Boolean Algebra

CS 47

Professor Seshadri Paravastu

Naya Singhania

1. Definition
A mathematical system dealing with binary variables and logic operations

2. Origin
Developed by George Boole in the mid-19th century

3. Importance
Fundamental to digital logic design and circuit analysis

4. Boolean Variables
A boolean variable assumes two values
- FALSE (0)
- TRUE (1)

4.1. Basic Operations
- AND (.) or conjunction ($\wedge$)
- OR (+) or disjunction ($\vee$)
- NOT (‘) or negation ($\neg$)

4.2. Logic Gates

4.2.1. Definition
Electronic devices that perform logical operations on one or more binary inputs to produce a single binary output

4.2.2. Types

4.2.2.1. AND
Ensures all inputs must be true for an output to be true

4.2.2.2. OR
Ensures any input can be true for an output to be true

4.2.2.3. NOT
Ensures any input is inverted for an output to be the inversion of an input

4.2.2.4. NAND
Ensures only one input can be true for an output to be true

4.2.2.5. NOR
Ensures no inputs can be true for an output to be true

4.2.2.6. XOR
Ensures one input must be true for an output to be true

4.2.2.7. XNOR
Ensures both inputs must be identical for an output to be true

4.2.3. Applications in Computing

4.2.3.1. CPUs
- Fundamental building blocks of CPU architecture
- Used extensively in various parts of the CPU

4.2.3.2. ALU
- Consists of a network of logic gates that manipulate binary data to perform tasks like addition, subtraction, AND, OR, and NOT operations

4.2.3.3. Registers
- Each register in the register file is implemented using a group of flip-flops and logic gates to store and manipulate data
- **Flip Flops** A group of gates arranged such that they have memory of previous inputs

4.2.3.4. Control Unit
- It uses logic gates to decode instruction opcodes, determine the

sequence of operations, and control the flow of data within the CPU

4.2.4. Importance of Logic Gate Design
- **Speed** Logic gates enable fast computation and processing, contributing to the overall performance of the laptop
- **Efficiency** Compact and optimized logic gate designs help conserve space and power, essential for portable devices like laptops
- **Versatility** Logic gates allow for the implementation of a wide range of functions and operations, making laptops capable of handling diverse tasks efficiently.
- **Low Power Consumption** Modern logic gate technologies, such as CMOS (Complementary Metal-Oxide-Semiconductor), offer low power consumption characteristics, extending the battery life of laptops and improving energy efficiency

4.2.5. Universal Gate

4.2.5.1. Definition
A gate that can be used to implement any other gate

4.2.6. Primary Gates
- NAND (AND followed by NOT)
- NOR (OR followed by NOT)

5. Functions and Terms

5.1. Example(s)

$$F(X, Y, Z) = XY' + Z'$$

- This is a boolean expression - function list of variables
- Each part of the right hand side of the equation is called a ‘term’

6. Identities

1. $X + 0 = X$
2. $X * 1 = X$
3. $X + 1 = X$
4. $X * 0 = 0$
5. $X + X = X$
6. $X * X = X$
7. $X + \overline{X} = 1$
8. $X * \overline{X} = 0$
9. $\overline{X} = X$

7. Laws

1. $X + Y = Y + X$
2. $XY = YX$
3. $X + (Y + Z) = (X + Y) + Z$
4. $X(YZ) = (XY)Z$
5. $X(Y + Z = XY + XZ)$
6. $X + YZ = (X + Y)(X + Z)$
7. $\overline{X + Y} = \overline{X} * \overline{Y}$
8. $\overline{X * Y} = \overline{X} + \overline{Y}$

8. Consensus Theorem

8.1. Example(s)

$$XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

9. Complement of a function

9.1. Definition

Apply DeMorgan's theorem as many times as needed to obtain the complement

9.2. Example(s)

$$F_1 = \overline{X}Y\overline{Z} + \overline{XY}Z$$

$$\overline{\overline{X}Y\overline{Z} + \overline{XY}Z} = \left(\overline{\overline{X}Y\overline{Z}}\right) * \left(\overline{XY}Z\right) = \left(X + \overline{Y} + Z\right)\left(X + Y + \overline{Z}\right)$$

10. Sum of Products (SOP)

Minterm A product term in which all variables appear exactly one (either complemented or uncomplemented)

10.1. SOP of Minterms

X	Y	Z	Product Term	Symbol
0	0	0	X'Y'Z'	m_0
0	0	1	X'Y'Z	m_1
0	1	0	X'YZ'	m_2
0	1	1	X'YZ	m_3
1	0	0	XY'Z'	m_4
1	0	1	XY'Z	m_5
1	1	0	XYZ'	m_6
1	1	1	XYZ	m_7

11. Product of Sums (POS)

Maxterm A sum term in which all variables appear exactly one (either complemented or uncomplemented)

11.1. POS of Maxterms

X	Y	Z	Sum Term	Symbol
0	0	0	X+Y+Z	M_0
0	0	1	X+Y+Z'	M_1
0	1	0	X+Y'+Z	M_2
0	1	1	X+Y'+Z'	M_3
1	0	0	X'+Y+Z	M_4
1	0	1	X'+Y+Z'	M_5
1	1	0	X'+Y'+Z	M_6
1	1	1	X'+Y'+Z'	M_7

12. Karnaugh Maps (K-maps)

- A visual tool for simplifying boolean expressions
- Organizes minterms in a grid
- Adjacent cells represent minterms that differ by only one variable

12.1. Benefits

- Simplifies complex boolean expressions quickly

12.2. Usage

1. Create the maps

- Determine the number of variables (n)
- Create a 2^n celled grid
- Label the rows and columns with binary values

2. Plot the minterms

- Place a 1 in the cell corresponding to each minterm

3. Group the 1s

- Group adjacent 1s in powers of 2 (1, 2, 4, 8, etc.)
- Groups should be as large as possible
- Groups can wrap around the edges of the K-map

4. Write the simplified expression

- Write a product term for each group (terms that remain constant in the group)
- The product term includes variables that remain constant within the group
- Combine the product terms using the OR operator

12.3. Simplification Rules

1. We can either group 0s with 0s or 1s with 1s but we cannot groups 0s and 1s, and x's representing don't cares can be grouped with 0s as well as 1s
2. Groups may overlap each other
3. A group must contains a number of cells that is a power of 2, so only groups with 2^n cells are allowed
4. Groups can only be horizontal or vertical, not diagonal or any other shape
5. Each group should be as large as possible
6. Opposite grouping and corner groupings are allowed
7. There should be as few groups as possible

12.4. Example(s)

1.

$F(A, B, C, D) = \sum m(0, 1, 2, 5, 7, 8, 9, 10, 13, 15)$

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	1		1
$\overline{A}B$		1	1	
AB		1	1	
$A\overline{B}$	1	1		1

$F(A, B, C, D) = BD + C'D + B'D'$

12.5. Implicants

- A product term is an implicant of the function if the unction has value 1 or all minterms of the product term (also defined as a single 1 or a group of adjacent 1s that can be combined)