Computers Fundamentals

Subject 2. Principles of digital design

Aims and objectives

- What a logic function is and its mathematical representation
- How to design basic logic circuits
- Fundamentals of Boole’s Algebra
- How to simplify circuits.
  - Karnaugh’s Maps

Outline

- Introduction
- Logic functions and *truth tables*
- Basic logic gates
- Boole's Algebra
- Canonical forms of boolean functions
- Logic function's simplification
  - Karnaugh's maps

http://en.wikipedia.org/wiki/Canonical_form_(Boolean_algebra)

Introduction

- Transistor
  - Minimal digital design-unit of design
- Logic gate
  - Minimal logic-unit of digital design
- Combinatorial circuit
  - Output values only depend on input values at any time

Introduction

- Sequential circuits
  - Output values depend on the input values and on the memory of the circuit.
- Functional unit
  - A circuit that realize a well defined function

Logic functions and thruth tables

- Logic function
  - Mathematical representation of the behavior of a digital circuit
  - Used to evaluate the output's circuit value in function of the values of the input variables
  - Arity= total number of input variables
  - Valuation= the value to which a logic function evaluates when each one of the input variables are assigned a value 1 (for truth) or 0 (for falsity).

http://en.wikipedia.org/wiki/Truth_table

http://en.wikipedia.org/wiki/Propositional_formula

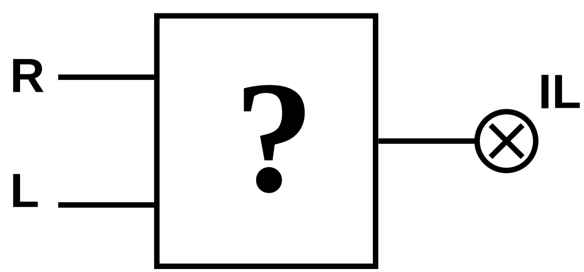
Logic functions and truth tables

- Truth tables
  - Tabular representation of a logic function
  - A truth table has as many columns as input variables has the logic function
  - Each row of the table correspond to a valuation of the logic function
    - The total number of different valuations of a logic function is: 2^{arity}
  - Table organization:
    - Left columns corresponds to Input variable(s)
    - Right column/columns corresponds/correspond to output/outputs

Examples of Truth Tables

Interior light of a car

- Given the input variables ***R***, ***L*** (right and left doors of a 2 doors car) you have to design a logic circuit for turning on or off the interior light (**IL**) of a car.
- Behavior:
  - If at least one door of the car is open the IL is on.
  - If both doors are closed the IL is off.



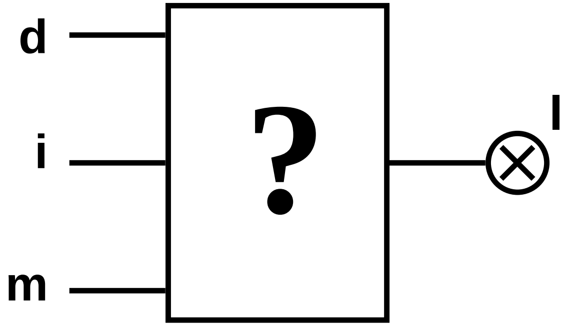
R	L	IL
0	0	0
0	1	1
1	0	1
1	1	1

Examples of Truth Tables

Interior light of a car (version 2)

- Most cars have a special input for turning on or off the interior light independent of the doors state (open or close)
- We add a input variable ***M*** corresponding to the manual turn on/off of the IL.
- Behavior:
  - If input ***M*** is activated (***M=1***) IL is on, even if the doors of the car are closed
  - If input ***M*** is not activated (***M=0***) IL state (on or off) depends on the state of the doors
    - If at least one door of the car is open the IL is on.
    - If both doors are closed the IL is off.

Examples of Truth Tables



m	d	i	l
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

m	d	i	l
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	X	X	1

Reduced truth table

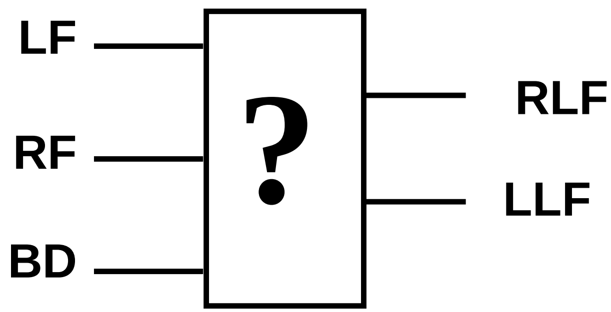
Examples of Truth Tables

- Functions with <u>don’t care inputs</u>
  - Depending on the function, there are valuations for which inputs values do not care when evaluating the value of the output because:
    - The behavior of the circuit is not defined for some combination(s) of the input values
    - The nature of the circuit does not allow such input combination
  - Output value for ***don’t care inputs*** is X

Examples of Truth Tables

Car's light flasher

- Given the control input signals:
  left flasher (<u>**LF**</u>), right flasher (***RF***) and breakdown (***BD***)
- You have to generate the output values that activate the light flashers of a car, Left light flasher (LLF) right light flasher (RLF)



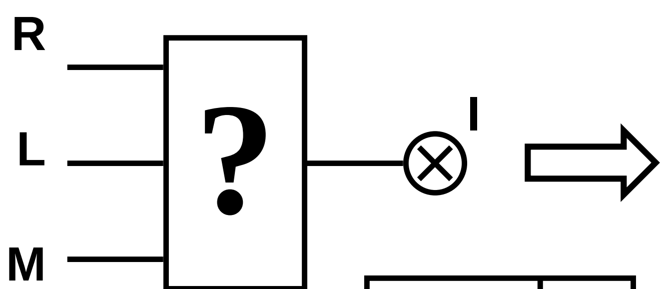
BD	LF	RF	LLF	RLF
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	X	X
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	X	X

Truth Table of a function with don't care inputs

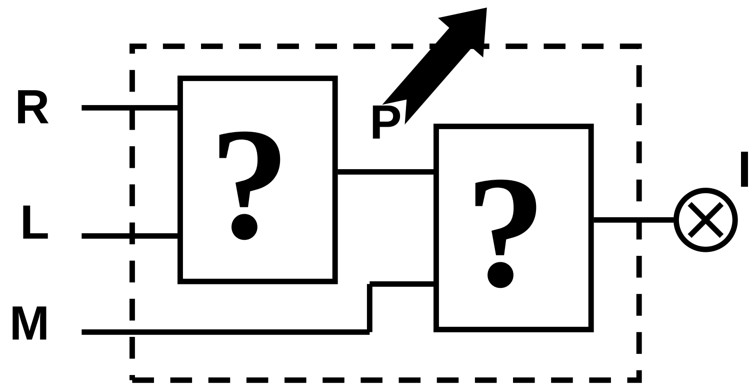
Composition of functions

- Composite function: a function whose values are found from two given functions by applying one function to an independent variable and then applying the second function to the result
- Example: Car’s interior light with manual control

P ≡ “There is one door open”



R	L	P
0	0	0
0	1	1
1	0	1
1	1	1



M	P	I
0	0	0
0	1	1
1	0	1
1	1	1

Logic gates

- Logic gate: electronic circuit that implements a basic logic function

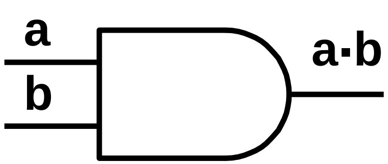
- Types
  - Active-high outputs: AND, OR, NOT, XOR
  - Active-low outputs: NAND, NOR, XNOR

- Technologies
  - TTL, CMOS

Logic gates

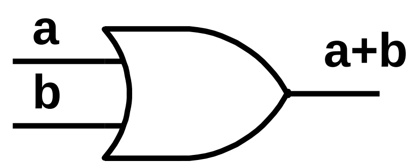
- AND
  - Logical conjunction
  - Arity: 2,3,..

b	a	a·b
0	0	0
0	1	0
1	0	0
1	1	1



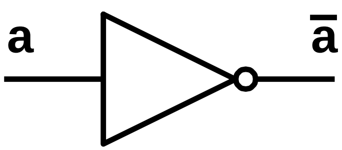
- OR
  - Logical disjunction
  - Arity: 2,3,…

b	a	a+b
0	0	0
0	1	1
1	0	1
1	1	1



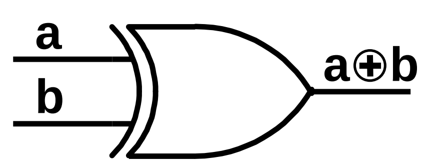
- NOT
  - Logical negation (“inverter”)
  - Arity: 1

a	$\bar{a}$
0	1
1	0



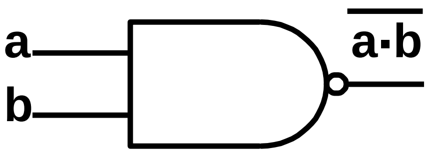
- XOR
  - Exclusive disjunction
  - Arity: 2

b	a	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0



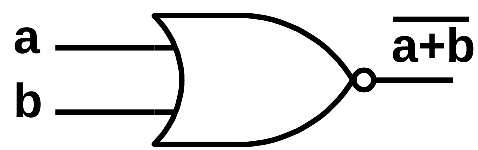
Logic gates with low-level output

- NAND = NOT (AND)
  - Arity: 2,3,…



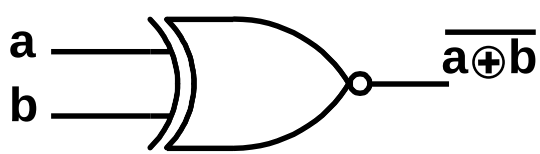
b	a	$\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

- NOR = NOT (OR)
  - Arity: 2,3,…



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

- XNOR = NOT (XOR)
  - Arity: 2

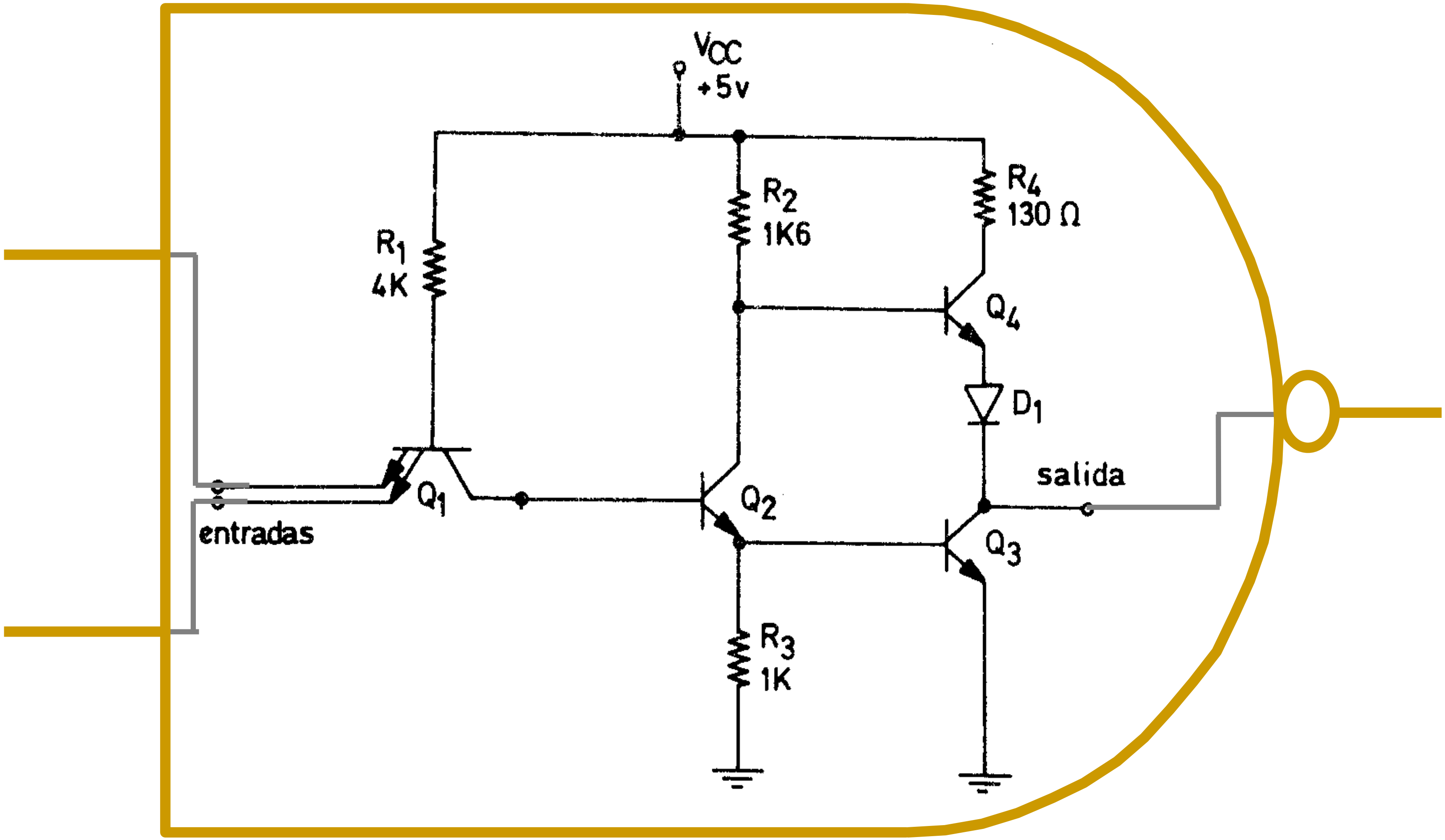


b	a	$\overline{a \oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

Logic gates. Technologies

- Each technology uses different physical elements (transistors) and different power levels to represent logical values 0 and 1
- TTL = Transistor-Transistor Logic
  - Based on bipolar transistors
  - High speed, high power consumption, hard integration
- CMOS = Complementary Metal Oxide Semiconductor
  - Based on MOSFET transistors
  - Speed slower than bipolar, low power consumption, large scale of integration

NAND (TTL) Physical Implementation FCO



DISCA

Logic gates: Technologies

National Semiconductor

54LS00/DM54LS00/DM74LS00
Quad 2-Input NAND Gates

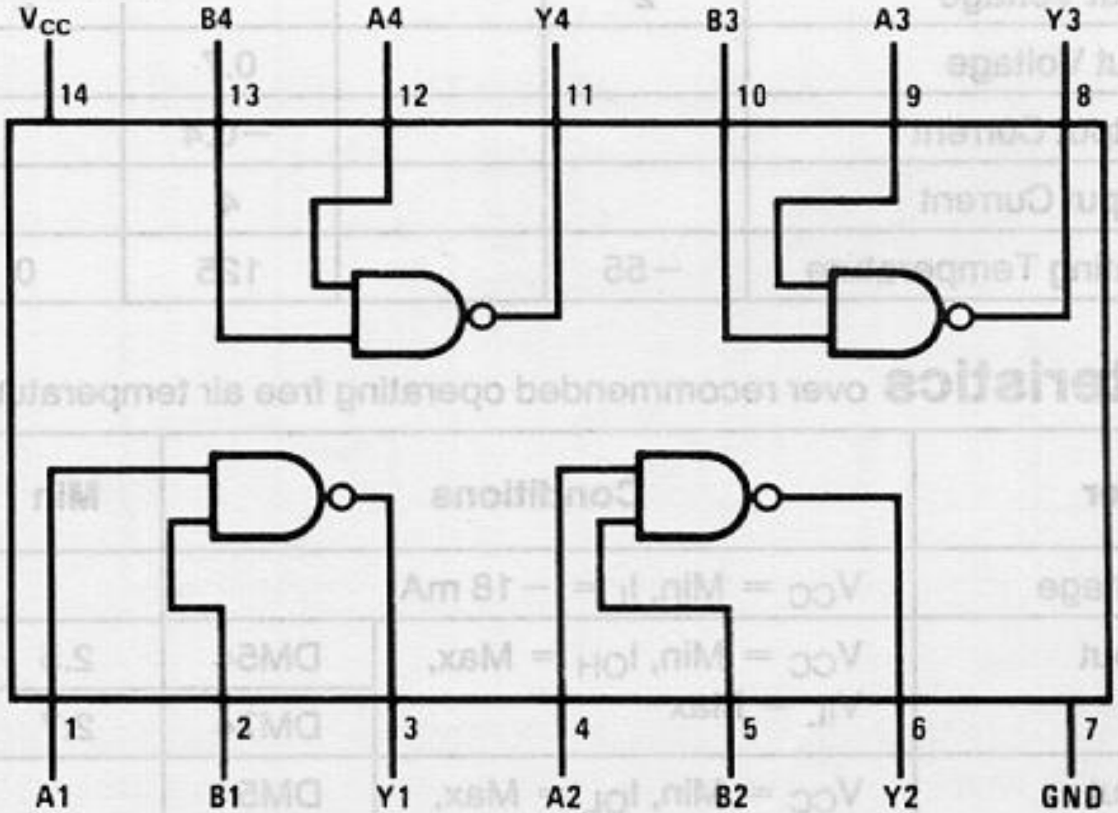
Function Table

$$Y = \overline{AB}$$

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

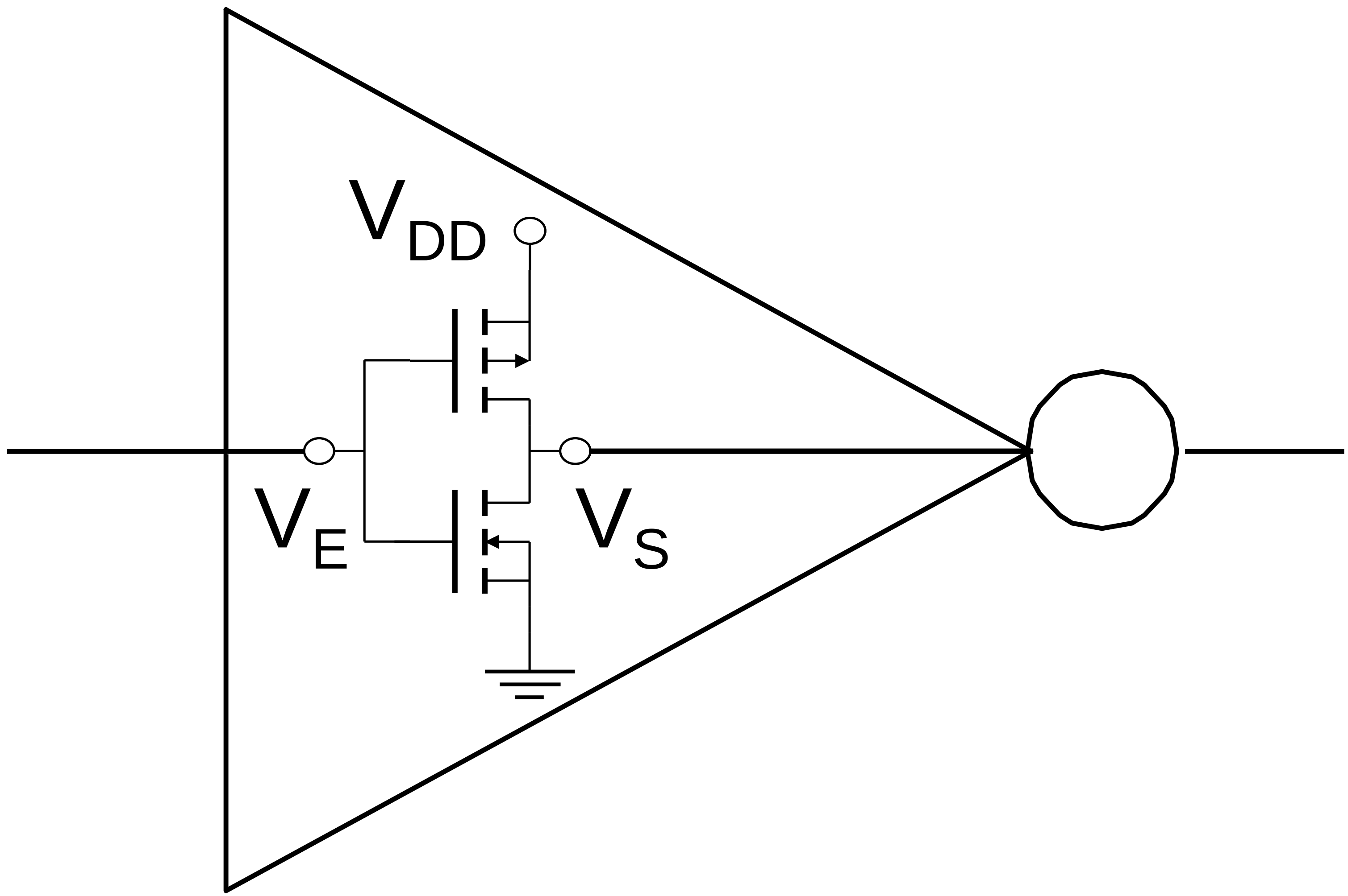
H = High Logic Level
L = Low Logic Level

Dual-In-Line Package



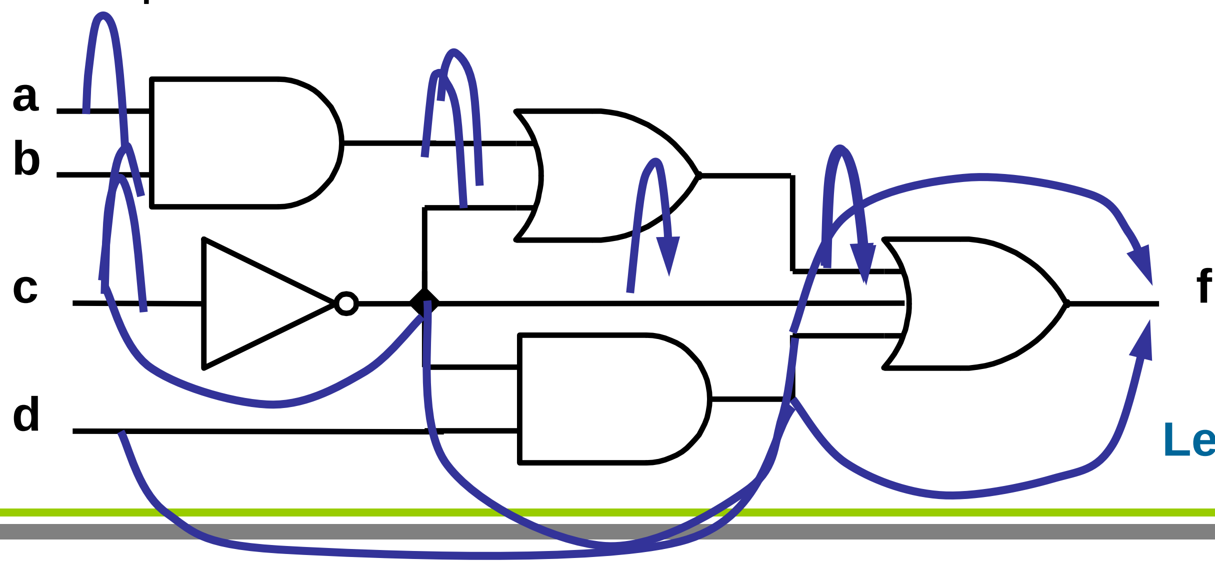
DISCA

Inverter implemented using CMOS Tech. FCO



Level of a logic circuit

- Level
  - Number of gates that a signal need to go through before it can get out from the circuitry of the logic circuit.
    - If there are many paths to go through a circuit It is considered the worst case
  - It is a figure-of-merit of the circuit delay
  - Each gate has a delay of T
  - Inputs are at level 0



through 2 gates
through 3 gates
through 2 gates
through 3 gates
through 3 gates

Level= worst case = 3 gates

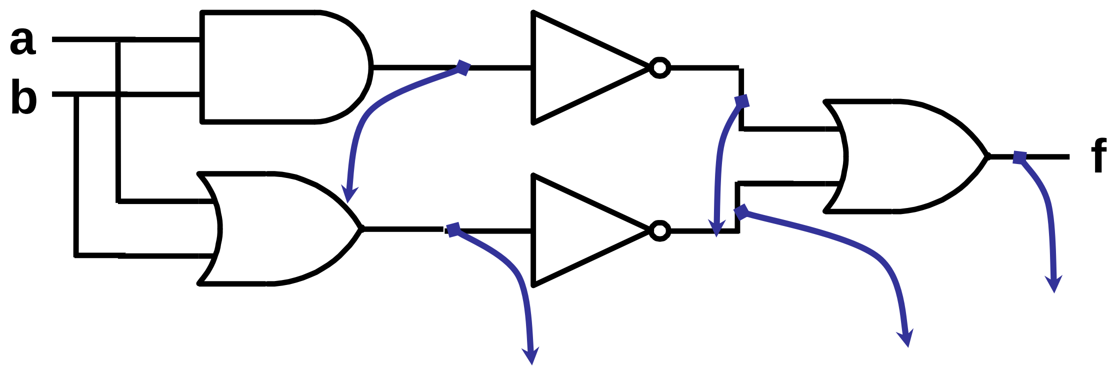
DISCA

Circuit Analysis

- Given a logic circuit it is necessary to obtain its logic function and its truth table
  - Logic function: It is obtained doing the composition of the sub functions of each node of the circuit
  - Truth table: obtained evaluating the output for each possible combination of the values that the inputs can take

Circuit Analysis

- Example



Logic Function

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

b	a	f
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

Boole's Algebra

- George Boole (s. XIX)
  - English mathematician and philosopher
  - Boole developed an algebraic structure based on 2 values (true, false) and 2 composition laws (and, or)
  - Boole's studies have been used to formalize the rules of logical reasoning
- Claude Shannon (1938, Bell Labs.)
  - Adapted boole's algebra to computer science
    - Values 0 y 1, composition laws AND y OR
  - Shannon studies have been used to formalize the rules of logic circuit design

Precedence (if there is no parenthesis)

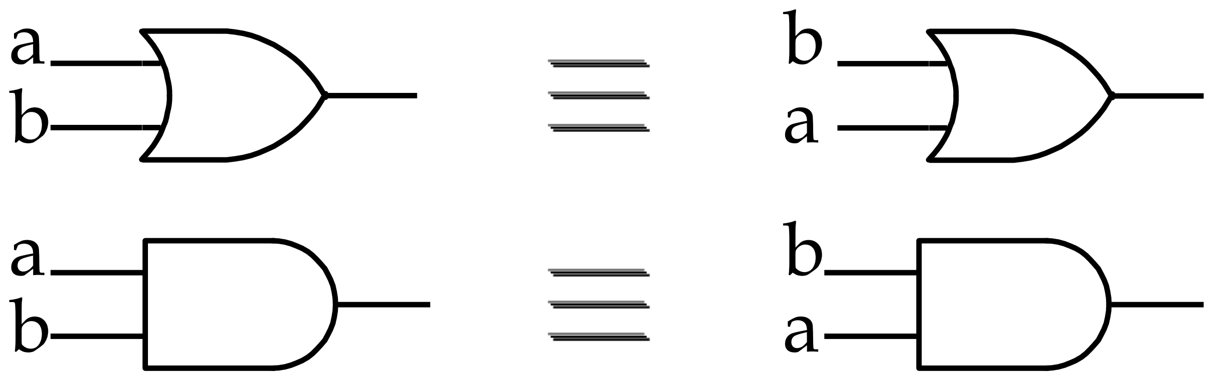
Puerta lógica	Símbolo estándar
AND	·
OR	+
NOT	—
XOR	⊕

Axioms of the Boole's Algebra

- Commutativity of addition and multiplication

$$a + b = b + a$$

$$a \cdot b = b \cdot a$$



- Distributivity

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

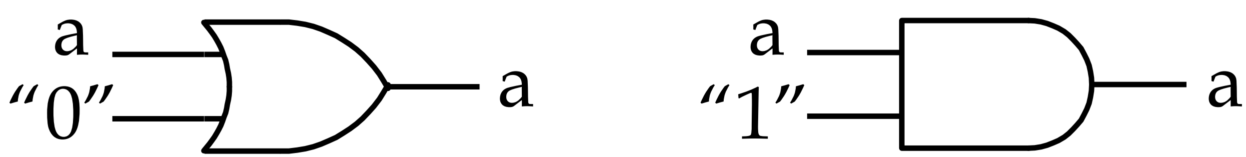
$$(a \cdot b) + (a \cdot c) = a \cdot (b + c)$$

Axioms of the Boole's Algebra

- Neutral elements

$$a + 0 = a$$

$$a \cdot 1 = a$$



- Complements

$$a + \overline{a} = 1$$

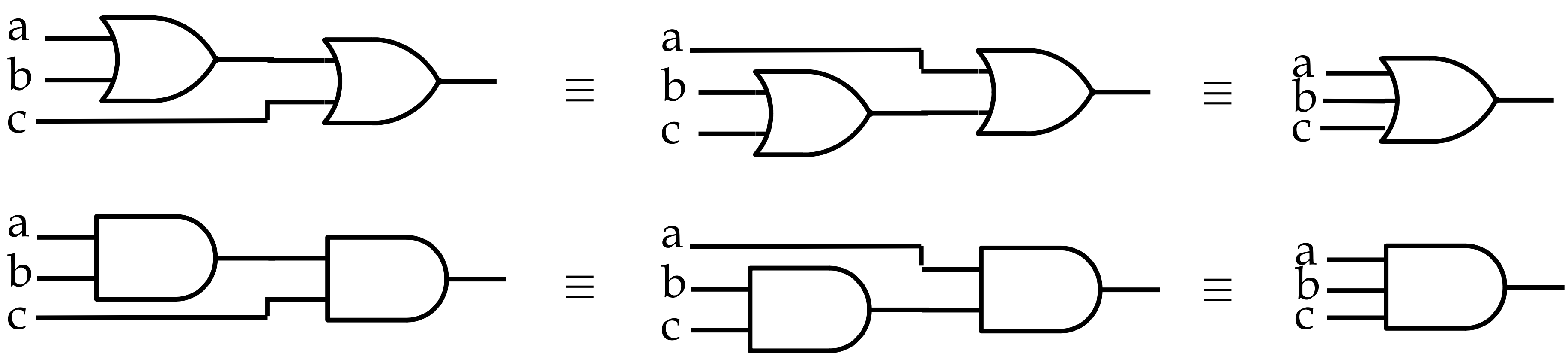
$$a \cdot \overline{a} = 0$$

Properties of the Boole's Algebra

- Associativity

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

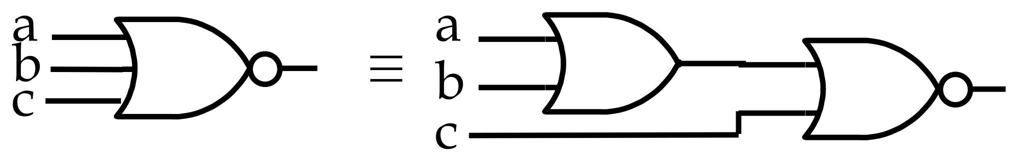
  - Associativity allows to increment the arity of gates starting from gates with small arity

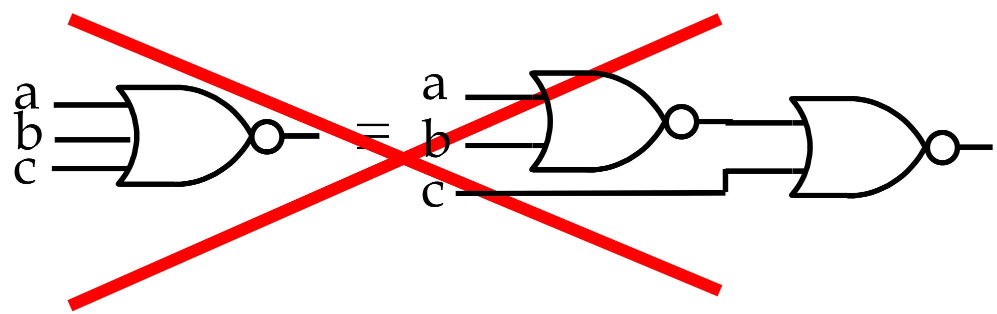


Properties of the Boole's Algebra

- Associativity
  - Be careful with gates with output active at low level

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

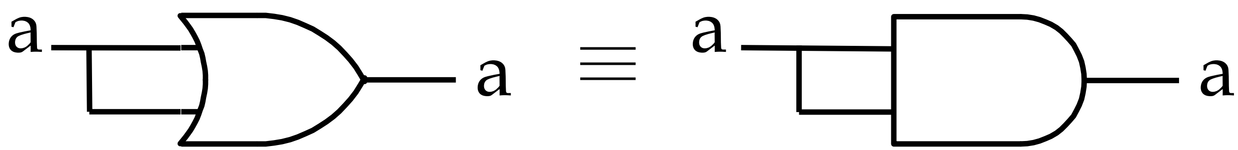




Properties of the Boole's Algebra

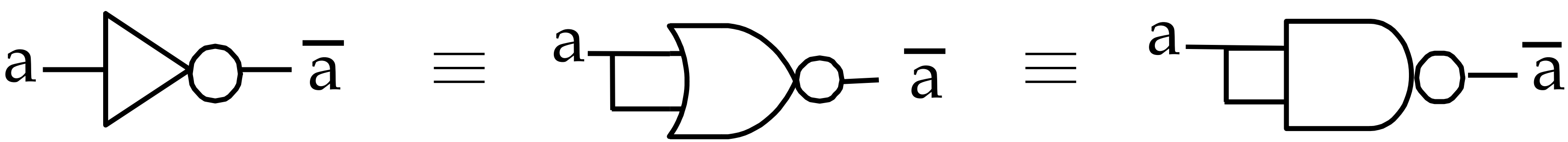
- Idempotence

$$a + a = a \qquad a \cdot a = a$$



  - Idempotence allows to build not gates from NAND gates or NOR gates

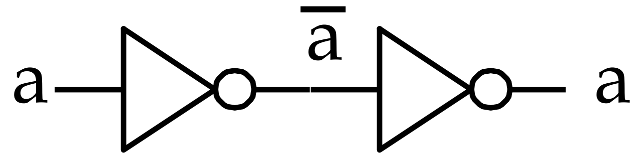
$$\overline{a + a} = \overline{a} = \overline{a \cdot a}$$



Properties of the Boole’s Algebra

- Involution

$$\overline{\overline{a}} = a$$



- De Morgan’s Laws

$$\overline{(a + b + \ldots + n)} = \overline{a} \cdot \overline{b} \cdot \ldots \cdot \overline{n}$$

$$\overline{(a \cdot b \cdot \ldots \cdot n)} = \overline{a} + \overline{b} + \ldots + \overline{n}$$

  – ¡Be careful!

$$\cancel{\overline{(a + b)} = \overline{a} + \overline{b}} \qquad \overline{(a + b)} = \overline{a} \cdot \overline{b}$$

Canonical forms in Boole's Algebra

- Canonical form (noun): the simplest form of something
- Any Boolean function can be expressed in a **canonical form** using the dual concepts of *minterms* and *maxterms*.

Canonical forms: Minterm of order n

- For a boolean function of *n* variables , a product term in which each of the *n* variables appears **once** (in either its complemented or uncomplemented form) is called a *minterm*.
- A minterm is a logical expression of *n* variables that employs only the *complement* operator and the *conjunction* operator.
- Minterm of order n
  - A product term in which each of the n input variables appears once
  - A variable appears complemented if its value is 0
  - Each valuation provides a different minterm
  - Minterms are numbered

Canonical forms: Maxterm of order n

- For a boolean function of *n* variables, a sum term in which each of the *n* variables appears **once** (in either its complemented or uncomplemented form) is called a *maxterm*.
- A maxterm is a logical expression of *n* variables that employs only the *complement* operator and the *disjunction* operator.
- Maxterm of order n
  - A sum term in which each of the n input variables appears once
  - A variable appears complemented if its value is 1
  - Each valuation provides a different maxterm
  - Minterms are numbered

Disjunctive canonical form

- Disjunctive canonical form or sum of products (SoP)
  - Sum of the minterms pertaining to the function
  - Pertains to the function those minterms which valuations are equal to 1

$$\sum_{list\,of\,the\,input\,variables}\left(numbered\ list\ of\ the\ function's\min\ terms\right)$$

b	a	f	minterm	nº
0	0	0	$\overline{b} \cdot \overline{a}$	0
0	1	1	$\overline{b} \cdot a$	1
1	0	0	$b \cdot \overline{a}$	2
1	1	1	$b \cdot a$	3

Canonical form

$$f = \sum_{b,a}(1,3) = \overline{b} \cdot a + b \cdot a$$

Equivalent algebraic expression

Conjunctive canonical form

- Conjunctive canonical form or product of sums (PoS)
  - Product of the maxterms pertaining to the function
  - Pertains to the function those maxterms which valuations are equal to 0

$$\prod_{list\ of\ the\ input\ variables} \left(numbered\ list\ of\ the\ function's\ max\ terms \right)$$

b	a	f	maxterm	nº
0	0	0	$b + a$	0
0	1	1	$b + \overline{a}$	1
1	0	0	$\overline{b} + a$	2
1	1	1	$\overline{b} + \overline{a}$	3

Canonical form

$$f = \prod_{b,a}(0,2) = (b + a)\cdot(\overline{b} + a)$$

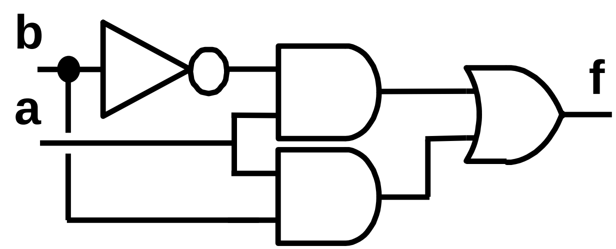
Equivalent algebraic expression

Why are interesting the canonical forms? FCO

- It is a mathematical expression unique and compact of a logic function
- It is the first aproximation to the synthesis of a circuit starting from a truth table

b	a	f
0	0	0
0	1	1
1	0	0
1	1	1

$$f = \sum_{b,a}(1,3) = \bar{b}\cdot a + b\cdot a$$



- Any logical function can be implemented by a circuit of level ≤ 3

Canonical forms with don't care inputs FCO

- Canonical forms for functions with don't care inputs
  - The valuations are grouped separately as follows:

	a	pi	pd	id
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	X
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	X

$$id = \sum_{a,\, pi,\, pd} (1, 4, 5, 6) + \sum_{\phi} (3, 7)$$

$$id = \prod_{a,\, pi,\, pd} (0, 2) \bullet \prod_{\phi} (3, 7)$$

Functions simplification

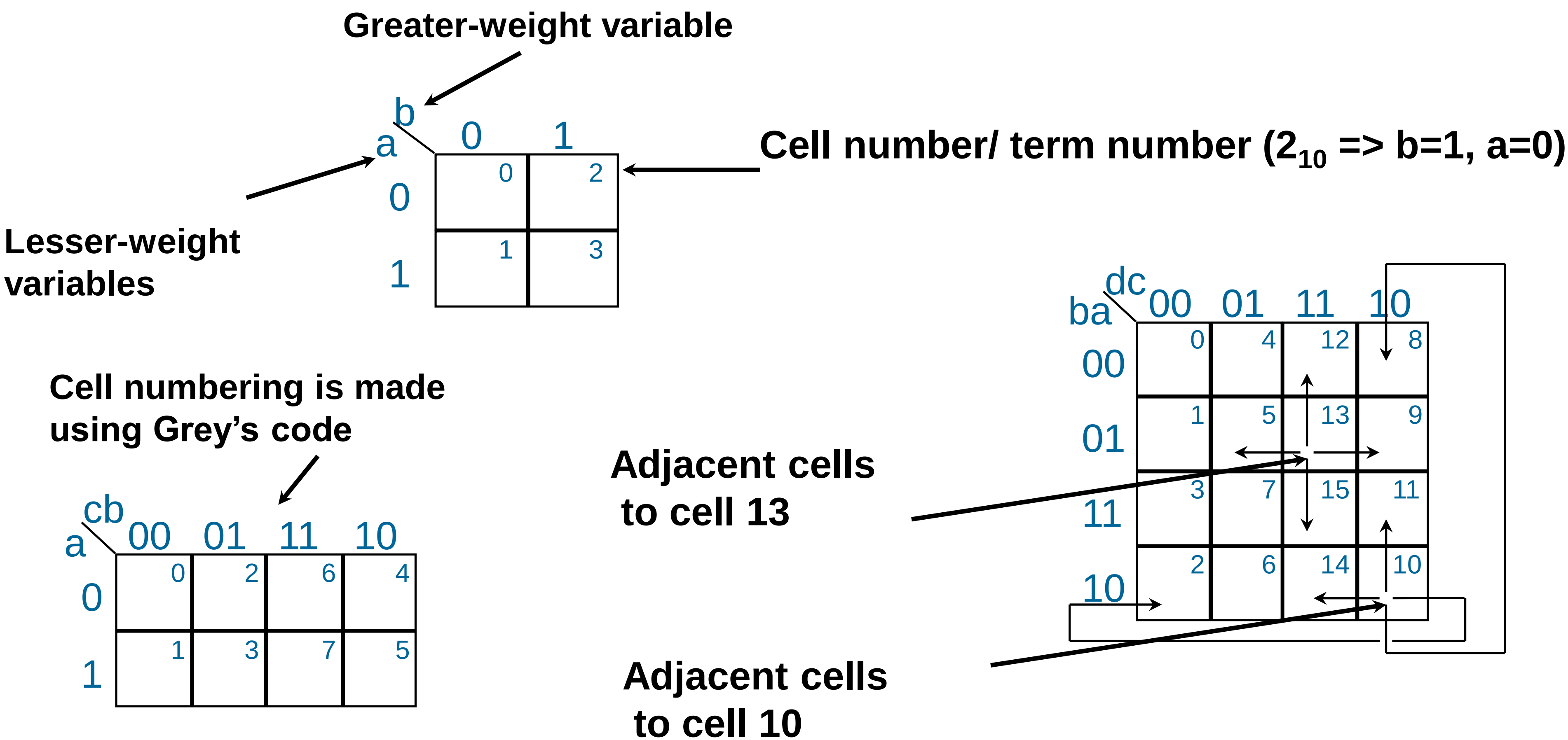
- Function Simplification
  - Get an algebraic expression equivalent to the starting expression but smaller (less valuations which implies terms with less variables)
  - Aim: reduce the complexity of a circuit that implements a function
- Methodology
  - Algebraic. Applying axioms and properties of Boole’s Algebra
    - Complementary element, neutral element, distributivity and associativity
  - Graphical. Karnaugh’s map

Karnaugh’s simplification

- Karnaugh’s map
  - Matrix representation of a Truth table
  - A cell of the map represents a valuation of the Truth table
  - Inside a cell it is written the output of a valuation of the Truth table
  - The spatial distribution of the cells is made in such a way that adjacent terms of the valuations are written in adjacent cells
  - Two terms are said adjacent if their valiuations differn only in just one variable
  - The corners of the karnaugh’s map are adjacent

Karnaugh's simplification

- Maps for functions with 2, 3 and 4 variables

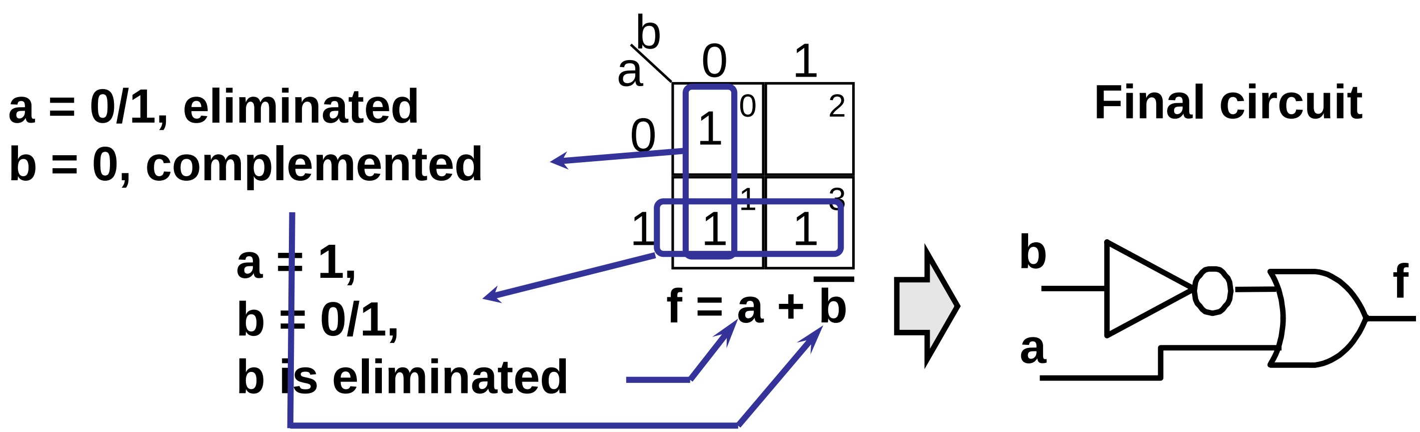


Karnaugh's simplification

- Method
  - Adjacent cells with the same value are grouped
  - The number of adjacent cells must be a multiple of 2^n
  - The groups should be the biggest possible
  - The total number of groups created should be the minimal one possible
  - A cell can be in several different groups

Ones-simplification's Method

- Adjacent cells with value one are grouped
- Each group represents a product term (It is not a minterm. Why?).
- Variables with value 0 will apper in its complemented form
- A group of 2^k cell cancel k variables in the resulting expression, in other words: the resulting expression will contains n-k variables
- When obtaining the simplified expression, the variables whose value changes are not included



Ones-simplification

– Each cell with value one represents a minterm of the function

a \ cb	00	01	11	10
0	0^{0}	1^{2}	1^{6}	0^{4}
1	1^{1}	0^{3}	0^{7}	0^{5}

Minterm 2 (cba=010)

Minterm 6 (cba=110)

Minterm 1 (cba=001)

– The function without simplification includes all the minterms

$$f = \sum_{c,b,a}(1, 2, 6) = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot b \cdot \bar{c}$$

Ones-simplification

– Adjacent cells-groups detect minterms with a common value

a \ cb	00	01	11	10
0	0 [0]	1 [2]	1 [6]	0 [4]
1	1 [1]	0 [3]	0 [7]	0 [5]

Minterm 2 (cba=010) $\longrightarrow \bar{c}\cdot b\cdot\bar{a}$

Minterm 6 (cba=110) $\longrightarrow c\cdot b\cdot\bar{a}$

– Its addition can be simplified. In algebraic form

$$\bar{c}\cdot b\cdot\bar{a}+c\cdot b\cdot\bar{a} \overset{\text{ASOCIATIVA}}{=} \bar{c}\cdot(b\cdot\bar{a})+c\cdot(b\cdot\bar{a}) \overset{\text{DISTRIBUTIVA}}{=} (\bar{c}+c)\cdot(b\cdot\bar{a}) \overset{\text{ELEM. COMPLEM.}}{=} 1\cdot(b\cdot\bar{a}) \overset{\text{ELEM. NEUTRO}}{=} \boxed{b\cdot\bar{a}}$$

– Karnaugh gets the same result:

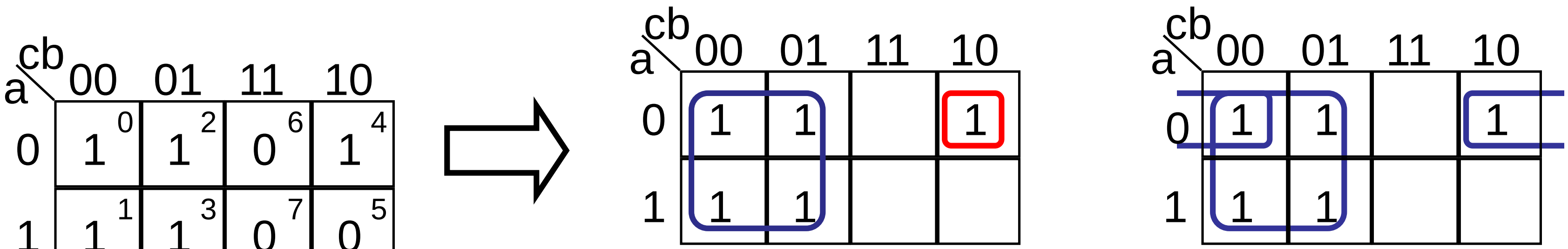
Cell number 2 (cba=010)
Cell number 6 (cba=110)

c = 0/1, eliminated
b = 1, included
a = 0, complemented

$\boxed{b\cdot\bar{a}}$

Ones-simplification

- Examples

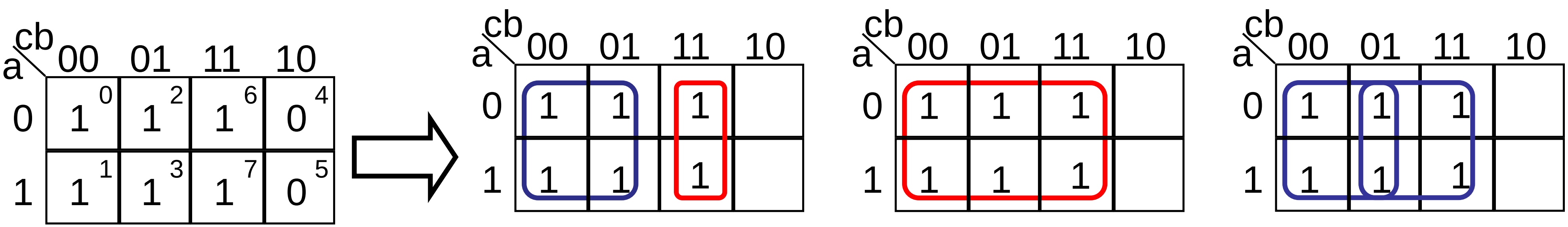


$f = \overline{c} + c \cdot \overline{a} \cdot \overline{b}$

Wrong

$f = \overline{c} + \overline{a} \cdot \overline{b}$

Good



$f = \overline{c} + c \cdot b$

Wrong

$f = ?$

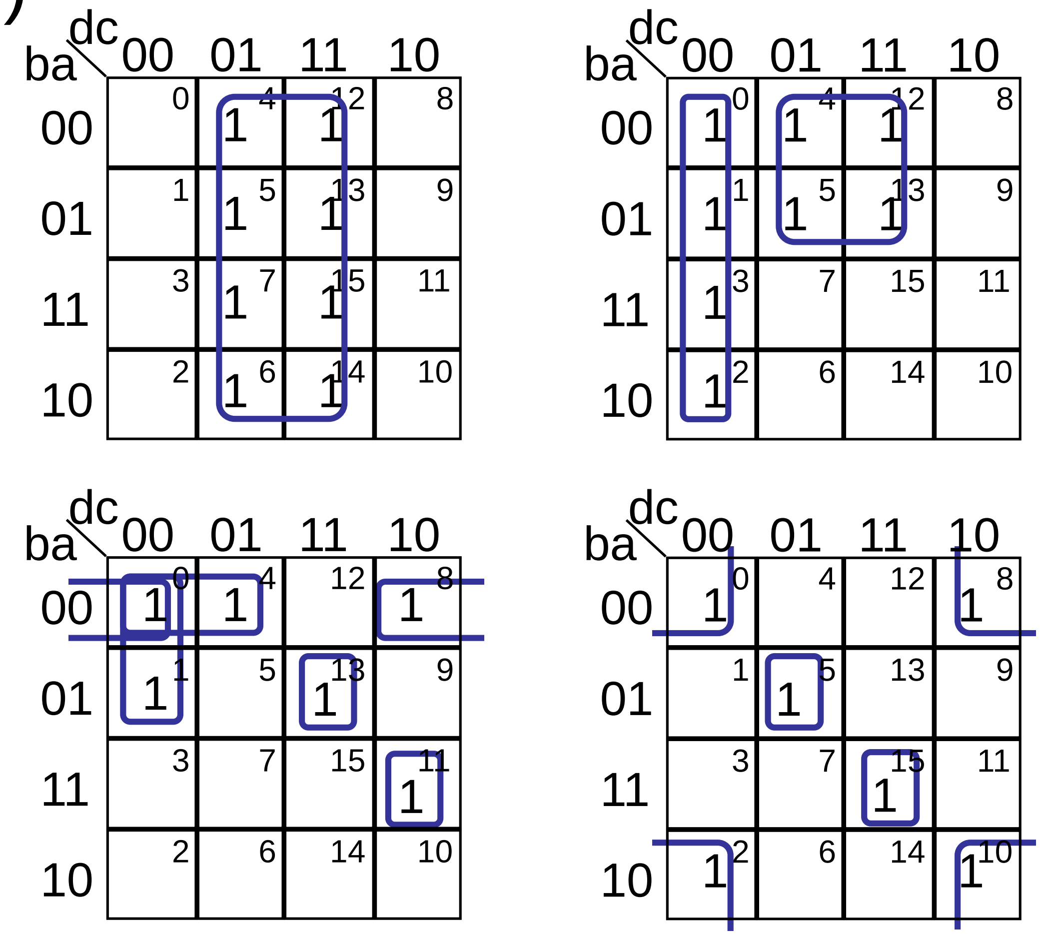
Wrong

$f = \overline{c} + b$

Good

Ones-simplification

- Examples (cont.)



Zeros-simplification

- Group cells with value 0
- Each group represents a sum term (It is not a maxterm because it does not contain all input variables). Variables with value 1 will appear in its complemented form
- A group of 2^k cells cancels k variables from the resultant term and the term will have n-k variables
- When a group is formed, variables with different value (inside the group) are eliminated

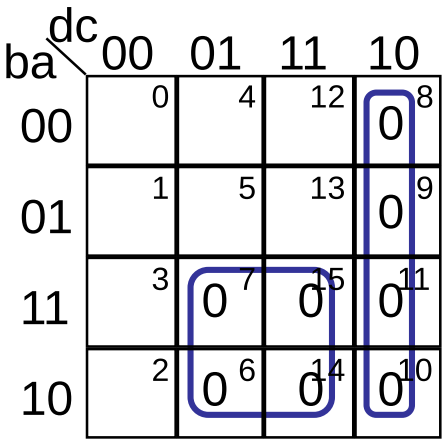
a \ cb	00	01	11	10
0	1^{0}	1^{2}	1^{6}	$\boxed{0}^{4}$
1	1^{1}	1^{3}	1^{7}	$\boxed{0}^{5}$

a = 0/1, eliminated
b = 0, included
c = 1, complemented

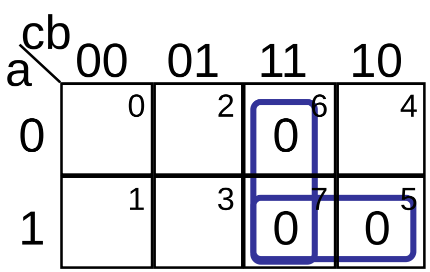
f = $\overline{c}$ + b

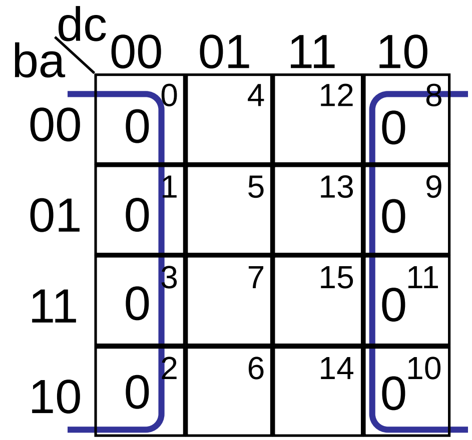
Zeros-simplification

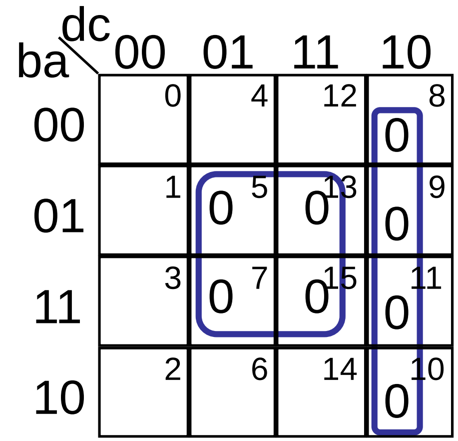
- Examples

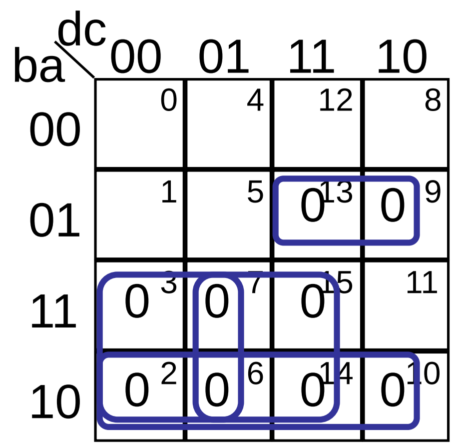


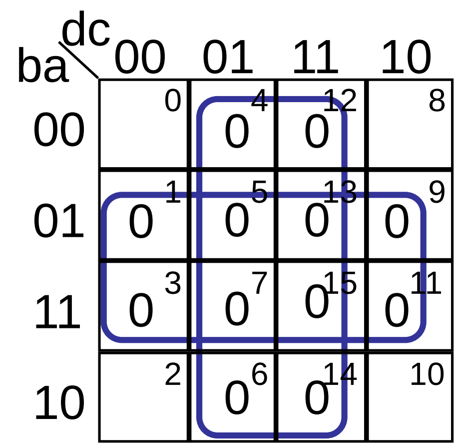
$$f = (\overline{d} + c) \cdot (\overline{c} + \overline{b})$$











Simplification. Don’t care inputs

– Cell with X value are very important. They can be grouped with 0’s or with 1’s

d	c	b	a	f
0	0	0	0	0
0	0	0	1	x
0	0	1	0	0
0	0	1	1	x
0	1	0	0	0
0	1	0	1	x
0	1	1	0	x
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	1
1	1	0	0	x
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$$f = \sum_{d,c,b,a}(7,11,13,14,15) + \sum_{\phi}(1,3,5,6,10,12) =$$

$$\prod_{d,c,b,a}(0,1,2,4,8,9) \bullet \prod_{\phi}(1,3,5,6,10,12)$$

ba \ dc	00	01	11	10
00	0^{0}	0^{4}	x^{12}	0^{8}
01	x^{1}	x^{5}	1^{13}	0^{9}
11	x^{3}	1^{7}	1^{15}	1^{11}
10	0^{2}	x^{6}	1^{14}	x^{10}

“Ones”

ba \ dc	00	01	11	10
00	0^{0}	0^{4}	x^{12}	0^{8}
01	x^{1}	x^{5}	1^{13}	0^{9}
11	x^{3}	1^{7}	1^{15}	1^{11}
10	0^{2}	x^{6}	1^{14}	x^{10}

“Zeros”

Simplification. Don’t care inputs

- Common mistakes:
  - Using all “x”

ba \ dc	00	01	11	10
00	0 (0)	0 (4)	x (12)	0 (8)
01	x (1)	x (5)	1 (13)	0 (9)
11	x (3)	1 (7)	1 (15)	1 (11)
10	0 (2)	x (6)	1 (14)	x (10)

Wrong

ba \ dc	00	01	11	10
00	0 (0)	0 (4)	x (12)	0 (8)
01	x (1)	x (5)	1 (13)	0 (9)
11	x (3)	1 (7)	1 (15)	1 (11)
10	0 (2)	x (6)	1 (14)	x (10)

Wrong

  - Grouping ”x” when not needed

ba \ dc	00	01	11	10
00	0 (0)	0 (4)	x (12)	0 (8)
01	x (1)	x (5)	1 (13)	0 (9)
11	x (3)	1 (7)	1 (15)	1 (11)
10	0 (2)	x (6)	1 (14)	x (10)

Wrong

ba \ dc	00	01	11	10
00	0 (0)	0 (4)	x (12)	0 (8)
01	x (1)	x (5)	1 (13)	0 (9)
11	x (3)	1 (7)	1 (15)	1 (11)
10	0 (2)	x (6)	1 (14)	x (10)

Wrong

Computers Fundamentals

Subject 1. Introduction to computers