EXPT.-03

AIM: - Design and Simulate Full subtractor using Verilog HDL.

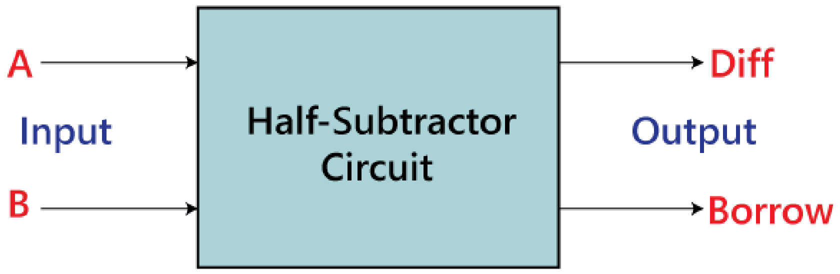
Software: - Xilinx Vivado 2019.1

Theory: -

Half Subtractor

The half subtractor is also a building block for subtracting two binary numbers. It has two inputs and two outputs. This circuit is used to subtract two single bit binary numbers A and B. The **'diff'** and **'borrow'** are two output states of the half subtractor.

Block diagram



Truth Table

Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

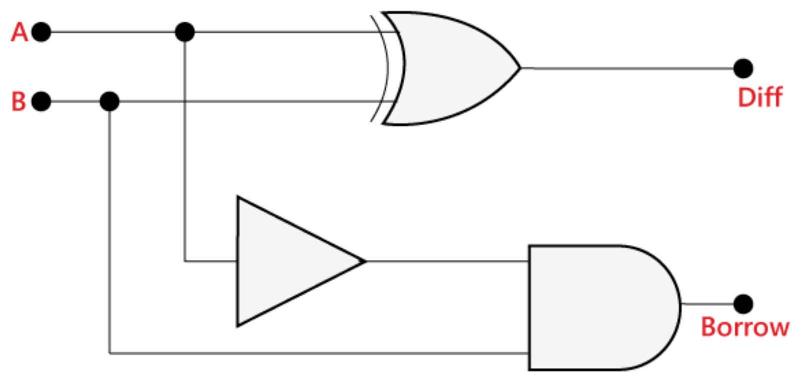
The SOP form of the **Diff** and **Borrow** is as follows:

Diff=A'B+AB'
Borrow = A'B

In the above table,

- 'A' and 'B' are the input variables whose values are going to be subtracted.
- The 'Diff' and 'Borrow' are the variables whose values define the subtraction result, i.e., difference and borrow.
- The first two rows and the last row, the difference is 1, but the 'Borrow' variable is 0.

- The third row is different from the remaining one. When we subtract the bit 1 from the bit 0, the borrow bit is produced.

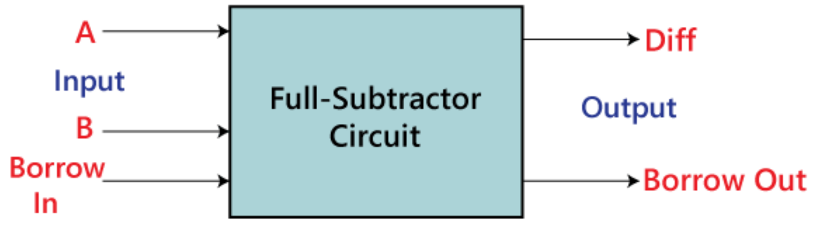


Half-Subtractor Circuit

Full Subtractor

The Half Subtractor is used to subtract only two numbers. To overcome this problem, a full subtractor was designed. The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., diff and borrow.

Block diagram



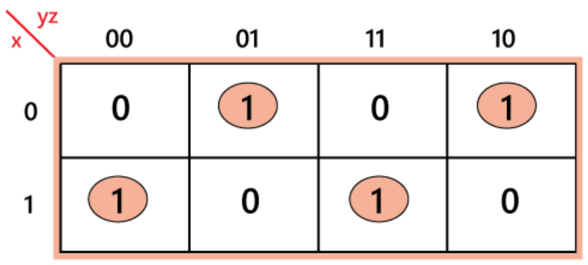
Truth Table

Inputs			Outputs	
A	B	$Borrow_{in}$	Diff	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

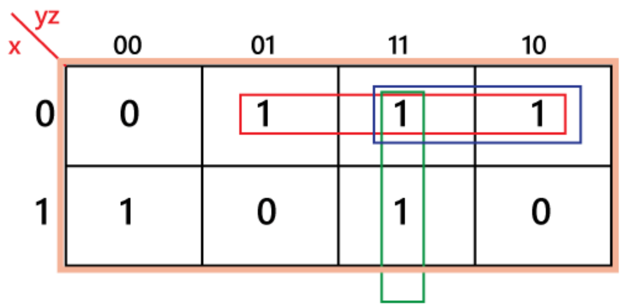
In the above table,

- 'A' and' B' are the input variables. These variables represent the two significant bits that are going to be subtracted.
- 'Borrow$_{in}$' is the third input which represents borrow.
- The 'Diff' and 'Borrow' are the output variables that define the output values.
- The eight rows under the input variable designate all possible combinations of 0 and 1 that can occur in these variables.

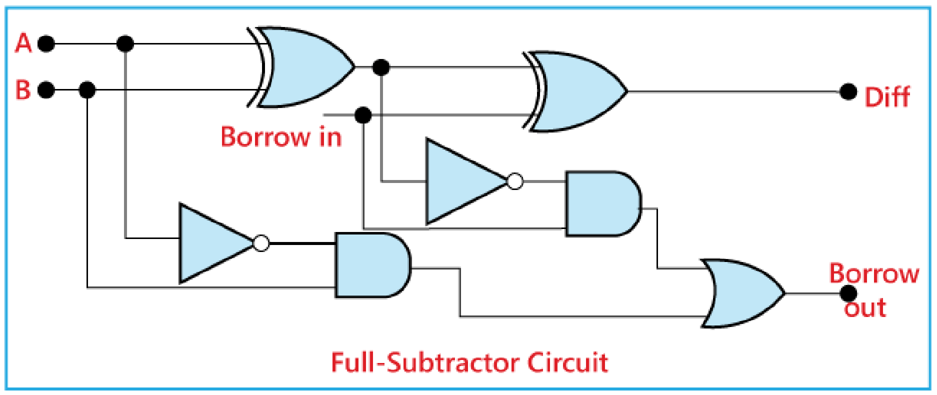
The SOP form can be obtained with the help of K-map as:

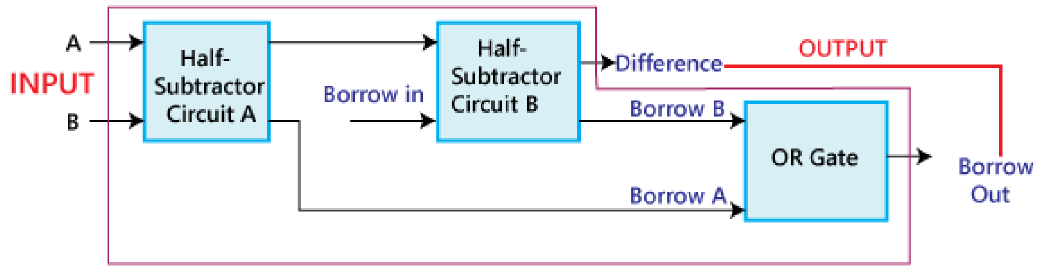


Diff=xy' z'+x' y' z+xyz+x'yz'



Borrow=x' z+x' y+yz





Code: -

```
//HS
`timescale 1ns / 1ps

module
Half_subtractor(a,b,diff,borrow);
input a,b;
output diff,borrow;
assign diff = a^b;
assign borrow= ~a & b;

endmodule
```

```
/tb HS
`timescale 1ns / 1ps

module FS_test_bench();
//reg x,y,z;
reg x,y;
//wire diff_out,borrow_out;
wire diff,borrow;
//full_subtractor FS(x,y,z,diff_out,borrow_out);
Half_subtractor HS1(x,y,diff,borrow);

integer i;
initial
begin
for(i=0;i<4;i=i+1)
begin
{x,y}=i;
#10;
end
end

initial
//$monitor($time,"input x=%b,y=%b,z=%b,output
diff_out=%b,borrow_out=%b",x,y,z,diff_out,borrow_out);
$monitor($time,"input x=%b,y=%b,output
diff=%b,borrow=%b",x,y,diff,borrow);
initial
#40 $finish;

endmodule
```

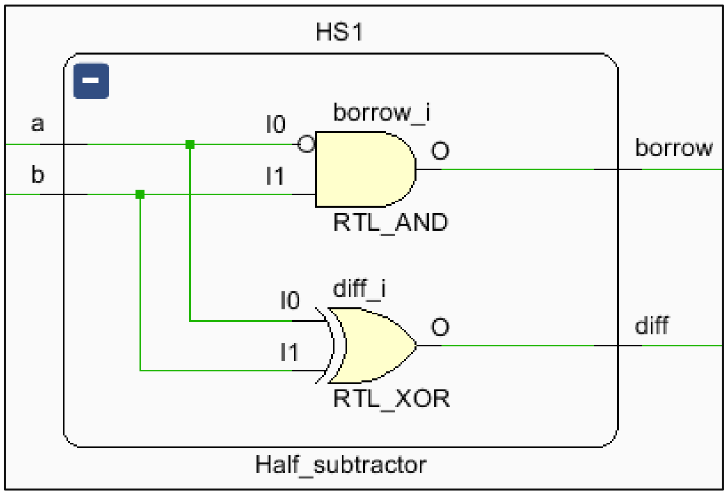
```
//tb FS
`timescale 1ns / 1ps

module FS_test_bench();
reg x,y,z;
//reg x,y;
wire diff_out,borrow_out;
//wire diff,borrow;
full_subtractor FS(x,y,z,diff_out,borrow_out);
//Half_subtractor HS1(x,y,diff,borrow);

integer i;
initial
begin
for(i=0;i<8;i=i+1)
begin
{x,y,z}=i;
#10;
end
end

initial
$monitor($time,"input x=%b,y=%b,z=%b,output
diff_out=%b,borrow_out=%b",x,y,z,diff_out,borrow_out);
//$monitor($time,"input x=%b,y=%b,output
diff=%b,borrow=%b",x,y,diff,borrow);

initial
#40 $finish;

endmodule
```

```
//FS
`timescale 1ns / 1ps

module
full_subtractor(a1,b1,c1,diff_out,borrow_out);
input a1,b1,c1;
output diff_out,borrow_out;
wire [2:0] t;

Half_subtractor
HS1(.a(a1),.b(b1),.diff(t[0]),.borrow(t[1]));
Half_subtractor
HS2(.a(t[0]),.b(c1),.diff(diff_out),.borrow(t[2]));
or or1(borrow_out,t[1],t[2]);
endmodule
```

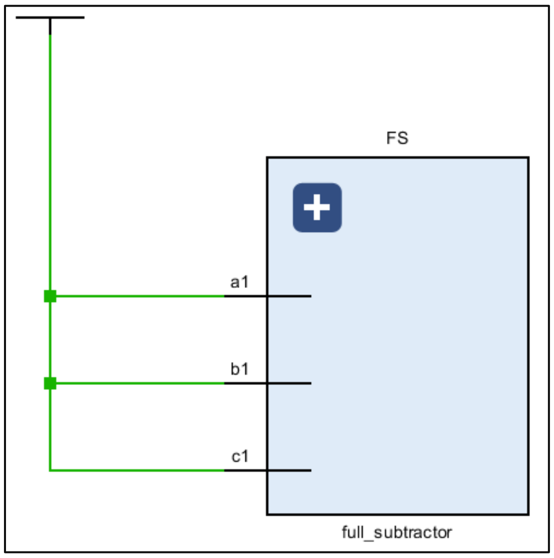
Schematics and Simulated Waveforms: -

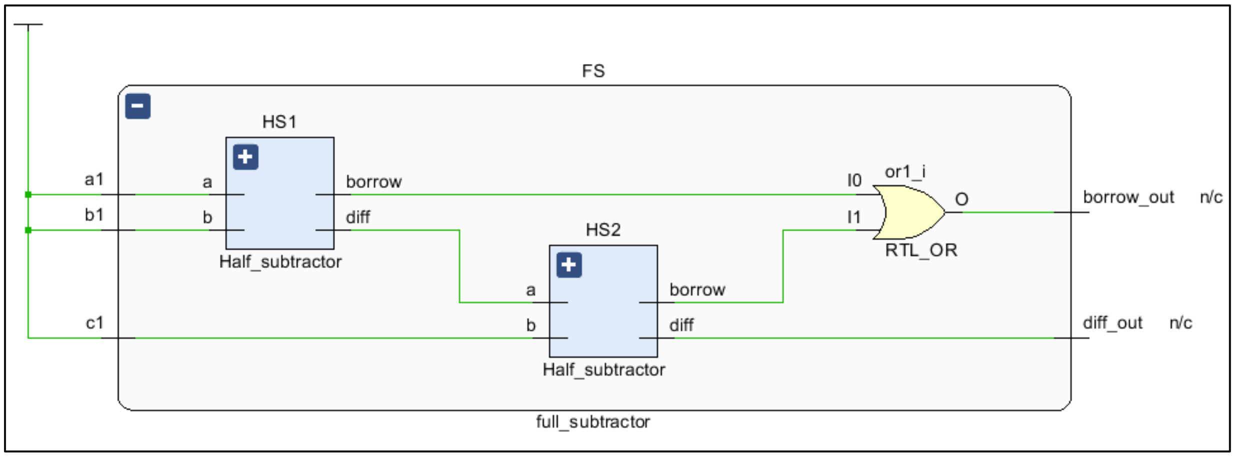
Schematic:

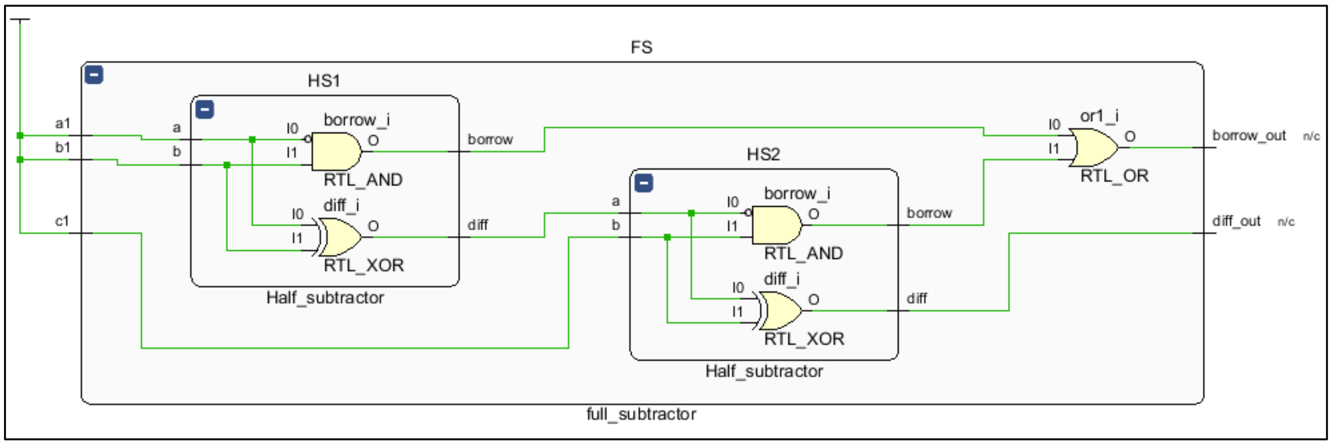
Half subtractor



Full subtractor

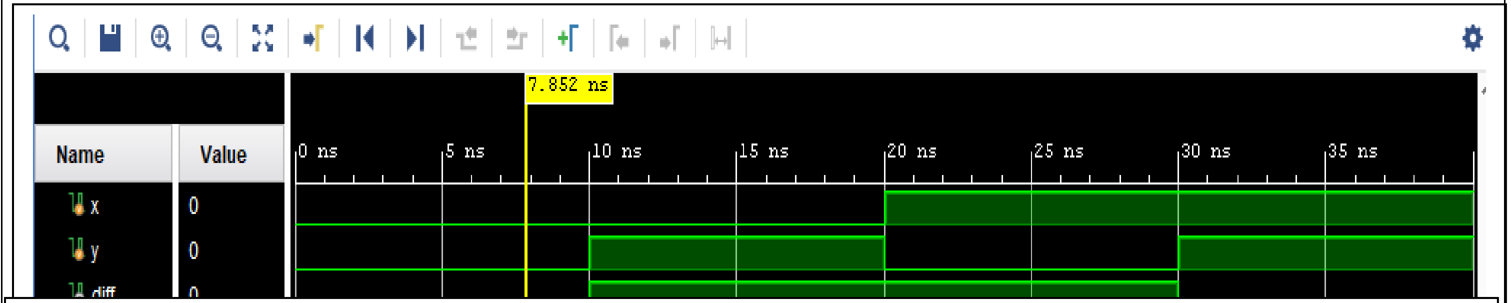


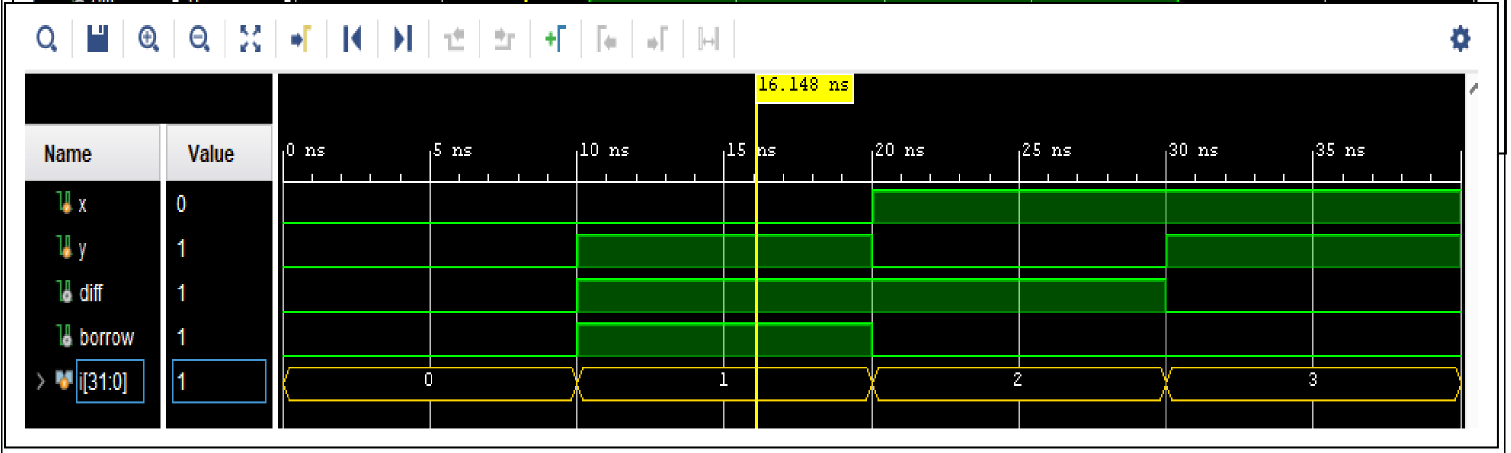


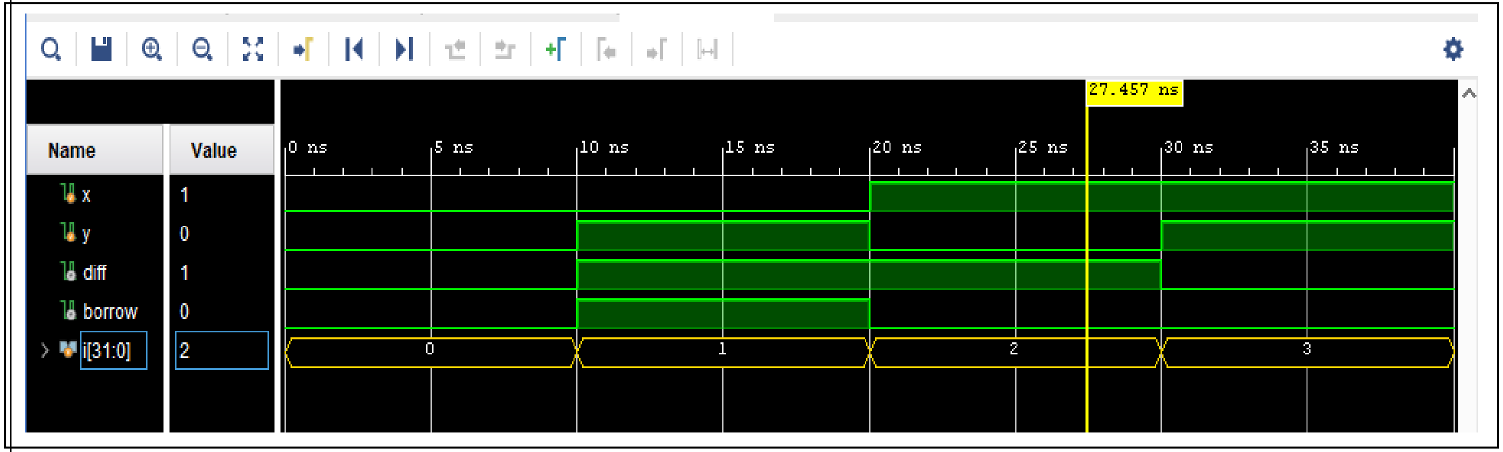


Waveform:

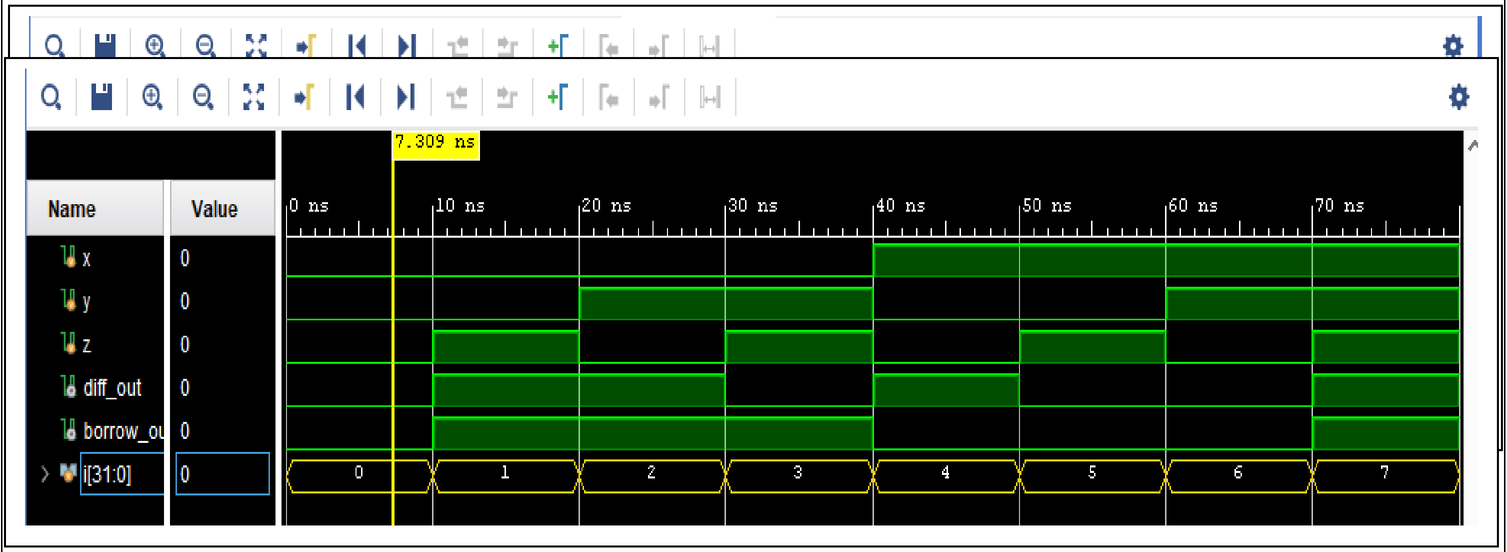
Half subtractor

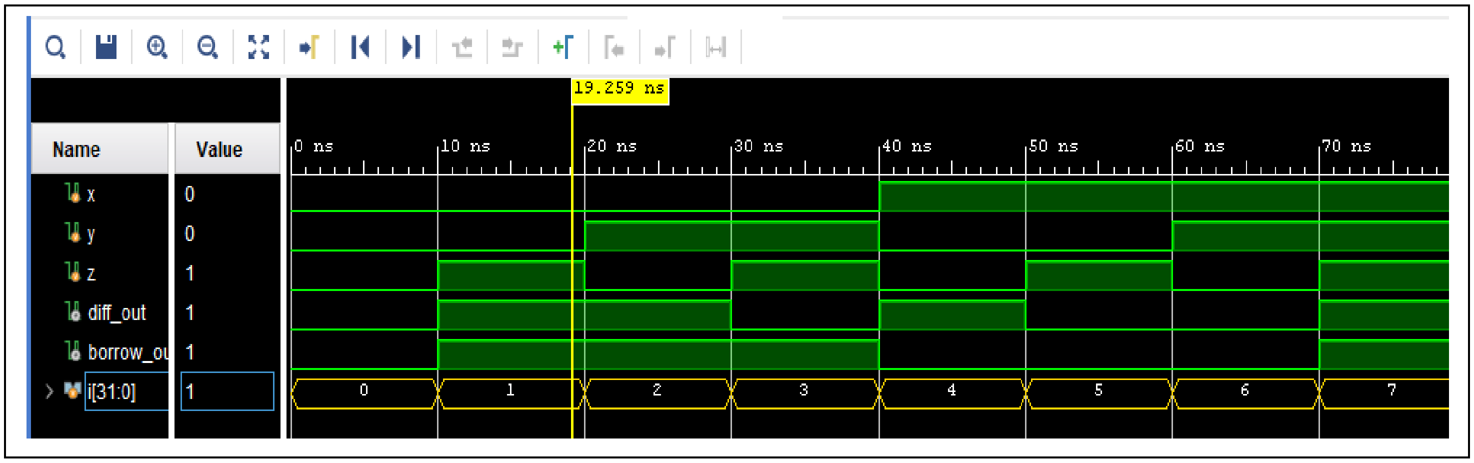




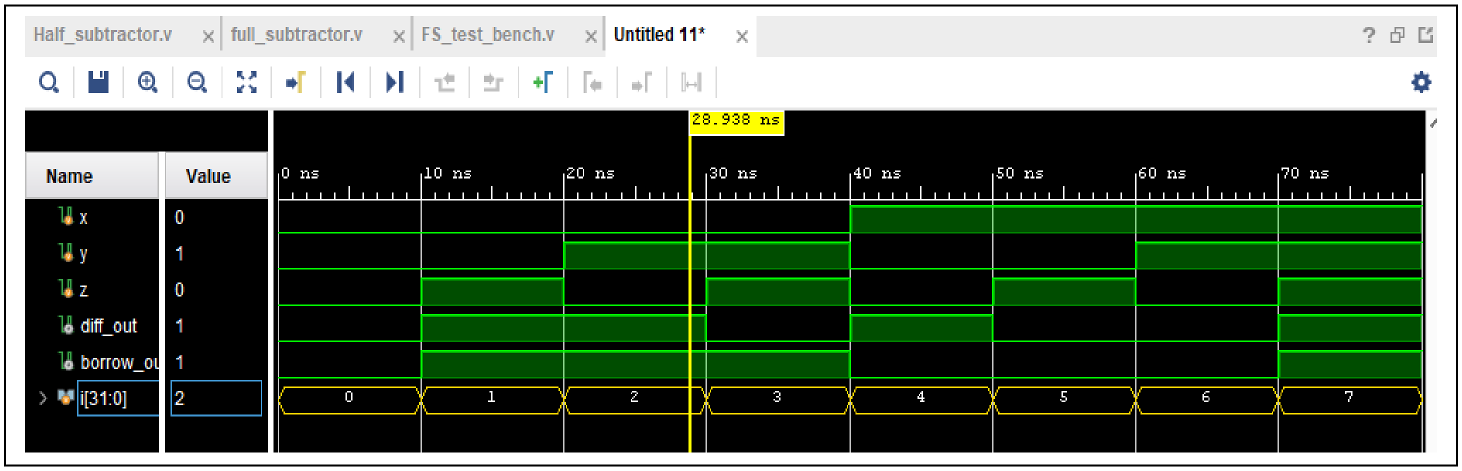


Full subtractor

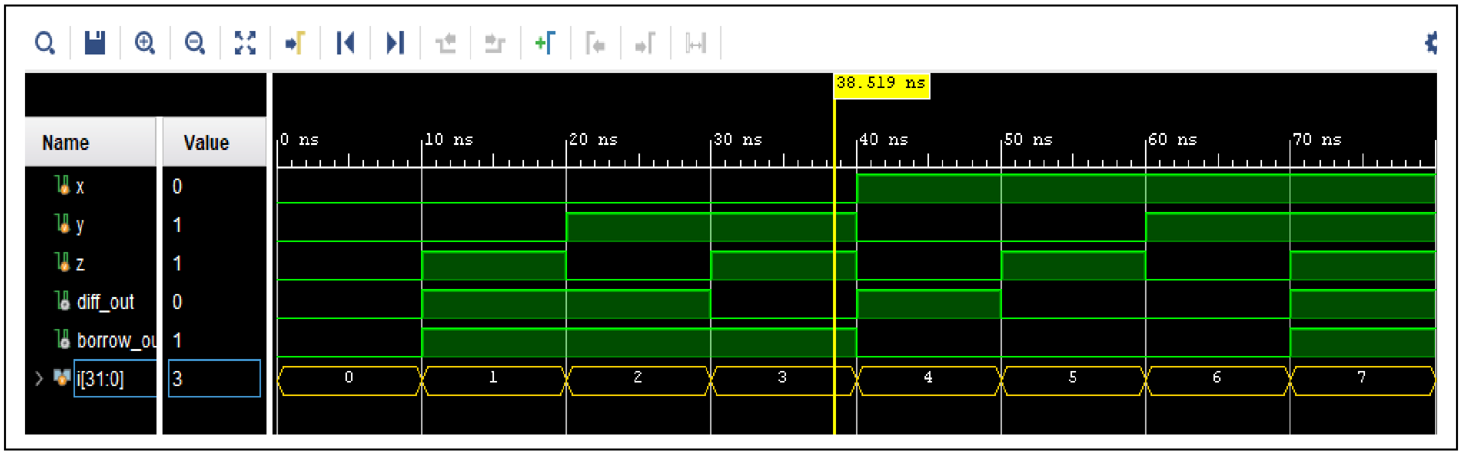




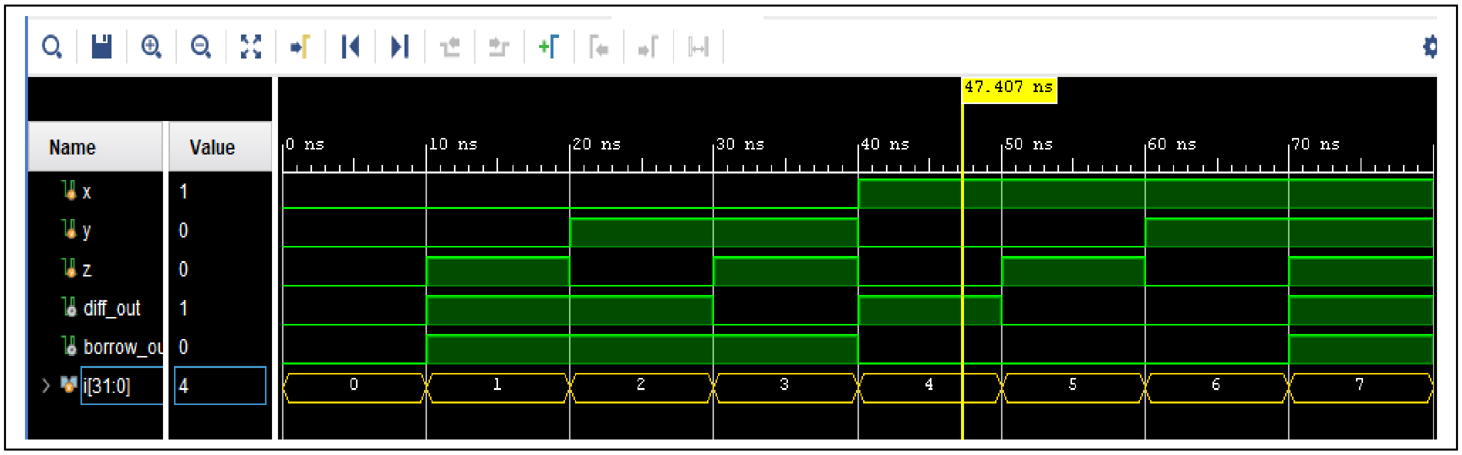
Name	Value
x	0
y	0
z	1
diff_out	1
borrow_ou	1
i[31:0]	1

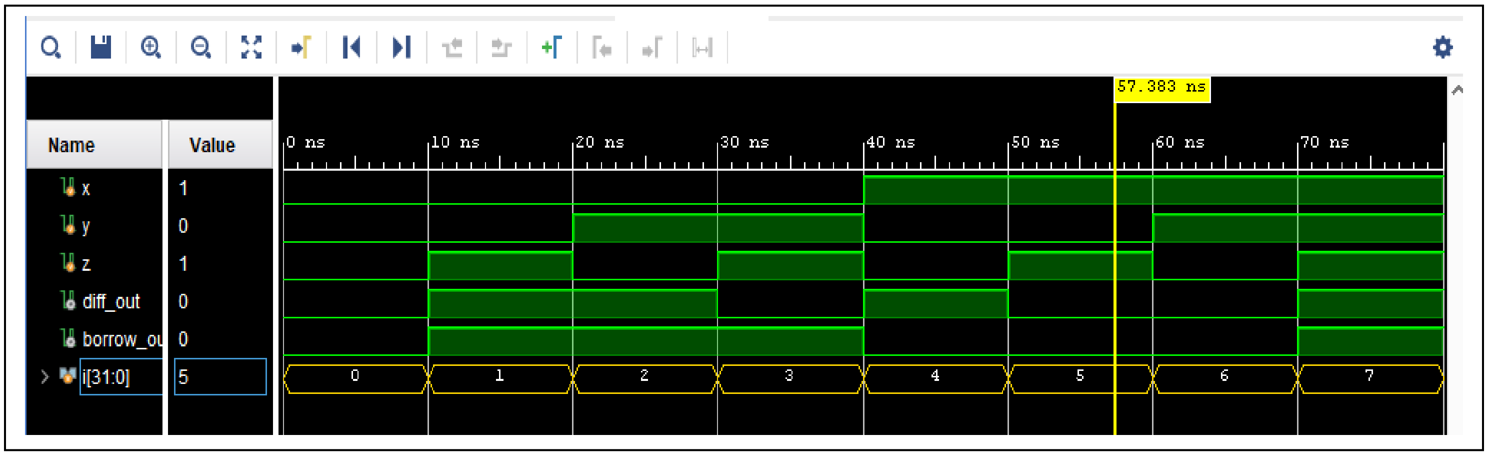


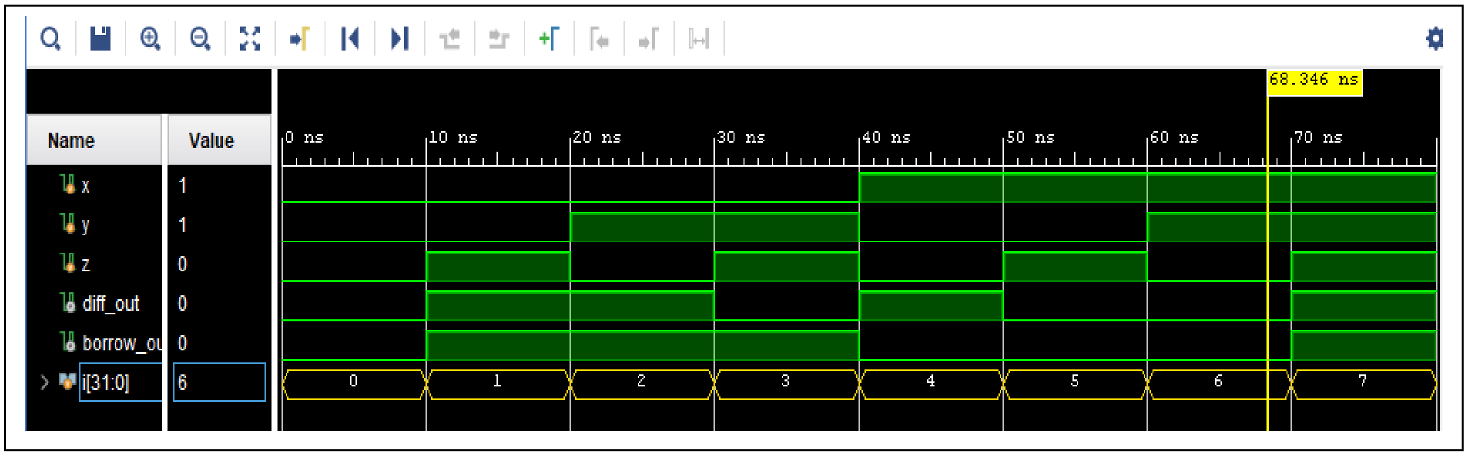
Name	Value
x	0
y	1
z	0
diff_out	1
borrow_ou	1
i[31:0]	2

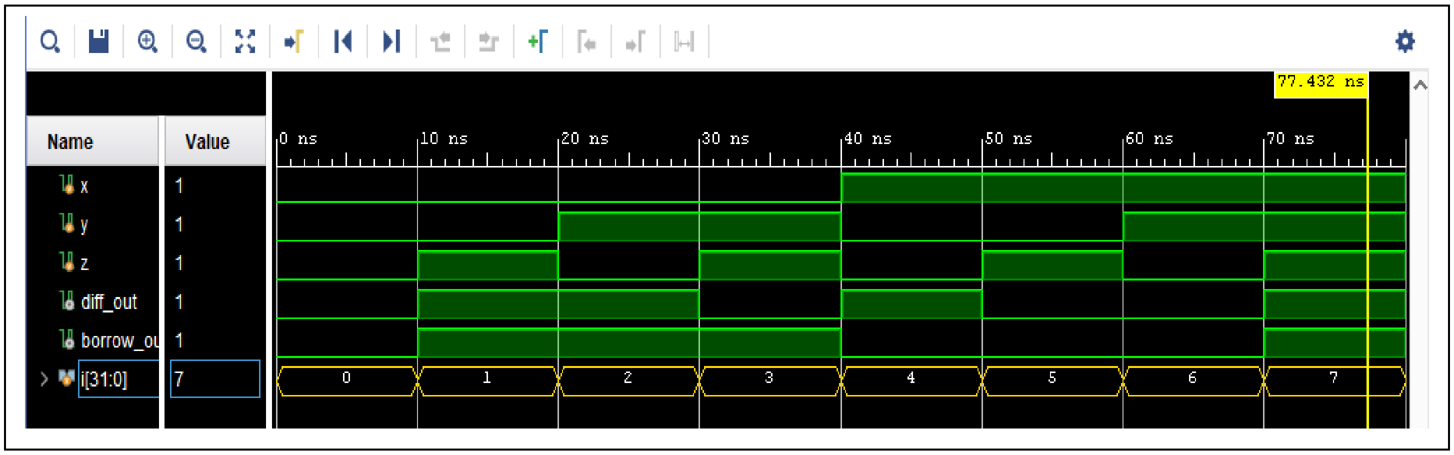


Name	Value
x	0
y	1
z	1
diff_out	0
borrow_ou	1
i[31:0]	3


47.407 ns
Name
Value
x
1
y
0
z
0
diff_out
1
borrow_ou
0
i[31:0]
4
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
0
1
2
3
4
5
6
7


57.383 ns
Name
Value
x
1
y
0
z
1
diff_out
0
borrow_ou
0
i[31:0]
5
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
0
1
2
3
4
5
6
7


68.346 ns
Name
Value
x
1
y
1
z
0
diff_out
0
borrow_ou
0
i[31:0]
6
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
0
1
2
3
4
5
6
7



Name	Value
x	1
y	1
z	1
diff_out	1
borrow_ou	1
i[31:0]	7

<u>Conclusion: </u>-

Hence , we have successfully designed and simulated Half subtractor and Full subtractor using gate-level modeling style in Verilog HDL. Also verified simulated waveforms with actual truth tables using Xilinx Vivado 2019.1