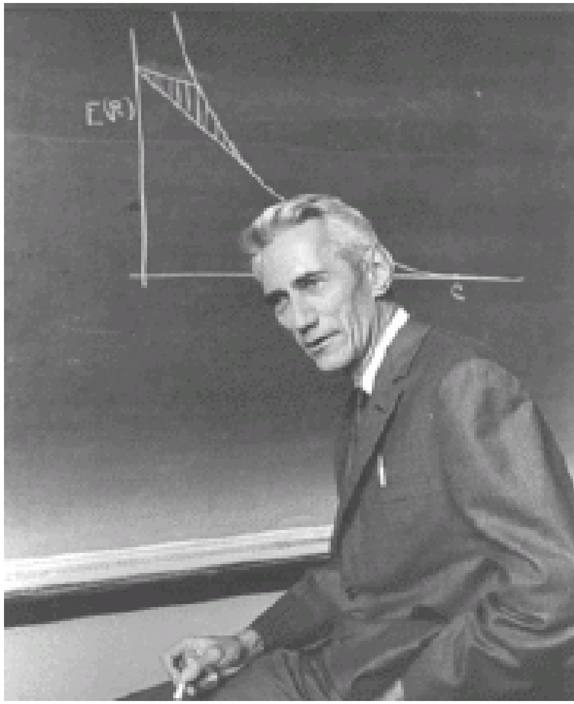
Binary Logic and Gates

- ✓ Binary variables take on one of two values.
- ✓ Logical operators operate on binary values and binary variables.
- ✓ Basic logical operators are the logic functions AND, OR and NOT.
- ✓ Logic gates implement logic functions.
- ✓ Boolean Algebra: a useful mathematical system for specifying and transforming logic functions.
- ✓ We study Boolean algebra as a foundation for designing and analyzing digital systems!

George Boole (1815-1864)

An Investigation of the Laws of Thought, on Which are founded the Mathematical Theories of Logic and Probabilities (1854)

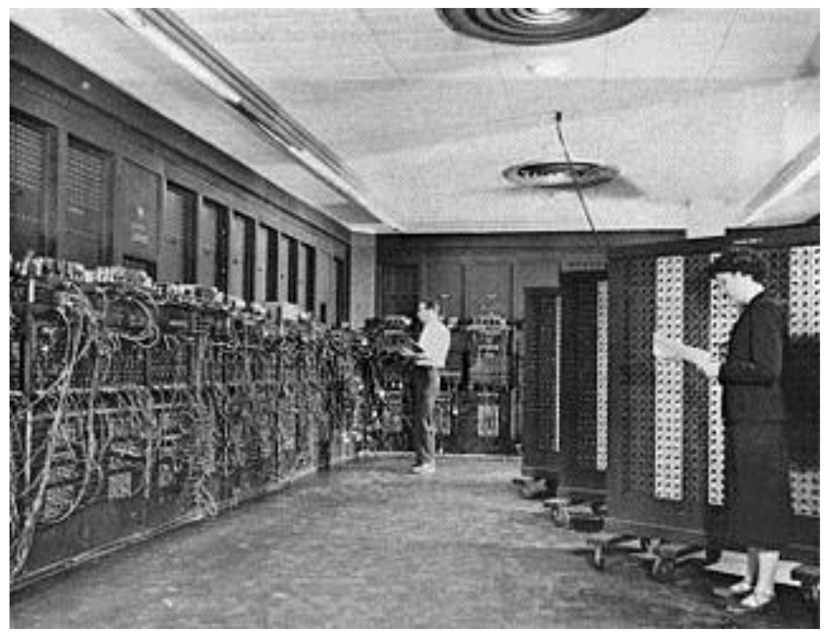
Claude Shannon (1916-2001)



A Symbolic Analysis of Relay and Switching Circuits (1938)

ENIAC (1946)

Electronic
Numerical
Integrator
And
Calculator



Binary Variables

- ✓ Recall that the two binary values have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- ✓ We use 1 and 0 to denote the two values.
- ✓ Variable identifier examples:
  - A, B, y, z, or X_1 for now
  - RESET, START_IT, or ADD1 later

Boolean functions

✓ Basic logical operators are the **boolean functions**

$$\underbrace{f(x_1,\ldots,x_n)}_{\text{arguments}}: \quad \underbrace{\{0,1\}^n}_{\text{domain}} \longrightarrow \underbrace{\{0,1\}}_{\text{codomain}}$$

The Boolean n-cube B^n

- $B = \{0,1\}$
- $B^2 = \{0,1\} \times \{0,1\} = \{00, 01, 10, 11\}$
- Arrangement of function table on a hypercube
  - The function value f_j is adjacent in each dimension of the hypercube to f_k where K is obtained from j by complementing one and only one input variable: $X_0 X_1 \ldots X_n$ is adjacent to

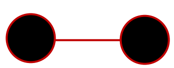
$\bar{X}_0 X_1 \ldots X_n$

$X_0 \bar{X}_1 \ldots X_n$

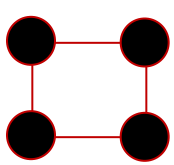
$\ldots$

$X_0 X_1 \ldots \bar{X}_n$

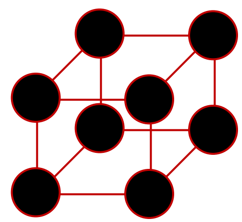
B^1



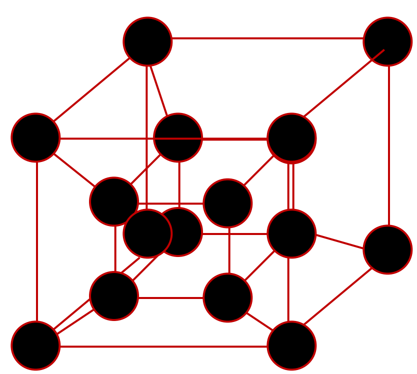
B^2



B^3



B^4



Boolean Functions

Boolean Function: $f(x): B^n \to B$

$B = \{0,1\}$

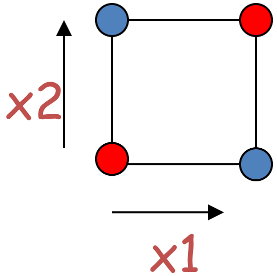
$x = (x_1, x_2, \ldots, x_n) \in B^n; x_i \in B$

- $x_1, x_2, \ldots$ are variables
- $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots$ are literals
- essentially: f maps each vertex of B^n to 0 or 1

Example:

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

x_1 \ x_2		
	0	1
	1	0

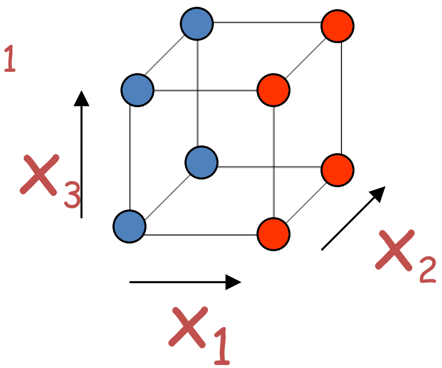


Boolean Functions

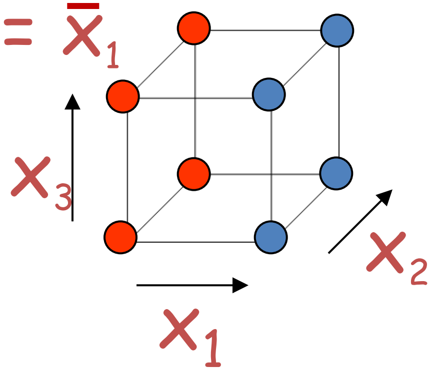
- The Onset of f is $\{x \mid f(x) = 1\} = f^{-1}(1) = f^1$
- The Offset of f is $\{x \mid f(x) = 0\} = f^{-1}(0) = f^0$

- if $f^1 = B^n$, f is the tautology. i.e. $f \equiv 1$
- if $f^0 = B^n$ $(f^1 = \emptyset)$, f is not satisfyable, i.e. $f \equiv 0$
- if $f(x) = g(x)$ *for all* $x \in B^n$, then f and g are equivalent
- we say f instead of f^1
- literals: A literal is a variable or its negation x, $\bar{x}$ and represents a logic function

$f = x_1$



$f = \bar{x}_1$



Logical Operations

- ✓ The three basic logical operations are:
  - AND
  - OR
  - NOT
- ✓ AND is denoted by a dot (•).
- ✓ OR is denoted by a plus (+).
- ✓ NOT is denoted by an overbar (¯), a single quote mark (') after, or (~) before the variable.
- ✓ The order of evaluation in a Boolean expression is:
  1. Parentheses
  2. NOT
  3. AND
  4. OR
- ✓ Consequence: Parentheses appear around OR expressions
- ✓ Example: F = A(B + C)(C + D)

Fundamentals of Boolean Algebra

✓ *Basic Postulates*

- ***Postulate 1 (Definition)***: A Boolean algebra is a closed algebraic system containing a set K of two or more elements and the operators $\bullet$, + and $\overline{}$.
- ***Postulate 2 (Existence of 1 and 0 element)***:

  a) $a + 0 = a$ (identity for +), b) $a \bullet 1 = a$ (identity for $\bullet$)
- ***Postulate 3 (Commutativity)***:

  a) $a + b = b + a$ b) $a \bullet b = b \bullet a$
- ***Postulate 4 (Associativity)***:

  a) $a + (b + c) = (a + b) + c$ b) $a \bullet (b \bullet c) = (a \bullet b) \bullet c$
- ***Postulate 5 (Distributivity)***:

  a) $a + (b \bullet c) = (a + b) \bullet (a + c)$ b) $a \bullet (b + c) = a \bullet b + a \bullet c$
- ***Postulate 6 (Existence of complement)***:
- a) $a + \overline{a} = 1$ b) $a \bullet \overline{a} = 0$

Normally $\bullet$ is omitted. **A switching algebra is a BA with K={0,1}**

Notation Examples

✓ Examples:

- $Y = A \cdot B$ is read "Y is equal to A AND B"
- $z = x + y$ is read "z is equal to x OR y"
- $X = \overline{A}$ is read "X is equal to NOT A"

- Note: The statement:

  1 + 1 = 2 (+ is an algebraic operator, read "one plus one equals two")

  is not the same as

  1 + 1 = 1 (+ is a logic operator, read "1 or 1 equals 1").

Operator Definitions

- Operations are defined on the values "0" and "1" for each operator:

AND	OR	NOT
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\overline{1} = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

Boolean Algebra

- An algebraic structure defined on a set of at least two elements, B, together with three binary operators (denoted +, • and ¯) that satisfies the following basic identities:

1. $X + 0 = X$	2. $X \cdot 1 = X$	Existence of 0 and 1
3. $X + 1 = 1$	4. $X \cdot 0 = 0$	
5. $X + X = X$	6. $X \cdot X = X$	Idempotence
7. $X + \overline{X} = 1$	8. $X \cdot \overline{X} = 0$	Existence of complement
9. $\overline{\overline{X}} = X$		Involution
10. $X + Y = Y + X$	11. $XY = YX$	Commutative
12. $(X + Y) + Z = X + (Y + Z)$	13. $(XY)Z = X(YZ)$	Associative
14. $X(Y + Z) = XY + XZ$	15. $X + YZ = (X + Y)(X + Z)$	Distributive
16. $\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17. $\overline{X \cdot Y} = \overline{X} + \overline{Y}$	De Morgan's

Generalized De Morgan's theorems

$$\overline{X_1 X_2 \ldots X_n} = \overline{X_1}+\overline{X_2}+\ldots+\overline{X_n}$$

$$\overline{X_1+X_2+\ldots+X_n} = \overline{X_1}\,\overline{X_2}\ldots\overline{X_n}$$

- ✓ Proof Generalized De Morgan's theorems by general induction:

  Two steps:

  - Show that the statement is true for two variables
  - Show that if is true for n variable , than is also true for n+1 variables:

    Let $Z = X_1 + X_2 + \ldots + X_n$

    $\overline{(X_1 + X_2 + \ldots + X_n + X_{n+1})} = \overline{(Z + X_{n+1})} = (\overline{Z} \bullet \overline{X_{n+1}}) =$

    $(\overline{X_1} \bullet \overline{X_2} \bullet \ldots \bullet \overline{X_n}) \bullet \overline{X_{n+1}}$ by induction hypothesis

Example 1: Boolean Algebraic Proof

✓ $A + A \cdot B = A$ Absorption Theorem

Proof Steps	Justification
$A + A \cdot B$	
$= A \cdot 1 + A \cdot B$	$X = X \cdot 1$
$= A \cdot (1 + B)$	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ Distributive Law
$= A \cdot 1$	$1 + X = 1$
$= A$	$X \cdot 1 = X$

Some Properties of Identities & the Algebra

- The dual of an algebraic expression is obtained by interchanging + and • and interchanging 0's and 1's.
- Unless it happens to be self-dual, the dual of an expression does not equal the expression itself.
- Example: $F = (A + \overline{C}) \cdot B + 0$

  dual $F = ((A \cdot \overline{C}) + B) \cdot 1 = A \cdot \overline{C} + B$
- Example: $G = X \cdot Y + \overline{(W + Z)}$

  dual $G = (X+Y) \cdot \overline{(W \cdot Z)} = (X+Y) \cdot (\overline{W} + \overline{Z})$
- Example: $H = A \cdot B + A \cdot C + B \cdot C$

  dual $H = (A + B)(A + C)(B + C) = (A + AC + BA + BC)\ (B + C)$

  $= (A + BC)\ (B+C) = AB + AC + BC$. So H is self-dual.
- Are any of these functions self-dual?

Some Properties of Identities & the Algebra

- ✓ There can be more that 2 elements in B, i. e., elements other than 1 and 0. What are some common useful Boolean algebras with more than two elements?
  1. Algebra of Sets
  2. Algebra of n-bit binary vectors
  3. Quantified Boolean Algebra (QBA)
- ✓ If B contains only 1 and 0, then B is called the <u>switching algebra</u> which is the algebra we use most often.

Quantified Boolean formulas (QBFs)

- *Generalize (quantifier-free) Boolean formulas* with the additional universal and existential quantifiers: $\forall$ and $\exists$, respectively.
- In writing a QBF, we assume that the precedences of the quantifiers are lower than those of the Boolean connectives.
- In a QBF, variables being quantified are called *bound variables*, whereas those not quantified are called *free variables*.
- Any QBF can be rewritten as a quantifier-free Boolean formula through quantifier elimination by formula expansion (among other methods), e.g.,

$$\forall x{:}f(x; y) = f(0; y) \cdot f(1; y)$$

and

$$\exists x{:}f(x; y) = f(0; y) + f(1; y)$$

- Consequently, for any QBF φ, there exists an equivalent quantifier-free Boolean formula that refers only to the free variables of φ.
- QBFs are thus of the same expressive power as quantifier-free Boolean formulas, but can be more succinct.

Example 2: Boolean Algebraic Proofs

✓ $AB + \overline{A}C + BC = AB + \overline{A}C$ Consensus Theorem

Proof Steps

$$AB + \overline{A}C + BC$$
$$= AB + \overline{A}C + 1 \cdot BC$$
$$= AB + \overline{A}C + (A + \overline{A}) \cdot BC$$
$$= AB + \overline{A}C + ABC + \overline{A}BC$$
$$= AB \cdot (1 + C) + \overline{A}C \cdot (1 + B)$$
$$= AB + \overline{A}C$$

✓ $(A+B) \cdot (\overline{A}+C) \cdot (B+C) = (A+B) \cdot (\overline{A}+C)$ dual identity

Example 3: Boolean Algebraic Proofs

✓ $\overline{(X + Y)}Z + X\overline{Y} = \overline{Y}(X + Z)$

Proof Steps	Justification	
$\overline{(X + Y)}Z + X\overline{Y}$		
$= X' Y' Z + X Y'$	$(A + B)' = A' \cdot B'$	De Morgan's Law
$= Y' X' Z + Y' X$	$A \cdot B = B \cdot A$	Commutative Law
$= Y' (X' Z + X)$	$A(B + C) = AB + AC$	Distributive Law
$= Y' (X' + X)(Z + X)$	$A + BC = (A + B)(A + C)$	Distributive Law
$= Y' \cdot 1 \cdot (Z + X)$	$A + A' = 1$	
$= Y' (X + Z)$	$1 \cdot A = A$, $A + B = B + A$	Commutative Law

Useful Theorems

- ✓ $X \cdot Y + \overline{X} \cdot Y = Y$ $(X + Y) \cdot (\overline{X} + Y) = Y$ Minimization
- ✓ $X + X \cdot Y = X$ $X \cdot (X + Y) = X$ Absorption
- ✓ $X + \overline{X} \cdot Y = X + Y$ $X \cdot (\overline{X} + Y) = X \cdot Y$ Simplification
- ✓ $X \cdot Y + \overline{X} \cdot Z + Y \cdot Z = X \cdot Y + \overline{X} \cdot Z$ Consensus

  $(X + Y) \cdot (\overline{X} + Z) \cdot (Y + Z) = (X + Y) \cdot (\overline{X} + Z)$
- ✓ $\overline{X + Y} = \overline{X} \cdot \overline{Y}$ $\overline{X \cdot Y} = \overline{X} + \overline{Y}$ De Morgan's Law

Boolean Function Evaluation

$F1 = xy\bar{z}$
$F2 = x + \bar{y}z$
$F3 = \bar{x}\bar{y}\bar{z} + \bar{x}yz + x\bar{y}$
$F4 = x\bar{y} + \bar{x}z$

x	y	z	F1	F2	F3	F4
0	0	0	0	0		
0	0	1	0	1		
0	1	0	0	0		
0	1	1	0	0		
1	0	0	0	1		
1	0	1	0	1		
1	1	0	1	1		
1	1	1	0	1		

Expression Simplification

- ✓ An application of Boolean algebra
- ✓ Simplify to contain the smallest number of literals (complemented and un-complemented variables):

$$AB+\overline{A}CD+\overline{A}BD+\overline{A}C\overline{D}+ABCD$$

$= AB+ABCD+\overline{A}CD+\overline{A}C\overline{D}+\overline{A}BD$

$= AB+AB(CD)+\overline{A}C(D+\overline{D})+\overline{A}BD$

$= AB+\overline{A}C+\overline{A}BD = B(A+\overline{A}D)+\overline{A}C$

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

Complementing Functions

- Use DeMorgan's Theorem to complement a function:
  1. Interchange AND and OR operators
  2. Complement each constant value and literal
- Example: Complement $F = \overline{x}y\overline{z} + x\overline{y}\overline{z}$

  $\overline{F} = (x + \overline{y} + z)(\overline{x} + y + z)$
- Example: Complement $G = (\overline{a} + bc)\overline{d} + e$

  $\overline{G} =$

  $((a(\overline{b} + \overline{c})) + d)\overline{e} = (a(\overline{b} + \overline{c}) + d)\overline{e}$

Shannon Expansion

- Let $f : B^n \rightarrow B$ be a Boolean function, and $x = (x_1, x_2, \ldots, x_n)$ the variables in the support of f. The cofactor (residual) f_a of f by a literal $a=x_i$ or $a=\overline{x_i}$ is:

$$f_{x_i}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

$$f_{\overline{x_i}}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$

- Shannon theorem:

$$f = x_i f_{x_i} + \bar{x}_i f_{\overline{x_i}}$$

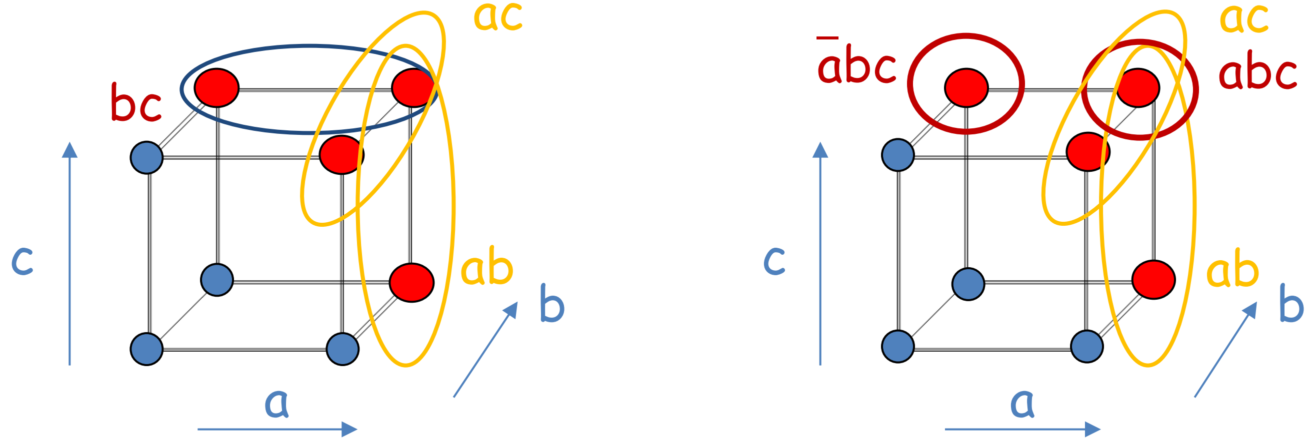
- We say that f is expanded about x_i. x_i is called the splitting variable.

Example

$F = ab + ac + bc$

$F = a\,F_a + \bar{a}\,F_{\bar{a}}$

$F = ab + ac + abc + \bar{a}bc$



Cube bc got split into two cubes

Representation of Boolean Functions

- We need representations for Boolean Functions for two reasons:
  - to represent and manipulate the actual circuit we are "synthesizing"
  - as mechanism to do efficient Boolean reasoning
- Forms to represent Boolean Functions
  - Truth table
  - List of cubes (Sum of Products, Disjunctive Normal Form (DNF))
  - List of conjuncts (Product of Sums, Conjunctive Normal Form (CNF))
  - Boolean formula
  - Binary Decision Tree, Binary Decision Diagram
  - Circuit (network of Boolean primitives)

Truth Table

- ✓ Truth table (Function Table):

  The truth table of a function $f : B^n \rightarrow B$ is a tabulation of its value at each of the 2^n vertices of B^n.

- ✓ In other words the truth table lists all minterms

  Example: $f = \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d + a\bar{b}\bar{c}d + a\bar{b}cd + ab\bar{c}d + abc\bar{d} + abcd$

  The truth table representation is
  - intractable for large n
  - canonical

	abcd	f		abcd	f
0	0000	0	8	1000	0
1	0001	1	9	1001	1
2	0010	0	10	1010	0
3	0011	1	11	1011	1
4	0100	0	12	1100	0
5	0101	1	13	1101	1
6	0110	0	14	1110	1
7	0111	0	15	1111	1

Truth Tables

- ✓ *Truth table* – a tabular listing of the values of a function for all possible combinations of values on its arguments
- ✓ Example: Truth tables for the basic logic operations:

AND

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

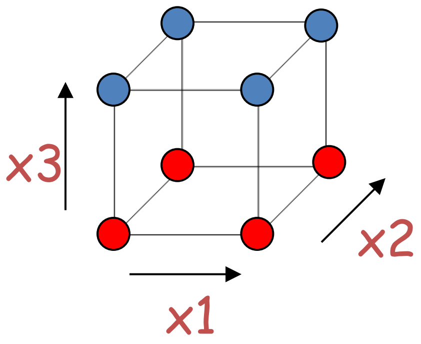
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	$Z = \overline{X}$
0	1
1	0

Set of Boolean Functions

✓ Truth Table or Function table:



$x_1x_2x_3$	
0 0 0	1
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0 ⇒	1
1 0 1	0
1 1 0	1
1 1 1	0

✓ There are 2^n vertices in input space B^n

✓ There are 2^{2^n} distinct logic functions.

- Each subset of vertices is a distinct logic function: $f \subseteq B^n$

Cubes

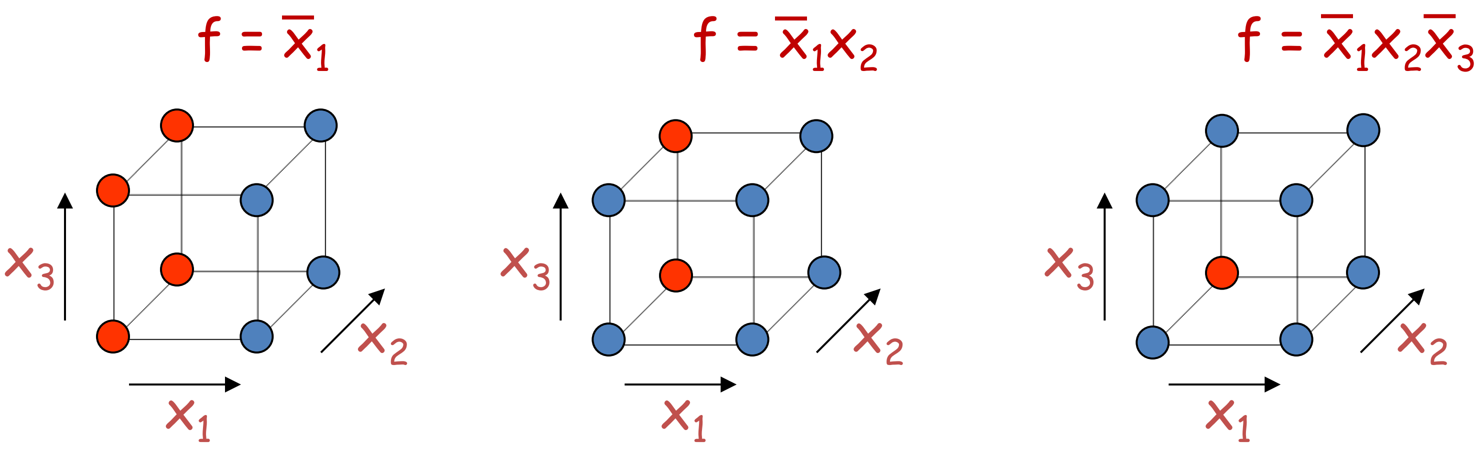
- ✓ A <u>cube</u> is defined as the AND of a set of literals ("conjunction" of literals).

Example:

$C = \overline{x}_1 x_2 \overline{x}_3$

represents the following function

$f = (x_1{=}0)(x_2{=}1)(x_3{=}0)$



Cubes

- ✓ If $C \subseteq f$, C a cube, then C is an implicant of f.
- ✓ If $C \subseteq B^n$, and C has k literals, then $|C|$ covers 2^{n-k} vertices.

  Example:

  $C = x\bar{y} \subseteq B^3$

  $k = 2$, $n = 3$ => $|C| = 2 = 2^{3-2}$.

  $C = \{100, 101\}$

- ✓ An implicant with n literals is a minterm.

List of Cubes

✓ Sum of Products (SOP):

- A function can be represented by a sum of cubes (products):

  $f = ab + ac + bc$

  Since each cube is a product of literals, this is a "sum of products" (SOP) representation

- A SOP can be thought of as a set of cubes F

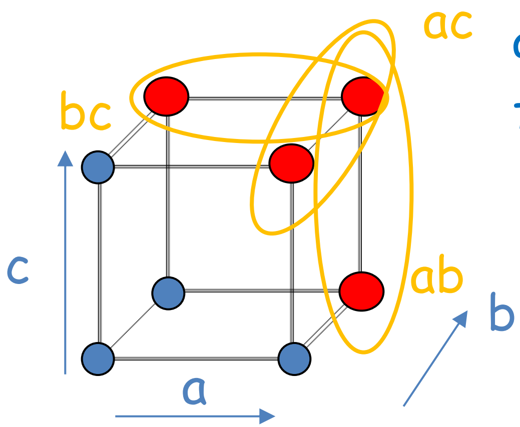
  $$F = \{ab, ac, bc\}$$

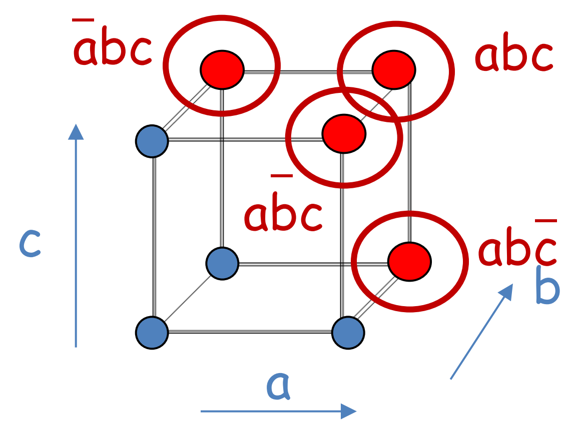
- A set of cubes that represents f is called a cover of f.

  $F_1=\{ab, ac, bc\}$ and $F_2=\{abc, ab\bar{c}, a\bar{b}c, \bar{a}bc\}$

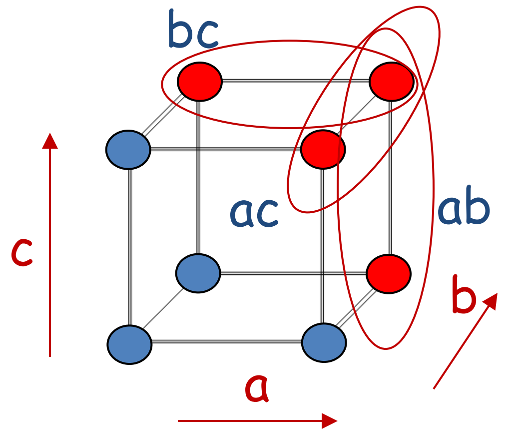
  are covers of

  $f = ab + ac + bc.$





Sum Of Products - SOP



= onset minterm

Note that each onset minterm is "covered" by at least one of the cubes, and these do not covers offset minterm.

- ✓ Two-level minimization seeks the minimum size cover (least number of cubes)

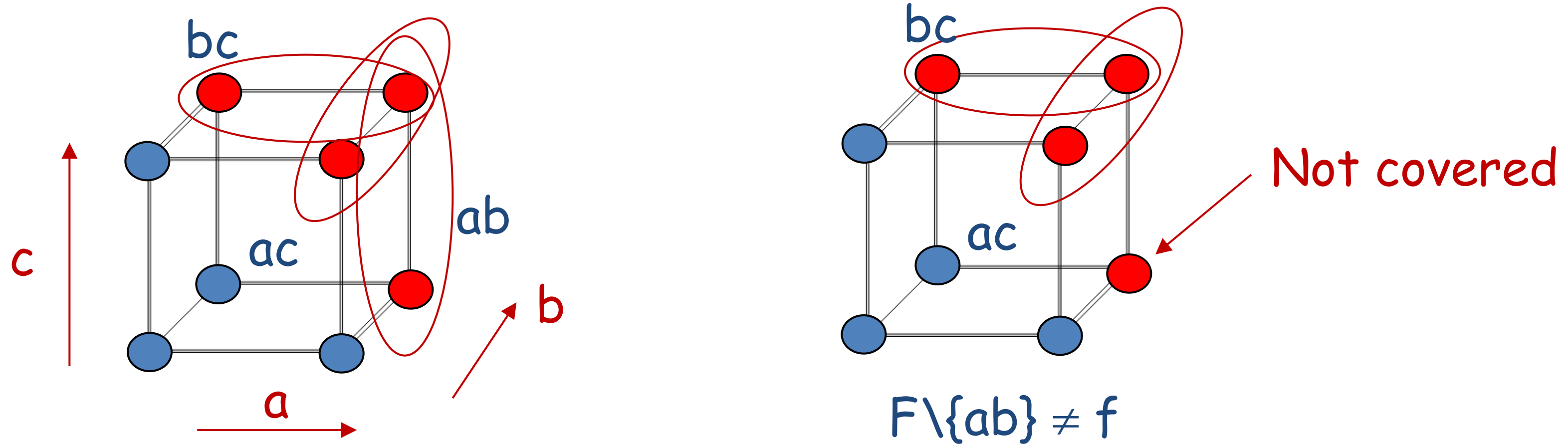
Irredundant

✓ Let $F = \{c_1, c_2, \ldots, c_k\}$ be a cover for f:

$$f = \sum_{i=1}^{k} c_i$$

A cube $c_i \in F$ is irredundant if $F \setminus \{c_i\} \neq f$

Example 2: $f = ab + ac + bc$



Prime

- ✓ A literal j of cube $c_i \in F$ ($= f$) is prime if

$$(F \setminus \{c_i\}) \cup \{c'_i\} \neq f$$

where c'_i is c_i with literal j of c_i deleted.

- ✓ A cube of F is prime if all its literals are prime.

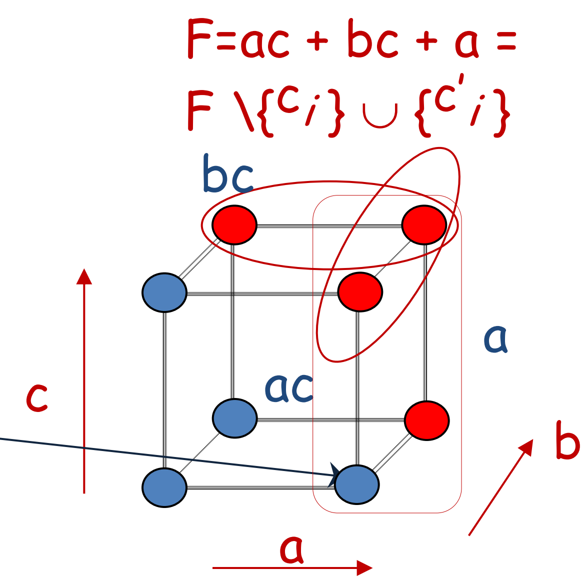
Example 3

$f = ab + ac + bc$

$c_i = ab$; $c'_i = a$ (literal b deleted)

$F \setminus \{c_i\} \cup \{c'_i\} = a + ac + bc$

Not equal to f since offset vertex is covered



Prime and Irredundant Covers

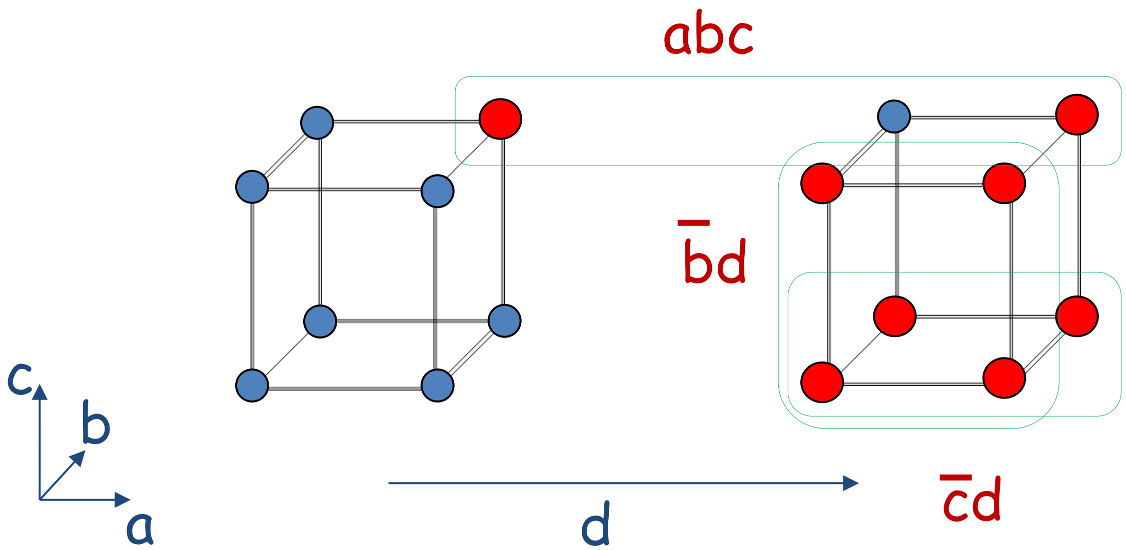
- ✓ Definition 1 A cover is prime (irredundant) if all its cubes are prime (irredundant).
- ✓ Definition 2 A prime of f is essential (essential prime) if there is a minterm (essential vertex) in that prime but in no other prime.

Prime and Irredundant Covers

Example 4

$f = abc + \bar{b}d + \bar{c}d$ is prime and irredundant.

abc is **essential** since $abcd \in abc$, but not in $\bar{b}d$ or $\bar{c}d$

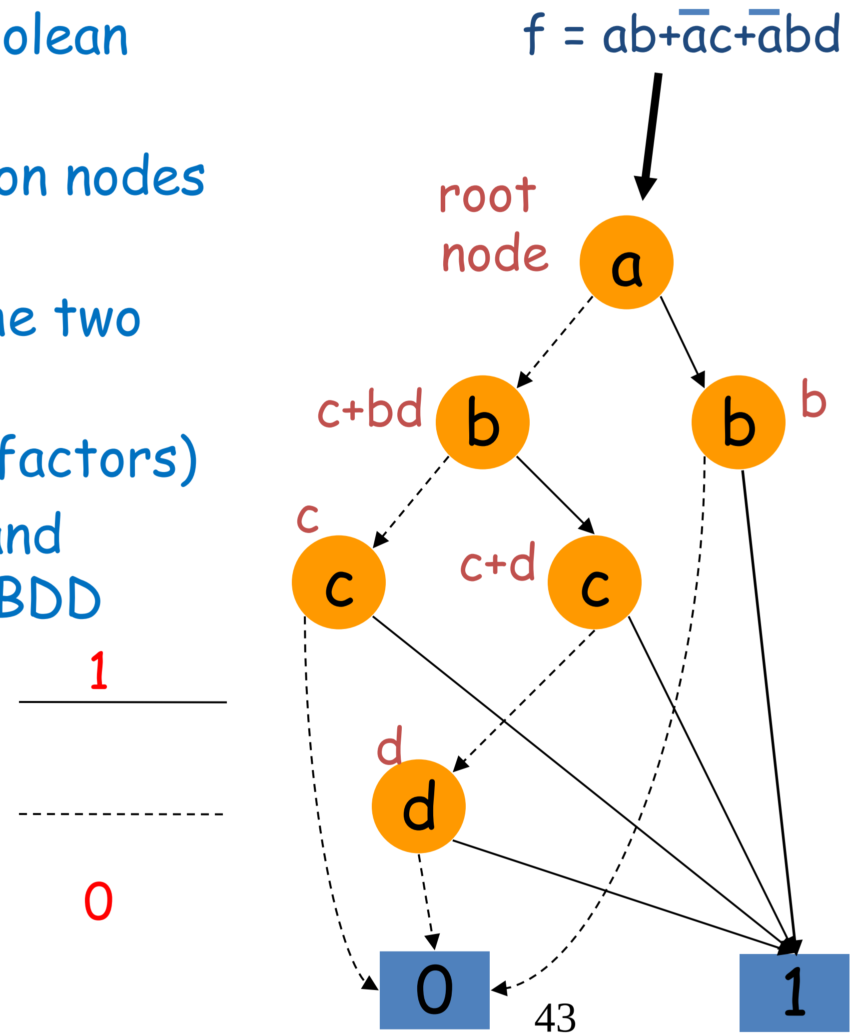


Binary Decision Diagram (BDD)

✓ Graph representation of a Boolean function f

- vertices represent decision nodes for variables
- two children represent the two subfunctions

  f(x = 0) and f(x = 1) (cofactors)
- restrictions on ordering and reduction rules can make a BDD representation canonical



Logic Functions

- ✓ However, there are infinite number of logic formulas and each one can have various forms:

$$f = x + y$$
$$= \overline{x}y + xy + x\overline{y}$$
$$= \overline{x}x + x\overline{y} + y$$
$$= (x + y)(\overline{x} + \overline{y}) + xy$$

x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

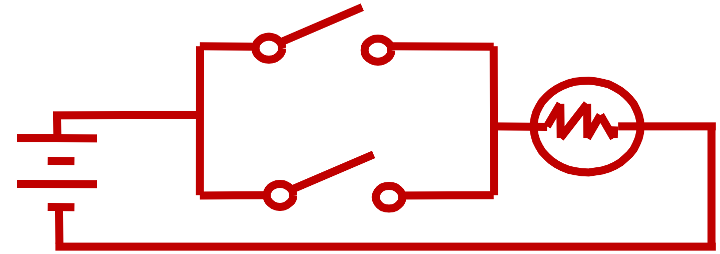
- ✓ Synthesis = Find the best formula (or "representation")

Logic Function Implementation

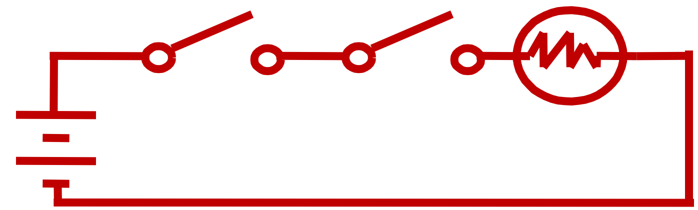
✓ Using Switches

- For inputs:
  - logic 1 is switch closed
  - logic 0 is switch open
- For outputs:
  - logic 1 is light on
  - logic 0 is light off.
- NOT uses a switch such that:
  - logic 1 is switch open
  - logic 0 is switch closed

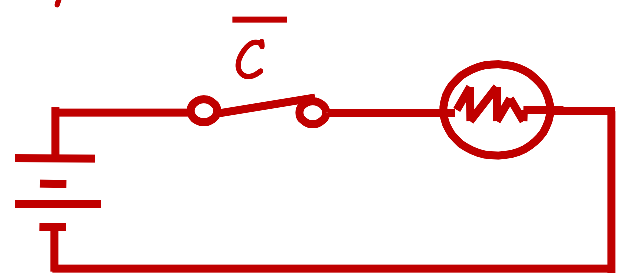
Switches in parallel ⇨ OR



Switches in series ⇨ AND

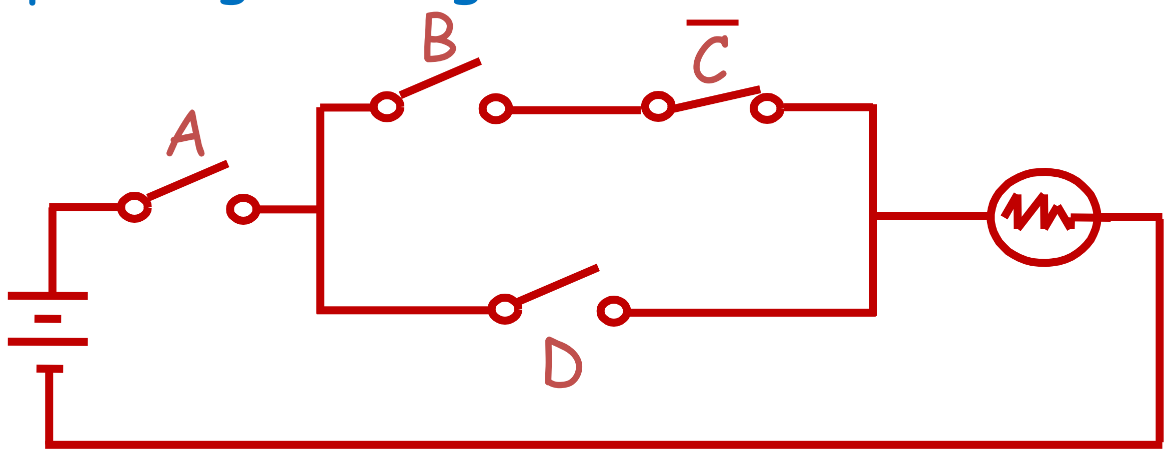


Normally-closed switch ⇨ NOT



Logic Function Implementation (Continued)

- Example: Logic Using Switches



- Light is on (L = 1) for

  $L(A, B, C, D) = AD + AB\overline{C}$

  and off (L = 0), otherwise.

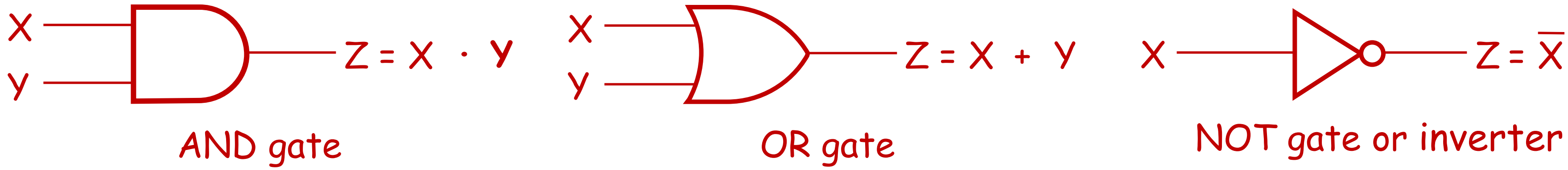
- Useful model for relay circuits and for CMOS gate circuits, the foundation of current digital logic technology

Logic Gates

- ✓ In the earliest computers, switches were opened and closed by magnetic fields produced by energizing coils in *relays*. The switches in turn opened and closed the current paths.
- ✓ Later, *vacuum tubes* that open and close current paths electronically replaced relays.
- ✓ Today, *transistors* are used as electronic switches that open and close current paths.

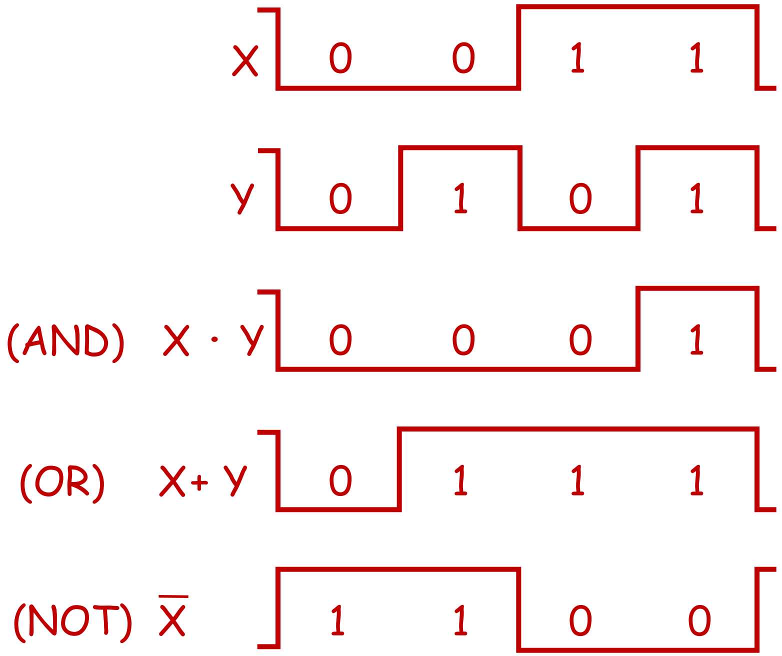
Logic Gate Symbols and Behavior

- ✓ Logic gates have special symbols:



(a) Graphic symbols

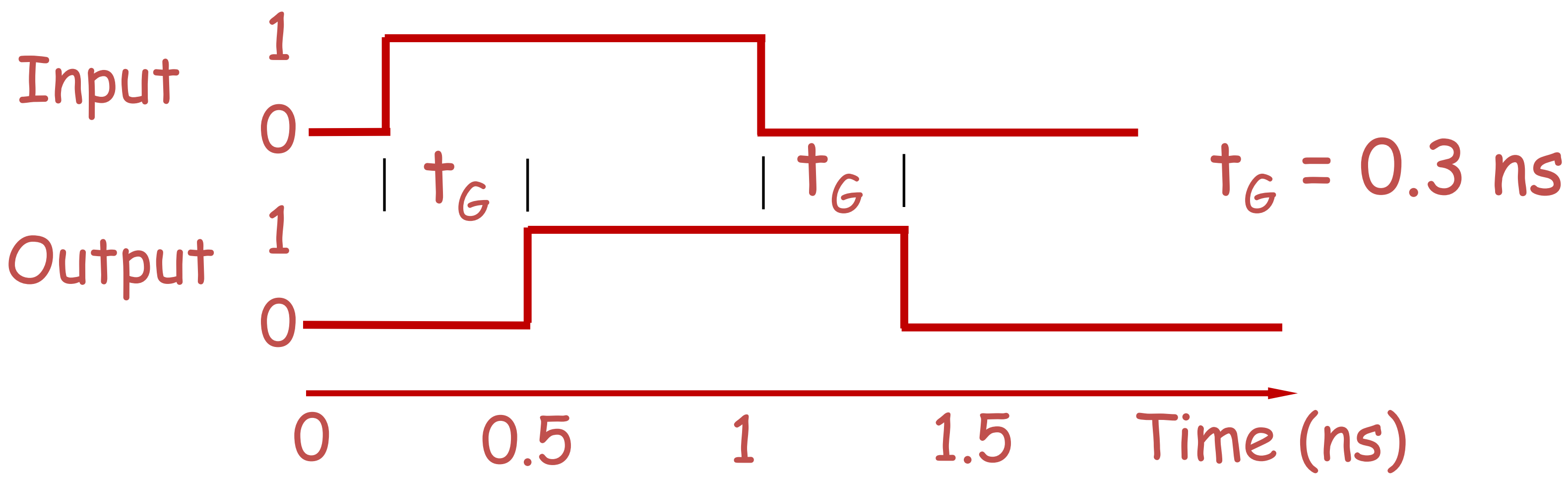
- ✓ And waveform behavior in time as follows:



(b) Timing diagram

Gate Delay

- ✓ In actual physical gates, if one or more input changes causes the output to change, the output change does not occur instantaneously.
- ✓ The delay between an input change(s) and the resulting output change is the *gate delay* denoted by t_G:



Logic Diagrams and Expressions

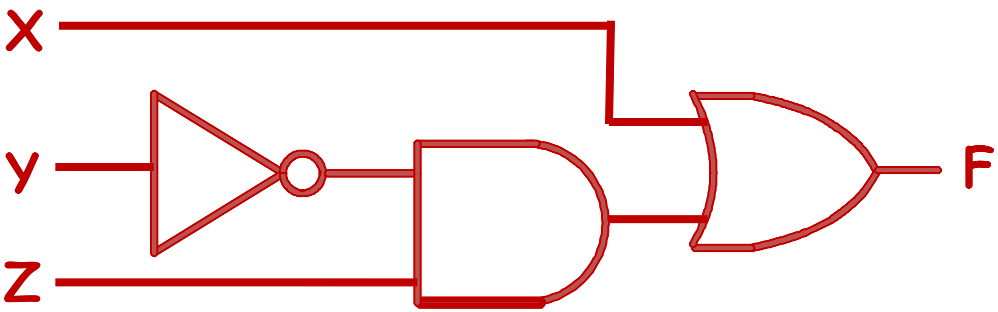
Truth Table

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

Equation

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

Logic Diagram



- ✓ Boolean equations, truth tables and logic diagrams describe the same function!
- ✓ Truth tables are unique; expressions and logic diagrams are not. This gives flexibility in implementing functions.

Boolean Circuits

Definition:

- ✓ A **Boolean circuit** is a directed graph $C(G,N)$ where G are the gates and $N \subseteq G \times G$ is the set of directed edges (nets) connecting the gates.

- ✓ Some of the vertices are designated:

  Inputs: $I \subseteq G$

  Outputs: $O \subseteq G$, $I \cap O = \emptyset$

- ✓ Each gate g is assigned a Boolean function f_g which computes the output of the gate in terms of its inputs.

Definitions

- The fanin $FI(g)$ of a gate g are all predecessor vertices of g:

$$FI(g) = \{g' \mid (g',g) \in N\}$$

- The fanout $FO(g)$ of a gate g are all successor vertices of g:

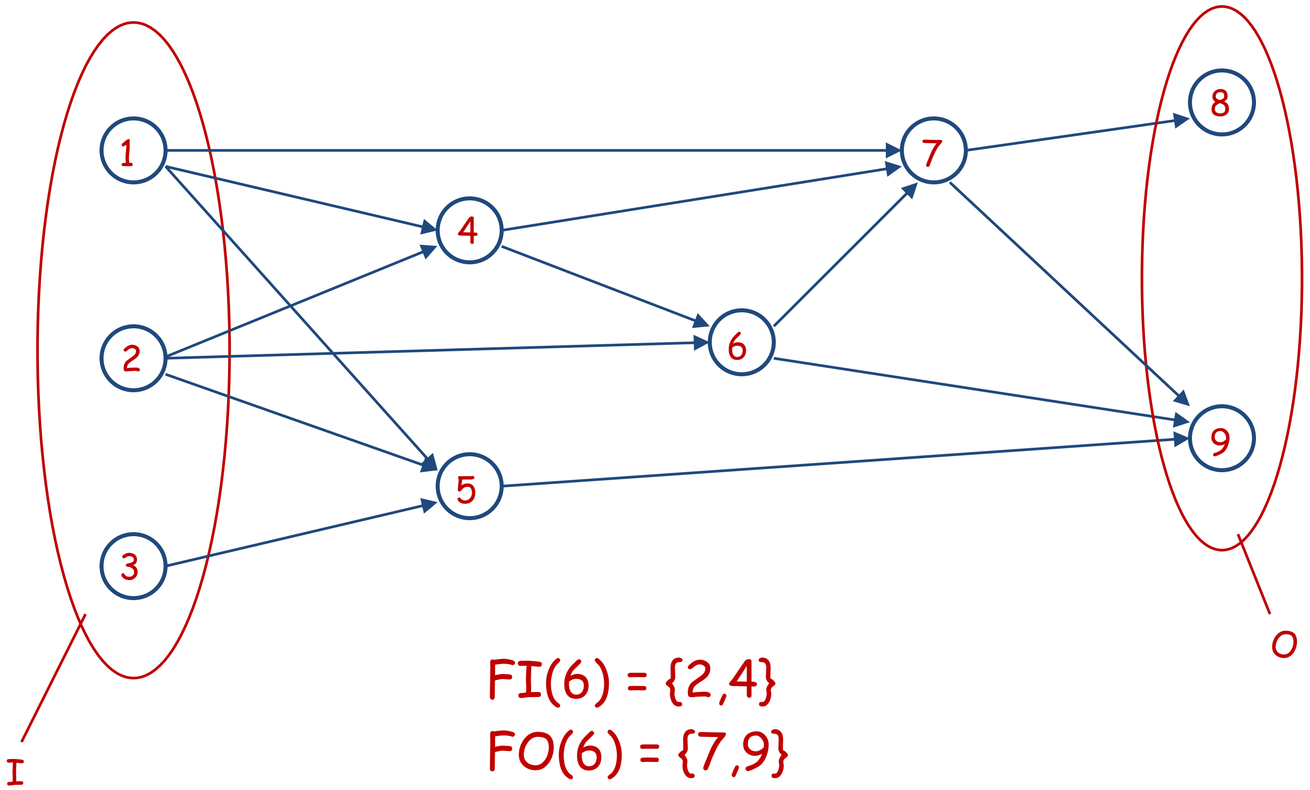
$$FO(g) = \{g' \mid (g,g') \in N\}$$

- The cone $CONE(g)$ of a gate g is the transitive fanin of g and g itself.

- The support $SUPPORT(g)$ of a gate g are all inputs in its cone:

$$SUPPORT(g) = CONE(g) \cap I$$

Example



FI(6) = {2,4}

FO(6) = {7,9}

CONE(6) = {1,2,4,6}

SUPPORT(6) = {1,2}

Canonical Forms

- ✓ It is useful to specify Boolean functions in a form that:
  - Allows comparison for equality.
  - Has a correspondence to the truth tables
- ✓ Canonical Forms in common usage:
  - Sum of Minterms (SOM)
  - Product of Maxterms (POM)

Minterms

- ✓ Minterms are AND terms with every variable present in either true or complemented form.
- ✓ Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\overline{X}$), there are 2^n minterms for *n* variables.
- ✓ Example: Two variables (X and Y)produce 2 x 2 = 4 combinations:
  - $X\ Y$ (both normal)
  - $X\ \overline{Y}$ (X normal, Y complemented)
  - $\overline{X}\ Y$ (X complemented, Y normal)
  - $\overline{X}\ \overline{Y}$ (both complemented)
- ✓ Thus there are four minterms of two variables.

Maxterms

- Maxterms are OR terms with every variable in true or complemented form.
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\overline{x}$), there are 2^n maxterms for *n* variables.
- Example: Two variables (X and Y) produce 2 x 2 = 4 combinations:

$X\ Y$	(both normal)
$X\ \overline{Y}$	(x normal, y complemented)
$\overline{X}\ Y$	(x complemented, y normal)
$\overline{X}\ \overline{Y}$	(both complemented)

Maxterms and Minterms

- ✓ Examples: Two variable minterms and maxterms.

Index		Minterm	Maxterm
0	00	$\bar{x}\,\bar{y}$	$x + y$
1	01	$\bar{x}\,y$	$x + \bar{y}$
2	10	$x\,\bar{y}$	$\bar{x} + y$
3	11	$x\,y$	$\bar{x} + \bar{y}$

- ✓ The index above is important for describing which variables in the terms are true and which are complemented.

Standard Order

- ✓ Minterms and maxterms are designated with a subscript
- ✓ The subscript is a number, corresponding to a binary pattern
- ✓ The bits in the pattern represent the complemented or normal state of each variable listed in a standard order.
- ✓ All variables will be present in a minterm or maxterm and will be listed in the same order (usually alphabetically)
- ✓ Example: For variables a, b, c:
  - Maxterms: $(a + b + \bar{c})$, $(a + b + c)$
  - Terms: $(b + a + c)$, $a\,\bar{c}\,b$, and $(c + b + a)$ are NOT in standard order.
  - Minterms: $a\,\bar{b}\,c$, $a\,b\,c$, $a\,\bar{b}\,\bar{c}$
  - Terms: $(a + c)$, $\bar{b}\,c$, and $(\bar{a} + b)$ do not contain all variables

Purpose of the Index

- ✓ The index for the minterm or maxterm, expressed as a binary number, is used to determine whether the variable is shown in the true form or complemented form.
- ✓ For Minterms:
  - "1" means the variable is "Not Complemented" and
  - "0" means the variable is "Complemented".
- ✓ For Maxterms:
  - "0" means the variable is "Not Complemented" and
  - "1" means the variable is "Complemented".

Index Example in Three Variables

- Example (for three variables):
- Assume the variables are called X, Y, and Z.
- The standard order is X, then Y, then Z.
- The <u>Index 0</u> (base 10) = 000 (base 2) (for three variables). All three variables are complemented for <u>minterm 0</u> ($\overline{X},\overline{Y},\overline{Z}$) and no variables are complemented for <u>Maxterm 0</u> (X,Y,Z).
  - Minterm 0, called m_0 is $\overline{X}\,\overline{Y}\,\overline{Z}$.
  - Maxterm 0, called M_0 is (X + Y + Z).
  - Minterm 6 ?

    $X\,Y\,\overline{Z}$

  - Maxterm 6 ?

    $(\overline{X} + \overline{Y} + Z)$

Index Examples - Four Variables

Index i	Binary Pattern	Minterm m_i	Maxterm M_i
0	0000	$\bar{a}\bar{b}\bar{c}\bar{d}$	$a + b + c + d$
1	0001	$\bar{a}\bar{b}\bar{c}d$	$a + b + c + \bar{d}$
3	0011	?	?
5	0101	$\bar{a}b\bar{c}d$	$a + \bar{b} + c + \bar{d}$
7	0111	?	$a + \bar{b} + \bar{c} + \bar{d}$
10	1010	$\bar{a}b\bar{c}d$	$\bar{a} + b + \bar{c} + d$
13	1101	$\bar{a}\bar{b}c\bar{d}$	?
15	1111	$abcd$	$\bar{a} + \bar{b} + \bar{c} + \bar{d}$

Minterm and Maxterm Relationship

- Review: De Morgan's Theorem
  $\overline{x \cdot y} = \overline{x} + \overline{y}$ and $\overline{x + y} = \overline{x} \cdot \overline{y}$
- Two-variable example:
  $M_2 = \overline{x} + y$ and $m_2 = x \cdot \overline{y}$
  Thus M_2 is the complement of m_2 and vice-versa.
- Since De Morgan's Theorem holds for *n* variables, the above holds for terms of *n* variables giving:

$$M_i = \overline{m_i} \text{ and } m_i = \overline{M_i}$$

Thus M_i is the complement of m_i.

Function Tables for Both

✓ Minterms of 2 variables

Maxterms of 2 variables

x y	m_0	m_1	m_2	m_3
0 0	1	0	0	0
0 1	0	1	0	0
1 0	0	0	1	0
1 1	0	0	0	1

x y	M_0	M_1	M_2	M_3
0 0	0	1	1	1
0 1	1	0	1	1
1 0	1	1	0	1
1 1	1	1	1	0

✓ Each column in the maxterm function table is the complement of the column in the minterm function table since M_i is the complement of m_i.

Observations

- ✓ In the function tables:
  - Each minterm has one and only one 1 present in the 2^n terms (a minimum of 1s). All other entries are 0.
  - Each maxterm has one and only one 0 present in the 2^n terms All other entries are 1 (a maximum of 1s).
- ✓ We can implement any function by "ORing" the minterms corresponding to "1" entries in the function table. These are called the minterms of the function.
- ✓ We can implement any function by "ANDing" the maxterms corresponding to "0" entries in the function table. These are called the maxterms of the function.
- ✓ This gives us two canonical forms:
  - Sum of Minterms (SOM)
  - Product of Maxterms (POM)

for stating any Boolean function.

Minterm Function Example

- ✓ Example: Find $F_1 = m_1 + m_4 + m_7$
- ✓ $F1 = \overline{x}\ \overline{y}\ z + x\ \overline{y}\ \overline{z} + x\ y\ z$

x y z	index	m_1	+	m_4	+	m_7	$= F_1$
0 0 0	0	0	+	0	+	0	= 0
0 0 1	1	1	+	0	+	0	= 1
0 1 0	2	0	+	0	+	0	= 0
0 1 1	3	0	+	0	+	0	= 0
1 0 0	4	0	+	1	+	0	= 1
1 0 1	5	0	+	0	+	0	= 0
1 1 0	6	0	+	0	+	0	= 0
1 1 1	7	0	+	0	+	1	= 1

Minterm Function Example

- ✓ $F(A, B, C, D, E) = m_2 + m_9 + m_{17} + m_{23}$
- ✓ $F(A, B, C, D, E) =$

  $A'B'C'DE' + A'BC'D'E + AB'C'D'E + AB'CDE$

Maxterm Function Example

- ✓ Example: Implement F1 in maxterms:

$$F_1 = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

$$F_1 = (x + y + z) \cdot (x + \overline{y} + z) \cdot (x + \overline{y} + \overline{z}) \cdot (\overline{x} + y + \overline{z}) \cdot (\overline{x} + \overline{y} + z)$$

x y z	i	$M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 = F1$
0 0 0	0	0 · 1 · 1 · 1 · 1 = 0
0 0 1	1	1 · 1 · 1 · 1 · 1 = 1
0 1 0	2	1 · 0 · 1 · 1 · 1 = 0
0 1 1	3	1 · 1 · 0 · 1 · 1 = 0
1 0 0	4	1 · 1 · 1 · 1 · 1 = 1
1 0 1	5	1 · 1 · 1 · 0 · 1 = 0
1 1 0	6	1 · 1 · 1 · 1 · 0 = 0
1 1 1	7	1 · 1 · 1 · 1 · 1 = 1

Maxterm Function Example

- ✓ $F(A,B,C,D) = M_3 \cdot M_8 \cdot M_{11} \cdot M_{14}$
- ✓ $F(A,B,C,D) =$

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

Canonical Sum of Minterms

- ✓ Any Boolean function can be expressed as a **Sum of Minterms**.
  - For the function table, the minterms used are the terms corresponding to the 1's
  - For expressions, expand all terms first to explicitly list all minterms. Do this by "ANDing" any term missing a variable v with a term $(v + \bar{v})$.
- ✓ Example: Implement $f = x + \bar{x}\,\bar{y}$ as a sum of minterms.

  First expand terms: $f = x(y + \bar{y}) + \bar{x}\,\bar{y}$

  Then distribute terms: $f = xy + x\bar{y} + \bar{x}\,\bar{y}$

  Express as sum of minterms: $f = m_3 + m_2 + m_0$

Another SOM Example

- ✓ Example: $F = A + \overline{B}C$
- ✓ There are three variables, A, B, and C which we take to be the standard order.
- ✓ Expanding the terms with missing variables:
- ✓ Collect terms (removing all but one of duplicate terms):
- ✓ Express as SOM:

$$F = A(B + B')(C + C') + (A + A')\,B'\,C$$
$$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$$
$$= ABC + ABC' + AB'C + AB'C' + A'B'C$$
$$= m7 + m6 + m5 + m4 + m1 = m1 + m4 + m5 + m6 + m7$$

Shorthand SOM Form

- From the previous example, we started with:

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

- We ended up with:

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

- This can be denoted in the formal shorthand:

$$F(A, B, C) = \sum_m (1, 4, 5, 6, 7)$$

- Note that we explicitly show the standard variables in order and drop the "m" designators.

Canonical Product of Maxterms

- ✓ Any Boolean Function can be expressed as a Product of Maxterms (POM).
  - For the function table, the maxterms used are the terms corresponding to the 0's.
  - For an expression, expand all terms first to explicitly list all maxterms. Do this by first applying the second distributive law , "ORing" terms missing variable v with a term equal to $v \cdot \overline{v}$ and then applying the distributive law again.
- ✓ Example: Convert to product of maxterms:

$$f(x, y, z) = x + \overline{x}\,\overline{y}$$

Apply the distributive law:

$$x + \overline{x}\,\overline{y} = (x + \overline{x})(x + \overline{y}) = 1 \cdot (x + \overline{y}) = x + \overline{y}$$

Add missing variable z:

$$x + \overline{y} + z \cdot \overline{z} = (x + \overline{y} + z)(x + \overline{y} + \overline{z})$$

Express as POM: $f = M_2 \cdot M_3$

Function Complements

- ✓ The complement of a function expressed as a sum of minterms is constructed by selecting the minterms missing in the sum-of-minterms canonical forms.
- ✓ Alternatively, the complement of a function expressed by a Sum of Minterms form is simply the Product of Maxterms with the same indices.
- ✓ Example: Given $F(x,y,z) = \sum_m(1,3,5,7)$

$$\overline{F}(x,y,z) = \sum_m(0,2,4,6)$$

$$\overline{F}(x,y,z) = \prod_M(1,3,5,7)$$

- ✓ So, to convert between sum-of-minterms and product-of-maxterms form (or vice-versa) we follow one of these steps:
  - Find the function complement by swapping terms in the list with terms not in the list.
  - Change from SOM to POM

Standard Forms

- ✓ Standard Sum-of-Products (SOP) form: equations are written as an OR of AND terms
- ✓ Standard Product-of-Sums (POS) form: equations are written as an AND of OR terms
- ✓ Examples:
  - SOP: $A\,B\,C + \overline{A}\,\overline{B}\,C + B$
  - POS: $(A + B) \cdot (A + \overline{B} + \overline{C}) \cdot C$
- ✓ These "mixed" forms are neither SOP nor POS
  - $(A\,B + C)\,(A + C)$
  - $A\,B\,\overline{C} + A\,C\,(A + B)$

Standard Sum-of-Products (SOP)

- ✓ A sum of minterms form for *n* variables can be written down directly from a truth table.
  - Implementation of this form is a two-level network of gates such that:
  - The first level consists of *n*-input AND gates, and
  - The second level is a single OR gate (with fewer than 2^n inputs).
- ✓ This form often can be simplified so that the corresponding circuit is simpler.

Standard Sum-of-Products (SOP)

- ✓ A Simplification Example:

  $F(A, B, C) = \sum_m (1, 4, 5, 6, 7)$

- ✓ Writing the minterm expression:

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

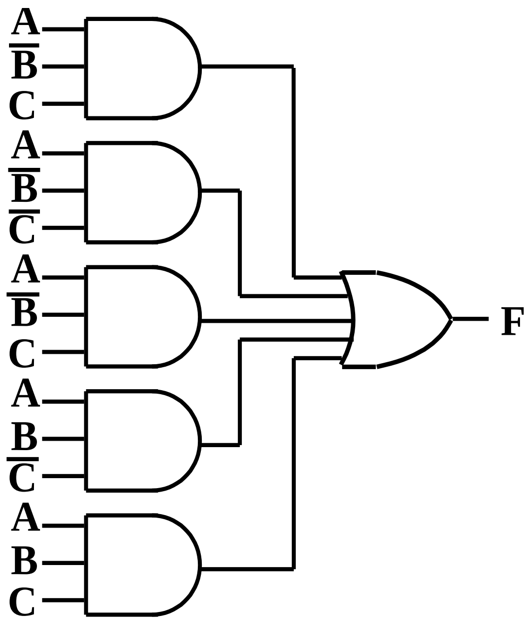
- ✓ Simplifying:

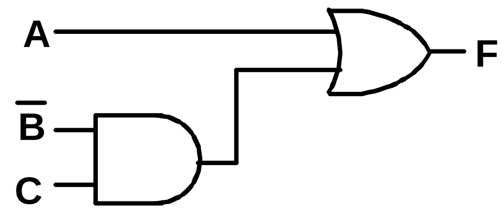
  $$F = A'\,B'\,C + A\,(B'\,C' + B\,C' + B'\,C + B\,C)$$
  $$= A'\,B'\,C + A\,(B' + B)\,(C' + C)$$
  $$= A'\,B'\,C + A \cdot 1 \cdot 1$$
  $$= A'\,B'\,C + A$$
  $$= B'C + A$$

- ✓ Simplified F contains 3 literals compared to 15 in minterm F

AND/OR Two-level Implementation of SOP Expression

✓ The two implementations for F are shown below – it is quite apparent which is simpler!





SOP and POS observations

- ✓ The previous examples show that:
  - Canonical Forms (Sum-of-minterms, Product-of-Maxterms), or other standard forms (SOP, POS) differ in complexity
  - Boolean algebra can be used to manipulate equations into simpler forms.
  - Simpler equations lead to simpler two-level implementations
- ✓ Questions:
  - How can we attain a "simplest" expression?
  - Is there only one minimum cost circuit?
  - The next part will deal with these issues.