**Experiment / Assignment / Tutorial No. 1**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: COMPS (A1) Roll No.: 1911004 Experiment / assignment / tutorial No.: 1** |

|  |
| --- |
| **Title: Basic Gates & Universal Gates** |

**Objective: To study the basic gates: AND, OR, NOT :: universal gates: NAND, NOR, XOR,XNOR**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO1: Recall basic gates and binary, octal & hexadecimal calculations and conversions.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* **Vlab Link:** [**http://vlabs.iitkgp.ernet.in/dec/#**](http://vlabs.iitkgp.ernet.in/dec/)
* **R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill**
* **http://www.ee.surrey.ac.uk/Projects/Labview/gatesfunc/**
* **http://www.electronics-tutorials.ws/boolean/bool\_6.html**

**Pre Lab/ Prior Concepts:**

**Gate is a logic circuit with one or more inputs but only one output. Gates are digital (two state) circuit because the input & output are either low or high. Gates provide high output for certain combinations of input & for other combinations the output is low. Total number of combinations for a gate is 2^n; where n is number of input.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Classification: The two types of gate are:**

1. **Basic or Fundamental Gates:**

* **AND , OR , NOT GATES:**

**These gates are basic/fundamental logic gates as all other gates can be constructed and derived from them using different combination of these gates .**

* **Eg. XOR , XNOR , NAND ,NOR GATES can be constructed from basic gates**

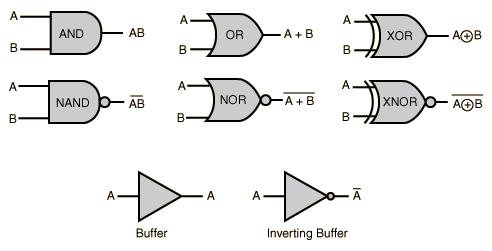
1. **Derived Gates:**

* **XOR , XNOR , NAND ,NOR GATES :**

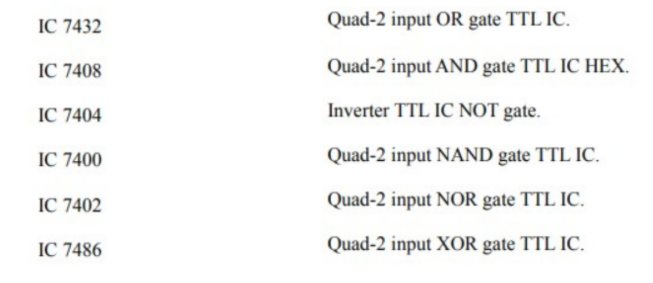
**These gates are derived gates as they are formed from AND , NOT ,OR GATES using basic gate different combination .**

**NAND & NOR Gates are universal gates from which AND,OR,NOT can be constructed**

**Symbols of gates**

****

**LOGIC GATE (IC NUMBER) SPECIFICATION**

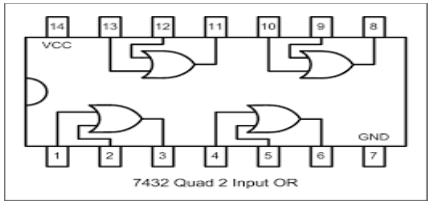
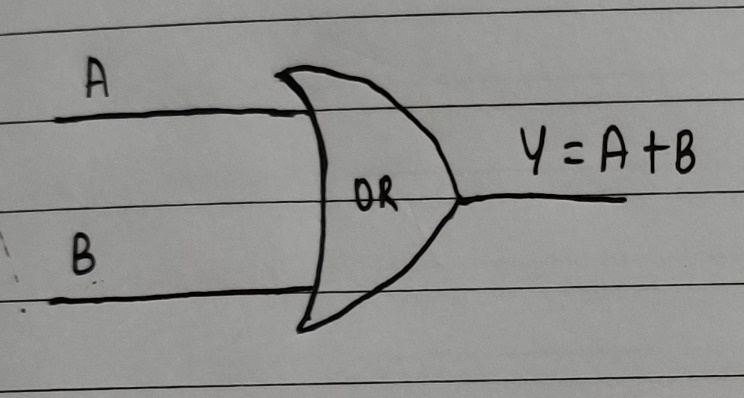


**Implementation Details:**

**Basic Gates**

1. **OR gate: The OR gate has two or more inputs but only 1 output. If any or all the inputs are high, the output is high. If all the inputs are low, the output is low.**

**Y= A+B**



**Symbol for OR gate Pin Diagram For IC 7432**

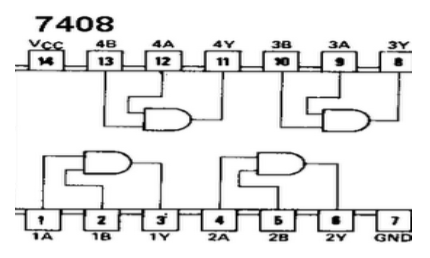
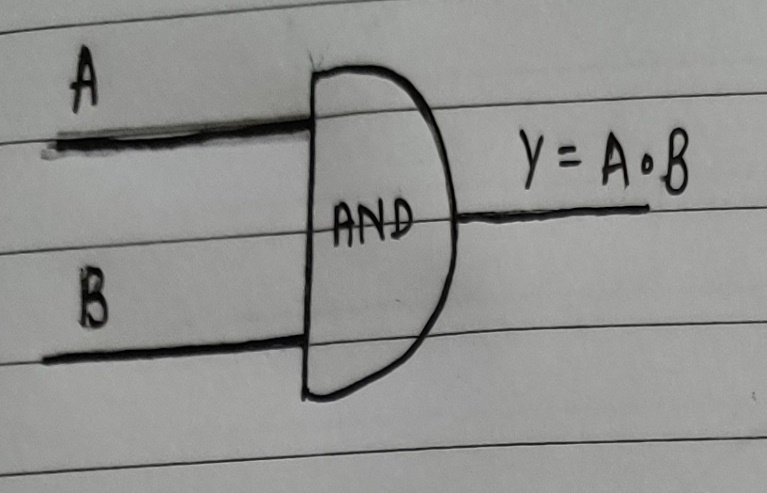
**The truth table for OR operations are:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y=A+B** |
| **1** | **1** | **1** |
| **1** | **0** | **1** |
| **0** | **1** | **1** |
| **0** | **0** | **0** |

1. **AND gate: The AND gate has two or more inputs but only one output. If any or all inputs are high then output is also high**

**Y= A \* B**

**Symbol for AND gate Pin Diagram For IC 7408**



**The truth table for AND operations are:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y=A\*B** |
| **1** | **1** | **1** |
| **1** | **0** | **0** |
| **0** | **1** | **0** |
| **0** | **0** | **0** |

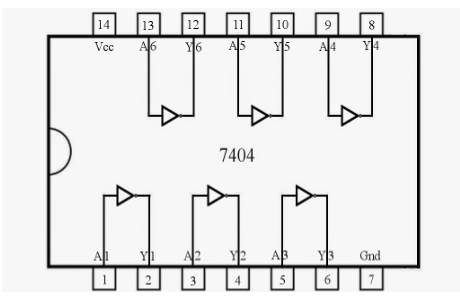
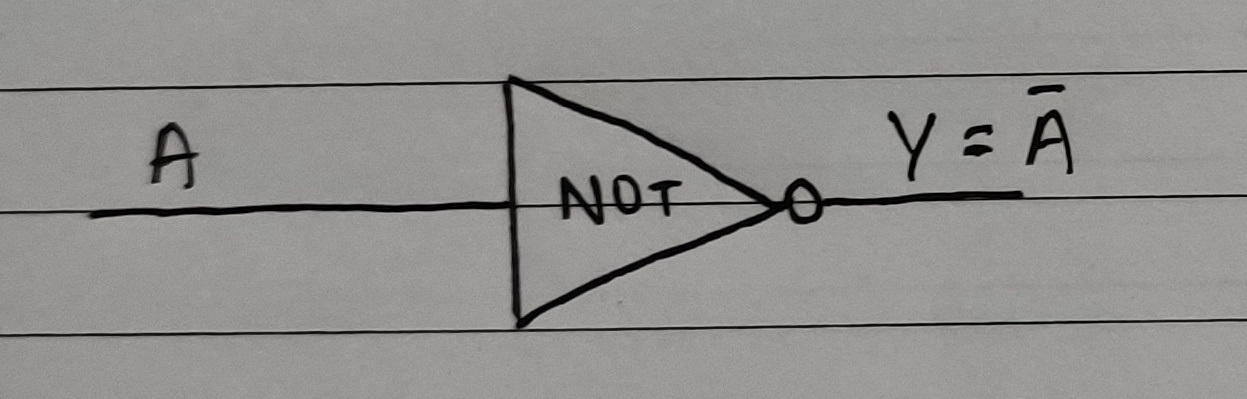
1. **NOT gate: The Not gate is a gate with only one input and one output. The output is always in opposite state of an input. A NOT gate is also called as Inverter because it performs inversion.**

**\_**

**Y= A**

****

**Symbol for NOT gate Pin Diagram For IC 7404**



**The truth table for NOT operations is:**

|  |  |
| --- | --- |
| **A** | **Y= A ‘** |
| **1** | **0** |
| **0** | **1** |

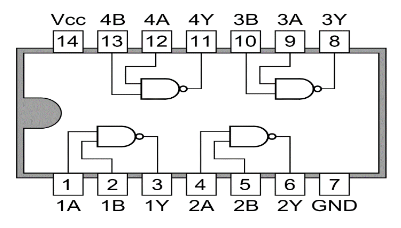
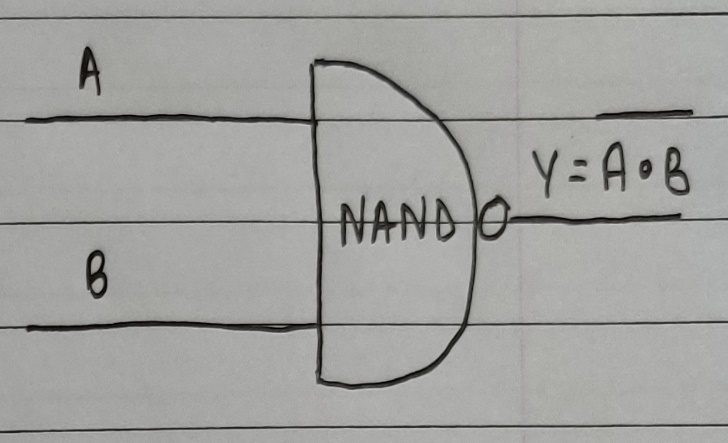
**Derived Gates/Universal Gates**

* + **NAND gate**
  + **NOR gate**
  + **EX-OR gate**
  + **EX-NOR gate**

1. **NAND gate: This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.**

**Y= (A\*B)’**

**Symbol Pin Diagram for IC 7400**

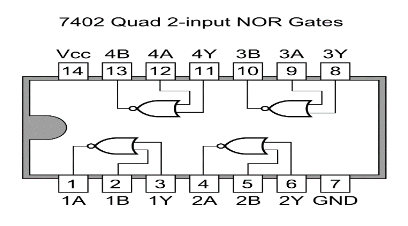
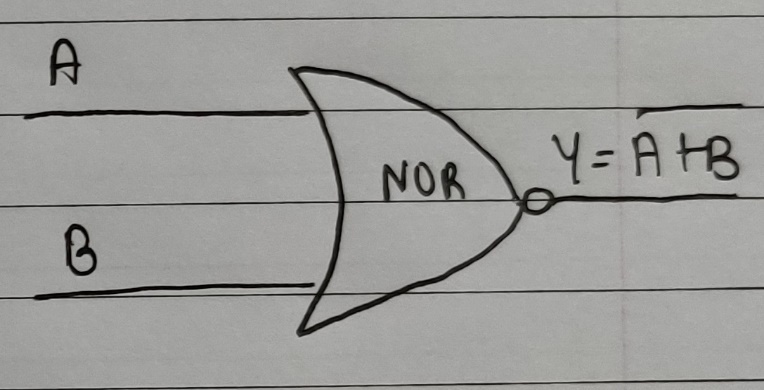


**The truth table for NAND operations is:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y= (A\*B)’** |
| **1** | **1** | **0** |
| **1** | **0** | **1** |
| **0** | **1** | **1** |
| **0** | **0** | **1** |

1. **NOR gate: This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if any of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.**

**Y= (A+B)’**



**Symbol for NOR gate Pin Diagram For IC 7402**

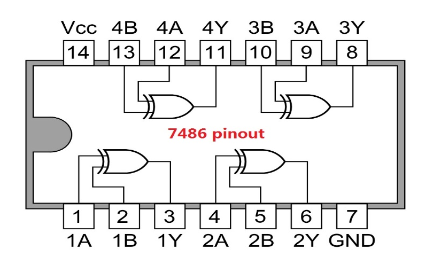
**The truth table for NOR operations are:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y= (A+B ) ‘** |
| **1** | **1** | **0** |
| **1** | **0** | **0** |
| **0** | **1** | **0** |
| **0** | **0** | **1** |

****

1. **EX-OR gate: The 'Exclusive-OR' gate is a circuit which will give a high output if either, but not both, of its two inputs are high. An encircled plus sign ( ) is used to show the EX-OR operation**

**Y= A  B**



**Symbol for Ex-OR gate Pin Diagram For IC 7486**

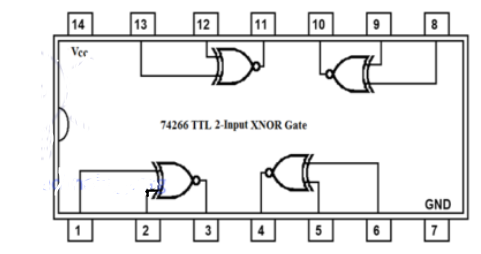
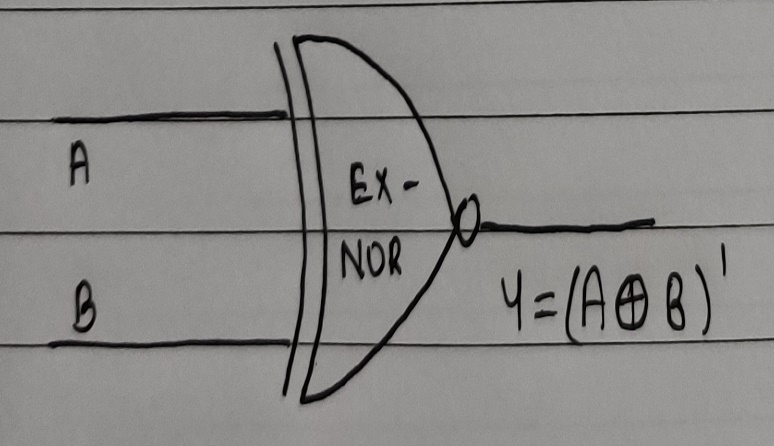
**The truth table for XOR operations is:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y= A  B** |
| **1** | **1** | **0** |
| **1** | **0** | **1** |
| **0** | **1** | **1** |
| **0** | **0** | **0** |

****

1. **EX-NOR gate: The 'Exclusive-NOR' gate circuit does the opposite to the EOR gate. It will give a low output if either, but not both, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion**

**Y= ( A  B ) ‘**

**Symbol for Ex-NOR gate Pin Diagram For IC 74266**

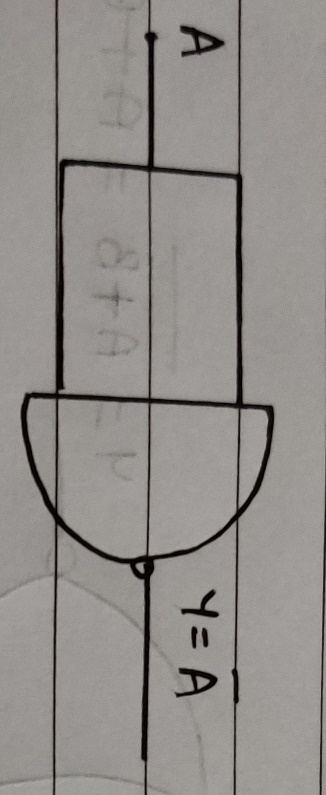
**The truth table for XNOR operations is:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y= (A  B )’** |
| **1** | **1** | **1** |
| **1** | **0** | **0** |
| **0** | **1** | **0** |
| **0** | **0** | **1** |

**Implementation Using NAND Gate STEPS**

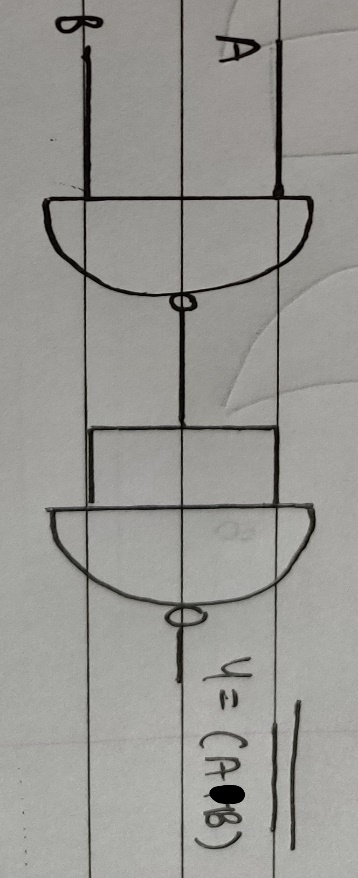
**NOT GATE**

|  |
| --- |
| **1.We short both the inputs of NAND gate.**  **2. When both the inputs are same, the output is the inverse of it and a NOT gate is created.** |



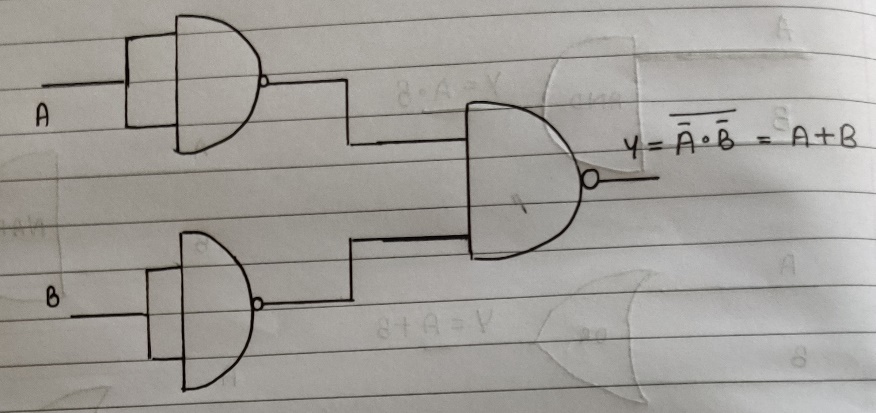
**AND GATE**

|  |
| --- |
| 1. **We can create an AND gate by using two NAND gates.** 2. **The first NAND gate returns LOW if both inputs are HIGH and returns HIGH if both inputs are anything else.** 3. **Then the second NAND gate is configured as a NOT gate to invert the output from the first NAND gate.** |



|  |
| --- |
| 1. **Two NAND gates are used, and the input terminal of each gate has been shorted.** 2. **This output is now given to another NAND gate and we get the desired output (i.e. OR gate is created) according to De Morgan’s Law.** |

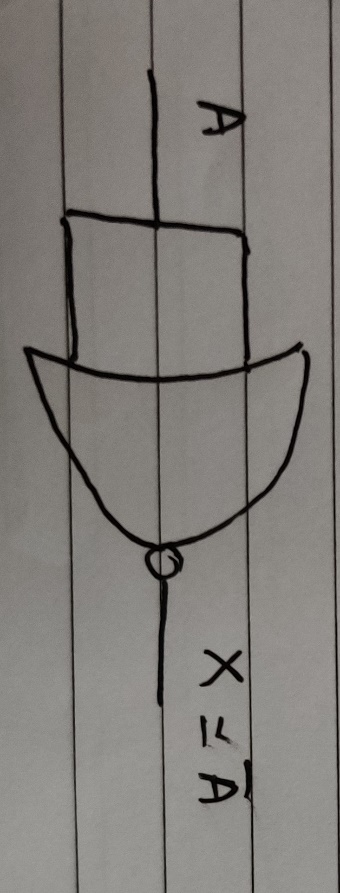
**OR GATE**



**IMPLEMENTATION USING NOR GATE STEPS**

**NOT GATE**

|  |
| --- |
| 1. **Both the input terminals of NOR gate are shorted.** 2. **When both the inputs are same, the output is the inverse of it and a NOT gate is created.** |



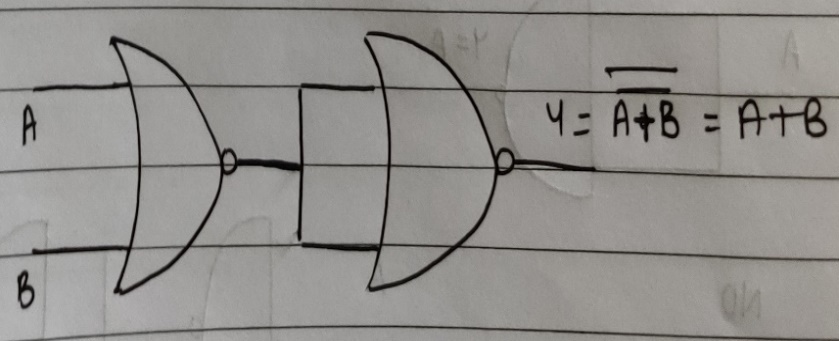
**AND GATE**

|  |
| --- |
| 1. **Two NOR gates are used and we short the input terminals of each gate.** 2. **This output is now given to another NOR gate and we get the desired output (i.e. AND gate is created) according to De Morgan’s Law.** |



**OR GATE**

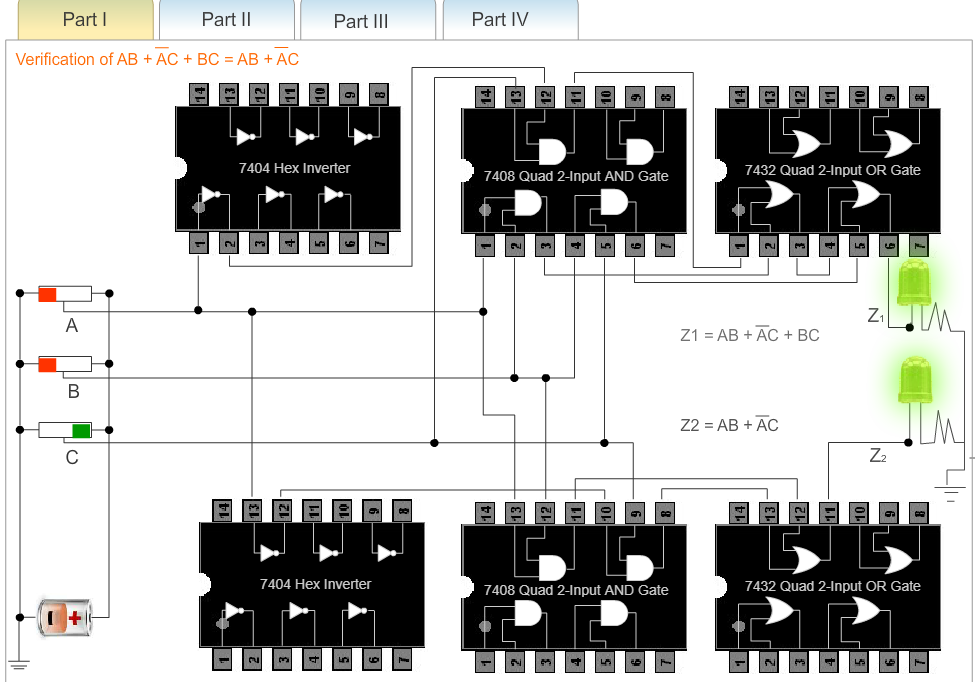
|  |
| --- |
| 1. **We can create an OR gate by using two NOR gates.** 2. **The first NOR gate returns HIGH if both inputs are LOW and returns LOW if both inputs are anything else.** 3. **Then the second NOR gate is configured as a NOT gate to invert the output from the first NOR gate.** |



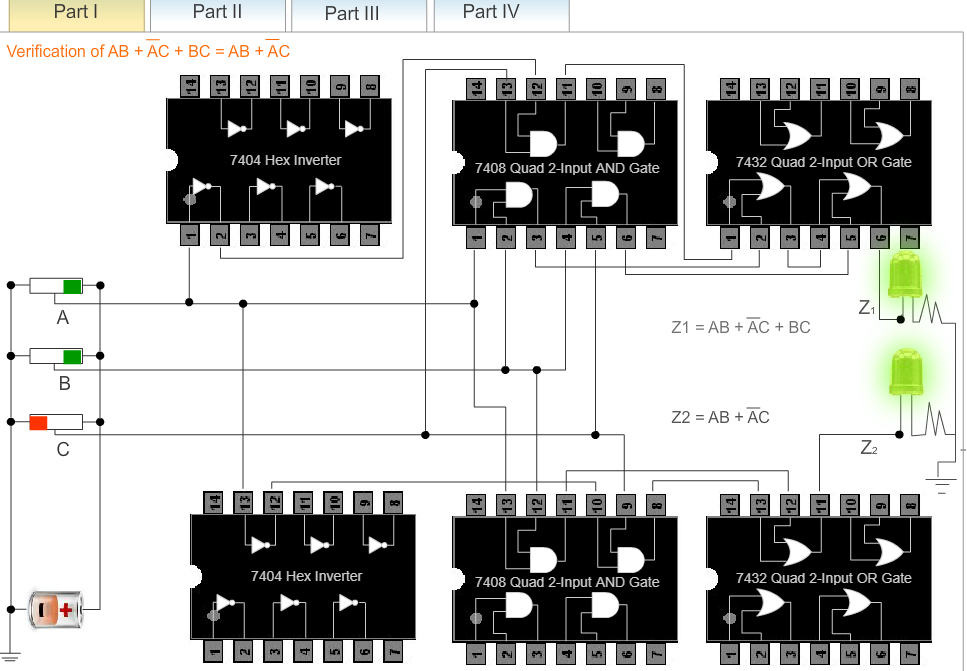
**PART 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A’** | **AB** | **A’C** | **BC** | **AB+A’C+ BC** | **AB+A’C** |
| **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **1** |

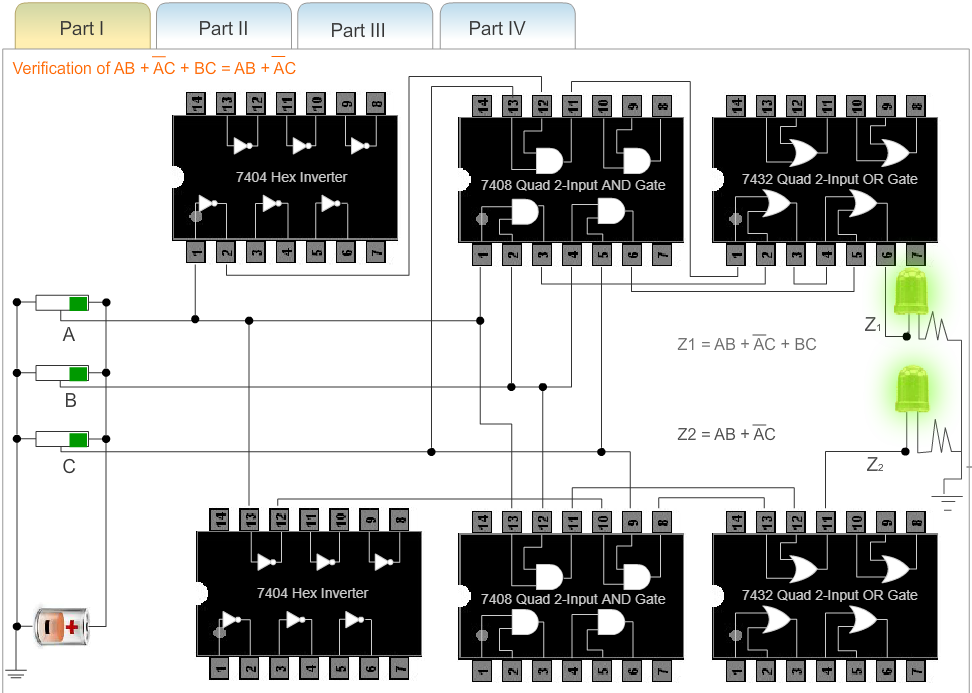
1. **A= 0 B= 0 C= 1**



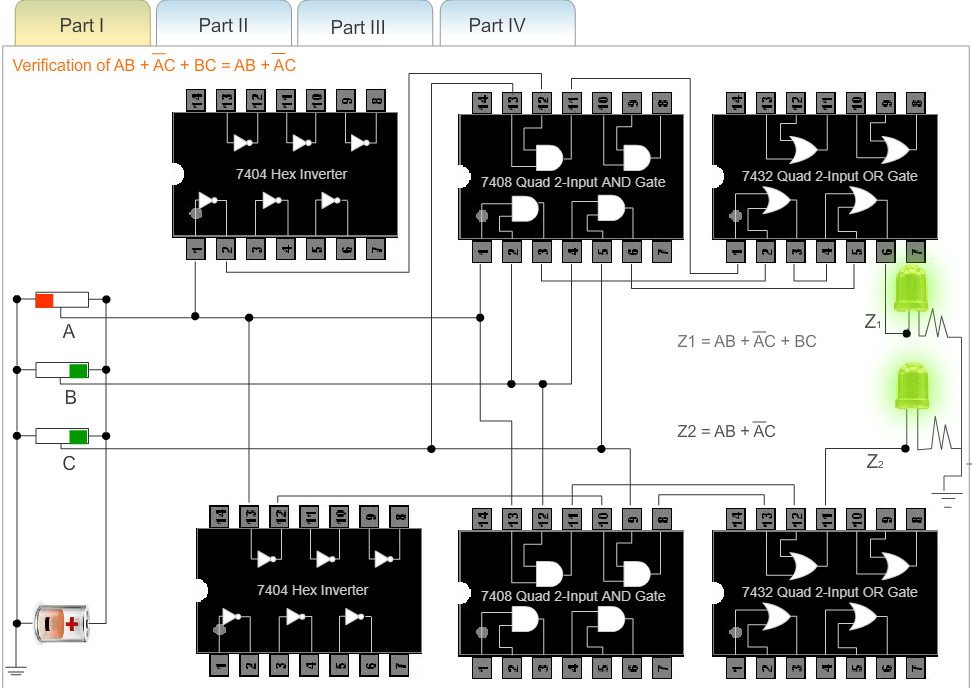
1. **A = 1 B =1 C = 0**



1. **A = 1 B = 1 C = 1**



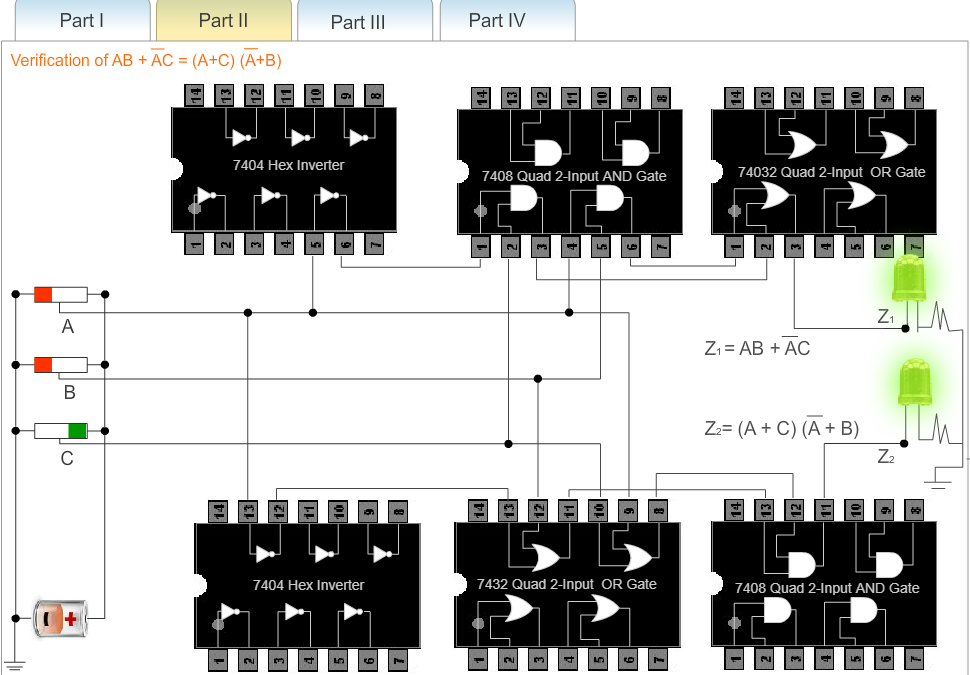
1. **A = 0 B = 1 C = 1**

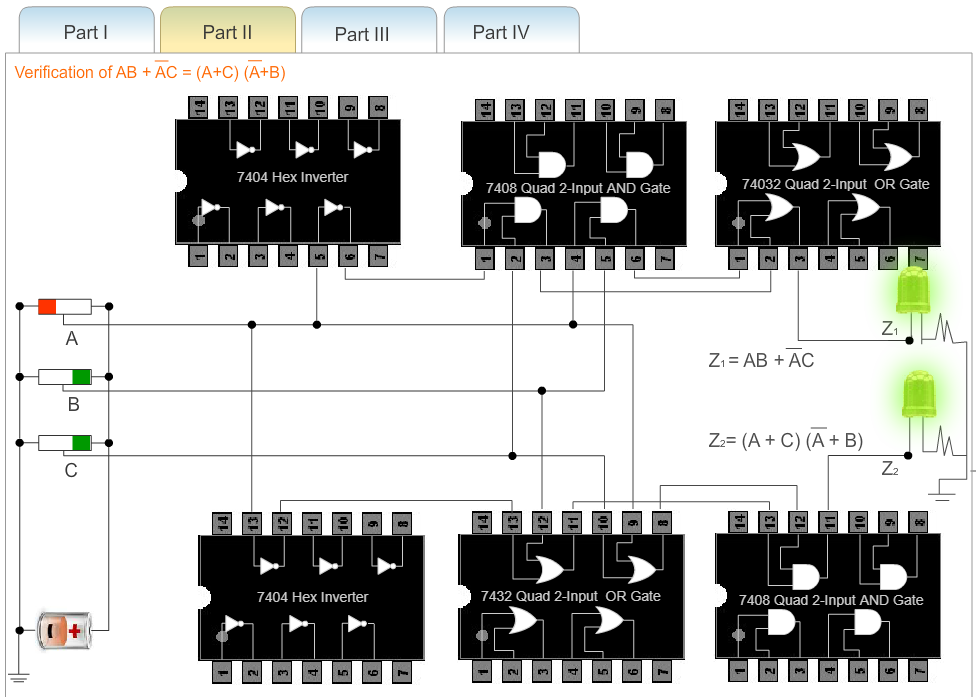


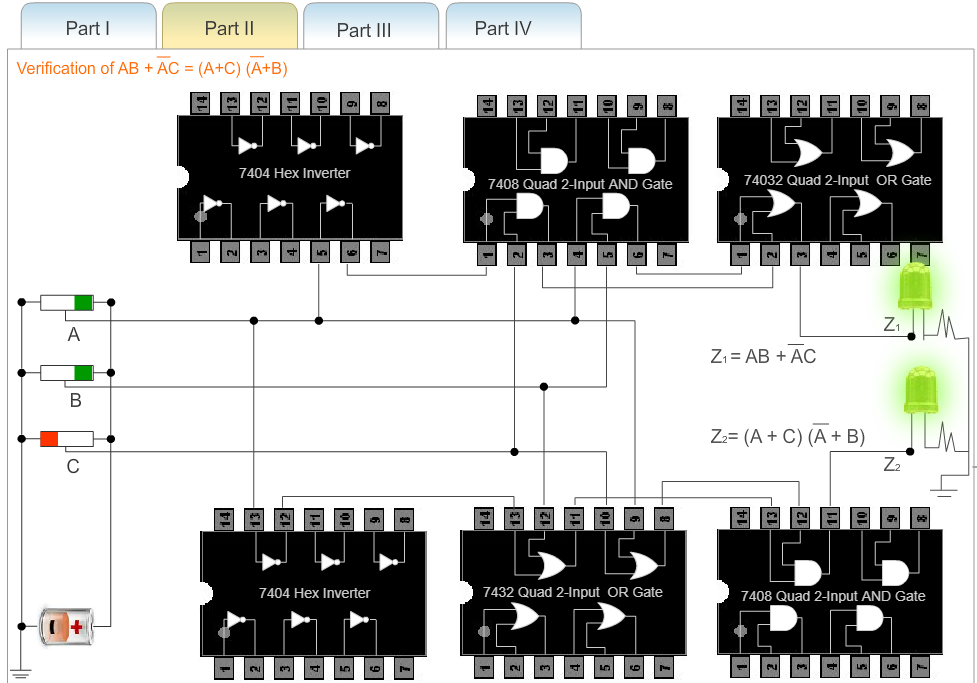
**PART 2**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **A’** | **AB** | **A+C** | **A’+B** | **A’C** | **AB+A’C** | **(A+C)(A’+B)** |
| **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** |

1. **A = 0 B = 0 C = 1**



1. **A = 0 B = 1 C = 1**
2. **A = 1 B = 1 C = 0**

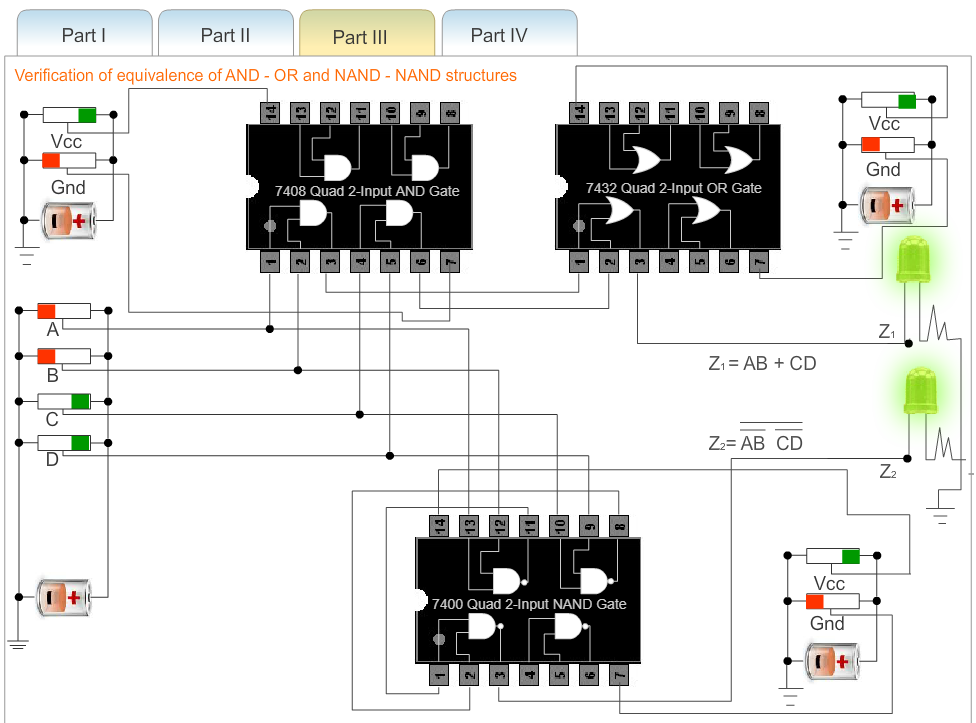
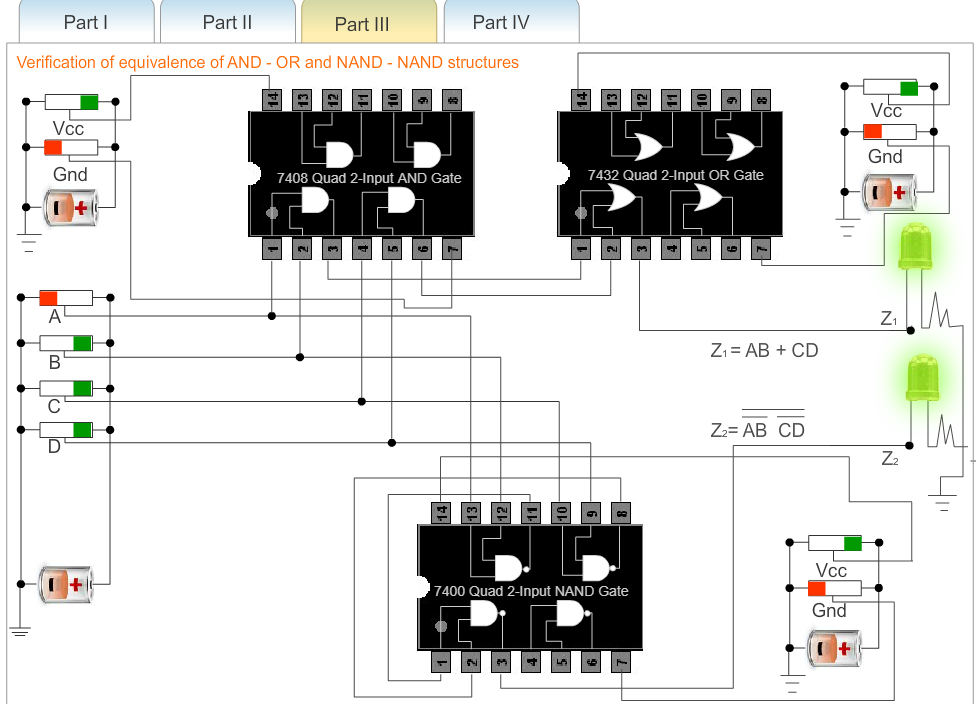


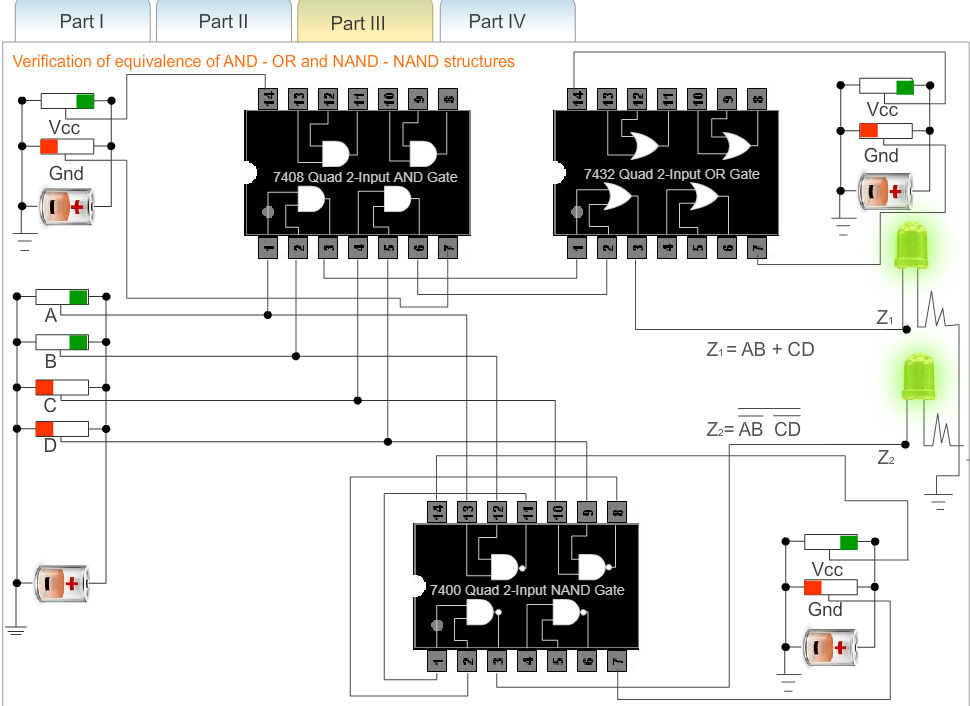
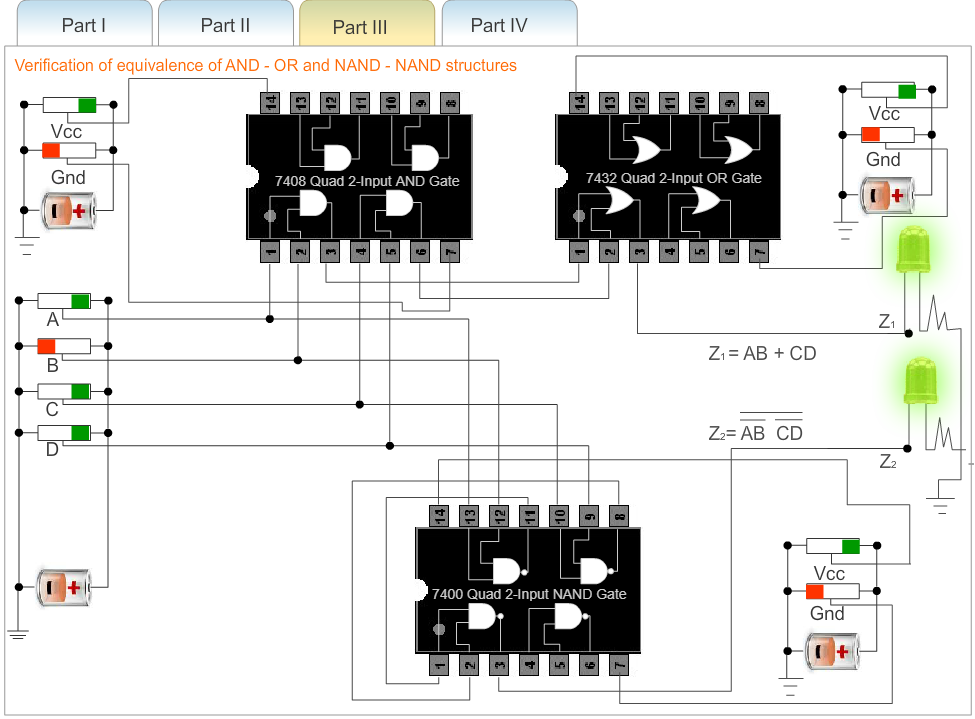


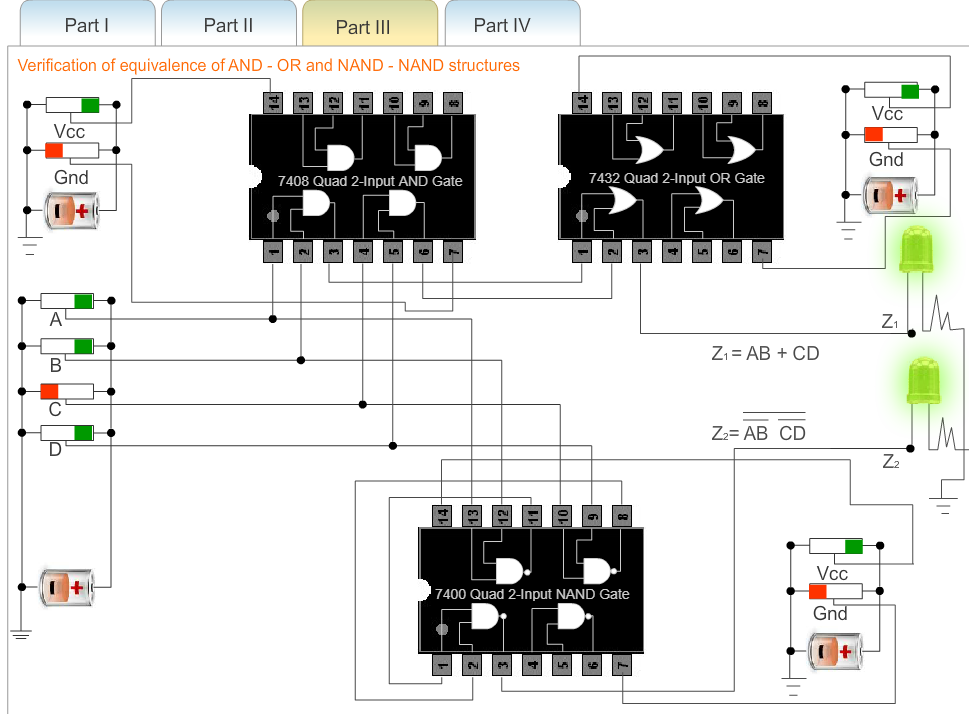
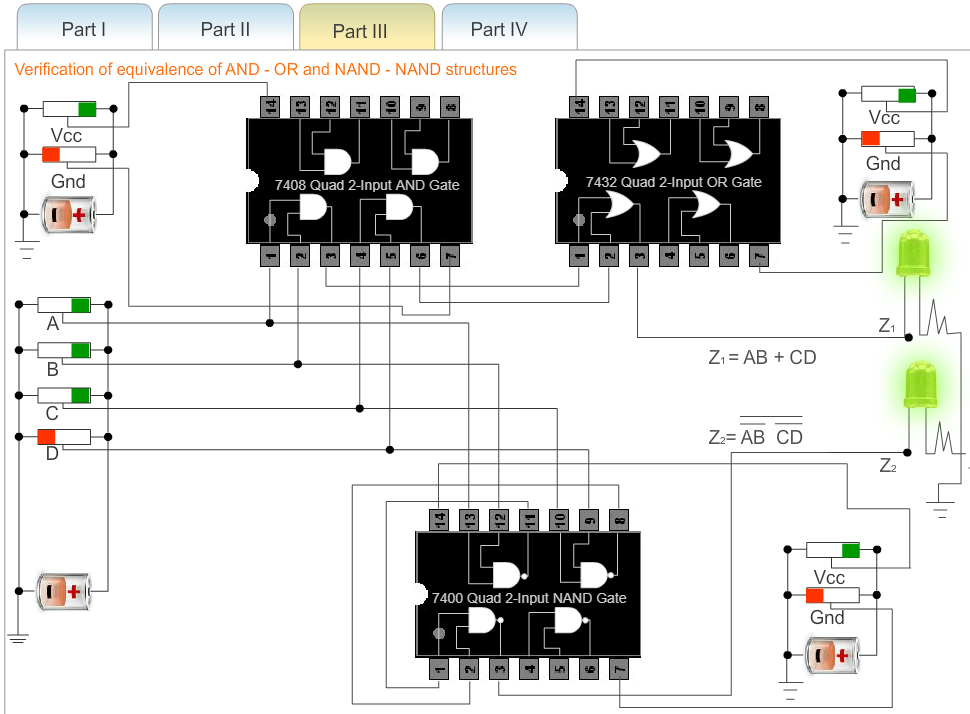
1. **A = 1 B = 1 C = 1**

**PART 3**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **AB** | **CD** | **(AB)’** | **(CD)’** | **(AB)+(CD)** | **((AB)’(CD)’)’** |
| **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** |  | **1** | **1** |

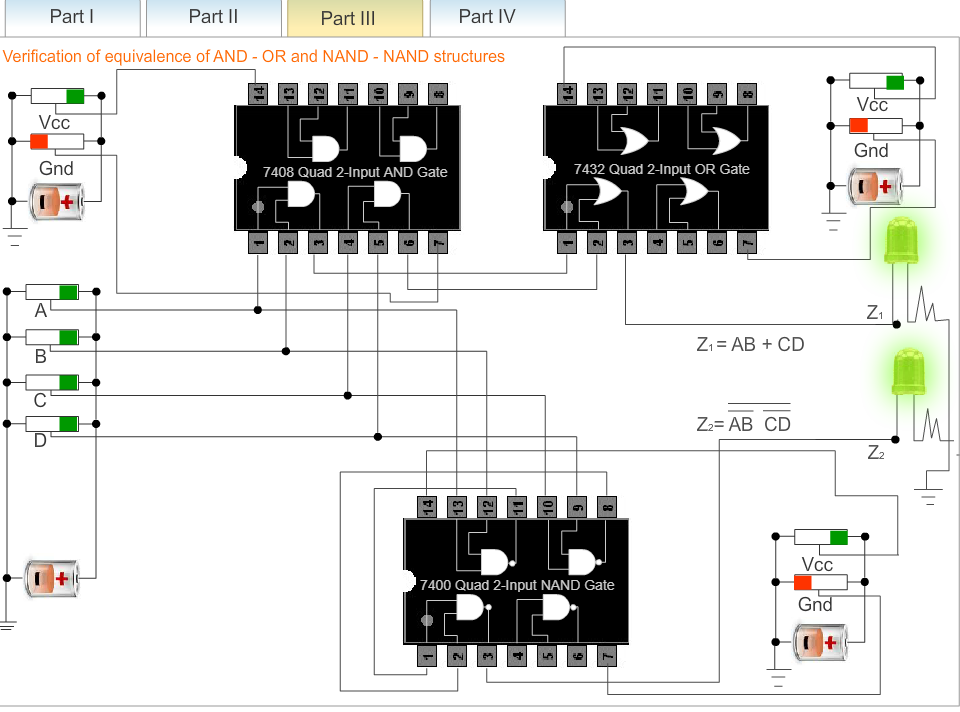
1. **A = 0 B = 0 C = 1 D = 1**
2. **A = 0 B = 1 C = 1 D = 1**

**c) A = 1 B = 0 C = 1 D = 1** **d) A = 1 B = 1 C = 0 D = 0**

**e) A = 1 B = 1 C = 0 D = 1****.**

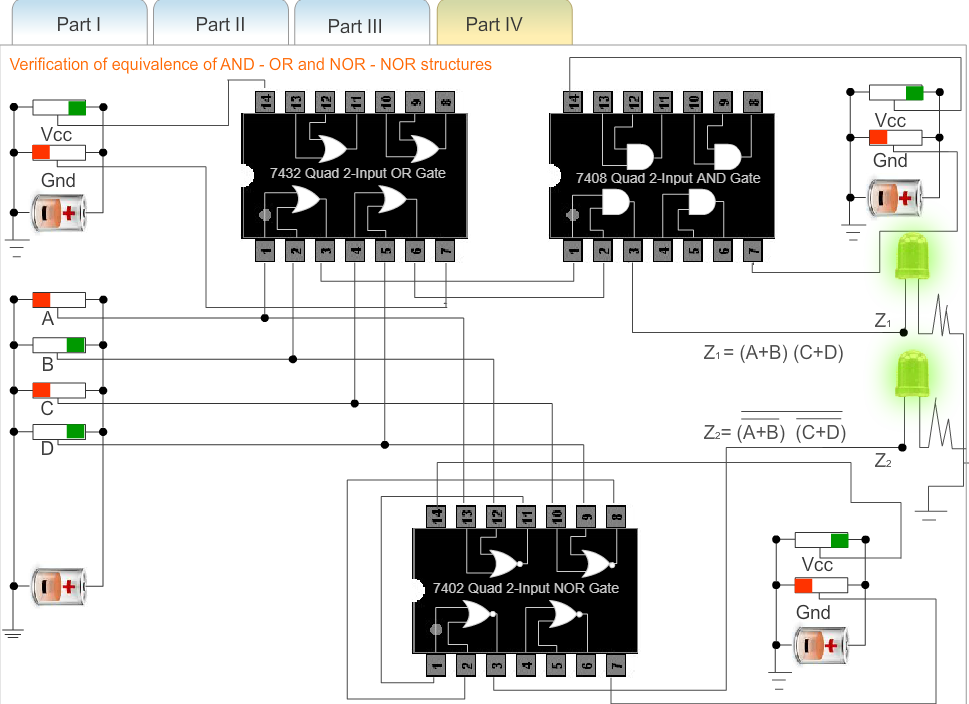
**f) A = 1 B = 1 C = 1 D = 0**

1. **A = 1 B = 1 C = 1 D = 1**

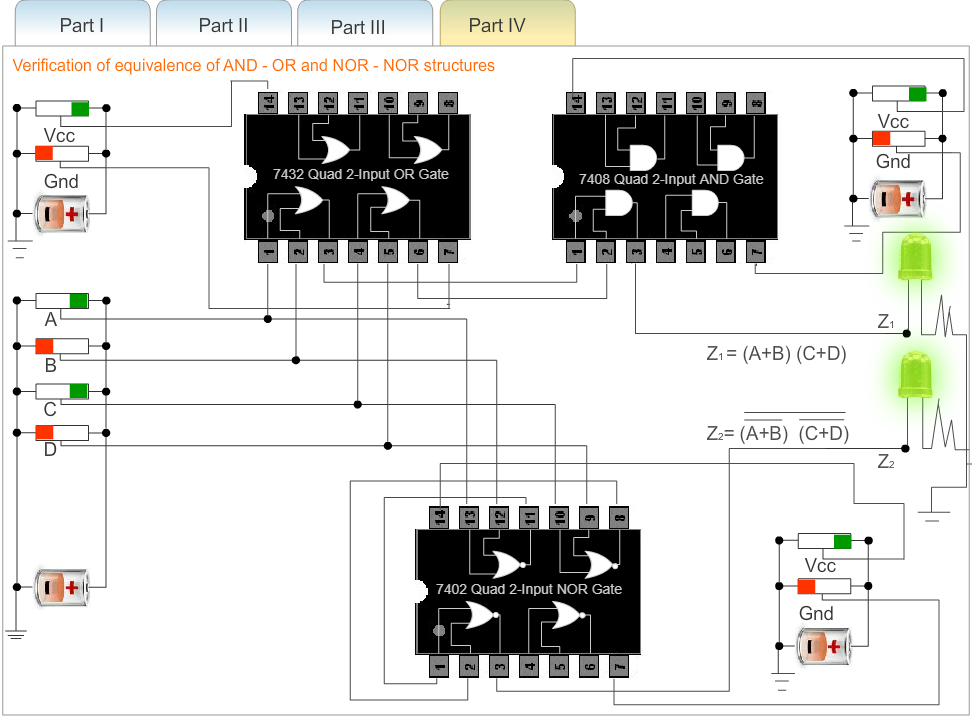


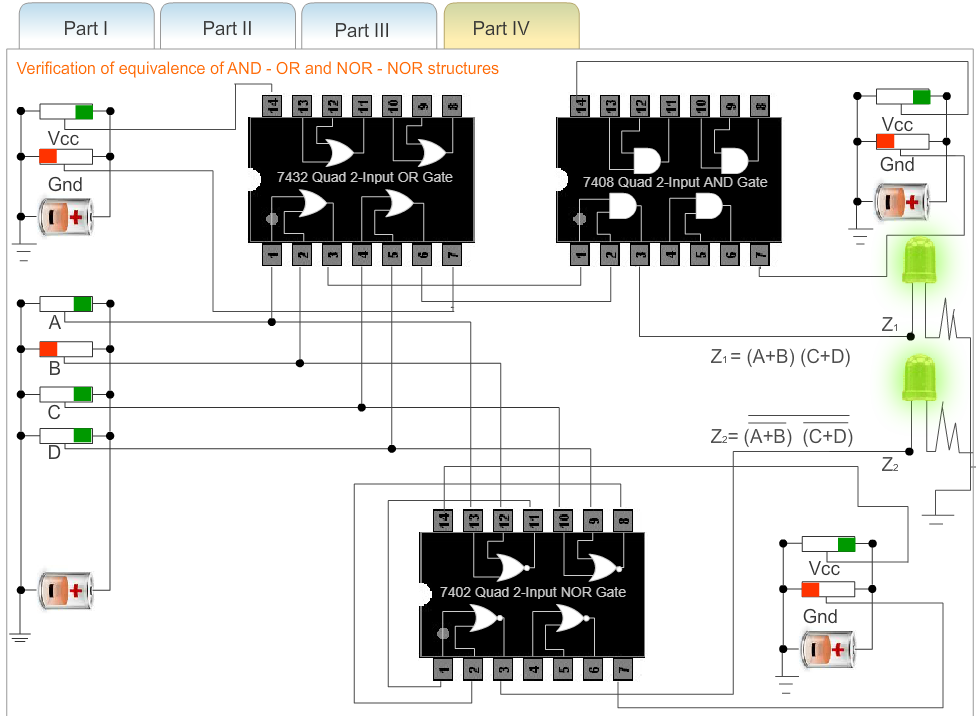
**PART 4**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **A+B** | **C+D** | **(A+B)’** | **(C+D)’** | **(A+B)(C+D)** | **((A+B)’+(C+D)’)’** |
| **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** |
| **0** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** |
| **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |



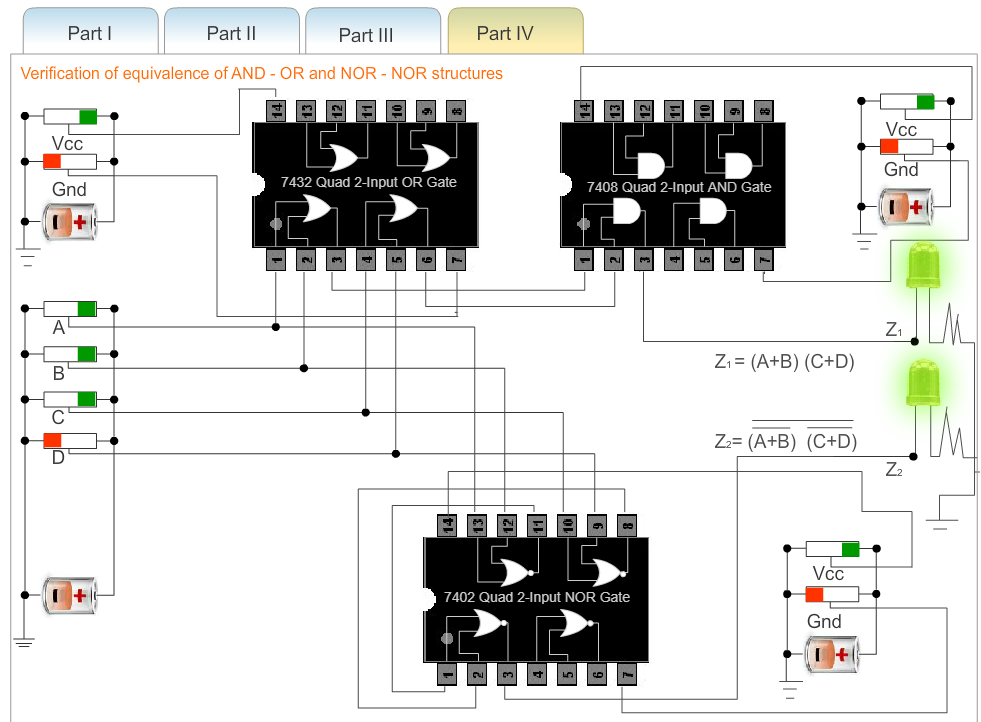
1. **A = 0 B = 1 C = 0 D = 1**

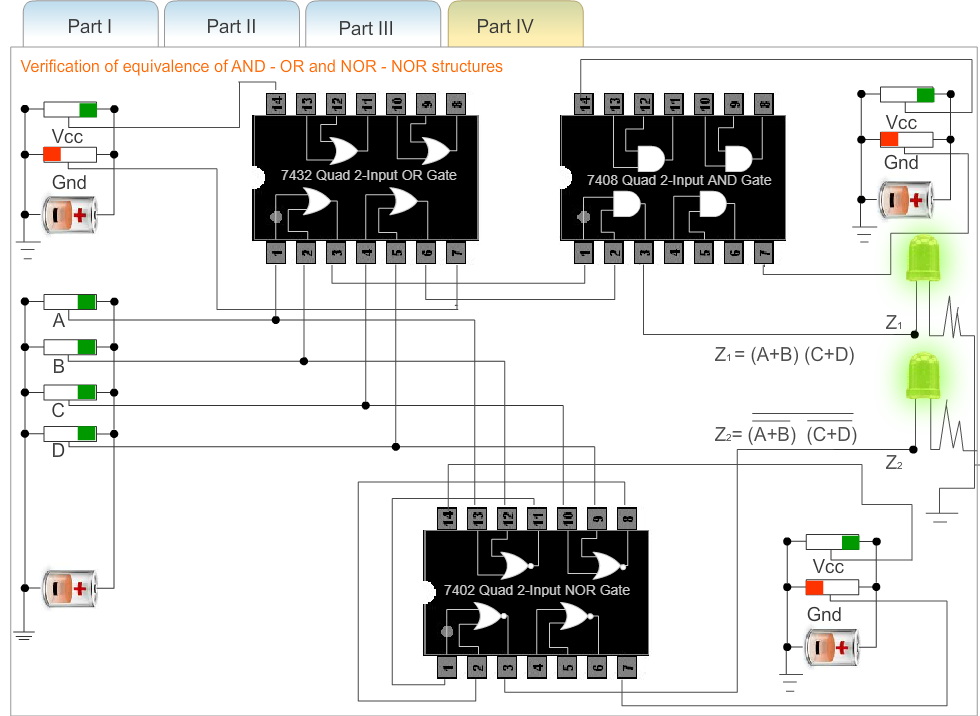
**b) A = 1 B = 0 C = 1 D = 0**

**c) A = 1 B = 0 C = 1 D = 1**

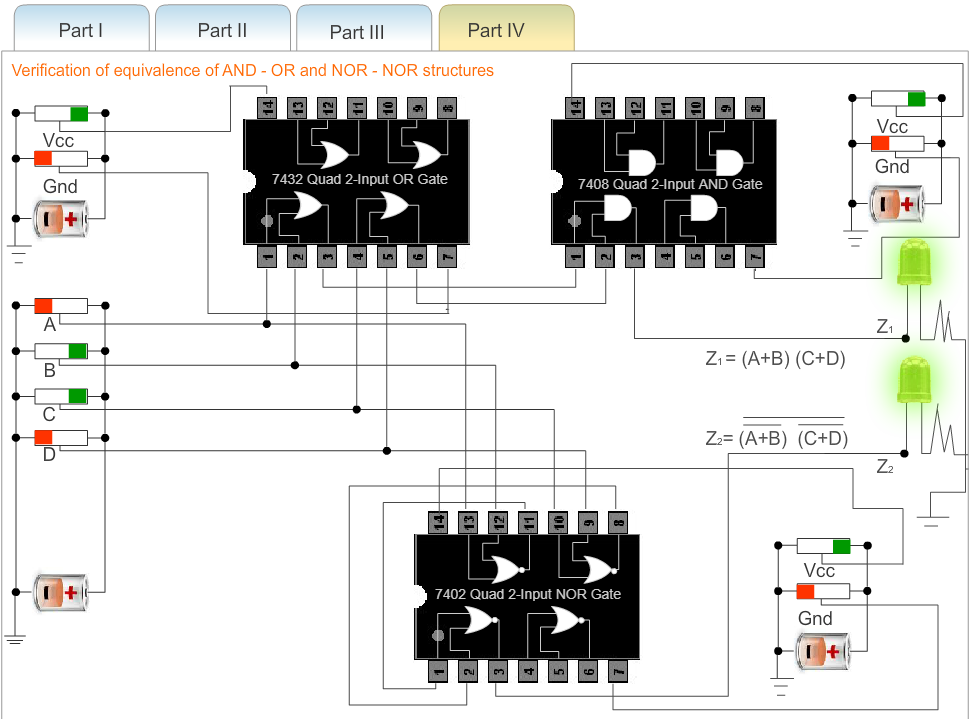


**d) A = 1 B = 1 C = 0 D = 1**

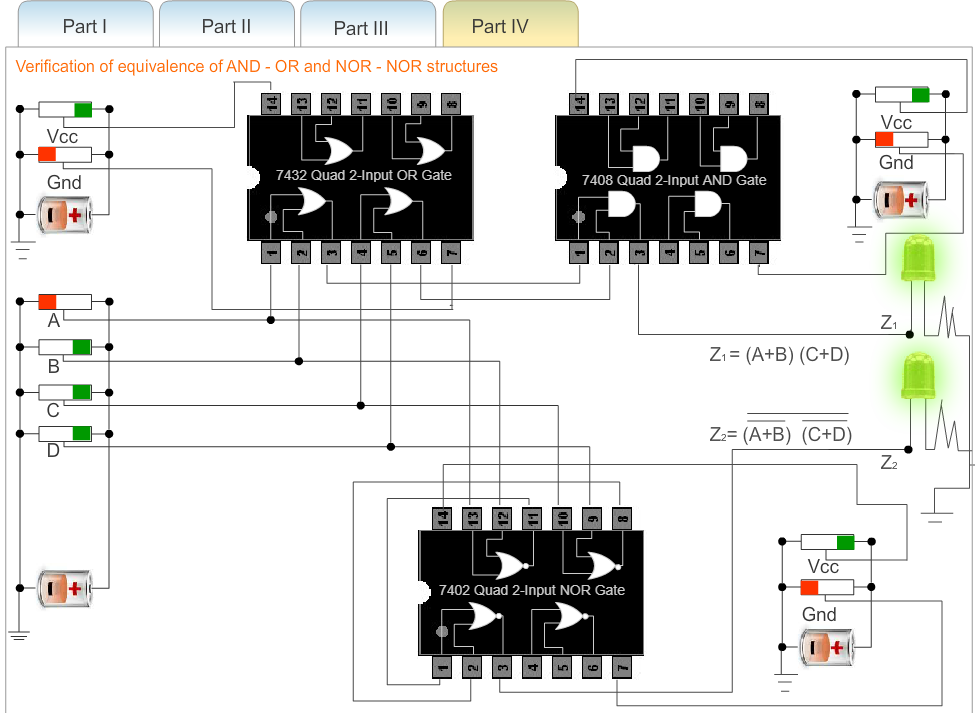
 **e) A = 1 B = 1 C = 1 D = 0**

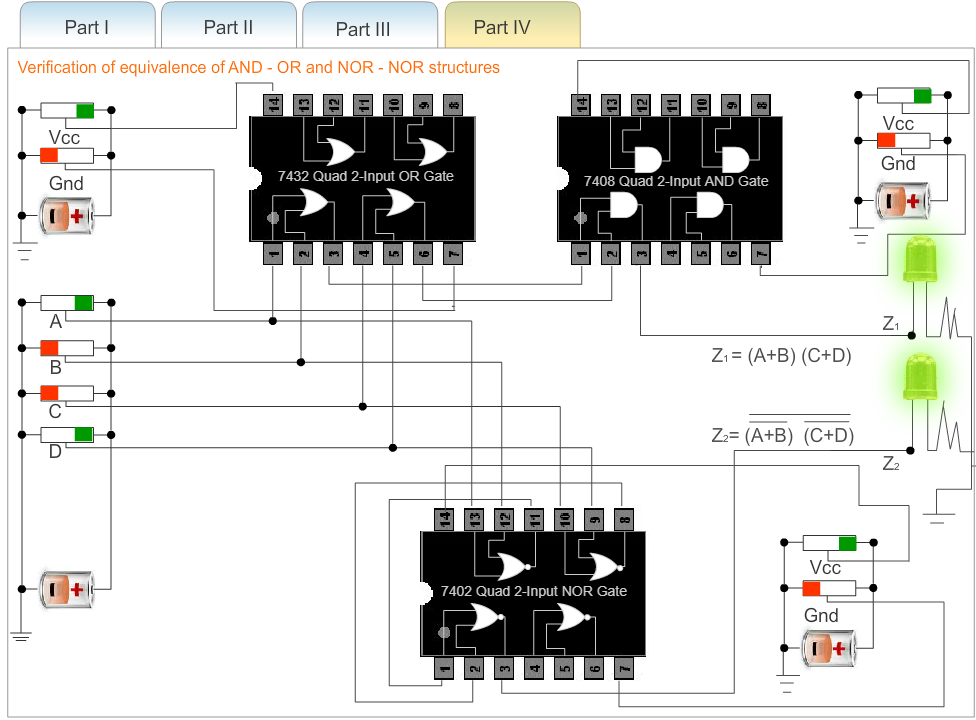


**f) A = 1 B = 1 C = 1 D = 1**



**g) A = 0 B = 1 C = 1 D = 0**

 **h) A = 0 B = 1 C = 1 D = 1**



1. **A = 1 B = 0 C = 1 D = 0**

**Conclusion: The basic gates and derived gates and their truth tables and symbols were studied . Using which Combination of all Logic gates was implemented and Boolean expressions using logic gates were studied using virtual lab.**

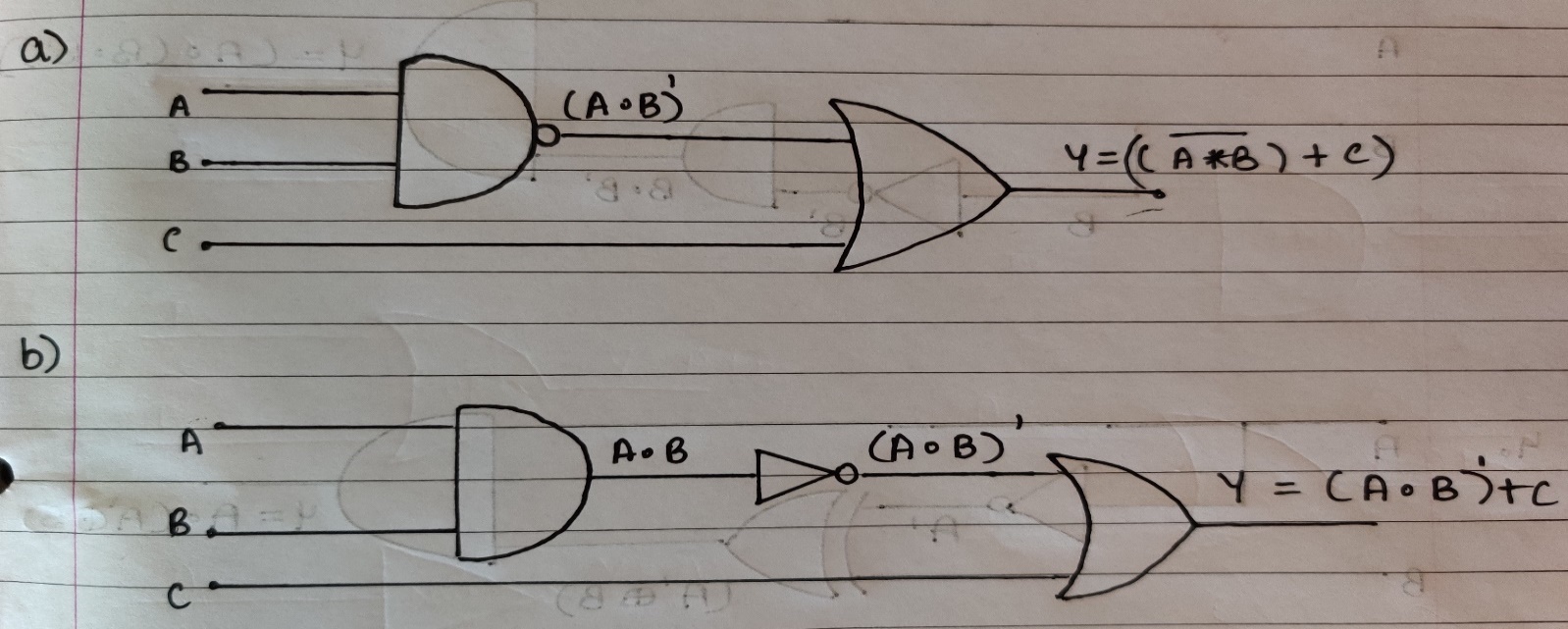
**Post Lab Questions**

1. **Verify the expression (A∙B)' + C by:**
2. **Using NAND Gate directly**

|  |  |  |
| --- | --- | --- |
| **(A.B)’** | **C** | **(A.B)’+C** |
| **1** | **1** | **1** |
| **1** | **0** | **1** |
| **0** | **1** | **1** |
| **0** | **0** | **0** |

1. **Using AND & NOT gate consecutively.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **(A.B)’** | **C** | **(A.B)’ +C** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** |

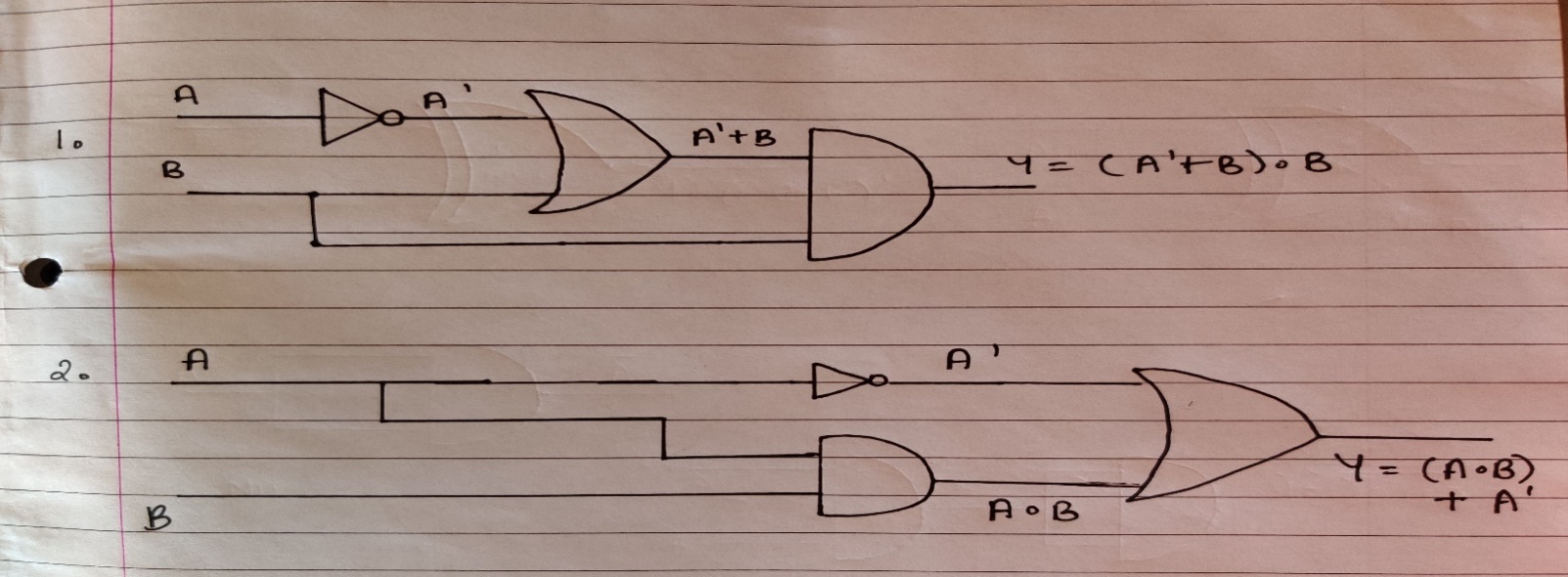


1. **Implement the following expressions using combination of gates:**
2. **(A'+B)∙B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **A’** | **A’+B** | **(A’+B) .B** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** |

1. **(A∙B)+A'**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **A’** | **A.B** | **(A.B) +A’** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **0** | **1** |



1. **A∙ (B∙B')**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **B’** | **B.B’** | **(A’+B) .B** |
| **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **0** |
| **0** | **1** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** |

1. **(A'⊕B)∙A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **A’** | **A’⊕B** | **(A'⊕B)∙A** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** |

