Digital Electronics Verilog Lab 3

Submitted By:
Saurabh Kumar
SC ID: SC22B146
Course: DIGITAL ELECTRONICS AND VLSI DESIGN LAB (AV232)

LAB SHEET

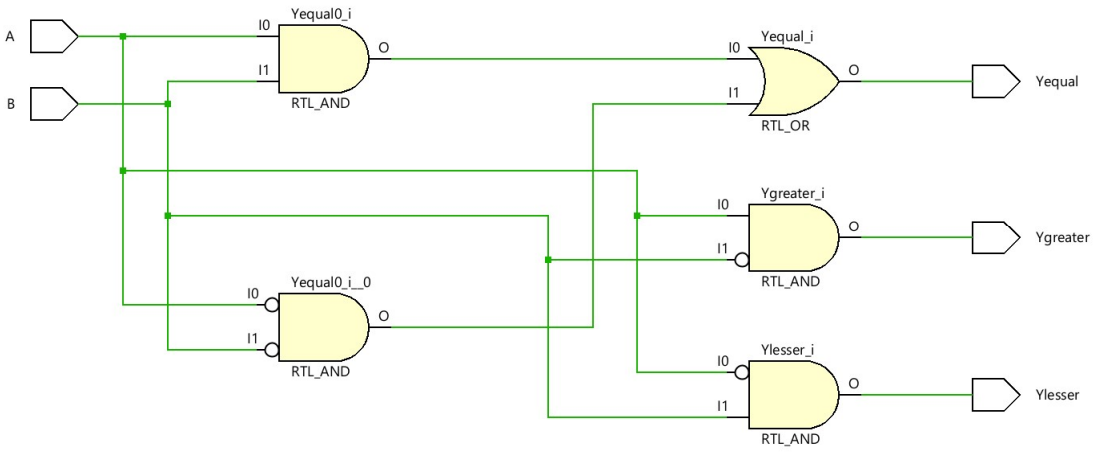
Question No. 1 Make an one bit comparator circuit through HDL
Sol: Boolean expression should be:

To check if (A=B) : Y = (A and B) or (not A and not B) = A xnor B
To check if (A>B) : Y = A and (not B)
To check if (A<B) : Y = (not A) and B

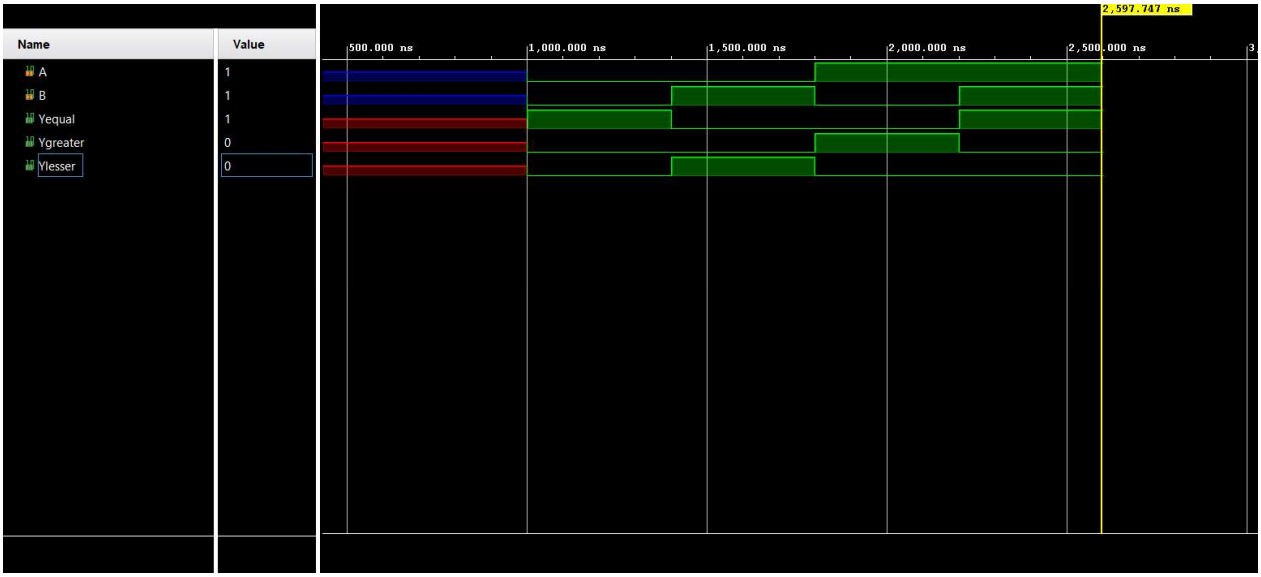
Code:

```
module comparator(
   input A,B,
   output Yequal, Ygreater, Ylesser);
assign Yequal = (A&B)|((~A)&(~B));  //A=B
assign Ygreater = A&(~B);   //A>B
assign Ylesser = (~A)&B;    //A<B
endmodule
```

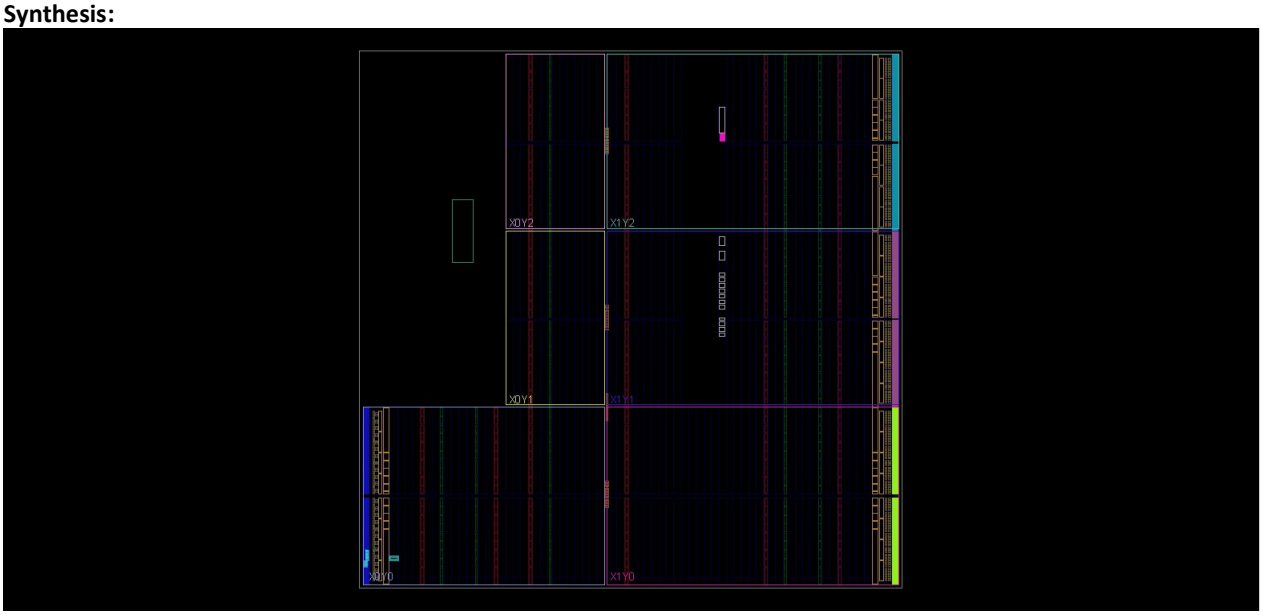
Schematic:



Simulation:



Synthesis:



Question No. 2 Make an 4 bit Ripple Carry Adder circuit by module Instantiation technique.
[Hint: Make one full adder circuit and then instantiate (call) it for four times.]

Sol: The **boolean** expression for Full Adder:

Sum = A xor B xor Cin

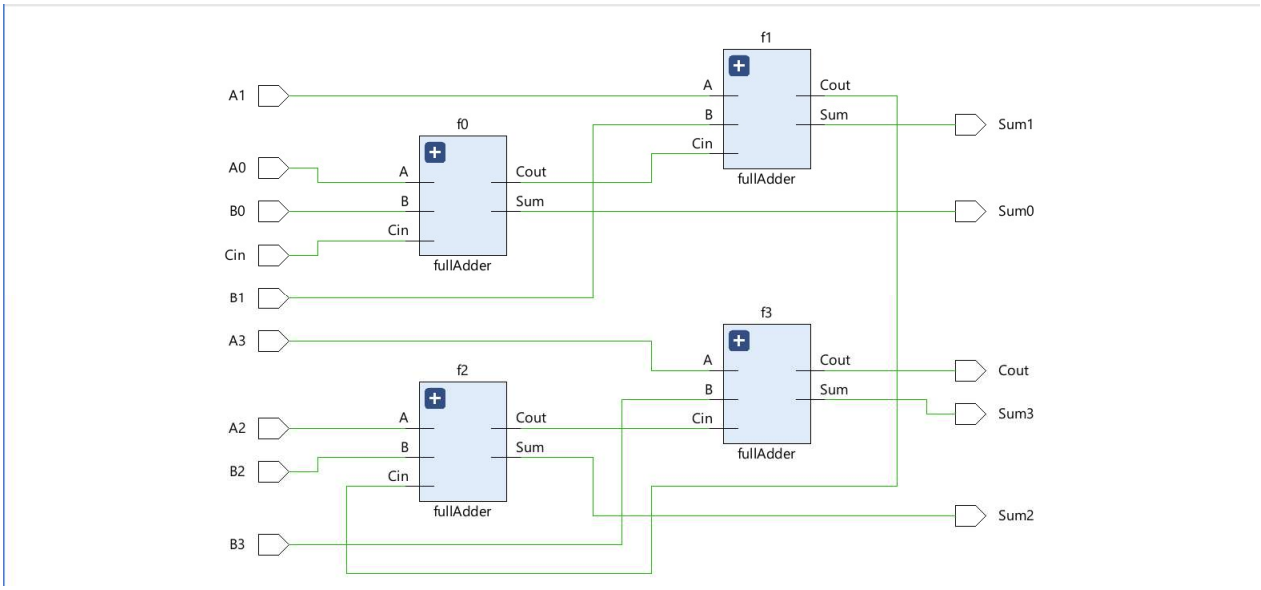
Cout = (A and B) or (B and Cin) or (Cin and A)

Code:

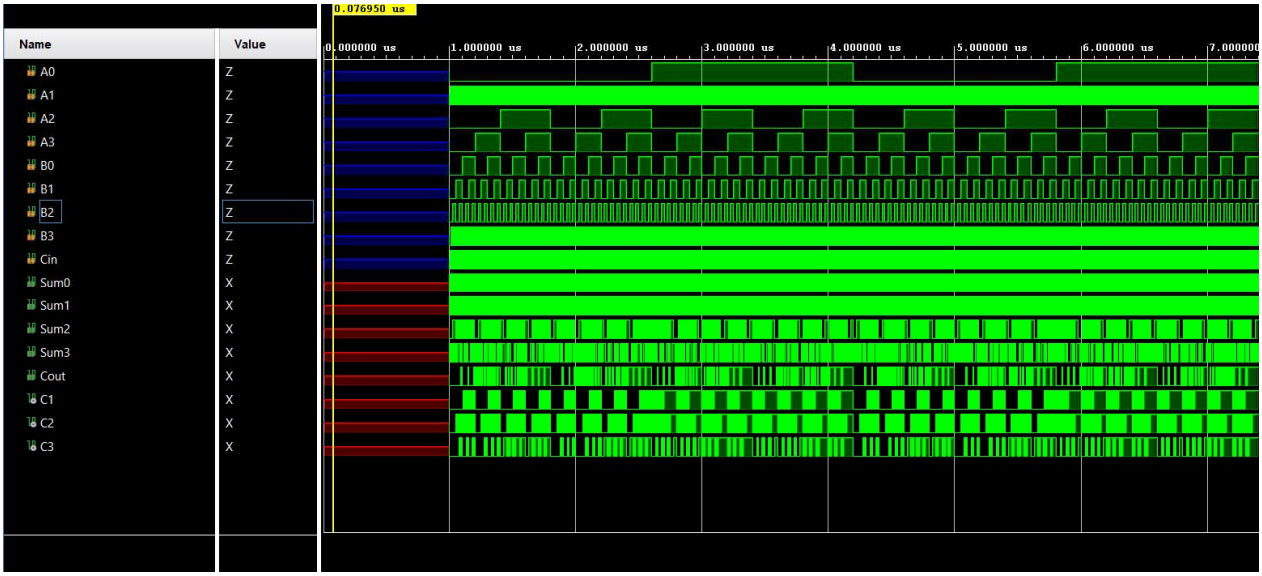
```
module fullAdder(Sum,Cout,A,B,Cin);
input A,B,Cin;
output Sum,Cout;
assign Sum = A^B^Cin;
assign Cout = (A&B) | (B&Cin) | (Cin&A);
endmodule

module rippleCarryAddder_4bit(
   input A0,A1,A2,A3,B0,B1,B2,B3,
   input Cin,
   output Sum0,Sum1,Sum2,Sum3,
   output Cout);
 wire C1,C2,C3;

 //Instantiate four 1-bit full adders
 fullAdder f0(Sum0,C1,A0,B0,Cin);
 fullAdder f1(Sum1,C2,A1,B1,C1);
```

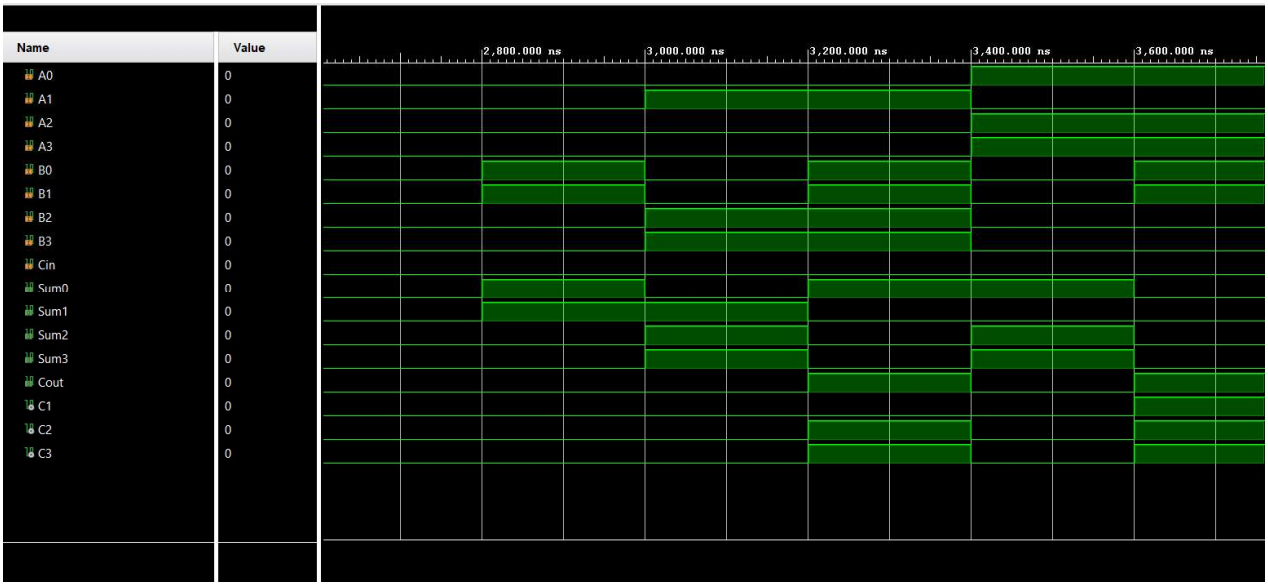
fullAdder f2(Sum2,C3,A2,B2,C2);
fullAdder f3(Sum3,Cout,A3,B3,C3);

Schematic:

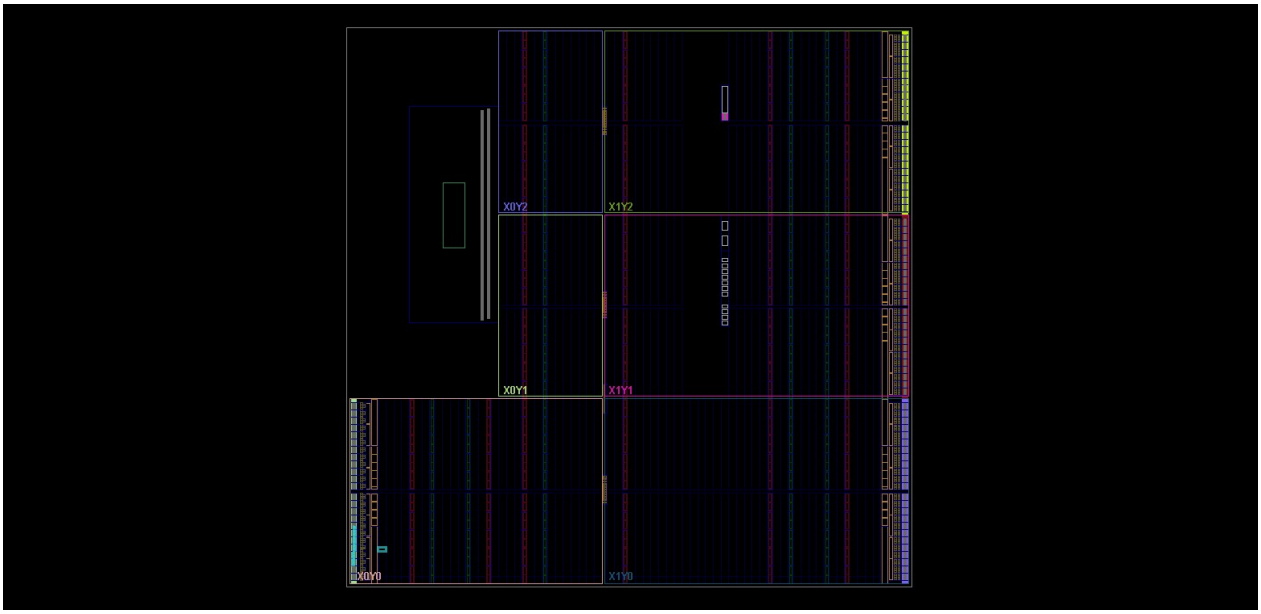


Simulation:





Synthesis:



Question No. 3 Make an SR Latch.

[Hint: Latch is level sensitive not an edge sensitive]

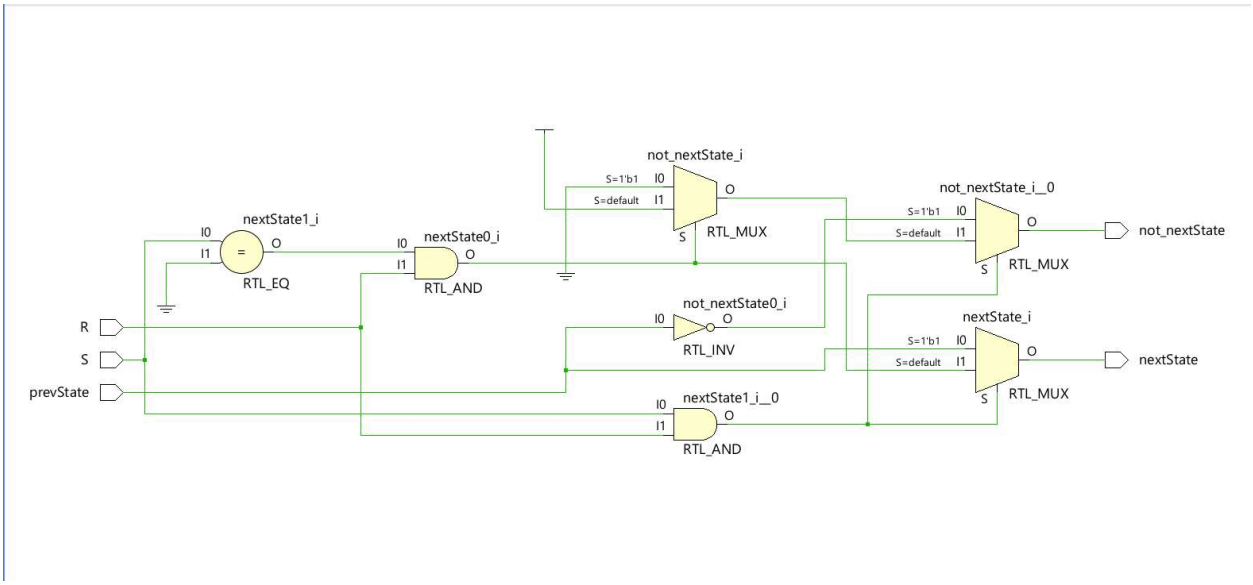
Sol: SR Latch truth table using NAND Gates:

S	R	Q(t+1)	Q'(t+1)
1	1	Q(t)	Q'(t)
0	1	1	0
1	0	0	1
0	0	Undetermined	Undetermined

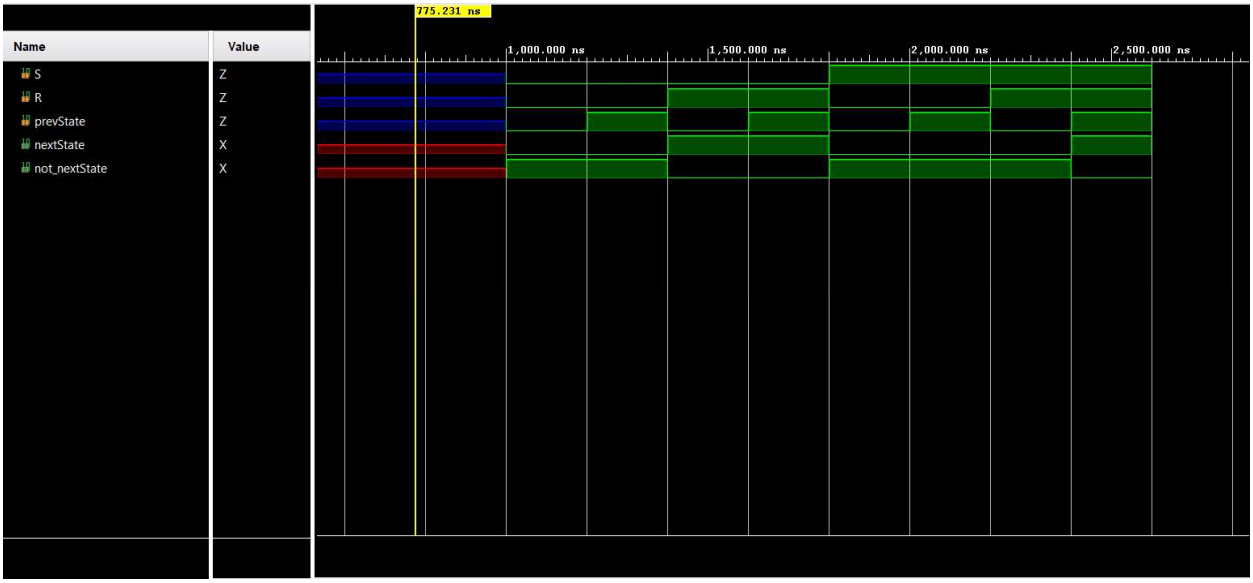
Code:

```
module srLatch(
   input S, R, prevState,
   output reg nextState, not_nextState);
always@(*)
begin
   if(S==1'b1 & R==1'b1)
      begin
      nextState = prevState;
      not_nextState = ~prevState;
      end
    else if(S==1'b0 & R==1'b1)
      begin
      nextState = 1'b1;
      not_nextState = 1'b0;
      end
    else
      begin
      nextState = 1'b0;
      not_nextState = 1'b1;
      end
    end
endmodule
```

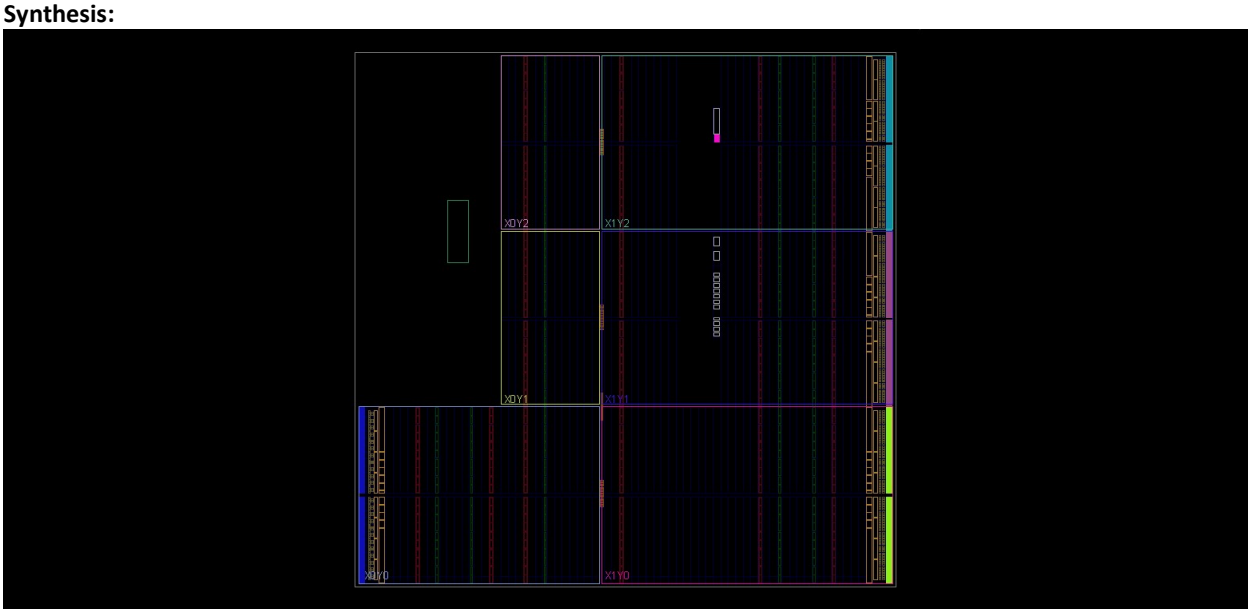
Schematic:



Simulation:



Synthesis:



ASSIGNMENT

Question No. 1 Make one 3:1 MUX by using 2:1 MUX.

Sol: Code:

```
module MUX2_1(F,I0,I1,S);
   input I0,I1,S;
   output F;
   assign F = ((~S)&I0) | (S&I1);
endmodule

module MUX3_1(A0,A1,A2,S0,S1,Y);
   input A0,A1,A2,S0,S1;
   output Y;
   wire Y1;

   MUX2_1 m1(Y1,A0,A1,S0);
   MUX2_1 m2(Y,Y1,A3,S1);
endmodule
```

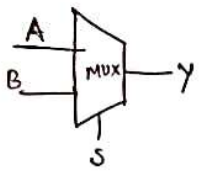
Schematic and Boolean Expression:

Verilog Lab Assignment 3

① Make one 3:1 MUX by using 2:1 MUX

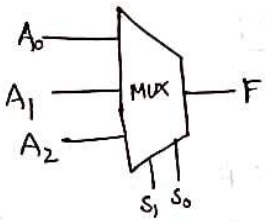
Soln:

2:1 MUX:



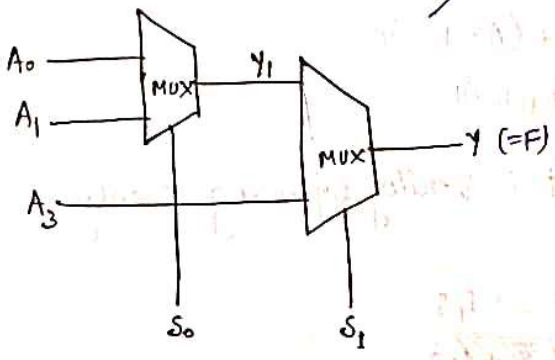
$Y = \overline{S}A + SB$

3:1 MUX:



$F = A_0 \overline{S_1}\,\overline{S_0} + A_1 \overline{S_1} S_0 + A_2 S_1 \overline{S_0} + A_2 S_1 S_0$

3:1 MUX using 2:1 MUX's:



S_1	S_0	F
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_2

Question No. 2 Make 4 bit Carry Look Ahead (CLA) Adder circuit using HDL

Sol: Code:

```
module CLA(
  input A0,B0,A1,B1,A2,B2,A3,B3,Cin,
  output S0,S1,S2,S3,Cout
  );
  wire P0,P1,P2,P3;
  wire G0,G1,G2,G3;
  wire C1,C2,C3;

  assign P0 = A0^B0;
  assign P1 = A1^B1;
  assign P2 = A2^B2;
  assign P3 = A3^B3;

  assign G0 = A0&B0;
  assign G1 = A1&B1;
  assign G2 = A2&B2;
  assign G3 = A3&B3;

  assign C1 = G0|(P0&Cin);
  assign C2 = G1|(P1& (G0|(P0&Cin)) );
  assign C3 = G2|(P2& (G1|(P1&(G0|(P0&Cin)))) );
  assign Cout = G3|(P3& (G2|(P2& (G2|(P2& (G1|(P1&(G0|(P0&Cin)))))) )));

  assign S0 = P0^Cin;
  assign S1 = P1^C1;
  assign S2 = P2^C2;
  assign S3 = P3^C3;
endmodule
```

Boolean Expression and Diagram:

Carry look ahead adder:

$S_i = P_i \oplus C_i$ | $P_i = A_i \oplus B_i$

$C_{i+1} = G_i + P_i C_i$ | $G_i = A_i B_i$

$C_0 = C_0$

$C_1 = G_{10} + P_0 C_0$

$C_2 = G_{11} + P_1 C_1$

$= G_{11} + P_1 G_{10} + P_1 P_0 C_0$

$C_3 = G_{12} + P_2 C_2$

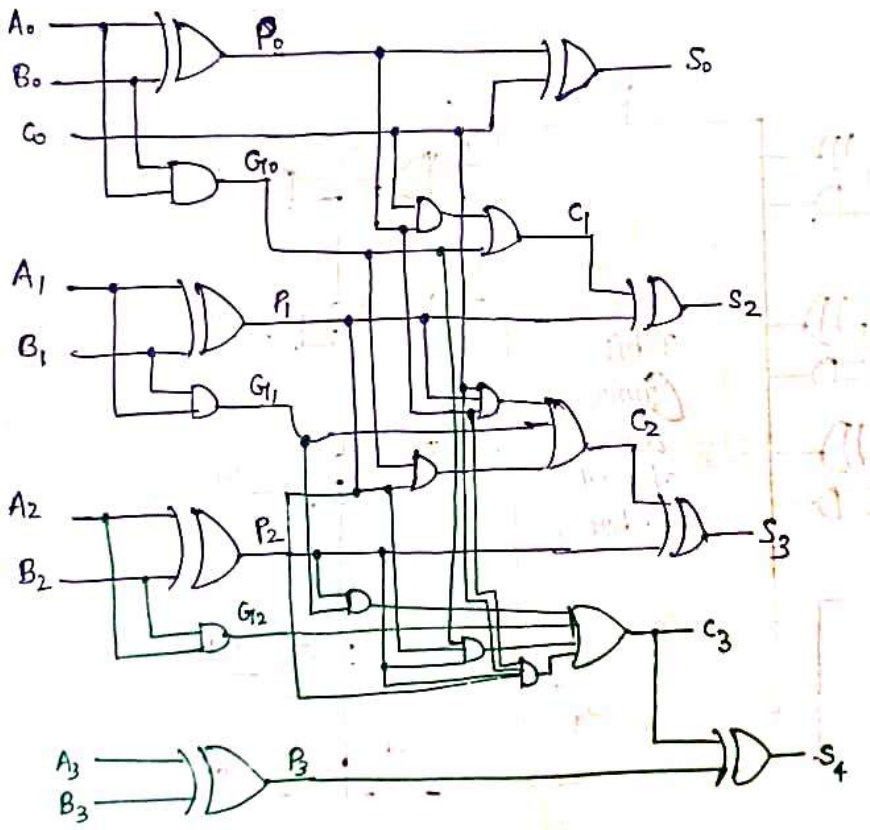
$= G_{12} + P_2 G_{11} + P_2 P_1 G_{10} + P_2 P_1 P_0 C_0$

$S_0 = P_0 \oplus C_0$

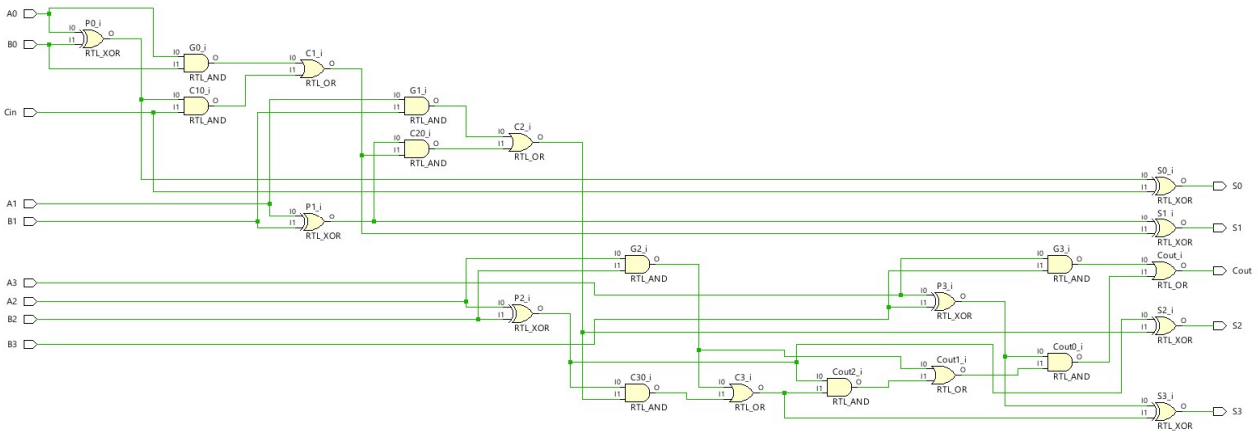
$S_1 = P_1 \oplus C_1$

$S_2 = P_2 \oplus C_2$

$S_3 = P_3 \oplus C_3$



Schematic:


A0
B0
Cin
A1
B1
A3
A2
B2
B3
P0_i
G0_i
C10_i
C1_i
G1_i
C20_i
C2_i
P1_i
G2_i
P2_i
C30_i
C3_i
P3_i
G3_i
Cout2_i
Cout1_i
Cout0_i
Cout_i
S0_i
S1_i
S2_i
S3_i
RTL_XOR
RTL_AND
RTL_OR
S0
S1
S2
S3
Cout