|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Digital Design Laboratory** | **Semester:** | **III** |
| **Date of Performance:** | **\_\_\_ / \_\_\_ / \_\_\_\_\_\_** | **Batch No:** | **A4** |
| **Faculty Name:** | **Kiran Ajetrao** | **Roll No:** | **16010122083** |
| **Faculty Sign & Date:** |  | **Grade/Marks:** | **\_\_\_/25** |

**Experiment No: 1**

**Title: Study of Basic Gates and Universal Gates**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Understand Basic Logic Gates and Universal Gates |

|  |
| --- |
| **COs to be achieved:** |
| **CO1**: Recall basic gates & logic families and binary, octal & hexadecimal calculations and conversions. |

|  |
| --- |
| **Tools used:** |
| Trainer kits |

|  |
| --- |
| **Theory:** |
| Logic gates are electronic circuits that perform logical operations on one or more input signals to produce an output signal based on a set of logical rules. Logic gates can be classified into the following categories:   1. Basic Gates:    1. AND Gate: The AND gate produces a high output (1) only when all of its inputs are high (1).    2. OR Gate: The OR gate produces a high output (1) if any of its inputs is high (1).    3. NOT Gate (Inverter): The NOT gate produces the logical complement of its input. It takes a single input and produces the opposite value as the output. 2. Derived Gates:    1. NAND Gate: The NAND gate is a combination of an AND gate followed by a NOT gate. It produces the inverse of the AND gate's output. It outputs a low (0) only when all of its inputs are high (1).    2. NOR Gate: The NOR gate is a combination of an OR gate followed by a NOT gate. It produces the inverse of the OR gate's output. It outputs a high (1) only when all of its inputs are low (0).    3. XOR Gate (Exclusive OR): The XOR gate produces a high output (1) when the number of high inputs is odd. It outputs a low (0) when the number of high inputs is even.    4. XNOR Gate (Exclusive NOR): The XNOR gate produces a high output (1) when the number of high inputs is even. It outputs a low (0) when the number of high inputs is odd. 3. Universal Gates:   NAND and NOR gates are considered universal gates because any logic function can be implemented using only NAND gates or only NOR gates. This means that with a sufficient number of NAND or NOR gates, you can create circuits that can perform any logical operation. |

|  |
| --- |
| **Implementation Details** |
| 1. AND Gate: Y =   Symbol    Pin Diagram    Truth Table:  Image result for truth table and   1. OR Gate: Y =   Symbol    Pin Diagram    Truth Table:  Image result for truth table or   1. NOT Gate: Y =   Symbol    Pin Diagram    Truth Table:  Introduction to NOT Gate - projectiot123 Technology Information Website  worldwide   1. NAND Gate: Y =   Symbol    Pin Diagram    Truth Table:  Image result for truth table nand   1. NOR Gate: Y =   Symbol    Pin Diagram  7402 Technical Data  Truth Table:  Introduction to NOT Gate - projectiot123 Technology Information Website  worldwide   1. XOR Gate: Y =   Symbol    Pin Diagram    Truth Table:  Image result for truth table nor   1. XNOR Gate: Y =   Symbol    Pin Diagram  XNOR Gate Circuit Diagram & Working Explanation  Truth Table:    **Implementation Using NAND Gate**  **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.     |  |  | | --- | --- | | **A** | **Q** | | 0 | 1 | | 1 | 0 |     **AND GATE**  An AND gate is made by inverting the output of a NAND gate as shown below     |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |   **OR GATE**  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.     |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 |   **Implementation Using NOR Gate**  **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.     |  |  | | --- | --- | | **A** | **Q** | | 0 | 1 | | 1 | 0 |   **AND GATE**  An AND gate gives a 1 output when both inputs are 1. Therefore, an AND gate is made by inverting the inputs of a NOR gate. Again, note that a NOT gate is equivalent to a NOR with its inputs joined.     |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |   **OR GATE**  An OR gate is made by inverting the output of a NOR gate. Note that we already know that a NOT gate is equivalent to a NOR gate with its inputs joined.     |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| 1. Implement the Boolean function using NAND gates and NOR gates F=A’B + AB’ 2. Implement using combination of gates F = ABC + AB’C + ABC’ |

|  |
| --- |
| **Conclusion:** |
| Through this experiment we learn various types of various types of logic gates |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |