

## **EX.NO: 1 DESIGN AND SIMULATION OF HALF AND FULL ADDER/SUBTRACTORS**

**DATE: 24/01/2022**

### **AIM:**

To verify the functionality and timing of your design or portion of your design. To interpret Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation and to create and verify complex functions in a relatively small amount of time.

### **TOOLS REQUIRED:**

1. Xilinx Vivado Design Suite: WebEdition

### **PROCEDURE:**

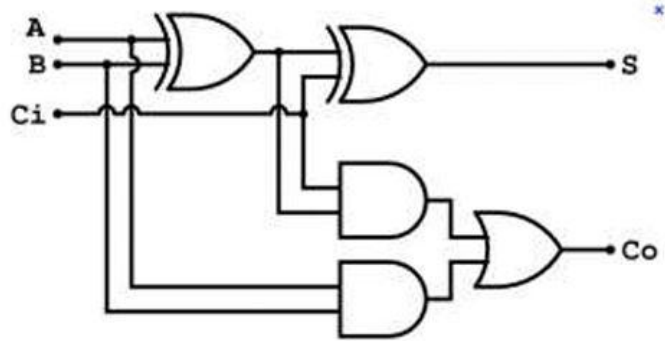
1. Start by clicking **Vivado Design Suite: WebEdition**
2. Go to **File** → **new project** → **Enter project name**, select the top level source as **HDL** & click **next**.
3. Enter **device properties** as  
Product Category : General Purpose  
Family : Kintex 7  
Device : xc7k70t  
Package :fbg484  
Speed : -1  
Top Level Source Type : HDL  
Synthesis Tool : XST (VHDL/Verilog)  
Simulator : ISim (VHDL/Verilog)  
Preferred Language : Verilog  
& **Click next**.
4. **Right click** the device name (XCS3540) in the source window to create **new source**.
5. Select **Verilog module** and enter **file name** in the new source window & click **next**.

6. Write the **Verilog code** in the **Verilog editor window**.
7. Write the **testbench** by create **new source simulation source**.
8. Run Check syntax through **Process window**→ **synthesize**→ double click **Behavioral Check Syntax**→ and removes error if present, with proper syntax & coding.
9. Click on the symbol of FPGA device and then right click→ click on **new source**.
10. Select the desired parameters for simulating the design. In this case **combinational circuit** and **simulation time** click **finish**.
11. Assign **all input signal (high or low)** using just **click** on this and save file.
12. From the **source process window**. Click **Behavioral simulation** from drop-down menu.
13. Double click the **Simulation Behavioral Model**.
14. Verify your design in **wave window** by seeing behavior of output signal with respect to input signal.

## Full Adder:

### b) FULL ADDER:

#### Logic Circuit



#### Truth Table

Input			Output	
A	B	Cin	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

#### Expression

$$\text{Sum} = A \oplus B \oplus \text{Cin}$$

$$\text{Cout} = AB + \text{BCin} + \text{CinA}$$

## PROGRAM

### Gate Level

```
22 ⊞ //1960628 Prem Full Adder Gate Level
23 ⊞ module fulladder(
24 |     input a,
25 |     input b,
26 |     input ci,
27 |     output s,
28 |     output co
29 | );
30 | wire w0,w1,w2;
31 | ○ xor xort(w0,a,b);
32 | ○ xor xort1(s,w0,ci);
33 | ○ and and1(w1,ci,w0);
34 | ○ and and2(w2,a,b);
35 | ○ or ort(co,w1,w2);
36 |
37 ⊞ endmodule
38 |
```

## Data Flow
































```
22 //1960628 Prem Full adder Dataflow
23 module fa_dataflow(
24     input a,
25     input b,
26     input ci,
27     output s,
28     output co
29 );
30 assign s = a^b^ci;
31 assign co = (a&b) | (b&ci) | (ci&a);
32 endmodule
```

## Behavioural

```
22 //1960628 Prem full adder behav
23 module fa_behav(
24     input a,
25     input b,
26     input ci,
27     output reg s,
28     output reg co
29 );
30 always @(a,b,ci)
31 begin
32     if (a == 1'b0 & b == 1'b0 & ci == 1'b0)
33     begin
34         s = 1'b0;
35         co = 1'b0;
36     end
37     if (a == 1'b0 & b == 1'b0 & ci == 1'b1)
38     begin
39         s = 1'b1;
40         co = 1'b0;
41     end
42     if (a == 1'b0 & b == 1'b1 & ci == 1'b0)
43     begin
44         s = 1'b1;
45         co = 1'b0;
46     end
47     if (a == 1'b0 & b == 1'b0 & ci == 1'b1)
48     begin
49         s = 1'b0;
50         co = 1'b1;
51     end
52     if (a == 1'b1 & b == 1'b0 & ci == 1'b0)
53     begin
```

```
54 | s = 1'b1;
55 | co = 1'b0;
56 | end
57 | if (a == 1'b1 & b == 1'b0 & ci == 1'b1)
58 | begin
59 | s = 1'b0;
60 | co = 1'b1;
61 | end
62 | if (a == 1'b1 & b == 1'b1 & ci == 1'b0)
63 | begin
64 | s = 1'b0;
65 | co = 1'b1;
66 | end
67 | if (a == 1'b1 & b == 1'b1 & ci == 1'b1)
68 | begin
69 | s = 1'b1;
70 | co = 1'b1;
71 | end
72 | end
73 | endmodule
74 |
```

## TESTBENCH

```
22  //1960628 Prem Testbench Full adder
23  module tb_fa();
24 | wire s,co;
25 | reg a,b,ci;
26 |
27 | fulladder I1(a,b,ci,s,co);
28  initial
29  begin
30 |  a = 1'b0;
31 |  b = 1'b0;
32 |  ci = 1'b0;
33 |  #10
34 |  a = 1'b0;
35 |  b = 1'b1;
36 |  ci = 1'b0;
37 |  #10
38 |  a = 1'b1;
39 |  b = 1'b0;
40 |  ci = 1'b0;
41 |  #10
42 |  a = 1'b1;
43 |  b = 1'b1;
44 |  ci = 1'b0;
45 |  #10
46 |  a = 1'b0;
47 |  b = 1'b0;
48 |  ci = 1'b1;
49 |  #10
50 |  a = 1'b0;
51 |  b = 1'b1;
52 |  ci = 1'b1;
53 |  #10
54 |  a = 1'b1;
55 |  b = 1'b0;
56 |  ci = 1'b1;
57 | #10
```



```

58 | ○ | a = 1'b1;
59 | ○ | b = 1'b1;
60 | ○ | ci = 1'b1;
61 | ○ → $finish();
62 | ⊖ | end
63 | ⊖ | endmodule

```

### Behavioural testbench

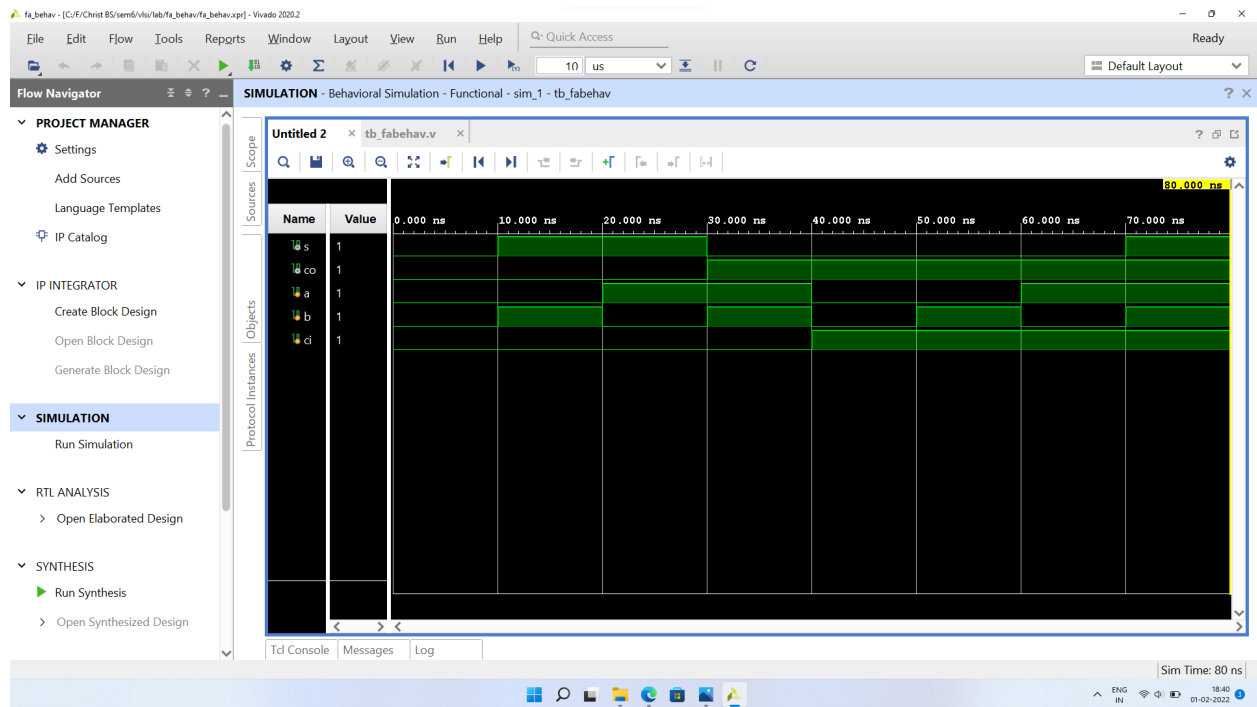
```

23 | ⊖ //1960628 Prem Testbench Full adder
24 | ⊖ module tb_fabehav();
25 |   wire s,co;
26 |   reg a,b,ci;
27 |
28 |   fa_behav I1(a,b,ci,s,co);
29 | ⊖ initial
30 | ⊖ begin
31 |   a = 1'b0;
32 |   b = 1'b0;
33 |   ci = 1'b0;
34 |   #10
35 |   a = 1'b0;
36 |   b = 1'b1;
37 |   ci = 1'b0;
38 |   #10
39 |   a = 1'b1;
40 |   b = 1'b0;
41 |   ci = 1'b0;
42 |   #10
43 |   a = 1'b1;
44 |   b = 1'b1;
45 |   ci = 1'b0;
46 |   #10
47 |   a = 1'b0;
48 |   b = 1'b0;
49 |   ci = 1'b1;
50 |   #10
51 |   a = 1'b0;
52 |   b = 1'b1;
53 |   ci = 1'b1;

```

```
54 | #10
55 | a = 1'b1;
56 | b = 1'b0;
57 | ci = 1'b1;
58 | #10
59 | a = 1'b1;
60 | b = 1'b1;
61 | ci = 1'b1;
62 | #10
63 | $finish();
64 | end
65 | endmodule
```

## OUTPUT

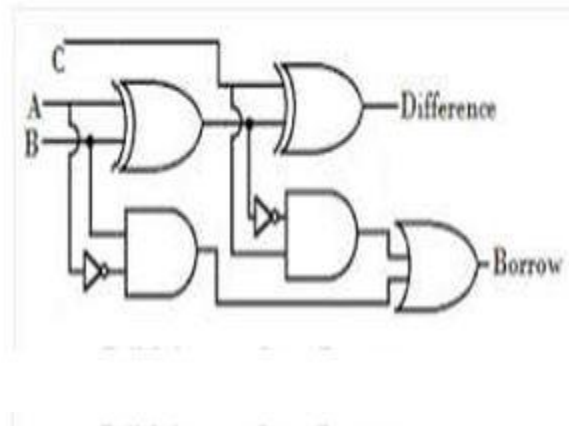


Result:

Thus the Verilog code was scripted, simulated and waveforms were verified using Xilinx Vivado Webpack.

## Full Subtractor:

### Logic Circuit



### Truth Table

Full Subtractor-Truth Table				
Input			Output	
A	B	C	Difference	Borrow
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

### Expression

$$\text{Difference} = A \oplus B \oplus C$$

$$\text{Borrow} = A'B + BC + A'C$$

## PROGRAM









### Gate Level

```
22 //1960628 Prem full subtractor gate level
23 module fsgate(
24     input a,
25     input b,
26     input c,
27     output d,
28     output bor
29 );
30 wire w0,w1,w2,w3,w4;
31 xor xort(w0,a,b);
32 xor xort2(d,c,w0);
33 not nort(w1,w0);
34 not nort2(w2,a);
35 and and1(w3,w2,b);
36 and and2(w4,c,w1);
37 or ort(bor,w3,w4);
38 endmodule
39
```

### Data Flow

```
22 //1960628 Prem full subtractor dataflow
23 module fs_dataflow(
24     input a,
25     input b,
26     input c,
27     output d,
28     output bor
29 );
30 assign d = a^b^c;
31 assign bor = ((~a)&b) | (b&c) | ((~a)&c);
32 endmodule
33
```

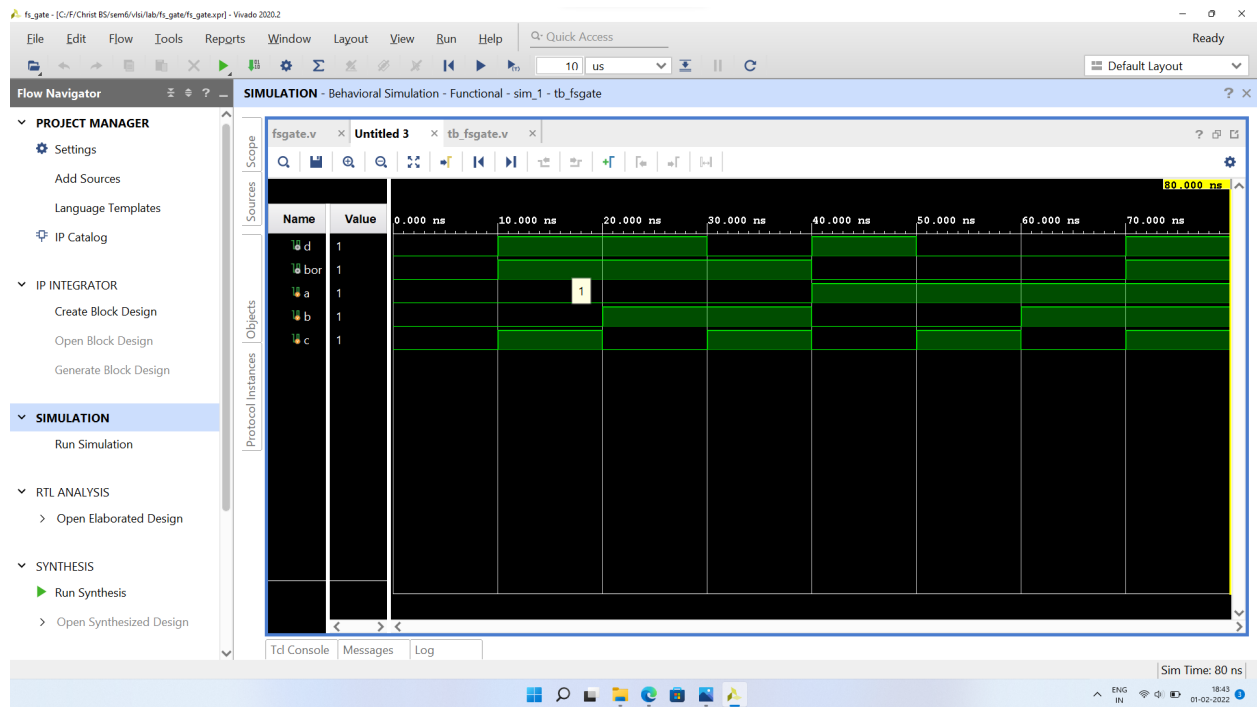
### Behavioural

```
22  //1960628 Prem Full Subtractor Behavioural
23  module fsbehav(
24 |     input [2:0]a,
25 |     output reg [1:0]d
26 | );
27  always@ (a,d)
28  begin
29  case(a)
30 |     3'b000: begin d = 2'b00; end
31 |     3'b001: begin d = 2'b11; end
32 |     3'b010: begin d = 2'b11; end
33 |     3'b011: begin d = 2'b01; end
34 |     3'b100: begin d = 2'b10; end
35 |     3'b101: begin d = 2'b00; end
36 |     3'b110: begin d = 2'b00; end
37 |     3'b111: begin d = 2'b11; end
38  endcase
39  end
40  endmodule
.. |
```

## TESTBENCH

```
--  
22 //1960628 Prem full adder testbench  
23 module tb_fsbehav();  
24     wire [1:0]d;  
25     reg [2:0]a;  
26  
27     fsbehav I1(a,d);  
28     initial  
29     begin  
30         a = 3'b000;  
31         #10  
32         a = 3'b001;  
33         #10  
34         a = 3'b010;  
35         #10  
36         a = 3'b011;  
37         #10  
38         a = 3'b100;  
39         #10  
40         a = 3'b101;  
41         #10  
42         a = 3'b110;  
43         #10  
44         a = 3'b111;  
45         #10  
46         $finish();  
47     end  
48  
49 endmodule
```

## OUTPUT



Result:

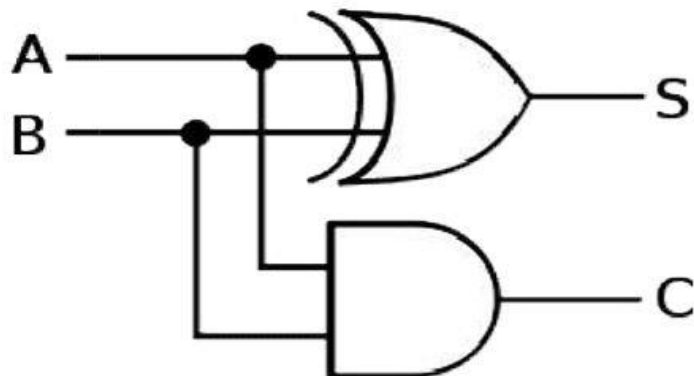
Thus the Verilog code was scripted, simulated and waveforms were verified using Xilinx Vivado Webpack.



**Half Adder:**

**b) HALF ADDER:**

**Logic Circuit**



**Truth Table**

Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

**Expression**

$$\text{Sum} = A \oplus B$$

$$\text{Carry} = A.B$$

## PROGRAM

### Gate Level

```
22 //1960628 Prem half adder gate level
23 module hagate(
24     input a,
25     input b,
26     output s,
27     output c
28 );
29 xor xort(s,a,b);
30 and and1(c,a,b);
31 endmodule
32
```

### Data Flow

```
22 //1960628 Prem Half adder dataflow
23 module hadataflow(
24     input a,
25     input b,
26     output s,
27     output c
28 );
29 assign s = a^b;
30 assign c = a&b;
31
32 endmodule
33
```

## Behavioural

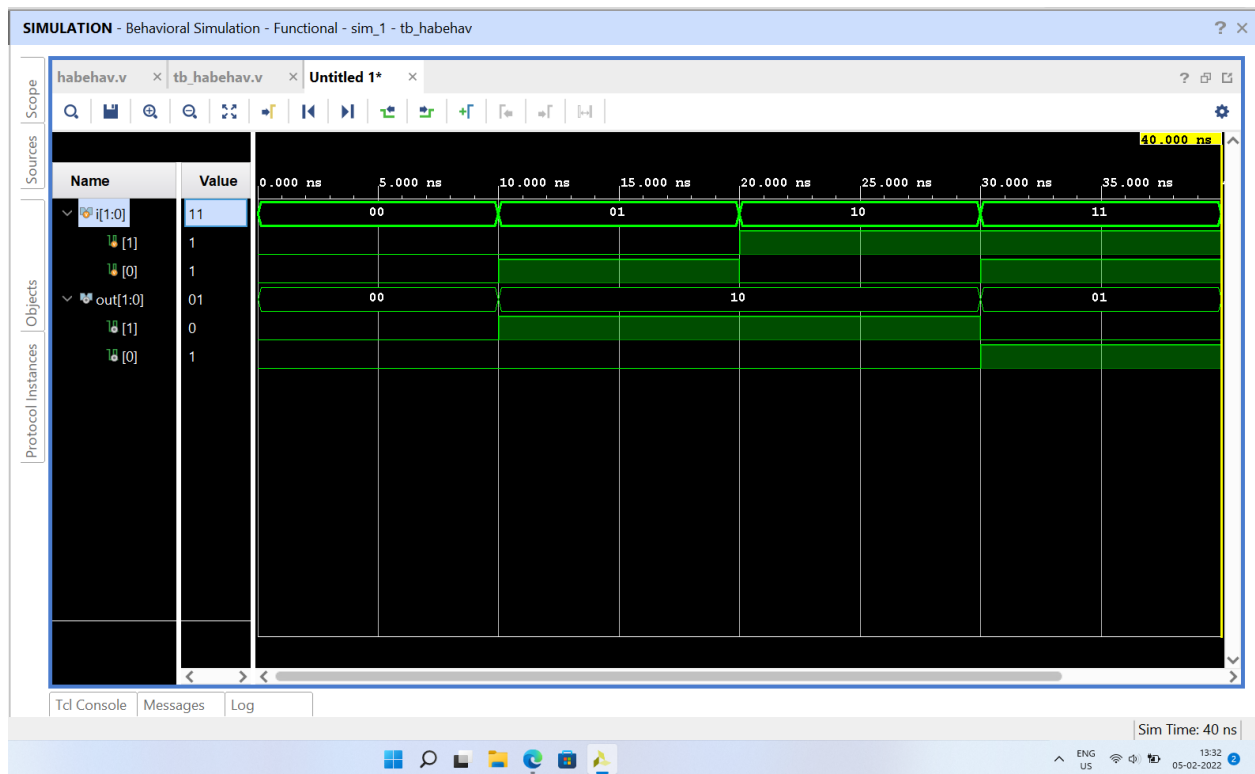
```
-- |
22 ⊞ //1960628 prem half adder behavioural
23 ⊞ module hsbehav(
24 |     input [1:0]a,
25 |     output reg [1:0]d
26 | );
27 ⊞ always @(d,a)
28 ⊞     begin
29 ⊞         case(a)
30 |             2'b00: begin d = 2'b00; end
31 |             2'b01: begin d = 2'b11; end
32 |             2'b10: begin d = 2'b10; end
33 |             2'b11: begin d = 2'b00; end
34 ⊞         endcase
35 ⊞     end
36 ⊞ endmodule
37 |
```

---

## TESTBENCH

```
22 //1960628 Prem Half Adder testbench
23 module tb_hsbehav();
24     wire [1:0]d;
25     reg [1:0]a;
26
27     hsbehav I1(a,d);
28     initial
29     begin
30         a = 2'b00;
31         #10
32         a = 2'b01;
33         #10
34         a = 2'b10;
35         #10
36         a = 2'b11;
37         #10
38         $finish();
39     end
40
41 endmodule
42
```

## OUTPUT



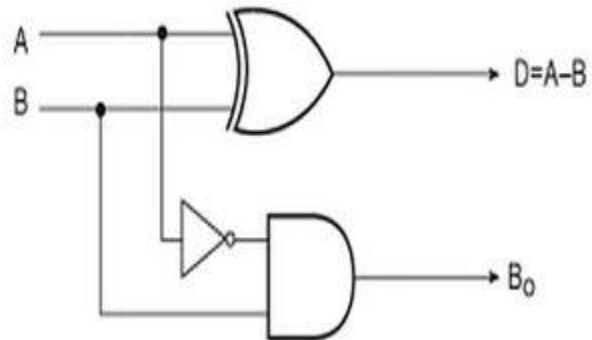
Result:

Thus the Verilog code was scripted, simulated and waveforms were verified using Xilinx Vivado Webpack.

## Half Subtractor:

### c) HALF SUBTRACTOR:

#### Logic Circuit



#### Truth Table

A	B	D	B <sub>0</sub>
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0







#### Expression

$$D = A \oplus B$$




$$B = A' \cdot B$$

## PROGRAM









### Gate Level

```
22  //1960628 Prem Half Subtracotr Gate level
23  module hsgate(
24 |         input a,
25 |         input b,
26 |         output d,
27 |         output bo
28 |     );
29 |     wire w0;
30 |  xor xort(d,a,b);
31 |  not not1(w0,a);
32 |  and and1(bo,b,w0);
33  endmodule
34 |
```

### Data Flow




















```
22  //1960628 Prem Half Subtractor Data flow
23  module hsdata(
24 |     input a,
25 |     input b,
26 |     output d,
27 |     output bo
28 | );
29 | assign d = a^b;
30 | assign bo = (~a)&b;
31  endmodule
```

### Behavioural

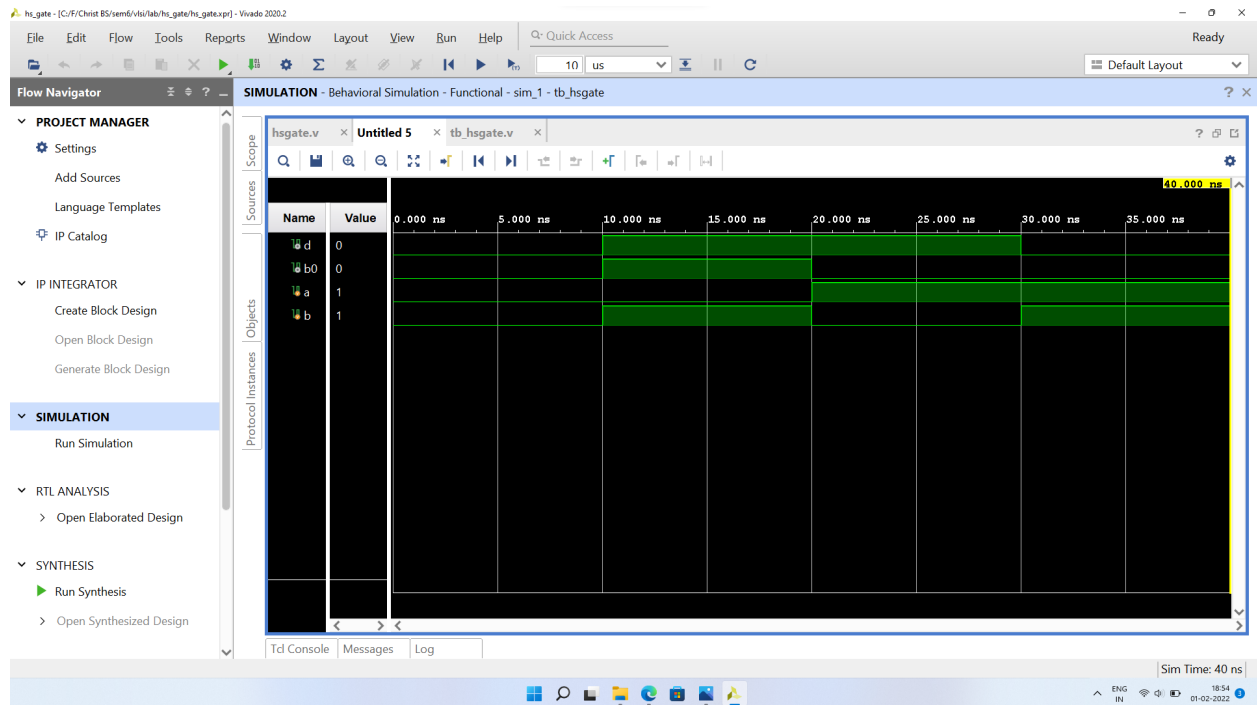
```
22  //1960628 prem half subtracotr behavioural
23  module hsbehav(
24 |       input [1:0]a,
25 |       output reg [1:0]d
26 |       );
27  always @(d,a)
28      begin
29          case(a)
30 |             2'b00: begin d = 2'b00; end
31 |             2'b01: begin d = 2'b11; end
32 |             2'b10: begin d = 2'b10; end
33 |             2'b11: begin d = 2'b00; end
34          endcase
35      end
36  endmodule
-- |
```



## TESTBENCH

```
22  //1960628 Prem Half Subtracotr testbench
23  module tb_hsgate();
24 | wire d,b0;
25 | reg a,b;
26 |
27 | hsgate I1(a,b,d,b0);
28  initial
29  begin
30 |  a = 1'b0;
31 |  b = 1'b0;
32 |  #10
33 |  a = 1'b0;
34 |  b = 1'b1;
35 |  #10
36 |  a = 1'b1;
37 |  b = 1'b0;
38 |  #10
39 |  a = 1'b1;
40 |  b = 1'b1;
41 |  #10
42 |  → $finish();
43  end
44  endmodule
```

## OUTPUT



Result:

Thus the Verilog code was scripted, simulated and waveforms were verified using Xilinx Vivado Webpack.