Chapter 2

Introduction to Logic Circuits

Contents

- Basic Logic Gates
- Boolean Algebra
- Canonical and Standard Forms
- Synthesis Using AND, OR and NOT Gates
- NAND and NOR Logic Networks
- Design examples

BASIC LOGIC GATES

Logic Gates

- Basic Logic Gates
  - AND
  - OR
  - NOT
- Related Logic Gates
  - NAND (AND-NOT)
  - NOR (OR-NOT)
  - X-OR (Combination of AND, OR and NOT)
  - X-NOR (Equivalence)

AND

- X AND Y.
  - X & Y are propositions.
  - A proposition can be either TRUE or FLASE.
- (Today is Tuesday) AND (Today is sunny).

AND in Logic

- AND may have many inputs.
- AND produces the value TRUE if all the inputs are TRUE.
- So, with the inputs:

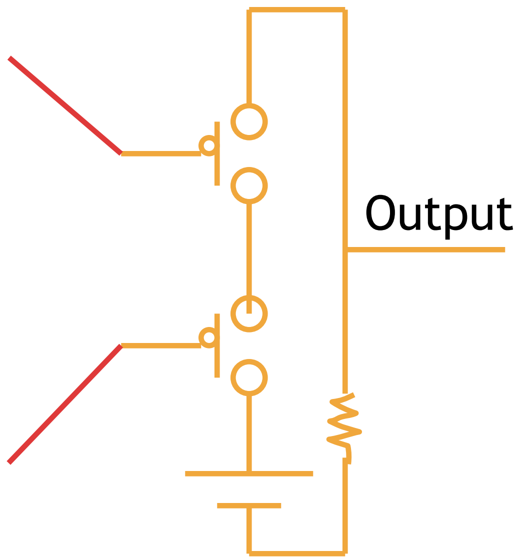
T AND T	output is: T
T AND F	output is: F
F AND T	output is: F
F AND F	output is: F

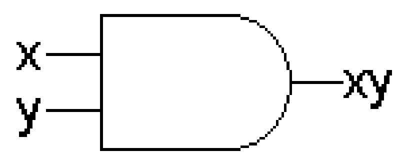
(Just like in the real world.)

TRUE = HIGH = ON = ‘1’

The AND Gate

- The *gate* is a circuit that performs a basic logic operation.
- Inputs are on the left, outputs are on the right of a logic symbol.
- The AND gate has two or more inputs and a single output.





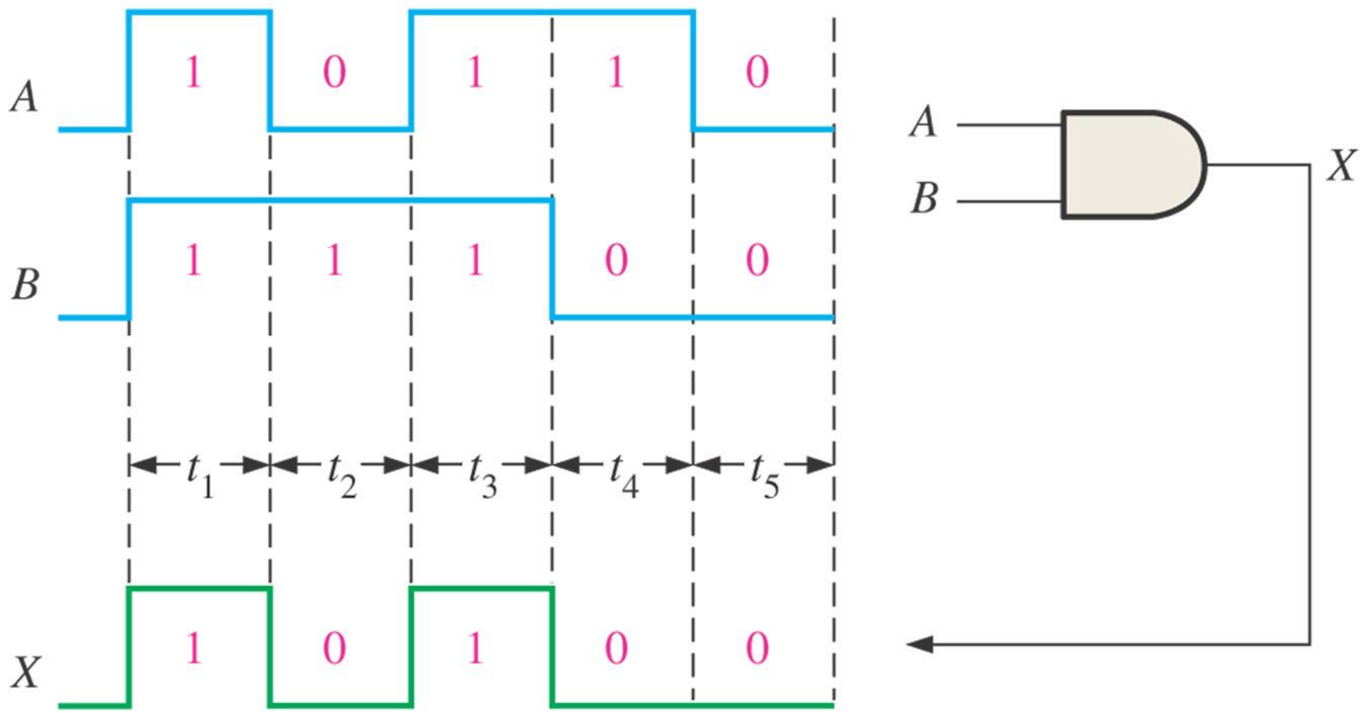
The AND Gate

- An AND gate produces a HIGH output only when all the inputs are HIGH.
- A truth table is a convenient way to write down the definition of a logic operation.

Inputs		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table for a 2-input AND gate

The AND Gate

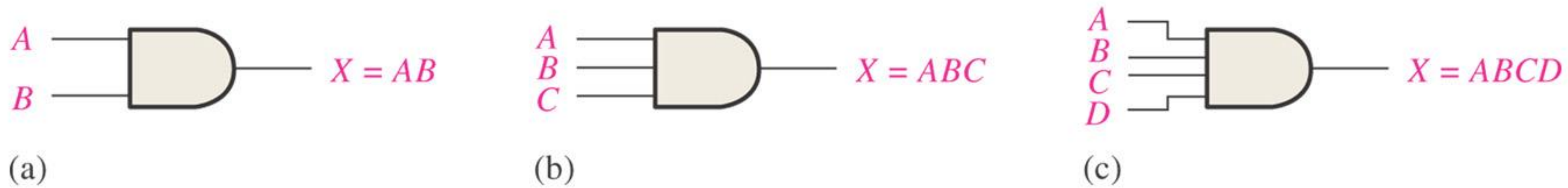


Example of AND gate operation with a timing diagram showing input and output relationships

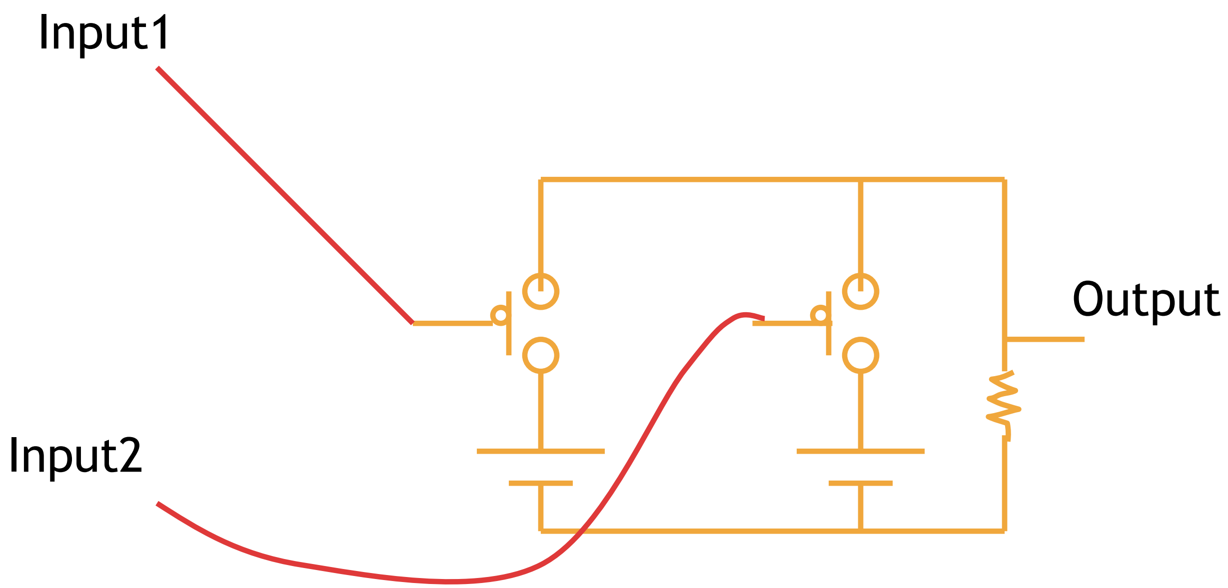
The AND Gate

- The logical AND function is equivalent to the *multiplication* in Boolean algebra.
- If the input variable is A, B and the output variable is X, then, for an AND gate,

$$X = AB \text{ or } X = A \cdot B$$

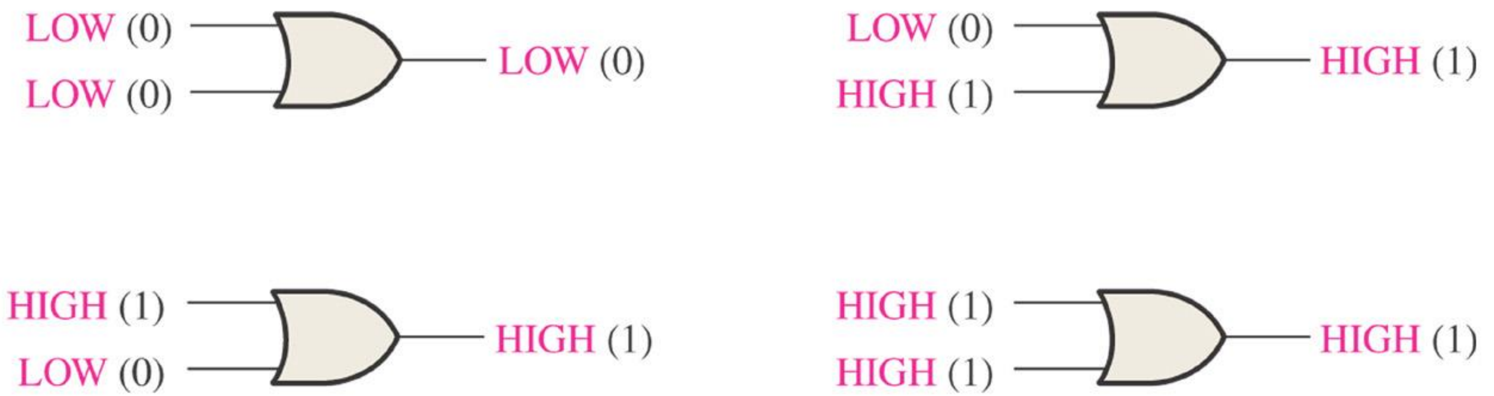


The OR Gate



- Produces a HIGH on the output when any of the inputs is HIGH.

The OR Gate

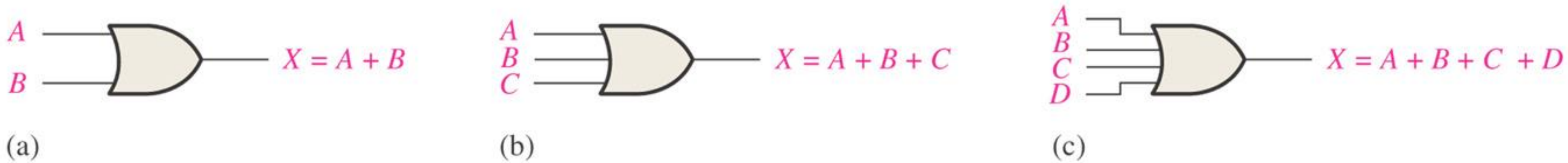


Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table for a 2-input OR gate

The OR Gate in Boolean algebra

- The logical OR function is equivalent to the *addition* in Boolean algebra.
- X = A+B



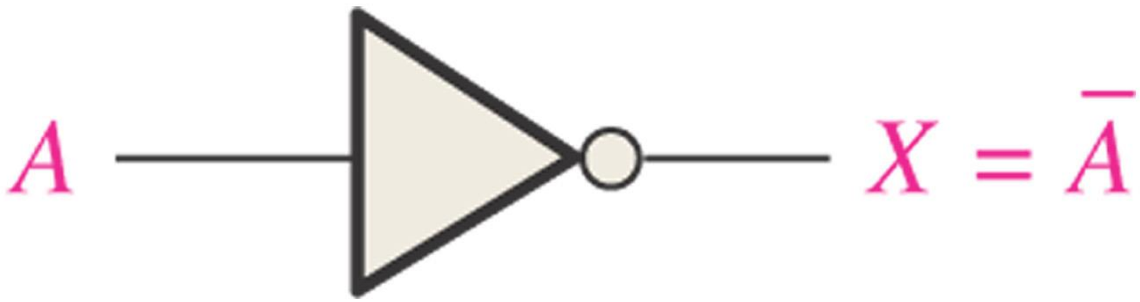
Inverter

- In Boolean algebra, for an inverter,

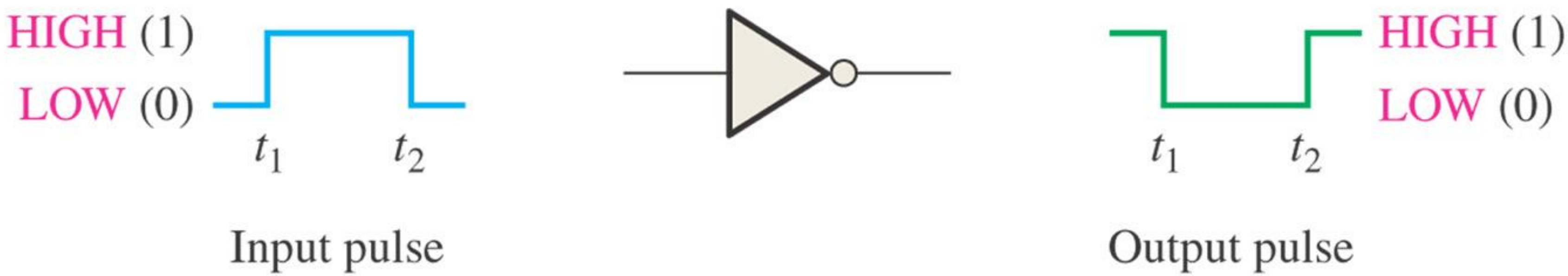
$$X = \overline{A}$$

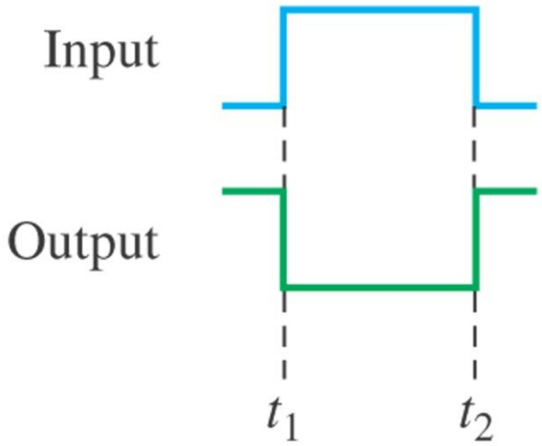
$$X = A'$$

- Can be read "Not A", "A complement", "A prime", "A bar"



Inverter

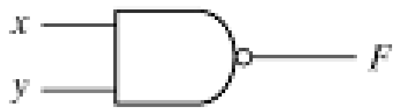




Timing Diagram

Digital Logic Gates

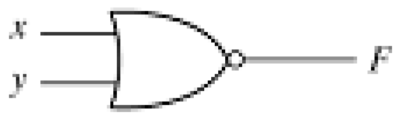
NAND



$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

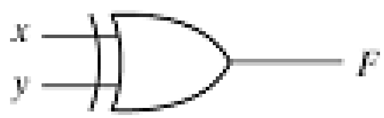
NOR



$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive-OR (XOR)

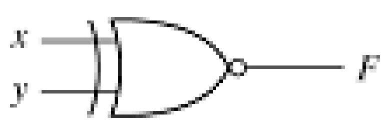


$F = xy' + x'y$
$= x \oplus y$

Mod 2 addition

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR or equivalence



$F = xy + x'y'$
$= (x \oplus y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

BOOLEAN ALGEBRA

Boolean algebra

- In the 1850s, George Boole developed a mathematical system for formulating logic statements with symbols.
- Logic problems can be written and solved like an ordinary algebra.

Boolean Constants and Variables

- In Boolean algebra, there are only two <u>constants</u>.
  - True and False
  - 1 and 0
- Boolean <u>variables</u> are variables that store values that are Boolean constants.

Properties of Boolean Algebra

- Duality
  - Interchange OR and AND operators
  - Replace 1's by 0's and 0's by 1's

e.g. $x + 0 = x \leftrightarrow x \cdot 1 = x$

- Operator precedence

1. Parentheses
2. NOT
3. AND
4. OR

Properties of Boolean Algebra

- Basic OR, AND operation
  - $x + 0 = x \quad \leftrightarrow \quad x \cdot 1 = x$
  - $\boxed{x + x' = 1 \quad \leftrightarrow \quad x \cdot x' = 0}$
  - $\boxed{x + x = x \quad \leftrightarrow \quad x \cdot x = x}$
  - $\boxed{x + 1 = 1} \quad \leftrightarrow \quad x \cdot 0 = 0$
- Involution
  - $(x')' = x$
- Commutative
  - $x + y = y + x \quad \leftrightarrow \quad xy = yx$
- Associative
  - $x + (y + z) = (x + y) + z \quad \leftrightarrow \quad x(yz) = (xy)z$

Properties of Boolean Algebra

- Distributive
  - $x(y+z) = xy + xz \leftrightarrow \boxed{x + yz = (x+y)(x+z)}$

$$RHS = xx + xz + xy + yz$$
$$= x + xz + xy + yz$$
$$= x(1 + y + z) + yz$$
$$= x + yz$$

- DeMorgan
  - $(x+y)' = x'y' \leftrightarrow (xy)' = x' + y'$

NOR NAND

- Absorption
  - $x + xy = x \leftrightarrow x(x+y) = x$

$x(1+y)$ $xx + xy = x + xy$

Algebraic Manipulation

- Ex 2-1) Simplify the following Boolean functions to a minimum number of literals

1. $x(x' + y) = xx' + xy = 0 + xy = xy$ $\quad xx' = 0$

2. $x + x'y = (x + x')(x + y)$ $\quad x + yz = (x + y)(x + z)$
$= 1(x + y) = x + y$ $\quad x + x' = 1$

3. $(x + y)(x + y') = xx + xy + xy' + yy'$
$= x + xy + xy' + yy'$ $\quad xx = x$
$= x(1 + y + y') = x$ $\quad 1 + x = 1$

4. $xy + x'z + yz = xy + x'z + yz(x + x')$ $\quad x + x' = 1$
$= xy + x'z + xyz + x'yz$ $\quad 1 + x = 1$
$= xy(1 + z) + x'z(1 + y) = xy + x'z$

5. $(x + y)(x' + z)(y + z) = (x + y)(x'+z)$ by duality from function 4

Algebraic Manipulation

- DeMorgan Law

$$(A + B + C)' = (A + x)' \qquad let\ B + C = x$$
$$= A'x'$$
$$= A'(B + C)'$$
$$= A'(B'C')$$
$$= A'B'C'$$

- Generally,
  - $(A + B + C + D + \cdots F)' = A'B'C'D' \cdots F'$
  - $(ABCD \cdots F)' = A' + B' + C' + D' + \cdots + F'$

Complement of a Function

- Ex 2-2) Find the complement of the functions

$$F_1 = x'yz' + x'y'z, F_2 = x(y'z' + yz)$$

Sol.)

DeMorgan's Law

$(x + y)' = x'y'$

$(xy)' = x' + y'$

$$\begin{aligned} F_1' &= (x'yz' + x'y'z)' \\ &= (x'yz')'(x'y'z)' \\ &= (x + y' + z)(x + y + z') \end{aligned}$$

$$\begin{aligned} F_2' &= [x(y'z' + yz)]' \\ &= x' + (y'z' + yz)' \\ &= x' + (y'z')'(yz)' \\ &= x' + (y + z)(y' + z') \end{aligned}$$

Complement of a Function

- Ex 2-3) Find the complement of the functions F1 And F2 Ex 2-2 by taking their duals and complementing each literal.

1. Find its dual (AND↔OR, 1↔0)
2. Complement of each literal

$$F_1 = x'yz' + x'y'z$$

The dual of F_1 is $(x' + y + z')(x' + y' + z)$

Complement of each literal:

$$(x + y' + z)(x + y + z') = F_1'$$

$$F_2 = x(y'z' + yz)$$

The dual of F_2 is $x + (y' + z')(y + z)$

Complement of of each literal:

$$x' + (y + z)(y' + z') = F_2'$$

Minimization by Boolean Functions

Table 2-2
Truth Tables for F_1 and F_2

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1 (1)	1 (1)
0	1	0	0	0
0	1	1	0	1 (2)
1	0	0	1 (2)	1 (3)
1	0	1	1 (3)	1 (4)
1	1	0	1 (4)	0
1	1	1	1 (5)	0

$$F_1 = \underset{(1)}{x'y'z} + \underset{(2)}{xy'z'} + \underset{(3)}{xy'z} + \underset{(4)}{xyz'} + \underset{(5)}{xyz}$$

Minimization by Boolean Functions

$$
\begin{aligned}
F_1 &= \overset{(1)}{x'y'z} + \overset{(2)}{xy'z'} + \overset{(3)}{xy'z} + \overset{(4)}{xyz'} + \overset{(5)}{xyz} \\
&= y'z(x' + x) + xy'(z' + z) + xy(z' + z) \\
&= y'z + xy' + xy \\
&= x(y' + y) + y'z = \boxed{x + y'z}
\end{aligned}
$$

(1)+(3), (2)+(4), (4)+(5)

$x + x = x$

$x + x' = 1$

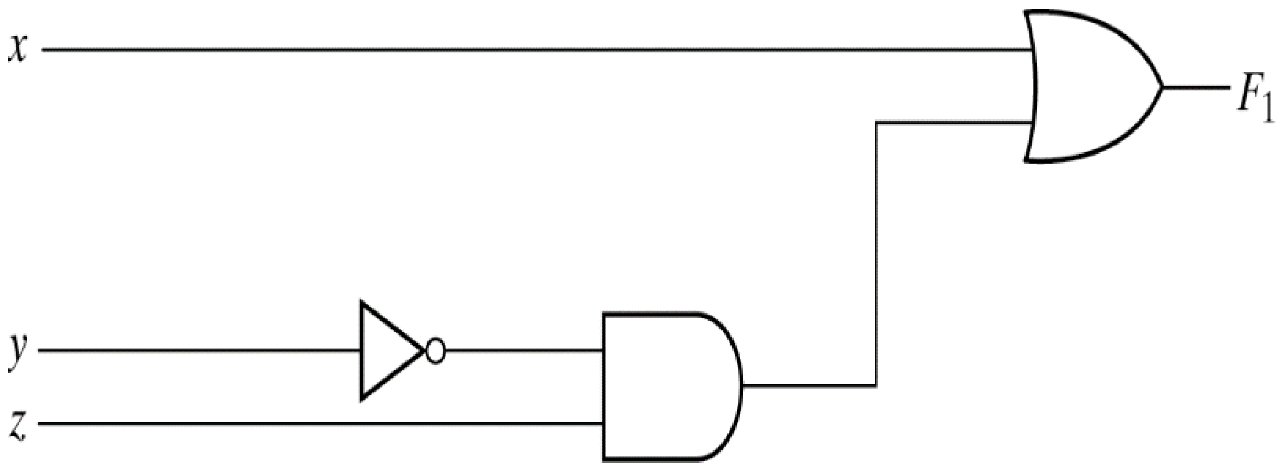
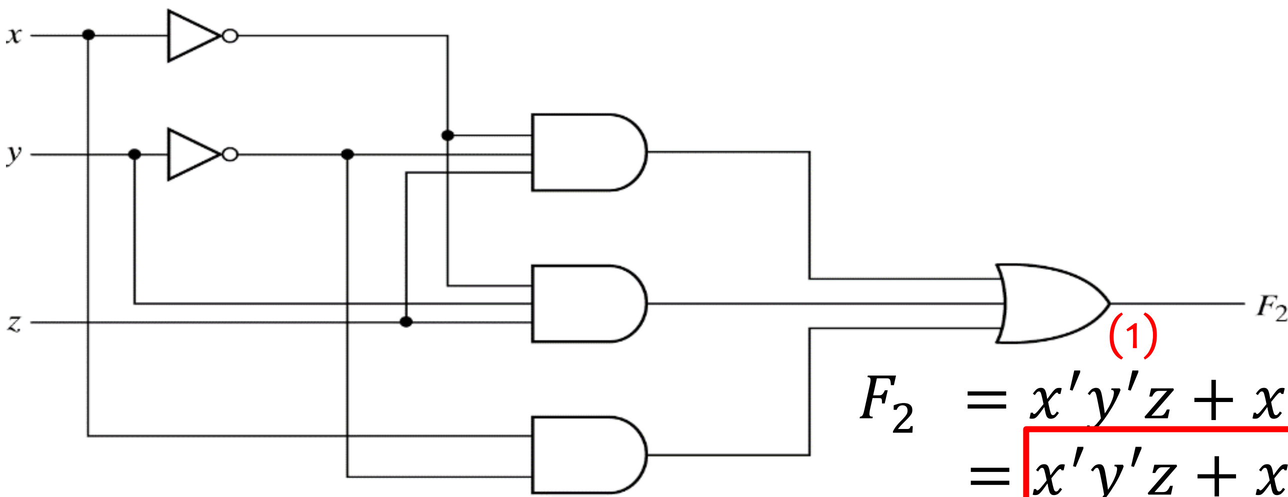


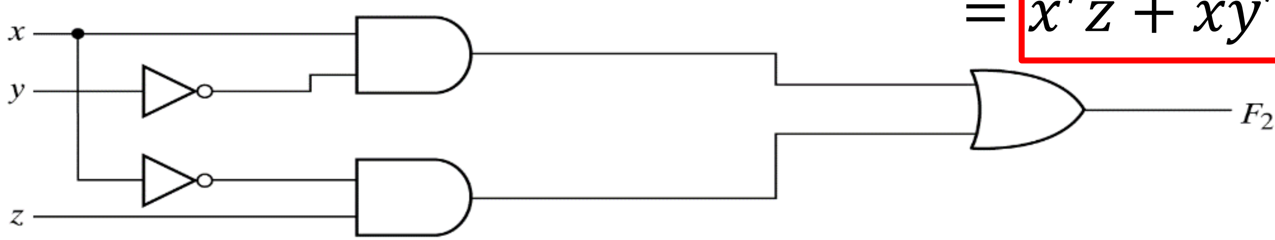
Fig. 2-1 Gate implementation of $F_1 = x + y'z$

Minimization by Boolean Functions



(a) $F_2 = x'y'z + x'yz + xy'$

$$
\begin{aligned}
F_2 &= \overset{(1)}{x'y'z} + \overset{(2)}{x'yz} + \overset{(3)}{xy'z'} + \overset{(4)}{xy'z} \\
&= \boxed{x'y'z + x'yz + xy'(z' + z)} \qquad \text{(a)} \\
&= x'z(y' + y) + xy' \\
&= \boxed{x'z + xy'} \qquad \text{(b)}
\end{aligned}
$$



(b) $F_2 = xy' + x'z$

Fig. 2-2 Implementation of Boolean function F_2 with gates

CANONICAL AND STANDARD FORMS

Canonical Forms

- Minterms and Maxterms

Table 2-3
Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical SOP/POS Forms

Table 2-4
Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Canonical sum of product (SOP)

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

Canonical product of sum (POS)

$$f_1 = (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) = M_0 M_2 M_3 M_5 M_6$$

$$f_2 = (x + y + z)(x + y + z')\,(x + y' + z)(x' + y + z)(x' + y + z) = M_0\ M_1 M_2 M_4$$

Canonical SOP Forms

Conversion from standard SOP to canonical SOP form

- Ex 2-4) Express the Boolean function F=A+B'C in a sum of minterms.

$$A = A(B + B') = AB + AB'$$
$$= AB(C + C') + AB'(C + C')$$
$$= ABC + ABC' + AB'C + AB'C'$$
$$B'C = B'C(A + A') = AB'C + A'B'C$$
$$F = A + B'C$$
$$= ABC + ABC' + \boxed{AB'C} + AB'C'$$
$$+ \boxed{AB'C} + AB'C'$$
$$= m_1 + m_4 + m_5 + m_6 + m_7$$
$$= \sum(1,4,5,6,7)$$

Table 2-5
Truth Table for $F = A + B'C$

A	B	C	F
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

Canonical POS Forms

Conversion from standard SOP to canonical POS form

- Ex 2-5) Express the Boolean function $F = xy + x'z$ in a product of maxterm form.

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

$let\ A = xy$

$$F = xy + x'z = (xy + x')(xy + z)$$
$$= (x' + x)(y + x')(x + z)(y + z)$$
$$= (x' + y)(x + z)(y + z)$$

$xy + x' = x' + xy = (x' + x)(x' + y)$

$x + x' = 1$

$xx' = 0, x + 0 = x \qquad let\ A = x' + y$

$$x' + y = x' + y + zz' = (x' + y + z)(x' + y + z') = M_4M_5$$
$$x + z = x + z + yy' = (x + y + z)(x + y' + z) = M_0M_2$$
$$y + z = y + z + xx' = (x + y + z)(x' + y + z) = M_0M_4$$

$$F = M_0M_2M_4M_5 = \prod(0, 2, 4, 5)$$

Conversion between Canonical Forms

- Relation between SOP and POS

$$\boxed{m_i = M_i' \text{ and } m_i' = M_i}$$

$$F(x, y, z) = \sum(1, 3, 6, 7)$$

$$F'(x, y, z) = \sum(0, 2, 4, 5)$$

$$= m_0 + m_2 + m_4 + m_5$$

$$F = (F')' = (m_0 + m_2 + m_4 + m_5)'$$

$$= m_0' m_2' m_4' m_5' = M_0 M_2 M_4 M_5$$

Ex) $F = xy + xz'$

$$= \sum(1,3,6,7)$$

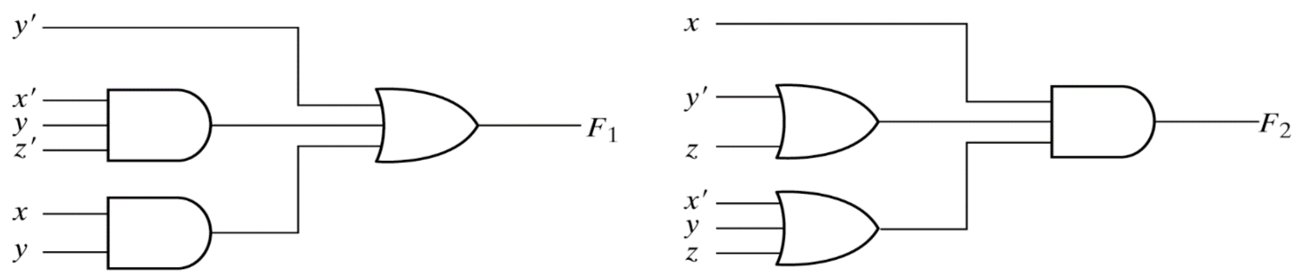
$$= \prod(0,2,4,5)$$

Table 2-6
Truth Table for $F = xy + x'z$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Standard SOP/POS Forms

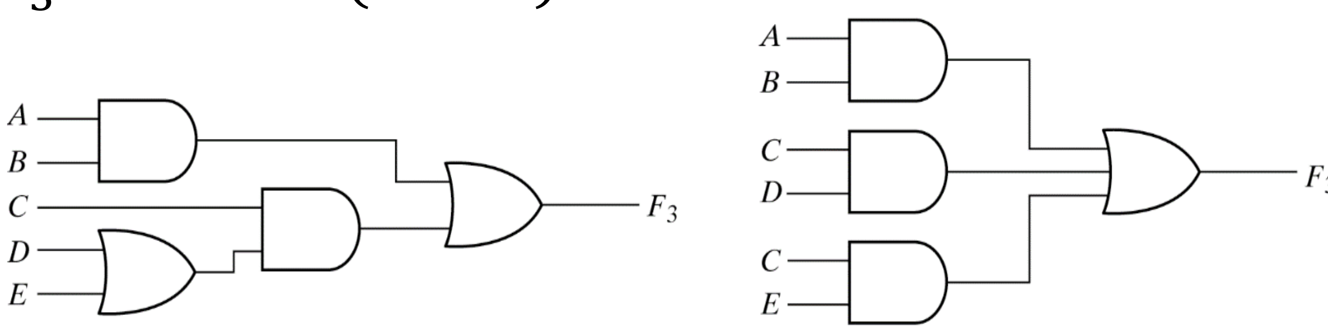
- SOP (Sum of product) : $F_1(x, y, z) = y' + xy + x'yz'$
- POS (Product of sum) : $F_2(x, y, z) = x(y' + z)(x' + y + z)$



(a) Sum of Products (b) Product of Sums

Fig. 2-3 Two-level implementation

- Ex) $F_3 = AB + C(D + E) = AB + CD + CE$



(a) $AB + C(D + E)$ (b) $AB + CD + CE$

Fig. 2-4 Three- and Two-Level implementation

SYNTHESIS USING AND, OR AND NOT GATES

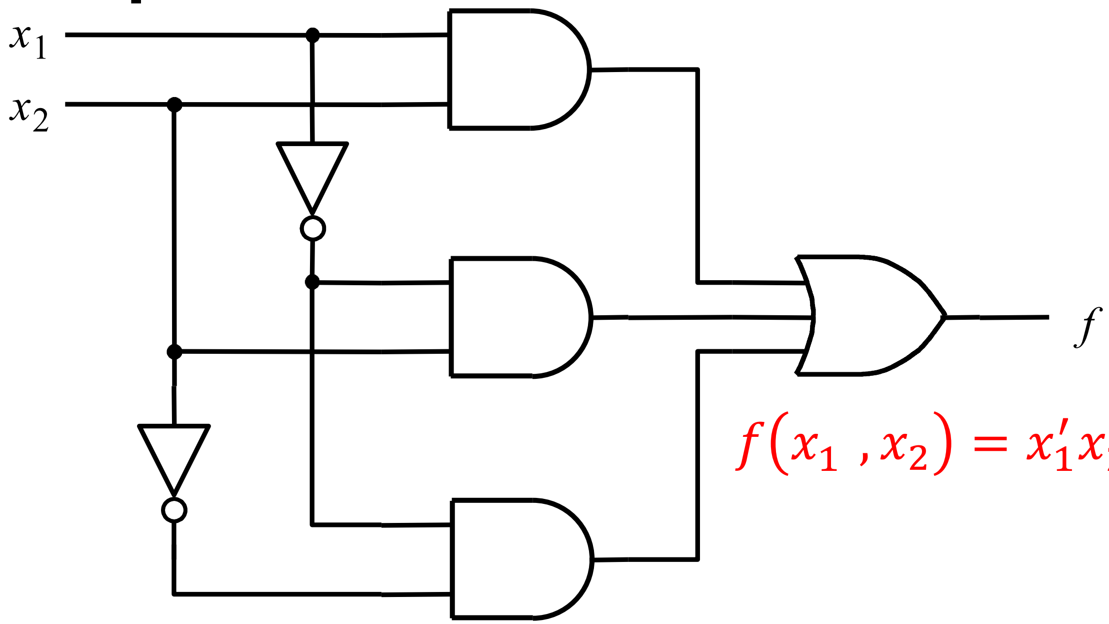
Synthesis Using AND, OR and NOT Gates

x_1	x_2	$f(x_1, x_2)$	
0	0	1	$m_0 = x_1' x_2'$
0	1	1	$m_1 = x_1' x_2$
1	0	0	$M_2 = x_1 + x_2'$
1	1	1	$m_3 = x_1 x_2$

Figure 2.15. A function to be synthesized.

$$
\begin{aligned}
f(x_1, x_2) &= m_0 + m_1 + m_3 \\
&= x_1' x_2' + x_1' x_2 + x_1 x_2 \\
&= x_1'(x_2' + x_2) + x_2(x_1' + x_1) \\
&= x_1' + x_2 \\
&= M_2
\end{aligned}
$$

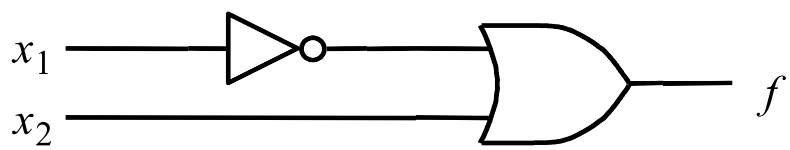
Two implementations of a function



$$f(x_1, x_2) = x_1' x_2' + x_1' x_2 + x_1 x_2$$

(a) Canonical sum-of-products

$$f(x_1, x_2) = x_1' + x_2 = M_2$$



(b) Minimal-cost realization

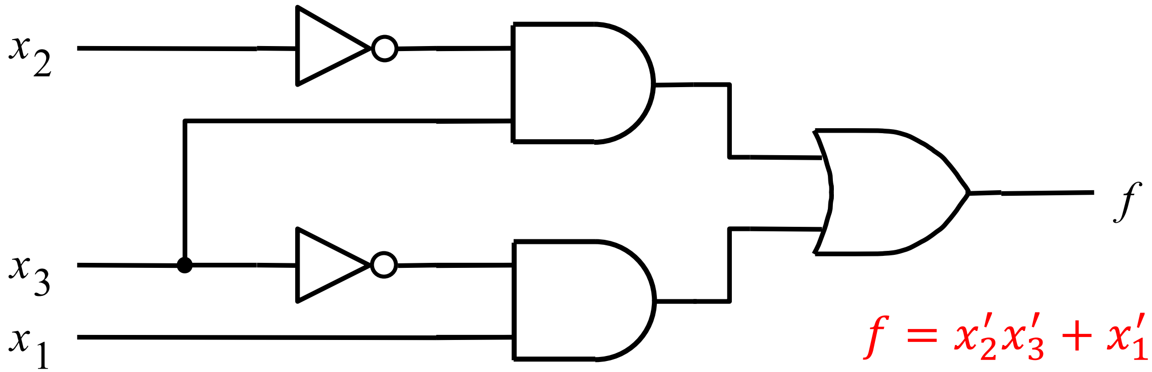
Three Variable Function

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$
\begin{aligned}
f &= x_1'x_2'x_3 + x_1 x_2'x_3' + x_1 x_2'x_3 + x_1 x_2 x_3 \\
&= x_2'x_3 (x_1' + x_1') + x_1 x_3 \left(x_2' + x_2\right) \\
&= x_2'x_3 + x_1 x_3
\end{aligned}
$$

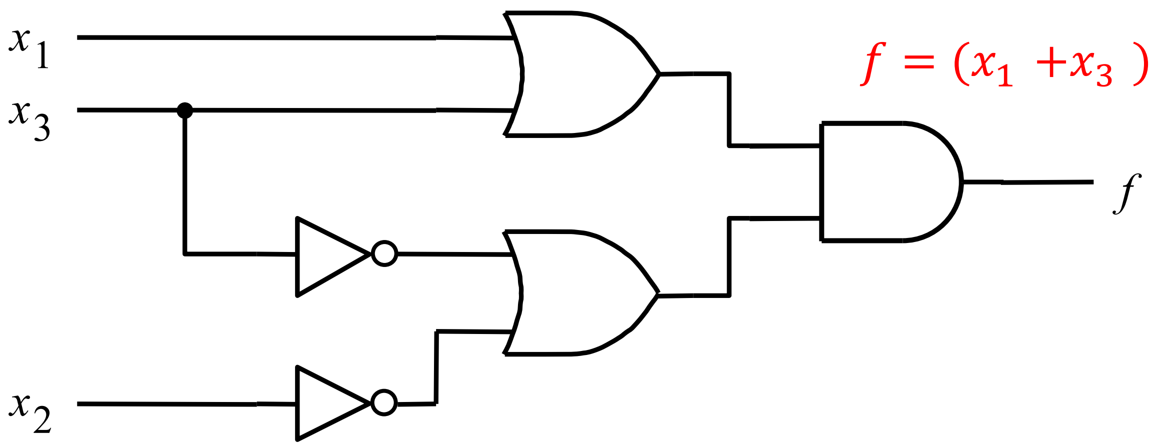
$$
\begin{aligned}
f &= (x_1 + x_2 + x_3)(x_1 + x_2' + x_3) \\
&\quad (x_1 + x_2' + x_3')(x_1' + x_2' + x_3') \\
&= \{(x_1 + x_3) + x_2 x_2'\} \\
&\quad \{(x_2' + x_3') + x_1 x_1'\} \\
&= (x_1 + x_3)(x_2' + x_3')
\end{aligned}
$$

Two realizations of a function



$f = x_2'x_3' + x_1'x_3'$

(a) A minimal sum-of-products realization

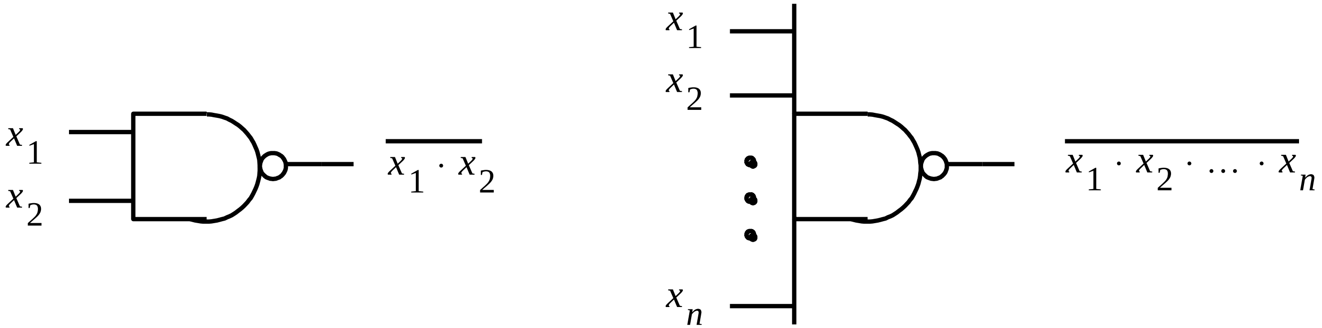


$f = (x_1 + x_3)(x_2' + x_3')$

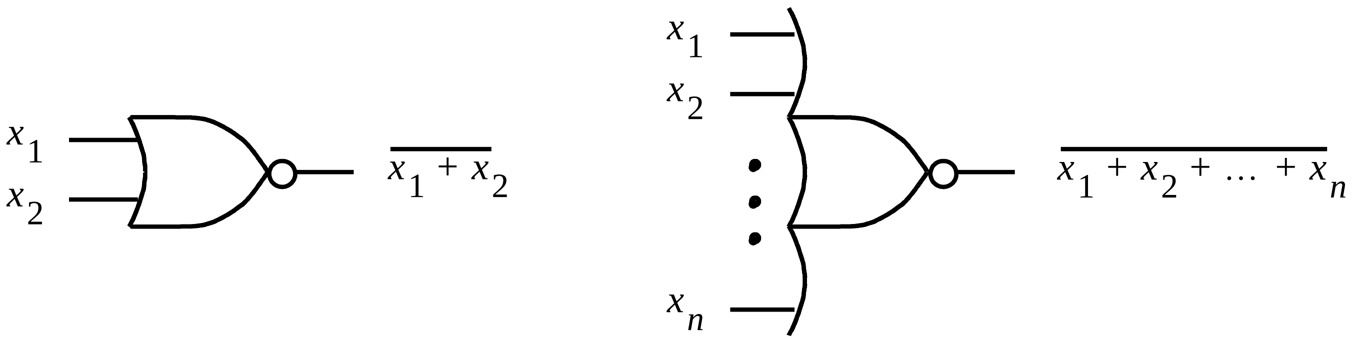
(b) A minimal product-of-sums realization

NAND AND NOR LOGIC NETWORKS

NAND and NOR Logic Networks

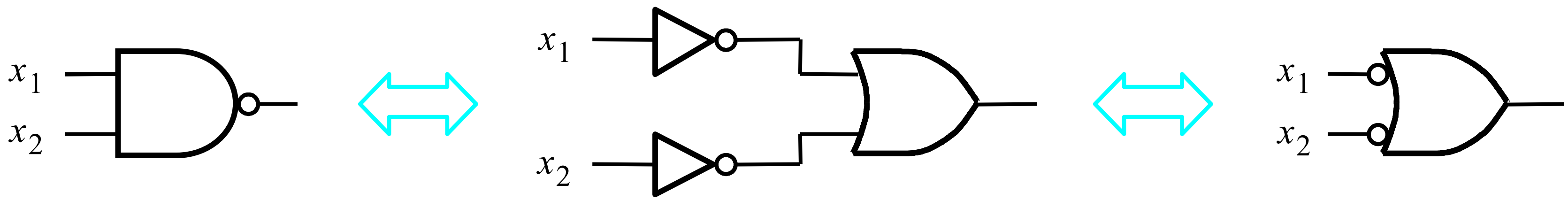


(a) NAND gates



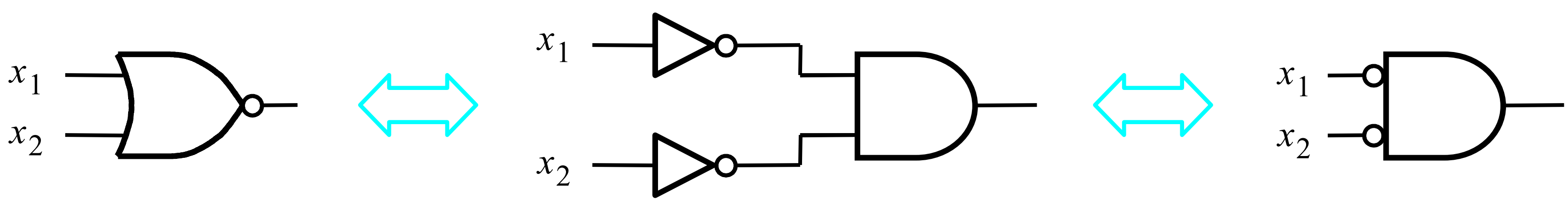
(b) NOR gates

DeMorgan's theorem



NAND

$$(x_1 x_2)' = x_1' + x_2'$$

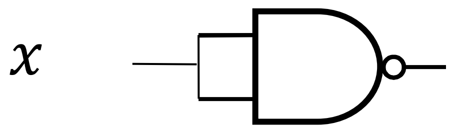


NOR

$$(x_1 + x_2)' = x_1' x_2'$$

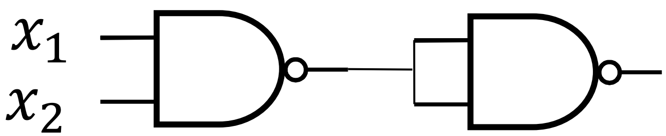
Basic Gates Using NAND

NOT



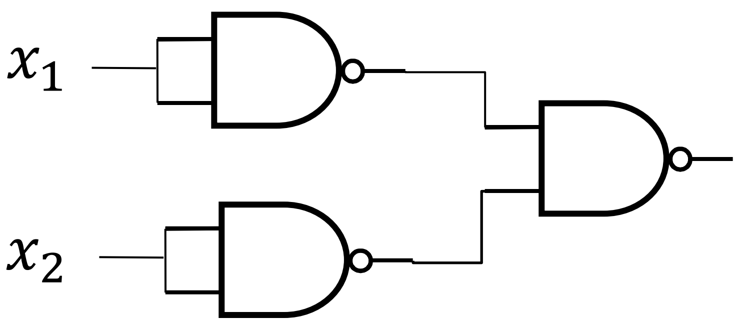
$$f = (xx)' = x'$$

AND



$$f = \{(x_1 \; x_2)'\}' = x_1 x_2$$

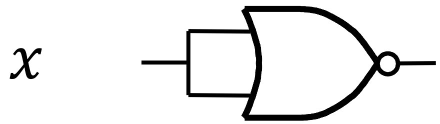
OR



$$f = (x_1' x_2')' = x_1 + x_2$$

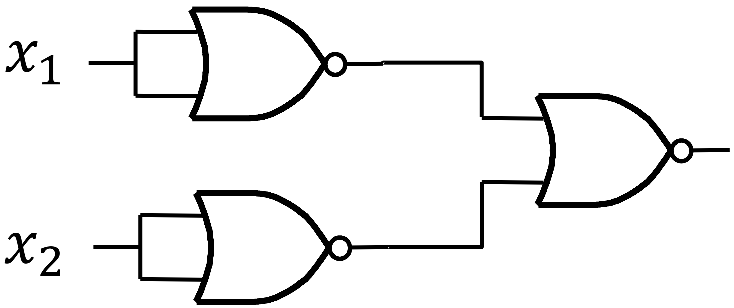
Basic Gates Using NOR

NOT



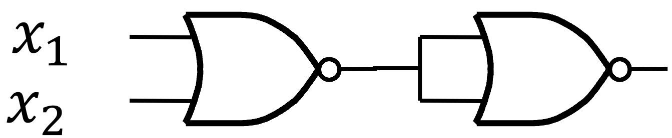
$f = (xx)' = x'$

AND



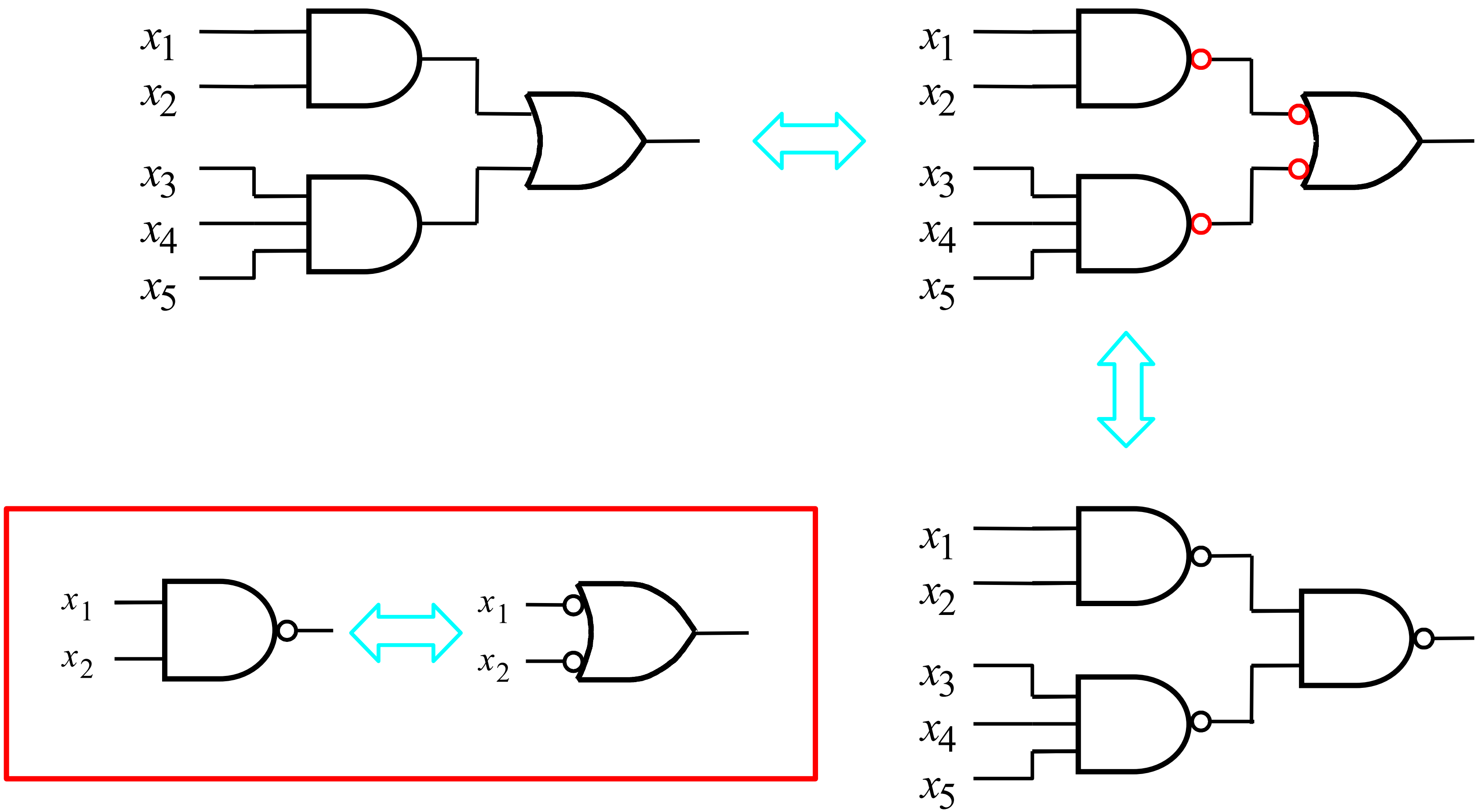
$f = (x_1' + x_2')' = x_1 x_2$

OR

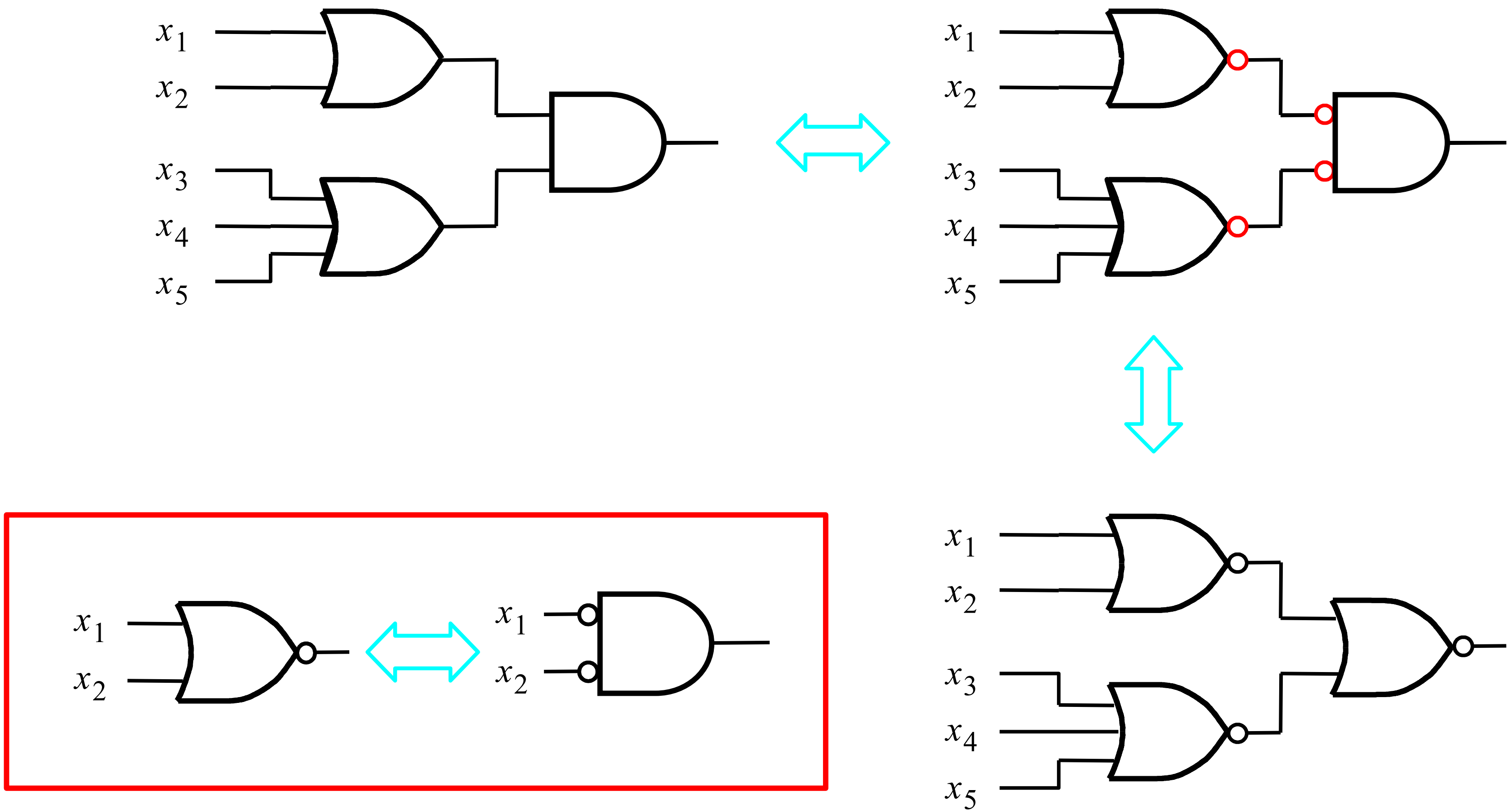


$f = \left\{ \left(x_1 + x_2 \right)' \right\}' = x_1 + x_2$

Using NAND gates to implement a sum-of-products

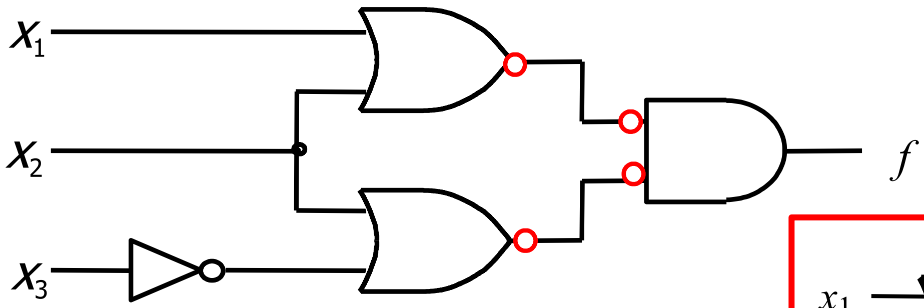


Using NOR gates to implement a product-of sums

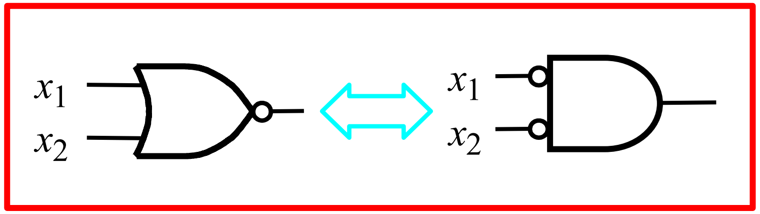


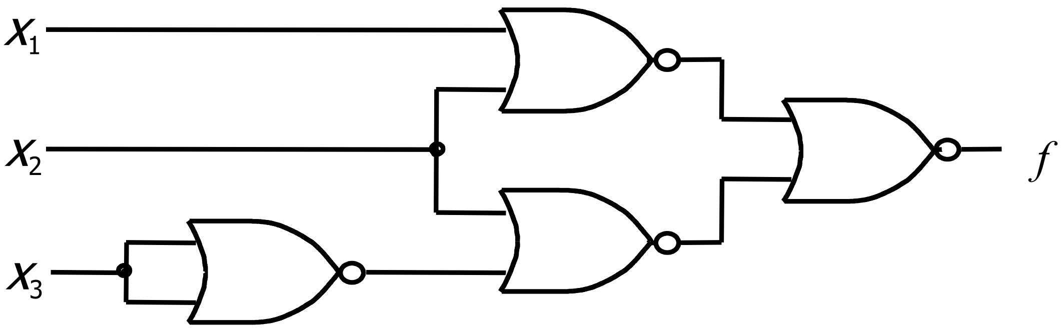
NOR-gate realization of the function

$$f = (x_1 + x_2)(x_2 + x_3')$$



(a) POS implementation

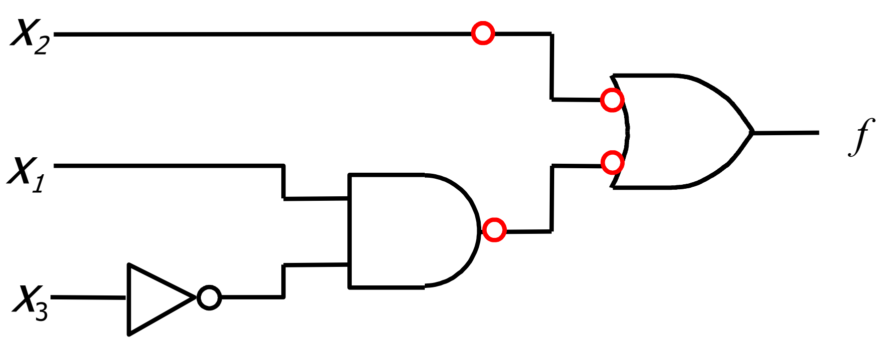




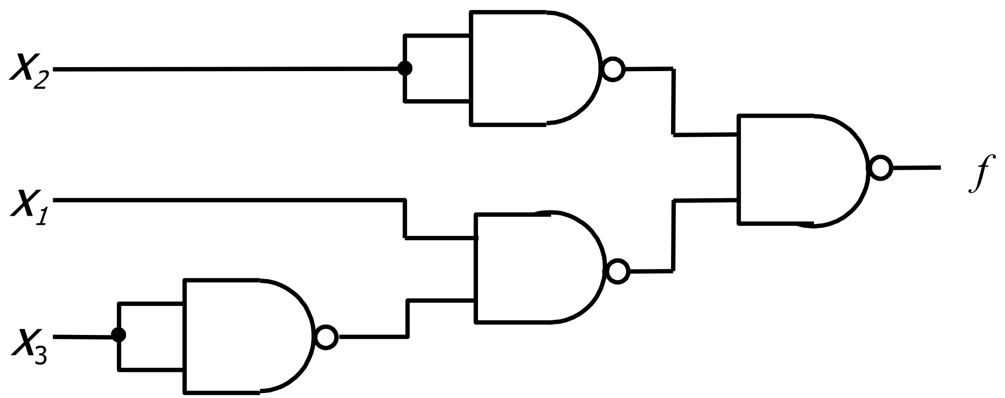
(b) NOR implementation

NAND-gate realization of the function

$f = x_2 + x_1 x_3'$



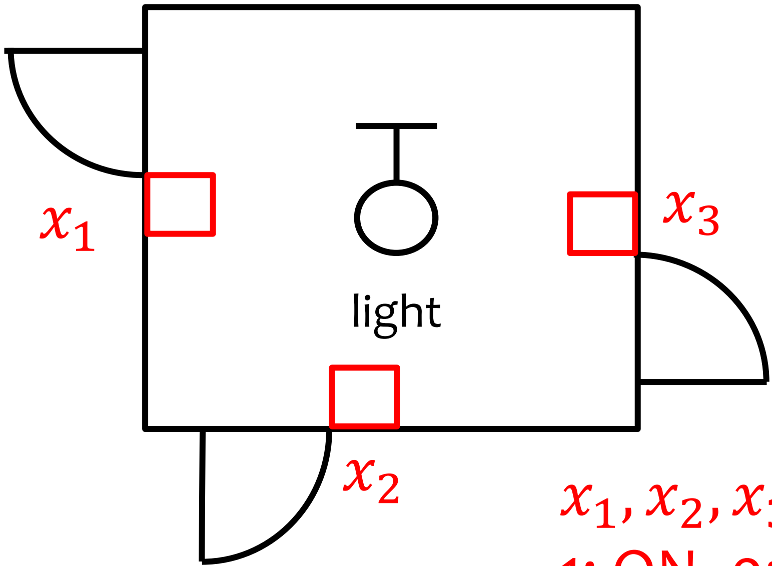
(a) SOP implementation



(b) NAND implementation

DESIGN EXAMPLES

Three-way Light Control

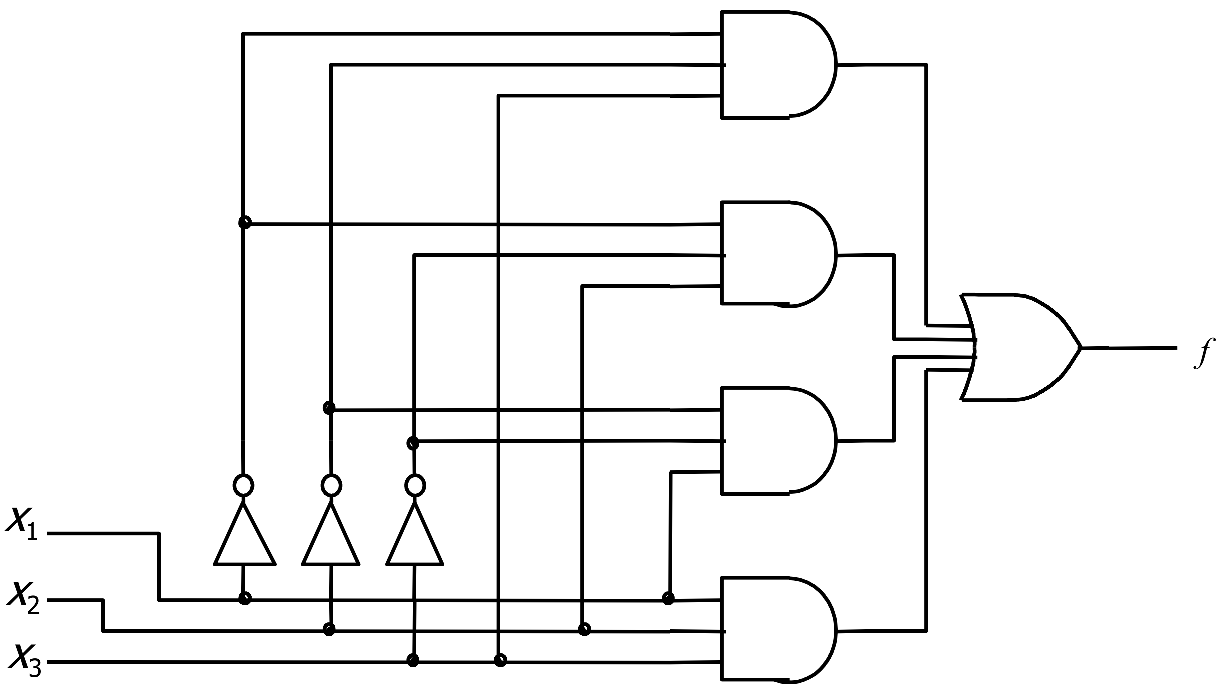


x_1, x_2, x_3: Switch
1: ON, 0: OFF

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

SOP Implementation

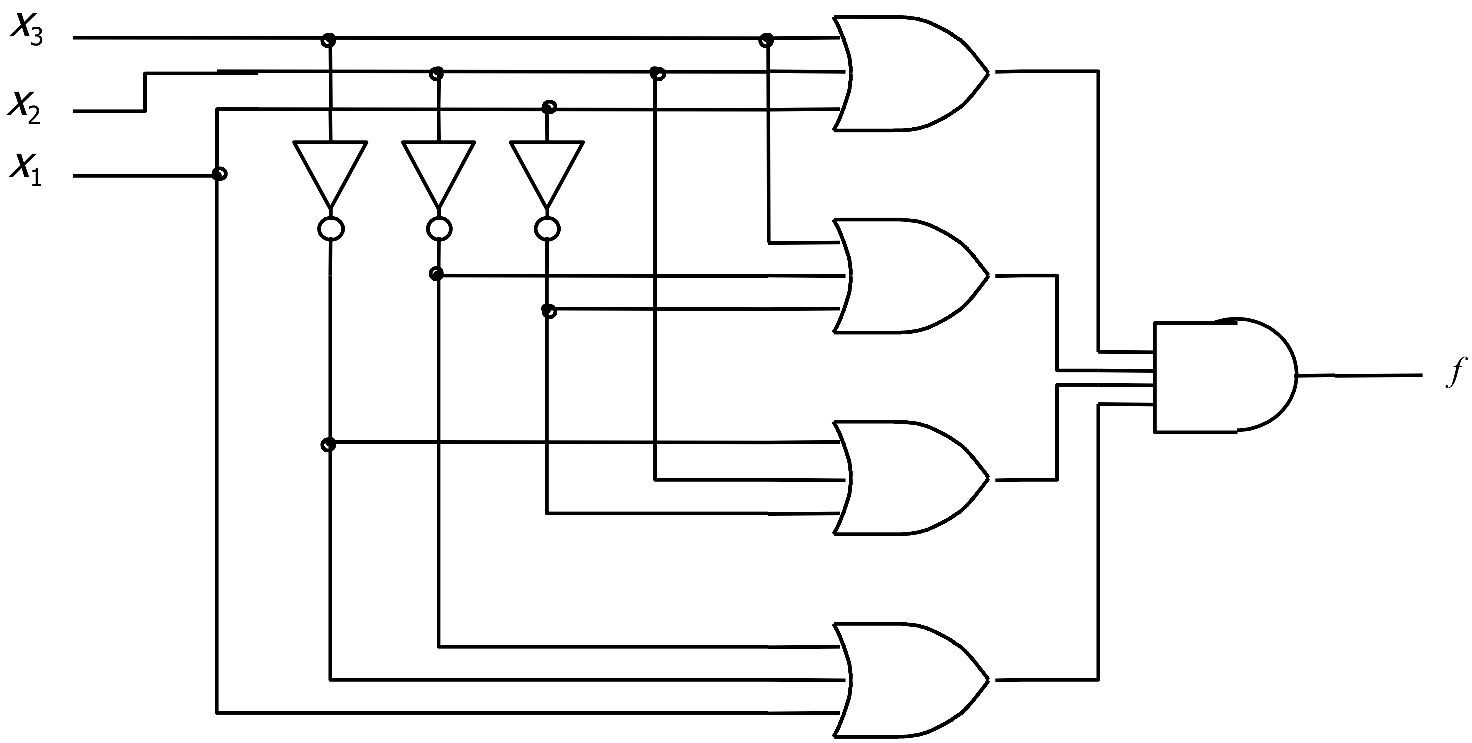
$$f = x_1' x_2' x_3' + x_1' x_2' x_3' + x_1' x_2' x_3' + x_1' x_2' x_3'$$



(a) Sum-of-products realization

POS Implementation

$$f = (x_1 + x_2 + x_3)(x_1 + x_2' + x_3')(x_1' + x_2 + x_3')(x_1' + x_2' + x_3)$$



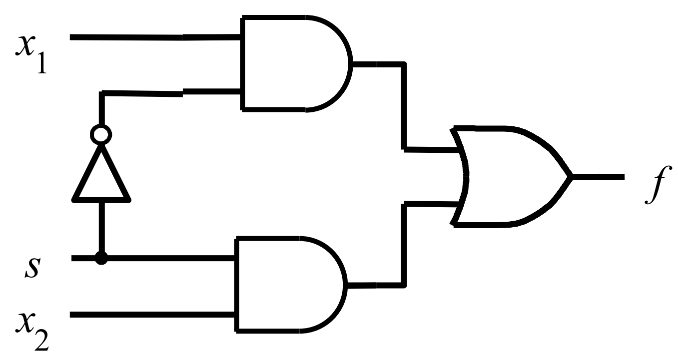
(b) Product-of-sums realization

Multiplexer Circuit

s	x_1	x_2	$f(s, x_1, x_2)$	
0	0	0	0	
0	0	1	0	
0	1	0	1	(1)
0	1	1	1	(2)
1	0	0	0	
1	0	1	1	(3)
1	1	0	0	
1	1	1	1	(4)

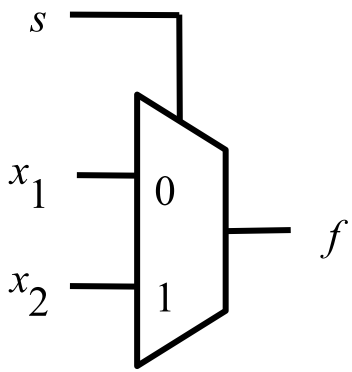
(a) Truth Table

$$\begin{aligned} f &= \overset{(1)}{s'x_1 x_2'} + \overset{(2)}{s'x_1 x_2} + \overset{(3)}{sx_1'x_2} + \overset{(4)}{sx_1 x_2} \\ &= s'x_1\left(x_2' + x_2\right) + sx_2\left(x_1' + x_1\right) \\ &= s'x_1 + sx_2 \end{aligned}$$



(b) Circuit

Implementation of a multiplexer



(c) Graphical symbol

$$f = s'x_1 + sx_2$$

$$f = \begin{cases} x_1 & s = 0 \\ x_2 & s = 1 \end{cases}$$

s	$f(s, x_1, x_2)$
0	x_1
1	x_2

(d) More compact truth-table representation

Summary

- Basic gates including AND, OR, NOT are investigated in terms of the truth table, Boolean representation and the timing diagram.
- Several properties of Boolean algebra are found different from ordinary algebra.
- All logic function can be represented by canonical SOP/POS forms based on the truth table.
- Canonical forms can be converted to the minimized standard form by Boolean algebra.
- Instead of AND, OR and NOT gates, any logic functions can be implemented by only NOR or NAND