

● **Theorems:**

6Ba. $(x \vee x) = x$
b. $(x \wedge x) = x$

idempotent laws

7Ba. $(x \vee 1) = 1$
b. $(x \wedge 0) = 0$

more identity laws

8Ba. $(x \wedge y) \vee x = x$
b. $(x \vee y) \wedge x = x$

absorption laws

9Ba. $(x \vee y)' = x' \wedge y'$
b. $(x \wedge y)' = x' \vee y'$

DeMorgan's laws

● **Lemma:**

In a Boolean algebra,

- (a) $x \vee y = y$ iff $x \wedge y = x$
(b) $x \wedge y = x$ iff $x \vee y = y$

● **Lemma:**

Define the following relations:

\leq : $x \leq y$ iff $x \vee y = y$

$<$: $x < y$ if $x \leq y$ but $x \neq y$

\geq : $x \geq y$ means $y \leq x$

$>$: $x > y$ means $y < x$

In a Boolean algebra,

(a) If $x \leq y$ and $y \leq z$, then $x \leq z$ (transitivity)

(b) If $x \leq y$ and $y \leq x$, then $x = y$

(c) If $x < y$ and $y < z$, then $x < z$

● **Lemma:**

In a Boolean Algebra,

(a) $(x \wedge y) \leq x \leq (x \vee y)$ for every x and y

(b) $0 \leq x \leq 1$ for every x

Atoms

- An **atom** of a Boolean algebra is a nonzero element a that cannot be written in the form $a = b \vee c$ with $a \neq b$ and $a \neq c$.

- Example:

- $B = \{0,1\}$

The only atom in B is 1.

- $B^n = B \times B \times \dots \times B$ (with n factors)

(a set of n -tuples of 0's and 1's)

$$(a_1, \dots, a_n) \vee (b_1, \dots, b_n) = (a_1 \vee b_1, \dots, a_n \vee b_n)$$

$$(a_1, \dots, a_n) \wedge (b_1, \dots, b_n) = (a_1 \wedge b_1, \dots, a_n \wedge b_n)$$

$$(a_1, \dots, a_n)' = (a_1', \dots, a_n')$$

The atoms are the n -tuples with exactly one entry 1.

- **Proposition:**

A nonzero element x of a Boolean algebra is an atom iff there is no element y with $0 < y < x$.

- **Theorem:**

Let B be a finite Boolean algebra with set of atoms $A = \{a_1, \dots, a_n\}$. Each nonzero x in B can be written as a join of distinct atoms:

$$x = a_{i_1} \vee \dots \vee a_{i_k}.$$

Moreover, such an expression is unique, except for the order of the atoms.

- An **n-variable Boolean function** is a function $f: B^n \rightarrow B$.
- $\text{BOOL}(n) = \{\text{all } n\text{-variable Boolean functions}\}$.
- if $f \in \text{BOOL}(n)$ and $(x_1, \dots, x_n) \in B^n$, we write $f(x_1, \dots, x_n)$ for the value $f((x_1, \dots, x_n))$.
Note: $f \in \text{BOOL}(n)$ can be viewed as a 2^n -tuple.

- Example - sample function of $\text{BOOL}(3)$:

x	y	z	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

- Note: $|\text{BOOL}(n)| = 2^{2^n}$
- Note: The atoms of $\text{BOOL}(n)$ are 2^n -tuples.

Boolean Expressions

- **Boolean expressions in n variables** x_1, x_2, \dots, x_n are defined recursively as follows:

(B) The symbols 0, 1 and x_1, x_2, \dots, x_n are Boolean expressions in x_1, x_2, \dots, x_n .

(R) If E_1 and E_2 are Boolean expressions in x_1, x_2, \dots, x_n , so are $(E_1 \vee E_2)$, $(E_1 \wedge E_2)$, and E_1' .

- Examples of Boolean expressions:

$$(x \vee y) \wedge (x' \vee z) \wedge 1$$

$$(x' \wedge z) \vee (x' \wedge y) \vee z'$$

$$x \vee y$$

$$z$$

- Note: The connective \wedge can be replaced by \cdot or no dot at all.

$$(x \vee y) \wedge (x' \vee z) \wedge 1 = (x \vee y) \cdot (x' \vee z) \cdot 1 = (x \vee y)(x' \vee z)1$$

$$(x' \wedge z) \vee (x' \wedge y) = (x' \cdot z) \vee (x' \cdot y) = (x'z) \vee (x'y)$$

- In general if E is a Boolean expression in the n variables x_1, x_2, \dots, x_n , then E defines a **Boolean function** $B^n \rightarrow B$ whose function value at (a_1, a_2, \dots, a_n) is the element of B obtained by replacing x_i by a_i in E for all i .

- Two Boolean expressions are **equivalent** if their corresponding Boolean functions are the same.
- Boolean expressions satisfy all the laws of a Boolean algebra.

- Boolean expressions consisting of a single variable or its complement, such as x or y' , are called **literals**.
- The functions that correspond to a literal have the value 1 at half the elements of B^n .
- A **minterm** in n variables is a Boolean expression consisting of a meet (i.e., product) of exactly n literals each involving a different variable.
- Each minterm corresponds to an atom of $\text{BOOL}(n)$.
- The following table lists the eight elements of B^3 and the corresponding minterms that take the value 1 at the indicated elements. Note that the literals corresponding to 0 entries are complemented, while the other literals are not.

(x,y,z)	Minterm with value 1 at (x,y,z)
$(0,0,0)$	$x'y'z'$
$(0,0,1)$	$x'y'z$
$(0,1,0)$	$x'yz'$
$(0,1,1)$	$x'yz$
$(1,0,0)$	$xy'z'$
$(1,0,1)$	$xy'z$
$(1,1,0)$	xyz'
$(1,1,1)$	xyz

Canonical Forms

- **Disjunctive Normal Form - DNF:**

Every Boolean expression E can be written as a join of atoms (see earlier theorem) and thus, a join of minterms. This join of minterms is unique (except for the order in which the minterms are written) and is called the **DNF** (or **minterm canonical form**) of E.

- Example:

$$E = x'yz' \vee xy'z' \vee xy'z \vee xyz' \quad (\text{This is in DNF})$$

x	y	z	$x'yz'$	$xy'z'$	$xy'z$	xyz'	E
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	0	0	1	0	1
1	1	0	0	0	0	1	1
1	1	1	0	0	0	0	0

● Example:

$$E = (x \vee yz')(yz)' \quad (\text{This is not in DNF})$$

To find the DNF of E, we first use the laws of Boolean algebra to obtain a join of meets (sum of products) expression for E.

$$\begin{aligned}
 E &= (x \vee yz')(yz)' \\
 &= (x \vee yz')(y' \vee z') && \text{DeMorgan's law} \\
 &= (x(y' \vee z')) \vee (yz'(y' \vee z')) && \text{distributive law} \\
 &= (xy' \vee xz') \vee (yz'y' \vee yz'z') && \text{distributive law} \\
 &= (xy' \vee xz') \vee (0 \vee yz') && y y'=0 \text{ and } z'z'=z' \\
 &= xy' \vee xz' \vee yz' && \text{associative law} \\
 &&& \text{and property of 0}
 \end{aligned}$$

Now, we form the DNF by ANDing to each product term a sum term that has value 1 expressed in terms of the missing literals.

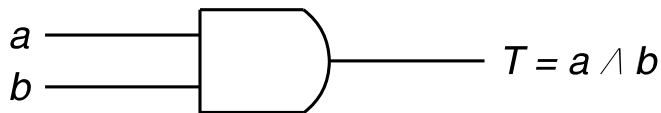
$$\begin{aligned}
 &= xy'(z \vee z') \vee x(y \vee y')z' \vee (x \vee x')yz' \\
 &= (xy'z \vee xy'z') \vee (xyz' \vee xy'z') \vee (xyz' \vee x'yz') \\
 &= xy'z \vee xy'z' \vee xyz' \vee x'yz'
 \end{aligned}$$

Logic Networks

- The basic building block of logic networks are units called gates.

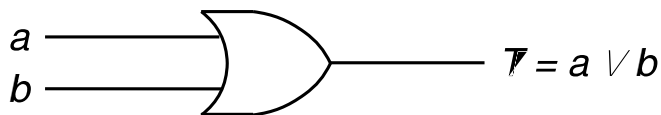
Major Logic Gate Types

- **AND Gate:**



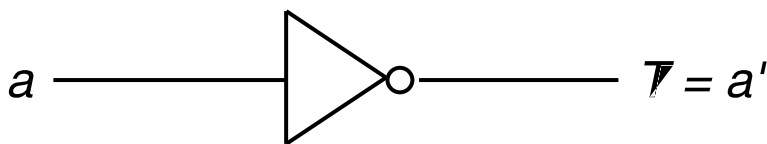
a	b	T
0	0	0
0	1	0
1	0	0
1	1	1

- **OR Gate**



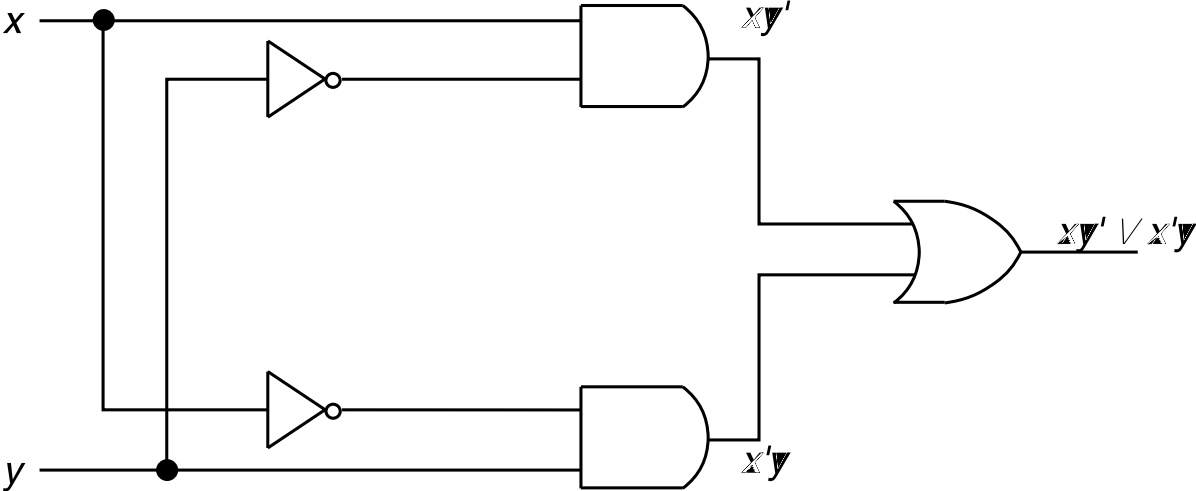
a	b	T
0	0	0
0	1	1
1	0	1
1	1	1

- **NOT Gate (Inverter):**



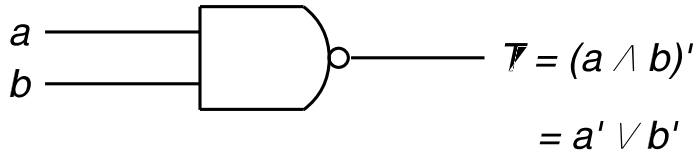
a	T
0	1
1	0

● Example: $f(x,y) = xy' \vee x'y = x \oplus y$



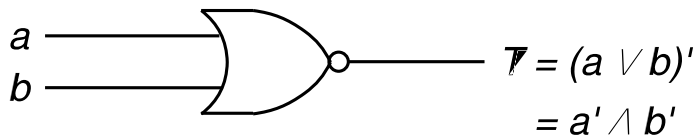
Major Logic Gate Types Cont.

● NAND Gate:



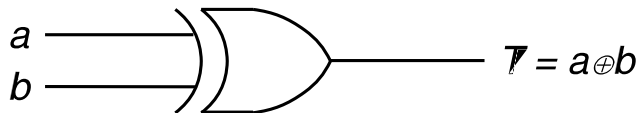
<i>a</i>	<i>b</i>	\overline{F}
0	0	1
0	1	1
1	0	1
1	1	0

● NOR Gate



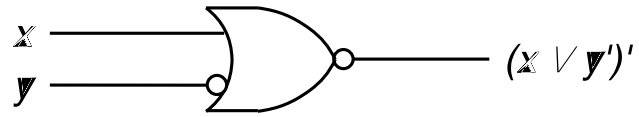
<i>a</i>	<i>b</i>	\overline{F}
0	0	1
0	1	0
1	0	0
1	1	0

● Exclusive OR (XOR) Gate:

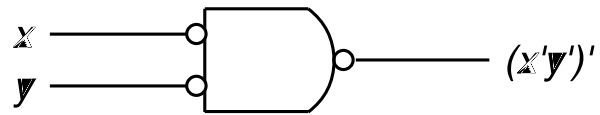


<i>a</i>	<i>b</i>	\overline{F}
0	0	0
0	1	1
1	0	1
1	1	0

- Example: $f(x,y) = (x \vee y)'$

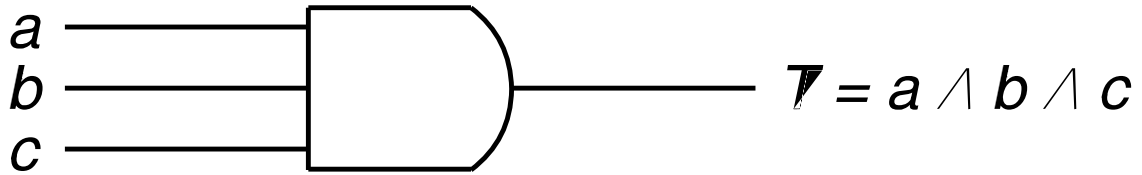


- Example: $f(x,y) = (x'y)'$

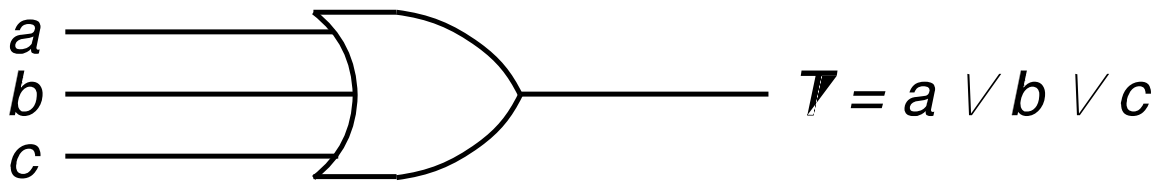


3-input Gates

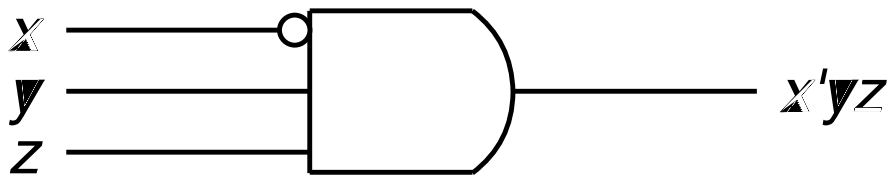
- **AND Gate:**



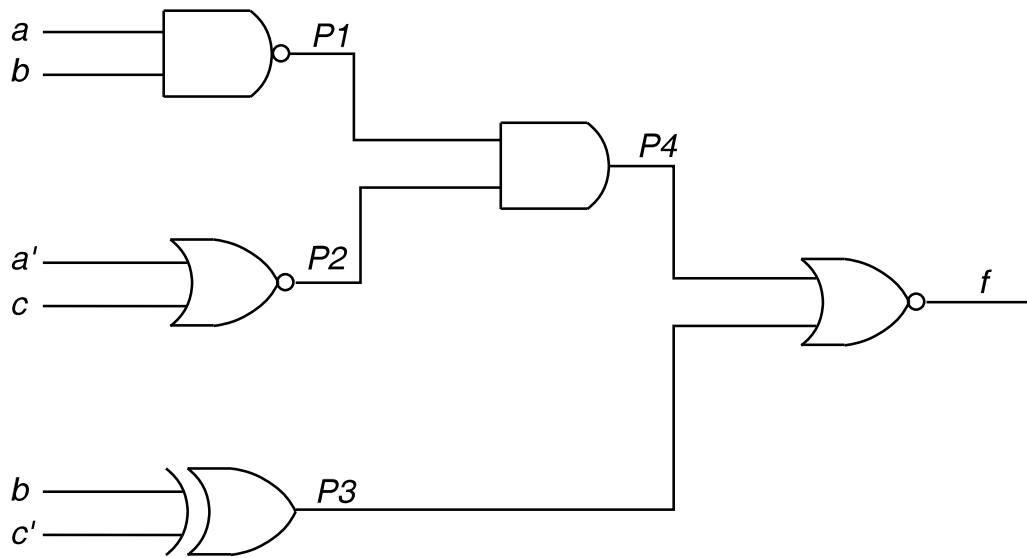
- **OR Gate**



- **Example $f(x,y,z) = x'yz$**



Analysis of Logic Networks



$$\begin{aligned}
 P_1 &= (ab)' \\
 P_2 &= (a' \vee c)' = ac' \\
 P_3 &= b \oplus c' \\
 P_4 &= P_1 P_2 \\
 &= (ab)'(a' \vee c)' \\
 &= (ab)'(ac') \\
 &= (a' \vee b')(ac') \\
 &= ab'c' \\
 f &= (P_3 \vee P_4)' \\
 &= ((b \oplus c') \vee ab'c')' \\
 &= (bc \vee b'c' \vee ab'c')' \\
 &= (bc \vee b'c')' \\
 &= b \oplus c
 \end{aligned}$$

Synthesis of Boolean Expressions

- **Two Level AND-OR and NAND Networks:**

$$\begin{aligned} f(w,x,y,z) &= wy' \vee xyz \vee w'z \\ &= ((wy')'(xyz)'(w'z)')' \end{aligned}$$

- **Two Level OR-AND and NOR Networks:**

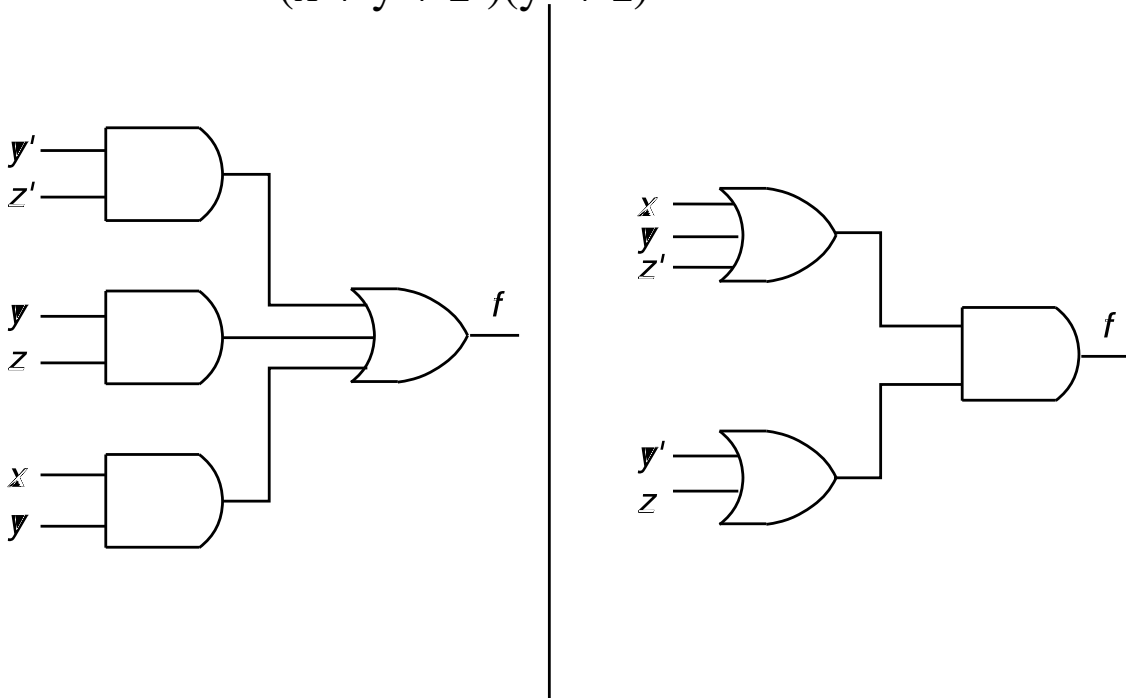
$$\begin{aligned} f(w,x,y,z) &= (w' \vee x \vee y)(x \vee y \vee z)(w' \vee z) \\ &= ((w' \vee x \vee y)' \vee (x \vee y \vee z)' \vee (w' \vee z)')' \end{aligned}$$

- **Example:**

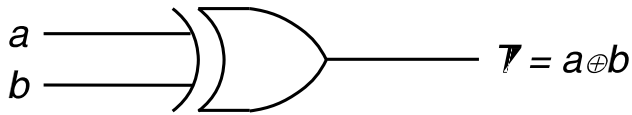
$$\begin{aligned} f(x,y,z) &= x'y'z' \vee x'yz \vee xy'z' \vee xy'z \vee xyz \\ &= (x' \vee x)y'z' \vee (x \vee x')yz \vee x(y \vee y')z \\ &= y'z' \vee yz \vee xz \end{aligned}$$

This function can be shown equivalent to:

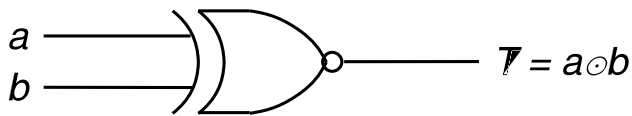
$$\begin{aligned} &= (x \vee y \vee z')(x \vee y' \vee z)(x' \vee y' \vee z) \\ &= (x \vee y \vee z')(xx' \vee y' \vee z) \\ &= (x \vee y \vee z')(y' \vee z) \end{aligned}$$



Use of Exclusive-OR and Exclusive-NOR Gates



a	b	F
0	0	0
0	1	1
1	0	1
1	1	0



a	b	F
0	0	1
0	1	0
1	0	0
1	1	1

SUM Circuit:

- The Exclusive-Or performs the arithmetic sum of its two inputs.

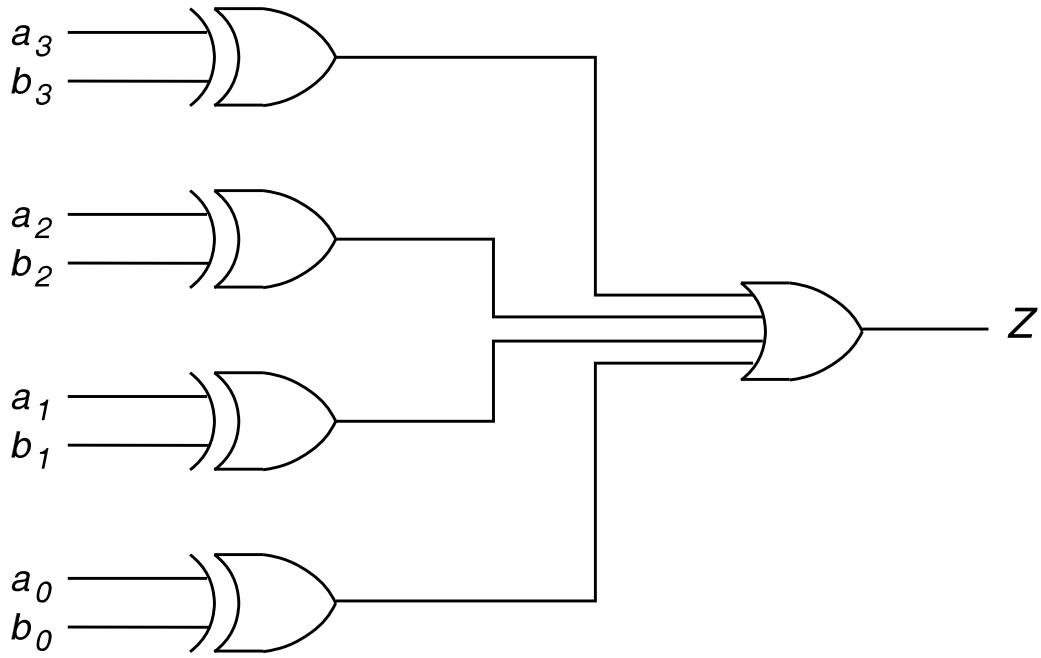
Comparator Circuit:

- The Exclusive-OR (or Exclusive-NOR) performs a comparison function of its two inputs.

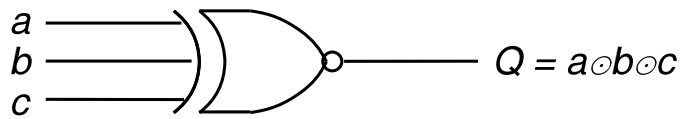
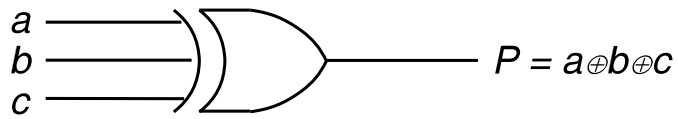
Comparator

$$A = a_3 a_2 a_1 a_0$$

$$B = b_3 b_2 b_1 b_0$$



Three-Input Exclusive-OR (Exclusive-NOR)



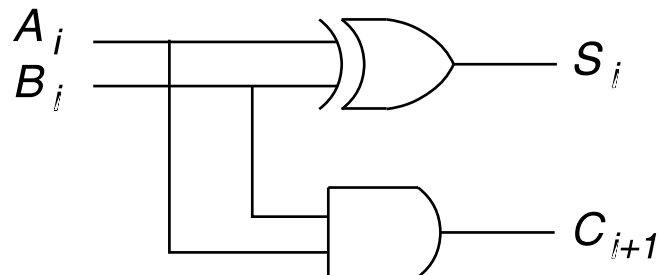
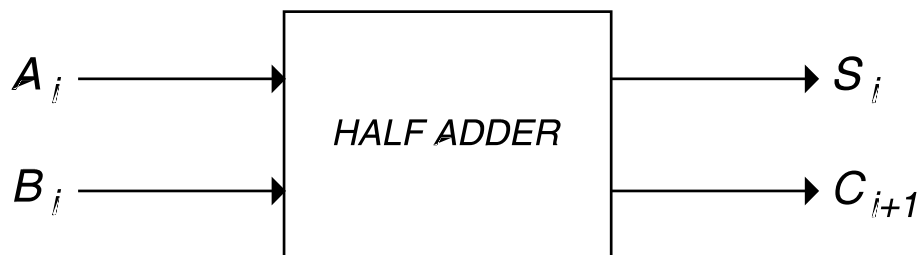
a	b	c	P	Q
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The Half Adder

Addend A_i	0	0	1	1
Augend B_i	0	1	0	1
Carry-Sum	00	01	01	10
(C_{i+1}, S_i)				

$A_i B_i$	S_i	C_{i+1}
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

$$S_i = A_i \oplus B_i$$
$$C_i = A_i B_i$$



The Addition of Two n-bit N

$$\begin{array}{cccccc}
 & C_{n-1} & & C_2 & & C_1 & & \\
 A = & A_{n-1} & \dots & A_2 & & A_1 & & A_0 \\
 B = & B_{n-1} & \dots & B_2 & & B_1 & & B_0 \\
 \hline
 & C_n S_{n-1} & \dots & C_3 S_2 & & C_2 S_1 & & C_1 S_0
 \end{array}$$

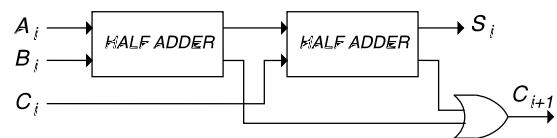
The Full Adder



$C_i A_i B_i$	S_i	C_{i+1}
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1

$$S_i = A_i \oplus B_i \oplus C_i$$

$$\begin{aligned}
 C_{i+1} &= A_i B_i + A_i C_i + B_i C_i \\
 &= A_i B_i + C_i (A_i \oplus B_i)
 \end{aligned}$$



Karnaugh Maps

- A **Karnaugh Map** is a modified form of a truth table arranged in a convenient way.
- **2-variable Map:**

Location of Minterms

$y \backslash x$	0	1
0	$x'y'$	xy'
1	$x'y$	xy

- Note: Each pair of **adjacent** cells differ by just one variable value and can be combined using:

$$xy \vee xy' = x$$

- Example:

$$f(x,y) = xy' \vee xy$$

$y \backslash x$	0	1
0	0	1
1	0	1

$$f(x,y) = x$$

● **3-variable Map:**

$z \backslash xy$	00	01	11	10
0	$x'y'z'$	$x'yz'$	xyz'	$xy'z'$
1	$x'y'z$	$x'yz$	xyz	$xy'z$

● **Example:**

$$f(x,y,z) = x'yz' \vee xyz' \vee xyz$$

$z \backslash xy$	00	01	11	10
0		1	1	
1			1	

$$f(x,y,z) = yz' \vee xy$$

● **4-variable Map:**

$yz \backslash wx$	00	01	11	10
00	$w'x'y'z'$	$w'xy'z'$	$wxy'z'$	$wx'y'z'$
01	$w'x'y'z$	$w'xy'z$	$wxy'z$	$wx'y'z$
11	$w'x'yz$	$w'xyz$	$wxyz$	$wx'yz$
10	$w'x'yz'$	$w'xyz'$	$wxyz'$	$wx'yz'$

● **Example:**

$yz \backslash wx$	00	01	11	10
00		1	1	1
01		1	1	
11			1	
10			1	

$$f(w,x,y,z) = wx \vee xy' \vee wy'z'$$

Summary

The combination of adjacent squares that is useful during the simplification process is:

1. A single square represents a minterm.
2. Two adjacent squares represents a 1-cube and eliminates one variable.
3. Four adjacent squares represents a 2-cube and eliminates two variables.
4. Eight adjacent squares represents a 3-cube and eliminates three variables.
5. etc.

Minimization of Boolean Expressions

1. Start by covering those 1s that can't be combined with any other 1 cells.
2. Then cover those 1s which only have a single adjacent 1 cell by forming 1-cubes.
3. Then form 2-cubes (group of four), 3-cubes (group of eight), etc.
4. A minimal sum of products expression is one that corresponds to a collection of subcubes which are as large and as few as possible so that every 1 cell is covered.

● Example:

yz\wx	00	01	11	10
00	1	1		1
01		1	1	1
11		1	1	
10				

$$f_1 = x'y'z' \vee w'xy' \vee wy'z \vee xz$$

$$f_2 = w'y'z' \vee wx'y' \vee xz$$

- We have two irredundant expressions for $f(w,x,y,z)$; however, only f_2 is minimal.

● Example:

yz\wx	00	01	11	10
00			1	
01	1	1	1	
11		1	1	1
10		1		

$$f = wxy' \vee wyz \vee w'xy \vee w'y'z$$

● Example

yz\wx	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$$f = w \oplus x \oplus y \oplus z$$

Isomorphism

- **Boolean Algebra Isomorphism:**

A **Boolean algebra isomorphism** is a one-to-one correspondence ϕ between Boolean algebras B_1 and B_2 that satisfies:

$$(1) \phi(x \vee y) = \phi(x) \vee \phi(y),$$

$$(2) \phi(x \wedge y) = \phi(x) \wedge \phi(y),$$

and

$$(3) \phi(x') = \phi(x)'.$$

for all $x, y \in B_1$

Two Boolean algebras are said to be **isomorphic** if there is an isomorphism between them.

- **Theorem:**

If B_1 is a finite Boolean algebra with set of atoms $A_1 = \{a_1, \dots, a_n\}$ and if B_2 is another finite Boolean algebra with set of atoms $A_2 = \{b_1, \dots, b_n\}$, then there is a Boolean algebra isomorphism ϕ of B_1 onto B_2 such that $\phi(a_i) = b_i$ for each i .

- **Corollary:**

Every finite Boolean algebra with n atoms is isomorphic to the Boolean algebra $\mathcal{P}(S)$ of all subsets of an n -element set S , and hence has exactly 2^n elements. In particular, B^n is isomorphic to $\mathcal{P}(\{1, 2, \dots, n\})$.