# Chapter II: Combinatorial logic

## 1. Definition

A logic circuit is an electronic component that processes and executes logical (Boolean) operations.

There are two types of logic circuits:

- Combinatorial (combinational) logic circuits
- Sequential logic circuits

## 2. Combinatorial circuits

A logic circuit is said to be combinatorial if the state (value) of its outputs depends only on the state (value) of its inputs. The combinational circuit must therefore not present any reactions from the output to the input, so that the state of the output does not depend on the history of the circuit.

## 3. Steps for designing a combinational circuit

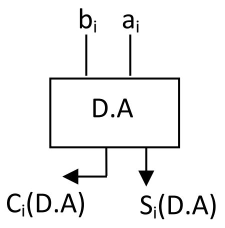
The design (synthesis) of a combinational circuit goes through three main stages after a thorough and careful reading to fully understand the statement in order to determine the input and output signals and their designation by symbols. The three main steps are:

- **Establishment of the truth table**
  View in the STM1 course.
- **Simplification of logic functions**
  View in the STM1 course.
- **Creation of the logical diagram**
  View in the STM1 course.

## 4. Study of some common combinational circuits:

### 4.1. The half adder

- **Definition :**
  The half-adder is a combinatorial logic circuit which allows an arithmetic addition operation to be carried out between two bits without taking into account the carry of the previous step.
- **General diagram:**



Noticed :
$b_i$ and $a_i$ represent the bits to be added.

$S_i$ represents the result of the addition (sum).

$C_i$ represents the carryover of the addition operation.

- **Creation of the circuit:**

  **Truth table :**

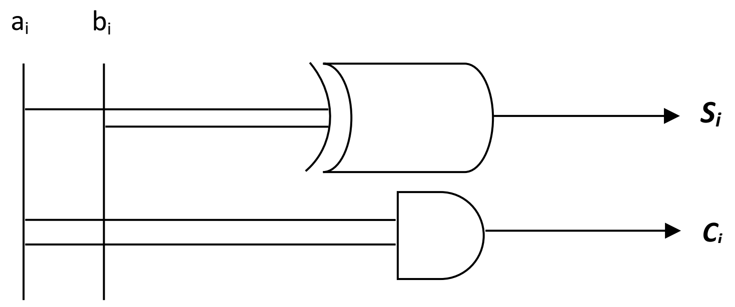
| $a_i$ | $b_i$ | $S_i$ | $C_i$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**Simplification:** directly from the truth table

$$S_i = a_i\overline{b_i} + \overline{a}_i b_i (*)$$
$$= a_i \oplus b_i (**)$$

$$C_i = a_i b_i$$

**Logic diagram:** for $S_i$, the representation could be made by (*) or (**)

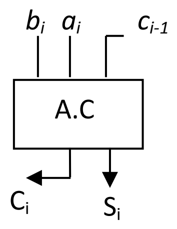


**4.2. The full adder**

- **Definition :**

  The full-adder is a combinatorial logic circuit which allows an arithmetic addition operation to be carried out between two bits while taking into account the carryover from the previous step.

- **General diagram:**



Noticed :

$b_i$ and $a_i$ represent the bits to be added.

$c_{i-1}$ represents the carryover from the previous step.

$S_i$ represents the result of the addition (sum).

$C_i$ represents the carryover of the addition operation.

- **Creation of the circuit:**

  **Truth table:**

| $c_{i-1}$ | $a_i$ | $b_i$ | $S_i$ | $C_i$ |
|---|---|---|---|---|
| 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 |

**<u>Karnaugh table</u> :**

$C_i$

| $c_{i-1}$ \ $a_i b_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **0** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **1** |

$S_i$

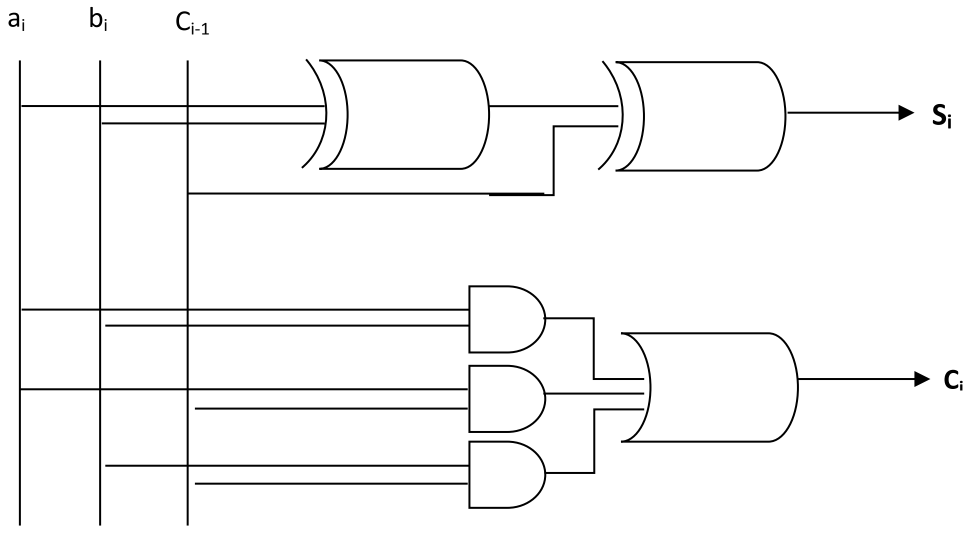
| $c_{i-1}$ \ $a_i b_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **0** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **0** |

$$C_i = a_i b_i + c_{i-1} a_i + c_{i-1} b_i$$

$$S_i = \overline{c_{i-1}}\,\overline{a_i}\,b_i + \overline{c_{i-1}}\,a_i\overline{b_i} + c_{i-1}\overline{a_i}\,\overline{b_i} + c_{i-1}a_ib_i \quad (*)$$

$$= a_i \oplus b_i \oplus c_{i-1} (**)$$

**<u>Logic diagram</u>:** for $S_i$, the representation could be made by (*) or (**)

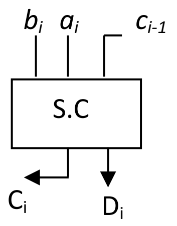


## 4.3. The Adder Subtractor (Complete Subtractor)

- **Definition :**

  The Adder Subtractor (Complete Subtractor) is a combinatorial logic circuit which allows an arithmetic subtraction operation to be carried out between two bits while taking into account the carryover from the previous step.

- **General diagram:**



- **Creation of the circuit:**

  <u>**Truth table:**</u>

| $c_{i-1}$ | $a_i$ | $b_i$ | $D_i$ | $C_i$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

<u>**Karnaugh table**</u> **:**

**$C_i$**

| $c_{i-1}$ \ $a_i b_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

**$D_i$**

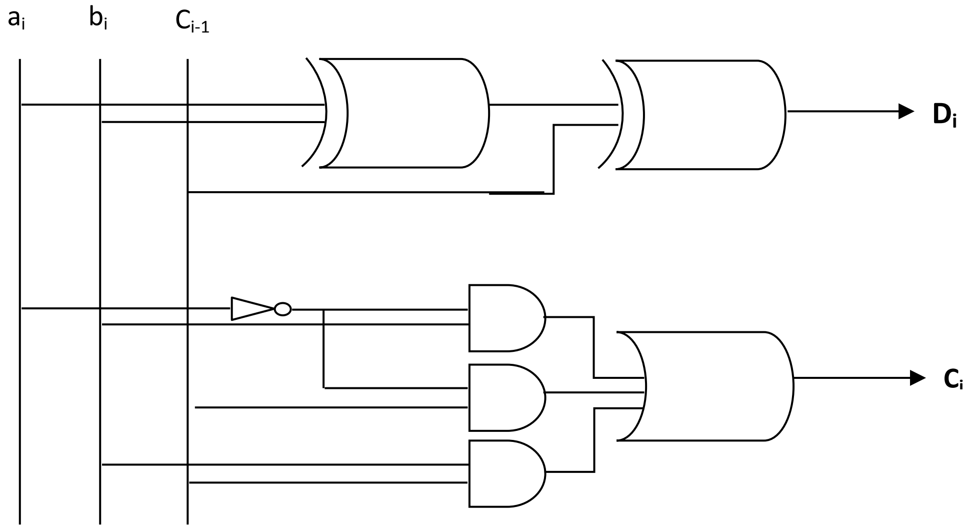
| $c_{i-1}$ \ $a_i b_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

$$C_i = \overline{a_i} b_i + c_{i-1}\overline{a_i} + c_{i-1} b_i$$

$$S_i = \overline{c_{i-1}}\,\overline{a_i} b_i + \overline{c_{i-1}} a_i \overline{b_i} + c_{i-1}\overline{a_i}\,\overline{b_i} + c_{i-1} a_i b_i \quad (*)$$

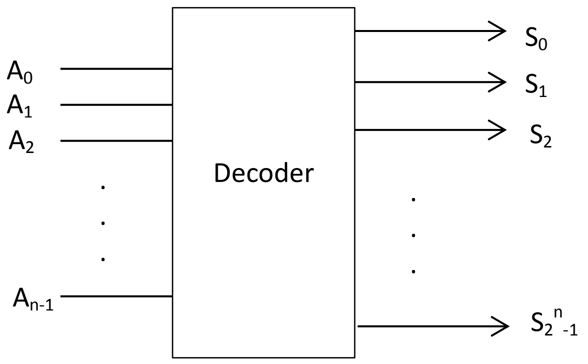
$$= a_i \oplus b_i \oplus c_{i-1} (**)$$

<u>**Logic diagram**</u>**:** for $D_i$, the representation could be made by (*) or (**)

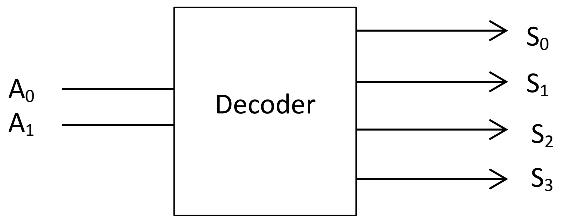


## 4.4. Decoders

- **Definition :**
  The decoder is a combinatorial system whose function is to activate one of the $2^n$ outputs. The selection is made using n address lines and the outputs are mutually exclusive.
- **Notation:** Decoder 1 among $2^n$ (Decoder like a DEMUX with E=1)
- **General representation:**



- **Example:** creation of a Decoder 1 of 4.
  We have 4 outputs => 4 = $2^2$ => n=2
  => We have 2 address lines.
  => The general diagram of this decoder is as follows:



**<u>Truth table:</u>**

| $A_1$ | $A_0$ | $S_0$ | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |

**<u>Simplification</u> :** directly from the truth table

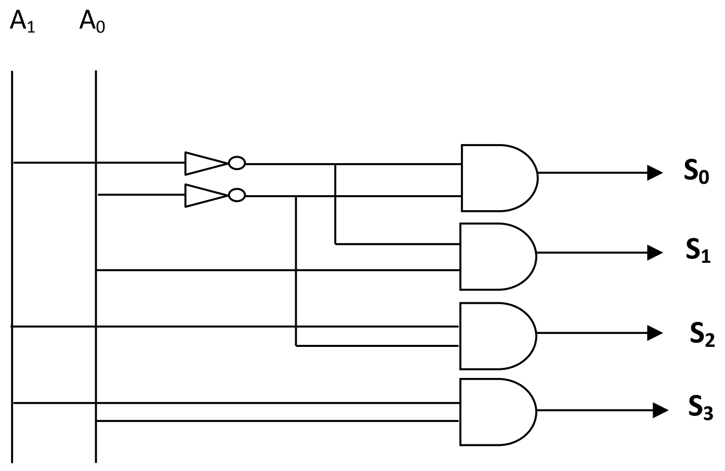
$$S_0 = \overline{A_1}.\overline{A_0}$$

$$S_1 = \overline{A_1}.A_0$$

$$S_2 = A_1.\overline{A_0}$$

$$S_3 = A_1.A_0$$

**<u>Logic diagram</u>:**

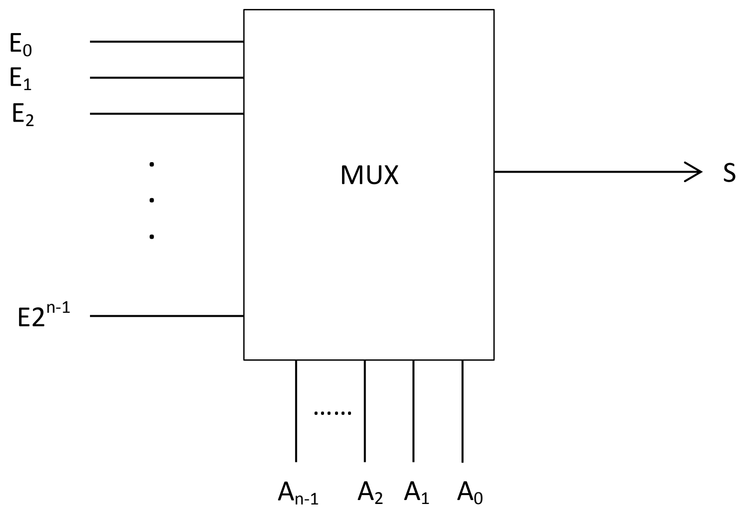


<u>Noticed</u> :
The decoder can be used for memory addressing.

**4.5. Multiplexers**

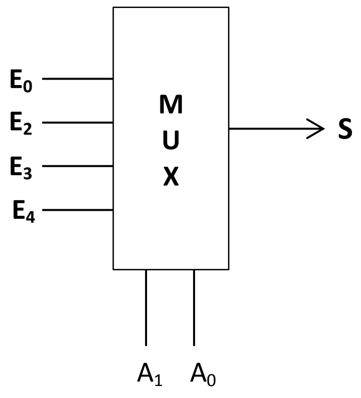
- **Definition :**
  The multiplexer is a combinatorial system whose function is to select one of $2^n$ inputs and transmit it to the output. The selection is made using n address lines.
- **Notation:** MUX $2^n$ to 1.
- **General representation:**

Input information.



Address lines (Selection).

- **Example :** creation of a 4 to 1 MUX multiplexer.
  We have 4 inputs => $4 = 2^2$ => n=2
  => We have 2 address lines.
  => The general diagram of this multiplexer is as follows:



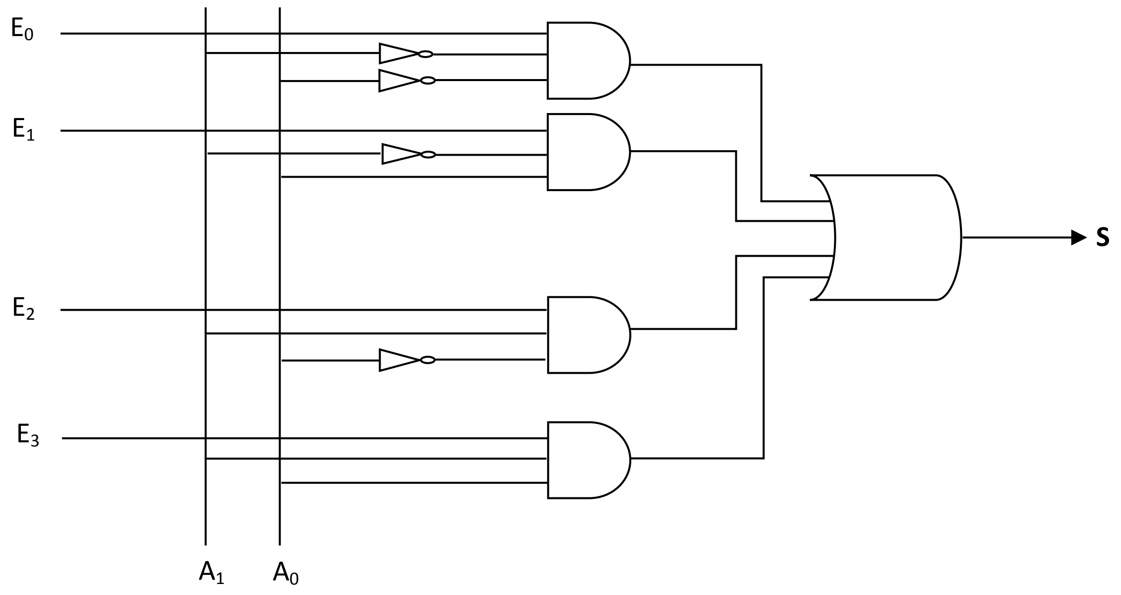
**Truth table:**

| $A_1$ | $A_0$ | $S$ |
|---|---|---|
| 0 | 0 | $E_0$ |
| 0 | 1 | $E_1$ |
| 1 | 0 | $E_2$ |
| 1 | 1 | $E_3$ |

**Simplification:** directly from the truth table, we obtain the following expressions:

$$\boxed{\rightarrow S = E_0\,\bar{A}_1\,\bar{A}_0 + E_1\bar{A}_1 A_0 + E_2 A_1 \bar{A}_0 + E_3 A_1 A_0}$$

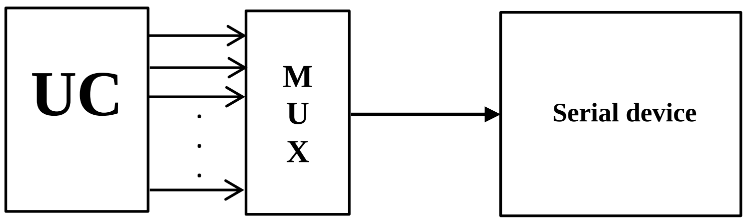
**Logic diagram:**



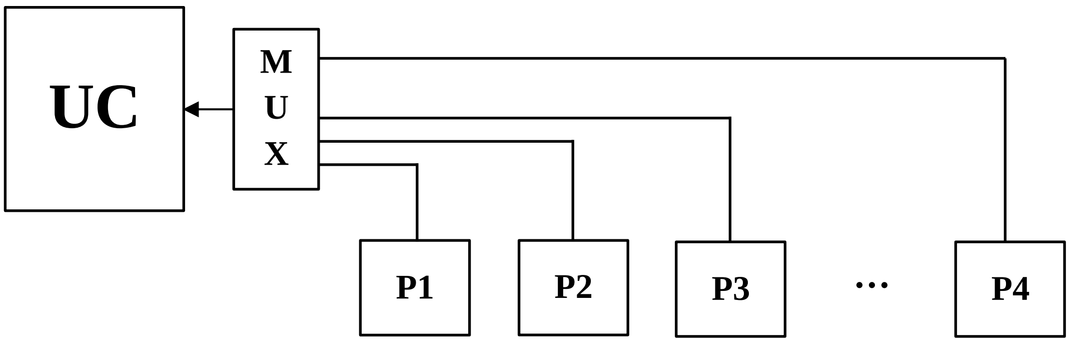
Noticed :

The main applications of MUXs are:

☞ Parallel/serial conversion.

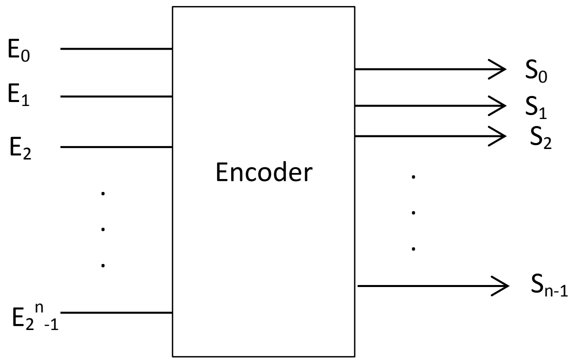


☞ Concentration of information on a single line.

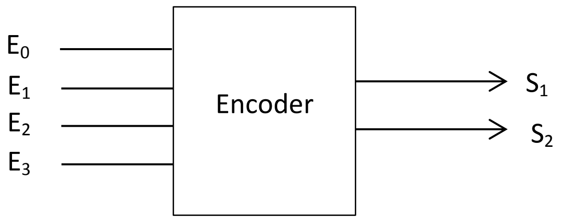


## 4.6. Priority encoders

- **Definition :**
  The encoder is a combinatorial system whose function is to return the activation index of one of $2^n$ inputs. The activation index is given on n address lines. When multiple inputs are activated, the encoder prioritizes the highest priority active input among the active inputs.
- **Notation:** Encoder $2^n$ to n
- **General representation:**



- **Example:** creation of a 4 to 2 Encoder.
  The general diagram of this decoder is as follows:



The encoder prioritizes the inputs as follows: $E_0 > E_1 > E_2 > E_3$

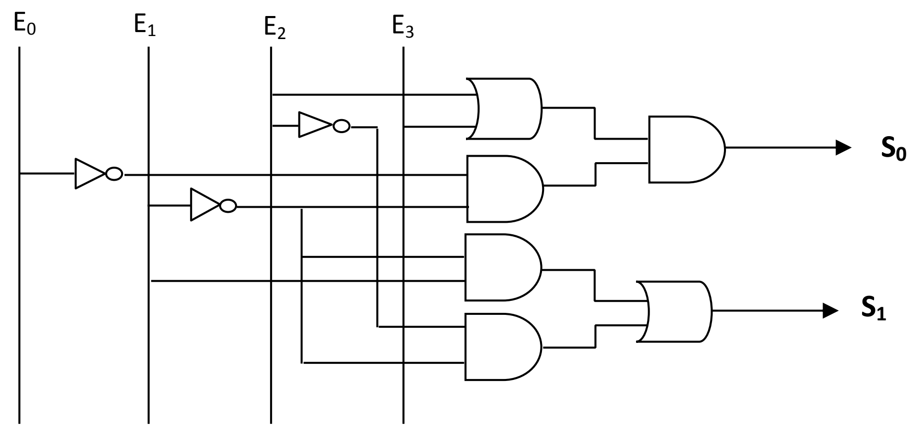
**Truth table:**

| $E_0$ | $E_1$ | $E_2$ | $E_3$ | $S_0$ | $S_1$ |
|---|---|---|---|---|---|
| 1 | X | X | X | 0 | 0 |
| 0 | 1 | X | X | 0 | 1 |
| 0 | 0 | 1 | X | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

**Simplification :** directly from the truth table, uses its property: $\alpha + \bar{\alpha}\beta = \alpha + \beta$

$$\begin{aligned} S_0 &= \overline{E_0}.\overline{E_1}.E_2 + \overline{E_0}.\overline{E_1}.\overline{E_2}.E_3 \\ &= \overline{E_0}.\overline{E_1}.(E_2 + \overline{E_2}.E_3) \\ &= \overline{E_0}.\overline{E_1}.(E_2 + E_3) \\ S_1 &= \overline{E_0}.E_1 + \overline{E_0}.\overline{E_1}.\overline{E_2}.E_3 \\ &= \overline{E_0}.(E_1 + \overline{E_1}.\overline{E_2}.E_3) \\ &= \overline{E_0}.(E_1 + \overline{E_2}.E_3) \\ &= \overline{E_0}.E_1 + E_1.\overline{E_2}.E_3) \end{aligned}$$

**Logic diagram:**

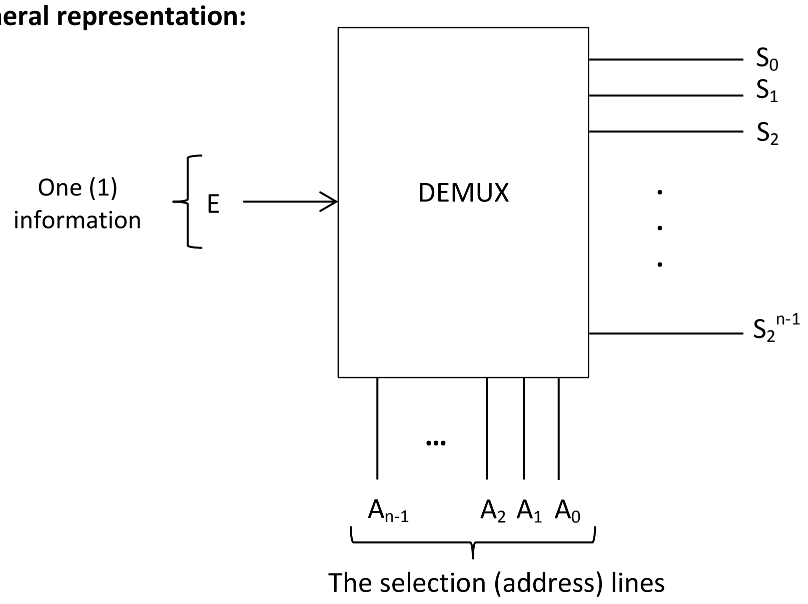


Noticed :

The encoder can be used for interfacing a keyboard and switching from the decimal system to the BCD system.

**4.7. Demultiplexers**

- **Definition :**
  The demultiplexer is a combinatorial system whose function is to transmit an input to one of the 2n outputs. The selection is made using n address lines and the outputs are mutually exclusive.
- **Notation:** DEMUX 1 to $2^n$
- **General representation:**

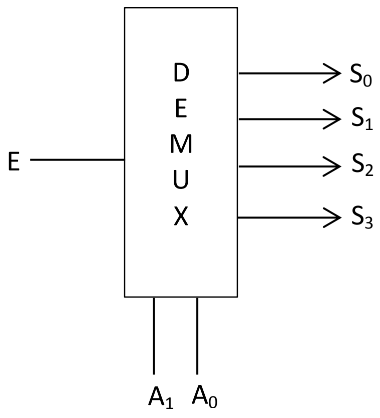


- **Example :** creation of a DEMUX 1 to 4 demultiplexer.

  We have 4 inputs => $4 = 2^2$ => n=2

  => We have 2 address lines.

  => The general diagram of this multiplexer is as follows:



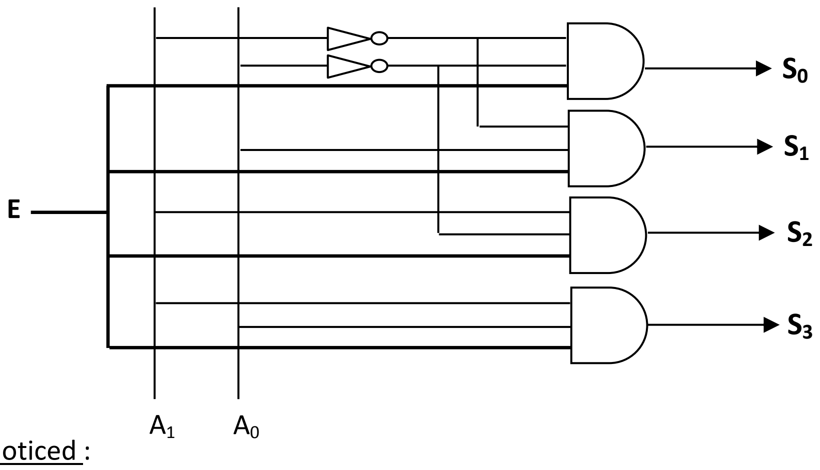
**Truth table:**

| $A_1$ | $A_0$ | $S_0$ | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|---|---|
| 0 | 0 | $E$ | 0 | 0 | 0 |
| 0 | 1 | 0 | $E$ | 0 | 0 |
| 1 | 0 | 0 | 0 | $E$ | 0 |
| 1 | 1 | 0 | 0 | 0 | $E$ |

**Simplification :** directly from the truth table, we obtain the following expressions:

$$\boxed{\begin{aligned} S_0 &= E\,\overline{A_1}\,\overline{A_0} \\ S_1 &= E\,\overline{A_1}\,A_0 \\ S_2 &= E\,A_1\,\overline{A_0} \\ S_3 &= E\,A_1A_0 \end{aligned}}$$

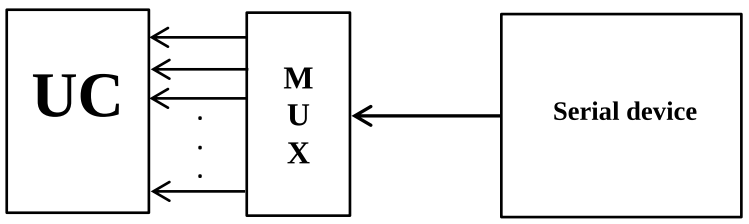
**Logic diagram:**



Noticed :

The main applications of DEMUXs are:

☞ Series/parallel conversion.



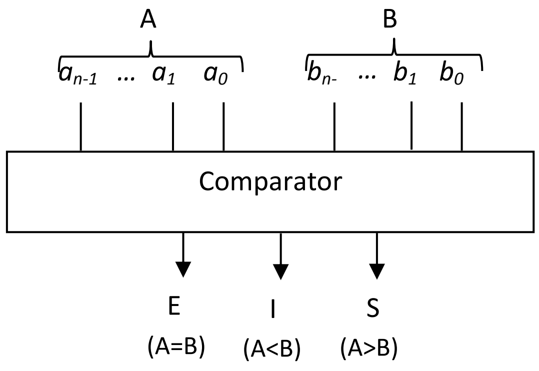
**5. Other examples of combinational circuits**

**5.1. The comparator**

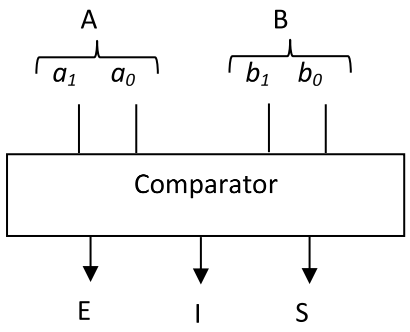
- **Definition :**

  The comparator is a combinational logic circuit allowing two (n-bit) binary numbers A( $a_{n-1}$,…, $a_1$, $a_0$ ) and B( $b_{n-1}$,…, $b_1$, $b_0$ ) to be compared with each other by giving one of the following three results: equal (A=B), lower (A<B), and higher (A>B).

- **General representation:**



- **Example:** making a comparator of two numbers with two bits each: A( $a_1 a_0$ ) and B( $b_1 b_0$ ).
  The general diagram of this comparator is as follows:

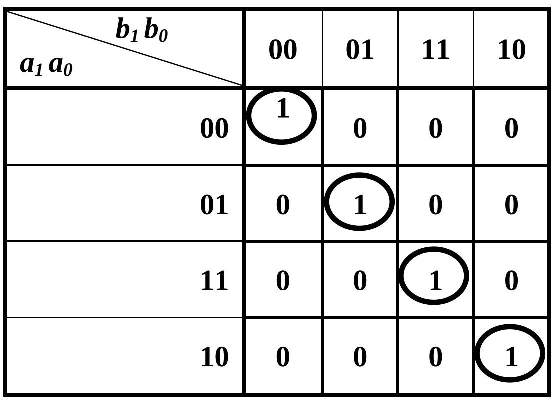


**Truth table:**

| $a_1$ | $a_0$ | $b_1$ | $b_0$ | **E** | **I** | **S** |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | **1** | **0** | **0** |
| 0 | 0 | 0 | 1 | **0** | **1** | **0** |
| 0 | 0 | 1 | 0 | **0** | **1** | **0** |
| 0 | 0 | 1 | 1 | **0** | **1** | **0** |
| 0 | 1 | 0 | 0 | **0** | **0** | **1** |
| 0 | 1 | 0 | 1 | **1** | **0** | **0** |
| 0 | 1 | 1 | 0 | **0** | **1** | **0** |
| 0 | 1 | 1 | 1 | **0** | **1** | **0** |
| 1 | 0 | 0 | 0 | **0** | **0** | **1** |
| 1 | 0 | 0 | 1 | **0** | **0** | **1** |
| 1 | 0 | 1 | 0 | **1** | **0** | **0** |
| 1 | 0 | 1 | 1 | **0** | **1** | **0** |
| 1 | 1 | 0 | 0 | **0** | **0** | **1** |
| 1 | 1 | 0 | 1 | **0** | **0** | **1** |
| 1 | 1 | 1 | 0 | **0** | **0** | **1** |
| 1 | 1 | 1 | 1 | **1** | **0** | **0** |

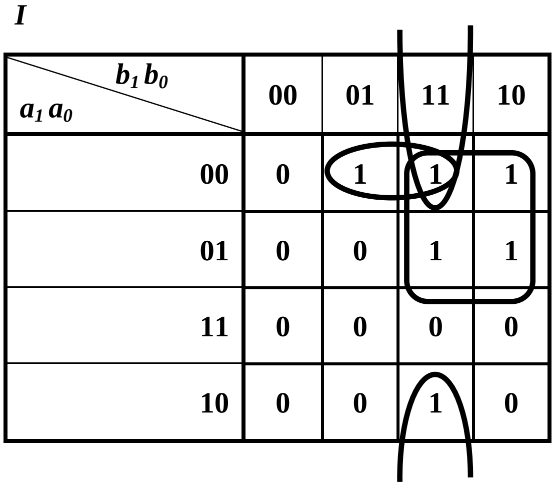
**Karnaugh table :**

***E***



| $b_1 b_0$ / $a_1 a_0$ | **00** | **01** | **11** | **10** |
|---|---|---|---|---|
| **00** | 1 | 0 | 0 | 0 |
| **01** | 0 | 1 | 0 | 0 |
| **11** | 0 | 0 | 1 | 0 |
| **10** | 0 | 0 | 0 | 1 |

***I***



| $b_1 b_0$ / $a_1 a_0$ | **00** | **01** | **11** | **10** |
|---|---|---|---|---|
| **00** | 0 | 1 | 1 | 1 |
| **01** | 0 | 0 | 1 | 1 |
| **11** | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 1 | 0 |

$$\boldsymbol{E} = \overline{a_1}\,\overline{a_0}\,\overline{b_1}\,\overline{b_0} + \overline{a_1}\,a_0\overline{b_1}\,b_0 + a_1a_0b_1b_0 + a_1a_0b_1\overline{b_0} + a_1\overline{a_0}b_1\overline{b_0}$$

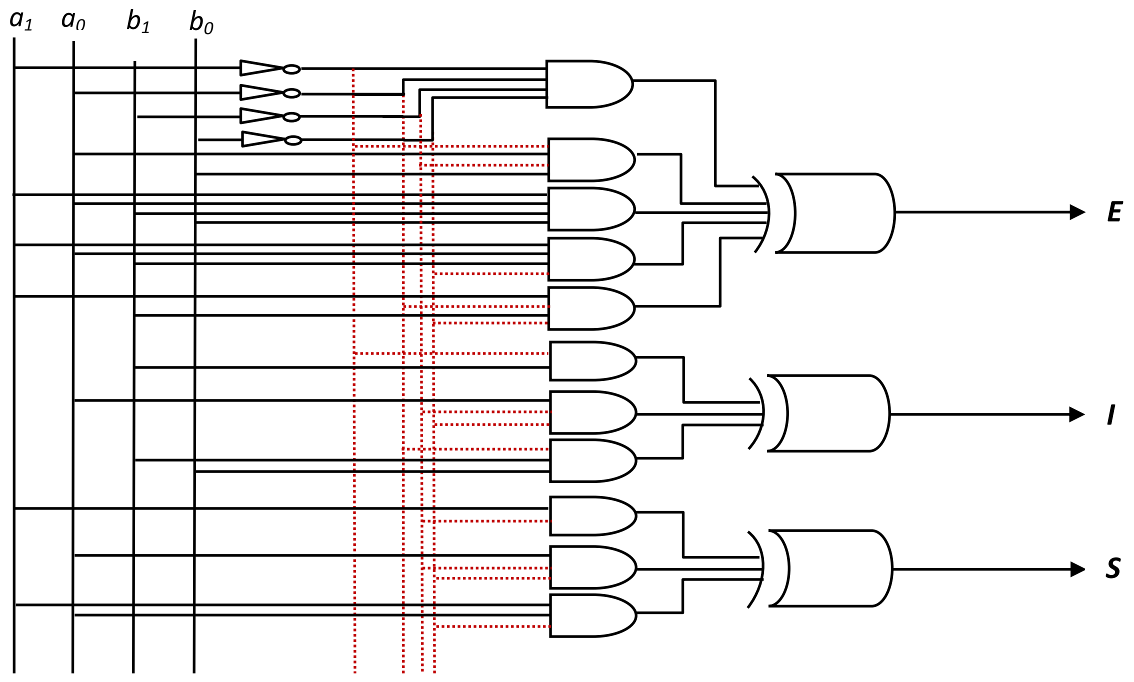
$$\boldsymbol{I} = \overline{a_1}\,b_1 + \overline{a_1}\,\overline{a_0}\,b_0 + \overline{a_0}\,b_1b_0$$

***S***

| $a_1 a_0$ \ $b_1 b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 1 |
| 10 | 1 | 1 | 0 | 0 |

$$S = a_1\overline{b_1} + a_0\overline{b_1}\,\overline{b_0} + a_1 a_0\overline{b_0}$$

**<u>Logic diagram</u>:**



**5.2. The parity generator**

- **Definition :**

A parity generator logic circuit is a circuit which receives a sequence of bits as input and gives as output the input sequence of bits as well as a new so-called parity bit added to this sequence allowing the verification of the accuracy of this sequence after transmission.

- **Example :** creation of a parity bit generator for a sequence of eight bits: $a_7\,a_6\,a_5\,a_4\,a_3\,a_2\,a_1\,a_0$.
  If the XOR (or exclusive) logical operator concerns a number of bits, the result is 0 if the number of 1s is even, and 1 if the number of 1s is odd.
  - <u>the number of 1s is even</u> : 10110010
    F= $(1 \oplus 0) \oplus (1 \oplus 1) \oplus (0 \oplus 0) \oplus (1 \oplus 0)$
      $= (1 \oplus 0) \oplus (0 \oplus 1)$

= $1 \oplus 1$
= 0

- <u>the number of 1s is odd</u> : 11010110

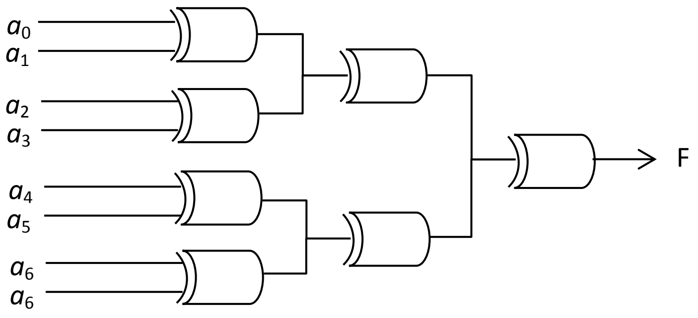
  F= $(1 \oplus 1) \oplus (0 \oplus 1) \oplus (0 \oplus 1) \oplus (1 \oplus 0)$
  = $(0 \oplus 1) \oplus (1 \oplus 1)$
  = $1 \oplus 0$
  = 1

**Logic diagram:**

This circuit can be represented in two different ways: as a pyramid and as a cascade.

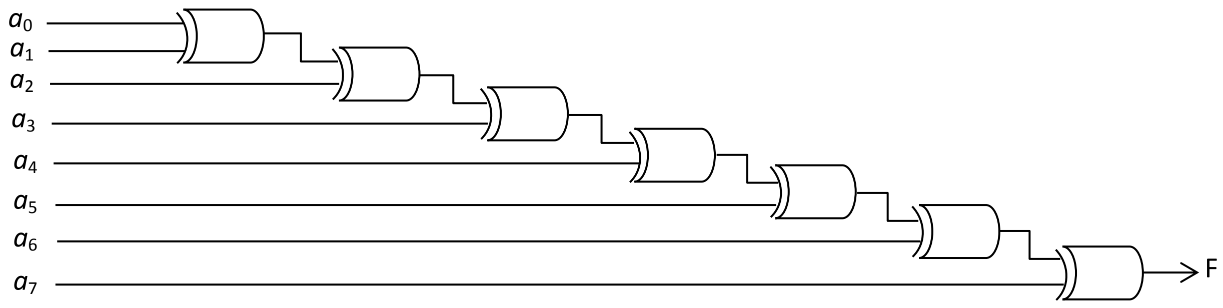
**In pyramid**

F =$(a_7 \oplus a_6) \oplus (a_5 \oplus a_4) \oplus (a_3 \oplus a_2) \oplus (a_1 \oplus a_0)$



**Cascade**

F =$(a_7 \oplus (a_6 \oplus (a_5 \oplus (a_4 \oplus (a_3 \oplus (a_2 \oplus (a_1 \oplus a_0)))))))$


$a_0$
$a_1$
$a_2$
$a_3$
$a_4$
$a_5$
$a_6$
$a_7$
F