COURS LUMIÈRE
Primaire - Collège - Lycée

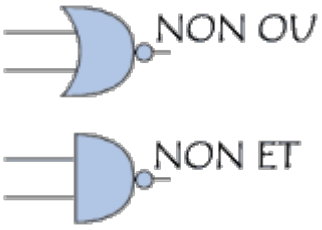
PREPA ESEO-COURS LUMIERE
Technologie & International

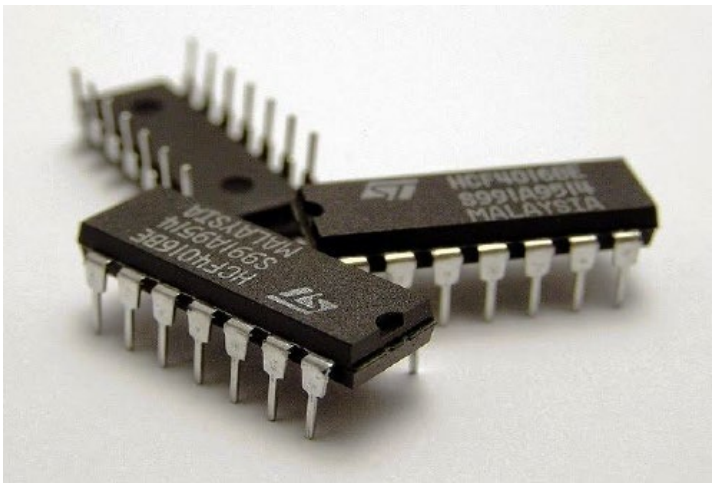
Semestre 3 et Semestre 4

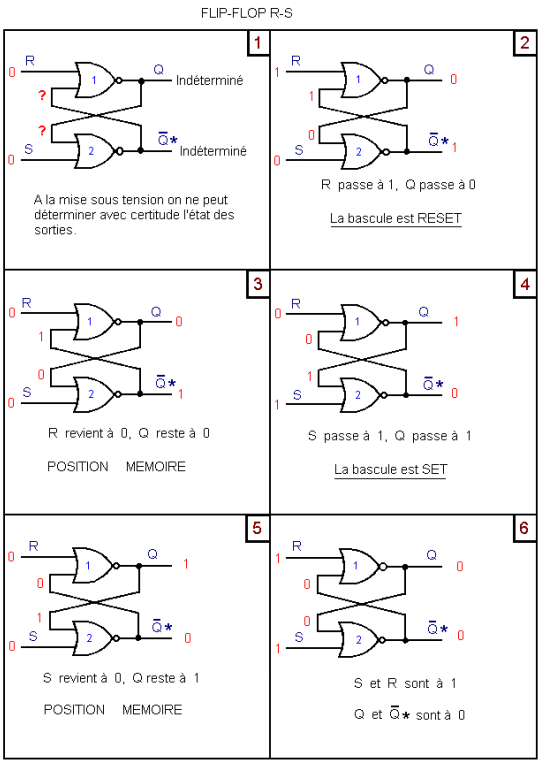
Electronique Numérique

Digital Electronics

(édition.-2025-2026)







Dr. Akpé Komi AGBOSSOU
Maître Assistant
au département Génie Electrique
Ecole Polytechnique de Lomé /
Université de Lomé.
Septembre 2026

Table des Matières

''L'enseignant est celui qui ouvre la porte de la connaissance ; c'est à l'étudiant de se lever, de la passer pour y accéder''.

Proverbe chinois traduit par AGBOSSOU.

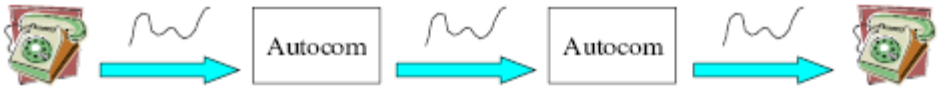
Dr Akpé AGBOSSOU
2026

Introduction

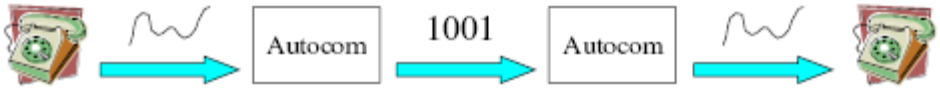
Vers le « tout numérique »

Un exemple : évolution technologique du téléphone

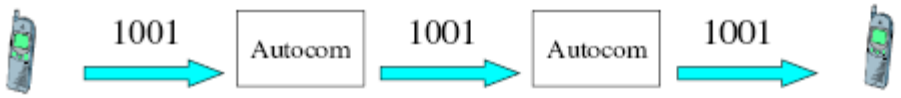
1900 : transmission analogique par fils, commutateurs électromécaniques.



1970 : numérisation du réseau entre centraux, la transmission reste analogique entre le poste de l'abonné et le central local.



1995 : portables,GSM : transmission numérique de l'appelant à l'appelé.

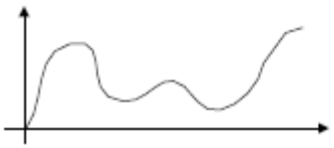


5

Université de Nice Sophia Antipolis - Département EEA - Laboratoire I3S DEUG 1 - premier semestre

Pourquoi cette évolution ?

Transmission analogique

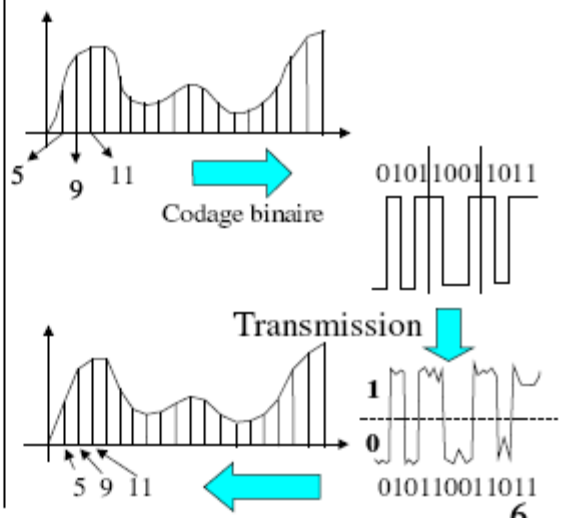


Ensemble continu de valeurs

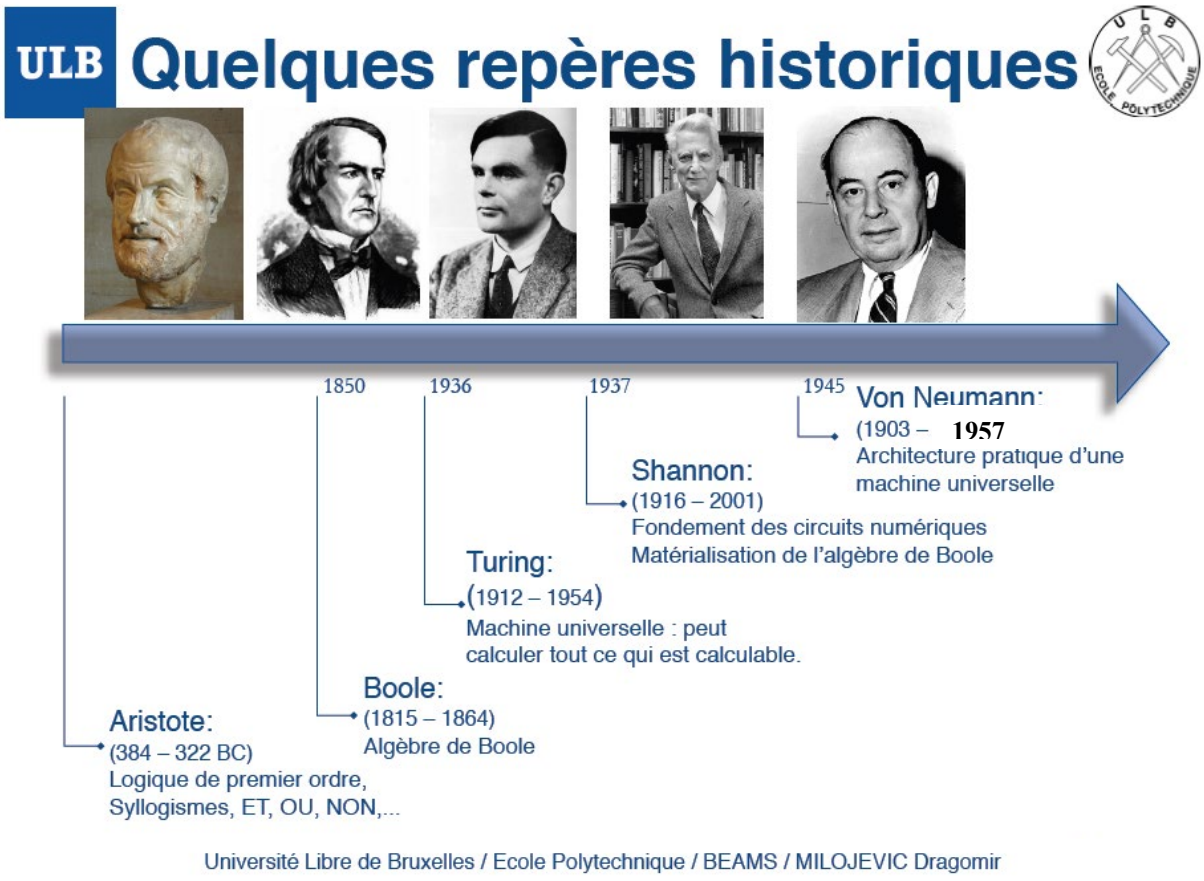
Transmission

Trop sensible aux fluctuations et aux parasites.

Transmission numérique



6



1.1 Représentations

Dans les sciences, dans les techniques, dans les affaires et dans la plupart des domaines, nous sommes amenés à utiliser des grandeurs de toutes sortes. Ces grandeurs sont mesurées, surveillées, enregistrées, transformées mathématiquement, observées et exploitées de diverses façons dans des systèmes réels différents. Il y a fondamentalement deux manières de représenter les valeurs numériques de ces grandeurs : **la manière analogique** et **la manière numérique**.

*There are two basic ways of representing the numerical values of the various physical quantities with which we constantly deal in our day-to-day lives. One of the ways is referred to as **analogue** and the other as **digital**.*

1.2 Représentation analogique (analogue)

Exemple le tachymètre d'une automobile : déviation d'une aiguille proportionnelle à la vitesse du véhicule. Les grandeurs analogiques varient de façon continue à l'intérieur d'une gamme de valeurs.

Compteur de tours

Thermomètre à mercure

Vielle montre à remontée manuelle

1.3 Représentation numérique (digital)

Une grandeur que l'on représente numériquement n'est pas proportionnelle à une autre grandeur, mais exprimée au moyen de symboles appelés chiffres. Exemple : une montre électronique. La représentation numérique évolue de façon discontinue. A cause de sa nature discontinue la représentation numérique évite des erreurs de lecture qui pourrait résulter d'une mauvaise interprétation d'une représentation analogique.

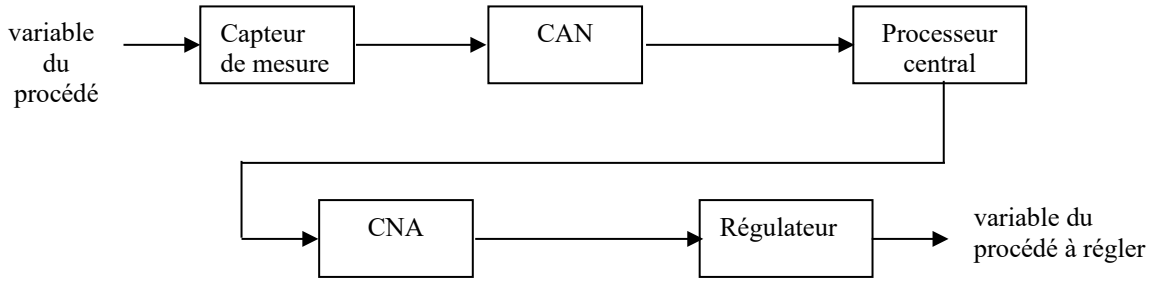
Montre numérique

Compteur numérique

1.4 Systèmes numériques et systèmes analogiques

Digital systems versus analogue systems

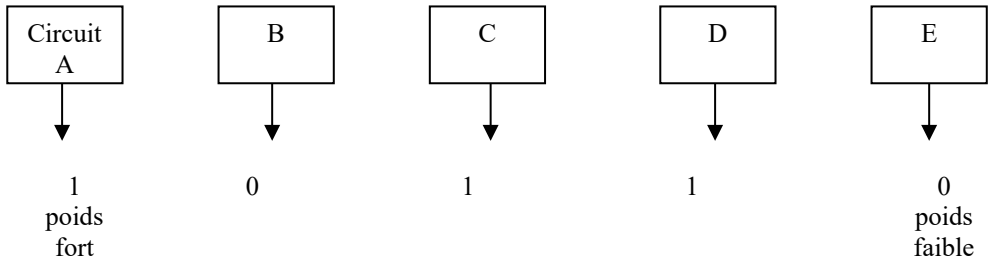
Un système numérique est une combinaison de dispositifs (électriques, mécaniques, photoélectriques, ….) organisés de manière à réaliser certaines fonctions qui traitent des grandeurs de nature numérique. Dans un système analogique, les grandeurs physiques sont principalement de nature analogique. Dans notre vie de tous les jours on trouve **des systèmes hybrides** dans lesquels interviennent à la fois des grandeurs numériques et analogiques et où sont effectués sans cesse des conversions numérique-analogique (CNA) ou des conversions analogique – numérique (CAN).
D'une manière générale, on peut dire que les systèmes numériques ont comme avantages d'être programmables, rapides et précis, et capables de garder des données en mémoires.



1.5 Représentation parallèle et représentation série

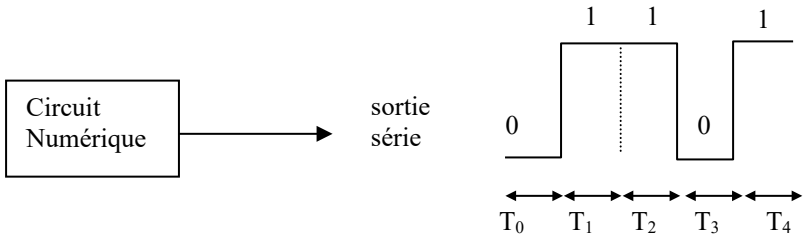
Il y a deux façons de représenter et de transmettre de l'information binaire : en parallèle ou en série.

Lorsqu'on choisit la représentation parallèle, chaque bit d'un nombre binaire est représenté par la sortie d'un circuit distinct.



Sortie parallèle

Lorsqu'on choisit la représentation série, la sortie d'un seul circuit suffit pour représenter tous les bits d'un nombre binaire.



Sortie Série

Remarque : la transmission parallèle nécessite beaucoup de circuits mais elle est plus rapide que la transmission série.

Système de numération (Number system)

2.1 Base d'un système de numération

La base d'un système de numération est le nombre de chiffres différents qu'utilise ce système.

2.1.1 Système Décimal (decimal number system)

C'est le système à base 10 le plus utilisé et qui nous est le plus familier. Il comprend dix chiffres différents : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

The decimal number system is a radix-10 number system and therefore has 10 different digits or symbols. These are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. ***(base of a number system is also called radix)***

- Exemple

$N = (1976)_{10}$. Ce nombre peut s'écrire : $N = 1x10^3 + 9x10^2 + 7x10^1 + 6x10^0$. Le nombre N est la somme de termes de forme générale $a.b^p$ où :

- a représente un chiffre de N,
- b représente la base 10,
- p le rang du chiffre et
- b^p le poids du chiffre.

De manière générale tout nombre décimal entier peut s'écrire $N = \sum_{i=0}^{i=n} a_i x10^i$. Dans le cas d'un nombre décimal présentant une virgule (15,43 par exemple) la virgule sépare la partie entière de la partie fractionnaire.

The place values of different digits in a mixed decimal number, starting from the decimal point, are 10^0, 10^1, 10^2 and so on (for the integer part) and 10^{-1}, 10^{-2}, 10^{-3} and so on (for the fractional part).

Lorsque le nombre est écrit sous forme d'un polynôme, la virgule indique la séparation entre les puissances de dix positives et les puissances de dix négatives :

$15,43 = 1x10^1 + 5x10^0 + 4x10^{-1} + 3x10^{-2}$.

Décimal en virgule fixe, base 10 (nombre réel)

La représentation d'un nombre réel se fait selon le modèle de l'exemple suivant :
NB nous utilisons le point décimal dans cet exemple.

$N = (1372.6450)_{10}$

1 3 7 2	•	6 4 5 0
Partie entière	Point décimal	Partie fractionnaire
10^3 10^2 10^1 10^0		10^{-1} 10^{-2} 10^{-3} 10^{-4}

N= $1x10^3 + 3x10^2 + 7x10^1 + 2x10^0$	+	$6x10^{-1} + 4x10^{-2} + 5x10^{-3} + 0x10^{-4}$

On obtient la forme généralisée suivante :

- Soit N un nombre en base r noté Nr
- On a n+1 chiffres de $a_0, \ldots, a_n$ pour la partie entière (indexe i)
- m chiffres de $b_1, \ldots, b_m$ pour la partie fractionnaire (indexe j)

Ce nombre s'écrit dans la base r :

$N = (a_n a_{n-1} \ldots a_1 a_0 . b_1 b_2 \ldots b_m)_r$

$N = a_n.r^n + a_{n-1}.r^{n-1} + \ldots + a_0.r^0 + b_1.\ r^{-1} + b_2.r^{-2} + \ldots + b_m.r^{-m}$ ou encore

$$N = \sum_{i=0}^{i=n} a_i.r^i + \sum_{j=1}^{j=m} b_j.r^{-j}$$

Partie entière — Partie fractionnaire

Dans cette écrire les indexes i et j désignent les poids des chiffres (en fait r^i et r^{-j})

- Pour la partie entière
    - i = n désigne le bit de poids le plus fort a_n (Most Significant Bit **MSB**)
    - i = 0 désigne le bit de poids le plus faible a_0 (Least Significant Bit **LSB)**

2.1.2 Système binaire (binary number system)

Ce système dit à base 2 comprend deux symboles qui sont les chiffres : 0 et 1. Chacun d'eux est aussi appelé bit ou élément binaire.

The binary number system is a radix-2 number system with '0' and '1' as the two independent digits. All larger binary numbers are represented in terms of '0' and '1'.

- **Exemple**

$N = (10110)_2$. Sous forme d'un polynôme $N = 1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = (22)_{10}$.

Le dernier chiffre de droite (0) est le chiffre de poids le plus faible son rang est 0 ; le chiffre le plus à gauche (1) est le chiffre de poids le plus fort, son rang est 4.

L'expression générale d'un nombre binaire, présentée sous forme d'un polynôme est $N = \sum_{i=0}^{i=n} a_i\ x2^i$ où a_i vaut 0 ou 1.

NB : Les coefficients a_i sont appelés digits en français, ou bits en anglais. a_0 est appelé bit de plus faible poids, et a_n bit de plus fort poids. 8 bits forment un octet , ou byte, 2 octets forment un mot, ou word et 4 octets forment un mot long, ou longword.

Definition

Bit is an abbreviation of the term 'binary digit' and is the smallest unit of information. It is either '0' or '1'.
A byte is a string of eight bits. The byte is the basic unit of data operated upon as a single unit in computers. A computer word is again a string of bits whose size, called the 'word length' or 'word size', is fixed for a specified computer, although it may vary from computer to computer. The word length may equal one byte, two bytes, four bytes or be even larger.

- **Exercice**

Consider an arbitrary number system with the independent digits as 0, 1 and X. What is the radix of this number system? List the first 10 numbers in this number system.
Solution
• The radix of the proposed number system is 3.
• The first 10 numbers in this number system would be 0, 1, X, 10, 11, 1X, X0, X1, XX and 100

2.1.3 Système octal (octal Number System)

Ce système dit à base 8, comprend huit symboles qui sont les chiffres : 0, 1, 2, 3, 4, 5, 6, 7.

The octal number system has a radix of 8 and therefore has eight distinct digits. The place values for the different digits in the octal number system are 8^0, 8^1, 8^2 and so on (for the integer part) and 8^{-1}, 8^{-2}, 8^{-3} and so on (for the fractional part).

- **Exemple**

N = $(6543)_8$. 3 est le chiffre de poids le plus faible et 6 est le chiffre de poids le plus fort. Sous forme polynomiale on a : N $(6543)_8 = 6x8^3 + 5x8^2 + 4x8^1 + 3x8^0 = (3427)_{10}$.

L'expression générale d'un nombre octal est N = $\sum_{i=0}^{i=n} a_i x8^i$ avec a_i l'un des huit chiffres.

2.1.4 Système hexadécimal (Hexadecimal Number System)

Ce système dit à base 16 comprend seize symboles, dix chiffres et six lettres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

The hexadecimal number system is a radix-16 number system and its 16 basic digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. The place values or weights of different digits in a mixed hexadecimal number are 16^0, 16^1, 16^2 and so on (for the integer part) and 16^{-1}, 16^{-2}, 16^{-3} and so on (for the fractional part). The decimal equivalent of A, B, C, D, E and F are 10, 11, 12, 13, 14 and 15 respectively, for obvious reasons.

- **Exemple**

N = $(AC53)_{16} = Ax16^3 + Cx16^2 + 5x16^1 + 3x16^0 = 10x16^3 + 12x16^2 + 5x16^1 + 3x16^0 = (44115)_{10}$.

L'expression générale d'un nombre hexadécimal est N = $\sum_{i=0}^{i=n} a_i x16^i$ avec a_i l'un des chiffres de la base.

The hexadecimal number system provides a condensed way of representing large binary numbers stored and processed inside the computer. One such example is in representing addresses of different memory locations. Let us assume that a machine has 64K of memory. Such a memory has 64K (= 2^{16} = 65 536) memory locations and needs 65 536 different addresses. These addresses can be designated as 0 to 65 535 in the decimal number system and 00000000 00000000 to 11111111 11111111 in the binary number system. The decimal number system is not used in computers and the binary notation here appears too cumbersome and inconvenient to handle. In the hexadecimal number system, 65 536 different addresses can be expressed with four digits from 0000 to FFFF. Similarly, the contents of the memory when represented in hexadecimal form are very convenient to handle.

2.2 Changement de base

2.2.1 Equivalent decimal (Decimal Equivalent)

The decimal equivalent of a given number in another number system is given by the sum of all the digits multiplied by their respective place values. The integer and fractional parts of the given number should be treated separately. Binary-to-decimal, octal-to-decimal and hexadecimal-to-decimal conversions are illustrated below with the help of examples.

2.2.2 Conversion d'un nombre décimal en un nombre d'un système d'une autre base

Problème

Soit N un nombre donné en base 10, écrire ce nombre dans un système de base b.

Première méthode :

Chercher le grand multiple de la plus grande puissance entière de b contenu dans N puis le retrancher de N ; recommencer le processus avec le reste obtenu et ainsi de suite.

Exemple : convertir N = 3786 en nombre hexadécimal (b = 16). Le tableau ci-dessous montre que la plus grande puissance de 16 contenue dans N est 16^2. Le plus grand multiple de 16^2 contenu dans N est 14.
On a $3786 = 14x16^2 + 202$. On recommence la même décomposition avec 202 et on trouve :
$N = 3786 = 14x16^2 + 12x16^1 + 10x16^0$.
Soit $N = 3786 = Ex16^2 + Cx16^1 + Ax16^0 = (ECA)_{16}$.

i	16^i
0	1
1	16
2	256
3	4096

Deuxième méthode

Diviser le nombre décimal à convertir par la base b et conserver le reste. Diviser encore le reste par b et conserver le reste. Répéter l'opération sur chaque quotient obtenu. Ecrire les restes successifs en commençant par le dernier, de la gauche vers la droite pour former l'expression de N dans le système de base b.

Exemples :

A. Décimal vers binaire /Partie entière

$(245)_{10} = (\ ?)_2$

Méthode de divisions successives :

Nombre à convertir	Base		Reste de la division
	245	: 2	1
	122	: 2	0
	61	: 2	1
	30	: 2	0
	15	: 2	1
	7	: 2	1
	3	: 2	1
	1	: 2	1
	0		

Sens de calcul ↓

Sens de lecture MSB → LSB ↑

245 : 2 = 122 reste 1

122 : 2 = 61 reste 0

…….

On trouve comme résultat $(245)_{10} = (1\ 1\ 1\ 1\ 0\ 1\ 0\ 1)_2$

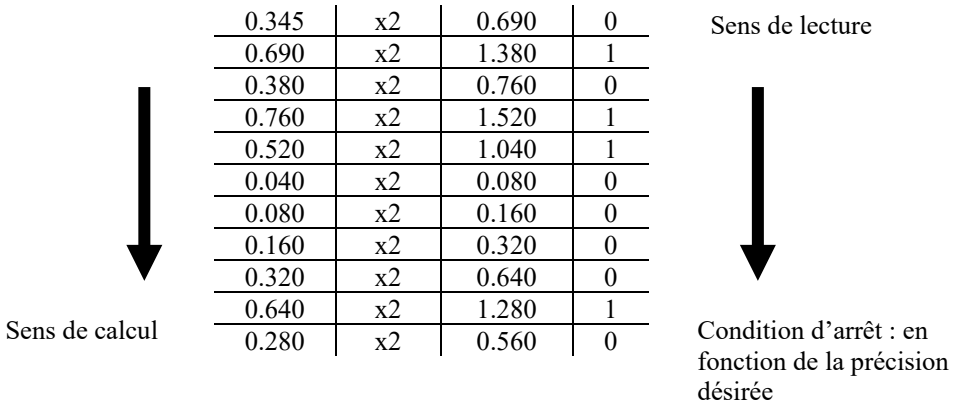
MSB
Bit de poids le plus

LSB
Bit de poids le plus faible

B. Décimal vers binaire /Partie fractionnaire

$(0.345)_{10} = (\ ?)_2$

Méthode de la multiplication successive

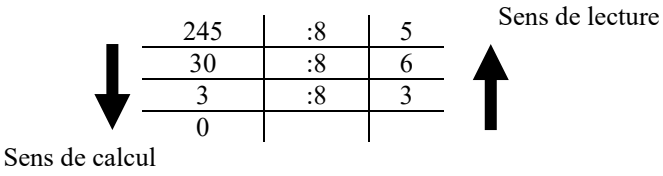


0.345	x2	0.690	0
0.690	x2	1.380	1
0.380	x2	0.760	0
0.760	x2	1.520	1
0.520	x2	1.040	1
0.040	x2	0.080	0
0.080	x2	0.160	0
0.160	x2	0.320	0
0.320	x2	0.640	0
0.640	x2	1.280	1
0.280	x2	0.560	0

Le résultat est le suivant : $(0.345)_{10} = (\ .01011000010)_2$

C. Décimal vers base 8 (octal)

$(245)_{10} = (\ ?)_8$

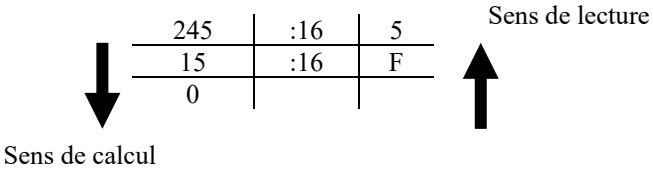


245	:8	5
30	:8	6
3	:8	3
0		

Le résultat est : $(245)_{10} = (\ 365)_8$

D. Décimal vers base 16 (hexadécimal)

$(245)_{10} = (\ ?)_{16}$



245	:16	5
15	:16	F
0		

Le résultat est : $(245)_{10} = (\ F5)_{16}$

2.2.3 Conversion d'un octal en un nombre binaire

Chaque symbole du nombre écrit dans le système octal est remplacé par son équivalent écrit dans le système binaire.

Exemple : $N = (257)_8 \Longrightarrow N = (\underbrace{010}_{2}\ \underbrace{101}_{5}\ \underbrace{111}_{7})_2. = 10101111$

An octal number can be converted into its binary equivalent by replacing each octal digit with its three-bit binary equivalent. We take the three-bit equivalent because the base of the octal number system is 8 and it is the third power of the base of the binary number system, i.e. 2. All we have then to remember is the three-bit binary equivalents of the basic digits of the octal number system.

2.2.4 Conversion d'un nombre binaire en un nombre octal

C'est l'opération inverse de la précédente. Il faut donc regrouper les 1 et 0 du nombre par trois en commençant par la droite puis chaque groupe est remplacé par le chiffre octal correspondant.

$N = (11001101111)_2 = (\underbrace{11}_{3}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{111}_{7})_2 \Longrightarrow N = (3157)_8.$

A binary number can be converted into an equivalent octal number by splitting the integer and fractional parts into groups of three bits, starting from the binary point on both sides. The 0s can be added to complete the outside groups if needed.

2.2.5 Conversion d'un nombre hexadécimal en nombre binaire

Chaque symbole du nombre écrit dans le système hexadécimal est remplacé par son équivalent écrit dans le système binaire.

A hexadecimal number can be converted into its binary equivalent by replacing each hex digit with its four-bit binary equivalent.

Exemple :

$N = (ECA)_{16} = (\underbrace{1110}_{E}\ \underbrace{1100}_{C}\ \underbrace{1010}_{A})_2$

2.2.6 Conversion d'un nombre binaire en un nombre hexadécimal

C'est l'inverse de la précédente opération. Il faut donc regrouper les 1 et 0 du nombre par quatre en commençant par la droite, puis chaque groupe est remplacé par le symbole hexadécimal correspondant.

A given binary number can be converted into an equivalent hexadecimal number by splitting the integer and fractional parts into groups of four bits, starting from the binary point on both sides. The 0s can be added to complete the outside groups if needed.

Exemple :

$N = (100001101111)_2 = (\underbrace{1000}_{8}\ \underbrace{0110}_{6}\ \underbrace{1111}_{F})_2 \quad \Rightarrow \quad N = (86F)_{16}.$

Dans le tableau suivant, on trouve les correspondances de différentes bases.

Conversions généralisées

Pour une conversion d'une base p vers une base q quelconque :

$(N)_p \longrightarrow (\ ?)_q$

Il est conseillé de passer par des bases intermédiaires 10 ou 2.

$(N)_p \longrightarrow (N)_{10} \longrightarrow (\ ?)_q$

Exercice : montrer que $(25.34)_8 = (41.2043220432\ldots.)_5$ en passant par la base 10.

Base			
16	10	8	2
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

Les Opérations Arithmétiques en Binaire

3.1. Addition

L'algorithme de l'addition des nombres binaires est le même que celui de l'addition des nombres décimaux. La table de l'addition est la suivante :

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 et **report de 1** (*nous verrons plus tard ce qu'on fait du report dans un calcul*).

Exercice : utiliser l'addition binaire pour calculer : 19 +11

3.2. Soustraction

Lorsque le diminuteur est plus petit que le diminuende, on aura un résultat positif. Dans le cas contraire, intervertir les termes et affecter le résultat du signe moins. La table de la soustraction est la suivante :

0 - 0 = 0
0 - 1 = 1 et **retenue de 1**, la retenue sera retranchée du chiffre de rang supérieur,
1 - 0 = 1
1 - 1 = 0.

Exercice : utiliser la soustraction binaire pour calculer : 13-7

3.3 Multiplication

La disposition des nombres est le même en binaire qu'en décimal, l'algorithme est aussi le même. La table de multiplication est la suivante :

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Exercice : utiliser la multiplication binaire pour calculer 9 x 5

Remarque :

- Multiplier un nombre binaire par $(10)_2$ revient à lui ajouter un 0 à droite. :
  - 1 x 10 = 10
  - 101 x 10 = 1010.

- Ajouter à un nombre son égal revient à le multiplier par $(10)_2$:
  - 1 + 1 = 10,
  - 101 + 101 = 1010.

3.4. Division

La disposition classique de la division est aussi valable en binaire.

Montrer que 101110111,0 / 110 = 111110,1.

Exercice : utiliser la division binaire pour calculer 39/6

3.5. La complémentation

3.5.1 Complément à 1 (1's complement).

En binaire on forme le complément à 1 d'un nombre en soustrayant de 1 chaque bit de ce nombre. Autrement dit : Complémenter un nombre binaire à 1 revient à inverser chacun de ses bits.

Pour un nombre : $A_n \ldots A_4A_3A_2A_1$ son complément à 1 est : $Y_n \ldots Y_4Y_3Y_2Y_1 = \overline{A_n} \ldots \overline{A_4}\ \overline{A_3}\ \overline{A_2}\ \overline{A_1}$

In the 1's complement format, the positive numbers remain unchanged. The negative numbers are obtained by taking the 1's complement of the positive counterparts. For example, +9 will be represented as 00001001 in eight-bit notation, and −9 will be represented as 11110110, which is the 1's complement of 00001001. Again, n-bit notation can be used to represent numbers in the range from $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ using the 1's complement format. The eight-bit representation of the 1's complement format can be used to represent decimal numbers in the range from −127 to +127.

3.5.2 Complément à 2 (2's complement)

Trouver le complément à 2 revient à trouver le complément à 1 et à ajouter 1 au résultat.

In the 2's complement representation of binary numbers, the MSB represents the sign, with a '0' used for a plus sign and a '1' used for a minus sign. The remaining bits are used for representing magnitude. Positive magnitudes are represented in the same way as in the case of sign-bit or 1's complement representation. Negative magnitudes are represented by the 2's complement of their positive counterparts. For example, +9 would be represented as 00001001, and −9 would be written as 11110111. Please note that, if the 2's complement of the magnitude of +9 gives a magnitude of −9, then the reverse process will also be true, i.e. the 2's complement of the magnitude of −9 will give a magnitude of +9. The n-bit notation of the 2's complement format can be used to represent all decimal numbers in the range from $+(2^{n-1}-1)$ to $-(2^{n-1}$-1). The 2's complement format is very popular as it is very easy to generate the 2's complement of a binary number and also because arithmetic operations are relatively easier to perform when the numbers are represented in the 2's complement format.

3.5.3 Soustraction par complément à 1

Exemple

```
  1101011101                1101011101
- 1011100111   ------->    +0100011000   complément à 1
  ?                         10001110101
                            ....>    + 1
                           ------------
                            0001110110
```

Au lieu d'ajouter un 1 à la fin ; on obtient le résultat si on prend le complément à 2 du nombre à soustraire.

Dans la plupart des cas **on représente les nombres négatifs sous la forme complément à 2**. De cette façon un calculateur additionne et soustrait automatiquement à l'aide d'un seul circuit dit additionneur. Faire attention au débordement (**overflow**).

3.6 Signe et valeur absolue (SVA) / Sign-Bit Magnitude

3.6.1 Représentation de nombre avec signe

Par convention :

Le bit 0 est ajouté au mot pour indiquer le signe + (plus).

Le bit 1 est ajouté au mot pour indiquer le signe – (moins).

Pour un mot de 8 bits : un bit est réservé pour le signe et 7 bits pour la valeur absolue. On peut ainsi représenter des valeurs de -127 à + 127. Car **1** 1111111 représente -127 et **0** 1111111 représente +127.

An n−bit binary representation can be used to represent decimal numbers in the range of $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$. That is, eight-bit representation can be used to represent decimal numbers in the range from −127 to +127 using the sign-bit magnitude format.

3.6.2 Addition Using the 2's Complement Method

Different steps to be followed to do addition in 2's complement arithmetic are summarized as follows:

1. Represent the two numbers to be added in 2's complement form.
2. Do the addition using basic rules of binary addition.
3. Disregard the final carry, if any.
4. The result of addition is in 2's complement form.

3.6.3 Subtraction Using 2's Complement Arithmetic

Subtraction is similar to addition. Adding 2's complement of the subtrahend to the minuend and disregarding the carry, if any, achieves subtraction. The process is illustrated by considering six different cases:

1. Both minuend and subtrahend are positive. The subtrahend is the smaller of the two.
2. Both minuend and subtrahend are positive. The subtrahend is the larger of the two.
3. The minuend is positive. The subtrahend is negative and smaller in magnitude.
4. The minuend is positive. The subtrahend is negative and greater in magnitude.
5. Both minuend and subtrahend are negative. The minuend is the smaller of the two.
6. Both minuend and subtrahend are negative. The minuend is the larger of the two.

3.6.2 Virgule flottante (Floating-Point Numbers)

Le format d'un nombre à virgule flottante est

$$N = \text{mantisse}.(\text{base})^{\text{exposant}}$$

Floating-point notation can be used conveniently to represent both large as well as small fractional or mixed numbers. This makes the process of arithmetic operations on these numbers relatively much easier. Floating-point representation greatly increases the range of numbers, from the smallest to the largest, that can be represented using a given number of digits. Floating-point numbers are in general expressed in the form:

$$N = m.b^e$$

where m is the fractional part, called the significand or mantissa, e is the integer part, called the exponent, and b is the base of the number system or numeration. Fractional part m is a p-digit number of the form (±d.dddd……dd), with each digit d being an integer between 0 and b – 1 inclusive. If the leading digit of m is nonzero, then the number is said to be normalized.

Les codes

Les systèmes numériques traitent des signaux qui représentent les symboles que sont les éléments binaires (bits). La correspondance entre signaux et bits est définie par un code binaire. Les codes binaires permettent aussi la détection et la correction des erreurs.

4.1 Codes pondérés

Dans les codes pondérés les positions des éléments binaires sont affectées d'un poids : $(N)_b = \sum_i a_i b^i$, expression dans laquelle :

- N est le nombre encore appelé mot,
- b est la base,
- i le rang
- a_i le chiffre de rang i et
- b^i le poids du chiffre a_i.

4.1.1 Code DCB : décimal codé binaire (BCD binary coded decimal en anglais)

Ce code conserve les avantages du système binaire naturel et du système décimal. A chaque chiffre d'un nombre du système décimal correspond son équivalent écrit dans le système binaire. Le code binaire du chiffre décimal le plus élevé 9 étant 1001, mot de quatre bits, chaque chiffre en notation binaire est codé avec quatre bits. D'où le tableau suivant :

Code décimal	0	1	2	3	4	5	6	7	8	9
Code DCB	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

- **Exemple**

Le nombre décimal 2583, s'écrit en DCB : $\underbrace{0010}_{2}\ \underbrace{0101}_{5}\ \underbrace{1000}_{8}\ \underbrace{0011}_{3}$

Remarques :

Il y a 16 différents mots binaires de 4 bits. Le code DCB n'en retient que 10. Les mots 1010, 1011, 1100, 1101, 1110 et 1111 n'étant pas utilisés, l'apparition de l'un d'eux signifie qu'une erreur s'est produite.

Très peu d'ordinateurs utilisent le code DCB car il nécessite plus de bits que le code binaire naturel, donc plus de place dans la mémoire de l'ordinateur. De plus les opérations en DCB sont plus complexes donc plus lentes.

$(58)_{10} = (111010)_2 = 01011000$ en DCB.

Effectuons en DCB la somme 148 + 23. On utilise l'addition binaire classique pour chaque colonne.

$$\begin{array}{r} 0001\ 0100\ 1000 \\ +\ \underline{0000\ 0010\ 0011} \\ 0001\ 0110\ 1011 \end{array}$$

$$(\ 1 \quad 6 \quad ?\ )_{10}$$

L'une des combinaisons interdites apparaît dans la colonne de droite. Cela se produit chaque fois que la somme de deux chiffres décimaux donne un résultat supérieur à 9.

Pour effectuer la correction, on montre qu'il faut ajouter 6 soit 0110 en DCB (cela correspond au fait que l'on saute les 6 combinaisons non valides) d'où :

0001 0100 1000
+0000 0010 0011
0001 0110 1011
\+ 0110
0001 0111 0001
$(\mathbf{1}71)_{10}$

4.1.2 Code ‘’plus 3’’ (ou majoré de 3)

C’est un code semblable au code DCB : 3 est ajouté à chaque chiffre décimal avant d’appliquer le code DCB. Ce qui donne le tableau de conversion suivant :

Code décimal	0	1	2	3	4	5	6	7	8	9
Code (+3)	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

Les combinaisons situées avant 0011 et celles situées après 1100 ne sont pas utilisées.

4.2 Codes non-pondérés

Dans les codes non-pondérés, les positions binaires ne sont pas affectées d’un poids. Aussi ne conviennent-ils pas aux opérations arithmétiques.

4.2.1 Code GRAY ou code binaire réfléchi

Dans les conversions d’une grandeur analogique (par exemple la position de l’axe d’un moteur) en grandeur numérique on a besoin d’un code dans lequel les grandeurs successives ne diffèrent que d’un caractère. Cela évite des erreurs de détection. Soit le passage de 7 à 8, par exemple. Selon la règle binaire naturelle, on passera de 0111 à 1000 : les quatre bits changent. Dans les états intermédiaires, s’ils ne changent pas tous en même temps, on peut détecter des valeurs erronées.

Le tableau ci-dessous donne l’équivalent en code gray des entiers de 0 à 15.

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110

12	1010
13	1011
14	1001
15	1000

Ce code binaire est dit réfléchi, car n -1 de ses bits peuvent être générés par réflexion comme l'illustre le tableau suivant :

00	000	0000
01	001	0001
11	011	0011
10	010	0010
	110	0110
	111	0111
	101	0101
	100	0100
		1100
		1101
		1111
		1110
		1010
		1011
		1001
		1000

Illustration de la façon d'écrire un nombre décimal en code Gray.

Binary–Gray Code Conversion

A given binary number can be converted into its Gray code equivalent by going through the following steps:

1. Begin with the most significant bit (MSB) of the binary number. The MSB of the Gray code equivalent is the same as the MSB of the given binary number.

2. The second most significant bit, adjacent to the MSB, in the Gray code number is obtained by adding the MSB and the second MSB of the binary number and ignoring the carry, if any. That is, if the MSB and the bit adjacent to it are both '1', then the corresponding Gray code bit would be a '0'.
3. The third most significant bit, adjacent to the second MSB, in the Gray code number is obtained by adding the second MSB and the third MSB in the binary number and ignoring the carry, if any.
4. The process continues until we obtain the LSB of the Gray code number by the addition of the LSB and the next higher adjacent bit of the binary number.

Example : Conversion Binary 1011 into Gray code 1110

Gray Code–Binary Conversion

A given Gray code number can be converted into its binary equivalent by going through the following steps:
1. Begin with the most significant bit (MSB). The MSB of the binary number is the same as the MSB of the Gray code number.
2. The bit next to the MSB (the second MSB) in the binary number is obtained by adding the MSB in the binary number to the second MSB in the Gray code number and disregarding the carry, if any.
3. The third MSB in the binary number is obtained by adding the second MSB in the binary number to the third MSB in the Gray code number. Again, carry, if any, is to be ignored.

4. The process continues until we obtain the LSB of the binary number.

*Example: Gray code 1110 into **Binary** 1011*

4.3 Code alphanumérique

Un ordinateur est prévu pour traiter des informations non numériques, c'est-à-dire qu'il doit reconnaître des caractères de l'alphabet, des caractères spéciaux des chiffres etc …. Tous ces éléments sont associés à un code appelé alphanumérique. On a pour habitude de considérer que le minimum de caractères d'un code alphanumérique est 92.

26 pour les lettres minuscules,
26 pour les majuscules,
10 pour les chiffres décimaux
et 30 pour les caractères spéciaux (<, ≤, % ,+ ….).

Le code le plus connu est le code ASCII qui est quasi universel.

4.3.1 Le code ASCII (American Standard Code for Information Interchange)

Ce code est une norme presque universelle dans les transmissions . Il comprend sept ou huit caractères. Le huitième caractère dit de parité sert à détecter les erreurs de transmission. On étudiera ce mode de détection d'erreurs plus loin.

Remarque : Le code ASCII est utilisé pour la transmission d'informations alphanumériques entre l'ordinateur et ses périphériques. La séquence de transmission est accompagnée de signaux indiquant le début (start) et la fin (stop) du message. Cette transmission s'effectue avec une vitesse exprimée en bauds : inverse de la durée (en seconde) nécessaire pour transmettre un bit. Exemple : 300 bauds = 300 bits / s.

				b_7	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	TC	SP	0	@	P	`	p
0	0	0	1	1	TC_1	DC_1	!	1	A	Q	a	q
0	0	1	0	2	TC_2	DC_2	"	2	B	R	b	r
0	0	1	1	3	TC_3	DC_3	#	3	C	S	c	s
0	1	0	0	4	TC_4	DC_4	¤	4	D	T	d	t
0	1	0	1	5	TC_5	TC_8	%	5	E	**U**	e	u
0	1	1	0	6	TC_6	TC_9	&	6	**F**	V	f	v
0	1	1	1	7	BEL	TC_{10}	’	7	G	W	g	w
1	0	0	0	8	FE_0	CAN	(	8	H	X	h	x
1	0	0	1	9	FE_1	EM	)	9	I	Y	i	y
1	0	1	0	10	FE_2	SUB	*	:	J	Z	j	z
1	0	1	1	11	FE_3	ESC	+	;	**K**	[	k	{
1	1	0	0	12	FE_4	IS_4	,	<	L	\	l	|
1	1	0	1	13	FE_5	IS_3	-	=	M	]	m	}
1	1	1	0	14	SO	IS_2	.	>	N	^	n	¯
1	1	1	1	15	SI	IS_1	/	?	O	_	o	DEL

Exemples de codes

Le code binaire est donné par $b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1$, où b_1 désigne le bit de poids faible.

Le code de :

U est 55_{16} soit 1010101 en binaire
F est 46_{16} soit 1000110
K est $4B_{16}$ soit 1001011

4.4 Codes détecteurs d'erreurs

Dans un code de quatre bits par exemple (de sept bits pour le code ascii) une erreur simple sur un bit peut faire apparaître un autre mot de code. Par exemple, en DCB le mot de code 0010 transmis par erreur donne 0110 (6, inversion du troisième bit due aux parasites de la ligne). Il sera interprété comme un 6 par le récepteur, mais ce sera une donnée erronée.

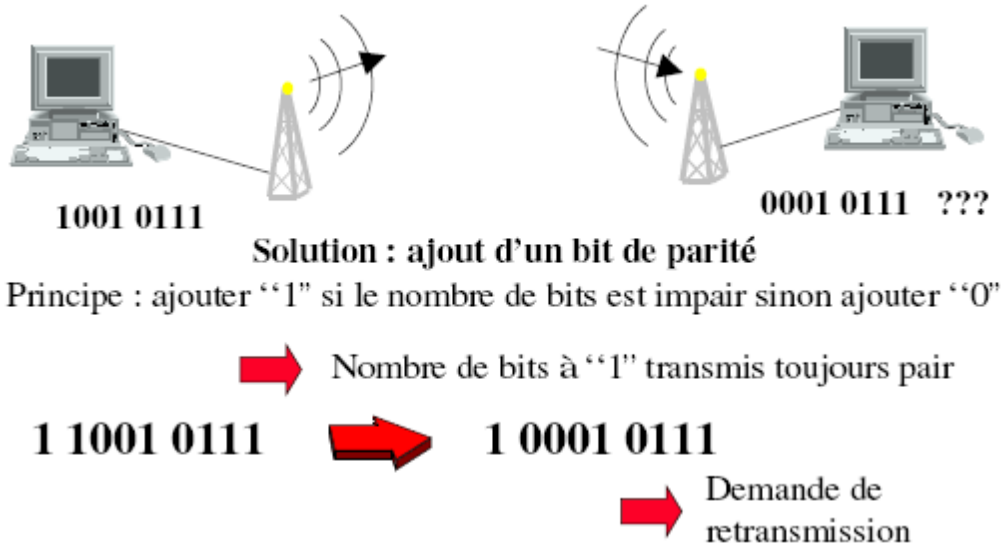
Il existe plusieurs façons de détecter ce type d'erreurs. La plus utilisée est celle du bit de parité paire ou impaire.

Exemple de code de parité paire en DCB :

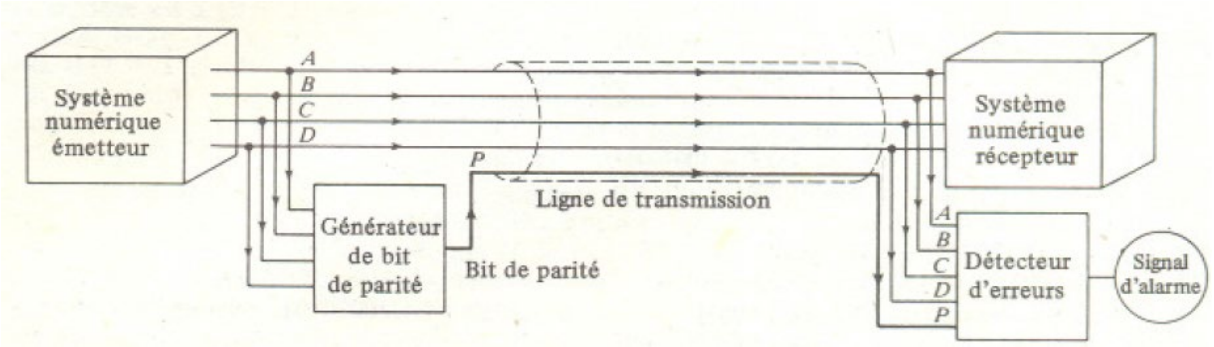
p est le bit de parité**. Il vaut 0 si le nombre de 1 du mot de code est pair et 1 autrement**.

		p
0	0000	0
1	0001	1
2	0010	1
3	0011	0
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	0

Si on a une erreur simple dans le mot de code ou dans le bit de parité, elle sera immédiatement détectée. Le récepteur pourra demander une transmission jusqu'à réception du mot correct.



Noter qu'il existe d'autres codes de détection d'erreurs.


Système numérique émetteur
A
B
C
D
P
Générateur de bit de parité
Bit de parité
Ligne de transmission
Système numérique récepteur
A
B
C
D
P
Détecteur d'erreurs
Signal d'alarme

Algèbre de Boole

5.1. Introduction

Inventée vers 1850, l'algèbre de BOOLE est un ensemble de variables à deux états, de valeurs de vérité 1 (vrai), 0 (faux), muni d'un nombre limité d'opérateurs : NON, ET, OU etc…. La manipulation de ces variables dites booléennes à l'aide de ces opérateurs donne des fonctions, booléennes elles aussi, car leur résultat est une variable booléenne.

5.2. Opérations ou fonctions de base de l'algèbre de BOOLE

5.2.1 Opération à une variable : opération NON (*NOT*)

Soit x une variable Booléenne, sa négation NONx ou son complément est noté $\overline{x}$. Les combinaisons possibles de la variable soumises à l'opérateur donnent une table qu'on nomme table de vérité.

Entrées	Sorties
x	$\overline{x}$
0	1
1	0

Table de vérité

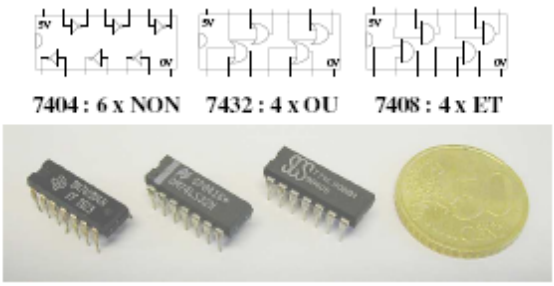
Symbole 

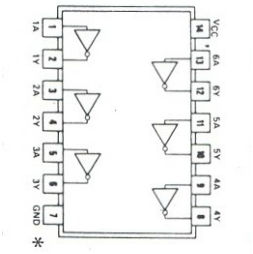
Remarques : Pour illustrer ce cours nous choisissons des circuits intégrés en technologie à l'aide de portes (de fabrication) TTL (Transistor Transistor Logic). Dans cette technologie toutes les fonctions sont implantées et logées dans des boîtiers rectangulaires normalisés comportant pour la plupart 14 broches. La tension d'alimentation est de 5V par rapport à la masse. Les tensions de sortie possèdent deux niveaux :

- Un niveau HAUT pour une tension de sortie comprise entre 3 et 5 Volts,
- Un niveau BAS pour une tension de sortie comprise entre 0 et 0,8 Volts.

Un circuit intégré avec des portes NON : le circuit de référence 7404

Brochage

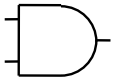




5.2.2 Opérations à deux variables

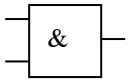
5.2.2.1 Opération ET (*AND*)

ET est désignée par AND (en anglais)
Nous illustrons le fonctionnement de cette porte par sa table de vérité ci-dessous



Symbole américain

entrées		sorties
x	y	x et y
0	0	0
0	1	0
1	0	0
1	1	1



Symbole européen

- x et y se note x.y par analogie avec la multiplication.

Un circuit intégré avec des portes AND : le circuit de référence 7408

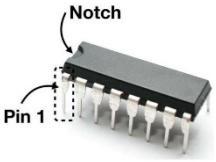
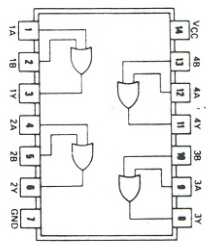
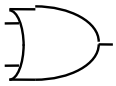


Figure 11-2 Pin 1 is Immediately Counterclockwise of the Notch
(Image Credit: Shutterstock/Cristian Storto)

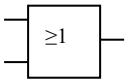
5.2.2.2 Opération OU (*OR*)

OU est désignée par OR (en anglais)
Sa table de vérité est représentée ci-dessous.





entrées		Sorties
x	y	x ou y
0	0	0
0	1	1
1	0	1
1	1	1



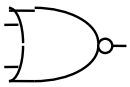
Symbole

x ou y se note x + y par analogie avec l'addition.
Un circuit intégré avec des portes OU : le circuit de référence 7432

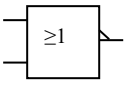
5.2.2.3 Opération NON-OU (*NOR*)

NON OU se désigne par NOR ou NI en (en anglais)

Sa table de vérité est donnée par :



entrées		sorties
x	y	$\overline{xouy}$
0	0	1
0	1	0
1	0	0
1	1	0

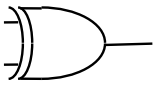


Un circuit intégré avec des portes NON OU: le circuit de référence 7402

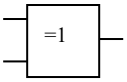
5.2.2.4 Opération OU exclusif (*XOR*)

OU exclusif se désigne par EXOR en (en anglais).

La table de vérité est :



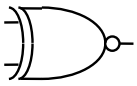
entrées		Sorties
x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0



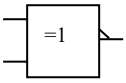
Un circuit intégré avec des portes OU EXCLUSIF : le circuit de référence 7486

5.2.2.5 Fonction égalité (*Ex-NOR*)

EXCLUSIVE-NOR (commonly written as EX-NOR) means NOT of EX-OR, i.e. the logic gate that we get by complementing the output of an EX-OR gate.



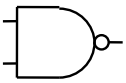
entrées		sorties
x	y	$\overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1



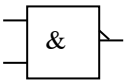
Un circuit intégré avec des portes EGALITE : le circuit de référence 74266

5.2.2.6 Fonction NON ET (*NAND*)

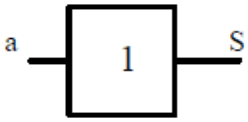
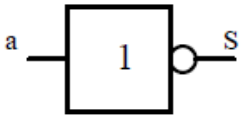
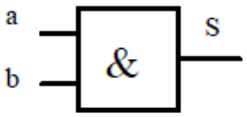
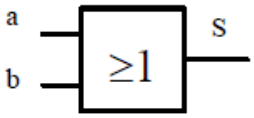
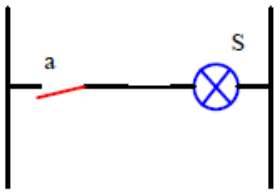
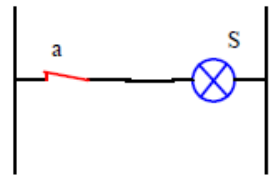
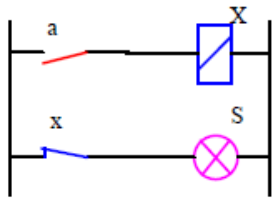
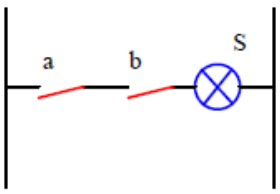
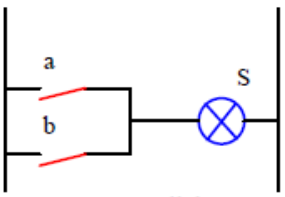
Appelée NAND en anglais



entrées		sorties
x	y	$\overline{x \ et \ y}$
0	0	1
0	1	1
1	0	1
1	1	0

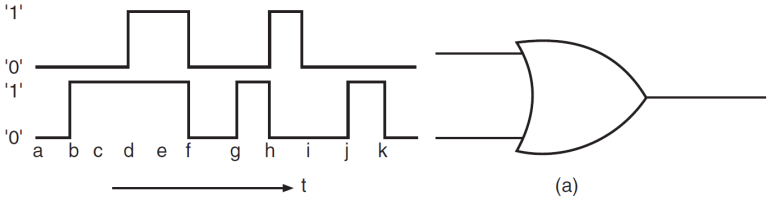


Un circuit intégré avec des portes NON ET: le circuit de référence 7400

OUI	NON	ET	OU
Amplification égalité	Inversion logique Complémentation	Produit logique	Somme logique
a | S 0 | 0 1 | 1	a | S 0 | 1 1 | 0	a | b | S 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	a | b | S 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
$S = a$	$S = \overline{a}$	$S = a \cdot b$ a ET b	$S = a + b$ a OU b
			
	utilisation d'un contact NC   Utilisation d'un relais	 Contacts en série	 contacts en parallèle

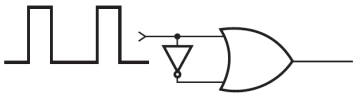
Exercices

1 How would you hardware-implement a four-input OR gate using two-input OR gates only?
2 Draw the output waveform for the OR gate and the given pulsed input waveforms of Fig. below.

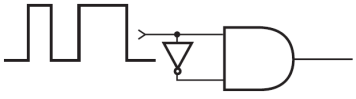


3 Show the logic arrangement for implementing a four-input AND gate using two-input AND gates only.

4 For the logic circuit arrangements of Figs below (a) and (b), draw the output waveform.



(a)



(b)

5.3. Axiomes ou lois fondamentales de l'algèbre de BOOLE

Soient A, B, C trois variables Booléennes on a :

5.3.1. Commutativité (*Commutative Laws*)

A.B = B.A

A + B = B + A

5.3.2. Associativité *(Associative Laws)*

A.(B.C) =(A B).C

A + (B + C) =(A + B) + C

Remarque : Puisque les regroupements ne changent les résultats on peut les omettre.

A.(B.C) =(A B).C s'écrira ABC

A + (B + C) =(A + B) + C s'écrira A + B + C

5.3.3. Distributivité *(Distributive Laws)*

Distributivité de ET par rapport à OU

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

Distributivité de OU par rapport à ET

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

5.3.4. Lois d'idempotence *Idempotent or Identity Laws*

A + A = A

A.A = A

5.3.5. Lois de complémentarité *Complementation Law*

$A + \overline{A} = 1$

$A.\overline{A} = 0$

5.3.6. Identités remarquables

1.A = A

1 + A = 1

0.A = 0

0 + A =A

5.3.7. Lois de distributivité interne

A + (B + C) = (A + B) + (A + C)

A.(B.C) = (A.B).(A.C)

5.4. Relations de base de l'algèbre booléenne

Complément d'une fonction logique

Le complément d'une fonction logique F(A,B,C) est obtenue en complémentant chaque variable de l'expression booléenne, en changeant tous les + en . , tous les . en +, tous les 1en 0 et tous les 0 en 1.

Equivalent and Complement of Boolean Expressions

The complement of a given Boolean expression is obtained by complementing each literal, changing all '.' to '+' and all '+' to '.', all 0s to 1s and all 1s to 0s.

Si F (A,B,C) = $\overline{A}$.B +A.$\overline{B}$ alors le complément $\overline{F}$(A,B,C) = (A+$\overline{B}$).($\overline{A}$.+B).

Expression duale

L'expression duale d'une fonction booléenne est obtenue en remplaçant tous les . par des +, tous les + par des ., tous les 0 par des 1 et tous les 1 par des 0 en gardant les variables inchangées (normale ou complémentée).

Si F (A,B,C) = $\overline{A}$.B +A.$\overline{B}$ alors F_{duale} = ($\overline{A}$+B).(A.+$\overline{B}$).

- (1) : $xy + x\overline{y} = x$ **(1')** : $(x + y)(x + \overline{y}) = x$
- (2) : $x + xy = x$ (*absorption*) **(2')** : $x(x + y) = x$
- (3) : $x + \overline{x}y = x + y$ **(3')** : $x(\overline{x} + y) = x.y$
- (4) : $xy + \overline{x}z + yz = xy + \overline{x}z$ (*Consensus Theorem*) **(4')** : $(x + y)(\overline{x} + z)(y + z) = (x + y)(\overline{x} + z)$
- (5) : $xy + x\overline{y}z = xy + xz$ **(5')** : $(x + y)(x + \overline{y} + z) = (x + y)(x + z)$.
- (6) : $xy + \overline{x}z = (x+z)(\overline{x} + y)$ *Transposition Theorem* **(6')** : $(x + y)(\overline{x} + z) = xz + \overline{x}y$

Remarque : Pour retrouver les relations (x') il faut remplacer les plus (+) par des points (.) dans les relations (x) : c'est la règle de dualité. Dans cette règle en général 1 est remplacé par 0 et vice versa.

Dual of a Boolean Expression

The dual of a Boolean expression is obtained by replacing all '.' operations with '+' operations, all '+' operations with '.' operations, all 0s with 1s and all 1s with 0s and leaving all literals unchanged.

- Réaliser les implantations à l'aide des différentes relations. Comparer les logigrammes.

5.5 Théorèmes de Morgan (Morgan's Laws)

Théorème 1 : La négation d'un produit de variables est égale à la somme des négations des variables.

- $\overline{xyz} = \overline{x} + \overline{y} + \overline{z}$

Preuve par la table de vérité.

Généralisation : $\overline{\prod_{i=1}^{n} a_i} = \sum_{i=1}^{n} \overline{a_i}$

Théorème 2 : La négation d'une somme de variables est égale au produit des négations des variables.

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

Preuve par la table de vérité.

Généralisation : $\overline{\sum_{i=1}^{n} a_i} = \prod_{i=1}^{n} \overline{a_i}$

5.6 Simplification algébrique des équations booléennes

Simplifier algébriquement une fonction booléenne c'est la développer, effectuer des mises en facteur et simplifier, selon les lois fondamentales et les relations démontrées précédemment. Simplifier une fonction revient donc à l'écrire à l'aide d'un nombre minimum de termes.

Exemples :
Simplifier les fonctions logiques suivantes :

- $F_1 = (a + b)(\overline{b} + c)(\overline{a} + c)$,
- $F_2 = (a + b)(\overline{a} + b)$,
- $F_3 = ab + \overline{c} + c(\overline{a} + \overline{b})$,
- $F_4 = bd + cd + \overline{c}d + \overline{a}b\,\overline{c}\,\overline{d} + ab\overline{c}$.

Practice your english

Combinational logic design

In contrast to analog circuits, which utilize continuous variables, in digital circuits the electrical variables are usually restricted to two discrete values, with reasonable tolerances for component variations and noise. Two values are arbitrarily designated as state 0 and state 1. They are frequently, but not always, controlled by a transistor driven out of its normal region of operation so that it acts approximately either as a short-circuit (ON) or an open-circuit (OFF). Complicated systems can be built by properly arranging large numbers of few elementary circuits, called logic gates, which implement desired logical operations of 0 and 1 states.
Combinational logic systems consists of gates with the outputs that depend on the inputs present at the time. In contrast, the outputs of the gates of sequential systems depend on not only on the inputs present but also on the past history, or sequence, of the inputs. Sequential systems have memory capability and are considered later and at the end of this course.

AND gate

An important logic function is called the AND operation. The AND operation on two logic variables, A and B, is represented as AB, read "A and B". The AND operation is also called logical multiplication. One way to specify a combinatorial logic system is to list all the possible combinations of the input variables and the corresponding output values. Such a listing is called a truth table. The truth table for AND operation of two variables is shown Figure *(refer to your course in French)*

Logic inverter (NOT gate)

The NOT operation on a logic variable is represented by placing a bar over the symbol for the logic variable. The symbol $\overline{A}$ is read as "not A" or as " A inverse. If A is 0, $\overline{A}$ is 1, and vice versa. Circuits that perform the NOT operation are called inverters. The truth table and circuit symbol for an inverter are shown in Figure *(refer to your course in French)*. The bubble placed at the output of the symbol is used to indicate inversion.

OR gate

The OR operation of logic variables is written as A+B, which is read as "A or B". The truth table and the circuit symbol for a two input OR gate are shown in Figure *(refer to your course in French)*. Notice that A+B is 1 if A or B (or both) are 1. The OR operation is also called logical addition.

Boolean Algebra

The mathematical theory of logic variables is called Boolean algebra, named for mathematician George BOOLE. One way to prove a Boolean algebra expression is to produce a truth table that lists all possible combinations of the variables and to show that both sides of the expression yield the same results.

De Morgan's laws

Two important results in Boolean algebra are De Morgan's laws, which are given by :

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

And

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

Another way to state these laws is: If the variables in a logic expression are replaced by their inverses, the AND operation is replaced by OR, the OR operation is replaced by AND, and the entire expression is inverted, the resulting logic expression yields the same values as before the changes.

NAND, NOR, XOR gates

Some additional logic gates are shown in figure *(refer to your course in French)*. The NAND gate is equivalent to an AND gate followed by an inverter. Notice that the symbol is the same as for an AND gate, with a bubble at the output terminal to indicate that the output has been inverted after the AND operation. Similarly, the NOR gate is equivalent to an OR gate followed by an inverter.
The exclusive OR (XOR) operation for two logic variables A and B is represented by $A \oplus B$.
Notice that the XOR operation yields 1 if A is 1 or if B is 1, but yields 0 if both A and B are 1. The XOR operation is also known as modulo-two addition.
A buffer has a single input and produces an output with the same value as the input. (Buffers are commonly used to provide large currents when a logic signal must be applied to a low impedance load).
The equivalent gate produces a high output only if the inputs have the same value. In effect, it is an XOR followed by an inverter as the symbol of figure *(refer to your course in French)* implies.

Quelques circuits pratiques

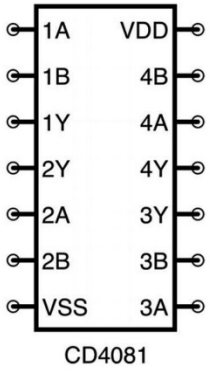


Figure 12-2 The Pinout of a CD4081 Chip

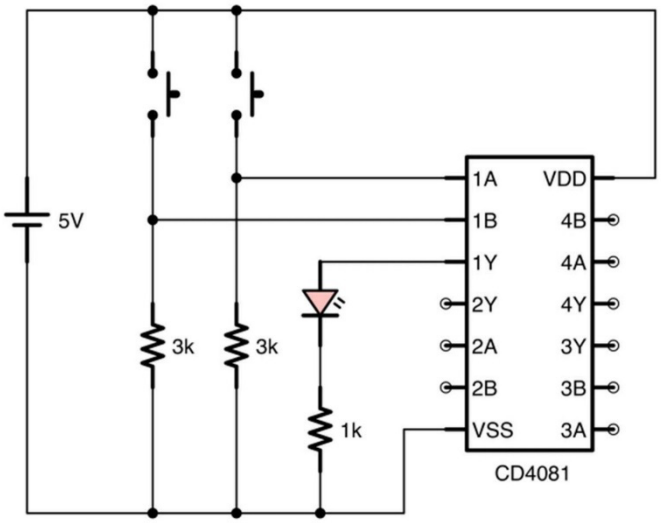


Figure 12-3 Example Circuit Using an AND Gate

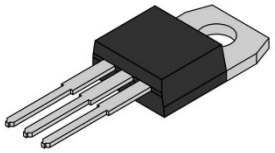


Figure 12-4 A 7805 Voltage Regulator in a To-220 Package

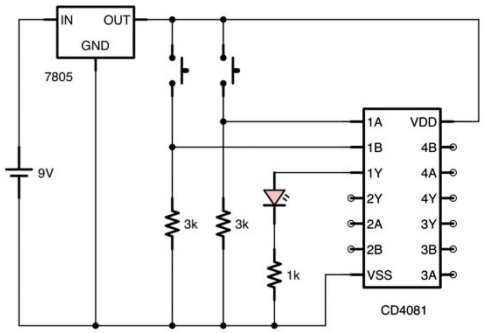


Figure 12-5 Logic Gate Circuit with a Voltage Regulator

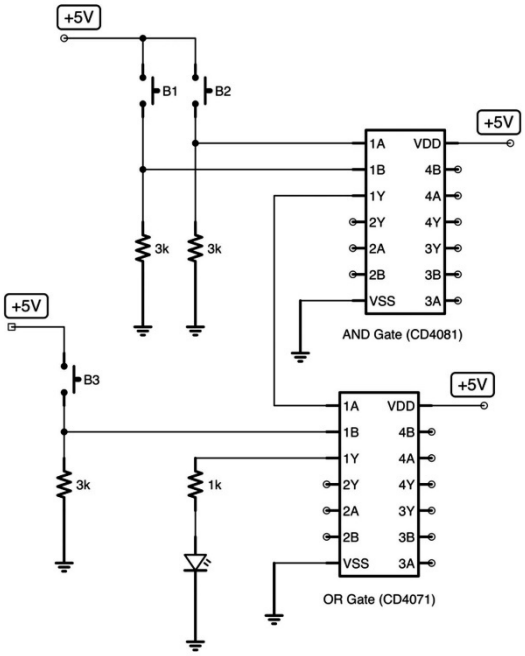


Figure 12-7 Multiple Logic Gates Combined in a Circuit

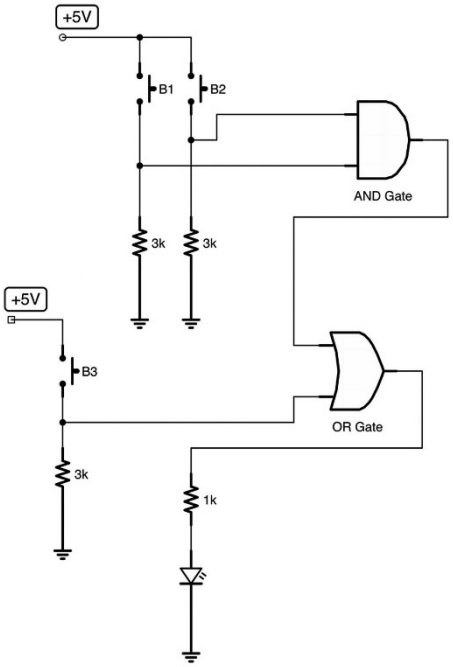


Figure 12-8 Logic Gates Represented as Shapes Instead of IC Pins

6 Représentation, simplification, implantation des fonctions logiques

6.1 Modes de représentation des fonctions logiques

6.1.1 Ecriture algébrique

Définition 1: On appelle minterme de n variables, un produit logique de ces dernières (complémentées ou non). Avec n variables, on construit 2^n mintermes, c'est-à-dire autant que de combinaisons possibles de n éléments prenant deux états.

Exemple : pour 2 variables a et b, voici les 4 mintermes : ab, $\overline{a}b$, a$\overline{b}$, $\overline{a}\overline{b}$.

Définition 2 : On appelle maxterme de n variables, une somme logique de ces dernières (complémentées ou non). De la même manière que pour les mintermes, on construit 2^n maxtermes avec n variables.

Exemple : pour 2 variables a et b, voici les 4 maxtermes : a+b, $\overline{a}+b$, $a+\overline{b}$, $\overline{a}+\overline{b}$.

Nous donnons ici un exemple d'écriture algébrique.

Soient les fonctions logiques S et C suivantes :

$S = x\,\overline{y}\,\overline{c} + \overline{x}y\overline{c} + \overline{x}\,\overline{y}c + xyc$

$C = xy\overline{c} + x\overline{y}c + \overline{x}yc + xyc$

La forme simplifiée de C s'écrit xy + xc + yc *(on le démontrera en utilisant la table Karnaugh)*.

6.1.2 Table de vérité (*Truth table*)

On peut également représenter ces fonctions sous formes d'une table de vérité. Les entrées sont dans l'ordre binaire naturel. Les fonctions S et C sont fonctions de trois variables. Il y a donc $2^3 = 8$ combinaisons possibles.

entrées			Sorties	
x	y	c	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

6.1.3 Table de Karnaugh (*Karnaugh Map*)

La table de vérité présente un inconvénient : le nombre de lignes croît avec le nombre de variables. Si n est le nombre de variables le nombre de lignes est 2^n. On contourne cette difficulté en plaçant les variables d'entrées aux entrées d'une table aussi carrée que possible. On obtient la table de Karnaugh à double entrées : une entrée pour les lignes et une pour les colonnes. Soit n variables, la table sera la plus carrée possible pour des entiers p et q tels p + q = n, lorsque p = q = $\frac{n}{2}$ pour n pair, ou $|p - q| = 1$ pour n impair. La table comportera 2^p colonnes et 2^q lignes. On numérote ensuite les colonnes et les lignes selon **le code binaire réfléchi (code Gray)**.
Applications aux fonctions S et C.

c \ xy	00	01	11	10
0				
1				

Fonction **S**

c \ xy	00	01	11	10
0				
1				

Fonction **C**

Exemples : représenter sous forme de table de Karnaugh les fonctions suivantes :

$Z_1 = \overline{a}\,\overline{b}\,\overline{d} + \overline{a}\,\overline{b}\,\overline{c}d + abc + \overline{a}bcd + \overline{a}b\overline{c}d$

$Z_2 = \overline{ac + bd}$

ab \ cd	00	01	11	10
00				
01				
11				
10				

Z_1

ab \ cd	00	01	11	10
00				
01				
11				
10				

Z_2

ab \ cd	00	01	11	10
00				
01				
11				
10				

6.1.4 Logigramme, diagramme synoptique ou schéma logique

Représenter S et C sous forme de logigramme

6.2 Formes canoniques

Les formes qui permettent de localiser chaque case d'une table de Karnaugh comportant un 1 ou un 0 ou chaque ligne d'une table de vérité comportant un 1 ou un 0 sont appelées formes canoniques. Ces formes sont en général

simplifiables. Elles sont utiles dans certains cas de simplification ou de recherche d'implantations symétriques pour faciliter les dépannages.
On distingue quatre formes canoniques :

- La première est une somme de produits à implantation par portes ET reliées à une porte OU.
- La deuxième est un produit de sommes à implantation par des portes OU reliées à une porte ET
- La troisième est la forme NON-ET
- La quatrième est la forme NON-OU.

Ces deux dernières formes sont implantées par un seul type de porte. Elles se déduisent des deux premières formes.

6.2.1 Première forme canonique : Somme de produits (*SOP sum of products*)

La première forme canonique d'une expression booléenne est composée d'une somme de **mintermes exclusivement**. Pour une expression donnée cette forme est unique.
Remarque : la somme de tous les mintermes de n variables vaut toujours 1 puisqu'il existe toujours un minterme de n variables valant 1.

Considérer la table de vérité ou la table de Karnaugh. A chaque 1 de la table de sortie, faire correspondre un produit de n variables d'entrée sous la forme normale lorsque la variable d'entrée est à 1, sous la forme complément si la variable d'entrée est à 0. S'il y a p 1, faire la somme logique de ces p produits. **Chaque produit doit contenir toutes les variables**. L'expression obtenue est en général simplifiable.

Applications aux tables de vérité de S et C.

S =

C =

Remarque : L'écriture des expressions logiques a cet inconvénient d'être assez longue. Chaque minterme parmi les 2^n de n variables correspond à un nombre représentant son ordre, c'est pourquoi on préfère parfois utiliser une écriture indiquant la liste classée des numéros des mintermes de la première forme canonique.

Exemple : F= abcd + $\overline{a}$bc$\overline{d}$ + $a\overline{b}c\overline{d}$ + $\overline{a}\overline{b}\overline{c}\overline{d}$ peut aussi s'écrire F(a,b,c,d) = Σ 0, 6, 10, 15. Certains auteurs écrivent aussi : Σ m(0, 6, 10, 15) = $m_0 + m_6 + m_{10} + m_{15}$.

Une définition analogue existe pour les fonctions logiques écrites sous formes de produit de sommes (maxtermes). Exemple : G(a, b, c, d) = Π 3, 6, 14, 15

6.2.2 Deuxième forme canonique : Produits de sommes (*POS Product of sums*)

La seconde forme canonique d'une expression booléenne est composée d'un produit de maxtermes exclusivement. Pour une expression donnée cette forme est unique.

Considérer la table de vérité ou la table de Karnaugh. A chaque 0 de la table de sortie, faire correspondre une somme de n variables d'entrée sous la forme normale lorsque la variable d'entrée est à 0, sous la forme complément si la variable d'entrée est à 1. S'il y a q 0, faire le produit logique de ces q sommes. Chaque somme doit contenir toutes les variables. L'expression obtenue est en général simplifiable.

Remarque : Le produit de tous les maxtermes de n variables vaut toujours 0 puisqu'il existe toujours un maxterme de n variables valant 0.

Applications aux tables de vérité de S et C.

S =

C =

6.2.3 Troisième forme canonique : Forme NON – ET (*NAND*)

On la déduit de la première forme canonique et elle conduit à des diagrammes n'utilisant que des portes NON ET (Nand). C'est la raison pour laquelle la porte logique Nand est aussi appelée **porte universelle.**

S =

On complémente deux fois S et on applique les théorèmes de Morgan

Implanter S

C =

Implanter C

6.2.4 Quatrième forme canonique : Forme NON – OU (*NOR*)

On la déduit de la deuxième forme canonique et elle conduit à des diagrammes n'utilisant que des portes NON OU. Pour cette raison la porte NOR est aussi une porte universelle.

S =

On complémente deux fois S et on applique les théorèmes de Morgan

Implanter S

Reprendre le même exercice pour C.

Universal Gates

OR, AND and NOT gates are the three basic logic gates as they together can be used to construct the logic circuit for any given Boolean expression. NOR and NAND gates have the property that they individually can be used to hardware-implement a logic circuit corresponding to any given Boolean expression. That is, it is possible to use either only NAND gates or only NOR gates to implement any Boolean expression. This is so because a combination of NAND gates or a combination of NOR gates can be used to perform functions of any of the basic logic gates. It is for this reason that NAND and NOR gates are universal gates.

6.3 Simplification par la table de Karnaugh

La table de Karnaugh permet d'opérer des simplifications qui conduisent le plus souvent à une expression minimale d'une fonction logique.

A Karnaugh map is a graphical representation of the logic system. It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions. Drawing a Karnaugh map from the truth table involves an additional step of writing the minterm or maxterm expression depending upon whether it is desired to have a minimized sum-of-products or a minimized product-of- sums expression.

Exemple 1

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	1	1

A

Ecrire A selon la première forme canonique

A =

Observer les termes correspondant aux 1 adjacents. Opérer les simplifications qui s'imposent.

A =

On peut alors énoncer la règle suivante :

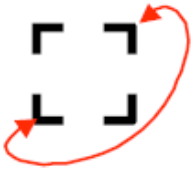
Première règle : On peut regrouper deux 1 adjacents dans une somme de produit en éliminant la variable qui change d'état. Il faut noter que l'adjacence existe aux extrémités de la table de Karnaugh.

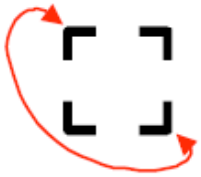
ab \ cd	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

B

Les bords haut et bas sont adjacents.

Les bords droit et gauche sont adjacents.





En conséquence les coins sont adjacents (deux à deux ou en combiné)

En regroupant les quatre 1 on obtient la forme simplifiée suivante B = $\overline{b}\,\overline{d}$

De la même façon dans le tableau ci-dessous par regroupement des quatre 1 on a : C = b$\overline{c}$

ab \ cd	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	0	0
10	0	0	0	0

C

On énonce alors la règle suivante :

Deuxième règle : on peut regrouper quatre 1 dans une somme de produits et éliminer les deux variables qui changent d'état. Cette règle se généralise à toutes les puissances de 2 : 2, 4, 8, etc…

Il est intéressant de former des groupes de 1 aussi important que possible.

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	1	0
11	1	1	0	1
10	1	0	0	0

D

Troisième règle : un même 1 peut être introduit dans plusieurs groupes. Les regroupements doivent être indépendants.

6.4. Règles de simplification

- Constituer des groupes de « 1 » ces groupes de taille maximale, doivent contenir un nombre de cases égal à un poids binaire (1,2, 4, 8, 16,...) et doivent respecter les symétries de la table ; les bords de la table sont adjacents, ce qui permet d'élargir les possibilités de regroupement grâce à des repliements.
- Un regroupement constitue un ensemble de mintermes (cases) liés par l'opération OU. Du fait de la symétrie, des variables se simplifient deux à deux. Un regroupement constitue donc une expression logique simplifiée. Pour extraire cette expression, on ne retient que les variables dont l'état logique n'est pas modifié par déplacement de case en case à l'intérieur du groupement.

- Couvrir tous les « 1 » même avec chevauchements. Une ou plusieurs cases peuvent être communes à plusieurs regroupements. On dit que l'on effectue des **chevauchements** pour augmenter la taille des groupes. **Cependant les inclusions ne sont pas autorisées**. La confection des groupes cesse lorsque tous les 1 appartiennent à au moins l'un d'entre eux. Toutes les expressions trouvées sont sommées (OU logique) pour constituer l'équation de la sortie considérée.

Remarque :

- Suivant la forme du résultat souhaité (canonique 1 ou 2), on extrait les « 1 » (équation de la fonction) ou les « 0 » (équation du complément logique de la fonction).

- Noter qu'on peut aussi dresser la table de Karnaugh en ne considérant que les maxtermes et appliquer les mêmes règles de simplification pour obtenir une expression minimale d'une fonction logique.

A	B	C	Y
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	0

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	1			1
A	1			1

Sum-of-products K-map

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

Product-of-sums K-map

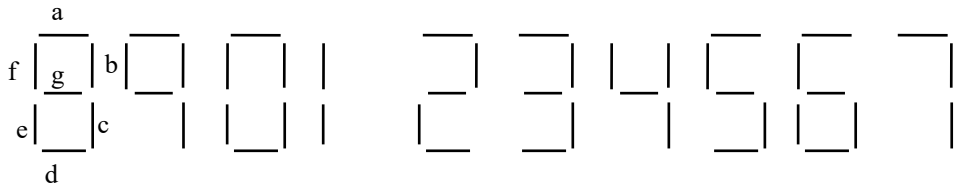
6.4 Cas d'une table incomplète.

Dans les problèmes techniques, il arrive que la fonction logique ne soit pas spécifiée pour certaines combinaisons des variables d'entrée. Cela survient lorsque les combinaisons ne se présentent jamais en fonctionnement normal.

Définition : On appelle fonction non complètement spécifiée, toute fonction booléenne dont l'évaluation n'est pas définie (ou n'a pas de sens) pour certaines configurations de ses entrées.

Exemple :

Décoder un nombre BCD pour commander un segment d'un afficheur à 7 segments notés a, b, c, d, e, f, g constituant les chiffres décimaux comme ci – dessous.



Considérons, par exemple le segment b. Dresser sa table de Karnaugh en considérant les variables DCBA. Remarquer que l'état de certaines cases n'est pas défini. On dit que les valeurs de ces cases sont indifférentes *(don't care conditions)* et on les désigne par X. Ces valeurs indifférentes sont astucieusement choisies pour faciliter la simplification car les combinaisons correspondantes ne se présentent jamais en fonctionnement normal.

DC \ BA	00	01	11	10
00	1	0	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

b

N	DCBA	b
0	0000	1
1	0001	0
2	0010	1
3	0011	1
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	1

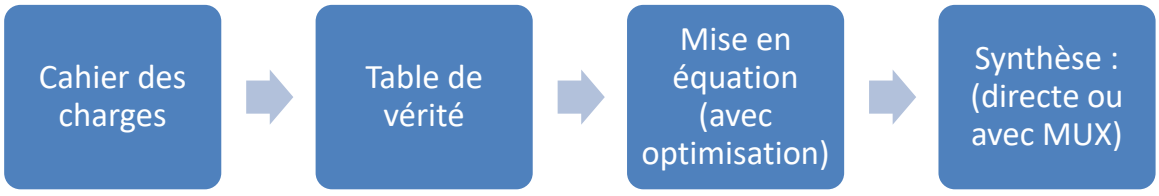
On trouve après simplification :

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

Synthèse des systèmes logiques combinatoires

A partir du cahier des charges, on construit la table de vérité. Une mise en équation directe permet d'exprimer la sortie à partir des différentes variables d'entrée, mais la recherche de l'expression logique peut être optimisée à l'aide des tables de Karnaugh.

Disposant de la fonction, la synthèse devient possible de manière directe à l'aide d'opérateurs logiques, de multiplexeur ou à l'aide de circuits spécialisés, les réseaux logiques programmables.



Descriptif de la synthèse d'une fonction logique combinatoire

6.5 Annexe de table de Karnaugh à 5 et 6 variables

Table de Karnaugh à 5 variables

ab \ cde	000	001	011	010	110	111	101	100
00								
01		1						
11	1	1	1				1	
10		1						

Table de Karnaugh à 6 variables

abc \ def	000	001	011	010	110	111	101	100
000		1						
001	1	1	1				1	
011		1						
010								
110								
111								
101		1						
100								

PROBLEMES DE LOGIQUE COMBINATOIRE

Définitions

Circuit combinatoire (combinational circuit)

Un circuit est combinatoire quand ses sorties ne dépendent que de ses entrées et non pas aussi de ses états antérieurs : à chaque combinaison des variables d'entrées correspond toujours une seule combinaison des fonctions de sortie et toujours la même. Il n'y a donc pas action des sorties sur les entrées du système.

Transcodeurs

Ce mot désigne l'ensemble des codeurs (encoders), des décodeurs (decoders) et des convertisseurs de codes (translators) . Ces circuits transforment une information présente à leurs entrées sous une forme donnée (code 1) en la même information présente en leur sortie sous une forme différente (code 2) ; il s'agit bien d'un transcodage ; on appelle :

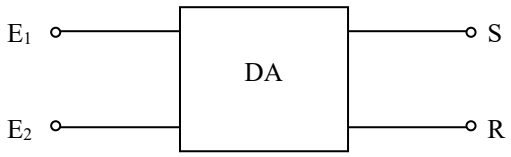
- Codeur : un circuit à 2^n entrées et n sorties ;
- Décodeur : un circuit à n entrées et 2^n sorties dont une seule est validée, chaque fois ;
- Convertisseur ou transcodeur tout circuit différent des précédents à p entrées et k sorties.

7.1 Introduction

Dans un problème de logique combinatoire les variables de sorties S_i dépendent uniquement des variables d'entrées E_i.

7.2 Demi – Additionneur (Half adder)

Un Demi – Additionneur permet de réaliser l'addition de deux bits représentés par les variables E_1 et E_2. Le résultat de l'addition est la variable S et éventuellement une retenue R.



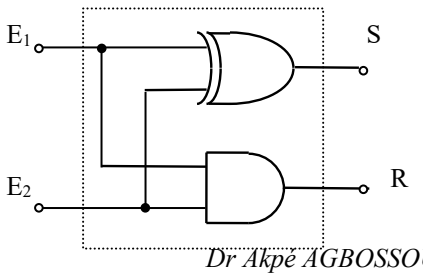
La table de vérité est donnée ci-dessous.

E_1	E_2	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Les équations booléennes qui correspondent à cette table sont :

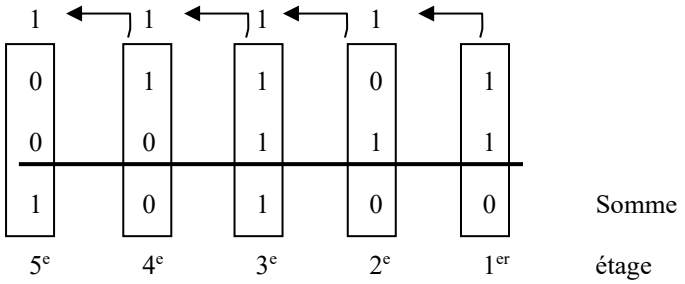
$S = \overline{E}_1 E_2 + E_1.\overline{E}_2 = E_1 \oplus E_2$

$R = E_1.E_2$



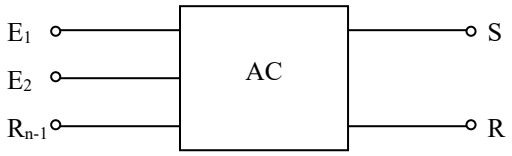
On en déduit le logigramme de la figure ci-contre.
Remarque : Pour additionner deux nombres à plusieurs bits, il faut tenir compte de l'éventuelle retenue de l'étage précédent. Voir exemple ci-dessous. Il faut donc un circuit à trois entrées : un additionneur complet.

Exemple :



7.3 Additionneur complet (Full adder)

Les variables de sorties S et R correspondent au résultat de l'addition de E_1, E_2 et R_{n-1} ; la sortie R étant toujours une éventuelle retenue R_n.



La table de vérité est donnée par :

entrées			Sorties	
E_1	E_2	R_{n-1}	S	R
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Les tables de Karnaugh sont :

E_1E_2 / R_{n-1}	00	01	11	10
0				
1				

Fonction S

R_{n-1} \ E_1E_2	00	01	11	10
0				
1				

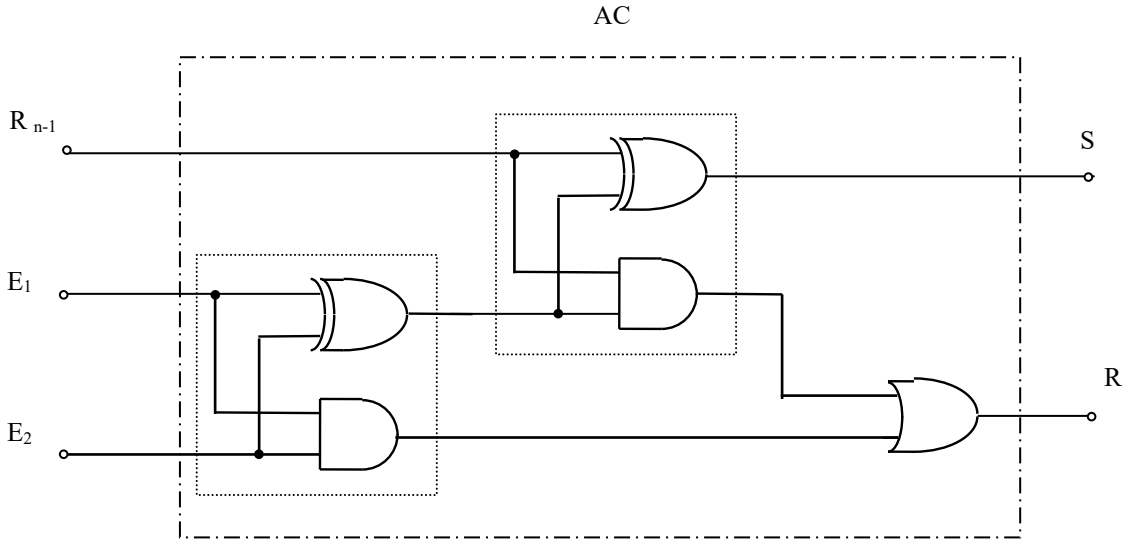
Fonction R

Ecrire l'expression de S et R et montrer que S et R donnent :

$S = E_1 \oplus E_2 \oplus R_{n-1}$

$R = E_1.E_2 + (E_1 \oplus E_2)R_{n-1}$

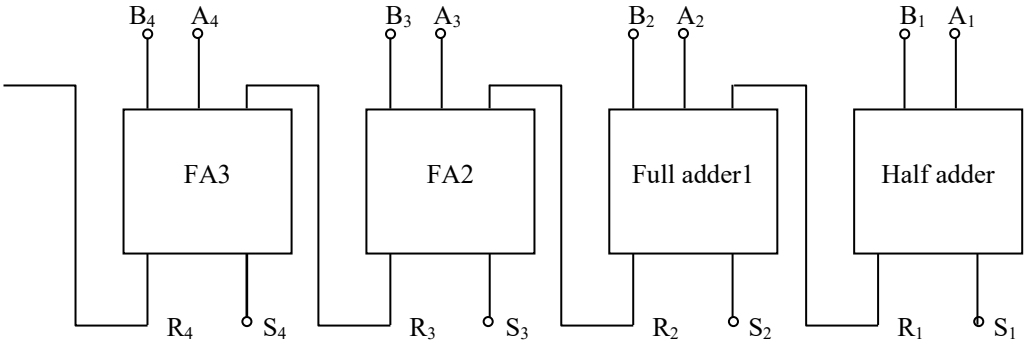
On obtient alors le logigramme suivant :



Considérons l'addition de deux nombres binaires de 4 bits :

$A_4A_3A_2A_1$
$+ \quad B_4B_3B_2B_1$
$=$

On aura le montage suivant avec quatre additionneurs complets:



7.4 Demi-Soustracteur (Half substractor)

Un demi – soustracteur effectue la différence $S = E_2 – E_1$ de deux bits avec éventuellement un report R. La table de vérité est donnée par :

E_1	E_2	S	R
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

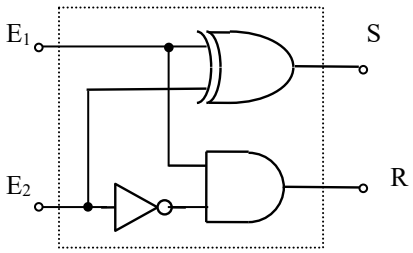
NB : Le report en anglais est désigné par Borrow (emprunt)

Les équations booléennes sont :

$S = \overline{E}_1 E_2 + E_1.\overline{E}_2 = E_1 \oplus E_2$

$R = E_1.\overline{E}_2$

D'où le logigramme :



7.5 Soustracteur Complet (Full substractor)

Les variables de sortie S et R_n correspondent au résultat de la différence des trois variables d'entrée : $E_2 - (E_1 + R_{n-1})$, la variable de sortie R_n étant toujours un éventuel report.

La table de vérité est :

entrées			Sorties	
E_1	E_2	R_{n-1}	S	R
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Les tables de Karnaugh sont :

R_{n-1} \ E_1E_2	00	01	11	10
0				
1				

Fonction S

R_{n-1} \ E_1E_2	00	01	11	10
0				
1				

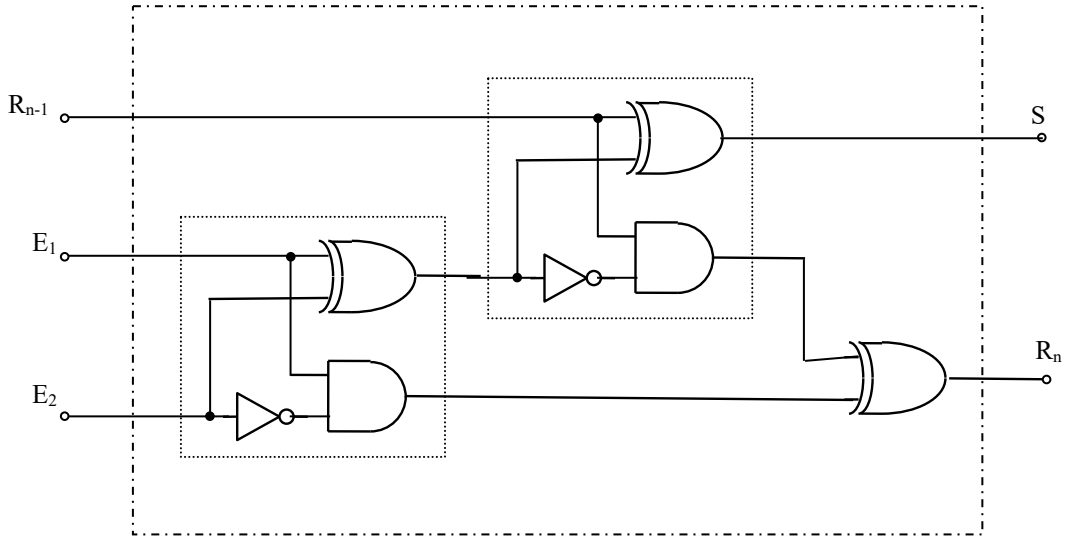
Fonction R_n

On obtient les équations booléennes :

$S = E_1 \oplus E_2 \oplus R_{n-1}$

$R_n = E_1.\overline{E}_2 + \overline{(E_1 \oplus E_2)}R_{n-1}$

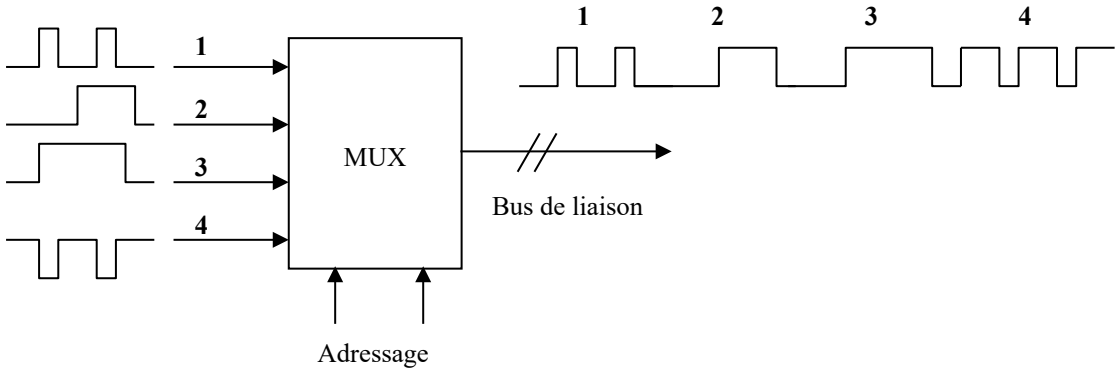
On en déduit le logigramme suivant :



7.8 Multiplexage (multiplexer)

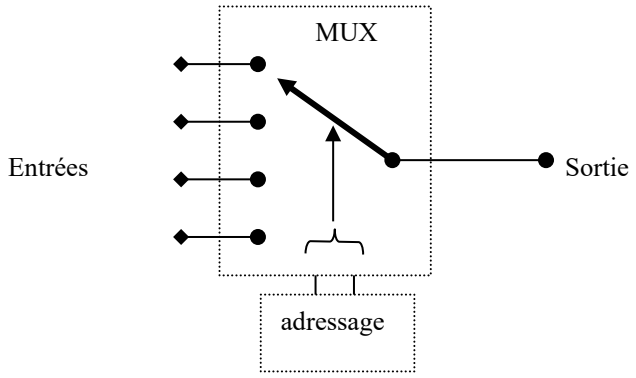
Un microprocesseur peut traiter des milliers d'informations à lui seul. La difficulté réside dans le fait qu'on ne peut établir un réseau de milliers de lignes autour d'un microprocesseur. Toutes ces informations doivent transiter par un conducteur. Le réseau téléphonique fonctionne sur le même système. Chaque paire de postes est connectée sur la ligne un certain temps à tour de rôle. C'est le principe du multiplexage.

Définition : un multiplexeur est un circuit combinatoire d'aiguillage qui regroupe en série sur une voie les signaux venant de n voies en parallèle.



La voie n°1 est d'abord sélectionnée par l'intermédiaire d'une adresse ; les informations de cette voie passe sur le bus de liaison. Ensuite c'est au tour de la voie n°2. Ainsi de suite. Nous pouvons donc transporter sur un conducteur un très grand nombre d'informations venues de points différents.

Le multiplexage est une sorte d'aiguillage selon le schéma ci-dessous :



Il est commandé par un système d'adressage. En fait la scrutation des voies d'entrée ne se fait pas n'importe comment. C'est le bloc d'adressage qui détermine quelle entrée sera en contact avec la sortie. On indiquera au bloc d'adressage, sous forme de combinaison binaire, l'entrée à connecter à la sortie. Il faut donc pouvoir réaliser un nombre de combinaisons correspondant au nombre d'entrées. On utilise généralement le code binaire pour l'adressage.

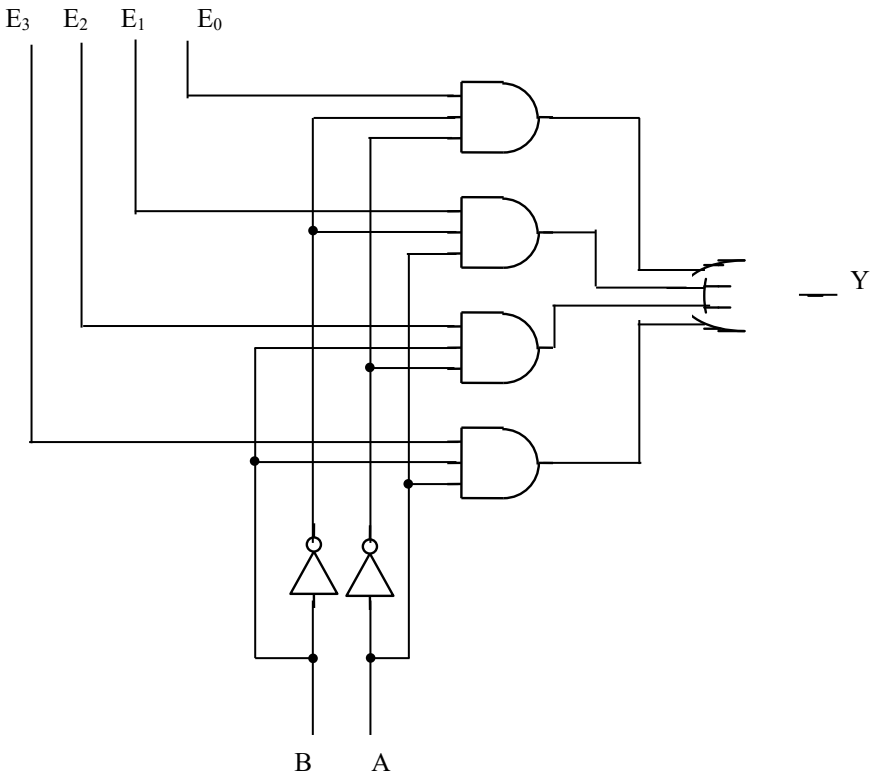
Pour un multiplexeur à 4 entrées, il faut 4 combinaisons possibles, c'est-à-dire 2^2 combinaisons (on exprime le nombre de combinaisons avec une puissance de 2 et l'exposant représente le nombre de chiffres binaires de la combinaison.

Exemple du multiplexeur de quatre lignes ($2^n = 4$).

L'adresse aura n = 2 lignes et on veut à la sortie l'état de la ligne E_0, E_1, E_2 ou E_3, le chiffre représentant l'adresse est affichée par les deux lignes B et A. Soit Y la sortie désirée. Sa table de vérité est donnée par :

B	A	Y
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

On en déduit l'équation $Y = \overline{A}\overline{B}E_0 + A\overline{B}E_1 + \overline{A}BE_2 + ABE_3$



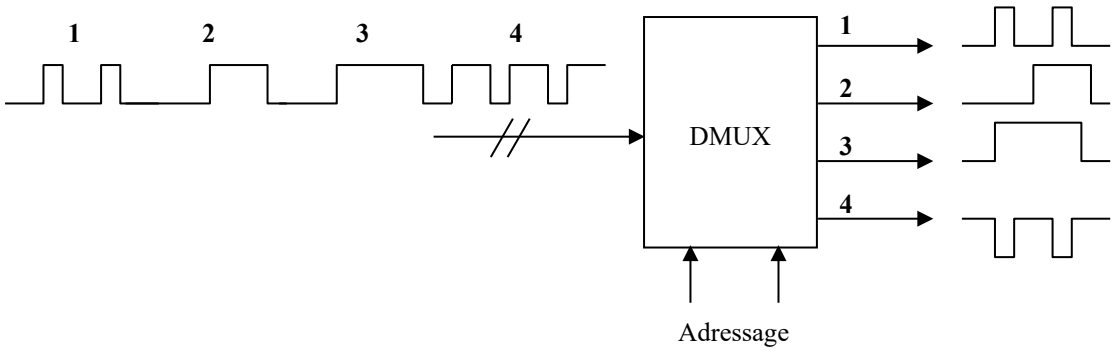
Le multiplexeur permet de transmettre sur la ligne Y les données apparaissant sur les lignes d'entrée E_i en faisant varier l'adresse.

Exemple de multiplexeurs disponibles en circuits intégrés : 74150 (16 entrées), 74151 (8 entrées).

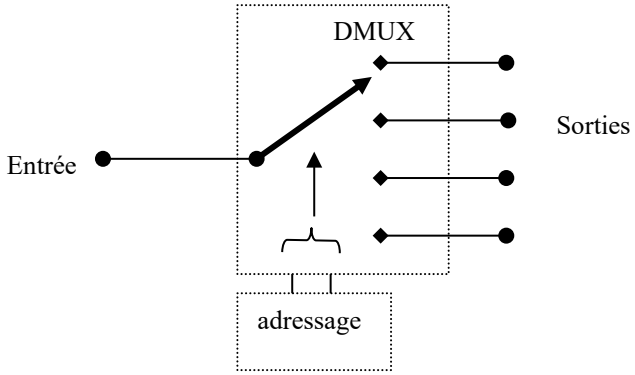
7.9 Démultiplexage (Demultiplexer)

Définition : un démultiplexeur est un circuit combinatoire d'aiguillage qui aiguille vers n voies en parallèle les signaux venant en série d'une voie.

Si nous avons une trame d'informations série sur le bus d'entrée, elle sera découpée et distribuée sur les sorties selon l'adressage.



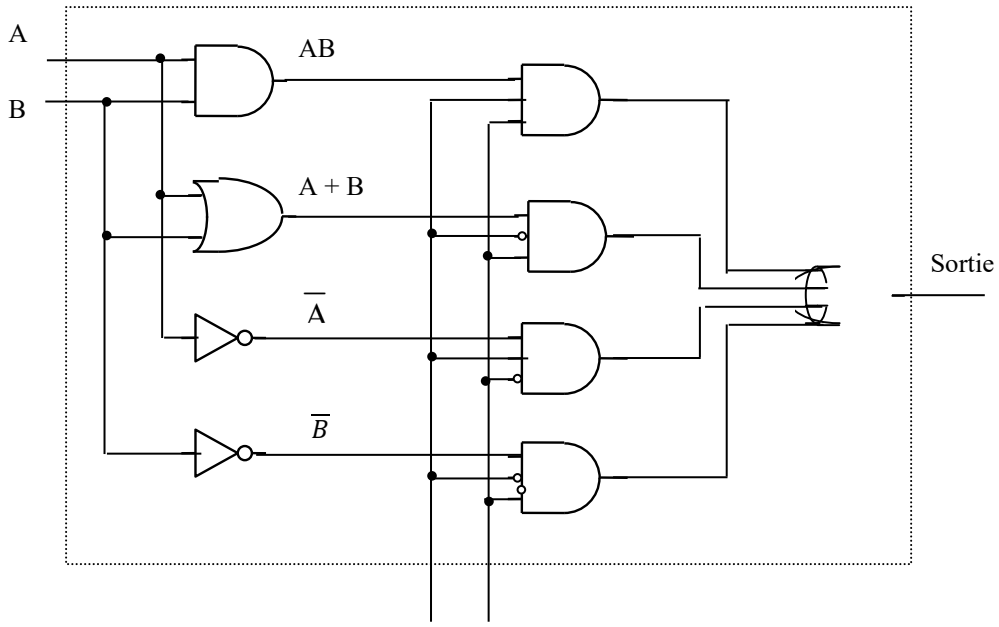
L'entrée est aiguillée sur une sortie selon l'adresse appliquée.



7.11 Unité Arithmétique et Logique (UAL)

L'UAL regroupe plusieurs fonctions arithmétiques : l'addition, la multiplication et plusieurs fonctions logiques : AND, OR, etc…

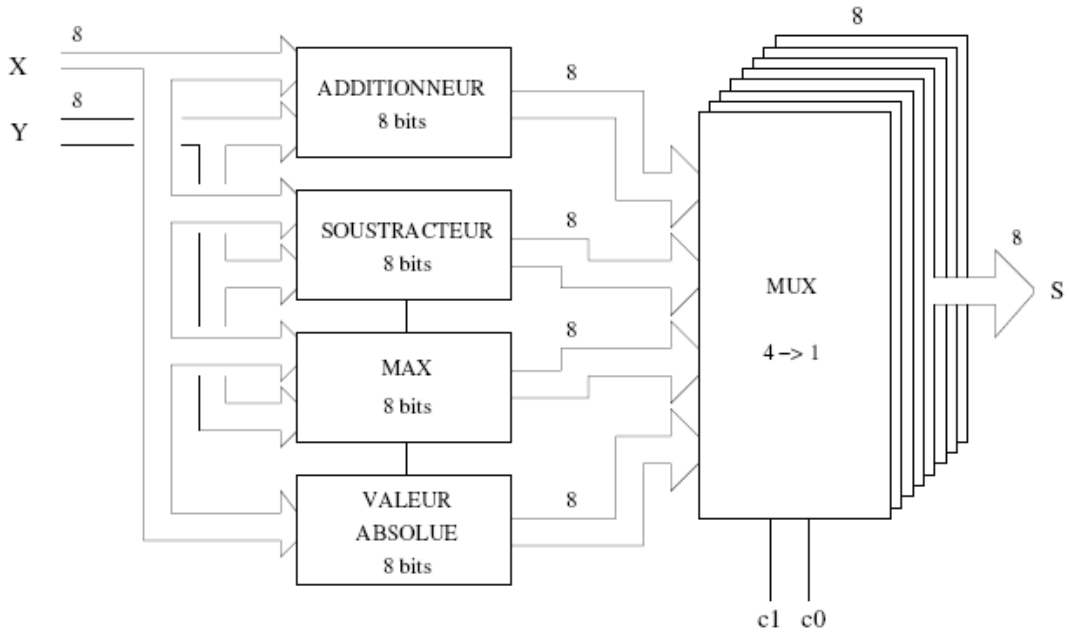
En étendant les principes de multiplexage et de démultiplexage, on peut concevoir un circuit qui permet de choisir non plus une entrée et une sortie mais une fonction logique parmi plusieurs.

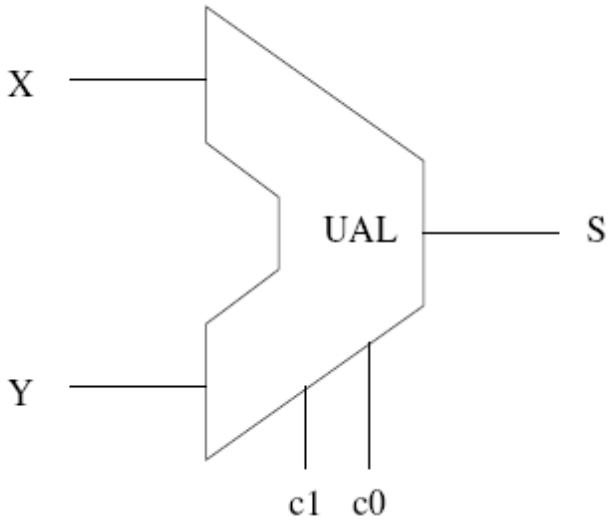


N° binaire de la fonction à exécuter

Le circuit ci-dessus permet de réaliser en sortie l'une des quatre opérations élémentaires sur deux bits :
AB ; A+B ; $\overline{A}$ et $\overline{B}$.

Les UALs (Unité arithmétique et logique) sont une des briques de base de tout microprocesseur actuel. Elles sont composées de 2^n circuits arithmétiques différents et de multiplexeurs à n bits de sélections qui permettent de transmettre en sortie, un seul des 2^n résultats calculés. Notons que le nombre de bits occupés par le résultat (par exemple 8 bits) conditionne le nombre de multiplexeurs nécessaires (ici 8) pour transmettre correctement ce résultat. Le schéma ci-dessous illustre un UAL à 8 bits quatre opérations.

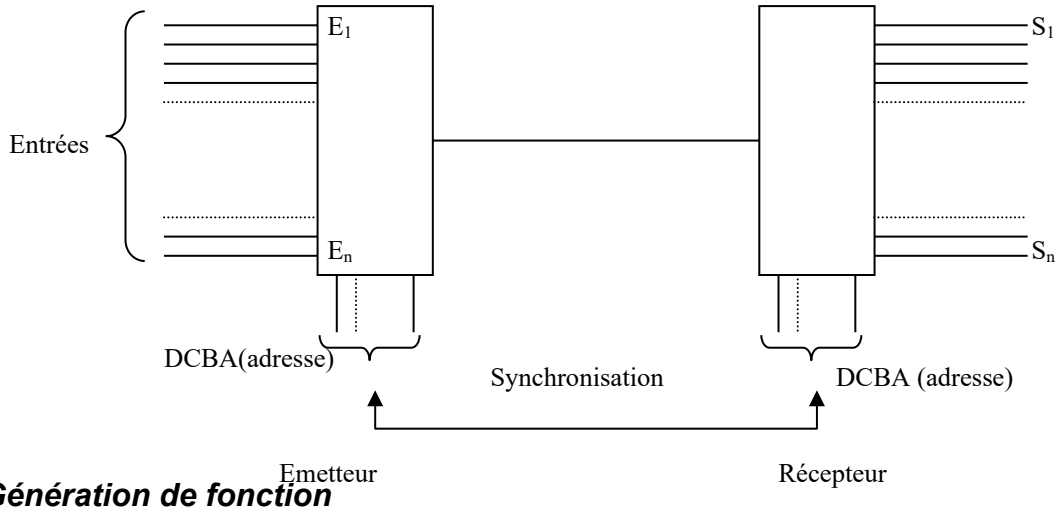




Symbole normalisé de l'UAL

7.12 Transmission de données en binaire

La figure ci-dessous illustre le principe d'association d'un multiplexeur et d'un démultiplexeur pour la transmission de données.
Avec un tel montage on peut lire la valeur de n lignes en un temps donné (temps de balayage de toutes les adresses) et la transmettre sur une autre ligne. En synchronisant la lecture du démultiplexeur, on peut reconstituer la valeur binaire de la ligne lue par l'adresse et la transmettre sur n lignes.



7.13 Génération de fonction

Le multiplexeur peut servir à générer des fonctions logiques de plusieurs variables. Nous présentons le principe à travers l'exemple suivant.
Réaliser la fonction S = $\overline{XY}Z + \overline{X}YZ + XY\overline{Z}$ à l'aide du multiplexeur à 8 entrées (SN74151).

1- Donner la table de vérité de la fonction S
2- Déterminer les données aux entrées pour avoir S à la sortie du multiplexeur pour les combinaisons d'adresse XYZ.

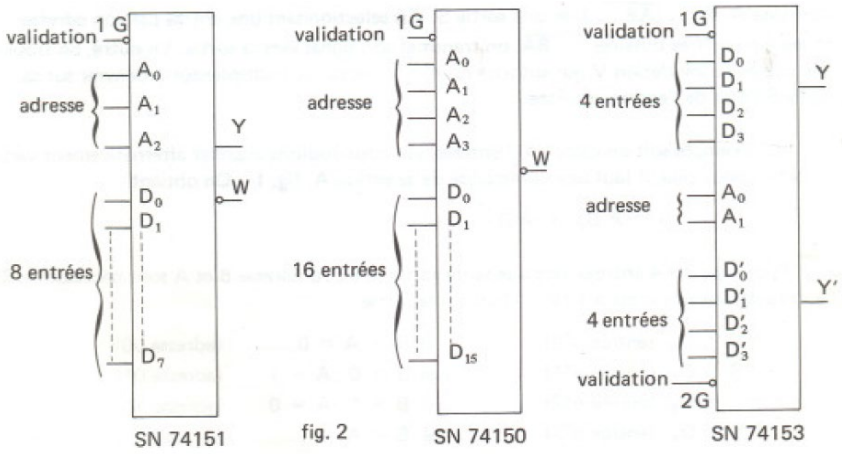
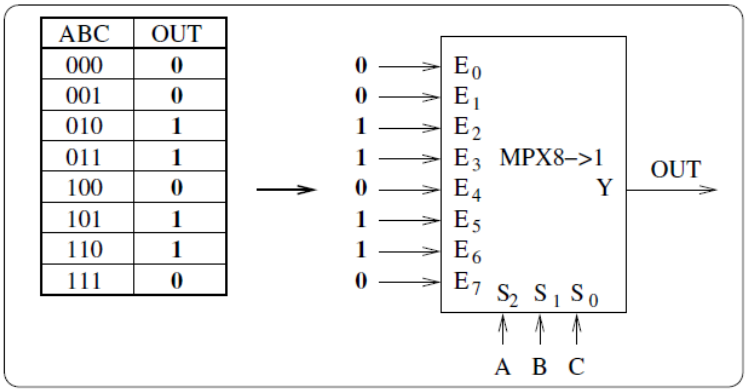


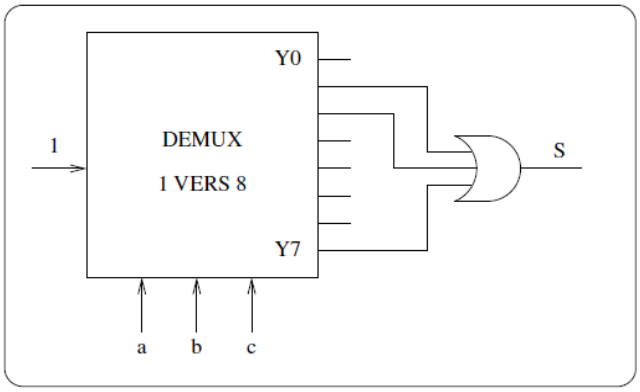
fig. 2

Exemple1 : Vérifier que le montage avec le multiplexeur ci-dessous réalise la fonction OUT donnée par sa table de vérité.

ABC	OUT
000	0
001	0
010	1
011	1
100	0
101	1
110	1
111	0



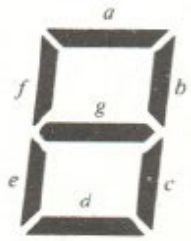
Exemple 2 : Vérifier que le montage avec le démultiplexeur ci-dessous génère la fonction
$S = \bar{a}\bar{b}c + \bar{a}b\bar{c} + abc$



7.14 Décodeur de DCB vers sept lignes

Le problème

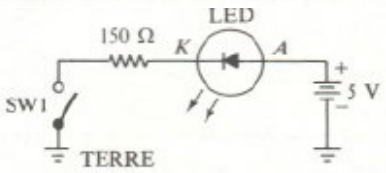
In a calculator or watch, we may represent information to be displayed in binary-coded decimal BCD form. Thus 0000 is for 0, 0001 is for 1, 0010 is for 2, 0011is for 3, and so on. Sixteen combinations are possible using four-bit words. However, only 10 combinations are used in BCD. Codes such as 1010 and 1011 dot not occur in BCD. The calculator display typically consists of light-emitting diodes (LEDs) or liquid crystals with seven segments, as illustrated below. The digit 0 through 9 are displayed by turning on appropriate segments as shown in the figure. Thus a decoder is needed to translate the four-bit binary-coded decimal words into seven-bit word of the form ABCDEFG for which A is high if segment A of the display is required to be on, B is high if segment B is required to be on, and so on. Thus 0000 is translated to 1111110 because all segments except G are on to display the symbol for zero. Similarly, 0001 becomes 0110000, and 0010 becomes 1101101. Thus the BCD-to-seven-segment decoder is a combinatorial circuit having four inputs and seven outputs.



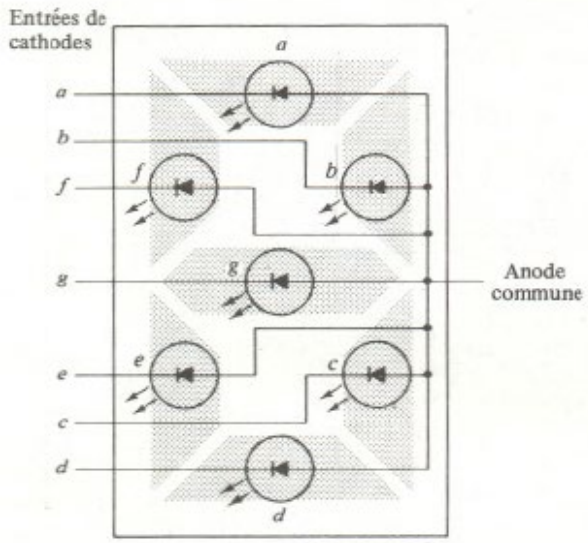
(*a*) Identification des segments

(*b*) Affichage typique des décimaux

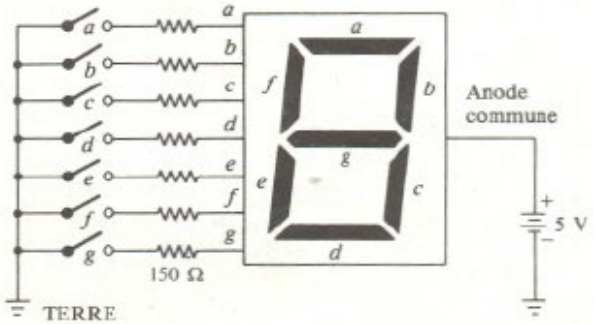
Fig. 6-10 L'afficheur à 7-traits



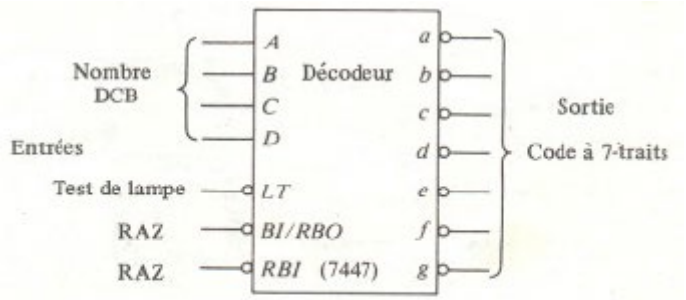
(*a*) Fonctionnement d'une diode photoémettrice (LED)



(*b*) Câblage des LED de l'afficheur à 7-traits



(*c*) Fonctionnement de l'afficheur à 7-traits



(*a*) Symbole logique

Décimal ou fonction	Entrées						BI/RBO	Sorties							Notes
	LT	*RBI*	*D*	*C*	*B*	*A*		*a*	*b*	*c*	*d*	*e*	*f*	*g*	
0	H	H	B	B	B	B	H	ON	ON	ON	ON	ON	ON	OFF	1
1	H	X	B	B	B	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	H	X	B	B	H	B	H	ON	ON	OFF	ON	ON	OFF	ON	
3	H	X	B	B	H	H	H	ON	ON	ON	ON	OFF	OFF	ON	
4	H	X	B	H	B	B	H	OFF	ON	ON	OFF	OFF	ON	ON	
5	H	X	B	H	B	H	H	ON	OFF	ON	ON	OFF	ON	ON	
6	H	X	B	H	H	B	H	OFF	OFF	ON	ON	ON	ON	ON	
7	H	X	B	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF	
8	H	X	H	B	B	B	H	ON	ON	ON	ON	ON	ON	ON	
9	H	X	H	B	B	H	H	ON	ON	ON	OFF	OFF	ON	ON	
10	H	X	H	B	H	B	H	OFF	OFF	OFF	ON	ON	OFF	ON	
11	H	X	H	B	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON	
12	H	X	H	H	B	B	H	OFF	ON	OFF	OFF	OFF	ON	ON	
13	H	X	H	H	B	H	H	ON	OFF	OFF	ON	OFF	ON	ON	
14	H	X	H	H	H	B	H	OFF	OFF	OFF	ON	ON	ON	ON	
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
BI	X	X	X	X	X	X	B	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	H	B	B	B	B	B	B	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	B	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON	4

ON : en marche, OFF : arrêté. H = niveau HAUT, B = niveau BAS, X = non pertinent

Notes : 1. L'entrée RAZ (*BI*) doit être ouverte ou en statu quo à un niveau logique HAUT quand on désire les fonctions de sortie de 0 à 15. L'entrée de RAZ d'impulsions (*RBI*) doit être ouverte ou HAUT si l'on ne désire pas le RAZ d'un zéro décimal.

2. Quand un niveau logique BAS est appliqué directement à l'entrée de RAZ (*BI*), toutes les sorties de segments sont OFF, quel que soit le niveau de toute autre entrée.

3. Quand la RAZ d'impulsions (*RBI*) et les entrées *A*, *B*, *C* et *D* sont à un niveau BAS, l'entrée de test de lampe (*LT*) étant HAUT, toutes les sorties de segments passent en OFF et la sortie de RAZ d'impulsions (*RBO*) passe à un niveau BAS (condition de réponse).

4. Quand l'entrée RAZ/sortie du RAZ d'impulsions (*BI/RBO*) est ouverte ou en statu quo HAUT et qu'un BAS est appliqué à l'entrée de test de lampe (*LT*), toutes les sorties de segments sont ON.

(*b*) Table de vérité (*Reproduit avec l'autorisation de Texas Instruments, Inc.*)

Fig. 6-12 Décodeur/commande CI 7447 DCB-7 traits du commerce

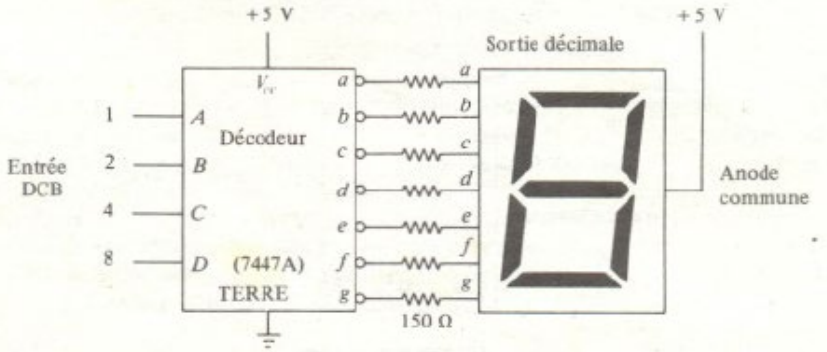
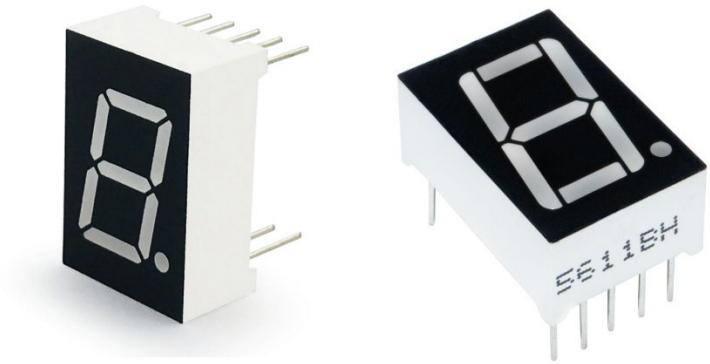


Fig. 6-13 Câblage d'un décodeur et d'un afficheur à 7-trait à LED

https://youtu.be/iRmZK1PAB_I

https://youtube.com/shorts/D_D7cX90AJE?feature=share

https://youtu.be/9UV2laBRPfI



La solution

Etape 1 : la table de fonctions

Dresser la table des sorties en fonction des entrées.

N	DCBA	a	b	c	d	e	f	g
0	0000	1	1	1	1	1	1	0
1	0001							
2								
3								
4								
5								
6								
7								
8								
9								

Etape 2 : Mise en équation

Il faut établir la table de Karnaugh de chaque segment. On établira donc 7 tables de Karnaugh. Les 0 étant moins nombreux dans la table des fonctions on choisit de définir les équations de commande d'extinction des segments.

Segment **a**

BA / DC	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	X	X	X	X
10	1	1	X	X

En considérant les X pertinents on trouve $\overline{a} = A\overline{B}\overline{C}\overline{D} + \overline{A}C$

Segment **b**

DC \ BA	00	01	11	10
00				
01				
11				
10				

Résultat : $\overline{b} = A\overline{B}C + \overline{A}\mathrm{BC}$

Segment **c**

	00	01	11	10
00				
01				
11				
10				

Résultat $\overline{c} = \overline{A}B\overline{C}$

Segment **d**

	00	01	11	10
00				
01				
11				
10				

Résultat $\overline{d} = \overline{A\overline{B}C} + \mathrm{ABC} + A\overline{B\overline{C}}$

Segment **e**

	00	01	11	10
00				
01				
11				
10				

Résultat $\overline{e} = A + \overline{B}C$

Segment **f**

	00	01	11	10
00				
01				
11				
10				

Résultat $\overline{f} = A\overline{CD} + \mathrm{AB} + B\overline{C}$

Segment **g**

	00	01	11	10
00				
01				
11				
10				

Résultat $\overline{g} = \overline{B\overline{C}D} + \mathrm{ABC}$

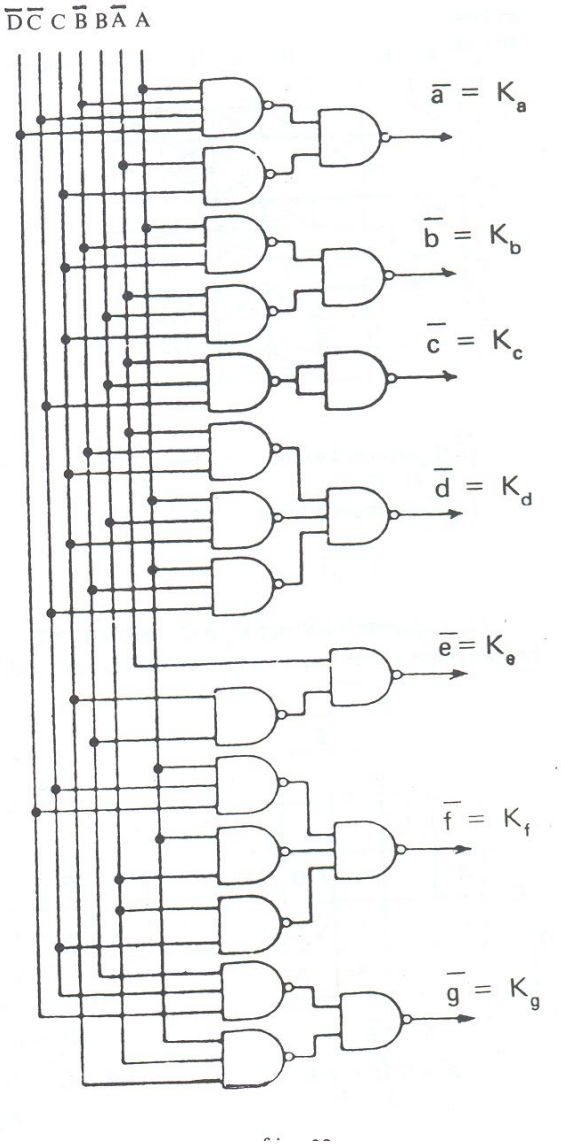
Etape 3 Implantation

Les fonctions logiques ci-dessus définies sont implantées à l'aide de portes Nand. (Figure ci-dessous).

Remarque sur l'utilisation des tables de Karnaugh

En conclusion la simplification par table de Karnaugh donne une vue graphique des règles algébriques à appliquer pour conduire au mieux les calculs et une représentation synthétique de la fonction. La limite commence à apparaître pour n = 5 et Karnaugh devient totalement inutilisable au-dessus de n = 6.

Pour un nombre de variables supérieur à 6, il existe d'autres représentations utilisées par les ordinateurs comme les BDD (Binary Decision Diagram) que vous ne verrez, peut-être que dans le cadre d'un cours avancés.



7.15 Conversion de codes

Dans les machines les unités numériques ne traitent que des bits 0 et 1. Pour les humains il est difficile de comprendre les longues chaines de 0 et de 1. Il devient alors nécessaire de traduire en langage usuel le langage des machines.

Exemple du fonctionnement d'un calculateur.

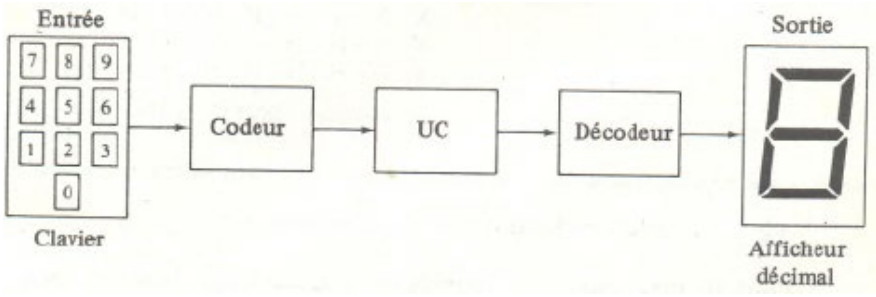
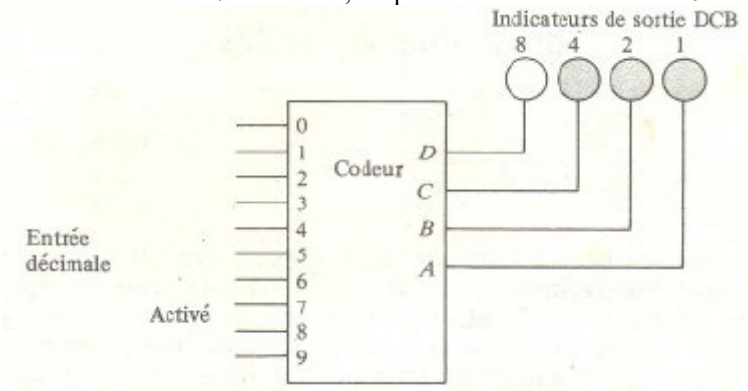
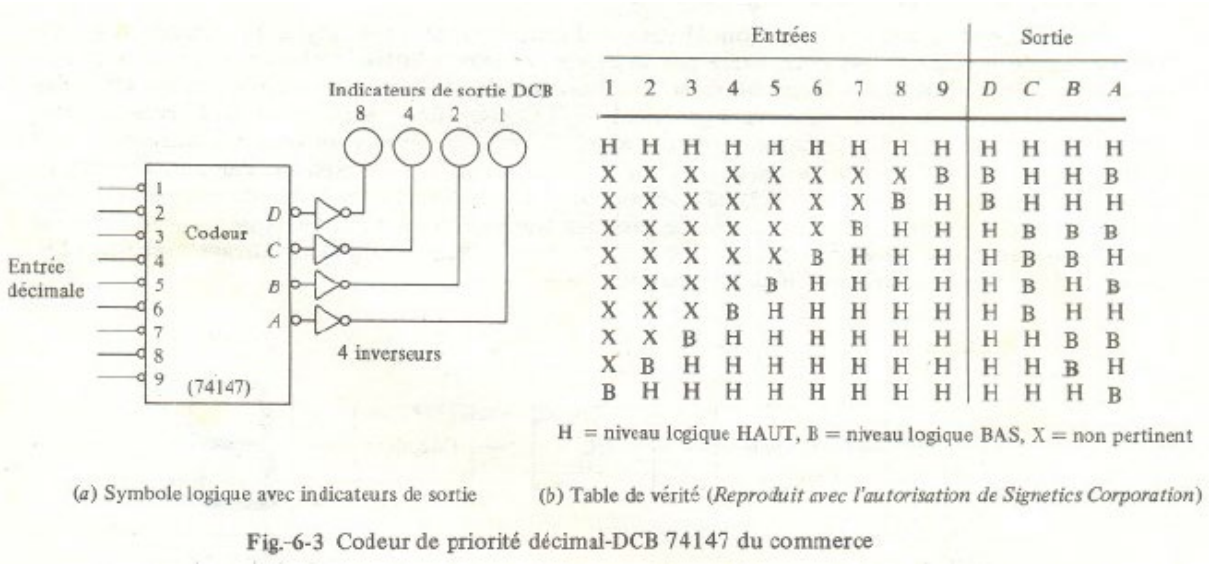


Schéma fonctionnel d'un calculateur

Le codage : la fonction du codeur (dans la calculette) est de traduire l'entrée décimale en un nombre DCB. Le codeur possède 10 entrées (via les touches) et 4 sorties à droite. Il doit avoir une entrée active qui produit à son tour une sortie unique.
Sur la figure ci-dessous l'entrée décimale 7 est activée, ce qui entraîne une sortie DCB 0111 (dans l'ordre DCBA).



La figure ci-dessous donne le diagramme fonctionnel d'un codeur décimal-DCB qu'on trouve dans le commerce Réf :CI TTL 74147.



Entrées									Sortie			
1	2	3	4	5	6	7	8	9	*D*	*C*	*B*	*A*
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	B	B	H	H	B
X	X	X	X	X	X	X	B	H	B	H	H	H
X	X	X	X	X	X	B	H	H	H	B	B	B
X	X	X	X	X	B	H	H	H	H	B	B	H
X	X	X	X	B	H	H	H	H	H	B	H	B
X	X	X	B	H	H	H	H	H	H	B	H	H
X	X	B	H	H	H	H	H	H	H	H	B	B
X	B	H	H	H	H	H	H	H	H	H	B	H
B	H	H	H	H	H	H	H	H	H	H	H	B

H = niveau logique HAUT, B = niveau logique BAS, X = non pertinent

(*a*) Symbole logique avec indicateurs de sortie (*b*) Table de vérité (*Reproduit avec l'autorisation de Signetics Corporation*)

Fig.-6-3 Codeur de priorité décimal-DCB 74147 du commerce

Symbole logique avec indicateurs de sortie et table de vérité

Remarque : les entrées et les sorties sont actives aux niveaux bas. Voir table de vérité.

Le schéma logique du CI TTL 74147 ci-dessous donné par le constructeur Texas Instruments Inc. comporte 30 portes logiques. Ce codeur offre une caractéristique de priorité qui se traduit par l'activation du chiffre le plus élevé qui possède une entrée Bas. Vérifier le cas 9 et 5 au niveau BAS. Indiquer la sortie.

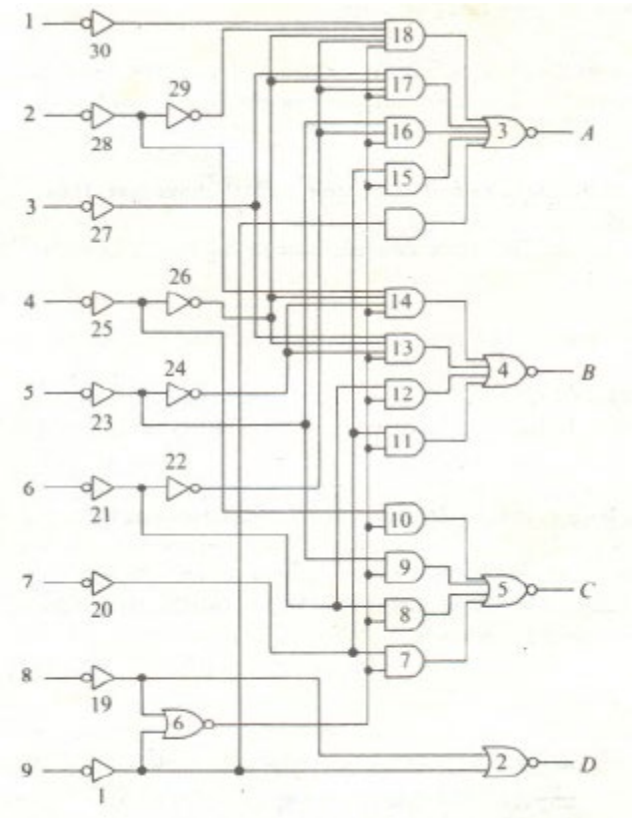


Diagramme logique du codeur TTL 74147.

Le décodage

Un décodeur peut être considéré comme l'opposé d'un codeur. Dans notre exemple ici le décodeur convertirait un code DCB en décimal. Une seule ligne de sortie est active à chaque instant. Si aucune entrée n'était activée, l'indicateur de sortie 0 serait allumé.

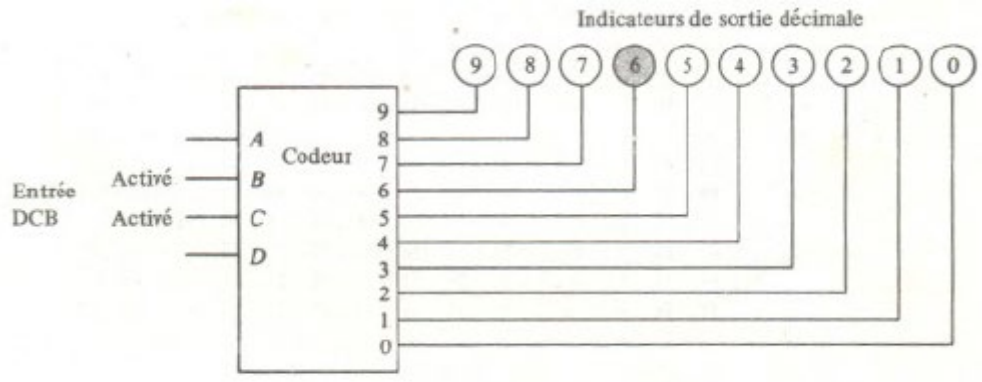
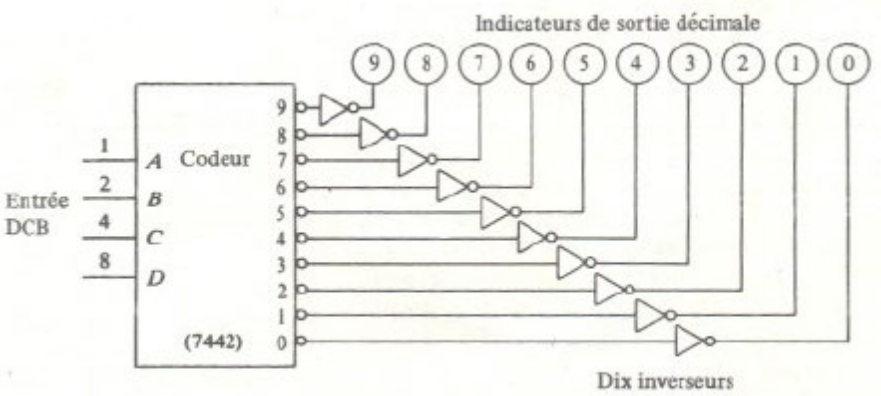


Fig. 6-6 Codeur DCB-décimal (schéma logique)

Le schéma ci-dessous représente un décodeur DCB-décimal du commerce (7442 TTL). Les quatre entrées à gauche sont référencées DCBA et correspondent respectivement aux entrées des 8, 4, 2 et 1.

Remarque les sorties sont actives à des niveaux bas. Les dix sorties flottent normalement à niveau Haut. Pour des raisons de commodité, 10 inverseurs ont été introduits dans le circuit pour commander des indicateurs décimaux lumineux.



(*a*) Schéma logique avec indicateurs de sortie

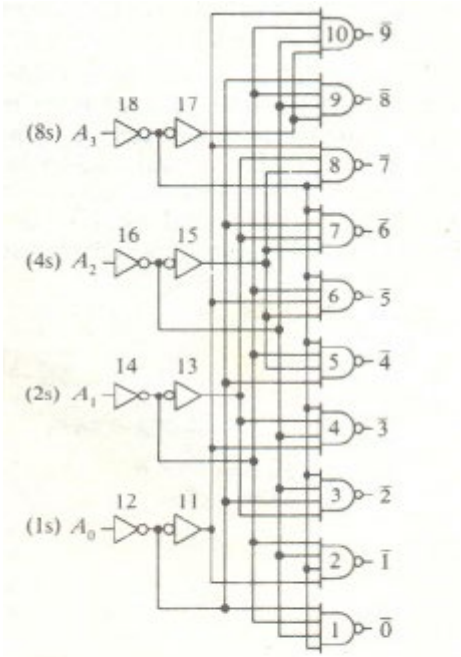
La table de vérité et le schéma logique du codeur ci-dessous illustrent le fonctionnement du CI 7442.

Ligne	N°	Entrées DCB				Sortie décimale									
		D	*C*	*B*	*A*	0	1	2	3	4	5	6	7	8	9
1	0	B	B	B	B	B	H	H	H	H	H	H	H	H	H
2	1	B	B	B	H	H	B	H	H	H	H	H	H	H	H
3	2	B	B	H	B	H	H	B	H	H	H	H	H	H	H
4	3	B	B	H	H	H	H	H	B	H	H	H	H	H	H
5	4	B	H	B	B	H	H	H	H	B	H	H	H	H	H
6	5	B	H	B	H	H	H	H	H	H	B	H	H	H	H
7	6	B	H	H	B	H	H	H	H	H	H	B	H	H	H
8	7	B	H	H	H	H	H	H	H	H	H	H	B	H	H
9	8	H	B	B	B	H	H	H	H	H	H	H	H	B	H
10	9	H	B	B	H	H	H	H	H	H	H	H	H	H	B
11	Invalide	H	B	H	B	H	H	H	H	H	H	H	H	H	H
12		H	B	H	H	H	H	H	H	H	H	H	H	H	H
13		H	H	B	B	H	H	H	H	H	H	H	H	H	H
14		H	H	B	H	H	H	H	H	H	H	H	H	H	H
15		H	H	H	B	H	H	H	H	H	H	H	H	H	H
16		H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = HAUT B = BAS

(*b*) Table de vérité (*Reproduit avec l'autorisation de Signetics Corporation*)

Fig. 6-7 Codeur/commande 7442 du commerce (DCB-décimal)



ig. 6-8 Schéma logique du codeur 7442
(Reproduit avec l'autorisation de
Signetics Corporation)

LES BASCULES et leurs APPLICATIONS

8.1. Les Bascules (Flip-flops)

8.1.1 Rappels sur les circuits logiques combinatoires et introduction aux circuits séquentielles

On distingue deux types de circuits logiques :

- les circuits combinatoires (combinatorial circuits)
- les circuits séquentiels (sequential logic circuits)

Les circuits combinatoires sont des circuits tels que les grandeurs de sorties ne dépendent strictement que des grandeurs d'entrées. Ils peuvent s'étudier facilement à l'aide de l'algèbre de BOOLE. Cependant il ne faut pas oublier que toute matérialisation d'un opérateur booléen introduit obligatoirement un élément retard qui n'est pas pris en compte par l'algèbre de Boole.

Les éléments retards peuvent donc introduire des perturbations dans le fonctionnement des circuits réels qui présentent alors un comportement non prévu par une étude menée à l'aide de l'algèbre de Boole.

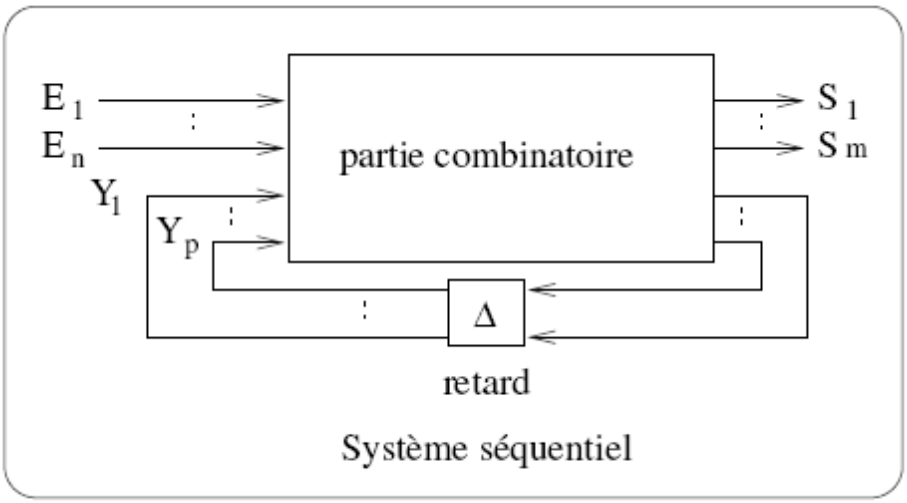
Les principaux circuits logiques combinatoires sont :

- les circuits arithmétiques
- les codeurs, transcodeurs et décodeurs
- les multiplexeurs et démultiplexeurs

Nous avons étudié jusqu'ici des réseaux que nous avons appelés combinatoires, c'est-à-dire qui associent à toute combinaison binaire d'entrée une combinaison de sortie qui est unique. De ce type de fonctionnement il résulte en particulier que chaque fois qu'on présentera une combinaison C donnée à l'entrée, à des instants distincts, on verra apparaître la même combinaison à la sortie. La correspondance entrée-sortie établie par un tel réseau est donc ponctuelle, elle est indifférente à ce qui s'est passé entre les différents instants où on a appliqué la combinaison C : elle ne fait intervenir aucun phénomène de mémoire de ce qui s'est passé avant. La différence essentielle entre un système combinatoire et un système séquentiel est que l'état du second ne dépend pas uniquement de la combinaison d'entrée à un moment donné mais de la séquence des combinaisons précédentes et de son état initial.

Un circuit séquentiel possède donc un état interne qui caractérise le chemin chronologique qu'il a parcouru depuis sa création. En conséquence, à une combinaison des variables d'entrées (e_1, …e_n) correspondent plusieurs états possibles du système. Pour différencier ces états, il faut introduire un certain nombre de variables supplémentaires, appelées variables internes (y_1,… y_p), l'état du système étant fixé par la combinaison (e_1,…, e_n, y_1,…, y_p). Un système séquentiel peut alors être considéré comme constitué d'un circuit combinatoire ayant pour entrée e_1…..e_n, y_1….y_p, ils apparaissent ainsi comme des systèmes bouclés.

Au niveau de la structure des circuits logiques, ils se caractérisent par au minimum une boucle de rétroaction (la sortie réagit sur l'entrée). L'étude des circuits séquentiels fait apparaître la notion de stabilité (comme dans tout système bouclé).



Suivant les conditions de changement des variables de sortie, on distingue deux types de système séquentiel :

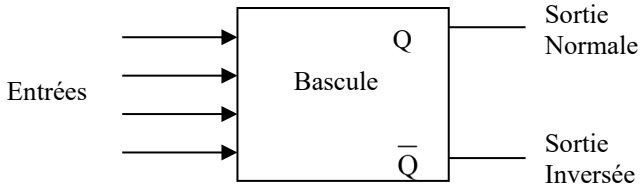
- les systèmes séquentiels asynchrones pour lesquels l'état des sorties évolue spontanément à la suite d'un changement de configuration des variables d'entrée.
- les systèmes séquentiels synchrones pour lesquels l'évolution des sorties est assujettie à une commande spécifique généralement appelée horloge. Le système est alors synchronisé sur le signal d'horloge.

Partout où la sûreté de fonctionnement est un élément déterminant d'une conception, il n'est utilisé que des circuits séquentiels synchrones. La conception moderne des systèmes séquentiels ne fait donc plus appels qu'à des systèmes synchrones dans lesquels sont parfaitement distinguées les parties combinatoires, des parties de mémorisation. La fonction mémorisation est généralement réalisée par des dispositifs physiques capables de mémoriser un bit : les bascules.

8.1.2 Principes de fonctionnement des bascules

Une bascule est un circuit logique qui a deux sorties : Q appelée sort sortie normale et $\overline{Q}$appelée sortie inversée. Quand on dit qu'une bascule est au niveau haut (1) ou au niveau bas (0) on fait référence à la sortie Q. Une bascule possède deux états de fonctionnement stables :

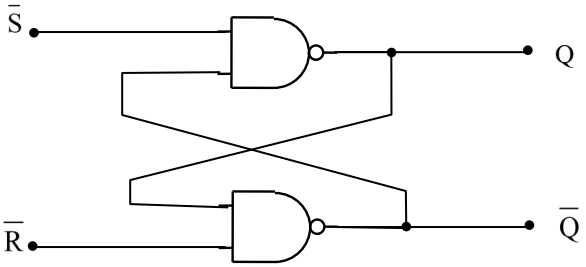
- Etat Q = 1, $\overline{Q}$ = 0
- Etat Q = 0, $\overline{Q}$ = 1



La bascule comprend une ou plusieurs entrées qui déterminent le passage de la bascule d'un état à l'autre (basculement). Quand une impulsion est appliquée à une entrée pour imposer un certain état à la bascule, celle-ci demeure dans cet état même après le retrait de l'impulsion. C'est ce qu'on appelle la mémoire de la bascule.

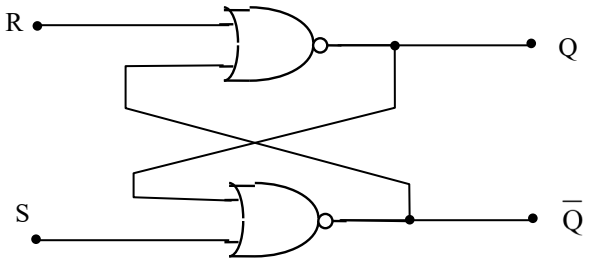
8.1.3 La bascule RS en porte Nand,

On désigne par Q^+ l'état futur de la bascule après tout changement d'état sur les entrées.



On écrit l'équation logique de la sortie $Q^+ = S + Q\overline{R}$

8.1.4 Bascule RS en porte NOR



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

Dans les deux cas de figure on peut écrire $Q^+ = (S + Q)\ \overline{R} = S + \overline{R}Q$. La bascule comporte 2 entrées :

S : mis pour **Set** (mise a un) : si S= 1 et R = 0 alors $Q^+ = 1$
R : mis pour **Reset** (mise à zéro) : si R = 1 et S = 0 alors $Q^+ = 0$

Si R = S = 0 alors $Q^+ = Q$, soit maintien de l'état, ce qui est bien la fonction mémorisation.

On peut remarquer que dans ces trois cas, les sorties des deux portes sont dans un état complémentaire. Ceci n'est plus le cas pour S = R = 1. (Vérifier que l'on a Q = $\overline{Q}$= 0 ou 1).

Devant l'intérêt que l'on a d'avoir les deux sorties complémentées, on décrète alors l'état S = R = 1 état interdit.

Par analogie avec les circuits combinatoires, on dresse une table de fonctionnement sous une forme identique à celle d'une table de vérité.

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	interdit

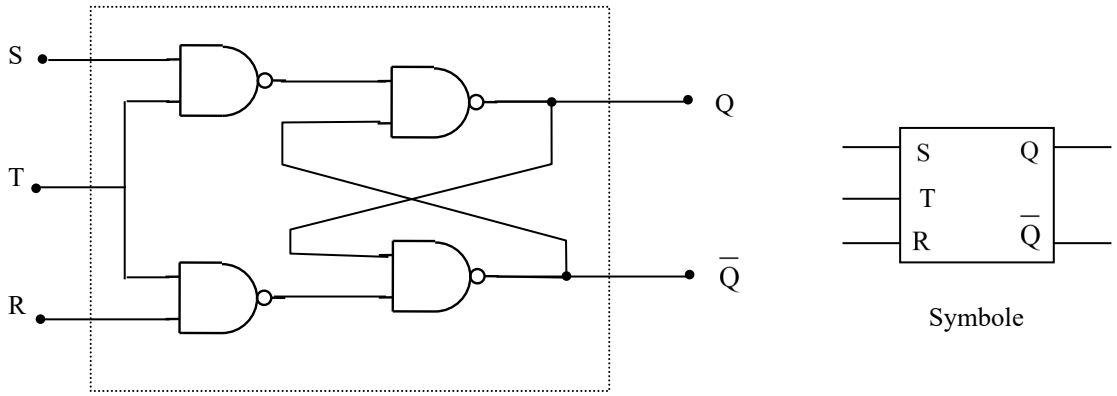
Table de fonctionnement d'une bascule RS

Symboles de bascules RS.

Il s'agit dans ce qui précède d'une bascule asynchrone. Sa variante synchrone est la bascule RST ci-dessous.

8.1.5 La bascule RST

Pour la rendre synchrone, en considérant que si les entrées S et R sont à l'état 0, il y a maintien de l'information, il suffit de se rendre maître du transfert de l'état des entrées à l'aide de deux portes And, selon le schéma suivant :



Symbole

Bascule RST (clocked RS flip-flop)

L'entrée T joue le rôle d'entrée horloge, le circuit ne réagissant aux entrées que pour T = 1.

8.1.6 La bascule D ou Latch (verrouillage)

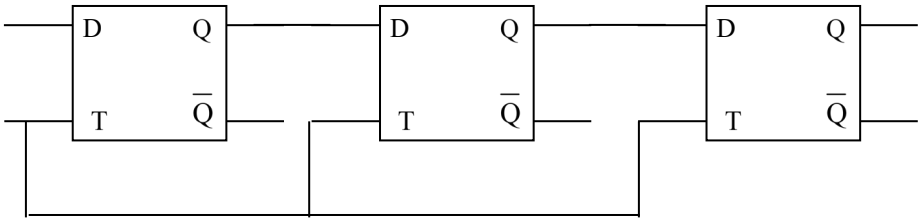
La bascule RST n'est pas souvent utilisée. Comme pour la bascule RS la présence d'un état interdit en limite beaucoup les applications. La bascule D, qui permet de verrouiller (latch) l'état d'une information est par contre d'un emploi très fréquent. Il s'agit d'une bascule RST telle que les entrées R et S soient inversées. **On évite ainsi l'état interdit.**

Bascule D

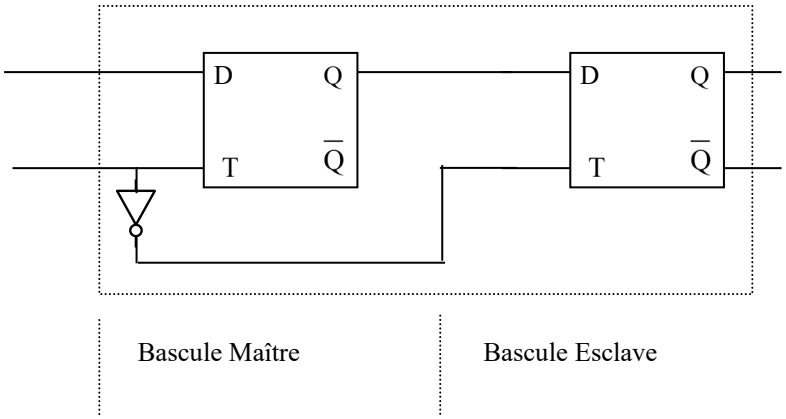
Symbole Bascule D
(*clocked D flip-flop*)

Tant que l'entrée d'horloge (Clk ou T) est à l'état 1, la sortie Q suit l'état de l'entrée D. Lorsque l'entrée Clk ou T passe à l'état 0, la sortie Q maintien l'état présent en D avant la transition de l'horloge.

Dans la pratique il est souvent nécessaire de placer plusieurs bascules D (Maître Esclave) en cascades, par exemple pour former un registre à décalages.



Dans un tel montage, toutes les bascules prennent le même état, en fonction de l'entrée E, lorsque Clk ou T est à l'état 1. On dit que la bascule D est transparente. Pour éliminer ce phénomène il est nécessaire de découpler temporellement la sortie Q de l'entrée D. Ceci est réalisé en utilisant une structure Maître-Esclave dont le schéma de principe est le suivant :



Principe de la bascule Maître-Esclave.

L'entrée de l'information dans le maître se fait lorsque l'horloge est au niveau 1, la bascule esclave étant figée. Quand l'horloge repasse au niveau 0, le maître est verrouillé et l'information est transférée dans l'esclave. A cause du retard inhérent à la porte inverseuse, il n'y a pas de problème de recouvrement.

Remarque : Les bascules D, RS et JK existent en version Maître Esclave (Master/Slave). On retrouve ce type de bascule en particulier dans les registres à décalage.

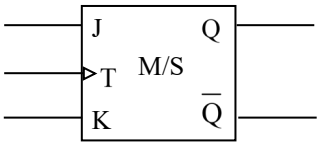
8.1.7 Les bascules JK

La conception actuelle des circuits logiques se fait à l'aide de blocs de mémorisation et de blocs combinatoires. Pour réaliser les ensembles de mémorisation, il est important de disposer de bascules qui soient facilement prépositionables à un état donné.

Par analogie avec le tableau de fonctionnement, on peut imaginer comme solution la bascule suivante.

J=S	K=R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Pour la condition R = S =1 il y a toujours basculement.



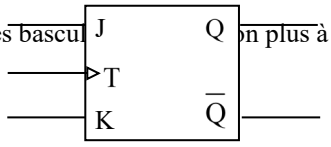
Bascule JK maître-escalve

Pour la distinguer de la bascule RS, on adopte les conventions suivantes :

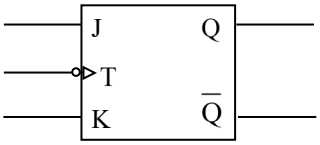
J = S
K = R

Et la bascule est nommée JK.

8.1.8 Les bascules déclenchées par front (edge triggered Flip flop)

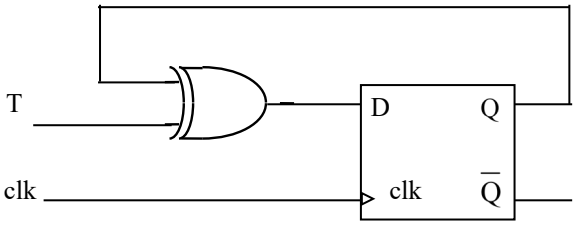
Ce sont des bascul 

n plus à un état de l'horloge mais à un front. Elles sont dites « edge triggered »



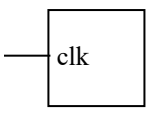
8.1.8.1 Bascules T (trigger)

C'est une bascule à entrée T qui change d'état à chaque front actif de l'horloge si T est à l'état 1 ou qui garde son état si T = 0. Son schéma de principe est le suivant :

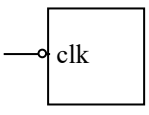


La bascule T trouve son emploi privilégié dans les compteurs.

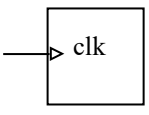
Récapitulation



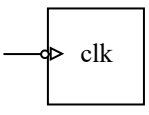
Bascule sensible à l'état 1



Bascule sensible à l'état 0



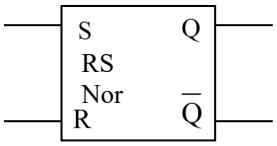
Bascule sensible au front montant (*Positive-edged-triggered flip flop*)



Bascule sensible au front descendant (*Negative-edged-triggered flip flop*)

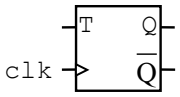
Clocked flip flop

Bascule RS



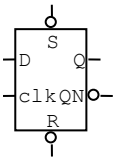
S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	interdit

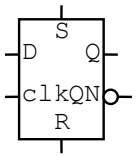
Bascule T

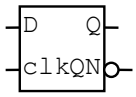


$$Q^+ = T\overline{Q} + \overline{T}Q$$

Bascule D

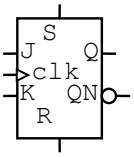






$$Q^+ = D$$

Bascule JK



J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Remarques :

1. En général les bascules synchrones possèdent aussi des entrées asynchrones souvent prioritaires Reset (R), Set (S). Ces entrées sont actives soit au niveau haut soit au niveau bas.

2. Selon les fabricants on adopte les notations suivantes :

Remise à 1	Remise à 0
DC set	DC clear
Preset	Clear
Set	Reset
RAU	RAZ
S_D(direct set)	C_D(direct clear)

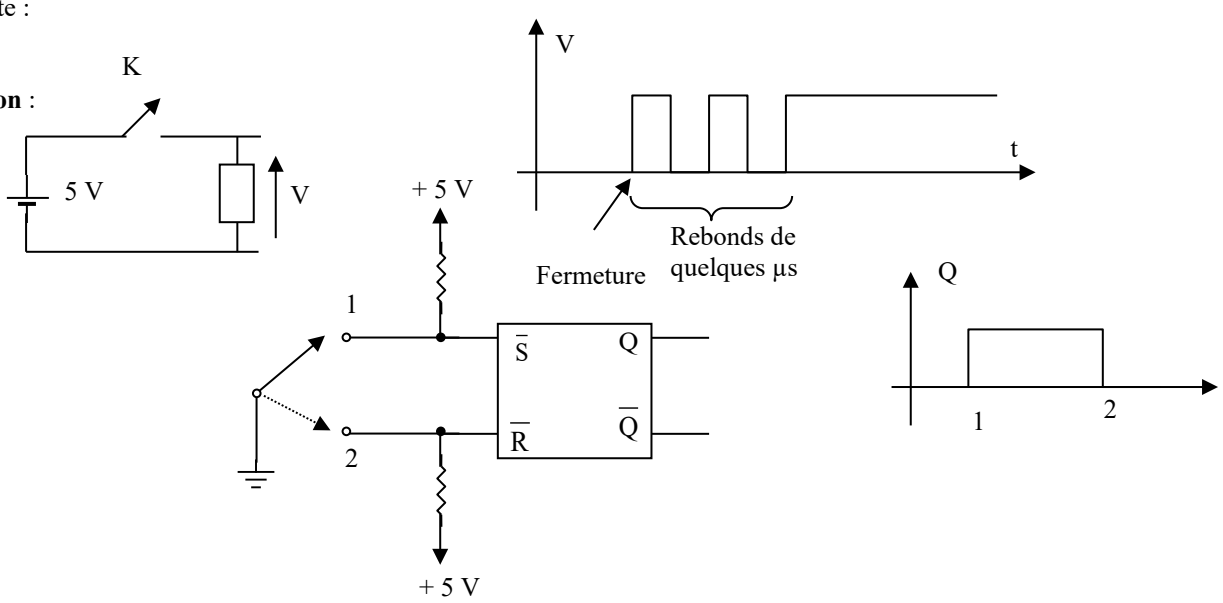
3. Selon les fabricants, l'entrée horloge se désigne par : Clk, Ck, T, CP, etc…

8.2. Applications des bascules

8.2.1 Dispositif antirebond

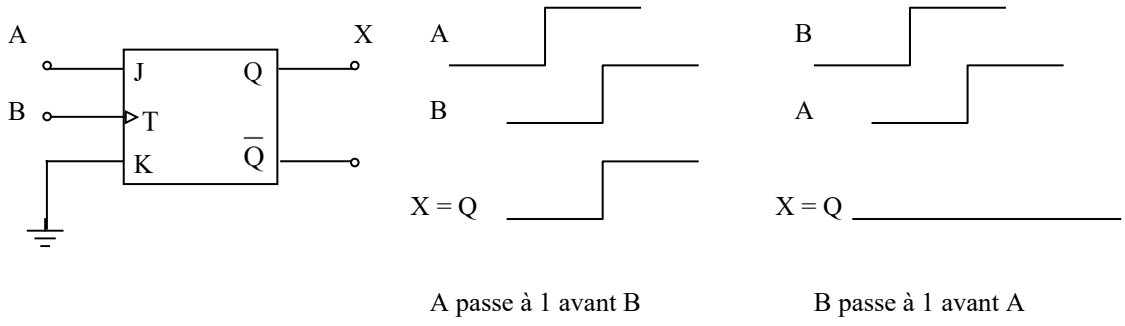
Pour créer des états logiques électriques on utilise des interrupteurs. Malheureusement les pastilles des interrupteurs mécaniques rebondissent toujours l'une sur l'autre quand on les met en contact (même les meilleurs interrupteurs rebondissent). Si on observe la tension obtenue avec un tel interrupteur on a la forme d'onde suivante :

Solution :



Using RS Flip flop to debounce a switch (*Bascule RS utilisée dans un dispositif anti-rebonds*)

8.2.2 Détection d'une séquence d'entrée



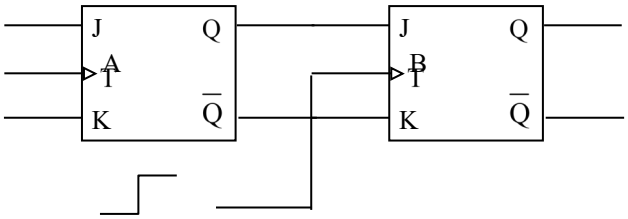
A passe à 1 avant B

B passe à 1 avant A

8.2.3 Stockage et transfert de données

Le stockage de données ou d'information a généralement lieu dans des groupes de bascules, appelés registres. La manipulation la plus fréquente qu'on fait subir aux données dans les registres est le transfert : échange de données d'un registre à un autre.

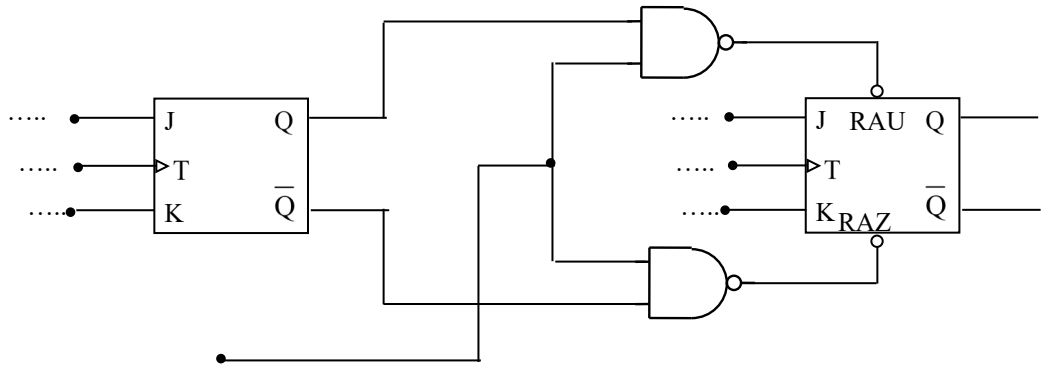
A l'arrivée du front déclencheur la valeur logique en mémoire dans la bascule A est transférée dans la bascule B.



Transfert

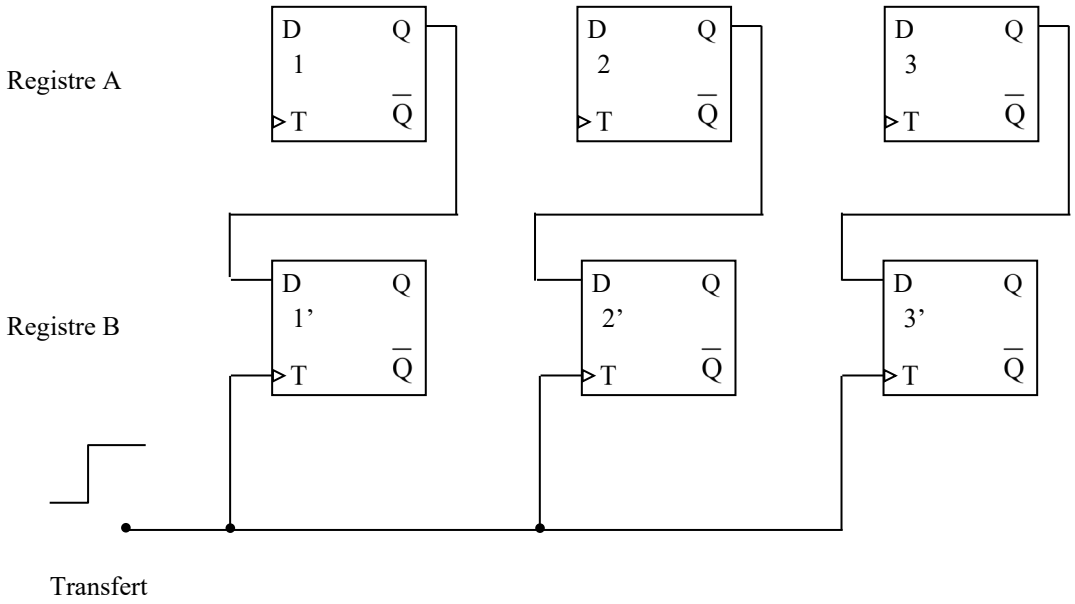
Transfert synchrone

Il est aussi possible d'opérer un transfert en utilisant les entrées asynchrones (prioritaires)



Entrée validation de transfert

8.2.4 Transfert de données en parallèle



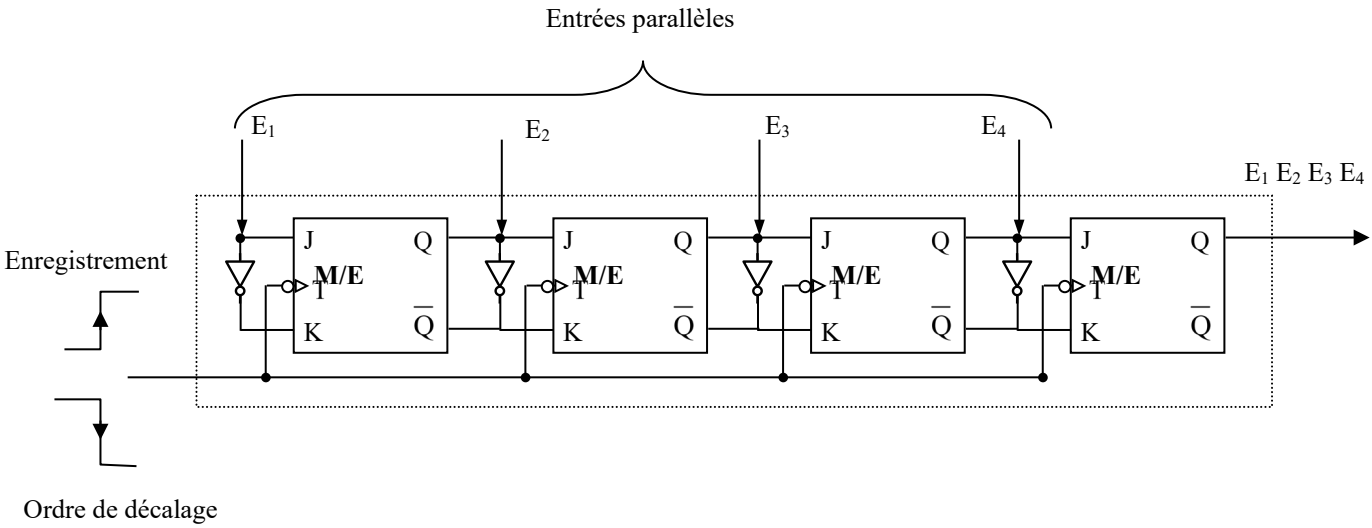
Transfert parallèle du registre A dans le registre B.

8.2.5 Les registres série

On en distingue deux types :

- registres à entrées parallèles et à sortie série
- registres à entrées série et sorties parallèles

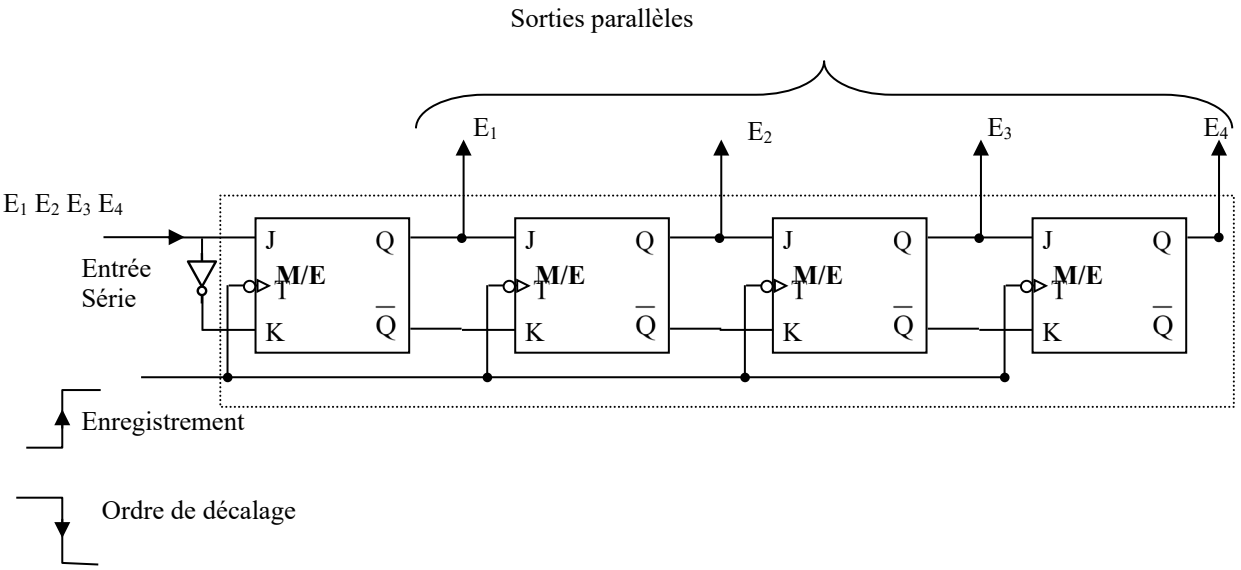
8.2.5.1 Registres à entrées parallèles et à sortie série



Principe

Un mot de 4 bits (dans le cas présent) est présenté à l'entrée parallèle du registre. Ce mot est mémorisé en un seul coup d'horloge à travers le registre, puis ressorti bit à bit en sortie aux coups d'horloge suivants en commençant par le bit de poids le plus faible.

8.2.5.2 Registres à entrée série et à sorties parallèles

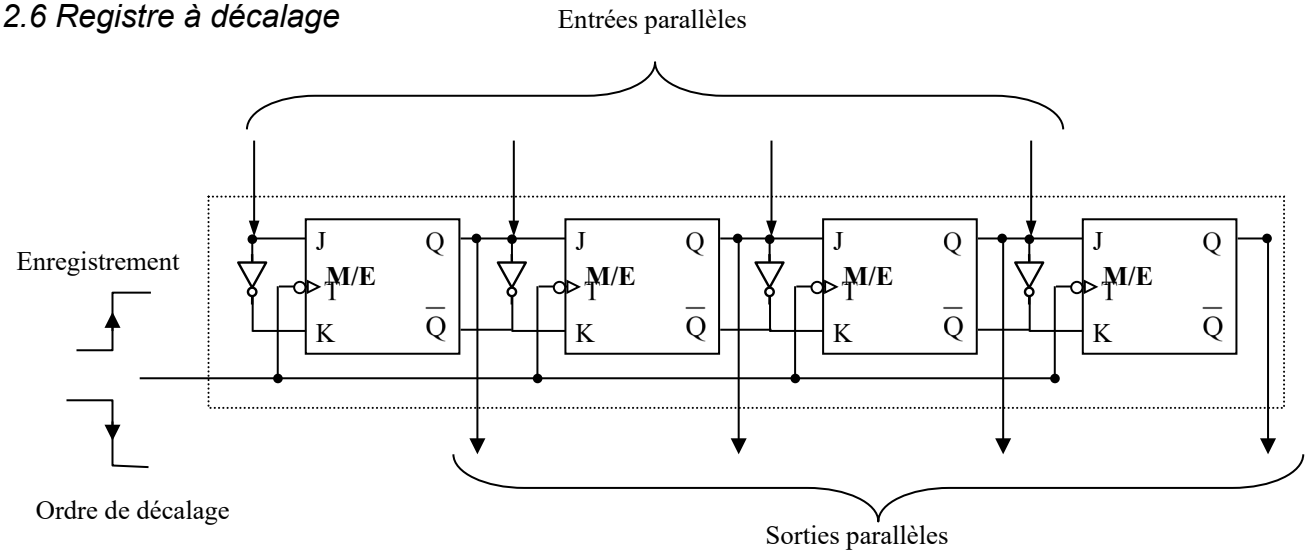


Principe

Ce registre reçoit un mot binaire en série et le ressort en parallèle. Les bits présentés sur l'entrée seront mémorisés sur la bascule JK de gauche lors du premier coup d'horloge, puis seront décalés vers la droite lors des coups d'horloge suivants. Ainsi en 4 coups d'horloge on pourra reconstituer un mot de 4 bits qui aurait été sérialisé.

Application : Ces registres sont très utiles quand on veut faire passer plusieurs bits sur même fil, par exemple un fil téléphonique. On peut ainsi coupler deux ordinateurs qui s'échangent des octets (mots de 8 bits), en se servant comme moyen de couplage d'une simple ligne téléphonique.

8.2.6 Registre à décalage



8.2.6.1 Principe

Au premier coup d'horloge le mot binaire est mémorisé dans le registre puis au second coup d'horloge il est décalé vers la droite.

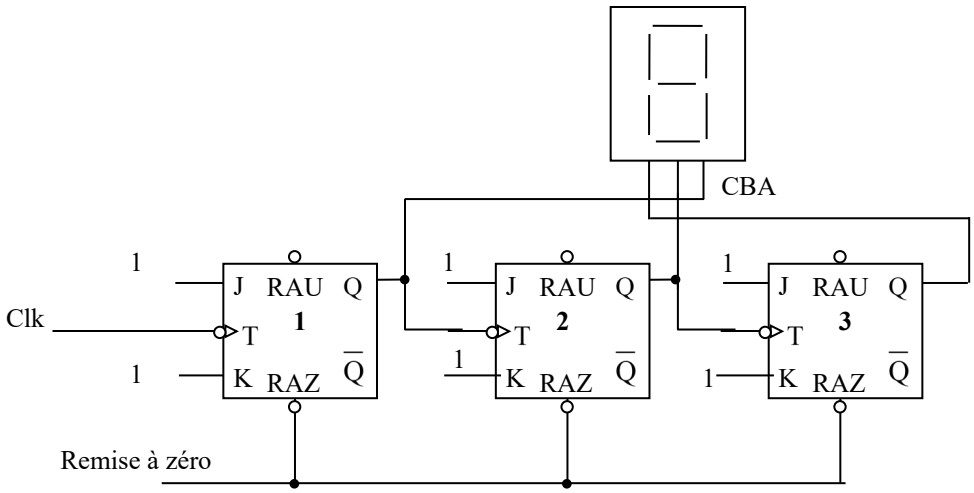
Mot mémorisé : 0 1 0 0 = 4
↘ ↘ ↘
Mot décalé 0 0 1 0 = 2
Si le premier bit du mot décalé est maintenu à 0 on remarque que ce décalage revient à une division par 2.

Le décalage pourra se faire aussi vers la gauche.

8.2.7 Les compteurs (Counters)

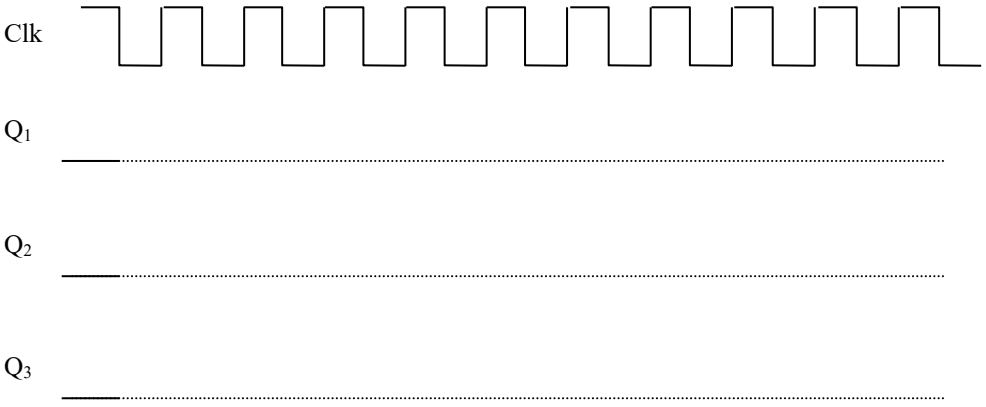
8.2.7.1 Compteurs asynchrones Modulo 2^N

On considère N bascules connectées en compteurs progressifs. La sortie de chaque bascule opère une division de fréquence par 2. Pour un compteur octal (modulo 8) on obtient les chronogrammes ci-dessous.



Ripple counter : It consists of a cascade of JK flip-flops. The input pulses to be counted are connected to the clock input of the first stage and the output of the first stage is connected to the clock of the second stage.

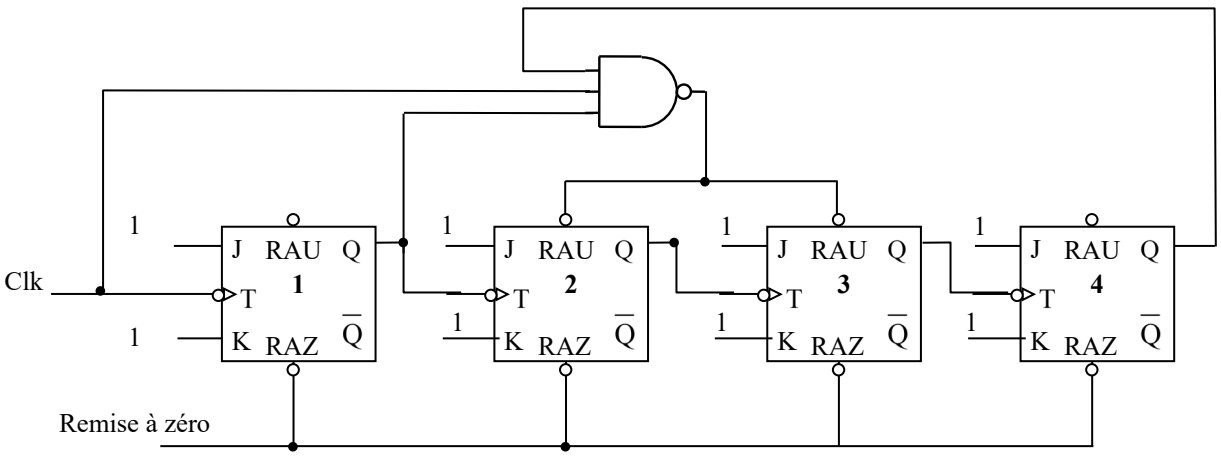
Compteur octal et affichage.

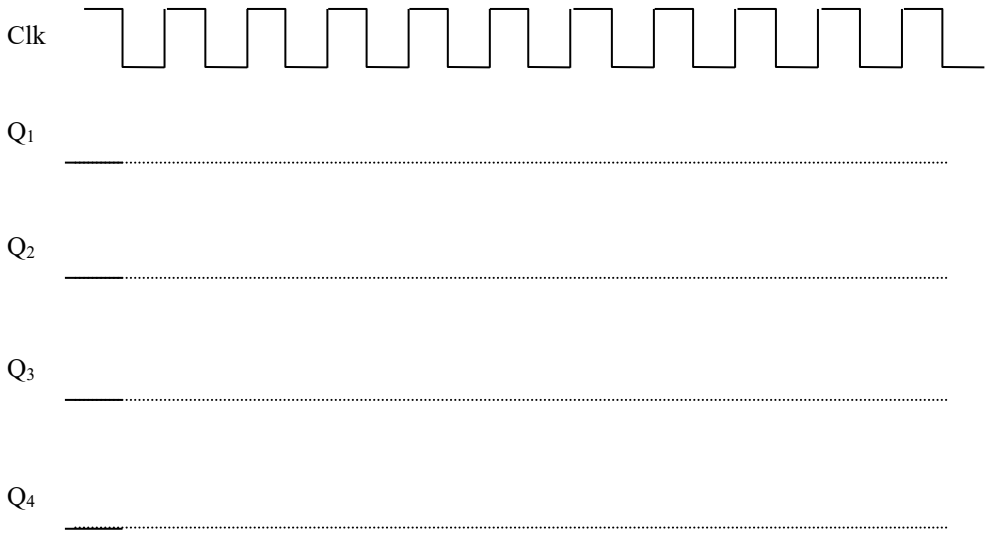


8.2.7.2 Compteurs à Modulo < 2^N

Si on veut construire un compteur modulo N c'est-à-dire comptant jusqu'à N-1 et revenant à zéro, on cherche la puissance de 2 immédiatement supérieure à N. Soit, par exemple, un compteur modulo 10 ou compteur décimal, comptant de 0 à 9 puis repassant à 0 à la dixième impulsion d'horloge. La puissance de 2 immédiatement supérieure à 10 est $2^4 = 16$. L'exposant de cette puissance donne le nombre de bascule à utiliser, quatre dans notre cas.

On câble ces bascules en compteur progressif, on connecte ensuite toutes les sorties Q qui sont à 1 pour N-1 (en binaire) à l'entrée d'une porte Nand et on ajoute une entrée d'horloge à cette porte. La sortie de la porte Nand va aux entrées RAU des bascules restantes.





8.2.7.3 Méthodes de synthèse des bascules synchrones

Table d'excitation de la bascule JK

Considérons la table de vérité d'une bascule JK

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Table de vérité.

La table d'excitation permet de déterminer les commandes d'entrée J et K à appliquer pour obtenir la sortie désirée connaissant la sortie avant l'impulsion d'horloge ou le front actif d'horloge.

On distingue 4 cas de figure :

1er cas Q = 0 et on désire $Q^+ = 0$

Selon la table de vérité, J = 0 et K = 1 commandent $Q^+ = 0$ et J = K = 0 commandent $Q^+ = Q = 0$ par hypothèse. D'où l'excitation à appliquer : J = 0 quel que soit l'état de K (0 ou 1). Dans un tel cas de valeur indifférente on note cette valeur X. D'où la première ligne de la table d'excitation (figure ci-dessous).

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table d'excitation des bascules JK

2éme cas Q = 0 et on désire Q+ = 1

Selon la table de vérité, J = 1 et K = 0 commandent $Q^+ = 1$ et J =K = 1 commandent $Q^+ = \overline{Q} = 1$ par hypothèse. D'où l'excitation à appliquer : J = 1 quel que soit l'état de K. Nous en déduisons la deuxième ligne de la table d'excitation.

3éme cas Q = 1 et on désire Q+ = 0

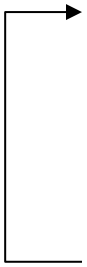
Selon la table de vérité, J = 0 et K = 1 commandent $Q^+ = 0$ et J = K = 1 commandent $Q^+ = \overline{Q} = 0$ par hypothèse. D'où l'excitation à appliquer : K = 1 quel que soit l'état de J. On en déduit la troisième ligne de la table d'excitation.

4éme cas Q = 1 et on désire Q+ = 1

Selon la table de vérité, J = 1et K = 0 commandent $Q^+ = 1$ et J = K = 0 commandent $Q^+ = Q = 0$ par hypothèse. D'où l'excitation à appliquer : K = 0 quel que soit l'état de J. On en déduit la troisième ligne de la table d'excitation.

Compteurs synchrones

Dans les compteurs synchrones le signal commande toutes les bascules en même temps. Les états des bascules JK par exemple dépendent des entrées JK. A titre d'exemple nous choisissons de faire la synthèse d'un compteur modulo $8 = 2^3$. Il nous faut donc 3 bascules (A, B, C).Si ces bascules permettent l'affichage d'un nombre CBA on a la table des fonctions suivantes :



C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

La première bascule change donc d'état à chaque impulsion d'horloge. Ceci s'obtient en faisant J = K= 1 et à l'aide d'un signal d'horloge. La bascule B change d'état toutes les deux impulsions d'horloge. Son état dépend de A. L'état de la bascule C dépend de A et B. Pour résoudre ce genre de problème, dressons tout d'abord la table des états recherchés.

Q_C \ Q_BQ_A	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Table des différents états du compteur.

Q_A, Q_B, Q_C représentent les sorties du compteur octal.

Indiquons ensuite dans les tables, l'état commandé à la bascule. Cet état dépend de J et de K. Nous dressons donc 2 tables pour chaque bascule, une pour J et l'autre pour K. Pour la bascule A, on doit avoir $J_A = K_A = 1$. Retrouvons ce résultat pour la bascule A.

A l'état (0) la sortie Q_A est 0. A l'impulsion d'horloge suivante la sortie Q_A doit être égale à 1, il faut donc commander (d'après la table d'excitation) J = 1 et K = X. Ecrivons ces valeurs dans la case de l'état (0) pour arriver à l'état (1) avec $Q_A = 1$ à l'impulsion suivante.

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

A l'état (1) Q_A vaut 1, il faut commander 0 donc il faut J = X et K = 1. Ecrivons ces valeurs dans la case de l'état (1).

De même :

Dans les cases de l'état (2) écrit J = 1 et K = X
Dans les cases de l'état (3) écrit J = X et K = 1
Dans les cases de l'état (4) écrit J = 1 et K = X
Dans les cases de l'état (5) écrit J = X et K = 1
Dans les cases de l'état (6) écrit J = 1 et K = X
Dans les cases de l'état (7) écrit J = X et K = 1

Nous revenons de cette façon à l'état 0. Si on considère les X comme des 1, on obtient les commandes $J_A = 1$ et $K_A = 1$.

Table de Karnaugh de J_A

Q_C \ Q_BQ_A	00	01	11	10
0	1	X	X	1
1	1	X	X	1

$J_A = 1$

Table de Karnaugh de K_A

Q_C \ Q_BQ_A	00	01	11	10
0	X	1	1	X
1	X	1	1	X

$K_A = 1$

Procédons de la même façon pour la bascule B.

Dans les cases de l'état (0) écrit J = 0 et K = X
Dans les cases de l'état (1) écrit J = 1 et K = X
Dans les cases de l'état (2) écrit J = X et K = 0
Dans les cases de l'état (3) écrit J = X et K = 1
Dans les cases de l'état (4) écrit J = 0 et K = X
Dans les cases de l'état (5) écrit J = 1 et K = X
Dans les cases de l'état (6) écrit J = X et K = 0
Dans les cases de l'état (7) écrit J = X et K = 1

Table de Karnaugh de J_B

Q_C \ Q_BQ_A	00	01	11	10
0	0	1	X	X
1	0	1	X	X

$J_B = Q_A$

Table de Karnaugh de K_B

Q_C \ Q_BQ_A	00	01	11	10
0	X	X	1	0
1	X	X	1	0

$K_B = Q_A$

D'où les équations $J_B = Q_A = K_B$ déduites des tables de Karnaugh.

La bascule C on aura les tables de Karnaugh suivantes

Table de Karnaugh de J_C

Q_C \ Q_BQ_A	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_C = Q_AQ_B$

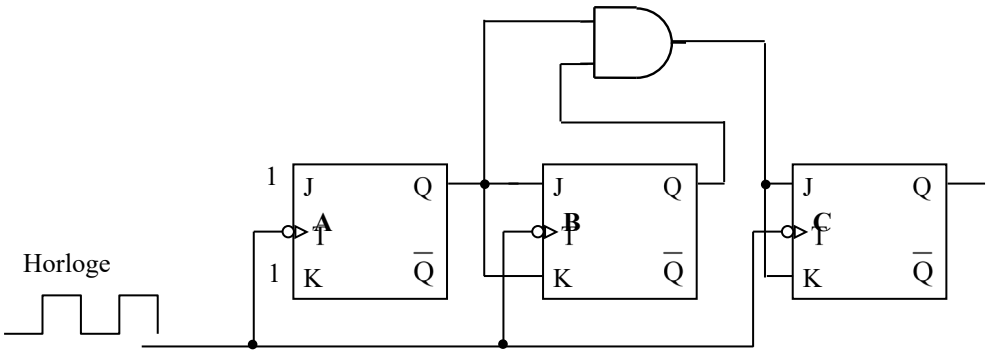
Table de Karnaugh de K_C

Q_C \ Q_BQ_A	00	01	11	10
0	X	X	X	X
1	0	0	1	0

$K_C = Q_AQ_B$

D'où $J_C = K_C = Q_AQ_B$

D'où le schéma le compteur.



Compteur synchrone modulo 8.

Pour ce type de compteur, il est intéressant d'utiliser des bascules en circuits intégrés avec des portes And sur les entrées J et K. On n'a pas besoin d'ajouter de portes supplémentaires. Exemple CI : 7470, 7472, 74102, 74110.

On peut à titre d'exercice essayer de construire un compteur synchrone décimal.

Bibliographie

1- Jean LETOCHA
Introduction aux circuits logiques

2- Ronald J. TOCCI
Circuits numériques : Théorie et Applications
Edition Dunod

3- Roger L. TOKHEIM
Techniques Numériques : Cours et Problèmes
Série Schaum

4- Circuits logiques et numériques
Université Libre de Bruxelles / Ecole Polytechnique / BEAMS / MILOJEVIC Dragomir

Exercices

Exercice 1

2./ Déterminer les bases dans lesquelles les nombres suivants sont exprimés :

$(34)_{?} = (22)_{10}$

$(75)_{?} = (117)_{10}$

3./ Effectuer les opérations suivantes:

a./ $(1011,1101)_2 + (11,1)_2 = (\ldots\ldots)_2$

b./ $(1010,0101)_2 - (110,1001)_2 = (\ldots\ldots)_2$

c./ $(110)_2 \times (10,1)_2 = (\ldots\ldots)_2$

d./ $(1111)_2 \div (110)_2 = (\ldots\ldots)_2$

Exercice 2

Soit la table de vérité suivante:

B3	B2	B1	G3	G2	G1
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Questions:

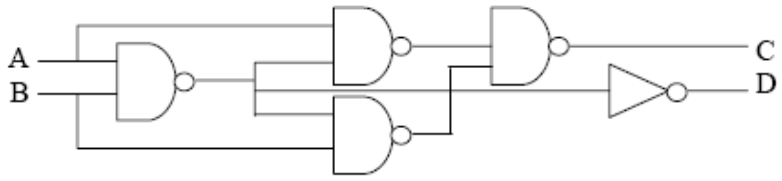
a./ Donner les équations algébriques de G3, G2, G1 en fonction de B3, B2, B1.

b./ Simplifier algébriquement G3, G2, G1.

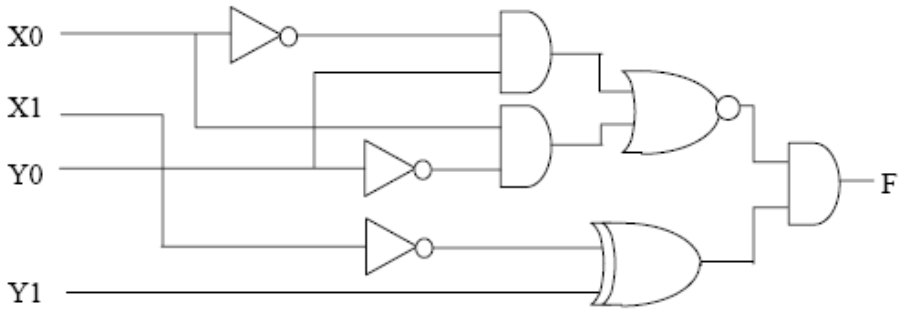
Exercice 3

Etudier les circuits ci-dessous et dites la fonction qu'ils réalisent.

a.

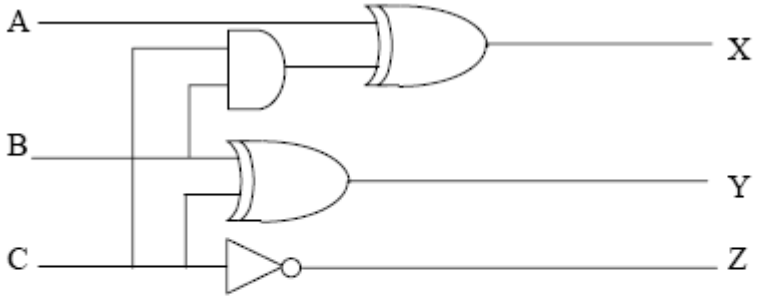


b.



BB

c.



Remarque: Vous pouvez éventuellement utiliser une table de vérité afin d'identifier plus facilement la fonction du circuit.

Exercice 4

Soient les deux fonctions suivantes :

$S = X.\overline{Y}$

$I = \overline{X}.\ Y$

1/ Donner l'équation simplifiée (algébriquement) de la fonction **E** telle que: $E = \overline{S}.\ \overline{I}$

2/ Représenter sur un seul schéma, en utilisant que des NORs à 2 entrées et des inverseurs, les trois fonctions.

3/ Analyser par table de vérité et dites quel est le rôle de chacune des fonctions.

Exercice 5

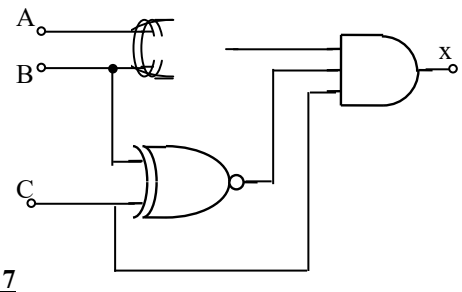
Soit F(e,d,c,b,a) une fonction logique, où e, d, c, b, a ont respectivement les poids 16,8,4,2,1.
Cette fonction est définie par la table de Karnaugh suivante:

cba / ed	000	001	011	010	110	111	101	100
00	0	0	0	0	0	0	1	0
01	0	0	0	0	1	1	1	0
11	0	0	1	0	0	1	1	1
10	1	0	1	1	1	1	0	1

1- Simplifier cette fonction en indiquant le meilleur choix de regroupements.

2- Dessiner le schéma de la réalisation ‘’somme de produits’’ de F à l’aide d’opérateurs NAND et d’inverseurs.

Exercice 6



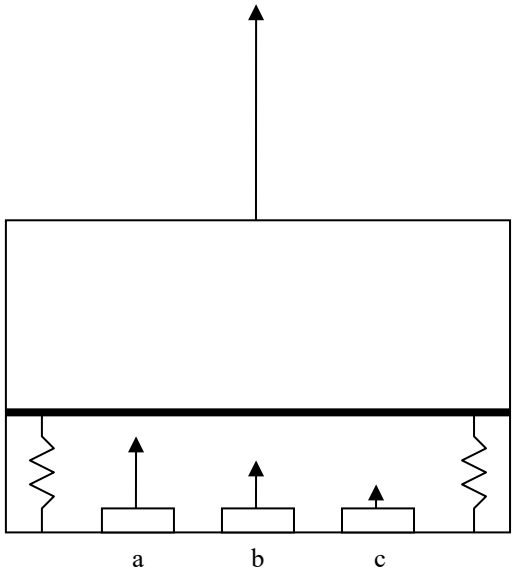
Déterminer les conditions d’entrée qui produisent x = 1 à la figure ci-contre.

Exercice 7

Un monte-charge doit permettre le levage de masses comprises entre 10 kg et 60 kg. Pour cela il comporte une plate forme reposant sur des ressorts. Suivant l’importance des charges à soulever, trois contacts réglables sont mis en circuit. (voir figure ci-dessous).
Les conditions de fonctionnements sont les suivantes :

Condition 1 : A vide le monte charge peut fonctionner : aucun des trois contacts a, b, c n’est actionné.
Condition 2 : Pour des charges comprises entre 5 et 10 kg, le monte charge ne peut fonctionner : le contact a seul est actionné.
Condition 3 : Pour des charges comprises entre 10 et 60 kg, le monte charge doit fonctionner : deux contacts a et b sont actionnés.
Condition 4 : Pour des charges supérieures à 60 kg, le monte charge ne peut fonctionner : les trois contacts sont actionnés.

Concevoir le circuit qui réalise l’autorisation S = 1 ou l’interdiction S = 0 du déplacement du monte-charge.



Exercice 8

La figure 1 montre le schéma d'un circuit d'ouverture d'une serrure de sécurité en fonction de 4 clefs binaires (A,B,C,D).

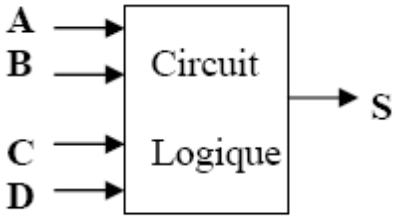


Figure 1

D'après le mode de fonctionnement, la serrure de sécurité (S) est ouverte chaque fois qu'au moins deux (2) clefs situées (Côte à Côte) sont introduites, et il est impossible que les Clefs A et D soient introduites en même temps.

1. Etablir la table de vérité de S.
2. Donner les formes Simplifiées Conjonctives et Disjonctives de S par la méthode de Karnaugh.
3. Représenter S simplifiée à l'aide uniquement de portes NOR à 2 entrées.

Exercice 9

a/ Démontrer algébriquement les égalités suivantes

1/ $A\overline{B} + B\overline{C} + \overline{A}C = \overline{A}B + \overline{B}C + A\overline{C}$

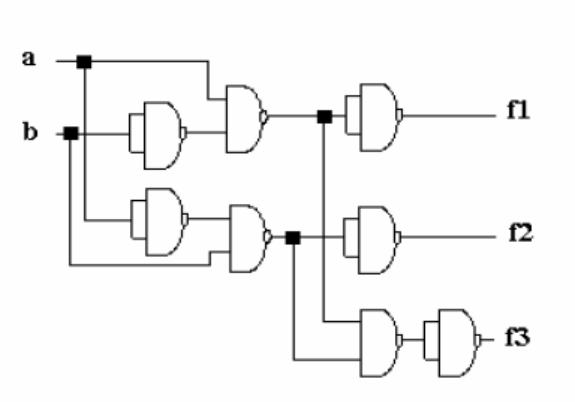
2/ $AB+(A+B)(\overline{AB})C = AB + AC + BC$

b/ Soit la fonction

$f(a,b,c,d) = \overline{a}\,\overline{b}\,\overline{c}\,\overline{d} + \overline{a}\,\overline{c}\,d + a\overline{c}\,\overline{d} + abc + a\overline{b}\,c + \overline{a}\,\overline{b}\,c\,\overline{d}$

1. Donnez la forme canonique disjonctive.
2. Simplifiez algébriquement $f(a,b,c,d)$.
3. Retrouvez le même résultat en utilisant un tableau de Karnaugh à 3 variables.

c/ Etudiez le circuit ci-dessous et dites ce qu'il fait (vous pouvez vous aider d'une table de vérité).



Exercice 10

On désire réaliser le circuit de commande de la distribution d'eau d'un petit immeuble de 3 étages. Chaque étage est alimenté par deux tuyaux d'eau, chaque tuyau est commandé par un robinet.

- Le premier étage est alimenté par les tuyaux : T_{1a} et T_{1b}.
- Le second étage est alimenté par les tuyaux : T_{2a} et T_{2b}.
- Le troisième étage est alimenté par les tuyaux : T_{3a} et T_{3b}.

L'eau étant rare, le circuit doit gérer sa distribution de la manière suivante :

- Si les trois étages demandent de l'eau alors chaque étage est alimenté par son tuyau , respectivement T_{1a} pour le premier,T_{2a} pour le second et T_{3a} pour le troisième.
- Si deux étages seulement demandent de l'eau alors chaque étage est alimenté par son tuyau, respectivement T_{1b} pour le premier, T_{2b} pour le second et T_{3b} pour le troisième.
- Si un seul étage demande de l'eau alors il est alimenté par ses deux tuyaux T_{1a} et T_{1b} pour le premier, T_{2a} et T_{2b} pour le second , T_{3a} et T_{3b} pour le troisième.

Questions

1. Définir les variables d'entrées et les fonctions de sorties
2. Donner la table de vérité.
3. Donner les équations simplifiées.
4. Faire le schéma.

Exercice 11

Tracer les logigrammes des expressions logiques suivantes :

$S_1=b\bar{c}\bar{d}+ab\bar{d}+\bar{a}bc\bar{d}$ $\quad$ $S_2=x+\bar{y}z+xy\bar{t}\oplus\bar{z}yt$

Exercice 12

Dessiner la forme d'onde en sortie S du système logique de la ***Figure 1***.

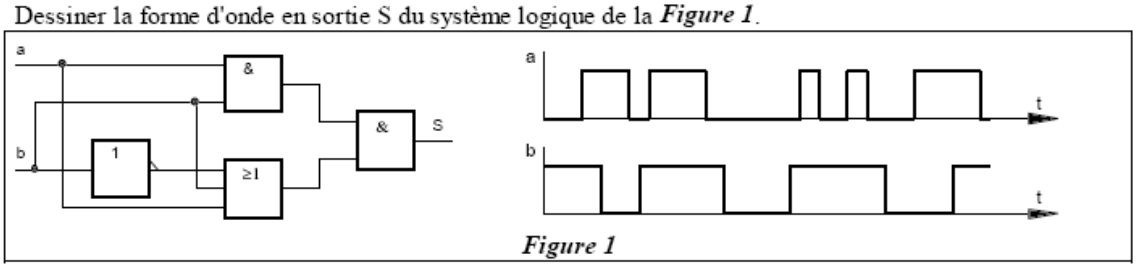


Figure 1

Exercice 13

Trois interrupteurs a, b et c commandent l'allumage de deux lampes R et S suivant les conditions :

- dès qu'un ou plusieurs interrupteurs sont activés la lampe R s'allume,
- la lampe S ne doit s'allumer que si au moins deux interrupteurs sont activés.

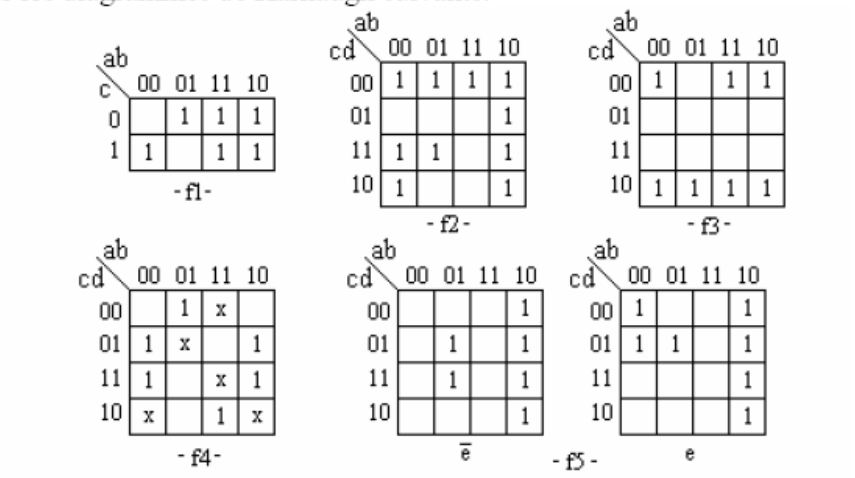
Trouver les expressions de R et S et dessiner les logigrammes.

Exercice 14

1./ Simplifier algébriquement les fonctions suivantes:

$$f1 = a.b.c + a.\overline{b}.c + a.b.\overline{c}.d$$
$$f2 = a.\overline{c} + b.d + a.c.\overline{d} + \overline{b}.d$$
$$f3 = \Re(2,3,6,7)$$
$$f4 = \Re(1,2,5,6,8,11,12,15)$$

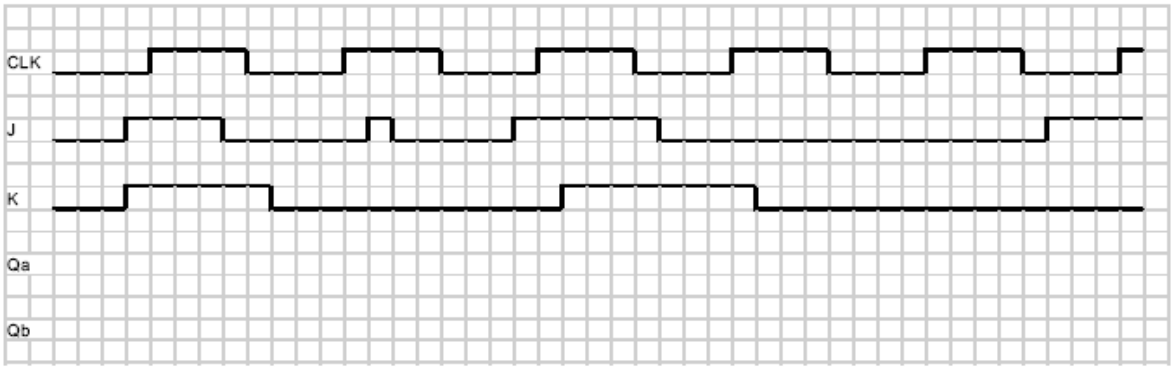
2./ Simplifier les diagrammes de Karnaugh suivants:



Exercice 15

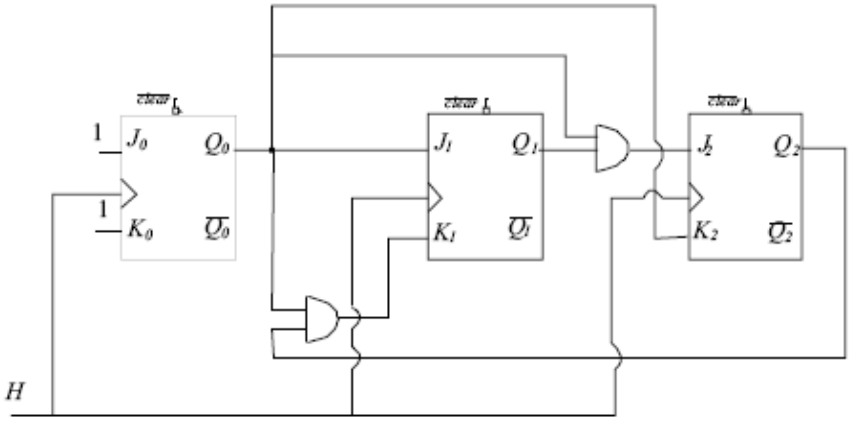
Compléter le chronogramme suivant pour une bascule JK avec:

1. Déclenchement sur niveau haut de l'horloge.
2. Déclenchement sur front descendant de l'horloge.



Exercice 16

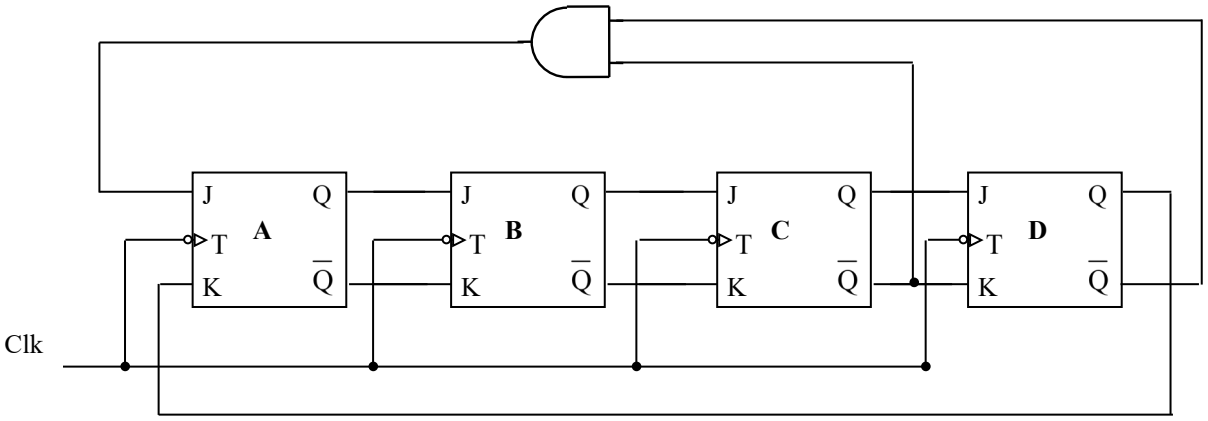
Sur le circuit suivant (compteur synchrone), donner le graphe des états $Q_2Q_1Q_0$ obtenus après chaque impulsion d'horloge. On suppose initialement $Q_2Q_1Q_0 = 000$.



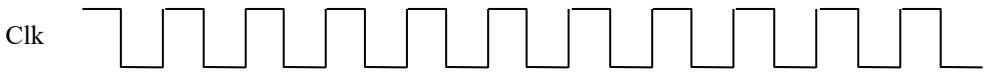
Que se passerait il si initialement $Q_2Q_1Q_0 = 100$ ou 101 ? Proposer un câblage permettant d'éviter cette situation à la mise sous tension.

Exercice 17

On considère le compteur ci-dessous :



Le signal d'horloge est donné par Clk indiqué ci-dessous.



Tracer les sorties Q_A, Q_B, Q_C, et Q_D en précisant la séquence des états du compteur à partir de l'état initial DCBA = 1001.

NB : Ce compteur n'est pas classique. Ne pas s'inquiéter de sa manière de compter! C'est le raisonnement scientifique qui compte.