Chapitre 1
Les circuits logiques

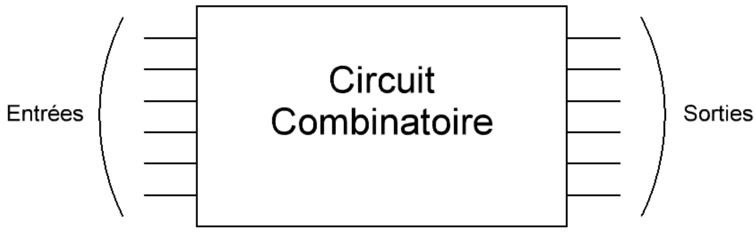
Version du 09/10/2020

Table des matières

I. Les circuits combinatoires

1. Définition

Un circuit combinatoire est un circuit logique dont l'état des sorties se déduit uniquement de l'état des entrées à un instant donné.



L'état d'une entrée ou d'une sortie est soit l'état logique 0, soit l'état logique 1.

Un circuit combinatoire peut se définir par une table de vérité. Une table de vérité est un tableau qui exprime la valeur des sorties en fonction de la valeur des entrées.

2. Les portes logiques

Les portes logiques sont les éléments de base des circuits combinatoires.

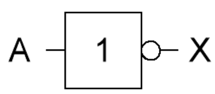
Il existe deux symboles pour chaque porte. Les symboles à forme rectangulaire et les symboles à forme distincte. Dans ce cours, **nous utiliserons les symboles à forme distincte**. Néanmoins, il est nécessaire de bien connaître les symboles à forme rectangulaire, car ils sont présents dans un grand nombre d'ouvrages et de documentations techniques.

2.1. La porte NON (*NOT*)

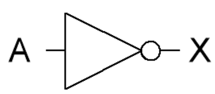
$$\mathbf{X = \overline{A}}$$

A	X
0	1
1	0

Table de vérité



Symbole à forme rectangulaire



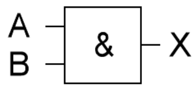
Symbole à forme distincte

2.2. La porte ET (*AND*)

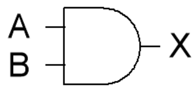
$$X = A \cdot B$$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité



Symbole à forme rectangulaire



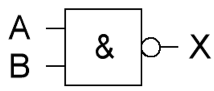
Symbole à forme distincte

2.3. La porte NON-ET (*NAND*)

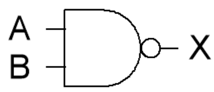
$$X = \overline{A \cdot B}$$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Table de vérité



Symbole à forme rectangulaire



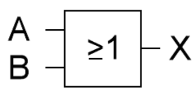
Symbole à forme distincte

2.4. La porte OU (*OR*)

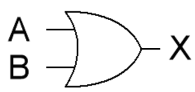
$$X = A + B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Table de vérité



Symbole à forme rectangulaire



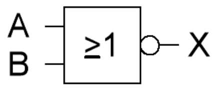
Symbole à forme distincte

2.5. La porte NON-OU (*NOR*)

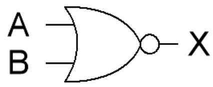
$X = \overline{A + B}$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Table de vérité



Symbole à forme rectangulaire



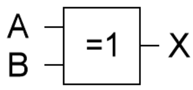
Symbole à forme distincte

2.6. La porte OU EXCLUSIF (*XOR*)

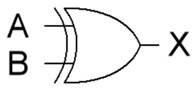
$X = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité



Symbole à forme rectangulaire



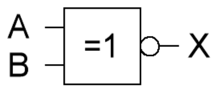
Symbole à forme distincte

2.7. La porte NON-OU EXCLUSIF (*NXOR*)

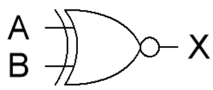
$X = \overline{A \oplus B}$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Table de vérité



Symbole à forme rectangulaire

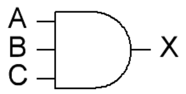


Symbole à forme distincte

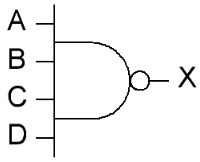
2.8. Les portes à plus de deux entrées

Les portes logiques peuvent accepter plus de deux entrées.

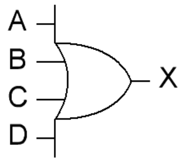
Exemples :



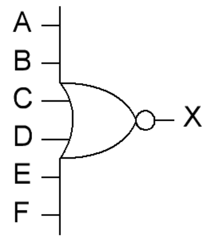
$X = A.B.C$



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



$X = A + B + C + D$

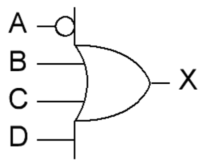


$X = \overline{A + B + C + D + E + F}$

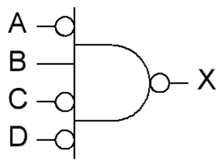
2.9. Les bulles et les portes

Les petites bulles sur les symboles de portes sont des indicateurs de négation. Ils indiquent une inversion logique sur les entrées ou les sorties. Toute combinaison est possible.

Exemples :



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



$X = \overline{\overline{A}\,.\,B.\overline{C}\,.\overline{D}}$

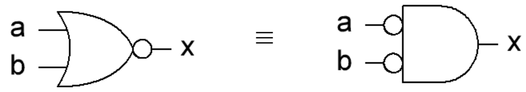
2.10. Les portes synonymes

La bulle représentant l'inversion peut également se placer sur les entrées des portes logiques. À l'aide des théorèmes de De Morgan, il est donc possible d'obtenir des symboles différents pour une même porte.

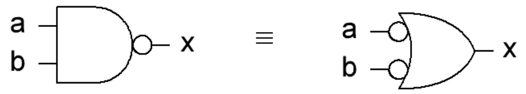
Théorèmes de De Morgan :

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

Exemples :



Porte NON-OU : $x = \overline{a + b} = \overline{a} . \overline{b}$

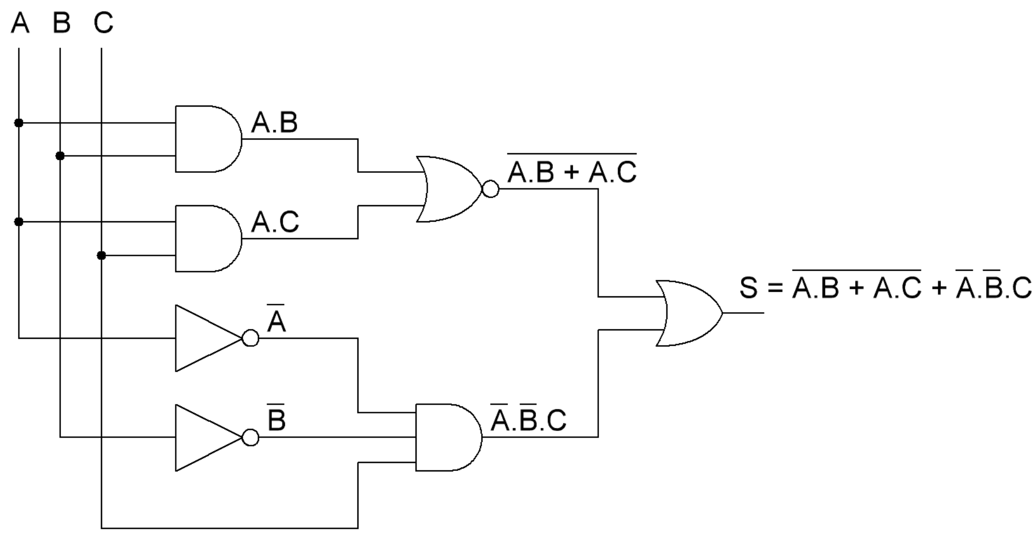


Porte NON-ET : $x = \overline{a . b} = \overline{a} + \overline{b}$

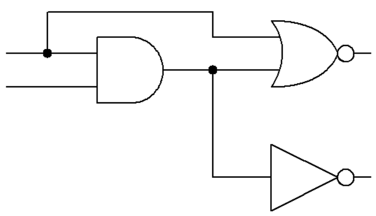
2.11. Associations de portes logiques

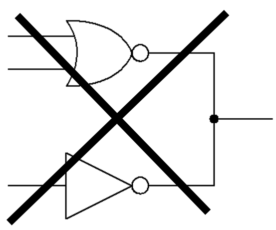
Tout type de circuit combinatoire peut être obtenu en connectant les portes logiques entre elles.

Par exemple :



Toute entrée ou sortie peut se connecter à n'importe quelle entrée, mais une sortie ne peut pas être connectée à une autre sortie.





3. Quelques règles de l'algèbre booléenne

3.1. Introduction

L'algèbre booléenne (ou algèbre de Boole) manipule des expressions constituées de variable logique. Elle nous permettra de décrire et de simplifier les circuits combinatoires.

Les trois opérateurs fondamentaux de l'algèbre booléenne sont le NON, le ET et le OU.

Le OU EXCLUSIF n'est pas un opérateur fondamental.

3.2. Principaux axiomes et théorèmes

NON	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A} . A = 0$
ET	$A . (B . C) = (A . B) . C = A . B . C$ $A . B = B . A$ $A . A = A$ $A . 1 = A$ $A . 0 = 0$
OU	$A + (B + C) = (A + B) + C = A + B + C$ $A + B = B + A$ $A + A = A$ $A + 0 = A$ $A + 1 = 1$
OU EXCLUSIF	$A \oplus B = \overline{A} . B + A . \overline{B}$ $\overline{A \oplus B} = A . B + \overline{A} . \overline{B}$ $A \oplus (B \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$ $A \oplus B = B \oplus A$ $A \oplus 0 = A$ $A \oplus 1 = \overline{A}$ $A \oplus A = 0$ $A \oplus \overline{A} = 1$

Distributivité	$A \,.\, (B + C) = A \,.\, B + A \,.\, C$ $A + B \,.\, C = (A + B) \,.\, (A + C)$
Théorèmes de De Morgan	$\overline{A \,.\, B} = \overline{A} + \overline{B}$ $\overline{A + B} = \overline{A} \,.\, \overline{B}$
Autres théorèmes	Théorème 1 : $A + A \,.\, B = A$ Théorème 2 : $A + \overline{A} \,.\, B = A + B$

Démonstration	
Théorème 1	**Théorème 2**
$A + A \,.\, B$ $= A \,.\, 1 + A \,.\, B$ $= A \,.\, (1 + B)$ $= A \,.\, 1$ $= A$	$A + \overline{A} \,.\, B$ $= (A + \overline{A}) \,.\, (A + B)$ → *distributivité* $= 1 \,.\, (A + B)$ $= A + B$

3.3. Principe de dualité

Les égalités booléennes restent vraies si l'on inverse les 0 et les 1 ainsi que les ET et les OU.

Exemples		
$1 + 0 = 1$	↔	$0.1 = 0$
$X + 1 = 1$	↔	$X.0 = 0$
$X.\overline{X} = 0$	↔	$X + \overline{X} = 1$
$X + \overline{X}.Y = X + Y$	↔	$X.(\overline{X} + Y) = X.Y$

4. Synthèse d'un circuit combinatoire

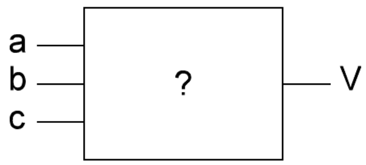
La synthèse d'un circuit combinatoire doit respecter les trois étapes suivantes :

- Établir la table de vérité du circuit à concevoir ;
- Déduire de cette table de vérité l'expression de la sortie en fonction des entrées ;
- Simplifier au maximum l'expression obtenue afin de minimiser le nombre de composants utilisés.

Le plus simple est d'étudier un exemple concret. Soit la table de vérité suivante :

Entrées			**Sortie**
a	*b*	*c*	*V*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Nous allons chercher le circuit combinatoire associé à cette table de vérité. Ce circuit comprend donc trois entrées (*a*, *b*, *c*) et une sortie (*V*).



L'objectif est de déterminer les portes qui constituent ce circuit combinatoire. Pour cela, il faut commencer par obtenir l'expression de la sortie *V* en fonction des entrées *a*, *b* et *c*.

4.1. Déterminer l'expression d'une sortie à partir d'une table de vérité

En observant la table de vérité, on s'aperçoit que ***V* = 1** si ***a* = *0*** et ***b* = *0*** et ***c* = 0** ou si ***a* = 0** et ***b* = *1*** et ***c* = 1** ou si ***a* = 1** et ***b* = *0*** et ***c* = 0** ou si ***a* = 1** et ***b* = 0** et ***c* = 1** ou si ***a* = *1*** et ***b* = *1*** et ***c* = 1**.

En remplaçant les « et » et les « ou » par leurs symboles logiques, on obtient l'expression suivante :
$V = \overline{a}.\overline{b}.\overline{c} + \overline{a}.b.c + a.\overline{b}.\overline{c} + a.\overline{b}.c + a.b.c$

On pourrait s'arrêter là, mais il est préférable de simplifier cette expression au maximum afin de minimiser le nombre de portes du circuit.

4.2. Simplifier une expression

Il existe deux méthodes de simplification : la méthode algébrique et la méthode des tableaux de Karnaugh.

4.2.1. Simplification par la méthode algébrique

En s'aidant des règles de l'algèbre booléenne, on obtient :

$V = \overline{a}.\overline{b}.\overline{c} + \overline{a}.b.c + a.\overline{b}.\overline{c} + a.\overline{b}.c + a.b.c$
$V = \overline{a}.\overline{b}.\overline{c} + b.c.(\overline{a} + a) + a.\overline{b}.(\overline{c} + c)$
$V = \overline{a}.\overline{b}.\overline{c} + b.c.1 + a.\overline{b}.1$
$V = \overline{a}.\overline{b}.\overline{c} + b.c + a.\overline{b}$
$V = b.c + \overline{b}.(a + \overline{a}.\overline{c})$
$V = b.c + \overline{b}.(a + \overline{c})$
$\mathbf{V = b.c + \overline{b}.\overline{c} + a.\overline{b}}$

Une simplification **facultative** peut être effectuée. En effet, on peut remarquer que $\mathbf{b.c + \overline{b}.\overline{c}}$ est un NON-OU EXCLUSIF entre *b* et *c*. Ce qui donne :
$\mathbf{V = \overline{b \oplus c} + a.\overline{b}}$

4.2.2. Simplification par les tableaux de Karnaugh

Un tableau de Karnaugh est en quelque sorte une façon différente de représenter une table de vérité.

Il utilise le code Gray (ou binaire réfléchi) dans lequel un seul bit est modifié entre deux valeurs consécutives. Ci-dessous, un exemple de code Gray sur trois bits :

Binaire Classique	Code Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

Dans le circuit que nous souhaitons concevoir, il y a trois variables d'entrée (*a*, *b* et *c*) ; ce qui donne huit combinaisons possibles ($2^3 = 8$). Notre tableau de Karnaugh comportera donc huit cases : quatre colonnes et deux lignes (l'inverse serait équivalent) :

bc

	V	**00**	**01**	**11**	**10**
a	**0**				
	1				

← Code Gray (ligne d'en-tête **00 01 11 10**)

↑
Code Gray (colonne **0 1**)

Il est préférable de conserver le même ordre des variables d'entrée que celui présent dans la table de vérité : *a*, *b*, *c* de gauche à droite.

S'il y avait quatre variables, on aurait dessiné un tableau carré de quatre colonnes sur quatre lignes et l'on aurait respecté le code Gray pour l'ordre des lignes comme pour l'ordre des colonnes.

On remplit le tableau en reportant dans les cases correspondantes les valeurs de la table de vérité, ce qui donne le résultat suivant :

bc

	V	**00**	**01**	**11**	**10**
a	**0**	1	0	1	0
	1	1	1	1	0

Maintenant, l'astuce consiste à regrouper les cases adjacentes contenant des 1 par puissance de deux : on peut regrouper deux cases, quatre cases, huit cases (si le tableau comporte suffisamment de cases), etc.

bc

	V	**00**	**01**	**11**	**10**
a	**0**	1	0	1	0
	1	1	1	1	0

L'objectif est d'**encercler tous les 1** en ayant **un minimum de bulles** avec **des bulles les plus grandes possible**. Plusieurs solutions équivalentes sont parfois possibles. Par exemple, on aurait pu obtenir la combinaison de bulles suivantes :

bc

	V	**00**	**01**	**11**	**10**
a	**0**	1	0	1	0
	1	1	1	1	0

Le nombre de bulles nous donne le nombre de termes présents dans l'expression. Nous avons trois bulles, l'expression comportera donc trois termes. Un terme est un groupe de variables séparées par un ET.

Une bulle permet de regrouper les variables sous forme de terme en respectant les règles suivantes :

- Si une variable à plusieurs valeurs dans la bulle, elle n'apparaît pas dans le terme ;
- Si une variable est toujours à 0 dans la bulle, elle apparaît dans le terme avec une complémentation ;
- Si une variable est toujours à 1 dans la bulle, elle apparaît dans le terme sans complémentation.

Commençons par observer la bulle verticale de gauche :

bc

	V	**00**	**01**	**11**	**10**
a	**0**	1	0	1	0
	1	1	1	1	0

- La variable *a* est à 0 et à 1 : elle n'apparaîtra pas dans le terme ;
- Les variables *b* et *c* sont toujours à 0 : elles apparaîtront dans le terme avec une complémentation.

Le terme associé à cette bulle est donc : $\overline{b}.\overline{c}$

Intéressons-nous maintenant à la bulle horizontale :

bc

	V	**00**	**01**	**11**	**10**
a	**0**	1	0	1	0
	1	1	1	1	0

- La variable *a* est toujours à 1 : elle apparaîtra dans le terme sans complémentation ;
- La variable *b* est toujours à 0 : elle apparaîtra dans le terme avec une complémentation ;
- La variable *c* est à 0 et à 1 : elle n'apparaîtra pas dans le terme.

Le terme associé à cette bulle est : $a.\overline{b}$

Enfin, pour la dernière bulle :

bc

	V	**00**	**01**	**11**	**10**
a	**0**	1	0	1	0
	1	1	1	1	0

- La variable *a* est à 0 et à 1 : elle n'apparaîtra pas dans le terme ;
- Les variables *b* et *c* sont toujours à 1 : elles apparaîtront dans le terme sans complémentation.

Le terme associé à cette bulle est : b.c

L'expression complète qui contient les trois termes est donc :

$\mathbf{V = b.c + \overline{b}.\overline{c} + a.\overline{b}}$

Nous obtenons le même résultat qu'avec la simplification algébrique.

En utilisant la simplification **facultative** du NON-OU EXCLUSIF, on obtient :

$\mathbf{V = \overline{b \oplus c} + a.\overline{b}}$

Remarque :

Il est important de signaler qu'un tableau de Karnaugh doit être considéré comme enveloppé dans un cylindre. C'est-à-dire que les cases de la colonne de gauche sont considérées comme étant adjacentes aux cases de la colonne de droite et que les cases de la ligne du haut sont considérées comme étant adjacentes aux cases de la ligne du bas. Il est donc possible de regrouper dans une même bulle certaines cases de gauche avec certaines cases de droite et certaines cases du haut avec certaines cases du bas.

cd

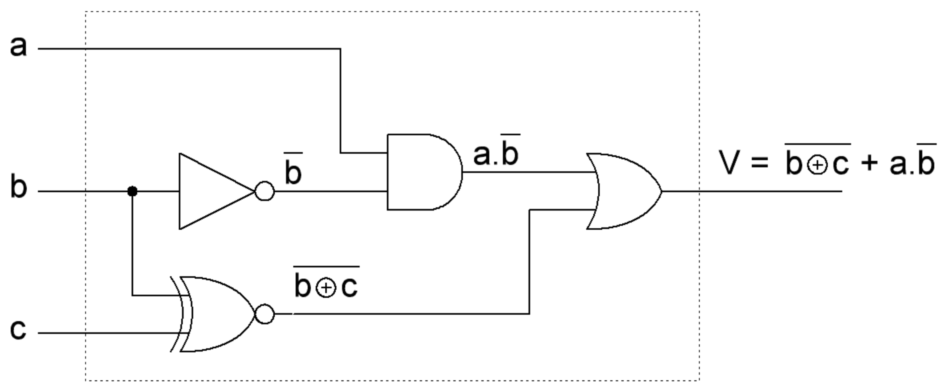
ab

X	**00**	**01**	**11**	**10**
00	1	0	0	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	0

$\mathbf{X = a.b.\overline{d} + \overline{b}.\overline{c}.\overline{d}}$

4.3. Dessiner un circuit combinatoire à partir de son expression

On peut maintenant représenter le circuit combinatoire à partir de son expression. Cette étape ne comporte aucune difficulté particulière. À partir de l'expression simplifiée, il suffit d'effectuer les connexions entre les différentes portes afin d'obtenir le résultat attendu en sortie.



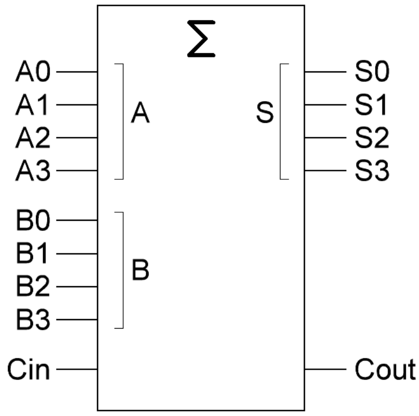
5. Les principaux circuits combinatoires

5.1. L'additionneur binaire parallèle

Un additionneur binaire parallèle permet d'additionner deux nombres binaires. Il possède généralement les entrées-sorties suivantes :

- Deux entrées parallèles sur *n* bits contenant les nombres binaires à additionner ;
- Une retenue d'entrée ;
- Une sortie parallèle sur *n* bits contenant le résultat de l'addition ;
- Une retenue de sortie.

Exemple d'un additionneur sur 4 bits :



Si *A* = **1010**, *B* = **1100** et *Cin* = **1**, on obtient :

```
     1 ← Cin
  1010 ← A
+ 1100 ← B
 10111 ← S
 ↑
Cout
```

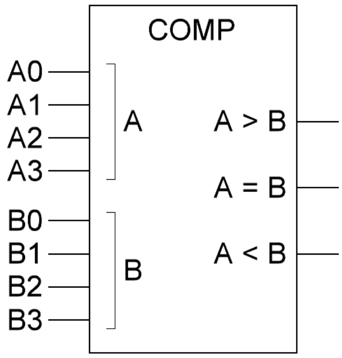
Remarque :
A, *B* et *S* sont des nombres binaires codés sur 4 bits. *A0*, *B0* et *S0* sont respectivement leurs bits de poids faible. *A3*, *B3* et *S3* sont respectivement leurs bits de poids fort.

5.2. Le comparateur

Un comparateur effectue la comparaison de deux nombres binaires afin de déterminer la relation existant entre ces deux nombres. Il possède généralement les entrées-sorties suivantes :

- Deux entrées parallèles sur *n* bits contenant les nombres binaires à comparer ;
- Une sortie indiquant si les deux nombres sont égaux ;
- Une sortie indiquant si le premier nombre est strictement inférieur au second ;
- Une sortie indiquant si le premier nombre est strictement supérieur au second.

Exemple d'un comparateur sur 4 bits :



Exemples :

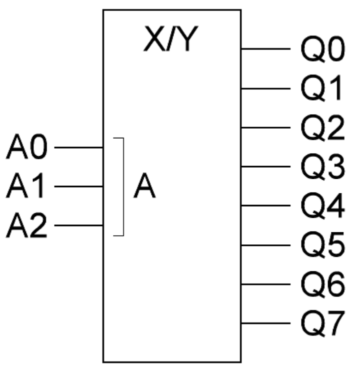
- Si $\boldsymbol{A} = \mathbf{0110}$ et $B = \mathbf{0011}$, alors $'A > B' = 1$, $'A = B' = 0$ et $'A < B' = 0$.
- Si $\boldsymbol{A} = \mathbf{0110}$ et $B = \mathbf{0110}$, alors $'A > B' = 0$, $'A = B' = 1$ et $'A < B' = 0$.
- Si $\boldsymbol{A} = \mathbf{0110}$ et $B = \mathbf{1110}$, alors $'A > B' = 0$, $'A = B' = 0$ et $'A < B' = 1$.

5.3. Le décodeur

Le décodeur permet de détecter une combinaison spécifique de bits présente sur ses entrées et de l'indiquer par un niveau spécifique de sortie. Il possède généralement les entrées-sorties suivantes :

- Une entrée parallèle sur *n* bits contenant le nombre à décoder ;
- Une sortie parallèle sur $\mathbf{2}^{\boldsymbol{n}}$ bits indiquant quel nombre est présent sur l'entrée.

Exemple d'un décodeur *1 parmi 8* (3 lignes d'entrée, 8 lignes de sortie) :



A	A2	A1	A0	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	0	0	**1**	0	0	0	0	0	0	0
1	0	0	1	0	**1**	0	0	0	0	0	0
2	0	1	0	0	0	**1**	0	0	0	0	0
3	0	1	1	0	0	0	**1**	0	0	0	0
4	1	0	0	0	0	0	0	**1**	0	0	0
5	1	0	1	0	0	0	0	0	**1**	0	0
6	1	1	0	0	0	0	0	0	0	**1**	0
7	1	1	1	0	0	0	0	0	0	0	**1**

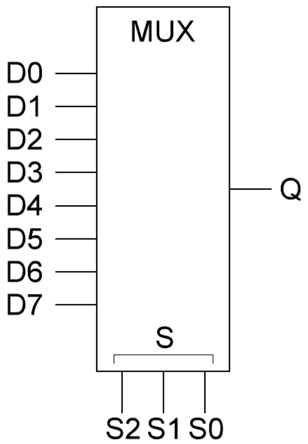
5.4. Le multiplexeur

Un multiplexeur permet d'envoyer les informations numériques de plusieurs sources vers une seule ligne de sortie. Il possède généralement les entrées-sorties suivantes :

- Une entrée parallèle de sélection sur *n* bits ;
- Une entrée parallèle de donnée sur 2^n bits ;
- Une ligne de sortie.

Les entrées de sélection permettent de sélectionner le bit de donnée qui sera copié sur la sortie.

Exemple d'un multiplexeur *8 vers 1* (3 lignes de sélection, 8 lignes de donnée, 1 ligne de sortie) :



S	S2	S1	S0	Q
0	0	0	0	D0
1	0	0	1	D1
2	0	1	0	D2
3	0	1	1	D3
4	1	0	0	D4
5	1	0	1	D5
6	1	1	0	D6
7	1	1	1	D7

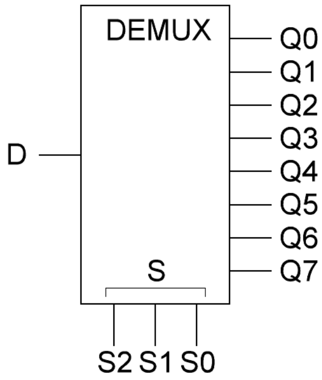
Les multiplexeurs sont souvent utilisés pour transformer une liaison parallèle en une liaison série. À l'intérieur d'un ordinateur, les données sont véhiculées par paquets de bits (par exemple 8, 16, 32 ou encore 64 bits). Lorsque l'on souhaite échanger des données avec des périphériques externes (imprimantes, appareils photo, etc.), il ne serait pas très pratique d'utiliser un câble ou un cordon comportant 64 fils. Il est donc préférable d'utiliser une liaison série qui permet de faire transiter les données sur un seul fil. Par exemple, la liaison USB (*Universal Serial Bus*) est une liaison série.

5.5. Le démultiplexeur

Un démultiplexeur effectue la fonction inverse d'un multiplexeur. Il permet de distribuer les informations numériques présentes sur une seule ligne vers un nombre donné de lignes de sortie. Il possède généralement les entrées-sorties suivantes :

- Une ligne d'entrée ;
- Une entrée parallèle de sélection sur *n* bits ;
- Une sortie parallèle sur 2^n bits recopiant la ligne d'entrée sur le bit sélectionné.

Exemple d’un démultiplexeur *1 parmi 8* (3 lignes de sélection, 1 ligne de donnée, 8 lignes de sortie) :



S	S2	S1	S0	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	0	0	**D**	0	0	0	0	0	0	0
1	0	0	1	0	**D**	0	0	0	0	0	0
2	0	1	0	0	0	**D**	0	0	0	0	0
3	0	1	1	0	0	0	**D**	0	0	0	0
4	1	0	0	0	0	0	0	**D**	0	0	0
5	1	0	1	0	0	0	0	0	**D**	0	0
6	1	1	0	0	0	0	0	0	0	**D**	0
7	1	1	1	0	0	0	0	0	0	0	**D**

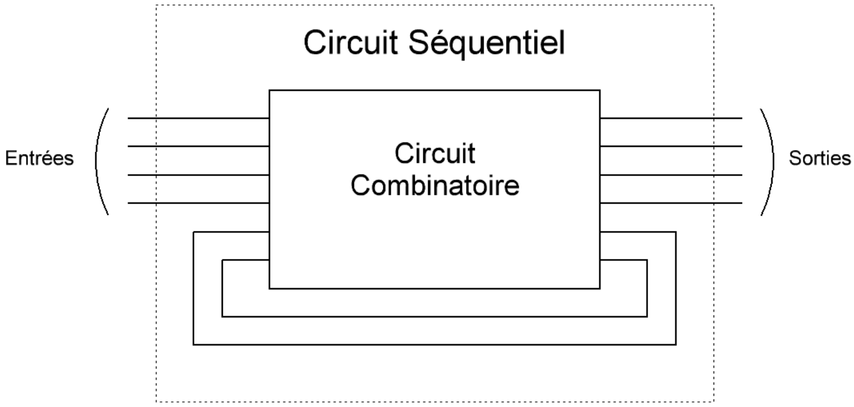
Remarque :

Si son entrée *D* est toujours à 1, alors un démultiplexeur se comporte comme un décodeur.

II. Les circuits séquentiels

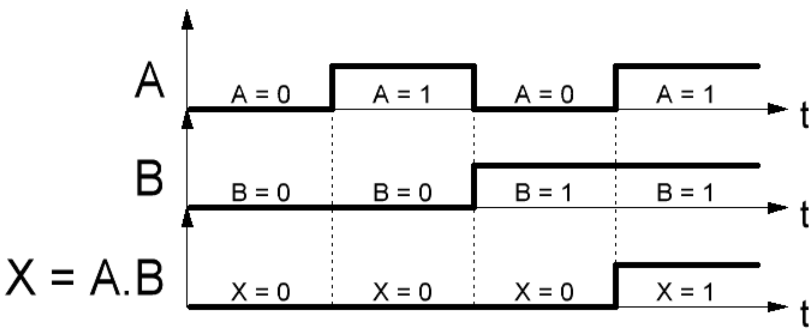
1. Définition

Un circuit séquentiel est un circuit logique dont l'état des sorties ne se déduit pas uniquement de l'état des entrées à un instant donné. Il est nécessaire de connaître l'histoire du circuit. Un circuit séquentiel se distingue d'un circuit combinatoire par la présence d'un retour des sorties vers les entrées.

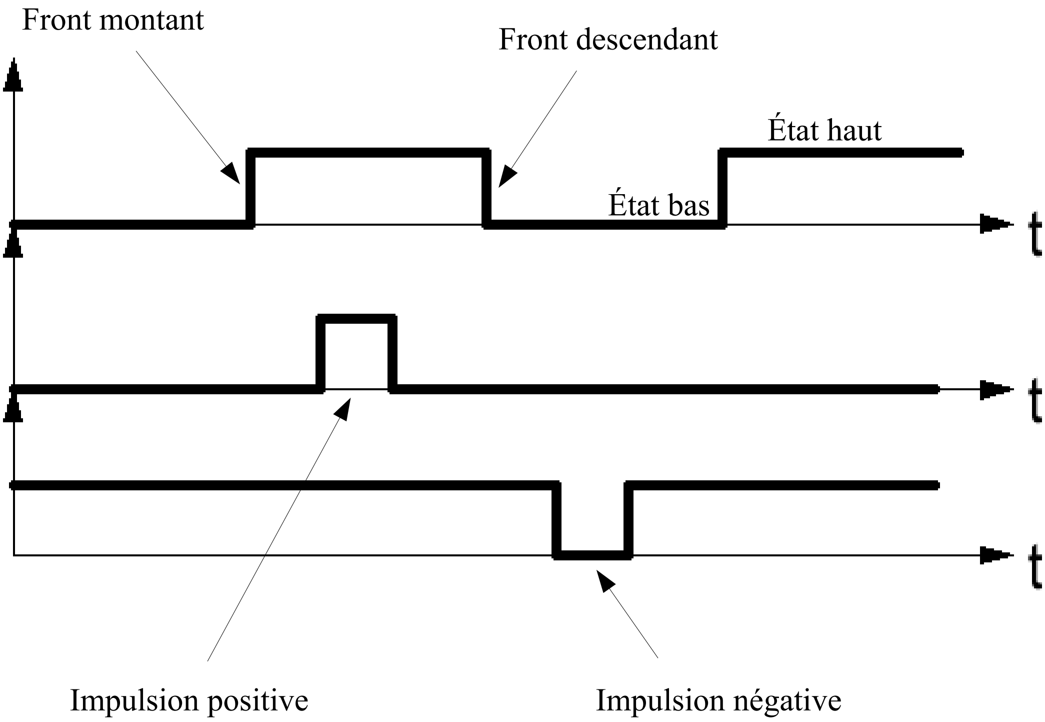


2. Les chronogrammes

Jusqu'à présent, nous avons vu différentes façons de décrire un circuit combinatoire : expression booléenne, tables de vérité et diagrammes de Karnaugh. Toutefois, ces représentations ne sont plus suffisantes quand une analyse temporelle devient nécessaire. C'est pourquoi nous allons maintenant introduire les chronogrammes. Ces derniers permettent de visualiser l'évolution d'un signal dans le temps. Par exemple, voici le chronogramme d'une porte ET :



Terminologie



3. Les bascules

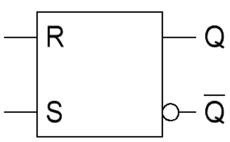
Les bascules sont les éléments de base des circuits séquentiels.

Il existe des bascules asynchrones et des bascules synchrones :

- Une bascule est asynchrone quand ses sorties réagissent immédiatement aux changements d'état des entrées ;
- Une bascule est synchrone quand ses sorties réagissent uniquement au moment de l'activation d'une entrée auxiliaire dite entrée d'horloge.

Seules les bascules principales seront détaillées dans ce cours.

3.1. La bascule RS asynchrone



R	S	Q	Remarque
0	0	q	État mémoire
0	1	1	Mise à 1 (*Set*)
1	0	0	Mise à 0 (*Reset*)
1	1	×	État interdit

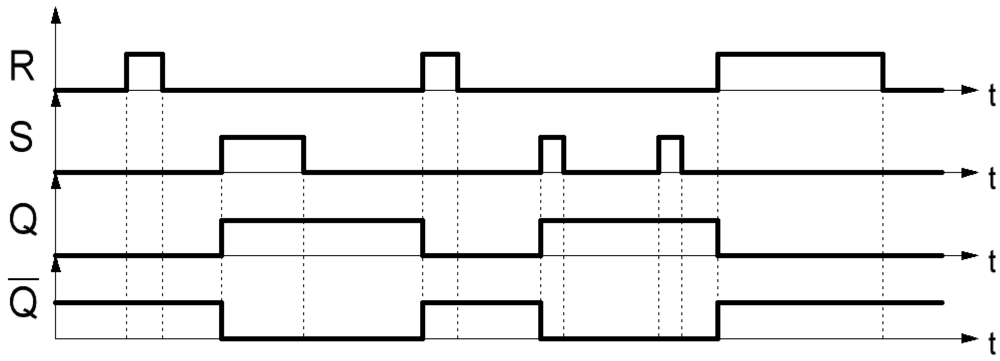
q = valeur de *Q* juste avant le passage de la bascule dans l'état mémoire.

Les entrées *R* (*Reset*) et *S* (*Set*) d'une bascule RS sont activent à l'état haut :

- *R* = 1 → *Reset* actif (la sortie *Q* est mise à 0) ;
- *S* = 1 → *Set* actif (la sortie *Q* est mise à 1).

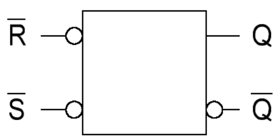
Si aucun *set* ou *reset* n'est actif, la sortie ne change pas. La bascule est dans son état mémoire.
On considérera que le *set* et le *reset* ne doivent jamais être actifs en même temps (état interdit).

Exemple d'un chronogramme d'une bascule RS asynchrone :



Il existe également une bascule RS aux entrées *set* et *reset* activent à l'état bas. Il s'agit de la bascule $\overline{RS}$:

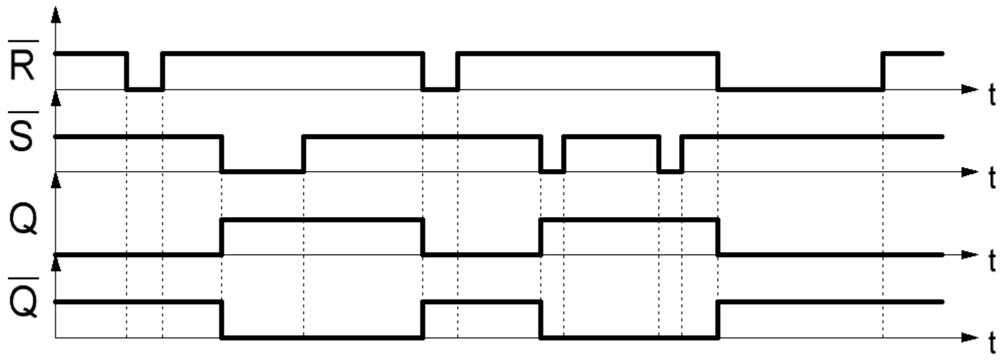
- $\overline{R}$ = 0 → *Reset* actif (la sortie *Q* est mise à 0) ;
- $\overline{S}$ = 0 → *Set* actif (la sortie *Q* est mise à 1).



$\overline{R}$	$\overline{S}$	Q	Remarque
0	0	×	État interdit
0	1	0	Mise à 0 (*Reset*)
1	0	1	Mise à 1 (*Set*)
1	1	q	État mémoire

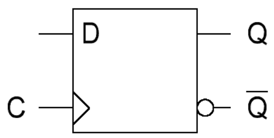
q = valeur de *Q* juste avant le passage de la bascule dans l'état mémoire.

Exemple d'un chronogramme d'une bascule $\overline{RS}$ asynchrone :



3.2. La bascule D synchrone

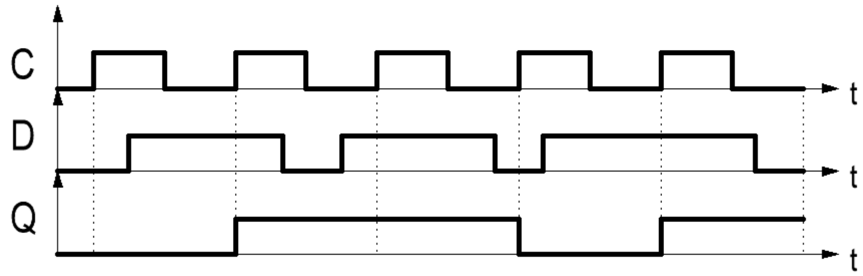
3.2.1. La bascule D synchrone sur front montant



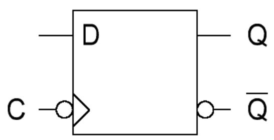
C	D	Q
↑	0	0
↑	1	1

La sortie *Q* recopie l'entrée *D* au moment d'un front montant de l'entrée d'horloge *C*. Les sorties sont modifiées uniquement lors de ce front (entre deux fronts montants, les sorties restent inchangées).

Exemple d'un chronogramme d'une bascule D synchrone sur front montant :



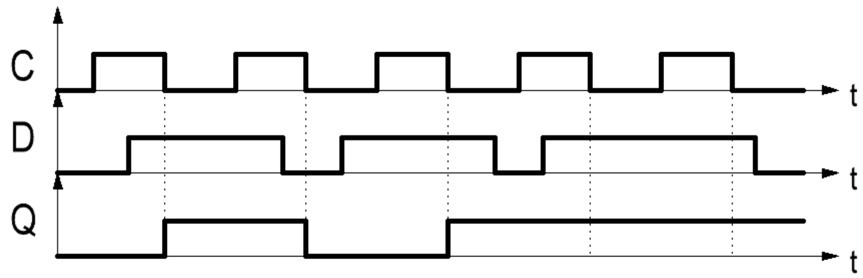
3.2.2. La bascule D synchrone sur front descendant



C	D	Q
↓	0	0
↓	1	1

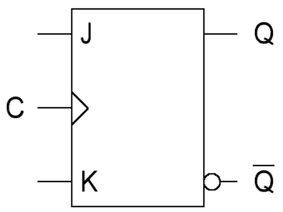
La sortie *Q* recopie l'entrée *D* au moment d'un front descendant de l'entrée d'horloge *C*. Les sorties sont modifiées uniquement lors de ce front (entre deux fronts descendants, les sorties restent inchangées).

Exemple d'un chronogramme d'une bascule D synchrone sur front descendant :



3.3. La bascule JK synchrone

3.3.1. La bascule JK synchrone sur front montant

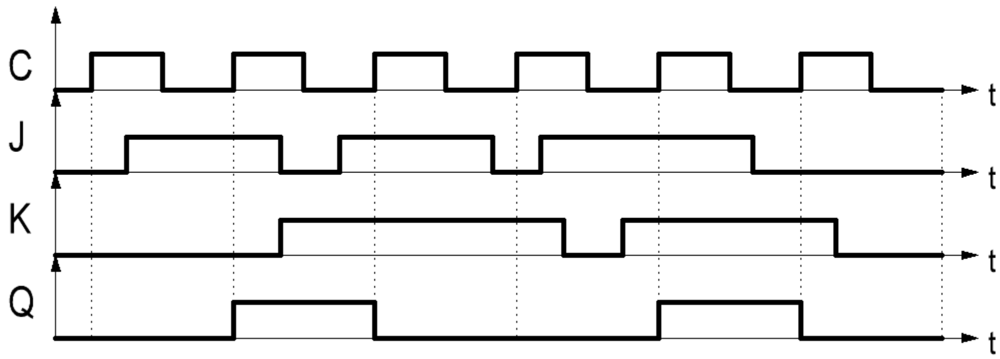


C	J	K	Q	Remarque
↑	0	0	q	État mémoire
↑	0	1	0	Mise à 0
↑	1	0	1	Mise à 1
↑	1	1	$\overline{q}$	Basculement

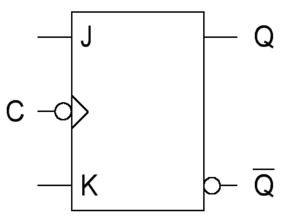
q = valeur de *Q* juste avant le front montant de l'entrée d'horloge.

Les sorties sont modifiées uniquement lors d'un front montant de l'entrée d'horloge *C* (entre deux fronts montants, les sorties restent inchangées).

Exemple d'un chronogramme d'une bascule JK synchronisée sur front montant :



3.3.2. La bascule JK synchrone sur front descendant

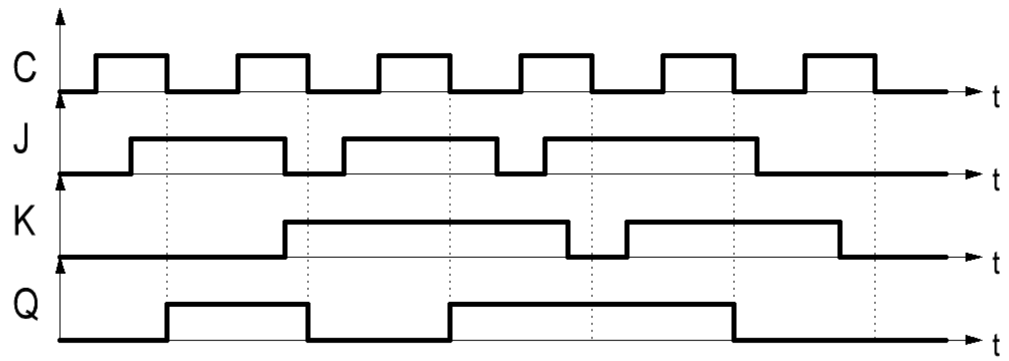


C	J	K	Q	Remarque
↓	0	0	q	État mémoire
↓	0	1	0	Mise à 0
↓	1	0	1	Mise à 1
↓	1	1	$\overline{q}$	Basculement

q = valeur de *Q* juste avant le front descendant de l'entrée d'horloge.

Les sorties sont modifiées uniquement lors d'un front descendant de l'entrée d'horloge *C* (entre deux fronts descendants, les sorties restent inchangées).

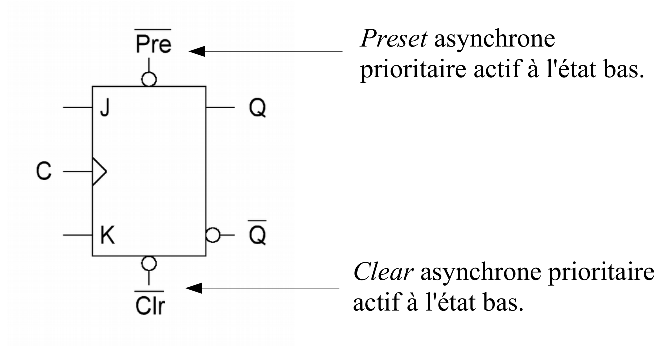
Exemple d'un chronogramme d'une bascule JK synchrone sur front descendant :

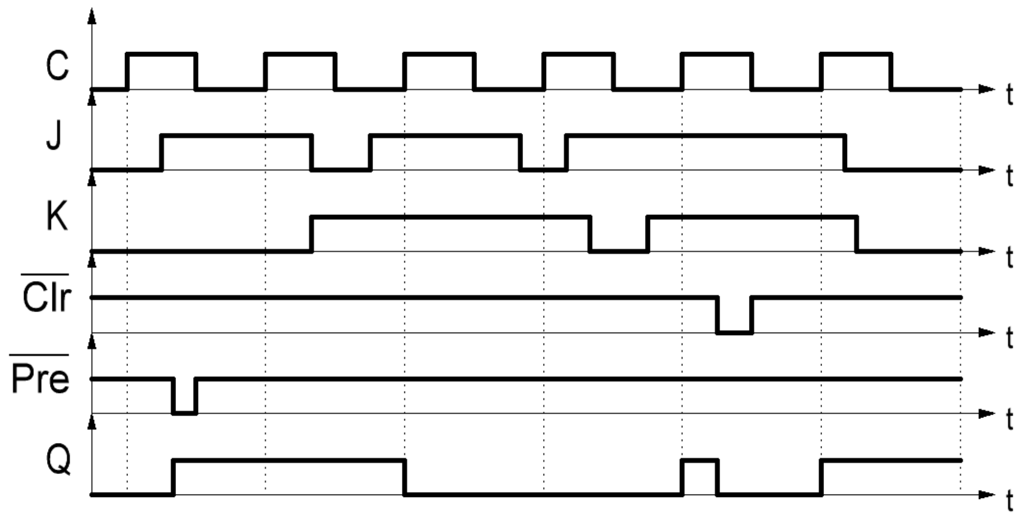


3.4. Entrées *preset* et *clear* asynchrones

La plupart des circuits intégrés de bascules D et JK sont également munis d'entrées asynchrones *preset* et *clear*. Ces entrées permettent d'affecter l'état de sortie des bascules indépendamment du signal d'horloge. Elles sont prioritaires et peuvent être actives à l'état haut ou à l'état bas.

Exemple :



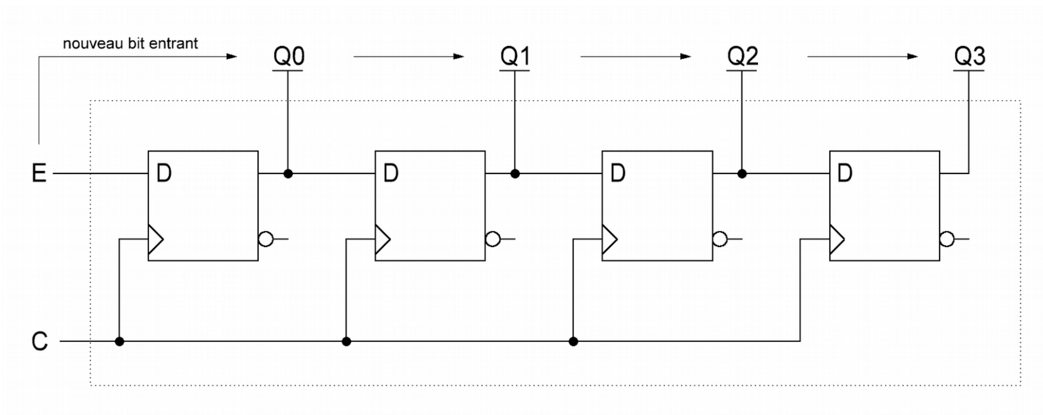


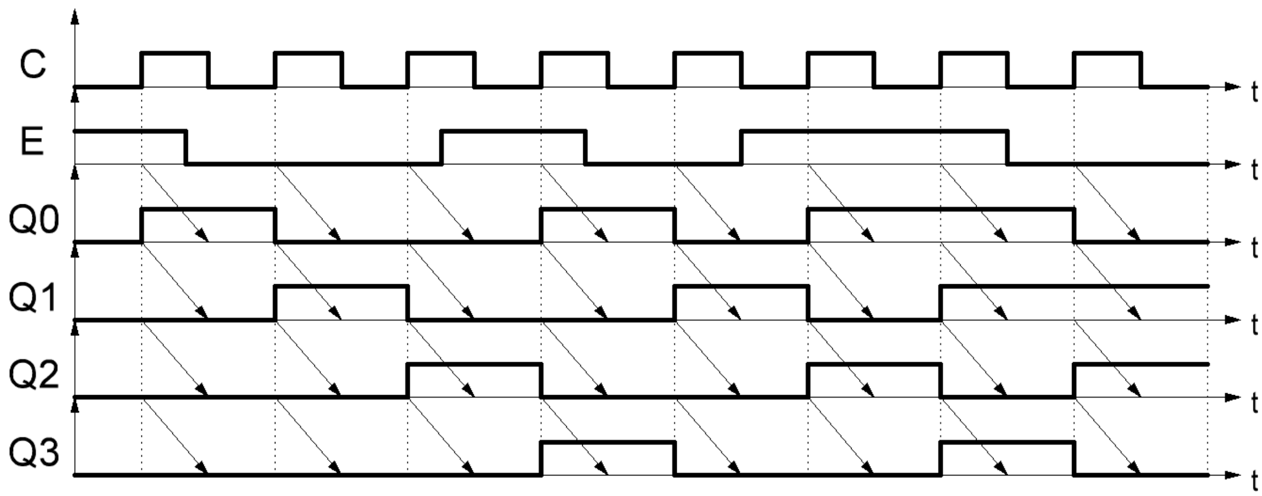
4. Les principaux circuits séquentiels

4.1. Le registre à décalage

Un registre à décalage est un circuit qui dispose d'une sortie parallèle sur *n* bits avec la possibilité d'effectuer un décalage sur l'ensemble de ses bits. Selon le type de registre, le décalage peut s'effectuer vers la droite ou vers la gauche.

Exemple d'un registre à décalage sur 4 bits (constitué de bascules D) :





À chaque front montant sur l'entrée *C*, la sortie d'une bascule est recopiée sur la sortie de la bascule suivante. Le bit de poids faible *Q0* prend la valeur de l'entrée *E* (entrée de donnée série) et le bit de poids fort *Q3* est perdu.

Quelques exemples d'utilisation des registres à décalage :

- Conversion série-parallèle : si l'on envoie les bits d'un nombre les uns après les autres (c'est-à-dire en série) sur l'entrée *E*, au bout de quelques impulsions d'horloge, on les retrouvera présents en parallèle sur les sorties ;
- Multiplication ou division par 2^n selon le sens du décalage ;
- Ligne à retard numérique : si l'on veut retarder une donnée de *n* tops d'horloge, il suffit de l'envoyer sur l'entrée *E* et de la lire sur la sortie de la énième bascule.

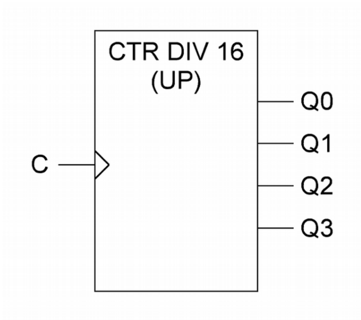
4.2. Le compteur

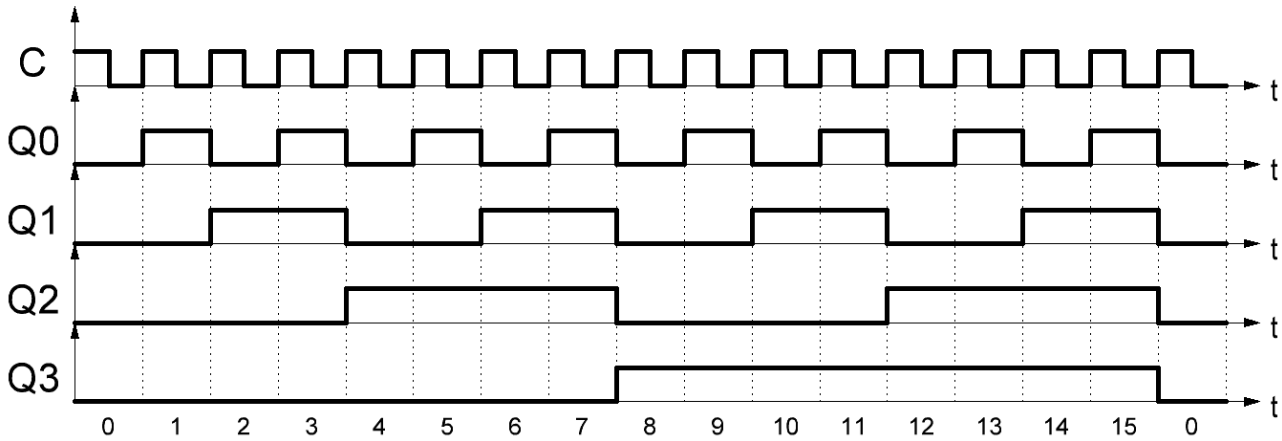
Un compteur est un circuit séquentiel qui, à chaque front d'horloge, incrémente un nombre binaire présent sur ses sorties.

On dit qu'un compteur est modulo *n* lorsqu'il compte de **0** à ***n* – 1**. Par exemple :

- Un compteur modulo 8 compte de 0 à 7 ;
- Un compteur modulo 10 compte de 0 à 9 ;
- Un compteur modulo 16 compte de 0 à 15.

Exemple d'un compteur modulo 16 :





On constate que pour un compteur, le basculement d'une sortie se fait au moment du **front descendant** de la sortie précédente :

- *Q1* doit basculer sur chaque **front descendant** de *Q0* ;
- *Q2* doit basculer sur chaque **front descendant** de *Q1* ;
- *Q3* doit basculer sur chaque **front descendant** de *Q2*.

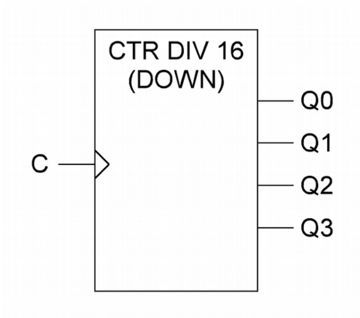
4.3. Le décompteur

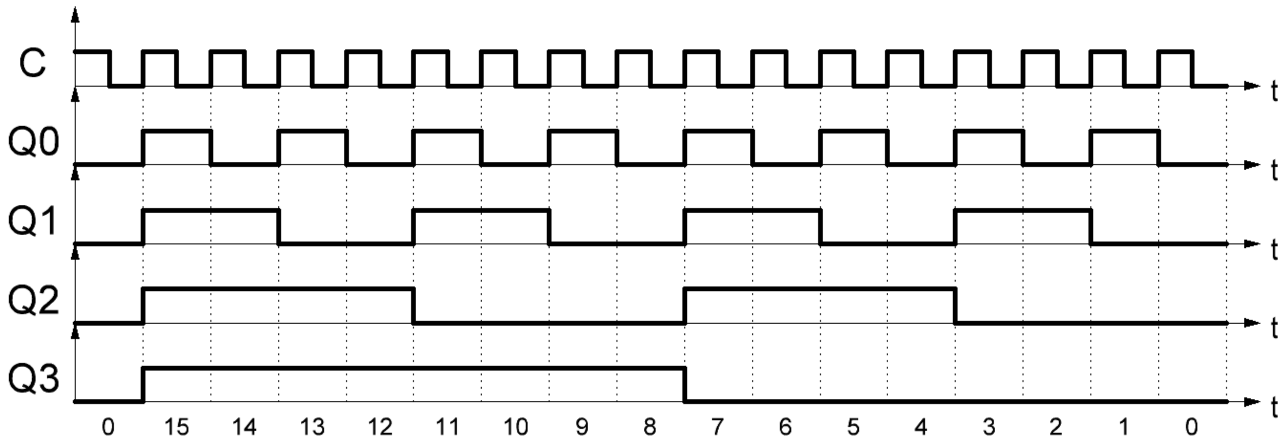
Un décompteur est un circuit séquentiel qui, à chaque front d'horloge, décrémente un nombre binaire présent sur ses sorties.

On dit qu'un décompteur est modulo *n* lorsqu'il décompte de ***n* – 1** à **0**. Par exemple :

- Un décompteur modulo 8 décompte de 7 à 0 ;
- Un décompteur modulo 10 décompte de 9 à 0 ;
- Un décompteur modulo 16 décompte de 15 à 0.

Exemple d'un décompteur modulo 16 :





On constate que pour un décompteur, le basculement d'une sortie se fait au moment du **front montant** de la sortie précédente :

- *Q1* doit basculer sur chaque **front montant** de *Q0* ;
- *Q2* doit basculer sur chaque **front montant** de *Q1* ;
- *Q3* doit basculer sur chaque **front montant** de *Q2*.

III. D’une grandeur analogique à un niveau logique

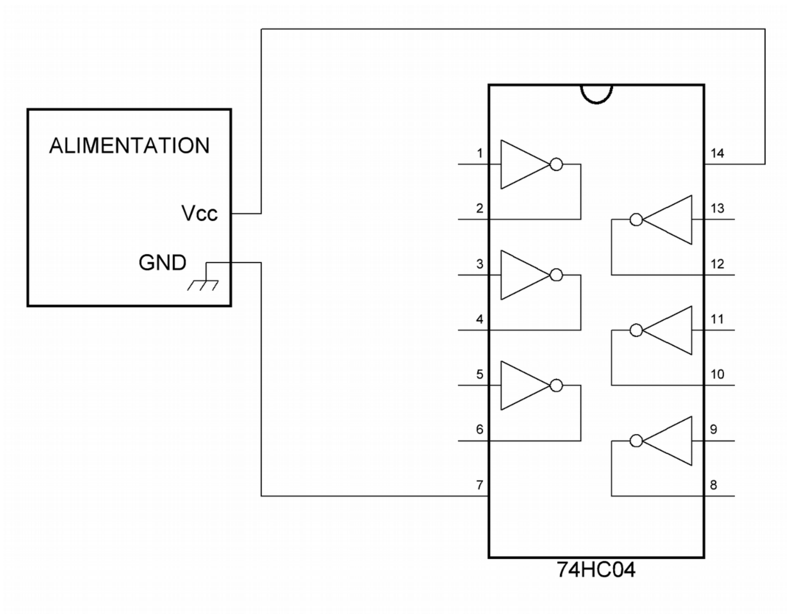
Jusqu’à présent, nous avons abordé uniquement le fonctionnement logique des circuits. Toutefois, d’un point de vue physique, ce ne sont pas des niveaux logiques (0 et 1) que l’on trouve en entrée ou en sortie des circuits, mais des niveaux de tension électrique.

Une tension électrique est une grandeur analogique qui se mesure en volts.

Les portes logiques et les bascules sont contenues dans des circuits intégrés. Les technologies utilisées pour fabriquer ces circuits ne sont pas toutes les mêmes. Les circuits intégrés peuvent donc se regrouper par familles de technologie. Il existe un certain nombre de familles (par exemple : TTL, CMOS, etc.). Chaque famille possède des caractéristiques plus ou moins différentes : temps de réponse, consommation d’énergie, source d’alimentation, performances, etc. Certaines familles sont compatibles entre elles, d’autres ne le sont pas.

Plusieurs chapitres seraient nécessaires afin de détailler le fonctionnement des principales familles et de leurs différences ; ce qui n’est pas l’objectif de ce cours. Dans une optique purement pédagogique, nous ne nous intéresserons donc pas à une technologie en particulier et nous n’aborderons que **des principes de base simplifiés**. Cette simplification sera tout à fait suffisante afin de comprendre le passage d’une grandeur analogique à un niveau logique.

Commençons par un exemple de circuit intégré contenant six portes NON : le **74HC04**.



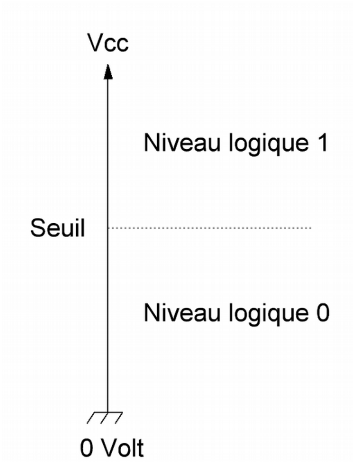
Ce circuit est alimenté par les broches 7 et 14 :

- La broche 7 est reliée à la masse (*ground*) qui vaut généralement zéro volt ;
- La broche 14 est reliée à la tension d’alimentation (que nous noterons Vcc).

La tension d’alimentation Vcc peut varier d’une famille de technologie à une autre. Elle peut être par exemple de 5 volts pour une famille, ou encore de 3,3 volts pour une autre.

Quand un circuit intégré est alimenté entre la masse et Vcc, il est possible d'appliquer sur l'entrée d'une porte (par exemple la broche 1), toutes les grandeurs analogiques comprises entre ces deux tensions. Une grandeur analogique étant une grandeur continue, cela représente donc une infinité de valeurs. La question qui se pose est alors la suivante : comment la porte fait-elle pour passer d'une grandeur analogique exprimée en volts à un niveau logique 0 ou 1 ?

Eh bien, ce passage se fait à l'aide d'un seuil. La porte considérera toute tension inférieure à ce seuil comme un niveau logique 0 et toute tension supérieure à ce seuil comme un niveau logique 1. La valeur du seuil peut varier en fonction des technologies.



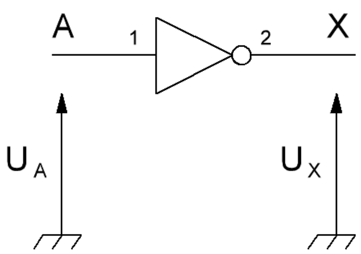
Comme énoncé précédemment, les choses ne sont pas aussi simples que cela, mais pour une première approche, cette simplification sera largement suffisante.

Pour la suite, nous considérerons que la tension de seuil est Vcc / 2.

Concernant la sortie d'une porte (par exemple la broche 2), la tension générée par le circuit sera de :

- zéro volt (la masse) si le niveau logique de sortie est 0 ;
- Vcc si le niveau logique de sortie est 1.

Afin d'illustrer ce qui vient d'être dit, intéressons-nous à l'exemple ci-dessous :



On considère les grandeurs suivantes :

- A = Variable logique d'entrée ;
- X = Variable logique de sortie ($X = \overline{A}$) ;
- $\mathbf{U_A}$ = Tension analogique d'entrée (tension mesurée entre la broche 1 et la masse) ;
- $\mathbf{U_X}$ = Tension analogique de sortie (tension mesurée entre la broche 2 et la masse).

Si par exemple, on applique une tension triangulaire à l'entrée de la porte (U_A), on obtient les chronogrammes suivants :