
Cours d'électronique numérique

Maryam Siadat & Camille Diou


Version du 2 novembre 2004

[Notes sur cet ouvrage]

Ce document est à la date d'aujourd'hui (2 novembre 2004) toujours en phase d'écriture. Il est donc nécessairement incomplet et peut même encore comporter des erreurs qui n'auraient pas été détectées.

Ce document doit notamment s'enrichir à l'avenir des points suivants (dans le désordre) :

- ☞ la logique mixte
- ☞ compléter la simplification des fonctions logiques
 - ☞ méthode de Quine/McCluskey
 - ☞ diagrammes de Venn, Johnston et Carroll
 - ☞ familles logiques et spécifications électriques
- ☞ étude des systèmes programmables évolués (en complément du chapitre actuel)
- ☞ synthèse des systèmes séquentiels synchrones
 - ☞ machines d'états (Moore, Huffman, Mealey)
- ☞ synthèse des systèmes séquentiels asynchrones
- ☞ arithmétique binaire et opérateurs arithmétiques
- ☞ compléter les exercices et corrigés

Ce document a été réalisé à l'aide des logiciels $\text{T}_{\text{E}}\text{X}$ et $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ sous les environnements $\text{T}_{\text{E}}\text{XLive}$ et $\text{TeT}_{\text{E}}\text{X}$. Les diagrammes sont réalisés à l'aide de Xy-pic . Une partie des schémas électronique est réalisée à l'aide du paquetage CIRC .

La police utilisée pour le texte principale est Fourier.

Les descriptions bibliographiques/historiques présentes dans les entêtes de chapitres sont composéee en DayRoman.

L'extrait du texte de Blaise Pascal du chapitre II est également composé dans la police DayRoman, mais dotée – notamment – de la ligature ct et du S long (f).

1 4 6 7 3 9 2 3

Première partie

Les nombres

The image is a collage of various mathematical and linguistic elements. At the top, there is a row of numbers: 1, 4, 6, 7, 3, 9, 2, 3. Below this, there are several dice, some showing numbers like 2, 3, 4, 5, 6, 7, 8, 9. A calendar grid is visible on the left side, showing a grid of numbers from 0 to 31. In the center, there is a hand chart with the text 'MAIN DROITE' and numbers 25, 20, 15, 10, 15, 10, 15, 10. At the bottom, there is a large row of numbers: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0. The background is filled with various symbols, including musical notes, geometric shapes, and other numbers.

Chapitre I

Les systèmes de numération

Gottfried Wilhelm von Leibniz
★ juil. 1646, Allemagne
† 1716



Ce philosophe d'origine Allemande est l'inventeur d'une machine permettant de calculer directement les 4 opérations de base. Il est aussi celui qui a introduit la notion de binaire en Occident.

I.1 La représentation polynomiale

Si nous manipulons les nombres de manière intuitive, c'est la plupart du temps dans la base décimale, naturelle et universelle. Mais cela ne doit pas masquer la nature même de la numération qui peut prendre plusieurs formes, parmi lesquelles on trouve la théorie des ensembles et la représentation polynomiale.

La représentation polynomiale d'un nombre est sa représentation sous la forme suivante :

$$a_{n-1}b^{n-1} a_{n-2}b^{n-2} a_{n-3}b^{n-3} \dots a_2b^2 a_1b a_0 a_{-1}b^{-1} a_{-2}b^{-2} \dots a_{-m}b^{-m}$$

où b est appelée la base.



Si la base 10 nous est familière, d'autres bases existent et les bases les plus utilisées en informatique sont les bases 10, 2, 8 et 16 appelées respectivement « décimale », « binaire », « octale » et « hexadécimale ».



Si la base 10 nous est familière, d'autres bases existent et les bases les plus utilisées en informatique sont les bases 10, 2, 8 et 16 appelées respectivement « décimale », « binaire », « octale » et « hexadécimale ».

I.2 Le système binaire

I.2.1 Introduction

Le système décimal est malheureusement difficile à adapter aux mécanismes numériques, car il est difficile de concevoir du matériel électronique fonctionnant sur dix plages de tensions différentes.

On lui préférera donc le système binaire :

- base $B=2$;
- 2 symboles : $\{0,1\}$ appelés « éléments binaires » ou « bits » (bit=*Binary digIT*) ;
- le système binaire est pondéré par 2 : les poids sont les puissances de 2 ;

▷ **Exemple 1.1**

$$\begin{array}{cccccccccccc}
 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & & 2^{-1} & 2^{-2} & 2^{-3} \\
 1 & 0 & 1 & 1 & 0 & 0 & 1 & , & 0 & 1 & 1
 \end{array}$$

- les différentes puissances de 2 sont :

$$\begin{array}{cccccccccccc}
 2^0 & 2^1 & 2^2 & 2^3 & 2^4 & 2^5 & 2^6 & 2^7 & 2^8 & 2^9 & 2^{10} \\
 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 & 512 & 1024
 \end{array}$$

- un ensemble de 8 bits est appelé « octet » (ou *byte*).

I.2.2 Comptage binaire

On présente les nombres binaires en général avec un nombre fixe de bits, nombre imposé par les circuits mémoires utilisés pour représenter ces nombres.

Suite des nombres binaires à 4 bits :

Poids :	2^3	2^2	2^1	2^0	B10
	0	0	0	0	0
	0	0	0	1	1
	0	0	1	0	2
	0	0	1	1	3
	0	1	0	0	4
	0	1	0	1	5
	0	1	1	0	6
	0	1	1	1	7
	1	0	0	0	8
	1	0	0	1	9
	1	0	1	0	10
	1	0	1	1	11
	1	1	0	0	12

1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Le bit le plus significatif – le bit le plus à gauche – est appelé « bit de poids fort » ou MSB (*Most Significant Bit*).

Le bit le moins significatif – le bit le plus à droite – est appelé « bit de poids faible » ou LSB (*Less Significant Bit*).

Si on utilise N bits, on peut représenter 2^N valeurs différentes de 2^0 à 2^{N-1}

▷ **Exemple 1.2**

$$N \ 8 : 00000000 \rightarrow 11111111 \leftrightarrow 255$$

► **Remarque 1.1**

Comme l'on traite souvent en micro-informatique de nombres à 8 ou 16 éléments binaires (e.b.), on se sert des systèmes :

- octal : à base 8 ;
- hexadécimal : à base 16.

I.3 Le système octal

- base $B=8$;
- 8 symboles : $\{0, 1, 2, 3, 4, 5, 6, 7\}$;

L'intérêt de ce système est que la base 8 est une puissance de 2 ($8 = 2^3$), donc les poids sont aussi des puissances de 2.

Chaque symbole de la base 8 est exprimé sur 3 e.b. : $(a_i)_8 \ b_{i_2} b_{i_1} b_{i_0}$

▷ **Exemple 1.3**

$$(52,3)_8 \ 101 \ 010,011$$

I.4 Le système hexadécimal

- base $B=16$;
- 15 symboles : $\{0, 1, 2, \dots, 9, A, B, C, D, E, F\}$ appelés « digits » ;
- chaque symbole est exprimé en binaire sur 4 bits ;

▷ **Exemple 1.4**

$$(F3D,2)_{16} \quad 1111 \ 0111 \ 1101,0010$$

I.5 Conversion d'un système de numération à un autre

I.5.1 Base B vers base 10

$$(a_n \dots a_0)_B \quad a_n B^{-n} \dots a_0 B^0 \quad (a'_m \dots a'_0)_{10}$$

▷ **Exemple 1.5**

$$(1001,1)_2 \quad 1.2^3 \ 0.2^2 \ 0.2^1 \ 1.2^0 \ 1.2^{-1} \quad 8 \ 0 \ 0 \ 1 \ 0,5 \ 9,5$$

$$(A12)_{16} \quad A.16^2 \ 1.16^1 \ 2.16^0 \quad 2560 \ 16 \ 2 \ 2578$$

I.5.2 Base 10 vers base B

I.5.2.a Première méthode

Elle consiste à soustraire successivement la plus grande puissance de B

▷ **Exemple 1.6**

$$\begin{array}{rcl} 100 & 1.2^6 & 36 \\ 36 & 1.2^5 & 4 \\ 4 & 1.2^2 & 0 \end{array} \} \rightarrow (100)_{10} \ (1100100)_2$$

I.5.2.b Deuxième méthode

Elle consiste à diviser par B autant de fois que cela est nécessaire pour obtenir un quotient nul. Ensuite on écrit les restes dans l'ordre inverse de celui dans lequel ils ont été obtenus.

Pour la partie fractionnaire on multiplie par B (résultat nul ou selon la précision demandée)

▷ **Exemple 1.7**

$$(20,4)_{10} \quad (?)_2$$

Partie entière :

$$\begin{array}{r|l}
 20 & 2 \\
 \hline
 0 & 10 \\
 & 0 \\
 & \hline
 & 2 \\
 & 5 \\
 & 1 \\
 & \hline
 & 2 \\
 & 2 \\
 & 0 \\
 & \hline
 & 1 \\
 & 1 \\
 & \hline
 & 2 \\
 & 0
 \end{array}$$

Partie fractionnaire :

$$\begin{array}{r}
 0,4 \times \\
 \hline
 0 \quad 0,8
 \end{array}
 \rightarrow
 \begin{array}{r}
 0,8 \times \\
 \hline
 1 \quad 1,6
 \end{array}
 \rightarrow
 \begin{array}{r}
 0,6 \times \\
 \hline
 1 \quad 1,2
 \end{array}$$

Le résultat est donc 10100,0110.

I.5.3 Base 2^n vers base 2

Chaque symbole de la base B 2^n peut être représenté par n e.b.

▷ **Exemple 1.8**

$$\begin{array}{l}
 (3A9)_{16} \quad (0011 \ 1010 \ 1001)_2 \\
 (742,5)_8 \quad (111 \ 100 \ 010, \ 101)_2
 \end{array}$$

I.5.4 Base 2 vers base 2^n

Il suffit de regrouper les e.b. par paquets de n e.b.

▷ **Exemple 1.9**

$$\begin{array}{rcccl}
 (1011011)_2 & \underbrace{(001)}_1 & \underbrace{011}_3 & \underbrace{011}_3 &)_2 & (133)_8 \\
 & & \underbrace{(0101)}_5 & \underbrace{1011}_B &)_2 & (5B)_{16}
 \end{array}$$

I.5.5 Base i vers base j

– si i et j sont des puissances de 2, on utilise la base 2 comme relais ;

▷ **Exemple 1.10**

base 8 \rightarrow base 2 \rightarrow base 16

– sinon, on utilise la base 10 comme relais.

▷ **Exemple 1.11**

base 5 \rightarrow base 10 \rightarrow base 2

Chapitre II

Codage des nombres dans les machines numériques



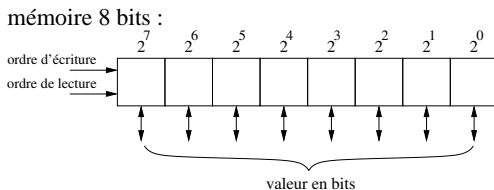
Blaise Pascal
★ 19 juin 1623, Clermont, France
† 19 août 1662, Paris, France

« Ami lecteur, cet avertissement servira pour te faire savoir que j'expose au public une petite machine de mon invention, par le moyen de laquelle seul tu pourras, sans peine quelconque, faire toutes les opérations de l'arithmétique, et te soulager du travail qui t'a souvent fatigué l'esprit, lorsque tu as opéré par le jeton ou par la plume : je puis, sans présomption, espérer qu'elle ne te déplaira pas, après que Monseigneur le Chancelier l'a honorée de son estime, et que, dans Paris, ceux qui sont les mieux versés aux mathématiques ne l'ont pas jugée indigne de leur approbation. Néanmoins, pour ne pas paraître négligent à lui faire acquiescer aussi la tienne, j'ai cru être obligé de t'éclaircir sur toutes les difficultés que j'ai estimées capables de choquer ton sens lorsque tu prendras la peine de la considérer. »

(Blaise Pascal, Avis nécessaire à ceux qui auront curiosité de voir la machine d'arithmétique, et de l'en servir, 1645).

Les systèmes logiques sont constitués de mécanismes qui ne permettent de noter que 2 états : « 0 » ou « 1 ». Une mémoire élémentaire est donc une unité contenant « 0 » ou « 1 ». Plusieurs de ces unités sont assemblées pour représenter un nombre binaire.

▷ Exemple 2.1



Ces mémoires sont indissociables et l'ordre d'assemblage donne le poids de chaque bit.

II.1 Représentation des nombres entiers positifs

Les nombres sont représentés en binaire sur n bits : $n =$ nombre d'unités mémoires ($n = 8, 16, 32, 64, \dots$)

On peut représenter des nombres allant de 0 à $2^n - 1$.

II.2 Représentation binaire des entiers signés

Traditionnellement on met un signe « - » pour représenter les nombres négatifs. Mais les systèmes logiques ne permettent de présenter qu'un des deux symboles « 0 » et « 1 », il faut chercher une convention pour remplacer le « - ».

II.2.1 Représentation module et signe

Solution la plus simple : on ajoute un e.b. à gauche du module pour le signe. Ainsi, un nombre commençant par un « 0 » sera positif, alors qu'un nombre commençant par un « 1 » sera négatif : $\begin{cases} 0 \leftrightarrow + \\ 1 \leftrightarrow - \end{cases}$

▷ **Exemple 2.2**

avec 4 e.b. Les valeurs vont de -7 à 7

Signe	Module	Valeur	Signe	Module	Valeur
1	111	-7	0	111	7
1	110	-6	0	110	6
1	101	-5	0	101	5
1	100	-4	0	100	4
1	011	-3	0	011	3
1	010	-2	0	010	2
1	001	-1	0	001	1
1	000	0	0	000	0

Problème : on a ici deux représentations différentes pour le zéro : '00...0' et '10...0'.

II.2.2 Représentation en complément restreint (complément à 1)

$-A \bar{A}$: pour prendre l'inverse d'un nombre, il suffit de le complémenter (inversion de tous ses bits). Comme dans le cas précédent, la nature du premier bit donnera le signe :

$$\begin{cases} 0 \leftrightarrow + \\ 1 \leftrightarrow - \end{cases}$$

► Exemple 2.3

$$\text{avec 4 e.b. : } \begin{cases} 5 & \rightarrow & 0101 \\ -5 & \rightarrow & 1010 \end{cases}$$

Problème : de nouveau, on a deux représentations différentes pour le zéro.

II.2.3 Représentation en complément vrai (complément à 2)

C'est la représentation la plus utilisée. Le bit le plus à gauche est encore le bit de

$$\text{signe : } \begin{cases} 0 \leftrightarrow + \\ 1 \leftrightarrow - \end{cases}$$

$$\begin{array}{r} -A \quad \bar{A} \quad 1 \\ \bar{A} \quad \bar{a}_{n-1} \quad \bar{a}_{n-2} \quad \dots \quad \bar{a}_0^1 \\ A \quad \bar{A} \quad 1 \quad 1 \quad \dots \quad 1 \\ 1 \quad \bar{A} \quad A \quad \underbrace{1 \leftarrow 00 \dots 0^2}_0 \end{array}$$

$\Rightarrow -A \bar{A} 1$ est appelé complément à 2

► Remarque 2.1

- pour passer d'une valeur négative à une valeur positive, on applique aussi le complément à 2 ;
- une seule représentation pour le zéro ;
- avec des mots de n e.b., on obtient 2^n valeurs différentes, de 0 à $2^{n-1} - 1$ pour les valeurs positives, et de -1 à -2^{n-1} pour les valeurs négatives ;

¹on complémente chaque coefficient

²car on représente sur n bits seulement

▷ **Exemple 2.4**

$$n \text{ sur } 8 \Rightarrow \begin{cases} \text{nb 0 de 0 à } 127 \\ \text{nb 0 de } -1 \text{ à } -128 \end{cases}$$

- $nb \geq 0 \rightarrow$ bit de signe 0 $nb < 0 \rightarrow$ bit de signe 1
- pour représenter un nombre positif sur une mémoire de taille donnée, on complète les cases vides de gauche par des 0; pour représenter un nombre négatif sur une mémoire de taille donnée, on complète les cases vides de gauche par des 1;

▷ **Exemple 2.5**

+13 sur 8 bits : 00001101, -13 sur 8 bits : 11110011

II.2.4 Représentation en code relatif à 2^{n-1}

Les nombres x sont représentés par $2^{n-1} x$.

On constate ici que le bit de signe est inversé par rapport aux représentations précédentes : ce code est en fait identique au codage en complément à 2 avec le bit de signe complété.

On calcule l'inverse d'un nombre en relatif à 2^{n-1} comme en complément à 2 en complétant le nombre puis en ajoutant 1.

II.3 Représentation des nombres réels dans un calculateur

Dans un calculateur, un nombre est toujours écrit sous forme d'un bloc de n e.b. (considéré comme un entier N).

Pour représenter les nombres fractionnaires il est nécessaire de définir la position de la virgule : pour ce faire, il existe deux méthodes.

II.3.1 La représentation en virgule fixe

On décide que la virgule est toujours à une position donnée (un entier peut être représentatif d'un nombre fractionnaire si on connaît la place de la virgule).

▷ **Exemple 2.6**

Virgule au rang K (K chiffres après la virgule) :

La valeur N écrite en mémoire aura les poids suivants :

$$N \ 2^{N-1-K} \dots 2^0 \ 2^{-1} \ 2^{-K} \qquad 0 \leq N \leq (2^n - 1)2^{-K}$$

Virgule au rang 0 :

$$N \ 2^{N-1} \dots 2^0 \qquad 0 \leq N \leq 2^N - 1$$

Inconvénient de la méthode :

- problème de gestion de la virgule notamment dans les multiplications (pour les additions et soustractions pas de problème, la position de la virgule ne change pas) ;

▷ **Exemple 2.7**

Si on décide 2 symboles pour les parties entières et 2 symboles pour les parties fractionnaires, on ne peut plus écrire 256,1.

- utilisation limitée lorsqu'on traite des données de grandeurs différentes, car on doit prendre un grand nombre de bit de part et d'autre de la virgule pour pouvoir représenter des grandeurs très faibles et des grandeurs très importantes.

II.3.2 La représentation en virgule flottante simplifiée

II.3.2.a Introduction [WWW01]

Il arrive dans de nombreux domaines que l'intervalle des valeurs numériques pertinentes soit particulièrement étendu. L'astronomie en est un exemple extrême puisque certains calculs peuvent faire intervenir simultanément la masse du soleil (2.10^{30} kg) et la masse de l'électron (9.10^{-31} kg). Ces deux nombres diffèrent de plus de 60 ordres de grandeur (10^{60}) !

Des calculs faisant intervenir ces nombres pourraient s'effectuer en précision multiple, avec par exemple des nombres de 62 digits. Tous les opérandes et tous les résultats seraient représentés par des nombres de 62 digits. Cependant, la masse du soleil n'est connue qu'avec une précision de 5 digits, et il n'y a en physique pratiquement aucune mesure que l'on puisse réaliser avec une précision de 62 digits. Une solution serait alors d'effectuer les calculs avec une précision de 62 digits et de laisser tomber 50 ou 60 d'entre eux avant d'annoncer les résultats, mais ceci est coûteux à la fois en espace mémoire et en temps de calcul.

En fait, ce qu'il faut est un système permettant de représenter des nombres, tel que la taille de l'intervalle des nombres "exprimables" soit indépendante du nombre de digits significatifs.

II.3.2.b Principe de la représentation en virgule flottante

Le nombre N est représenté sous la forme :

exposant	mantisse
----------	----------

1^{ère} approche

Soit $N = a_3 a_2 a_1 a_0, a_{-1} a_{-2} a_{-3}$: N peut se noter :

$$\underbrace{(a_6 a_5 a_4 a_3 a_2 a_1 a_0)}_{\text{mantisse}} \cdot \underbrace{2^{-3}}_{\text{exp}}$$

$$\Rightarrow \begin{cases} \text{exposant} = & -3 \\ \text{mantisse} = & a_6 a_5 a_4 a_3 a_2 a_1 a_0 \end{cases}$$

Les valeurs de la mantisse et l'exposant seront notés en complément à 2 en mémoire du calculateur

▷ **Exemple 2.8**

Soit la mémoire de taille suivante :

4 bits	12 bits
exposant	mantisse

Coder la valeur 26,75 en virgule flottante.

$$(26,75)_{10} \quad (11010,110)_2$$

$$(11010,11)_2 \quad (11010110) \cdot 2^{-3}$$

$$\Rightarrow \begin{cases} \text{exposant} = & -3 \\ \text{mantisse} = & 11010110 \end{cases}$$

1101	0000011010110
exp-3	mantisse214

$$26,75 \quad 214 \cdot 2^{-3}$$

2^{ème} approche

Méthode inverse → on considère que le bit le plus à gauche de la mantisse a pour poids 2^{-1} .

Soit $N = a_3 a_2 a_1 a_0, a_{-1} a_{-2} a_{-3}$

$$N \text{ peut aussi se noter } (0, \underbrace{a_{-1} a_{-2} a_{-3} a_{-4} a_{-5} a_{-6} a_{-7}}_{\text{mantisse}}) \cdot \underbrace{2^4}_{\text{exp}}$$

▷ **Exemple 2.9**

Même exemple que précédemment :

$$(26,75)_{10} (11010,110)_2 \longrightarrow (0,11010110).2^5$$

0101	110101100000
------	--------------

► **Remarque 2.2**

Les ordinateurs utilisent cette représentation avec 32 bits pour la mantisse et 8 bits pour l'exposant. En général, on utilise la représentation inverse, avec le bit le plus à gauche = 1, soit une mantisse normalisée $\Rightarrow 0,5 \leq M < 1$

II.3.3 La représentation IEEE 754 [WWW01]

II.3.3.a Présentation

Le standard IEEE 754 définit trois formats : les nombres en simple précision sur 32 bits, les nombres en double précision sur 64 bits, et les nombres en représentation intermédiaire sur 80 bits. La représentation sur 80 bits est principalement utilisée en interne par les processeurs pour minimiser les erreurs d'arrondi.

Un nombre N de 32 bits est représenté sous la forme :

s	exposant	mantisse
---	----------	----------

où le signe « s » est codé sur 1 bit, l'exposant est codé sur 8 bits en code relatif à 127 (cf. §II.2.4 page 16), et la mantisse sur 23 bits.

Un nombre de 64 bits (double précision) utilise la même représentation à ceci près que la taille de l'exposant est portée à 11 bits en code relatif à 1023, et celle de la mantisse à 52 bits.

Une mantisse normalisée commence toujours par un bit 1, suivi par la virgule, puis par le reste de la mantisse. Le bit initial, toujours présent et toujours à 1 dans une mantisse normalisée est implicite et non représenté. La valeur de la mantisse est appelée « significande » ; le significande a donc une valeur implicite $1 \leq x < 2$.

► **Exemple 2.10**

$$- 1 \cdot 2^0 \times (1 \cdot 0)$$

Le bit de signe sera 0, l'exposant, en code relatif à 127 sera représenté par 127 = 01111111, et le significande vaut 1, ce qui résulte en une mantisse dont tous les bits sont à 0. La représentation IEEE simple precision IEEE 754 du nombre 1 est donc :

$$\text{Code}(1) = \begin{array}{|c|c|c|} \hline 0 & 01111111 & 0000\dots0 \\ \hline s & e & m \\ \hline \end{array} = 3F800000$$

$$- 0,5 \cdot 2^{-1} \times (1 \cdot 0)$$

Le bit de signe est 0, l'exposant, en code relatif à 127 est représenté par 127 - 1 = 01111110, et le significande vaut 1, ce qui résulte en une mantisse

dont tous les bits sont à 0. La représentation IEEE simple précision IEEE 754 du nombre 0.5 est donc :

$$\text{Code}(0.5) = \begin{array}{|c|c|c|} \hline 0 & 01111110 & 0000\dots0 \\ \hline s & e & m \\ \hline \end{array} = 3F000000 \\ - 1.5 \cdot 2^0 \times (1 \cdot 2^{-1})$$

Le bit de signe est 0, l'exposant, en code relatif à 127 est représenté par 127 = 01111111, et le significande vaut 1.1, ce qui résulte en une mantisse dont le premier bit est à 1 et les 22 suivants à 0. La représentation IEEE simple précision IEEE 754 du nombre 1.5 est donc :

$$\text{Code}(1.5) = \begin{array}{|c|c|c|} \hline 0 & 01111111 & 1000\dots0 \\ \hline s & e & m \\ \hline \end{array} = 3FC00000$$

II.3.3.b Nombres spéciaux

En arithmétique à virgule flottante on peut obtenir un résultat valable, ou alors rencontrer un problème de dépassement par valeur supérieure (*overflow*) lorsque le résultat est trop grand pour pouvoir être représenté, ou par valeur inférieure (*underflow*) lorsque le résultat est trop petit.

Dépassement par valeur inférieure

Cette situation arrive lorsqu'un résultat est trop petit pour pouvoir être représenté. Le standard IEEE 754 résout partiellement le problème en autorisant dans ce cas une représentation dénormalisée. Une représentation dénormalisée est caractérisée par le fait d'avoir un code d'exposant complètement nul, ce qui est interprété comme une indication du fait que le bit de poids fort de la mantisse, implicite, est cette fois à 0 au lieu d'être à 1. De cette façon, le plus petit nombre « exprimable » est : $2^{-127} \times 2^{-23} = 2^{-150} \sim 10^{-45}$.

Cependant, il faut remarquer que plus le nombre représenté est petit, moins sa mantisse comportera de bits significatifs. Ce schéma permet une approche « douce » du phénomène de dépassement par valeur inférieure, en sacrifiant la précision lorsqu'un résultat est trop petit pour admettre une représentation normalisée.

Zéro

Zéro est représenté sous la forme d'un nombre dénormalisé. Ceci résulte en deux représentations possibles pour zéro : l'une pour 0, l'autre pour -0. Ces représentations sont caractérisées par un bit de signe suivi par 31 zéros.

Dépassement par valeurs supérieures

Le dépassement par valeurs supérieures ne peut pas être traité comme le dépassement par valeurs inférieures, et est indiqué par un code d'exposant dont tous les bits sont à 1, suivi par une mantisse dont tous les bits sont à 0. Ceci est interprété comme représentant l'infini. L'infini peut être positif ou négatif, en fonction de la valeur du bit de signe. L'infini peut être utilisé dans les calculs et les résultats correspondent au sens commun : $\infty \infty \infty$; $x/\infty = 0$; $x/0 = \infty$.

Not a Number (NaN)

Cependant, certaines opérations peuvent ne conduire à aucun résultat exprimable, comme ∞/∞ ? ou $0 \times \infty$?.

Le résultat de telles opérations est alors indiqué par un autre code spécial : le code d'exposant a tous les bits à 1, suivi par une mantisse non nulle. Le « nombre » correspondant est appelé NaN (*Not a Number*) : c'est un non-nombre.

II.3.3.c Résumé

Nombre	Signe	Exposant	Mantisse
nombre normalisé	0/1	01 à FE	quelconque
nombre dénormalisé	0/1	00	quelconque
zéro	0/1	00	0
∞	0/1	FF	0
NaN	0/1	FF	tout sauf 0

IEEE 754	Simple précision	Double précision
exposant	-126 à 127	-10^{22} à 10^{23}
mantisse	1 à $2 - 2^{-23}$	1 à $2 - 2^{-52}$
+ pt # normalisé	2^{-126}	2^{-1022}
+ gd # normalisé	presque 2^{128}	presque 2^{1024}
intervalle utile	$\approx 10^{-38}$ à 10^{38}	$\approx 10^{-308}$ à 10^{308}
+ pt # dénormalisé	$2^{-150} \approx 10^{-45}$	$2^{-1074} \approx 10^{-324}$

II.4 Arithmétique binaire**II.4.1 Addition**

L'addition en binaire se fait avec les mêmes règles qu'en décimal : on commence par additionner les bits de poids faibles ; on a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande (dans le cas du binaire : 1) ; cette retenue est reportée sur le bit de poids plus fort suivant.

La table d'addition binaire est la suivante :

A	B	C	retenue	(carry)	
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

▷ **Exemple 2.11**

Addition des nombres de 4 bits :

$$\begin{array}{r|cccc|c}
 & 0 & 0 & 1 & 1 & 3 \\
 + & 1 & 0 & 1 & 0 & -6 \\
 \hline
 = & 1 & 1 & 0 & 1 & -3
 \end{array}$$

$$\begin{array}{r|cccc|c|c|c}
 & 0 & 1 & 1 & 1 & , & 1 & 1 & 7,75 \\
 + & 0 & 1 & 0 & 1 & , & 0 & 1 & 5,25 \\
 \hline
 = & \mathbf{1} & 1 & 0 & 1 & , & 0 & 0 & -3,00
 \end{array}$$

La retenue de la deuxième opération indique un dépassement de capacité (*overflow*) : le bit de signe est à 1 alors qu'il aurait dû être à 0 (addition de deux nombres positifs).

Conditions de dépassement lors de l'addition de deux nombres A et B de 16 bits :

a_{15}	b_{15}	r_{15}	opérandes	résultat	R	D
0	0	0	$a\ 0\ b\ 0$	$r\ 0$	0	non
0	0	1	$a\ 0\ b\ 0$	$r\ 0$	0	oui
0	1	0	$a\ 0\ b\ 0$	$r\ 0$	1	non
0	1	1	$a\ 0\ b\ 0$	$r\ 0$	0	non
1	0	0	$a\ 0\ b\ 0$	$r\ 0$	1	non
1	0	1	$a\ 0\ b\ 0$	$r\ 0$	0	non
1	1	0	$a\ 0\ b\ 0$	$r\ 0$	1	oui
1	1	1	$a\ 0\ b\ 0$	$r\ 0$	1	non

R : retenue ; D : dépassement

Ce tableau nous permet de déterminer la condition de dépassement (OF : *overflow flag*) : OF $\overline{a_{15}} \cdot \overline{b_{15}} \cdot r_{15} \vee a_{15} \cdot b_{15} \cdot \overline{r_{15}}$.

- si OF est à 0, le bit de poids fort (r_{15}) donne le signe du résultat dont la valeur est disponible sur les 15 bits de poids faible.
- si OF est à 1, l'indicateur de retenue (C) donne le signe du résultat qui est lui-même sur 16 bits. Dans ce dernier cas, le bit de poids fort ne donne pas le signe du résultat !

II.4.2 Soustraction

Dans la soustraction binaire, on procède comme en décimal. Quand la quantité à soustraire est supérieure à la quantité dont on soustrait, on emprunte 1 au voisin de gauche. En binaire, ce 1 ajoute 2 à la quantité dont on soustrait, tandis qu'en décimal il ajoute 10.

La table de soustraction binaire est la suivante :

A	B	C	retenue	(borrow)	
0	-	0	=	0	0
0	-	1	=	1	1
1	-	0	=	1	0
1	-	1	=	0	0

▷ **Exemple 2.12**

0	←	1 0 1 , 0	-	5
		0 1 1 , 1		3,5
		0 0 1 , 1		1,5
1	←	0 0 0 1 1	-	3
		0 1 1 0 0		12
		1 0 1 1 1		-9

► **Remarque 2.3**

On peut utiliser le complément à 2 de la valeur à soustraire puis on additionne. Dans ce cas, il faut compléter la retenue (carry) pour obtenir la retenue soustractive (borrow). Cela se passe de cette manière dans certains calculateurs.

▷ **Exemple 2.13**

7 - 2 :						
7 =	0	0	1	1	1	
2 =	0	0	0	1	0	
-2 =	1	1	1	1	0	1 ←
						0 0 1 1 1
						1 1 1 1 0
						0 0 1 0 1

On ne tient pas compte de la retenue.

II.4.3 Multiplication

La table de multiplication en binaire est très simple :

A	x	B	=	C
0	x	0	=	0
0	x	1	=	0
1	x	0	=	0
1	x	1	=	1

La multiplication se fait en formant un produit partiel pour chaque digit du multiplieur (seul les bits non nuls donneront un résultat non nul). Lorsque le bit du multiplieur

est nul, le produit partiel est nul, lorsqu'il vaut un, le produit partiel est constitué du multiplicande décalé du nombre de positions égal au poids du bit du multiplicateur.

▷ **Exemple 2.14**

$$\begin{array}{r}
 \times \quad \begin{array}{cccc} 0 & 1 & 0 & 1 \end{array} \text{ multiplicande} \\
 \hline
 \begin{array}{cccc} 0 & 0 & 1 & 0 \end{array} \text{ multiplicateur} \\
 \hline
 \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 \end{array} \\
 \hline
 \end{array}
 \quad \times \quad \begin{array}{r} 5 \\ 2 \\ \hline 10 \end{array}$$

► **Remarque 2.4**

La multiplication binaire par 2^N , se résume à un décalage de N bits vers la gauche. On introduira donc à droite N zéro.

▷ **Exemple 2.15**

8×4 sur 8 bits :

$$\boxed{0 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0 \mid 0 \mid 0}$$

$$\Rightarrow \boxed{0 \mid 0 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0} \leftarrow 0$$

-16×4 sur 8 bits :

$$\boxed{1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0}$$

$$\Rightarrow \boxed{1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0} \leftarrow 0$$

II.4.4 Division

La table de division binaire est la suivante :

A	/	B	=	C
0	/	0	=	impossible
0	/	1	=	0
1	/	0	=	impossible
1	/	1	=	1

La division binaire s'effectue à l'aide de soustractions et de décalages, comme la division décimale, sauf que les digits du quotient ne peuvent être que 1 ou 0. Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0.

▷ **Exemple 2.16**

Division du nombre $(10010000111)_2$ par $(1011)_2$ ($(1101001)_2$ reste $(100)_2$).

c'est-à-dire $1159/11$ 105, reste 4.

$$\begin{array}{r}
 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \mid 1\ 0\ 1\ 1 \\
 -\ 1\ 0\ 1\ 1 \\
 \hline
 0\ 1\ 1\ 1\ 0 \\
 -\ 1\ 0\ 1\ 1 \\
 \hline
 0\ 0\ 1\ 1\ 0\ 0 \\
 -\ 1\ 0\ 1\ 1 \\
 \hline
 0\ 0\ 0\ 1\ 1\ 1\ 1 \\
 -\ 1\ 0\ 1\ 1 \\
 \hline
 0\ 1\ 0\ 0\ 0
 \end{array}$$

► **Remarque 2.5**

La division binaire par 2^N , se résume à un décalage de N bits vers la droite. En arithmétique signée, il faut penser à recopier à gauche le bit de signe autant de fois que nécessaire.

▷ **Exemple 2.17**

$8/4$ sur 8 bits :

$$0 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0 \mid 0 \mid 0$$

$$\Rightarrow 0 \rightarrow 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0$$

$-16/4$ sur 8 bits :

$$1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0$$

$$\Rightarrow 1 \rightarrow 1 \mid 1 \mid 1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0$$

II.5 En résumé

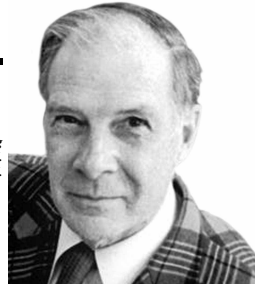
- La valeur d'un nombre est indépendant de la base dans laquelle il est noté.

- Un nombre binaire peut avoir plusieurs valeurs différentes selon le système de représentation. Soit le nombre binaire $a_n a_{n-1} \dots a_1 a_0$. Ce nombre vaut :
 - ☞ $a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2 + a_0$
en représentation non signée
 - ☞ $-a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2 + a_0$
en représentation signée complément à 2
 - ☞ $1 - a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2 + a_0$
en représentation signée complément à 1
 - ☞ $-1^{a_n} \times (a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2 + a_0)$
en représentation module et signe
- Les opérations arithmétiques obéissent en binaire aux mêmes règles qu'en décimal, il suffit juste de se rappeler que la base de numération est 2 et non plus 10.

Chapitre III

Les codes numériques

Richard Wesley Hamming
★ 11 fév. 1915 à Chicago, E.-U.
† 7 jan. 1998 à Monterey, E.-U.



« Indeed, one of my major complaints about the computer field is that whereas Newton could say, “If I have seen a little farther than others, it is because I have stood on the shoulders of giants,” I am forced to say, “Today we stand on each other’s feet.” Perhaps the central problem we face in all of computer science is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way. Science is supposed to be cumulative, not almost endless duplication of the same kind of things. »
(Richard W. Hamming, One Man’s View of Computer Science, 1968, Turing Award Lecture)

Codage : opération qui établit une correspondance entre un ensemble source (nombre, caractère, symbole) vers un ensemble but contenant des combinaisons de 0 et de 1.

III.1 Codes numériques pondérés

III.1.1 Code binaire pur

→ code pondéré par des puissances de 2. Utilisé en arithmétique binaire. Ses dérivées sont le code octal et le code hexadécimal.

III.1.2 Code DCB (Décimal Codé Binaire)

→ chaque chiffre décimal (0,1,...,9) est codé en binaire avec 4 e.b. Code pondéré avec les poids 1,2,4,8,10,20,40,80,100,... Plus facile pour coder des grands nombre, il est surtout utilisé pour l'affichage des nombres.

► **Remarque 3.1**

Ne pas confondre DCB et code binaire pur : quand on code selon le code binaire pur on prend le nombre dans son intégralité et on le convertit ; par contre, quand on code en DCB on code chaque chiffre indépendamment les uns des autres.

▷ **Exemple 3.1**

$$(137)_{10} \quad (010001001)_2 \\ (001011111)_{DCB}$$

III.1.2.a Addition en DCB

L'addition de deux nombres codés en DCB ne pose pas de problème tant que le résultat est inférieur ou égal à 9 :

0000 0010	02
0000 0101	05
0000 0111	07

Par contre, dès que le résultat est supérieur à 9, il faut apporter une correction en additionnant 6, de manière à obtenir une réponse valide :

0000 0110	06	
0000 0100	04	
0000 1010	0?	erreur !
0000 0110	06	
0001 0000	10	

La correction est ici évidente, puisque la valeur obtenue est invalide en codage DCB. L'exemple suivant est moins évident :

0000 1001	09	
0000 1000	08	
0001 0001	11	erreur !
0000 0110	06	
0001 0111	17	

Dans ce dernier exemple, la correction est due au fait qu'il a eu débordement sur la 4 bits de poids faible du mot DCB : il faut donc apporter une correction sur ces 4 bits de poids faible.

►Note 3.1

- lorsque le résultat de l'addition est inférieur à 9, on ne change pas le résultat ;
- lorsque le résultat de l'addition est supérieur à 9, on ajoute 6 au résultat pour obtenir la valeur exacte ;
- lorsqu'il y a une retenue auxiliaire (ou décimale) (auxiliary ou decimal carry), on ajoute également 6 au résultat obtenu, même si la valeur est inférieure à 9.

Les propriétés énoncées ci-dessus pour les chiffres des unités sont évidemment variables pour les dizaines, les centaines, etc. La correction à apporter sera alors – selon les circonstances – 6, 60, 66, etc.

III.1.2.b Soustraction en DCB

La soustraction en DCB se comporte exactement comme l'addition, au signe près.

►Note 3.2

- lorsque le résultat de la soustraction est inférieur à 9, on ne change pas le résultat ;
- lorsque le résultat de la soustraction est supérieur à 9, on soustrait 6 au résultat pour obtenir la valeur exacte ;
- lorsqu'il y a une retenue soustractive (borrow), on soustrait également 6 au résultat obtenu, même si la valeur est inférieure à 9.

III.1.3 Code binaire de Aiken

Pondéré par 2421, c'est un code autoccomplémentaire. (les représentations de 2 chiffres dont la somme est 9 sont complémentaires l'une de l'autre.

Il peut être constitué par les règles suivantes :

- de 0 à 4 on code en binaire pur ;
- de 5 à 9 on ajoute 6 et on code en binaire pur. (c.à.d. $5 \rightarrow 5\ 6\ 11, 6 \rightarrow 6\ 6\ 12, \dots$)

▷ **Exemple 3.2**

décimal	Aiken			
	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0

décimal	Aiken			
	2	4	2	1
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

Ce code est utilisé dans certains calculateurs pour effectuer des soustractions par additions de la forme complémentaire.

III.1.4 Les codes biquinaires

C'est un code composé d'un groupe de n bits (en général 5) dont un seul parmi n progresse à la fois, et d'un groupe de m bits (1 à 2) assurant la distinction entre $n - 5$ et $n \geq 5$.

▷ **Exemple 3.3**

décimal	S	O	4	3	2	1	0
0	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0
2	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	1	0	0
8	1	0	0	1	0	0	0
9	1	0	1	0	0	0	0

Chaque combinaison a un nombre pair de 1 : sécurité de transmission.

Ce code est utilisé dans les calculatrices.

III.2 Codes numériques non pondérés

III.2.1 Code majoré de trois (*excédant de neuf*)

On prend chaque chiffre décimal +3, puis on convertit en binaire. On a parfois recours à ce code en raison de la facilité avec laquelle on peut faire certains calculs arithmétiques. La valeur d'un mot en code majoré de trois est en fait égale au code DCB auquel on a ajouté 3.

▷ **Exemple 3.4**

$$\begin{array}{r} (48)_{10} \quad 4 \quad 8 \\ \quad \quad \quad 3 \quad 3 \\ \hline \quad \quad \quad 7 \quad 11 \\ \quad \quad \quad \downarrow \quad \downarrow \\ \quad \quad \quad 0111 \quad 1011 \end{array}$$

III.2.2 Code de Gray (*binaire réfléchi*)

Un seul bit change entre deux nombres consécutifs (notion d'adjacence).

Ce code est utilisé dans les tableaux de Karnaugh (*cf.* section [V.1.1.c](#) page 61, dans des circuits d'entrée/sortie, et dans certains convertisseurs analogique/numérique.

Il ne convient pas pour l'arithmétique binaire.

▷ **Exemple 3.5**

Décimal	Gray
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0
16	0 0 0 0

Le code présente 4 symétries miroir. Il est cyclique : il se referme sur lui-même.

Pour convertir un nombre en code binaire naturel (CBN) vers un nombre en code binaire réfléchi (CBR), il faut ajouter le CBN trouvé à lui-même décalé d'un rang vers la gauche, sans tenir compte de l'éventuelle retenue et en abandonnant dans le résultat le bit de poids faible.

▷ **Exemple 3.6**

Soit le nombre décimal 87 ; sa valeur binaire est 1010111. Donc :

$$\begin{array}{r}
 1010111 \\
 +10101110 \\
 \hline
 11111001
 \end{array}$$

L'équivalent en code binaire réfléchi de $(87)_{10}$ est 1111100

III.2.2.a Conversion du code binaire naturel vers binaire réfléchi

	Binaire naturel				Binaire réfléchi			
	n_3	n_2	n_1	n_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Les équation logiques pour un mot de 4 bits sont :

$$\Leftrightarrow g_0 = a_1 \oplus a_0$$

$$\Leftrightarrow g_1 = a_2 \oplus a_1$$

$$\Leftrightarrow g_2 = a_3 \oplus a_2$$

$$\Leftrightarrow g_3 = a_3$$

Pour un mot binaire de format n on a donc :

$$\Leftrightarrow g_i = a_{i+1} \oplus a_i, \text{ pour } n-2 \geq i \geq 0$$

$$\Leftrightarrow g_{n-1} = a_{n-1}$$

On peut également exprimer g_n de manière récursive :

$$\Leftrightarrow g_0 = g_3 \oplus g_2 \oplus g_1 \oplus a_0$$

$$\Leftrightarrow g_1 = g_3 \oplus g_2 \oplus g_1$$

$$\Leftrightarrow g_2 = g_3 \oplus g_2$$

$$\Leftrightarrow g_3 = a_3$$

III.2.2.b Conversion du code binaire naturel vers binaire réfléchi

	Binaire réfléchi				Binaire naturel			
	g_3	g_2	g_1	g_0	n_3	n_2	n_1	n_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Les équation logiques pour un mot de 4 bits sont :

$$\Leftrightarrow a_3 \ g_3$$

$$\Leftrightarrow a_2 \ g_3 \oplus g_2$$

$$\Leftrightarrow a_1 \ g_3 \oplus g_2 \oplus g_1$$

$$\Leftrightarrow a_0 \ g_3 \oplus g_2 \oplus g_1 \oplus g_0$$

Pour un mot binaire de format n on a donc :

$$\Leftrightarrow a_{n-1} \ g_{n-1}$$

$$\Leftrightarrow a_i \oplus \sum_{j=1}^{n-1} g_i \oplus a_{i1} \oplus g_i, \text{ pour } n-2 \geq i \geq 0$$

III.3 Codes détecteurs d'erreurs et autocorrecteurs

Ces codes sont utilisés pour contrôler la transmission des données.

Souvent, on utilise un nombre de bits supérieur à celui strictement nécessaire pour coder l'information elle-même.

III.3.1 Codes biquinaires

Cf. III.1.4 page 30.

III.3.2 Les codes p parmi n

Ce sont des codes autovérificateurs (détecteurs d'erreurs mais pas autocorrecteurs). Ces codes possèdent n e.b. dont p sont à 1 ; la position des « 1 » permet de reconnaître un élément codé. Le nombre de combinaisons répondant à cette définition est :

$$C_n^p = \frac{n!}{p!(n-p)!}$$

▷ **Exemple 3.7**

Pour transmettre l'information numérique dans les centraux téléphoniques (*cross bar*), on utilise un code 2 parmi 5 (ou code 74210) pour représenter les chiffres décimaux.

Il existe 10 combinaisons :

Déc.	2 parmi 5				
	7	4	2	1	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0

Déc.	2 parmi 5				
	7	4	2	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

III.3.3 Les codes à contrôle de parité

Dans ces codes, on ajoute un e.b. de sorte que l'ensemble des bits à transmettre (ou le mot) ait un nombre pair (parité paire) ou impaire (parité impaire) de « 1 ».

▷ **Exemple 3.8**

0101 → 00101

0111 \rightarrow 10111

► **Remarque 3.2**

Dans l'application de la méthode de la parité, l'émetteur et le récepteur se mettent d'accord à l'avance sur la parité à surveiller (paire ou impaire).

► **Remarque 3.3**

Pour détecter la place d'un e.b. faux, il faut coder dans 2 dimensions selon les lignes et les colonnes.

► **Exemple 3.9**

0	1	0	0	1	Transmission	\rightarrow	0	1	0	0	1
1	0	0	1	0			1	0	0	0	0
0	0	0	1	1			0	0	0	1	1
1	1	1	0	1			1	1	1	0	1
0	0	1	0	1			0	0	1	0	1
										↑	

Ce code détecte les erreurs simples à condition que l'e.b. de parité ne soit pas erroné.

III.3.4 Code de Hamming

Ce code est utilisé dans les transmissions de données. Il localise et corrige les chiffres erronés (en ajoutant des e.b. supplémentaires aux e.b. de l'information).

Le nombre binaire d'information effective est : $N ABCD$ 4

Le nombre binaire d'information transmise est : $N abcdefg$ 7

avec	a	$A \oplus B \oplus C \oplus D$
	b	$A \oplus C \oplus D$
	c	A
	d	$B \oplus C \oplus D$
	e	B
	f	C
	g	D

III.4 Les codes alphanumériques

Ils servent à coder des chiffres, des lettres, des signes de ponctuations et des caractères spéciaux (26 caractères minuscules, 26 caractères majuscules, 7 signes, 20 à 40 caractères spéciaux comme +,.,≠,%,...)

III.4.1 Le code ASCII (American Standard Code for Information Interchange)

C'est le plus répandu. On le retrouve pratiquement dans tous les ordinateurs et leurs organes périphériques, pour leurs dialogues et la représentation des textes en mémoire.

Chaque symbole (caractère d'imprimerie) est codé par 7 e.b. (un 8^{ème} e.b. peut servir de parité) : 2^7 128 combinaisons différentes.

[Exercices sur les nombres]

► **Exercice 1.1**

Convertir en binaire, octal et hexadécimal les nombres décimaux suivants :
43 ; 154 ; 25740

► **Exercice 1.2**

Convertir en décimal et hexadécimal les nombres suivants :
(10010101)_{DCB} ; (1101110)₂ ; (75)₈ ; (587)₈

► **Exercice 1.3**

Convertir en binaire et hexadécimal les nombres suivants :
(166,25)₁₀ ; (126,34)₈ ; (231,1)₄

► **Exercice 1.4**

Convertir en binaire le nombre décimal suivant : 24537

► **Exercice 1.5**

Convertir en décimal les nombres suivants :
(10010101)_{DCB} ; (D9,4)_H ; (576)₈

► **Exercice 1.6**

Que peuvent représenter les octets suivants ?
01111001 ; 10100100 ; 01101010 ; 10010111

► **Exercice 1.7**

En parité impaire, quel est le bit de parité à associer aux octets suivants ?
EC ; F1 ; 69 ; A3

► **Exercice 1.8**

En parité paire, quel est le bit de parité à associer aux octets suivants ?
CD ; 6E ; B8 ; A4

► **Exercice 1.9**

On effectue les opérations suivantes sur des octets signés (représentation en complément à 2). Donner les résultats en discutant leur validité. Vérifier en prenant les équivalents décimaux.

$5F+6D$; $E8+C7$; $9A-17$; $5B-C4$; $A4-62$

► **Exercice 1.10**

Une mémoire contient des octets stockés entre les adresses $(9400)_H$ et $(B3FF)_H$. Combien d'octets contient-elle ? Quelle est la capacité totale en kbits ?

► **Exercice 1.11**

Une mémoire contient 2k octets stockés à partir de l'adresse $(700)_H$. Quelle est la dernière adresse ?

1 4 6 7 3 9 2 3



Deuxième partie

La logique combinatoire

2 2

0 1 2 3 4 5 6 7 8 9



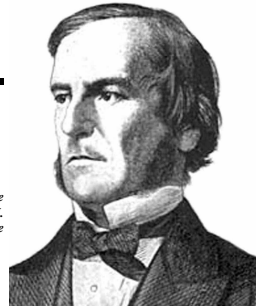
1 2 3 4 5 6 7 8 9 0



Chapitre IV

Algèbre booléenne et opérateurs logiques

George Boole
★ 2 nov. 1815, Lincoln, R.-U.
† 8 déc. 1864, Ballintemple, Irlande



« Une proposition peut être vraie ou fausse, mais ne peut pas être vraie et fausse. »
(Aristote ■ 384, † 322 av. J.-C.)

IV.1 Introduction

Les systèmes logiques fonctionnent en mode binaire \rightarrow les variables d'entrée et de sortie ne prennent que deux valeurs : « 0 » ou « 1 ». Ces valeurs (états) « 0 » et « 1 » correspondent à des plages définies à l'avance.

▷ **Exemple 4.1**

- Technologie électrique TTL :
 - « 1 » \leftrightarrow 2,4 à 5 V
 - « 0 » \leftrightarrow 0 à 0,8 V
- Technologie pneumatique :
 - « 1 » \leftrightarrow présence de pression
 - « 0 » \leftrightarrow absence de pression

Les valeurs « 0 » et « 1 » ne représentent pas des nombres réels mais plutôt l'état d'une variable (logique) \rightarrow on les appelle donc « niveaux logiques ».

IV.1.1 Convention de nommage des synonymes des « 0 » et « 1 » :

Ces deux valeurs peuvent être nommées de différentes façons :

- Niveau logique « 1 » : Vrai, Fermé, Marche, Haut, Allumé, Oui ;
- Niveau logique « 0 » : Faux, Ouvert, Arrêt, Bas, Éteint, Non.

IV.1.2 Types de logiques

On définit deux types de logiques :

- Logique positive :
 - niveau haut \rightarrow état logique « 1 » (5V)
 - niveau bas \rightarrow état logique « 0 » (0V)
- Logique négative :
 - niveau haut \rightarrow état logique « 0 » (0V)
 - niveau bas \rightarrow état logique « 1 » (5V)

La logique binaire basée sur l'algèbre de Boole permet de décrire dans un modèle mathématique les manipulations et traitement des informations binaires, et d'analyser les systèmes numériques.

Il existe 3 fonctions élémentaires dans l'algèbre de Boole :

- addition logique : appelée OU, symbolisée par un plus : « + » ;
- multiplication logique : appelée ET, symbolisée par un point : « . » ; $_$
- complémentation : appelée NON, symbolisée par un surlignement : « $\bar{\quad}$ »
 \rightarrow tout circuit numérique peut être défini à l'aide d'une fonction logique (expression logique) qui représente la variable de la sortie en fonction des variables d'entrée.

IV.1.3 Variables logiques (binaires)

Ce sont des variables ne pouvant prendre que deux valeurs distinctes : « 0 » ou « 1 ». Une variable binaire peut représenter n'importe quel dispositif binaire (contact, lampe, électro-vanne...)

IV.1.4 Convention :

Tout appareil est schématisé à l'état de repos. Dans tous les cas, l'action sur un appareil sera notée a, b, \dots et la non action \bar{a}, \bar{b}, \dots

▷ Exemple 4.2

Bouton poussoir \rightarrow contact repos et contact travail.

1^{er} cas : schéma d'un contact ouvert au repos dit « contact travail ».

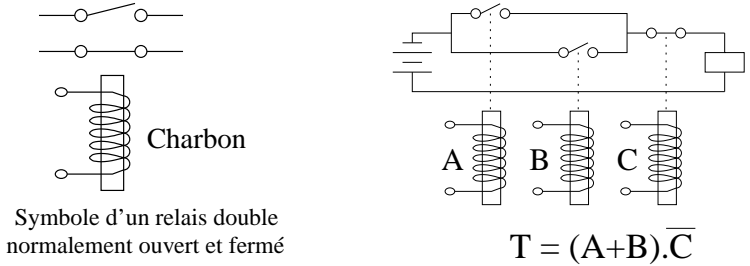
2^e cas : schéma d'un contact fermé au repos dit « contact repos ».

▷ Exemple 4.3

Relais : c'est un interrupteur opérant de façon électromagnétique ; lorsqu'un courant approprié passe dans le charbon, une force magnétique déplace les armatures imposant l'ouverture ou la fermeture des contacts. Il est présenté dans sa position non alimentée (au repos).



Ils peuvent être fermés ou ouverts au repos.



IV.2 Propriétés de l'algèbre booléenne

IV.2.1 Présentation

L'algèbre booléenne définit un cadre mathématique d'étude de propositions logiques portant sur des ensembles E d'éléments.

Définition 4.1

Algèbre booléenne : un ensemble E d'éléments (a, b, c, \dots) associé à deux opérations binaires $+$ et \cdot constitue une algèbre booléenne si et seulement si les postulats suivants sont satisfaits :

- ☞ **P1** Les opérations sont commutatives ;
- ☞ **P2** Chacune des opérations est distributive sur l'autre ;
- ☞ **P3** Il existe les éléments identité 0 et 1 respectivement pour $+$ et \cdot ;
- ☞ **P4** Pour chaque élément $a \in E$, il existe un élément $\bar{a} \in E$ tel que : $a \bar{a} = 0$ et $a \bar{a} = 1$.

À partir de ces postulats, il est possible de démontrer les théorèmes d'idempotence (cf. §IV.4.3), de l'élément nul, d'involution (cf. §IV.4.5), d'absorption (cf. §IV.5.6), d'associativité ainsi que la loi de De Morgan (cf. §IV.7). Tous ces théorèmes seront présentés plus loin.

Le lecteur attentif aura remarqué après la lecture des quatre postulats ci-dessus qu'il n'est jamais fait mention du nombre d'éléments dans l'ensemble E , ni encore moins que ce nombre d'éléments est limité à deux !

L'algèbre booléenne n'est pas restreinte aux ensembles binaires.

En fait, le nombre d'éléments dans E peut être infini, mais doit au moins comporter les éléments 0 et 1. Ainsi l'algèbre binaire, qui ne contient que les éléments 0 et 1, constitue l'algèbre booléenne la plus simple.

▷ **Exemple 4.4**

Algèbre booléenne portant sur 4 éléments : $E \{0, a, b, 1\}$

	0	a	b	1
0	0	a	b	1
a	a	a	1	1
b	b	1	b	1
1	1	1	1	1

.	0	a	b	1
0	0	0	0	0
a	0	a	0	a
b	0	0	b	b
1	0	a	b	1

IV.3 Algèbre binaire ou algèbre de commutation

IV.3.1 Postulats de base

Le domaine de définition B_2 de l'algèbre de commutation comprend donc deux éléments 0 et 1 ($B_2 \{0, 1\}$).

Si a est une variable logique on a :

⇨ **P1** $a \cdot 0$ si et seulement si $a \neq 1$

⇨ **P1*** $a \cdot 1$ si et seulement si $a \neq 0$

L'opération NON(ou complément), notée « $\bar{}$ » est définie par :

⇨ **P2** $0 \bar{1}$

⇨ **P2*** $1 \bar{0}$

L'opération OU(ou disjonction), notée « $+$ » est définie par :

⇨ **P3** $1 \cdot 1 \cdot 1 \cdot 0 \cdot 0 \cdot 1 \cdot 1$

⇨ **P4** $0 \cdot 0 \cdot 0$

L'opération ET(ou intersection), notée « \cdot » est définie par :

⇨ **P3*** $0 \cdot 0 \cdot 0 \cdot 1 \cdot 1 \cdot 0 \cdot 0$

⇨ **P4*** $1 \cdot 1 \cdot 1$

L'algèbre de commutation est le système algébrique constitué de l'ensemble $\{0, 1\}$ et des opérateurs ET, OU, NON.

À partir de ces quatre postulats, on peut construire les différents théorèmes présentés dans les sections §IV.4 page suivante et §IV.5 page 48.

IV.3.2 Hiérarchie des opérations

Dans une expression sans parenthèses, on effectue d'abord les opérations ET et, par la suite, les OU.

IV.3.3 Induction parfaite

Dans le domaine linéaire, il n'est pas possible de prouver une équation en la vérifiant pour toutes les valeurs des variables.

En logique binaire, puisque les variables sont limitées à deux états, on peut prouver une relation en la vérifiant pour toutes les combinaisons de valeurs pour les variables d'entrée. Ainsi, toutes les propriétés présentés dans les sections §IV.4 de la présente page et §IV.5 page suivante peuvent être démontrées par induction parfaite.

On notera qu'il n'est pas évident de démontrer ces relations par induction parfaite en algèbre booléenne de plus de deux variables. La preuve de de ces théorèmes peut être consultée notamment dans [WHI61].

IV.4 Théorèmes monovariabiles

IV.4.1 Identité

À chaque opérateur correspond un élément neutre qui, lorsqu'il est opéré avec une variable quelconque A , donne un résultat identique à cette variable.

$$A \ 0 \ A \quad A \ 1 \ A$$

IV.4.2 Élément nul

À chaque opérateur correspond un élément nul qui, lorsqu'il est opéré avec une variable quelconque A , donne un résultat identique à cet élément nul.

$$A \ 1 \ 1 \quad A \ 0 \ 0$$

IV.4.3 Idempotence

Le résultat d'une opération entre une variable A et elle-même est égal à cette variable.

$$A \ A \ A \quad A \cdot A \ A$$

IV.4.4 Complémentation

$$A \ \bar{A} \ 1 \quad A \cdot \bar{A} \ 0$$

IV.4.5 Involution

Le complément du complément d'une variable A est égal à cette variable.

$$\overline{\bar{A}} \ A$$

IV.5 Théorèmes multivariables

IV.5.1 Équivalence

Deux fonctions sont équivalentes si on peut leur faire correspondre la même table de vérité.

Si $F \overline{A.B}$ et $G \overline{A \overline{B}}$, alors $F \equiv G$, et on dit que F est équivalente à G .

IV.5.2 Complémentarité

Deux fonctions sont dites complémentaires si l'une est l'inverse de l'autre pour toutes les combinaisons d'entrées possibles.

Si $F \overline{A.B}$ et $G \overline{A \overline{B}}$, alors $F \equiv \overline{G}$, et on dit que F et G sont complémentaires.

IV.5.3 Associativité

Les opérations \cdot , \wedge et \oplus sont associatives :

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

$$A \oplus B \oplus C = (A \oplus B) \oplus C = A \oplus (B \oplus C)$$

IV.5.4 Commutativité

Les opérations \cdot , \wedge et \oplus sont commutatives :

$$A \cdot B = B \cdot A \quad A \wedge B = B \wedge A \quad A \oplus B = B \oplus A$$

IV.5.5 Distributivité

Chacune des opérations \cdot et \wedge est distributive sur l'autre :

$$A \cdot (B \wedge C) = (A \cdot B) \wedge (A \cdot C) \quad A \wedge (B \cdot C) = (A \wedge B) \cdot (A \wedge C)$$

On peut remarquer que ce théorème est particulier dans l'algèbre booléenne puisqu'ici les deux expressions sont vraies, alors que seule la première l'est dans l'algèbre ordinaire.

IV.5.6 Absorption

$$\text{Absorption 1 :} \quad A \cdot (A \cdot B) = A \quad A \cdot (A \vee B) = A$$

$$\text{Absorption 2 :} \quad (A \vee \overline{B}) \cdot B = AB \quad (A \vee \overline{B}) \vee B = A \vee B$$

Ce théorème est particulièrement intéressant pour la conception de circuits numériques puisqu'il permet d'éliminer les termes inutiles et par là-même de réduire la complexité du circuit.

IV.5.7 Dualité

Deux expressions sont dites duales si l'on obtient l'une en changeant dans l'autre, les ET par des OU, les OU par des ET, les « 1 » par des « 0 » et les « 0 » par des « 1 ».

Si on sait que $\overline{A.B} \overline{A} \overline{B}$, alors, on saura que $\overline{A+B} \overline{A} \overline{B}$ par dualité.

IV.5.8 Théorème de De Morgan

Le théorème de De Morgan est une expression du principe de dualité.

Première forme : $\overline{A+B+C+\dots} \overline{A} \overline{B} \overline{C} \dots$

Deuxième forme : $\overline{A.B.C.\dots} \overline{A} \overline{B} \overline{C} \dots$

Cf. §IV.7 page 55 pour plus de précisions.

IV.5.9 Sommes de produits, produits de sommes et forme canonique

Les expressions booléennes peuvent être manipulées sous différentes formes, certaines d'entre elles étant nécessaires pour simplifier ces expressions :

– somme de produits ;

ex. : $F(A, B, C, D) A.B.A.\overline{C}.D.B.D$

– produit de sommes ;

ex. : $F(A, B, C, D) (A.B).(A.\overline{C}.D).(B.D)$

Une expression est sous sa forme canonique si tous les symboles qui représentent les variables apparaissent dans tous les termes qui la constituent. Lorsqu'une équation est écrite à partir de sa table de vérité, elle est dans sa forme canonique.

IV.5.9.a Forme disjonctive et sommes de mintermes

Si une fonction est une somme de produits, on a une somme canonique ou forme disjonctive .

Exemple : $F \overline{A}.B.C.A.B.C.A.\overline{B}.\overline{C}.\overline{A}.\overline{B}.\overline{C}$

Une fonction booléenne peut être représentée sous forme d'une somme de produits utilisant les mintermes. Les mintermes sont représentés par des « 1 » dans une table de vérité.

La table suivante donne les mintermes d'une fonction de trois variables :

A	B	C	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
			$\overline{A}.\overline{B}.\overline{C}$	$\overline{A}.B.\overline{C}$	$\overline{A}.\overline{B}.C$	$\overline{A}.B.C$	$A.\overline{B}.\overline{C}$	$A.\overline{B}.C$	$A.B.\overline{C}$	$A.B.C$
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

IV.5.9.b Forme conjonctive et produits de maxtermes

Si une fonction est un produit de somme, on a un produit canonique ou forme conjonctive .

Exemple : $G (\bar{A} B C).(A B C).(A \bar{B} \bar{C}).(\bar{A} \bar{B} \bar{C})$

Une fonction booléenne peut être représentée sous forme d'un produit de sommes utilisant les maxtermes. Les maxtermes sont représentés par des « 0 » dans une table de vérité.

La table suivante donne les maxtermes d'une fonction de trois variables :

A	B	C	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
			$A+B+C$	$A+B+\bar{C}$	$A+\bar{B}+C$	$A+\bar{B}+\bar{C}$	$\bar{A}+B+C$	$\bar{A}+B+\bar{C}$	$\bar{A}+\bar{B}+C$	$\bar{A}+\bar{B}+\bar{C}$
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

IV.5.9.c Représentations d'une fonction sous forme de mintermes et maxtermes

Soit la fonction \mathcal{F} telle que $\mathcal{F}(A,B,C) A.B \bar{B}.(\bar{A} \bar{C})$.

Cette fonction peut être représentée sous sa :

- première forme canonique (somme de mintermes) : on développe la fonction sous la forme d'une somme de produits puis on prend chaque terme avec pour variable manquante X et on applique un ET logique avec $X \bar{X}$;
- deuxième forme canonique (produit de maxtermes) : on développe la fonction sous la forme d'un produit de sommes puis on prend chaque terme avec pour variable manquante X et on applique un OU logique avec $X.\bar{X}$;

▷ **Exemple 4.5**

Représentation sous forme de somme de mintermes :

$$\begin{aligned} \mathcal{F}(A,B,C) & A.B \bar{B}.(\bar{A} \bar{C}) \\ & A.B \bar{A}.\bar{B} \bar{B}.\bar{C} \\ & A.B.(C \bar{C}).\bar{A}.\bar{B}.(C \bar{C}).\bar{B}.\bar{C}.(A \bar{A}) \\ & \bar{A}.\bar{B}.\bar{C} \bar{A}.\bar{B}.\bar{C} A.\bar{B}.\bar{C} A.B.\bar{C} A.B.C \\ & \sum m(0, 1, 4, 6, 7) \end{aligned}$$

▷ Exemple 4.6

Représentation sous forme de produit de maxtermes

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

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

$$(A.\overline{B}).(A.\overline{B}.\overline{C}).(\overline{A}.\overline{B}.\overline{C}) \text{ par distributivité}$$

$$(A.\overline{B}.\overline{C}.\overline{C}).(A.\overline{B}.\overline{C}).(\overline{A}.\overline{B}.\overline{C})$$

$$(A.\overline{B}.\overline{C}).(A.\overline{B}.\overline{C}).(\overline{A}.\overline{B}.\overline{C})$$

$$\prod M(2, 3, 5)$$

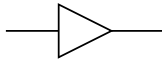
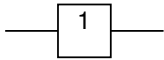
IV.5.10 Résumé des propriétés des opérateurs OU et ET

Propriété	OU	ET
Identité	$a \ 0 \ a$	$a.1 \ a$
Élément neutre	$a \ 0 \ a$	$a.1 \ a$
Élément absorbant	$a \ 1 \ 1$	$a.0 \ 0$
Idempotence	$a \ a \ a$	$a.a \ a$
Complémentation	$a \ \overline{a} \ 1$	$a.\overline{a} \ 0$
Involution	$\overline{\overline{a}} \ a$	$\overline{\overline{a}} \ a$
Commutativité	$a \ b \ b \ a$	$a.b \ b.a$
Associativité	$a \ (b \ c)$ $(a \ b) \ c$	$a.(b.c)$ $(a.b).c$
Distributivité	$a \ (b.c)$ $(a \ b).(a \ c)$	$a.(b \ c)$ $(a.b).(a.c)$
Absorption 1	$a \ a.b \ a$	$a.(a \ b) \ a$
Absorption 2	$a \ \overline{a}.b \ a \ b$	$a.(\overline{a} \ b) \ a.b$
Consensus	$a.b \ \overline{a}.c \ bc$ $a.b \ \overline{a}c$	$(a \ b).(\overline{a} \ c).(b \ c)$ $(a \ b).(\overline{a} \ c)$
	$(a \ b).(\overline{a} \ b)$	$(a.b)(\overline{a}.b)$
De Morgan	$\overline{a \ b} \ \overline{a}.b$	$a.b \ \overline{a \ b}$

IV.6 Opérateurs logiques élémentaires et composés

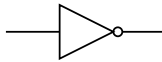
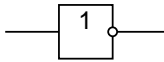
Les fonctions logiques sont conçues à partir d'un groupe d'opérateurs élémentaires appelés « portes ». Chaque opérateur est représenté par un symbole et sa fonction est définie par une table de vérité.

IV.6.1 OUI : identité ou transfert



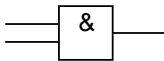
A	S A
0	0
1	1

IV.6.2 NON (NOT) : complément « - »



A	S \bar{A}
0	1
1	0

IV.6.3 ET (AND) : produit logique « . »



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

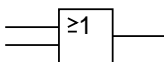
Propriétés du ET :

$a.1 = a$ $a.\bar{a} = 0$ $a.0 = 0$ $a.a = a$

Élément neutre : 1

Élément absorbant : 0

IV.6.4 OU (OR) : somme logique « + »



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

Propriétés du OU :

$$a \ 1 \ 1 \quad a \bar{a} \ 1 \quad a \ 0 \ a \quad a \ a \ a$$

Élément neutre : 0

Élément absorbant : 1

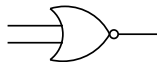
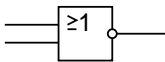
► **Remarque 4.1**

Les opérateurs {ET,OU,NON} permettent à eux trois de réaliser n'importe quelle fonction logique : on dit qu'ils forment un groupe complet.

Le théorème de De Morgan permet de dire que les groupes {ET,NON} et {OU,NON} sont également des groupes complets.

IV.6.5 NON-OU (NOR) « ↓ »

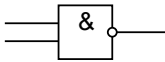
Les deux opérateurs OU et NON peuvent être combinés en un seul opérateur NON-OU : NON-OU est donc un opérateur complet.



A	B	S	$\overline{A.B}$
0	0	1	
0	1	0	
1	0	0	
1	1	0	

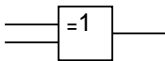
IV.6.6 NON-ET (NAND) « ↑ »

Les deux opérateurs ET et NON peuvent être combinés en un seul opérateur NON-ET : NON-ET est donc un opérateur complet.



A	B	S	$\overline{A.B}$
0	0	1	
0	1	1	
1	0	1	
1	1	0	

IV.6.7 OUX (XOR) : ou exclusif ou dilemme « ⊕ »



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

Propriétés du OUX :

Le ou exclusif est commutatif et associatif

$$a \oplus 0 \ a \quad a \oplus 1 \ \bar{a} \quad a \oplus \bar{a} \ 1 \quad a \oplus a \ 0$$

Élément neutre : 0

Élément absorbant : a, \bar{a}

► **Remarque 4.2**

Le ou exclusif est souvent utilisé dans les circuits numériques du fait de ses propriétés :

- le ou exclusif est l'opérateur somme modulo 2, on le retrouve donc dans les additionneurs ou la sortie $S = a \oplus b \oplus r$;
- il est également largement utilisé dans les circuits de correction d'erreurs (calcul de parité) : $b_0 \oplus b_1 \oplus b_2 \oplus \dots \oplus b_n$ est égal à 0 si le nombre de bits à 1 est pair, à 0 sinon ;
- $a \oplus 1 = \bar{a}$ et $a \oplus 0 = a$: le OU exclusif peut être utilisé comme inverseur commandé.

Le ou exclusif n'est pas un opérateur complet, mais comme il peut être utilisé pour réaliser la complémentation, les groupes {OUX,ET} et {OUX,OU} sont des groupes complets.

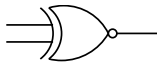
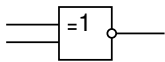
► **Remarque 4.3**

Relations d'identité utilisables avec l'opérateur ou exclusif :

1. $a \oplus b = a\bar{b} + \bar{a}b = (a \cdot b) \cdot (\bar{a} \bar{b})$
2. $\overline{(a \oplus b)} = a\bar{b} + \bar{a}b = ab + \bar{a}\bar{b} = (\bar{a} \cdot b)(a \cdot \bar{b})$
3. $a \oplus a = 0$ et $a \oplus \bar{a} = 1$
4. $a \oplus 1 = \bar{a}$ et $a \oplus 0 = a$
5. $a(b \oplus z) = ab \oplus az$
6. $a \cdot b = a \oplus b \oplus ab = a \oplus \bar{a}b$
7. $a \cdot b = a \oplus b$ si $ab = 0$
8. $a \oplus b = c \Rightarrow c \oplus b = a, c \oplus a = b, a \oplus b \oplus c = 0$
9. $a \oplus (a \cdot b) = \bar{a}b$
10. $a \oplus ab = \bar{a}b$

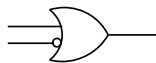
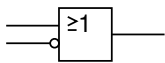
IV.6.8 NON-OUX (XNOR) : coïncidence ou équivalence

« \odot »



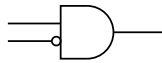
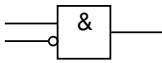
A	B	S	$A \odot B$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

IV.6.9 IMP (IMP) : implication « \subset » ou « \supset »



A	B	S	$A \supset B$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	1	1

IV.6.10 INH (INIB) : inhibition « / »



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

IV.6.11 Résumé : les différents opérateurs

Nom	Symbole	Valeur de xy				Expression algébrique
		00	01	10	11	
Zéro		0	0	0	0	$F_0 \ 0$
Et	$x.y$	0	0	0	1	$F_1 \ x.y$
Inhibition	x/y	0	0	1	0	$F_2 \ x.\bar{y}$
Transfert		0	0	1	1	$F_3 \ x$
Inhibition	y/x	0	1	0	0	$F_4 \ \bar{x}.y$
Transfert		0	1	0	1	$F_5 \ y$
Ou exclusif	$x\oplus y$	0	1	1	0	$F_6 \ x\bar{y} \bar{x}y$
Ou	$x \vee y$	0	1	1	1	$F_7 \ x \vee y$
Non-ou	$x \downarrow y$	1	0	0	0	$F_8 \ \bar{x} \bar{y}$
Équivalence	$x\odot y$	1	0	0	1	$F_9 \ xy \ \bar{x}\bar{y}$
Complément	\bar{y}	1	0	1	0	$F_{10} \ \bar{y}$
Implication	$x \subset y$	1	0	1	1	$F_{11} \ x \bar{y}$
Complément	\bar{x}	1	1	0	0	$F_{12} \ \bar{x}$
Implication	$x \supset y$	1	1	0	1	$F_{13} \ \bar{x} y$
Non-et	$x \uparrow y$	1	1	1	0	$F_{14} \ \bar{x}.\bar{y}$
Un		1	1	1	1	$F_{15} \ 1$

IV.7 Universalité des portes NON-ET et NON-OU

►Note 4.1

Théorème de De Morgan :

1. Le complément d'un produit est égal à la somme des compléments des termes du produit : $\overline{S \ a.b} = \bar{a} \bar{b}$
2. Le complément d'une somme est égal au produit des compléments des termes de la somme : $\overline{S \ a+b} = \bar{a} \bar{b}$

Le théorème de De Morgan et ses conséquences est :

- très utile pour simplifier des expressions ;

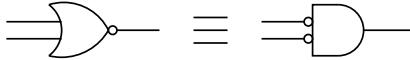
- valable également si a ou b sont des expressions contenant plusieurs variables

► **Exemple 4.7**

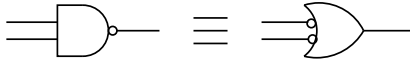
$$\overline{(A \cdot \overline{B} \cdot C)} = (\overline{A \cdot \overline{B}}) \cdot \overline{C} = \overline{A} \cdot B \cdot \overline{C}$$

- conséquences :

1. une porte NON-OU est une porte ET avec ses entrées inversées :



2. une porte NON-ET est une porte OU avec ses entrées inversées :

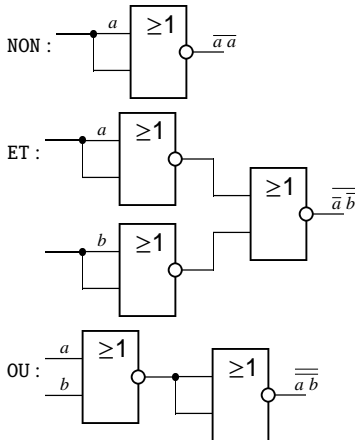


► **Note 4.2**

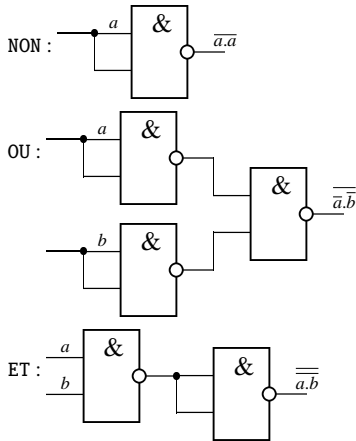
Universalité des portes NON-ET et des portes NON-OU :

Toutes les portes logiques élémentaires (ET, OU, NON) peuvent être réalisées avec des portes NON-OU ou NON-ET.

IV.7.1 Universalité des portes NON-OU

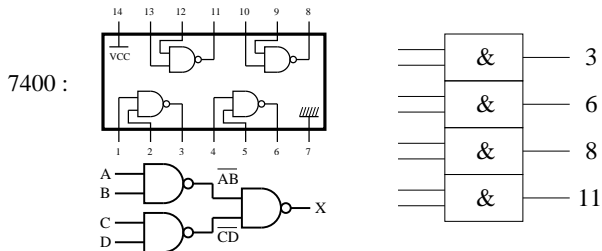


IV.7.2 Universalité des portes NON-ET



► Exemple 4.8

Réaliser la fonction $X=AB+CD$ à l'aide du CI (circuit intégré) suivant :



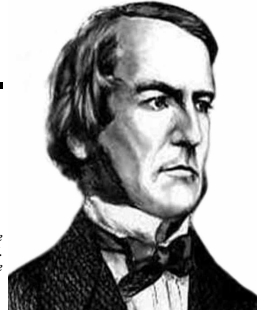
► Remarque 4.4

- le groupe d'opérateurs {ET,OU,NON} permet de réaliser toutes les fonctions logiques : on dit que c'est un « groupe complet », ainsi que les groupes {ET,NON} et {OU,NON} ;
- de même, les opérateurs NON-ET, NON-OU, sont appelés des « opérateurs complets » ;
- comme l'opérateur OUX peut être utilisé pour réaliser un inverseur, les groupes {ET,OUX} et {OU,OUX} sont également des groupes complets ; le groupe {ET,OUX} est un anneau booléen appelé corps de Galois.

Chapitre V

Représentation et simplification des fonctions logiques

George Boole
* 2 nov. 1815, Lincoln, R.-U.
† 8 déc. 1864, Ballinacorney, Irlande



« Une proposition peut être vraie ou fausse, mais ne peut pas être vraie et fausse. »
(Aristote ■ 384, † 322 av. J.-C.)

Tout circuit logique peut être décrit par des fonctions logiques et/ou une table de vérité, et être réalisé à partir des opérateurs logiques élémentaires.

V.1 Méthodes de représentation des fonctions logiques

En dehors de la représentation algébrique que nous avons utilisée jusqu'à présent, d'autres méthodes permettent de représenter les fonctions logiques. Les plus couramment employées sont les représentations tabulaires, implicites, et graphiques.

V.1.1 Représentations tabulaires

V.1.1.a Table de vérité

La table de vérité nous fait connaître la réaction d'un circuit logique aux diverses combinaisons de niveaux logiques appliquées à ses entrées. Chaque ligne présente la combinaison des variables d'entrée ainsi que la ou les sorties correspondante(s).

▷ **Exemple 5.1**

La table de vérité d'un additionneur complet est la suivante :

A	B	C	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

$$\left\{ \begin{array}{l} S \ A \oplus B \oplus C \\ R \ A.B \ A.C \ B.C \end{array} \right.$$

▷ Exemple 5.2

Donner la table de vérité d'un circuit à 3 entrées A,B,C et 2 sorties X,Y tel que :

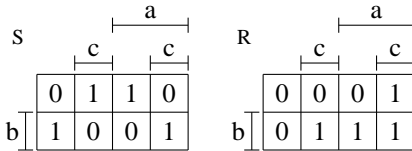
$$\left\{ \begin{array}{l} X=1 \quad \text{si les 3 entrées ont le même niveau} \\ Y=1 \quad \text{si } A=B \end{array} \right.$$

Le principal inconvénient de la table de vérité est qu'elle devient rapidement très encombrante lorsque le nombre de variables d'entrée augmente.

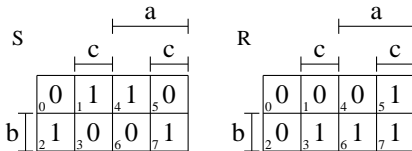
V.1.1.b Diagramme de Veitch

Le diagramme de Veitch est une table sur laquelle on représente les n variables d'entrée selon les deux axes vertical et horizontal. En général, pour $n = p + q$ on porte sur les colonnes p variables où p est la partie entière de $n/2$, et les q variables restantes sur les lignes. Les colonnes et les lignes sont numérotées selon l'ordre binaire naturel.

Le diagramme de Veitch de l'exemple précédent est le suivant :



On peut également numéroté les cases du diagramme de Veitch selon l'image décimale de la fonction représentée. Chaque case correspond à une ligne de la table de vérité, et peut donc être représentée par son image décimale (cf. §V.1.2.b) :



V.1.1.c Diagramme de Karnaugh[KAR53] et termes adjacents

Le diagramme de Karnaugh est un outil graphique, méthodique. Il permet d'obtenir une solution optimale à la simplification logique (cf. §V.2.3 page 64).

Comme la table de vérité, le diagramme de Karnaugh met en évidence le rapport entre les entrées et les sorties (chaque ligne de la table de vérité correspond à une case du diagramme de Karnaugh).

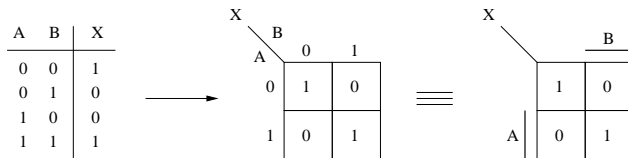
Deux termes sont adjacents quand ils ne diffèrent l'un de l'autre que par une seule variable. ABC et ABC sont adjacents. Un diagramme – ou tableau – de Karnaugh est une table d'implication logique disposée de telle manière que deux termes logiquement adjacents soient également adjacents géométriquement.

Le diagramme de Karnaugh est très proche du diagramme de Veitch présenté §V.1.1.b, mais afin d'exploiter la notion d'adjacence entre les termes, les cases sont ordonnées selon le code binaire réfléchi, au lieu du code binaire naturel.

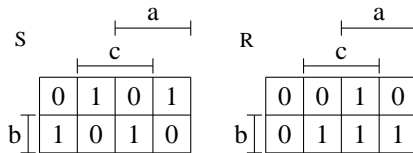
► **Remarque 5.1**

Les tableaux de Karnaugh se présentent comme des cylindres fermés dans les deux sens.

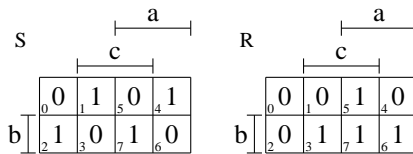
▷ **Exemple 5.3**



Le diagramme de Karnaugh de l'exemple précédent est le suivant :



Comme pour le diagramme de Veitch, on peut numéroter les cases du diagramme de Karnaugh selon l'image décimale de la fonction représentée :



► **Remarque 5.2**

- il peut exister des états indifférents (notés « X »). Ces états correspondent à des combinaisons d'entrée impossibles. On les remplacera par 1 ou 0 de façon à avoir la simplification la plus optimale ;
- on peut utiliser une même case plusieurs fois, puisque $x x x \cdots x x$.

Chaque case du tableau représente une combinaison et une seule des variables de la fonction. Dans cette case, on inscrit « 0 » ou « 1 » selon la valeur prise par la fonction. Cette combinaison exclusive de variables peut être notée par un ET entre les variables représentées.

Par exemple, la case pour laquelle $a = 0, b = 1, c = 0$ et $d = 1$ sera notée $\bar{a}b\bar{c}d$: c'est un « minterme ».

La représentation de la fonction sera alors la somme logique (OU) de toutes les combinaisons pour lesquelles la fonction vaut « 1 ».

Quelquefois, on peut préférer considérer la seconde forme canonique. La combinaison exclusive de variables sera alors notée par un OU entre les variables représentées.

Par exemple, la case pour laquelle $a = 0, b = 1, c = 0$ et $d = 1$ sera notée $\bar{a} b \bar{c} d$: c'est un « maxterme ».

V.1.1.d Diagramme de Venn

À venir ...

V.1.1.e Diagramme de Johnston

À venir ...

V.1.1.f Diagramme de Carroll

À venir ...

V.1.2 Représentations implicites

V.1.2.a Image caractéristique

L'image caractéristique d'une fonction \mathcal{F} à n entrée est constituée des 2^n valeurs de cette fonction, ordonnées selon l'ordre binaire naturel.

Ainsi, soit la fonction $\mathcal{F}(x_0, x_1)$ suivante, définie par sa table de vérité :

x_0	x_1	\mathcal{F}
0	0	0
0	1	1
1	0	0
1	1	0

On peut représenter \mathcal{F} par son image caractéristique, soit $I_c[\mathcal{F}(x_0, x_1)] = 0100$.

Reprenons la table de vérité d'un additionneur complet :

A	B	C	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

L'image caractéristique de S est $I_c[S(A,B,C)]$ 01101001.

L'image caractéristique de R est $I_c[R(A,B,C)]$ 00010111.

V.1.2.b Image décimale

Nous avons vu que toute fonction logique \mathcal{F} peut s'exprimer par ses formes canoniques, soit comme somme de produits, soit comme produit de sommes.

On notera donc la fonction \mathcal{F} comme :

⇨ la somme des états pour lesquels elle vaut « 1 » que l'on notera : \mathcal{F}_1

$$\sum (d_1, \dots, d_p)$$

⇨ le produit des états pour lesquels elle vaut « 0 » que l'on notera : \mathcal{F}_0

$$\prod (d_1, \dots, d_p)$$

où d_1 à d_p représentent les valeurs décimales des nombres binaires représentés par les variables de la fonction

Reprenons comme exemple la fonction $\mathcal{F}(x_0, x_1)$ présentée §V.1.2.a. On numérote les différents états de cette fonction en attribuant des poids aux variables selon l'ordre binaire naturel ; notons N la valeur décimale de ces états :

N	x_0	x_1	\mathcal{F}
0	0	0	0
1	0	1	1
2	1	0	0
3	1	1	0

La fonction $\mathcal{F}(x_0, x_1)$ peut s'écrire :

$$\mathcal{F}_1 \overline{x_0}.x_1 \sum (1)$$

$$\mathcal{F}_0 (\overline{x_0} \overline{x_1}).(x_0 \overline{x_1}).(x_0 x_1) \prod (0, 2, 3)$$

Le principal avantage de la notation décimale est le risque d'erreur très faible lors de son écriture. En effet, il est plus difficile de remplacer un 3 par un 1 que d'oublier une barre de complémentération sur une variable.

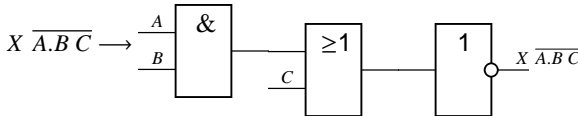
De plus, on a vu dans les sections V.1.1.b et V.1.1.c que cette notation est utilisée pour numérotter les cases des diagrammes de Veitch et de Karnaugh, et faciliter ainsi la représentation d'une fonction sous forme de diagramme.

V.1.3 Représentations graphiques

V.1.3.a Logigramme

Un logigramme est un schéma illustrant l'expression d'une fonction logique sans tenir compte des constituants technologiques.

▷ **Exemple 5.4**



► **Remarque 5.3**

Notation : Par convention, une entrée ou une sortie d'opérateur logique active à un niveau haut sera notée a , b , sel , etc.

Une entrée ou une sortie d'opérateur logique active à un niveau bas sera notée \bar{c} , \bar{d} , \overline{MEM} , etc.

V.2 Simplification d'expressions logiques

À venir ...

V.2.1 Formes canoniques d'une fonction logique

À venir ...

V.2.2 Méthode algébrique

Il n'est pas facile de trouver le résultat minimal → application des théorèmes de De Morgan, factorisation, astuce, ...

▷ **Exemple 5.5**

$$x + \bar{x}y = x(1+y) + \bar{x}y = x + xy + \bar{x}y = x + y \quad (\text{théorème d'allègement})$$

$$x.(x+y) = x + xy = x \quad (\text{absorption})$$

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

V.2.3 Simplification par diagramme de Karnaugh

La méthode de simplification d'une fonction par diagramme de Karnaugh s'appuie sur l'adjacence entre les termes de la fonction pour en extraire la représentation la plus simple possible.

Les diagrammes de Karnaugh contiennent des ensembles de termes (« 0 » ou « 1 ») nommés implicants. Ces ensembles sont des :

- implicants simples lorsqu'il s'agit de termes isolés ;
- implicants majeurs lorsqu'il s'agit d'ensembles contenant 2^n termes aussi grands que possible ;
- implicants majeurs essentiels lorsque les termes considérés ne sont présents dans aucun autre implicant ;
- implicants majeurs non essentiels lorsqu'un terme est présent dans plusieurs implicants.

V.2.3.a Simplification par extraction des sommes de produits

La méthode est la suivante :

1. dessiner la table de Karnaugh correspondant à la fonction ;
 - on entame les 1 isolés ;
 - on réunit les octets de 1 adjacents ;
 - on réunit les quartets de 1 adjacents ;
 - on réunit les doublets de 1 adjacents pour réunir tous les 1 du tableau ;
2. identifier tous les implicants majeurs essentiels pour les « 1 » ;
3. identifier tous les implicants majeurs non essentiels pour les « 1 » ;
4. pour tous les implicants majeurs essentiels et un des implicants majeurs non essentiels sélectionné dans chaque ensemble, déterminer les termes de produits correspondant ;
5. effectuer l'addition logique de tous les termes précédents, sachant que :
 - un octet de 1 permet d'éliminer les 3 variables qui se trouvent sous les deux formes (complémenté et non complémenté) ;
 - un quartet de 1 permet d'éliminer les 2 variables qui se trouvent sous les deux formes (complémenté et non complémenté) ;
 - un doublet de 1 permet d'éliminer la variable qui se trouve sous les deux formes (complémenté et non complémenté) ;

► Exemple 5.6

Simplifier la fonction :

$$\mathcal{F}(A, B, C) \sum m(0, 1, 4, 5) \bar{A}.\bar{B}.\bar{C} \bar{A}.\bar{B}.C A.\bar{B}.\bar{C} A.\bar{B}.C$$

Solution :

	----- A				
F	----- C				
	1	1	1	1	1
B	0	0	0	0	0
	2	3	7	6	

$$\left\{ \begin{array}{l} \mathcal{F}_1 (0, 1, 4, 5) \\ \mathcal{F}_0 (2, 3, 6, 7) \end{array} \right.$$

- l'implicant majeur essentiel est \bar{B}

– il n’y a aucun implicant majeur non essentiel

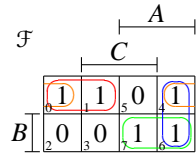
La solution est $\mathcal{F}(A, B, C) \bar{B}$

▷ Exemple 5.7

Simplifier la fonction :

$$\mathcal{F}(A, B, C) \sum m(0, 1, 4, 6, 7) \bar{A}.\bar{B}.\bar{C} \bar{A}.\bar{B}.C A.\bar{B}.\bar{C} A.B.\bar{C} A.B.C$$

Solution :



$$\begin{cases} \mathcal{F}_1 (0, 1, 4, 6, 7) \\ \mathcal{F}_0 (2, 3, 5) \end{cases}$$

– les implicants majeurs essentiels sont $\bar{A}.\bar{B}$ et $A.B$

– les implicants majeurs non essentiels sont $\bar{B}.\bar{C}$ ou $A.\bar{C}$

La solution est $\mathcal{F}(A, B, C) A.B \bar{A}.\bar{B} \bar{B}.\bar{C}$

ou $\mathcal{F}(A, B, C) A.B \bar{A}.\bar{B} A.\bar{C}$

Normalement, l’utilisation des tableaux de Karnaugh pour la simplification par extraction des sommes de produits exploite l’adjacence entre les « 1 » pour représenter la fonction à simplifier. Cependant, il est possible d’utiliser les « 0 » en procédant exactement de la même manière : on obtiendra alors une représentation de la fonction complé- mentée.

V.2.3.b Simplification par extraction des produits de sommes

La méthode est la suivante :

1. dessiner la table de Karnaugh correspondant à la fonction ;
 - on entame les 0 isolés ;
 - on réunit les octets de 0 adjacents ;
 - on réunit les quartets de 0 adjacents ;
 - on réunit les doublets de 0 adjacents pour réunir tous les 1 du tableau ;
2. identifier tous les implicants majeurs essentiels pour les « 0 » ;
3. identifier tous les implicants majeurs non essentiels pour les « 0 » ;
4. pour tous les implicants majeurs essentiels et un des implicants majeurs non essentiels sélectionné dans chaque ensemble, déterminer les termes de sommes correspondant ;
5. effectuer l’addition logique de tous les termes précédents, sachant que :
 - un octet de 0 permet d’éliminer les 3 variables qui se trouvent sous les deux formes (complémenté et non complémenté) ;

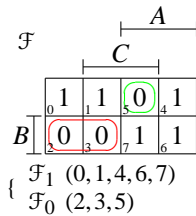
- un quartet de 0 permet d'éliminer les 2 variables qui se trouvent sous les deux formes (complémenté et non complémenté);
- un doublet de 0 permet d'éliminer la variable qui se trouve sous les deux formes (complémenté et non complémenté);

▷ **Exemple 5.8**

Simplifier la fonction :

$$\mathcal{F}(A, B, C) \prod M(2, 3, 5) \quad A.\bar{B}.C \quad A.\bar{B}.\bar{C} \quad \bar{A}.B.\bar{C}$$

Solution :



- les implicants majeurs essentiels sont $A\bar{B}$ et $\bar{A}B\bar{C}$
- il n'y a aucun implicant majeur non essentiel

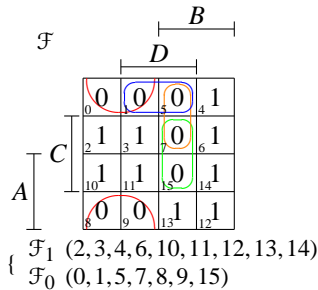
La solution est $\mathcal{F}(A, B, C) (A\bar{B}).(\bar{A}B\bar{C})$

▷ **Exemple 5.9**

Simplifier la fonction :

$$\mathcal{F}(A, B, C) \prod M(0, 1, 5, 7, 8, 9, 15)$$

Solution :



- les implicants majeurs essentiels sont BC et $\bar{B}\bar{C}\bar{D}$
- les implicants majeurs non essentiels sont $A\bar{B}\bar{D}$ ou $A\bar{C}\bar{D}$

La solution est $\mathcal{F}(A, B, C) (BC).(\bar{B}\bar{C}\bar{D}).(A\bar{B}\bar{D})$

$$\text{ou } \mathcal{F}(A, B, C) (B C).(\overline{B} \overline{C} \overline{D}).(A C \overline{D})$$

Normalement, l'utilisation des tableaux de Karnaugh pour la simplification par extraction des produits de sommes exploite l'adjacence entre les « 0 » pour représenter la fonction à simplifier. Cependant, il est possible d'utiliser les « 1 » en procédant exactement de la même manière : on obtiendra alors une représentation de la fonction complé- mentée.

V.2.3.c Cas des états indéterminés ou indifférents

⚡ Dans le cas général, l'utilisation des « 1 » ou des « 0 » doit conduire à des fonc- tions équivalentes (l'une étant la complémentaire de l'autre), même si les écritures peuvent être différentes. Cependant, il faut considérer avec attention le cas particulier des fonctions non complètement définies.

Certaines fonctions logiques sont données comme étant incomplètes (avec des états indéterminés) ou avec des états indifférents (combinaisons de variables d'entrées n'in- fluençant pas le résultat). Ces conditions permettent de simplifier le tableau de Karnaugh, et par là-même, l'implantation de la fonction sous forme matérielle.

En plus des ensembles de « 0 » et des ensembles de « 1 », il y a donc également des ensembles de « X » ou « - » qui représentent les états indéterminés/indifférents de la fonction à minimiser. Ces états « X » ou « - » peuvent être rassemblés indifféremment avec des « 0 » ou « 1 » pour simplifier la minimisation logique dans les tableaux de Karnaugh.

Ainsi :

- les cases non définies d'un diagramme de Karnaugh peuvent être exploitées dans une simplification par les « 1 » comme dans une simplification par les « 0 » ;
- en conséquence, une même case pourra avoir été utilisée à la fois dans la repré- sentation directe de la fonction, et dans sa représentation complé- mentée ;
- ainsi, si les deux représentations obtenues sont toutes deux justes, elles ne sont en aucun cas identiques, ni même équivalente : les fonctions sont différentes, bien que correspondant toutes deux au même diagramme de Karnaugh.

▷ **Exemple 5.10**

Soit le tableau de Karnaugh suivant à simplifier sous forme de somme de produit :

		B			
		-----	-----	-----	-----
A	C	D			
		-----	-----	-----	-----
	-----	-----	-----	-----	-----
	-----	-----	-----	-----	-----
		0	1	5	4
		0	0	0	1
		2	3	7	6
		1	1	0	1
		10	11	15	14
		X	1	0	X
		8	9	13	12
		0	0	X	1

\mathcal{F}_1 (2, 3, 4, 6, 11, 12)
 $\{ \mathcal{F}_0$ (0, 1, 5, 7, 8, 9, 15)
 \mathcal{F}_X (10, 13, 14)

Solution :

- les implicants majeurs essentiels sont $B.\bar{D}$ et $\bar{B}.C$
- il n'y a aucun implicant majeur non essentiel

La solution $\sum \prod$ est $\mathcal{F}(A, B, C) B.\bar{D} \bar{B}.C$

▷ **Exemple 5.11**

Soit le tableau de Karnaugh suivant à simplifier sous forme de produit de somme :

		B			
		D			
A	C	0	1	5	4
		2	3	7	6
	10	11	13	14	
	8	9	15	12	
		0	1	0	1
		1	1	0	1
		X	1	0	X
		0	0	X	1

\mathcal{F}_1 (2, 3, 4, 6, 11, 12)
 $\{ \mathcal{F}_0$ (0, 1, 5, 7, 8, 9, 15)
 \mathcal{F}_X (10, 13, 14)

Solution :

- les implicants majeurs essentiels sont $B C$ et $\bar{B} \bar{D}$
- il n'y a aucun implicant majeur non essentiel

La solution $\prod \sum$ est $\mathcal{F}(A, B, C) (B C).(\bar{B} \bar{D})$

V.2.4 Méthodes algorithmiques

Au delà de 6 variables, on utilise des méthodes algorithmiques (méthode de Quine).

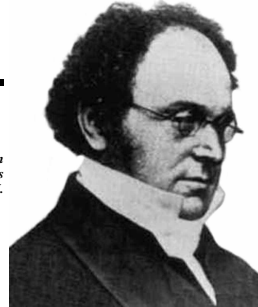
V.2.4.a Algorithme de Quine-McCluskey

À venir ...

Chapitre VI

Les circuits combinatoires

Augustus De Morgan
★ 27 juin 1806, Madura, Indes
† 18 mars 1871, Londres, R.-U.



$\text{comp}(\bigcap_j A_j) \bigcup_j \text{comp}(A_j)$ i.e. le complément de l'intersection d'un nombre quelconque d'ensembles est égal à l'union de leurs compléments.

$\text{comp}(\bigcup_j A_j) \bigcap_j \text{comp}(A_j)$ i.e. le complément de l'union d'un nombre quelconque d'ensembles est égal à l'intersection de leurs compléments.

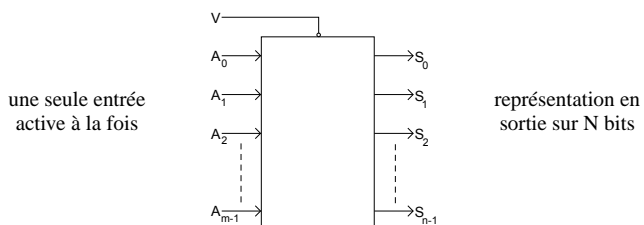
VI.1 Circuits logiques combinatoires usuels

Circuit combinatoire : circuit dont les sorties dépendent uniquement de la combinaison des états des entrées à l'instant de l'observation.

VI.1.1 Circuits de transcodage (codeurs, décodeurs, convertisseurs)

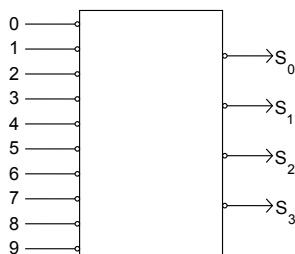
VI.1.1.a Codeur (encodeur)

Circuit à $M=2^N$ entrées et N sorties qui code en binaire le rang de la seule entrée active.



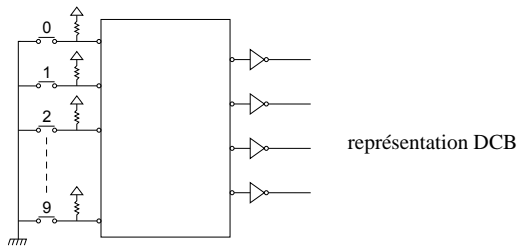
▷ **Exemple 6.1**

Codeur décimal-DCB : 10 entrées, 4 sorties



$\overline{A_9}$	$\overline{A_8}$	$\overline{A_7}$	$\overline{A_6}$	$\overline{A_5}$	$\overline{A_4}$	$\overline{A_3}$	$\overline{A_2}$	$\overline{A_1}$	$\overline{A_0}$	$\overline{S_3}$	$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$
1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	0	1	1	0	0	1	0
1	1	1	1	1	1	0	1	1	1	0	0	1	1
1	1	1	1	1	0	1	1	1	1	0	1	0	0
1	1	1	1	0	1	1	1	1	1	0	1	0	1
1	1	1	0	1	1	1	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0	1

Application : codeur de clavier numérique



► **Remarque 6.1**

Les codeurs de priorités sont une version modifiée du codeur : quand deux entrées sont actives, c'est l'entrée correspondant au nombre le plus haut qui est choisi.

VI.1.1.b Décodeur

Le décodeur est un circuit qui établit la correspondance entre un code d'entrée sur N bits et M lignes de sortie ($M \leq 2^N$).

Pour chacune des combinaisons d'entrée, une seule ligne de sortie est validée.

► **Exemple 6.2**

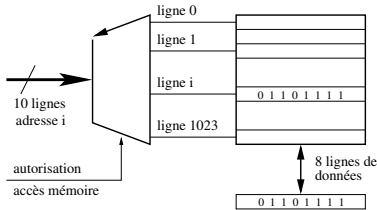
Décodeur DCB-décimal : 4 entrées, 10 sorties.

► **Remarque 6.2**

La plupart des décodeurs sont dotés d'une ou plusieurs entrées de validation qui commandent son fonctionnement.

Applications des décodeurs

1 : Adressage d'une mémoire



- une mémoire est un tableau d'éléments binaires (divisés en lignes et colonnes) ;
- pour lire un mot mémoire, il faut lui envoyer le numéro de ligne souhaité (adresse) ;
- souvent, le décodeur est interne à la mémoire.

2 : Génération de fonction

Toute fonction logique peut être réalisée à partir d'une combinaison de décodeur.

▷ **Exemple 6.3**

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

▶ **Remarque 6.3**

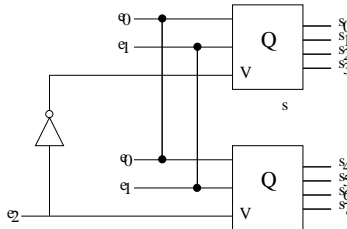
Il n'est pas nécessaire de simplifier la fonction avant la réalisation.

Mise en cascade des décodeurs

Utilisation de l'entrée de validation.

▷ **Exemple 6.4**

Réaliser un décodeur à 3 entrées en utilisant 2 décodeurs à 2 entrées.



Réaliser un décodeur à 16 sorties à l'aide de décodeurs à 4 sorties.

VI.1.1.c Transcodeurs (convertisseurs)

Circuit à p entrées et k sorties qui convertit un nombre écrit dans un code C1 en un nombre écrit dans un code C2.

▷ **Exemple 6.5**

Code binaire → code Gray

Code DCB → code affichage chiffre (décodeur 7 segments)

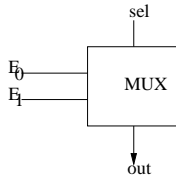
VI.1.2 Multiplexeurs–démultiplexeurs

VI.1.2.a Multiplexeurs (MUX)

Circuit à 2ⁿ entrées d’informations, n entrées de sélection, et une sortie. Il permet l’aiguillage de l’une de ces entrées vers la sortie.

▷ **Exemple 6.6**

MUX à 2 entrées de données



E ₁	E ₀	sel	out
X	X	0	E ₀
X	X	1	E ₁

→ $S = \overline{\text{sel}}.E_0 + \text{sel}.E_1$

► **Remarque 6.4**

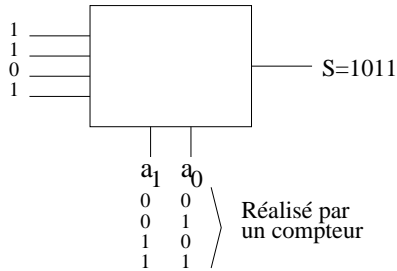
La table de vérité devient rapidement très importante (à partir de 4 entrées). On exprime alors la fonction de sortie directement

▷ **Exemple 6.7**

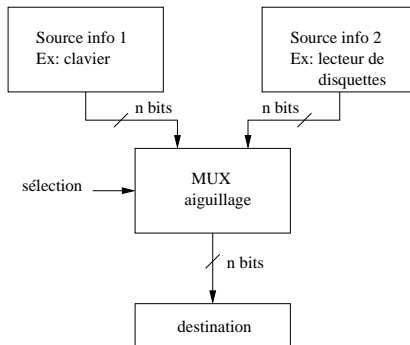
MUX à 4 entrées (→ 2 entrées de sélection a₁ a₀) $S = \overline{a_1}.\overline{a_0}.E_0 + \overline{a_1}.a_0.E_1$
...

VI.1.2.b Application des MUX

1. **Conversion parallèle–série** : on place successivement les valeurs 00, 01, 10, 11 sur a₁ a₀.



2. **Générateur de fonctions** : toute fonction logique peut être réalisée à partir des MUX. Les entrées de sélection (commande) sont alors les variables de la fonction.
3. **Sélection de mots** : le MUX est réalisé à partir de n MUX à 2 entrées travaillant avec la même commande de sélection.



► **Remarque 6.5**

Intérêt : il n'est pas nécessaire de simplifier la fonction avant de la réaliser.

▷ **Exemple 6.8**

$F \text{ ABC } \overline{ABC}$
 Utilisation de MUX 8 vers 1.
 $S \text{ ABCE}_0 \text{ ABCE}_1 \dots \text{ ABCE}_4 \dots$

VI.1.2.c Démultiplexeurs (DEMUX)

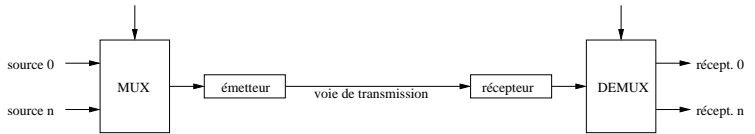
Circuit à 2^n sorties, 1 entrée d'information, n entrées de commande. Il permet l'agencement d'information de l'entrée vers l'une des sorties.

► **Remarque 6.6**

Le MUX-DEMUX est un circuit programmable : les relations entre entrées et sorties sont modifiables.

▷ **Exemple 6.9**

Transmission avec multiplexage/démultiplexage.



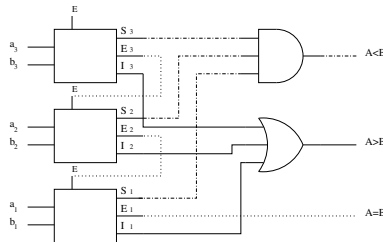
VI.1.3 Le comparateur

Il détecte l'égalité entre deux nombres A et B. Certains circuits permettent également de détecter si A est supérieur ou bien inférieur à B.

VI.1.3.a Comparateur de 2 e.b.

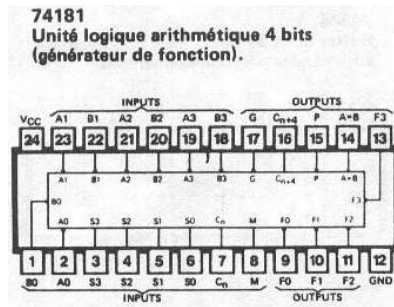
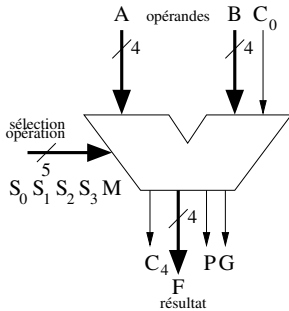
a_i	b_i	E_i	S_i	I_i	
0	0	1	0	0	$E_i = a_i b_i = \overline{a \oplus b}$
0	1	0	0	1	$S_i = a_i \bar{b}_i = \overline{a \cdot b}$
1	0	0	1	0	$I_i = \bar{a}_i b_i = \overline{\bar{a} \cdot b}$
1	1	1	0	0	$D_i = a_i \neq b_i = a \oplus b$

VI.1.3.b Comparateur de 2 nombres

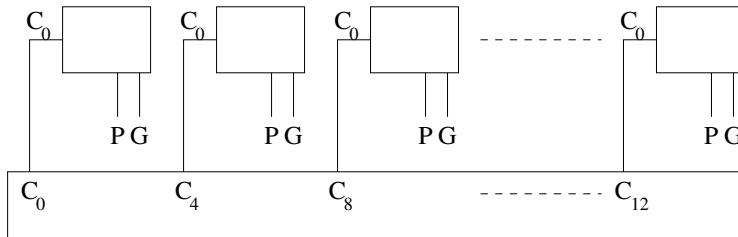


VI.1.4 L'unité arithmétique et logique (UAL)

Utilisée dans pratiquement tous les systèmes informatiques, elle réalise des opérations arithmétiques (addition, soustraction, etc.) et logiques (ET, OU, etc.). C'est un circuit programmable : les relations entre les données en sortie et les données en entrée sont modifiables.



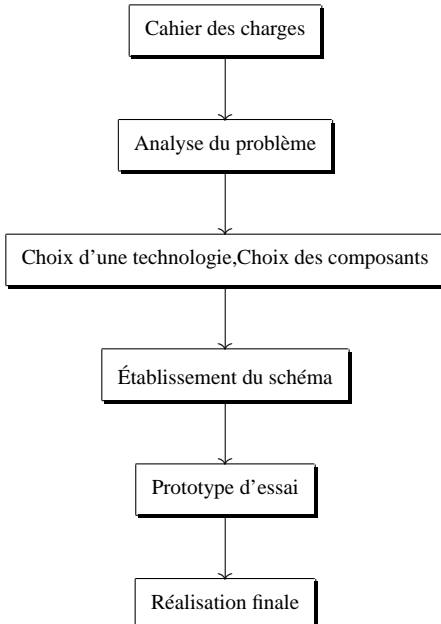
Les sorties P et G servent à la mise en cascade des ALUs, et donc au calcul de retenue anticipée.



Générateur de retenue anticipée

VI.2 Synthèse des circuits combinatoires

VI.2.1 Présentation



Si le nombre de variables mises en œuvre est faible (typiquement inférieur à 10), les circuits sont réalisés directement à l'aide de la table de vérité, éventuellement après simplification de la fonction logique. Dans le cas contraire, la fonction est décomposée en différents blocs fonctionnels analysés séparément.

Le choix des composants utilisés est basé sur différents critères : nombre de boîtiers, coût, disponibilité, points test, complexité des connexions, etc.

Les différents choix sont :

- utilisation de portes simples (OU, ET, NON) ou des portes NON-OU et NON-ET ;
- développement de circuits intégrés (CI) spécialisés. Le problème du coût de développement et de fabrication impose une production en très grandes séries ;
- utilisation de circuits intégrés combinatoires :
 - MUX, DEMUX ;
 - décodeurs ;
 - circuits logiques programmables : PROM, PAL, etc.

VI.2.2 Circuits logiques programmables

VI.2.2.a Introduction

La réalisation pratique d'un système logique dit « câblé » consiste à utiliser les composants CI disponibles sur le marché. Cela oblige le concepteur à décomposer un système donné en blocs fonctionnels proposés par les constructeurs et à optimiser son choix.

L'apparition des circuits adaptables dits « programmables » par le constructeur ou l'utilisateur apporte une solution à ce problème.

VI.2.2.b Structure des circuits logiques programmables

Toute fonction logique de n variables peut se mettre sous la forme d'une somme de produits. Cela implique que toute fonction logique peut être réalisée par l'utilisation d'une structure comportant deux ensembles fonctionnels :

- un ensemble d'opérateurs ET organisés sous forme de matrice permet de générer les produits des variables d'entrée ;
- un ensemble d'opérateurs OU permet de sommer les produits.

La programmation de ces circuits est possible grâce à des fusibles placés à chaque noeud, et consiste à griller les fusibles de manière à supprimer le contact entre les lignes.

1. PROM (*Programmable Read-Only Memory*) ou PLE (*Programmable Logic Element*)

Contrairement au FPLA dont les deux matrices sont programmables (cf. ¶3 page 82), les structures de type PROM voient leur matrice ET figée en usine, formant les 2^n fonctions possibles des n entrées. La matrice OU reste quant à elle entièrement programmable.

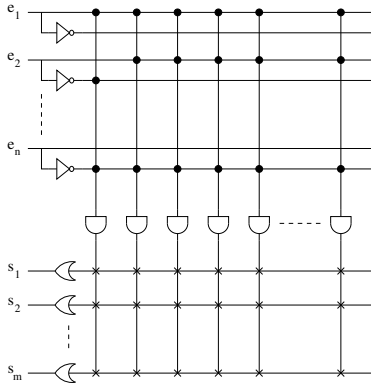
- chaque sortie de la mémoire correspond à une fonction (sortie 3 états) ;
- la matrice ET correspond en fait à un décodeur $n \rightarrow 2^n$ (décodeur d'adresse) ;
- une fonction est réalisée en programmant sa table de vérité, c'est-à-dire en mettant en mémoire la valeur de f pour l'ensemble des combinaisons des entrées.



: interconnexion non programmée

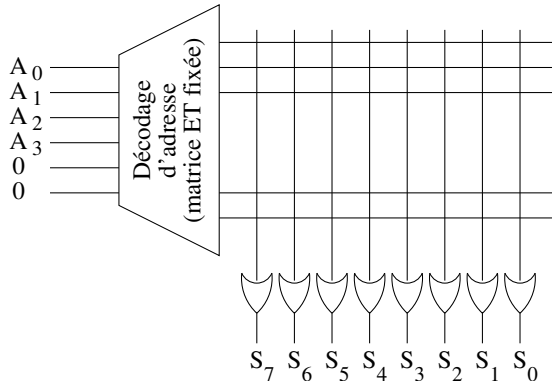


: interconnexion programmée



▷ Exemple 6.10

Réaliser le circuit $N \rightarrow N^2$ (N : nombre codé en DCB sur 4 bits) à l'aide de la PROM suivante (PROM à 6 entrées et 8 sorties → capacité de 2^6 64 octets) :

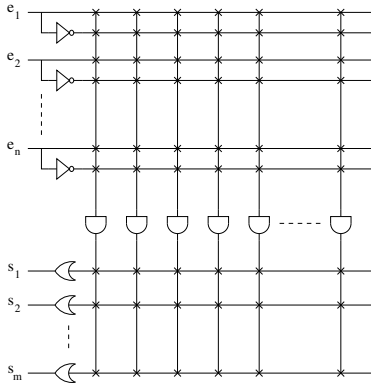


2. PAL (Programmable Array Logic)

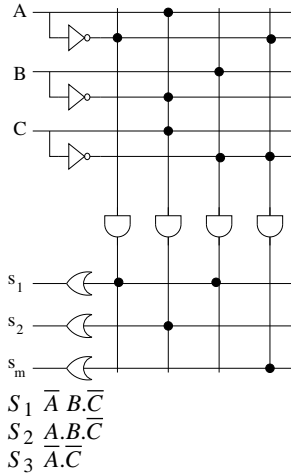
La structure des PAL est opposée à celle des PROM : la matrice OU est fixée alors que la matrice ET est programmable.



Les circuits PAL existent également en logique séquentielle.



▷ Exemple 6.11



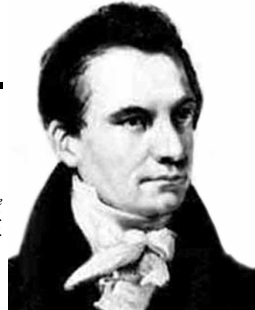
VI.2.3 Programmation des circuits logiques programmables

- Les PROMs et PALs se programment assez facilement avec des « programmeurs universels » standards dans lesquels est incorporé un module spécifique pour chaque constructeur ;
- les FPLAs nécessitent des programmeurs plus sophistiqués à cause des doubles matrices à programmer.

Chapitre VII

Fonctions et opérateurs arithmétiques

Charles Babbage
★ 26 déc. 1791, Teignmouth, R.-U.
† 1871, London, R.-U.



«... I was sitting in the rooms of the Analytical Society, at Cambridge, my head leaning forward on the table in a kind of dreamy mood, with a table of logarithms lying open before me. Another member, coming into the room, and seeing me half asleep, called out, Well, Babbage, what are you dreaming about” to which I replied “I am thinking that all these tables” (pointing to the logarithms) “might be calculated by machinery.” »

(Charles Babbage)

[Exercices sur les systèmes combinatoires]

►Exercice 2.1

Développer et simplifier algébriquement les expressions booléennes suivantes :

- $F_1 (x \bar{y}).(x z)$
- $F_2 (x.\bar{y} z).(x \bar{y}).z$
- $F_3 (x y).z \bar{x}.\bar{y} z \bar{y}$
- $F_4 bd \bar{c}d \bar{a}b \bar{c}d \bar{a}b \bar{c}$
- $F_5 \bar{a}\bar{b} \bar{c} b.(a \bar{c}) \bar{a} \bar{b} \bar{a}\bar{c}$

►Exercice 2.2

Faire le schéma des fonctions suivantes avec les portes indiquées :

- $x \bar{a}\bar{b} \bar{c} d$ (3 portes NOR)
- $y \overline{a(b c)}$ (3 portes NAND)
- $z \overline{abc}$ (3 NAND à 2 entrées)
- $f a \oplus b$ (4 NAND à 2 entrées)

►Exercice 2.3

Simplifier les expressions logiques suivantes :

- $F_1 ab \oplus abcd$
- $F_2 a \oplus (a b)$
- $F_3 a (a \oplus b)$
- $F_4 (a \oplus b) \oplus (a \oplus c)$
- $F_5 (a \oplus b) \oplus (\bar{a} \bar{b})$

►Exercice 2.4

Chercher les formes canoniques des expressions suivantes :

- $F_1 a \oplus (b c)$
- $F_2 (a \bar{c}).\bar{b} (\bar{a} c).b$

►Exercice 2.5

Montrer algébriquement que $\bar{a}\bar{b} bc \bar{a}\bar{c} \bar{a}b \bar{b}\bar{c} \bar{a}c$. Vérifier à l'aide d'un diagramme de Karnaugh.

► **Exercice 2.6**

Simplifier cette expression à l'aide d'un diagramme de Karnaugh :

$$F \bar{a}(b \oplus c) \bar{a}c\bar{d} \bar{a}d(b \oplus \bar{c}) (a \oplus d)\bar{b}\bar{c} \bar{a}c\bar{b} \oplus \bar{d}$$

Faire le schéma avec 2 portes dont un XOR.

► **Exercice 2.7**

Une fonction $f(a, b, c, d)$ est incomplètement définie. On code ses états sur le mot binaire $abcd$, a représentant le poids fort. La fonction est vraie pour les états 0, 1, 3, 4, 6, A, B ; elle est fautive pour les états 7, 8, D, E. Tracer le diagramme de Karnaugh. Simplifier la fonction en vue d'une réalisation en portes NAND. Même question avec des portes NOR. Quelle est la meilleure solution ?

1 4 6 7 3 9 2 3



Troisième partie

Les circuits séquentiels

2 2

0 1 2 3 4 5 6 7 8 9



25 20 MAIN DROITE 15 10

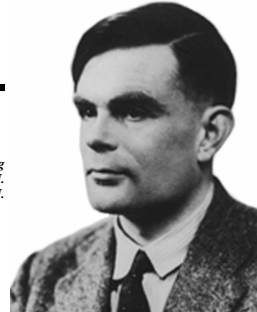
1 2 3 4 5 6 7 8 9 0



Chapitre VIII

Les bascules

Alan Mathison Turing
★ 23 juin 1912, Londres, R.-U.
† 8 juin 1954, R.-U.



« [A universal machine] ... which can be made to do the work of any special-purpose machine, that is to say to carry out any piece of computing, if a tape bearing suitable "instructions" is inserted into it. »

(Alan M. Turing, 1936, à propos de la « machine de Turing »)

VIII.1 Introduction

Circuit séquentiel : circuit dont l'état des sorties dépend non seulement des entrées mais également de l'état antérieur des sorties. Ces circuits doivent donc être capables de mémoriser.

▷ **Exemple 8.1**

$$1 \begin{cases} M 0 \\ A 0 \end{cases} \rightarrow L 0$$

$$3 \begin{cases} M 0 \\ A 0 \end{cases} \rightarrow L 1$$

$$5 \begin{cases} M 0 \\ A 0 \end{cases} \rightarrow L 0$$

$$2 \begin{cases} M 1 \\ A 0 \end{cases} \rightarrow L 1$$

$$4 \begin{cases} M 0 \\ A 1 \end{cases} \rightarrow L 0$$

Dans un tel système, à une même combinaison des variables d'entrée ne correspond pas toujours la même valeur à la sortie (3 et 5). La fonctionnalité dépend de l'ordre des opérations (ordre de déroulement des séquence) → système séquentiel.

Les fonctions séquentielles de base sont :

- mémorisation ;
- comptage ;
- décalage.

Les circuits séquentiels fondamentaux sont :

- bascules (3 types) ;
- compteurs ;
- registres ;
- RAM (Random Access Memory).

Ces circuits peuvent travailler soit en mode synchrone, soit en mode asynchrone :

- mode asynchrone À tout moment, les signaux d'entrée peuvent provoquer le changement d'état des sorties (après un certain retard qu'on appelle « temps de réponse ». Ces systèmes sont difficiles à concevoir et à dépanner.
- mode synchrone Le moment exact où les sorties peut changer d'état est commandé par un signal d'horloge (train d'ondes carrées ou rectangulaires). Les changements d'état s'effectuent tous pendant une transition appelée « front » (montant ou descendant).

La majorité des systèmes numériques séquentiels sont synchrones même si certaines parties peuvent être asynchrone (ex. : reset).

Les avantages principaux du mode synchrone sont :

- préparer les entrées sans perturber les sorties ;
- protéger des parasites survenant en entrée.

Les bascules que l'on peut considérer comme des mémoires élémentaires, sont les briques de base des circuits séquentiels.

Ce sont les circuits de mémorisation les plus répandus dans les systèmes numériques en raison de leur rapidité de fonctionnement, de la facilité d'écriture et de lecture d'information, et de la grande simplicité de leur interconnexion avec des portes logiques.

On trouve deux grandes familles de bascules :

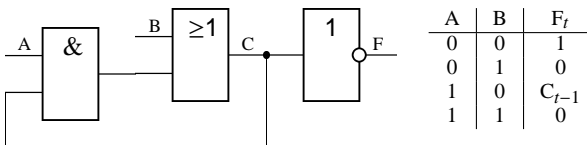
- bascules de mémorisation : elles possèdent les commandes de mise à zéro, mise à un, mémorisation ;
- bascules de comptage : elles possèdent en outre une commande de changement d'état.

VIII.2 Point mémoire

La principale différence entre un système séquentiel et un système combinatoire est que lorsque l'on présente plusieurs fois de suite un même vecteur d'entrée à un système séquentiel, celui-ci – contrairement au système combinatoire – ne délivre pas nécessairement un le même vecteur de sortie à chaque fois.

En d'autres termes, l'état de la sortie d'un système séquentiel dépend non seulement de l'état des variables d'entrée, mais également du paramètre « temps », lequel paramètre est la plupart du temps concrétisé par « l'état antérieur » du système.

Soient le circuit et sa table de vérité associée suivants :



La sortie de la fonction F ci-dessus est dépendante d'une variable interne C. On peut en effet constater que l'état de la variable C dépend de l'état des entrées A Et B, mais également de son état antérieur : C mémorise donc liée aux entrées appliquées antérieurement au circuit. On constate sur ce circuit que l'effet de mémorisation est dû à la boucle de rétroaction présente entre la sortie du OU et l'entrée du ET. À cette boucle est associée la variable C qui constitue le point mémoire.

Définition 8.1

Circuit séquentiel : un circuit séquentiel est un système bouclé permettant la conservation d'un état dépendant de la valeur des variables d'entrée ainsi que de l'état antérieur du système.

La bascule constitue le système séquentiel de base et permet de mémoriser un élément d'information élémentaire appelé bit.

► Exercice 3.1

Quel sera l'état de sortie du système F à l'issue des deux séquences (00, 10) et (01,10) ?

Nous avons brièvement présenté en introduction de ce chapitre ce qu'étaient les systèmes séquentiels synchrones et asynchrones. Une autre façon de décrire ces systèmes est donnée par les définitions 8.2 et 8.3 suivantes :

Définition 8.2

Système asynchrone : un système séquentiel est asynchrone si à partir de l'instant où on applique un vecteur d'entrée, son évolution est incontrôlable de l'extérieur.

Définition 8.3

Système synchrone : un système séquentiel est synchrone si son évolution est contrôlable de l'extérieur par un signal d'horloge.

VIII.3 Bascule RS

La bascule RS est le circuit séquentiel le plus simple. C'est une bascule asynchrone, et toutes les autres bascules, synchrones ou asynchrones, reposent sur cette bascule.

Son rôle consiste à mémoriser une information fugitive, selon le fonctionnement suivant : une apparition, même fugitive, de S entraîne un état stable $Q=1$, et une apparition, même fugitive, de R entraîne un état stable $Q=0$.

Symbole

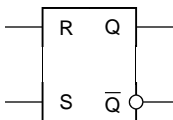


Tableau de Karnaugh

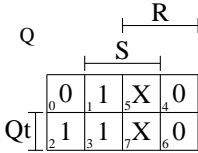


Diagramme temporel

Quand une impulsion est appliquée à 1 entrée pour imposer un certain état à la bascule, celle-ci demeure dans cet état, même après que l'impulsion ait disparu. Q garde son état lorsque S passe de 1 à 0 et lorsque R passe de 1 à 0.

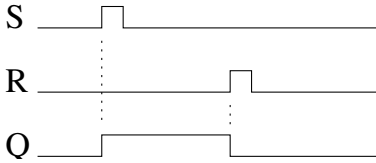


Table de vérité

S	R	Qt	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

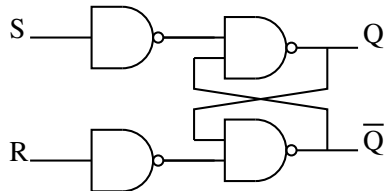
S	R	Q
0	0	Q
0	1	0
1	0	1
1	1	X

→ mémorisation
→ mise à 0
→ mise à 1
→ interdit

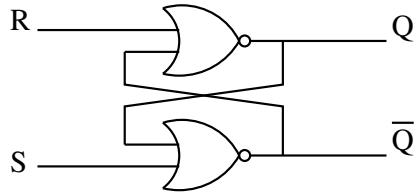
Réalisation

Si $X \rightarrow Q\bar{S}R$, Q Les états indéterminés sont forcés à 1 : la bascule est dite à enclenchement prioritaire.

⇒ somme de produit ⇒ réalisation à l'aide de portes NAND.



Si $X=0 \rightarrow Q = \bar{R} \cdot (S+Q)$. Les états indéterminés sont forcés à 0 : la bascule est dite à déclenchement prioritaire.
 \Rightarrow produit de sommes \Rightarrow réalisation à l'aide de portes NOR.



► **Remarque 8.1**

Dans les deux cas, lorsqu'on passe de l'état $(R,S)=(1,1)$ à $(R,S)=(0,0)$ en passant soit par l'état stable correspondant à $(R,S)=(1,0)$, soit par l'état stable correspondant à $(R,S)=(0,1)$, selon la rapidité relative des passages $0 \rightarrow 1$ de chacun des signaux, alors la sortie peut prendre aussi bien l'état $Q=1$ que $Q=0$.

\Rightarrow il faut donc interdire la combinaison $R=S=1$ afin de lever l'ambiguïté pour un état $R=S=0$ venant après un état $R=S=1$.

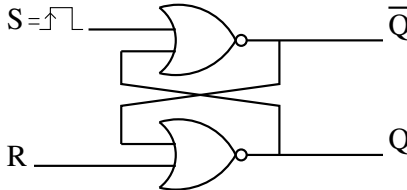
Fonctionnement de la bascule avec des NOR

- quand $R=S=0$, il y a deux possibilités et nous verrons que l'état pris par la bascule dépend des valeurs appliquées précédemment aux entrées :

- si $Q=0 \xrightarrow{S=0} \bar{Q}=1$ et $Q=0$) \rightarrow memorisation
- si $Q=1 \xrightarrow{S=0} \bar{Q}=0$ et $Q=1$

- Examinons si $S \uparrow$ et $R=0$

Si $Q=0$ à l'arrivée de l'impulsion sur S, alors $S=1 \rightarrow \bar{Q}=0 \rightarrow Q=1$



Si $Q=1$ à l'arrivée de l'impulsion sur S, alors $S=1 \rightarrow \bar{Q}=0 \rightarrow Q$ reste à 1

\Rightarrow l'application d'une impulsion de niveau haut sur S place la bascule dans l'état $Q=1$.

\rightarrow opération de mise à 1 \rightarrow SET

- Si on applique $R \uparrow$ et $S=0$

- Si $Q=0 \xrightarrow{R \uparrow} Q=0 \rightarrow \bar{Q}=1$ } $\rightarrow R \uparrow$
- Si $Q=1 \xrightarrow{R \uparrow} Q=0 \rightarrow \bar{Q}=1$

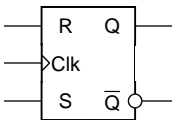
- ⇒ l'application d'une impulsion de niveau haut sur R place la bascule dans l'état Q 0.
- opération de mise à 0 → RESET
- R S $\overline{1}$
 - ⇒ Q \overline{Q} 0
 - condition indésirable, puisque \overline{Q} et Q doivent être l'inverse l'un de l'autre
 - de plus, incertitude lorsque S et R reviennent à 0
 - R S 1 ne doit pas servir

L'avantage principal (unique ?) de la bascule RS est sa simplicité. Ses principaux inconvénients sont le fait qu'elle soit asynchrone, sa sensibilité aux parasites (tout bruit présent sur l'une des entrées de la bascule RS peut modifier l'état de la sortie), et le fait qu'il existe un état interdit pour R=S=1.

VIII.4 Bascule RS synchrone ou bascule RSH

La bascule RSH¹ est une bascule RS synchronisée par un signal d'horloge H. Lorsque H est au niveau bas, la bascule fonctionne comme une mémoire, et lorsque H est au niveau haut, la bascule fonctionne comme une bascule RS classique, et conserve donc les états interdits pour R=S=1.

Symbole



La sortie est indiquée est vaut Q_N avant le front de l'horloge et Q_{N+1} après le front de l'horloge.

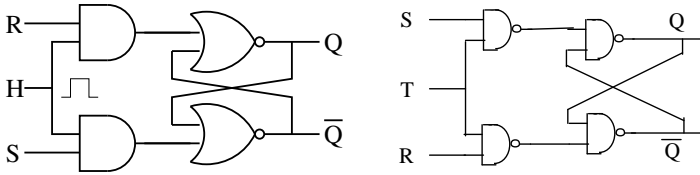
S et R n'influencent Q que lorsque l'horloge est au niveau haut.

Table de vérité

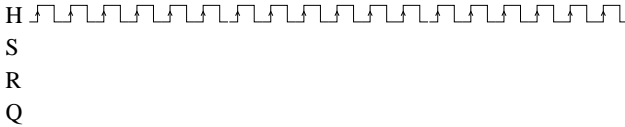
S	R	H	Q_{N+1}
X	X	0	Q_N
0	0	1	Q_N
0	1	1	1
1	0	1	0
1	1	X	X

Réalisation

¹La bascule RSH est également appelée bascule RST; on préférera néanmoins le terme RSH, plus explicite.



▷ Exemple 8.2



L'avantage de la bascule RSH par rapport à la bascule RS est sa sensibilité moindre aux parasites. Comme la bascule n'est sensible au bruit que lorsque l'horloge est au niveau haut, plus les états haut de l'horloge seront brefs, moins la bascule sera sensible.

VIII.5 Bascule à verrouillage (D-latch)

La D-latch est une bascule RST pour laquelle on n'a conservé que les deux combinaisons $RS=(0,1)$ et $RS=(1,0)$. La D-latch a une seule entrée, nommée D.

Symbole

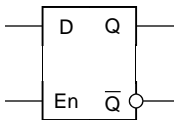
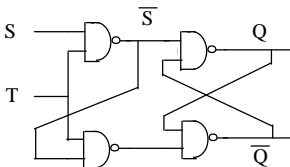


Table de vérité

D_N	Q_{N+1}
0	0
1	1

Réalisation



Fonctionnement

- quand $Clk=0 \rightarrow$ l'entrée D n'a aucun effet (mémoire);
- quand $Clk=1 \rightarrow Q$ suit les changements de D \rightarrow la bascule est transparente.

La D-latch n'a pas d'état interdit et est transparente sur le niveau haut de l'horloge.

► Remarque 8.2

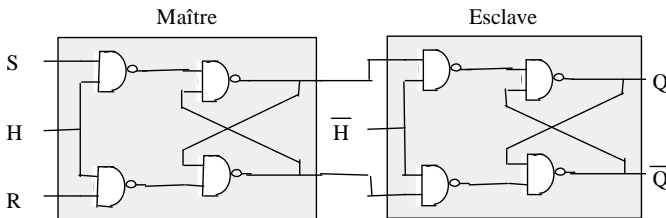
Notez l'absence du symbole \triangleright sur l'entrée d'horloge.

VIII.6 Bascules maître-esclave

Les bascules maître-esclaves permettent de diminuer la sensibilité aux parasites en minimisant la période de transparence. La nature des bascules maître-esclave vient du fait que deux bascules RST montées en cascade et commandées par deux horloges en opposition de phase réalisent la même fonction qu'une seule bascule. La différence tient seulement au fait que la bascule ne fonctionne plus sur le niveau haut de l'horloge, mais sur son front descendant

- ☞ sur le niveau bas de l'horloge, le premier étage (maître) fonctionne en mode « mémorisation », et le deuxième étage (esclave) est en mode RS ;
- ☞ sur le niveau haut de l'horloge, le maître fonctionne en mode RS, et l'esclave est dans l'état « mémorisation »

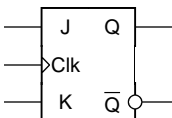
La période pendant laquelle la bascule est sensible aux parasites se résume donc à la durée de commutation de l'horloge du niveau haut au niveau bas (front descendant).



VIII.7 Bascule JK

Les bascules JK sont des bascules maître-esclave fonctionnant seulement en mode synchrone. Elles sont plus polyvalentes que les bascules RS, car elles n'ont pas d'état ambigu et $R S 1 \rightarrow Q_{N+1} \overline{Q}_N$.

Symbole



$$Q_{N+1} = J \cdot \overline{Q_N} + \overline{K} \cdot Q_N$$

Tableau de Karnaugh

		J			
		K			
Qt	Q	0	1	0	1
	0	0	1	0	1
1	0	1	0	0	1

Sachant que les sorties sont toujours complémentaires, leur rebouclage sur les entrées élimine l'état interdit. Il n'y a pas d'inconvénient à ce rebouclage car les sorties de l'esclave ne change d'état que lorsque le maître est bloqué. Les bascules JK sont très courantes dans les systèmes numériques

Cette bascule fonctionne toujours sur les fronts descendant.

Réalisation

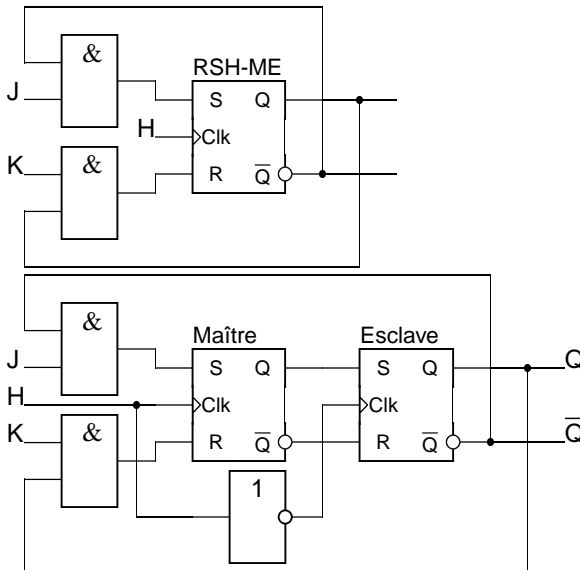


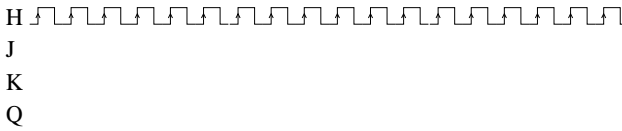
Table de vérité

J	K	H	Q_{N+1}		J	K	Q_N	H	Q_{N+1}
X	X	—	Q_N		X	X	X	—	Q_N
0	0	ϕ	Q_N	mémorisation	X	0	1	ϕ	1
1	0	ϕ	1	forçage à 1	X	1	1	ϕ	0
0	1	ϕ	0	forçage à 0	0	X	0	ϕ	0
1	1	ϕ	$\overline{Q_N}$	commutation	1	X	0	ϕ	1

► **Remarque 8.3**

Pour que le basculement fonctionne, il faut avoir H très étroite, autrement il y a rebasculément.

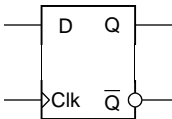
▷ **Exemple 8.3**



VIII.8 Bascule D synchrone

La bascule D est une bascule maître-esclave conçue sur le même principe que la JK. La bascule D est une bascule n’ayant qu’une seule entrée nommée D.

Symbole

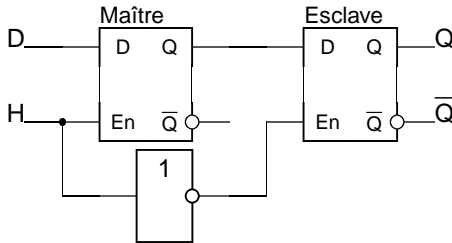


Le symbole de la bascule D est identique à celui de la D-latch à ceci près que l’entrée d’activation est remplacée par une entrée d’horloge, qui dispose donc du symbole associé.

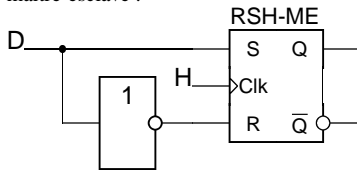
Table de vérité

H	D_N	Q_{N+1}
ϕ	1	1
ϕ	0	0

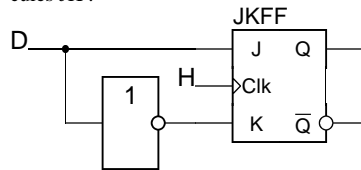
Q_{N+1} prend la valeur de D_N après le front actif : $Q_{N+1} = D_N$
 C’est une bascule de recopie : on l’emploie seulement en synchrone.

Réalisation

Réalisation à partir de bascules RSH maître-esclave :



Idem pour la réalisation à partir de bascules JK :

**► Remarque 8.4**

La sortie Q n'est égale à l'entrée D qu'à des moments bien précis \rightarrow le signal Q est différent du signal D .

La bascule D fonctionne sur fronts d'horloge. En fait, la donnée d'entrée D est transférée à travers le maître lors du front montant et à travers l'esclave lors du front descendant. Pour fonctionner, cette bascule nécessite donc les deux front d'horloge. Différentes structures de bascules D existent, certaines pouvant transférer une donnée en ne recevant qu'un seul front d'horloge.

VIII.9 Bascule T

La bascule T s'obtient par exemple à partir d'une bascule JK dont on a relié les entrées J et K entre elles. Elle est utilisable uniquement en mode synchrone, et ne fonctionne qu'en commutation.

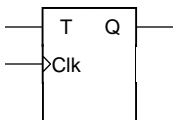
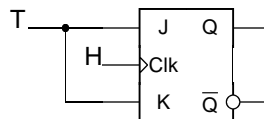
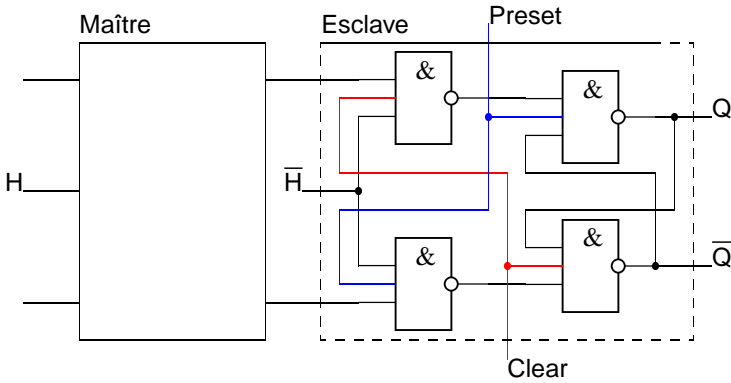
Symbole**Réalisation**

Table de vérité

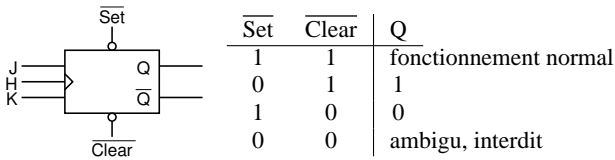
T	Q_{N+1}
0	Q_N
1	$\overline{Q_N}$

VIII.10 Entrées prioritaires asynchrones des bascules

La plupart des bascules synchrones possèdent des entrées prioritaires asynchrones. Elles agissent indépendamment de l'horloge et des entrées synchrones des bascules. Elles servent à forcer, à tout moment, la mise à 1 ou à 0 de la bascule, quelles que soient les conditions d'entrée. Elle agissent sur l'étage esclave des bascules.



▷ Exemple 8.4



Les entrées asynchrones peuvent être vraies à l'état bas (cas le plus fréquent) ou à l'état haut. En général, on applique juste une impulsion à ces entrées pour faire une initialisation.

Désignations synonymes :	Clear	RAZ	[c]et	DC clear
	Preset	RAU	Set	DC set

► **Remarque 8.5**

Les entrées synchrones sont des niveaux de tension continue

VIII.11 Paramètres temporels des bascules

Pour qu'une bascule fonctionne correctement, il est nécessaire que le signal présent sur les entrées de la bascule (D ou JK) soit stabilisé depuis un certain temps lorsque le front d'horloge actif intervient (temps de « *setup* ») et reste stable pendant un certain temps après ce front d'horloge (temps de « *hold* » ou de maintien).

D'autre part, la commutation des sorties d'une bascule se fait avec un certain temps de retard par rapport au signal qui a produit cette commutation (Horloge, Reset ou Preset). Ces retards peuvent être différents selon le signal qui a produit la commutation, mais également selon que la commutation du signal de sortie est montante ou descendante. Ces retards seront notés T_{pLH} et T_{pHL} pour « Temps de Propagation *Low High* » et « Temps de Propagation *High Low* ».

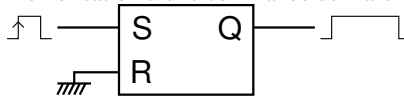
VIII.12 Applications des bascules

VIII.12.1 Mémoire

→ mémorisation d'une information fugitive

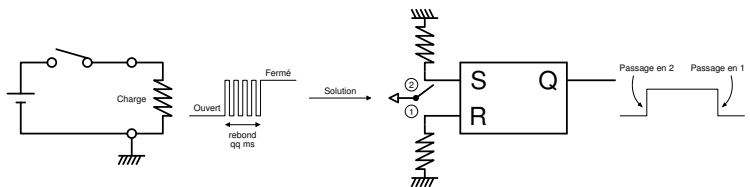
► **Exemple 8.5**

Mémorisation d'une commande de marche



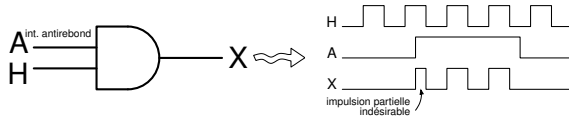
VIII.12.2 Antirebond pour commutateur

► **Exemple 8.6**

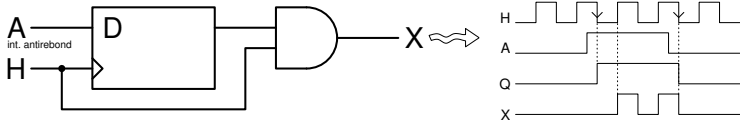


VIII.12.3 Synchronisation

▷ Exemple 8.7

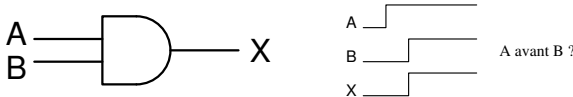


Solution :

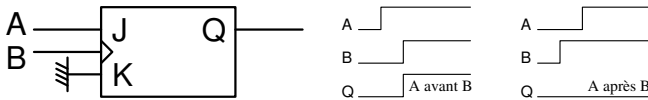


VIII.12.4 Détection d'une séquence d'entrée

▷ Exemple 8.8



Solution :



→ détection du sens de rotation d'un moteur.

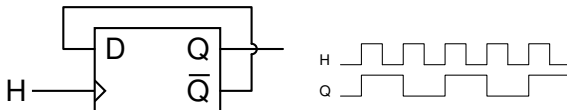
VIII.12.5 Division de fréquence

La division de fréquence par 2 (et donc 2^N) peut être réalisée facilement à l'aide des différents registres.

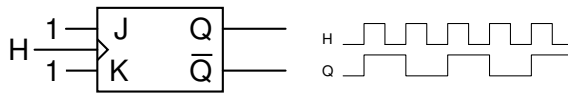
Bascule D

$$D_N = Q_{N+1}$$

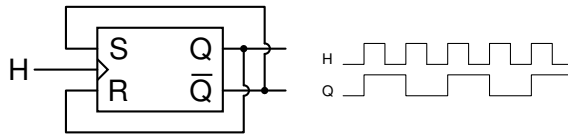
$$\text{On veut } Q_{N+1} = \bar{Q}_N \Rightarrow D_N = \bar{Q}_N$$



Bascule JK



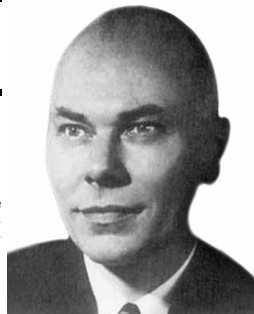
Bascule RS



Chapitre IX

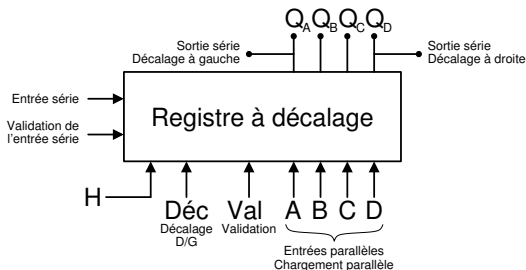
Registres : stockage et transfert de données

Howard Hathaway Aiken
★ 9 mars 1900, Hoboken, E.-U.
† 14 mars 1973, St Louis, E.-U.



[En 1964 Aiken reçoit le Harry M Goode Memorial Award, une médaille et \$2,000 offert par la Computer Society] for his original contribution to the development of the automatic computer, leading to the first large-scale general purpose automatic digital computer.

Registre : ensemble de n bascules synchronisées permettant de stocker momentanément une information sur n bits.



IX.1 Définition

Un registre est un circuit constitué de n bascules synchronisées permettant de stocker temporairement un mot binaire de n bits en vue de son transfert dans un autre circuit (pour traitement, affichage, mémorisation, etc.)

Le schéma d'un tel système comporte autant de bascules (de type D) que d'éléments binaires à mémoriser. Toutes les bascules sont commandées par le même signal d'horloge.

Moyennant une interconnexion entre les cellules (les bascules D), un registre est capable d'opérer une translation des chiffres du nombre initialement stocké. Le dépla-

cement s'effectue soit vers la droite soit vers la gauche. Le registre est alors appelé « registre à décalage ».

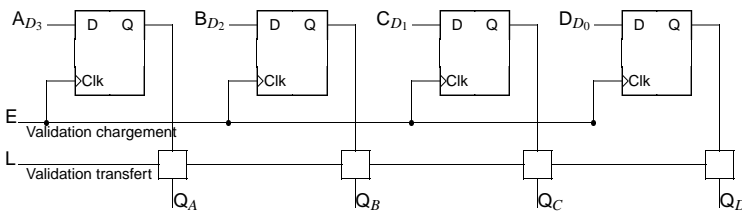
Applications :

- conversion série-parallèle d'une information numérique ;
- opérations de multiplications et divisions par deux ;
- ligne à retard numérique ;
- mémoires à accès séquentiel

« Registre universel » : il résume les différentes entrées et sorties d'un registre à décalage procurant tous les modes de fonctionnement possibles.

IX.2 Registre de mémorisation : écriture et lecture parallèles

Tous les bits du mot à traiter sont écrits (entrée écriture E=1), ou lus, (entrée lecture L=1), simultanément.



→ stockage en parallèle et transfert en parallèle d'un mot de 4 bits.

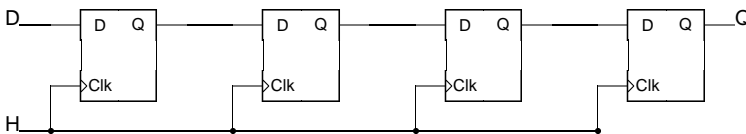
IX.3 Registres à décalage

Comme son nom l'indique, un registre à décalage consiste à décaler bit par bit un mot binaire soit vers la gauche, soit vers la droite. Le registre à décalage peut être à écriture et à lecture série ou parallèle.

► Remarque 9.1

Un registre à décalage à droite peut être utilisé comme un diviseur par 2 alors qu'un registre à décalage à gauche peut être utilisé comme un multiplieur par 2.

IX.3.1 Registre à écriture série et lecture série

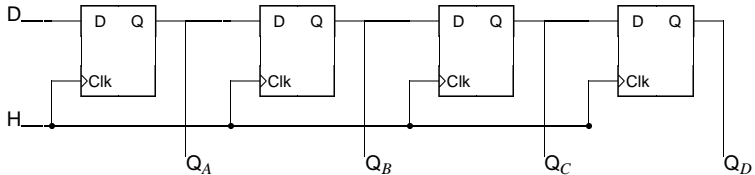


Après 4 pulsations de CLK, les 4 bits sont entrés dans le registre.

Après 4 autres cycles d'horloge, les 4 bits sont déplacés vers la sortie.

Leur application est essentiellement le calcul arithmétique binaire. CLK est alors l'entrée de décalage.

IX.3.2 Registre à écriture série et lecture parallèle

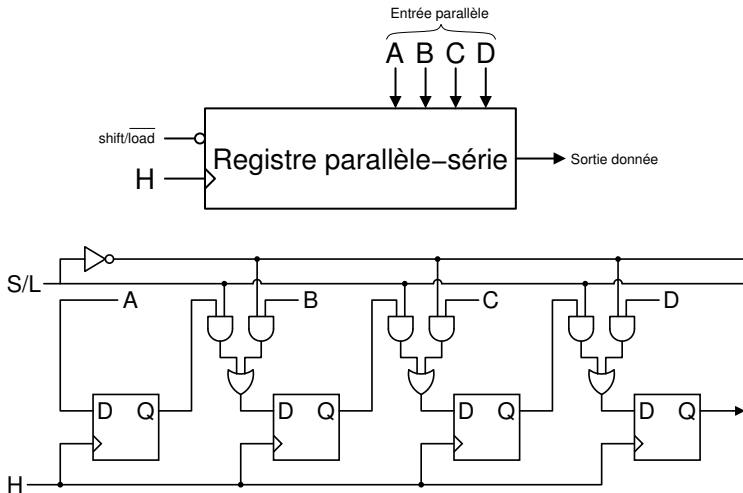


Lorsque l'entrée est stockée, chaque bit apparaît simultanément sur les lignes de sortie.

Le registre à décalage est utilisé comme convertisseur série-parallèle. Il est nécessaire à la réception lors d'une transmission série.

IX.3.3 Registre à écriture parallèle et lecture série

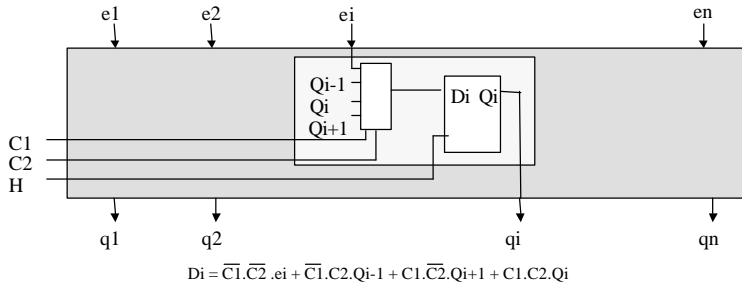
Utilisé comme convertisseur parallèle-série, il est nécessaire à l'émission lors d'une transmission série.



IX.4 Registre universel

Le registre universel permet quatre modes de fonctionnement commandés par deux variables S_1 et S_2 .

S_1	S_2	Mode
0	0	Chargement parallèle
0	1	Décalage à droite
1	0	Décalage à gauche
1	1	Inhibition de l'horloge

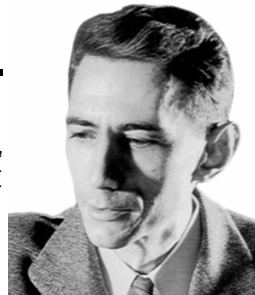


Ces entrées de sélection S_1 et S_2 sont en fait les entrées de sélection de multiplexeurs connectés aux entrées des bascules. Ces multiplexeurs à quatre entrées permettent donc quatre modes de fonctionnement : l'entrée D de chaque bascule est ainsi fonction du mode de fonctionnement désiré.

Chapitre X

Les compteurs

Claude Elwood Shannon
★ 30 avr. 1916, Gaylord, E.-U.
† 24 fév. 2001, Medford, E.-U.



« The most important results [mostly given in the form of theorems with proofs] deal with conditions under which functions of one or more variables can be generated, and conditions under which ordinary differential equations can be solved. Some attention is given to approximation of functions (which cannot be generated exactly), approximation of gear ratios and automatic speed control. »

(Claude E. Shannon, Mathematical theory of the differential analyzer, 1941)

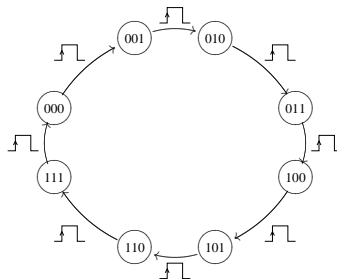
Définition 10.1

Compteur : un compteur est un circuit séquentiel comportant n bascules décrivant au rythme d'une horloge un cycle de comptage régulier ou quelconque d'un maximum de 2^n combinaisons.

Définition 10.2

État, Modulo : la combinaison de sortie d'un compteur est appelé état, et le nombre d'états possibles d'un compteur est appelé modulo.

Un compteur modulo N passera donc successivement par N états. Un compteur binaire naturel comptera donc de 0 à $N-1$. Le graphe suivant présente les différents états parcourus par un compteur modulo 8.



X.1 Compteur asynchrone (à propagation)

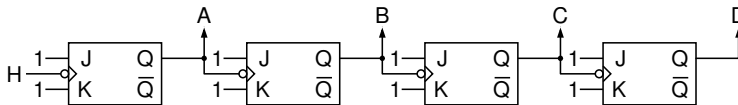
Nous avons vu dans la section §VIII.12.5 page 104 comment réaliser une division par deux à l'aide de bascules JK. En cascade des bascules JK montées en diviseurs de fréquence, on peut donc réaliser un compteur dont le modulo dépendra du nombre de bascules.

X.1.1 Compteur asynchrone à cycle régulier

▷ **Exemple 10.1**

Compteur asynchrone à 4 bits (compte de 0 à 15).

X.1.1.a Réalisation à l'aide de bascules JK



La sortie de chaque bascule agit comme le signal d'horloge de la suivante.

Fonctionnement

- $J=K=1$; toutes les bascules commutent sur des fronts descendants ;
- la bascule A commute à chaque front descendant du signal d'horloge ;
- la sortie de la bascule 1 sert d'horloge pour la bascule 2 → B commute chaque fois que A passe de 1 à 0 ;
- de la même manière, C commute lorsque B passe de 1 à 0, et D commute lorsque C passe de 1 à 0.

X.1.1.b Table d'implication séquentielle

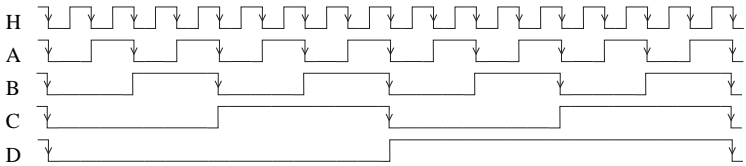
Elle montre les états binaires pris par les bascules après chaque front descendant.

N°	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
⋮	⋮	⋮	⋮	⋮
15	1	1	1	1
16	0	0	0	0
⋮	⋮	⋮	⋮	⋮

Si on imagine que DCBA représente un nombre binaire, le compteur réalise la suite des nombres binaires allant de 0000 à 1111 (soit de 0 à 15).

A près la 15^{ème} impulsion, les bascules sont dans la condition 1111. Quand la 16^{ème} impulsion arrive, le compteur affiche 0000 : un nouveau cycle commence.

X.1.1.c Chronogramme



→ chaque bascule divise par deux la fréquence d'horloge qui alimente son entrée

$$CLK : f_D = \frac{f_{initiale}}{16}$$

Application : avec ce genre de circuit, on peut diviser la fréquence initiale par n'importe quelle puissance de 2.

X.1.1.d Modulo

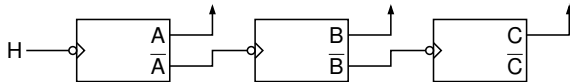
- c'est le nombre d'état occupés par le compteur pendant un cycle complet ;
- le modulo maximal d'un compteur à n bits (n bascules) est 2^n ;
- ex. : compteur 4 bits → 16 états distincts → modulo 16.

X.1.2 Décompteurs asynchrones

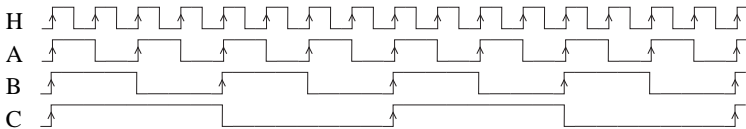
Il suffit de piloter chaque entrée CLK des bascules au moyen de la sortie complémentée de la bascule précédente.

▷ **Exemple 10.2**

Décompteur modulo 8



Chronogramme :



X.1.3 Compteur asynchrone à modulo $N 2^n$ (à cycle régulier)

X.1.3.a Méthode

Pour réaliser un compteur ou un décompteur dont le cycle n'est pas une puissance de 2, la seule solution est d'agir sur l'entrée « Clear » lorsque la combinaison correspondant au modulo du compteur se produit sur les sorties de celui-ci.

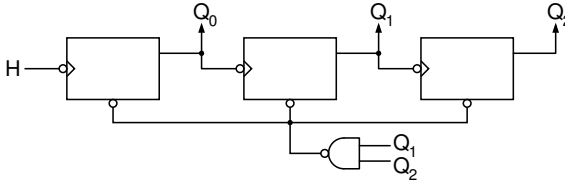
Ainsi, pour $2^{N-1} N 2^N$, on réalise un compteur modulo 2^n (avec n bascules), puis on raccourcit le cycle en jouant sur les entrées RAZ des bascules.

▷ Exemple 10.3

Compteur asynchrone modulo 6 : $2^2 6 2^3 \rightarrow$ on réalise un compteur modulo 3 avec 3 bascules, et on ramène le compteur à 000 dès que $Q_2 Q_1 Q_0 110$.

\rightarrow dès que la sortie de la porte NAND passe à 0, les bascules sont forcées à 0 : le compteur se remet à compter à partir de 0.

\Rightarrow le compteur réalisé compte de 000 à 101 (de 0 à 5) puis recommence un nouveau cycle \rightarrow modulo 6



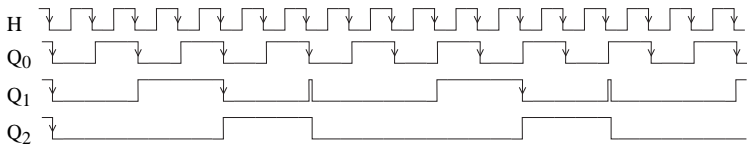
X.1.3.b Table d'implication séquentielle

N°	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0

$\rightarrow 0 0 0$

$Q_2 Q_1 Q_0 110$ est un état temporaire. Il existe mais pendant une durée très courte. C'est un état indésirable que l'on nomme parfois *glitch*.

X.1.3.c Chronogramme



► **Remarque 10.1**

Les sorties Q_2 et Q_1 ne sont pas des ondes carrées.

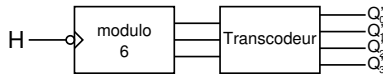
X.1.4 Comptage asynchrone dans un ordre quelconque (cycle irrégulier)

1^{ère} méthode

On réalise un compteur de même modulo, puis on transcode ses sorties pour obtenir le cycle demandé.

► **Exemple 10.4**

Cycle 2, 5, 6, 8, 4, 10



N°	Q_2	Q_1	Q_0	Q'_3	Q'_2	Q'_1	Q'_0	
0	0	0	0	0	0	1	0	→ 2
1	0	0	1	0	1	0	1	→ 5
2	0	1	0	0	1	1	0	→ 6
3	0	1	1	1	0	0	0	→ 8
4	1	0	0	0	1	0	0	→ 4
5	1	0	1	1	0	1	0	→ 10

2^{ème} méthode

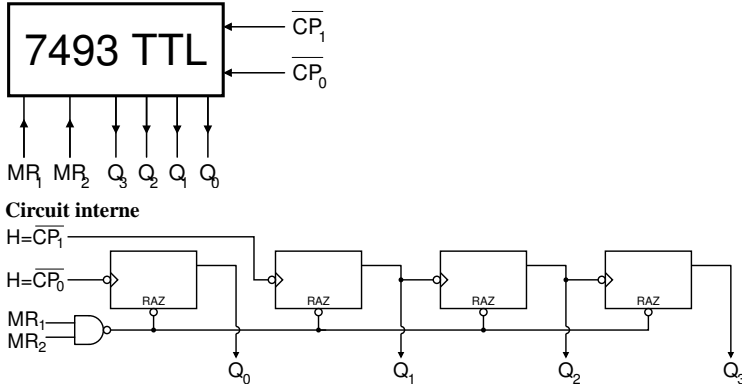
Utilisation des entrées RAZ et RAU.

► **Exemple 10.5**

Cycle 0, 1, 2, 3, 5, 6, 8, 9, 11, 12, 15 : on réalise un compteur modulo 16 et on agit sur les RAU pour sauter les étapes.

X.1.5 Exemple de CI

Il existe de nombreuses puces en technologies TTL et CMOS. Parmi les plus populaires on trouve en TTL le 7493 qui est un compteur 4 bits, et en CMOS le 4024 qui est un compteur 7 bits.



MR → Master Reset.

X.1.6 Inconvénients des compteurs asynchrones

Chaque bascule introduit un retard de D_p ($D_p=25ns$). Comme les bascules ne commutent pas sur le même signal d'horloge, les retards s'additionnent : à la $n^{ième}$ bascule, on a un retard T_m de $n \times D_p$.

Ainsi, la fréquence maximum de fonctionnement F_H d'un compteur modulo n , constitué de n bascules de délai de propagation D_p dépend du nombre de bascules du compteur et donc du modulo du compteur. Cette fréquence peut être établie comme suit :

- $T_m D_p \times n$: Délai de propagation du compteur
- $T_H 2 \times T_m$: Période min de l'horloge
- $F_H 1/(2 \times T_m)$: Fréquence max de l'horloge $1/(2 \times n \times D_p)$

L'accumulation des retards des bascules implique une utilisation du compteur limitée en fréquence, particulièrement lorsque le nombre de bits est élevé, puisque le nombre de bascules augmente en même temps que le nombre de bits.

Les fronts des signaux appliqués sur les entrées d'horloge des bascules n'ayant pas lieu au même instant à cause des retards différents, les sorties ne changent pas d'état en même temps, ce qui implique un problème d'interface avec des circuits rapides (temps de lecture inférieur au retard entre plusieurs bits).

D'autre part, ces retards de commutation introduisent des états transitoires relativement conséquents, particulièrement lorsque le nombre de bascules traversées est important.

Mais l'inconvénient le plus important est lié au fait que cette structure nécessite de la logique sur des signaux asynchrones (l'horloge est générée par une bascule et le signal Clear est généré par une structure combinatoire). Cette logique combinatoire peut donc engendrer (ou propager) des états transitoires qui peuvent entraîner des dysfonctionnements du compteur.

X.2 Compteur synchrone (parallèle)

Toutes les bascules sont déclenchées en même temps par le même signal d'horloge. Ceci évite le problème du retard de propagation.

X.2.1 Réalisation

Elle est possible avec des bascules JK, D ou T.

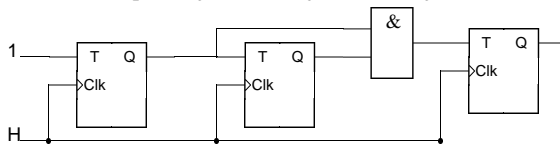
▷ Exemple 10.6

Réalisation d'un compteur modulo 8 (à cycle complet) à l'aide de bascules T

Table d'excitation

Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	0	1	1

On constate que : $T_0 = 1$ et $T_1 = Q_0$ et $T_2 = Q_1 Q_0$



▷ Exemple 10.7

Réalisation d'un compteur synchrone décrivant le cycle 4, 9, 1, 3, 2.

a) À l'aide de bascules JK :



b) À l'aide de bascules D :



c) À l'aide de bascules T :



X.2.2 Exemples de circuit intégré

Compteur pré-réglable 74160 (74161, 74162, 74163)

- l'état initial du compteur est réglable à l'aide des entrées D_1, D_2, D_3, D_4 ;
- validation : elle permet de verrouiller le compteur.

Circuit	Comptage	Chargement	RAZ
74160	synchr. DCB	synchrone	asynchrone
74161	synchr. bin.	synchrone	asynchrone
74162	synchr. DCB	synchrone	synchrone
74163	synchr. bin.	synchrone	synchrone

- RAZ synchrone : indépendant de l'horloge.
- RAZ asynchrone : 000 est obtenu au coup d'horloge suivant l'instant où clear est porté à l'état actif 0.

Compteur réversible pré-réglable 74193

- MR : entrée de réinitialisation asynchrone
- $Q_0 \dots Q_3$: sorties des bascules
- $P_0 \dots P_3$: entrée des données parallèles
- \overline{PL} : entrée de chargement asynchrone
- CP_U : entrée du signal d'horloge de comptage
- CP_D : entrée du signal d'horloge de décomptage
- E_C : valide le comptage
- E_D : valide le décomptage

► **Remarque 10.2**

Toutes les commandes agissant sur le comptage sont regroupées sur la figure ci-dessous :

X.2.3 Applications

X.2.3.a Compteur de fréquence

Circuit qui mesure et affiche la fréquence d'un signal impulsionnel (mesure de fréquence inconnue).

Principe d'un compteur de fréquence

comptées : durée pendant laquelle les impulsions sont

: RAZ met le compteur à zéro

: $f = \frac{\text{contenu}}{t_2 - t_1}$

► **Remarque 10.3**

Le compteur est un montage en cascade de compteurs DCB, chacun ayant une unité décodeur/afficheur (affichage décimal).

La précision de cette méthode est fonction de l'intervalle d'échantillonnage.

X.2.3.b Horloge numérique

Chapitre XI

Méthodes d'étude des circuits séquentiels

Charles Lutwidge Dodgson
ou Lewis Carroll
★ 1832 – † 1898



« Can you do addition? the White Queen asked. What's one and one and one and one and one and one and one and one and one and one and one? »
– I don't know, said Alice I lost count. » (Lewis Carroll, Through the Looking Glass)

De nombreux outils permettent d'analyser le fonctionnement et/ou de prévoir l'évolution d'un système séquentiel :

1°) Méthodes descriptives :

a) les tables d'état : elles donnent l'état futur des sorties pour les éléments de mémoire inclus dans les systèmes et l'état des sorties :

A	B	S	S
---	---	---	---

 ;

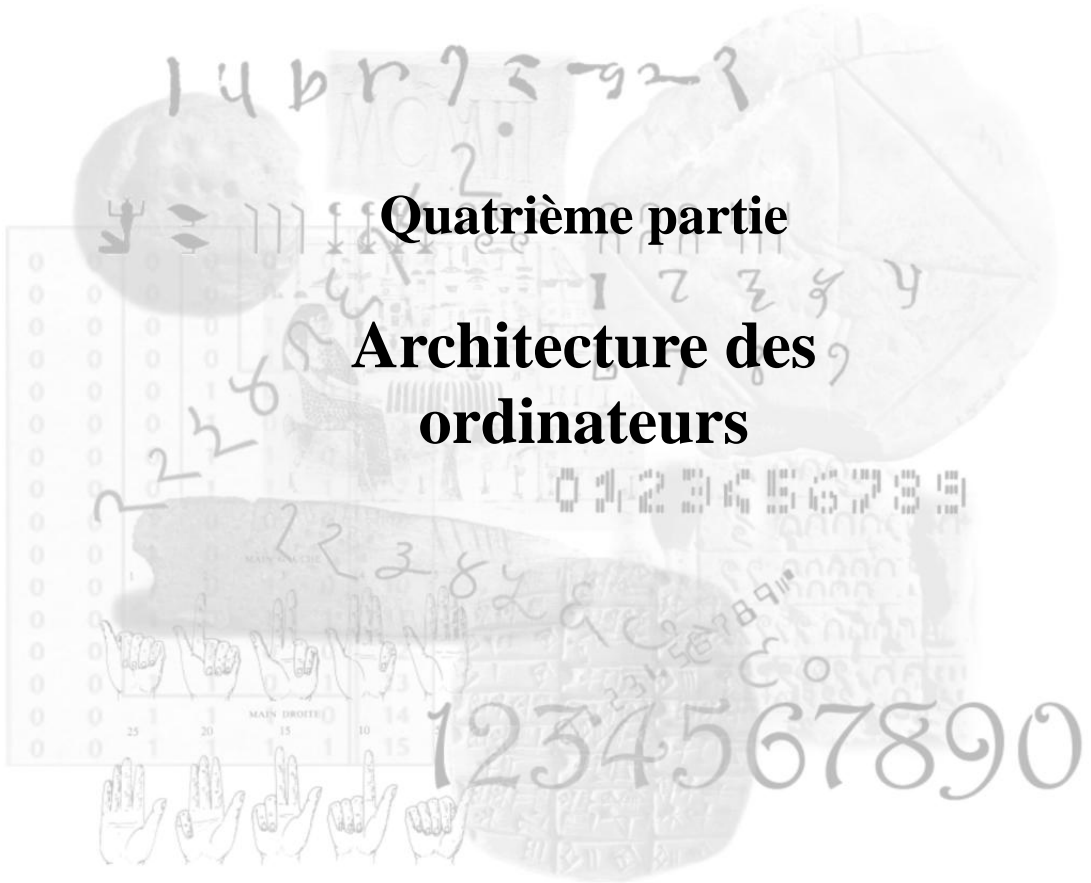
b) les diagrammes des temps (chronogrammes) : ils décrivent la succession des signaux d'entrée, des états des éléments de mémoire. Ils représentent la succession des états logiques en fonction du temps.

2°) Les diagrammes d'états ou graphes : ce sont des représentations formelles avec nœuds et flèches pour représenter les états stables et les transitions. Le graphe donne une image géométrique d'une table de vérité.

3°) Le grafcet : automatismes industriels : étape → transition → étape.

4°) Les théories formelles : équations qui représentent l'action à effectuer et l'état futur d'un élément de mémoire en fonction des entrées et de l'état présent des mémoires.

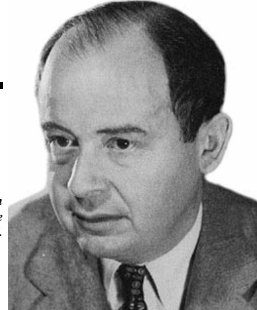
Quatrième partie
Architecture des
ordinateurs



Chapitre XII

Concepts de base des processeurs

John von Neumann
★ 28 déc. 1903, Budapest, Hongrie
† 8 fév. 1957, Washington D.C., E.-U.



« Si quelqu'un croit que les mathématiques sont difficiles, c'est simplement qu'il ne réalise pas comme la vie est complexe! »
(John von Neumann)

Cinquième partie
Technologie des portes
logiques



1234567890

Chapitre XIII

Famille des circuits logiques

William Bradford Shockley

★ 13 fév. 1910, London, R.-U. ; † 12 août 1989, London, R.-U.

John Bardeen

★ 23 mai 1908, Madison, Wisconsin, E.-U. ; † 30 jan. 1991

Walter Houser Brattain

★ 10 fév. 1902, Amoy, Chine ; † 13 oct. 1987

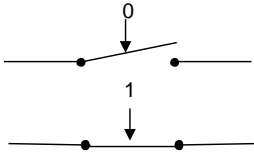


Lauréats du Prix Nobel de Physique de 1956 pour l'invention du transistor. John Bardeen obtiendra un second Prix Nobel de Physique en 1972 pour ses travaux sur la supraconductivité. La qualité des travaux de Shockley sur le transistor ne doit cependant pas crédibiliser ses théories eugéniques d'un autre âge ...

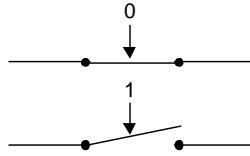
- TTL (*Transistor-Transistor Logic*) → le plus populaire (54xx, 74xx) ;
- ECL (*Emitter-Coupled Logic*) → pour des circuits rapides (10xxx) ;
- MOS (*Metal Oxid Semiconductor*) → haute intégration ;
- CMOS (*Complementary Metal Oxid Semiconductor*) → faible consom. (40xx) ;
- I²L (*Integrated Injection Logic*) → haute intégration.

Certains fonctionnent en logique positive, d'autres en logique négative.

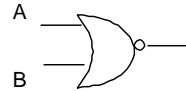
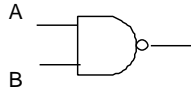
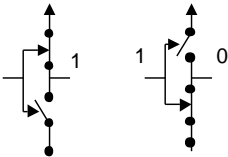
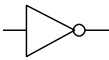
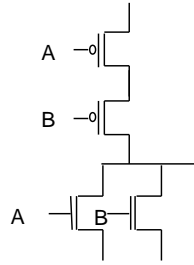
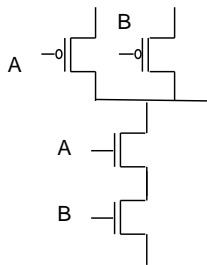
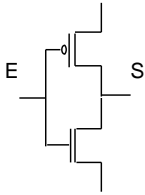
XIII.1 Implantation des opérateurs en technologie CMOS



Transistor N :
ouvert si grille =0
fermé si grille =1

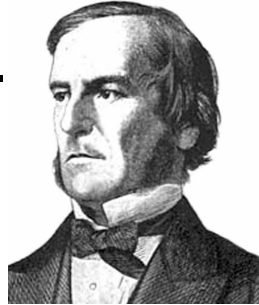


Transistor P :
ouvert si grille =1
fermé si grille =0



Annexe A

Correction des exercices



[Index]

— Symbols —

etat.....111

— A —

absorption.....45, 48, 51, 64
allègement.....64
associativité.....48, 51

— B —

bascule
à verrouillage.....97
D.....100
JK.....98
maître-esclave.....98
RS.....93
RS synchrone.....96
RST.....96
T.....101
base.....7
conversion de base.....10–12
binaire.....8

— C —

canonique
forme canonique.....49, 64, 87
produit canonique.....50
somme canonique.....49
commutativité.....45, 48, 51, 53
complémentarité.....48

complémentation.....47
compteur.....111
consensus.....51

— D —

De Morgan.....45, 49, 51, 53, 55, 64, 71
distributivité.....45, 48, 51
dualité.....49

— E —

élément absorbant.....51
élément neutre.....51
élément nul.....47
équivalence.....48

— F —

forme conjonctive.....50
forme disjonctive.....49

— H —

hexadécimal.....9

— I —

idempotence.....45, 47, 51
identité.....47, 51
identité.....45, 52
induction parfaite.....47
involution.....45, 47, 51

— **L** —

latch 97
LSB 9

— **M** —

modulo 111
MSB 9

— **O** —

octal 9

— **P** —

polynomiale 7

— **V** —

virgule
virgule fixe 16
virgule flottante 17–21

[Bibliographie]

- [WWW01] Site Web <http://www.trotek.ec-lyon.fr/~muller/cours/index.html>
- [WHI61] J.E. Whitesitt, *Boolean algebra and its applications*, 1961, Addison-Wesley.
- [KAR53] Maurice Karnaugh, "The Map Method for Synthesis of Combinational Logic Circuits", *Trans. AIEE. pt I*, 72(9) :593-599, November 1953.
- [LAF] J.-C. Lafont & J.-P. Vabre, « Cours et Problèmes d'Électronique Numérique », Éd. Ellipses.
- [TOC] R. J. TOCCI, « Circuits Numériques, Théorie et Applications », Éd. Dunod.

[Table des matières]

Notes sur cet ouvrage	3
Partie 1 : Les nombres	5
I Les systèmes de numération	7
I.1 La représentation polynomiale	7
I.2 Le système binaire	8
I.3 Le système octal	9
I.4 Le système hexadécimal	9
I.5 Conversion d'un système de numération à un autre	10
II Codage des nombres dans les machines numériques	13
II.1 Représentation des nombres entiers positifs	14
II.2 Représentation binaire des entiers signés	14
II.3 Représentation des nombres réels dans un calculateur	16
II.4 Arithmétique binaire	21
II.5 En résumé	25
III Les codes numériques	27
III.1 Codes numériques pondérés	27
III.2 Codes numériques non pondérés	31
III.3 Codes détecteurs d'erreurs et autocorrecteurs	34
III.4 Les codes alphanumériques	36
Exercices sur les nombres	39
Partie 2 : La logique combinatoire	41
IV Algèbre booléenne et opérateurs logiques	43
IV.1 Introduction	43
IV.2 Propriétés de l'algèbre booléenne	45
IV.3 Algèbre binaire ou algèbre de commutation	46

IV.4	Théorèmes monovariabiles	47
IV.5	Théorèmes multivariabiles	48
IV.6	Opérateurs logiques élémentaires et composés	52
IV.7	Universalité des portes NON-ET et NON-OU	55
V	Représentation et simplification des fonctions logiques	59
V.1	Méthodes de représentation des fonctions logiques	59
V.2	Simplification d'expressions logiques	64
VI	Les circuits combinatoires	71
VI.1	Circuits logiques combinatoires usuels	71
VI.2	Synthèse des circuits combinatoires	79
VII	Fonctions et opérateurs arithmétiques	85
	Exercices sur les systèmes combinatoires	87
 Partie 3 : Les circuits séquentiels		 89
VIII	Les bascules	91
VIII.1	Introduction	91
VIII.2	Point mémoire	92
VIII.3	Bascule RS	93
VIII.4	Bascule RS synchrone ou bascule RSH	96
VIII.5	Bascule à verrouillage (<i>D-latch</i>)	97
VIII.6	Bascules maître-esclave	98
VIII.7	Bascule JK	98
VIII.8	Bascule D synchrone	100
VIII.9	Bascule T	101
VIII.10	Entrées prioritaires asynchrones des bascules	102
VIII.11	Paramètres temporels des bascules	103
VIII.12	Applications des bascules	103
IX	Registres : stockage et transfert de données	107
IX.1	Définition	107
IX.2	Registre de mémorisation : écriture et lecture parallèles	108
IX.3	Registres à décalage	108
IX.4	Registre universel	110
X	Les compteurs	111
X.1	Compteur asynchrone (à propagation)	112
X.2	Compteur synchrone (parallèle)	117
XI	Méthodes d'étude des circuits séquentiels	121

Partie 4 : Architecture des ordinateurs	123
XII Concepts de base des processeurs	125
Partie 5 : Technologie des portes logiques	127
XIII Famille des circuits logiques	129
XIII.1 Implantation des opérateurs en technologie CMOS	130
Partie 6 : Annexes	131
A Correction des exercices	133
Index	135
Bibliographie	137