Experiment -01

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.

SETP: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.

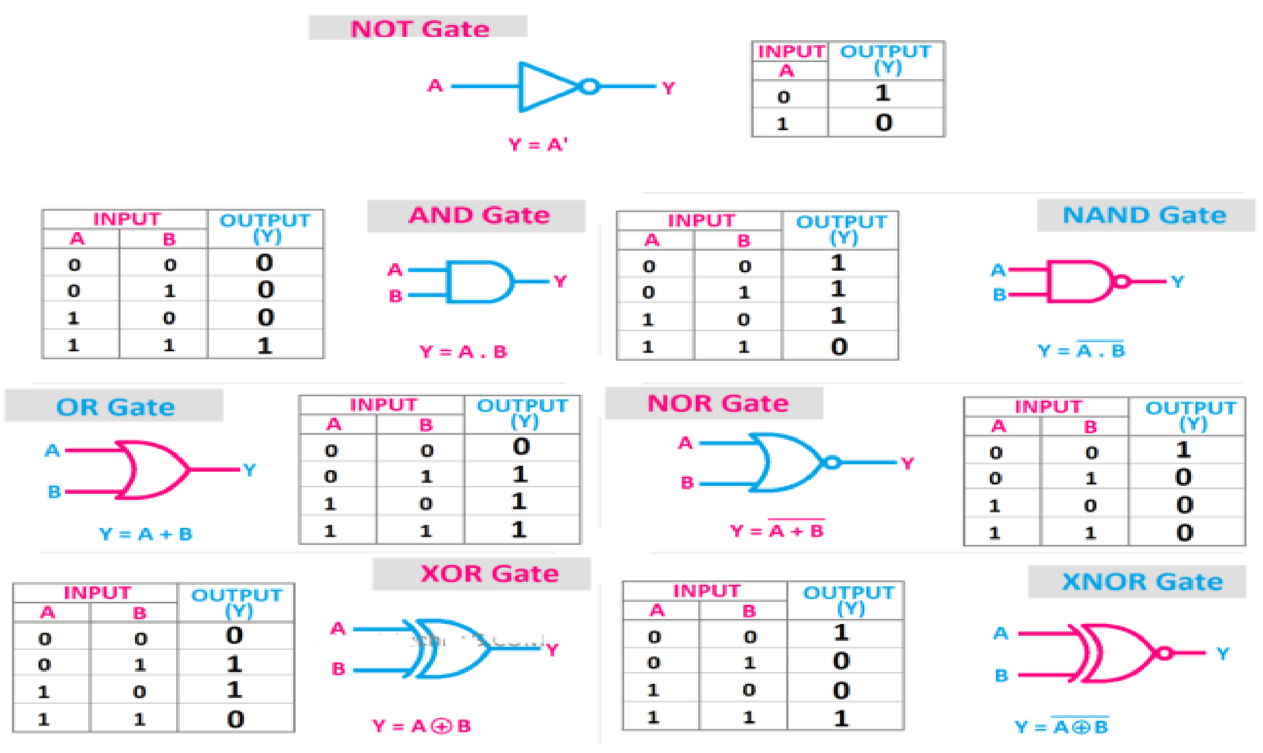
STEP:10 Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

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 GATES:

LOGIC DIAGRAM:



NOT Gate

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

NAND Gate

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

OR Gate

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

NOR Gate

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

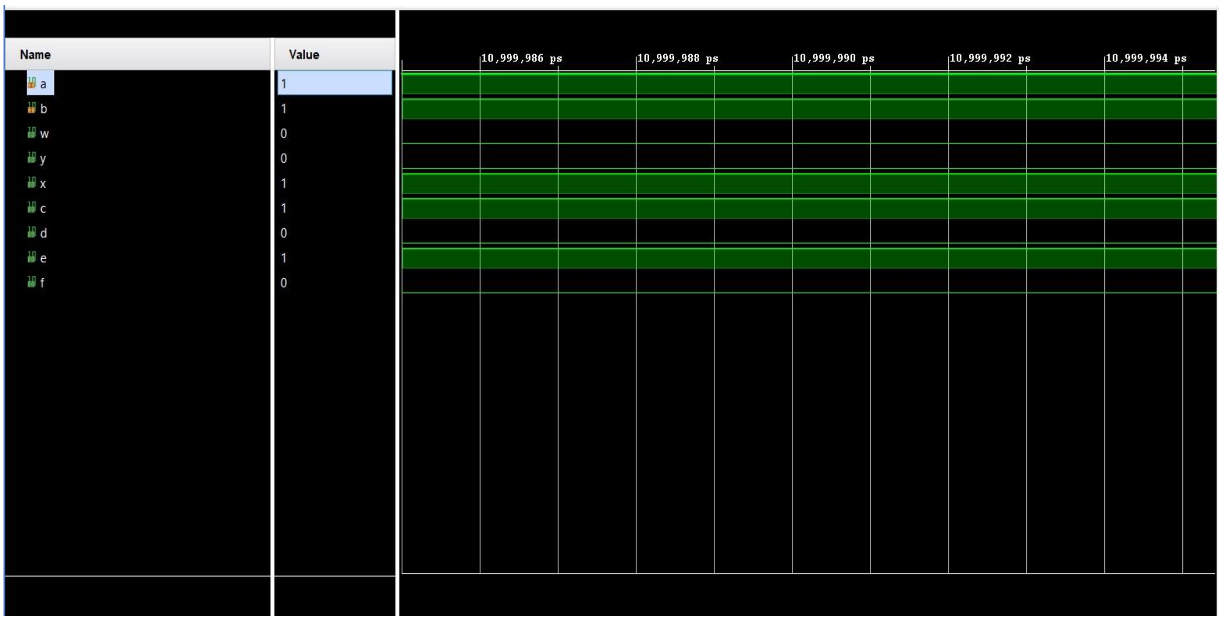
XNOR Gate

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

VERILOG CODE:

```
Module logicgates(a,b,andgate,orgate,xorgate,nandgate,norgate,xnorgate,notgate);
input a,b;
output andgate,orgate,xorgate,nandgate,norgate,xnorgate,notgate;
and(andgate,a,b);
or(orgate,a,b);
xor(xorgate,a,b);
nand(nandgate,a,b);
nor(norgate,a,b);
xnor(xnorgate,a,b);
not(notgate,a);
endmodule
```

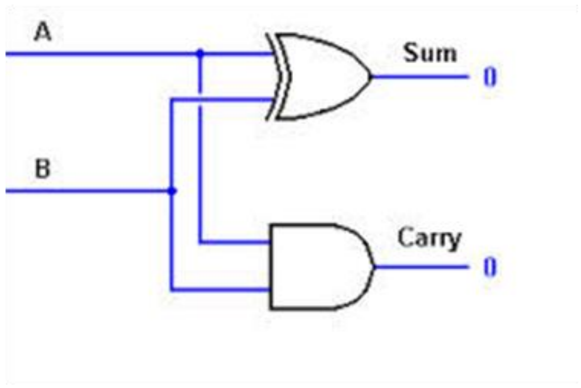
OUTPUT:



```
a 1
b 1
w 0
y 0
x 1
c 1
d 0
e 1
f 0
```

Half Adder:

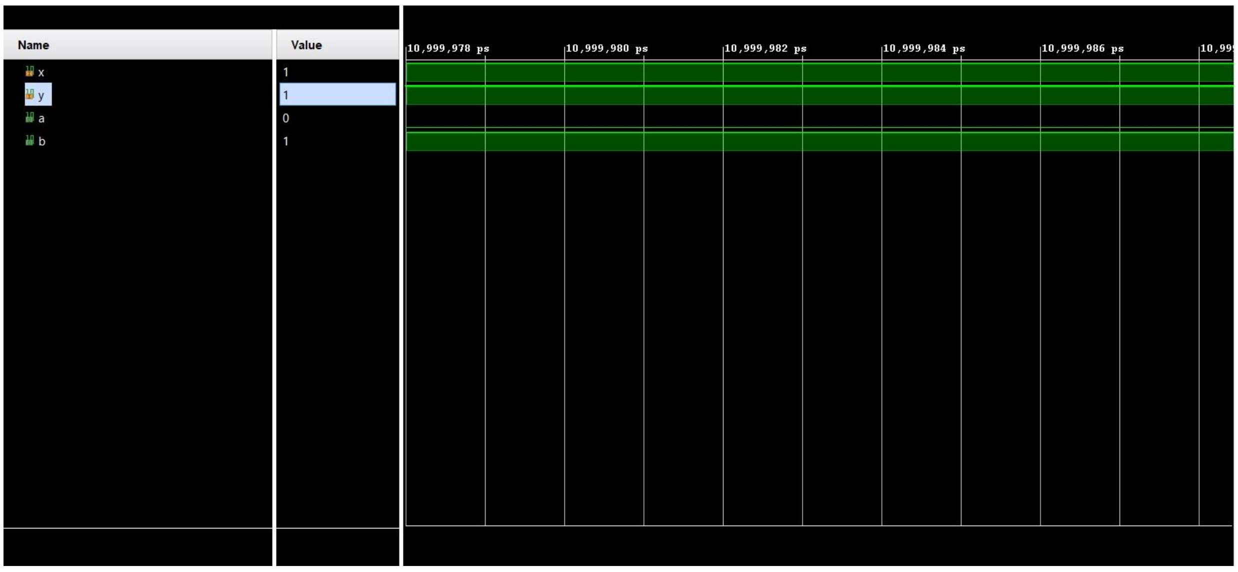
LOGIC DIAGRAM:



VERILOG CODE:

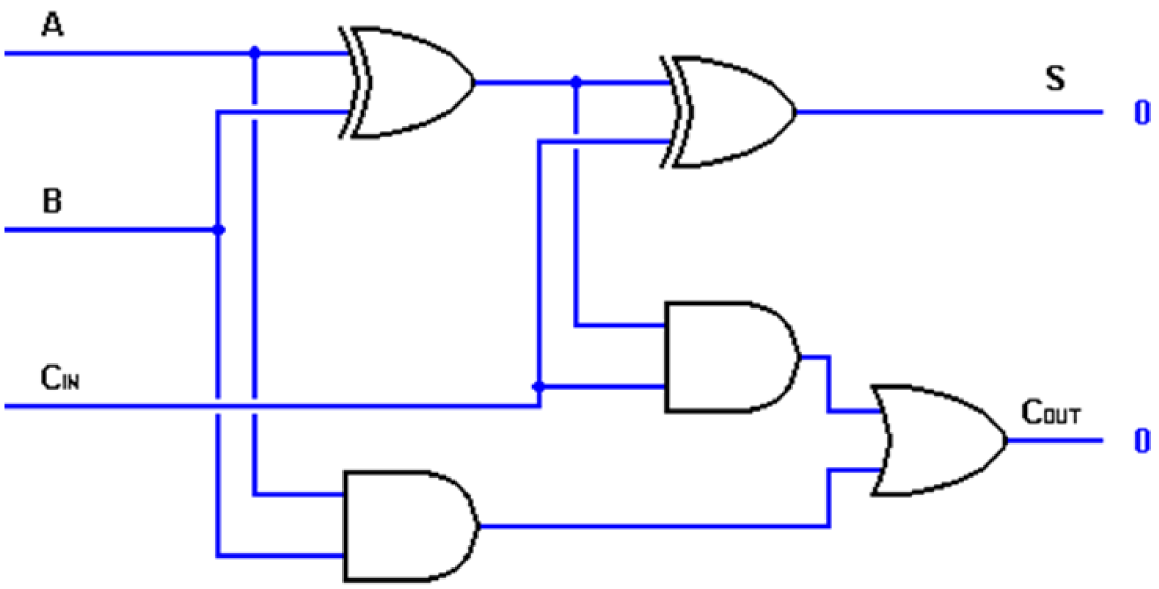
```
module ha(x,y,a,b);
input x,y;
output a,b;
xor x1(a,x,y);
and x2(b,x,y);
endmodule
```

OUTPUT:



```
Name    Value   10,999,978 ps   10,999,980 ps   10,999,982 ps   10,999,984 ps   10,999,986 ps   10,99
x       1
y       1
a       0
b       1
```

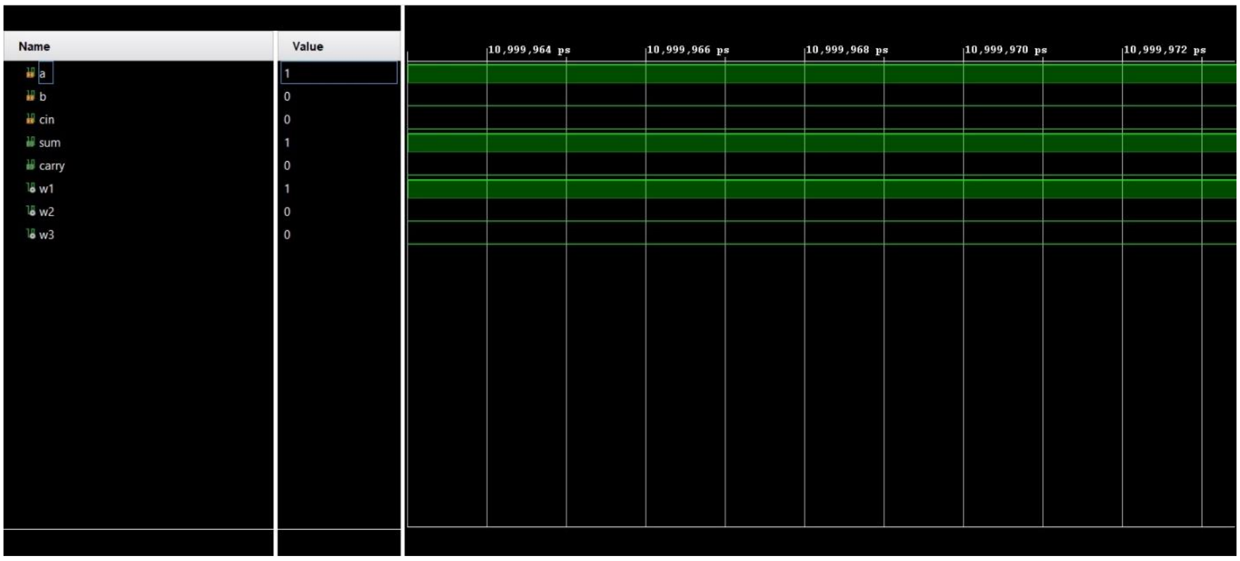
Full adder:
LOGIC DIAGRAM:



VERILOG CODE:

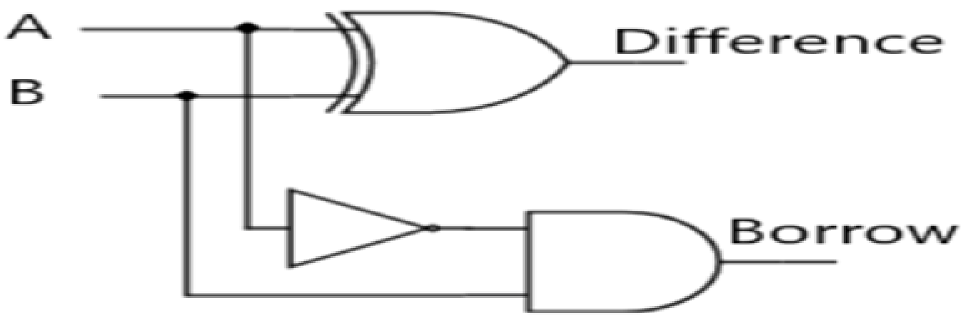
```
module fa(a,b,cin,sum,carry);
input a,b,cin;
output sum,carry;
wire w1,w2,w3;
xor x1(w1,a,b);
and x2(w2,a,b);
xor x3(sum,w1,cin);
and x4(w3,w1,cin);
or x5(carry,w3,w2);
endmodule
```

OUTPUT:



```
Name    Value
a       1
b       0
cin     0
sum     1
carry   0
w1      1
w2      0
w3      0
```

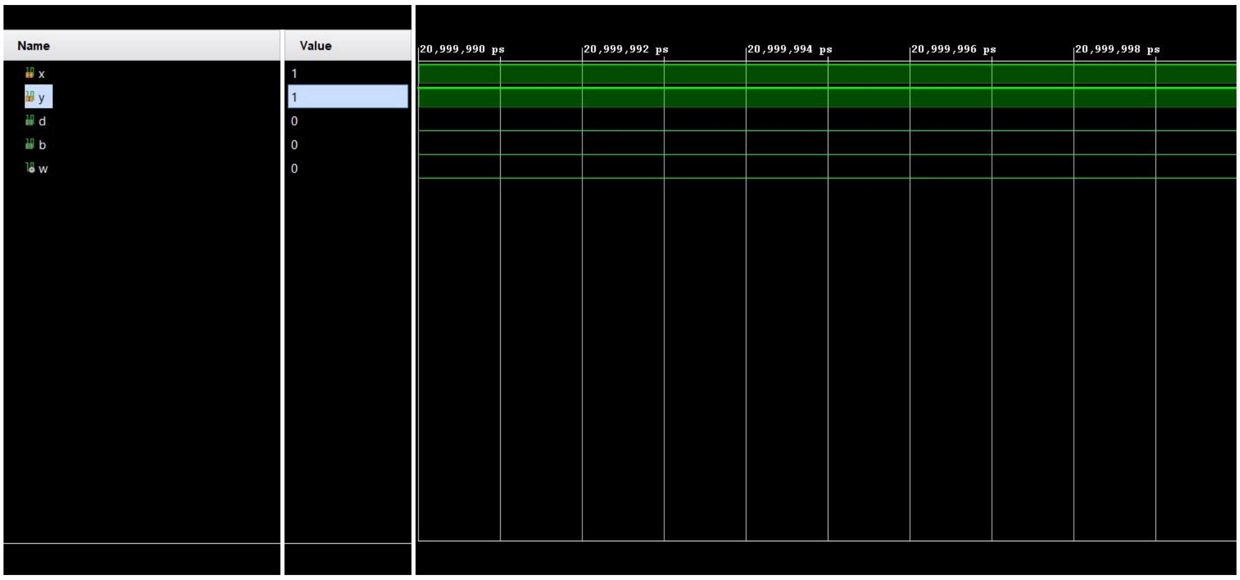
Half Subtractor:
LOGIC DIAGRAM:



VERILOG CODE:

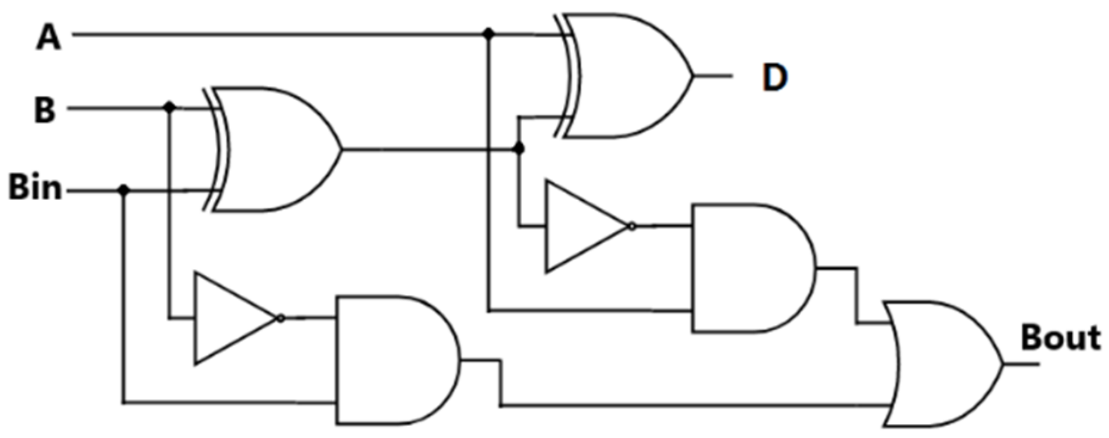
```
module hs(x,y,d,b);
input x,y;
output d,b;
wire w;
xor x1(d,x,y);
not n1(w,x);
and a1(b,y,w);
endmodule
```

OUTPUT:



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

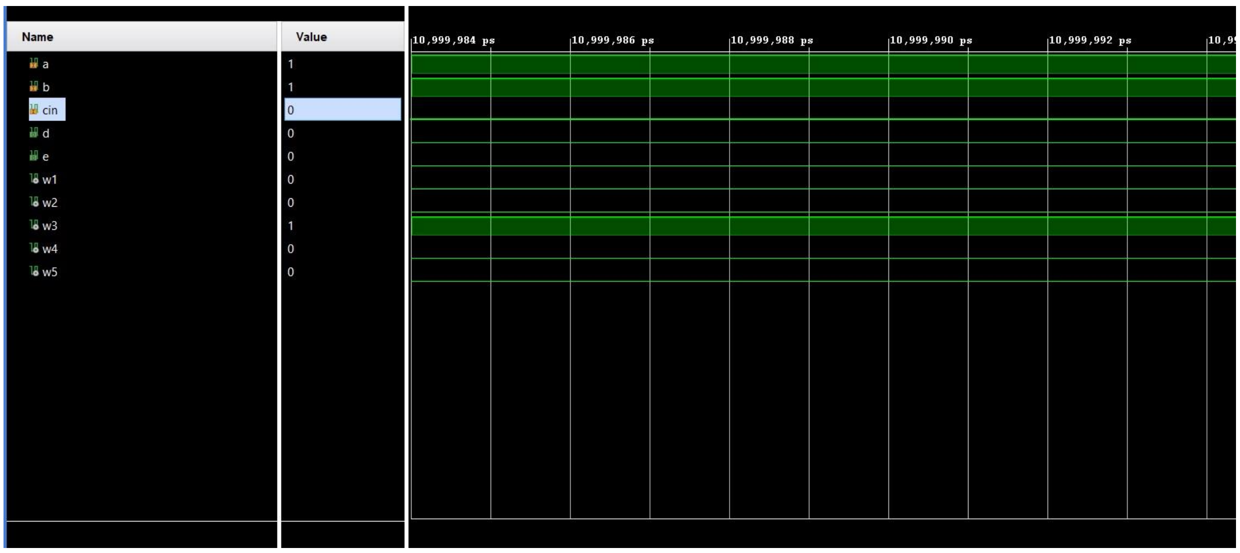
Full Subtractor:
LOGIC DIAGRAM:



VERILOG CODE:

```
module fs(a,b,cin,d,e);
input a,b,cin;
output d,e;
wire w1,w2,w3,w4,w5;
xor x1(w2,a,b);
not n1(w1,a);
and a1(w4,w1,b);
xor x2(d,w2,cin);
not n2(w3,w2);
and a2(w5,w3,cin);
or o1(e,w5,w4);
endmodule
```

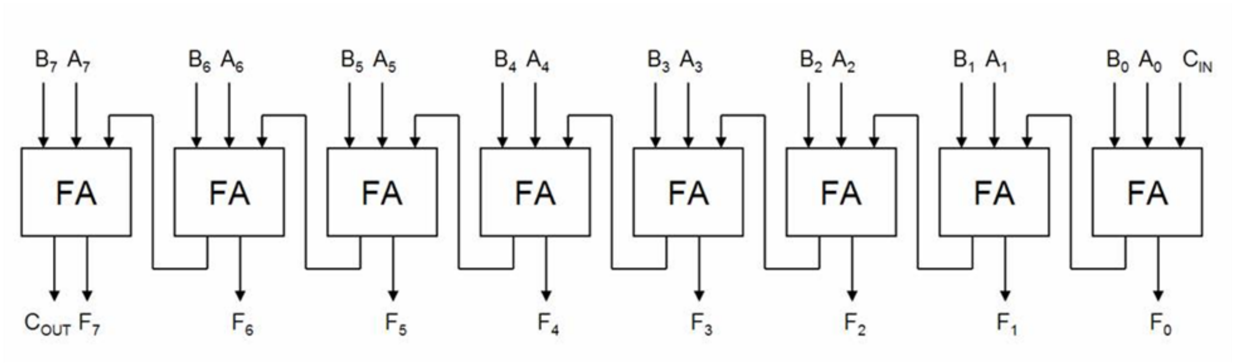
OUTPUT:



```
Name   Value
a      1
b      1
cin    0
d      0
e      0
w1     0
w2     0
w3     1
w4     0
w5     0
```

8 Bit Ripple Carry Adder

LOGIC DIAGRAM:

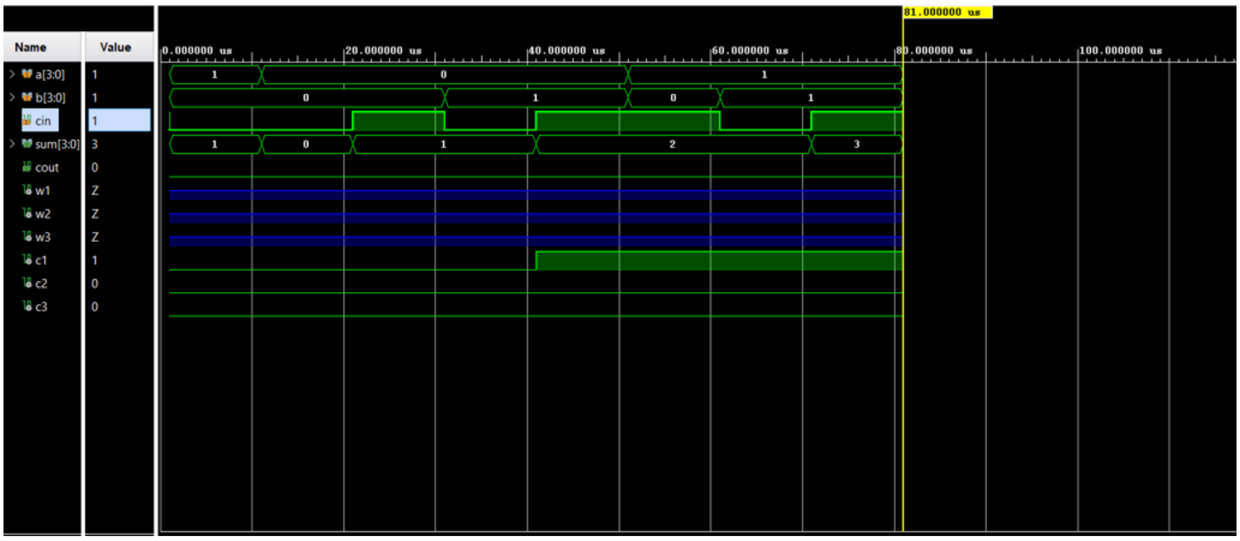


VERILOG CODE:

```
module fa(a,b,cin,sum,carry);
input a,b,cin;
output sum,carry;
wire w1,w2,w3;
xor x1(w1,a,b);
and x2(w2,a,b);
xor x3(sum,w1,cin);
and x4(w3,w1,cin);
or x5(carry,w3,w2);
endmodule

module rca(a,b,cin,sum,carry);
input [3:0]a,b;
input cin;
output[3:0]sum;
output carry;
wire w1,w2,w3;
fa g1(.a(a[0]),
      .b(b[0]),
      .cin(cin),
      .s(sum[0]),
      .carry(w1)
      );
fa g2(.a(a[1]),
      .b(b[1]),
      .cin(w1),
      .s(sum[1]),
      .carry(w2)
      );
 fa g3(.a(a[2]),
      .b(b[2]),
      .cin(w2),
      .s(sum[2]),
      .carry(w3)
      );
 fa g4(.a(a[3]),
      .b(b[3]),
      .cin(w3),
      .s(sum[3]),
      .carry(carry)
      );
  endmodule
```

OUTPUT:



RESULT:

Thus ,the given logic gates,half adder,full adder,half subtractor are simulated and synthesis are excuted successfully .