**Microprocessor Lab  
Lab Experiment No. 1**

Name: Ninad Rao Roll No. 53

**Aim**: Verifying the truth table of various logic gates.

**Gate selection**:

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a certain logic. Based on this, it is divided into two types:

**Generating the requirement for the gate**:

Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as you build them, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller subcircuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

Requirements:

1. Logisim requires Java 5 or later. If you do not already have it on your computer, Java is available from [java.sun.com](http://java.sun.com/).
2. Download Logisim from [Logisim's SourceForge.net page](http://sourceforge.net/projects/circuit/). You will have three choices of which release to download.
   1. A .jar file - runs on any platform, though not necessarily conveniently.
   2. A MacOS .tar.gz file
   3. A Windows .exe file

To execute the program:

1. **With the generic .jar file**: On Windows and MacOS systems, you will likely be able to start Logisim by double-clicking the JAR file. If that doesn't work, or if you use Linux or Solaris, you can type ``java -jar logisim-XX.jar'' at the command line.
2. **With the MacOS X version**: Once the downloaded .tar.gz version is uncompressed (this will likely happen automatically), just double-click the Logisim icon to start. You may want to place the icon into the Applications folder.
3. **With the Windows version**: Just double-click the Logisim icon. You may want to create a shortcut on the desktop and/or in the Start menu to make starting Logisim easier.

**Basic gates**: These gates are the basic building blocks in the digital IC’s (integrated circuits). The basic logic gates are used to perform fundamental logic functions. Examples of basic gates are:

1. **AND gate**: The AND gate is a basic digital logic gate that implements logical conjunction.

**Statement**:

Input 1 = A

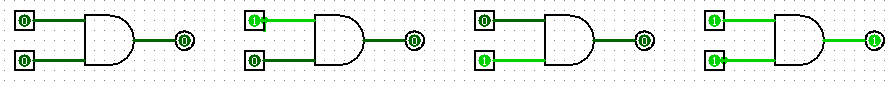
Input 2 = B

Output = A . B

**Symbol**:



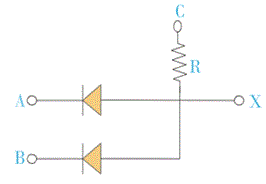
**Working of AND gate**: A HIGH output results only if all the inputs to the AND gate are HIGH. If none or not all inputs to the AND gate are HIGH, LOW output results. The function can be extended to any number of inputs.



**Truth Table**:

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

**Circuit Diagram**:



2. **OR gate**: The OR gate is a digital [logic gate](https://en.wikipedia.org/wiki/Logic_gate) that implements [logical disjunction](https://en.wikipedia.org/wiki/Logical_disjunction).

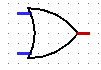
**Statement**:

Input 1 = A

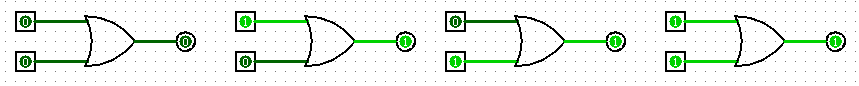
Input 2 = B

Output = A + B

**Symbol**:



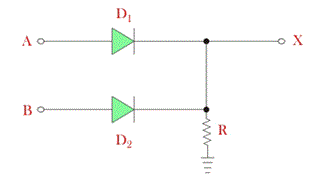
**Working of OR gate**: A HIGH output results if one or both the inputs to the gate are HIGH. If neither input is high, a LOW output results. In another sense, the function of OR effectively finds the maximum between two binary digits, just as the complementary [AND function](https://en.wikipedia.org/wiki/AND_gate) finds the minimum.



**Truth Table**:

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

**Circuit Diagram**:



3. **NOT gate**: In digital logic, an inverter or NOT gate is a [logic gate](https://en.wikipedia.org/wiki/Logic_gate) which implements [logical negation](https://en.wikipedia.org/wiki/Logical_negation).

**Statement**:

Input = A

Output = A’

**Symbol**:



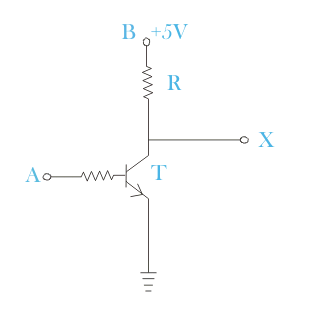
**Working of NOT gate**: A HIGH output results if the input is LOW and LOW output results if the input is HIGH.



**Truth Table**:

|  |  |
| --- | --- |
| **A** | **C = A’** |
| 0 | 1 |
| 1 | 0 |

**Circuit Diagram**:



**Universal gates**: A universal logic gate is a logic gate that can be used to construct all other logic gates. A universal gate is a gate which can implement any Boolean function without need to use any other gate type.

1. **NAND gate**: In [digital electronics](https://en.wikipedia.org/wiki/Digital_electronics), a NAND gate (NOT-AND) is a [logic gate](https://en.wikipedia.org/wiki/Logic_gate) which produces an output which is false only if all its inputs are true. Thus its output is [complement](https://en.wikipedia.org/wiki/Complement_(set_theory)) to that of an [AND gate](https://en.wikipedia.org/wiki/AND_gate).

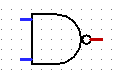
**Statement**:

Input 1 = A

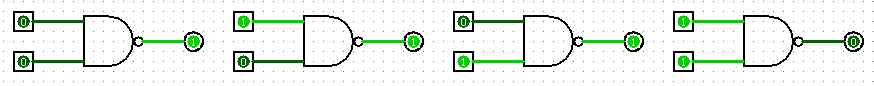
Input 2 = B

Output = (A . B)’

**Symbol**:



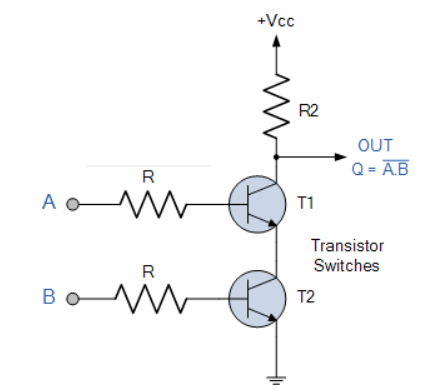
**Working of NAND gate:** A LOW output results only if all the inputs to the gate are HIGH. If any input is LOW, a HIGH output results.

****

**Truth Table**:

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

**Circuit Diagram**:



2. **NOR gate**: The NOR gate is a digital [logic gate](https://en.wikipedia.org/wiki/Logic_gate) that implements [logical NOR](https://en.wikipedia.org/wiki/Logical_NOR). NOR is the result of the [negation](https://en.wikipedia.org/wiki/Negation) of the [OR](https://en.wikipedia.org/wiki/OR_gate) operator. It can also in some senses be seen as the inverse of an [AND gate](https://en.wikipedia.org/wiki/AND_gate).

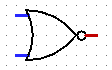
**Statement**:

Input 1 = A

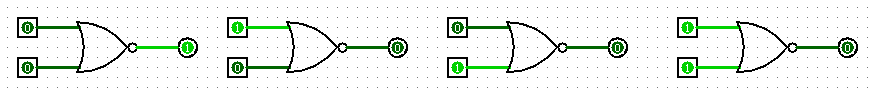
Input 2 = B

Output = (A + B)’

**Symbol**:



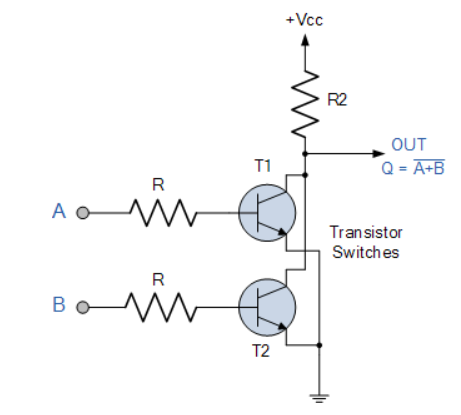
**Working of NOR gate**: A HIGH output results if both the inputs to the gate are LOW. If one or both input is HIGH, a LOW output results.



**Truth Table**:

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

**Circuit Diagram**:



**Conclusion**: Thus, we have studied and verified the truth tables of various logic gates.