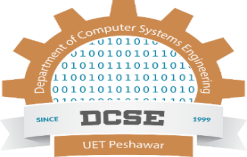
****

**University of engineering & technology Peshawar**

**Digital Logic &computer Design-lab**

**Lab report no#04**

**Fall 2020**

**Submitted by: Ashfaq Ahmad**

**Section: B**

**Reg No: 19PWCSE1795**

**Semester: 3rd**

**“On my honor, as a student of University of Engineering and Technology Peshawar, I have neither given nor received unauthorized assistance on this academic work”**

Student signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted to:**

**Eng: Abdullah Hamid**

**Department Of Computer System Engineering**

**TITLE:** **The UNIVERSAL GATES**

**AIM:**

* To study the realization of basic gates using universal gates (NAND & NOR).

**APPARATUS:**

* Power Supply, Breadboard.

**COMPONENTS:**

* ICs 7400 (quad 2-input NAND gate), 7402 (quad 2-input NOR gate), DIP Switch, LED.

**THEORY:**

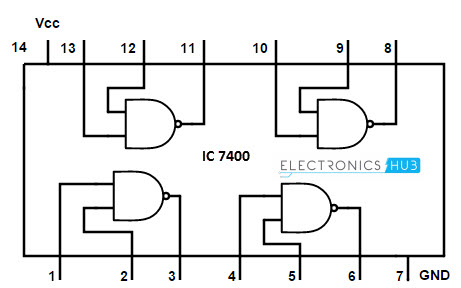
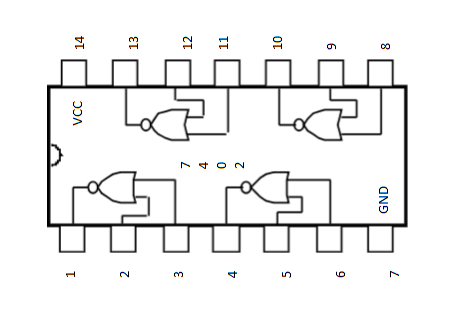
AND, OR, NOT are called basic gates as their logical operation cannot be simplified further.

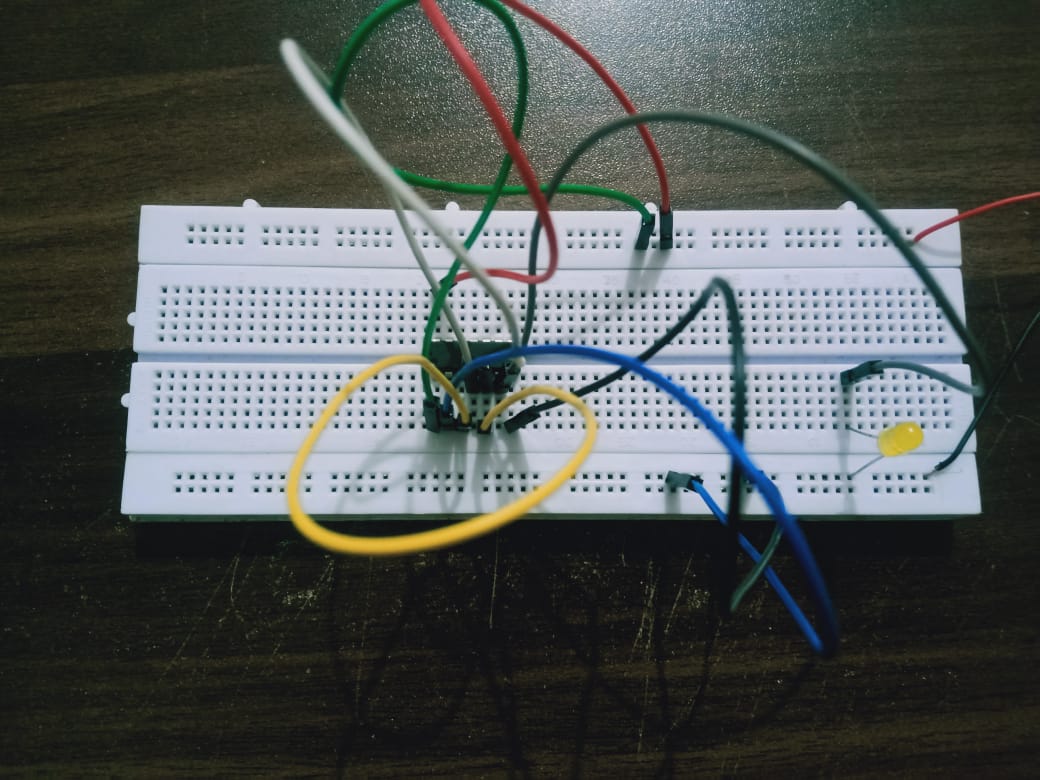
Universal Gate A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families. In fact, an AND gate is typically implemented as a NAND gate followed by an inverter not the other way around!! Likewise, an OR gate is typically implemented as a NOR gate followed by an inverter not the other way around. Using NAND and NOR gates and De-Morgan's Theorems different basic gates & EX-OR gates are realized.

**PROCEDURE**

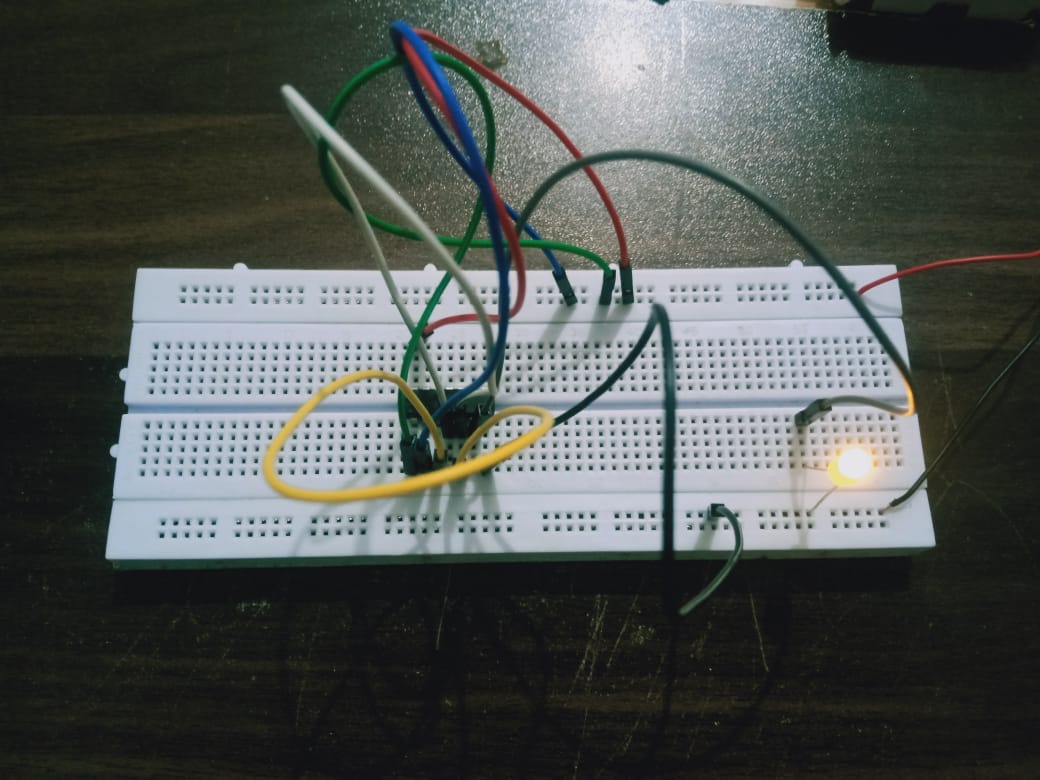
1. Give biasing to the IC and do necessary connections as shown in the circuit diagram.
2. Give various combinations of inputs and note down output using LED.
3. Repeat the procedure for all gates.

**Circuit diagram of NAND & NOR IC:**

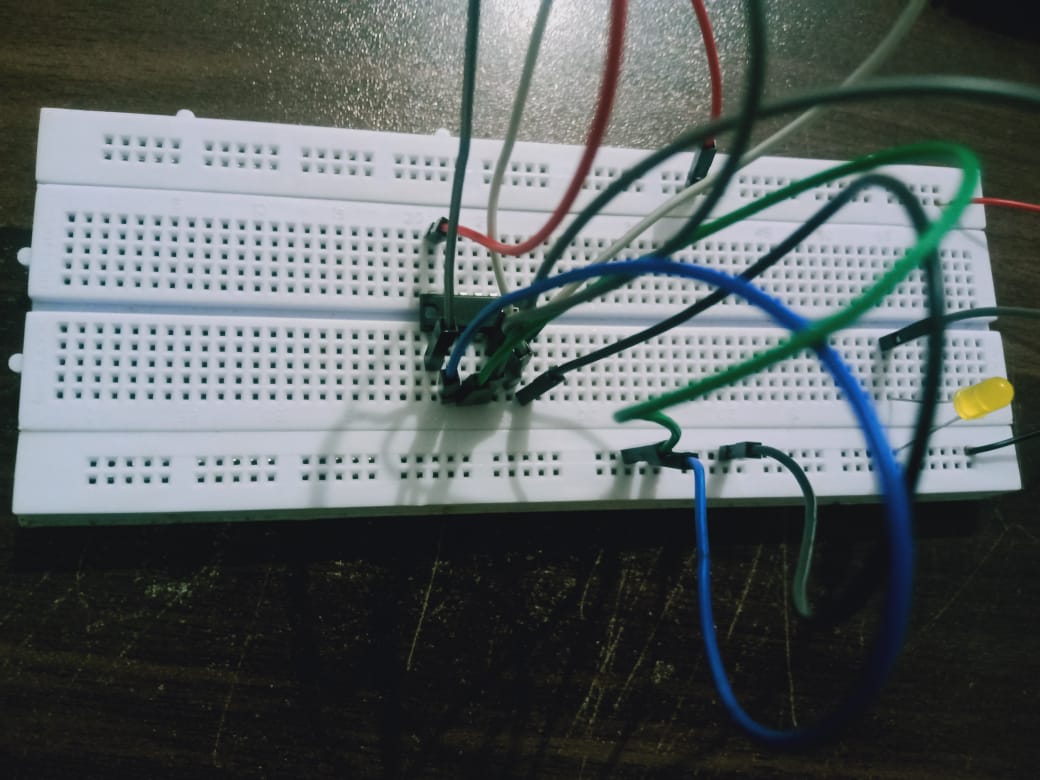


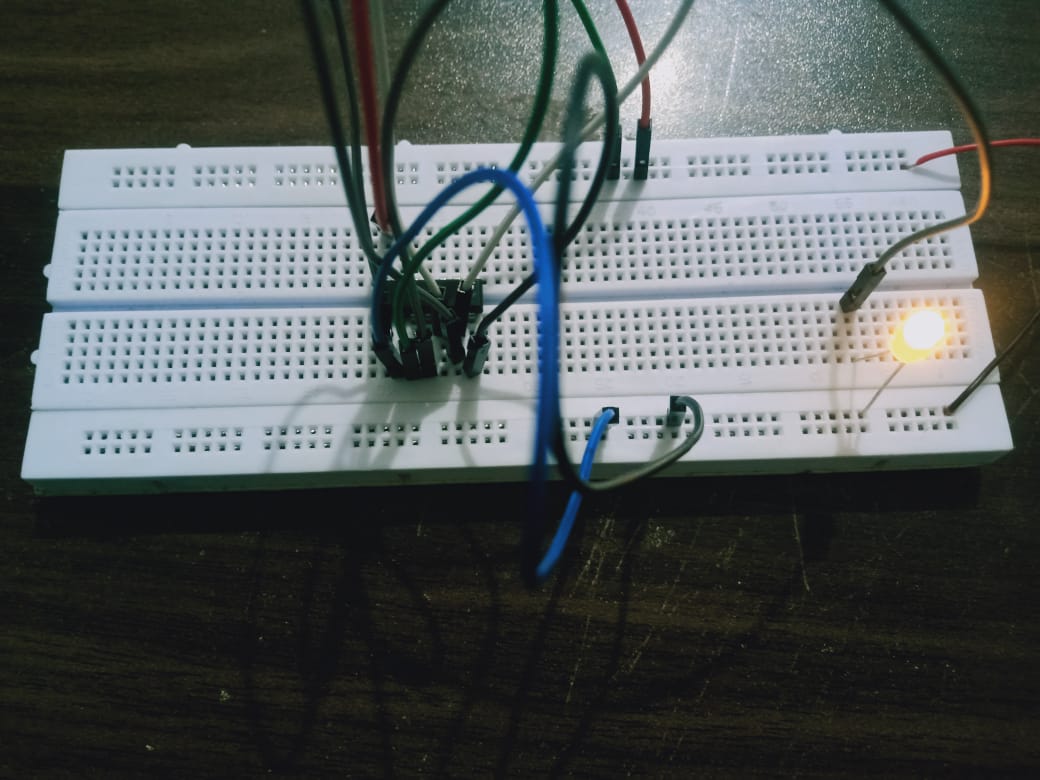


And using NAND when A=1,B= 0.

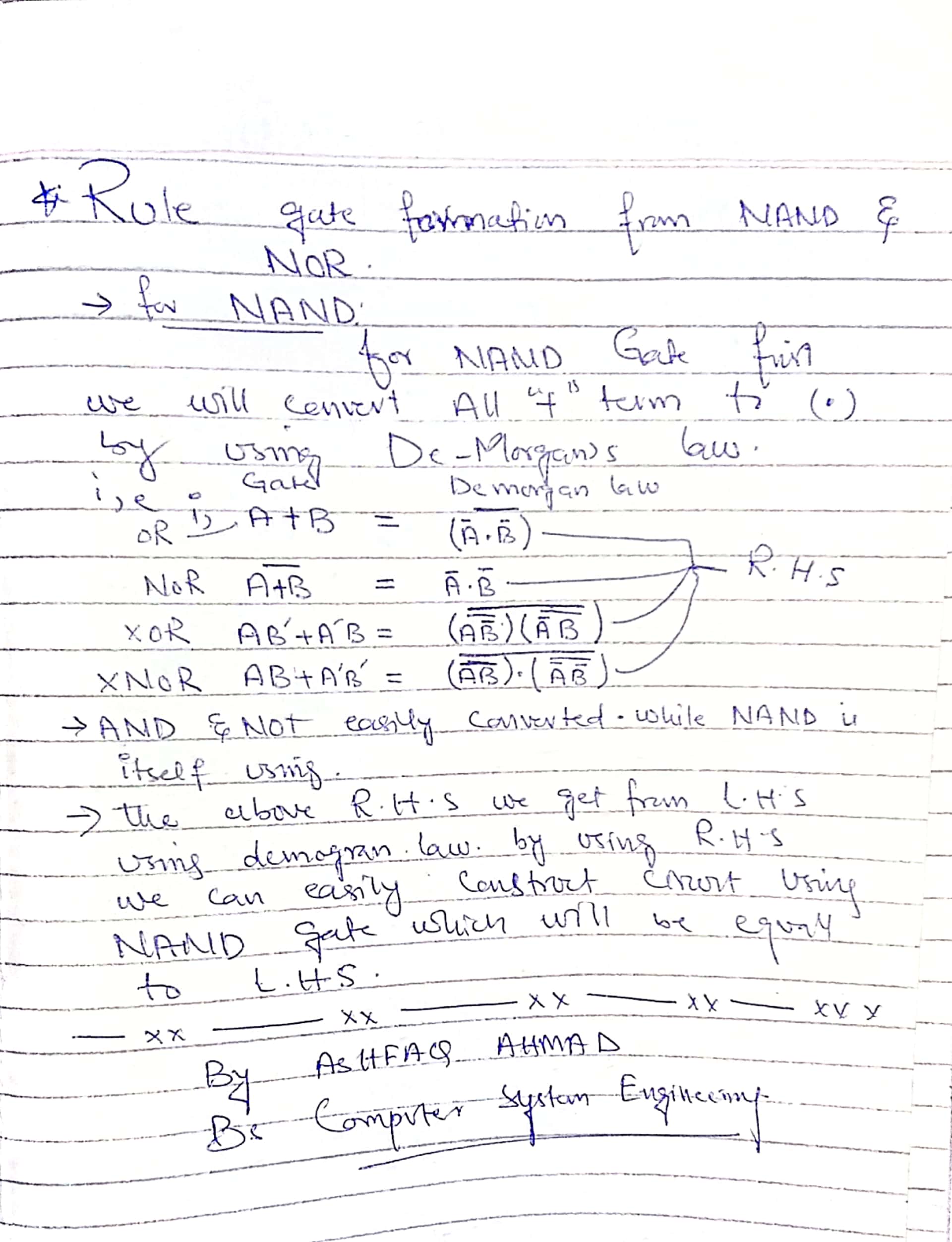


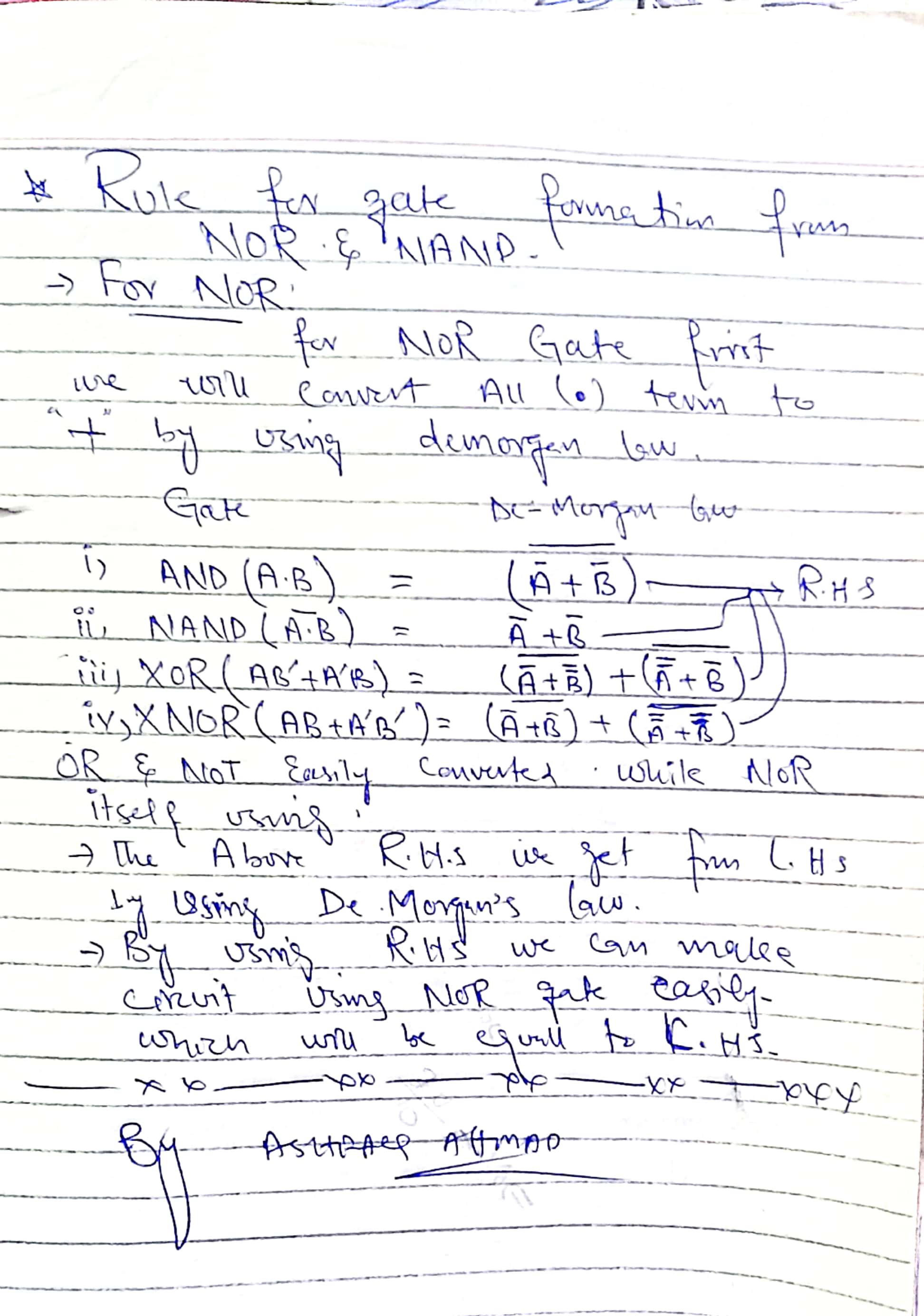
AND using NAND whan A=1,B=1.

  
OR using NOR at A=0,B=0.



OR using NOR at A=1,B=0.





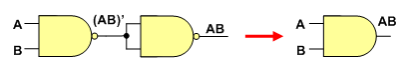
1. **NAND universal gate:**

To prove that any Boolean function can be implemented using only NAND gates, we will show that the AND, OR, NOT, EX-OR and NOR operations can be performed using only NAND.

gates.

* **Implementing AND Using only NAND Gates:**

An AND gate can be replaced by NAND gates as shown in the figure (The AND is Replaced by a NAND gate with its output complemented by a NAND gate inverter).



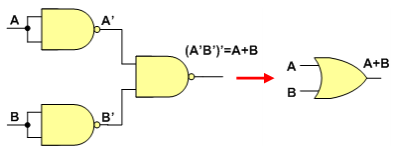
After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of A.B. its mean that above circuit act as AND gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AND gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y** |
| 0 |  | **0** | |  | **0** |
| 1 |  | **0** | |  | **0** |
| 0 |  | **1** | |  | **0** |
| 1 |  | **1** | |  | **1** |

* **Implementing OR Using only NAND Gates:**

An OR gate can be replaced by NAND gates as shown in the figure (The OR gate is Replaced by a NAND gate with all its inputs complemented by NAND gate inverters).



After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of A+B. its mean that above circuit act as OR gate.

**Truth table:**

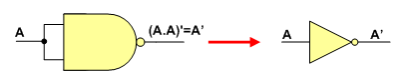
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y** |
| 0 |  | **0** | |  | **0** |
| 1 |  | **0** | |  | **1** |
| 0 |  | **1** | |  | **1** |
| 1 |  | **1** | |  | **1** |

.

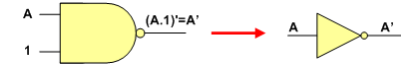
* **Implementing NOT Using only NAND Gates:**

One NAND input pin is connected to the input signal A while all other input pins

are connected to logic 1. The output will be A’.



OR



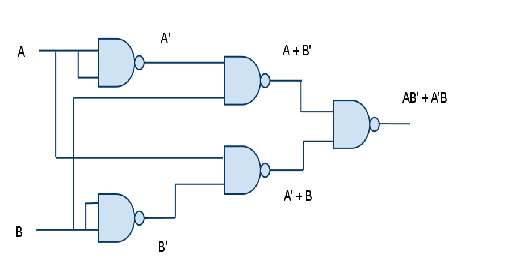
The figure shows two ways in which a NAND gate can be used as an inverter (NOT gate).

After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of A’. Its mean that above circuit act as NOT gate.

**Truth table:**

|  |  |  |  |
| --- | --- | --- | --- |
| NOT gate | | | |
| Inputs | | **Output** | |
| A |  | **Y** |  |
| 0 |  | **1** |  |
| 1 |  | **0** |  |

* **Implementing EX-OR Using only NAND Gates.**

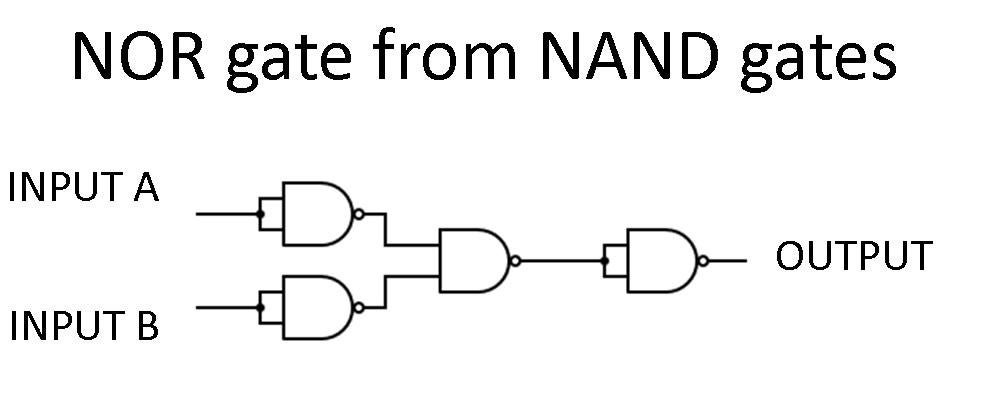


After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of AB’+A’B. its mean that above circuit act as X-OR gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XOR,EX-OR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y=(A.B’)+(A’.B) or Y=(A+B).(A’+B’)** |
| 0 |  | **0** | |  | **0** |
| 1 |  | **0** | |  | **1** |
| 0 |  | **1** | |  | **1** |
| 1 |  | **1** | |  | **0** |

* **Implementing NOR Using only NAND Gates.**

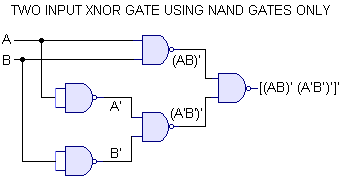


After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of (A+B)’. Its mean that above circuit act as NOR gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NOR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y** |
| 0 |  | **0** | |  | **1** |
| 1 |  | **0** | |  | **0** |
| 0 |  | **1** | |  | **0** |
| 1 |  | **1** | |  | **0** |

* **Implementing X-NOR Using only NAND Gates.**



After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of AB’+A’B. Its mean that above circuit act as X-OR gate.

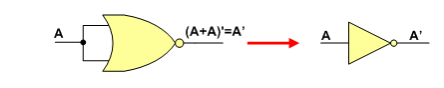
**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XNOR,EX-NOR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y=(A.B)+(A’.B’) or Y=(A+B’).(A’+B)** |
| 0 |  | **0** | |  | **1** |
| 1 |  | **0** | |  | **0** |
| 0 |  | **1** | |  | **0** |
| 1 |  | **1** | |  | **1** |

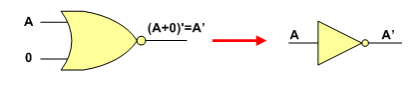
1. **NOR Gate is a Universal Gate:**

To prove that any Boolean function can be implemented using only NOR gates, we will show that the AND, OR, NOT, NAND and EX-OR operations can be performed using only NOR gates.

* **Implementing an Inverter NOT Using only NOR Gates.**



OR



The figure shows two ways in which a NOR gate can be used as an inverter (NOT gate).

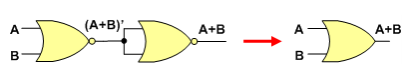
After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of (A)’. its mean that above circuit act as NOT gate.

**Truth table:**

|  |  |  |  |
| --- | --- | --- | --- |
| NOT gate | | | |
| Inputs | | **Output** | |
| A |  | **Y** |  |
| 0 |  | **1** |  |
| 1 |  | **0** |  |

* **Implementing OR Using only NOR Gates**

An OR gate can be replaced by NOR gates as shown in the figure (The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter)



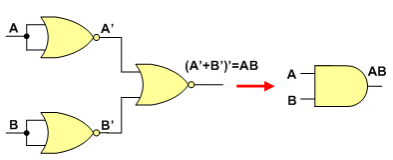
After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of (A+B). its mean that above circuit act as OR gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y** |
| 0 |  | **0** | |  | **0** |
| 1 |  | **0** | |  | **1** |
| 0 |  | **1** | |  | **1** |
| 1 |  | **1** | |  | **1** |

* **Implementing AND Using only NOR Gates**

An AND gate can be replaced by NOR gates as shown in the figure (The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters)

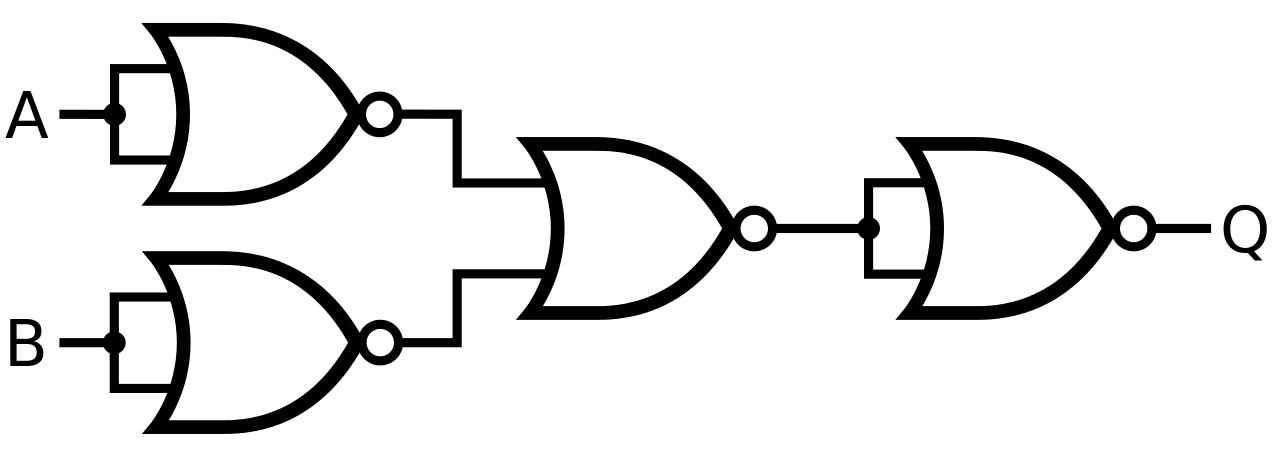


After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of (A.B). its mean that above circuit act as AND gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AND gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y** |
| 0 |  | **0** | |  | **0** |
| 1 |  | **0** | |  | **0** |
| 0 |  | **1** | |  | **0** |
| 1 |  | **1** | |  | **1** |

* **Implementing an NAND Using only NOR Gates.**

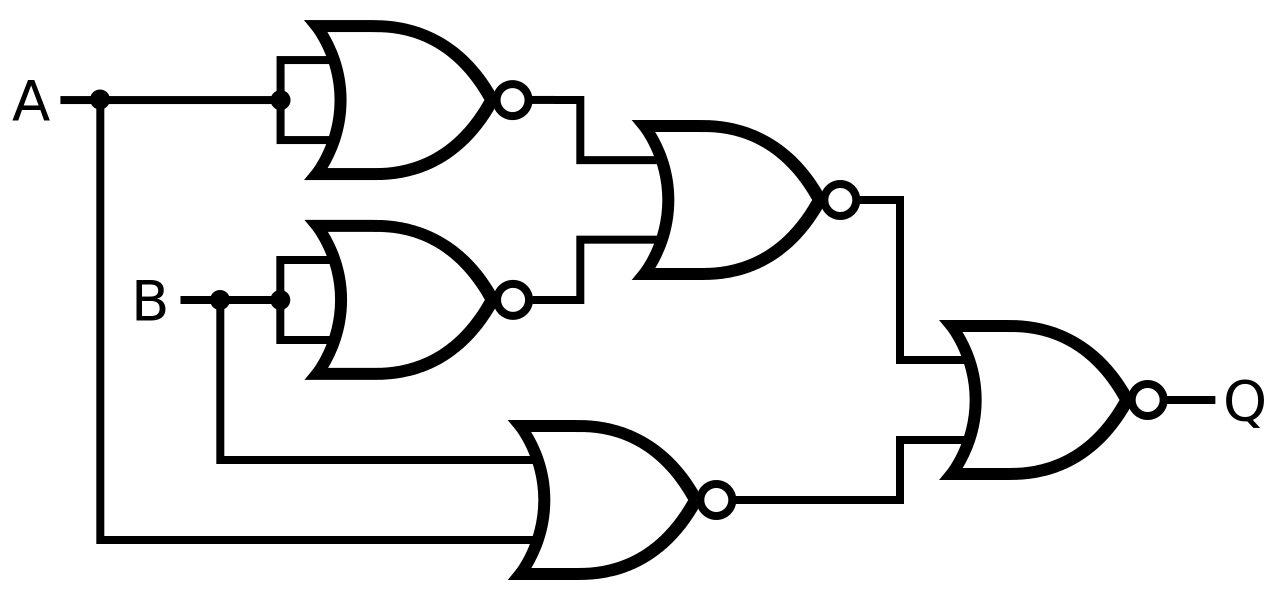


After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of (A.B)’. its mean that above circuit act as NAND gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NAND gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y=(A.B)’** |
| 0 |  | **0** | |  | **1** |
| 1 |  | **0** | |  | **1** |
| 0 |  | **1** | |  | **1** |
| 1 |  | **1** | |  | **0** |

* **Implementing an EX-OR Using only NOR Gates.**

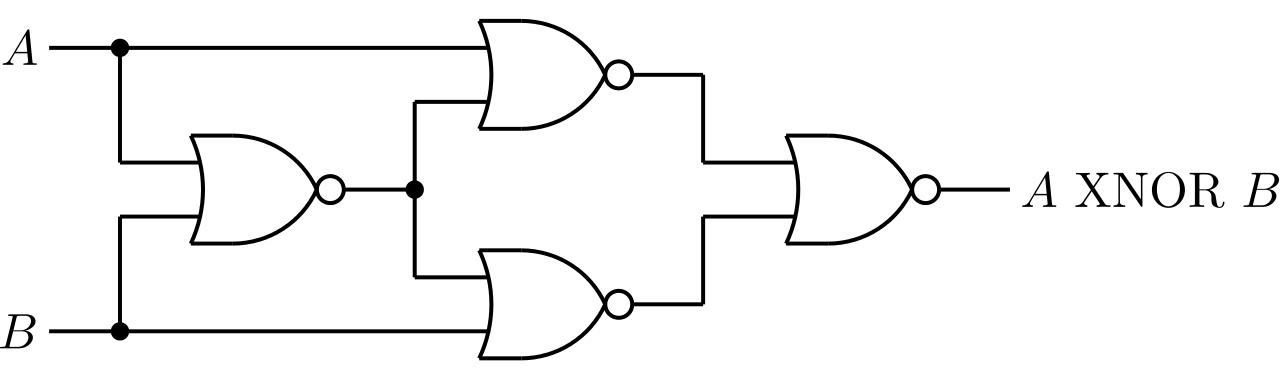


After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of AB’+A’B. its mean that above circuit act as X-OR gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XOR,EX-OR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y=(A.B’)+(A’.B) or Y=(A+B).(A’+B’)** |
| 0 |  | **0** | |  | **0** |
| 1 |  | **0** | |  | **1** |
| 0 |  | **1** | |  | **1** |
| 1 |  | **1** | |  | **0** |

* **Implementing an EX-NOR Using only NOR Gates.**



After implementing above circuit on breadboard we get the following data at different inputs. The output of above circuit is equal to Y, which is required result of AB’+A’B. its mean that above circuit act as X-OR gate.

**Truth table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XNOR,EX-NOR gate | | | | | |
| Inputs | | |  | | **output** |
| A |  | **B** | |  | **Y=(A.B)+(A’.B’) or Y=(A+B’).(A’+B)** |
| 0 |  | **0** | |  | **1** |
| 1 |  | **0** | |  | **0** |
| 0 |  | **1** | |  | **0** |
| 1 |  | **1** | |  | **1** |

**CONCLUSION**: Thus universal gates are studied.

**THE END**