CIRCUIT DESIGN

Agenda

- NAND AND NOR GATES
- Approaches to Circuit Design,
- Basic combination of logic gates(circuit design)
- Determining the output of a circuit
- Its Boolean algebra.
- Writing truth tables
- Circuit simplification methods: Algebraic method, Karnaugh Maps.

How to Design Logic Circuits & Logic Gates

- Logic gates are the building block of digital circuits.
- Do you want to design and build logic circuits that are able to control spaceships, driver-less cars or IoT (Internet of Things) systems? The basic concepts to learn in this lesson are the same way that HW designers design complex systems.In this lesson, we will learn how, given a specification, to design the corresponding logic circuit using basic logic gates.

Introduction

- Last wk, we covered basic Boolean logic aspects. We will use the knowledge you acquired about Boolean logic, Boolean expressions and Boolean operators in **designing logic circuits**.
- Let us start by differentiating between two activities related to digital systems: **design** and **analysis**. System analysis refers to understanding what a system or a circuit does. In other words, analyzing a digital circuit means that we want to move from a circuit to a truth table or a 'human' readable specification. This is opposed to designing a digital system or circuit, which refers to the transition of a humanly understandable specification, based on non-technical language, into a functioning circuit, or at least a diagram of a circuit.

Designing Logic Circuits

- In this lesson, we are interested in the design process. The following is a **systematic procedure** to design a logic circuit:
- Deduct the truth table from the human-readable specification
- Transfer the truth table into a Karnaugh map in order to simplify the function ((or any other simplification method)if possible)
- Deduct the circuit and draw the gate diagram (and the wired-circuit if required)

Example 1

- Let us start with a simple example of designing a simple voting system. Suppose you want to design a voting system for a company. This company has a board of directors composed of 3 directors having different voting weights as follows:
- Director A: 20%
- Director B: 30%
- Director C: 50%
- A decision passes if it gets more than 50% of votes. We want to design the circuit that implements this voting system.
- Supposing the output of the circuit is named Z, in order to design this system, we will start by utilizing the truth table:

Introduction
Simplifying logic circuits

- Is a predominant task when designing a digital system. When you are able to design simpler circuits, you will be able to place more functionality on integrated circuits, such as microprocessors. Also, the simpler your circuits are, the less power they consume.
- In this lesson, we will **build** a Boolean circuit, first using its Boolean expression, and then we will simplify the expression and build the corresponding circuit.

How We Simplify and Build Logic Circuits?

- Often, when you deduce a Boolean expression from the system's specifications, you will get a complex expression that can be simplified. When you simplify a Boolean expression, you aim to get a simpler or less complex circuit in terms of the hardware. Designers always strive to build less complex circuits as this leads to many desirable features, such as **less power consumption** and **less expensive circuits**.

Methods of Ccircuit Simplification

- There are many methods to simplify a logic expression. Some of these methods are using

1. **Boolean Algebra laws,**
2. **Karnaugh maps**
3. **Quine-McCluskey algorithm**.

- Boolean Algebra laws is usually used for simple expressions.
- Karnaugh maps for expressions with less than 6 variables
- Quine-McCluskey algorithm for complex expression with more than 6 variables.
- Let us work on an example. Given the following Boolean expression:

F= ab'c'd'+a'b'cd'+ab'cd'+a'b'c'd+ab'c'd+a'bc'd+ abc'd+a'b'cd+ab'cd+abcd