Machine Structure 02

Chapiter 01

Combinatorial Circuits

Introduction and Recap

- To understand the operation of the main elements of a computer, such as the Arithmetic and Logic Unit (ALU).

You should have achieved the following objectives:

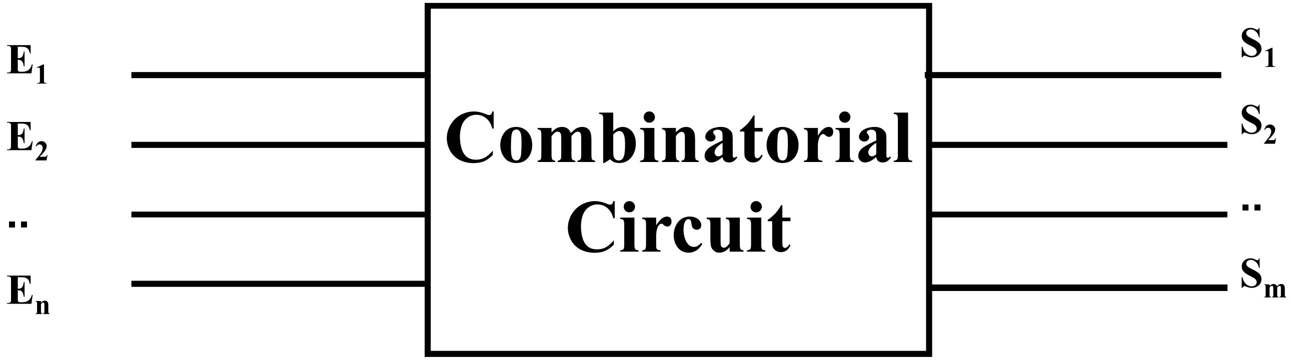
1. Describe the operation and properties of logic gates, simple combinational circuits such as adders, decoders, multiplexers, and demultiplexers...;
2. Use the theorems and identities of Boolean algebra to synthesize a circuit from its truth table and simplify the obtained result.

Objectives

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

1. Combinatorial Circuits

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



Block Diagram

- It is possible to use combinational circuits to **implement 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,,,,,

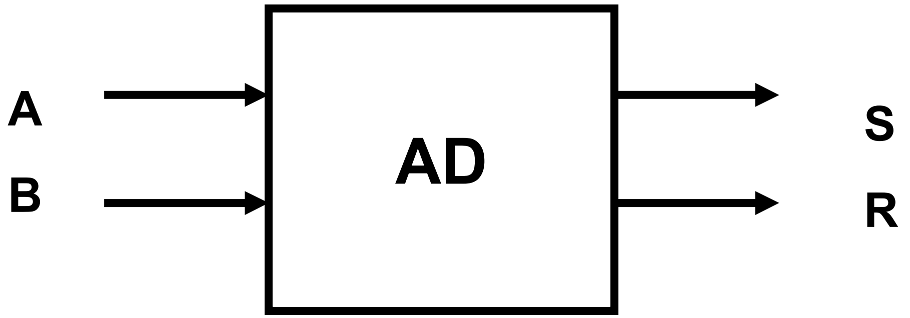
Examples of Combinational Circuits

- In a computer, we can distinguish three different classes of combinational logic circuits.

1. Combinational circuits for arithmetic and logic operations, such as adders, subtractors, comparators, etc.
2. Combinational circuits for data routing and transmission, such as encoders, decoders, multiplexers, demultiplexers, etc.
3. Combinational circuits for coding and code conversion, such as transcoders, 7-segment displays, etc.

Half Adder

- The half adder is a combinational circuit that allows for the arithmetic sum of two numbers A and B, each on one bit. At the output, we will have the Sum **S** and the Carry **R**

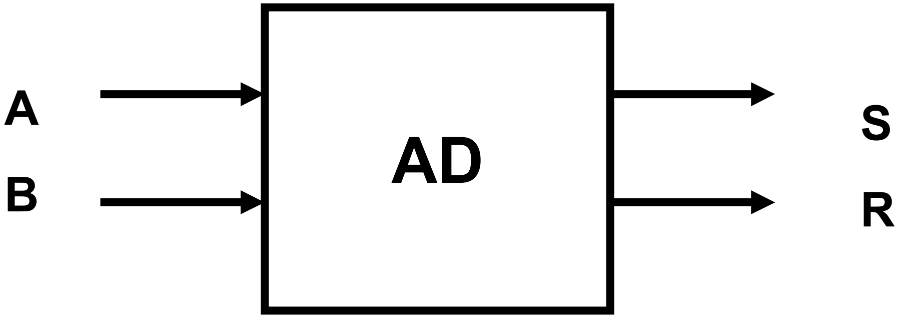


To find the structure (the diagram) of this circuit, we first need to create its truth table.

- **En binaire l’addition sur un seul bit se fait de la manière suivante:**

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

- **dresser sa table de vérité :**



- In binary, addition on a single bit is done as follows:

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

•The associated truth table:

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

From the truth table, we can find...

$$R=$$

$$S=$$

- In binary, addition on a single bit is done as follows:

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

- The associated truth table:

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 can find...

$$R =$$

$$S =$$

- In binary, addition on a single bit is done as follows:

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

- The associated truth table:

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

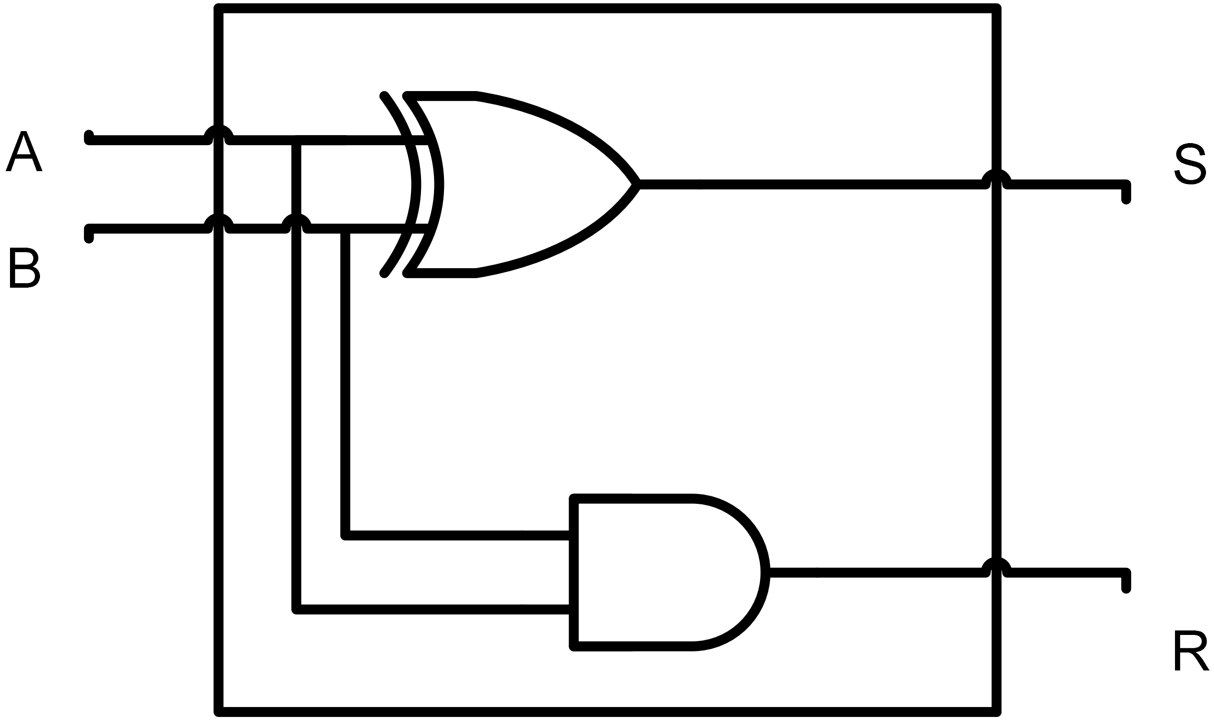
From the truth table, we can find...

$$R = A.B$$

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

$$R = A.B$$

$$S = A \oplus B$$



Full Adder

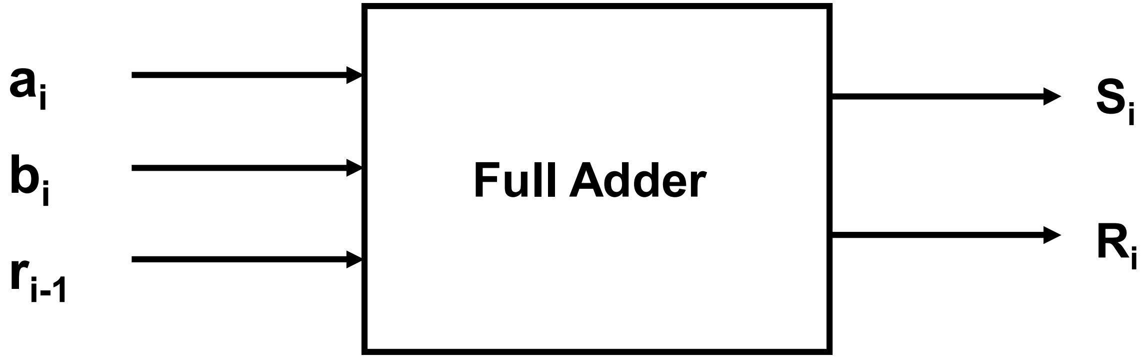
- In binary, when performing addition, the incoming carry must be taken into account.

$$\begin{array}{ccccc} r_4 & r_3 & r_2 & r_1 & r_0 = 0 \\ & a_4 & a_3 & a_2 & a_1 \\ + & b_4 & b_3 & b_2 & b_1 \\ \hline r_4 & s_4 & s_3 & s_2 & s_1 \end{array}$$

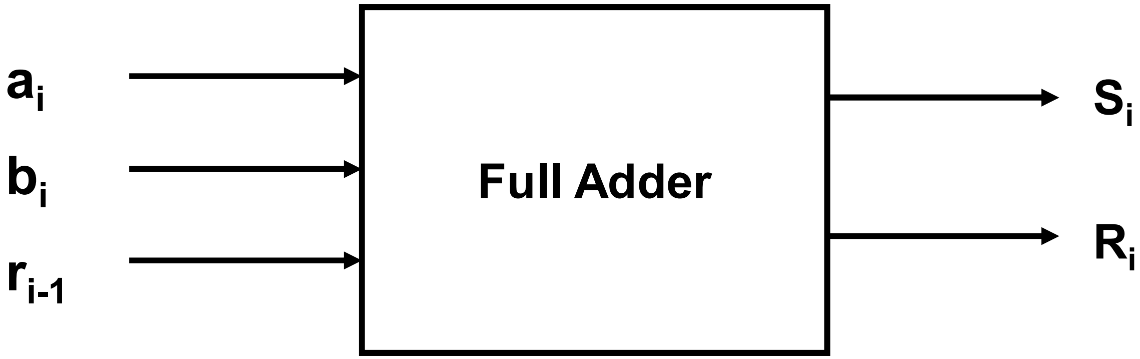
$$\begin{array}{ccc} & & r_{i-1} \\ & & a_i \\ + & & b_i \\ \hline & r_i & s_i \end{array}$$

Full Adder : 1-Bit

- The full adder for one bit has 3 inputs:
  - ➢ ai : the first number on one bit.
  - ➢ bi : the second number on one bit.
  - ➢ ri-1 : the incoming carry on one bit.
- It has two outputs:
  - Si: the sum
  - Ri: the outgoing carry



Truth table of a full adder for 1 bit



Create its truth table?

Truth table of a full adder for 1 bit

- It has **3 inputs**:

1. a_i: the first number.
2. b_i: the second number.
3. r_{i-1}: the incoming carry.

- It has **2 outputs**:

1. S_i: the sum
2. R_i: the outgoing carry

a_i	b_i	r_{i-1}		r_i	s_i

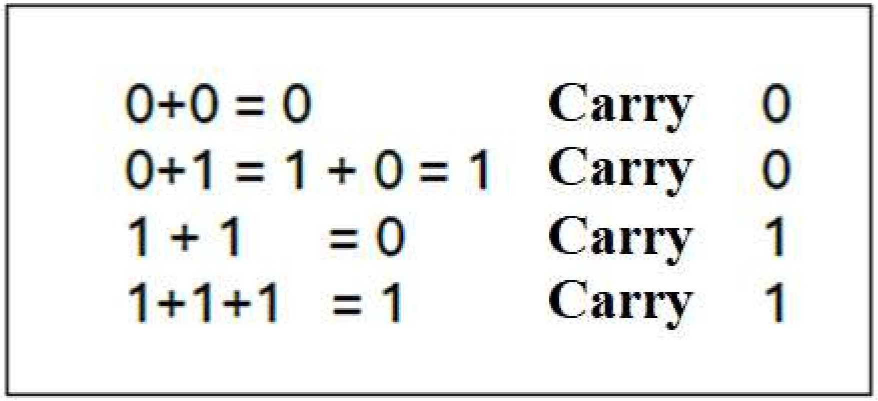
Truth table of a full adder for 1 bit

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 full adder for 1 bit



0+0 = 0 Carry 0

0+1 = 1 + 0 = 1 Carry 0

1 + 1 = 0 Carry 1

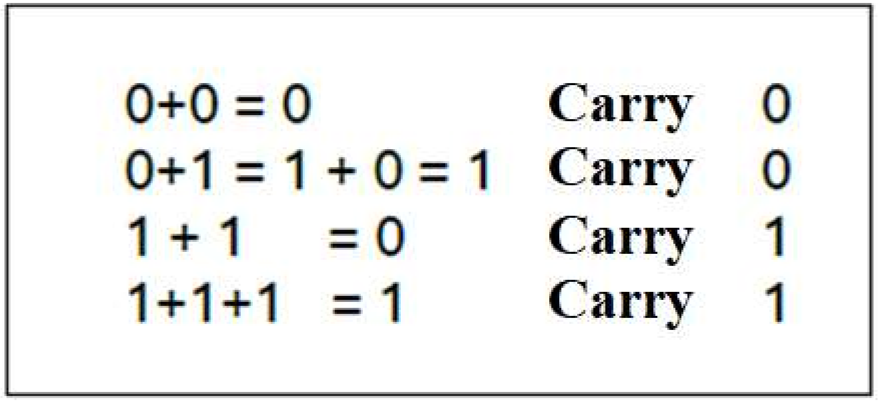
1+1+1 = 1 Carry 1

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 full adder for 1 bit



0+0 = 0 Carry 0
0+1 = 1 + 0 = 1 Carry 0
1 + 1 = 0 Carry 1
1+1+1 = 1 Carry 1

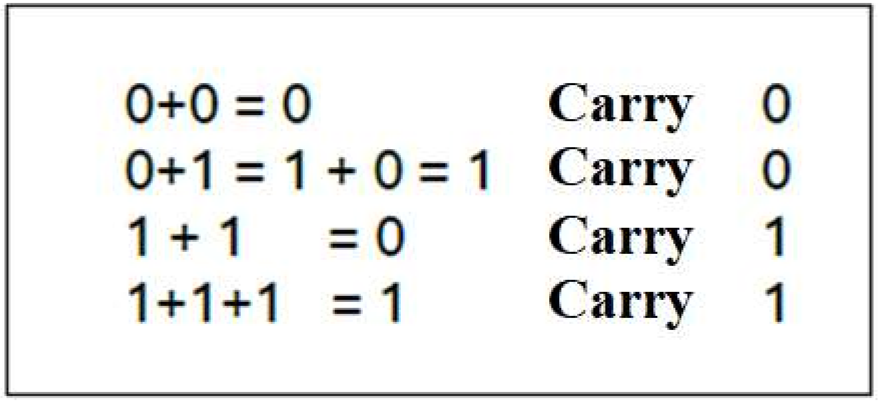
a_i	b_i	r_{i-1}		r_i	s_i
0	0	0		0	0
0	0	1		0	1
0	1	0		0	1
0	1	1		1	0
1	0	0		0	1
1	0	1		1	0
1	1	0		1	0
1	1	1		1	1

$$S_i =$$

$$R_i =$$

Truth table of a full adder for 1 bit

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	Carry	0
0+1 = 1 + 0 = 1	Carry	0
1 + 1 = 0	Carry	1
1+1+1 = 1	Carry	1

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

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

If we want to simplify the equations, we obtain:

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

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

If we want to simplify the equations, we obtain:

$$S_i = \boxed{\overline{A}_i}.\underline{\overline{B}_i.R_{i-1}} + \boxed{\overline{A}_i}.\underline{B_i.\overline{R}_{i-1}} + \boxed{A_i}.\underline{\overline{B}_i.\overline{R}_{i-1}} + \boxed{A_i}.\underline{B_i.R_{i-1}}$$

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

$$S_i = \underline{\overline{A}_i \boxed{X = B_i \oplus R_{i-1}} + A_i.\overline{\boxed{X = B_i \oplus R_{i-1}}}} \rightarrow \boxed{= A_i \oplus X}$$

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

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

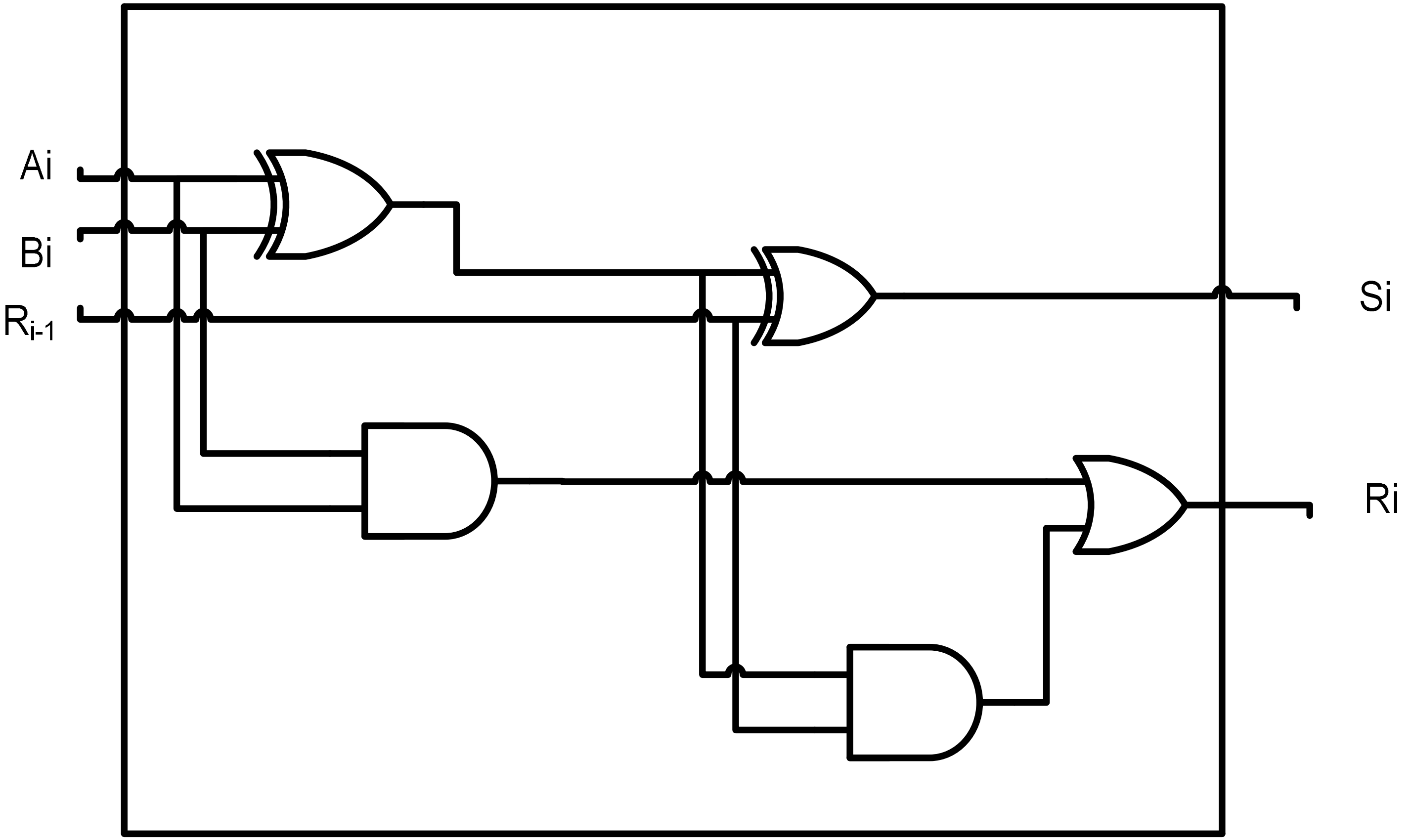
$$R_i = \boxed{R_{i-1}}.(\underline{\overline{A}_i.B_i + A_i.\overline{B}_i}) + \boxed{A_i B_i}(\underline{\overline{R}_{i-1} +_i R_{i-1}})$$

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

3.2 Diagram of a full adder

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

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



Using Half Adders

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

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

Si on pose $X = \boxed{A_i \oplus B_i}$ et $Y = \boxed{A_i B_i}$

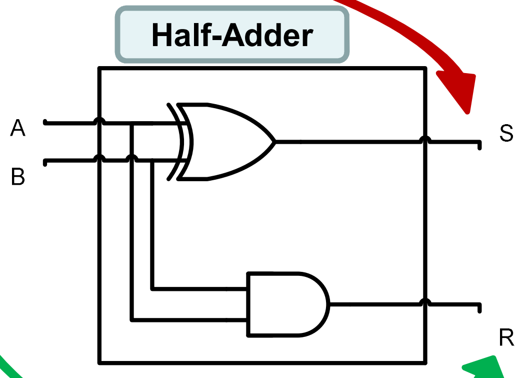
On obtient :

$$\boxed{\begin{array}{l} R_i = Y + R_{i-1}.X \\ S_i = X \oplus R_{i-1} \end{array}}$$

et si on pose $Z = \boxed{X \oplus R_{i-1}}$ et $T = \boxed{R_{i-1}.X}$

On obtient :

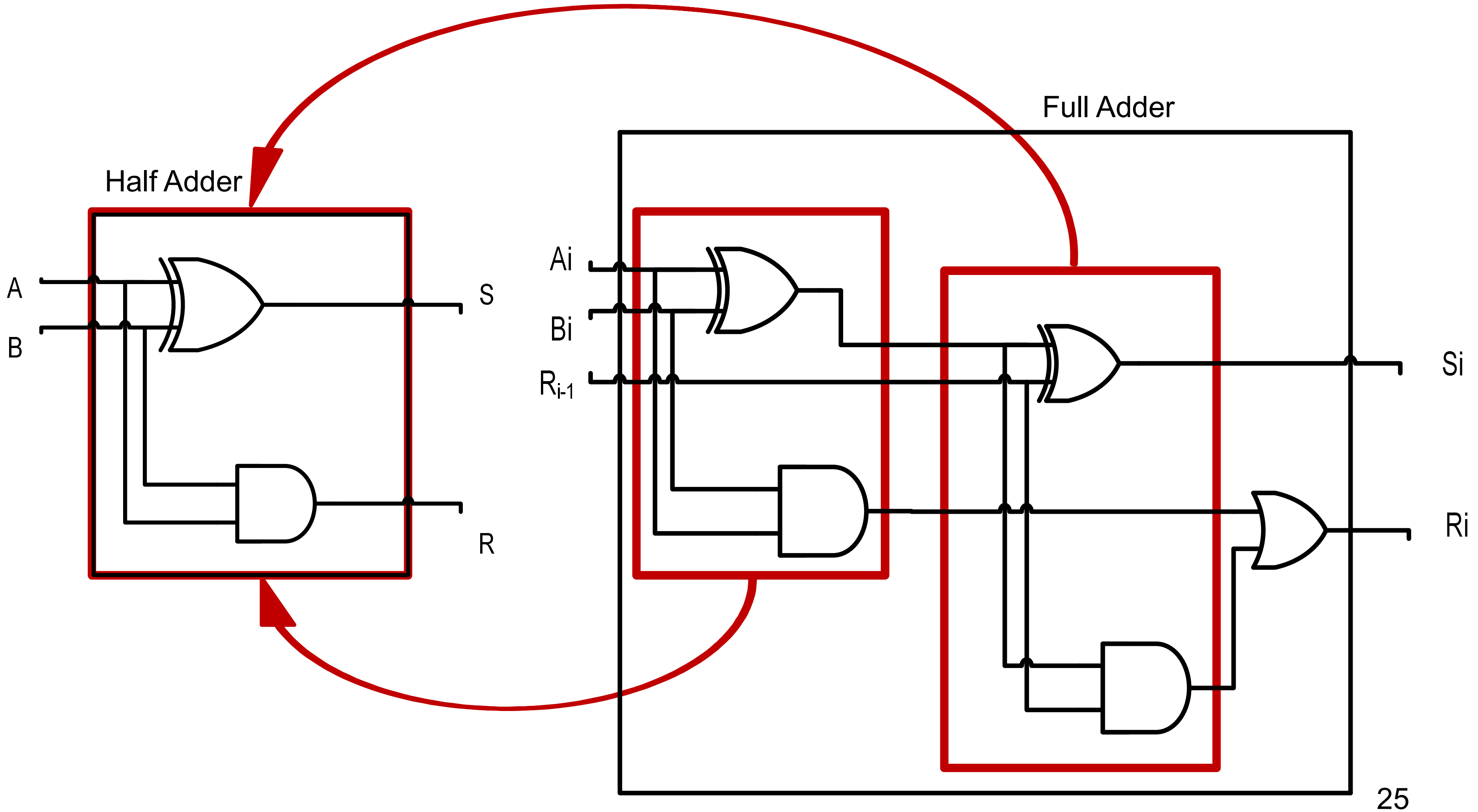
$$\boxed{\begin{array}{l} R_i = Y + T \\ S_i = Z \end{array}}$$



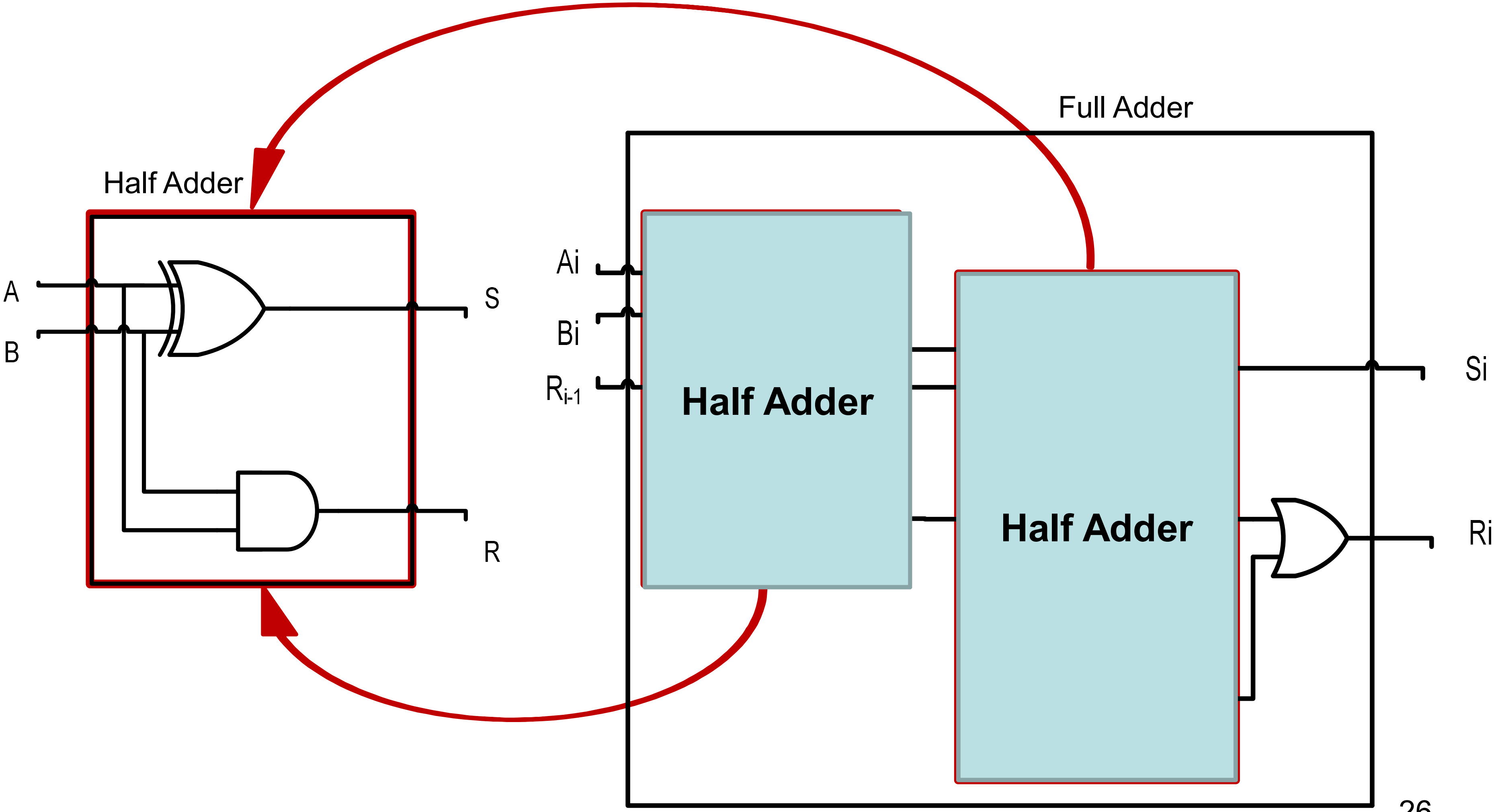
$$R = A.B$$
$$S = A \oplus B$$

- We note that X and Y are the outputs of a half adder with inputs A and B. We observe that Z and T are the outputs of a half adder with inputs X and Ri-1.

Comparison of a Half Adder with a Full Adder



3.2 Comparaison d'un demi additionneur avec additionneur complet



$$X = A_i \oplus B_i$$

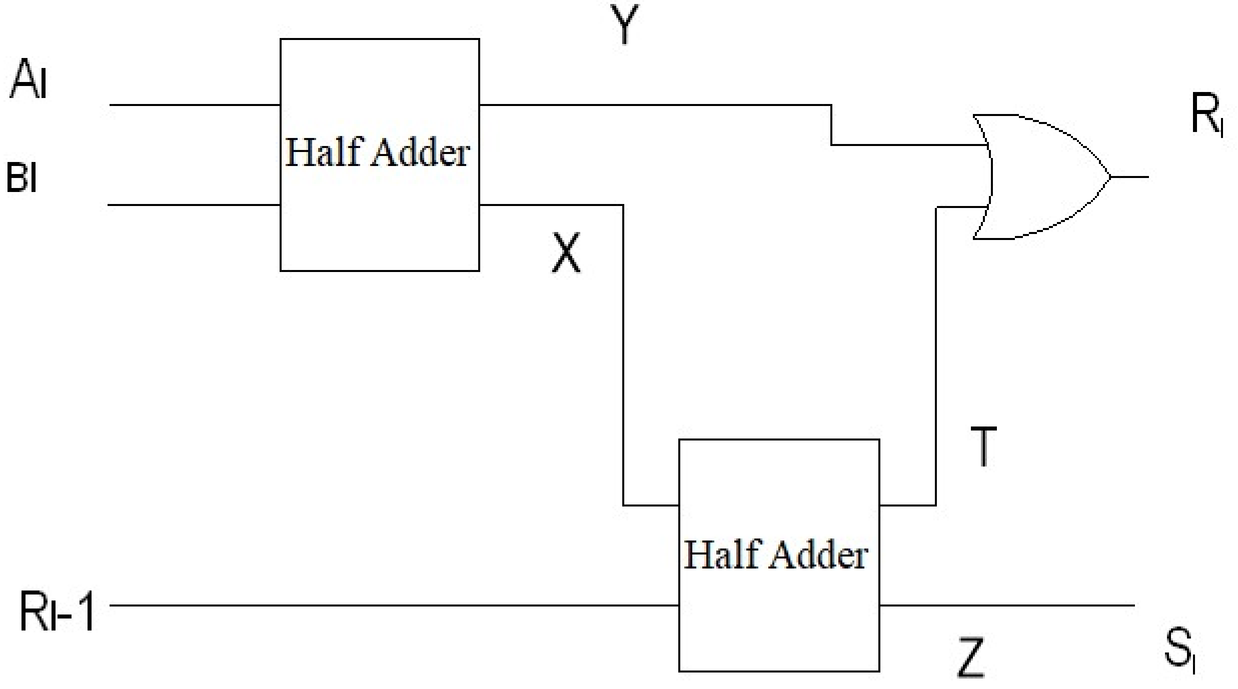
$$Y = A_i B_i$$

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

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

$$R_i = Y + T$$

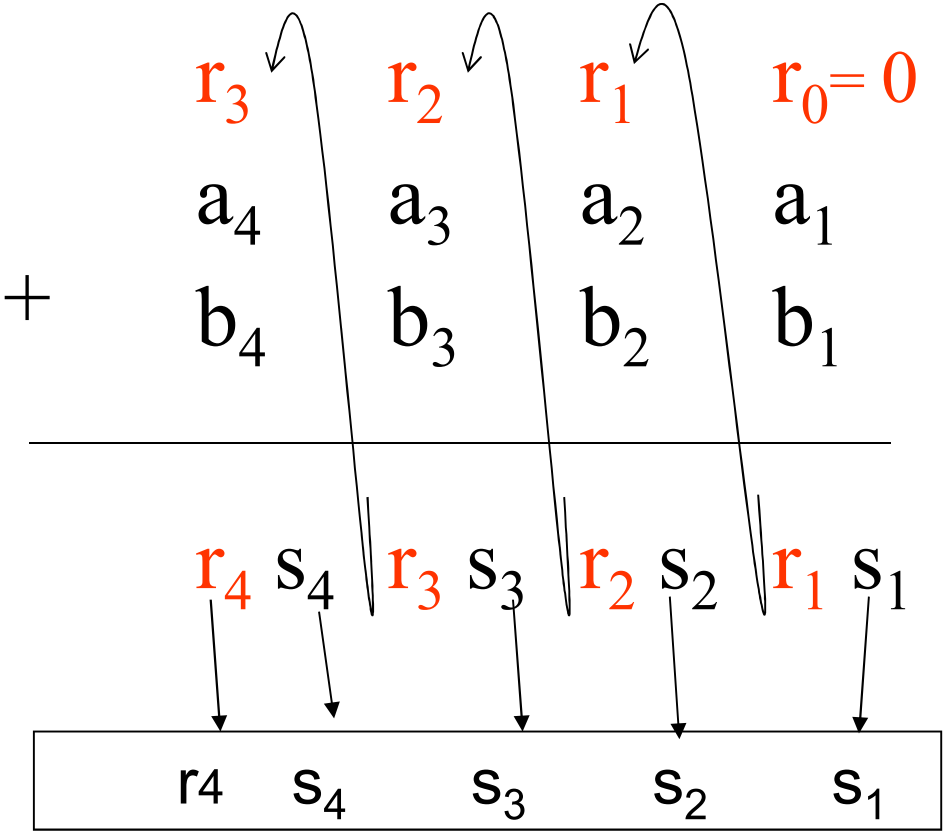
$$S_i = Z$$



(Four) 4-Bits Adder

- A 4-bit adder is a circuit that performs the addition of two 4-bit numbers, A and B, where:
  - A(a_3 a_2 a_1 a_0)
  - B(b_3 b_2 b_1 b_0)
- Additionally, it takes into account the incoming carry. The output consists of the 4-bit result and the carry (5 bits in total). Therefore, the circuit has 9 inputs and 5 outputs.
- With 9 inputs, we have a total of **2^9** = 512 combinations!!! How can we represent the truth table?
- We need to find a simpler and more efficient solution to design this circuit.

- When performing binary addition, we add bit by bit starting from the least significant bit, and each time we propagate the outgoing carry to the higher-order bit. Addition on a single bit can be accomplished using a full adder for 1 bit.



r_3 r_2 r_1 $r_0 = 0$

a_4 a_3 a_2 a_1

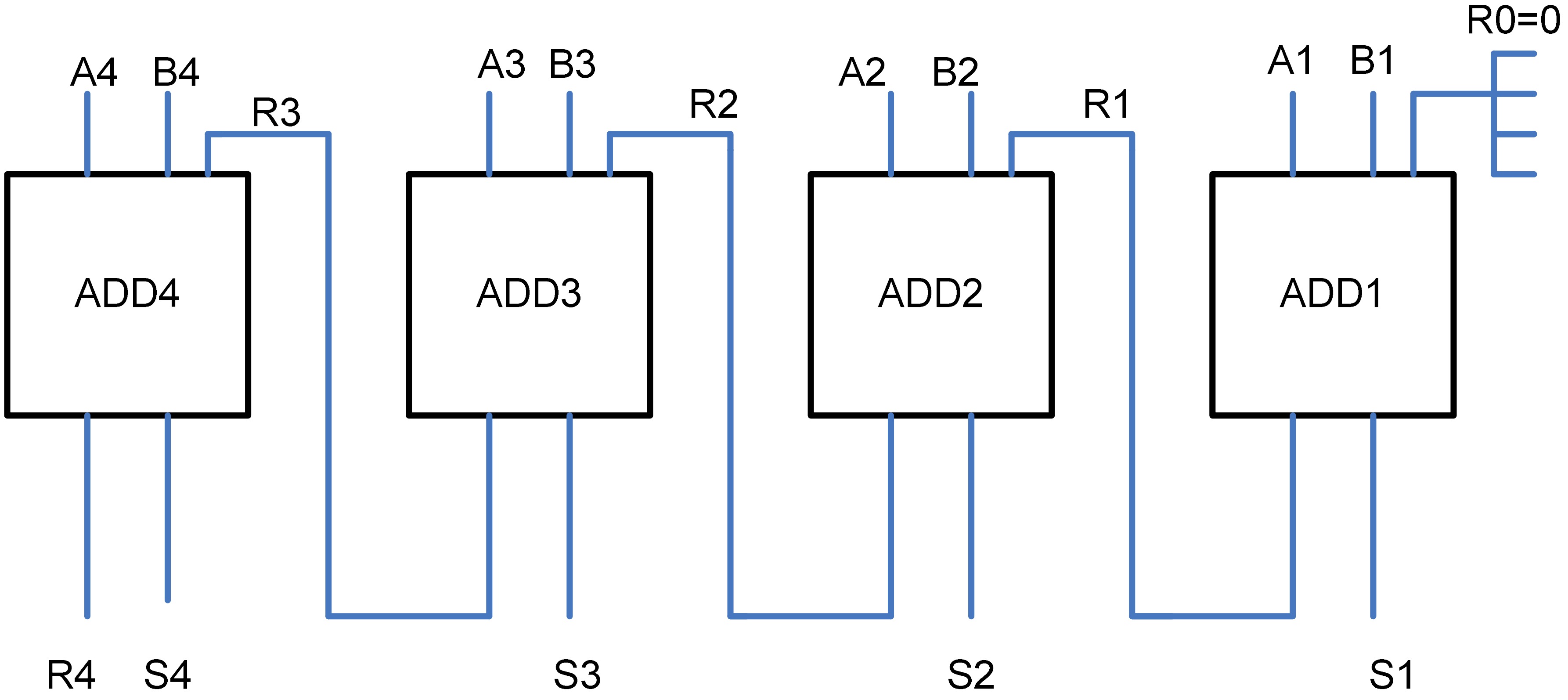
$+$ b_4 b_3 b_2 b_1

r_4 s_4 r_3 s_3 r_2 s_2 r_1 s_1

r4 s_4 s_3 s_2 s_1

Final Result

(Four) 4-Bit Adder (Diagram)



Exercice 1

- An odd (*number* not divided by two) parity generator is a function that returns 1 if the number of set bits is uncommon, and 0 otherwise.

- Define this function for a 4-bit word. Provide a logical circuit implementing this function.

Exercice 1

- What is the parity bit?
- Definition: The parity bit, or check bit, is a bit added to a binary code to check whether the code has an even or odd number of 1s. In odd parity, the code must have an odd number of 1s.
- For example, the code 10011 has odd parity because there are three 1s.
- On the other hand, the code 101101 is said to have even parity because there are four 1s.

Exercice 1

- **Qu'est-ce que le bit de parité?**
- **Définition:** Le bit de parité ou le bit de contrôle sont les bits ajoutés au code binaire pour vérifier si le code est en parité ou non, Dans le bit de parité impair, le code doit être dans un nombre impair de 1 . Exemple, Le code: 10011, à une parité impaire car il y a trois nombres de 1. Le code : 101101, est dit à parité paire car il y a quatre nombres de 1 .

Exercice 1

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

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

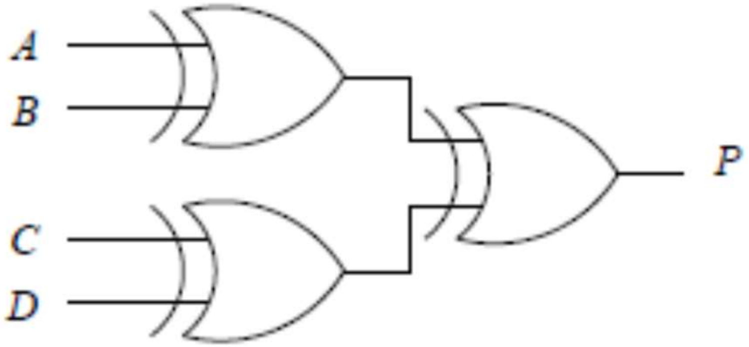
$$+\cdots+\overline{A}\cdot B\cdot C\cdot D + A\cdot\overline{B}\cdot C\cdot D + A\cdot B\cdot\overline{C}\cdot D + A\cdot B\cdot C\cdot\overline{D}$$

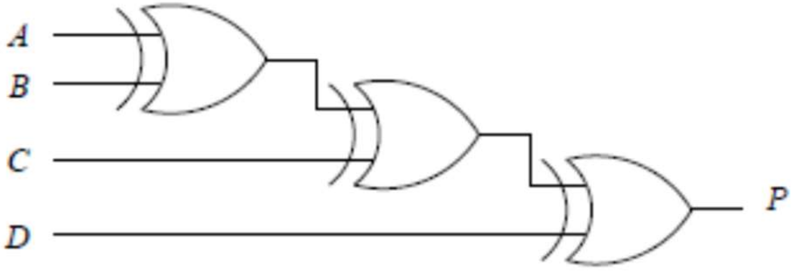
- which would result in a much too complicated circuit! We notice that for two bits, P = A⨁B:

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

Exercice 1

- The following circuits are deduced:



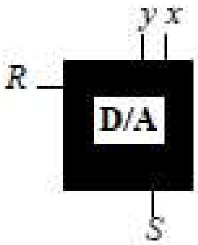


Exercice 2

- Recall the principles of a half-adder and then a full-adder. Deduce from these principles a logical circuit that implements the two's complement on n bits.

- Correction: The half-adder has two inputs (x and y) and two outputs (R and S). S corresponds to the zeroth bit of the result of the binary addition of x and y, and R to the first bit (carry).

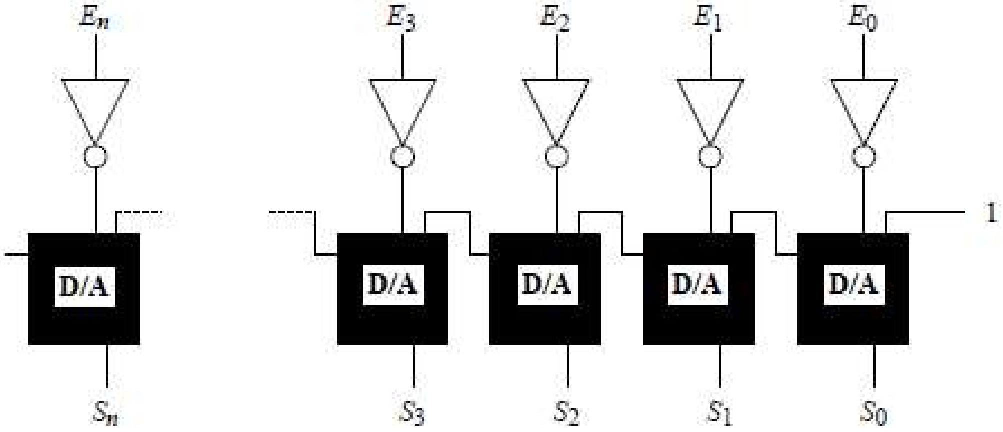
Exercice



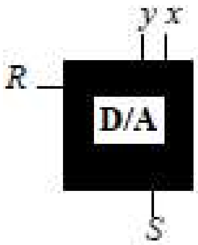
x	y	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = x \oplus y$$
$$R = x \cdot y$$

- Un additionneur complet s’obtient en enchaînant des demi-additionneurs de manière à propager correctement la retenue. On obtient selon le même principe le circuit effectuant un complément à deux :



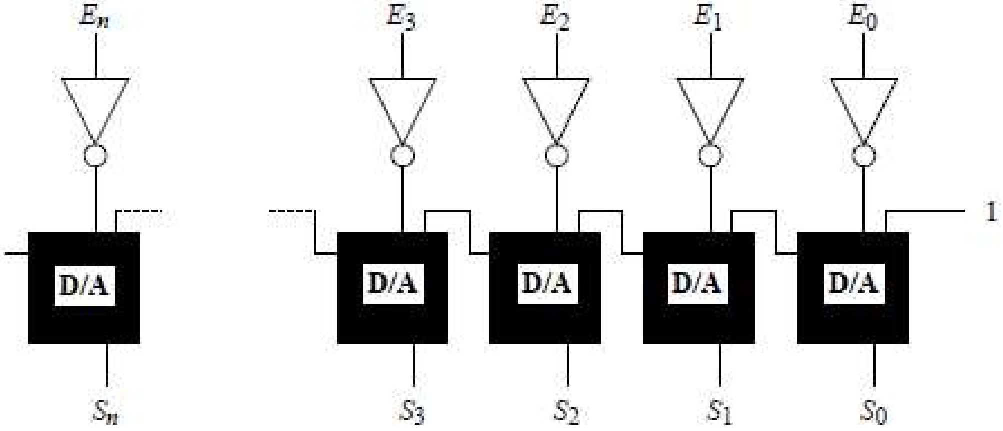
Exercice



x	y	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = x \oplus y$$
$$R = x \cdot y$$

- A full adder is obtained by chaining half adders together in such a way as to correctly propagate the carry. We obtain according to the same principle the circuit carrying out a two's complement:

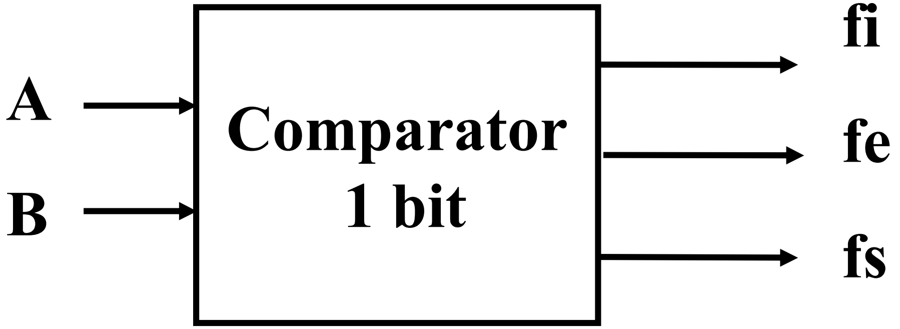


Comparator

- A comparator is an arithmetic circuit used to compare two binary numbers, A and B.
- The numbers A and B must have the same length (number of bits).
- We want to determine if $A > B$, $A < B$, or $A = B$. Therefore, the circuit provides a three-way answer.
- Our final circuit should produce three signals
  - **fs** (active if $A > B$),
  - **fi** (active if $A < B$), and
  - **fe** (active if $A = B$)
- by taking signals A and B as inputs.

Comparator

- It is a combinational circuit that allows comparison between two binary numbers, A and B.
- It has 2 inputs:
  - A: one bit
  - B: one bit
- It has 3 outputs:
  - fe: equality (A=B)
  - fi: less than (A < B)
  - fs: greater than (A > B)



Comparator

Truth table of a 1-bit comparator

Compile its truth table:

Implementation of a 1-Bit Comparator

A	B		fs	fe	fi
0	0				
0	1				
1	0				
1	1				

$fs =$

$fi =$

$fe =$

Implementation of a 1-Bit Comparator

A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

$fs =$

$fi =$

$fe =$

Implementation of a 1-Bit Comparator

A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

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

$$fi =$$

$$fe =$$

Implementation of a 1-Bit Comparator

A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

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

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

$$fe =$$

Implementation of a 1-Bit Comparator

A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

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

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

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

Implementation of a 1-Bit Comparator

A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

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

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

$$\boxed{fe} = \overline{A}\,\overline{B} + AB = \boxed{\overline{A \oplus B}} = \boxed{\overline{fs + fi}}$$

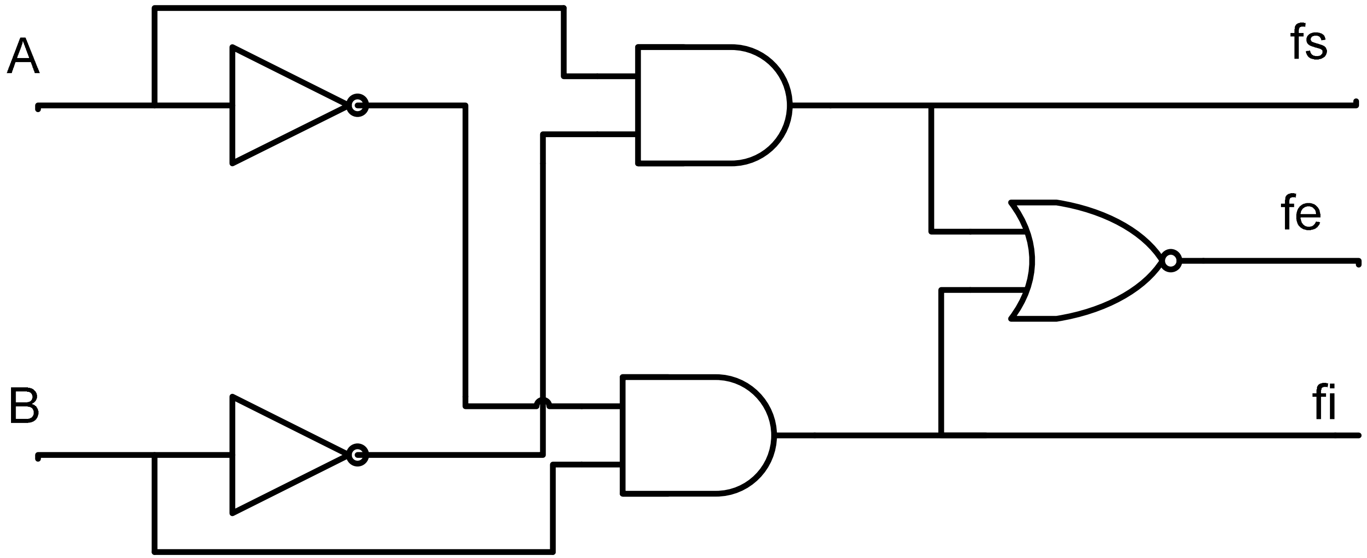
Because $fs + fi = A.\overline{B} + \overline{A}B = A \oplus B$

Implementation of a 1-Bit Comparator

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

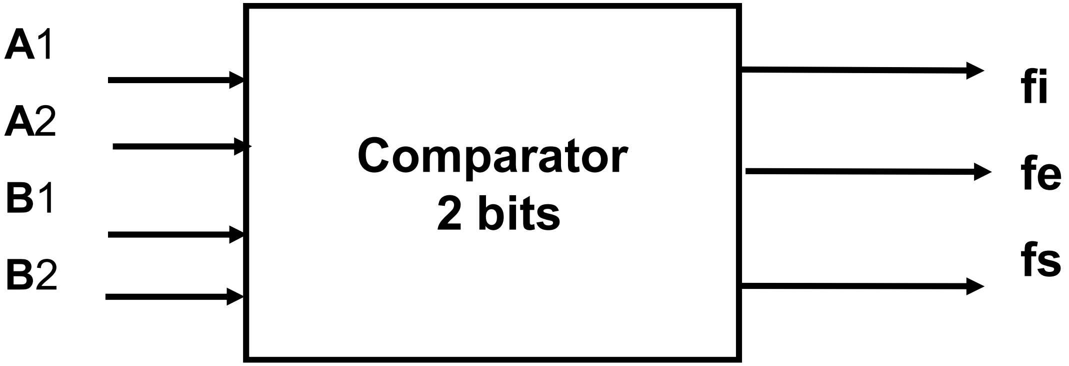
$$fi = \overline{A}B$$

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



Implementation of a 2-Bit Comparator

- It allows the comparison between two numbers A (a2a1) and B (b2b1), each with two bits.



A2	A1	B2	B1		fs	fe	fi
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				

Comparator one bit

1. A=B si

A2=B2 et A1=B1

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

2. A>B si

A2 > B2 ou (A2=B2 et A1>B1)

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

3. A<B si

A2 < B2 ou (A2=B2 et A1<B1)

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

A2	A1	B2	B1		fs	fe	fi
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

1 =fi

1 =fs

1 =fe

1. A=B si

A2=B2 et A1=B1

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

2. A>B si

A2 > B2 ou (A2=B2 et A1>B1)

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

3. A<B si

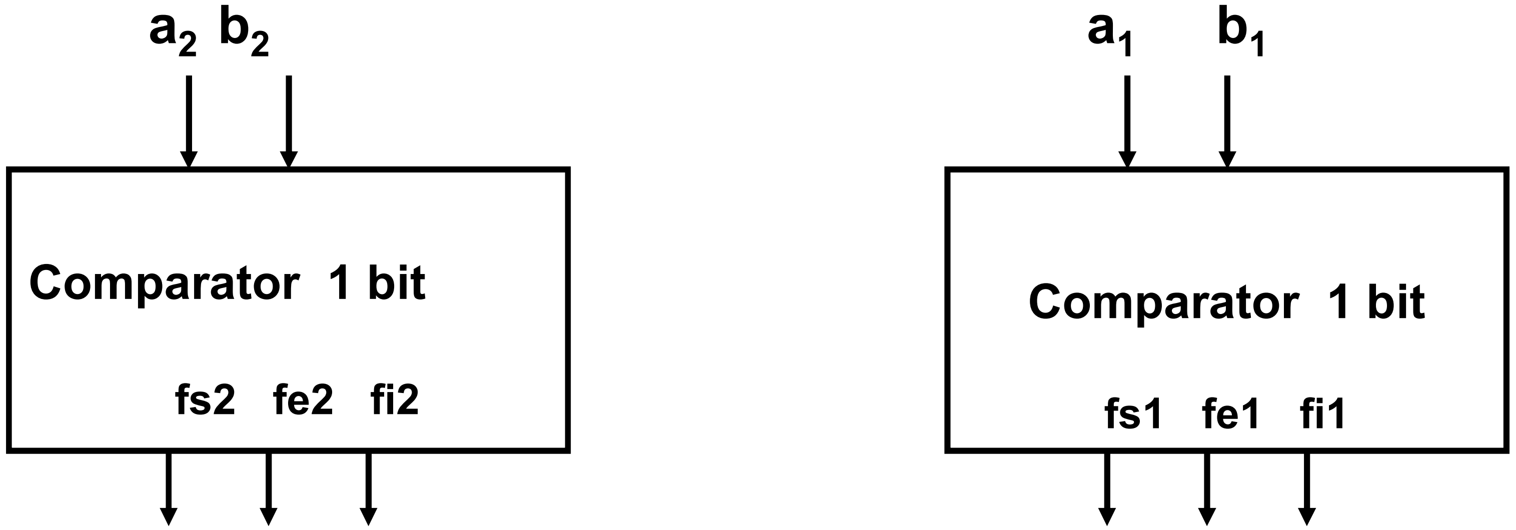
A2 < B2 ou (A2=B2 et A1<B1)

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

A2	A1	B2	B1		fs	fe	fi
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

Implementation of a 2-Bit Comparator using 1-bit Comparator

- It is possible to implement a 2-bit comparator using 1-bit comparators and logical gates. One comparator is used to compare the least significant bits, and another is used to compare the most significant bits. The outputs of these two comparators are combined to generate the final outputs of the overall comparator.



1. A=B si

A2=B2 et A1=B1

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

2. A>B si

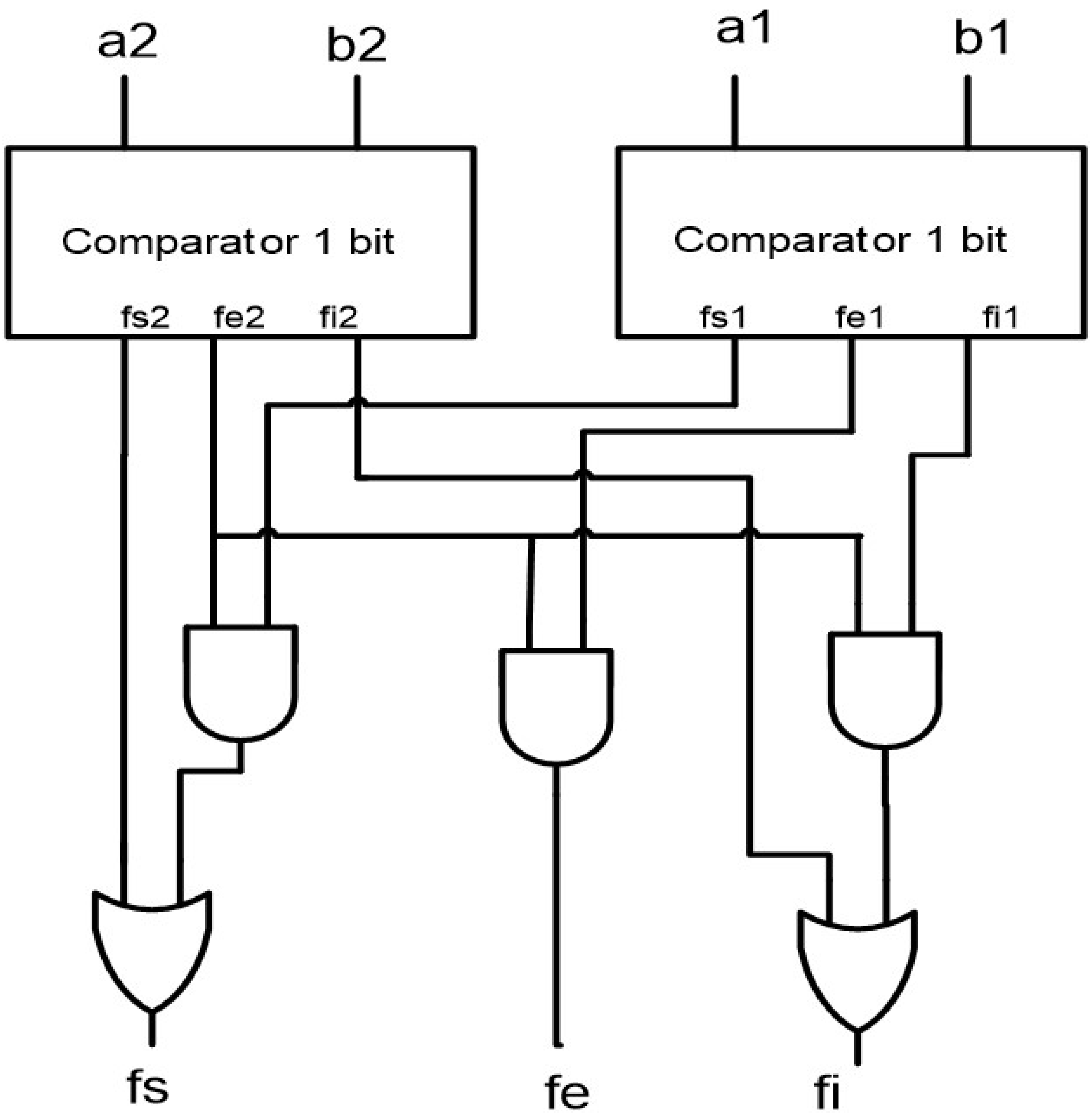
A2 > B2 ou (A2=B2 and A1>B1)

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

3. A<B si

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

$$fi = \overline{A2}.B2 + (\overline{A2 \oplus B2}).(\overline{A1}.B1) = fi2 + fe2.fi1$$

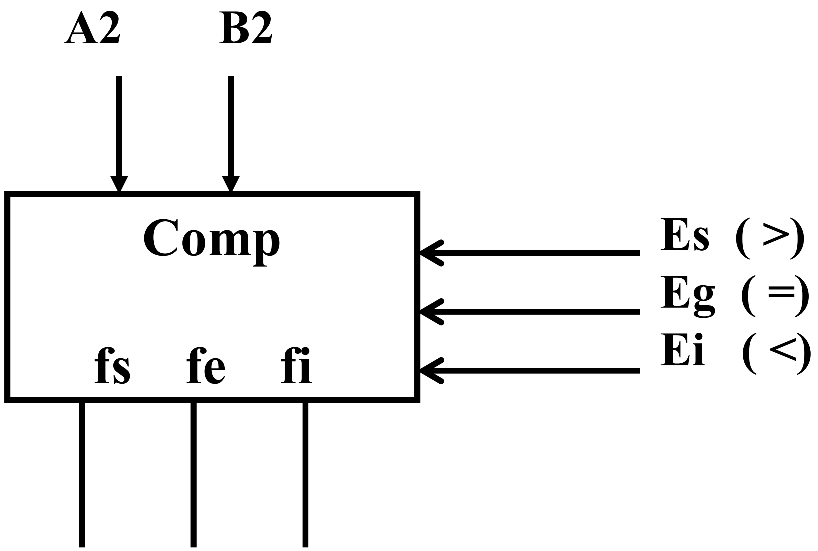


a2
b2
a1
b1
Comparator 1 bit
fs2
fe2
fi2
Comparator 1 bit
fs1
fe1
fi1
fs
fe
fi

Comparator with cascading inputs

- We notice that:
  - If $A2 > B2$ then $A > B$
  - If $A2 < B2$ then $A < B$
- However, if $A2 = B2$, then we need to consider the comparison result of the least significant bits.
- To do this, we add to the comparator inputs that indicate the result of the previous comparison.
- These inputs are called cascading inputs.

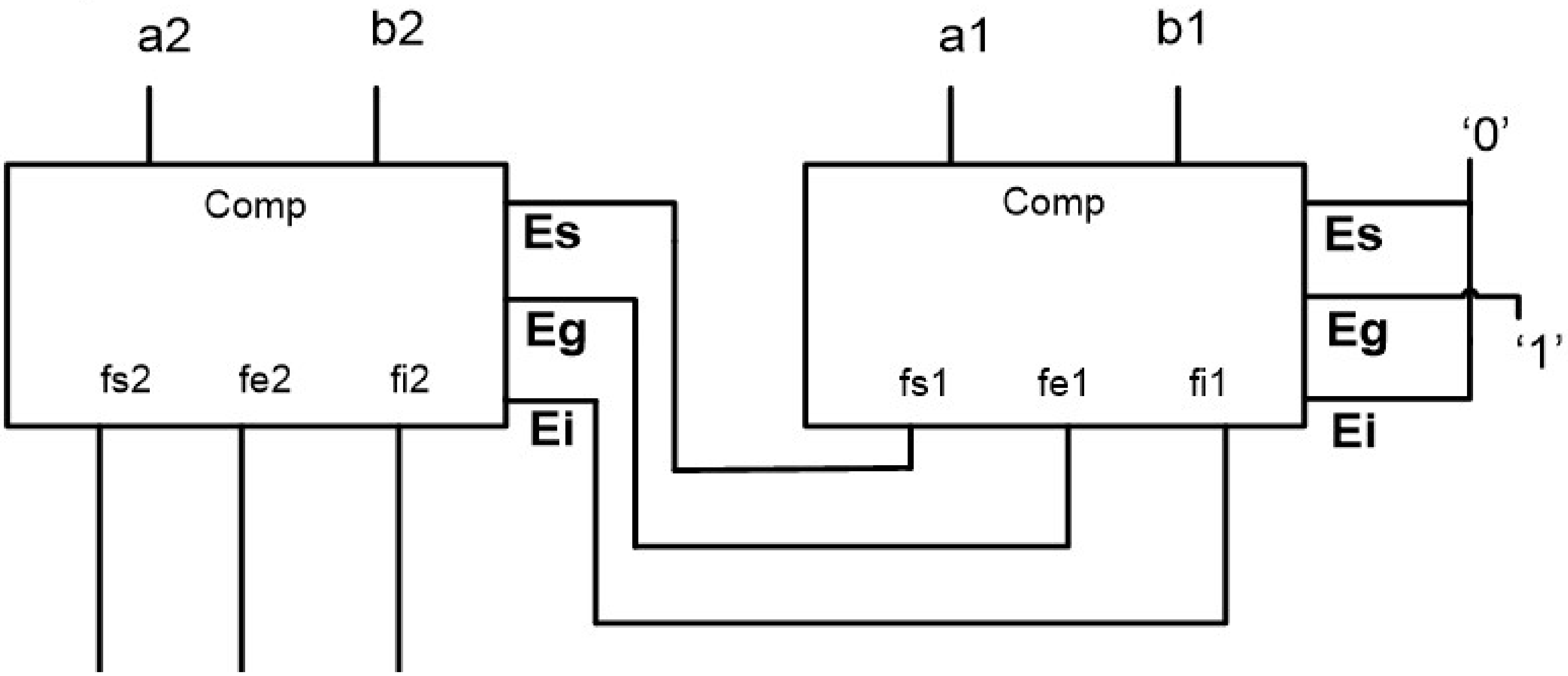
A2	B2	Es	Eg	Ei		fs	fe	fs
A2>B2		X	X	X		1	0	0
A2<B2		X	X	X		0	0	1
A2=B1		1	0	0		1	0	0
		0	1	0		0	1	0
		0	0	1		0	0	1



fs= (A2>B2) ou (A2=B2).Es

fi= (A2<B2) ou (A2=B2).Ei

fe= (A2=B2).Eg

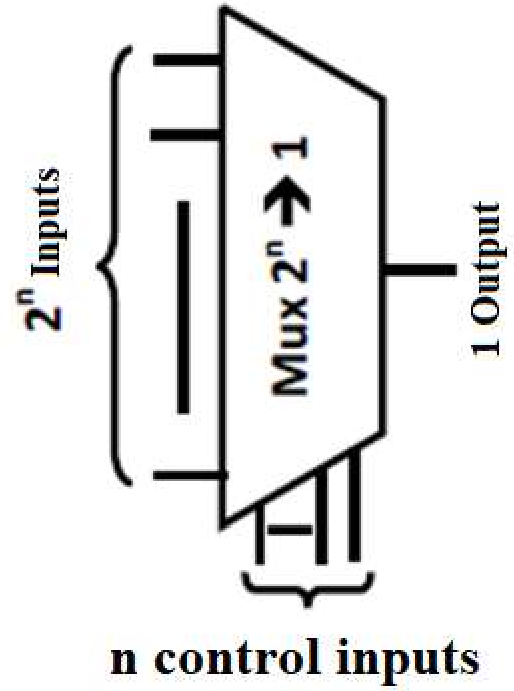

a2
b2
a1
b1
'0'
Comp
Comp
Es
Eg
Ei
Es
Eg
'1'
Ei
fs2
fe2
fi2
fs1
fe1
fi1

Exercice

- Implement a 4-bit comparator using cascaded 2-bit comparators?

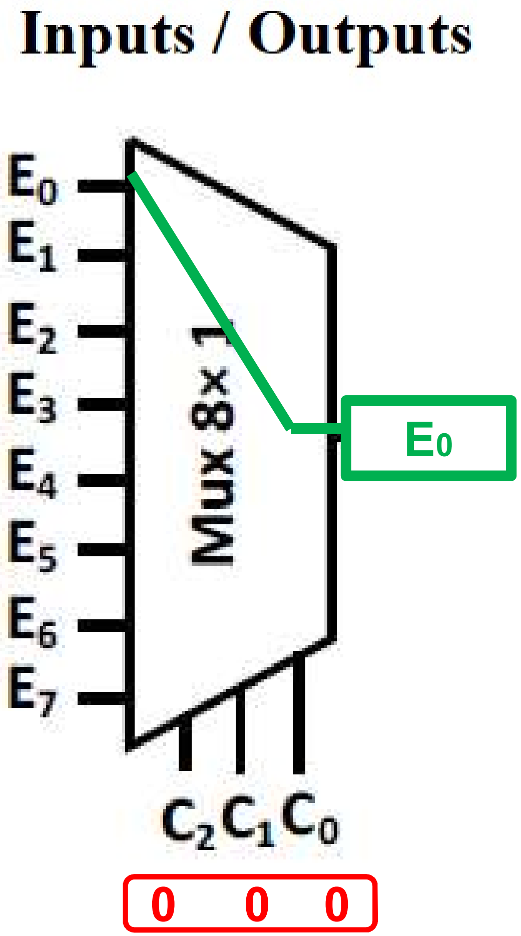
Multiplexer

- Definition: A multiplexer is a combinational logic circuit that has 2^n inputs, n control inputs, and a single output. It allows routing the value of the input line specified by its control inputs to the output line.



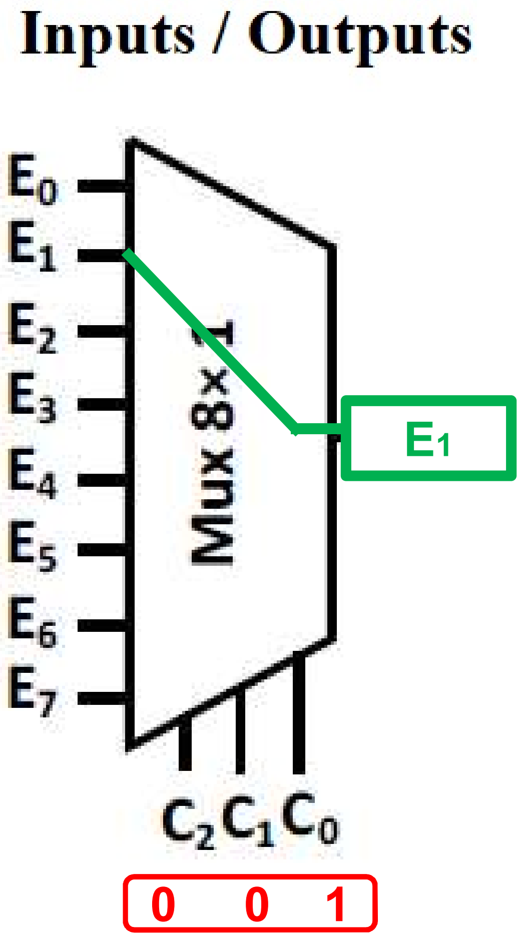
Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:



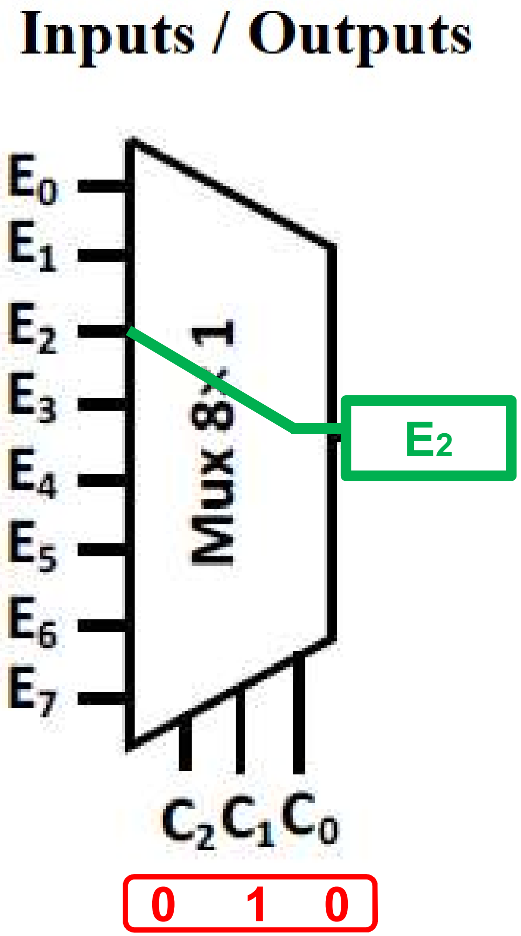
Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:



Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

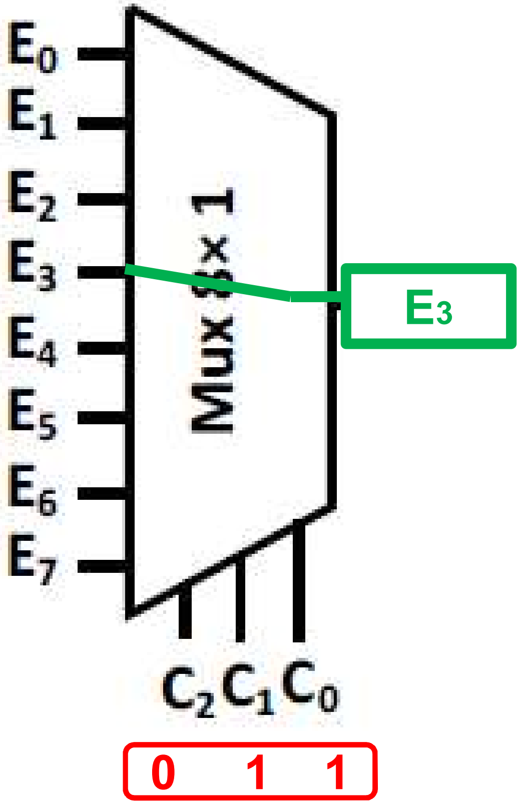


Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

Inputs / Outputs

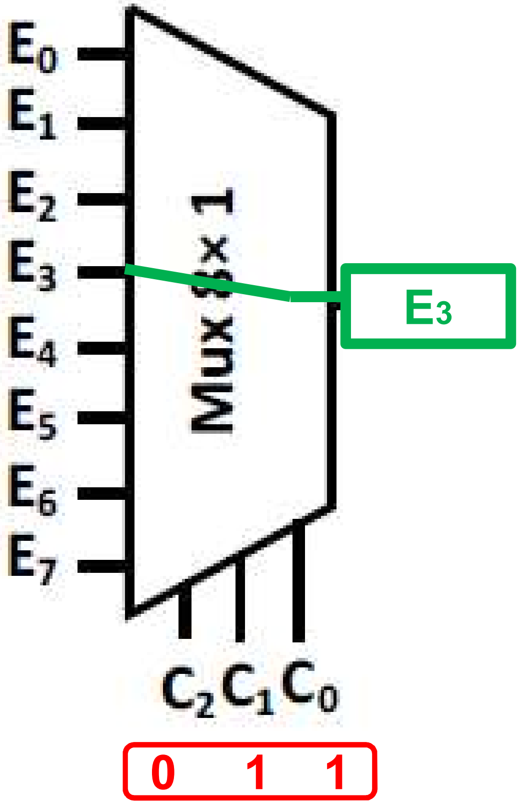
the truth table



Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

Inputs / Outputs

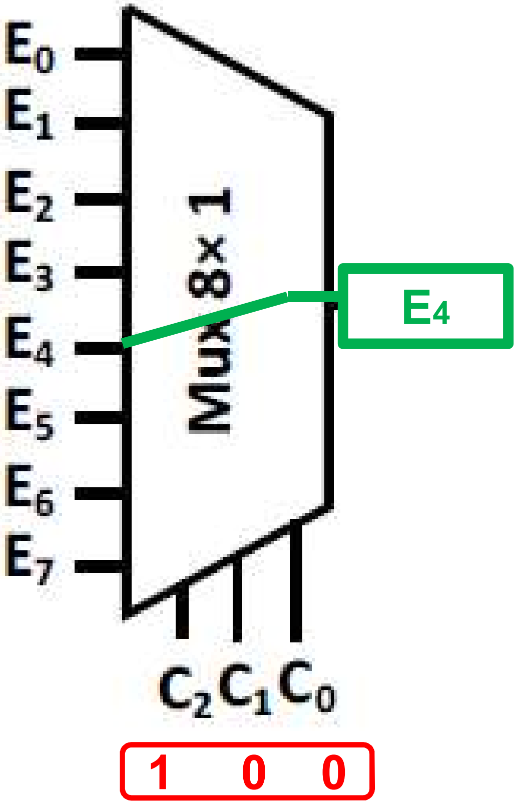


the truth table

Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

Inputs / Outputs



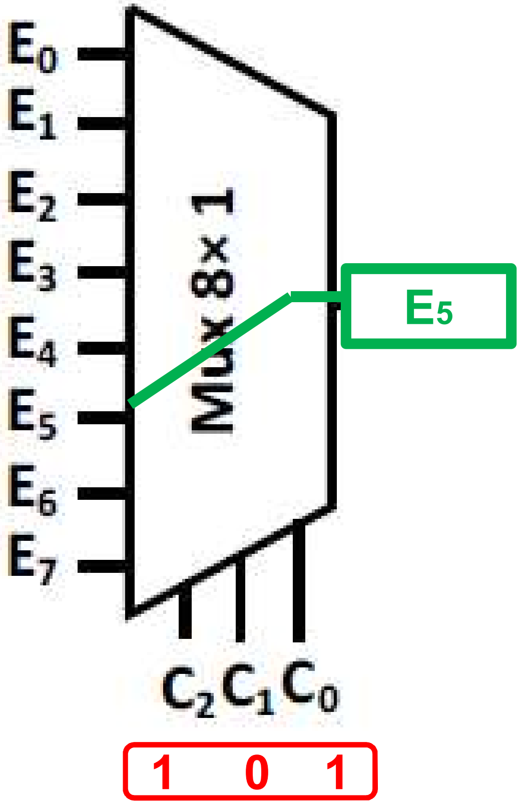
the truth table

C_2	C_1	C_0	S

Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

Inputs / Outputs



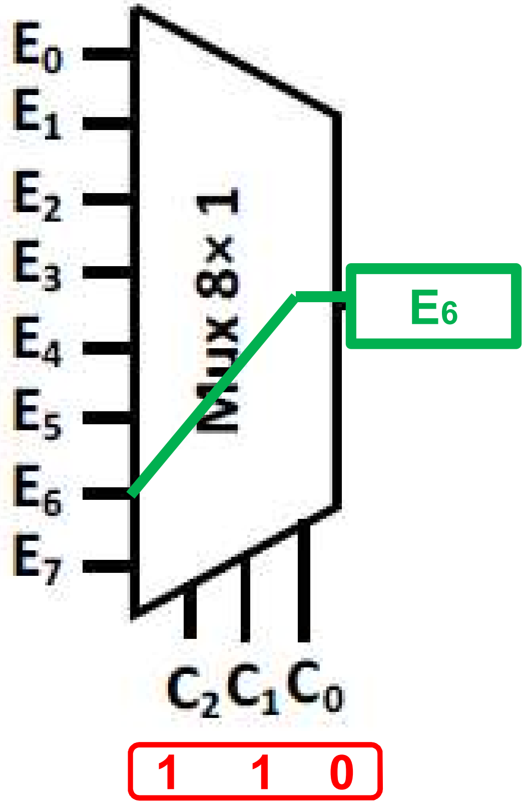
the truth table

C_2	C_1	C_0	S
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

Inputs / Outputs



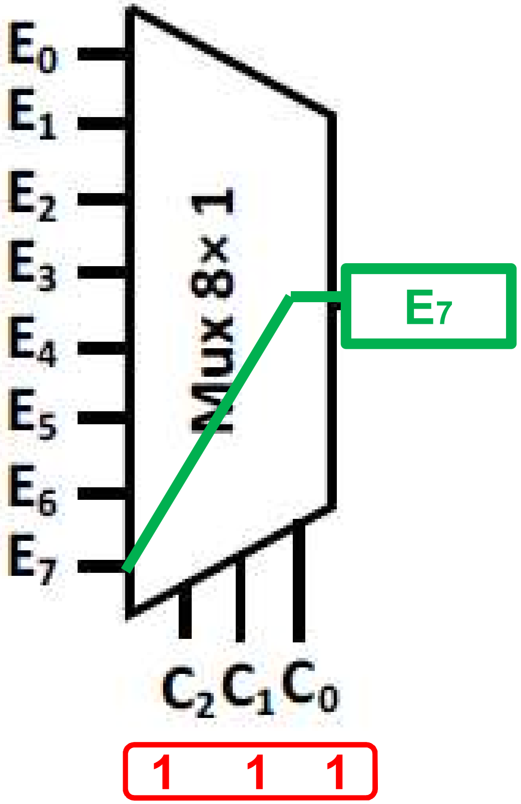
the truth table

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

Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

Inputs / Outputs



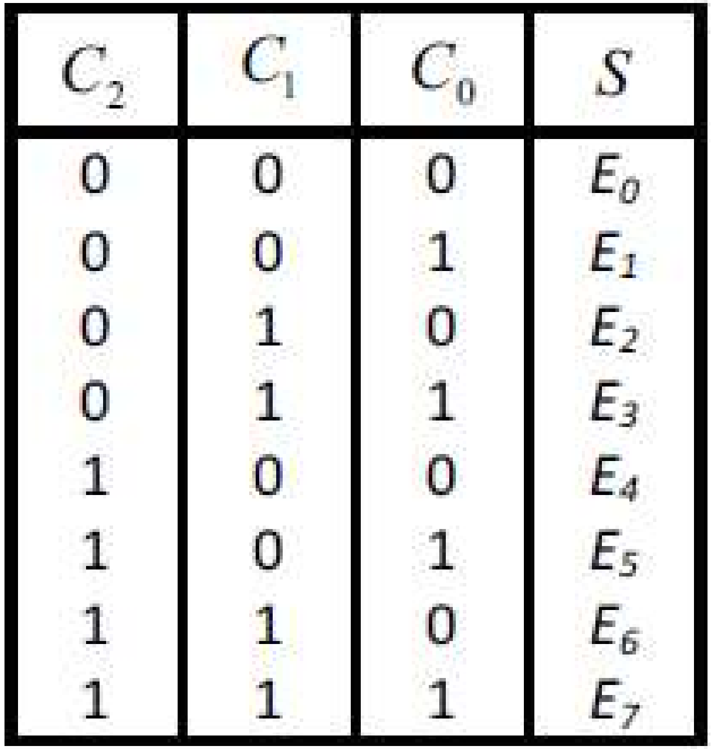
the truth table

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

Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

the truth table



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

logical functions

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

Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

logical functions

$$
\begin{aligned}
S = {} & \overline{C}_2.\overline{C_1}.\overline{C_0}.E_0 \quad + \\
& \overline{C_2}.\overline{C_1}.C_0.E_1 \quad + \\
& \overline{C_2}.C_1.\overline{C_0}.E_2 \quad + \\
& \overline{C_2}.C_1.C_0.E_3 \quad + \\
& C_2.\overline{C_1}.\overline{C_0}.E_4 \quad + \\
& C_2.\overline{C_1}.C_0.E_5 \quad + \\
& C_2.C_1.\overline{C_0}.E_6 \quad + \\
& C_2.C_1.C_0.E_7
\end{aligned}
$$

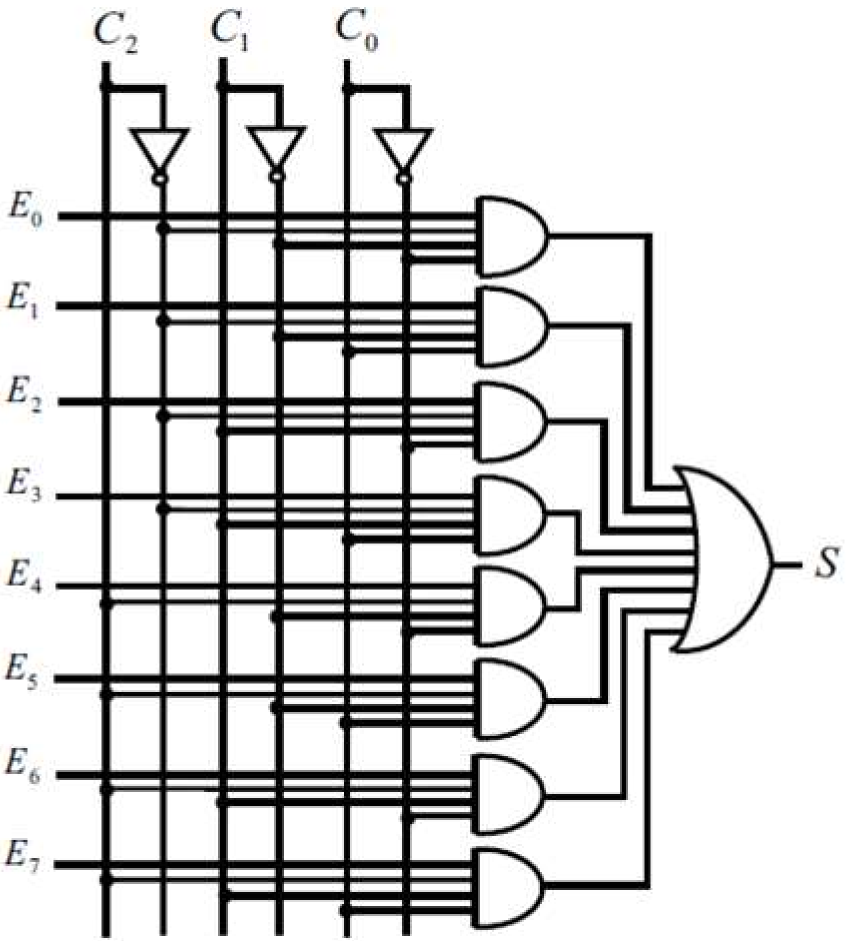
Multiplexer

- Synthesis of the circuit (8x1 multiplexer) example:

logical functions

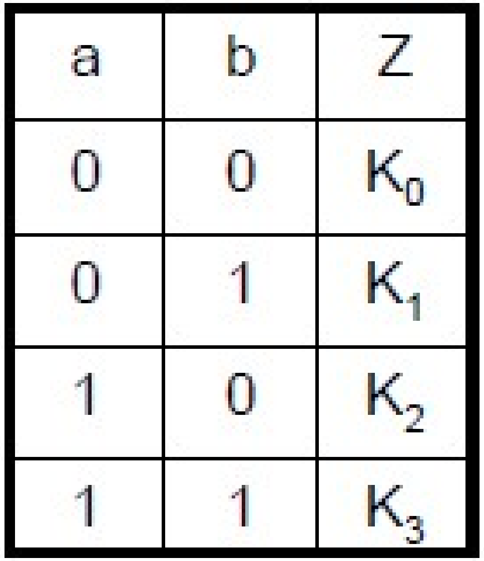
$$
\begin{aligned}
S = {} & \overline{C_2}.\overline{C_1}.\overline{C_0}.E_0 \; + \\
& \overline{C_2}.\overline{C_1}.C_0.E_1 \; + \\
& \overline{C_2}.C_1.\overline{C_0}.E_2 \; + \\
& \overline{C_2}.C_1.C_0.E_3 \; + \\
& C_2.\overline{C_1}.\overline{C_0}.E_4 \; + \\
& C_2.\overline{C_1}.C_0.E_5 \; + \\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

the circuit diagram



Multiplexer

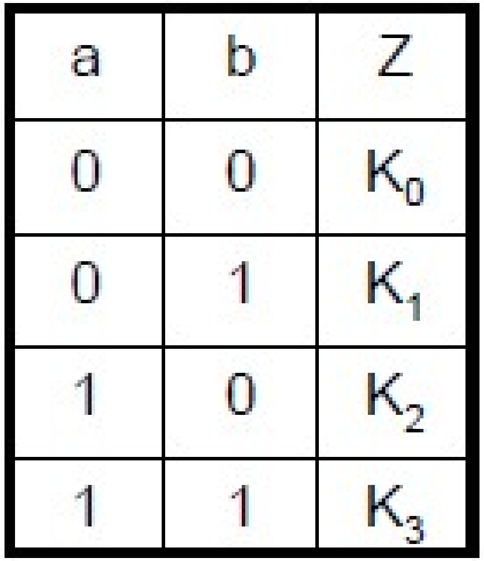
- The multiplexer is a combinational **selector** circuit that has **2^n** data inputs, **n** control inputs, and a single output. Its role is to select, using control signals, one of the inputs and connect it to the output.



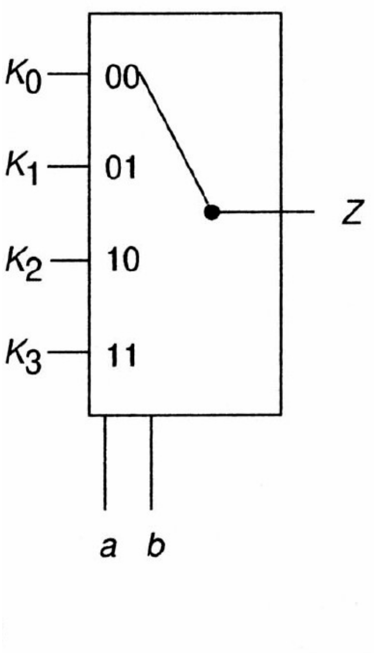
a	b	Z
0	0	K_0
0	1	K_1
1	0	K_2
1	1	K_3

Multiplexer

- The multiplexer is a combinational **selector** circuit that has **2^n** data inputs, **n** control inputs, and a single output. Its role is to select, using control signals, one of the inputs and connect it to the output.

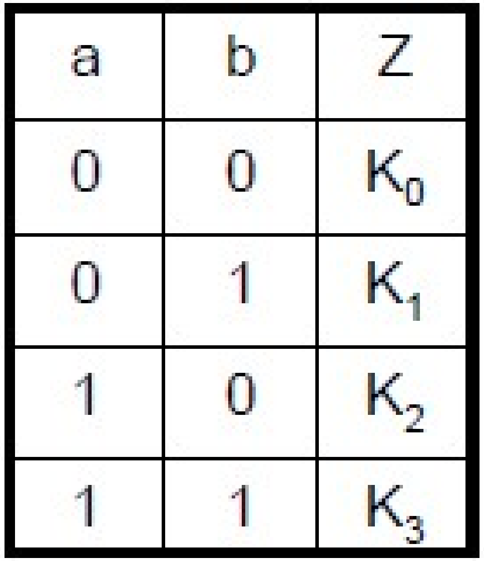


a	b	Z
0	0	K_0
0	1	K_1
1	0	K_2
1	1	K_3

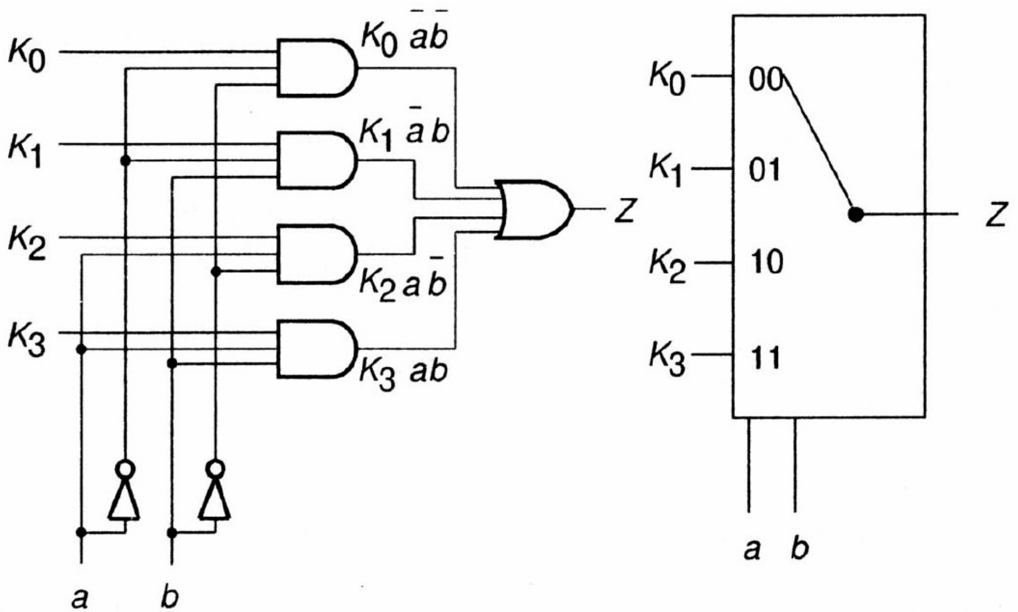


Multiplexer 2 →1

- The multiplexer is a combinational **selector** circuit that has **2^n** data inputs, **n** control inputs, and a single output. Its role is to select, using control signals, one of the inputs and connect 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)

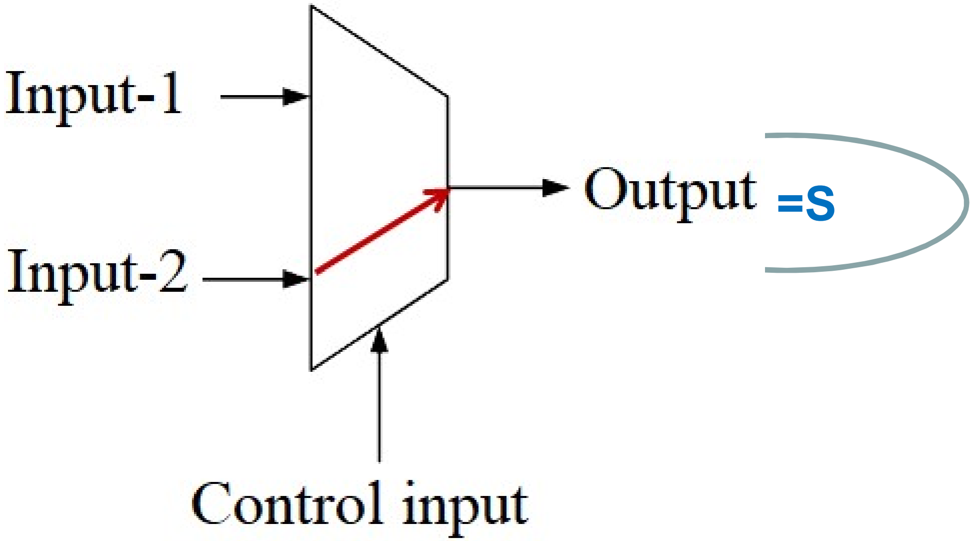
Multiplexer

Description of the behavior of the 2-to-1 multiplexer:

The output S takes on the value of the data input:

- En1 when the selection input Com is active (logic level 1).
- En2 when the selection input Com is inactive (logic level 0). The input Com thus directs either the information arriving from input En1 or that arriving from input En2

to the output S.



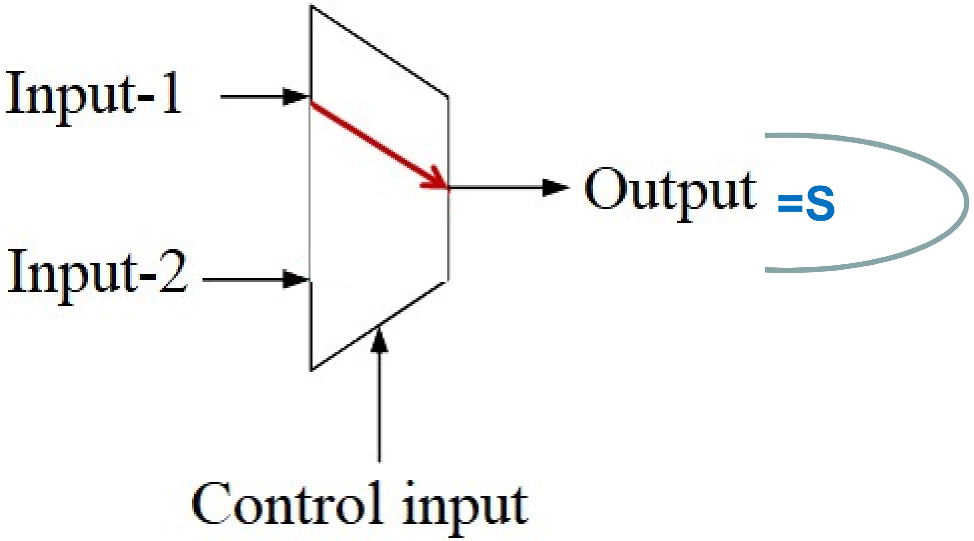
Multiplexer

Description of the behavior of the 2-to-1 multiplexer:

The output S takes on the value of the data input:

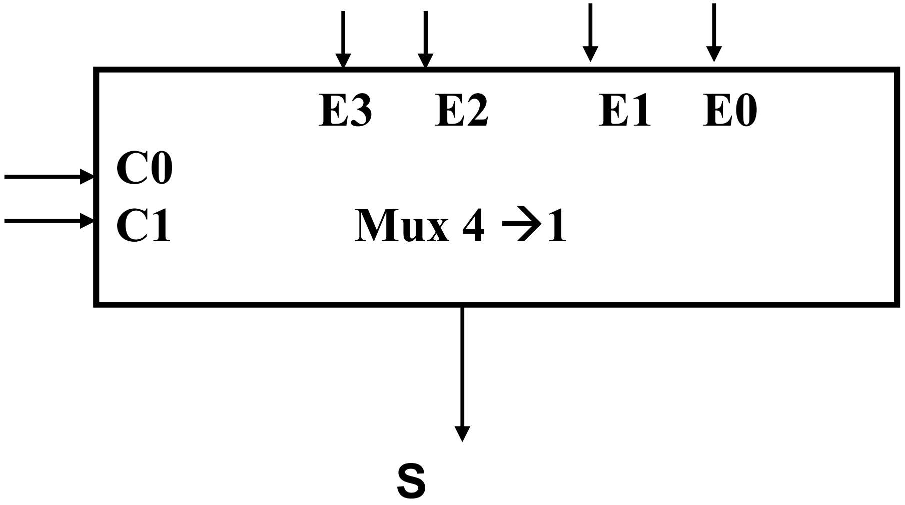
- En1 when the selection input Com is active (logic level 1).
- En2 when the selection input Com is inactive (logic level 0). The input Com thus directs either the information arriving from input En1 or that arriving from input En2

  to the output S.



Multiplexer 4 →1

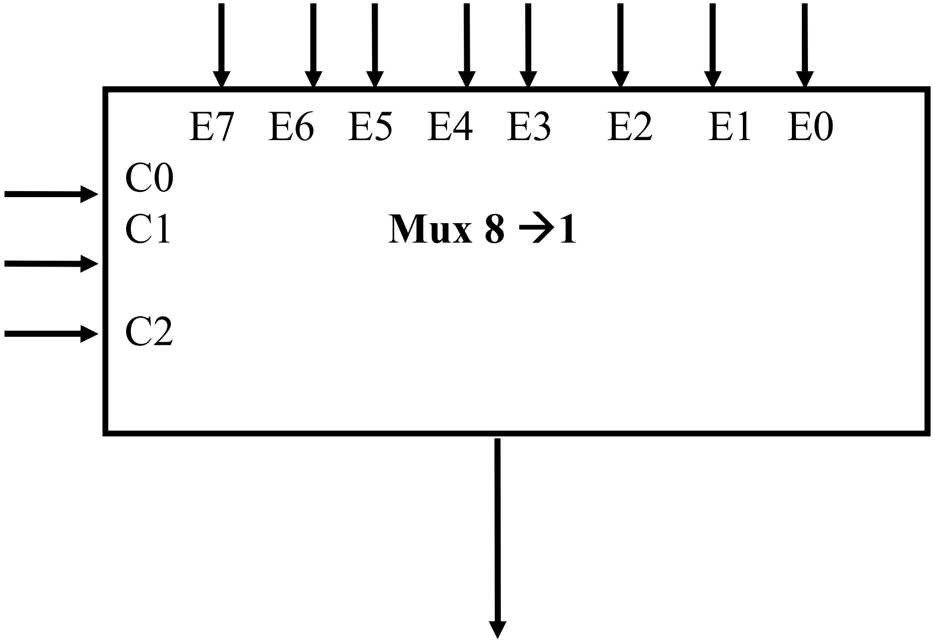
C1	C0		S
0	0		E0
0	1		E1
1	0		E2
1	1		E3



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

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



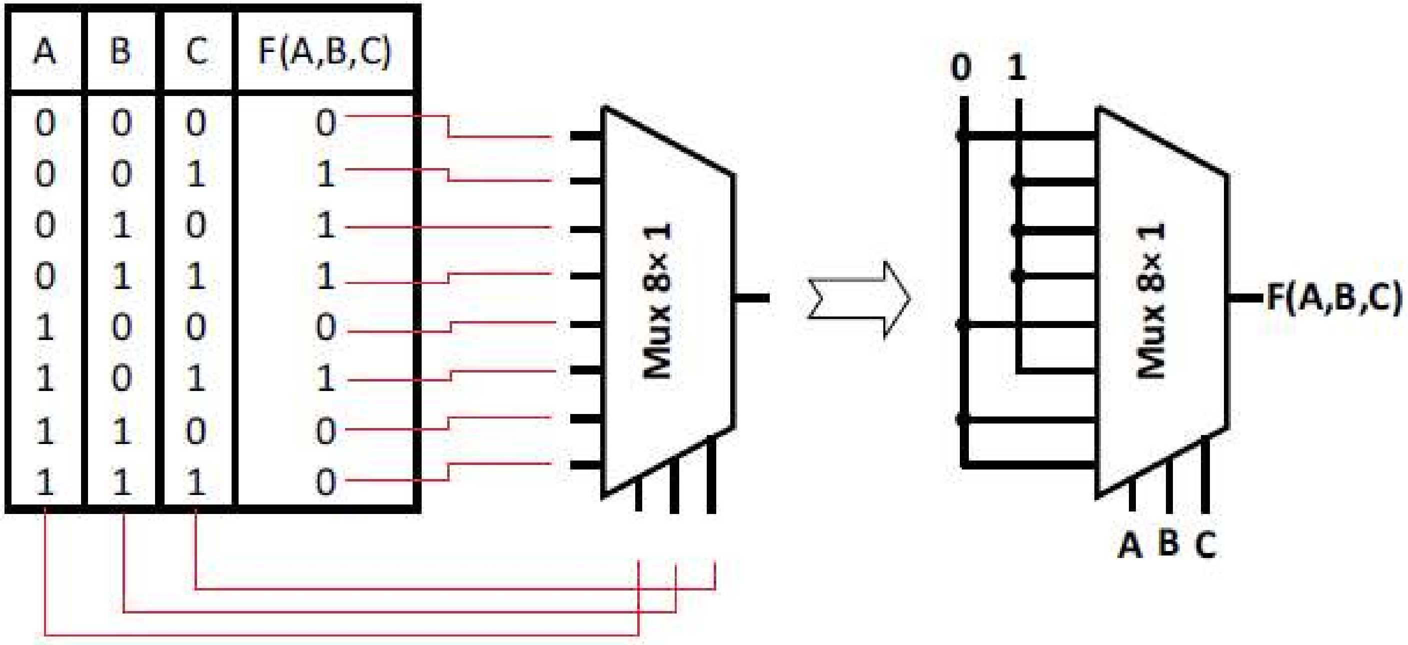
$$S = \overline{C2}.\overline{C1}.\overline{C0}.(E0) + \overline{C2}.\overline{C1}.C0(E1) + \overline{C2}.C1.\overline{C0}(E2) + \overline{C2}.C1.C0(E3) +$$
$$C2.\overline{C1}.\overline{C0}(E4) + C2.\overline{C1}.C0(E5) + C2.C1.\overline{C0}(E6) + C2.C1.C0(E7)$$

Logical functions using multiplexers:

One can always generate any logical functions using multiplexers and basic logic gates. It is sufficient to connect the variables of the function to be generated to the various inputs of the multiplexer (standard inputs and control inputs).

Example 1: Implement the following 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

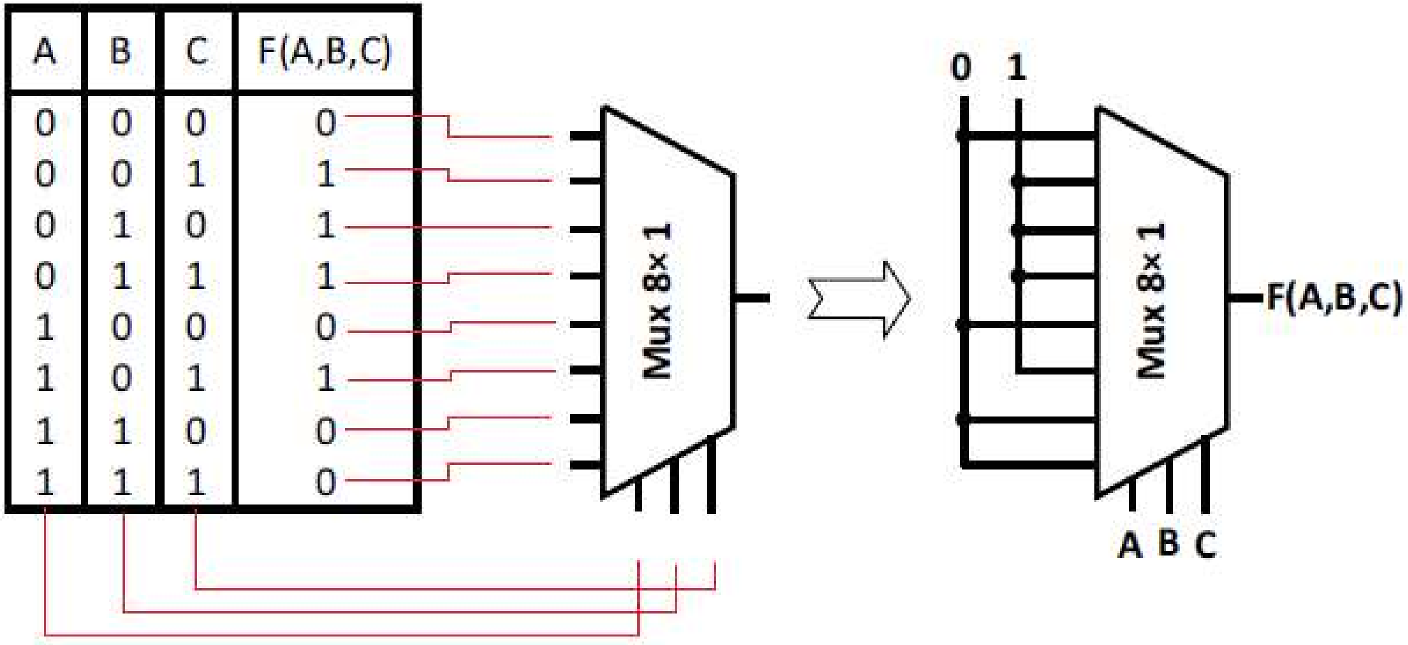
Logical functions using multiplexers:

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

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

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

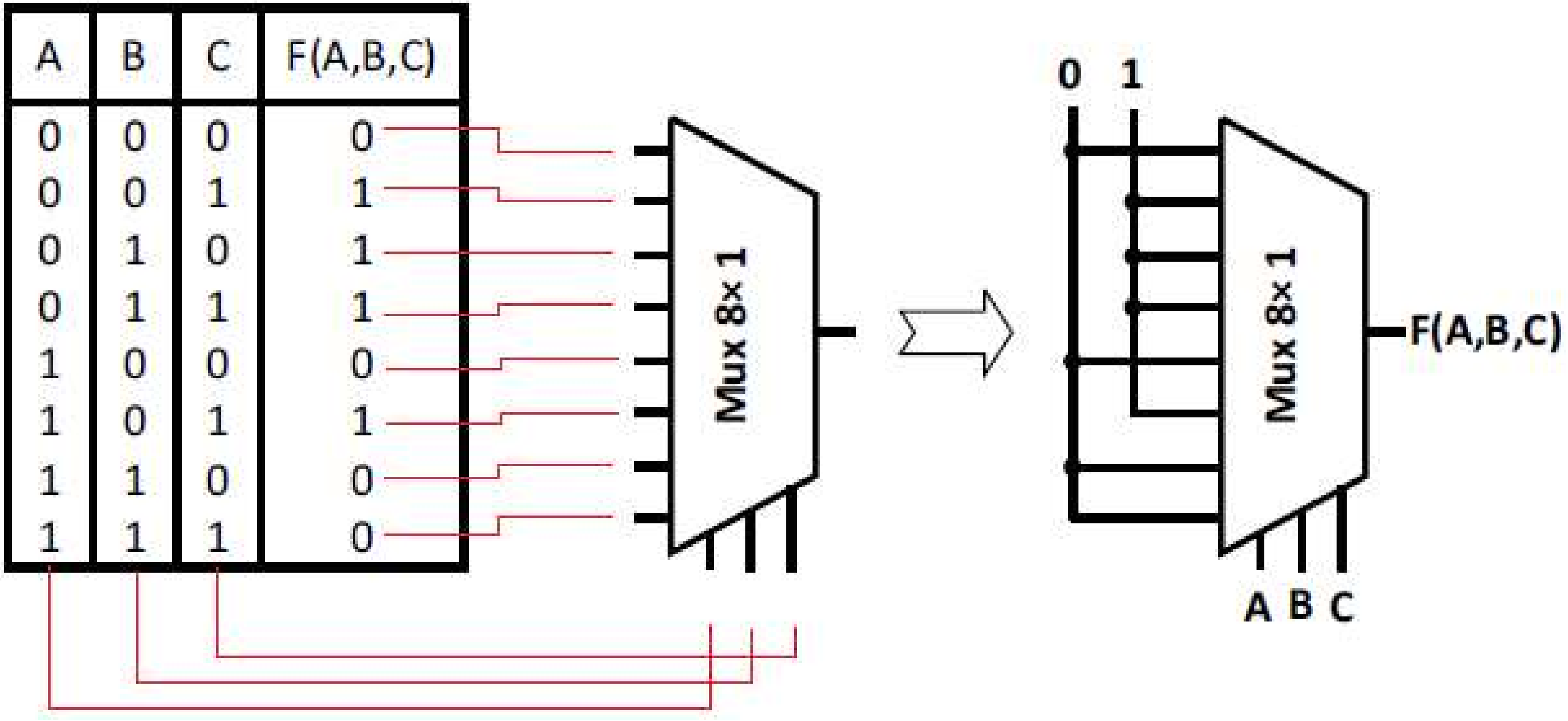
$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$



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

Logical functions using multiplexers:

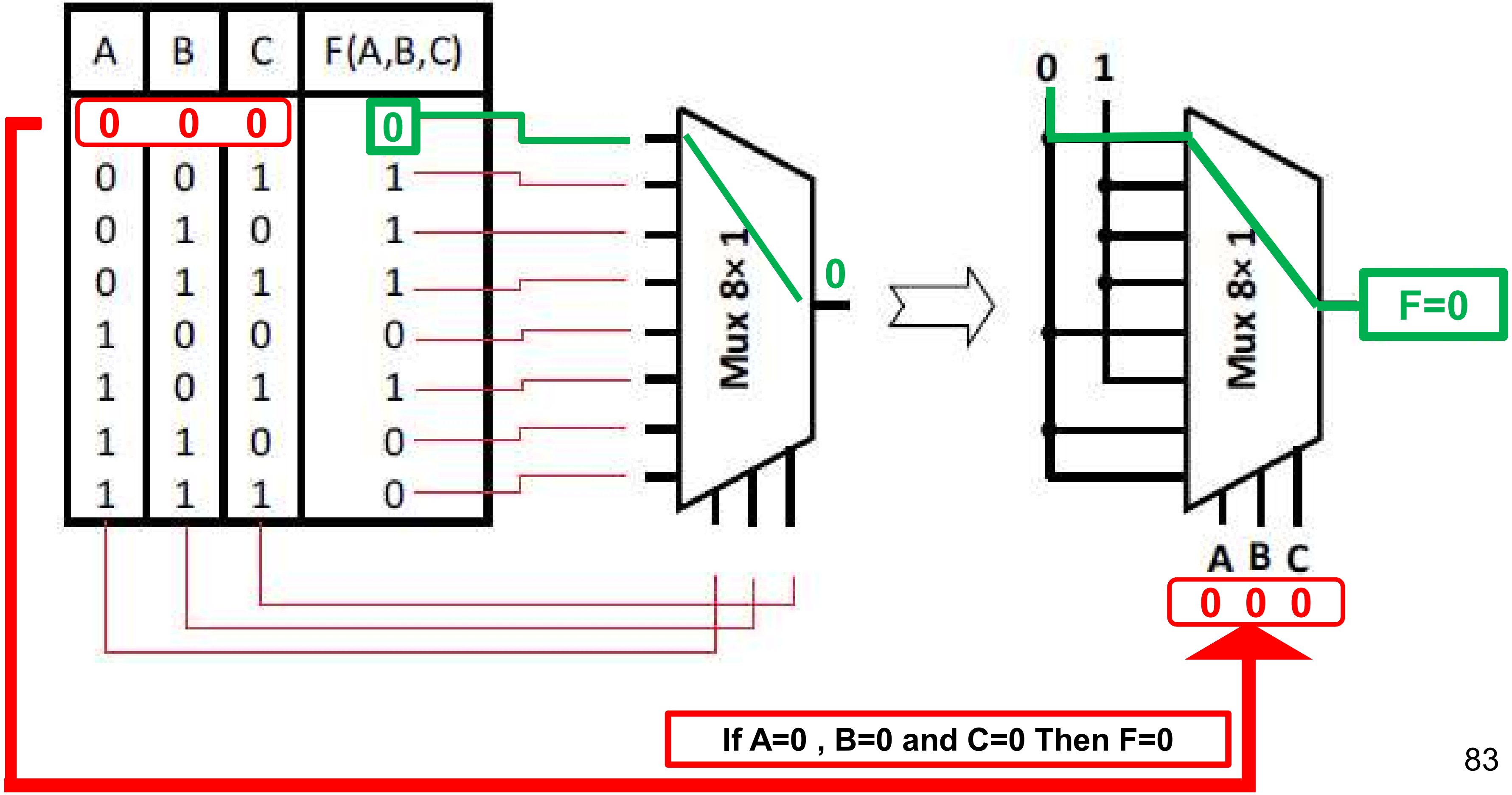
$$F(A,B,C)=\bar{A}\,\bar{B}\,C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$



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

Logical functions using multiplexers:

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

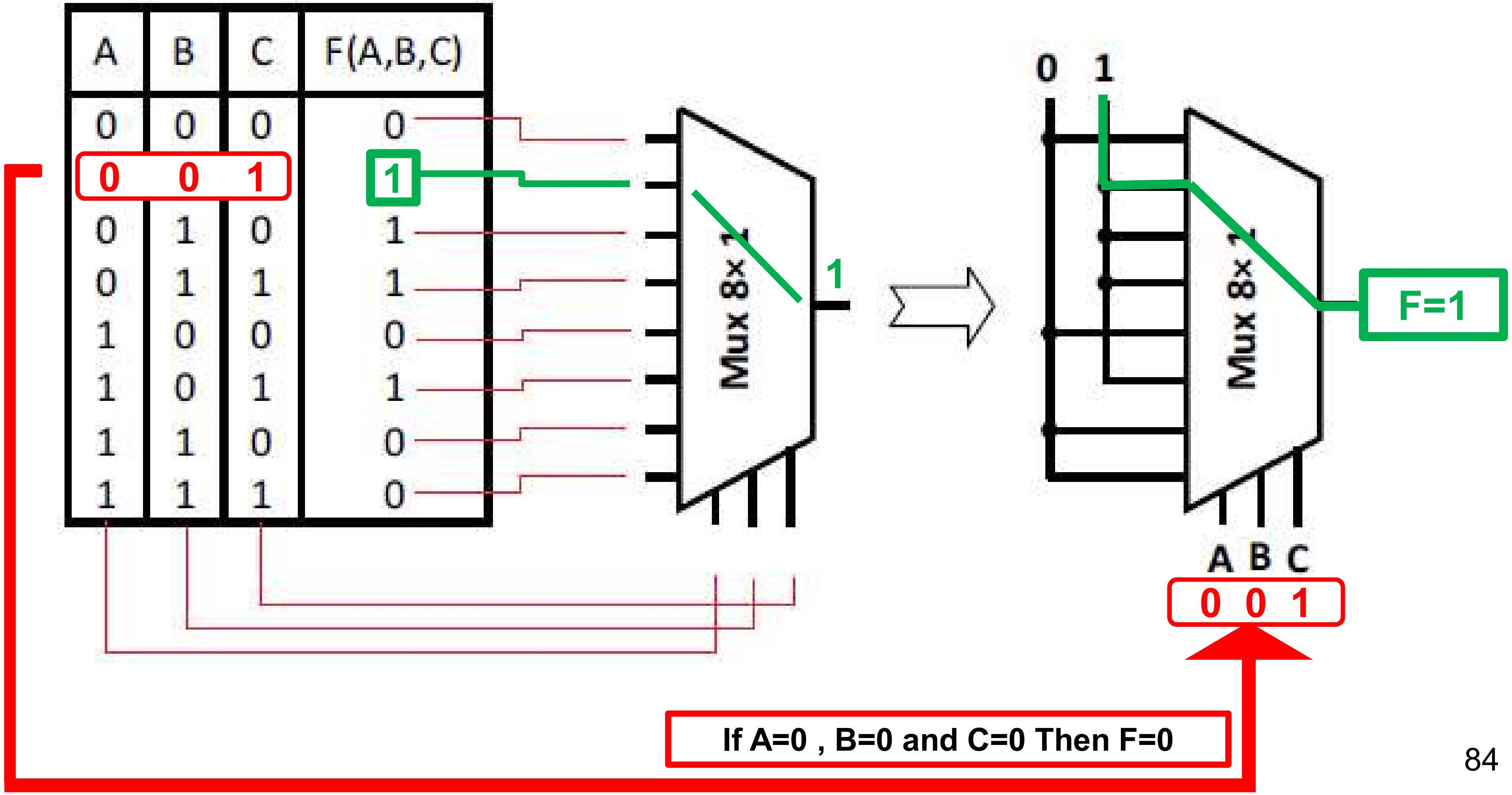


A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

If A=0 , B=0 and C=0 Then F=0

Logical functions using multiplexers:

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

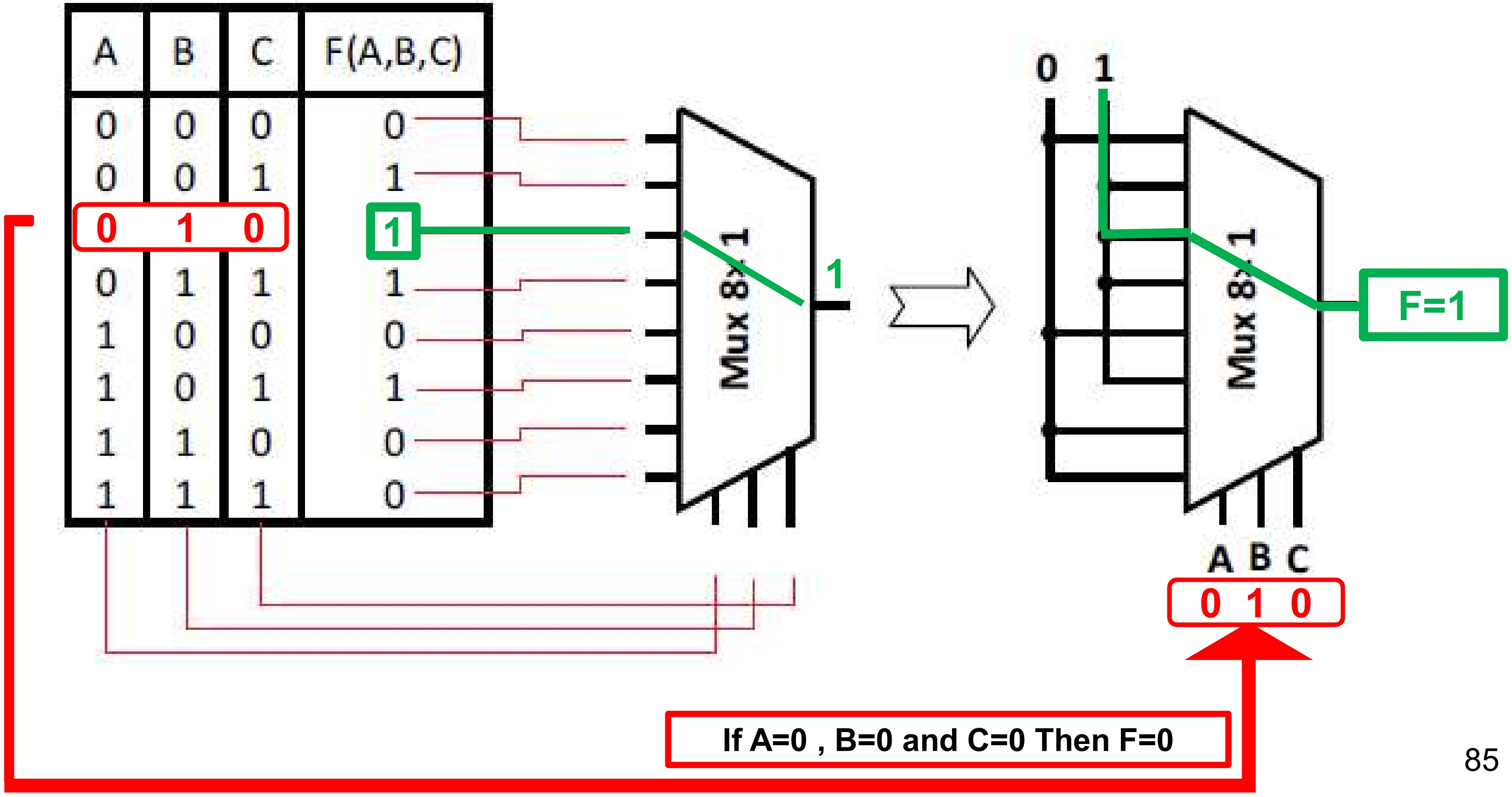


A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

If A=0 , B=0 and C=0 Then F=0

Logical functions using multiplexers:

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

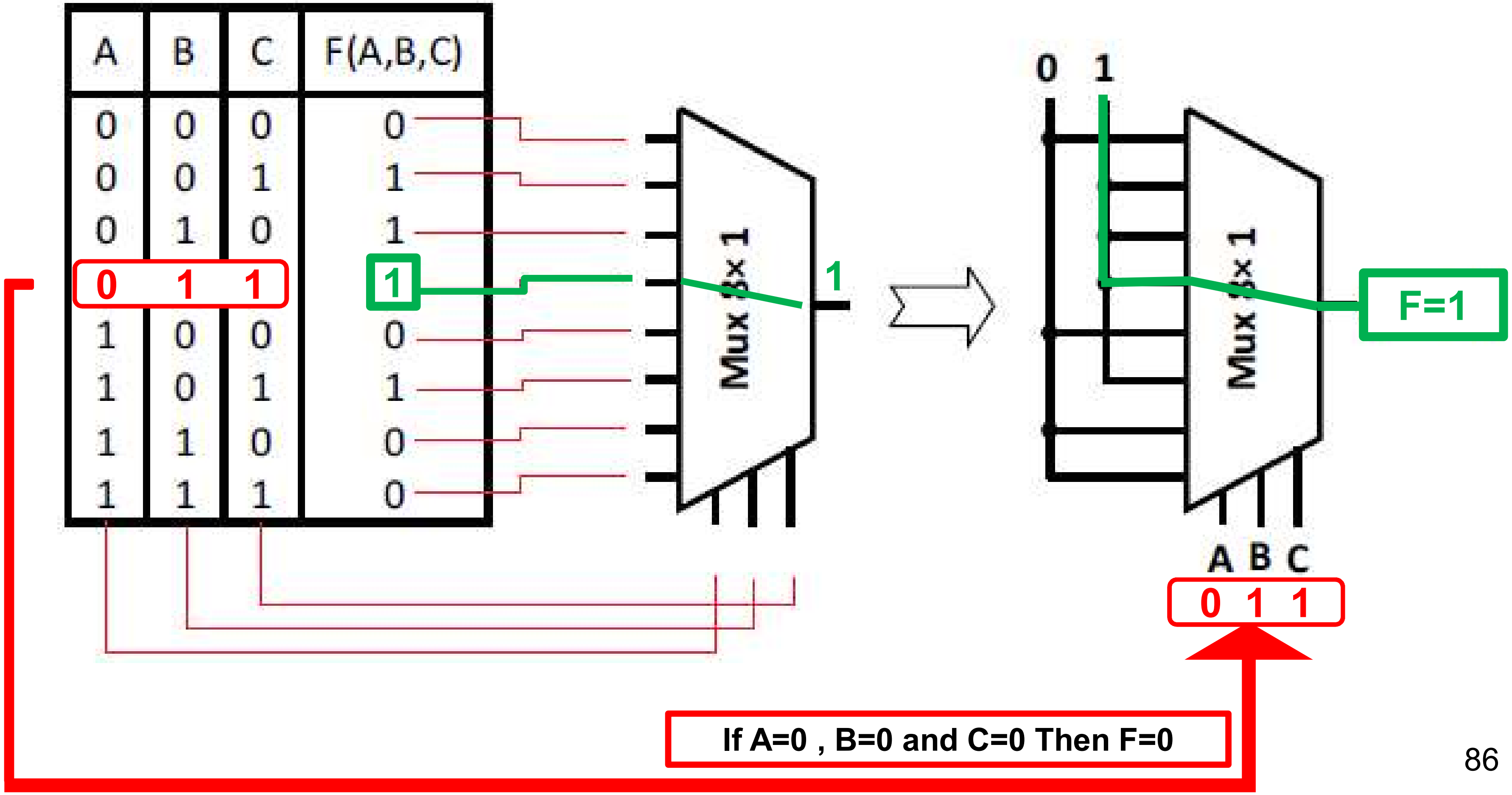


A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

If A=0 , B=0 and C=0 Then F=0

Logical functions using multiplexers:

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

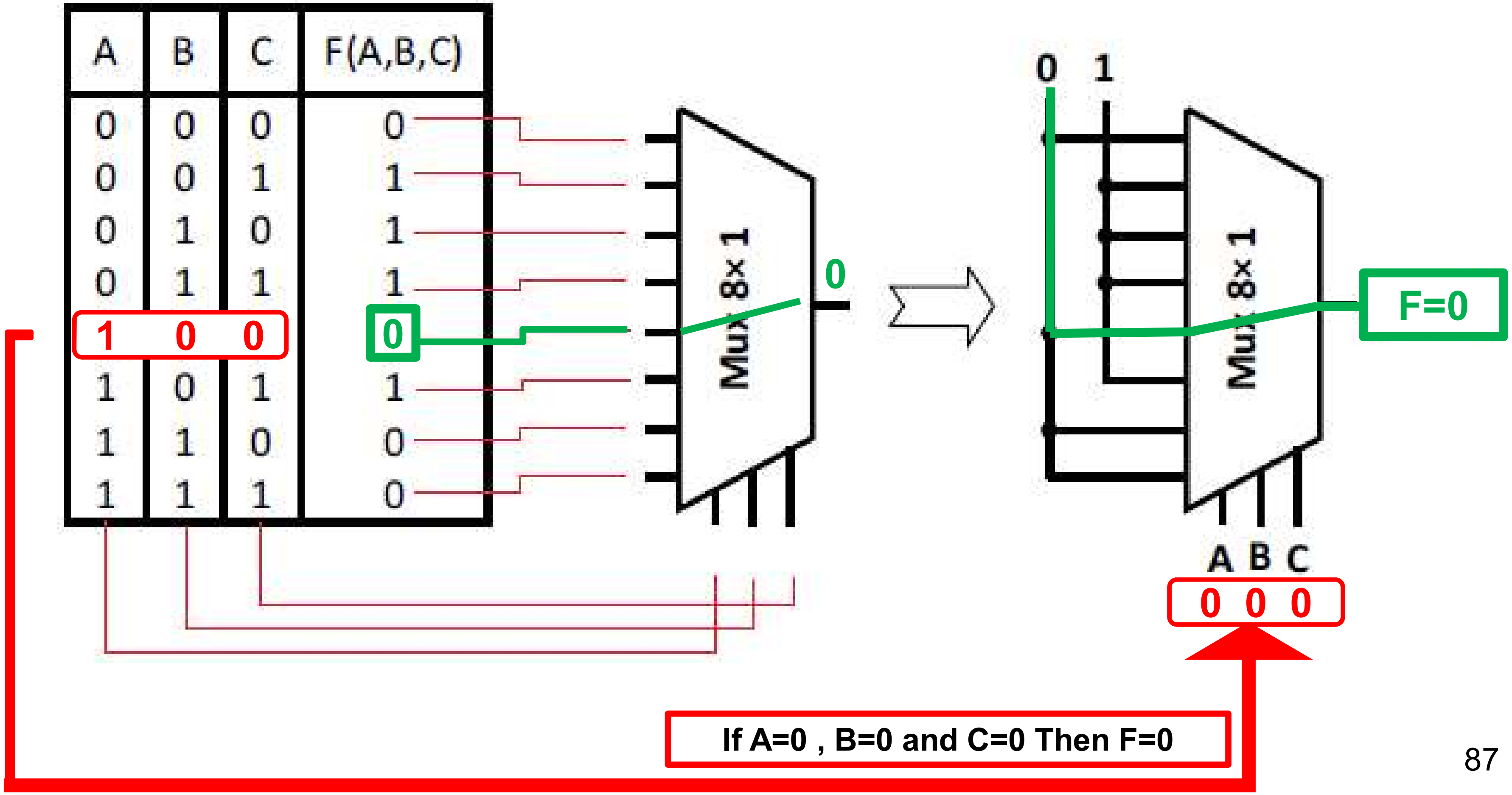


A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

If A=0 , B=0 and C=0 Then F=0

Logical 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

If A=0 , B=0 and C=0 Then F=0

Logical functions using multiplexers:

Example 1: Implement $F(A,B,C) = \overline{A}.B + \overline{B}.C$

The function using a 4x1 multiplexer.

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

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

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

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

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

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

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

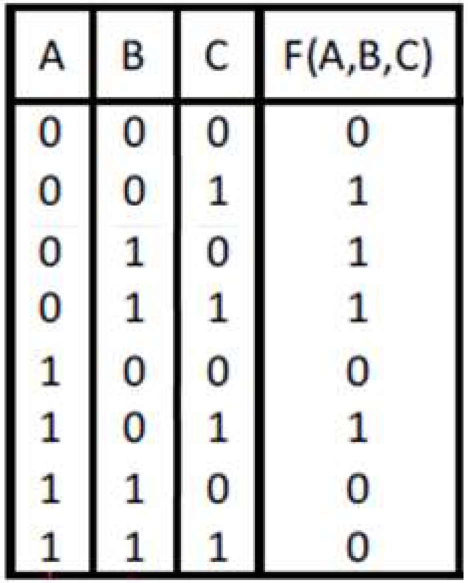
$$F(A,B,C)=\overline{A}.B+\overline{B}.C$$

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

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

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

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



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

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

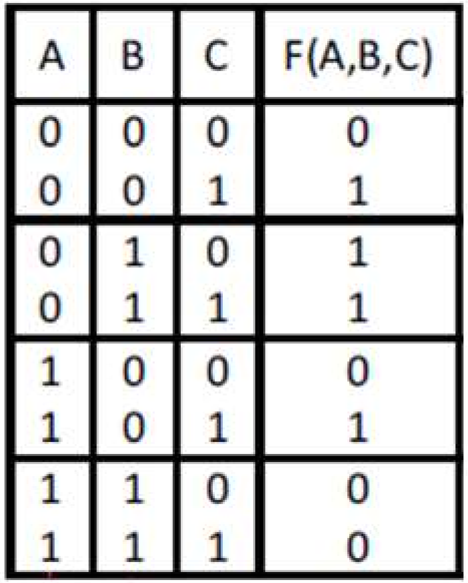
$$F(A, B, C) = \overline{A}.B + \overline{B}.C$$

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

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

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

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



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

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

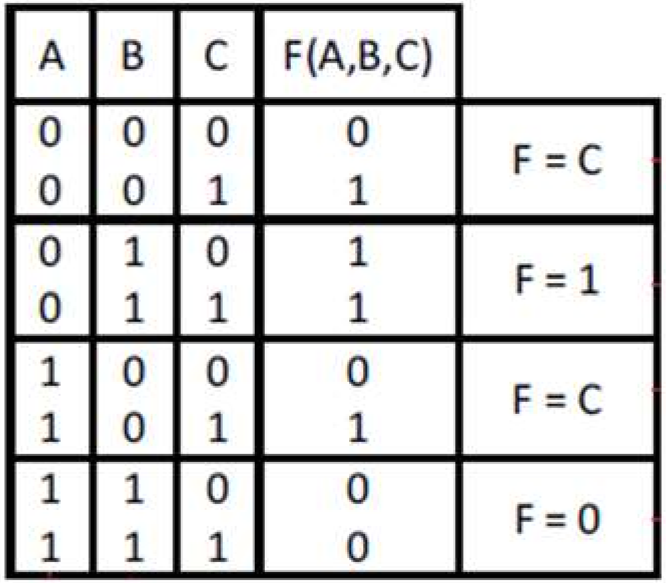
$$F(A,B,C)=\overline{A}.B+\overline{B}.C$$

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

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

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

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



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	

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

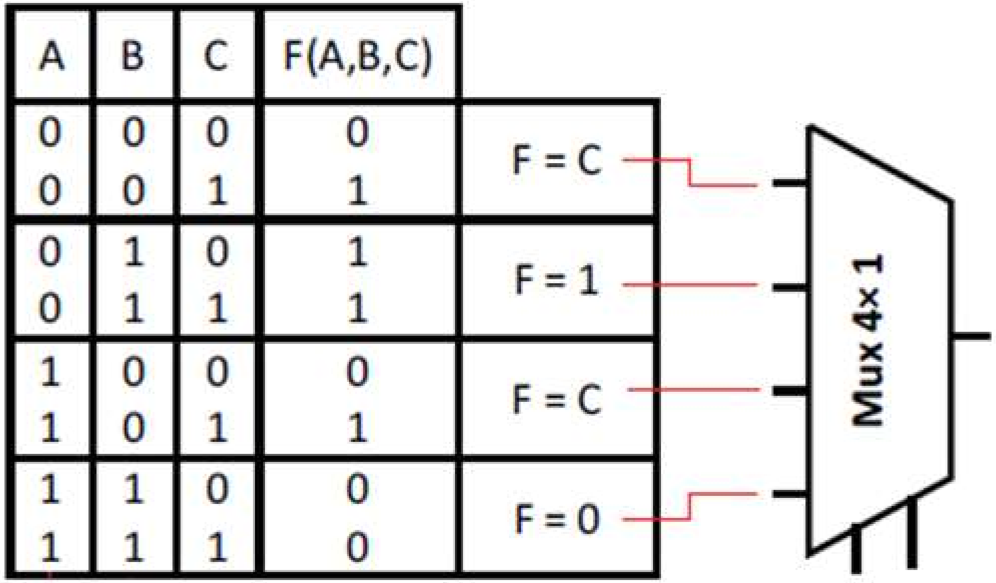
$$F(A,B,C)=\overline{A}.B+\overline{B}.C$$

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

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

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

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



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	

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

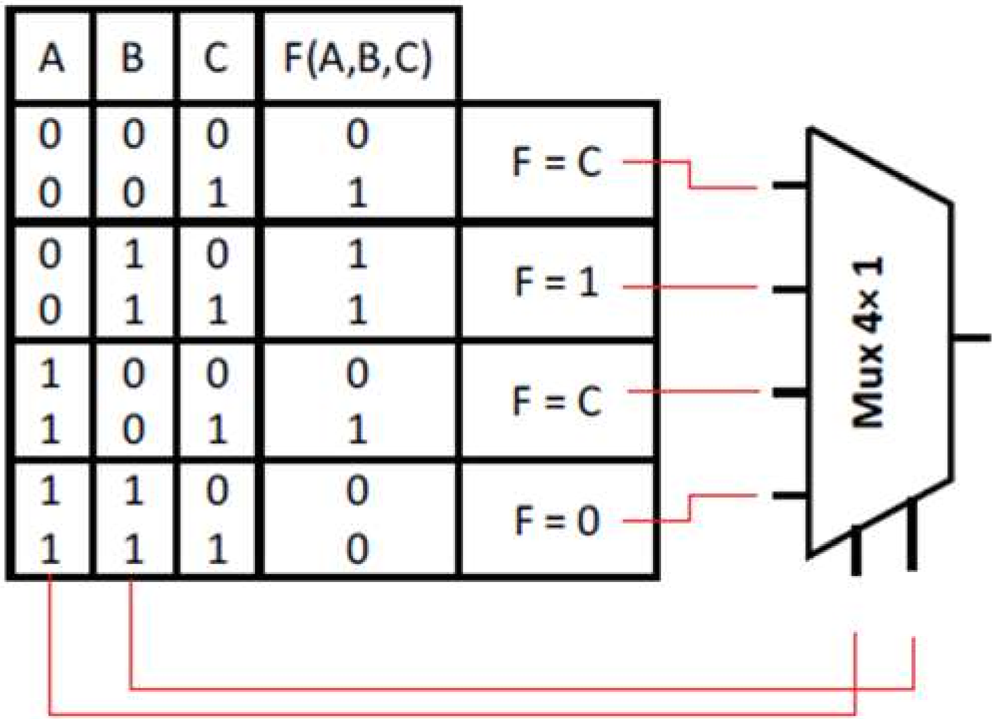
$$F(A,B,C) = \overline{A}.B + \overline{B}.C$$

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

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

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

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



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	

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

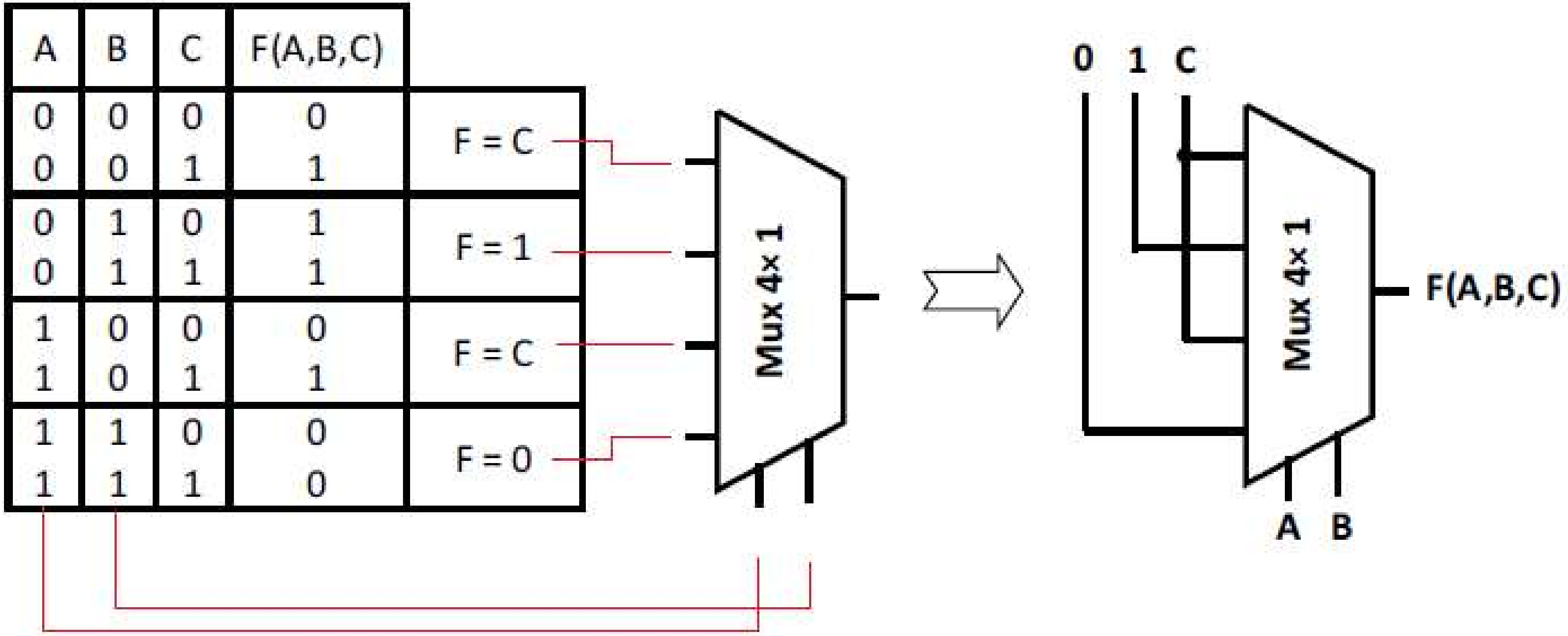
$$F(A,B,C)=\overline{A}.B+\overline{B}.C$$

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

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

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

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



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	

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

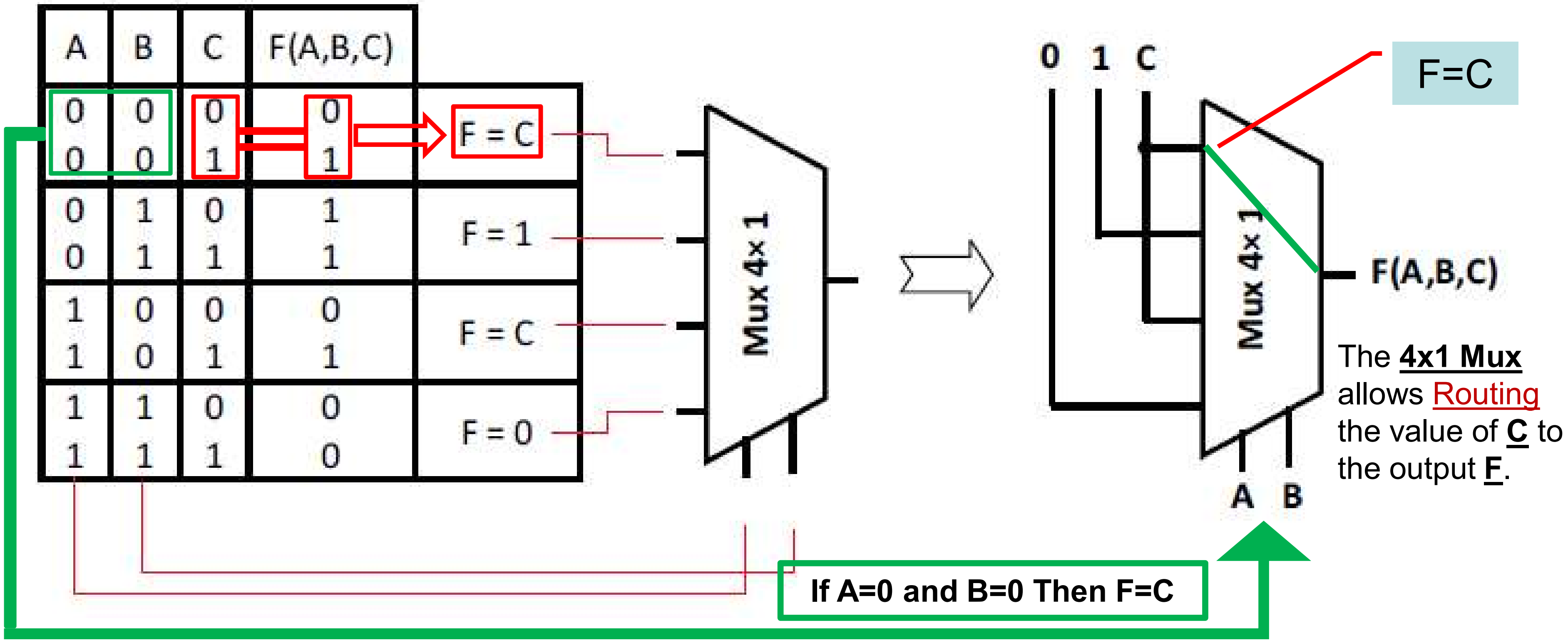
$$F(A,B,C) = \bar{A}.B + \bar{B}.C$$

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

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

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

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



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

F=C

The **4x1 Mux** allows Routing the value of **C** to the output **F**.

If A=0 and B=0 Then F=C

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

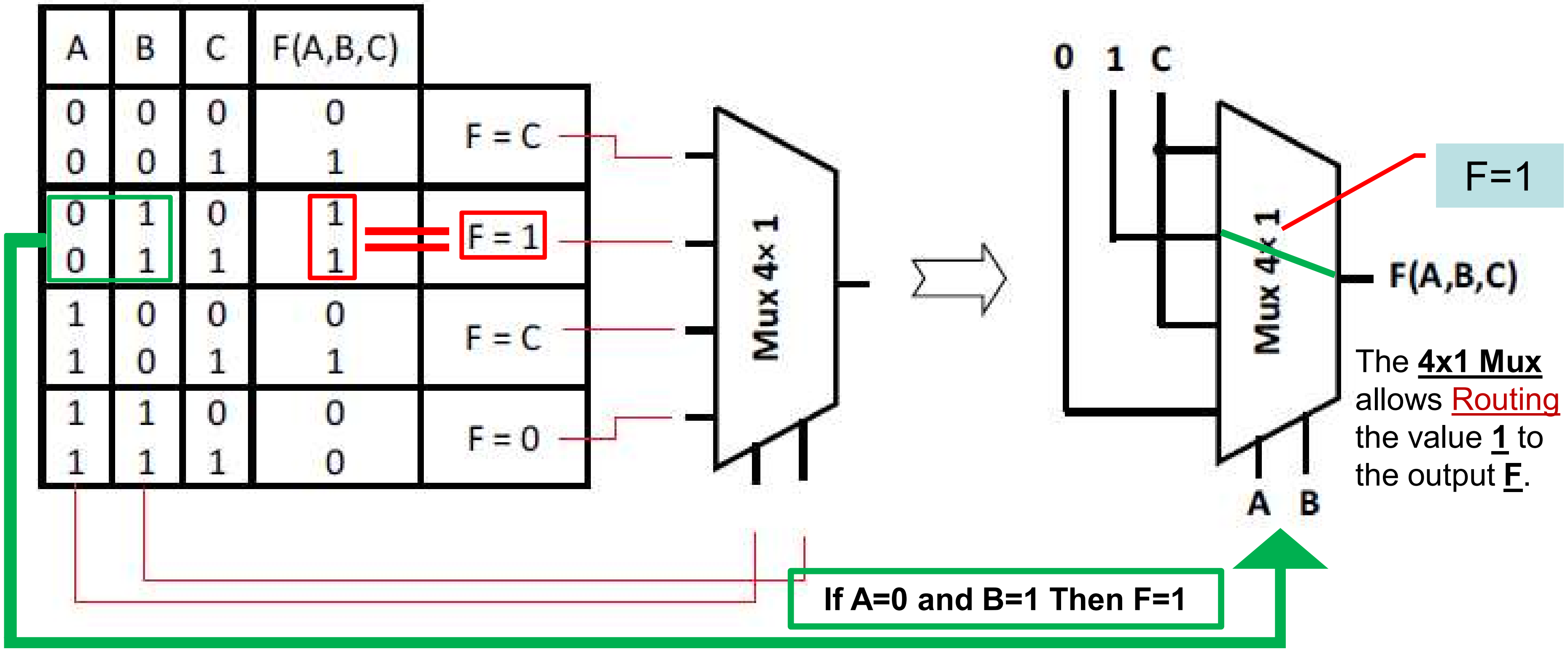
$$F(A,B,C) = \bar{A}.B + \bar{B}.C$$

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

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

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

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



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

If A=0 and B=1 Then F=1

The **4x1 Mux** allows Routing the value **1** to the output **F**.

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

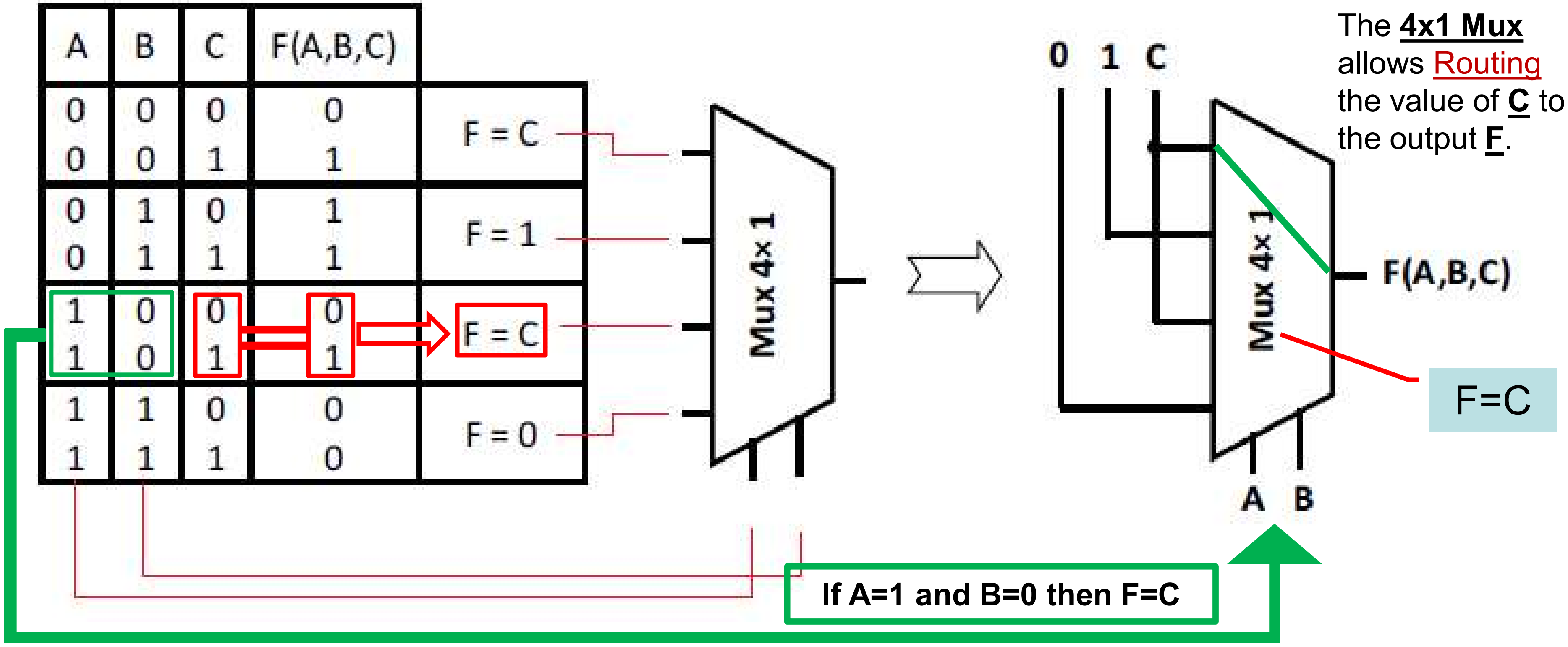
$$F(A,B,C) = \bar{A}.B + \bar{B}.C$$

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

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

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

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



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

The **4x1 Mux** allows Routing the value of **C** to the output **F**.

If A=1 and B=0 then F=C

Logical functions using multiplexers:

Example 1: Implement the Function using a 4x1 multiplexer.

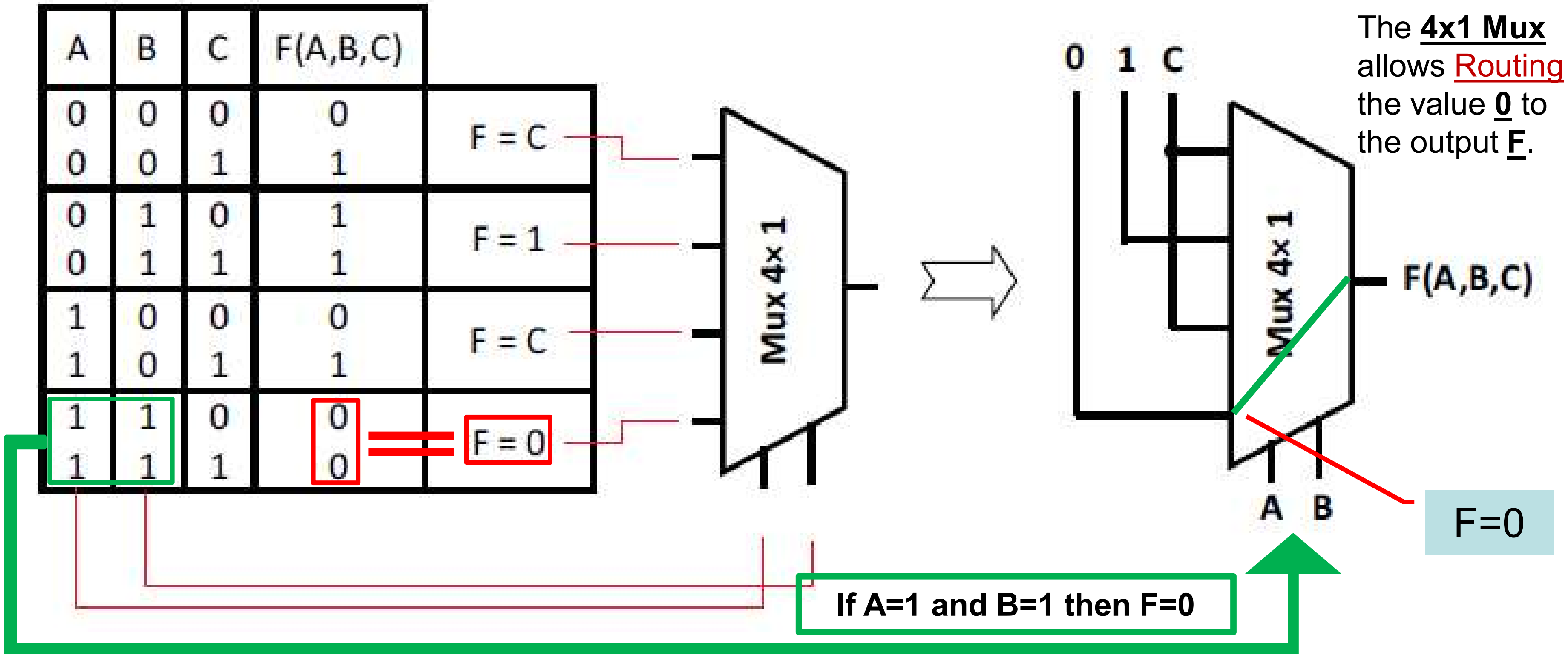
$$F(A,B,C) = \bar{A}.B + \bar{B}.C$$

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

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

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

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



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

The **4x1 Mux** allows Routing the value **0** to the output **F**.

If A=1 and B=1 then F=0

Example: Implementation of a full adder using 8x1 multiplexers

- We need to use two multiplexers: the first one to implement the sum function and the other to provide the carry.

Example: Implementation of a full adder using 8x1 multiplexers

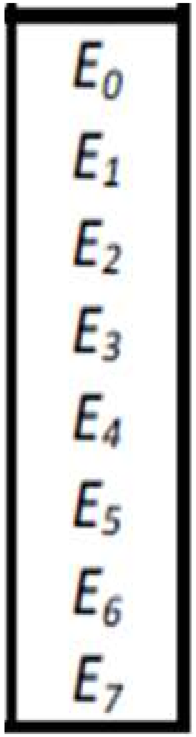
- We need to use two multiplexers: the first one to implement the sum function and the other to provide the carry.

a_i	b_i	r_{i-1}		S_i
0	0	0		0
0	0	1		1
0	1	0		1
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		0
1	1	1		1

Example: Implementation of a full adder using 8x1 multiplexers

- We need to use two multiplexers: the first one to implement the sum function and the other to provide the carry.

a_i	b_i	r_{i-1}		S_i
0	0	0		0
0	0	1		1
0	1	0		1
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		0
1	1	1		1



E_0

E_1

E_2

E_3

E_4

E_5

E_6

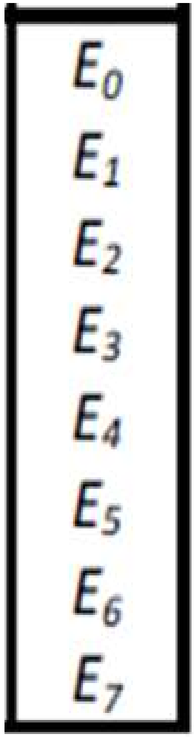
E_7

Example: Implementation of a full adder using 8x1 multiplexers

- We need to use two multiplexers: the first one to implement the sum function and the other to provide the carry.

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

a_i	b_i	r_{i-1}		S_i
0	0	0		0
0	0	1		1
0	1	0		1
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		0
1	1	1		1



E_0

E_1

E_2

E_3

E_4

E_5

E_6

E_7

Example: Implementation of a full adder using 8x1 multiplexers

- We need to use two multiplexers: the first one to implement the sum function and the other to provide the carry.

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

E_0
E_1
E_2
E_3
E_4
E_5
E_6
E_7

a_i	b_i	r_{i-1}		S_i
0	0	0		0
0	0	1		1
0	1	0		1
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		0
1	1	1		1

E_0
E_1
E_2
E_3
E_4
E_5
E_6
E_7

Implementation of the function for: the sum

$$S_i = \overline{A}_i.\overline{B}_i.\overline{R}_{i-1}(0) + \overline{A}_i.\overline{B}_i.R_{i-1}(1) + \overline{A}_i.B_i.\overline{R}_{i-1}(1) + \overline{A}_i.B_i.R_{i-1}(0) +$$

$$A_i.\overline{B}_i.\overline{R}_{i-1}(1) + A_i.\overline{B}_i.R_{i-1}(0) + A_i.B_i.\overline{R}_{i-1}(0) + A_i.B_i.R_{i-1}(1)$$

We set :

$C2=A_i$

$C1=B_i$

$C0=R_{i-1}$

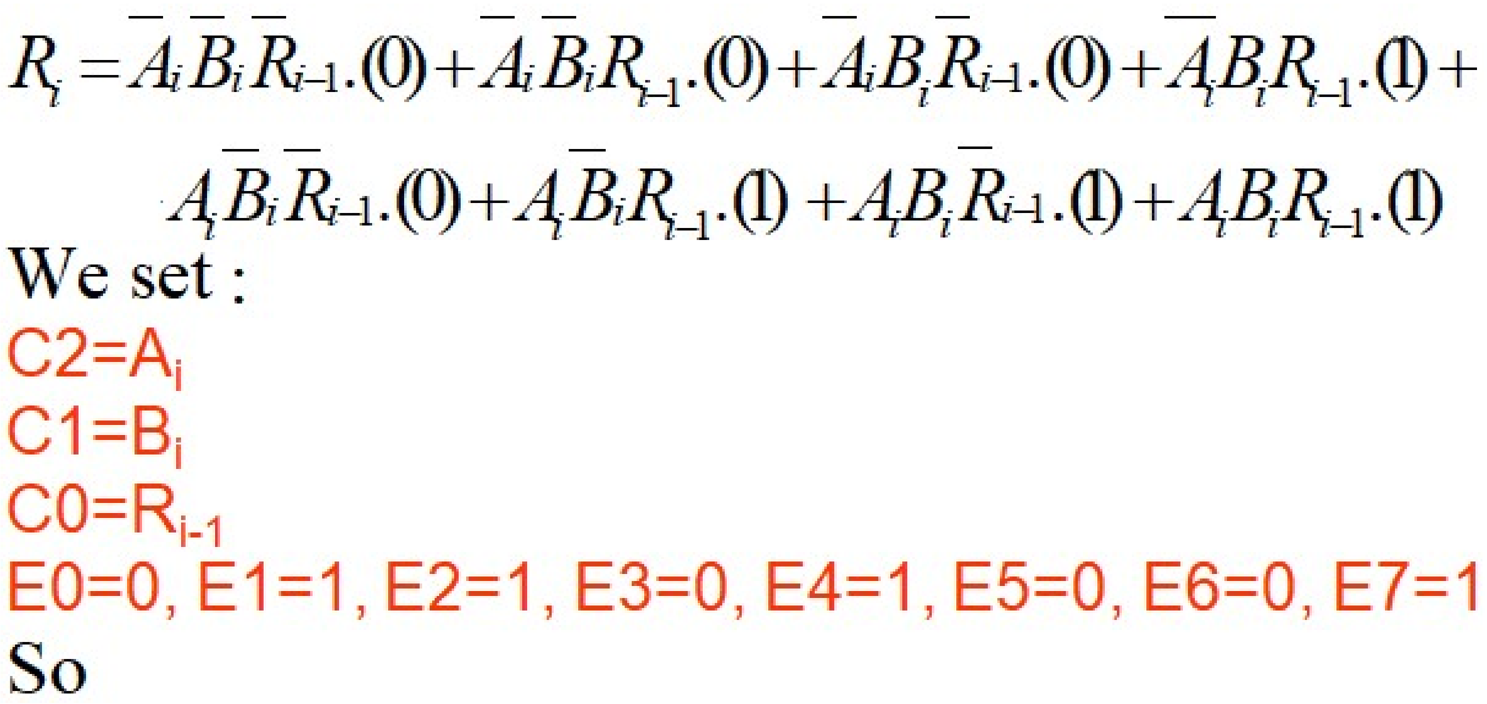
E0=0, E1=1, E2=1, E3=0, E4=1, E5=0, E6=0, E7=1

So

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

$$C2.\overline{C1}.\overline{C0}(E4) + C2.\overline{C1}.C0(E5) + C2.C1.\overline{C0}(E6) + C2.C1.C0(E7)$$

Implementation of the function for: the carry



$$R_i = \bar{A}_i \bar{B}_i \bar{R}_{i-1}.(0) + \bar{A}_i \bar{B}_i R_{i-1}.(0) + \bar{A}_i B_i \bar{R}_{i-1}.(0) + \bar{A}_i B_i R_{i-1}.(1) +$$

$$A_i \bar{B}_i \bar{R}_{i-1}.(0) + A_i \bar{B}_i R_{i-1}.(1) + A_i B_i \bar{R}_{i-1}.(1) + A_i B_i R_{i-1}.(1)$$

We set :

$C2=A_i$

$C1=B_i$

$C0=R_{i-1}$

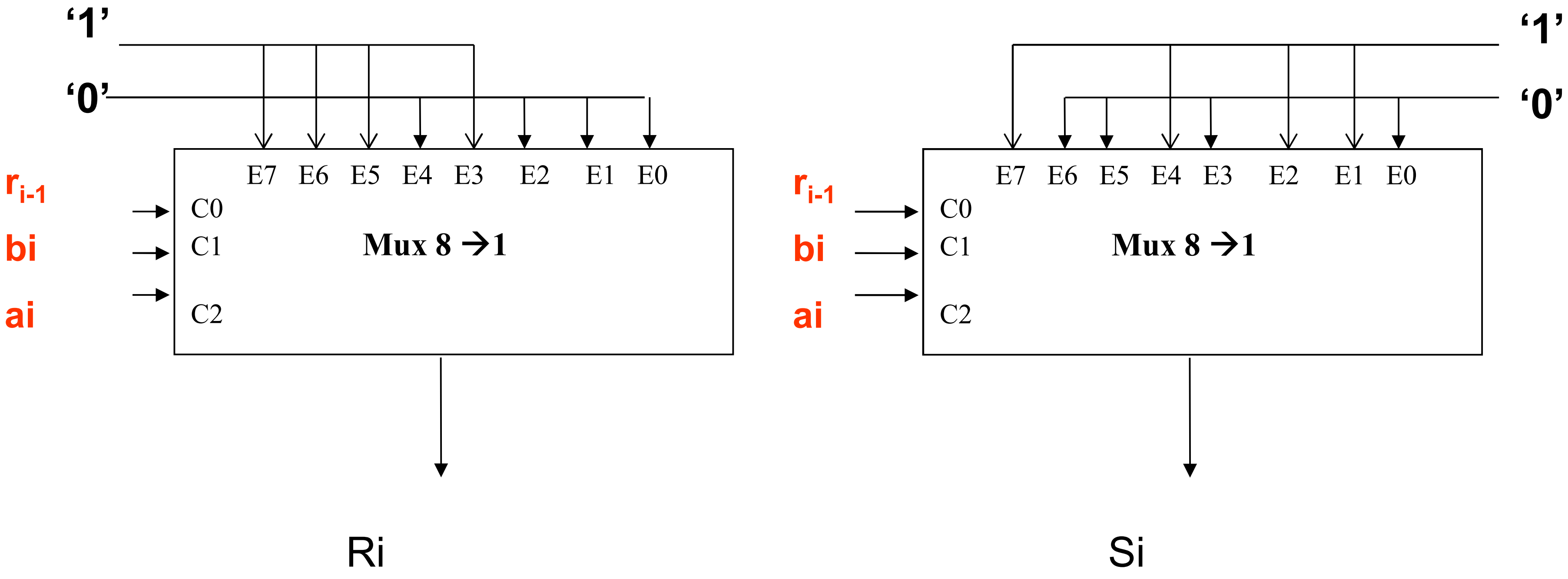
E0=0, E1=1, E2=1, E3=0, E4=1, E5=0, E6=0, E7=1

So

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

$$C2.\overline{C1}.\overline{C0}(E4) + C2.\overline{C1}.C0(E5) + C2.C1.\overline{C0}(E6) + C2.C1.C0(E7)$$

Implementation of a full adder using 8x1 multiplexers



Exercice

- Design the circuit that allows finding the maximum between two numbers A and B on one bit using the minimum number of logic gates and combinational circuits?