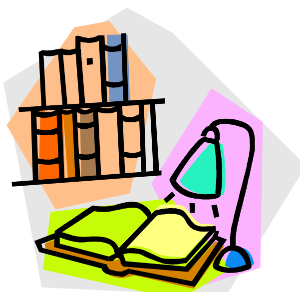
Overview: Switching Algebra & Combinational Logic Design

* **Switching Algebra**
* **Combinational Circuit Analysis & Synthesis**

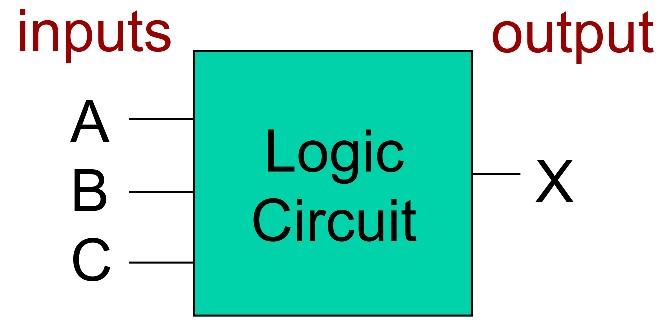


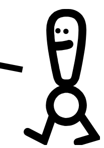
Chapters 4 & 6 – "Digital Design: Principles and Practices" book

Queen Mary University of London

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.



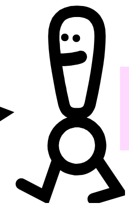


Studied later in the course.

- These slides are concerned with the ***analysis*** and ***synthesis*** of ***combinational*** logical ***circuits***.
  - **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.
  - For a variable *X*:
    - ***X* = 0** *or* ***X* = 1**



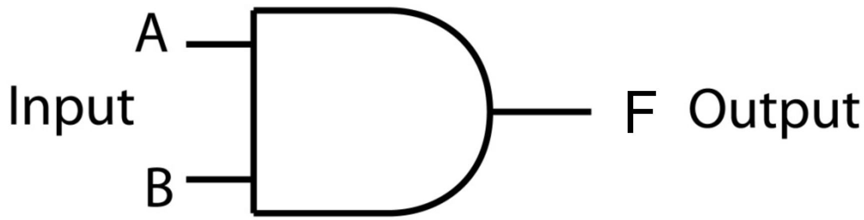
No other values are allowed.

  - Many other possible "Boolean" representations:
    - *X* = *Light on* or *X* = *Light off*
    - *X* = voltage HIGH or *X* = voltage LOW
  - *Basic operations* are AND, OR and complement (or *invert*).

Basic Gates (1/3): AND

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

Symbol →

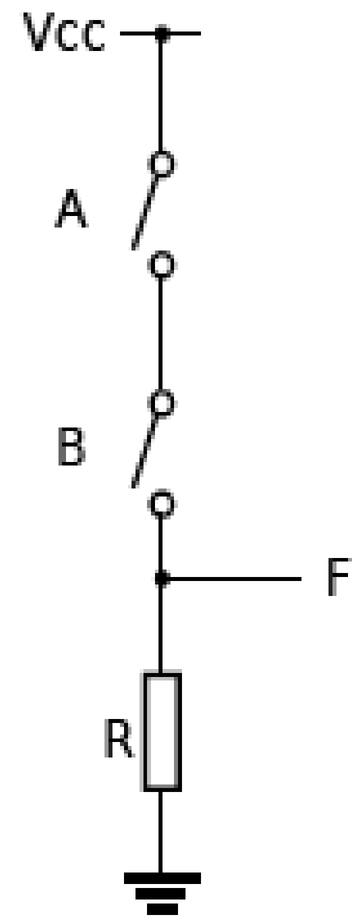


Truth table →

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

Binary ascending counting order.

Circuit Switching Diagram →

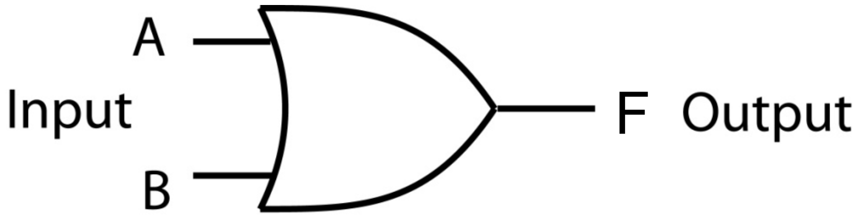


Switching Algebra Equation → $F = A \cdot B$ *or* $F = AB$

Basic Gates (2/3): OR

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

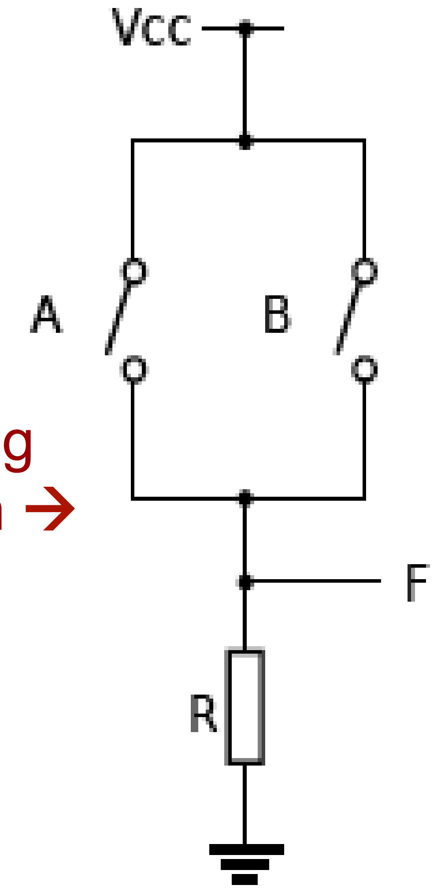
Symbol →



Truth table →

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

Circuit Switching Diagram →

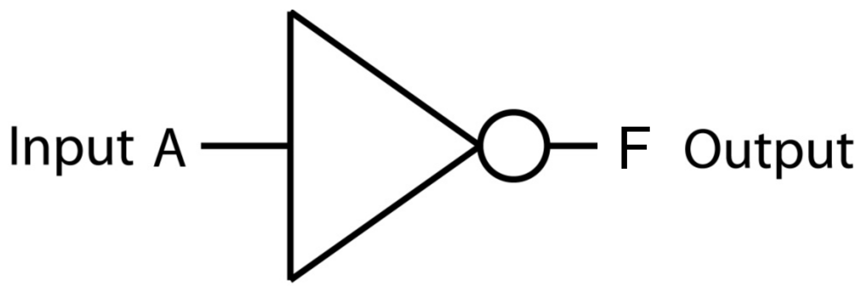


Switching Algebra Equation → $F = A+B$

Basic Gates (3/3): NOT

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

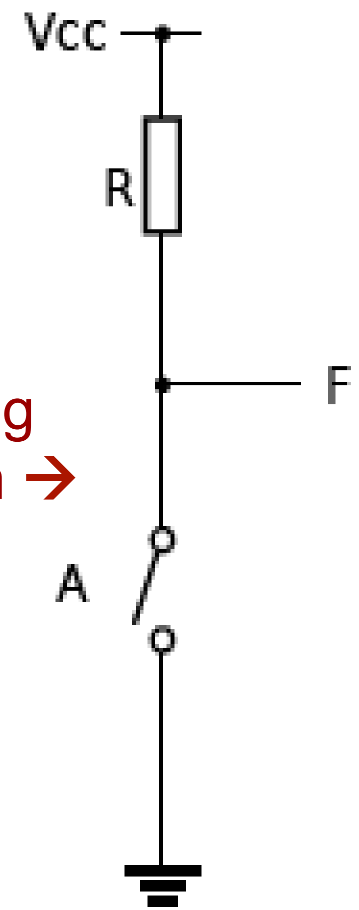
Symbol →



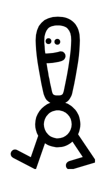
Truth table →

Input A	Output F
0	1
1	0

Circuit Switching Diagram →



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



Other notation (not used here): F=~A *or* F=¬A.

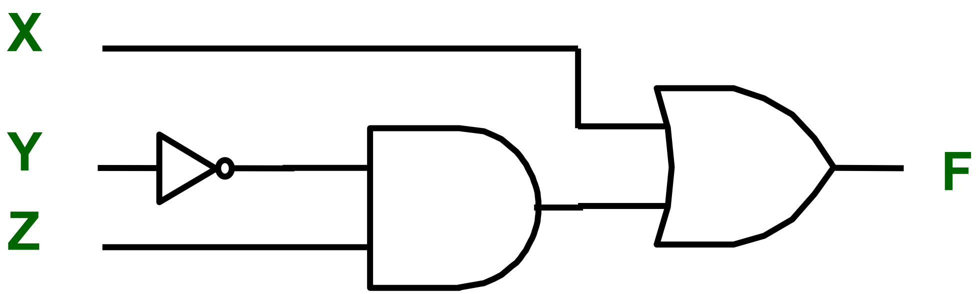
Example: Combining Basic Gates

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

Switching Algebra

$F(X, Y, Z) = X + Y' \cdot Z$

Logic Gate Diagram



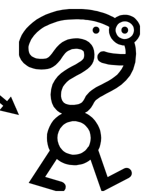
Truth Table

X	Y	Z	F = X+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

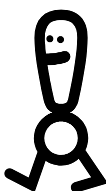
Queen Mary University of London

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.



Why do you think that is the case?

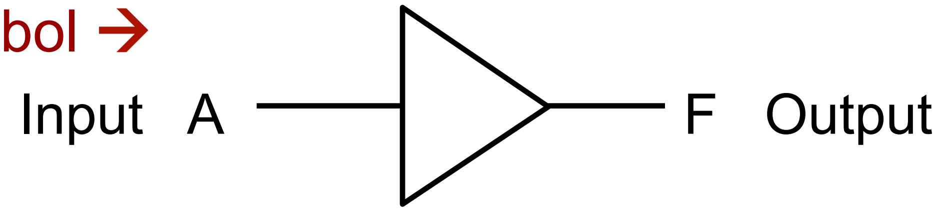


Any digital circuit of any complexity can be made from **NAND** gates.

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 →

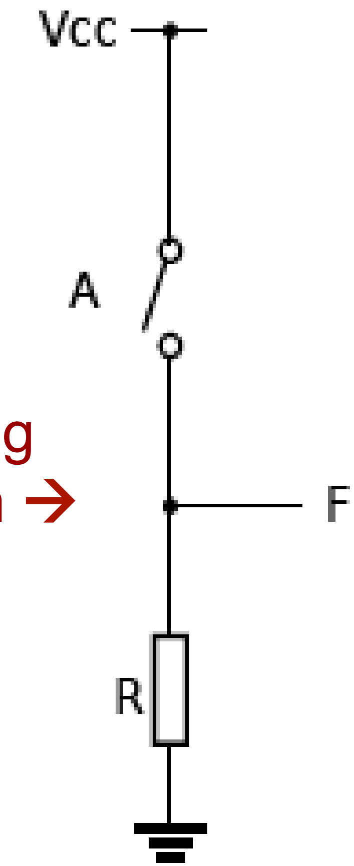


Truth table →

A	F
0	0
1	1

Switching Algebra Equation → F = A

Circuit Switching Diagram →

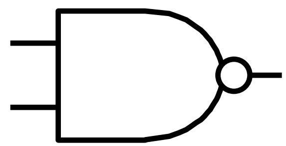


Other Gates (2/5): NAND

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

Symbol →

Inputs A B



F Output

Truth table →

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

Switching Algebra Equation →

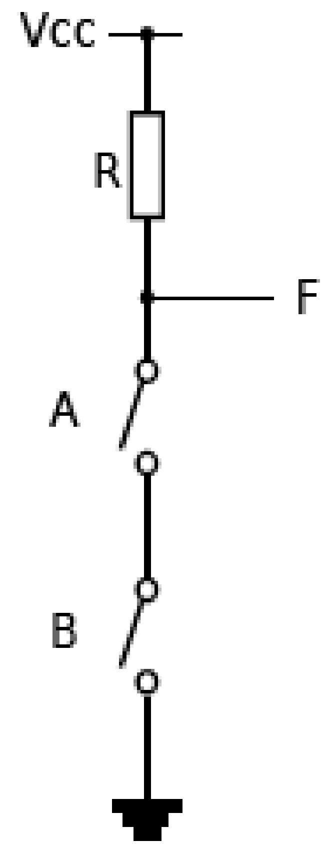
$F = (A \cdot B)'$ *or*

$F = (AB)'$ *or*

$F = \overline{(A \cdot B)}$ *or*

$F = \overline{(AB)}$

Circuit Switching Diagram →

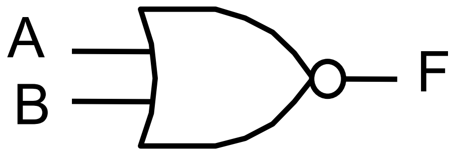


Other Gates (3/5): NOR

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

Symbol →

Inputs



Output

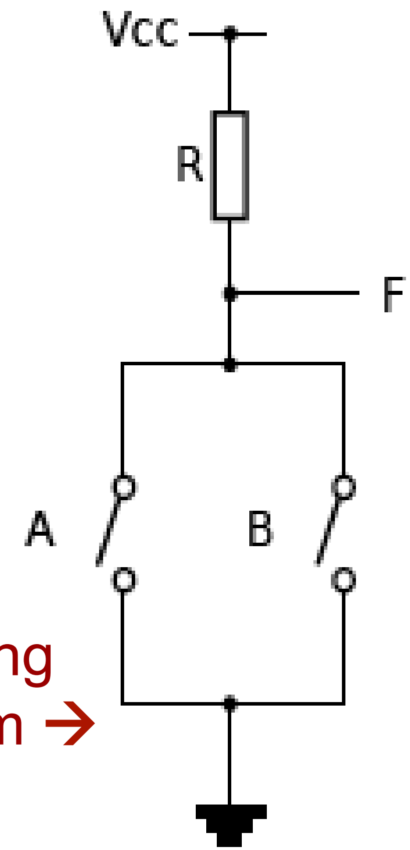
Truth table →

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

Switching Algebra Equation → $F = (A+B)'$ *or*

$F = (\overline{A+B})$

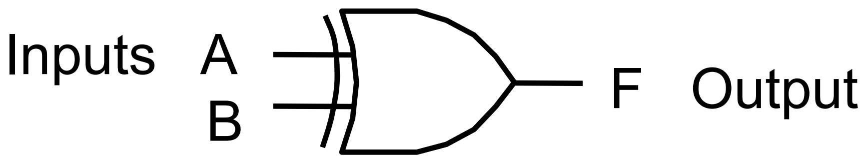
Circuit Switching Diagram →



Other Gates (4/5): XOR

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

Symbol →



Truth table →

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

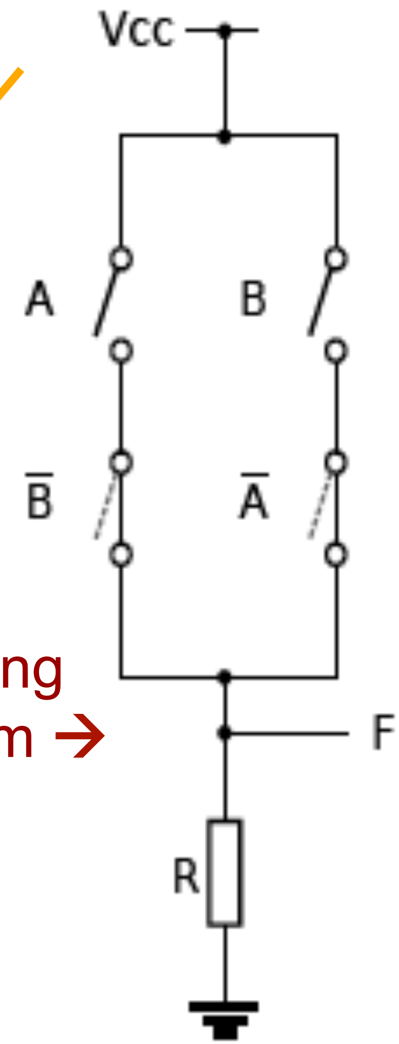
Switching Algebra Equation →

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

$F = A'B + AB'$ *or*

$F = \bar{A}B + A\bar{B}$

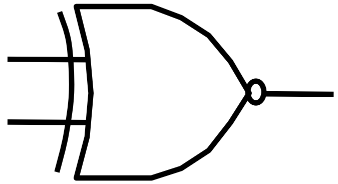
Circuit Switching Diagram →



Other Gates (5/5): XNOR

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

Symbol → Inputs



Truth table →

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

Switching Algebra Equation →

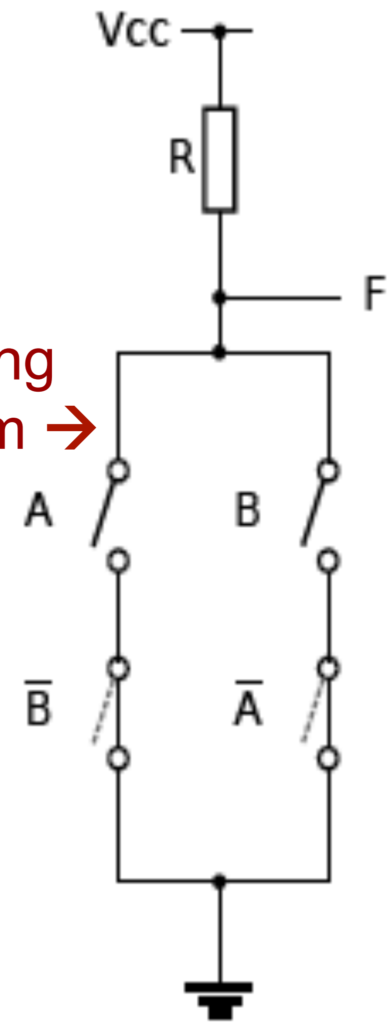
$F = (A \oplus B)'$ *or*

$F = \overline{(A \oplus B)}$ *or*

$F = A'B' + AB$ *or*

$F = \bar{A}\bar{B} + AB$

Circuit Switching Diagram →



Queen Mary University of London

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·Y·Z'**, **X·Y'**
- ***Sum of Products Expression***: logical sum of product terms
  - **Example**: **X·Z' + X·Y·Z' + X·Y'**
- ***Sum Term***: single literal or logical sum of 2+ literals.
  - **Examples**: **X**, **X + Y + Z'**, **X + Y'**
- ***Product of Sums Expression***: logical product of sum terms
  - **Example**: **(X + Z')·(X + Y + Z')·(X + Y')**

Precedence Rules & Axioms

- **Precedence Rules**: **'**, **·** and **+**.
- **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**.

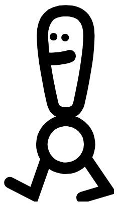
(A1) $X = 0$ if $X \neq 1$	(A1') $X = 1$ if $X \neq 0$
(A2) $X = 0$ then $X' = 1$	(A2') $X = 1$ then $X' = 0$
(A3) $0 \cdot 0 = 0$	(A3') $1 + 1 = 1$
(A4) $1 \cdot 1 = 1$	(A4') $0 + 0 = 0$
(A5) $0 \cdot 1 = 1 \cdot 0 = 0$	(A5') $0 + 1 = 1 + 0 = 1$

Queen Mary
University of London

Single Variable Theorems & Perfect Induction

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

(T1) X + 0 = X	**(T1') X·1 = X**
(T2) X + 1 = 1	**(T2') X·0 = 0**
(T3) X + X = X	**(T3') X·X = X**
(T4) (X')' = X	
(T5) X + X' = 1	**(T5') X·X' = 0**



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

- **Perfect Induction** (better known as ***common sense***).
  - ***Example***:
    - **(T1') X·1 = X**

Proof

2 possible values for literal ***X***

$$\begin{cases} [X = 0] \rightarrow 0\cdot 1 = 0 & (\textit{by axiom } \boldsymbol{A5}) \\ [X = 1] \rightarrow 1\cdot 1 = 1 & (\textit{by axiom } \boldsymbol{A4}) \end{cases}$$

Hence, **X·1=X**.

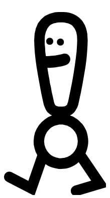
Queen Mary University of London

Two (and Three) Variable Theorems

(T6) $X + Y = Y + X$	(T6') $XY = YX$	(Communtativity)
(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)
(T10) $XY + XY' = X$	(T10') $(X + Y)(X + Y') = X$	(Combining)

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

Absorption Theorem (not in the textbook):
(T*) $X + X'Y = X + Y$ (T*)' $X(X' + Y) = XY$



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

Queen Mary
University of London

Example: Proof by Truth Table

- Show that …

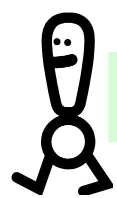
(T7') $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$

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

Queen Mary
University of London

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.



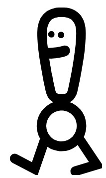
Examples of this later on!

Multi-Variable Theorems

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

(T12) $X + X + \ldots + X = X$

(T12') $X \cdot X \cdot \ldots \cdot X = X$



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

  - **DeMorgan's Theorems**

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

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

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

  - **Generalised DeMorgan's Theorem**

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

  - **Shannon's Expansion Theorems**

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

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

DeMorgan'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

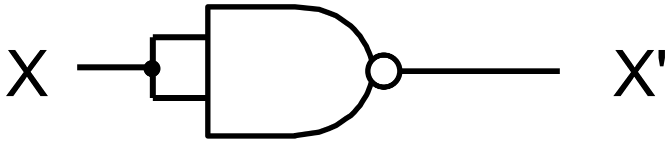
Like *Theorem 8*, **DeMorgan's law** may be used to convert between a sum of products and a product of sums.

Designing with only NAND gates

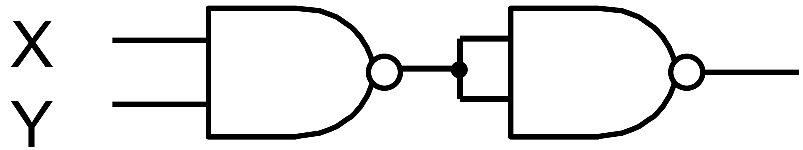
- ***Examples***:

This is only to simplify the implementation!

INV

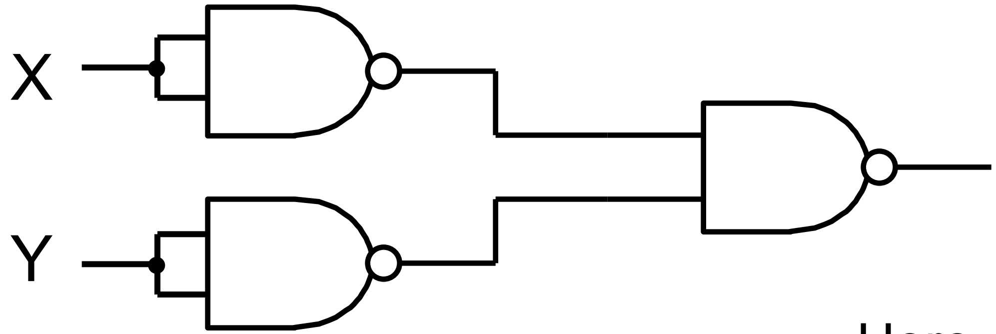


AND



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

OR

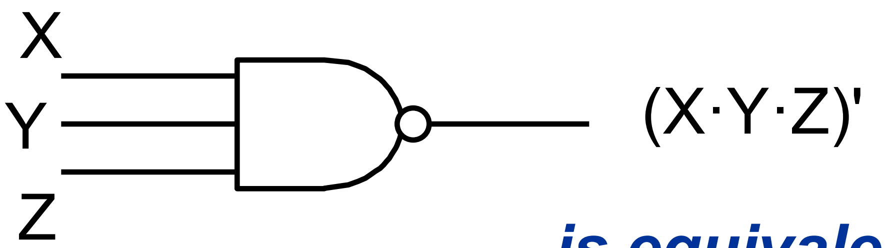


$(X' \cdot Y')' = X+Y$

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

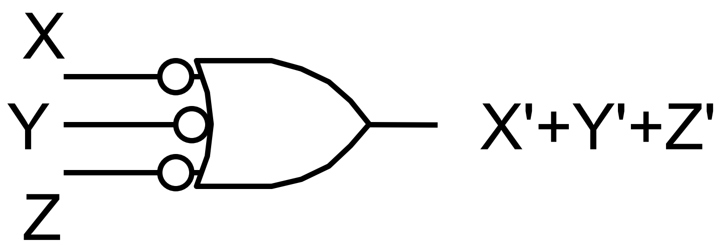
NAND Gates: 2 ways to view

- ***3-input NAND*** gate:



is equivalent to

- ***3 INVERTED input OR*** gate:



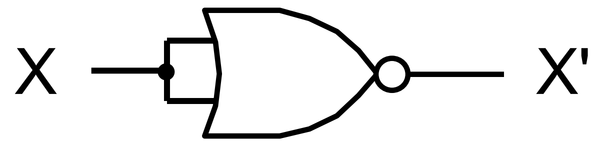
X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Designing with NOR gates

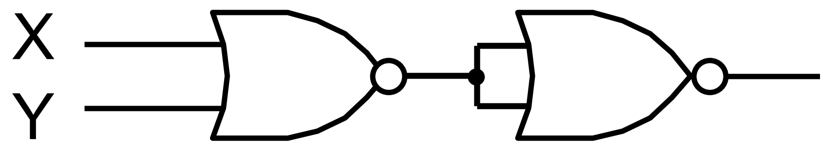
- ***Examples***:

This is only to simplify the implementation!

INV

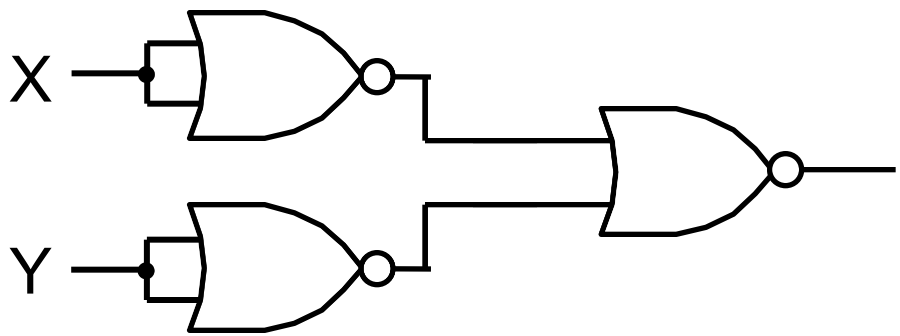


OR



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

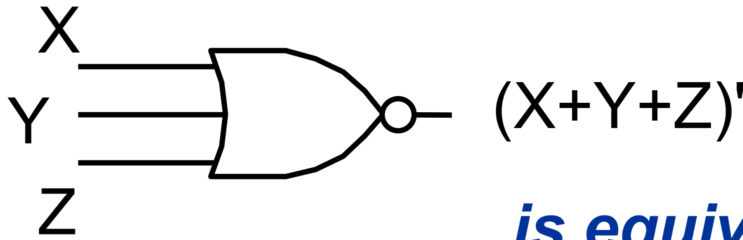
AND



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

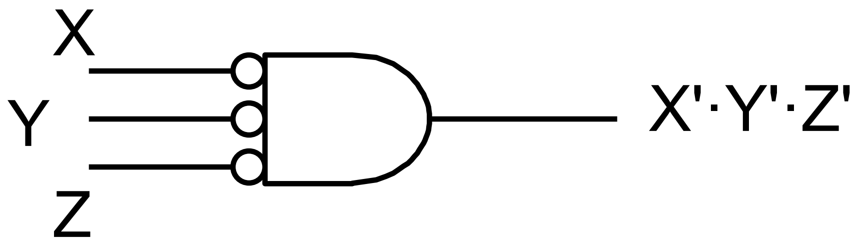
NOR Gates: 2 ways to view

- ***3-input NOR*** gate:



is equivalent to

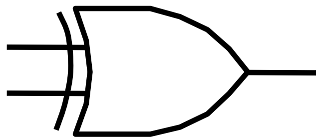
- ***3 INVERTED input AND*** gate:



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

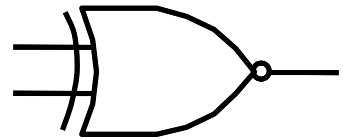
Exclusive OR Logic Gates: Properties

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



$Z = X \oplus Y$

XOR



$Z = \overline{X \oplus Y}$

XNOR

- ***Identities***:

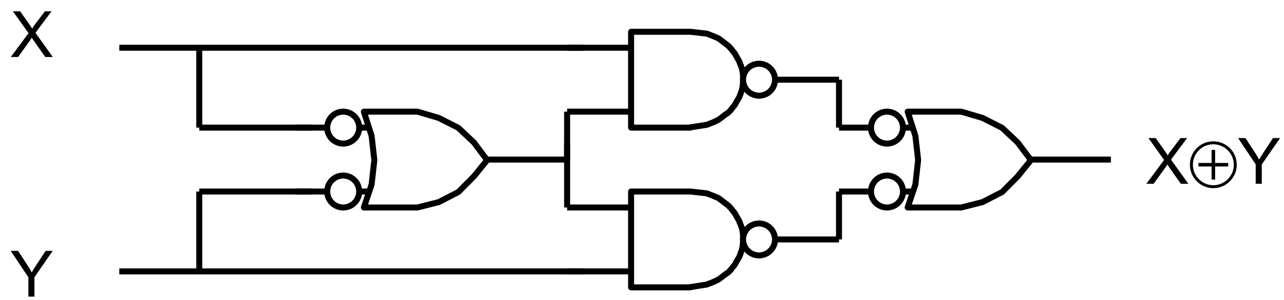
$X \oplus 0 = X$ $\qquad$ $X \oplus 1 = X'$ $\qquad$ $X \oplus X = 0$

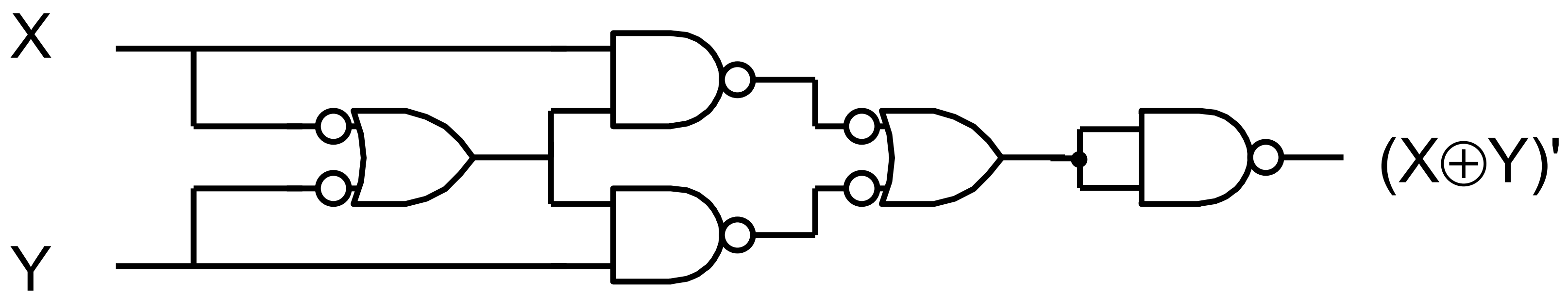
$X \oplus X' = 1$ $\qquad$ $X \oplus Y' = (X \oplus Y)'$ $\qquad$ $X' \oplus Y = (X \oplus Y)'$

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

XOR & XNOR Gates

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





Queen Mary University of London

Shannon's Expansion Theorem (1/2)

- *Shannon's Expansion Theorem (T15)* says:
  – Any Boolean switching function **F** can be written as:

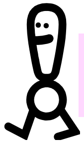
$$F(X_1, X_2, \ldots, X_n) = X_1 \cdot F(1, X_2, \ldots, X_n) + X_1' \cdot F(0, X_2, \ldots, X_n)$$

  – Using induction and letting $X_1 = 1$, the equation becomes an identity:

$$F(1, X_2, \ldots, X_n) = 1 \cdot F(1, X_2, \ldots, X_n) + \cancel{0 \cdot F(0, X_2, \ldots, X_n)}$$

  – Similarly, letting $X_1 = 0$, the equation also becomes an identity:

$$F(0, X_2, \ldots, X_n) = \cancel{0 \cdot F(1, X_2, \ldots, X_n)} + 1 \cdot F(0, X_2, \ldots, X_n)$$



This proves that any function can be expressed this way.

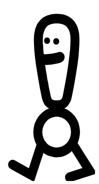
Shannon's Expansion Theorem (2/2)

- Repeated application of the expansion results in the canonical ***sum of products*** form.
  - For example, expanding a second variable yields:

$$\mathbf{F}(X_1, X_2, \ldots, X_n) = X_1X_2 \cdot F(1, 1, \ldots, X_n) + X_1X_2' \cdot F(1, 0, \ldots, X_n) + X_1'X_2 \cdot F(0, 1, \ldots, X_n) + X_1'X_2' \cdot F(0, 0, \ldots, X_n)$$

- Example: Consider **F = AB + CD + A'BE**,
  - **F(A=0) = (0)B + CD + (0)'BE = CD + BE**
  - **F(A=1) = (1)B + CD + (1)'BE = B + CD**

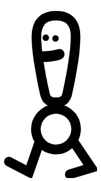
So **F = A'(CD + BE) + A(B + CD)**.



Terms can be introduced with this approach. If equation is reduced another way, it might lead to a simpler solution!

The Principle of Duality

- The **Principle of Duality**:
  - Used in *simplifying logic equations*.
  - Obtain the *dual* by interchanging OR and AND operations and replacing *1's* with *0's* and *vice versa*.
    - **Example**:
      - The dual of $\mathbf{F = X \cdot Y + Z \cdot W}$ is $\mathbf{F^D = (X + Y) \cdot (Z + W)}$.



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

Complements

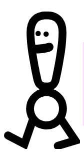
- How do we ***get the complement of a function***?
  - Can derive the complement of a function algebraically by applying DeMorgan's theorem.

    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 DeMorgan'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.

$\mathbf{F^D = (A' + B + C')(A' + B' + C)}$

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.

Show: $XY + X'Z + YZ = XY + X'Z$

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

$$F = XY + X'Z + YZ = XY + X'Z + YZ\overbrace{(X + X')}^{=1}$$

expand $\rightarrow$ $= XY + X'Z + XYZ + X'YZ$

reorder $\rightarrow$ $= XY + XYZ + X'Z + X'YZ$

factorise $\rightarrow$ $= XY(1 + \cancel{Z}) + X'Z(1 + \cancel{Y})$

=1

$$G = XY + X'Z$$

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

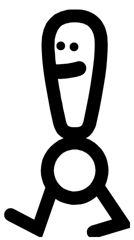
Example (1/3): Two Equations

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

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

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

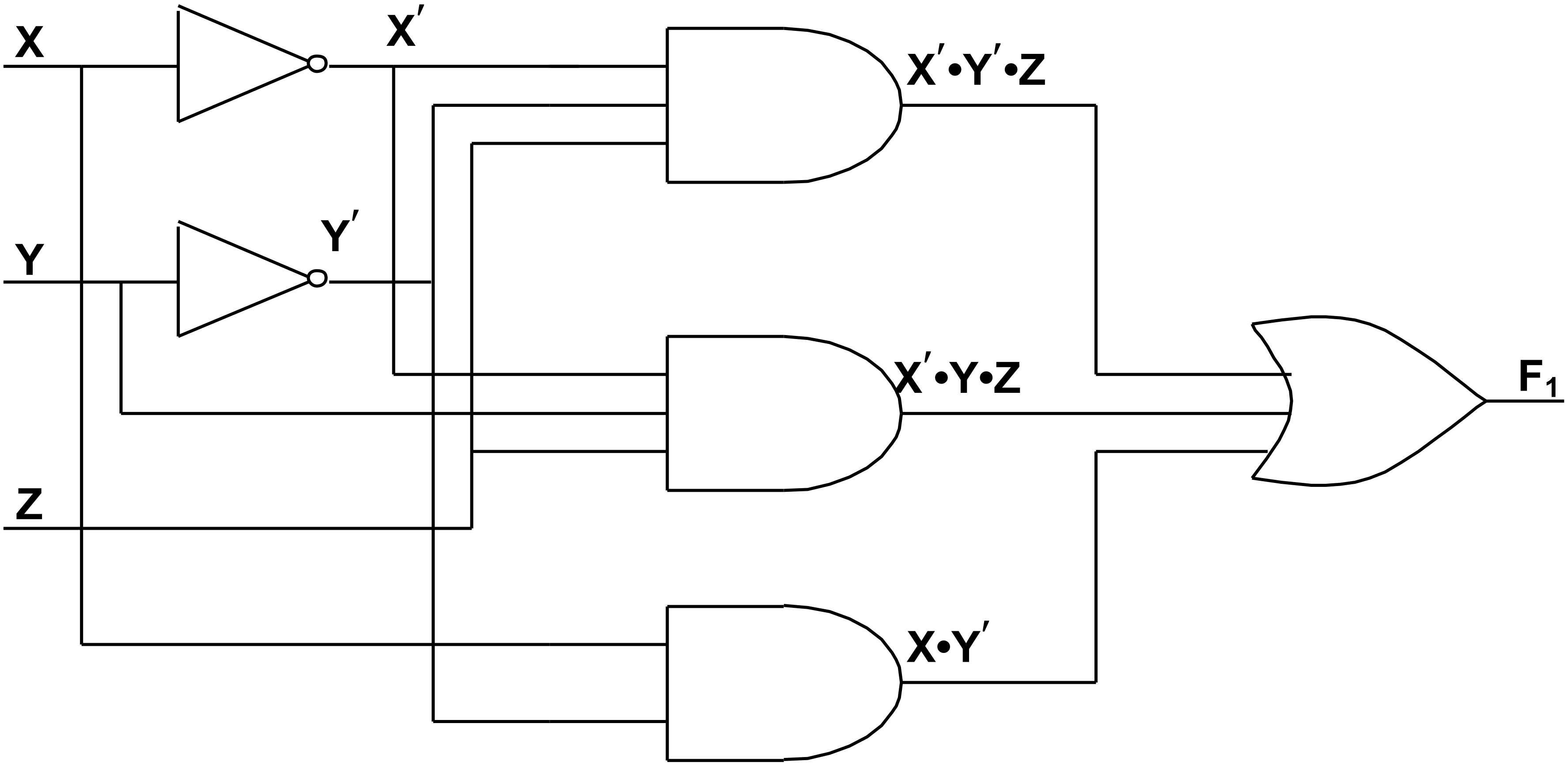
X	Y	Z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0



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

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:

$F = B'C(A' + A)$ **(using T8)**

$A' + A = 1$ **(using T5)**

therefore, $F = B'C$

Queen Mary
University of London

Example 2: Minimisation (1)

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

A'B'C = A'B'C + A'B'C　　　*(T3)* $X + X = X$

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

T10　　T10

(T10) $XY + XY' = X$

F = A'B' + A'C

Circuit Diagrams for these functions?

Example 2: Minimisation (2)

To be completed in class …

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

F = A'B' + A'C

Queen Mary
University of London

Law of Consensus: Example (1/2)

Problem: Suppose an alarm is to sound if the key is in the ignition and the door is open, ***or*** if the key is out of the ignition and the brake is off, ***or*** if the door is open and the brake is off. At all other times, the alarm must be silent.

Implementation: Derive a logic equation for the alarm going off.

Law of Consensus: Example (2/2)

To be completed in class …

Queen Mary
University of London

Example: Minimisation of F

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•X=X

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

EXPAND TERMS:

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

$= A'A' + A'B + A'C' + BA' + BB + BC' + A'BC'$

APPLY (T3') → $= A' + A'B + A'C' + BA' + B + BC' + A'BC'$

APPLY (T1') → $= A'(1 + B + C' + B + BC') + B(1 + C')$

APPLY (T2) → $= A'(1) + B(1)$

$= A' + B$

Queen Mary University of London

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***.

Queen Mary University of London

Logic Functions (2/2)

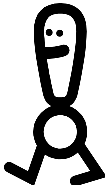
- Logic Functions' Representations (cont.):
  - ***Algebraic product of maxterms*** (*Canonical Product*):
    - ***maxterm***: is the sum of ***N*** distinctive logic variables or their complements e.g., ***X+Y+Z***;
    - the ***product of maxterms*** corresponds to the product of Truth Table rows 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	$X' \cdot Y' \cdot Z'$	m_0	$X+Y+Z$	M_0
0	0	1	$X' \cdot Y' \cdot Z$	m_1	$X+Y+Z'$	M_1
0	1	0	$X' \cdot Y \cdot Z'$	m_2	$X+Y'+Z$	M_2
0	1	1	$X' \cdot Y \cdot Z$	m_3	$X+Y'+Z'$	M_3
1	0	0	$X \cdot Y' \cdot Z'$	m_4	$X'+Y+Z$	M_4
1	0	1	$X \cdot Y' \cdot Z$	m_5	$X'+Y+Z'$	M_5
1	1	0	$X \cdot Y \cdot Z'$	m_6	$X'+Y'+Z$	M_6
1	1	1	$X \cdot Y \cdot Z$	m_7	$X'+Y'+Z'$	M_7

Maxterms

Minterms



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

Queen Mary University of London

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	F'	
0	0	0	1	0	$\mathbf{m_0}$
0	0	1	0	1	$\mathbf{m_1}$
0	1	0	1	0	$\mathbf{m_2}$
0	1	1	0	1	$\mathbf{m_3}$
1	0	0	0	1	$\mathbf{m_4}$
1	0	1	1	0	$\mathbf{m_5}$
1	1	0	0	1	$\mathbf{m_6}$
1	1	1	1	0	$\mathbf{m_7}$

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

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

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

$F'' = (\mathbf{m_1} + \mathbf{m_3} + \mathbf{m_4} + \mathbf{m_6})'$

$F'' = \mathbf{m_1'} \cdot \mathbf{m_3'} \cdot \mathbf{m_4'} \cdot \mathbf{m_6'}$

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

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_0
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	0	m_3
1	0	0	1	m_4
1	0	1	1	m_5
1	1	0	0	m_6
1	1	1	0	m_7

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

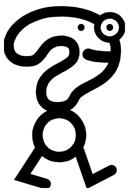
So:

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

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

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

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



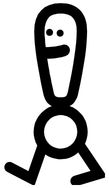
Which equation is easier to build?

$$F = \underbrace{X' \cdot Y' \cdot Z' + X' \cdot Y' \cdot Z}_{T10} + X' \cdot Y \cdot Z' + \underbrace{X \cdot Y' \cdot Z' + X \cdot Y' \cdot Z}_{T10}$$

$$F = \underbrace{X' \cdot Y' + X' \cdot Y \cdot Z' + X \cdot Y'}_{T10}$$

$$F = \underbrace{Y' + X' \cdot Y \cdot Z'}_{T^* \text{ (Absorption Theorem)}}$$

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



Simplification of **F** using algebraic manipulation.

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*).

F(A,B,C) = A'B' + BC

= A'B'(C + C') + BC(A + A') *by (T5), (T1')*

= A'B'C + A'B'C' + ABC + A'BC *by (T8), (T6), (T6')*

To get the Σm expression, replace all *normal literals with a 1* and all *primed literals with a 0*: (***Example***: A'B'C ⇒ A' = 0, B' = 0, C = 1 ⇒ 001).

F(*A*,*B*,*C*) = Σm(001, 000, 111, 011)

Now convert to decimal (and put them in order), *F*(*A*,*B*,*C*) = Σm(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*).

F(A,B,C)
= A'B' + BC
= (A'B' + B)(A'B' + C) *by variable substitution, (T8')*
= (A' + B)(B' + B)(A' + C)(B' + C) *by (T8')*
= (A' + B + CC')(A' + C + BB')(B' + C + AA') *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) *by (T3'), (T6)*

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

F(*A*,*B*,*C*) = ΠM (100, 101, 110, 010)

Now convert to decimal (and order them): *F*(*A*,*B*,*C*) = ΠM (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')(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**.