Assignment Submission Sheet

Term: 321221	**Submission Date: 28-09-2021**	
Lecture Date: 15-09-2021	**Assignment Number: 02**	
Course Code: ECE290	**Section: E1901**	**Group: A**
Registration Number: 11904463	**Student Name: Mohit Rawat**	**Roll No: 09**

1. **Concept Learned**

   I have learned how to make half adder and full adder using verilog and how to make 3-bit ripple carry adder that addes 3 bit number and make wave form of input and output signal.

2. **Key Observations &Insights**

   My key observation was how ripple carry adder addes two number and gives result and how our test bench make output form of output given by 3-bit ripple carry adder.

3. **Application Areas**

   The application of adder is in ALU of processor. This is unit of processor where all mathematical operation done and addition is basic process of ALU. It use addition to for subtraction, division and multiplication like operation.

4. **A)**

   Half Adder/ Full Adder

   B)

   3-bit ripple carry adder using half adder / full adder.

Half - Adder

1

a) Implement Half Adder / full adder using gate level Modelling.

Half Adder.

Truth table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map

	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

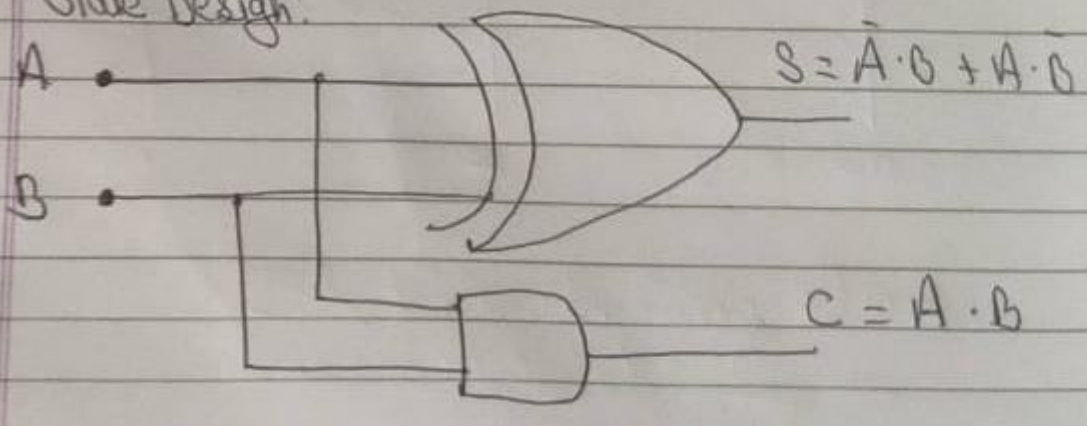
$S = A\bar{B} + \bar{A}B$

$= A \oplus B$

	$\bar{B}$	B
$\bar{A}$	0	0
A	0	1

$C = A \cdot B$

Gate Design:



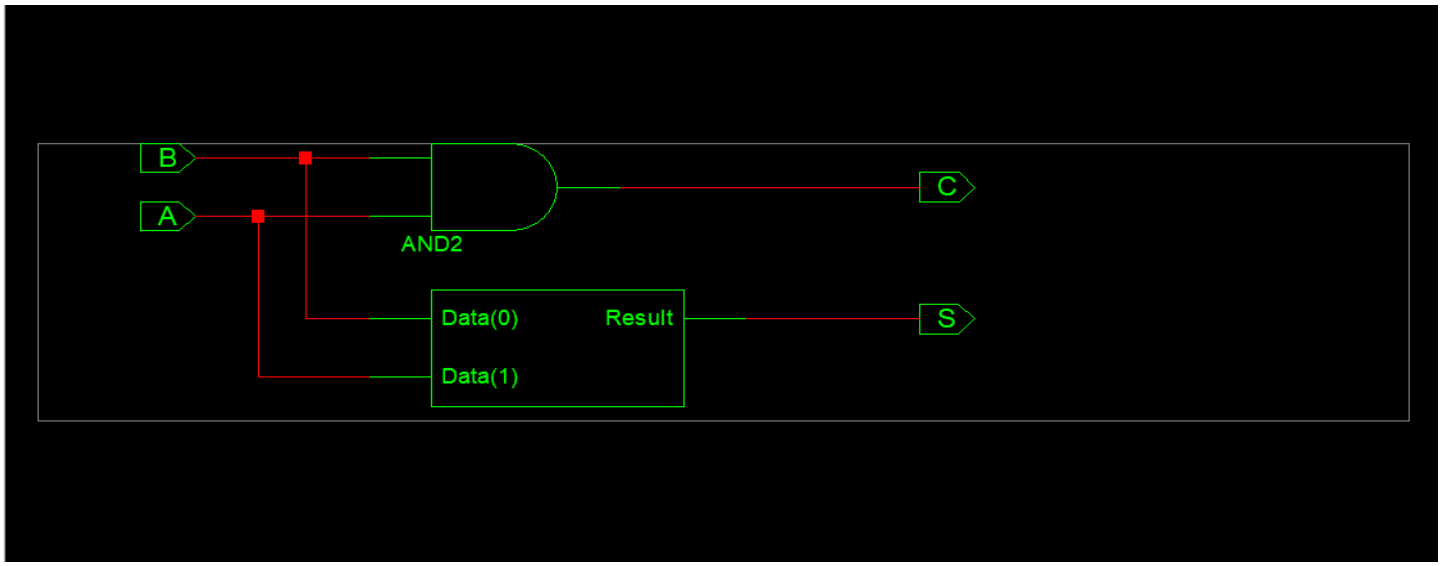
Verilog Code

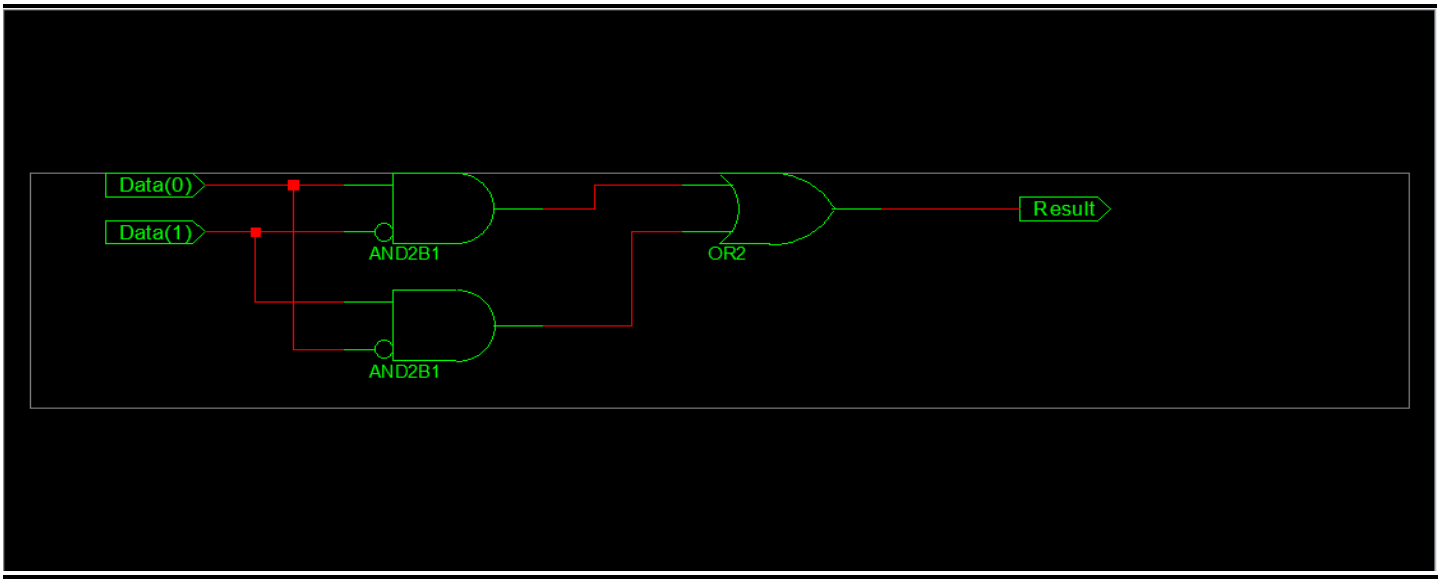
```
1  `timescale 1ns / 1ps
2
3  module mohit(
4     input A, B,
5     output S, C
6  );
7
8
9  xor (S, A, B);
10 and (C, A, B);
11
12 endmodule
13
14
```

Test bench code

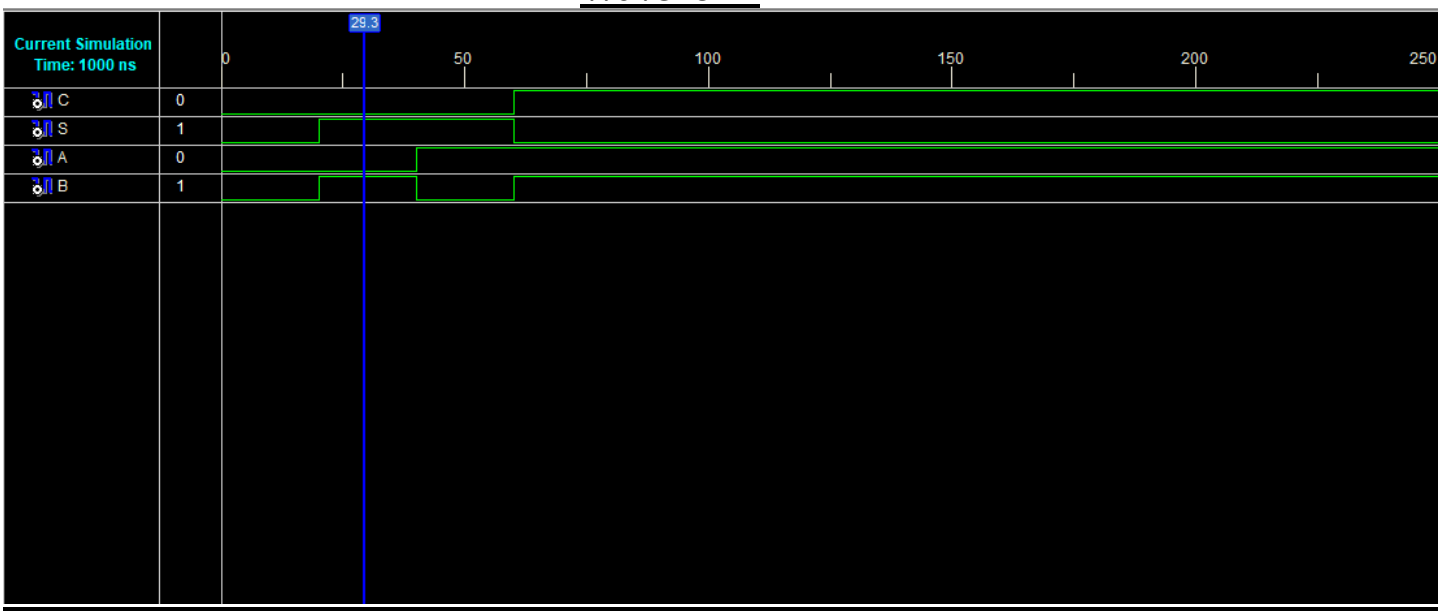
```
1  `timescale 1ns / 1ps
2
3  module rawat();
4  reg A, B;
5  wire S, C;
6
7  mohit(A, B, S, C);
8
9  initial
10    begin
11       A=0; B=0;  #20;
12       A=0; B=1;  #20;
13       A=1; B=0;  #20;
14       A=1; B=1;  #20;
15    end
16
17 endmodule
18
19
```

Circuit design





Wave form



Full adder

Full Adder:-

Truth Table.

A	B	C_{in}	S	C_o
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

K-Map.

Sum

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	(1)	0	(1)
A	(1)	0	(1)	0

$S = A \oplus B \oplus C$

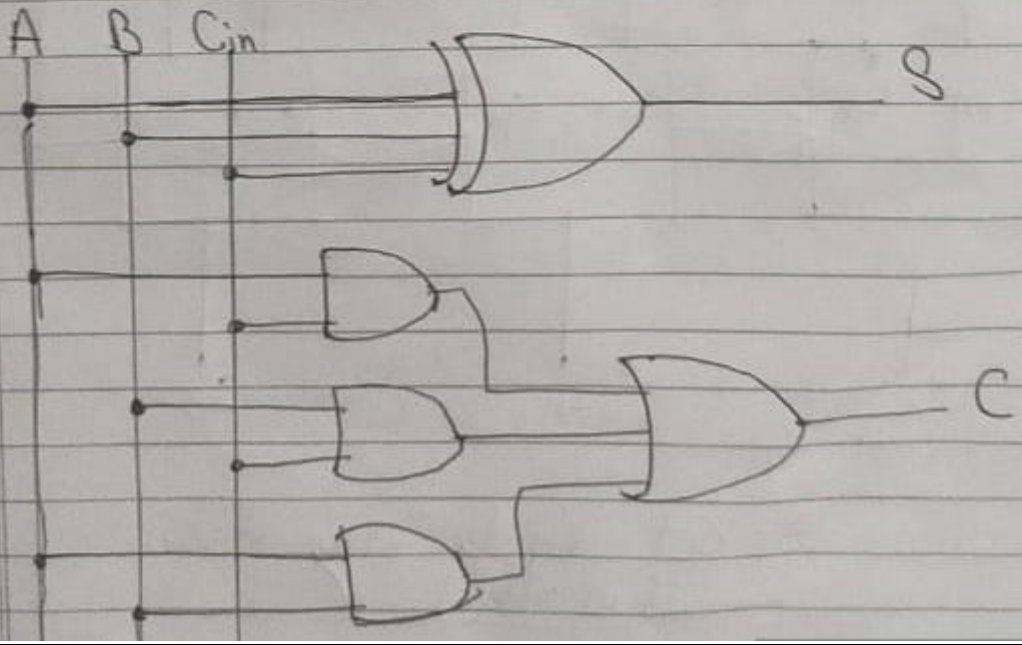
Carry

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	0	(1)	0
A	0	(1)	(1)	(1)

$C_o = A \cdot C_{in} + B \cdot C_{in} + A \cdot B$

or ~~C_{in}~~ $A \cdot B + C_{in} \cdot (A \oplus B)$

Gate Design



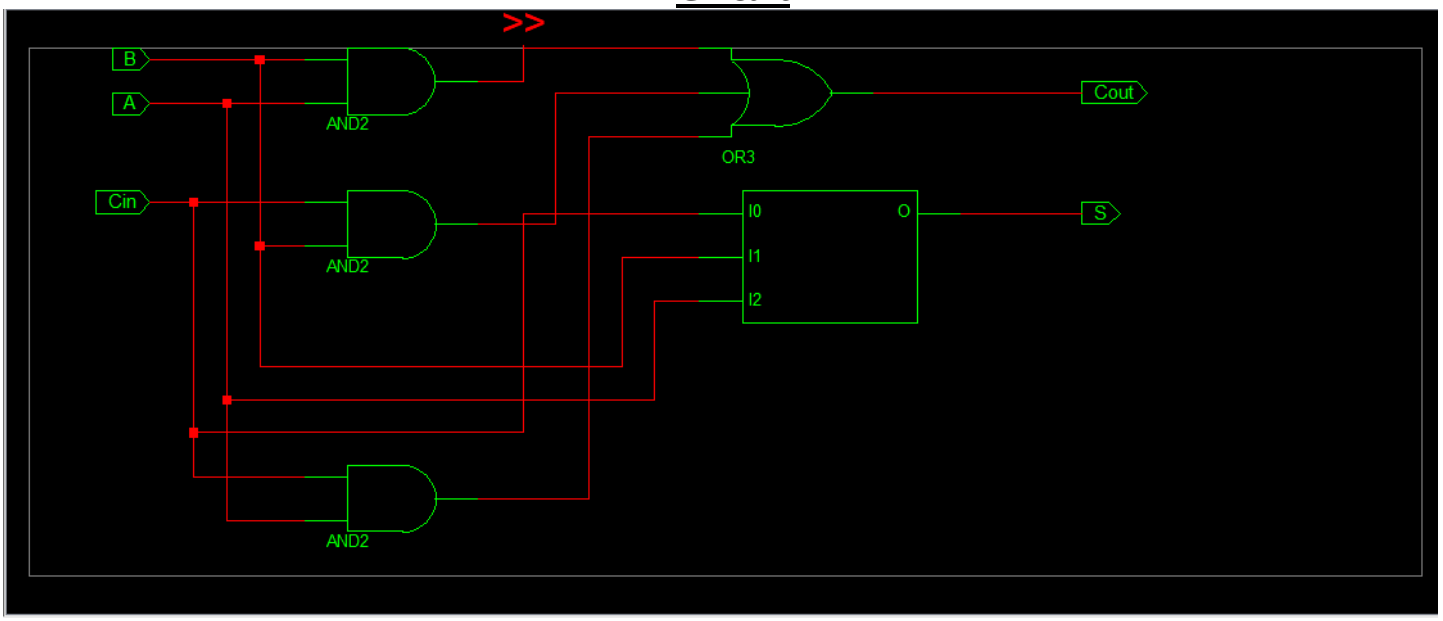
Verilog Code

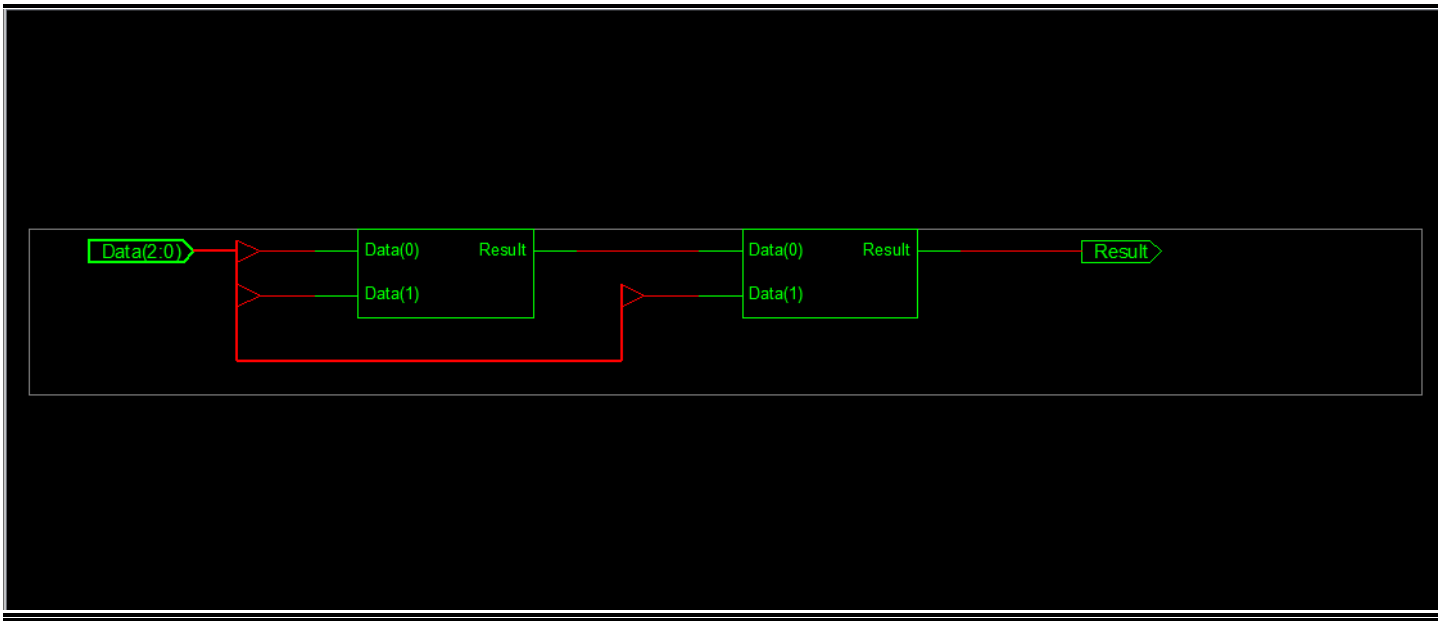
```
1  `timescale 1ns / 1ps
2
3  module mohit(
4     input A, B, Cin,
5     output S, Cout
6  );
7
8  wire w1, w2, w3;
9
10 and (w1, A, Cin);
11 and (w2, B, Cin);
12 and (w3, A, B);
13
14 xor (S, A, B, Cin);
15 or (Cout, w1, w2, w3);
16
17 endmodule
18
```

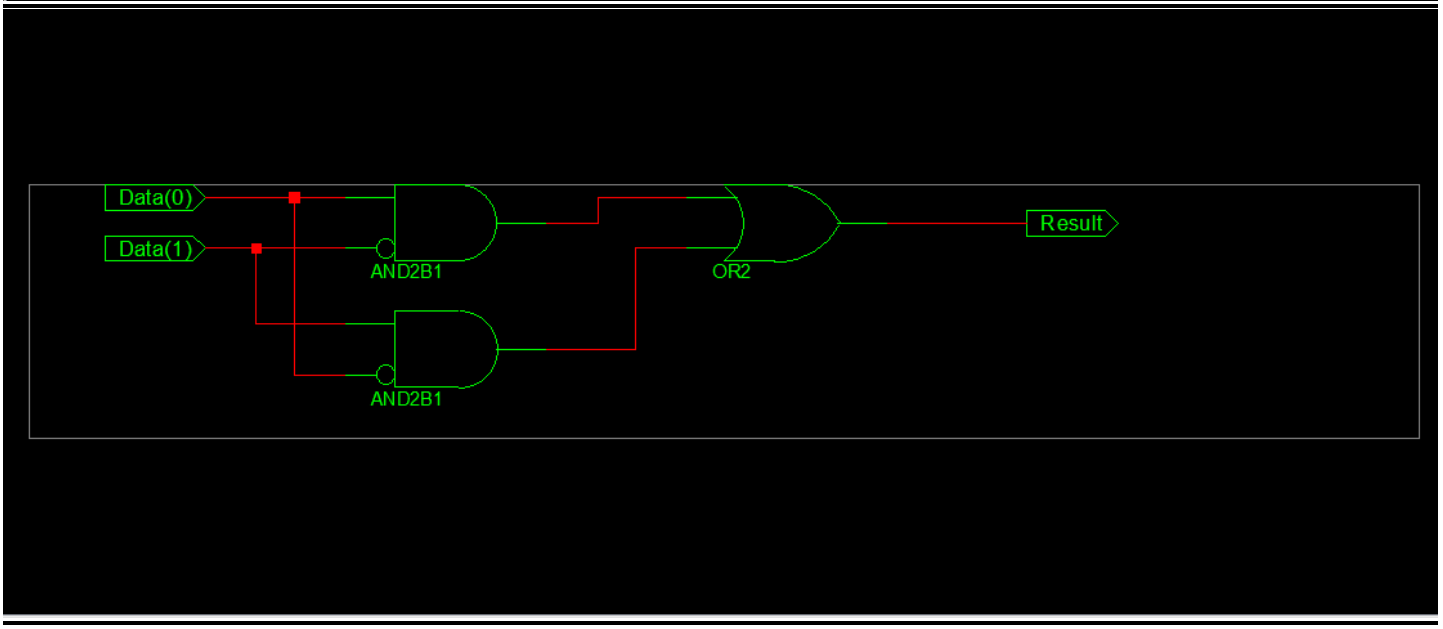
Test bench code

```
1  `timescale 1ns / 1ps
2
3  module rawat();
4  reg A, B, Cin;
5  wire S, Cout;
6
7  mohit(A, B, Cin, S, Cout);
8
9  initial
10    begin
11       A=0; B=0; Cin=0;  #20;
12       A=0; B=0; Cin=1;  #20;
13       A=0; B=1; Cin=0;  #20;
14       A=0; B=1; Cin=1;  #20;
15       A=1; B=0; Cin=0;  #20;
16       A=1; B=0; Cin=1;  #20;
17       A=1; B=1; Cin=0;  #20;
18       A=1; B=1; Cin=1;  #20;
19    end
20
21 endmodule
22
23
```

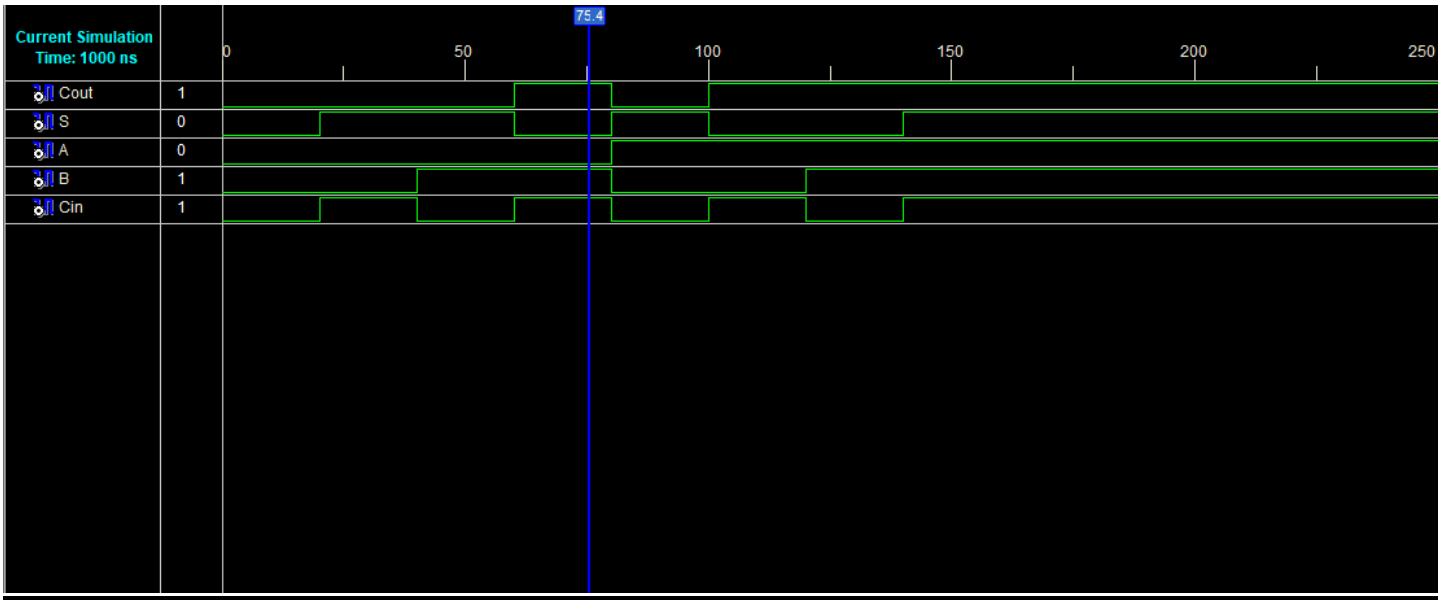
Circuit


>>
B
A
AND2
Cout
OR3
Cin
AND2
I0
O
S
I1
I2
AND2


Data(2:0)
Data(0)
Result
Data(1)
Data(0)
Result
Data(1)
Result


Data(0)
Data(1)
AND2B1
OR2
Result
AND2B1

Waveform


Current Simulation Time: 1000 ns
75.4
0
50
100
150
200
250
Cout
1
S
0
A
0
B
1
Cin
1

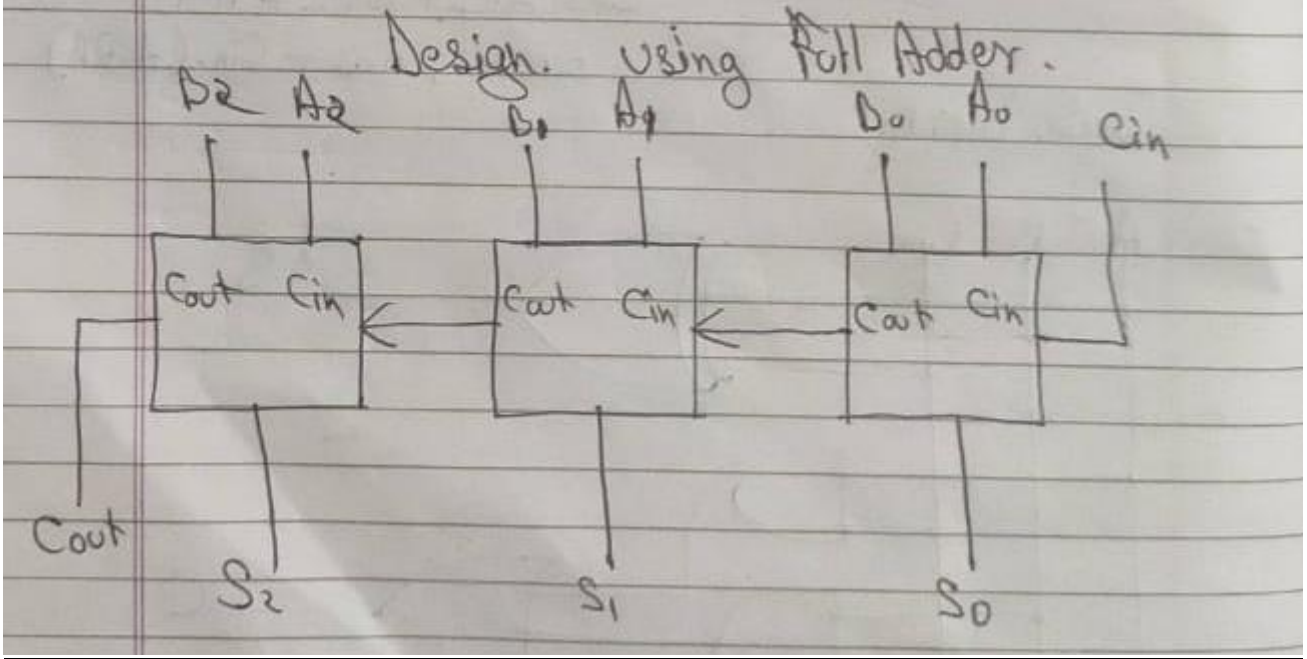
3-bit Ripple carry Adder (using Full Adder)

b) 3-bit ripple carry adder.

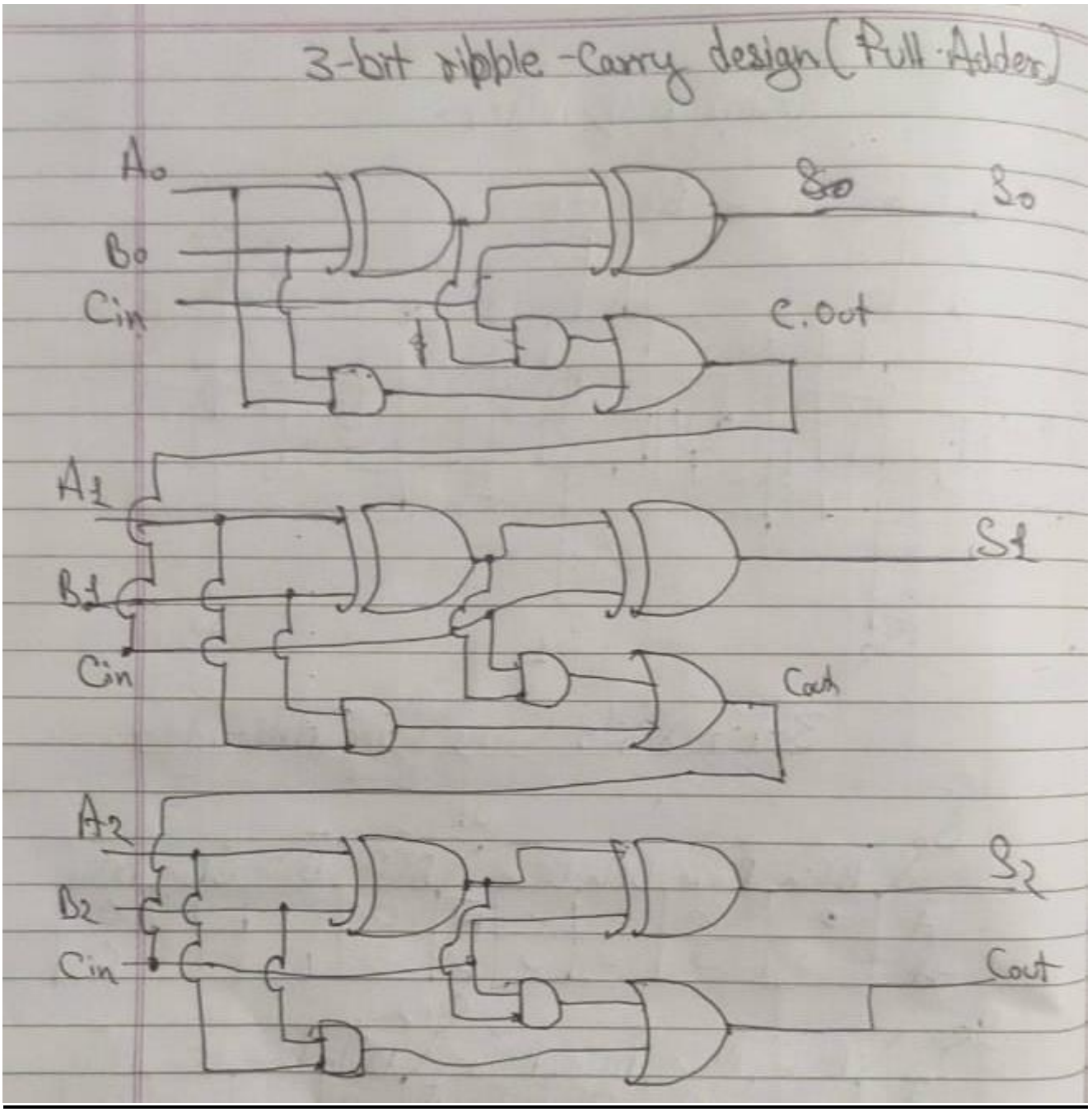
Truth Table

Carry	A			B			Sum			Carry
Cin	A_2	A_1	A_0	B_2	B_1	B_0	S_2	S_1	S_0	Cout
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1	0	0	0
0	0	1	1	0	1	1	1	1	0	0
0	1	0	0	1	0	0	0	0	0	1
0	1	0	1	1	0	1	0	1	0	1
0	1	1	0	1	1	0	1	0	0	1
0	1	1	1	1	1	1	1	1	0	1

Design using Full Adder.



Circuit


3-bit ripple-Carry design (Full Adder)
A0
B0
Cin
S0
C.out
A1
B1
Cin
S1
Cout
A2
B2
Cin
S2
Cout

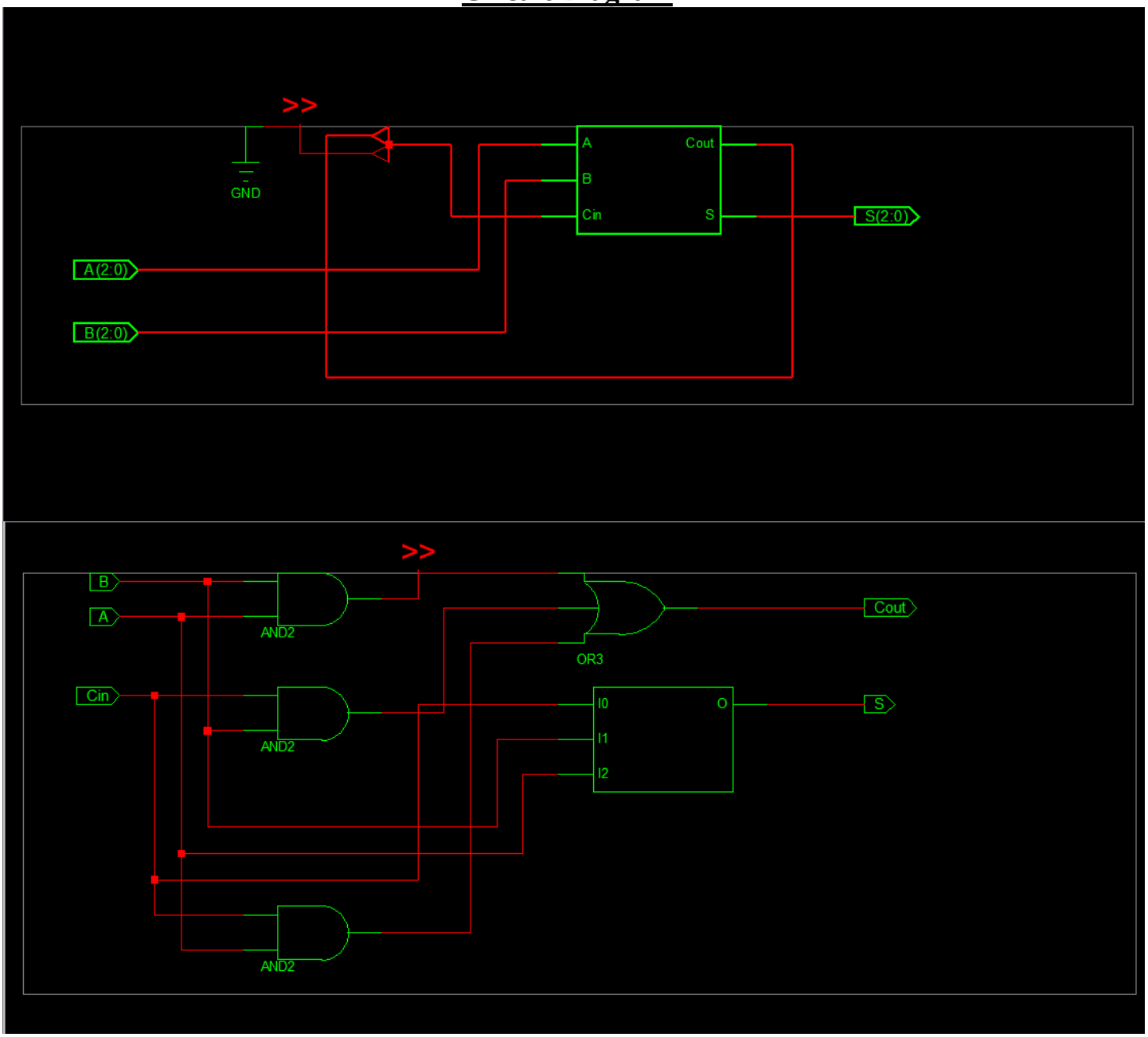
Verilog Code

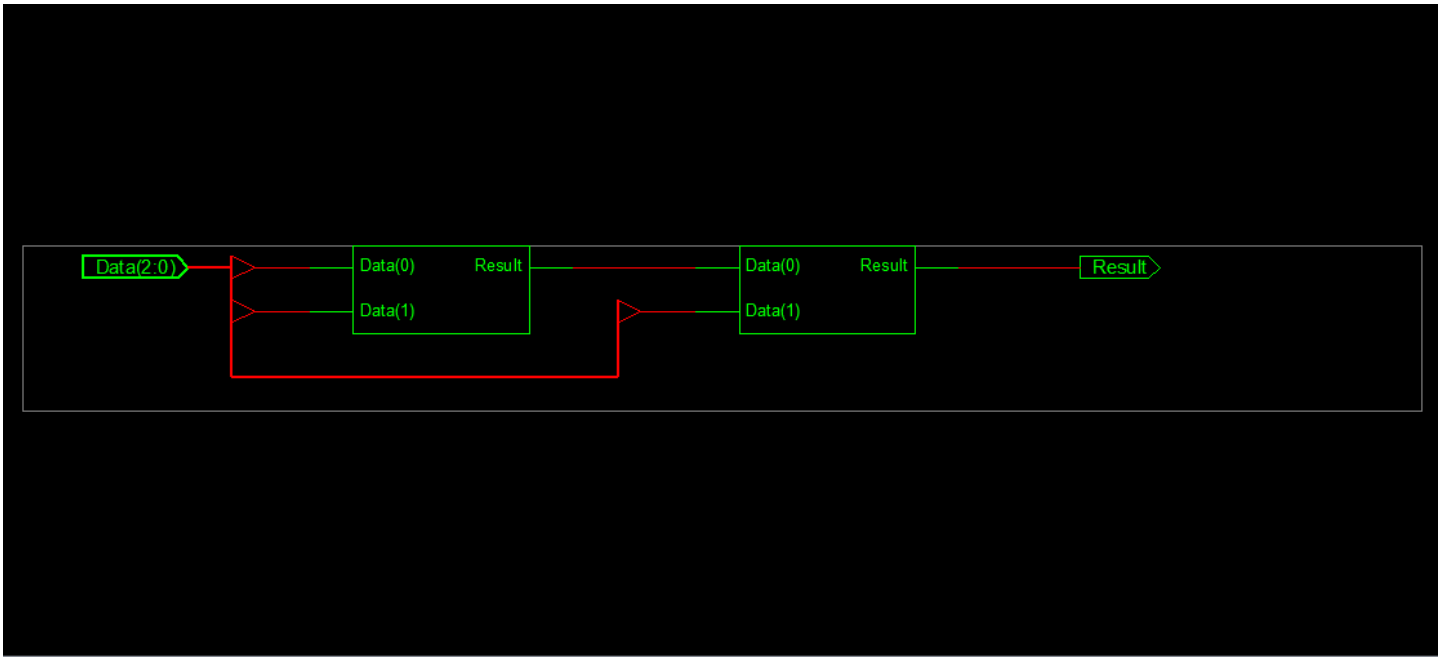
```
1  `timescale 1ns / 1ps
2  module Full_Adder(A, B, Cin, S, Cout);
3  input A, B, Cin;
4  output S, Cout;
5  wire w1, w2, w3;
6  and (w1, A, Cin);
7  and (w2, B, Cin);
8  and (w3, A, B);
9  xor (S, A, B, Cin);
10 or (Cout, w1, w2, w3);
11 endmodule
12 module mohit(A, B, S, Cout);
13 input [2:0] A;
14 input [2:0] B;
15 output [2:0] S;
16 output Cout;
17 wire c0, c1;
18 Full_Adder A1(A[0], B[0], 1'b0, S[0], c0);
19 Full_Adder A2(A[1], B[1], c0,   S[1], c1);
20 Full_Adder A3(A[2], B[2], c1,   S[2], Cout);
21 endmodule
22
```

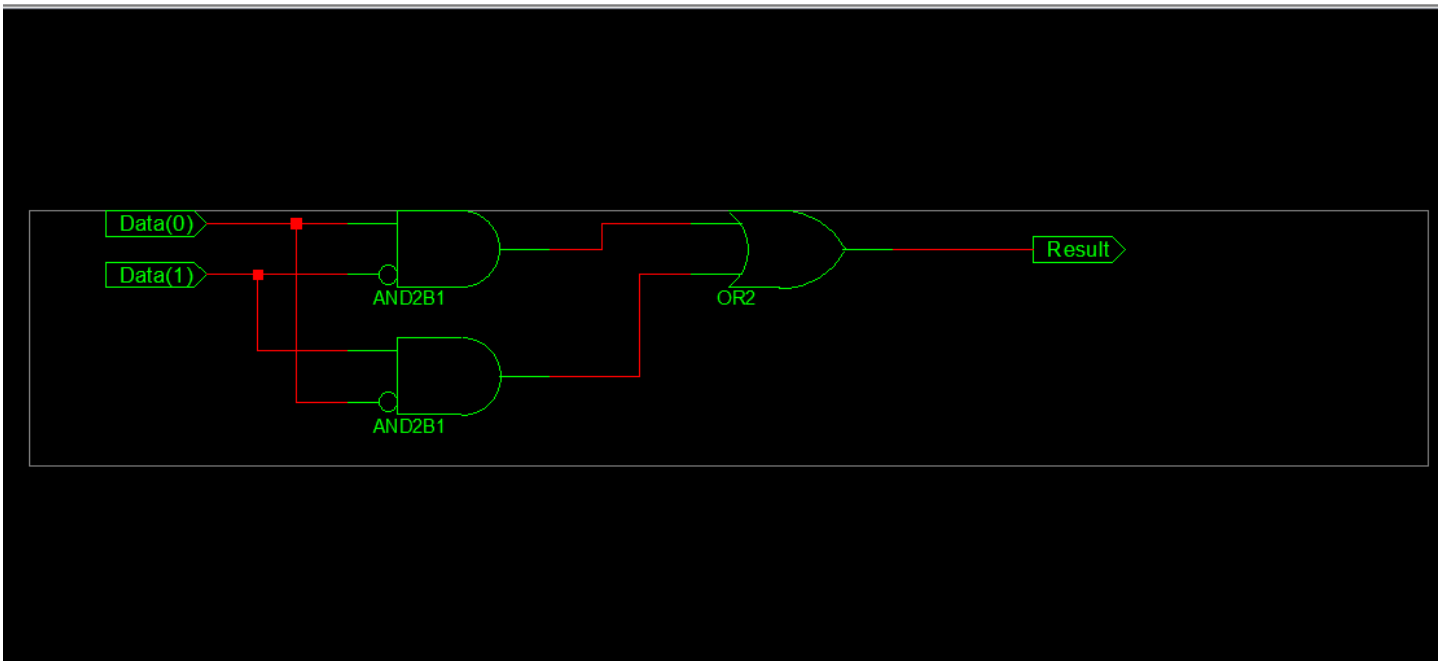
Test bench Program

```
1  `timescale 1ns / 1ps
2
3  module mytb();
4  reg [2:0] A;
5  reg [2:0] B;
6  wire [2:0] S;
7  wire Cout;
8
9  mohit ad(A, B, S, Cout);
10
11 initial begin
12 A=3'b000; B=3'b000; #100;
13 A=3'b001; B=3'b001; #100;
14 A=3'b010; B=3'b010; #100;
15 A=3'b011; B=3'b011; #100;
16 A=3'b100; B=3'b100; #100;
17 A=3'b101; B=3'b101; #100;
18 A=3'b110; B=3'b110; #100;
19 A=3'b111; B=3'b111; #100;
20 end
21 endmodule
22
```

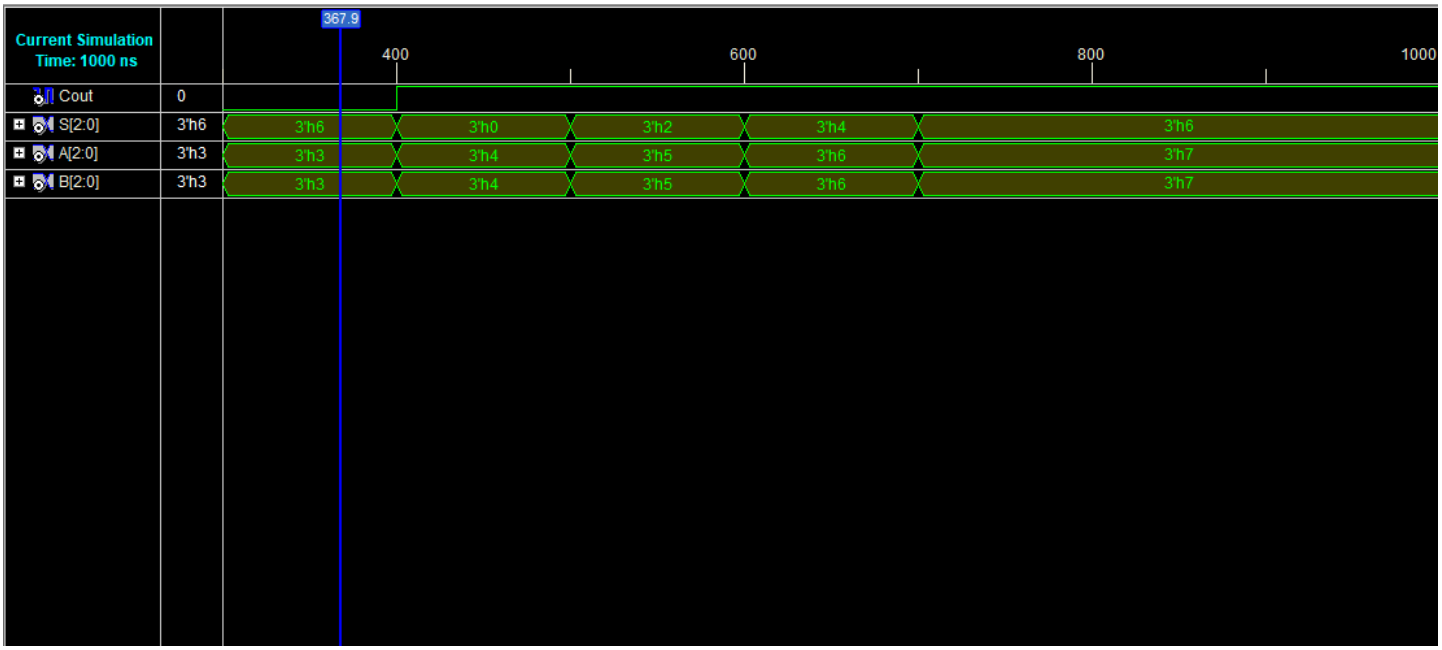
Circuit Diagram




Data(2:0)
Data(0)
Result
Data(1)
Data(0)
Result
Data(1)
Result

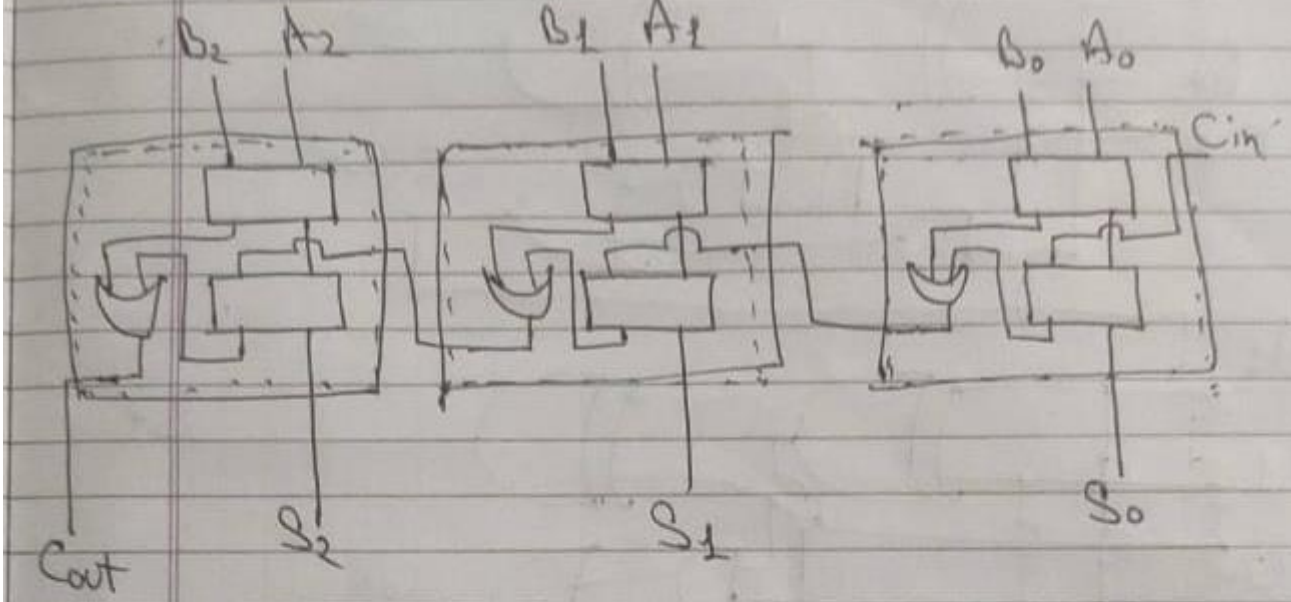

Data(0)
Data(1)
AND2B1
OR2
Result
AND2B1

Waveform

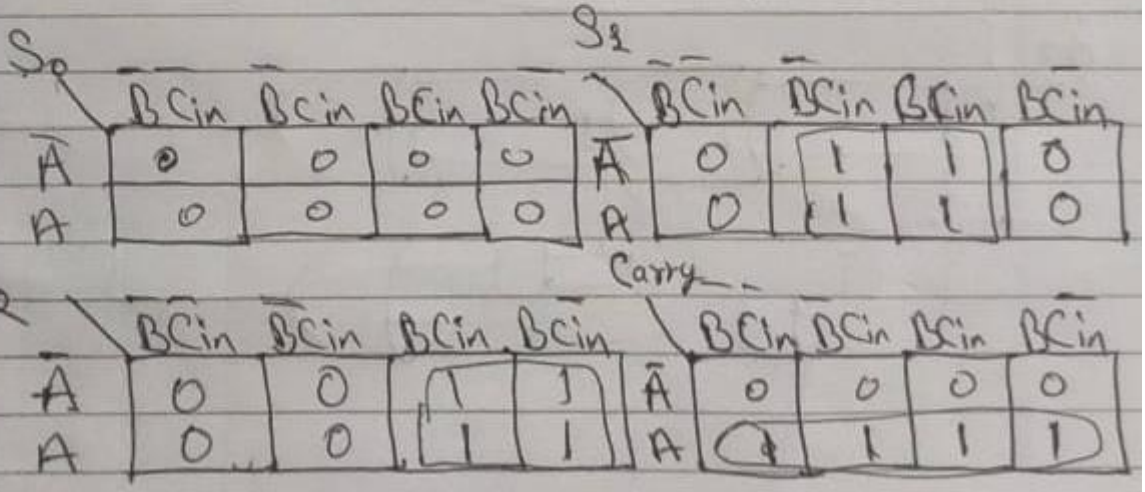


3-bit Ripple Carry Adder(Using Half Adder)

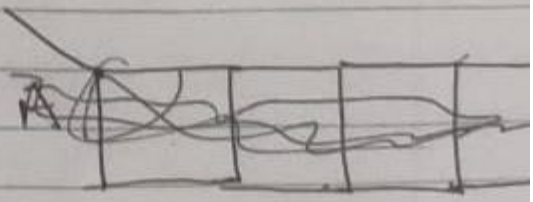
Using half-Adder.



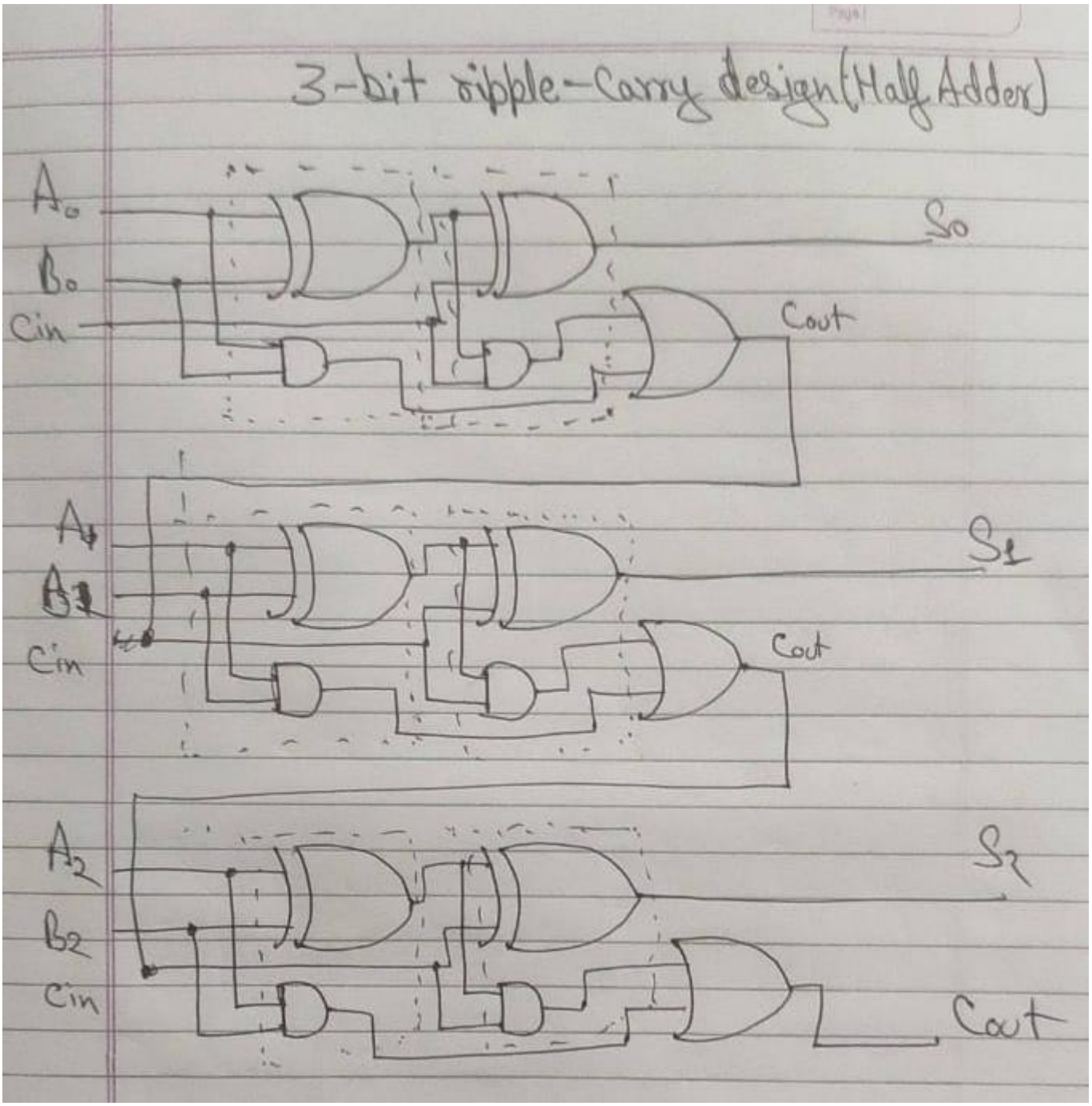
3-bit ripple Carry using Half-adder.



$S_0 = 0$
$S_1 = C_{in}$
$S_2 = B$
CarryOut = A



Circuit Diagram



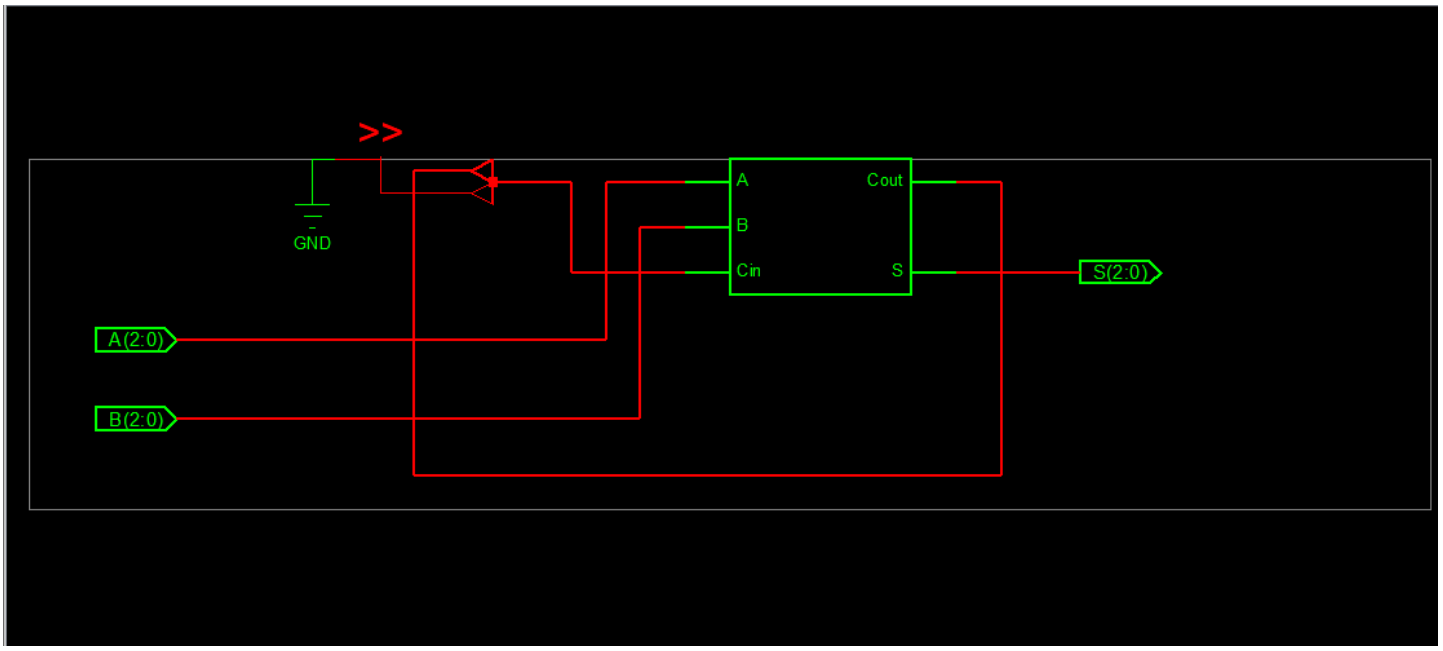
Verilog code

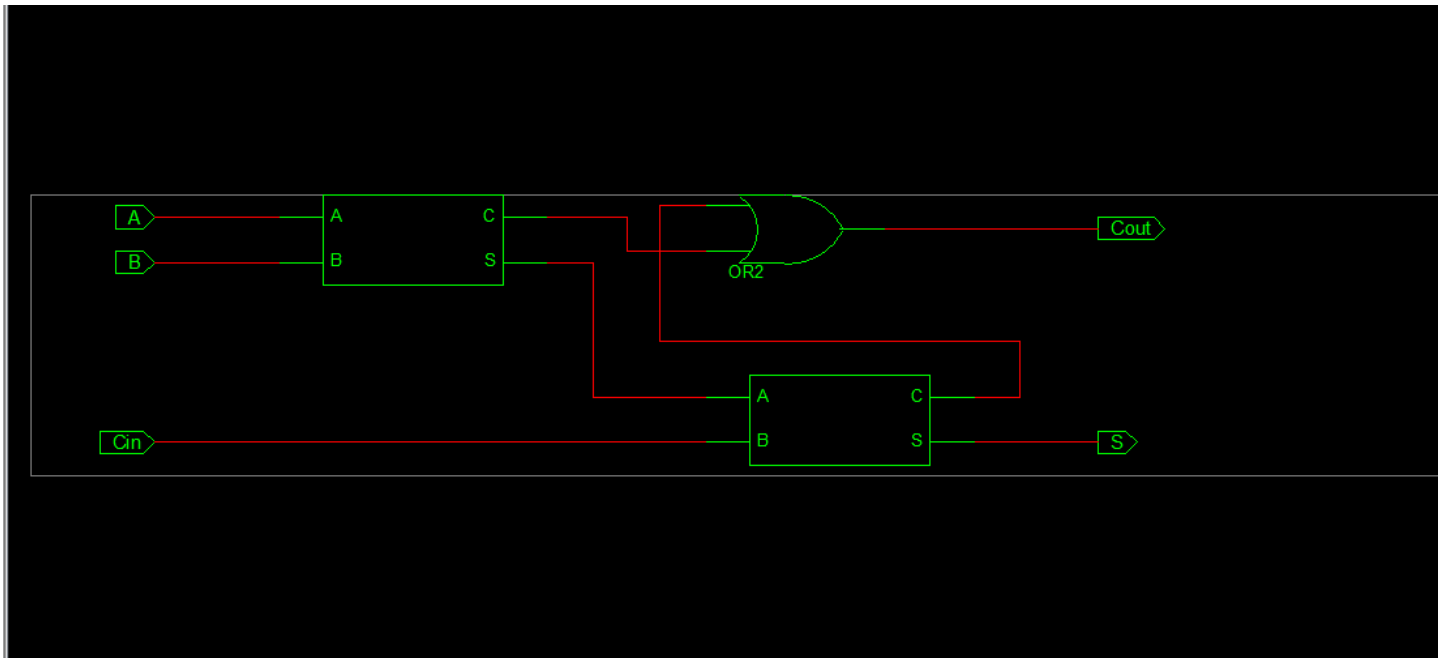
```
1  `timescale 1ns / 1ps
2  module Half_Adder(A, B, S, C);
3  input A;
4  input B;
5  output S;
6  output C;
7  xor (S, A, B);
8  and (C, A, B);
9  endmodule
10 module Full_Adder(A, B, Cin, S, Cout);
11 input A, B, Cin;
12 output S, Cout;
13 wire c1, c2;
14 Half_Adder h1(A, B, s1, c1);
15 Half_Adder h2(s1, Cin, S, c2);
16 or o1(Cout, c1, c2);
17 endmodule
18 module mohit(
19    input [2:0] A,
20    input [2:0] B,
21    output [2:0] S,
22    output Cout
23 );
24
25 wire c0, c1;
26 Full_Adder A1(A[0], B[0], 1'b0, S[0], c0);
27 Full_Adder A2(A[1], B[1], c0,   S[1], c1);
28 Full_Adder A3(A[2], B[2], c1,   S[2], Cout);
29 endmodule
30
```

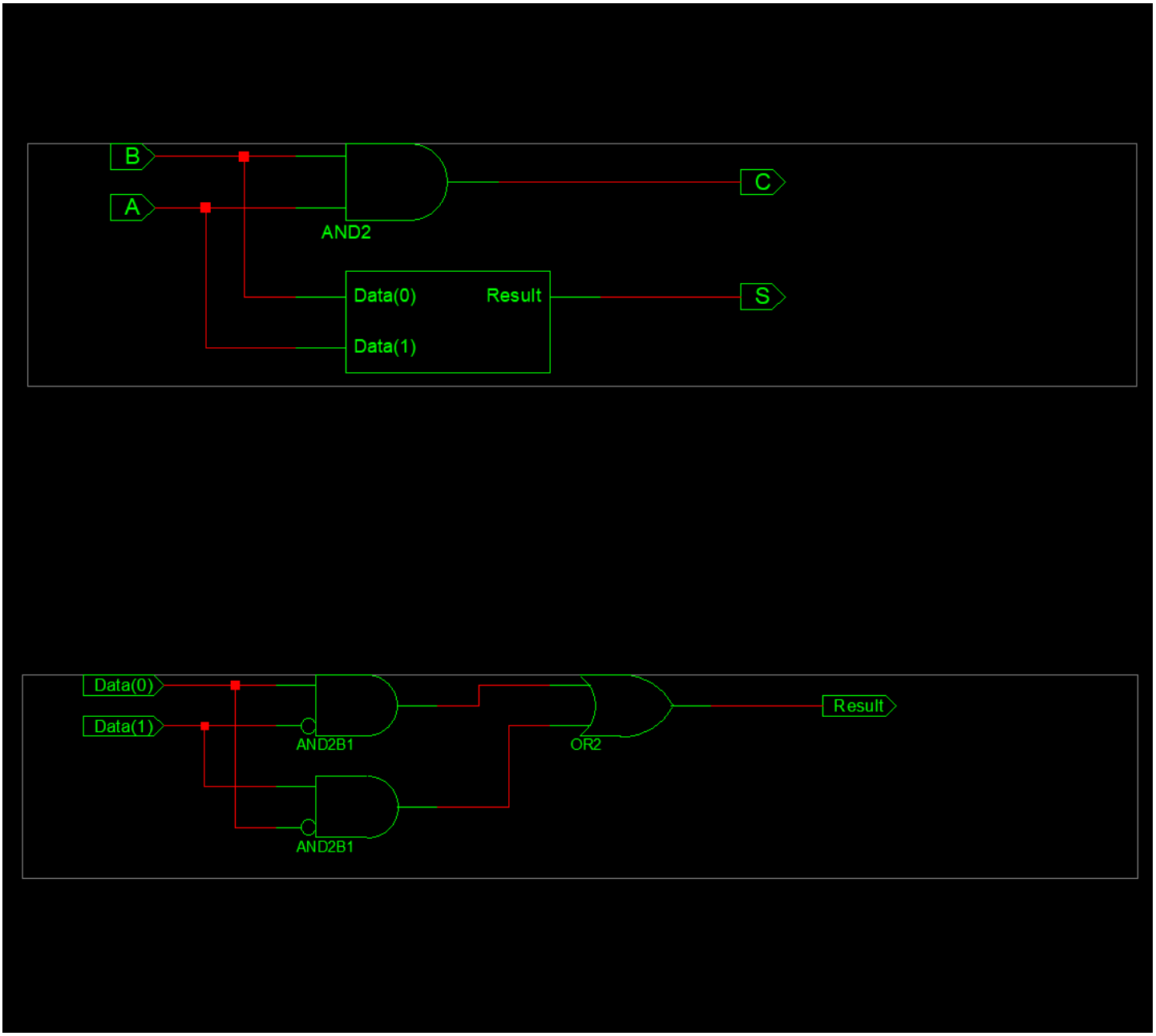
Test bench code

```
1  `timescale 1ns / 1ps
2
3  module myTB();
4  reg [2:0] A;
5  reg [2:0] B;
6  wire [2:0] S;
7  wire Cout;
8
9  mohit adH(A, B, S, Cout);
10
11 initial begin
12 A=3'b000; B=3'b000; #100;
13 A=3'b001; B=3'b001; #100;
14 A=3'b010; B=3'b010; #100;
15 A=3'b011; B=3'b011; #100;
16 A=3'b100; B=3'b100; #100;
17 A=3'b101; B=3'b101; #100;
18 A=3'b110; B=3'b110; #100;
19 A=3'b111; B=3'b111; #100;
20 end
21 endmodule
22
```

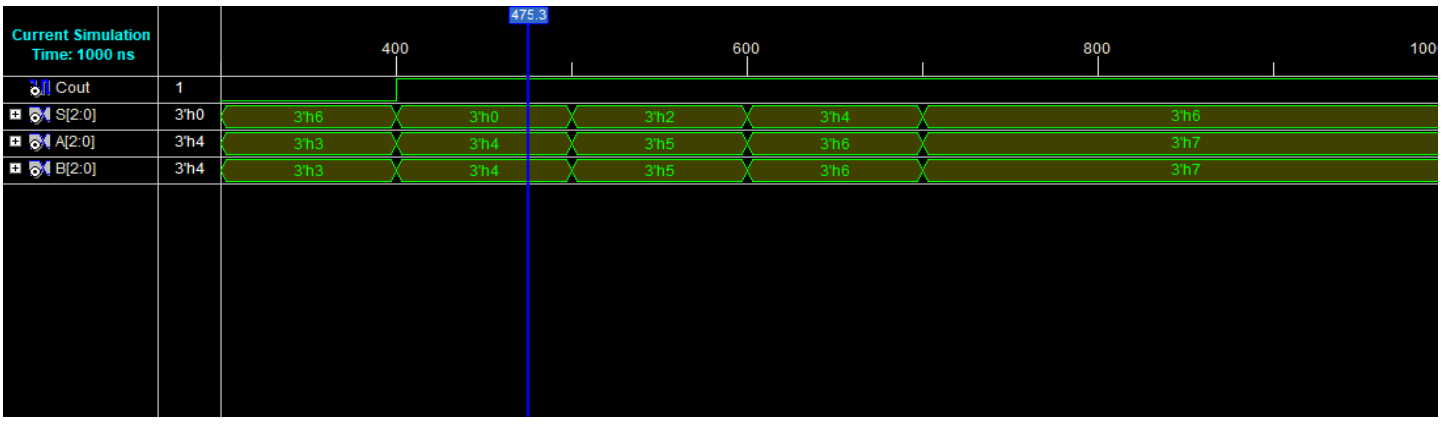
Circuit Diagram


>>
GND
A
B
Cin
Cout
S
A(2:0)
B(2:0)
S(2:0)


A
B
A
B
C
S
OR2
Cout
Cin
A
B
C
S
S



Waveform



END