Boolean Algebra

George Boole (1815-1864) English mathematician and philosopher.

- Defined on the set B = {0,1}
- Binary operators: AND, OR { • , + }
- Unary operation: Complement (NOT) {'}
  Another symbol for the NOT operator: $\overline{a}$

a • b (AND)

a \ b	0	1
0	0	0
1	0	1

a + b (OR)

a \ b	0	1
0	0	1
1	1	1

Complement

a	a' ($\overline{a}$)
0	1
1	0

- **Axioms (Laws):**

For any a, b ∈ B

•: AND +: OR

1. Closure:	a + b ∈ B	a • b ∈ B
2. Commutative:	a + b = b + a	a • b = b • a
3. Associative:	a + (b + c) = (a + b) + c	a • (b • c) = (a • b) • c
4. Identity:	a + 0 = a	a • 1 = a
5. Distributive:	a + (b • c) = (a + b) • (a + c)	a • (b + c) = (a • b) + (a • c)
6. Inverse:	$a + \overline{a} = 1$	$a \bullet \overline{a} = 0$

- **Order of operations (operator precedence)** from highest to lowest precedence:
  1. Parentheses, 2. NOT (Complementation) (Negation), 3. AND, 4. OR

- **The Duality principle:**

To obtain the **dual** of a logic expression: Replace • by +, + by •, 0 by 1, and 1 by 0, but do not change the variables.

a + b + 0 ... ⇔ a • b • 1 ...

Example: The dual of the expression a + a·b is a·(a+b) .

Principle: Duals of all proven theorems are also theorems.

Given a Boolean algebra identity, another identity can be obtained by taking the dual of both sides of the identity.

Note that in the previous slide, axioms were presented with their duals (in two columns).

Example:

Absorption theorem (given in the next slide):

If we can prove the theorem a + a·b = a, then its dual a·(a+b) = a is also true.

General Duality Principle:

$f(X1, X2, ..., Xn, 0, 1, +, \bullet) \Leftrightarrow f^D(X1, X2, ..., Xn, 1, 0, \bullet, +)$

- Duality establishes a relationship between proofs of theorems.
- Duals are not equal.

Theorems:

These theorems are derived from the operations and axioms of Boolean algebra.
They can be proven using the axioms.

1. Annihilator (or Dominance):
   $a + 1 = 1$ $\qquad$ $a \bullet 0 = 0$
2. Involution: $(a')' = a$ or $\overline{\overline{a}} = a$
3. Idempotency:
   $a+a+a+....+a = a$ $\qquad$ $a \bullet a \bullet a \bullet \ldots \bullet a = a$
4. Absorption: (Proof in 2.4)
   $a + a \cdot b = a$ $\qquad$ $a \cdot (a+b) = a$
5. De Morgan's Theorem: *Augustus De Morgan, British mathematician and logician (1806 - 1871)*
   $\overline{(a + b)} = \overline{a} \bullet \overline{b}$ $\qquad$ $\overline{(a \bullet b)} = \overline{a} + \overline{b}$
5. General form of De Morgan's Theorem:
   $\overline{f\ (X1, X2, ..., Xn, 0, 1, +, \bullet)} = g(\overline{X1}, \overline{X2}, ..., \overline{Xn}, 1, 0, \bullet, +)$
   - It establishes a relationship between AND and OR ($\bullet$ and $+$).

Proving the theorems:

a) Using Axioms

Example:

Theorem: $X \cdot Y + X \cdot \overline{Y} = X$

Proof:

Distributive	$X \cdot Y + X \cdot \overline{Y}$	$= X \cdot (Y + \overline{Y})$
Inverse (Complement)	$X \cdot (Y + \overline{Y})$	$= X \cdot (1)$
Identity	$X \cdot (1)$	$= X$ ✓

Example:

Theorem: $X + X \bullet Y = X$ (*Absorption*)

Proof:

Identity	$X + X \bullet Y$	$= X \bullet 1 + X \bullet Y$
Distributive	$X \bullet 1 + X \bullet Y$	$= X \bullet (1 + Y)$
Annihilator	$X \bullet (1 + Y)$	$= X \bullet (1)$
Identity	$X \bullet (1)$	$= X$ ✓

Proving the theorems: **b) Using truth tables**

Note that to denote negation (NOT), we can also use the notation $\overline{A}$.

De Morgan:

$\overline{(X + Y)} = \overline{X} \bullet \overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{(X + Y)}$	$\overline{X} \bullet \overline{Y}$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

$\overline{(X \bullet Y)} = \overline{X} + \overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{(X \bullet Y)}$	$\overline{X} + \overline{Y}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Although truth tables may contain many rows, computer programs can fill in and compare these tables very quickly.

Minimizing logic expressions using axioms and theorems:

Minimizing a logic expression means:

- finding the shortest expression
- with the fewest operations and variables
- that generates the same output values as the original expression.

Example:

$$\begin{aligned}
Z(A,B,C) &= \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C \quad \text{Original expression}\\
&= \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C + A\,B\,C\\
&= \overline{A}\,B\,C + A\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C\\
&= (\overline{A} + A)\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C\\
&= (1)\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C\\
&= B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C + A\,B\,C\\
&= B\,C + A\,\overline{B}\,C + A\,B\,C + A\,B\,\overline{C} + A\,B\,C\\
&= B\,C + A\,(\overline{B} + B)\,C + A\,B\,\overline{C} + A\,B\,C\\
&= B\,C + A\,(1)\,C + A\,B\,\overline{C} + A\,B\,C\\
&= B\,C + A\,C + A\,B\,(\overline{C} + C)\\
&= B\,C + A\,C + A\,B\,(1)\\
&= B\,C + A\,C + A\,B \quad \text{Minimized expression}
\end{aligned}$$

Logic (Boolean) Expressions

A logic expression is a finite combination of variables, constants, and operators that are well-formed according to the rules.

It is represented as E(X), where X= $(x_1, x_2, ... x_n) \in B^n$ and each $x_i \in \{0,1\}$.

B^n is the set of vectors with n binary variables.

Examples:

$$E(x_1, x_2, x_3, x_4) = x_2\overline{x_3} + x_1x_4 + \overline{x_1}x_2$$

$$E(a, b, c) = a\bar{c} + ab$$

$$E(a, b, c, d) = (a + b + \bar{c})(\bar{a} + d)(b + \bar{d})$$

Literal:

In a Boolean expression, each separate occurrence of a variable, either in normal (uncomplemented) or inverse (complemented) form, is a **literal**.

For example, the expression $E(a, b, c) = a\bar{c} + ab$ has three variables (a, b, c) and contains four literals $(a\bar{c} + ab)$.

However, two of the literals are identical (a appears twice).

If E_1 and E_2 are logic expressions, then $\overline{E_1}$, $\overline{E_2}$, $E_1 + E_2$, $E_1 \cdot E_2$, and all possible combinations are also logic expressions.

Normal forms of logic expressions:

Each logic expression can be written in two special forms.

1. Disjunctive normal form (DNF): $\Sigma\Pi$

Logic sum of logic products (SOP). OR of ANDs.

Example: $b\bar{c} + ad + \bar{a}b$

The OR (logic sum) operation is also called *logical disjunction*.

2. Conjunctive normal form (CNF): $\Pi\Sigma$

Logic product of logic sums (POS). AND of ORs.

Example: $(a + b + \bar{c})(a + d)(\bar{a} + b)$

The AND (logic product) operation is also called *logical conjunction*.

Any logic expression can be written in CNF (POS form) and DNF (SOP form).

Any expression in CNF can be converted to DNF and vice versa ($\Sigma\Pi \leftrightarrow \Pi\Sigma$).

Conventions:

- Write the literals in the terms in alphabetical order: $ab\bar{c}$ (**not** $a\bar{c}b$)
- Write the expression starting with the term that has the fewest (or the most) literals, and then proceeding in ascending (or descending) order of literals per term, such as: $ab + a\bar{c}d + ab\bar{c}d$ or $ab\bar{c}d + a\bar{c}d + ab$

Value of a logic expression:

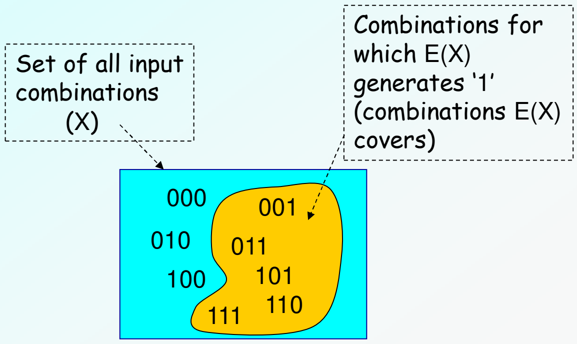
An expression, E(X), generates for each combination of the input vector $X=(x_1, ... x_n)$, a value from the set of B={0,1}.

These values constitute the truth table for the expression.

Example: $E(X) = x_1x_2 + x_3$

Truth table for the expression

X= x_1	x_2	x_3	E(X)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Applying axioms and theorems to expressions

The axioms and theorems of Boolean algebra defined for binary values are also valid for expressions due to the closure property.

Remember: According to the closure property, the value generated by an expression E is a binary value, i.e., $E(X) \in B = \{0,1\}$

Examples:

$$E(a, b, c, d) = b\bar{c} + ad + \bar{a}b$$

Identity: $E(X) + 0 = E(X)$ $\quad E(X) \cdot 1 = E(X)$

$$E(a, b, c, d) + 0 = (b\bar{c} + ad + \bar{a}b) + 0 = b\bar{c} + ad + (\bar{a} + 0)(b + 0)$$
$$= b\bar{c} + ad + \bar{a}b = E(a, b, c, d)$$

$$E(a, b, c, d) \cdot 1 = (b\bar{c} + ad + \bar{a}b) \cdot 1 = b\bar{c}1 + ad1 + \bar{a}b1$$
$$= b\bar{c} + ad + \bar{a}b = E(a, b, c, d)$$

Annihilator (or Dominance): $E(X) + 1 = 1$ $\quad E(X) \cdot 0 = 0$

$$E(a, b, c, d) + 1 = (b\bar{c} + ad + \bar{a}b) + 1 = 1$$

$$E(a, b, c, d) \cdot 0 = (b\bar{c} + ad + \bar{a}b) \cdot 0 = 0$$

"Order Relation" between binary vectors:

To explain some properties of logic expressions, we can define and use the following two order relations "<" and "≤":

1. An order relation "<" between elements of set B = {0,1}: 0 < 1
   - Read as "0 precedes 1" or "0 is smaller than 1".
2. Another order relation " ≤ " between $X \in B^n$ vectors can be defined as follows:
   - If each component of X1 is smaller than (precedes) or equal to the component of vector X2 in the corresponding position, then X1 ≤ X2.

Example:

X1=1001 , X2 = 1101

X1 ≤ X2.

The order relation "≤" **may not** exist between all vectors.

Example: X1 = 0011 , X2 = 1001

There is no order relation between X1 and X2 (Neither X1 ≤ X2 nor X2 ≤ X1 is true).

Order relation between expressions:

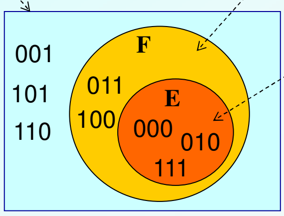
E(X) ≤ F(X) denotes that values of E generated by combinations of input X are always smaller than (or equal to) all values of F generated by the same input combinations.

Example :

x_1	x_2	x_3	E(X)		F(X)
0	0	0	1	=	1
0	0	1	0	=	0
0	1	0	1	=	1
0	1	1	0	<	1
1	0	0	0	<	1
1	0	1	0	=	0
1	1	0	0	=	0
1	1	1	1	=	1

Set of all input combinations (X)

Combinations for which F(X) generates '1' (combinations F(X) covers)



Combinations for which E(X) generates '1' (combinations E(X) covers)

For each input combination where E(X) generates "1", F(X) also generates "1".

This is a special case. The order relation ≤ may not exist between all expressions (see slide 2.15).

If E(X) ≤ F(X),
E(X) implies F(X), E(X)⇒F(X),
F(X) covers E(X).

Using the order relation to show absorption properties of expressions

Remember: The theorems given for binary values on slide 2.3 are also valid for expressions due to the closure property.

Consequently, the absorption theorem given for binary values ($a + a \cdot b = a$ and $a \cdot (a+b) = a$) is also valid for expressions because of the closure property.

However, the order relation ≤ makes it easier to grasp the absorption properties of expressions.

Since absorption is an important theorem used to simplify expressions, we will again illustrate this theorem using the order relation ≤.

We will consider two cases.

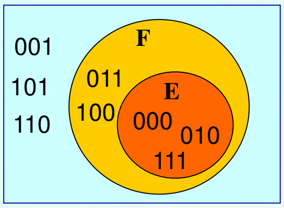
- *Case A*: Special case: E(X) ≤ F(X) (as in the example on slide 2.12)
- *Case B*: General case: The order relation ≤ does not exist between E(X) and F(X)

Using the order relation to show absorption properties of expressions (cont'd)

CASE A: Special case: E(X) ≤ F(X)

Absorptions:

1. E(X) + F(X) = F(X)
2. E(X) • F(X) = E(X)



Example:

Consider the example given on slide 2.12.

Absorption properties (1 and 2) can be seen on the diagram on the right.

Expressions for the functions on slide 2.12:

$$\mathrm{E}(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 , \qquad \mathrm{F}(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_3} + x_2 \cdot x_3$$

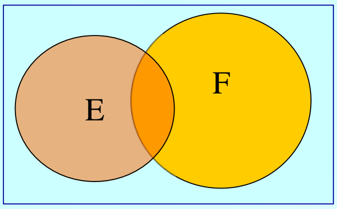
Absorptions:

1. $\mathrm{E(X)} + \mathrm{F(X)} = \overbrace{\overline{x_1} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3}^{\mathrm{E(X)}} + \overbrace{\overline{x_1} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_3} + x_2 \cdot x_3}^{\mathrm{F(X)}}$
   $= \overline{x_1} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_3} + x_2 \cdot x_3 = \mathrm{F(X)}$

2. $\mathrm{E(X)} \bullet \mathrm{F(X)} = (\overline{x_1} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3) \cdot (\overline{x_1} \cdot \overline{x_3} + \overline{x_2} \cdot \overline{x_3} + x_2 \cdot x_3)$
   $= \overline{x_1} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 = \mathrm{E(X)}$

Using the order relation to show absorption properties of expressions (cont'd)

Case B: General case: The order relation ≤ does not exist between E(X) and F(X)



The following inequalities are always true for any E and F (as can be seen on the diagram on the left):

$$E{\cdot}F \le E \le E{+}F$$

$$E{\cdot}F \le F \le E{+}F$$

Absorption properties of expressions follow directly from these inequalities (we can generalize the absorption theorems for Boolean variables and apply them to expressions due to the closure property):

Absorption Properties of expressions:

$\boxed{E + E \cdot F = E}$ dual $\boxed{E(E+F) = E}$

Proof: $E(E+F) = EE+EF = E+EF = E(1+F) = E$

$\boxed{E + \bar{E} \cdot F = E + F}$ dual $\boxed{E(\bar{E}+F) = E \cdot F}$

Proof: $E + \bar{E} \cdot F = (E+\bar{E})(E+F) = 1(E+F) = E+F$

These properties are used to minimize (simplify) logic expressions.

Example (general case):

E(a,b,c,d) = abc' , F(a,b,c,d) = bd ➡ The order relation (≤) does not exist between the E and F expressions given in this example.

E·F = abc'd E+F = abc' + bd

a b c d	E	F	E·F	E+F
0 0 0 0	0	0	0	0
0 0 0 1	0	0	0	0
0 0 1 0	0	0	0	0
0 0 1 1	0	0	0	0
0 1 0 0	0	0	0	0
0 1 0 1	0	1	0	1
0 1 1 0	0	0	0	0
0 1 1 1	0	1	0	1
1 0 0 0	0	0	0	0
1 0 0 1	0	0	0	0
1 0 1 0	0	0	0	0
1 0 1 1	0	0	0	0
1 1 0 0	1	0	0	1
1 1 0 1	1	1	1	1
1 1 1 0	0	0	0	0
1 1 1 1	0	1	0	1

From the truth table, we observe

E·F ≤ E and E·F ≤ F.

We can check the absorption properties:

E·F + E $\stackrel{?}{=}$ E

abc'd + abc' $\stackrel{?}{=}$ abc' ✓

and

E·F + F $\stackrel{?}{=}$ F

abc'd + bd $\stackrel{?}{=}$ bd ✓

Again, from the truth table, we observe

E ≤ E+F and F ≤ E+F.

We can check the absorption properties:

E·(E + F) $\stackrel{?}{=}$ E

abc' (abc' + bd) $\stackrel{?}{=}$ abc' ✓

and

F·(E + F) $\stackrel{?}{=}$ F

bd (abc' + bd) $\stackrel{?}{=}$ bd ✓

We have thus shown the absorption properties using a truth table instead of axioms.

The Consensus Theorem (SOP Form)

Assume that E_1 and E_2 are two expressions that do not contain the literal x_1 :
$E_1(x_2, \ldots x_n)$ and $E_2(x_2, \ldots x_m)$

We can create a new expression by multiplying one term by x_1 and the other one by the complement of x_1 ($\overline{x_1}$).

$$E = \boldsymbol{x_1} E_1 + \overline{\boldsymbol{x_1}} E_2$$

Here, x_1 is called a **biform** variable because it appears both positively (as itself: $\boldsymbol{x_1}$) and negatively (as complement: $\overline{\boldsymbol{x_1}}$) in the expression.

Examples: a) $\boldsymbol{x_1}(x_2+\overline{x_3}) + \overline{\boldsymbol{x_1}}(x_3+x_4)$,
b) $\boldsymbol{x_1} x_2 \overline{x_3} + \overline{\boldsymbol{x_1}} x_4 + x_5$

- Given a pair of product terms that include a biform variable, **the consensus term** (SOP) is formed by multiplying (ANDing) the two original terms together, leaving out the selected biform variable and its complement.
- **E_1E_2** is **the consensus term** of $x_1E_1 + \overline{x_1}E_2$ with respect to variable x_1.

Example: Consensus of $\boldsymbol{abc} + \boldsymbol{\bar{a}cd}$ (with respect to a) is $bccd = \boldsymbol{bcd}$.

Theorem: The consensus term is redundant and can be eliminated.

$$\boxed{x_1E_1 + \overline{x_1}E_2 + \boldsymbol{E_1E_2} = x_1E_1 + \overline{x_1}E_2}$$

The Consensus Theorem (POS Form)

According to the duality principle, the consensus theorem is also valid for expressions written in product-of-sums (POS) form.

Assume that E_1 and E_2 are two expressions that do not contain the literal x_1:
$E_1(x_2, \ldots x_n)$ and $E_2(x_2, \ldots x_m)$

We can create a new expression by adding x_1 to one term and the complement of x_1 to the other one.

$$E = (\boldsymbol{x_1}+E_1)(\overline{\boldsymbol{x_1}}+E_2)$$

Here, x_1 is a **biform** variable.

Examples: a) $(\boldsymbol{x_1} + x_2 + \overline{x_3})(\overline{\boldsymbol{x_1}} + x_3 + x_4)$,
b) $(\boldsymbol{x_1} + x_2\overline{x_3})(\overline{\boldsymbol{x_1}} + x_3\, x_4)$

- Given a pair of sums that include a biform variable, **the consensus term** (POS) is formed by adding (ORing) the two original terms together, leaving out the selected biform variable and its complement.
- **$E_1 + E_2$** is **the consensus term** of $(x_1+E_1)(\overline{x_1}+E_2)$ with respect to the variable x_1.

Example: Consensus of $(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})(\boldsymbol{\bar{a}}+\boldsymbol{c}+\boldsymbol{d})$ is: $b+c+c+d = \boldsymbol{b}+\boldsymbol{c}+\boldsymbol{d}$.

Theorem: The consensus term is redundant and can be eliminated.

$$\boxed{(x_1+E_1)(\overline{x_1}+E_2)(\boldsymbol{E_1}+\boldsymbol{E_2}) = (x_1+E_1)(\overline{x_1}+E_2)}$$

Example: Minimization of a logic expression using the consensus theorem

$$F(A,B,C) = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B\,C + A\,B\,\bar{C}$$

Consensus (with respect to C) is added

$$= \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B\,C + A\,B\,\bar{C} + A\,B$$

Absorption

$$= \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,\bar{B}\,C + A\,\cancel{B}\,C + A\,\cancel{B}\,\bar{C} + A\,B$$

$$= \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B$$

Consensus (with respect to B) is added

$$= \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + \bar{A}\,C + A\,\bar{B}\,C + A\,B$$

Absorption

$$= \bar{A}\,\cancel{\bar{B}}\,C + \bar{A}\,\cancel{B}\,C + \bar{A}\,C + A\,\bar{B}\,C + A\,B$$

$$= \bar{A}\,C + A\,\bar{B}\,C + A\,B$$

Consensus (with respect to B) is added

Absorption

$$= \bar{A}\,C + A\,\cancel{\bar{B}}\,C + A\,B + A\,C$$

$$= \bar{A}\,C + A\,B + A\,C$$

Consensus (with respect to A) is added

Absorption

$$= \cancel{\bar{A}\,C} + A\,B + \cancel{A\,C} + C$$

$$= A\,B + C$$

Logic (Boolean) Functions

Logic functions are defined on the input set B^n (vectors with n binary variables).
There are three types of logic functions:

1. Simple (basic) functions: Multiple inputs, single output

$y = f(X): B^n \rightarrow B$

$\forall X_i \in B^n \; ; \exists! \; y_i \in B \; ; y=f(X)$

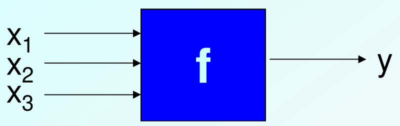
"For any X, there is exactly one y such that f(X) = y."

Example:

$y = f(X)$

$X \in B^3$

$y \in B$



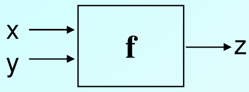
The truth table for y = f(X)

$x_1\ x_2\ x_3$	y
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

The number of basic logic functions for n binary variables:

There are $2^{(2^n)}$ possible basic logic functions for n binary variables (inputs).

For example,
there are 16 possible basic logic functions for 2 binary variables (inputs).



Truth table for 16 possible basic logic functions (F0–F15) for 2 binary variables:

Inputs		Functions															
X	Y	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
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

0
X **AND** Y
X
Y
X **XOR** Y
X **OR** Y
X **NOR** Y
(X **OR** Y)'
X = Y
Y'
X'
X **NAND** Y
(X **AND** Y)'
1

2. General functions: Multiple inputs, multiple outputs

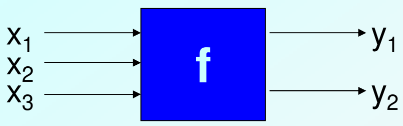
$Y = f(X): B^n \rightarrow B^m$, $X=(x_1, ... x_n)$, $Y=(y_1, ... y_m)$,

Example:

$Y = f(X)$

$X \in B^3$

$Y \in B^2$



The truth table for Y = f(X)

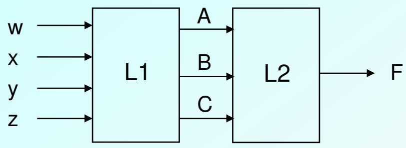
x_1 x_2 x_3	y_1 y_2
0 0 0	1 1
0 0 1	1 0
0 1 0	0 0
0 1 1	0 0
1 0 0	1 1
1 0 1	0 1
1 1 0	0 1
1 1 1	1 0

3. Incompletely specified functions:

A large digital system usually consists of many subcircuits.

Example:

In the following example, the output of logic circuit L1 drives the output of logic circuit L2.



- Let us assume that the output of L1 does not generate all possible combinations of values for A, B, and C.
- In particular, we will assume that there are no combinations of values for w, x, y, and z, which cause A, B, and C to assume values of 001 and 110.
- In other words, L1 never generates the output combinations 001 and 110.
- Hence, when we design L2, it is not necessary to specify values of F for ABC = 001 or 110 because these combinations of values can never occur as inputs to L2.

3. Incompletely specified functions (cont'd):

- For example, F might be specified by the following table:

A	B	C	F	
0	0	0	1	
0	0	1	X	Don't care
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	0	
1	1	0	X	Don't care
1	1	1	1	

These input combinations can never occur.

- The X's in the table indicate that we do not care whether the value of 0 or 1 is assigned to F for the combinations ABC = 001 or 110.
- In this example, we do not care what the value of F is because these input combinations never occur anyway.
- The function F is then **incompletely specified**.
- The terms $\bar{A}\,\bar{B}\,C$ and $A\,B\,\bar{C}$ are referred to as "**don't care**" terms because we do not care whether they are present in the function or not.

3. Incompletely specified functions (cont'd):

- When we realize the function, we must specify values for the don't cares.
- It is desirable to choose values which will help simplify the function.
- For the example on Slide 2.24:
  - If we assign the value 0 to both X's, then
    $\mathbf{F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,B\,C + A\,B\,C = \bar{A}\,\bar{B}\,\bar{C} + B\,C}$
  - If we assign 0 to the first X and 1 to the second, then
    $\mathbf{F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C} + A\,B\,C = \bar{A}\,\bar{B}\,\bar{C} + B\,C + A\,B}$
  - If we assign 1 to the first X and 0 to the second, then
    $\mathbf{F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,B\,C = \bar{A}\,\bar{B} + B\,C}$
  - If we assign 1 to both X's, then
    $\mathbf{F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,B\,\bar{C} + A\,B\,C = \bar{A}\,\bar{B} + B\,C + A\,B}$

Finding expression from truth table
See 2.32 canonical forms

The third choice of values leads to the simplest solution.

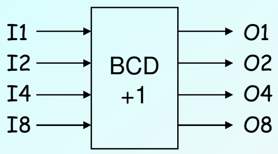
- In Section 4, we will see the selection of unspecified (don't care) values and the simplification of incompletely specified functions in detail.
- Incompletely specified functions can arise in the following cases:
  - Certain combinations of inputs cannot occur (as in the above example).
  - All input combinations may occur, but the function output is used in such a way that we do not care whether it is 0 or 1 for certain input combinations.

3. Incompletely specified functions (cont'd):

Example: A function that increments BCD numbers

We will create a general function to increment BCD numbers given on slide 1.9.

This function will have 4 inputs and 4 outputs because BCD numbers are 4-bit patterns.



Since BCD numbers are represented using binary code words between 0000-1001, combinations between 1010-1111 will never appear as inputs to this function.

Even if these values are applied to the inputs of the function, we do not care what the output values are.

I8	I4	I2	I1	O8	O4	O2	O1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

For these input combinations, the output values of the circuit (function) are not specified.

An X or Φ represents a don't care.

Representation of Logic (Boolean) Functions

There are different ways of representing (expressing) the same logic function.

When designing logic circuits, we choose the most suitable representation.

Truth Table Representation :

We write the output values for all possible input combinations (variables) in a table.

We usually write the input columns so that they follow the order of binary counting.

Input variables are encoded as binary numbers.

(See examples in 2.20-2.22)

Numbered (Indexed) Representation:

Input variables are encoded as binary numbers.

We assign a decimal number for each input combination based on its binary value.

To represent the function, we list the decimal number of each input combination for which the function generates "1" (or logic "0" or "Φ").

Example: Indexed representation of a completely specified basic logic function:

Truth Table:

Row Num.	Input x_1 x_2	Output y
0	0 0	1
1	0 1	0
2	1 0	1
3	1 1	0

Numbered (Indexed) Representation:

$y = f(x_1,x_2) = \cup_1(0,2)$

$\cup$ denotes "union" or "set of".

$\cup_1$ denotes "set of 1-generating points".

The order of the variables (x_1,x_2) is important. It must be the same as in the truth table. Otherwise, the decimal numbers identifying the combinations will change.

The same function; only the order of the variables (x_2,x_1) is changed.

Row Num.	Input x_2 x_1	Output y
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	0

$y = f(\mathbf{x_2},\mathbf{x_1}) = \cup_1(0,1)$

We can represent the same function with "0"-generating combinations. $y = f(x_1,x_2) = \cup_0(1,3)$

$$\boxed{y = f(x_1,x_2) = \cup_1(0,2) \quad = \quad f(x_2,x_1) = \cup_1(0,1) \quad = \quad f(x_1,x_2) = \cup_0(1,3)}$$

Example: Representation of a completely specified general logic function :

We apply the numbered representation to all outputs.

Truth Table:

Num.	x_1	x_2	y_1	y_2
0	0	0	1	1
1	0	1	0	1
2	1	0	1	0
3	1	1	0	0

Numbered (Indexed) Representation:

$y_1 = f(x_1,x_2) = \cup_1(0,2)$

$y_2 = f(x_1,x_2) = \cup_1(0,1)$

Representation of the same function with "0"-generating combinations:

$y_1 = f(x_1,x_2) = \cup_0(1,3)$

$y_2 = f(x_1,x_2) = \cup_0(2,3)$

Example: Representation of an incompletely specified general logic function:

In this case, writing only 1-generating or only 0-generating input combinations is not sufficient.

We must write at least two of the three groups (1-generating, 0-generating, don't care).

Truth Table:

N	x_1	x_2	y_1	y_2
0	0	0	1	1
1	0	1	0	Φ
2	1	0	Φ	0
3	1	1	0	Φ

Numbered (Indexed) Representations:

$y_1 = f(x_1,x_2) = \cup_1(0) + \cup_0(1,3)$

or $y_1 = f(x_1,x_2) = \cup_1(0) + \cup_\Phi(2)$

or $y_1 = f(x_1,x_2) = \cup_0(1,3) + \cup_\Phi(2)$

$y_2 = f(x_1,x_2) = \cup_1(0) + \cup_0(2)$

or $y_2 = f(x_1,x_2) = \cup_1(0) + \cup_\Phi(1,3)$

or $y_2 = f(x_1,x_2) = \cup_0(2) + \cup_\Phi(1,3)$

Algebraic Representation (Expressions) and Canonical Forms

The word description of a real-world logic design problem can be translated into a truth table.

Example: Assume that input variable **A** represents the phrase "the car door is open" and **B** represents "the key is inserted",
then the truth table can specify the action **Z** to be taken, where Z=1 means that the alarm sounds.

Num.	A	B	Z
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Truth tables of real-world logic design problems are more complicated.

To handle a logic design problem and implement the solution using logic gates, we need to obtain an algebraic **expression** for the output function. Z = f (A,B)

Logic expressions of the Boolean functions can be obtained in **canonical forms** from their truth tables.

There are two types of canonical forms:

- 1st canonical form: SOP ($\Sigma\Pi$) form. Example: $a\,b\,c + a\,\bar{b}\,c$
  The sum of products, each of which corresponds to a "1"-generating combination.
- 2nd canonical form: POS ($\Pi\Sigma$) form. Example: $(a + b + \bar{c})(a + \bar{b} + \bar{c})$
  The product of sums, each of which corresponds to a "0"-generating combination.

1st Canonical Form: Sum of Products

- The 1st canonical form is the sum of special products called **minterms**.
- **Minterm:** For a Boolean function of n variables, a product of n literals in which each variable appears exactly once (in either true or complemented form, but not both) is called a **minterm**.

  For example, a function with 3 variables (a, b, c) has 8 minterms:

  $\bar{a}\,\bar{b}\,\bar{c},\ \bar{a}\,\bar{b}\,c,\ \bar{a}\,b\,\bar{c},\ \bar{a}\,b\,c,\ a\,\bar{b}\,\bar{c},\ a\,\bar{b}\,c,\ a\,b\,\bar{c},\ a\,b\,c$
- Each minterm that appears in the 1st canonical form covers only one row of the truth table with the output "1".

  For example, the minterm $\bar{a}\,\bar{b}\,\bar{c}$ can generate "1" only for the input value $a\,b\,c$ = 000.

  For all other input combinations, the minterm $\bar{a}\,\bar{b}\,\bar{c}$ generates "0".

Summary:

- The 1st canonical form of a function is the sum of minterms.
- In the 1st canonical form, each product (minterm) in the sum corresponds to a row in the truth table with the output "1".

1st Canonical Form (cont'd) :

Finding minterms:

Truth Table → Expression in SOP form

- To find minterms, we locate all rows in the truth table where the output is "1".
- To obtain the individual minterms, we substitute variables (for example, A, B, or C) for ones (of the inputs) and complements of variables ($\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, or $\overline{\mathbf{C}}$) for zeros in the truth table.

Example:

"True" (1) combinations: 001 011 101 110 111

Sum of Minterms: $F(A,B,C) = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$

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

The 1st canonical form of the complement of a function:

We can similarly obtain the 1st canonical form of the complement of a function by considering the "false" (0) rows.

Example:

Obtain the 1st canonical form of the complement of a function F from the previous example.

1st canonical form of $\overline{F}$:

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

A	B	C	F	$\overline{F}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Simplification of expressions in the 1st canonical form:

Canonical forms are usually not the simplest (optimal) algebraic expression of the function.
They can usually be simplified.

Minimization:

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

- A Boolean function may have many possible logic expressions. They produce the same result given the same inputs.
- Since the minterms in the 1st canonical form are in one-to-one correspondence with the 1's of the truth table, the 1st canonical (standard) form expression for a function is <u>unique</u>.
- A function can have only one expression in the 1st canonical form.

Indexing minterms:

We assign each minterm an index (number) based on the binary encoding of the variables.
This decimal number represents the row number (Row numbers start at 0).
For example, we assign the index 5 to the minterm $\mathbf{A\,\bar{B}\,C}$ (101) and designate it m5.

Minterms for 3 variables (A,B,C):

Inputs:

A	B	C	Minterms
0	0	0	$\mathbf{\bar{A}\,\bar{B}\,\bar{C}}$ = m0
0	0	1	$\mathbf{\bar{A}\,\bar{B}\,C}$ = m1
0	1	0	$\mathbf{\bar{A}\,B\,\bar{C}}$ = m2
0	1	1	$\mathbf{\bar{A}\,B\,C}$ = m3
1	0	0	$\mathbf{A\,\bar{B}\,\bar{C}}$ = m4
1	0	1	$\mathbf{A\,\bar{B}\,C}$ = m5
1	1	0	$\mathbf{A\,B\,\bar{C}}$ = m6
1	1	1	$\mathbf{A\,B\,C}$ = m7

Example:

Expression of function F in (2.33) in 1st canonical form:

$$\begin{aligned} F(A, B, C) &= \Sigma m(1,3,5,6,7) \\ &= m1 + m3 + m5 + m6 + m7 \\ &= \mathbf{\bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C} \end{aligned}$$

$$F = \Sigma_{A,\,B,\,C}(1,3,5,6,7) \quad \text{(Sum of minterms)}$$

2nd Canonical Form: Product of Sums

- The 2nd canonical form is the product of special sum terms called **maxterms**.
- **Maxterm:** For a Boolean function of n variables, a <u>sum</u> of n literals in which <u>each</u> variable appears <u>exactly once</u> (in either true or complemented form, but not both) is called a **maxterm**.
- For example, a function with 3 variables (a, b, c) has 8 maxterms:

$$\mathbf{a+b+c,\ a+b+\bar{c},\ a+\bar{b}+c,\ a+\bar{b}+\bar{c},\ \bar{a}+b+c,\ \bar{a}+b+\bar{c},\ \bar{a}+\bar{b}+c,\ \bar{a}+\bar{b}+\bar{c}}$$

- Each maxterm has a value of "0" for exactly **one** combination of values for the input variables and "1" for all other combinations.

  For example, the maxterm $\mathbf{a+b+c}$ can generate "0" only for the input value $\mathbf{a\,b\,c}$ =000.

  For all other input combinations, the maxterm $\mathbf{a+b+c}$ generates "1".

Summary:

- The 2nd canonical form of a function is the product of maxterms.
- In the 2nd canonical form, each sum term (maxterm) in the product corresponds to a row in the truth table with the output "0".

2nd Canonical Form (cont'd):

Finding maxterms:

Truth Table → Expression in POS form

- To find the maxterms, we locate all rows in the truth table where the output is "0".
- To obtain the individual maxterms, we substitute variables (for example, A, B, or C) for zeros (of the inputs) and complements of variables ($\mathbf{\bar{A}}$, $\mathbf{\bar{B}}$, or $\mathbf{\bar{C}}$) for ones in the truth table.

Example:

"False" (0-generating) combinations: *000* *010* *100*

Product of maxterms: F(A,B,C) = $\mathbf{(A+B+C)\ (A+\bar{B}+C)\ (\bar{A}+B+C)}$

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

Note that this function F has the same truth table as the function on slide 2.33.

The expressions in the 1st and 2nd canonical forms both correspond to the same truth table.

The 2nd canonical form of the complement of a function:

We can similarly obtain the 2nd canonical form of the complement of a function by considering the "true" (1) rows:

Example:

Obtain the 2nd canonical form of the complement of a function F from the previous example.

2nd canonical form of $\overline{\text{F}}$:

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

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

Simplification of expressions in the 2nd canonical form:

Canonical forms are usually not the simplest (optimal) algebraic expression of the function.
They can usually be simplified.

Minimization:

$$\begin{aligned}\mathbf{F}(\mathbf{A},\mathbf{B},\mathbf{C}) &= (\mathbf{A}+\mathbf{B}+\mathbf{C})(\mathbf{A}+\overline{\mathbf{B}}+\mathbf{C})(\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C})\\ &= \big((\mathbf{A}+\mathbf{C})(\mathbf{B}+\overline{\mathbf{B}})\big)(\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C})\\ &= (\mathbf{A}+\mathbf{C})(\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C})\\ &= (\mathbf{A}+\mathbf{C})(\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C})(\mathbf{B}+\mathbf{C})\ \text{(consensus)}\\ &= (\mathbf{A}+\mathbf{C})(\mathbf{B}+\mathbf{C})\end{aligned}$$

- A Boolean function may have many possible logic expressions. They produce the same result given the same inputs.
- Since the maxterms in the 2nd canonical form are in one-to-one correspondence with the 0's of the truth table, the 2nd canonical (standard) form expression for a function is unique.
- A function can have only one expression in the 2nd canonical form.

Indexing maxterms:

We assign each maxterm an index (number) based on the binary encoding of the variables. This is a decimal number that represents the row number (Row numbers start at 0).
For example, we assign the index 5 to the maxterm $\bar{a} + b + \bar{c}$ (101) and designate it M5.

Maxterms for 3 variables (A,B,C):

Inputs:

A	B	C	Maxterms	
0	0	0	$A + B + C$	M0
0	0	1	$A + B + \bar{C}$	M1
0	1	0	$A + \bar{B} + C$	M2
0	1	1	$A + \bar{B} + \bar{C}$	M3
1	0	0	$\bar{A} + B + C$	M4
1	0	1	$\bar{A} + B + \bar{C}$	M5
1	1	0	$\bar{A} + \bar{B} + C$	M6
1	1	1	$\bar{A} + \bar{B} + \bar{C}$	M7

Example:
Expression of function F in (2.38) in 2nd canonical form:

$$\begin{aligned} F(A,B,C) &= \Pi M(0,2,4) \\ &= M0 \bullet M2 \bullet M4 \\ &= (A + B + C)\,(A + \bar{B} + C)\,(\bar{A} + B + C) \end{aligned}$$

$$F = \Pi_{A,B,C}(0,2,4)$$ product of maxterms.

Conversions Between Canonical Forms

- Converting from 1st (sum of minterms) form to 2nd (product of maxterms) form:
  - Replace the minterms with maxterms, and assign them numbers of minterms that do not appear in the 1st canonical form.
  - Example: $F(A,B,C) = \Sigma m(1,3,5,6,7) = \Pi M(0,2,4)$
    $F(A,B,C) = m1 + m3 + m5 + m6 + m7 = M0 \cdot M2 \cdot M4$
    $F(A,B,C) = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C = (A + B + C)(A + \bar{B} + C)(\bar{A} + B + C)$
- Converting from 2nd (product to maxterms) form to 1st (sum of minterms) form:
  - Replace the maxterms with minterms, and assign them numbers of maxterms that do not appear in the 2nd canonical form.
  - Example: $F(A,B,C) = \Pi M(0,2,4) = \Sigma m(1,3,5,6,7)$
- Finding the complement of the function in sum of minterms form:
  - Select the minterms that do not appear in the 1st canonical form.
  - Example: $F(A,B,C) = \Sigma m(1,3,5,6,7) \Rightarrow \overline{F(A,B,C)} = \Sigma m(0,2,4)$
- Finding the complement of the function in the product of maxterms form:
  - Select the maxterms that do not appear in the 2nd canonical form.
  - Example: $F(A,B,C) = \Pi M(0,2,4) \Rightarrow \overline{F(A,B,C)} = \Pi M(1,3,5,6,7)$

Canonical Forms and the De Morgan's Theorem

- Applying De Morgan's law to the complement of the function in the 1st canonical form generates the expression in the 2nd canonical form.

Example:

Complement of the function in SOP form (Slide 2.34):

$$\overline{F} = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C}$$

Applying De Morgan

$$\overline{\overline{F}} = \overline{\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C}}$$

$$F = (A + B + C)(A + \overline{B} + C)(\overline{A} + B + C)$$ 2nd canonical form (2.38).

- Applying the De Morgan's theorem to the complement of the function in the 2nd canonical form generates the expression in the 1st canonical form.

Example:

Complement of the function in POS form (2.39):

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

Applying De Morgan

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

$$F = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$$ 1st canonical form (2.33).

Graphical Representation

The input variables (combinations) of a logic function are elements (vectors) of the set B^n. Example: B^3={000, 001, …, 111}

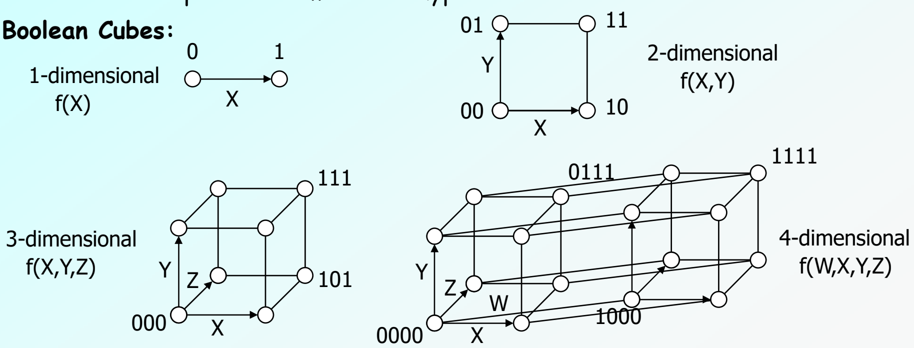
We can represent these variables as vertices of an n-dimensional hypercube.

Vertices that correspond to 1-generating combinations are colored (marked).

The number of inputs of the function determines how many dimensions the hypercube has.

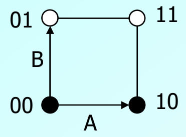
n-bit input → n-dimensional hypercube

Boolean Cubes:



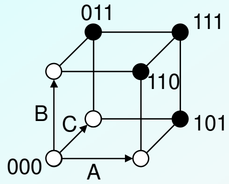
Example:
F(A,B)

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



Example:
F(A,B,C)

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



As the number of variables (inputs) increases, drawing a Boolean cube becomes more difficult.

Therefore, Boolean cubes are not practical for representing Boolean functions with many inputs.

However, they make it easier to visualize some properties (especially adjacent combinations) of the functions and to explain further topics.

Karnaugh Maps

Maurice Karnaugh (1924-2022), American physicist and mathematician

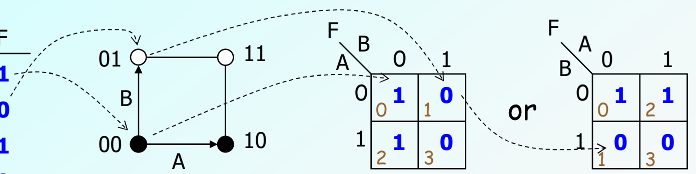
The Karnaugh map is a tool for representing and simplifying Boolean functions.
The Boolean variables and related output values are transferred (generally from a truth table) into a table that is in a matrix form.

Example: F(A,B)

Truth table:

Row Num.	A	B	F
0	0	0	**1**
1	0	1	**0**
2	1	0	**1**
3	1	1	**0**

Boolean Cube: Karnaugh maps:



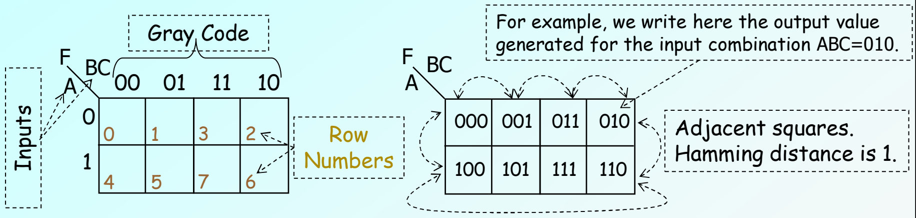
A: row, B: column A: column, B: row

We can place different variables in columns or rows.

Karnaugh map (cont'd)

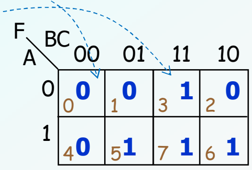
The rows and columns are labeled according to the **Gray** code property so that variables in adjacent squares (horizontal and vertical) of the map differ in only one variable.

Format of the Karnaugh map for a function with 3 inputs: F(A,B,C)

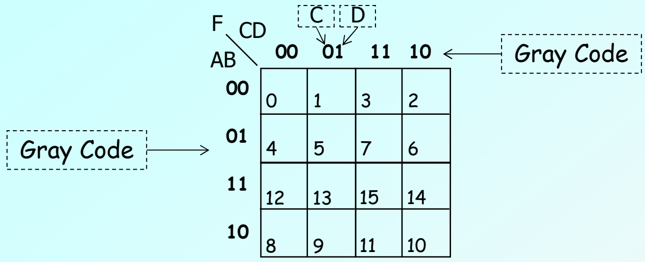


Example:

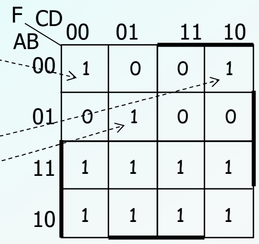
Num	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Format of the Karnaugh map of a function with 4 inputs: F(A,B,C,D)



Example:
Draw the Karnaugh map for the following function.
F(A,B,C,D) = $\cup_1$ (0,2,5,8,9,10,11,12,13,14,15)



We will use Karnaugh maps to simplify Boolean functions in the coming lectures.