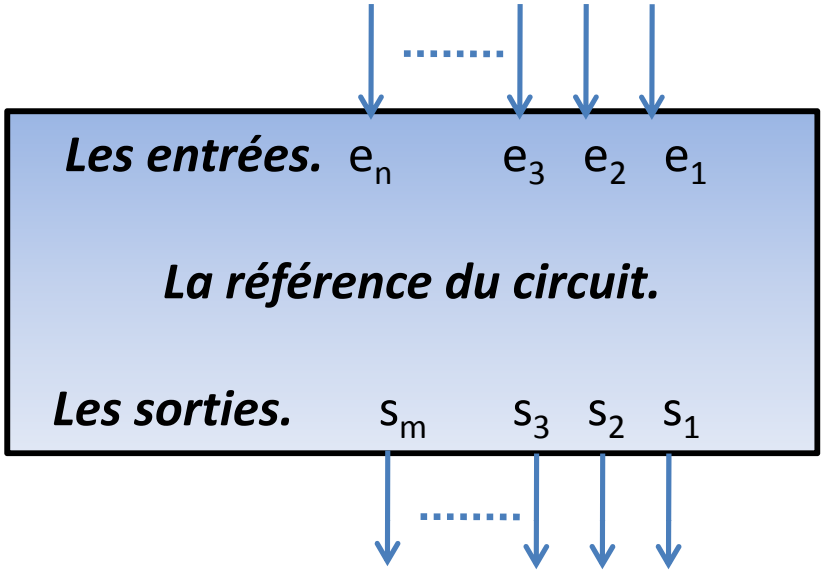
Logique combinatoire et séquentielle.

Un circuit numérique (ou logique ou digital) est un réseau de composants qui traitent des données à valeurs discrètes et produit des résultats , également à valeurs discrètes. Il peut être présenté comme une boite noire avec les indications suivantes:

- ✓ Ses entrées.
- ✓ Ses sorties.
- ✓ La spécification (non ambiguë de sa fonction).
- ✓ Et pour une utilisation concrète, ses spécifications électriques, temporelles, mécaniques et de température.

Il est souvent représenté par un schéma dit ***schéma bloc*** ou ***bloc diagram***, comme suit:



Logique combinatoire et séquentielle.

Les circuits logiques sont de deux types: les ***combinatoires*** et les ***séquentiels.***
Les circuits séquentiels seront abordés en deuxième position dans cette partie.

A) Les circuits logiques combinatoires:

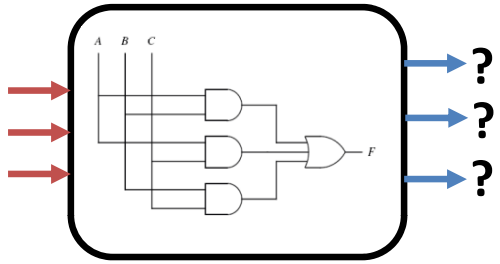
Ce sont ceux ou les sorties à l'instant ***t*** ne dépendent que des entrés à l'instant ***t***. Autrement dit c'est des circuits sans état, ils n'ont aucune idée de leur passé, ils sont dits sans mémoire. Les sorties ne sont jamais rebouclées vers les entrées.

Par ailleurs ils sont souvent ***asynchrones***: dès que leurs entrées changent, leurs sorties changent en conséquence (***après un certain délai***); ils n'attendent pas un quelconque signal d'autorisation (ce signal peut exister dans certains cas, mais ceci ne change rien à leur caractère combinatoire)

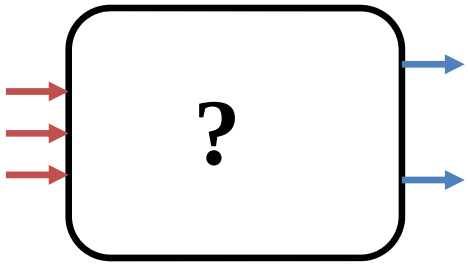
Les questions qui se posent au sujet des circuits logiques ,qu'ils soient combinatoires ou séquentiels sont de deux ordres:

- ***Analyse.***
- ***Conception (ou synthèse).***

- Analyse
  - Etant donné un circuit, trouver sa fonction.
  - La fonction peut être exprimée par:
    - Une équation Booléenne.
    - Une table de vérité.



- Conception (ou synthèse).
  - Partant d'une fonction recherchée, déterminer le circuit.
  - La fonction peut être exprimée par:
    - Une équation Booléenne.
    - Une table de vérité.
    - Un schéma de réalisation-simulation.



Nous allons surtout nous intéresser à l'activité de synthèse et de conception de quelques circuits combinatoires standards tels que :

- Les additionneurs/soustracteurs,
- Les comparateurs,
- Les multiplexeurs, démultiplexeurs
- Les décodeurs et les encodeurs.

Un circuit combinatoire est constitué d'une interconnexion d'autre circuits combinatoires moins élaborés et de portes logiques.
Ses entrées, **binaires**, proviennent d'autres circuits; et ses sorties, **binaires**, constituent des entrées d'autres circuits.
Ses n entrées binaires peuvent prendre 2^n combinaisons possibles.
Chacune des 2^n combinaisons engendre un seul état pour chacune de ses sorties.
Un circuit combinatoire peut être spécifié par une table de vérité qui donne les valeurs de sortie pour chacune des combinaisons en entrée.
De même, il peut être décrit par autant d'équations Booléennes qu'il y a de sortie.

• *Activité de conception.*

Etant donné une fonctionnalité recherchée, la démarche de conception est:

- *Esquisser un schéma bloc.*
- *Déterminer les entrées et les sorties.*
- *Dériver la table de vérité.*
- *Simplifier l'équation logique pour chaque sortie.*
- *Produire le circuit (par son schéma interne ou par simulateur).*
- *(Il peut aussi être défini au moyen de langages spécifiques).*

Cette procédure n'est possible que pour des circuits ou la table de vérité reste de l'échelle du maitrisable.

Dans tous les autres cas, il faut ***décomposer*** le problème et faire jouer le principe :

"Diviser pour régner".

Les circuits combinatoires que nous allons étudier existent déjà en tant que circuits MSI standards (compte tenu de leurs fonctionnalités usuelles) ou en tant que éléments dans des bibliothèques de composants dans des environnement de conception et/ou de simulation.

• ***Synthèse du circuit demi additionneur.***

Schéma bloc :
Les entrés **a** et **b** sur 1 bit.
c, sur 1 bit est la somme.
d, sur 1 bit est la retenue.

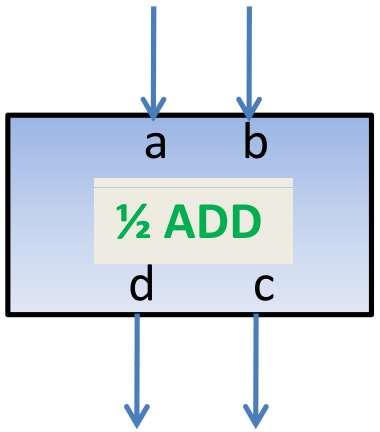
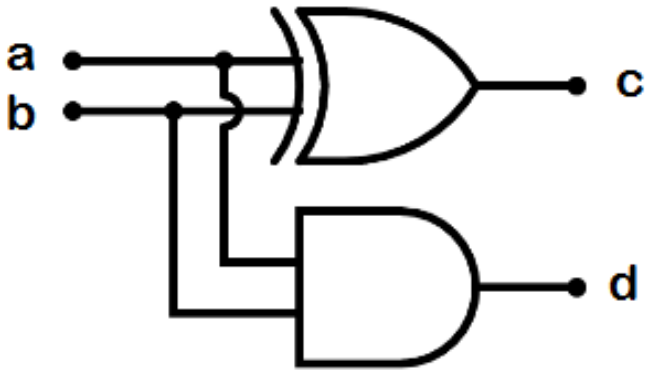


Table de vérité

a	b	c	d
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

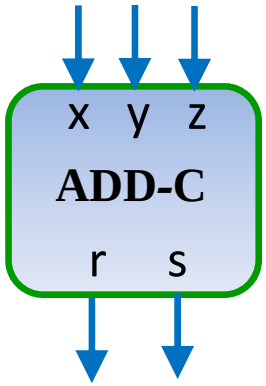
On tire les équations logiques de s (la somme) et r (la retenue)
c = a ⊕ b
d = a.b

De là on déduit le schéma de réalisation
ou le ***contenu*** de la boite noire.



- ***Synthèse du circuit additionneur complet.***

Schéma bloc:



Les entrées:

x , y, z (la retenue entrante) sur 1 bit.

Les sorties:

s est la somme, sur 1 bit.

r, la retenue sortante, sur 1 bit.

La fonction: est claire

$$\begin{array}{r} z \\ + \quad y \\ + \quad x \\ \hline r \quad s \end{array}$$

Table de vérité

x	y	z	s	r
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

- ***Synthèse du circuit additionneur complet.***

	xy 00	01	11	10
$\bar{z}$ 0		1		1
z 1	1		1	

Tab. K. de s

	xy 00	01	11	10
$\bar{z}$ 0			1	
z 1		1	1	1

Tab. K. de r

x	y	z	s	r
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

$$s = \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + x.y.z + x.\bar{y}.\bar{z} = \bar{z}.(x.\bar{y} + \bar{x}y) + z.(\bar{x}.\bar{y} + x.y)$$

$$s = \bar{z}.(x \oplus y) + z.(\overline{x \oplus y}) = (x \oplus y) \oplus z$$

$$r = x.y + y.z + x.z$$

on peut également l'écrire :

$$r = x.y + x.\bar{y}.z + \bar{x}.y.z = x.y + z.(x \oplus y)$$

- ***Synthèse du circuit additionneur complet.***

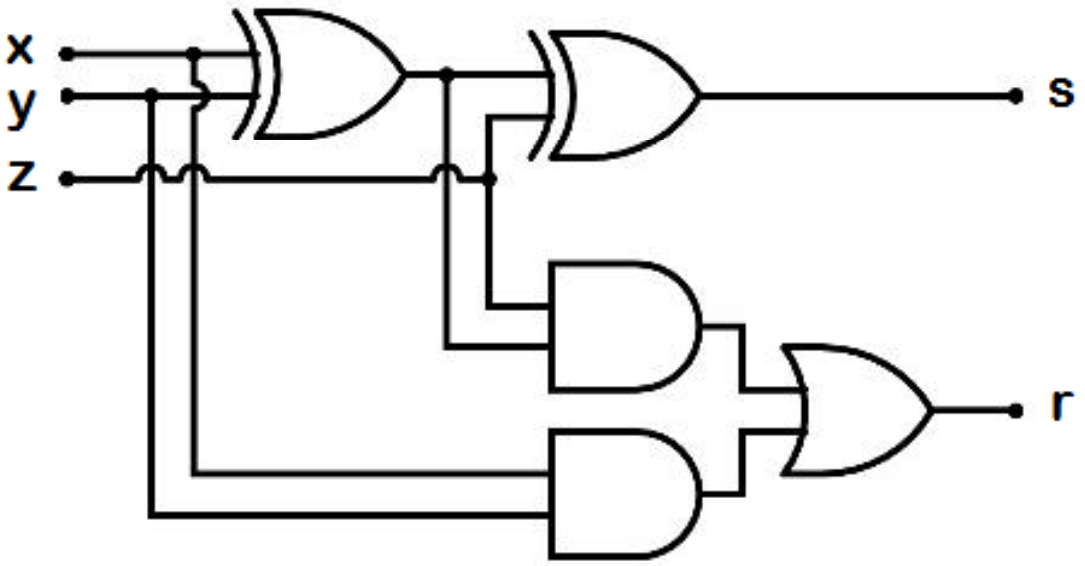
$$s = \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + x.y.z + x.\bar{y}.\bar{z} = \bar{z}.(x.\bar{y} + \bar{x}y) + z.(\bar{x}.\bar{y} + x.y)$$

$$s = \bar{z}.(x \oplus y) + z.(\overline{x \oplus y}) = (x \oplus y) \oplus z$$

$$r = x.y + y.z + x.z$$

on peut également l'écrire :

$$r = x.y + x.\bar{y}.z + \bar{x}.y.z = x.y + z.(x \oplus y)$$



x	y	z	s	r
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

- ***Synthèse d'un circuit à base d'autres circuits:***

L'additionneur à base de demi-additionneur(s).

les équations de l'additionneur complet,

sont reprises ci - après :

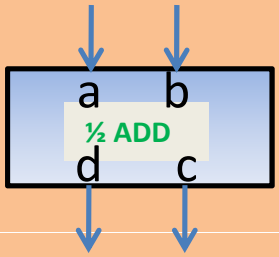
$$s = (x \oplus y) \oplus z$$

$$r = x.y + z.(x \oplus y)$$

Le demi additionneur:

La somme c = a ⊕ b

La retenue d = a.b



On reprend s =(x ⊕ y) ⊕ z et r = x.y +z.(x ⊕ y)

Soit c1 = (x ⊕ y) et c2 = c1 ⊕ z

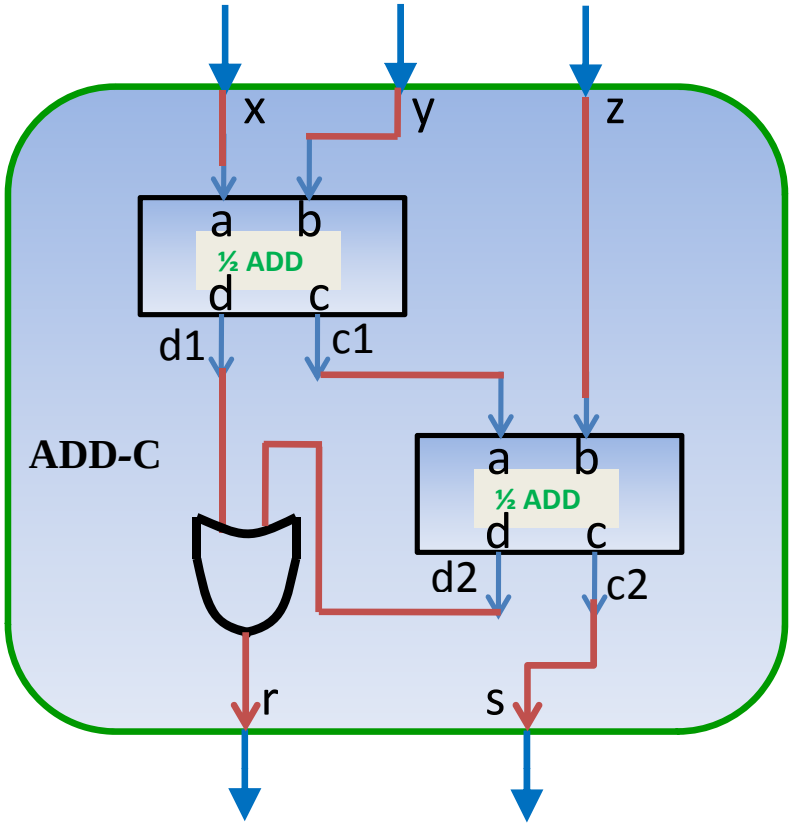
s = c2

donc s peut être élaboré grâce aux sorties c de deux (½ ADD)s.

Soit d1=xy et d2 = z.c1;

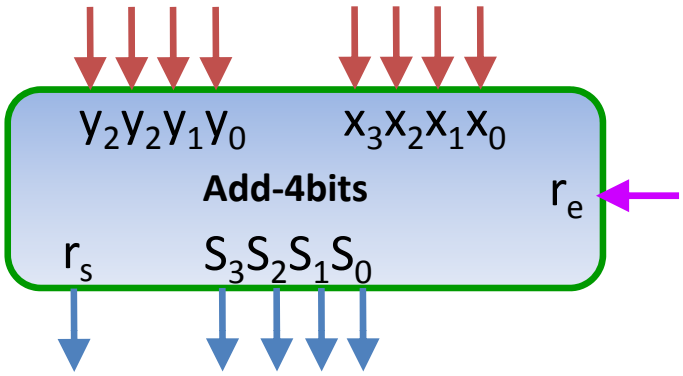
on peut réécrire r =d1 + d2.

donc r est un OU Logique entre les sorties d de deux (½ ADD)s.



- ***Synthèse du circuit additionneur binaire pur 4 bits.***

Schéma bloc:



Les entrées:

X_{3-0}, Y_{3-0} : X & Y sur 4 bits.
La retenue entrante r_e sur 1 bit.

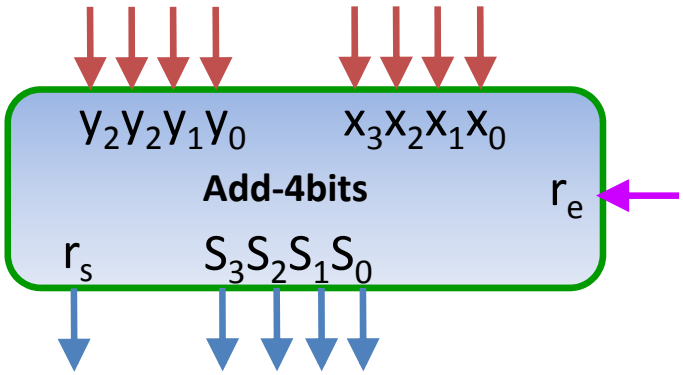
Les sorties:

S_{3-0} , la somme sur 4 bits.
r_s, la retenue sortante sur 1 bit.

La fonction: S = X+ Y, avec prise en compte d'une retenue de mise en cascade.

- ***Synthèse du circuit additionneur binaire pur 4 bits.***

Si on tente d'appliquer la procédure suggérée précédemment, l'étape suivante serait la table de vérité.
La TV comporte ici 4+4+1=9 entrées et 5+1 sorties.
Elle a dépassé l'échelle de synthèse manuelle.
Décomposer, diviser pour régner est la voie du salut.

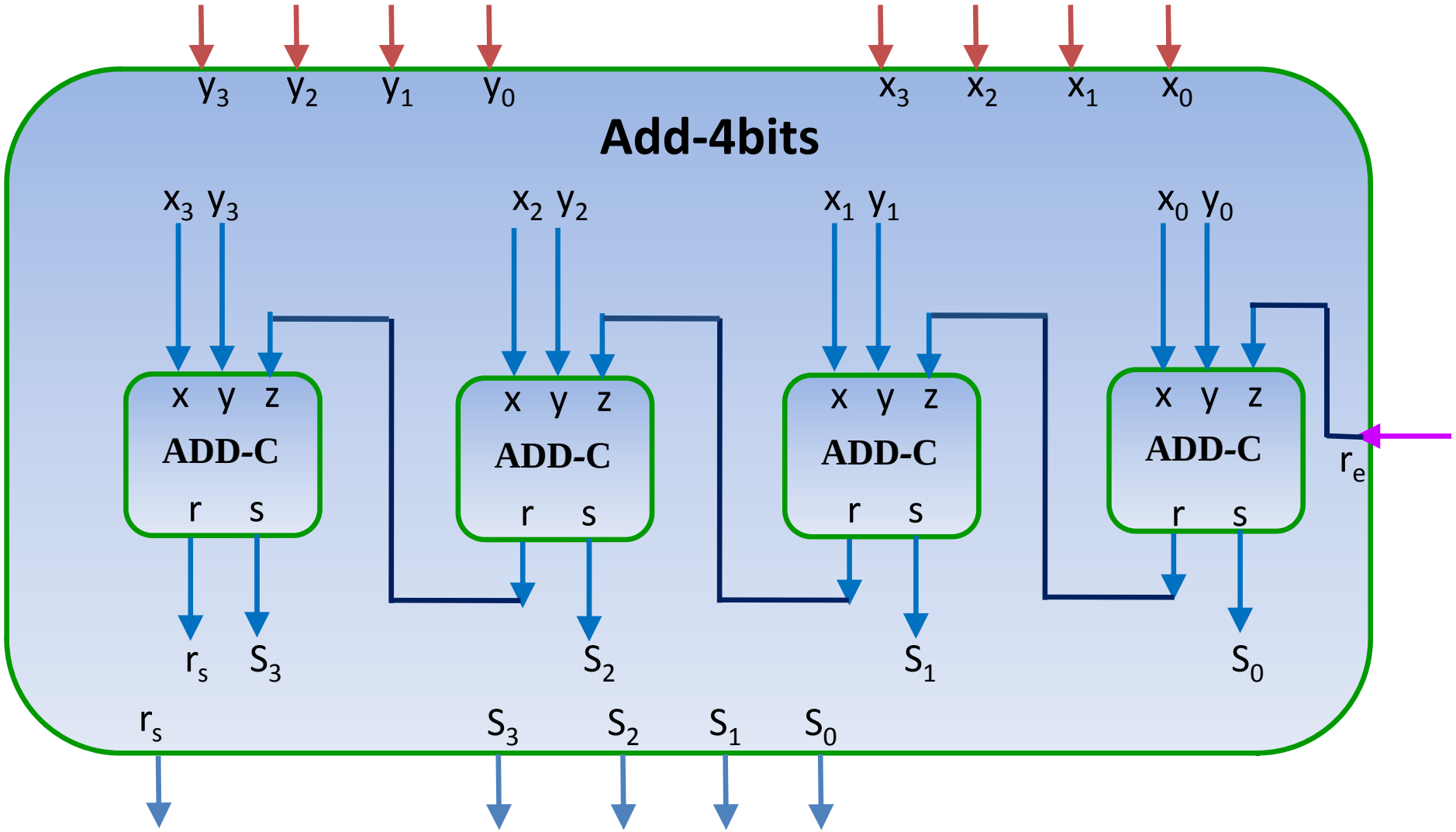


L'addition peut s'effectuer par étage, ou rang de droite à gauche avec propagation de la retenue. Comme l' illustre la figure suivante:

On voit aisément ici que l'addition
$S_i = X_i + Y_i + r_i$ se répète 4 fois.
Or l'addition s= x+y+z et la génération d'une retenue, c'est exactement ce que sait faire un certain circuit appelé **ADD-C** ou ***additionneur complet***. d'où le schéma de la diapositive suivante.

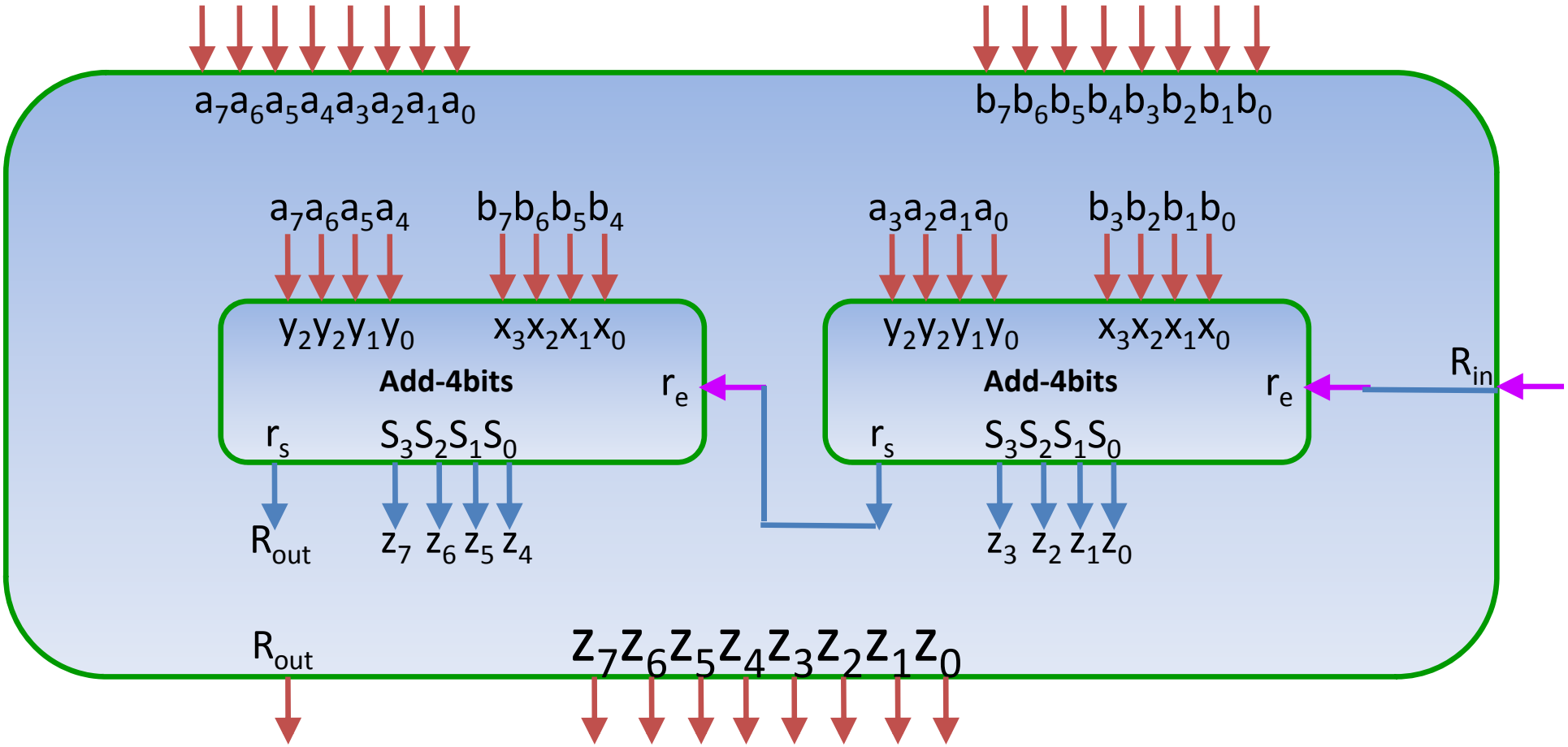
$$\begin{array}{rrrrr} & r_2 & r_1 & r_0 & r_e \\ + & x_3 & x_2 & x_1 & x_0 \\ + & y_3 & y_2 & y_1 & y_0 \\ \hline r_s & S_3 & S_2 & S_1 & S_0 \end{array}$$

- ***Synthèse du circuit additionneur binaire pur 4 bits.***



- ***Exemple d'utilisation de circuit additionneur binaire pur 4 bits ADD-C.*** Soit à réaliser un circuit additionneur de deux nombre entiers naturels A et B codés chacun sur 8 bits, et produisant Z. **Z = A+B**

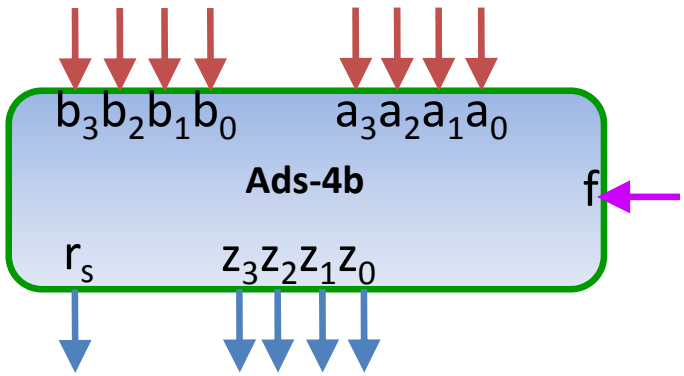
Schéma bloc, entrées et sorties sont évidents.



- ***Additionneur soustracteur binaire 4 bits Ads-4b.***

Soit à réaliser un circuit additionneur soustracteur de deux nombre A et B codés chacun sur 4 bits.
Dès lors qu'on parle de soustraction, donc de nombres négatifs, il faut au préalable dire comment ces nombres vont être codés: réponse, *Le complément à deux*.

Schéma bloc:



Les entrées: a_{3-0}, b_{3-0} *les données, f la fonction pour spécifier l'opération.*
Les sorties: z_{3-0} *la somme ou la différence, r une retenue sortante.*
La fonction :

$$Z = A + B \text{ si } f = 0; \quad Z = A - B \text{ si } f = 1.$$

• ***Additionneur soustracteur binaire 4 bits Ads-4b.***

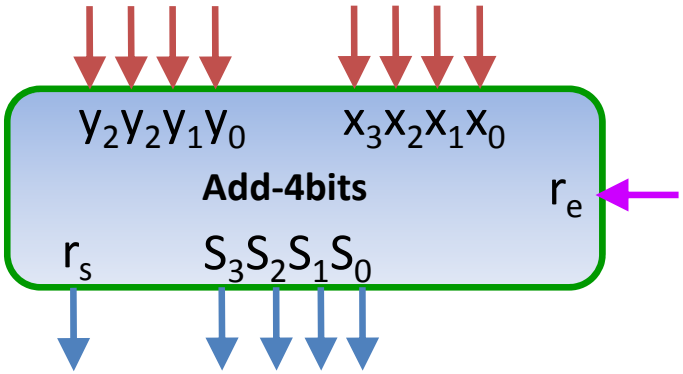
✓*Pour commencer, on écarte l'approche par Table de vérité.*

✓*Par ailleurs on ramène la soustraction à une addition: A – B = A + Complément à 2 de (B).*

Le complément à deux s'obtient en faisant le complément à 1 puis, on ajoute 1.

L'ajout de 1 peut nécessiter un additionneur. Une astuce consiste à l'obtenir par l'exploitation du bit de la retenue entrante en la mettant à 1 lors de la soustraction.

On utilise un additionneur que nous avons conçu auparavant (**Add-4bits**).



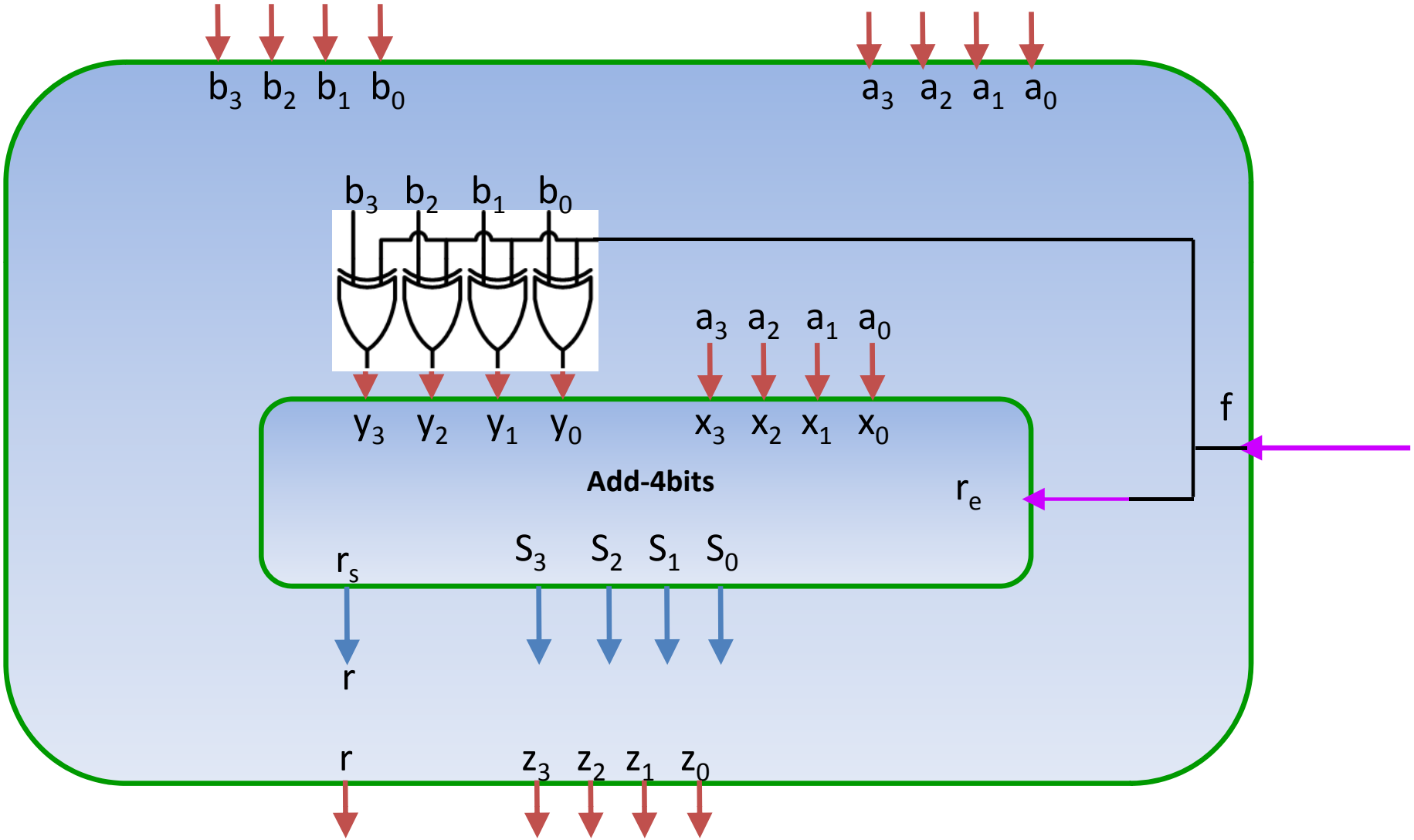
Aux entrées x_i on acheminera les A_i.
Y_i est soit B_i soit le complément *de* B_i selon *f*
Une petite table de vérité peut nous aider à trouver Y_i

$$Y_i = \bar{f}.B_i + f.\bar{B}_i = f \oplus B_i$$

f	B_i	Y_i
0	0	0
0	1	1
1	0	1
1	1	0

D'où le schéma de la diapo suivante.

Additionneur soustracteur binaire 4 bits Ads-4b.



- ***Dépassement de capacité (ou overflow).***

(Error 215: Arithmetic Overflow) de l'environnement de programmation T. Pascal.

Dépassement lors de l'arithmétique avec les nombre signés.

Soit à faire S = X + Y. dans l'ensemble des nombres signés codés dans la notation complément à deux. Il se produit dans deux cas de figures: X et Y sont de même signe et le résultat est de signe opposé. On peut générer l' indicateur de débordement à partir d'une table de vérité classique.

x_3	y_3	s_3	V
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

V étant le drapeau de débordement (*overflow flag*). son équation est simple à trouver:

$$V = \overline{x_3}.\,\overline{y_3}\,.\,s_3 \;+\; x_3\,.\,x_3\,.\,\overline{s_3}$$

Cette équation ne peut se simplifier et nécessite pour sa réalisation

- 3 inverseurs.
- 2 portes ET à deux entrées.
- 1 porte OU à deux entrées.

D'où l' idée de la diapo suivante.

- ***Dépassement de capacité (ou overflow).***

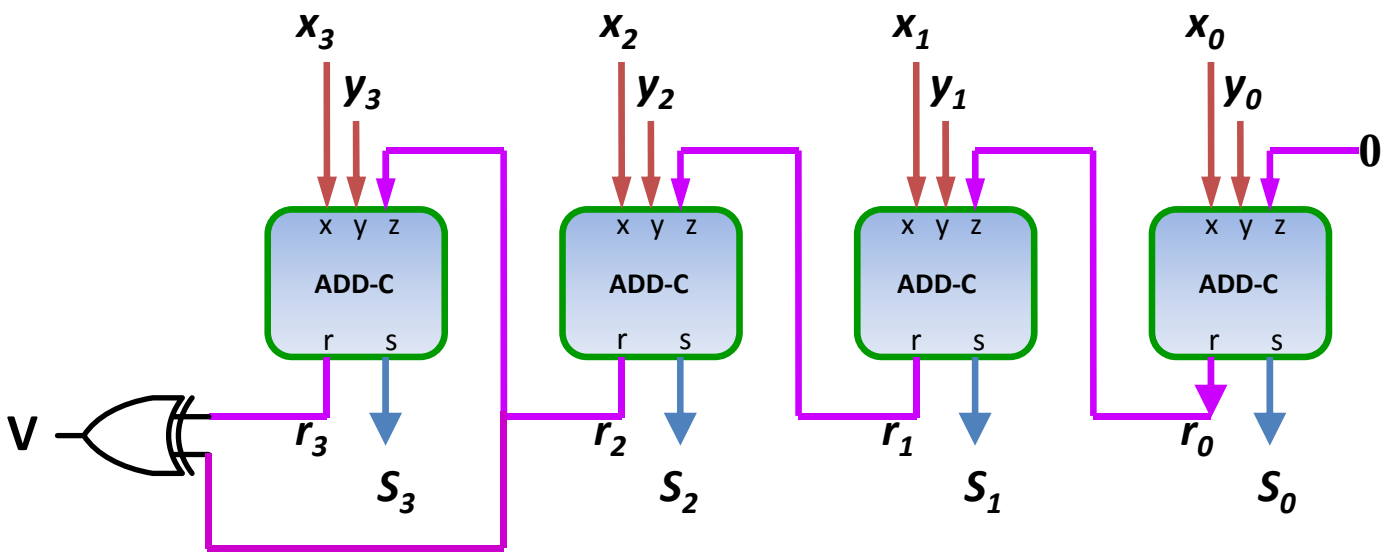
(Error 215: Arithmetic Overflow) de l'environnement de programmation T. Pascal.

Dépassement lors de l'arithmétique avec les nombre signés.

L'idée consiste à scruter ce qui se passe au dernier étage de l'additionneur. Ceci est illustré par la table de vérité ci-dessous.

x_3	y_3	r_2	s_3	r_3	V
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	1	0

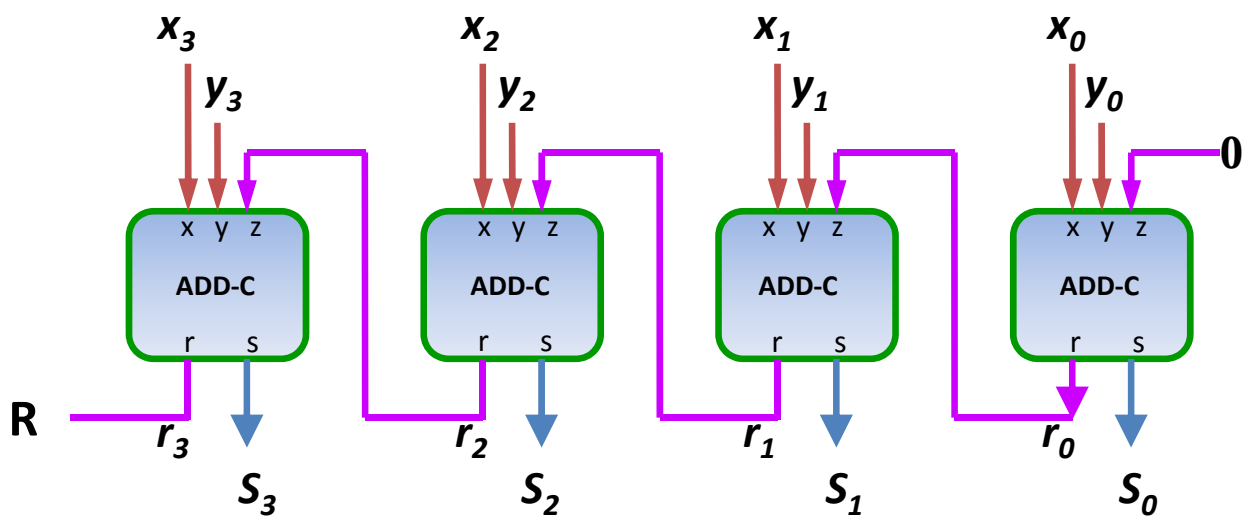
On tire $V = r_2 \oplus r_3$



- ***Dépassement de capacité (ou overflow).***

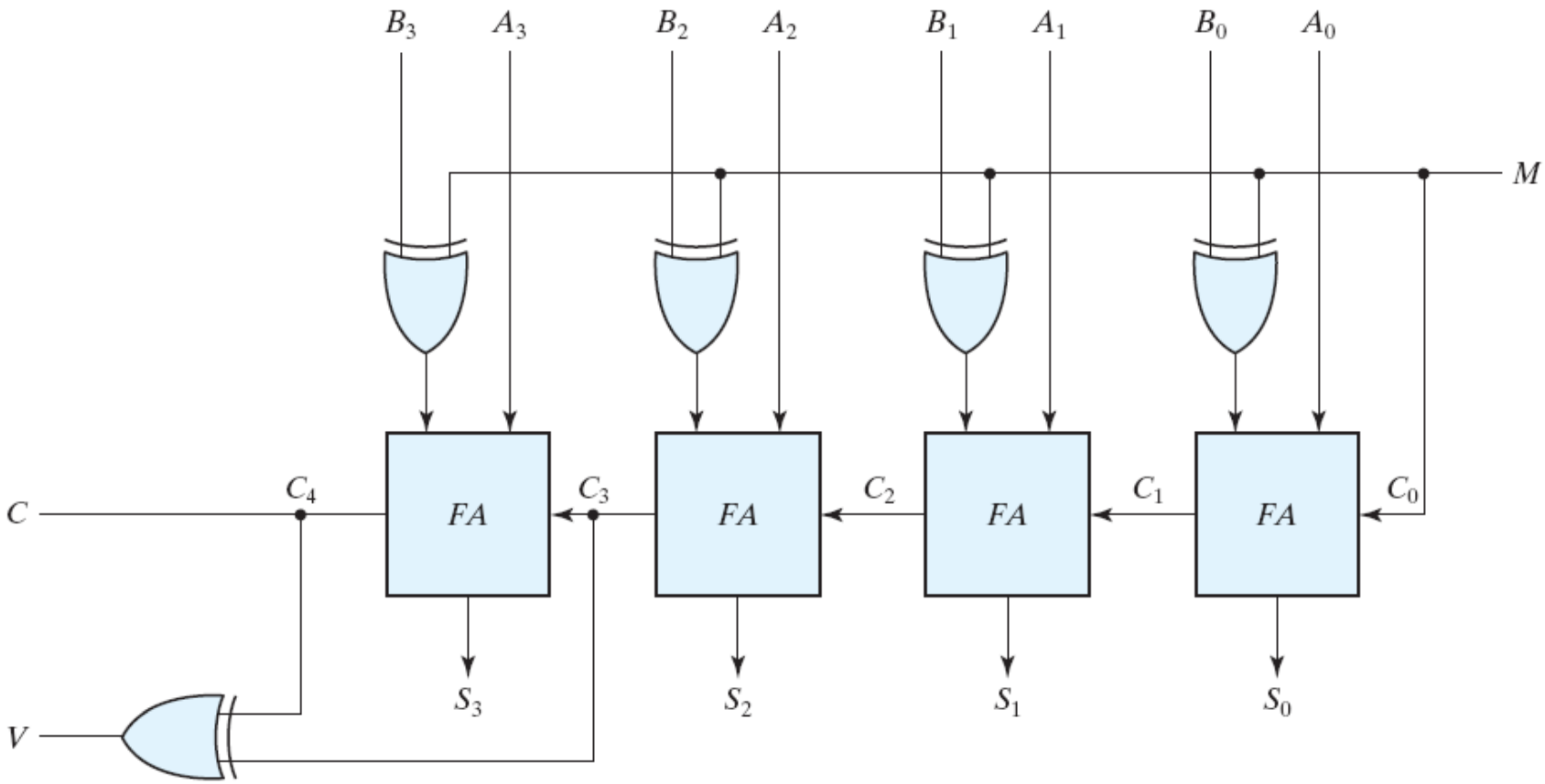
Dépassement lors de l'arithmétique avec les nombre NON signés.

La retenue finale constitue le drapeau du dépassement de capacité.



- ***Dépassement de capacité (ou overflow).***

Circuit générant soit A+B soit A-B selon la valeur de M, avec signalement de dépassement.



- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

Comparateur de deux nombres A et B codés chacun sur un bit.

Schéma bloc

Entrées A et B.

Trois sorties exclusives

- fe : égalité (A=B)
- fi : inférieur (A < B)
- fs : supérieur (A > B)

Ceci est une façon de représenter le résultat de la comparaison.

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

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

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

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

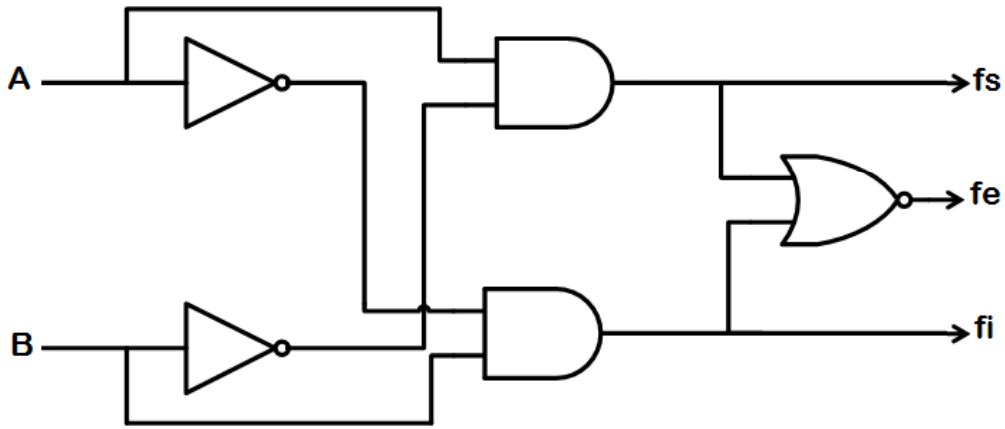
- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

Schéma interne de la boite noire.

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

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

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



- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

Comparateur de deux nombres A et B codés chacun sur deux bits.
Schéma bloc et table de vérité, les équations, sans les schémas.
Exercice : Réalisez ce comparateur à base du précédent.

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

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)

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

- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

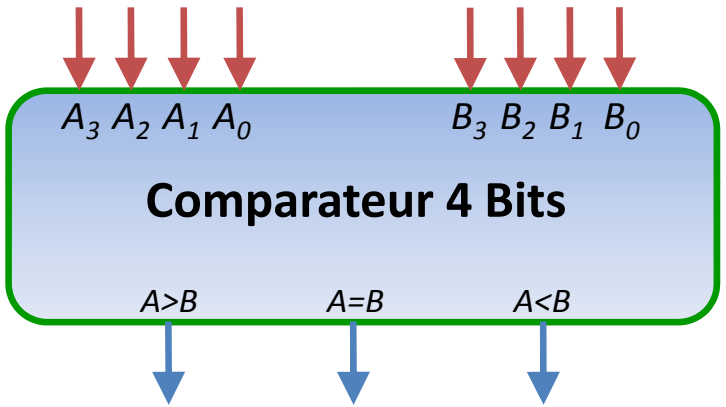
Comparateur de deux nombres A et B codés chacun sur quatre bits.
Schéma bloc et table de vérité, les équations.

$$x_3 = \overline{A}_3\,\overline{B}_3 + A_3\,B_3$$

$$x_2 = \overline{A}_2\,\overline{B}_2 + A_2\,B_2$$

$$x_1 = \overline{A}_1\,\overline{B}_1 + A_1\,B_1$$

$$x_0 = \overline{A}_0\,\overline{B}_0 + A_0\,B_0$$



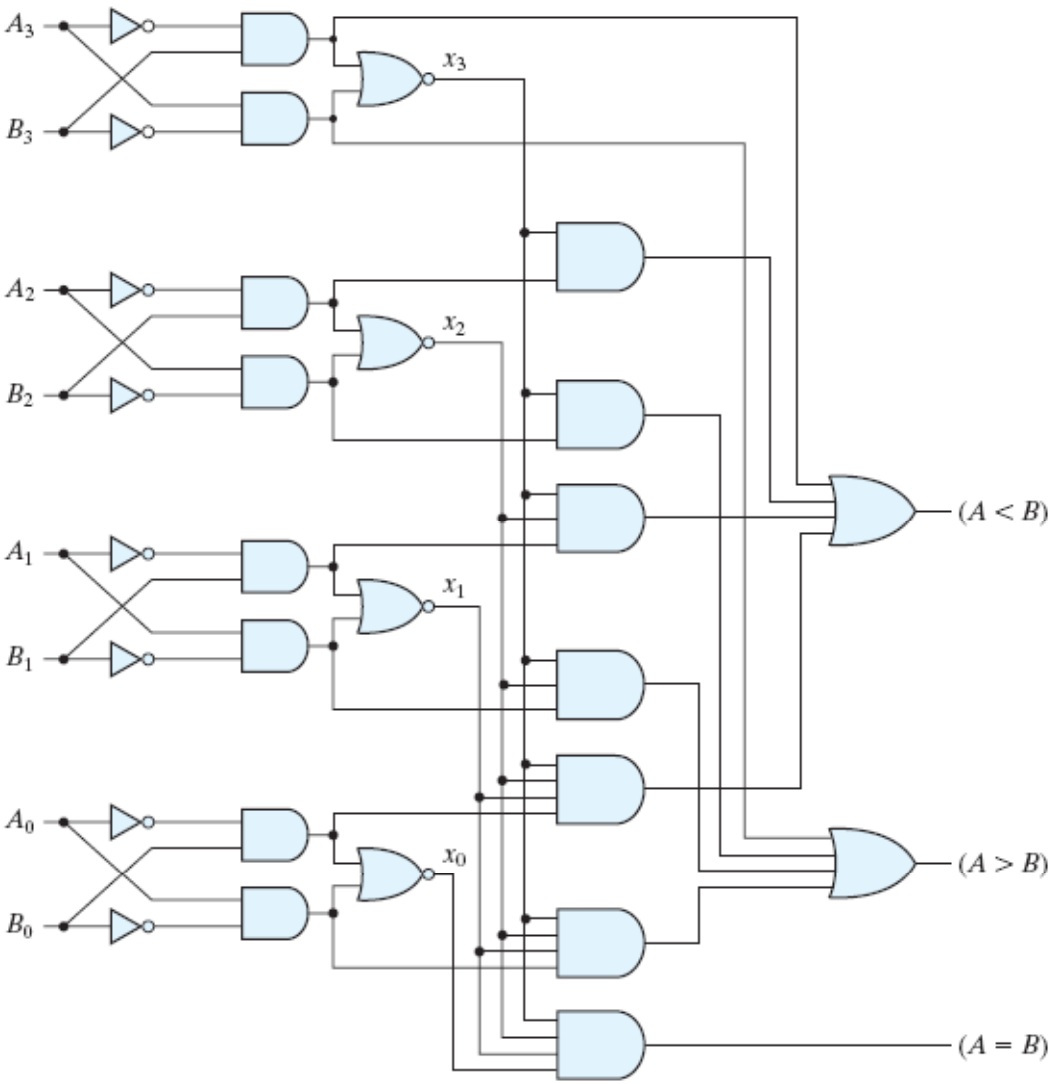
$$(A = B) = x_3\,x_2\,x_1\,x_0$$

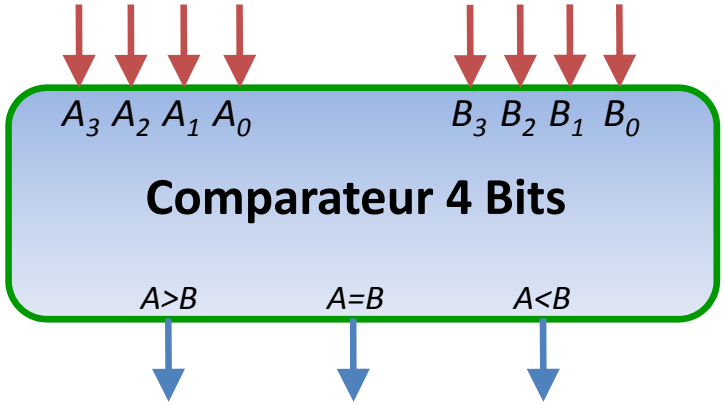
$$(A > B) = A_3\,\overline{B}_3 + x_3\,A_2\,\overline{B}_2 + x_3\,x_2\,A_1\,\overline{B}_1 + x_3\,x_2\,x_1\,A_0\,\overline{B}_0$$

$$(A < B) = \overline{A}_3\,B_3 + x_3\,\overline{A}_2\,B_2 + x_3\,x_2\,\overline{A}_1\,B_1 + x_3\,x_2\,x_1\,\overline{A}_0\,B_0$$

- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

Schéma interne du circuit appelé "comparateur 4 Bits".





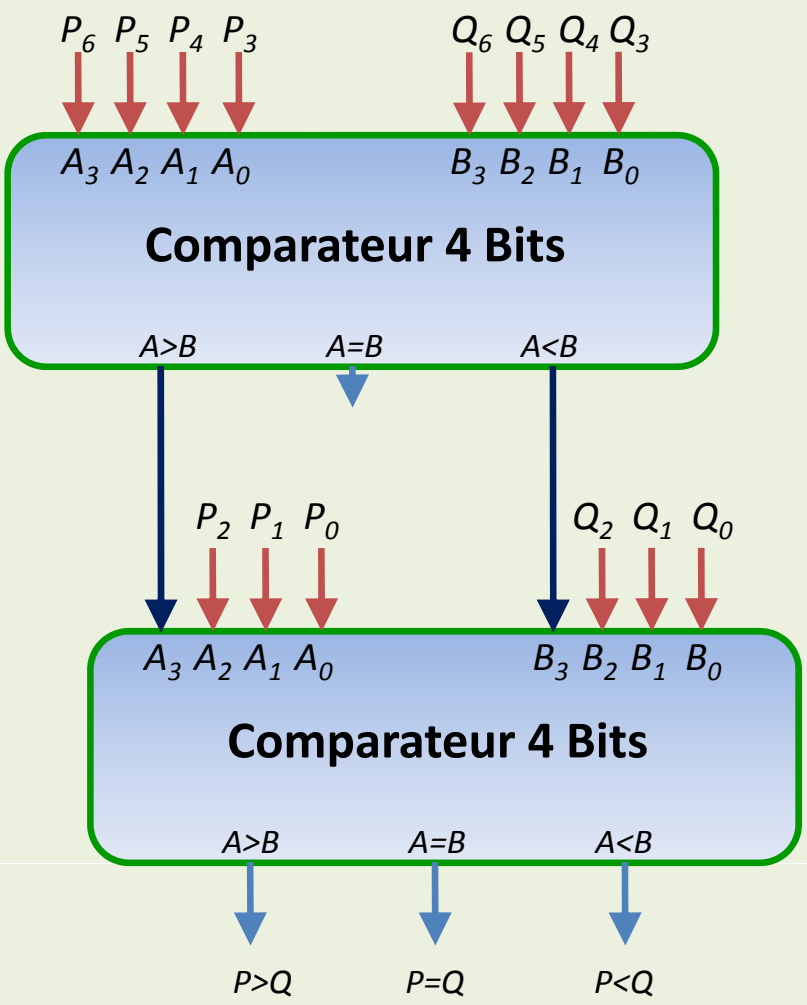
$$(A = B) = x_3\, x_2\, x_1\, x_0$$

$$(A > B) = A_3\, \overline{B}_3 + x_3\, A_2\, \overline{B}_2 + x_3\, x_2\, A_1\, \overline{B}_1 + x_3\, x_2\, x_1\, A_0\, \overline{B}_0$$

$$(A < B) = \overline{A}_3\, B_3 + x_3\, \overline{A}_2\, B_2 + x_3\, x_2\, \overline{A}_1\, B_1 + x_3\, x_2\, x_1\, \overline{A}_0\, B_0$$

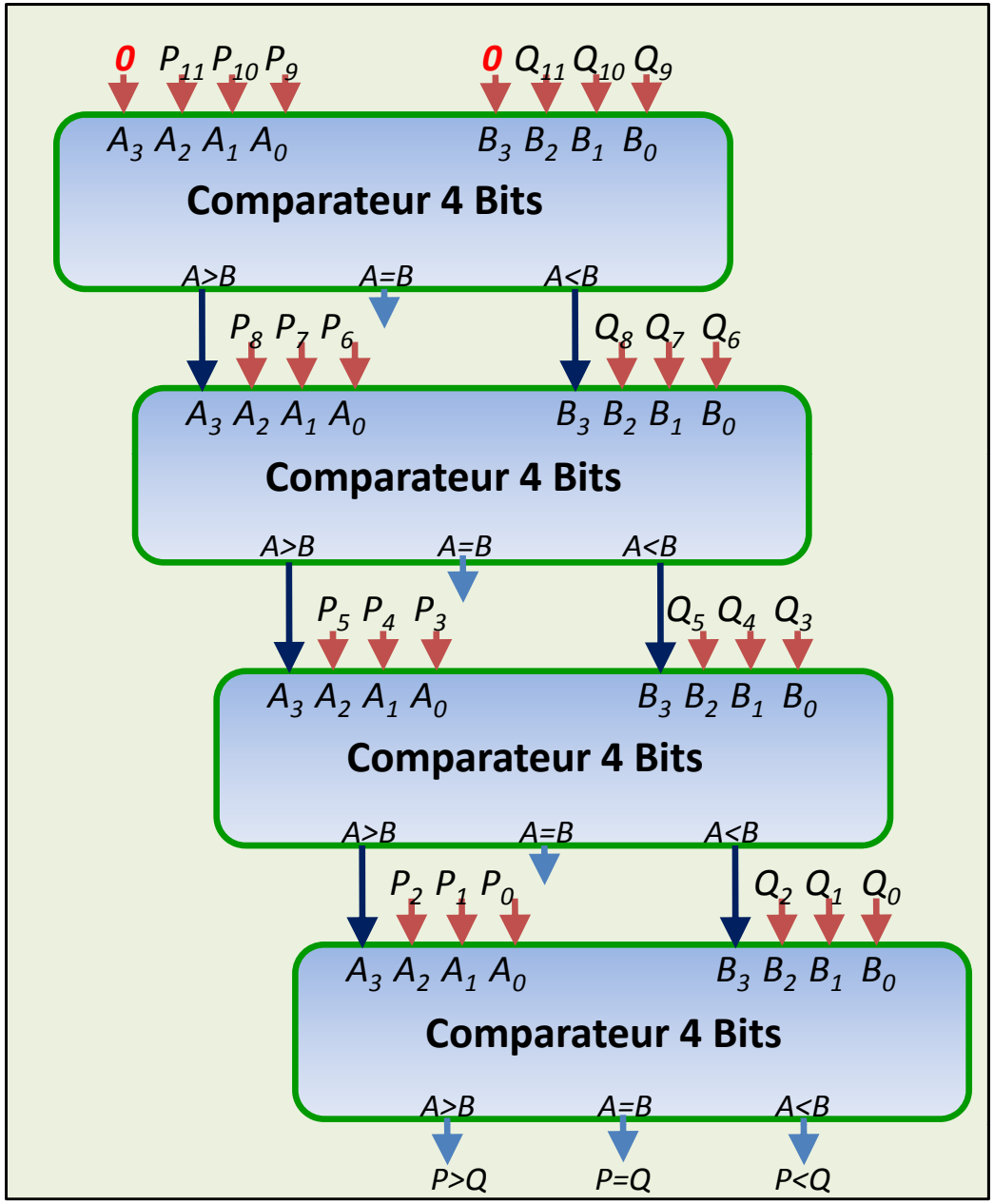
- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

Au-delà de 4 bits, on utilise la mise en cascade de comparateurs de taille modeste. L'idée consiste à soumettre les bits de poids fort au premier comparateur d'une série de comparateurs.
De deux choses l'une.
Soit que la comparaison peut se décider à ce premier étage, auquel cas le premier comparateur impose son résultat aux suivants.
Soit que les bits sont égaux, il laisse faire les autres.
Voici, ci contre un comparateur de deux nombres entiers naturels, P et Q, codés chacun sur 7 bits.



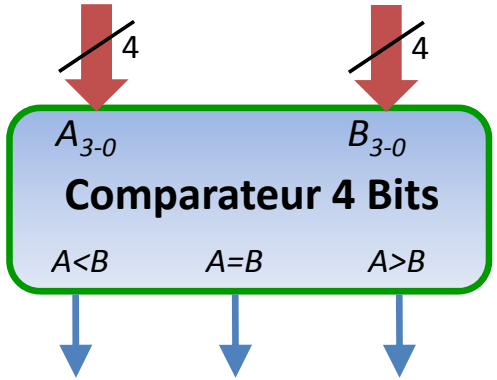
- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

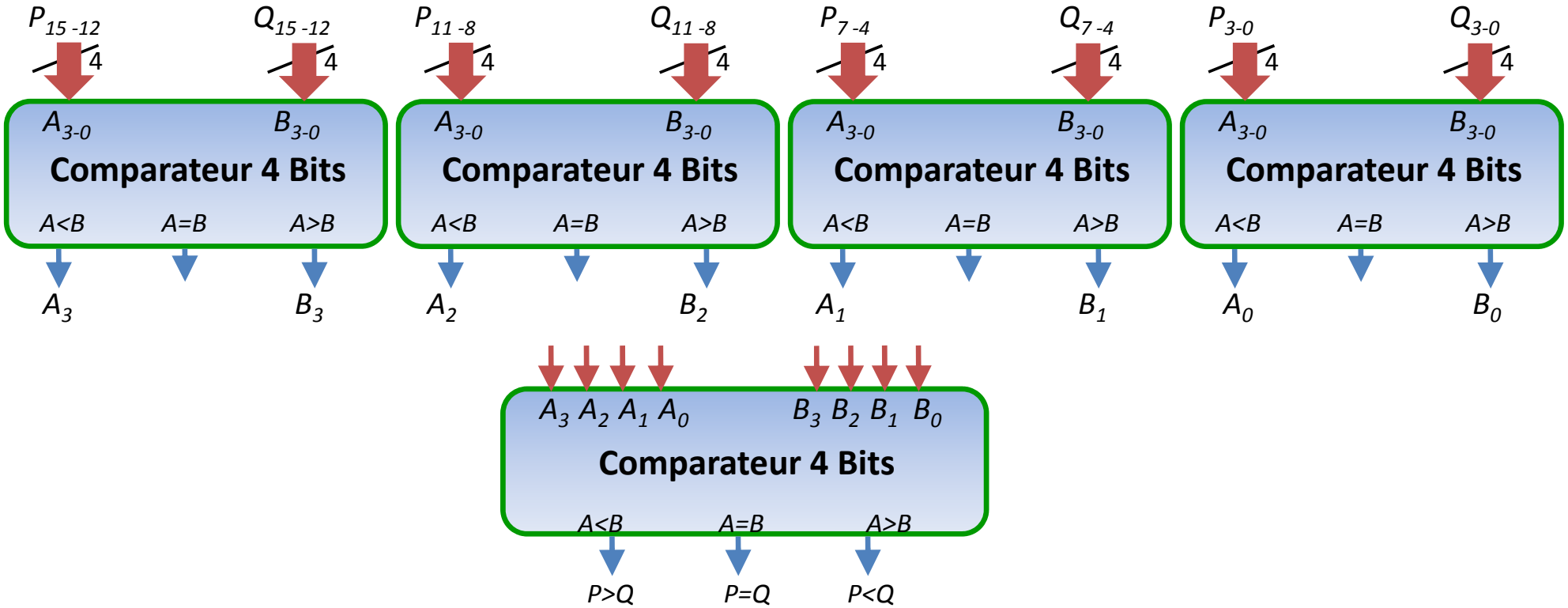
Comparateur de deux nombre P et Q,
De 12 bits chacun.



- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

Le "Comparateur 4 Bits" va être désormais représenté par un schéma bloc condensé comme ci-contre. La notation Signifie un ensemble de 4 fils parallèles appelé bus de largeur 4 bits.
Voici ci-dessous le schéma d'un comparateur de deux Nombre de 16 bits chacun où l'agencement des blocs de base est différent.





- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

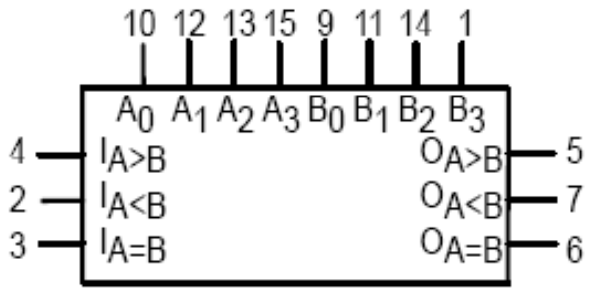
Vous avez remarqué que la mise en cascade des comparateurs précédents est artificielle.

Il existe une variété de comparateurs avec "des entrées et des sorties de mise en cascade" en natif.

TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L

LOGIC SYMBOL

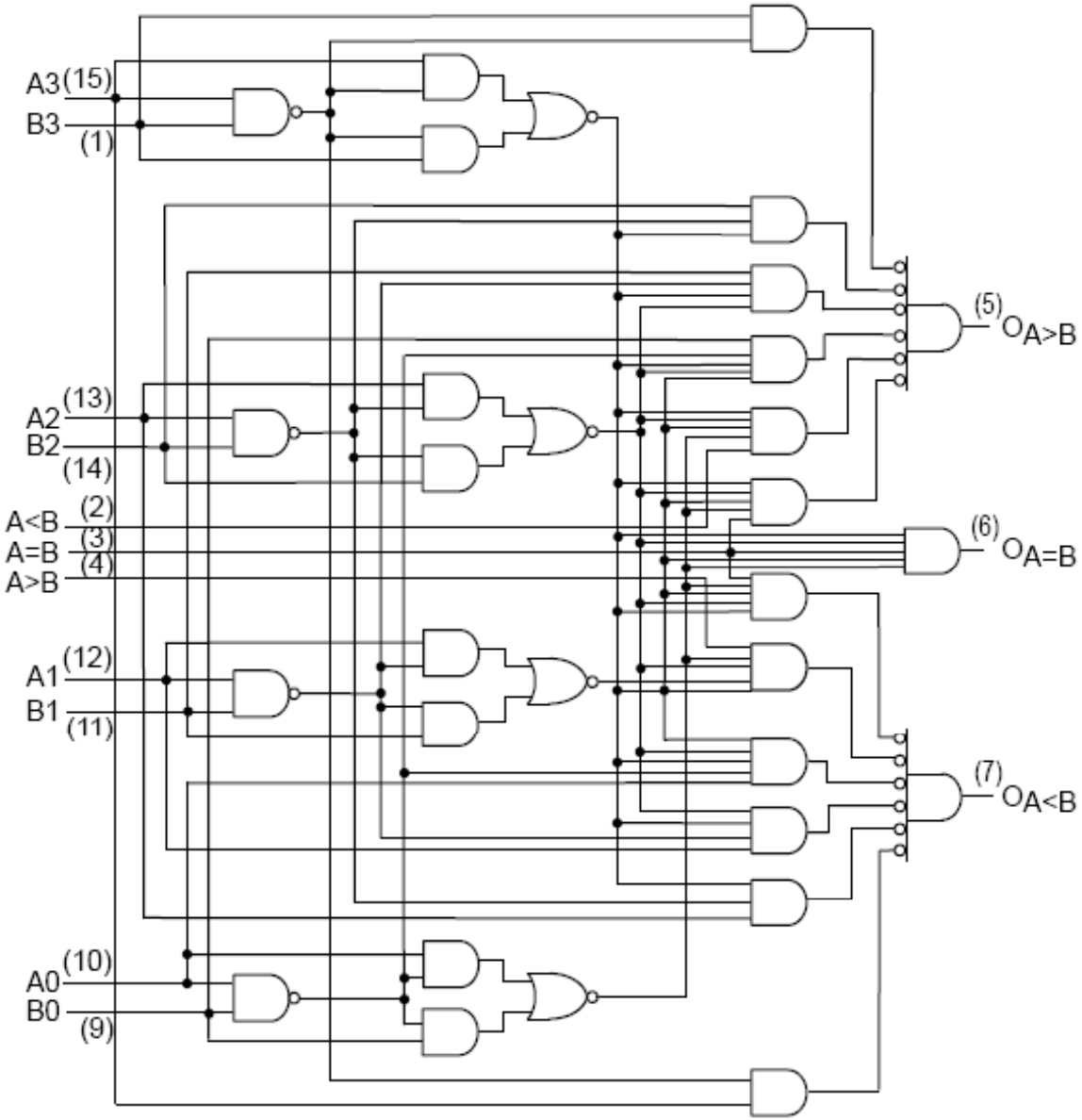


H = HIGH Level
L = LOW Level
X = IMMATERIAL

- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

LOGIC DIAGRAM

SN54/74LS85



- ***Les comparateurs.*** *(de valeurs absolues ou d'entiers naturels).*

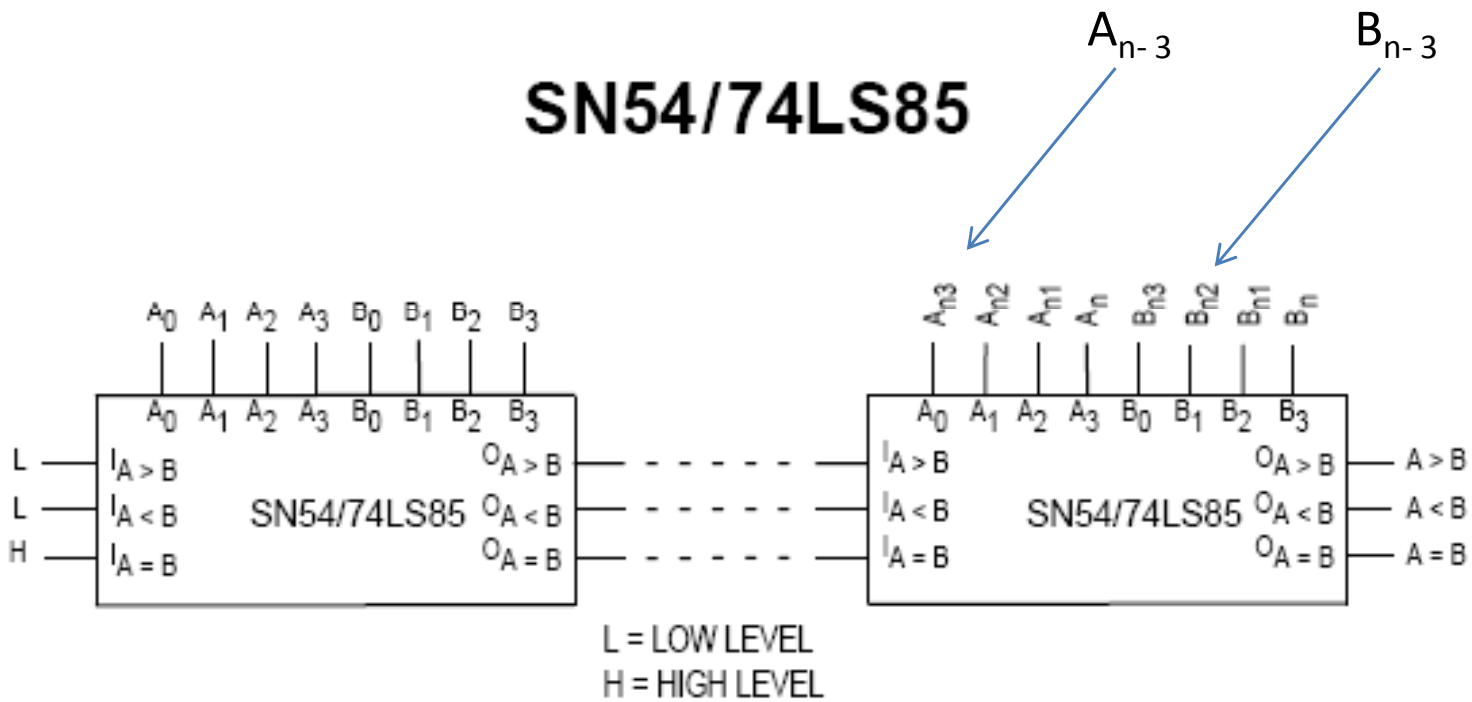
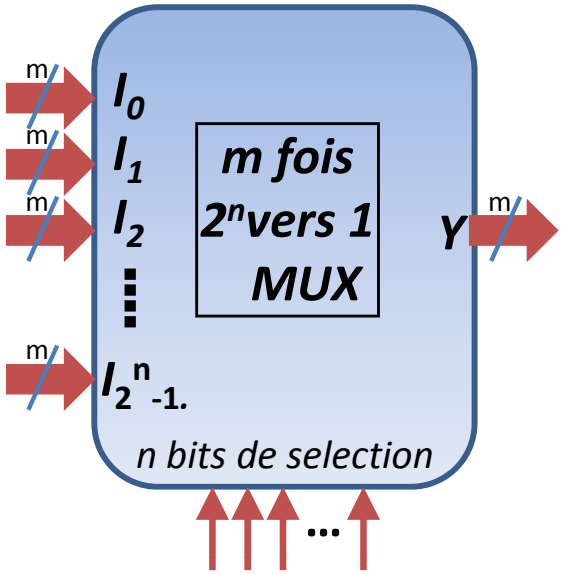


Figure 1. Comparing Two n-Bit Words

- ***Les multiplexeurs.***

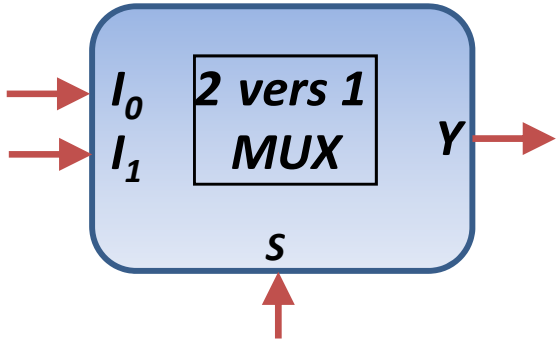
Un circuit multiplexeur permet d'amener à sa sortie ***unique*** une de ses ***multiples entrées*** dites de donnée, choisie par des entrées de ***sélection***. Les entrées de donnée sont sur m bits chacune, la sortie aussi. Dans le cas simple m=1.

Généralement il possède 2^n entrées de données, plus **n** entrées de sélection. Son schéma bloc générique et sa table de vérité simplifiée sont repris ci-après:



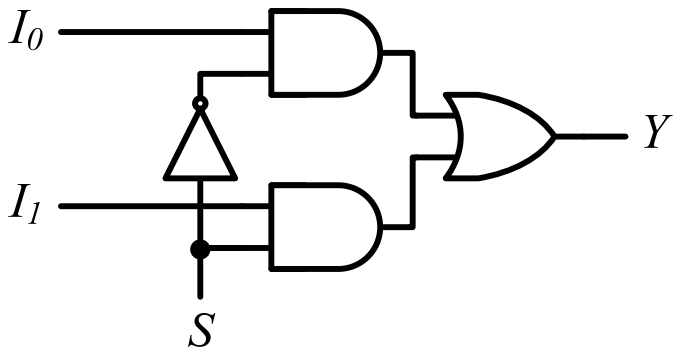
n bits de selection	Y
000	I_0
001	I_1
010	I_2
...	I_i
2^n-1	I_{2^n-1}

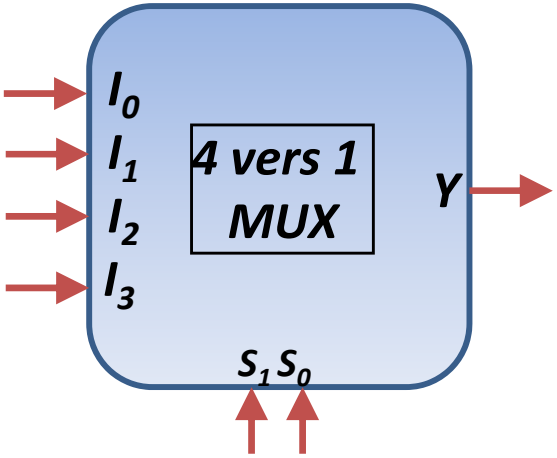
- ***Les multiplexeurs:*** *Le multiplexeur 2 vers 1. Celui 4 vers 1. (m=1)*



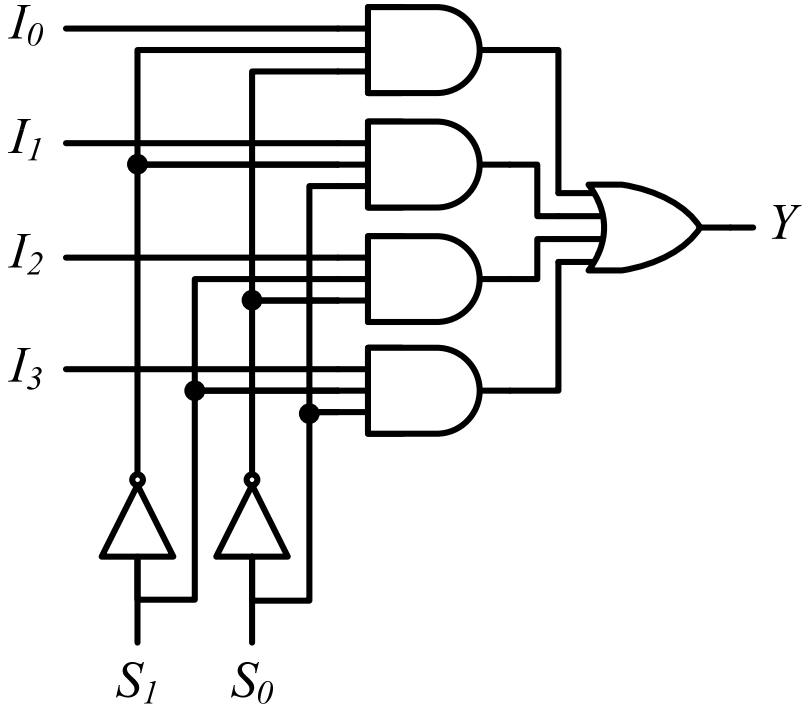
S	Y
0	I_0
1	I_1

$$Y = \overline{S}.I_0 + S.I_1$$





S_1 S_0	Y
00	I_0
01	I_1
10	I_2
11	I_3

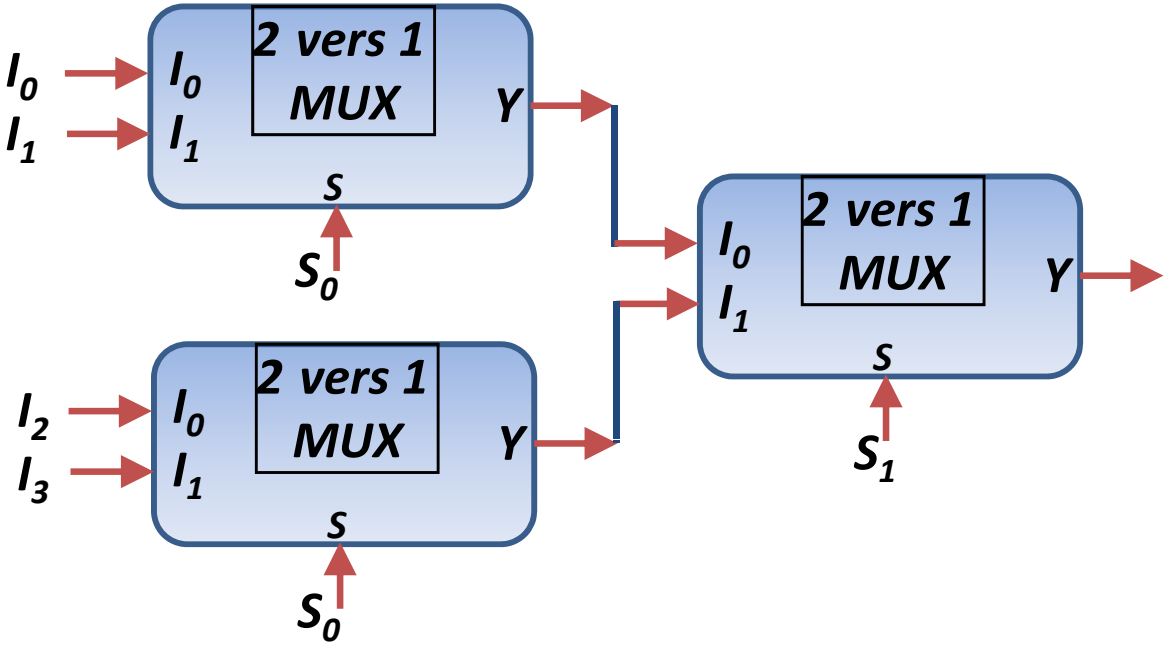


$$Y = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$

- ***Les multiplexeurs.*** *Les Mux plus larges.*

Les multiplexeurs plus larges peuvent être montés à partir de mux plus petits:

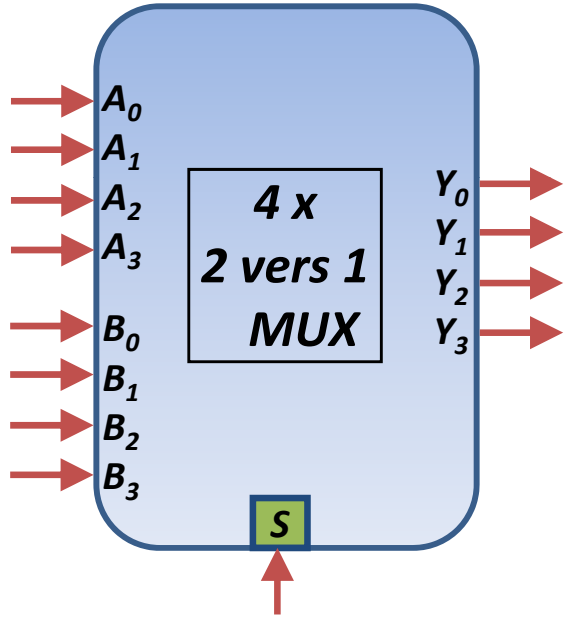
Déjà le "**4 vers 1 mux**" de la page précédente pouvait être câblé à partir de mux de la famille des "**2 vers 1 mux**". Un examen de la table de vérité, montre que l'entrée de sélection S_1 permet de choisir entre le premier groupe (I_0, I_1) ou le second groupe (I_2, I_3). Par ailleurs S_0 permet de choisir un élément du groupe.



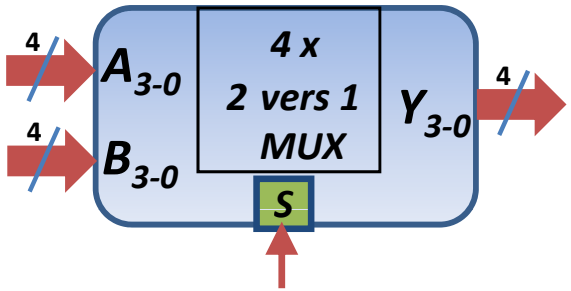
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

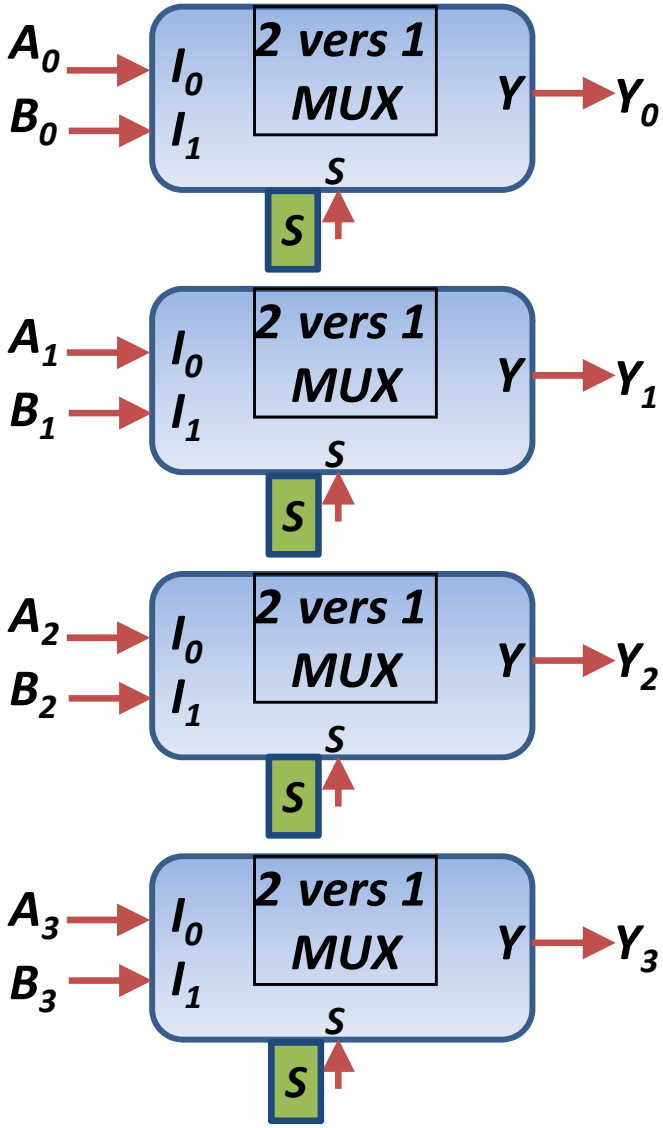
- ***Les multiplexeurs.*** *Les Mux plus larges: Mux 4 fois (2 vers1).*

Les deux schémas bloc sont donnés à gauche.
Le contenu, à base de "***2 vers 1 MUX***" , est donné à droite.



S	Y_{3-0}
0	A_{3-0}
1	B_{3-0}

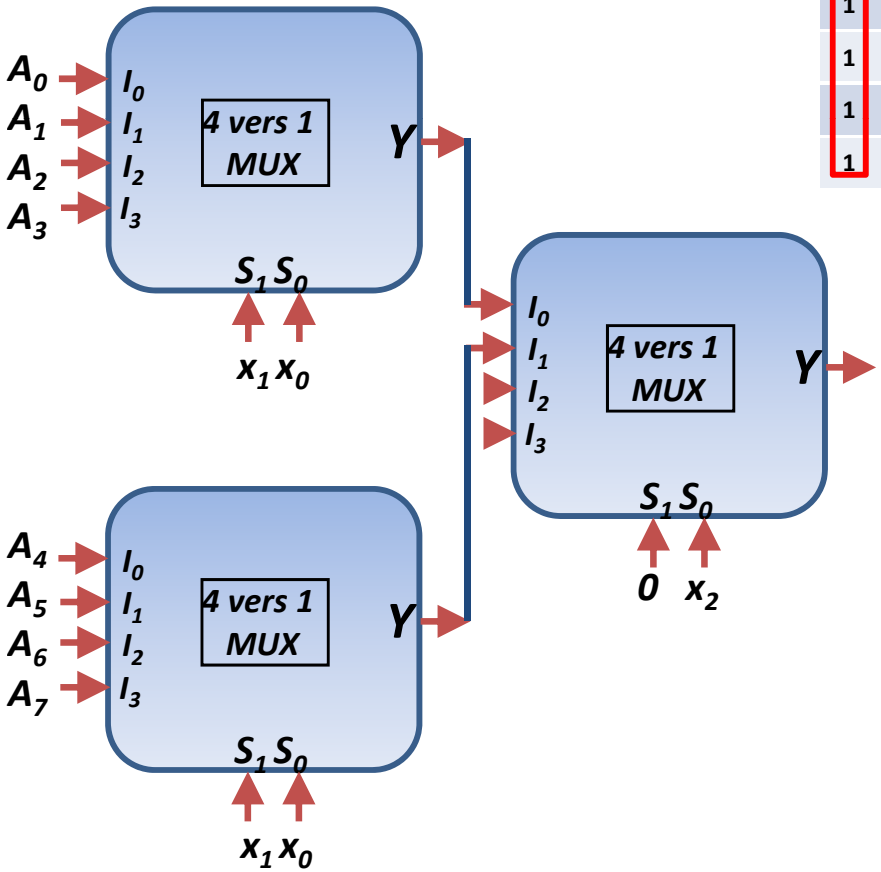


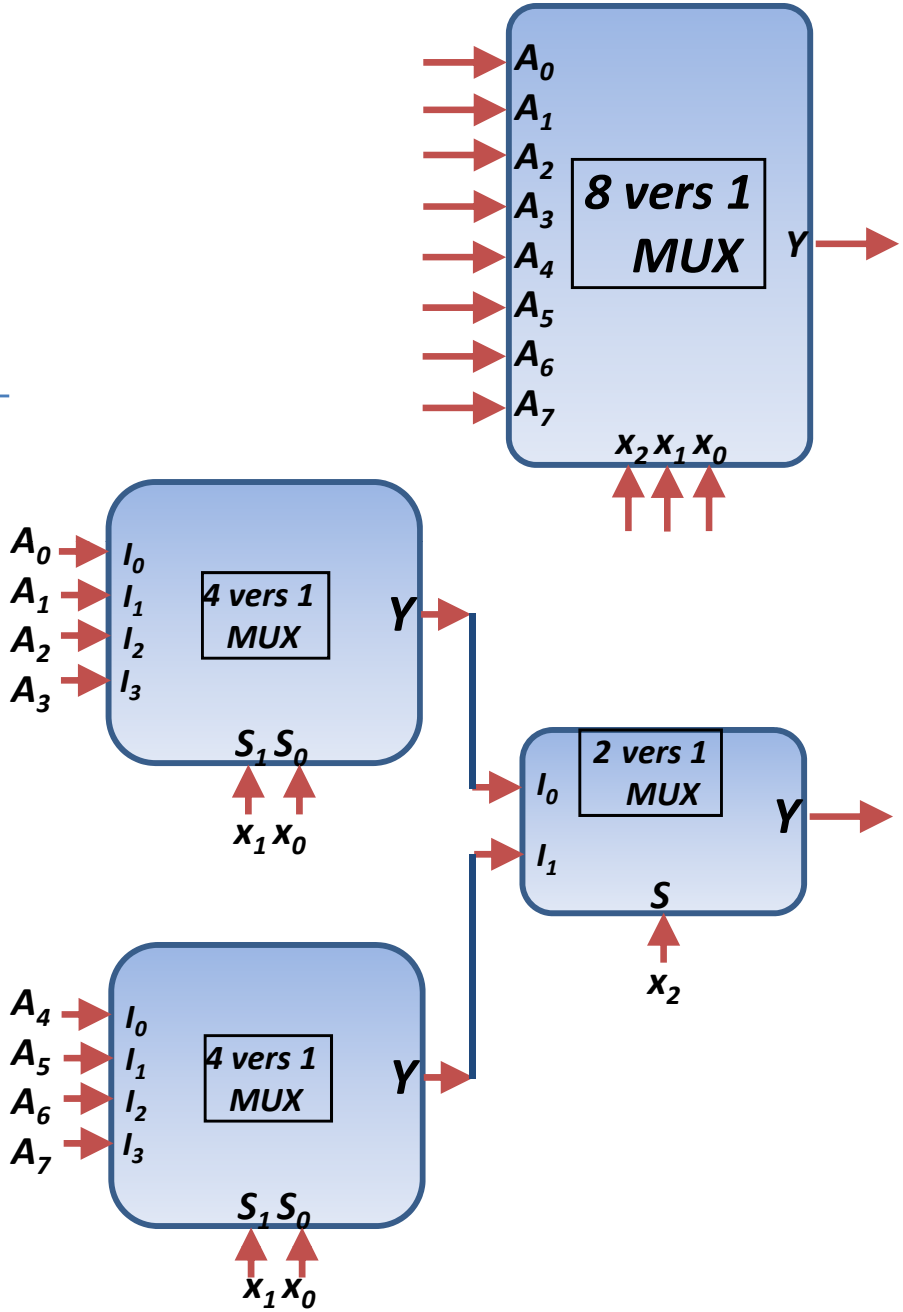


- ***Les multiplexeurs***: *Mux (8 vers1).*

Table de vérité, schéma bloc.
Une 1ere réalisation.
Une deuxième réalisation.

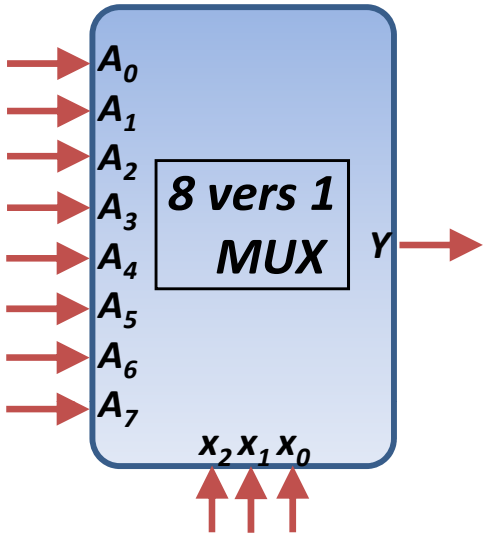
x_2	x_1	x_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7



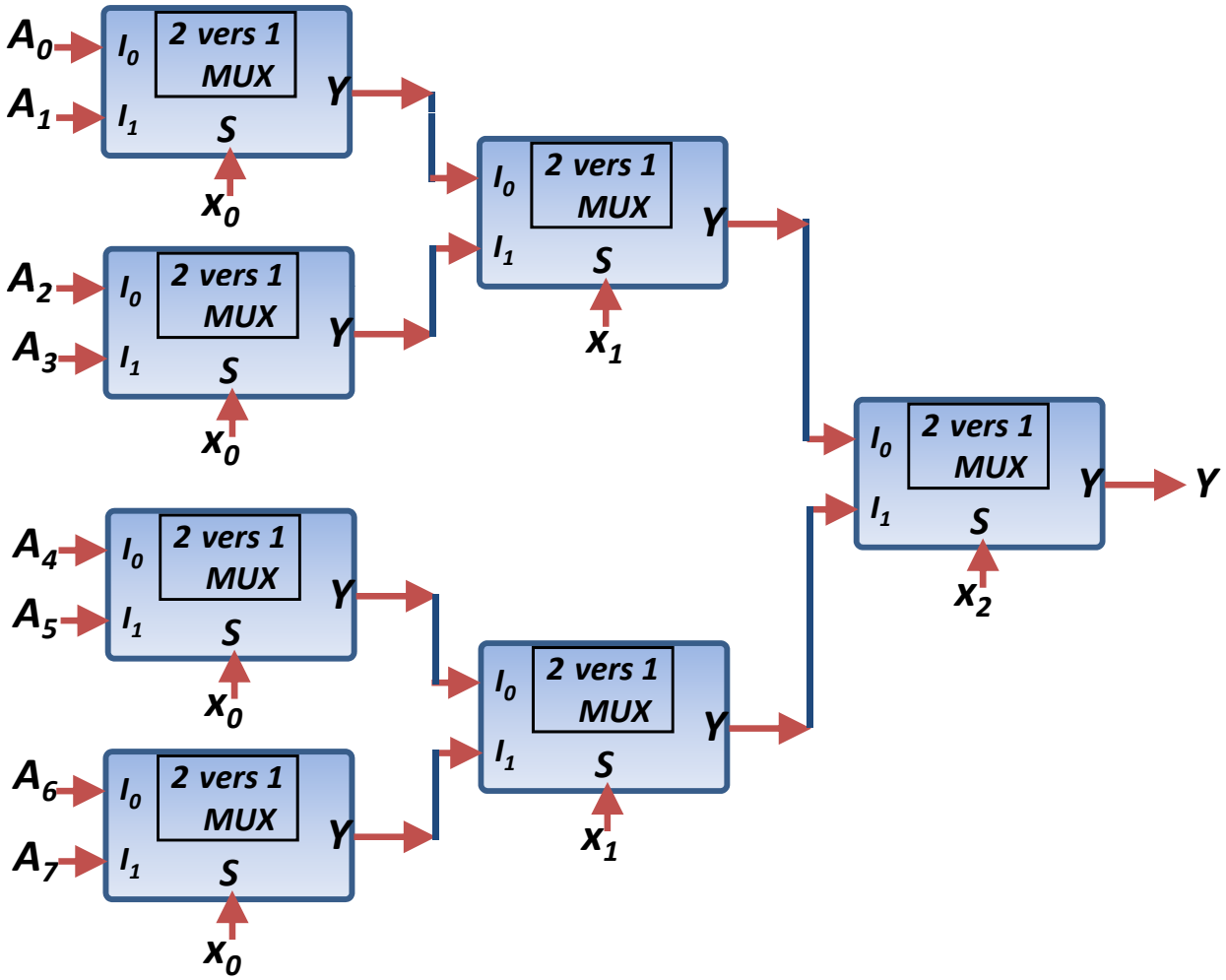


- ***Les multiplexeurs**: Mux (8 vers1).*

Table de vérité, schéma bloc.
Une 3ieme réalisation.



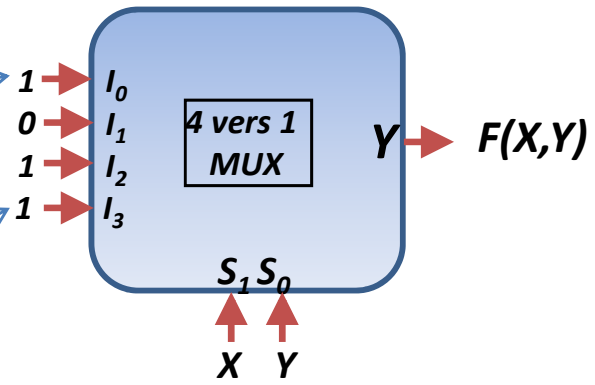
$x_2 x_1 x_0$			Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7



- ***Les multiplexeurs**: Applications.*

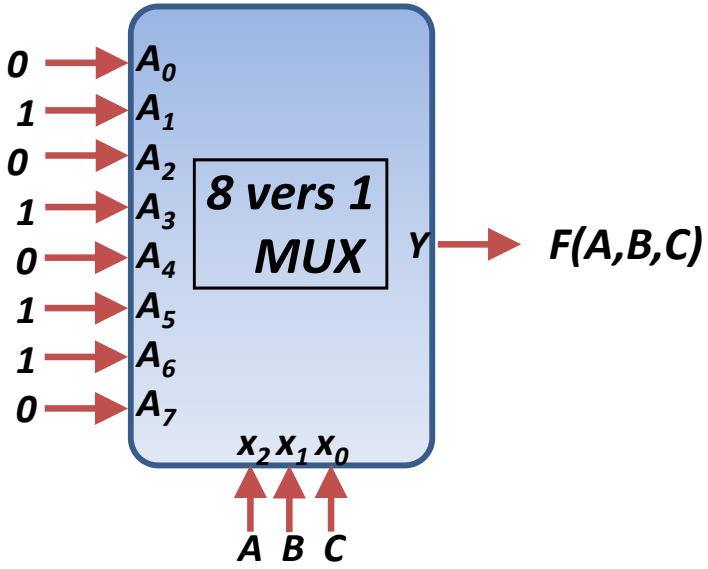
Utilisation des multiplexeurs pour réaliser des fonctions.

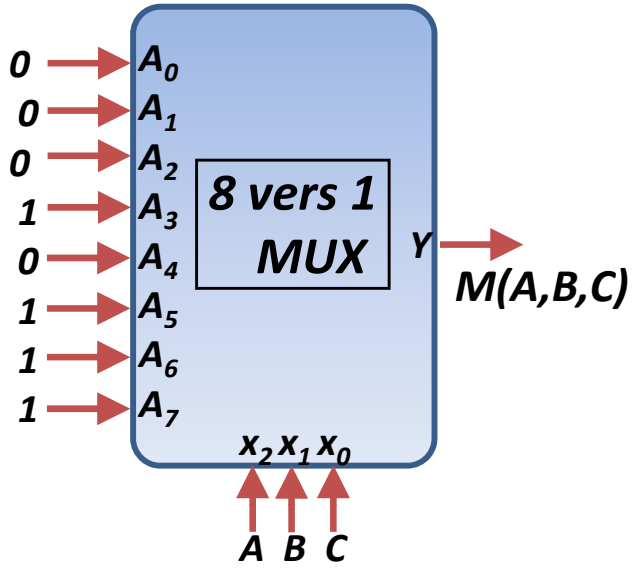
X	Y	F(X,Y)
0	0	1
0	1	0
1	0	1
1	1	1



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





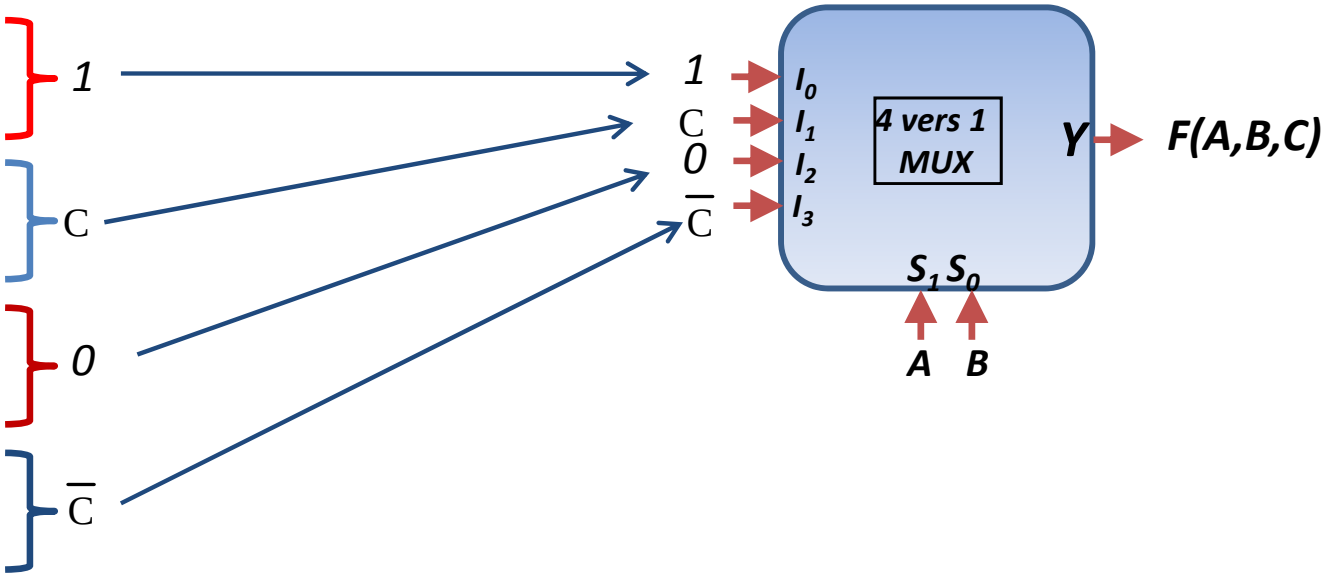
- ***Les multiplexeurs**: Applications.*

Utilisation des multiplexeurs pour réaliser des fonctions.

F(A,B,C) = Σ (0,1,3,6)

$$F(A,B,C) = \underbrace{\overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C}_{1} + \underbrace{\overline{A}.B.C}_{C} + \underbrace{A.B.\overline{C}}_{\overline{C}}$$

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

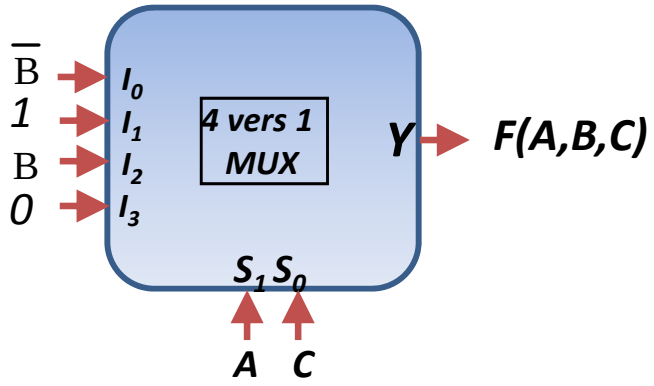


- ***Les multiplexeurs**: Applications.*

Utilisation des multiplexeurs pour réaliser des fonctions.

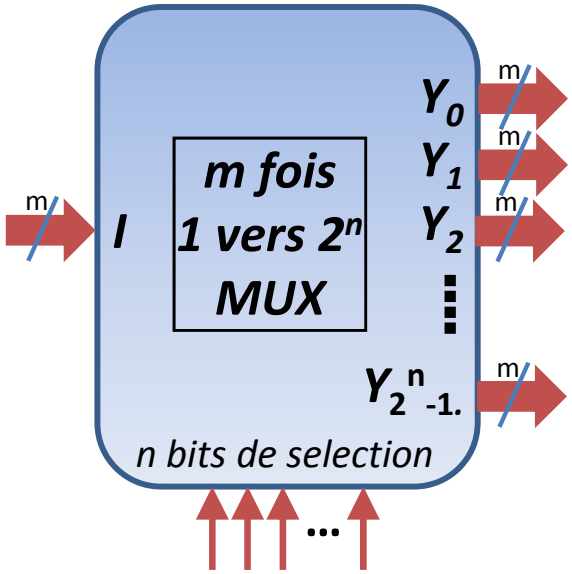
F(A,B,C) = Σ (0,1,3,6) on peut prendre A et C comme bits de selection.

A	B	C	F(A,B,C)	
0	0	0	**1**	$\overline{B}$
0	0	1	**1**	*1*
0	1	0	**0**	$\overline{B}$
0	1	1	**1**	*1*
1	0	0	**0**	B
1	0	1	**0**	*0*
1	1	0	**1**	B
1	1	1	**0**	*0*



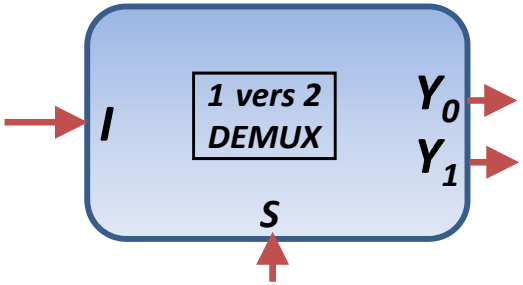
- ***Les démultiplexeurs.***

Remplissent la fonction inverse d'un multiplexeur.
Ils ont généralement une entrée de données (sur m bits) et l'envoie vers une sortie parmi 2^n ,en fonction de l'état des n bits de sélection. Cas particulier (m=1).



n bits de selection	Y_{2^n-1}	Y_i	Y_2	Y_1	Y_0
000	0	0	0	0	*I*
001	0	0	0	*I*	0
010	0	0	*I*	0	0
...	0	*I*	0	0	0
2^n-1	*I*	0	0	0	0

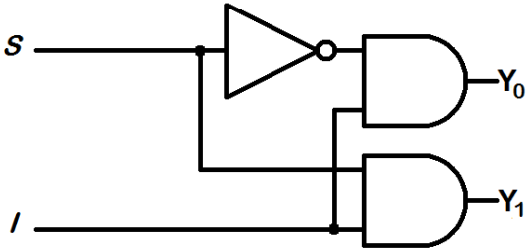
- ***Les démultiplexeurs*. Demux (1 vers 2) & (1 vers 4)**

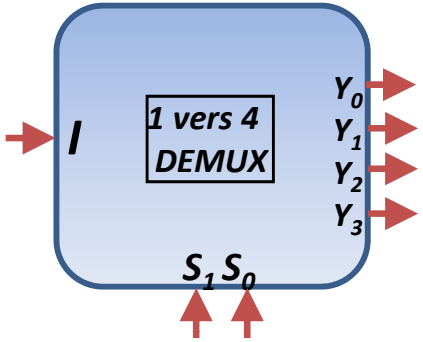


I	S	Y_1	Y_0
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

$$Y_1 = S.I$$

$$Y_0 = \overline{S}.I$$





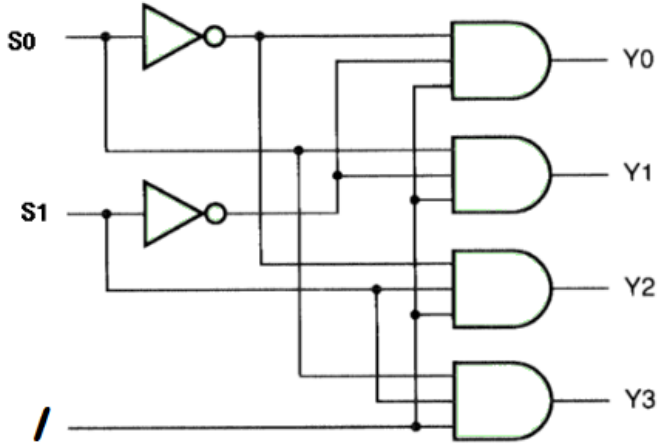
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	/
0	1	0	0	/	0
1	0	0	/	0	0
1	1	/	0	0	0

$$Y_0 = \overline{S_1}.\overline{S_0}.I$$

$$Y_1 = \overline{S_1}.S_0.I$$

$$Y_2 = S_1.\overline{S_0}.I$$

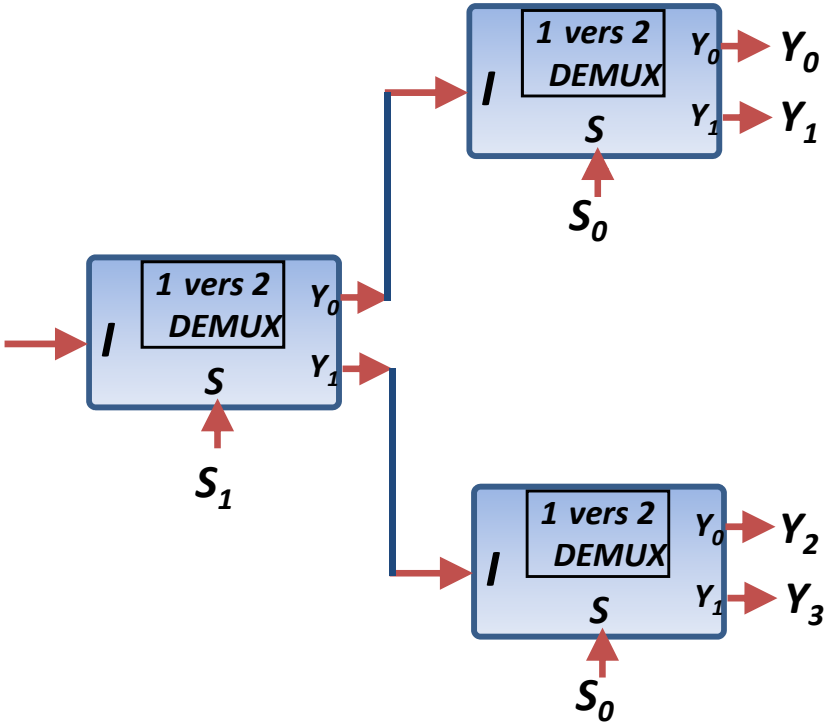
$$Y_3 = S_1.S_0.I$$

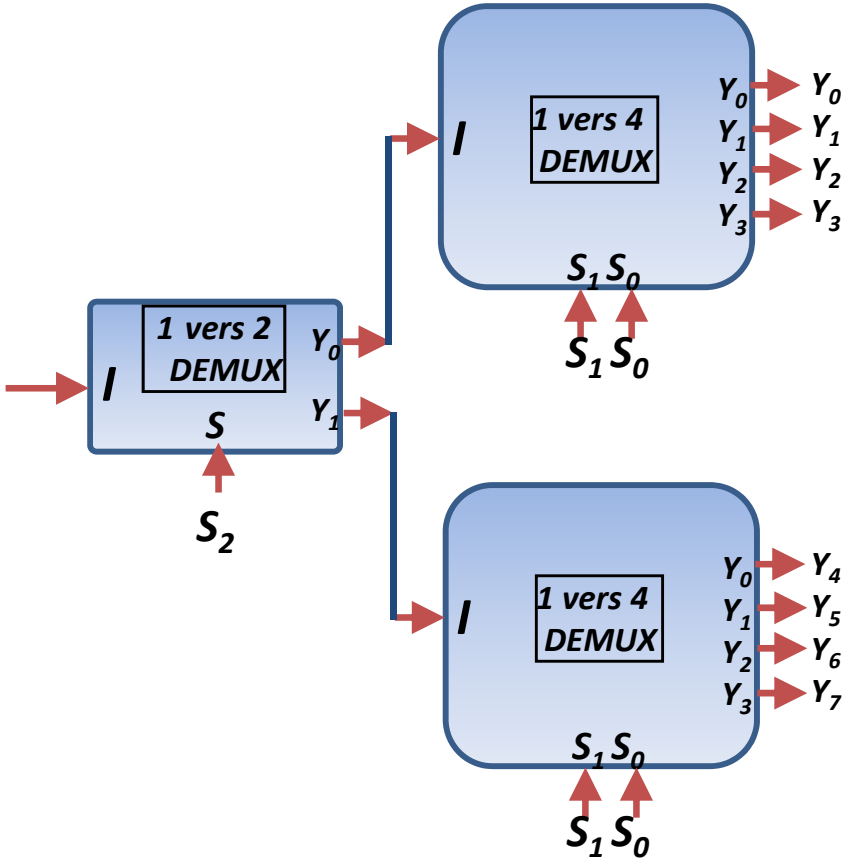


- ***Les démultiplexeurs.***

demux (1 vers 4) à base de (1 vers 2) (Gauche)

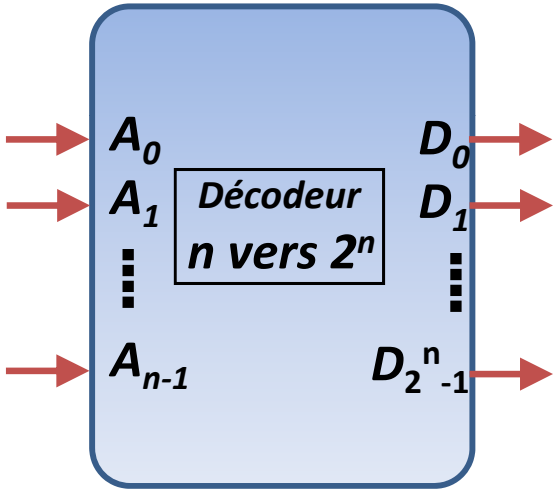
demux (1 vers 8) à base de (1 vers 2) et de (1 vers 4) (droite).





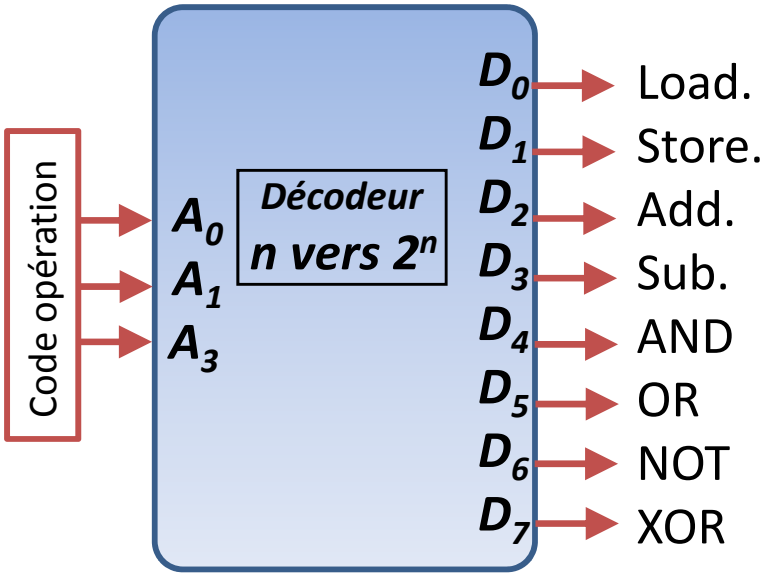
- ***Les décodeurs.***

Le décodeur est un circuit qui permet de convertir une configuration binaire (le code sur n bits) en une seule sortie active sur ses (au plus) 2^n sorties.
Le décodeur est aussi appelé générateur de mintermes de fonctions logiques.



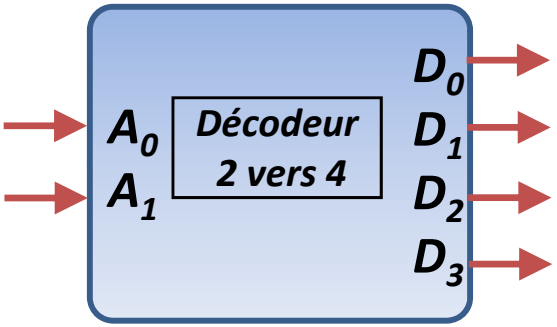
$A_{n-1} \dots A_1 A_0$	$D_{2^n-1} \dots D_2 D_1 D_0$
***i* en binaire.**	Seule la sortie D_i est à 1

Exemple: Le décodeur d'instructions.



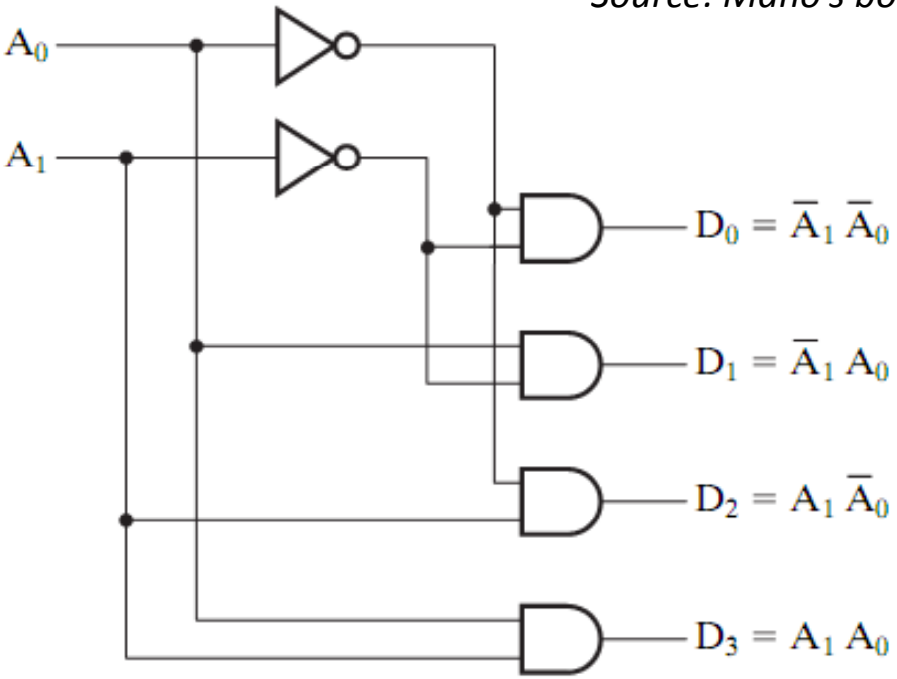
- ***Les décodeurs.*** *Le décodeur 2 vers 4.*

Deux entrées A_1, A_0.
Quatre sorties D_0, D_1, D_2, D_3.
Schéma bloc, table de vérité, schéma interne:



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

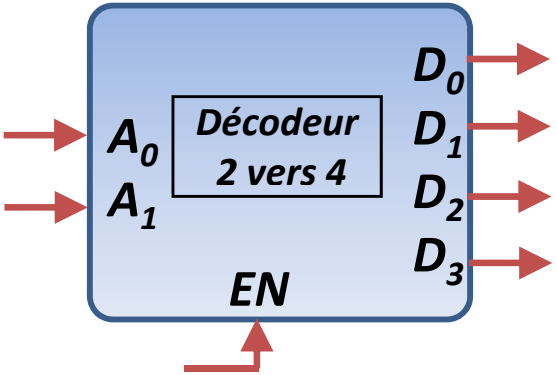
Source: Mano's book



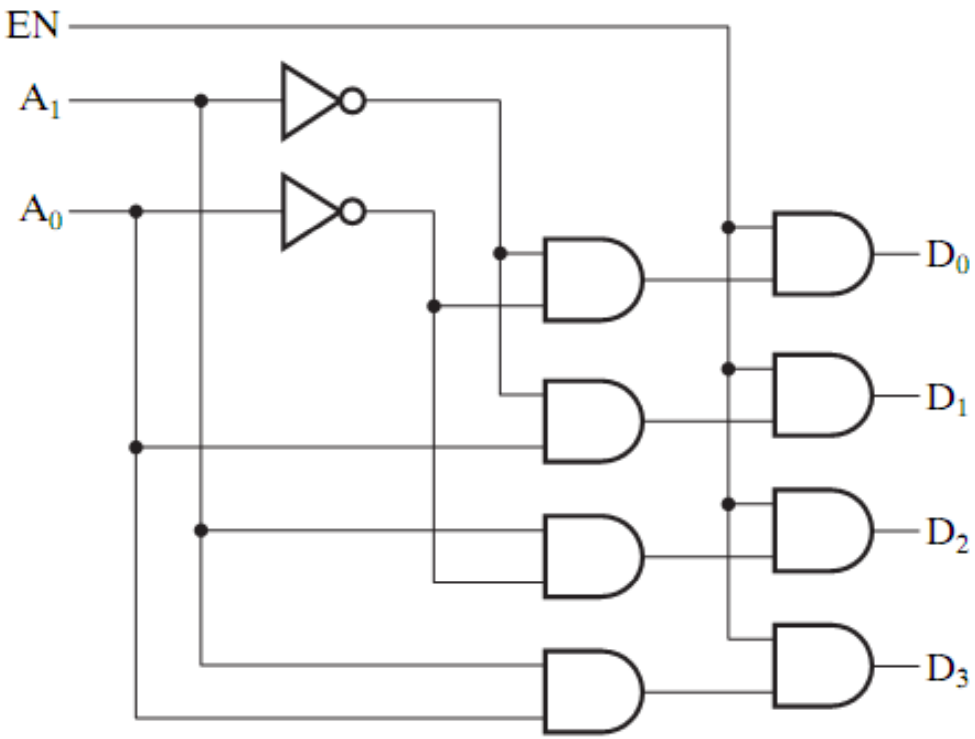
Remarquez que les D_i sont tous les mintermes de la fonction $F(A_0, A_1)$.

- ***Les décodeurs.*** *Le décodeur 2 vers 4 avec l'entrée "E: Enable".*

Deux entrées A_1, A_0 et l'entrée *"E: Enable"*.
Quatre sorties D_0, D_1, D_2, D_3.
Schéma bloc, table de vérité, schéma interne:



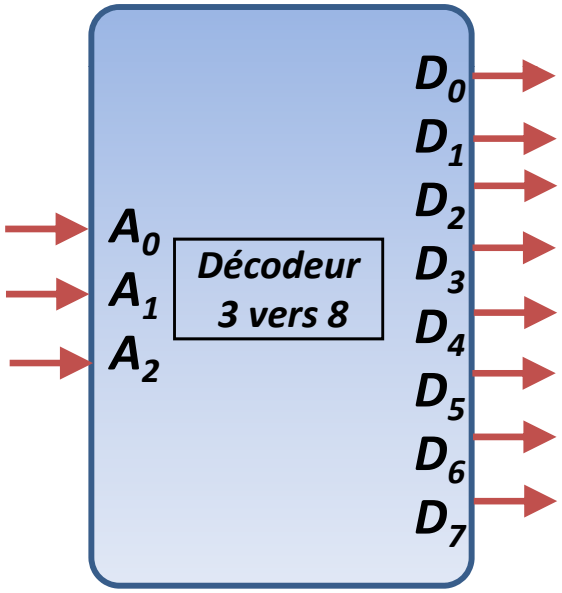
EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



Source: This is from Mano's book

- ***Les décodeurs.*** *Le décodeur 3 vers 8.*

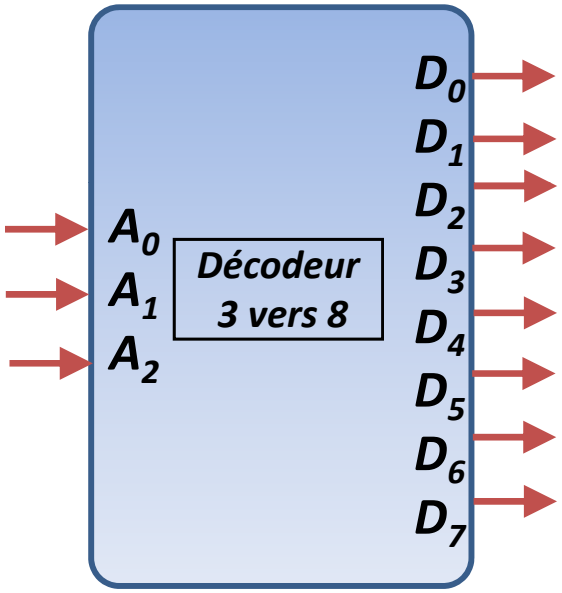
Schéma bloc et table de vérité

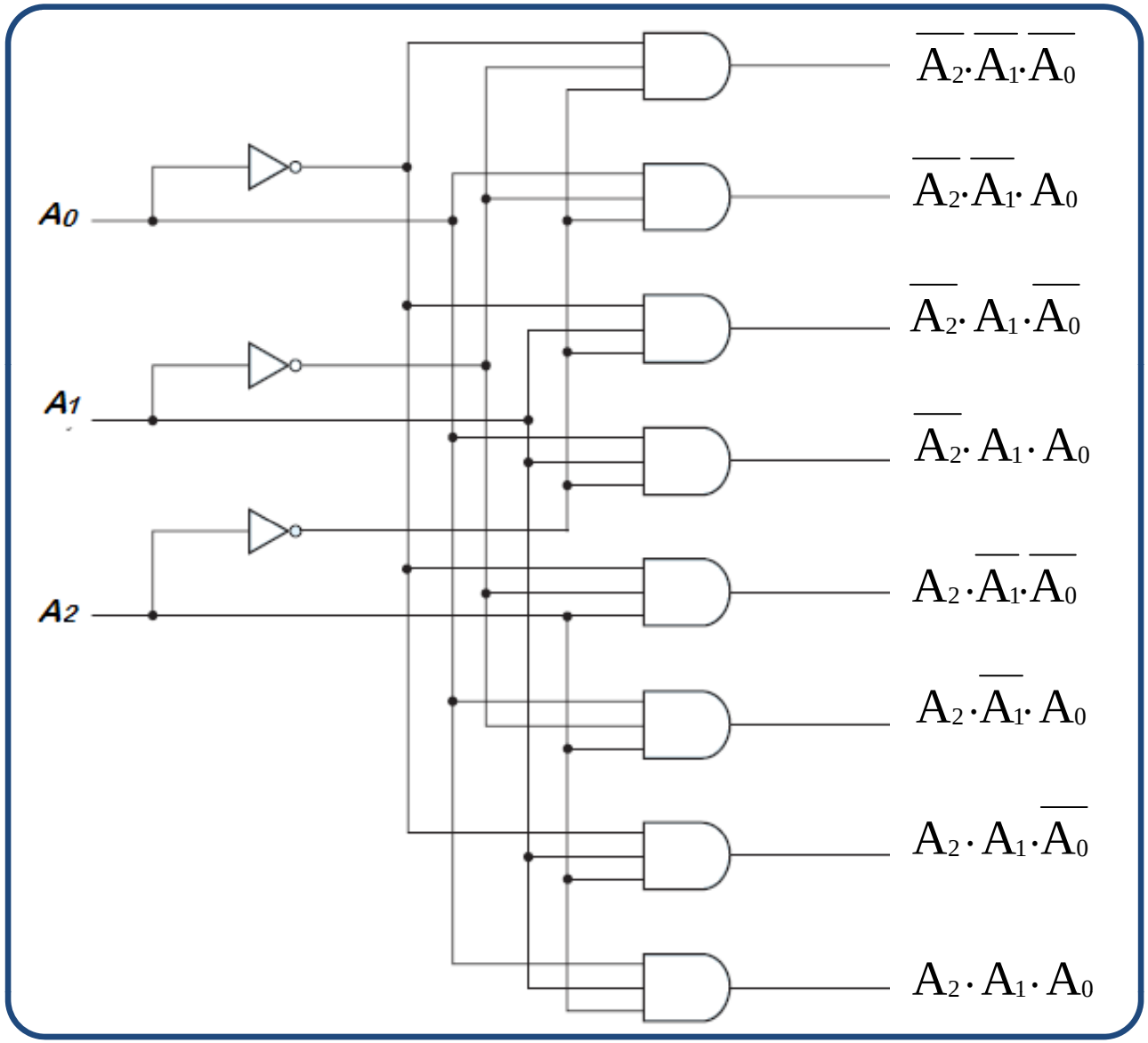


A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

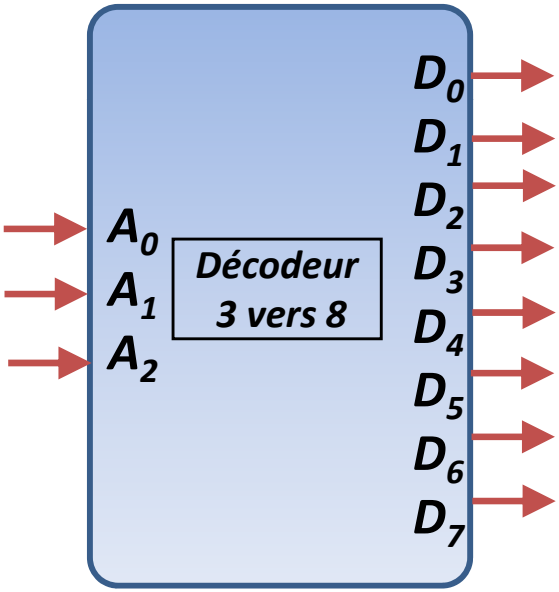
- ***Les décodeurs.*** *Le décodeur 3 vers 8.*

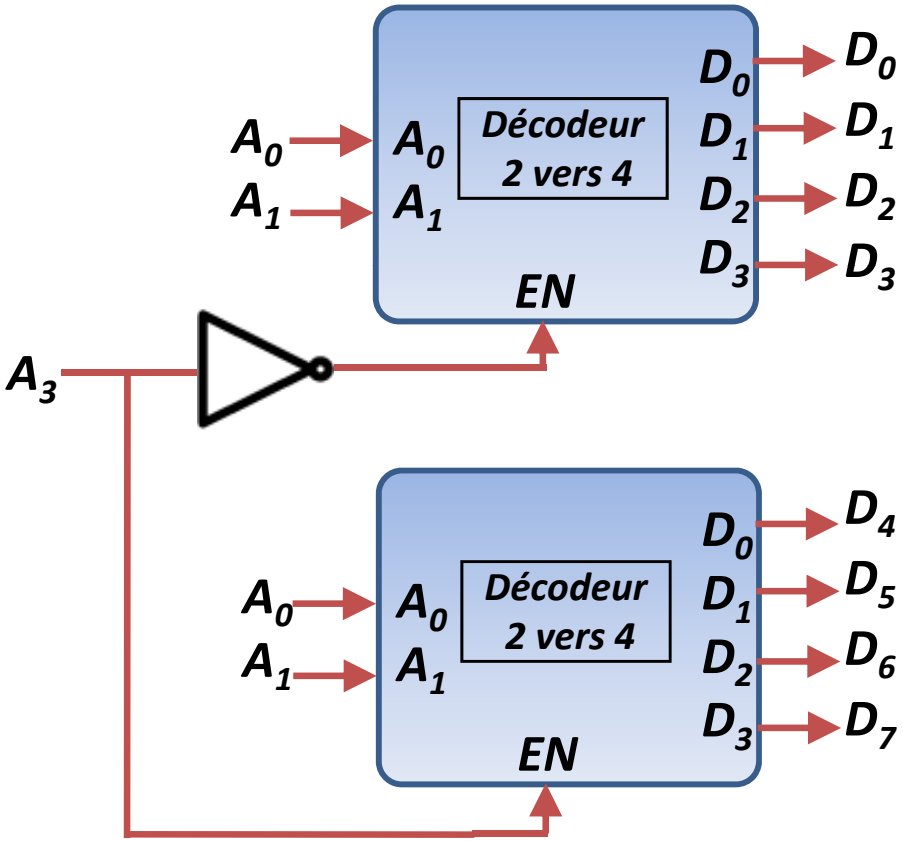
Schéma bloc et schéma interne.





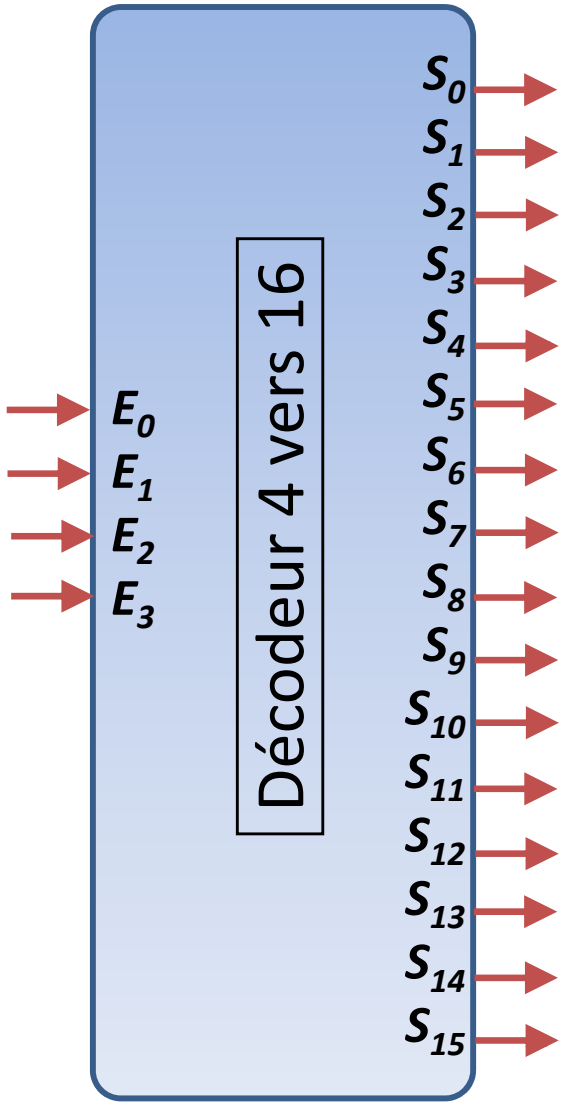
- ***Les décodeurs.*** *Le décodeur 3 vers 8 à base de décodeur 2 vers 4.*

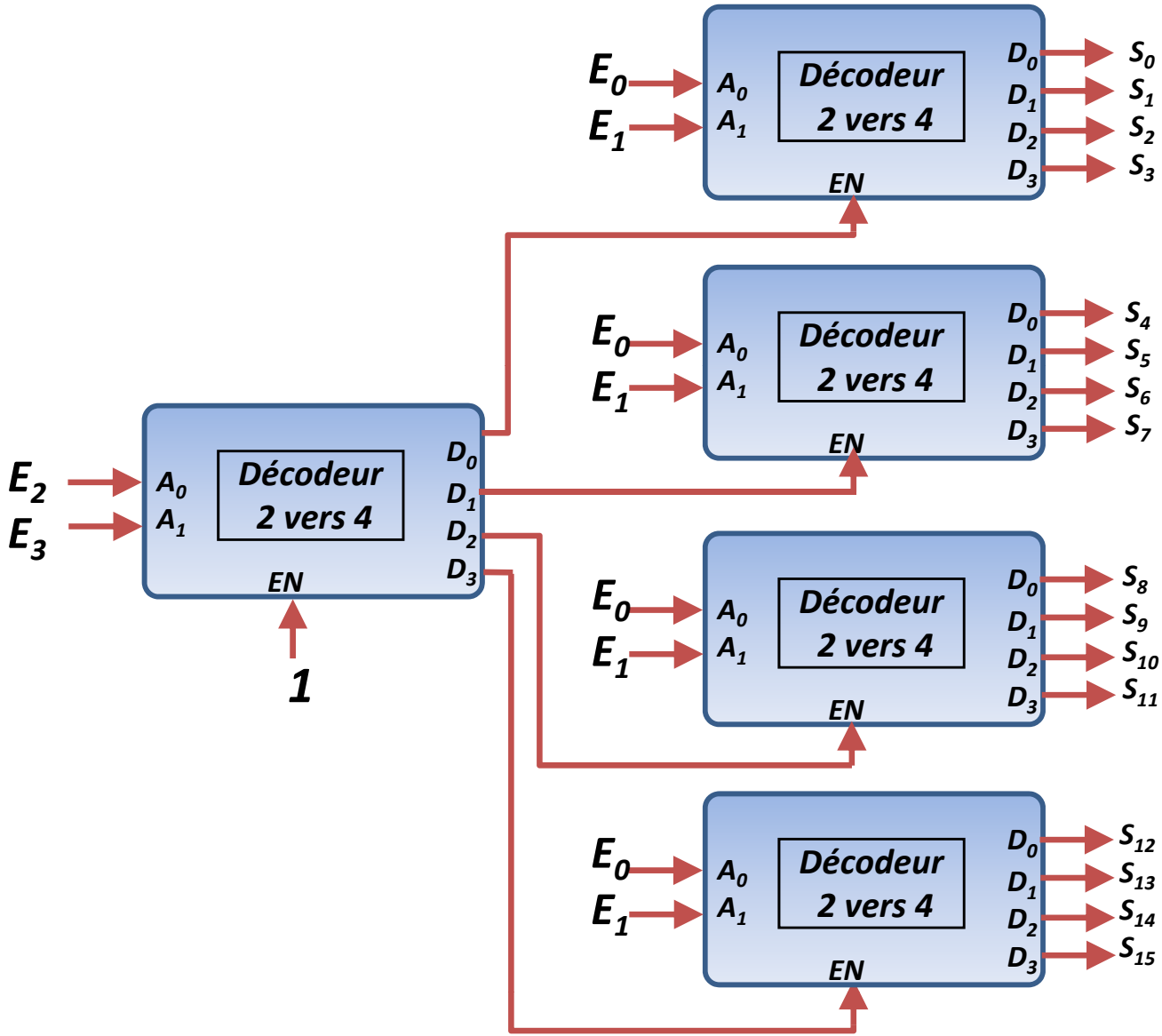




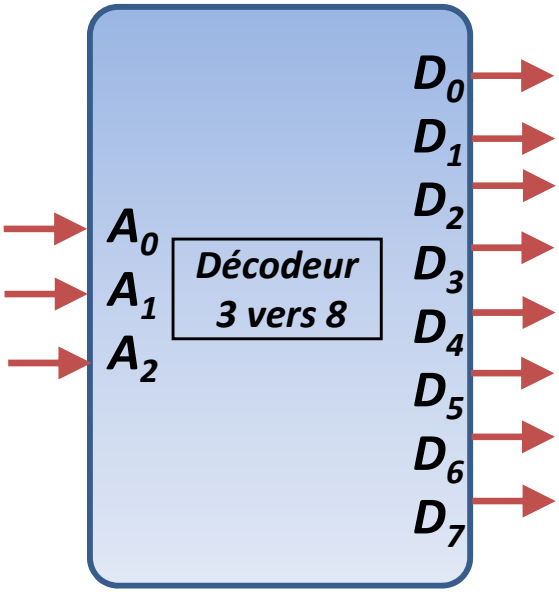
- ***Les décodeurs.*** *Le décodeur 4 vers 16.* *(à base de décodeurs 2 vers 4).*

Schéma bloc





- ***Les décodeurs.*** ***Réalisation de fonctions logiques à base de décodeurs.***



Soit à réaliser les fonctions suivantes:
F1(x,y,z) = Σ(1,2,4,7)
F2(x,y,z) = Σ(3,5,6,7)

Comme les sorties du décodeur sont des mintermes, il suffit de créer un somme logique pour générer une fonction logique.

F1(x,y,z) = $D_1 + D_2 + D_4 + D_7$
F2(x,y,z) = $D_3 + D_5 + D_6 + D_7$

La réalisation de fonction à base de décodeurs est intéressante lorsqu'on à réaliser plusieurs fonctions avec les mêmes variables d'entrée et que chaque fonction n'exploite que très peu de mintermes.
On peut rechercher à utiliser la fonction inverse et associer les mintermes dans un NOR.

- ***Les Encodeurs.***
  - Remplissent la fonction inverse d'un décodeur.
  - "Priority encoder" et sortie pour dire qu'il y a au moins une entrée positionnée.

- ***Les transcodeurs.*** Passage d'un code à un autre.

  Exemple:

  ✓ ***Binaire - excess 3 ou l'inverse.***

- ***Les décodeurs spéciaux.***

  ✓ ***BCD - 7 segments.***

Logique combinatoire et séquentielle.

Les circuits logiques sont de deux types: les ***combinatoires*** et les ***séquentiels.***
Les circuits combinatoires ont fait l'objet du cours précédent.

B) Les circuits logiques séquentiels:

Ce sont ceux où les sorties à l'instant ***t*** dépendent non seulement des entrées à l'instant ***t*** *mais aussi des entrées précédentes ou de leur* ***état interne*** . Ils ont la faculté de ***mémoire***.

Ils sont souvent ***synchrones***: leur état interne ne change, et par conséquent leurs sorties ne changent que sur autorisation d'un signal particulier qui marque les instants significatifs dans le temps.

Les circuits logiques séquentiels sont bâtis au moyen des circuits combinatoires et des éléments, basiques, de mémorisation appelés ***bascules***, ***bistables*** ou ***flip-flops.***

Ces ***bascules***, ***bistables*** ou ***flip-flops***, à ***deux états*** seulement, constituent les briques de base, les composants fondamentaux des circuits logiques séquentiels. Ils sont assimilés à des éléments de mémoire, de nature à stocker l'état interne de ces circuits.

Cette partie du cours porte sur l'étude des bistable les plus utilisés, et à leur emploi dans quelques circuits séquentiels.

Cette partie porte sur l'examen des bistables suivants:

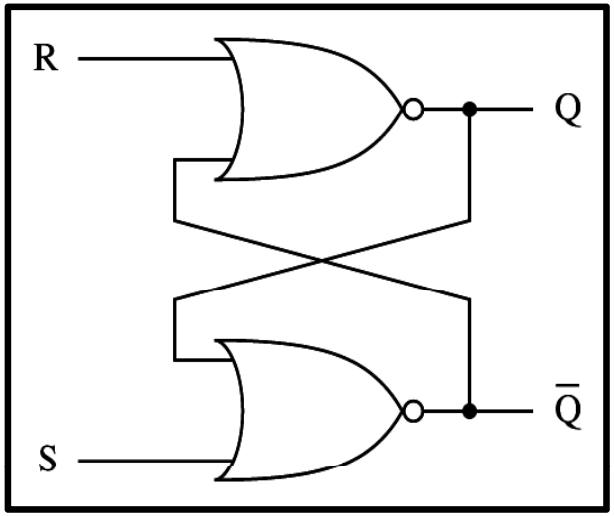
✓– *Le bistable RS dans sa version asynchrone et synchrone.*
✓– *Le bistable D.*
✓– *Le bistable JK.*

Elle porte également sur les circuits logiques séquentiels suivants:

✓– Les registres.
✓– Les bus et l'état haute impédance.
✓– Les registres à décalage.
✓– Les générateurs de séquence en général,
les compteurs/décompteurs en particulier.

B-1: Le bistable RS asynchrone.

Son schéma typique est donné ci-contre :



Rappel sur l'opérateur NOR:

$\overline{1 + a} = 0$ & $\overline{0 + a} = \overline{a}$

Considérons les 4 cas possibles pour R et S.

<u>Cas 1</u>: R=0, S=1.

La sortie $\overline{Q} = \overline{S + Q} = \overline{1 + Q} = 0$

et la sortie $Q = \overline{R + \overline{Q}} = \overline{0 + 0} = 1$

Le circuit évolue vers l'état stable : $Q = 1, \overline{Q} = 0.$

<u>Cas 2</u>: R=1, S=0.

La sortie $Q = \overline{R + \overline{Q}} = \overline{1 + \overline{Q}} = 0$

et la sortie $\overline{Q} = \overline{S + Q} = \overline{0 + 0} = 1$

Le circuit évolue vers l'état stable : $Q = 0, \overline{Q} = 1.$

<u>Cas 3</u>: R=1, S=1.

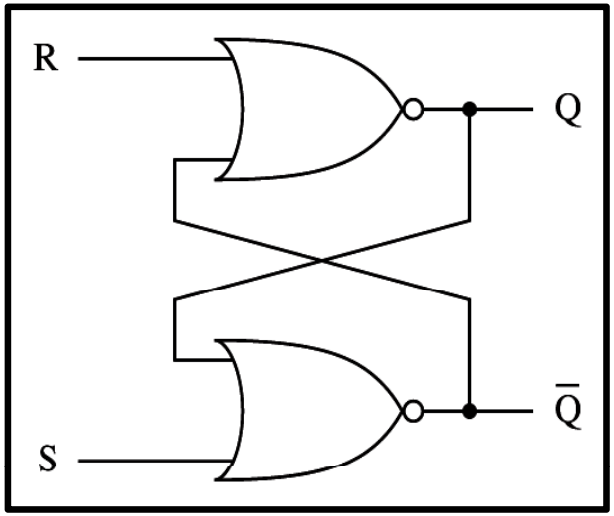
La sortie $Q = \overline{R + \overline{Q}} = \overline{1 + \overline{Q}} = 0$

et la sortie $\overline{Q} = \overline{S + Q} = \overline{1 + Q} = 0$

Le circuit évolue vers l'état stable $Q = 0, \overline{Q} = 0.!!$

B-1: Le bistable RS asynchrone.

$\overline{1 + a} = 0$ & $\overline{0 + a} = \overline{a}$



<u>Cas 4</u>: R=0, S=0.

La sortie $Q = \overline{R + \overline{Q}} = \overline{0 + \overline{Q}} = \overline{\overline{Q}} = Q$

et la sortie $\overline{Q} = \overline{S + Q} = \overline{0 + Q} = \overline{Q}$

Le circuit garde son état.

En résume, si on considère que l'état du bistable est matérialisé par la sortie Q, alors les entrées R et S ont les effets suivants:

***R=0, S=0**, le bistable mémorise son état.*

***R=0, S=1**, le bistable est mis à 1. **S** abréviation de **Set**.*

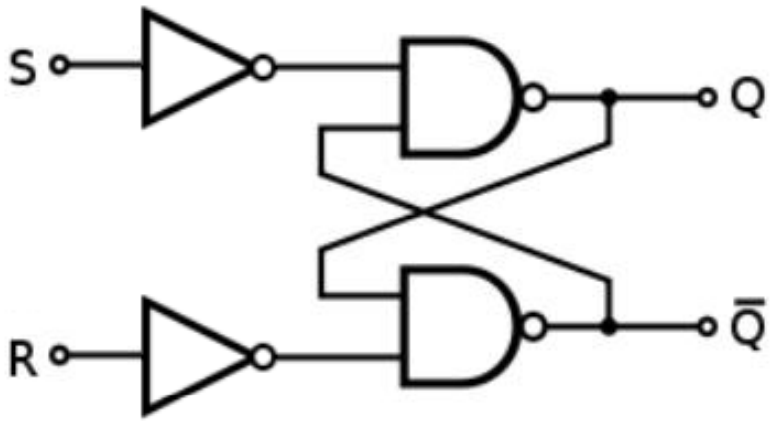
***R=1, S=0**, le bistable est mis à 0. **R** abréviation de **Reset**.*

***R=1, S=1**, combinaison interdite pas seulement parce que les deux sorties ne sont plus complémentaires, mais surtout à cause du fait que le départ de cet état vers la configuration (**R=0, S=0**) engendre une sortie aléatoire.*

B-1: Le bistable RS asynchrone.

Fait à base de portes NAND.

On peut faire fonctionner ce circuit comme précédemment et aboutir au même résultat suivant:



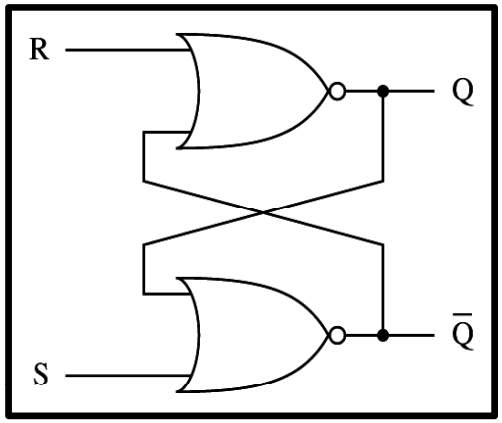
***R=0, S=0**, le bistable mémorise son état.*

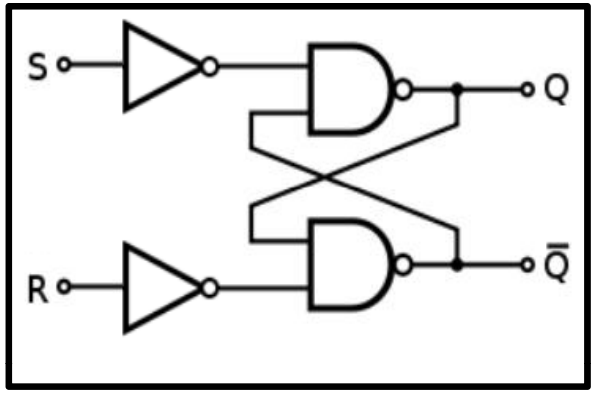
***R=0, S=1**, le bistable est mis à 1. **S** abréviation de **Set**.*

***R=1, S=0**, le bistable est mis à 0. **R** abréviation de **Reset**.*

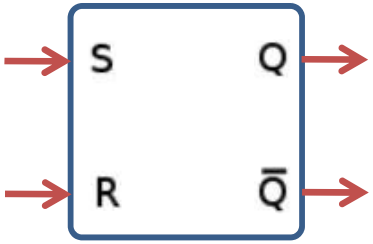
***R=1, S=1**, combinaison interdite, pour les mêmes raisons.*

B-1: Le bistable RS asynchrone.





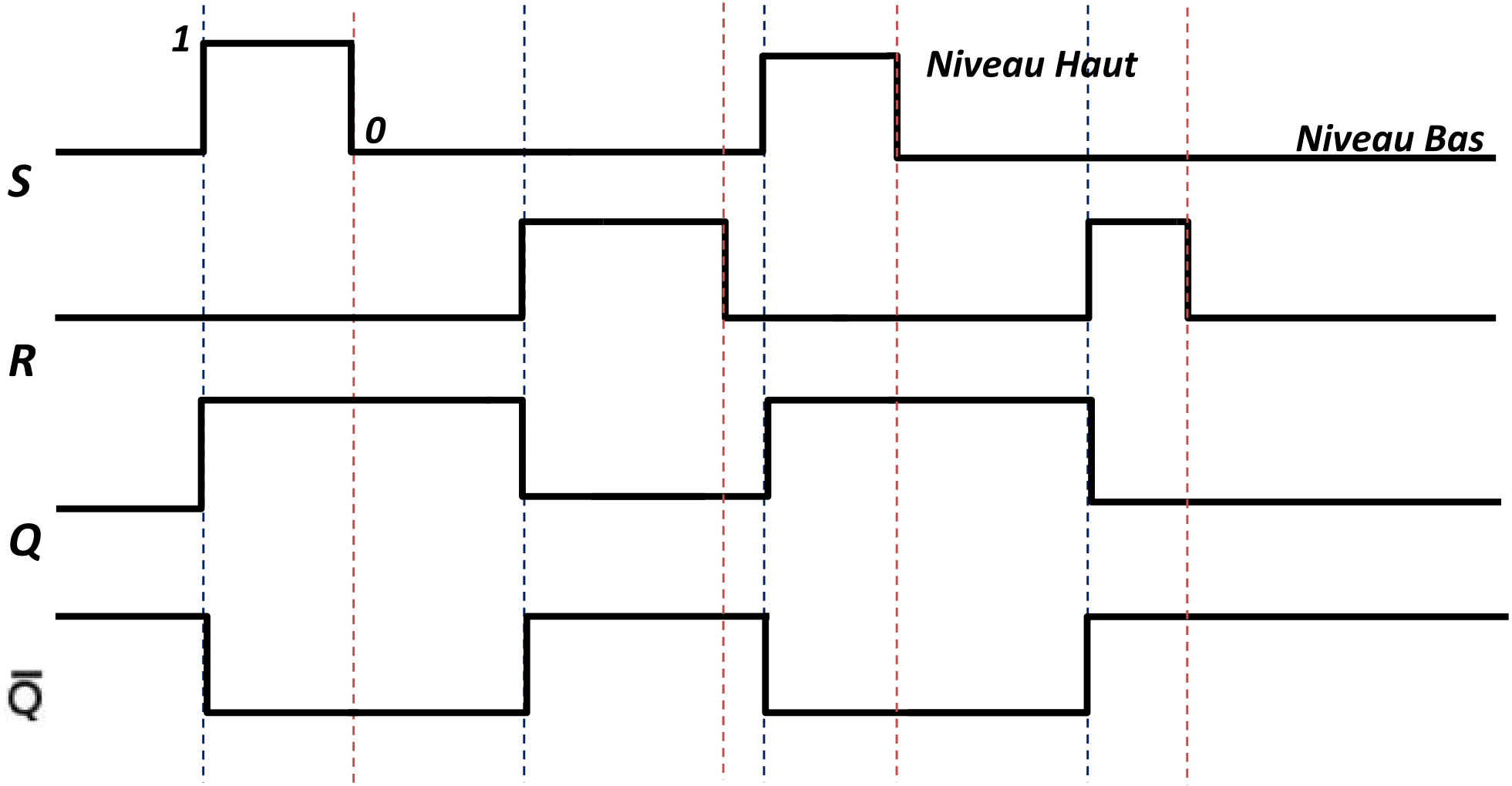
Qu'il soit à base de portes NOR ou de portes NAND, le bistable RS asynchrone sera désormais représenté par son schéma bloc et sa table de vérité, comme suit:



R	S	Q_{t+1}
0	0	Q_t
0	1	1
1	0	0
1	1	interdit

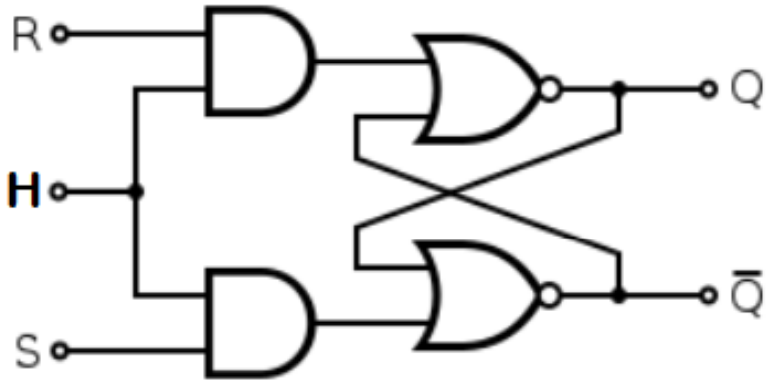
B-1: Le bistable RS asynchrone.

Notion de chronogramme: graphique qui montre l'évolution des signaux, en entée comme en sortie, dans le temps.

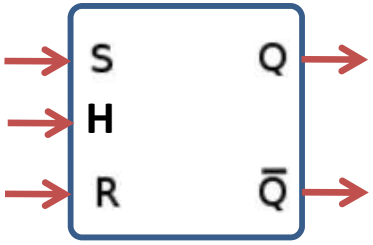


B-2: Le bistable RS synchrone.

Les entrés R et S n'ont un effet sur le bistable qu'à des moments discrets matérialisés par un signal dit d'*Horloge.*



Le schéma bloc et la table de vérité deviennent:



H	*R*	*S*	Q_{t+1}
0	*X*	*X*	Q_t
1	*0*	*0*	Q_t
1	*0*	*1*	*1*
1	*1*	*0*	*0*
1	*1*	*1*	*interdit*

B-3: *L'horloge (Clock)*.

Le fait que les sorties changent aussitôt que les entrées changent, n'est pas toujours souhaitable.
Nous avons, souvent, besoin d'organiser et de n'autoriser ces changement qu'à des moments précis dans le temps. Ces moments précis sont concrétisés par un signal appelé signal ***Horloge*** ou ***Clock***.

L'horloge ***bat*** le temps à un rythme déterminé et marque des instants significatifs où il peut se produire un éventuel changement.

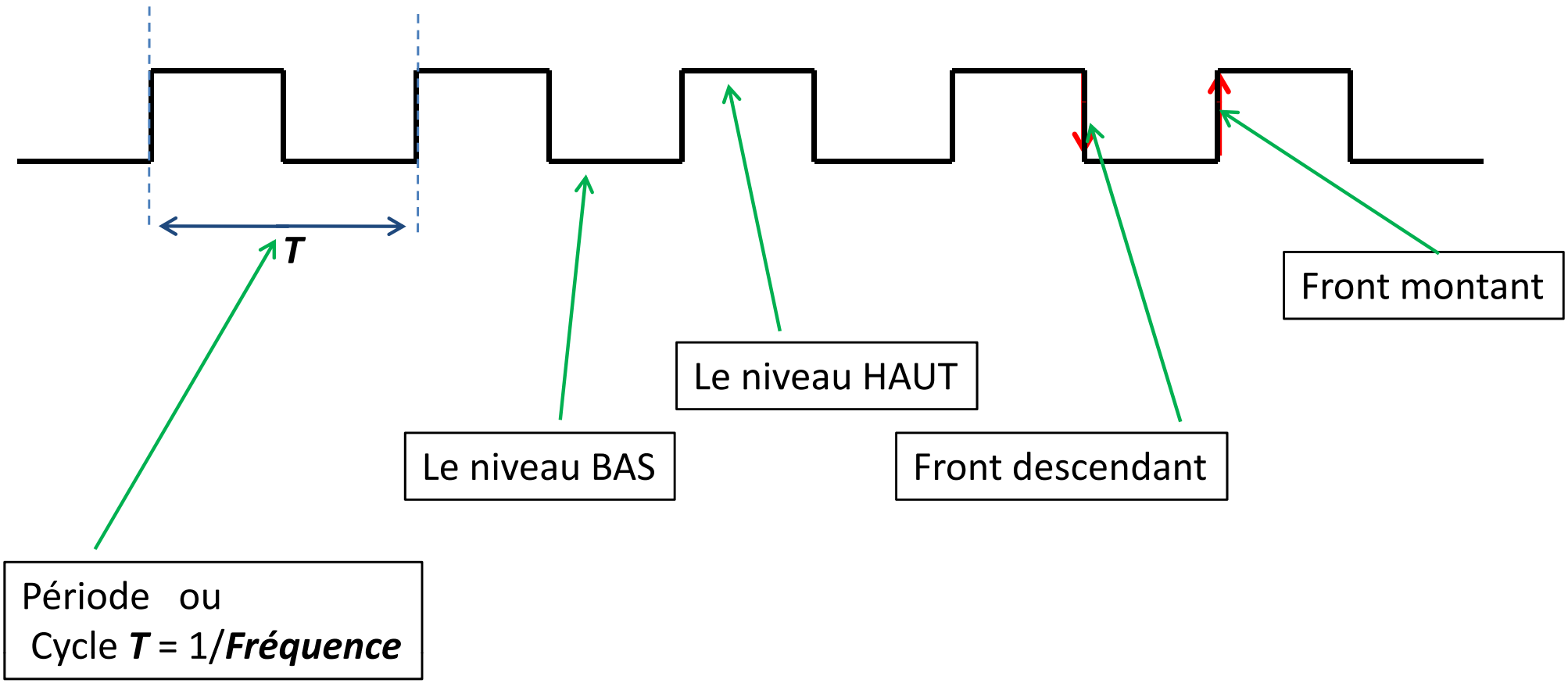
Les circuits qui sont pilotés par, non seulement des entrées de données mais aussi par un signal d'horloge sont dits ***synchrones***.

Ces circuits savent ***quoi*** et ***quand*** faire quelque chose.

Le synchronisme obtenu grâce à l'horloge permet de prétendre à des systèmes dits ***déterministes***, c'est-à-dire que si on soumet la même séquence des entrées deux fois, on obtient en sortie deux fois la ***même chose***. Ce n'est pas le cas des systèmes asynchrones.

B-3: *L'horloge (Clock)*.

L'horloge est souvent représentée par un train d'impulsion, comme l'illustre la figure suivante, qui définit aussi quelques termes clés associés au signal d'horloge:



B-4: *Sensibilité sur niveau, Sensibilité sur front de l'horloge.*

Un bistable synchrone, donc piloté par une horloge, peut changer son état interne soit sur ***niveau*** (haut ou bas) de l'horloge, soit sur un ***front*** (montant ou descendant) du même signal.

La sensibilité sur niveau n'est pas très intéressante car elle ne met pas les circuits à l'abri de l'indéterminisme. Tant que le niveau est maintenu, les bistables traitent leurs entrées si elles changent.

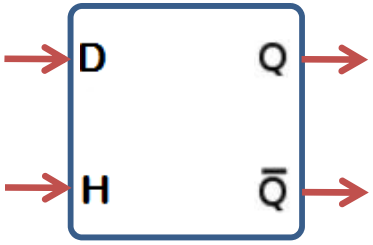
La sensibilité sur front (qui ne dure que pendant la montée ou la descente du signal d'horloge) rend le changement prompt.

Tous les bistables considérés par la suite sont du type sensibles sur front: (***Edge Triggered***).

B-5: Le bistable D *(Par essence synchrone).*

Fonction: le bistable D copie l'entrée D vers Q.
Soit sur niveau (haut ou bas) de l'horloge: on dit qu'il est de type "latch".
Soit sur front (montant ou descendant): on dit de lui qu'il est du type flip-flop ou Edge-triggered.

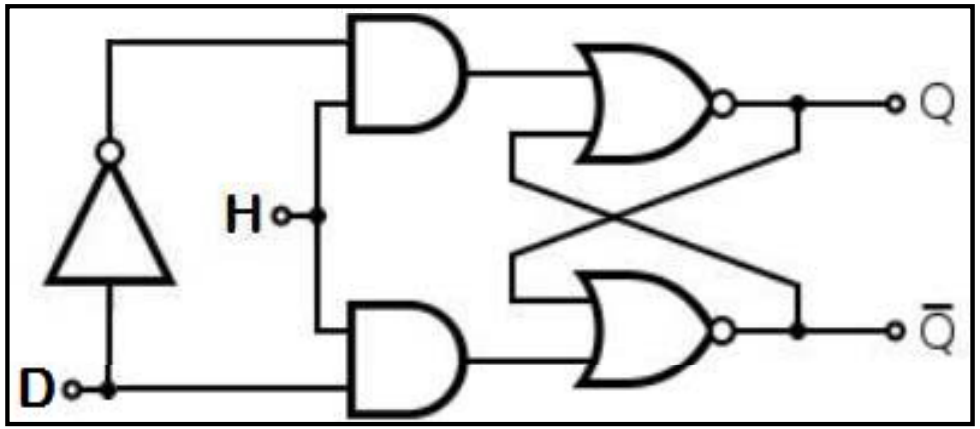
Il sera désormais représenté par son schéma bloc et sa table de vérité, comme suit:

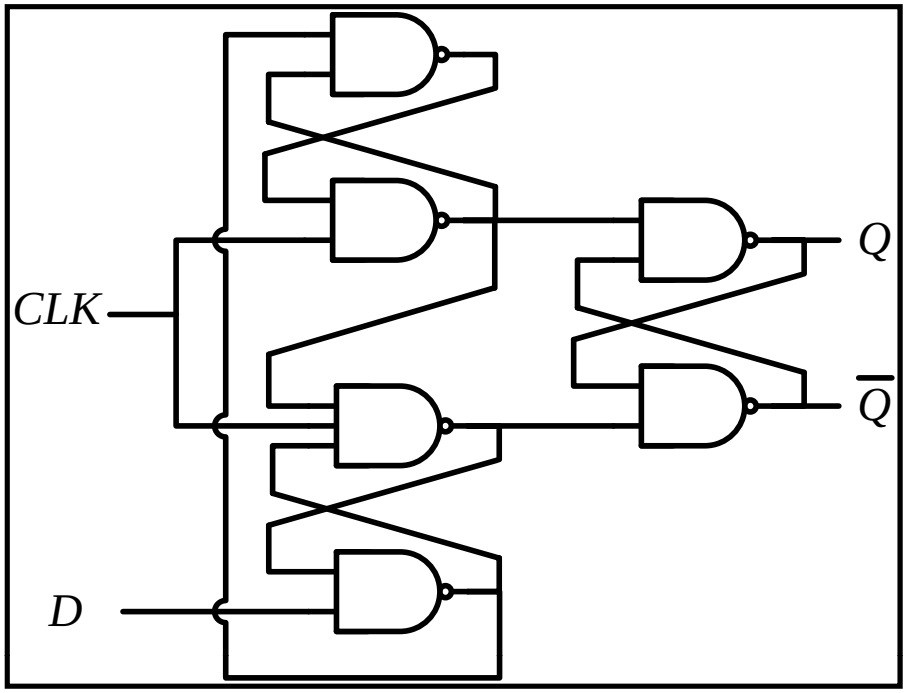


D	Q_{t+1}
0	0
1	1

B-5: Le bistable D *(Schéma interne).*

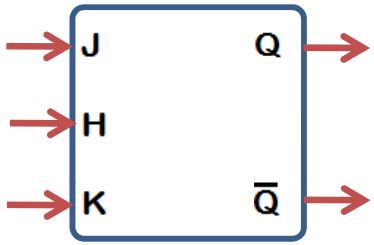
Lorsqu'il est de type latch, il peut être construit à partir d'un bistable RS synchrone comme illustré en bas à gauche.
Lorsqu'il est de type sensible sur front, sa constitution est un peu plus complexe comme le montre le schéma en bas à droite





B-6: Le bistable *JK*.

Son schéma bloc et sa table de vérité sont donnés ci-après:



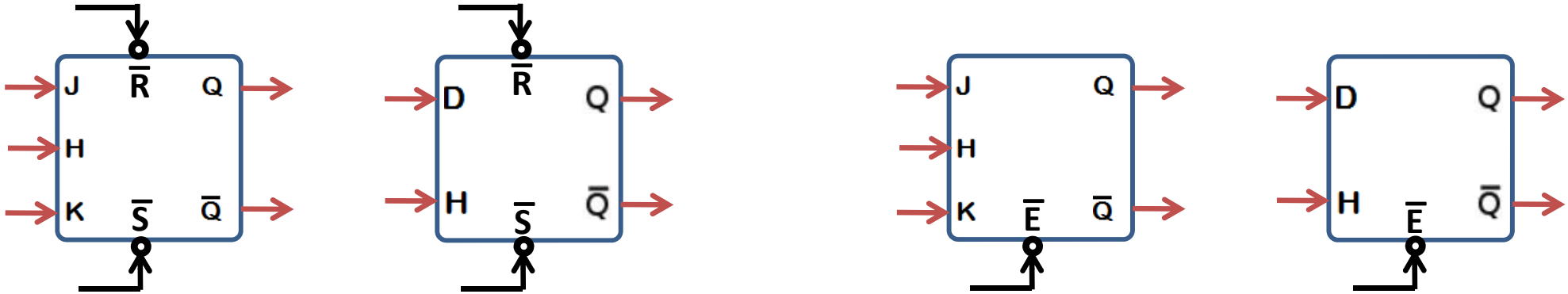
J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\bar{Q}_t$

Le bistable ***JK*** est toujours synchrone.
Le bistable ***JK*** possède une autre table appelée ***table de transition.*** Elle est donnée ci-contre:

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

B-7: Les entrées asynchrones & "*Enable*".

En plus des entrées *synchrones* , certains bistables possèdent des entrées asynchrones de mise à 1 et de mise à 0. Ces entrées sont nommées de diverses manières (***R*** et ***S*** pour ***R****eset* et ***S****et*; ***PR*** et ***CLR*** pour ***PR****eset* et ***CL****e****a****R* …). Par ailleurs elles peuvent être actives au niveau bas ou au niveau haut (souvent bas, auquel cas leurs entrées sont annotées avec un petit rond). On s'interdit de les activer simultanément. Elles ont la priorité. L'entrée ***Enable***, lorsqu'elle est présente , permet au circuit globalement d'être actif ou inactif.



B-8: Application des bistables.

Les applications des bistables sont nombreuses. Nous allons examiner deux grandes familles: les registres et les générateurs de séquences et particulièrement les compteurs/décompteurs.

Un ***registre*** est un composant de mémorisation de ***n bits***. Chaque bit est supporté, ici, par la sortie d'un bistable.

Les opérations qu'on peut faire sur un registre:

- fixer son contenu, sa valeur: chargement (Load) ou écriture (Write).
- l'écriture dans le registre et la lecture du registre peuvent se faire en parallèle ou en série.
- le mettre à zéro.
- faire subir à son contenu une opération de décalage (à droite, à gauche, circulaire)

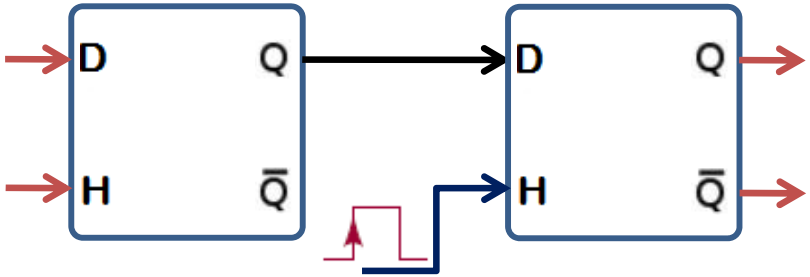
Toutes ces opérations peuvent se faire soit de façon synchrone ou asynchrone. (Le plus souvent synchrone).

B-8: Application des bistables.

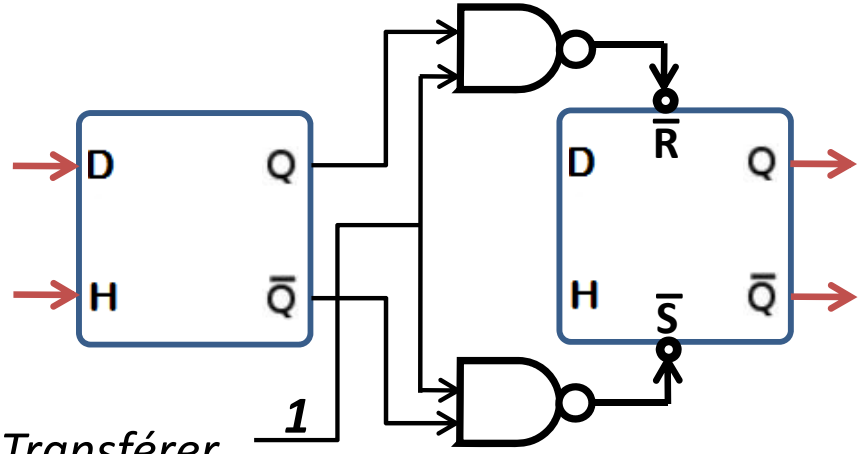
B-8-1: *chargement, transfert ou écriture dans un registre de 1 bit.*

- *Opération synchrone à gauche.*
- *Opération asynchrone à droite.*

✓ l'opération sur n bits est une duplication parallèle de n fois ces schémas.



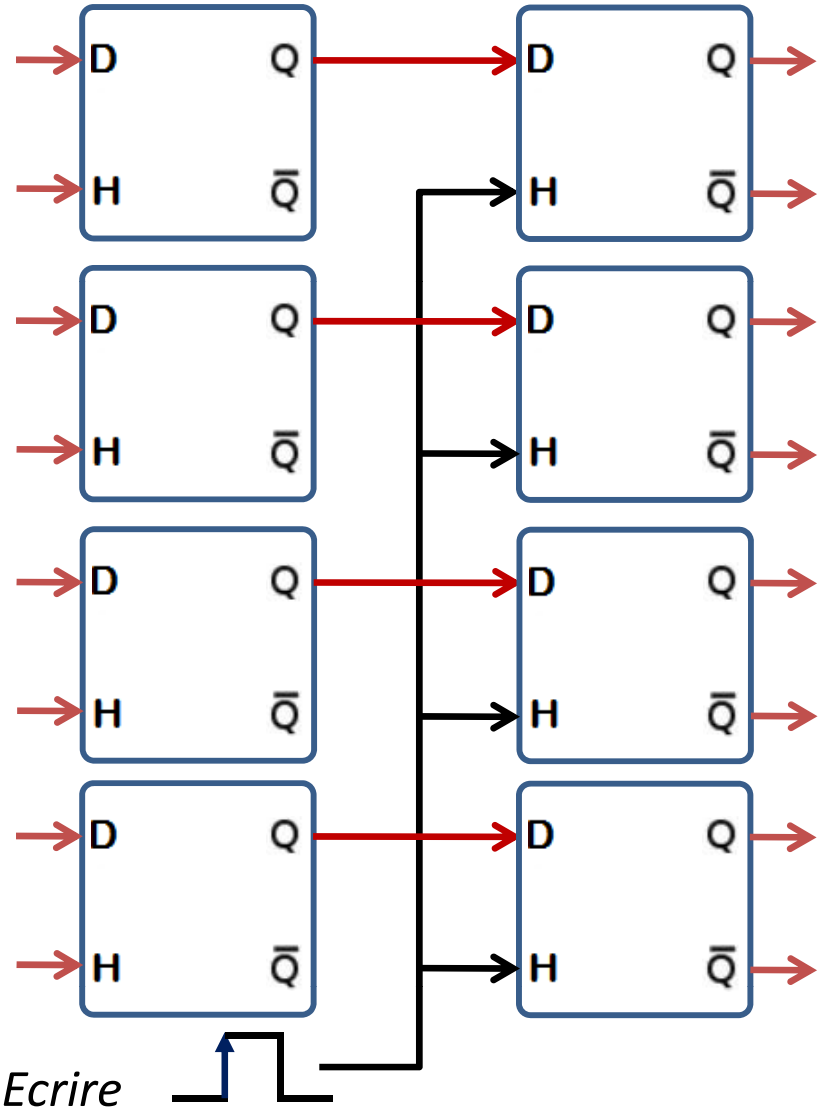
- *Transférer*
- *Charger*
- *Ecrire*



- *Transférer*
- *Charger*
- *Ecrire*

B-8: Application des bistables.

B-8-2: *chargement, transfert ou écriture parallèle.*



- *Transférer, Charger, Ecrire*

B-8: Application des bistables.

B-8-3: *Registre 4 bits, chargement parallèle synchrone, mise à zéro asynchrone.*

L'entrée H permet le chargement des $\boldsymbol{D_i}$ *.*

L'entrée Master Reset permet de mettre le registre à 0 de façon asynchrone.

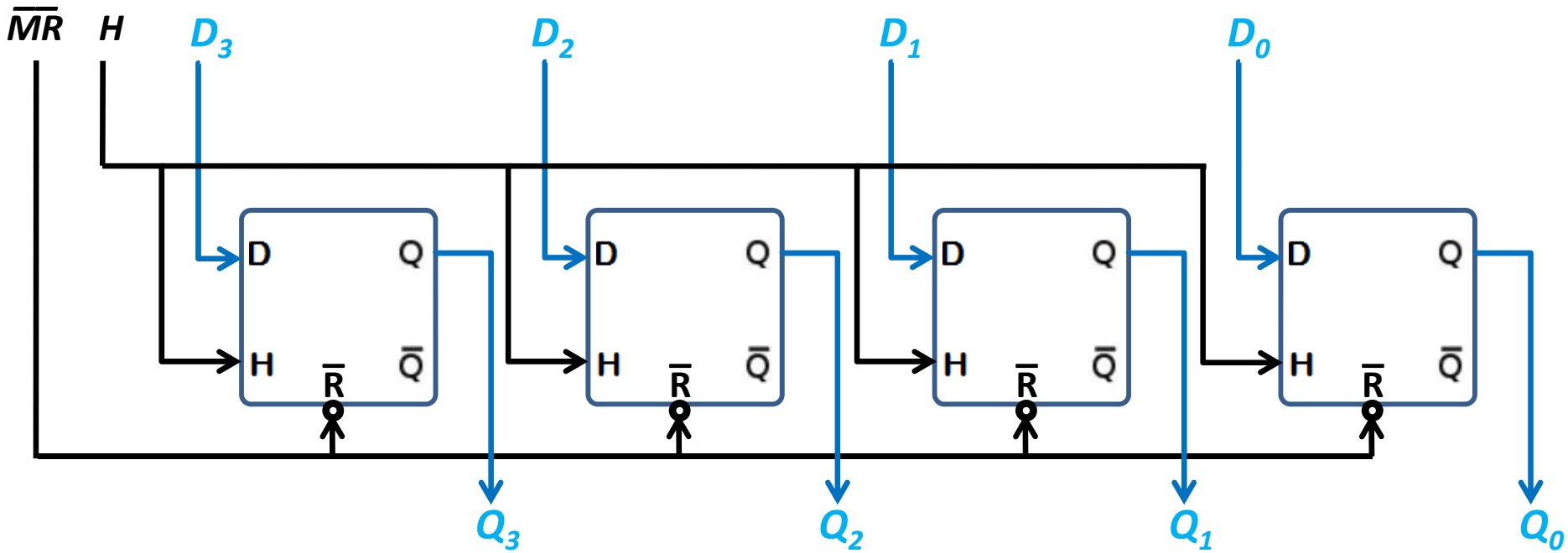
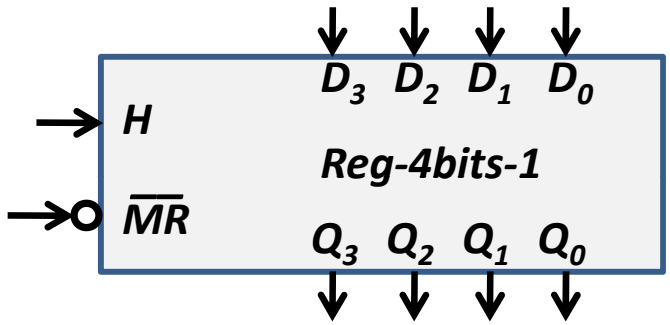
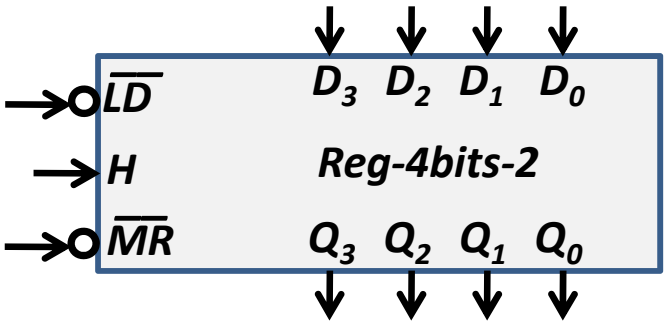


Schéma bloc



B-8: Application des bistables.

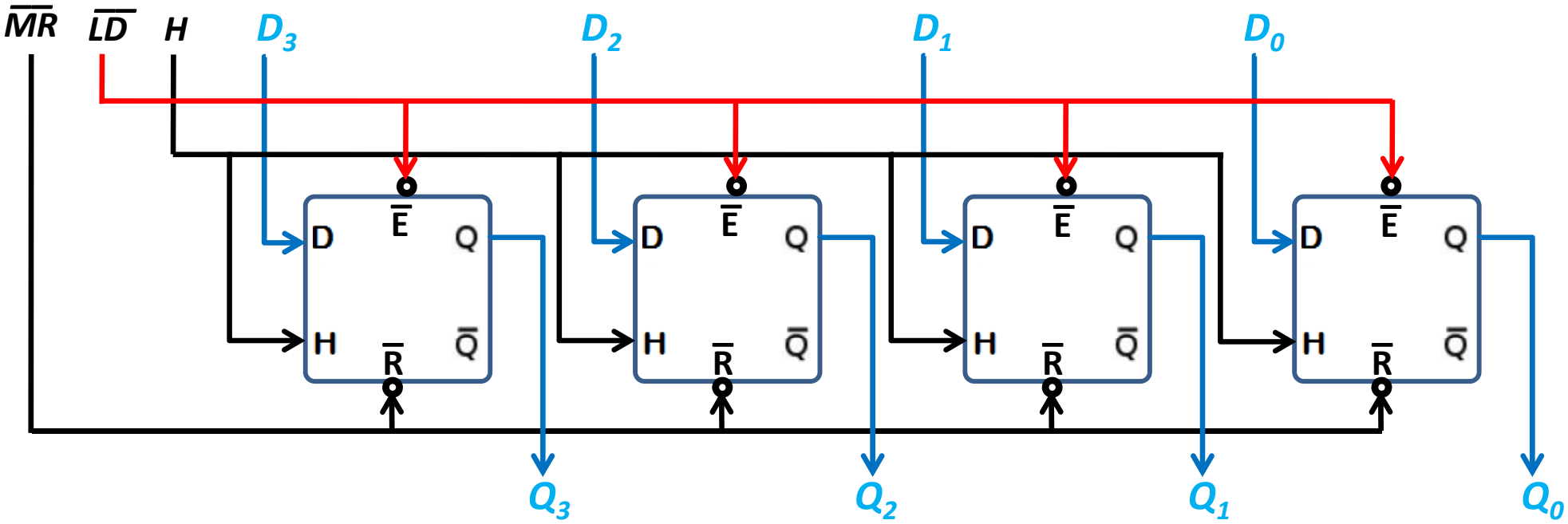
Schéma bloc



B-8-4: *Registre 4 bits, avec les opérations suivantes:*

- ✓ *Chargement parallèle synchrone*
  permis par un signal nommé LD (pour Load),
  actif au niveau bas.
- ✓ *Mise à zéro asynchrone, active au niveau bas.*

Le bistable D possède l'entrée Enable,
active au niveau bas.



B-8: Application des bistables.

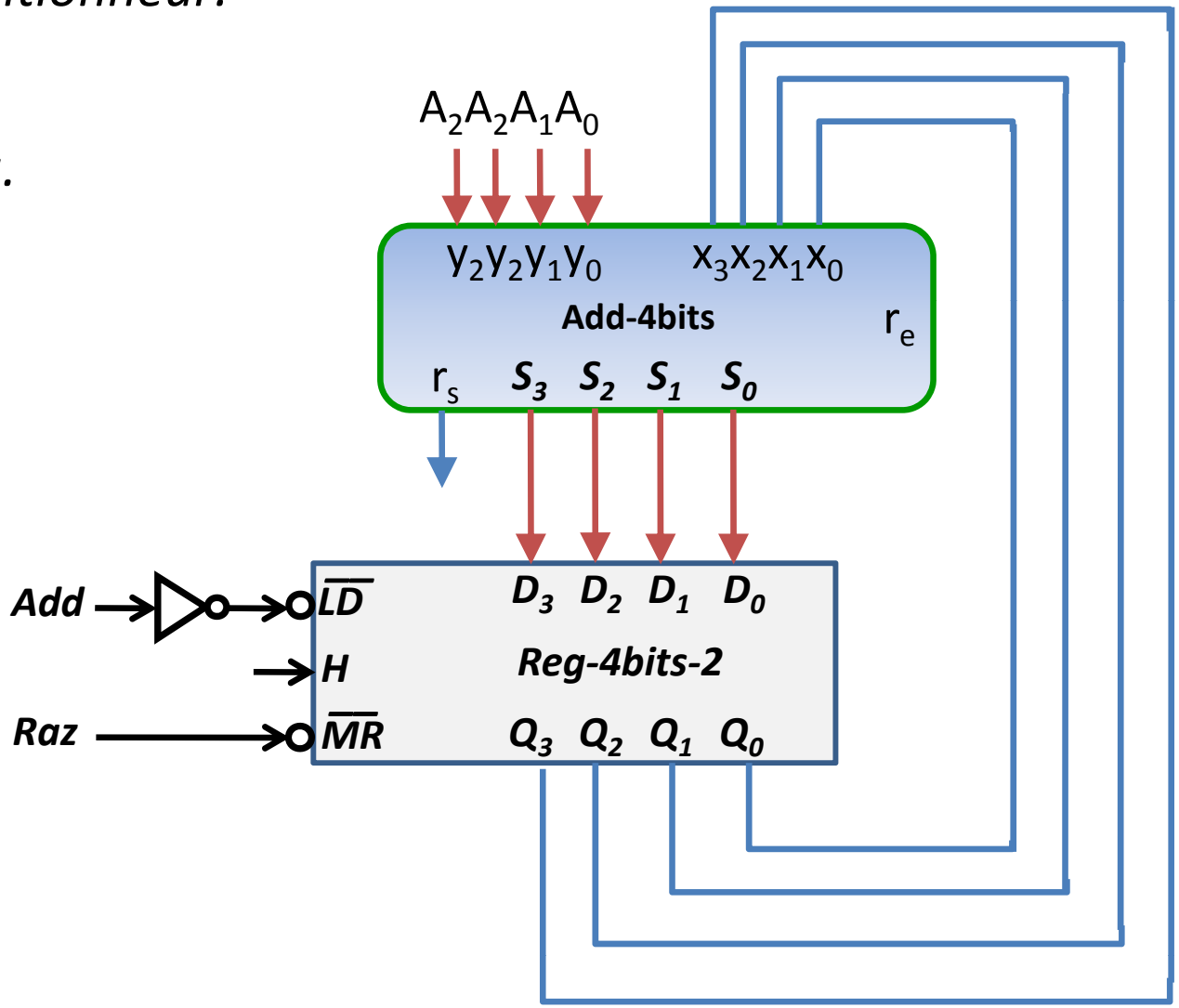
B-8-5: *Accumulateur additionneur:*

Opérations:

***Raz**: Mise à zéro de l'Accu.*
***Add**: Accu := Accu + A.*

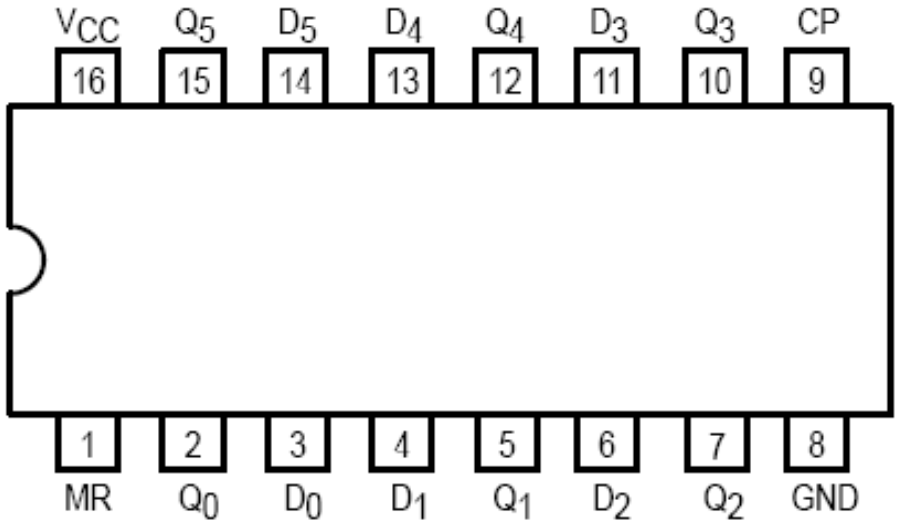
Séquence d'opérations
pour calculer:
Z:=B+C+D+E

1) Mise à zéro de l'Accu;
2) Accu := B; /*B+0
3) Accu := Accu + C;
4) Accu := Accu + D;
5) Accu := Accu + E;

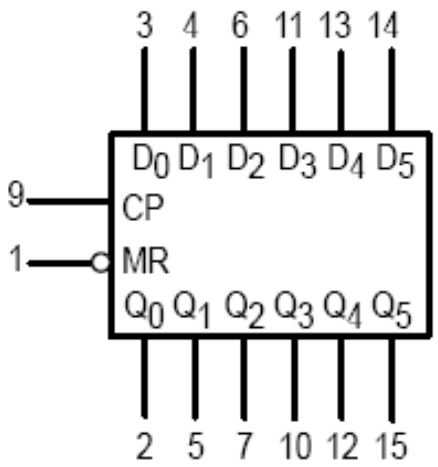


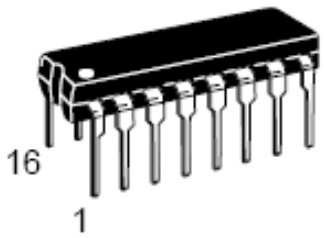
B-8: Application des bistables.

B-8-6: *le registre 6 Bits 74174.*



LOGIC SYMBOL





PIN NAMES

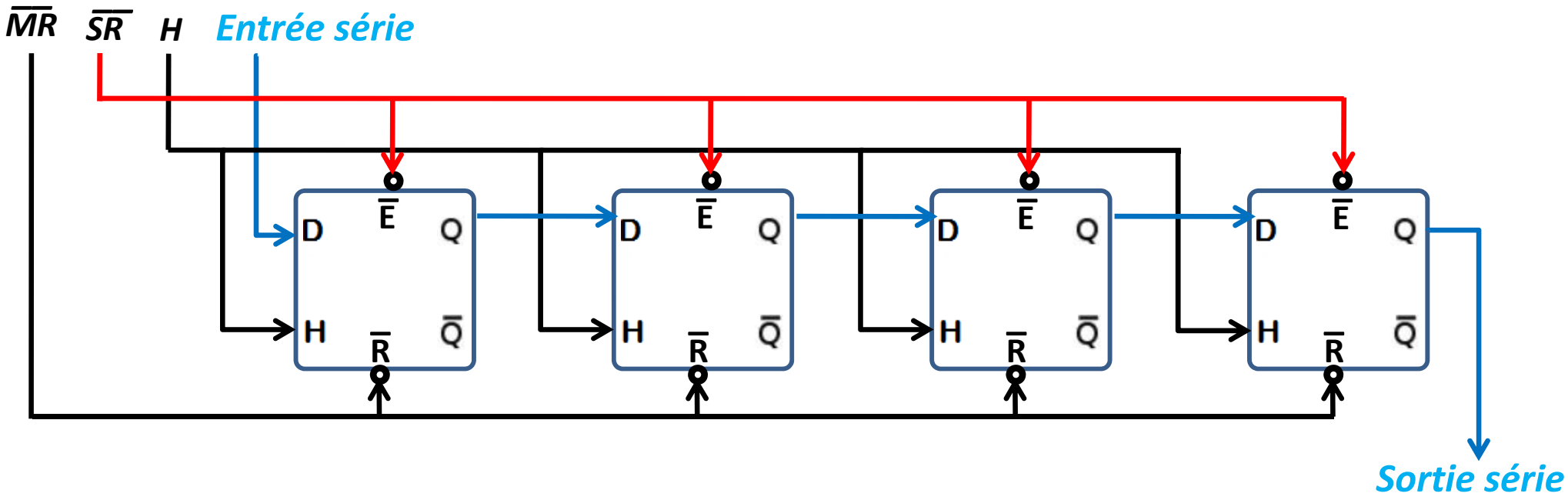
$D_0 - D_5$	Data Inputs
CP	Clock (Active HIGH Going Edge) Input
MR	Master Reset (Active LOW) Input
$Q_0 - Q_5$	Outputs (Note b)

B-8: Application des bistables.

B-8-7: *Les registres à décalage (Shift registers).*

Les bistables de type D suivants sont montés de façon à réaliser un registre avec les fonctionnalités suivantes:

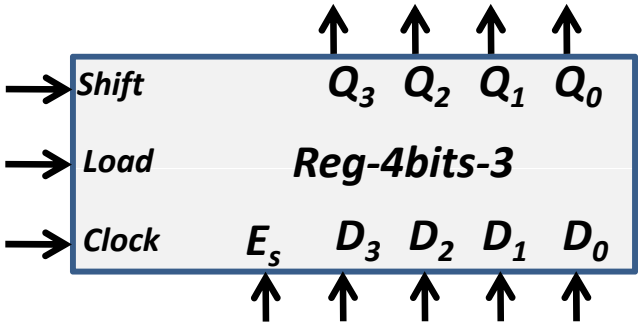
- ✓ *Décalage à droite permis par l'entrée (**SR**: **S**hift **R**ight), active au niveau bas.*
- ✓ *Mise à zéro asynchrone par le signal (**MR**: **M**aster **R**eset) actif au niveau bas.*

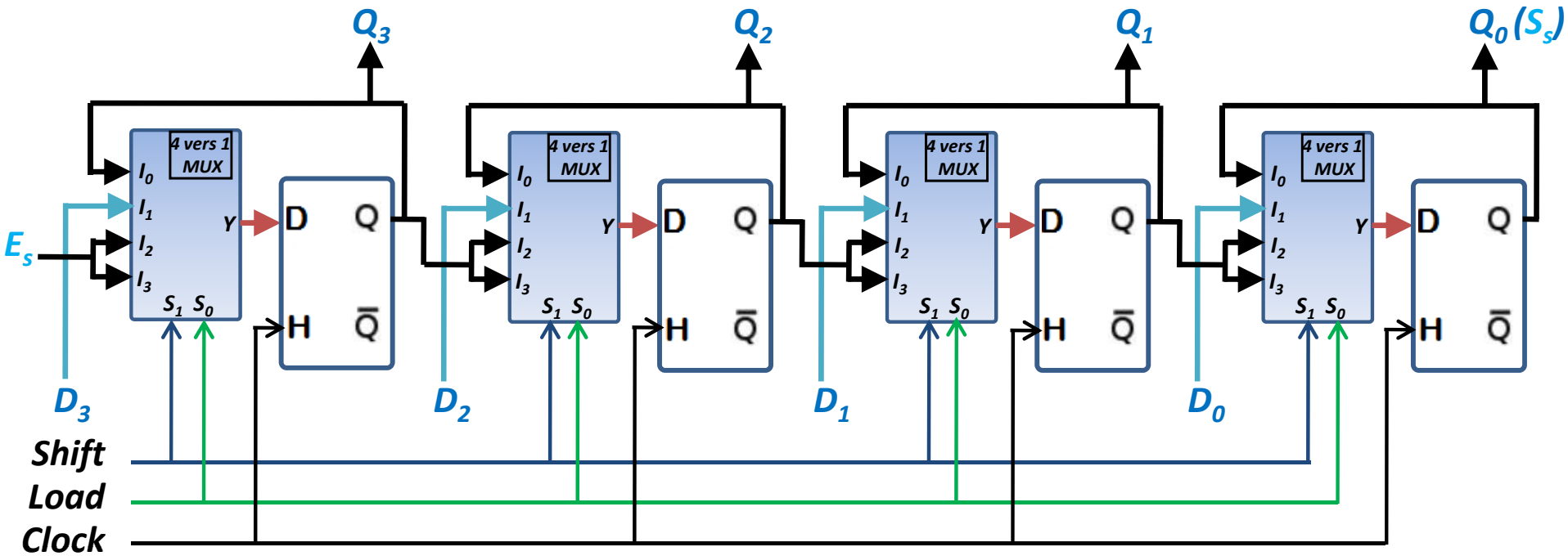


B-8: Application des bistables.

B-8-8: *Les registres à chargement parallèle et décalage à droite.*

Shift	Load	opération
0	0	mémorisation
0	1	Chargement //
1	X	Décalage à droite



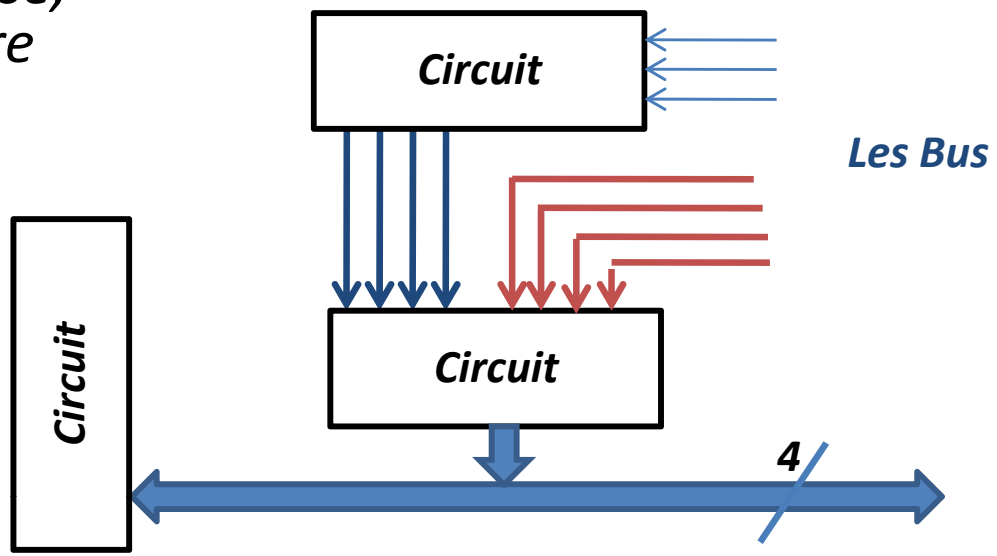


B-8: Application des bistables.

B-8-9: *Notion de bus.*

Les entrés des registres des schémas précédents viennent des sorties d'autres circuits. Elle sont véhiculées sur des fils parallèles appelés BUS.
Les sorties des registres alimentent d'autres bus .
*Les schémas précédents comportaient deux type de bus: les données (**D_i** **Q_i** , **X_i**, …) et les contrôles (MR, load, shift, Add, Raz, etc)*

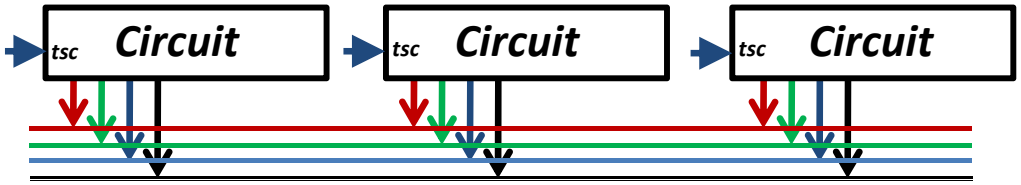
Les bus peuvent être monodirectionnels ou bidirectionnels et ont une certaine largeur (nombre de fils //). Ils sont souvent représentes dans les schémas par une flèche grasse, barré d'un trait et un nombre pour indiquer la largeur du bus (nombre de fils).

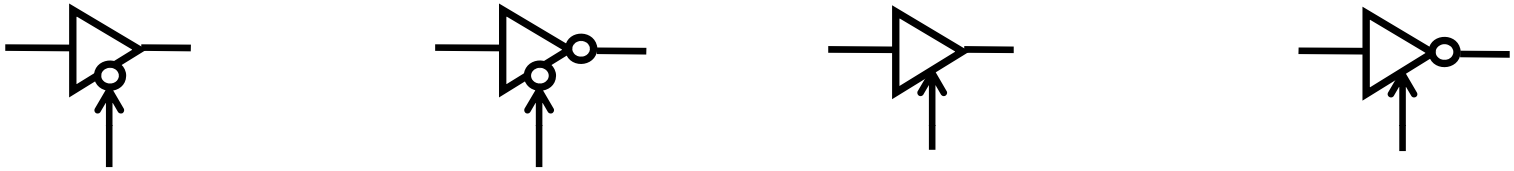


B-8: Application des bistables.

B-8-10: *Notion d'état haute impédance. (HiZ)*

On ne peut connecter une sortie d'un circuit qu'à un nombre très limité d'entrées. Une partie, électrique, appelée ***buffer*** *peut être ajoutée à une sortie pour augmenter sa capacité à "**attaquer**" plusieurs entrées. Il s'agit d'amplifier le signal.*
Ce buffer, dans le cas simple, ne modifie pas l'état 1 ou 0 de la sortie. Il peut au plus l'inverser.
Il existe une variété de buffers, ***dits à trois états*** *(Three states (ts) buffers ou tri-state buffers) qui peuvent soit, juste propager l'état d'une sortie, soit* ***isoler*** *cette sortie du bus auquel elle est connectée. C'est ce dernier cas qui est assimilé à un troisième état, appelé également l'état haute impédance (HiZ).*
Les sorties qui proviennent des circuits qui possèdent ce troisième état peuvent être connectées ensembles et éviter donc l'utilisation de multiplexeurs.
Seule une sortie est active, grâce au signal TSC , les autres sont isolées du bus.
*tsc :**t**ris-**s**tate **c**ontrol.*

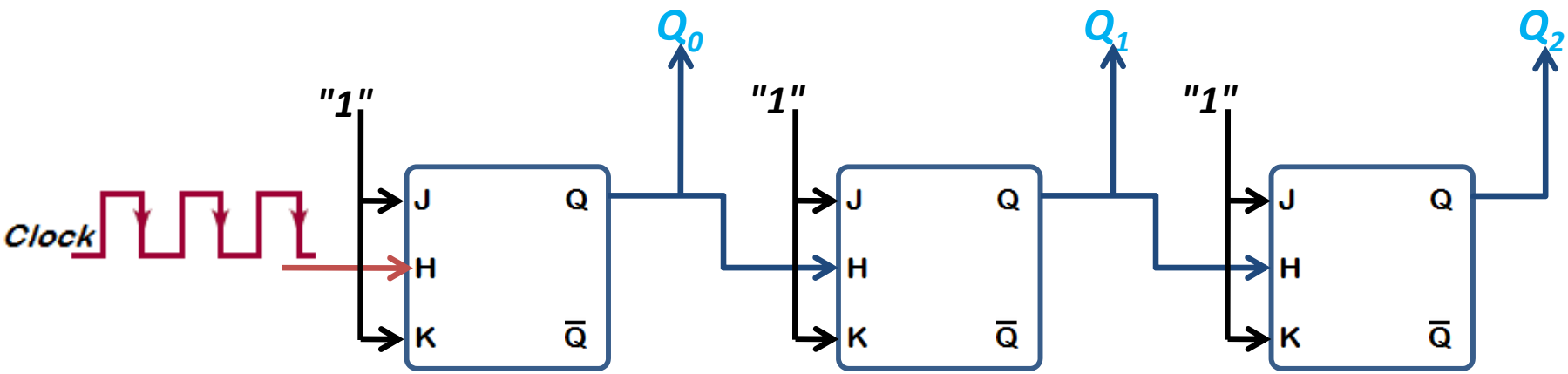


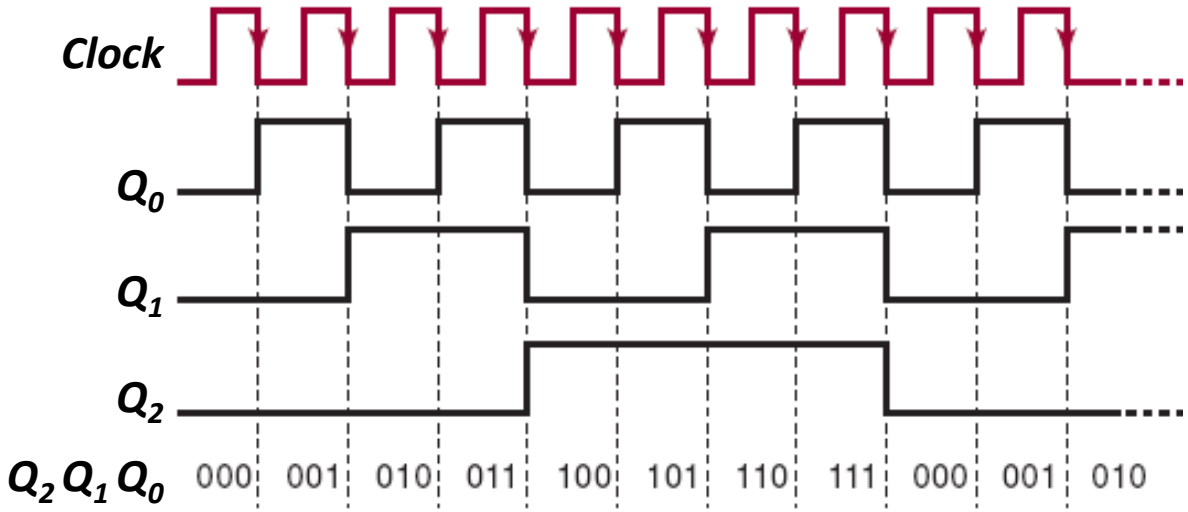


Symbolique: *Ts Buffer normal ,Ts Buffer inverseur, sortie enabled au niveau Bas/haut*

B-8: Application des bistables.

B-8-11: *Compteur asynchrone modulo 8.(Le bistable JK est sensible sur front descendant de H).*

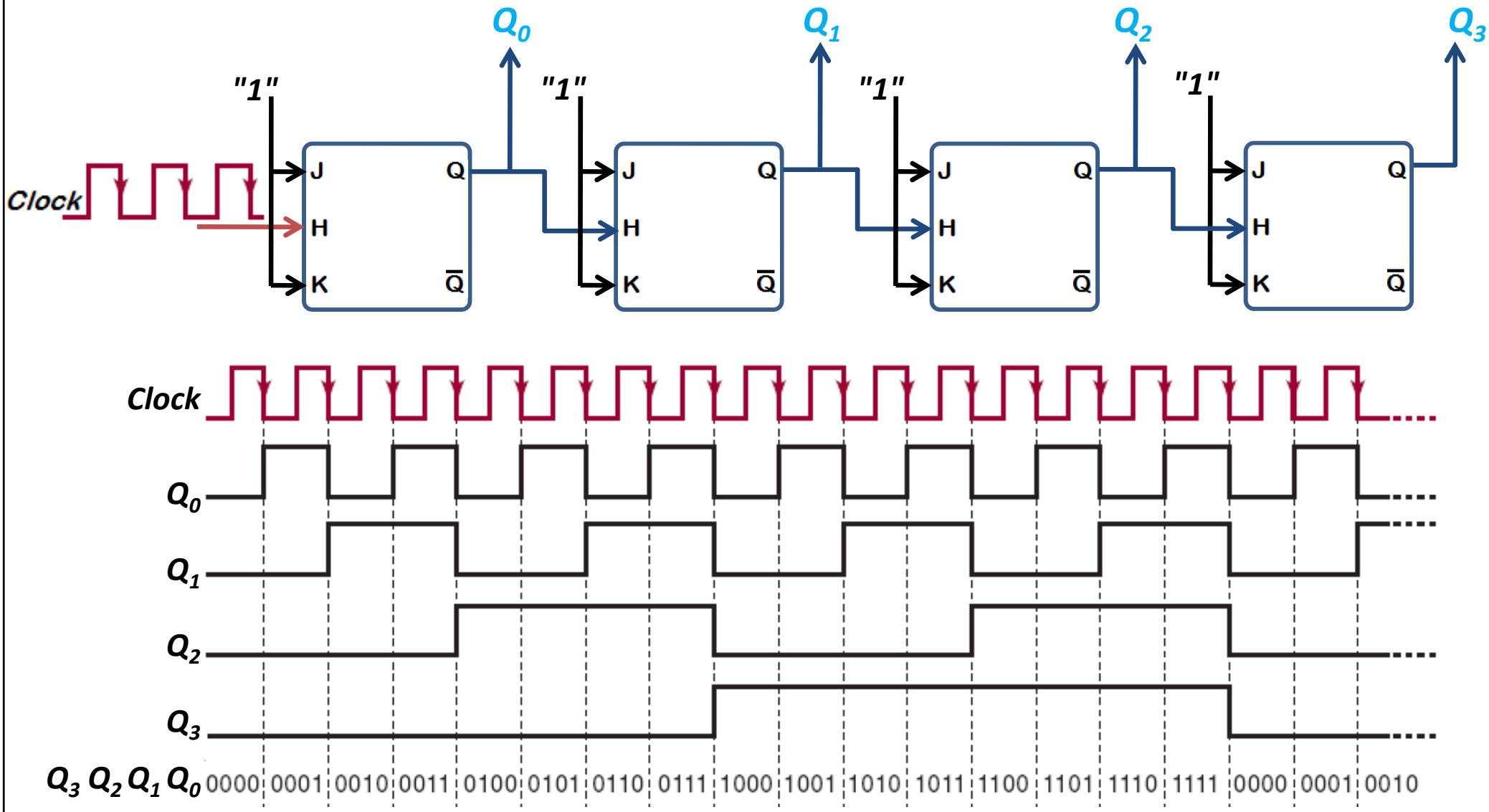




***Question**:*
*Si la fréquence du signal **Clock** est de 1Hz, que compte ce circuit?*

B-8: Application des bistables.

B-8-12: *Compteur asynchrone modulo 16.* *(Le bistable JK est sensible sur front descendant de H).*



B-8: Application des bistables.

B-8-13: *Compteur synchrone modulo 8.* *(En utilisant le bistable JK).*

✓ *Rappel de la table de transition (Dite également table d'excitation du bistable JK)*

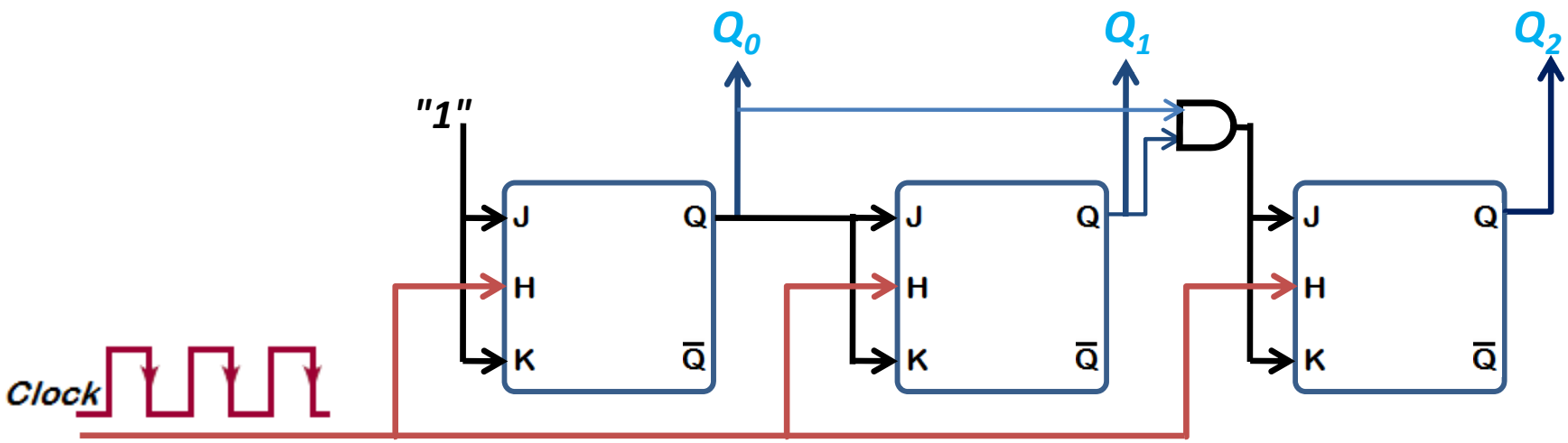
Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

$$J_0 = K_0 = 1;$$
$$J_1 = K_1 = Q_0;$$
$$J_2 = K_2 = Q_0 . Q_1;$$

B-8: Application des bistables.

B-8-13: *Compteur synchrone modulo 8.* *(Le bistable JK est sensible sur front descendant de H).*



B-8: Application des bistables.

B-8-14: *Compteur synchrone modulo 16.*

Table de vérité ci-contre.

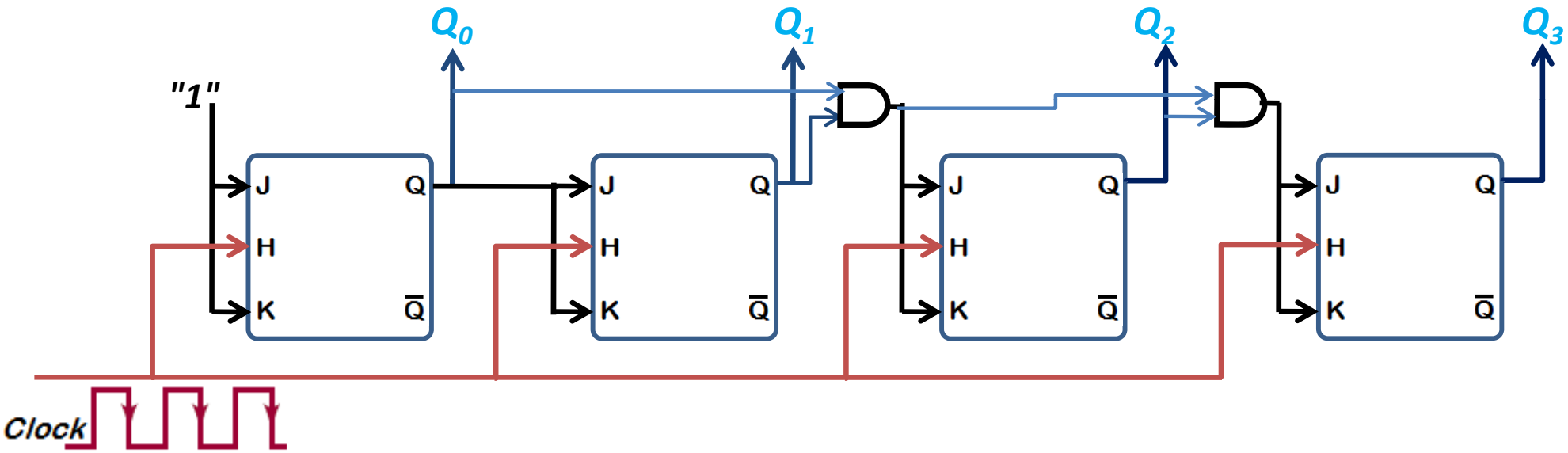
Schéma:

Diapo suivante.

Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

B-8: Application des bistables.

B-8-14: *Compteur synchrone modulo 16.* *(Le bistable JK est sensible sur front descendant de H).*

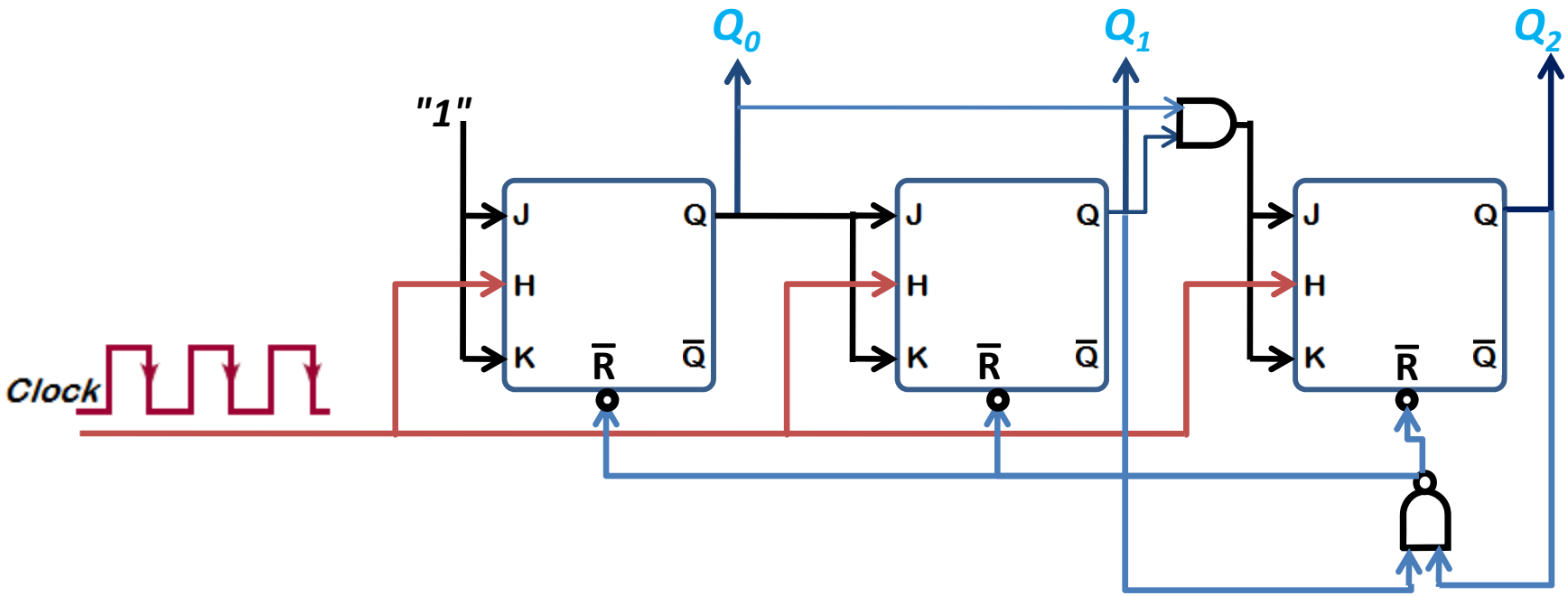


B-8: Application des bistables.

B-8-15: *Compteur synchrone modulo 6.* *(Le bistable JK est sensible sur front descendant de H).*

Idée prendre un compteur modulo 8 et le remettre à zéro dès qu'il atteint l'état 110.

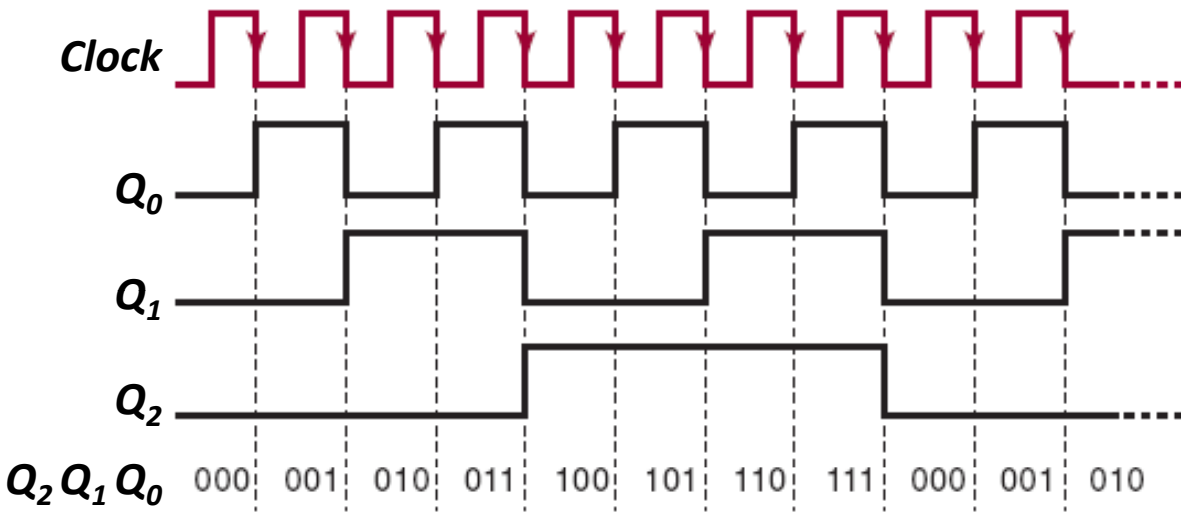
Lorsque $Q_2 Q_1$ *= 11 la porte NAND génère une sortie à 0 permettant de remettre les bistables à l'état 000*



Pour obtenir un compteur modulo 10, le NAND pour la RAZ se fait entre $Q_3 Q_1$

B-8: Application des bistables.

B-8-16: *Décompteur 7 à 0.*

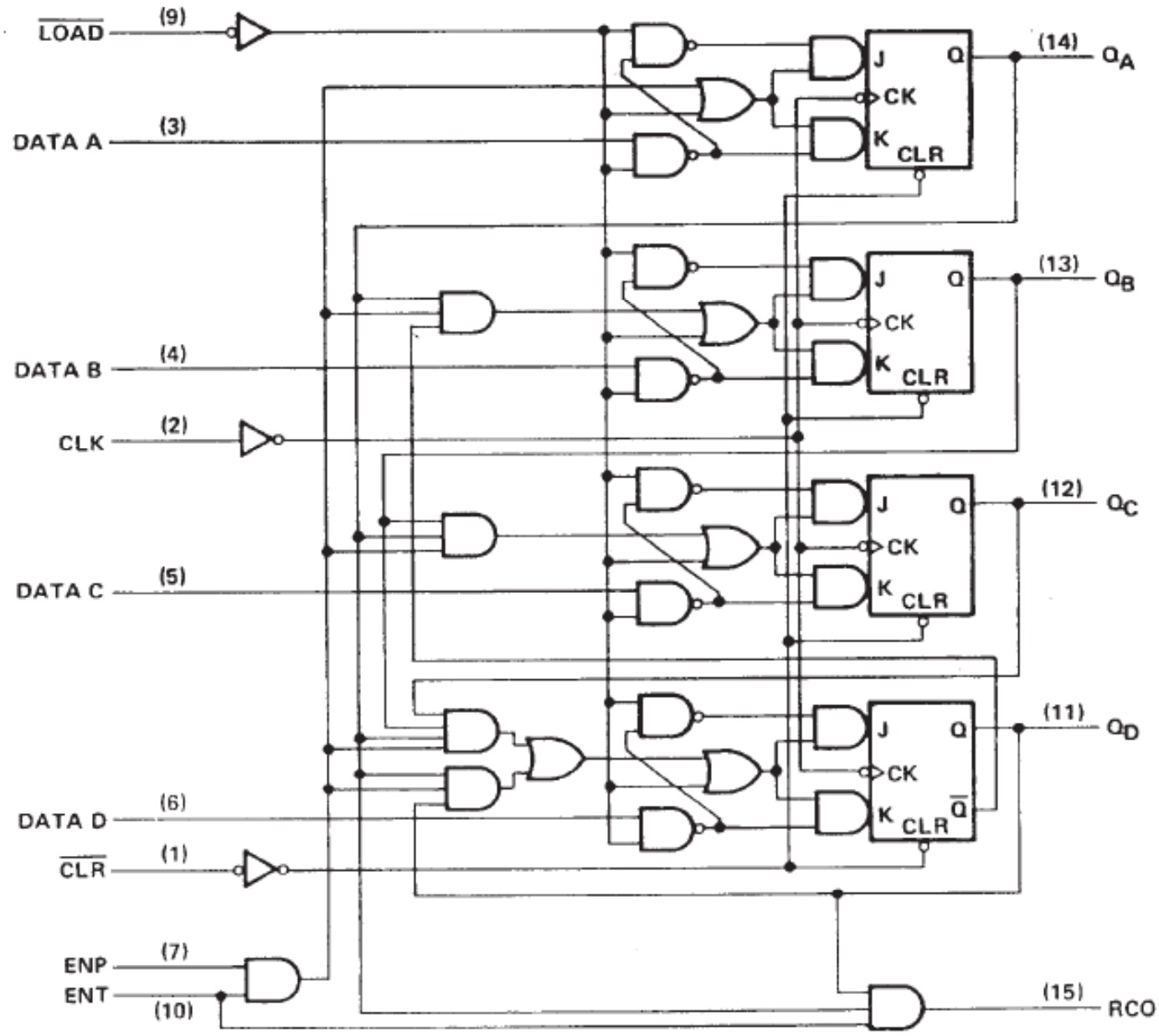


Si on prend les sorties $Q'_2 Q'_1 Q'_0$ *(les sorties Q des bistables complémentées),* *on obtient un décompteur asynchrone.*

La méthode pour développer un compteur synchrone peut être utilisée pour un décompteur synchrone.

B-8: Application des bistables.

B-8-17: Un exemple de compteur en circuit MSI: le compteur décimal 74160.



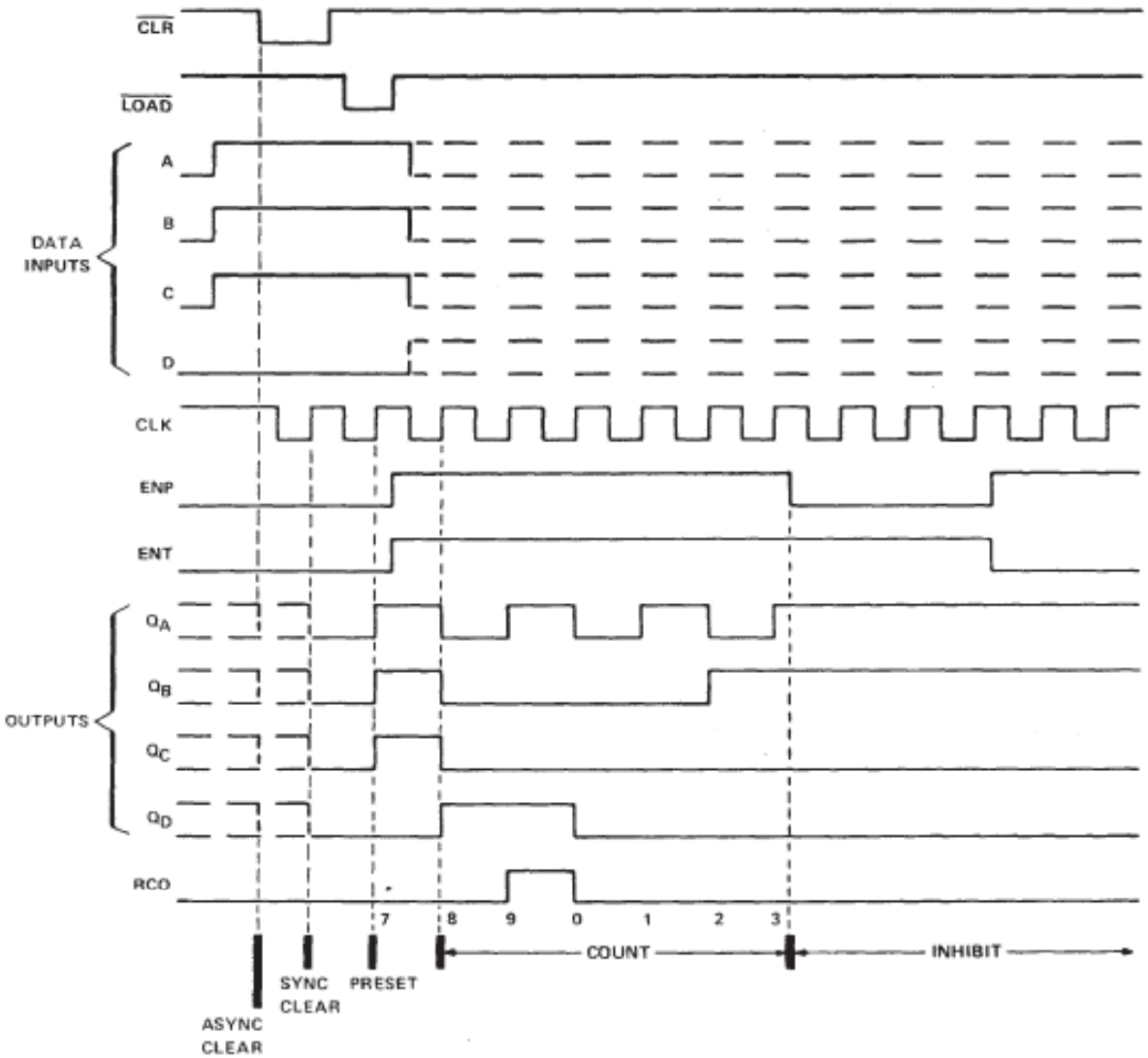
B-8: Application des bistables.

B-8-17: Un exemple de compteur en circuit MSI: le compteur décimal 74160.

typical clear, preset, count, and inhibit sequences

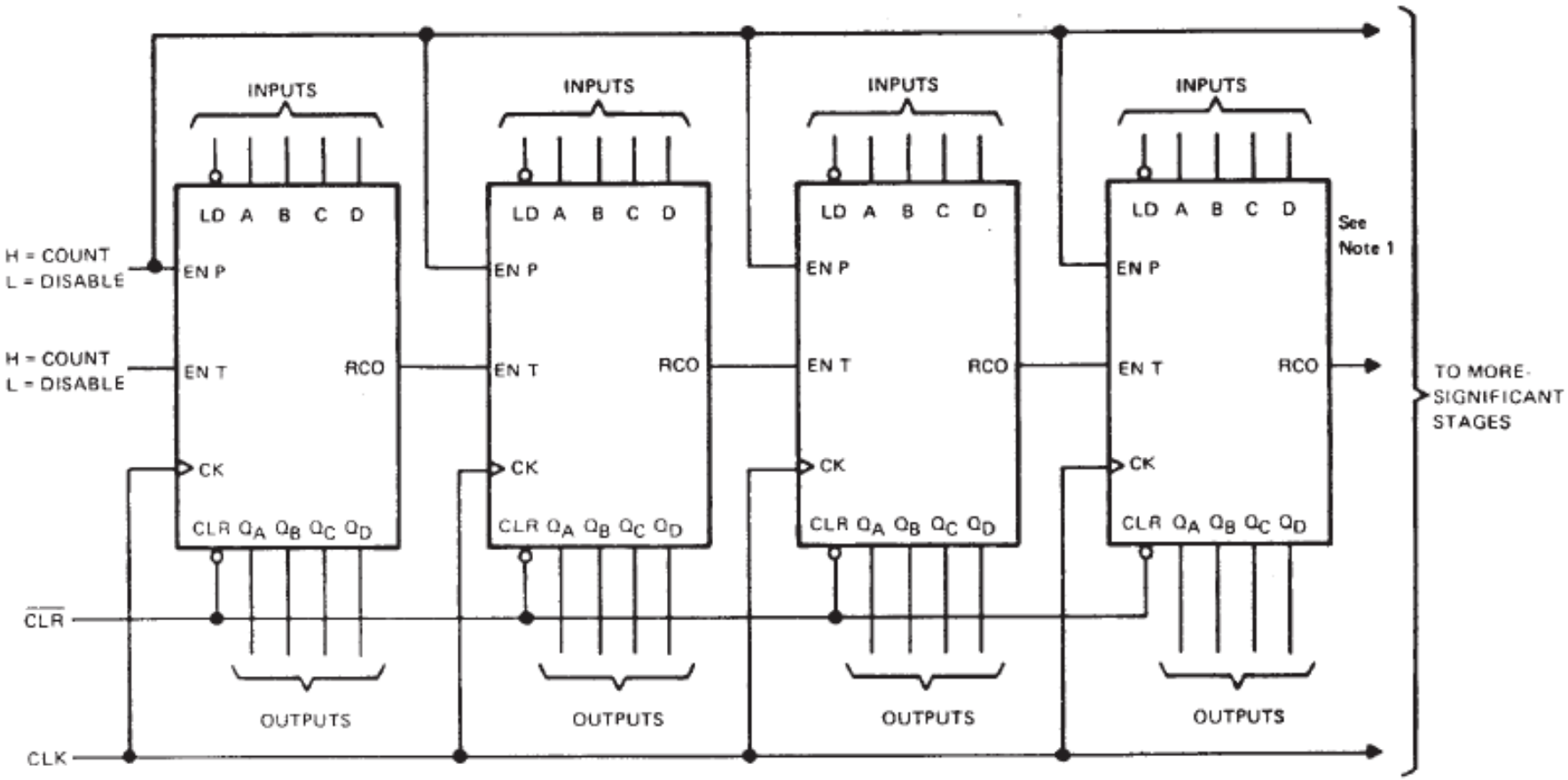
Illustrated below is the following sequence:

1. Clear outputs to zero ('160 and 'LS160A are asynchronous; '162, 'LS162A, and 'S162 are synchronous)
2. Preset to BCD seven
3. Count to eight, nine, zero, one, two, and three
4. Inhibit



B-8: Application des bistables.

B-8-17: Un exemple de compteur en circuit MSI: le compteur décimal 74160.



C) *Les réseaux logiques programmables.*

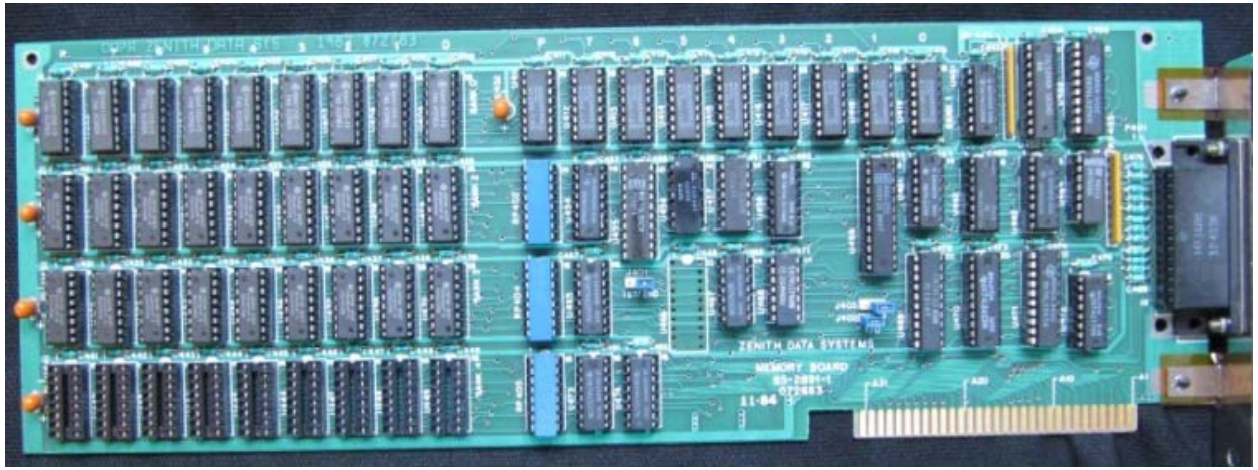
PLDs (***P***rogammable ***L***ogic ***D***evice***s***).

Quel est le problème qu'essaye de résoudre les PLDs?

Eviter de réaliser les fonctions logiques en utilisant les portes logiques de base, parce que ceci donne naissance à des solutions non optimisées sur divers plan:

- Espace
- Souplesse de modification.
- Coût.

L'idéal serait d'avoir ce qui est appelé "***The single chip solution", modifiable à volonté, là où le circuit réside.***



C) *Les réseaux logiques programmables. Les réponses.*

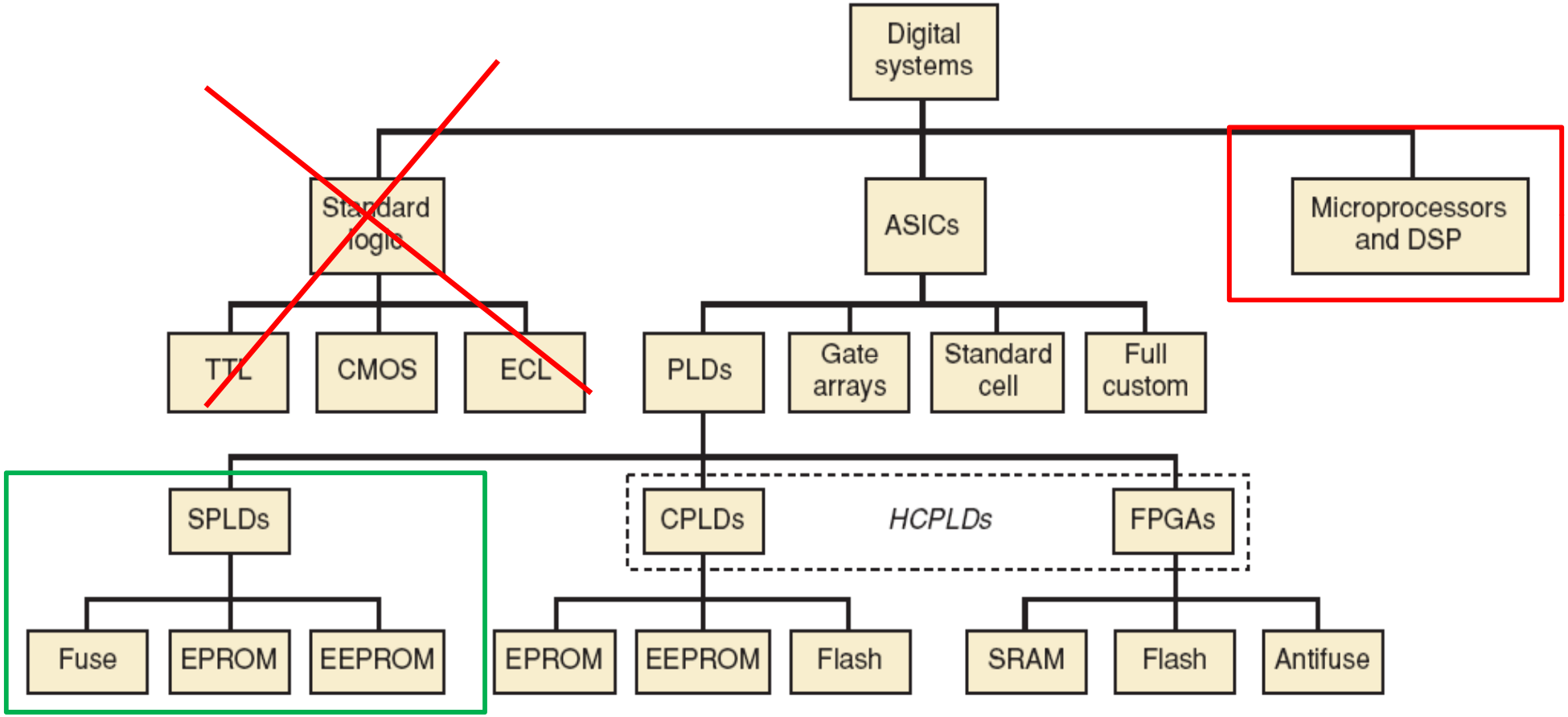


FIGURE 13-1 Digital system family tree.

Source: Ronald J. Tocci's Book

C) *Les réseaux logiques programmables.*

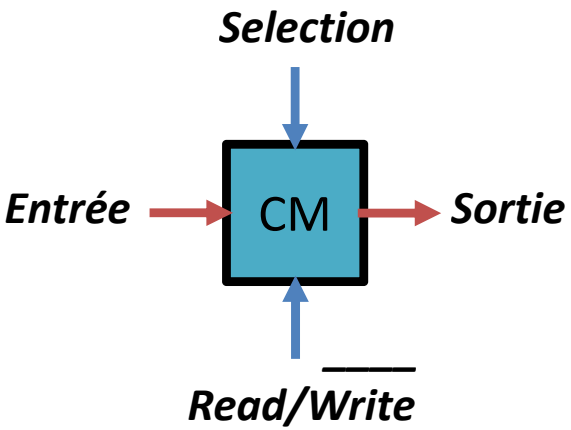
C-1: *Les Mémoires*

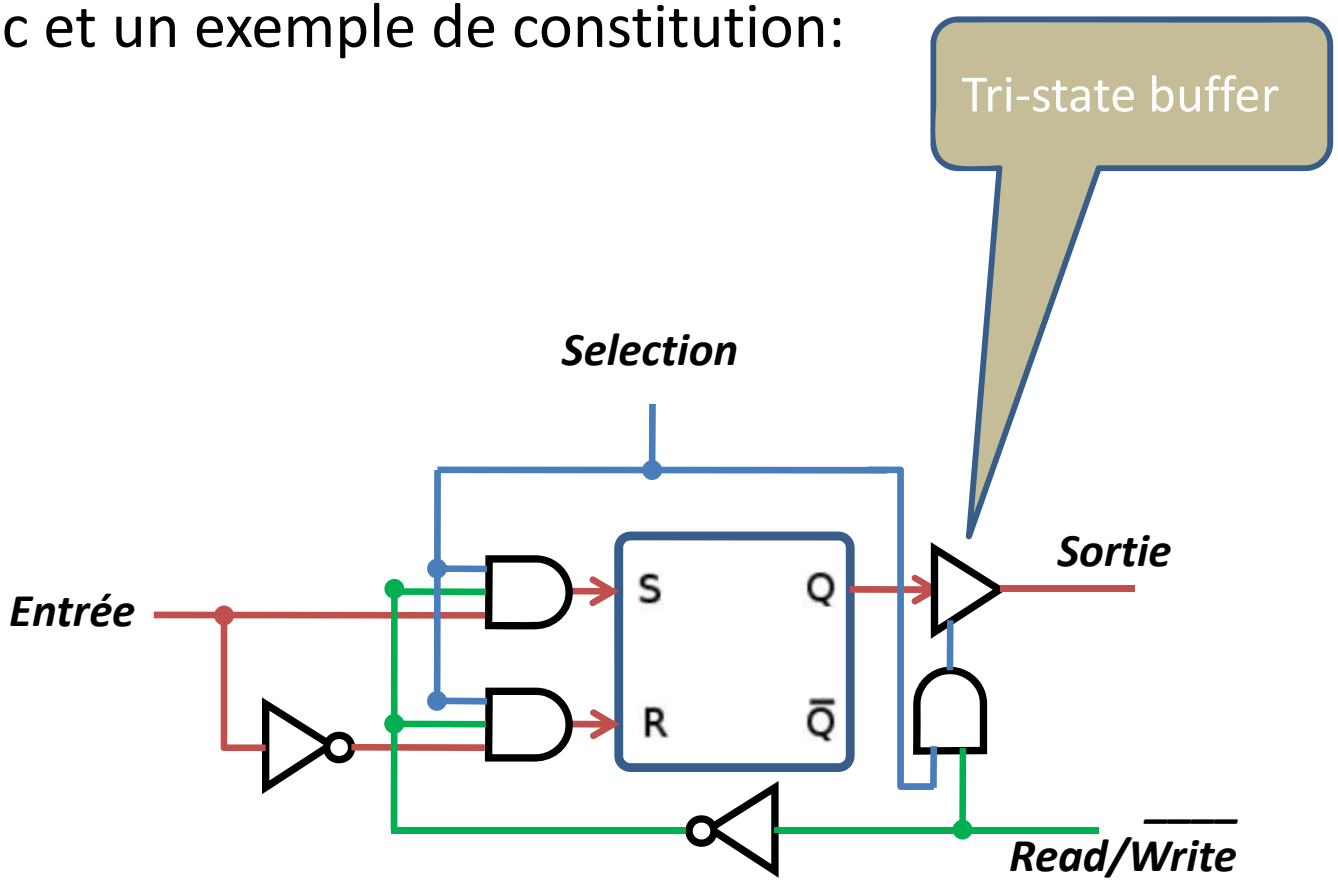
Question: c'est quoi un circuit mémoire?

Un circuit mémoire est un circuit dans lequel on peut ***écrire*** des données, et à partir duquel on peut ***lire*** les ***dernières*** données écrites.

À la base il y a la ***Cellule Mémoire*** (CM)(1 bit)

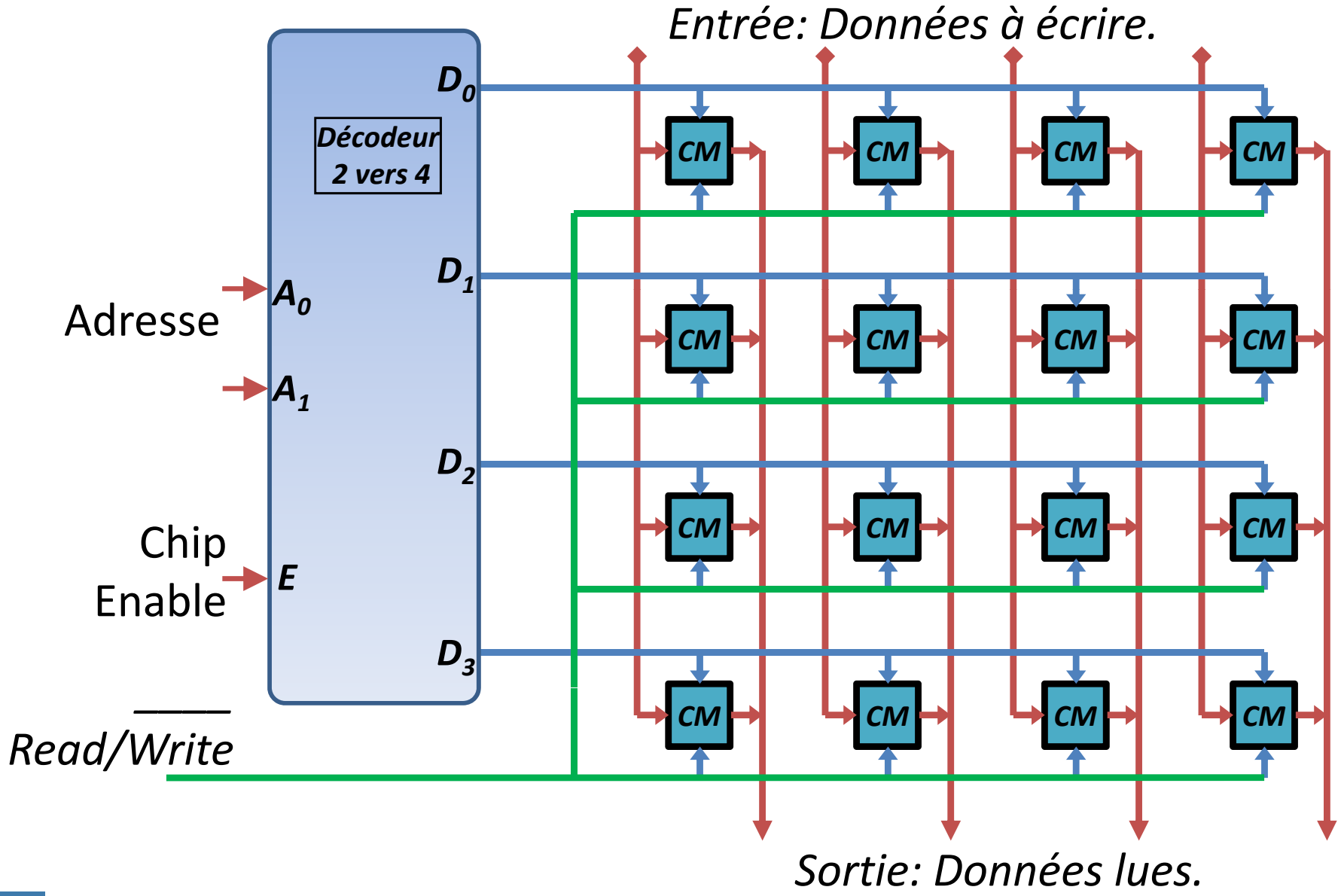
Ci-après son schéma bloc et un exemple de constitution:





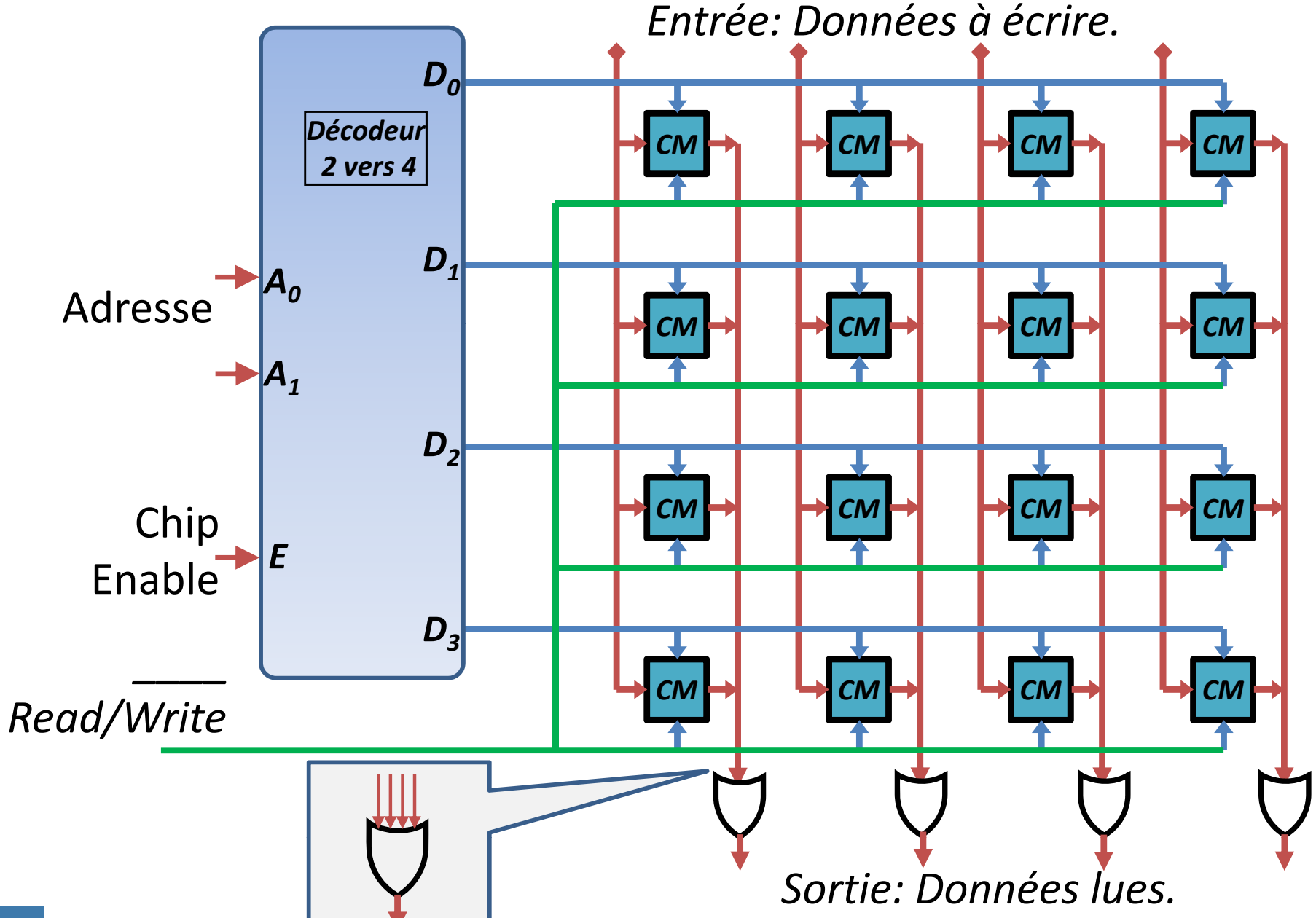
C) *Les réseaux logiques programmables.*

C-2: *Exemple d'un circuit mémoire (RAM): 4 Mots de 4 bits chacun.*



C) *Les réseaux logiques programmables.*

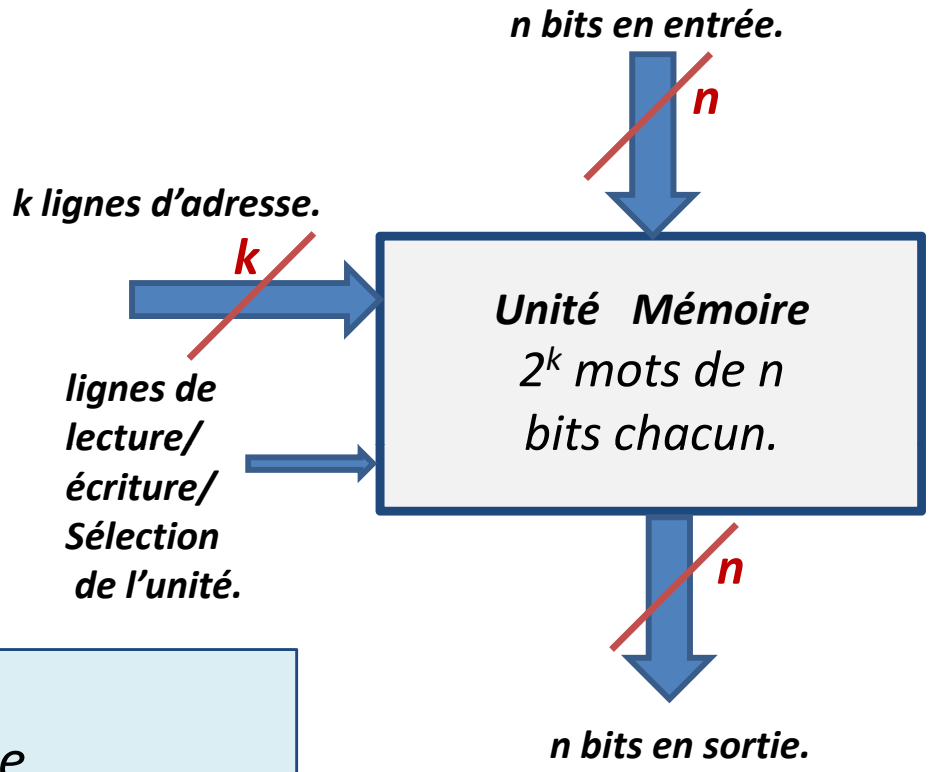
C-3: *Exemple d'un circuit mémoire (RAM): 4 Mots de 4 bits chacun.*



C) *Les réseaux logiques programmables.*

C-4: type de mémoire.

- ✓ n lignes de données appelées généralement bus de donnés.
- ✓ k lignes d'adresse appelées généralement bus d'adresse.
- ✓ Les bus de donnés en entrée et en sortie peuvent être séparés ou confondus (bidirectionnels)
- ✓ Capacité ou taille : 2^k *mots X n bits.*



- Deux types de mémoire

❖La ***RAM*** (**R**andom **A**ccess **M**emory) *volatile.*

❖La **ROM** (**R**ead **O**nly **M**émory). *Non volatile.*

- ➢Factory rom ou **mask-programmed ROM**.
- ➢PROM Programmable ROM.
- ➢EPROM REPROM Erasable Reprogrammable ROM
- ➢EEROM Electrically erasable programmable ROM.

C) *Les réseaux logiques programmables.*

C-5: Exemple de contenu mémoire. (différenciez adresse et contenu).

Chapter 7 Memory and Programmable Logic

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Source: Mano's Book.

FIGURE 7.3
Contents of a 1024 × 16 memory

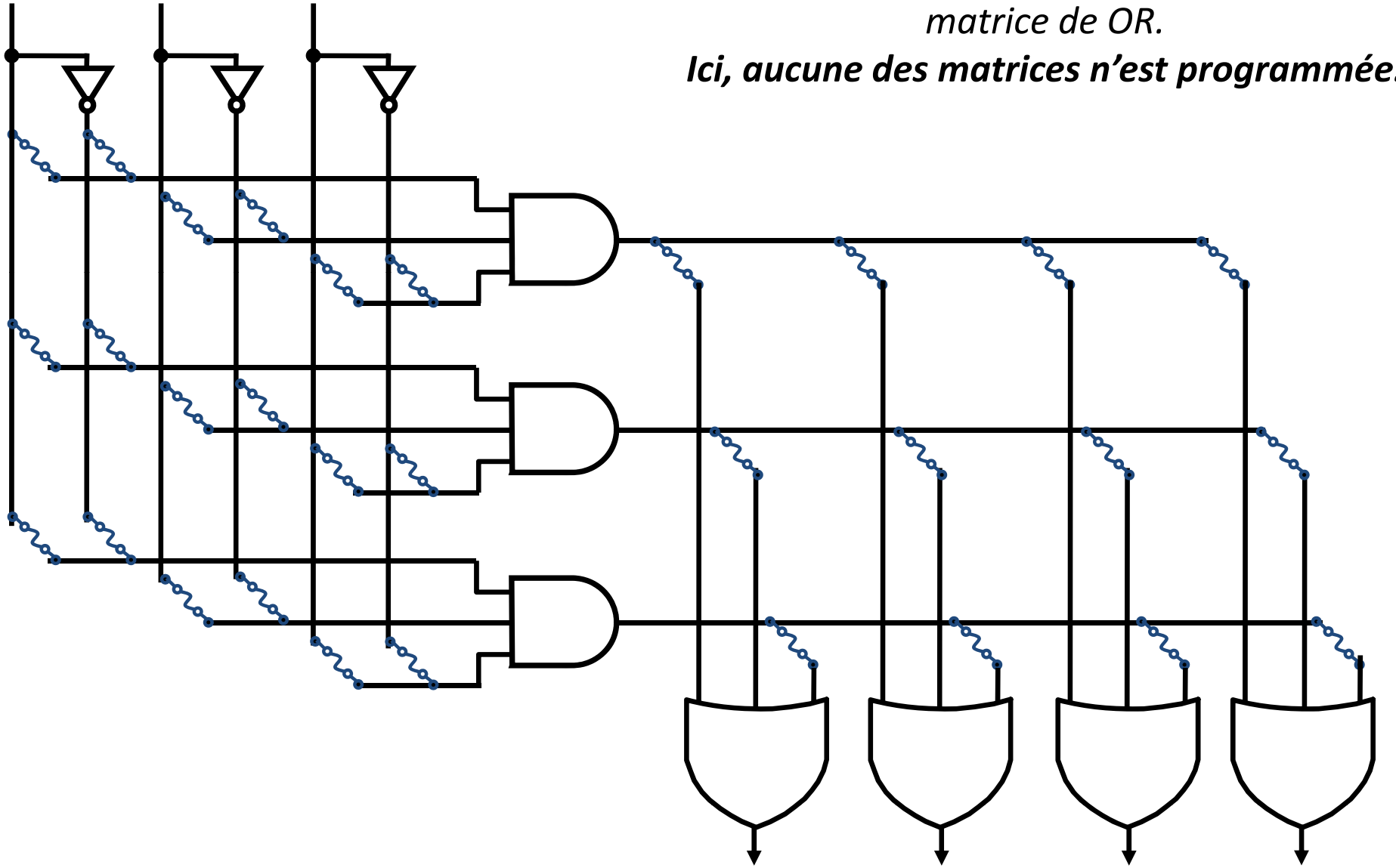
C) *Les réseaux logiques programmables.*

C-6: *Schéma générique, détaillé, d'un PLD simple.*

Les entrées.

Matrice de AND suivie d'une matrice de OR.

Ici, aucune des matrices n'est programmée.



Les sorties.

C) *Les réseaux logiques programmables.*

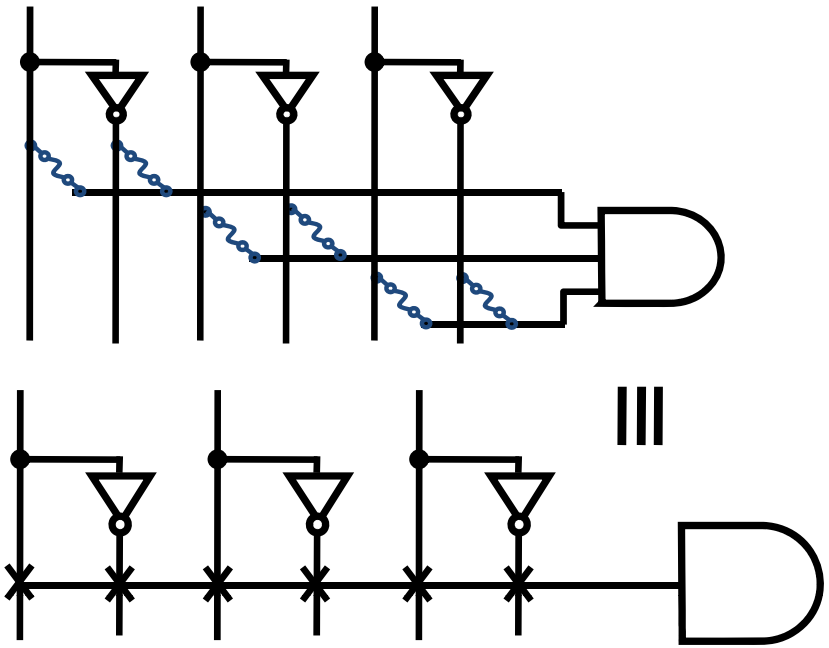
C-7: *Schéma générique d'un PLD simple et équivalence.*

- ✓ *Un fusible intacte, dans la matrice des AND, fait intervenir une entrée ou son complément dans le minterme généré par la porte AND.*
- ✓ *Son absence est équivalent à un 1 logique, élément neutre dans le produit logique.*

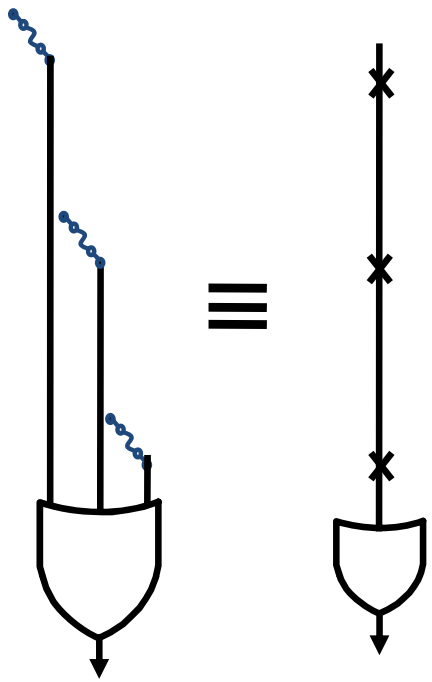
- ✓ *Un fusible intacte, dans la matrice des OR de sortie, fait intervenir un minterme dans la somme générée par la porte OR.*
- ✓ *Son absence est équivalent à un 0 logique, élément neutre dans le somme logique.*

les schémas des PLD simples seront représentés comme suit:

Les entrées.



≡

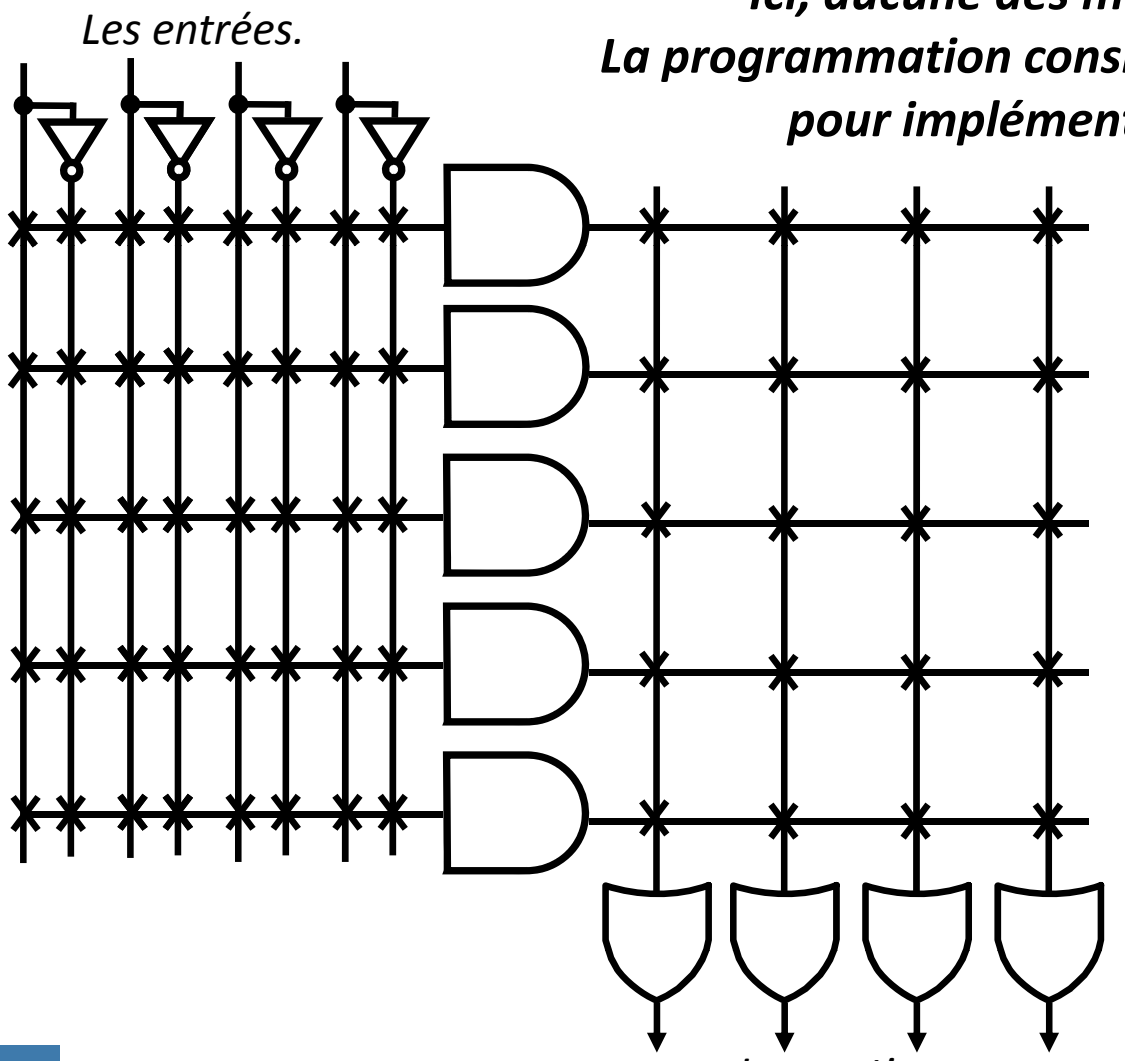


C) *Les réseaux logiques programmables.*

C-8: *Schéma générique, succinct, d'un PLD simple.*

Matrice de AND suivie d'une matrice de OR.

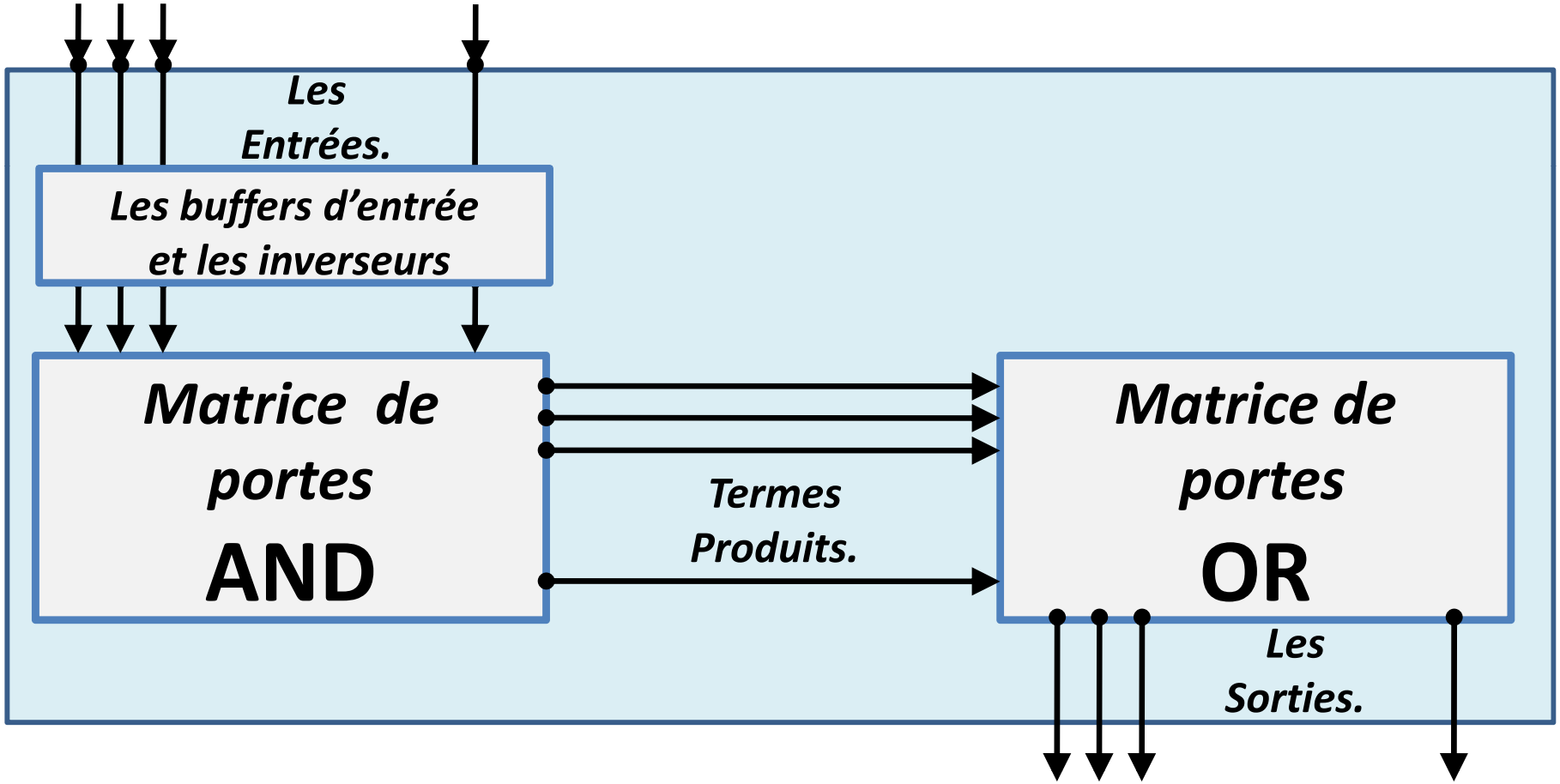
Ici, aucune des matrices n'est programmée.
La programmation consiste à garder ou bruler un fusible pour implémenter une fonction logique.



C) *Les réseaux logiques programmables. Les SPLDs.*

- Simple PLDs

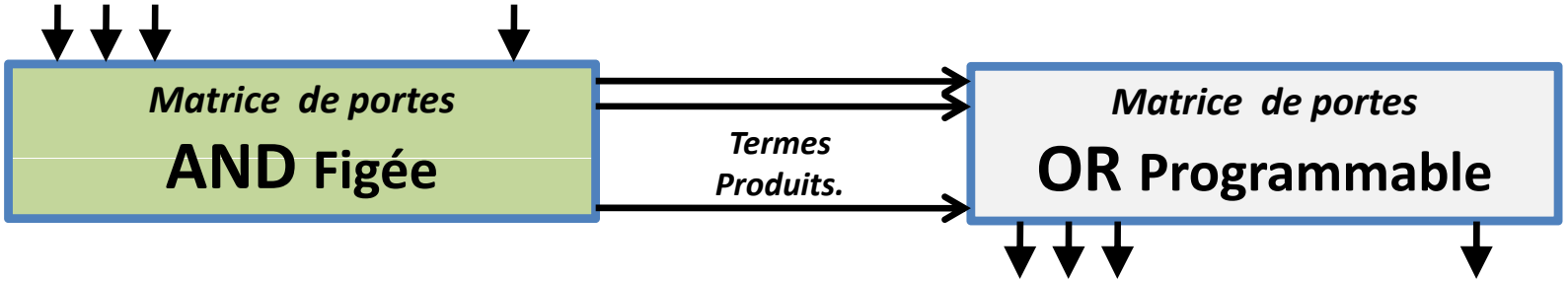
C-9: *Structure générale.*



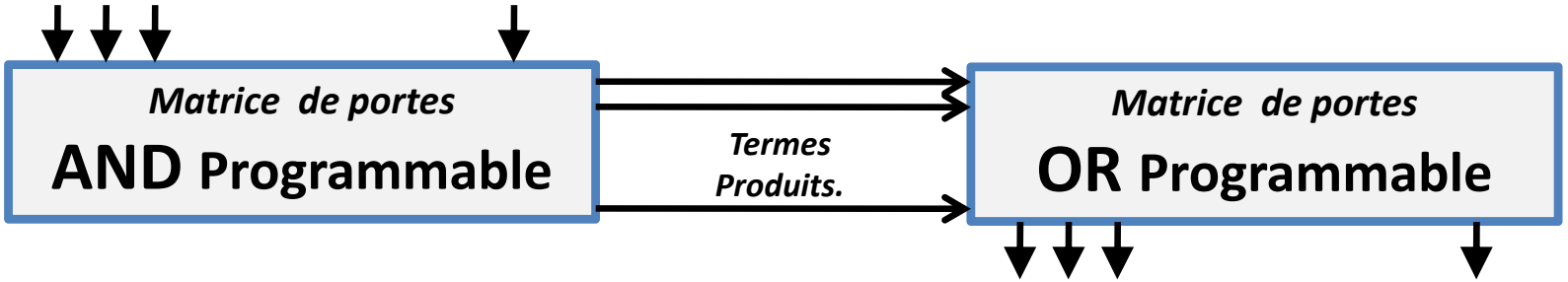
C) *Les réseaux logiques programmables. Les SPLDs.*

C-10: Les variantes.

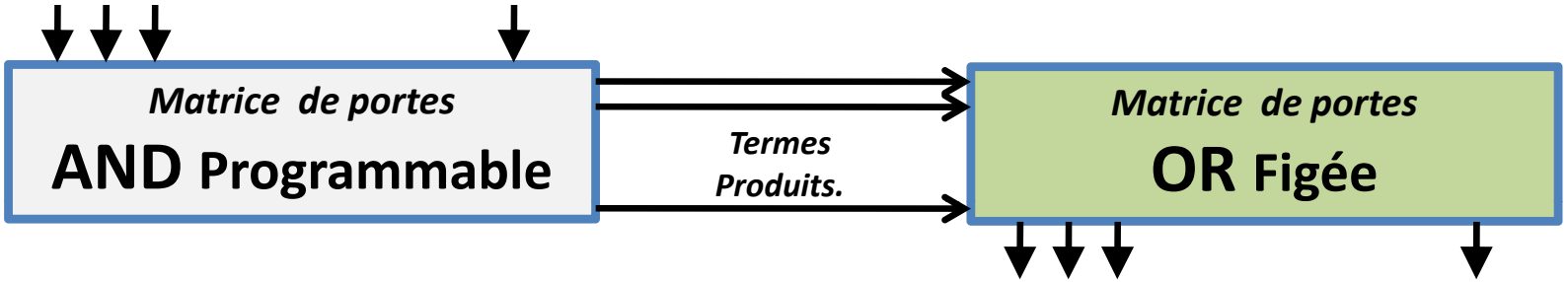
A) Les PROMs Progammable Read Only Memory.



B) Les PLAs Programmable Logic Array.



C) Les PALs Programmable Array Logic.

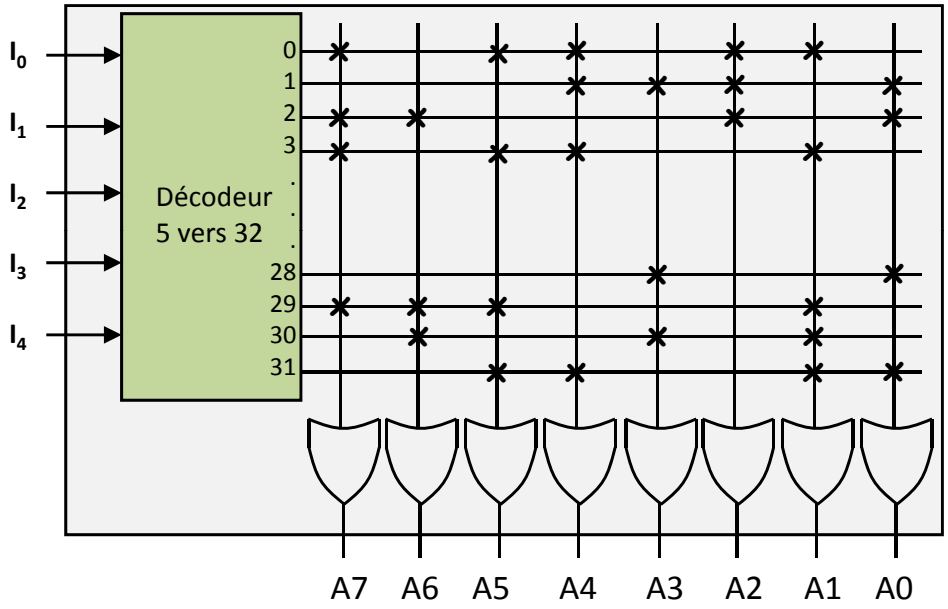


C) *Les réseaux logiques programmables.*

C-11: *Utilisation des PROMs pour réaliser des fonctions logiques.*

La table de vérité des fonctions est reportée en contenu de la ROM.

Entrées					Sorties							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		.						.				
		.						.				
		.						.				
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1



Source:
Roth's
Book.

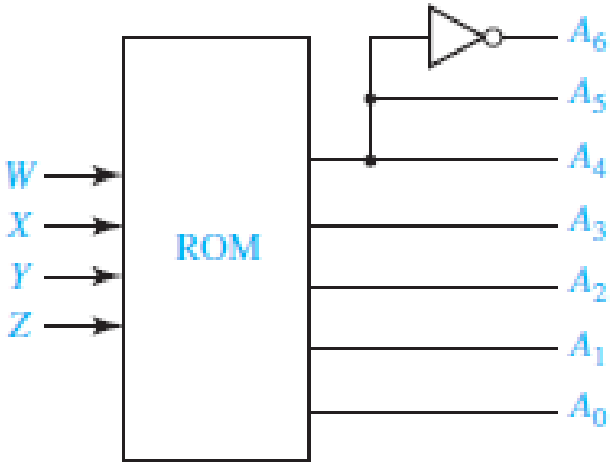
C) *Les réseaux logiques programmables.*

C-11: *Utilisation des PROMs pour réaliser des fonctions logiques.*

Exemple 1*. Conversion hexadécimal – ASCII.*

FIGURE 9-22
Hexadecimal-to-ASCII Code Converter

Input $W\ X\ Y\ Z$	Hex Digit	ASCII Code for Hex Digit $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$
0 0 0 0	0	0 1 1 0 0 0 0
0 0 0 1	1	0 1 1 0 0 0 1
0 0 1 0	2	0 1 1 0 0 1 0
0 0 1 1	3	0 1 1 0 0 1 1
0 1 0 0	4	0 1 1 0 1 0 0
0 1 0 1	5	0 1 1 0 1 0 1
0 1 1 0	6	0 1 1 0 1 1 0
0 1 1 1	7	0 1 1 0 1 1 1
1 0 0 0	8	0 1 1 1 0 0 0
1 0 0 1	9	0 1 1 1 0 0 1
1 0 1 0	A	1 0 0 0 0 0 1
1 0 1 1	B	1 0 0 0 0 1 0
1 1 0 0	C	1 0 0 0 0 1 1
1 1 0 1	D	1 0 0 0 1 0 0
1 1 1 0	E	1 0 0 0 1 0 1
1 1 1 1	F	1 0 0 0 1 1 0



Source: Roth's Book.

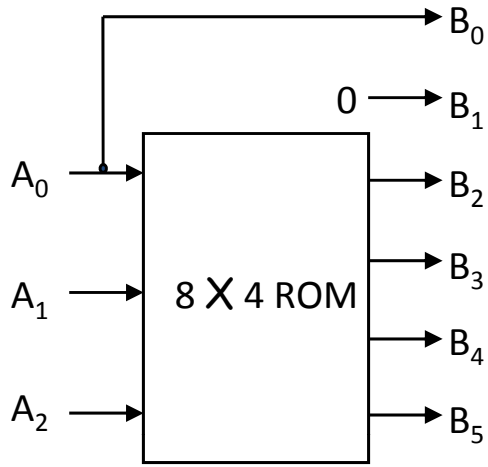
C) *Les réseaux logiques programmables.*

C-11: *Utilisation des PROMs pour réaliser des fonctions logiques.*

Exemple 2*.génération des carrés de nombres de 0 à 7.*

Entrées			Sorties						
A2	A1	A0	B5	B4	B3	B2	B1	B0	Carré
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

Source: Roth's Book.



C) *Les réseaux logiques programmables.*

C-11: *Utilisation des PROMs pour réaliser des fonctions logiques.*

Exemple 3*. Calcul de la taille d'une ROM.*

Problème: *Trouvez la taille de la ROM (Nombre de mots mémoire et le nombre de bits par mot mémoire) pour réaliser, un additionneur soustracteur de deux données codées chacune sur 8 bits , avec prise en compte des deux retenues (entrante/sortante). Le circuit doit générer la somme et la différence.*

- ✓ *Les entrées (8 Bits) + (8 Bits) + retenue entrante font 17*
  Donc le nombre de mots de la ROM est $2^{17} = 2^7 * 2^{10}$ *=128 K mots.*

- ✓ *Les sorties (8 bits de somme) + (8 bits de différence) + retenue sortante font 17 bits par mot.*

La capacité de la ROM: 128 K mots de 17 bits chacun.

C) *Les réseaux logiques programmables.*

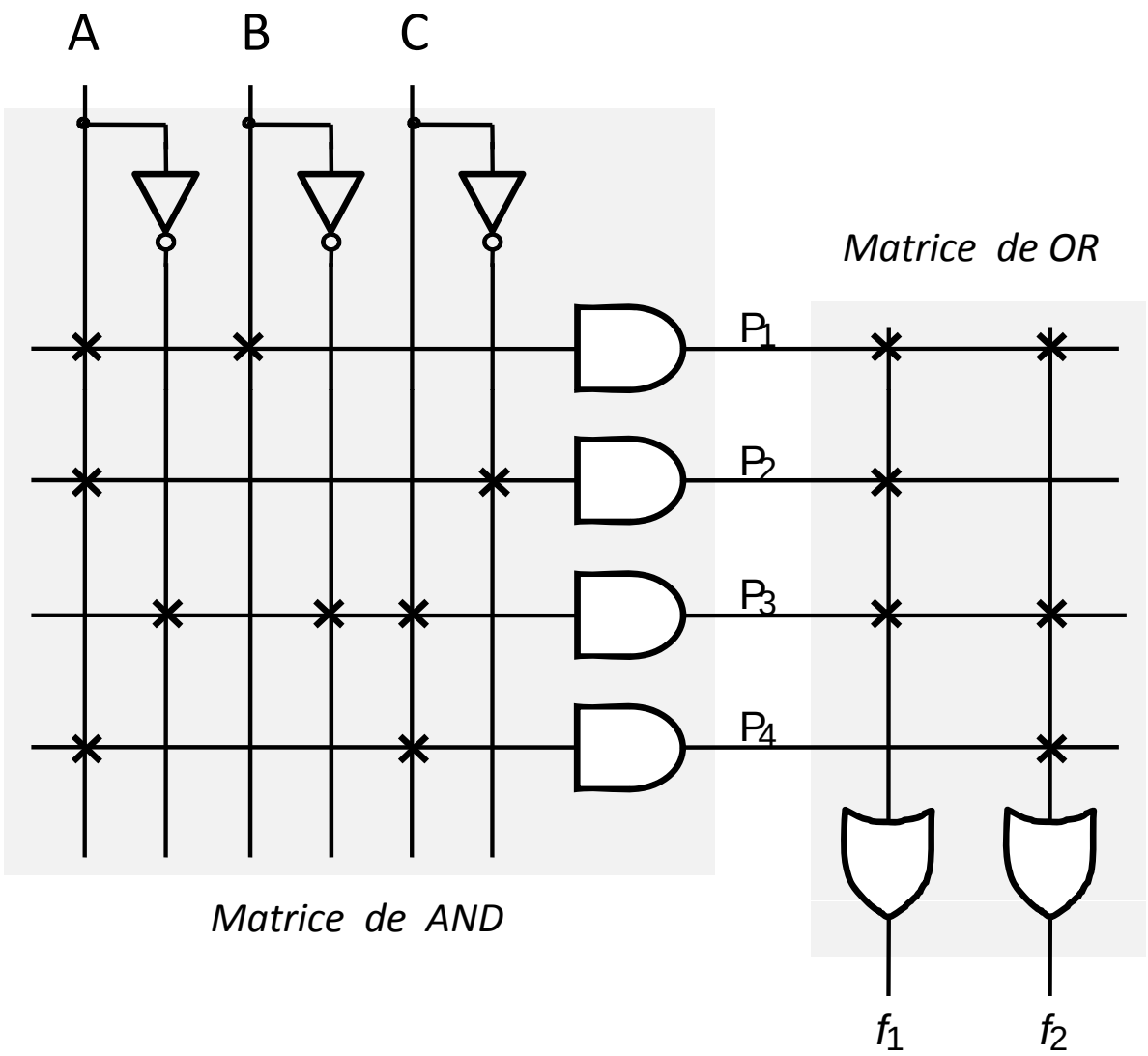
C-12: *Utilisation des **PLA**s pour réaliser des fonctions logiques.*

Exemple 1.

Soit à réaliser les deux fonctions suivantes:

$$f_1 = A.B + A.\overline{C} + \overline{A}.\overline{B}.C$$
$$f_2 = A.B + \overline{A}.\overline{B}.C + A.C$$

Certains produits (P_1 et P_3) sont communs aux deux fonctions, on cherchera à les faire apparaitre.



C) *Les réseaux logiques programmables.*

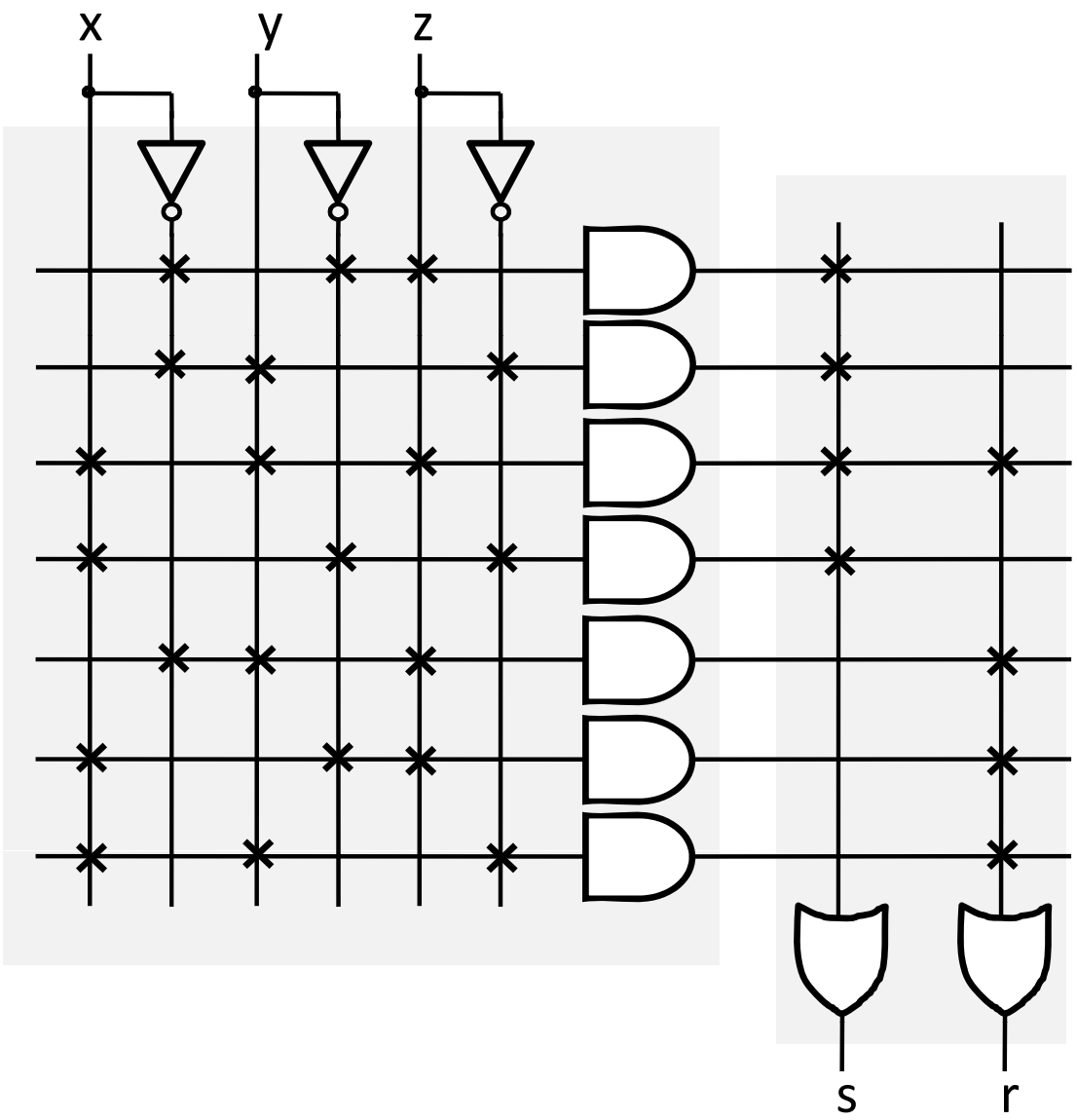
C-12: *Utilisation des **PLA**s pour réaliser des fonctions logiques.*

Exemple 2*. réalisation de l'additionneur complet.*

Rappel des équations logiques des sorties de l'additionneur complet:

$$s = \overline{x}.\overline{y}.z + \overline{x}.y.\overline{z} + x.y.z + x.\overline{y}.\overline{z}$$

$$r = \overline{x}.y.z + x.\overline{y}.z + x.y.\overline{z} + x.y.z$$



C) *Les réseaux logiques programmables.*

C-12: *Utilisation des **PLA**s pour réaliser des fonctions logiques.*

***Exemple 3**. Comparateur*

Comparateur de deux nombres formés de deux bits chacun
$N1=(AB)_2$
$N2=(CD)_2$

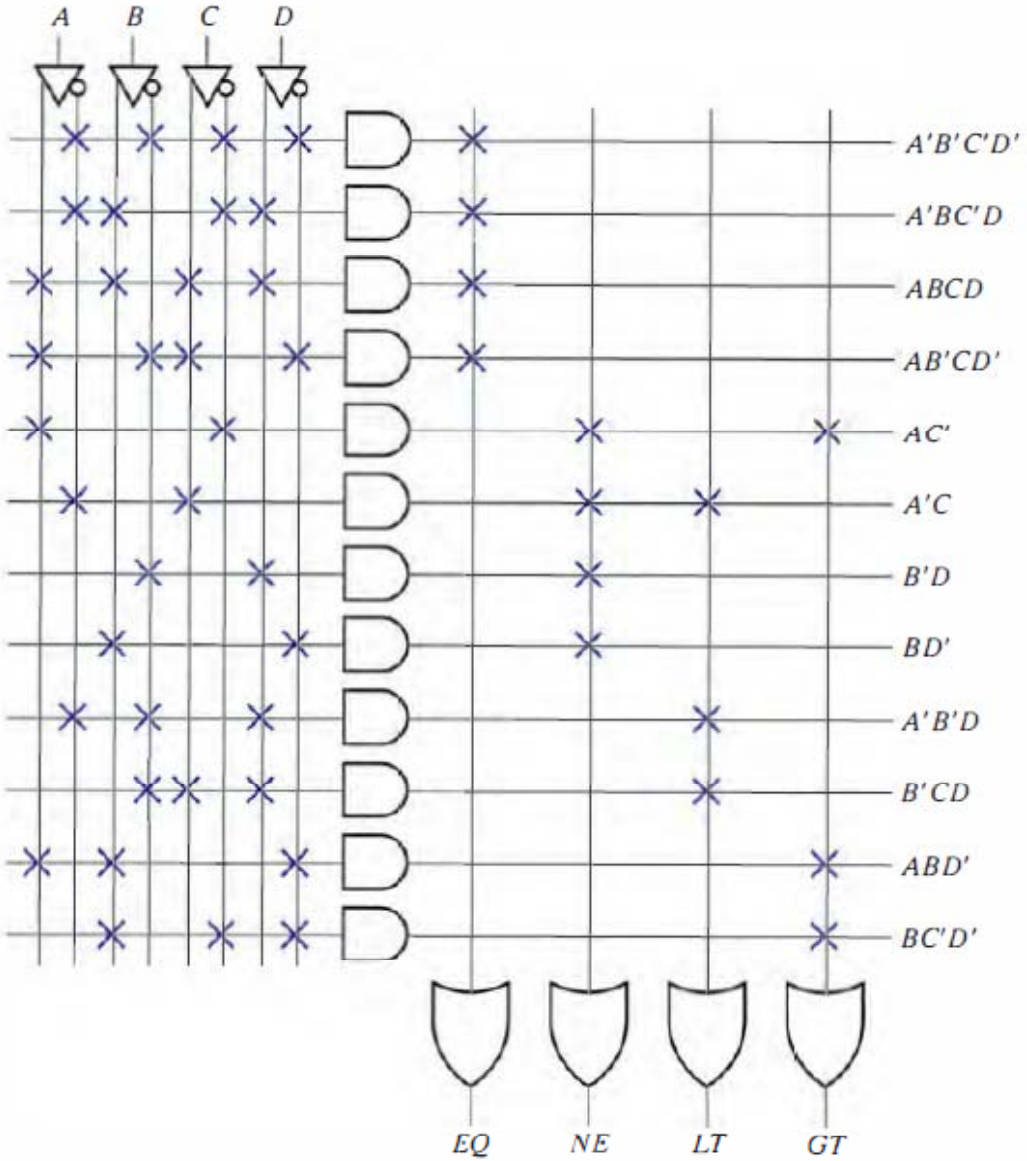


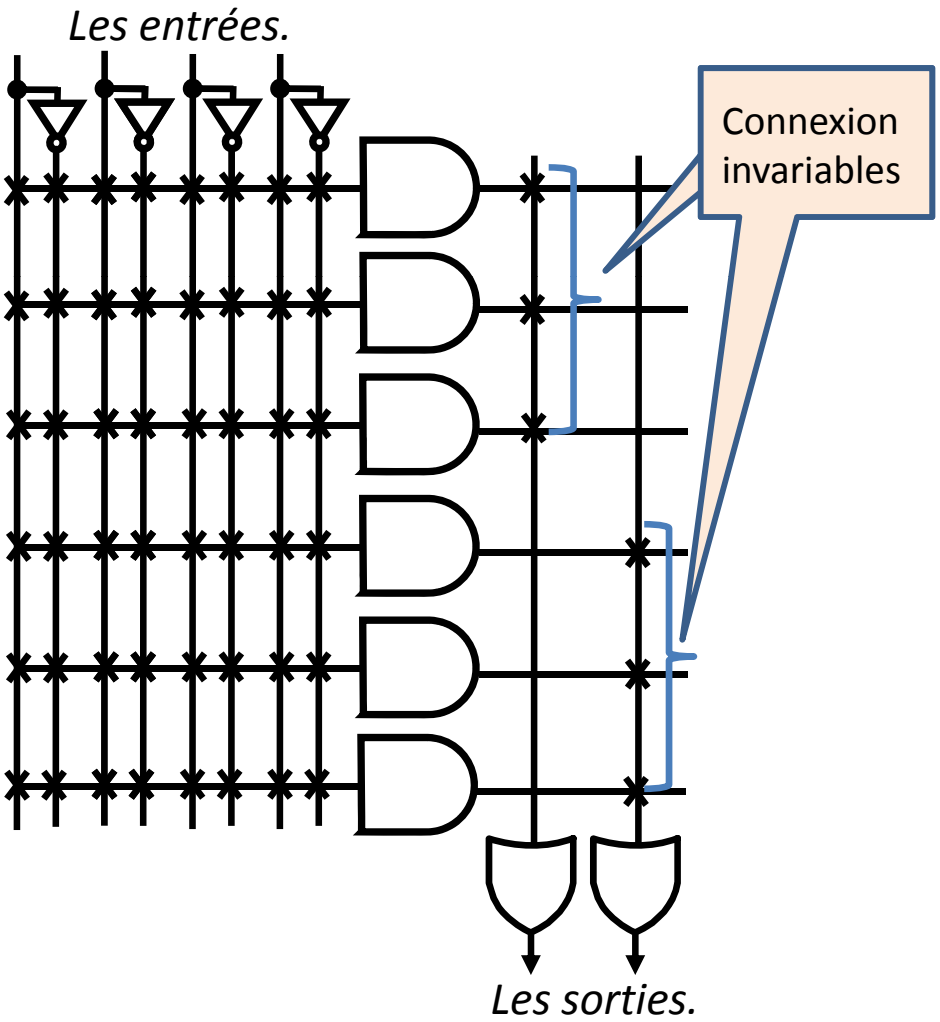
Figure 4.40 **PLA implementation of magnitude comparator.**

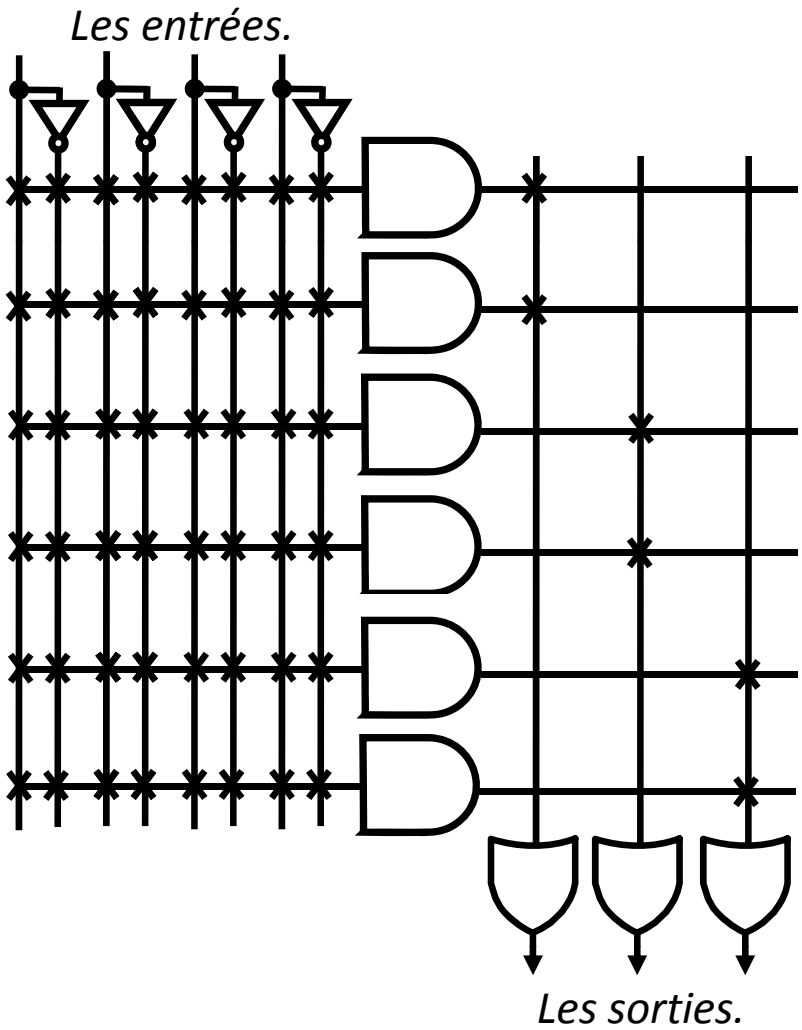
Source: Randy's Book

C) *Les réseaux logiques programmables.*

C-13: *Schéma générique, succinct, d'un PAL.*

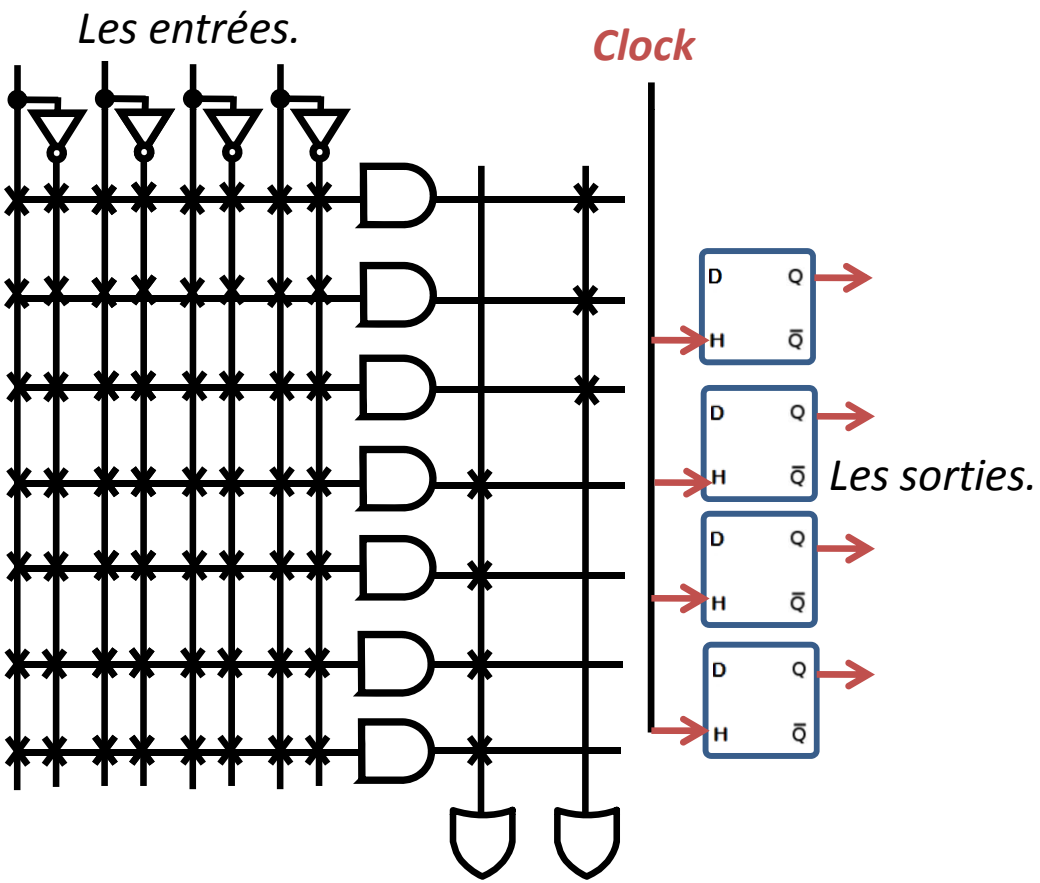
Matrice de AND programmable suivie d'une matrice de OR figée.





C) *Les réseaux logiques programmables.*

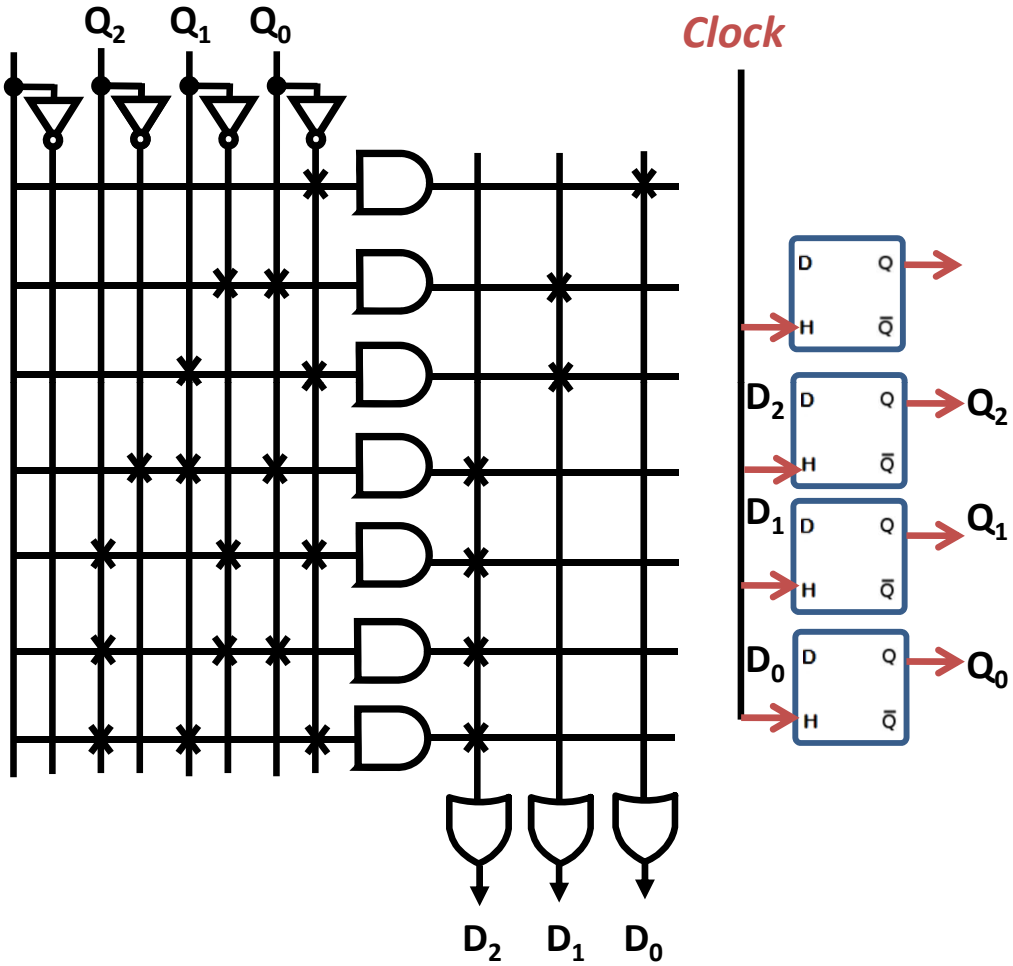
C-13: *Schéma générique, succinct, d'un PAL (séquentiel) avec des éléments de mémorisation (des bistables D). Les connexions aux entrées D sont également programmables à partir des sorties des OR.*



C) *Les réseaux logiques programmables.*

C-14: Exemple: Compteur synchrone Modulo 8.

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	D_2	D_1	D_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0



$$D_0 = \overline{Q_0}$$

$$D_1 = \overline{Q_1}.Q_0 + Q_1.\overline{Q_0}$$

$$D_2 = \overline{Q_2}.Q_1.Q_0 + Q_2.\overline{Q_1}.\overline{Q_0} + Q_2.\overline{Q_1}.Q_0 + Q_2.Q_1.\overline{Q_0}$$