Part II

COMBINATIONAL AND SEQUENTIAL LOGIC

Chapter 5:

COMBINATIONAL LOGIC

Introduction

To understand the operation of the main components of a computer, such as the arithmetic-logic unit (ALU), you'll need to achieve the following objectives:

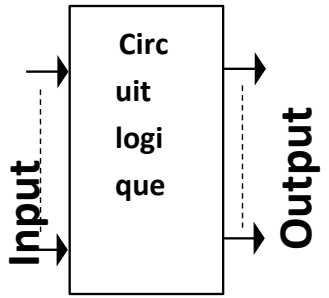
1. Describe the operation and properties of logic gates, simple combinational circuits such as adders, decoders, multiplexers and demultiplexers...;
2. Use the theorems and identities of Boolean algebra to synthesize a circuit from its truth table and simplify the result obtained.

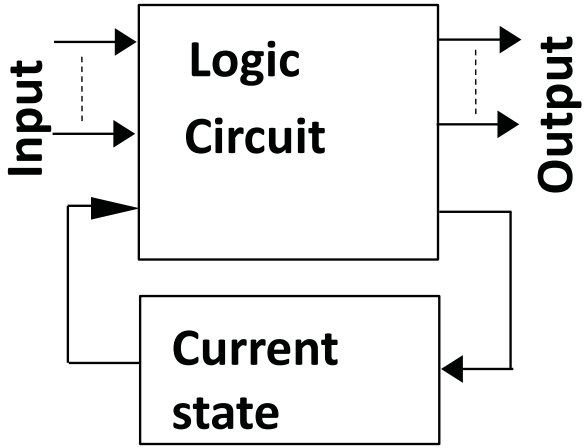
Objectives

- Learn the structure of some frequently used combinatorial circuits (half adder, full adder, ...).
- Learn how to use combinatorial circuits to design other, more complex circuits.

Logic Circuits

- Electronic circuit performing one or more logic functions.
- A logic circuit consists of:
  - A set of logic gates
  - Logic sub-circuits
  - All are interconnected

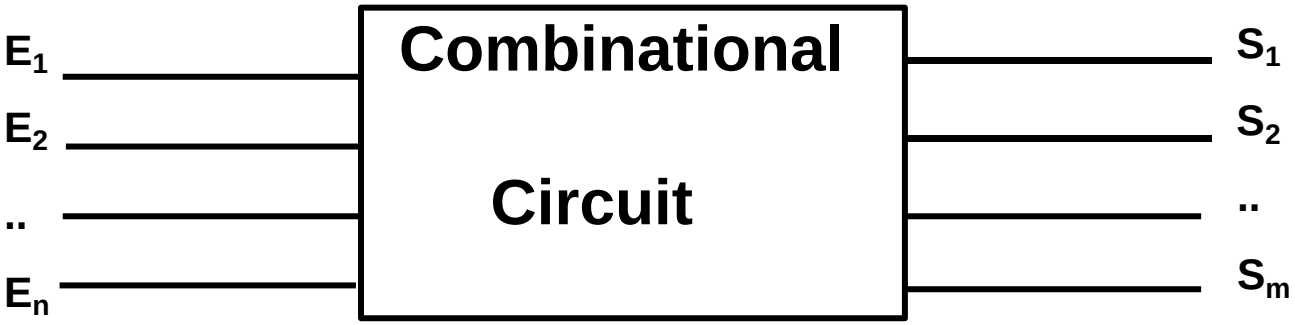




- Two types of logic circuit
  - **Combinational Circuits** : S = f(E)
  - **Sequential Circuits** : notion of state and memory

1. Combinational circuit

- **A combinational circuit is a digital circuit whose outputs depend solely on the inputs.**
- **$S_i=F(E_i)$**
- **$S_i=F(E_1,E_2,\ldots,E_n)$**



- **It's possible to use combinational circuits to create other, more complex circuits.**

Examples of combinational circuits

1. Half adder
2. Full adder
3. Comparator
4. Multiplexer
5. Demultiplexer
6. Encoder
7. Decoder
8. Transcoder

Combinational circuits

- A combinational circuit is defined by one or more logic functions
  - Definition of output values as a function of circuit inputs.
  - Boolean algebra and logic functions are the theoretical underpinnings of combinatorial circuits.
- A circuit is represented by a logic diagram.

Example of combinational circuits

- In a computer, we can distinguish three different classes of combinatorial logic circuits:
  1. Combinatorial circuits for arithmetic and logic calculations, such as adders, subtracters, comparators, etc.
  2. Combinatorial circuits for data routing and transmission, such as encoders, decoders, multiplexers, demultiplexers, etc.
  3. Combinatorial circuits for coding and code conversion, such as transcoders, 7-segment displays, etc.

Reminder: Synthesis of a logic circuit

- From a logic function  Find the flow chart corresponding to this function

Principle

- Simplify the logic function using two methods:
  1. The algebraic method (Boolean algebra)
  2. Karnaugh map method
- Derive the corresponding logigram

Reminder: Analysis of a logic circuit

- From a circuit diagram ⇒ Find its logic function
- **Principle:**

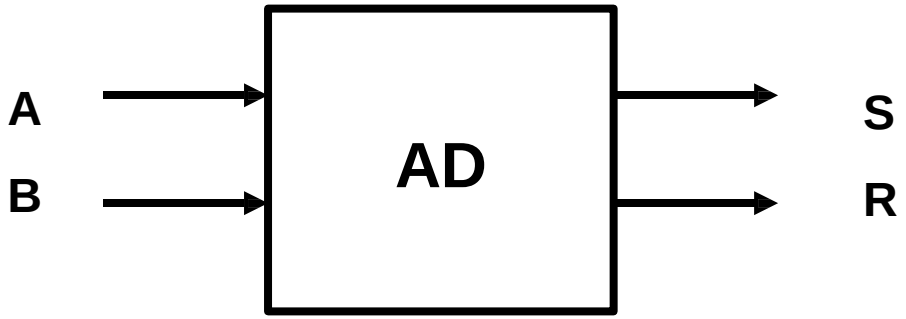
1. Give the expression of the outputs of each gate/component as a function of the values of its inputs.
2. Finally, deduce the logic function(s) of the circuit.

- Then, we can:

1. Determine the circuit's truth table.
2. Simplify the logic function using the properties of Boolean algebra or Karnaugh maps.

2. Half-adder

- **The half adder is a combinatorial circuit that performs the arithmetic sum of two numbers A and B, each one is on one bit.**
- **The output is the sum S and the carry R.**



To find the structure (schematic) of this circuit, first draw up its truth table.

Half-adder: truth table

- **In binary, single-bit addition is performed as follows:**

$$\begin{cases} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{cases}$$

- **The related truth table is :**

A	B		R	S
0	0			
0	1			
1	0			
1	1			

From the truth table we get:

$$R =$$

$$S =$$

Half-adder: truth table

- **In binary, single-bit addition is performed as follows:**

$$\begin{cases} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{cases}$$

- **The related truth table is :**

A	B		R	S
0	0		0	0
0	1		0	1
1	0		0	1
1	1		1	0

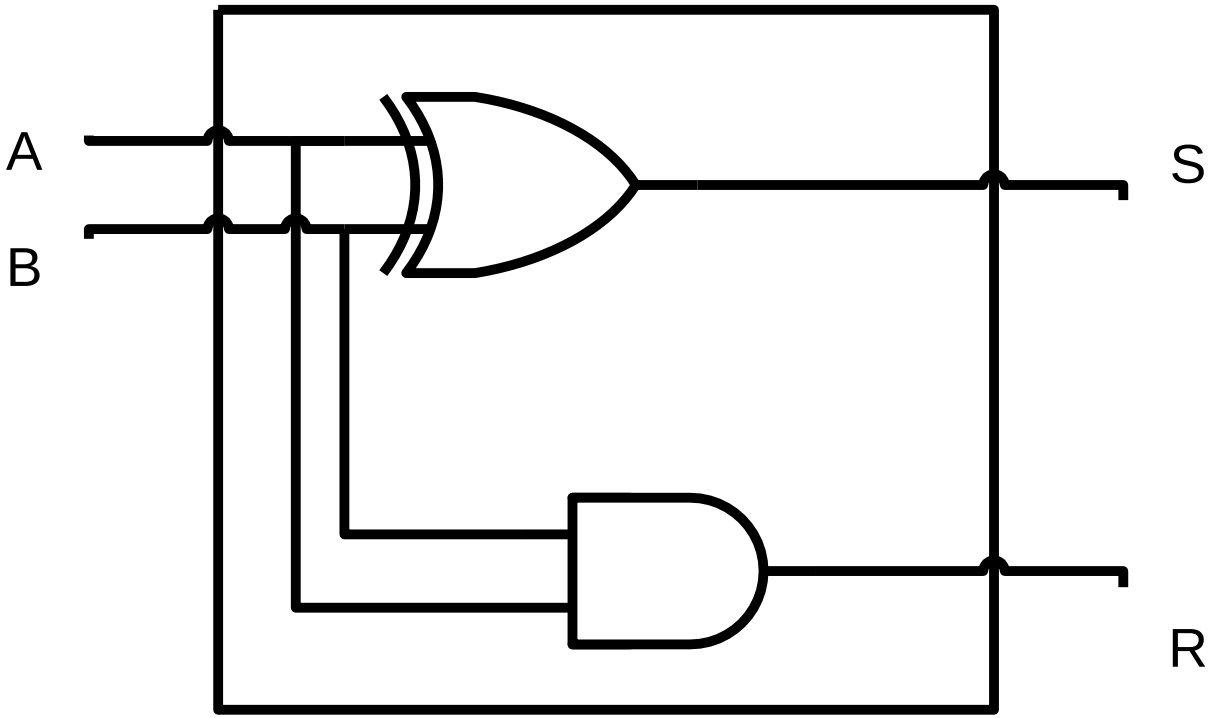
From the truth table we get:

$$R = A.B$$

$$S = A.\overline{B} + \overline{A}.B = A \oplus B$$

Half-adder: logic circuit

$$R = A.B$$
$$S = A \oplus B$$



3. The full adder

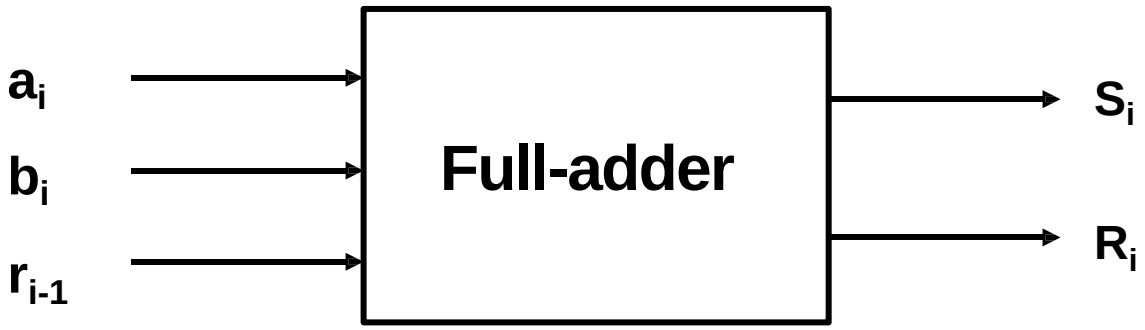
- **In binary, when you do the addition operation, you have to take into account the incoming carry.**

r_4	r_3	r_2	r_1	$r_0 = 0$
	a_4	a_3	a_2	a_1
+	b_4	b_3	b_2	b_1
r_4	s_4	s_3	s_2	s_1

	r_{i-1}	
	a_i	
+	b_i	
r_i	s_i	

3.1 1-bit full adder

- **The full one-bit adder has 3 inputs :**
  - **a_i: the first number on a bit.**
  - **b_i: the second number on one bit.**
  - **r_{i-1}: the one-bit carry-in.**
- **It has two outputs:**
  - **Si: the sum**
  - **Ri: the outgoing deducation/carry**



Truth table of a 1-bit full adder

a_i	b_i	r_{i-1}		r_i	s_i
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

$S_i =$

$R_i =$

Truth table of a 1-bit full adder

a_i	b_i	r_{i-1}		r_i	s_i
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

0+0 = 0 retenue 0
0+1 = 1 + 0 = 1 retenue 0
1 + 1 = 0 retenue 1
1+1+1 = 1 retenue 1

$$S_i = \overline{A_i}.\overline{B_i}.R_{i-1} + \overline{A_i}.B_i.\overline{R}_{i-1} + A_i.\overline{B_i}.\overline{R}_{i-1} + A_i.B_i.R_{i-1}$$

$$R_i = \overline{A_i}B_iR_{i-1} + A_i\overline{B}_iR_{i-1} + A_iB_i\overline{R}_{i-1} + A_iB_iR_{i-1}$$

If we want to simplify the equations, we get :

$$R_i = \overline{A_i} B_i R_{i-1} + A_i \overline{B}_i R_{i-1} + A_i B_i \overline{R}_{i-1} + A_i B_i R_{i-1}$$

$$R_i = R_{i-1}.(\overline{A_i}.B_i + A_i.\overline{B}_i) + A_i B_i (\overline{R}_{i-1} +_i R_{i-1})$$

$$R_i = R_{i-1}.(A_i \oplus B_i) + A_i B_i$$

If we want to simplify the equations, we get

$$S_i = \overline{A_i}.B_i.R_{i-1} + \overline{A_i}.B_i.\overline{R_{i-1}} + A_i.\overline{B_i}.\overline{R_{i-1}} + A_i.B_i.R_{i-1}$$

$$S_i = \overline{A_i}.(B_i \oplus R_{i-1}) + A_i.\overline{(B_i \oplus R_{i-1})}$$

$X = B_i \oplus R_{i-1}$ $X = B_i \oplus R_{i-1}$

$S_i = A_i \oplus X$

$$R_i = \overline{A_i}B_iR_{i-1} + A_i\overline{B}_iR_{i-1} + A_iB_i\overline{R}_{i-1} + A_iB_iR_{i-1}$$

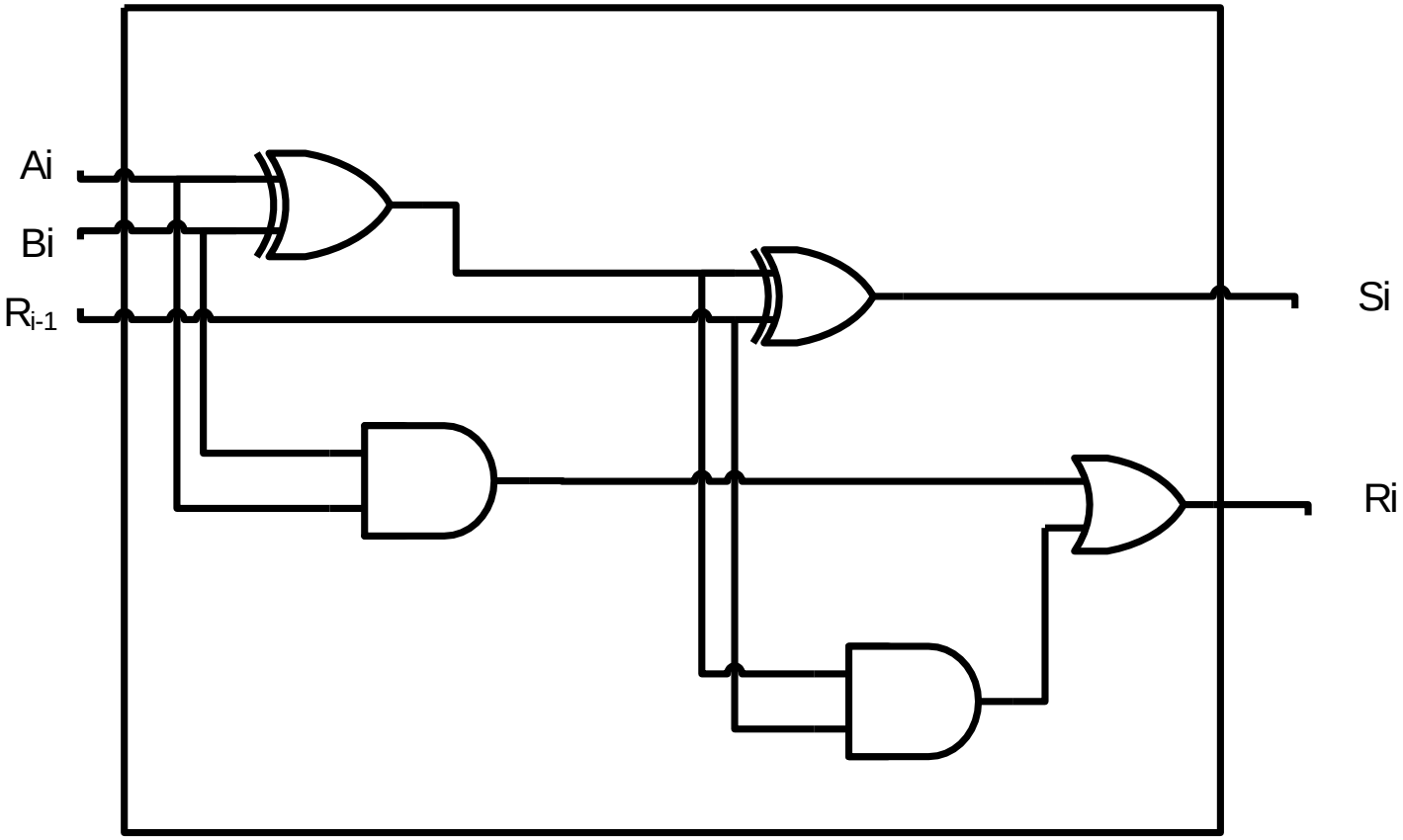
$$R_i = R_{i-1}.(\overline{A_i}.B_i + A_i.\overline{B}_i) + A_iB_i(\overline{R}_{i-1} + {}_i\ R_{i-1})$$

$$R_i = R_{i-1}.(A_i \oplus B_i) + A_iB_i$$

3.2 Schematic diagram of a full adder

$$R_i = A_i . B_i + R_{i-1} . (B_i \oplus A_i)$$

$$S_i = A_i \oplus B_i \oplus R_{i-1}$$



Application exercise

An odd parity generator is a function that returns 1 if the number of bits to 1 is odd and 0 otherwise.

1. **Define this function for a 4-bit word.**

2. **Give a logic circuit implementing this function.**

Solution of the application exercise

The formula for the 4-bit odd parity generator (P) obtained directly from the truth table is :

$P = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$

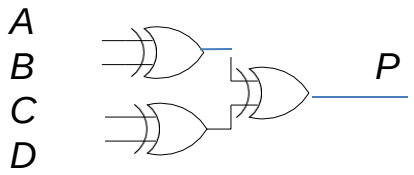
$+ \ldots$

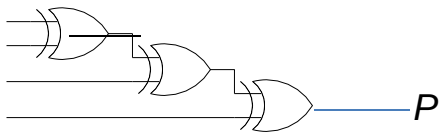
$+ A \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot C \cdot D + A \cdot B \cdot \overline{C} \cdot D + A \cdot B \cdot C \cdot \overline{D}$ which would make the circuit far too complicated.

We note that for two bits, $P = A \oplus B$

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

The following circuits are derived from this:





4. The Comparator

- A comparator is an arithmetic circuit for comparing two binary numbers A and B.
- A and B must have the same length (number of bits).

- The question is whether A > B, A<B or A=B. So we understand that the circuit answers a three-choice question.
- Our final circuit must produce three signals:
  1. fs(active if A>B),
  2. fl(active if A<B) and
  3. fe(active if A=B)

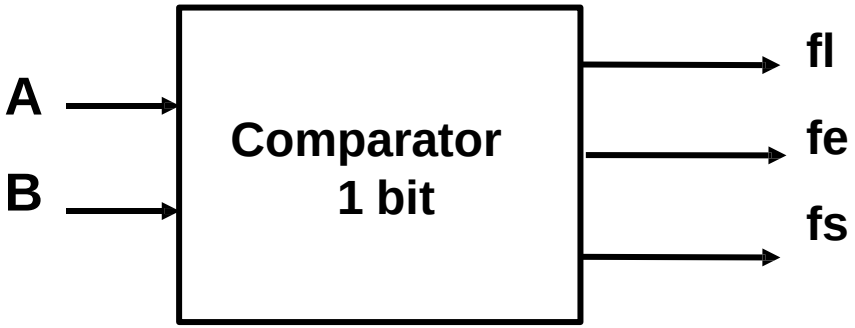
  taking signals A and B as inputs.

4. The Comparator

- **It's a combinational circuit that compares two binary numbers A and B.**

- **It has 2 inputs:**
  - **A: on one bit**
  - **B: on one bit**

- **It has 3 outputs**
  - **fe : equality (A=B)**
  - **fl : lower (A < B)**
  - **fs : superior (A > B)**



4.1 One-bit comparator

A	B		fs	fe	fl
0	0				
0	1				
1	0				
1	1				

$$fs =$$

$$fl =$$

$$fe =$$

4.1 One-bit comparator

A	B		fs	fe	fl
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

$$fs = A.\overline{B}$$

$$fl = \overline{A}B$$

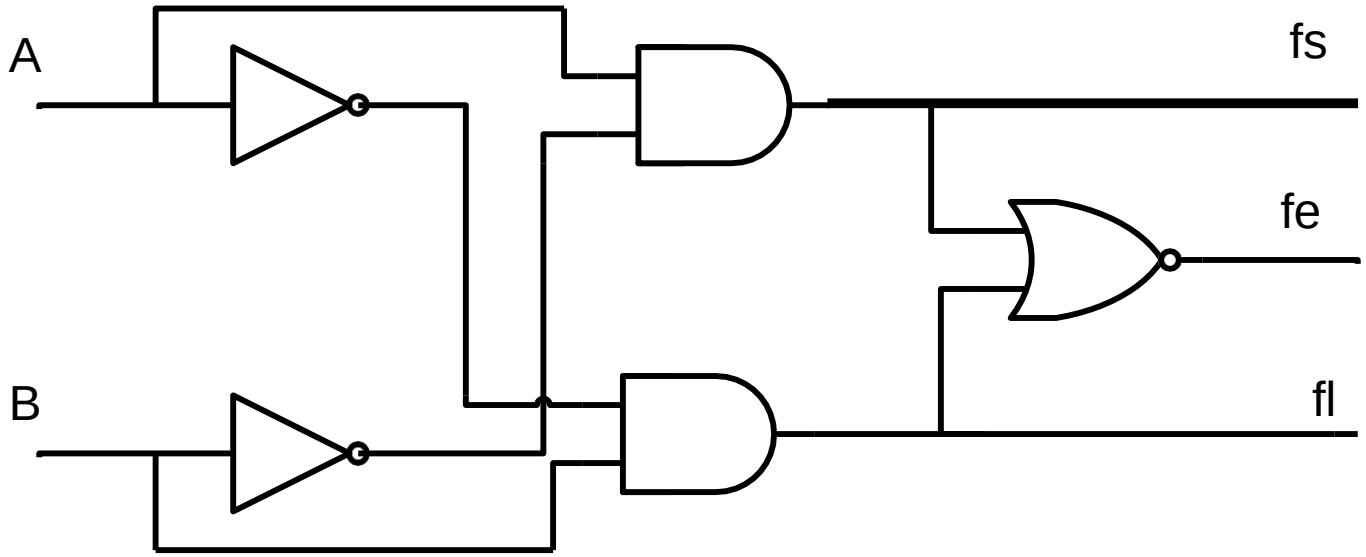
$$fe = \overline{A}\,\overline{B} + AB$$

Schéma d'un comparateur sur un bit

$$fs = A.\overline{B}$$

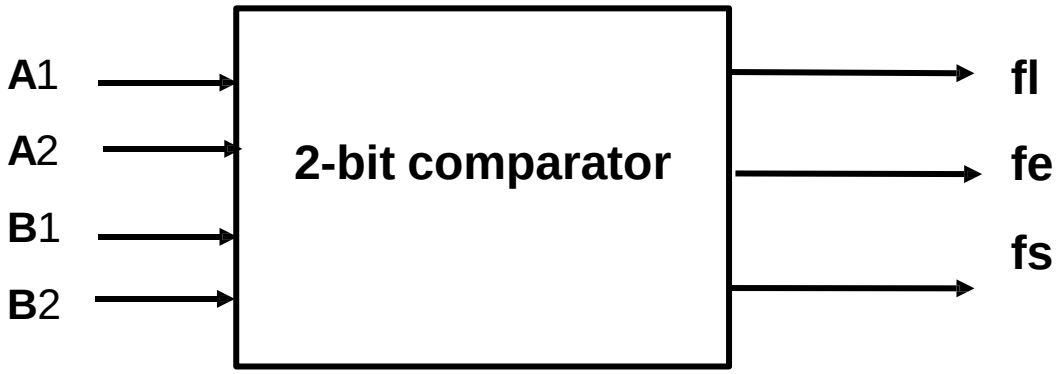
$$fl = \overline{A}B$$

$$fe = \overline{fs + fi}$$



4.2 2-bit comparator

- **It compares two numbers, A(A1A2) and B(B1B2), each on two bits.**



4.2 2-bit comparator

A2	A1	B2	B1		fs	fe	fl
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	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				

4.2 2-bit comparator

1-bit Comparator

1. A=B if

A2=B2 and A1=B1

$$fe = \overline{(A2 \oplus B2)}.\overline{(A1 \oplus B1)}$$

2. A>B if

A2 > B2 or (A2=B2 and A1>B1)

$$fs = A2.\overline{B2} + \overline{(A2 \oplus B2)}.(A1.\overline{B1})$$

3. A<B if

A2 < B2 or (A2=B2 and A1<B1)

$$fl = \overline{A2}.B2 + \overline{(A2 \oplus B2)}.(\overline{A1}.B1)$$

A2	A1	B2	B1		fs	fe	fl
0	0	0	0		0	1	0
0	0	0	1		0	0	1
0	0	1	0		0	0	1
0	0	1	1		0	0	1
0	1	0	0		1	0	0
0	1	0	1		0	1	0
0	1	1	0		0	0	1
0	1	1	1		0	0	1
1	0	0	0		1	0	0
1	0	0	1		1	0	0
1	0	1	0		0	1	0
1	0	1	1		0	0	1
1	1	0	0		1	0	0
1	1	0	1		1	0	0
1	1	1	0		1	0	0
1	1	1	1		0	1	0

4.2 2-bit Comparator

1. A=B if

A2=B2 and A1=B1

$$fe = \overline{(A2 \oplus B2)}.\overline{(A1 \oplus B1)}$$

2. A>B if

A2 > B2 or (A2=B2 and A1>B1)

$$fs = A2.\overline{B2} + \overline{(A2 \oplus B2)}.(A1.\overline{B1})$$

3. A<B if

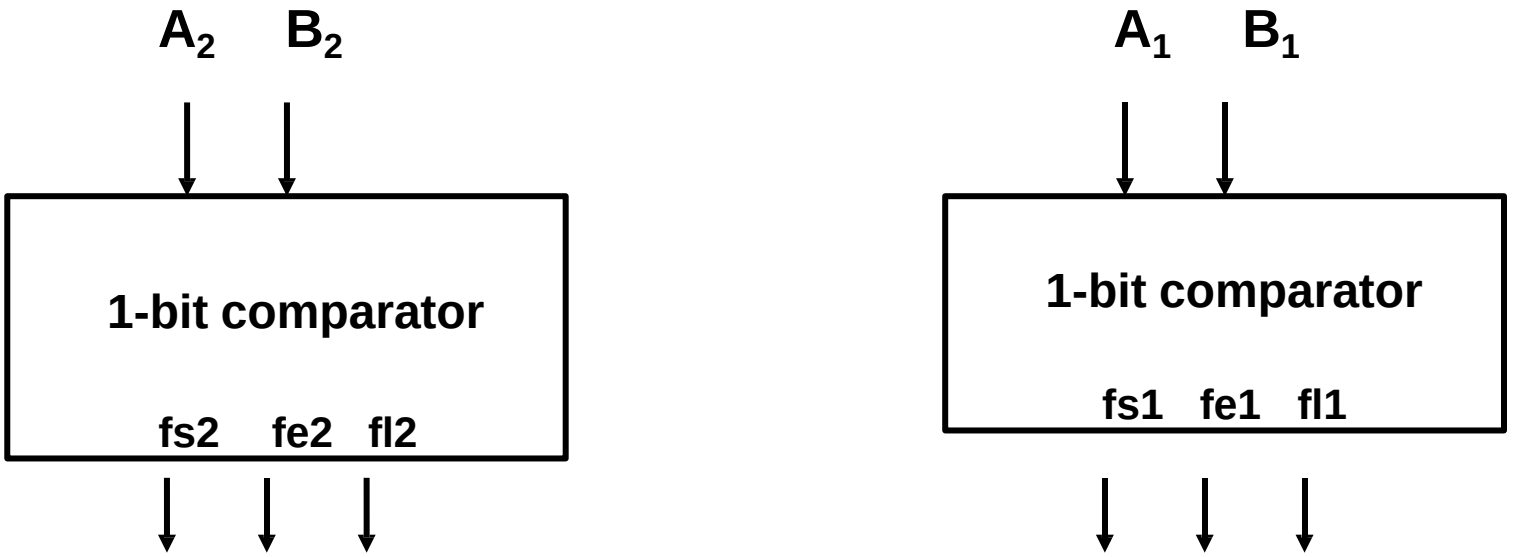
A2 < B2 or (A2=B2 and A1<B1)

$$fl = \overline{A2}.B2 + \overline{(A2 \oplus B2)}.(\overline{A1}.B1)$$

A2	A1	B2	B1		fs	fe	fl
0	0	0	0		0	1	0
0	0	0	1		0	0	1
0	0	1	0		0	0	1
0	0	1	1		0	0	1
0	1	0	0		1	0	0
0	1	0	1		0	1	0
0	1	1	0		0	0	1
0	1	1	1		0	0	1
1	0	0	0		1	0	0
1	0	0	1		1	0	0
1	0	1	0		0	1	0
1	0	1	1		0	0	1
1	1	0	0		1	0	0
1	1	0	1		1	0	0
1	1	1	0		1	0	0
1	1	1	1		0	1	0

4.3. 2-bit comparator with 1-bit comparators

- It is possible to make a 2-bit comparator using 1-bit comparators and logic gates.
- One comparator is used to compare the least significant bits, and another to compare the most significant bits.
- The outputs of the two comparators must be combined to produce the outputs of the final comparator.



1. A=B if

A2=B2 et A1=B1

$$fe = \overline{(A2 \oplus B2)}.\overline{(A1 \oplus B1)} = fe2.fe1$$

2. A>B if

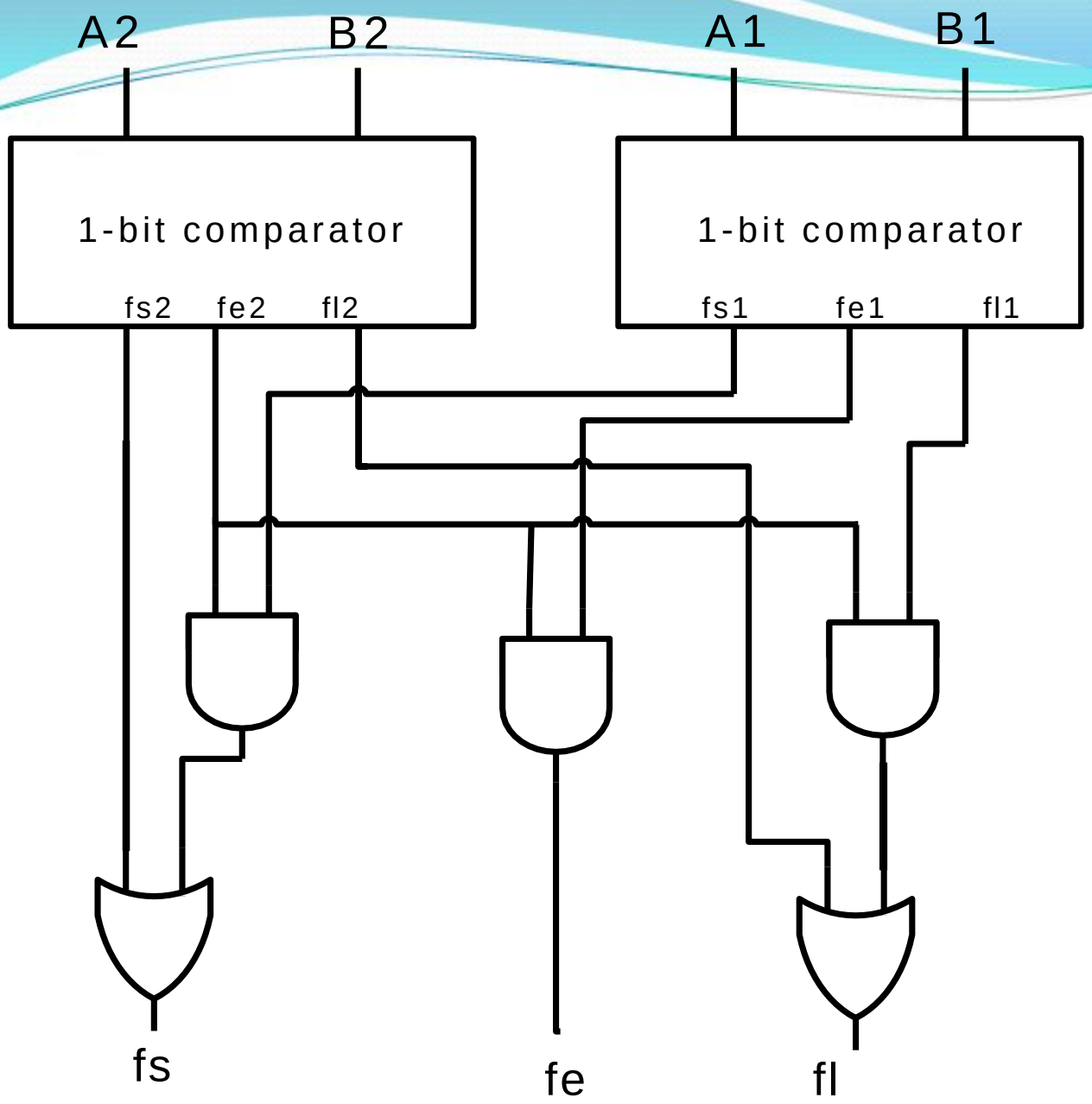
A2 > B2 or (A2=B2 and A1>B1)

$$fs = A2.\overline{B2} + \overline{(A2 \oplus B2)}\ A1.\overline{B1} = fs2 + fs1.fe2$$

3. A<B if

A2 < B2 or (A2=B2 and A1<B1)

$$fl = \overline{A2}.B2 + \overline{(A2 \oplus B2)}.\overline{(A1.B1)} = fl2 + fe2.fl1$$


A2
B2
A1
B1
1-bit comparator
fs2
fe2
fl2
1-bit comparator
fs1
fe1
fl1
fs
fe
fl

4.4 Comparator with cascading inputs

- Note that :

✓ If A2 >B2 then A > B

✓ If A2<B2 then A < B

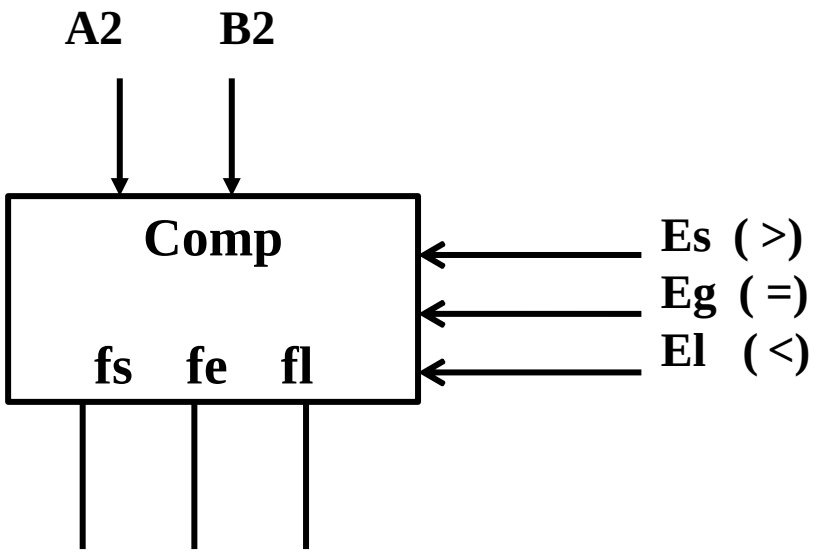
- On the other hand, if A2=B2, then the result of the comparison of the least significant bits must be taken into account.

- To do this, we add inputs to the comparator that tell us the result of the previous comparison.

- These inputs are called cascading inputs.

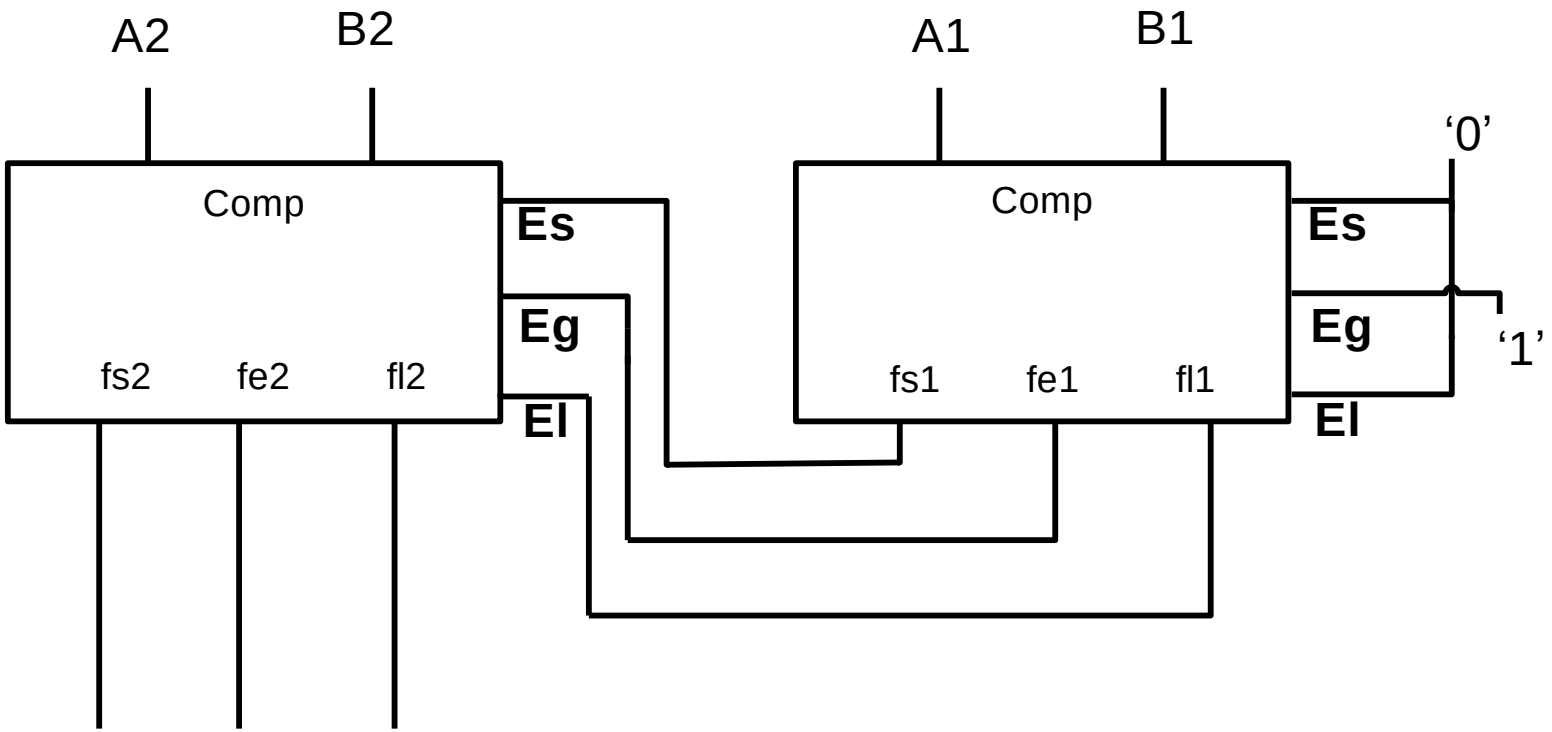
4.4 Comparator with cascading inputs

A2	B2	Es	Eg	El		fs	fe	fl
A2>B2		X	X	X		1	0	0
A2<B2		X	X	X		0	0	1
A2=B2		1	0	0		1	0	0
		0	1	0		0	1	0
		0	0	1		0	0	1



fs= (A2>B2) or (A2=B2).Es
fl= (A2<B2) or (A2=B2).El
fe= (A2=B2).Eg

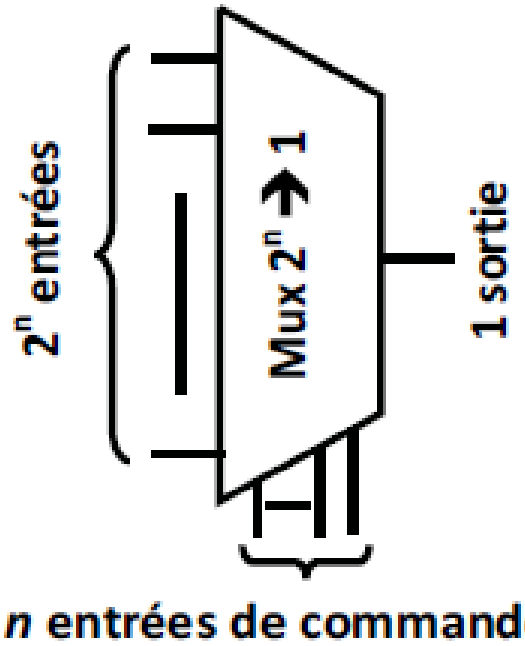
4.4 Comparator with cascading inputs



5. The Multiplexer

- **Definition:** A multiplexer is a combinational logic circuit:
  - n control inputs and
  - a single output.

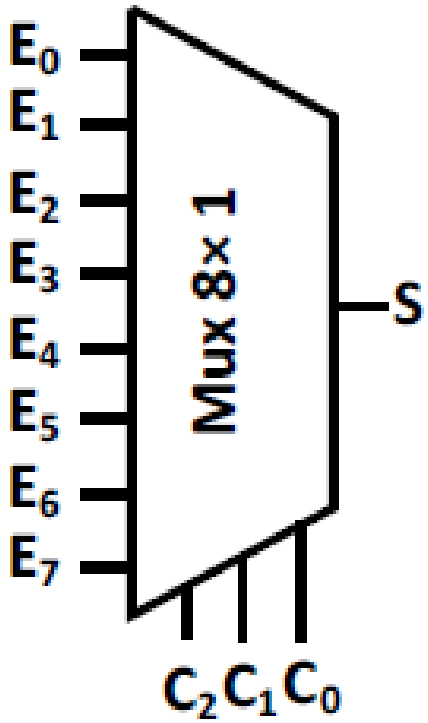
- It switches the value of the input line indicated in its control inputs to the output line.



5. The 8x1 multiplexer

- For example: Circuit synthesis (8x1 multiplexer)

Les entrées/sorties.



La table de vérité.

C_2	C_1	C_0	S
0	0	0	E_0
0	0	1	E_1
0	1	0	E_2
0	1	1	E_3
1	0	0	E_4
1	0	1	E_5
1	1	0	E_6
1	1	1	E_7

5. The 8x1 multiplexer

- Circuit synthesis (8x1 multiplexer)

La table de vérité.

C_2	C_1	C_0	S
0	0	0	E_0
0	0	1	E_1
0	1	0	E_2
0	1	1	E_3
1	0	0	E_4
1	0	1	E_5
1	1	0	E_6
1	1	1	E_7

Les fonctions logiques.

$$S = \overline{C_2}.\overline{C_1}.\overline{C_0}.E_0 + \overline{C_2}.\overline{C_1}.C_0.E_1 + \overline{C_2}.C_1.\overline{C_0}.E_2 + \overline{C_2}.C_1.C_0.E_3 + C_2.\overline{C_1}.\overline{C_0}.E_4 + C_2.\overline{C_1}.C_0.E_5 + C_2.C_1.\overline{C_0}.E_6 + C_2.C_1.C_0.E_7$$

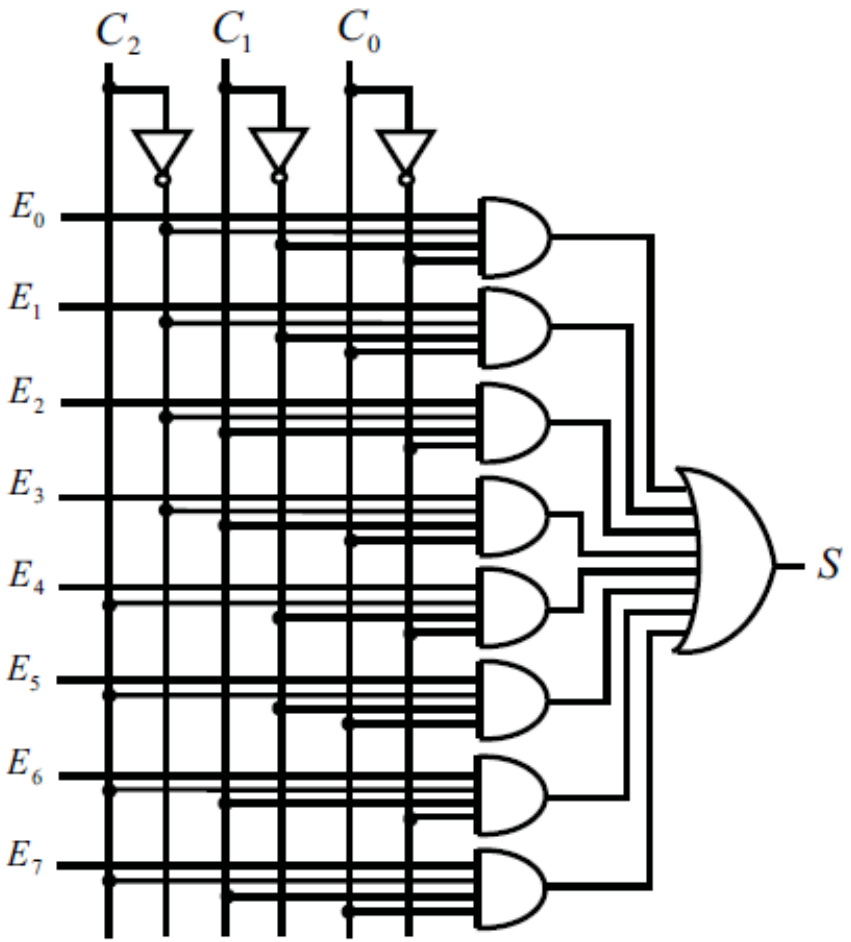
5. The 8x1 multiplexer

- Circuit synthesis (8x1 multiplexer):

Les fonctions logiques.

$$S = \overline{C_2}.\overline{C_1}.\overline{C_0}.E_0 + \overline{C_2}.\overline{C_1}.C_0.E_1 + \overline{C_2}.C_1.\overline{C_0}.E_2 + \overline{C_2}.C_1.C_0.E_3 + C_2.\overline{C_1}.\overline{C_0}.E_4 + C_2.\overline{C_1}.C_0.E_5 + C_2.C_1.\overline{C_0}.E_6 + C_2.C_1.C_0.E_7$$

Le schéma du circuit.



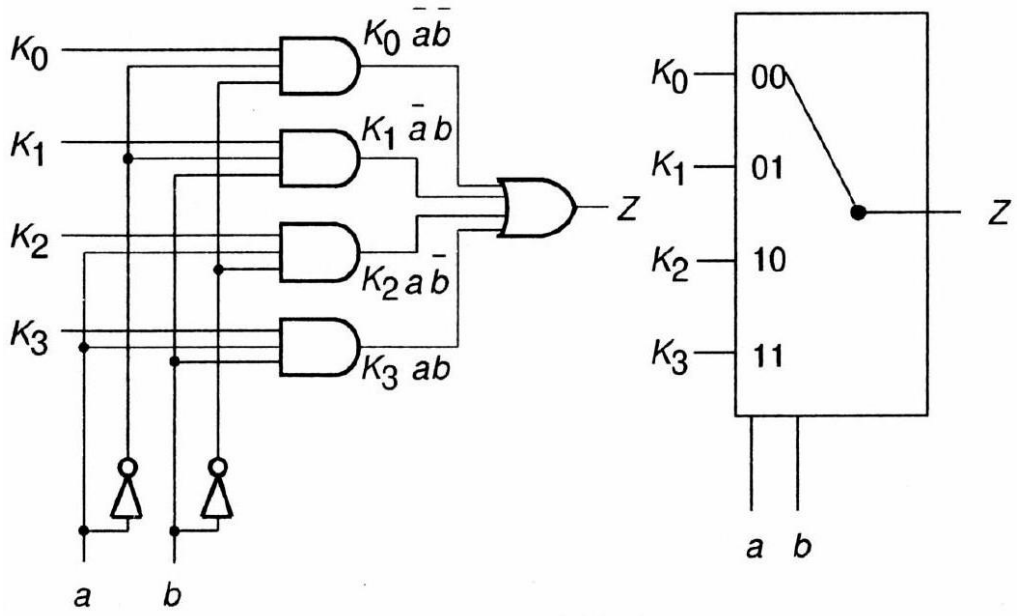
5. The Multiplexer

The multiplexer is a combinational **selector** circuit with :

- 2^n information inputs,
- n control inputs and
- a single output.

Its role is to select one of the inputs, using control signals, and link it to the output.

a	b	Z
0	0	K_0
0	1	K_1
1	0	K_2
1	1	K_3



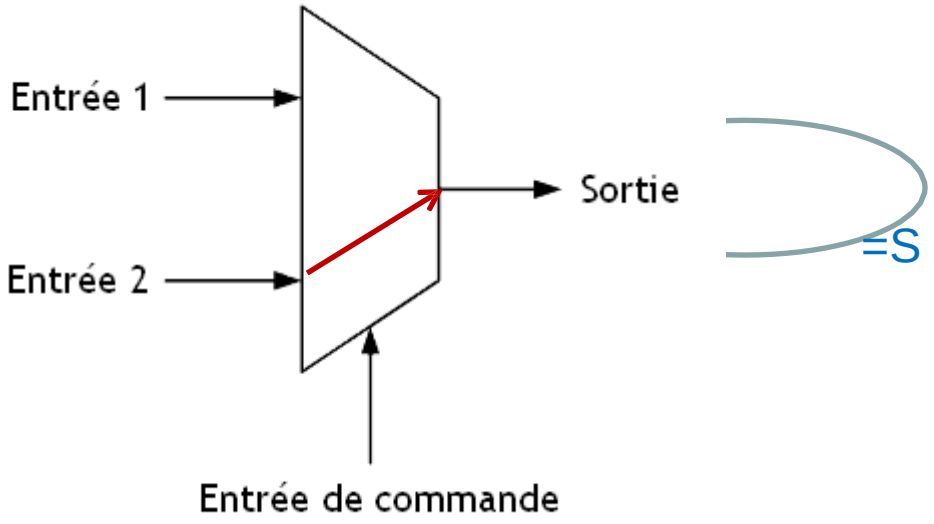
MUX (2 variables)

5.1 Multiplexer 2 →1

Description of 2 to 1 multiplexer behavior:

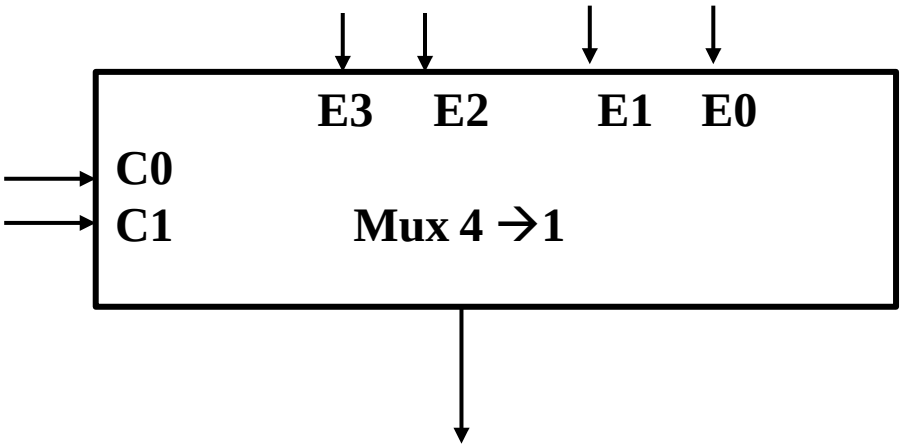
- Output S takes the value of the data input:
- En1 when Com selection input is active (level 1).
- En2 when the Com selection input is inactive (level 0).
- The Com input is therefore used to route information arriving via the En1 input or the En2 input to the S output.

Description of 2 to 1 multiplexer behavior :



5.2 Multiplexer 4 →1

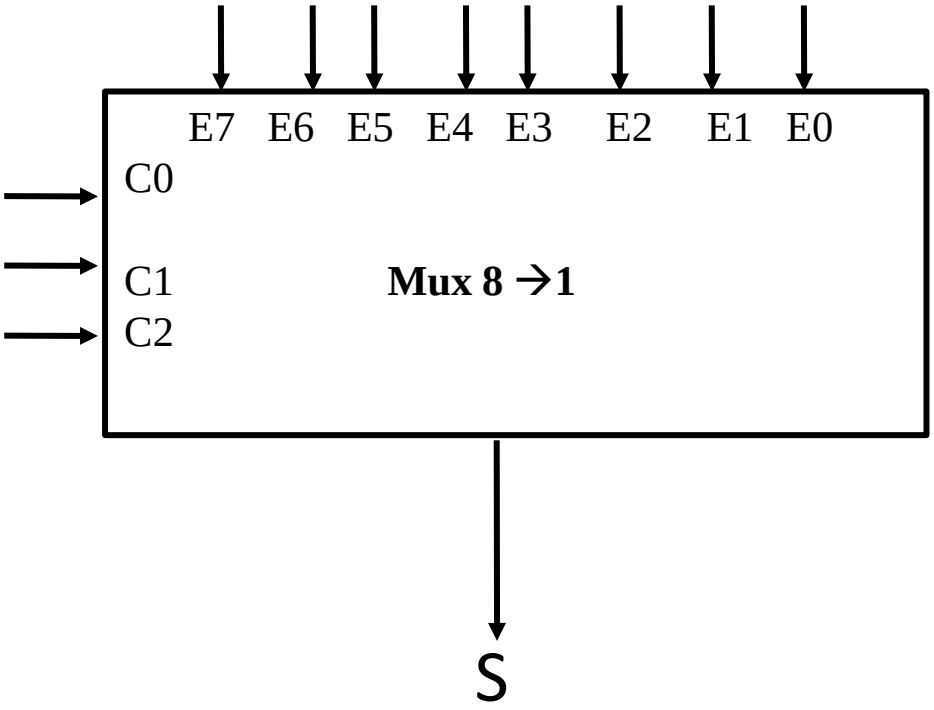
C1	C0		S
0	0		E0
0	1		E1
1	0		E2
1	1		E3



$$S = \overline{C1}.\overline{C0}.(E0) + \overline{C1}.C0(E1) + C1.\overline{C0}\,(E2) + C1.C0(E3)$$

5.3 Multiplexer 8→1

C2	C1	C0		S
0	0	0		E0
0	0	1		E1
0	1	0		E2
0	1	1		E3
1	0	0		E4
1	0	1		E5
1	1	0		E6
1	1	1		E7



$$S = \overline{C2}.\overline{C1}.\overline{C0}.(E0) + \overline{C2}.\overline{C1}.C0(E1) + \overline{C2}.C1.\overline{C0}(E2) + \overline{C2}.C1.C0(E3) +$$
$$C2.\overline{C1}.\overline{C0}(E4) + C2.\overline{C1}.C0(E5) + C2.C1.\overline{C0}(E6) + C2.C1.C0(E7)$$

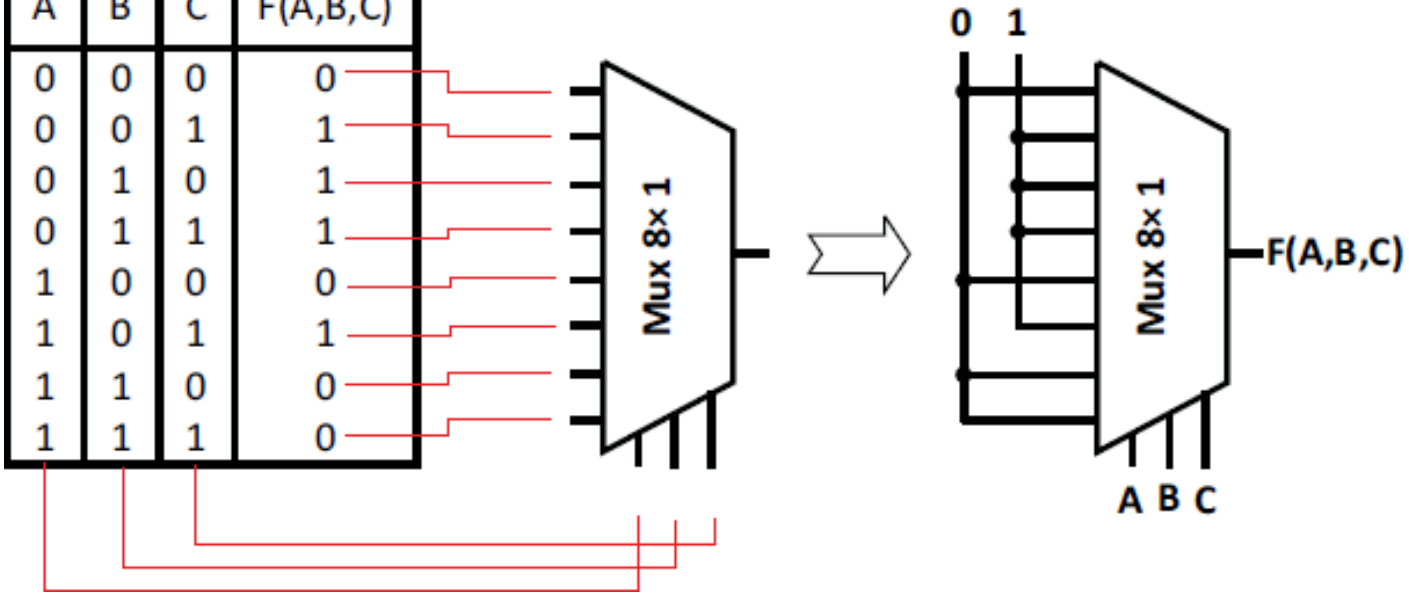
Logic functions using multiplexers

Any number of logic functions can be generated using multiplexers and basic logic gates.

Simply connect the variables of the function to be generated to the various inputs of the multiplexer (standard inputs and control inputs).

Example 1 : Perform the function $F(A, B, C) = \bar{B}C + \bar{A}B$ using an 8x1 multiplexer.

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic functions using multiplexers

Any number of logic functions can be generated using multiplexers and basic logic gates.

Simply connect the variables of the function to be generated to the various inputs of the multiplexer (standard inputs and control inputs).

Example 1 : Perform the function using an 8x1 multiplexer

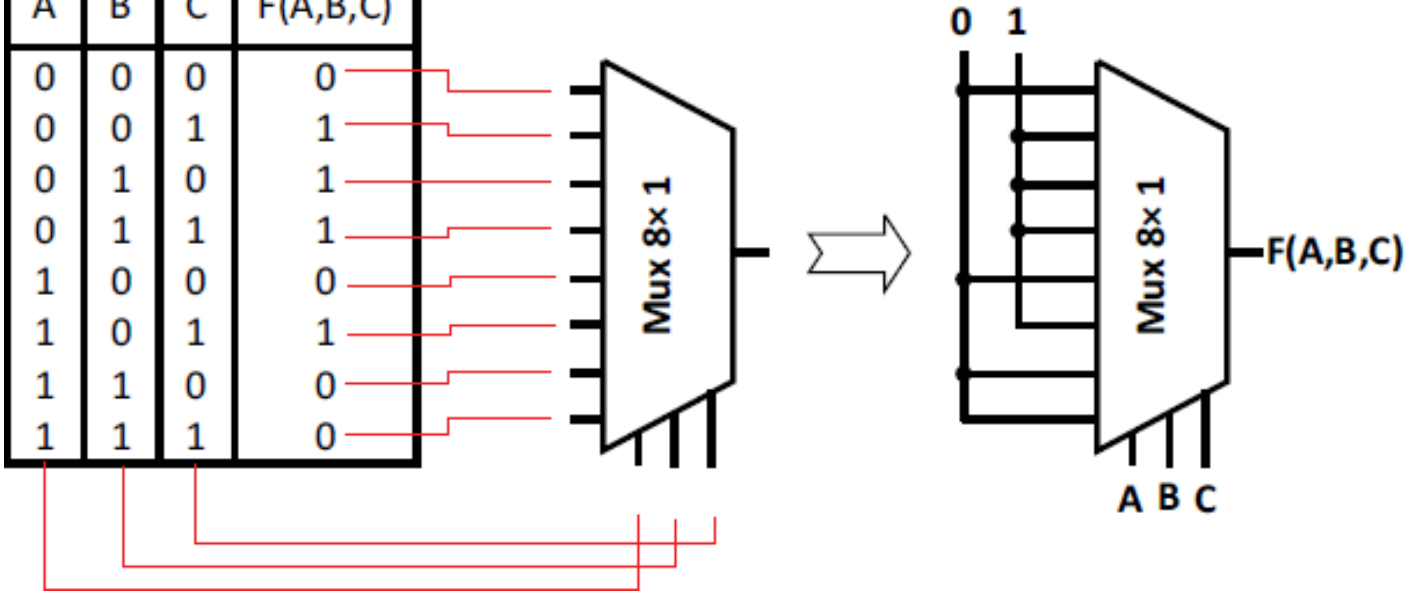
$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

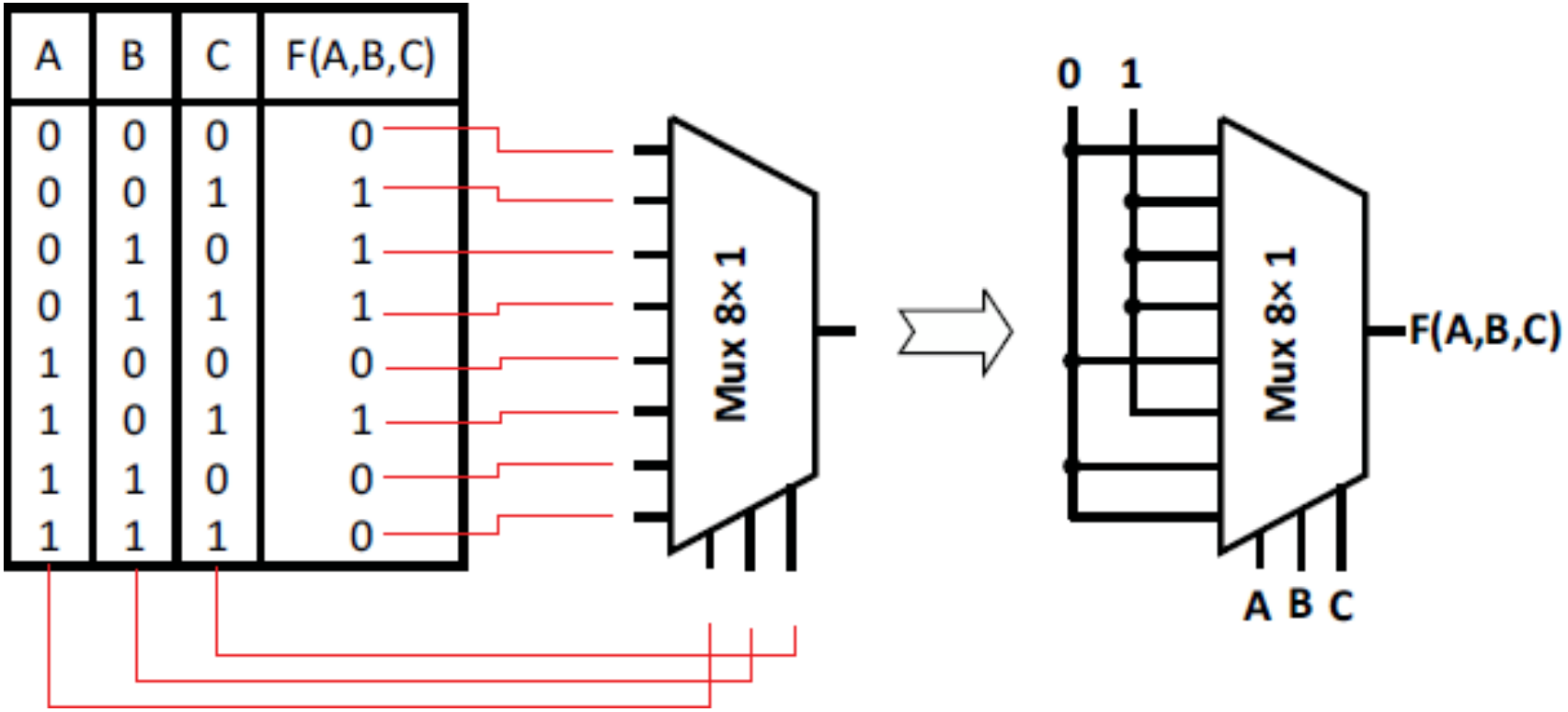
A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic functions using multiplexers

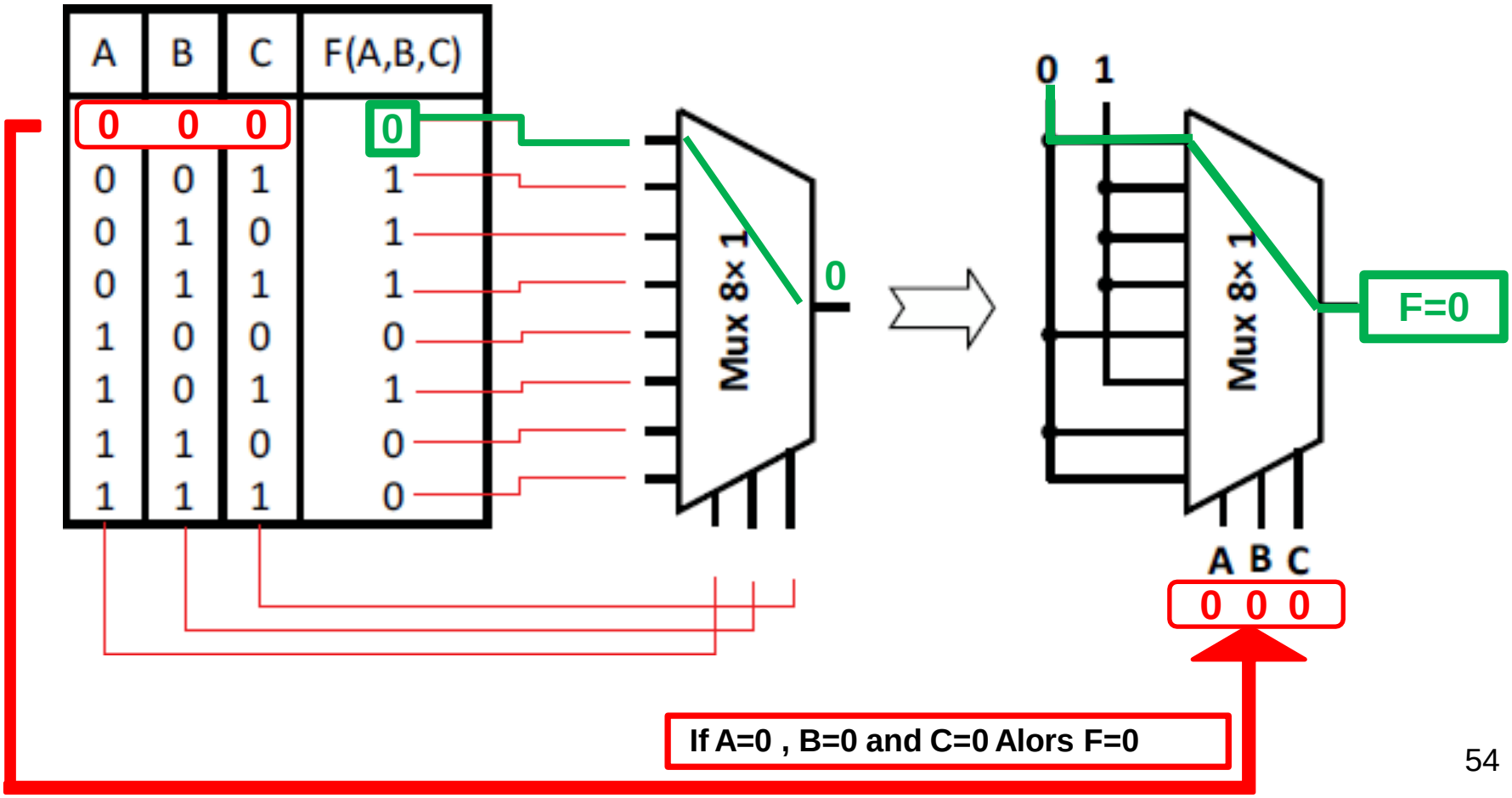
$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



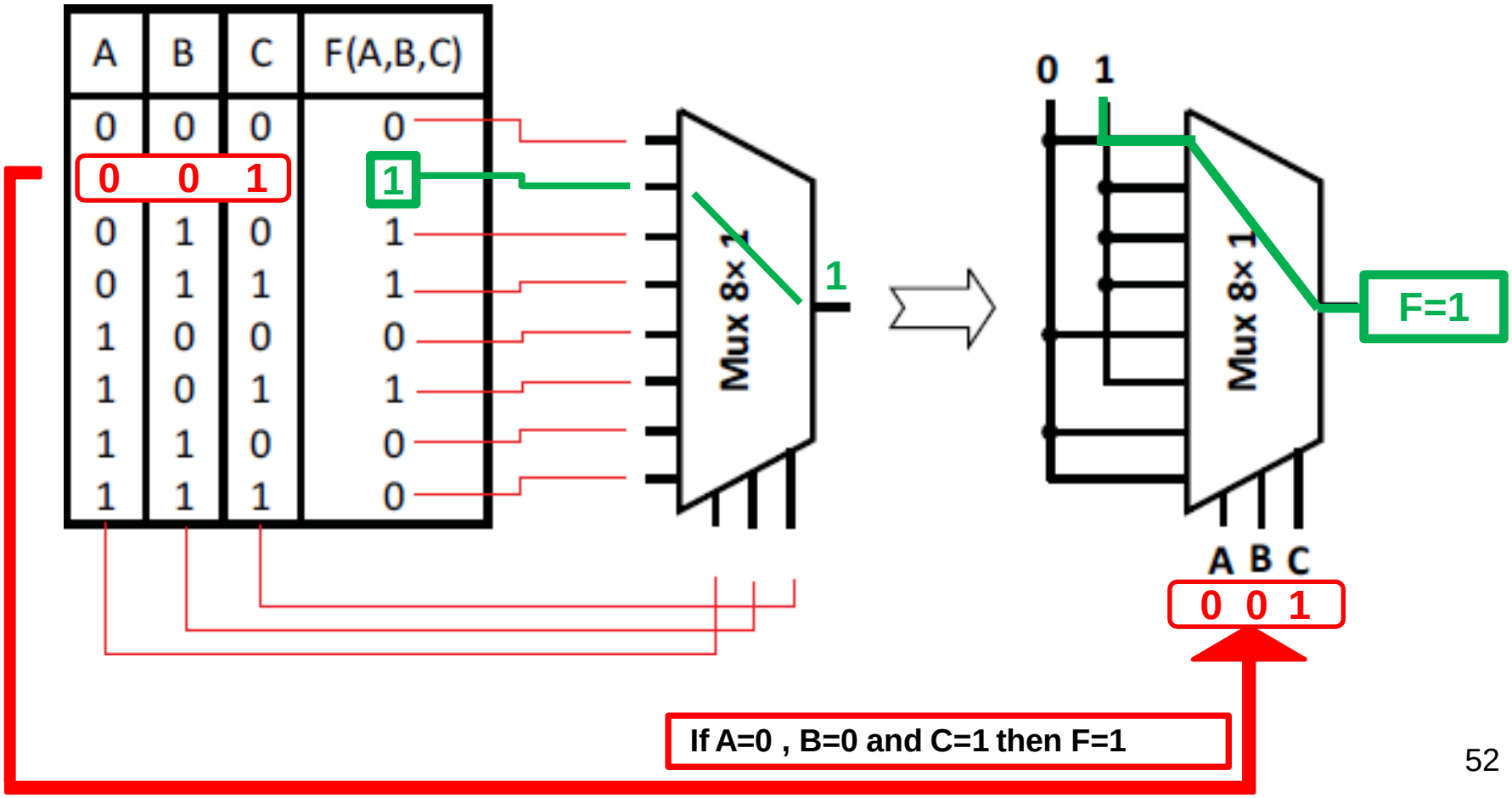
Logic functions using multiplexers

$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



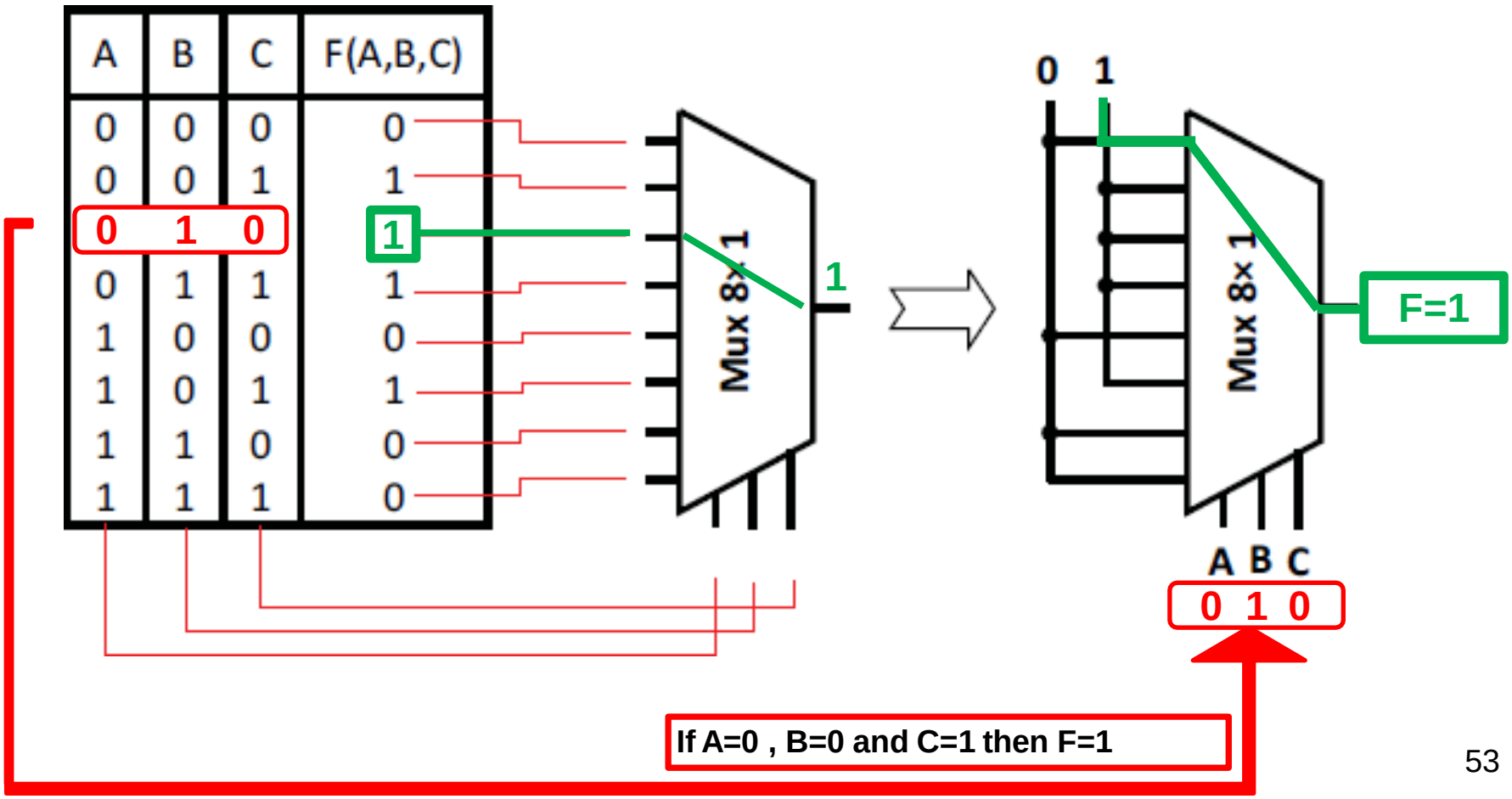
Logic functions using multiplexers

$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



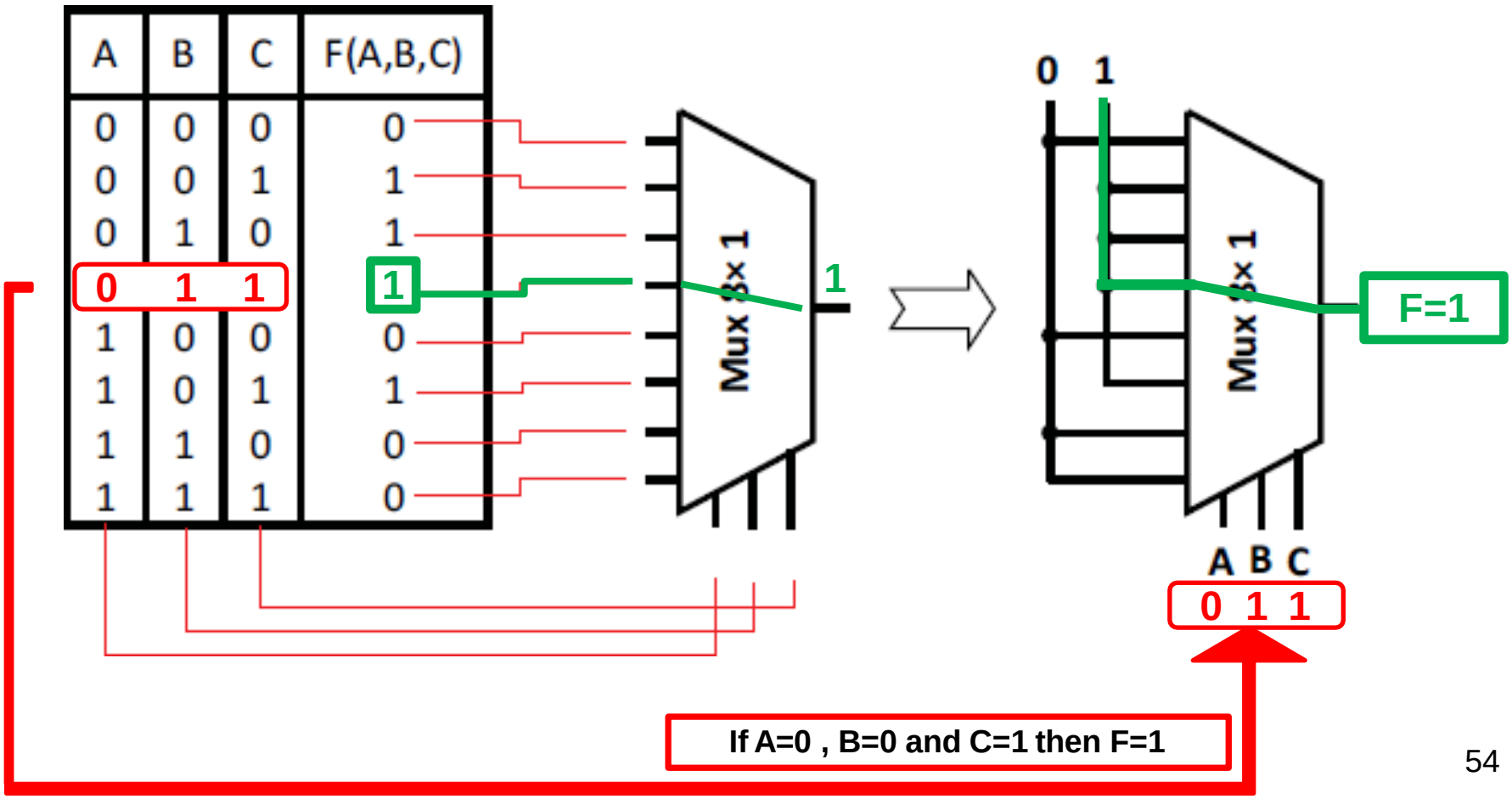
Logic functions using multiplexers

$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



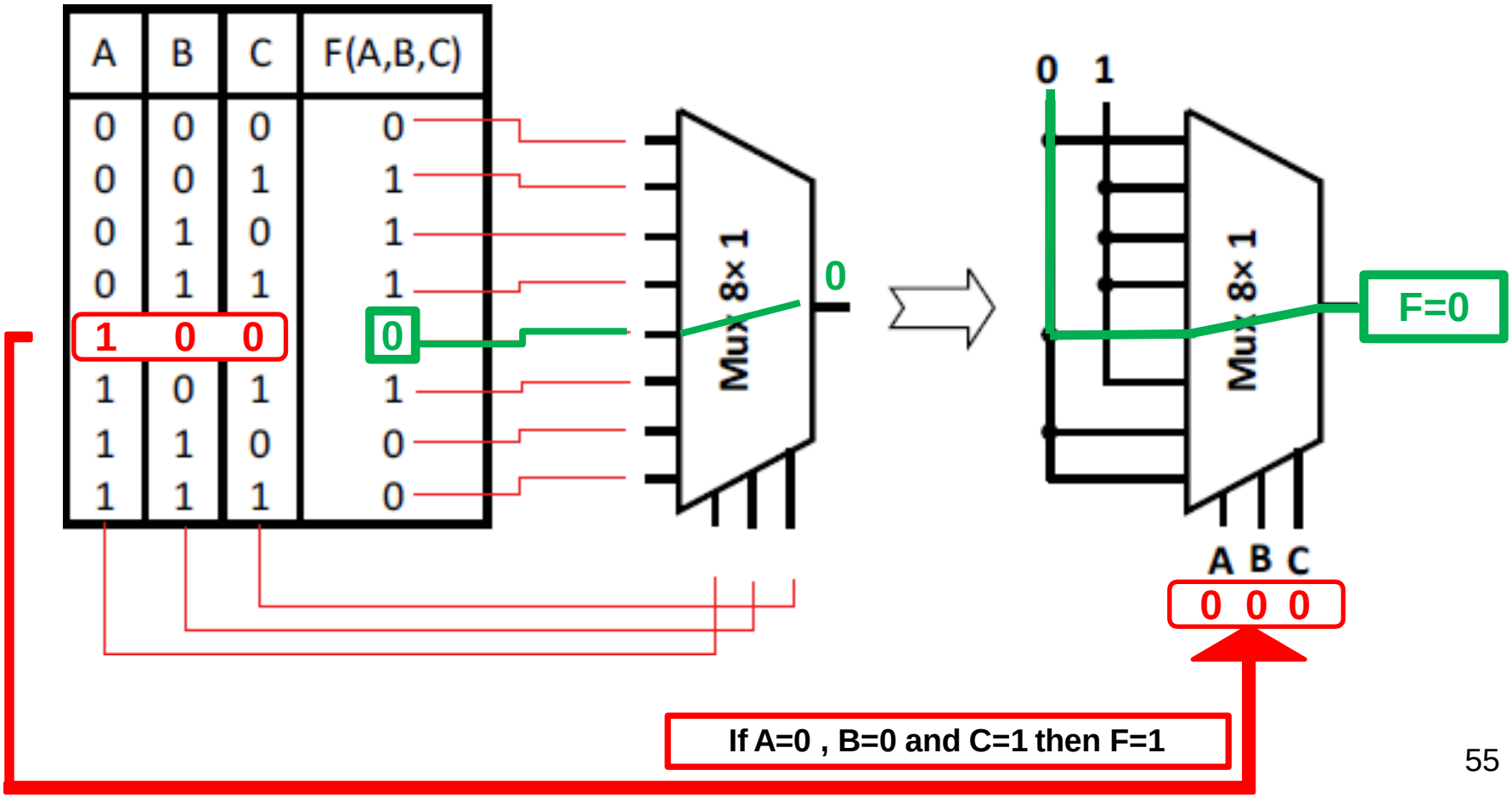
Logic functions using multiplexers

$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



Logic functions using multiplexers

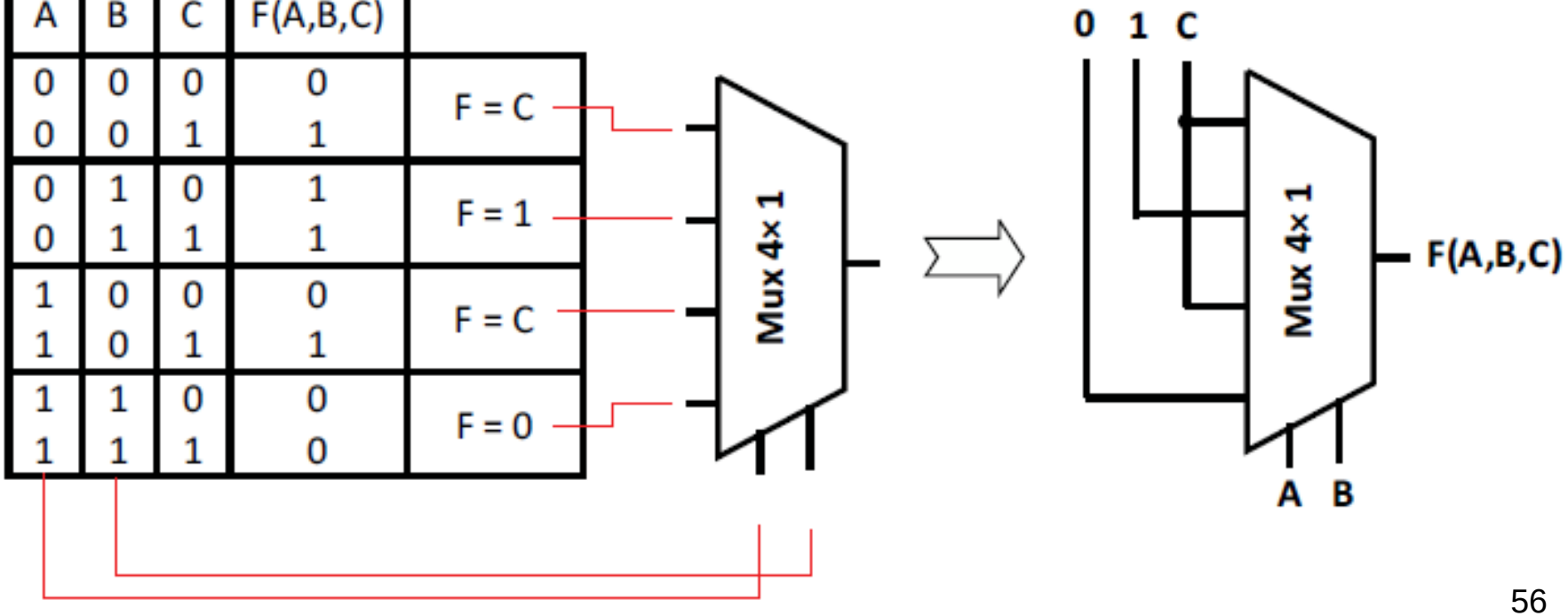
$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



Logic functions using multiplexers

Example 2 : Perform the function $F(A,B,C) = \overline{A}.B + \overline{B}.C$ using a multiplexer 4x1.

A	B	C	F(A,B,C)	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = 1
0	1	1	1	
1	0	0	0	F = C
1	0	1	1	
1	1	0	0	F = 0
1	1	1	0	



Logic functions using multiplexers

Example 2 : Perform the function $F(A,B,C) = \overline{A}.B + \overline{B}.C$ using a multiplexer 4x1.

$$F(A,B,C) = \overline{A}.B + \overline{B}.C$$

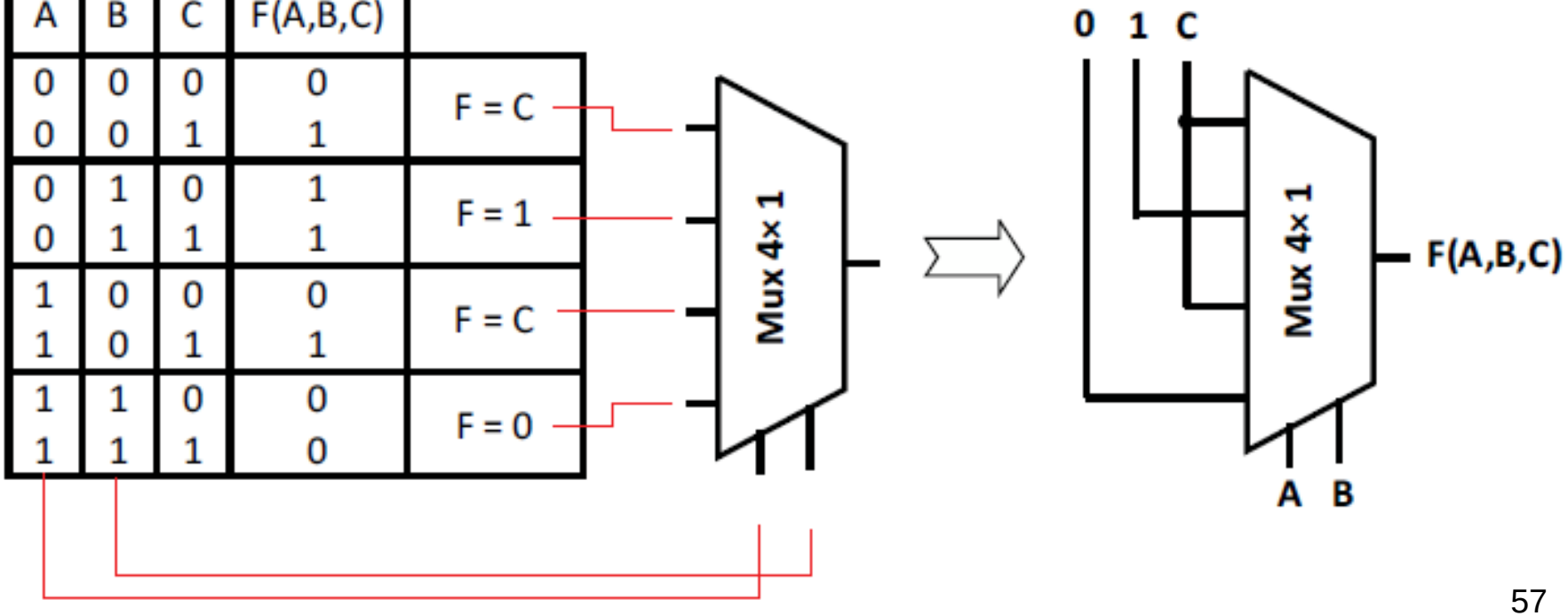
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

$$F(A,B,C) = \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C})$$

$$F(A,B,C) = (\overline{A}\,\overline{B}C + A\,\overline{B}C) + (\overline{A}B\overline{C} + \overline{A}BC)$$

$$F(A,B,C) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\,\overline{B}C$$

A	B	C	F(A,B,C)	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = 1
0	1	1	1	
1	0	0	0	F = C
1	0	1	1	
1	1	0	0	F = 0
1	1	1	0	



Logic functions using multiplexers

Example 2 : Perform the function $F(A,B,C) = \overline{A}.B + \overline{B}.C$ using a multiplexer 4x1.

$$F(A,B,C) = \overline{A}.B + \overline{B}.C$$

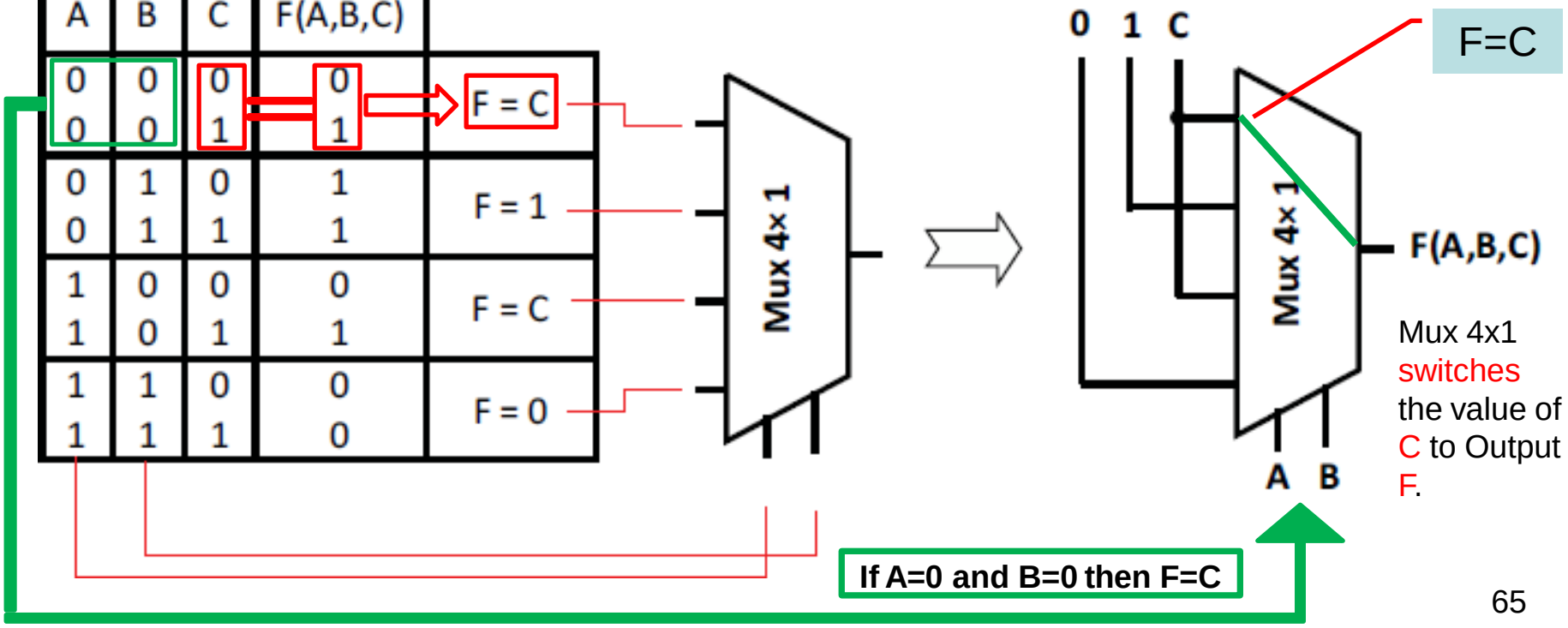
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

$$F(A,B,C) = \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C})$$

$$F(A,B,C) = (\overline{A}\,\overline{B}C + A\,\overline{B}C) + (\overline{A}B\overline{C} + \overline{A}BC)$$

$$F(A,B,C) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\,\overline{B}C$$

A	B	C	F(A,B,C)	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = 1
0	1	1	1	
1	0	0	0	F = C
1	0	1	1	
1	1	0	0	F = 0
1	1	1	0	



Mux 4x1 switches the value of C to Output F.

Fonctions logiques via des multiplexeurs

Example 2 : Perform the function $F(A,B,C) = \overline{A}.B + \overline{B}.C$ using a multiplexer 4x1.

$$F(A,B,C) = \overline{A}.B + \overline{B}.C$$

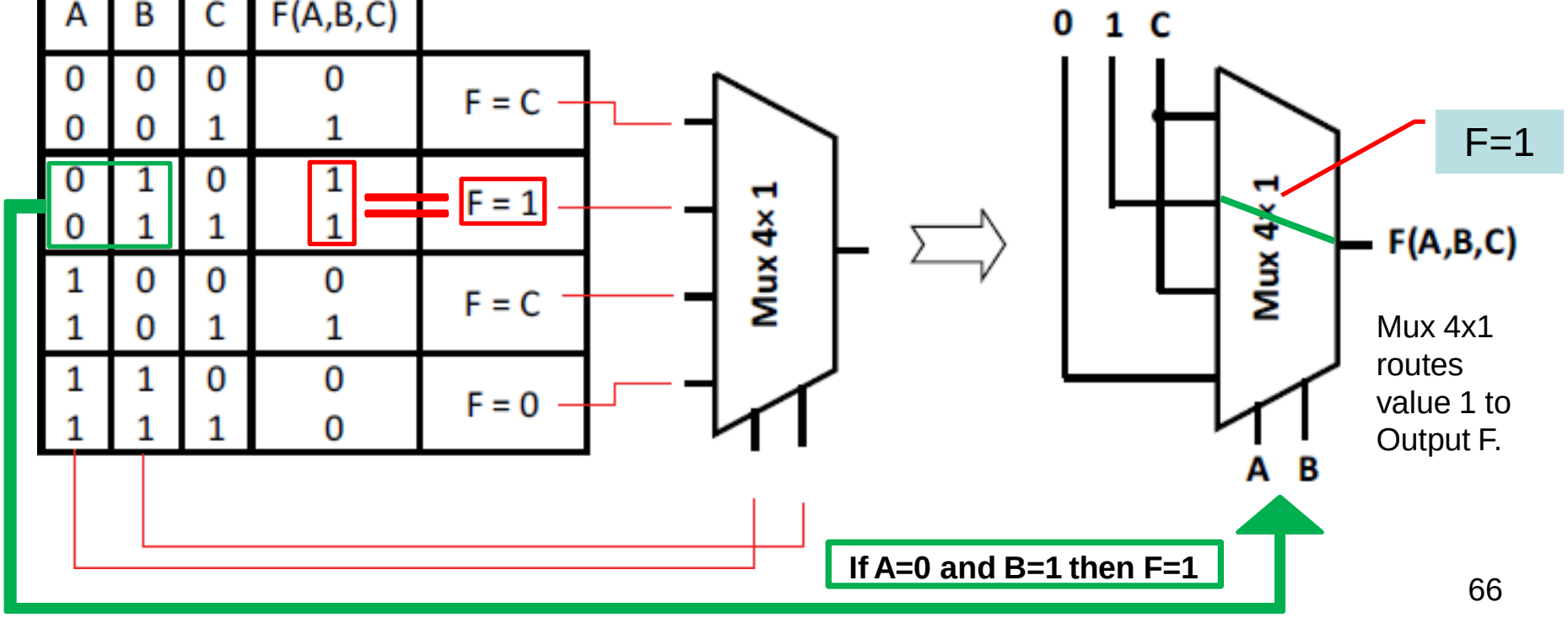
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

$$F(A,B,C) = \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C})$$

$$F(A,B,C) = (\overline{A}\,\overline{B}C + A\,\overline{B}C) + (\overline{A}B\overline{C} + \overline{A}BC)$$

$$F(A,B,C) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\,\overline{B}C$$

A	B	C	F(A,B,C)	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = 1
0	1	1	1	
1	0	0	0	F = C
1	0	1	1	
1	1	0	0	F = 0
1	1	1	0	



Fonctions logiques via des multiplexeurs

Example 2 : Perform the function $F(A,B,C) = \overline{A}.B + \overline{B}.C$ using a multiplexer 4x1.

$$F(A,B,C) = \overline{A}.B + \overline{B}.C$$

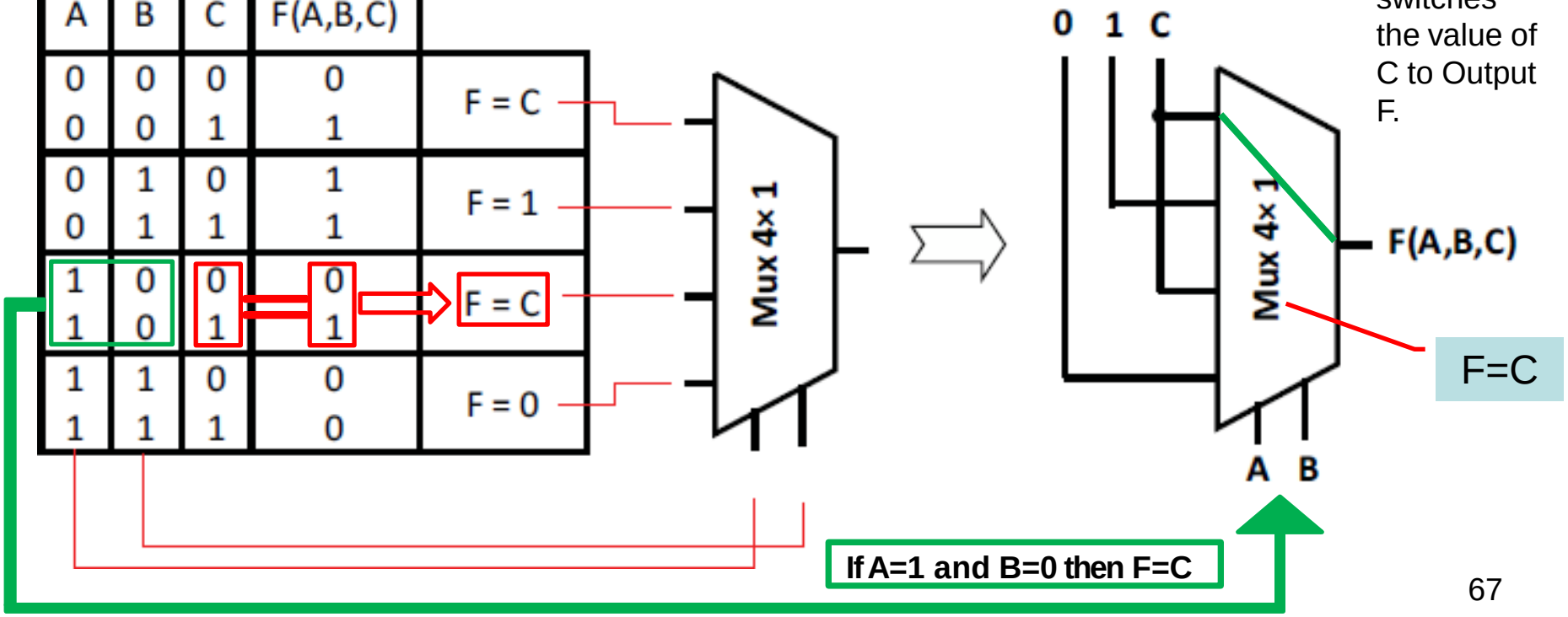
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

$$F(A,B,C) = \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C})$$

$$F(A,B,C) = (\overline{A}\,\overline{B}C + A\,\overline{B}C) + (\overline{A}B\overline{C} + \overline{A}BC)$$

$$F(A,B,C) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\,\overline{B}C$$

A	B	C	F(A,B,C)	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = 1
0	1	1	1	
1	0	0	0	F = C
1	0	1	1	
1	1	0	0	F = 0
1	1	1	0	



Mux 4x1 switches the value of C to Output F.

Fonctions logiques via des multiplexeurs

Example 2 : Perform the function $F(A,B,C) = \overline{A}.B + \overline{B}.C$ using a multiplexer 4x1.

$$F(A,B,C) = \overline{A}.B + \overline{B}.C$$

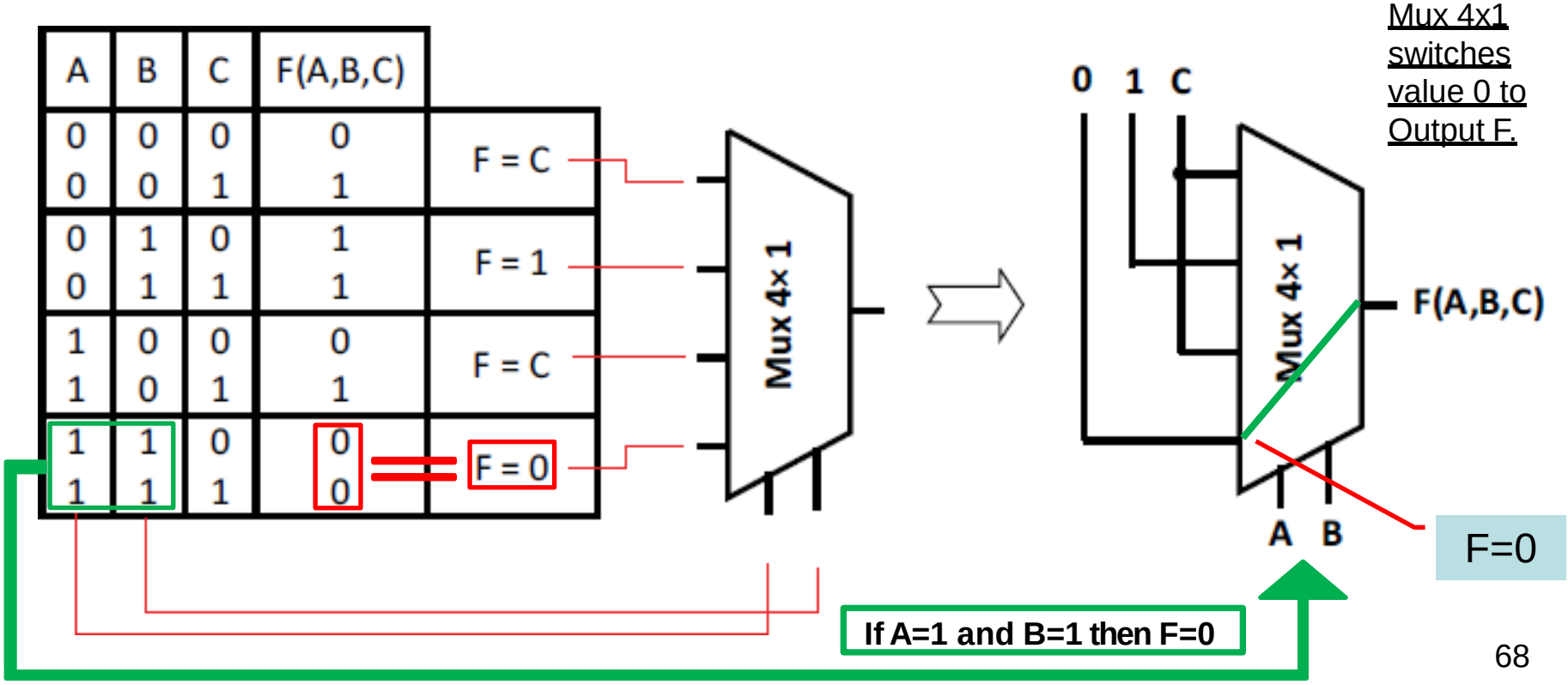
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

$$F(A,B,C) = \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C})$$

$$F(A,B,C) = (\overline{A}\,\overline{B}C + A\,\overline{B}C) + (\overline{A}B\overline{C} + \overline{A}BC)$$

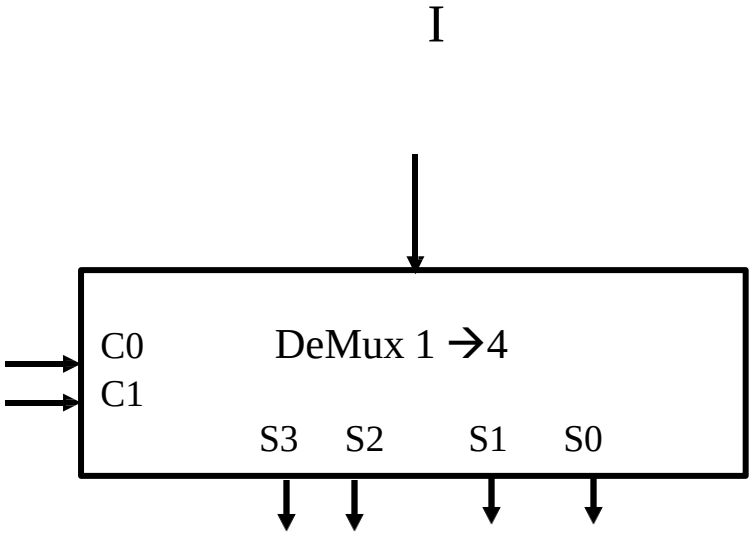
$$F(A,B,C) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\,\overline{B}C$$

A	B	C	F(A,B,C)	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = 1
0	1	1	1	
1	0	0	0	F = C
1	0	1	1	
1	1	0	0	F = 0
1	1	1	0	



6. Demultiplexers

- It plays the opposite role to a multiplexer.
- It allows information to be passed to one of the outputs according to the values of the control inputs.
- It has :
  - a single input
  - 2^n outputs
  - N selection inputs/lines (commands)

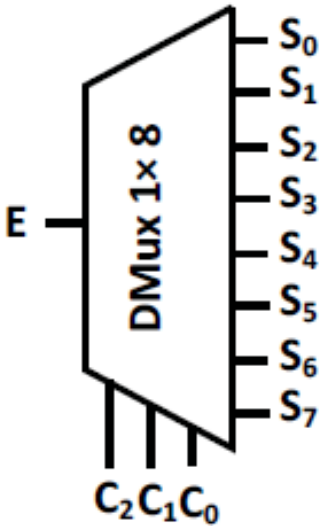


6. Demultiplexers

- **Definition:** A demultiplexer is a combinational logic circuit with :
  a single input, n control inputs/lines and 2^n outputs.
- It switches the value of the input line to the output line indicated in its control inputs.

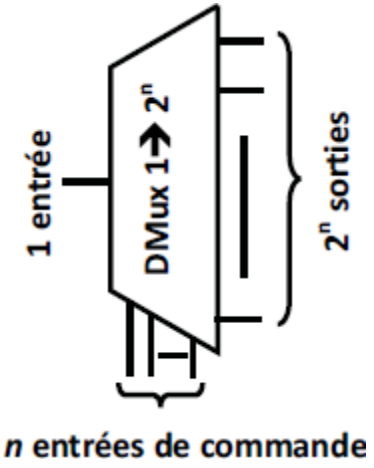
Circuit synthesis (1x8 demultiplexer)

Les entrées/sorties.



La table de vérité.

C_2	C_1	C_0	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	E	0	0	0	0	0	0	0
0	0	1	0	E	0	0	0	0	0	0
0	1	0	0	0	E	0	0	0	0	0
0	1	1	0	0	0	E	0	0	0	0
1	0	0	0	0	0	0	E	0	0	0
1	0	1	0	0	0	0	0	E	0	0
1	1	0	0	0	0	0	0	0	E	0
1	1	1	0	0	0	0	0	0	0	E



6. Demultiplexers

La table de vérité.

C_2	C_1	C_0	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	E	0	0	0	0	0	0	0
0	0	1	0	E	0	0	0	0	0	0
0	1	0	0	0	E	0	0	0	0	0
0	1	1	0	0	0	E	0	0	0	0
1	0	0	0	0	0	0	E	0	0	0
1	0	1	0	0	0	0	0	E	0	0
1	1	0	0	0	0	0	0	0	E	0
1	1	1	0	0	0	0	0	0	0	E

Les fonctions logiques.

$$S_0 = \overline{C_2}.\overline{C_1}.\overline{C_0}.E$$

$$S_1 = \overline{C_2}.\overline{C_1}.C_0.E$$

$$S_2 = \overline{C_2}.C_1.\overline{C_0}.E$$

$$S_3 = \overline{C_2}.C_1.C_0.E$$

$$S_4 = C_2.\overline{C_1}.\overline{C_0}.E$$

$$S_5 = C_2.\overline{C_1}.C_0.E$$

$$S_6 = C_2.C_1.\overline{C_0}.E$$

$$S_7 = C_2.C_1.C_0.E$$

6. Demultiplexers

Les fonctions logiques.

$$S_0 = \overline{C_2}.\overline{C_1}.\overline{C_0}.E$$

$$S_1 = \overline{C_2}.\overline{C_1}.C_0.E$$

$$S_2 = \overline{C_2}.C_1.\overline{C_0}.E$$

$$S_3 = \overline{C_2}.C_1.C_0.E$$

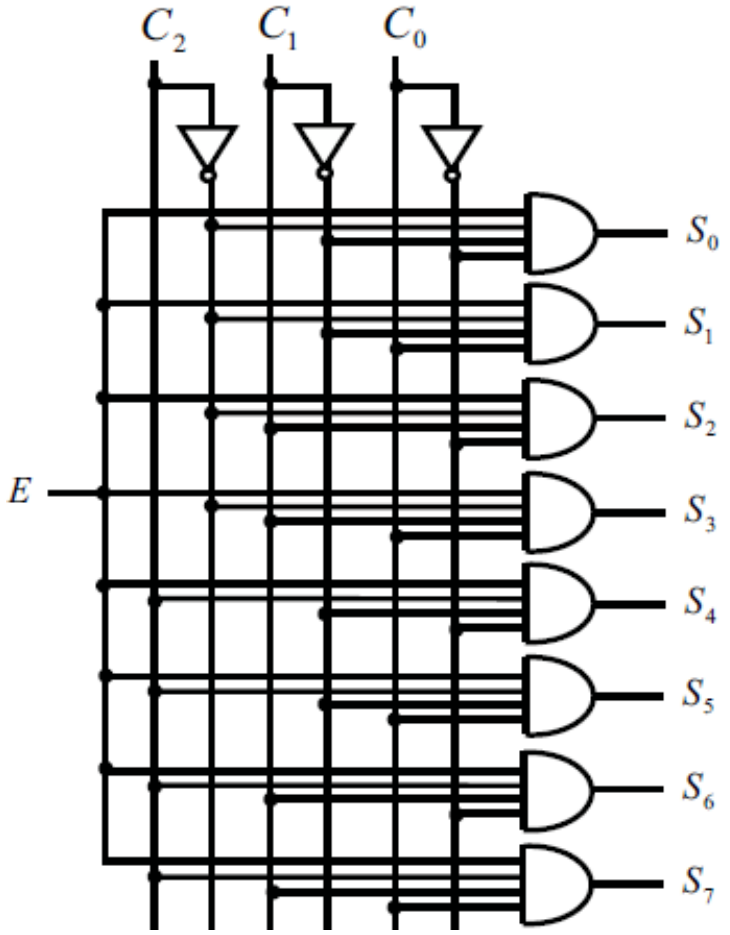
$$S_4 = C_2.\overline{C_1}.\overline{C_0}.E$$

$$S_5 = C_2.\overline{C_1}.C_0.E$$

$$S_6 = C_2.C_1.\overline{C_0}.E$$

$$S_7 = C_2.C_1.C_0.E$$

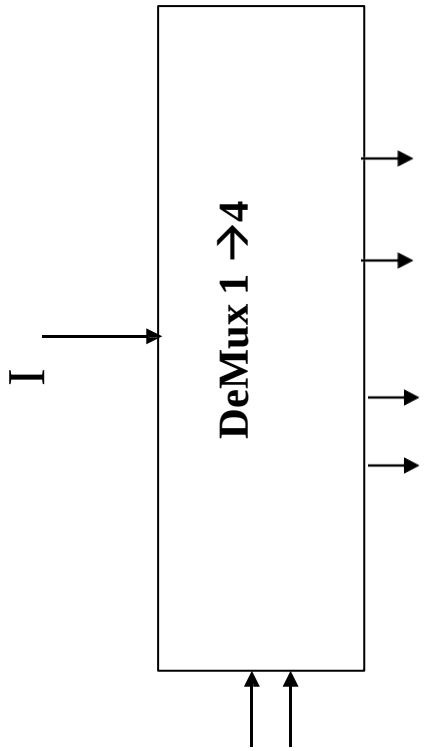
Le schéma du circuit.



6.1 Demultiplexer 1→4

C1	C0		S3	S2	S1	S0

- Demultiplexer 1→4 circuit design



6.1 Demultiplexer 1→4

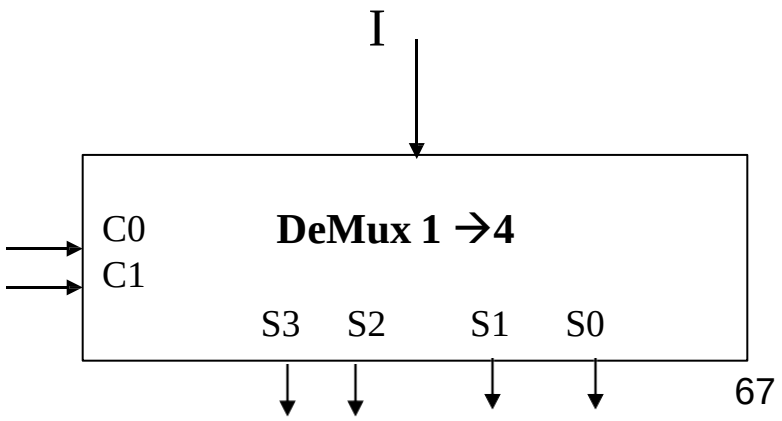
C1	C0		S3	S2	S1	S0
0	0		0	0	0	i
0	1		0	0	i	0
1	0		0	i	0	0
1	1		i	0	0	0

$$S0 = \overline{C1}.\overline{C0}.(I)$$

$$S1 = \overline{C1}.C0.(I)$$

$$S2 = C1.\overline{C0}.(I)$$

$$S3 = C1.C0.(I)$$

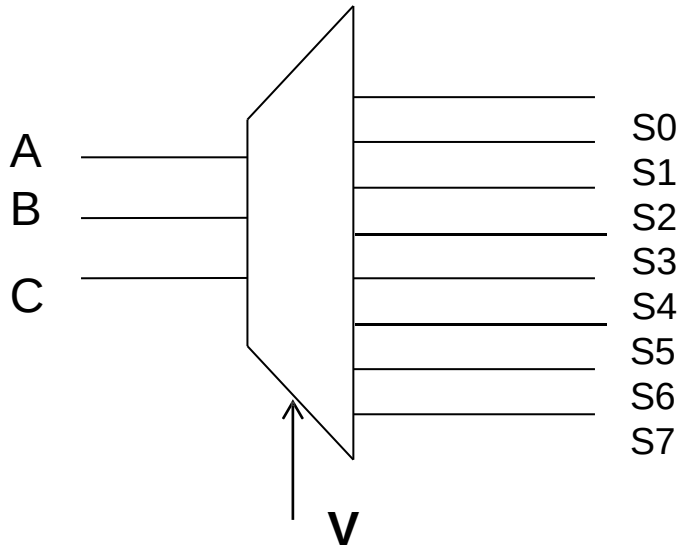


7. The binary decoder

It is a combinational circuit consisting of :

- n : data inputs
- 2^n outputs
- For each input combination, only one output is active at a time.

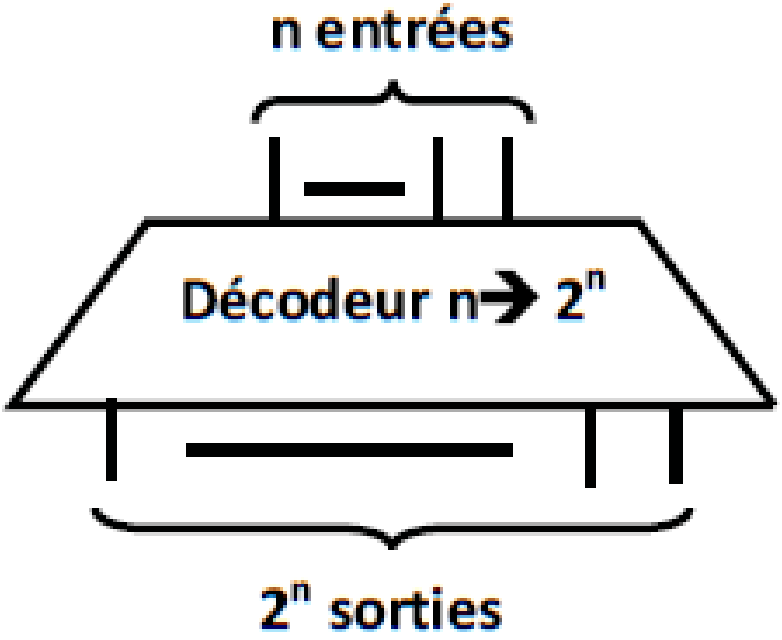
3→8 Decoder

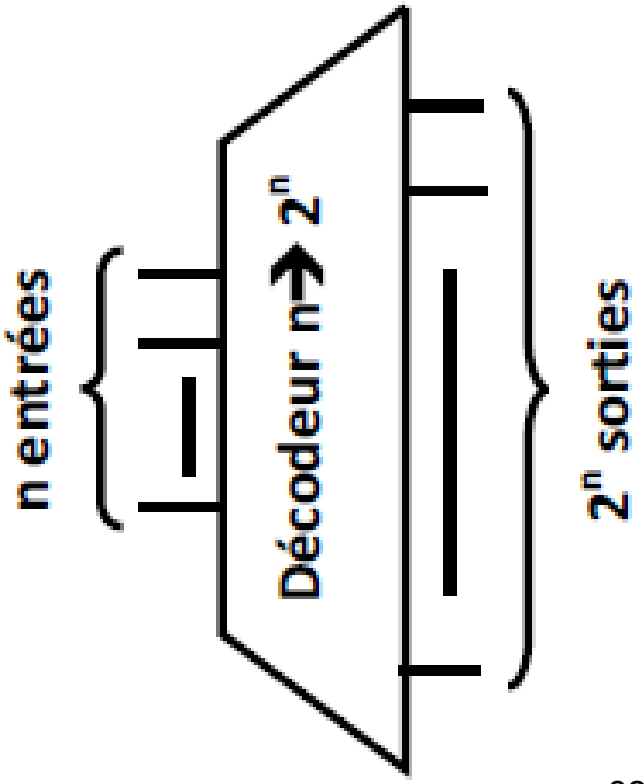


7. The binary decoder

Definition: An n-bit decoder is a combinatorial logic circuit with n inputs and 2^n outputs.

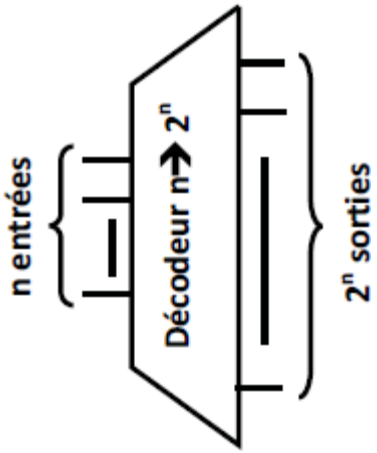
It activates the output line corresponding to the binary code present at the input.



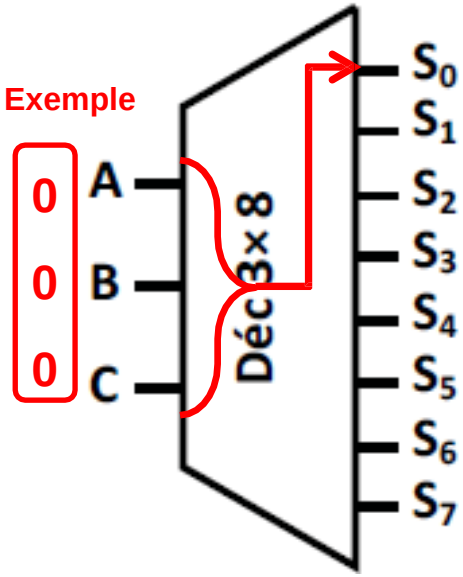


7. The binary decoder

Circuit synthesis: 3x8 decoder



Les entrées/sorties.

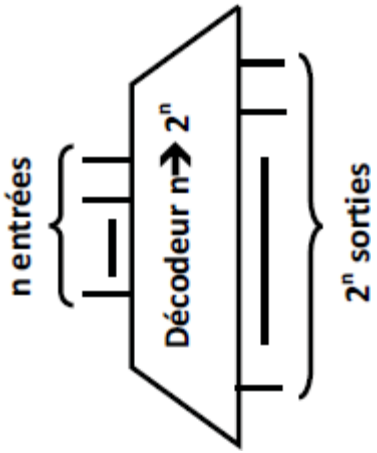


La table de vérité.

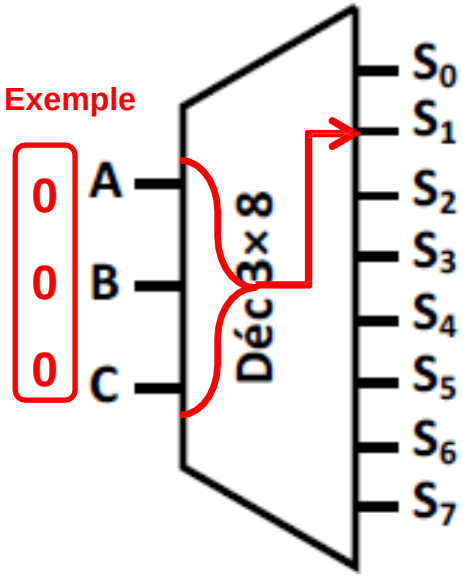
A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

7. The binary decoder

Circuit synthesis: 3x8 decoder



Les entrées/sorties.



La table de vérité.

A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

7. The binary decoder

Circuit synthesis: 3x8 decoder

La table de vérité.

A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Les expréssions logiques.

$$S_0 = \overline{A}.\overline{B}.\overline{C}$$

$$S_1 = \overline{A}.\overline{B}.C$$

$$S_2 = \overline{A}.B.\overline{C}$$

$$S_3 = \overline{A}.B.C$$

$$S_4 = A.\overline{B}.\overline{C}$$

$$S_5 = A.\overline{B}.C$$

$$S_6 = A.B.\overline{C}$$

$$S_7 = A.B.C$$

7. The binary decoder

Les expréssions logiques.

$S_0 = \overline{A}.\overline{B}.\overline{C}$

$S_1 = \overline{A}.\overline{B}.C$

$S_2 = \overline{A}.B.\overline{C}$

$S_3 = \overline{A}.B.C$

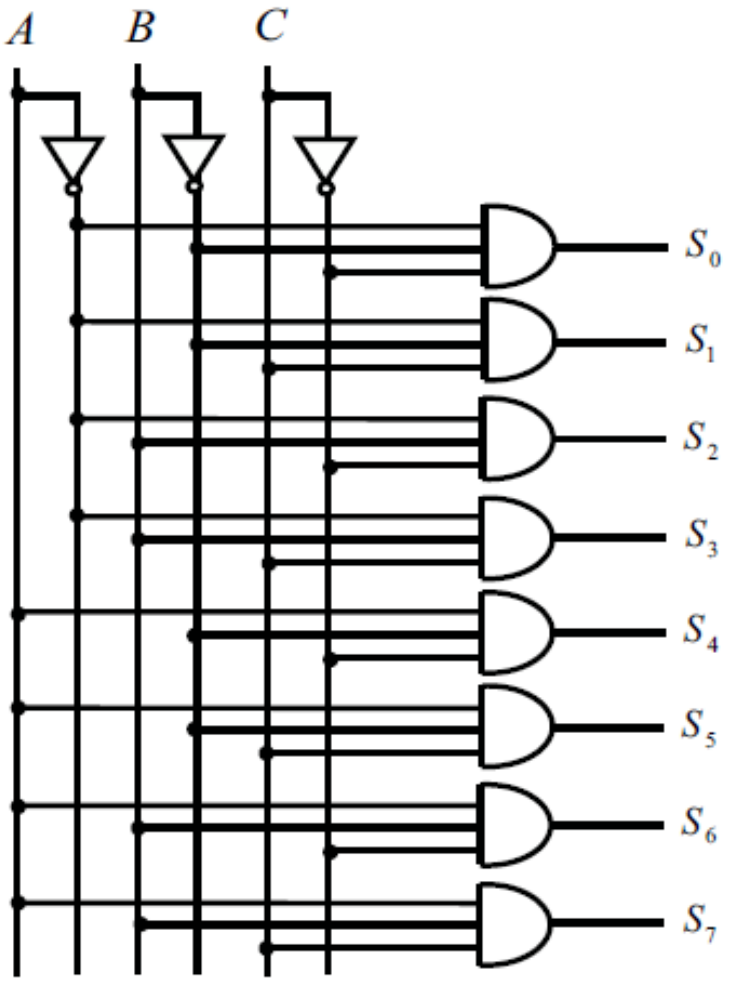
$S_4 = A.\overline{B}.\overline{C}$

$S_5 = A.\overline{B}.C$

$S_6 = A.B.\overline{C}$

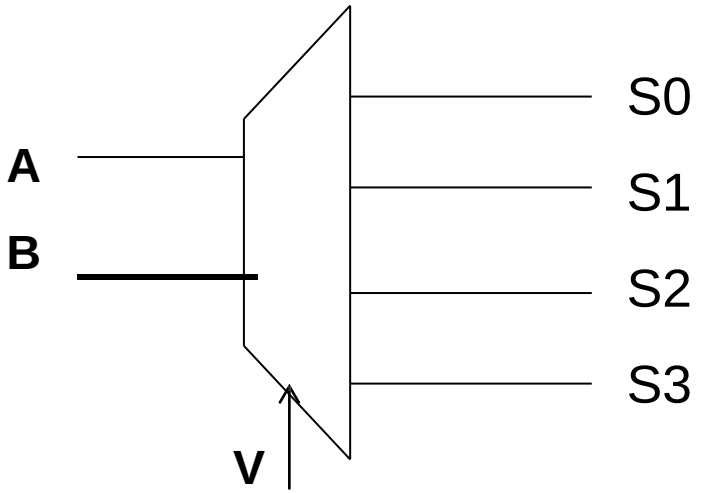
$S_7 = A.B.C$

Le schéma du circuit.



Decoder 2→4

V	A	B		S0	S1	S2	S3
0	X	X		0	0	0	0
1	0	0		1	0	0	0
1	0	1		0	1	0	0
1	1	0		0	0	1	0
1	1	1		0	0	0	1



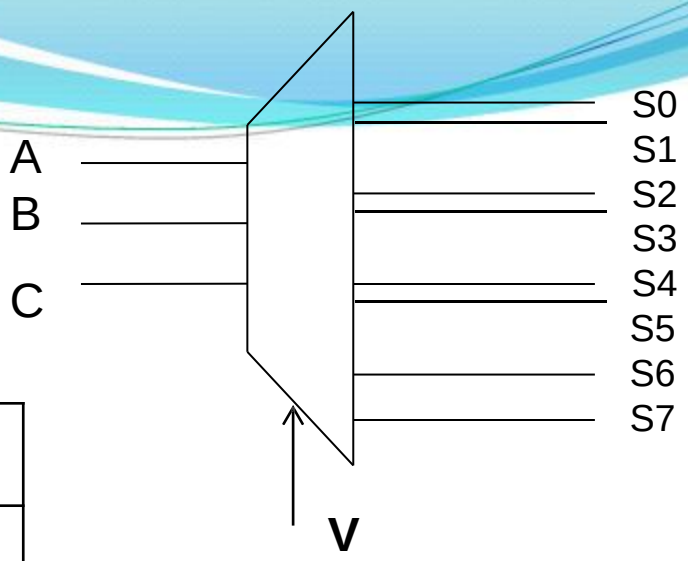
$$S_0 = (\overline{A}.\overline{B}).V$$

$$S_1 = (\overline{A}.B).V$$

$$S_2 = (A.\overline{B}).V$$

$$S_3 = (A.B).V$$

Decoder 3→8



A	B	C		S0	S1	S2	S3	S4	S5	S6	S7
0	0	0		1	0	0	0	0	0	0	0
0	0	1		0	1	0	0	0	0	0	0
0	1	0		0	0	1	0	0	0	0	0
0	1	1		0	0	0	1	0	0	0	0
1	0	0		0	0	0	0	1	0	0	0
1	0	1		0	0	0	0	0	1	0	0
1	1	0		0	0	0	0	0	0	1	0
1	1	1		0	0	0	0	0	0	0	1

$$S_0 = \overline{A}.\overline{B}.\overline{C}$$

$$S_1 = \overline{A}.\overline{B}.C$$

$$S_2 = \overline{A}.B.\overline{C}$$

$$S_3 = \overline{A}.B.C$$

$$S_4 = A.\overline{B}.\overline{C}$$

$$S_5 = A.\overline{B}.C$$

$$S_6 = A.B.\overline{C}$$

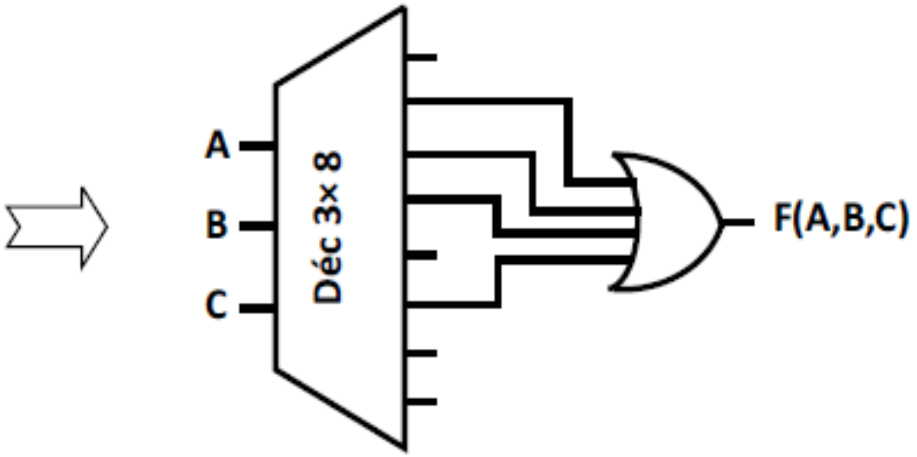
$$S_7 = A.B.C$$

Logic functions using Decoders

We can also generate any logic functions using decoders and basic logic gates. Simply connect the variables of the function to be generated to the decoder inputs, and the outputs corresponding to the various Minterms of the function to the inputs of one or more basic logic gates.

Example:Perform the following function $F(A,B,C)=\bar{B}C+\bar{A}B$ using a 3x8 decoder.

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic functions using Decoders

Example : Perform the following function using a 3x8 decoder. $F(A,B,C)=\bar{B}C+\bar{A}B$

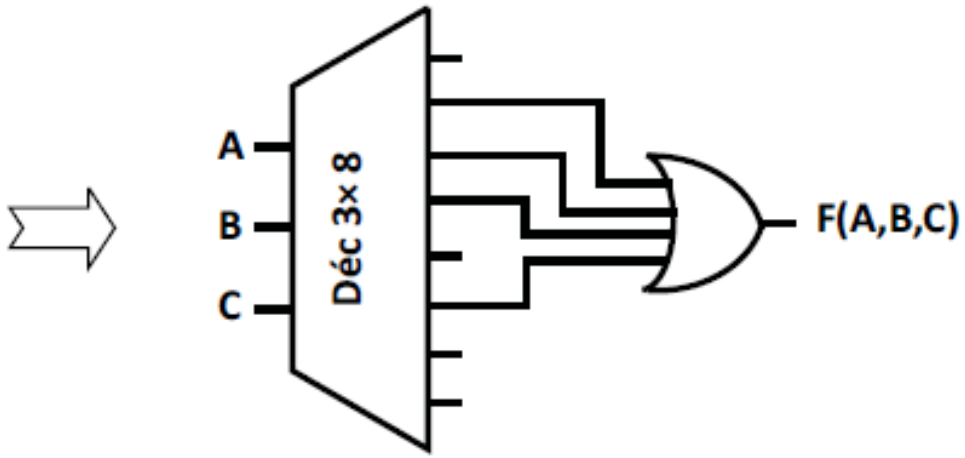
$$F(A,B,C)=\bar{B}C+\bar{A}B$$

$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$

$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$

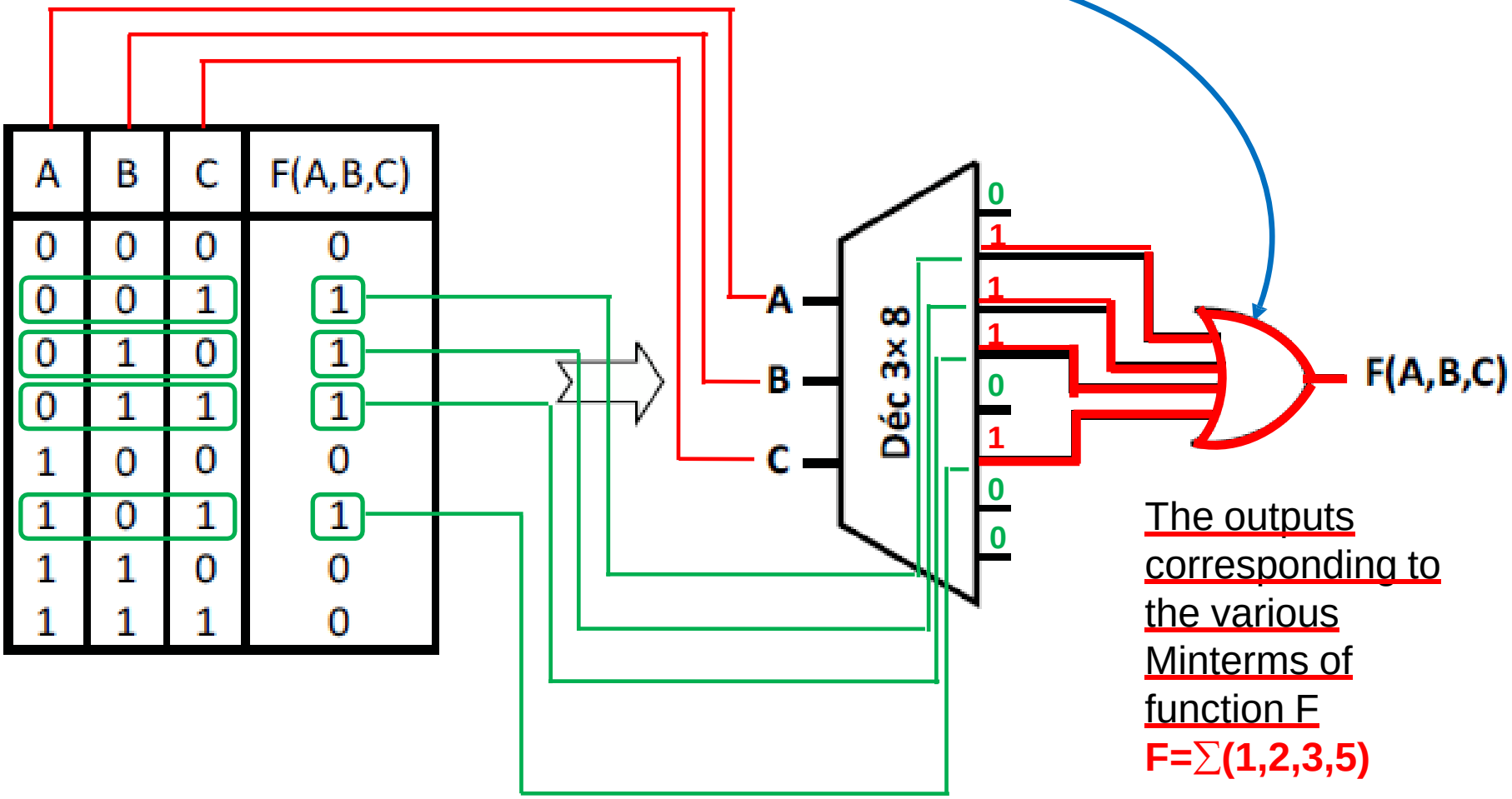
$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic functions using Decoders

$$F(A,B,C)=\bar{A}\,\bar{B}\,C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

The outputs corresponding to the various Minterms of function F

F=∑(1,2,3,5)

Building a full adder with binary decoders 3→8

Reminder :
Truth table of a 1-bit full adder

A_i	B_i	R_{i-1}		R_i	S_i
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

0+0 = 0 retenue 0
0+1 = 1 + 0 = 1 retenue 0
1 + 1 = 0 retenue 1
1+1+1 = 1 retenue 1

Building a complete adder with binary decoders 3→8

$$S_i = \overline{A_i}\,\overline{B_i}\,R_{i-1} + \overline{A_i}\,B_i\,\overline{R_{i-1}} + A_i\,\overline{B_i}\,\overline{R_{i-1}} + A_i\,B_i\,R_{i-1}$$

0 0 1 0 1 0 1 0 0 1 1 1

$$R_i = \overline{A_i}\,B_i\,R_{i-1} + A_i\,\overline{B_i}\,R_{i-1} + A_i\,B_i\,\overline{R_{i-1}} + A_i\,B_i\,R_{i-1}$$

0 1 1 1 0 1 1 1 0 1 1 1

Building a complete adder with binary decoders 3→8

Let put$A=A_i$, $B=B_i$, $C=R_{i-1}$

Hence:

$$S_0 = \overline{A}.\overline{B}.\overline{C}, \quad S_1 = \overline{A}.\overline{B}.C, \quad S_2 = \overline{A}.B.\overline{C}, \quad S_3 = \overline{A}.B.C,$$

$$S_4 = A.\overline{B}.\overline{C}, \quad S_5 = A.\overline{B}.C, \quad S_6 = A.B.\overline{C}, \quad S_7 = A.B.C$$

$$R_i = S3 + S5 + S6 + S7$$

$$S_i = S1 + S2 + S4 + S7$$

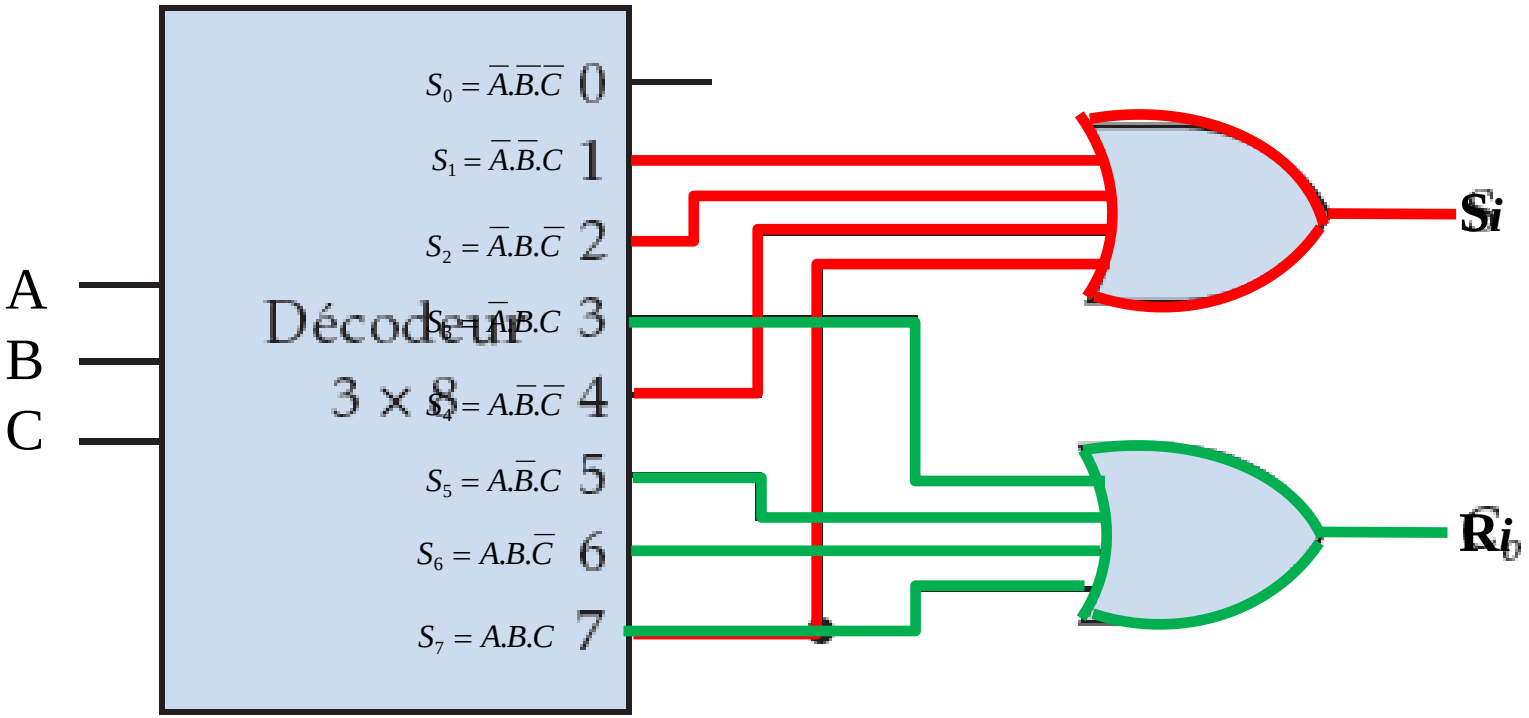
Building a complete adder with binary decoders 3→8

Let put $A=A_i$, $B=B_i$, $C=R_{i-1}$

Hence:

$$S_i = S1 + S2 + S4 + S7$$

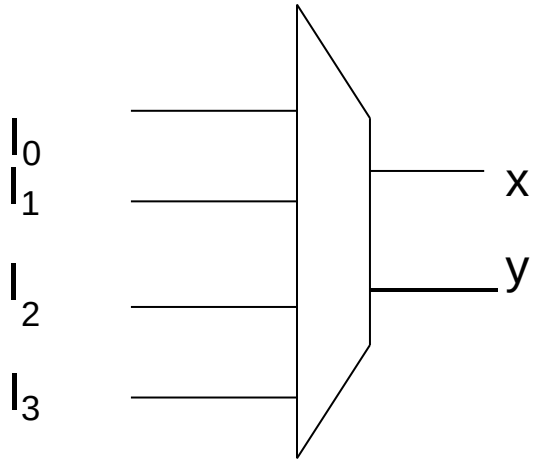
$$R_i = S3 + S5 + S6 + S7$$



8. The binary encoder

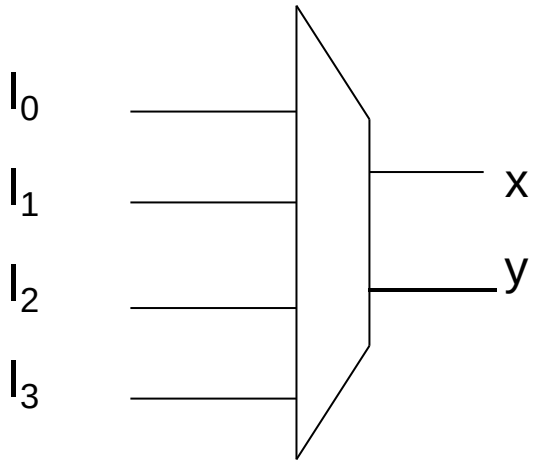
- It plays the inverse role of a decoder:
  - It has 2^n inputs
  - n outputs
  - For each input combination, we'll get its number (in binary) at the output.

Encoder 4→2



The binary encoder(4→2)

I_0	I_1	I_2	I_3		x	y
0	0	0	0		0	0
1	x	x	x		0	0
0	1	x	x		0	1
0	0	1	x		1	0
0	0	0	1		1	1

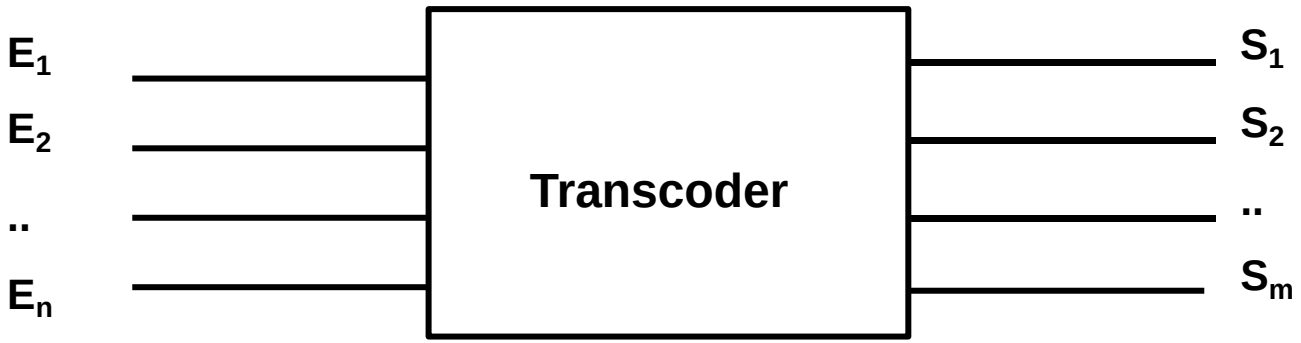


$$X = \overline{I0}.\overline{I1}.(I2 + I3)$$

$$Y = \overline{I0}.(I1 + .\overline{I2}.I3)$$

9. The transcoder

- It's a combinational circuit that transforms an input X code (n bits) into an output Y code (m bits).



Appendix

How to conceive a full adder using half adders?

1. A full adder by using Half Adders

$$R_i = A_i.B_i + R_{i-1}.(B_i \oplus A_i)$$

$$S_i = A_i \oplus B_i \oplus R_{i-1}$$

We put: $X = A_i \oplus B_i$ et $Y = A_i B_i$

We obtain:

$$R_i = Y + R_{i-1}.X$$

$$S_i = X \oplus R_{i-1}$$

If we put $Z = X \oplus R_{i-1}$ et $T = R_{i-1}.X$

We obtain:

$$R_i = Y + T$$

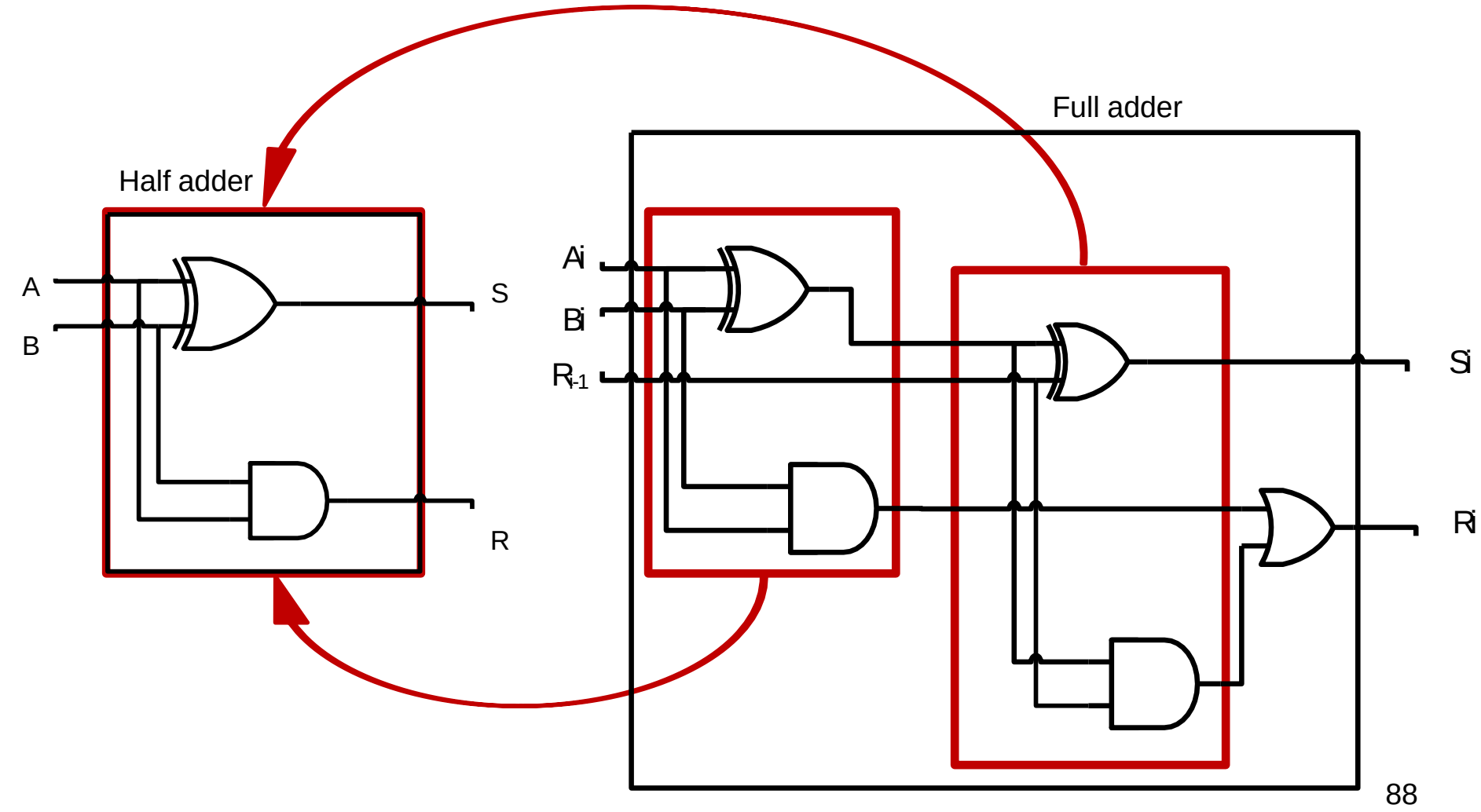
$$S_i = Z$$

$$R = A.B$$

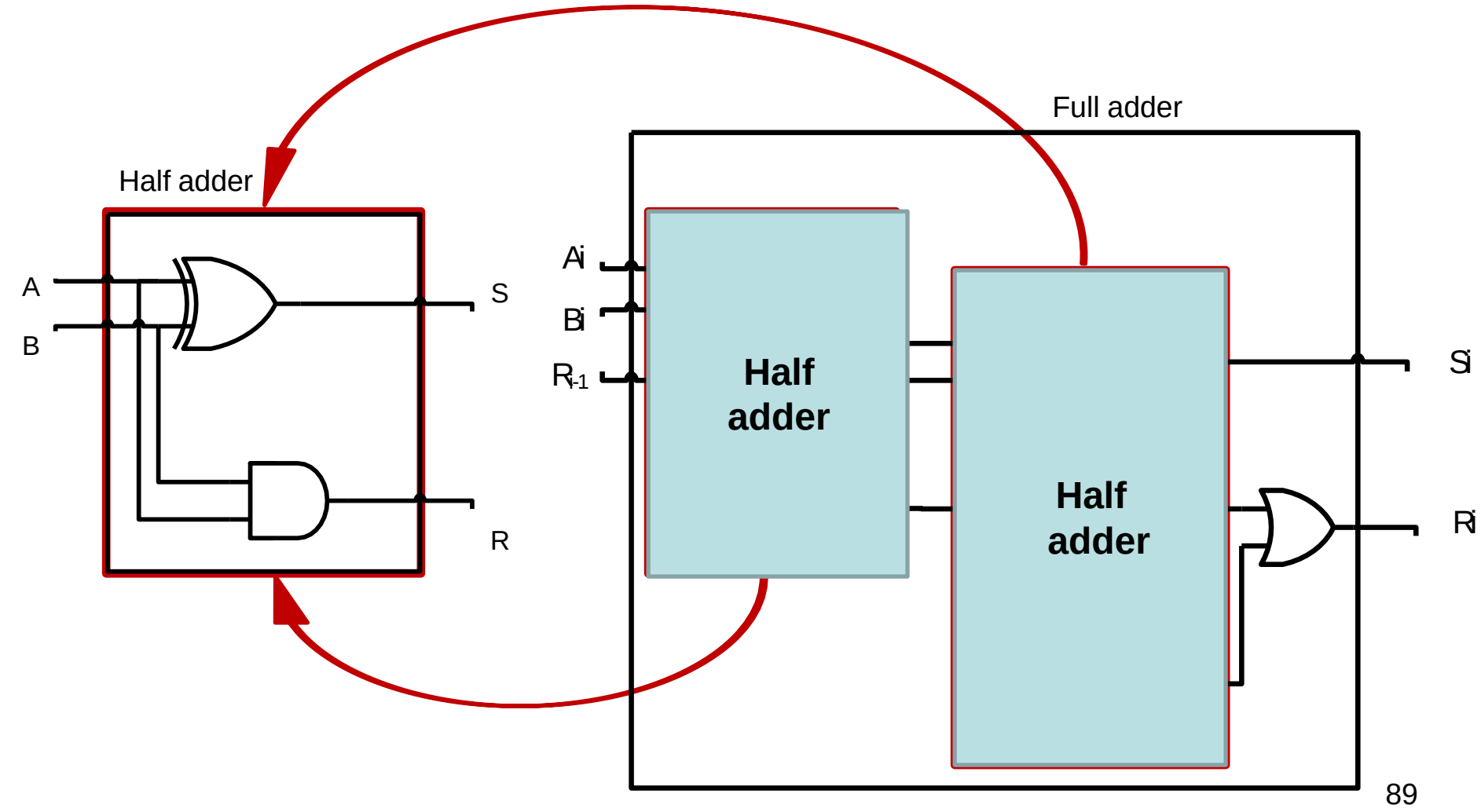
$$S = A \oplus B$$

- **We observe that X and Y are the outputs of a half-adder with A and B as inputs.**
- **We observe that Z and T are the outputs of a half-adder with inputs X and R_{i-1}**

2. Comparison of a half adder with a full adder



Comparison of a half adder with a full adder



$$X = A_i \oplus B_i$$

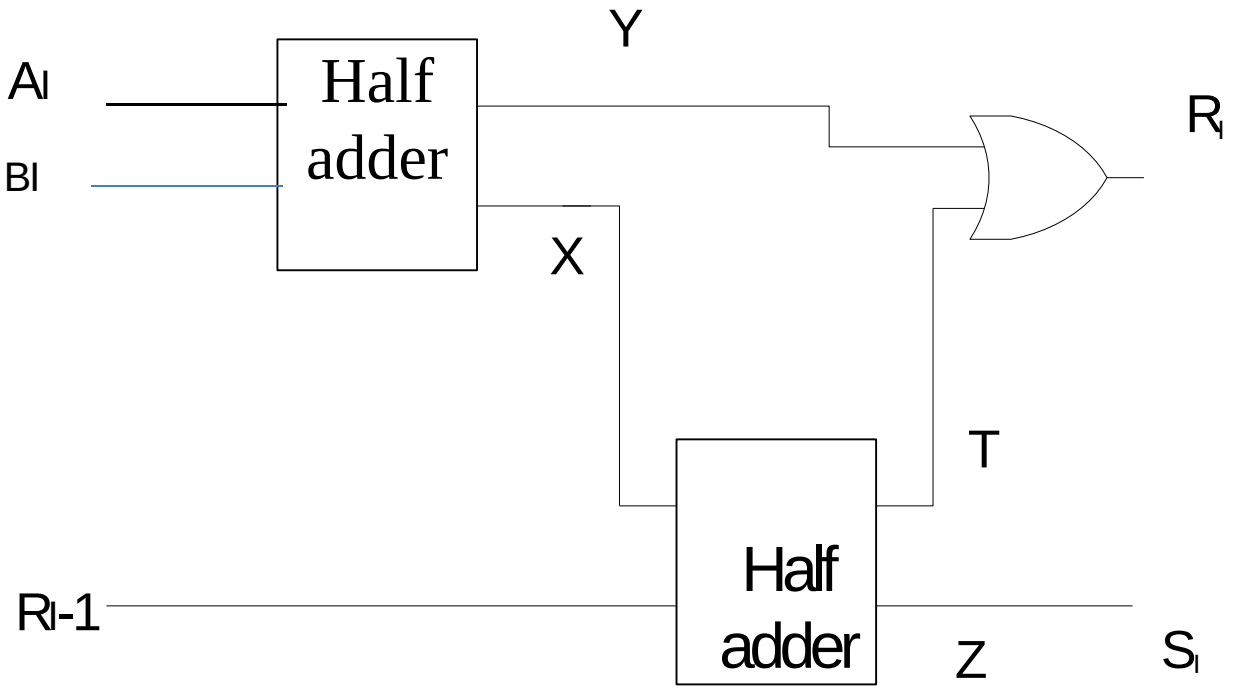
$$Y = A_i B_i$$

$$Z = X \oplus R_{i-1}$$

$$T = R_{i-1} . X$$

$$R_i = Y + T$$

$$S_i = Z$$



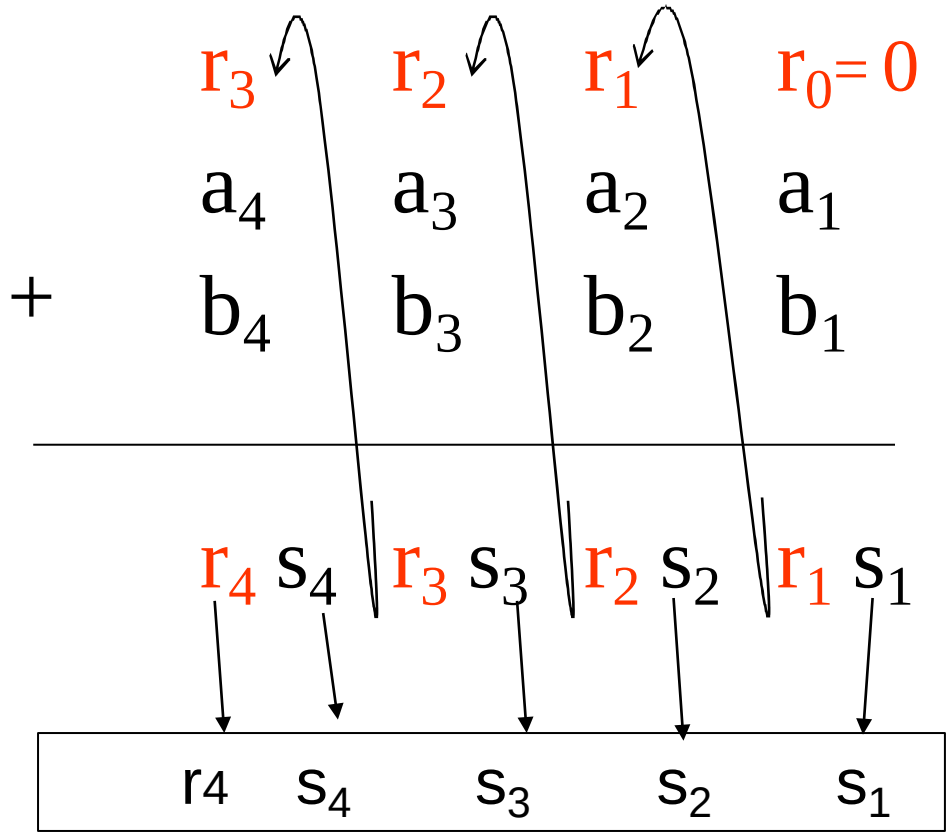
3. 4-bit adder

- A 4-bit adder is a circuit that adds two numbers A and B, each with 4 bits.
  - $A(a_3a_2a_1a_0)$
  - $B(b_3b_2b_1b_0)$

  In addition, it takes into account incoming deduction/carry
- At the output, we'll have the result on 4 bits, plus the carry (5 bits at the output).
- So, in total, the circuit has 9 inputs and 5 outputs. With 9 inputs: we have 2^9=512 combinations.....that's a lot! How can we represent the truth table?
- So we need to find an easier, more efficient way of designing this circuit?

4-bit adder (continued)

When we add in binary, we add bit by bit, starting from the reliable weight, and each time we propagate the outgoing carry to the bit of the higher rank. Single-bit addition can be performed by a full 1-bit adder.



Final result

4-bit adder (schema)