Logic Design

Chapter 2: Introduction to Logic Circuits

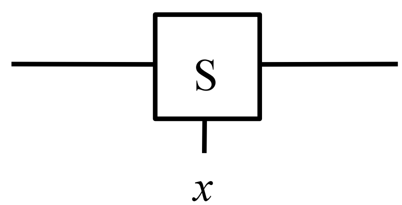
McMaster University

Introduction

- Logic circuits perform operation on digital signal
- Digital signal: signal values are restricted to a few discrete values
- Binary logic circuits: signals can have two values represented by 0 and 1.

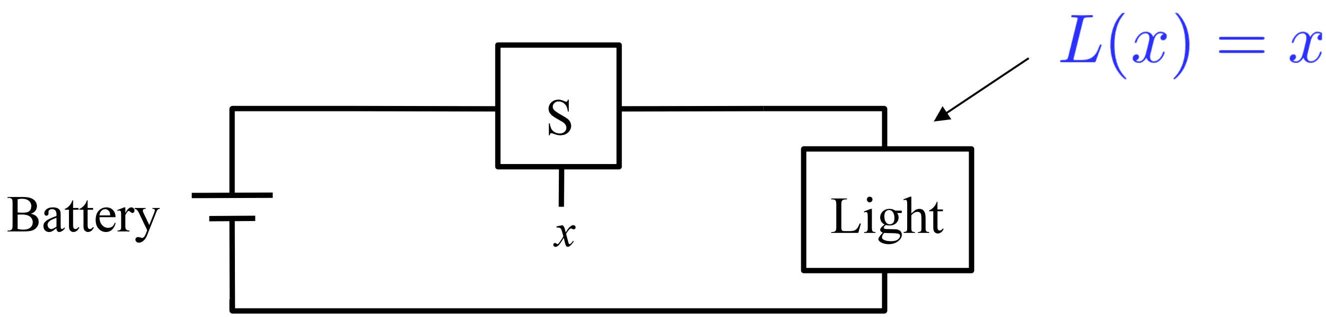
Switch networks

(a) Two states of a switch

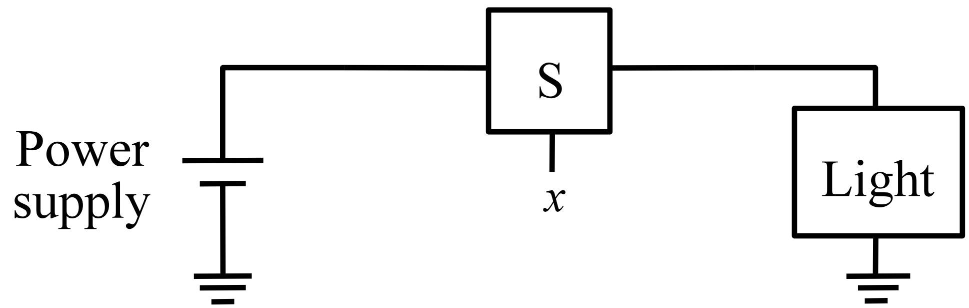


(b) Symbol for a switch

Switch networks

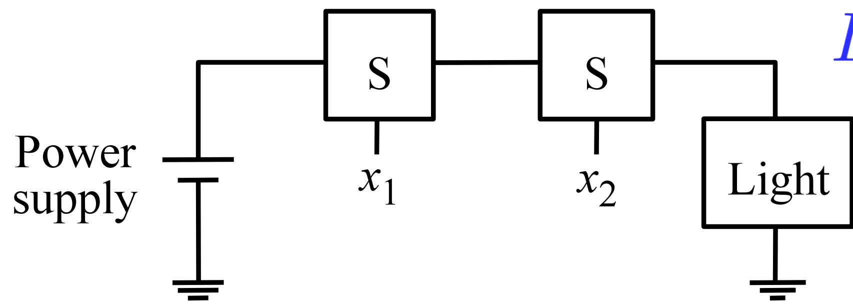


(a) Simple connection to a battery



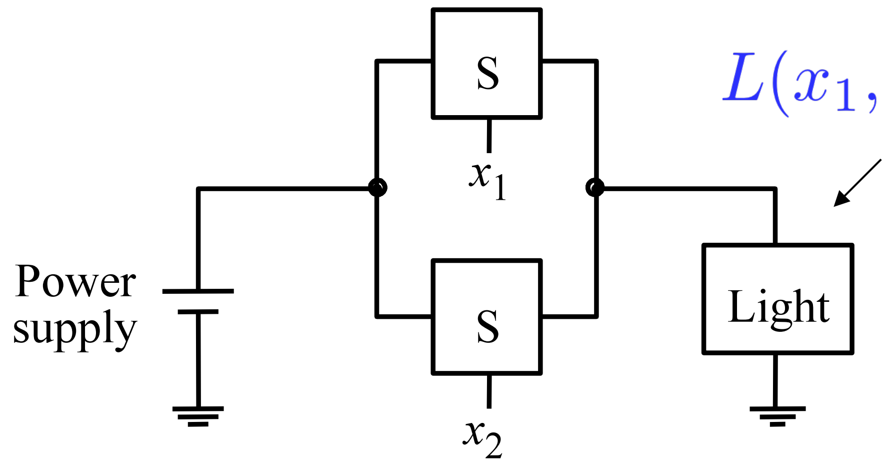
(b) Using a ground connection as the return path

Switch networks



$$L(x_1, x_2) = x_1 . x_2$$

(a) The logical AND function (series connection)

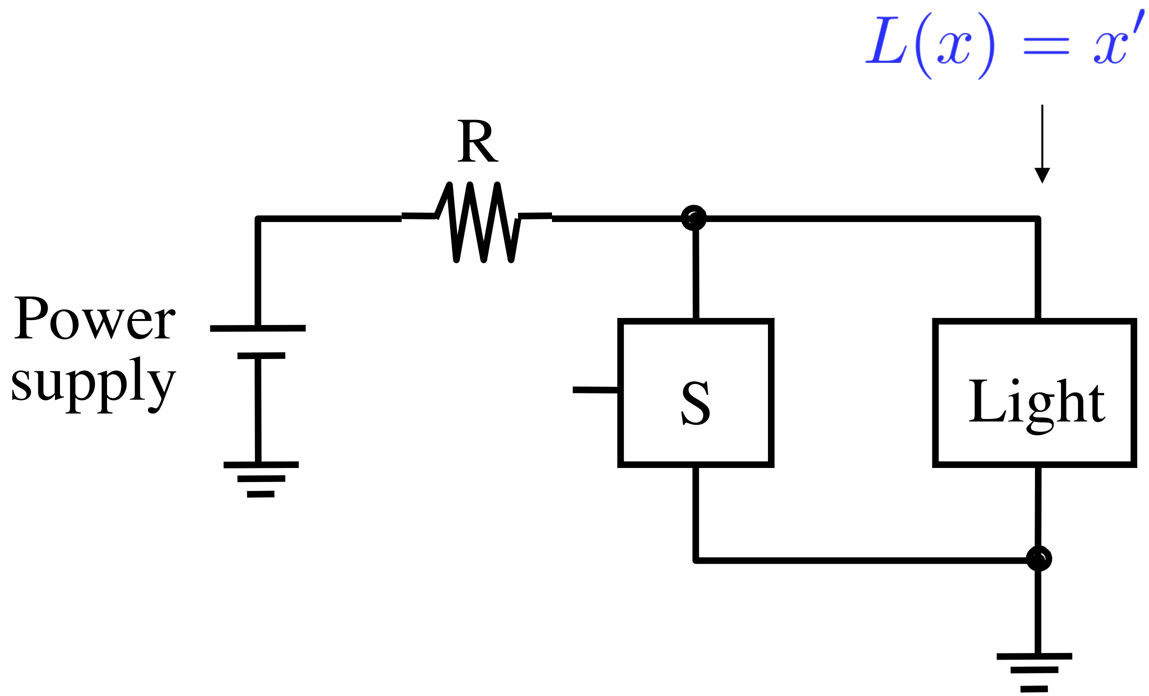


$$L(x_1, x_2) = x_1 + x_2$$

(b) The logical OR function (parallel connection)

McMaster University

Switch networks



Logic Operations

- The *fundamental* logic operations are:

- AND $F = X.Y$
- OR $F = X + Y$
- NOT $F = X'$ (complement)
- Note:
- X' and $\bar{X}$ are used interchangeably!

McMaster University

Logic Operations

- Don’t confuse the AND symbol (.) and OR symbol (+) with arithmetic multiplication and addition
- There are some differences:
- Example:
  - Arithmetic addition: 1+1=2
  - OR operation: 1+1=1
- Based on the context you should recognize if it is AND/OR or addition/multiplication
- One more thing: sometimes we drop the . symbol
- Example: a.b is the same as ab

McMaster University

Truth table

- The most basic representation of a logic function is a truth table.
- A truth table lists the output of the circuit for every possible input combination.
- There are 2^n rows in a truth table for an n-variable function

Truth table:

X	Y	XY	X + Y	X'
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

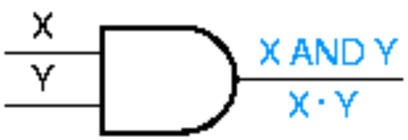
McMaster University

Logic Gate

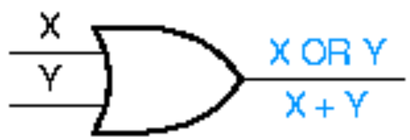
- Binary signals are manipulated using *logic gates*. These are electronic devices whose inputs and outputs are interpreted with only two values, representing logic 0 and logic 1.

Logic Gate

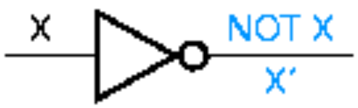
- The bubble on the inverter output denotes "inverting" behavior



X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1



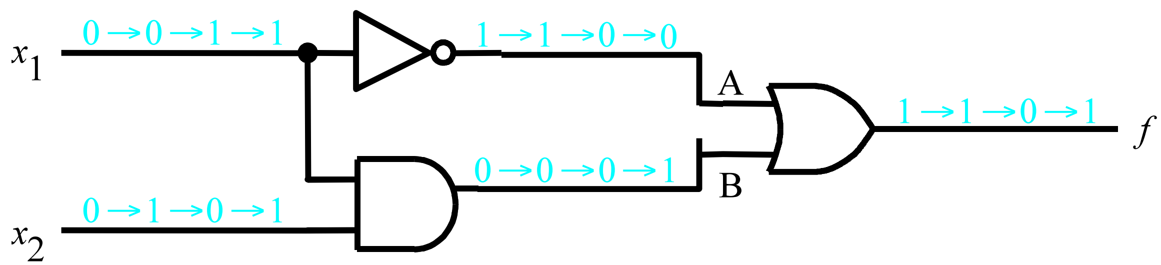
X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1



X	NOT X
0	1
1	0

Analysis and Synthesis of a Logic Network

- Combinations of gates form a logic circuit or logic network
- Analysis: For an existing network determine the function performed by the network
- Synthesis: Design a network that implements a desired function

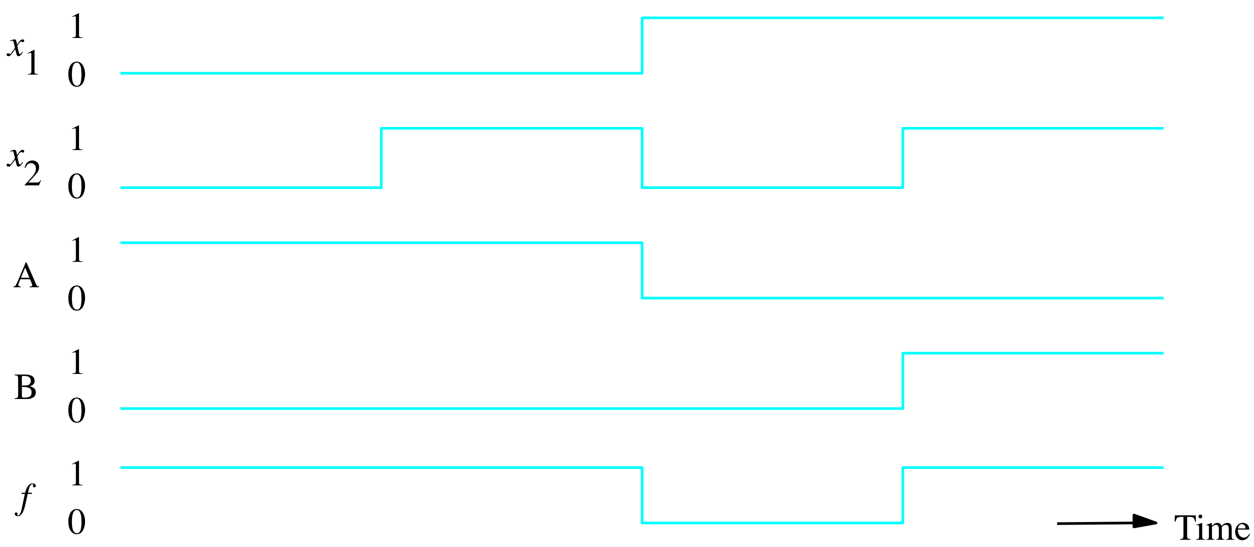


(a) Network that implements $f = \bar{x}_1 + x_1 \cdot x_2$

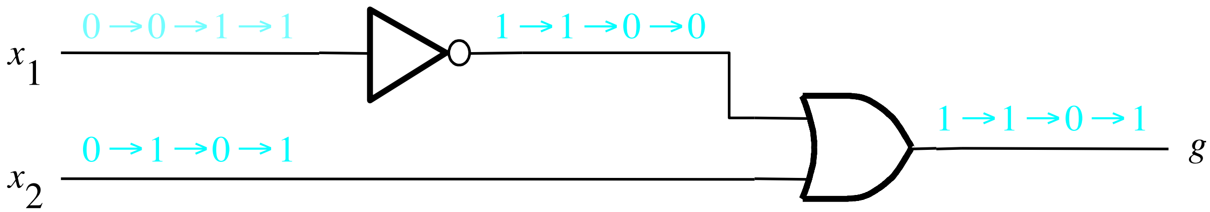
x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

A	B
1	0
1	0
0	0
0	1

(b) Truth table



(c) Timing diagram



(d) Network that implements $g = \bar{x}_1 + x_2$

McMaster University

Boolean Algebra

- A variety of implementations are available for a logic function
- How to find the best implementation?

Boolean Algebra

- To design logic circuits and describe their operation we use a mathematical tool called *Boolean algebra* (from English mathematician George Boole in 1800's) that operates on two-valued functions.

Axioms of Boolean algebra

- The axioms (or postulates) of a mathematical system are a minimal set of basic definitions that we assume to be true.
- The first three pairs of axioms state the formal definitions of the AND (logical multiplication) and OR (logical addition) operations:

(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) $1+0 = 0+1 = 1$

- The next axioms embody the complement notation:

(4a) If $X=0$, then $X'=1$ (4b) If $X=1$, then $X'=0$

McMaster University

Theorems of Boolean algebra

- Theorems are statements, known to be true, that allow us to manipulate algebraic expressions to have simpler analysis or more efficient synthesis of the corresponding circuits.
- Theorems involving a single variable:

(5a) $X \cdot 0 = 0$ (5b) $X+1 = 1$ (Null elements)

(6a) $X \cdot 1 = X$ (6b) $X+0 = X$ (Identities)

(7a) $X \cdot X = X$ (7b) $X+X = X$ (Idempotency)

(8a) $X \cdot X' = 0$ (8b) $X+X' = 1$ (Complements)

(9) $(X')' = X$ (Involution)

- These theorems can be proved to be true. Let us prove 6b:

[X=0] 0+0=0 (true, according to 2b)

[X=1] 1+0=1 (true, according to 3b)

McMaster University

Theorems of Boolean algebra

- Theorems involving two or three variables:

(10a) $X \cdot Y = Y \cdot X$ (10b) $X+Y = Y+X$ (Commutativity)

(11a) $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$ (11b) $(X+Y)+Z = X+(Y+Z)$ (Associativity)

(12a) $X \cdot Y+X \cdot Z = X \cdot (Y+Z)$ (12b) $(X+Y) \cdot (X+Z) = X+Y \cdot Z$

(Distributivity)

(13a) $X+X \cdot Y = X$ (13b) $X \cdot (X+Y) = X$ (Absorption)

(14a) $X \cdot Y+X \cdot Y' = X$ (14b) $(X+Y) \cdot (X+Y') = X$ (Combining)

(15a) $(X_1 \cdot X_2)' = X_1' + X_2'$

(15b) $(X_1 + X_2)' = X_1' \cdot X_2'$ DeMorgan's theorems

(16a) $X+X' \cdot Y = X+Y$ (16b) $X \cdot (X'+Y) = X \cdot Y$ (simplification)

(17a) $X \cdot Y+X' \cdot Z+Y \cdot Z = X \cdot Y+X' \cdot Z$ (Consensus)

(17b) $(X+Y) \cdot (X'+Z) \cdot (Y+Z) = (X+Y) \cdot (X'+Z)$

McMaster University

Duality

- Theorems were presented in pairs.
- The b version of a theorem is obtained from the a version by swapping "0" and "1", and "·" and "+".
- Principle of Duality: Any theorem or identity in Boolean algebra remains true if 0 and 1 are swapped and · and + are swapped throughout.
- Duality is important because it doubles the usefulness of everything about Boolean algebra and manipulation of logic functions.

McMaster University

Consensus Theorem:

- $XY + \overline{X}Z + YZ = XY + \overline{X}Z$

redundant

Note: Y and Z are associated with X and $\overline{X}$, and appear together in the term that is eliminated.

Consensus theorem

- Using duality:

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

Boolean Algebra

$X + 0 = X$	$X \cdot 1 = X$	Identity
$X + 1 = 1$	$X \cdot 0 = 0$	
$X + X = X$	$X \cdot X = X$	Idempotent Law
$X + X' = 1$	$X \cdot X' = 0$	Complement
$(X')' = X$		Involution Law
$X + Y = Y + X$	$XY = YX$	Commutativity
$X + (Y + Z) = (X + Y) + Z$	$X(YZ) = (XY)Z$	Associativity
$X(Y + Z) = XY + XZ$	$X + YZ = (X + Y)(X + Z)$	Distributivity
$X + XY = X$	$X(X + Y) = X$	Absorption Law
$X + X'Y = X + Y$	$X(X' + Y) = XY$	Simplification
$(X + Y)' = X'Y'$	$(XY)' = X' + Y'$	DeMorgan's Law
$XY + X'Z + YZ = XY + X'Z$	$(X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)$	Consensus Theorem

McMaster University

Boolean Algebra

- Differences between Boolean and ordinary algebra:
- Distributive law of + over . $x+(y.z)=(x+y).(x+z)$ is not valid in ordinary algebra
- Boolean algebra does not have additive or multiplicative inverse so there is no subtraction or division operations

McMaster University

Boolean Algebra

- Boolean algebra is used for manipulating logical functions when designing digital hardware.

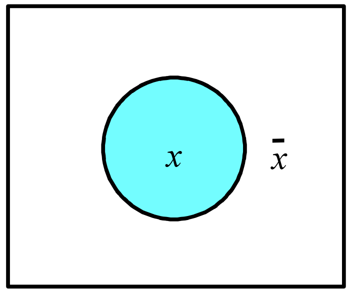
- However, today most design is done using Computer-Aided Design (CAD) software that includes schematic capture, logic simplification and simulation.

- Other methods include truth tables, Venn diagrams and Karnaugh Maps.

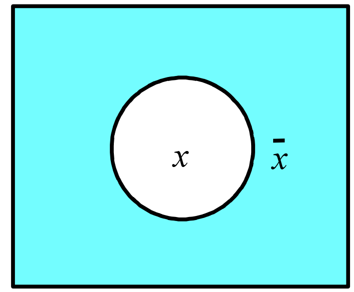
Venn Diagram

- A graphical tool that can be used for Boolean algebra
- A binary variable s is represented by a contour
- Area within the contour corresponds to s=1
- Area outside the contour corresponds to s=0
- Two variables are represented by two overlapping circles

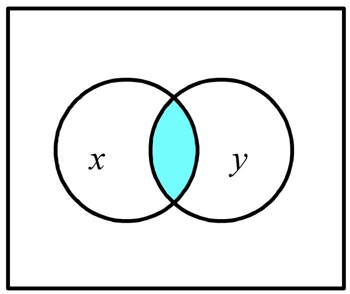
Venn Diagram



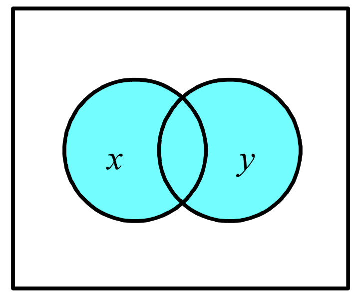
(c) Variable x



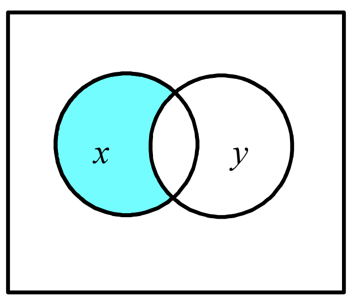
(d) $\bar{x}$



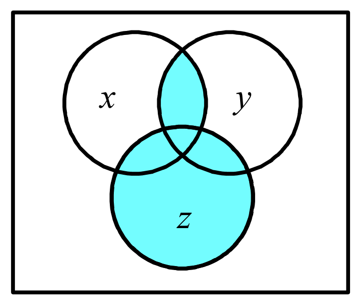
(e) $x \cdot y$



(f) $x + y$



(g) $x \cdot \bar{y}$



(h) $x \cdot y + z$

Venn Diagram

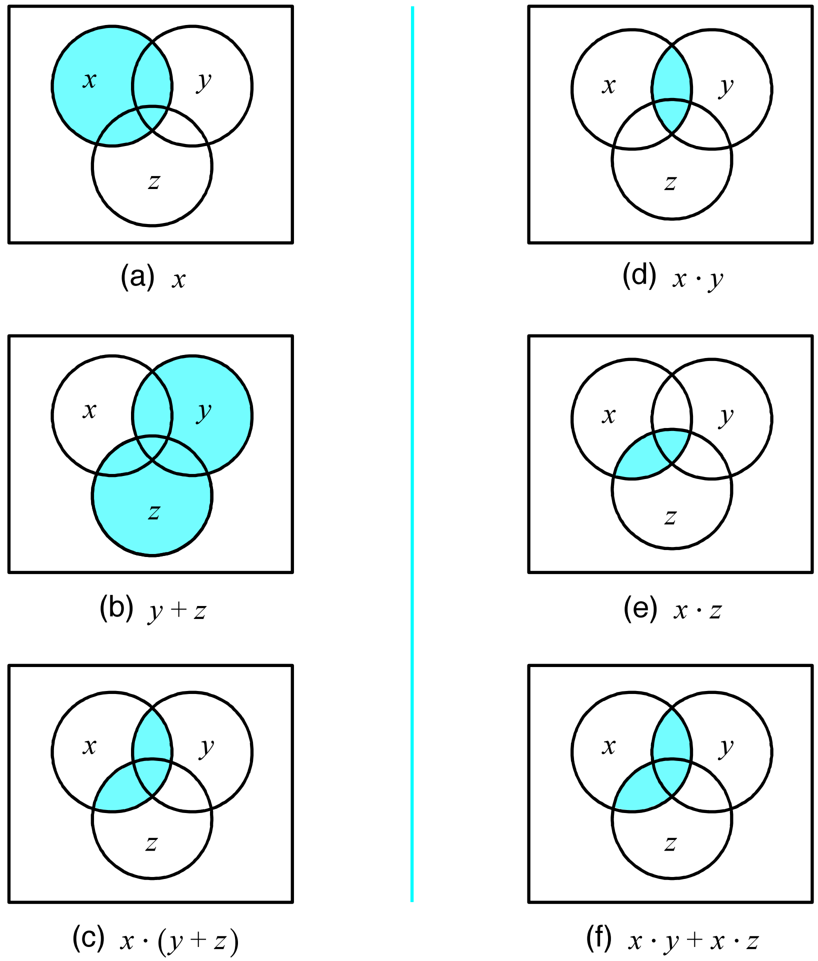


Figure 2.13. Verification of the distributive property

$$x \quad (y + z) = x \quad y + x \quad z$$

Precedence of operations

- In the absence of parentheses, operations in a logic expression must be performed in the order: NOT, AND, OR
- Example:

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

Synthesis using AND, OR and NOT

- One way of designing a logic circuit that implements a truth table is to create a product term that has a value of 1 for each valuation for which the output function has to be 1.
- Then we take the logical sum of these product terms to realize f

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

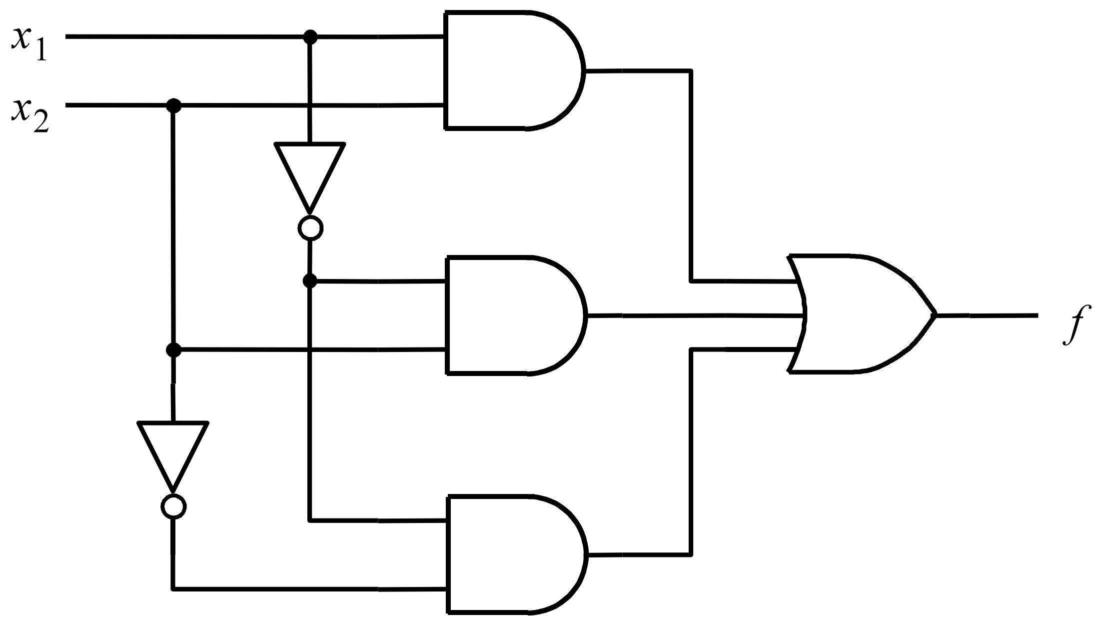
McMaster University

$$f(x_1, x_2) = \overline{x_1}\,\overline{x_2} + \overline{x_1} x_2 + x_1 x_2$$

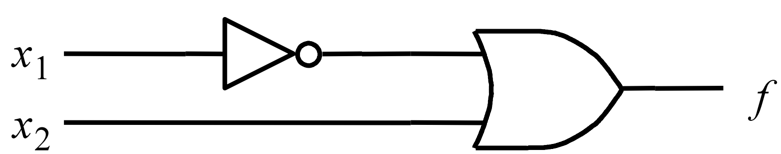
$$f = \overline{x_1} + x_2$$

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

McMaster University



(a) Canonical sum-of-products



(b) Minimal-cost realization

McMaster University

Minterm, Maxterm

- Minterm
- A product term in which all variables of a function appear exactly once, uncomplemented or complemented.

- Maxterm
- A sum term in which all variables of a function appear exactly once, uncomplemented or complemented.

Minterm, Maxterm

For a Boolean function of n variables, there are 2^n minterms:

$$m_0 \,..\, m_{2^n-1}$$

and 2^n maxterms:

$$M_0 \,..\, M_{2^n-1}$$

Note that: $M_i = \overline{m_i}$

McMaster University

Minterm, Maxterm

Row No.	A B C	Minterms	Maxterms
0	0 0 0	$A'B'C' = m_0$	$A+B+C = M_0$
1	0 0 1	$A'B'C = m_1$	$A+B+C' = M_1$
2	0 1 0	$A'BC' = m_2$	$A+B'+C = M_2$
3	0 1 1	$A'BC = m_3$	$A+B'+C' = M_3$
4	1 0 0	$AB'C' = m_4$	$A'+B+C = M_4$
5	1 0 1	$AB'C = m_5$	$A'+B+C' = M_5$
6	1 1 0	$ABC' = m_6$	$A'+B'+C = M_6$
7	1 1 1	$ABC = m_7$	$A'+B'+C' = M_7$

$$M_i = m_i'$$

McMaster University

Canonical Sum of Products Form

- A Boolean function f(x1,x2,x3) can be expressed algebraically as a logical *sum of minterms*:

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

McMaster University

Canonical Sum of Products Form

- f can be expressed as sum of product terms (SOP)

$$f(x1, x2, x3) = \sum (m1, m4, m5, m6)$$

$$f(x1, x2, x3) = \sum m(1,4,5,6)$$

Canonical Product of Sums Form

- The *complement* of f(x1,x2,x3) can be formed as the logical sum of all minterms not used in f(x1,x2,x3):

$$\bar{f}(x1, x2, x3) = m0 + m2 + m3 + m7$$

$$f = \overline{m0 + m2 + m3 + m7}$$

$$f = \overline{m0}.\overline{m2}.\overline{m3}.\overline{m7}$$

$$f = M0.M2.M3.M7$$

This is called the product of sum presentation of f

McMaster University

Conversion Between the Canonical Forms

- It is easy to convert from one canonical form to other one, simply use the DeMorgan's theorem.
- Example:

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

$$F^{'}(A,B,C) = \sum(0,2,3)$$

$$F(A,B,C) = (m0 + m2 + m3)^{'} = m_0^{'}m_2^{'}m_3^{'} = M_0M_2M_3$$

$$F(A,B,C) = \prod(0,2,3)$$

McMaster University

Cost of a Logic Circuit

- Cost of a logic circuit: total number of gates plus total number of inputs to all gates in the circuit
- The canonical SOP and POS implementations described before are not necessarily minimum cost
- We can simplify them to obtain minimum-cost SOP and POS circuits

Reducing Cost

- How can we simplify a logic function?
- There are systematic approached for doing this (e.g., Karnaugh map) that we will learn later
- The other way is to use theorems and properties of Boolean algebra and do algebraic manipulations.
- Do an example on the board.

Reducing Cost

- The simplified version of SOP is called minimal SOP
- The simplified version of POS is called minimal POS
- We cannot in general predict whether the minimal SOP expression or minimal POS expression will result in the lowest cost.
- It is often useful to check both expressions to see which gives the best result.

Other Logic Operations

- NAND,
- NOR,
- XOR,
- XNOR

McMaster University

NAND

- NAND: a combination of an AND gate followed by an inverter.

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

- Symbol for NAND is $\uparrow$
- NAND gates have several interesting properties:

$$A \uparrow A = A'$$
$$(A \uparrow B)' = AB$$
$$(A' \uparrow B') = A + B$$

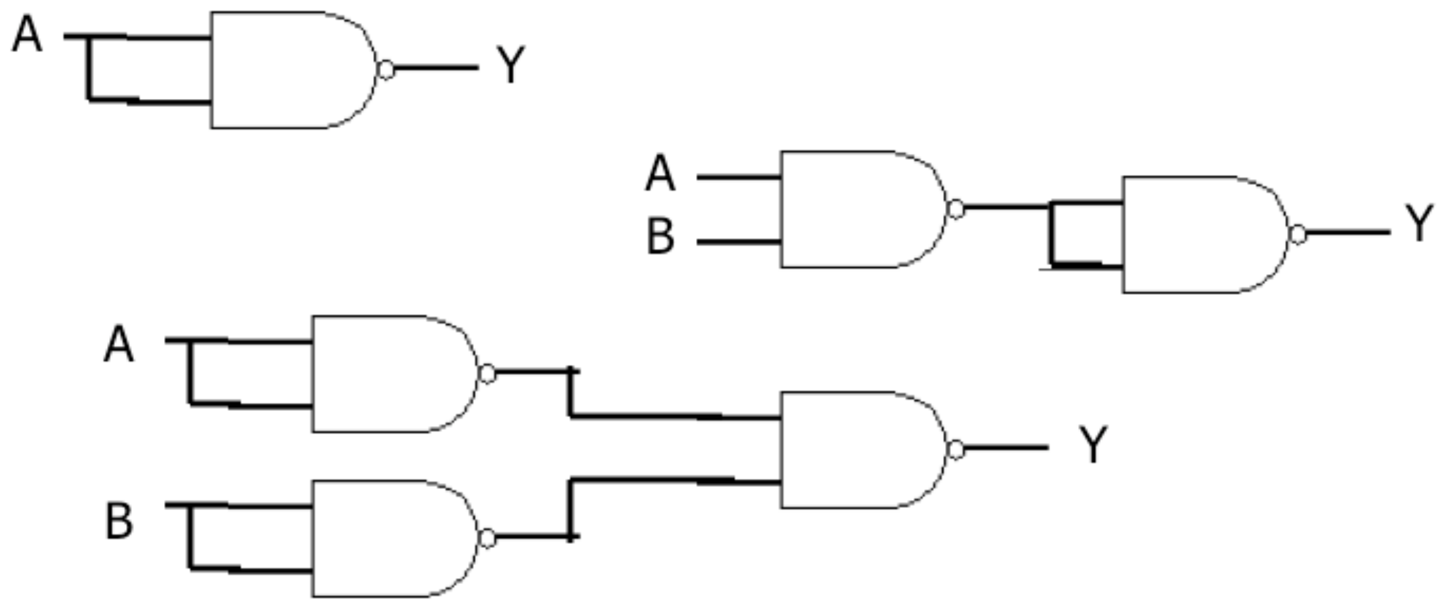
McMaster University

NAND

- These three properties show that a NAND gate with both of its inputs driven by the same signal is equivalent to a NOT gate
- A NAND gate whose output is complemented is equivalent to an AND gate, and a NAND gate with complemented inputs acts as an OR gate.
- Therefore, we can use a NAND gate to implement all three of the *elementary operators* (AND,OR,NOT).
- Therefore, ANY Boolean function can be constructed using only NAND gates.

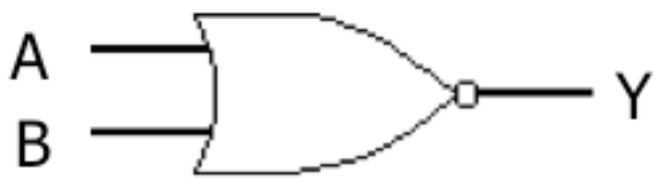
McMaster University

NAND



NOR

- NOR: a combination of an OR gate followed by an inverter.



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

- NOR gates also have several interesting properties:

$$A \downarrow A = A'$$
$$(A \downarrow B)' = A + B$$
$$A' \downarrow B' = AB$$

McMaster University

NOR

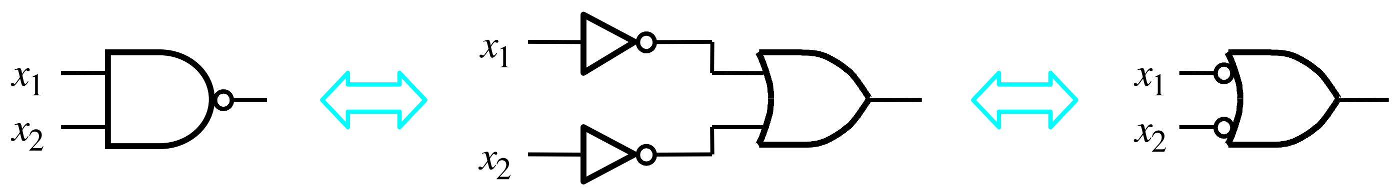
- Just like the NAND gate, any logic function can be implemented using just NOR gates.
- Both NAND and NOR gates are very valuable as any design can be realized using either one.
- It is easier to build an IC chip using all NAND or NOR gates than to combine AND,OR, and NOT gates.
- NAND/NOR gates are typically faster at switching and cheaper to produce.

NAND and NOR networks

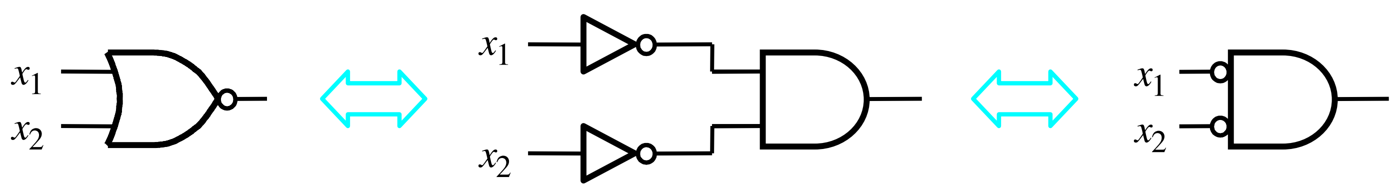
- NAND and NOR can be implemented by simpler electronic circuits than the AND and OR functions
- Can these gates be used in synthesis of logic circuits?

McMaster University

NAND and NOR networks

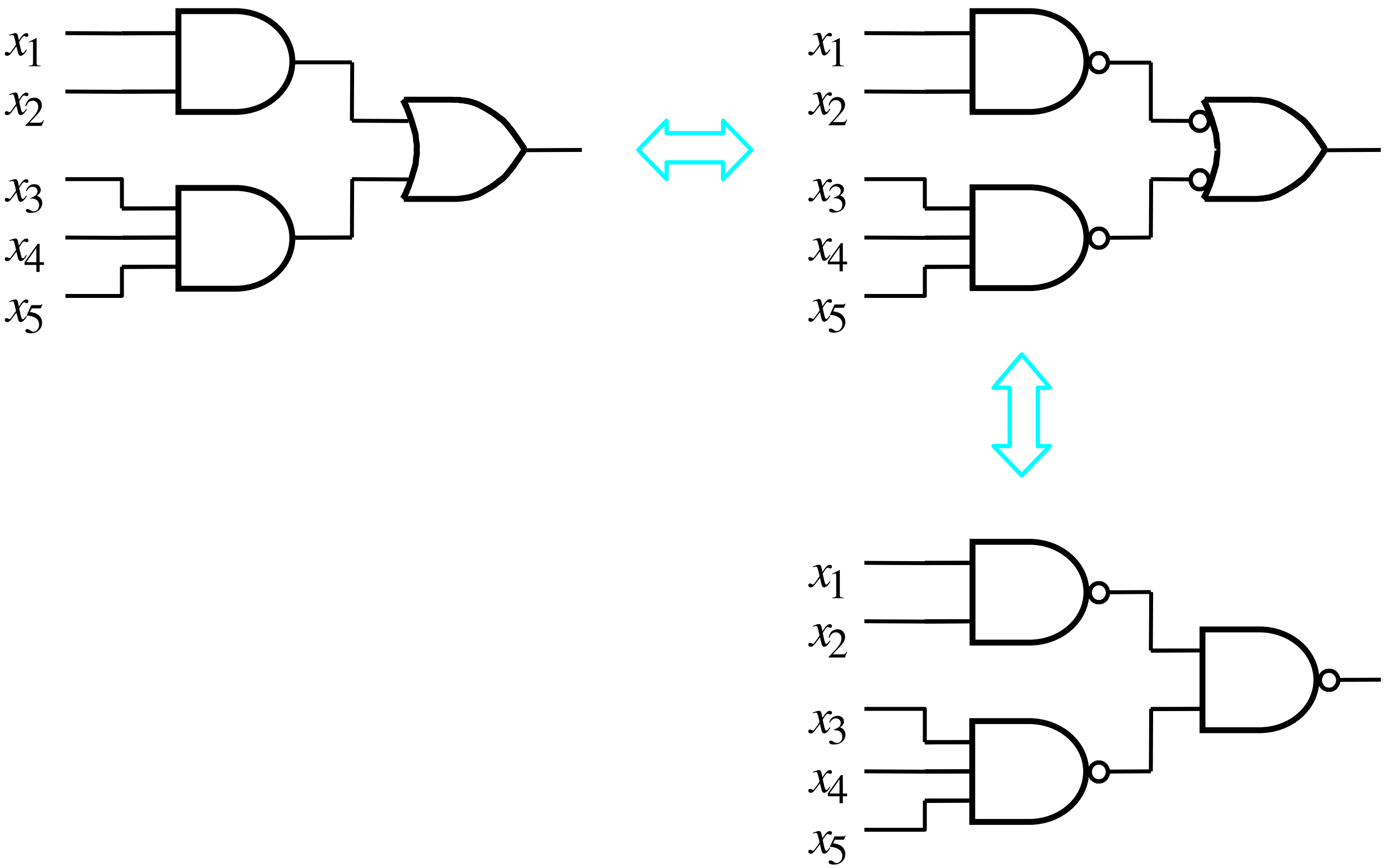


(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$

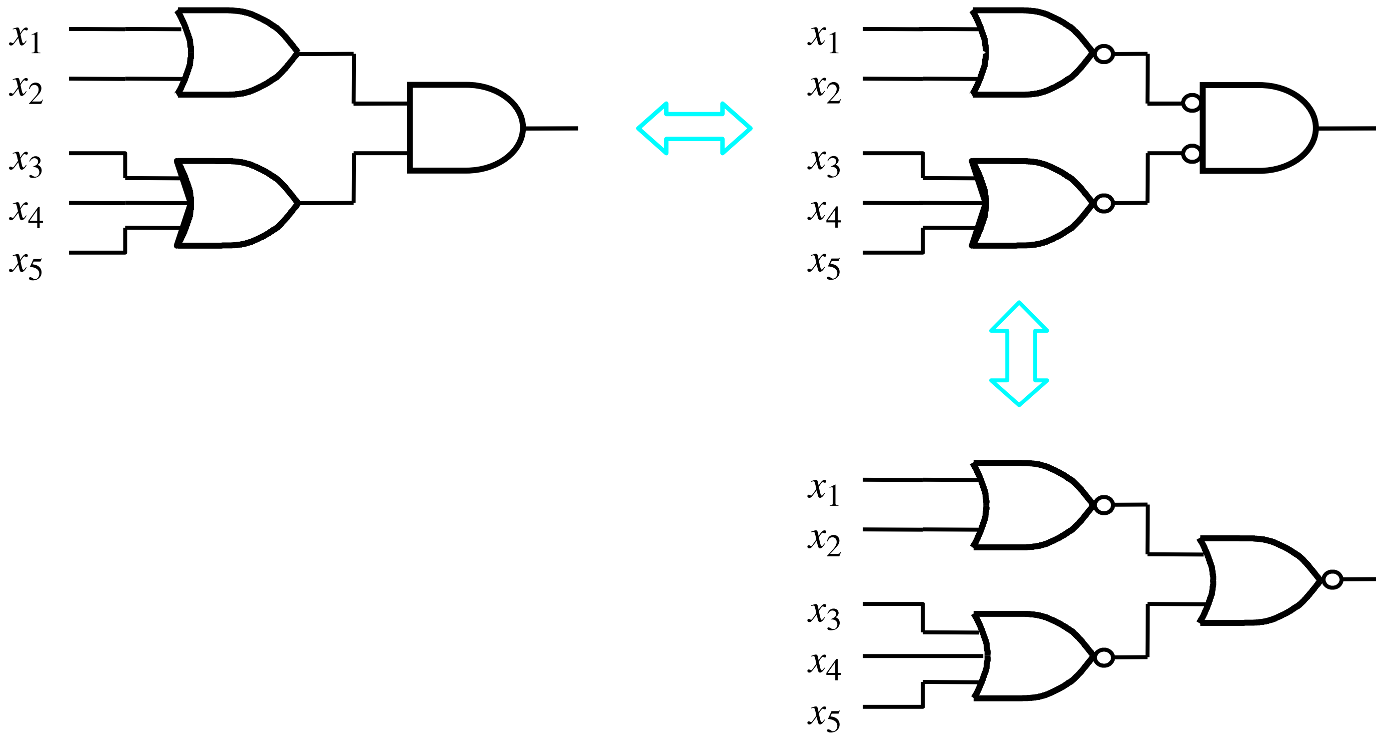


(b) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

NAND and NOR networks



NAND and NOR networks



Exclusive OR (XOR)

- The eXclusive OR (XOR) function is an important Boolean function used extensively in logic circuits.
- The XOR function maybe:
  - implemented directly as an electronic circuit (truly a gate) or
  - implemented by interconnecting other gate types (used as a convenient representation)
- The XOR function means:

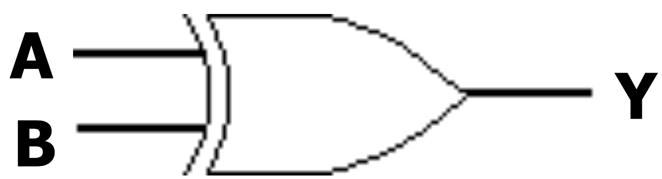
  X OR Y, but NOT BOTH

McMaster University

XOR

- XOR gates assert their output when exactly one of the inputs is asserted, hence the name.
- The symbol for this operation is $\oplus$

$$Y = A'B + AB'$$



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

XNOR

- The eXclusive NOR function is the complement of the XOR function
- The symbol for this operation is $\odot$, i.e. $1 \odot 1 = 1$ and $1 \odot 0 = 0$.

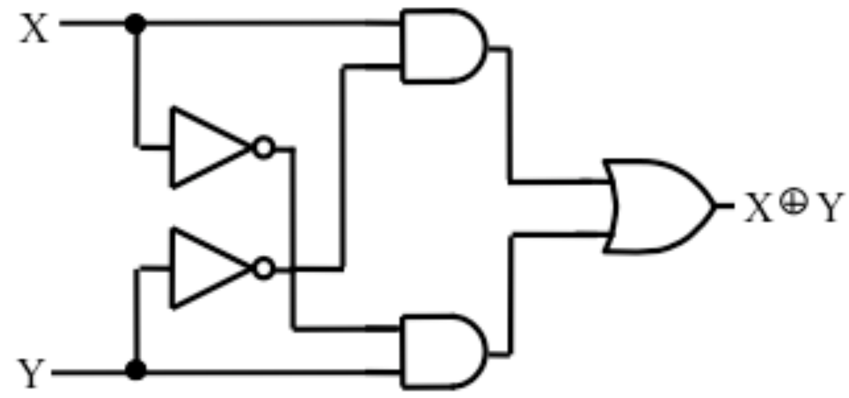
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$$Y = A'B' + AB$$

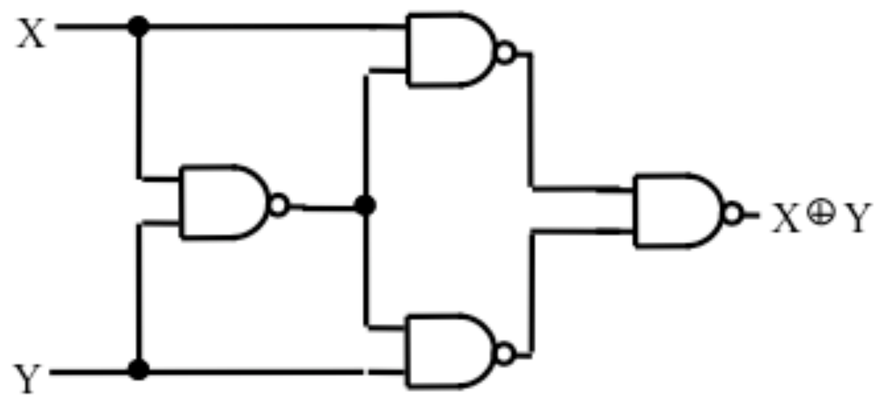
- Why is the XNOR function also known as the *equivalence* function?

XOR Implementations

- A SOP implementation



- A NAND implementation



XOR and XNOR

- Uses for the XOR and XNORs gate include:
  - Adders/subtractors/multipliers
  - Counters/incrementers/decrementers
  - Parity generators/checkers

XOR

- XOR identities:

$$X \oplus 0 = X$$

$$X \oplus X = 0$$

$$X \oplus Y = Y \oplus X$$

$$X \oplus 1 = X'$$

$$X \oplus X' = 1$$

McMaster University

Gates with more than two inputs

- A gate can be extended to have multiple inputs if the binary operation it represents is commutative and associative.
- AND and OR operations have these two properties
- NAND and NOR are not associative:

$$(A \downarrow B) \downarrow C \neq A \downarrow (B \downarrow C)$$
$$(A \uparrow B) \uparrow C \neq A \uparrow (B \uparrow C)$$

Gates with more than two inputs

- We define multiple input NAND and NOR gates as follows:

$$A \downarrow B \downarrow C = (A + B + C)'$$
$$A \uparrow B \uparrow C = (ABC)'$$

Gates with more than two inputs

- XOR and XNOR are both commutative and associative
- Definition of XOR should be modified for more than two inputs
- For more than 2 inputs, XOR is called an *odd function*: it is equal to 1 if the input variables have an odd number of 1's
- Similarly, for more than 2 inputs, XNOR is called an *even function*: it is equal to 1 if the input variables have an even number of 1's

McMaster University

Learning Objectives

- List the three basic logic operations
- Draw the truth table for the basic logic operations
- Build truth table for an arbitrary number of variables
- Draw schematic for basic logic gates
- Perform analysis on simple logic circuits
- Draw timing diagram for simple logic circuits