AIM:

To simulate and synthesis All gates 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

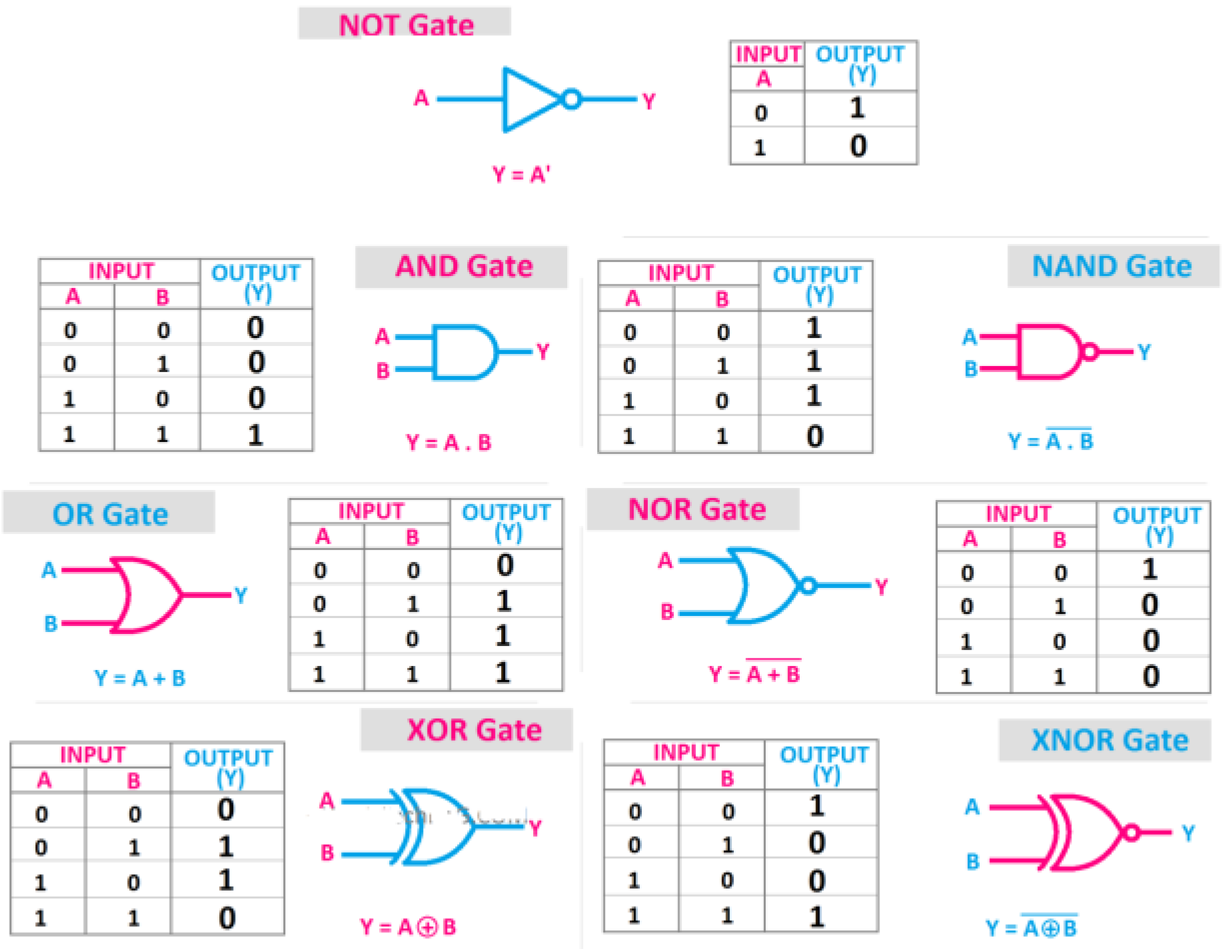
On the board, by giving required input, the LED start to glow light ,indicating the output.

STEP:12

Load the Bit file into the SPARTAN 6 FPGA

Logic Diagram :

Logic Gates:



NOT Gate

INPUT A	OUTPUT (Y)
0	1
1	0

AND Gate

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

NAND Gate

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

OR Gate

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

NOR Gate

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

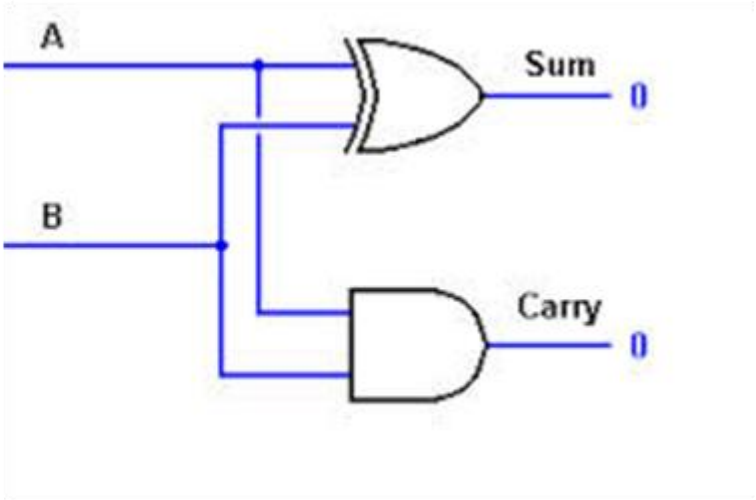
XOR Gate

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

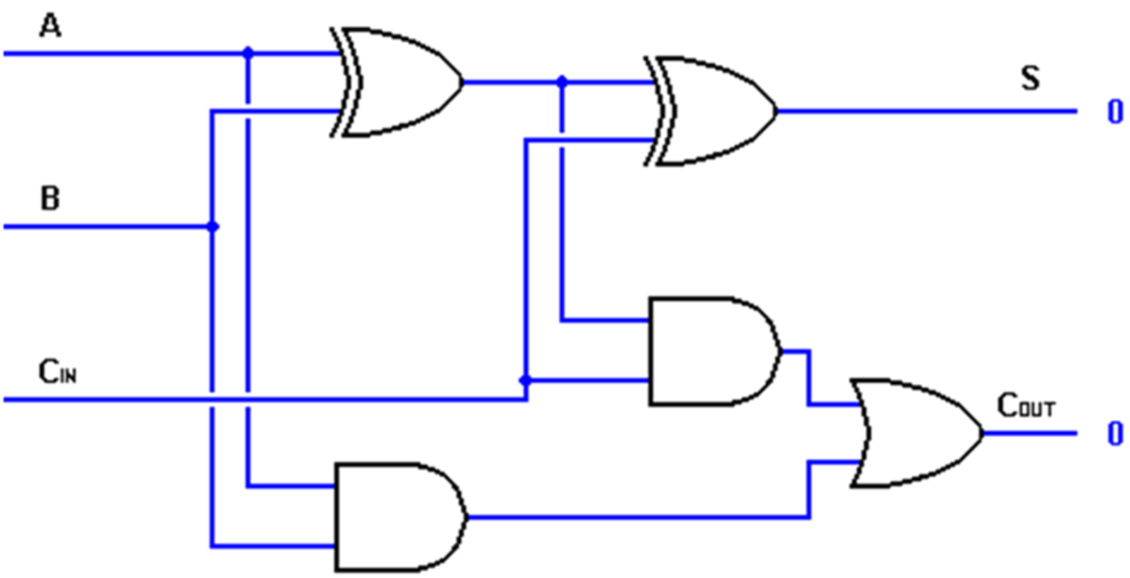
XNOR Gate

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

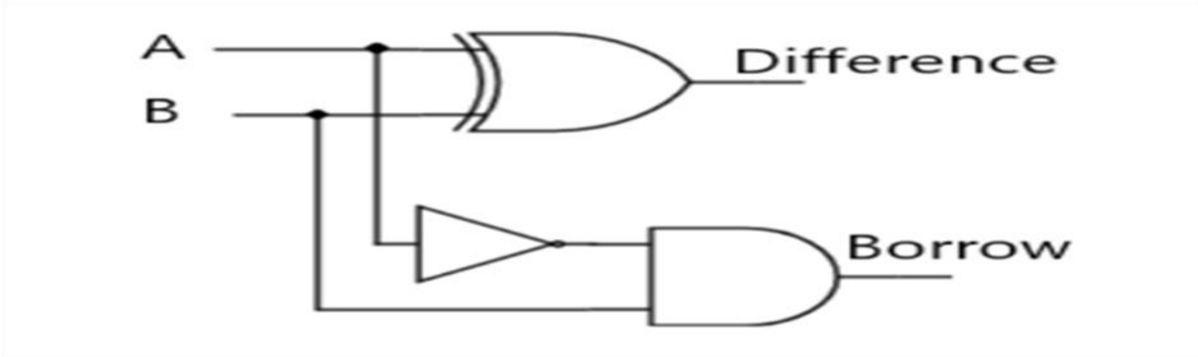
Half Adder:



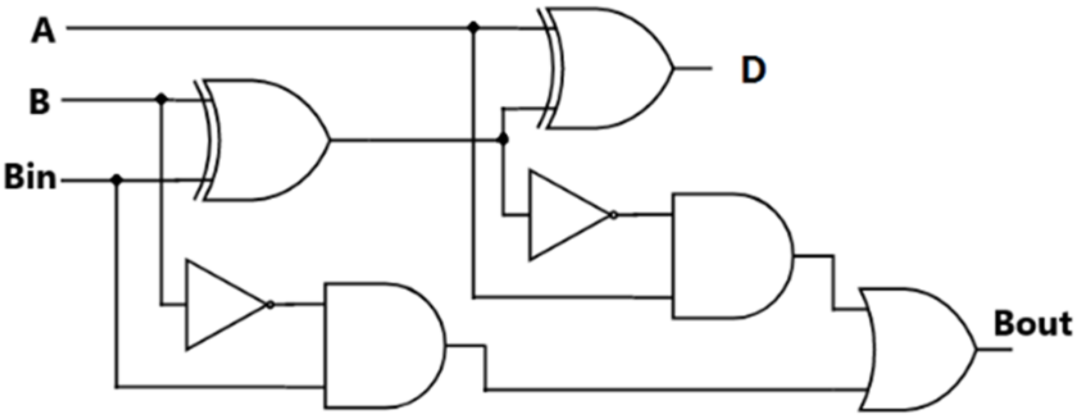
Full Adder:



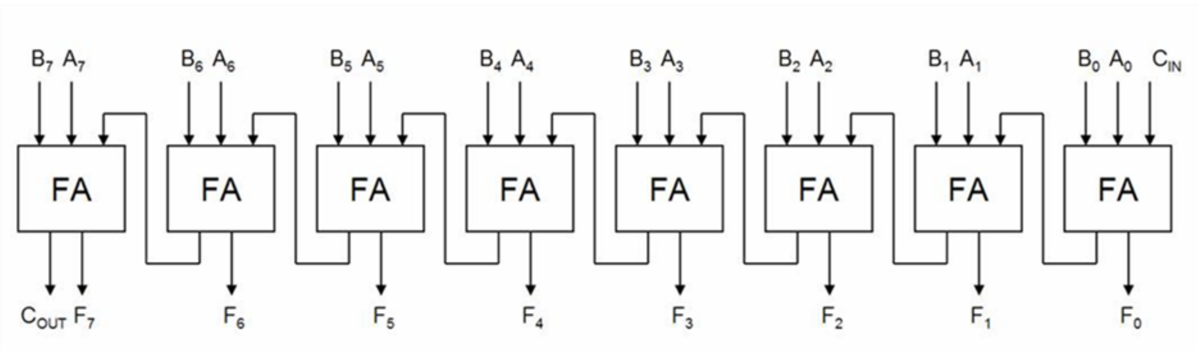
Half Subtractor:



Full Subtractor:



8 Bit Ripple Carry Adder:



VERILOG CODE:

Logic Gates:

```
module gate(a,b,w1,w2,w3,w4,w5,w6,w7);
    input a,b;
    output w1,w2,w3,w4,w5,w6,w7;
    and g1(w1,a,b);
    or g2(w2,a,b);
    not g3(w3,a);
    xor g4(w4,a,b);
    xnor g5(w5,a,b);
 nand g6(w6,a,b);
 nor g7(w7,a,b);
endmodule
```

Half Adder:

```
module ha( a,b,sum,carry);
input a,b;
output sum,carry;
xor g1(sum,a,b);
and g2(carry,a,b);
endmodule
```

Full Adder:

```
module fa(a,b,cin,sum,carry);
    input a,b,cin;
    output sum,carry;
    wire w1,w2,w3;
    xor g1(w1,a,b);
```

```
    xor g2(sum,w1,cin);

    and g3(w2,w1,cin);

    and  g4(w3,a,b);

   or g5(carry,w2,w3);

  endmodule
```

Half Subtractor:

```
module hs(a,b,difference,borrow);

      input a,b;

     output difference,borrow;

     wire w;

     xor g1(difference,a,b);

     and g2(borrow,w,b);

     not g3(w,a);

endmodule
```

Full Subtractor:

```
module fs(a,b,bin,diff,borrow);

   input a,b,bin;

   output diff,borrow;

   wire w1,w2,w3,w4,w5;

   xor g1(w3,a,b);

   xor g2(diff,bin,w3);

   and g3(w2,b,w1);

   and g4(w5,w4,bin);

   not g5(w1,a);
```

```
    not g6(w4,w3);

    nor g7(borrow,w5,w2);

endmodule
```

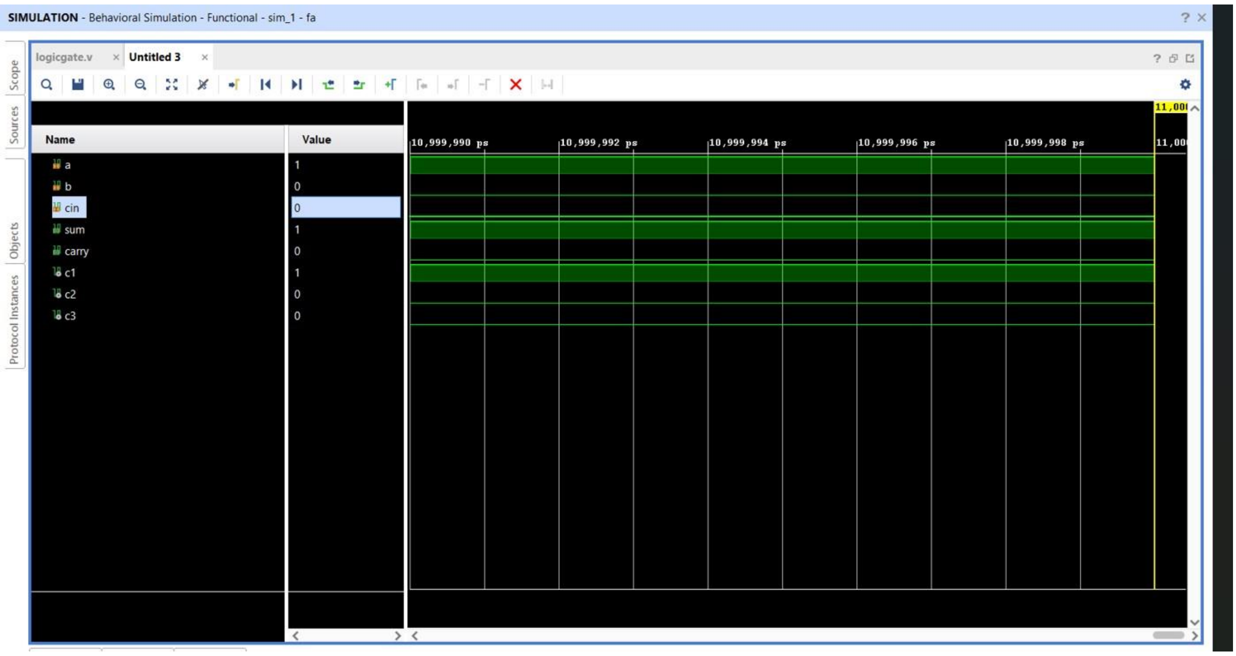
8 Bit Ripple Carry Adder:

```
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[3:0]a,b;

        input cin;

       output [3:0]sum;

        output cout;

        wire w1,w2,w3;

      fa g1(.a(a[0]),

            .b(b[0]),

              .cin(cin),
```

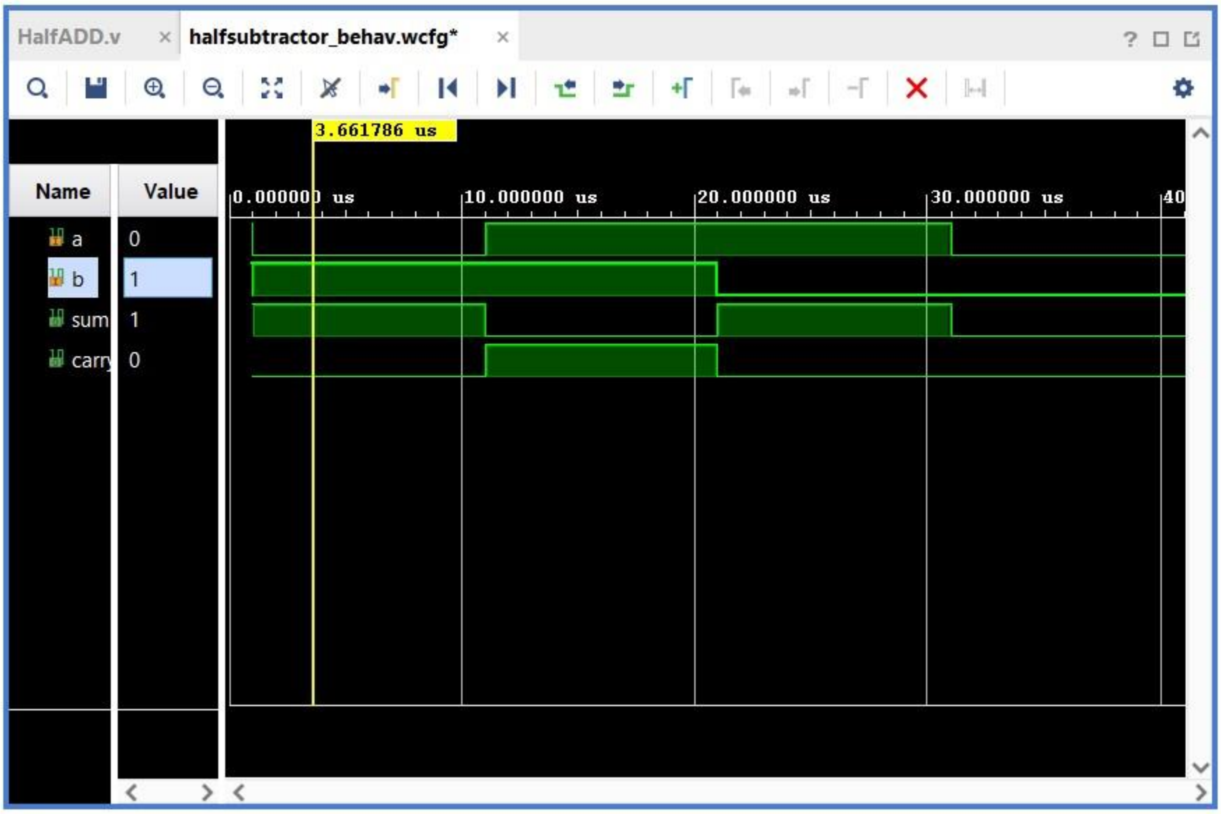
```
        .sum(sum[0]),
        .carry(c1)
        );
    fa g2(.a(a[1]),
        .b(b[1]),
        .cin(c1),
        .sum(sum[1]),
        .carry(c2)
        );
    fa g3(.a(a[2]),
        .b(b[2]),
        .cin(c2),
        .sum(sum[2]),
        .carry(c3)
        );
    fa g4(.a(a[3]),
        .b(b[3]),
        .cin(c3),
        .sum(sum[3]),
        .carry(cout)
        );
endmodule
```

OUTPUT:

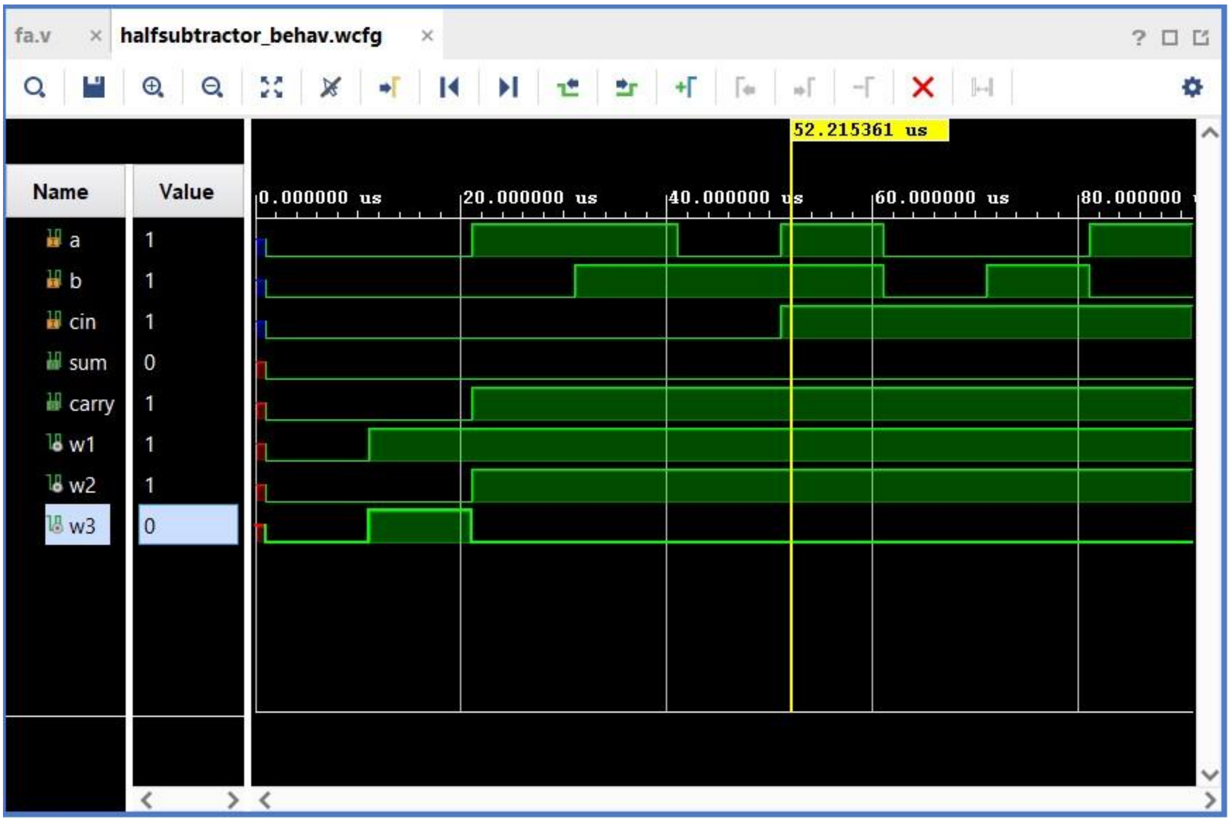
Logic Gates:



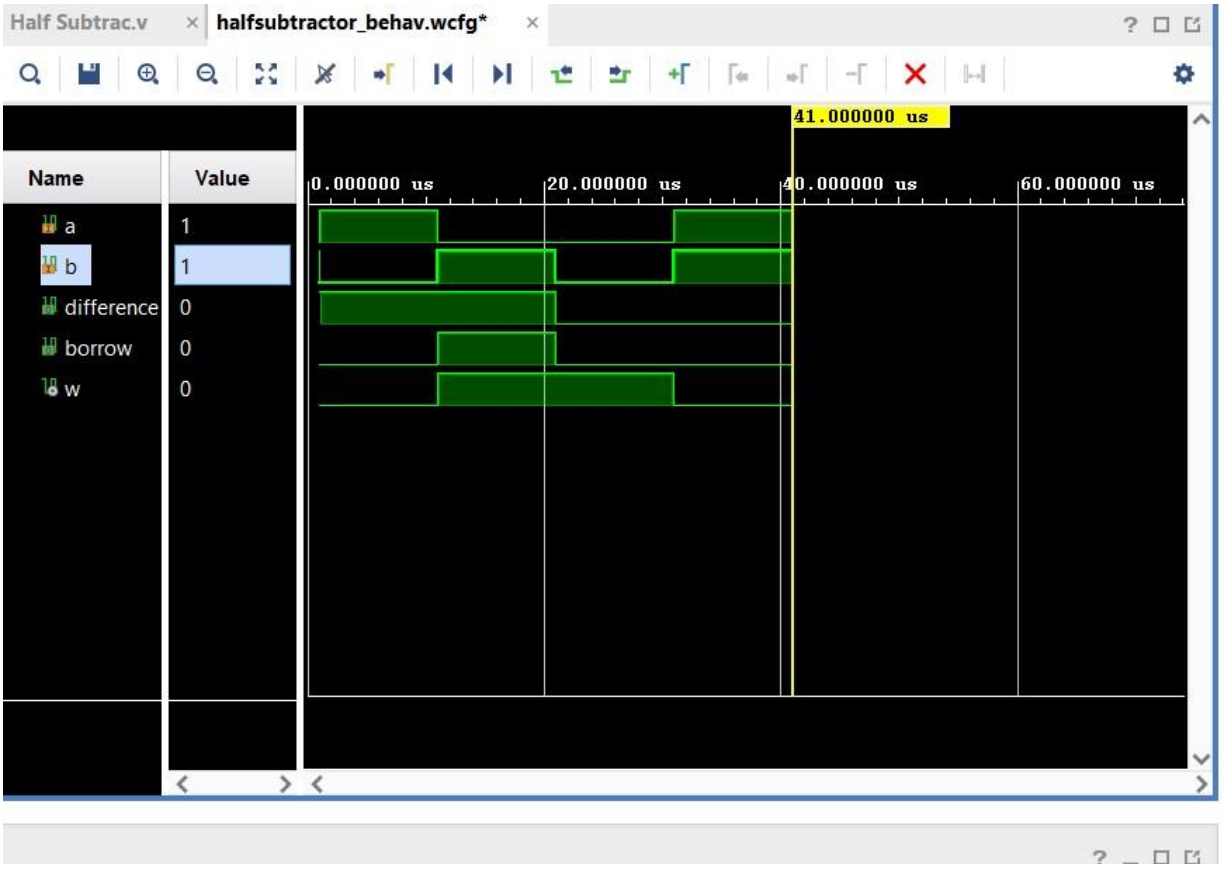
Half Adder:



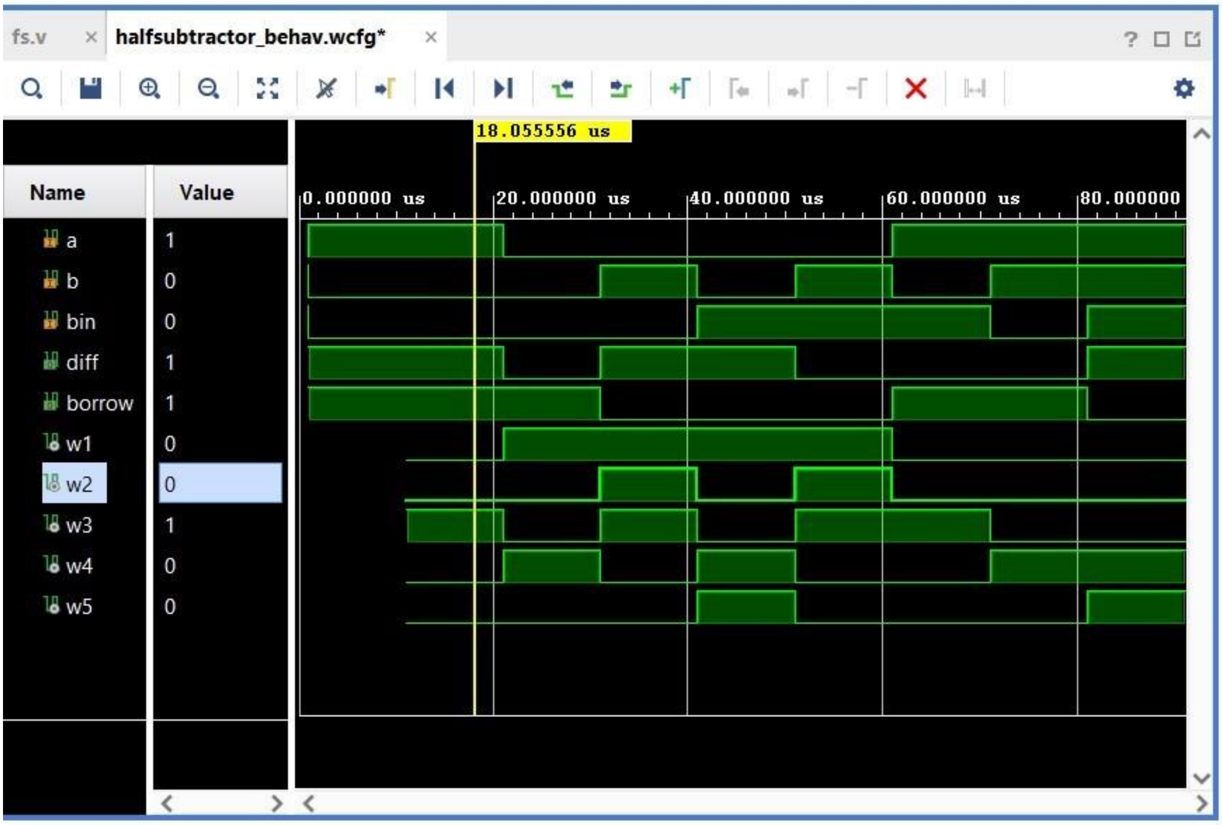
Full adder:



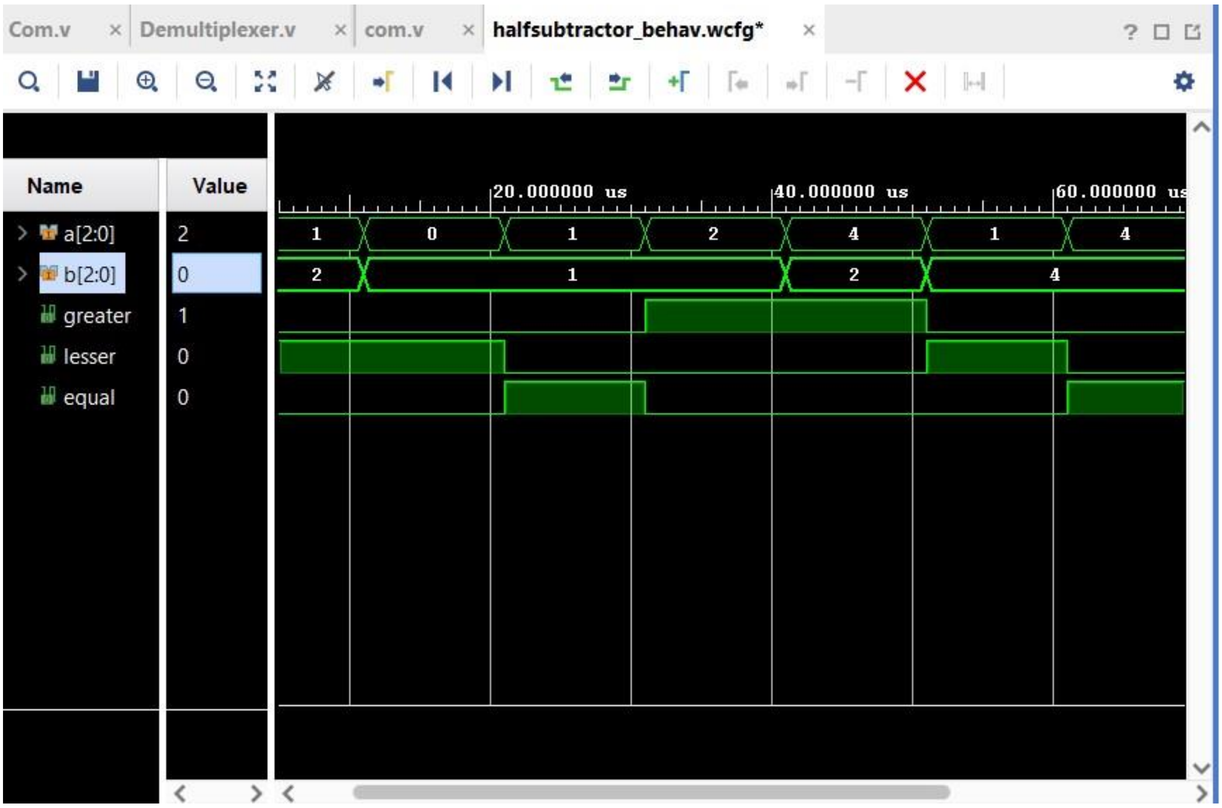
Half Subtractor:



Full Subtractor:



8 Bit Ripple Carry Adder:



RESULT:

Hence, The simulation and synthesis Logic Gates, Adders, Subtractor, Ripple Carry Adder was running successfully using Xilinx ISE.