



Department of Electrical Engineering

Faculty Member: Mr. Arshad Nazir

Dated: 5th October 2023

Semester: Fall 2023

Section: BEE-14-D

Group No.: 10

EE-221: Digital Logic Design

Lab 4: Design and Implementation of Simple Practical Circuits with Full Adder

Name	Reg. No	PLO4/CLO4 Viva / Lab Performance 5 Marks	PLO4/CLO4 Analysis of data in Lab Report 5 Marks	PLO5/CLO5 Modern Tool Usage 5 Marks	PLO8/CLO6 Ethics and Safety 5 Marks	PLO9/CLO7 Individual and Team Work 5 Marks	Total marks Obtained 25 Marks
Arooj Fatima	423365						
Irfan Farooq	412564						
Haseeb Umar	427442						
Ahmad Nasir	409959						



Lab 4: Design and Implementation of Simple Practical Circuits

This Lab has been divided into two parts.

In first part you are required to design and implement simple practical circuits.
In the second part you are required to write Verilog code for the practical circuit.

Objectives:

- ✓ Design the Simple practical circuits using digital logic
- ✓ Hardware Implementation of Basic Logic Circuits
- ✓ Verilog coding of the practical circuit

Lab Instructions

- ✓ This lab activity comprises three parts, namely Pre-lab, Lab tasks, and Post-Lab Viva session.
- ✓ The lab report will be uploaded on LMS three days before scheduled lab date. The students will get hard copy of lab report, complete the Pre-lab task before coming to the lab and deposit it with teacher/lab engineer for necessary evaluation. Alternately each group to upload completed lab report on LMS for grading.
- ✓ The students will start lab task and demonstrate design steps separately for step-wise evaluation (course instructor/lab engineer will sign each step after ascertaining functional verification)
- ✓ Remember that a neat logic diagram with pins numbered coupled with nicely patched circuit will simplify trouble-shooting process.
- ✓ After the lab, students are expected to unwire the circuit and deposit back components before leaving.
- ✓ The students will complete lab task and submit complete report to Lab Engineer before leaving lab.
- ✓ There are related questions at the end of this activity. Give complete answers.

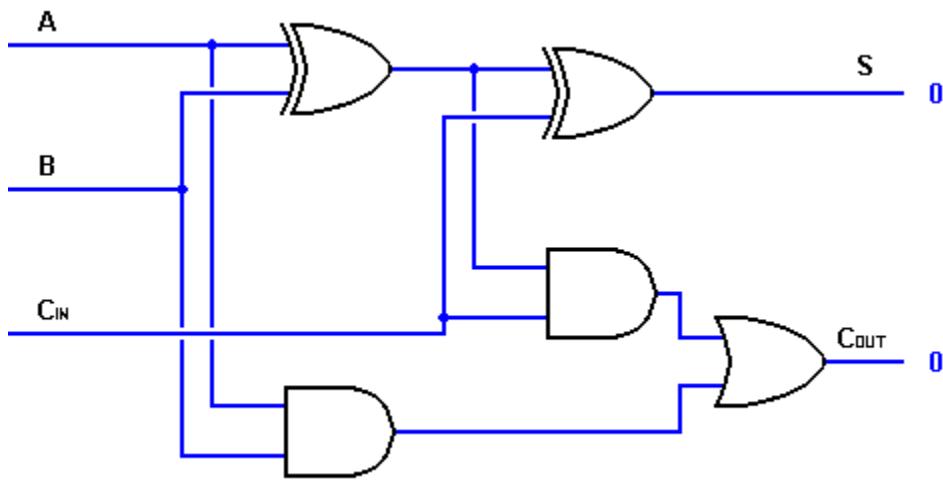


Lab Task 1:

(5 Marks)

Implement the Full Adder Circuit given below, make its truth table and Schematic Diagram. Show your results to the lab engineer.

Logic Diagram:



Algebraic Expressions for S and Cout are:

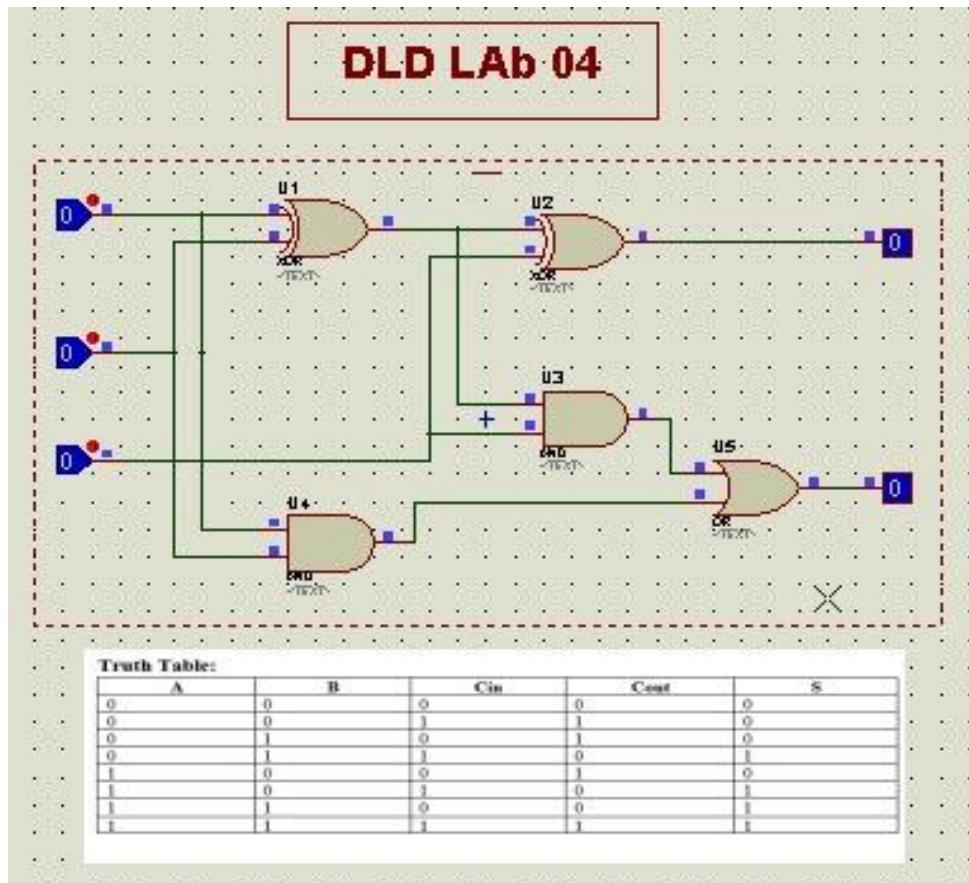
$$[S = ((A \text{ XOR } B) \text{ XOR } C_{in})], [C_{out} = ((A \text{ XOR } B) \text{ AND } C_{in}) \text{ OR } (A \text{ AND } B)]$$

Truth Table:

A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Proteus Schematic:



Hardware:





Lab Task 2:

(5 Marks)

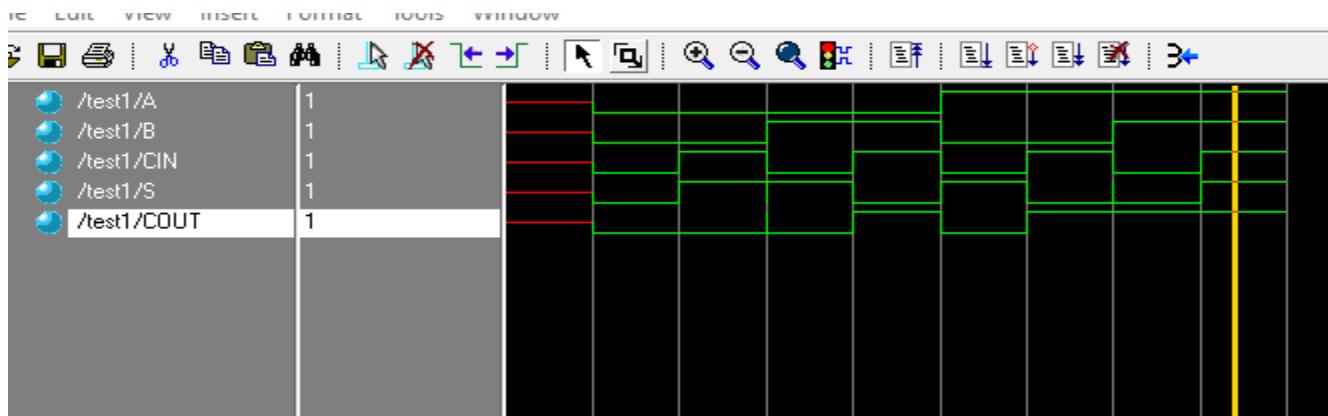
Design and Simulate the gate-level model of the circuit you patched in task 1. Give the code (including test bench) and resulted waveform in the space provided below.

Design and Stimulus Code:

```
In #  
1 module circuit (s, cout, a,b, cin);  
2 input a,b,cin;  
3 output s,cout;  
4 wire w1,w2,w3;  
5  
6 xor xor1 ( w1,a,b);  
7 and and1 ( w2,a,b);  
8 and and2 ( w3,w1,cin);  
9 xor xor2 ( s,w1,cin);  
10 or orl ( cout,w2,w3);  
11 endmodule  
12  
13 module test1;  
14 reg A,B,CIN;  
15 wire S, COUT;  
16 circuit test1 ( S,COUT, A,B,CIN);  
17 initial  
18 begin  
19     #100 A=l'b0;B=l'b0;CIN=l'b0;  
20     #100 A=l'b0;B=l'b0;CIN=l'b1;  
21     #100 A=l'b0;B=l'b1;CIN=l'b0;  
22     #100 A=l'b0;B=l'b1;CIN=l'b1;  
23     #100 A=l'b1;B=l'b0;CIN=l'b0;  
24     #100 A=l'b1;B=l'b0;CIN=l'b1;  
25     #100 A=l'b1;B=l'b1;CIN=l'b0;  
26     #100 A=l'b1;B=l'b1;CIN=l'b1;  
27 end  
28 endmodule
```



Output Wave:



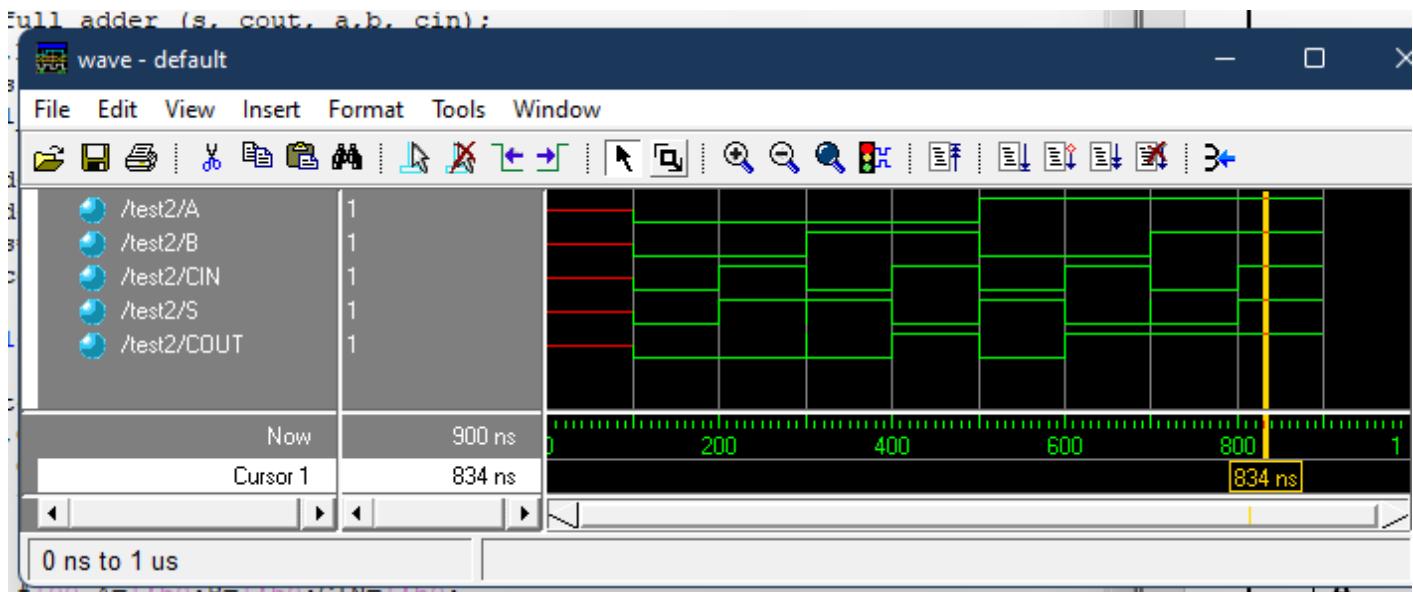
Task 3: Structural Modelling of the Given Circuit:

Design and Stimulus Code:

```
1 module half_adder (sum, cout, a,b);
2 input a,b;
3 output sum,cout;
4 assign sum = a ^ b;
5 assign cout = a & b;
6 endmodule
7
8 module full_adder (s, cout, a,b, cin);
9 input a,b,cin;
10 output s,cout;
11 wire hal_sum, ha2_sum, hal_carry,ha2_carry;
12
13 half_adder hal ( hal_sum, hal_carry, a,b);
14 half_adder ha2 (ha2_sum, ha2_carry, hal_sum, cin);
15 assign s= ha2_sum;
16 assign cout = ha2_carry | hal_carry;
17
18 endmodule
19
20 module test2;
21 reg A,B,CIN;
22 wire S, COUT;
23 circuit test2 ( S,COUT, A,B,CIN);
24 initial
25 begin
26     #100 A=l'b0;B=l'b0;CIN=l'b0;
27     #100 A=l'b0;B=l'b0;CIN=l'b1;
28     #100 A=l'b0;B=l'b1;CIN=l'b0;
29     #100 A=l'b0;B=l'b1;CIN=l'b1;
30     #100 A=l'b1;B=l'b0;CIN=l'b0;
31     #100 A=l'b1;B=l'b0;CIN=l'b1;
32     #100 A=l'b1;B=l'b1;CIN=l'b0;
33     #100 A=l'b1;B=l'b1;CIN=l'b1;
34 end
35 endmodule
```



Output Wave:



Observations/Comments:

In this lab we implemented a Full-Adder circuit comprising of two exclusive-OR gates, two AND gates and one OR gate. This is essentially two half-adder circuits joined by an OR gate. We also simulated the circuit on Proteus Software to verify the circuit truth table obtained by hardware circuit set-up. To back up our results, we also tested the circuit on ModelSim using Gate-Level Coding which produced the same output as our Proteus Simulation.

In this lab, we additionally wrote the Verilog code for the circuit using Structural Modelling. The stark difference is that structural modelling involves the use of creating sub-modules and utilizing them in a bigger module e.g., we can instantiate the module for half-adder twice in the full-adder module. Had we used dataflow or gate-level modelling, we would not be able to reuse the module and would have to write it twice. In addition, while patching the circuit, if two wires overlap or they touch the IC pin, short-circuit will give incorrect output.