

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

07.09.2020

# Boolean algebra

Boolean algebra, like any other deductive mathematical system, may be defined with a set of elements, a set of operators, and a number of unproved axioms or postulates. The postulates of a mathematical system form the basic assumptions

1. Closure - If  $x * y = z$  then

$$x, y, z \in S$$

2. Associative law

$$x * (y * z) = (x * y) * z$$

3. Commutative law

$$(x * y) = (y * x)$$

4. Identity element

$$e * x = x * e = x$$

5. Inverse

$$x * y = e$$

6. Distributive law

$$x * (y \cdot z) = (x * y) \cdot (x * z)$$

## Field

# AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA

- ▶ In 1854, George Boole developed an algebraic system now called Boolean algebra.
- ▶ In 1938, Claude E. Shannon introduced a two-valued Boolean algebra called switching algebra that represented the properties of bistable electrical switching circuits.
- ▶ For the formal definition of Boolean algebra, we shall employ the postulates formulated by E. V. Huntington in 1904.

Boolean algebra is an algebraic structure defined by a set of elements,  $B$ , together with two binary operators,  $+$  and  $\bullet$ , and it satisfies Huntington postulates

# Huntington postulates

1. (a) The structure is closed with respect to the operator  $+$ .  
 (b) The structure is closed with respect to the operator  $\bullet$ .

# Huntington postulates

1. (a) The structure is closed with respect to the operator  $+$ .  
 (b) The structure is closed with respect to the operator  $\bullet$ .
2. (a) The element 0 is an identity element with respect to  $+$ ;  
 $x + 0 = 0 + x = x$ .  
 (b) The element 1 is an identity element with respect to  $\bullet$ ;  
 $x \bullet 1 = 1 \bullet x = x$ .

# Huntington postulates

1. (a) The structure is closed with respect to the operator  $+$ .  
 (b) The structure is closed with respect to the operator  $\bullet$ .
2. (a) The element 0 is an identity element with respect to  $+$ ;  
 $x + 0 = 0 + x = x$ .  
 (b) The element 1 is an identity element with respect to  $\bullet$ ;  
 $x \bullet 1 = 1 \bullet x = x$ .
3. (a) The structure is commutative with respect to  $+$ ;  
 $x + y = y + x$ .  
 (b) The structure is commutative with respect to  $\bullet$ ;  
 $x \bullet y = y \bullet x$ .

# Huntington postulates

4. (a) The operator  $\bullet$  is distributive over  $+$ ;  

$$x \bullet (y + z) = (x \bullet y) + (x \bullet z).$$
- (b) The operator  $+$  is distributive over  $\bullet$ ;  

$$x + (y \bullet z) = (x + y) \bullet (x + z).$$

# Huntington postulates

4. (a) The operator  $\bullet$  is distributive over  $+$ ;  

$$x \bullet (y + z) = (x \bullet y) + (x \bullet z).$$
 (b) The operator  $+$  is distributive over  $\bullet$ ;  

$$x + (y \bullet z) = (x + y) \bullet (x + z).$$
5. For every element  $x \in B$ , there exists an element  $x' \in B$  (called the complement of  $x$ ) such that
 

(a)  $x + x' = 1$

(b)  $x \bullet x' = 0$



# Huntington postulates

4. (a) The operator  $\bullet$  is distributive over  $+$ ;  

$$x \bullet (y + z) = (x \bullet y) + (x \bullet z).$$
 (b) The operator  $+$  is distributive over  $\bullet$ ;  

$$x + (y \bullet z) = (x + y) \bullet (x + z).$$
5. For every element  $x \in B$ , there exists an element  $x' \in B$  (called the complement of  $x$ ) such that
 

(a)  $x + x' = 1$

(b)  $x \bullet x' = 0$
6. There exist at least two elements  $x, y \in B$  such that  $x \neq y$ .

# Two-Valued Boolean Algebra

$x$	$y$	$x \bullet y$
0	0	0
0	1	0
1	0	0
1	1	1

$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

$x$	$x'$
0	1
1	0

# Duality Principle

## Definition (Duality Principle)

It states that every algebraic expression deducible from the postulates of Boolean algebra remains **valid** if the operators and identity elements are interchanged.

# Duality Principle

## Definition (Duality Principle)

It states that every algebraic expression deducible from the postulates of Boolean algebra remains **valid** if the operators and identity elements are interchanged.

**In short...**

If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

# Duality Principle

## Definition (Duality Principle)

It states that every algebraic expression deducible from the postulates of Boolean algebra remains **valid** if the operators and identity elements are interchanged.

**In short...**

If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

## Example (Postualte 2)

Expression

$$x + 0 = x$$

Dual

$$x \bullet 1 = x$$

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

08.09.2020

# Postulates

## Postulate 2

a)  $x + 0 = 0 + x = x$

b)  $x \bullet 1 = 1 \bullet x = x$

## Postulate 3

a)  $x + y = y + x$

b)  $x \bullet y = y \bullet x$

## Postulate 4

a)  $x \bullet (y + z) = (x \bullet y) + (x \bullet z)$

b)  $x + (y \bullet z) = (x + y) \bullet (x + z)$

## Postulate 5

a)  $x + x' = 1$

b)  $x \bullet x' = 0$

# Theorem 1

## Theorem (1 a)

$$x + x = x$$



# Theorem 1

## Theorem (1 a)

$$x + x = x$$

Proof.

$$x + x = (x + x) \bullet 1 \quad \text{Postulate 2b}$$

# Theorem 1

## Theorem (1 a)

$$x + x = x$$

Proof.

$$\begin{aligned} x + x &= (x + x) \bullet 1 && \text{Postulate 2b} \\ &= (x + x)(x + x') && \text{Postulate 5a} \end{aligned}$$

# Theorem 1

## Theorem (1 a)

$$x + x = x$$

Proof.

$$\begin{aligned}x + x &= (x + x) \bullet 1 && \text{Postulate 2b} \\ &= (x + x)(x + x') && \text{Postulate 5a} \\ &= x + xx' && \text{Postulate 4b}\end{aligned}$$

# Theorem 1

## Theorem (1 a)

$$x + x = x$$

Proof.

$x + x$	$= (x + x) \bullet 1$	Postulate 2b
	$= (x + x)(x + x')$	Postulate 5a
	$= x + xx'$	Postulate 4b
	$= x + 0$	Postulate 5b

# Theorem 1

## Theorem (1 a)

$$x + x = x$$

Proof.

$x + x$	$= (x + x) \bullet 1$	Postulate 2b
	$= (x + x)(x + x')$	Postulate 5a
	$= x + xx'$	Postulate 4b
	$= x + 0$	Postulate 5b
	$= x$	Postulate 2a



# Theorem 1

## Theorem (1 b)

$$x \bullet x = x$$

# Theorem 1

## Theorem (1 b)

$$x \bullet x = x$$

Proof.

$$x \bullet x = (x \bullet x) + 0 \quad \text{Postulate 2a}$$

# Theorem 1

## Theorem (1 b)

$$x \bullet x = x$$

Proof.

$$\begin{aligned} x \bullet x &= (x \bullet x) + 0 && \text{Postulate 2a} \\ &= (xx) + (xx') && \text{Postulate 5b} \end{aligned}$$



# Theorem 1

## Theorem (1 b)

$$x \bullet x = x$$

Proof.

$$\begin{aligned} x \bullet x &= (x \bullet x) + 0 && \text{Postulate 2a} \\ &= (xx) + (xx') && \text{Postulate 5b} \\ &= x(x + x') && \text{Postulate 4a} \end{aligned}$$

# Theorem 1

## Theorem (1 b)

$$x \bullet x = x$$

Proof.

$x \bullet x$	$= (x \bullet x) + 0$	Postulate 2a
	$= (xx) + (xx')$	Postulate 5b
	$= x(x + x')$	Postulate 4a
	$= x \bullet 1$	Postulate 5a

# Theorem 1

## Theorem (1 b)

$$x \bullet x = x$$

Proof.

$x \bullet x$	$= (x \bullet x) + 0$	Postulate 2a
	$= (xx) + (xx')$	Postulate 5b
	$= x(x + x')$	Postulate 4a
	$= x \bullet 1$	Postulate 5a
	$= x$	Postulate 2b



# Theorem 2

## Theorem (2 a)

$$x + 1 = 1$$

## Theorem 2

### Theorem (2 a)

$$x + 1 = 1$$

Proof.

$$x + 1 = 1 \bullet (x + 1) \quad \text{Postulate 2b}$$

## Theorem 2

### Theorem (2 a)

$$x + 1 = 1$$

Proof.

$$\begin{aligned} x + 1 &= 1 \bullet (x + 1) && \text{Postulate 2b} \\ &= (x + x')(x + 1) && \text{Postulate 5a} \end{aligned}$$

## Theorem 2

### Theorem (2 a)

$$x + 1 = 1$$

Proof.

$$\begin{aligned}x + 1 &= 1 \bullet (x + 1) && \text{Postulate 2b} \\ &= (x + x')(x + 1) && \text{Postulate 5a} \\ &= x + (x' \bullet 1) && \text{Postulate 4b}\end{aligned}$$

## Theorem 2

### Theorem (2 a)

$$x + 1 = 1$$

Proof.

$x + 1$	$= 1 \bullet (x + 1)$	Postulate 2b
	$= (x + x')(x + 1)$	Postulate 5a
	$= x + (x' \bullet 1)$	Postulate 4b
	$= x + x'$	Postulate 2b



## Theorem 2

### Theorem (2 a)

$$x + 1 = 1$$

Proof.

$x + 1$	$= 1 \bullet (x + 1)$	Postulate 2b
	$= (x + x')(x + 1)$	Postulate 5a
	$= x + (x' \bullet 1)$	Postulate 4b
	$= x + x'$	Postulate 2b
	$= x$	Postulate 5a



## Theorem 2

### Theorem (2 a)

$$x + 1 = 1$$

Proof.

$x + 1$	$= 1 \bullet (x + 1)$	Postulate 2b
	$= (x + x')(x + 1)$	Postulate 5a
	$= x + (x' \bullet 1)$	Postulate 4b
	$= x + x'$	Postulate 2b
	$= x$	Postulate 5a



### Theorem (2 b)

$$x \bullet 0 = 0$$

*by duality*

# Basic Theorems

## Duality Principle

### Theorem 1

a)  $x + x = x$

### Theorem 2

a)  $x + 1 = 1$

### Theorem 3

$$(x')' = x$$

### Theorem 4

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

### Theorem 5

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

### Theorem 6

a)  $x + xy = x$

b)  $x \bullet x = x$

b)  $x \bullet 0 = 0$

**(involution)**

**(Associative)**

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

**(DeMorgan)**

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

**(Absorption)**

b)  $x(x + y) = x$

# Theorem 5 - DeMorgan

## Theorem (5 a)

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

# Theorem 5 - DeMorgan

## Theorem (5 a)

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

Proof.

$x$	$y$	$x + y$	$(x + y)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$x'$	$y'$	$x'y'$
1	1	1
1	0	0
0	1	0
0	0	0



## Theorem 5 - DeMorgan

### Theorem (5 a)

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

Proof.

$x$	$y$	$x + y$	$(x + y)'$	$x'$	$y'$	$x'y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

### Theorem (5 b)

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

*by duality*



# Theorem 6 - Absorption theorem

## Theorem (6 a)

$$x + xy = x$$

# Theorem 6 - Absorption theorem

## Theorem (6 a)

$$x + xy = x$$

Proof.

$$x + xy = x \bullet 1 + xy \quad \text{Postulate 2b}$$



# Theorem 6 - Absorption theorem

## Theorem (6 a)

$$x + xy = x$$

Proof.

$$\begin{aligned} x + xy &= x \bullet 1 + xy && \text{Postulate 2b} \\ &= x(1 + y) && \text{Postulate 4a} \end{aligned}$$

## Theorem 6 - Absorption theorem

### Theorem (6 a)

$$x + xy = x$$

Proof.

$$\begin{aligned}x + xy &= x \bullet 1 + xy && \text{Postulate 2b} \\&= x(1 + y) && \text{Postulate 4a} \\&= x \bullet 1 && \text{Postulate 2a}\end{aligned}$$

## Theorem 6 - Absorption theorem

### Theorem (6 a)

$$x + xy = x$$

Proof.

$x + xy$	$= x \bullet 1 + xy$	Postulate 2b
	$= x(1 + y)$	Postulate 4a
	$= x \bullet 1$	Postulate 2a
	$= x$	Postulate 2b



### Theorem (6 b)

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

*by duality*

# Operator Precedence

The operator precedence for evaluating Boolean expressions is

- † parentheses
- † NOT
- † AND
- † OR

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

09.09.2020

# Boolean Functions

Consider the Boolean Function  $F = xy + y'z$

Truth Table

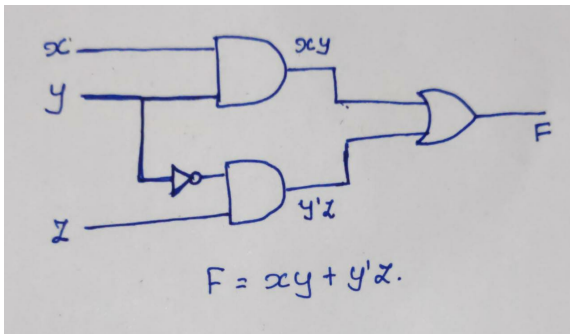
x	y	z	xy	y'z	F
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	1

# Circuit Diagram

A Boolean function can be transformed from an algebraic expression into a circuit diagram composed of logic gates connected in a particular structure.








## Circuit Diagram

A Boolean function can be transformed from an algebraic expression into a circuit diagram composed of logic gates connected in a particular structure.



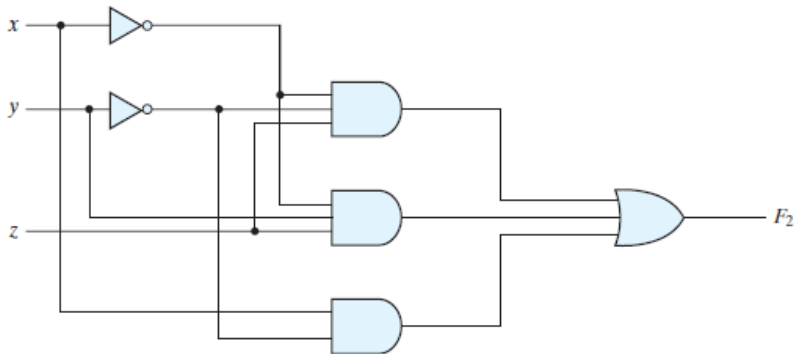


# Logic Gates

NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
$\overline{A}$	$AB$	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$																																																																																																
																																																																																																						
<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																					
0	1																																																																																																					
1	0																																																																																																					
B	A	X																																																																																																				
0	0	0																																																																																																				
0	1	0																																																																																																				
1	0	0																																																																																																				
1	1	1																																																																																																				
B	A	X																																																																																																				
0	0	1																																																																																																				
0	1	1																																																																																																				
1	0	1																																																																																																				
1	1	0																																																																																																				
B	A	X																																																																																																				
0	0	0																																																																																																				
0	1	1																																																																																																				
1	0	1																																																																																																				
1	1	1																																																																																																				
B	A	X																																																																																																				
0	0	1																																																																																																				
0	1	0																																																																																																				
1	0	0																																																																																																				
1	1	0																																																																																																				
B	A	X																																																																																																				
0	0	0																																																																																																				
0	1	1																																																																																																				
1	0	1																																																																																																				
1	1	0																																																																																																				
B	A	X																																																																																																				
0	0	1																																																																																																				
0	1	0																																																																																																				
1	0	0																																																																																																				
1	1	1																																																																																																				

## Boolean Function

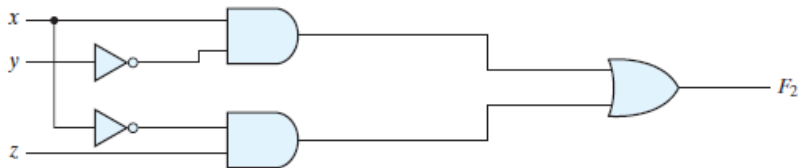
Consider the boolean function  $F_2 = x'y'z + x'yz + xy'$



## Boolean Function

The boolean function  $F_2 = x'y'z + x'yz + xy'$  can be reduced as follows

$$\begin{aligned} x'y'z + x'yz + xy' &= x'z(y' + y) + xy' && \text{Postulate 4a} \\ &= x'z + xy' \end{aligned}$$



(b)  $F_2 = xy' + x'z$

# Truth Table

$x$	$y$	$z$	$x'y'z$	$x'yz$	$xy'$	$x'y'z + x'yz + xy'$	$x'z$	$x'z + xy'$
0	0	0	0	0	0	<b>0</b>	0	<b>0</b>
0	0	1	1	0	0	<b>1</b>	1	<b>1</b>
0	1	0	0	0	0	<b>0</b>	0	<b>0</b>
0	1	1	0	1	0	<b>1</b>	1	<b>1</b>
1	0	0	0	0	1	<b>1</b>	0	<b>1</b>
1	0	1	0	0	1	<b>1</b>	0	<b>1</b>
1	1	0	0	0	0	<b>0</b>	0	<b>0</b>
1	1	1	0	0	0	<b>0</b>	0	<b>0</b>

## Simplification

**Question** Simplify the following Boolean function to a minimum number of literals

a.  $F = (x + y)(x + y')$

b.  $F = xy + x'z + yz$

## Simplification

**Question** Simplify the following Boolean function to a minimum number of literals

a.  $F = (x + y)(x + y')$

b.  $F = xy + x'z + yz$

$$\begin{aligned}(x + y)(x + y') &= xx + xy' + xy + yy' \\ &= x + x(y' + y) + 0 = x + x + 0 \\ &= x\end{aligned}$$

# Simplification

**Question** Simplify the following Boolean function to a minimum number of literals

a.  $F = (x + y)(x + y')$

b.  $F = xy + x'z + yz$

$$\begin{aligned}(x + y)(x + y') &= xx + xy' + xy + yy' \\ &= x + x(y' + y) + 0 = x + x + 0 \\ &= x\end{aligned}$$

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

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

14.09.2020



## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of  $0's$  for  $1's$  and  $1's$  for  $0's$  in the value of  $F$ .

## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of  $F$ .
- ▶ The complement of a function may be derived algebraically through DeMorgan's theorems.

## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of  $F$ .
- ▶ The complement of a function may be derived algebraically through DeMorgan's theorems.

$$F = (A + B + C)$$

## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of  $0's$  for  $1's$  and  $1's$  for  $0's$  in the value of  $F$ .
- ▶ The complement of a function may be derived algebraically through DeMorgan's theorems.

$$F = (A + B + C)$$

$$F' = (A + B + C)'$$

$$\text{Let } x = B + C$$

## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of  $F$ .
- ▶ The complement of a function may be derived algebraically through DeMorgan's theorems.

$$\begin{aligned} F &= (A + B + C) \\ F' &= (A + B + C)' \\ &= (A + x)' \end{aligned}$$

$$\begin{aligned} \text{Let } x &= B + C \\ \text{Apply DeMorgan Law} \\ (x + y)' &= x'y' \end{aligned}$$

## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of  $F$ .
- ▶ The complement of a function may be derived algebraically through DeMorgan's theorems.

$$\begin{aligned}F &= (A + B + C) \\F' &= (A + B + C)' \\&= (A + x)' \\&= A'x' = (A)'(B + C)'\end{aligned}$$

$$\begin{aligned}\text{Let } x &= B + C \\ \text{Apply DeMorgan Law} \\ (x + y)' &= x'y'\end{aligned}$$

## Complement of a Function

- ▶ The complement of a function  $F$  is  $F'$  and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of  $F$ .
- ▶ The complement of a function may be derived algebraically through DeMorgan's theorems.

$$F = (A + B + C)$$

$$F' = (A + B + C)'$$

$$= (A + x)'$$

$$= A'x' = (A)'(B + C)'$$

$$F' = (A')(B'C') = A'B'C'$$

$$\text{Let } x = B + C$$

Apply DeMorgan Law

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

# Compliment of a function

AN EXAMPLE : Find the compliment of the given function

$$F = x'y'z + xyz'$$



## Compliment of a function

AN EXAMPLE : Find the compliment of the given function

$$F = x'y'z + xyz'$$

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

$$\text{Let } A = x'y'z \text{ and } B = xyz'$$

## Compliment of a function

AN EXAMPLE : Find the compliment of the given function

$$F = x'y'z + xyz'$$

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

$$= (A + B)'$$

$$\text{Let } A = x'y'z \text{ and } B = xyz'$$

$$\text{DeMorgan Law } (x + y)' = x'y'$$

## Compliment of a function

AN EXAMPLE : Find the compliment of the given function

$$F = x'y'z + xyz'$$

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

$$= (A + B)'$$

$$= A'B' = (x'y'z)'(xyz')'$$

$$\text{Let } A = x'y'z \text{ and } B = xyz'$$

$$\text{DeMorgan Law } (x + y)' = x'y'$$

## Compliment of a function

AN EXAMPLE : Find the compliment of the given function

$$F = x'y'z + xyz'$$

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

$$= (A + B)'$$

$$= A'B' = (x'y'z)'(xyz')'$$

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

$$\text{Let } A = x'y'z \text{ and } B = xyz'$$

$$\text{DeMorgan Law } (x + y)' = x'y'$$

## Compliment of a function

ANOTHER EXAMPLE : Find the compliment of the given function

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

## Compliment of a function

ANOTHER EXAMPLE : Find the compliment of the given function

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

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

$$\text{Let } A = (x'y + xy')$$

## Compliment of a function

ANOTHER EXAMPLE : Find the compliment of the given function

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

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

$$\text{Let } A = (x'y + xy')$$

$$\text{DeMorgan Law } (xy)' = x' + y'$$

## Compliment of a function

ANOTHER EXAMPLE : Find the compliment of the given function

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

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

$$= (Az)'$$

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

$$\text{Let } A = (x'y + xy')$$

$$\text{DeMorgan Law } (xy)' = x' + y'$$



## Compliment of a function

ANOTHER EXAMPLE : Find the compliment of the given function

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

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

$$= (Az)'$$

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

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

$$\text{Let } A = (x'y + xy')$$

$$\text{DeMorgan Law } (xy)' = x' + y'$$

## Compliment of a function

ANOTHER EXAMPLE : Find the compliment of the given function

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

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

$$= (Az)'$$

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

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

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

$$\text{Let } A = (x'y + xy')$$

$$\text{DeMorgan Law } (xy)' = x' + y'$$

# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).

# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).
- ▶ Consider two binary variables  $x$  and  $y$  combined with an **AND** operation

# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).
- ▶ Consider two binary variables  $x$  and  $y$  combined with an **AND** operation
- ▶  $x'y'$       ▶  $x'y$       ▶  $xy$       ▶  $xy'$

# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).
- ▶ Consider two binary variables  $x$  and  $y$  combined with an **AND** operation
- ▶  $x'y'$       ▶  $x'y$       ▶  $xy$       ▶  $xy'$
- ▶ Each of these four AND terms is called a *minterm*, or a *standard product*.

# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).
- ▶ Consider two binary variables  $x$  and  $y$  combined with an **AND** operation
- ▶  $x'y'$       ▶  $x'y$       ▶  $xy$       ▶  $xy'$
- ▶ Each of these four AND terms is called a *minterm*, or a *standard product*.
- ▶  $n$  variables can be combined to form  $2^n$  minterms

# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).
- ▶ Consider two binary variables  $x$  and  $y$  combined with an **AND** operation
- ▶  $x'y'$                       ▶  $x'y$                       ▶  $xy$                       ▶  $xy'$
- ▶ Each of these four AND terms is called a *minterm*, or a *standard product*.
- ▶  $n$  variables can be combined to form  $2^n$  minterms

## Maxterms



# MINTERMS AND MAXTERMS

## Minterms

- ▶ A binary variable may appear either in its normal form ( $x$ ) or in its complement form ( $x'$ ).
- ▶ Consider two binary variables  $x$  and  $y$  combined with an **AND** operation
- ▶  $x'y'$       ▶  $x'y$       ▶  $xy$       ▶  $xy'$
- ▶ Each of these four AND terms is called a *minterm*, or a *standard product*.
- ▶  $n$  variables can be combined to form  $2^n$  minterms

## Maxterms

- ▶  $n$  variables forming an OR term, with each variable being primed or unprimed, provide  $2^n$  possible combinations, called *maxterms*, or *standard sums*.

# MINTERMS AND MAXTERMS

			Minterms		Maxterms	
x	y	z	Terms	Designation	Terms	Designation
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	$xyz$	$m_7$	$x' + y' + z'$	$M_7$

# CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in ***canonical form***.

## CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$\begin{aligned}F_1 &= x'y'z' + x'yz + xy'z' + xyz \\ &= m_0 + m_3 + m_4 + m_7\end{aligned}$$

## CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$F_1 = x'y'z' + x'yz + xy'z' + xyz$$

$$= m_0 + m_3 + m_4 + m_7$$

$$F_1' = x'y'z + x'yz' + xy'z + xyz'$$

$$= m_1 + m_2 + m_5 + m_6$$

# CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$F_1 = x'y'z' + x'yz + xy'z' + xyz$$

$$= m_0 + m_3 + m_4 + m_7$$

$$F_1' = x'y'z + x'yz' + xy'z + xyz'$$

$$= m_1 + m_2 + m_5 + m_6$$

$$F_1' = (x'y'z' + x'yz + xy'z' + xyz)'$$

# CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$F_1 = x'y'z' + x'yz + xy'z' + xyz$$

$$= m_0 + m_3 + m_4 + m_7$$

$$F_1' = x'y'z + x'yz' + xy'z + xyz'$$

$$= m_1 + m_2 + m_5 + m_6$$

$$F_1' = (x'y'z' + x'yz + xy'z' + xyz)'$$

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

# CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$F_1 = x'y'z' + x'yz + xy'z' + xyz$$

$$= m_0 + m_3 + m_4 + m_7$$

$$F_1' = x'y'z + x'yz' + xy'z + xyz'$$

$$= m_1 + m_2 + m_5 + m_6$$

$$F_1' = (x'y'z' + x'yz + xy'z' + xyz)'$$

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

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



# CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$F_1 = x'y'z' + x'yz + xy'z' + xyz$$

$$= m_0 + m_3 + m_4 + m_7$$

$$F'_1 = x'y'z + x'yz' + xy'z + xyz'$$

$$= m_1 + m_2 + m_5 + m_6$$

$$F'_1 = (x'y'z' + x'yz + xy'z' + xyz)'$$

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

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

$$= M_0 \bullet M_3 \bullet M_4 \bullet M_7$$

# CANONICAL FORM

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in **canonical form**.

$$F_1 = x'y'z' + x'yz + xy'z' + xyz$$

$$= m_0 + m_3 + m_4 + m_7$$

$$F_1' = x'y'z + x'yz' + xy'z + xyz'$$

$$= m_1 + m_2 + m_5 + m_6$$

$$F_1' = (x'y'z' + x'yz + xy'z' + xyz)'$$

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

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

$$= M_0 \bullet M_3 \bullet M_4 \bullet M_7$$

$$= M_0 M_3 M_4 M_7$$

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

14.09.2020

# CANONICAL FORM

Boolean functions expressed as a  
sum of minterms or  
product of maxterms  
are said to be in ***canonical form***.

# CANONICAL FORM

Boolean functions expressed as a  
sum of minterms or  
product of maxterms  
are said to be in **canonical form**.

Question: Is the Boolean function  $F = A + B'C$  in canonical form?

# CANONICAL FORM

Boolean functions expressed as a  
sum of minterms or  
product of maxterms  
are said to be in **canonical form**.

Question: Is the Boolean function  $F = A + B'C$  in canonical form?

Answer: NO

# CANONICAL FORM

Boolean functions expressed as a  
sum of minterms or  
product of maxterms  
are said to be in **canonical form**.

Question: Is the Boolean function  $F = A + B'C$  in canonical form?

Answer: **NO**

Question: Express the Boolean function  $F = A + B'C$  as a sum of minterms.

$$F = A + B'C$$

Three variables -  $A$ ,  $B$ , and  $C$



$$F = A + B'C$$

Three variables -  $A$ ,  $B$ , and  $C$

Two terms

▶  $A$

▶  $B'C$

$$F = A + B'C$$

Three variables -  $A$ ,  $B$ , and  $C$

Two terms

▶  $A$

▶  $B'C$

The first term  $A$  is missing two variables

$$F = A + B'C$$

Three variables -  $A$ ,  $B$ , and  $C$

Two terms

▶  $A$

▶  $B'C$

The first term  $A$  is missing two variables

$$A = A(B + B') = AB + AB'$$

$$F = A + B'C$$

Three variables -  $A$ ,  $B$ , and  $C$

Two terms

▶  $A$

▶  $B'C$

The first term  $A$  is missing two variables

$$\begin{aligned} A &= A(B + B') = AB + AB' \\ &= AB(C + C') + AB'(C + C') \end{aligned}$$

$$F = A + B'C$$

Three variables -  $A$ ,  $B$ , and  $C$

Two terms

▶  $A$

▶  $B'C$

The first term  $A$  is missing two variables

$$\begin{aligned} A &= A(B + B') = AB + AB' \\ &= AB(C + C') + AB'(C + C') \\ A &= \textcolor{violet}{ABC} + \textcolor{violet}{ABC'} + \textcolor{violet}{AB'C} + \textcolor{violet}{AB'C'} \end{aligned}$$

$$F = A + B'C$$

▶  $A = ABC + ABC' + AB'C + AB'C'$

$$F = A + B'C$$

- ▶  $A = ABC + ABC' + AB'C + AB'C'$
- ▶ The second term  $B'C$  is missing two variables

$$F = A + B'C$$

- ▶  $A = ABC + ABC' + AB'C + AB'C'$
- ▶ The second term  $B'C$  is missing two variables
- ▶  $B'C = B'C(A + A') = AB'C + A'B'C$



$$F = A + B'C$$

- ▶  $A = ABC + ABC' + AB'C + AB'C'$
- ▶ The second term  $B'C$  is missing two variables
- ▶  $B'C = B'C(A + A') = AB'C + A'B'C$
- ▶ Combining the two terms

$$F = ABC + ABC' + AB'C + AB'C' + A'B'C$$

$$F = A + B'C$$

- ▶  $A = ABC + ABC' + AB'C + AB'C'$
- ▶ The second term  $B'C$  is missing two variables
- ▶  $B'C = B'C(A + A') = AB'C + A'B'C$
- ▶ Combining the two terms

$$F = ABC + ABC' + AB'C + AB'C' + A'B'C$$

$$F = m_7 + m_6 + m_5 + m_4 + m_1$$

$$F = m_1 + m_4 + m_5 + m_6 + m_7$$

$$F = A + B'C$$

- ▶  $A = ABC + ABC' + AB'C + AB'C'$
- ▶ The second term  $B'C$  is missing two variables
- ▶  $B'C = B'C(A + A') = AB'C + A'B'C$
- ▶ Combining the two terms

$$F = ABC + ABC' + AB'C + AB'C' + A'B'C$$

$$F = m_7 + m_6 + m_5 + m_4 + m_1$$

$$F = m_1 + m_4 + m_5 + m_6 + m_7$$

$$F = \Sigma(1, 4, 5, 6, 7)$$

$$F = A + B'C = \Sigma(1, 4, 5, 6, 7)$$

$A$	$B$	$C$	$B'C$	$F = A + B'C$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

Question: Express the Boolean function  $F = A + B'C$  as a product of maxterms.

Any missing variable  $x$  in each OR term is ORed with  $xx'$ .

Question: Express the Boolean function  $F = A + B'C$  as a product of maxterms.

Any missing variable  $x$  in each OR term is ORed with  $xx'$ .

$$\begin{aligned} F &= A + B'C \\ &= (A + B')(A + C) \end{aligned}$$

Question: Express the Boolean function  $F = A + B'C$  as a product of maxterms.

Any missing variable  $x$  in each OR term is ORed with  $xx'$ .

$$\begin{aligned} F &= A + B'C \\ &= (A + B')(A + C) \end{aligned}$$

There are two terms  $(A + B')$  and  $(A + C)$

$$(A + B') = (A + B') + (CC')$$

$$(A + B') = (A + B' + C)(A + B' + C')$$

$$(A + C) = (A + C) + (BB')$$

$$(A + C) = (A + B + C)(A + B' + C)$$

$$F = (A + B')(A + C)$$

$$(A + B') = (A + B' + C)(A + B' + C')$$

$$(A + C) = (A + B + C)(A + B' + C)$$

Combining the two terms,

$$F = (A + B + C)(A + B' + C)(A + B' + C')$$

$$F = M_0 M_2 M_3$$



$$F = (A + B')(A + C)$$

$$(A + B') = (A + B' + C)(A + B' + C')$$

$$(A + C) = (A + B + C)(A + B' + C)$$

Combining the two terms,

$$F = (A + B + C)(A + B' + C)(A + B' + C')$$

$$F = M_0 M_2 M_3$$

$$F = \Pi(0, 2, 3)$$

$$F = A + B'C = (A + B')(A + C) = \Pi(0, 2, 3)$$

$A$	$B$	$C$	$B'C$	$F = A + B'C$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

# Conversion between Canonical Forms

$$F(A, B, C) = A + B'C = \Sigma(1, 4, 5, 6, 7)$$

# Conversion between Canonical Forms

$$F(A, B, C) = A + B'C = \Sigma(1, 4, 5, 6, 7)$$

$$F'(A, B, C) = \Sigma(0, 2, 3)$$

# Conversion between Canonical Forms

$$F(A, B, C) = A + B'C = \Sigma(1, 4, 5, 6, 7)$$

$$F'(A, B, C) = \Sigma(0, 2, 3)$$

If we take the complement of  $F'$  by DeMorgan's theorem, we obtain  $F$  in a different form:

# Conversion between Canonical Forms

$$F(A, B, C) = A + B'C = \Sigma(1, 4, 5, 6, 7)$$

$$F'(A, B, C) = \Sigma(0, 2, 3)$$

If we take the complement of  $F'$  by DeMorgan's theorem, we obtain  $F$  in a different form:

$$F = (m_0 + m_2 + m_3)' = m'_0 \bullet m'_2 \bullet m'_3 = M_0M_2M_3 = \Pi(0, 2, 3)$$

$$F(A, B, C) = \Pi(0, 2, 3)$$

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

16.09.2020

# Standard Forms

- ▶ Another way to express Boolean functions is in *standard form*.



# Standard Forms

- ▶ Another way to express Boolean functions is in *standard form*.
- ▶ In this configuration, the terms that form the function may contain one, two, or any number of literals.

# Standard Forms

- ▶ Another way to express Boolean functions is in *standard form*.
- ▶ In this configuration, the terms that form the function may contain one, two, or any number of literals.
- ▶ There are two types of standard forms:

# Standard Forms

- ▶ Another way to express Boolean functions is in *standard form*.
- ▶ In this configuration, the terms that form the function may contain one, two, or any number of literals.
- ▶ There are two types of standard forms:
  - ▶ **Sum of products**

# Standard Forms

- ▶ Another way to express Boolean functions is in *standard form*.
- ▶ In this configuration, the terms that form the function may contain one, two, or any number of literals.
- ▶ There are two types of standard forms:
  - ▶ **Sum of products**
  - ▶ Example:  $F = x + y'z + x'y'z$

# Standard Forms

- ▶ Another way to express Boolean functions is in *standard form*.
- ▶ In this configuration, the terms that form the function may contain one, two, or any number of literals.
- ▶ There are two types of standard forms:
  - ▶ **Sum of products**
  - ▶ Example:  $F = x + y'z + x'y'z$
  - ▶ **Products of sums**

# Standard Forms

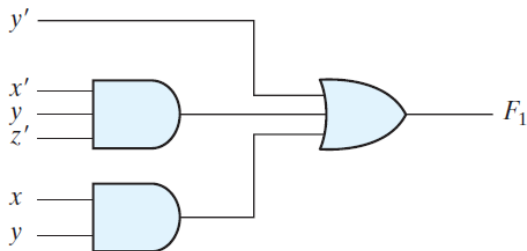
- ▶ Another way to express Boolean functions is in *standard form*.
- ▶ In this configuration, the terms that form the function may contain one, two, or any number of literals.
- ▶ There are two types of standard forms:
  - ▶ **Sum of products**
    - ▶ Example:  $F = x + y'z + x'y'z$
  - ▶ **Products of sums**
    - ▶ Example:  $F = x(y + z')(x' + y' + z')$

# Two-level implementation

$$F_1 = y' + xy + x'yz'$$

# Two-level implementation

$$F_1 = y' + xy + x'yz'$$



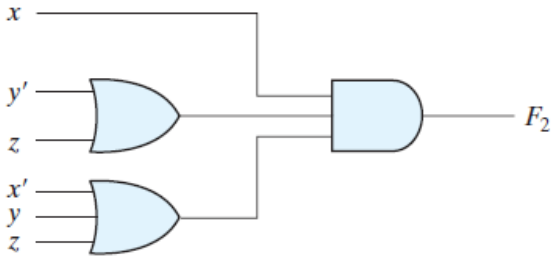


## Two-level implementation

$$F_2 = x(y' + z)(x' + y + z)'$$

## Two-level implementation

$$F_2 = x(y' + z)(x' + y + z)'$$



## Two-level implementation

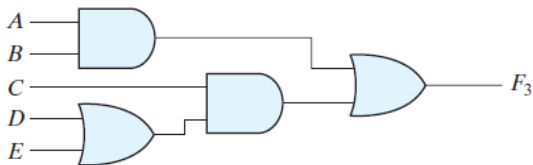
►  $F_3 = AB + C(D + E)$

## Two-level implementation

- ▶  $F_3 = AB + C(D + E)$
- ▶ The function is not in standard form.

## Two-level implementation

- ▶  $F_3 = AB + C(D + E)$
- ▶ The function is not in standard form.
- ▶ 2-level representation is not possible.



(a)  $AB + C(D + E)$

## Two-level implementation

►  $F_3 = AB + C(D + E)$

## Two-level implementation

- ▶  $F_3 = AB + C(D + E)$
- ▶ The function can be converted to standard form.

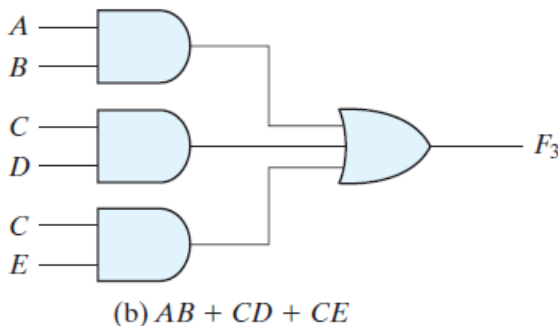
## Two-level implementation

- ▶  $F_3 = AB + C(D + E)$
- ▶ The function can be converted to standard form.
- ▶  $F_3 = AB + C(D + E) = AB + CD + CE$



## Two-level implementation

- ▶  $F_3 = AB + C(D + E)$
- ▶ The function can be converted to standard form.
- ▶  $F_3 = AB + C(D + E) = AB + CD + CE$



# 16 Functions of Two Binary Variables

*Truth Tables for the 16 Functions of Two Binary Variables*

<i>x</i>	<i>y</i>	<i>F</i> <sub>0</sub>	<i>F</i> <sub>1</sub>	<i>F</i> <sub>2</sub>	<i>F</i> <sub>3</sub>	<i>F</i> <sub>4</sub>	<i>F</i> <sub>5</sub>	<i>F</i> <sub>6</sub>	<i>F</i> <sub>7</sub>	<i>F</i> <sub>8</sub>	<i>F</i> <sub>9</sub>	<i>F</i> <sub>10</sub>	<i>F</i> <sub>11</sub>	<i>F</i> <sub>12</sub>	<i>F</i> <sub>13</sub>	<i>F</i> <sub>14</sub>	<i>F</i> <sub>15</sub>
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

*Boolean Expressions for the 16 Functions of Two Variables*

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	$x$ and $y$
$F_2 = xy'$	$x/y$	Inhibition	$x$ , but not $y$
$F_3 = x$		Transfer	$x$
$F_4 = x'y$	$y/x$	Inhibition	$y$ , but not $x$
$F_5 = y$		Transfer	$y$
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	$x$ or $y$ , but not both
$F_7 = x + y$	$x + y$	OR	$x$ or $y$
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	$x$ equals $y$
$F_{10} = y'$	$y'$	Complement	Not $y$
$F_{11} = x + y'$	$x \subset y$	Implication	If $y$ , then $x$
$F_{12} = x'$	$x'$	Complement	Not $x$
$F_{13} = x' + y$	$x \supset y$	Implication	If $x$ , then $y$
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

# Karnaugh map or K-map

- ▶ Karnaugh map or K-map
- ▶ The map method presented here provides a simple, straightforward procedure for minimizing Boolean functions.
- ▶ This method may be regarded as a pictorial form of a truth table.

# Karnaugh map or K-map

- ▶ Karnaugh map or K-map
- ▶ The map method presented here provides a simple, straightforward procedure for minimizing Boolean functions.
- ▶ This method may be regarded as a pictorial form of a truth table.

## Two-Variable K-Map

$m_0$	$m_1$	$x'y'$	$x'y$
$m_2$	$m_3$	$xy'$	$xy$

# K-Map

Consider the function

$$F = m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

# K-Map

Consider the function

$$F = m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

		$y$	
		0	1
$x$	0	$m_0$ $x'y'$	$m_1$ $x'y$
	1	$m_2$ $xy'$	$m_3$ $xy$

# Three-Variable K-Map

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$

$x \backslash yz$		$y$			
		00	01	11	10
$x$	0	$m_0$ $x'y'z'$	$m_1$ $x'y'z$	$m_3$ $x'yz$	$m_2$ $x'yz'$
	1	$m_4$ $xy'z'$	$m_5$ $xy'z$	$m_7$ $xyz$	$m_6$ $xyz'$
		$z$			

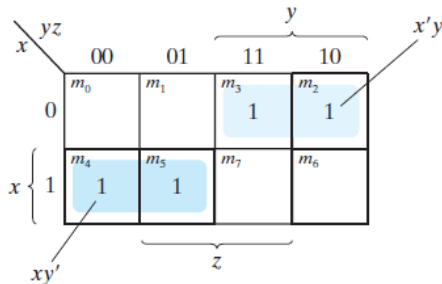


## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(2, 3, 4, 5)$

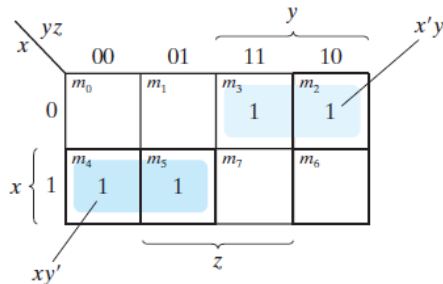
## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(2, 3, 4, 5)$



## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(2, 3, 4, 5)$



$$F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$$

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

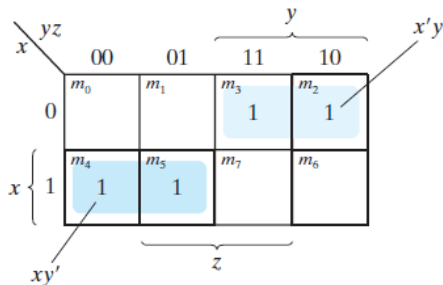
Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

22.09.2020

## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(2, 3, 4, 5)$



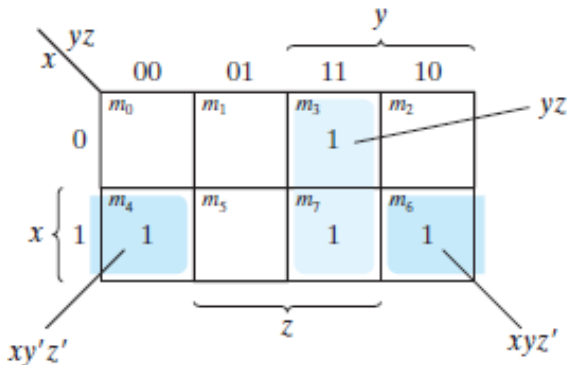
$$F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$$

## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(3, 4, 6, 7)$

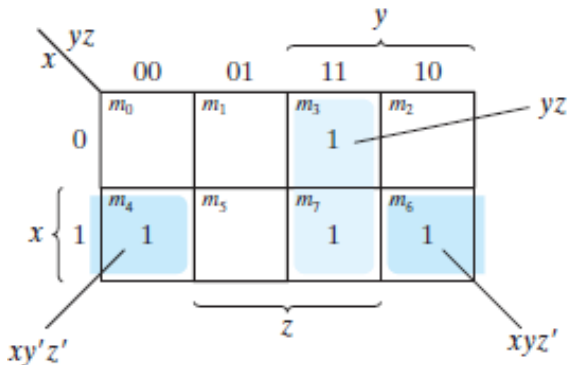
## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(3, 4, 6, 7)$



## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(3, 4, 6, 7)$



$$F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$$

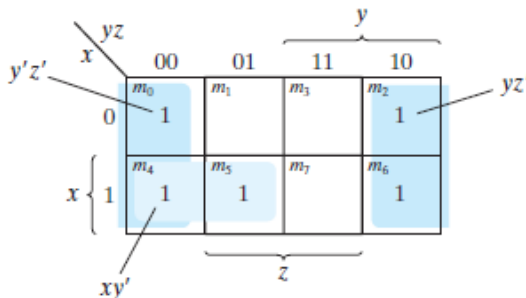


## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$

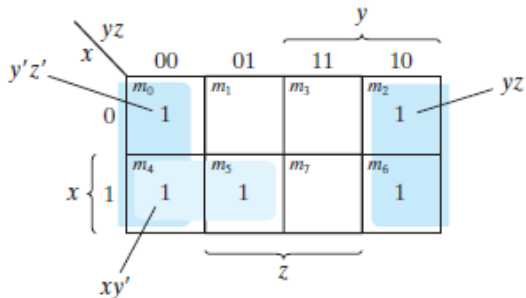
## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$



## Three-Variable K-Map

Simplify the Boolean function  $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$



$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$$

## Three-Variable K-Map

Simplify the Boolean function  $F = A'C + A'B + AB'C + BC$

## Three-Variable K-Map

Simplify the Boolean function  $F = A'C + A'B + AB'C + BC$

Express this function as a sum of minterms.

Find the minimal sum-of-products expression.

## Three-Variable K-Map

Simplify the Boolean function  $F = A'C + A'B + AB'C + BC$

Express this function as a sum of minterms.

Find the minimal sum-of-products expression.

$$F = A'C + A'B + AB'C + BC$$

$$= A'BC + A'B'C + A'BC' + AB'C + ABC$$

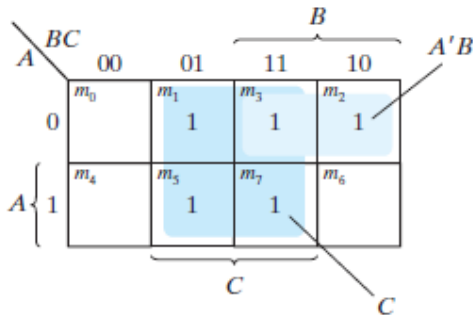
$$F = \Sigma(1, 2, 3, 5, 7)$$

## Three-Variable K-Map

Simplify the Boolean function  $F = \Sigma(1, 2, 3, 5, 7)$

## Three-Variable K-Map

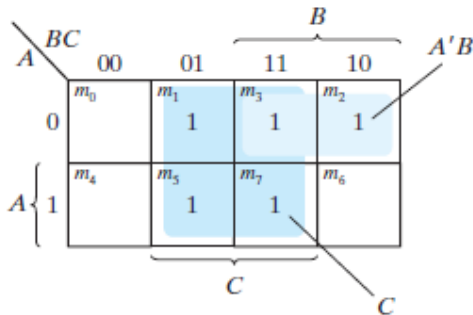
Simplify the Boolean function  $F = \Sigma(1, 2, 3, 5, 7)$





## Three-Variable K-Map

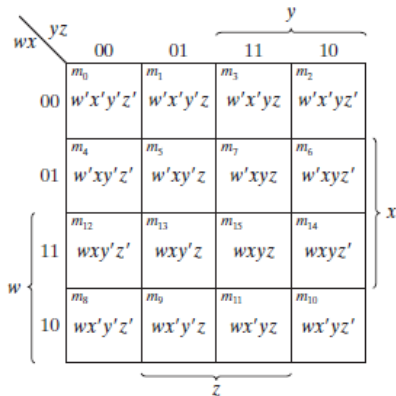
Simplify the Boolean function  $F = \Sigma(1, 2, 3, 5, 7)$



$$F = A'C + A'B + AB'C + BC = \Sigma(1, 2, 3, 5, 7) = C + A'B$$

# FOUR-VARIABLE K-MAP

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$
$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$m_8$	$m_9$	$m_{11}$	$m_{10}$

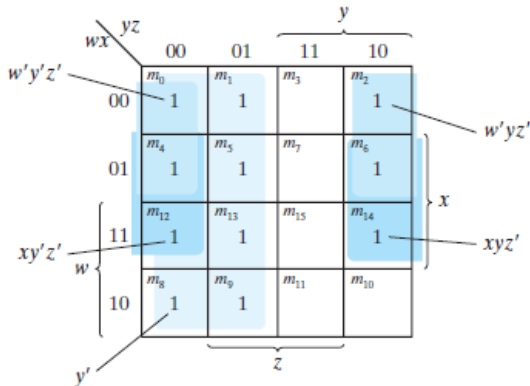


# FOUR-VARIABLE K-MAP

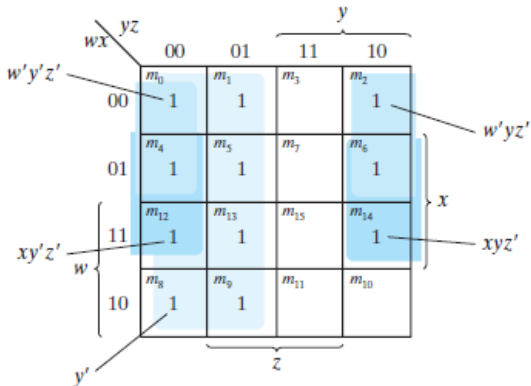
Simplify the Boolean function

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$



$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$



$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$$

# FOUR-VARIABLE K-MAP

Simplify the Boolean function

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

		$yz$			
		00	01	11	10
$wx$	00	$m_0$ 1	$m_1$ 1	$m_3$	$m_2$
	01	$m_4$	$m_5$	$m_7$	$m_6$
	11	$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
	10	$m_8$	$m_9$	$m_{11}$	$m_{10}$

$$F = A'B'C'$$

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

		yz			
		00	01	11	10
wx	00	$m_0$	$m_1$	$m_3$	$m_2$ 1
	01	$m_4$	$m_5$	$m_7$	$m_6$
	11	$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
	10	$m_8$	$m_9$	$m_{11}$	$m_{10}$ 1

$$F = B'CD'$$

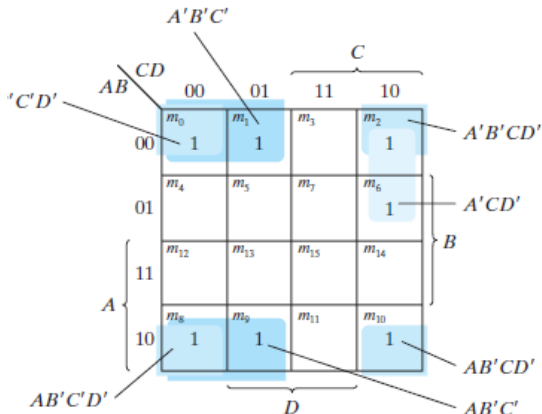


$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

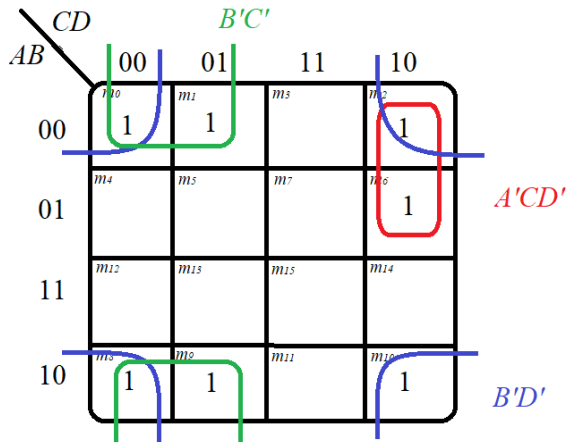
		yz			
		00	01	11	10
wx	00	$m_0$	$m_1$	$m_3$	$m_2$
	01	$m_4$	$m_5$	$m_7$	$m_6$
	11	$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
	10	$m_8$ 1	$m_9$ 1	$m_{11}$	$m_{10}$

$$F = AB'C'$$

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$



$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$



$$F = B'C' + B'D' + A'CD'$$

## Prime Implicants

- ▶ In choosing adjacent squares in a map, we must ensure that
  1. all the minterms of the function are covered when we combine the squares,
  2. the number of terms in the expression is minimized, and
  3. there are no redundant terms (i.e., minterms already covered by other terms).

## Prime Implicants

- ▶ In choosing adjacent squares in a map, we must ensure that
  1. all the minterms of the function are covered when we combine the squares,
  2. the number of terms in the expression is minimized, and
  3. there are no redundant terms (i.e., minterms already covered by other terms).
- ▶ A **prime implicant** is a product term obtained by combining the maximum possible number of adjacent squares in the map.

## Prime Implicants

- ▶ In choosing adjacent squares in a map, we must ensure that
  1. all the minterms of the function are covered when we combine the squares,
  2. the number of terms in the expression is minimized, and
  3. there are no redundant terms (i.e., minterms already covered by other terms).
- ▶ A **prime implicant** is a product term obtained by combining the maximum possible number of adjacent squares in the map.
- ▶ If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be **essential**.

Consider the Boolean function

$$F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

		<i>CD</i>			
		00	01	11	10
<i>AB</i>	00	<i>m</i> <sub>0</sub> 1	<i>m</i> <sub>1</sub>	<i>m</i> <sub>3</sub> 1	<i>m</i> <sub>2</sub> 1
	01	<i>m</i> <sub>4</sub>	<i>m</i> <sub>5</sub> 1	<i>m</i> <sub>7</sub> 1	<i>m</i> <sub>6</sub>
	11	<i>m</i> <sub>12</sub>	<i>m</i> <sub>13</sub> 1	<i>m</i> <sub>15</sub> 1	<i>m</i> <sub>14</sub>
	10	<i>m</i> <sub>8</sub> 1	<i>m</i> <sub>9</sub> 1	<i>m</i> <sub>11</sub> 1	<i>m</i> <sub>10</sub> 1

# Prime Implicants

$AB \backslash CD$					
		00	01	11	10
00	$m_0$	1		1	1
01	$m_4$		1	1	
11	$m_{12}$		1	1	
10	$m_8$	1	1	1	1

►  $CD$



# Prime Implicants

		$CD$			
		00	01	11	10
$AB$	00	$m_0$ 1	$m_1$	$m_3$ 1	$m_2$ 1
	01	$m_4$	$m_5$ 1	$m_7$ 1	$m_6$
	11	$m_{12}$	$m_{13}$ 1	$m_{15}$ 1	$m_{14}$
	10	$m_8$ 1	$m_9$ 1	$m_{11}$ 1	$m_{10}$ 1

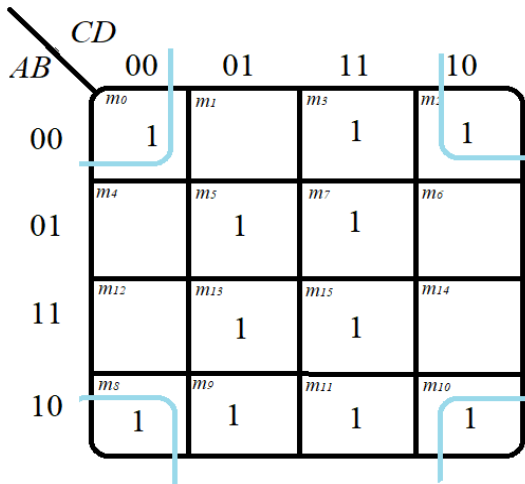
$CD$   
▶  $AB'$

# Prime Implicants

$AB \backslash CD$					
		00	01	11	10
00	$m_0$	1		1	1
01	$m_4$		1	1	
11	$m_{12}$		1	1	
10	$m_8$	1	1	1	1

$CD$   
 $AB'$   
▶  $BD$

# Prime Implicants



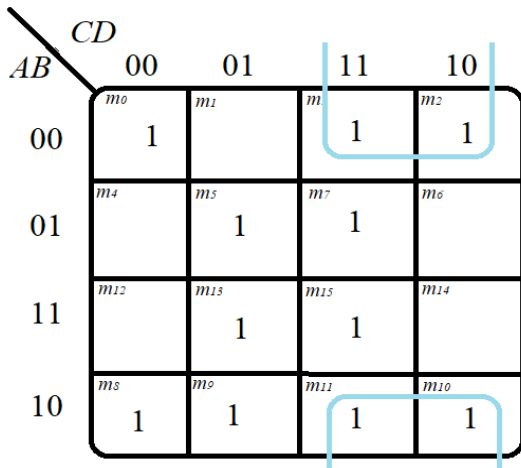
$CD$   
 $AB'$   
 $BD$   
 $\blacktriangleright B'D'$

# Prime Implicants

		$CD$			
$AB$		00	01	11	10
00	$m_0$	1		1	1
01	$m_4$		1	1	
11	$m_{12}$		1	1	
10	$m_8$	1	1	1	1

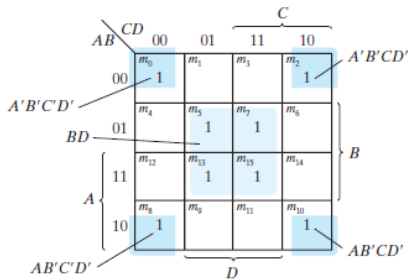
$CD$   
 $AB'$   
 $BD$   
 $B'D'$   
 ►  $AD$

# Prime Implicants

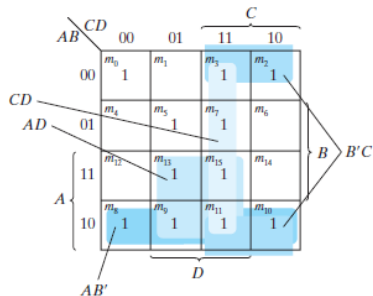


$CD$   
 $AB'$   
 $BD$   
 $B'D'$   
 $AD$   
 ►  $B'C$

# Prime Implicants



(a) Essential prime implicants  
 $BD$  and  $B'D'$



(b) Prime implicants  $CD$ ,  $B'C$ ,  
 $AD$ , and  $AB'$

# Prime Implicants

*BD*

*B'D'*

*CD*

*AB'*

*AD*

*B'C*

$$\begin{aligned} F &= BD + B'D' + CD + AD \\ &= BD + B'D' + CD + AB' \\ &= BD + B'D' + B'C + AD \\ &= BD + B'D' + B'C + AB' \end{aligned}$$

## Don't Care Conditions

- Functions that have unspecified outputs for some input combinations are called *incompletely specified functions*



## Don't Care Conditions

- ▶ Functions that have unspecified outputs for some input combinations are called *incompletely specified functions*
- ▶ it is customary to call the unspecified minterms of a function **don't-care conditions**

## Don't Care Conditions

- ▶ Functions that have unspecified outputs for some input combinations are called *incompletely specified functions*
- ▶ it is customary to call the unspecified minterms of a function **don't-care conditions**
- ▶ A don't-care minterm is a combination of variables whose logical value is not specified.
- ▶ Such a minterm cannot be marked with a 1 in the map, because it would require that the function always be a 1 for such a combination.
- ▶ Likewise, putting a 0 on the square requires the function to be 0. To distinguish the don't-care condition from 1's and 0's, an X is used.
- ▶ Thus, an X inside a square in the map indicates that we don't care whether the value of 0 or 1 is assigned

## An example

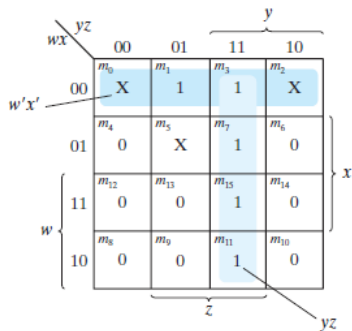
Simplify the Boolean function

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

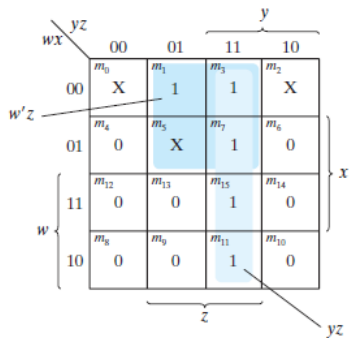
which has the don't-care conditions

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$

# An example



(a)  $F = yz + w'x'$



(b)  $F = yz + w'z$

# Digital System Design

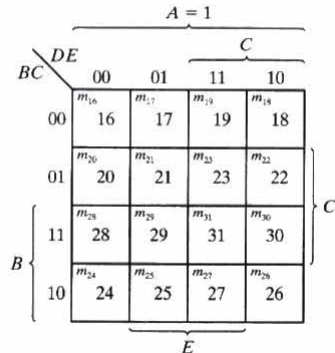
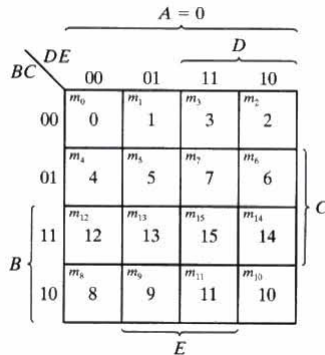
## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

23.09.2020

# Five-Variable K-Map



## Simplify the Boolean function

Figure 1 shows two Karnaugh maps for the function  $F(A, B, C, D)$ . The left map is for  $A=0$  and the right map is for  $A=1$ . Both maps show prime implicants as red circles.

**Map for  $A=0$ :**

	$D'E'$	$D'E$	$DE$	$DE'$
$B'C'$	1			1
$B'C$	1			1
$BC$		1		
$BC'$		1		

**Map for  $A=1$ :**

	$D'E'$	$D'E$	$DE$	$DE'$
$B'C'$				
$B'C$		1	1	
$BC$		1	1	
$BC'$		1		

►  $A'B'E'$

## Five-variable K-Map

Simplify the Boolean function

$$F(A, B, C, D, E) = \Sigma(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$$

	D'E'	D'E	DE	DE'		D'E'	D'E	DE	DE'
B'C'	0 1	1	3	2 1	B'C'	16	17	19	18
B'C	4 1	5	7	6 1	B'C	20	21 1	22 1	22
BC	12	13 1	15	14	BC	28	29 1	30 1	30
BC'	8	9 1	11	10	BC'	24	25 1	27	26
A=0					A=1				

►  $A'B'E'$   
 $ACE$



## Five-variable K-Map

Simplify the Boolean function

$$F(A, B, C, D, E) = \Sigma(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$$

	D'E'	D'E	DE	DE'		D'E'	D'E	DE	DE'
B'C'	0 1	1	3	2 1	B'C'	16	17	19	18
B'C	4 1	5	7	6 1	B'C	20	21 1	23 1	22
BC	12	13 1	15	14	BC	28	29 1	31 1	30
BC'	8	9 1	11	10	BC'	24	25 1	27	26
	A=0					A=1			

$$A'B'E'$$

$$ACE$$

►  $BD'E$

$$F(A, B, C, D, E) = \Sigma(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$$
$$= A'B'E' + ACE + BD'E$$

# Quine–McCluskey algorithm

## K-Map Pros and Cons

- ▶ K-Map is systemic
- ▶ Require the ability to identify and visualize the prime implicants in order to cover all minterms
- ▶ But effective only up to 5-6 input variables!

## Quine-McCluskey Algorithm

- ▶ Tabular Method
  - ▶ Compute all prime implicants
  - ▶ Find a minimum expression for Boolean functions
- ▶ No visualization of prime implicants
- ▶ Can be programmed and implemented in a computer

## QM Method Example

$$F(W, X, Y, Z) = \Sigma_m(0, 3, 5, 6, 7, 10, 12, 13) + \Sigma_d(2, 9, 15)$$

FOR MINTERMS

Minterm	w	x	y	z
0	0	0	0	0
3	0	0	1	1
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
10	1	0	1	0
12	1	1	0	0
13	1	1	0	1

FOR DON'T CARES:

Minterm	w	x	y	z
2	0	0	1	0
9	1	0	0	1
15	1	1	1	1

Step 1 : Divide all the minterms (and don't cares) of a function into groups

Step 1 : Divide all the minterms (and don't cares) of a function into groups

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0	0	0	0	0	0	
G1	2	0	0	1	0	
G2	3	0	0	1	1	
	5	0	1	0	1	
	6	0	1	1	0	
	9	1	0	0	1	
	10	1	0	1	0	
	12	1	1	0	0	
G3	7	0	1	1	1	
	13	1	1	0	1	
G4	15	1	1	1	1	

Step 2: Merge minterms from adjacent groups to form a new implicant table

Step 2: Merge minterms from adjacent groups to form a new implicant table









Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0	0	0	0	0	0	
G1	2	0	0	1	0	
G2	3	0	0	1	1	
	5	0	1	0	1	
	6	0	1	1	0	
	9	1	0	0	1	
	10	1	0	1	0	
G3	12	1	1	0	0	
	7	0	1	1	1	
G4	13	1	1	0	1	
	15	1	1	1	1	

Groups	Minterm ID	W	X	Y	Z
G0'	0, 2	0	0	d	0
G1'	2, 3	0	0	1	d
	2, 6	0	d	1	0
	2, 10	d	0	1	0
G2'	3, 7	0	d	1	1
	5, 7	0	1	d	1
	6, 7	0	1	1	d
	5, 13	d	1	0	1
	9, 13	1	d	0	1
	12, 13	1	1	0	d
	12, 13	1	1	0	d
G3'	7, 15	d	1	1	1
	13, 15	1	1	d	1



Step 3: Repeat step 2 until no more merging is possible

Step 3: Repeat step 2 until no more merging is possible

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0'	0, 2	0	0	d	0	
G1'	2, 3	0	0	1	d	
	2, 6	0	d	1	0	
	2, 10	d	0	1	0	
G2'	3, 7	0	d	1	1	
	5, 7	0	1	d	1	
	6, 7	0	1	1	d	
	5, 13	d	1	0	1	
	9, 13	1	d	0	1	
	12, 13	1	1	0	d	
G3'	7, 15	d	1	1	1	
	13, 15	1	1	d	1	

Groups	Minterm ID	W	X	Y	Z
G1''	2, 3, 6, 7	0	d	1	d
	2, 6, 3, 7	0	d	1	d
G2''	5, 7, 13, 15	d	1	d	1
	5, 7, 13, 15	d	1	d	1

No more merging is possible.

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0"	0, 2	0	0	d	0	
G1"	2, 3, 6, 7	0	d	1	d	
	2, 10	d	0	1	0	
G2"	5, 7, 13, 15	d	1	d	1	
	9, 13	1	d	0	1	
	12, 13	1	1	0	d	

Step 4: Put all prime implicants in a cover table (don't cares excluded) Step 5: Identify essential minterms, and hence essential prime implicants

	0	3	5	6	7	10	12	13
$w'x'z'$	X							
$w'y$		X		X	X			
$x'yz'$						X		
$xz$			X		X			X
$wy'z$								X
$wxy'$							X	X

Step 6: Add prime implicants to the minimum expression of  $F$  until all minterms of  $F$  are covered

	0	3	5	6	7	10	12	13
$w'x'z'$	X							
$w'y$		X		X	X			
$x'yz'$						X		
$xz$			X		X			X
$wy'z$								X
$wxy'$							X	X

$$F = w'x'y' + w'y + x'yz' + xz + wxy'$$

## Another Example

F(W,X,Y,Z) = $\Sigma(2,3,6,7,8,10,11,12,14,15)$											
Step 1											
	2	0010									
	8	1000									
	3	0011									
	6	0110									
	10	1010									
	12	1100									
	7	0111									
	11	1011									
	14	1110									
	15	1111									

F(W,X,Y,Z) = $\Sigma(2,3,6,7,8,10,11,12,14,15)$										
Step 1			Step 2							
✓	2	0010		2,3	001-					
✓	8	1000		2,6	0-10					
				2,10	-010					
✓	3	0011		8,10	10-0					
✓	6	0110		8,12	1-00					
✓	10	1010								
✓	12	1100		3,7	0-11					
				3,11	-011					
✓	7	0111		6,7	011-					
✓	11	1011		6,14	-110					
✓	14	1110		10,11	101-					
				10,14	1-10					
✓	15	1111		12,14	11-0					
				7,15	-111					
				11,15	1-11					
				14,15	111-					



F(W,X,Y,Z) = $\Sigma(2,3,6,7,8,10,11,12,14,15)$											
Step 1			Step 2			Step 3			Step 4		
✓	2	0010	✓	2,3	001-		2,3,6,7	0-1-			
✓	8	1000	✓	2,6	0-10		2,3,10,11	-01-			
			✓	2,10	-010		2,6,3,7	0-1-			
✓	3	0011	✓	8,10	10-0		2,6,10,14	--10			
✓	6	0110	✓	8,12	1-00		2,10,3,11	-01-			
✓	10	1010					2,10,6,14	--10			
✓	12	1100	✓	3,7	0-11		8,10,12,14	1--0			
			✓	3,11	-011		8,12,10,14	1--0			
✓	7	0111	✓	6,7	011-						
✓	11	1011	✓	6,14	-110		3,7,11,15	--11			
✓	14	1110	✓	10,11	101-		3,11,7,15	--11			
			✓	10,14	1-10		6,7,14,15	-11-			
✓	15	1111	✓	12,14	11-0		6,14,7,15	-11-			
							10,14,11,15	1-1-			
			✓	7,15	-111		10,11,14,15	1-1-			
			✓	11,15	1-11						
			✓	14,15	111-						

F(W,X,Y,Z) = $\Sigma(2,3,6,7,8,10,11,12,14,15)$											
Step 1			Step 2			Step 3			Step 4		
✓	2	0010	✓	2,3	001-	✓	2,3,6,7	0 - 1 -	2,3,6,7,10,14,11,15	--1-	
✓	8	1000	✓	2,6	0-10	✓	2,3,10,11	-01-	2,3,10,11,6,14,7,15	--1-	
			✓	2,10	-010	✓	2,6,3,7	0-1-	2,6,3,7,10,11,14,15	--1-	
✓	3	0011	✓	8,10	10-0	✓	2,6,10,14	--10	2,6,10,14,3,7,11,15	--1-	
✓	6	0110	✓	8,12	1-00	✓	2,10,3,11	-01-	2,10,3,11,6,7,14,15	--1-	
✓	10	1010				✓	2,10,6,14	--10	2,10,6,14,3,11,7,15	--1-	
✓	12	1100	✓	3,7	0-11		8,10,12,14	1 - - 0			
			✓	3,11	-011		8,12,10,14	1 - - 0			
✓	7	0111	✓	6,7	011-						
✓	11	1011	✓	6,14	-110	✓	3,7,11,15	-- 11			
✓	14	1110	✓	10,11	101-	✓	3,11,7,15	-- 11			
			✓	10,14	1-10	✓	6,7,14,15	- 11 -			
✓	15	1111	✓	12,14	11-0	✓	6,14,7,15	- 11 -			
						✓	10,14,11,15	1 - 1 -			
			✓	7,15	-111	✓	10,11,14,15	1 - 1 -			
			✓	11,15	1-11						
			✓	14,15	111-						

Step 5: Put all prime implicants in a cover table (don't cares excluded)

Step 6: Identify essential minterms, and hence essential prime implicants

Step 7: Add prime implicants to the minimum expression of  $F$  until all minterms of  $F$  are covered

	2	3	6	7	8	10	11	12	14	15
$y$	X	X	X	X		X	X		X	X
$wz'$					X	X		X	X	

	2	3	6	7	8	10	11	12	14	15
$y$	X	X	X	X		X	X		X	X
$wz'$					X	X		X	X	

$$F(w, x, y, z) = \Sigma(2, 3, 6, 7, 8, 10, 11, 12, 14, 15) = y + wx'$$

# Digital System Design

## Module 2 - BOOLEAN ALGEBRA & LOGIC GATES

Dr. Deepthi Sasidharan

Assistant Professor, Department of Information Technology  
GEC Barton Hill, Thiruvananthapuram

24.09.2020

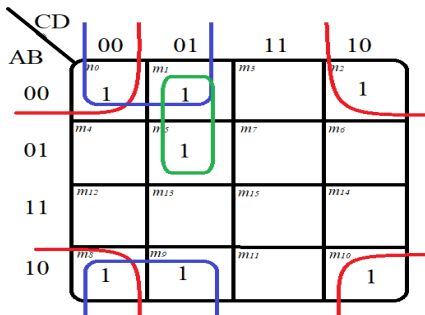
## Five-Variable K-Map

- ▶ The procedure for obtaining a minimized function in product-of-sums form follows from the basic properties of Boolean functions.
- ▶ The minterms not included in the standard sum-of-products form of a function denote the complement of the function.
- ▶ If we mark the empty squares by 0's and combine them into valid adjacent squares, we obtain a simplified sum-of-products expression of the complement of the function (i.e., of  $F'$ ).
- ▶ The complement of  $F'$  gives us back the function  $F$  in product-of-sums form (a consequence of DeMorgan's theorem).

# Product of Sums Simplification

Simplify the following Boolean function into (a) sum-of-products form and (b) product-of-sums form:

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$



$$B'D'$$

$$B'C'$$

$$A'C'D$$

$$F = B'D' + B'C' + A'C'D$$

# Product of Sums Simplification

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

		CD			
		00	01	11	10
AB	00	$m_0$ 1	$m_1$ 1	$m_5$ 0	$m_4$ 1
	01	$m_4$ 0	$m_5$ 1	$m_1$ 0	$m_0$ 0
	11	$m_{12}$ 0	$m_{13}$ 0	$m_9$ 0	$m_{14}$ 0
	10	$m_8$ 1	$m_9$ 1	$m_{11}$ 0	$m_{10}$ 1

►  $(C' + D')$

# Product of Sums Simplification

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

		CD			
		00	01	11	10
AB	00	$m_0$ 1	$m_1$ 1	$m_5$ 0	$m_4$ 1
	01	$m_4$ 0	$m_5$ 1	$m_1$ 0	$m_0$ 0
	11	$m_{12}$ 0	$m_{13}$ 0	$m_9$ 0	$m_{14}$ 0
	10	$m_8$ 1	$m_9$ 1	$m_{11}$ 0	$m_{10}$ 1

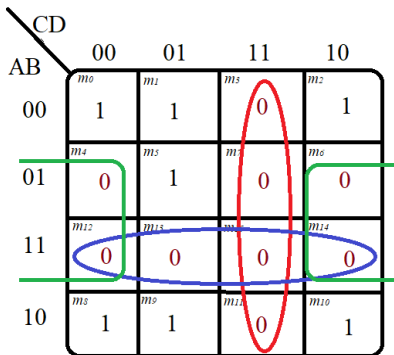
$$(C' + D')$$

$$\blacktriangleright (A' + B')$$



# Product of Sums Simplification

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$



$$(C' + D')$$

$$(A' + B')$$

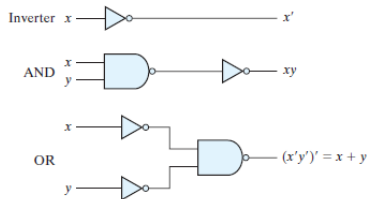
$$(B' + D)$$

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

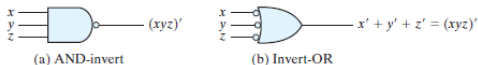
## NAND Circuits

- ▶ NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.
- ▶ Because of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from Boolean functions given in terms of AND, OR, and NOT into equivalent NAND and NOR logic diagrams.
- ▶ The **NAND** gate is said to be a **universal gate** because any logic circuit can be implemented with it.
- ▶ A convenient way to implement a Boolean function with NAND gates is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND logic.

# NAND Circuits



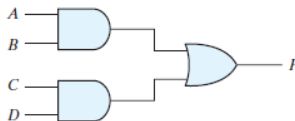
## Logic operations with NAND gates



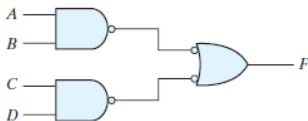
## Two graphic symbols for a three-input NAND gate

## Two-Level Implementation

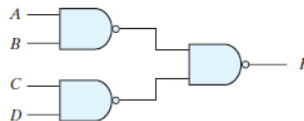
The implementation of Boolean functions with NAND gates requires that the functions be in **sum-of-products form**.



(a)



(b)

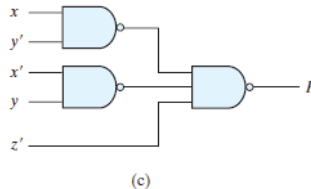
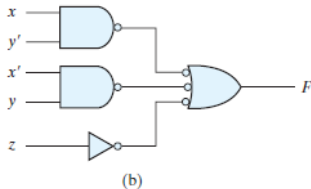
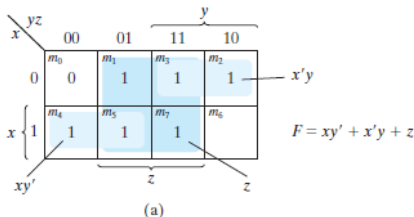


(c)

Three ways to implement  $F = AB + CD$

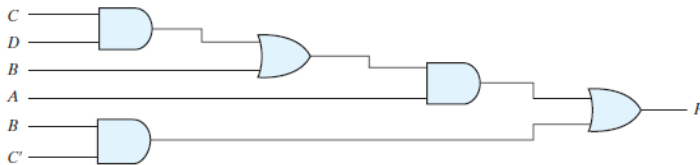
## An Example

Implement the following Boolean function with NAND gates:  $F(x, y, z) = \Sigma(1, 2, 3, 4, 5, 7)$

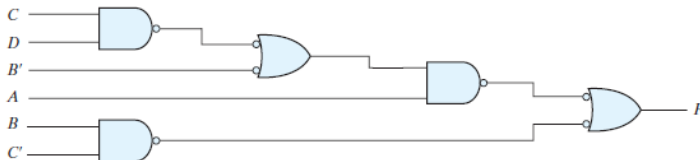


## Multilevel NAND Circuits

Consider the Boolean function  $F = A(CD + B) + BC'$



(a) AND-OR gates



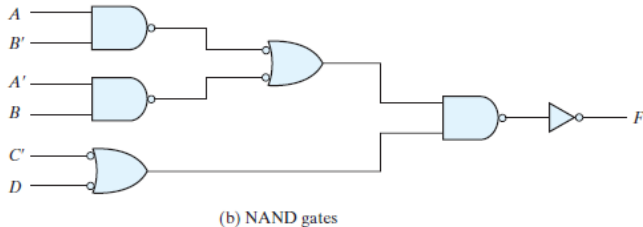
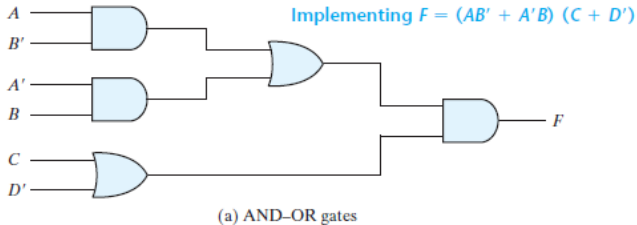
(b) NAND gates

## Multilevel NAND Circuits

The general procedure for converting a multilevel AND–OR diagram into an all-NAND diagram using mixed notation is as follows:

1. Convert all AND gates to NAND gates with AND-invert graphic symbols.
2. Convert all OR gates to NAND gates with invert-OR graphic symbols.
3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NAND gate) or complement the input literal.

# Another Example

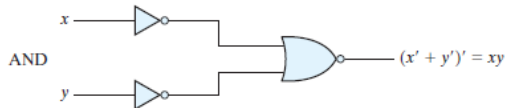




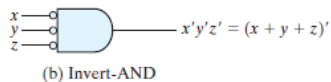
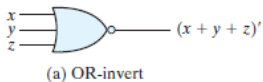
# NOR Implementation

- ▶ The NOR operation is the dual of the NAND operation.
- ▶ Therefore, all procedures and rules for NOR logic are the duals of the corresponding procedures and rules developed for NAND logic.
- ▶ The NOR gate is another universal gate that can be used to implement any Boolean function.
- ▶ A two-level implementation with NOR gates requires that the function be simplified into product-of-sums form.

# NOR Implementation

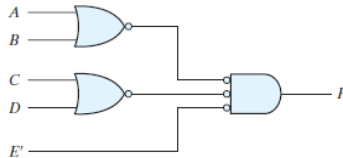


## Logic operations with NOR gates

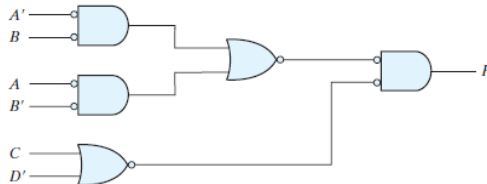


## Two graphic symbols for the NOR gate

# An Example



Implementing  $F = (A + B)(C + D)E$



Implementing  $F = (AB' + A'B)(C + D')$  with NOR gates