Worksheet of Experiment

Date of Experiment	27 Aug, 2024
Name	Akshada Jotiram Ghorpade
Registration Number	12302956

Title of Experiment: Develop RTL designs in Verilog to implement adders and subtractors and verify their functionality using cadence NCSIM.

Name of Language: Verilog.

Name of Software: Cadence NCSIM, simvision.

Theory of Experiment:

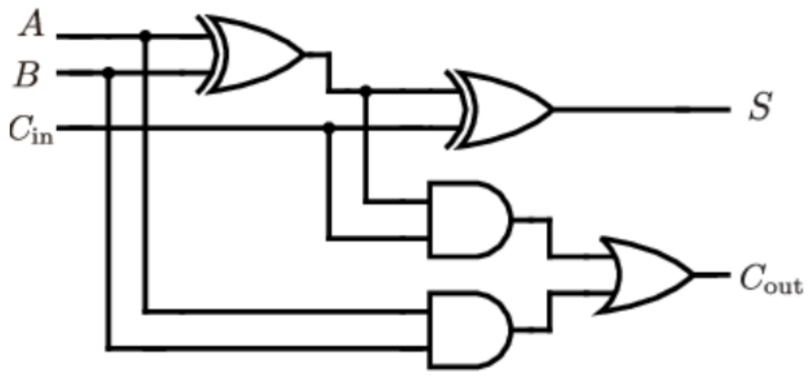
- **Full Adder:**

Truth Table:

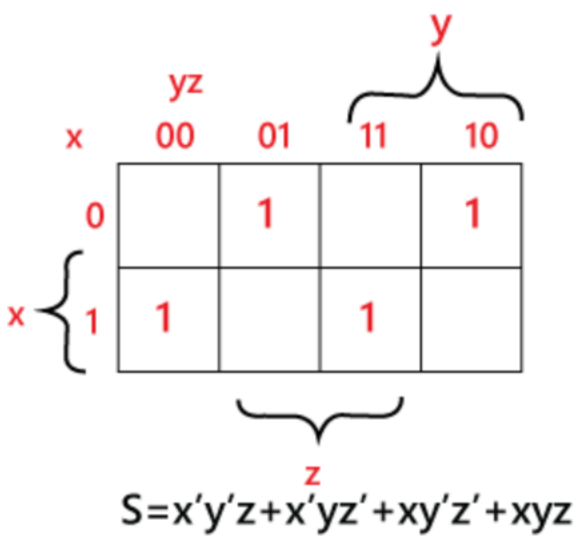
Input			Output	
A	**B**	**Cin**	**SUM**	**CARRY**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

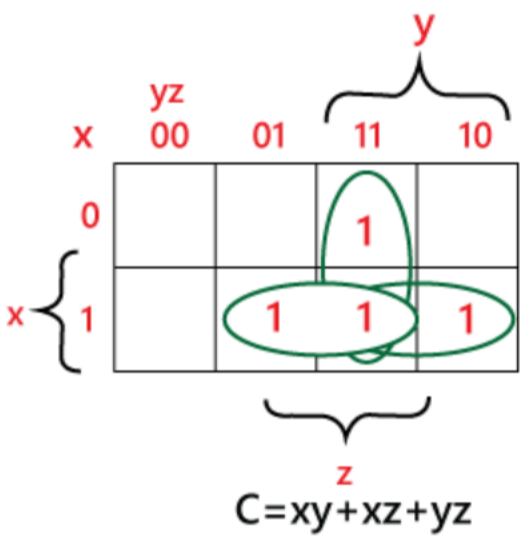
- 'A' and' B' are the input variables. These variables represent the two significant bits which are going to be added
- 'C_{in}' is the third input which represents the carry. From the previous lower significant position, the carry bit is fetched.
- The 'Sum' and 'Carry' are the output variables that 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.

Circuit Diagram:



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





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

Carry = xy+xz+yz

Boolean expression:

$$sum = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + BC_{in} + AC_{in}$$

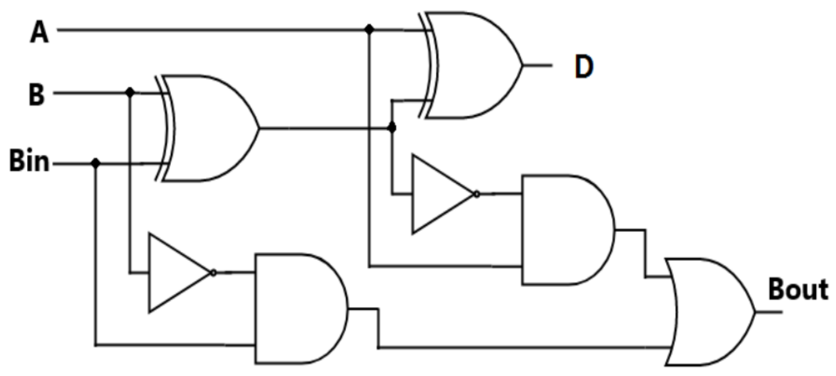
- **Full Subtractor**

Truth Table:

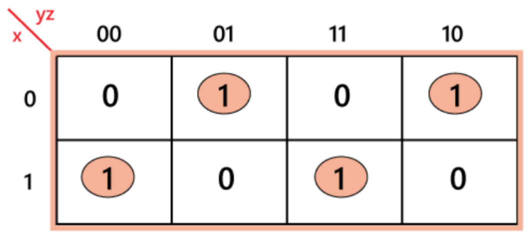
Input			Output	
A	B	$Borrow_{in}$	Diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- '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.

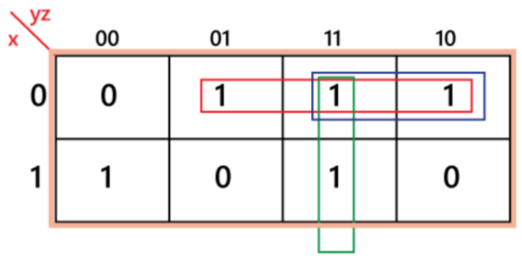
Circuit Diagram:



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

Boolean expression:

$$\boxed{\begin{array}{c} D = A \oplus B \oplus Bin \\ Bout = A'\,Bin + A'\,B + B\,Bin \end{array}}$$

Program and Stimulus (Testbench) program:

PROGRAM (Full Adder)

```
module full_adder (a, b, c, sum, carry);

input a, b, c;

output sum, carry;

Wire w1, w2, w3, w4;

xor x1(w1, a, b);

xor x2(sum, w1, c);

and a1(w2, a, b);

and a2(w3, b, c);

and a3(w4, a, c);

or o1(carry, w2, w3, w4);

endmodule
```

TESTBENCH (Full Adder)

```
module full_addertb;
reg a, b, c;
wire sum, carry;
full_adder dut (a, b, c, sum, carry);
initial
begin
a=0; b=0; c=0;
#10 a=0; b=0; c=1;
#10 a=0; b=1; c=0;
#10 a=0; b=1; c=1;
#10 a=1; b=0; c=0;
#10 a=1; b=0; c=1;
#10 a=1; b=1; c=0;
#10 a=1; b=1; c=1;
#10 $ stop;
end
endmodule
```

PROGRAM (Full Subtractor)

```
module full_subtractor (a, b, c, diff, bor);
input a, b, c;
output diff, bor;
wire (w1, w2, w3, w4, w5);
xor x1(w1, a, b);
xor x2(diff, w1, c);
not n1(w2, a);
```

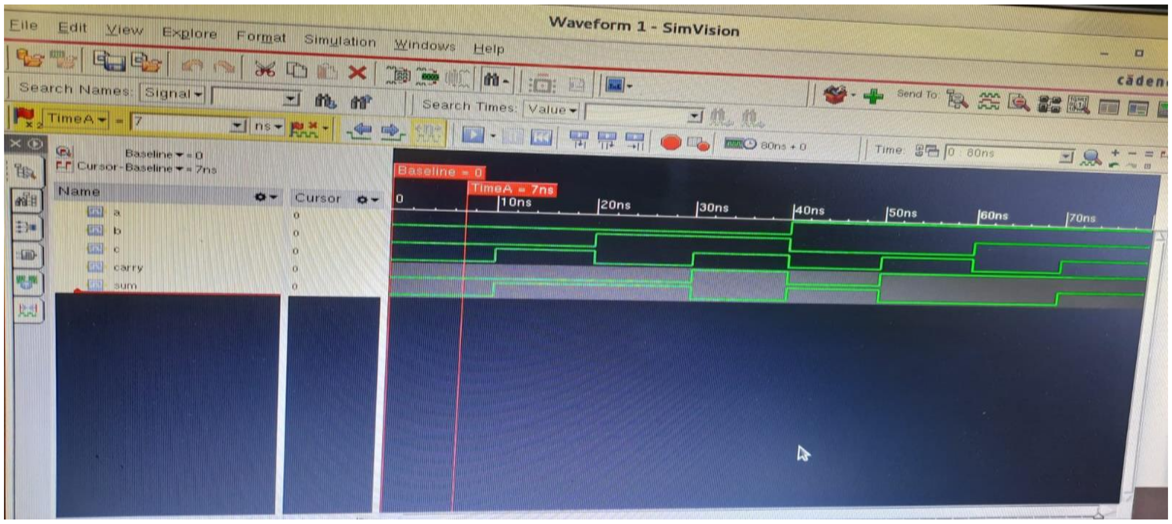
```
and a1(w3, w2, c);
and a2(w4, b, c);
and a3(w5, w2, b);
or o1(bor, w3, w4, w5);
endmodule
```

TESTBENCH (Full Subtractor)

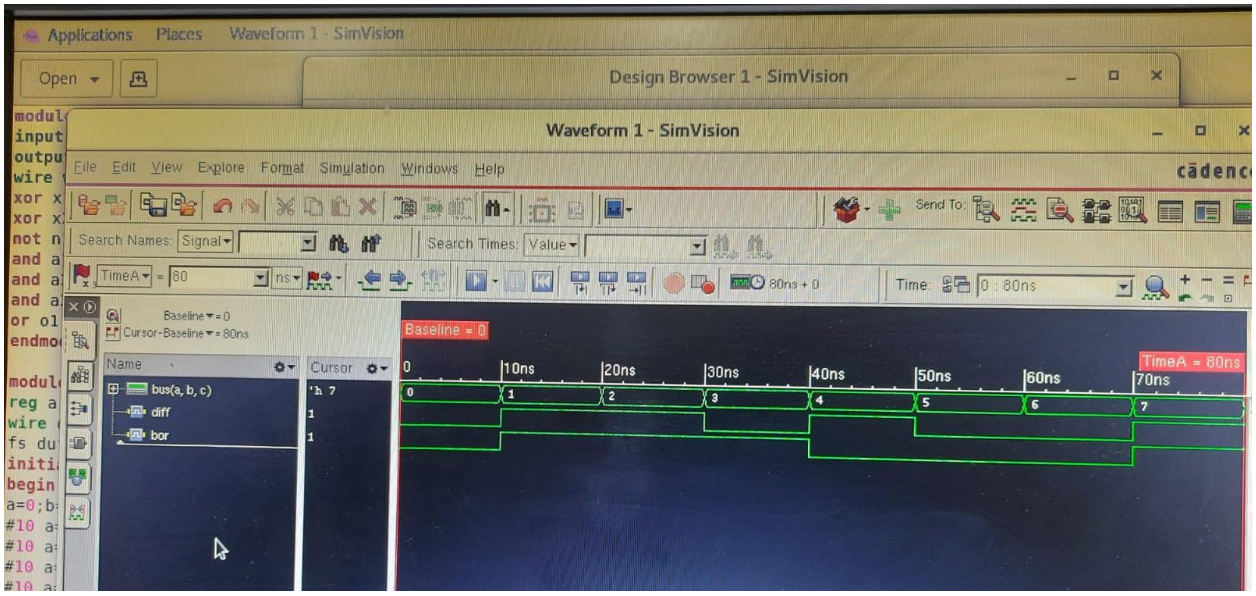
```
module full_subtractortb;
reg a, b, c;
wire diff, bor;
full_subtractor dut (a, b, c, diff, bor);
initial
begin
a=0; b=0; c=0;
#10 a=0; b=0; c=1;
#10 a=0; b=1; c=0;
#10 a=0; b=1; c=1;
#10 a=1; b=0; c=0;
#10 a=1; b=0; c=1;
#10 a=1; b=1; c=0;
#10 a=1; b=1; c=1;
#10 $ stop;
end
endmodule
```

Simulation waveform

Full Adder:

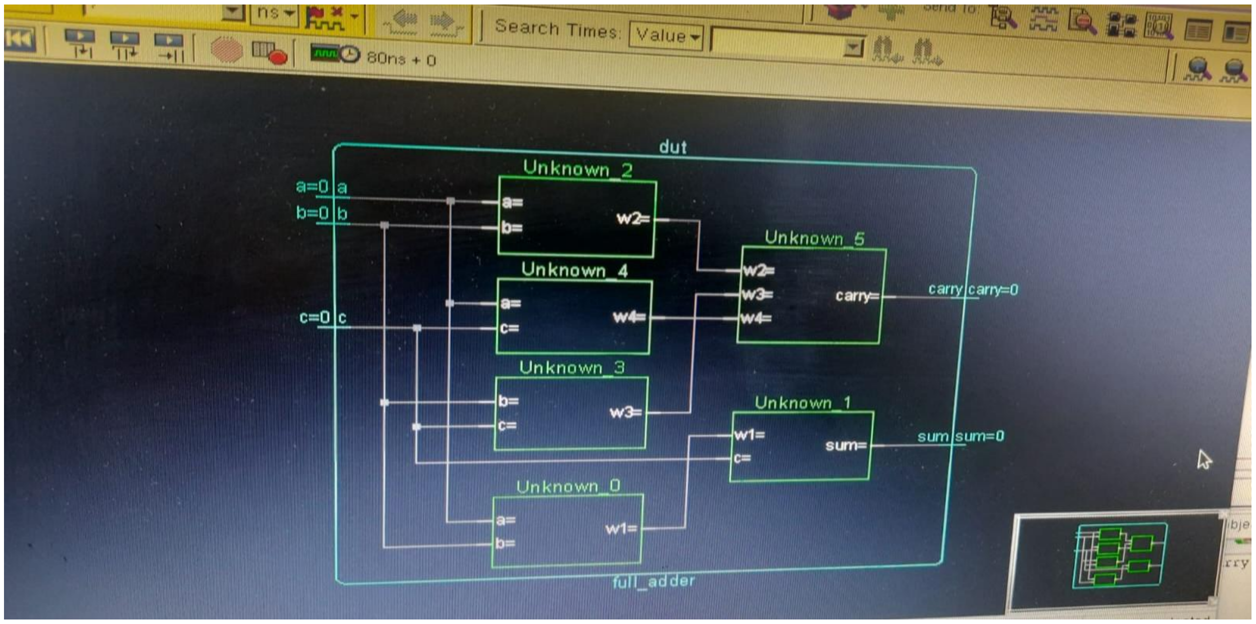


Full Subtractor:

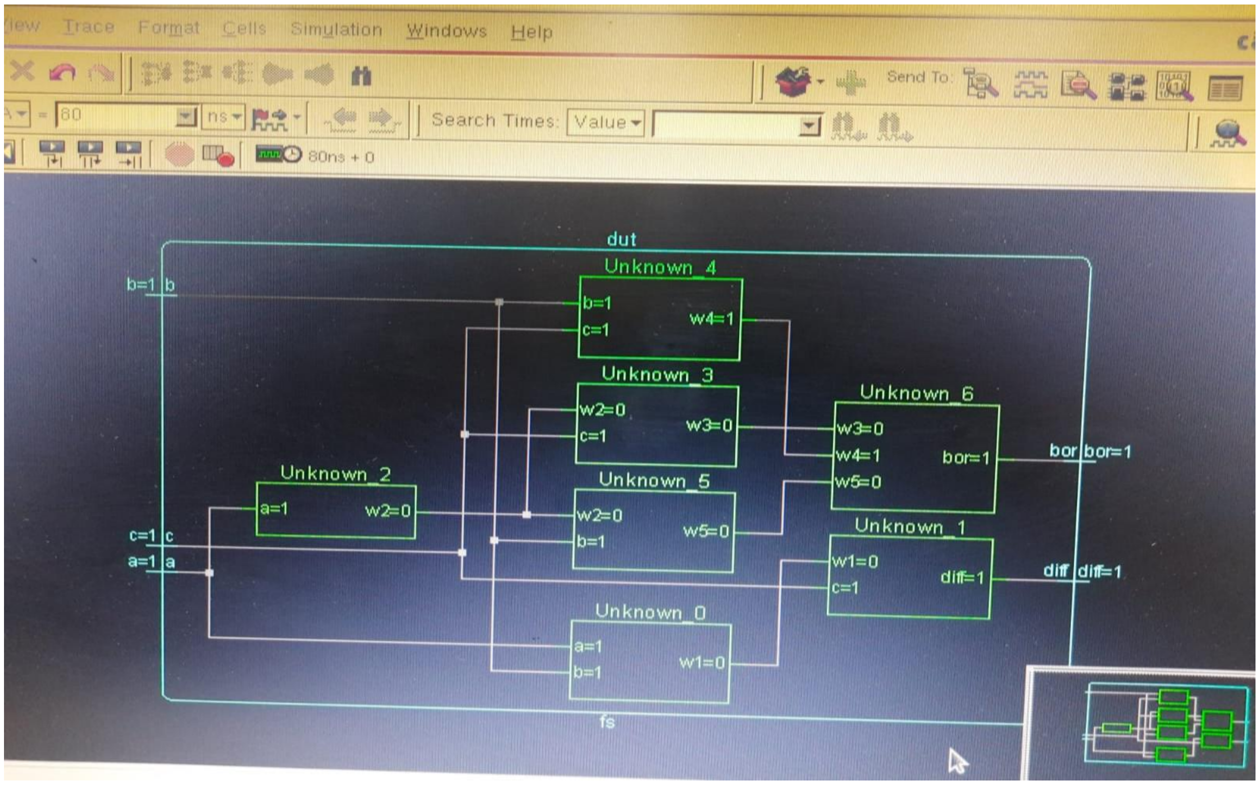


RTL Schematic

Full Adder:



Full Subtractor:



Learning Outcome:

1. Learn to write syntactically correct Verilog code for full adder and full subtractor.

2. Create testbenches to simulate and verify the functionality of full adder and full subtractor.

3. Apply techniques to optimize logic circuits for speed, area, and power efficiency.

4. Perform timing analysis to ensure circuits meet required timing constraints.

5. Analyze the performance of full adder and full subtractor and circuits in terms of timing, power, and area using simulation results.