**Department of Electrical Engineering**

**Faculty Member:** Arshad Nazir **Dated:** 19/11/2021

**Semester:** 3rd **Section:** BEE 12C

**Group No.: 7**

**EE-221: Digital Logic Design**

Lab 8: 2 – Bit Binary Adder / Subtractor

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4/CLO4** | **PLO4/CLO4** | **PLO5/CLO5** | **PLO8/CLO6** | **PLO9/CLO7** |  |
| **Name** | **Reg. No** | **Viva / Lab Performance** | **Analysis of data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** | **Total marks Obtained** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **25 Marks** |
| **Muhammad Adil Azeem** | **342459** |  |  |  |  |  |  |
| **Muhammad Umer** | **345834** |  |  |  |  |  |  |
| **Raahim Ahmad Waqar** | **342287** |  |  |  |  |  |  |
| **Saad Bakhtiar** | **341150** |  |  |  |  |  |  |
| **Syed Aun Ali Kazmi** | **342384** |  |  |  |  |  |  |

## 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 stepwise 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.
* **The Total duration for the lab is 3 hrs. After lab duration, a deduction of 5 marks per day will be done for late submission.**
* **A lab with in-complete lab tasks will not be accepted.**
* There are related questions at the end of this activity. Give complete answers.

**Pre – Lab Tasks**

**Pre – Lab Task 1**

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

* **Half Adder:**

A combinational circuit that adds two bits (Two Inputs) and generates a SUM and a CARRY. It is called a half adder because two half adders combined form a full adder.

**A, B → S, COUT**

* **Full Adder:**

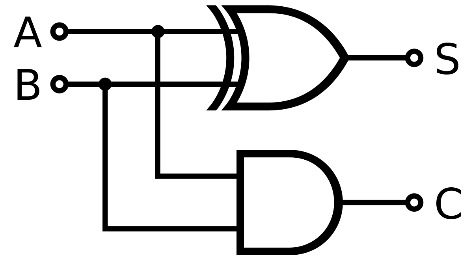
A combination circuit that adds three bits (Two Inputs and a Carry In) to yield a SUM and a CARRY. It is named so because complete addition can be performed with such an IC.

**A, B, CIN → S, COUT**

**Pre – Lab Task 2**

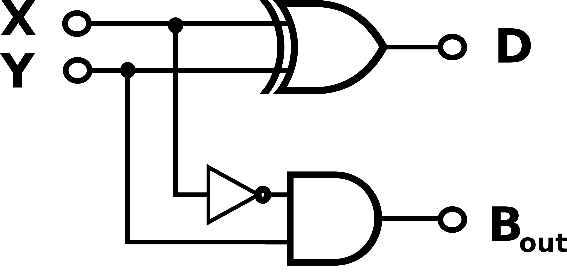
**Give the truth table and circuit for half adder and half subtractor.**

* **Half Adder**



|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **SUM** | **CARRY** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

* **Half Subtractor**



|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **DIFFERENCE** | **BORROW** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

**Pre – Lab Task 3**

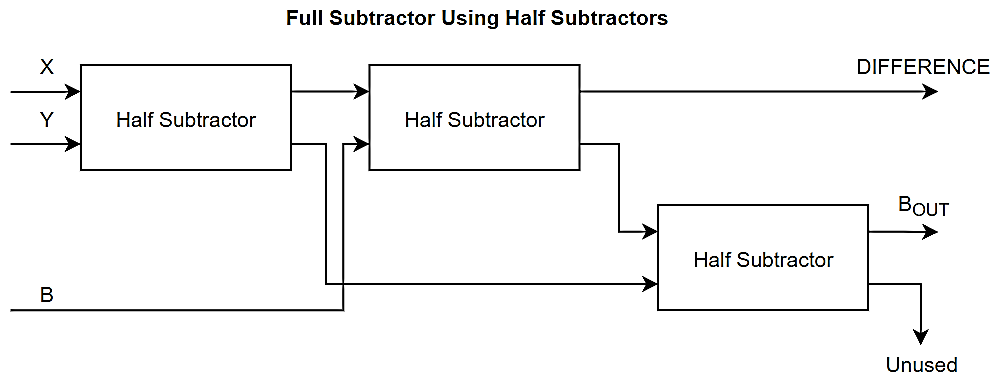
**Design a full adder/subtractor using the above designed half adders/subtractor.**

* **Full Adder Using Half Adders**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **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 |

* **Full Subtractor Using Half Subtractors**

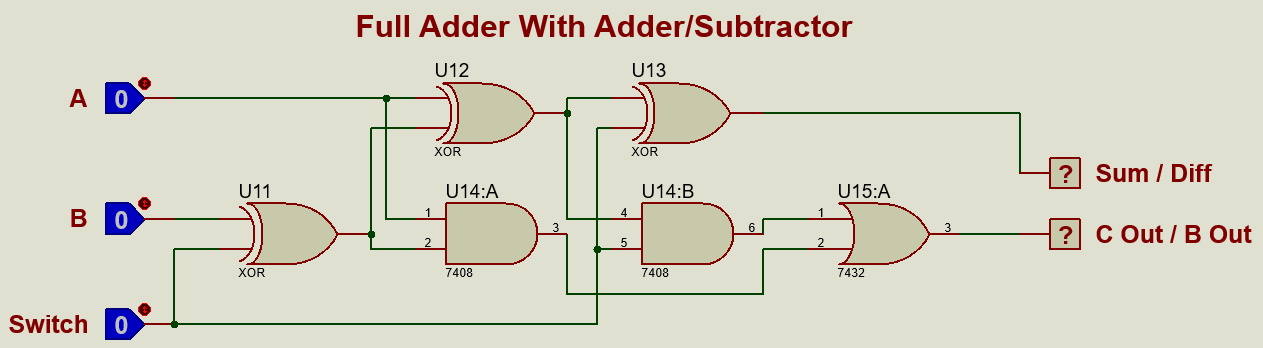


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **BIN** | **DIFF** | **BOUT** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Pre – Lab Task 4**

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

* **Full Adder / Subtractor**

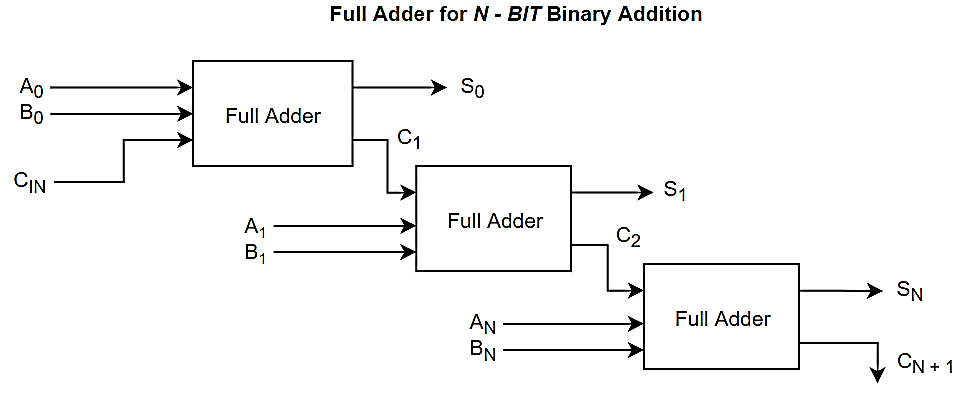


**Pre – Lab Task 5**

**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?**

Yes, this design can be extended to perform n – bit binary addition. An abstract for such a logic through blocks is given below:

* **Full Adder for N – Bit Binary Addition**



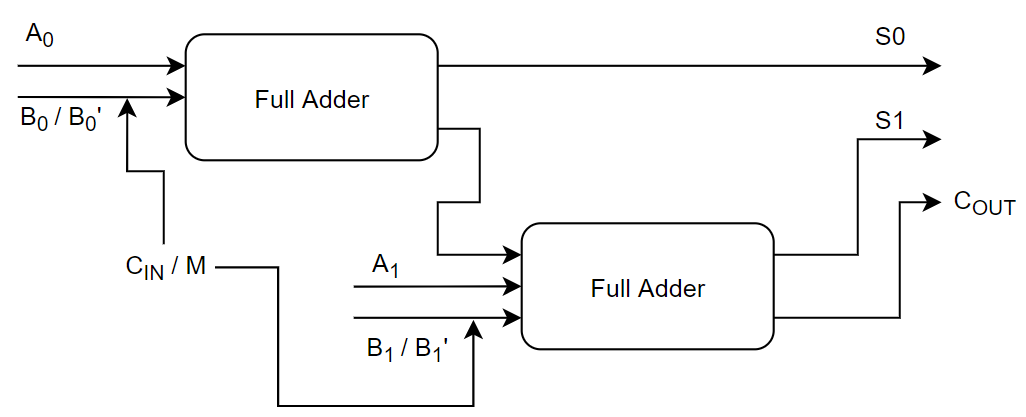
The input carry in each full adder has to propagate through an AND gate and an XOR gate, yielding two level gate implementation. For n – bits, there is a propagation delay of 2n; inferring that addition is drastically slow for a high number of bits.

Such a delay can be mitigated by using a Carry – Look Ahead Generator where each carry is propagated depending only the input carry CIN.

**Lab Tasks**

**Lab Task 1**

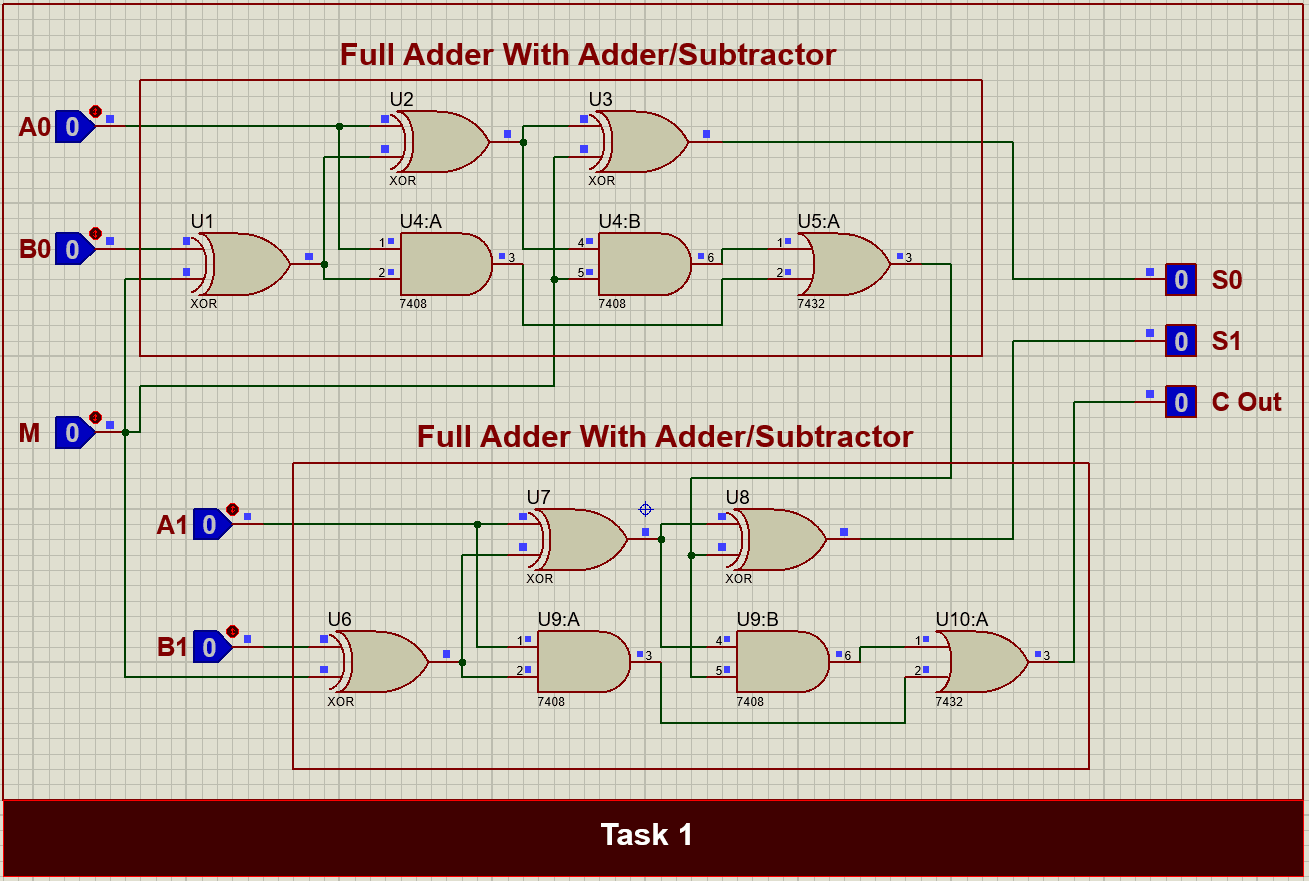
**Extend your design to 2-bit binary adder and subtractor. Draw the block diagram of the circuit with inputs, outputs and carry.**

****

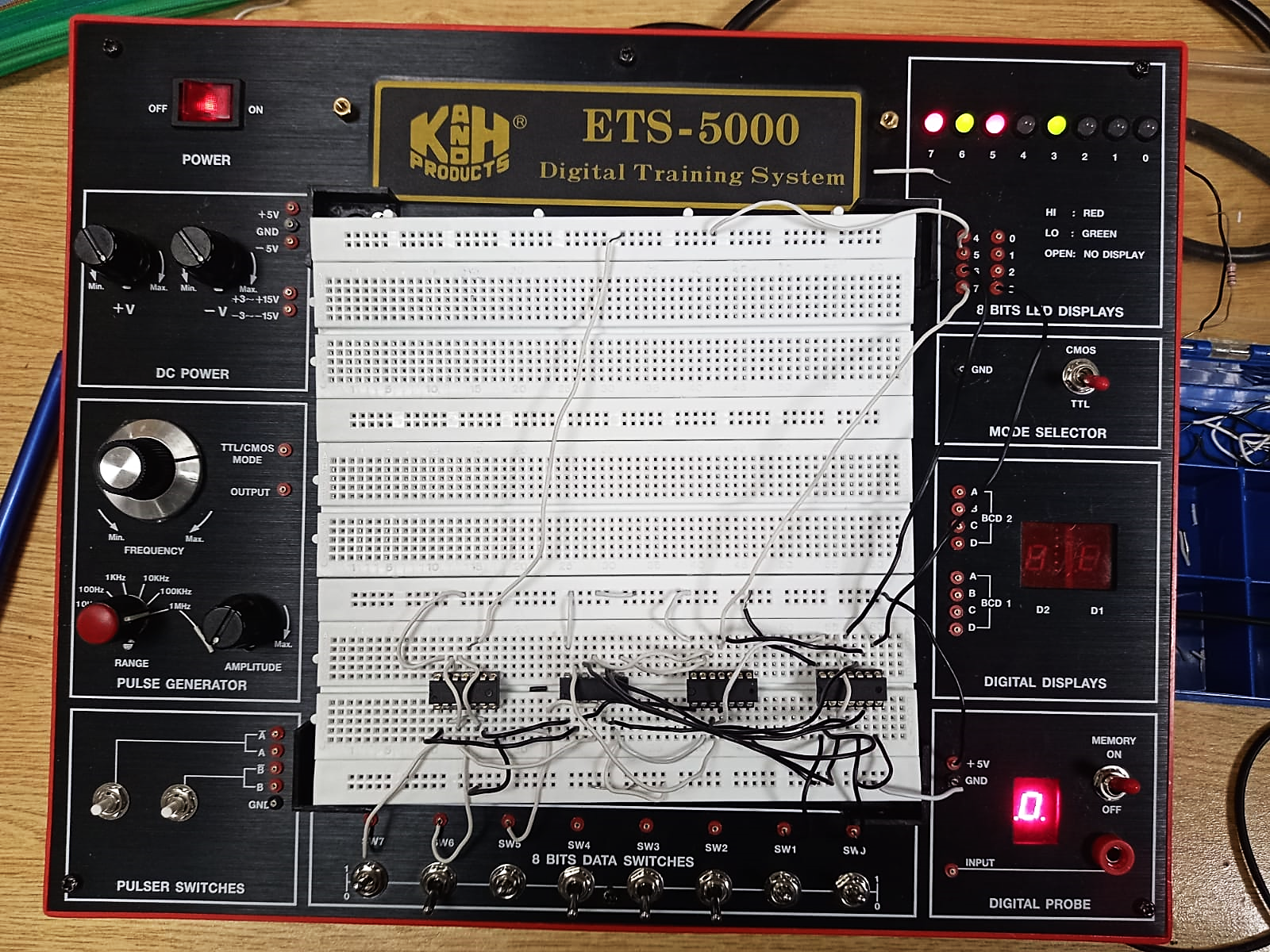
**Lab Task 2**

**Draw the schematic diagram of the circuit with complete pin configuration, number, each gate input output and carry.**

* **Proteus Simulation**



* **Hardware Implementation**

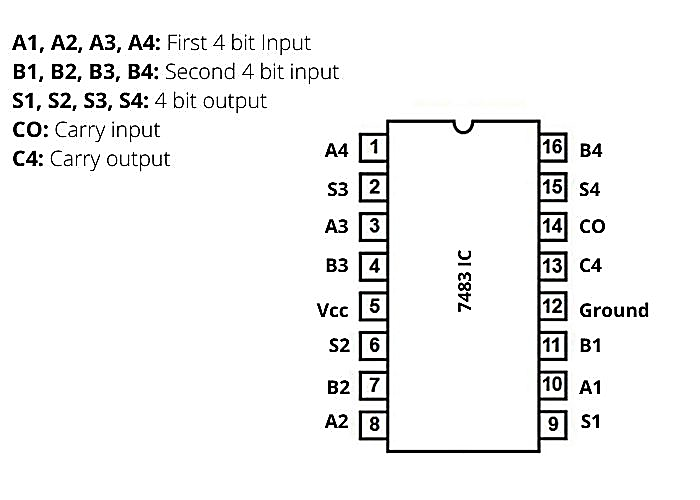


Which produces outputs that match the truth table made below and perform addition under the condition of the SW5 (Acting as Switch) being *LOW*, and subtraction under when SW5 is *HIGH.* An overflow detector was also implemented for addition / subtraction of signed numbers.

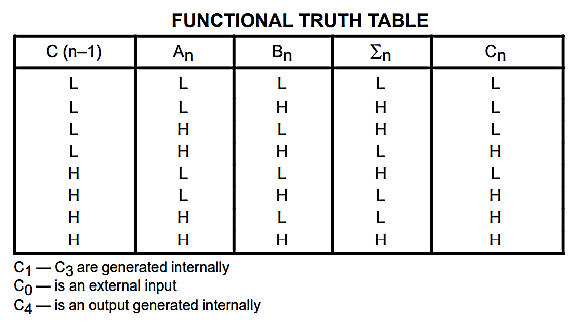
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A1** | **A0** | **B1** | **B0** | **CIN** | **CarryOUT** | **S1** | **S0** |
| 0 | 0 | 0 | 0 | 0 | **0** | **0** | **0** |
| 0 | 0 | 0 | 0 | 1 | **0** | **0** | **1** |
| 0 | 0 | 0 | 1 | 0 | **0** | **0** | **1** |
| 0 | 0 | 0 | 1 | 1 | **0** | **1** | **0** |
| 0 | 0 | 1 | 0 | 0 | **0** | **1** | **0** |
| 0 | 0 | 1 | 0 | 1 | **0** | **1** | **1** |
| 0 | 0 | 1 | 1 | 0 | **0** | **1** | **1** |
| 0 | 0 | 1 | 1 | 1 | **1** | **0** | **0** |
| 0 | 1 | 0 | 0 | 0 | **0** | **0** | **1** |
| 0 | 1 | 0 | 0 | 1 | **0** | **1** | **0** |
| 0 | 1 | 0 | 1 | 0 | **0** | **1** | **0** |
| 0 | 1 | 0 | 1 | 1 | **0** | **1** | **1** |
| 0 | 1 | 1 | 0 | 0 | **0** | **1** | **1** |
| 0 | 1 | 1 | 0 | 1 | **1** | **0** | **0** |
| 0 | 1 | 1 | 1 | 0 | **1** | **0** | **0** |
| 0 | 1 | 1 | 1 | 1 | **1** | **0** | **1** |
| 1 | 0 | 0 | 0 | 0 | **0** | **1** | **0** |
| 1 | 0 | 0 | 0 | 1 | **0** | **1** | **1** |
| 1 | 0 | 0 | 1 | 0 | **0** | **1** | **1** |
| 1 | 0 | 0 | 1 | 1 | **1** | **0** | **0** |
| 1 | 0 | 1 | 0 | 0 | **1** | **0** | **0** |
| 1 | 0 | 1 | 0 | 1 | **1** | **0** | **1** |
| 1 | 0 | 1 | 1 | 0 | **1** | **0** | **1** |
| 1 | 0 | 1 | 1 | 1 | **1** | **1** | **0** |
| 1 | 1 | 0 | 0 | 0 | **0** | **1** | **1** |
| 1 | 1 | 0 | 0 | 1 | **1** | **0** | **0** |
| 1 | 1 | 0 | 1 | 0 | **1** | **0** | **0** |
| 1 | 1 | 0 | 1 | 1 | **1** | **0** | **1** |
| 1 | 1 | 1 | 0 | 0 | **1** | **0** | **1** |
| 1 | 1 | 1 | 0 | 1 | **1** | **1** | **0** |
| 1 | 1 | 1 | 1 | 0 | **1** | **1** | **0** |
| 1 | 1 | 1 | 1 | 1 | **1** | **1** | **1** |

**Lab Task 3**

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



The IC performed the 4 – bit binary addition and also verified the functional truth table;



**Lab Task 4**

**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 halfadder(sum, carry, a, b);

    input  a, b;

    output sum, carry;

    xor xor1(sum, a, b);

    and and1( carry, a, b);

endmodule

module fulladder(sum, carryOut, a, b, Cin);

    input a, b, Cin;

    output sum, carryOut;

    wire w1, w2, w3, w;

    halfadder h1(w1, w2, a, b);

    halfadder h2(sum, w3, w1, Cin);

    halfadder h3(carryOut, w, w3, w2);

endmodule

module fourbitbinaryadder(S0, S1, S2, S3, Cout, A0, A1, A2, A3, B0, B1, B2, B3, Cin);

    input A0, A1, A2, A3, B0, B1, B2, B3, Cin;

    output S0, S1, S2, S3, Cout;

    wire w1, w2, w3;

    fulladder f1(S0, w1, A0, B0, Cin);

    fulladder f2(S1, w2, A1, B1, w1);

    fulladder f3(S2, w3, A2, B2, w2);

    fulladder f4(S3, Cout, A3, B3, w3);

endmodule

module testbenchfourbitbinaryadder();

reg A0, A1, A2, A3, B0, B1, B2, B3, Cin;

wire S0, S1, S2, S3, Cout;

fourbitbinaryadder f(S0, S1, S2, S3, Cout, A0, A1, A2, A3, B0, B1, B2, B3, Cin);

        initial

            begin

            Cin=0'b0; A0=1'b0;A1=1'b0;A2=1'b0;A3=1'b0;

            B0=1'b0;B1=1'b0;B2=1'b1;B3=1'b0;

            #100 Cin=0'b0; A0=1'b0;A1=1'b0;A2=1'b0;A3=1'b0;

            B0=1'b0;B1=1'b0;B2=1'b1;B3=1'b1;

            #100 Cin=0'b0; A0=1'b1;A1=1'b0;A2=1'b0;A3=1'b0;

            B0=1'b0;B1=1'b1;B2=1'b0;B3=1'b0;

            #100 Cin=0'b0; A0=1'b0;A1=1'b0;A2=1'b0;A3=1'b1;

            B0=1'b0;B1=1'b0;B2=1'b1;B3=1'b1;

            #100 Cin=0'b0; A0=1'b1;A1=1'b1;A2=1'b1;A3=1'b1;

            B0=1'b1;B1=1'b0;B2=1'b1;B3=1'b0;

            #100 Cin=1'b1; A0=1'b1;A1=1'b0;A2=1'b0;A3=1'b1;

            B0=1'b0;B1=1'b0;B2=1'b1;B3=1'b0;

            #100 Cin=1'b1; A0=1'b0;A1=1'b1;A2=1'b0;A3=1'b1;

            B0=1'b1;B1=1'b0;B2=1'b1;B3=1'b0;

            #100 Cin=1'b1; A0=1'b0;A1=1'b1;A2=1'b0;A3=1'b0;

            B0=1'b0;B1=1'b0;B2=1'b0;B3=1'b1;

            #100 Cin=1'b1; A0=1'b1;A1=1'b0;A2=1'b1;A3=1'b1;

            B0=1'b1;B1=1'b1;B2=1'b1;B3=1'b0;

            end

endmodule

* **Waveform / Output**

