AIM:

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

APPARATUS REQUIRED:

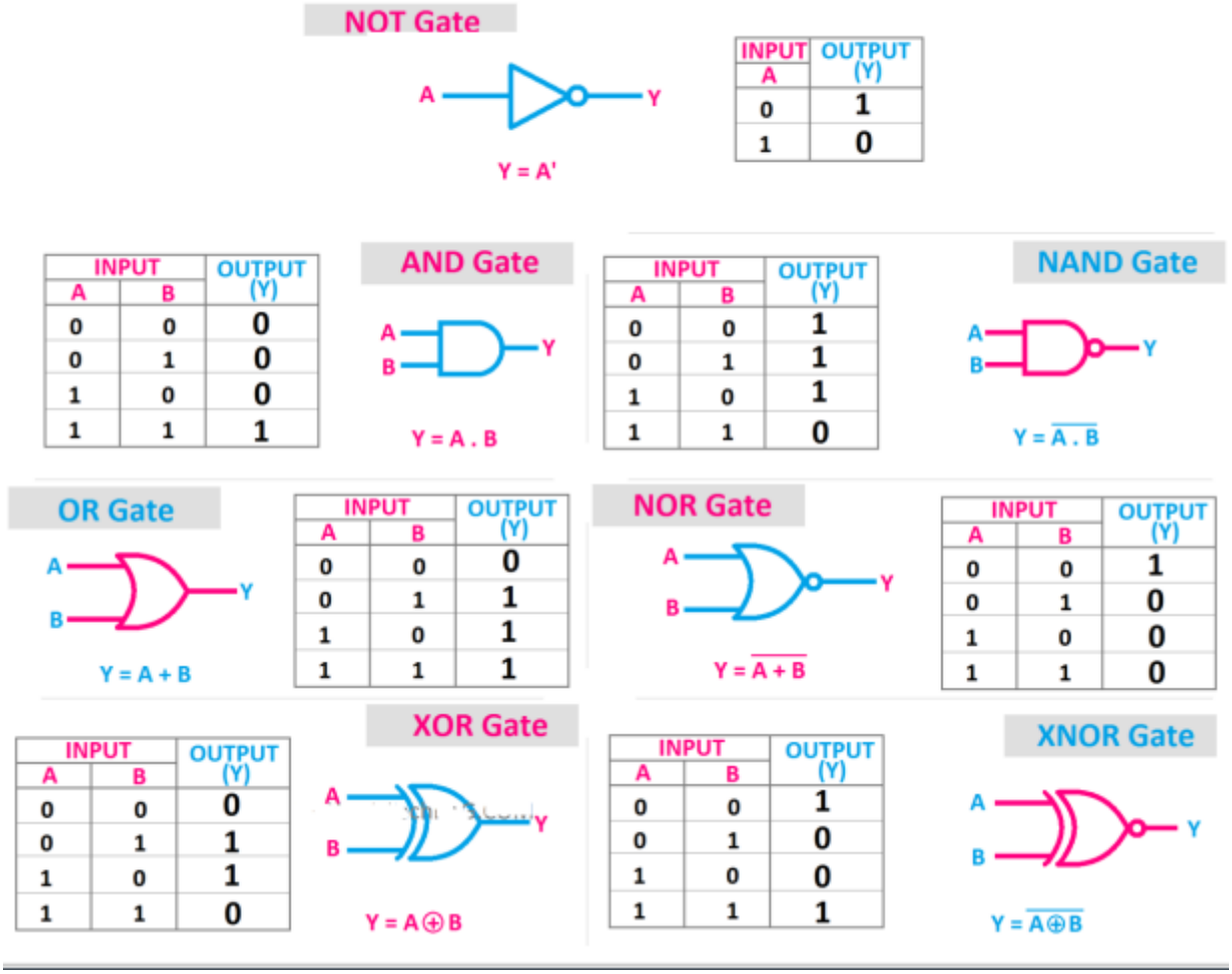
Xilinx 14.7 Spartan6 FPGA

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 Gates:

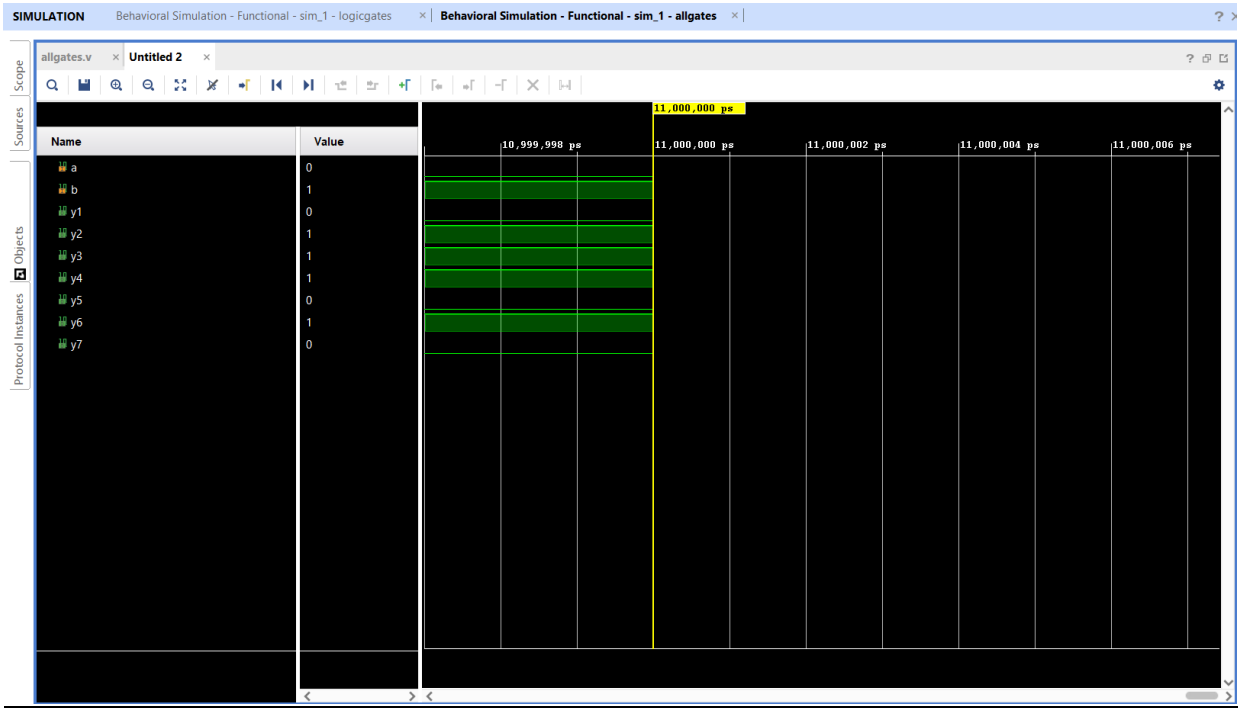
Logic Diagram:



Program:

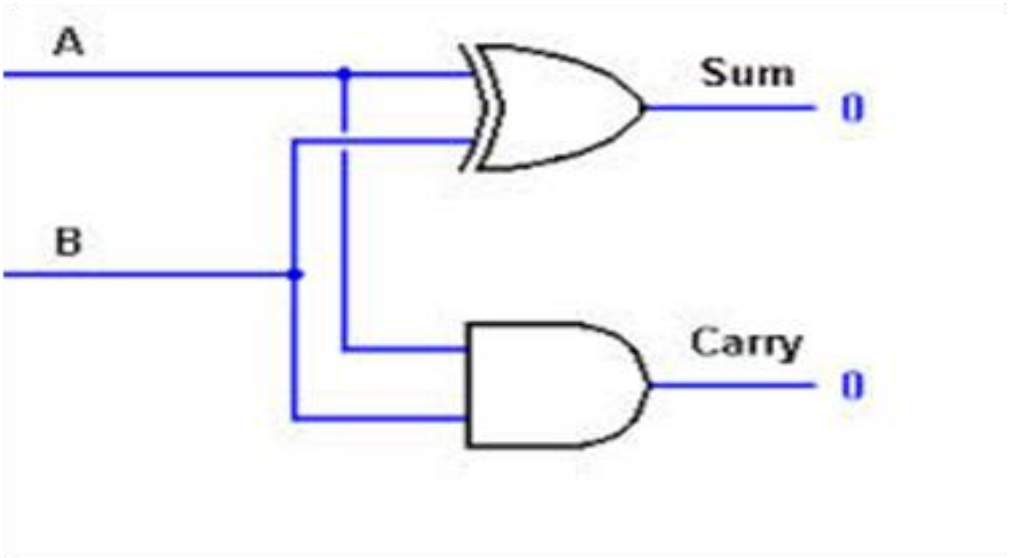
```
module allgates(a,b,y1,y2,y3,y4,y5,y6,y7);
input a,b;
output y1,y2,y3,y4,y5,y6,y7;
and g1(y1,a,b);
or  g2(y2,a,b);
not g3(y3,a);
xor g4(y4,a,b);
xnor g5(y5,a,b);
nand g6(y6,a,b);
nor g7(y7,a,b);
endmodule
```

Output:



Half Adder

Logic Diagram:

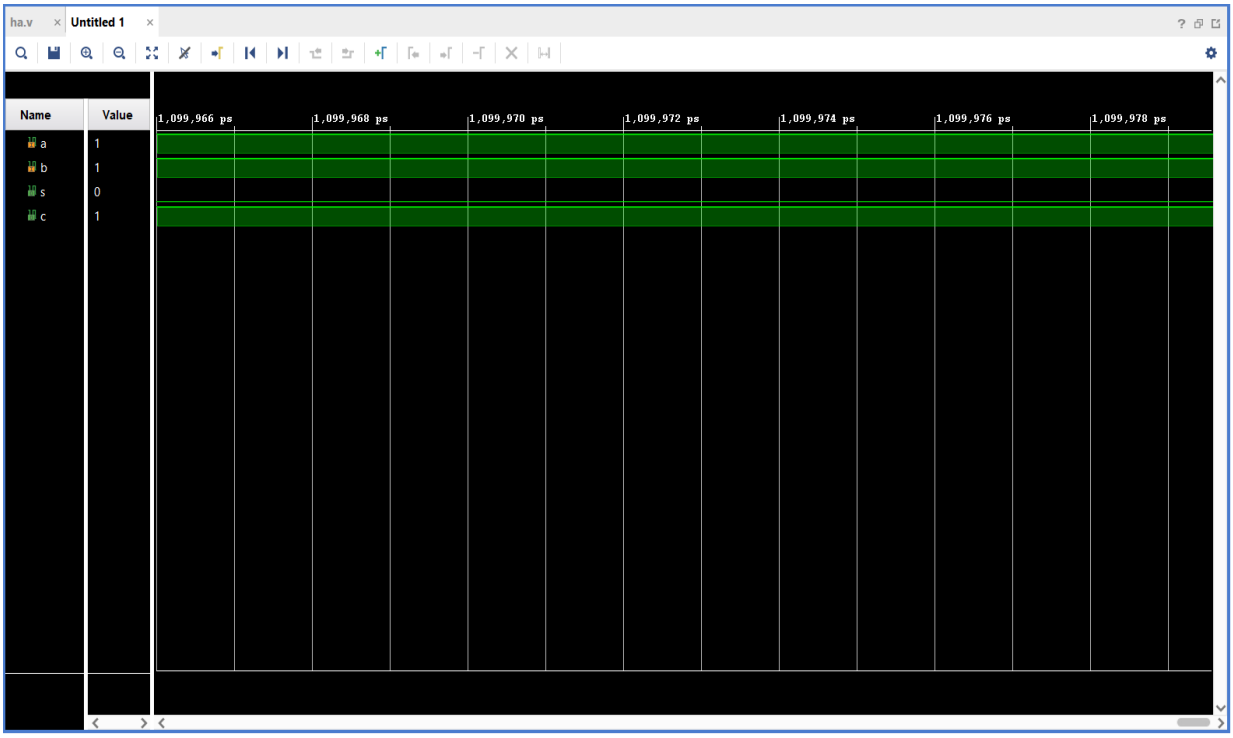


VERILOG CODE:

```
module halfadder(a,b,s,c);
input a,b;
```

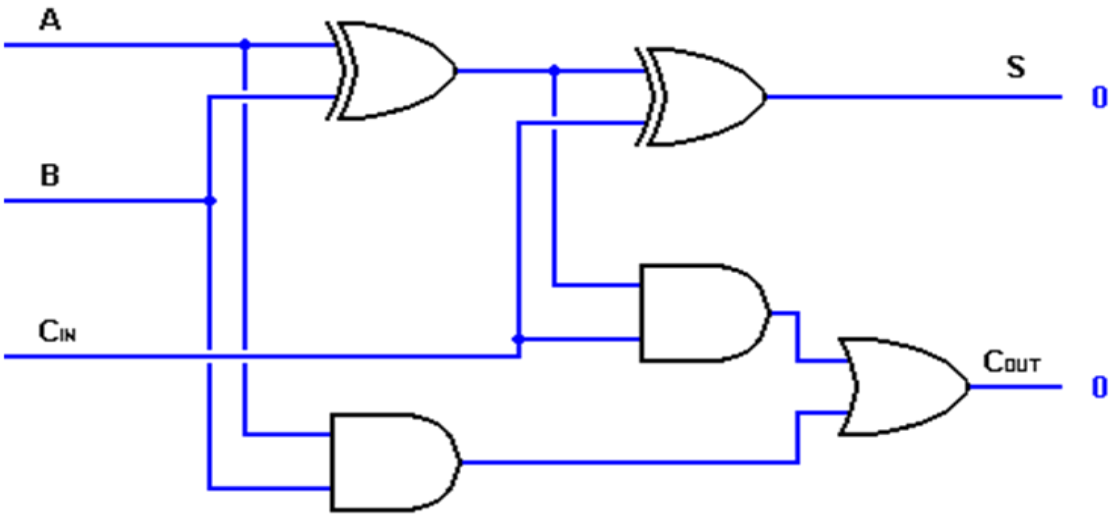
output s,c;
xor (s,a,b);
and (c,a,b);
endmodule

Output:



FULL ADDER

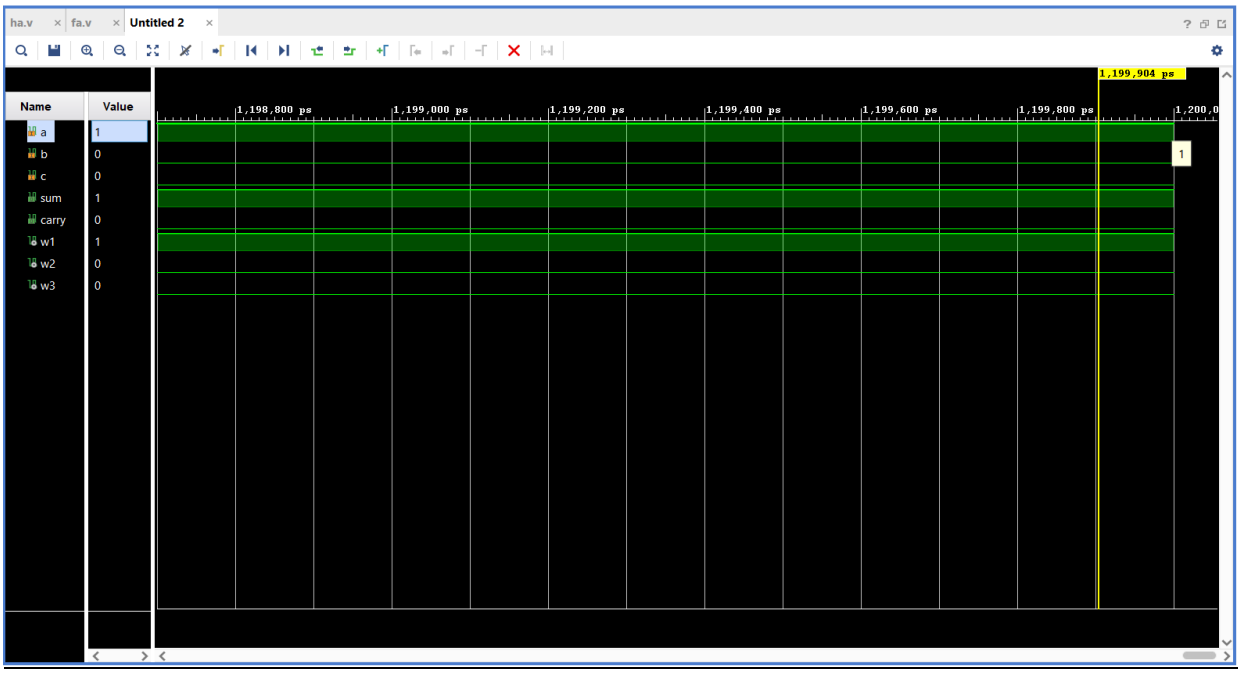
Logic Diagram:



VERILOG CODE:

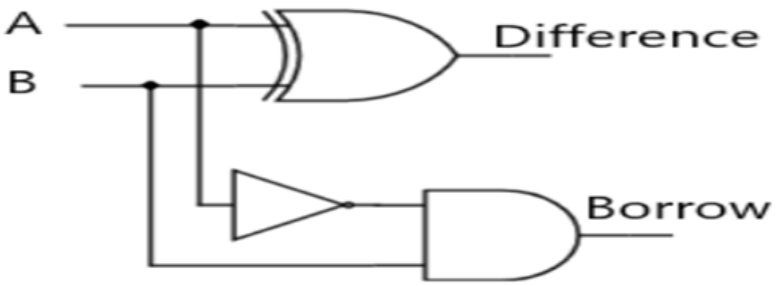
```
module fulladder(a,b,c,sum,carry);
input a,b,c;
output sum,carry;
wire w1,w2,w3;
xor g1(w1,a,b);
xor g2(sum,w1,c);
and g3(w2,a,b);
and g4(w3,w1,c);
or g5(carry,w3,w2);
endmodule
```

Output:



Half Subtractor

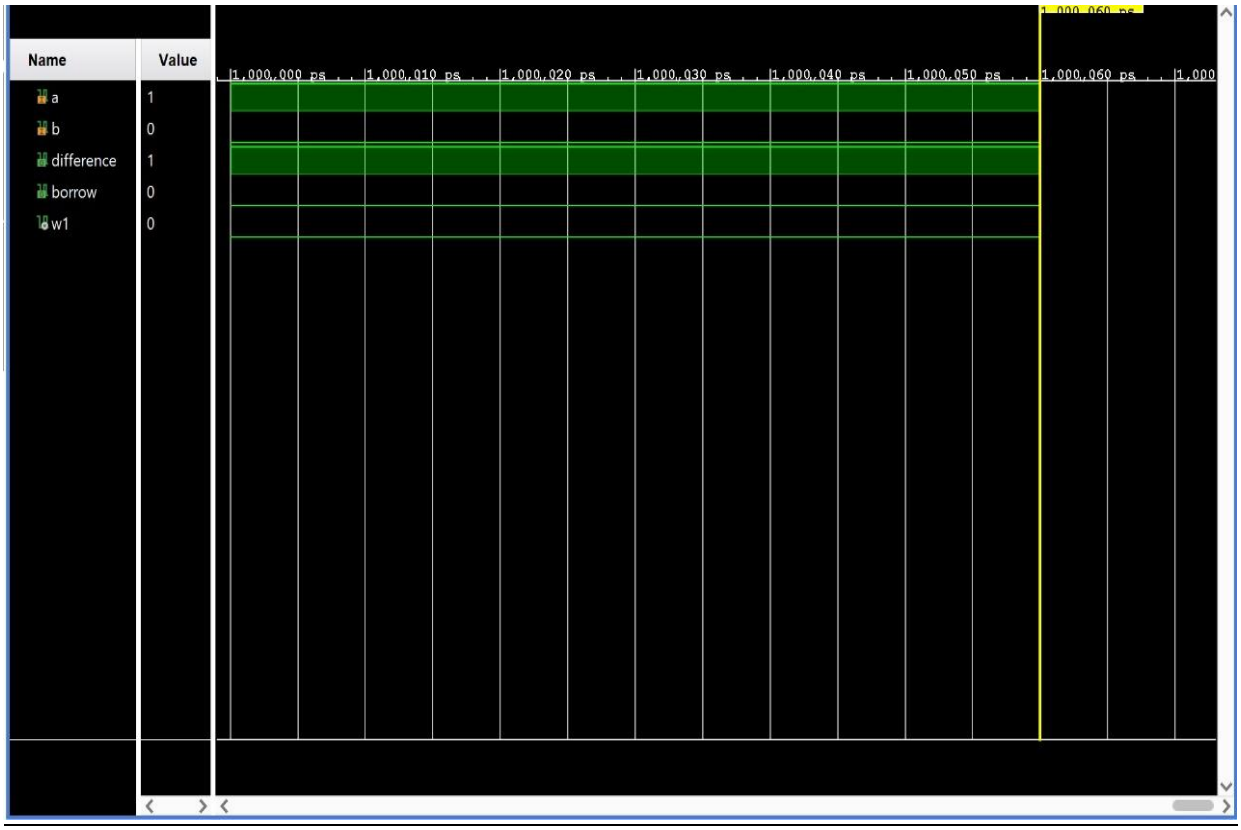
Logic Diagram:



VERILOG CODE:

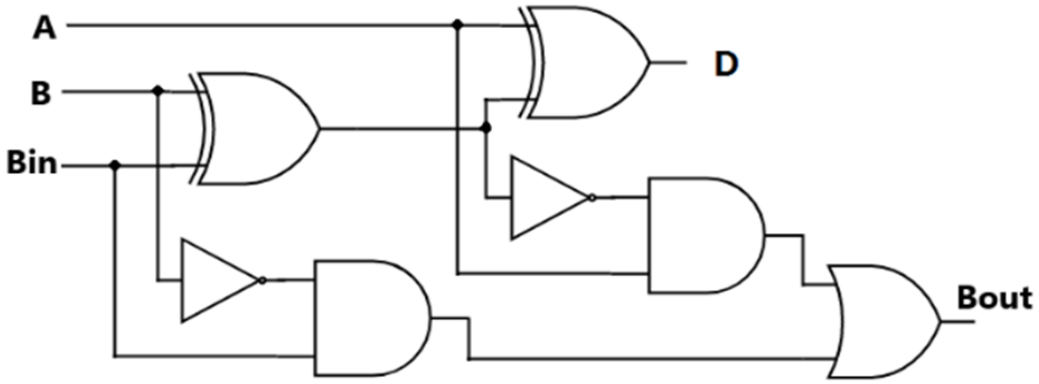
```
module hasub(a,b,difference,borrow);
input a,b;
output difference,borrow;
wire w1;
xor g1(difference,a,b);
not g2(w1,a);
and g3(borrow,w1,b);
endmodule
```

Output:



Full Subtractor:

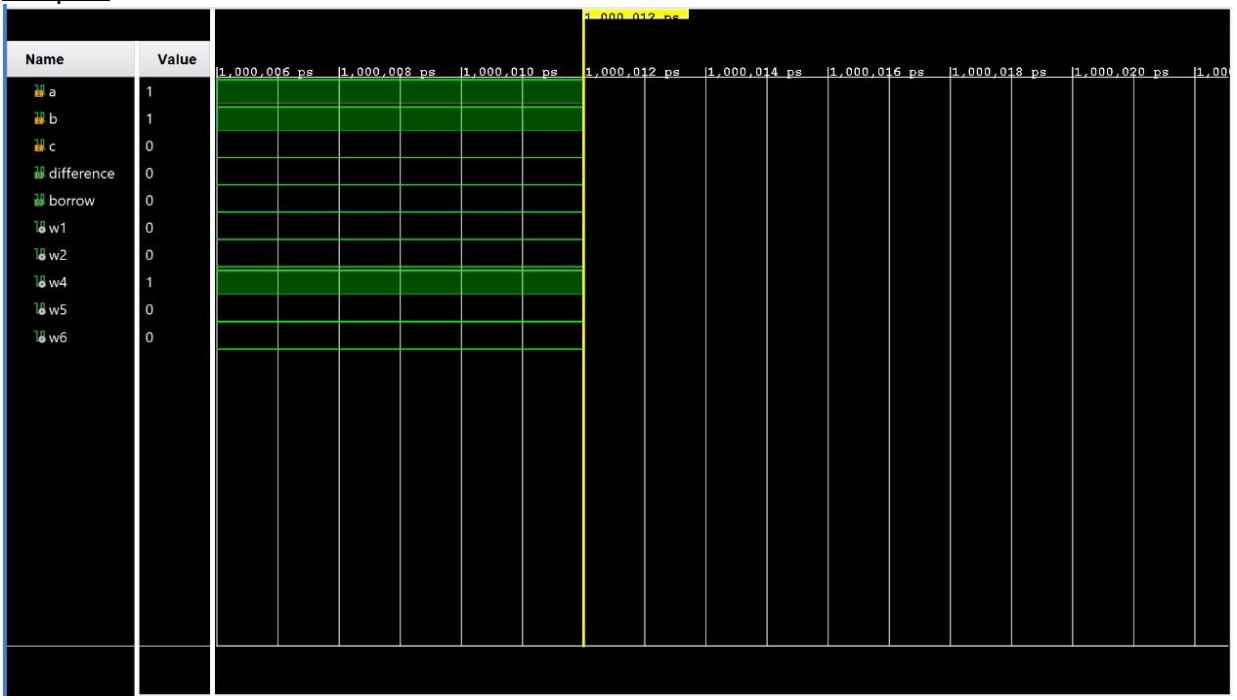
Logic Diagram:



VERILOG CODE:

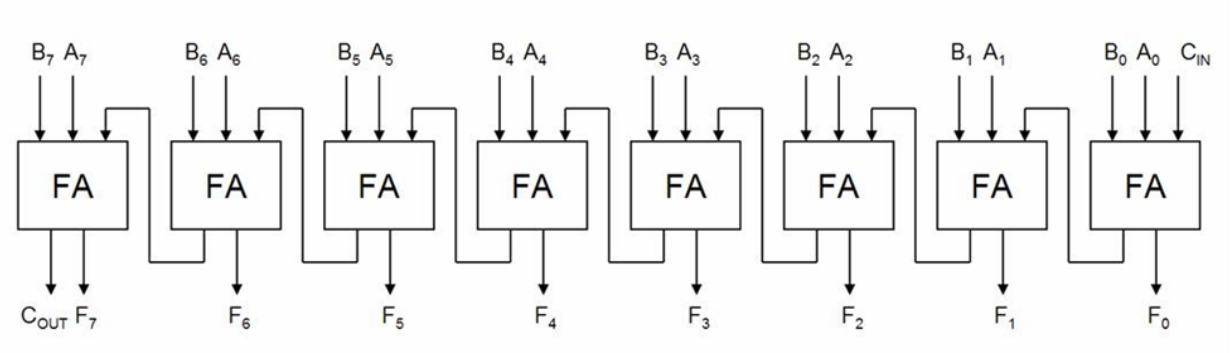
```
module fullsub(a,b,c,difference,borrow);
input a,b,c;
output difference,borrow;
wire w1,w2,w4,w5,w6;
xor x1(w2,a,b);
xor x2(difference,w2,c);
not n1(w1,a);
and a1(w5,w1,b);
not n2(w4,w2);
and a2(w6,w4,c);
or o1(borrow,w5,w6);
endmodule
```

Output:



8 Bit Ripple Carry Adder

Logic Diagram:

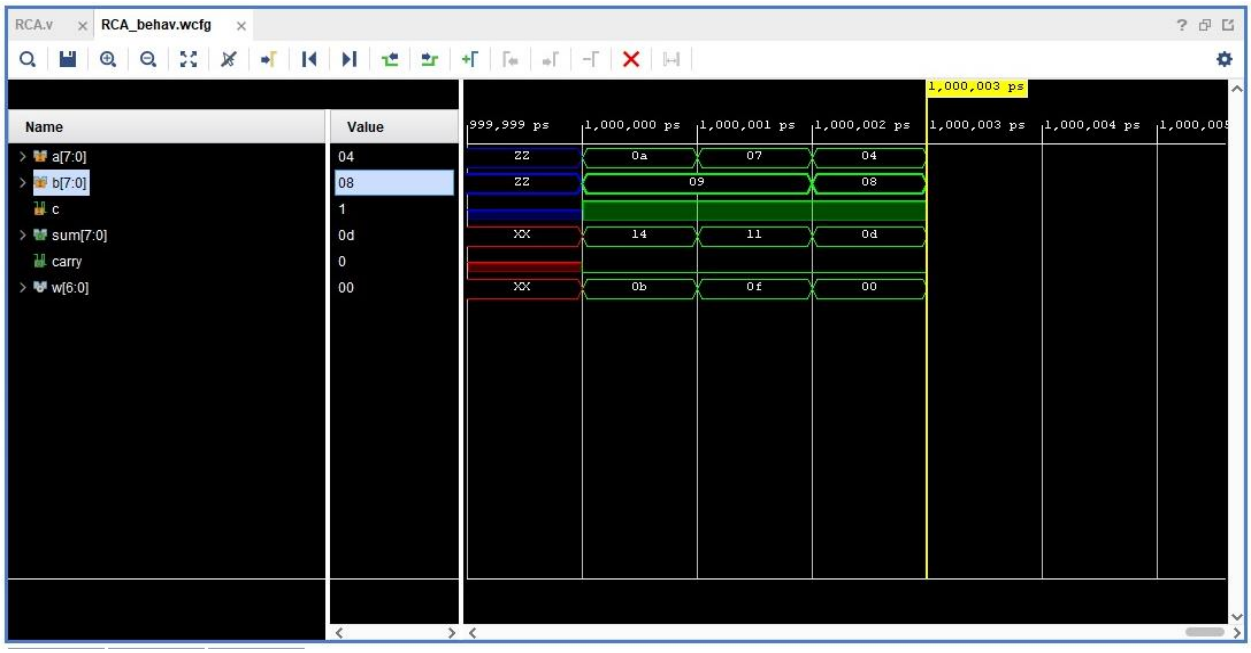


VERILOG CODE:

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

module ripplecarry(a,b,cin,sum,carry);
input [3:0]a,b;
input cin;
output [3:0]sum,carry;
wire c1,c2,c3;
rca g1(.a(a[0]),.b(b[0]),.cin(cin),.s(sum[0]),.carry(c1));
rca g2(.a(a[1]),.b(b[1]),.cin(c1),.s(sum[1]),.carry(c2));
rca g3(.a(a[2]),.b(b[2]),.cin(c2),.s(sum[2]),.carry(c3));
rca g4(.a(a[3]),.b(b[3]),.cin(c3),.s(sum[3]),.carry(cout));
endmodule
```

OUTPUT:



RESULT:

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