National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Sir Arshad Nazeer **Dated: 2nd Nov, 2023**

Semester: 3rd **Section: D**

Group No.: 10

EE-221: Digital Logic Design

Lab 8: 2-bit binary Adder and Subtractor

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

Lab 8: 2-bit binary Adder and Subtractor

This Lab Activity has been designed to familiarize the students with design and working of binary adders using basic logic gates.

Objectives:

- ✓ Design and Implementation of Half Adder
- ✓ Design and Implementation of a Full Adder using Half Adders
- ✓ Extending the design to add 2-bit binary numbers
- ✓ Verification of 4-bit adder IC
- ✓ Gate-Level Verilog code for 4-bit adder

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 Total duration for the lab is 3 hrs.**
- ✓ **A lab with in-complete lab tasks will not be accepted.**
- ✓ 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.

Pre-Lab Tasks: (2 marks)

1. **What do you understand by half and full adders and why are these circuits so named?**

Half Adder and Full Adder are essential building blocks in digital logic design, used for adding binary numbers together. They are named based on the type of addition they perform and whether they handle carry input.

- **Half Adder:**

A Half Adder adds two binary digits, A and B, to produce two outputs: the sum (S) and the carry (C). However, it can only consider the current pair of bits and does not take into account any carry from previous additions.

The term "half" signifies that it can only partially add binary numbers, addressing just the least significant bit. It lacks the capability to incorporate carry from higher-order bits, making it a basic building block of more complex adders.

- **Full Adder:**

A Full Adder, on the other hand, adds three binary digits: A, B, and an incoming carry (Cin). It produces two outputs: the sum (S) and the carry-out (Cout). This means it can handle the current bit's addition as well as any carry from previous additions.

The term "full" indicates that it is capable of performing complete binary addition, considering both the current bit and any carry from higher-order bits.

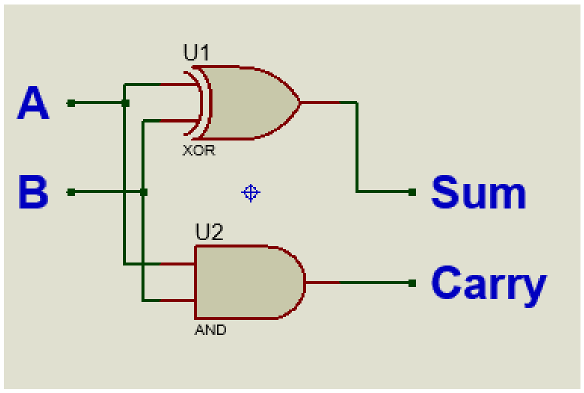
2. **Give the truth table and circuit for half adder and half subtractor.**

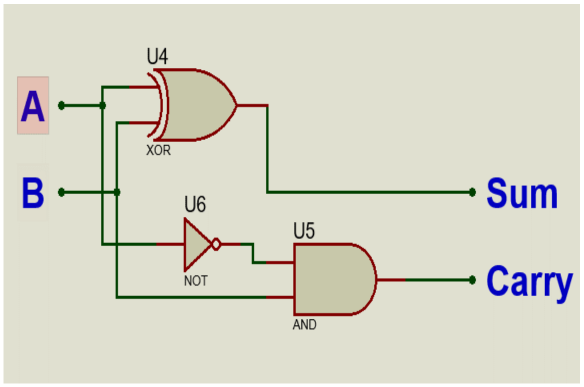
Truth Table:

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

A	B	D	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuit Diagram:



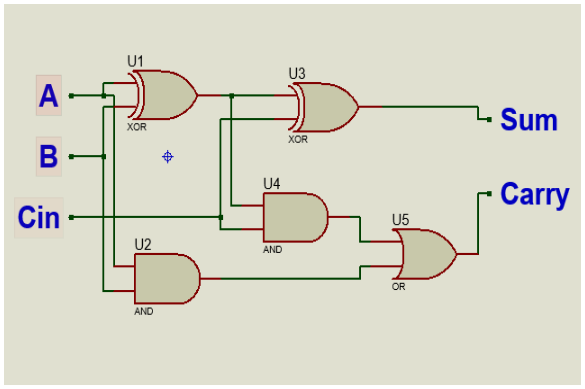


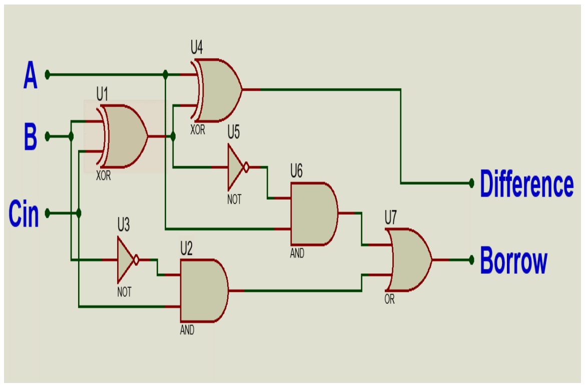
3. **Design a full adder/subtractor using the above designed half adders/subtractor.**

Truth Table:

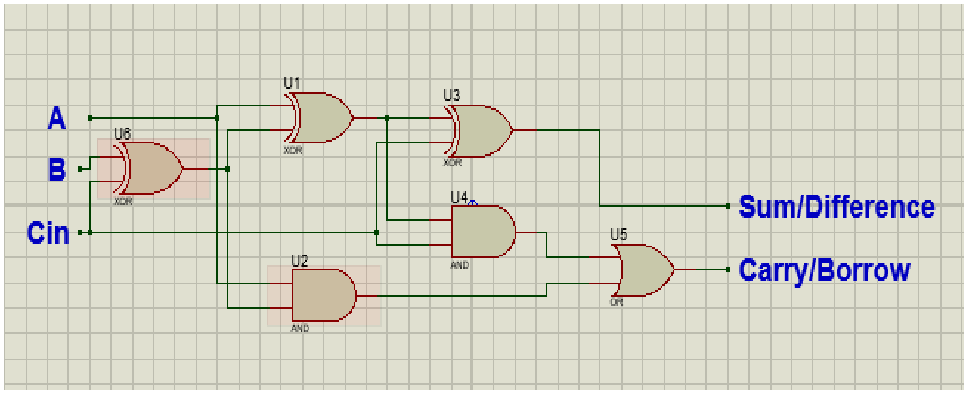
Full Adder					Full Subtractor				
A	**B**	**Cin**	**S**	**Cout**	**A**	**B**	**Bin**	**D**	**Bout**
0	0	1	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	1	1
0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	0	1	0	0	1
1	0	1	1	0	1	0	1	1	0
1	0	0	0	1	1	0	0	0	0
1	1	1	0	1	1	1	1	0	0
1	1	0	1	1	1	1	0	1	1

Circuit Diagram:





4. **Now add the subtraction option in your design and show the logic diagram of full adder with subtractor.**



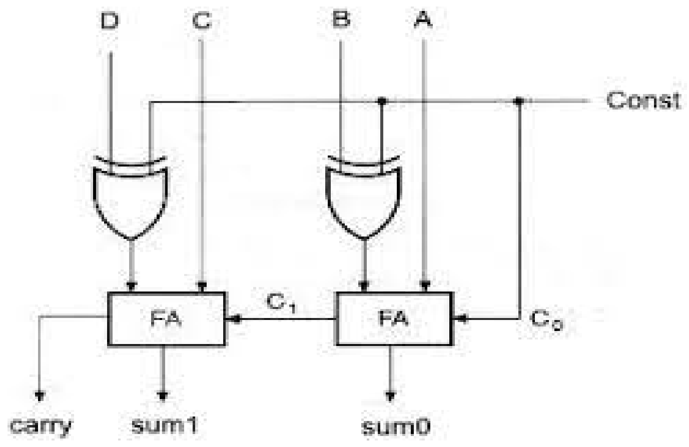
5. **Can you extend your design to n-bit binary addition? How does input carry propagates through full adder stages for such design and influences the speed? How can you overcome this problem?**

   - In the first adder, a carry is taken as an input, which is added to the input bits. This adder generates its own carry, which is the output of the OR gate which took inputs from both AND gates. The new carry is then passed into the second adder as an input in the same configuration as first adder's carry. Hence, each subsequent adder's carry is obtained as an output of the previous adder's OR gate.

   - This problem of relying on previous adder for next adder's input causes slower speed due to higher gate delay dependency. Instead of using the carry from previous adder only, the circuit can be modified to first calculate all the carries in a one-stage implementation by expressing each carry in terms of the original carry only, using Boolean Algebra. Hence all the carries will be available for each adder at the same time as the other inputs.

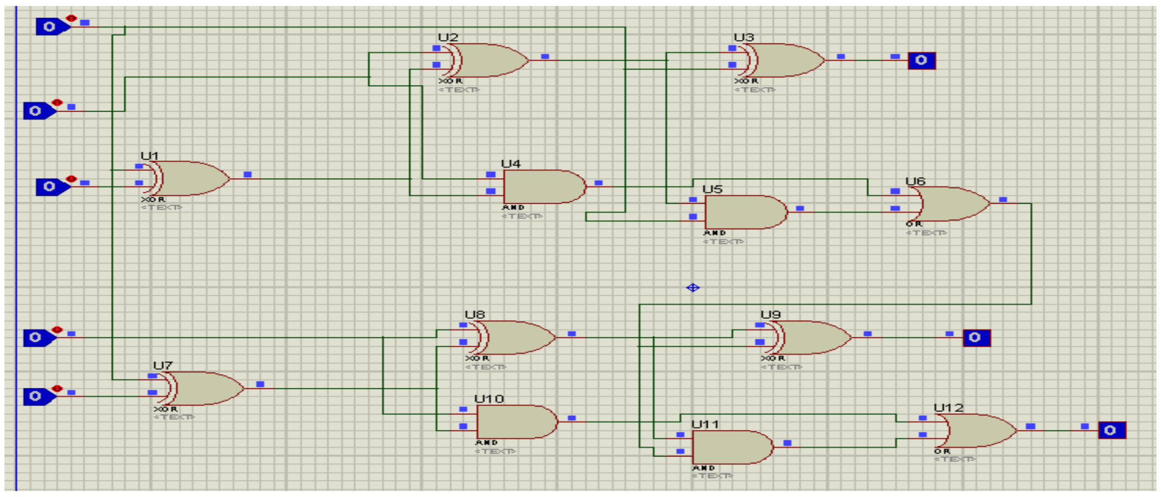
Lab Tasks: (3 marks)

6. **Extend your design to 2-bit binary adder and subtractor. Draw the block diagram of the circuit with inputs, outputs and carry.**

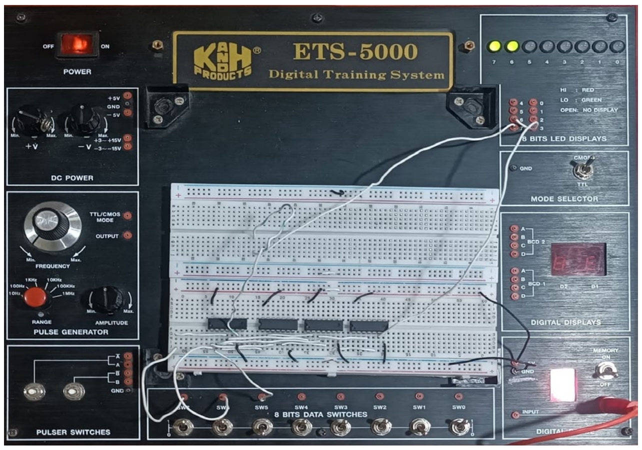


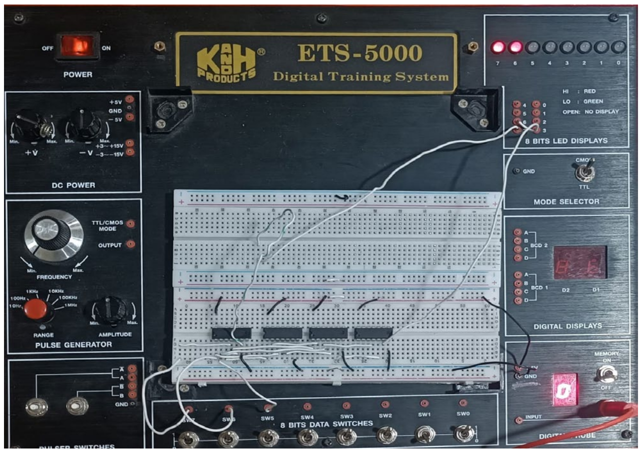
7. **Draw the schematic diagram of the circuit with complete pin configuration, number, each gate input output and carry. Carry out the hardware implementation of your 2-bit adder subtractor and show the results to lab instructor.**

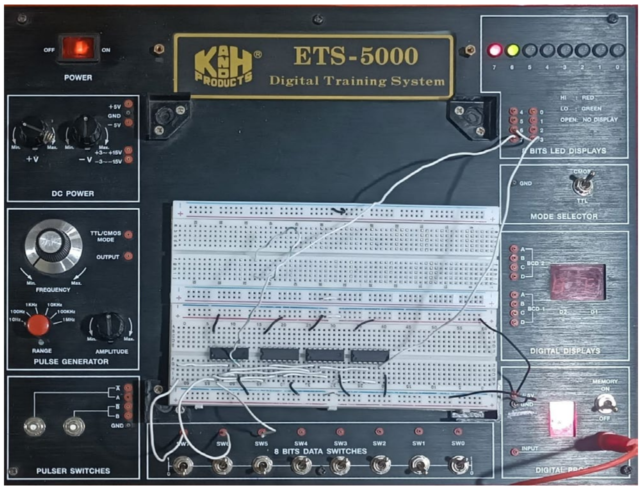
Schematics:

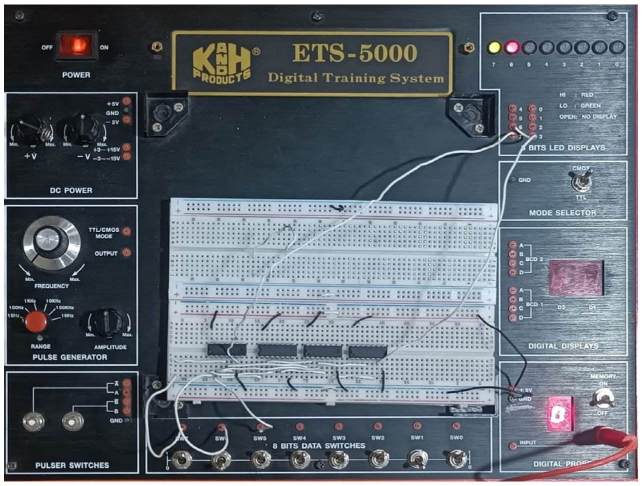


Hardware:





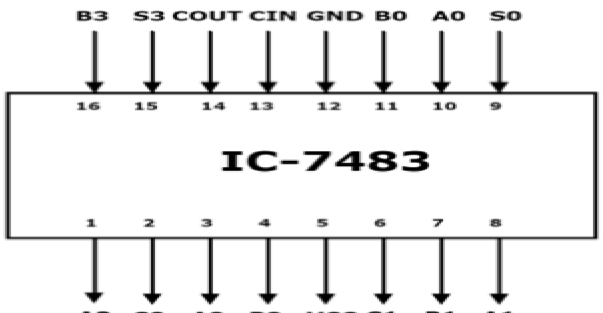




8. **Get the 4-bit binary adder IC from the lab and verify its functionality. Give IC number and pin-layout of the IC.**

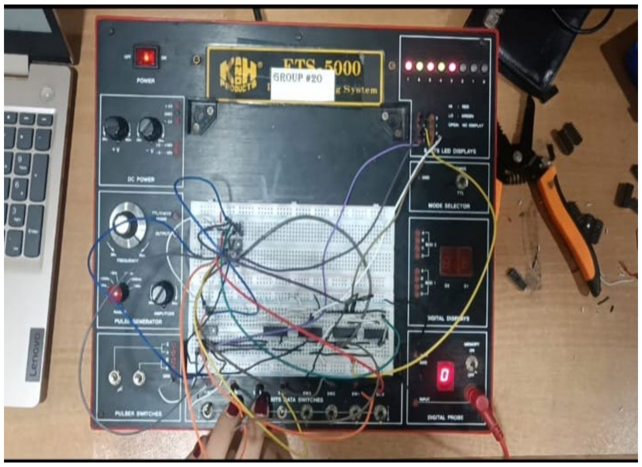
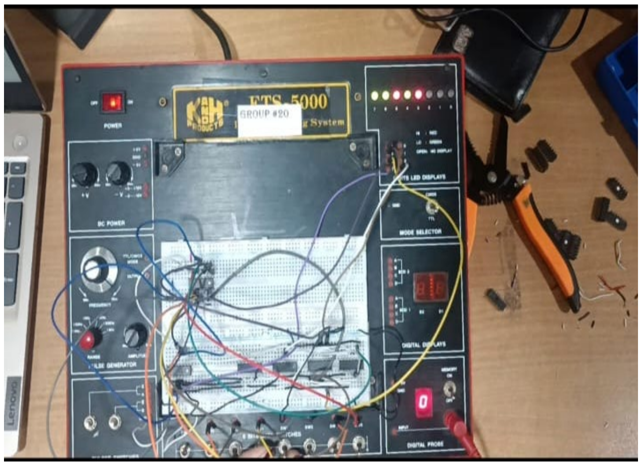
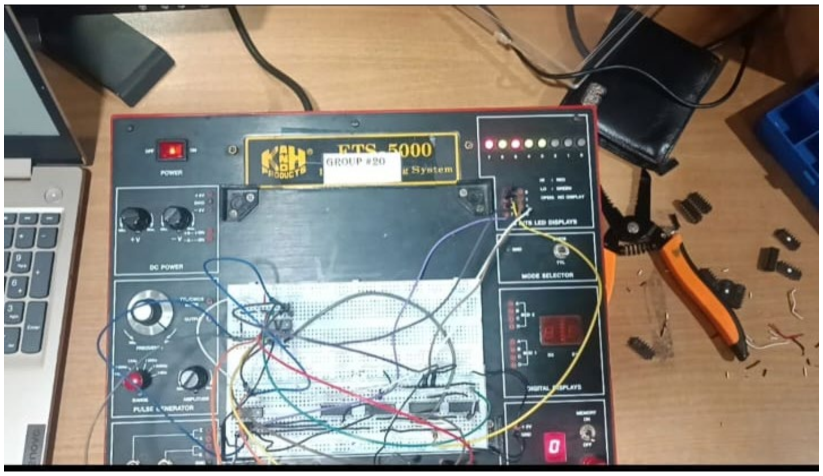
IC Number: 7483

Pin Layout:



IC-7483 Pinout

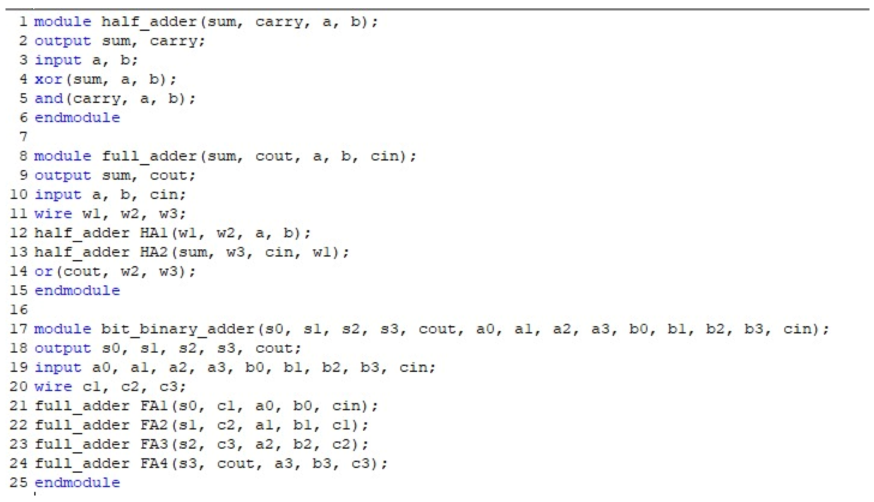
Hardware:

9. **Give the Gate-Level Verilog Code for four-bit adder and show the results on Simulation. Your code should contain following modules:**
   half adder
   full adder (by instantiating **half adder**)
   4_bit_ binary_ adder (by instantiating **full adder**)
   test bench_ 4_bit_binary_adder (for **4_bit_ binary_ adder**)

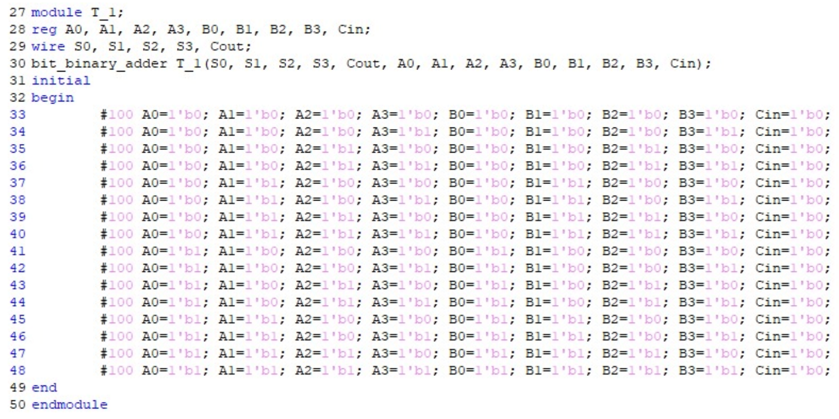
Verilog Code:

Module:



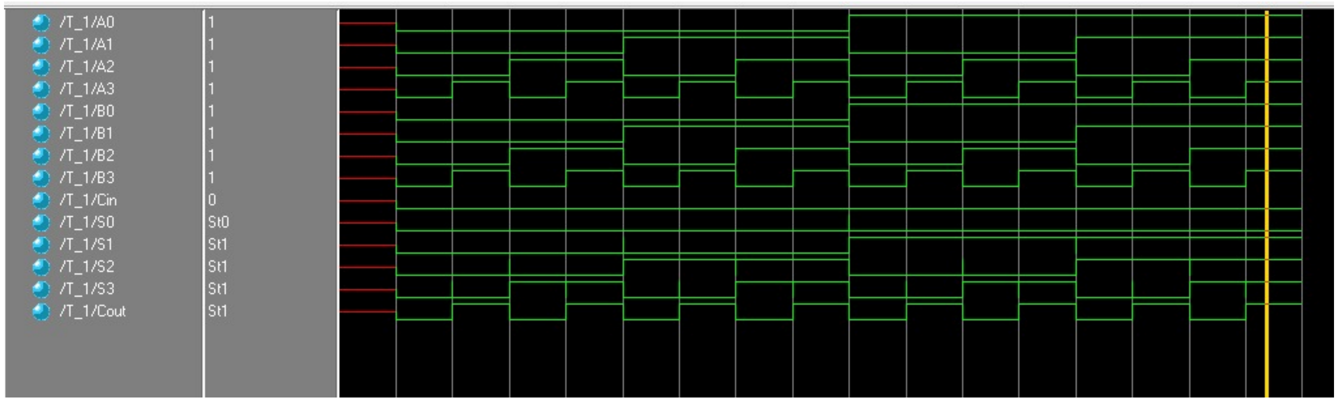
```verilog
module half_adder(sum, carry, a, b);
output sum, carry;
input a, b;
xor(sum, a, b);
and(carry, a, b);
endmodule

module full_adder(sum, cout, a, b, cin);
output sum, cout;
input a, b, cin;
wire w1, w2, w3;
half_adder HA1(w1, w2, a, b);
half_adder HA2(sum, w3, cin, w1);
or(cout, w2, w3);
endmodule

module bit_binary_adder(s0, s1, s2, s3, cout, a0, a1, a2, a3, b0, b1, b2, b3, cin);
output s0, s1, s2, s3, cout;
input a0, a1, a2, a3, b0, b1, b2, b3, cin;
wire c1, c2, c3;
full_adder FA1(s0, c1, a0, b0, cin);
full_adder FA2(s1, c2, a1, b1, c1);
full_adder FA3(s2, c3, a2, b2, c2);
full_adder FA4(s3, cout, a3, b3, c3);
endmodule
```

Test Bench Code:



```verilog
module T_1;
reg A0, A1, A2, A3, B0, B1, B2, B3, Cin;
wire S0, S1, S2, S3, Cout;
bit_binary_adder T_1(S0, S1, S2, S3, Cout, A0, A1, A2, A3, B0, B1, B2, B3, Cin);
initial
begin
        #100 A0=1'b0; A1=1'b0; A2=1'b0; A3=1'b0; B0=1'b0; B1=1'b0; B2=1'b0; B3=1'b0; Cin=1'b0;
        #100 A0=1'b0; A1=1'b0; A2=1'b0; A3=1'b1; B0=1'b0; B1=1'b0; B2=1'b0; B3=1'b1; Cin=1'b0;
        #100 A0=1'b0; A1=1'b0; A2=1'b1; A3=1'b0; B0=1'b0; B1=1'b0; B2=1'b1; B3=1'b0; Cin=1'b0;
        #100 A0=1'b0; A1=1'b0; A2=1'b1; A3=1'b1; B0=1'b0; B1=1'b0; B2=1'b1; B3=1'b1; Cin=1'b0;
        #100 A0=1'b0; A1=1'b1; A2=1'b0; A3=1'b0; B0=1'b0; B1=1'b1; B2=1'b0; B3=1'b0; Cin=1'b0;
        #100 A0=1'b0; A1=1'b1; A2=1'b0; A3=1'b1; B0=1'b0; B1=1'b1; B2=1'b0; B3=1'b1; Cin=1'b0;
        #100 A0=1'b0; A1=1'b1; A2=1'b1; A3=1'b0; B0=1'b0; B1=1'b1; B2=1'b1; B3=1'b0; Cin=1'b0;
        #100 A0=1'b0; A1=1'b1; A2=1'b1; A3=1'b1; B0=1'b0; B1=1'b1; B2=1'b1; B3=1'b1; Cin=1'b0;
        #100 A0=1'b1; A1=1'b0; A2=1'b0; A3=1'b0; B0=1'b1; B1=1'b0; B2=1'b0; B3=1'b0; Cin=1'b0;
        #100 A0=1'b1; A1=1'b0; A2=1'b0; A3=1'b1; B0=1'b1; B1=1'b0; B2=1'b0; B3=1'b1; Cin=1'b0;
        #100 A0=1'b1; A1=1'b0; A2=1'b1; A3=1'b0; B0=1'b1; B1=1'b0; B2=1'b1; B3=1'b0; Cin=1'b0;
        #100 A0=1'b1; A1=1'b0; A2=1'b1; A3=1'b1; B0=1'b1; B1=1'b0; B2=1'b1; B3=1'b1; Cin=1'b0;
        #100 A0=1'b1; A1=1'b1; A2=1'b0; A3=1'b0; B0=1'b1; B1=1'b1; B2=1'b0; B3=1'b0; Cin=1'b0;
        #100 A0=1'b1; A1=1'b1; A2=1'b0; A3=1'b1; B0=1'b1; B1=1'b1; B2=1'b0; B3=1'b1; Cin=1'b0;
        #100 A0=1'b1; A1=1'b1; A2=1'b1; A3=1'b0; B0=1'b1; B1=1'b1; B2=1'b1; B3=1'b0; Cin=1'b0;
        #100 A0=1'b1; A1=1'b1; A2=1'b1; A3=1'b1; B0=1'b1; B1=1'b1; B2=1'b1; B3=1'b1; Cin=1'b0;
end
endmodule
```

Waveform/Output:



Conclusion:

- This lab introduced us to the concept of a logic circuit that performs two arithmetic operations for which we made individual circuits separately and then integrated into one circuit using the choice bit with XOR gate.
- This allowed us to perform different operations with a single circuit instead of multiple circuits.
- Additionally, we learned how multiple modules of the same circuit can be cascaded with a reliable pattern to perform the same operation with a greater number of bits without needing to design a new circuit specifically for each number of bits.