**Department of Electrical Engineering**

**Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Semester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Group No.:*

**EE-221: Digital Logic Design**

**Lab 10: Magnitude Comparator**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **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 Team Work** | **Total marks Obtained** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **25 Marks** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## Lab 9: Magnitude Comparator

This Lab Activity has been designed to familiarize students with design and working of combinational circuits using basic logic gates.

**Objectives:**

* Design and Implementation of 2-bit magnitude comparator using classical design method learned in the class.
* Design of a 4-bit magnitude comparator using a 4-bit adder IC and logic gates
* Verification of 4 bit comparator IC
* Dataflow modeling in Verilog HDL

**Lab Instructions**

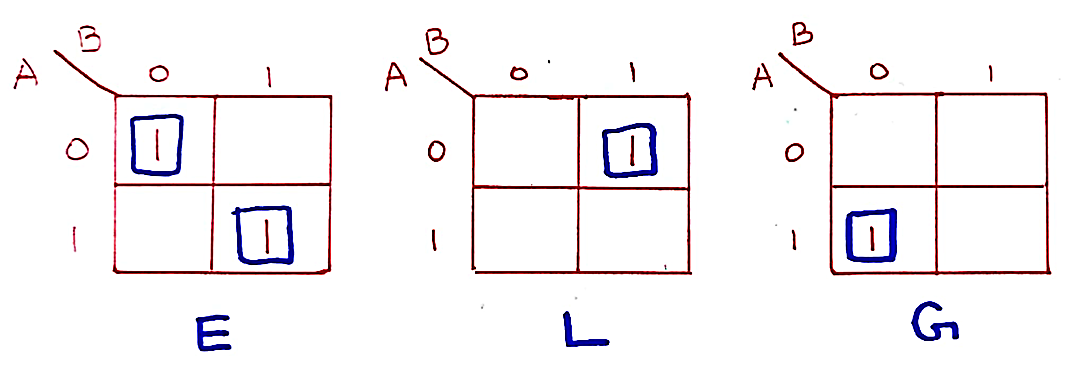
* This lab activity comprises three parts, namely Pre-lab, Lab tasks, and Post-Lab Viva session.
* The lab report will be uploaded on LMS three days before scheduled lab date. The students will get hard copy of lab report, complete the Pre-lab task before coming to the lab and deposit it with teacher/lab engineer for necessary evaluation.
* 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.

**Pre-Lab Tasks**

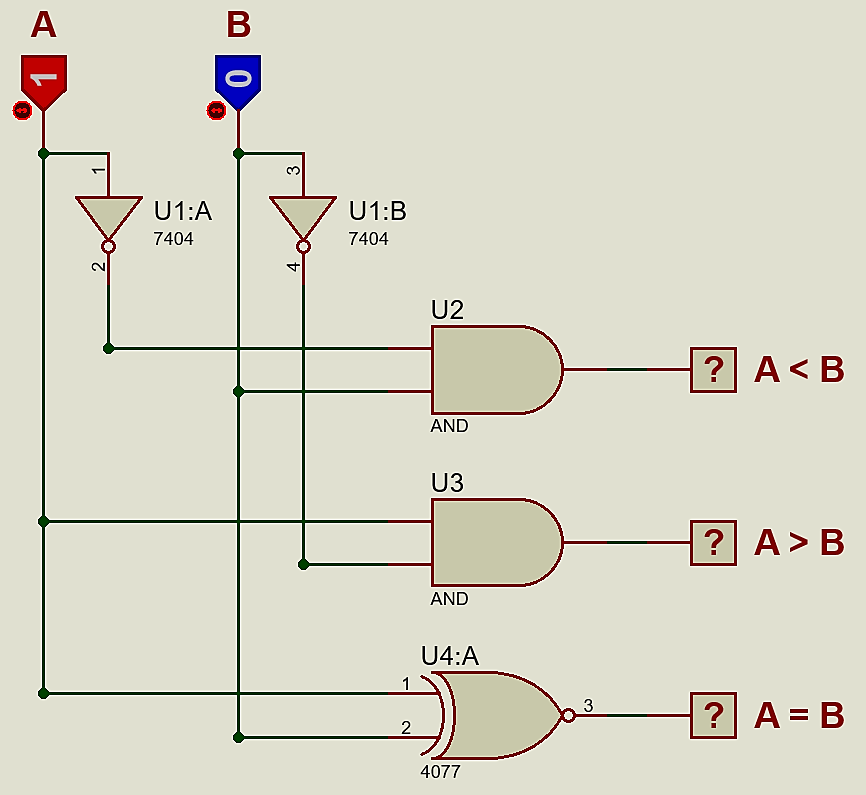
1. What do you mean by a comparator circuit? Draw the truth table for a 1-bit magnitude comparator. The comparator has 2 inputs (A and B) and 3 outputs E, L and G for (A=B), (A<B) and (A>B) respectively. For example, one combination is filled in below example, for your guidance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **E (A=B)** | **G (A>B)** | **L (A<B)** |
| 0 | 0 | **1** | **0** | **0** |
| 0 | 1 | **0** | **1** | **0** |
| 1 | 0 | **0** | **0** | **1** |
| 1 | 1 | **1** | **0** | **0** |

2. Simplify the functions E, G and L and give their Logic diagrams.



|  |  |
| --- | --- |
| **Function** | **Expression** |
| E | A’B’ + AB |
| L | A’B |
| G | AB’ |

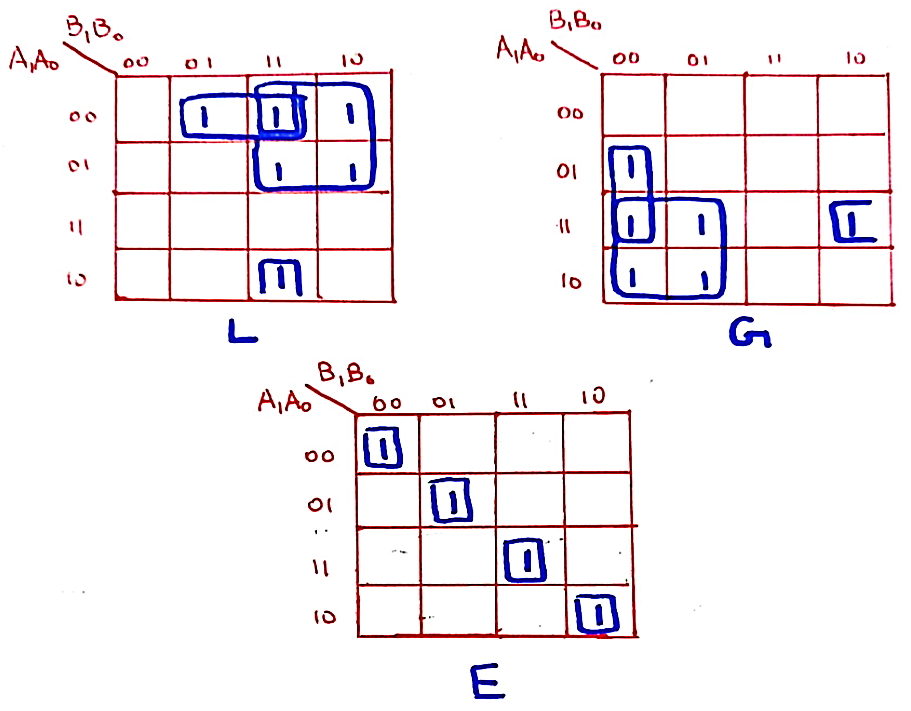


3. Design a 2-bit magnitude comparator.

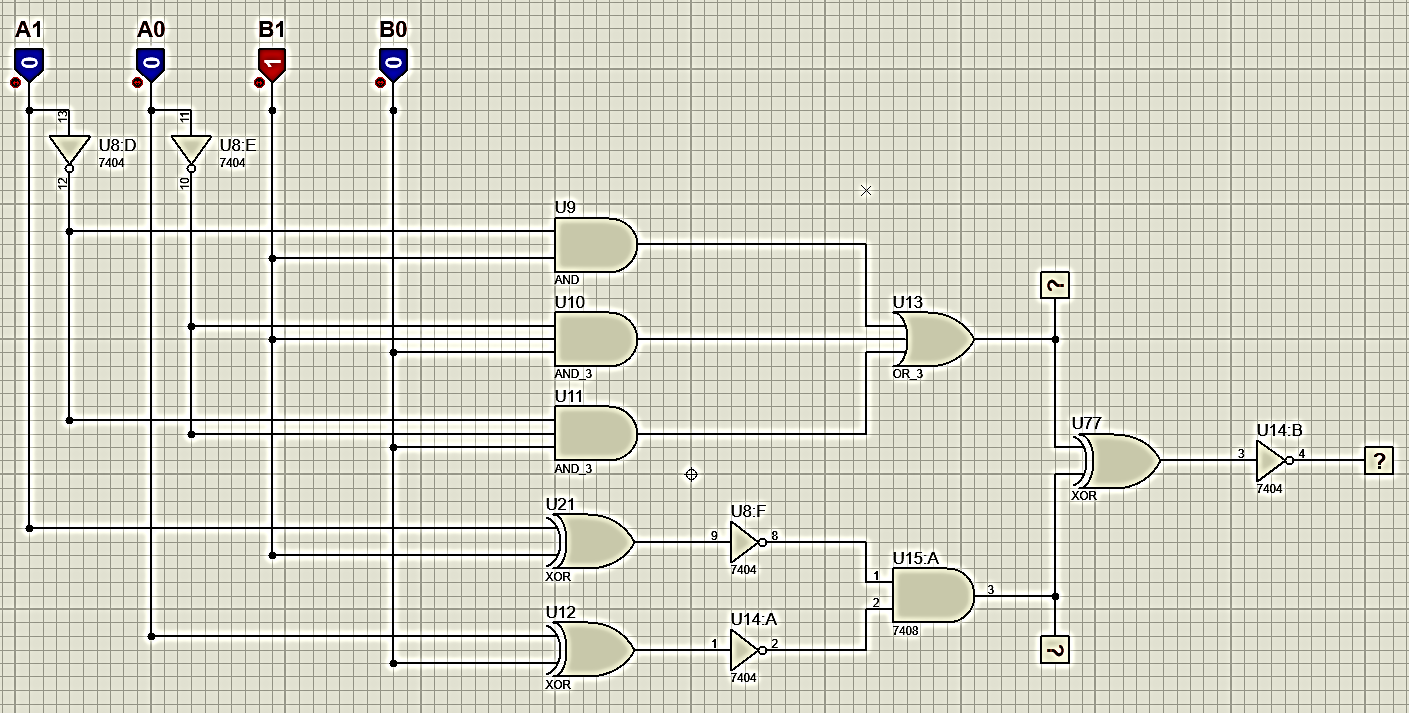
a) List the truth table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Inputs** | | | | **Outputs** | | |
| **A1** | **A0** | **B1** | **B0** | **E (A=B)** | **G (A>B)** | **L (A<B)** |
| 0 | 0 | 0 | 0 | **0** | **1** | **0** |
| 0 | 0 | 0 | 1 | **1** | **0** | **0** |
| 0 | 0 | 1 | 0 | **1** | **0** | **0** |
| 0 | 0 | 1 | 1 | **1** | **0** | **0** |
| 0 | 1 | 0 | 0 | **0** | **0** | **1** |
| 0 | 1 | 0 | 1 | **0** | **1** | **0** |
| 0 | 1 | 1 | 0 | **1** | **0** | **0** |
| 0 | 1 | 1 | 1 | **1** | **0** | **0** |
| 1 | 0 | 0 | 0 | **0** | **0** | **1** |
| 1 | 0 | 0 | 1 | **0** | **0** | **1** |
| 1 | 0 | 1 | 0 | **0** | **1** | **0** |
| 1 | 0 | 1 | 1 | **1** | **0** | **0** |
| 1 | 1 | 0 | 0 | **0** | **0** | **1** |
| 1 | 1 | 0 | 1 | **0** | **0** | **1** |
| 1 | 1 | 1 | 0 | **0** | **0** | **1** |
| 1 | 1 | 1 | 1 | **0** | **1** | **0** |

b) Simplify each of the functions E, G and L and give the logic diagram**.**



|  |  |  |
| --- | --- | --- |
| **Function** | **Expression** | **Simplified** |
| E | A1’A0’B1’B0’ + A1’A0B1’B0 + A1A0B1B0 + A1A0’B1B0’ | (A1⊙B1)(A0⊙B0) |
| L | A1’B1 + A0’B1B0 + A1’A0’B0 | N/A |
| G | A1B1’ + A0B1’B0’ + A1A0B0’ | N/A |



**Lab Tasks: 10 marks**

4. Implement the 2-bit Comparator circuit you arrived in your Pre-lab Task. Give the Complete circuit schematic diagram here again. (4 marks- Analysis)

You can use following gates

NOT, AND, OR, XOR, NAND, NOR

5. Get the 4-bit magnitude comparator IC (7485) from the lab. Give its function table, Pin Layout and show its implemented hardware. Does your results match with task4 results? If not state its reason. (3 marks = 1-mark Analysis, 2 marks Modern tool usage)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Inputs** | | | | | | | | **Outputs** | | |
| **A3** | **A2** | **A1** | **A0** | **B3** | **B2** | **B1** | **B0** | **E (A=B)** | **G (A>B)** | **L (A<B)** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | **0** | **0** |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | **0** | **1** | **0** |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **0** | **0** | **1** |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | **0** | **0** | **1** |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | **1** | **0** | **0** |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | **0** | **0** | **1** |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | **0** | **0** | **1** |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | **0** | **1** | **0** |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | **1** | **0** | **0** |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | **0** | **0** | **1** |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | **0** | **1** | **0** |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | **0** | **0** | **1** |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | **1** | **0** | **0** |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | **0** | **1** | **0** |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | **0** | **1** | **0** |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **1** | **0** | **0** |

6. Write verilog code of 2 bit comparator using dataflow modeling. (3 marks - Modern tool usage)

module comparator(e, g , l, a1, a0, b1, b0);

    input a1, a0, b1, b0;

    output e, g , l;

    assign e = (a0 ^~ b0) & (a1 ^~ b1);

    assign g = (a1 & ~b1) | (a0 & ~b1 & ~b0) | (a1 & a0 & ~b0);

    assign l = (~a1 & b1) | (~a0 & b1 & b0) | (~a1 & ~a0 & b0);

endmodule

module testbench();

    reg A1, A0, B1, B0;

    wire E, G, L;

    comparator t1(E, G, L, A1, A0, B1, B0);

    initial begin

        #100 A1 = 0; A0 = 0; B1 = 0; B0 = 0;

        #100 A1 = 0; A0 = 0; B1 = 0; B0 = 1;

        #100 A1 = 0; A0 = 0; B1 = 1; B0 = 0;

        #100 A1 = 0; A0 = 0; B1 = 1; B0 = 1;

        #100 A1 = 0; A0 = 1; B1 = 0; B0 = 0;

        #100 A1 = 0; A0 = 1; B1 = 0; B0 = 1;

        #100 A1 = 0; A0 = 1; B1 = 1; B0 = 0;

        #100 A1 = 0; A0 = 1; B1 = 1; B0 = 1;

        #100 A1 = 1; A0 = 0; B1 = 0; B0 = 0;

        #100 A1 = 1; A0 = 0; B1 = 0; B0 = 1;

        #100 A1 = 1; A0 = 0; B1 = 1; B0 = 0;

        #100 A1 = 1; A0 = 0; B1 = 1; B0 = 1;

        #100 A1 = 1; A0 = 1; B1 = 0; B0 = 0;

        #100 A1 = 1; A0 = 1; B1 = 0; B0 = 1;

        #100 A1 = 1; A0 = 1; B1 = 1; B0 = 0;

        #100 A1 = 1; A0 = 1; B1 = 1; B0 = 1;

    end

endmodule

