

# **SEN1 - Electronique Numérique**

## **Circuits combinatoires**

1

L1 Portail IE

Johanne Bézy

# Introduction / Plan

Définition : dans les schémas de « logique combinatoire », la **sortie** ne dépend que des **états des entrées au même instant. Il n'y a pas d'effet mémoire.**

Dans les systèmes numériques, les données codées en binaire sont soumises à un certain nombre d'opérations qui sont réalisées grâce à des circuits intégrés.

- ▶ 1. Circuits arithmétiques (additionneurs binaires/DCB)
- ▶ 2. Aiguillage d'information (multiplexeurs/démultiplexeurs)
- ▶ 3. Transformation de codes (codeurs/décodeurs)
- ▶ 4. Comparateurs

The background features a minimalist design with several thin, curved lines in shades of brown, beige, and grey. A prominent red arrow points from the bottom left towards the center. Inside the red arrow is the number '3'.

3

# Circuits combinatoires

1. Circuits arithmétiques

Université de Rennes 1 - L1 IE  
UE SEN1 - J. BEZY-WENDLING

# 1.1. Additionneur binaire

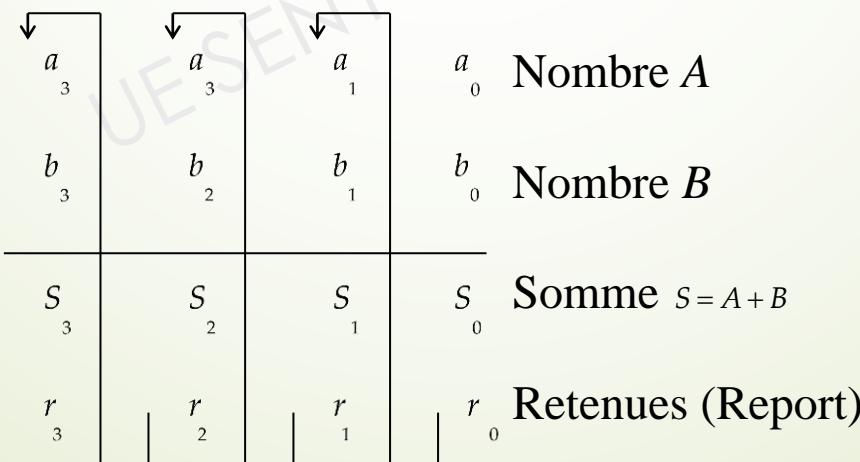
L'un des rôles essentiels des ordinateurs est d'effectuer des **opérations** arithmétiques : dans l'unité arithmétique où se trouvent des portes logiques et des bascules combinées de manière à additionner, soustraire, multiplier et diviser des nombres binaires.

## a) Addition et soustraction de deux nombres binaires non signés

Les ordinateurs ne peuvent additionner que deux nombres binaires à la fois, chacun de ces nombres binaires pouvant avoir plusieurs bits. En base 2, l'addition de deux bits s'écrit :

$$\left\{ \begin{array}{l} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{array} \right.$$

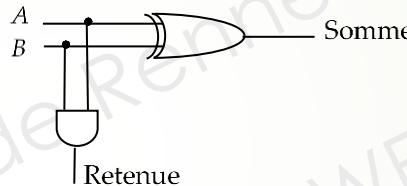
Comme en décimal, il faut tenir compte d'une éventuelle retenue. L'addition de deux nombres de 4 bits se fait de la façon suivante :



## b) Demi-additionneur

Un demi-additionneur permet d'additionner 2 bits. Il a deux sorties : la somme et la retenue. Sa table de vérité est :

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



D'après la table de vérité, on peut écrire l'expression de la somme S et de la retenue E :

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

$$R = A \cdot B$$

### c) Additionneur complet

Pour pouvoir additionner des nombres à plusieurs bits, il faut à chaque rang tenir compte de la retenue des bits de poids inférieur. Un circuit additionneur doit donc comporter 3 entrées et 2 sorties. La table de vérité d'un tel circuit est :

$A$	$B$	$R_e$	$S$	$R_s$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- A partir de cette table de vérité, nous pouvons remplir les tableaux de Karnaugh et simplifier les expressions de S et de Rs.

► S :

AB Re	00	01	11	10
0	0	1	0	1
1	1	0	1	0

► Rs :

AB Re	00	01	11	10
0	0	0	1	0
1	0	1	1	1

►  $S = A \oplus B \oplus Re$

►  $Rs = A \cdot B + A \cdot Re + B \cdot Re$

**$Rs = A \cdot B + Re \cdot (A + B)$**

Ou encore :

**$Rs = A \cdot B + Re(A \oplus B)$**

*Rm : cette deuxième expression permet de récupérer le  $A \oplus B$  de S pour le câblage du montage*

## Exercice d'application n°1

A	B	$R_e$	S	$R_s$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- A partir de la forme canonique Somme de Produits de S et des propriétés de l'algèbre de Boole, démontrer sa forme simplifiée obtenue précédemment.

démontrer sa forme simplifiée obtenue précédemment.

$$S = A \oplus B \oplus R_e$$

$$\begin{aligned}
 S &= \bar{A} \bar{B} R_e + \bar{A} B \bar{R}_e + A \bar{B} \bar{R}_e + A B R_e \\
 &= \overline{R_e} (\bar{A} B + A \bar{B}) + R_e (\bar{A} \bar{B} + A B) \\
 &= \overline{R_e} (A \oplus B) + R_e (\overline{A \oplus B}) \\
 &= R_e \oplus (A \oplus B) = R_e \oplus A \oplus B
 \end{aligned}$$

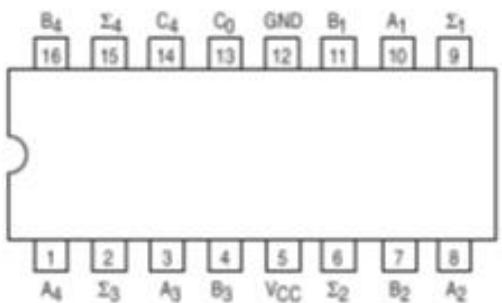
# Circuit intégré Additionneur binaire 4 bits (CI 7483 ou 5483)



## 4-BIT BINARY FULL ADDER WITH FAST CARRY

The SN54/74LS83A is a high-speed 4-Bit binary Full Adder with internal carry lookahead. It accepts two 4-bit binary words ( $A_1-A_4, B_1-B_4$ ) and a Carry Input ( $C_0$ ). It generates the binary Sum outputs ( $\Sigma_1-\Sigma_4$ ) and the Carry Output ( $C_4$ ) from the most significant bit. The LS83A operates with either active HIGH or active LOW operands (positive or negative logic). The SN54/74LS283 is recommended for new designs since it is identical in function with this device and features standard corner power pins.

CONNECTION DIAGRAM DIP (TOP VIEW)



**NOTE:**  
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

### PIN NAMES

	LOADING (Note a)	
	HIGH	LOW
$A_1-A_4$	Operand A Inputs	1.0 U.L.
$B_1-B_4$	Operand B Inputs	1.0 U.L.
$C_0$	Carry Input	0.5 U.L.
$\Sigma_1-\Sigma_4$	Sum Outputs (Note b)	10 U.L. 5 (2.5) U.L.
$C_4$	Carry Output (Note b)	10 U.L. 5 (2.5) U.L.

### NOTES:

a) 1 TTL Unit Load (U.L.) = 40  $\mu$ A HIGH/1.6 mA LOW.

b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

SN54/74LS83A

4-BIT BINARY FULL ADDER  
WITH FAST CARRY  
LOW POWER SCHOTTKY



J SUFFIX  
CERAMIC  
CASE 620-09



N SUFFIX  
PLASTIC  
CASE 648-08

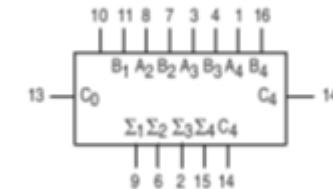


D SUFFIX  
SOIC  
CASE 751B-03

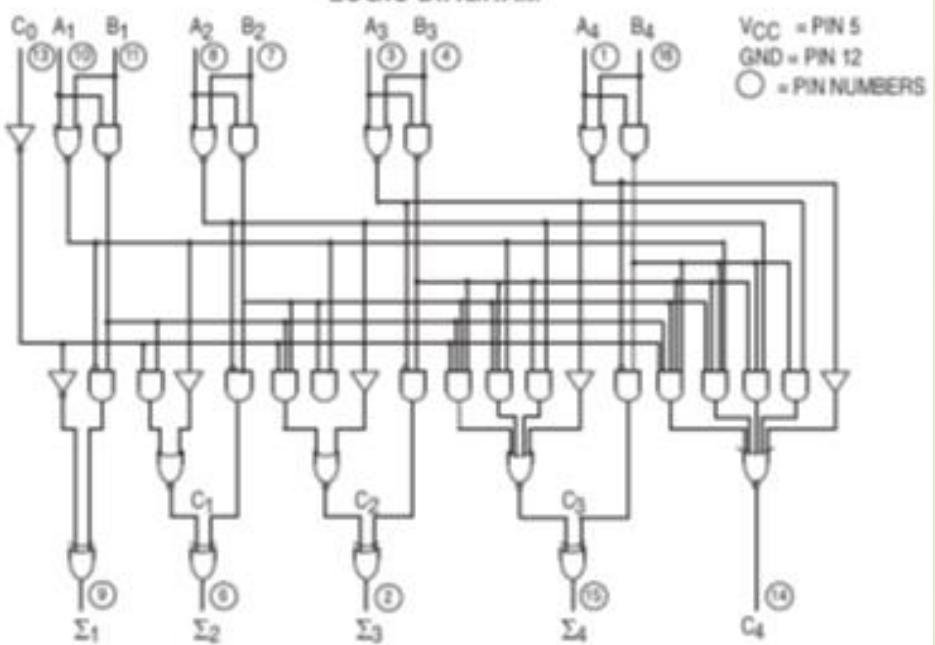
### ORDERING INFORMATION

SN54LSXXJ Ceramic  
SN74LSXX Plastic  
SN74LSXXD SOIC

### LOGIC SYMBOL



### LOGIC DIAGRAM



## 1.2. Additionneur-soustracteur de deux nombres de 8 bits représentés en Complément à 2, avec des CI 7483

### a) Rappel sur les notations

- ▶  $[x]_b$  interprétation du nombre  $x$ , en base  $b$
- ▶  $b[x]$  représentation du nombre  $x$ , dans la base  $b$
- ▶ Exemples :
  - $['100']_2 = 6$  (Interprétation du nombre binaire 100 : ce nombre vaut 6 ; sous entendu en base dix)
  - $2[6] = '100'$  (Représentation du nombre 6 en binaire : le nombre 6 est représenté par '100' en binaire)
- ▶  $2c[x]$  : c'est la représentation en complément à 2 du nombre ~~X~~ ~~X~~

## b) Rappel sur le complément à 2 : représentation d'un nombre

- La soustraction peut être transformée en addition et donc réalisée uniquement avec des additionneurs, grâce aux propriétés de la représentation des nombres en complément à 2 : la soustraction  $a-b$ , correspond à l'opération [ $a+2c[-b]$ ].

$$a-b=a+2c[-b]$$

*Rm : en complément à 2, la dernière retenue obtenue lors de l'addition (à l'issue de l'addition des bits de poids forts) est ignorée.*

- Représentation en complément à 2 d'un nombre positif, au format n bits :

- On représente un nombre positif par sa représentation en binaire en ajoutant devant l'amplitude un bit de signe égal à 0 (et éventuellement d'autres 0 pour atteindre le format désiré).
- Exemple : représentation de +5 sur 4 bits

$2[5] : 0101$

Représentation de +5 sur 4 bits

Représentation de +5 sur 8 bits

$2c[+5] : 0101$

$2c[+5] : 0000\ 0101$

► Représentation en complément à 2 d'un nombre négatif, au format n bits :

- ✓ On représente un nombre négatif en prenant le complément à deux (Cpt2) du nombre positif correspondant : pour cela on représente le nb positif sur n bits et on complémente à 2 après. On obtient alors un bit de signe égal à 1 (qui indique qu'on a affaire à un nombre négatif).
- ✓ Pour obtenir le Complément à 2, on prend le Complément à 1 (Cpt1) et on ajoute 1 au bit de poids le plus faible (avec propagation des retenues).
- ✓ Pour obtenir le Complément à 1 d'un nombre, on remplace chaque bit par son complément, un 0 est remplacé par un 1, et un 1 par un 0.
- ✓ Exemple : **représentation de -5 sur 4 bits**

2[5]=0101    **Représentation binaire sur 4 bits**  
1c[5]=1010  
2c[-5]=1010+1=1011

### c) Rappel sur le complément à 2 : interprétation d'un nombre

- On cherche à interpréter un nombre dont on sait qu'il est représenté en complément à 2 (trouver son signe et sa valeur absolue)
- Si le nombre est **positif** (bit de signe = 0) : pour obtenir l'équivalent décimal on fait la somme des bits (multipliés par leur poids).
  - Ex : 01100 est représenté en Cpt2 sur 6 bits : Bit de signe=0, donc nombre positif. Les quatre derniers chiffres représentent donc la grandeur exacte (ou valeur absolue) :  $+12_{(10)}$ .
  - Donc : **[01100]2c=+12**
- Si le nombre est **négatif** (bit de signe = 1) : on prend le Cpt2 pour obtenir le nombre positif correspondant (c'est-à-dire la valeur absolue), on calcule l'équivalent décimal et on le fait précédé du signe -.
- Ex : 101000 est un nombre représenté en Cpt2 sur 6 bits.

$$[101000]2c = ?$$

Bit de signe=1 : il est donc négatif.

$$\begin{array}{r}
 101000 \quad N \\
 010111 \quad \text{Cpt1 de } N \\
 +1 \\
 \hline
 011000 \quad \text{Cpt2 de } N = -N \\
 -N = 1 \times 2^4 + 1 \times 2^3 = 16 + 8 = 24 \quad \text{d'où} \quad N = -24
 \end{array}$$

- Donc : **[101000]2c=-24**

## d) Soustraction

- Pour faire la soustraction de deux nombres (opération A-B) :
  - On fait l'addition des deux nombres A, et (-B) représentés en complément à 2 :  $2c[A]+2c[-B]$
  - On ignore la dernière retenue
  
- Exemples d'additions et soustractions de nombres signés représentés en Cpt2 (Les nombres sont représentés en cpt2 sur 6 bits) :

$+13 \quad 001101$ $+11 \quad 001011$ <hr/> $+24 \quad 011000$	$+13 \quad 001101$ $-11 \quad 110101$ <hr/> $+2 \quad 000010$	$-13 \quad 110011$ $+11 \quad 001011$ <hr/> $-2 \quad 111110$	$-13 \quad 110011$ $-11 \quad 110101$ <hr/> $-24 \quad 101000$
--	---	---	--

- Interprétation des résultats :
  - Pour  $-13+11$  on trouve  $111110$  : nb négatif, donc pour avoir sa valeur absolue on doit en prendre le Cpt2 :  $000001+1=2$ .  $[111110]2c=-2$
  - Idem pour  $-13-11$  : on trouve  $101000$  : nb négatif, donc pour avoir sa valeur absolue on doit en prendre le Cpt2 :  $010111+1=24$ .  $[101000]2c=-24$

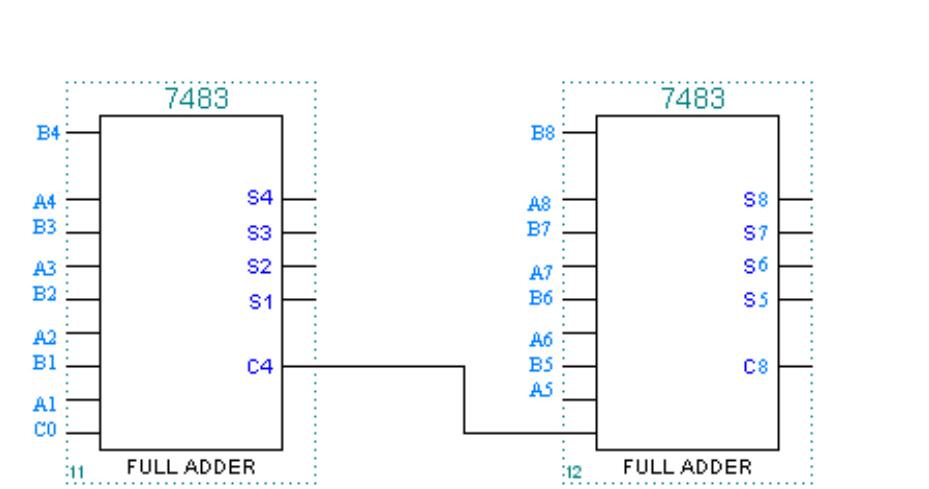
### e) Utilisation des CI pour réaliser un additionneur/soustracteur

- ▶ Les circuits 7483 sont des additionneurs 4bits.
- ▶ Un 7483 a 4 sorties sommes :  $S_1, S_2, S_3, S_4$  et une sortie retenue  $C_4$ .
  
- ▶ Pour additionner deux nombres de 8 bits chacun, on utilise deux circuits 7483.
- ▶ Supposons que l'on dispose d'un signal de contrôle M qui permet de réaliser :
  - soit l'addition de A et B ( $S=A+B$ ) si on met M à 0
  - soit la soustraction ( $S=B-A$ ) si on met M à 1
- ▶ Les deux nombres A et B sont considérés comme positifs et s'écrivent, en binaire :

$$A = A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1$$

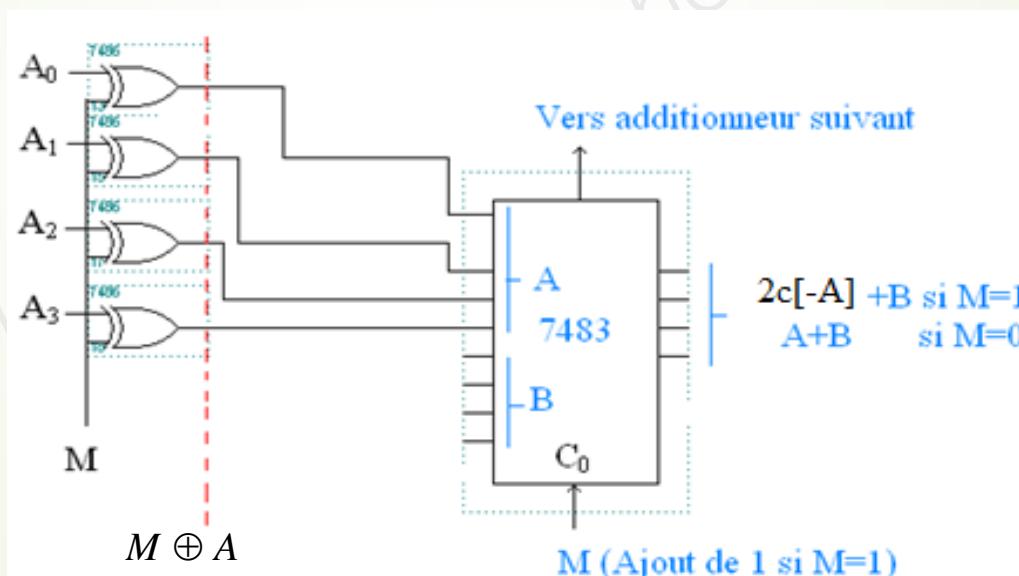
$$B = B_8 B_7 B_6 B_5 B_4 B_3 B_2 B_1$$

- ▶ On connecte deux additionneurs 7483 :



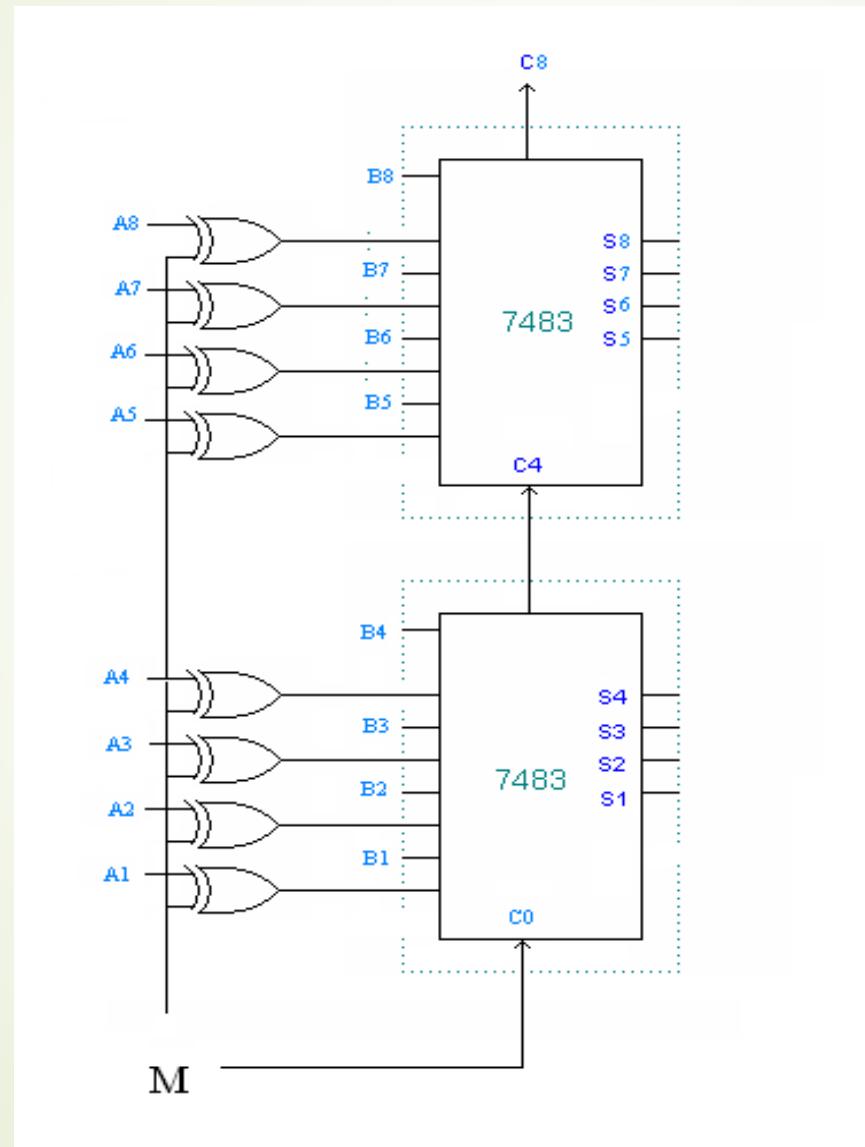
- Si on veut faire une soustraction ( $B-A$ ) on va ajouter à  $B$  la représentation en Complément à deux de  $(-A)$ .
- $2c[-A] = 1c[A]+1$

⇒ Pour faire  $B-A$  (quand  $M=1$ ), on ajoute à  $B$  le Cpt1 de  $A$  et une retenue  $C_0=1$



$$\begin{aligned}
 M \oplus A &= M \bar{A} + \bar{M}A \\
 &= Cpt_1(A) \text{ (si } M=1) \\
 &= A \text{ (si } M=0)
 \end{aligned}$$

## Schéma complet de l'additionneur soustracteur sur 8 bits



## 1.3. Additionneur DCB

### a) Rappel sur l'addition en DCB

- ▶ Ce code fait correspondre à chaque chiffre décimal un code 4 bits compris entre 0000 et 1001.
- ▶ Quand on additionne deux chiffres décimaux, 2 cas peuvent se produire .
- ▶ Cas 1 : la somme est inférieure ou égale à 9 :

- *Ex1* : additionnons 5 et 4 en utilisant leur représentation

$$\begin{array}{r} 5 \\ + 4 \\ \hline 9 \end{array}$$

0101 → DCB de 5  
 0100 → DCB de 4  
 1001 → DCB de 9

- *Ex2* : addition de 45 et 33

$$\begin{array}{r} 45 \\ + 33 \\ \hline \end{array}$$

0100	0101	→ DCB de 45
0011	0011	→ DCB de 33
<hr/>		
0111	1000	→ DCB de 78

Dans ces exemples, aucune somme de chiffre décimaux ( $5+4$ ,  $3+5$ ,  $3+4$ ) ne dépassait 9, donc il n'y a pas eu de report décimal.

► Cas 2 : la somme est supérieure à 9 :

- *Ex1* : additionnons 6 et 7 en DCB :

$$\begin{array}{r} 6 \\ +7 \\ \hline 13 \end{array}$$

0110	→DCB de 6
0111	→DCB de 7
<u>1101</u>	→Code invalide en DCB

Solution : il faut corriger la somme en additionnant 6 (0110) pour tenir compte du fait qu'on saute six représentations invalides

$$\begin{array}{r} 6 \\ +7 \\ \hline 13 \end{array}$$

0110	→DCB de 6
0111	→DCB de 7
<u>1101</u>	→Somme non valide
<u>0110</u>	→Additionner 6 pour corriger
<u>0001</u>	<u>0011</u> →DCB de 13
1	3

L'addition de 6 à une somme non valide donne une représentation DCB exacte.

- *Ex2* : addition de 47 et 35 en DCB :

$$\begin{array}{r} 47 \\ +35 \\ \hline 82 \end{array}$$

$$\begin{array}{r} 0100 \ 0111 \\ 0011 \ 0101 \\ \hline 0111 \ 1100 \\ \boxed{1} \ 0110 \\ \hline 1000 \ 0010 \end{array}$$

→ Somme non valide dans le premier chiffre

→ Addition de 6

→ Somme DCB exacte

$$8 \quad 2 \quad \rightarrow 82$$

La correction de la somme non valide provoque un report de 1, ne pas l'oublier au second rang !

- *Ex3* : Addition de 59 et 38 en DCB :

$$\begin{array}{r} 59 \\ +38 \\ \hline 97 \end{array}$$

$$\begin{array}{r} 0101 \ 1001 \\ 0011 \ 1000 \\ \hline 1001 \ 0001 \\ \quad 0110 \end{array}$$

→ Ajouter 6 pour corriger  
(car somme = 10001 = 17>9)

$$9 \quad 7$$

# Le schéma de l'additionneur DCB

► Résumé pour l'addition en DCB :

- Addition binaire ordinaire des représentations DCB de tous les rangs.
- Pour les rangs où la somme est inférieure ou égale à 9 la somme est valide.
- Pour les rangs où la somme est supérieure à 9 on ajoute 0110 pour obtenir une représentation exacte. Il se produit toujours un report sur le chiffre de rang immédiatement à gauche.

► L'additionneur doit être capable de :

- Additionner deux représentations DCB (de 4 bits 1 digit DCB) en suivant les règles de l'addition ordinaire.
- Tester la somme obtenue et détecter si elle est invalide (supérieure à 1001).
- Corriger les sommes invalides en ajoutant 6 (0110).

- La 1<sup>ère</sup> étape peut se faire à l'aide d'un CI 7483.
- Les sorties du 7483 sont  $C_4 S_4 S_3 S_2 S_1$
- En étudiant la table de vérité correspondant aux valeurs de ces 5 variables, identifions les cas où la fonction X est à 1, indiquant une représentation invalide.

valeur décim.	$C_4$	$S_4$	$S_3$	$S_2$	$S_1$	( $>9$ )
10	0	1	0	1	0	1
11	0	1	0	1	1	1
12	0	1	1	0	0	1
13	0	1	1	0	1	1
14	0	1	1	1	0	1
15	0	1	1	1	1	1
16	1	0	0	0	0	1
17	1	0	0	0	1	1
18	1	0	0	1	0	1

et X=0 dans les autres cas.

$S_4 S_3 \backslash S_2 S_1$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

→  $X = C_4 + \overline{C_4} \cdot I$

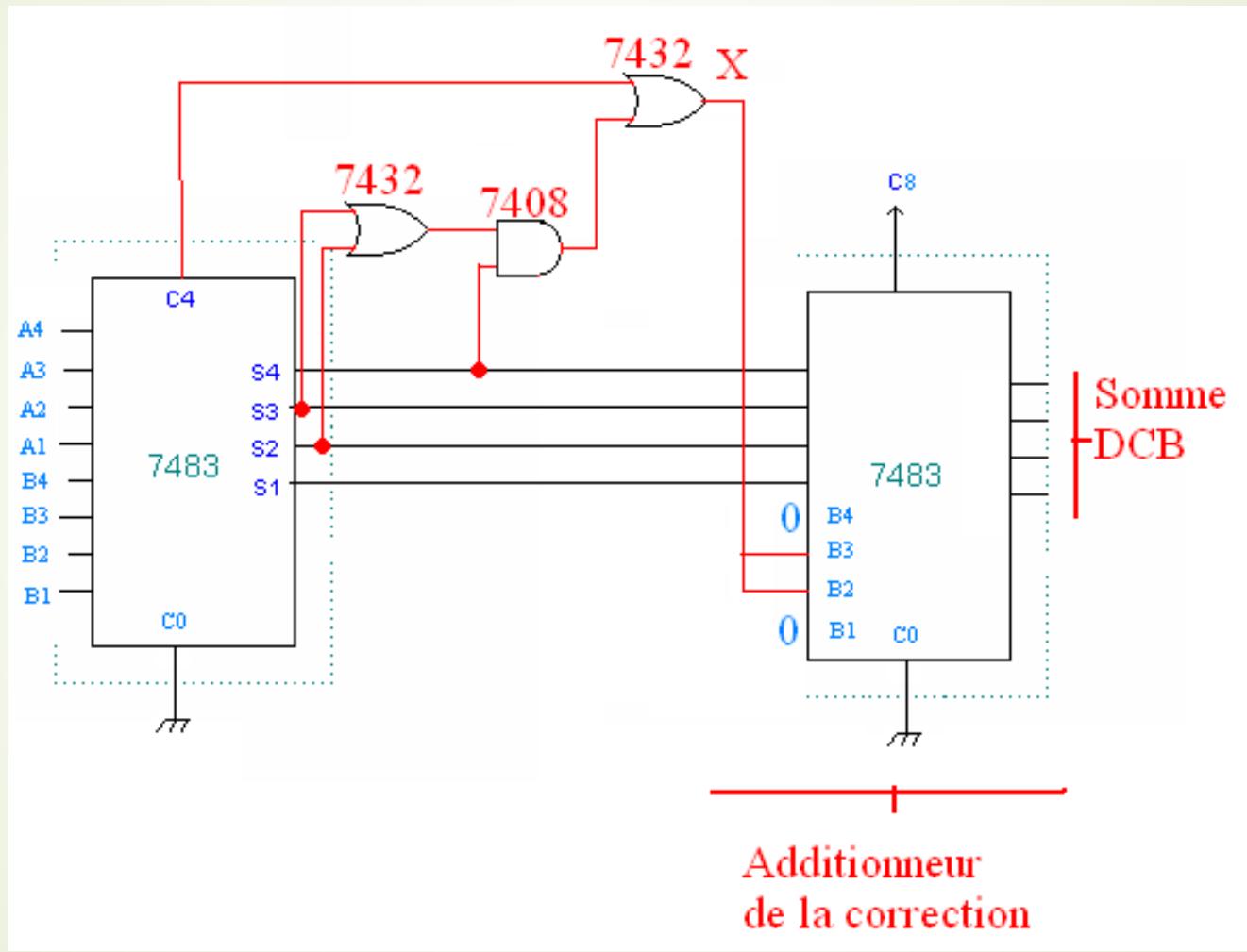
Avec  $I = S_4 S_3 + S_4 S_2$

$$X = C_4 + I$$

$$X = C_4 + S_4(S_3 + S_2)$$

Quand X=1, on doit ajouter 0110 à la somme obtenue en sortie du 7483.

# Schéma complet de l'additionneur DCB



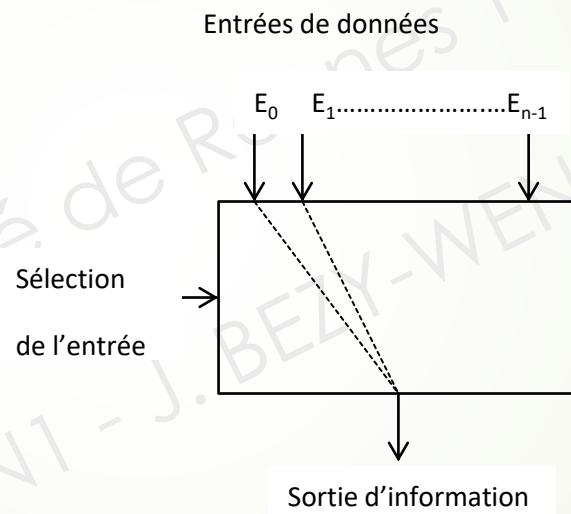
# Circuits combinatoires

2. Aiguillage d'information : Multiplexeurs / Démultiplexeurs

## 2.1. Multiplexeur

多路转换器

- ▶ Fonction : acheminer les informations numériques de plusieurs sources sur une seule ligne (destination commune)
- ▶ Plusieurs lignes de données d'entrée / plusieurs entrées de sélection de données /une seule ligne de sortie
- ▶ Principe et schéma :



- ▶ Fonction : la sortie S présente l'état de l'entrée  $E_i$  sélectionnée par l'adresse placée sur les voies de sélection (l'entrée  $E_i$  est orientée vers la sortie)

### a) Multiplexeur 2 vers 1

- 2 entrées de données
- 1 entrée d'adresse (sélection) / 1 sortie

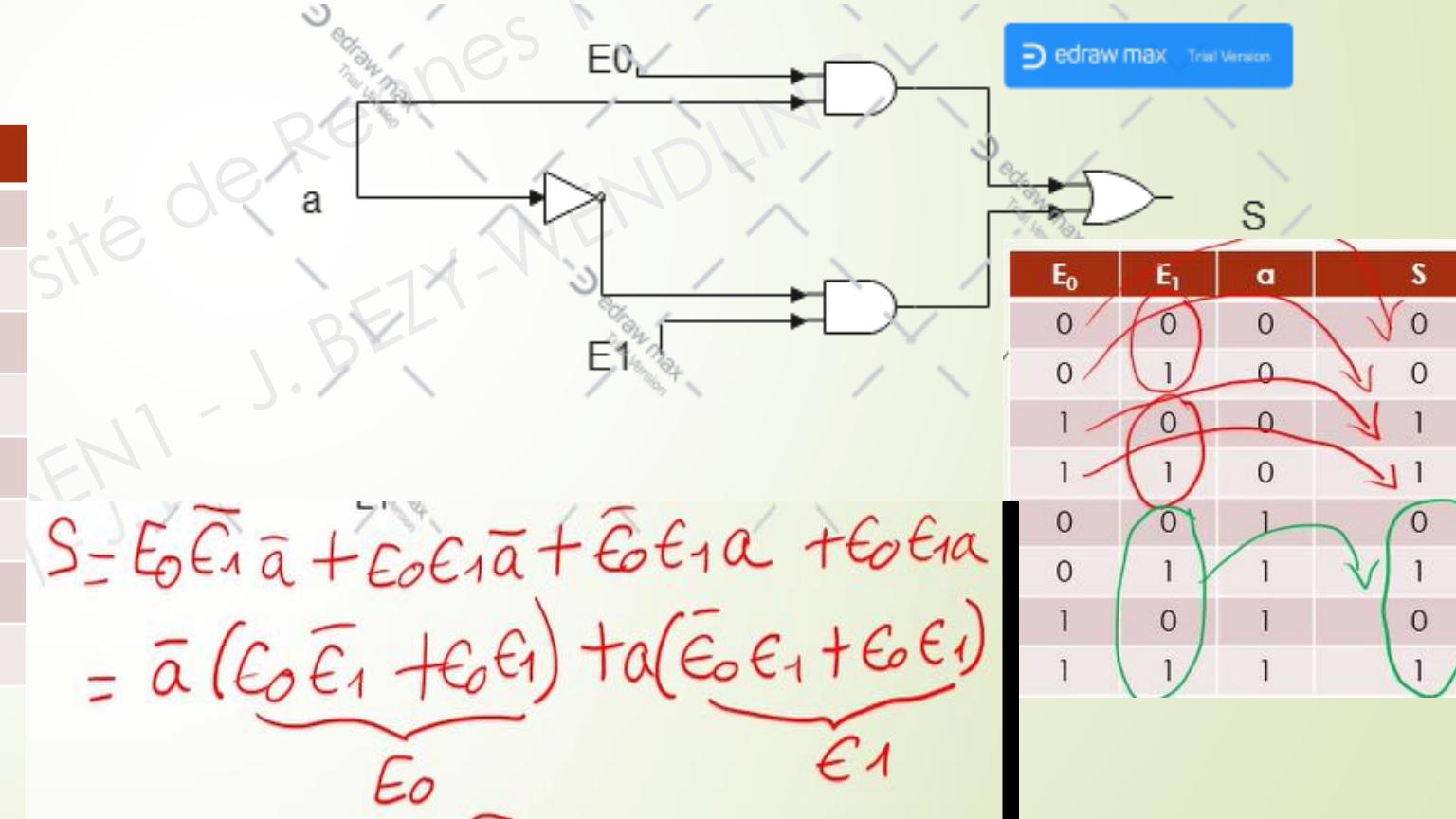
a	s
0	E0
1	E1

E <sub>0</sub>	E <sub>1</sub>	a	s
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

- Fonction de sortie :

$$S = \bar{a} E0 + aE1$$

- Schéma du MUX2 avec des portes logiques

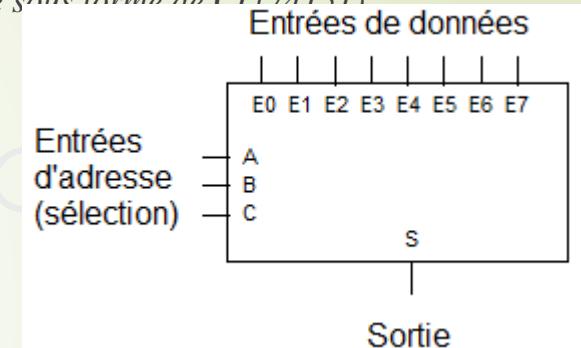


## b) Multiplexeur 8 vers 1

- 8 entrées commutable vers la sortie / 3 entrées de sélection de données / une seule ligne de sortie
- Table de vérité :

Adresse de sélection	Sortie
CBA	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

Rm : existe sous forme de CL74151



- Il faut indiquer au multiplexeur **quelle entrée parmi les huit doit être "reliée" à la sortie**. Cela se fait au moyen de trois entrées de sélection. Le nombre binaire appliqué à ces entrées indique l'entrée sélectionnée.
- A chaque entrée de donnée est associée un nombre compris entre 0 et 7. Il suffit d'appliquer aux entrées de sélection **le nombre binaire correspondant à l'entrée sélectionnée** pour que celle-ci soit commutée vers la sortie.
- Par exemple, l'entrée 5 est sélectionnée avec le nombre 101, l'entrée 6 avec le nombre 110 ..etc
- Le circuit intégré qui présente ces caractéristiques est le 74C151.
- Expression de la sortie du multiplexeur :**

$$S = \bar{C}\bar{B}\bar{A}E_0 + \bar{C}\bar{B}AE_1 + \bar{C}B\bar{A}E_2 + \bar{C}BAE_3 + C\bar{B}\bar{A}E_4 + C\bar{B}AE_5 + CB\bar{A}E_6 + CBAE_7$$

### c) Application du MUX : réalisation d'une fonction logique quelconque

- Un MUX peut être utilisé pour réaliser une fonction logique quelconque (Exemple avec un MUX 4)
- *Rappel* : équation logique d'un MUX 4 entrées  $E_0, E_1, E_2, E_3$  (avec donc 2 variables de sélection  $B, A$ ) :

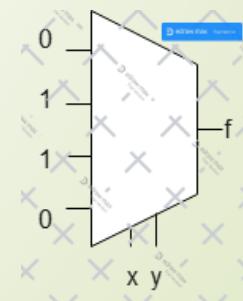
$$S = \bar{B}\bar{A}E_0 + \bar{B}AE_1 + B\bar{A}E_2 + BAE_3$$

- **1<sup>er</sup> cas** : Réalisation d'une fonction à n variables avec un MUX à n variables d'adresses : dans ce cas, on place les n variables de la fonction sur les n lignes d'adresse, et il s'agit ensuite de placer 0 ou 1 sur chaque entrée de donnée.

- Exemple : soit la fonction  $f = \bar{x}y + x\bar{y}$
- On met x sur l'entrée d'adresse B et y sur l'entrée d'adresse A (par exemple).
- Puis on identifie la fonction ainsi obtenue en sortie du multiplexeur (forme générale de la sortie, S), avec la fonction que l'on veut générer (celle qui nous intéresse, f) :

$$\begin{cases} S = \bar{x}\bar{y}E_0 + \bar{x}yE_1 + x\bar{y}E_2 + xyE_3 \\ f = \bar{x}y + x\bar{y} \end{cases}$$

- On déduit qu'il faut :  $E_0=0, E_1=1, E_2=1, E_3=0$

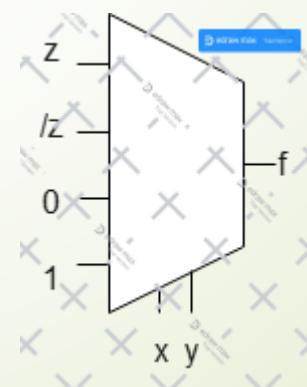


► **2<sup>ème</sup> cas** : Réalisation d'une fonction à  $n+1$  variables avec un MUX à  $n$  variables d'adresses :

- Par exemple, une fonction à 3 variables avec un MUX 4 vers 1 qui a donc 2 lignes d'adresse, ou une fonction à 4 variables avec un MUX 8 vers 1 qui a donc 3 lignes d'adresse.
- **Ex1** : réalisation de la fonction  $g$  à 3 variables :  $g = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy$
- On commence par placer 2 des 3 variables sur les lignes d'adresse (choix arbitraire : par exemple  $x$  sur l'entrée d'adresse B et  $y$  sur l'entrée d'adresse A).
- On cherche ensuite ce qu'il faut mettre sur les entrées d'adresse (0,1 ou la troisième variable, normale ou complémentée).

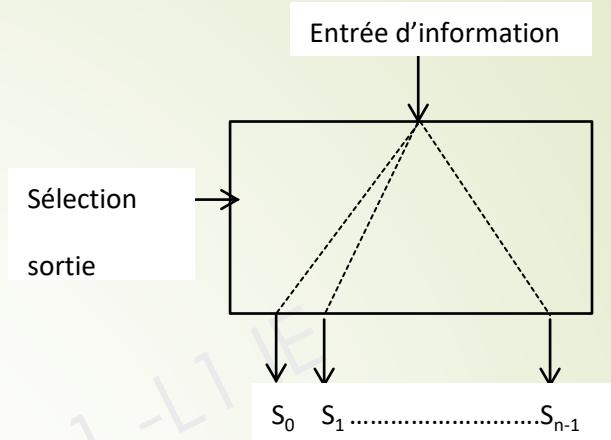
$$\begin{cases} S = \bar{x}\bar{y}E_0 + \bar{x}yE_1 + x\bar{y}E_2 + xyE_3 \\ g = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy \end{cases}$$

- On déduit qu'il faut :  $E_0=z$ ,  $E_1=\bar{z}$ ,  $E_2=0$ ,  $E_3=1$



## 2.2. Démultiplexeur

多路复用器

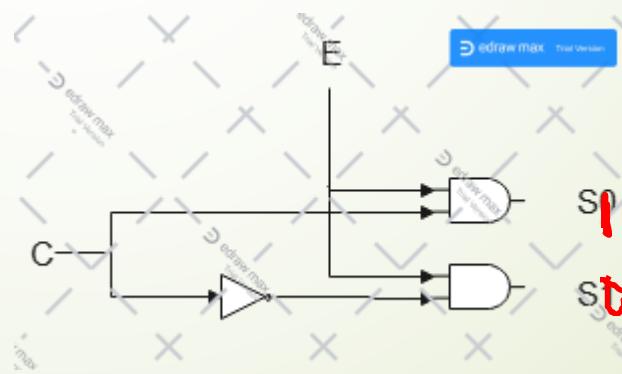


- ▶ L'information E d'entrée est aiguillée vers une sortie Si choisie parmi un groupe de sorties, grâce à une adresse (sélection) CBA
- ▶ Si=E quand  $CBA_2=i_{10}$

Adresse de sélection	Sortie d'un Demux 1 vers 8
CBA	S <sub>0</sub> S <sub>1</sub> S <sub>2</sub> S <sub>3</sub> S <sub>4</sub> S <sub>5</sub> S <sub>6</sub> S <sub>7</sub>
0 0 0	E X X X X X X X
0 0 1	X E X X X X X X
0 1 0	X X E X X X X X
0 1 1	X X X E X X X X
1 0 0	X X X X E X X X
1 0 1	X X X X X E X X
1 1 0	X X X X X X E X
1 1 1	X X X X X X X E

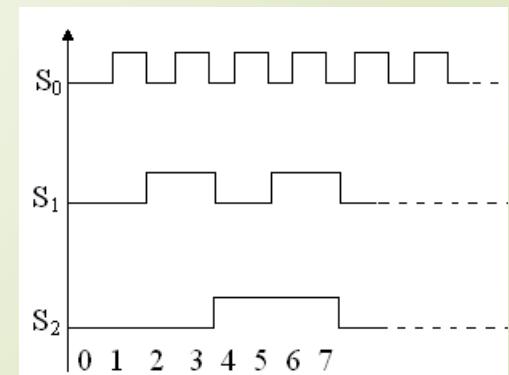
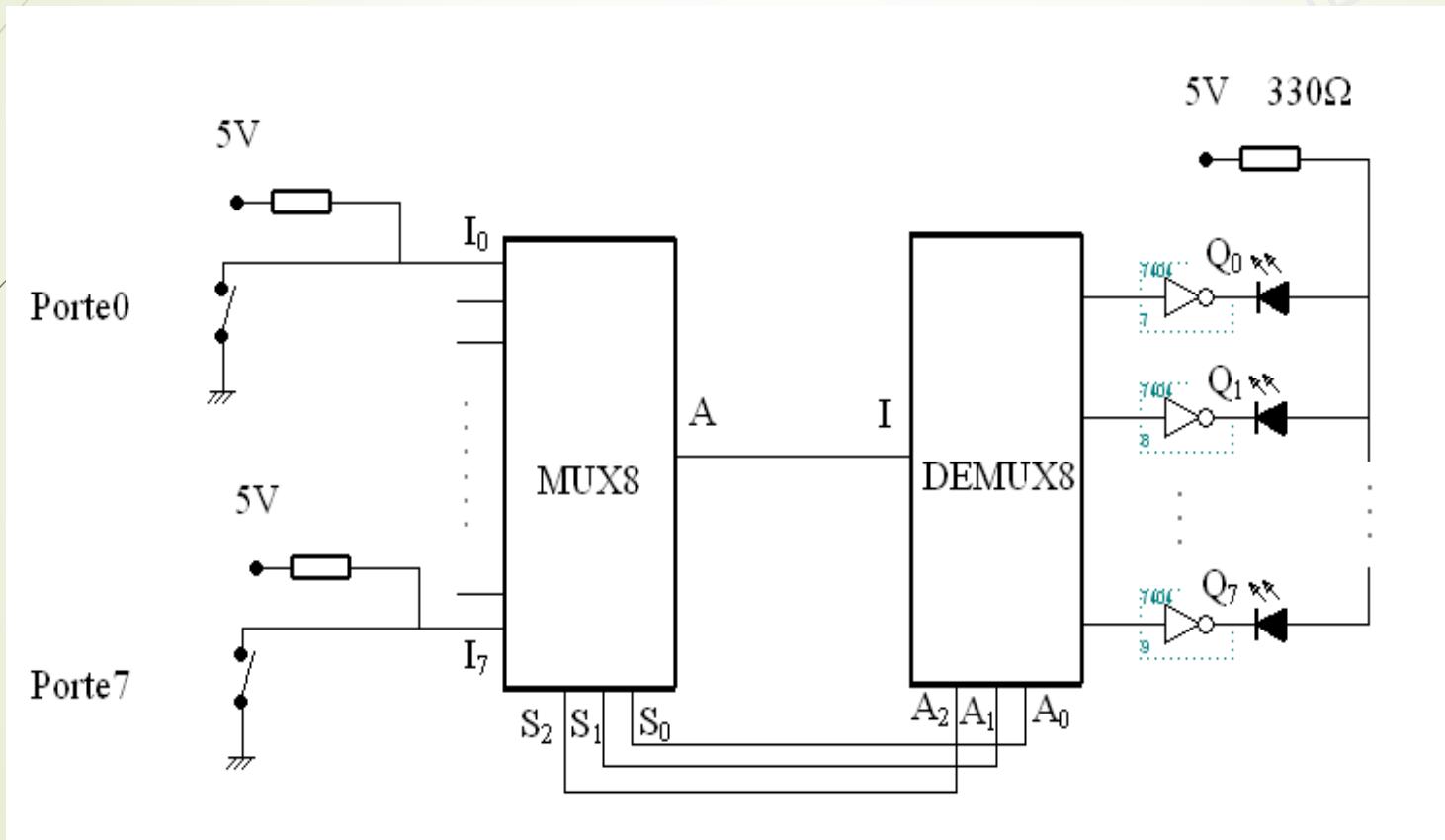
### a) Démultiplexeur 1 vers 2

- Un DEMUX 2 sorties a une entrée de sélection (C) qui permet d'orienter la donnée présente à l'entrée E, vers l'une ou l'autre des deux sorties.
- Si  $C=0 \Rightarrow$  l'entrée est orientée vers la sortie  $S_0 : S_0=E$
- Si  $C=1 \Rightarrow$  l'entrée est orientée vers la sortie  $S_1 : S_1=E$
- Le schéma d'un DEMUX 2 est :



## 2.3. Exemple d'application MUX+DEMUX

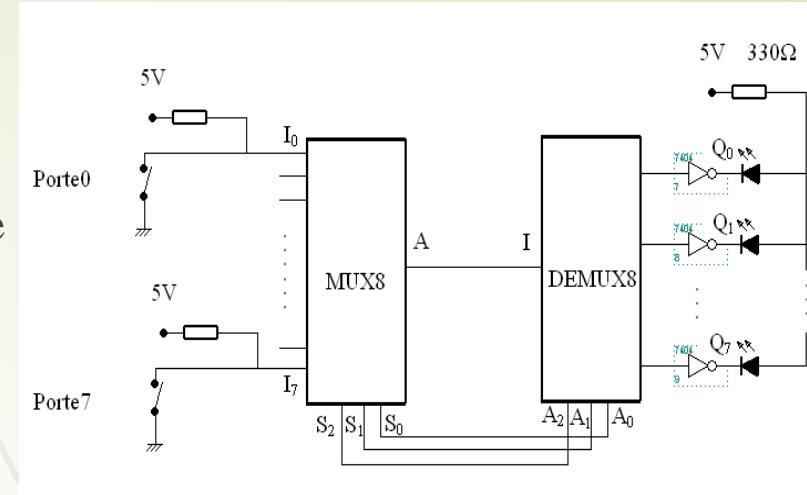
- Il s'agit d'un système de surveillance à distance permettant de détecter si une porte est été ouverte.



## Fonctionnement du système de surveillance :

- ▶ Chaque porte est associée à une DEL de la console de surveillance
- ▶ Porte ouverte :  $I=1$  (Niveau Haut) – Porte fermée  $I=0$
- ▶ Le compteur  $S_2 S_1 S_0$  parcourt les 8 états possibles de 000 à 111
- ▶ Pour chaque valeur  $n$  du compteur, l'entrée  $n$  est transmise à  $A$ .
- ▶ DEL : allumée si la sortie  $Q$  est au niveau bas (0).
- ▶ Ex : compteur = 110(6)
  - Si la porte 6 est fermée : niveau bas sur  $I_6$ , donc 0 sur  $A$ , donc 0 sur  $I$ , donc 1 sur  $Q_6$  et la LED ne s'allume pas
  - Si la porte 6 est ouverte : niveau haut sur  $I_6$ , donc 1 sur  $A$ , donc 1 sur  $I$ , donc 0 sur  $Q_6$   
➔ LED éclaire
  - Pendant que la porte 6 est sélectionnée, toutes les autres LED sont éteintes car seule la sortie  $Q_6$  est active (fonctionnement du DEMUX)

⇒ Une porte ouverte est signalée par un voyant qui clignote



# Circuits combinatoires

3. Transformation de code : Décodeurs / (En)codeurs

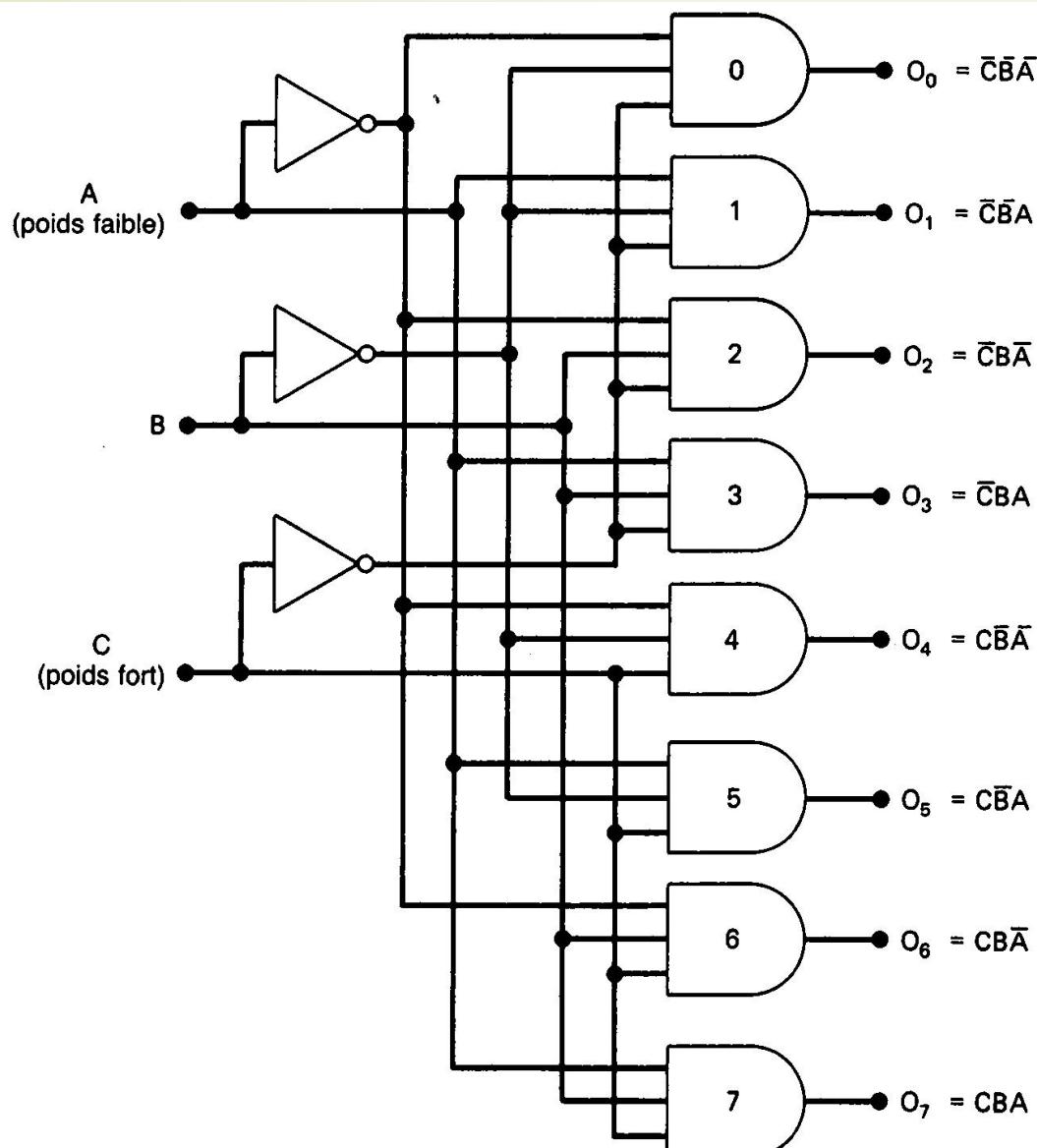
# Généralités

- ▶ Plusieurs façons de représenter l'information codée / Passer d'un code à l'autre
- ▶ Circuits Intégrés pour codes courants
- ▶ Exemples :
  - a) Circuits qui décodent une combinaison codée en binaire pur, en DCB, en code Gray.. en une combinaison « 1 parmi N »
    - Une seule sortie du circuit active à la fois (en fonction du code présent en entrée)
    - Appelés « **Décodeurs** »
  - b) Circuits opération inverse : 1 code « 1 parmi N » (une seule entrée active parmi N) transformée en code binaire
    - Valeur code binaire en sortie = rang de l'entrée active (une seule)
    - Appelés « **codeurs** » ou « **encodeurs** »
    - Si plusieurs entrées actives : affectation d'une priorité (encodeur de priorité)
  - c) Circuits transforment un code DCB en code n parmi 7 pour représenter visuellement les chiffres de 0 à 9 (« décodeur 7 segments »)

## 3.1. Décodeurs

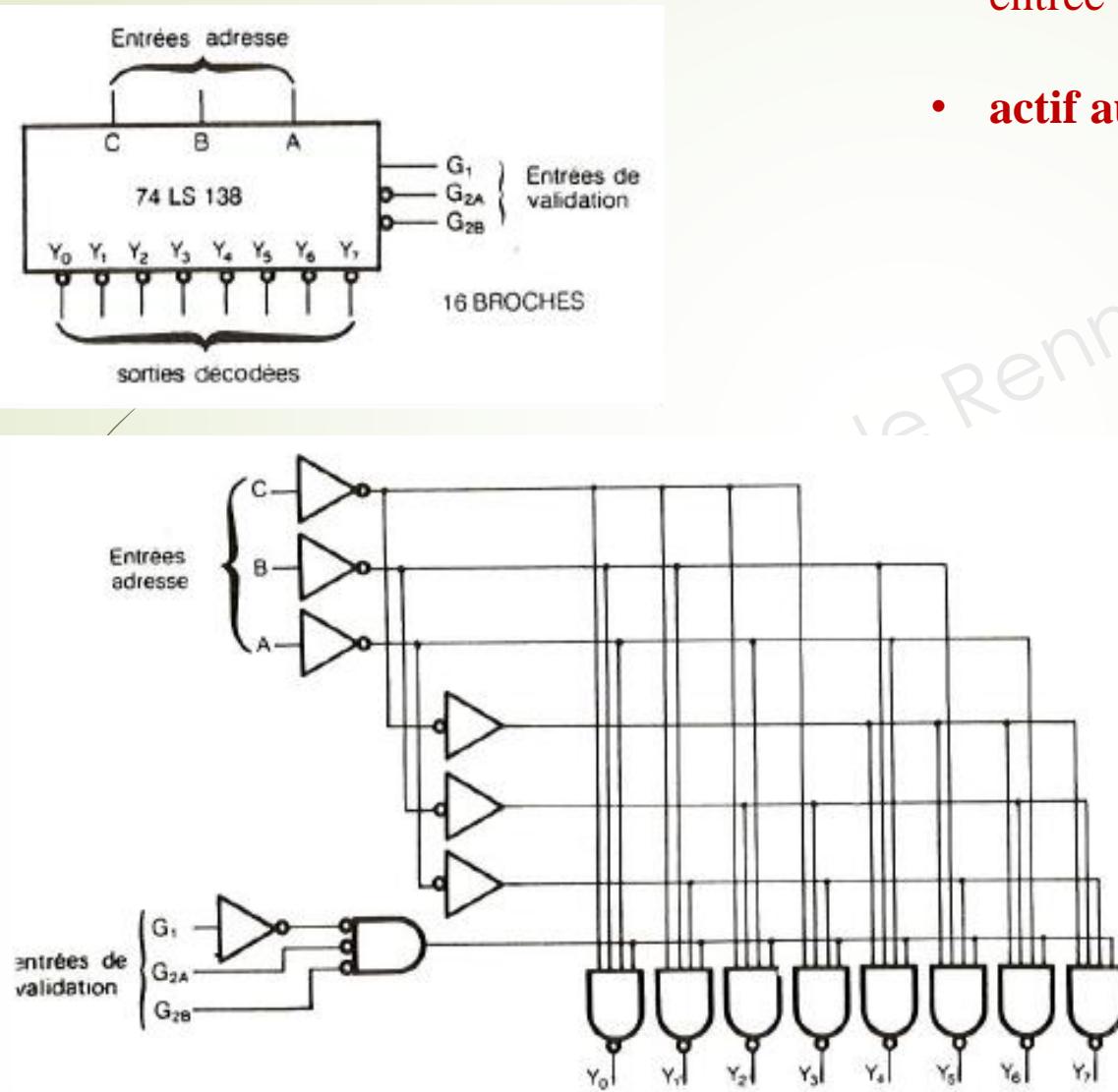
### a) Fonction de décodage binaire

- ▶ Transformer un codage binaire pur en codage 1 parmi N.
- ▶ En entrée : un code sur n bits ( $N=2^n$  codes possibles) et une entrée de validation
- ▶ En sortie : une seule sortie active parmi les N sorties.
- ▶ Entrée(s) supplémentaire(s) de validation souvent présente(s)



## Exemple de décodeur binaire

- entrée 3 voies, sortie 8 voies  
(encore appelé « 1 parmi 8 »)
  - **actif au niveau HAUT.**



## Exemple de décodeur binaire

- entrée 3 voies, sortie 8 voies
- actif au niveau BAS (CI 74138)**

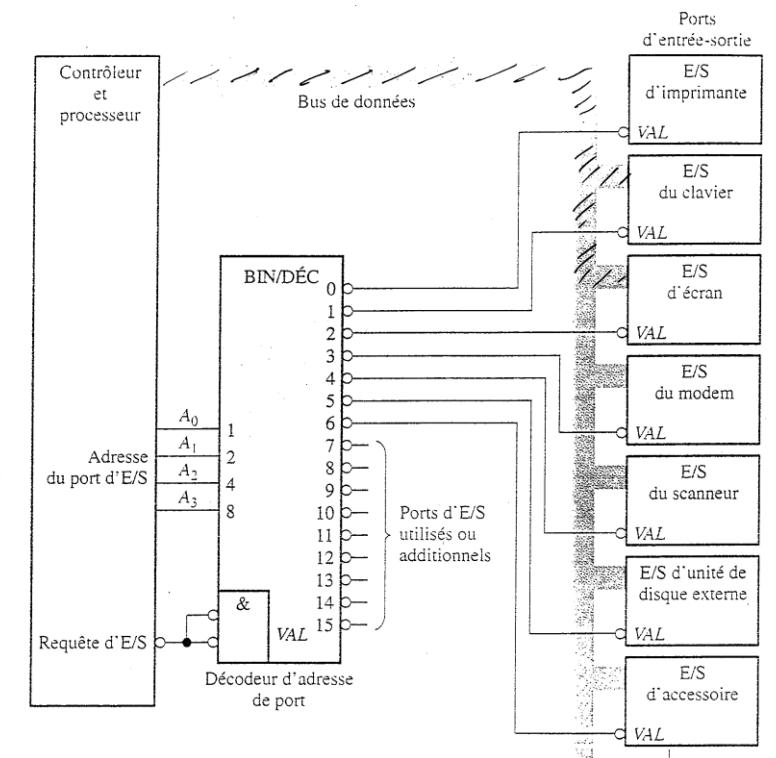
Entrées			Sorties							
		Adresse								
G <sub>1</sub>	G <sub>2</sub> *	C B A	Y <sub>0</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>	Y <sub>4</sub>	Y <sub>5</sub>	Y <sub>6</sub>	Y <sub>7</sub>
X	1	X X X	1	1	1	1	1	1	1	1
0	X	X X X	1	1	1	1	1	1	1	1
1	0	0 0 0	0	1	1	1	1	1	1	1
1	0	0 0 1	1	0	1	1	1	1	1	1
1	0	0 1 0	1	1	0	1	1	1	1	1
1	0	0 1 1	1	1	1	0	1	1	1	1
1	0	1 0 0	1	1	1	1	0	1	1	1
1	0	1 0 1	1	1	1	1	1	0	1	1
1	0	1 1 0	1	1	1	1	1	1	0	1
1	0	1 1 1	1	1	1	1	1	1	1	0

\*  $G_2 = G_{2A} + G_{2B}$

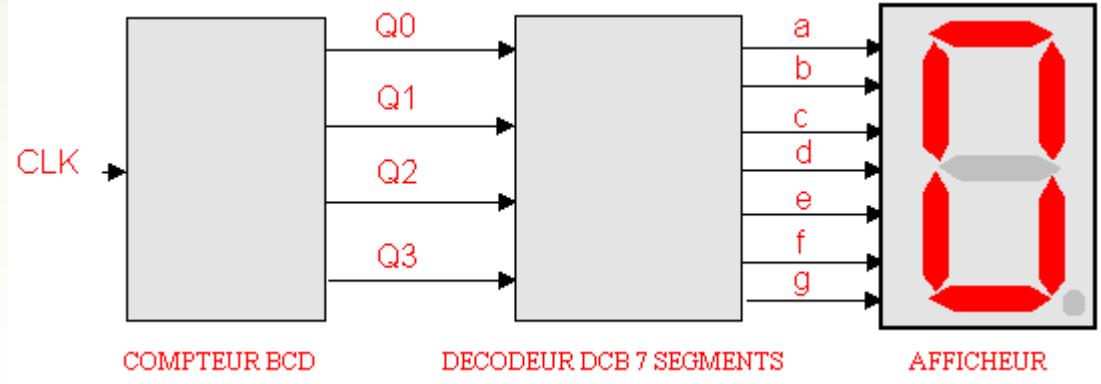
## b) Décodeur binaire comme décodeur d'adresse

- ▶ Les entrées correspondent aux bits d' adresse
- ▶ La sortie permet de sélectionner un dispositif particulier (lorsque son adresse apparaît sur l'entrée du décodeur)
- ▶ Fonctionnement d'un décodeur d'adresse :
  - En entrée : un nombre entier x codé en binaire, correspondant à l'adresse du périphérique à sélectionner (clavier, écran, scanner, disque dur externe..)
  - En sortie : la sortie numérotée x est mise à 1 (activée)
  - Toutes les autres sorties sont mises à 0 (désactivées)
- ▶ Exemple : sélection des entrées / sorties d'un ordinateur
  - Décodeur sélectionne le port d'E/S pour envoyer ou recevoir les données
  - Adresse émise décodée par le décodeur pour valider le port d'E/S à sélectionner.

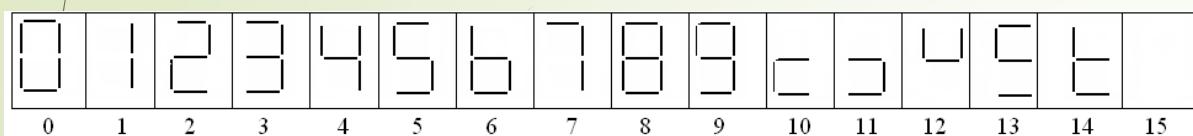
Ports d'E/S d'un ordinateur muni d'un décodeur d'adresse.



### c) Décodeur DCB – 7 segments



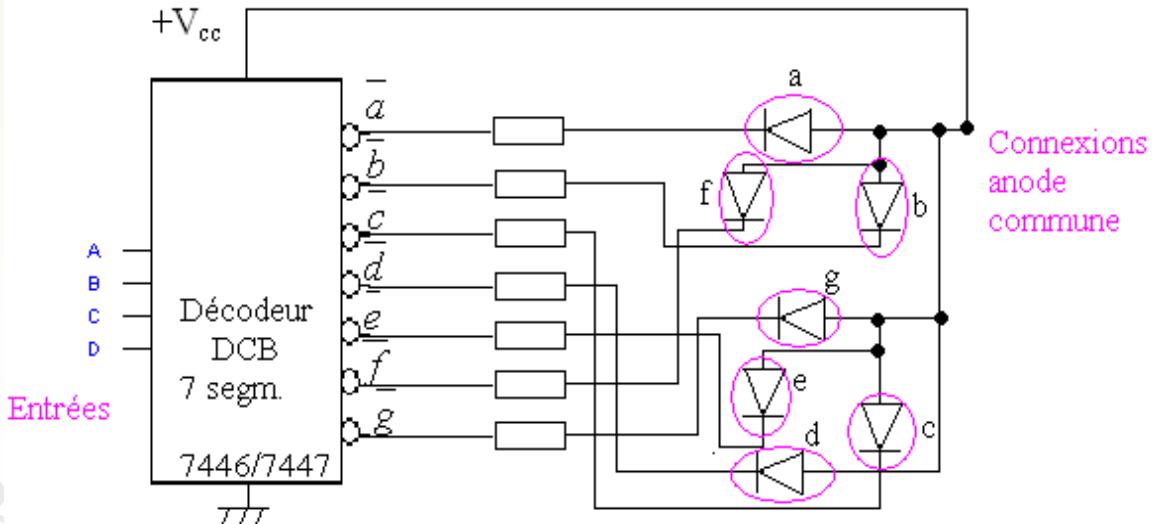
- ▶ Il reçoit un code DCB en entrée et produit des sorties pour piloter des afficheurs à 7 segments, afin d'obtenir un affichage décimal.
- ▶ Afficheur : Chaque segment est constitué d'un matériau qui émet de la lumière quand il est traversé par un courant (ex : diode électro-luminescente DEL=LED)
- ▶ Décodeur :
  - Un décodeur DCB-7 segments accepte en entrée les 4 bits DCB
  - Il active les sorties qui vont permettre des laisser passer un courant dans les segments formant le chiffre décimal correspondant.
  - Décodeur un peu différent des précédents : chaque sortie peut être activée dans plusieurs combinaisons des bits d'entrée (ex : segment e pour les chiffres 0,2,6,8 c'est à dire pour codes 0000, 0010, 0110, 1000)



### Fonctionnement :

- Segment = LED
  - Anodes reliées à  $V_{cc}$  (+5V)
  - Cathodes reliées aux sorties appropriées du décodeur (à travers R pour limiter le courant)
  - Sorties Actives au niveau BAS (ex : décodeur 7446 / 7447)
  - Si l'entrée DCB est : D=0, C=1, B=0, A=1 c'est-à-dire la représentation DCB de 5
  - Sorties activées sont : , les cathodes correspondantes sont alors à un niveau de tension très faible
  - Un courant passe à travers les diodes des segments a, f, g, c, d □ Allumage du nombre 5.
  - Les autres sorties restent au niveau HAUT et les segments éteints.
- Rm : les décodeurs 7446 et 7447 sont conçus de manière à afficher quelque chose même si le code >1001(9). Cet afficheur est dit à « anode commune » parce que toutes les anodes sont reliées à  $+V_{cc}$ . Il existe des afficheurs à « cathode commune » qui sont utilisés avec un décodeur sorties actives Niveau Haut (ex : Ci 7448)

Décodeur DCB-7-segments + Afficheur 7 segments

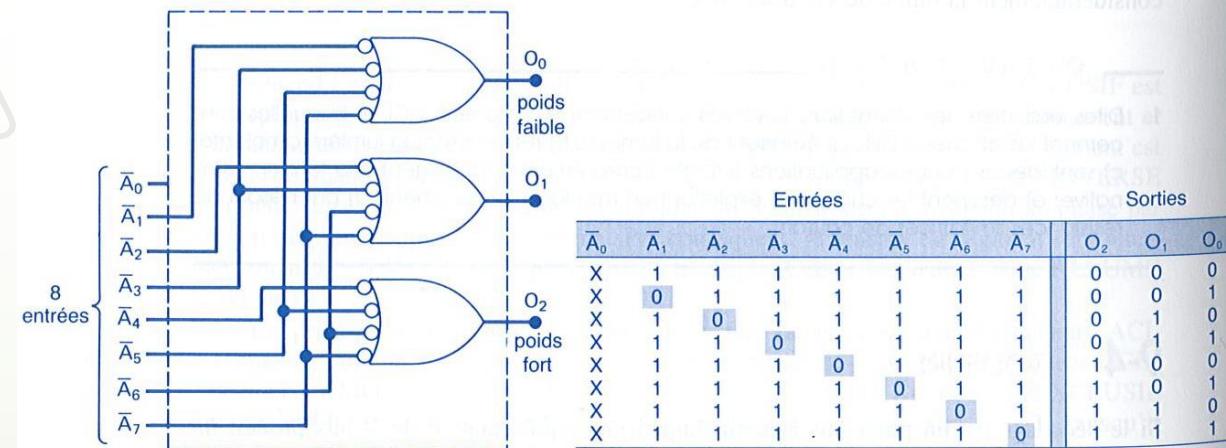
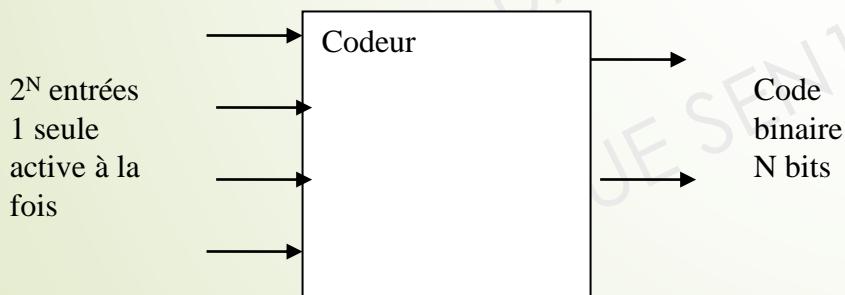


## 3.2. (En)codeurs

### a) Cas général : une seule entrée active

- **Décodage** (binaire) = processus à partir duquel une représentation de N bits produit un signal 1 (ou 0) sur une et seulement une des lignes de sortie du décodeur (celle correspondant au code mis en entrée)
- **Encodage** = processus inverse : un codeur a un certain nombre de voies d'entrée dont une seule est active à la fois.
- A une voie d'entrée correspond un code binaire en sortie.

► Schéma général d'un codeur :



► Exemple : Un codeur octal binaire a 8 voies d'entrées et produit une représentation binaire de 3 bits en sortie.

## b) Cas particulier : plusieurs entrées actives

- ▶ **Encodeur de priorité** (ex CI 74148=codeur de priorité binaire)
- ▶ C'est une version modifiée du codeur élémentaire.
- ▶ Attribue une **priorité** aux entrées (pour permettre de sélectionner une seule entrée dans le cas où plusieurs entrées sont mises à 1 – ex : si deux touches d'un clavier sont appuyées en même temps, ou bien si plusieurs périphériques d'entrée d'un ordinateur essaient d'envoyer des données en même temps.)
- ▶ En sortie on a le **code binaire de l'entrée la + élevée** si plusieurs entrées sont activées en même temps.
- ▶ Exemple : quand les entrées **E5 et E2** sont actives en même temps, la réponse donnée en **sortie est le code binaire de 5** (101).
- ▶ Souvent une sortie supplémentaire indiquant si au moins une entrée activée.
- ▶ [Datasheet 74147-74148](#)

Inputs				Outputs		
D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	Y <sub>1</sub>	Y <sub>0</sub>	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

Equations de sorties :

$$Y_1 = D_3 + \overline{D_3} D_2 = D_3 + D_2$$

$$Y_0 = D_3 + \overline{D_3} \overline{D_2} D_1 = D_3 + \overline{D_2} D_1$$

$$V = D_3 + D_2 + D_1 + D_0$$

Inputs								Outputs		
D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	Y <sub>2</sub>	Y <sub>1</sub>	Y <sub>0</sub>
1	0	0	0	0	0	0	0	0	0	0
x	1	0	0	0	0	0	0	0	0	1
x	x	1	0	0	0	0	0	0	1	0
x	x	x	1	0	0	0	0	0	1	1
x	x	x	x	1	0	0	0	1	0	0
x	x	x	x	x	1	0	0	1	0	1
x	x	x	x	x	x	1	0	1	1	0
x	x	x	x	x	x	x	1	1	1	1

# Circuits combinatoires

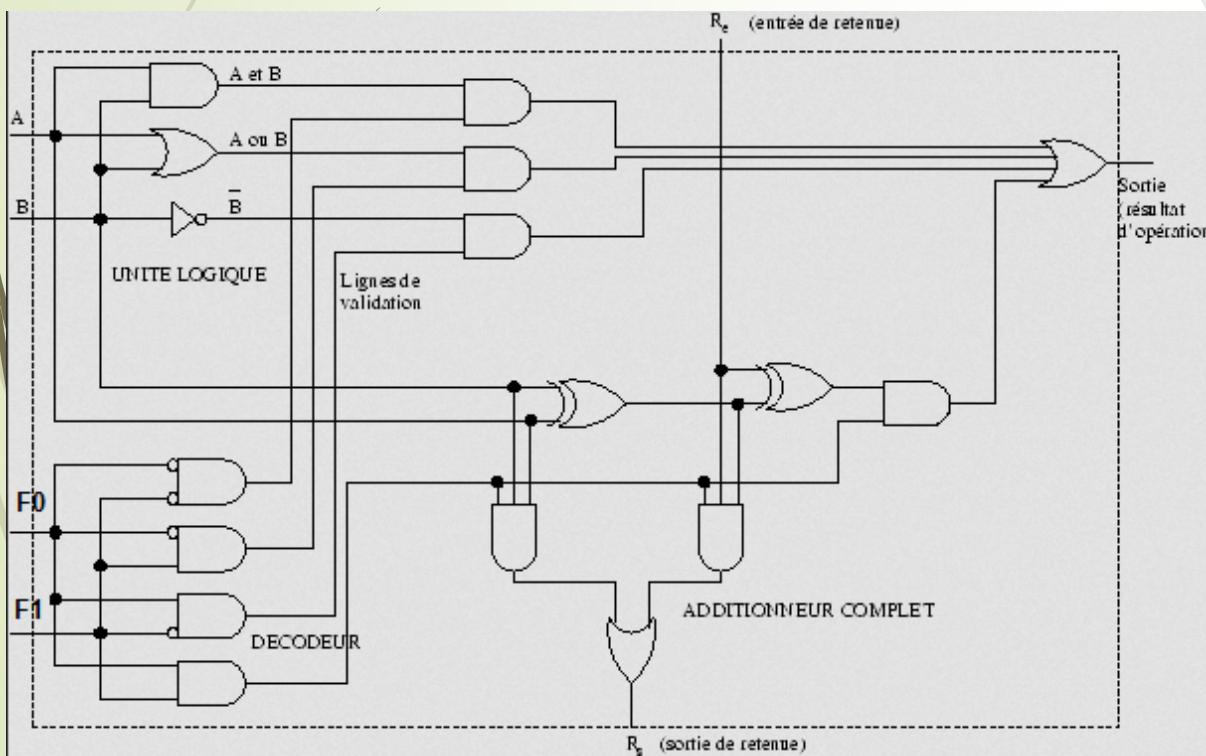
4. Comparateur / Unité Arithmétique et Logique

## 4.1. Comparateur

- ▶ Un comparateur permet de **comparer 2 mots binaires de plusieurs bits**.
- ▶ En TD : comparaison de deux mots de deux bits :  $A=a_1a_0$  et  $B=b_1b_0$
- ▶ Utilisations principales :
  - Circuits de décodage des adresses des ordinateurs, pour sélectionner le périphérique d'entrée / sortie, ou localiser la zone mémoire contenant les données à retrouver (**comparaison du code d'adresse** envoyé par le processeur, aux codes d'adresse des périphériques ou mémoire).
  - Application de régulation : nombre binaire représentant la variable physique régulée (vitesse, position..) est **comparé à une valeur de consigne**.
- ▶ Ex : CI 74 85 : comparateur de 2 nombres de 4 bits A et B, cascadable, avec 3 sorties
  - S vaut 1 si et seulement si  $A > B$
  - I vaut 1 si et seulement si  $A < B$
  - E vaut 1 si et seulement si  $A = B$

## 4.2. Unité Arithmétique et Logique (UAL)

- ▶ C'est la zone de l'ordinateur où sont effectuées les opérations arithmétiques et logiques sur les données.
- ▶ L'opération est choisie par une entrée de commande.
- ▶ Exemple simple d'une UAL qui traite 4 opérations (OU, ET, NON, addition)



- Les lignes F0, F1 permettent de sélectionner (par l'intermédiaire d'un décodeur) l'opération à réaliser
- Angle sup. gauche : on trouve les fonctions élémentaires
- Le résultat d'une seule opération est transmis à un instant donné vers la porte OU de sortie, en fonction du décodeur (les autres transmettent un 0 vers la porte OU)
- L'additionneur transmet à l'extérieur sa propre retenue de sortie
- L'association de telles unités (ou de circuits plus complexes) permet de réaliser des opérations de base sur des nombres de plusieurs bits.