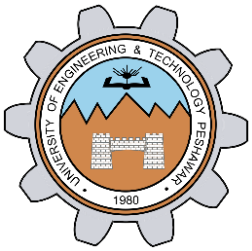
INTRODUCTION TO VERILOG

LAB # 06



Fall 2023

CSE-304L Computer Organization and Architecture Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Dr. Bilal Habib

Date:

9th November 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

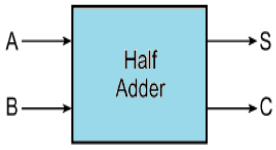
ASSESSMENT RUBRICS COA LABS

LAB REPORT ASSESSMENT				
Criteria	**Excellent**	**Average**	**Nill**	**Marks Obtained**
1. Objectives of Lab	All objectives of lab are properly covered [Marks 10]	Objectives of lab are partially covered [Marks 5]	Objectives of lab are not shown [Marks 0]	
2. MIPS instructions with Comments and proper indentations.	All the instructions are well written with comments explaining the code and properly indented [Marks 20]	Some instructions are missing are poorly commented code [Marks 10]	The instructions are not properly written [Marks 0]	
3. Simulation run without error and warnings	The code is running in the simulator without any error and warnings [Marks 10]	The code is running but with some warnings or errors. [Marks 5]	The code is written but not running due to errors [Marks 0]	
4. Procedure	All the instructions are written with proper procedure [Marks 20]	Some steps are missing [Marks 10]	steps are totally missing [Marks 0]	
5. OUTPUT	Proper output of the code written in assembly [Marks 20]	Some of the outputs are missing [Marks 10]	No or wrong output [Marks 0]	
6. Conclusion	Conclusion about the lab is shown and written [Marks 20]	Conclusion about the lab is partially shown [Marks 10]	Conclusion about the lab is not shown[Marks0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained:_____ Instructor Signature: _____				

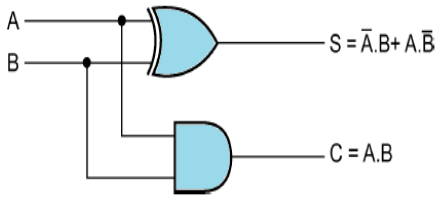
Task 1:

Implement half adder in Verilog using gate level modeling.

Block and circuit diagram of half adder



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



Code:

DUT Code:

4bit_adder.v | half_adder.v

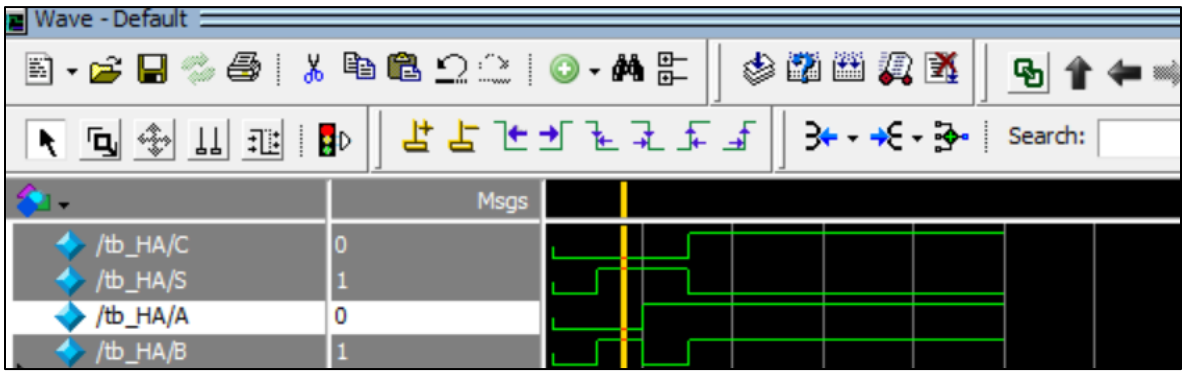
```
1  module half_adder(A,B,S,C);
2
3      input A,B;
4      output C,S;
5      and a(C, A,B);
6      xor x(S, A,B);
7
8  endmodule
```

Test bench Code:

4bit_adder.v | half_adder.v | test_bench_HA.v

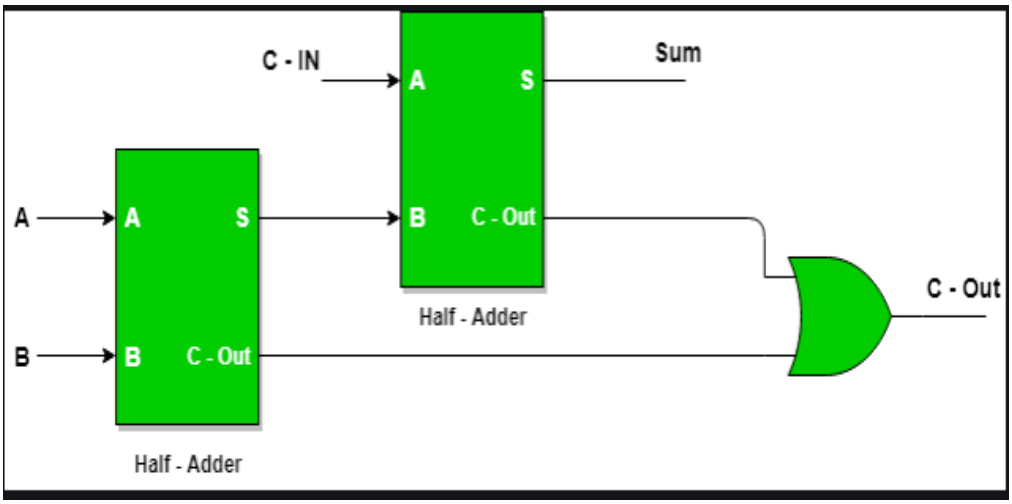
```
1  module tb_HA();
2      wire C, S;
3      reg A, B;
4      half_adder my_half_adder(A, B, C , S);
5
6  initial begin
7      A = 0;
8      B = 0;
9      #10;
10
11     A = 0;
12     B = 1;
13     #10;
14
15     A = 1;
16     B = 0;
17     #10;
18
19     A = 1;
20     B = 1;
21     #10;
22     end
23 endmodule
```

Output:



Task 2:

Implement full adder using two half adder. (use the above half adder to create full adder)



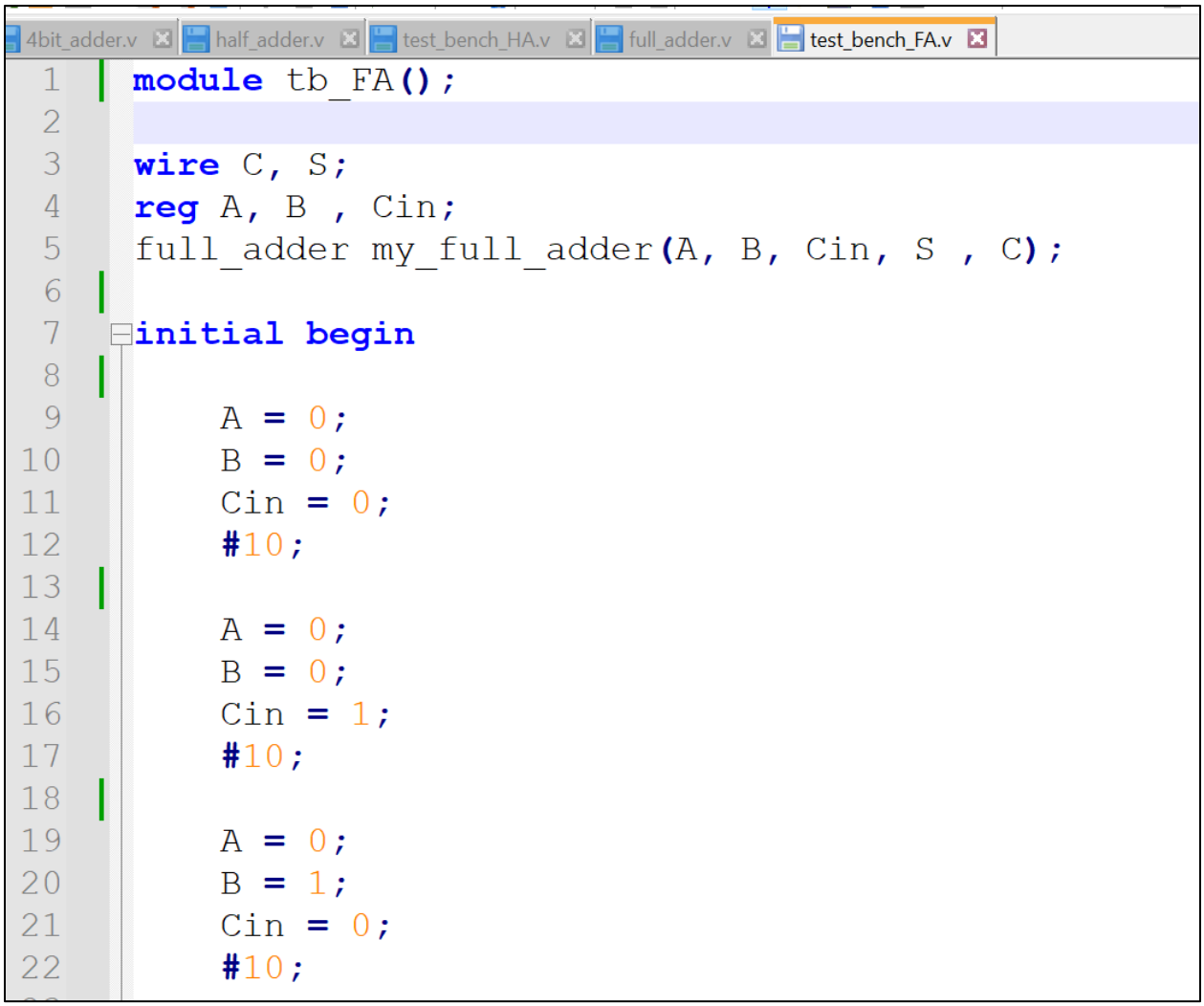
Code:

DUT Code:

4bit_adder.v | half_adder.v | test_bench_HA.v | full_adder.v

```
1   module full_adder(A,B,Cin,S,Cout);
2
3       input A,B,Cin;
4       output S,Cout;
5       wire sum1, carry1, carry2;
6
7       half_adder ha1(A, B, sum1, carry1);
8       half_adder ha2(Cin, sum1, S, carry2);
9       or o(Cout, carry1,carry2);
10
11  endmodule
12
```

Test bench Code:

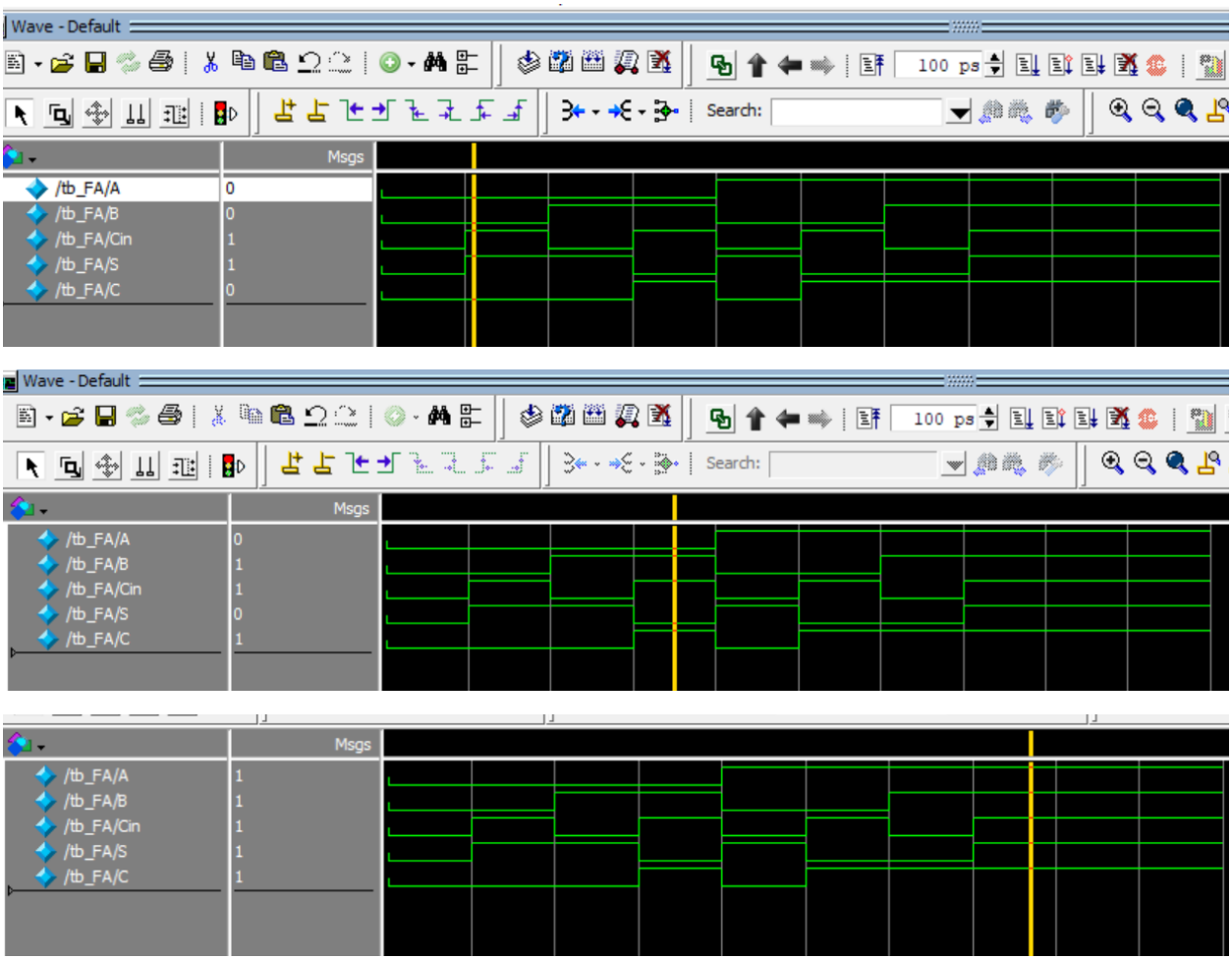



```
1  module tb_FA();
2
3  wire C, S;
4  reg A, B , Cin;
5  full_adder my_full_adder(A, B, Cin, S , C);
6
7  initial begin
8
9      A = 0;
10     B = 0;
11     Cin = 0;
12     #10;
13
14     A = 0;
15     B = 0;
16     Cin = 1;
17     #10;
18
19     A = 0;
20     B = 1;
21     Cin = 0;
22     #10;
```

4bit_adder.v | half_adder.v | test_bench_HA.v | full_adder.v | test_bench_FA.v

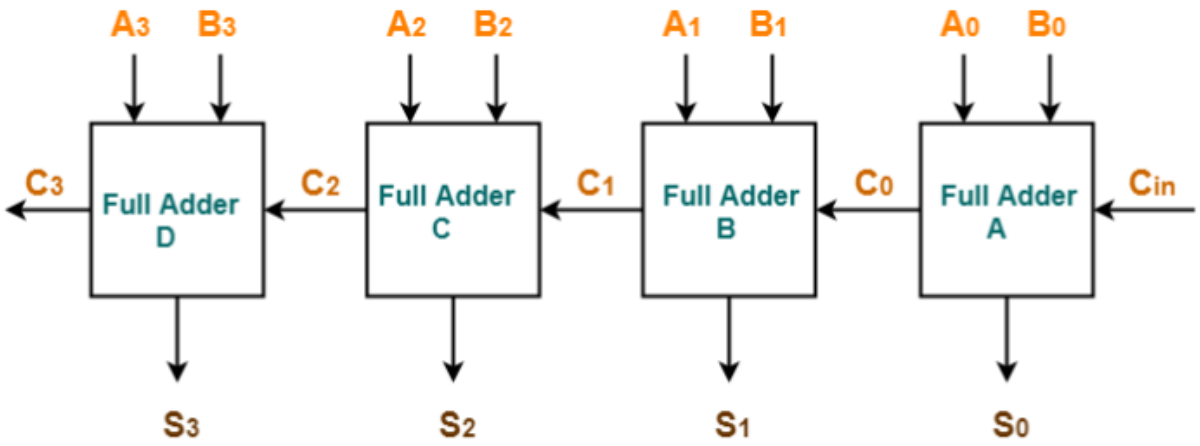
```
23
24      A = 0;
25      B = 1;
26      Cin = 1;
27      #10;
28
29      A = 1;
30      B = 0;
31      Cin = 0;
32      #10;
33
34      A = 1;
35      B = 0;
36      Cin = 1;
37      #10;
38
39      A = 1;
40      B = 1;
41      Cin = 0;
42      #10;
43
44      A = 1;
45      B = 1;
46      Cin = 1;
47      #10;
48      end
49
50  endmodule
51
```

Output:


Wave - Default
100 ps
Search:
Msgs
/tb_FA/A
/tb_FA/B
/tb_FA/Cin
/tb_FA/S
/tb_FA/C
0
0
1
1
0
Wave - Default
100 ps
Search:
Msgs
/tb_FA/A
/tb_FA/B
/tb_FA/Cin
/tb_FA/S
/tb_FA/C
0
1
1
0
1
Msgs
/tb_FA/A
/tb_FA/B
/tb_FA/Cin
/tb_FA/S
/tb_FA/C
1
1
1
1
1

Task 3:

Write a Verilog code for 4 bit ripple carry adder.



4-bit Ripple Carry Adder

Code:

DUT Code:

half_adder.v | test_bench_HA.v | full_adder.v | test_bench_FA.v | 4bit_adder.v | test_bench_four_bit_adder.v

```
1   module four_bit_adder(in1, in2, Cin, S, Cout);
2
3       //input A0,A1,A2,A3, B0, B1, B2, B3,Cin;
4       input [3:0] in1, in2;
5       input Cin;
6       output [3:0] S;
7       output Cout;
8       wire carry1, carry2, carry3;
9
10      full_adder f1(in1[0], in2[0], Cin, S[0], carry1);
11      full_adder f2(in1[1], in2[1], carry1, S[1], carry2);
12      full_adder f3(in1[2], in2[2], carry2, S[2], carry3);
13      full_adder f4(in1[3], in2[3], carry3, S[3], Cout);
14
15
16  endmodule
```

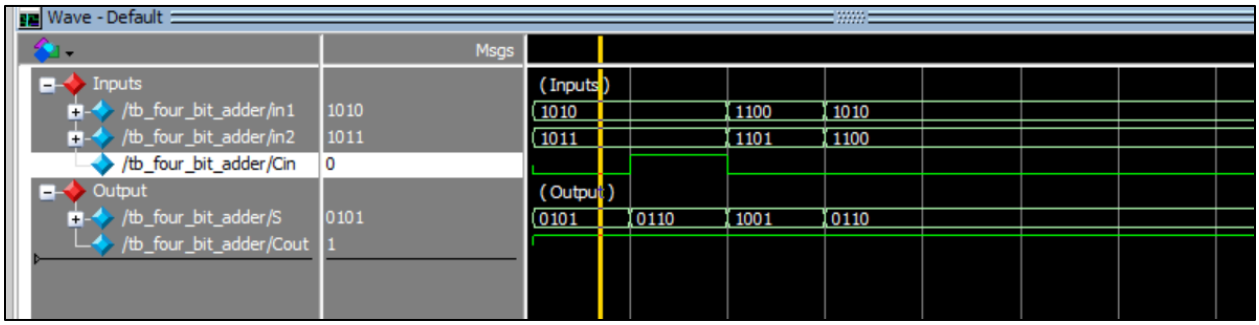
Test bench Code:

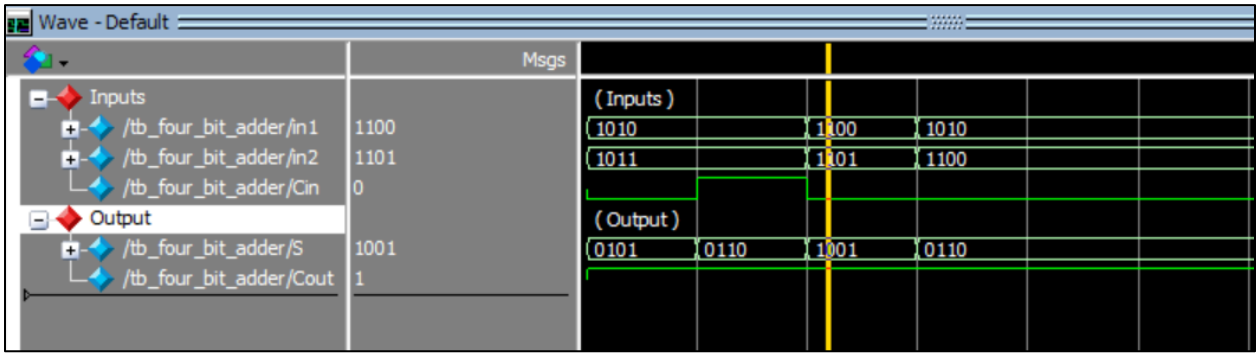
half_adder.v | test_bench_HA.v | full_adder.v | test_bench_FA.v | 4bit_adder.v | test_bench_four_bit_adder.v

```
module tb_four_bit_adder();
    reg [3:0] in1, in2;
    wire [3:0] S;
    reg Cin;
    wire Cout;

    four_bit_adder my_four_bit_adder(in1, in2,Cin, S, Cout);


initial begin
    in1 = 4'hA;
    in2 = 4'hB;
    Cin = 0;
    #10;

    in1 = 4'hA;
    in2 = 4'hB;
    Cin = 1;
    #10;

    in1 = 4'hC;
    in2 = 4'hD;
    Cin = 0;
```

```
    Cin = 0;
    #10;

    in1 = 4'hA;
    in2 = 4'hC;
    Cin = 0;
    #10;
    end

endmodule
```

Output:





Conclusion:

In this lab, I learned about the basics of Verilog (Hardware Descriptive Language) in using ModelSim.