# Logique et Calculateur

Dr ABDELLAOUI Ghouti

École Supérieur des Sciences Appliquées Tlemcen

19 février 2023

# Plan du cour

# Information sur le module

Charge horaire : 1,5h cours + 1,5h TD + 3h TP
Calcule moyenne :

$$\begin{aligned} Test &= \frac{Test1+Test2+MiniProjet+assiduite}{4} \\ ControleContinue &= \frac{Test+Moy(TP)}{2} \\ MoyenneGenerale &= \frac{ControleContinue+Examen}{2} \end{aligned}$$

# Chapitre 1 : Systèmes combinatoire et séquentielle

# Présentation des systèmes Numérique

Figure – Information sonore

# Présentation des systèmes Numérique

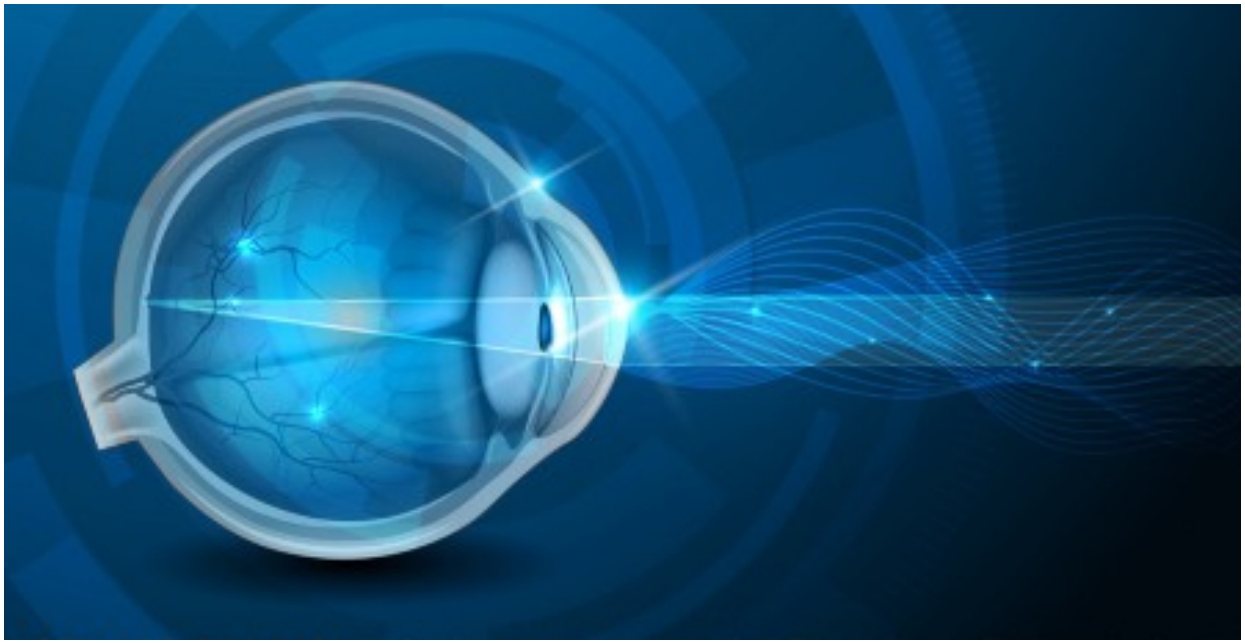


Figure – Information Visuel

# Présentation des systèmes Numérique

Figure – Information sensorielle

# Présentation des systèmes Numérique

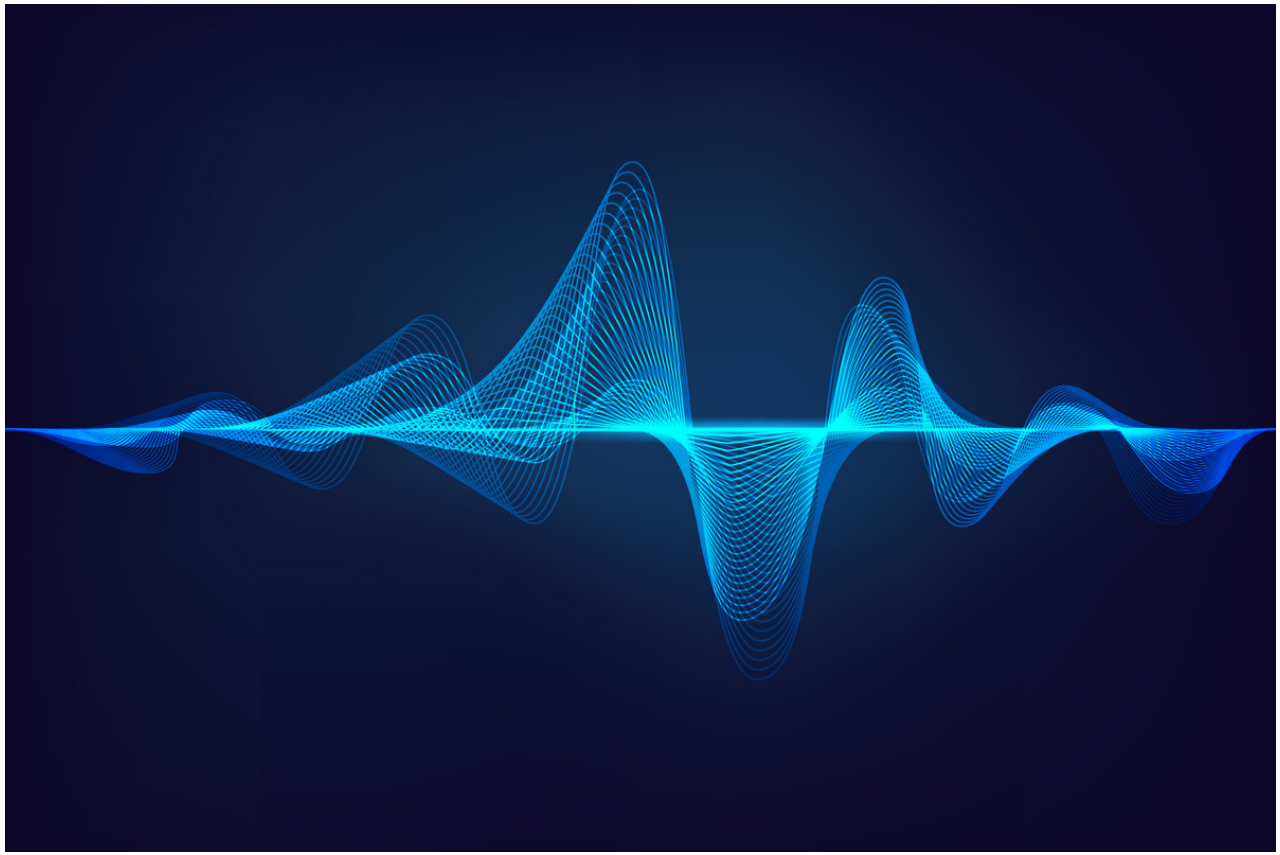


Figure – Analogique signal

# Présentation des systèmes Numérique

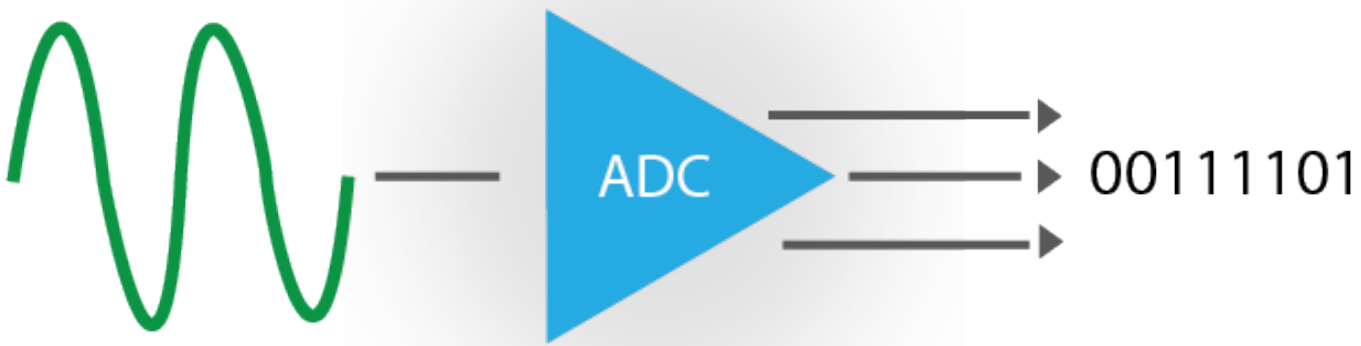



Figure – Analog/Digital Converter

# Présentation des systèmes Numérique

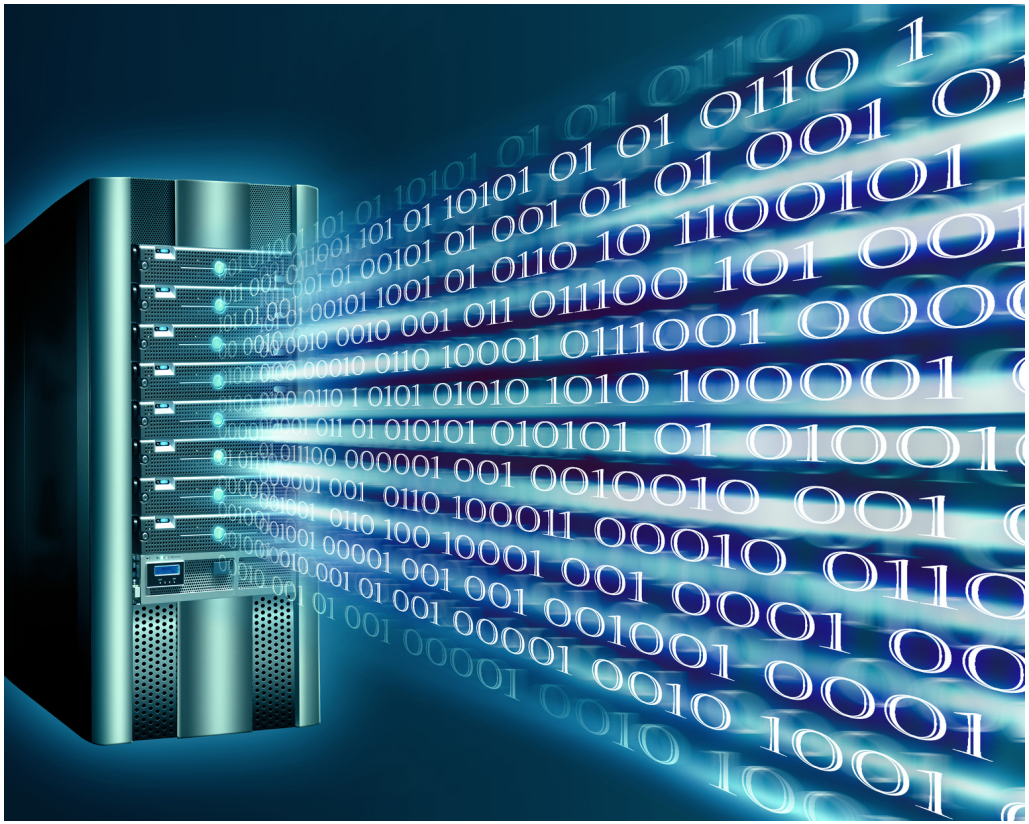


Figure – Digital Storage

# Présentation des systèmes Numérique

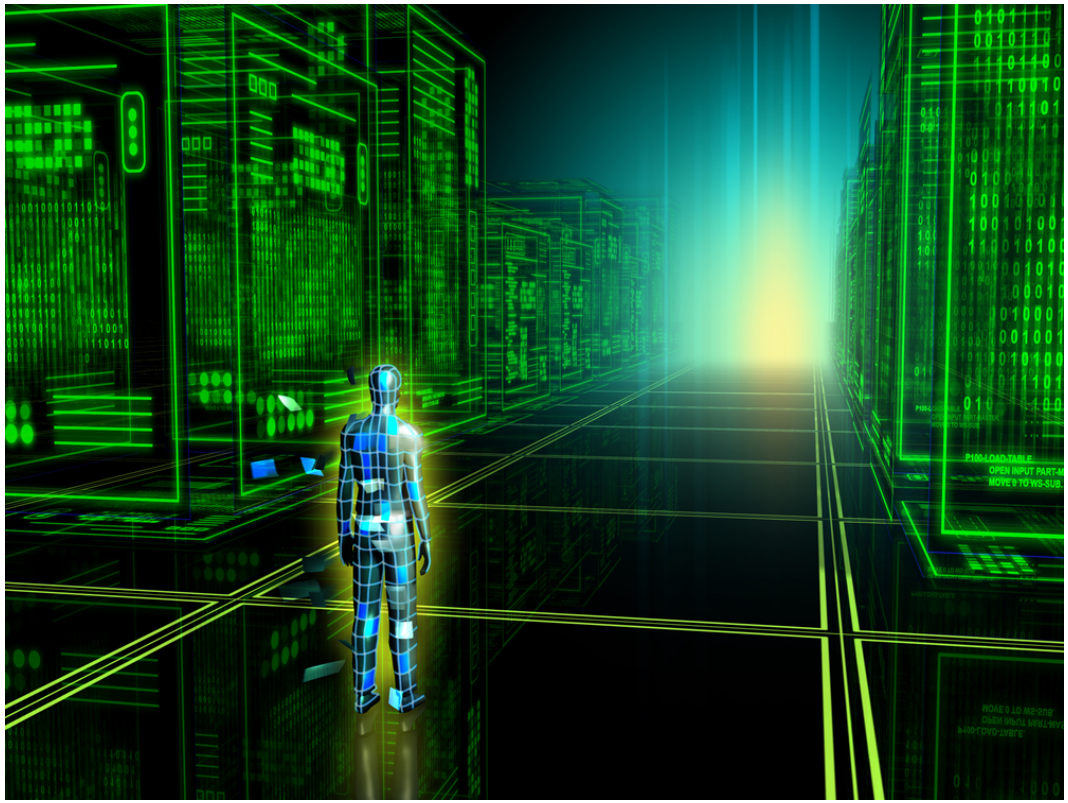


Figure – Digital World

# Présentation des systèmes Numérique

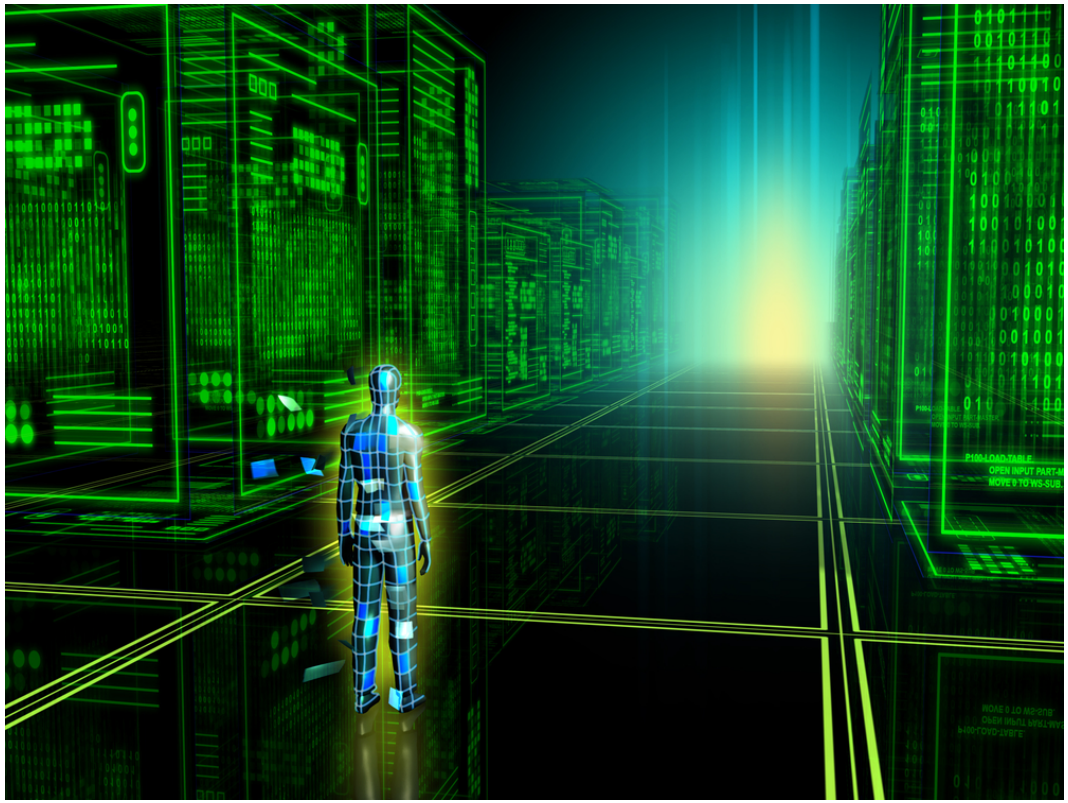


Figure – Digital World

# Présentation des systèmes Numérique

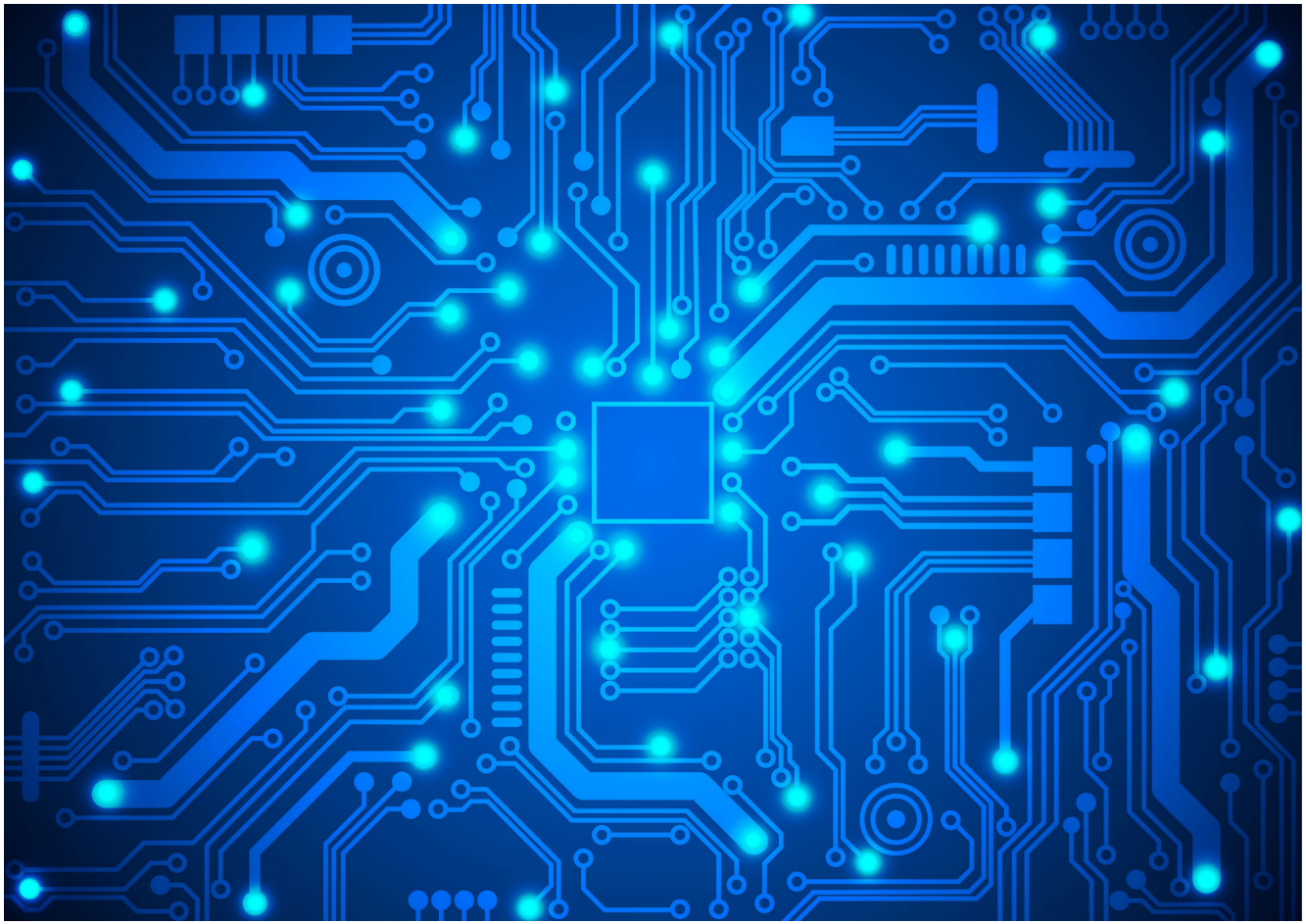


Figure – Digital Circuit

# Système de numérotation et codage des informations

**Définition**

*Pour qu'une information numérique soit traitée par un circuit, elle doit être mise sous forme adaptée à celui-ci. Pour cela Il faut choisir un système de numération de base B (B un nombre entier naturel $\geq 2$).*

De nombreux systèmes de numération sont utilisés en technologie numérique. Les plus utilisés sont les systèmes : Décimal (base 10), Binaire (base 2), Tétral (base4), Octal (base 8) et Hexadécimal (base 16).

# Tableau récapitulatif

| Décimal | Binaire | Tétral | Octal | Hexadécimal |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 | 2 |
| 3 | 11 | 3 | 3 | 3 |
| 4 | 100 | 10 | 4 | 4 |
| 5 | 101 | 11 | 5 | 5 |
| 6 | 110 | 12 | 6 | 6 |
| 7 | 111 | 13 | 7 | 7 |
| 8 | 1000 | 20 | 10 | 8 |
| 9 | 1001 | 21 | 11 | 9 |
| 10 | 1010 | 22 | 12 | A |
| 11 | 1011 | 23 | 13 | B |
| 12 | 1100 | 30 | 14 | C |
| 13 | 1101 | 31 | 15 | D |
| 14 | 1110 | 32 | 16 | E |

# Système de numérotation et codage des informations
## Changement de base

# Changement de base

Il s'agit de la conversion d'un nombre écrit dans une base $B_1$ à son équivalent dans une autre base $B_2$

Exemple :

$(1011101)_2 = 1*2^6+0*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0 = (93)_{10}$

$(231102)_4 = 2*4^5+3*4^4+1*4^3+1*4^2+0*4^1+2*4^0 = (2898)_{10}$

$(7452)_8 = 7*8^3+4*8^2+5*8^1+2*8^0 = (3882)_{10}$

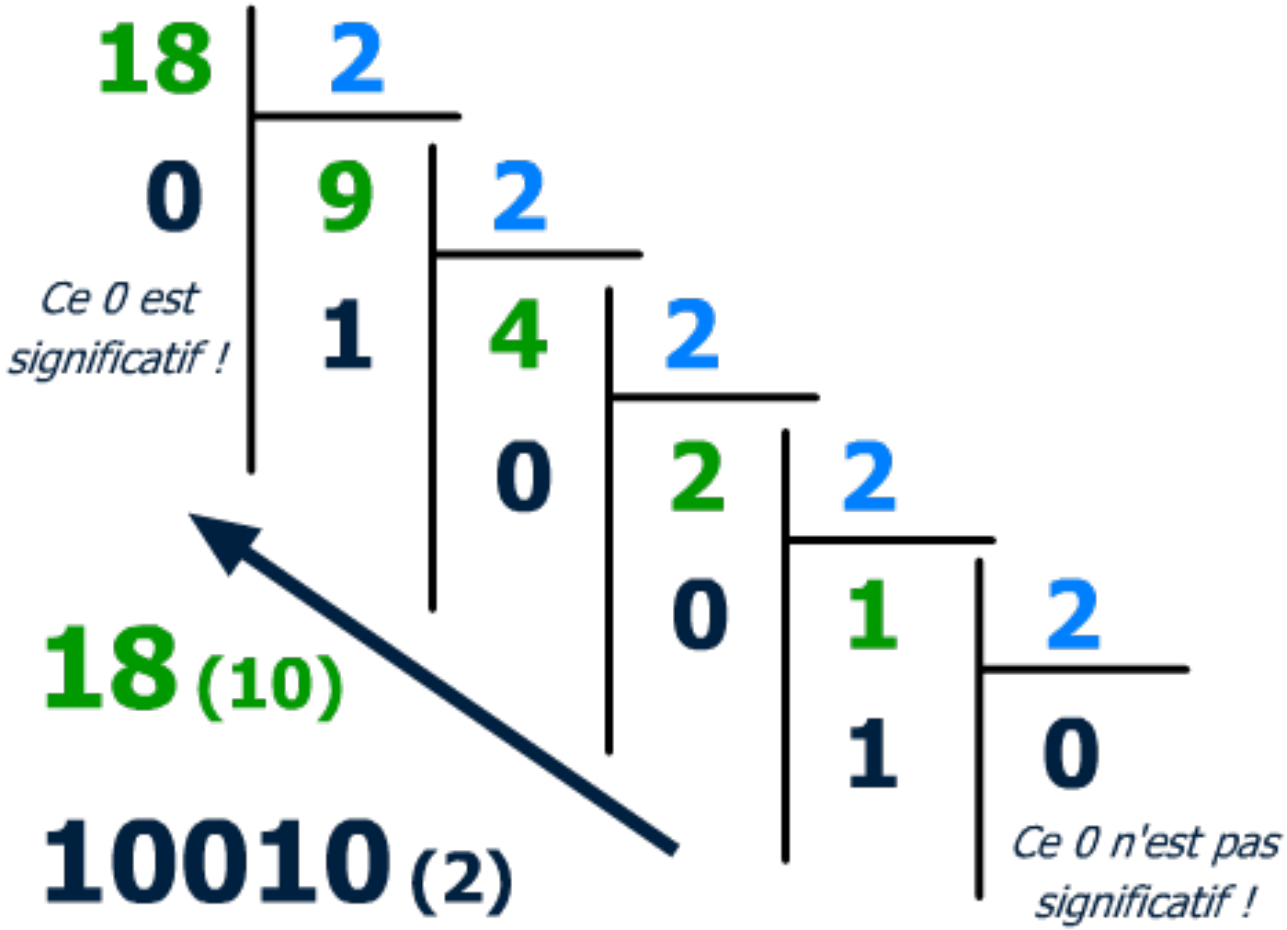
$(D7A)_16 = 13*16^2+7*16^1+10*16^0 = (3450)_{10}$

# Conversion d'un nombre décimal entier

Pour convertir un nombre décimal entier en un nombre de base **B** quelconque, il faut faire des divisions entières successives par la base **B** et conserver à chaque fois le reste de la division. On s'arrête lorsqu'on obtient un résultat inférieur à la base **B**. Le nombre recherche **N** dans la base **B** s'écrit de la gauche vers la droite en commençant par le dernier résultat allant jusqu'au premier reste.

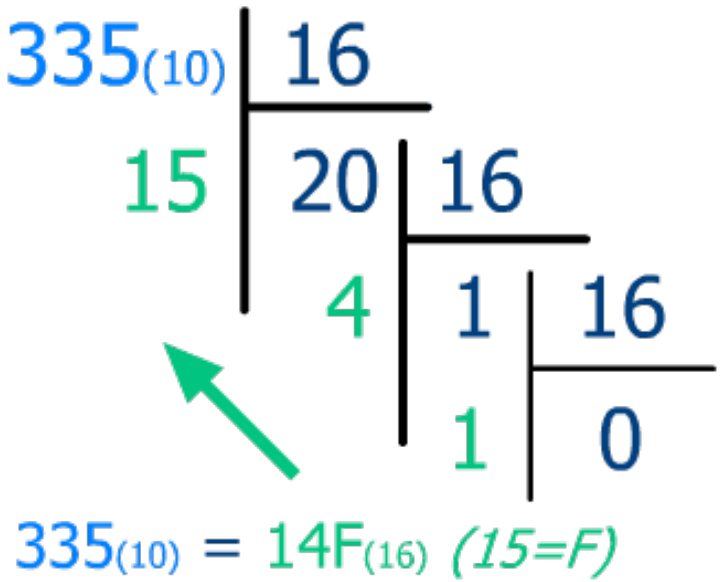
# Conversion d'un nombre décimal entier

$(18)_{10} = (?)_2$



# Conversion d'un nombre décimal entier

$(335)_{10} = (?)_{16}$




*Méthode intuitive, soustractions successives*

| 16^2 | 16^1 | 16^0 |
|---|---|---|
| 256 | 16 | 1 |
| 1 | 4 | F |

# Système de numérotation et codage des informations
## Opérations arithmétiques sur les nombres binaires

# Exemple d'addition en décimal

Avant de travailler en binaire, il est intéressant de se rappeler comment l'addition se réalise en calcul écrit. Considérons comme premier exemple $456 + 789$.

| | | | | | |
|---|---|---|---|---|---|
| | 1 | 1 | 1 | | « Reports |
| | | 4 | 5 | 6 | « Premier naturel |
| + | | 7 | 8 | 9 | « Second Naturel |
| | 1 | 2 | 4 | 5 | |

# Exemple d'addition en binaire

L'intérêt de cette approche est que l'addition avec des nombres en représentation binaire peut se faire de la même façon. Considérons quelques exemples avec des naturels représentés sur 4 bits.

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 0 | 1 | 0 | « Premier nombre binaire (2 en décimal) |
| + | 0 | 1 | 0 | 1 | « Deuxième nombre binaire (5 en décimal) |
| | 0 | 1 | 1 | 1 | « 7 en décimal |

# Exemple d'addition en binaire avec reports

Tout comme avec l'addition des naturels, le report est aussi possible avec le bit de poids fort. En toute généralité, lorsque l'on additionne deux quartets, la notation binaire du résultat devra parfois être stockée sur 5 bits et non 4. L'exemple ci-dessous illustre ce cas.

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | | | « Reports |
| | | 1 | 0 | 1 | 0 | « Premier nombre binaire $(10)_{10}$ |
| + | | 0 | 1 | 1 | 1 | « Deuxième nombre binaire $(7)_{10}$ |
| | 1 | 0 | 0 | 0 | 1 | « 17 en décimal |

# Représentation des nombres entiers

La solution présentée dans les diapos précédents permet de facilement représenter les nombres naturels qui sont nuls ou strictement positifs. En pratique, les ordinateurs doivent aussi pouvoir représenter les nombres négatifs et effectuer des soustractions. Différentes solutions sont envisageables pour représenter ces nombres entiers.

# Complément à un

Ce codage, fort simple, consiste à inverser la valeur de chaque bit composant une valeur binaire.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 5 en décimal |
| 1 | 0 | 1 | 0 | -5 en décimal |

# Complément à deux

Ce codage consiste à réaliser un complément à un de la valeur, puis d'ajouter 1 au résultat.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 5 en décimal |
| 1 | 0 | 1 | 0 | -5 Complément à un |
| 1 | 0 | 1 | 1 | -5 Complément à deux |

# Code de Gray ou binaire réfléchi

Ce codage permet de ne faire changer qu'un seul bit à la fois quand un nombre est augmenté d'une unité. Le nom du code vient de l'ingénieur américain Frank Gray qui déposa un brevet sur ce code en 1953.

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 1 | 0 | 1 |
| 7 | 0 | 1 | 0 | 0 |

Pour passer d'une ligne à la suivante, on inverse le bit le plus à droite possible conduisant à un nombre nouveau.

# Code de Gray ou binaire réfléchi

Ce code est surtout utilisé pour des capteurs de positions, par exemple sur des règles optiques. En effet, si on utilise le code binaire standard, lors du passage de la position un (01) à deux (10) – permutation simultanée de 2 bits – il y a risque de passage transitoire par trois (11) ou zéro (00), ce qu'évite le code de Gray.
On remarquera que le passage du maximum (sept sur 3 bits) à zéro se fait également en ne modifiant qu'un seul bit. Ceci permet par exemple d'encoder un angle, comme la direction d'une girouette : 0=Nord, 1=Nord-Est, 2=Est, ... 7=Nord-Ouest. Le passage de Nord-Ouest à Nord se fait également sans problème en ne changeant qu'un seul bit.

# Décimal codé binaire

Ce codage consiste à représenter chacun des chiffres de la numérotation décimale sur 4 bits :

$$2021 = \quad 0010 \quad 0000 \quad 0010 \quad 0001\,;$$

Avec n bits (n multiple de 4), il est possible de représenter les nombres entre 0 et $10^{\frac{n}{4}} - 1$ Soit approximativement entre 0 et $1.778^n - 1$.
Le BCD est un code redondant, en effet certaines combinaisons ne sont pas utilisées (comme 1111 par exemple).

# Application codage

## Théorie de l'information

En théorie de l'information, on peut utiliser le bit comme unité de mesure de l'information. La théorie elle-même est indifférente à la représentation des grandeurs qu'elle utilise.

## Logique

La logique classique est une logique bivalente : une proposition est soit vraie, soit fausse. Il est donc possible de représenter la vérité d'une proposition par un chiffre binaire 0 ou 1. On peut par exemple modéliser les opérations de l'arithmétique binaire à l'aide de l'algèbre de Boole.

L'algèbre de Boole représente un cas très particulier d'usage des probabilités ne faisant intervenir que les seules valeurs de vérité 0 et 1.

# Circuits combinatoires

# Algèbre do BOOLE

L'algèbre de Boole, ou calcul booléen, est la partie des mathématiques, de la logique et de l'électronique qui s'intéresse aux opérations et aux fonctions sur les variables logiques. Plus spécifiquement, l'algèbre booléenne permet d'utiliser des techniques algébriques pour traiter les expressions à deux valeurs du calcul des propositions. Elle fut initiée par le mathématicien britannique George Boole.

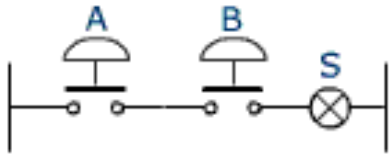
# Algèbre do BOOLE

On appelle B l'ensemble constitué de deux éléments appelés valeurs de vérité VRAI, FAUX. Cet ensemble est aussi noté

$B = \{1, 0\}$

Sur cet ensemble on peut définir deux lois (ou opérations ou foncteurs), les lois ET et OU et une transformation appelée complémentaire, inversion ou contraire.

# La fonction ET



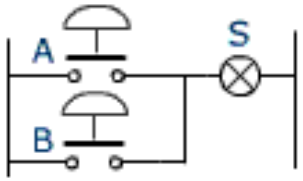

La lampe s'allume si on active simultanément les contacts A et B
(S = 1) si (A = 1) ET (B = 1)

| A | B | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

L'opérateur ET est représenté dans l'équation logique par un point. Ce signe convient parfaitement puisque la fonction ET donne le même résultat qu'une multiplication.

$$S = A \cdot B$$

# La fonction OU




La lampe s'allume si on active le contact A ou le contact B
(S = 1 ) si ( A = 1) OU ( B = 1)

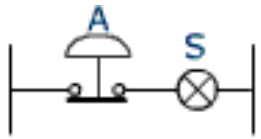
| A | B | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

L'opérateur OU est représenté dans l'équation logique par un signe plus.

$$S = A + B$$

# La fonction NON

Les contacts que nous avons utilisés jusqu'ici, sont des contacts "normalement ouverts". Quand le bouton poussoir est relâché ( quand A = 0) le courant ne passe pas.
Nous utilisons maintenant un contact "normalement fermé" pour illustrer la fonction NON. Au repos, le courant passe mais il se coupe quand le contact est activé ( quand A = 1).



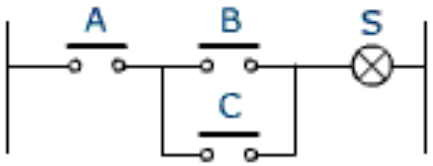

$$S = \overline{A}$$

# Combinaisons de fonctions logiques

Les trois fonctions de base que nous venons de voir se combinent de multiples façons. A chaque schéma imaginable correspond une équation. La correspondance entre un schéma et une fonction logique est systématique.

- Des contacts en parallèle correspondent à la fonction OU.
- Des contacts en série correspondent à la fonction ET
- Un contact normalement fermé représente la fonction NON

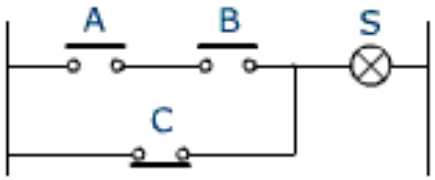
# Exemple fonctions logique




| A | B | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$S = A.(B + C)$$

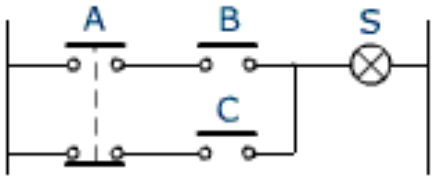
# Exemple fonctions logique




| A | B | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

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

# Exemple fonctions logique




| A | B | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 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 | 1 |

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

# Circuits combinatoires
## Propriétés de l'algèbre de bool

# Propriétés de l'algèbre de bool

Les propriétés de l'algèbre de bool sont sous forme de postulats et théorèmes qui nous aident à manipuler et simplifier les fonctions logiques.

# Les postulats de huntington

Postulat 1 : Fermeture par rapport aux opérateurs OU et AND
Un ensemble S est dit fermé par rapport à un opérateur si pour chaque paire de variables $\in$ S l'opérateur donne un résultat qui appartient à S.
$\forall x, y \in B, x + y \in B, B = 0, 1$
$\forall x, y \in Bx.y \in B, B = 0, 1$

# Les postulats de huntington

Postulat 2 : Loi de l'identité (éléments neutres)
Le 0 est l'élément d'identité du OU : $x + 0 = 0 + x = x$
Le 1 est l'élément d'identité du ET : $x.1 = 1.x = x$

# Les postulats de huntington

Postulat 3 : Loi de commutativité par rapport aux OU et AND

$x + y = y + x$ et $x.y = y.x$

Postulat 4 : Loi distribution par rapport aux OU et AND

$x + (y.z) = (x + y).(x + z)$ et $x.(y + z) = (x.y) + (x.z)$

# Les postulats de huntington

Postulat 5 : Loi de complémentarité $\forall x \in B \ \exists \overline{x} \in B$ tel que $x + \overline{x} = 1$ et $x.\overline{x} = 0$

# Les théorèmes

Théorème 1 : Le complément de x est unique

$\forall x \in B$, si $x = 0$ alors $\overline{x} = 1$

$\forall x \in B$, si $x = 1$ alors $\overline{x} = 0$

Théorème 2 : Loi d'idempotence

$x + x = x$ et $x.x = x$

Théorème 3 : Loi des éléments dominants

$x + 1 = 1$ et $x.0 = 0$

# Les théorèmes

Théorème 4 : Loi d'involution

$\overline{\overline{x}} = x$

Théorème 5 : Loi d'absorption

$x + x.y = x$ , $x.(x + y) = x$

Théorème 6 : Loi du consensus

$x + \overline{x}.y = x + y$, $x.(\overline{x} + y) = x.y$

# Les théorèmes

Théorème 8 : Loi de De Morgan

$\overline{x+y} = \overline{x}.\overline{y}$, $\overline{x.y} = \overline{x} + \overline{y}$

Théorème 9 : Loi de De Morgan généralisée

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

Théorème 10 : Loi du consensus généralisée

$x.y + \overline{x}.z + y.z = x.y + \overline{x}.z$, $(x+y).(\overline{x}+z).(y+z) = (x+y).(\overline{x}+z)$

# Circuits combinatoires
## Représentation des fonctions logiques

# Par son expression logique :

C'est une combinaison des variables de la fonction via les opérateurs de base de l'algèbre de Boole.
Exemple : Fonction f de trois variables x, y et z.
$f(x, y, z) = x.y + \overline{y}.z + x.\overline{z}$

# Par sa table de vérité :

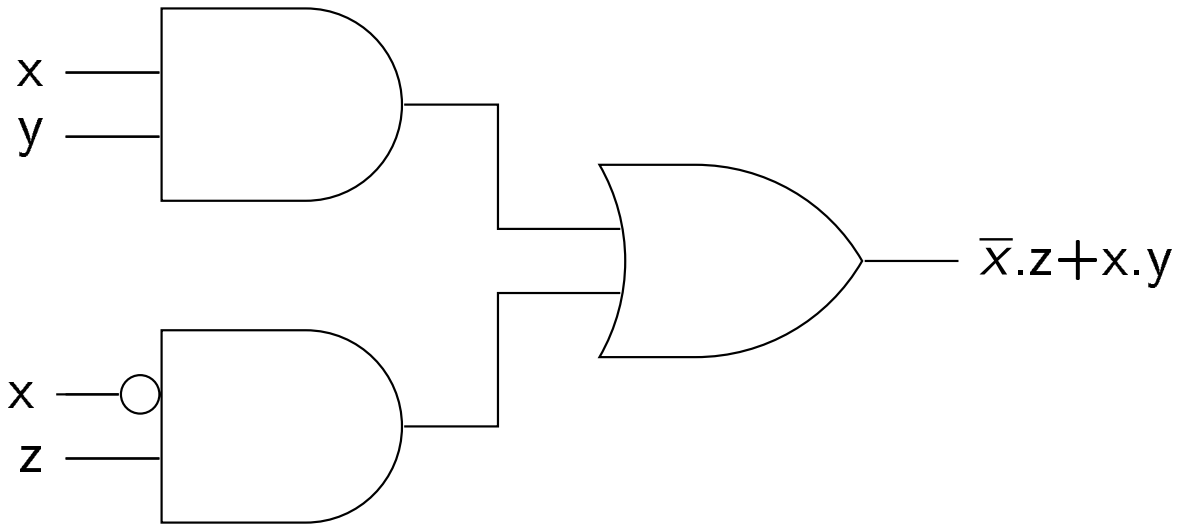
La Table définit la valeur de la fonction pour chaque combinaison des valeurs possible en entrée.
Exemple : Fonction f de trois variables x, y et z.
$f(x, y, z) = x.y + \overline{y}.z + x.\overline{z}$

| x y z | $\overline{y}$ | $\overline{z}$ | x y | $\overline{y}z$ | $x\overline{z}$ | f(x,y,z) |
|---|---|---|---|---|---|---|
| 0 0 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 0 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 1 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 1 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 0 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 0 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 1 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 1 1 | 0 | 0 | 1 | 0 | 0 | 1 |

# Par son circuit logique :

Le circuit logique est une schéma graphique de la fonction qui représente la forme pseudo électrique qui peut être ensuite convertit en un circuit électronique puis industrialiser

Chaque opérateur logique possède un circuit équivalent :



# Formes canoniques d'une fonction

Une fonction logique est dite sous forme canonique si dans tous ces termes tous les variables existent soit sous forme directe(sans bare) ou bien sou forme complémentaire.

- **Min terme** : Groupe de n variables (pouvant être complémentaires) liées par des ET.(Exemple : $x.y.\overline{z}$).
- **Max terme** : Groupe de n variables (pouvant être complémentaires) liées par des OU.(Exemple :$x + \overline{y} + z$).

# Circuits combinatoires
## Simplification des fonction logiques

# Minimisation par tableau de karnaugh

La méthode de tableau de karnaugh est une méthode simple et directe pour simplifier les fonctions logiques. Cette méthode a été proposée en premier temps par VEITCH et ensuite développée par kanaugh.

# Tableau de karnaugh à 2 variables :

Un tableau de karnaugh à 2 variables est représenté par 4 min termes (4 cases) :

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

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A/B | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 1 |

# Tableau de karnaugh à 3 variables

Un tableau de karnaugh à 3 variables est représenté par 8 min termes (8 cases) :

| A/BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $\overline{A}.\overline{B}.\overline{C}$ | $\overline{A}.\overline{B}.C$ | $\overline{A}.B.C$ | $\overline{A}.B.\overline{C}$ |
| 1 | $A.\overline{B}.\overline{C}$ | $A.\overline{B}.C$ | $A.B.C$ | $A.B.\overline{C}$ |

| A/BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 1 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |

# Tableau de karnaugh à 4 variables :

Un tableau de karnaugh à 4 variables est représenté par 16 min termes (16 cases).

| AB/CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 01 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |
| 11 | $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| 10 | $m_8$ | $m_9$ | $m_{11}$ | $m_{10}$ |

# Élimination des variables superflus

Une variable superflu est une variable qui change d'état d'une case à une case adjacente qui valent 1.

| AB/CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 2 | 3 | 4 |
| 01 | 5 | 6 | 7 | 8 |
| 11 | 9 | 10 | 11 | 12 |
| 10 | 13 | 14 | 15 | 16 |

# Élimination des variables superflus

Les cases adjacentes sont :

- (1,2) ;(2,3) ;(3,4) ;(5,6) ;(6,7) ;(7,8) ;(9,10) ;(10,11) ; (11,12) ;(13,14) ;(14,15) ;(15,16).
- (1,5) ;(5,9) ;(9,13) ;(2,6) ;(6,10) ;(10,14) ;(3,7) ;(7,11) ; (11,15) ;(4,8) ;(8,12) ;(12,16).
- (1,2,3,4) ;(5,6,7,8) ;(9,10,11,12) ;(13,14,15,16).
- (1,5,9,13) ;(2,6,10,14) ;(3,7,11,15) ;(4,8,12,16).
- (1,4,13,16).
- (1,2,13,14) ;(2,3,14,15) ;(3,4,15,16).
- (1,2,5,6) ;(2,3,6,7) ;(3,4,7,8),(5,6,9,10) ;... ;(11,12,15,16).
- (1,2,5,6,9,10,13,14) ;(2,3,6,7,10,11,14,15) ;(3,4,7,8,11,12,15,16).
- .....

# Élimination des variables superflus

Méthode Le tableau de Karnaugh

1. On détermine le nombre de variables d'entrée afin de connaître la taille des tableaux.
2. On détermine le nombre de variables de sortie afin de définir le nombre de tableaux à effectuer.
3. Affecter aux différents produits de l'équation non simplifiée une case du tableau en respectant le code Gray.
4. Introduire la fonction logique dans le tableau en positionnant à « 1 » les cases qui valident la fonction de sortie.
5. Effectuer les groupements de cases adjacentes.
6. Sortir la fonction simplifiée en éliminant la ou les variables d'entrée qui changent d'état.

# Notion de cas indéterminés

La logique ternaire, ou logique 3 états, est une branche du calcul des propositions qui étend l'algèbre de Boole, en considérant, en plus des états VRAI et FAUX, l'état indéterminé (inconnu) et on le note par X.

# Notion de cas indéterminés
## Table de vérité

| A | B | A+B | A.B | $\overline{A}$ | $A \Rightarrow B$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | X | X | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| X | 0 | X | 0 | X | X |
| X | X | X | X | X | X |
| X | 1 | 1 | X | X | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | X | 1 | X | 0 | X |
| 1 | 1 | 1 | 1 | 0 | 1 |

# Principe de la méthode

L'algorithme se déroule en deux étapes. Pour faciliter la mise en œuvre de la méthode, la fonction logique doit être exprimée soit sous forme tabulaire (table de vérité, tableau de Karnaugh), soit sous la forme suivantes :
$f(A_0, A_1, ..., A_{N-1}) = R(m_0, m_1, ..., m_{k-1}) + \phi(N_0, N_1, ..., N_{p-1})$
Où :

- $R(m_0, m_1, ..., m_{k-1})$ sont les k valeurs corresponds au k cas où $f$ vaux 1.
- $\phi(N_0, N_1, ..., N_{p-1})$ sont les p valeurs corresponds au p cas où $f$,vaux X.

Exemple : L'expression d'une porte NAND sous cette forme serait : $f_{NAND}(a, b) = R(0, 1, 2)$.

# Principe de la méthode
## Première étape

Elle correspond à la recherche de l'ensemble des termes générateurs. Un terme est générateur s'il ne peut être combiné avec aucun autre terme pour donner un terme plus simple.

**Exemple** : Dans le cas de la porte NAND, la fonction vaut 1 lorsque (ab) vaut 00, 01 ou 10. Le terme 00 n'est pas un terme générateur car combiné avec 01, il donne naissance à un terme plus simple 0x. En revanche, 0x ne peut être combiné avec un autre terme, c'est donc un terme générateur. De la même manière x0 est un autre terme générateur.

Pour trouver les termes générateur, on utilise les valeurs $m_0, m_1, ..., m_{k-1}$ et $N_0, N_1, ..., N_{p-1}$ car, comme dans un tableau de Karnaugh, les termes indéterminés peuvent conduire à des simplification.

# Principe de la méthode
## Deuxième étape

Elle correspond à l'élimination, parmi les termes générateurs, des termes redondants. Pour celà, on identifie les termes générateurs à partir desquels pour chaque $m_0, m_1, ..., m_{k-1}$ peut-être écrit et on élimine ceux qui sont en trop.

**Exemple** : Dans le cas de la porte NAND, le terme 00 et 01 peuvent être écrit avec le terme générateur 0x mais celui-ci n'est pas utilisable pour écrire 10. L'utilisation du second générateur x0 est indispensable. Il n'y a pas de terme redondant.

# Exemple d'application

Considérons la table de vérité suivante :

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

# Exemple d'application

L'exemple n'est pas totalement fortuit. Il s'agit en fait de la table de vérité servant au codage de la barre située en bas à gauche d'un afficheur 7 segments (allumée quand on souhaite afficher 0, 2, 6 ou 8, éteinte quand on a comme code 1, 3, 4, 5, 7 ou 9 et indéterminée pour les codes supérieurs à 10). L'expression de S s'écrit donc :

$$S = R(0, 2, 6, 8) + \phi(10, 11, 12, 13, 14, 15)$$

# Exemple d'application

## Etape $n°1$ : Identification des termes générateurs

Pour identifier les termes générateurs, on remplit un premier tableau de la manière suivantes :
Dans la première colonne, on écrit tous les termes portant la fonction à 1 ou à une forme indéterminée. Les termes sont triés en fonction du nombre de 1 qu'ils contiennent.

| **Colonne 0** |
|---|
| 0000 |
| 0010 |
| 1000 |
| 0110 |
| 1010 |
| 1100 |
| 1011 |
| 1101 |
| 1110 |
| 1111 |

# Exemple d'application

Dans cette liste, le terme dans le groupe 0 a zéro 1, les termes du groupe 1 ont un 1, le groupe 2 a deux 1 et enfin le groupe 3 a trois 1. Deux termes peuvent être combinés si la différence n'est qu'une variable.

- 0000 peut être recombiné avec 0010 pour former le terme 00x0.
- 0000 peut être recombiné avec 1000 pour former le terme x000.
- 0010 peut être recombiné avec 1010 pour former le terme x010.
- etc.

# Exemple d'application

| Colonne 0 | Colonne 1 |
|---|---|
| 0000 | 00x0 |
| 0010 | x000 |
| 1000 | 0x10 |
| 0110 | x010 |
| 1010 | 10x0 |
| 1100 | 1x00 |
| 1011 | x110 |
| 1101 | 101x |
| 1110 | 1x10 |
| 1111 | 110x |
| | 11x0 |
| | 111x |
| | 11x1 |
| | 1x11 |

# Exemple d'application

- 00x0 peut être recombiné avec 10x0 pour former le terme x0x0.
- x000 peut être recombiné avec x010 pour former le terme x0x0.
- 0x10 peut être recombiné avec 1x10 pour former le terme xx10.
- etc ...
- x110 ne peut être recombiné avec aucun autre terme de la colonne, c'est donc un terme générateur.

# Exemple d'application

| Colonne 0 | Colonne 1 | Colonne 2 |
|---|---|---|
| 0000 | 00x0 | x0x0 |
| 0010 | x000 | xx10 |
| 1000 | 0x10 | 1xx0 |
| 0110 | x010 | 1x1x |
| 1010 | 10x0 | 11xx |
| 1100 | 1x00 | |
| 1011 | x110 | |
| 1101 | 101x | |
| 1110 | 1x10 | |
| 1111 | 110x | |
| | 11x0 | |
| | 111x | |
| | 11x1 | |
| | 1x11 | |

# Exemple d'application

On réitère ce mécanisme jusqu'à ce qu'on ne puisse plus combiner de termes. Dans ce cas, tous les termes de la colonne 2 sont générateur, mais dans d'autre cas de figure, on pourrait trouver une colonne 3 avec des termes portant 3x. On a donc identifié 6 termes générateur :

x110, x0x0, xx10, 1xx0, 1x1x et 11xx.

# Exemple d'application

## Etape $n°2$ : Suppression des redondances

Pour identifier les termes redondants, on remplis un second tableau tel que en colonne on met tous les termes générateurs et on ligne on met que les termes de $R$ là ou la fonction vaux 1.

| | x110 | x0x0 | xx10 | 1xx0 | 1x1x | 11xx |
|---|---|---|---|---|---|---|
| 0000 | | • | | | | |
| 0010 | | • | • | | | |
| 0110 | • | | • | | | |
| 1000 | | | | • | | |

- Les termes x110 et 1xx0 sont à conserver car ils sont indispensables pour exprimer respectivement 0110 et 1000.
- x0x0 permet d'exprimer 0000 et 0010.
- xx10 est un terme redondant, il n'apporte pas d'information complémentaire car il permet de coder des termes qui peuvent l'être par des générateurs déjà identifiés.
- 1x1x et 11xx ne servent pas à exprimer des termes de $R$.

# Exemple d'application

Les termes finaux sont :

**x110 1xx0 x0x0**

Ce qui donne le résultat final :

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

# Circuits combinatoires
## Circuits intégrés logiques

# Circuits intégrés logiques

## Principaux circuits MSI combinatoires

- Encodeur
- Décodeur
- Multiplexeur
- Démultiplexeur

## Les circuits de traitement ou de calcul

- Un circuit de décalage
- Additionneur
- Unité arithmétique et logique

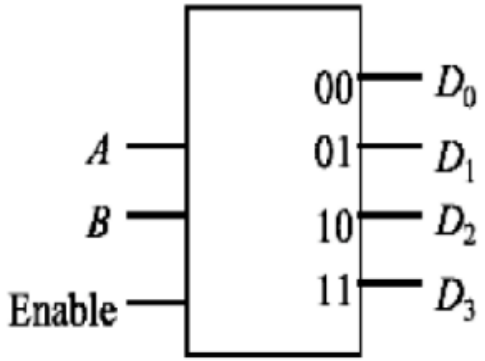
# Encodeur

- Active un code selon l'une des X entrées actives
- $2^n$ (en général) entrées, 1 entrée active (valeur 1), les autres sont toutes désactivées (valeur 0)
- Code en sortie : sur n bits
- Encodeur sur 3 bits

| $E_0$ | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ | $S_0$ | $S_1$ | $S_3$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | 0 | 0 | 0 |
| | 1 | | | | | | | 0 | 0 | 1 |
| | | 1 | | | | | | 0 | 1 | 0 |
| | | | 1 | | | | | 0 | 1 | 1 |
| | | | | 1 | | | | 1 | 0 | 0 |
| | | | | | 1 | | | 1 | 0 | 1 |
| | | | | | | 1 | | 1 | 1 | 0 |
| | | | | | | | 1 | 1 | 1 | 1 |

# Décodeur

- Active une des X sorties selon un code
- Code : sur n bits
- Nombre de sorties : $2^n$ (en général)




| Enable = 1 | | | | | |
|---|---|---|---|---|---|
| A | B | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

| Enable = 0 | | | | | |
|---|---|---|---|---|---|
| A | B | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |

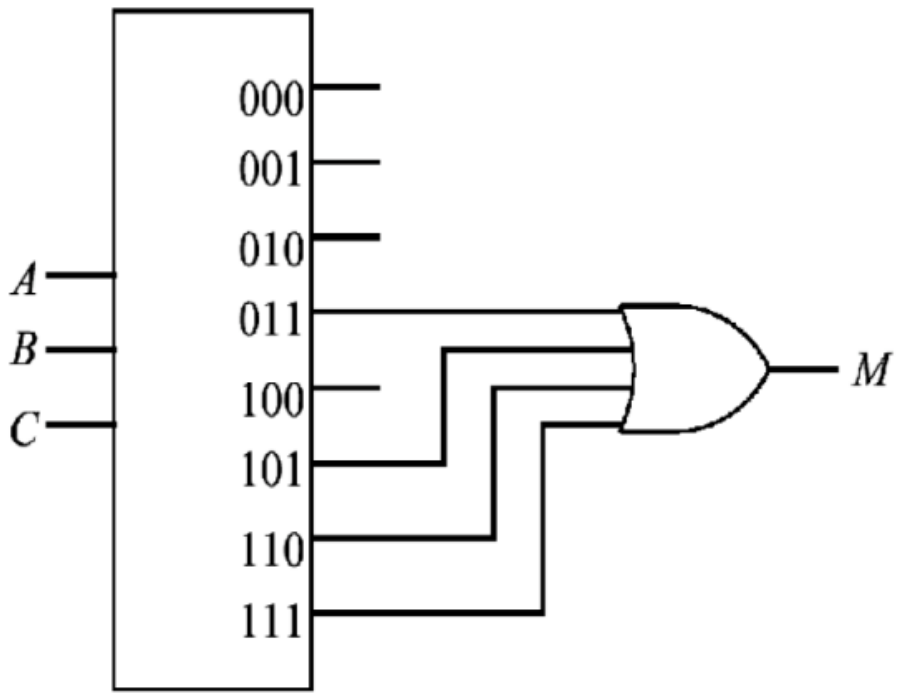
$$D_0 = \overline{A}.\overline{B} \ , \ D_1 = \overline{A}.B \ , \ D_2 = A.\overline{B} \ , \ D_3 = A.B$$

# Décodeur

Fonction de majorité : **La fonction vaut 1 si la majorité des variables vaut 1.**

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

| A | B | C | M |
|---|---|---|---|
| 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 |

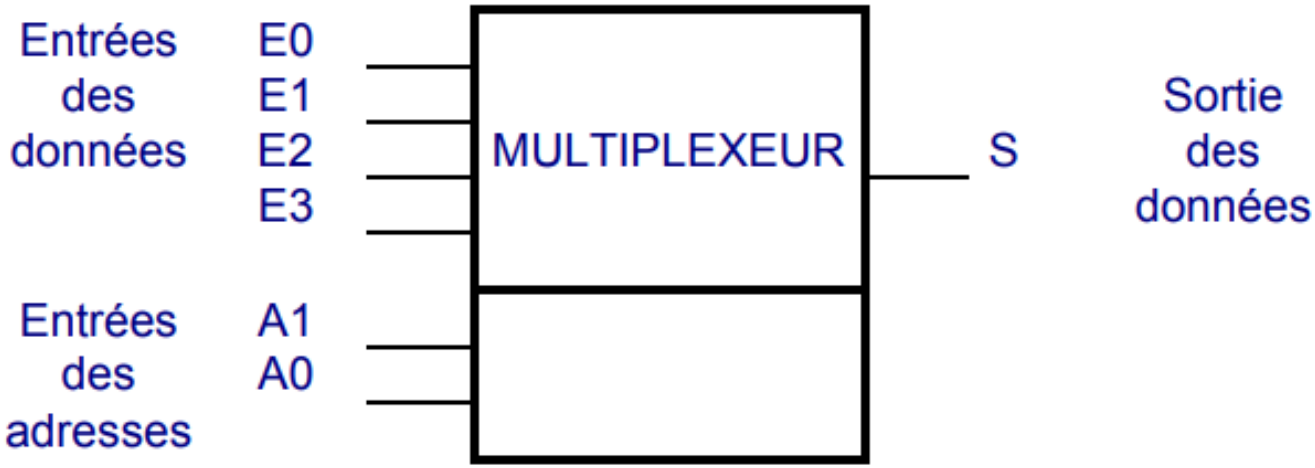


# Multiplexeur

Un multiplexeur ou sélecteur de données est un commutateur qui va pouvoir, à l'aide de n bits d'adresse, sélectionner une de ses $2^n$ entrées et la mettre en communication avec sa sortie.

# Multiplexeur

**Exemple** : Le schéma ci-dessous donne une image d'un multiplexeur 4 voies ($E_3$ à $E_0$) vers une (S) sélectables à l'aide des bits d'adresse $A_1$ et $A_0$.



# Multiplexeur

- Lorsque $A_1, A_0 = 00$ si $E_0 = 0 \Rightarrow S = 0$, si $E_0 = 1 \Rightarrow S = 1$ et ceci quelles que soient les entrées $E_1, E_2, E_3$
- Lorsque $A_1, A_0 = 01$ si $E_1 = 0 \Rightarrow S = 0$, si $E_1 = 1 \Rightarrow S = 1$ et ceci quelles que soient les entrées $E_0, E_2, E_3$.

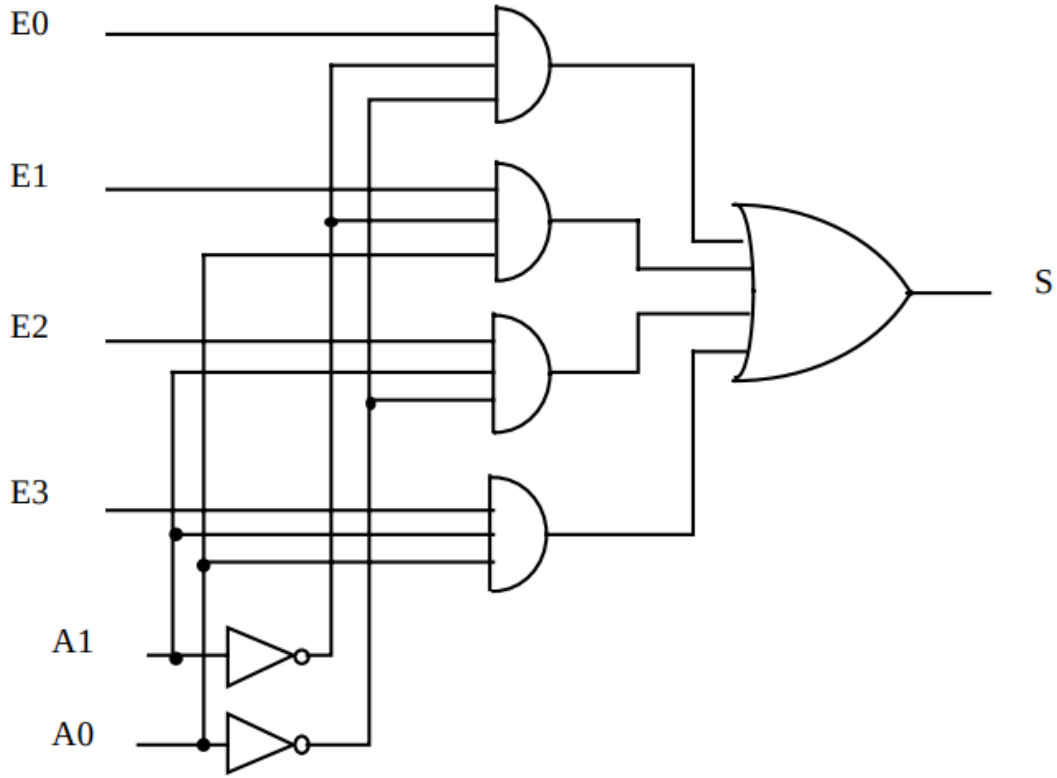
Donc,

- Lorsque $A_1, A_0 = 00$ $S = E_0$ on peut en déduire dans ces conditions $S = \overline{A}.\overline{B}.E_0$
- Lorsque $A_1, A_0 = 01$ $S = E_1$ on peut en déduire dans ces conditions $S = \overline{A}.B.E_1$
- Lorsque $A_1, A_0 = 10$ $S = E_2$ on peut en déduire dans ces conditions $S = A.\overline{B}.E_2$
- Lorsque $A_1, A_0 = 11$ $S = E_3$ on peut en déduire dans ces conditions $S = A.B.E_3$

$$S = \overline{A}.\overline{B}.E_0 + \overline{A}.B.E_1 + A.\overline{B}.E_2 + A.B.E_3$$

# Multiplexeur

$$S = \overline{A}.\overline{B}.E_0 + \overline{A}.B.E_1 + A.\overline{B}.E_2 + A.B.E_3$$



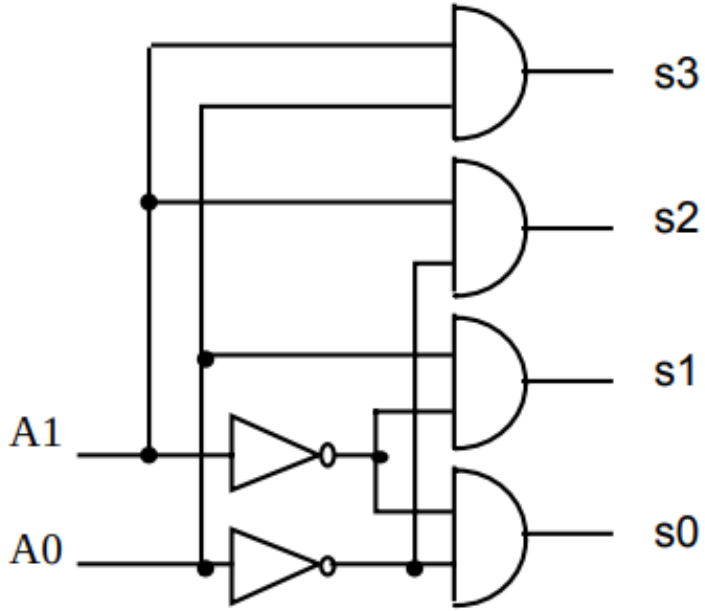
# Multiplexeur

## Multiplexeur avec décodeur d'adresses

Table de vérité

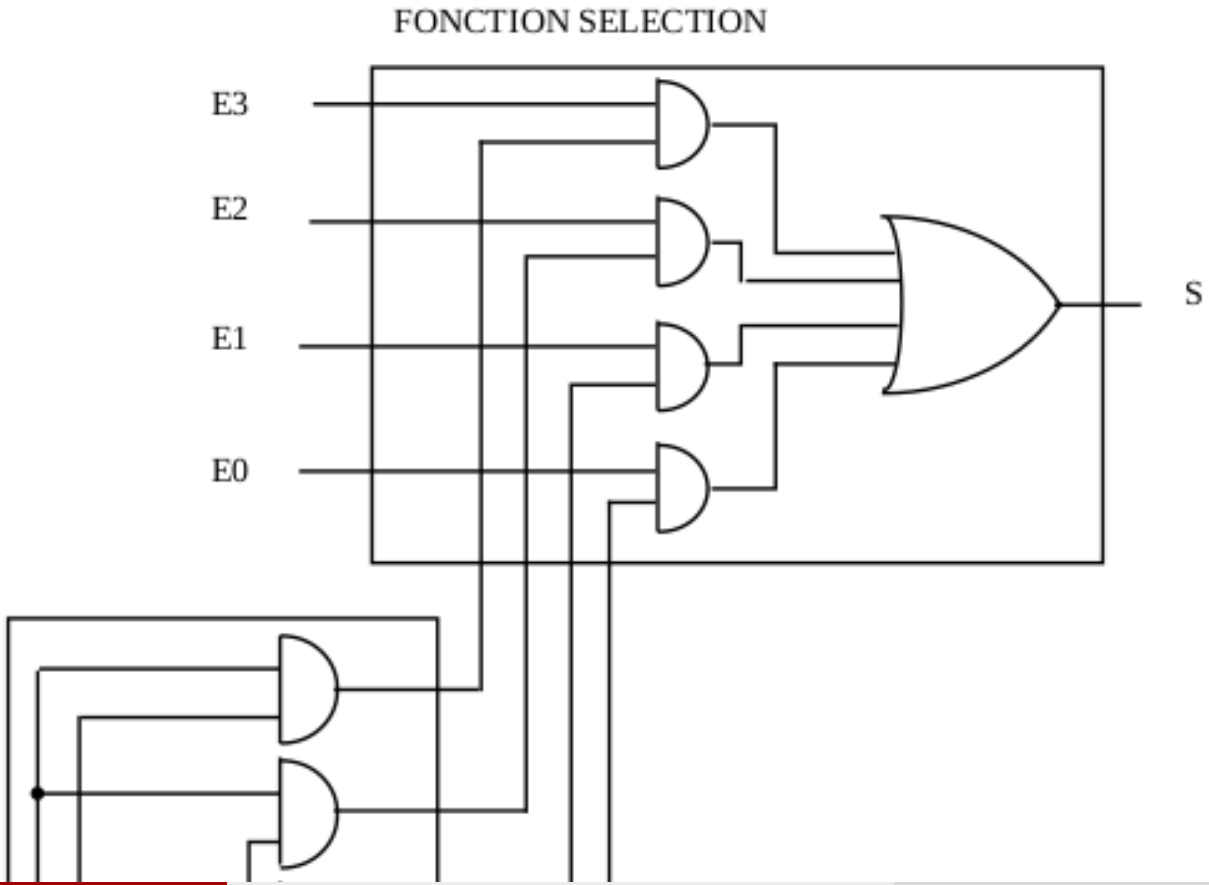
| Adresse | | Sélection | | | |
|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $S_3$ | $S_2$ | $S_1$ | $S_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

$$S = S_0.E_0 + S_1.E_1 + S_2.E_2 + S_3.E_3$$



# Multiplexeur

Le schéma ci-dessous fait apparaître le multiplexeur réalisé à l'aide de deux sous-ensembles, d'une part le décodeur d'adresses et d'autre part le sélecteur d'entrée dont les portes ET qui assurent la sélection sont validées par les sorties du décodeur



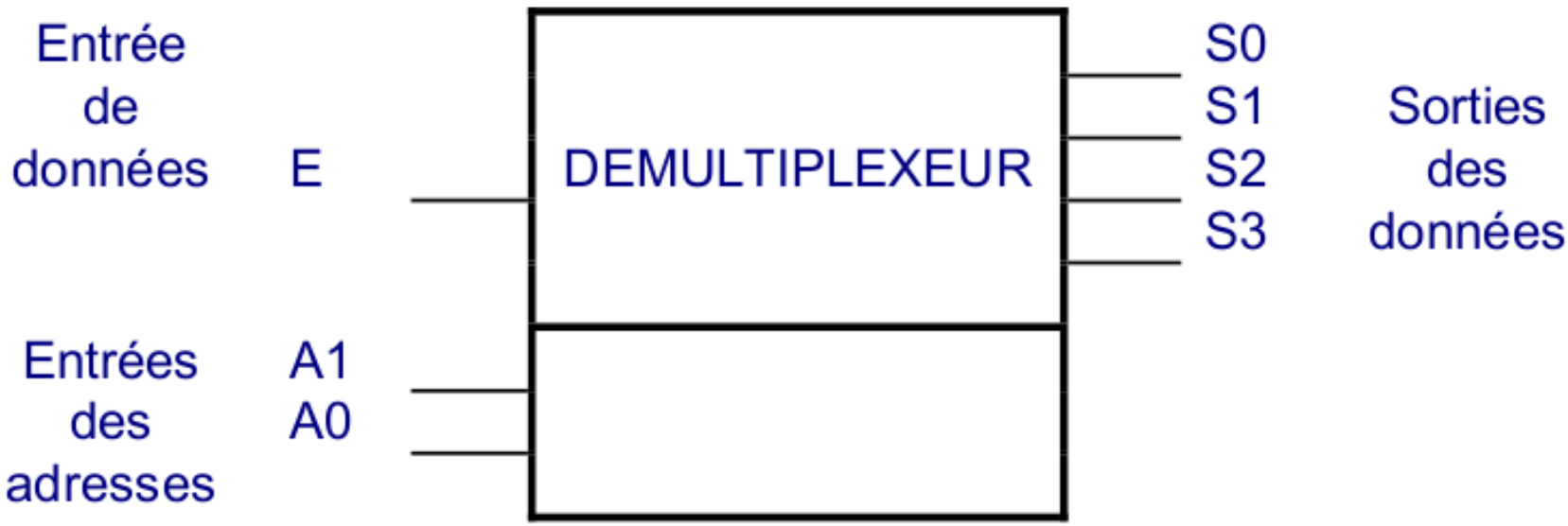
# Démultiplexeur

## Réalisation d'un démultiplexeur

Un démultiplexeur ou répartiteur de données est un commutateur qui va pouvoir, à l'aide de n bits d'adresse, aiguiller la donnée présente sur son entrée vers l'une de ses $2^n$ sorties.

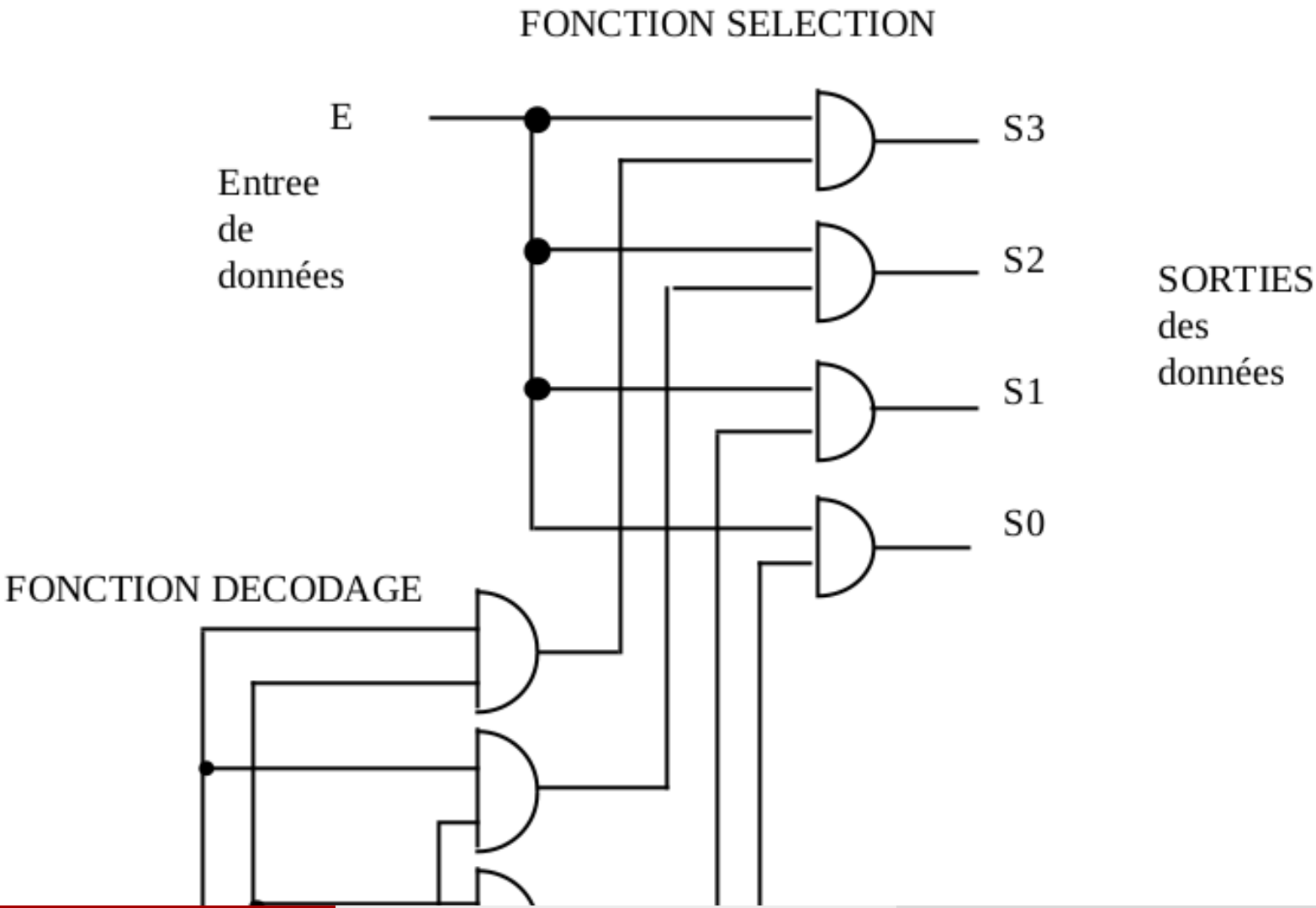
# Démultiplexeur

Le schéma ci-dessous donne une image d'un démultiplexeur une entrée (E) vers 4 sorties ($S_3$ à $S_0$) sélectables à l'aide des bits d'adresse $A_1$ et $A_0$.



# Démultiplexeur

Nous procéderons comme précédemment en séparant les fonctions :

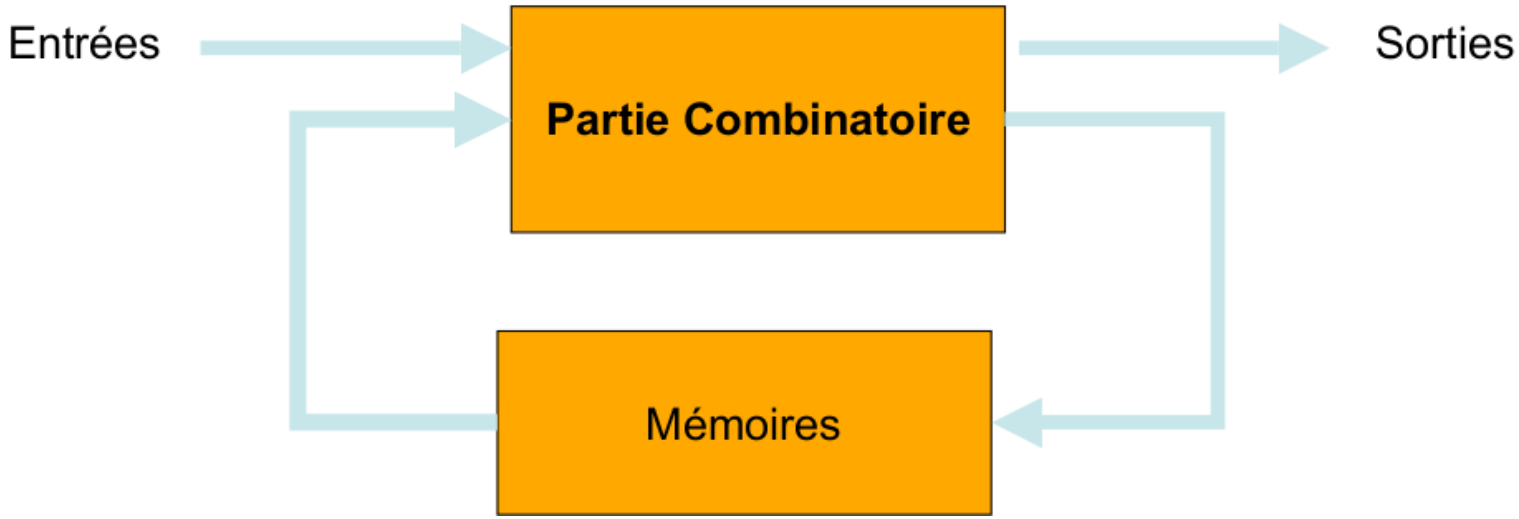


# Circuits séquentiels

# Circuits séquentiels
## Généralités sur les circuits séquentiels

# Généralités sur les circuits séquentiels

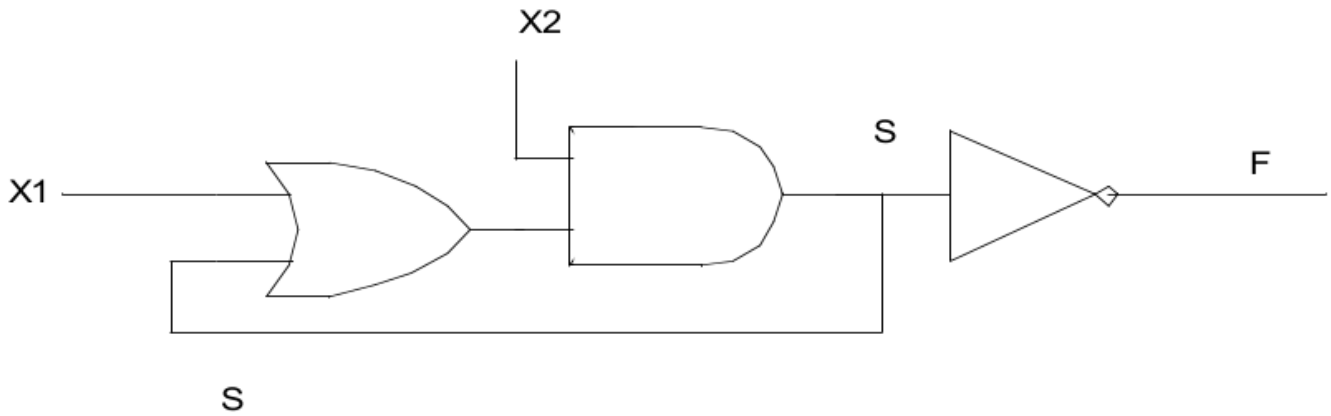
- Circuits séquentiels ou à mémoire :
  - Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de l'état antérieur de certaines variables de sortie (propriétés de mémorisation)



# Principe de fonctionnement

- Particularité des circuits séquentiels :
  - La sortie S du circuit est réinjectée à l'entrée du circuit.
  - Rétroaction.
  - L'état de sortie du circuit influencé par l'état antérieur.

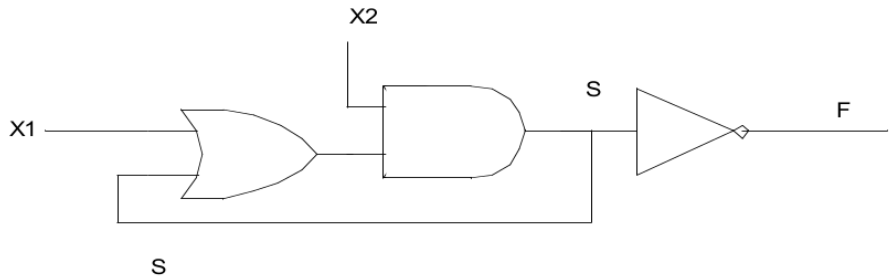
**Exemple :**



# Principe de fonctionnement

**Suite exemple :**

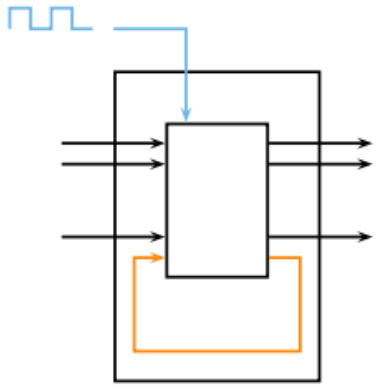
| X1 | X2 | S | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | S | $\overline{S}$ |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |




L'état pour lequel $X_1 = 0$ et $X_2 = 1$ correspond à l'état de mémorisation du circuit séquentiel.

# Circuits asynchrones vs. synchrones

- Les circuits séquentiels se divisent en deux catégories :
  - asynchrones : les variables du système évoluent librement au cours du temps.
  - synchrones : l'évolution des variables dépend d'une impulsion d'horloge comme un des signaux d'entrée.



$\Delta t$ : temps de propagation rétroactif

# Élément de mémoire

- L'élément de mémoire est défini comme un circuit séquentiel à deux états, 0 et 1, utilisé pour stocker le contenu d'un bit.
- L'état du circuit est modifié par des signaux de commutation aux entrées.
- Il possède deux sorties dont l'une est le complément de l'autre.

# Circuits séquentiels
## Les bascules

# Les Bascules

- Une bascule ou un basculeur est un circuit intégré logique doté d'une sortie et d'une ou plusieurs entrées.
- La sortie peut être au niveau logique 0 ou 1.
- Les changements d'état de la sortie sont déterminés par les signaux appliqués aux entrées.

Ce qui différencie les bascules des circuits logiques combinatoires (portes ET, OU, OU Exclusif, etc.), c'est que la sortie maintient son état même après disparition du signal de commande.

# Les bascules

- La bascule est l'élément de base de la logique séquentielle. En effet, en assemblant des bascules, on peut réaliser des compteurs, des registres, des registres à décalage, des mémoires.
- Il existe plusieurs types de bascules : RS, RSH, JK, D, verrous transparents (latchs), T. Citons également la bascule de Schmidt, qui est commandée par une tension analogique appliquée à son entrée.

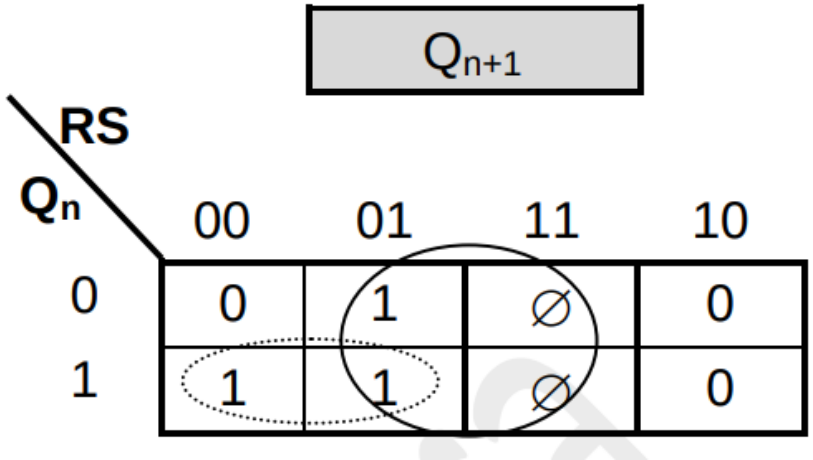
# Bascule RS

- Entrées/Sorties :
  - Entrées : R et S.
  - Sorties : Une seule sortie Q correspond à l'état stocké
- Principe : la valeur de Q à t+1 dépend de R, S et de la valeur de Q à t.
  - R : Reset, remise à 0 de Q.
  - S = Set : remise à 1 de Q.
  - S=1 et R=0 : Q mis à 1
  - S=0 et R=1 : Q mis à 0
  - S=0 et R=0 : Q garde sa valeur, maintien
  - S=1 et R=1 : Q indéterminé

# La bascule RS

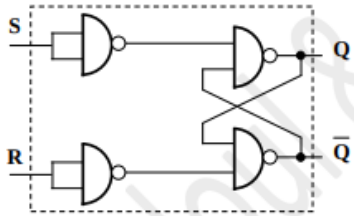
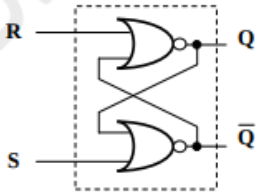
| Entrées | | | Sorties | | Mode de Fonctionnement |
|---|---|---|---|---|---|
| R | S | $Q_n$ | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | |
| 0 | 0 | 0 | 0 | 1 | État précédent |
| 0 | 0 | 1 | 1 | 0 | État précédent |
| 0 | 1 | 0 | 1 | 0 | Enclenchement (mis à 1) |
| 0 | 1 | 1 | 1 | 0 | Maintien à 1 |
| 1 | 0 | 0 | 0 | 1 | Maintien a 0 |
| 1 | 0 | 1 | 0 | 1 | Déclenchement (mis à 0) |
| 1 | 1 | 0 | X | $\overline{X}$ | Indéterminé |
| 1 | 1 | 1 | X | $\overline{X}$ | Indéterminé |

# Bascule RS



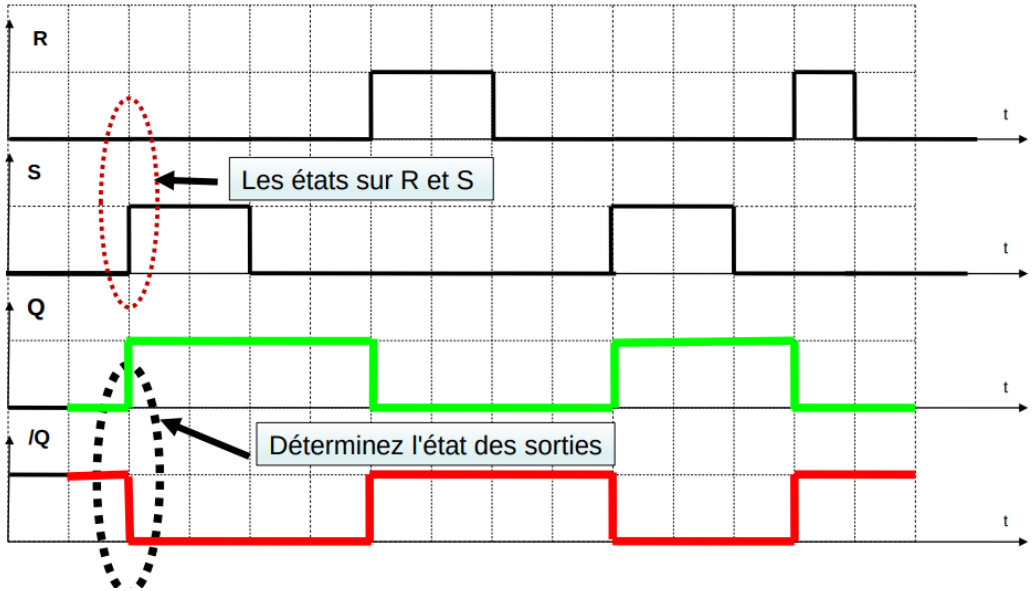

$$Q_{n+1} = Q_n\overline{R} + S$$

# Bascule RS

| Logigramme | |
|---|---|
| A l'aide des portes NAND | A l'aide des portes NOR |
|  |  |



Chronogramme :



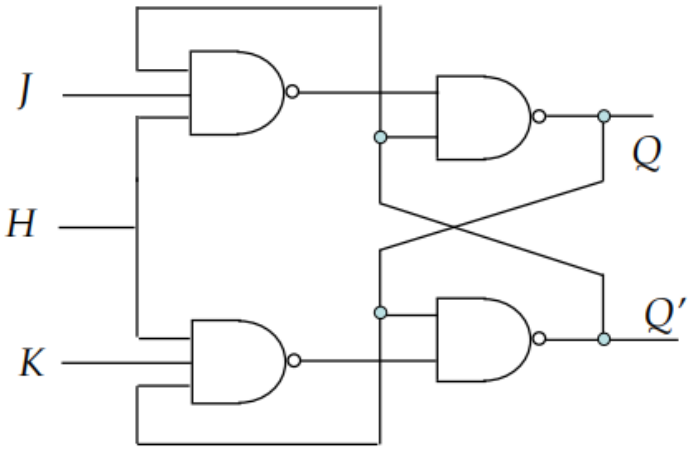
# Bascule JK

Bascule JK sur front d'horloge

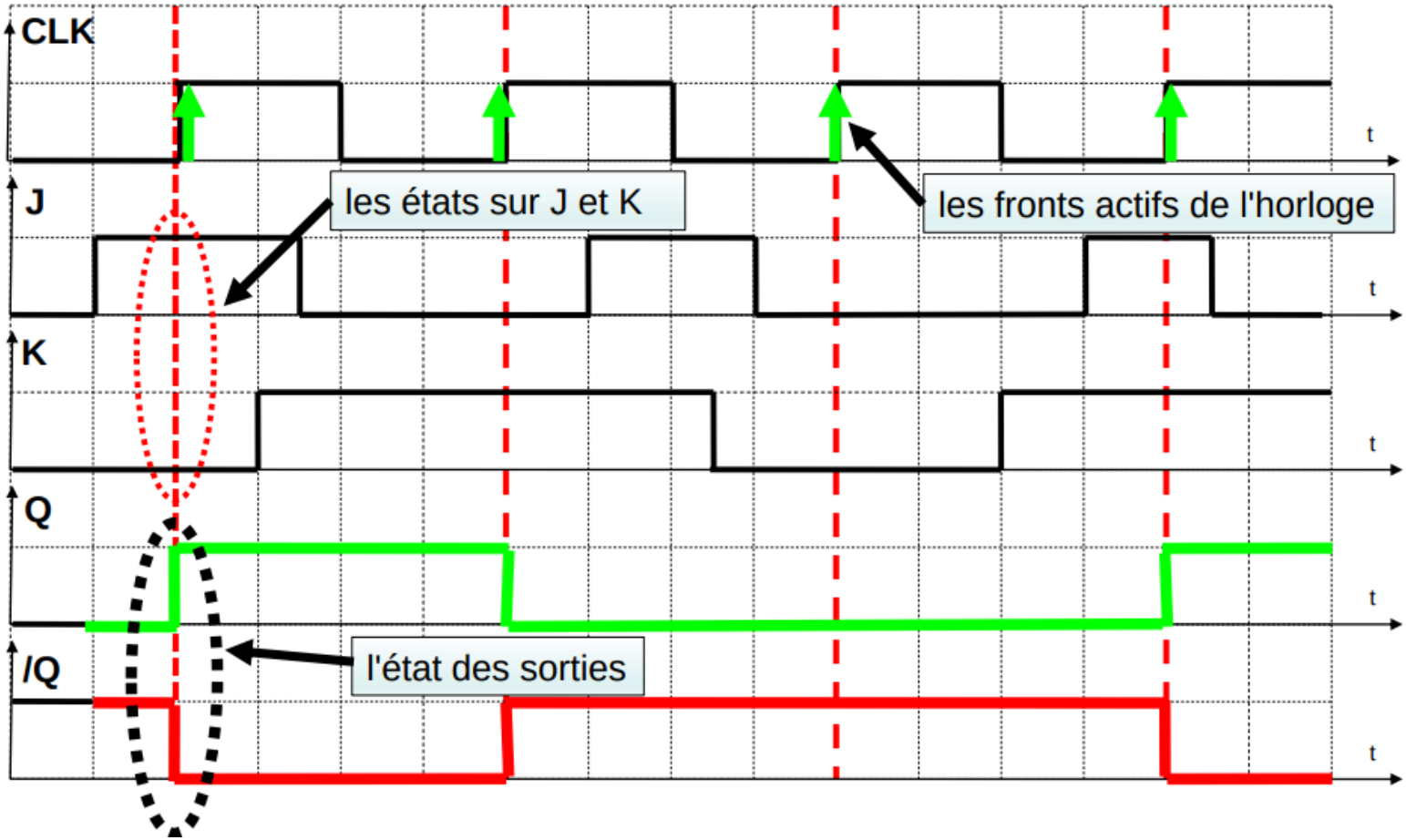
- Possède 2 entrées de données J et K.
- Fonctionnalité identique à la bascule RS à la différence près que l'état J=1, K=1 est autorisé.
- J= 1, K = 1 => inversion de l'état de la bascule.

# Bascule JK

| J | K | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q_n}$ |



# Bascule JK

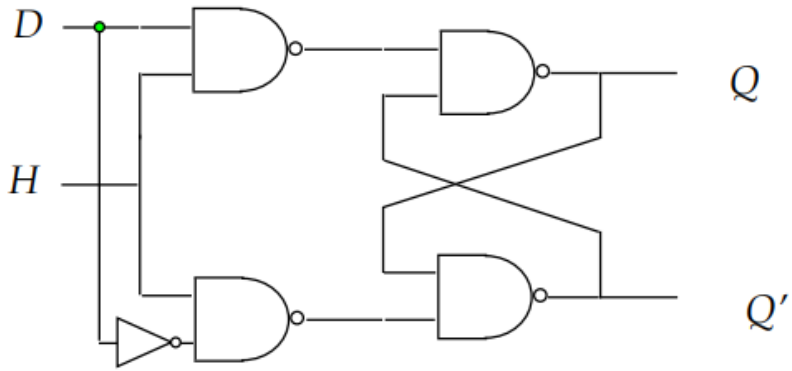

CLK
J
K
Q
/Q
t
les états sur J et K
les fronts actifs de l'horloge
l'état des sorties

# Bascule D

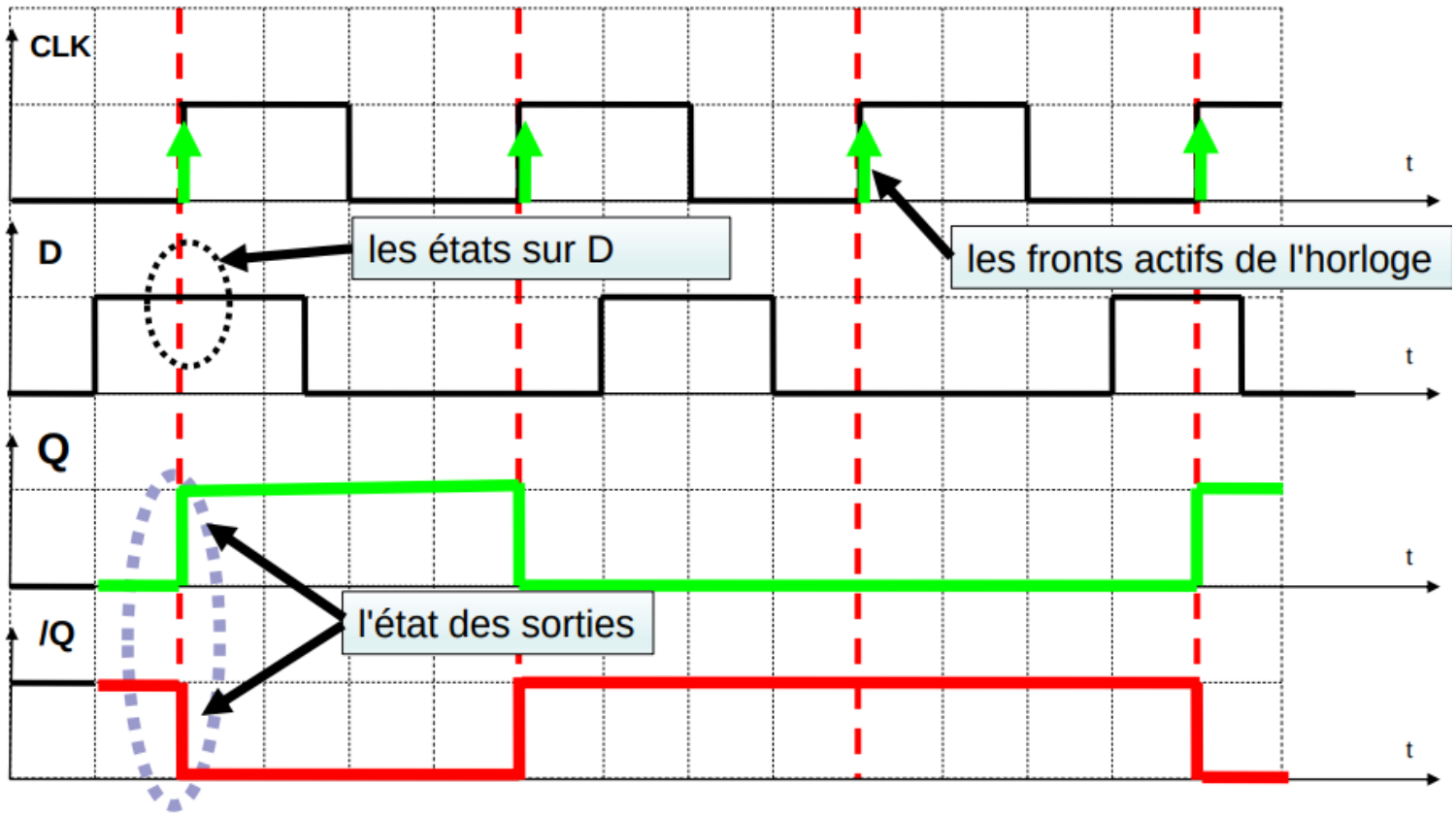
- D = delay.
- Deux entrées :
  - D : La valeur en entrée.
  - H : Entrée de contrôle, généralement un signal d'horloge
- 1 Sortie Q.
  - $Q_{n+1} = D$ Si H = 1.
  - $Q_{n+1} = Q_n$ Si H = 0 ;
- Si H est un signal d'horloge
  - Retarde le signal en entrée D d'une période d'horloge.

# Bascule D

| D | $Q_{n+1}$ |
|---|---|
| 0 | 0 |
| 1 | 1 |



# Bascule D


CLK
D
Q
/Q
t
t
t
t
les états sur D
les fronts actifs de l'horloge
l'état des sorties

# Circuits séquentiels
## Les compteurs

# Définition

## Définition

*Un compteur est un circuit logique séquentiel constitué d'un ensemble de n bascules interconnectées par des portes logiques. Il permet de dénombrer ou compter, suivant un système de numération binaire, le nombre d'impulsions appliquées à son entrée horloge : il reçoit les impulsions à compter et délivre en permanence en sa sortie une combinaison, des états logiques, image du nombre d'impulsions reçues.*

# Classification des compteurs

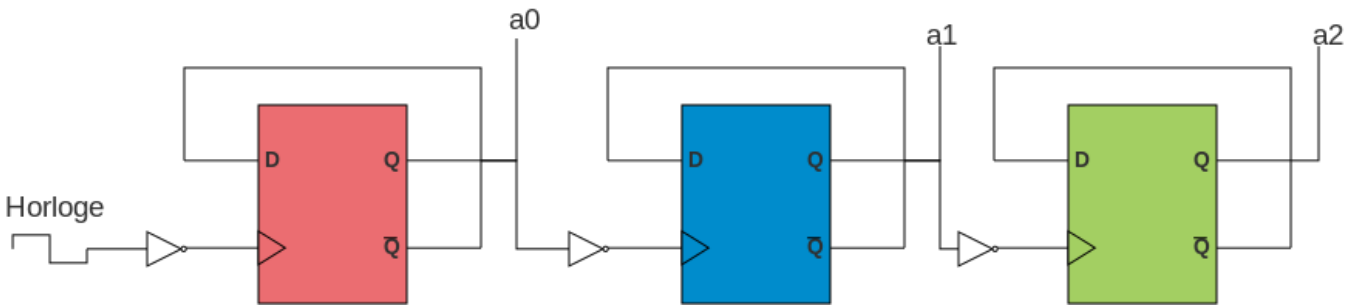
En logique séquentielle, les compteurs peuvent être décris en citant les caractéristiques suivantes :

1. Sens de comptage : Comptage croissant ou décroissant (**compteur**, **décompteur**).
2. Codage de comptage : Code dans lequel est exprimée la valeur de sortie (binaire naturel, code Gray,etc.).
3. Mode de comptage : Type de basculement asynchrone ou synchrone du compteur.
4. Modulo de comptage : Un compteur modulo M compte de 0 à (M-1) et comporte n bascules tel que : $2^{n-1} < M \leq 2^n$.
5. Cycle de comptage : précise si la valeur de sortie utilise ou non toutes les combinaisons possibles.
   - Comptage à cycle complet $M = 2^n$.
   - Comptage à cycle incomplet $M < 2^n$ .

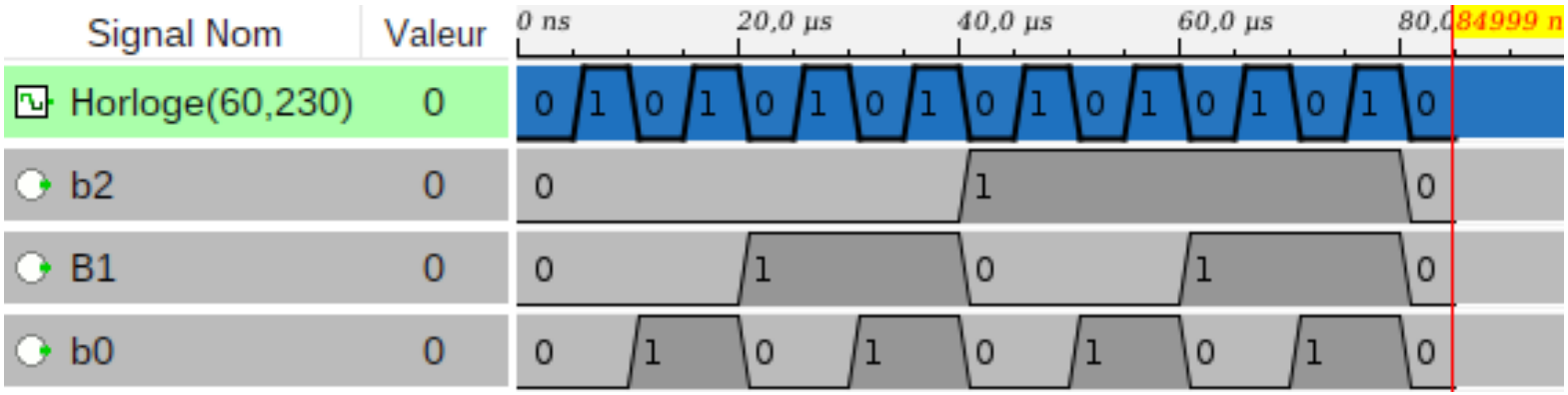
# Exemple : Compteur 3 bits

- Utilisation de 3 bascules D.
  - Principe :
    - Chaque bascule prend en entrée D un signal d'horloge.
    - Fournit en sortie un signal d'horloge de fréquence divisée par 2.
- En mettant en série les 3 bascules.
  - 3 signaux d'horloge à 3 fréquences différentes.
  - Représente les combinaisons de bits pour les valeurs de 0 à 7.

# Exemple : Compteur 3 bits



# Exemple : Compteur 3 bits



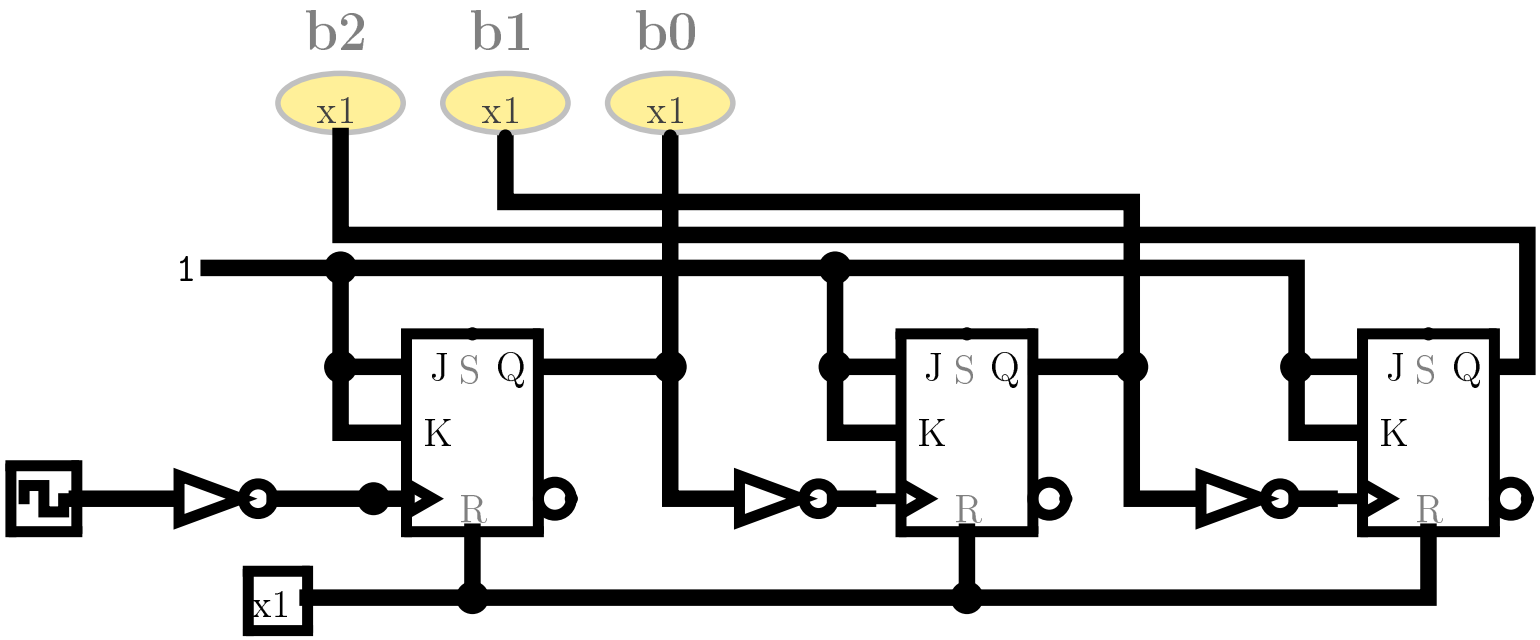
# Compteurs asynchrones

## Définition

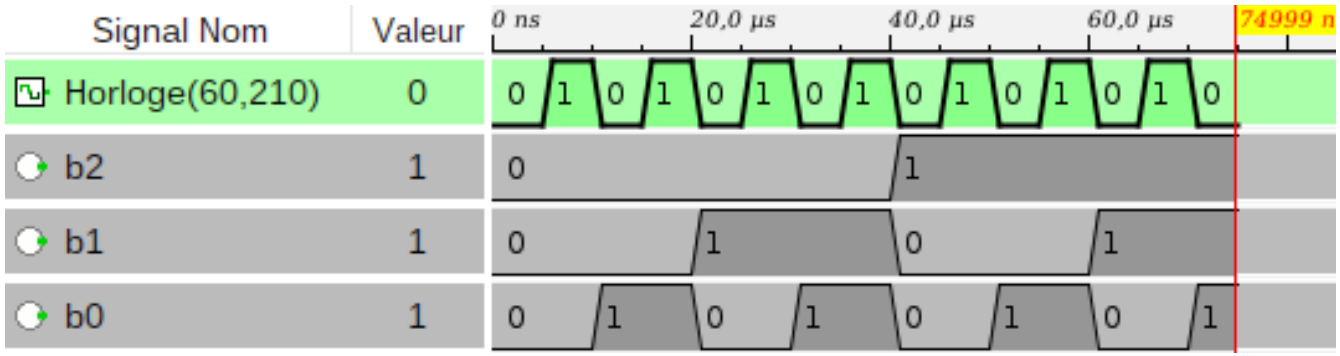
*Un compteur asynchrone est constitué de n bascules JK fonctionnant en mode T (Toggle) :* ***J=K=1.*** *Ces bascules sont montées en cascade c'est-à-dire le signal d'horloge commande uniquement la première bascule tandis que pour chacune des autres bascules le signal d'horloge est fourni par la sortie de la bascule de rang immédiatement inférieur.*

# Compteur asynchrone à cycle complet

Pour bien comprendre le principe, réalisons un compteur modulo 8 permettant de compter de 0 à 7. Pour cela, on a besoin de 3 bascules JK montées de la manière suivante :



# Compteur asynchrone à cycle complet

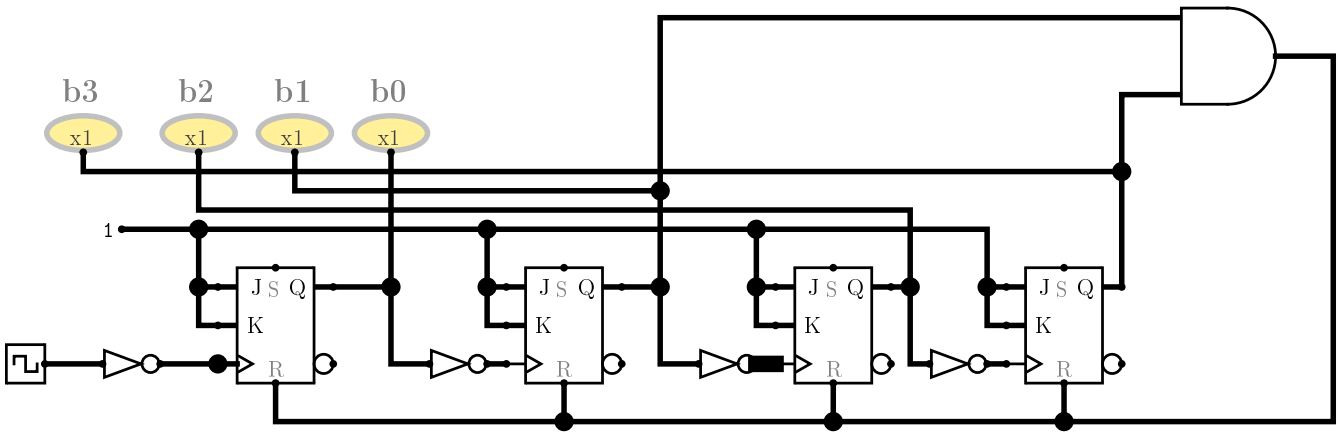


# Compteur asynchrone à cycle incomplet

Soit à concevoir un compteur asynchrone modulo 10 qui compte de 0 à 9.

- Le nombre n des bascules nécessaires : $10 < 2^n$ soit $n = 4$
- Avec 4 bascules, le modulo du compteur est 16.
- pour avoir un modulo 10 on procède à un forçage à 0 du compteur à la combinaison N=10 soit $Q_3 Q_2 Q_1 Q_0 = 1010$.
- Le forçage à 0 du compteur consiste à mettre à 0 toutes les sorties $Q_i$ des 4 bascules..

# Compteur asynchrone à cycle incomplet



# Compteurs synchrones

## Définition

*Un compteur synchrone est constitué de n bascules qui reçoivent en parallèle le même signal d'horloge. L'entrée horloge est donc commune à toutes les bascules dont les sorties changent d'états simultanément.*

# Compteur synchrone à cycle complet

Pour faire décrire au compteur une séquence déterminée il faut à chaque impulsion d'horloge définir les valeurs des entrées synchrones J et K. Pour cela on utilise la table de transition de la bascule JK ainsi que la table des états décrivant les différentes séquences du compteur.

| Table de transition | | | |
|---|---|---|---|
| $Q_n$ | $Q_{n+1}$ | J | K |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

# Compteur synchrone à cycle complet

| État actuel (n) | | | État Suivant (n+1) | | | Bascule 2 | | Bascule 1 | | Bascule 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 |

# Compteur synchrone à cycle complet

| $J_0$ : $Q_0 / Q_1 Q_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | X | X | 1 |
| 1 | 1 | X | X | 1 |

| $k_0$ : $Q_0 / Q_1 Q_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | 1 | 1 | X |
| 1 | X | 1 | 1 | X |

$$J_0 = K_0 = 1$$

| $J_1$ : $Q_2 / Q_1 Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | • | • | • | • |
| 1 | • | • | • | • |

| $k_1$ : $Q_2 / Q_1 Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | • | • | • | • |
| 1 | • | • | • | • |

| $J_2$ : $Q_2 / Q_1 Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | • | • | • | • |
| 1 | • | • | • | • |

| $k_2$ : $Q_2 / Q_1 Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | • | • | • | • |
| 1 | • | • | • | • |

# Compteur synchrone à cycle complet

| $J_0$ : $Q_2/Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | X | X | 1 |
| 1 | 1 | X | X | 1 |

| $k_0$ : $Q_2/Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | 1 | 1 | X |
| 1 | X | 1 | 1 | X |

$J_0 = K_0 = 1$

| $J_1$ : $Q_2/Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | X | X |
| 1 | 0 | 1 | X | X |

| $k_1$ : $Q_2/Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | 1 | 0 |
| 1 | X | X | 1 | 0 |

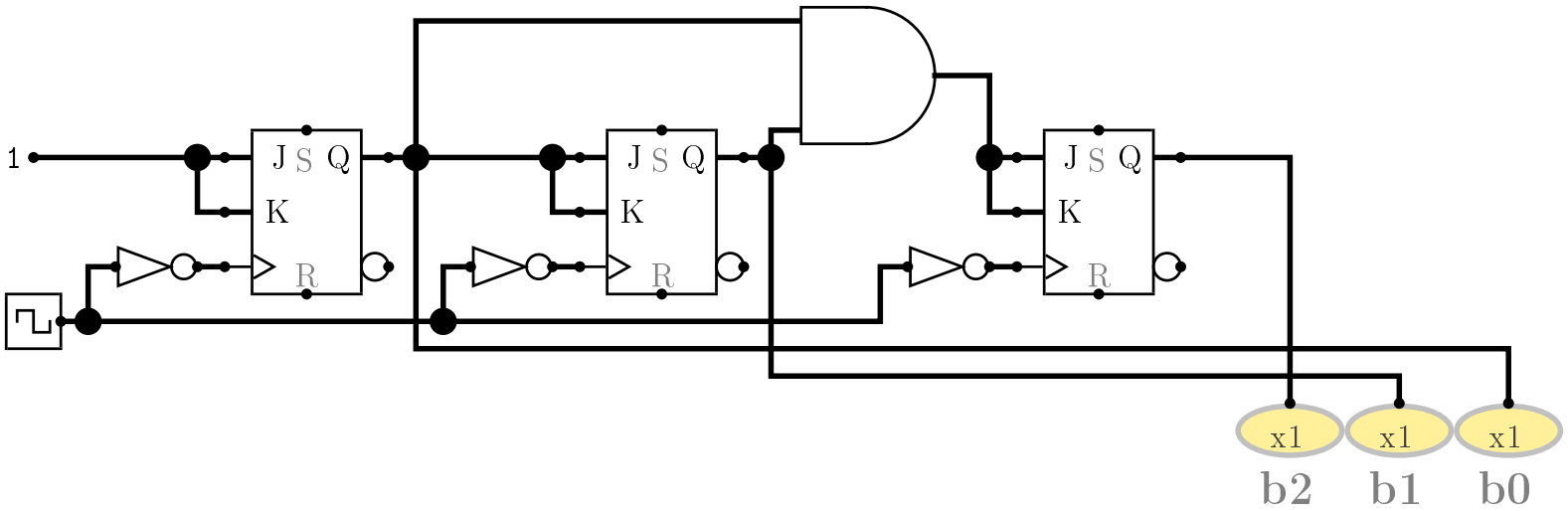
$J_1 = K_1 = Q_0$

| $J_2$ : $Q_2/Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | X | X | X | X |

| $k_2$ : $Q_2/Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | X | X |
| 1 | 0 | 0 | 1 | 0 |

$J_2 = K_2 = Q_1Q_0$

# Compteur synchrone à cycle complet



# Compteur synchrone à cycle incomplet

Soit à concevoir un compteur synchrone modulo 10.
Le nombre n des bascules nécessaires pour la réalisation du compteur est tel que : $10 < 2^n$ soit $n = 4$.

| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

# Compteur synchrone à cycle incomplet

| $j_0$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | x | x | 1 |
| 01 | 1 | x | x | 1 |
| 11 | x | x | x | x |
| 10 | 1 | x | x | x |

| $j_1$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | x | x |
| 01 | 0 | 1 | x | x |
| 11 | x | x | x | x |
| 10 | 0 | 0 | x | x |

| $j_2$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | x | x | x | x |
| 11 | x | x | x | x |
| 10 | 0 | 0 | x | x |

| $j_3$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | x | x | x | x |
| 10 | x | x | x | x |

| $k_0$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | 1 | 1 | x |
| 01 | x | 1 | 1 | x |
| 11 | x | x | x | x |
| 10 | x | 1 | x | x |

| $k_1$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | x | 1 | 0 |
| 01 | x | x | 1 | 0 |
| 11 | x | x | x | x |
| 10 | x | x | x | x |

| $k_2$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | x | x | x |
| 01 | 0 | 0 | 1 | 0 |
| 11 | x | x | x | x |
| 10 | x | x | x | x |

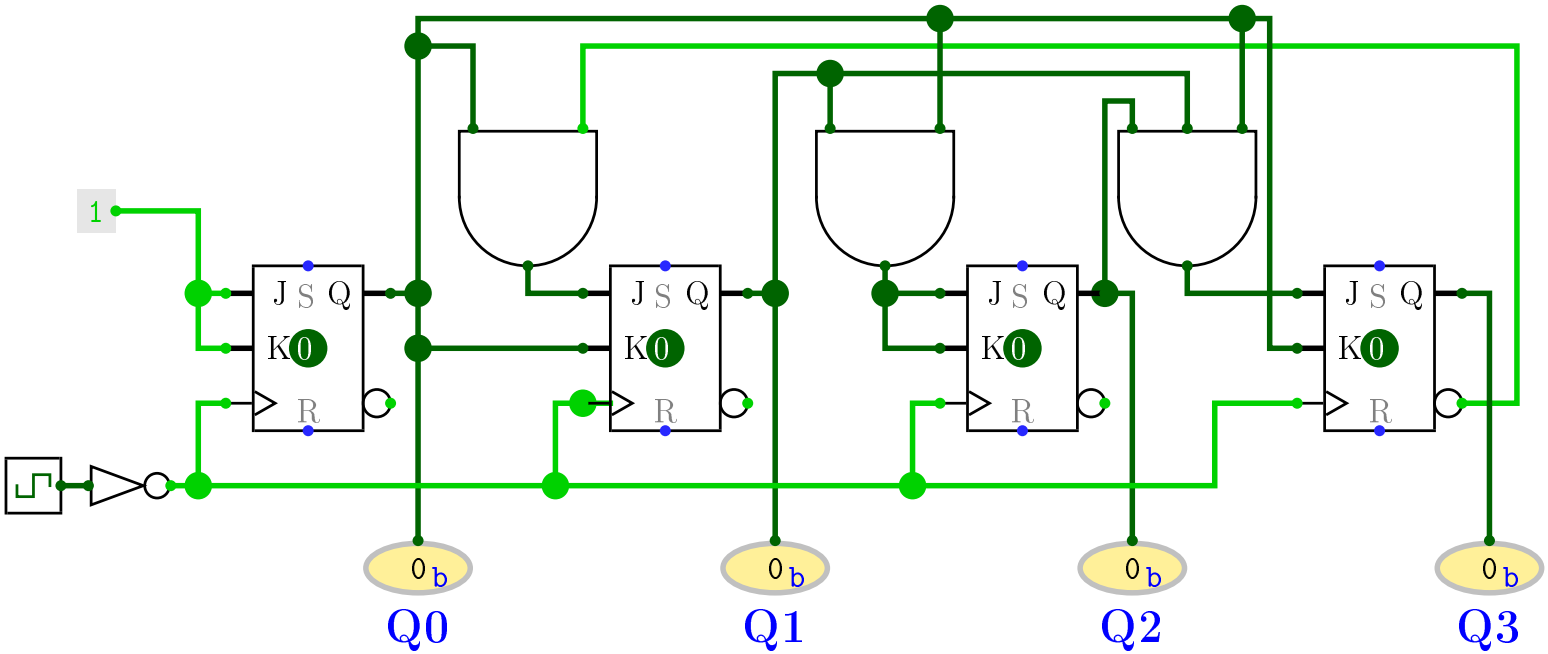
| $k_3$ : $q_3\,q_2 / q_1\,q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | x | x | x |
| 01 | x | x | x | x |
| 11 | x | x | x | x |
| 10 | 0 | 1 | x | x |

# Compteur synchrone à cycle incomplet

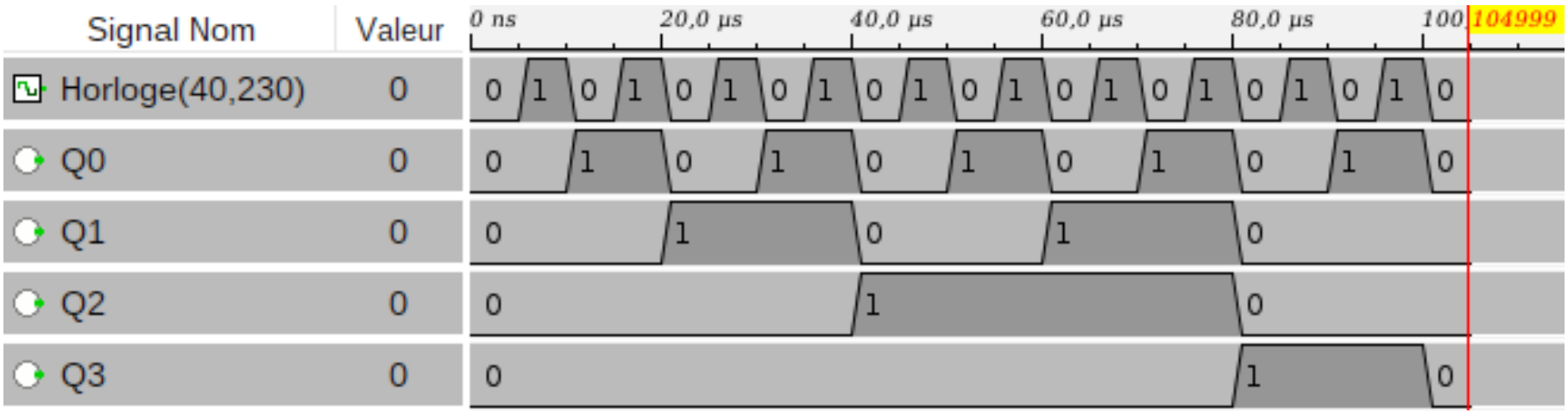
## Équations :

- $J_0 = K_0 = 1.$
- $J_1 = \overline{Q_3}\, Q_0$
- $K_1 = Q_0$
- $J_2 = K_2 = Q_1\, Q_0$
- $J_3 = Q_2\, Q_1\, Q_0$
- $K_3 = Q_0$

# Compteur synchrone à cycle incomplet



# Compteur synchrone à cycle incomplet



# Circuits séquentiels
## Les registres

# Définitions

## Définition

*Un registre est un dispositif qui permet de mémoriser une information et de la restituer autant de fois que désiré.*

## Définition

*Tout registre comporte un mécanisme de remise à zéro (**RAZ**), qui met tous les registres élémentaires qui le compose à zéro simultanément.*

## Remarque

Notons qu'on ne retrouve pas seulement des registres dans la mémoire centrale, mais aussi dans les autres composantes de l'ordinateur comme l'unité centrale de traitement (CPU), dans les unités d'entrées/sorties, etc.

# Définitions

Il existe dans grandes classes des registres :

- Les registres à décalage
  - Stockage et décalage.
- Les registres de mémorisation
  - Stockage.

# Les registres à décalage

## Définition

*Un registre à décalage est une association en cascade de bascules D qui sont interconnectées de façon à ce que l'état logique de la bascule de rang i puisse être transmis à la bascule de rang i+1 (décalage à droite) ou i-1 (décalage à gauche) quand un signal d'horloge est appliqué à l'ensemble des bascules.*

# Classification des registres à décalage

En logique séquentielle, les registres à décalage se distinguent selon les caractéristiques suivantes :

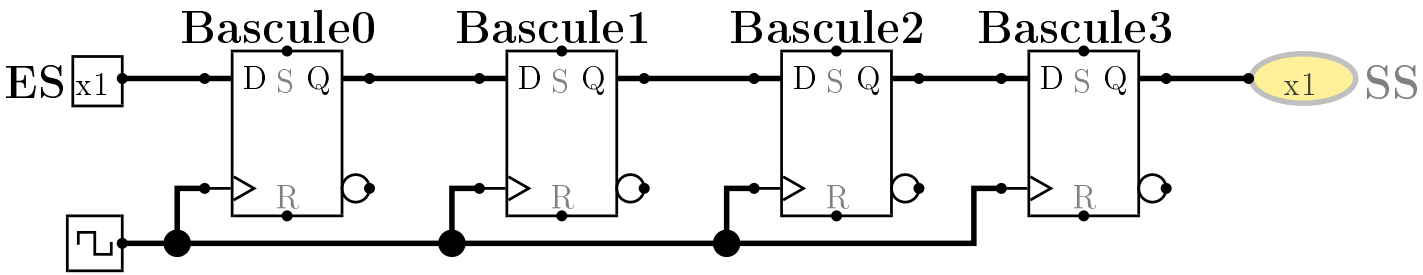
1. Type de décalage : L'information (une série de bits de donnée) peut être sérialisée suivant une direction de décalage à droite, à gauche ou dans ces deux directions (registre bidirectionnel).
2. Mode d'écriture : L'information peut être chargée selon un mode d'écriture série ou parallèle :
   - Mode parallèle :La donnée est chargée dans les bascules d'un seul coup d'horloge.
   - Mode série : La donnée est présentée séquentiellement bit après bit à l'entrée de la première bascule, le décalage peut être alors vers la gauche ou la droite.
3. Mode de lecture : De même, l'information peut être lue en série ou en parallèle.
4. Nombre de bits : Le nombre de bits dans les registres est de 4, 5 ou 8. Un registre composé de ces registres élémentaires peut contenir tout nombre de bits désiré.

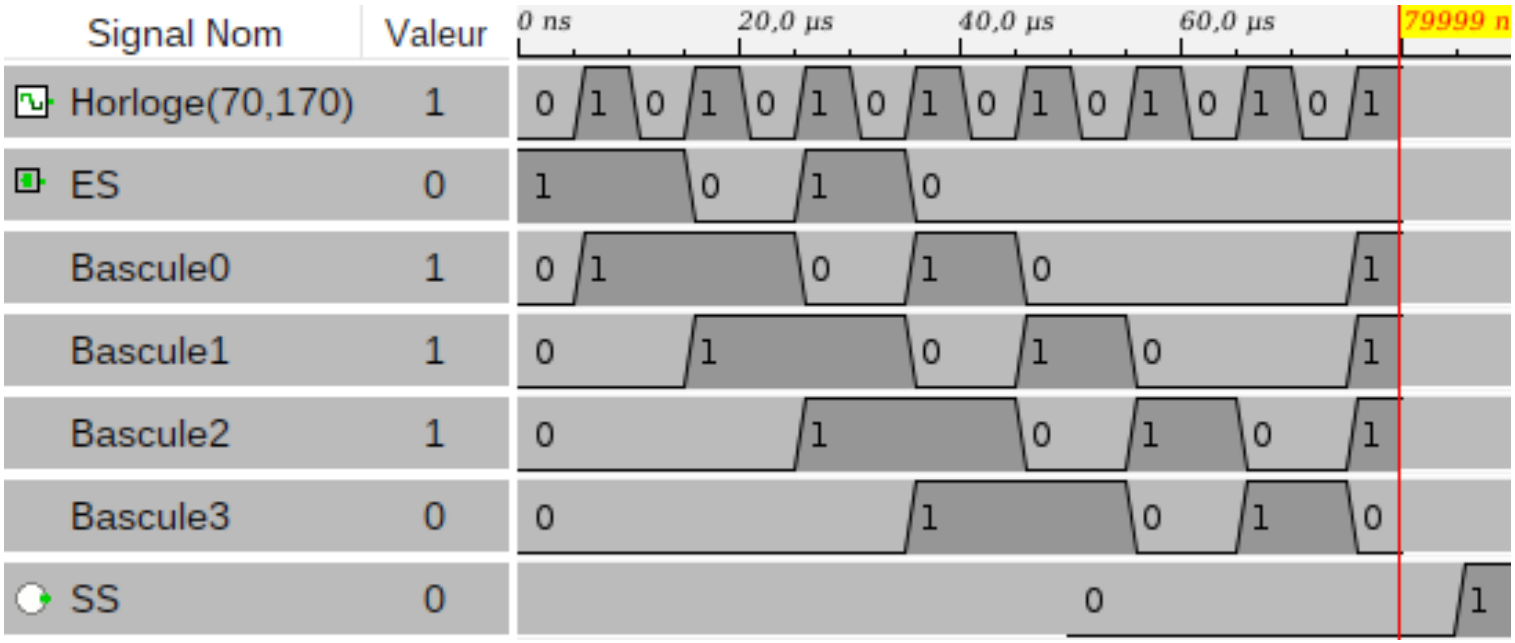
# Types des registres à décalage

- Registres à entrée parallèle et à sortie parallèle.
- Registres à entrée parallèle et à sortie série.
- Registres à entrée série et à sortie série.
- Registres à entrée série et à sortie parallèle.
- Les registres universels : rassemblent les 4 modes de fonctionnement décrits ci-dessus.

# Exemples :

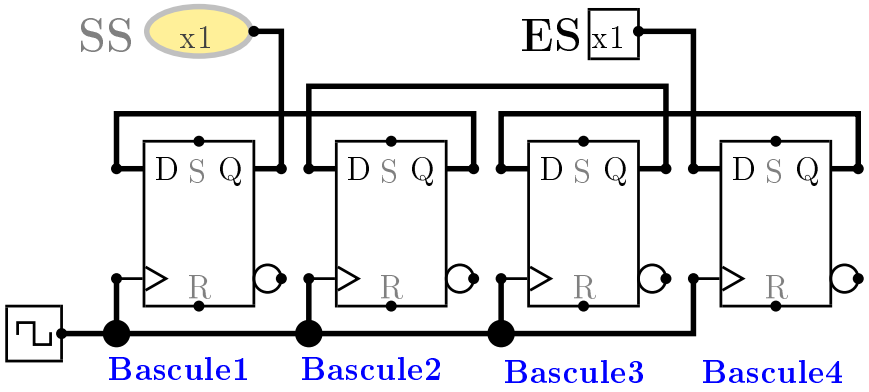
## Registre à décalage à droite Entrée Série(ES)/Sorite Série (SS) :

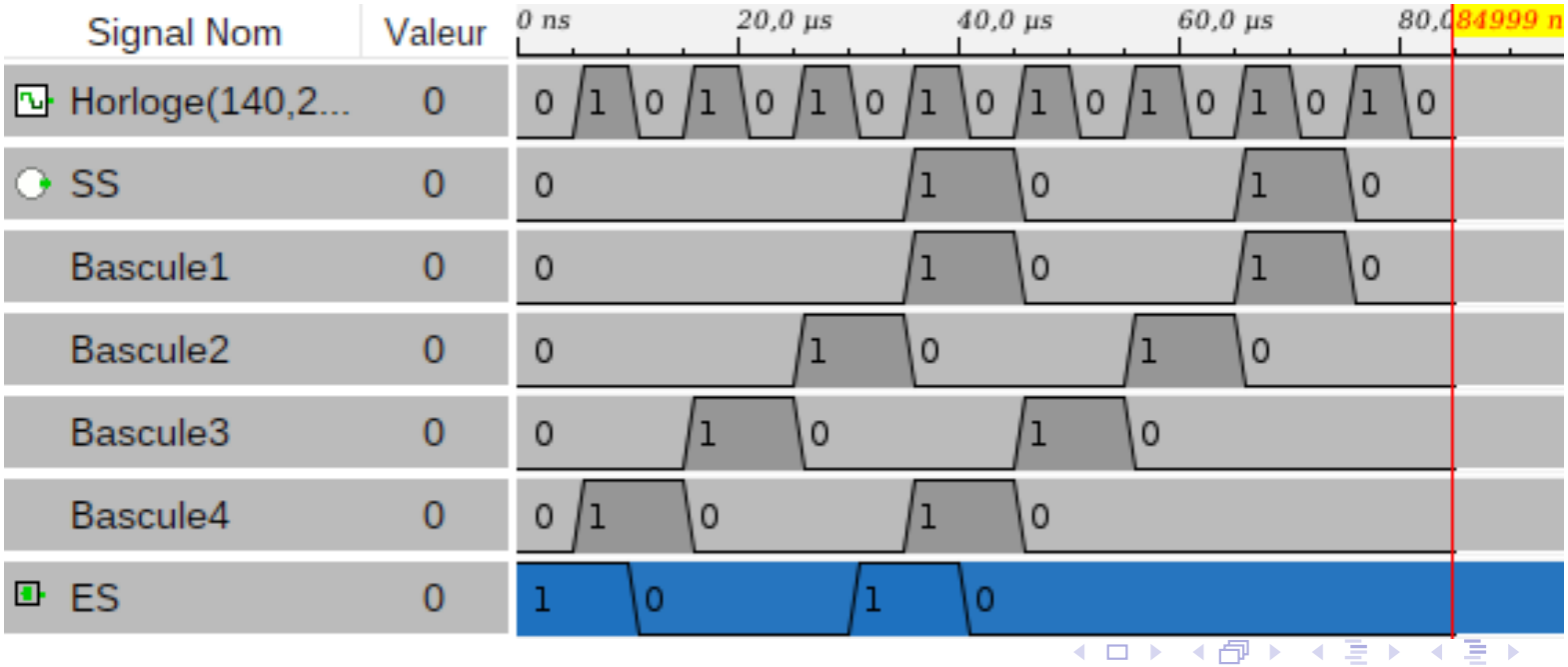


# Exemples :

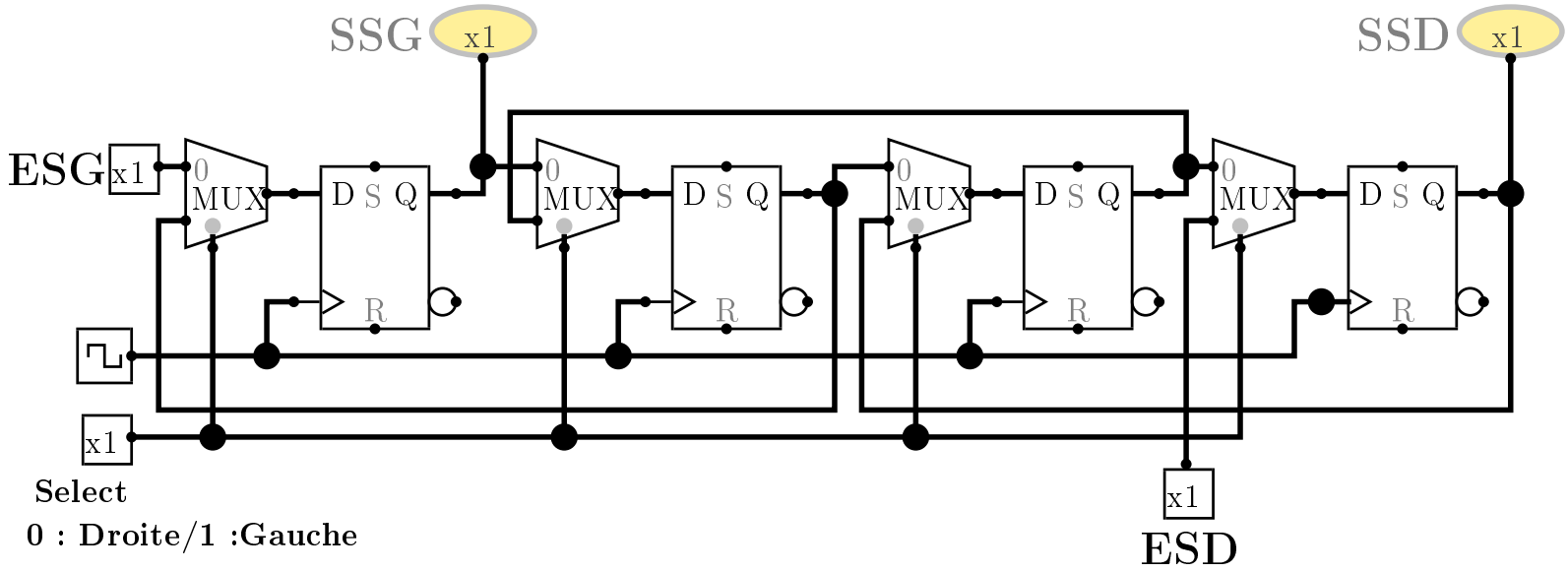
Registre à décalage à gauche Entrée Série(ES)/Sorite Série (SS) :



# Exemples :

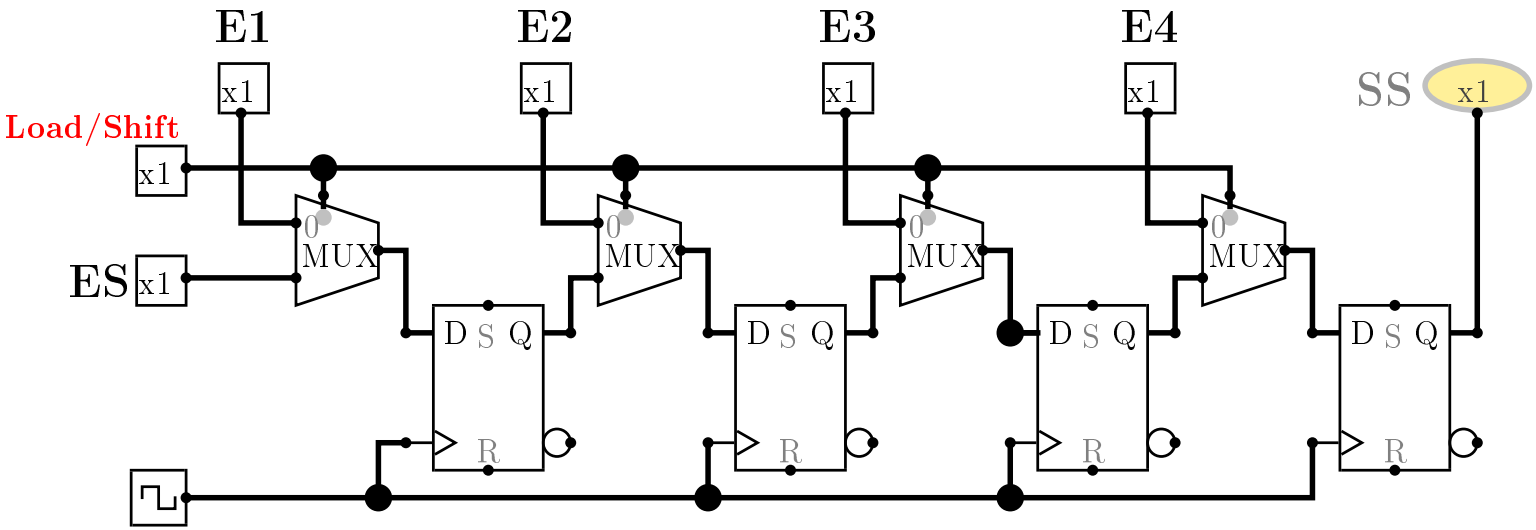
Registre à décalage à gauche/droite Entrée Série(ES)/Sorite Série (SS) :




- ESG : Entrée Série Gauche.
- ESD : Entrée Série Droite.
- Mux : Multiplexeur deux entrées / une sortie avec 1 bit de sélection.
- Select : le bit de sélection (0 :Droite/1 :Gauche).

# Exemples :

Registre à décalage à gauche (Entrée Série(ES)/Entrée parallèles Ei)/Sortie Série (SS) :




- Load/Shift
  - 0 : Load, chargement des entrées parallèles.
  - 1 : Shift, décalage des bits vers la droite.
- E1...E4 : Les entrées parallèles.
- ES/SS : Entrée série/ Sortie Série.