Unit 3. 2
Combinational circuits

Contents

- Combinational circuits: concept, analysis and synthesis.
- Methods to simplify logical functions.
- Basic combinational circuits;
  - Multiplexers
  - Demultiplexers
  - Decoders
  - Encoders
  - Comparators

Bibliography

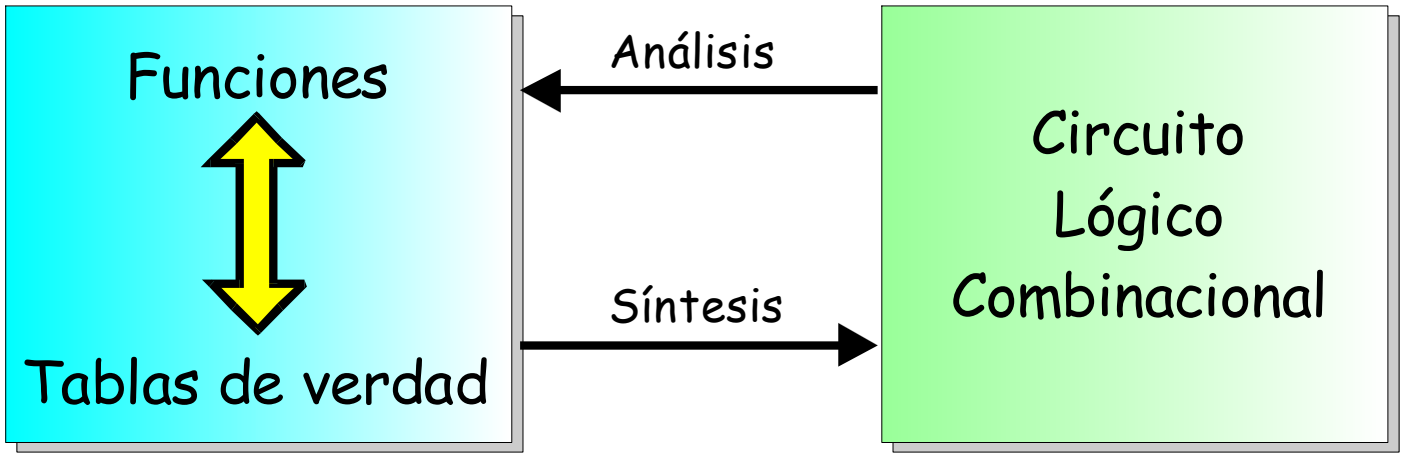
• Digital fundamentals.
Thomas Floyd. Prentice-Hall.

• Digital Design.
M. Morris Mano. Prentice-Hall

- Introduction to Digital Logic Design.
John P. Hayes. Addison-Wesley

Combinational Circuits (I)

- **Concept:** Circuits in which the outputs in a concrete instant are function exclusively of the inputs in that instant, i.e., they don't store information.



- **Analysis:** From the digital circuit, obtain the mathematical function that operates with the inputs to obtain the outputs.
- **Synthesis**: From the truth table or the function, implement the circuit.

Synthesis of combinational circuits

- From the problem statement obtain the **truth table**.
- From the truth table, obtain the **canonical function** (minterms or maxterms)
- **Simplify** the canonical function (via Boolean Algebra or Karnaugh maps).
- **Design the circuit** with the logical gates that correspond to the logical operations in the simplified function.

Methods to simplify logical functions (I)

Boolean Algebra

- Apply Boolean postulates and rules to eliminate variables and obtain a simpler function. Example:

$$f_1(d,c,b,a) = d\cdot c\cdot b\cdot a + d\cdot c\cdot b\cdot \overline{a} = d\cdot c\cdot b\cdot (a+\overline{a}) = d\cdot c\cdot b\cdot 1 = d\cdot c\cdot b$$

$$f_2(d,c,b,a) = (d+c+b+a)\cdot (d+c+\overline{b}+a) = (d+c+b\cdot \overline{b}+a) = (d+c+a)$$

- Caveats: Depending on how the original function is expressed, the difficulty of this method can vary considerably; and sometimes it is not obvious when the simpler expression is obtained.

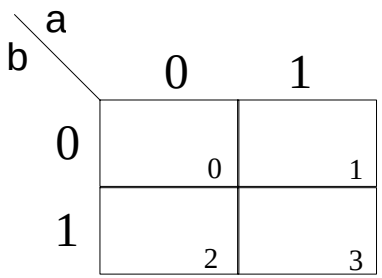
Methods to simplify logical functions (II)

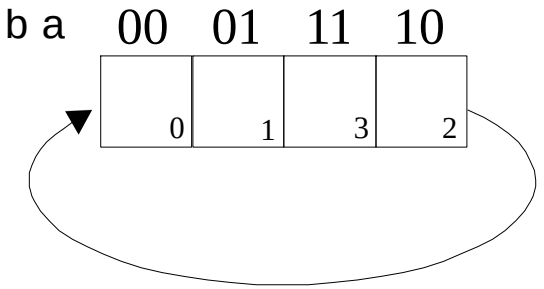
Karnaugh maps

- Algorithmic method based on extracting common factors systematically arranging the truth table in a special way called Karnaugh maps.
- Each map contains cells distributed in such a way that two adjacent cells represent canonical terms that differ just in one variable (their binary representation differs just in one bit)
- Common factor can be extarcted among two of this terms and by means of distributive property, a variable is eliminated.

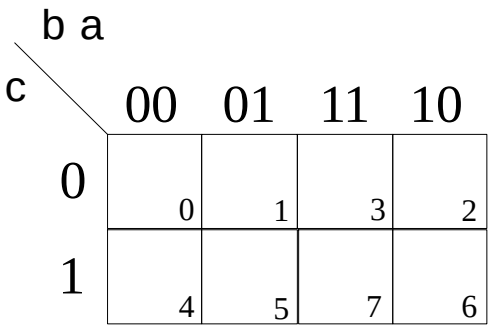
Methods to simplify logical functions (III)

- Karnaugh map for **two variable'** functions

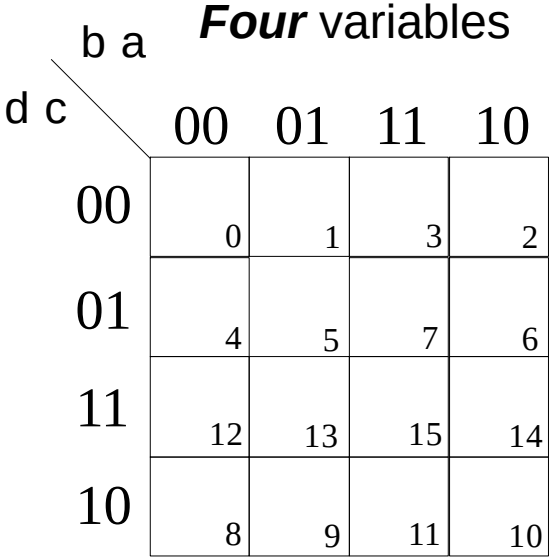




Three variables



Four variables



Methods to simplify logical functions (IV)

Simplification procedure with Karnaugh maps

1.- Draw the right map (2, 3 or 4 variables).

2.- Write “1” in the corresponding cell for minterms or (“0”) for maxterms

3.- Group with a close curve the maximum number of adjacent elements (always a power of 2 ⇨2, 4, 8, 16). Repeat until no element left .

4.- Obtain a simplified term for each group:

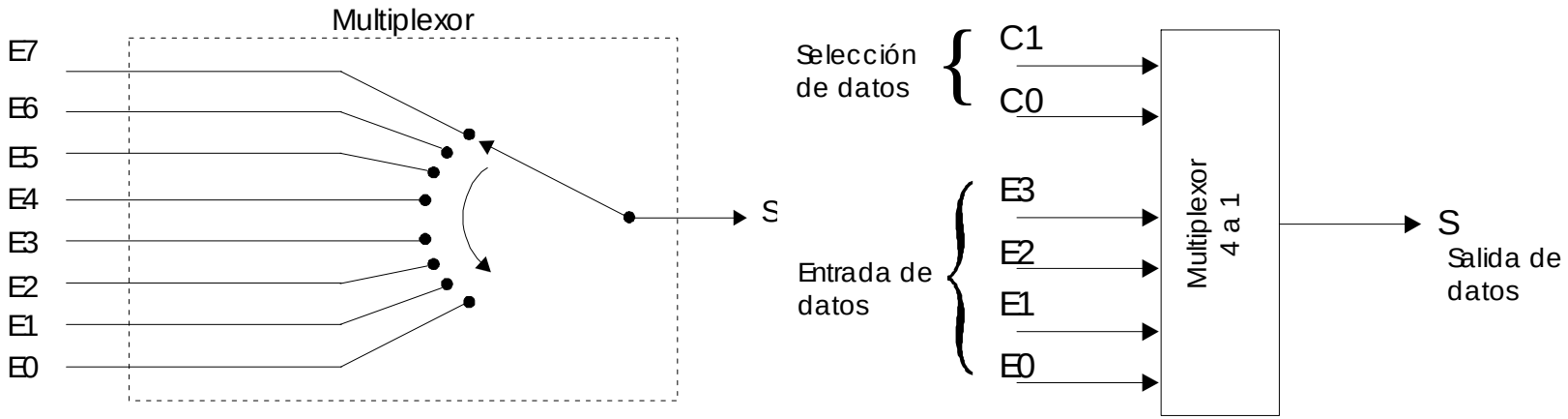
In each group, a variable whose value is “0” in half of the cells and “1” in the other half, disappears.

For minterms, remaining variables are taken as natural (not complemented) if they are “1” and complemented if they are “0” (the opposite for Maxterms)

If any bit has no adjacent cells it will yield a canonical term

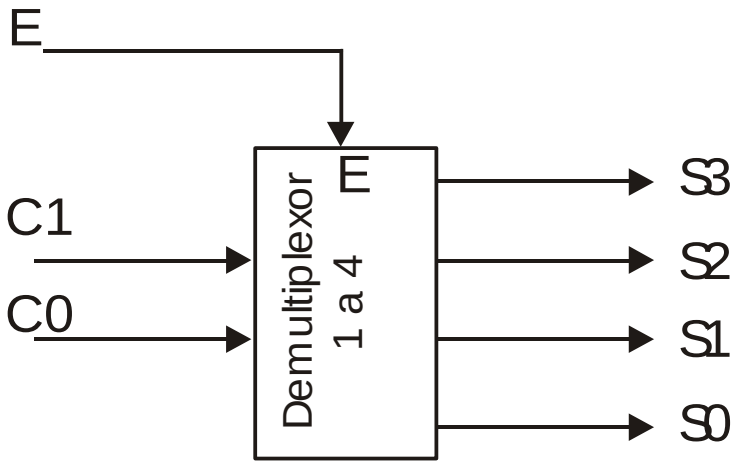
Basic combinational circuits (I)
Multiplexers

- Circuit with 2^n inputs, just one output and *n* selection inputs which select which input are transmitted to the output.



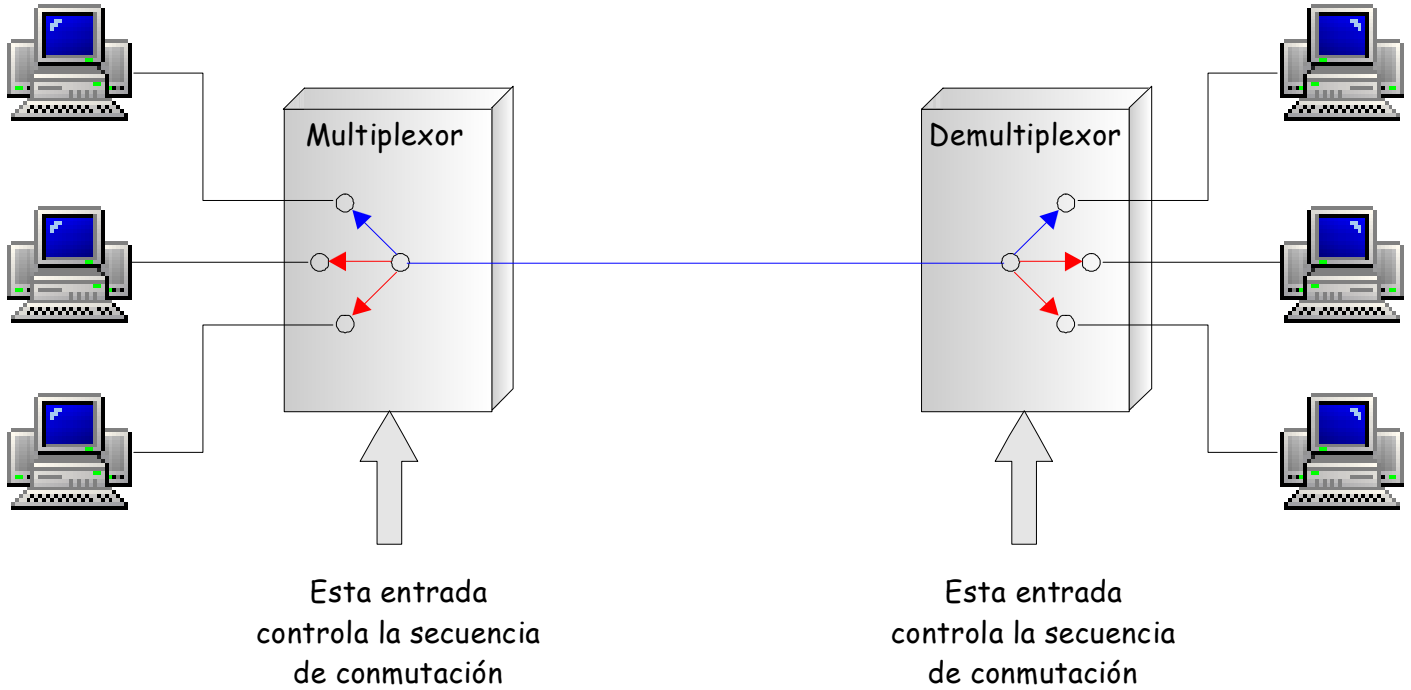
Basic combinational circuits (II)
Demultiplexers

- They perform the inverse operation: one input, 2^n outputs and *n* selection input which determines to which output the input is transmitted.



Basic combinational circuits (III)
Multiplexer/Demultiplexers

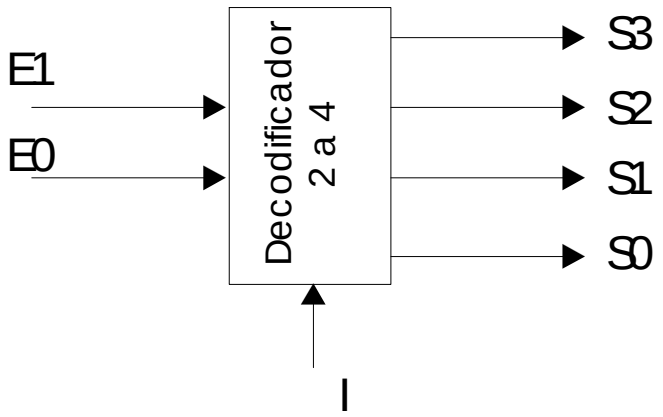
basic application



Basic combinational circuits (IV)
Decoders (1)

- Circuits with *n* inputs that codify a binary number that specify which of the 2^n outputs will be active

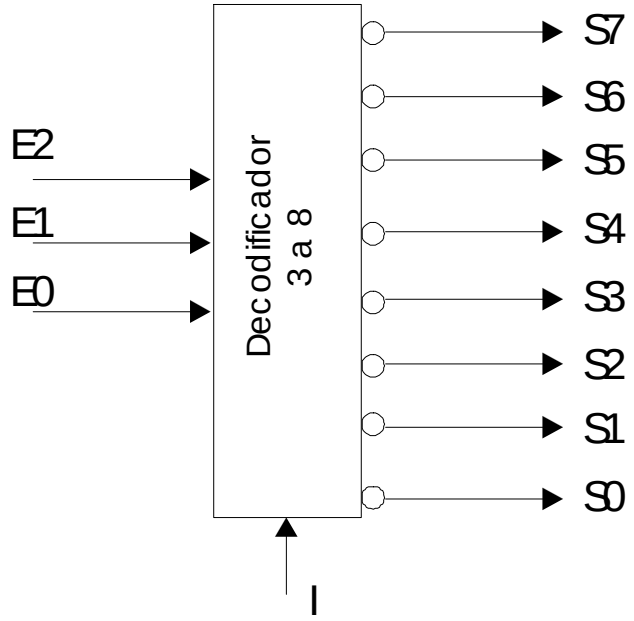
2 to 4 Decoder



I	E1	E0	S3	S2	S1	S0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	X	X	0	0	0	0

Basic combinational circuits (V)
Decoders (2)

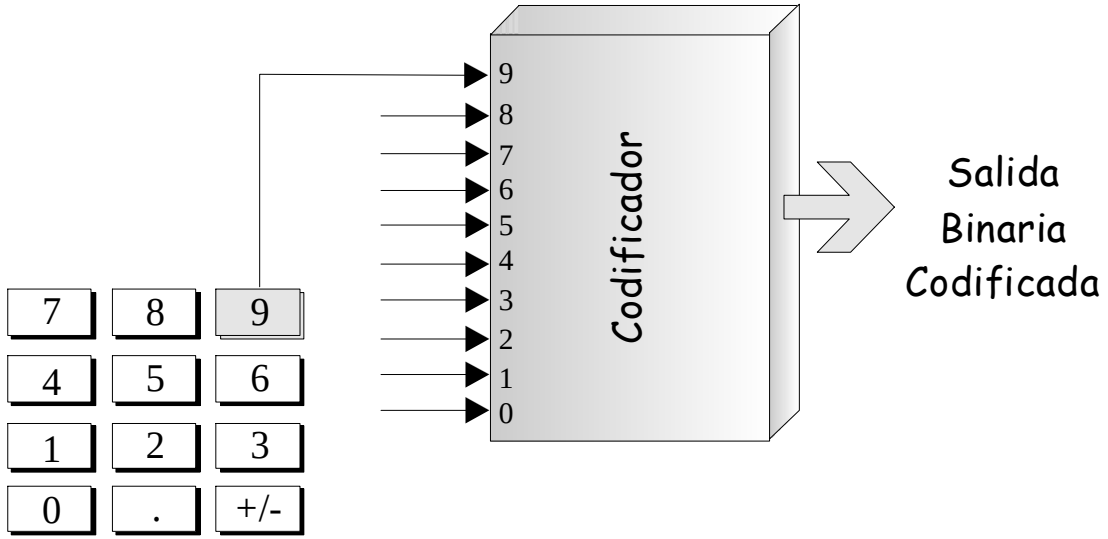
3 to 8 decoder



I	E2	E1	E0	S7	S6	S5	S4	S3	S2	S1	S0
0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	1	1
1	X	X	X	1	1	1	1	1	1	1	1

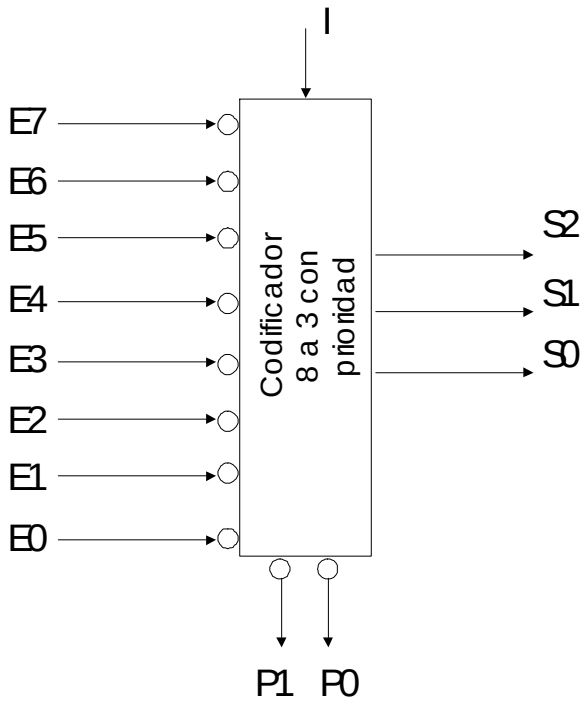
Basic combinational circuits (VI)
Encoder (1)

- They perform the inverse operation*: N* inputs and *n*-bit output representing in binary which input is active ($N \leq 2^n$). A priority among the inputs can be established



Basic combinational circuits (VII)
Encoder (2)

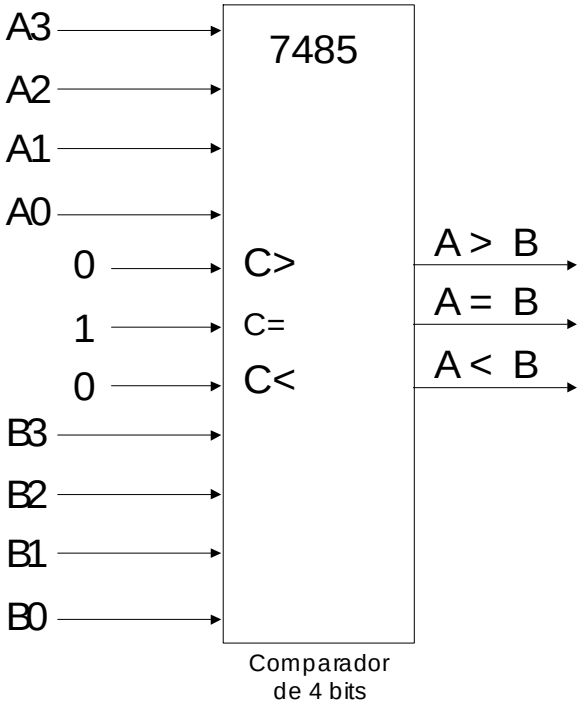
- **8 to 3 encoder with priority**



I	E7	E6	E5	E4	E3	E2	E1	E0	S2	S1	S0	P1	P0
1	X	X	X	X	X	X	X	X	0	0	0	1	1
0	1	1	1	1	1	1	1	1	0	0	0	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

Basic combinational circuits (VIII)
Comparators

- Compares two *n*-bit numbers, A and B, and determines which is bigger or whether they are equal. Generally is has three outputs: A>B, A=B y A<B



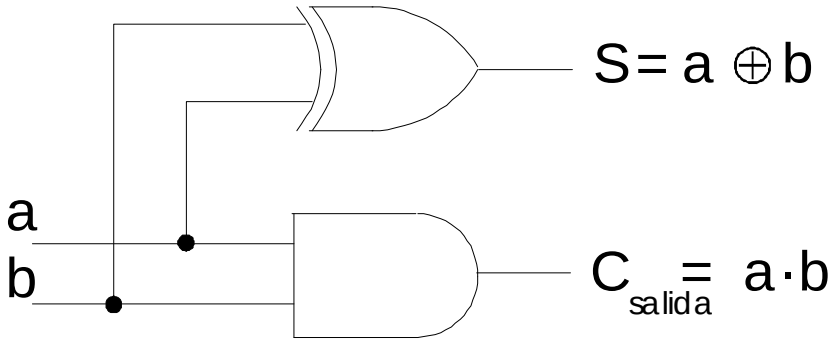
Ay B	C>	C=	C<	A>B	A=B	A<B
A >B	X	X	X	1	0	0
A <B	X	X	X	0	0	1
A = B	0	0	1	0	0	1
A = B	0	1	0	0	1	0
A = B	1	0	0	1	0	0

Adders (I)

Half-adder

It adds two bits yielding the addition and the carry

Entradas		Salidas	
Sumando a	Sumando b	Suma S	Acarreo C_{salida}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

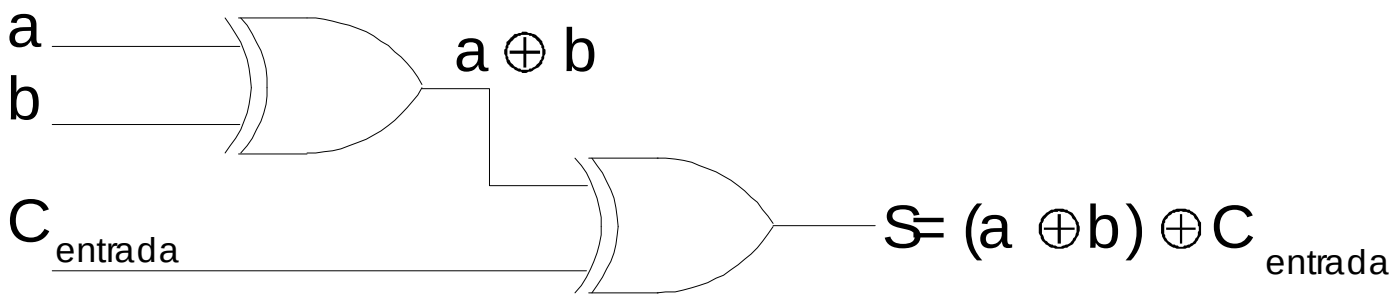


Adders (II)

Full adder

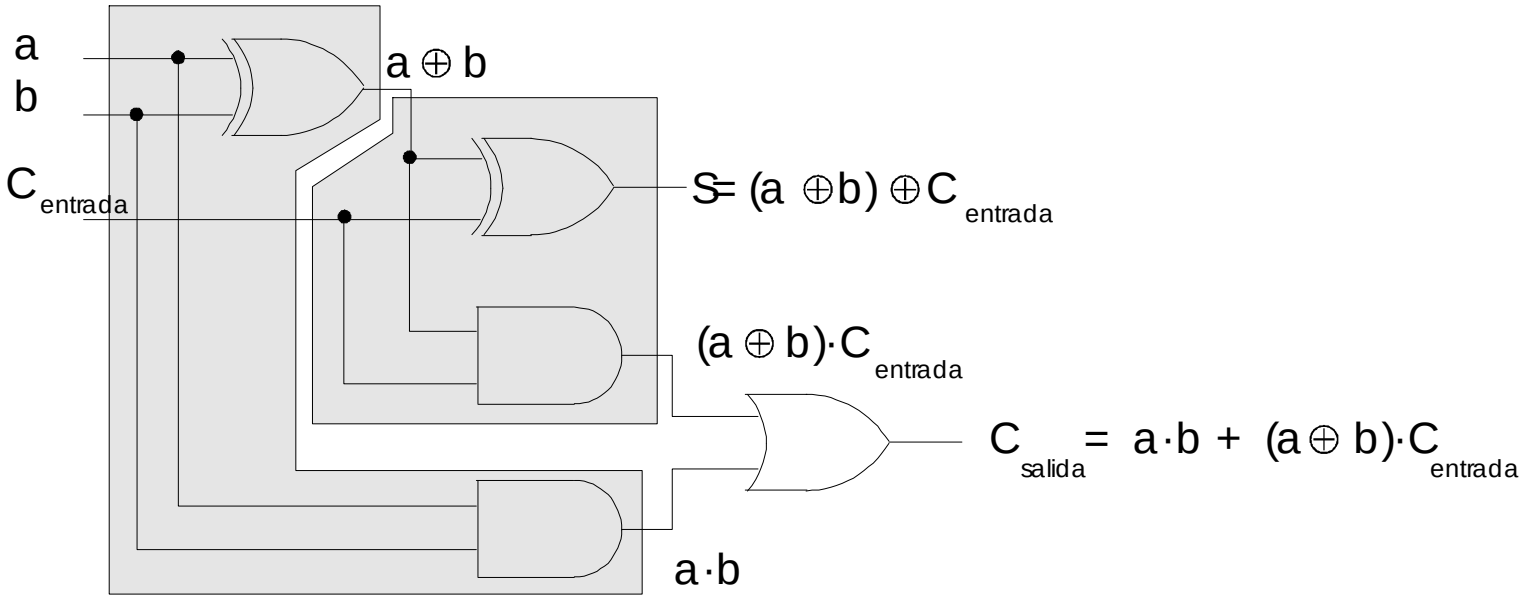
It adds as well an input carry

Entradas			Salidas	
Sumando a	Sumando b	Acarreo $C_{entrada}$	Suma S	Acarreo C_{salida}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



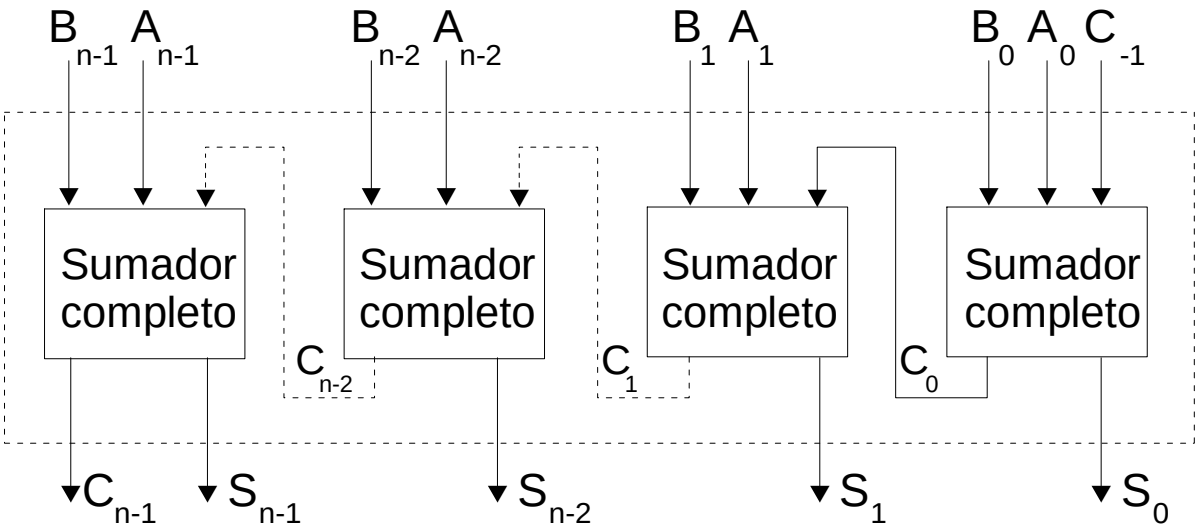
Adders (III)

Full adder built with 3 half adders



Adders (IV)

N-bit adder: connect full adders with the carry



Addition/subtraction circuit

Operations with sign in 2C complement