AIM: To simulate and synthesis Logic Gates ,Adders and Subtractor using Xilinx ISE.

APPARATUS REQUIRED: Xilinx 14.7 Spartan6 FPGA

PROCEDURE:

STEP:1 Start the Xilinx navigator, Select and Name the New project.

STEP:2 Select the device family, device, package and speed.

STEP:3 Select new source in the New Project and select Verilog Module as the Source type.

STEP:4 Type the File Name and Click Next and then finish button. Type the code and save it.

STEP:5 Select the Behavioral Simulation in the Source Window and click the check syntax.

STEP:6 Click the simulation to simulate the program and give the inputs and verify the outputs as per the truth table.

STEP:7 Select the Implementation in the Sources Window and select the required file in the Processes Window.

STEP:8 Select Check Syntax from the Synthesize XST Process. Double Click in the Floorplan Area/IO/Logic-Post Synthesis process in the User Constraints process group. UCF(User constraint File) is obtained.

STEP:9 In the Design Object List Window, enter the pin location for each pin in the Loc column Select save from the File menu.

STEP:10 Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

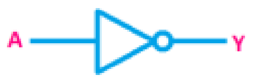
STEP:11 Load the Bit file into the SPARTAN 6 FPGA

STEP:12 On the board, by giving required input, the LEDs starts to glow light, indicating the output.

Logic Diagram :

LogicGates:

NOT Gate

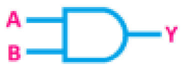


$Y = A'$

INPUT	OUTPUT
A	(Y)
0	1
1	0

AND Gate

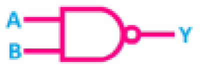
INPUT		OUTPUT
A	B	(Y)
0	0	0
0	1	0
1	0	0
1	1	1



$Y = A . B$

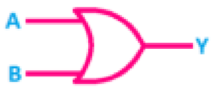
NAND Gate

INPUT		OUTPUT
A	B	(Y)
0	0	1
0	1	1
1	0	1
1	1	0



$Y = \overline{A . B}$

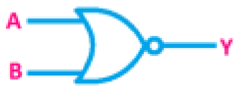
OR Gate



$Y = A + B$

INPUT		OUTPUT
A	B	(Y)
0	0	0
0	1	1
1	0	1
1	1	1

NOR Gate

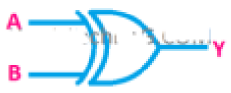


$Y = \overline{A + B}$

INPUT		OUTPUT
A	B	(Y)
0	0	1
0	1	0
1	0	0
1	1	0

XOR Gate

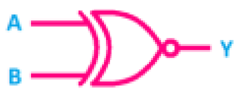
INPUT		OUTPUT
A	B	(Y)
0	0	0
0	1	1
1	0	1
1	1	0



$Y = A \oplus B$

XNOR Gate

INPUT		OUTPUT
A	B	(Y)
0	0	1
0	1	0
1	0	0
1	1	1

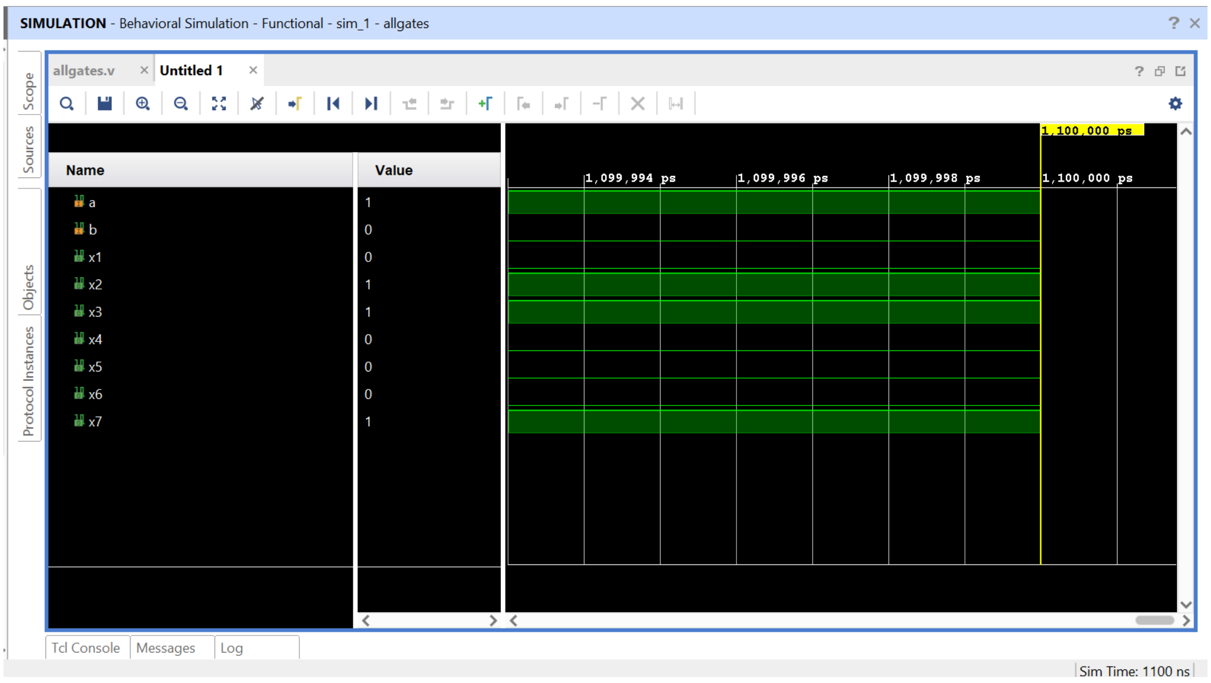


$Y = \overline{A \oplus B}$

VERILOG CODE:

```
module allgates(a,b,x1,x2,x3,x4,x5,x6,x7);
input a,b;
output x1,x2,x3,x4,x5,x6,x7;
and g1(x1,a,b);
nand g2(x2,a,b);
or g3(x3,a,b);
nor g4(x4,a,b);
xnor g5(x5,a,b);
not g6(x6,a);
xor g7(x7,a,b);
endmodule
```

OUTPUT:

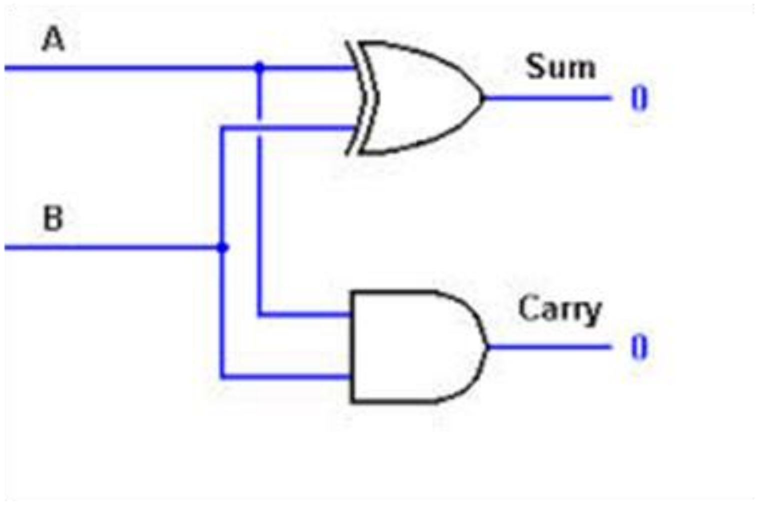


```
Name    Value
a       1
b       0
x1      0
x2      1
x3      1
x4      0
x5      0
x6      0
x7      1
```

1,099,994 ps 1,099,996 ps 1,099,998 ps 1,100,000 ps

Sim Time: 1100 ns

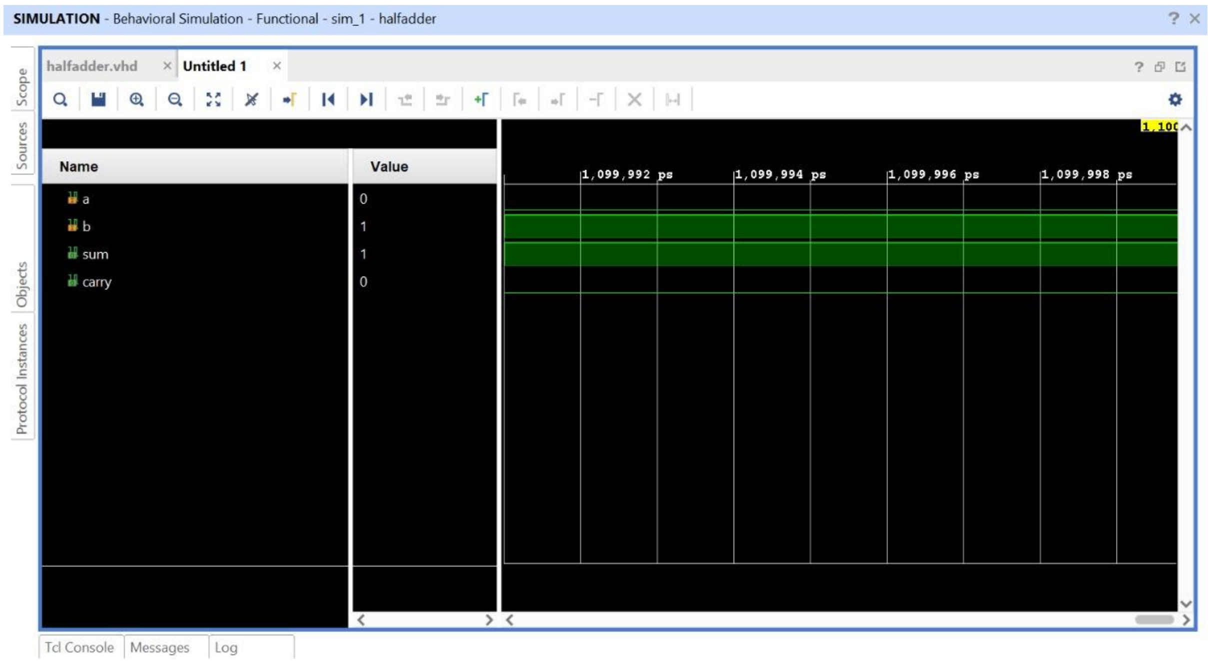
Half Adder:



VERILOG CODE:

```
module hadder(x,y,a,b);
input x,y;
output a,b;
xor g1(a,x,y);
and g2(b,x,y);
endmodule
```

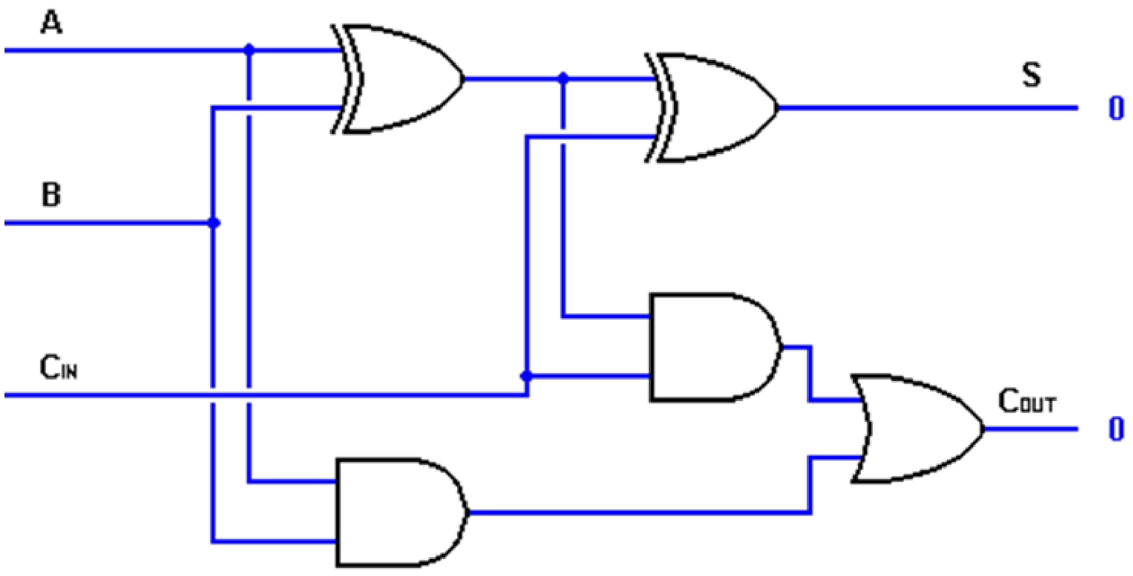
OUTPUT:


SIMULATION - Behavioral Simulation - Functional - sim_1 - halfadder

halfadder.vhd Untitled 1

Name	Value
a	0
b	1
sum	1
carry	0

Full adder:

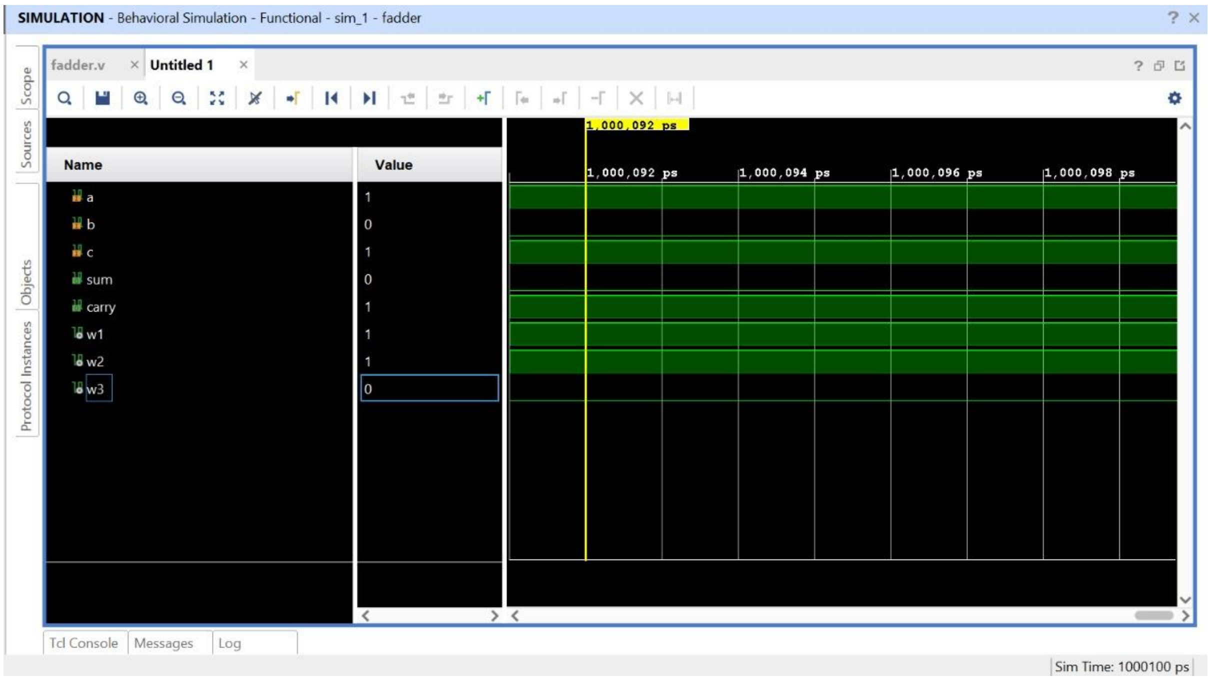


VERILOG CODE:

```
module fadder(a,b,c,sum,carry);
input a,b,c;
output sum ,carry;
wire w1,w2,w3;
xor g1(w1,a,b);
and g2(w3,a,b);
```

```
xor g3(sum,w1,c);
and g4(w2,w1,c);
or g5(carry,w2,w3);
endmodule
```

OUTPUT:



SIMULATION - Behavioral Simulation - Functional - sim_1 - fadder

fadder.v × Untitled 1 ×

Name	Value
a	1
b	0
c	1
sum	0
carry	1
w1	1
w2	1
w3	0

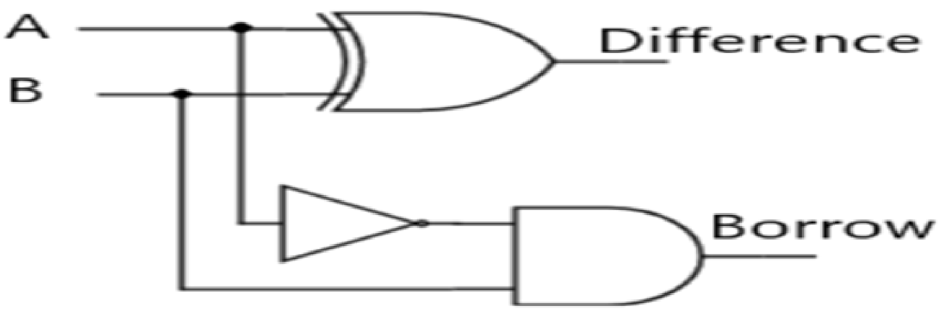
1,000,092 ps

1,000,092 ps   1,000,094 ps   1,000,096 ps   1,000,098 ps

Tcl Console   Messages   Log

Sim Time: 1000100 ps

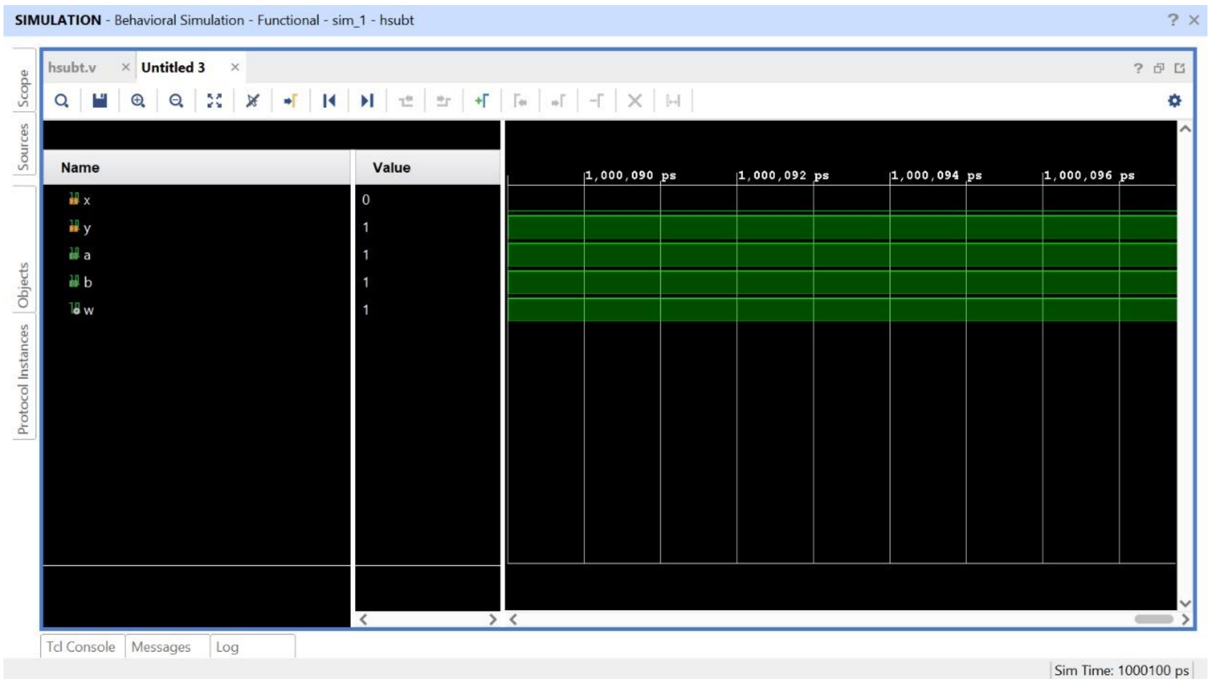
Half Subtractor:



VERILOG CODE:

```
module hsubt(x,y,a,b);
input x,y;
output a,b;
wire w;
xor g1(a,x,y);
not g2(w,x);
and g3(b,w,y);
endmodule
```

OUTPUT:

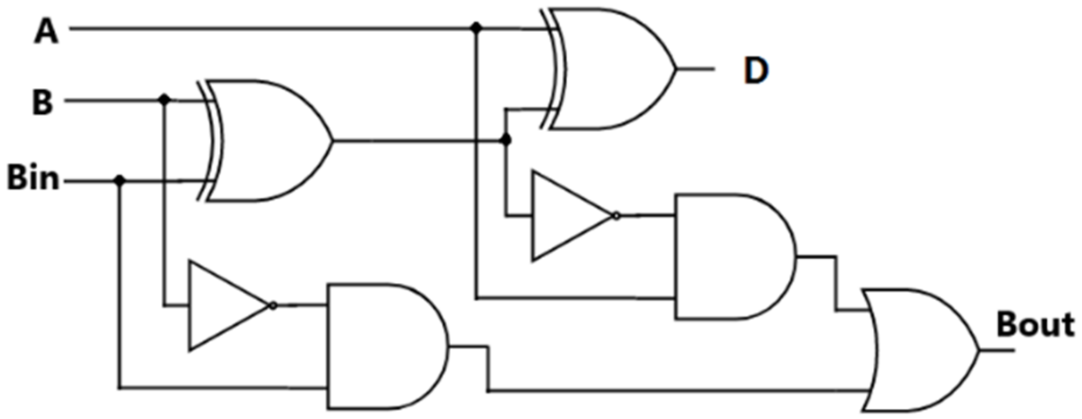


```
Name    Value
x       0
y       1
a       1
b       1
w       1
```

```
1,000,090 ps    1,000,092 ps    1,000,094 ps    1,000,096 ps
```

Sim Time: 1000100 ps

Full Subtractor:

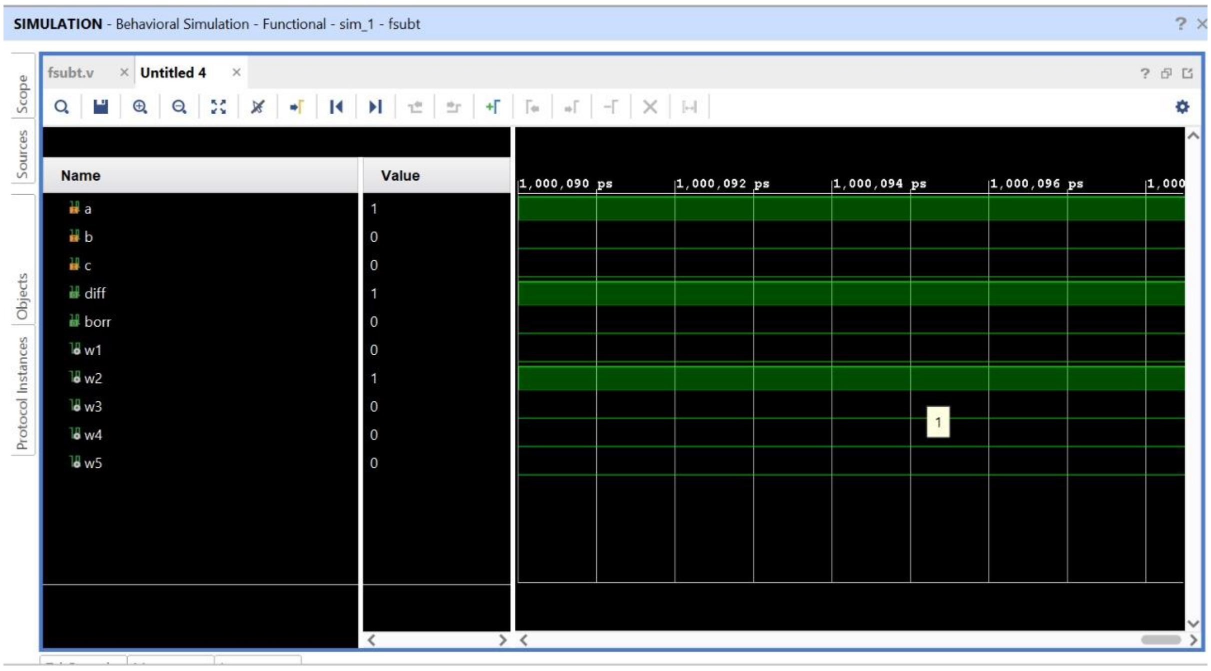


VERILOG CODE:

```
module fsubt(a,b,c,diff,borr);
input a,b,c;
output diff,borr;
wire w1,w2,w3,w4,w5;
xor g1(w2,a,b);
not g2(w1,a);
xor g3(diff,w2,c);
and g4(w4,w1,b);
not g5(w3,w2);
and g6(w5,w3,c);
or g7(borr,w5,w4);
```

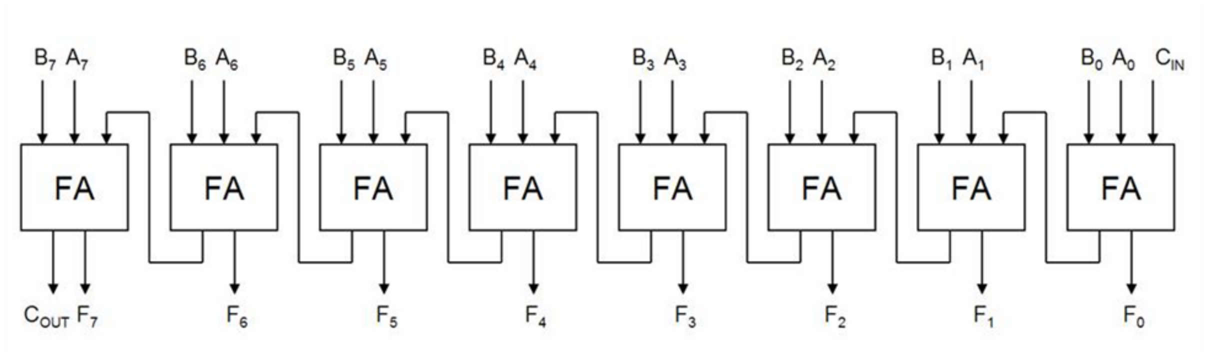
```
endmodule
```

OUTPUT:


SIMULATION - Behavioral Simulation - Functional - sim_1 - fsubt

```
Name    Value
a       1
b       0
c       0
diff    1
borr    0
w1      0
w2      1
w3      0
w4      0
w5      0
```

8 Bit Ripple Carry Adder

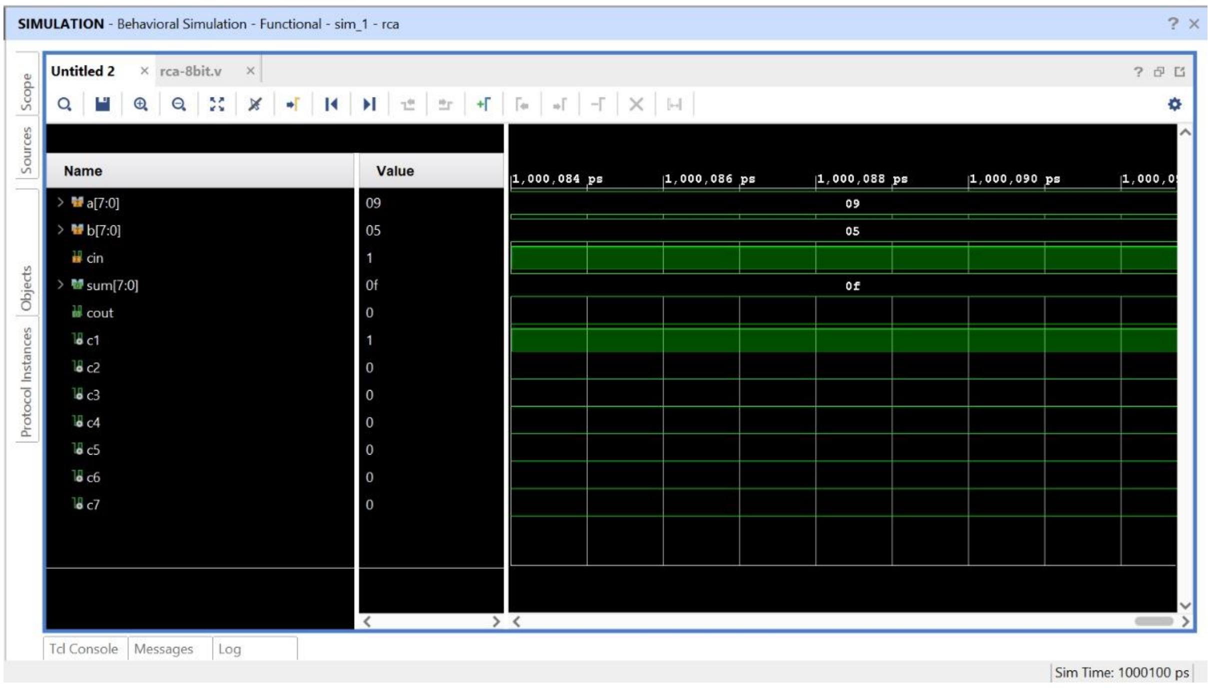


VERILOG CODE:

```
module fa(a,b,cin,sum,carry);
input a,b,cin;
output sum,carry;
wire w1,w2,w3;
xor g1(w1,a,b);
and g2(w3,a,b);
xor g3(sum,w1,cin);
and g4(w2,w1,cin);
or g5(carry,w2,w3);
```

```
endmodule
module rca(a,b,cin,sum,cout);
input [7:0]a,b;
input cin;
output [7:0]sum;
output cout;
wire c1,c2,c3;
fa g1(.a(a[0]),
      .b(b[0]),
      .cin(cin),
      .sum(sum[0]),
      .carry(c1)
      );
fa g2(.a(a[1]),
      .b(b[1]),
      .cin(c1),
      .sum(sum[1]),
      .carry(c2)
      );
fa g3(.a(a[2]),
      .b(b[2]),
      .cin(c2),
      .sum(sum[2]),
      .carry(c3)
      );
fa g4(.a(a[3]),
      .b(b[3]),
      .cin(c3),
      .sum(sum[3]),
      .carry(c4)
      );
fa g5(.a(a[4]),
      .b(b[4]),
      .cin(c4),
      .sum(sum[4]),
      .carry(c5)
      );
fa g6(.a(a[5]),
      .b(b[5]),
      .cin(c5),
      .sum(sum[5]),
      .carry(c6)
      );
fa g7(.a(a[6]),
      .b(b[6]),
      .cin(c6),
      .sum(sum[6]),
      .carry(c7)
      );
fa g8(.a(a[7]),
      .b(b[7]),
      .cin(c7),
      .sum(sum[7]),
      .carry(cout)
      );
endmodule
```

OUTPUT:



SIMULATION - Behavioral Simulation - Functional - sim_1 - rca

Name	Value
a[7:0]	09
b[7:0]	05
cin	1
sum[7:0]	0f
cout	0
c1	1
c2	0
c3	0
c4	0
c5	0
c6	0
c7	0

Sim Time: 1000100 ps

RESULT:

Thus the simulation and synthesis of Logic Gates ,Adders and Subtractor using Xilinx ISE is verified successfully.