Logique Combinatoire, Séquentielle et applications
LST GI

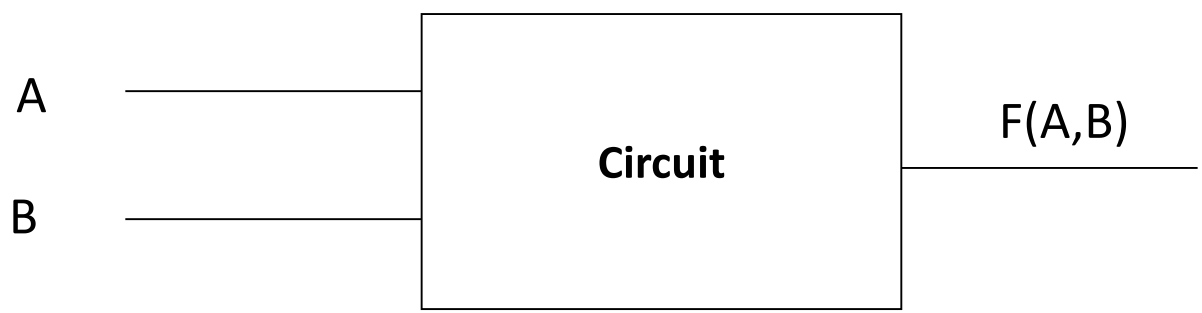
par
Mohamed HASSOUN

Partie 1 :Algèbre de Boole

- Définition des variables et fonctions logiques
- Les opérateurs de base et les portes logiques
- Les lois fondamentales de l'algèbre de Boole

Algèbre de Boole

- Les machines numériques sont constituées d'un ensemble de circuits électroniques.
- Chaque circuit fournit une fonction logique bien déterminée (addition, comparaison ,....).



La fonction F(A,B) peut être : la somme de A et B , ou le résultat de la comparaison de A et B ou une autre fonction

Algèbre de Boole

Pour concevoir et réaliser ce circuit on doit avoir un modèle mathématique de la fonction réalisée par ce circuit .

Ce modèle doit prendre en considération le système binaire.

Le modèle mathématique utilisé est celui de Boole.

Exemple de systèmes à deux états

- Un interrupteur est ouvert ou non ouvert (fermé)
- Une lampe est allumée ou non allumée (éteinte)
- **Remarque :**

On peut utiliser les conventions suivantes :

OUI → VRAI (true)
NON → FAUX (false)

OUI → 1 (Niveau Haut)
NON → 0 (Niveau Bas)

Algèbre de Boole

Définitions et conventions

Niveau logique : Lorsque on fait l'étude d'un système logique il faut bien préciser le niveau du travail.

Niveau	Logique positive	Logique négative
H (Hight) haut	1	0
L (Low) bas	0	1

Exemple :

Logique positive :

lampe allumée : 1

lampe éteinte : 0

Logique négative

lampe allumée : 0

lampe éteinte : 1

Algèbre de Boole

Définitions et conventions

- Une variable logique (booléenne) est une variable qui peut prendre soit la valeur 0 ou 1 .
- Généralement elle est exprimée par un seul caractère alphabétique en majuscule (A , B, S , …)

- **Exemple :**

  - Une lampe : allumée L = 1
    éteinte L = 0

  - interrupteur ouvert : I1 =1
    fermé : I1 =0

Fonction logique

- C'est une fonction qui relie N variables logiques avec un ensemble d'opérateurs logiques de base.
- Dans l'Algèbre de Boole il existe trois opérateurs de base : NON , ET , OU.
- La valeur d'une fonction logique est égale à 1 ou 0 selon les valeurs des variables logiques.
- Si une fonction logique possède N variables logiques → 2^n combinaisons → la fonction possède 2^n valeurs.
- Les 2^n combinaisons sont représentées dans une table qui s'appelle table de vérité (TV).

Fonction logique

Exemple d'une fonction logique

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

La fonction possède 3 variables → 2^3 combinaisons

Opérateurs logiques de base

- NON : est un opérateur unaire (une seule variable) qui à pour rôle d'inverser la valeur d'une variable .

$$F(A)= \text{Non } A = \overline{\mathbf{A}}$$

A	$\overline{\mathbf{A}}$
0	**1**
1	**0**

Algèbre de Boole

Opérateurs logiques de base

- Le ET est un opérateur binaire (deux variables) , à pour rôle de réaliser le Produit logique entre deux variables booléennes.
- Le ET fait la conjonction entre deux variables.

- Le ET est défini par : $F(A,B)= A \cdot B$

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

Algèbre de Boole

Opérateurs logiques de base

- Le OU est un opérateur binaire, à pour rôle de réaliser la somme logique entre deux variables logiques.
- Le OU fait la disjonction entre deux variables.
- Le OU est défini par F(A,B)= A + B

(il ne faut pas confondre avec la somme arithmétique)

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Algèbre de Boole

- Dans la définition des opérateurs ET , OU , nous avons juste donner la définition de base avec deux variables logiques.

- L'opérateur ET peut réaliser le produit de plusieurs variables logique (ex : A . B . C . D).

- L'opérateur OU peut aussi réaliser la somme logique de plusieurs variables logiques (ex : A + B + C +D).

- Dans une expression on peut aussi utiliser les parenthèses.

Algèbre de Boole

- Pour évaluer une expression logique (fonction logique) :
  - on commence par évaluer les sous expressions entre les parenthèses.
  - puis le complément (NON) ,
  - en suite le produit logique (ET)
  - enfin la somme logique (OU)

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

si on veut calculer $F(0,1,1)$ alors:

$$F(0,1,1) = (\overline{0.1})(1+1) + 0.\overline{1}.1$$

$$F(0,1,1) = (\overline{0}\,)\,(1) + 0.0.1$$

$$F(0,1,1) = 1.1 + 0.0.1$$

$$F(0,1,1) = 1 + 0$$

$$F(0,1,1) = 1$$

Exercice :

Trouver la table de vérité de la fonction suivante

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

Lois fondamentales

- **L'opérateur NON**

$$\overline{\overline{A}} = A$$

$$\overline{A} + A = 1$$

$$\overline{A}.A = 0$$

Algèbre de Boole

Lois fondamentales

- **L'opérateur ET**

$(A.B).C = A.(B.C) = A.B.C$	Associativité
$A.B = B.A$	Commutativité
$A.A = A$	Idempotence
$A.1 = A$	Elément neutre
$A.0 = 0$	Elément absorbant

Algèbre de Boole

Lois fondamentales

- **L'opérateur OU**

$(A+B)+C = A+(B+C) = A+B+C$	Associativité
$A+B = B+A$	Commutativité
$A+A = A$	Idempotence
$A+0 = A$	Elément neutre
$A+1 = 1$	Elément absorbant

Lois fondamentales

- **Dualité de l'algèbre de Boole**

  - Toute expression logique reste vrais si on remplace le ET par le OU , le OU par le ET , le 1 par 0 , le 0 par 1.

  - Exemple :

$$A + 1 = 1 \longrightarrow A \,.\, 0 = 0$$

$$A + \overline{A} = 1 \longrightarrow A \,.\, \overline{A} = 0$$

Algèbre de Boole

Lois fondamentales

- **Théorème de DE-MORGANE**

- La somme logique complimentée de deux variables est égale au produit des compléments des deux variables.

$$\overline{A+B} = \overline{A} \, . \, \overline{B}$$

- Le produit logique complimenté de deux variables est égale au somme logique des compléments des deux variables.

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

Autres opérateurs logiques
OU exclusif (XOR)

$$F(A,B) = A \oplus B$$

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

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Algèbre de Boole

NAND (NON ET)

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

$$F(A, B) = A \uparrow B$$

A	B	$\overline{A \bullet B}$
0	0	1
0	1	1
1	0	1
1	1	0

Algèbre de Boole

NOR (NON OU)

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

$$F(A, B) = A \downarrow B$$

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

7.4 NAND et NOR sont des opérateurs universels

- En utilisant les NAND et les NOR on peut exprimer n'importe qu'elle expression (fonction) logique.

- Pour cela , Il suffit d'exprimer les opérateurs de base (NON , ET , OU) avec des NAND et des NOR.

Réalisation des opérateurs de base avec des NOR

$$\overline{A} = \overline{A + A} = A \downarrow A$$

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

$$A.B = \overline{\overline{A.B}} = \overline{\overline{A} + \overline{B}} = \overline{A} \downarrow \overline{B} = (A \downarrow A) \downarrow (B \downarrow B)$$

Algèbre de Boole

Propriétés des opérateurs NAND et NOR

$$A \uparrow 0 = 1$$

$$A \uparrow 1 = \overline{A}$$

$$A \uparrow B = B \uparrow A$$

$$(A \uparrow B) \uparrow C \neq A \uparrow (B \uparrow C)$$

$$A \downarrow 0 = \overline{A}$$

$$A \downarrow 1 = 0$$

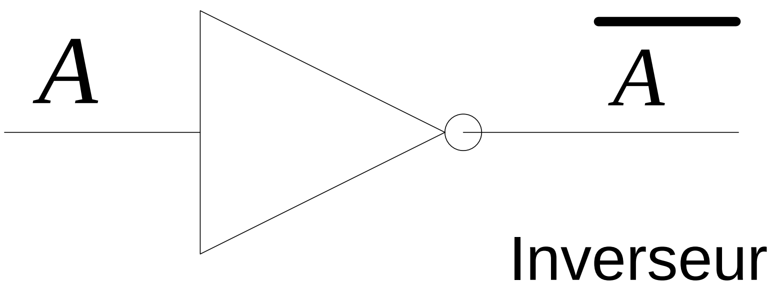
$$A \downarrow B = B \downarrow A$$

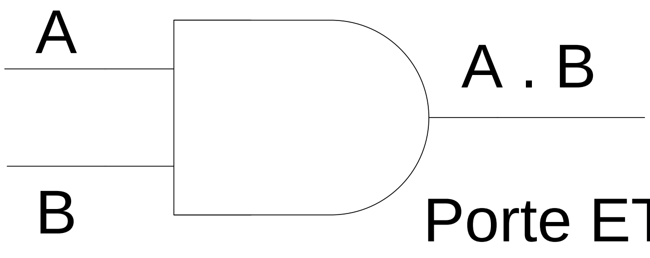
$$(A \downarrow B) \downarrow C \neq A \downarrow (B \downarrow C)$$

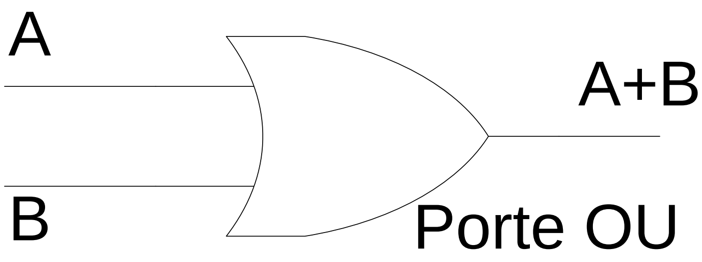
Les portes logiques

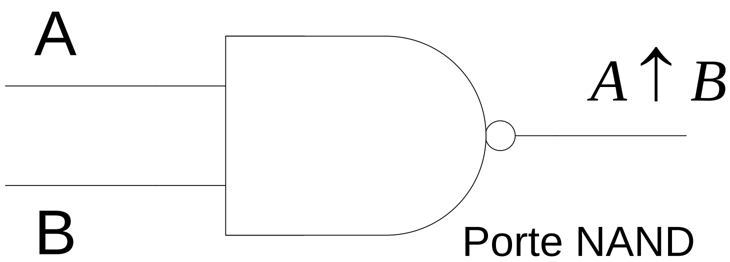
Portes logiques

Une porte logique est un circuit électronique élémentaire qui Permet de réaliser la fonction d'un opérateur logique de base .

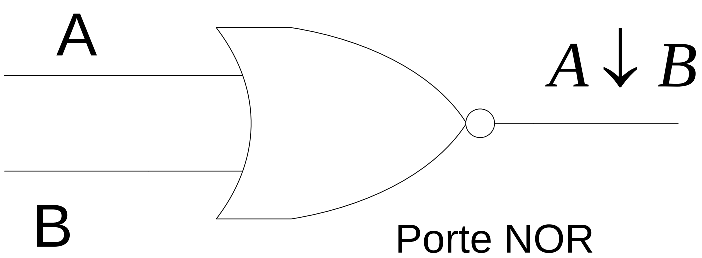




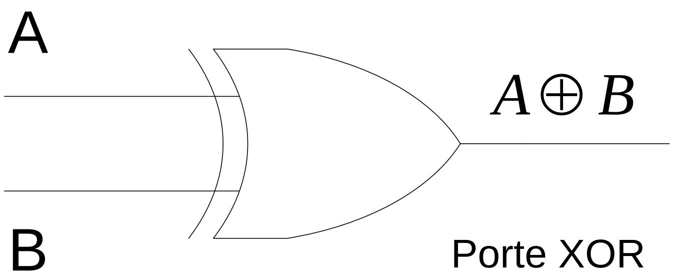




Porte NAND



Porte NOR



Porte XOR

Remarque :

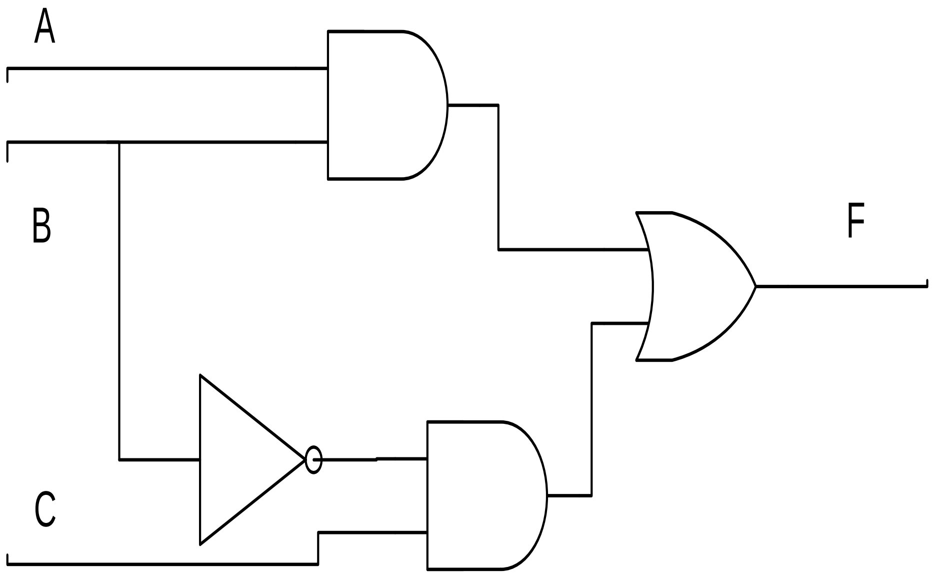
- Les portes ET , OU , NAND , NOR peuvent avoir plus que deux entrées
- Il n'existe pas de OU exclusif à plus de deux entrées

Schéma d'un circuit logique (Logigramme)

- C'est la traduction de la fonction logique en un schéma électronique.
- Le principe consiste à remplacer chaque opérateur logique par la porte logique qui lui correspond.

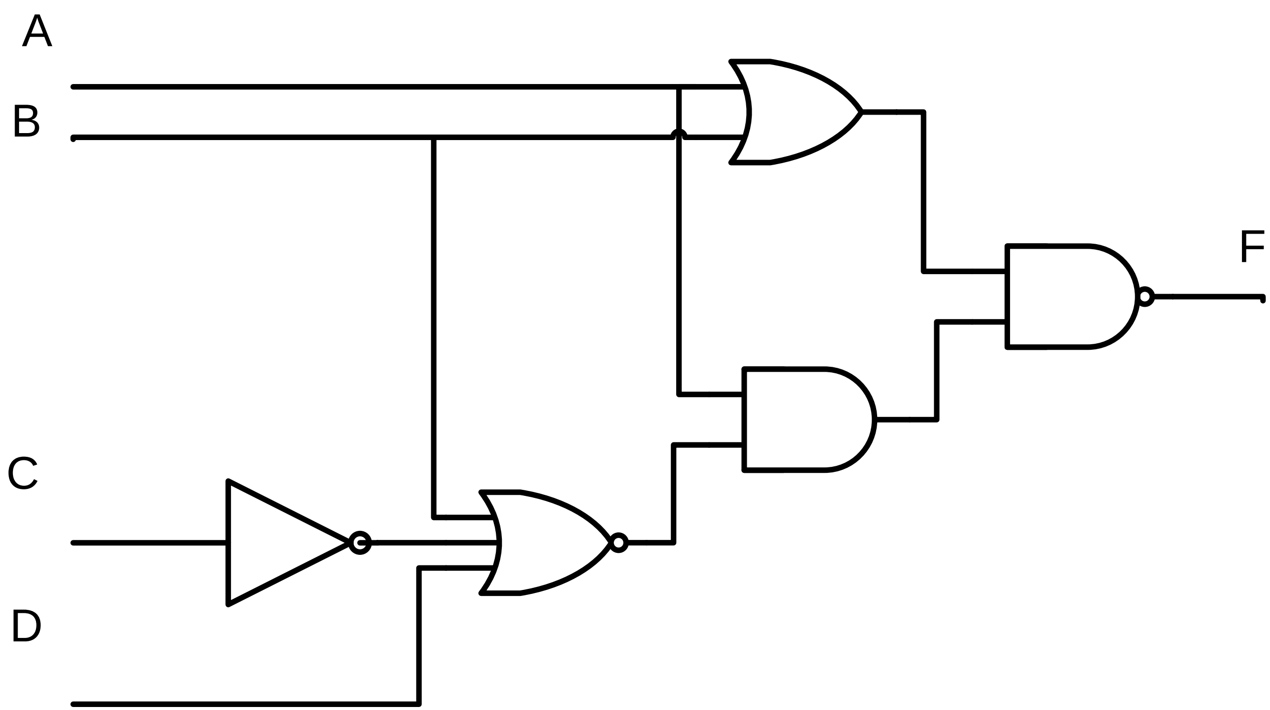
Exemple1

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



Exemple 2

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



Exercice 1

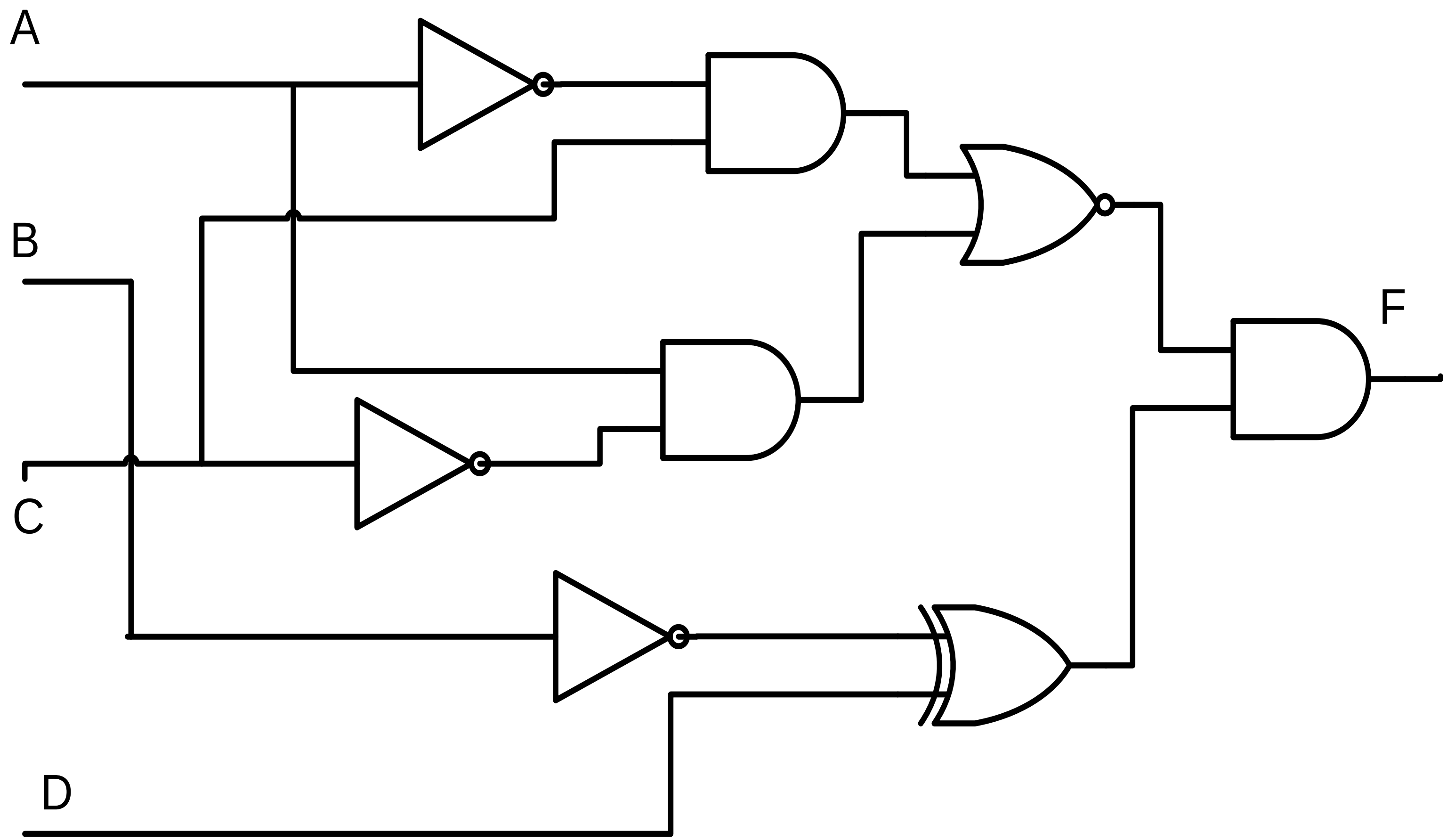
- Donner le logigramme des fonctions suivantes :

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

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

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

Exercice 2 : Donner l'équation de F ?



Définition textuelle d'une fonction logique, table de vérité , formes algébriques , simplification algébrique.

Définition textuelle d'une fonction logique

- Généralement la définition du fonctionnement d'un système est donnée sous un format textuelle .
- Pour faire l'étude et la réalisation d'un tel système on doit avoir son modèle mathématique (fonction logique).
- Donc il faut tirer (déduire) la fonction logique a partir de la description textuelle.

Exemple : définition textuelle du fonctionnement d’un système

- Une serrure de sécurité s’ouvre en fonction de trois clés. Le fonctionnement de la serrure est définie comme suit :
  - La serrure est ouverte si au moins deux clés sont utilisées.
  - La serrure reste fermée dans les autres cas .

Donner la schéma du circuit qui permet de contrôler l’ouverture de la serrure ?

Étapes de conception et de réalisation d’un circuit numérique

- Pour faire l’étude et la réalisation d’un circuit il faut suivre le étapes suivantes :

1. Il faut bien comprendre le fonctionnement du système.
2. Il faut définir les variables d’entrée.
3. Il faut définir les variables de sortie.
4. Etablir la table de vérité.
5. Ecrire les équations algébriques des sorties (à partir de la table de vérité).
6. Effectuer des simplifications (algébrique ou par Karnaugh).
7. Faire le schéma avec un minimum de portes logiques.

Si on reprend l'exemple de la serrure :

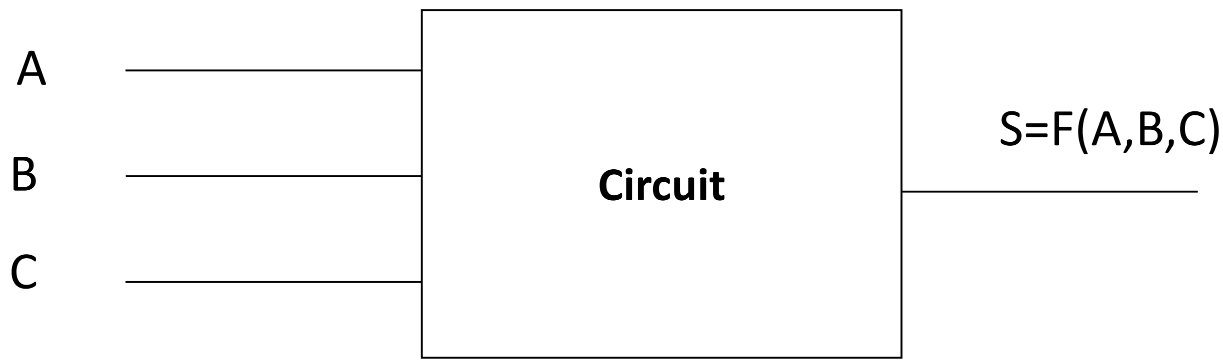
- Le système possède trois entrées : chaque entrée représente une clé.
- On va correspondre à chaque clé une variable logique: clé 1 → A , la clé 2 → B , la clé 3 → C
  - Si la clé 1 est utilisée alors la variable A=1 sinon A =0
  - Si la clé 2 est utilisée alors la variable B=1 sinon B =0
  - Si la clé 3 est utilisée alors la variable C=1 sinon C =0

- Le système possède une seule sortie qui correspond à l'état de la serrure (ouverte ou fermé).
- On va correspondre une variable S pour designer la sortie :
  - S=1 si la serrure est ouverte ,
  - S=0 si elle est fermée

S=F(A,B,C)

F(A,B,C)= 1 si au mois deux clés sont introduites

F(A,B,C)=0 si non .



<u>Remarque :</u>

Il est important de préciser aussi le niveau logique avec lequel on travail (logique positive ou négative).

Table de vérité

- Si une fonction logique possède N variables logiques → 2^n combinaisons → la fonction possède 2^n valeurs.

- Les 2^n combinaisons sont représentées dans une table qui s'appelle table de vérité.

Table de vérité (Exemple)

A	B	C		S
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**

$\rightarrow A + B + C$: max terme

$\rightarrow A + B + \overline{C}$: max terme

$\rightarrow A + \overline{B} + C$: max terme

$\rightarrow \overline{A} . B . C$: min terme

$\rightarrow \overline{A} + B + C$: max terme

$\rightarrow A . \overline{B} . C$: min terme

$\rightarrow A . B . \overline{C}$: min terme

$\rightarrow A . B . C$: min terme

Extraction de la fonction logique à partir de la T.V

- F = somme min termes

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

- F = produit des max termes

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

Forme canonique d’une fonction logique

- On appel forme canonique d’une fonction la forme ou chaque terme de la fonction comportent toutes les variables.

- Exemple :

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

Il existent plusieurs formes canoniques : les plus utilisées sont la première et la deuxième forme .

Première forme canonique

- Première forme canonique (forme disjonctive) : somme de produits
- C'est la somme des min termes.
- Une disjonction de conjonctions.

- Exemple :

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

- Cette forme est la forme la plus utilisée.

Deuxième forme canonique

- Deuxième forme canonique (conjonctive): produit de sommes
- Le produit des max termes
- Conjonction de disjonctions
- Exemple :

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

La première et la deuxième forme canonique sont équivalentes .

Remarque 1

- On peut toujours ramener n'importe qu'elle fonction logique à l'une des formes canoniques.

- Cela revient à rajouter les variables manquants dans les termes qui ne contiennent pas toutes les variables (les termes non canoniques).

- Cela est possible en utilisant les règles de l'algèbre de Boole :
  - Multiplier un terme avec une expression qui vaut 1
  - Additionner à un terme avec une expression qui vaut 0
  - Par la suite faire la distribution

Exemple :

$$1. F(A, B) = A + B$$

$$2. F(A, B, C) = AB + C$$

Remarque 2

- Il existe une autre représentation des formes canoniques d'une fonction , cette représentation est appelée forme numérique.
- R : pour indiquer la forme disjonctive
- P : pour indiquer la forme conjonctive.

Exemple : si on prend une fonction avec 3 variables

$$R(2,4,6) = \sum(2,4,6) = R(010,100,110) = \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + AB\overline{C}$$

$$P(0,1,3,5,7) = \prod(0,1,3,5,7) = P(000,001,011,101,111)$$

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

Exercice 3

Un jury composé de 4 membres pose une question à un joueur, qui à son tour donne une réponse. Chaque membre du jury positionne son interrupteur à " 1 " lorsqu'il estime que la réponse donnée par le joueur est juste (avis favorable) et à " 0 " dans le cas contraire (avis défavorable). On traite la réponse de telle façon à positionner :

- Une variable succès (S=1) lorsque la décision de la majorité des membres de jury est favorable,
- une variable Échec (E=1) lorsque la décision de la majorité des membres de jury est défavorable
- et une variable Égalité (N=1) lorsqu’il y a autant d'avis favorables que d'avis défavorables.

Question :

a./ Déduire une table de vérité pour le problème,

b./ Donner les équations de S, E,

c./ En déduire l’équation de N,

Simplification des fonctions logiques

Simplification des fonctions logiques

- L'objectif de la simplification des fonctions logiques est de :
  - réduire le nombre de termes dans une fonction
  - et de réduire le nombre de variables dans un terme

- Cela afin de réduire le nombre de portes logiques utilisées → réduire le coût du circuit

- Plusieurs méthodes existent pour la simplification :
  - La Méthode algébrique
  - Les Méthodes graphiques : (ex : table de karnaugh)
  - Les méthodes programmables

Règles de simplification

- Règles 1 : regrouper des termes à l'aide des règles précédentes

- Exemple

$$
\begin{aligned}
ABC + AB\overline{C} + A\overline{B}CD &= AB\,(C + \overline{C}) + A\overline{B}CD \\
&= AB + A\overline{B}CD \\
&= A\,(B + \overline{B}\,(CD)) \\
&= A\,(B + CD) \\
&= AB + ACD
\end{aligned}
$$

- Règles 2 : Rajouter un terme déjà existant à une expression

- Exemple :

$$A\,B\,C + \overline{A}BC + A\overline{B}C + AB\overline{C} =$$

$$ABC + \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} =$$

$$BC + AC + AB$$

- **Règles 3 :** il est possible de supprimer un terme superflu (un terme en plus), c'est-à-dire déjà inclus dans la réunion des autres termes.

- Exemple 1 :

$$
\begin{aligned}
F(A,B,C) = A\,B + \overline{B}C + AC &= AB + \overline{B}C + AC\,(\,B + \overline{B}) \\
&= AB + \overline{B}C + ACB + A\overline{B}C \\
&= AB\,(1 + C) + \overline{B}C\,(1 + A) \\
&= AB + \overline{B}C
\end{aligned}
$$

Exemple 2 : il existe aussi la forme conjonctive du terme superflu

$$
\begin{aligned}
F(A,B,C) &= (A+B)\,.\,(\overline{B}+C)\,.\,(A+C) \\
&= (A+B)\,.\,(\overline{B}+C)\,.\,(A+C+B.\overline{B}) \\
&= (A+B)\,.\,(\overline{B}+C)\,.\,(A+C+B)\,.(A+C+\overline{B}) \\
&= (A+B)\,.\,\ (A+C+B)\,.\,(\overline{B}+C)\,.(A+C+\overline{B}) \\
&= (A+B)\,.\,\ (\overline{B}+C)
\end{aligned}
$$

Simplification par la table de Karnaugh

Les termes adjacents

- Examinons l'expression suivante :

$$\mathrm{A.B+A.\overline{B}}$$

- Les deux termes possèdent les même variables. La seule différence est l'état de la variable B qui change.
- Si on applique les règles de simplification on obtient :

$$AB + A\overline{B} = A(B + \overline{B}) = A$$

- Ces termes sont dites adjacents.

Exemple de termes adjacents

Ces termes sont adjacents

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

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

$$A.B.C.D + A.B.\overline{C}.D = A.B.D$$

Ces termes ne sont pas adjacents

$$A.B + \overline{A}.\overline{B}$$

$$A.B.C + A.\overline{B}.\overline{C}$$

$$A.B.C.D + \overline{A}.B.\overline{C}.D$$

Description de la table de karnaugh

- La méthode de Karnaugh se base sur la règle précédente.
- La méthode consiste a mettre en évidence par une méthode graphique (un tableaux) tous les termes qui sont adjacents (qui ne différent que par l'état d'une seule variable).
- La méthode peut s'appliquer aux fonctions logiques de 2,3,4,5 et 6 variables.
- Un tableau de Karnaugh comportent 2^n cases (N est le nombre de variables).

B \ A	0	1
0		
1		

Tableau à 2 variables

C \ AB	00	01	11	10
0				
1				

Tableaux à 3 variables

Tableau à 4 variables

CD \ AB	00	01	11	10
00				
01				
11				
10				

Tableau à 5 variables

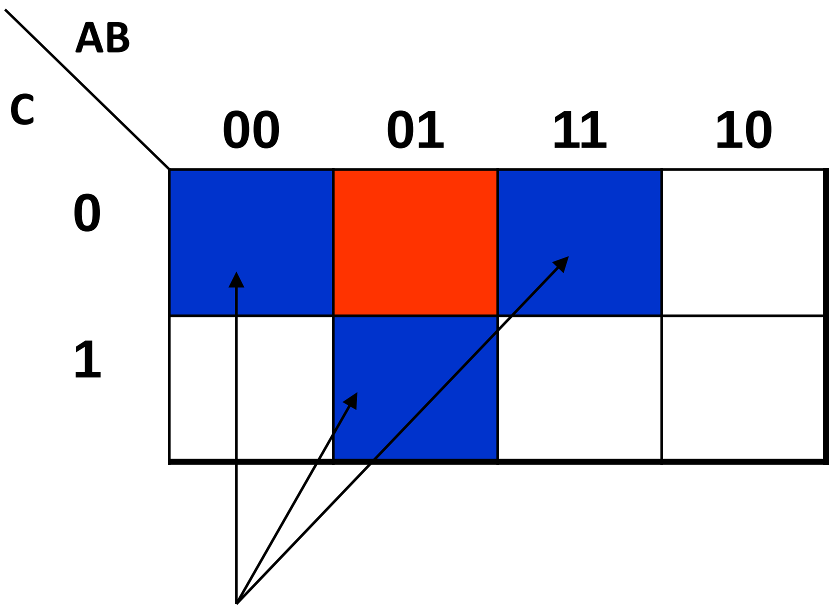
CD \ AB	00	01	11	10
00				
01				
11				
10				

U = 0

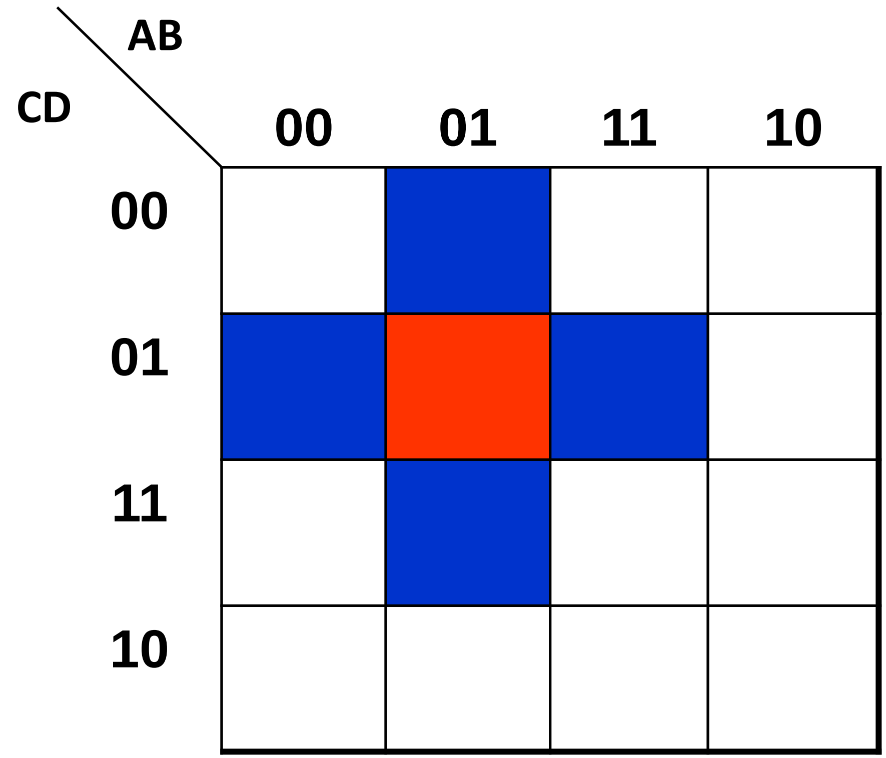
CD \ AB	00	01	11	10
00				
01				
11				
10				

U= 1

Dans un tableau de karnaugh , chaque case possède un certain nombre de cases adjacentes.



Les trois cases bleues sont des cases adjacentes à la case rouge



Passage de la table de vérité à la table de Karnaugh

- Pour chaque combinaisons qui représente un min terme lui correspond une case dans le tableau qui doit être mise à 1 .

- Pour chaque combinaisons qui représente un max terme lui correspond une case dans le tableau qui doit être mise à 0 .

- Lorsque on remplis le tableau , on doit soit prendre les min terme ou les max terme

Exemple :

A	B	C		S
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

C \ AB	00	01	11	10
0			1	
1		1	1	1

Passage de la forme canonique à la table de Karnaugh

- Si la fonction logique est donnée sous la première forme canonique (disjonctive), alors sa représentation est directe : pour chaque terme lui correspond une seule case qui doit être mise à 1.

- Si la fonction logique est donnée sous la deuxième forme canonique (conjonctive), alors sa représentation est directe : pour chaque terme lui correspond une seule case qui doit être mise à 0 .

Exemple

$$F1(A,B,C) = \sum (1,2,5,7)$$

C \ AB	00	01	11	10
0		1		
1	1		1	1

$$F2(A,B,C) = \prod (0,2,3,6)$$

C \ AB	00	01	11	10
0	0	0	0	
1		0		

Méthode de simplification (Exemple : 3 variables)

- L’idée de base est d’essayer de regrouper (faire des regroupements) les cases adjacentes qui comportent des 1 (rassembler les termes adjacents).
- Essayer de faire des regroupements avec le maximum de cases (16,8,4 ou 2)
- Dans notre exemple on peut faire uniquement des regroupements de 2 cases .

C \ AB	00	01	11	10
0			1	
1		1	1	1

$$AB\overline{C} + ABC = AB$$

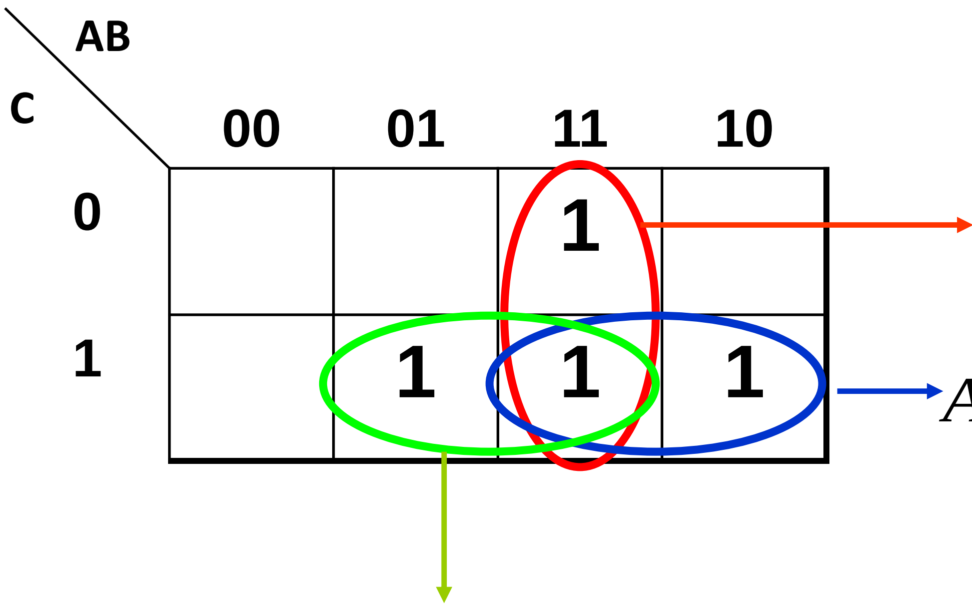
- Puisque il existent encore des cases qui sont en dehors d’un regroupement on refait la même procédure : former des regroupements.
- Une case peut appartenir à plusieurs regroupements

C \ AB	00	01	11	10
0			1	
1		1	1	1

$$AB\overline{C} + ABC = AB$$

$$ABC + A\overline{B}C = AC$$

•On s’arrête lorsque il y a plus de 1 en dehors des regroupements
•La fonction final est égale à la réunion (somme) des termes après simplification.



$$AB\overline{C} + ABC = AB$$

$$ABC + A\overline{B}C = AC$$

$$\overline{A}BC + ABC = BC$$

$$F(A,B,C) = AB + AC + BC$$

Donc , en résumé pour simplifier une fonction par la table de karnaugh il faut suivre les étapes suivantes :

1. Remplir le tableau à partir de la table de vérité ou à partir de la forme canonique.
2. Faire des regroupements : des regroupements de 16,8,4,2,1 cases (Les même termes peuvent participer à plusieurs regroupements) .
3. Dans un regroupement :
   - Qui contient un seule terme on peut pas éliminer de variables.
   - Qui contient deux termes on peut éliminer une variable (celle qui change d'état).
   - Qui contient 4 termes on peut éliminer 2 variables.
   - Qui contient 8 termes on peut éliminer 3 variables.
   - Qui contient 16 termes on peut éliminer 4 variables.
5. L'expression logique finale est la réunion (la somme) des groupements après simplification et élimination des variables qui changent d'état.

Exemple 1 : 3 variables

C \ AB	00	01	11	10
0			1	
1	1	1	1	1

$$F(A,B,C) = C + AB$$

Exemple 2 : 4 variables

CD \ AB	00	01	11	10
00				1
01	1	1	1	1
11				
10		1		

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

Exemple 3 : 4 variables

CD \ AB	00	01	11	10
00	1			1
01		1	1	1
11				1
10	1			1

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

Exemple 4 : 5 variables

CD \ AB	00	01	11	10
00	1			
01	1		1	
11	1		1	
10	1			

U = 0

CD \ AB	00	01	11	10
00	1			
01	1			1
11	1			1
10	1	1		

U= 1

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

Exercice

Trouver la forme simplifiée des fonctions à partir des deux tableaux ?

C \ AB	00	01	11	10
0		1	1	1
1	1		1	1

CD \ AB	00	01	11	10
00	1		1	1
01				
11				
10	1	1	1	1

Cas d'une fonction non totalement définie

- Examinons l'exemple suivant :

Une serrure de sécurité s'ouvre en fonction de quatre clés A, B, C D. Le fonctionnement de la serrure est définie comme suite :

S(A,B,C,D)= 1 si au moins deux clés sont utilisées

S(A,B,C,D)= 0 sinon

Les clés A et D ne peuvent pas être utilisées en même temps.

- On remarque que si la clé A et D sont utilisées en même temps l'état du système n'est pas déterminé.

- Ces cas sont appelés cas impossibles ou interdites → comment représenter ces cas dans la table de vérité ?.

- Pour les cas impossibles ou interdites il faut mettre un X dans la T.V .
- Les cas impossibles sont représentées aussi par des X dans la table de karnaugh

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

A	B	C	D		S
0	0	0	0		0
0	0	0	1		0
0	0	1	0		0
0	0	1	1		1
0	1	0	0		0
0	1	0	1		1
0	1	1	0		1
0	1	1	1		1
1	0	0	0		0
1	0	0	1		X
1	0	1	0		1
1	0	1	1		X
1	1	0	0		1
1	1	0	1		X
1	1	1	0		1
1	1	1	1		X

- Il est possible d’utiliser les X dans des regroupements :
  - Soit les prendre comme étant des 1
  - Ou les prendre comme étant des 0
- Il ne faut pas former des regroupement qui contient uniquement des X

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD$$

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD + BD$$

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD + BD + AC$$

CD \ AB	00	01	11	10
00			1	
01		1	X	X
11	1	1	X	X
10		1	1	1

$$AB + CD + BD + AC + BC$$

Exercices

Simplifier.

		ab			
		"00"	"01"	11	10
cd	"00"		1		
	"01"		1		
	11	1	1		1
	10	1	1		1

		ab			
		"00"	"01"	11	10
cd	"00"	1			1
	"01"		1		
	11	1	1	1	1
	10	1	1	1	1

		ab			
		"00"	"01"	11	10
cd	"00"			x	
	"01"	1	1	x	1
	11	1	1		
	10		x		

		ab			
		"00"	"01"	11	10
cd	"00"		1		1
	"01"	1	x		1
	11	x	1	x	1
	10		x		

Exercices

cd \ ab	"00"	"01"	11	10
"00"		1		
"01"		1		
11	1	1		1
10	1	1		1

/a . b

/b . c

S = /a . b + /b . c

cd \ ab	"00"	"01"	11	10
"00"	1			1
"01"		1		
11	1	1	1	1
10	1	1	1	1

/a . b . d

/b . /d

c

S = /a . b . d + /b . /d +c

cd \ ab	"00"	"01"	11	10
"00"			x	
"01"	1	1	x	1
11	1	1		
10		x		

/a . d

/c . d

S = /a . d + /c . d = d . (/a . /c)

cd \ ab	"00"	"01"	11	10
"00"		1		1
"01"	1	x		1
11	x	1	x	1
10		x		

a . /b . /c

/a . b

/b . d

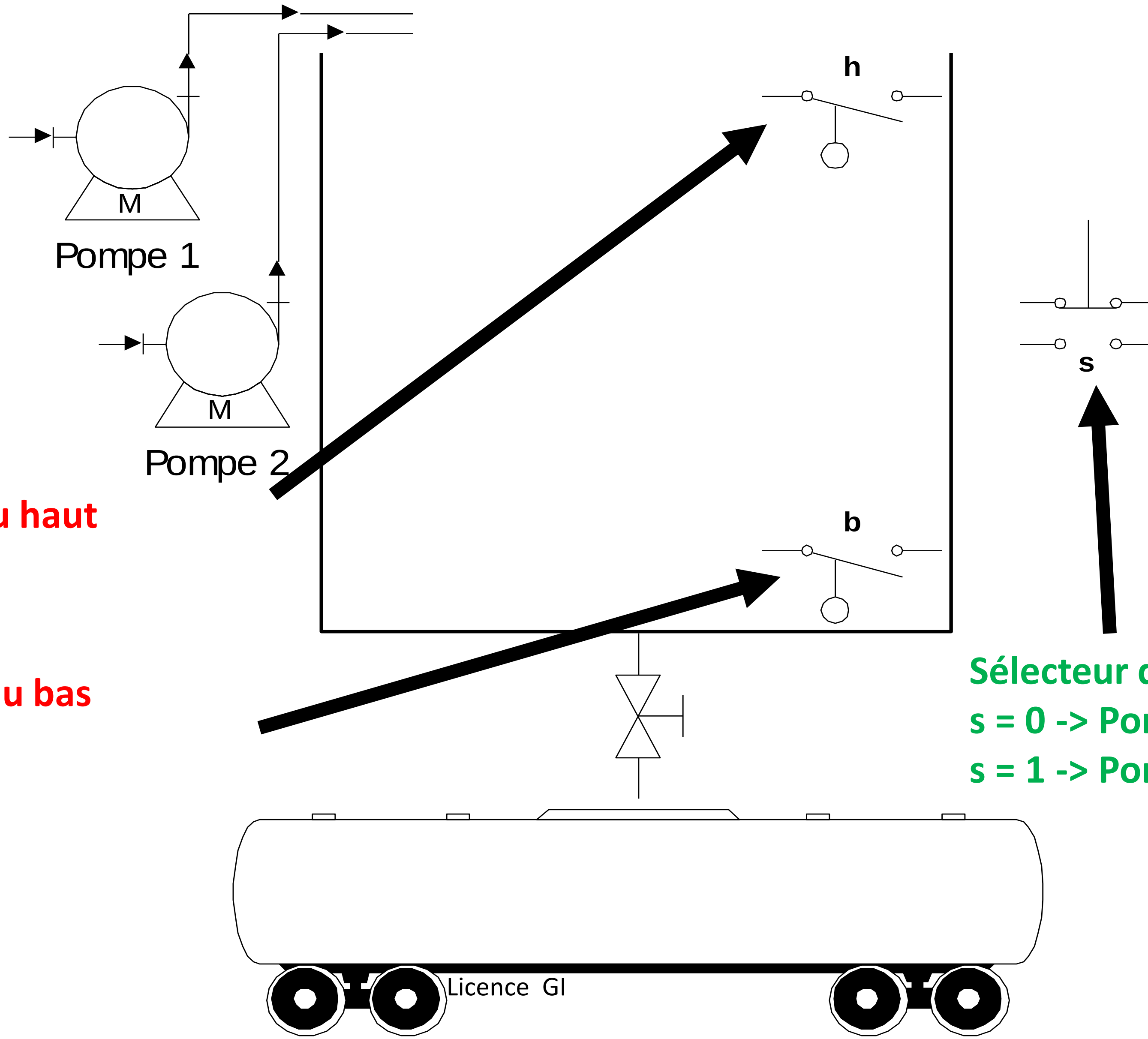
S = a . /b . /c + /a . b + /b . d = /a . b + /b . (a . /c + d)

Les états indifférents

- Ils sont représentés par des X

- En sortie, ils correspondent à des combinaisons d'entrées pour lesquelles la sortie n'a pas été définie.
  - Ex.: Un réservoir ne peut être à la fois vide et plein.

Contrôle de niveau d'un réservoir



Capteur de niveau haut
h = 1 -> plein

Capteur de niveau bas
b = 0 -> vide

Sélecteur de pompe
s = 0 -> Pompe 1
s = 1 -> Pompe 2

Contrôle de niveau

- Si réservoir plein: Aucune pompe en marche;
- Si réservoir vide: Les 2 pompes en marche;
- Si réservoir ni vide, ni plein: Faire fonctionner la pompe sélectionnée par le sélecteur « s ».

Contrôle de niveau

b = 0 vide
h = 1 plein
s = 0 ->P1
s = 1 ->P2

- Table de vérité:

Entrées			Sorties	
h	b	s	P_1	P_2
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	X	X
1	0	1	X	X
1	1	0	0	0
1	1	1	0	0

Réservoir vide

Réservoir à 1/2

Réservoir plein et vide ?!?

Réservoir plein

Contrôle de niveau

- Tables de Karnaugh:

$P_2 = /b + /h.s$

h \ bs	00	01	11	10
0	1_{0}	1_{1}	1_{3}	0_{2}
1	X_{4}	X_{5}	0_{7}	0_{6}

$P_1 = /b + /h./s$

h \ bs	00	01	11	10
0	1_{0}	1_{1}	0_{3}	1_{2}
1	X_{4}	X_{5}	0_{7}	0_{6}

Contrôle de niveau

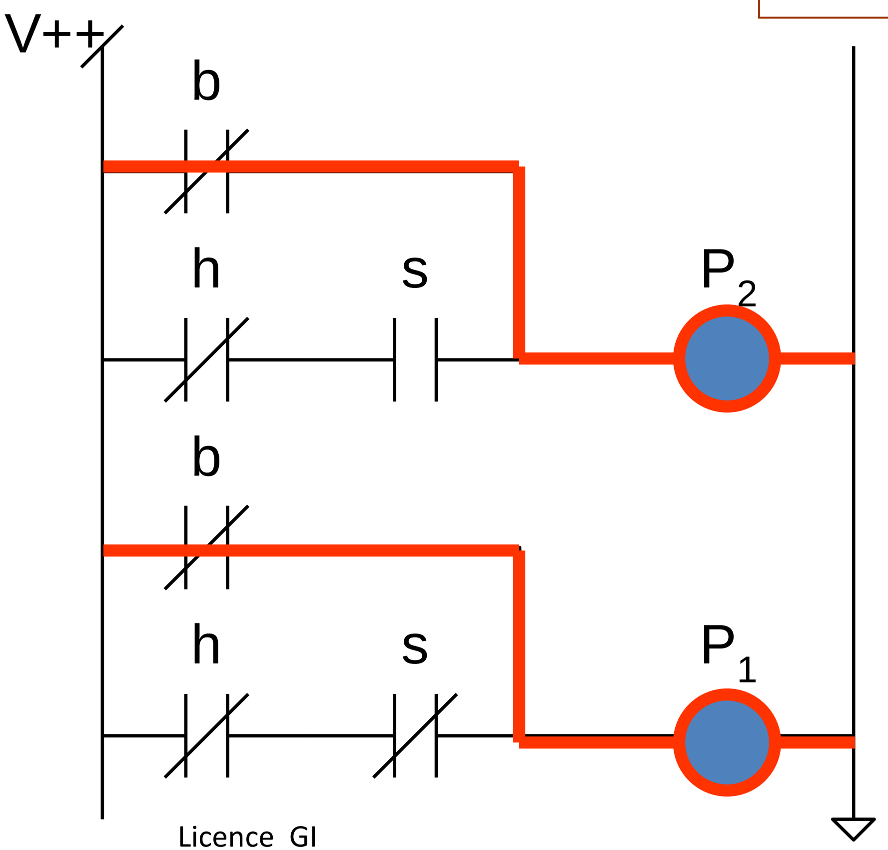
P_2 = /b + /h.s

P_1 = /b + /h./s

- Diagramme échelle:

Seul risque:
- si le capteur b est en panne (b=0) alors que le réservoir est plein...

Les deux pompes seront en marche !!!



Contrôle de niveau

- Si on considère les X comme des 0.

P_2 = /b./h + /h.s

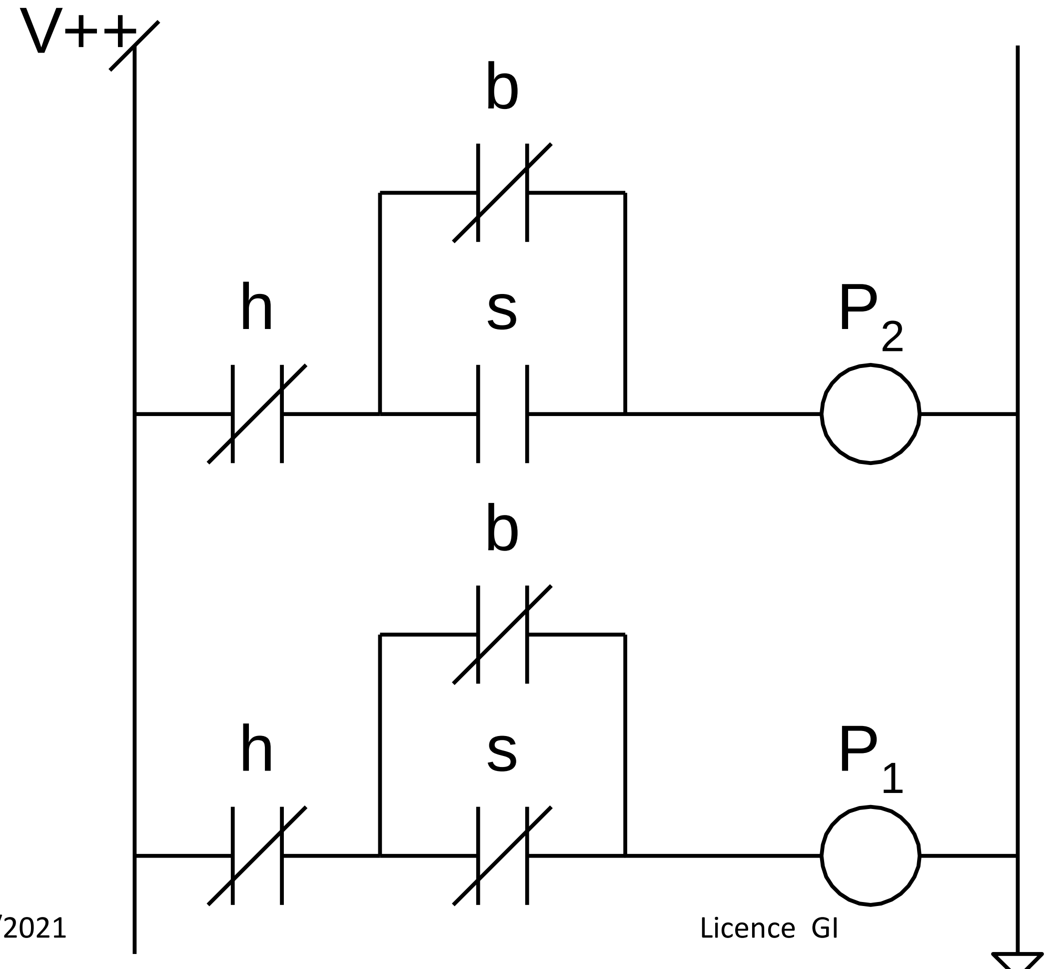
h \ bs	00	01	11	10
0	1_{0}	1_{1}	1_{3}	0_{2}
1	0_{4}	0_{5}	0_{7}	0_{6}

P_1 = /b./h + /h./s

h \ bs	00	01	11	10
0	1_{0}	1_{1}	0_{3}	1_{2}
1	0_{4}	0_{5}	0_{7}	0_{6}

Contrôle de niveau

- Diagramme échelle (sécuritaire):



$P_2 = /b./h + /h.s$

$P_1 = /b./h + /h./s$

Conclusion de l'exemple

- Les « X » peuvent êtres utilisés dans des groupes de 1 pour en augmenter la taille.
  - Cela implique des équations plus simples;

- Du point de vue sécurité, il peut s 'avérer nécessaire de considérer les « X » comme des « 0 ».

Les circuits combinatoires

Objectifs

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

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

Les Circuits combinatoires

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

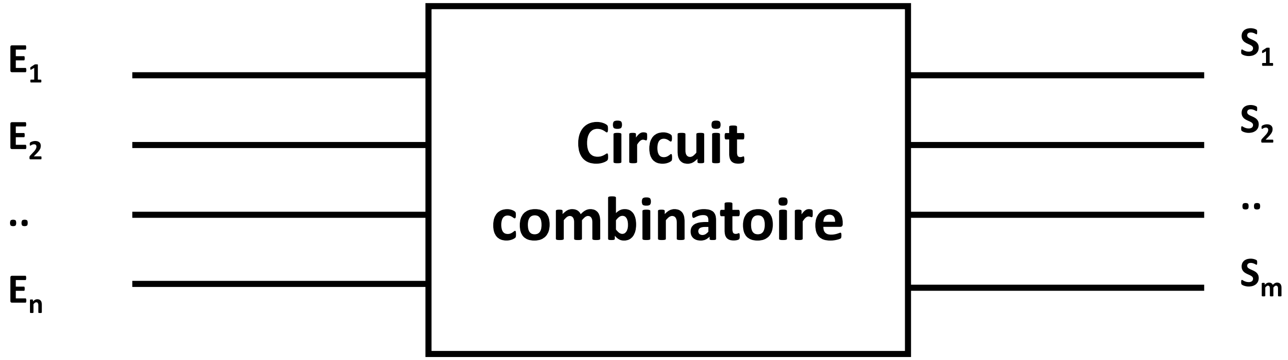


Schéma Bloc

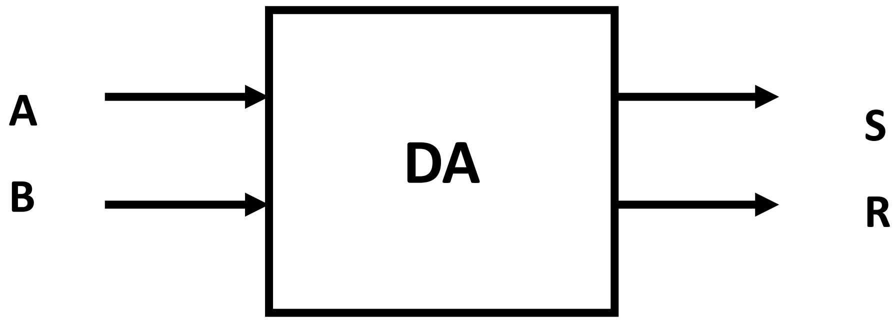
- C'est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits plus complexes.

Exemple de Circuits combinatoires

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

Demi Additionneur

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



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

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

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

- La table de vérité associée :

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

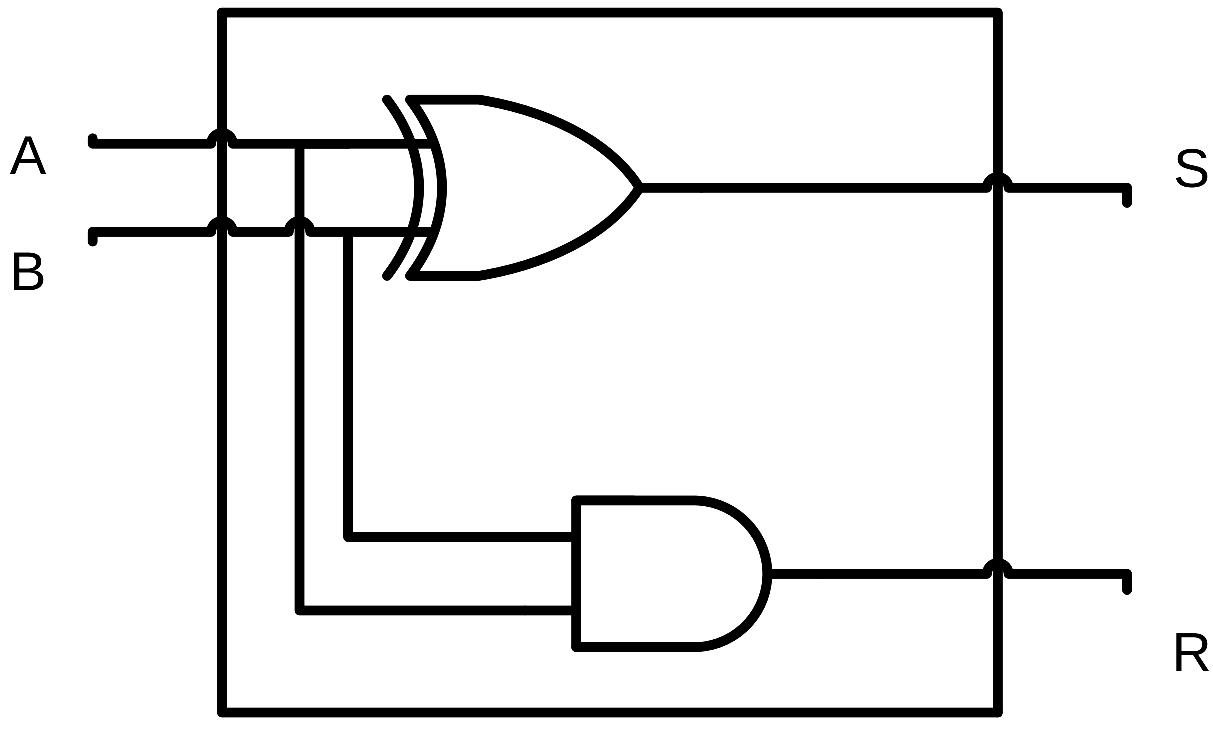
De la table de vérité on trouve :

$$R = A.B$$

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

$$R = A.B$$

$$S = A \oplus B$$



L'additionneur complet

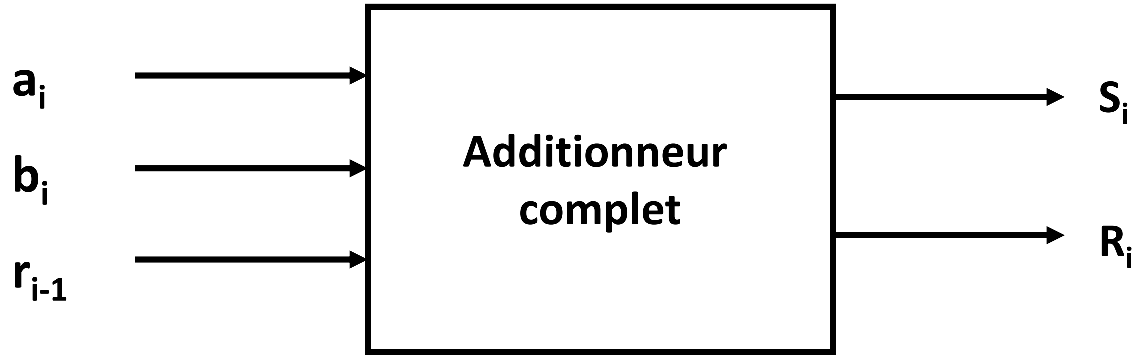
- En binaire lorsque on fait une addition il faut tenir en compte de la retenue entrante.

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

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

Additionneur complet 1 bit

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



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

Table de vérité d'un additionneur complet sur 1 bit

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

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

Si on veut simplifier les équations on obtient :

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

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

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

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

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

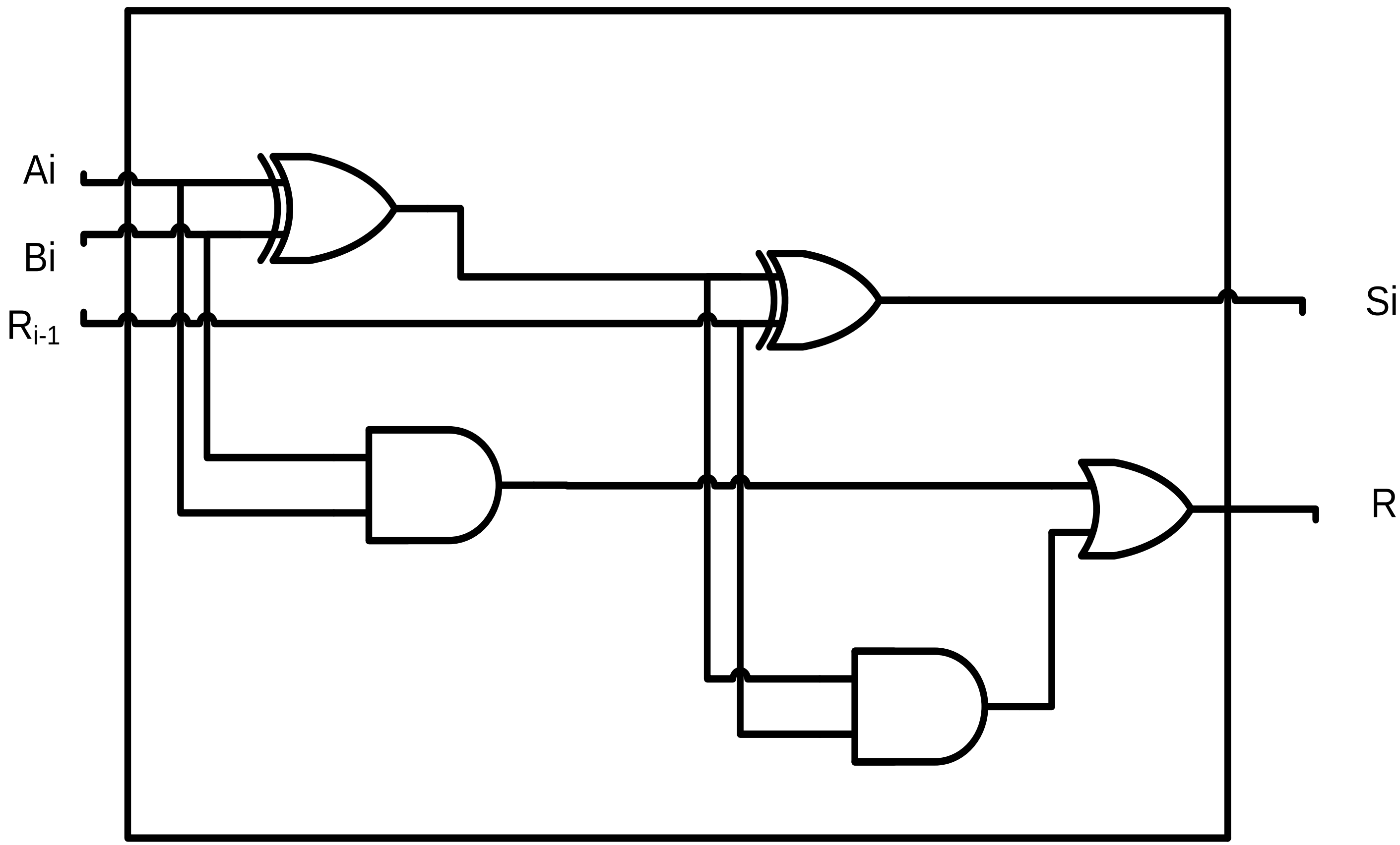
$$R_i = R_{i-1}.(\overline{A_i}.B_i + A_i.\overline{B}_i) + A_iB_i(\overline{R}_{i-1} +_i R_{i-1})$$

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

Schéma d'un additionneur complet

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

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



En utilisant des Demi Additionneurs

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

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

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

On obtient :

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

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

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

On obtient :

$$R_i = Y + T$$

$$S_i = Z$$

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

$$X = A_i \oplus B_i$$

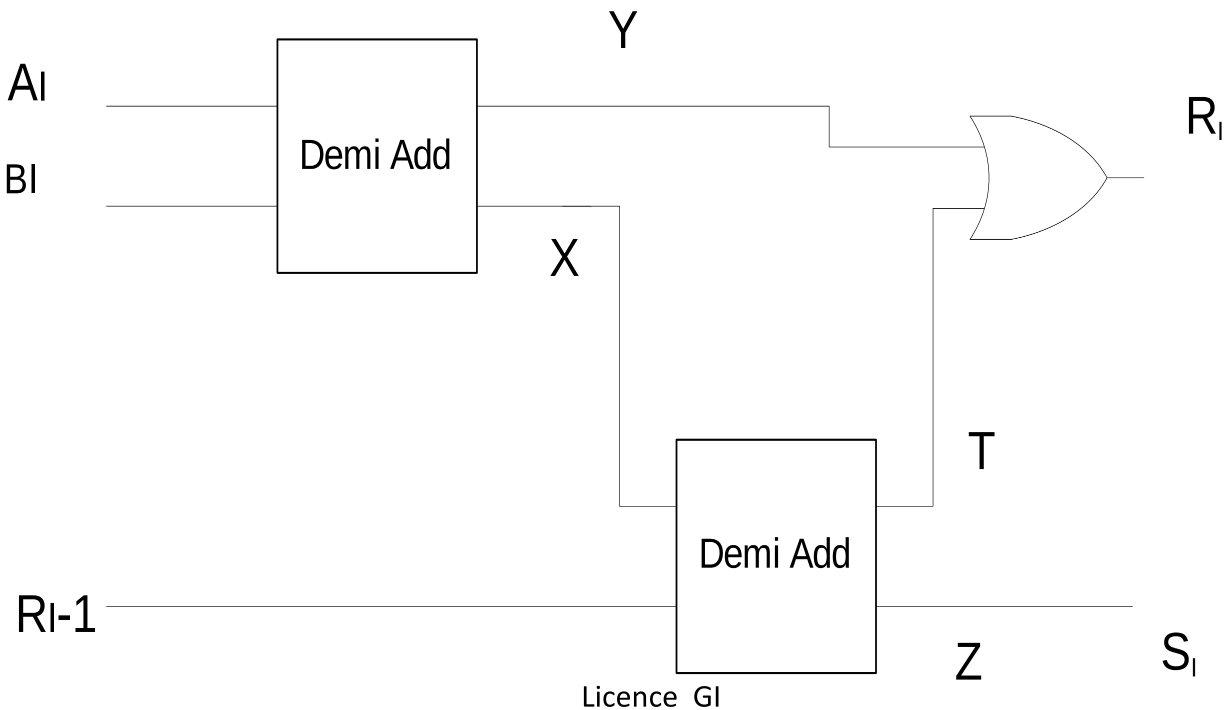
$$Y = A_i B_i$$

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

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

$$R_i = Y + T$$

$$S_i = Z$$



Additionneur sur 4 bits

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

  En plus il tient en compte de la retenu entrante

- En sortie on va avoir le résultat sur 4 bits ainsi que la retenu (5 bits en sortie)

- Donc au total le circuit possède 9 entrées et 5 sorties.

- Avec 9 entrées on a $2^9=512$ combinaisons !!!!!! Comment faire pour représenter la table de vérité ?????

- Il faut trouver une solution plus facile et plus efficace pour concevoir ce circuit ?

•Lorsque on fait l’addition en binaire , on additionne bit par bit en commençant à partir du poids fiable et à chaque fois on propage la retenue sortante au bit du rang supérieur.

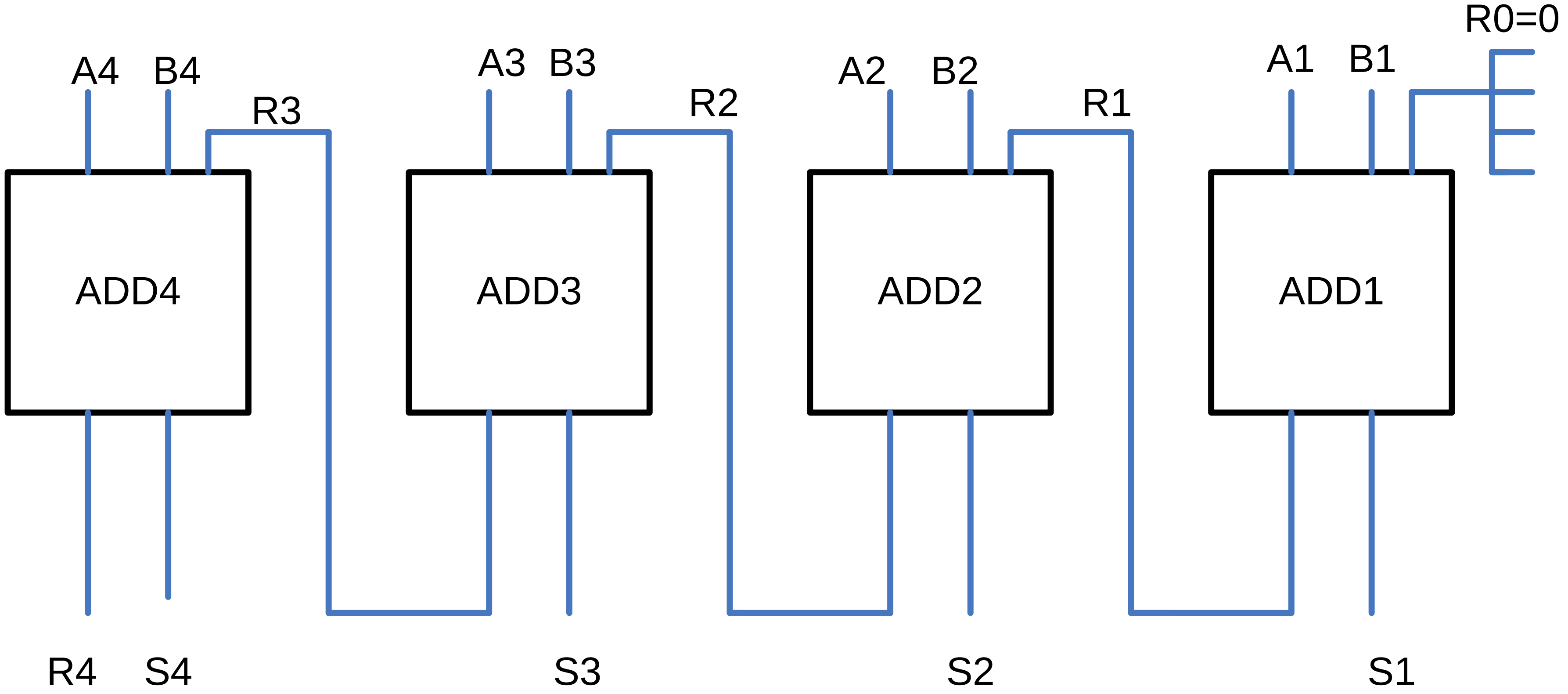
L’addition sur un bit peut se faire par un additionneur complet sur 1 bits.

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

$$\boxed{r4 \quad s_4 \quad s_3 \quad s_2 \quad s_1}$$

Résultat final

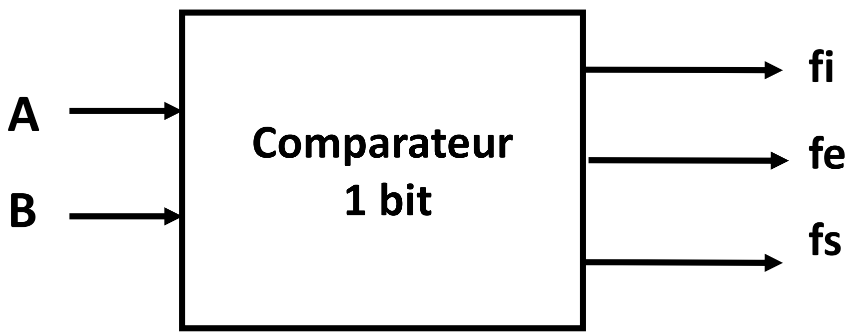
Additionneur 4 bits (schéma)



Le Comparateur

- C'est un circuit combinatoire qui permet de comparer entre deux nombres binaire A et B.
- Il possède 2 entrées :
  - A : sur un bit
  - B : sur un bit

- Il possède 3 sorties
  - fe : égalité (A=B)
  - fi : inférieur (A < B)
  - fs : supérieur (A > B)



Comparateur sur un bit

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

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

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

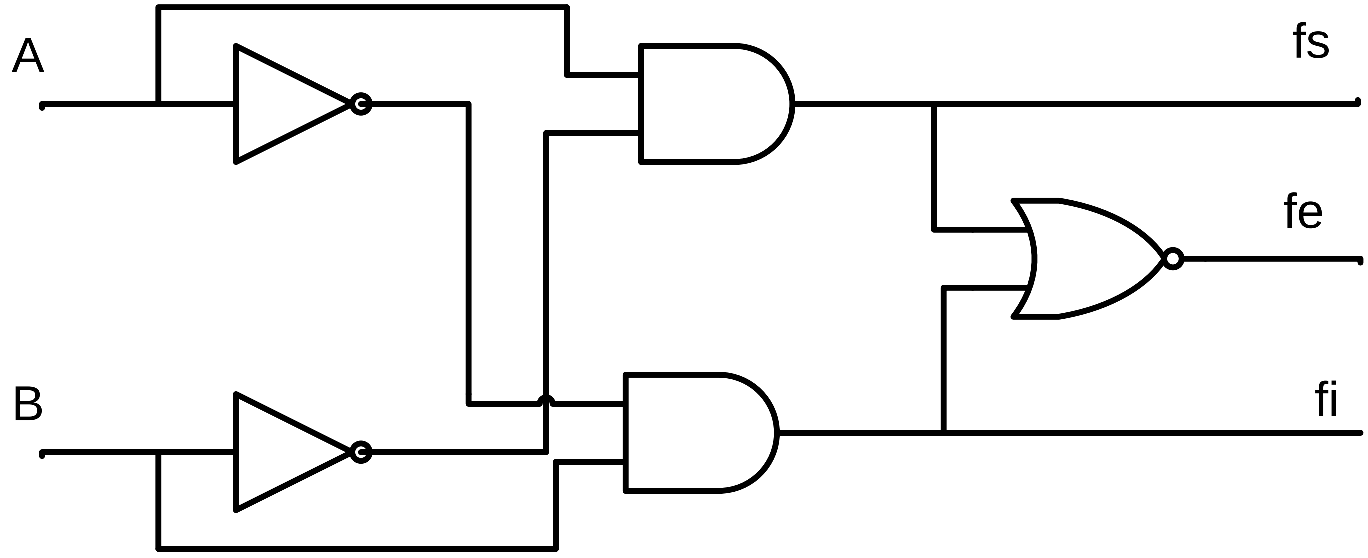
$$fe = \overline{A}\,\overline{B} + AB = \overline{A \oplus B} = \overline{fs + fi}$$

Schéma d'un comparateur dur un bit

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

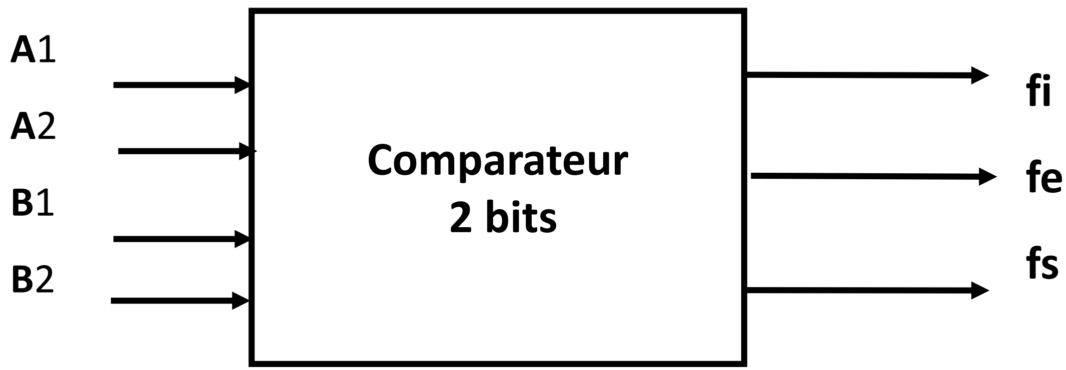
$$fi = \overline{A}B$$

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



Comparateur 2 bits

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



1. A=B si

A2=B2 et A1=B1

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

2. A>B si

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

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

3. A<B si

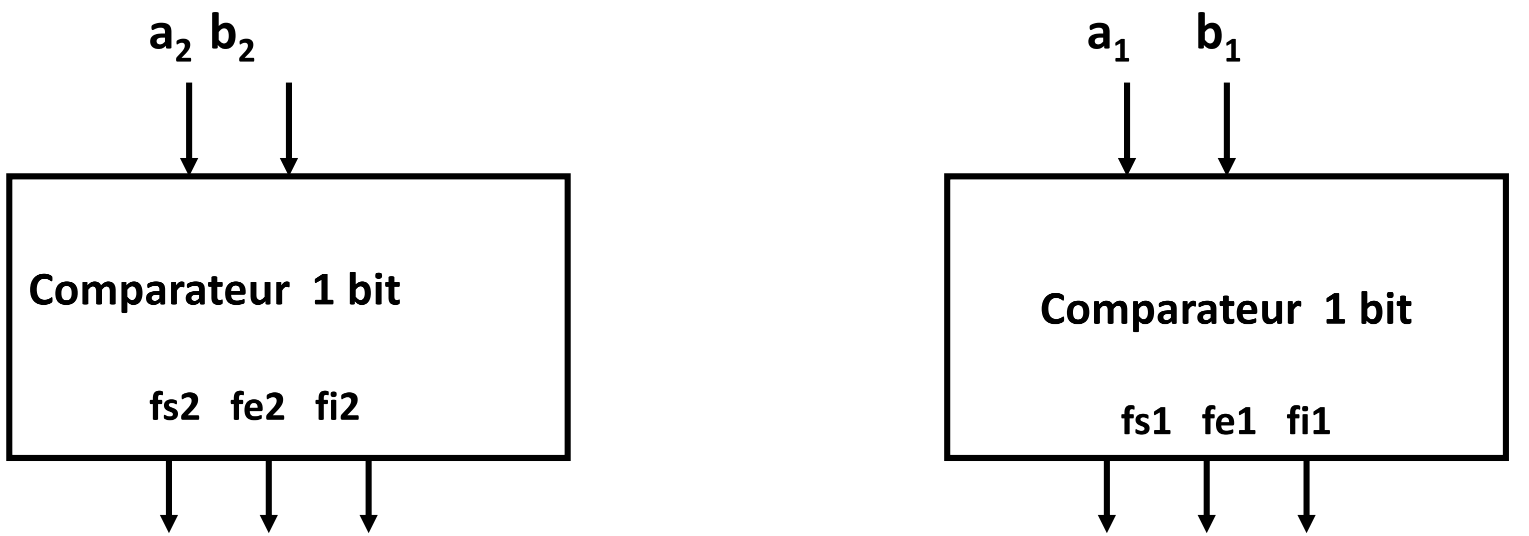
A2 < B2 ou (A2=B2 et A1<B1)

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

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

comparateur 2 bits avec des comparateurs 1 bit

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



1. A=B si

A2=B2 et A1=B1

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

2. A>B si

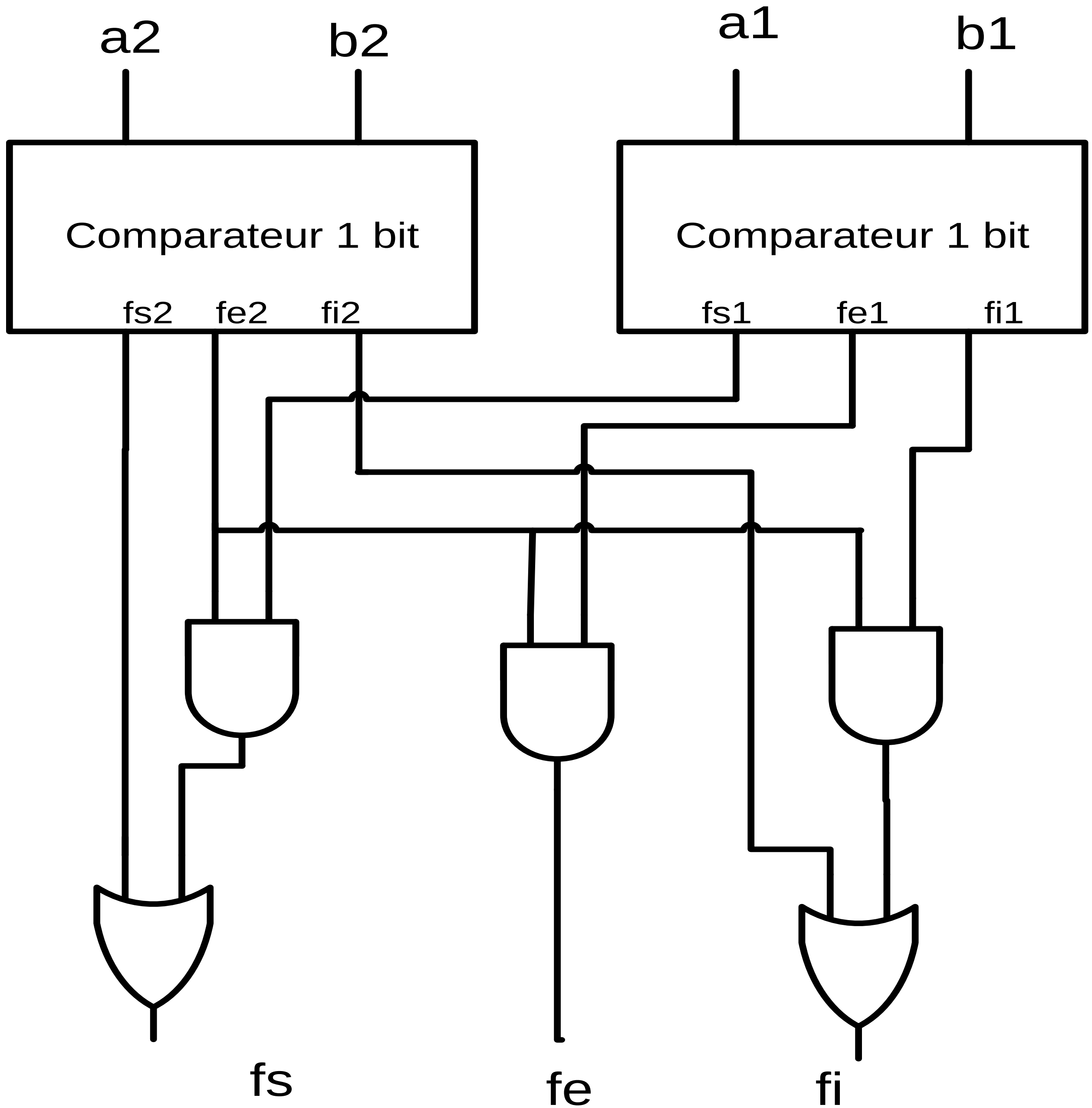
A2 > B2 ou (A2=B2 et A1>B1)

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

3. A<B si

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

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


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

Comparateur avec des entrées de mise en cascade

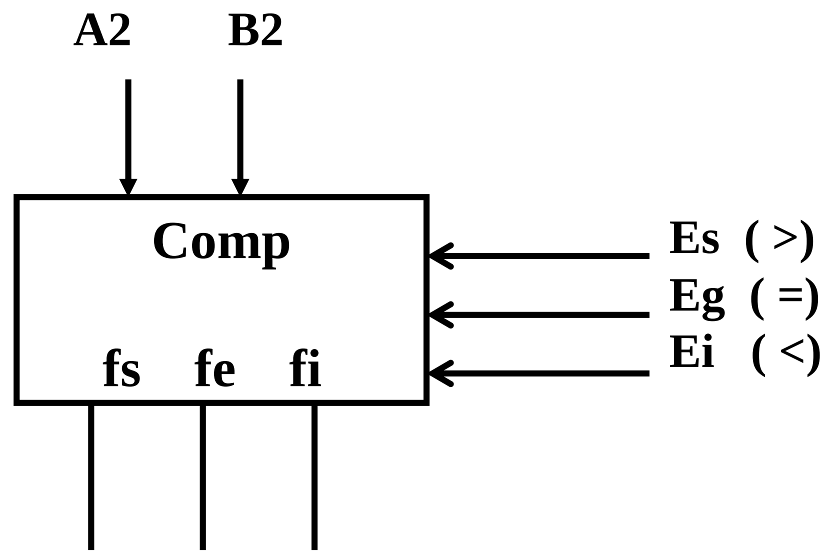
- On remarque que :
  - Si A2 >B2 alors **A > B**
  - Si A2<B2 alors **A < B**

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

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

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

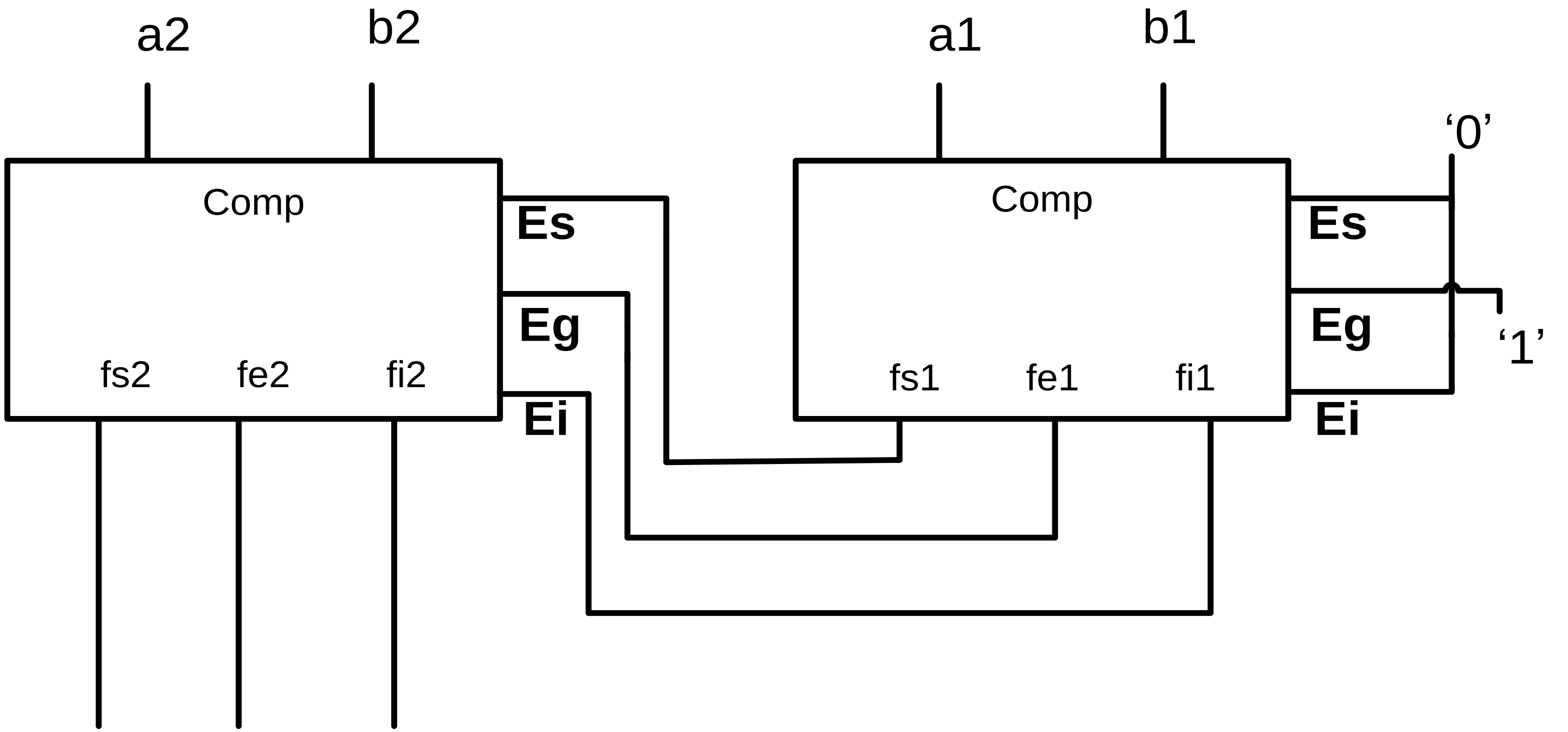
A2	B2	Es	Eg	Ei		fs	fe	fi
A2>B2		X	X	X		1	0	0
A2<B2		X	X	X		0	0	1
A2=B2		1	0	0		1	0	0
		0	1	0		0	1	0
		0	0	1		0	0	1



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

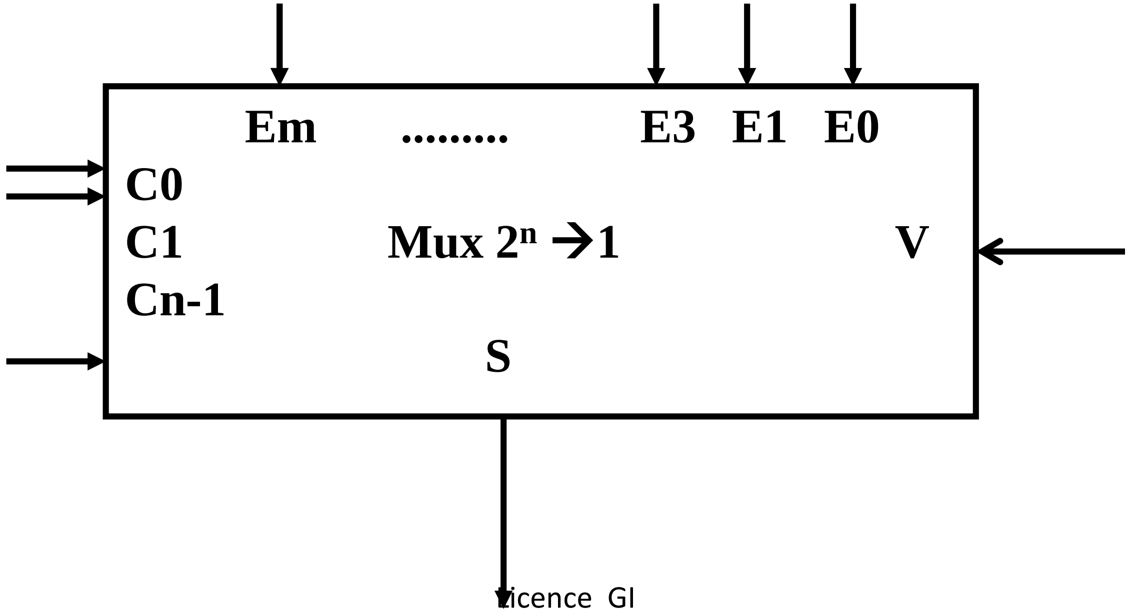
fi= (A2<B2) ou (A2=B2).Ei

fe=(A2=B2).Eg


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

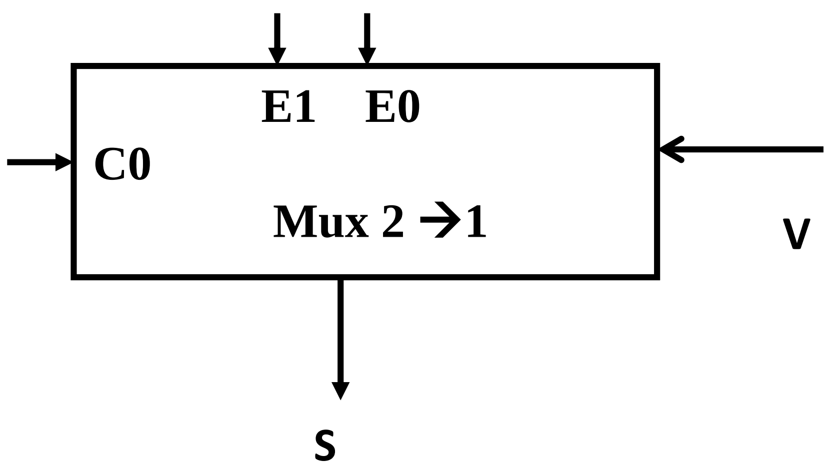
Le Multiplexeur

- Un multiplexeur est un circuit combinatoire qui permet de sélectionner une information (1 bit) parmi 2^n valeurs en entrée.
- Il possède :
  - 2^n entrées d'information
  - Une seule sortie
  - N entrées de sélection (commandes)



Multiplexeur 2 →1

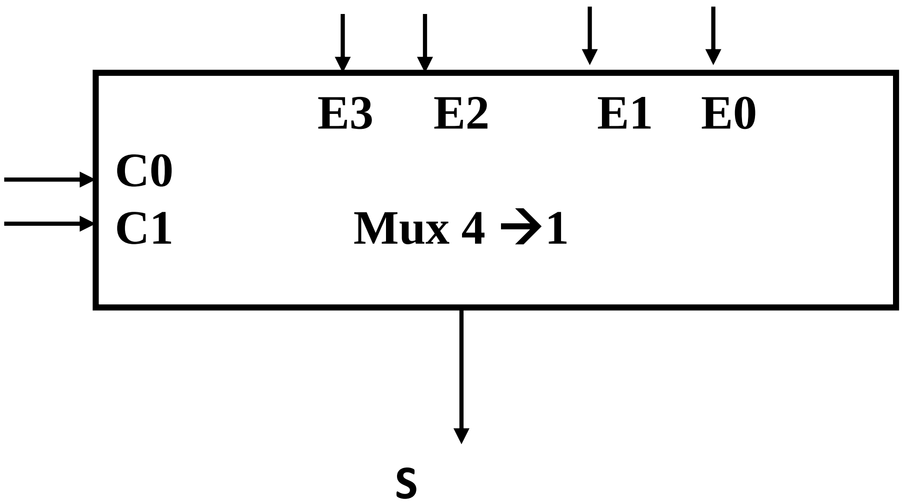
V	C_0		S
0	X		0
1	0		E0
1	1		E1



$$S = V.(\overline{C_0}.E0 + C_0.E1)$$

Multiplexeur 4 →1

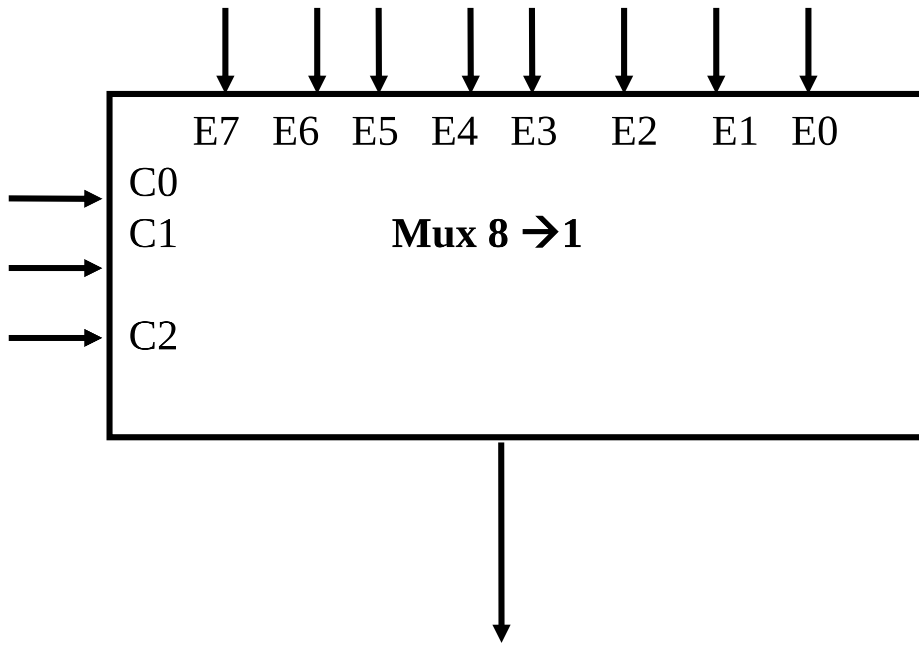
C1	C0		S
0	0		E0
0	1		E1
1	0		E2
1	1		E3



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

Multiplexeur 8→1

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



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

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

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

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

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

Réalisation de la fonction de la somme

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

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

On pose :

$C2=A_i$

$C1=B_i$

$C0=R_{i-1}$

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

Réalisation de la fonction de la retenue

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

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

On pose :

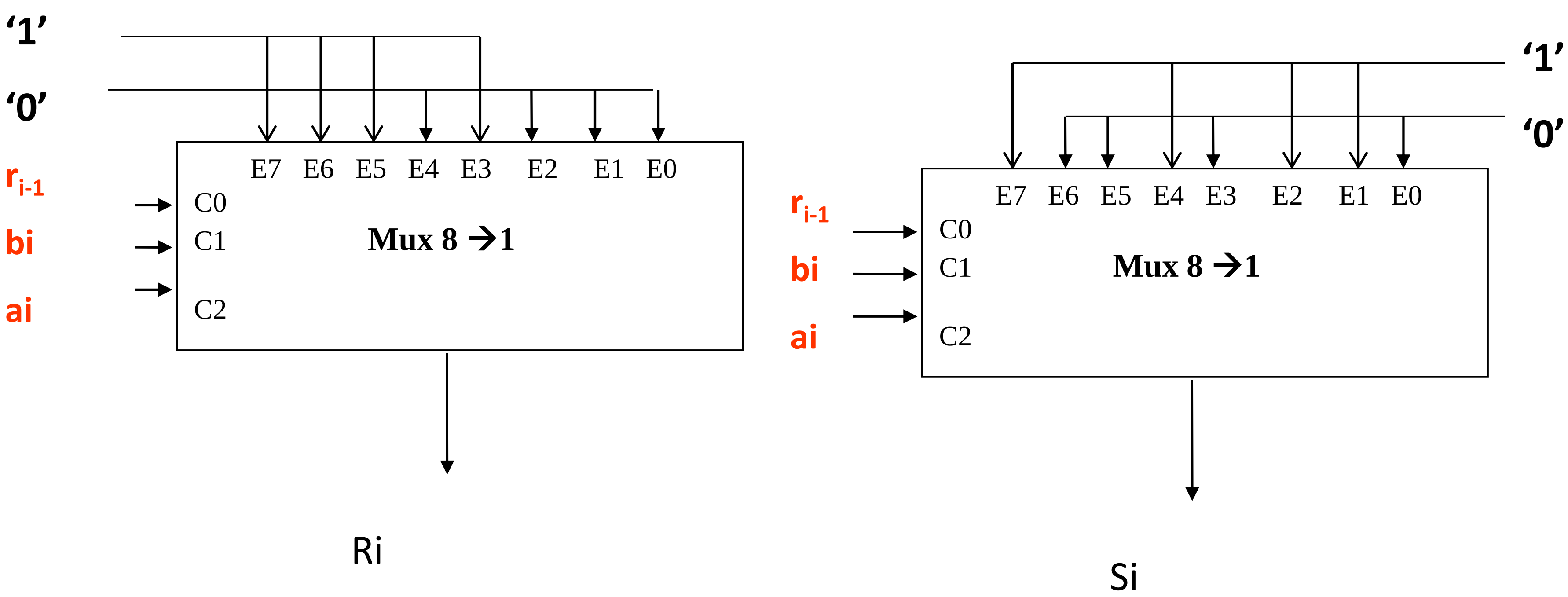
$C2=A_i$

$C1=B_i$

$C0=R_{i-1}$

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

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

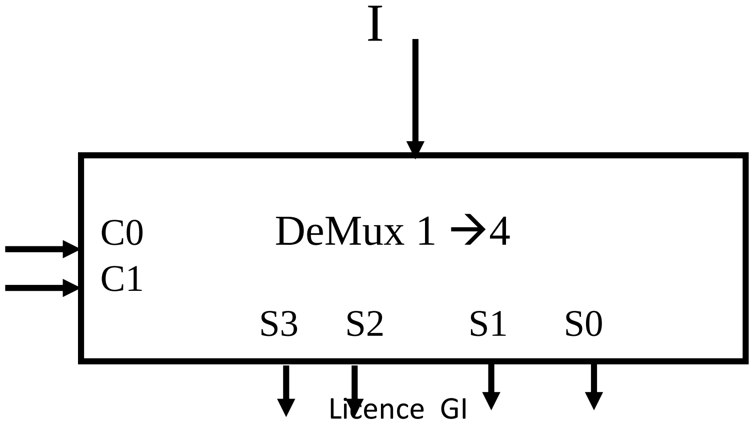


Exercice

- Réaliser le circuit qui permet de trouver le maximum entre deux nombres A et B sur un Bit en utilisant le minimum de portes logiques et de circuits combinatoires?

Démultiplexeurs

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



Démultiplexeur 1→4

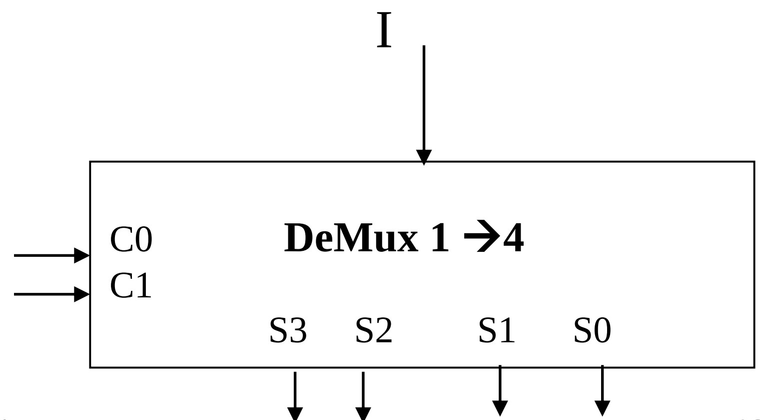
C1	C0		S3	S2	S1	S0
0	0		0	0	0	i
0	1		0	0	i	0
1	0		0	i	0	0
1	1		i	0	0	0

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

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

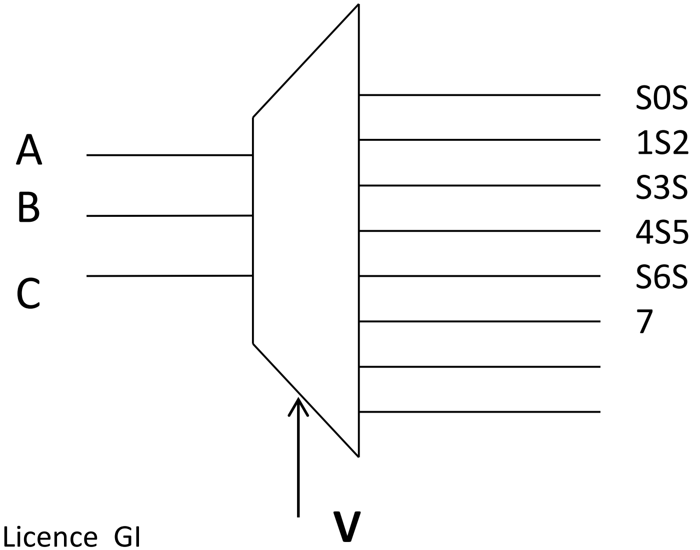
$$S2 = C1.\overline{C0}.(I)$$

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



Le décodeur binaire

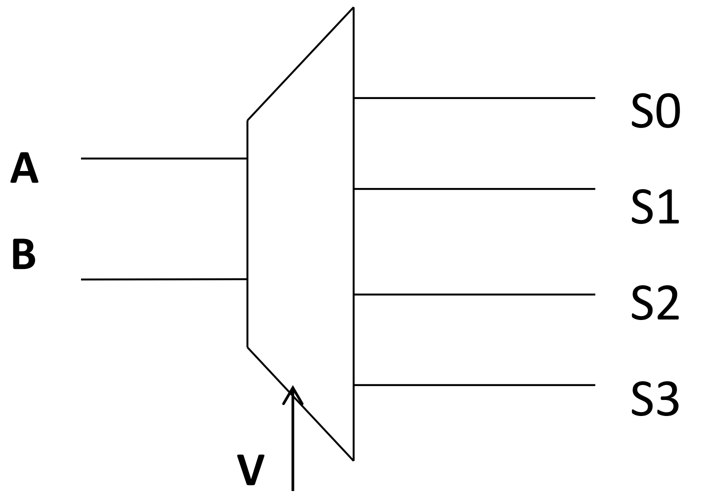
- C'est un circuit combinatoire qui est constitué de :
  - N : entrées de données
  - 2^n sorties
  - Pour chaque combinaison en entrée une seule sortie est active à la fois



Un décodeur 3→8

Décodeur 2→4

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



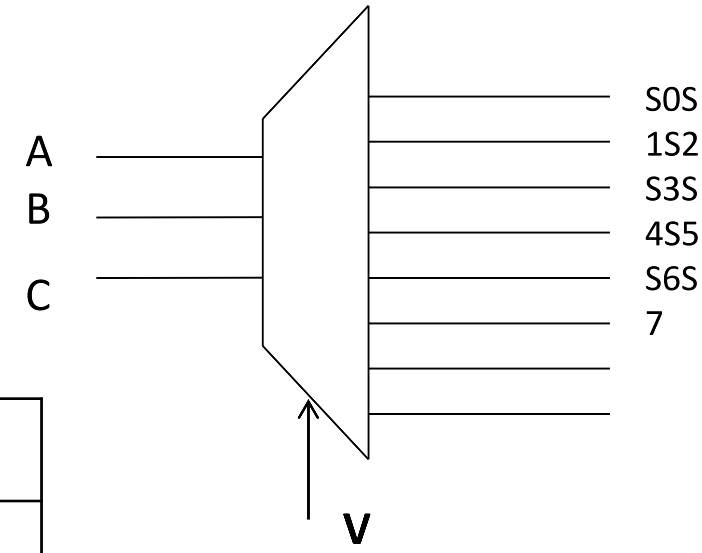
$$S_0 = (\overline{A}.\overline{B}).V$$

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

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

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

Décodeur 3→8



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

$$S_0 = \overline{A}.\overline{B}.\overline{C}$$
$$S_1 = \overline{A}.\overline{B}.C$$
$$S_2 = \overline{A}.B.\overline{C}$$
$$S_3 = \overline{A}.B.C$$
$$S_4 = A.\overline{B}.\overline{C}$$
$$S_5 = A.\overline{B}.C$$
$$S_6 = A.B.\overline{C}$$
$$S_7 = A.B.C$$

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

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

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

$$R_i = \overline{A_i}B_iR_{i-1} + A_i\overline{B_i}R_{i-1} + A_iB_i\overline{R_{i-1}}. + A_iB_iR_{i-1}$$

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

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

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

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

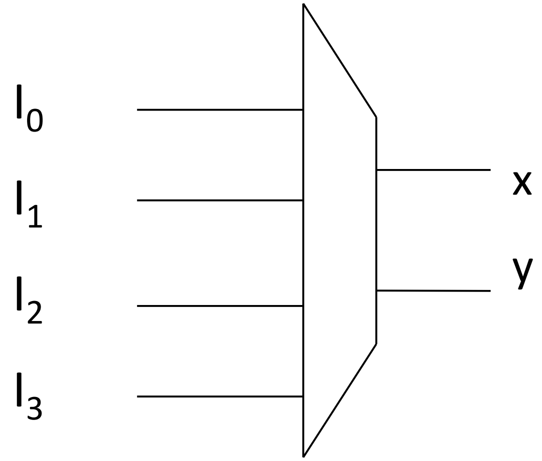
$$R_i = S3 + S5 + S6 + S7$$

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

L'encodeur binaire

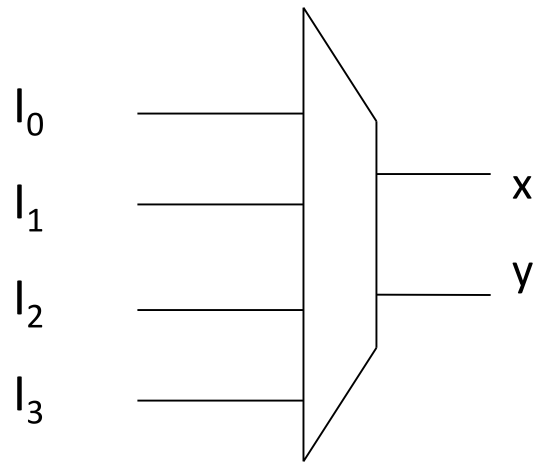
- Il joue le rôle inverse d'un décodeur
  - Il possède 2^n entrées
  - N sortie
  - Pour chaque combinaison en entrée on va avoir sont numéro (en binaire) à la sortie.

Encodeur 4→2



L'encodeur binaire (4→2)

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

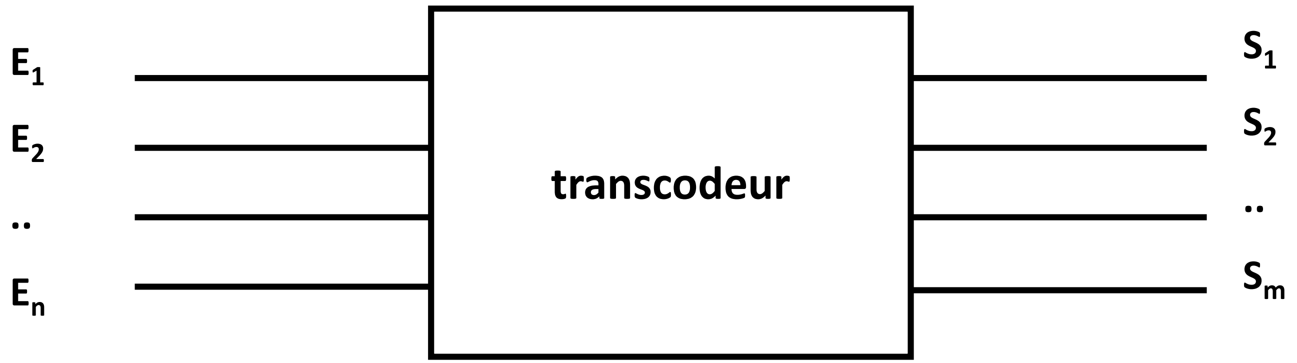


$$X = \overline{I0}.\overline{I1}.(I2 + I3)$$

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

Le transcodeur

- C'est un circuit combinatoire qui permet de transformer un code X (sur n bits) en entrée en un code Y (sur m bits) en sortie.



Exemple : Transcodeur BCD/EXESS3

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

Les circuits séquentiels

- Introduction
- Notion d'horloge (système synchrone et système asynchrone)
- Les bascules
  - T
  - RS
  - RST
  - D et D latch
  - JK
- Les registres
- Les compteurs/decompteurs

Introduction

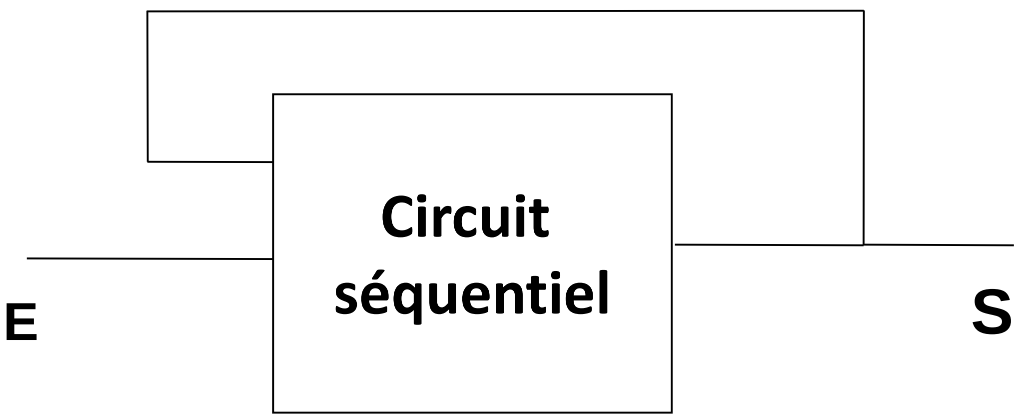
- Un circuit combinatoire est un circuit numérique dont les sorties dépendent uniquement des entrées:

$$S = f(E)$$

- L'état du système ne dépend pas de l'état interne du système.
- Pas de mémoration de l'état du système.

Les circuits séquentiels

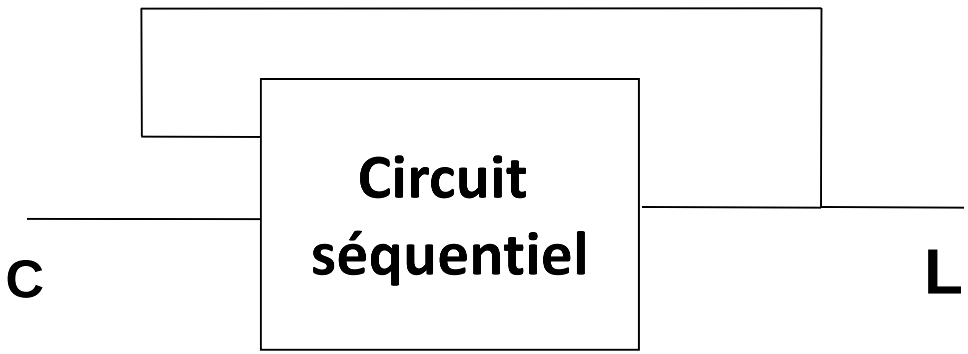
- Un circuit séquentiel est un circuit numérique (logique) dont **l'état** à l'instant **t+1** est une fonction **des entrées** en même instant **t+1** et de **l'état précédente du système** (l'instant t)



$$S_{t+1} = f(E, S_t)$$

$$S^+ = f(E, S)$$

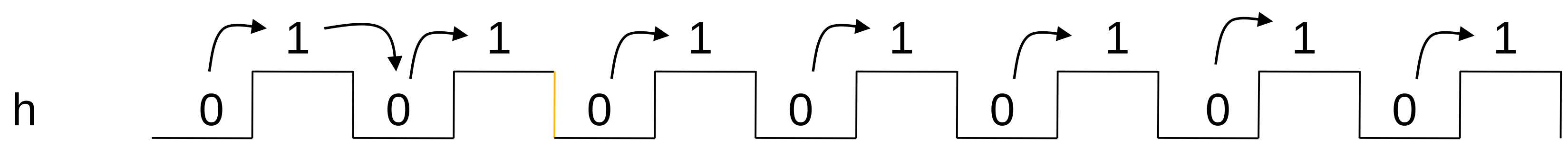
Exemple d'un circuit séquentiel

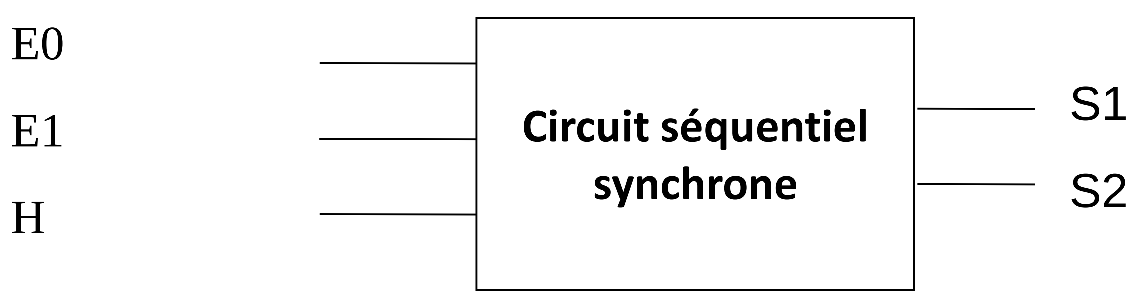


C	L		L^+	
0	X		L	Mémoire
1	0		1	basculement
1	1		0	basculement

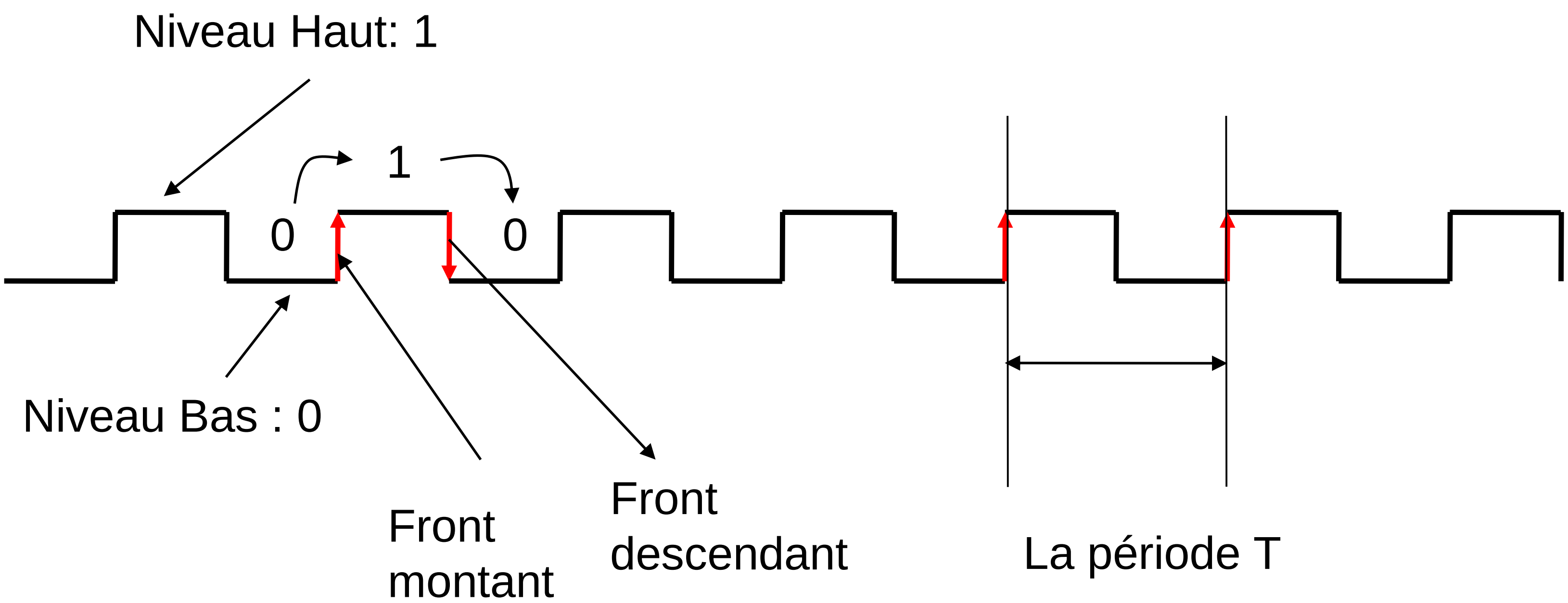
Système synchrone (Notion de l'horloge)

- Une horloge est une variable logique qui passe successivement de 0 à 1 et de 1 à 0 d'une façon périodique.
- Cette variable est utilisée souvent comme une entrée des circuits séquentiels → le circuit est dit synchrone.
- L'horloge est notée par h ou ck (clock).





L’horloge

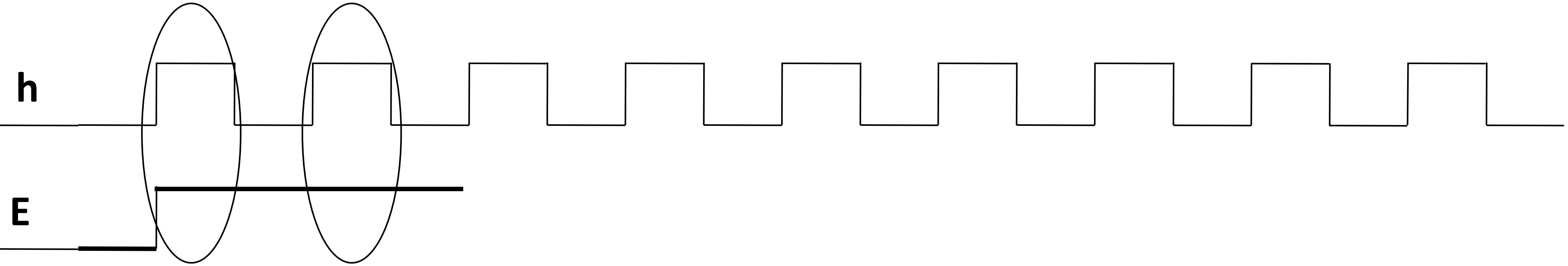


La période T est en seconde

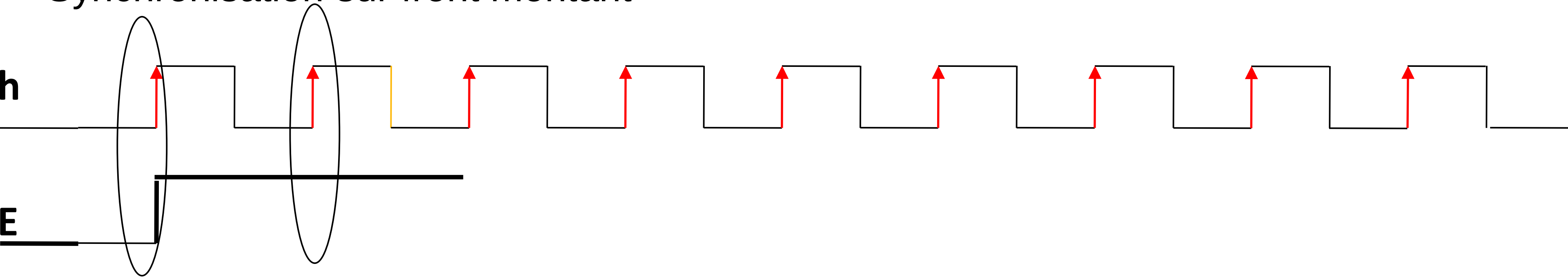
Fréquence F $f = \frac{1}{T}$

La fréquence est en hertz

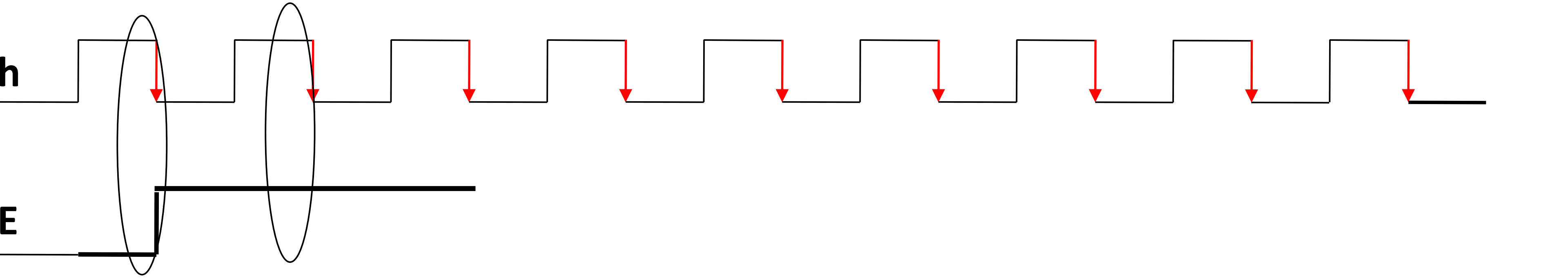
Synchronisation sur niveau Haut



Synchronisation sur front montant

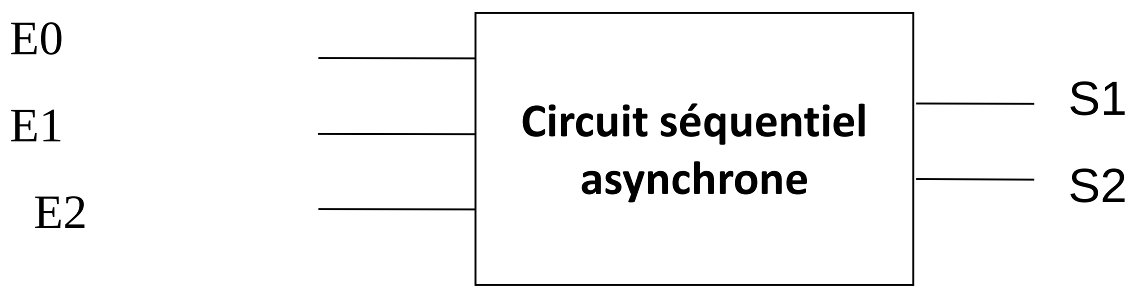


Synchronisation sur front descendant



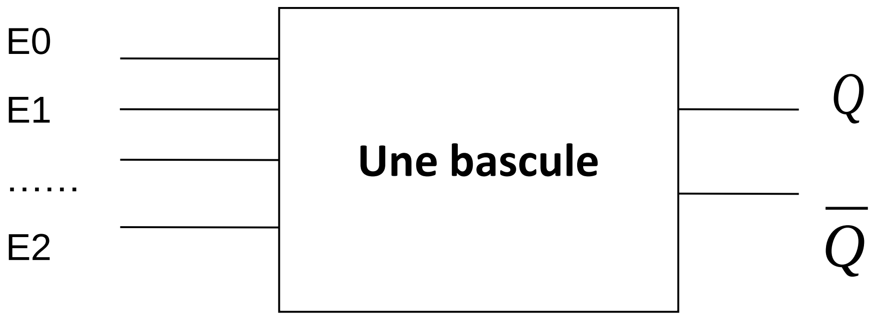
Les systèmes Asynchrones

- Lorsque un circuit séquentiel n'a pas d'horloge comme variable d'entrée ou si le circuit fonctionne indépendamment de cette horloge alors ce circuit est asynchrone.



Les bascules (flip-flops)

- Les bascules sont les circuits de bases de la logique séquentiel .
- Une bascule peut posséder une horloge (synchrone) ou non (asynchrone) .
- Chaque bascule possède des entrées et deux sorties et Q . $\overline{Q}$
- Une bascule possède la fonction de mémoration et de basculement.



$$Q^{+} = F(Ei, Q)$$

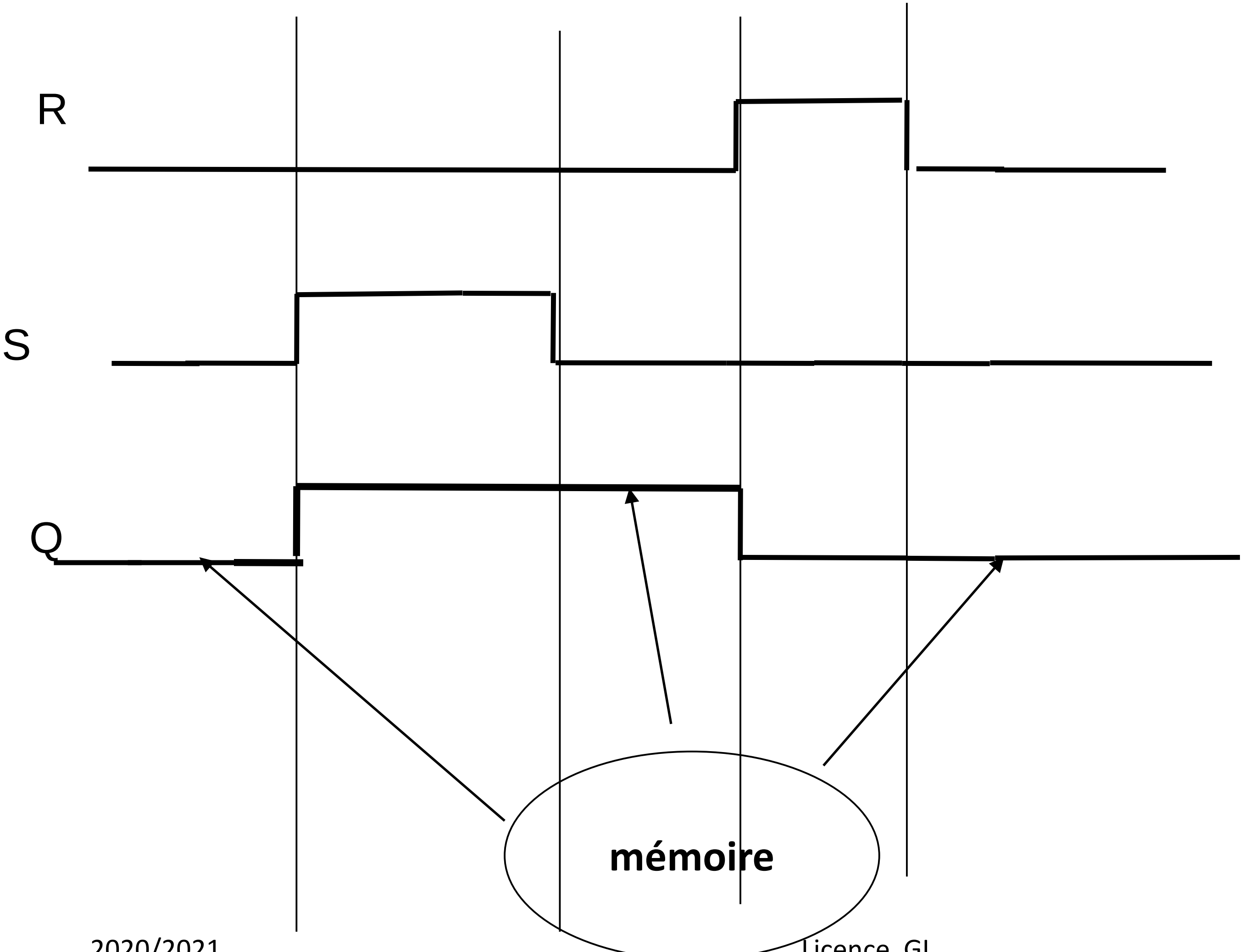
Il existe plusieurs types de bascules :T ,RS, RST ,D ,JK

Les bascules RS (Reset,Set)

R	S		Q+
0	0		Q-
0	1		1
1	0		0
1	1		X

R	S	Q-		Q+	
0	0	0		0	Etat mémoire
0	0	1		1	
0	1	0		1	Remise à 1
0	1	1		1	
1	0	0		0	Remise à 0
1	0	1		0	
1	1	0		X	État interdite
1	1	1		X	

Chronogramme d'une bascule RS



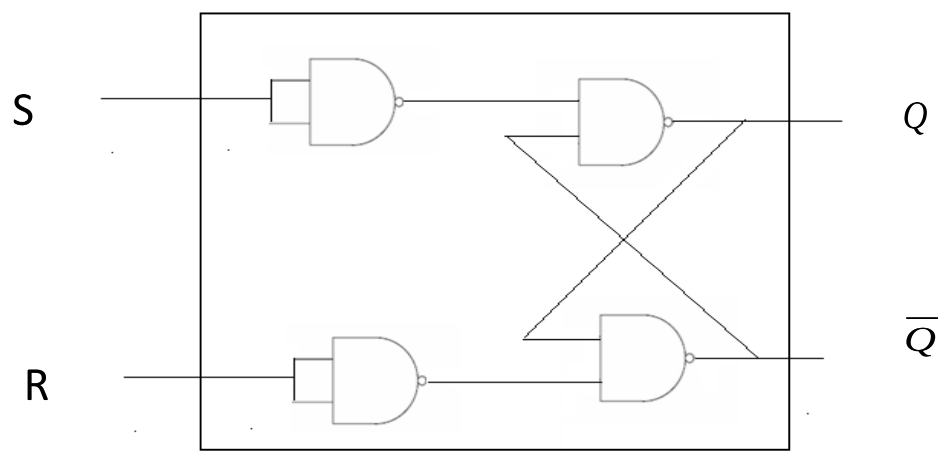
Structure interne d'une bascule RS

$$Q^+ = S + \overline{R}.Q$$

$$\overline{Q^+} = R + \overline{S}.Q$$

$$Q^+ = S + \overline{R}.Q = \overline{\overline{S + \overline{R}.Q}} = \overline{S} \uparrow (\overline{R} \uparrow Q) = (S \uparrow S) \uparrow ((R \uparrow R) \uparrow Q)$$

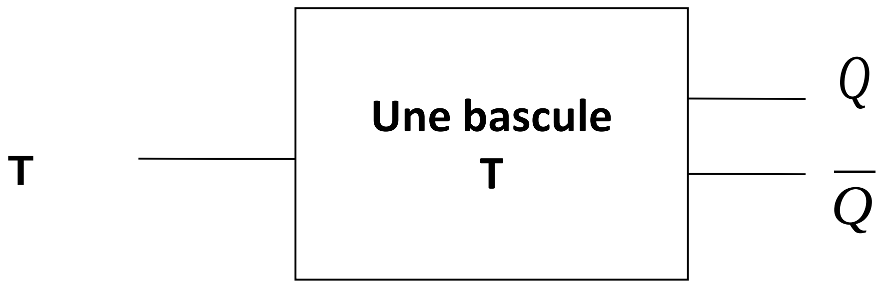
$$\overline{Q^+} = R + \overline{S}.Q = \overline{\overline{R + \overline{S}.Q}} = \overline{R} \uparrow (\overline{S} \uparrow Q) = (R \uparrow R) \uparrow ((S \uparrow S) \uparrow Q)$$



Les bascules RST

T	R	S		Q^+
0	X	X		Q
1	0	0		Q
1	0	1		1
1	1	0		0
1	1	1		X

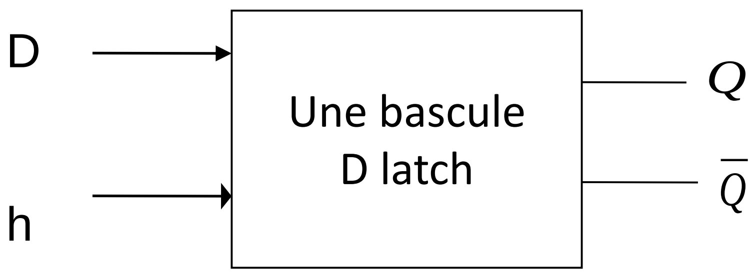
Les bascules T



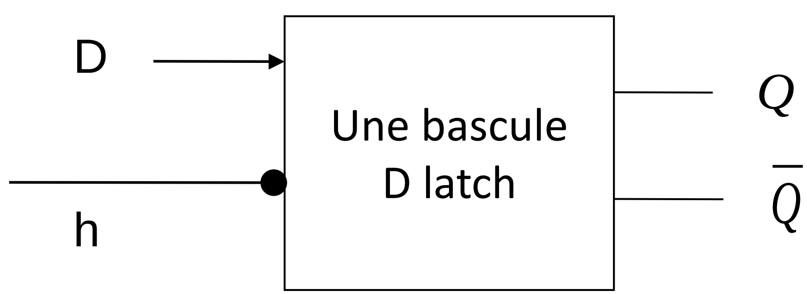
T		Q+
0		Q
1		$\overline{Q}$

Les bascules D latch

- C'est une bascule synchrone (utilise une horloge) sur niveau Haut ou niveau Bas



Sur niveau Haut

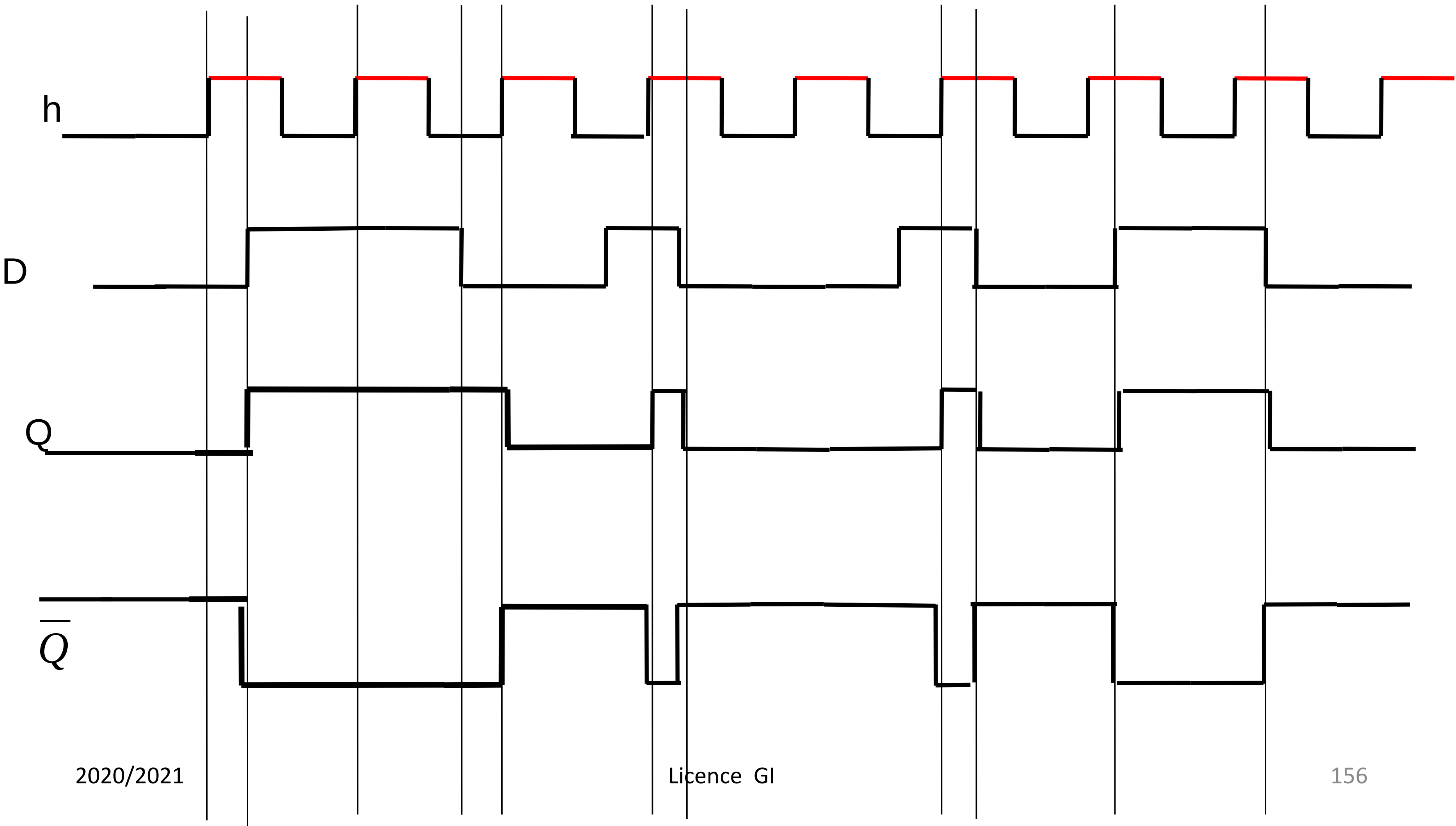


Sur niveau bas

h	**D**		**Q+**
0	0		Q-
0	1		Q-
1	0		0
1	1		1

Si h=1 Q^+=D

Chronogramme d'une bascule D latch (niveau haut)



Exercice

- Transformer une bascule RST pour quelles agisse comme une bascule D-latch ?

T	R	S		Q^+
0	X	X		Q
1	0	0		Q
1	0	1		1
1	1	0		0
1	1	1		X

T = h

S= D

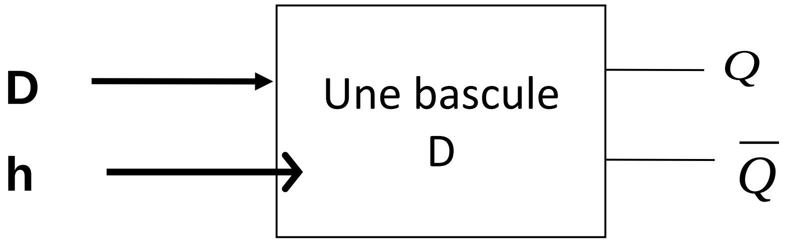
R= $\overline{D}$

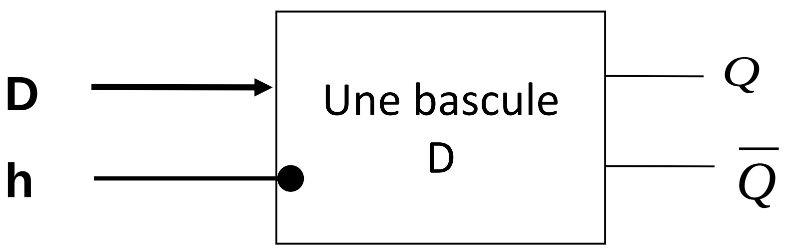
Les bascules D

- C'est une bascule synchronisée sur front montant ou descendant

h	D		Q+
0/1	0		Q-
0/1	1		Q-
↑	0		0
↑	1		1

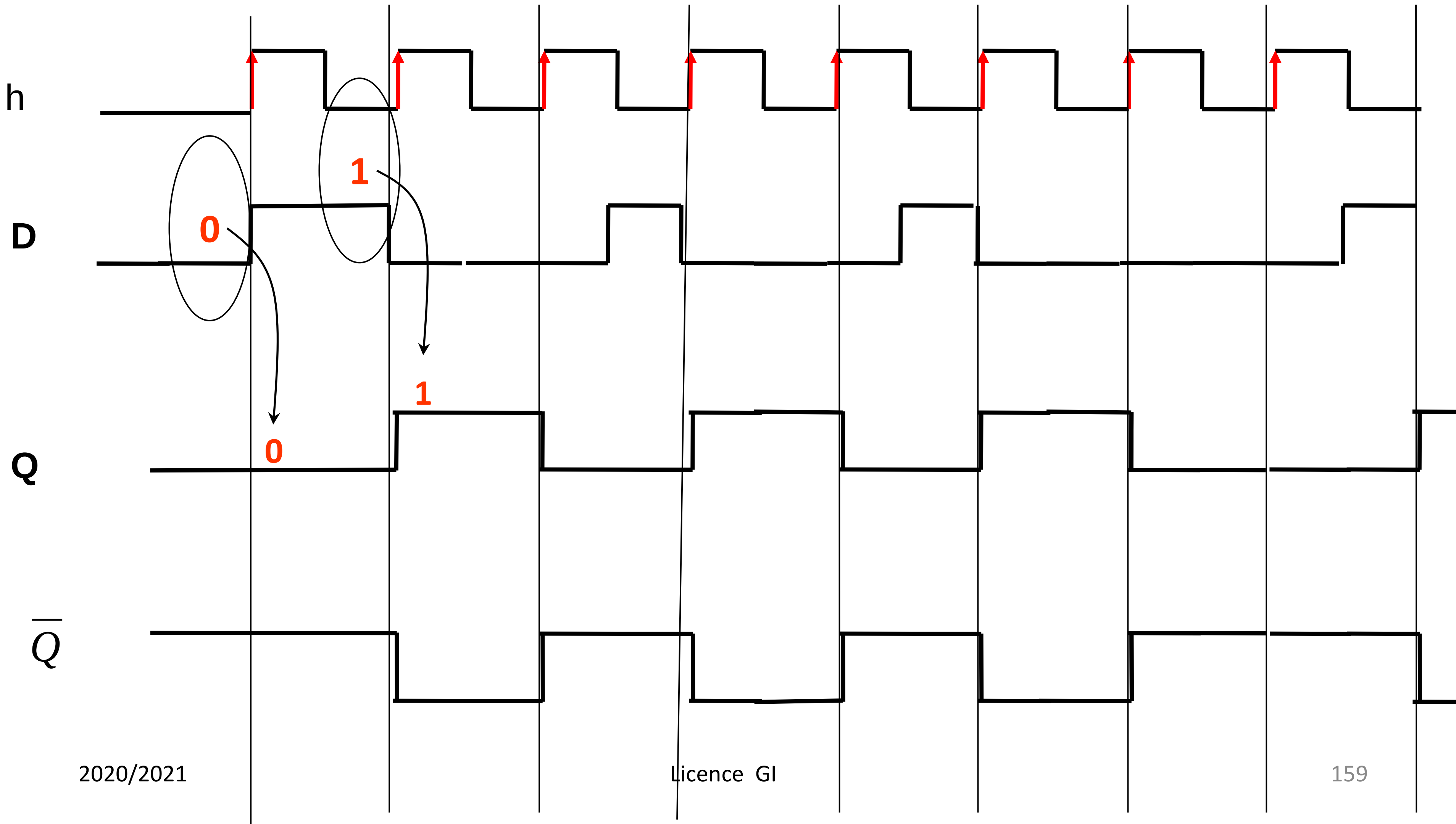
Sur front montant





Sur front descendant

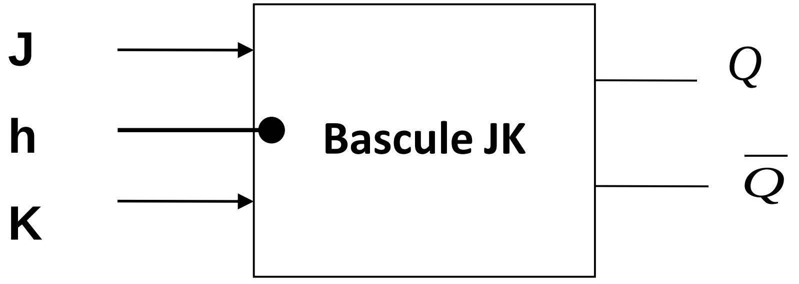
Chronogramme d'une bascule D



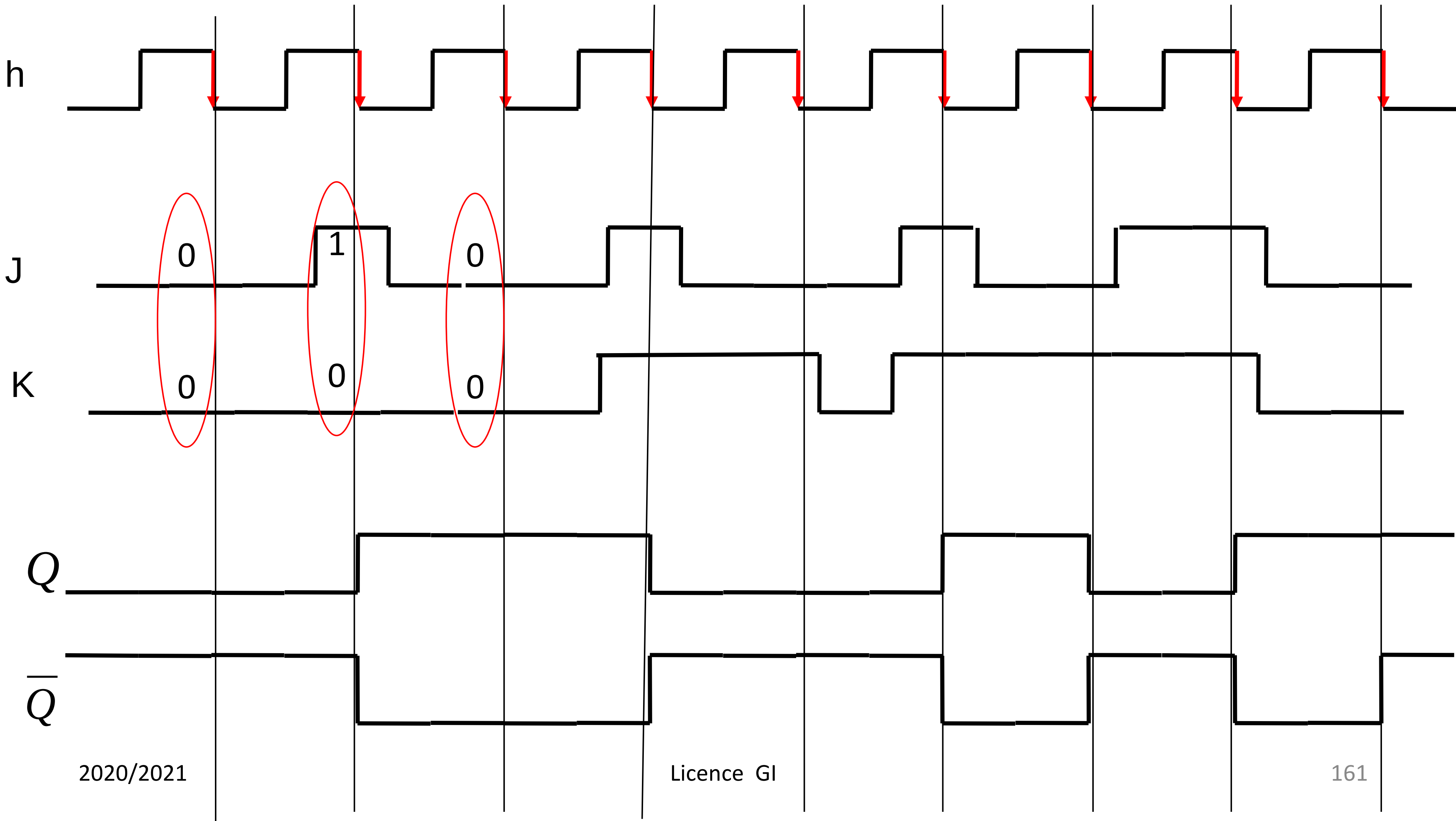
Les bascules J.K en mode synchrone

- Une bascule avec deux entrée J , K et une horloge (front montant ou descendant)

h	J	K		Q+
0/1	x	x		Q-
↓	0	0		Q-
↓	0	1		0
↓	1	0		1
↓	1	1		$\overline{Q}$



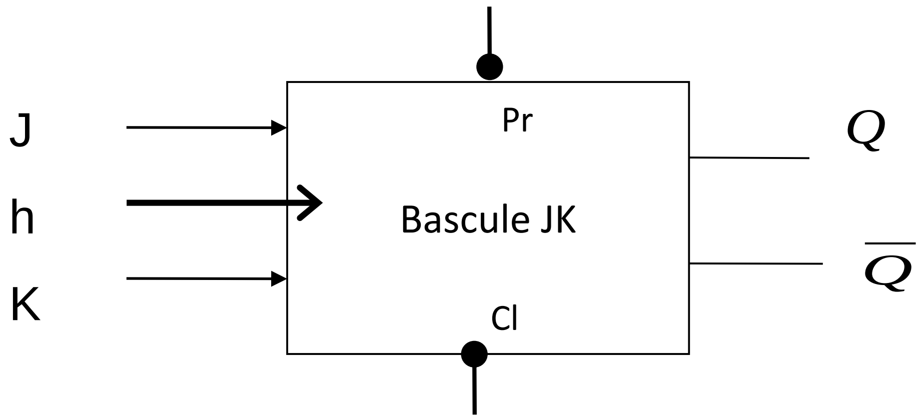
Chronogramme d'une bascule J.K



Les bascules J.K en mode asynchrone

- Deux entrées Pr (preset) et cl (clear) asynchrone
- Plus prioritaire que l'horloge
- Pr et Cl fonctionne avec la logique negative.

Sur front montant



Sur front descendant

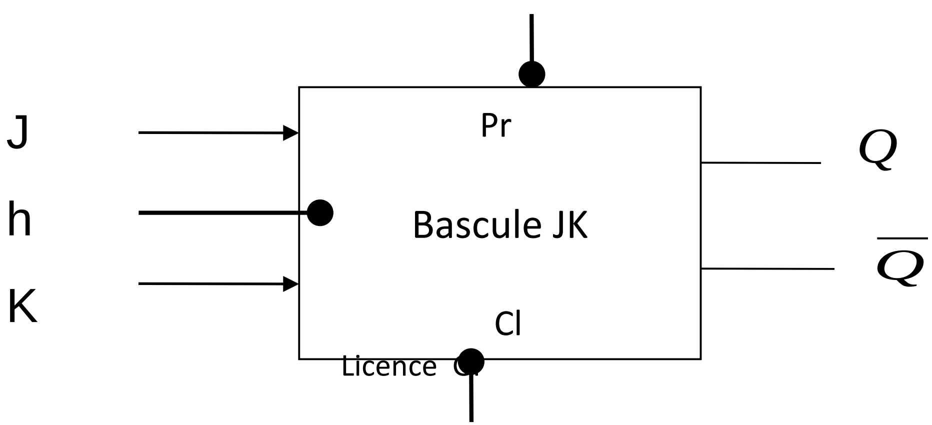


Table de vérité d'une bascule J.K

	Pr	Cl	h	J	K		Q^+	
Mode Asynchrone	**0**	**0**	**X**	**X**	**X**		**X**	**État interdit**
	0	**1**	**X**	**X**	**X**		**1**	**Remise à 1**
	1	**0**	**X**	**X**	**X**		**0**	**Remise à 0**
Mode Synchrone	**1**	**1**	**0/1**	**x**	**x**		Q^-	**Etat mémoire**
	1	**1**	$\downarrow$	**0**	**0**		Q^-	**Etat mémoire**
	1	**1**	$\downarrow$	**0**	**1**		**0**	**Remise à 0**
	1	**1**	$\downarrow$	**1**	**0**		**1**	**Remise à 1**
	1	**1**	$\downarrow$	**1**	**1**		$\overline{Q}$	**Basculement**

Exercice

- Transformer une bascule JK en une bascule D ?

h	J	K		Q+
0/1	x	x		Q-
$\downarrow$	0	0		Q-
$\downarrow$	0	1		0
$\downarrow$	1	0		1
$\downarrow$	1	1		$\overline{Q}$

$$J = D$$
$$K = \overline{D}$$
$$h = \overline{h1}$$

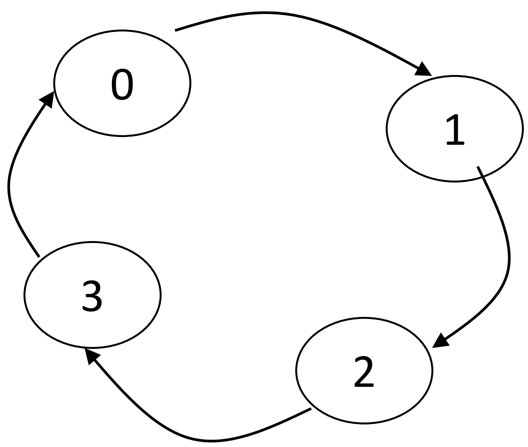
Table de transition d’une bascule JK

- On connait les valeurs des sorties , comment determiner les valeurs des entrées JK ?

Q	Q+		J	K	
0	0		0	X	Remise à 0 ou état mémoire
0	1		1	X	Remise à 1 ou basculement
1	0		X	1	Remise à 0 ou basculement
1	1		X	0	Remise à 1 ou état mémoire

Exercice

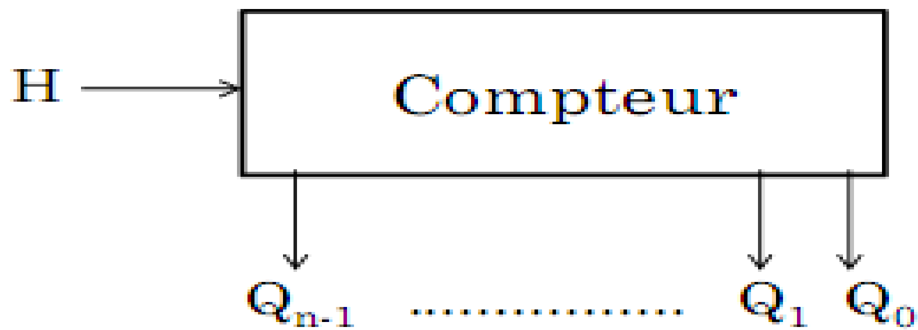
- Réaliser le circuit qui permet de réaliser le cycle suivant 0,1,2,3 à l'aide de bascules JK?



Les compteurs

COMPTEURS

DÉFINITION



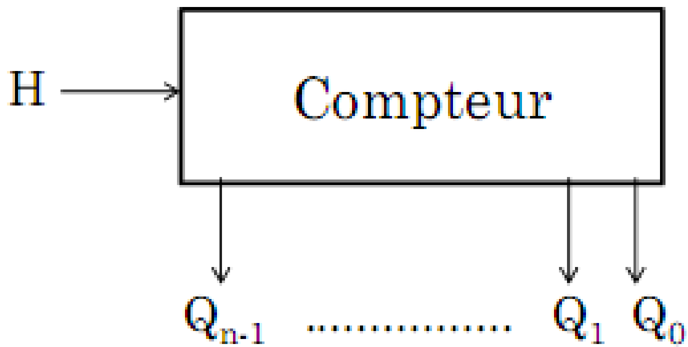
- Un compteur est une association de n bascules permettant de décrire, au rythme d'une horloge, une séquence déterminée:

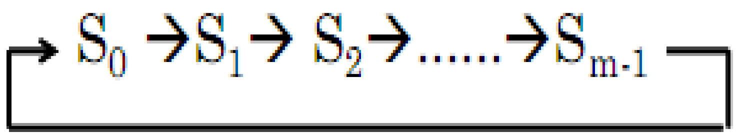
$$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow \dots\dots \rightarrow S_{m-1}$$

- Cette séquence est appelée cycle du compteur

COMPTEURS

DÉFINITION





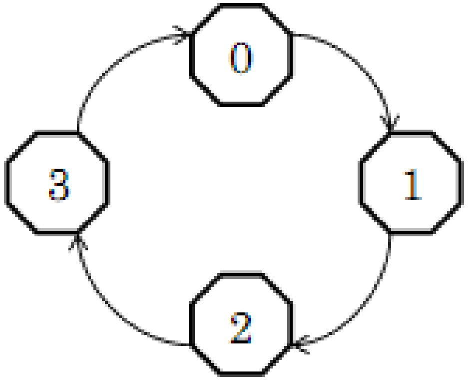
Une combinaison de sortie d'un compteur (Qn-1 .... Q1Q0) est appelée état.

Le nombre d'états différents (Si) pour un compteur est appelé le modulo (m) de ce compteur: m<2n

COMPTEURS
EXEMPLES

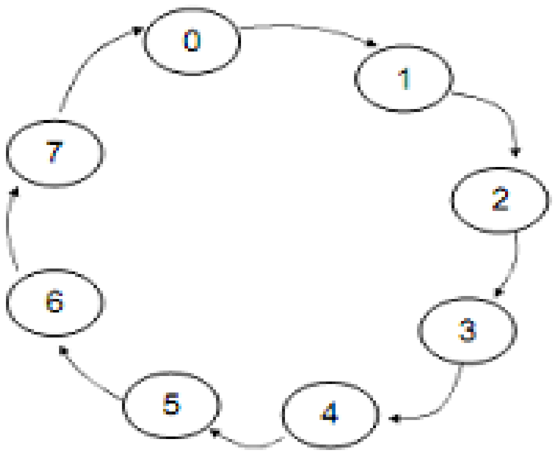
- Compteur modulo 4 (cycle complet)

Nombre d'impulsion (H)	Sorties		Valeur décimale
	Q1	Q0	
0	0	0	0
1	0	1	1
2	1	0	2
3	1	1	3
4	0	0	0
5	0	1	1

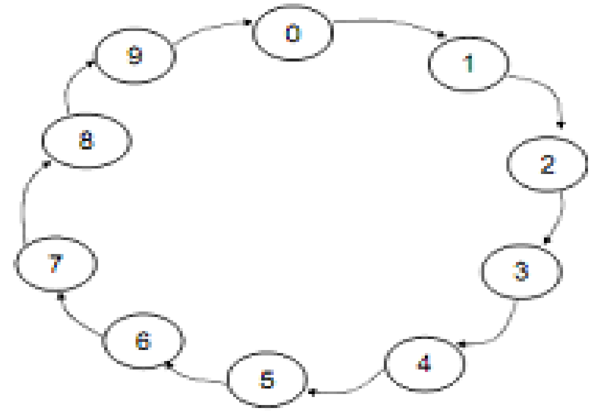


COMPTEURS

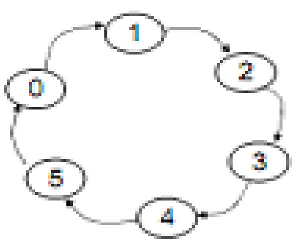
EXEMPLES



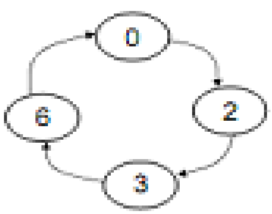
Compteur modulo 8
(cycle complet)
n = 3



Compteur modulo 10
(cycle incomplet)
n = 4



Compteur modulo 6
(cycle incomplet)
n = 3



Compteur modulo 4
Cycle quelconque
n = 3

COMPTEURS

TYPE

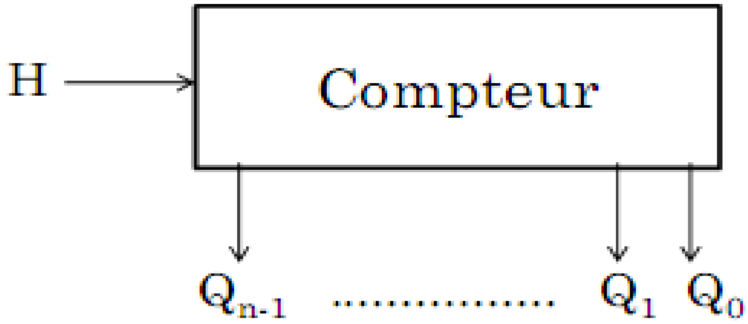
Selon le cycle des compteurs, nous distinguons entre:

- **Les compteurs modulo 2n (cycle complet):**
  - n=2 : 0 ,1,2,3,0 modulo 4
  - n=3 : 0,1,2,3,4,5,6,7,0 modulo 8
  - n=4 : 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0 modulo 16

- **Les compteurs modulo N (cycle incomplet)**
  - Pour N=5 : 0,1,2,3,4,0 n=3
  - Pour N= 10 : 0,1,2,3,4,5,6,7,8,9,0 n = 4
- **Les compteurs à cycle quelconque :**
  - 0,2,3,6,0 n = 3
  - 0,2,5,6,7,8,10,0 n = 4

COMPTEURS

TYPE



Selon l'horloge des bascules, nous distinguons entre :

- **Les Compteurs Asynchrones:** les bascules possèdent des horloges différentes.

- **Les Compteurs Synchrones:** les bascules possèdent la même horloge.

COMPTEURS ASYNCHRONES

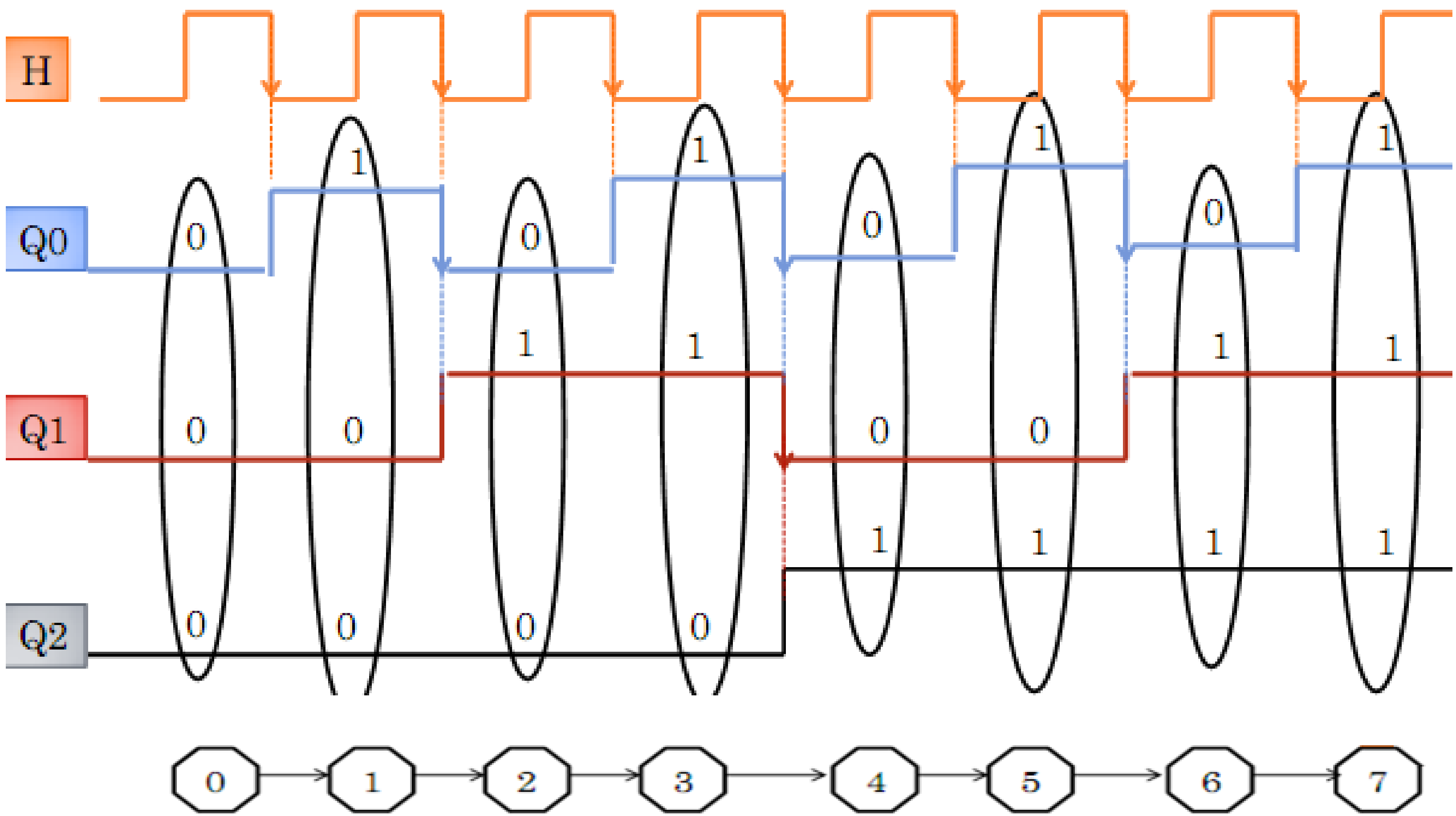
COMPTEURS ASYNCHRONES

Exemple: compteur modulo 2^3

États présents			États suivants		
Q2	Q1	Q0	Q2+	Q1+	Q0+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

COMPTEURS ASYNCHRONES

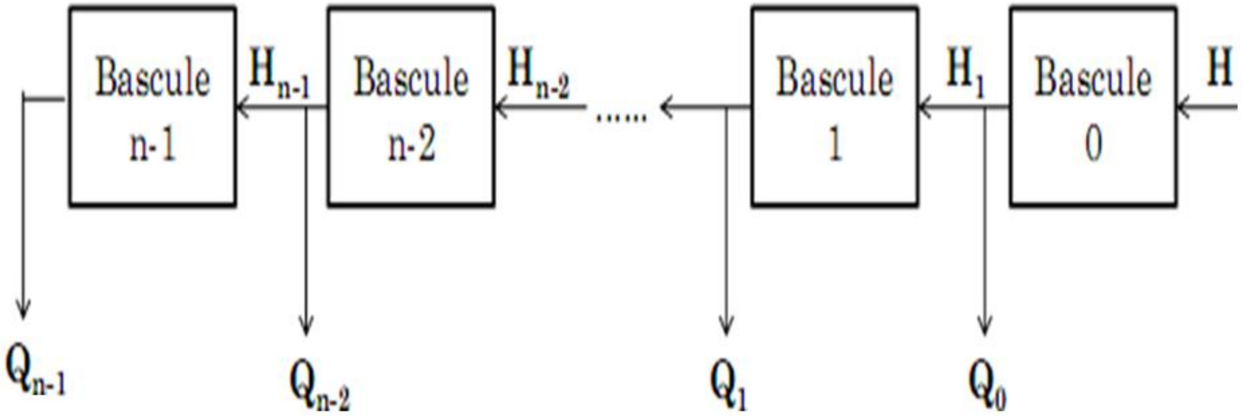
Exemple: compteur modulo 2^3



COMPTEURS ASYNCHRONE MODULO 2^N

Principe de fonctionnement

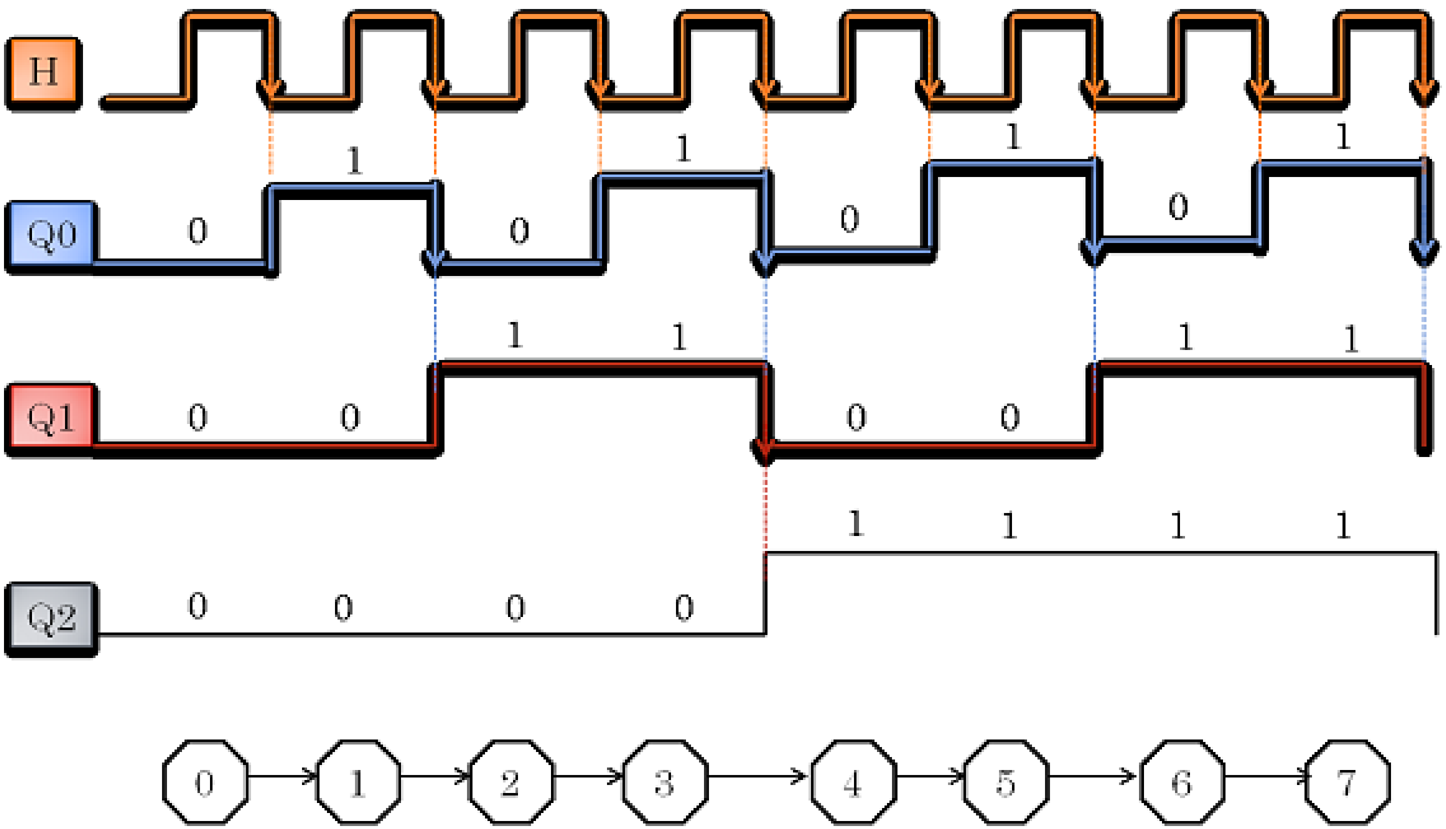
De manière générale, seule la première bascule reçoit le signal d'horloge. Toutes les bascules qui suivent celle-ci sont commandées par la bascule précédente.



COMPTEURS ASYNCHRONE

BASCULES APPROPRIÉES

Quelles sont les bascules appropriées pour construire les compteurs?

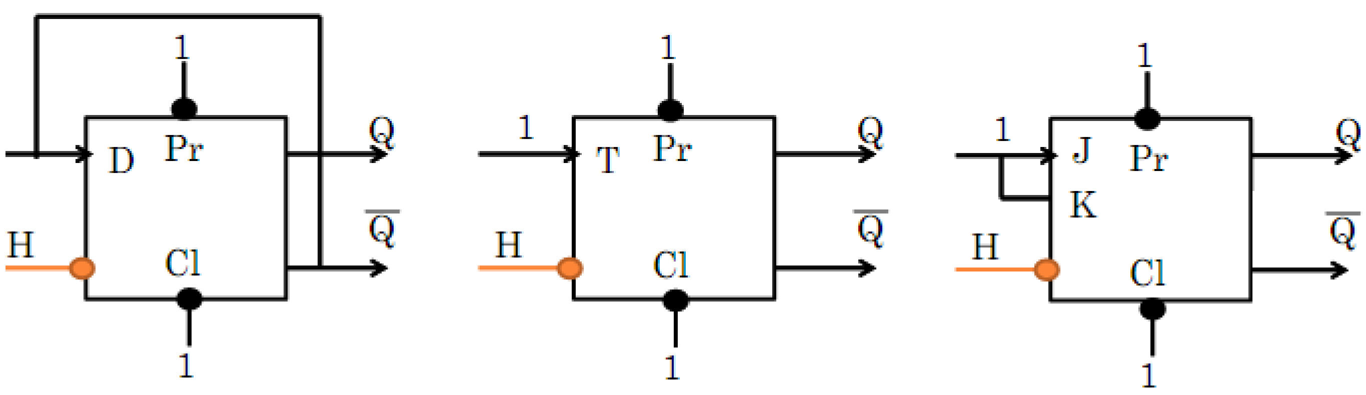


COMPTEURS ASYNCHRONE

BASCULES APPROPRIÉES

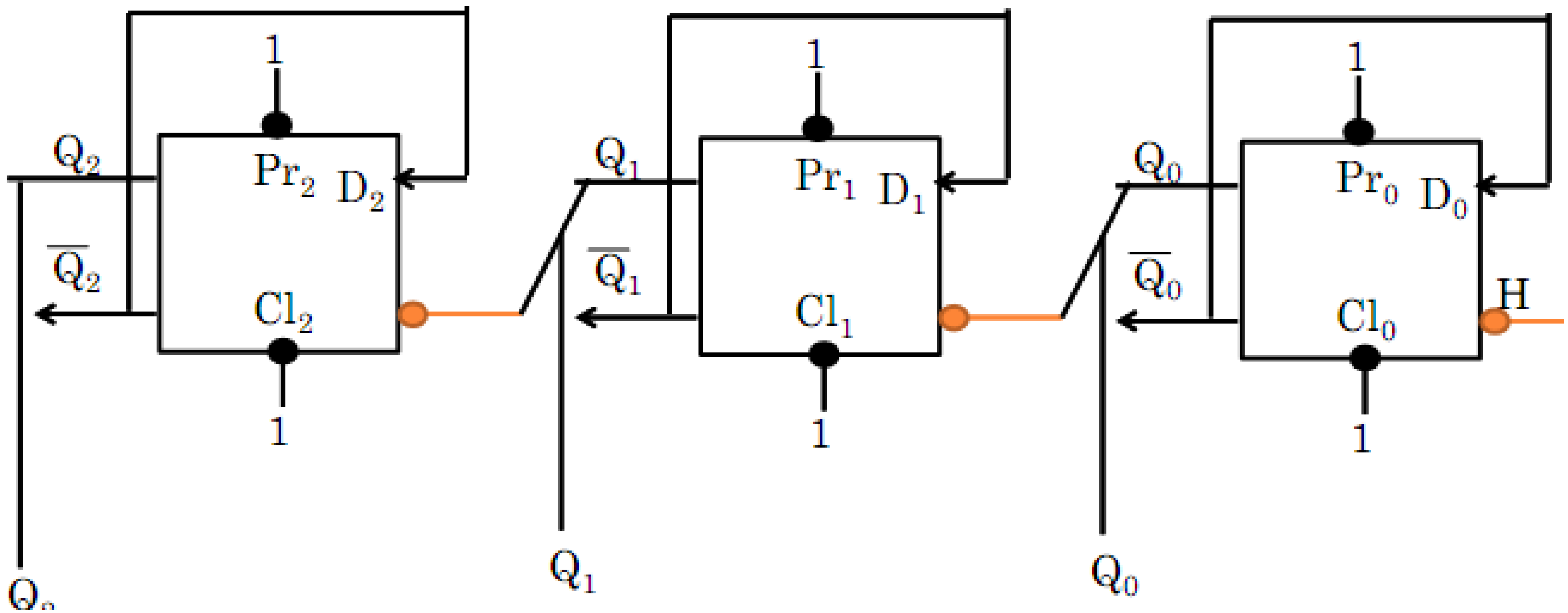
Quelles sont les bascules appropriées pour construire les compteurs?

Les bascules synchrones sur front qui permettent de réaliser l'état de basculement Q+ = /Q-



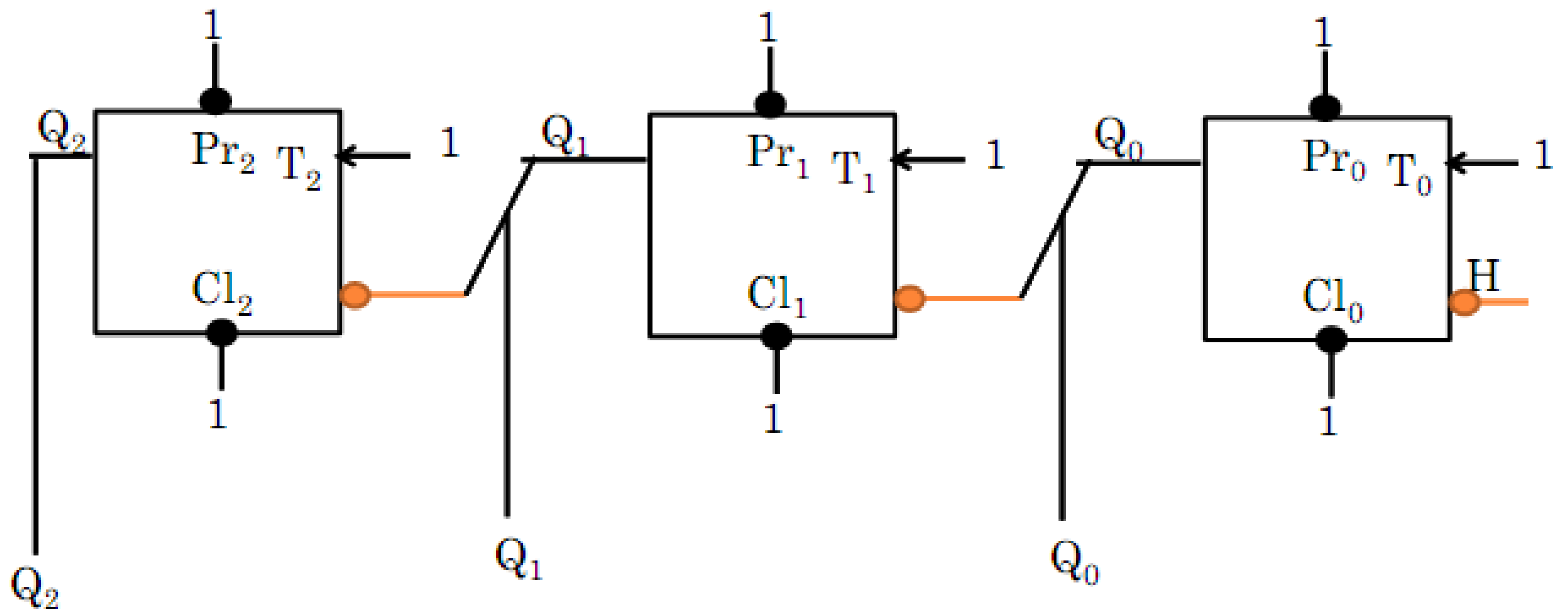
COMPTEURS ASYNCHRONE

EXEMPLE: COMPTEUR MODULO 2^3 (BASCULE D)



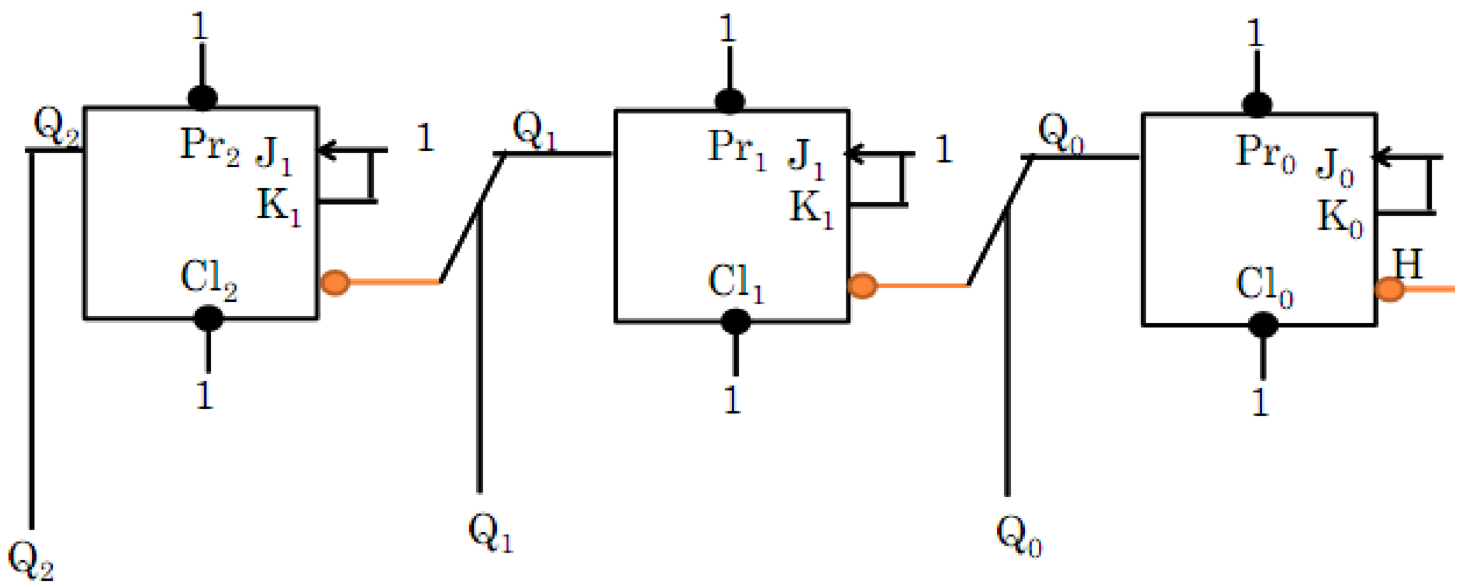
COMPTEURS ASYNCHRONE

EXEMPLE: COMPTEUR MODULO 2^3 (BASCULE T)



COMPTEURS ASYNCHRONE

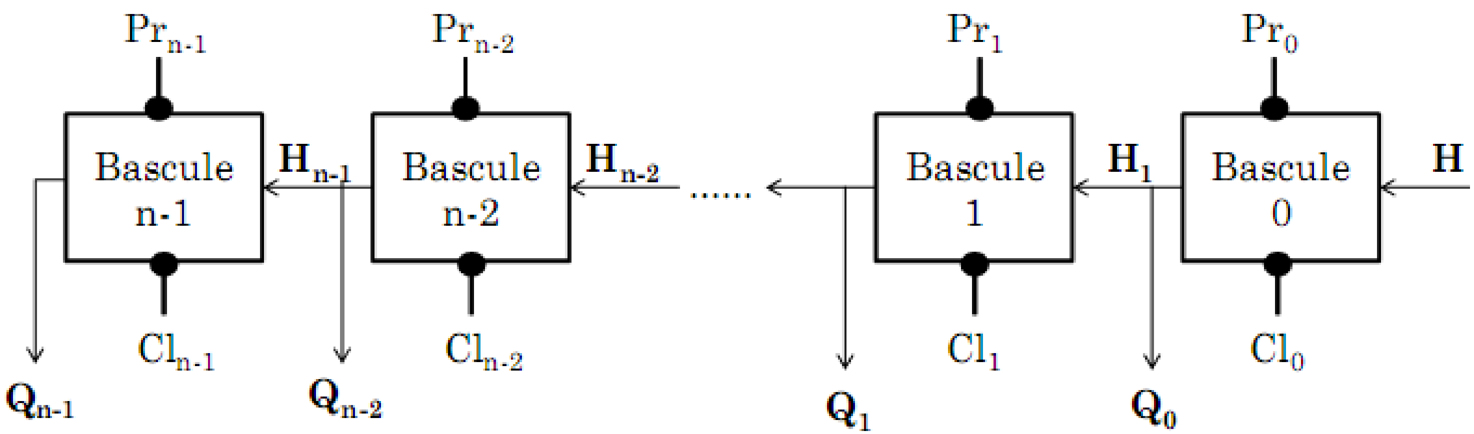
EXEMPLE: COMPTEUR MODULO 2^3 (BASCULE JK)



COMPTEURS ASYNCHRONE MODULO N

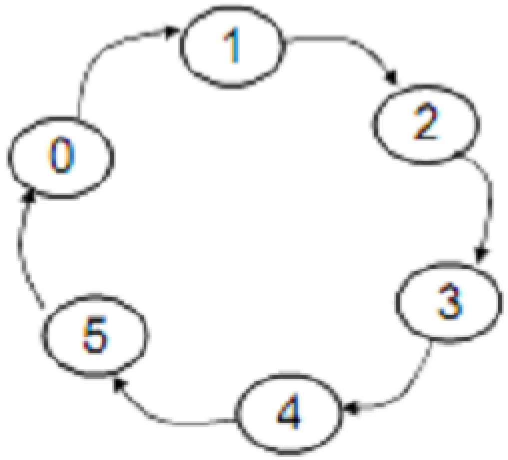
PRINCIPE DE FONCTIONNEMENT

Pour réaliser un compteur asynchrone modulo N, il faut agir sur les entrées d'initialisation (Clear et Preset) lorsque la combinaison correspondant au modulo N se produit sur les sorties du compteur.



COMPTEURS ASYNCHRONE

EXEMPLE: COMPTEUR MODULO 6



COMPTEURS ASYNCHRONE

EXEMPLE: COMPTEUR MODULO 6

États présents			États suivants				
Q2	Q1	Q0	Q2+	Q1+	Q0+	$\mathrm{Cl_i}$	$\mathrm{Pr_i}$
0	0	0	0	0	1	1	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	1	1
0	1	1	1	0	0	1	1
1	0	0	1	0	1	1	1
1	0	1	1	1	0	1	1
1	1	0	0	0	0	0	1
1	1	1	X	X	X	1	1

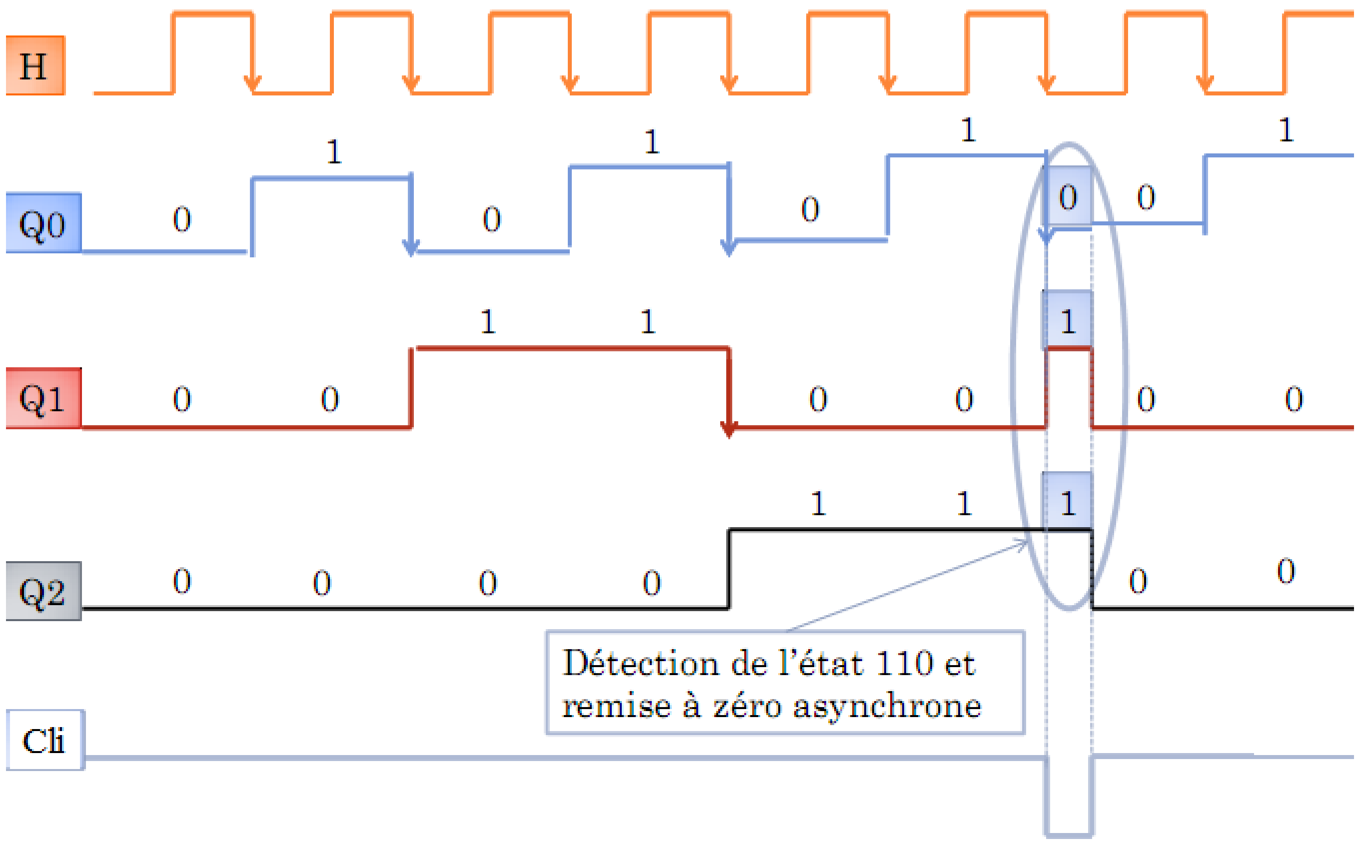
$$\overline{\mathrm{Cl_i}} = Q_2 Q_1 \overline{Q_0}$$

$$\mathrm{Cl_i} = \overline{Q_2 Q_1 \overline{Q_0}}$$

Détection de l'état 110 et remise à zéro asynchrone

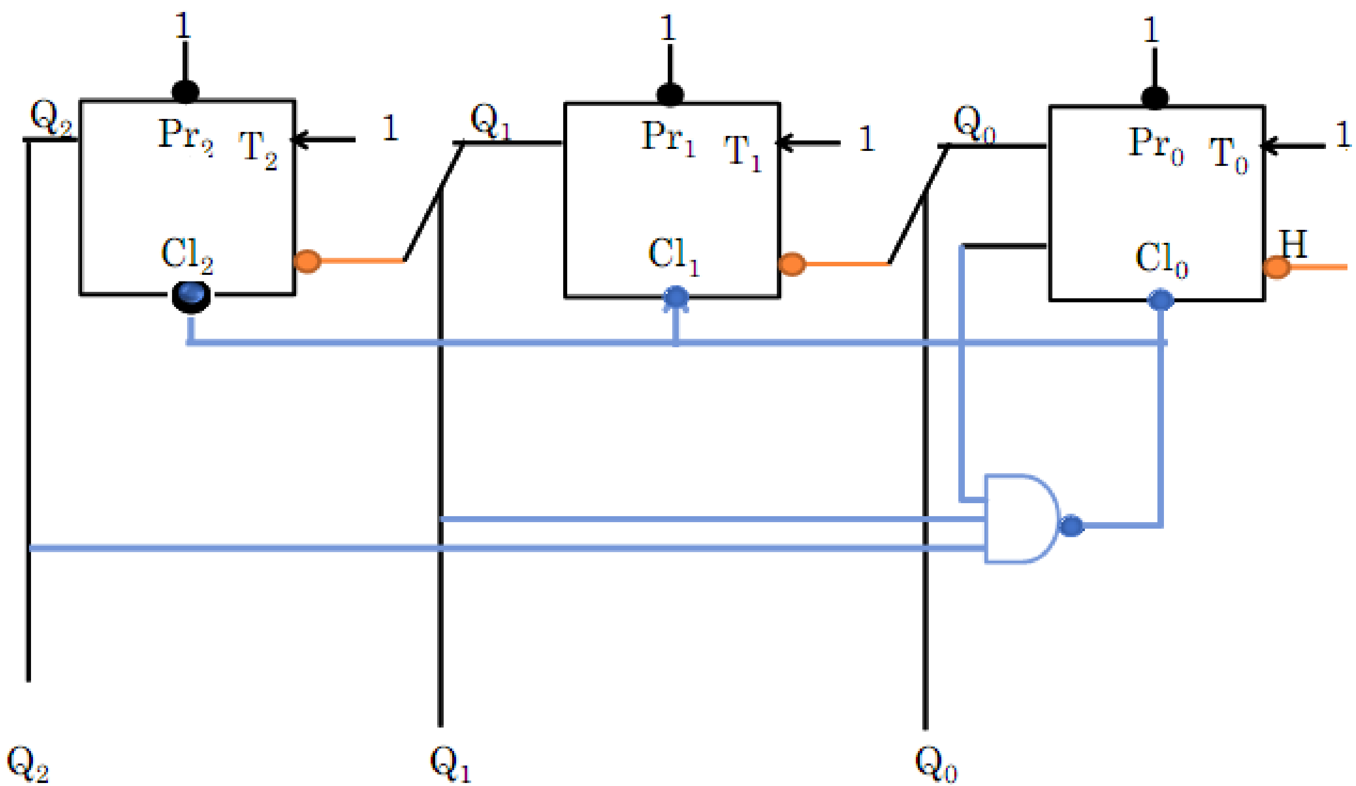
COMPTEURS ASYNCHRONE

EXEMPLE: COMPTEUR MODULO 6



COMPTEURS ASYNCHRONE

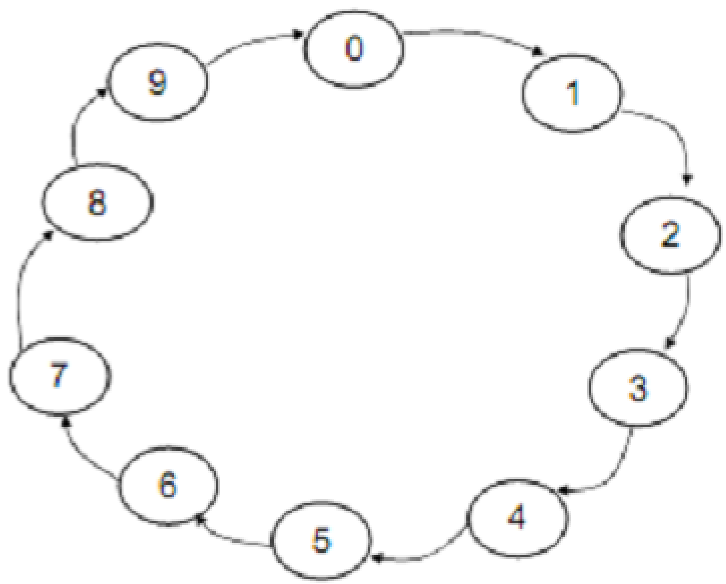
EXEMPLE: COMPTEUR MODULO 6



COMPTEURS ASYNCHRONE

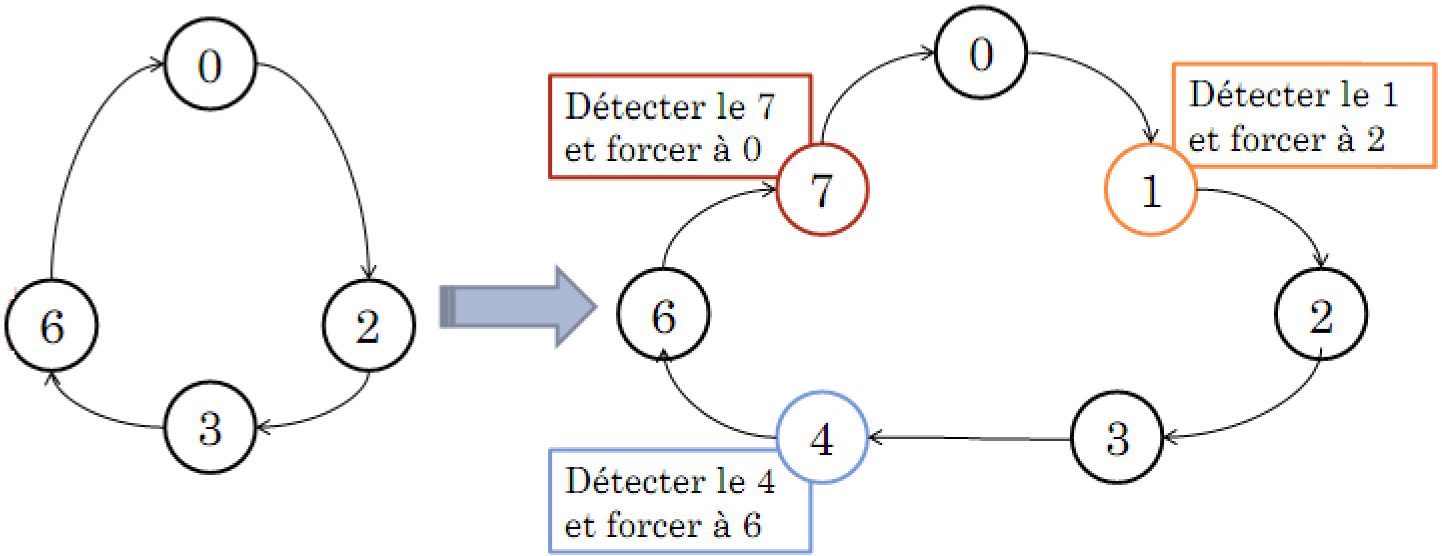
EXERCICE

Réaliser un compteur asynchrone décimale (modulo 10)



COMPTEURS ASYNCHRONES A CYCLE QUELCONQUE

Soit le compteur ayant le cycle suivant



Pour forcer le compteur d'un état à un autre, il faut agir sur les entrées asynchrone Cli et Pri des bascules

COMPTEURS ASYNCHRONES A CYCLE QUELCONQUE

Q2	Q1	Q0	Q2+	Q1+	Q0+	Cl2	Pr2	Cl1	Pr1	Cl0	Pr0
0	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	0
0	1	0	0	1	1	1	1	1	1	1	1
0	1	1	1	0	0	1	1	1	1	1	1
1	0	0	1	1	0	1	1	0	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	1	0	1	0	1	0

$$\begin{cases} Cl_2 = 1 \\ Pr_2 = \overline{Q_2 Q_1 Q_0} \end{cases}; \quad \begin{cases} Cl_1 = \overline{\overline{Q_2}\,\overline{Q_1} Q_0 + Q_2 \overline{Q_1}\,\overline{Q_0}} \\ Pr_1 = \overline{Q_2 Q_1 Q_0} \end{cases}; \quad \begin{cases} Cl_0 = 1 \\ Pr_0 = \overline{\overline{Q_2}\,\overline{Q_1} Q_0 + Q_2 Q_1 Q_0} \end{cases}$$

COMPTEURS ASYNCHRONES A CYCLE QUELCONQUE

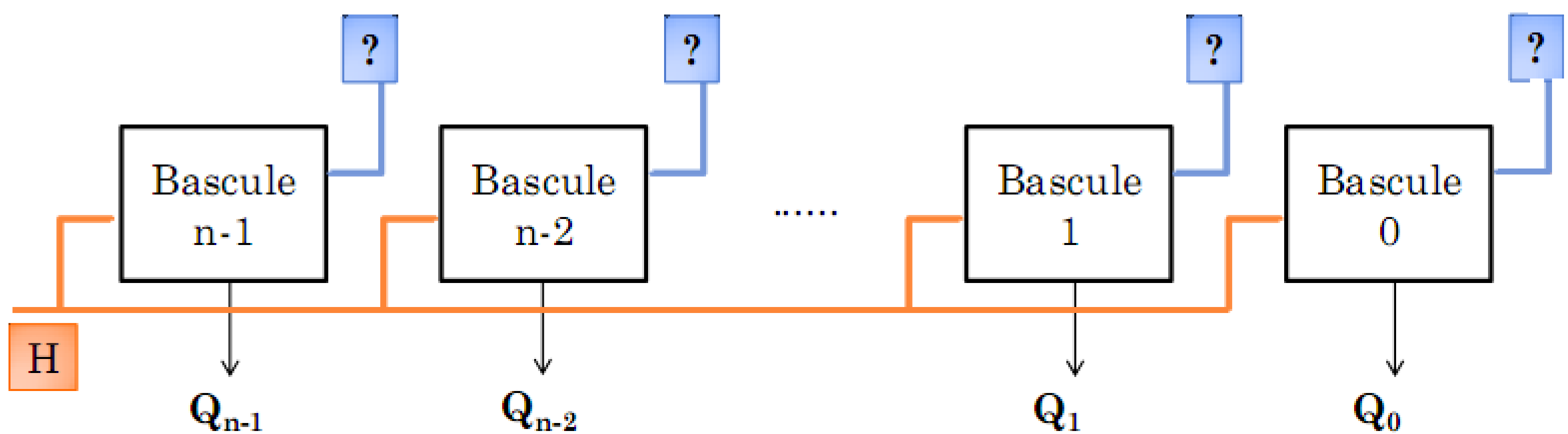
Pour réaliser un compteur asynchrone à cycle quelconque, il faut agir sur les entrées d’initialisation (Clear et Preset) lorsque une combinaison interdite (n’appartient pas au cycle) se produit sur les sorties du compteur.

COMPTEURS SYNCHRONES

COMPTEURS SYNCHRONE

STRUCTURE GÉNÉRALE

Un compteur synchrone est une structure où toutes les bascules reçoivent le même signal d'horloge. La fonction comptage est réalisée par l'intermédiaire des fonctions appliquées sur les entrées synchrones des bascules.



COMPTEURS SYNCHRONE

ÉTAPES DE RÉALISATION

- Déterminer le nombre de bascules nécessaires « n »
- Établir la table de transition du compteur [état suivant (Qi+) en fonction de l'état présent (Qi)]
- Déterminer l'expression des entrées des bascules

COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR MODULO 23 (BASCULE JK)

Q2	Q1	Q0	Q2+	Q1+	Q0+	J2	K2	J1	K1	J0	K0
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

J0=K0=1, J1= K1= Q0, J2=K2=Q0.Q1

COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR MODULO 23 (BASCULE T)

Q2	Q1	Q0	Q2+	Q1+	Q0+	T2	T1	T0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

T0=1, T1= Q0, T2=Q0.Q1

COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR MODULO 23 (BASCULE D)

Q2	Q1	Q0	Q2+	Q1+	Q0+	D2	D1	D0
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

$$D0 = \overline{Q0}$$

$$D1 = Q1 \oplus Q0$$

$$D2 = Q2 \oplus (Q1 . Q0)$$

COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR MODULO 2^3 (BASCULE D)

Q2	Q1	Q0	Q2+	Q1+	Q0+	D2	D1	D0
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

$$D0 = \overline{Q0}$$

$$D1 = Q1 \oplus Q0$$

$$D2 = Q2 \oplus (Q1.Q0)$$

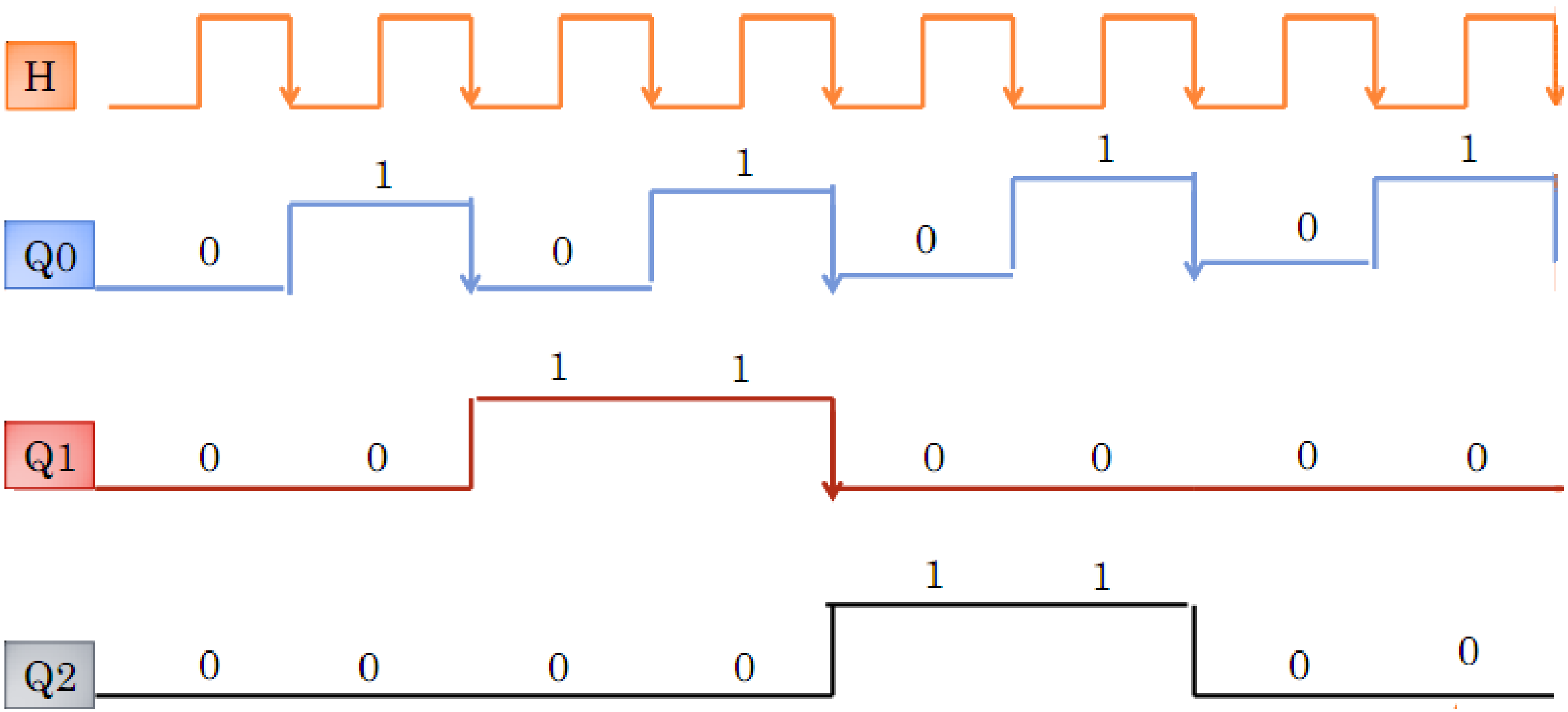
COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR MODULO 6 (BASCULE JK)

Q2	Q1	Q0	Q2+	Q1+	Q0+	J2	K2	J1	K1	J0	K0
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**	**0**	**0**	**0**	**X**	**1**	**0**	**X**	**X**	**1**
1	**1**	**0**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

COMPTEURS SYNCHRONE

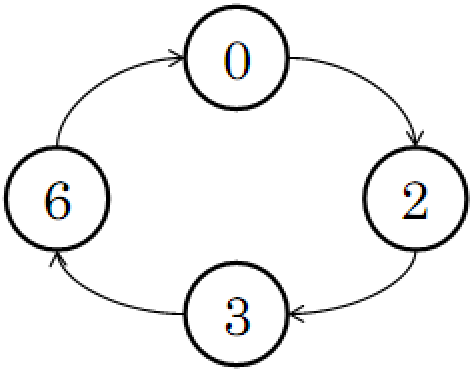
EXEMPLE: COMPTEUR MODULO 6 (BASCULE JK)



COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR A CYCLE QUELCONQUE

Soit le compteur ayant le cycle suivant



- Pour forcer le compteur d'un état à un autre, il faut agir sur les entrées synchrones (Di, Ji et Ki ou Ti).

- Pour les états qui n'appartiennent pas au cycle du compteur, il faut les considérer comme étant des états indéterminés.

COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR A CYCLE QUELCONQUE(BASCULE JK)

Q2	Q1	Q0	Q2+	Q1+	Q0+	J2	K2	J1	K1	J0	K0
0	0	0	0	1	0	0	X	1	X	0	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	1	0	1	X	X	0	X	1
1	1	0	0	0	0	X	1	X	1	0	X
0	0	1	X	X	X	X	X	X	X	X	X
1	0	0	X	X	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

$$\begin{cases} J_2 = K_2 = Q_1(Q_2 \oplus Q_0) \\ J_1 = K_1 = \overline{Q_0}\,\overline{(Q_2 \oplus Q_1)} \\ J_0 = K_0 = \overline{Q_2}Q_1 \end{cases}$$

COMPTEURS SYNCHRONE

EXEMPLE: COMPTEUR A CYCLE QUELCONQUE(BASCULE JK)

Q2	Q1	Q0	Q2+	Q1+	Q0+	J2	K2	J1	K1	J0	K0
0	0	0	0	1	0	0	X	1	X	0	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	1	0	1	X	X	0	X	1
1	1	0	0	0	0	X	1	X	1	0	X
0	0	1	X	X	X	X	X	X	X	X	X
1	0	0	X	X	X	X	X	X	X	X	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

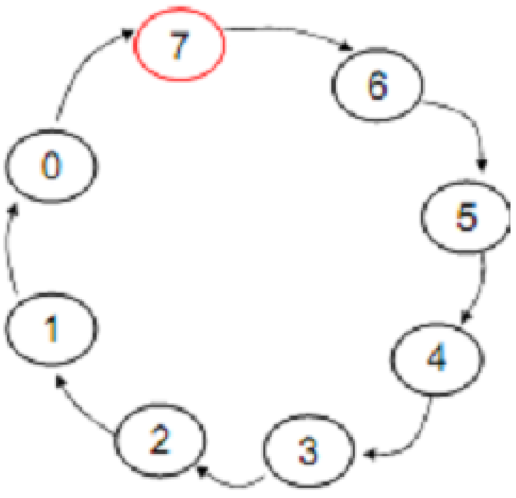
$$\begin{cases} J_2 = K_2 = Q_1(Q_2 \oplus Q_0) \\ J_1 = K_1 = \overline{Q_0}\,\overline{(Q_2 \oplus Q_1)} \\ J_0 = K_0 = \overline{Q_2}\,Q_1 \end{cases}$$

DÉCOMPTEURS

L’études des décompteurs se fait exactement de la même manière que l’étude des compteurs.

Exemple d’un décompteur modulo 8:

Q2	Q1	Q0	Q2+	Q1+	Q0+
1	1	1	1	1	0
1	1	0	1	0	1
1	0	1	1	0	0
1	0	0	0	1	1
0	1	1	0	1	0
0	1	0	0	0	1
0	0	1	0	0	0
0	0	0	1	1	1



DÉCOMPTEURS

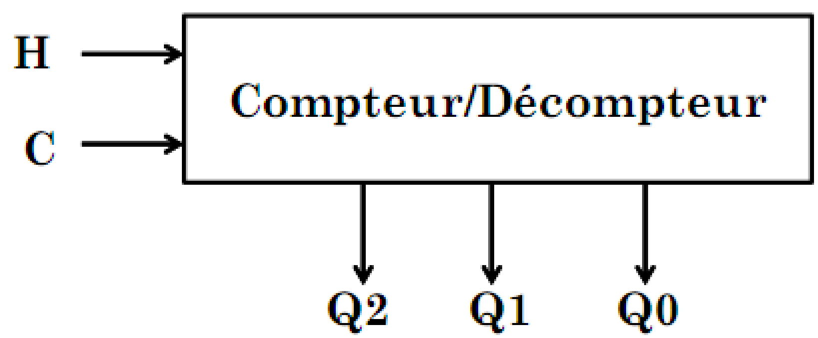
EXEMPLE: DÉCOMPTEUR SYNCHRONE MODULO 23 (BASCULE T)

Q2	Q1	Q0	Q2+	Q1+	Q0+	T2	T1	T0
1	1	1	1	1	0	0	0	1
1	1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	0	1
1	0	0	0	1	1	1	1	1
0	1	1	0	1	0	0	0	1
0	1	0	0	0	1	0	1	1
0	0	1	0	0	0	0	0	1
0	0	0	1	1	1	1	1	1

$$T0=1, \quad T1=\overline{Q0}, \quad T2=\overline{Q0}.\overline{Q1}$$

COMPTEURS/DÉCOMPTEURS

Le circuit Compteur/Décompteur peut offrir à la fois l'opération de comptage et décomptage. Pour ce faire, il faut rajouter une entrée de commande C qui indique le type de l'opération (par exemple: si C=0 alors comptage, sinon décomptage)



COMPTEURS/DÉCOMPTEURS MODULO 8

EXEMPLE

C	Q2	Q1	Q0	Q2+	Q1+	Q0+	T2	T1	T0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	0	1	1	0	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	0	1	1	0	1	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	0	1	0	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1

Compteur

Décompteur

COMPTEURS/DÉCOMPTEURS MODULO 8

EXEMPLE

C	Q2	Q1	Q0	T2	T1	T0
0	0	0	0	0	0	1
0	0	0	1	0	1	1
0	0	1	0	0	0	1
0	0	1	1	1	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	0	1
0	1	1	1	1	1	1
1	1	1	1	0	0	1
1	1	1	0	0	1	1
1	1	0	1	0	0	1
1	1	0	0	1	1	1
1	0	1	1	0	0	1
1	0	1	0	0	1	1
1	0	0	1	0	0	1
1	0	0	0	1	1	1

$T0=1,$

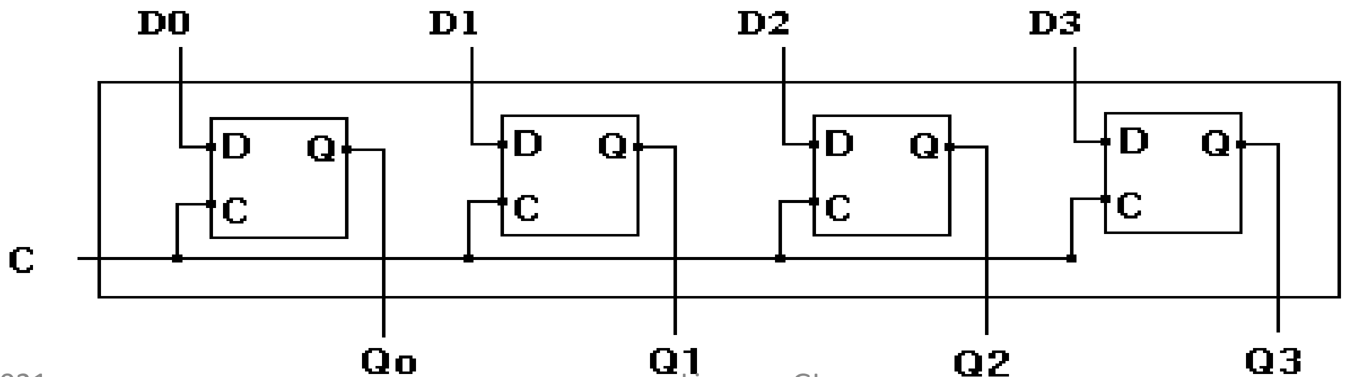
$T1= C \oplus Q0,$

$T2=\overline{C}\, Q0.Q1 + C\, \overline{Q0}.\overline{Q1}$

Les registres

Définition

- Une bascule est l'élément de base de la logique séquentielle.
- Une bascule permet de mémoriser un seul bit.
- Un registre est ensemble un ordonné de n bascules.
- Un registre permet de mémoriser (sauvegarder) une information sur n bits.
- Exemple :

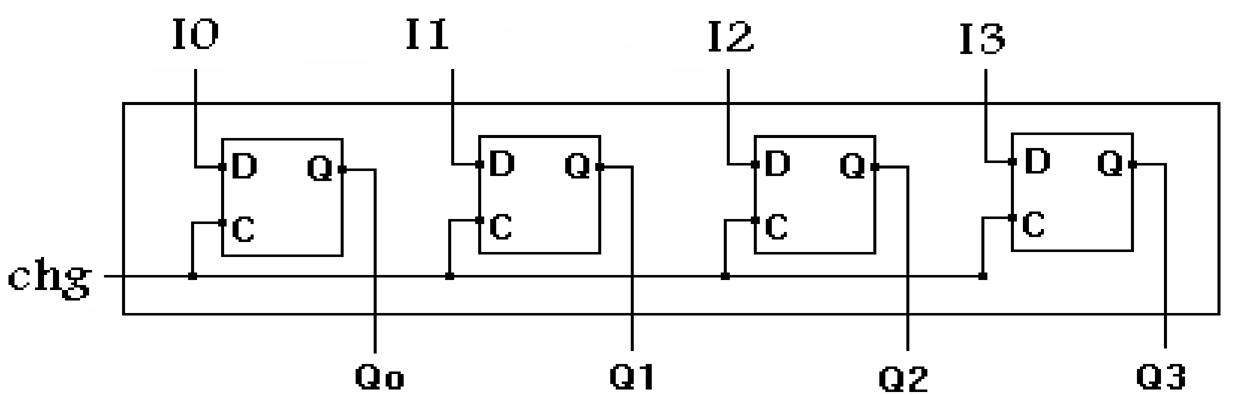


Type de registres

- Il existe plusieurs types de registres :
  - Registre à entrées parallèles et sorties parallèles (Registre à chargement parallèle).
  - Registre à entrée série et sortie série
  - Registre à entrée série et sortie parallèle.
  - Registre à entrée parallèle et sortie série.
  - Registre à décalage circulaire.

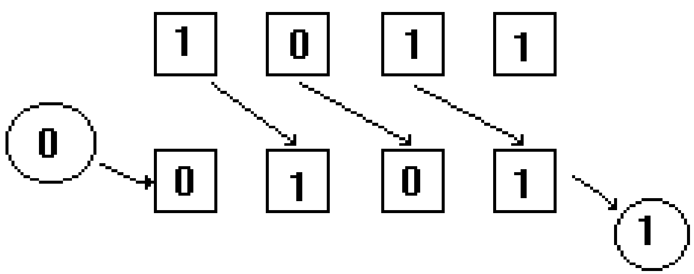
Registre à entrées parallèles et sorties parallèles (Registre à chargement parallèle).

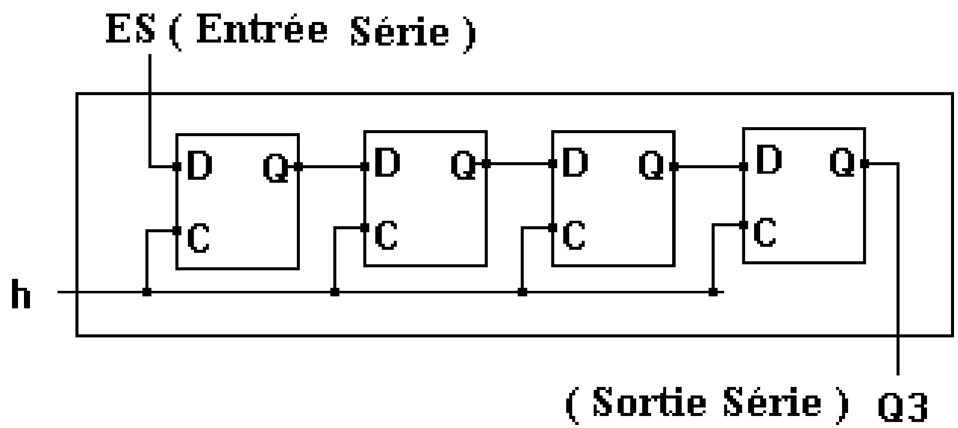
- Il peut charger une information sur N bits en même temps.
- Les n bascules changement d'états en même temps.
- Chaque bascule Bi prend la valeur de l'information i.
- Il possède une entrée de chargement chg (chg=0 → état mémoire, chg=1 chargement)



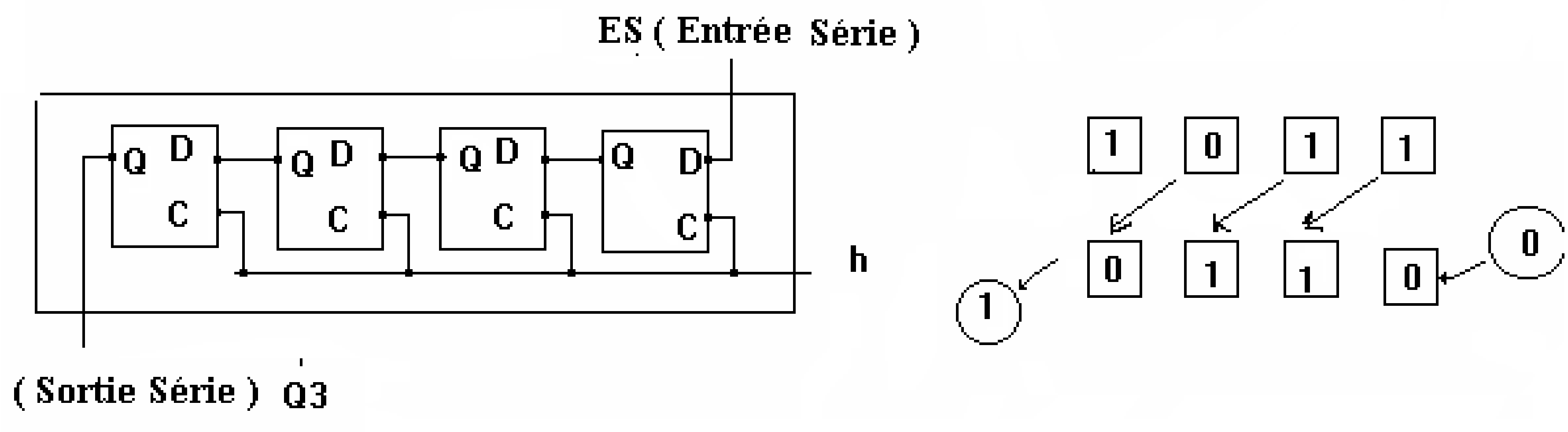
Registre à entrée série et sortie série

- L'information est introduite bit par bit (en série).
- L'ensemble du registre est décalé d'une position (Bi, Bi+1) et la bascule B0 reçoit une nouvelle entrée ES.
- Un tel registre est appelé registre à entrée série à gauche et à sortie série à droite.

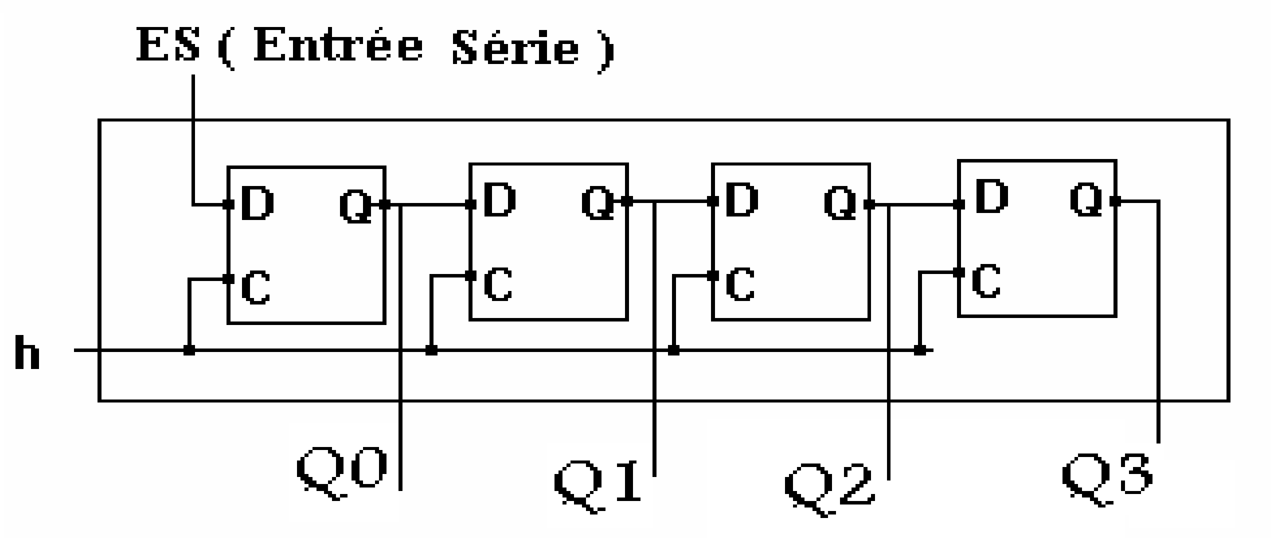




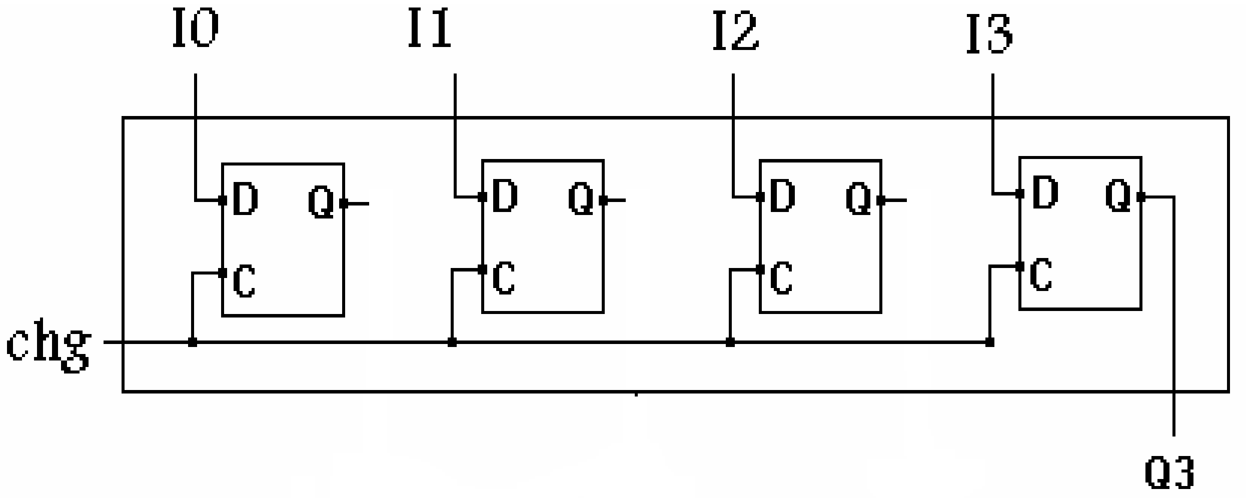
registre à entrée série à droite et à sortie série à gauche.



Registre à entrée série et sortie parallèle.

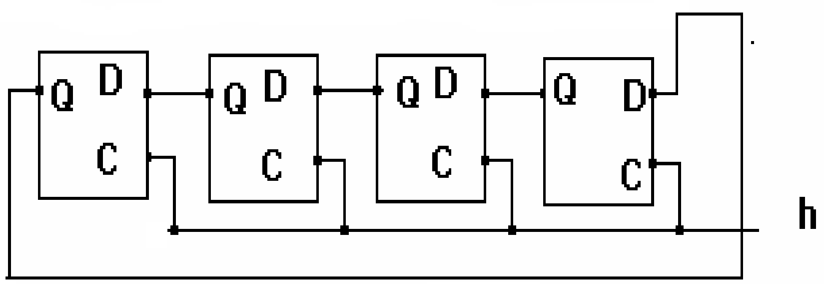


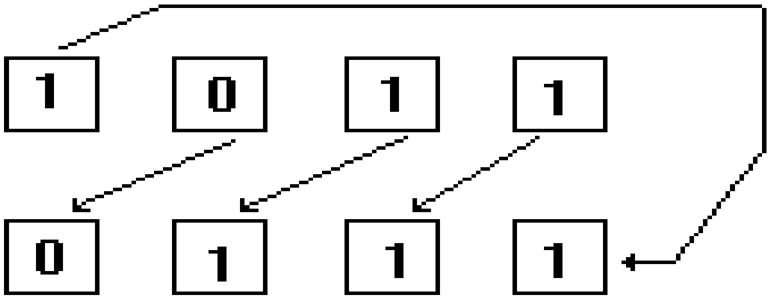
Registre à entrée parallèle et sortie série.



Registre à décalage circulaire

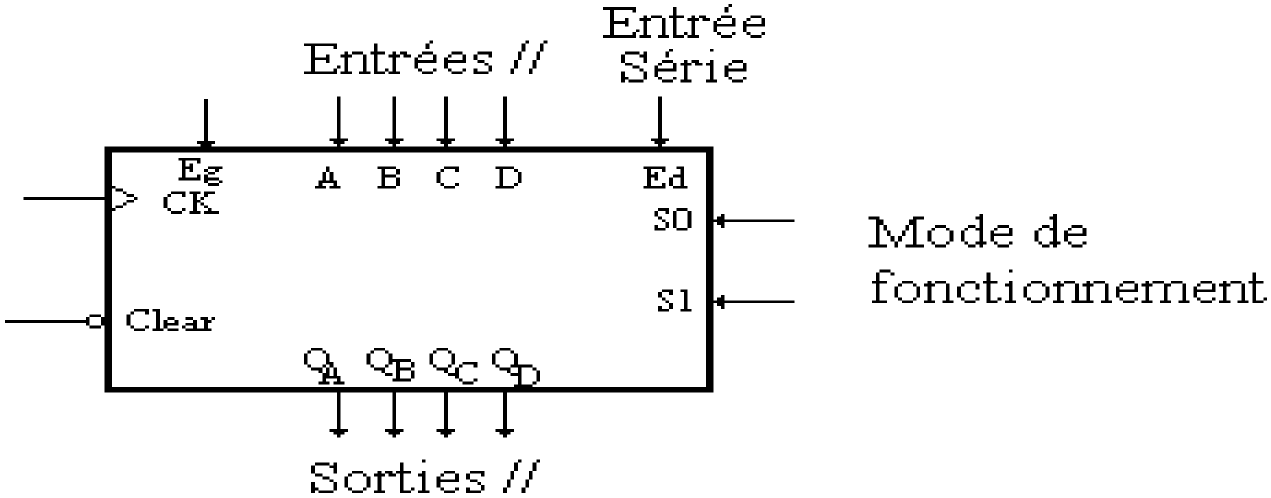
- C'est un registre qui effectue un décalage vers la gauche en répercutant la sortie de la derniére bascule vers l'entrée de la dernière bascule.
- Le décalage peut être un décalage droite (circulaire droite) ou gauche (circulaire gauche)





Registre programmable

- Il existe des registres qui permettent :
  - le décalage à droite (ou circulaire droite)
  - Le décalage à gauche (ou circulaire gauche)
  - Chargement parallèle.



Registre programmable (table de vérité)

h	S0	S1	QA+	QB+	QC+	QD+	Obs.
X	0	0	QA	QB	QC	QD	Mémoire
↑	0	1	Eg	QA	QB	QC	Décalage à droite
↑	1	0	QB	QC	QD	Ed	Décalage à gauche
↑	1	1	A	B	C	D	Chargement Synchrone