## ASSIGNMENT 1:

## TITLE: Introduction to Digital Fundamental Circuit Design using Proteus.

PRN: 22510064

DATE: 9/9/2023

AIM: a) Implement basic gates -  OR, NAND, NOR, Ex-OR, Ex-NOR. Verify truth table for each gate.

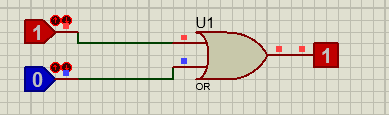
b)  Demonstrate universal gates – NAND and NOR. Verify the truth table.

HARDWARE USED: No

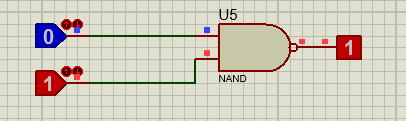
SOFTWARE USED: Proteus 8 Professional

PROCEDURE:

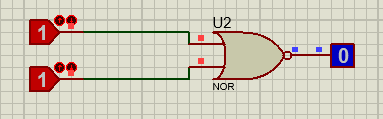
* Start Proteus and click on the P button to open the Pick Library window.
* Search for the gate you want to use, such as AND, OR, NOT, NAND, Ex-OR, Ex-NOR etc. and drag it to the workspace.
* Connect the gate inputs to Logic Toggle components and the gate output to an logic state.
* Connect all the components to a Ground Terminal.
* Run the simulation and observe the Logic state behaviour according to the gate’s truth table.



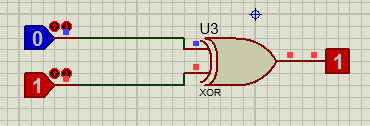
|  |  |  |
| --- | --- | --- |
| ***OR GATE:*** | **Y = A + B** |  |
| A | B | Y |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |



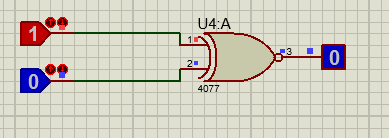
|  |  |  |
| --- | --- | --- |
| ***NAND GATE:*** | (A.B)0 or A + B |  |
| A | B | Y |
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |



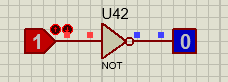
|  |  |  |
| --- | --- | --- |
| NOR GATE: | (A + B)0 or A.B |  |
| A | B | Y |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |



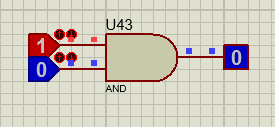
|  |  |  |
| --- | --- | --- |
| EXOR GATE: | A ⊕ B or A.B + A.B |  |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



|  |  |  |
| --- | --- | --- |
| EXNOR GATE: | [**A ⊙ B** (OR) **A.B + A.B**](https://www.build-electronic-circuits.com/xnor-gate/). |  |
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



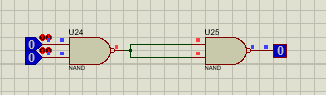
|  |  |  |
| --- | --- | --- |
| NOT GATE | A’ = Y |  |
| A | Y |  |
| 0 | 1 |  |
| 1 | 0 |  |



|  |  |  |
| --- | --- | --- |
| AND GATE | Y = A.B |  |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

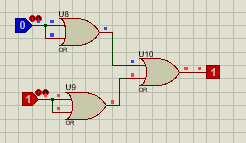
**UNIVERSAL GATES:**

**AND GATE USING NAND:**

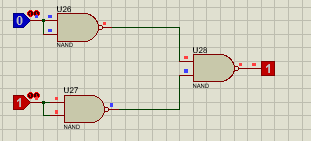
****

|  |  |  |
| --- | --- | --- |
| AND GATE | Y = A.B |  |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**AND USING NOR:**

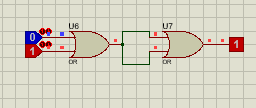


**OR USING NAND:**

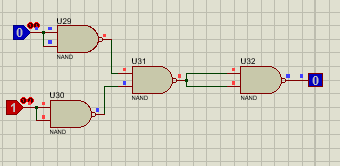
****

|  |  |  |
| --- | --- | --- |
| ***OR GATE:*** | **Y = A + B** |  |
| A | B | Y |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

**OR USING NOR:**

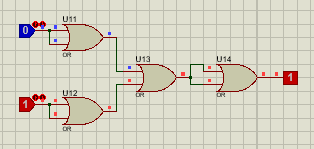


**NOR USING NAND:**

****

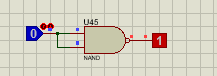
|  |  |  |
| --- | --- | --- |
| NOR GATE: | (A + B)0 or A.B |  |
| A | B | Y |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |

**NAND USING NOR:**

****

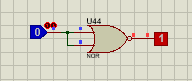
|  |  |  |
| --- | --- | --- |
| ***NAND GATE:*** | (A.B)0 or A + B |  |
| A | B | Y |
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

**NOT USING NAND:**

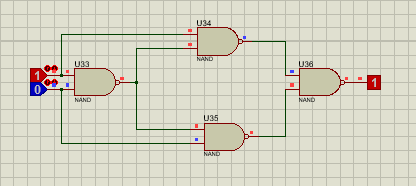
****

|  |  |  |
| --- | --- | --- |
| NOT GATE | A’ = Y |  |
| A | Y |  |
| 0 | 1 |  |
| 1 | 0 |  |

**NOT USING NOR:**

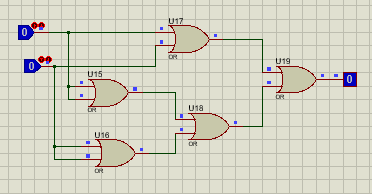
****

**EXOR USING NAND:**

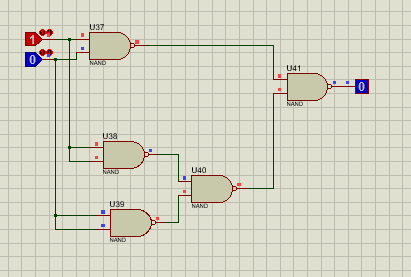


|  |  |  |
| --- | --- | --- |
| EXOR GATE: | A ⊕ B or A.B + A.B |  |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**EXOR USING NOR:**

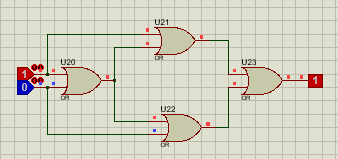
****

**EXNOR GATE USING NAND GATE:**



|  |  |  |
| --- | --- | --- |
| EXNOR GATE: | [**A ⊙ B** (OR) **A.B + A.B**](https://www.build-electronic-circuits.com/xnor-gate/). |  |
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**EXNOR USING NOR:**

****

RESULTS: The results of using gates in Proteus depend on the type of gate and the inputs that are applied to it. Here are some examples:

* An AND gate will output a HIGH signal only if both of its inputs are HIGH.
* An OR gate will output a HIGH signal if either of its inputs is HIGH.
* A NOT gate will output a LOW signal if its input is HIGH, and a HIGH signal if its input is LOW.
* A NAND gate is the inverse of an AND gate, so it will output a LOW signal if both of its inputs are HIGH.
* A NOR gate is the inverse of an OR gate, so it will output a LOW signal if either of its inputs is HIGH.
* A XOR gate will output a HIGH signal if its inputs are different, and a LOW signal if its inputs are the same.
* A XNOR gate is the inverse of an XOR gate, so it will output a LOW signal if its inputs are different, and a HIGH signal if its inputs are the same.

The results of using gates in Proteus can be simulated by connecting the gates to each other and to input and output signals. The simulation will show the output of each gate for different combinations of input signals. This can be used to verify the operation of the gates and to design more complex circuits.

CONCLUSION: Logic gates are the basic building blocks of digital circuits, and they can be used to create a wide variety of electronic devices.