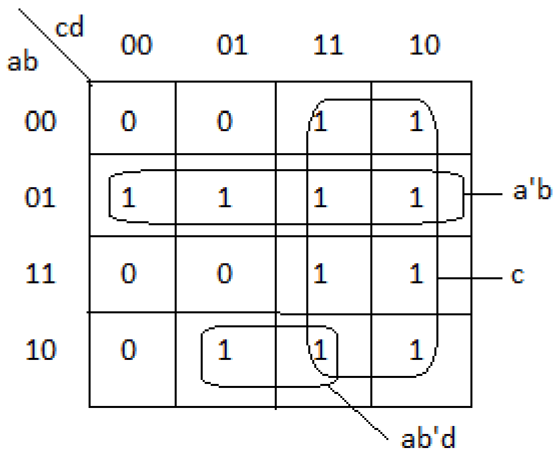
Program No. 1

1.Given a 4-variable logic expression, simplify it using appropriate technique and simulate the same using basic gates.

b) Simplification of 4 variable SOP expression:

$Y = f(a,b,c,d) = \Sigma m(2,3,4,5,6,7,9,10,11,14,15)$.

K-MAP:



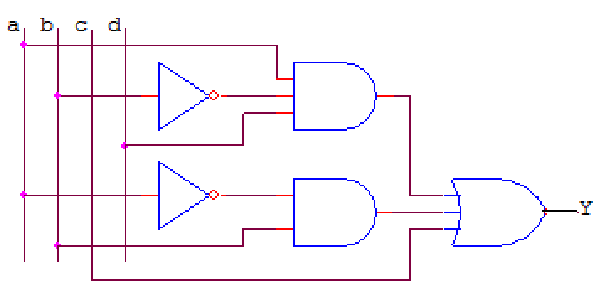
TRUTH TABLE:

a	b	c	d	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

EQUATION:

Y= c + a`b + ab`d

LOGIC DIAGRAM:



PROGRAM:

```
module eqn2 (a,b,c,d,Y);
input a,b,c,d;
output Y ;
assign Y= ( c | (~a&b) | ( (a&~b& d) ));
endmodule
```

EXERCISE:

c) Simplification of 3 variable POS expression

$Y= f(a,b,c) = \Pi M(0,1,2,4,5)$.

d) Simplification of 4 variable POS expression

$Y= f(a,b,c,d) = \Pi M(2,3,4,5,6,7,8,12,13,14,15)$.

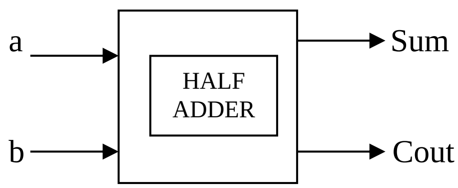
Program No. 2

2.Design Verilog HDL to implement Binary Adder-Subtractor – Half and Full Adder, Half and Full Subtractor.

i) <u>HALF ADDER</u>

Half adder is an arithmetic combinational logic circuit designed to perform addition of two single bits. It contains two inputs and produces two outputs Sum and Carry.

BLOCK DIAGRAM:



TRUTH TABLE :

Input		Output	
a	b	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K –MAPS:

Sum:

a \ b	0	1
0	0	1
1	1	0

Cout:

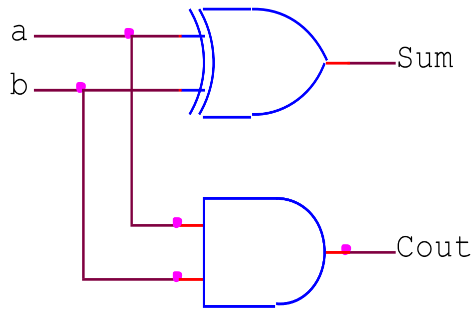
a \ b	0	1
0	0	0
1	0	1

EQUATIONS:

Sum= $a \oplus b$

Cout= a.b

LOGIC CIRCUIT:



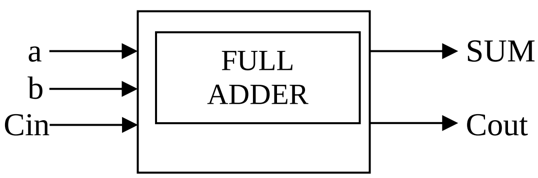
PROGRAM:

```
module    half_add(a,b, Sum,Cout);
input  a,b;
output Sum,Cout;
assign Sum= a ^b;
assign Cout = a & b;
endmodule
```

i) FULL ADDER

Full adder is an arithmetic combinational logic circuit that performs addition of three single bits. It contains three inputs (a, b, C_{in}) and produces two outputs (Sum and C_{out}) where, C_{in} is Carry In and C_{out} is Carry Out.

BLOCK DIAGRAM:

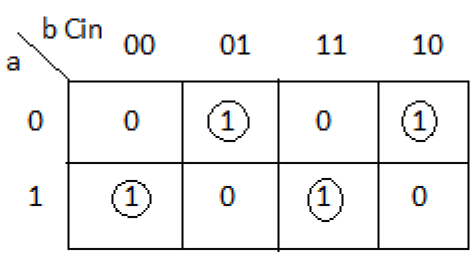


TRUTH TABLE:

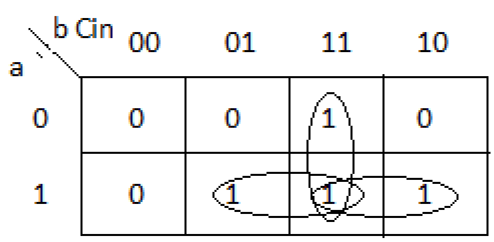
Input			Output	
a	b	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K –MAPS:

Sum:



Cout:

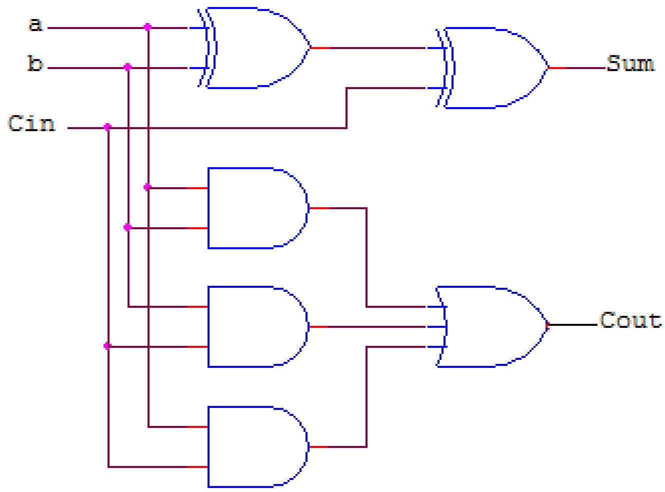


EQUATIONS:

Sum = a $\oplus$ b$\oplus$ Cin

Cout = a b + b Cin + a Cin

LOGIC CIRCUIT:



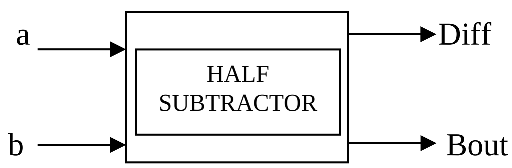
PROGRAM:

```
module full_add(a,b,Cin,Sum,Cout);
input  a,b,Cin;
output  Sum,Cout;
assign Sum= a ^b^Cin;
assign Cout = (a&b) | (b&Cin) | (a&Cin);
endmodule
```

i) HALF SUBTRACTOR

Half subtractor is a combinational logic circuit designed to perform the subtraction of two single bits. It contains two inputs (a and b) and produces two outputs (Difference and Borrow-out).

BLOCK DIAGRAM:

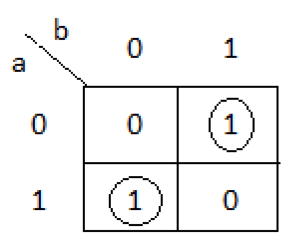


TRUTH TABLE:

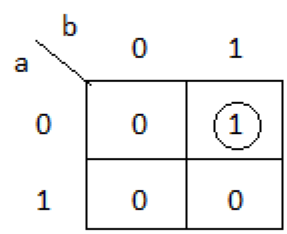
Input		Output	
a	b	Diff	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K –MAPS:

Diff:



Bout:

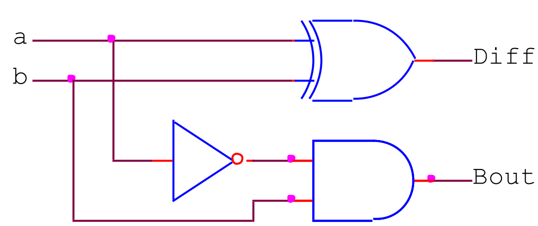


EQUATIONS:

Diff = $a \oplus b$

Bout= a`b

LOGIC DIAGRAM:



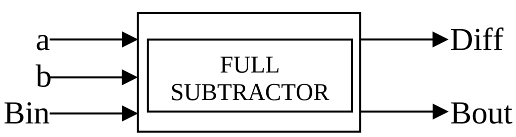
PROGRAM:

```
module  half_sub(a,b, Diff,Bout);
input  a,b;
output Diff,Bout;
assign Diff= a ^ b;
assign Bout = ~a  & b;
endmodule
```

i) FULL SUBTRACTOR

Full subtractor is a Combinational logic circuit designed to perform subtraction of three single bits. It contains three inputs(a, b, B_{in}) and produces two outputs (Diff, B_{out}) where B_{in} is Borrow-In and B_{out} is Borrow-Out.

BLOCK DIAGRAM:

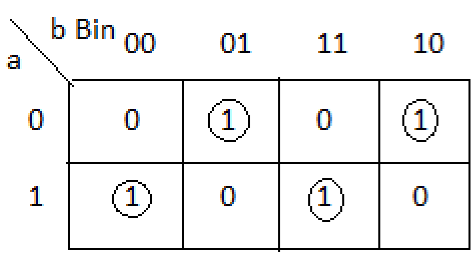


TRUTH TABLE :

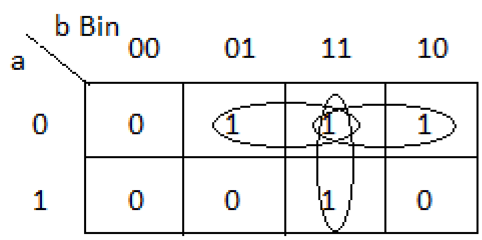
Input			Output	
a	b	Bin	Diff	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K –MAP:

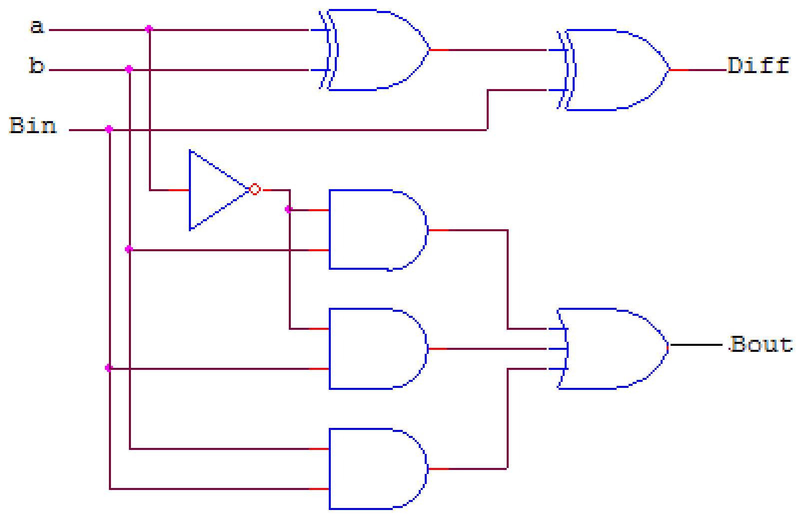
Diff:



Bout:



LOGIC CIRCUIT:



PROGRAM:

```
module full_sub(a,b,Bin, Diff,Bout);
input  a,b,Bin;
output  Diff, Bout;
assign Diff= a ^b ^Bin;
assign Bout = (~a & b) | (~a & Bin) | (b & Bin)
endmodule
```

EQUATIONS:

$\text{Diff} = a \oplus b \oplus \text{Bin}$

$\text{Bout} = a`\, b + a`\, \text{Bin} + b\, \text{Bin}$

3.Design Verilog HDL to implement simple circuits using structural, Data flow and Behavioural model.

Structural Model

```
module p3structural(a,b,c,d,e,y);
input a;
input b;
input c;
input d;
input e;
output y;
wire Y1,Y2;
and G1(Y1,a,b);
and G2(Y2,c,d,e);
or G3(Y,Y1,Y2);
endmodule
```

Data Flow Model

```
module p31(a,b,c,d,e,y);
input a;
input b;
input c;
input d;
input e;
output y;
wire Y1,Y2;
assign Y1=a & b;
assign Y2= c&d&e;
assign y= Y1|Y2;
endmodule
```

Behavioral Model

```
module p3behavioral(a,b,c,d,e,y);
input a;
input b;
input c;
input d;
input e;
output y;
reg y;
always @(a,b,c,d,e)
begin
```

```
y= (a & b) | (c & d &e);
end
endmodule
```

4.Design a 4-bit full adder and subtractor and simulate the same using basic gates.

Full adder
BOOLEAN EXPRESSIONS:
sum= A ⨁ B ⨁ C

Full adder

$$Sum = A \oplus B \oplus C$$

To be implemented using Basic gates.

$$(A \oplus B) \oplus C$$

$$\left(\overline{A \oplus B}\right) C + (A \oplus B)\,\overline{C}$$

$$\left(\overline{\bar{A}B + A\bar{B}}\right) C + \left(\bar{A}B + A\bar{B}\right)\overline{C}$$

$$\left(\overline{\bar{A}B} \cdot \overline{A\bar{B}}\right) C + \left(\bar{A}B + A\bar{B}\right)\overline{C}$$

$$\left((\bar{\bar{A}} + \bar{B}) \cdot (\bar{A} + \bar{\bar{B}})\right) C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

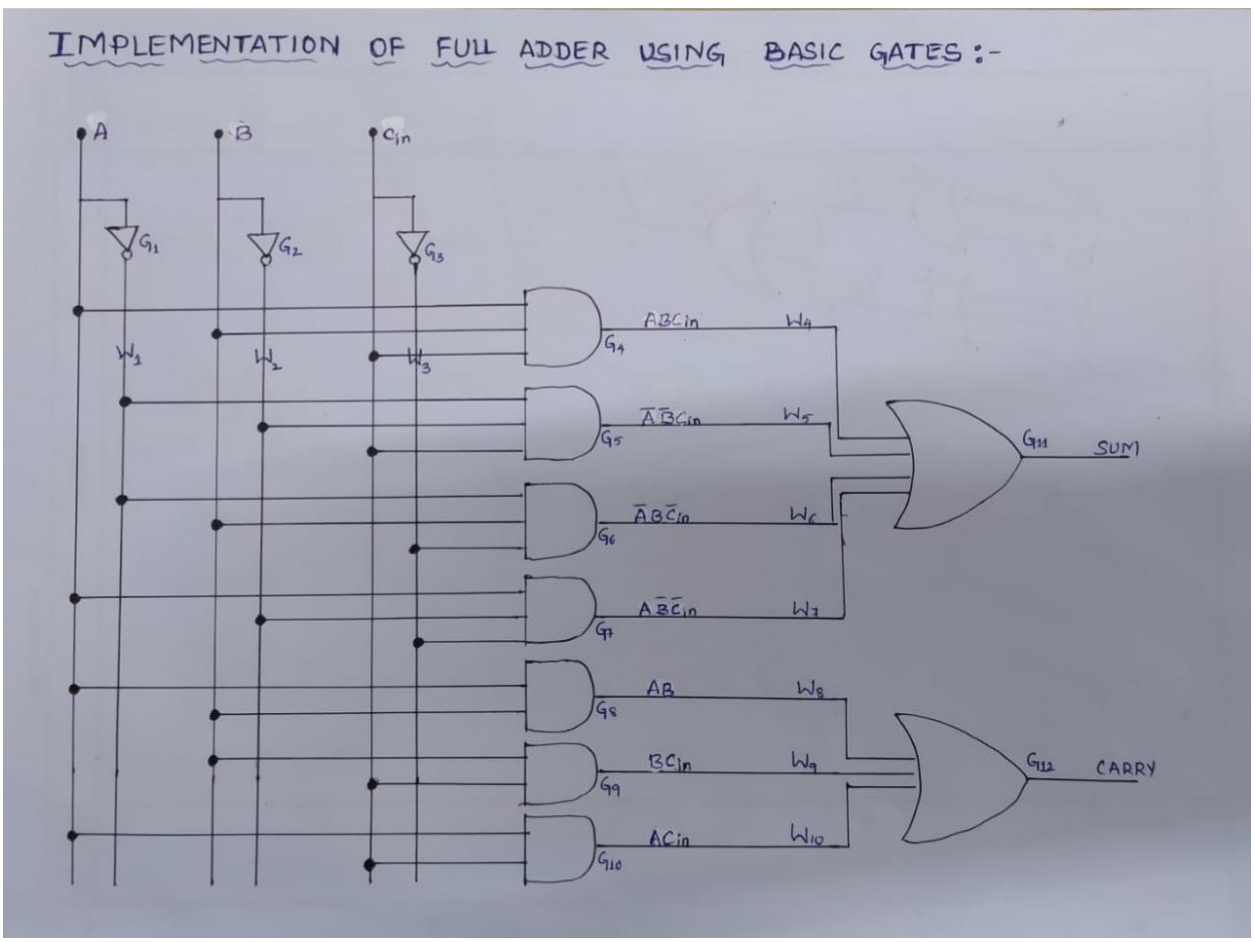
$$\left((A + \bar{B}) \cdot (\bar{A} + B)\right) C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

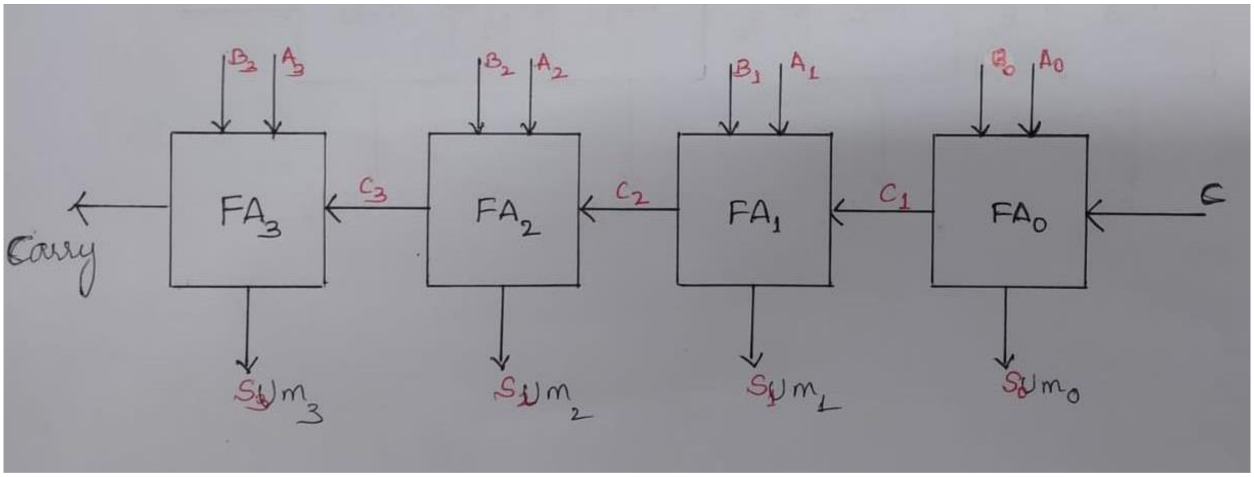
$$\left(\cancel{A\bar{A}} + AB + \bar{A}\bar{B} + \cancel{B\bar{B}}\right) C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

$$Sum = \boxed{ABC + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}}$$

carry=A B + B C + A C
TRUTH TABLE:

INPUTS			OUTPUTS	
A	B	C	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

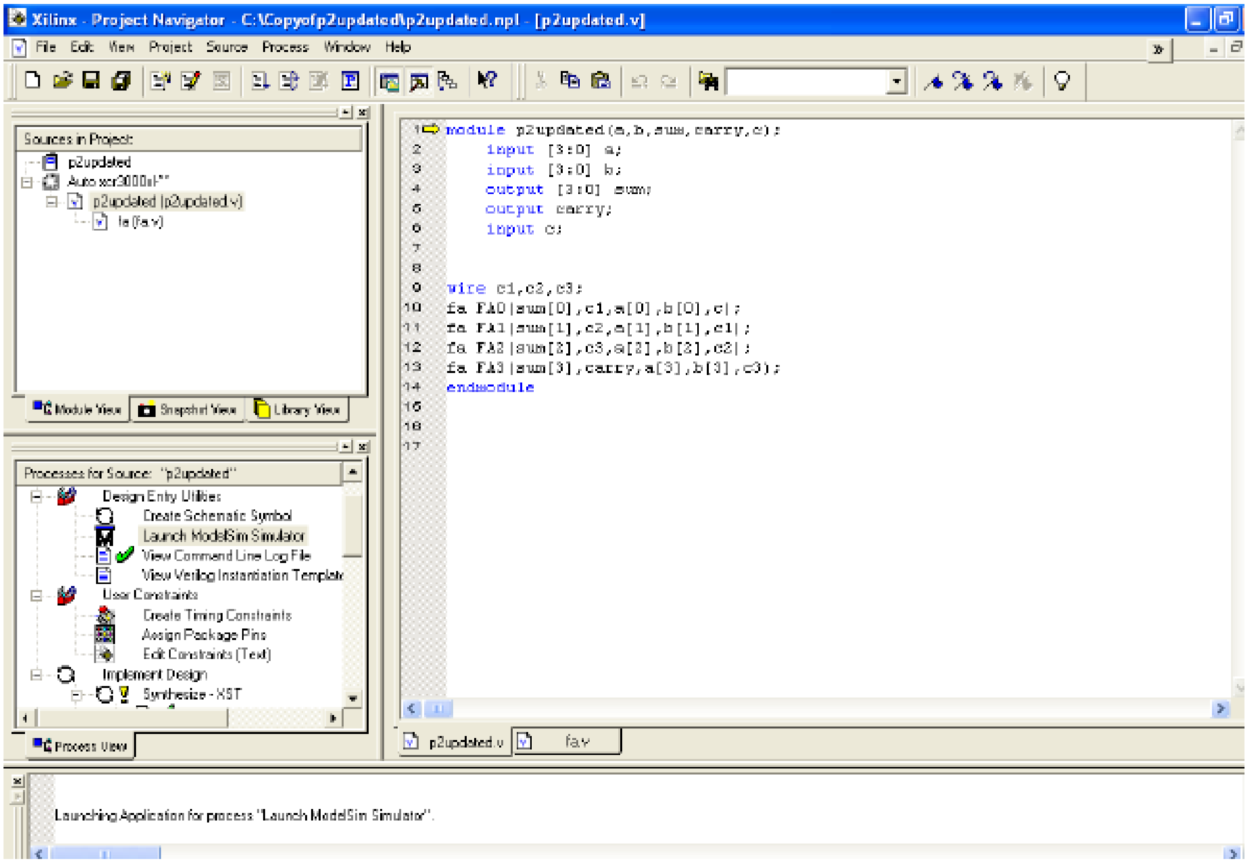




Full adder: -

Step 1

```
module p2updated(a,b,sum,carry,c);
input [3:0] a;
input [3:0] b;
output [3:0] sum;
output carry;
input c;
wire c1,c2,c3;
fa FA0(sum[0],c1,a[0],b[0],c);
fa FA1(sum[1],c2,a[1],b[1],c1);
fa FA2(sum[2],c3,a[2],b[2],c2);
fa FA3(sum[3],carry,a[3],b[3],c3);
endmodule
```



Step 2

(create the file fa.v under the main module p2updated.v)

i) right click on main module p2updated.v

ii) select new source ->Verilog module ->enter file name as fa.v

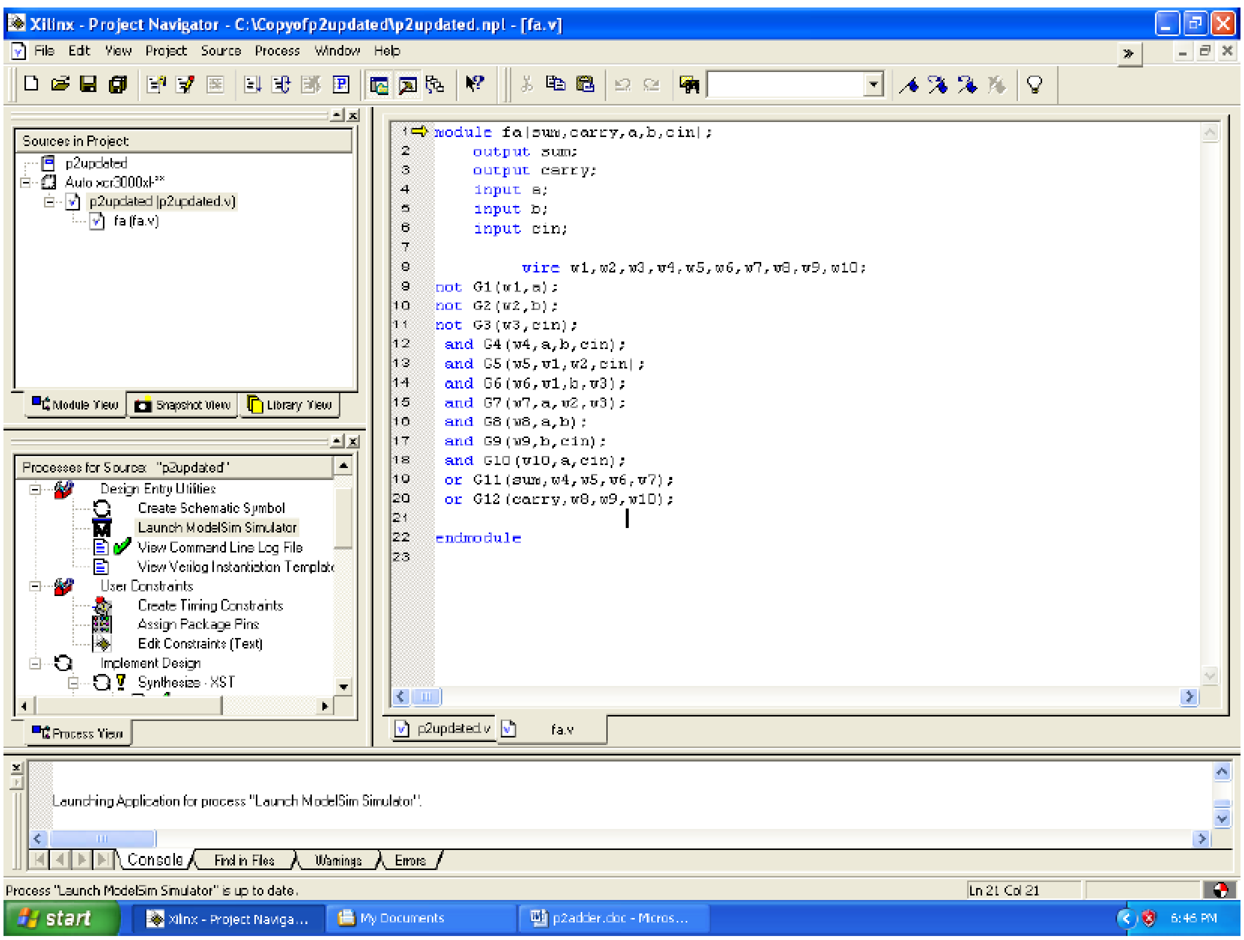
iii) fa.v sub module gets created under the main module as seen in the process window

iv) type the code in the sub module fa.v and save.

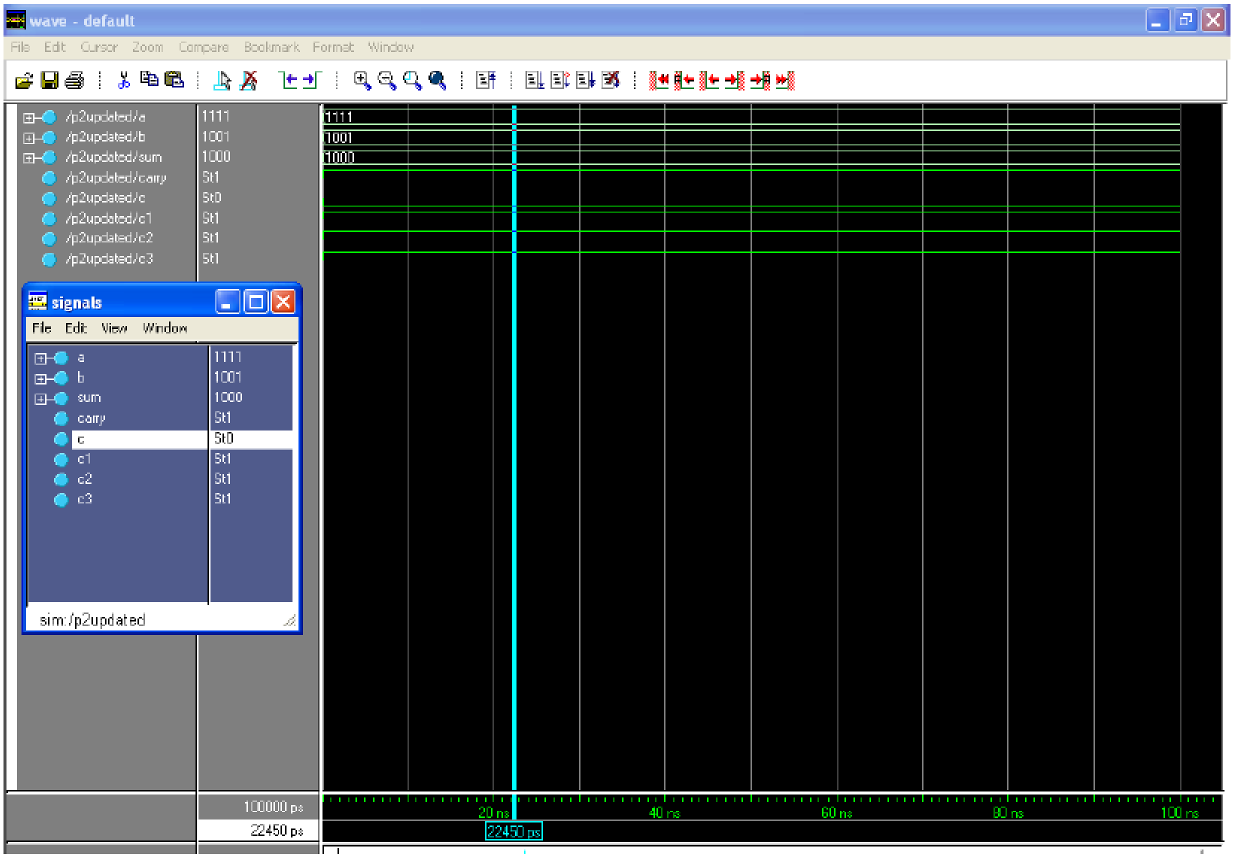
v) compile both the main module p2updated.v and sub module fa.v

```
module fa(sum,carry,a,b,cin);

output sum;
output carry;
input a;
input b;
input cin;
wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10;
```

```
not G1(w1,a);
not G2(w2,b);
not G3(w3,cin);
and G4(w4,a,b,cin);
and G5(w5,w1,w2,cin);
and G6(w6,w1,b,w3);
and G7(w7,a,w2,w3);
and G8(w8,a,b);
and G9(w9,b,cin);
and G10(w10,a,cin);
or G11(sum,w4,w5,w6,w7);
or G12(carry,w8,w9,w10);

endmodule
```



Step1 output



Step2 output