**CHAPTER - 2**

**NON-REVERSIBLE GATES**

# 2. NON-REVERSIBLE GATES

**2.1 INTRODUCTION**

Classical computation theory began for the most part when Church and Turing independently published their inquiries into the nature of computability in 1936. For our purposes, it will suffice to take as our model for classical discrete computation, a block diagram of the form,

**a1**

**a2**

**an**

**f(a1, ... , an)**

**b**

**Figure 2.1 Single valued functional block**

where b = f (a1, a2, ... , an) describes a single-valued function on n discrete inputs. We will assume that such a function can be simulated or computed physically. As it is usually done in classical computation, we can use base-2 arithmetic to describe the inputs and outputs, in which case a1,..,an, and b become binary variables, or bits, taking on one of two values, 0 or 1. In this case, the function f (a1, a2, ... , an) is known as an n-bit Boolean function.

The central problem that we will concern ourselves with repeatedly in these notes is the problem of universality. That is, given an arbitrarily large function f, it is possible to identify a universal set of simple functions – called gates – that can be used repeatedly in sequence to simulate f on its inputs. The gate functions would be restricted to operating on a small number of inputs, say two or three at a time, taken from a1, a2, ... , an. These gates would be done in sequence, creating a composite function that represents f on all of its n inputs

**2.2 BASIC LOGIC GATES**

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables

**2.2.1 AND GATE**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **AB** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**AB**

**A**

**B**

**AND**

**AND**

**Figure 2.2 AND gate and its truth table**

The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high.  A dot (.) is used to show the AND operation i.e. A.B.  Bear in mind that this dot is sometimes omitted i.e. AB

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

**2.2.2 OR GATE**

**A+B**

**A**

**B**

**OR**

**Figure 2.3 OR gate and its truth table**

The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high.  A plus (+) is used to show the OR operation.

**2.2.3 NOT GATE**

|  |  |
| --- | --- |
| **A** |  |
| 0 | 1 |
| 1 | 0 |

**NOT**

**A**

**Figure 2.4 NOT gate and its truth table**

The NOT gate is an electronic circuit that produces an inverted version of the input at its output.  It is also known as an inverter.  If the input variable is A, the inverted output is known as NOT A.  This is also shown as A', or A with a bar over the top, as shown at the outputs.

|  |  |  |
| --- | --- | --- |
| **A** | **B** |  |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**2.2.4 NAND GATE**

**NAND**

**A**

**B**

**AND**

**Figure 2.5 NAND gate and its truth table**

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

**2.2.5 NOR GATE**

|  |  |  |
| --- | --- | --- |
| **A** | **B** |  |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**A**

**B**

**NOR**

**Figure 2.6 NOR gate and its truth table**

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate.  The outputs of all NOR gates are low if **any** of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A⊕B** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**2.2.6 XOR GATE**

**XOR**

**A⊕B**

**A**

**B**

**Figure 2.7 XOR gate and its truth table**

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high.  An encircled plus sign (http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/graphics/enplus.gif) is used to show the EOR operation.

**2.2.7 XNOR GATE**

|  |  |  |
| --- | --- | --- |
| **A** | **B** |  |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**A**

**B**

**XNOR**

**Figure 2.8 XNOR gate and its truth table**

The '**Exclusive-NOR'**gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

 The NAND and NOR gates are called universal functions since with either one the AND and OR functions and NOT can be generated.

**Note:**

A function in sum of products form can be implemented using NAND gates by replacing all AND and OR gates by NAND gates.

A function in product of sums form can be implemented using NOR gates by replacing all AND and OR gates by NOR gates.