Structure Machine 2

Chapitre 01
Les circuits combinatoires

Introduction (Rappel):

Pour comprendre le fonctionnement des principaux éléments d'un ordinateur Comme l'unité logique et arithmétique UAL.

Vous devrez avoir atteint les objectifs suivants :

1- Décrire le fonctionnement et les propriétés des portes logiques, de circuits combinatoires simples tels que l'additionneur, le décodeur, le multiplexeur et le démultiplexeur…;

2- Utiliser les théorèmes et les identités de l'algèbre de Boole pour synthétiser un circuit à partir de sa table de vérité et simplifier le résultat obtenu.

Rappel: Notion de circuit logique

- Fonctions logiques
- Une fonction logique est une fonction qui agit sur une ou plusieurs variables logiques.
- Une variable logique est une variable qui peut prendre l'une de deux valeurs : vrai ou faux, 1 ou 0.
- Les circuits logiques sont des circuits électroniques servant à effectuer physiquement des fonctions logiques.

Objectifs

- **Apprendre la structure de quelques circuits combinatoires souvent utilisés (demi additionneur , additionneur complet,……..).**

- **Apprendre comment utiliser des circuits combinatoires pour concevoir d'autres circuits plus complexes.**

1. Les Circuits combinatoires

- **Un circuit combinatoire est un circuit numérique dont les sorties dépendent uniquement des entrées.**
- **$S_i=F(E_i)$**
- **$S_i=F(E_1,E_2,\ldots,E_n)$**

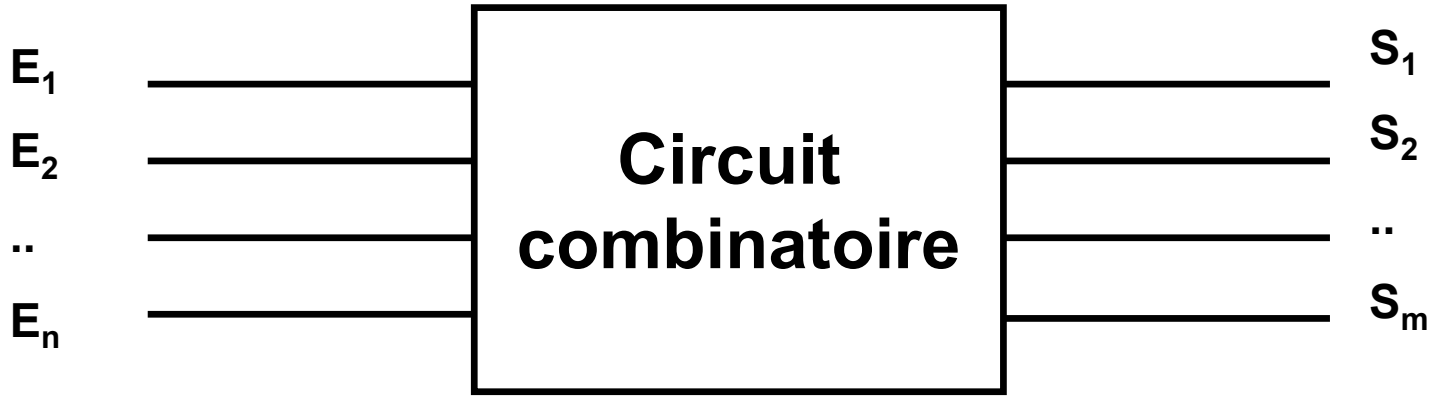


Schéma Bloc

- **C'est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits plus complexes.**

Exemple de Circuits combinatoires

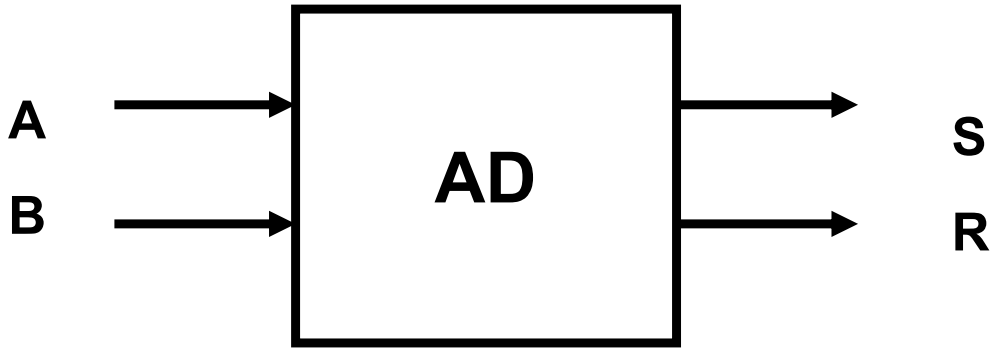
1. Demi Additionneur
2. Additionneur complet
3. Comparateur
4. Multiplexeur
5. Démultiplexeur
6. Encodeur
7. Décodeur

Exemple de Circuits combinatoires

- Dans un ordinateur, nous pouvons distinguer trois classes différentes de circuits logiques combinatoires.
- 1- Les circuits combinatoires de calcul arithmétiques et logiques, tels que les additionneurs, les soustracteurs, les comparateurs, etc.
- 2- Les circuits combinatoires d'aiguillage et de transmission de données, tels que les codeurs, les décodeurs, les multiplexeurs, les démultiplexeurs, etc.
- 3- Les circuits combinatoires de codage et de conversion de codes, tels que les transcodeurs, les afficheurs 7 segments, etc.

2. Demi Additionneur

- **Le demi additionneur est un circuit combinatoire qui permet de réaliser la somme arithmétique de deux nombres A et B chacun sur un bit.**
- **A la sotie on va avoir la Somme S et la Retenu R (Carry).**

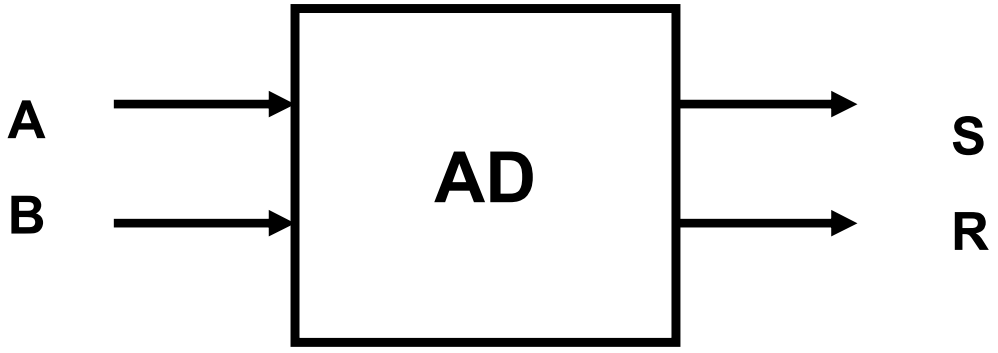


Pour trouver la structure (le schéma) de ce circuit on doit en premier dresser sa table de vérité

- **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 :**



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

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

- **La table de vérité associée :**

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

De la table de vérité on trouve :

$$R =$$

$$S =$$

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

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

- **La table de vérité associée :**

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

De la table de vérité on trouve :

$$R =$$

$$S =$$

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

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

- **La table de vérité associée :**

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

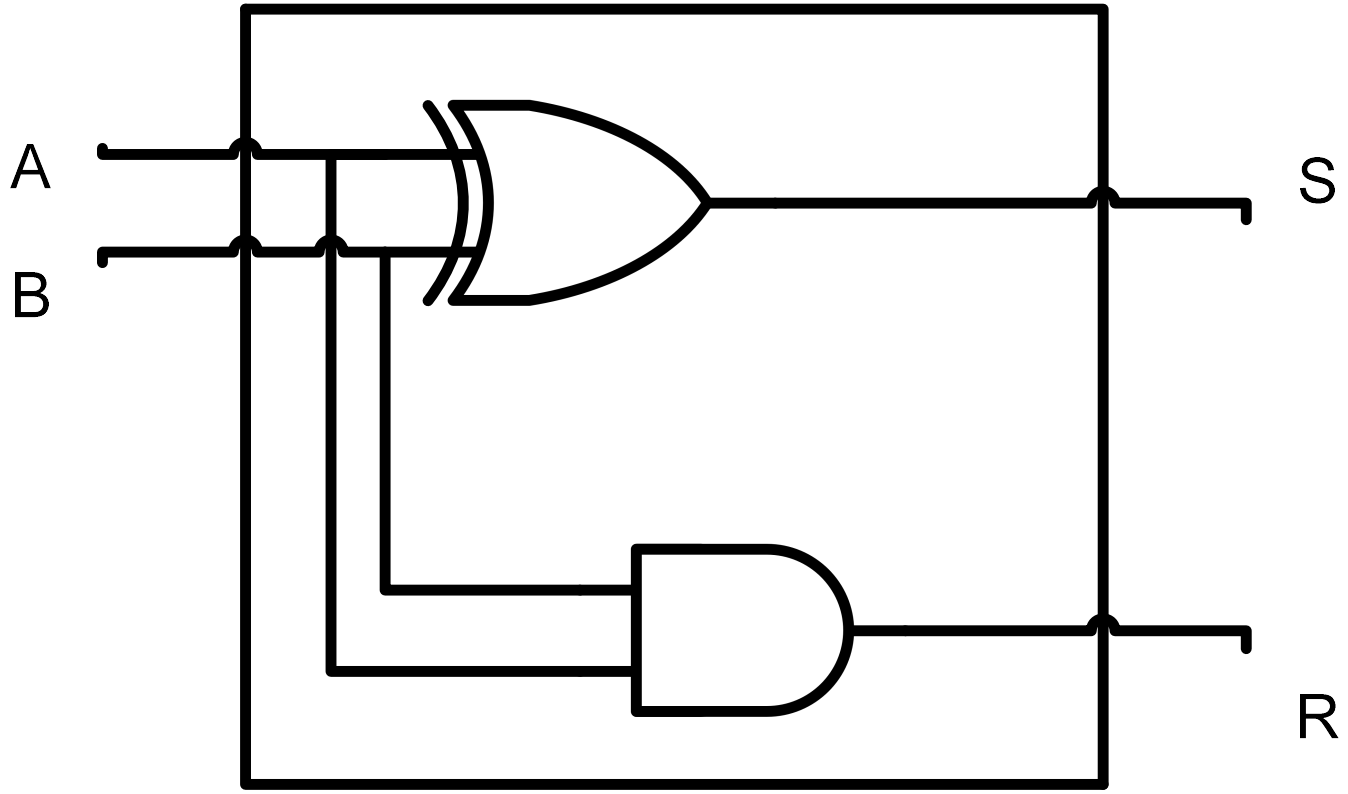
De la table de vérité on trouve :

$$R = A.B$$

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

$$R = A.B$$

$$S = A \oplus B$$



3. L'additionneur complet

- **En binaire lorsque on fait une addition il faut tenir en compte de la retenue entrante.**

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 Additionneur complet 1 bit

- L'additionneur complet un bit possède 3 entrées :
  - a_i : le premier nombre sur un bit.
  - b_i : le deuxième nombre sur un bit.
  - r_{i-1} : le retenue entrante sur un bit.
- Il possède deux sorties :
  - S_i : la somme
  - R_i la retenue sortante

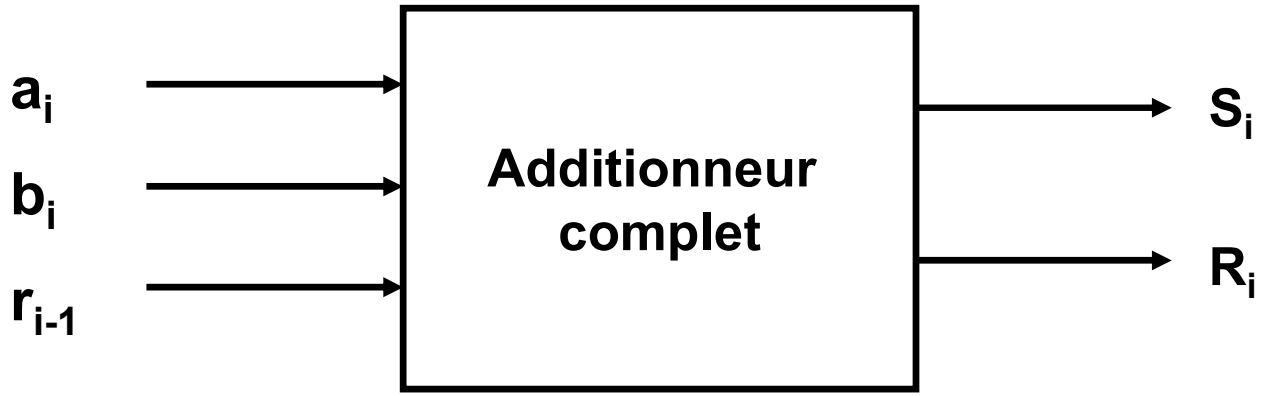
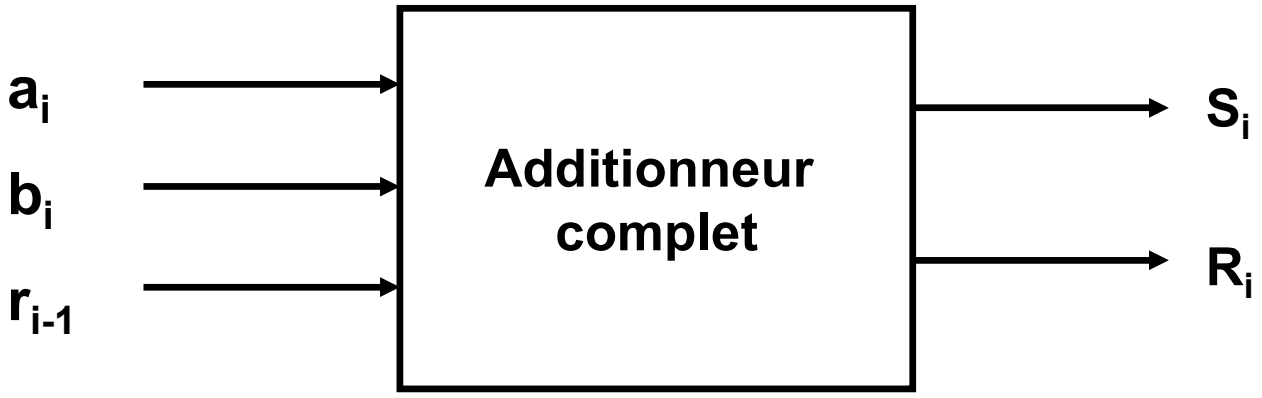


Table de vérité d'un additionneur complet sur 1 bit



Dresser sa table de vérité :

Table de vérité d'un additionneur complet sur 1 bit

- possède 3 entrées :
  - a_i : le premier nombre.
  - b_i : le deuxième nombre.
  - r_{i-1} : le retenue entrante.
- Il possède deux sorties :
  - S_i : la somme
  - R_i la retenue sortante

a_i	b_i	r_{i-1}		r_i	s_i

Table de vérité d'un additionneur complet sur 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			

$$S_i =$$

$$R_i =$$

Table de vérité d'un additionneur complet sur 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			

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

$$S_i =$$

$$R_i =$$

Table de vérité d'un additionneur complet sur 1 bit

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

$$S_i =$$

$$R_i =$$

Table de vérité d'un additionneur complet sur 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 retenue 0
0+1 = 1 + 0 = 1 retenue 0
1 + 1 = 0 retenue 1
1+1+1 = 1 retenue 1

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

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

Si on veut simplifier les équations on obtient :

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

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

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

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

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

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

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

Si on veut simplifier les équations on obtient :

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

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

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

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

$= A_i \oplus X$

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

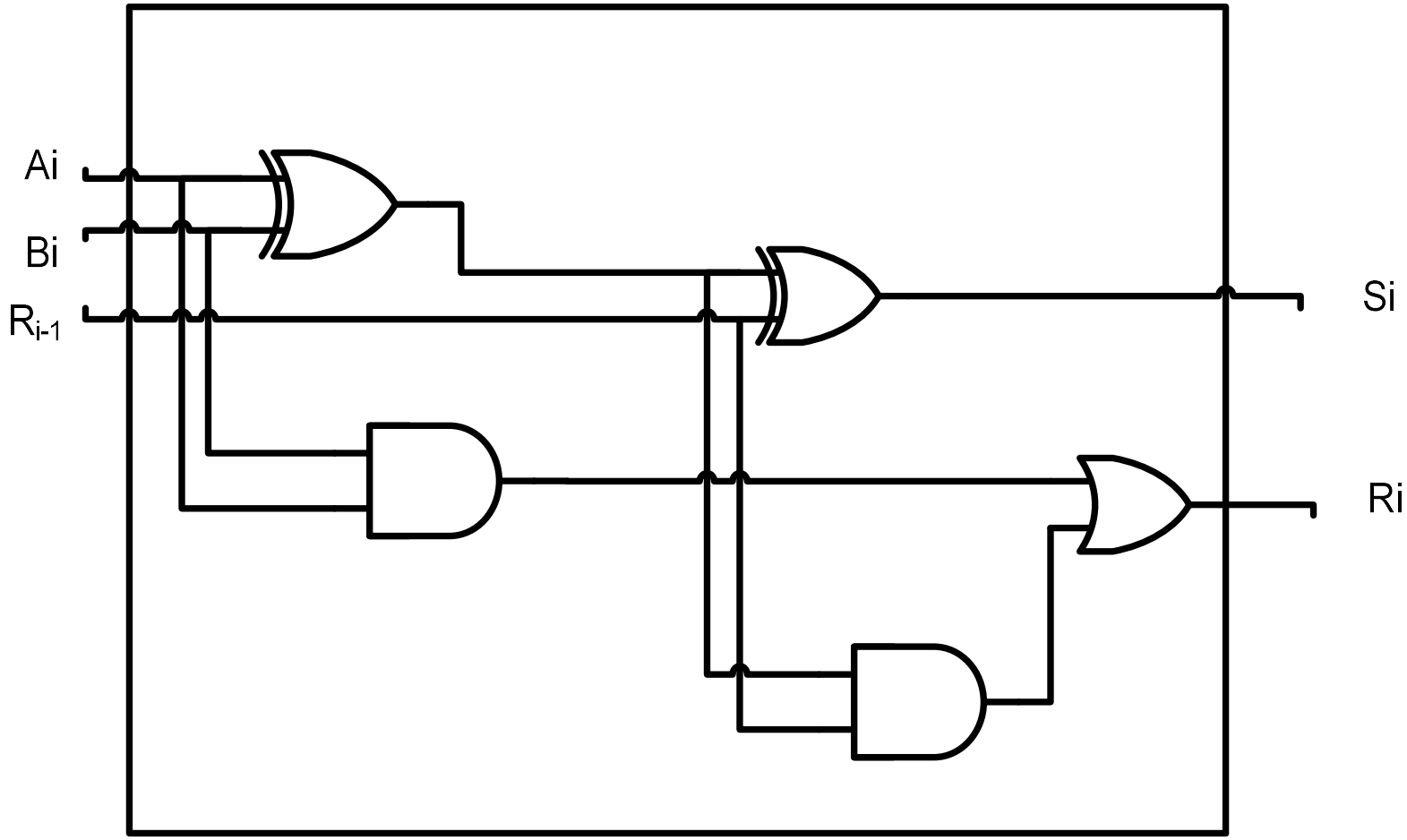
$$R_i = R_{i-1}.(\overline{A_i}.B_i + A_i.\overline{B}_i) + A_iB_i(\overline{R}_{i-1} + {}_i\ R_{i-1})$$

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

3.2 Schéma d'un additionneur complet

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

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



3.3 En utilisant des Demi Additionneurs

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

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

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

On obtient :

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

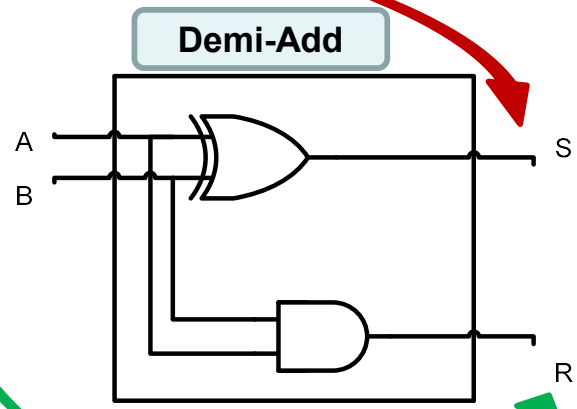
$$S_i = X \oplus R_{i-1}$$

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

On obtient :

$$R_i = Y + T$$

$$S_i = Z$$

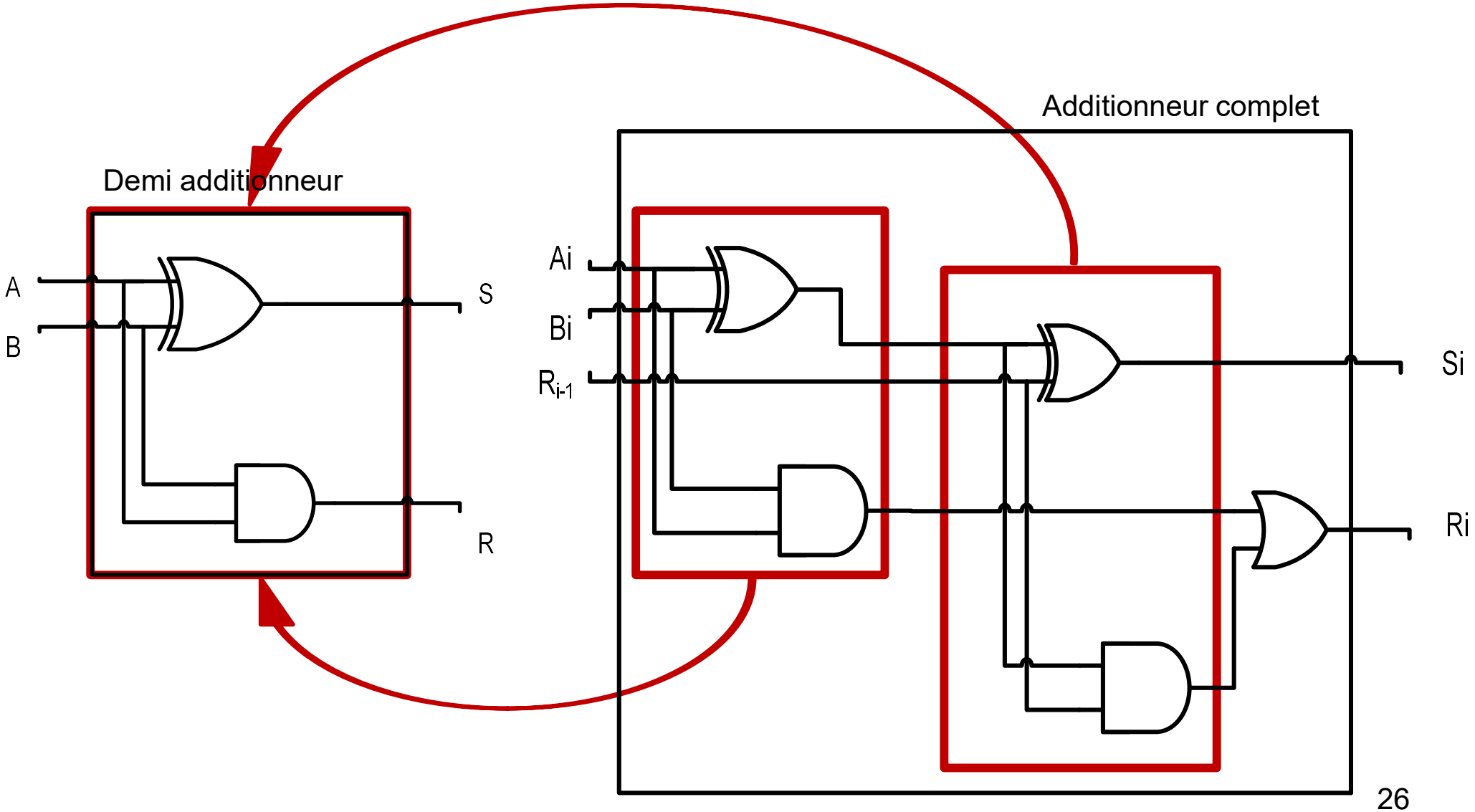


$$R = A.B$$

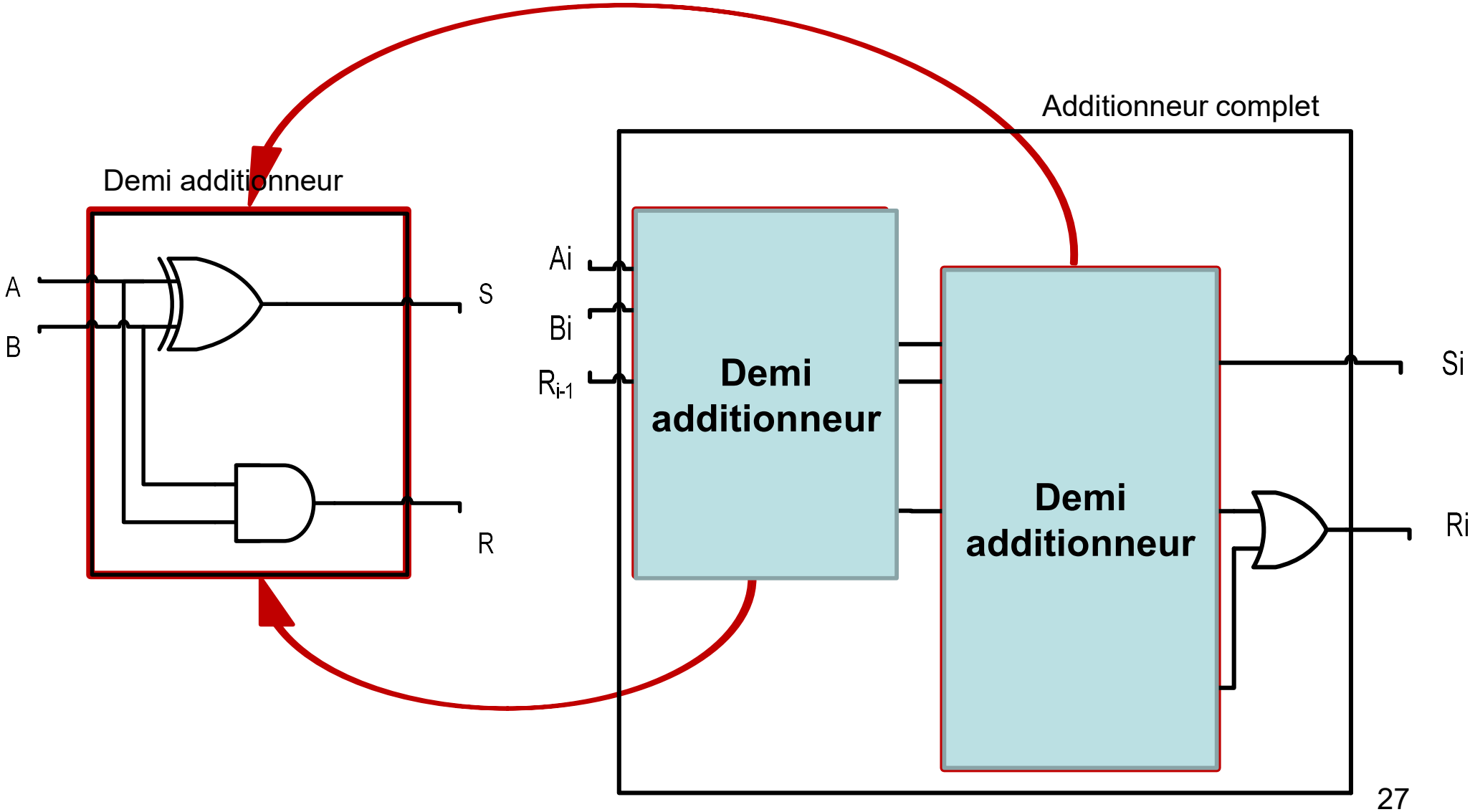
$$S = A \oplus B$$

- **On remarque que X et Y sont les sorties d'un demi additionneur ayant comme entrées A et B**
- **On remarque que Z et T sont les sorties d'un demi additionneur ayant comme entrées X et R_{i-1}**

3.2 Comparaison d'un demi additionneur avec additionneur complet



3.2 Comparaison d'un demi additionneur avec additionneur complet



$$X = A_i \oplus B_i$$

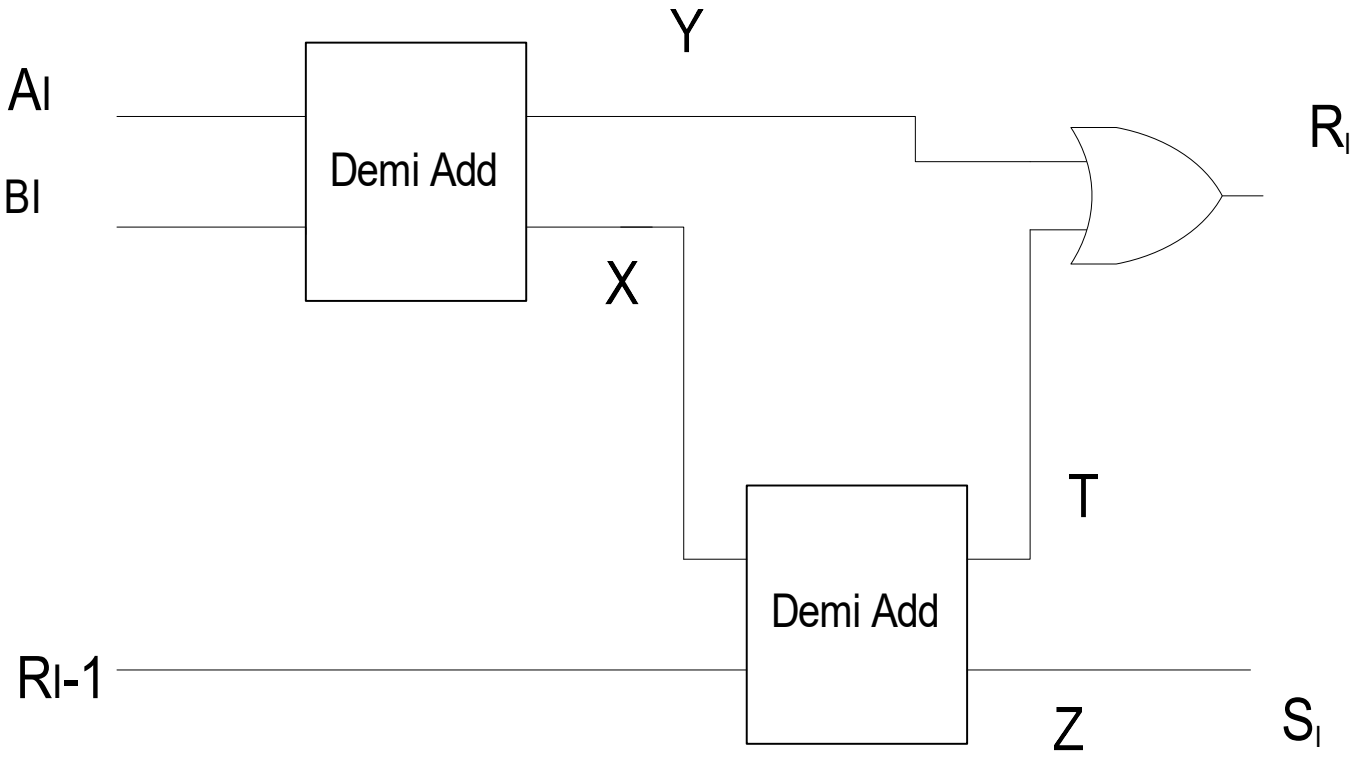
$$Y = A_i B_i$$

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

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

$$R_i = Y + T$$

$$S_i = Z$$

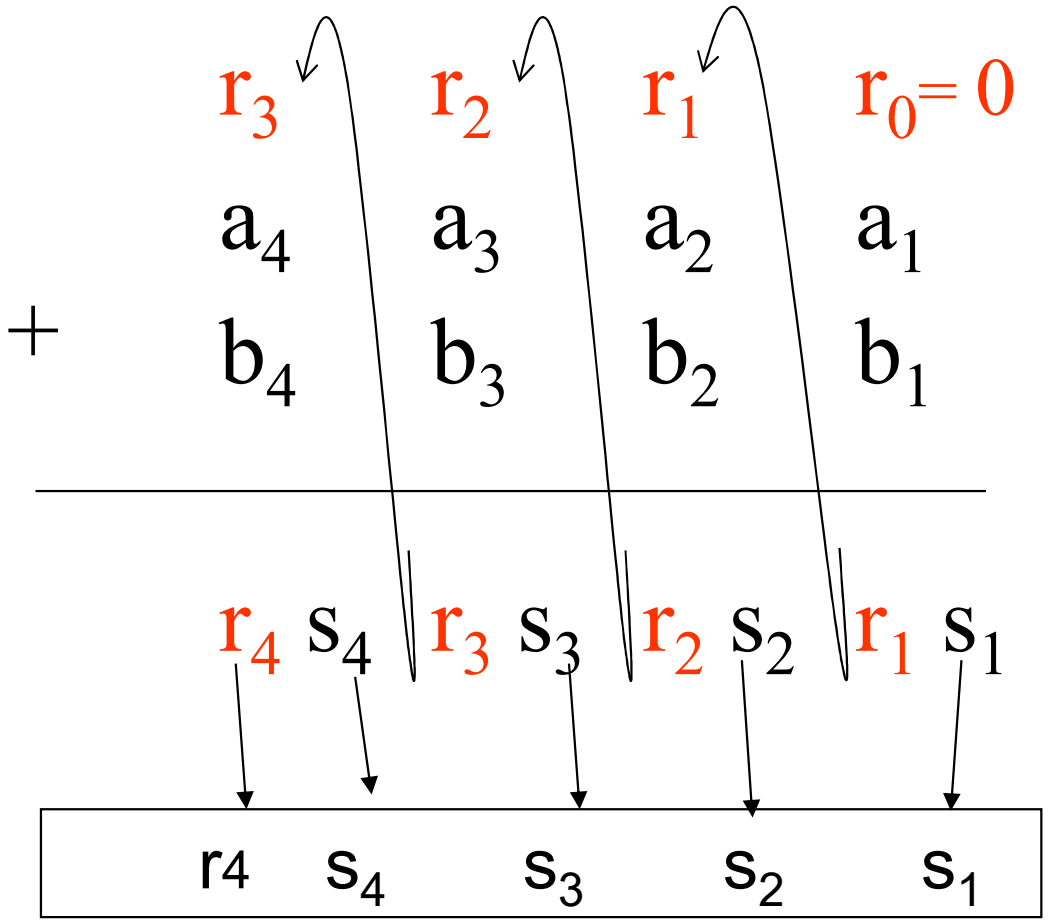


3.4 Additionneur sur 4 bits

- **Un additionneur sur 4 bits est un circuit qui permet de faire l'addition de deux nombres A et B de 4 bits chacun**
  - **$A(a_3a_2a_1a_0)$**
  - **$B(b_3b_2b_1b_0)$**

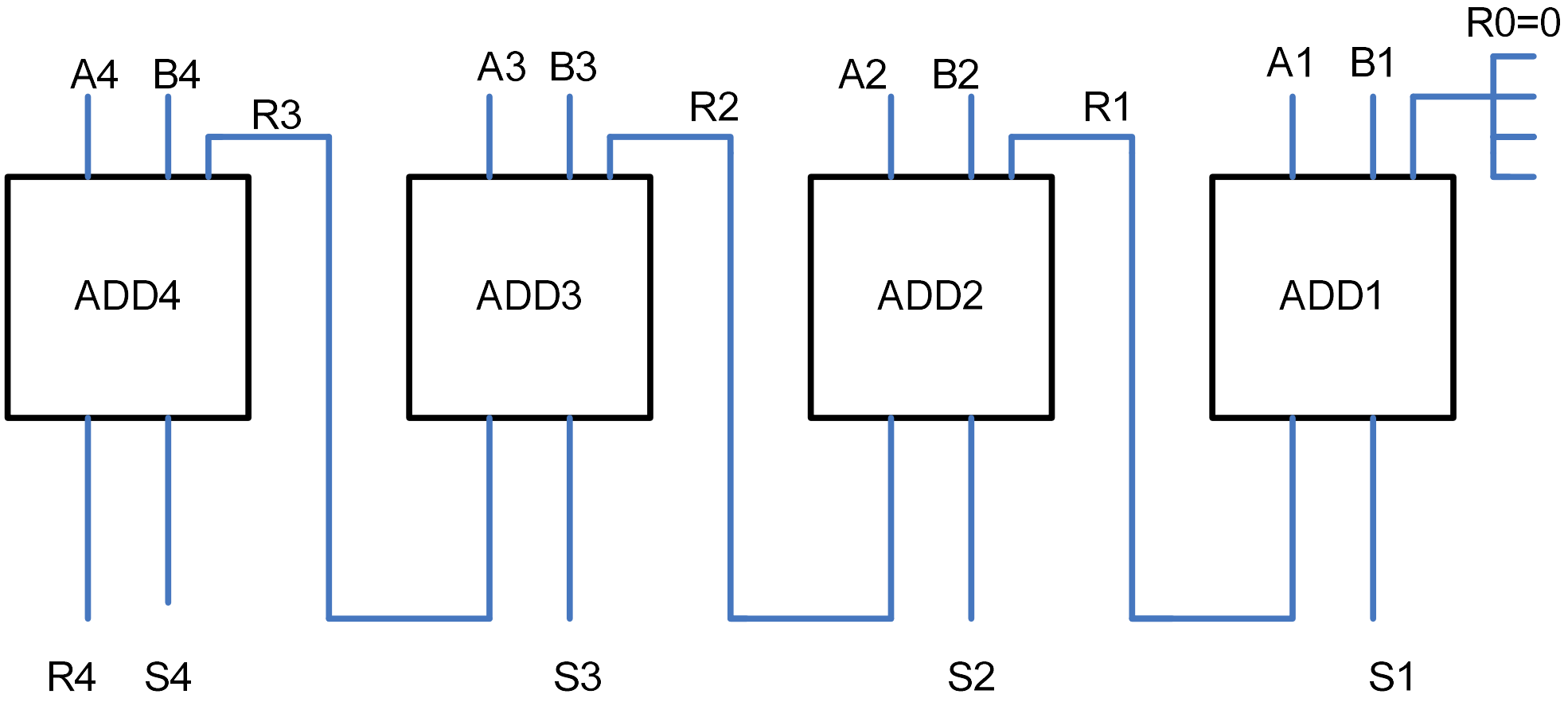
  En plus il tient en compte de la retenu entrante
- **En sortie on va avoir le résultat sur 4 bits ainsi que la retenu (5 bits en sortie).Donc, au total le circuit possède 9 entrées et 5 sorties.**
- **Avec 9 entrées on a 2^9=512 combinaisons !!!!!! Comment faire pour représenter la table de vérité ?????**
- **Il faut trouver une solution plus facile et plus efficace pour concevoir ce circuit ?**

•**Lorsque on fait l'addition en binaire , on additionne bit par bit en commençant à partir du poids fiable et à chaque fois on propage la retenue sortante au bit du rang supérieur. L'addition sur un bit peut se faire par un additionneur complet sur 1 bits.**



Résultat final

3.4.1 Additionneur 4 bits (schéma)



Exercice 1

- **Un générateur de parité impaire est une fonction qui retourne 1 si le nombre de bits à 1 est impair et 0 sinon.**

- **Définir cette fonction pour un mot de 4 bits. Donner un circuit logique implémentant cette fonction.**

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**: La formule pour le générateur de parité impaire sur 4 bits (P) obtenue directement à partir de la table de vérité est :

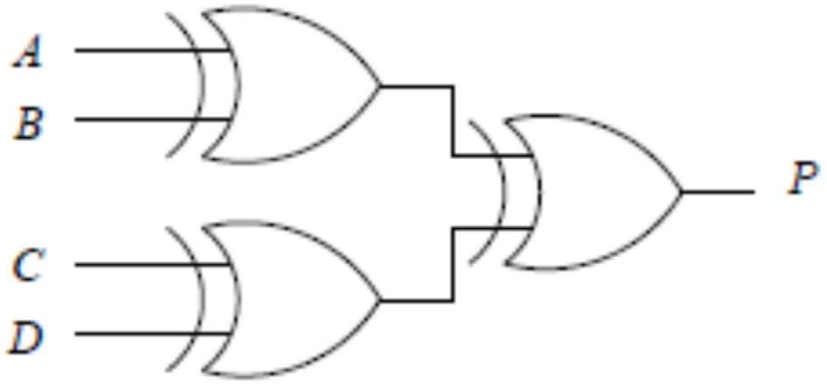
$$P = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}$$
$$+ \cdots$$
$$+ \overline{A} \cdot B \cdot C \cdot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

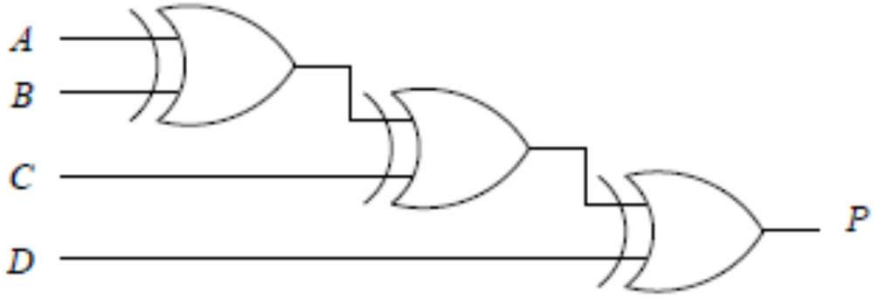
- ce qui donnerait un circuit beaucoup trop compliqué !
- On remarque que pour deux bits, P = A⊕B :

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

Exercice 1

- On en déduit les circuits suivants :





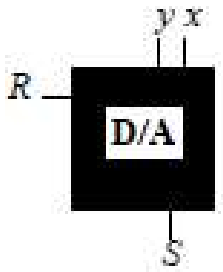
Exercice 2

- Rappeler les principes d'un demi-additionneur puis d'un additionneur complet.
- Déduire de ces principes un circuit logique qui implémente le complément à 2 sur *n* bits.

Correction : *Le demi-additionneur possède deux entrées (x et y) et deux sorties (R et S).*

S correspond au bit de rang zéro du résultat de l'addition binaire de x et y, R au bit de rang 1 *(retenue)*.

Exercice

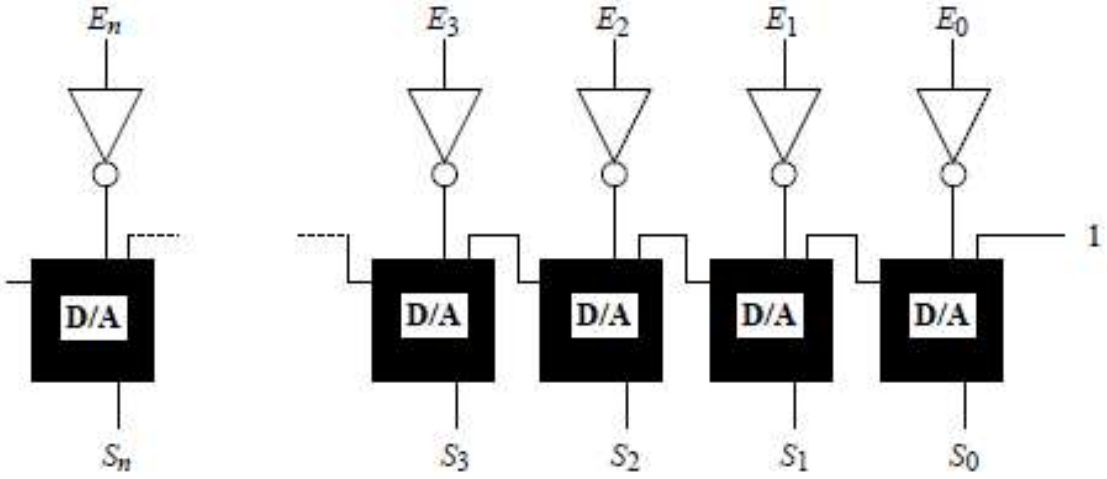


x	y	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = x \oplus y$$

$$R = x \cdot y$$

- *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 :*

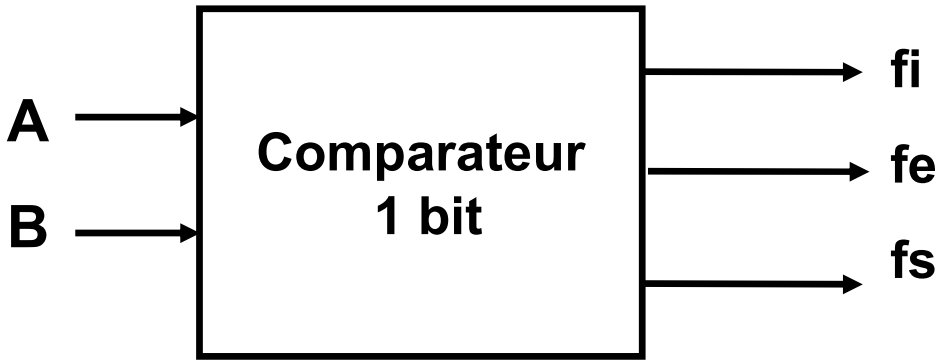


4. Le Comparateur

- Un comparateur est un circuit arithmétique permettant de comparer deux nombres binaires A et B. Les nombres A et B doivent avoir la même longueur (nombre de bits). On cherche à savoir si A > B, A<B ou A=B. On comprend donc que le circuit répond à une question à trois choix. Notre circuit final doit produire trois signaux f**S**(actif si A>B), f**i**(actif si A<B) et f**e**(actif si A=B) en prenant en entrée les signaux A et B.

4. Le Comparateur

- C'est un circuit combinatoire qui permet de comparer entre deux nombres binaire A et B.
- Il possède 2 entrées :
  - A : sur un bit
  - B : sur un bit
- Il possède 3 sorties
  - fe : égalité (A=B)
  - fi : inférieur (A < B)
  - fs : supérieur (A > B)



4. Le Comparateur

Table de vérité d'un comparateur sur 1 bit

Dresser sa table de vérité :

4.1 Comparateur sur un bit

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

$$fs =$$

$$fi =$$

$$fe =$$

4.1 Comparateur sur un bit

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

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

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

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

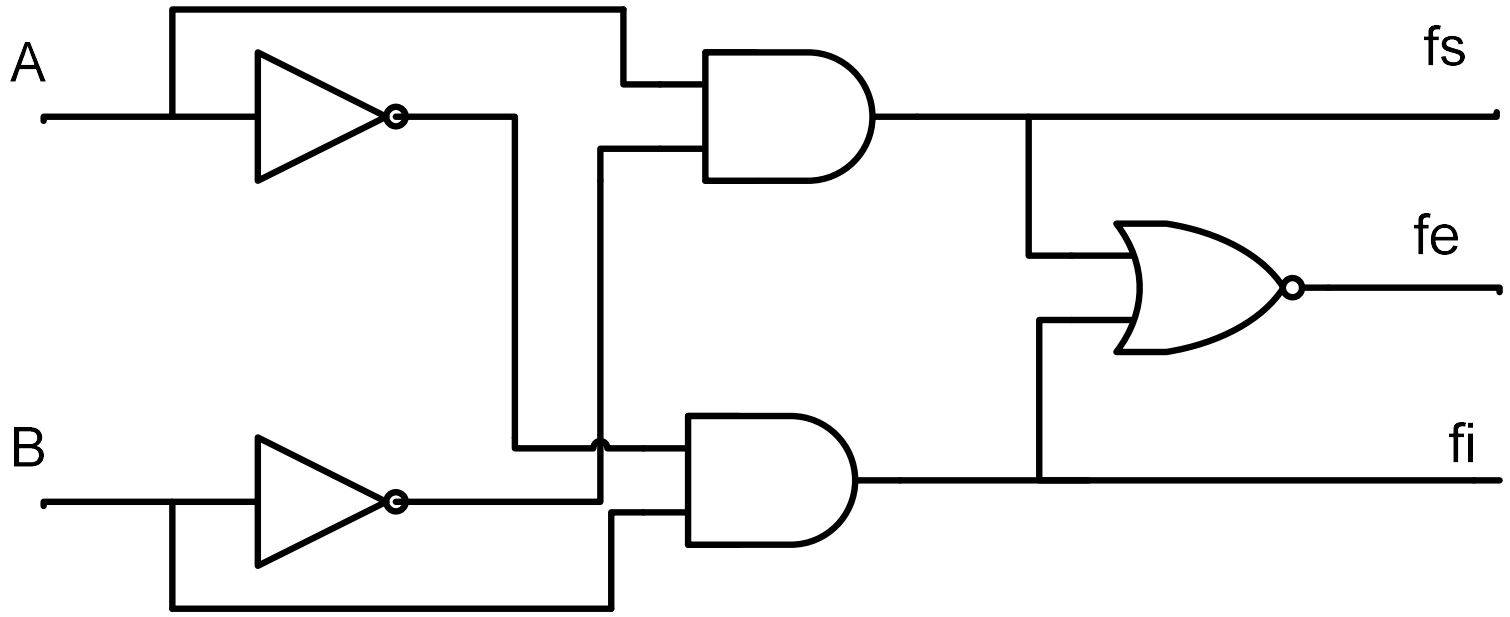
car: $fs + fi = A.\overline{B} + \overline{A}B = A \oplus B$

Schéma d'un comparateur sur un bit

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

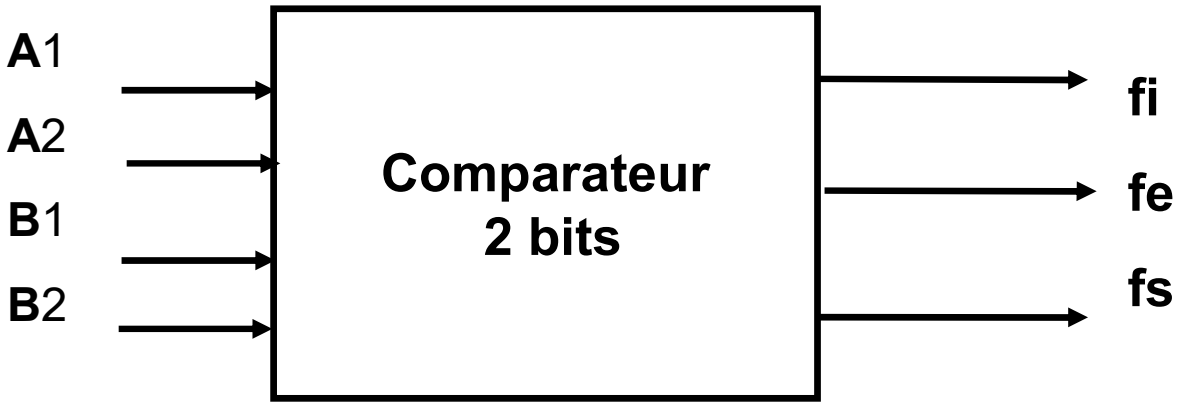
$$fi = \overline{A}B$$

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



4.2 Comparateur 2 bits

- **Il permet de faire la comparaison entre deux nombres A (a_2a_1) et B(b_2b_1) chacun sur deux bits.**



Comparateur un bit

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

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

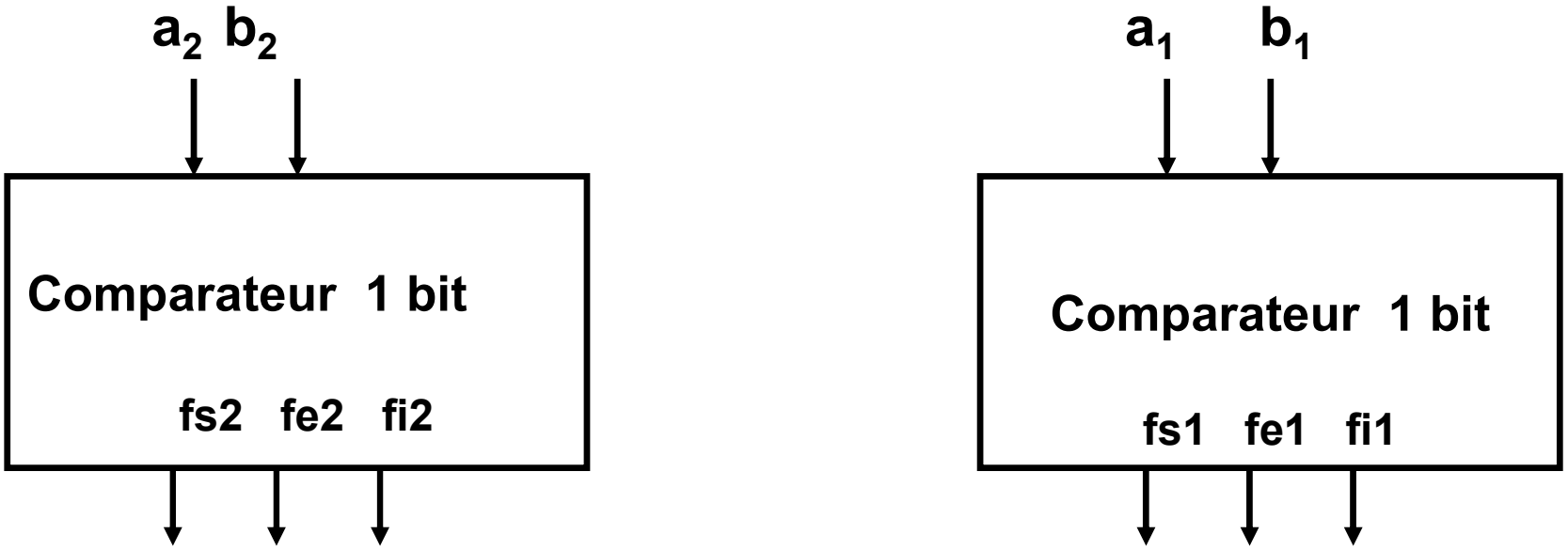
A2 < B2 ou (A2=B2 et A1<B1)

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

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

4.2.2 comparateur 2 bits avec des comparateurs 1 bit

- **C'est possible de réaliser un comparateur 2 bits en utilisant des comparateurs 1 bit et des portes logiques.**
- **Il faut utiliser un comparateur pour comparer les bits du poids faible et un autre pour comparer les bits du poids fort.**
- **Il faut combiner entre les sorties des deux comparateurs utilisés pour réaliser les sorties du comparateur final.**



1. A=B si

A2=B2 et A1=B1

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

2. A>B si

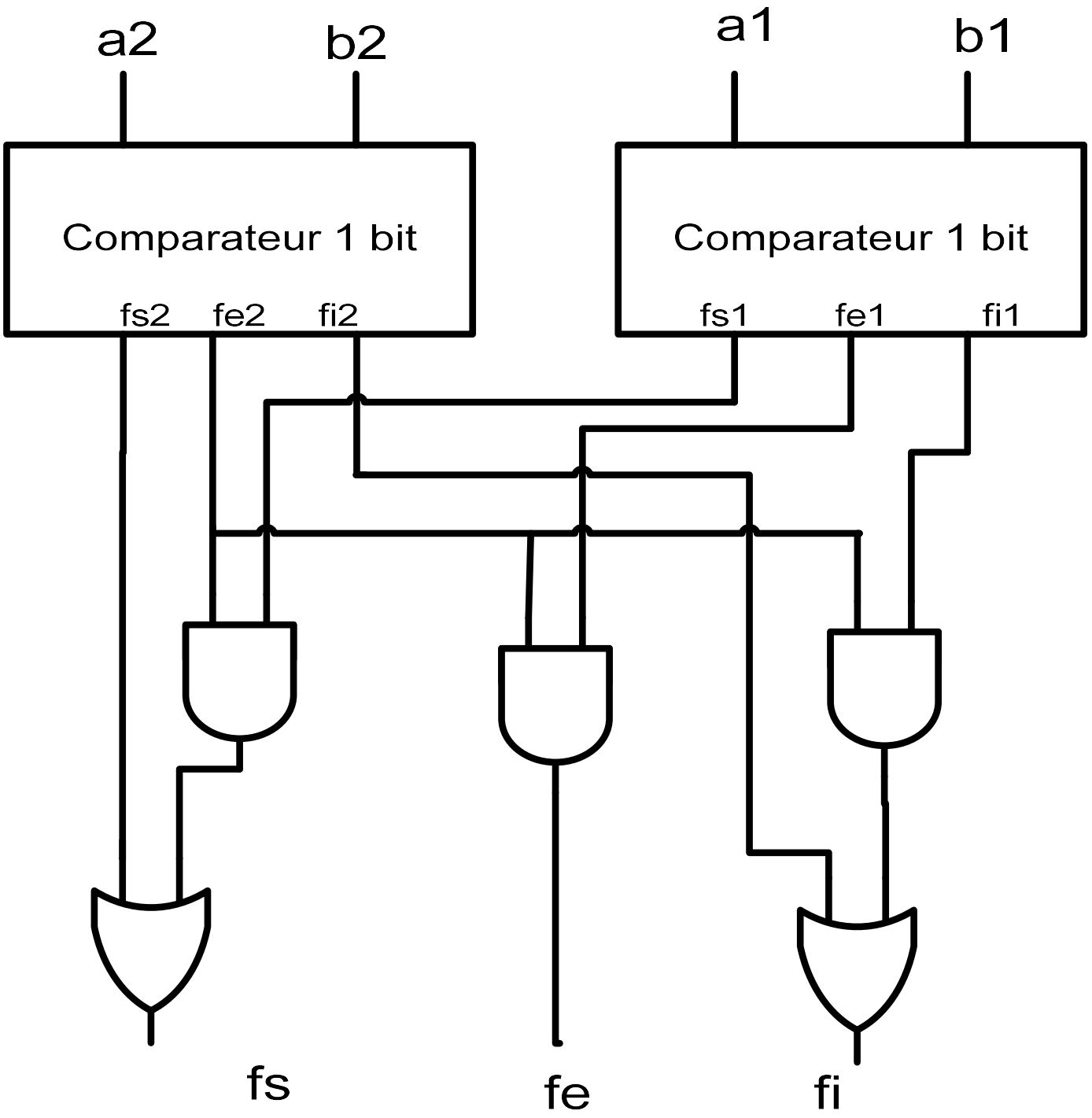
A2 > B2 ou (A2=B2 et A1>B1)

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

3. A<B si

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

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


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

4.2.3 Comparateur avec des entrées de mise en cascade

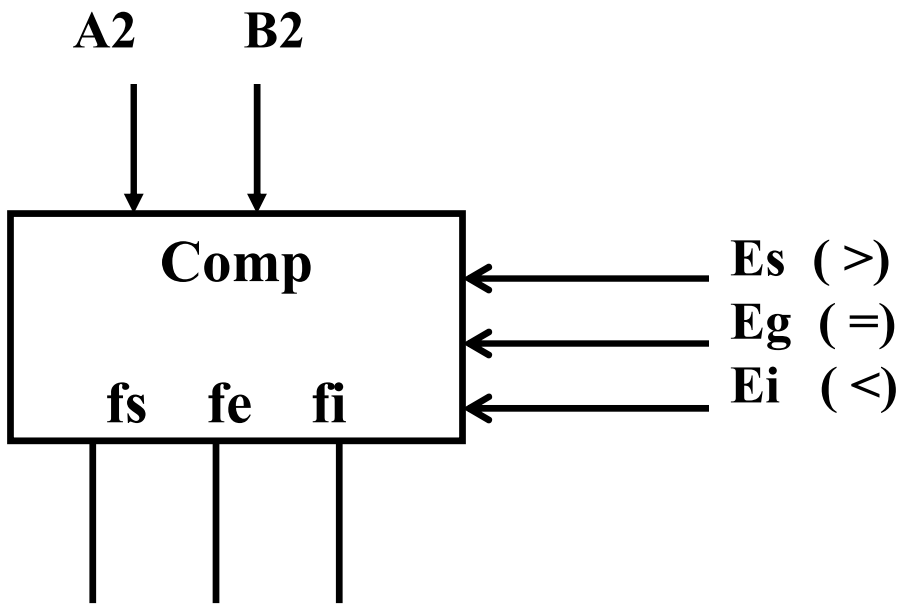
- **On remarque que :**
  - **Si A2 >B2 alors A > B**
  - **Si A2<B2 alors A < B**

- **Par contre si A2=B2 alors il faut tenir en compte du résultat de la comparaison des bits du poids faible.**

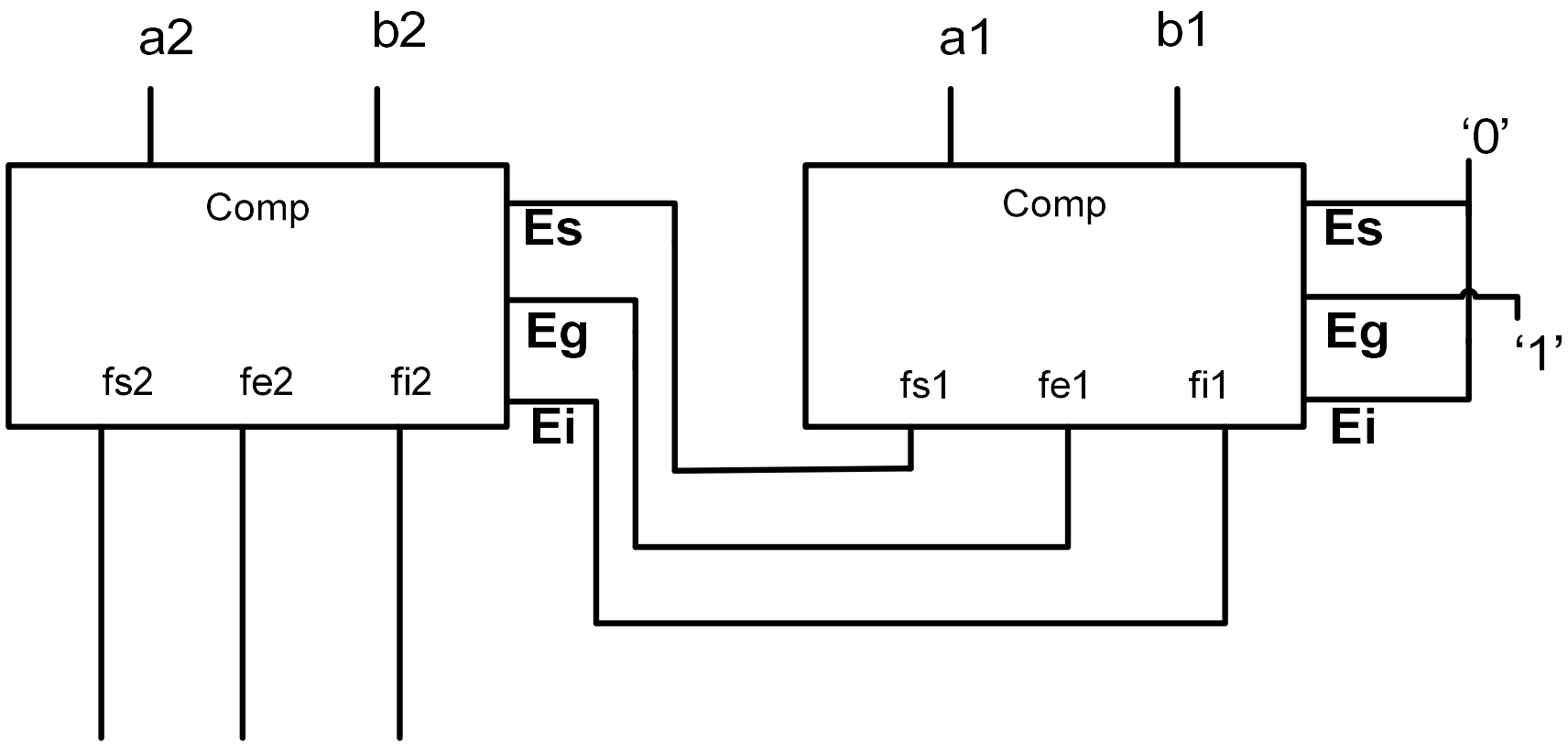
- **Pour cela on rajoute au comparateur des entrées qui nous indiquent le résultat de la comparaison précédente.**

- **Ces entrées sont appelées des entrées de mise en cascade.**

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

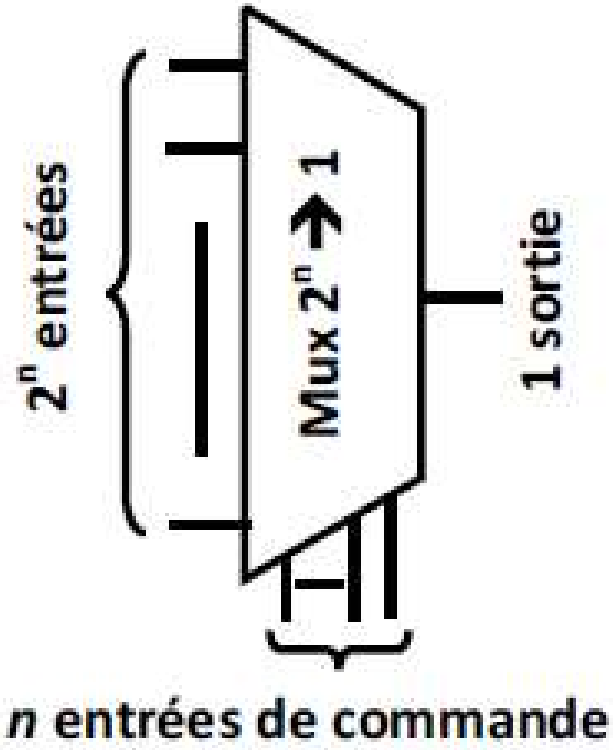

a2
b2
a1
b1
'0'
Comp
Comp
Es
Eg
Ei
Es
Eg
Ei
'1'
fs2
fe2
fi2
fs1
fe1
fi1

Exercice

- **Réaliser un comparateur 4 bits en utilisant des comparateurs 2 bits avec des entrées de mise en cascade?**

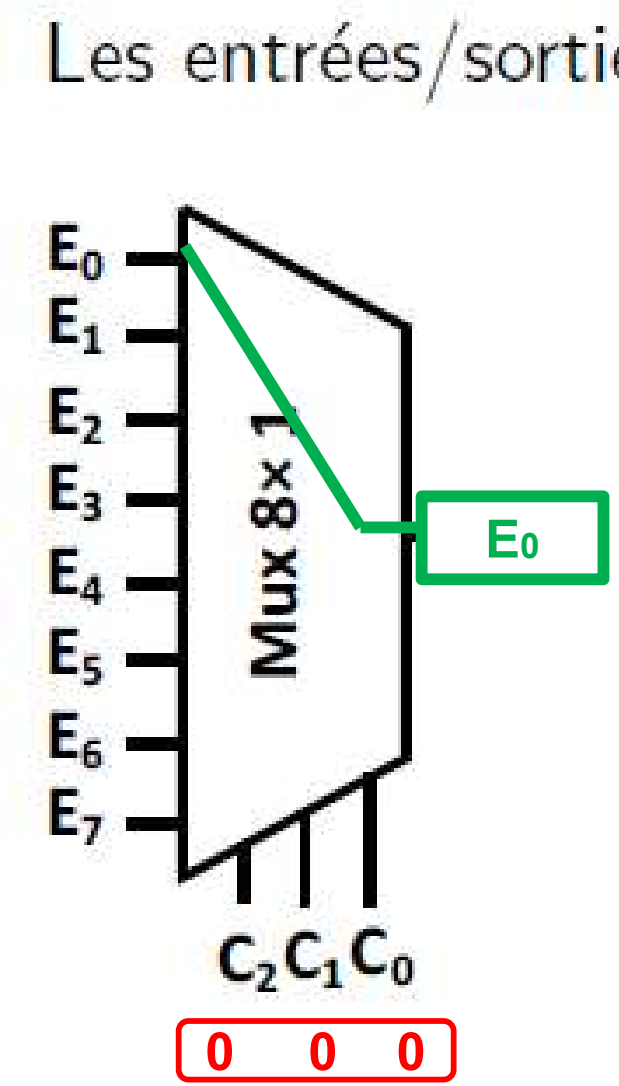
5. Le Multiplexeur

- **Définition :** Un multiplexeur est un circuit logique combinatoire qui comporte 2^n entrées,
- n entrées de commande et une seule sortie.
- Il permet d'aiguiller la valeur de la ligne d'entrée indiquée dans ses entrées de commande vers la ligne de sortie.



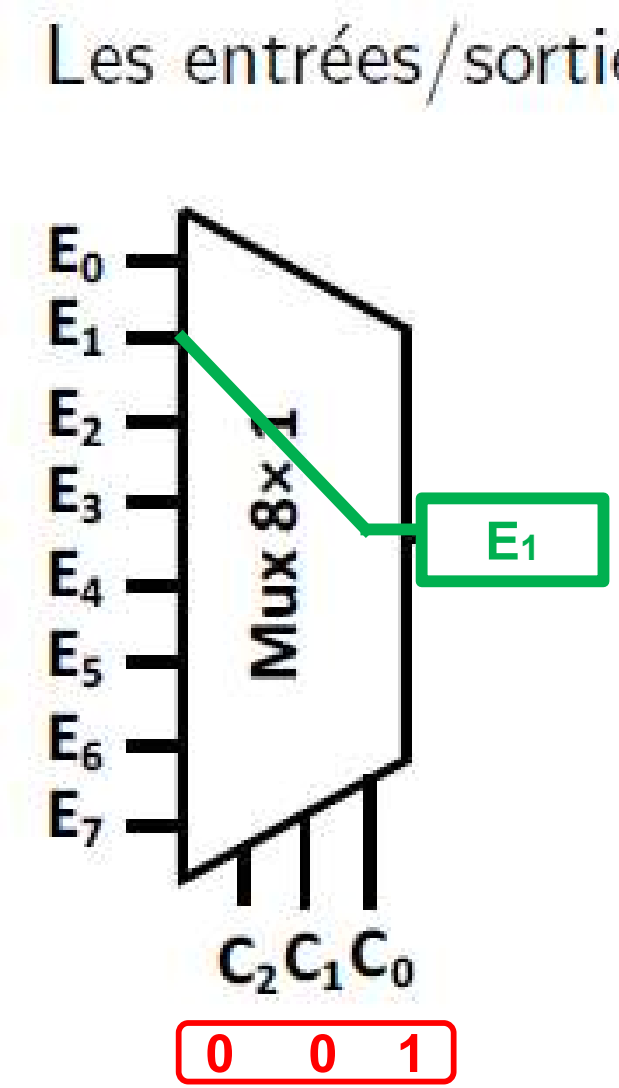
5. Le Multiplexeur

- Synthèse du circuit (multiplexeur 8x1) exemple:



5. Le Multiplexeur

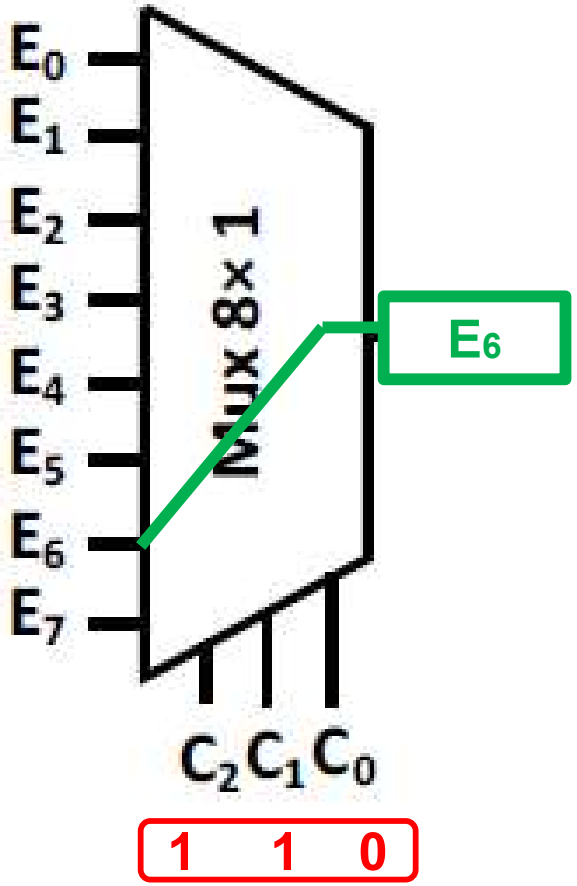
- Synthèse du circuit (multiplexeur 8x1) exemple:



5. Le Multiplexeur

- Synthèse du circuit (multiplexeur 8x1) exemple:

Les entrées/sorties.



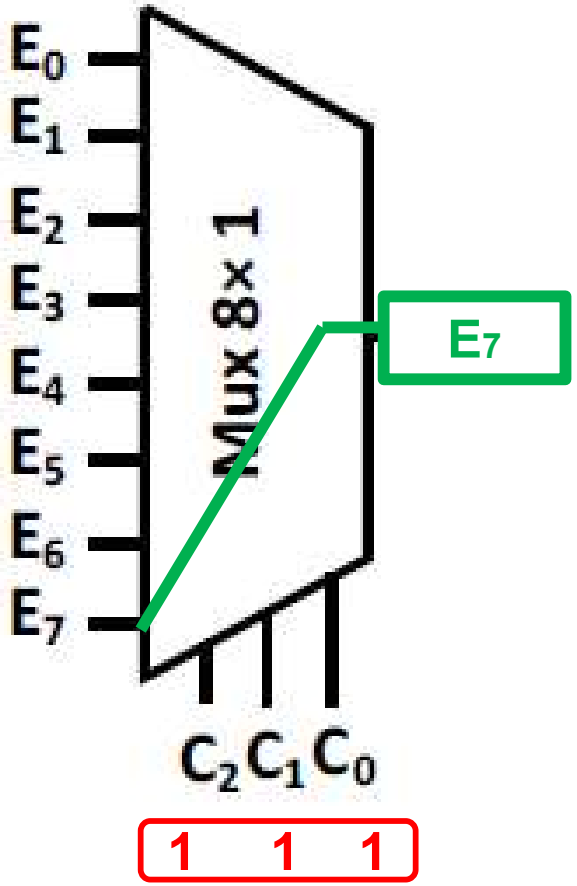
La table de vérité.

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. Le Multiplexeur

- Synthèse du circuit (multiplexeur 8x1) exemple:

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. Le Multiplexeur

- Synthèse du circuit (multiplexeur 8x1) exemple:

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.

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

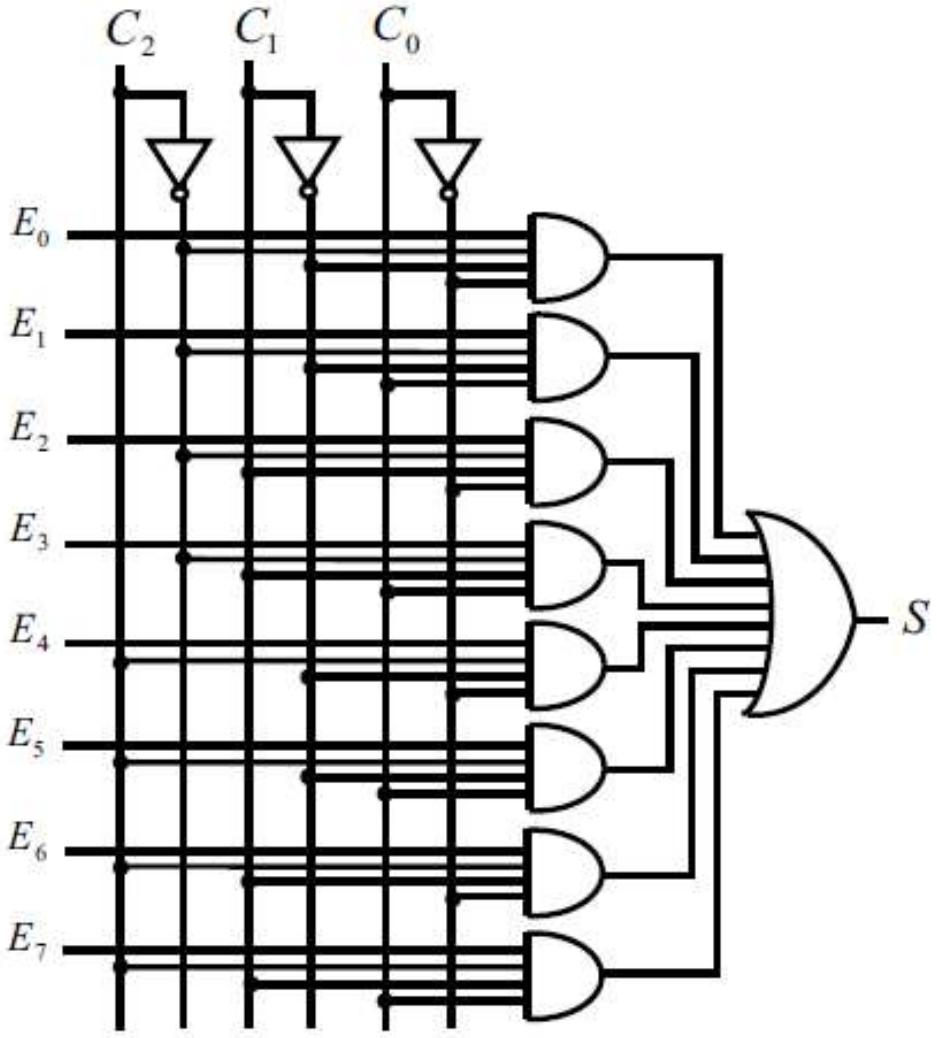
5. Le Multiplexeur

- Synthèse du circuit (multiplexeur 8x1) exemple:

Les fonctions logiques.

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

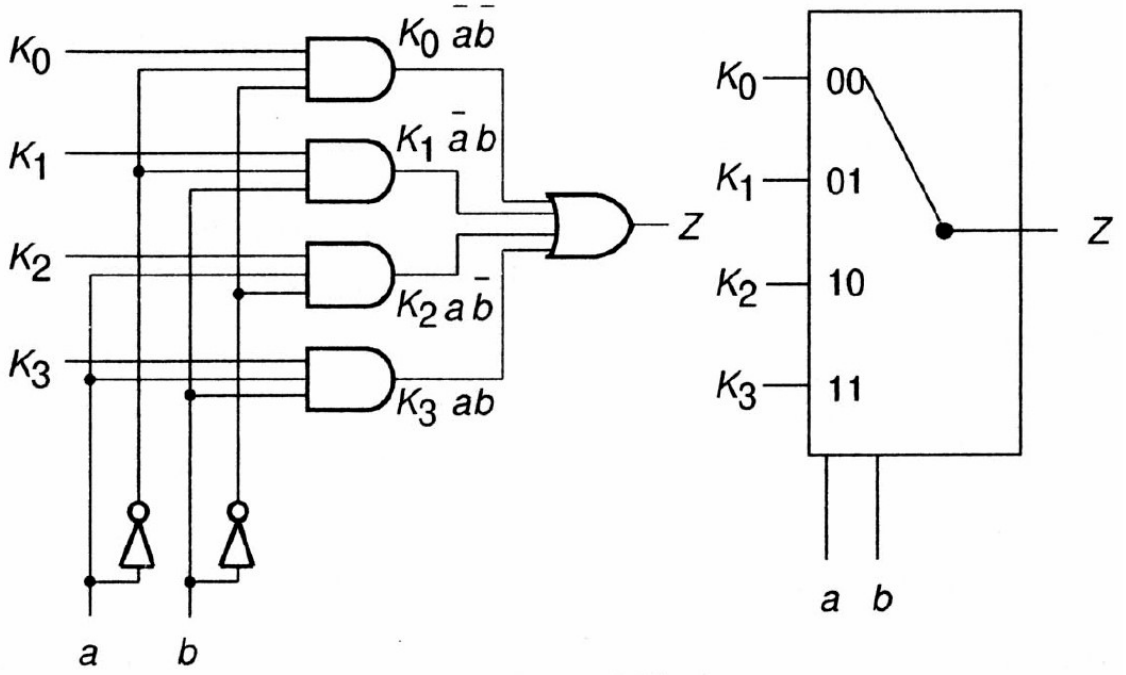
Le schéma du circuit.



5. Le Multiplexeur

- Le multiplexeur est un circuit combinatoire Sélecteur qui possède 2^n entrées d'information, n entrées de commande et une seule sortie. Son rôle consiste à sélectionner, à l'aide de signaux de commande, une des entrées et à la lier à la sortie.

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



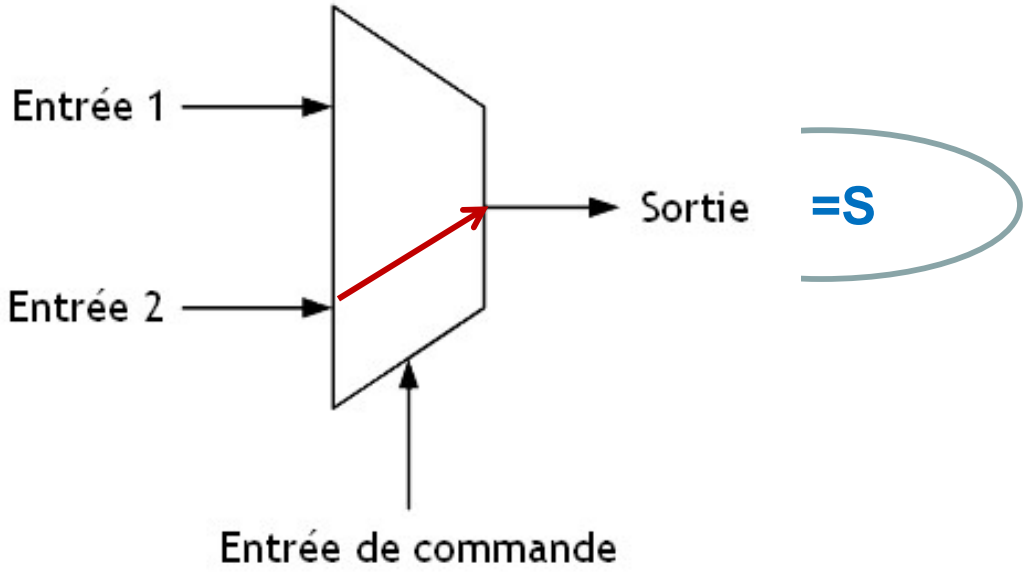
MUX (2 variables)

5.1 Multiplexeur 2 →1

Description du comportement du multiplexeur 2 à 1 :

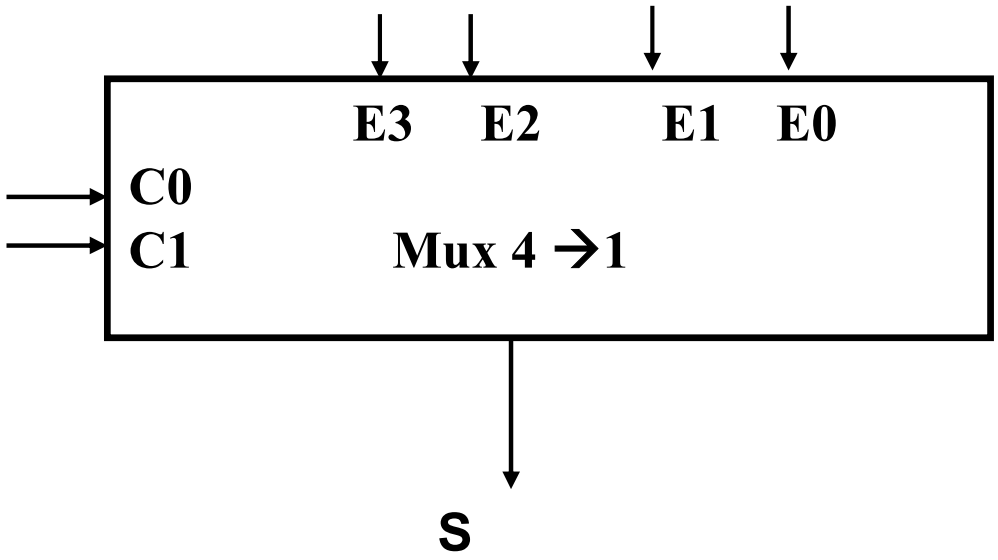
- La sortie S prend la valeur de l'entrée de donnée:
- En1 lorsque l'entrée de sélection Com est active (niveau 1).
- En2 lorsque l'entrée de sélection Com est inactive (niveau 0).

L'entrée Com permet donc d'aiguiller vers la sortie S soit l'information arrivant par l'entrée En1, soit celle arrivant par l'entrée En2.



5.2 Multiplexeur 4 →1

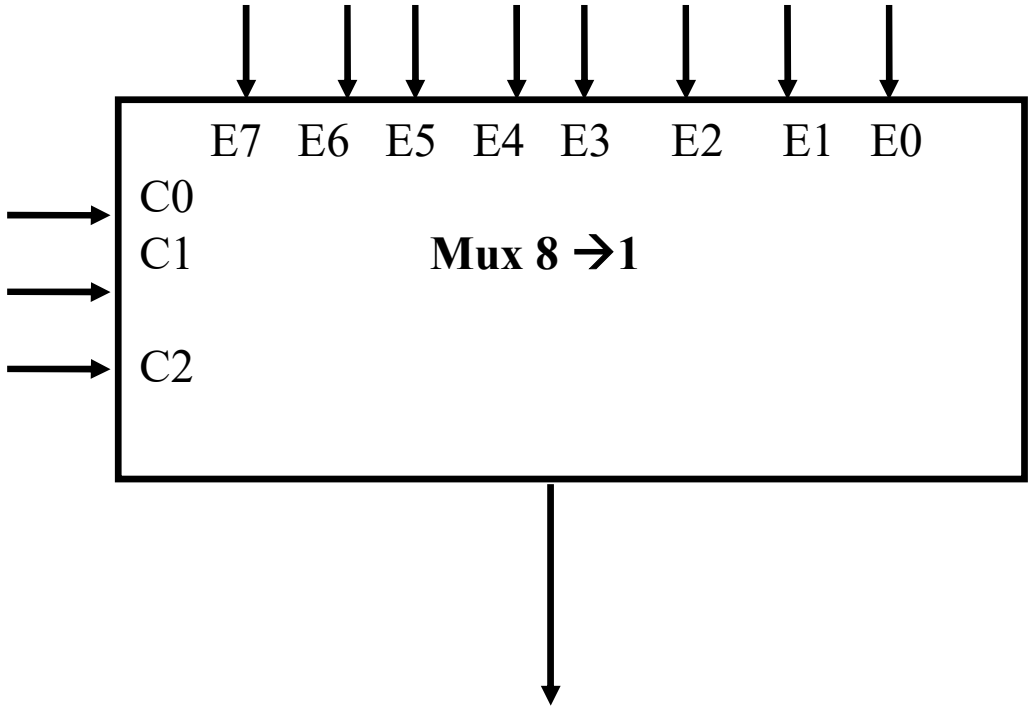
C1	C0		S
0	0		E0
0	1		E1
1	0		E2
1	1		E3



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

5.3 Multiplexeur 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



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

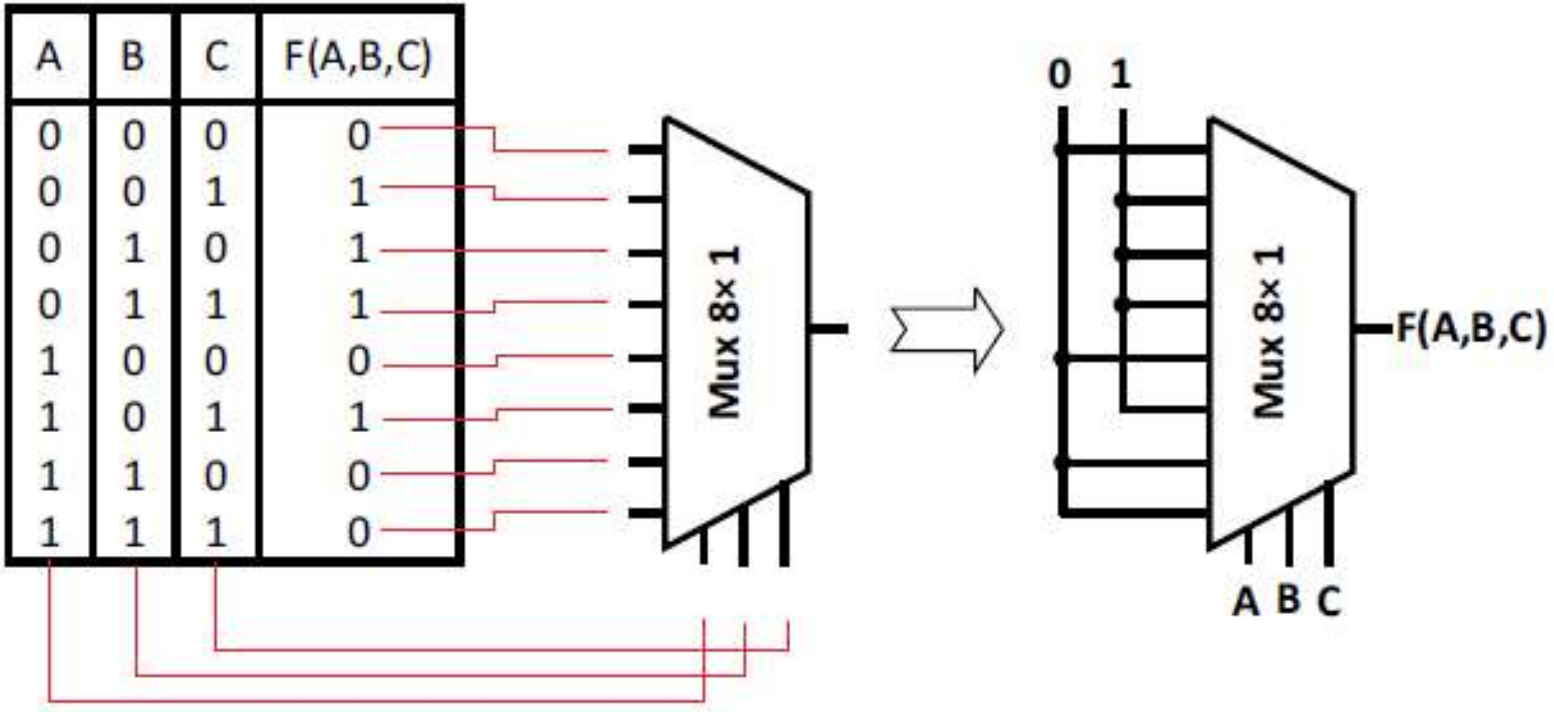
Fonctions logiques via des multiplexeurs:

On peut toujours générer des fonctions logiques quelconques en utilisant des multiplexeurs et des portes logiques de base. Il suffit de relier les variables de la fonction a générer aux déférentes entrées du multiplexeur (entrées standards et entrées de commandes).

Exemple 1 : réaliser

la fonction $F(A,B,C)=\bar{B}C+\bar{A}B$ en utilisant un multiplexeur 8x1.

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	0	0



Fonctions logiques via des multiplexeurs:

On peut toujours générer des fonctions logiques quelconques en utilisant des multiplexeurs et des portes logiques de base. Il suffit de relier les variables de la fonction a générer aux déférentes entrées du multiplexeur (entrées standards et entrées de commandes).

Exemple 1 : réaliser la fonction en utilisant un multiplexeur 8x1.

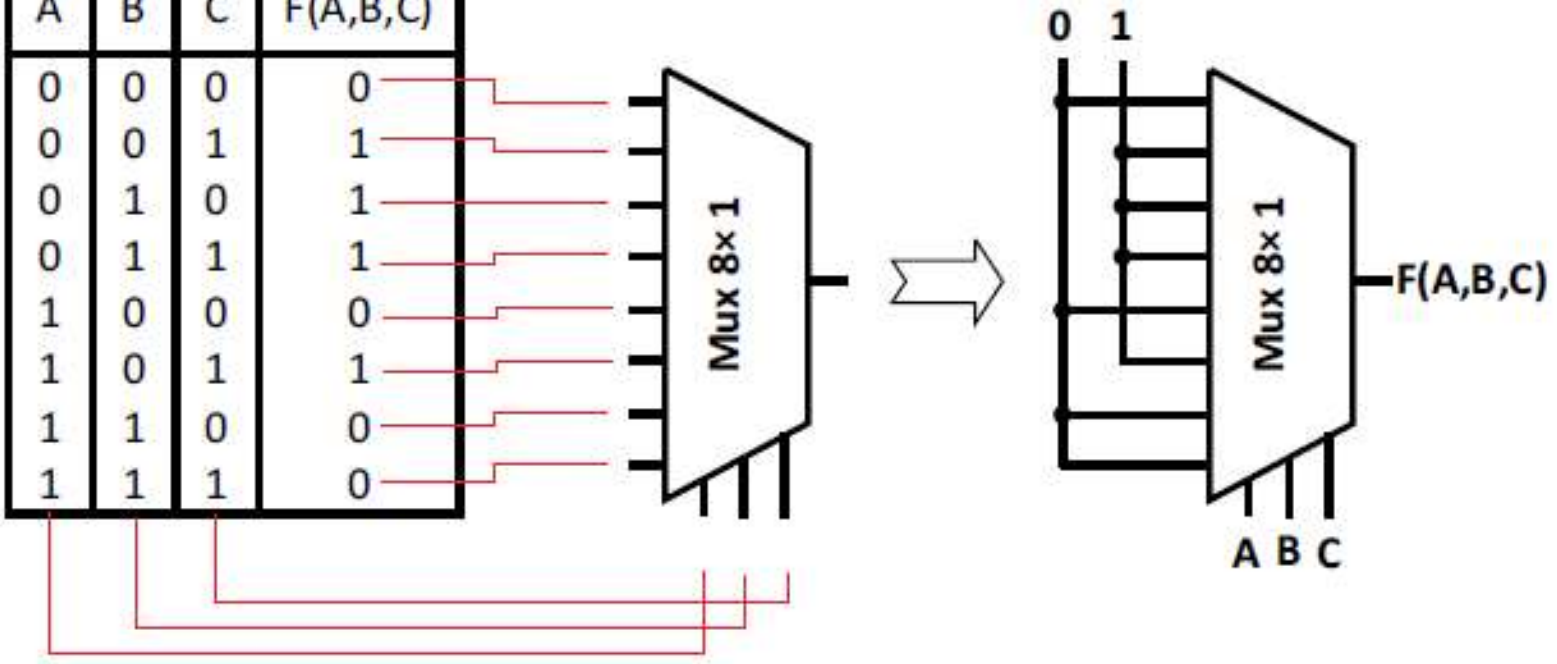
$$F(A,B,C)=\bar{B}C+\bar{A}B$$

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

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

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

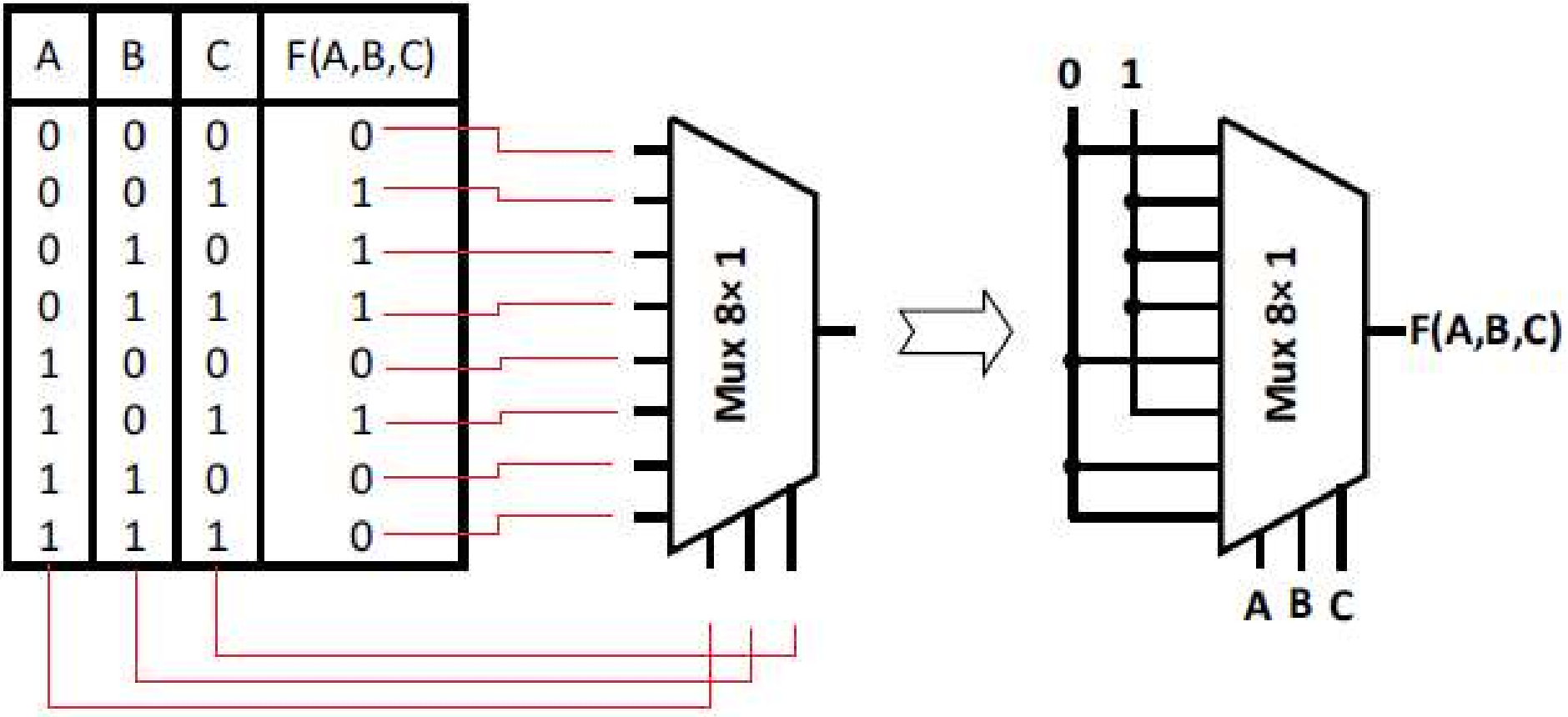
A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Fonctions logiques via des multiplexeurs:

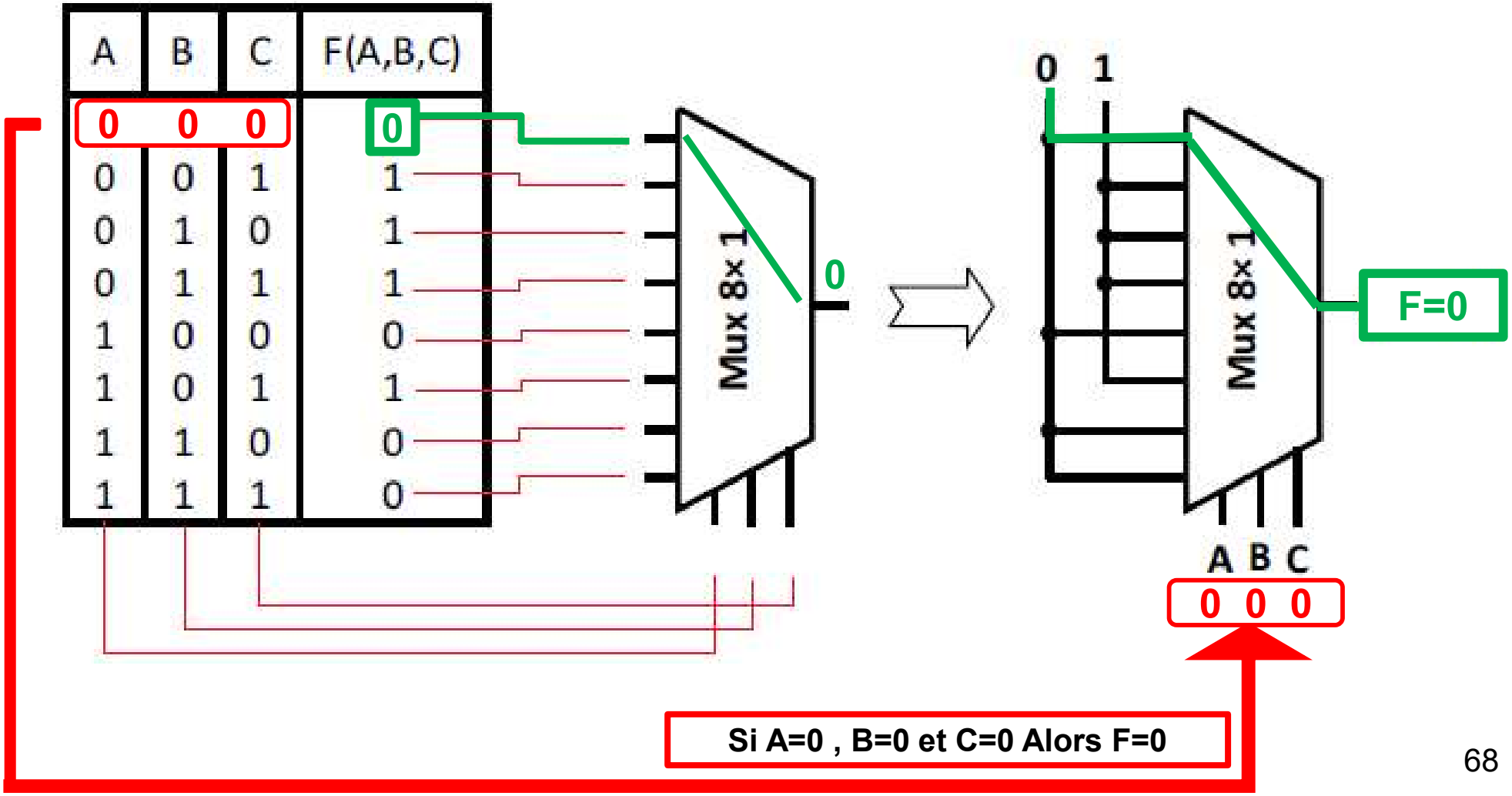
$$F(A,B,C)=\bar{A}\,\bar{B}C+\bar{A}\,B\,\bar{C}+\bar{A}\,B\,C+A\,\bar{B}\,C$$

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



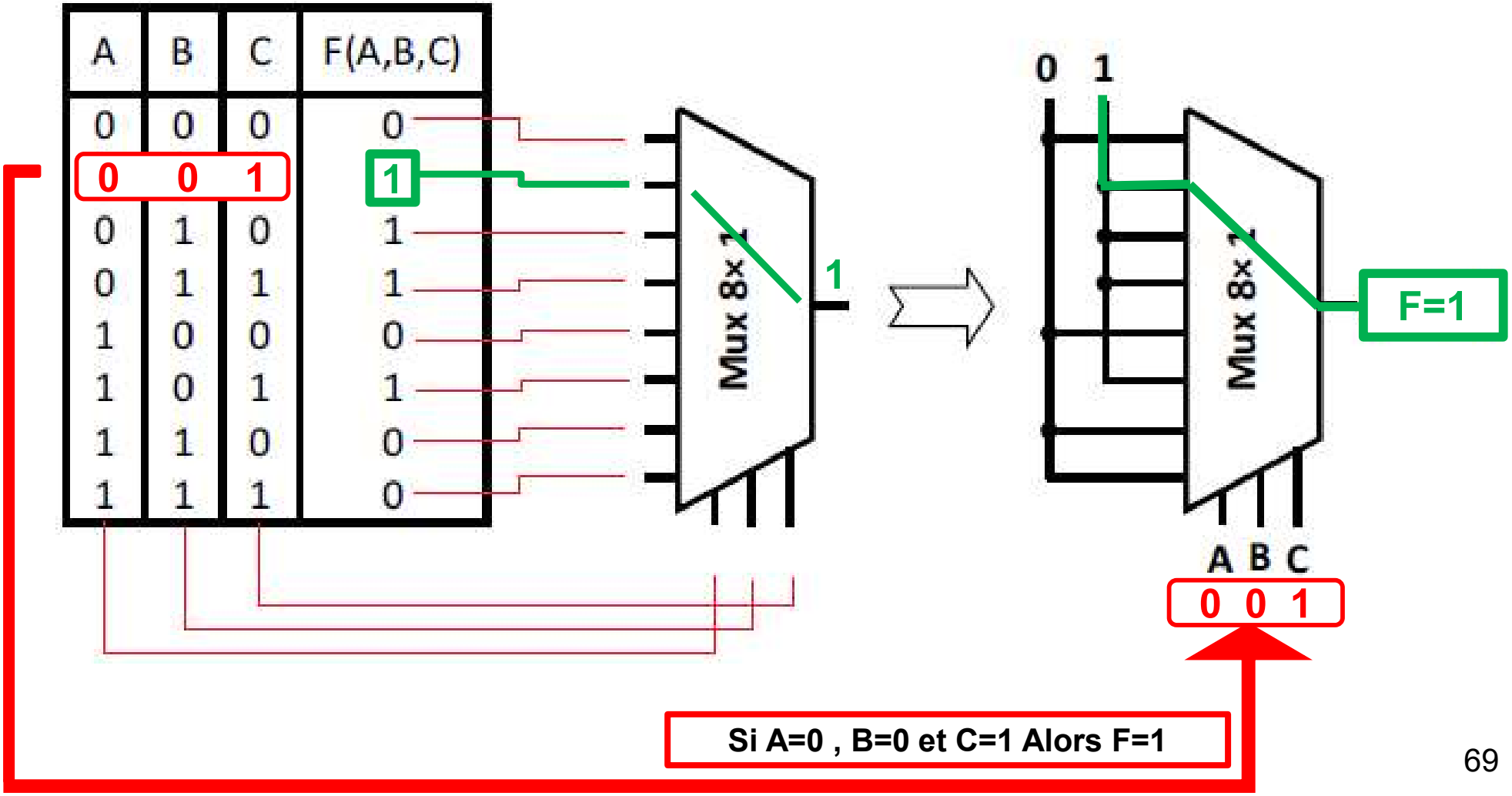
Fonctions logiques via des multiplexeurs:

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



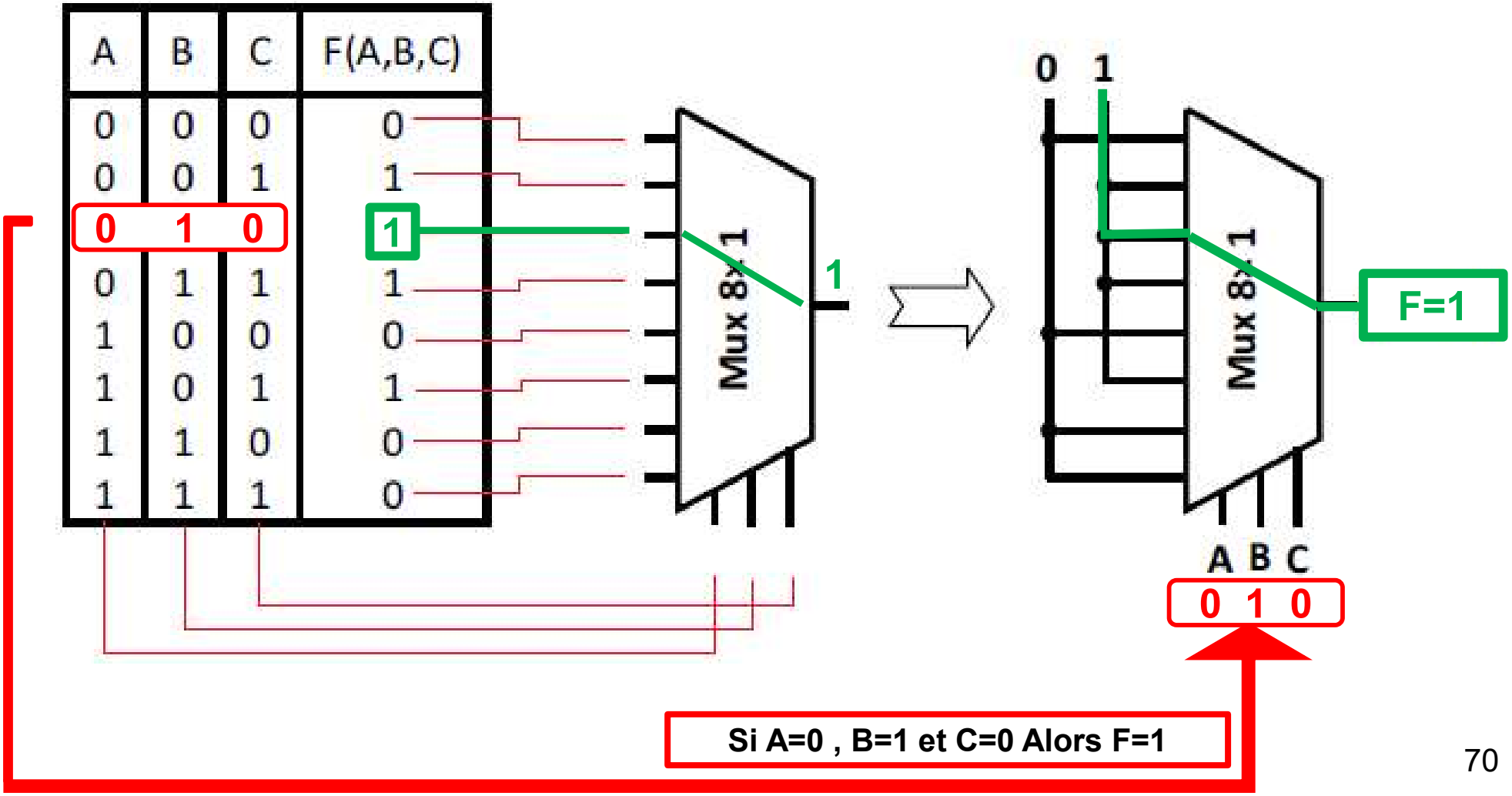
Fonctions logiques via des multiplexeurs:

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



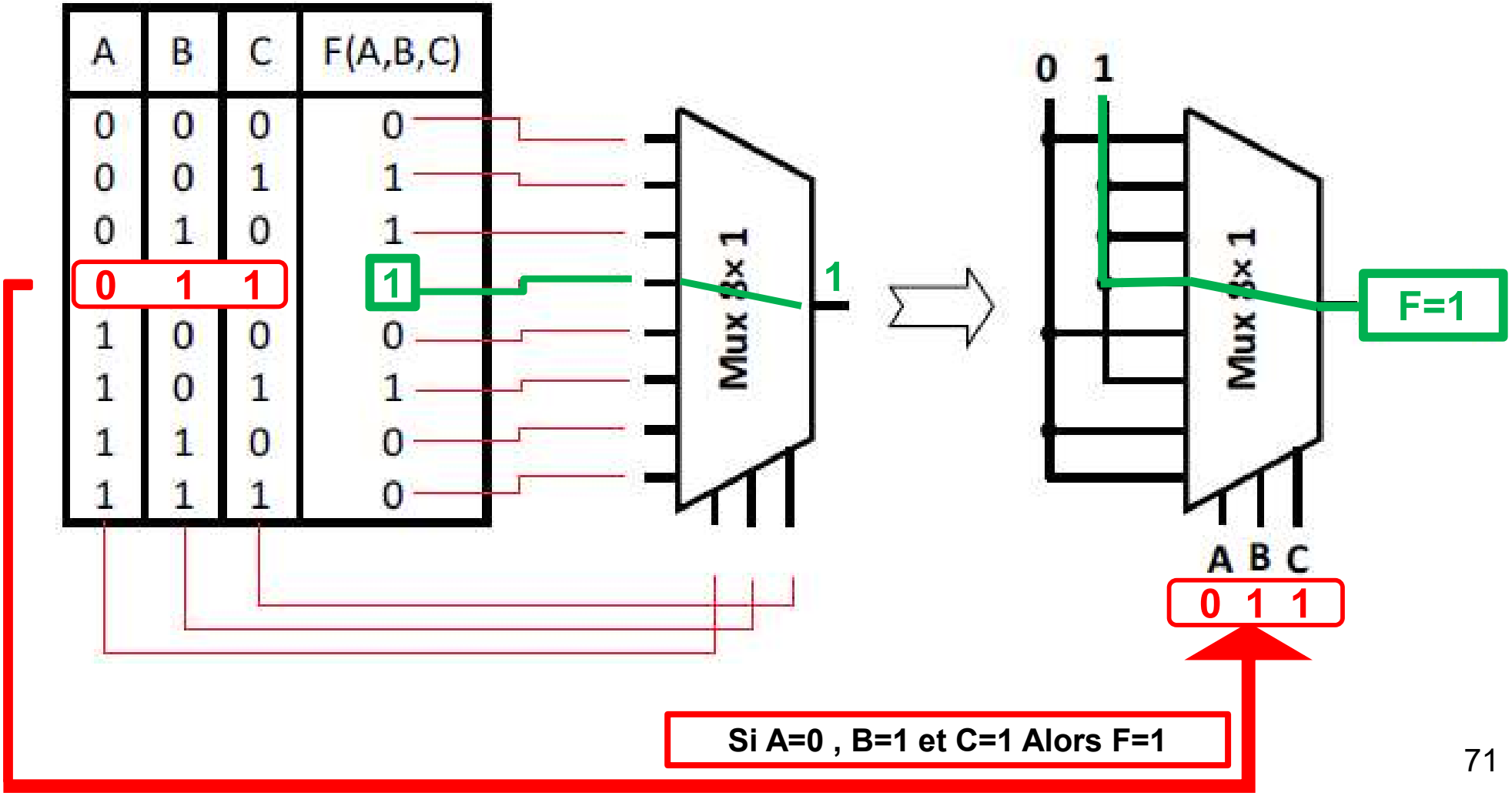
Fonctions logiques via des multiplexeurs:

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



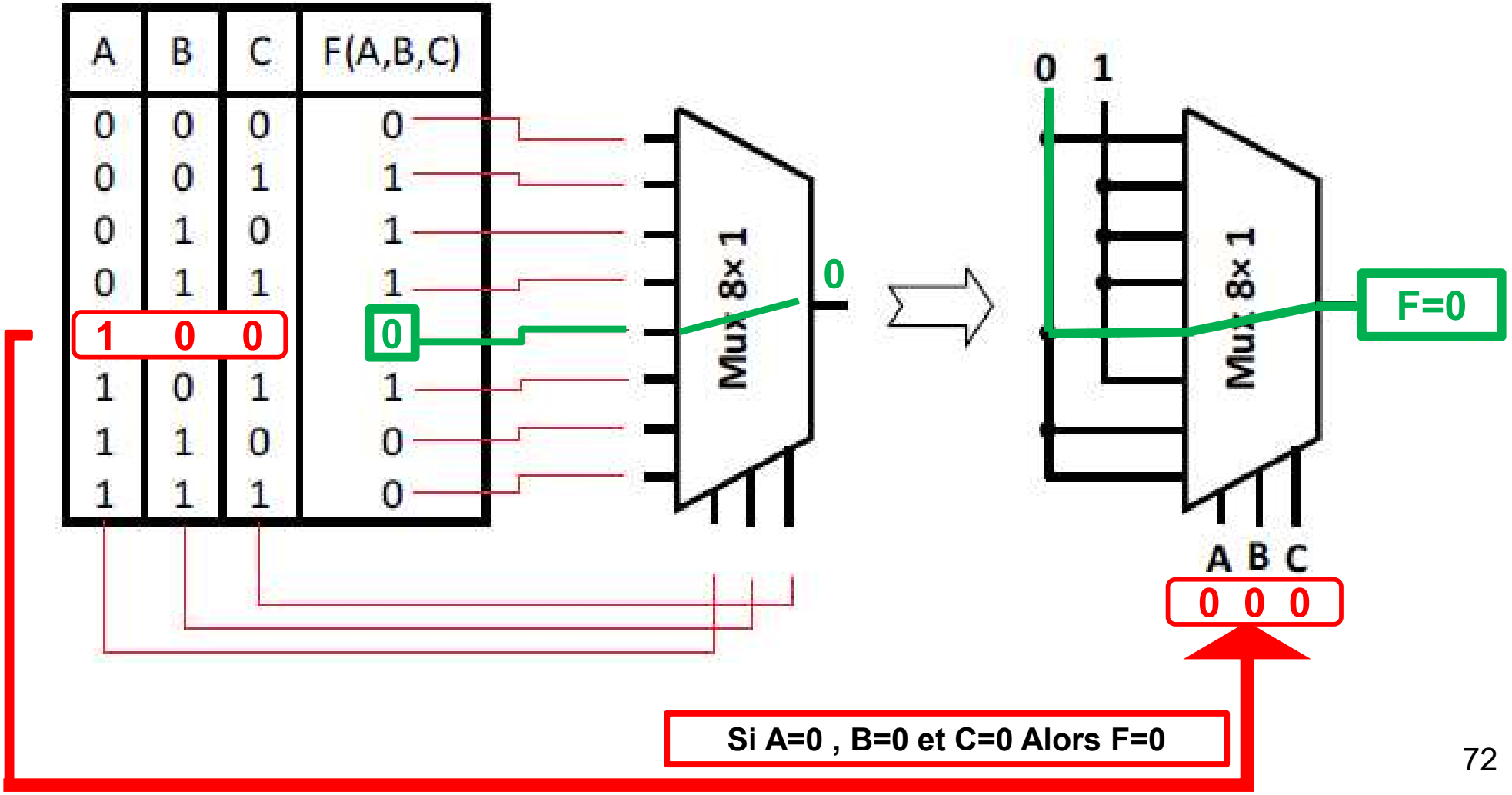
Fonctions logiques via des multiplexeurs:

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



Fonctions logiques via des multiplexeurs:

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



Fonctions logiques via des multiplexeurs:

Exemple 1 : réaliser la fonction en utilisant un multiplexeur 4x1.

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

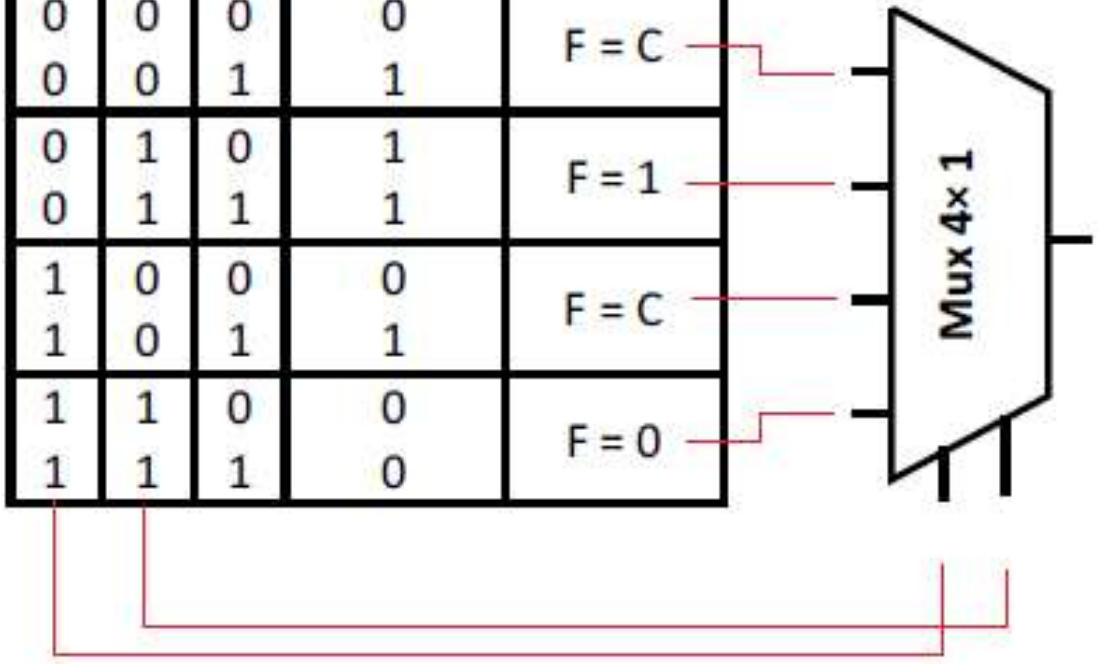
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

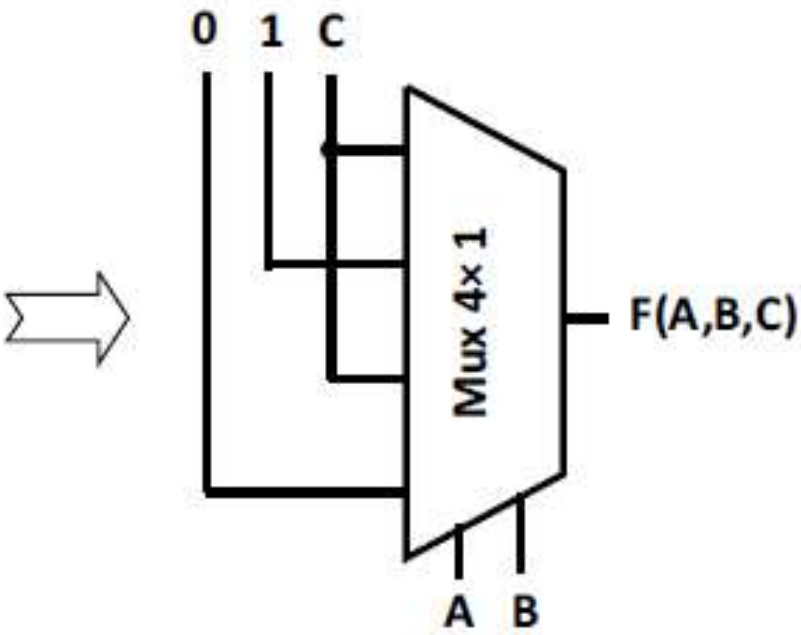
$$F(A,B,C) = \overline{B}C(A+\overline{A}) + \overline{A}B(C+\overline{C})$$

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

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

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:

Exemple 1 : réaliser la fonction en utilisant un multiplexeur 4x1.

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

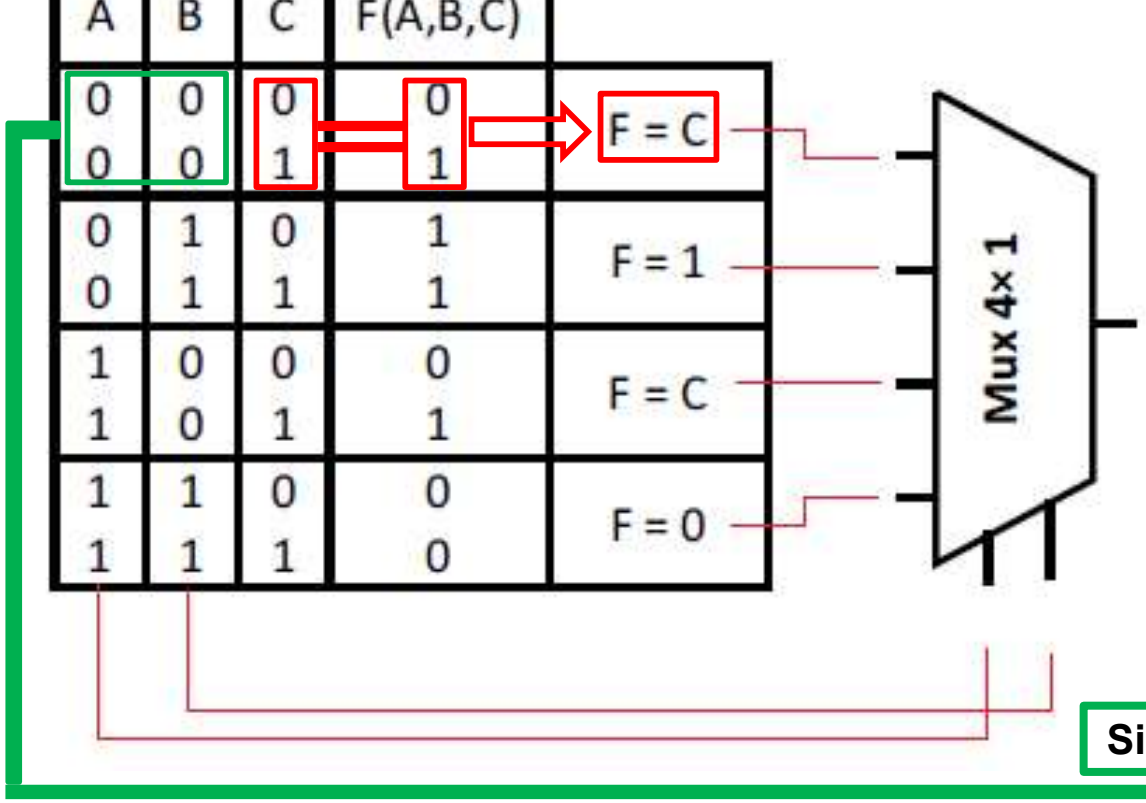
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

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

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

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

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	



Le Mux 4x1 permet d'aiguiller la valeur de **C** vers la Sortie **F**.

Fonctions logiques via des multiplexeurs:

Exemple 1 : réaliser la fonction en utilisant un multiplexeur 4x1.

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

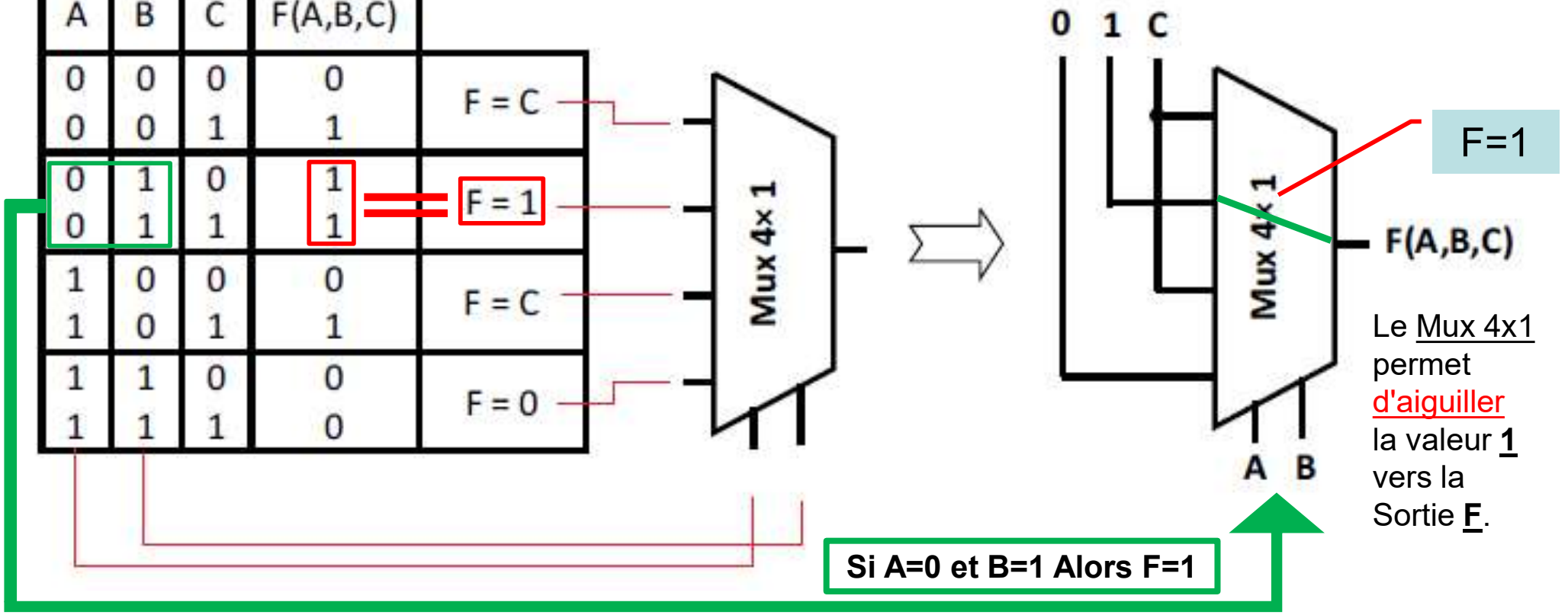
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

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

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

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

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	



Le Mux 4x1 permet d'aiguiller la valeur **1** vers la Sortie **F**.

Fonctions logiques via des multiplexeurs:

Exemple 1 : réaliser la fonction en utilisant un multiplexeur 4x1.

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

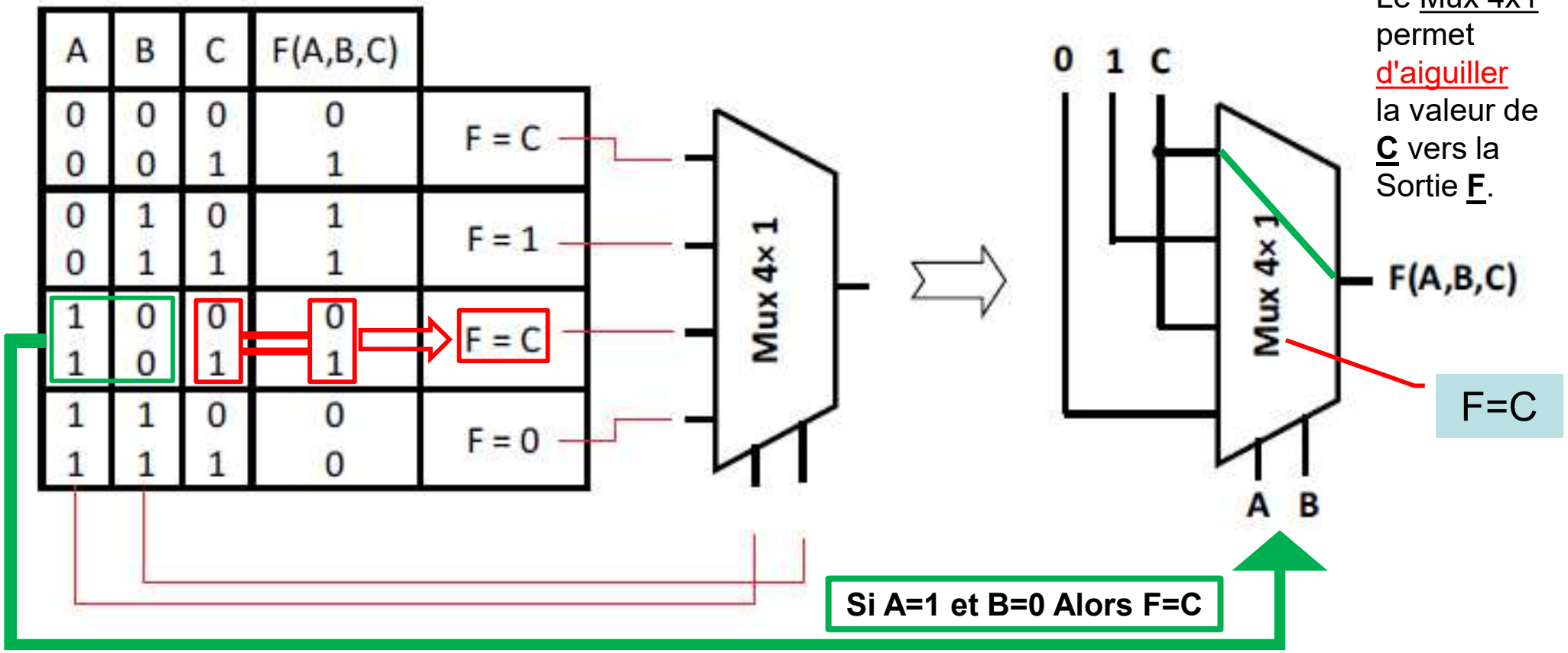
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

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

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

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

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	



Le Mux 4x1 permet d'aiguiller la valeur de **C** vers la Sortie **F**.

Fonctions logiques via des multiplexeurs:

Exemple 1 : réaliser la fonction en utilisant un multiplexeur 4x1.

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

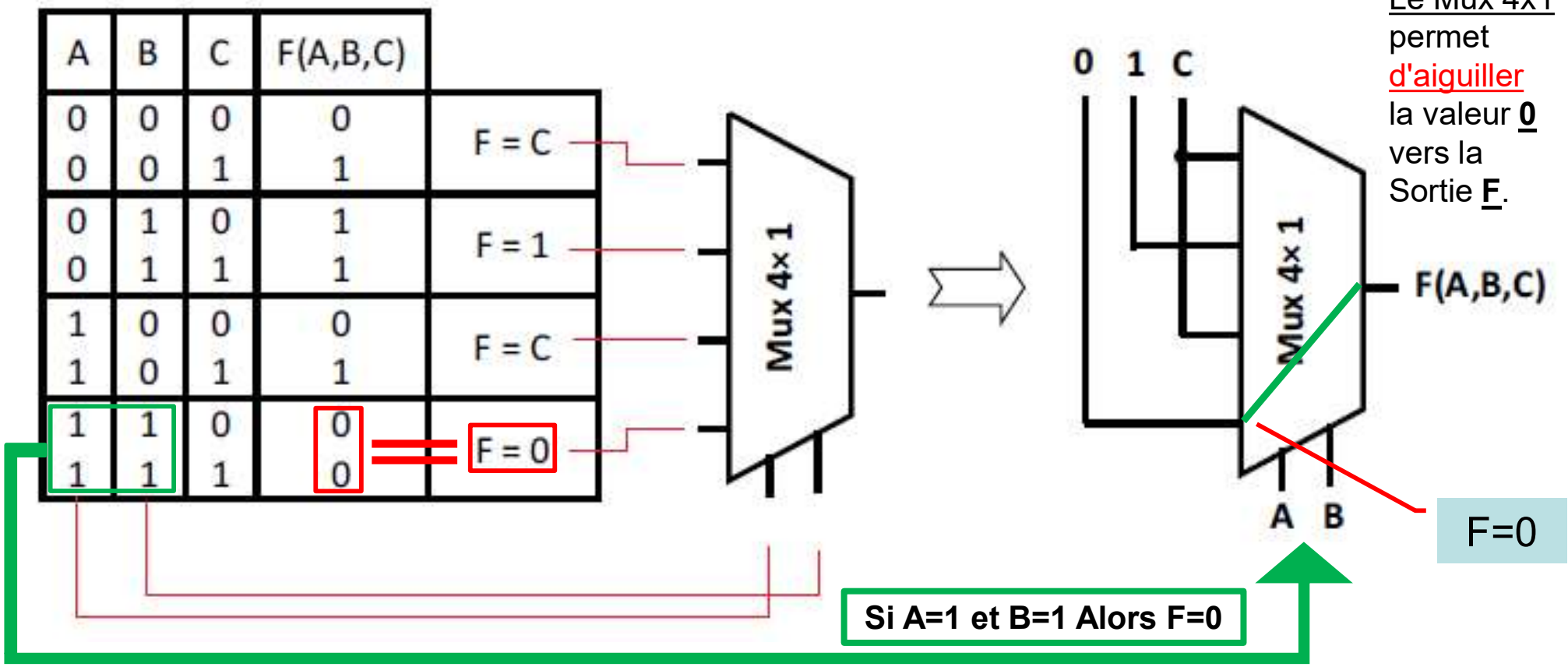
$$F(A,B,C) = \overline{B}C + \overline{A}B$$

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

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

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

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	

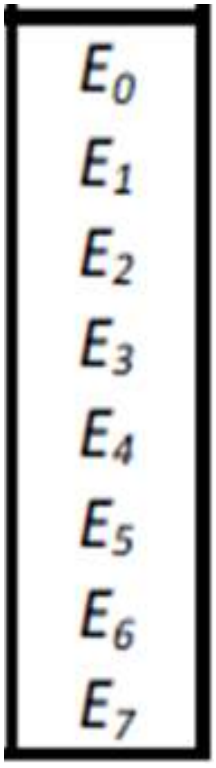


Le Mux 4x1 permet d'aiguiller la valeur **0** vers la Sortie **F**.

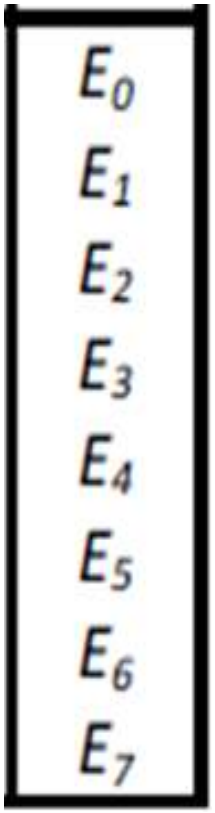
Exemple : Réalisation d'un additionneur complet avec des multiplexeurs 8→1

•Nous avons besoin d'utiliser deux multiplexeurs :Le premier pour réaliser la fonction de la somme et l'autres pour donner la retenue.

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



Réalisation de la fonction de: la somme

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

On pose :

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

Alors:

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

Réalisation de la fonction de la: retenue

$$R_i = \overline{A_i}\,\overline{B_i}\,R_{i-1}.(0) + \overline{A_i}\,B_i\,R_{i-1}.(0) + A_i\,\overline{B_i}\,R_{i-1}.(0) + A_i\,B_i\,R_{i-1}.(1) +$$
$$A_i\,\overline{B_i}\,\overline{R_{i-1}}.(0) + A_i\,\overline{B_i}\,R_{i-1}.(1) + A_i\,B_i\,\overline{R_{i-1}}.(1) + A_i\,B_i\,R_{i-1}.(1)$$

On pose :

$C2=A_i$

$C1=B_i$

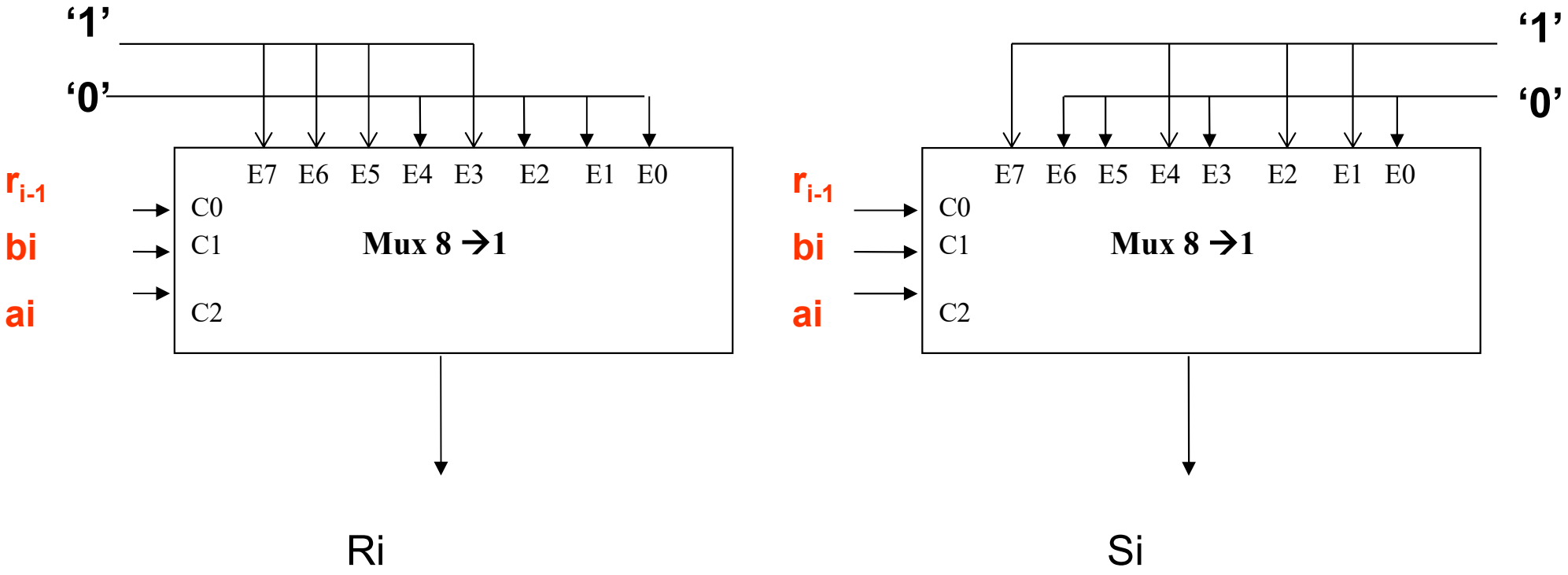
$C0=R_{i-1}$

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

Alors:

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

Réalisation d'un additionneur complet avec des multiplexeurs 8→1



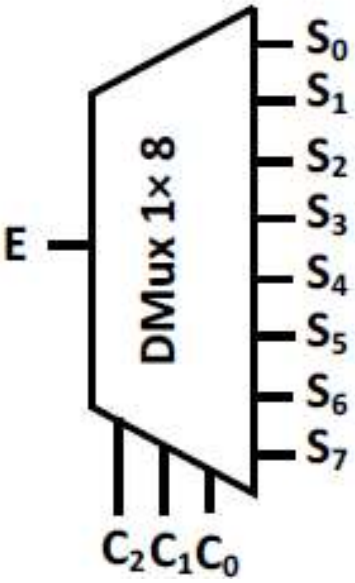
6. Démultiplexeurs

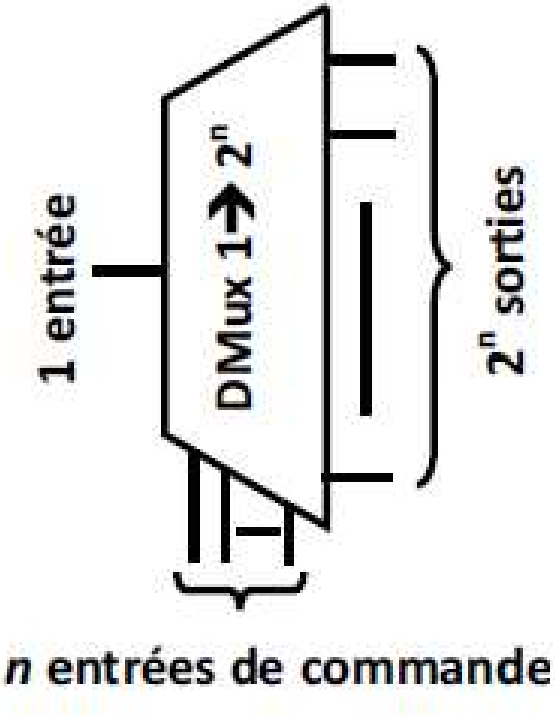
- **Il joue le rôle inverse d'un multiplexeurs, il permet de faire passer une information dans l'une des sorties selon les valeurs des entrées de commandes.**
- **Il possède :**
  - **une seule entrée**
  - **2^n sorties**
  - **N entrées de sélection (commandes)**

6. Démultiplexeurs

- **Définition :** Un démultiplexeur est un circuit logique combinatoire qui comporte une seule entrée, n entrées de commande et 2^n sorties. Il permet d'aiguiller la valeur de la ligne d'entrée vers la ligne de sortie indiquée dans ses entrées de commande.
- Synthèse du circuit (démultiplexeur 1x8)

Les entrées/sorties.

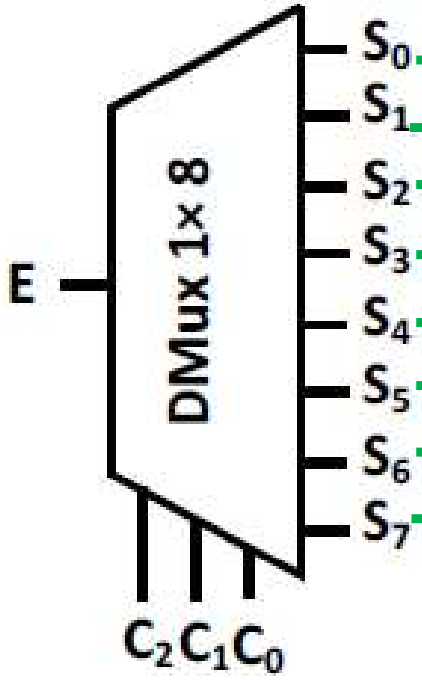




6. Démultiplexeurs

- **Définition :** Un démultiplexeur est un circuit logique combinatoire qui comporte une seule entrée, n entrées de commande et 2^n sorties. Il permet d'aiguiller la valeur de la ligne d'entrée vers la ligne de sortie indiquée dans ses entrées de commande.
- Svnthèse du circuit (démultiplexeur 1x8)

Les entrées/sorties.



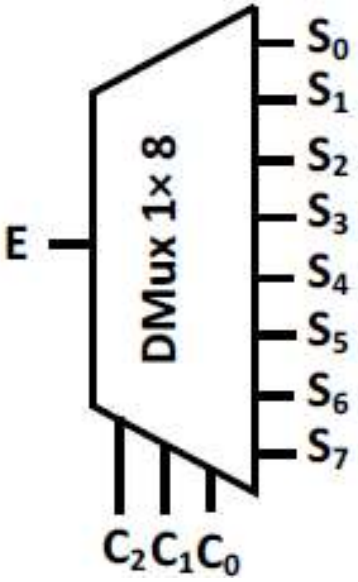
La table de vérité.

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

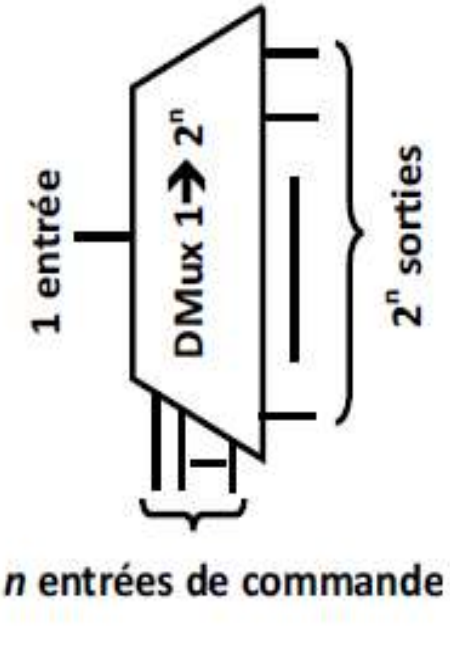
6. Démultiplexeurs

- **Définition :** Un démultiplexeur est un circuit logique combinatoire qui comporte une seule entrée, n entrées de commande et 2^n sorties. Il permet d'aiguiller la valeur de la ligne d'entrée vers la ligne de sortie indiquée dans ses entrées de commande.
- Synthèse du circuit (démultiplexeur 1x8)

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. Démultiplexeurs

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. Démultiplexeurs

Les fonctions logiques.

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

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

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

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

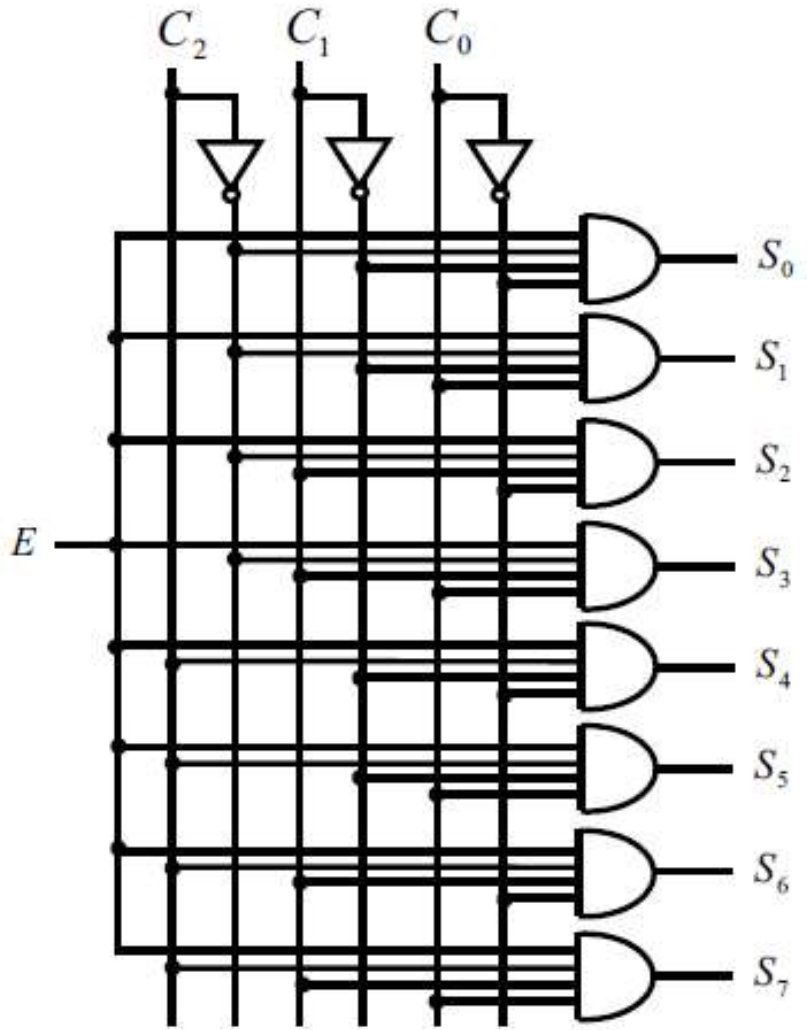
$$S_4 = C_2.\overline{C_1}.\overline{C_0}.E$$

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

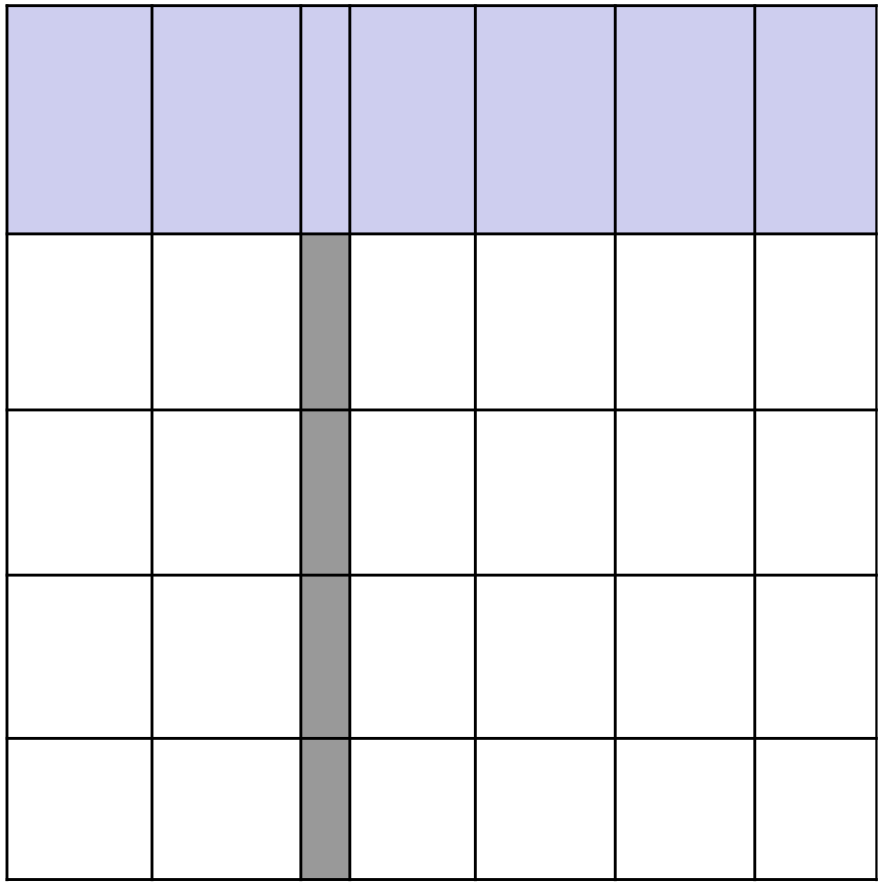
$$S_6 = C_2.C_1.\overline{C_0}.E$$

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

Le schéma du circuit.

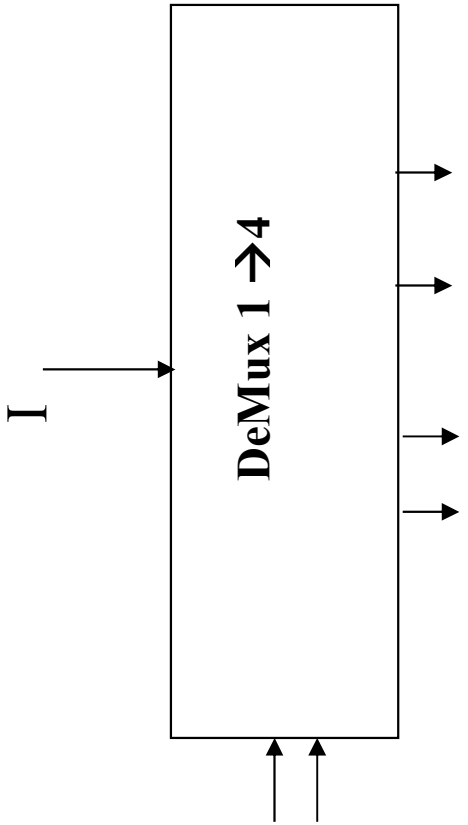


6.1 Demultiplexeur 1→4



- **Réaliser un circuit**

Démultiplexeur 1→4



6.1 Demultiplexeur 1→4

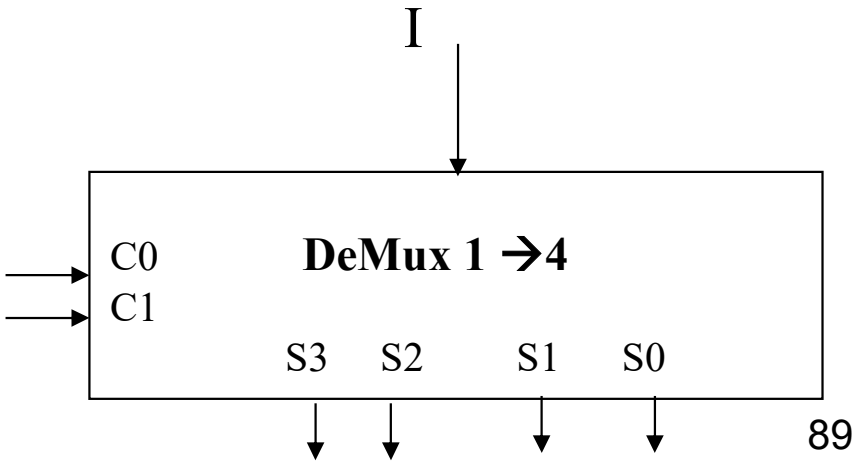
C1	C0		S3	S2	S1	S0
0	0		0	0	0	i
0	1		0	0	i	0
1	0		0	i	0	0
1	1		i	0	0	0

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

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

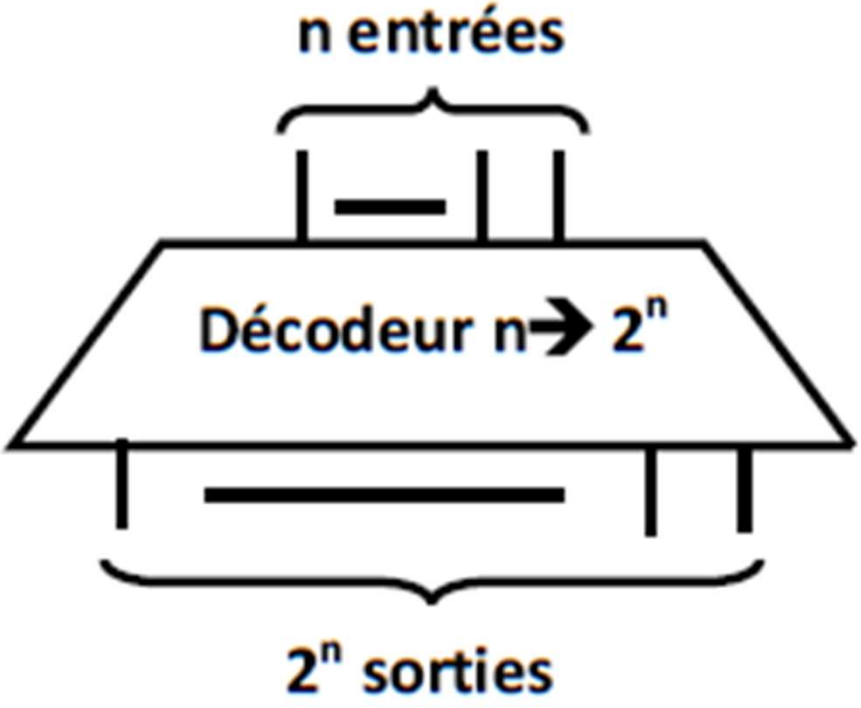
$$S2 = C1.\overline{C0}.(I)$$

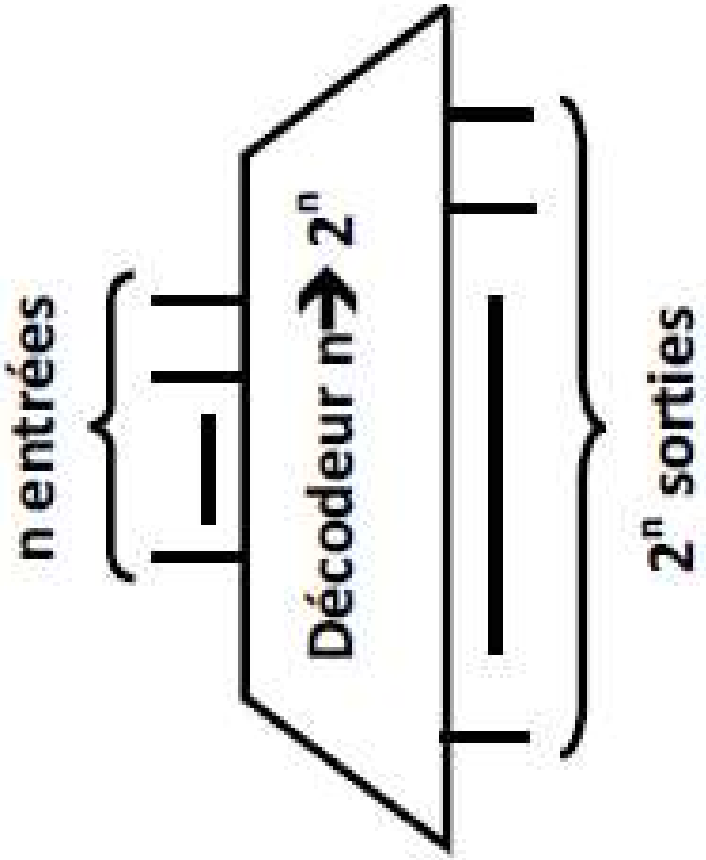
$$S3 = C1.C0.(I)$$



7. Le décodeur binaire

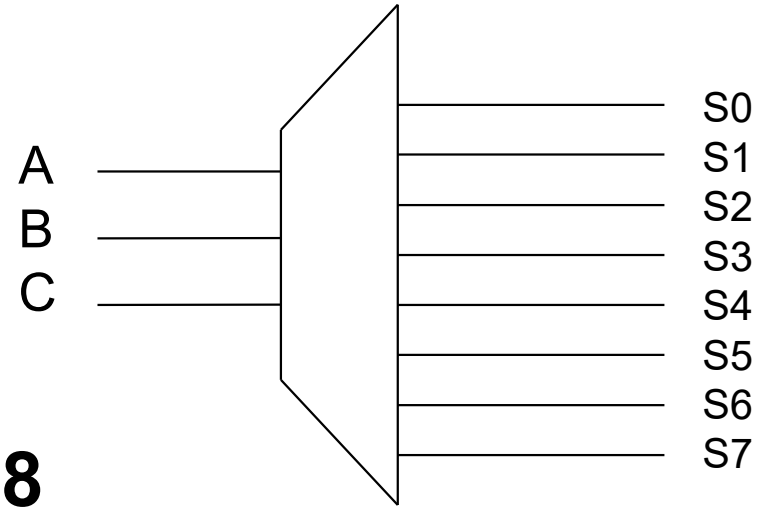
Définition : Un décodeur n bits est un circuit logique combinatoire de n entrées et 2^n sorties. Il permet d'activer la ligne de sortie correspondant au code binaire présent en entrée.





7. Le décodeur binaire

- C’est un circuit combinatoire qui est constitué de :
  - N : entrées de données
  - 2^n sorties
  - Pour chaque combinaison en entrée une seule sortie est active à la fois



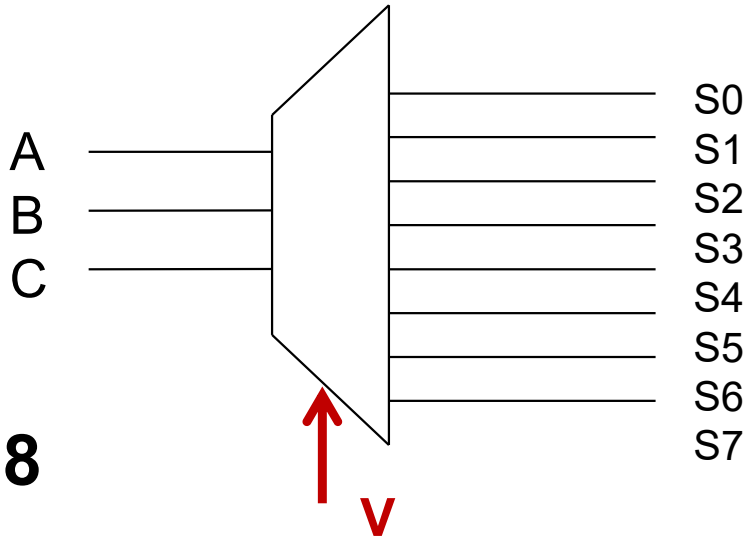
Exemple: Un décodeur 3→8

7. Le décodeur binaire

- Décodeur avec entrée de validation
  La plupart des décodeurs auront une entrée de validation (enable). Cette entrée permet d'activer le fonctionnement du décodeur.
- Si EN = 0, toutes les sorties sont à 0.
- Si EN = 1, le décodeur fonctionne normalement.

On peut aussi avoir:

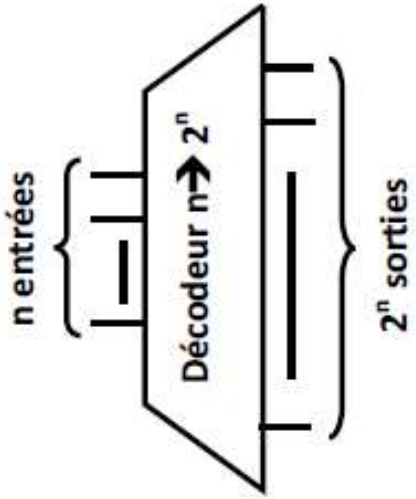
un signal de contrôle inversé (EN).



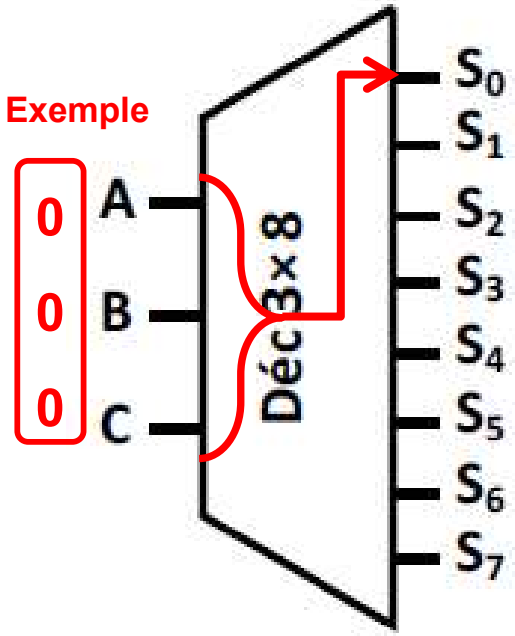
Exemple: Un décodeur 3→8

7. Le décodeur binaire

Synthèse du circuit (décodeur 3x8)



Les entrées/sorties.

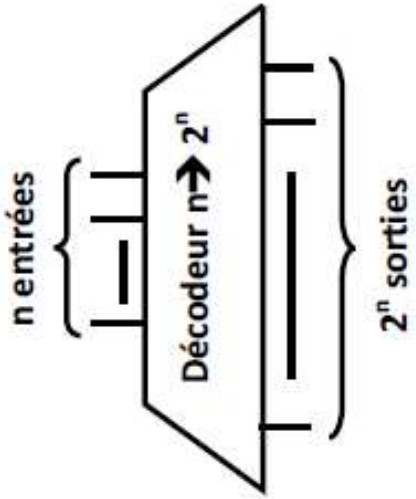


La table de vérité.

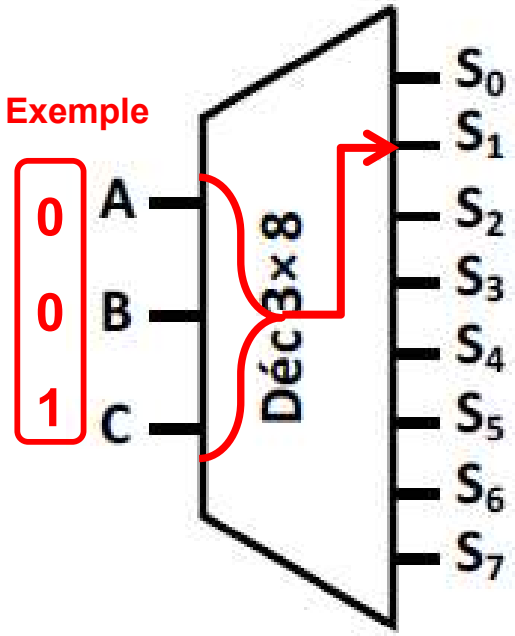
A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	**0**	**0**	**1**	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

7. Le décodeur binaire

Synthèse du circuit (décodeur 3x8)



Les entrées/sorties.

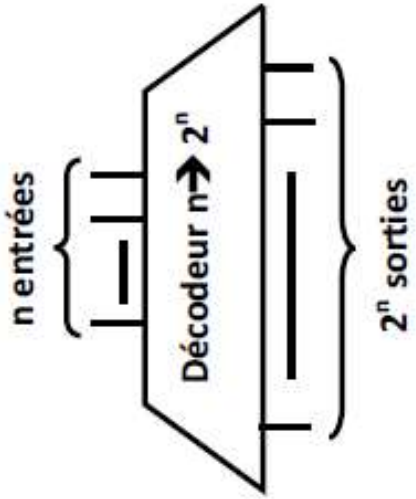


La table de vérité.

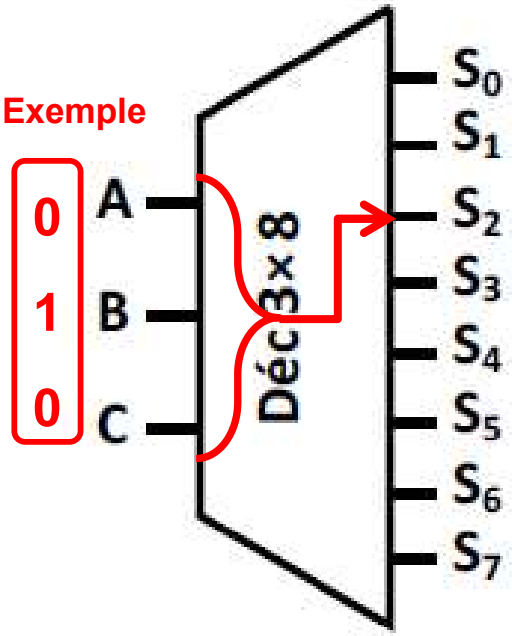
A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	**0**	**1**	0	**1**	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

7. Le décodeur binaire

Synthèse du circuit (décodeur 3x8)



Les entrées/sorties.

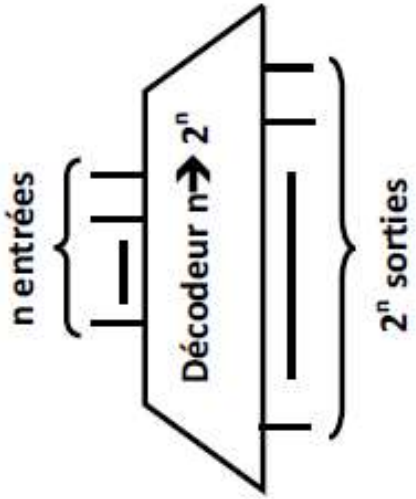


La table de vérité.

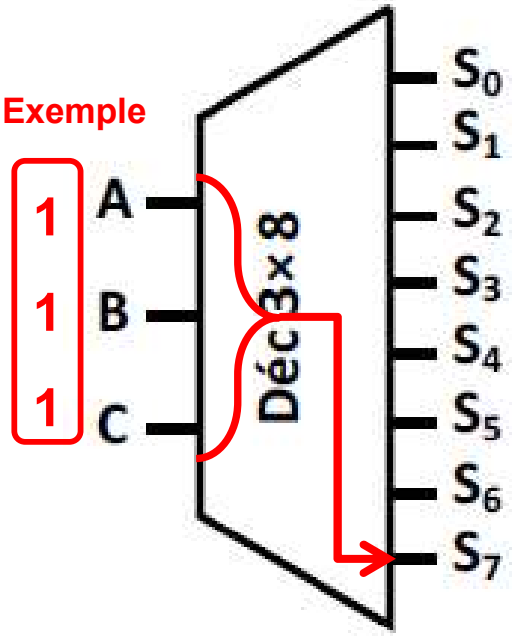
A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	**1**	**0**	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

7. Le décodeur binaire

Synthèse du circuit (décodeur 3x8)



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. Le décodeur binaire

Synthèse du circuit (décodeur 3x8)

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. Le décodeur binaire

Les expréssions logiques.

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

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

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

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

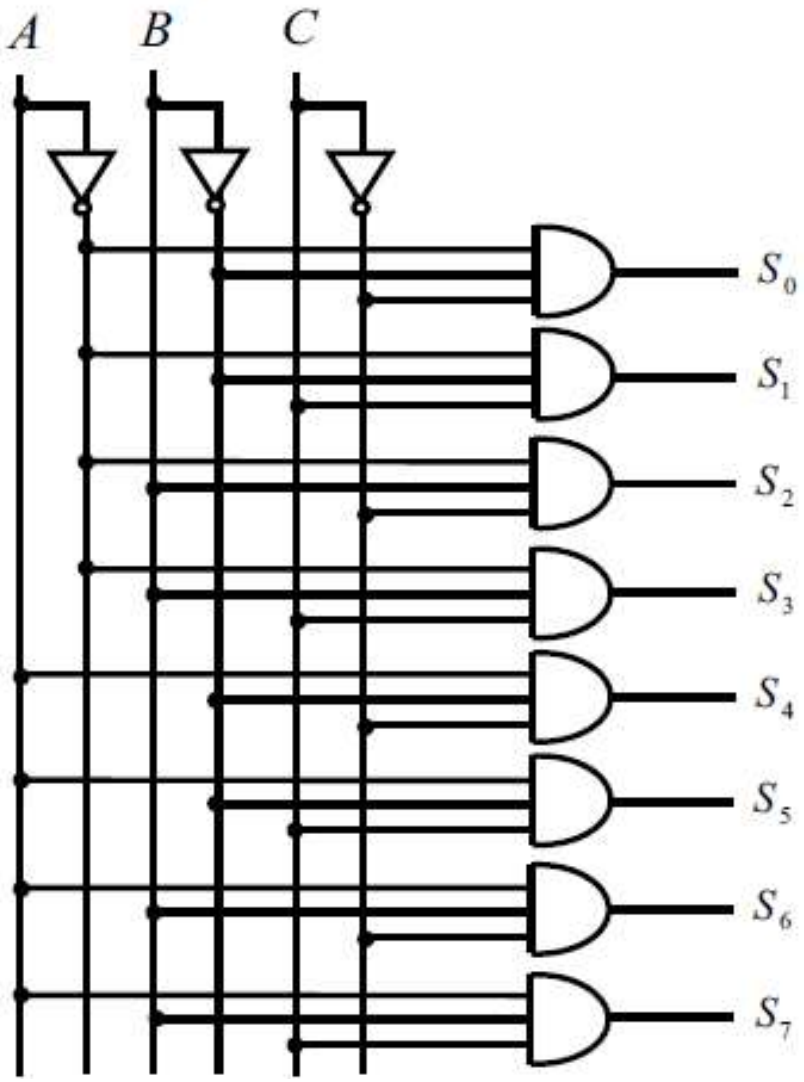
$$S_4 = A.\overline{B}.\overline{C}$$

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

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

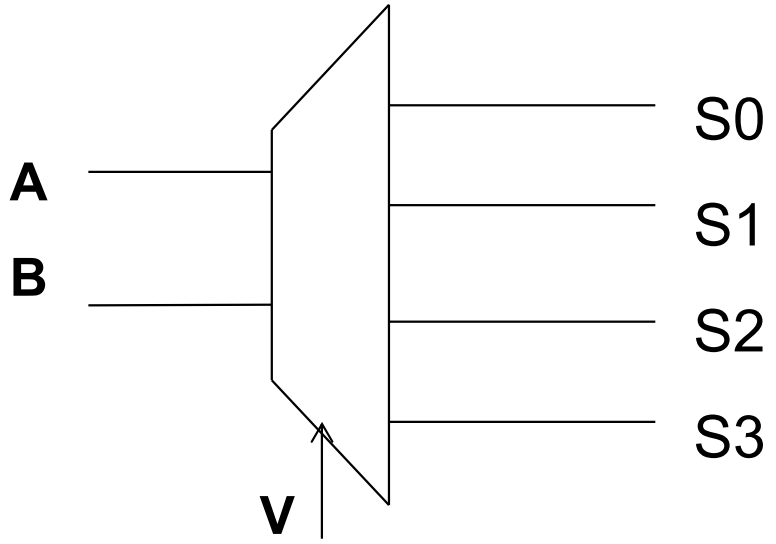
$$S_7 = A.B.C$$

Le schéma du circuit.



Décodeur 2→4 avec signal d'activation

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



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

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

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

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

Fonctions logiques via des Décodeurs:

On peut également générer des fonctions logiques quelconques en utilisant des décodeurs et des portes logiques de base.
Il suffit de relier les variables de la fonction a générer aux entrées du décodeur et les sorties correspondant aux différents Mintermes de la fonction aux entrées d'une ou plusieurs portes logiques de base.

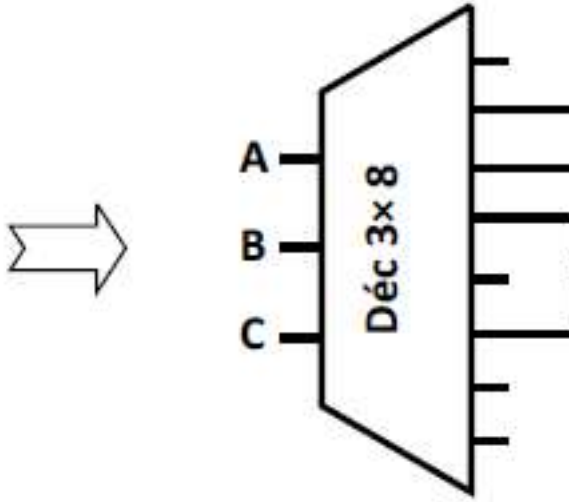
Exemple 1 : réaliser la fonction: $F(A,B,C)=\bar{B}C+\bar{A}B$ en utilisant un multiplexeur 8x1.

Fonctions logiques via des Décodeurs:

Exemple 1 : réaliser la fonction $F(A,B,C)=\bar{B}C+\bar{A}B$ en utilisant un décodeur 8x1.

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

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

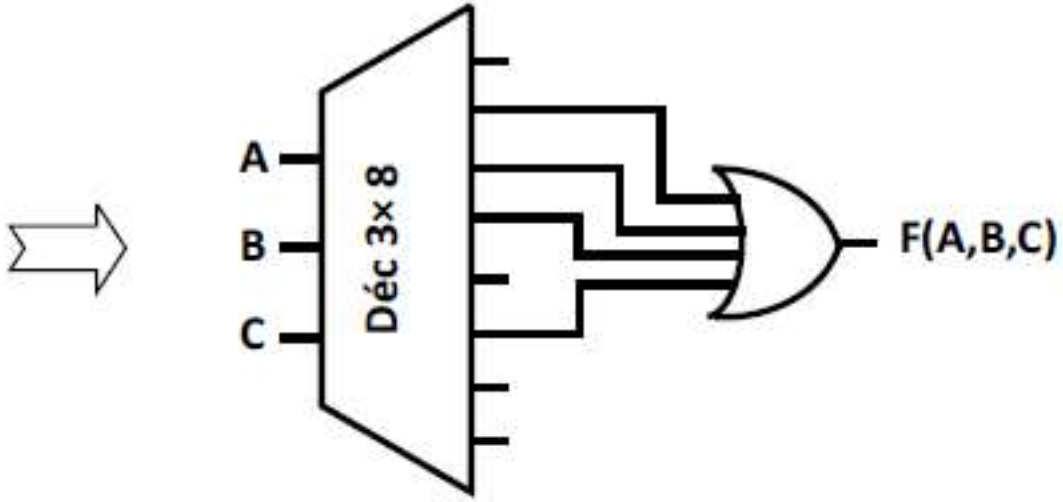


Fonctions logiques via des Décodeurs:

Exemple 1 : réaliser la fonction $F(A,B,C)=\bar{B}C+\bar{A}B$ en utilisant un décodeur 8x1.

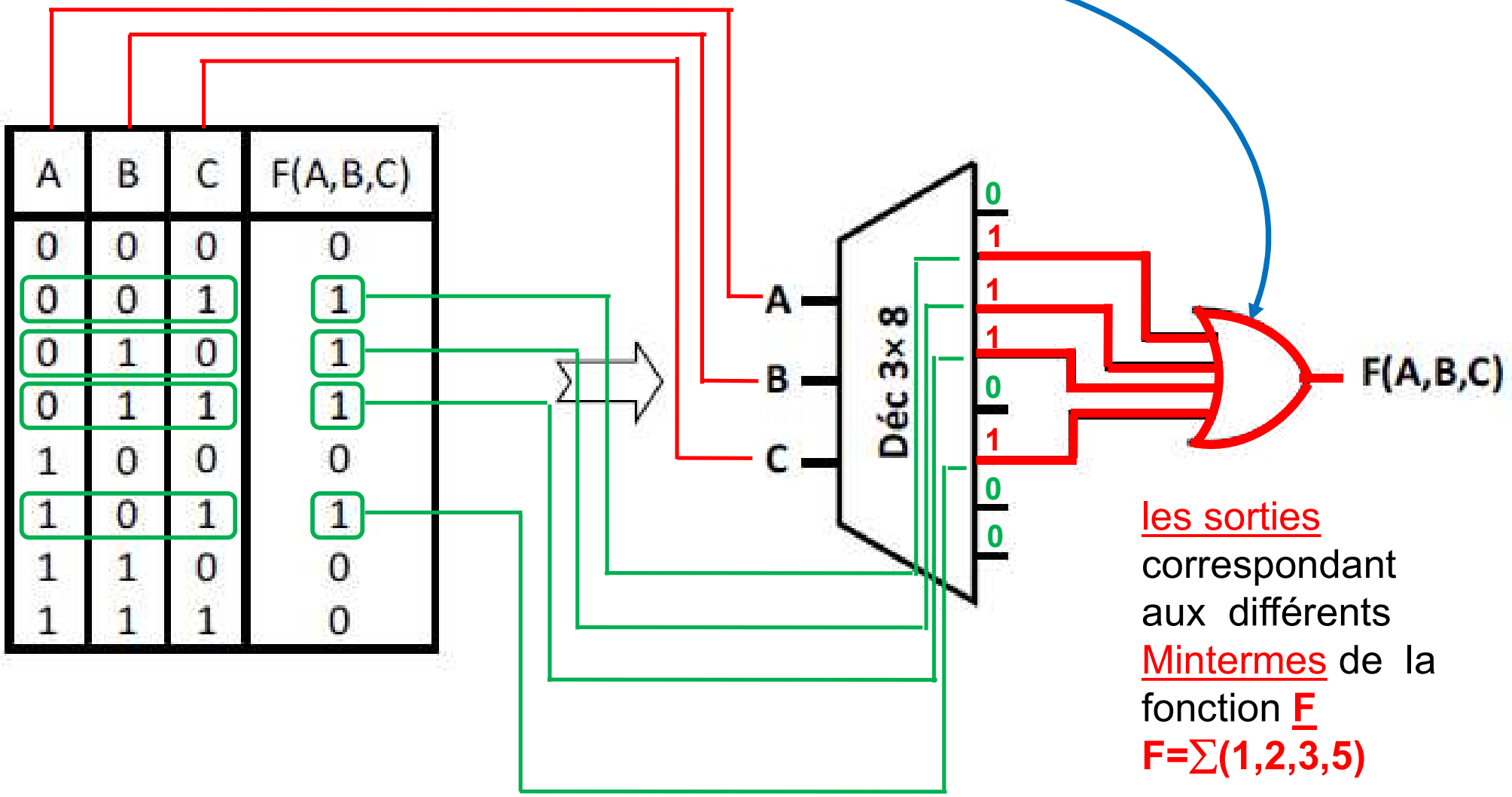
$$F(A,B,C)=\bar{B}C+\bar{A}B$$
$$F(A,B,C)=\bar{B}C(A+\bar{A})+\bar{A}B(C+\bar{C})$$
$$F(A,B,C)=(\bar{A}\bar{B}C+A\bar{B}C)+(\bar{A}B\bar{C}+\bar{A}BC)$$
$$F(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C$$

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



Fonctions logiques via des Décodeurs:

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



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

les sorties correspondant aux différents Mintermes de la fonction **F**

F=∑(1,2,3,5)

Réalisation d'un additionneur complet avec des décodeurs binaire 3→8

Rappel :

Table de vérité d'un additionneur complet sur 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 retenue 0
0+1 = 1 + 0 = 1 retenue 0
1 + 1 = 0 retenue 1
1+1+1 = 1 retenue 1

Réalisation d'un additionneur complet avec des décodeurs binaire 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

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

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**

Réalisation d'un additionneur complet avec des décodeurs binaire 3→8

On pose $A=A_i$, $B=B_i$, $C=R_{i-1}C$

Alors:

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

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

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

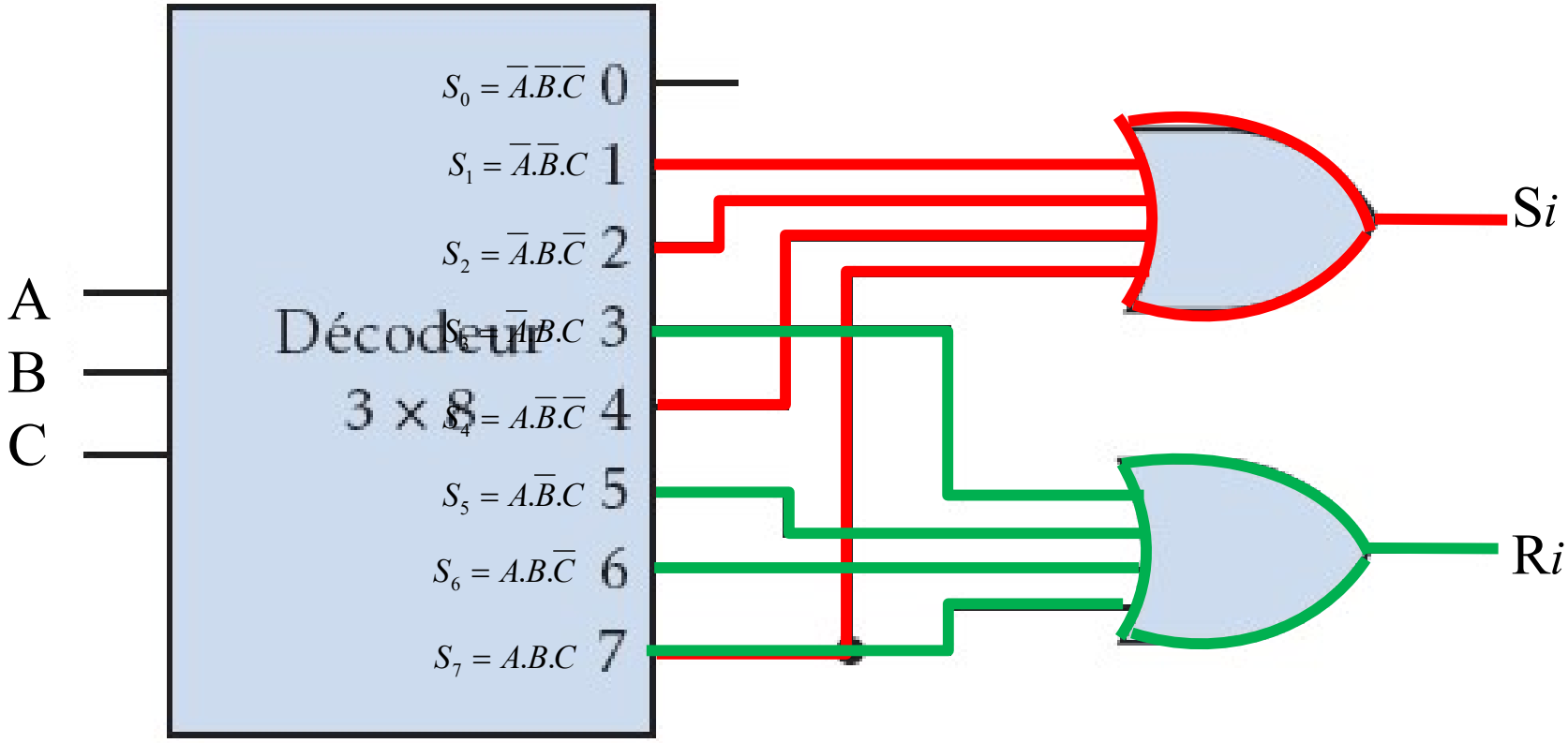
$$S_i = S1 + S2 + S4 + S7$$

Réalisation d'un additionneur complet avec des décodeurs binaire 3→8

On pose $A=A_i$, $B=B_i$, $C=R_{i-1}C$

Alors:

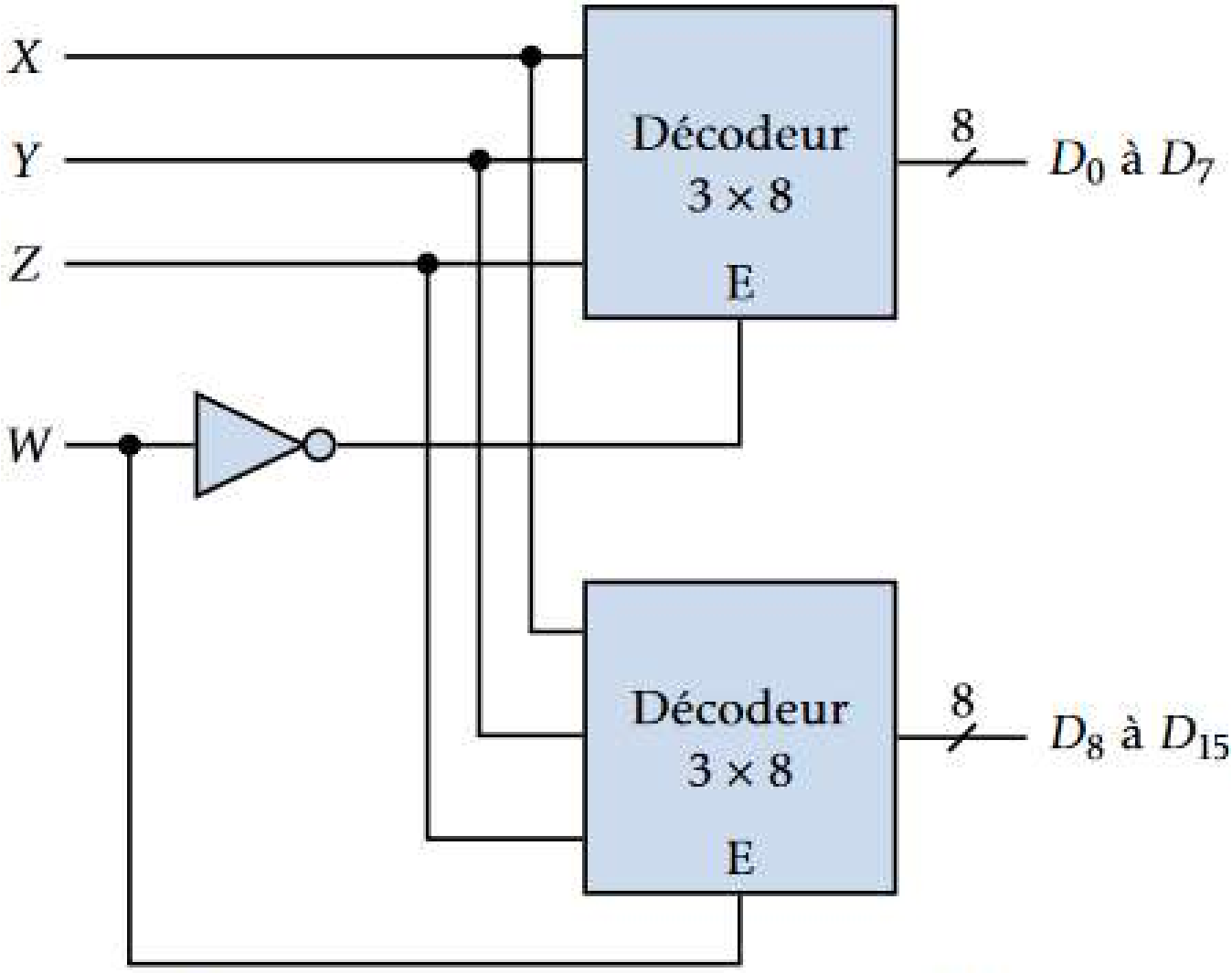
$$S_i = S1 + S2 + S4 + S7$$
$$R_i = S3 + S5 + S6 + S7$$



Synthèse de grands décodeurs binaire Exemple: 3→8

- Utiliser 2 décodeurs 3 × 8 pour faire un décodeur 4 × 16.
- Des décodeurs avec des entrées de validation peuvent être combinés pour créer des plus gros décodeurs.
- Par exemple, on peut utiliser 2 décodeurs 3 × 8 pour faire un décodeur 4 × 16.
- La quatrième variable est utilisée pour activer un ou l'autre des décodeurs 3 × 8.
- Dans la figure suivante:
- L'entrée W active seulement un des deux décodeurs à la fois.

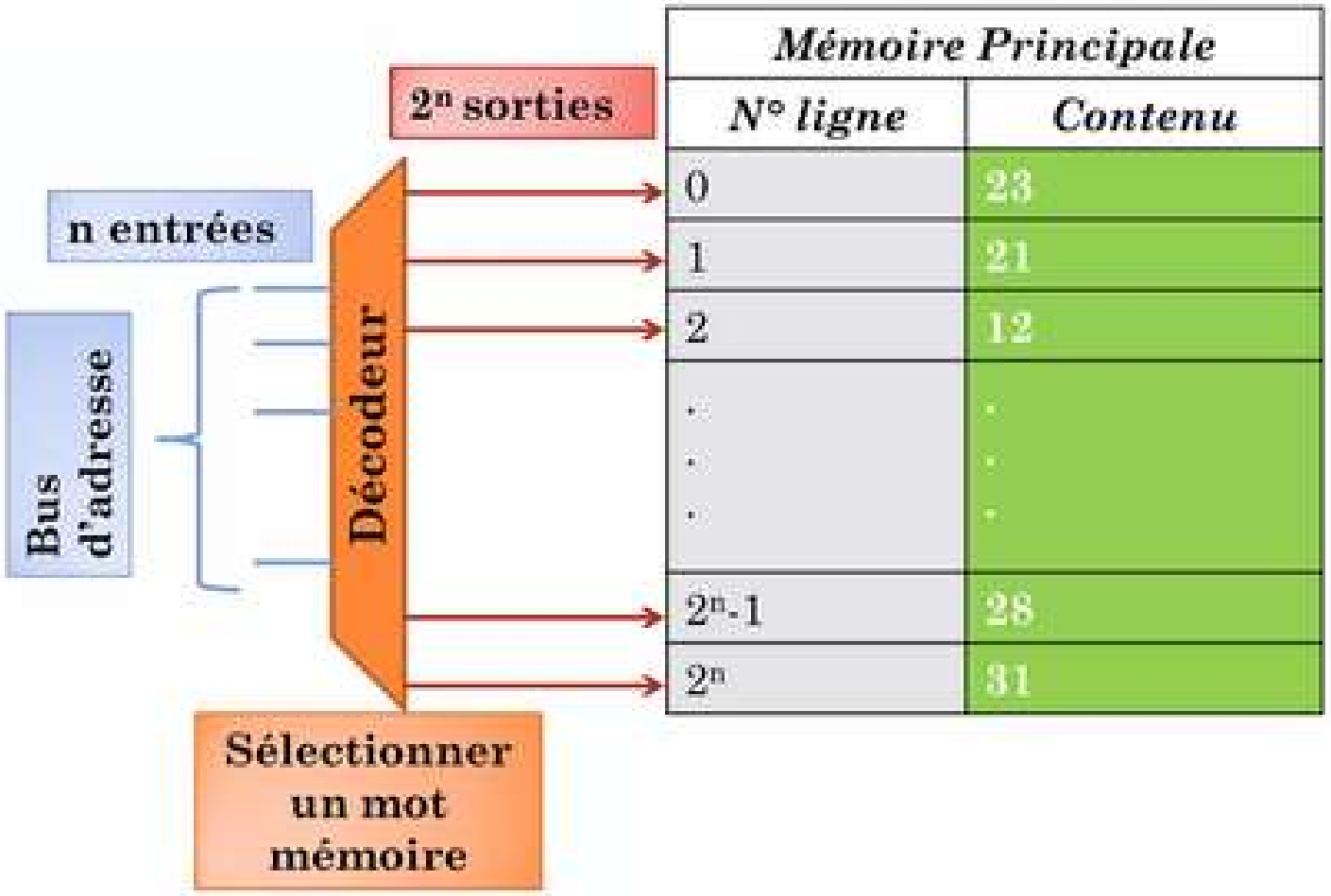
Synthèse de grands décodeurs binaire
Exemple: 3→8



– Décodeur 4 × 16 créé avec deux décodeurs 3 × 8

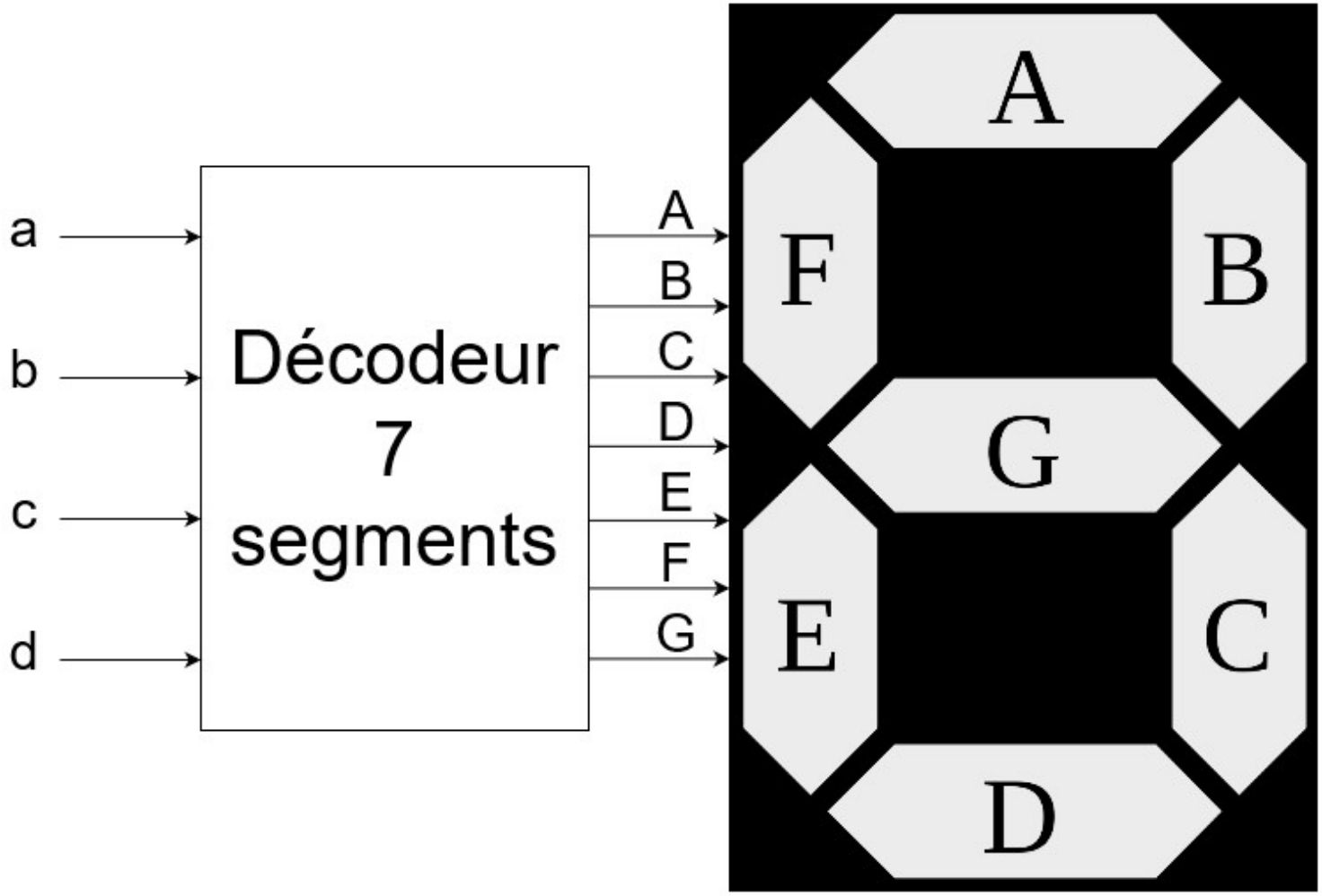
Exemple d'application de décodeur

- Le décodeur est un dispositif essentiel à l'entrée de la mémoire principale:



Exemple d'application de décodeur

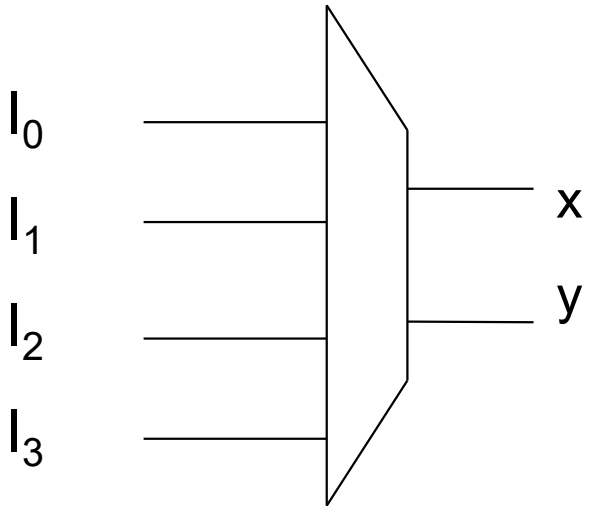
- Afficheur 7 segment:



8. L'encodeur binaire

- Il joue le rôle inverse d'un décodeur
  - Il possède 2^n entrées
  - N sortie
  - Pour chaque combinaison en entrée on va avoir sont numéro (en binaire) à la sortie.

Encodeur 4→2



8. L'encodeur binaire

- Exemple:
- La table de vérité d'un encodeur 8 à 3 est montré à la figure suivante:

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

– Table de vérité d'un encodeur 8 à 3

8. L'encodeur binaire

- Exemple:
- La table de vérité d'un encodeur 8 à 3 est montré à la figure suivante:

Les sorties sont obtenues avec des portes OU :

comme exemple, la sortie $A_0 = 1$

Lorsque les entrées 1, 3, 5 ou 7 sont 1.

On obtient alors les équations suivantes :

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = D_4 + D_5 + D_6 + D_7$

8. L'encodeur binaire

- On peut donc implémenter l'encodeur 8 à 3 avec trois portes OU de quatre entrées.

Remarque:

- Une seule entrée doit être activée à la fois, si non il y a erreur.
- Par exemple, si $D_3 = D_6 = 1$, la sortie sera: $A_2 = 1$, $A_1 = 1$ et $A_0 = 1$, ce qui voudrait dire que c'est l'entrée 7 qui est activée.

$$A_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$

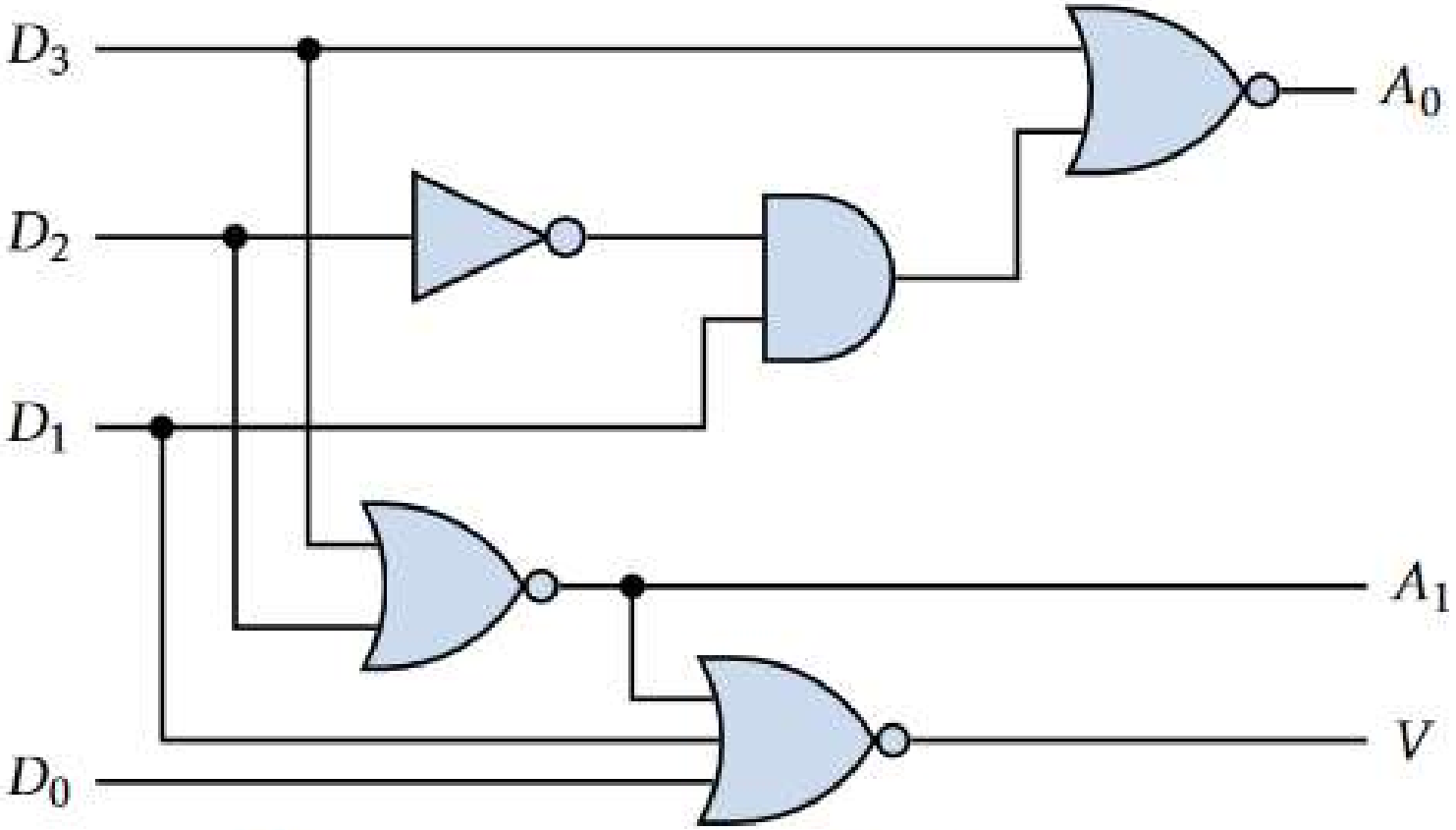
8. L'encodeur binaire

- Pour régler ce problème: on modifie l'encodeur pour que l'entrée la plus élevée ait la priorité : c'est un encodeur prioritaire. La table de vérité d'un encodeur 4 à 2 est montrée à la figure. Noter l'utilisation de conditions indifférentes. On a aussi une sortie de validation : V = 1 si une des entrées est 1, sinon V = 0.
  Le circuit est montré à la figure

D_0	D_1	D_2	D_3	A_1	A_0	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

8. L’encodeur binaire

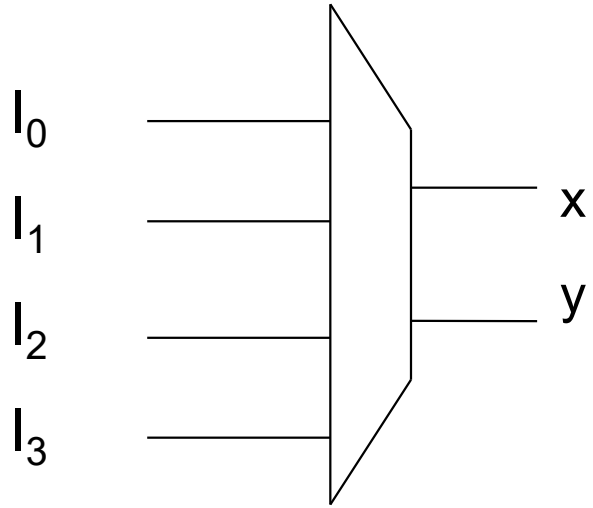
- Le circuit est montré à la figure



– Encodeur prioritaire 4 à 2

L'encodeur binaire (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



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

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

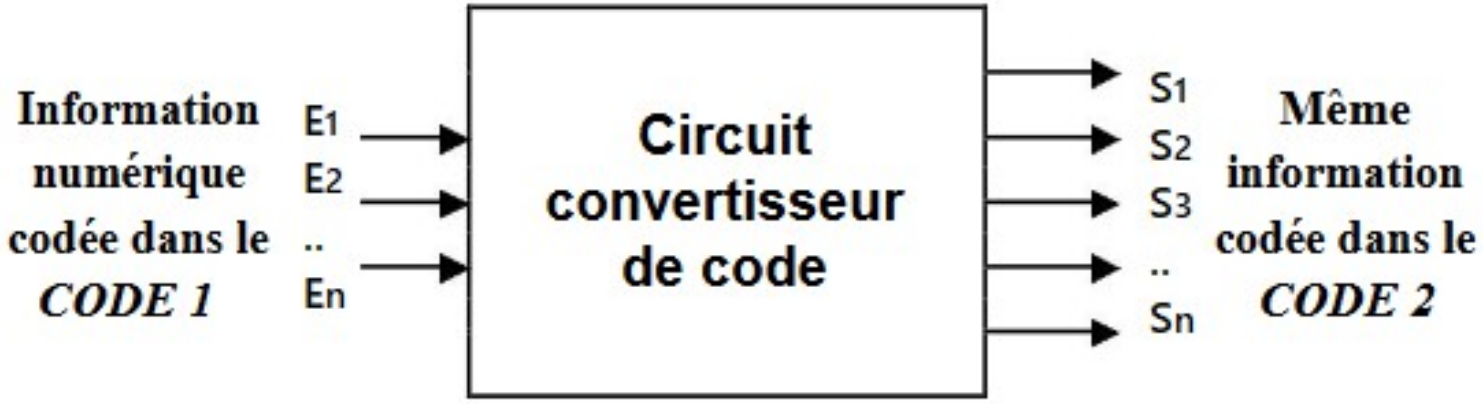
9. Le transcodeur

- Les circuits combinatoires de transcodage (appelés aussi convertisseurs de code),
- se répartissent en 3 catégories :
- Codeurs, décodeurs et transcodeurs
  Tous ces circuits logiques transforment une information présente à leurs entrées sous une forme donnée (code 1) en la même information présente à leurs sorties sous une forme différente (code 2).

9. Le transcodeur

- On appelle:

- Codeur: un circuit à 2^n entrées et n sorties
- Décodeur: un circuit à n entrées et 2^n sorties dont une seule est validée à la fois
- Transcodeur: tout autre circuit convertisseur de code différent des précédents, à **P** entrées et à **K** sorties.



9. Le transcodeur

- **Exemple**
- La réalisation d'un transcodeur :
- on désire réaliser un transcodage du code BCD vers le code à excès de trois.
- Les nombres d'entrée et de sortie sont exprimés sur 4 bits, et ce transcodeur pourra convertir tous les chiffres de 0 à 9.
- **Solution :**
- Etape n°1 de la conception du transcodeur : Ecriture de la table de vérité :

9. Le transcodeur

Chiffre converti	Entrées (BCD)				Sorties (XS 3)			
	E_3	E_2	E_1	E_0	S_3	S_2	S_1	S_0

9. Le transcodeur

- Etape n°2 de la conception du transcodeur :
- Recherche et simplification des équations des sorties : S0=....., S1=....., S2=....., S3=.....
- **Remarque :** parmi les 16 combinaisons possibles applicables sur les 4 entrées du transcodeur, seules 10 combinaisons seront utilisées (pour coder les 10 chiffres à convertir). Les 6 autres ne seront jamais présentes à l'entrée du transcodeur. Des croix apparaissent alors dans 6 cases des tableaux de Karnaugh des sorties, ce qui permet de simplifier considérablement les équations logiques.

9. Le transcodeur

- Etape n°3 de la conception du transcodeur : Dessin du logigramme
- **Remarque :**
- Parmi les transcodeurs que l'on trouve en circuits intégrés :
  Les transcodeurs décimal / BCD (circuit 74147)
  Les transcodeurs BCD / décimal (circuits 7442, 7445, et 4028)
  Les transcodeurs XS 3 / décimal (circuit 7443)
  Les transcodeurs Gray excédant 3 (code Gray+3) / décimal (circuit 7444)
  Les transcodeurs DCB / afficheur 7 segments (circuits 7448, 7511, 4543, 4511)
  Les transcodeurs binaire 5 bits / DCB (circuit 74185)
  Les transcodeurs DCB / binaire 5 bits (circuit 74184)

Exemple : Transcodeur BCD/EXESS3

A	B	C	D		X	Y	Z	T
0	0	0	0		0	0	1	1
0	0	0	1		0	1	0	0
0	0	1	0		0	1	0	1
0	0	1	1		0	1	1	0
0	1	0	0		0	1	1	1
0	1	0	1		1	0	0	0
0	1	1	0		1	0	0	1
0	1	1	1		1	0	1	0
1	0	0	0		1	0	1	1
1	0	0	1		1	1	0	0
1	0	1	0		x	x	x	x
1	0	1	1		x	x	x	x
1	1	0	0		x	x	x	x
1	1	0	1		x	x	x	x
1	1	1	0		x	x	x	x
1	1	1	1		x	x	x	x