**Objective:**

* Knowing about the Universal Gates(NAND & NOR)
* Creating basic gates using universal gates differently
* Making Boolean function using NAND and OR gate
* Knowing gate level minimization.

**List of Equipment**

* Trainer board
* IC 7400 quadruple 2-input NAND gates
* IC 7402 quadruple 2 input NOR gates
* Wires.

**Theory:**

NAND gate:

A NAND gate is an inverted AND gate. It has the following truth table:

|  |
| --- |
| [NAND ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:NAND_ANSI_Labelled.svg) |
| **Q** = NOT( **A** AND **B** )   |  |  |  |  | | --- | --- | --- | --- | | **Truth Table** | | | | | **Input A** | **Input B** |  | **Output Q** | | 0 | 0 |  | 1 | | 0 | 1 |  | 1 | | 1 | 0 |  | 1 | | 1 | 1 |  | 0 | | |

Making other gates with NAND gate:

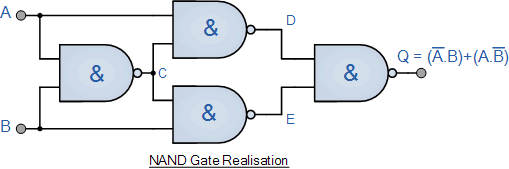
A NAND gate is a universal gate, meaning that any other gate can be represented as a combination of NAND gates.

XOR

The Exclusive-OR logic function is a very useful circuit that can be used in many different types of computational circuits. Although not a basic logic gate in its own right, its usefulness and versatility has turned it into a standard logical function complete with its own Boolean expression, operator and symbol.

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Truth Table | | |
| 2-input exclusive-or gate  XOR Gate | B | A | Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| Boolean Expression Q = A ⊕ B | A **OR** B but NOT **BOTH** gives Q | | |

### XOR Function Realization using NAND gates

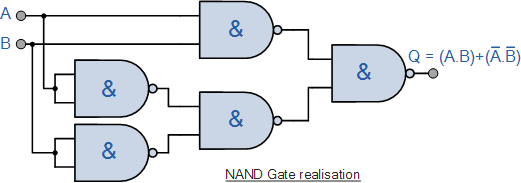


### XNOR Gate

Ex-NOR gates are used mainly in electronic circuits that perform arithmetic operations and data checking such as **Adders**,  **substractors** or **Parity Checkers**, etc. As the XNOR gate gives an output of logic level “1” whenever its two inputs are equal it can be used to compare the magnitude of two binary digits or numbers and so XNOR  gates are used in Digital Comparator circuits.

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Truth Table | | |
| 2-input exclusive-nor gate  XNOR Gate | B | A | Q |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| Boolean Expression Q = A ⊕ B | Read if A **AND** B the **SAME** gives Q | | |

### XNOR Function Realization using NAND gates



### [NOT](https://en.wikipedia.org/wiki/NOT_gate)

A NOT gate is made by joining the inputs of a NAND gate together. Since a NAND gate is equivalent to an AND gate followed by a NOT gate, joining the inputs of a NAND gate leaves only the NOT gate.

|  |
| --- |
| **Truth Table** |
| |  |  |  | | --- | --- | --- | | **Input A** |  | **Output Q** | | 0 |  | 1 | | 1 |  | 0 |   **Desired NOT Gate** | **NAND Construction** |
| [NOT ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:NOT_ANSI_Labelled.svg) | [NOT from NAND.svg](https://en.wikipedia.org/wiki/File:NOT_from_NAND.svg) |
| **Q** = NOT( **A** ) | = **A** NAND **A** |
|  | |

### [AND](https://en.wikipedia.org/wiki/AND_gate)

An AND gate is made by following a NAND gate with a NOT gate as shown below. This gives a NOT NAND, i.e. AND.

|  |
| --- |
| **Truth Table** |
| |  |  |  |  | | --- | --- | --- | --- | | **Input A** | **Input B** |  | **Output Q** | | 0 | 0 |  | 0 | | 0 | 1 |  | 0 | | 1 | 0 |  | 0 | | 1 | 1 |  | 1 |   **Desired AND Gate** | **NAND Construction** |
| [AND ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:AND_ANSI_Labelled.svg) | [AND from NAND.svg](https://en.wikipedia.org/wiki/File:AND_from_NAND.svg) |
| **Q** = **A** AND **B** | = ( **A** NAND **B** ) NAND ( **A** NAND **B** ) |
|  | |

### OR

If the truth table for a NAND gate is examined or by applying [De Morgan's Laws](https://en.wikipedia.org/wiki/De_Morgan%27s_Laws), it can be seen that if any of the inputs are 0, then the output will be 1. To be an OR gate, however, the output must be 1 if any input is 1. Therefore, if the inputs are inverted, any high input will trigger a high output.

|  |
| --- |
| **Truth Table** |
| |  |  |  |  | | --- | --- | --- | --- | | **Input A** | **Input B** |  | **Output Q** | | 0 | 0 |  | 0 | | 0 | 1 |  | 1 | | 1 | 0 |  | 1 | | 1 | 1 |  | 1 |   **Desired OR Gate** | **NAND Construction** |
| [OR ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:OR_ANSI_Labelled.svg) | [OR from NAND.svg](https://en.wikipedia.org/wiki/File:OR_from_NAND.svg) |
| **Q** = **A** OR **B** | = ( **A** NAND **A** ) NAND ( **B** NAND **B** ) |
|  | |

NOR:

A NOR gate is logically an inverted OR gate. By itself has the following truth table:

|  |
| --- |
| [NOR ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:NOR_ANSI_Labelled.svg) |
| **Q** = NOT( **A** OR **B** )   |  |  |  |  | | --- | --- | --- | --- | | **Truth Table** | | | | | **Input A** | **Input B** |  | **Output Q** | | 0 | 0 |  | 1 | | 0 | 1 |  | 0 | | 1 | 0 |  | 0 | | 1 | 1 |  | 0 | | |
|  | |

Making other gates using NOR gate:

A NOR gate is a universal gate, meaning that any other gate can be represented as a combination of NOR gates.

### [NOT](https://en.wikipedia.org/wiki/NOT_gate)

This is made by joining the inputs of a NOR gate. As a NOR gate is equivalent to an OR gate leading to NOT gate, this automatically sees to the "OR" part of the NOR gate, eliminating it from consideration and leaving only the NOT part.

|  |
| --- |
| **Truth Table** |
| |  |  |  | | --- | --- | --- | | **Input A** |  | **Output Q** | | 0 |  | 1 | | 1 |  | 0 |   **Desired NOT Gate** | **NOR Construction** |
| [NOT ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:NOT_ANSI_Labelled.svg) | [NOT from NOR.svg](https://en.wikipedia.org/wiki/File:NOT_from_NOR.svg) |
| **Q** = NOT( **A** ) | = **A** NOR **A** |
|  | |

### [OR](https://en.wikipedia.org/wiki/OR_gate)

The OR gate is simply one NOR gate followed by a second whose inputs are joined.

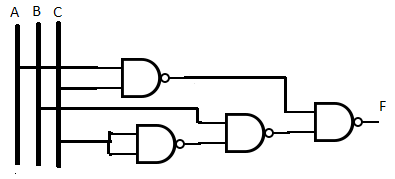
|  |
| --- |
| **Truth Table** |
| |  |  |  |  | | --- | --- | --- | --- | | **Input A** | **Input B** |  | **Output Q** | | 0 | 0 |  | 0 | | 0 | 1 |  | 1 | | 1 | 0 |  | 1 | | 1 | 1 |  | 1 |   **Desired OR Gate** | **NOR Construction** |
| [OR ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:OR_ANSI_Labelled.svg) | [OR from NOR.svg](https://en.wikipedia.org/wiki/File:OR_from_NOR.svg) |
| **Q** = **A** OR **B** | = ( **A** NOR **B** ) NOR ( **A** NOR **B**) |
|  | |

### [AND](https://en.wikipedia.org/wiki/AND_gate)

An AND gate gives a 1 output when both inputs are 1; a NOR gate gives a 1 output only when both inputs are 0. Therefore, an AND gate is made by inverting the inputs of a NOR gate.

|  |
| --- |
| **Truth Table** |
| |  |  |  |  | | --- | --- | --- | --- | | **Input A** | **Input B** |  | **Output Q** | | 0 | 0 |  | 0 | | 0 | 1 |  | 0 | | 1 | 0 |  | 0 | | 1 | 1 |  | 1 |   **Desired AND Gate** | **NOR Construction** |
| [AND ANSI Labelled.svg](https://en.wikipedia.org/wiki/File:AND_ANSI_Labelled.svg) | [AND from NOR.svg](https://en.wikipedia.org/wiki/File:AND_from_NOR.svg) |
| **Q** = **A** AND **B** | = ( **A** NOR **A** ) NOR ( **B** NOR **B**) |
|  | |

**Circuit Diagram:**



**Data/Truth table:**

Experiment 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | I1=AC | I2=BC’ | F=I1+I2 |
| A | B | C |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |

**Discussion:**

First time we could not build our circuit. Because there were problems in IC. Then when we changed the IC it worked properly. Finally we were able to make the circuit.

