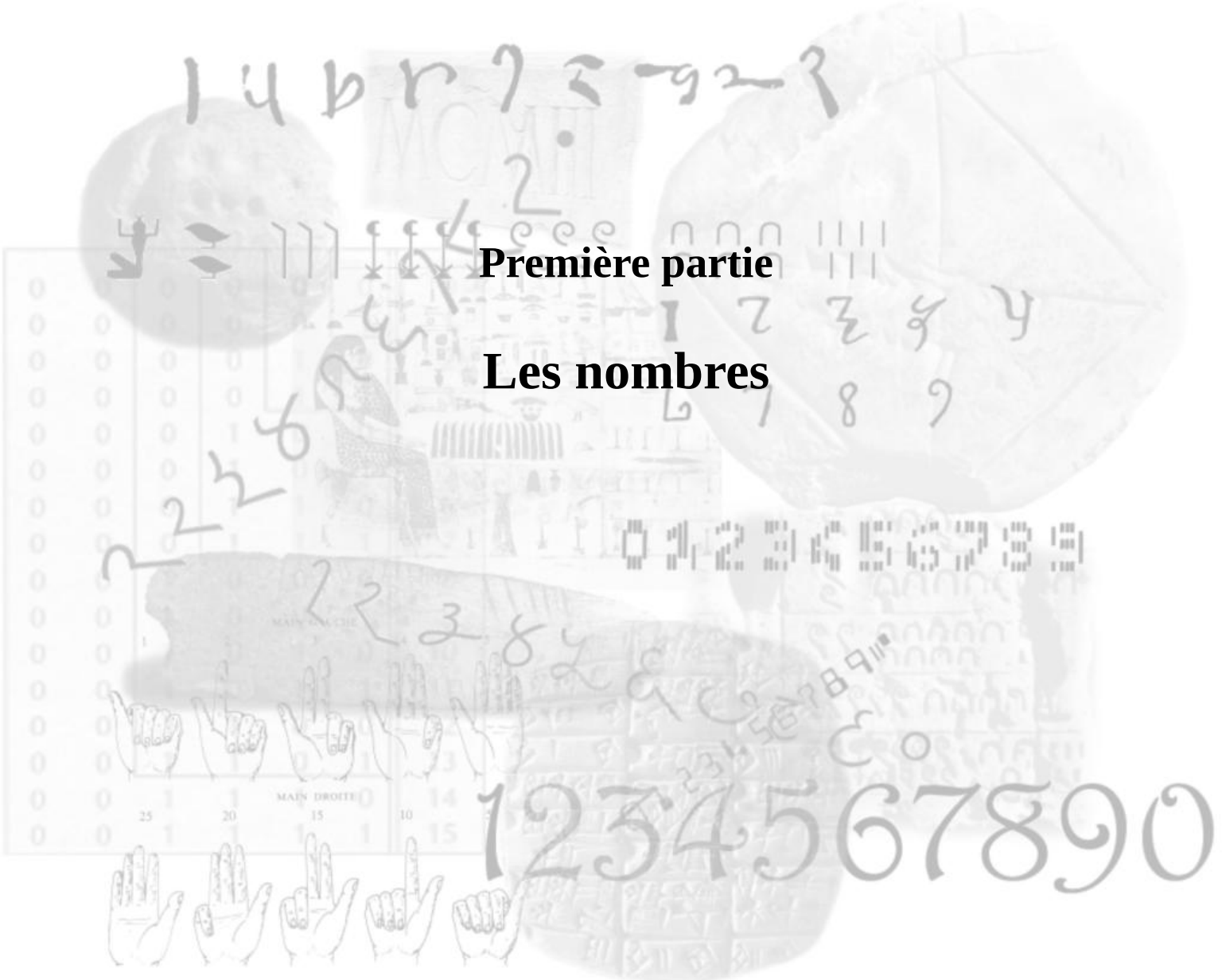
Cours d’électronique numérique

Maryam Siadat & Camille Diou

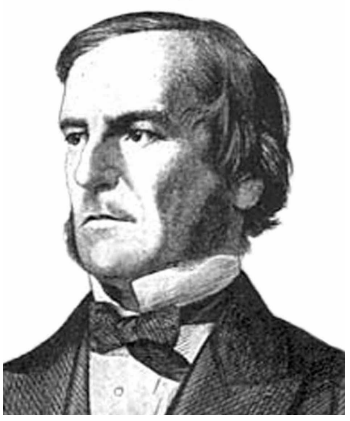


Première partie

Les nombres

Chapitre 1

Les systèmes de numération



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

1.1 La représentation polynomiale

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

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

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 ».

1.2 Le système binaire

1.2.1 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' et '1' appelés « éléments binaires » ou « bits » (bit=*Binary digIT*) ;

– le système binaire est pondéré par 2 : les poids sont les puissances de 2 ;

Exemple :

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-3}	2^{-3}
1	0	1	1	0	0	1	,	0	1	1

– les différentes puissances de 2 sont :

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

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

1.2.2 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	Poids :	2^3	2^2	2^1	2^0	B10
	0	0	0	0	0		1	0	0	0	8
	0	0	0	1	1		1	0	0	1	9
	0	0	1	0	2		1	0	1	0	10
	0	0	1	1	3		1	0	1	1	11
	0	1	0	0	4		1	1	0	0	12
	0	1	0	1	5		1	1	0	1	13
	0	1	1	0	6		1	1	1	0	14
	0	1	1	1	7		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 :

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

Remarque :
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.*

1.3 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 :

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

1.4 Le système hexadécimal

– base B=16 ;
– 15 symboles : ’0’, ’1’, ’2’, . . . , ’9’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’ appelés « digits » ;
– chaque symbole est exprimé en binaire sur 4 bits ;

Exemple :

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

1.5 Conversion d’un système de numération à un autre

1.5.1 Base *B* vers base 10

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

Exemple :

$(1001,1)_2 = 1.2^3 + 0.2^2 + 0.2^1 + 1.2^0 + 1.2^{-1} = 8 + 0 + 0 + 1 + 0,5 = 9,5$
$(A12)_{16} = A.16^2 + 1.16^1 + 2.16^0 = 2560 + 16 + 2 = 2578$

1.5.2 Base 10 vers base *B*

A Première méthode

Elle consiste à soustraire successivement la plus grande puissance de *B*

Exemple :

$$\left.\begin{array}{rcrcr} 100 & = & 1.2^6 & + & 36 \\ 36 & = & 1.2^5 & + & 4 \\ 4 & = & 1.2^2 & + & 0 \end{array}\right\} \rightarrow (100)_{10} = (1100100)_2$$

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 :

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

Partie entière :

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

Partie fractionnaire :

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

Le résultat est donc 10100,0110.

1.5.3 Base 2^n vers base 2

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

Exemple :

$(3A9)_{16} = (0011\ 1010\ 1001)_2$
$(742,5)_8 = (111\ 100\ 010,101)_2$

1.5.4 Base 2 vers base 2^n

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

Exemple :

$$(1011011)_2 = (\underbrace{001}_{1}\ \underbrace{011}_{3}\ \underbrace{011}_{3})_2 = (133)_8$$
$$= (\underbrace{0101}_{5}\ \underbrace{1011}_{B})_2 = (5B)_{16}$$

1.5.5 Base *i* vers base *j*

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

Exemple :

base 8 → base 2 → base 16

– sinon, on utilise la base 10 comme relais.

Exemple :

base 5 → base 10 → base 2

Chapitre 2

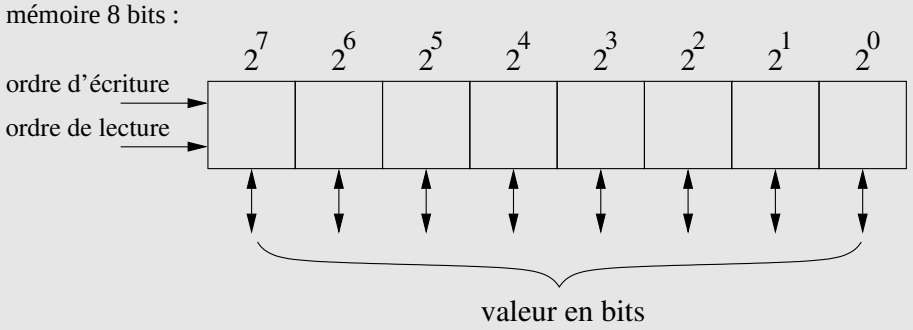
Codage des nombres dans les machines numériques

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

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 :

mémoire 8 bits :



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

2.1 Représentation des nombres entiers positifs

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

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

2.2 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 $-$.

2.2.1 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 :

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'.

2.2.2 Représentation en complément restreint (complément à 1)

$-A = \overline{A}$: 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 :

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.

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

$$\begin{aligned} -A &= \overline{A}+1 \\ \overline{A} &= \overline{a_{n-1}}\,\overline{a_{n-2}}\cdots\overline{a_0} \qquad \text{(on complémente chaque coefficient)} \\ A+\overline{A} &= 1\,1\ldots1 \\ 1+\overline{A}+A &= \mathbf{1} \leftarrow \underbrace{0\,0\ldots0}_{0} \qquad \text{(car on représente sur } n \text{ bits seulement)} \end{aligned}$$

$\Rightarrow -A = \overline{A}+1$ est appelé compément à 2

Remarque :

- *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 ;*

Exemple :

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

- 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 :

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

2.3 Représentation des nombres réels dans un calculateur

Dans un calculateur, un nombre est toujours écrit sous forme d'1 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.

2.3.1 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 :

Virgule au rang K (K chiffres après la virgule) :
La valeur N écrite en mémoire aura les poids suivants :
$N = 2^{N-1-K} \cdots 2^0\, 2^{-1}\, 2^{-K}$ $\qquad 0 \leq N \leq (2^n - 1)2^{-K}$
Virgule au rang 0 :
$N = 2^{N-1} \cdots 2^0$ $\qquad 0 \leq N \leq 2^N - 1$

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) ;
- limitation de la taille de la partie entière et de la partie décimale

Exemple :

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 faible et des grandeurs très importantes.

2.3.2 La représentation en virgule flottante

Le nombre N est représenté sous la forme :

exposant	mantisse

1ère approche

Soit $N = a_3\, a_2\, a_1\, a_0\, ,\, a_{-1}\, a_{-2}\, a_{-3}$: N peut se noter : $(a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0).2^{-3}$

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

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

Exemple :

Soit la mémoire de taille suivante :

4 bits	12 bits
exposant	mantisse

Coder la valeur 26,75 en virgule flottante.

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

$(11010,11)_2 = (11010110).2^{-3} \rightarrow \begin{cases} \text{exposant} = & -3 \\ \text{mantisse} = & 11010110 \end{cases}$

1101	0000011010110
exp=−3	mantisse=214

$26,75 = 214.2^{-3}$

2ème approche

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

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

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

Exemple :

Même exemple que précédemment :

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

0101	110101100000

Remarque :

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 $\Rightarrow 0,5 \leq M < 1$

2.4 Arithmétique binaire

2.4.1 Addition

L'addition en binaire se fait avec les mêmes règles qu'en décimal : on commence à additionner les bits de poids faibles puis 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 :

Addition des nombres de 4 bits :

	0	0	1	1		(+3)		0	1	1	1	,	1	1		7,75
+	1	0	1	0	+	(-6)	+	0	1	0	1	,	0	1	+	5,25
=	1	1	0	1	=	-3	=	**1**	1	0	1	,	0	0	=	-3,00

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).

2.4.2 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 :

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

Remarque :

On peut utiliser le complément à 2 de la valeur à soustraire puis on additionne. Cela se passe de cette manière dans les calculateurs.

Exemple :

$7-2$:

7 :	0	0	1	1	1			0	0	1	1	1
2 :	0	0	0	1	0		+	1	1	1	1	0
-2 :	1	1	1	1	0	**1**	←	0	0	1	0	1

On ne tient pas compte de la retenue.

2.4.3 Multiplication

La table de multiplication en binaire est très simple :

A		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 multiplieur.

Exemple :

		0	1	0	1	multiplicande		5
	×	0	0	1	0	multiplieur	×	2
		0	0	0	0			
	0	1	0	1	=			
0	0	0	0	=	=			
	0	1	0	1	0		=	10

Remarque :
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 :

8×4 *sur 8 bits :*

0	*0*	*0*	*0*	*1*	*0*	*0*	*0*	$\Longrightarrow$	*0*	*0*	*1*	*0*	*0*	*0*	***0***	***0***	← ***0***

-16×4 *sur 8 bits :*

1	*1*	*1*	*1*	*0*	*0*	*0*	*0*	$\Longrightarrow$	*1*	*1*	*0*	*0*	*0*	*0*	***0***	***0***	← ***0***

2.4.4 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 :

Division du nombre $(10010000111)_2$ par $(1011)_2 = (1101001)_2$ reste $(100)_2$, c'est-à-dire $1159/11 = 105$, reste 4.

```
1  0  0  1  0  0  0  0  1  1  1 | 1  0  1  1
-  1  0  1  1                   | 0  1  1  0  1  0  0  1
   0  1  1  1  0                |
   -  1  0  1  1                |
      0  0  1  1  0  0          |
         -  1  0  1  1          |
            0  0  0  1  1  1  1 |
               -  1  0  1  1    |
                  0  1  0  0    |
```

Remarque :
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 :

$8/4$ *sur 8 bits :*

0	0	0	0	1	0	0	0	$\Longrightarrow$	**0**→	**0**	**0**	0	0	0	0	1	0

$-16/4$ *sur 8 bits :*

1	1	1	1	0	0	0	0	$\Longrightarrow$	**1**→	**1**	**1**	1	1	1	1	0	0

Chapitre 3

Les codes numériques

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

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.

3.1 Codes numériques pondérés

3.1.1 Code binaire pur

→ code pondéré par des puissances de 2.

Utilisé en arithmétique binaire.

Ses dérivées sont le code octal et le code hexadécimal.

3.1.2 Code DCB (Décimal Codé Binaire)

→ chaque chiffre décimal $(0, 1, \ldots, 9)$ est codé en binaire avec 4 e.b.

Code pondéré avec les poids $1, 2, 4, 8, 10, 20, 40, 80, 100, \ldots$

Plus facile pour coder des grands nombre, il est surtout utilisé pour l'affichage des nombres.

Remarque :
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 :

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

3.1.3 Code binaire de Aiken

Pondéré par 2421, c'est un code autocomplémentaire. (les représentations de 2 chiffres dont la somme est 2 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. $5 \rightarrow 5+6=11, 6 \rightarrow 6+6=12, \ldots$)

Exemple :

décimal	Aiken 2	4	2	1		décimal	Aiken 2	4	2	1
0	0	0	0	0		5	1	0	1	1
1	0	0	0	1		6	1	1	0	0
2	0	0	1	0		7	1	1	0	1
3	0	0	1	1		8	1	1	1	0
4	0	1	0	0		9	1	1	1	1

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

3.1.4 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 :

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

3.2 Codes numériques non pondérés

3.2.1 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.

Exemple :

$(48)_{10}$	4	8
	+3	+3
	7	11
	↓	↓
	0111	1011

3.2.2 Code de Gray (binaire réfléchi)

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

Exemple :

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.

Ce code est utilisé dans les tableaux de Karnaugh (*c.f.* section **??** page **??**), dans des circuits d'entrée/sortie, et dans certains convertisseurs analogique/numérique.

Il ne convient pas pour l'arithmétique binaire.

3.3 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.

3.3.1 Codes biquinaires

c.f. section **??** page **??**.

3.3.2 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.

Exemple :

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 :

Décimal	2 parmi 5				
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
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

3.3.3 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 :

0101 $\longrightarrow$ **0**0101
0111 $\longrightarrow$ **1**0111

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

Exemple :

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

Transmission →

→

0	1	0	0	1
1	0	0	**0**	0
0	0	0	1	1
1	1	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.

3.3.4 Code de Hamming

Ce code est utilisé dans les transmissions de données. Il localise et corrige les chiffres erronnés (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 $a = A \oplus B \oplus C \oplus D$
$b = A \oplus C \oplus D$
$c = A$
$d = B \oplus C \oplus D$
$e = B$
$f = C$
$g = D$

3.4 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 +,|,≠,%,...)

3.4.1 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 $8^{\text{ème}}$ e.b. peut servir de parité) : $2^7 = 128$ combinaisons différentes.

Deuxième partie

La logique combinatoire

Chapitre 4

Fonctions et opérateurs logiques

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

4.1 Introduction

Les sytèmes logiques fonctionnent en mode binaire ⟶ 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 :

Technologie électrique TTL :			Technologie pneumatique :		
« 1 »	↔	2,4 à 5 V	« 1 »	↔	présence de pression
« 0 »	↔	0 à 0,8 V	« 0 »	↔	abscence de pression

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

Convention des synonymes des « 0 » et « 1 » :

Ces deux valeurs peuvent être nommées de différentes façons :

Niv. log. « 0 »	Niv. log. « 1 »
Faux	Vrai
Ouvert	Fermé
Arrêt	Marche
Bas	Haut
Non	Oui

On définit deux types de logiques :

Logique positive :
- niveau haut $\longrightarrow$ état logique « 1 » (+5 V)
- niveau bas $\longrightarrow$ état logique « 0 » (0 V)

Logique négative :
- niveau haut $\longrightarrow$ état logique « 0 » (0 V)
- niveau bas $\longrightarrow$ état logique « 1 » (+5 V)

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 sytèmes 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 : $\overline{\ }$

$\longrightarrow$ 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.

4.2 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, électrovanne,...)

Convention :

Tout appareil est schématisé à l'état de repos.

Dans tous les cas, l'action sur un appareil sera notée a, b, ... et la non action $\overline{a}$, $\overline{b}$, ...

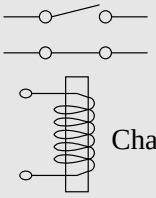
Exemple :

Bouton poussoir $\longrightarrow$ contact repos et contact travail.
1er cas : schéma d'un contact ouvert au repos dit « contact travail ».
2^{è} cas : schéma d'un contact fermé au repos dit « contact repos ».

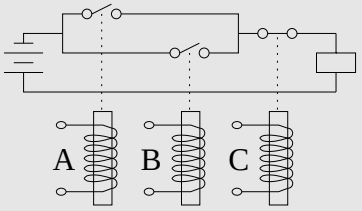
Exemple :

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 alimentéee (au repos).

Ils peuvent être fermés ou ouverts au repos.



Symbole d'un relais double normalement ouvert et fermé



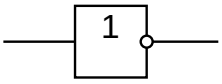
$T = (A+B).\overline{C}$

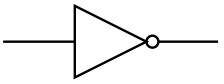
4.3 Opérateurs logiques élémentaires

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.

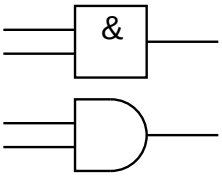
4.3.1 NON (NOT)





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

4.3.2 ET (AND)



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

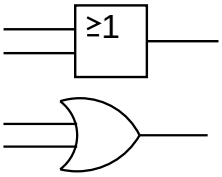
Propriétés du ET :

$x.1 = x$ $\quad x.\overline{x} = 0$
$x.0 = 0$ $\quad x.x = x$

Élément neutre : 1

Élément absorbant : 0

4.3.3 OU (OR)



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

Propriétés du OU :

$x+1=1 \qquad x+\overline{x}=1$
$x+0=x \qquad x+x=x$

Élément neutre : 0

Élément absorbant : 1

4.3.4 NON-OU (NOR)

S=$\overline{A+B}$

4.3.5 NON-ET (NAND)

S=$\overline{A.B}$

4.3.6 OU exclusif (XOR)

S=A$\oplus$B

4.3.7 Coïncidence : NON-OU exclusif (XNOR)

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

4.4 Description des circuits logiques

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.

4.4.1 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.

Exemple :

A	B	C	X	Y
0	0	0	?	?
	⋮		⋮	⋮
1	1	1	?	?

Exemple :

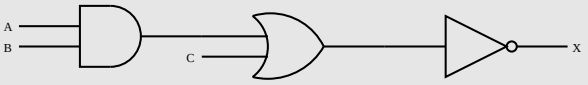
Donner la table de vérité d'un circuit à 3 entrées A,B,C et 2 sorties X,Y tel que :
- X=1 si les 3 entrées ont le même niveau
- Y=1 si A=B

4.4.2 Logigramme

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

Exemple :

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



Remarque :
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 $\overline{c}$, $\overline{d}$, $\overline{MEM}$, etc.

4.5 Universalité des portes NON-ET et NON-OU

Théorème de De Morgan :

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

Le théorème de De Morgan et ses conséquences est :
– très utile pour simplifier des expressions ;

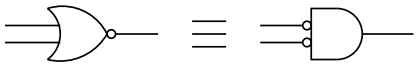
– valable également si a ou b sont des expressions contenant plusieurs variables

Exemple :

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

– conséquences :

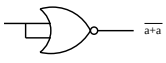
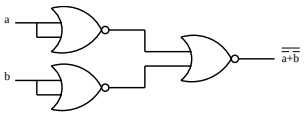
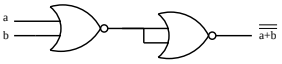
1. une porte NON-OU est une porte ET avec ses entrées inversées :



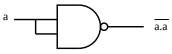
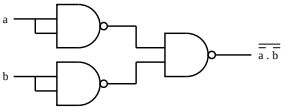
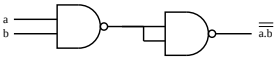
2. une porte NON-ET est une porte OU avec ses entrées inversées :

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.

4.5.1 Universalité des portes NON-OU

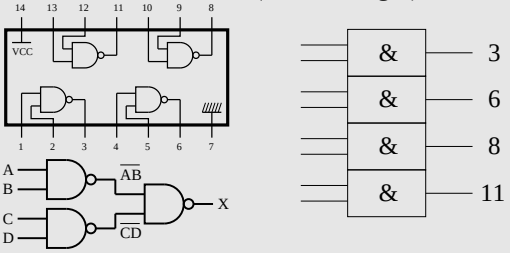
NON :  ET :  OU : 

4.5.2 Universalité des portes NON-ET

NON :  OU :  ET : 

Exemple :

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

7400 : 

Remarque :

- *la même approche peut etre faite pour les portes OU exclusif ;*
- *les opérateurs NAND, NOR, OU exclusif, ainsi que le groupe d'opérateurs (ET, OU, NON) sont appelés des « opérateurs complets » puisqu'ils peuvent réaliser seuls toutes les opérations logiques.*

4.5.3 Famille des circuits logiques

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

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

4.6 Simplification d'expressions logiques

4.6.1 Méthode algébrique

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

Exemple :

$x+\overline{x}y = x(1+y)+\overline{x}y = x+xy+\overline{x}y = x+y$ (théorème d'allègement)
$x.(x+y) = x+xy = x$ (absorption)
$ABC + A\overline{B}C + AB\overline{C} + \overline{A}BC = AC + AB + BC$

4.6.2 Diagramme de Karnaugh et termes adjacents

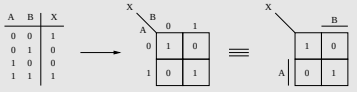
Deux termes sont adjacents quand ils ne diffèrent l'un de l'autre que par une seule variable. $A\overline{B}C$ et ABC sont adjacents. Un digramme de Karnaugh est une table d'implication logique disposée de telle manière que deux termes logiquement adjacents soient également adjacents géométriquement.

Le diagramme de Karnaugh est un outil graphique, méthodique. Il permet d'obtenir une solution optimale à la simplification logique.

A Forme du diagramme de Karnaugh

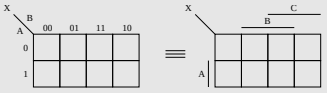
Comme la table de vérité, il met en évidence le rapport entre les entrées et les sorties (chaque ligne de la table de vérité correspond à une case du diagramme de Karnaugh).

Exemple :



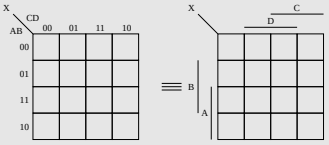
Exemple :

à 3 entrées :



Exemple :

à 4 entrées :

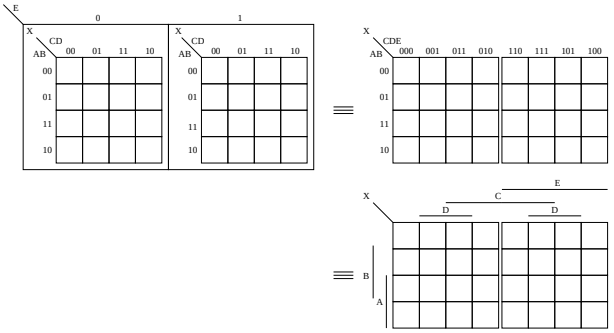


Remarque :
Les tableaux de Karnaugh se présentent comme des cylindres fermés dans les deux sens.

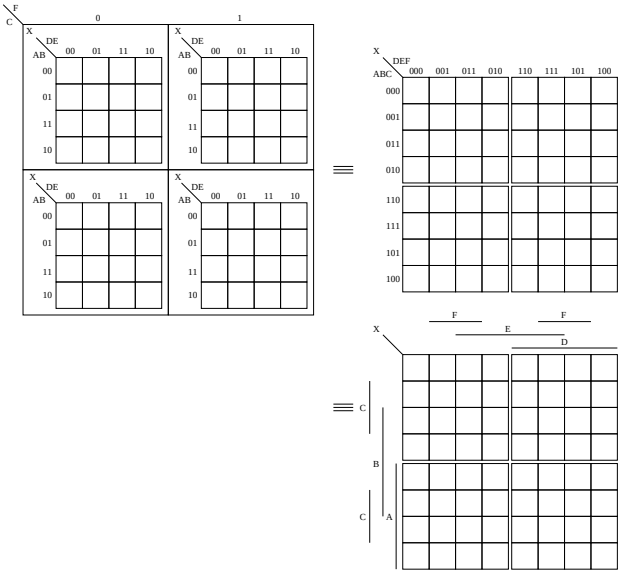
B Méthode de simplification

1. dessiner la table de Karnaugh correspondant à la fonction ;
2. on entame les 1 isolés ;
3. on réunit les octets de 1 adjacents ;
4. on réunit les quartes de 1 adjacents ;
5. on réunit les doublets de 1 adjacents pour réunir tous les 1 du tableaux ;
6. effectuer l'addition logique e tous les termes résultants des réunions, 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) ;

C Tableau de Karnaugh à 5 variables



D Tableau de Karnaugh à 6 variables



Remarque :

- *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 meme case plusieurs fois, puisque* $x + x + x + \cdots + x = x$.

4.6.3 Méthodes algorithmiques

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

Chapitre 5

Les circuits combinatoires

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

5.1 Circuits logiques combinatoires usuels

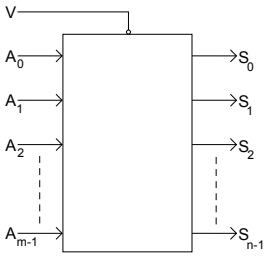
Circuit combinatoire : circuit dont les sorties dépendent uniquement de la combinaison des états des entrées à l'instant de l'observation.

5.1.1 Circuits de transcodage (codeurs, décodeurs, convertisseurs)

A Codeur (encodeur)

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

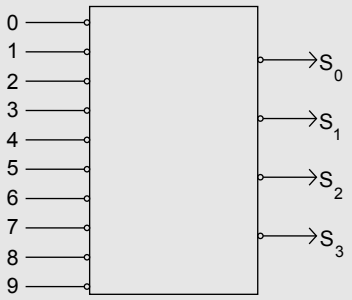
une seule entrée active à la fois



représentation en sortie sur N bits

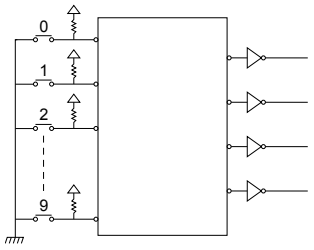
Exemple :

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



représentation DCB

Remarque :

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.

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 :

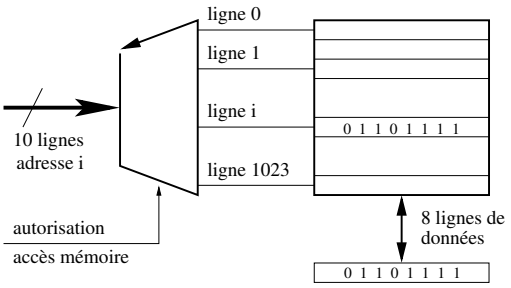
Décodeur DCB-décimal : 4 entrées, 10 sorties.

Remarque :

La plupart des décodeurs sont dotés d'une ou plusieurs entrées de validation qui comandent 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 :

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

Remarque :

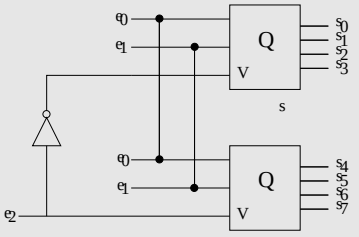
Il n'est pas nécessaire de simplifier la fonction avant la réalisation.

Mise en cascade des décodeurs

Utilisation de l'entrée de validation.

Exemple :

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.

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 :

Code binaire → code Gray
Code DCB → code affichage chiffre (décodeur 7 segments)

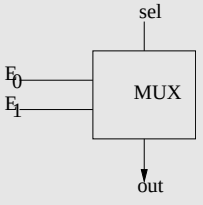
5.1.2 Multiplexeurs–démultiplexeurs

A Multiplexeurs (MUX)

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

Exemple :

MUX à 2 entrées de données



E_1	E_0	sel	out
X	X	0	E_0
X	X	1	E_1

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

Remarque :

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

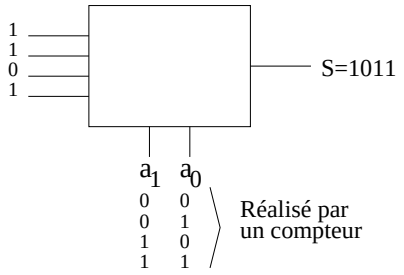
fonction de sortie directement

Exemple :

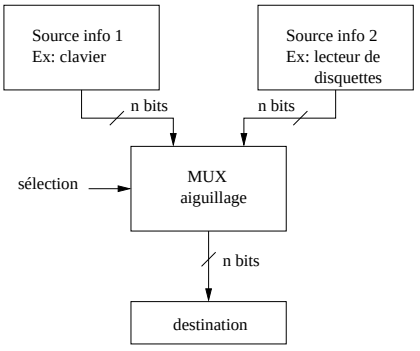
MUX à 4 entrées ($\rightarrow$ 2 entrées de sélection a_1a_0) $S=\overline{a_1}.\overline{a_0}.E_0 + \overline{a_1}.a_0.E_1 + \cdots$

B Application des MUX

1. **Conversion parallèle–série :** on place successivement les valeurs 00, 01, 10, 11 sur a_1a_0.



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

Exemple :

$F = ABC + A\overline{BC}$
Utilisation de MUX 8 vers 1.
$S = ABCE_0 + AB\overline{C}E_1 + \cdots + A\overline{BC}E_4 + \cdots$

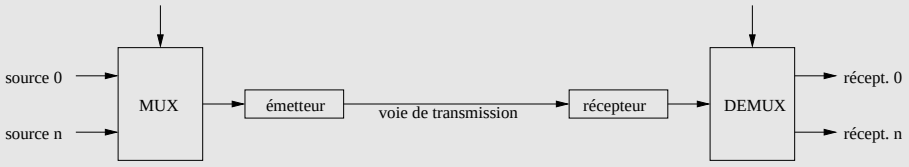
C Démultiplexeurs (DEMUX)

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

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

Exemple :

Transmission avec multiplexage/démultiplexage.



5.1.3 Le comparateur

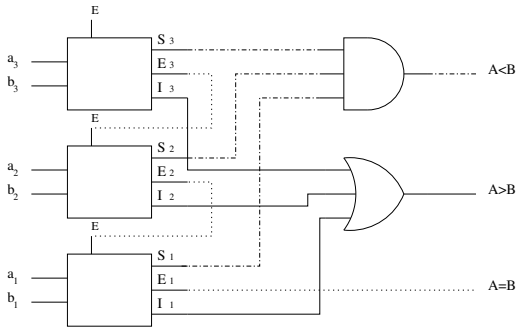
Il détecte l'égalité entre deux nombres A et B. Certains circuits permettent également de détecter si A est supérieur ou bien inférieur à B.

A Comparateur de 2 e.b.

a_i	b_i	E_i	S_i	I_i
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

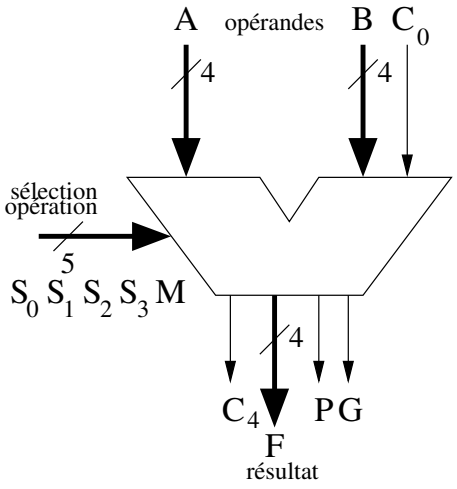
$E_i = \quad a_i = b_i \quad = \overline{a \oplus b}$
$S_i = \quad a_i > b_i \quad = a.\overline{b}$
$I_i = \quad a_i < b_i \quad = \overline{a}.b$
$D_i = \quad a_i \neq b_i \quad = a \oplus b$

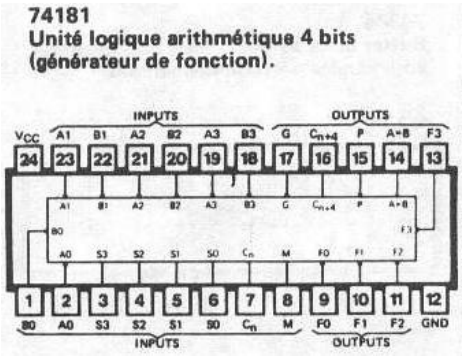
B Comparateur de 2 nombres



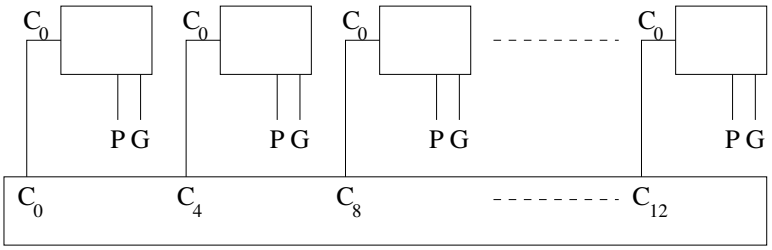
5.1.4 L'unité arithmétique et logique (UAL)

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.





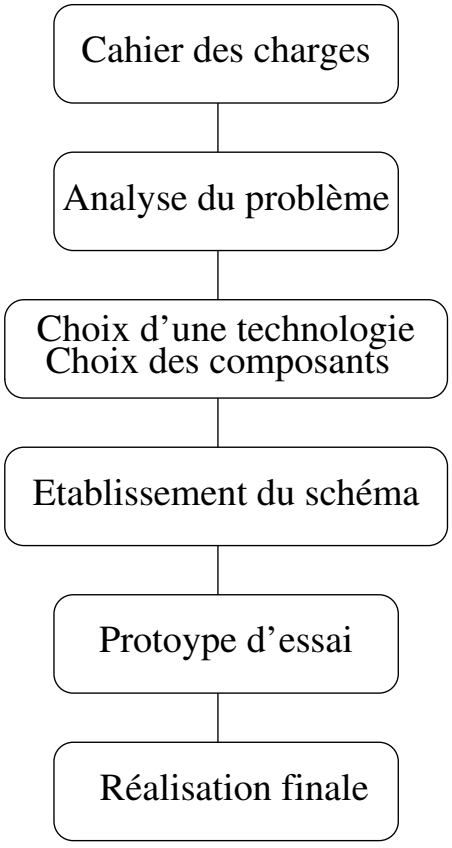
Les sorties P et G servent à la mise en cascade des ALUs, et donc au calcul de retenue anticipée.



Générateur de retenue anticipée

5.2 Synthèse des circuits combinatoires

5.2.1 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 boitiers, coût, disponibilité, points test, complexité des connexions, etc.

Les différents choix sont :

a) utilisation de portes simples (OU, ET, NON) ou des portes NON-OU et NON-ET ;

b) développement de circuits intégrés (CI) spécialisés. Le problème du coût de développement et de fabrication impose une production en très grandes séries ;

c) utilisation de circuits intégrés combinatoires :
   – MUX, DEMUX ;
   – décodeurs ;
   – circuits logiques programmables : PROM, PAL, etc.

5.2.2 Circuits logiques programmables

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.

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 :

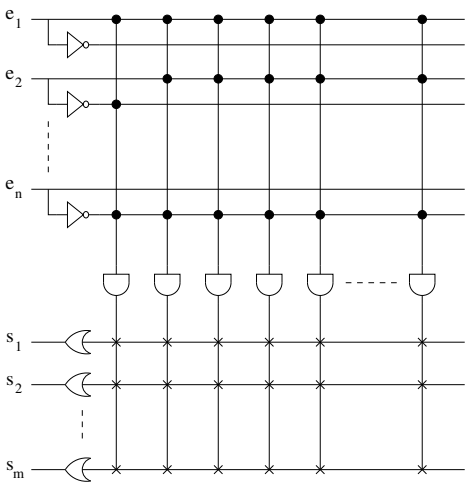
- un ensemble d'opérateurs ET organisés sous forme de matrice permet de générer les produits des variables d'entrée ;
- un ensemble d'opérateurs OU permet de sommer les produits.

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.* section c **??** page **??**), 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.

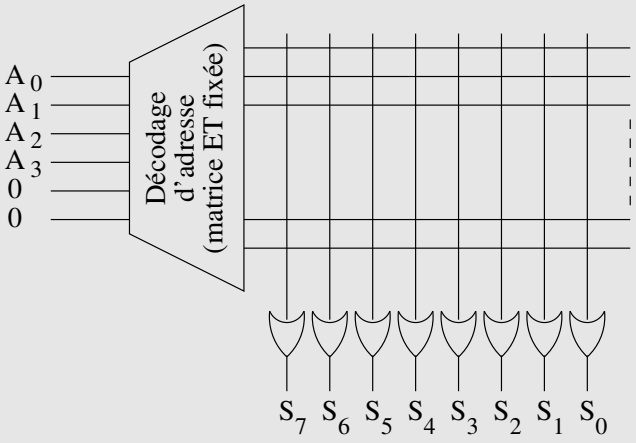
   - chaque sortie de la mémoire correspond à une fonction (sortie 3 états) ;
   - la matrice ET correspond en fait à un décodeur $n \rightarrow 2^n$ (décodeur d'adresse) ;
   - une fonction est réalisée en programmant sa table de vérité, c'est-à-dire en mettant en mémoire la valeur de f pour l'ensemble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



 : interconnexion non programmée  : interconnexion programmée

Exemple :

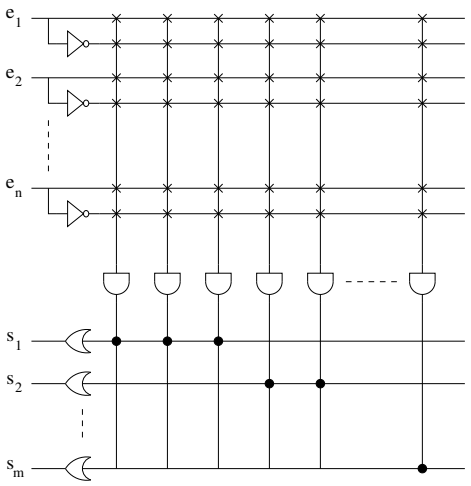
Réaliser le circuit $N \rightarrow N^2$ (N : nombre codé en DCB sur 4 bits) à l'aide de la PROM suivante (PROM à 6 entrées et 8 sorties → capacité de $2^6 = 64$ octets) :



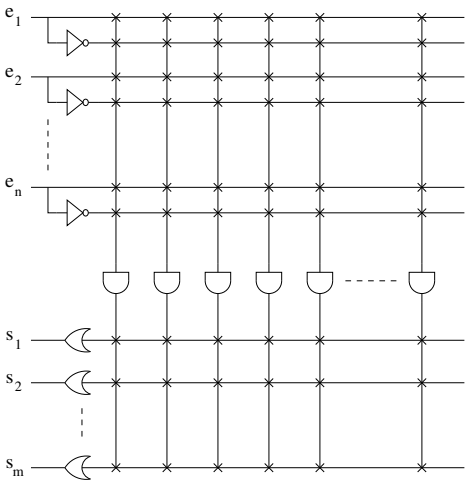
2. PAL (*Programmable Array Logic*)

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

   Les circuits PAL existent également en logique séquentielle.

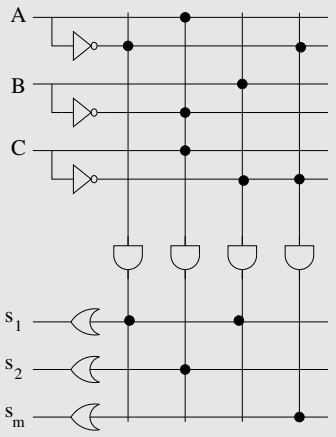


3. FPLA (*Field Programmable Logic Array*) : matrice OU et ET programmable



La structure des FPLA autorise une très grande souplesse dans la programmation. Par conséquent, c'est le circuit le plus souvent proposé pour la réalisation des fonctions logiques.

Exemple :



$S_1 = \overline{A} + B\overline{C}$
$S_2 = AB\overline{C}$
$S_3 = \overline{AC}$

5.2.3 Programmation des circuits logiques programmables

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

Chapitre 6

Fonctions et opérateurs arithmétiques

John von Neumann
28 déc. 1903, Budapest, Hongrie
8 fév. 1957, Washington DC, E.-U.

Troisième partie

Les circuits séquentiels

Chapitre 7

Les bascules

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

7.1 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 :

$$1\begin{cases} M=0 \\ A=0 \end{cases} \rightarrow L=0 \qquad 3\begin{cases} M=0 \\ A=0 \end{cases} \rightarrow L=1 \qquad 5\begin{cases} M=0 \\ A=0 \end{cases} \rightarrow L=0$$

$$2\begin{cases} M=1 \\ A=0 \end{cases} \rightarrow L=1 \qquad 4\begin{cases} M=0 \\ A=1 \end{cases} \rightarrow L=0$$

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.

7.2 Bascule RS

La bascule RS est le circuit séquentiel le plus simple. Son rôle consiste à mémoriser une information fugitive.

Symbole

R : reset

S : set

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

$\longrightarrow$

S	R	Q^+	
0	0	Q	$\rightarrow$ mémorisation
0	1	0	$\rightarrow$ mise à 0
1	0	1	$\rightarrow$ mise à 1
1	1	X	$\rightarrow$ interdit

Tableau de Karnaugh

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

Si X=1 $\rightarrow$ Q=S+$\overline{R}$.Q (avec priorité de mise à 1).

$\rightarrow$ somme de produit $\Rightarrow$ réalisation à l'aide de portes NAND.

Si X=0 $\rightarrow$ Q=$\overline{R}$.(S+Q) (avec priorité de mise à 0).

$\rightarrow$ produit de sommes $\Rightarrow$ réalisation à l'aide de portes NOR.

Remarque :
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→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 :

$$\left.\begin{array}{l} -\text{ si } Q=0 \xrightarrow{S=0} \overline{\text{G}}=1 \text{ et } Q=0 \\ -\text{ si } Q=1 \xrightarrow{S=0} \overline{\text{G}}=0 \text{ et } Q=1 \end{array}\right) \rightarrow \text{memorisation}$$

– Examinons si S :0$\rightarrow$1 et R=0
Si Q=0 à l'arrivée de l'impulsion sur S, alors $\text{S}=1 \rightarrow \overline{\text{Q}}=0 \rightarrow Q=1$

Si Q=1 à l'arrivée de l'impulsion sur S, alors $\text{S}=1 \rightarrow \overline{\text{Q}}=0 \rightarrow$ 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 :0$\rightarrow$1 et S=0
Si Q=0 $\xrightarrow{\text{R :0}\rightarrow\text{1}}$ Q=0 $\rightarrow \overline{\text{Q}}=1$
Si Q=1 $\xrightarrow{\text{R :0}\rightarrow\text{1}}$ Q=0 $\rightarrow \overline{\text{Q}}=1$
$\Rightarrow$ l'application d'une impulsion de niveau haut sur R place la bascule dans l'état Q=0.
$\rightarrow$ opération de mise à 0 $\rightarrow$ RESET

– R=S=1
$\Rightarrow$ Q=$\overline{\text{Q}}$=0
$\rightarrow$ condition indésirable, puisque $\overline{\text{Q}}$ et Q doivent être l'inverse l'un de l'autre
$\rightarrow$ de plus, incertitude lorsque S et R reveiennent à 0
$\rightarrow$ R=S=1 ne doit pas servir

7.3 Bascule RS synchrone

Symbole

Réalisation

Table de vérité

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

La sortie est indicée 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 pas Q sauf pendant les quelques nanosecondes durant lesquelles CLK passe du niveau bas au niveau haut (pour les circuits actifs sur front montant).

Exemple :

7.4 Bascule JK

Les bascules JK sont seulement en fonctionnement synchrone.

Elles sont plus polyvalentes que les basculent RS, car elles n'ont pas d'état ambigu et $R = S = 1 \rightarrow Q_{N+1} = \overline{Q_N}$

Symbole

Réalisation

Table de vérité

J	K	H	Q_{N+1}	
X	X	–	Q_N	
0	0	↓	Q_N	mémorisation
1	0	↓	1	forçage à 1
0	1	↓	0	forçage à 0
1	1	↓	$\overline{Q}_N$	commutation

Remarque :

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

Tableau de Karnaugh et fonction logique

Q_{N+1}	J \ K	0	1	1	0
Q_N		1	1	0	0

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

Les bascules JK sont très courantes dans les systèmes numériques

Exemple :

7.5 Bascule D synchrone

Symbole

Table de vérité

H	D_N	Q_{N+1}
⎍	1	1
⎍	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

Idem pour la réalisation à partir des JK.

Remarque :

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

7.6 Bascule à verrouillage (*latch*)

Symbole

Fonctionnement

- quand CLK=0 → l'entrée D n'a aucun effet (mémorise) ;
- quand CLK=1 → Q suit les changements de D → la bascule est transparente.

Remarque :

Notez l'absence du symbole ▷ sur l'entrée CLK.

7.7 Bascule T

Symbole

Table de vérité

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

Cette bascule est utilisable uniquement en mode synchrone. Elle s'obtient par exemple à partir d'une bascule JK.

7.8 Entrées prioritaires asynchrones des bascules

La plupart des bascules synchrones possèdent des entrées prioritaires. 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.

Exemple :

$\overline{\text{Preset}}$	$\overline{\text{Clear}}$	Q
1	1	fonctionnement normal
0	1	1
1	0	0
0	0	ambigu, interdit

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	Preset
RAZ	RAU
Reset	Set
DC Clear	DC Set

Remarque :

Les entrées synchrones sont des niveaux de tension continue

7.9 Applications des bascules

7.9.1 Mémoire

→ mémorisation d'une information fugitive

Exemple :

Mémorisation d'une commande de marche

7.9.2 Antirebond pour commutateur

Exemple :

7.9.3 Synchronisation

Exemple :

Solution :

7.9.4 Détection d'une séquence d'entrée

Exemple :

Solution :

→ détection du sens de rotation d'un moteur.

7.9.5 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}.

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

Bascule JK

Bascule RS

Chapitre 8

Registres : stockage et transfert de données

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

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

8.1 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 déplacement 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.

8.2 Registre à 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.

8.3 Registre à décalage : 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.

8.4 Registre à décalage : 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.

8.5 Registre à décalage : écriture parallèle et lecture série

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

Chapitre 9

Les compteurs

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

Compteur : circuit séquentiel décrivant un cycle de comptage régulier ou quelconque.

9.1 Compteur asynchrone (à propagation)

9.1.1 Compteur asynchrone à cycle régulier

Exemple :

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

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.

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

c) Chronogramme

→ chaque bascule divise par deux la fréquence d'horloge qui alimente son entrée CLK : $f_D = \frac{f_{initale}}{16}$.

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

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.

9.1.2 Compteur asynchrone à modulo N $< 2^n$ (à cycle régulier)

a) Méthode

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 :

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 $Q_2Q_1Q_0 = 110$.

$\rightarrow$ 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

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

$Q_2Q_1Q_0 = 110$ est un état temporaire. Il existe mais pendant une dure très courte. C'est un état indésirable que l'on nomme parfois *glitch*.

c) Chronogramme

Remarque :

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

9.1.3 Comptage asynchrone dans un ordre quelconque (cycle irrégulier)

1ère méthode

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

Exemple :

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

2ème méthode

Utilisation des entrées RAZ et EAU.

Exemple :

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.

9.1.4 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.

Circuit interne

MR → Master Reset.

9.1.5 Décompteurs asynchrones

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

Exemple :

Décompteur modulo 8

Chronogramme :

9.1.6 Inconvénients des compteurs asynchrones

Chaque bascule introduit un retard de t_p (t_p=25ns). Comme les retards s'additionnent, à la $n^{ième}$ bascule, on a un retard de $n \times t_p$.

→ utilisation limitée en fréquence, particulièrement lorsque le nombre de bits est élevé.

→ 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).

9.2 Compteur synchrone (parallèle)

9.2.1 Réalisation

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

Exemple :

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 $T_2 = Q_1 Q_0$

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

9.2.3 Applications

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

: durée pendant laquelle les impulsions sont comptées

: RAZ met le compteur à zéro

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

Remarque :

 *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.*

B Horloge numérique

Chapitre 10

Méthodes d'étude des circuits séquentiels

Charles Lutwidge Dodgson
ou Lewis Carroll
1832-1898

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.