CSE 100: Computer Skills

Lecture 7: Digital Systems
-Boolean Algebra

Shahadat Hussain Parvez

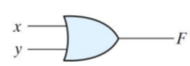
Boolean Algebra

- A Boolean variable is that has only 2 possible values
  - TRUE or FALSE
  - HIGH or LOW
- Boolean variables are logical variables
- Algebra with these Boolean variables is called Boolean Algebra
- Operator that operate on Boolean variables are called Boolean operator or logical operator

Logic operation and gates

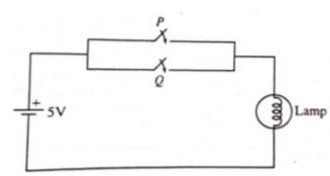
- Logical operations are implemented by various electronic circuits
- These circuits are called logic gates
- Fundamental Gates include
  - OR gate
  - AND gate
  - NOT gate
- Other gates include
  - NOR gate
  - NAND gate
  - XOR gate
  - XNOR gate

OR operation

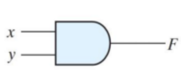


$F = x + y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

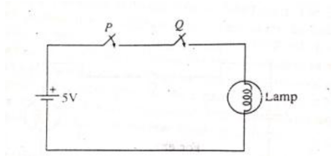


AND operation

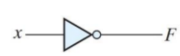


$F = x \cdot y$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1



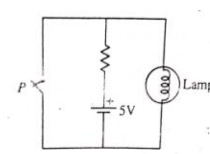
NOT operation



$F = x'$

x	F
0	1
1	0

Note that the NOT or complement can be written both as X' and $\overline{X}$

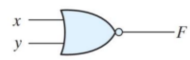


Functional Gates

- Some simple Boolean functions are often expressed as single operation
- Different gate is used to implement that
- Common functional gates are
  - NOR gate
  - NAND gate
  - XOR gate
  - XNOR gate

NOR Gate

- Basically a OR gate followed by a NOT gate

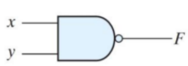


$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

NAND Gate

- Basically an AND gate followed by a NOT Gate

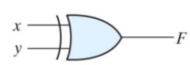


$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

XOR Gate

- Also known as exclusive OR

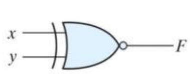


$$F = xy' + x'y = x \oplus y$$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

XNOR Gate

- Basically an XOR gate followed by a NOT gate



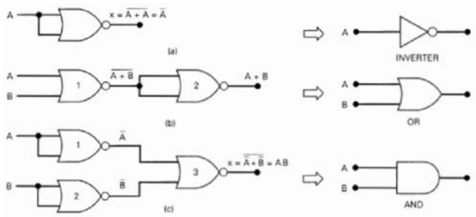
$$F = xy + x'y' = (x \oplus y)'$$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

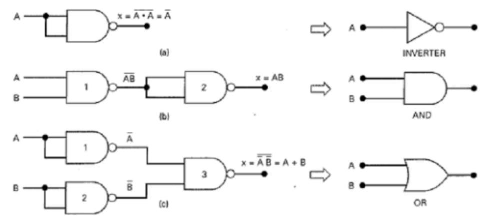
Universal Gates

- NOR and NAND gates called universal gate
- They can simulate any fundamental gates
- Therefore can implement any logic functions

Fundamental gates using NOR



Fundamental gates using NAND



Boolean Functions and Truth Table

- The OR and AND operators can be applied on more than two operands
- Combining the operators and operands Boolean expression are formed
- The Order of precedence is NOT, AND, and then OR

- A truth table is simply the tabular representation of Boolean function

Truth table from Boolean Functions

- Steps are as follows
  1. Count in binary with each number having as many bits as the number of variables. Put each number in a row in the table
  2. Now for each row calculate the value of the function

Finding Function from truth table

- Sum of minterms are used to make functions

			Minterms	
x	y	z	Term	Designation
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

Finding Function from truth table

X	Y	W	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

X	Y	W	Z	Minterms	
0	0	0	0	$\overline{X}.\overline{Y}.\overline{W}$	
0	0	1	1	$\overline{X}.\overline{Y}.W$	√
0	1	0	0	$\overline{X}.Y.\overline{W}$	
0	1	1	0	$\overline{X}.Y.W$	
1	0	0	1	$X.\overline{Y}.\overline{W}$	√
1	0	1	1	$X.\overline{Y}.W$	√
1	1	0	1	$X.Y.\overline{W}$	√
1	1	1	1	$X.Y.W$	√

The function is Z= X’Y’W + XY’W’ + XY’W + XYW’ + XYW

Boolean Postulates and Laws

Boolean algebra follows some postulates. For any boolean variable X, the postultes are the following.

(1) $X+0=X,\ X.1=X$ (2) $X+X=X,\ X.X=X$
(3) $X+1=1,\ X.0=0$ (4) $X+\overline{X}=1,\ X.\overline{X}=0$
(5) $\overline{\overline{X}}=X$

Apart from these postulates, boolean algebra also follows the commutative, associative and distributive laws like ordinary algebra. For any boolean variables W, X, Y and Z these law mean the following.

Commutative Law $X+Y=Y+X$
$X.Y=Y.X$
Associative Law $X+(Y+Z)=(X+Y)+Z=X+Y+Z$
$X.(Y.Z)=(X.Y).Z=X.Y.Z$
Distributive Law $X(Y+Z)=X.Y+X.Z$
$(W+X).(Y+Z)=W.Y+W.Z+X.Y+X.Z$

These postulates and laws are used to simplify boolean functions.

De Morgan's Theorem

Agustus De Morgan has given two useful theorems for boolean algebra. They are the following.

- **First Theorem:** For any two boolean variables X and Y, $\overline{X+Y}=\overline{X}.\overline{Y}$.
- **Second Theorem:** For any two boolean variables X and Y, $\overline{X.Y}=\overline{X}+\overline{Y}$.

These theorems are also true for any number of variables. Using truth table, they can be proved easily.

Simplification of Functions

- Simplification of Boolean Logic allows for implementation with fewer number of gates
- Similar to normal Algebra
- Steps of simplification are as follows
  - Terms with Parenthesis are evaluated first
  - The Order of precedence is NOT, AND, and then OR
  - In every step Boolean Postulates, laws and theorem are used where necessary

Simplification of Functions

Example 2.44: Simplify $F=\overline{X}(X+Y)$.
Solution:

$$\begin{aligned} F &= \overline{X}(X+Y) \\ &= \overline{X}X+\overline{X}Y && \text{[Using distributive law.]} \\ &= 0+\overline{X}Y && \text{[Using postulate } \overline{X}X=0.\text{]} \\ &= \overline{X}Y && \text{[Using postulate } X+0=0.\text{]} \end{aligned}$$

Example 2.45: Simplify $(\overline{A+B+\overline{C}})\overline{B}$.
Solution:

$$\begin{aligned} (\overline{A+B+\overline{C}})\overline{B} &= (\overline{A}.\overline{B}.\overline{\overline{C}})\overline{B} && \text{[Using De Morgan's law.]} \\ &= (\overline{A}.\overline{B}.C)\overline{B} && \text{[Using the postulate } \overline{\overline{X}}=X.\text{]} \\ &= \overline{A}.\overline{B}.\overline{B}.C && \text{[Using commutative law.]} \\ &= \overline{A}.\overline{B}.C && \text{[Using the postulate } X.X=X.\text{]} \end{aligned}$$

Simplification of Functions

Example 2.46: Simplify $F=A\overline{B}+\overline{A}.\overline{B}+AB$.
Solution:

$$\begin{aligned} F &= A\overline{B}+\overline{A}.\overline{B}+AB \\ &= (A+\overline{A})\overline{B}+AB \\ &= (1)\overline{B}+AB && \text{[Using postulate } \overline{X}+X=1.\text{]} \\ &= \overline{B}+AB \end{aligned}$$

LOGIC CIRCUIT

Logic Circuit from Logic Functions

Example 2.47: Implement the XOR function using basic gates.
Solution: From the truth table of XOR it is obvious that the output $X = \overline{P}Q + P\overline{Q}$.
Hence implementation of $X = \overline{P}Q + P\overline{Q}$ is shown in Figure 2.18.

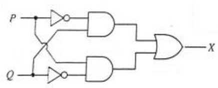


Figure 2.18: XOR function using basic gates.

Logic Circuit from Logic Functions

Example 2.48: Implement $X = AB + A(C + D)$ using logic gates.
Solution: The implementation is shown in Figure 2.19.

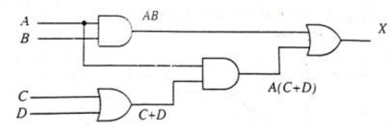


Figure 2.19: Implementation of $X = AB + A(C + D)$.

Logic Circuit from Logic Functions

Example 2.49: Draw a logic circuit for $X = \overline{(A\overline{B} + C).(AC \oplus \overline{B})}$.
Solution: The circuit is drawn in Figure 2.20.

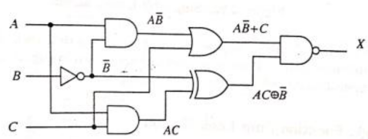


Figure 2.20: Implementation of $X = \overline{(A\overline{B} + C).(AC \oplus \overline{B})}$.

Logic Circuit from Logic Functions

Example 2.50: Draw implementation of $X = [D + (\overline{A+B}).\overline{C}].E$ using logic gates.
Solution: The implementation is drawn in Figure 2.21.

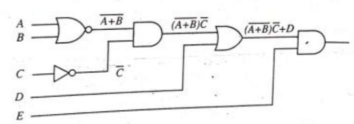


Figure 2.21: Implementation of $X = [D + (\overline{A+B}).\overline{C}].E$.

Logic Circuit from Logic Functions

Example 2.51: Implement a circuit to add two binary bits.
Solution: As we know, if two binary bits A and B are added there will be a sum (S) and a carry (C). For different values of A and B, values of S and C can be copied form the rules of binary addition and put into a truth table. The truth table is shown in Table 2.7.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

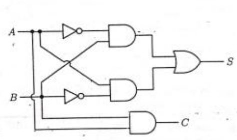


Figure 2.22: Single bit binary adder.

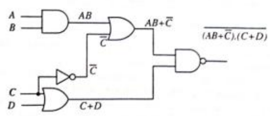
Logic Function from Logic circuit

- The steps involved are
  1. Label the inputs of all the first level gates
  2. Compute their outputs
  3. Propagate these outputs to inputs of other gates of next level
  4. Repeat until last level output is found

Logic Function from Logic circuit

Example 2.52: Derive the boolean function of the logic circuit of Figure 2.23.

Solution: As the inputs of gates on the left (level 1) are already labeled we proceed to find the outputs of these gates. The upper AND gate will output AB, the NOT will output $\overline{C}$ and the lower OR gate's output will be $C + D$. This is shown in Figure 2.24. Now propagate these to the inputs of OR and NAND gates on the right. The OR gate on top, will have AB and $\overline{C}$ on its inputs and so will produce $AB + \overline{C}$ on output. This will feed the last NAND gate. The other input of the NAND will come from the output of lower OR gate. This is $C + D$. If $AB + \overline{C} = X$ and $C + D = Y$ are the inputs of the NAND, its output will be $\overline{X.Y}$ which evaluates to $\overline{(AB + \overline{C}).(C + D)}$.



Homework Based on Lecture 7

- Questions from Chapter 2, Introduction to computers by Mohammed Alamgir
- Example 2.40-2.52
- Exercise 37, 38, 40, 41, 42, 43, 44, 47, 48, 49, 50, 51, 52, 53, 56, 58,

END