BME2322 – Logic Design

The Instructors:

Dr. Görkem SERBES (C317)

gserbes@yildiz.edu.tr

https://avesis.yildiz.edu.tr/gserbes/

Lab Assistants:

Nihat AKKAN

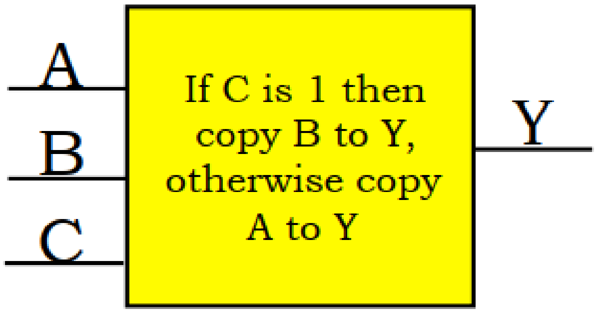
nakkan@yildiz.edu.tr

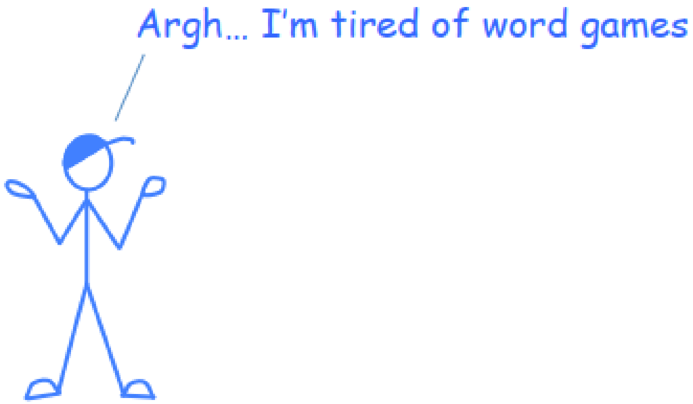
https://avesis.yildiz.edu.tr/nakkan

LECTURE 4

Functional Specifications

There are many ways of specifying the function of a combinational device, for example:

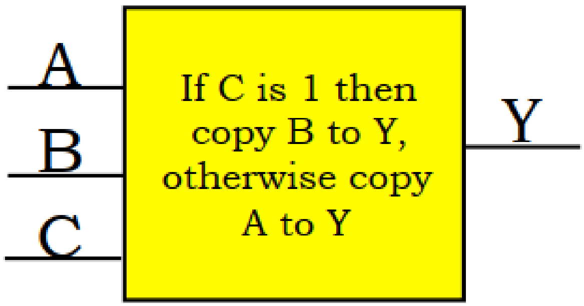


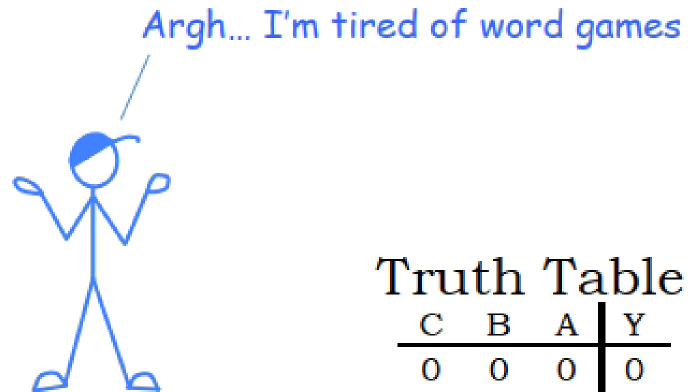


unless the words are very carefully crafted, there may be ambiguities introduced by words with multiple interpretations or by lack of completeness

Functional Specifications

There are many ways of specifying the function of a combinational device, for example:





Concise alternatives:

- *truth tables* are a concise description of the combinational system's function.
- *Boolean expressions* form an algebra whose operations are AND (multiplication), OR (addition), and inversion (overbar).

Truth Table

C	B	A	Y
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

$$Y = \bar{C} \cdot \bar{B} \cdot A + \bar{C}BA + CB\bar{A} + CBA$$

Any combinational (Boolean) function can be specified as a truth table or an equivalent sum-of-products Boolean expression!

Boolean Algebra

A **Boolean algebra** ***B*** is a finite set over which two binary operations + (sum) and · (product) and satisfy five postulates.

Boolean Algebra Postulates

P 1 - Operations + and · are internal: $\forall a,b \in B, \quad a+b \in B \ y \ a \cdot b \in B$

P 2 – To each operation corresponds a **neutral element**: $\forall a \in B, \quad a+0=a, \quad a \cdot 1 = a$

P 3 – To each element corresponds an **inverse element**: $\forall a \in B, \exists \ \overline{a} \in B \mid a+\overline{a}=1, \quad a \cdot \overline{a} = 0$

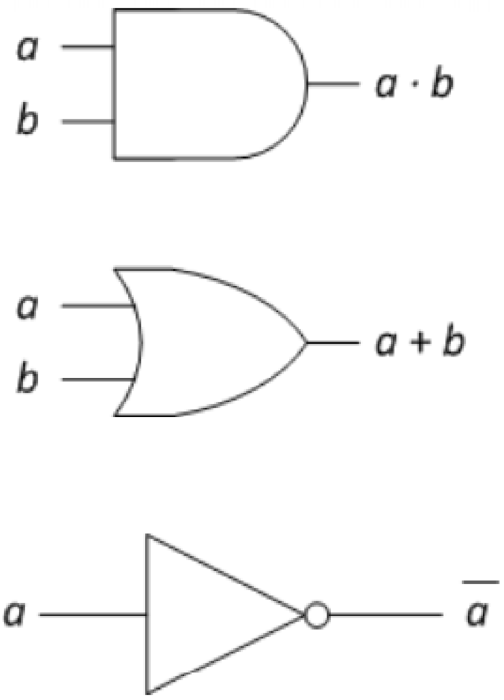
P 4 – Operations + and · are **commutative**: $a+b=b+a, \quad a \cdot b = b \cdot a$

P 5 –Operations + and · are **distributive:** $a \cdot (b+c) = a \cdot b + a \cdot c, \quad a+b \cdot c = (a+b) \cdot (a+c)$

Boolean Algebra

The set {0, 1} is a Boolean algebra if the operations are defined as follows:

$a\ b$	$a \cdot b$	$a + b$	$\overline{a}$
0 0	0	0	1
0 1	0	1	1
1 0	0	1	0
1 1	1	1	0



Boolean Algebra (distributive rule)

Example: check that $a \cdot (b+c) = a \cdot b + a \cdot c$

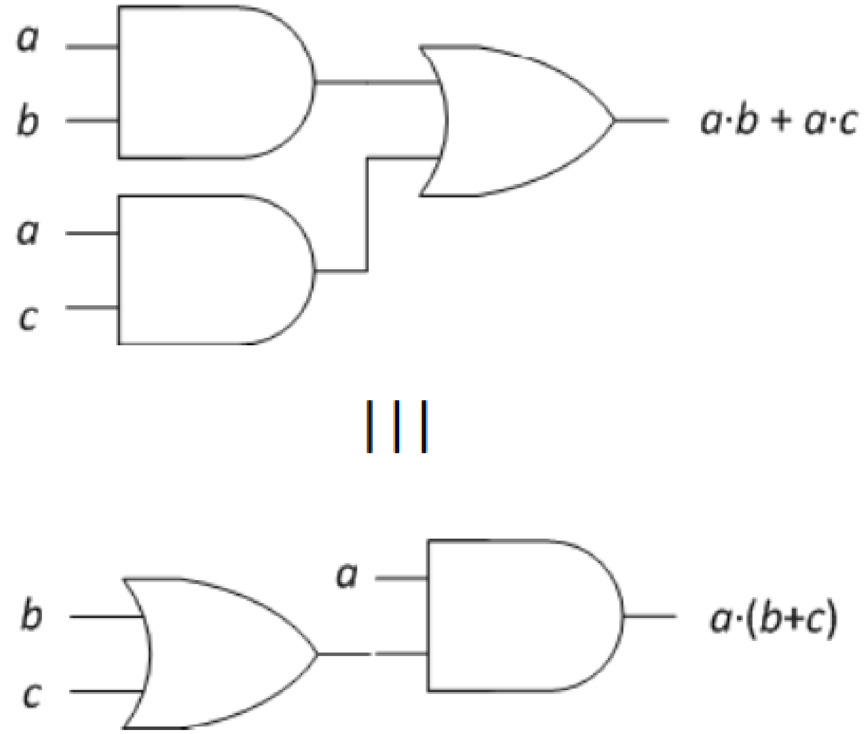
a b	$a \cdot b$	$a + b$	$\overline{a}$
0 0	0	0	1
0 1	0	1	1
1 0	0	1	0
1 1	1	1	0

a b c	$b+c$	$a \cdot (b+c)$	$a \cdot b$	$a \cdot c$	$a \cdot b + a \cdot c$
0 0 0	0	0	0	0	0
0 0 1	1	0	0	0	0
0 1 0	1	0	0	0	0
0 1 1	1	0	0	0	0
1 0 0	0	0	0	0	0
1 0 1	1	1	0	1	1
1 1 0	1	1	1	0	1
1 1 1	1	1	1	1	1

Boolean Algebra (distributive rule)

Comment:

$a \cdot (b+c) = a \cdot b + a \cdot c$ =>



Some useful properties

1 – Neutral element properties: $\bar{0} = 1, \quad \bar{1} = 0$

2 – Idempotence: $a + a = a, \quad a \cdot a = a$

$a = a + 0 = a + (a \cdot \bar{a}) = (a + a) \cdot (a + \bar{a}) =$
$(a + a) \cdot 1 = a + a$

P1 - $\forall a, b \in B, \quad a + b \in B \text{ y } a \cdot b \in B$

P2 - $\forall a \in B, \quad a + 0 = a, \quad a \cdot 1 = a$

P3 - $\forall a \in B, \exists \bar{a} \in B \mid a + \bar{a} = 1, \quad a \cdot \bar{a} = 0$

P4 - $a + b = b + a, \quad a \cdot b = b \cdot a$

P5 - $a \cdot (b + c) = a \cdot b + a \cdot c, \quad a + b \cdot c = (a + b) \cdot (a + c)$

Some useful properties - Exercise

Demonstrate that $a \cdot a = a$

Hint: Use the second part of P2, P3 and P5.

P1 - $\forall a, b \in B, \quad a + b \in B \; y \; a \cdot b \in B$

P2 - $\forall a \in B, \quad a + 0 = a, \quad a \cdot 1 = a$

P3 - $\forall a \in B, \exists \overline{a} \in B \mid a + \overline{a} = 1, \quad a \cdot \overline{a} = 0$

P4 - $a + b = b + a, \quad a \cdot b = b \cdot a$

P5 - $a \cdot (b + c) = a \cdot b + a \cdot c, \quad a + b \cdot c = (a + b) \cdot (a + c)$

Some useful properties - Exercise

Demonstrate that $a \cdot a = a$

Hint: Use the second part of P2, P3 and P5.

$a = a \cdot 1 = a \cdot (a + \overline{a}) = (a \cdot a) + (a \cdot \overline{a}) =$
$(a \cdot a) + 0 = a \cdot a$

$a = a + 0 = a + (a \cdot \overline{a}) = (a + a) \cdot (a + \overline{a}) =$
$(a + a) \cdot 1 = a + a$

P1 - $\forall a, b \in B, \quad a + b \in B \ y \ a \cdot b \in B$

P2 - $\forall a \in B, \quad a + 0 = a, \quad a \cdot 1 = a$

P3 - $\forall a \in B, \exists \overline{a} \in B \mid a + \overline{a} = 1, \quad a \cdot \overline{a} = 0$

P4 - $a + b = b + a, \quad a \cdot b = b \cdot a$

P5 - $a \cdot (b + c) = a \cdot b + a \cdot c, \quad a + b \cdot c = (a + b) \cdot (a + c)$

Some useful properties

1 – Neutral element properties: $\overline{0} = 1, \quad \overline{1} = 0$

2 – Idempotence: $a + a = a, \quad a \cdot a = a$

3 – Involution: $\overline{\overline{a}} = a$

4 – Asociativity: $a + (b + c) = (a + b) + c, \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c$

5 – Absortion law: $a + a \cdot b = a, \quad a \cdot (a + b) = a$

6 - (nameless): $a + \overline{a} \cdot b = a + b, \quad a \cdot (\overline{a} + b) = a \cdot b$

7 - de Morgan law: $(\overline{a + b}) = \overline{a} \cdot \overline{b}, \quad \overline{a \cdot b} = \overline{a} + \overline{b}$

8 – generalized de Morgan law: $(\overline{a_1 + a_2 + \ldots + a_n}) = \overline{a_1} \cdot \overline{a_2} \cdot \ldots \cdot \overline{a_n}, \quad \overline{a_1 \cdot a_2 \cdot \ldots \cdot a_n} = \overline{a_1} + \overline{a_2} + \ldots + \overline{a_n}$

Simplifying Boolean Equations 1

$Y = A(AB + ABC)$

$= A(AB(1 + C))$	Distributivity
$= A(AB(1))$	Null Element
$= A(AB)$	Identity
$= (AA)B$	Associativity
$= AB$	Idempotency

Simplifying Boolean Equations 2

$$Y = \overline{(A+\overline{BD})\overline{C}}$$

Simplifying Boolean Equations 3

$$Y = \overline{(\overline{AC}E+\overline{D}) + B}$$

Boolean functions and truth tables

Any Boolean function can be explicitely defined by a truth table

$f(a,b,c) = b.\overline{c} + \overline{a}.b$

$a\ b\ c$	$\overline{c}$	$b \cdot \overline{c}$	$\overline{a}$	$\overline{a} \cdot b$	f
0 0 0	1	0	1	0	0
0 0 1	0	0	1	0	0
0 1 0	1	1	1	1	1
0 1 1	0	0	1	1	1
1 0 0	1	0	0	0	0
1 0 1	0	0	0	0	0
1 1 0	1	1	0	0	1
1 1 1	0	0	0	0	0

Boolean functions and truth tables

Given a truth table can we find an equivalent Boolean function?...

Answer is YES

LITERAL

A variable or an inverted variable : $a, \overline{a}, b, \overline{b}, c, \overline{c}, \ldots$

n-variable **MINTERM**

A product of *n* literals such that each variable appears only once. Example: if *n*=3, there are eight *minterms*.

$$a.b.c,\ a.b.\overline{c},\ a.\overline{b}.c,\ a.\overline{b}.\overline{c},\ \overline{a}.b.c,\ \overline{a}.b.\overline{c},\ \overline{a}.\overline{b}.c,\ \overline{a}.\overline{b}.\overline{c}$$

Boolean functions and truth tables

Given a **MINTERM** *m*, there is one, an only one, set of variable values such that *m* = 1.

With *n* = 3:

		a	b	c	
$\overline{a}.\overline{b}.\overline{c} = 1$	$\Leftrightarrow$	0	0	0	$\rightarrow m_0 = \overline{a}.\overline{b}.\overline{c}$
$\overline{a}.\overline{b}.c = 1$	$\Leftrightarrow$	0	0	1	$\rightarrow m_1 = \overline{a}.\overline{b}.c$
$\overline{a}.b.\overline{c} = 1$	$\Leftrightarrow$	0	1	0	$\rightarrow m_2 = a.\overline{b}.c$
$\overline{a}.b.c = 1$	$\Leftrightarrow$	0	1	1	$\rightarrow m_3 = \overline{a}.b.c$
$a.\overline{b}.\overline{c} = 1$	$\Leftrightarrow$	1	0	0	$\rightarrow m_4 = a.\overline{b}.\overline{c}$
$a.\overline{b}.c = 1$	$\Leftrightarrow$	1	0	1	$\rightarrow m_5 = a.\overline{b}.c$
$a.b.\overline{c} = 1$	$\Leftrightarrow$	1	1	0	$\rightarrow m_6 = a.b.\overline{c}$
$a.b.c = 1$	$\Leftrightarrow$	1	1	1	$\rightarrow m_7 = a.b.c$

From Truth Table to Boolean Function

MINTERMS of an *n*-variable Boolean function *f* ?

= *minterms* that correspond to the 1s of *f*.

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

From Truth Table to Boolean Function

Canonical sum of products **representation** of an *n*-variable Boolean function.

Any Boolean function can be represented by the sum of its *minterm.*

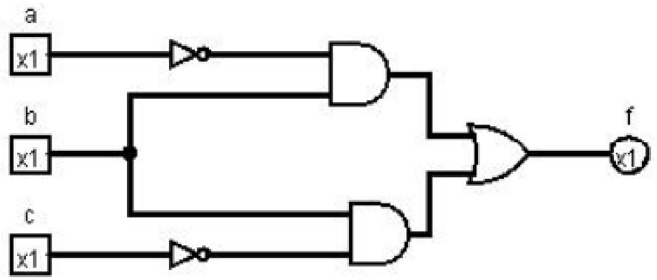
$$f(a,b,c) = \Sigma(m_2, m_3, m_6)$$

$$f(a,b,c) = \overline{a}.b.\overline{c} + \overline{a}.b.c + a.b.\overline{c}$$

a	b	c	$f(a,b,c)$	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\rightarrow m_2 = \overline{a}.b.\overline{c}$
0	1	1	1	$\rightarrow m_3 = \overline{a}.b.c$
1	0	0	0	
1	0	1	0	
1	1	0	1	$\rightarrow m_6 = a.b.\overline{c}$
1	1	1	0	

From Truth Table to Boolean Function

```
if ((a=1 and b=1 and c=0) or (a=0 and b=1)) then f=1;
                else f=0;
end if;
```

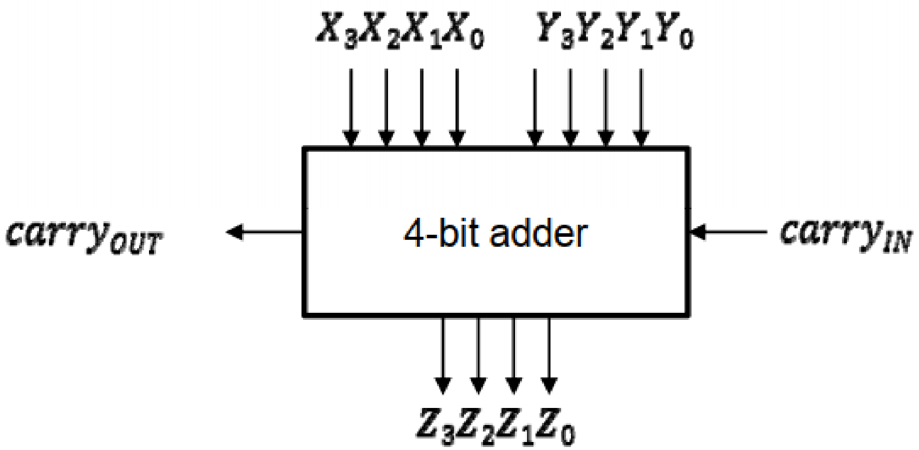


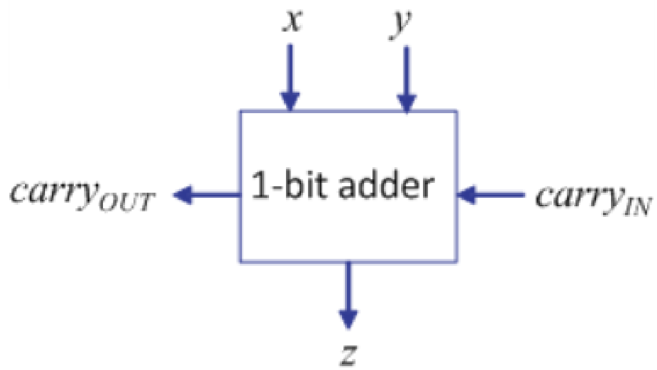
a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

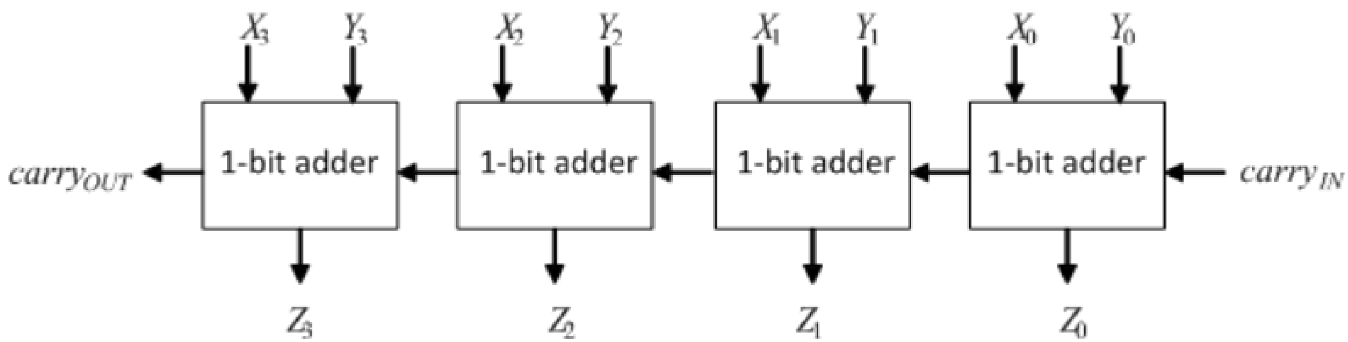
$$f(a,b,c) = \bar{a}.b.\bar{c} + \bar{a}.b.c + a.b.\bar{c} =$$
$$= \bar{a}.b(\bar{c} + c) + b.\bar{c}.(\bar{a} + a) = \bar{a}.b + b.\bar{c}$$

$$f(a,b,c) = \Sigma(m_2, m_3, m_6)$$
$$f(a,b,c) = \bar{a}.b.\bar{c} + \bar{a}.b.c + a.b.\bar{c}$$

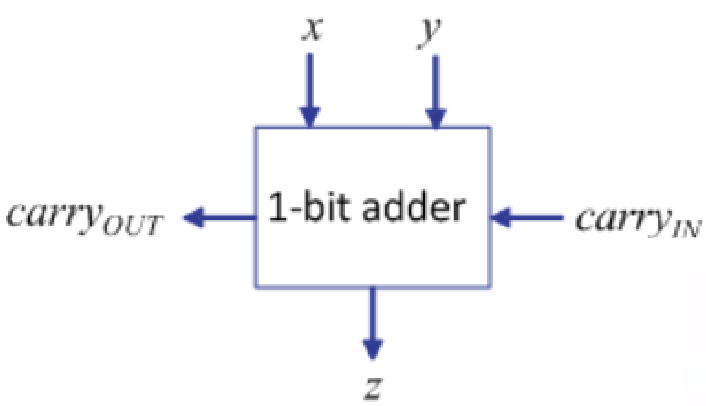
Example: 4 bit-adder







Example: 4 bit-adder



x	y	c_i	c_o	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$c_o = \bar{x} y c_i + x \bar{y} c_i + x y \bar{c_i} + x y c_i$$

$$\underbrace{(\bar{x} + x) y c_i}$$

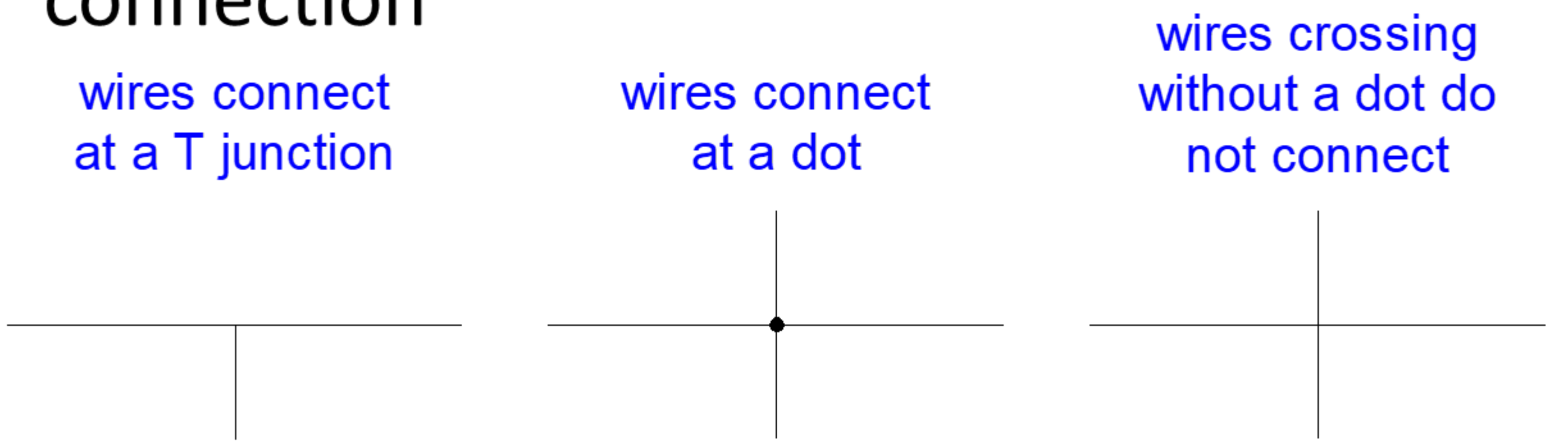
$$y c_i + x c_i + x y$$

Circuit Schematics Rules

- Inputs on the left (or top)
- Outputs on right (or bottom)
- Gates flow from left to right
- Straight wires are best

Circuit Schematics Rules (Cont.)

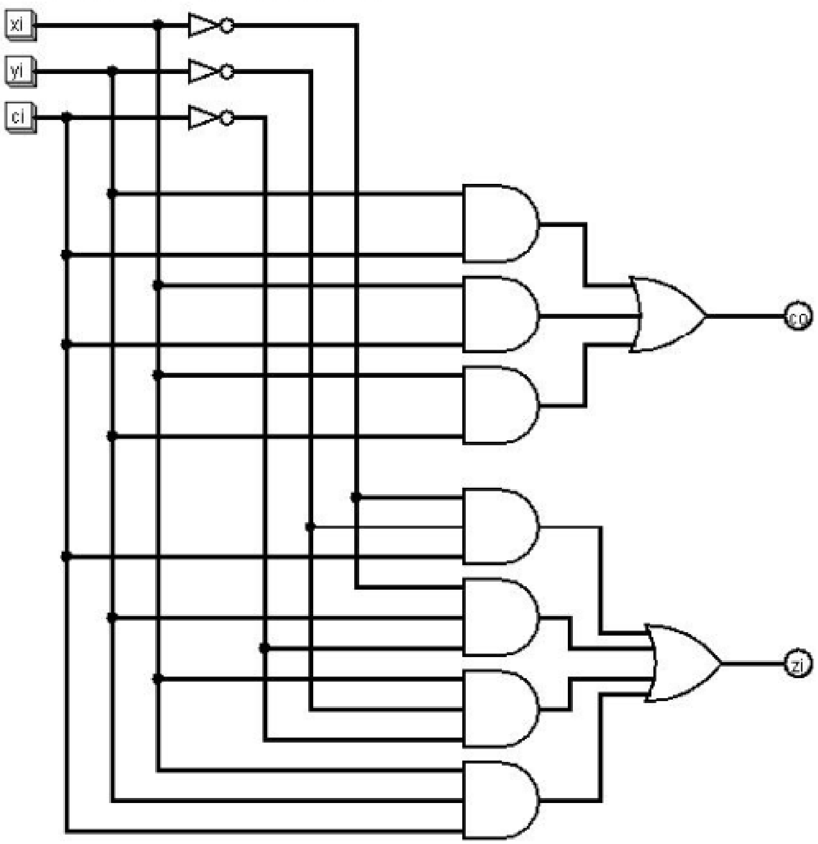
- Wires always connect at a T junction
- A dot where wires cross indicates a connection between the wires
- Wires crossing *without* a dot make no connection

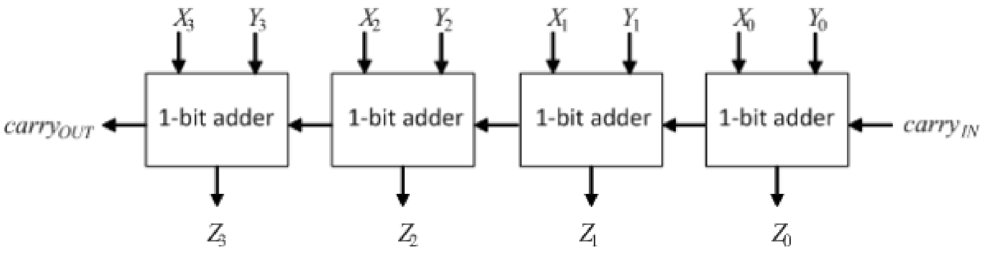


Example: 4 bit-adder

$$c_o = y.\,c_i + x.\,c_i + x.\,y$$

$$z = \bar{x}.\bar{y}.\,c_i + \bar{x}.\,y.\overline{c_i} + x.\bar{y}.\overline{c_i} + x.\,y.\,c_i$$

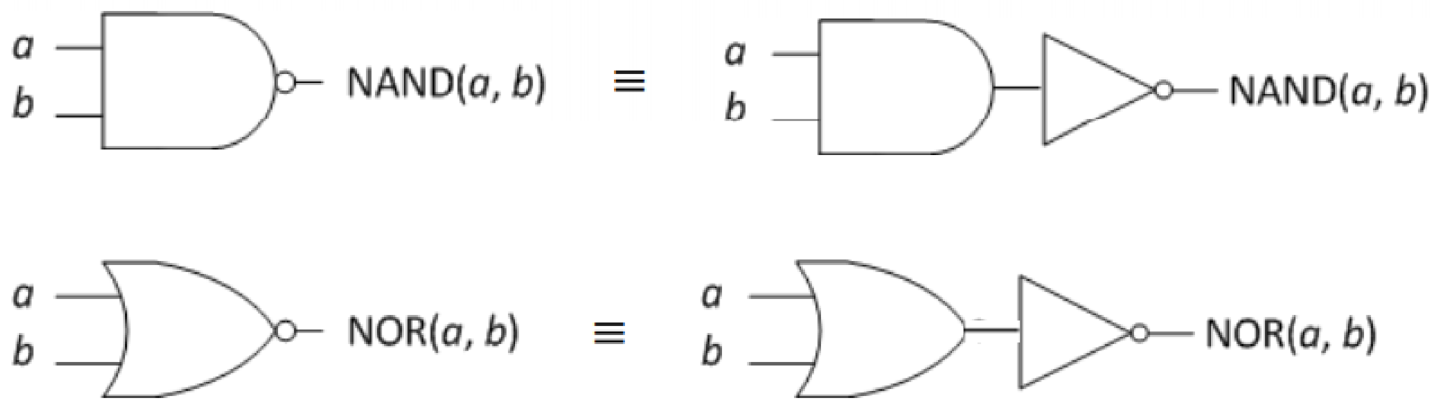




Circuit generation from a functional description:

(functional description → truth table → Boolean function(s) → circuit)

NAND – NOR Gates



a b	NAND(a,b)	NOR(a,b)
0 0	1	1
0 1	1	0
1 0	1	0
1 1	0	0

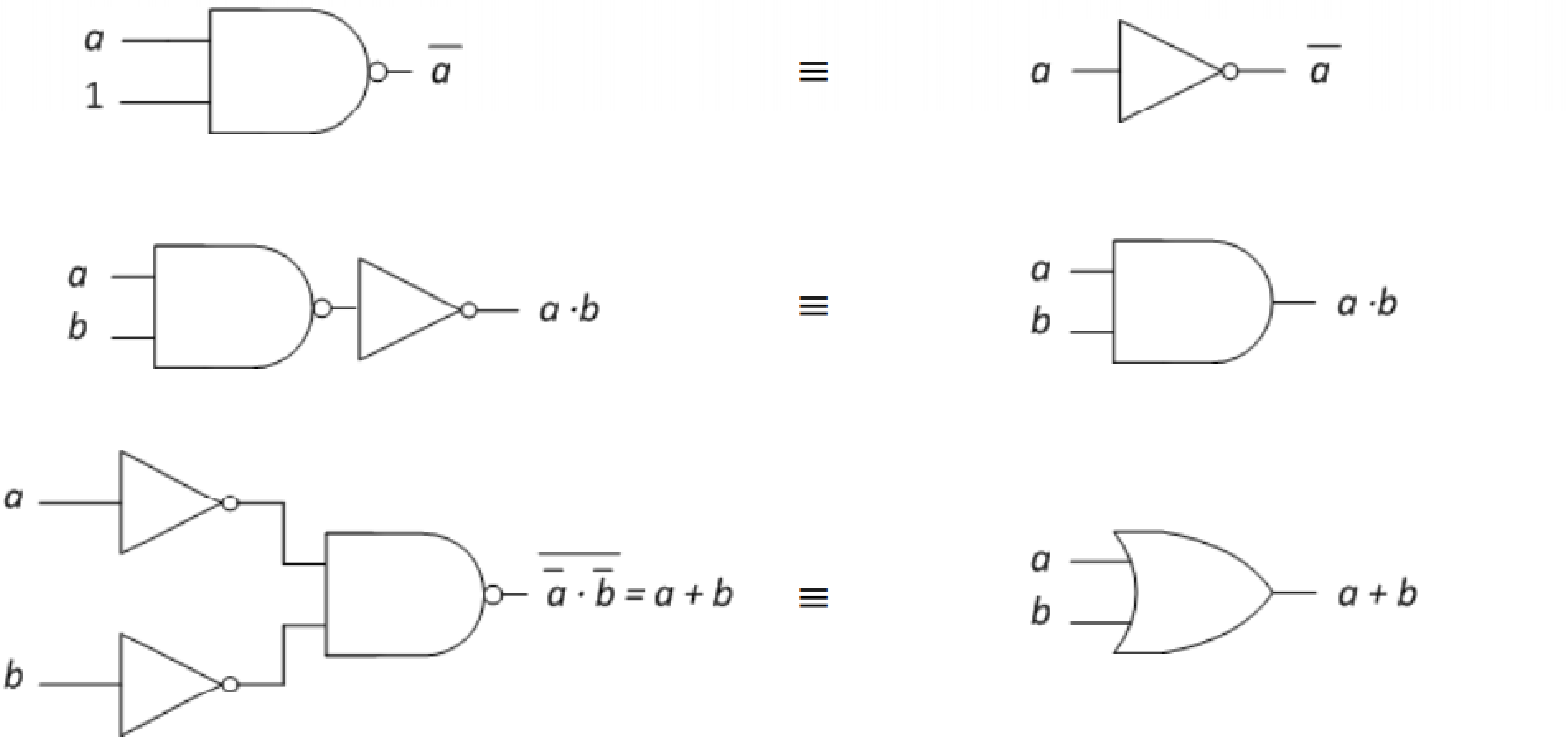
Algebraic symbols:

NAND(a, b) = $a \uparrow b$,

NOR(a, b) = $a \downarrow b$.

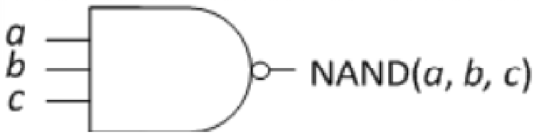
NAND – NOR Gates

NAND and NOR gates are **universal modules**. For example, with NAND gates:

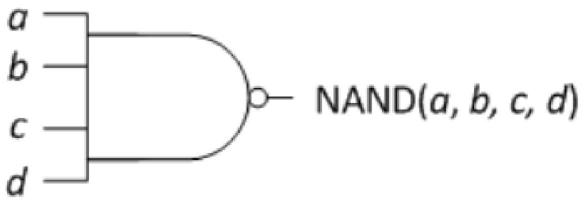


NAND – NOR Gates

3-input, 4-input, ⋯ NAND and NOR gates can be defined:

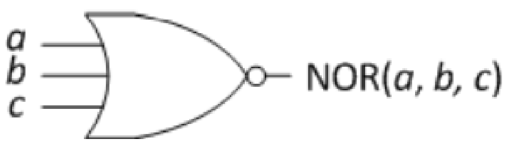


NAND(a, b, c) = 0 iff $a = b = c = 1$

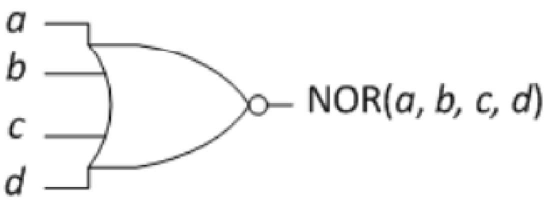


NAND(a, b, c, d) = 0 iff $a = b = c = d = 1$

....



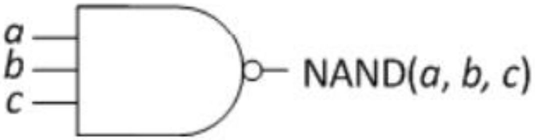
NOR(a, b, c) = 0 iff ($a = 1$) OR ($b = 1$) OR ($c = 1$)



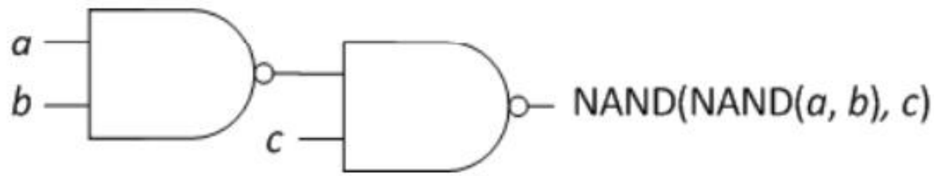
NOR(a, b, c, d) = 0 iff ($a = 1$) OR ($b = 1$) OR ($c = 1$) OR ($d = 1$)

NAND – NOR Gates

BUT NAND and NOR **are not associative operations**. In particular:

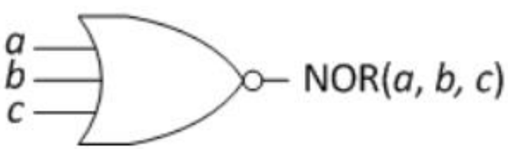


≠

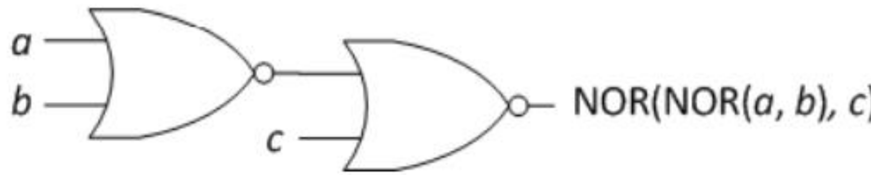


NAND(1, 1, 1) = 0

NAND(NAND(1, 1), 1) = NAND(0, 1) = 1



≠



NOR(0, 0, 0) = 1

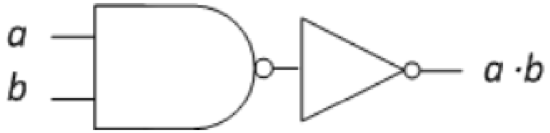
NOR(NOR(0, 0), 0) = NOR (1, 0) = 0

Why NAND (or NOR) Gates?

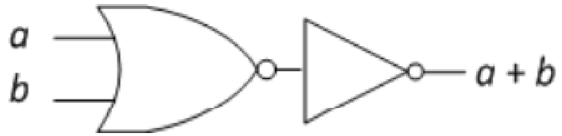
Why do we use NAND gates (or NOR gates) instead of AND and OR gates?

- If we use "of the shelf" components (laboratory) we only need one type of gate.
- In CMOS technology

  - an AND gate is implemented with a NAND and an INV,



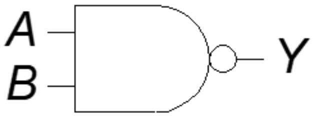
  - an OR gate is implemented with a NOR and an INV.

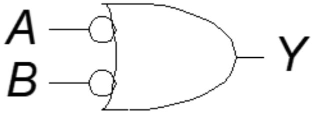


=> Within an IC (Integrated Circuit) NAND and NOR are "cheaper" than AND and OR.

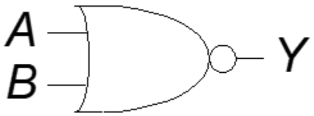
DeMorgan's Theorem

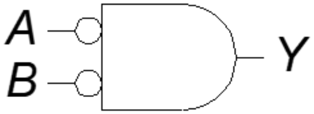
- $Y = \overline{AB} = \overline{A} + \overline{B}$





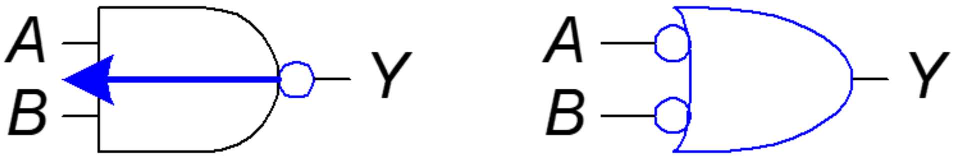
- $Y = \overline{A + B} = \overline{A} \bullet \overline{B}$



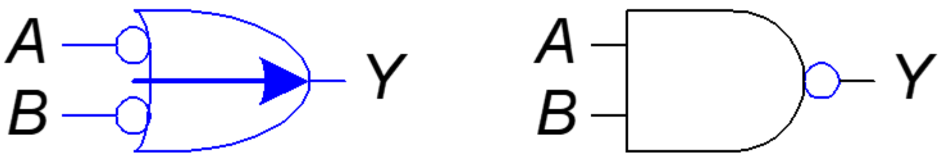


Bubble Pushing

- **Backward:**
  - Body changes
  - Adds bubbles to inputs

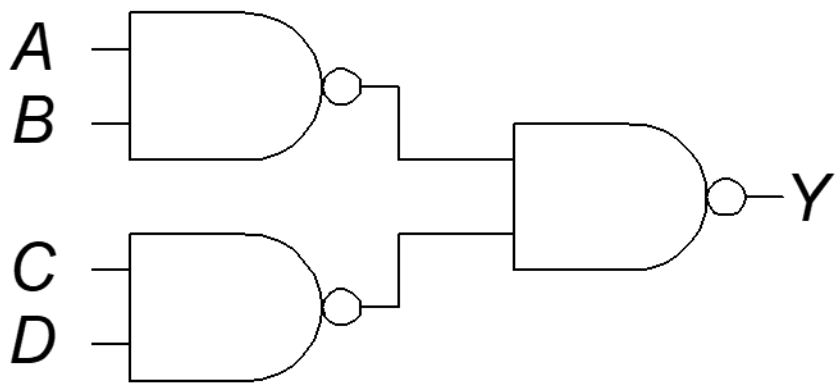


- **Forward:**
  - Body changes
  - Adds bubble to output



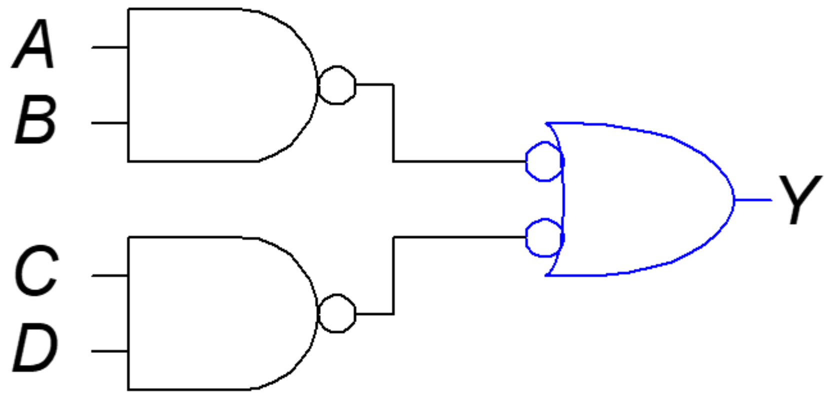
Bubble Pushing

- What is the Boolean expression for this circuit?



Bubble Pushing

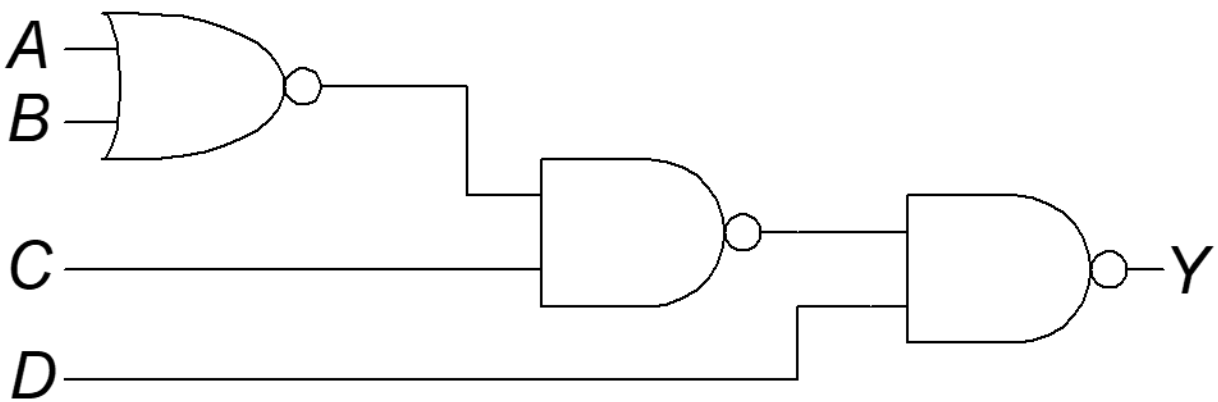
- What is the Boolean expression for this circuit?



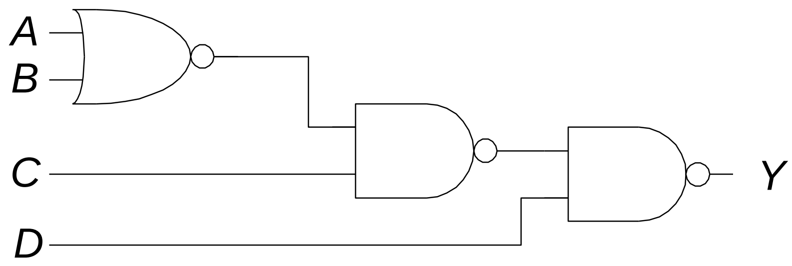
$$Y = AB + CD$$

Bubble Pushing Rules

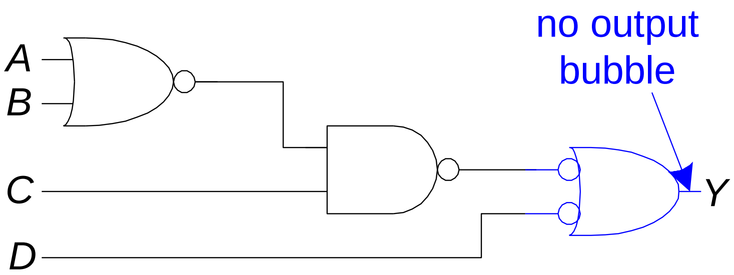
- Begin at output, then work toward inputs
- Push bubbles on final output back
- Draw gates in a form so bubbles cancel



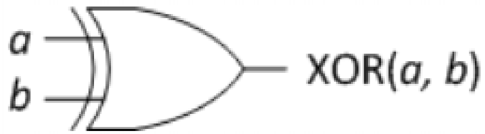
Bubble Pushing Example

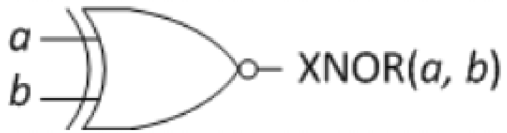


Bubble Pushing Example



XOR – XNOR Gates





a b	XOR(*a*,*b*)	XNOR(*a*,*b*)
0 0	0	1
0 1	1	0
1 0	1	0
1 1	0	1

XOR (= eXclusive OR): XOR(a, b) = 1 if $a \neq b$;

XNOR (= eXclusive NOR): XNOR(a, b) = 1 if $a = b$.

Algebraic symbols:

XOR(a, b) = $a \oplus b$,

(XNOR(a, b) = $a \equiv b$)

XOR – XNOR Gates

Equivalent definition:

$\mathrm{XOR}(a, b) = (a + b) \bmod 2 = a \oplus b,$

$\mathrm{XNOR}(a, b) = \mathrm{INV}(a \oplus b).$

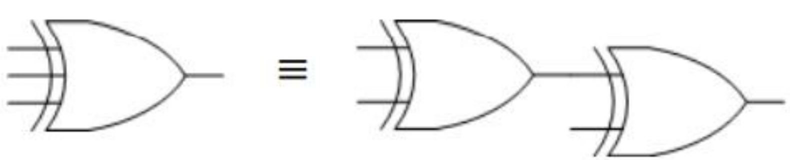
=> 3-input, 4-input, ⋯ XOR and XNOR gates can be defined:

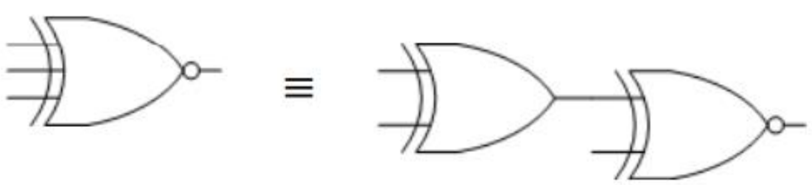
$\mathrm{XOR}(a, b, c) = (a + b + c) \bmod 2 = a \oplus b \oplus c,$ $\mathrm{XNOR}(a, b, c) = \mathrm{INV}(a \oplus b \oplus c),$

$\mathrm{XOR}(a, b, c, d) = (a + b + c + d) \bmod 2 = a \oplus b \oplus c \oplus d,$ $\mathrm{XNOR}(a, b, c, d) = \mathrm{INV}(a \oplus b \oplus c \oplus d),$

...

XOR is an associative operation =>



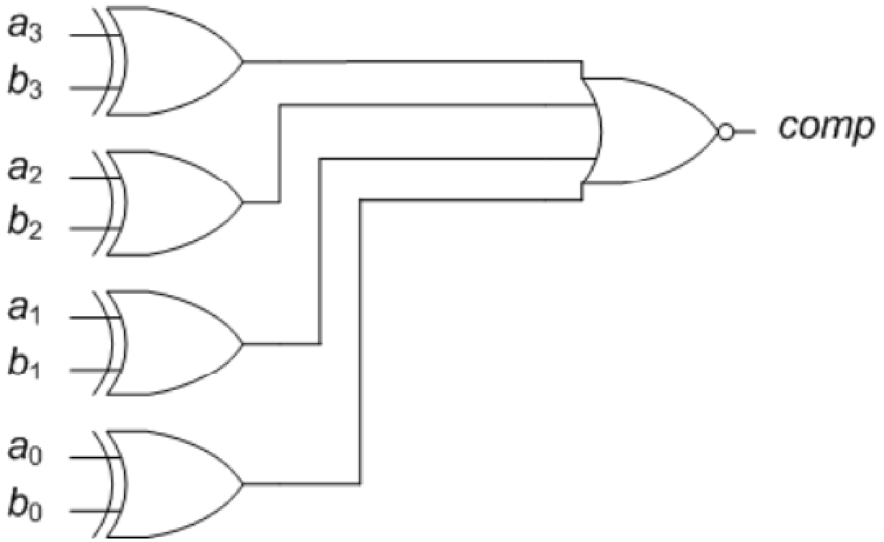


XOR – XNOR Gates

- XOR y NXOR are **not universal modules,**
- **useful functions.**

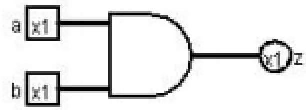
First example: magnitud comparator. Given two 4-input vectors $a = a_3\, a_2\, a_1\, a_0$ and $b = b_3\, b_2\, b_1\, b_0$, generate *comp* = 1 iff $a = b$.

Algorithm
if $(a_3 \neq b_3)$ or $(a_2 \neq b_2)$ or $(a_1 \neq b_1)$ or $(a_0 \neq b_0)$
then *comp* <= 0;
else *comp* <= 1;
end if;



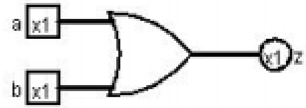
Summary

AND



$z = a.b$

OR

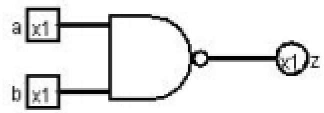


$z = a + b$

INV

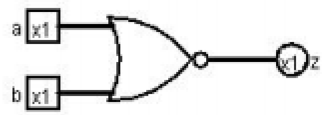
$z = \bar{a}$

NAND



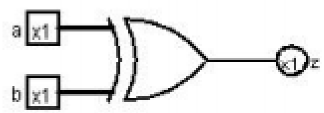
$z = a \uparrow b = \overline{a.b}$

NOR



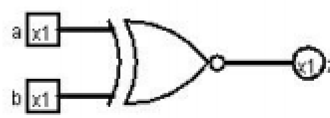
$z = a \downarrow b = \overline{a + b}$

XOR



$z = a \oplus b = \bar{a}.b + a.\bar{b}$

XNOR



$z = \overline{a \oplus b} = \bar{a}.\bar{b} + a.b$