



Queen Mary

University of London

Science and Engineering

**EBU4202: Digital Circuit Design**  
**Basic Logic Functions and Switching Algebra**

Dr. Md Hasanuzzaman Sagor (Hasan)

Dr. Chao Shu (Chao)

Dr. Farha Lakhani (Farha)

School of Electronic Engineering and Computer Science,  
Queen Mary University of London,  
London, United Kingdom.

# Overview: Switching Algebra & Combinational Logic Design

---

- \* **Switching Algebra**

- \* **Combinational Circuit Analysis & Synthesis**

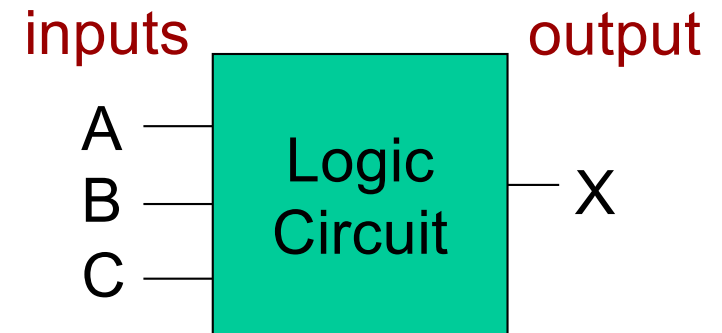


**Chapters 4 & 6** – “Digital Design: Principles and Practices” book

# Combinational Logic Circuits: Analysis & Synthesis

- **Digital Circuits:**

- **Combinational Logic Circuit:** a circuit whose outputs depend on its current inputs.
- **Sequential Logic Circuit:** a circuit whose outputs depend not only on current inputs, but also on past inputs.



- These slides are concerned with the **analysis** and **synthesis** of **combinational** logical **circuits**.



**Studied later**  
in the course.

- **Analysis** → start from a logic diagram of a circuit and derive a formal description of the function of the circuit.
- **Synthesis** → start with a formal description of the function of a circuit and proceed to a logic diagram that performs the required function.

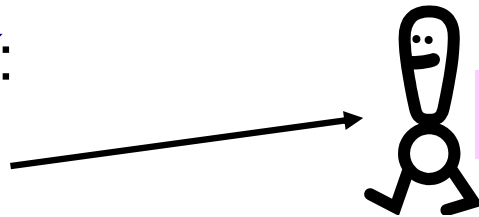
# Switching Algebra

- Switching Algebra:

- Two-valued boolean algebra (George Boole 1815-64)

- For a variable  $X$ :

- $X=0$  or  $X=1$



No other values are allowed.

- Many other possible “boolean” representations:

- $X = \text{Light on}$  or  $X = \text{Light off}$

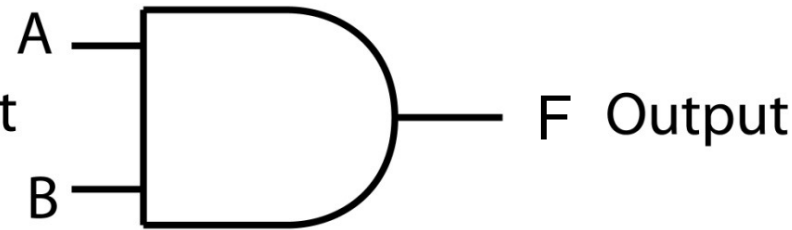
- $X = \text{voltage HIGH}$  or  $X = \text{voltage LOW}$

- *Basic operations* are AND, OR and complement (or *invert*) (AND/OR/NOT).

# Basic Gates (1/3): AND

- Performs the **AND logic operation** on its **inputs** and **outputs** its result.

Symbol →

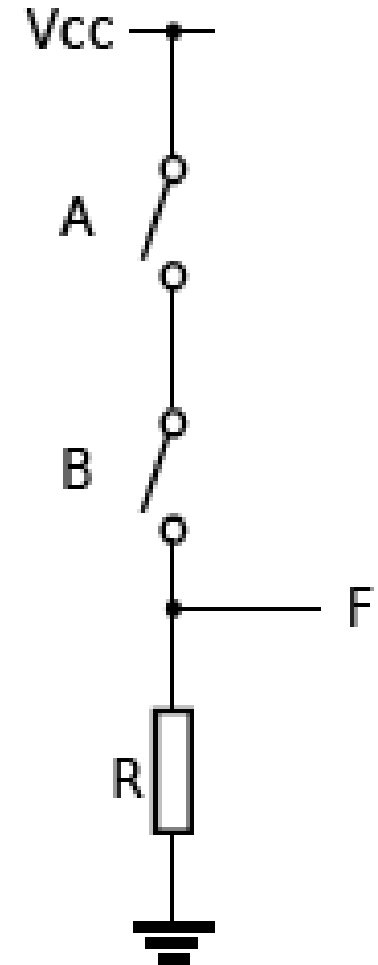


Truth table →

Input		Output
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Binary ascending  
counting order.

Circuit  
Switching  
Diagram →

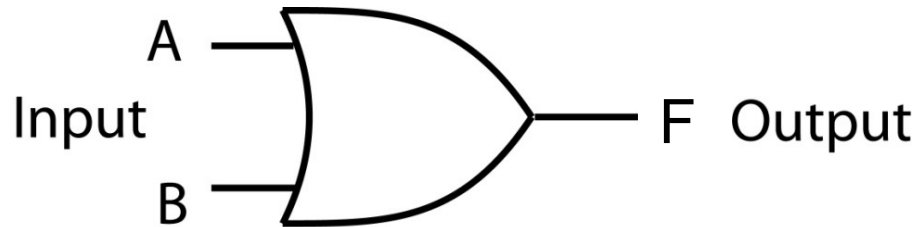


Switching Algebra Equation →  $F = A.B$  or  $F = AB$

# Basic Gates (2/3): OR

- Performs the **OR logic operation** on its *inputs* and *outputs* its result.

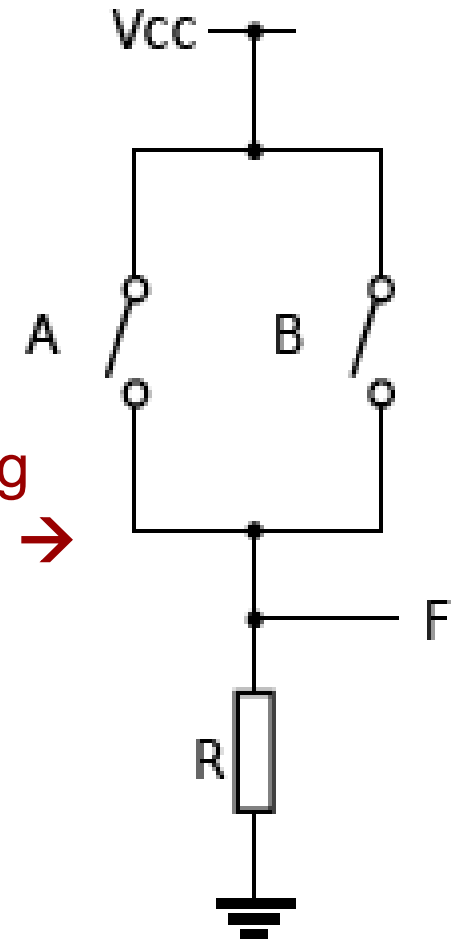
Symbol →



Truth table →

Input		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Circuit Switching Diagram →



Switching Algebra Equation →  $F = A + B$

# Basic Gates (3/3): NOT



Other notation (not used here):  $F = \sim A$  or  $F = \neg A$ .

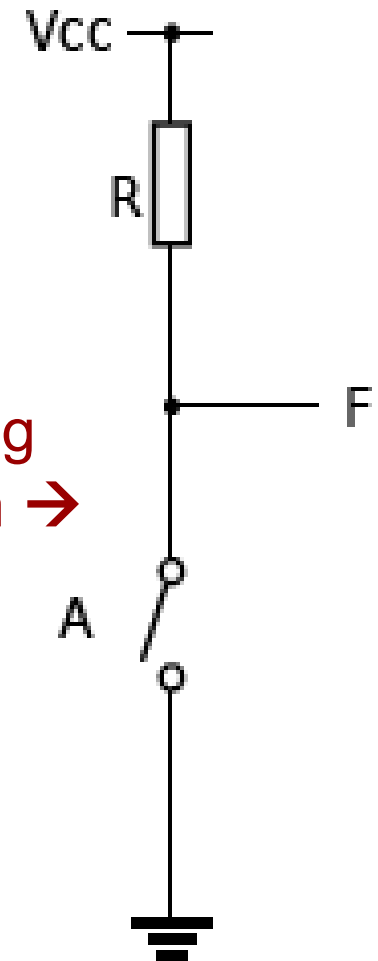
- Also called an **inverter**, it produces an **output** value that is the opposite of the **input** value.



Truth table →

Input	Output
A	F
0	1
1	0

Circuit Switching Diagram →



Switching Algebra Equation →  $F = A'$  or  $F = \overline{A}$

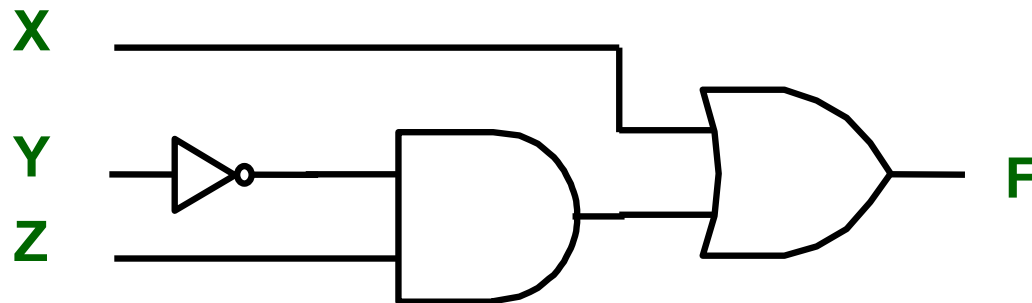
# Example: Combining Basic Gates

- Any *Boolean function* can be represented with a Truth Table.

**Switching Algebra**

$$F(X,Y,Z) = X + \overline{Y}.Z$$

**Logic Gate Diagram**



**Truth Table**

X	Y	Z	$F = X + \overline{Y}.Z$
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



# More Gates

- There are **more gates available for the designer** to use than those discussed so far.
- Of particular importance to the electronics industry are the **Buffer**, **NAND**, **NOR**, **Exclusive-OR** and **Exclusive-NOR** gates.
  - **NAND** and **NOR** gates are particularly important because they **are faster than AND** and **OR** gates.



Any digital circuit of any complexity can be made from **NAND** gates.

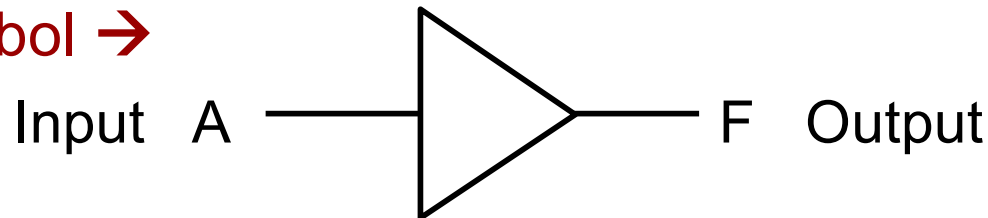


**Why** do you think that is the case?.....to answer this you need to look at the circuits that are inside the gates.....

# Other Gates (1/5): Buffer

- Also called a **non-inverting buffer**, it asserts its **output** signal if and only if its **input** is asserted.

Symbol →

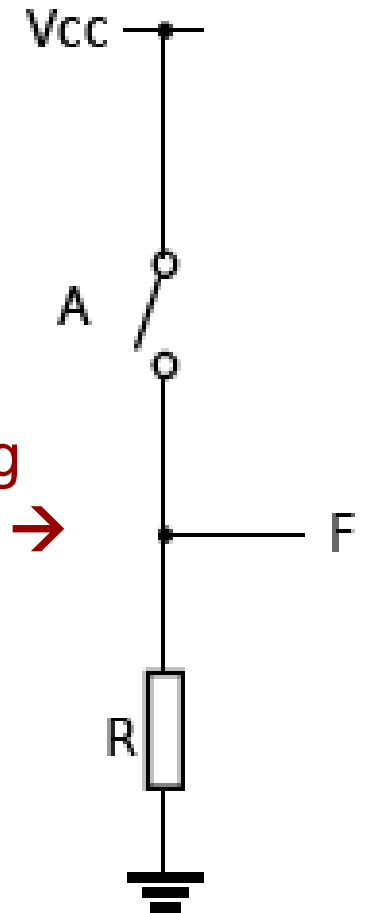


Truth table →

A	F
0	0
1	1

Switching Algebra Equation →  $F = A$

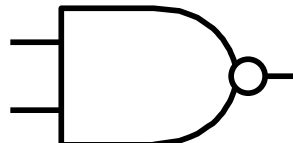
Circuit Switching Diagram →



NOTE: You might ask the question “Why do we need this gate, it does not seem to do anything”. It has important input and output characteristics that make it useful in practical circuits.

# Other Gates (2/5): NAND

- It does the *opposite of* an **AND** gate.

Symbol → Inputs A B  Output F

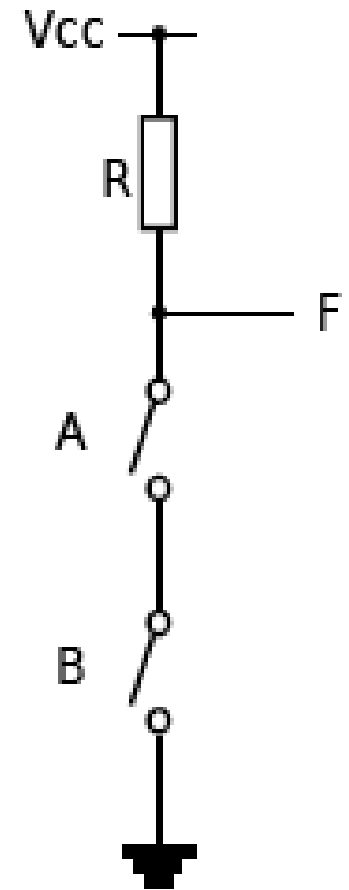
Truth table →

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Switching Algebra Equation →

$$F = (A.B)'$$
$$F = (AB)'$$
$$F = (\overline{A.B})$$
$$F = (\overline{AB})$$

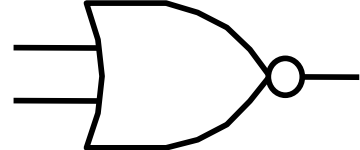
Circuit Switching Diagram →



# Other Gates (3/5): NOR

- It does the *opposite of* an OR gate.

Symbol → Inputs A B Output F

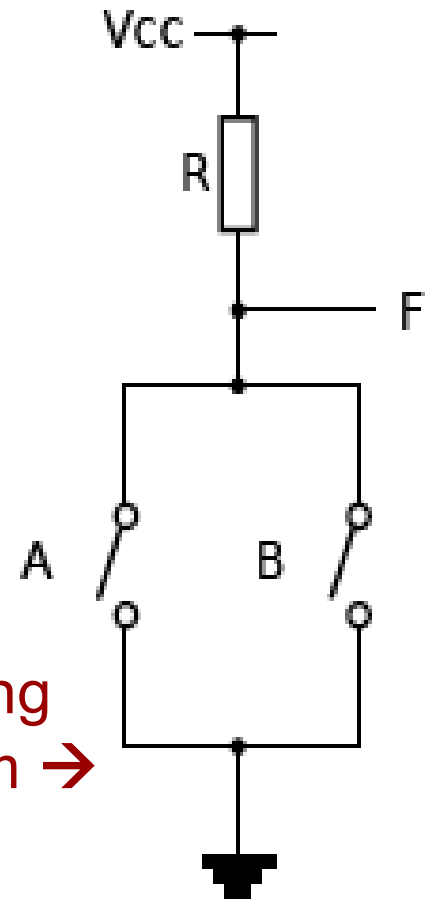


Truth table →

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

Switching Algebra Equation →  $F = (A+B)'$  or  $F = \overline{(A+B)}$

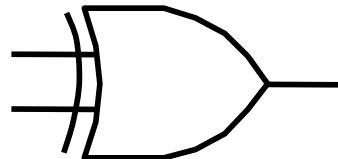
Circuit Switching Diagram →



# Other Gates (4/5): XOR

- Its *output* is *high*, only when the *inputs are different*.

Symbol → Inputs A B Output F

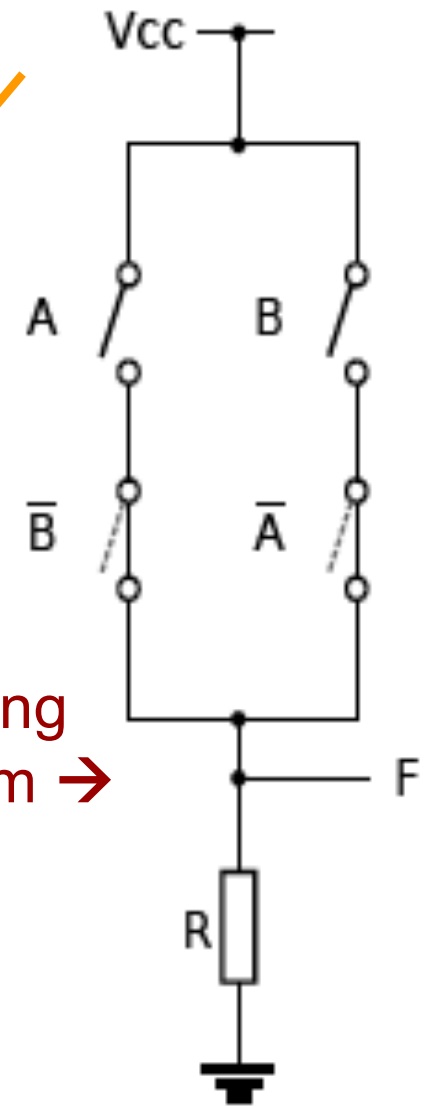


Truth table →

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Switching Algebra Equation →  $F = A \oplus B$  or  
 $F = A'B + AB'$  or  
 $F = \bar{A}B + A\bar{B}$

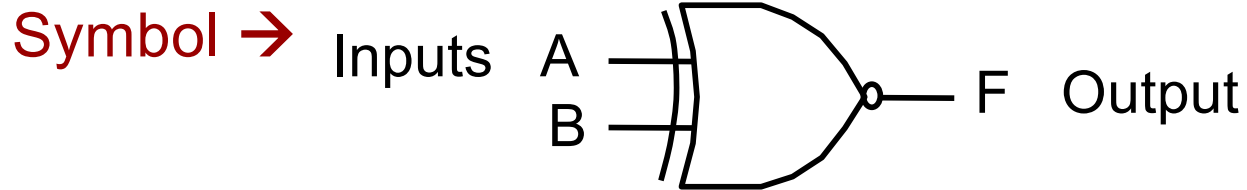
Circuit Switching Diagram →



XOR=Exclusive OR

# Other Gates (5/5): XNOR

- It does the *opposite of* an **XOR** gate.



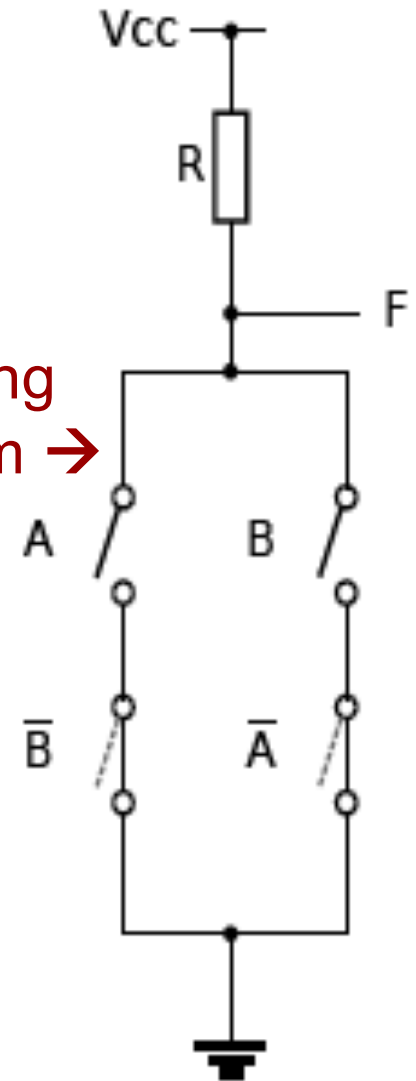
Truth table →

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Switching Algebra Equation →

$$F = (A \oplus B)'$$
$$F = \overline{(A \oplus B)}$$
$$F = A'B' + AB$$
$$F = \bar{A}\bar{B} + AB$$

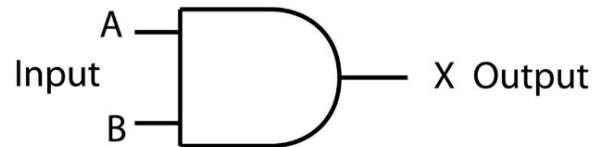
Circuit Switching Diagram →



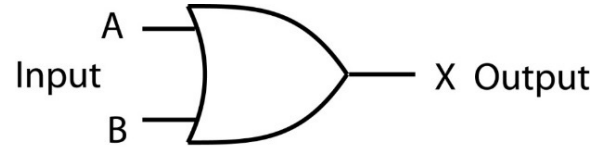
XNOR=Exclusive NOR

# Gates Summary (1/3)

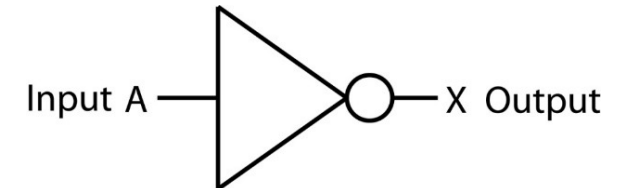
## AND



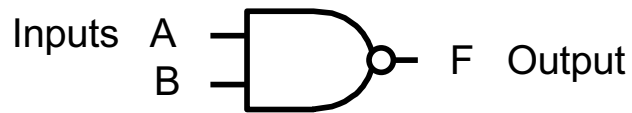
## OR



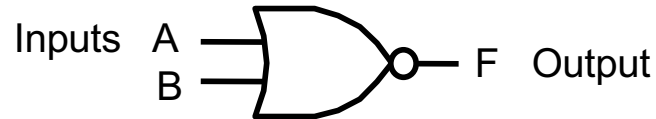
## NOT



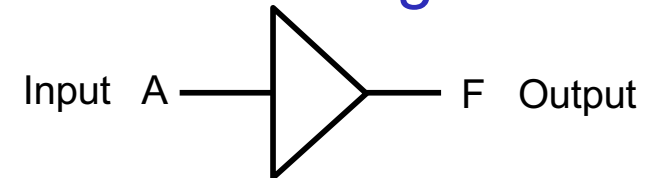
## NAND



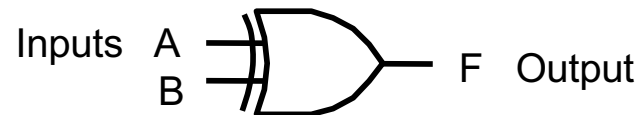
## NOR



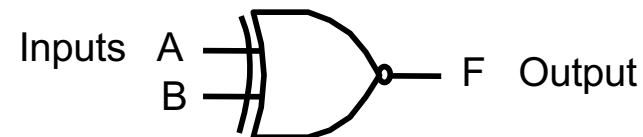
## non-inverting buffer



## XOR

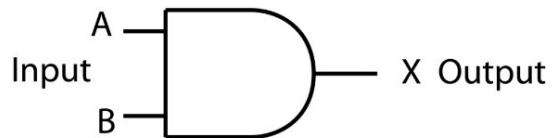


## XNOR



# Gates Summary (2/3)

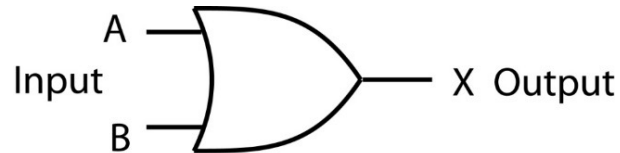
## AND



Input		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

$$X = A.B$$

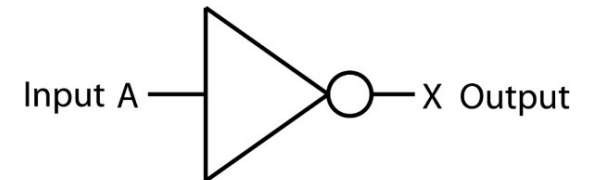
## OR



Input		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

$$X = A+B$$

## NOT



Input	Output
A	X
0	1
1	0

$$X = A'$$



# Gates Summary (3/3)

NAND

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

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

NOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

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

XOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

$$F = (A \oplus B)$$

XNOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

$$F = (A \oplus B)'$$

$$F = A'B + AB' \quad F = A'B' + AB$$

# Why do we manipulate equations?

---

- Before we get to the theorems, **why worry about manipulating Boolean equations?**
  - Simplification of digital circuits.
  - Fewer gates required.
  - Potential for faster operation.
  - Smaller is always better!!

# Switching Algebra: Basic Definitions

- **Literal**: variable or the complement of a variable.
  - **Examples**:  $X$ ,  $Y$ ,  $X'$ ,  $Y'$
- **Product Term**: single literal or logical product of 2+ literals.
  - **Examples**:  $X$ ,  $X \cdot Y \cdot Z'$ ,  $X \cdot Y'$
- **Sum of Products Expression (SOP)**: logical sum of product terms
  - **Example**:  $X \cdot Z' + X \cdot Y \cdot Z' + X \cdot Y'$
- **Sum Term**: single literal or logical sum of 2+ literals.
  - **Examples**:  $X$ ,  $X + Y + Z'$ ,  $X + Y'$
- **Product of Sums Expression (POS)**: logical product of sum terms
  - **Example**:  $(X + Z') \cdot (X + Y + Z') \cdot (X + Y')$

# Precedence Rules & Axioms

- **Precedence Rules:**  $'$ ,  $\cdot$  and  $+$ . (invert, and, or)
- **Axioms:**
  - A minimum set of basic mathematic definitions that are assumed to be always **True** and from which we can derive theorems.
  - Let **X** be a logic variable taking on values **0** or **1**.

(A1) $X=0$ if $X \neq 1$	(A1') $X=1$ if $X \neq 0$
(A2) $X=0$ then $X'=1$	(A2') $X=1$ then $X'=0$
(A3) $0 \cdot 0=0$	(A3') $1+1=1$
(A4) $1 \cdot 1=1$	(A4') $0+0=0$
(A5) $0 \cdot 1=1 \cdot 0=0$	(A5') $0+1=1+0=1$

# Single Variable Theorems & Perfect Induction

- Let **X** be a logic variable:

(T1) $X+0 = X$	(T1') $X \cdot 1 = X$
(T2) $X+1 = 1$	(T2') $X \cdot 0 = 0$
(T3) $X+X = X$	(T3') $X \cdot X = X$
(T4) $(X')' = X$	
(T5) $X+X' = 1$	(T5') $X \cdot X' = 0$



These theorems can be proved using axioms via *perfect induction*.

- Perfect Induction** (better known as *common sense*).

– *Example*:

- (T1')  $X \cdot 1 = X$

2 possible values for literal **X**

$$\left\{ \begin{array}{l} [X = 0] \rightarrow 0 \cdot 1 = 0 \text{ (by axiom A5)} \\ [X = 1] \rightarrow 1 \cdot 1 = 1 \text{ (by axiom A4)} \end{array} \right.$$

Proof

Hence,  $X \cdot 1 = X$ .

# Two (and Three) Variable Theorems

$$(T6) X + Y = Y + X$$

$$(T6') XY = YX$$

(Commutativity)

$$(T7) (X + Y) + Z = X + (Y + Z)$$

$$(T7') (XY)Z = X(YZ)$$

(Associativity)

$$(T8) XY + XZ = X(Y + Z)$$

$$(T8') (X + Y)(X + Z) = X + YZ$$

(Distributivity)

$$(T9) X + XY = X$$

$$(T9') X(X + Y) = X$$

(Covering) Absorption

$$(T10) XY + XY' = X$$

$$(T10') (X + Y)(X + Y') = X$$

(Combining)

$$(T11) XY + X'Z + YZ = XY + X'Z$$

(Consensus)

$$(T11') (X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)$$

**Adsorption Theorem** (not in the textbook):

$$(T^*) X + X'Y = X + Y \quad (T^*)' X(X' + Y) = XY$$

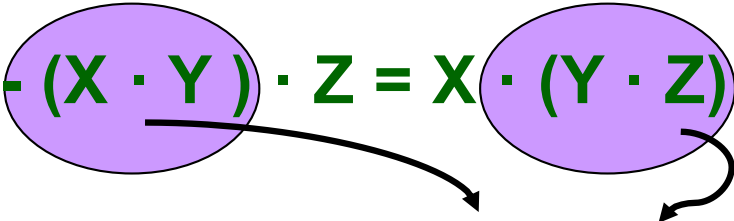


All theorems can be proved, either *algebraically* or through the use of a *truth table*.

**Principle of Duality:** Any theorem or identity in Switching Algebra remains True if **0's** and **1's** are swapped, and the same for **• (AND)** and **+ (OR)**.

# Example: Proof by Truth Table

- Show that ...

$$(T7') \quad (X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$


X	Y	Z	XY	YZ	(XY)Z	X(YZ)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

# N Variable Theorems

- **Two or three variables' theorems can be extended** to an arbitrary number of variables, **N**:

- **Generalised Idempotency**

$$(T12) \quad X + X + \dots + X = X$$

$$(T12') \quad X \cdot X \cdot \dots \cdot X = X$$



(T14) states: given any *N*variable logic expression, its complement can be obtained by swapping  $+$  and  $\cdot$ , and complementing all variables.

- **De Morgan's Theorems**

$$(T13) \quad (X_1 \cdot X_2 \cdot \dots \cdot X_n)' = X_1' + X_2' + \dots + X_n'$$

$$(T13') \quad (X_1 + X_2 + \dots + X_n)' = X_1' \cdot X_2' \cdot \dots \cdot X_n'$$

(T14): condensed form for (T13) & (T13').

- **Generalised De Morgan's Theorem**

$$(T14) \quad [F(X_1, X_2, \dots, X_n, +, \cdot)]' = F(X_1', X_2', \dots, X_n', \cdot, +)$$

- **Shannon's Expansion Theorems**

$$(T15) \quad F(X_1, X_2, \dots, X_n) = X_1 \cdot F(1, X_2, \dots, X_n) + X_1' \cdot F(0, X_2, \dots, X_n)$$

$$(T15') \quad F(X_1, X_2, \dots, X_n) = [X_1 + F(0, X_2, \dots, X_n)] \cdot [X_1' + F(1, X_2, \dots, X_n)]$$

**De Morgan: "Break the line and change the sign"**



# De Morgan's Theorems: Truth Tables

(T13)  $(X + Y)' = X' \cdot Y'$

(T13')  $(X \cdot Y)' = X' + Y'$



(T13), (T13') for **N=2** variables.

X	Y	X'	Y'	X+Y	(X+Y)'	X'•Y'	X•Y	(X•Y)'	X' + Y'
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0



Like *Theorem 8*, **De Morgan's law** may be used to convert between a sum of products and a product of sums.

# The Principle of Duality

- The **Principle of Duality**:
  - Used in *simplifying logic equations*.
  - Obtain the *dual* by interchanging OR and AND operations.
- **Example**:
  - The dual of  $F = X \cdot Y + Z \cdot W$  is  $F^D = (X + Y) \cdot (Z + W)$ .



**Principle of Duality**: Not the same as saying the two expressions are equal!

# Complements

- How do we **get the complement of a function**?
  - Can derive the complement of a function algebraically by applying De Morgan's theorem.
  - or**
  - Can obtain via Truth Table by interchanging **1's** and **0's** for the function **F**.
  - or**
  - Can obtain by interchanging **AND** and **OR** operations and complementing each variable.



This is a *straightforward application of De Morgan's theorem.*

# Example: Complement Function

Truth Table

A	B	C	F	F'
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Using a Truth Table to find  $F'$ .

Find the complement of  $F$ :

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

**Step 1:** Find the Dual.

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

**Step 2:** Complement each variable.

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

*The result is the Complement Function!*

# Consensus Theorem (T11)

- Useful in simplifying Boolean expressions
  - Allows the elimination of unnecessary terms.

Truth Table

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

**Show:**  $XY + \bar{X}Z + YZ = XY + \bar{X}Z$

$$F = XY + \bar{X}Z + YZ = XY + \bar{X}Z + YZ(\underbrace{X + \bar{X}}_{=1})$$

expand

$$= XY + \bar{X}Z + XYZ + \bar{X}YZ$$

reorder

$$= XY + XYZ + \bar{X}Z + \bar{X}YZ$$

factorise

$$= XY(1 + Z) + \bar{X}Z(1 + Y)$$

$$G = XY + \bar{X}Z$$

$\therefore YZ$  is an unnecessary (or redundant) term

# Adsorption Theorem

**Adsorption Theorem** (not in the textbook):

$$(T^*) \quad X + X'Y = X + Y \quad (T^*)' \quad X(X' + Y) = XY$$

**Proof:**

$$\begin{aligned} X + X'Y &= (X + X')(X + Y) \quad (T8') \\ &= 1.(X+Y) \quad (T5) \\ &= X+Y \quad (T1') \end{aligned}$$

$$(T8') \quad (X + Y)(X + Z) = X + YZ$$

$$(T5) \quad X+X' = 1$$

$$(T1') \quad X \cdot 1 = X$$

# Theorems (T8) – (T11)

- **Theorem (T8)** may be used to convert a *product of sums* expression to a *sum of products* expression.
- **Theorem (T8')** is often used to convert a *sum of products* expression to a *product of sums* expression.
- **Theorems (T9), (T10), and (T11)** are often used to **minimise** (or *simplify*) **a logic circuit**.
  - **Shared property** by these theorems: there is a reduction in the number of logic gates from left to right.

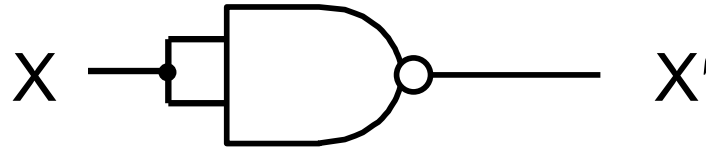


Examples of this later on!

# Designing with NAND gates

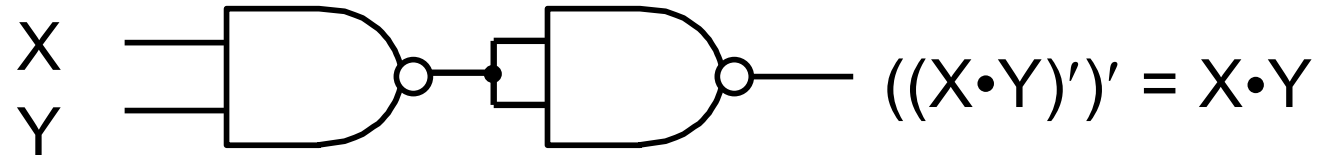
- Examples:**

**INVERTER**

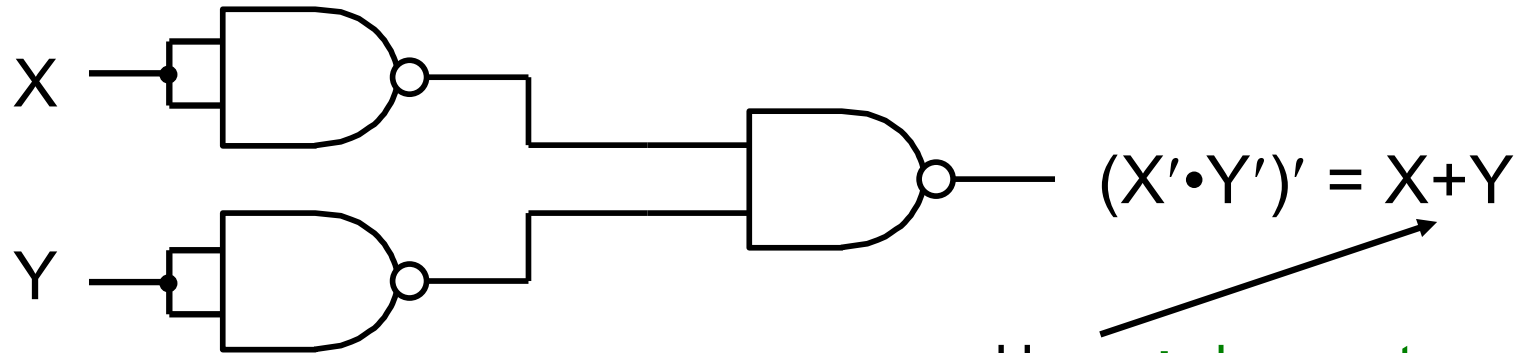


*This is only to simplify the implementation!*

**AND**



**OR**

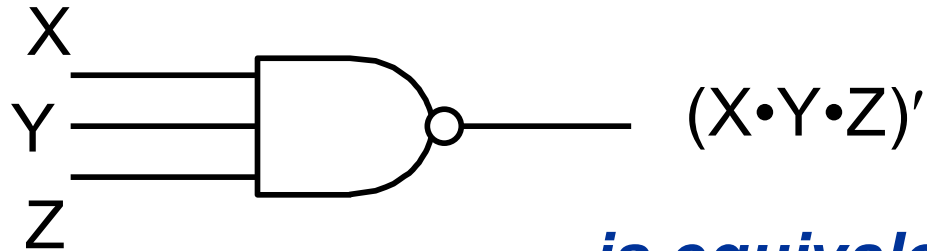


Here, **+** does not represent addition.



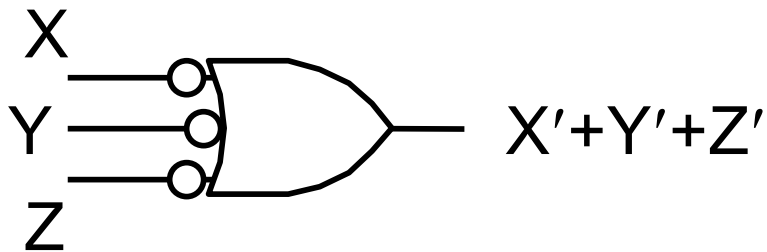
# NAND Gates: 2 ways to view

- **3-input NAND** gate:



*is equivalent to*

- **3 INVERTED input OR** gate:

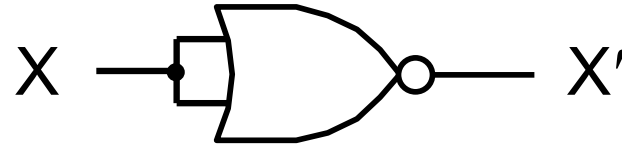


X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

# Designing with NOR gates

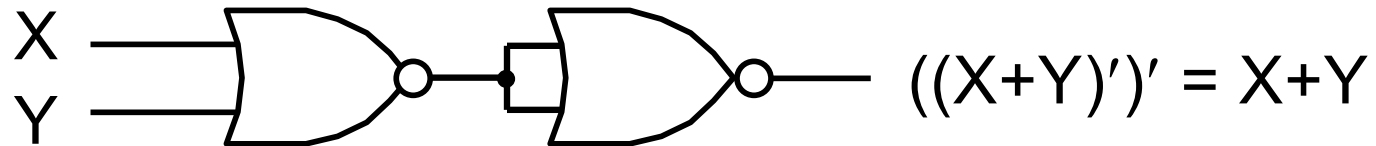
- Examples:**

## INVERTER

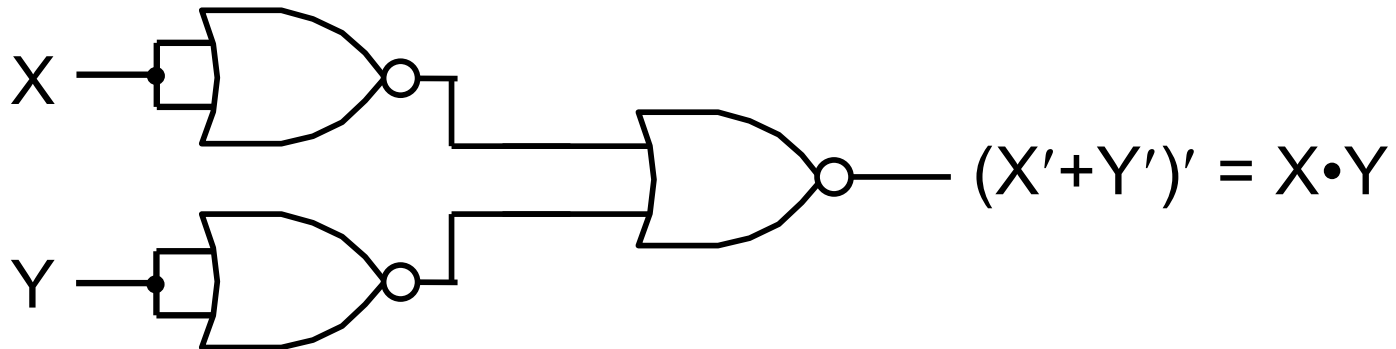


*This is only to simplify the implementation!*

## OR

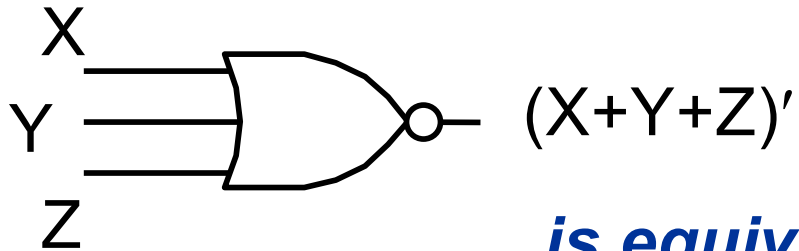


## AND



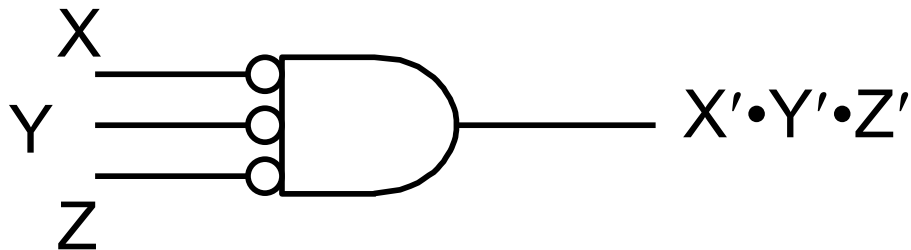
# NOR Gates: 2 ways to view

- **3-input NOR** gate:



*is equivalent to*

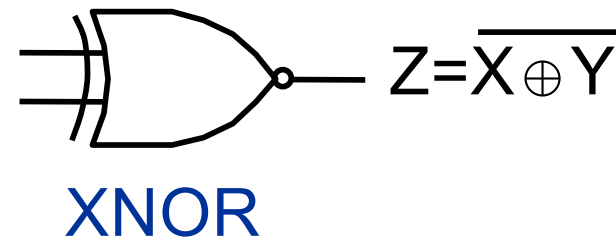
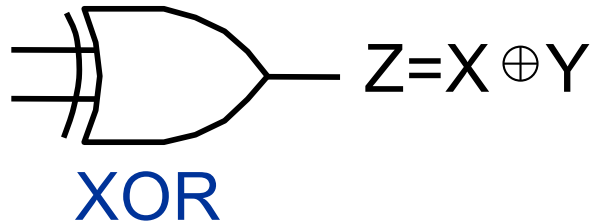
- **3 INVERTED input AND** gate:



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

# Exclusive OR Logic Gates: Properties

- XOR** and **XNOR** gates:



- Identities:**

$$X \oplus 0 = X$$

$$X \oplus 1 = X'$$

$$X \oplus X = 0$$

$$X \oplus X' = 1$$

$$X \oplus Y' = (X \oplus Y)'$$

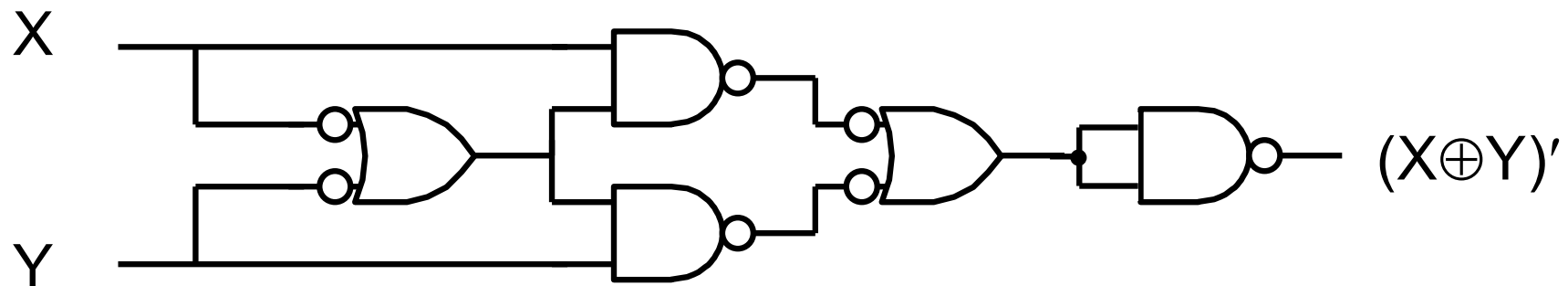
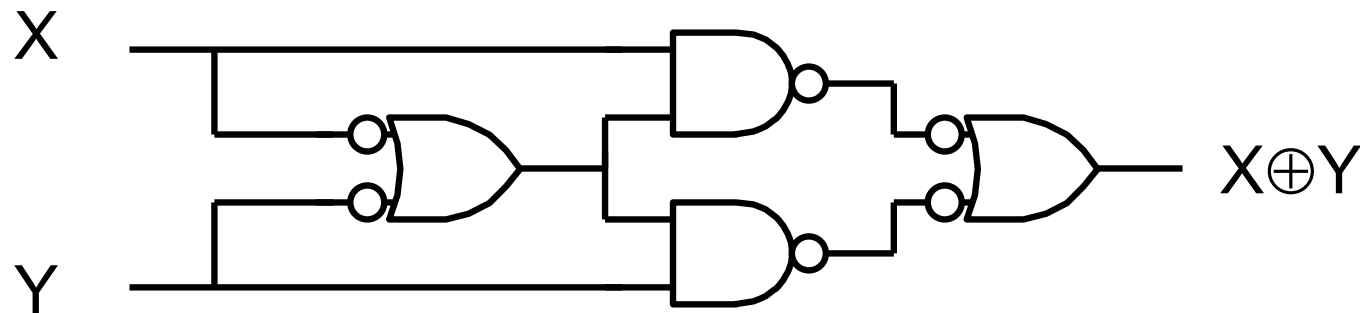
$$X' \oplus Y = (X \oplus Y)'$$

$$X \oplus Y = Y \oplus X$$

$$(X \oplus Y) \oplus Z = X \oplus Y \oplus Z$$

# XOR & XNOR Gates

- XOR** and **XNOR** gates can be designed using basic gates (i.e., *inverted input OR gates* and *NAND gates*):

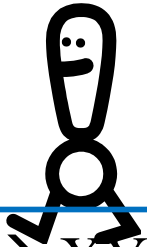


# Example (1/3): Two Equations

- Show that  $F_1 = F_2$  using a Truth Table.

$$F_1 = X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z + X \cdot Y'$$

$$F_2 = X' \cdot Z + X \cdot Y'$$



To prove the equality, we can use Switching Algebra theorems **T8 + T5**.

$$(T8) \rightarrow XY + XZ = X(Y + Z)$$

$$(T5) \rightarrow X + X' = 1$$

**Show:** prove that  $F_1 = F_2$  using Switching Algebra,

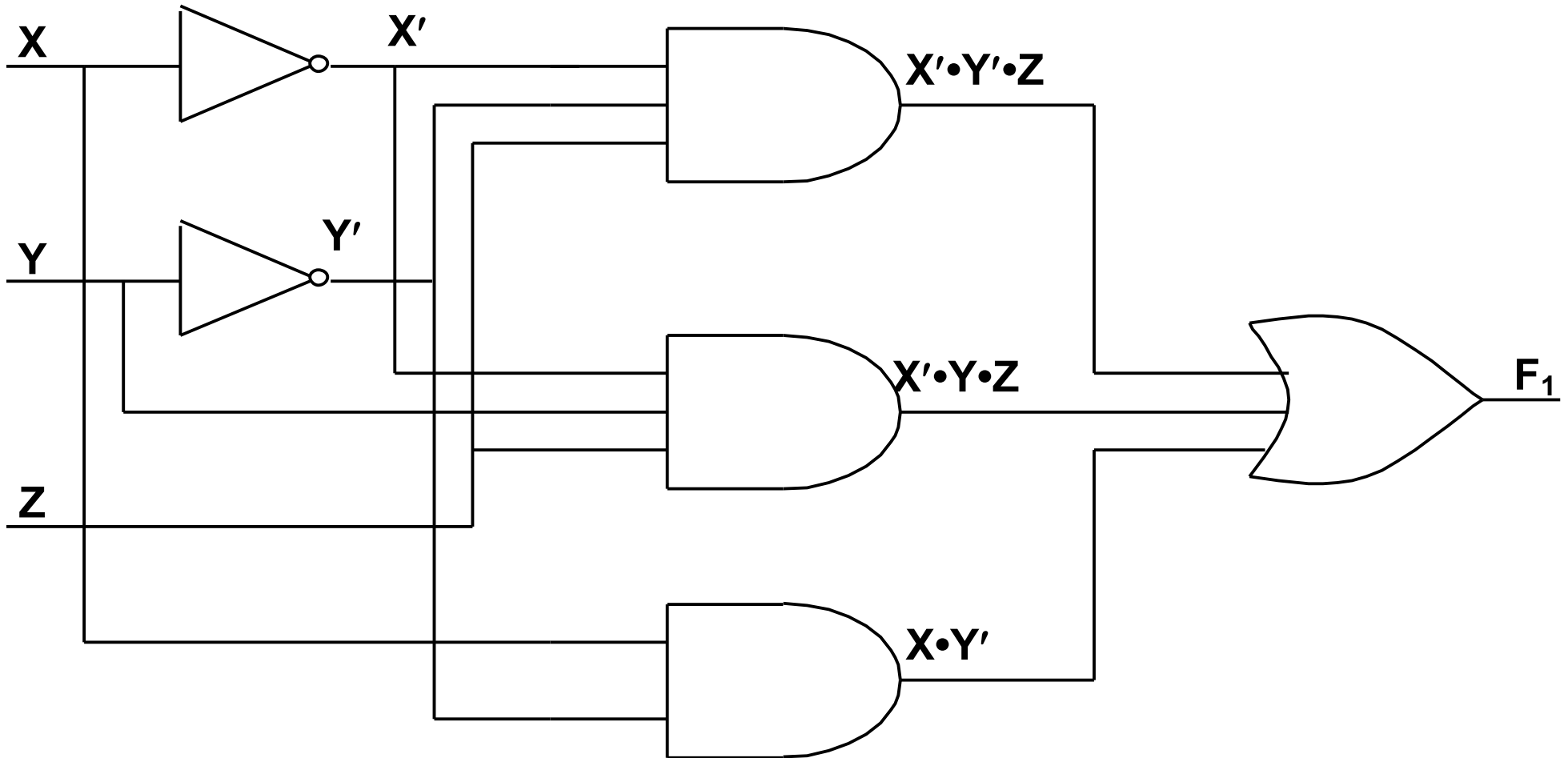
$$\begin{aligned} F_1 &= X'Y'Z + X'YZ + XY' = \\ &= X'Z(Y' + Y) + XY', \text{ using (T8)} \\ &= X'Z + XY', \text{ using (T5)} \end{aligned}$$

$$\Rightarrow F_2$$

X	Y	Z	F <sub>1</sub>	F <sub>2</sub>
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

# Example (2/3): Two Equations

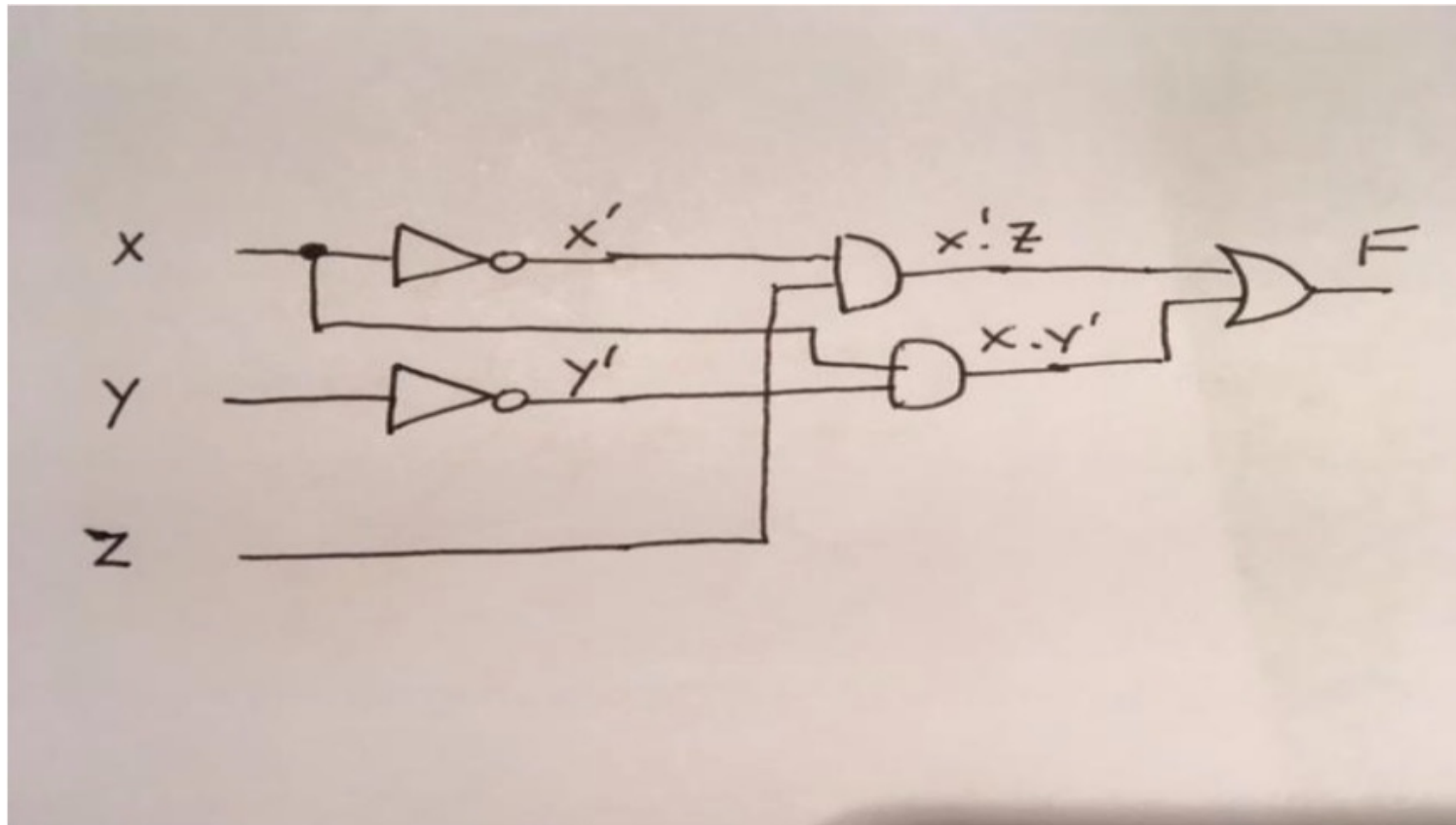
$$F_1 = X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z + X \cdot Y'$$



# Example (3/3): Two Equations

*To be completed in class ...*

$$F_2 = X' \cdot Z + X \cdot Y'$$





# Example 1: Minimisation

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



(T10) – Combining Theorem

$$F = B'C$$



How is (T10) applied here?

Alternatively, if we were to factor **A** out, it would have left us with:

$$F = B'C(A' + A) \quad (\text{using T8})$$

$$A' + A = 1 \quad (\text{using T5})$$

therefore,  $F = B'C$

# Example 2: Minimisation (1)

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

$$A'B'C = A'B'C + A'B'C$$

$$(T3) \quad X + X = X$$



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

$$\text{---} \quad \text{T10}$$

$$\text{---} \quad \text{T10}$$

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

$$(T10) \quad XY + XY' = X$$



*Circuit Diagrams for these functions?*

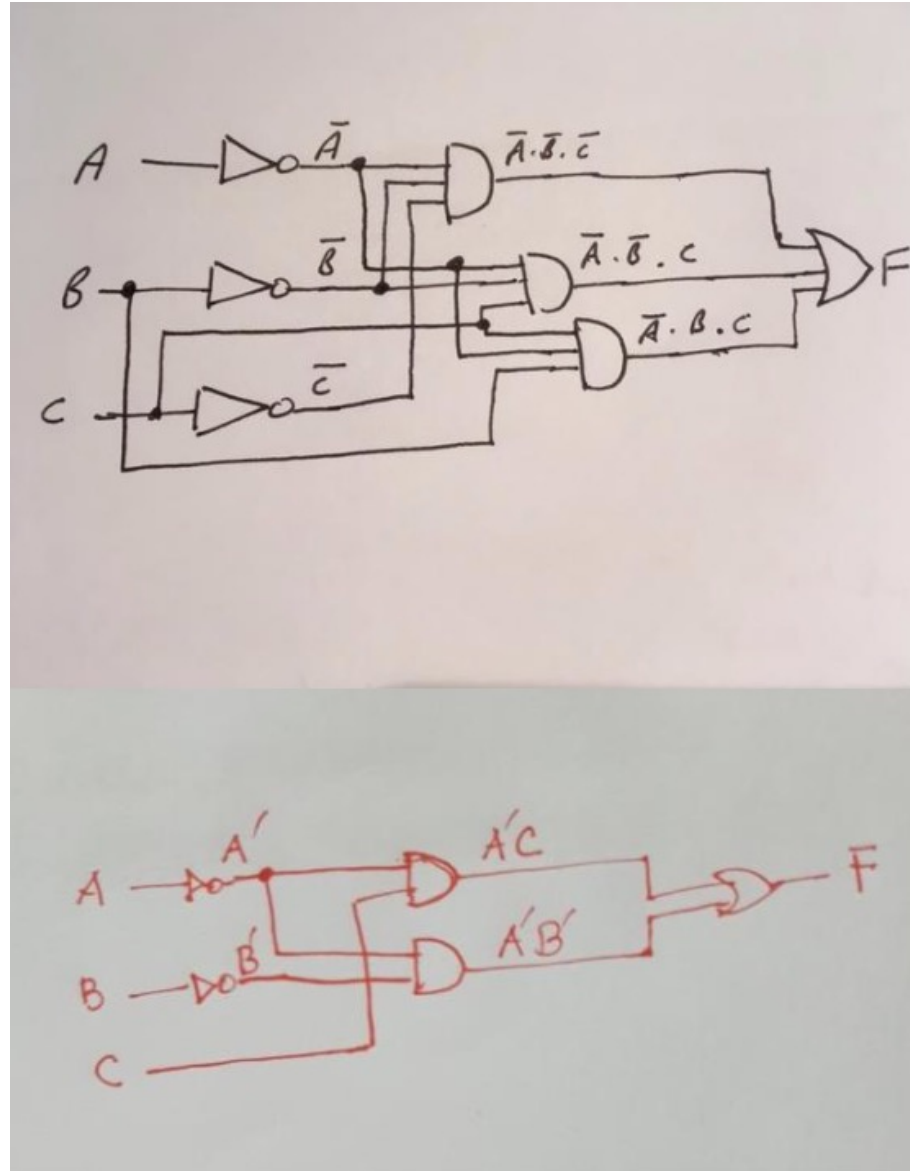
Note: Can be further simplified to  $F = A'(B'+C)$

# Example 2: Minimisation (2)

*To be completed in class ...*

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

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



# Example: Minimisation of F

$$\text{Simplify } F = (AB'(A + C))' + A'B(A + B' + C)'$$

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

MULTIPLY  
THROUGH  
TERM 1  
APPLYING  
(T3')  $X \cdot X = X$

APPLY (T13') to  
 $(AB' + AB'C)'$  &  
 $(A + B' + C)'$

$$\longrightarrow = (A' + B)(A' + B + C') + A'B(A'BC')$$

EXPAND TERMS

$$\left\{ \begin{aligned} &= (A' + B)(A' + B + C') + A'BC' \\ &= A'A' + A'B + A'C' + BA' + BB + BC' + A'BC' \end{aligned} \right.$$

$$\text{APPLY (T3')} \longrightarrow = A' + A'B + A'C' + BA' + B + BC' + A'BC'$$

$$\text{APPLY (T1')} \longrightarrow = A'(1 + B + C' + B + BC') + B(1 + C')$$

$$\text{APPLY (T2)} \longrightarrow = A'(1) + B(1)$$

$$= A' + B$$

# Logic Functions (1/2)

- Logic Functions' Representations:
  - **Truth Table**: practical only for a small number of variables.
  - **Algebraic sum of minterms** (*Canonical Sum*):
    - **minterm**: a product of  $n$  distinctive logic variables (or their complements); e.g.,  $X \cdot Y \cdot Z$ ;
    - the **sum of minterms** corresponds to the combination of Truth Table rows for which the function produces a **1** output;
    - for a  **$n$ -variable** logic function, each **minterm** must consist of  $n$  variables and within each **minterm**, each variable is represented by its complement if the variable value is **0**.

# Logic Functions (2/2)

- Logic Functions' Representations (cont.):
  - **Algebraic product of maxterms** (*Canonical Product*):
    - **maxterm**: is the sum of  $n$  distinctive logic variables or their complements e.g.,  $X+Y+Z$ ;
    - the **product of maxterms** corresponds to the product of Truth Table rows for which the function produces a **0** output;
    - for a  **$n$ -variable** logic function, each **maxterm** must consist of  $n$  variables and within each **maxterm**, each variable is represented by its complement if the variable value is **1**.

# Minterms & Maxterms for 3 Variables

X	Y	Z	Product Term	Symbol	Sum Term	Symbol
0	0	0	$\overline{X} \cdot \overline{Y} \cdot \overline{Z}$	$m_0$	$X+Y+Z$	$M_0$
0	0	1	$\overline{X} \cdot \overline{Y} \cdot Z$	$m_1$	$X+Y+\overline{Z}$	$M_1$
0	1	0	$\overline{X} \cdot Y \cdot \overline{Z}$	$m_2$	$X+\overline{Y}+Z$	$M_2$
0	1	1	$\overline{X} \cdot Y \cdot Z$	$m_3$	$X+\overline{Y}+\overline{Z}$	$M_3$
1	0	0	$X \cdot \overline{Y} \cdot \overline{Z}$	$m_4$	$\overline{X}+Y+Z$	$M_4$
1	0	1	$X \cdot \overline{Y} \cdot Z$	$m_5$	$\overline{X}+Y+\overline{Z}$	$M_5$
1	1	0	$X \cdot Y \cdot \overline{Z}$	$m_6$	$\overline{X}+\overline{Y}+Z$	$M_6$
1	1	1	$X \cdot Y \cdot Z$	$m_7$	$\overline{X}+\overline{Y}+\overline{Z}$	$M_7$

**Minterms**



**Maxterms**

A *minterm* and a *maxterm* with the same subscript number are complements of each other:  $M_0 = \overline{m_0}$

# Representing a Function in *Standard Form*

A **Boolean Function** can be expressed algebraically from a given Truth Table by forming the logical sum of all *minterms* which produce a '**1**' in the function.

**Truth Table**

X	Y	Z	F	$\bar{F}$	
0	0	0	1	0	$m_0$
0	0	1	0	1	$m_1$
0	1	0	1	0	$m_2$
0	1	1	0	1	$m_3$
1	0	0	0	1	$m_4$
1	0	1	1	0	$m_5$
1	1	0	0	1	$m_6$
1	1	1	1	0	$m_7$

**F = 1** for each of the combinations of variables:  
 $X' \cdot Y' \cdot Z'$ ,  $X' \cdot Y \cdot Z'$ ,  $X \cdot Y' \cdot Z$ ,  $X \cdot Y \cdot Z$ .

**So:**  $F = X' \cdot Y' \cdot Z' + X' \cdot Y \cdot Z' + X \cdot Y' \cdot Z + X \cdot Y \cdot Z$

$$F = m_0 + m_2 + m_5 + m_7 = \Sigma m(0, 2, 5, 7)$$

**And:**  $F' = m_1 + m_3 + m_4 + m_6 = \Sigma m(1, 3, 4, 6)$

$$F'' = (m_1 + m_3 + m_4 + m_6)'$$

$$F'' = m_1' \cdot m_3' \cdot m_4' \cdot m_6'$$

$$F = M_1 \cdot M_3 \cdot M_4 \cdot M_6 = \prod M(1, 3, 4, 6)$$



# Minterms & Sum of Products

---

- Properties of *Minterms*:

- There are  $2^n$  minterms for  $n$  Boolean variables.
  - Any Boolean function can be expressed as a logical sum of *minterms*.
  - A function's complement contains those *minterms* not included in the original function.
  - A function that includes all the  $2^n$  *minterms* is always equal to logic value  $1$ .
- 

- Sum of Products' Expressions:

- Similar to *sum of minterms*, but the *sum of products* is in simplified form and may not contain all the variables in each expression.
- Used in design process to achieve solutions with minimum gate counts.

# Example: Sum of Products & Simplification (1/2)



A Boolean Function in **Sum of Minterms** form can be simplified to a **Sum of Products** form by algebraic manipulation or map simplification.

Truth Table

X	Y	Z	F	
0	0	0	1	m <sub>0</sub>
0	0	1	1	m <sub>1</sub>
0	1	0	1	m <sub>2</sub>
0	1	1	0	m <sub>3</sub>
1	0	0	1	m <sub>4</sub>
1	0	1	1	m <sub>5</sub>
1	1	0	0	m <sub>6</sub>
1	1	1	0	m <sub>7</sub>

$F = 1$  for each of the combinations of variables:  
 $X' \cdot Y' \cdot Z'$ ,  $X' \cdot Y' \cdot Z$ ,  $X' \cdot Y \cdot Z'$ ,  $X \cdot Y' \cdot Z'$ ,  $X \cdot Y' \cdot Z$

So:

$$F = X' \cdot Y' \cdot Z' + X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z' + X \cdot Y' \cdot Z' + X \cdot Y' \cdot Z$$

$$F = \Sigma m(0, 1, 2, 4, 5)$$

∴  $F$  can be simplified by algebraic or map simplification techniques to a **Sum of Products**.

Then can use Boolean Algebra to simplify the function.....

# Example: Sum of Products & Simplification (2/2)



Which equation is easier to build?

$$F = X' \cdot Y' \cdot Z' + X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z' + X \cdot Y' \cdot Z' + X \cdot Y' \cdot Z$$

T10

T10

$$F = X' \cdot Y' + X' \cdot Y \cdot Z' + X \cdot Y'$$

T10

$$F = Y' + X' \cdot Y \cdot Z'$$

T\* (Adsorption Theorem)

$$F = Y' + X' \cdot Z'$$



Simplification of **F** using algebraic manipulation.

# Exercise

---

Simplify the following two functions:

a)  $F = AB'(C+D) + C'D'$

b)  $G = (A+B)(A+C')(A+D)(BC'D+E)$

# Exercise

Solutions.....

a)

$$\begin{aligned} F &= AB'(C+D) + C'D' \\ &= AB'(C'D')' + C'D' \quad [\text{de Morgan's Theorem}] \\ &= AB' + C'D' \quad [\text{T8'}] \end{aligned}$$

b)

$$\begin{aligned} G &= (A+B)(A+C')(A+D)(BC'D+E) \\ &= (A+BC')(A+D)(E+BC'D) \quad [\text{T8': } (X+Y)(X+Z)=X+YZ] \\ &= (A+BC'D)(E+BC'D) \quad [\text{T8': } (X+Y)(X+Z)=X+YZ] \\ &= AE + BC'D \quad [\text{T8'}] \end{aligned}$$

# Minterm & Maxterm Expansions of $F, F'$

**Example** for  $F(X, Y, Z)$ :  $2^3$  terms (numbered 0 through to 7).

Given Form	Minterm Expansion of $F$	Maxterm Expansion of $F$	Minterm Expansion of $F'$	Maxterm Expansion of $F'$
$F = \sum m(3,4,5,6,7)$	–	$\prod M(0,1,2)$	$\sum m(0,1,2)$	$\prod M(3,4,5,6,7)$
$F = \prod M(0,1,2)$	$\sum m(3,4,5,6,7)$	–	$\sum m(0,1,2)$	$\prod M(3,4,5,6,7)$

# Expansion into Canonical SOP & POS

---

- How to **expand Boolean function into canonical SOP**:
  - Use properties (T1') and (T5).
  - Take each *product term* with a missing literal variable (e.g.,  $A$ ) and **AND** it (logic operation) with the *sum term* ( $A + A'$ ).

---
- How to **expand Boolean function into canonical POS**:
  - Use the *distributive property* i.e., (T8').
  - Take each *sum term* with a missing literal variable (e.g.,  $A$ ) and **OR** it (logic operation) with the *product term* ( $A.A'$ ).

# Example: Sum of Products

**Problem:** Expand the boolean function  $F(A,B,C) = A'B' + BC$  into a standard *sum of products* (aka *minterm expansion*, *canonical SOP*).

$$\begin{aligned} F(A,B,C) &= A'B' + BC \\ &= A'B'(C+C') + BC(A+A') \quad \text{by (T5), (T1')} \\ &= A'B'C + A'B'C' + ABC + A'BC \quad \text{by (T8), (T6), (T6')} \end{aligned}$$

To get the  $\Sigma m$  expression, replace all *normal literals* with a 1 and all *primed literals* with a 0: (**Example:**  $A'B'C \Rightarrow A' = 0, B' = 0, C = 1 \Rightarrow 001$ ).

$$F(A,B,C) = \Sigma m(001, 000, 111, 011)$$

Now convert to decimal (and put them in order),  $F(A,B,C) = \Sigma m(0, 1, 3, 7)$ .



# Example: Product of Sums

**Problem:** Expand the boolean function  $F(A,B,C) = A'B' + BC$  into a standard *product of sums* (aka *maxterm expansion*, *canonical POS*).

$$\begin{aligned} F(A,B,C) &= A'B' + BC \\ &= (A'B' + B)(A'B' + C) \quad \longrightarrow \text{by variable substitution, (T8')} \\ &= (A' + B)(B' + B)(A' + C)(B' + C) \text{ by (T8')} \\ &= (A' + B + CC')(A' + C + BB')(B' + C + AA') \text{ by (T1), (T5')} \\ &= (A' + B + C)(A' + B + C')(A' + C + B)(A' + C + B')(B' + C + A)(B' + C + A') \\ &= (A' + B + C)(A' + B + C')(A' + B' + C)(A + B' + C) \text{ by (T3'), (T6)} \end{aligned}$$

To get the  $\Pi M$  expression replace all *normal literals with a 0* and all *primed literals with a 1*: (**Example:**  $A' + B' + C \Rightarrow A' = 1, B' = 1, C = 0 \Rightarrow 110$ ).

$$F(A,B,C) = \Pi M (100, 101, 110, 010)$$

Now convert to decimal (and order them):  $F(A,B,C) = \Pi M (2, 4, 5, 6)$ .

# Deriving the Standard *Product of Sums* Expression

- Remember:** If you are unsure of your answer, you can check it with a Truth Table; fill in Truth Table as per original  **$F = A'B' + BC$** !

For:  $F'$  (*minterms*) or  
 $F$  (*maxterms*) where  $F = 0$ .



Does our expression have  
all the terms indicated by  
the Truth Table?

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

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

For:  $F$  (*minterms*)  
or  $F'$  (*maxterms*)  
where  $F = 1$ .