NARAYANA SCHOOL STUDENT PROUDLY PRESENTS

SAMSUNG CHIP DESIGN FOR HIGH SCHOOL

WEEK 4

Vikranth majeti
1-20-2025

Digital Logic Design and Boolean Algebra

1. Introduction to Basic Logic Operations

Logic operations form the foundation of digital electronics and computing systems. The primary logic gates—AND, OR, NOT, XOR, XNOR—enable the construction of complex circuits that process binary data. Let us explore each in detail:

1.1 AND Gate

- **Definition: Outputs 1 only if both inputs are 1.**
- **Truth Table:**

  A B Output (A AND B)

  0 0 0

  0 1 0

  1 0 0

  1 1 1

1.2 OR Gate

- **Definition: Outputs 1 if at least one input is 1.**
- **Truth Table:**

  A B Output (A OR B)

  0 0 0

  0 1 1

  1 0 1

A B Output (A OR B)

1 1 1

1.3 XOR Gate

- **Definition: Outputs 1 if inputs are different.**
- **Truth Table:**

A B Output (A XOR B)

0 0 0

0 1 1

1 0 1

1 1 0

1.4 XNOR Gate

- **Definition: Outputs 1 if inputs are the same.**
- **Truth Table:**

A B Output (A XNOR B)

0 0 1

0 1 0

1 0 0

1 1 1

1.5 NOT Gate

- **Definition: Inverts the input.**
- **Truth Table:**

A Output (NOT A)

0 1

A Output (NOT A)

1 0

---

2. Introduction to Basic Switches

Switches are essential components in digital circuits. They can represent binary states (ON = 1, OFF = 0) and are often implemented using transistors.

- **Single-Pole Single-Throw (SPST): A simple switch that toggles between ON and OFF states.**
- **Single-Pole Double-Throw (SPDT): Connects a single input to one of two outputs.**
- **Double-Pole Double-Throw (DPDT): Can control two independent circuits.**

---

3. Advanced Logic Operations

Advanced operations build on basic gates to realise functions like multiplexing, demultiplexing, and flip-flops. These operations are crucial in building memory and control units.

---

4. Realizing Boolean Expressions Using Switches

Boolean expressions can be implemented using switches and basic logic gates. For example, the expression Y = A AND B can be represented with two switches in series.

Example: Implementing Y = A OR B with parallel switches.

---

5. Designing Combinational Logic to Add 1-Bit Numbers

The design of a 1-bit adder requires two inputs (A, B) and produces a Sum (S) and Carry (C).

- **Sum: S = A XOR B**
- **Carry: C = A AND B**

Truth Table:

A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

---

6. Designing Combinational Logic to Detect Prime Numbers

Prime numbers are integers greater than 1 with no divisors other than 1 and themselves. For simplicity, we design logic for detecting prime numbers within a small range.

---

7. Steps to Extract Boolean Functions for Prime Numbers (2 and 3)

- **Analyze the binary representations of numbers.**
- **Define conditions for detecting prime numbers.**
- **Create Boolean expressions based on truth tables.**

---

8. Steps to Extract Remaining Boolean Functions and Output Logic

For numbers beyond 3, extend the analysis by considering additional variables and simplifying Boolean expressions.

---

9. Steps to Simplify the Logic

Use Boolean algebra and truth table reduction techniques to simplify expressions, reducing the number of gates and switches required.

---

10. Need for Algorithmic Approach to Simplify Logic – K Map

Karnaugh Maps (KMaps) provide a visual method to minimize Boolean expressions, essential for efficient circuit design.

Foundation Concepts of KMap:

- **Cells represent minterms.**
- **Adjacent cells differ by one variable.**
- **Groups simplify terms.**

KMap for Any Random Boolean Function:

1. **Plot the function's minterms.**
2. **Identify adjacent groups.**
3. **Write simplified expressions.**

Applications of KMaps:

- **Reducing hardware complexity.**
- **Optimizing digital circuits.**