

## **Boolean Algebra**

---

### Why study Boolean Algebra?

It is highly desirable to find the simplest circuit implementation with the smallest number of gates or wires.

We can use *Boolean minimization* process to reduce a Boolean function (expression) to its simplest form: The result is an expression with the fewest literals and thus less wires in the final gate implementation.

## **Boolean Algebra (continued)**

---

- George Boole (1815-1864), a mathematician introduced a systematic treatment of logic.
- He developed a consistent set of postulates that were sufficient to define a new type of algebra: Boolean Algebra (similar to Linear Algebra)
- Many of the rules are the same as the ones in Linear Algebra.

## Laws of Boolean Algebra

---

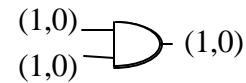
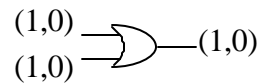
- There are 6 fundamental laws, or axioms, used to formulate various algebraic structures:

1. *Closure*: Boolean algebra operates over a field of numbers,  $B = \{0,1\}$ :

For every  $x, y$  in  $B$ :

- $x + y$  is in  $B$

- $x \cdot y$  is in  $B$



## Laws of Boolean Algebra (continued)

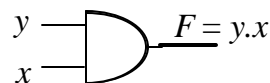
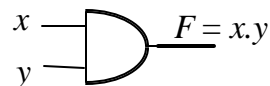
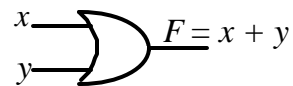
---

2. *Commutative* laws: For every  $x, y$  in  $B$ ,

- $x + y = y + x$

- $x \cdot y = y \cdot x$

» Similar to Linear Algebra

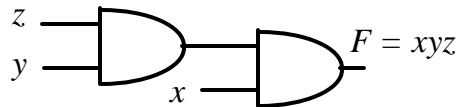
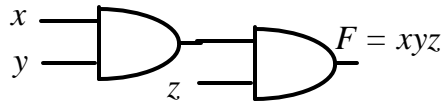


## Laws of Boolean Algebra (continued)

3. *Associative laws*: For every  $x, y, z$  in  $B$ ,

- $(x + y) + z = x + (y + z) = x + y + z$
- $(xy)z = x(yz) = xyz$

» Similar to Linear Algebra



## Laws of Boolean Algebra (continued)

4. *Distributive laws*: For every  $x, y, z$  in  $B$ ,

- $x + (y \cdot z) = (x + y)(x + z)$  [ $+$  is distributive over  $\cdot$ ]

» NOT Similar to Linear Algebra

- $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$  [ $\cdot$  is distributive over  $+$ ]

» Similar to Linear Algebra

## Laws of Boolean Algebra (continued)

### 5. Identity laws:

- A set  $B$  is said to have an identity element with respect to a binary operation  $\{.\}$  on  $B$  if there exists an element designated by 1 in  $B$  with the property:  $1 \cdot x = x$   
Example: AND operation

- A set  $B$  is said to have an identity element with respect to a binary operation  $\{+\}$  on  $B$  if there exists an element designated by 0 in  $B$  with the property:  $0 + x = x$   
Example: OR operation

» Similar to Linear Algebra

## Laws of Boolean Algebra (continued)

### 6. Complement

For each  $x$  in  $B$ , there exists an element  $x'$  in  $B$  (the complement of  $x$ ) such that:

- $x + x' = 1$
  - $x \cdot x' = 0$
- » Similar to Linear Algebra

We can also use  $\bar{x}$  to represent complement.

## Laws of Boolean Algebra (Summary)

### *Commutative*

$$x + y = y + x \quad xy = yx$$

### *Associative*

$$(x + y) + z = x + (y + z)$$

$$(xy)z = x(yz)$$

### *Distributive*

$$x + (yz) = (x + y)(x + z)$$

$$x(y + z) = (xy) + (xz)$$

### *Identity*

$$x + 0 = x \quad x \cdot 1 = x$$

### *Complement*

$$x + \bar{x} = 1 \quad x \cdot \bar{x} = 0$$

### **OR with 1    AND with 0**

$$x + 1 = 1 \quad x \cdot 0 = 0$$

## Other Theorems

### ▪ Theorem 1(a):

$$x + x = x$$

$$x + x = x$$

$$x + x = (x + x) \cdot 1$$

$$= (x + x)(x + x')$$

$$= xx + xx'$$

$$= x + 0$$

$$= x$$

### ▪ Theorem 1(b):

$$x \cdot x = x$$

$$x \cdot x = x$$

$$x \cdot x = xx + 0$$

$$= xx + xx'$$

$$= x(x + x')$$

$$= x \cdot 1$$

$$= x$$

## Other Theorems (continued)

---

- Theorem 2(a):

$$x + 1 = 1$$

$$x + 1 = 1 \cdot (x + 1)$$

$$= (x + x')(x + 1)$$

$$= xx + x' \cdot 1$$

$$= x + x'$$

$$= 1$$

- Theorem 2(b):

$$x + xy = x$$

$$x + xy = x(1 + y)$$

$$= x(y + 1)$$

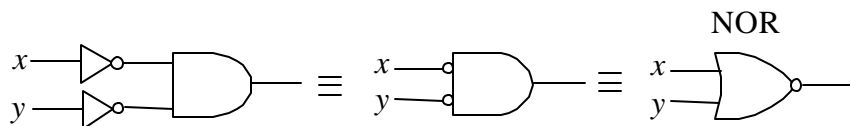
$$= x \cdot 1$$

$$= x$$

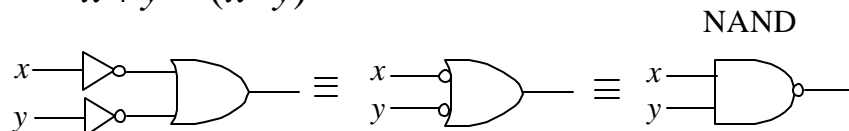
## Gate Equivalency and DeMorgan's Law

---

$$x' \cdot y' = (x + y)'$$



$$x' + y' = (x \cdot y)'$$



## Digital Logic Q's & A's

---

**Q:** Why is Gate Equivalency useful?

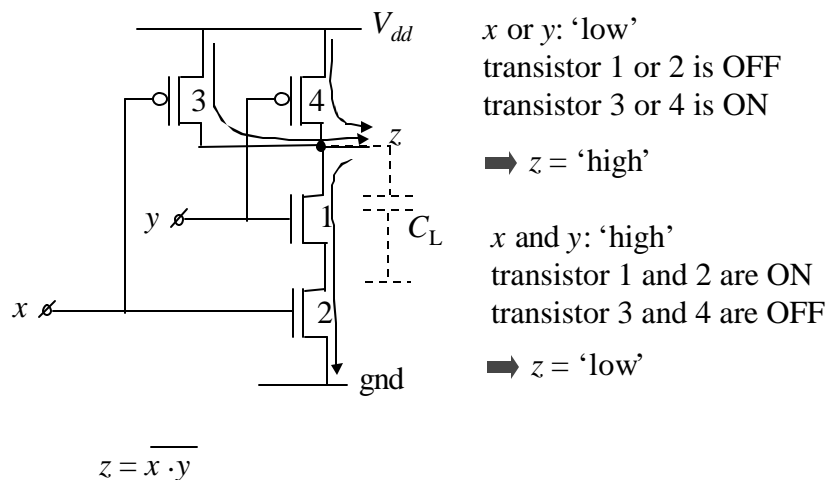
**A:** It allows us to build functions using only one gate type.

**Q:** Why are digital circuits constructed with NAND/NOR rather than with AND/OR?

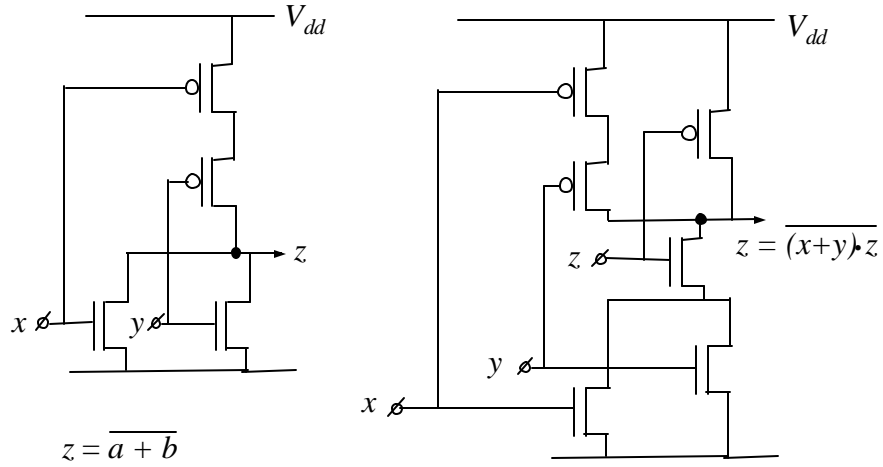
**A:** NAND and NOR gates are smaller, faster, and easier to fabricate with electronic components. They are the basic gates used in all IC digital logic.

## Digital IC's – Transistor Level

---

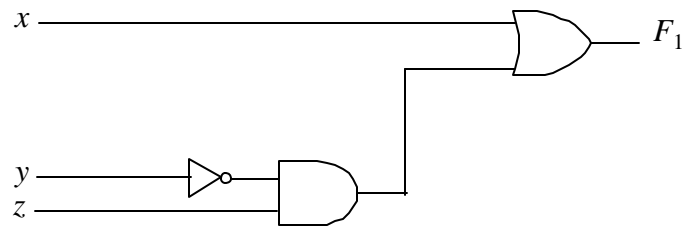


## Digital IC's (continued)



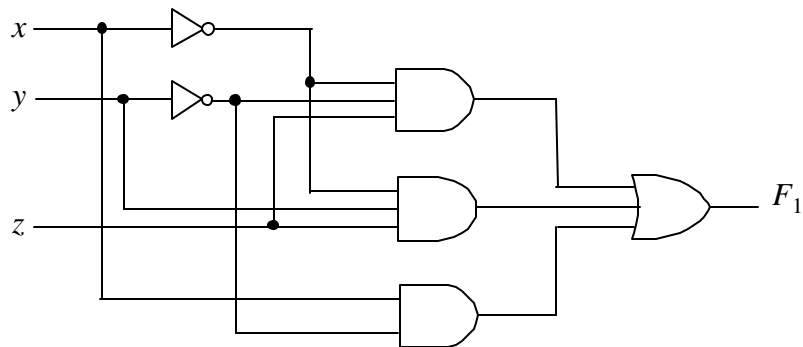
## Implementation of Boolean Functions

Example 1:  $F_1 = x + y'z$



## Implementation of Boolean Functions

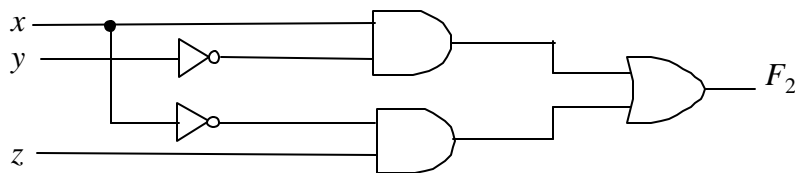
Example 2:  $F_1 = x'y'z + x'yz + xy'$



## Implementation of Boolean Functions

- Try another implementation using a simplified  $F_2$ :

$$\begin{aligned} F_2 &= x'y'z + x'yz + xy' \\ &= x'z(y'+y) + xy' \\ &= x'z(1) + xy' \\ &= x'z + xy' \end{aligned}$$



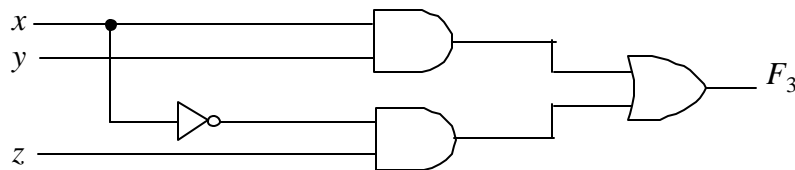
What are the advantages of this implementation?

This implementation has fewer gates and fewer inputs to the gates (or wires) than the previous one.

## Simplifying Boolean Functions

- Simplify the following Boolean function to a minimum number of terms:  $F_3 = xy + x'z + yz$

$$\begin{aligned}
 F_3 &= xy + x'z + yz \\
 &= xy + x'z + yz(x + x') \\
 &= xy + x'z + xyz + x'yz \\
 &= xy(1 + z) + x'z(1 + y) \\
 &= xy + x'z
 \end{aligned}$$



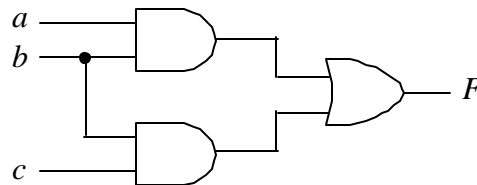
## More on complements (DeMorgan)

- Find the complement of:
 
$$\begin{aligned}
 F &= (AB'+C)D'+E \\
 F' &= [(AB'+C)D'+E]' \\
 &= [(AB'+C)D']'E' \\
 &= [(AB'+C)'+D'']E' \\
 &= [(AB')'C+D]E' \\
 &= (A'+B)C'E+DE'
 \end{aligned}$$
- Show that the complement of  $x(x+y) = x'$ 

$$\begin{aligned}
 [x(x+y)]' &= x'+(x+y)' \\
 &= x'+x'y' \\
 &= x'(1+y') \\
 &= x'(1) = x'
 \end{aligned}$$

## Implementation of Boolean Functions

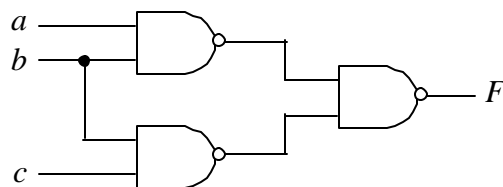
- Draw the logic diagram for the following function:  $F = (a.b) + (b.c)$



## Implementation of Boolean Functions

- Using ONLY NAND gates, draw a schematic for the following function:  $F = (a.b) + (b.c)$

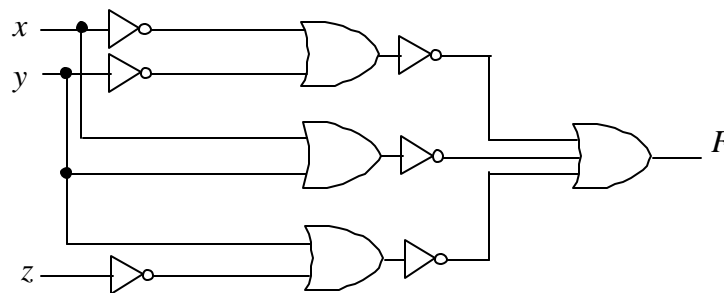
$$\begin{aligned}(F') &= [(a.b) + (b.c)]' \\ &= [(a.b)' . (b.c)']\end{aligned}$$



## Implementation of Boolean Functions

- Using only OR and NOT gates, draw a schematic for the following function:  $F = xy + x'y' + y'z$

$$\begin{aligned}(F') &= ((xy + x'y' + y'z)')' \\ &= [(xy)' \cdot (x'y')' \cdot (y'z)']' \\ &= [(x'+y') \cdot (x+y) \cdot (y+z)']' \\ &= (x'+y') + (x+y) + (y+z)'\end{aligned}$$



## Minterms and Maxterms

- MINTERMS AND MAXTERMS:

$n$  binary variables can be combined to form  $2^n$  terms (AND terms), called minterms (SOP).

In a similar fashion,  $n$  binary variables can be combined to form  $2^n$  terms (OR terms), called maxterms (POS).

\* Note that each maxterm is the complement of its corresponding minterm and vice versa.

## Minterms and Maxterms (continued)

Table 2-3:  
*Minterms and Maxterms for Three Binary Variables*

$x$	$y$	$z$	minterms		Maxterms	
0	0	0	$x'y'z'$	$m_0$	$x+y+z$	$M_0$
0	0	1	$x'y'z$	$m_1$	$x+y+z'$	$M_1$
0	1	0	$x'yz'$	$m_2$	$x+y'+z$	$M_2$
0	1	1	$x'yz$	$m_3$	$x+y'+z'$	$M_3$
1	0	0	$xy'z'$	$m_4$	$x'+y+z$	$M_4$
1	0	1	$xy'z$	$m_5$	$x'+y+z'$	$M_5$
1	1	0	$xyz'$	$m_6$	$x'+y'+z$	$M_6$
1	1	1	$xyz$	$m_7$	$x'+y'+z'$	$M_7$

## Sminterms and Pmaxterms

- Given the truth table, express  $F_1$  in sum of minterms

$x$	$y$	$z$	$F_1$	$F_2$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$$F_1(x, y, z) = \sum(1,4,5,6,7) = m_1 + m_4 + m_5 + m_6 + m_7$$

$$= (x'y'z) + (xy'z') + (xy'z) + (xyz') + (xyz)$$

- Find  $F_2$

## Sminterms and Pmaxterms

---

- Repeat for product of maxterms.

$x$	$y$	$z$	$F_1$	$F_2$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$$F_1(x, y, z) = \prod(0,2,3) = M_0 \cdot M_2 \cdot M_3$$

$$= (x + y + z)(x + y' + z)(x + y' + z')$$

## Sminterms and Pmaxterms

---

Express the Boolean function  $F = x + y'z$  in a sum of minterms, and then in a product of Maxterms.

$$x = x(y + y') = xy + xy'$$

$$xy = xy(z + z') = xyz + xyz'$$

$$xy' = xy'(z + z') = xy'z + xy'z'$$

$$y'z = y'z(x + x') = xy'z + x'y'z$$

Adding all terms and excluding recurring terms:

$$F(x, y, z) = x'y'z + xy'z' + xy'z + xyz' + xyz \quad (\text{SOP})$$

$$F(x, y, z) = m_1 + m_4 + m_5 + m_6 + m_7 = \sum(1,4,5,6,7)$$

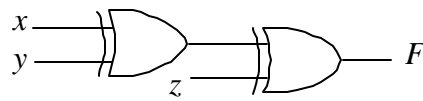
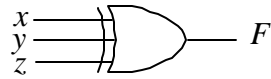
$$\text{Product of maxterms (POS)? } \prod(0,2,3) = M_0 \cdot M_2 \cdot M_3$$

## XOR Logic gate

---

3-input exclusive-OR (XOR) logic gate:

$$F = X \oplus Y \oplus Z$$



X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1