2.1 COMBINATIONAL CIRCUITS

Jean-Pierre Deschamps

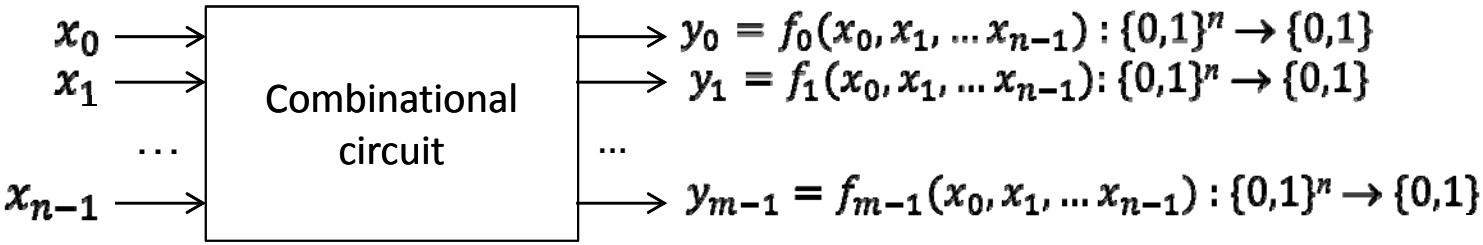
University Rovira i Virgili, Tarragona, Spain

1. Combinational circuits

Digital circuits that implement one or several **switching functions**, in such a way that, at any time, the output signal values only depend on the input signal values at the same time.

$$x_i \in \{0,1\}$$

$$y_i \in \{0,1\}$$

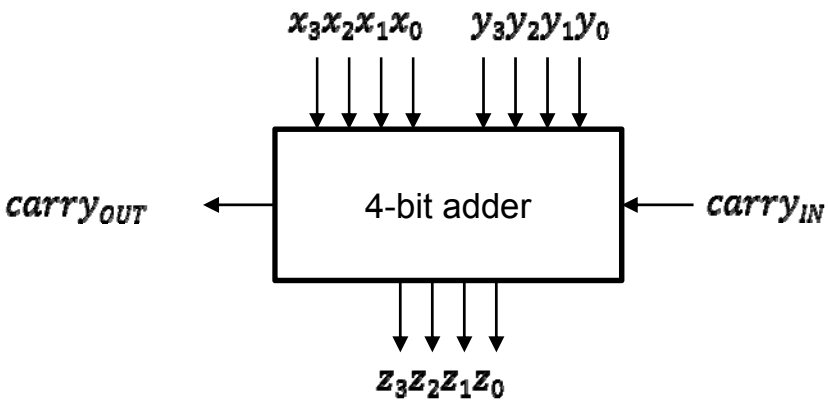


This is NOT the definition of a combinational circuit

temp	*onoff*
0	ON
1	ON
…	…
18	ON
19	ON
20	DON’T CHANGE
21	OFF
22	OFF
…	…
49	OFF
50	OFF

1. Combinational circuits

Adder of two 4-bit numbers (4-bit adder)



$$X = x_3x_2x_1x_0,$$
$$Y = y_3y_2y_1y_0$$

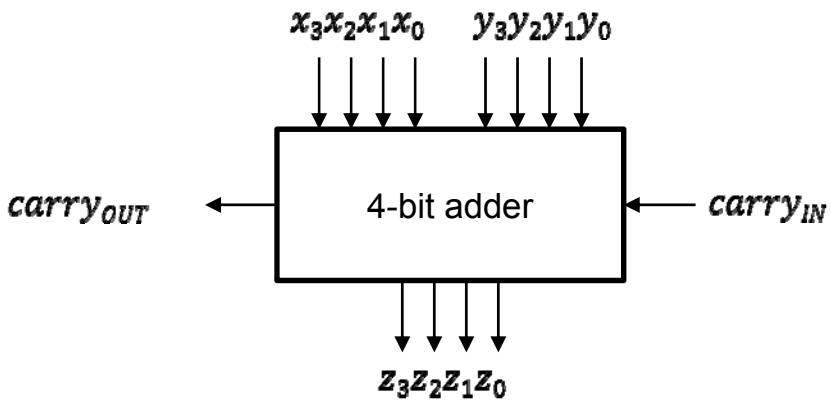
$$Z = X + Y + carry_{IN} = z_4z_3z_2z_1z_0$$

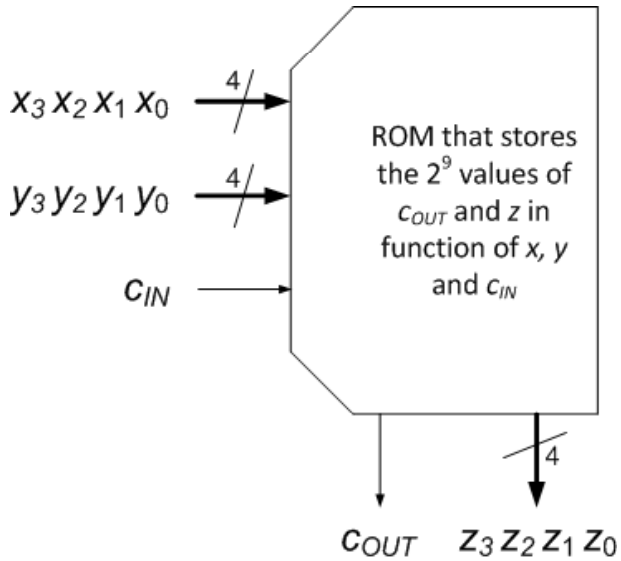
$$z_4 \equiv carry_{OUT}$$

```
s <= X + Y + carry_IN;
if s > 1111 then Z(3 downto 0) <= s - 10000; carry_OUT <= 1;
            else Z(3 downto 0) <= s; carry_OUT <= 0;
end if;
```

2.1 Synthesis from a table: ROM

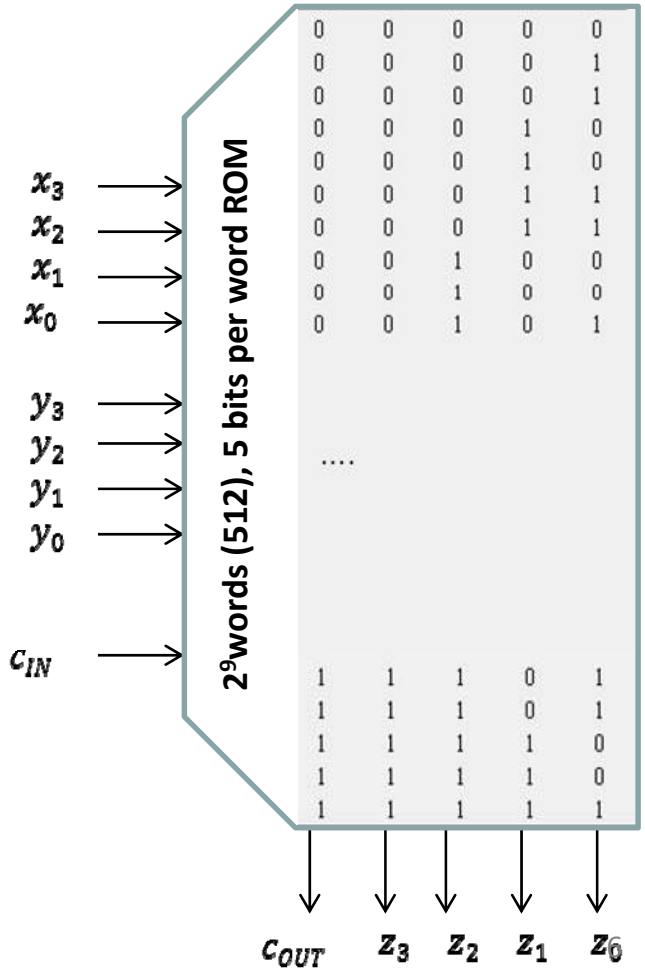
Adder of two 4-bit numbers (4-bit adder)





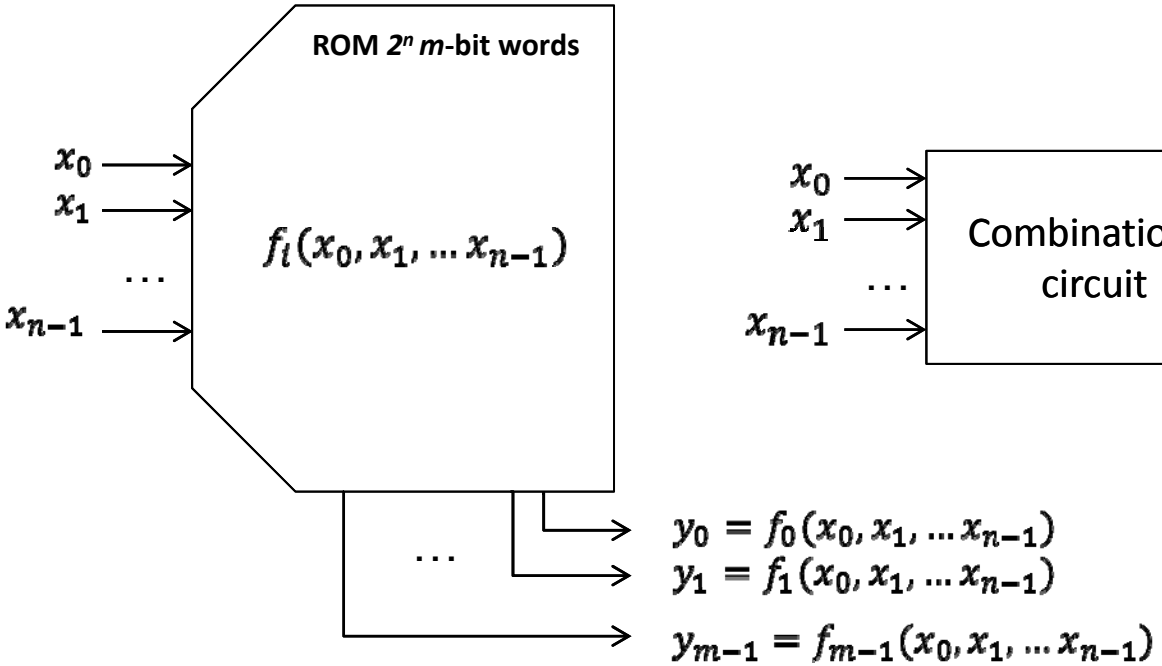
TRUTH TABLE

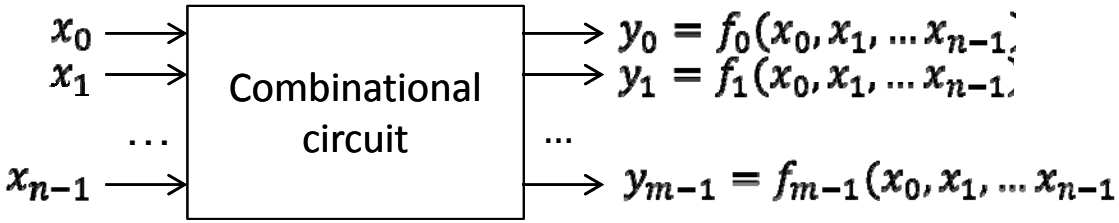
x3	x2	x1	x0	y3	y2	y1	y0	aIN	aOUT	z3	z2	z1	z0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	0	1	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	1	0	0	0	1	1
0	0	0	0	0	0	1	1	0	0	0	0	1	1
0	0	0	0	0	0	1	1	1	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	0	1	0	1
….									….				
1	1	1	1	1	1	0	1	1	1	1	1	0	1
1	1	1	1	1	1	1	0	0	1	1	1	0	1
1	1	1	1	1	1	1	0	1	1	1	1	1	0
1	1	1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1



2.1 Synthesis from a table: ROM

CC with *n* inputs and *m* outputs → ROM with 2^n words, *m* bits per word





BUT generally inefficient !

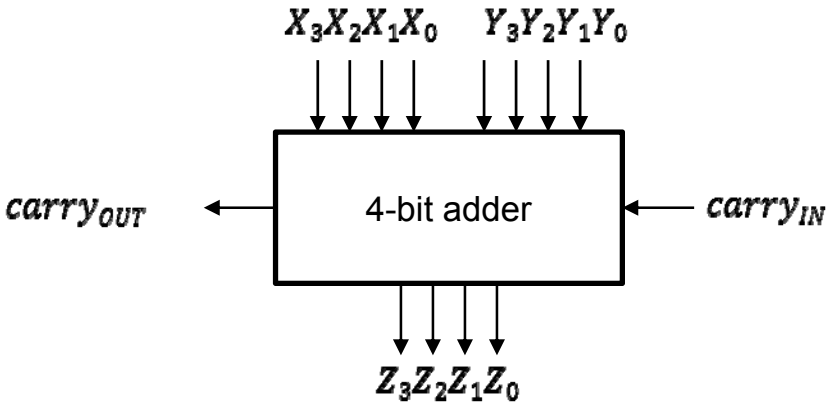
Universitat Autònoma de Barcelona

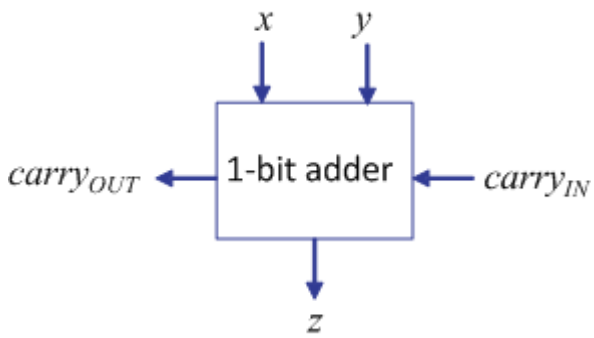
(quizz)

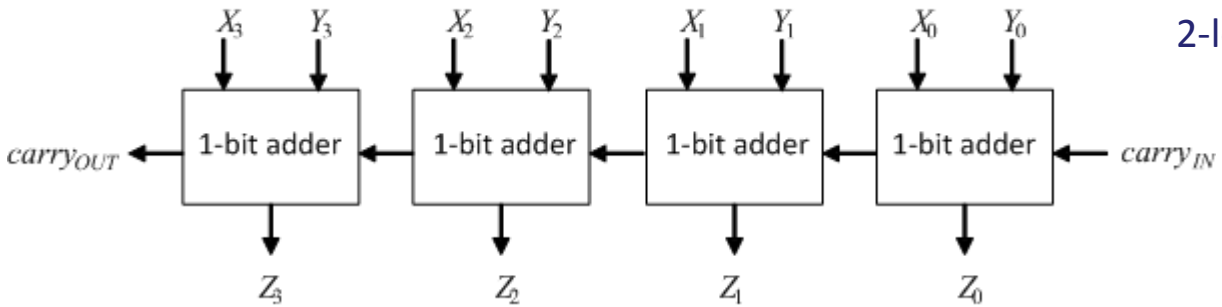
Minimum size (number of words, number of bits per word of a ROM that implements an 8-input, 16-output combinational circuit?

1. 8 16-bit words
2. 2^3 16-bit words
3. 2^8 16-bit words
4. 16 8-bit words
5. 2^4 8-bit words
6. 2^{16} 8-bit words

2.2 Synthesis from a table: logic Gates

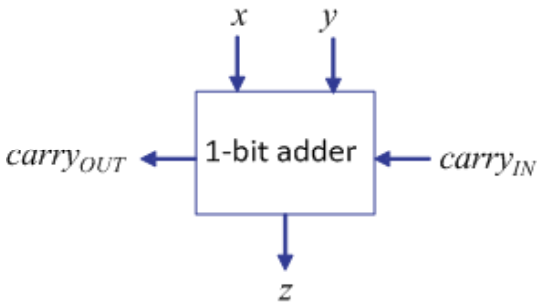




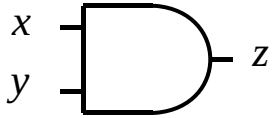


2-level hierarchy

2.2 Synthesis from a table: logic Gates

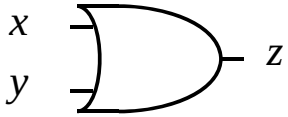


x	y	c_i	c_o	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



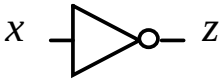
x	y	z
0	0	
0	1	
1	0	
1	1	

AND



x	y	z
0	0	
0	1	
1	0	
1	1	

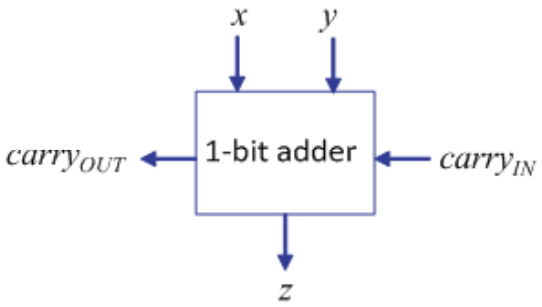
OR



x	z
0	
1	

INV

2.2 Synthesis from a table: logic Gates



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

$c_0 = 1$ iff
$(x\ y\ c_i = 011)$ OR $(x\ y\ c_i = 101)$ OR $(x\ y\ c_i = 110)$ OR $(x\ y\ c_i = 111)$

$x\ y\ c_i = 011$ iff
$(x = 0)$ AND $(y = 1)$ AND $(c_i = 1)$

$x = 0$ iff $INV(x) = 1$

Universitat Autònoma de Barcelona

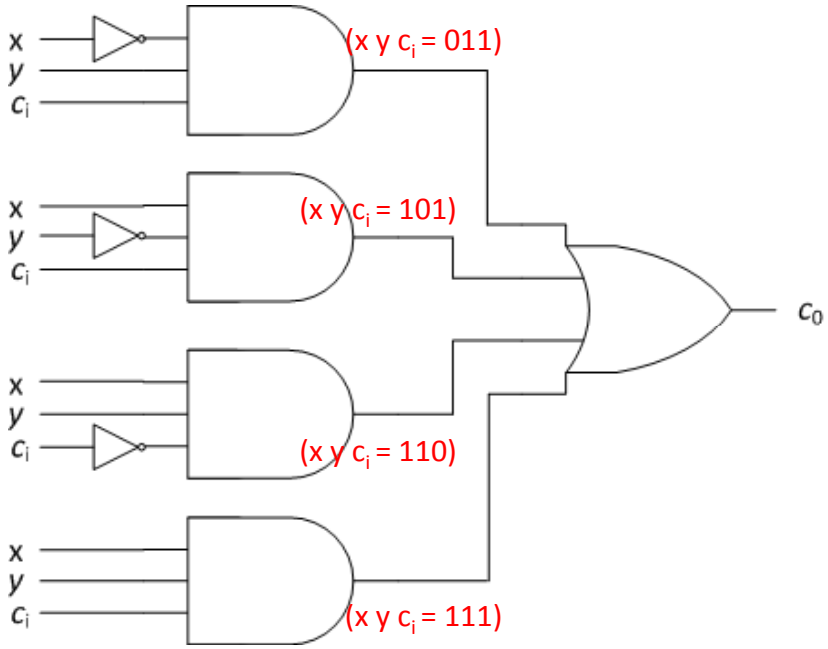
2.2 Synthesis from a table: logic Gates

$c_0 = 1$ iff
$(x\ y\ c_i = 011)$ OR $(x\ y\ c_i = 101)$ OR $(x\ y\ c_i = 110)$ OR $(x\ y\ c_i = 111)$

$x\ y\ c_i = 011$ iff
$(x = 0)$ AND $(y = 1)$ AND $(c_i = 1)$

$x = 0$ iff $INV(x) = 1$

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



Universitat Autònoma de Barcelona

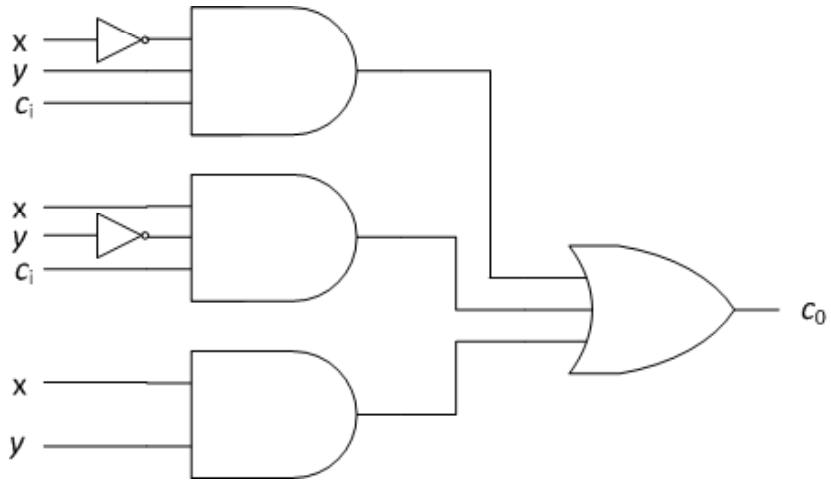
2.2 Synthesis from a table: logic Gates

$c_0 = 1$ iff
$(x\ y\ c_i = 011)$ OR $(x\ y\ c_i = 101)$ OR $(x\ y\ c_i = 110)$ OR $(x\ y\ c_i = 111)$

equivalent to

$c_0 = 1$ iff
$(x\ y\ c_i = 011)$ OR $(x\ y\ c_i = 101)$ OR $(x\ y = 11)$

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



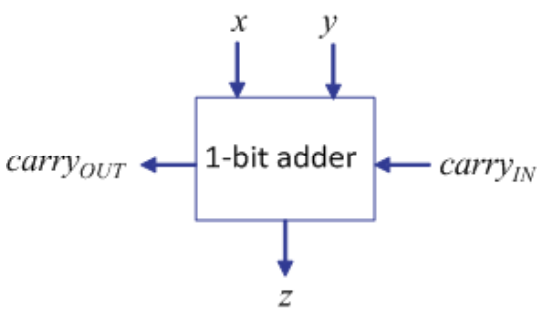
Universitat Autònoma de Barcelona

We need a tool that helps us to minimize the number of gates:

BOOLEAN ALGEBRA

Universitat Autònoma de Barcelona

(Exercise)

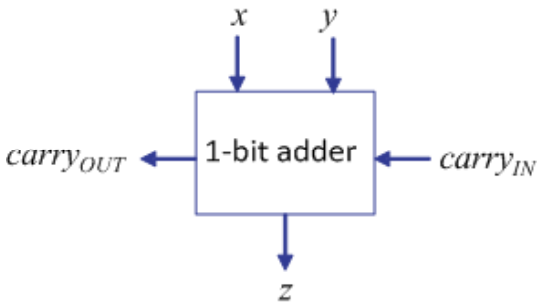


Synthesize the function z with logic gates.

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

Universitat Autònoma de Barcelona

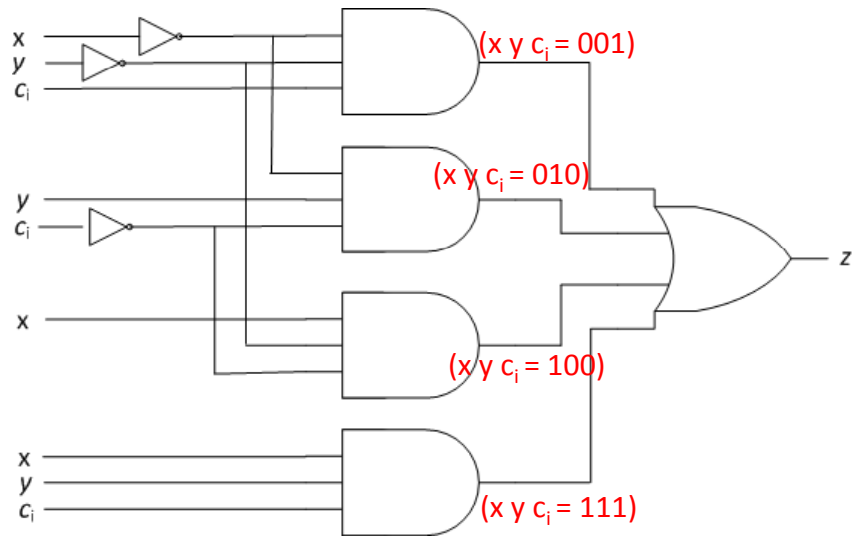
(solution)



Synthesize the function z with logic gates.

z = 1 iff
(x y c_i = 001) OR (x y c_i = 010) OR (x y c_i = 100) OR (x y c_i = 111)

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



Universitat Autònoma de Barcelona

SUMMARY

- Combinational circuits.
- ROM (table) implementation.
- A first approach to logic gate implementation.

2.2 BOOLEAN ALGEBRA

Jean-Pierre Deschamps

University Rovira i Virgili, Tarragona, Spain

UAB
Universitat Autònoma de Barcelona

1. Boolean algebra

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

1. Boolean algebra

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**: $\forall a \in B, \exists \; \overline{a} \in B \,|\, a+\overline{a}=1, \quad a \cdot \overline{a} = 0$

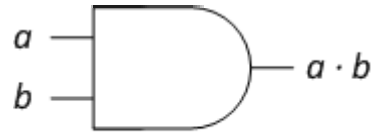
P 4 – Operations + and · are **conmutative**: $a+b=b+a, \quad a \cdot b = b \cdot a$

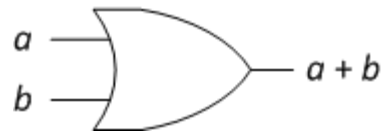
P 5 –Operations + and · are **distributive:** $a \cdot (b+c) = a \cdot b + a \cdot c, \quad a+b \cdot c = (a+b) \cdot (a+c)$

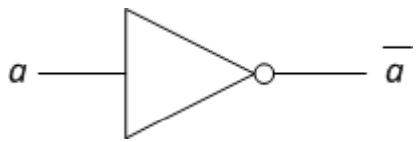
1. Boolean algebra

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







1. Boolean algebra

Universitat Autònoma de Barcelona

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

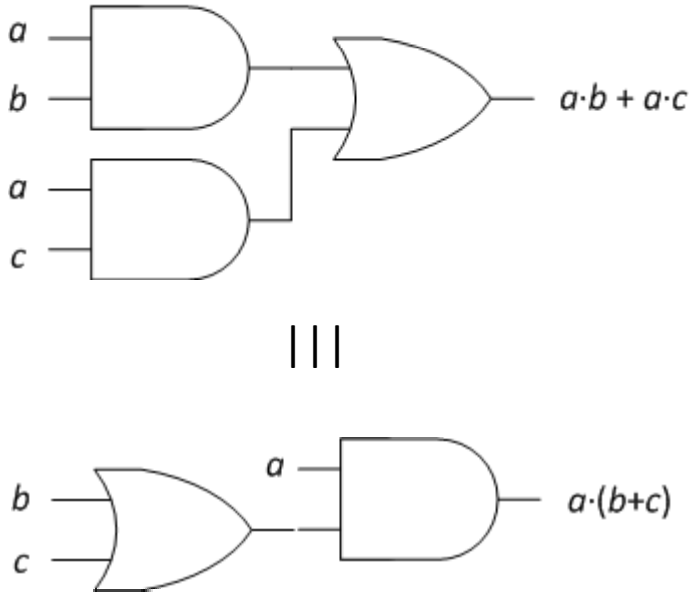
$a\ b$	$a\cdot b$	$a + b$	$\overline{a}$
0 0	0	0	1
0 1	0	1	1
1 0	0	1	0
1 1	1	1	0

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

1. Boolean algebra

Comment:

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



1. Boolean algebra

The set of n-variable switching functions

$$F: \{0, 1\}^n \rightarrow \{0, 1\}$$

is also a Boolean algebra. Given two switching functions f and g, then $f + g$, $f \cdot g$ and $\overline{f}$ are defined as follows:

$$(f + g)(x_0, x_1, \cdots, x_{n-1}) = f(x_0, x_1, \cdots, x_{n-1}) + g(x_0, x_1, \cdots, x_{n-1}),$$

$$(f \cdot g)(x_0, x_1, \cdots, x_{n-1}) = f(x_0, x_1, \cdots, x_{n-1}) \cdot g(x_0, x_1, \cdots, x_{n-1}),$$

$$\overline{f}(x_0, x_1, \cdots, x_{n-1}) = \overline{f(x_0, x_1, \cdots, x_{n-1})}$$

The neutral elements are the constant functions 0 and 1.

Universitat Autònoma de Barcelona

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

Universitat Autònoma de Barcelona

(Exercise)

Demonstrate that $a \cdot a = a$

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

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

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

Universitat Autònoma de Barcelona

(Solution)

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

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

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

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

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

Universitat Autònoma de Barcelona

2. 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: $\overline{(a_1+a_2+\ldots+a_n)}=\overline{a_1}\cdot\overline{a_2}\cdot\ldots\cdot\overline{a_n}, \quad \overline{a_1\cdot a_2\cdot\ldots\cdot a_n}=\overline{a_1}+\overline{a_2}+\ldots+\overline{a_n}$

Universitat Autònoma de Barcelona

(quizz)

What Boolean expression is equivalent to the following : $a \cdot (\overline{\overline{b} + cd}) + \overline{a}.b$?

Hint: Use postulates and properties

1. $\overline{a}.b + b.\overline{c} + \overline{d}$
2. $\overline{a}.b$
3. $a.\overline{b} + b.\overline{c} + \overline{d}$
4. $\overline{a}.b + b.\overline{c} + b.\overline{d}$

3. Boolean functions and truth tables

a) 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

3. Boolean functions and truth tables

b) 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*.

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

3. 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	
$\bar{a}.\bar{b}.\bar{c}=1$	$\Leftrightarrow$	0	0	0	$\rightarrow m_0=\bar{a}.\bar{b}.\bar{c}$
$\bar{a}.\bar{b}.c=1$	$\Leftrightarrow$	0	0	1	$\rightarrow m_1=\bar{a}.\bar{b}.c$
$\bar{a}.b.\bar{c}=1$	$\Leftrightarrow$	0	1	0	$\rightarrow m_2=a.\bar{b}.c$
$\bar{a}.b.c=1$	$\Leftrightarrow$	0	1	1	$\rightarrow m_3=\bar{a}.b.c$
$a.\bar{b}.\bar{c}=1$	$\Leftrightarrow$	1	0	0	$\rightarrow m_4=a.\bar{b}.\bar{c}$
$a.\bar{b}.c=1$	$\Leftrightarrow$	1	0	1	$\rightarrow m_5=a.\bar{b}.c$
$a.b.\bar{c}=1$	$\Leftrightarrow$	1	1	0	$\rightarrow m_6=a.b.\bar{c}$
$a.b.c=1$	$\Leftrightarrow$	1	1	1	$\rightarrow m_7=a.b.c$

(quiz)

What expression corresponds to *minterm-5* (m_5) of n = 4 variables?

1. $a.\overline{b}.c.\overline{d}$
2. $a.\overline{b}.c$
3. $\overline{a}.b.\overline{c}.d$
4. $a.b.\overline{c}.\overline{d}$

3. Boolean functions and truth tables

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

3. Boolean functions and truth tables

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

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

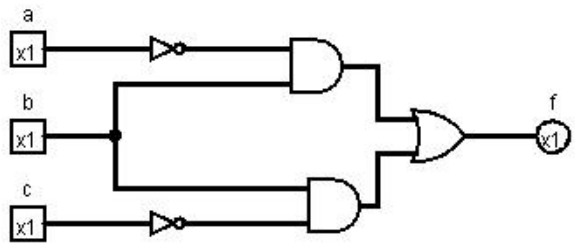
$$f(a,b,c) = \Sigma(m_2, m_3, m_6)$$

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

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

3. Boolean functions and truth tables

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

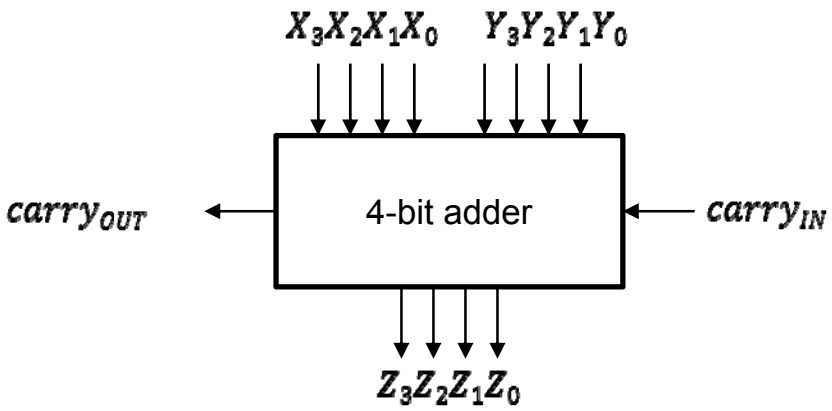


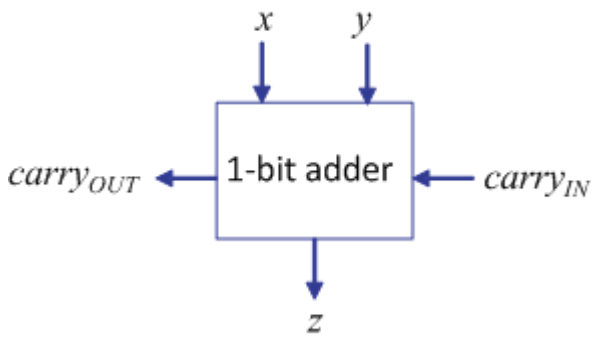
a	*b*	*c*	*f(a,b,c)*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

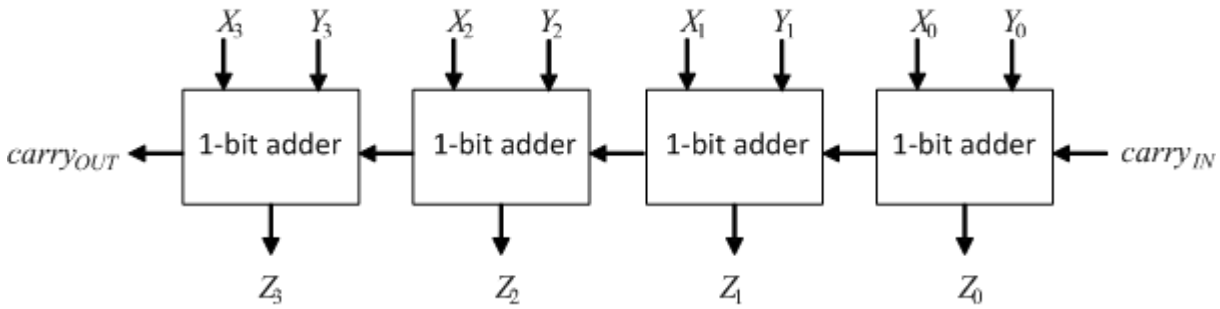
$$f(a,b,c) = \bar{a}.b.\bar{c} + \bar{a}.b.c + a.b.\bar{c} =$$
$$= \bar{a}.b(\bar{c}+c) + b.\bar{c}.(\bar{a}+a) = \bar{a}.b + b.\bar{c}$$

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

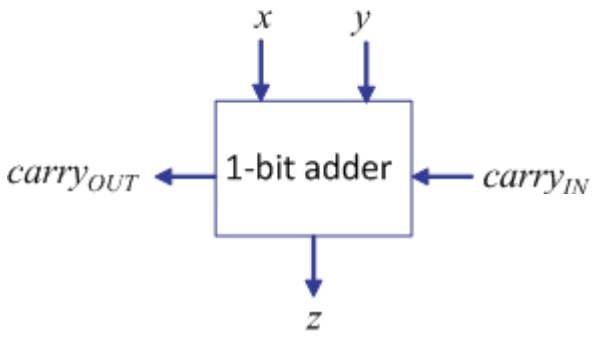
4. Example: 4 bit-adder







4. Example: 4 bit-adder

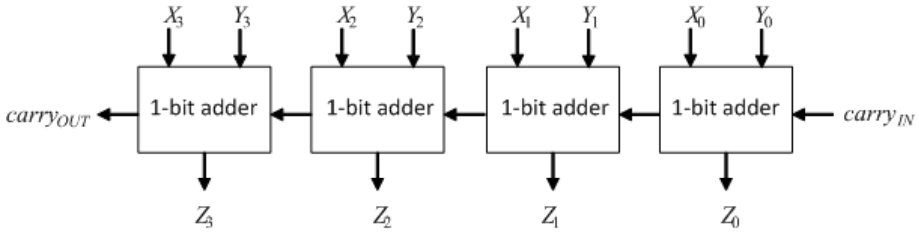


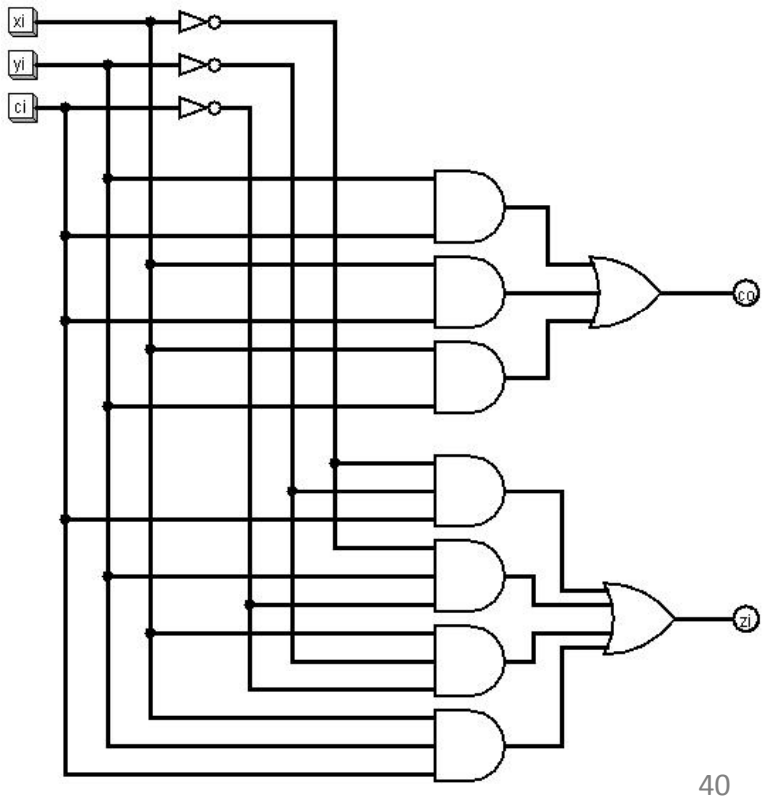
x	y	c_i	c_o	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

4. Example: 4 bit-adder

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

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





SUMMARY

- Boolean algebra. Postulates and properties.
- Tabular representation of Boolean functions.
- *Minterms* and canonical sum of products expression.
- Circuit generation from a functional description:

  (functional description $\rightarrow$ truth table $\rightarrow$ Boolean function(s) $\rightarrow$ circuit)

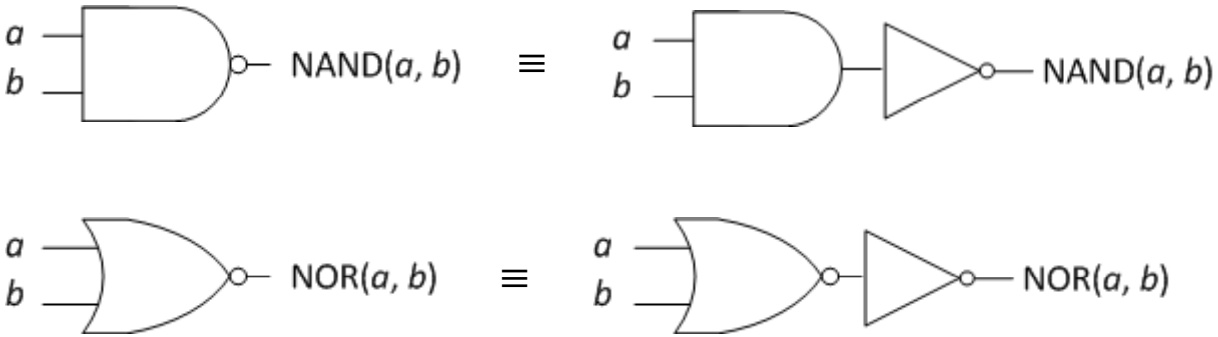
2.3 NAND, NOR, XOR, XNOR, TRI-STATE

Jean-Pierre Deschamps

University Rovira i Virgili, Tarragona, Spain

UAB
Universitat Autònoma
de Barcelona

1. NAND, NOR



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

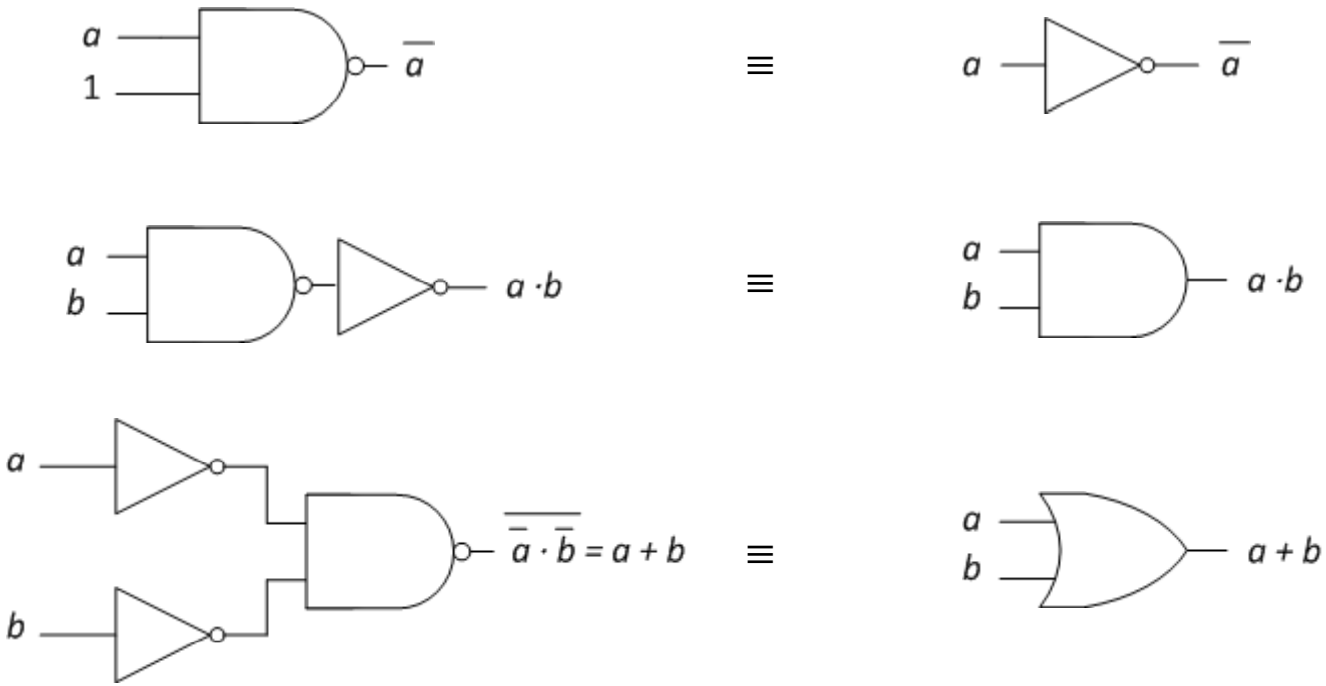
Algebraic symbols:

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

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

1. NAND, NOR

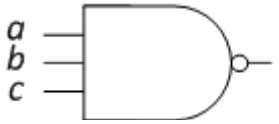
NAND and NOR gates are **universal modules**. For example, with NAND gates:



Exercise: the same for NOR gates

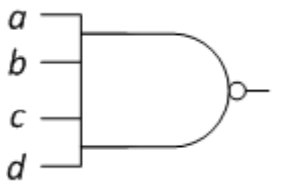
1. NAND, NOR

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



NAND(*a, b, c*)

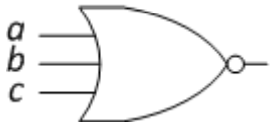
NAND(*a, b, c*) = 0 iff *a* = *b* = *c* = 1



NAND(*a, b, c, d*)

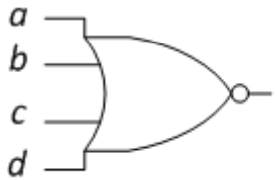
NAND(*a, b, c,* d) = 0 iff *a* = *b* = *c* = *d* = 1

.....



NOR(*a, b, c*)

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

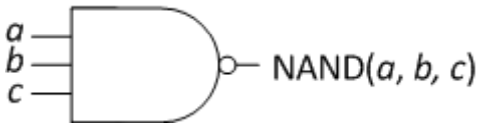


NOR(*a, b, c, d*)

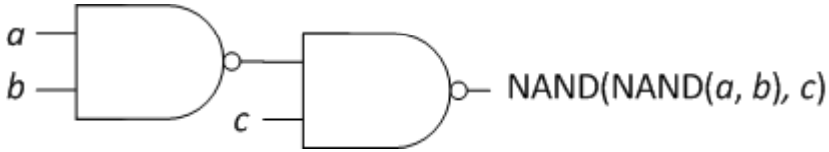
NOR(*a, b, c,* d) = 0 iff (*a* = 1) OR (*b* = 1) OR (*c* = 1) OR (*d* = 1)

1. NAND, NOR

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

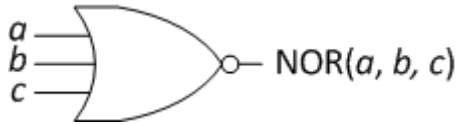


≠

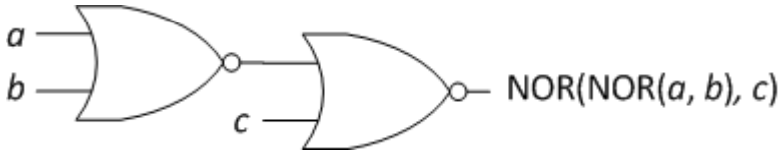


NAND(1, 1, 1) = 0

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



≠



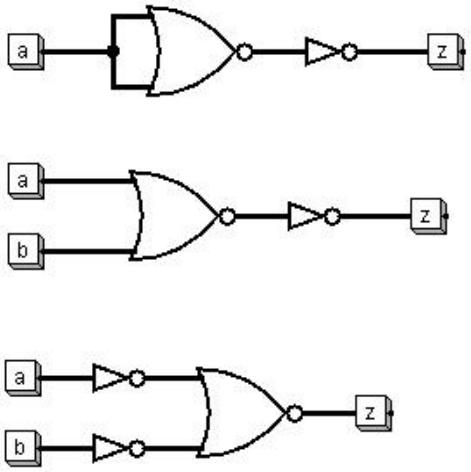
NOR(0, 0, 0) = 1

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

Universitat Autònoma de Barcelona

(quiz)

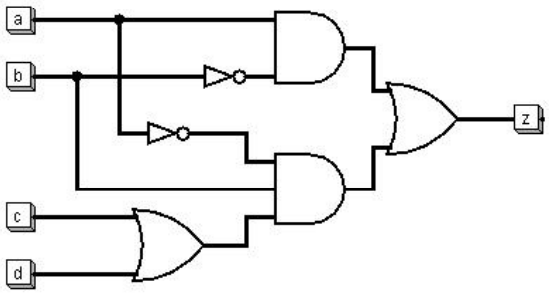
Which of the following circuits implements the AND function $z = a \cdot b$?



Universitat Autònoma de Barcelona

(Exercise)

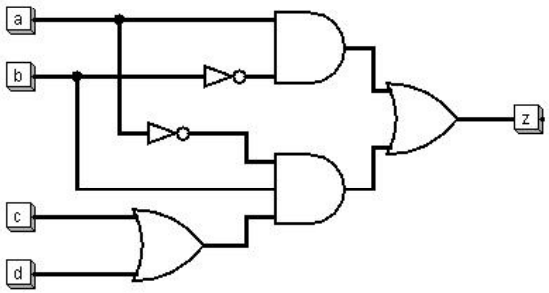
Implement the same function with NAND gates.

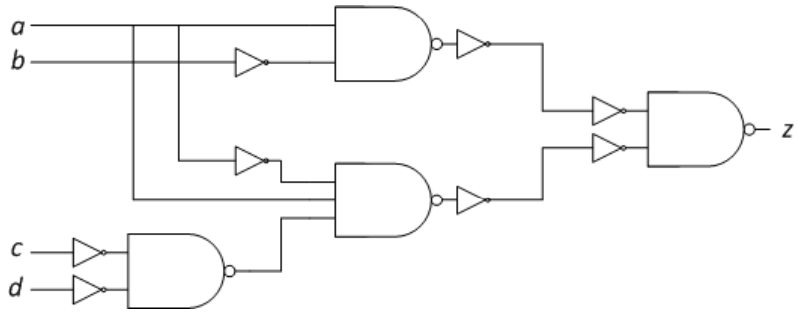


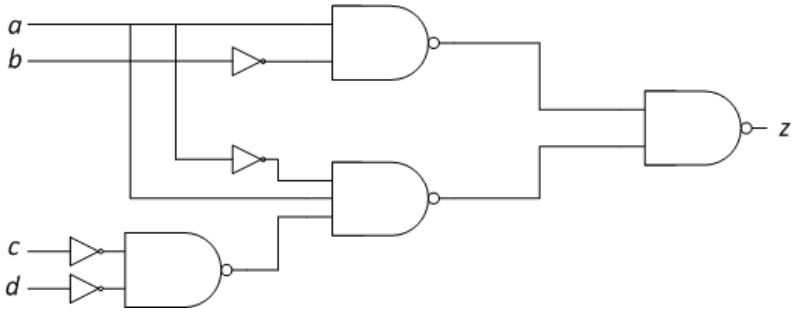
Universitat Autònoma de Barcelona

(solution)

Implement the same function with NAND gates.





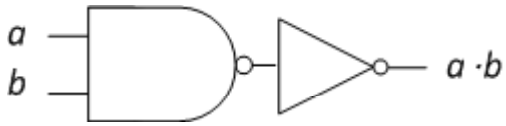


1. NAND, NOR

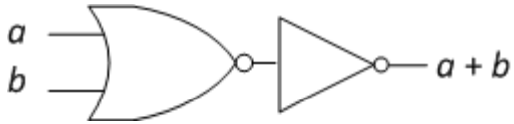
Why do we use NAND gates (or NOR gates) instead of AND and OR gates?

- 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,



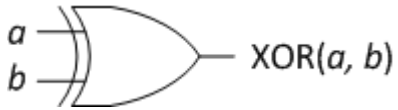
  - an OR gate is implemented with a NOR and an INV.

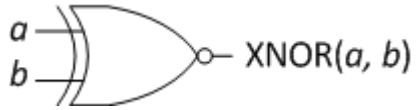


=> Within an IC (Integrated Circuit) NAND and NOR are “cheaper” than AND and OR.

Universitat Autònoma de Barcelona

2. XOR, XNOR





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) = $a \equiv b$)

2. XOR, XNOR

Equivalent definition:

XOR(a, b) = (a + b) mod 2 = $a \oplus b$,

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

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

XOR(a, b, c) = (a + b + c) mod 2 = $a \oplus b \oplus c$, XNOR(a, b, c) = INV($a \oplus b \oplus c$),

XOR(a, b, c, d) = (a + b + c + d) mod 2 = $a \oplus b \oplus c \oplus d$, XNOR($a$, b, c, d) = INV($a \oplus b \oplus c \oplus d$),

...

XOR is an associative operation =>

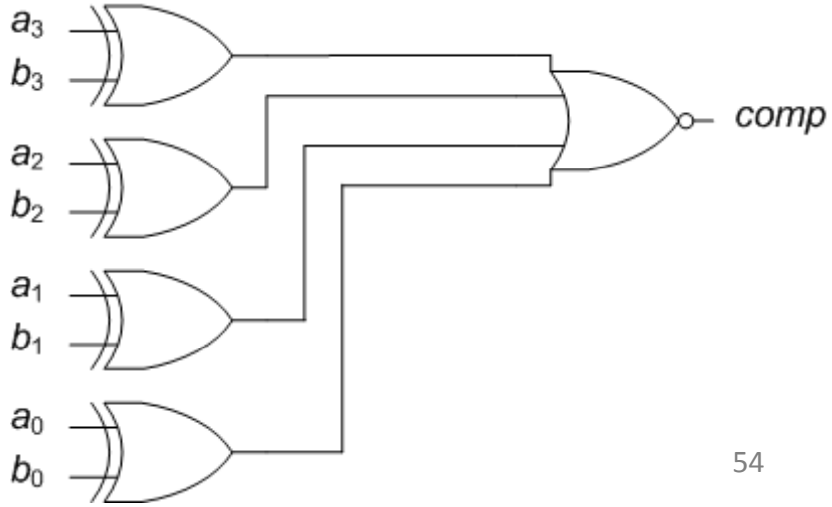
2. XOR, XNOR

- 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;



Universitat Autònoma de Barcelona

2. XOR, XNOR

Second example: parity bit generation.
Given an n-input vector

$$a = a_{n-1}\, a_{n-2} \cdots a_1\, a_0$$

its **parity bit** is and additional bit a_n such that the extended vector

$$a_{ext} = a_n\, a_{n-1}\, a_{n-2} \cdots a_1\, a_0$$

has an even number of 1's. It is used for **error detection** purpose:

Observation:

- a vector $a_{k-1}\, a_{k-2} \cdots a_0$ has an even number of 1's iff

  $$(a_{k-1} + a_{k-2} + \cdots + a_0) \bmod 2 = 0,$$

  and

  $$a_{k-1} \oplus a_{k-2} \oplus \cdots \oplus a_0 = 0.$$

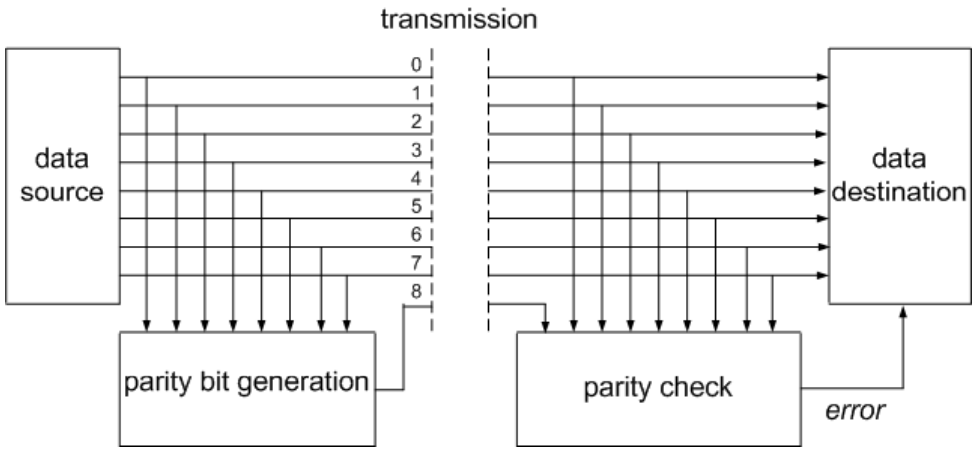
Algorithm – Parity bit generation

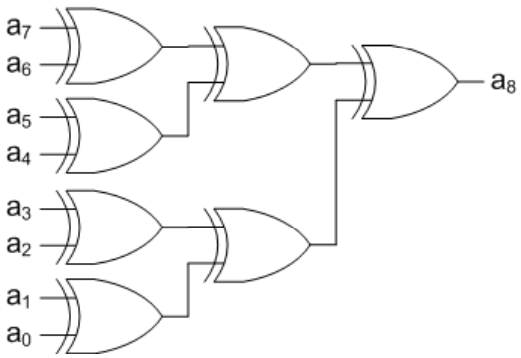
a(n) <= a(n-1) xor a(n-2) xor ··· xor a(0);

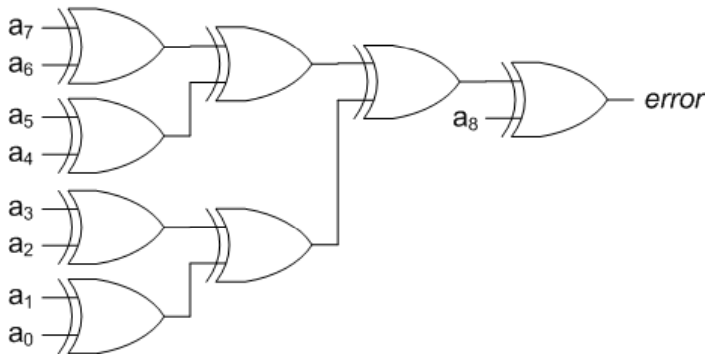
Algorithm – Parity check

error <=
a(n) xor a(n-1) xor a(n-2) xor ··· xor a(0);

2. XOR, XNOR


transmission
data source
0
1
2
3
4
5
6
7
8
data destination
parity bit generation
parity check
error


a_7
a_6
a_5
a_4
a_3
a_2
a_1
a_0
a_8


a_7
a_6
a_5
a_4
a_3
a_2
a_1
a_0
a_8
error

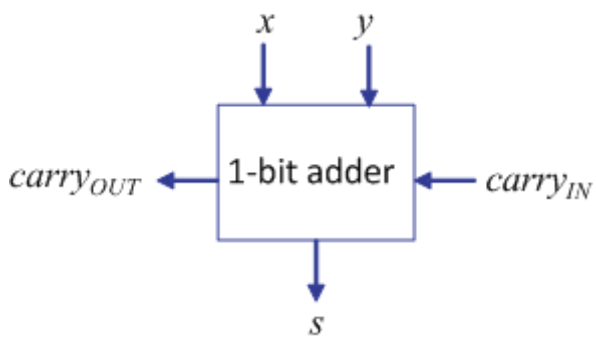
2. XOR, XNOR

The main application of XOR gates is **Arithmetic:**

1-bit adder is the basic component of practically all arithmetic circuits;

It computes two functions:

- $carry_{OUT} = 1$ iff $x + y + carry_{IN} \geq 2$;
- $s = (x + y + carry_{IN}) \bmod 2 = x \oplus y \oplus carry_{IN}$.



2. XOR, XNOR

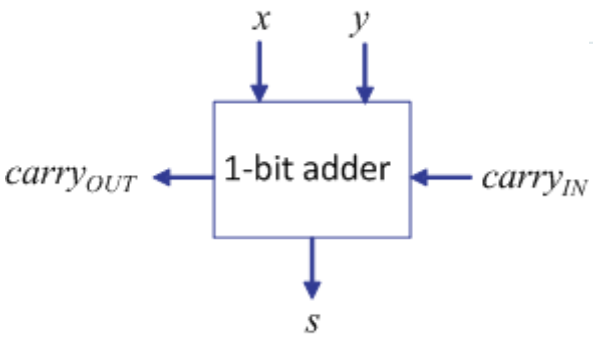
Condition $x + y + carry_{IN} \geq 2$ is equivalent to

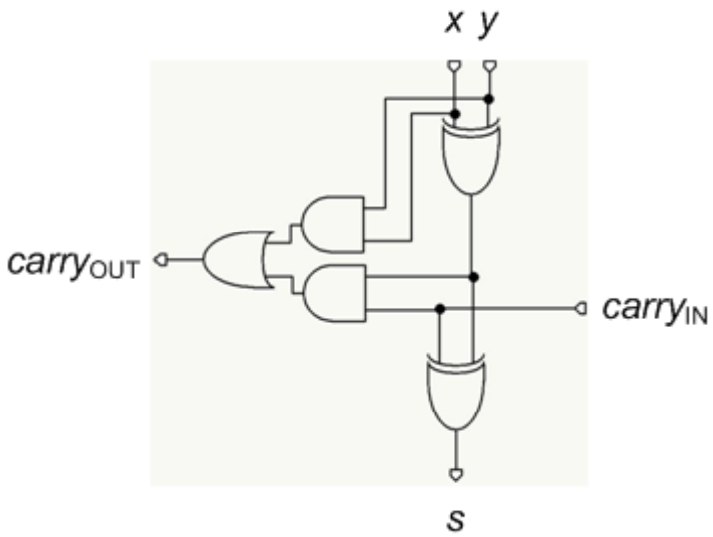
$((x = 1)$ and $(y = 1))$
or $((carry_{IN} = 1)$ and $(x \neq y))$;

Thus

$carry_{OUT} = x \cdot y + carry_{IN} \cdot (x \oplus y)$,

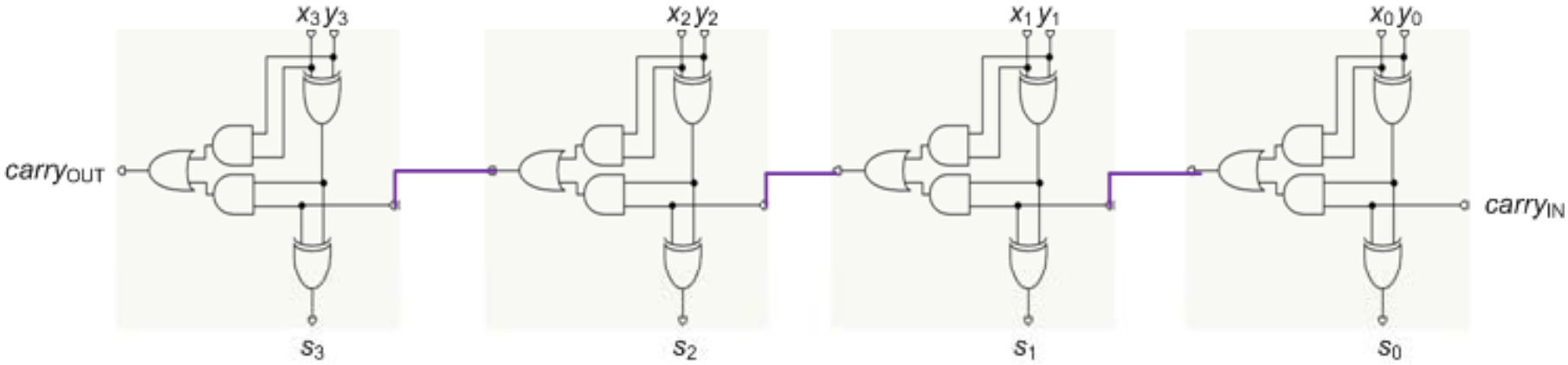
$s = x \oplus y \oplus carry_{IN}$.



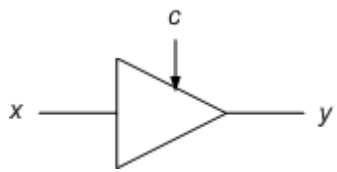


2. XOR, XNOR

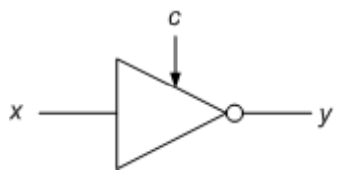
4-bit adder (new version):



3. BUFFER TRI-STATE, INVERSOR TRI-STATE

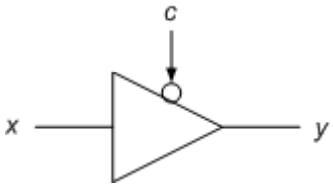


c x	*y*
0 0	High impedance (*Z*)
0 1	High impedance (*Z*)
1 0	0
1 1	1

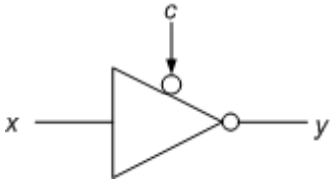


c x	*y*
0 0	High impedance (*Z*)
0 1	High impedance (*Z*)
1 0	1
1 1	0

3. BUFFER TRI-STATE, INVERSOR TRI-STATE



c x	*y*
0 0	0
0 1	1
1 0	High impedance (*Z*)
1 1	High impedance (*Z*)



c x	*y*
0 0	1
0 1	0
1 0	High impedance (*Z*)
1 1	High impedance (*Z*)

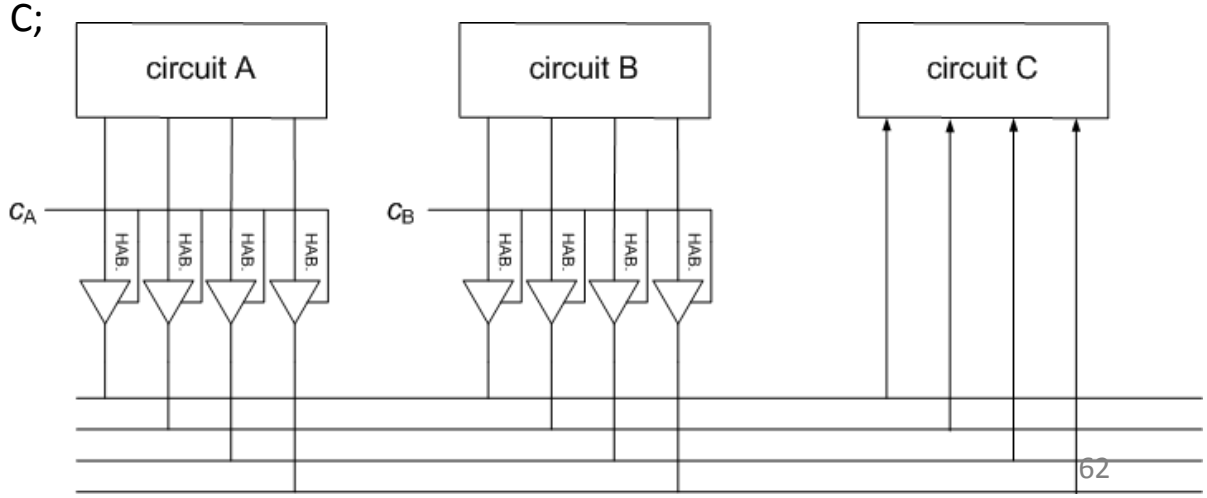
3. BUFFER TRI-STATE, INVERSOR TRI-STATE

Main application: **BUS**

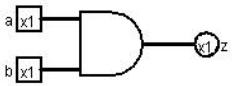
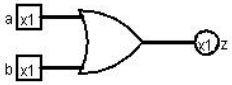
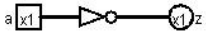
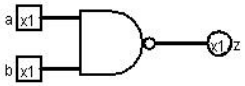
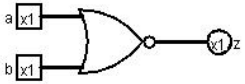
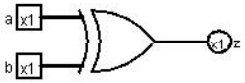
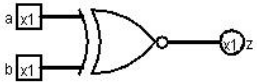
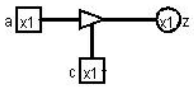
Example: 4-bit bus

$c_A = 1$ and $c_B = 0$: circuit A $\rightarrow$ circuit C;

$c_A = 0$ and $c_B = 1$: circuit B $\rightarrow$ circuit C;



Universitat Autònoma de Barcelona

nombre	símbolo	función
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		$z = \overline{a \oplus b} = \bar{a}.\bar{b} + a.b$
Tri-state		$z = a \; si \; c = 1, z = HI \; si \; c = 0$

Universitat Autònoma de Barcelona

SUMMARY

- NAND, NOR. Universal module concept.
- XOR, XNOR
- Tri-state buffers. Bus.