cout <= g | (p & cin);
          end
endmodule
```

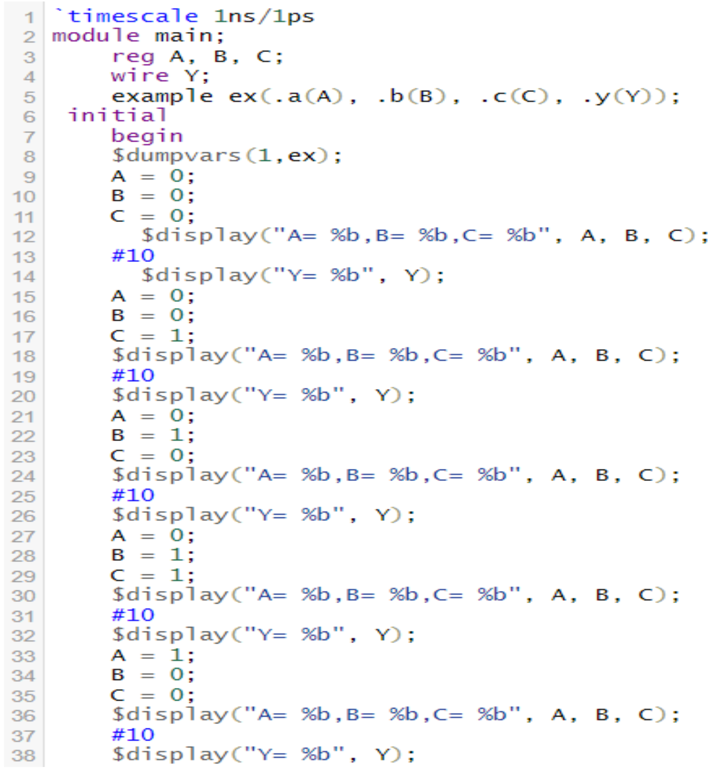
Result:

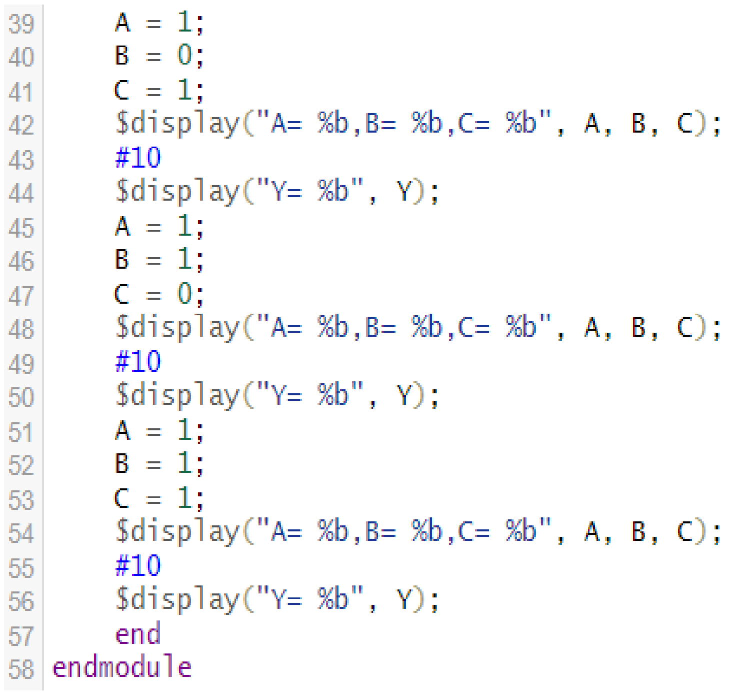


```
A= 1, B= 0, Cin= 0
sum= 1, Cout= 0
A= 1, B= 1, Cin= 0
sum= 0, Cout= 1
A= 0, B= 0, Cin= 1
sum= 1, Cout= 0
A= 1, B= 1, Cin= 1
sum= 1, Cout= 1
```

4. Logic gates with delay:

TestBench:





```
 `timescale 1ns/1ps
module main;
    reg A, B, C;
    wire Y;
    example ex(.a(A), .b(B), .c(C), .y(Y));
 initial
    begin
    $dumpvars(1,ex);
    A = 0;
    B = 0;
    C = 0;
      $display("A= %b,B= %b,C= %b", A, B, C);
    #10
      $display("Y= %b", Y);
    A = 0;
    B = 0;
    C = 1;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    A = 0;
    B = 1;
    C = 0;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    A = 0;
    B = 1;
    C = 1;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    A = 1;
    B = 0;
    C = 0;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    A = 1;
    B = 0;
    C = 1;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    A = 1;
    B = 1;
    C = 0;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    A = 1;
    B = 1;
    C = 1;
    $display("A= %b,B= %b,C= %b", A, B, C);
    #10
    $display("Y= %b", Y);
    end
endmodule
```

Nehal khaled Ibrahim 7118

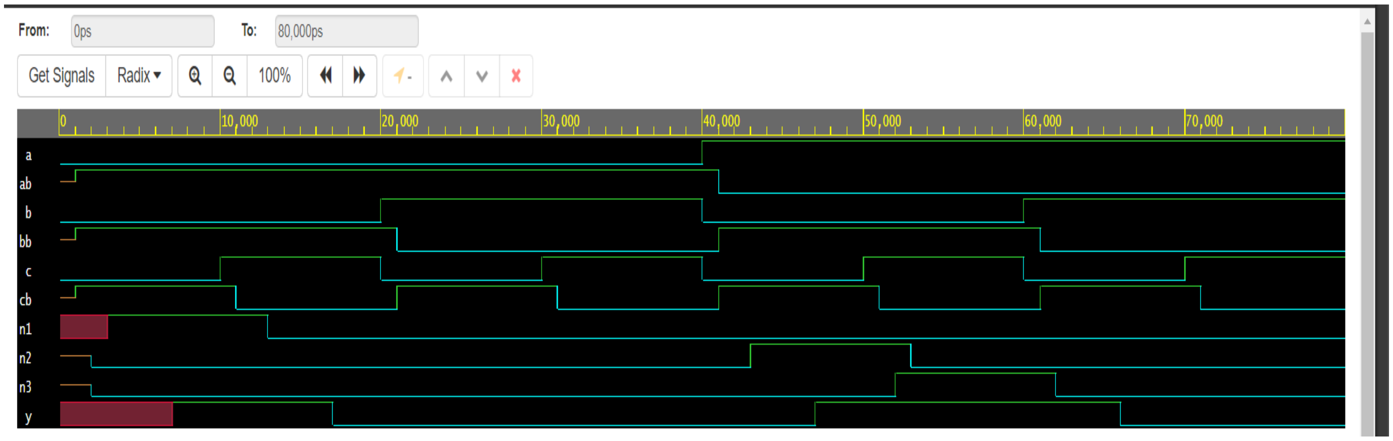
Design:

```
`timescale 1ns/1ps
module example (input a, b, c, output y);
    wire ab, bb, cb, n1, n2, n3;
    assign #1 {ab, bb, cb} = ~ {a, b, c};
    assign #2 n1 = ab & bb & cb;
    assign #2 n2 = a & bb & cb;
    assign #2 n3 = a & bb & c;
    assign #4 y = n1 | n2 | n3;
endmodule
```

Result:

```
A= 0,B= 0,C= 0
Y= 1
A= 0,B= 0,C= 1
Y= 0
A= 0,B= 1,C= 0
Y= 0
A= 0,B= 1,C= 1
Y= 0
A= 1,B= 0,C= 0
Y= 1
A= 1,B= 0,C= 1
Y= 1
A= 1,B= 1,C= 0
Y= 0
A= 1,B= 1,C= 1
Y= 0
```

Signal:


From: 0ps
To: 80,000ps
Get Signals
Radix
100%
0
10,000
20,000
30,000
40,000
50,000
60,000
70,000
a
ab
b
bb
c
cb
n1
n2
n3
y