Lecture 5 – Boolean Algebra and Logic Gates– Part I

Rami Abielmona
University of Ottawa
January 27, 2015
ITI 1100 B
Digital Systems I

Presentation Outline

- Boolean Expressions
- Truth Table
- Boolean Identities
- Minterms
- Simplification using Boolean Algebra
- Logic Gates
- Key terms

Boolean expressions (functions)

- **We can use the basic operations to form more complex expressions:**

$$f(x,y,z) = x\,y' + z\,x'$$

- **Some terminology and notation:**
  - **f** is the name of the function.
  - ***Term*** is an implementation with a gate: in this example f has two terms $x\,y'$ and $z\,x$
  - ***(x,y,z)*** are the ***input variables***, each representing 1 or 0.
  - *A **literal*** is any occurrence of an input variable or its complement. The function above has four literals: x, y', z, and x'.

Precedence for Evaluation of Boolean Expression

- Precedence are important.

  – Parentheses first (if any) then

  NOT has the highest precedence, followed by AND, and then OR.

  $\rightarrow f(x,y,z) = (x + y')z + x'$

  –Fully parenthesized, the function above would be kind of messy:

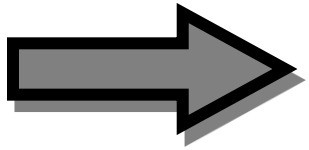
  $$f(x,y,z) = (((x +(y'))z) + x')$$

Truth Table

- A truth table shows all possible inputs and outputs of a function. Each input variable represents either 1 or 0.
  - A function with n variables has 2 power n possible combinations of inputs.
- Inputs are listed in binary order-example, from 000 to 111.

f(x,y,z) = (x + y')z + x'

f(0,0,0) = (0 + 1)0 + 1 = 1
f(0,0,1) = (0 + 1)1 + 1 = 1
f(0,1,0) = (0 + 0)0 + 1 = 1
f(0,1,1) = (0 + 0)1 + 1 = 1
f(1,0,0) = (1 + 1)0 + 0 = 0
f(1,0,1) = (1 + 1)1 + 0 = 1
f(1,1,0) = (1 + 0)0 + 0 = 0
f(1,1,1) = (1 + 0)1 + 0 = 1



x	y	z	f(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

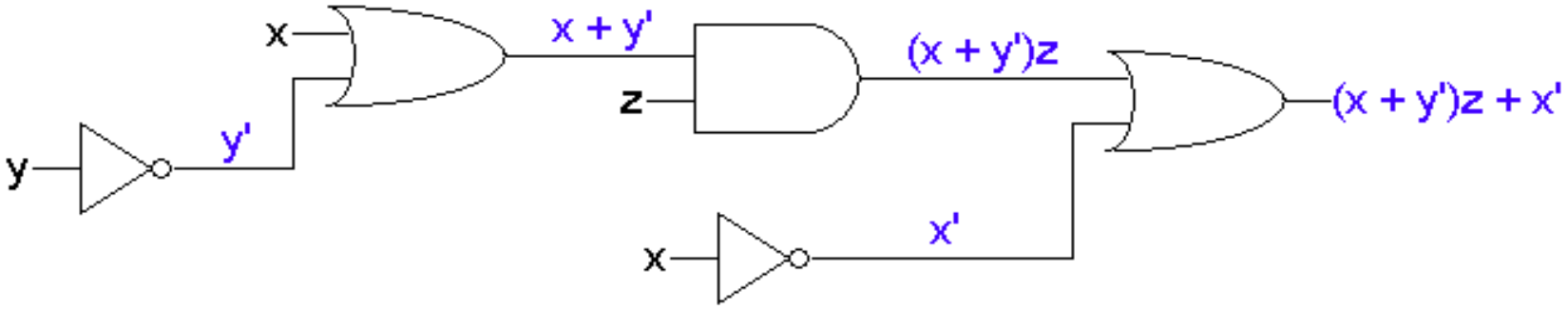
Boolean Expression and Logic Circuits

- *A* Boolean expression (function) can be converted into a circuit by *combining* basic gates.
- Example:

  - The diagram below shows the inputs and outputs of each gate.

  - The precedences are explicit in a circuit.

f(x,y,z) = (x + y')z + x'



Obtaining Boolean Expressions

- **Expressions may be obtained from:**
  - English language description
  - Truth table;
  - Logic circuit.

Obtaining Boolean Expressions

- The Boolean expression (un-simplified) can be obtained from the truth table: Consider the following arbitrary Truth Table

A	B	C	F_1	
0	0	0	0	
0	0	1	0	
0	1	0	1	*A'BC'*
0	1	1	1	*A'BC*
1	0	0	1	*AB'C'*
1	0	1	1	*AB'C*
1	1	0	1	*ABC'*
1	1	1	1	*ABC*

We can also write the function as:

$F_1(A,B,C) =$ A'BC'+ A'BC+ AB'C'+ AB'C+ ABC'+ ABC

Boolean Expressions (2 var)

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Minterms

ABC	m_0= ABC	m_1 = ABC	m_2 = ABC	m_3 = ABC	m_4 = ABC	m_5 = ABC	m_6 = ABC	m_7 = ABC
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	0	0	0	0	1

Obtaining Boolean Expressions

Using the false terms in the truth table

•Sometimes it is easier to work with the terms that describe when the function is false.

•For example, if a function has four variables then there are sixteen possible states.

→If for instance thirteen out of sixteen were true, then only three out of sixteen are false. Fewer terms makes it easier

• In our example, two out of eight are false.

Obtaining Boolean Expressions

- The Boolean expression (un-simplified) can be obtained from the truth table using false terms

A	B	C	F_1	
0	0	0	0	$A'B'C'$
0	0	1	0	$A'B'C$
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	

So we can also write the function **NOT** F_1 as:

$$\overline{F_1}(A,B,C) = A'B'C' + A'B'C$$

Obtaining The Truth Table - Example

Design a digital circuit that will be used to control an Alarm bell. This Alarm bell is to be installed in a room to protect it from unauthorized entry.

Sensor devices provide the following logic signals

C = 1 The control system is active

D = 1 The room door is closed

M = 1 There is a motion in the room

Q = 1 The room is open to the public

i) Obtain the truth table

ii) derive the Boolean expression using **true** terms

Truth table

C	D	M	Q	Alarm
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Alarm= CD'M'Q' + CD'MQ' + CDMQ'

Door should not be open

room is closed to the public → door open + motion

room is closed to the public + door closed → motion

Boolean Identities

• Boolean algebra is used in digital design to reduce any logical function (expression) to its simplest form

→ the minimization of the number of literals and the number of terms

→ a circuit with less equipment

• It is a hard problem (no specific rules to follow)

Boolean Identities

1.	$x + 0 = x$	
2.	$x \cdot 1 = x$	
3.	$x + x = 1$	
4.	$x \cdot x = 0$	
5.	$x + x = x$	Basic to Boolean algebra
6.	$x \cdot x = x$	
7.	$x + 1 = 1$	
8.	$x \cdot 0 = 0$	
9.	$(x')' = x$	
10.	$x + y = y + x$	Commutative
11.	$xy = yx$	Commutative
12.	$x + (y + z) = (x + y) + z$	Associative
13.	$x(yz) = (xy)z$	Associative
14.	$x(y + z) = xy + xz$	Distributive
15.	$x + yz = (x+y)(x+z)$	Associative
16.	$(x + y)' = x' y'$	DeMorgan
17.	$(xy)' = x' + y'$	DeMorgan
18.	$x + xy = x$	Absorption
19.	$x(x + y) = x$	Absorption

Verifying Boolean Identities-Examples

Theorem : $x+x = x$

$x+x = (x+x)\ 1$ — x.1=x

$= (x+x)\ (x+x')$ — x+x’=1

$= x+xx'$ — $x+yz = (x+y)(x+z)$

$= x+0$ — x.x’=0

$= x$ — x+0=x

Theorem : x x = x

$xx = x\ x + 0$

$= xx + xx'$

$= x\ (x + x')$

$= x\ 1$

$= x$

Verifying Boolean Identities-Examples

- DeMorgan's Theorems
  - → $(x+y)' = x'\,y'$
  - → $(x\,y)' = x' + y'$
- By means of truth table

x	y	x+y	(x+y)‘	x‘	y‘	x‘y’
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Verifying Boolean Identities-Examples

- Theorem $x + xy = x$

By means of truth table

x	y	xy	x + xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Simplifying Boolean Expressions

→ **Use the Rules of Boolean Algebra**

We can simply the function as:

$F_1(A,B,C) = A'BC' + A'BC + AB'C' + AB'C + ABC' + ABC$

$= A'B(C'+C) + AB'(C+C') + AB(C'+C)$

$= A'B + AB' + AB$

$= A'B + A(B'+B)$

$= A + A'B$

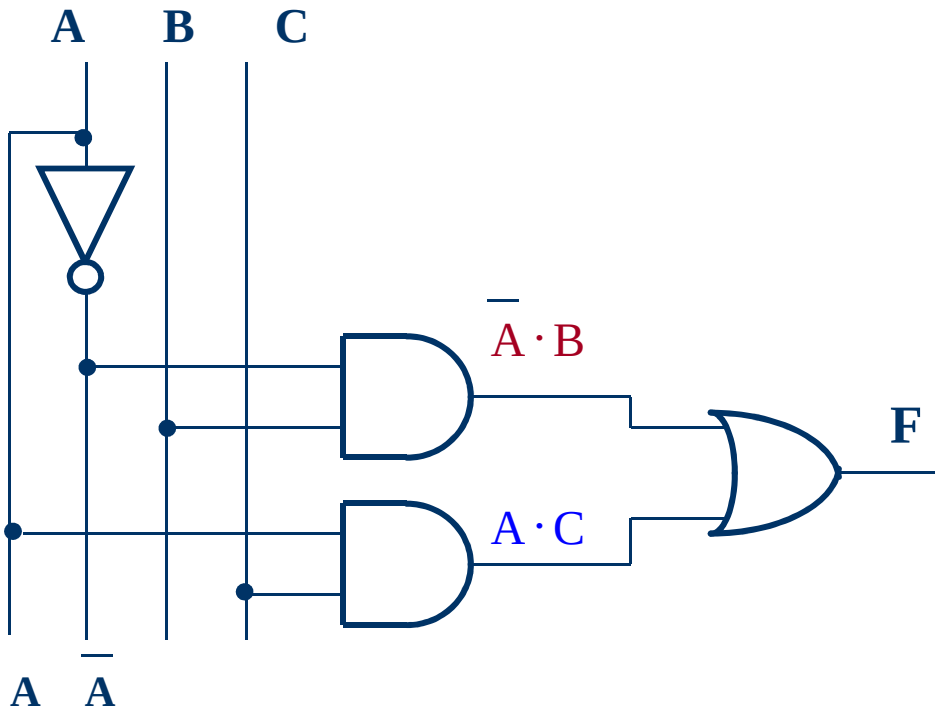
Simplification using Boolean Algebra Rules

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

$$F = (\overline{A}B\overline{C} + \overline{A}BC) + (A\overline{B}C + ABC)$$

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

$$F = \overline{A}B(\underbrace{\overline{C} + C}_{1}) + AC(\underbrace{\overline{B} + B}_{1})$$



$$F = \overline{A}B + AC$$

Simplifying Boolean Expressions

Function with four variables

- Giving the following function:

$F_{2a}(A,B,C,D) = (AB'(C + BD) + A'B')C$

$= (AB'C + \mathbf{AB'BD} + A'B')C$

$= (AB'C + \mathbf{A0D} + A'B')C$

$= (AB'C + \mathbf{0} + A'B')C$

$= (AB'C + A'B')C$

$= AB'\mathbf{CC} + A'B'C$

$= AB'C + A'B'C$

$= (\mathbf{A+A'})B'C$

$= B'C$

$\boldsymbol{F_{2b}(A,B,C,D)} = B'C$

→ the two expressions are equivalent!

→ **F_{2a} requires more logic gates than F_{2b}**

Basic and Other Logic gates

- **Basic Logic gate**

  - AND
  - OR
  - NOT

  These are called “fundamental logic gates” as all other gates and digital Circuits can be created from these gates.

- **Other Logic gates**

  - NAND
  - NOR

  These are called “Universal logic gates” as any digital circuit can be designed by just using these gates

  - XOR
  - XNOR

The NAND & NOR Gates

• We can use a NAND and NOR gates to implement all three of the *basic operations* (AND,OR,NOT).

→ They are said to be **functionally complete**

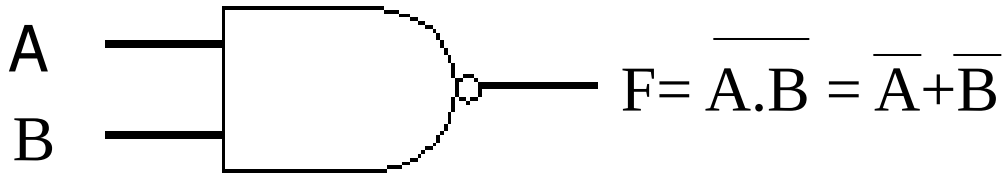
→ Both NAND and NOR gates are very valuable as any design can be realized using either one.

•It is easier to build digital circuits using all NAND or NOR gates than to combine AND,OR, and NOT gates.

•NAND/NOR gates are typically faster and cheaper to produce.

The NAND Gate

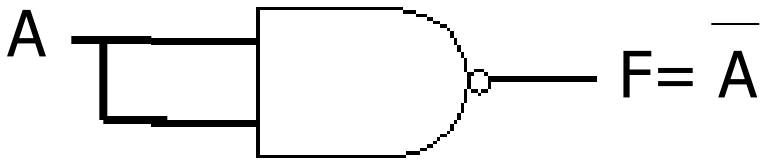
- The NAND gate is a combination of an AND gate followed by an inverter (NOT gate).
- We can use a NAND gate to implement all three of the *basic operations* (AND,OR,NOT).
- Such a gate is said to be **functionally complete**.



A	**B**	**F**
0	0	1
0	1	1
1	0	1
1	1	0

The NAND Gate

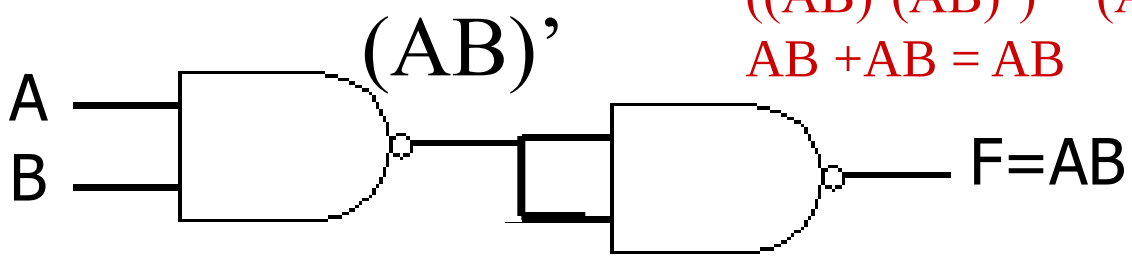
→ a NAND gate with both of its inputs driven by the same signal is equivalent to a NOT gate



NOT Gate

→a NAND gate whose output is complemented is equivalent to an AND gate

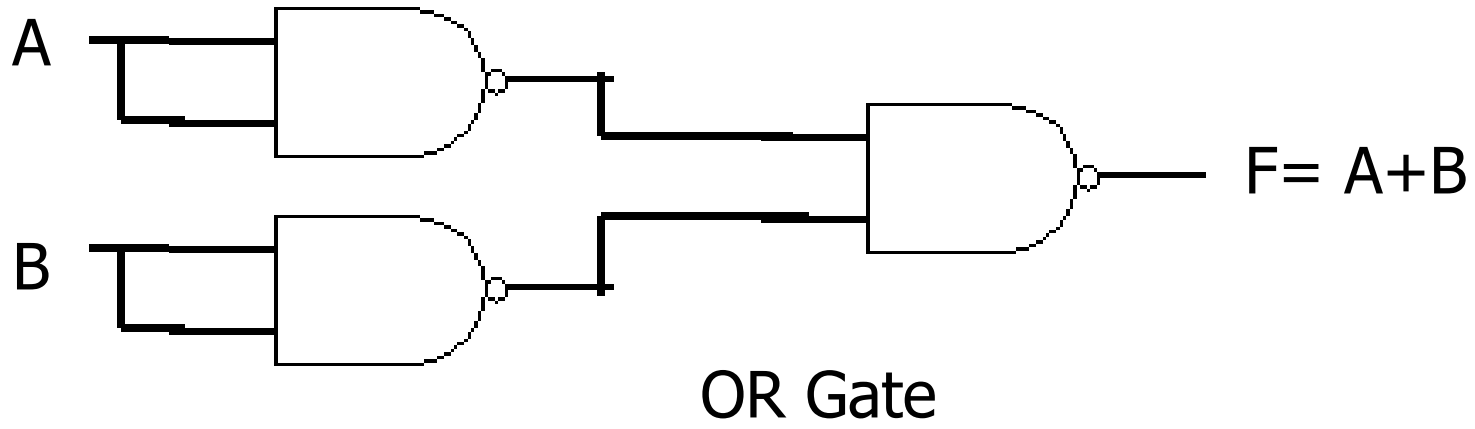
((AB)'(AB)')'= (AB)''+(AB)''=
AB +AB = AB



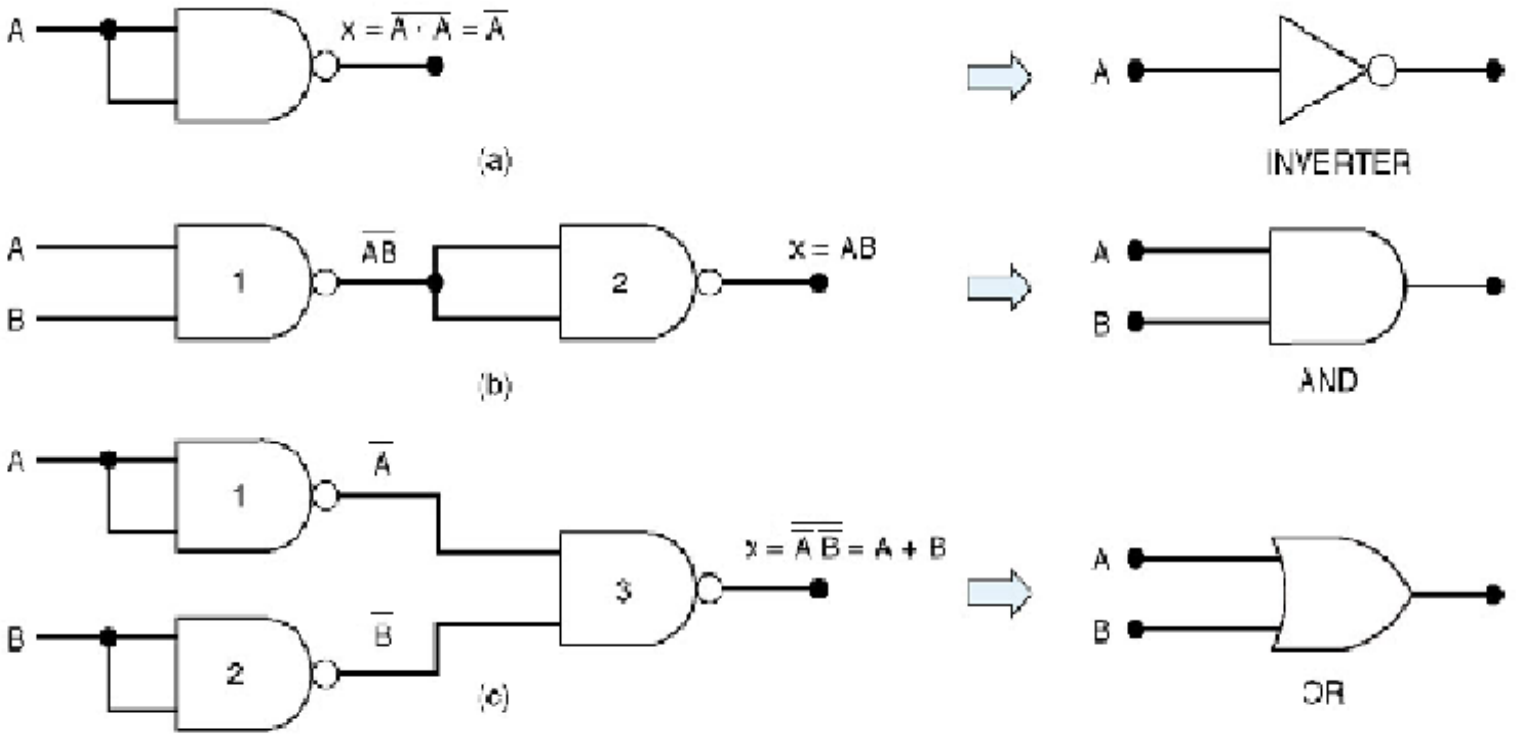
AND Gate

The NAND Gate

•a NAND gate with complemented inputs acts as an OR gate.



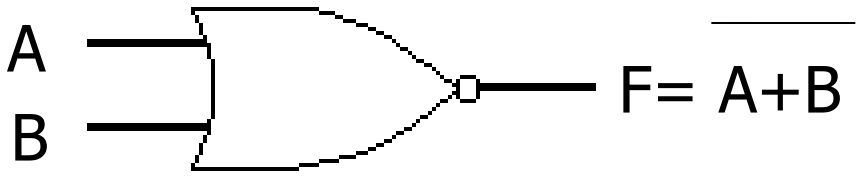
Universality of NAND



The NOR Gate

•This is a NOR gate. It is a combination of an OR gate followed by an inverter.

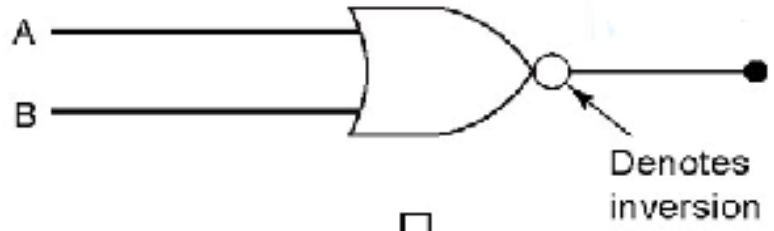
•like the NAND gate, the NOR gate is **functionally complete** →any logic function can be implemented using just NOR gates.



A	**B**	**F**
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate Equivalence

- NOR Symbol, Equivalent Circuit, Truth Table



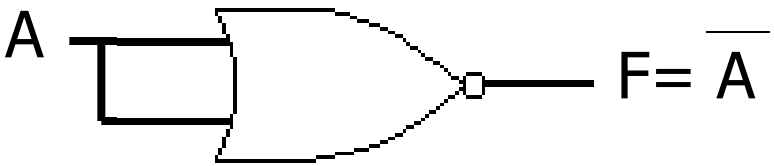
(a)

(b)

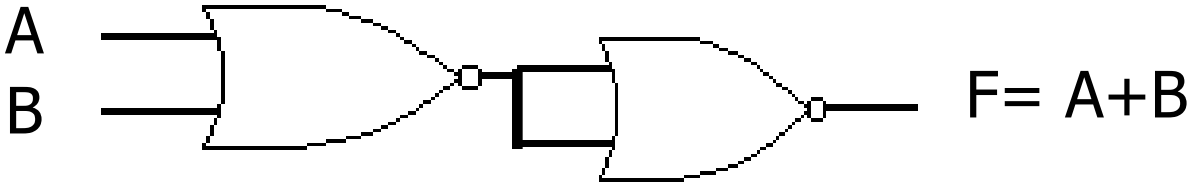
		OR	NOR
A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(c)

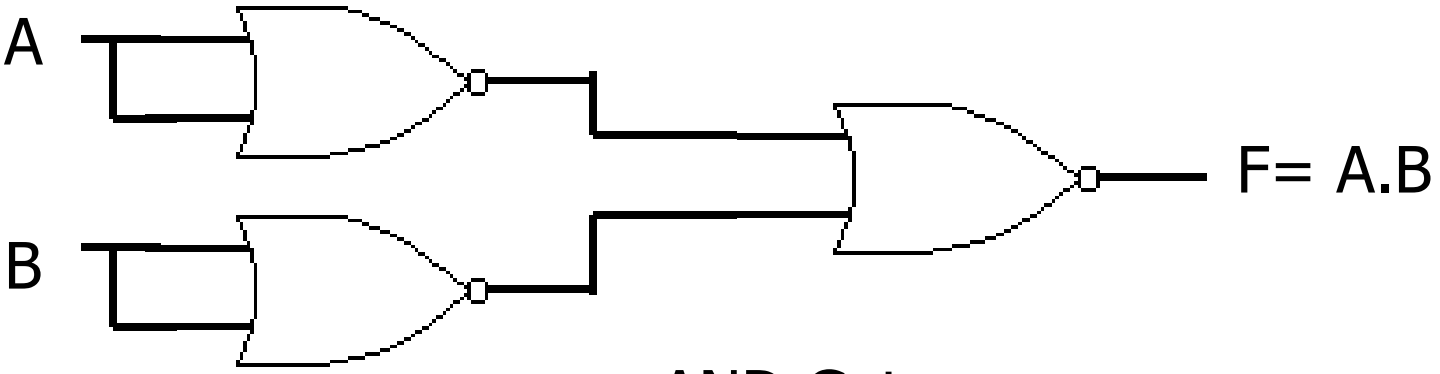
NOR Gates-**functionally complete**



NOT Gate

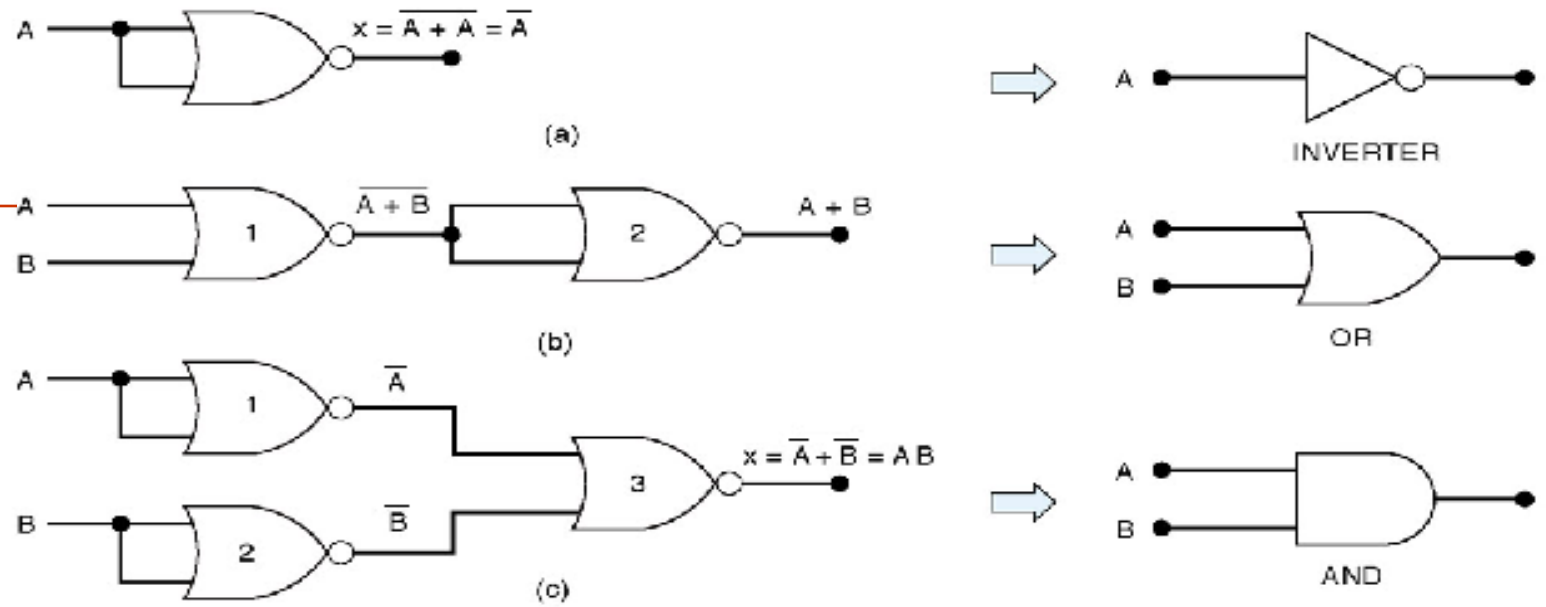


OR Gate



AND Gate

Universality of NOR gate



- Equivalent representations of the AND, OR, and NOT gates

(A+A)' = A'A' = A'

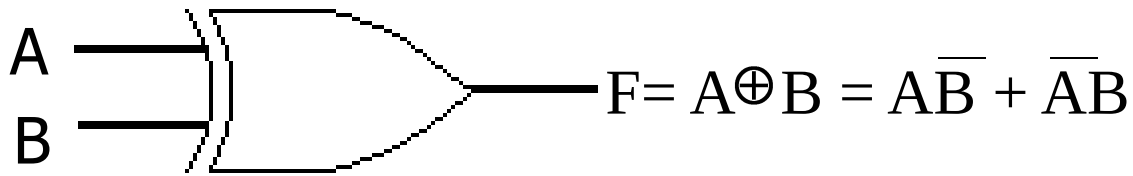
((A+B)' + (A + B)')' = (A+B)"(A+B)" = (A'B')' (A'B')'

= (A"+B") (A"+B") = (A+B) (A+B) = (A+B)

The XOR Gate (Exclusive-OR)

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

  $1 \oplus 1 = 0$ and $1 \oplus 0 = 1$.

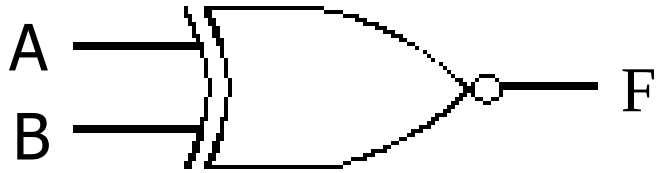


A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

The XNOR Gate

- This functions as an exclusive-NOR gate, or simply the complement of the XOR gate.
- The symbol for this operation is ⊙

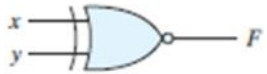
  $1 \odot 1 = 1$ and $1 \odot 0 = 0$.



$$F = \overline{A \oplus B} = (AB) + (\overline{A} \cdot \overline{B}) = AB + A'B'$$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Logic Gates

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$
OR		$F = x + y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 1$
Inverter		$F = x'$	$x \mid F$ $0 \mid 1$ $1 \mid 0$
Buffer		$F = x$	$x \mid F$ $0 \mid 0$ $1 \mid 1$
NAND		$F = (xy)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
NOR		$F = (x + y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 0$
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$

Key Terms and Review Points

- Absorption Boolean Identity
- Associative Boolean Identity
- Commutative Boolean Identity
- DeMorgan's Theorems
- Distributive Boolean Identity
- Functionally Complete Gates
- Identity Boolean Expression
- Implication Boolean Expression
- Inhibition Boolean Expression
- Literal
- Minterm
- Term
- Truth Table
- Universal Gates

- References: Dr. Karmouch and Dr. Groza ITI 1100 Slides