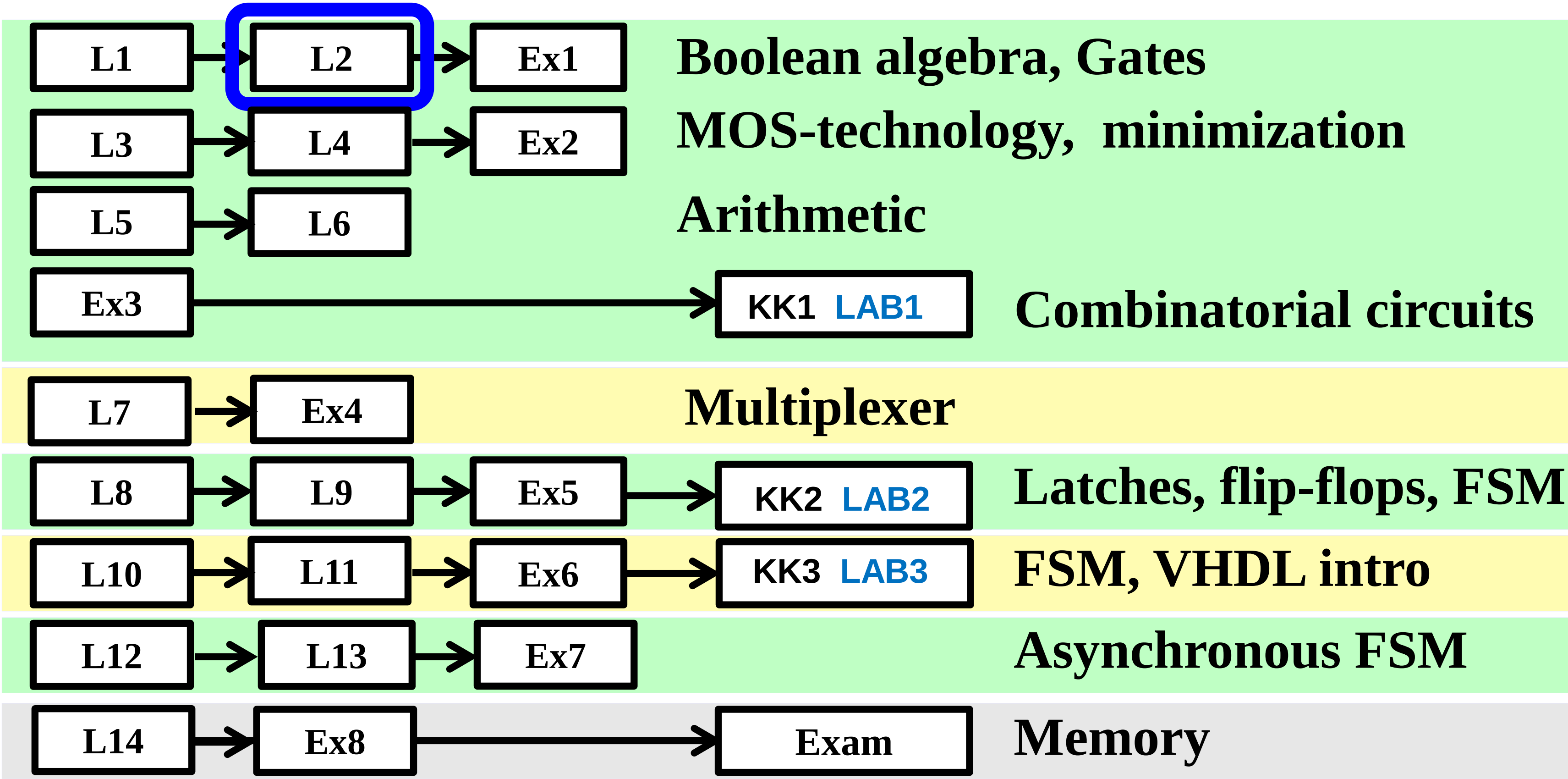
IE1204 Digital Design

L2: Logic gates and circuits, Boolean Algebra

KTH Informations- och kommunikationsteknik

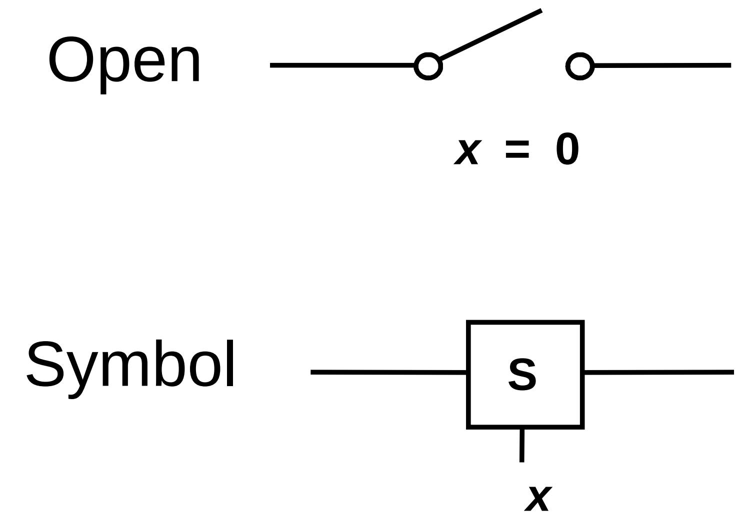
Masoumeh (Azin) Ebrahimi
KTH/ICT
mebr@kth.se

IE1204 Digital Design



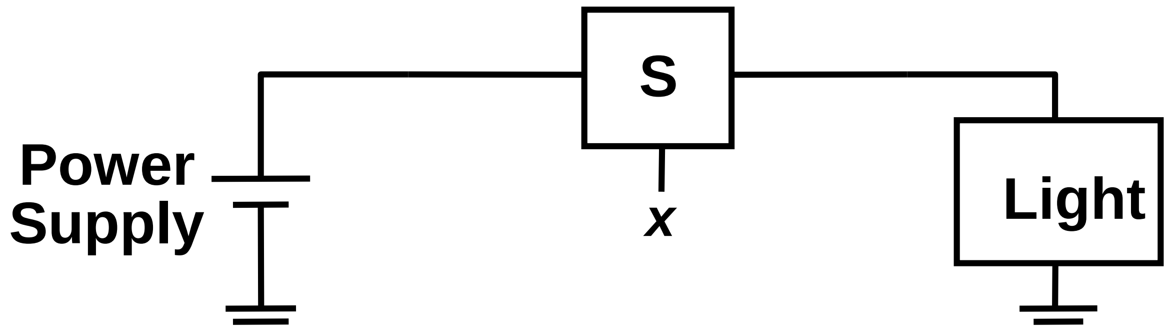
Switch

- A switch has two positions
  - Closed/On
  - Open/Off



Implementation of logic functions

- A switch can be used to implement logic functions



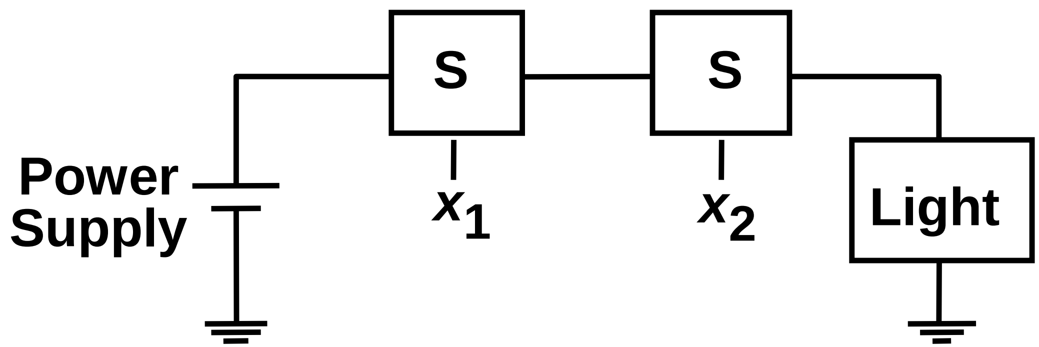
$$L(x) = \begin{cases} 0 & \text{Light Off} \\ 1 & \text{Light On} \end{cases}$$

L(*x*) is a logic function

x is a logic variable

Operation AND

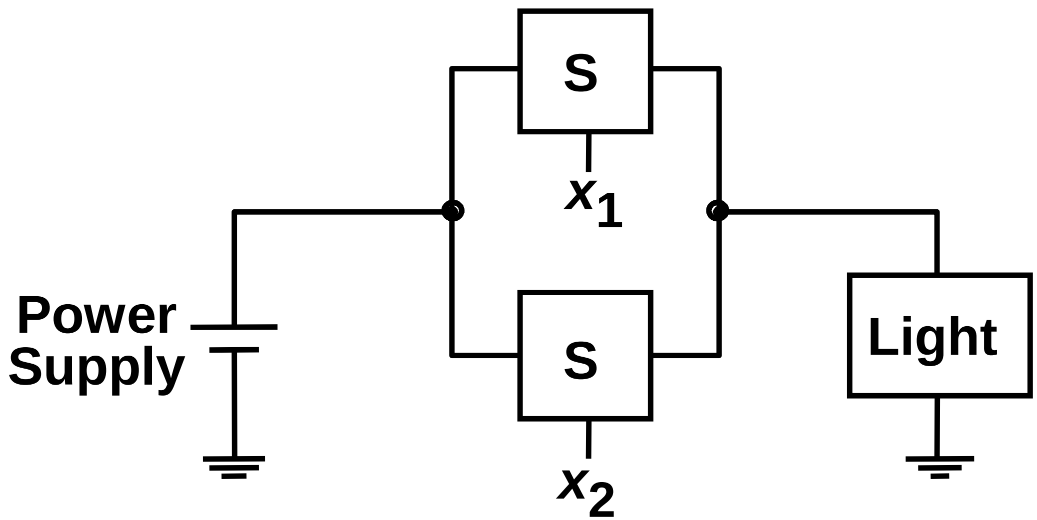
- The AND operation (·) is achieved by switches that are connected in series



$$L(x) = x_1 \cdot x_2$$

Operation OR

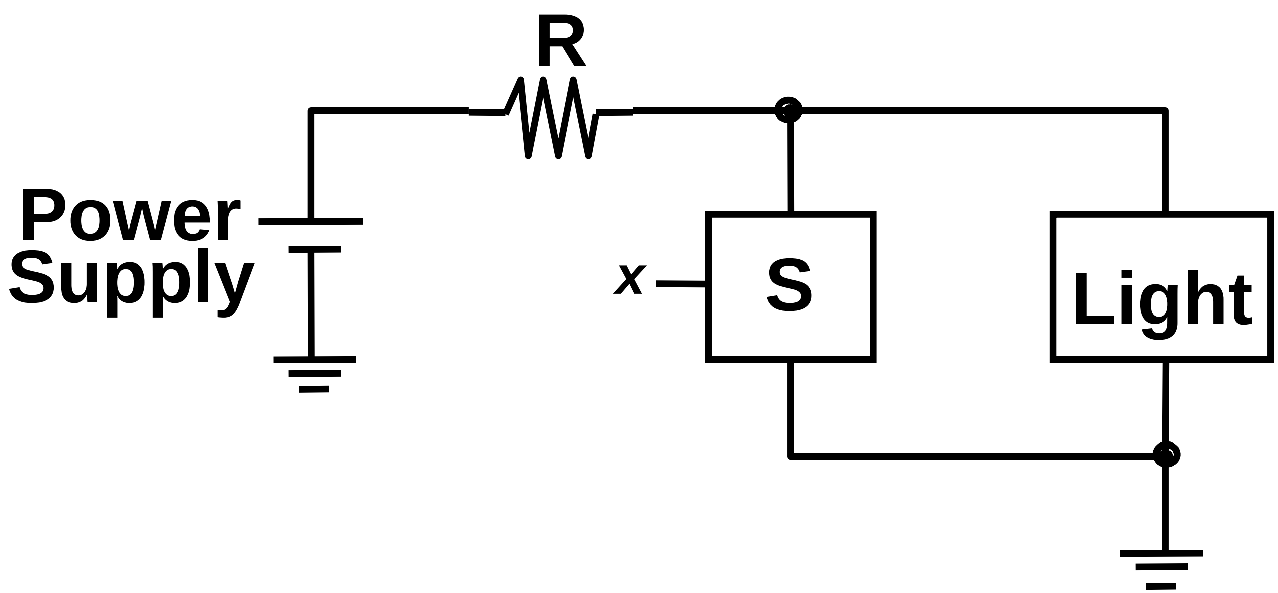
- The OR operation (+) is achieved by switches connected in parallel



$$L(x) = x_1 + x_2$$

Operation NOT

- NOT function inverts the logic value



$$L(x) = \overline{x}$$

Truth Table

- A logic function can also be described by a *truth table*

x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1
		AND	OR

1 stands for true

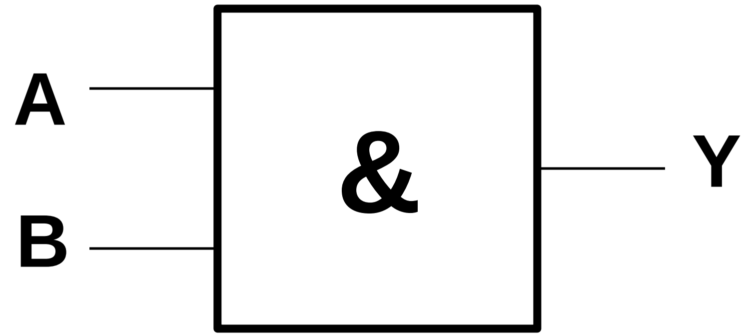
0 stands for false

Logic gates
AND gate

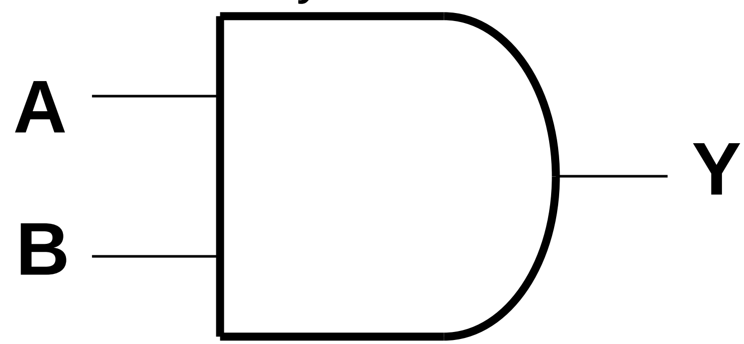
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Y = A \cdot B$$

IEC symbol
(International Electrotechnical Commission)



Traditional (American)
Symbol

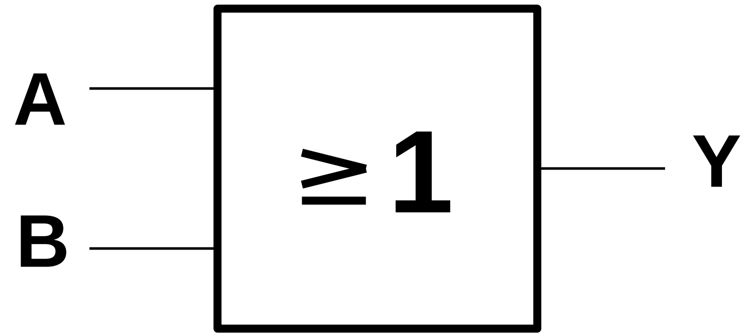


Logic gates
OR gate

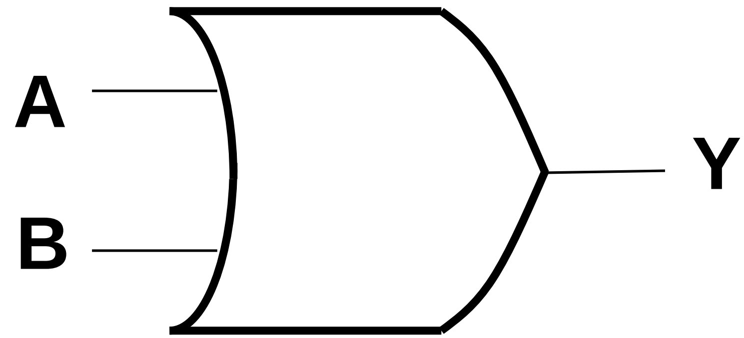
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$$Y = A + B$$

IEC symbol
(International Electrotechnical Commission)



Traditional (American)
Symbol

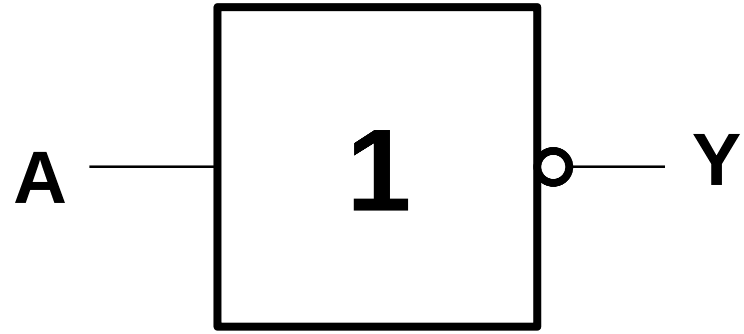


Logic gates
Inverter (NOT gate)

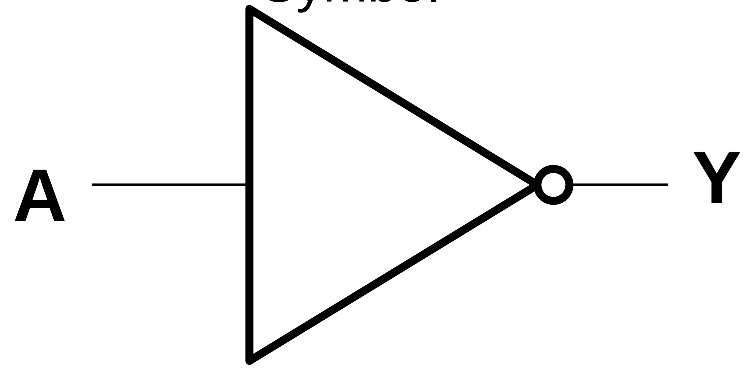
A	Y
0	1
1	0

$$Y = \overline{A}$$

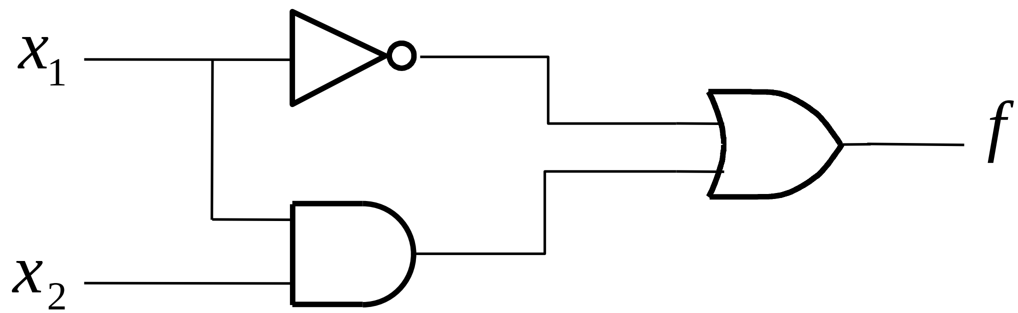
IEC symbol
(International Electrotechnical Commission)



Traditional (American)
Symbol

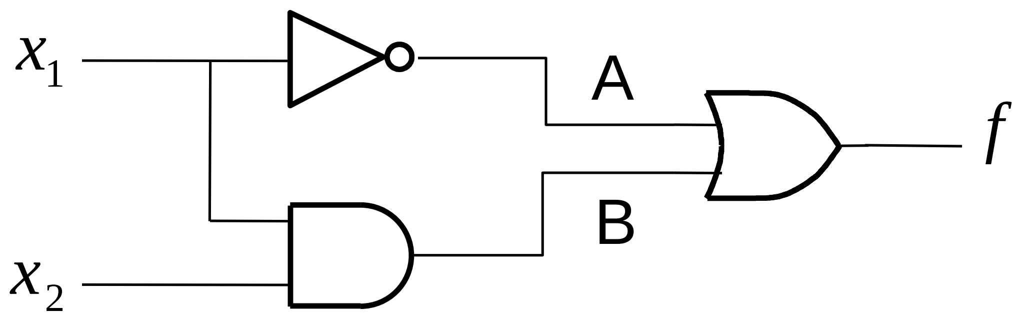


Which function is implemented by this logic circuit?



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

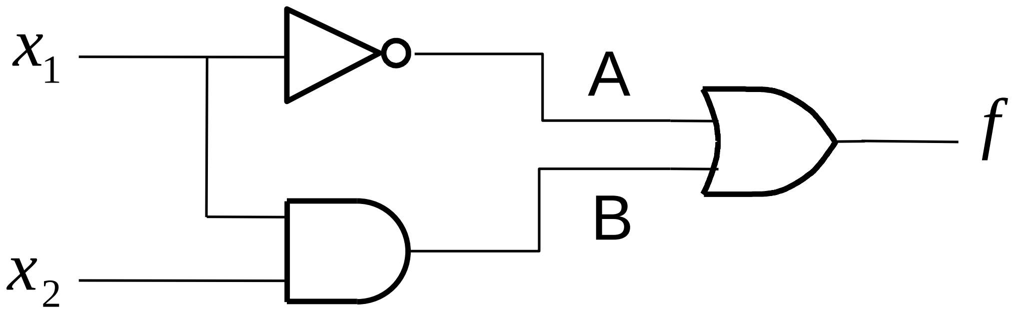
Truth Table

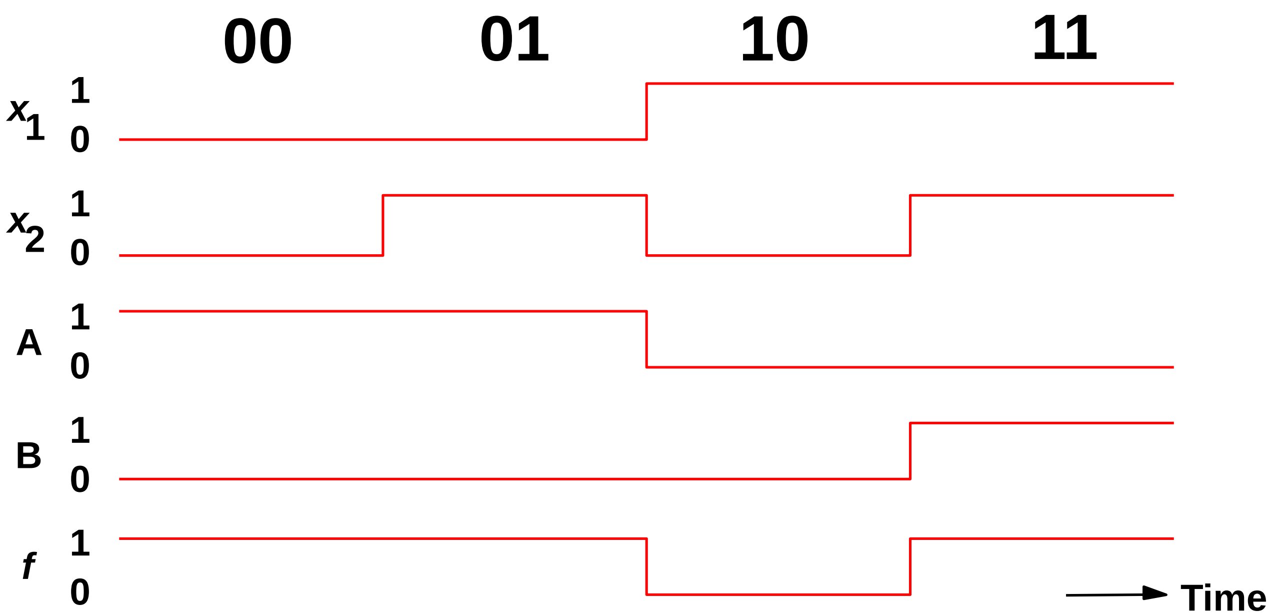


x_1	x_2	$f(x_1,x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

A	B
1	0
1	0
0	0
0	1

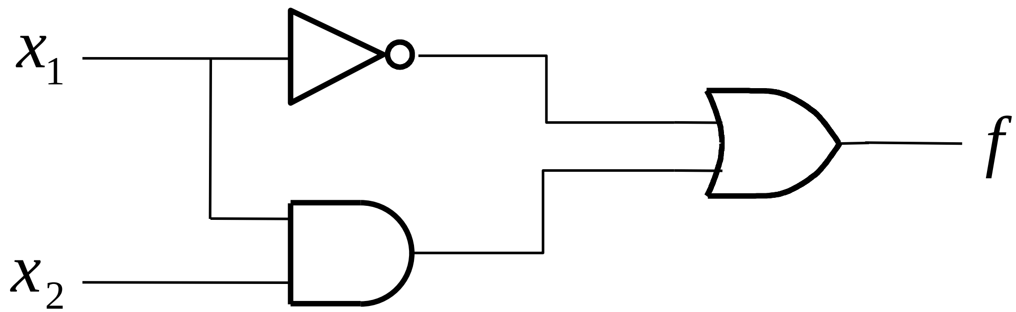
Timing Chart





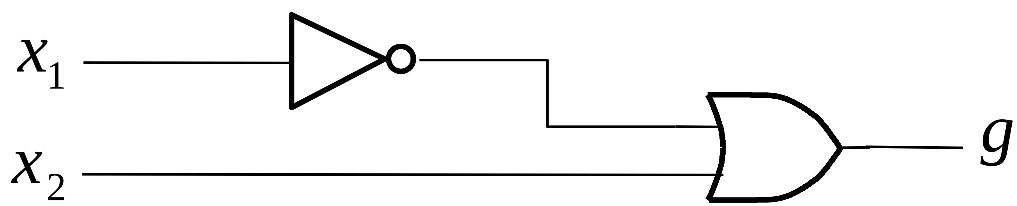
Different logic circuits can implement the same function

a)



$$f = \overline{x}_1 + x_1 \cdot x_2$$

b)



$$g = \overline{x}_1 + x_2$$

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

Boolean algebra

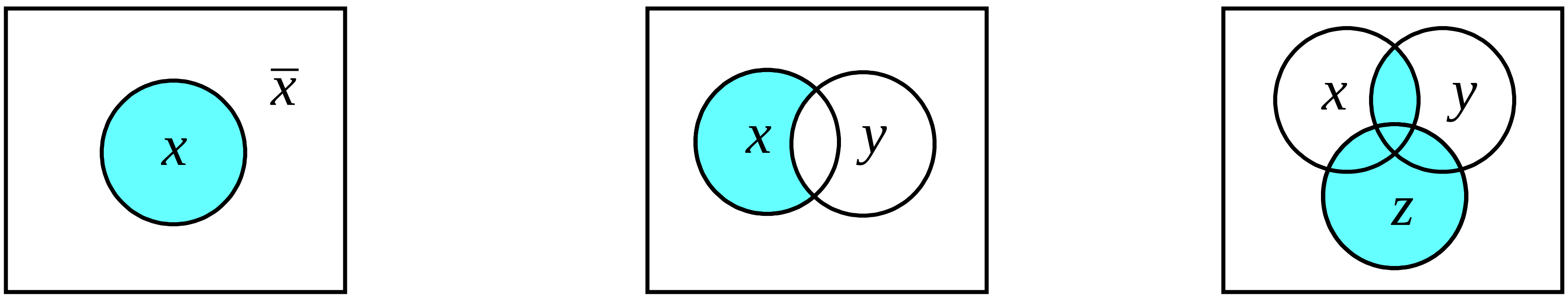
- Since *several logic circuits* can implement the same function, we want to find the *most cost-effective implementation*
- Logic circuits can be very large
- A *mathematical base* is needed so that the optimization of logic circuits can be *performed using computers*

Axioms of Boolean algebra

Axiomer			
(1a)	$0 \cdot 0 = 0$	(1b)	$1 + 1 = 1$
(2a)	$1 \cdot 1 = 1$	(2b)	$0 + 0 = 0$
(3a)	$0 \cdot 1 = 1 \cdot 0 = 0$	(3b)	$1 + 0 = 0 + 1 = 1$
(4a)	If $x = 0$, then $\overline{x} = 1$	(4b)	If $x = 1$, then $\overline{x} = 0$

Venn Diagrams

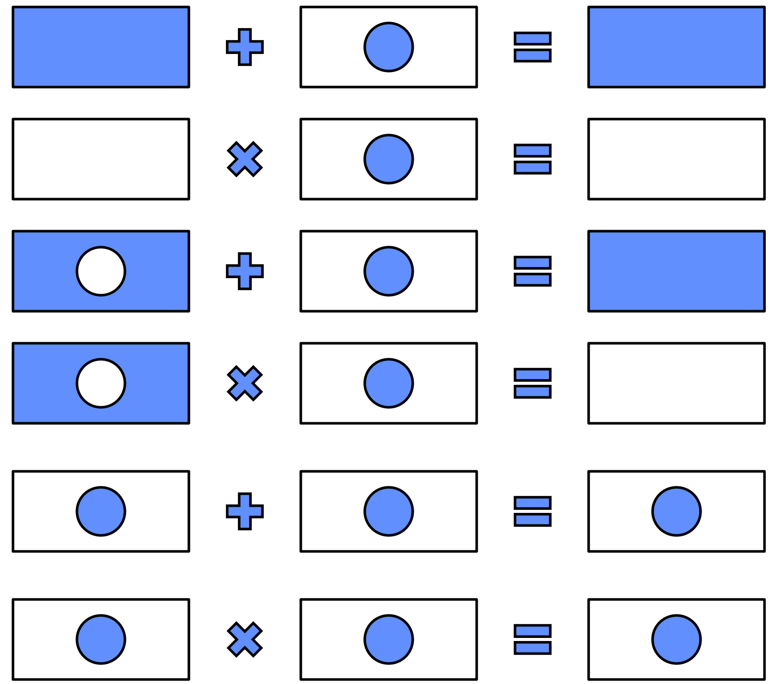
- Venn diagrams can be used to illustrate the logic operations



x $\qquad$ $x \cdot \bar{y}$ $\qquad$ $x \cdot y + z$

Boolean algebra

- $1 + A = 1$
- $0 \cdot A = 0$
- $\overline{A} + A = 1$
- $\overline{A} \cdot A = 0$
- $A + A = A$
- $A \cdot A = A$



Other properties of Boolean algebra with single variable

- Using the axioms, we can derive new properties

Räknelagar			
(5a)	$x \cdot 0 = 0$	(5b)	$\mathrm{x} + 1 = 1$
(6a)	$x \cdot 1 = x$	(6b)	$\mathrm{x} + 0 = \mathrm{x}$
(7a)	$x \cdot x = x$	(7b)	$\mathrm{x} + \mathrm{x} = \mathrm{x}$
(8a)	$x \cdot \overline{x} = 0$	(8b)	$x + \overline{x} = 1$
(9a)	$\overline{\overline{x}} = x$		

Duality principle

- If we have a valid Boolean expression, we can get another valid expression by changing
  - All 0s to 1s and all 1s to 0s
  - All ANDs to ORs and all ORs to ANDs

x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	$x_1 + x_2$
1	1	1
1	0	1
0	1	1
0	0	0

Boolean algebra properties with multiple variables

Räknelagar				
(10a)	$x \cdot y = y \cdot x$	(10b)	$x + y = y + x$	*kommutativ*
(11a)	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	(11b)	$x + (y + z) = (x + y) + z$	*associativ*
(12a)	$x \cdot (y + z) = x \cdot y + x \cdot z$	(12b)	$x + y \cdot z = (x + y) \cdot (x + z)$	*distributiv*
(13a)	$x + x \cdot y = x$	(13b)	$x \cdot (x + y) = x$	*absorption*
(14a)	$x \cdot y + x \cdot \overline{y} = x$	(14b)	$(x + y) \cdot (x + \overline{y}) = x$	
(15a)	$\overline{x \cdot y} = \overline{x} + \overline{y}$	(15b)	$\overline{x + y} = \overline{x} \cdot \overline{y}$	*DeMorgan*
(16a)	$x + \overline{x} \cdot y = x + y$	(16b)	$x \cdot (\overline{x} + y) = x \cdot y$	
(17a)	$x \cdot y + y \cdot z + \overline{x} \cdot z$ $= x \cdot y + \overline{x} \cdot z$	(17b)	$(x + y) \cdot (y + z) \cdot (\overline{x} + z)$ $= (x + y) \cdot (\overline{x} + z)$	*consensus*

Proof that consensus property holds

17a. $x \cdot y + y \cdot z + \bar{x} \cdot z = x \cdot y + \bar{x} \cdot z$

$$x \cdot y + y \cdot z + \bar{x} \cdot z \quad \text{(left side)}$$

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

$$= \quad x \cdot y \cdot z + x \cdot y \cdot \bar{z} + x \cdot y \cdot z + \bar{x} \cdot y \cdot z + \bar{x} \cdot y \cdot z + \bar{x} \cdot \bar{y} \cdot z$$

$$= \quad x \cdot y \cdot z + x \cdot y \cdot \bar{z} + \bar{x} \cdot y \cdot z + \bar{x} \cdot \bar{y} \cdot z$$

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

$$= \quad x \cdot y + \bar{x} \cdot z \quad (= \text{right side})$$

Notation options

- Different books use different notations

$\overline{x}$	x', $!x$, $/x$, $\neg x$
$x \cdot y$	xy, $x \wedge y$
$x + y$	$x \vee y$

Analysis and synthesis

- Synthesis
  - Construction of a logic circuit that implements a given logic function
- Analysis
  - The derivation of the logic function for a given logic circuit

How can the following truth table be implemented by a logic circuit?

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

How can the following truth table be implemented by a logic circuit?

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

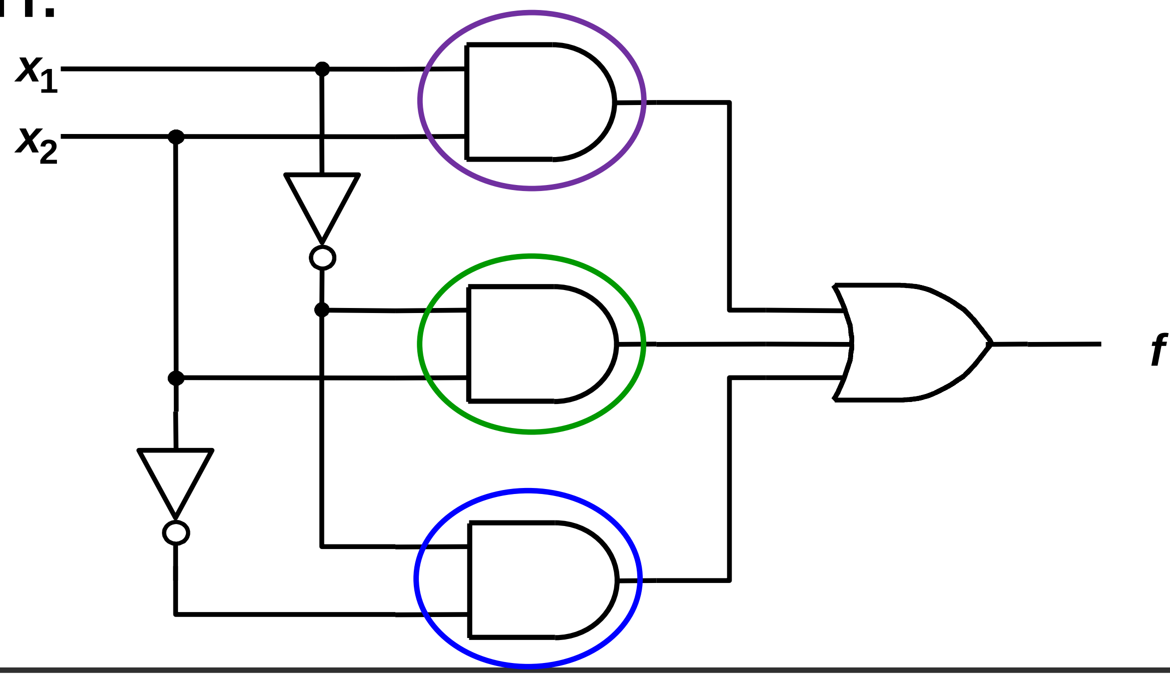
1. Write down the logic function:

$$f = \overline{x}_1\overline{x}_2 + \overline{x}_1 x_2 + x_1 x_2$$

How can the following truth table be implemented by a logic circuit?

$$f = \overline{x}_1\overline{x}_2 + \overline{x}_1 x_2 + x_1 x_2$$

2. Make a direct implementation of the logic function:



How can the following truth table be implemented by a logic circuit?

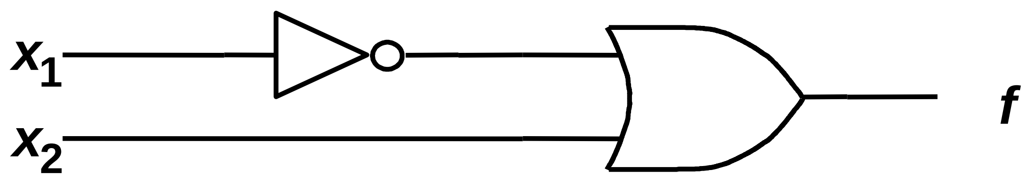
2. (Better) Minimize the logic function

$$
\begin{aligned}
f &= \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2 + x_1 x_2 \\
&= \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2 + \bar{x}_1 x_2 + x_1 x_2 && \text{Add redundant term } \bar{x}_1 x_2 \text{ (7b)} \\
&= \bar{x}_1(\bar{x}_2 + x_2) + (\bar{x}_1 + x_1) x_2 && \text{Distributive rule (12a)} \\
&= \bar{x}_1 \cdot 1 + 1 \cdot x_2 && \text{(8b)} \\
&= \bar{x}_1 + x_2
\end{aligned}
$$

How can the following truth table be implemented by a logic circuit?

3. Implement the minimized function

$$f = \overline{x}_1 + x_2$$



Much simpler implementation!

Discussion: Algebraic optimization

- Algebraic optimization of logic expressions can lead to efficient implementations
- But: For larger circuits, it can be very difficult to identify potential optimizations

We need a generic method that works for all logic circuits!

Minterms and Maxterms

- A **minterm** is a ***product*** term of a logic function in which *all* variables of the logic function are presented.

- A **maxterm** is a ***sum*** term for a logic function in which *all* variables of the logic function are presented.

Minterms and Maxterms

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x}_1\overline{x}_2 x_3$	$M_1 = x_1 + x_2 + \overline{x}_3$
2	0	1	0	$m_2 = \overline{x}_1 x_2\overline{x}_3$	$M_2 = x_1 + \overline{x}_2 + x_3$
3	0	1	1	$m_3 = \overline{x}_1 x_2 x_3$	$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$
4	1	0	0	$m_4 = x_1\overline{x}_2\overline{x}_3$	$M_4 = \overline{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x}_2 x_3$	$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$
6	1	1	0	$m_6 = x_1 x_2\overline{x}_3$	$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

Introduction to SOPs and POSs

- Describe the following logic function by a Boolean expression

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

Sum-of-Products (SOPs)

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	m_1
2	0	1	0	0	
3	0	1	1	0	
4	1	0	0	1	m_4
5	1	0	1	1	m_5
6	1	1	0	1	m_6
7	1	1	1	0	

$$f = \overline{x}_1\overline{x}_2 x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2 x_3 + x_1 x_2\overline{x}_3 = \sum m(1,4,5,6)$$

Sum-of-Products

- A ***sum-of-products*** is a logic expression which is obtained by *adding minterms for which f equals to 1*
  - Also called *disjunctive normal form (DNF)*

Product-of-Sums

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$	
0	0	0	0	0	M_0
1	0	0	1	1	
2	0	1	0	0	M_2
3	0	1	1	0	M_3
4	1	0	0	1	
5	1	0	1	1	
6	1	1	0	1	
7	1	1	1	0	M_7

$$f = (x_1 + x_2 + x_3) \cdot (x_1 + \overline{x}_2 + x_3) \cdot (x_1 + \overline{x}_2 + \overline{x}_3) \cdot (\overline{x}_1 + \overline{x}_2 + \overline{x}_3) = \prod M(0,2,3,7)$$

Product-of-Sums

- A *product-of-sums* is a logic expression which is obtained by multiplying maxterms for which *f* equals to 0
  - Also called *conjunctive normal form (CNF)*

Duality: Minterms and Maxterms, SOP and POS

- For each minterm, there is a corresponding maxterm

$$\overline{m_i} = M_i$$

$$M_0 = \overline{m_0} = \overline{\overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3} = \overline{\overline{x}}_1 + \overline{\overline{x}}_2 + \overline{\overline{x}}_3 = x_1 + x_2 + x_3$$

$$\text{(by DeMorgan 15a)}$$

- For each SOP, there is a corresponding POS

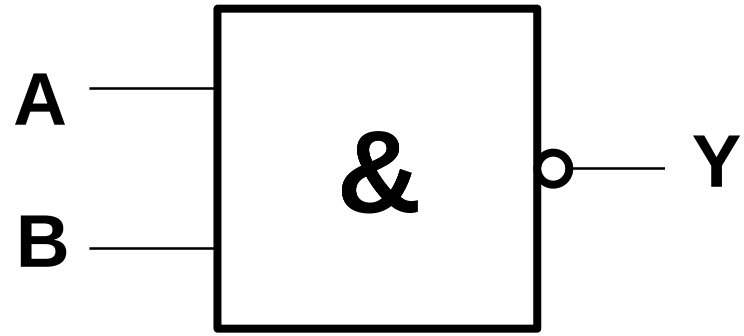
$$f = \sum m(1,4,5,6) = \prod M(0,2,3,7)$$

Logic gates: NAND gate

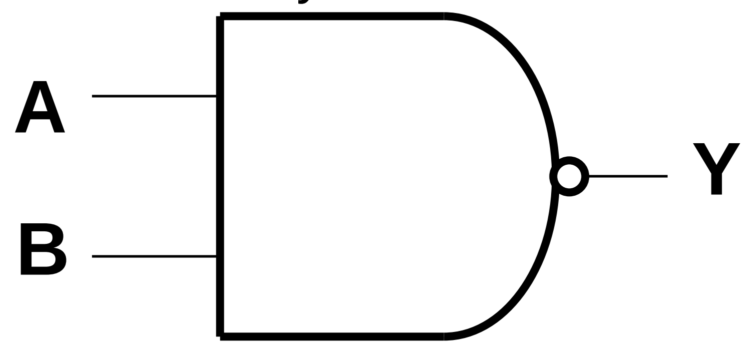
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

$$Y = \overline{A \cdot B}$$

IEC symbol
(International Electrotechnical Commission)

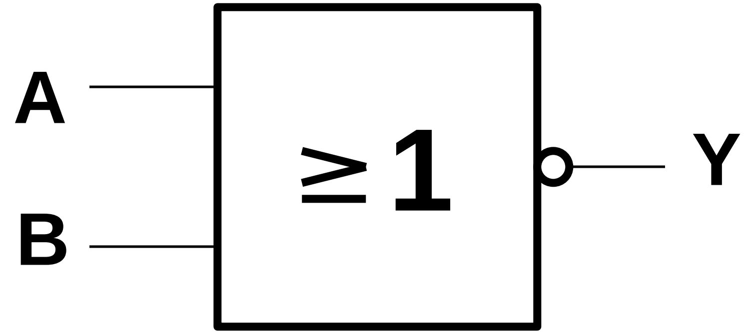


Traditional (American)
Symbol



Logic gates: NOR gate

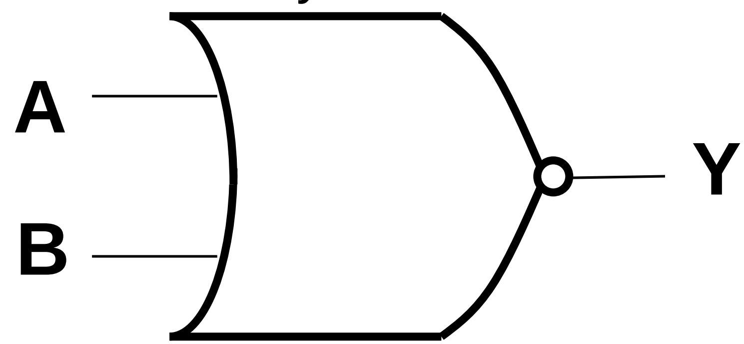
IEC symbol
(International Electrotechnical Commission)



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

$$Y = \overline{A + B}$$

Traditional (American)
Symbol



Only one gate is needed!

- All Boolean functions can be implemented using only NAND or NOR gates

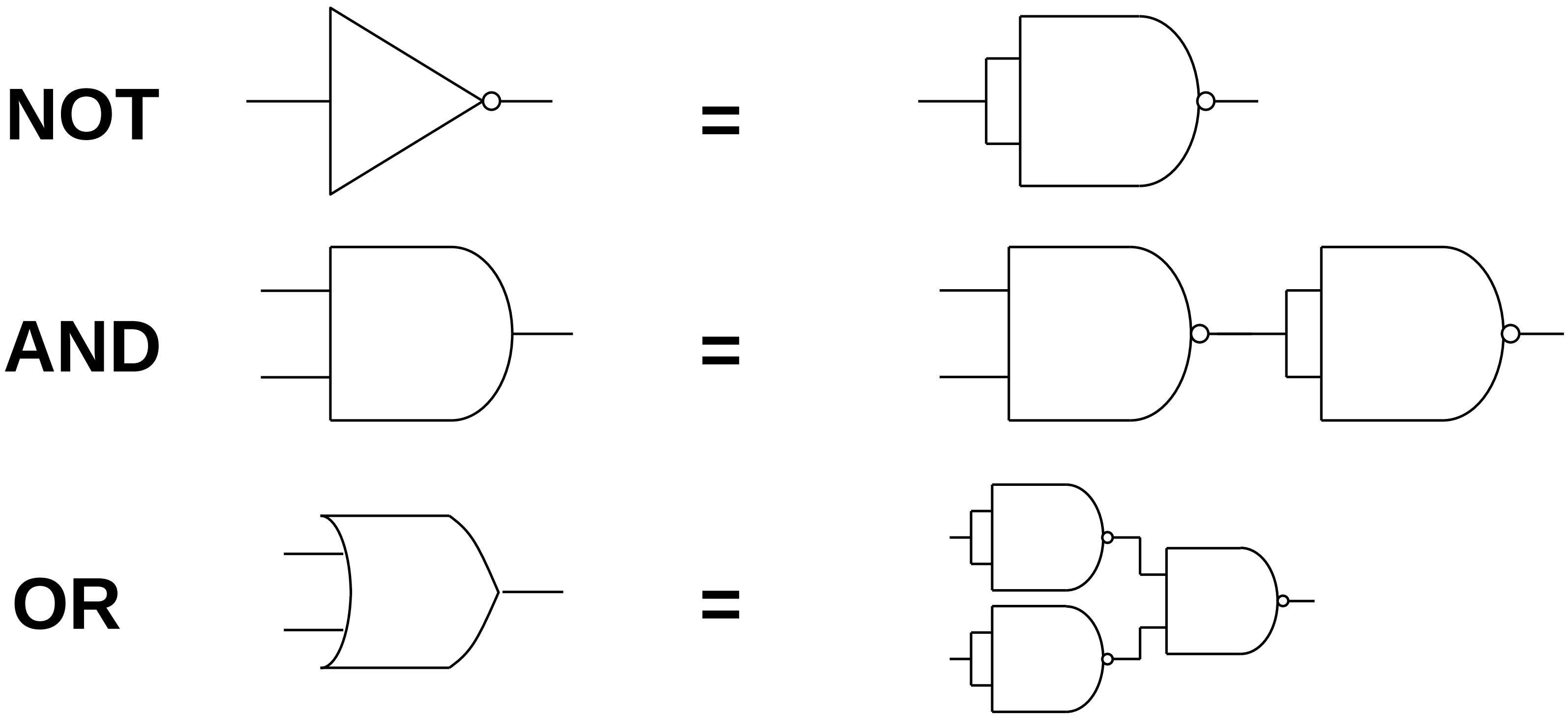
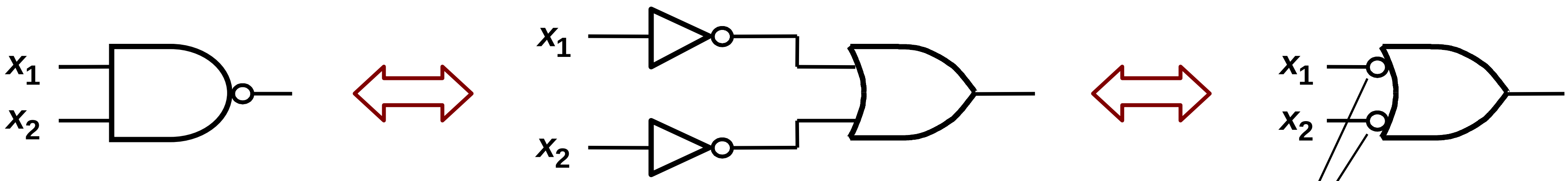
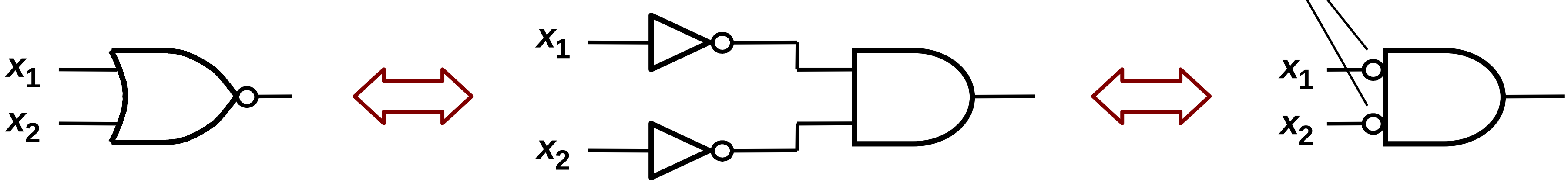


Illustration of DeMorgan's theorem



(A) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$

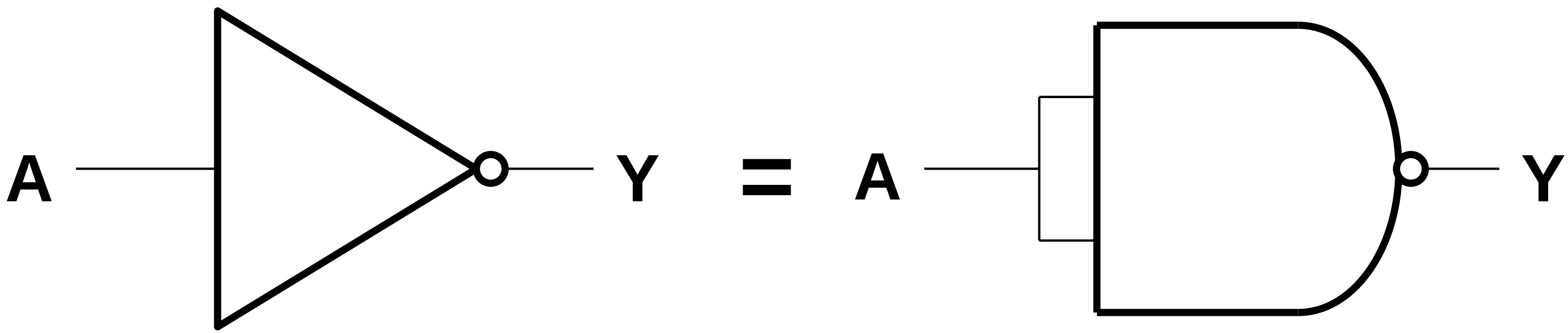
(DeMorgan (15a))



(B) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

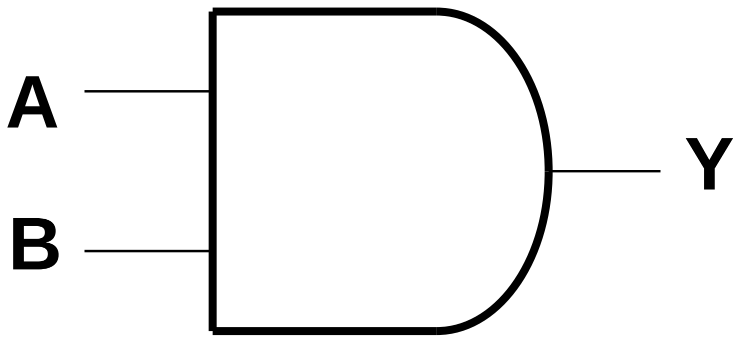
(DeMorgan (15b))

Inverter with NAND gates



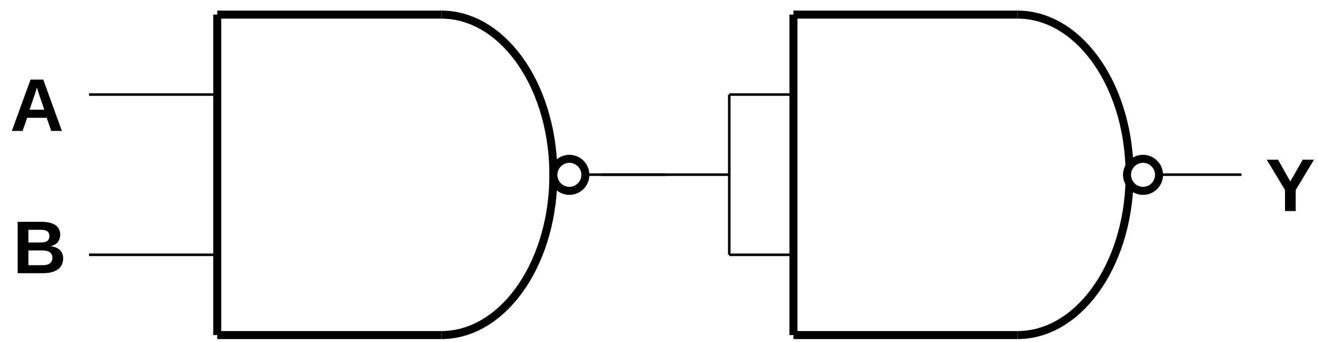
$$Y = \overline{A} = \overline{A \cdot A}$$

AND gate with NAND gates

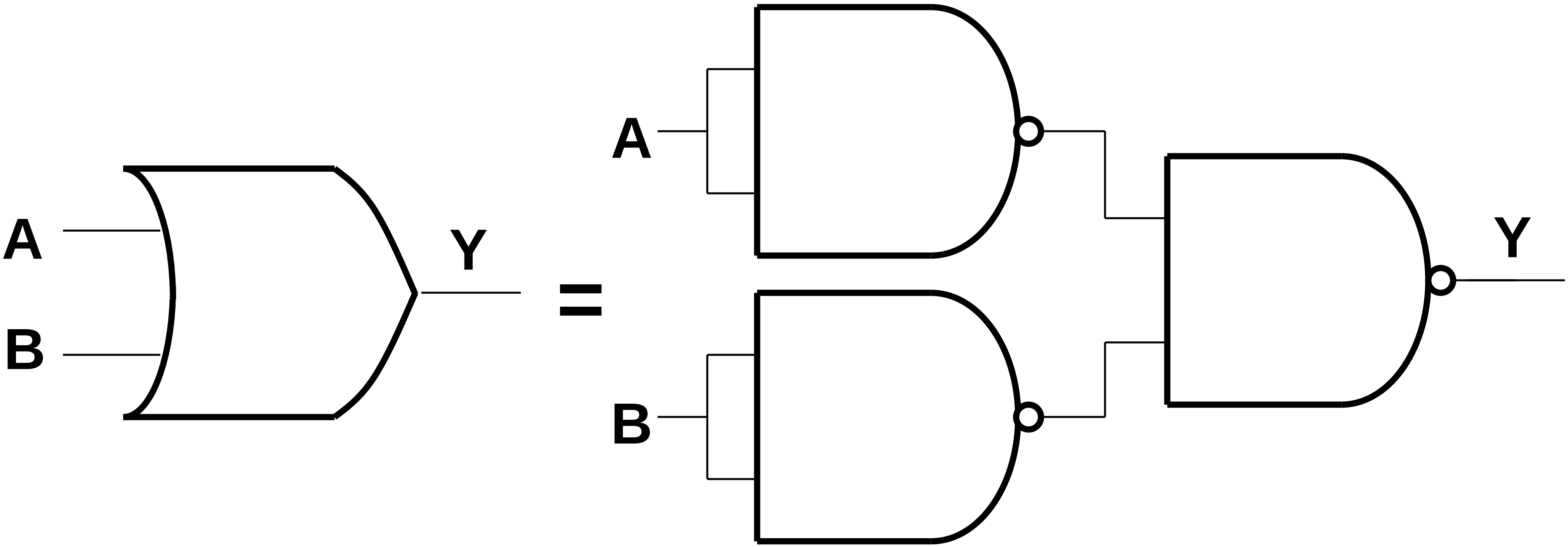


$$Y = A \cdot B = \overline{\overline{A \cdot B}}$$

=

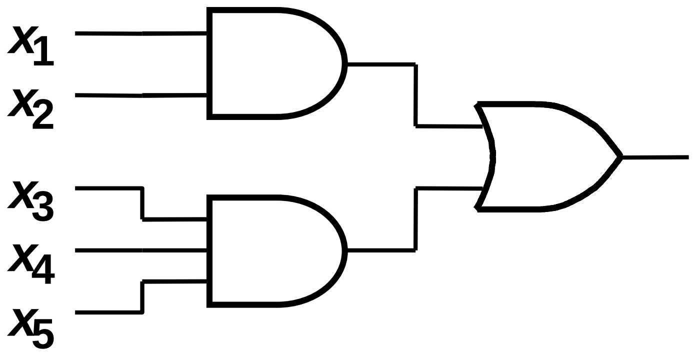


OR gate with NAND gates

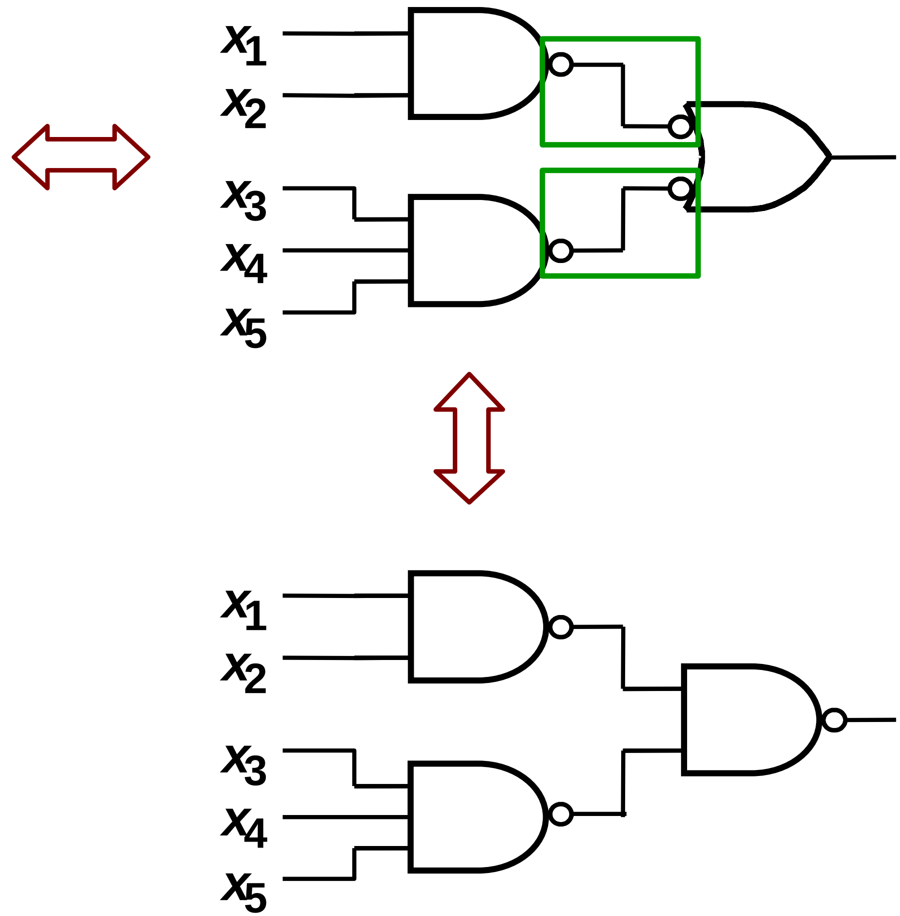


$$Y = A + B = \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A \cdot A} \cdot \overline{B \cdot B}}$$

Implementation of logic functions using only NAND gates



AND-OR function



Universal sets of gates

- A set of gates called universal or functionally complete if all Boolean functions can be implemented using this set
- Examples of universal sets:

{AND, OR, NOT} -> (DeMorgan) -> {AND, NOT} -> {NAND}

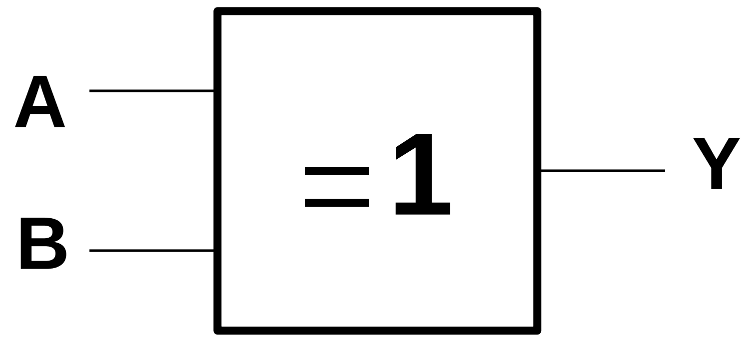
{AND, OR, NOT} -> (DeMorgan) -> {OR, NOT} -> {NOR}

Logic gates: XOR gate (Exclusive OR)

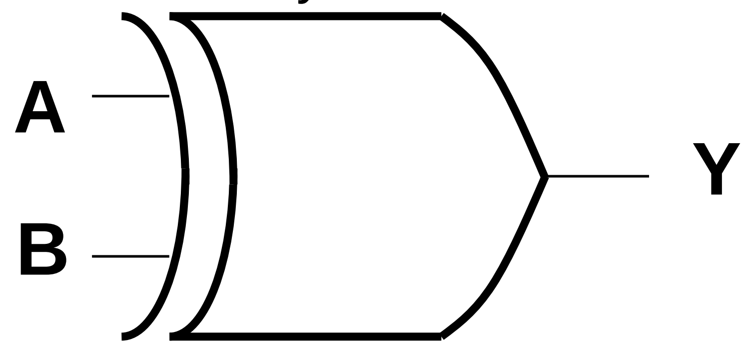
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$$

IEC symbol
(International Electrotechnical Commission)



Traditional (American)
Symbol

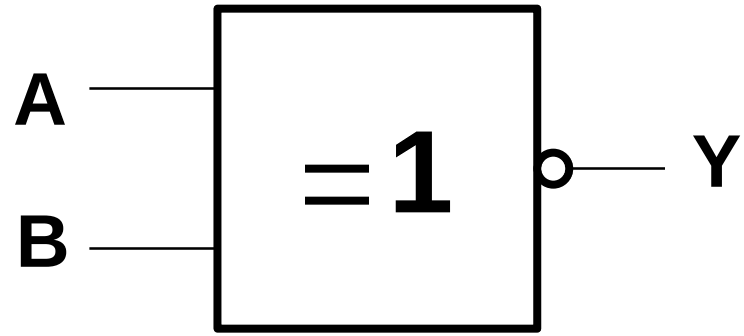


Logic gates: XNOR gate (Exclusive NOR)

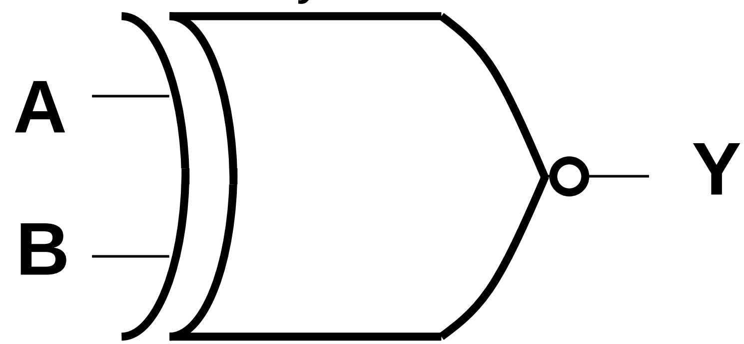
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$$Y = \overline{A \oplus B} = \overline{A} \cdot \overline{B} + A \cdot B$$

IEC symbol
(International Electrotechnical Commission)



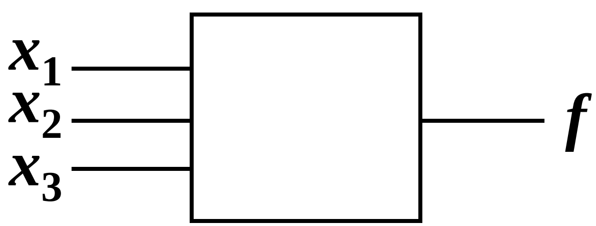
Traditional (American)
Symbol

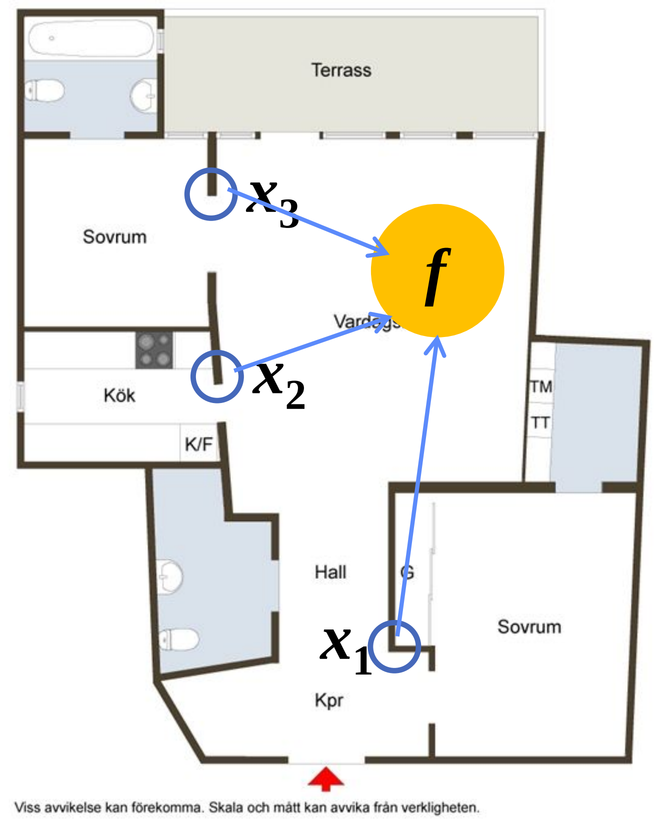


Example: Three-way light control

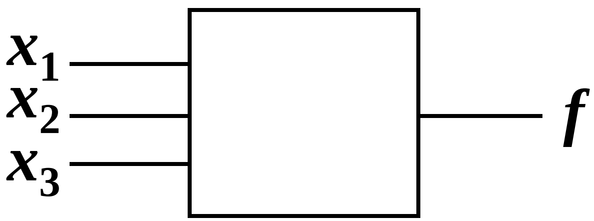
Brown/Vranesic: 2.8.1

Suppose that we need to be able to turn on / off the light from three different places.





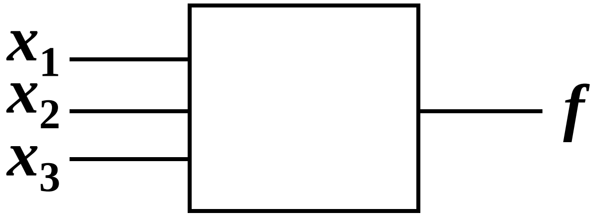
Three-way light control



One should always be able to change the light by changing *any* switch.

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

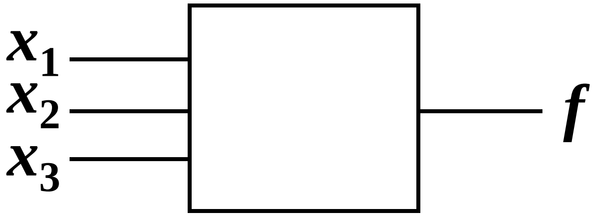
Three-way light control



One should always be able to change the light by changing *any* switch.

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

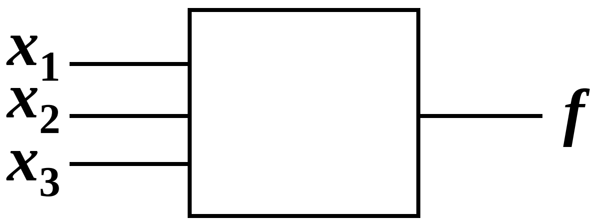
Three-way light control



One should always be able to change the light by changing *any* switch.

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	
1	1	0	
1	1	1	

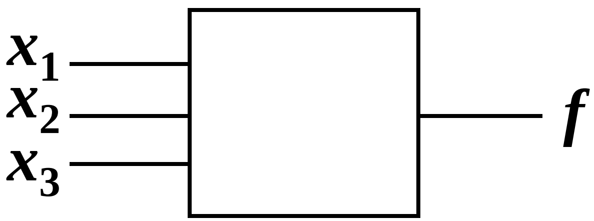
Three-way light control



One should always be able to change the light by changing *any* switch.

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	

Three-way light control



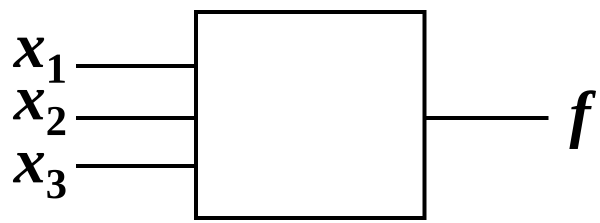
One should always be able to change the light by changing *any* switch.

The truth table now corresponds with the specifications!

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

Three-way light control

x_1	x_2	x_3	f	
0	0	0	0	M_0
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	0	M_3
1	0	0	1	m_4
1	0	1	0	M_5
1	1	0	0	M_6
1	1	1	1	m_7



$$f = \sum m(1,2,4,7) =$$

$$= \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2x_3$$

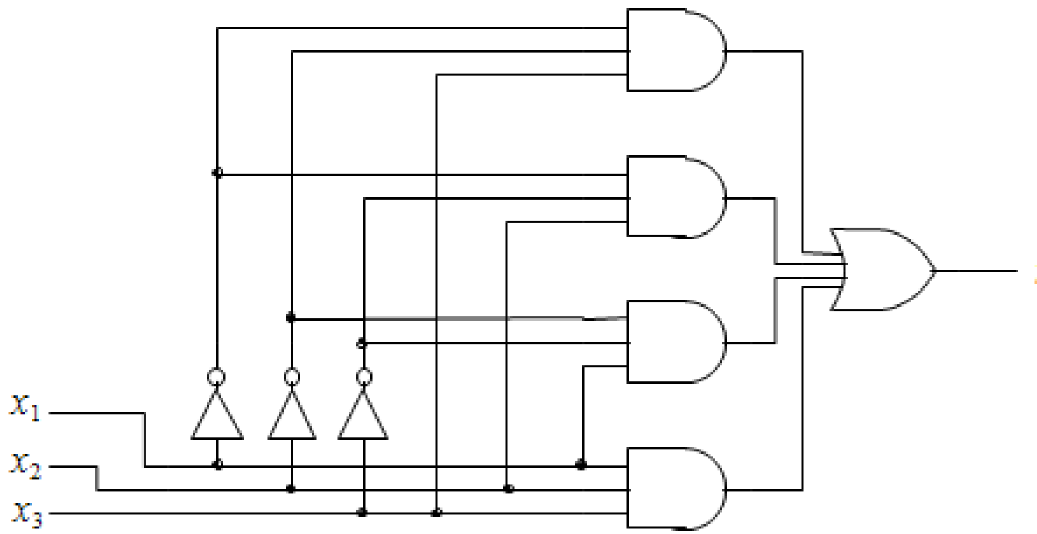
or

$$f = \prod M(0,3,5,6) =$$

$$= (x_1 + x_2 + x_3)\cdot(x_1 + \bar{x}_2 + \bar{x}_3)\cdot(\bar{x}_1 + x_2 + \bar{x}_3)\cdot(\bar{x}_1 + \bar{x}_2 + x_3)$$

Three-way light control

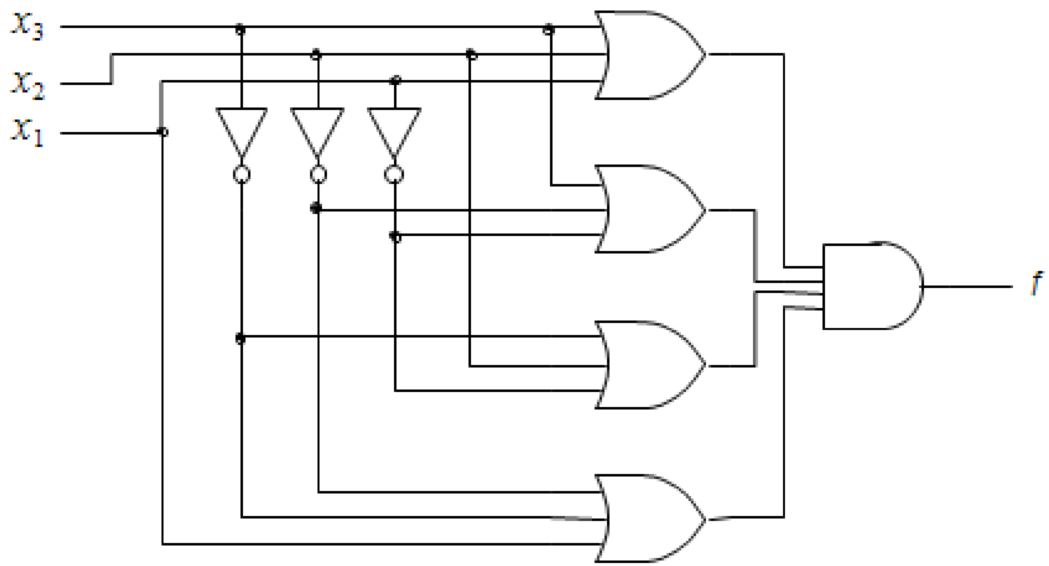
$$f = \sum m(1,2,4,7) = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2x_3$$



(a) Sum-of-products realization

Three-way light control

$$f = \prod M(0,3,5,6) = (x_1 + x_2 + x_3)\cdot(x_1 + \overline{x}_2 + \overline{x}_3)\cdot(\overline{x}_1 + x_2 + \overline{x}_3)\cdot(\overline{x}_1 + \overline{x}_2 + x_3)$$



(b) Product-of-sums realization

Summary

- Logic functions can be described using rules of Boolean algebra
- There are logic gates for many two-variable Boolean functions
- A logic function can be expressed as
  – SOP form (Sum of minterms) or
  – POS form (Product of maxterms)