# NAME: DHVANI PATEL

# ROLL NO: 21BCP116

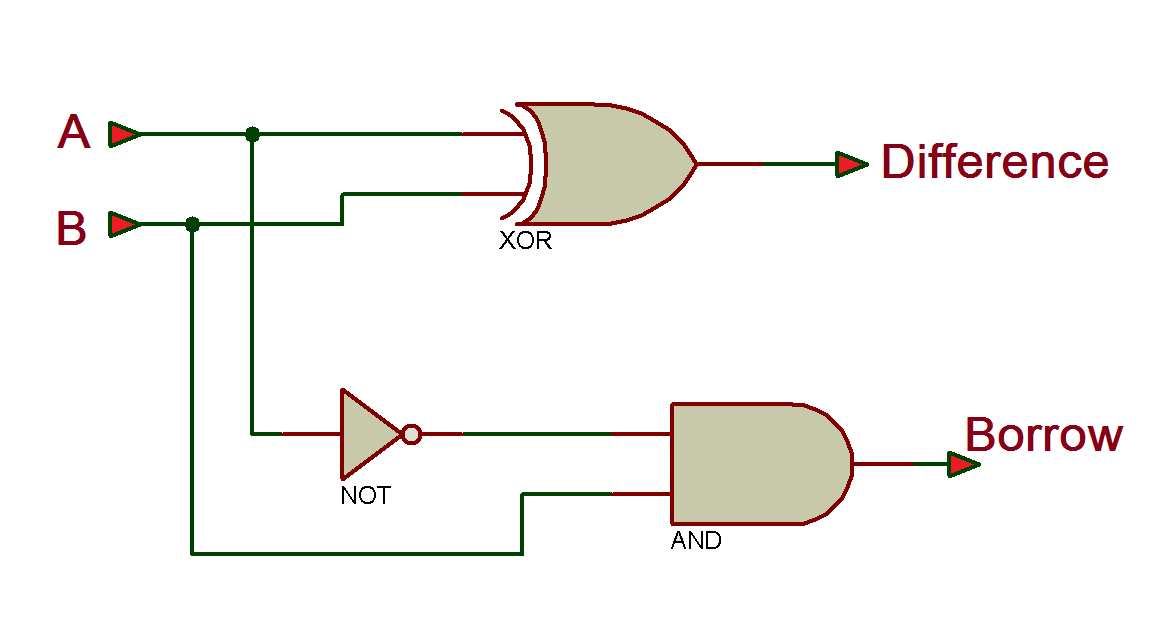
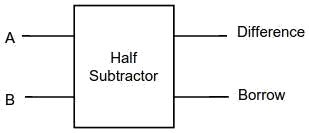
# DIVISION 2, G4

# DECO LAB

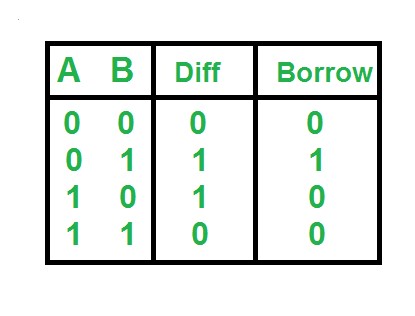
# Assignment 5 : Design a Half Subtractor, Full Subtractor using Half Subtractors

* **Description:**

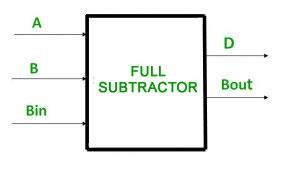
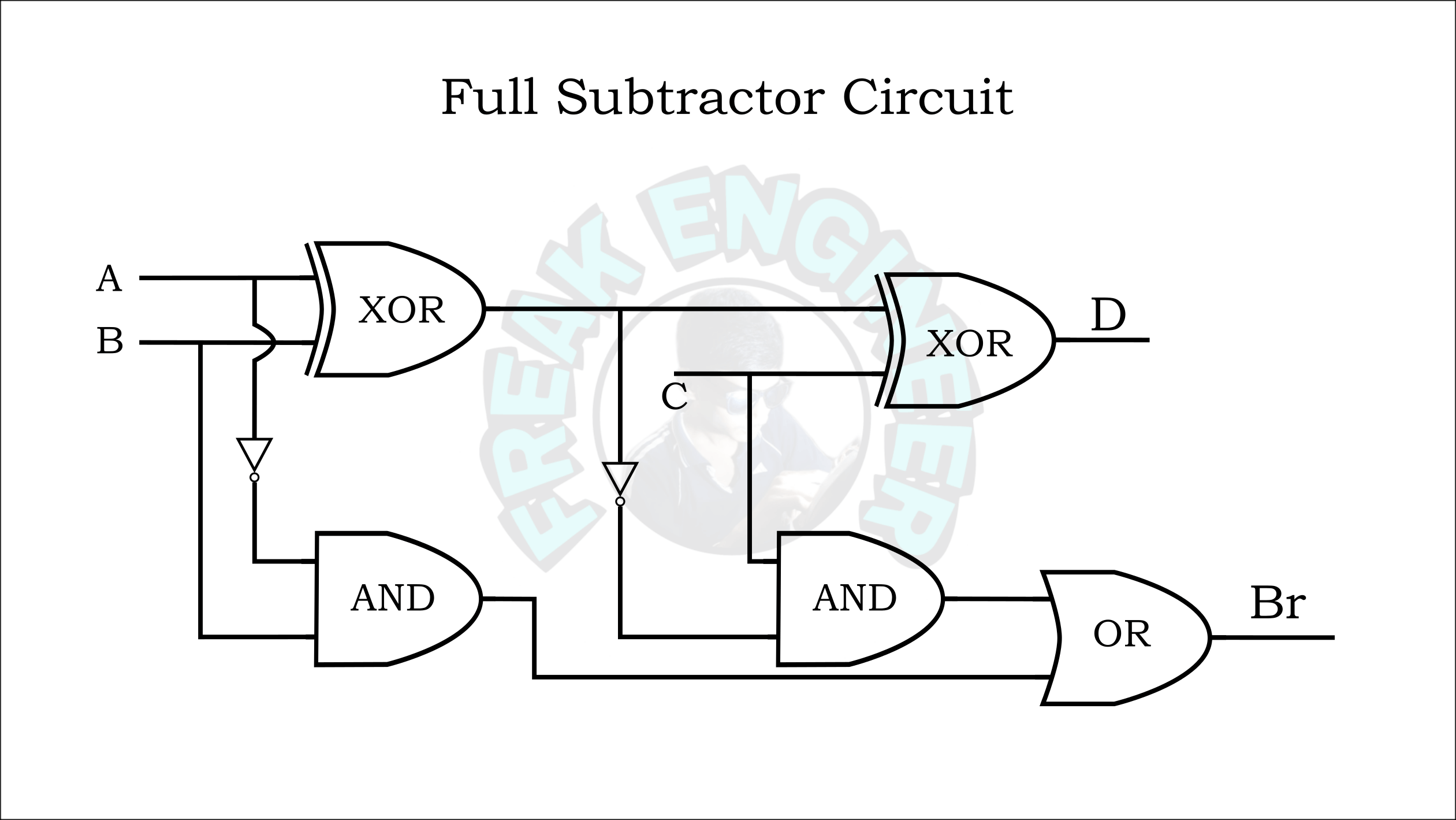
# Half Subtractor :

* ****The half subtractor is a [combinational circuit](https://en.wikipedia.org/wiki/Logic_circuit) which is used to perform subtraction of two bits. It has two inputs, the [minuend](https://en.wikipedia.org/wiki/Minuend) (**A**) and [subtrahend](https://en.wikipedia.org/wiki/Subtrahend) (**B**) and two outputs the difference **Diff** and borrows out **Bout**. Borrows out signal is set when the subtractor needs to borrow from the next digit in a multi digit subtraction. That is, **Bout = 1** when **A** < **B**. Since **A** and **B** are bits, **Bout = 1** if and only if **A = 0** and **B = 1**. An important point worth mentioning is that the half subtractor diagram aside implements **A - B** and not **B - A** since **Bout** on the diagram is given

by **Bout = NOT A. B**

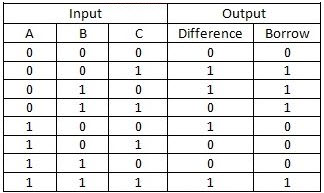
* **Truth Table :**

# Full Subtractor :

* A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit **has three inputs and two outputs**. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrows, respectively. Generally, the full subtractor is one of the most used and essential combinational logic circuits. It is a basic electronic device, used to perform subtraction of two binary numbers. Likewise, the full-subtractor uses binary digits like 0, 1 for the subtraction

# 

* **Truth Table :**

****

**Q 1. Write a Verilog code to design a Half subtractor and test it using the Waveform.**

* Code :

// Code your testbench here

module tb\_Half\_Subtractor;

reg A , B ;

wire S , C;

Half\_Subtractor a1(A , B, S, C);

initial

begin

A = 0 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=1; #1;

$display("a= %b , b = %b " , A , B);

$display("x= %b y=%b " , S,C);

A = 1 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=1; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Half\_Subtractor(a,b,diff ,bor);

input a,b;

output diff,bor;

wire a\_not;

not(a\_not,a);

xor( diff, a,b);

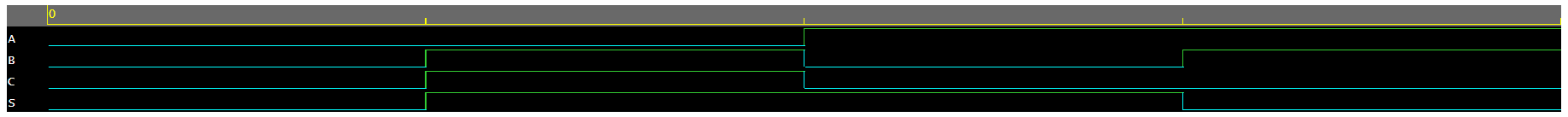
and( bor,a\_not,b);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.  
a= 0 , b = 0   
x= 0 y=0   
a= 0 , b = 1   
x= 1 y=1   
a= 1 , b = 0   
x= 1 y=0   
a= 1 , b = 1   
x= 0 y=0   
Finding VCD file...

* Waveform :



**Q 2. Write a Verilog code to design a Full subtractor using Half subtractors and test it using the Waveform.**

* Code :

// Code your testbench here

module tb\_Full\_Subtractor;

reg A , B , Cin ;

wire S , C;

Full\_Subtractor a1(Cin, A , B, S, C);

initial

begin

Cin=0; A = 0 ; B=0; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=0; A = 0 ; B=1; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B);

$display("x= %b y=%b " , S,C);

Cin=0; A = 1 ; B=0;#1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=0; A = 1 ; B=1; #1;

$display("cin= %b a= %b , b = %b " , Cin,A , B );

$display("x= %b y=%b " , S,C);

Cin=0; A = 0 ; B=0; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=1; A = 0 ; B=1; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B);

$display("x= %b y=%b " , S,C);

Cin=1; A = 1 ; B=0; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=1;A = 1 ; B=1; #1;

$display("cin= %b a= %b , b = %b " , Cin,A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Full\_Subtractor(bin ,a,b,diff,bout);

input a,b,bin;

output diff,bout;

wire x,y,z, a\_not,bin\_not;

xor(x, a,b);

not(a\_not , a);

and(y, a\_not ,b);

xor(diff , bin, x);

not(bin\_not,bin);

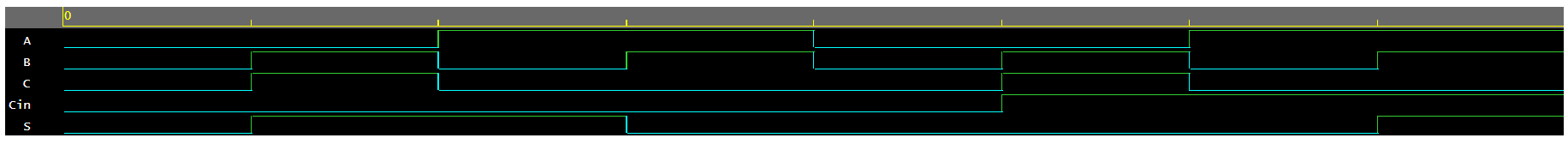
and(z , y ,bin\_not);

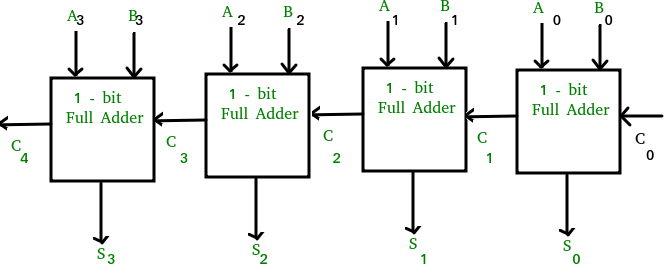
or(bout , z,y);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.  
cin= 0 a= 0 , b = 0   
x= 0 y=0   
cin= 0 a= 0 , b = 1   
x= 1 y=1   
cin= 0 a= 1 , b = 0   
x= 1 y=0   
cin= 0 a= 1 , b = 1   
x= 0 y=0   
cin= 0 a= 0 , b = 0   
x= 0 y=0   
cin= 1 a= 0 , b = 1   
x= 0 y=1   
cin= 1 a= 1 , b = 0   
x= 0 y=0   
cin= 1 a= 1 , b = 1   
x= 1 y=0   
Finding VCD file...  
./dump.vcd

* Waveform : 

**Q 3. Write the Verilog code (either structural or behavioral) for a 4-bit ripple carry adder. The block diagram of a 4-bit ripple carry adder has been given in Figure 1 for your reference.**

*Figure 1: The Block Diagram of four bit ripple carry adder.*

* Code :

// Code your testbench here

module tb\_RCA;

reg [3:0] A , B;

reg C;

wire [3:0]S;

wire Cout;

RCA rc(S,Cout , A,B,C);

initial

begin

A=1;

B=2;

C=0;

#2;

A=3;

B=1;

C=0;

#2;

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Full\_Adder(s,cout,a,b,cin);

input cin,a,b;

output s,cout;

wire p,q ,r;

xor(p,a,b);

xor(s,p,cin);

and(q,a,b);

and(r,p,cin);

or(cout,q,r);

endmodule

module RCA(S,CO , x,y,z);

output CO;

output[3:0] S;

input [3:0] x,y;

input z;

wire w1,w2,w3;

Full\_Adder fa1(S[0],w1,x[0],y[0],z);

Full\_Adder fa2(S[1],w2,x[1],y[1],w1);

Full\_Adder fa3(S[2],w3,x[2],y[2],w2);

Full\_Adder fa4(S[3],CO,x[3],y[3],w3);

endmodule

* Output :

a=1101,b=0011 and c=0

y=0000, y1=1

a=0101,b=0010 and c=0

y=0111, y1=0

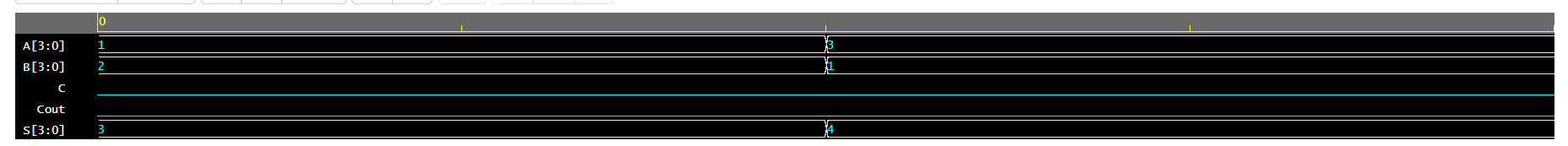
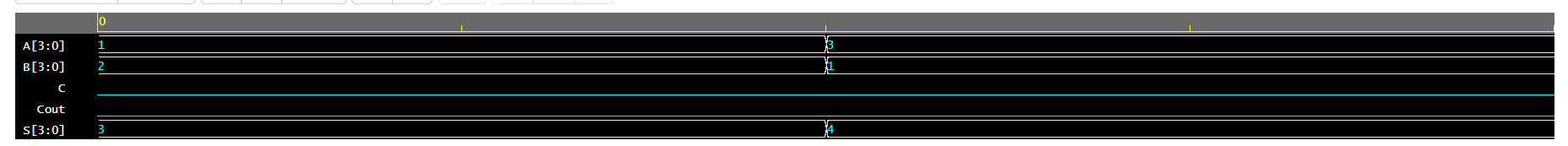
a=0001,b=0001 and c=0

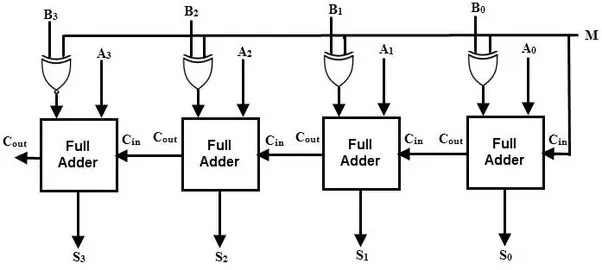
y=0010, y1=0

a=0110,b=0011 and c=0

y=0010, y1=0

* Waveform :



**Q 4. Write the Verilog code to construct a 4-bit adder-subtractor using. The logic diagram of a 4-bit adder-subtractor using a ripple carry adder has been given in Figure 2 for your reference.**

*Figure 2: The Block Diagram of four bit adder-cum-subtractor bit.*

* Code :

// Code your testbench here

module tb\_RCAS;

reg [3:0] A , B;

reg C,M;

wire [3:0]S;

wire Cout;

RCAS rc(S,Cout , A,B,M);

initial

begin

A=1;

B=2;

M=0;

#1;

A=2;

B=1;

M=1;

#1;

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Full\_Adder(s,cout,a,b,cin);

input cin,a,b;

output s,cout;

wire p,q ,r;

xor(p,a,b);

xor(s,p,cin);

and(q,a,b);

and(r,p,cin);

or(cout,q,r);

endmodule

module RCAS(S,CO , x,y,M);

output CO;

output[3:0] S;

input [3:0] x,y;

input M;

wire w1,w2,w3;

wire y0,y1,y2,y3;

xor(y0,M,y[0]);

xor(y1,M,y[1]);

xor(y2,M,y[2]);

xor(y3,M,y[3]);

Full\_Adder fa1(S[0],w1,x[0],y0,M);

Full\_Adder fa2(S[1],w2,x[1],y1,w1);

Full\_Adder fa3(S[2],w3,x[2],y2,w2);

Full\_Adder fa4(S[3],CO,x[3],y3,w3);

Endmodule

* Output :

a=0011,b=0010 and c=0

y=0101, y1=0

a=0101,b=0001 and c=1

y=0100, y1=1

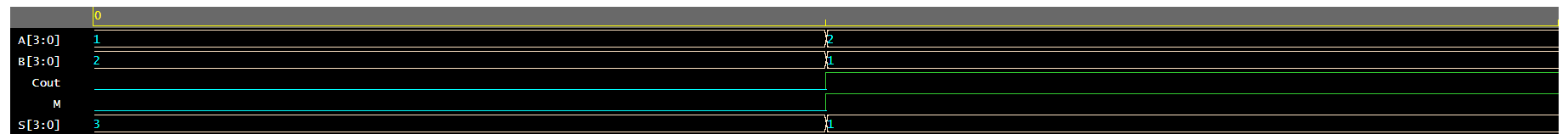
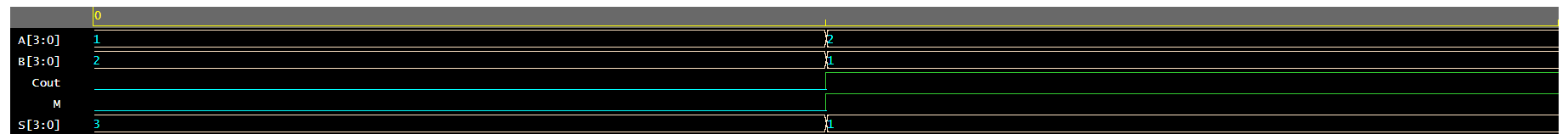
a=0100,b=0011 and c=0

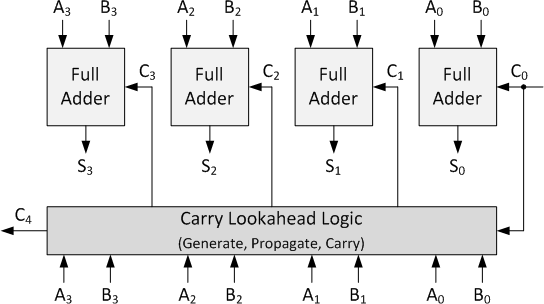
y=0111, y1=0

a=0110,b=0010 and c=1

y=0100, y1=1

* Waveform :



**Q 5. Write the Verilog code for a 4-bit carry look-ahead adder (CLA). The block diagram of the same has been given in Figure 3.**

*Figure 3: The Block Diagram of four bit carry look-ahead adder (CLA).*

* Code :

// Code your testbench here

module TestModule;

reg [3:0] A,B;

reg C;

wire [3:0] S;

wire cout;

CLA\_Adder uut (A,B,C,S,cout);

initial

begin

A = 4; B = 0; C = 1; #10;

$display("a=%b,b=%b and c=%b",A,B,C);

$display("y=%b, y1=%b",S,cout);

A = 2; B = 3; C = 1; #10;

$display("a=%b,b=%b and c=%b",A,B,C);

$display("y=%b, y1=%b",S,cout);

A = 3; B = 2;C = 1; #50;

$display("a=%b,b=%b and c=%b",A,B,C);

$display("y=%b, y1=%b",S,cout);

A = 7; B = 10; C = 0; #50;

$display("a=%b,b=%b and c=%b",A,B,C);

$display("y=%b, y1=%b",S,cout);

A = 15; B = 15; C = 1; #50;

$display("a=%b,b=%b and c=%b",A,B,C);

$display("y=%b, y1=%b",S,cout);

End

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module CLA\_Adder(a,b,cin,sum,cout);

input[3:0] a,b;

input cin;

output [3:0] sum;

output cout;

wire p0,p1,p2,p3,g0,g1,g2,g3,c0,c1,c2,c3,c4;

xor (p0, a[0], b[0]);

xor (p1, a[1], b[1]);

xor (p2, a[2], b[2]);

xor (p3, a[3], b[3]);

and (g0, a[0], b[0]);

and (g1, a[1], b[1]);

and (g2, a[2], b[2]);

and (g3, a[3], b[3]);

assign c0=cin,

c1=g0|(p0&cin),

c2=g1|(p1&g0)|(p1&p0&cin),

c3=g2|(p2&g1)|(p2&p1&g0)|(p1&p1&p0&cin),

c4=g3|(p3&g2)|(p3&p2&g1)|(p3&p2&p1&g0)|(p3&p2&p1&p0&cin);

assign sum[0]=p0^c0,

sum[1]=p1^c1,

sum[2]=p2^c2,

sum[3]=p3^c3;

assign cout=c4;

endmodule

* Output :

a=0100,b=0000 and c=1

y=0101, y1=0

a=0010,b=0011 and c=1

y=0110, y1=0

a=0011,b=0010 and c=1

y=0110, y1=0

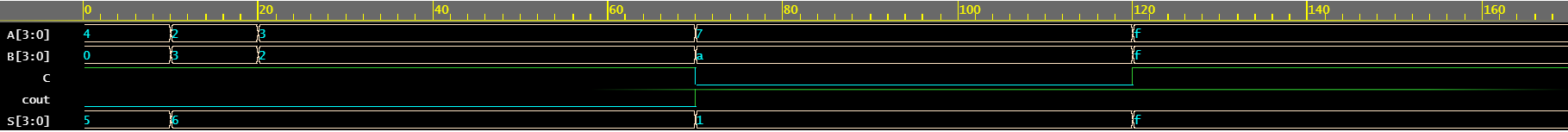
a=0111,b=1010 and c=0

y=0001, y1=1

a=1111,b=1111 and c=1

y=1111, y1=1

* Waveform :

****