République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

U n i v e r s i t é A B D E R R A H M A N E M I R A
B e j a i a

Faculté de Technologie
Département de Génie Electrique

Polycopié de cours UEF 2.2.1.2 intitulé

Logique Combinatoire et Séquentielle

Dr AMIMEUR Hocine

Maître de Conférences

Chargé de Recherche

Laboratoire de maîtrise des énergies renouvelables

Année universitaire 2016/2017

Ce cours est destiné aux étudiants en deuxième année génie électrique.

Table des matières

Chapitre 1

Système de numération et les codes

1.1 Système de numération

Le système de numération traitant des nombres binaires est appelé système binaire ou système à base 2. Ce système comporte deux chiffres 0 et 1. Les chiffres binaires sont aussi appelés bits (venu de binary digit). Physiquement, dans les circuits électroniques numériques, un bit 0 est représenté par une tension basse (LOW) et un bit 1, par une tension haute (HIGH).

Les êtres humains ont toujours travaillé avec le système décimal. Ce denier est malheureusement difficile à adapter aux mécanismes numériques, car il est difficile de concevoir du matériel électronique fonctionnant sur dix plages de tensions différentes.

Tous les systèmes de numération ont une caractéristique de valeur ou de position (ou de poids). Par exemple, si on prend un système à base a, un nombre N peut s'écrire :

$$N = N_n N_{n-1} \cdots N_1 N_0(a)$$

Sa valeur décimale est :

$$N = N_n a^n + N_{n-1} a^{n-1} + \cdots + N_1 a^1 + N_0 a^0$$

$N_n a^n$ représente le chiffre le plus significatif (appelé chiffre de poids fort) et $N_0 a^0$ est le moins significatif (appelé chiffre de poids faible).

Exemple 1.1

$1^4 1^3 0^2 0^1 1^0_{(2)} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

$123{,}561_{(10)} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 5 \cdot 10^{-1} + 6 \cdot 10^{-2} + 1 \cdot 10^{-3}$

1.1.1 Base du système de numération

C'est le nombre de chiffres utilisé par le système de numération.

1. Système décimal : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Sa base est 10 ($B = 10$);

2. Système binaire : 0, 1. La base est $B = 2$;
3. Système octal : 0, 1, 2, 3, 4, 5, 6, 7, dont le base est 8 ($B = 8$);
4. Système hexadécimal : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, où la base est $B = 16$.

Pour les quatre bases usuelles :

Décimal ($B = 10$)	Binaire ($B = 2$)	Octal ($B = 8$)	Hexadécimal ($B = 16$)
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

1.1.2 Rang d'un chiffre de numération

Le rang d'un chiffre de base quelconque est égal l'exposant de base B associé à ce chiffre.

Exemple 1.2

$2^3 0^2 1^1 6^0_{(8)} = 2 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 6 \cdot 8^0$

Le rang de 2 est 3, le rang de 0 est 2, celui du chiffre 1 est 1 et le 6 est de rang 0.

1.2 Conversion d'un système de numération à un autre

1.2.1 Conversion base B vers base 10

Exemple 1.3

1. *Base 2 vers base 10*

$$\begin{aligned} 10111_{(2)} &= ?_{(10)} \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 16 + 0 + 4 + 2 + 1 = 23_{(10)} \end{aligned}$$

2. *Base 8 vers base 10*
   $701_{(8)} =?_{(10)}$
   $= 7 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0$
   $= 448 + 0 + 1 = 449_{(10)}$
3. *Base 16 vers base 10*
   $15A_{(16)} =?_{(10)}$
   $= 1 \cdot 16^2 + 5 \cdot 16^1 + 10 \cdot 16^0$
   $= 256 + 80 + 10 = 346_{(10)}$

1.2.2 Conversion base 10 vers base B

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).

Conversion base 10 vers base 2

Exemple 1.4

$57{,}4375_{(10)} =?_{(2)}$

Partie entière :

★ Première façon

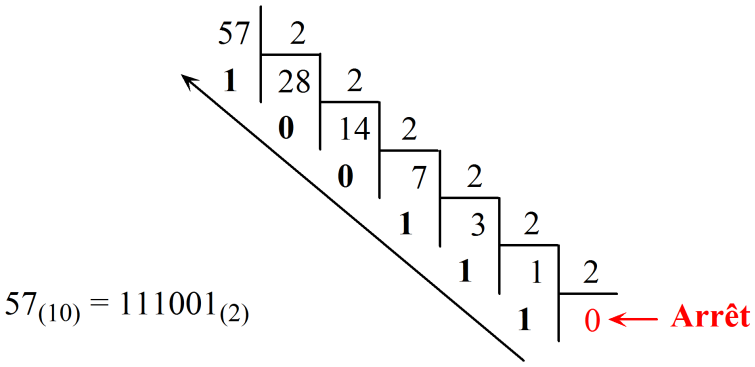
57	÷	2	=	28	reste	**1**
28	÷	2	=	14	reste	**0**
14	÷	2	=	7	reste	**0**
7	÷	2	=	3	reste	**1**
3	÷	2	=	1	reste	**1**
1	÷	2	=	0	reste	**1**
				↑		
				Arrêt		

$57_{(10)} = 111001_{(2)}$

★ Seconde façon

Partie fractionnaire :

0,4375	×	2	=	**0**,8750
0,8750	×	2	=	**1**,7500
0,7500	×	2	=	**1**,5000
0,5000	×	2	=	**1**,0000
				↑
				Arrêt



$0{,}4375_{(10)} = 0111_{(2)}$ donc $57{,}4375_{(10)} = 111001{,}0111_{(2)}$.

Conversion base 10 vers base 8

Exemple 1.5

$155_{(10)} = ?_{(8)}$

$$\begin{array}{rcrcrcl} 155 & \div & 8 & = & 19 & \text{reste} & \mathbf{3} \\ 19 & \div & 8 & = & 2 & \text{reste} & \mathbf{3} \\ 2 & \div & 8 & = & 0 & \text{reste} & \mathbf{2} \\ & & & & \uparrow & & \\ & & & & \text{Arrêt} & & \end{array}$$

$155_{(10)} = 233_{(8)}$

Conversion base 10 vers base 16

Exemple 1.6

$155_{(10)} = ?_{(16)}$

$$\begin{array}{rcrcrcl} 155 & \div & 16 & = & 9 & \text{reste} & \mathbf{B} \\ 9 & \div & 16 & = & 0 & \text{reste} & \mathbf{9} \\ & & & & \uparrow & & \\ & & & & \text{Arrêt} & & \end{array}$$

$155_{(16)} = 9\text{B}_{(16)}$

1.2.3 Conversion base 2 vers base 2^n

Conversion base 2 vers base 8 (2^3)

On regroupe le nombre en base 2 (binaire) en groupe de 3 bits.

Exemple 1.7

$111001110_{(2)} = ?_{(8)}$

$$\left.\begin{array}{rcl} 110_{(2)} & = & 6_{(8)} \\ 001_{(2)} & = & 1_{(8)} \\ 111_{(2)} & = & 7_{(8)} \end{array}\right| \Rightarrow 111001110_{(2)} = 716_{(8)}$$

Conversion base 2 vers base 16 (2^4)

On regroupe le nombre en base 2 en groupe de 4 bits pour avoir son équivalent en base 16 (2^4).

Exemple 1.8

$11011_{(2)} = ?_{(16)}$

$$\left.\begin{array}{rcl} 1011_{(2)} & = & B_{(16)} \\ 0001_{(2)} & = & 1_{(16)} \end{array}\right| \Rightarrow 00011011_{(2)} = 1B_{(16)}$$

1.2.4 Conversion base 2^n vers base 2

Conversion base 8 (2^3) **vers base 2**

Exemple 1.9

$572_{(8)} = ?_{(2)}$

$$\left.\begin{array}{rcl} 2_{(8)} & = & 010_{(2)} \\ 7_{(8)} & = & 111_{(2)} \\ 5_{(8)} & = & 101_{(2)} \end{array}\right| \Rightarrow 572_{(8)} = 101111010_{(2)}$$

Conversion base 16 (2^4) **vers base 2**

Exemple 1.10

$1D3_{(16)} = ?_{(2)}$

$$\left.\begin{array}{rcl} 3_{(16)} & = & 0011_{(2)} \\ D_{(16)} & = & 1101_{(2)} \\ 1_{(16)} & = & 0001_{(2)} \end{array}\right| \Rightarrow 1D3_{(16)} = 000111010011_{(2)} = 111010011_{(2)}$$

1.2.5 Conversion de la base 8 vers la base 16 et de la base 16 vers la base 8

Conversion base 8 (2^3) **vers base 16** (2^4)

On convertit le nombre en base 8 vers la base 2 par groupe de 3 bits, puis vers la base 16 par groupe de 4 bits.

Exemple 1.11

$330_{(8)} = ?_{(16)}$

Première étape (base 8 vers base 2) : $330_{(8)} = ?_{(2)}$

$$\left.\begin{array}{rcl} 0_{(8)} & = & 000_{(2)} \\ 3_{(8)} & = & 011_{(2)} \\ 3_{(8)} & = & 011_{(2)} \end{array}\right| \Rightarrow 330_{(8)} = 011011000_{(2)} = 11011000_{(2)}$$

Seconde étape (base 2 vers base 16) : $11011000_{(2)} = ?_{(16)}$

$$\left.\begin{array}{rcl} 1000_{(2)} & = & 8_{(16)} \\ 1101_{(2)} & = & \mathrm{D}_{(16)} \end{array}\right| \Rightarrow 11011000_{(2)} = \mathrm{D8}_{(16)}$$

Par suite : $330_{(8)} = \mathrm{D8}_{(16)}$

Conversion base 16 (2^4) vers base 8 (2^3)

On convertit le nombre en base 16 vers la base 2 par groupe de 4 bits, puis vers la base 8 par groupe de 3 bits.

Exemple 1.12

$13\mathrm{F}_{(16)} = ?_{(8)}$

Première étape (base 16 vers base 2) : $13F_{(16)} = ?_{(2)}$

$$\left.\begin{array}{rcl} \mathrm{F}_{(16)} & = & 1111_{(2)} \\ 3_{(16)} & = & 0011_{(2)} \\ 1_{(16)} & = & 0001_{(2)} \end{array}\right| \Rightarrow 13\mathrm{F}_{(8)} = 000100111111_{(2)} = 100111111_{(2)}$$

Seconde étape (base 2 vers base 8) : $100111111_{(2)} = ?_{(8)}$

$$\left.\begin{array}{rcl} 111_{(2)} & = & 7_{(8)} \\ 111_{(2)} & = & 7_{(8)} \\ 100_{(2)} & = & 4_{(8)} \end{array}\right| \Rightarrow 100111111_{(2)} = 477_{(8)}$$

Donc : $13\mathrm{F}_{(16)} = 477_{(8)}$

1.3 Arithmétique binaire

Les opérations d'addition, de soustraction, de division et de multiplication dans le système binaire se font de la même manière que dans le système décimal.

1.3.1 Addition

Exemple 1.13

$100101_{(2)} + 100011_{(2)} = ?_{(2)}$

				1	1	1		←	retenues
			1	0	0	1	0	1	← premier facteur
+			1	0	0	0	1	1	← second facteur
=	1	0	0	1	0	0	0	←	résultat (somme)

Alors, $100101_{(2)} + 100011_{(2)} = 1001000_{(2)}$.

1.3.2 Soustraction

Exemple 1.14

$1000_{(2)} - 0111_{(2)} =?_{(2)}$

	1	$^{1}0$	$^{1}0$	$^{1}0$	←	premier facteur
-	$^{1}0$	$^{1}1$	$^{1}1$	1	←	second facteur
=	0	0	0	1	←	résultat (différence)

Donc, $1000_{(2)} - 0111_{(2)} = 0001_{(2)} = 1_{(2)}$.

1.3.3 Multiplication

Exemple 1.15

$1101_{(2)} \times 101_{(2)} =?_{(2)}$

				1	1	0	1	←	multiplicande
×					1	0	1	←	multiplicateur
				$^{1}1$	1	0	1	←	premier produit
+			$^{1}0$	0	0	0	•	←	deuxième produit
+		$^{1}1$	1	0	1	•	•	←	troisième produit
=	1	0	0	0	0	0	1	←	résultat (produit final)

Donc, $1101_{(2)} \times 101_{(2)} = 1000001_{(2)}$

.

N.B : Les arithmétiques octal et hexadécimal se font de la même façon que dans les arithmétiques décimal et binaire.

1.4 Nombres entiers négatifs

Il existe trois types de représentation

1.4.1 Représentation par un bit de signe et une valeur absolue

Le premier bit indique le signe

$$\begin{cases} 0 \text{ pour le signe } +; \\ 1 \text{ pour le signe } -. \end{cases}$$

Le reste des bits représente la valeur absolue du nombre en base 2.

Exemple 1.16

$$\begin{array}{rcll} +3_{(10)} & = & 011_{(2)} & \text{où } 0 \longrightarrow \text{signe } - \text{ et } 11 \longrightarrow |3| \\ -3_{(10)} & = & 111_{(2)} & \text{où } 1 \longrightarrow \text{signe } + \text{ et } 11 \longrightarrow |3| \end{array}$$

$$\begin{array}{rcl} -8 & = & 11000 \\ +8 & = & 01000 \end{array}$$

Effectuer sur 4 bits les opérations suivantes :

1) $0_{(10)} - 1_{(10)} = ?_{(10)}$

$$\left.\begin{array}{rcl} 0_{(10)} & = & 0000 \\ -1_{(10)} & = & 1001 \end{array}\right| \Rightarrow \left|\begin{array}{rll} & 0000 & \\ + & 1001 & \\ \hline = & 1001 & \Leftrightarrow -1_{(10)} \text{ (le résultat est correct)} \end{array}\right.$$

2) $+1_{(10)} - 2_{(10)} = ?_{(10)}$

$$\left.\begin{array}{rcl} +1_{(10)} & = & 0001 \\ -2_{(10)} & = & 1010 \end{array}\right| \Rightarrow \left|\begin{array}{rll} & 0001 & \\ + & 1010 & \\ \hline = & 1011 & \Leftrightarrow -3_{(10)} \text{ (le résultat est faux)} \end{array}\right.$$

Inconvénient : la soustraction vue comme une addition bit à bit ne fonctionne pas.

1.4.2 Représentation par le complément à 1

Un nombre négatif est obtenu en complémentant tout ses bits (par exemple, le complément de 10010 est 01101).

Exemple 1.17

$$\begin{array}{rcl} +5_{(10)} & = & 101 \text{ et avec le bit de signe c'est : } 0101 \\ -5_{(10)} & = & 010 \text{ et en tenant compte du bit de signe : } 1010 \end{array}$$

Effectuer l'opération $+2_{(10)} - 3_{(10)} = ?_{(10)}$ en complément à 1 (cà1) sur 4 bits.

$$\left.\begin{array}{rcl} +2_{(10)} & = & 0010 \\ +3_{(10)} & = & 0111 \\ -3_{(10)} & = & 1000 \text{ en cà1} \end{array}\right| \Rightarrow \left|\begin{array}{rll} & 0010 & \\ + & 1000 & \\ \hline = & 1100 & \text{le cà1 de 100 est 001} \Leftrightarrow 1001 = -1_{(10)} \end{array}\right.$$

D'où : $+2_{(10)} - 3_{(10)} = -1_{(10)}$.

1.4.3 Représentation par le complément à 2

Un nombre négatif est obtenu en complémentant tout ses bits et en lui ajoutant 1 (c'est-à-dire cà2 = cà1 +1).

Exemple 1.18

$$\begin{array}{rl} +\,6_{(10)} = & 0110 \\ -\,6_{(10)} = & 1001 \quad \leftarrow \text{cà1} \\ +\,1 & \\ \hline = & 1010 \quad \leftarrow \text{cà2} \end{array} \Rightarrow -6_{(10)} = 1010 \text{ en cà2.}$$

1.5 Codes

1.5.1 Codage

Le codage est une opération qui établie une correspondance entre les éléments de deux ensembles différents (lettres, chiffres, des signes de ponctuation, etc.).

1.5.2 Codes pondérés

Codes binaire, octal, décimal et hexadécimal

Chaque chiffre en base B reçoit un poids proportionnel à sa position.

Exemple 1.19

$N_{(8)} = 153$

$$\begin{array}{ccc} 1 & 5 & 3 \\ \uparrow & \uparrow & \uparrow \\ 8^2 & 8^1 & 8^0 \end{array} \Rightarrow 153_{(8)} = 1 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 8^0.$$

$N_{(16)} = \text{D1E9}_{(16)} = \text{D} \cdot 16^3 + 1 \cdot 16^2 + \text{E} \cdot 16^1 + 9 \cdot 16^0.$

Code BCD (Binary Coded Decimal)

Chaque chiffre en décimal (base 10) est codé en 4 bits en binaire (base 2). Par exemple, $421_{(10)} = 0100\ 0010\ 0001_{(BCD)}$ au lieu de $110100101_{(2)}$ en binaire naturel.

- En BCD un nombre de n chiffres occupe $4n$ bits ;
- Pour l'addition en BCD, on ajoute le $6_{(10)}$ ($0110_{(BCD)}$) si le résultat est strictement supérieur à $9_{(10)}$ ($1001_{(BCD)}$) ;
- Pour la soustraction, il suffit de soustraire le $0110_{(BCD)}$ si le résultat est supérieur à $1001_{(BCD)}$;
- En code BCD, le code 8421 est plus répandu.

Décimal	BCD
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
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Code BCD : 8421

Les 4 bits pondèrent les chiffres 8, 4, 2 et 1. Ainsi, 0111 correspond à $0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 0 + 4 + 2 + 1 = 7$ donc $0111_{(BCD)} = 7_{(10)}$.

Il existe aussi d'autres codes BCD ; on cite à titre d'exemple les codes 2421 et 5421.

1.5.3 Codes non pondérés

Codes de Gray

Il encode les entiers de telle façon que le passage d'un nombre au nombre suivant ne change qu'un seul bit à la fois.

Le tableau suivant illustre l'exemple à 4 bits.

Décimal	Binaire	Gray
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

D'après le tableau précédent : $9_{(10)} = 1001_{(2)} = 1101_{(G)}$.

Exemple 1.20

$101001_{(2)} =?_{(G)}$ et $1011_{(G)} =?_{(2)}$

$$\begin{array}{l|cccccc|l} \text{Binaire}: & 1^{\rightarrow +} & 0^{\rightarrow +} & 1^{\rightarrow +} & 0^{\rightarrow +} & 0^{\rightarrow +} & 1 & \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \Rightarrow 101001_{(2)} = 111101_{(G)}. \\ \text{Gray}: & 1 & 1 & 1 & 1 & 0 & 1 & \end{array}$$

$$\begin{array}{l|cccc|l} \text{Gray}: & 1 & 0 & 1 & 1 & \\ & \downarrow + & \downarrow + & \downarrow + & \downarrow & \Rightarrow 1011_{(2)} = 1101_{(G)}. \\ \text{Binaire}: & 1^{\nearrow} & 1^{\nearrow} & 0^{\nearrow} & 1 & \end{array}$$

Code détecteur d'erreur par ajout d'un bit de parité

On ajoute un bit supplémentaire qui s'appel bit de parité. Le bit de parité est tel que :

$$\text{Parité paire (PP)}: \begin{cases} \text{PP} = 0, & \text{si le nombre des 1 est pair}; \\ \text{PP} = 1, & \text{si le nombre des 1 est impair}. \end{cases}$$

$$\text{Parité impaire (PI)}: \begin{cases} \text{PI} = 0, & \text{si le nombre des 1 est impair}; \\ \text{PI} = 1, & \text{si le nombre des 1 est pair}. \end{cases}$$

Le bit de parité paire (PP) est le plus utilisé.

Exemple 1.21

Code (information)	bit de parité (PP)
0 0 0 1	1
0 0 1 0	1
0 0 1 1	0
1 1 0 1	1
0 1 0 1	0
0 1 1 0	0

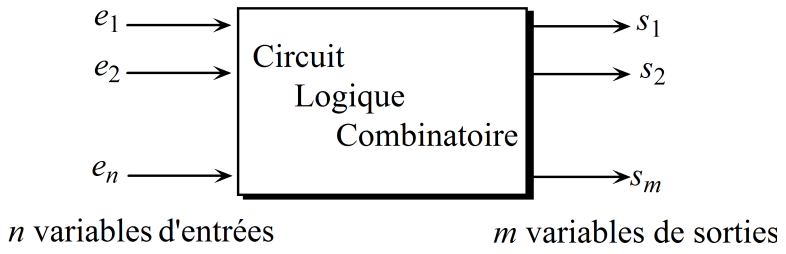
Chapitre 2

Circuits logiques combinatoires

2.1 Introduction

Un circuit gouverné par les règles de la logique combinatoire possède une ou plusieurs entrées, et une ou plusieurs sorties, et obéit à la propriété suivante :

L'état de la (ou des) sortie(s) à un instant donné ne dépend que du circuit et de la valeur des entrées à cet instant.



$$\begin{cases} s_1 = f\{e_1, e_2, \cdots, e_n\} \\ \vdots \\ s_n = f\{e_1, e_2, \cdots, e_n\} \end{cases}$$

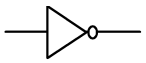
2.2 Fonctions logiques de base

2.2.1 Fonction complément (NON, NOT) (inverseur)

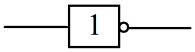
– Table de vérité

E	$S = f(E)$
A	$\overline{A}$
0	1
1	0

– Symbole (schéma logique)



Symbole Américains



Symbole Européens

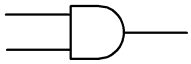
2.2.2 Fonction produit logique (intersection) (porte ET, AND)

– Table de vérité

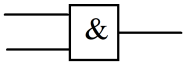
A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

La sortie ($S = A \cdot B$) est *vrai* si toutes les entrées (A et B) sont à l'état *vrai.*

– Symbole



Symbole Américains



Symbole Européens

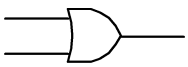
2.2.3 Fonction somme logique (réunion) (porte OU, OR)

– Table de vérité

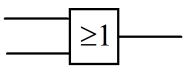
A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

La sortie ($S = A + B$) est *vrai* si au minimum une des entrées (A ou B) est à l'état *vrai.*

– Symbole



Symbole Américains



Symbole Européens

2.3 Propriétés relatives (NON, ET, OU)

Soient A, B et C trois variables logiques.

a) Commutativité

$$A \cdot B = B \cdot A$$

$$A + B = B + A$$

Démonstration en utilisant la table de vérité :

A	B	$A \cdot B$	$B \cdot A$	$A + B$	$B + A$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

b) Associativité

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

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

Démonstration par la table de vérité (T. V.) de $A \cdot (B \cdot C) = (A \cdot B) \cdot C$:

A	B	C	$B \cdot C$	$A \cdot (B \cdot C)$	$(A \cdot B)$	$(A \cdot B) \cdot C$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

c) Distributivité

$$\text{ET/OU (AND/OR) } A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

$$\text{OU/ET (OR/AND) } A + (B \cdot C) = (A + B) \cdot (A + C)$$

Démonstration par la T. V. de $A + (B \cdot C) = (A + B) \cdot (A + C)$:

A	B	C	$B \cdot C$	$A + (B \cdot C)$	$A + B$	$A + C$	$(A \cdot B) \cdot C$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

d) Complémentation

$$A \cdot \overline{A} = 0$$

$$A + \overline{A} = 1$$

e) Involution

$$\overline{\overline{A}} = A$$

f) Elément neutre

$$A \cdot 1 = A$$

$$A + 0 = A$$

g) Elément absorbant

$$A \cdot 0 = 0$$

$$A + 1 = 1$$

h) Idempotence

$$A \cdot A = A \text{ et } A \cdot A \cdot A \cdot \cdots \cdot A = A$$

$$A + A = A \text{ et } A + A + A + \cdots + A = A$$

i) De Morgan

1. *Premier théorème de De Morgan* (Complément d'une somme)

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

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

2. *Second théorème de De Morgan* (Complément d'un produit)

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

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

Démonstration par la T. V. de $\overline{A + B} = \overline{A} \cdot \overline{B}$:

A	B	$A + B$	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Exemple d'application des différentes propriétés

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

Démontrer que $F(A,B,C) = B \cdot \overline{C} + A$?

$$\begin{aligned} F(A,B,C) = & \; A \cdot B \cdot \overline{C} + B \cdot (A + \overline{C}) + \overline{\overline{A} + B + \overline{A} \cdot C} \\ = & \; A \cdot B \cdot \overline{C} + A \cdot B + B \cdot \overline{C} + \overline{\overline{A}} \cdot \overline{B} \cdot \overline{\overline{A} \cdot C} \\ = & \; A \cdot B \cdot \overline{C} + A \cdot B + B \cdot \overline{C} + A \cdot \overline{B} \cdot (A + \overline{C}) \\ = & \; A \cdot B \cdot \overline{C} + A \cdot B + B \cdot \overline{C} + A \cdot A \cdot \overline{B} + A \cdot \overline{B} \cdot \overline{C} \\ = & \; B \cdot \overline{C} \cdot (A + 1) + A \cdot (B + \overline{B}) + A \cdot \overline{B} \cdot \overline{C} \\ = & \; B \cdot \overline{C} + A + A \cdot \overline{B} \cdot \overline{C} \\ = & \; B \cdot \overline{C} + A \cdot (1 + \overline{B} \cdot \overline{C}) \\ = & \; B \cdot \overline{C} + A \end{aligned}$$

2.4 Fonctions NAND (NON ET) et NOR (NON OU)

2.4.1 Fonction NAND

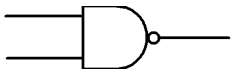
$$F(A,B) = \overline{A \cdot B}$$

– Table de vérité

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

La sortie ($S = \overline{A \cdot B}$) est à l'état *faux* si toutes les entrées (A et B) sont à l'état *vrai*.

– Symbole

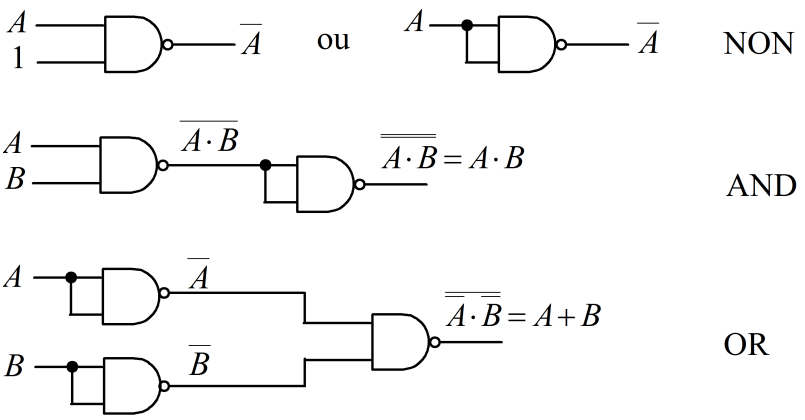


La fonction (l'opérateur) NAND permet la réalisation des trois fonctions de base comme indiqué par les logigrammes suivants :

NAND est donc un opérateur (fonction) complet.

2.4.2 Fonction NOR

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

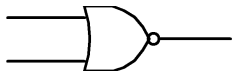


– Table de vérité

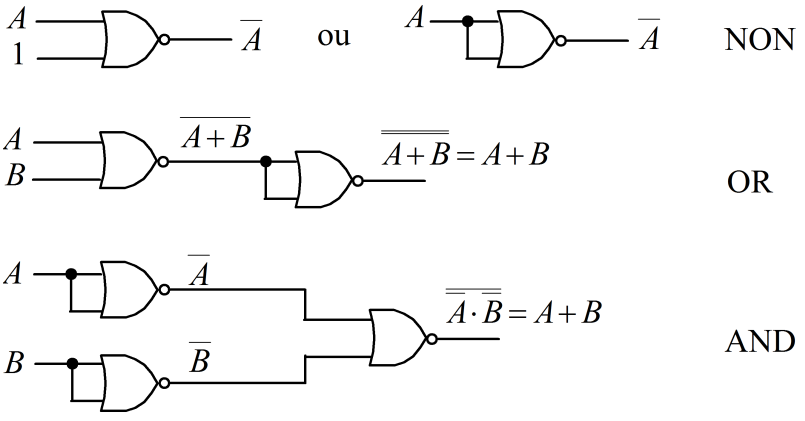
A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

La sortie ($S = \overline{A \cdot B}$) est *vrai* si toutes les entrées (A et B) sont à l'état *faux*.

– Symbole



La fonction (l'opérateur) NAND permet la réalisation des trois fonctions de base comme indiqué par les logigrammes suivants :



L'opérateur NOR permet la réalisation des trois fonctions logiques de base, c'est alors un opérateur logique complet.

2.5 Fonctions utiles

2.5.1 Fonction OU Exclusif (Exclusive OR, XOR)

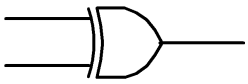
La fonction XOR est à deux entrées uniquement. La fonction XOR ne vaut 1 que si les deux entrées sont différentes.

$$F(A,B) = A \text{ XOR } B = A \oplus B$$

– Table de vérité

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

– Symbole



$$F(A,B) = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$$

2.5.2 Fonction coïncidence ou identité (NON OU Exclusif, Exclusive NOR, NXOR)

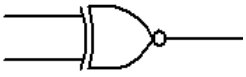
La fonction NXOR est à deux variables d'entrées seulement, qui prend la valeur de 1 dans les cas ou les deux entrées sont égales.

$$F(A,B) = A \text{ NXOR } B = \overline{A \oplus B} = A \odot B$$

– Table de vérité

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

– Symbole



2.6 Représentation des fonctions logiques

2.6.1 Formes canoniques

Les formes canoniques sont basées sur les *mintermes* et le *maxtermes* des fonctions logiques. Un minterme est le produit logique des variables de la même ligne de la table de vérité. Un maxterme s'obtient en faisant la somme logique des variables sous forme inversée de la même ligne de la table de vérité.

Exemple 2.1

Les mintermes m_i et les maxtermes M_A d'une fonction à 3 variables A, B et C sont :

Exemples des mintermes	A	B	C	Exemples des maxtermes
$m_{i0} = \overline{A} \cdot \overline{B} \cdot \overline{C} \rightarrow$	0	0	0	$\leftarrow M_{A0} = A + B + C$
	0	0	1	
$m_{i2} = \overline{A} \cdot B \cdot \overline{C} \rightarrow$	0	1	0	
	0	1	1	$\leftarrow M_{A3} = A + \overline{B} + \overline{C}$
	1	0	0	
	1	0	1	
$m_{i6} = A \cdot \overline{B} \cdot C \rightarrow$	1	1	0	
	1	1	1	$\leftarrow M_{A7} = \overline{A} + \overline{B} + \overline{C}$

2.6.2 Première forme canonique, disjonctive FNCD, ou somme des produits

Pour obtenir la FNCD, à chaque 1 de la variable de sortie, on fait correspondre le produit des variables d'entrée de la même ligne sous forme vraie. Par suite, on fait la somme des différents produits (mintermes).

Exemple 2.2

Soit la fonction $F(A,B,C)$ définie par la table de vérité suivante :

A	B	C	F	mintermes
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\leftarrow m_{i3} = \overline{A} \cdot B \cdot C$
1	0	0	0	
1	0	1	1	$\leftarrow m_{i5} = A \cdot \overline{B} \cdot C$
1	1	0	1	$\leftarrow m_{i6} = A \cdot B \cdot \overline{C}$
1	1	1	1	$\leftarrow m_{i7} = A \cdot B \cdot C$

$$F(A,B,C) = m_{i3} + m_{i5} + m_{i6} + m_{i7} = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

2.6.3 Deuxième forme canonique, conjonctive FNCC, ou produit des sommes

Pour obtenir la FNCC, à chaque 0 de la variable de sortie, on fait correspondre la somme des variables d'entrée de la même ligne sous forme inversée. Par suite, on fait le produit des différentes sommes (maxtermes).

Exemple 2.3

Soit la fonction $F(A,B)$ définie par la table de vérité suivante :

A	B	F	maxtermes
0	0	0	$\leftarrow M_{A0} = A + B$
0	1	1	
1	0	1	
1	1	0	$\leftarrow M_{A3} = \overline{A} + \overline{B}$

$$F(A,B) = M_{A0} \cdot M_{A3} = (A + B) \cdot (\overline{A} + \overline{B})$$

2.6.4 Troisième forme canonique ou forme "NON ET"

Elle est déduite de la première forme par utilisation du théorème de De Morgan et d'involution, afin d'exprimer la fonction à l'aide des portes NAND uniquement.

$$\text{Troisième FNC} = \overline{\overline{\text{Première FNC}}} \text{ (Involution)}$$

Exemple 2.4 : suite de l'exemple 2.2

$$\begin{aligned} F(A,B,C) = & \quad \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C & \leftarrow \text{FNCD} \\ = & \quad \overline{\overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C} & \leftarrow \text{Involution} \\ = & \quad \overline{\overline{A} \cdot B \cdot C} \cdot \overline{A \cdot \overline{B} \cdot C} \cdot \overline{A \cdot B \cdot \overline{C}} \cdot \overline{A \cdot B \cdot C} & \leftarrow \text{De Morgan} \end{aligned}$$

2.6.5 Quatrième forme canonique ou forme "NON OU"

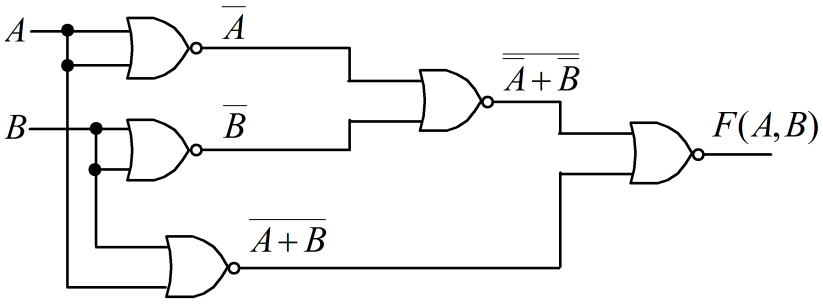
Elle est déduite de la deuxième forme par utilisation du théorème de De Morgan et d'involution, afin d'exprimer la fonction à l'aide des portes NOR seulement.

$$\text{Quatrième FNC} = \overline{\overline{\text{Deuxième FNC}}} \text{ (Involution)}$$

Exemple 2.5 : suite de l'exemple 2.3

$$\begin{aligned} F(A,B) = & \quad (A + B) \cdot (\overline{A} + \overline{B}) & \leftarrow \text{FNCC} \\ = & \quad \overline{\overline{(A + B) \cdot (\overline{A} + \overline{B})}} & \leftarrow \text{Involution} \\ = & \quad \overline{(A + B)} + \overline{(\overline{A} + \overline{B})} & \leftarrow \text{De Morgan} \end{aligned}$$

Le logigramme correspondant :



2.7 Simplification des fonctions logiques

2.7.1 Méthode algébrique

Exemple 2.6

Soit à démontrer les équations logiques suivantes par la méthode de simplification algébrique :

$$\begin{aligned} A \cdot B + A \cdot \overline{B} &= A && \dots (1) \\ A + A \cdot B &= A && \dots (2) \\ A + \overline{A} \cdot B &= A + B && \dots (3) \end{aligned}$$

Démonstration de la première relation :

$$\begin{aligned} (1) \Rightarrow A \cdot B + A \cdot \overline{B} &= A \cdot (B + \overline{B}) \\ &= A \cdot 1 \text{ car } B + \overline{B} = 1 \\ &= A \text{ puisque } A \cdot 1 = A \end{aligned}$$

Démonstration de la deuxième relation (théorème de l'absorption de $A \cdot B$) :

$$\begin{aligned} (2) \Rightarrow A + A \cdot B &= A \cdot (1 + B) \\ &= A \cdot 1 \text{ car } 1 + B = 1 \\ &= A \text{ puisque } A \cdot 1 = A \end{aligned}$$

Démonstration de la troisième relation (absorption de $\overline{A}$) :

$$\begin{aligned} (3) \Rightarrow A + \overline{A} \cdot B &= A + A \cdot B + \overline{A} \cdot B \text{ car d'après (2) } A = A + A \cdot B \\ &= A + B \cdot (A + \overline{A}) \\ &= A + B \cdot 1 \\ &= A + B \end{aligned}$$

2.7.2 Méthode graphique (méthode de Karnaugh)

Principe de construction de la table de Karnaugh

Soit une fonction à n variables d'entrées et soient p at q tel que :

$$p + q = n \Rightarrow \begin{cases} p = q = n/2, & \text{dans le cas } n \text{ pair} ; \\ |p - q| = 1, & \text{si } n \text{ est impair.} \end{cases}$$

Exemple 2.7

$$n = 4 \Rightarrow p = q = 4/2 = 2$$

$$n = 5 \Rightarrow \text{soit} \begin{cases} p = 3 \\ q = 2 \end{cases} \text{ ou } \begin{cases} p = 2 \\ q = 3 \end{cases}$$

La table de Karnaugh (T. K) comportera 2^p colonnes et 2^q lignes, qui seront remplies selon le code binaire réfléchi (code Gray), pour le quel deux nombres adjacents ne diffèrent que par un seul bit.

Exemple 2.8

$n = 2 \Rightarrow p = q = 1$ donc la table aura $2^1 = 2$ lignes et $2^1 = 2$ colonnes (N.B : $p = q \Rightarrow$la table est de forme carrée).

Soit la fonction $F(A,B)$ définie par la T. V suivante :

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Correspondant aux Tables de Karnaugh suivantes :

$A \diagdown B$	0	1
0	0	1
1	1	0

$\Leftrightarrow$

$A \diagdown B$	0	1
0	$\overline{A} \cdot \overline{B}$	$\overline{A} \cdot B$
1	$A \cdot \overline{B}$	$A \cdot B$

Exemple 2.9

$$n = 3 \text{ soit } \begin{cases} p = 1 \Rightarrow 2^1 = 2 & \text{colonnes ;} \\ q = 2 \Rightarrow 2^2 = 4 & \text{lignes.} \end{cases} \text{ ou } \begin{cases} p = 2 \Rightarrow 2^2 = 4 & \text{colonnes ;} \\ q = 2 \Rightarrow 2^1 = 2 & \text{lignes.} \end{cases}$$

N.B : n est impair, alors $|p - q| = 1 \Rightarrow$ la table de Karnaugh est de forme rectangulaire.

Soit la fonction $F(A,B,C)$ définie par la table de vérité suivante :

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

Correspondant aux tableaux de Karnaugh suivants :

Soit

$A\ B \diagdown C$	0	1
0 0	0	0
0 1	0	1
1 1	1	1
1 0	0	1

ou bien

$A \diagdown B\ C$	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	1	1	1

Exemple 2.10

Tableau à $n = 5$ variables; choisissant $p = 3 \Rightarrow 2^3 = 8$ colonnes et $q = 2 \Rightarrow 2^2 = 4$ lignes.

$A\ B \diagdown C\ D\ E$	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0	0	1	3	2	6	7	5	4
0 1	8	9	11	10	14	15	13	12
1 1	24	25	27	26	30	31	29	28
1 0	16	17	19	18	22	23	21	20

N.B : les cases sont remplies selon les numéros correspondant en décimal de ceux en binaire (exemple pour la case $14_{(10)} = 01110_{(2)}$).

La table de Karnaugh à $n = 5 \Rightarrow 32$ cases, correspond à deux tableaux à 16 cases, séparés par une ligne médiane. Cette ligne centrale peut être vue comme un miroir.

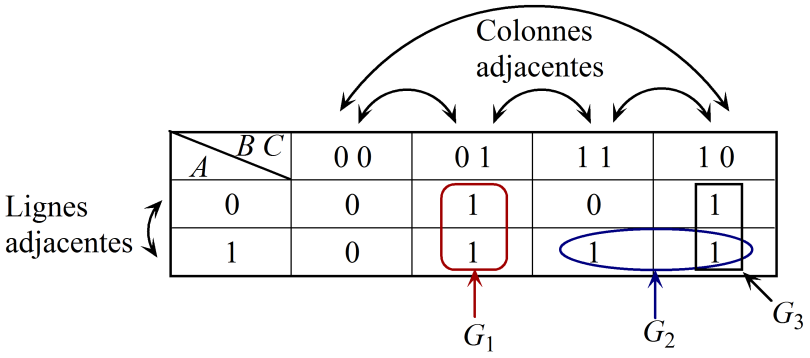
Méthode de simplification par le tableau de Karnaugh

Exemple 2.11

Soit à simplifier par la table de Karnaugh la fonction $F(A,B,C)$ définie par la table de vérité suivante :

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

Correspondant au tableau de Karnaugh suivant :



$$G_1 = \overline{B} \cdot C,\ G_2 = A \cdot B \text{ et } G_3 = B \cdot \overline{C}$$

$$F(A,B,C) = G_1 + G_2 + G_3 = \overline{B} \cdot C + A \cdot B + B \cdot \overline{C}$$

Les étapes de simplification à suivre sont :

1. Chercher à grouper tous les "1" (les "0" pour les maxtermes) pour former des boucles de 1, 2, 4, 8, etc. (2^n avec $n = 0, 1, 2, 3$, etc. , même si une ou plusieurs cases doivent faire partie de 2 ou plusieurs boucles ;
2. Tant que le nombre de cases regroupées est grand, l'expression sera plus simplifiée ;
3. Les groupements ne doivent contenir que des cases adjacentes ;
4. L'expression simplifiée comporte autant de termes qu'il y a de groupements.

Exemple 2.12

Soit à simplifier l'expression suivante :

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

La focntion F prend 1 dans les cases correspondant en décimal aux chiffres 4, 5, 6, 12, 13 et 14 ; qui s'écrit aussi sous la forme (somme des produits) suivante :

$$F(A,B,C,D) = \sum(4,5,6,12,13,14)$$

N.B : Pour les "0" (maxtermes, produit des sommes) on utilise le symbole $\prod$. Par suite, la fonction précédente peut aussi s'écrire :

$$F(A,B,C,D) = \prod(0,1,2,3,7,8,9,10,11,15)$$

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	1	0	1
1 1	1	1	0	1
1 0	0	0	0	0

G_1 G_2

$$G_1 = B\cdot\overline{C} \text{ et } G_2 = B\cdot\overline{D}$$

$$F(A,B,C) = G_1 + G_2 = B\cdot\overline{C} + B\cdot\overline{D}$$

2.8 Additionneur et soustracteur

2.8.1 Demi-additionneur

L'additionneur binaire portant sur un bit unique mène aux 4 cas notés dans la table de vérité suivante :

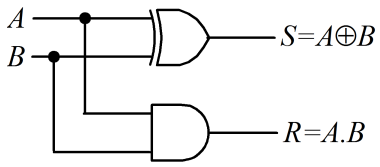
A (Cumulande)	B (Cumulateur)	S (Somme)	R (Retenue)	Mintermes
0	0	0	0	
0	1	1	0	$\overline{A} \cdot B$
1	0	1	0	$A \cdot \overline{B}$
1	1	0	1	$A \cdot B$

Les équations caractéristiques sont :

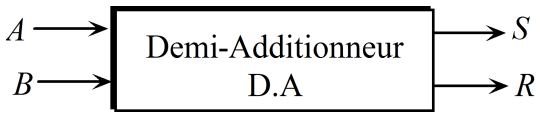
$$S = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$$

$$R = A \cdot B$$

Le logigramme correspondant :

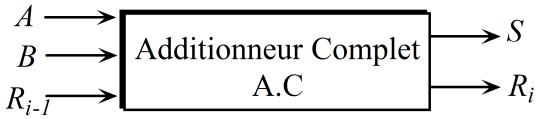


N.B : *Le demi-additionneur ne tient pas compte de la retenue précédente.*



2.8.2 Additionneur complet

La représentation de l'additionneur complet est donnée par le schéma suivant, où R_i indique la retenue et R_{i-1} la retenue précédente.



L'analyse du fonctionnement de ce dernier est illustrée par la table de vérité suivante :

A	B	R_{i-1}	S	R_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Les équations logiques des sorties S et R_i basées sur les mintermes :

$$S = \overline{A} \cdot \overline{B} \cdot R_{i-1} + \overline{A} \cdot B \cdot \overline{R_{i-1}} + A \cdot \overline{B} \cdot \overline{R_{i-1}} + A \cdot B \cdot R_{i-1}$$

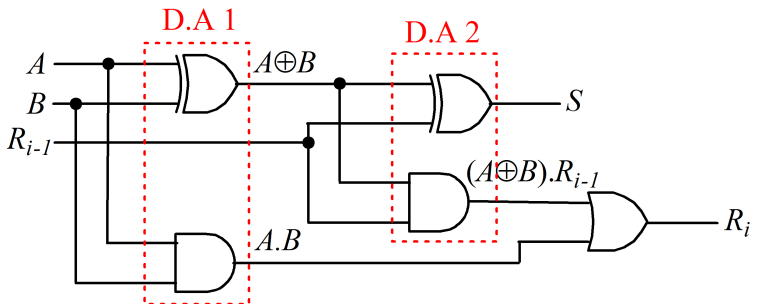
$$R_i = \overline{A} \cdot B \cdot R_{i-1} + A \cdot \overline{B} \cdot R_{i-1} + A \cdot B \cdot \overline{R_{i-1}} + A \cdot B \cdot R_{i-1}$$

Simplification de ces dernières :

$$\begin{aligned} S &= \overline{A} \cdot \overline{B} \cdot R_{i-1} + \overline{A} \cdot B \cdot \overline{R_{i-1}} + A \cdot \overline{B} \cdot \overline{R_{i-1}} + A \cdot B \cdot R_{i-1} \\ &= (\overline{A} \cdot \overline{B} + A \cdot B) \cdot R_{i-1} + (\overline{A} \cdot B + A \cdot \overline{B}) \cdot \overline{R_{i-1}} \\ &= (\overline{A \oplus B}) \cdot R_{i-1} + (A \oplus B) \cdot \overline{R_{i-1}} \\ \Rightarrow S &= A \oplus B \oplus R_{i-1} \end{aligned}$$

$$\begin{aligned} R_i &= \overline{A} \cdot B \cdot R_{i-1} + A \cdot \overline{B} \cdot R_{i-1} + A \cdot B \cdot \overline{R_{i-1}} + A \cdot B \cdot R_{i-1} \\ &= (\overline{A} \cdot B + A \cdot \overline{B}) \cdot R_{i-1} + A \cdot B \cdot (R_{i-1} + \overline{R_{i-1}}) \\ \Rightarrow R_i &= (A \oplus B) \cdot R_{i-1} + A \cdot B \end{aligned}$$

Logigramme de l'additionneur complet :



2.8.3 Demi-soustracteur

Le soustracteur binaire portant sur un bit unique mène aux 4 cas présentés par la table de vérité suivante :

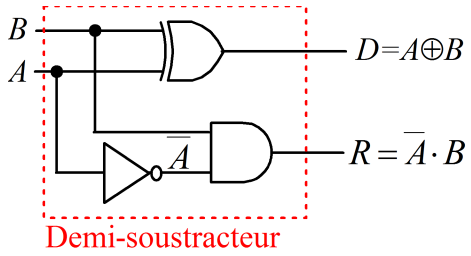
A	B	D (Différence)	R (Retenue)	Mintermes
0	0	0	0	
0	1	1	1	$\overline{A} \cdot B$
1	0	1	0	$A \cdot \overline{B}$
1	1	0	0	

Les équations logiques sont :

$$D = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$$

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

Le logigramme du demi-soustracteur :



2.8.4 Soustracteur complet

L'analyse de fonctionnement du soustracteur complet est illustrée par la table de vérité suivante :

A	B	R_{i-1}	D	R_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Les équations logiques des sorties D et R_i en utilisant les mintermes :

$$D = \overline{A} \cdot \overline{B} \cdot R_{i-1} + \overline{A} \cdot B \cdot \overline{R_{i-1}} + A \cdot \overline{B} \cdot \overline{R_{i-1}} + A \cdot B \cdot R_{i-1}$$

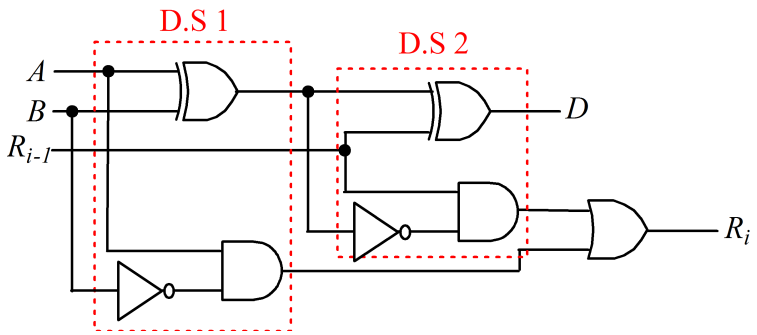
$$R_i = \overline{A} \cdot \overline{B} \cdot R_{i-1} + \overline{A} \cdot B \cdot \overline{R_{i-1}} + \overline{A} \cdot B \cdot R_{i-1} + A \cdot B \cdot R_{i-1}$$

Simplification de équations précédentes:

$$\begin{aligned} D &= \overline{A} \cdot \overline{B} \cdot R_{i-1} + \overline{A} \cdot B \cdot \overline{R_{i-1}} + A \cdot \overline{B} \cdot \overline{R_{i-1}} + A \cdot B \cdot R_{i-1} \\ &= (\overline{A} \cdot \overline{B} + A \cdot B) \cdot R_{i-1} + (\overline{A} \cdot B + A \cdot \overline{B}) \cdot \overline{R_{i-1}} \\ &= (\overline{A \oplus B}) \cdot R_{i-1} + (A \oplus B) \cdot \overline{R_{i-1}} \\ \Rightarrow D &= A \oplus B \oplus R_{i-1} \end{aligned}$$

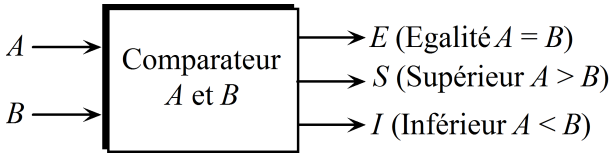
$$\begin{aligned} R_i &= \overline{A} \cdot \overline{B} \cdot R_{i-1} + \overline{A} \cdot B \cdot \overline{R_{i-1}} + \overline{A} \cdot B \cdot R_{i-1} + A \cdot B \cdot R_{i-1} \\ &= (\overline{A} \cdot \overline{B} + A \cdot B) \cdot R_{i-1} + \overline{A} \cdot B \cdot (R_{i-1} + \overline{R_{i-1}}) \\ \Rightarrow R_i &= (\overline{A \oplus B}) \cdot R_{i-1} + \overline{A} \cdot B \end{aligned}$$

Logigramme du soustracteur complet :



2.9 Comparateur

La représentation du comparateur entre 2 nombres A et B est donnée par le schéma suivant :



La table de vérité suivante donne l'analyse du fonctionnement d'un comparateur à 1 bit :

A	B	E	S	I
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

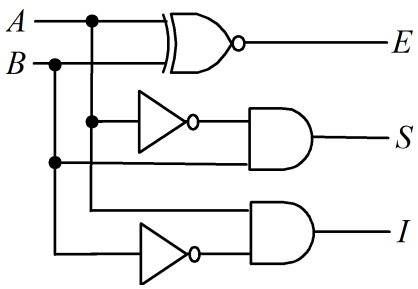
Les équations caractéristiques des sorties E, S et I en prenant les mintermes :

$$E = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B}$$

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

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

Le logigramme correspondant au comparateur à 1 bit :



2.10 Transcodeur, codeur et décodeur

2.10.1 Transcodeur

Un transcodeur est un circuit combinatoire permettant de passer d'un code à un autre.

Exemple 2.13 Concevoir un transcodeur binaire vers Gray à 4 bits

Les variables A, B, C et D représentent le nombre en code binaire, et X, Y, Z et T représentent le même nombre en code Gray.

Analyse par la table de vérité :

N° Décimal	$A\ B\ C\ D$	$X\ Y\ Z\ T$
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 équations caractéristiques des sorties en fonctions des entrées (en utilisant les tableaux de Karnaugh) :

1. $X = f(A,B,C,D)$?

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	0	0	0	0
11	1	1	1	1
1 0	1	1	1	1

$$X = A$$

2. $Y = f(A,B,C,D)$?

$$Y = G_1 + G_2 = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$$

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	1	1	1
11	0	0	0	0
1 0	1	1	1	1

G_1 (0 1 ligne) ; G_2 (1 0 ligne)

3. $Z = f(A,B,C,D)$?

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	0	1	1
0 1	1	1	0	0
11	1	1	0	0
1 0	0	0	1	1

G_2 (1 1 colonne + 0 1 et 1 0 colonnes) ; G_1 (0 1 et 11 lignes, 0 0 et 1 0 colonnes)

$$Z = G_1 + G_2 = B \cdot \overline{C} + \overline{B} \cdot C = B \oplus C$$

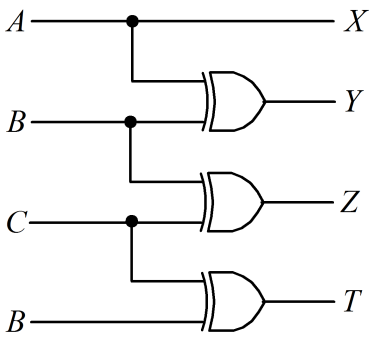
4. $T = f(A,B,C,D)$?

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	1	0	1
0 1	0	1	0	1
11	0	1	0	1
1 0	0	1	0	1

G_2 (0 1 et 1 0 colonnes) ; G_1 (0 1 colonne, 0 0 et 1 0 lignes)

$$Z = G_1 + G_2 = \overline{C} \cdot D + C \cdot \overline{D} = C \oplus D$$

Le logigramme est le suivant :



2.10.2 Codeur

Un codeur (ou encodeur) reçoit un niveau valide à l'une de see entrées, représentant par exemple un chiffre, une lettre, etc. Il le convertit en une sortie codée (par exemple en binaire ou en BCD).

Exemple 2.14 Codeur décimal - BCD

Il permet de traduire un nombre écrit en décimal, en son équivalent binaire.

La table de vérité est la suivante :

Décimal N°	BCD B_3	B_2	B_1	B_0
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
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Les expressions logiques sont :

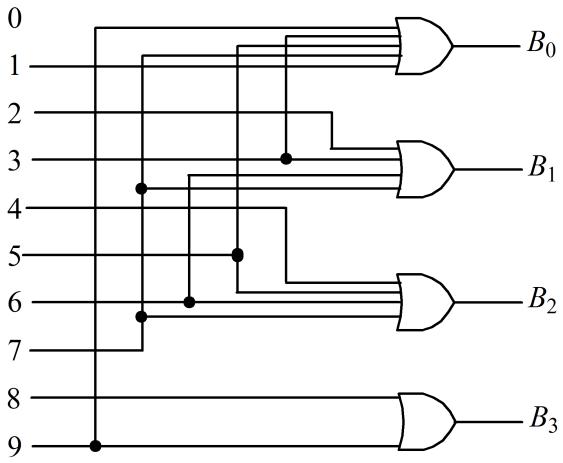
$$B_3 = 8 + 9$$

$$B_2 = 4 + 5 + 6 + 7$$

$$B_1 = 2 + 3 + 6 + 7$$

$$B_0 = 1 + 3 + 5 + 7 + 9$$

Le schéma logique correspondant est donné par la figure suivante :



2.10.3 Décodeur

Un décodeur est un circuit logique réalisant la fonction inverse du codeur.

Exemple 2.15 Décodeur BCD - décimal

Le décodeur BCD - décimal transforme un nombre écrit en BCD en son équivalent décimal.

La table de vérité est la suivante (où les case vides correspondent à des 0) :

N°	B_3	B_2	B_1	B_0	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
0	0	0	0	0	1									
1	0	0	0	1		1								
2	0	0	1	0			1							
3	0	0	1	1				1						
4	0	1	0	0					1					
5	0	1	0	1						1				
6	0	1	1	0							1			
7	0	1	1	1								1		
8	1	0	0	0									1	
9	1	0	0	1										1

Afin d'obtenir les équations de sortie, on établit un tableau de Karnaugh de 16 cases pour chaque S_i ($i = 0, \cdots, 9$). On complète les cases allant de 10 à 15 par des $\emptyset$ qu'on choisit librement (0 ou 1) afin d'avoir l'équation la plus simplifiée. On obtient :

$$S_0 = \overline{B_0} \cdot \overline{B_1} \cdot \overline{B_2} \cdot \overline{B_3}$$

$$S_1 = B_0 \cdot \overline{B_1} \cdot \overline{B_2} \cdot \overline{B_3}$$

$$S_2 = \overline{B_0} \cdot B_1 \cdot \overline{B_2}$$

$$S_3 = B_0 \cdot B_1 \cdot \overline{B_2}$$

$$S_4 = \overline{B_0} \cdot \overline{B_1} \cdot B_2$$

$$S_5 = B_0 \cdot \overline{B_1} \cdot B_2$$

$$S_6 = \overline{B_0} \cdot B_1 \cdot B_2$$

$$S_7 = B_0 \cdot B_1 \cdot B_2$$

$$S_8 = \overline{B_0} \cdot B_3$$

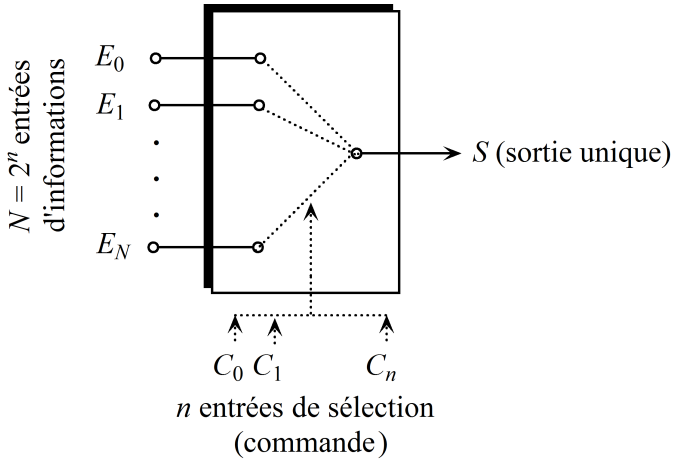
$$S_9 = B_0 \cdot B_3$$

On réalise ensuite le logigramme qui nécessite 4 inverseurs et 9 portes AND.

2.11 Circuits d'aiguillage d'information

2.11.1 Multiplexeur

C'est un circuit combinatoire permettant de réaliser un aiguillage de l'une des entrées en une sortie unique, dont le représentation est donnée par le schéma suivant :



N.B : Pour $N = 2^n$ entrées (avec n entier positif) correspond n éléments binaire de commande (sélection).

Exemple 2.16 Multiplexeur 2 vers 1

Il s'agit d'un multiplexeur à 2 (2^1) entrées (qu'on note E_0 et E_1), qui nécessite une (1) entrée de commande (qu'on nomme C_0) et une seule sortie (S).

Son fonctionnement en aiguillage se résume par :

$$\begin{cases} S = E_0 \text{ si } C_0 = 0\,; \\ S = E_1 \text{ si } C_0 = 1. \end{cases}$$

L'analyse du fonctionnement est portée la table de vérité suivante :

C_0	E_1	E_0	S	
0	0	0	0	$S = E_0$
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	$S = E_1$
1	0	1	0	
1	1	0	1	
1	1	1	1	

Donc la table de vérité précédente peut être simplifiée :

C_0	S
0	E_0
1	E_1

Par suite, l'équation caractéristique est :

$$S = \overline{C_0} \cdot E_0 + C_0 \cdot E_1$$

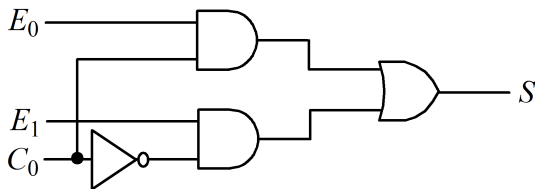
En utilisant le tableau de Karnaugh :

C_0 \ $E_1 E_0$	0 0	0 1	1 1	1 0
0	0	1	1	0
1	0	0	1	1

G_1 G_2

$$S = G_1 + G_2 = \overline{C_0} \cdot E_0 + C_0 \cdot E_1$$

Le logigramme du multiplexeur 2 vers 1 :



Exemple 2.17 Multiplexeur 4 vers 1

C'est un multiplexeur à 4 (2^2) entrées (E_0, E_1, E_2 et E_3), qui nécessite 2 entrées de commande (C_0 et C_1) et une seule sortie (S).

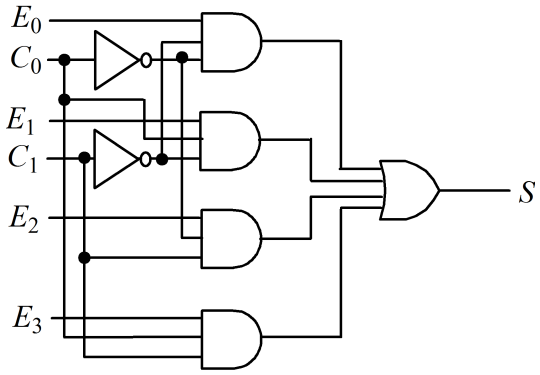
Son fonctionnement est donné par la table de vérité simplifiée suivante :

C_0	C_1	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

Son équation logique est :

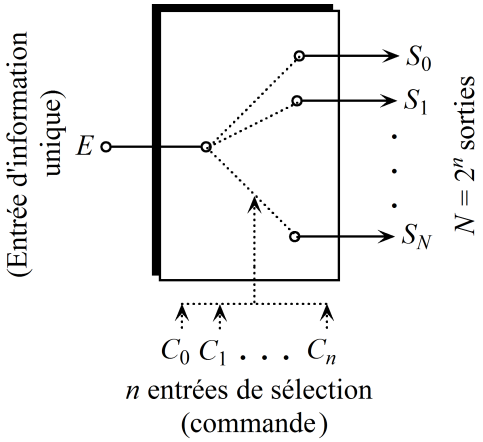
$$S = \overline{C_0} \cdot \overline{C_1} \cdot E_0 + C_0 \cdot \overline{C_1} \cdot E_1 + \overline{C_0} \cdot C_1 \cdot E_2 + C_0 \cdot C_1 \cdot E_3$$

Son logigramme est illustré par la figure suivante :



2.11.2 Démultiplexeur

Le démultiplexeur réalise l'opération inverse de celle du multiplexeur. Il comporte une seule entrée d'information (ou de données) E, n entrées de commande C_i avec $i = 0,1,...,n$ (appelées aussi entrées d'adresse ou de sélection) et $N = 2^n$ sorties (S_0, S_1, ... , S_N). Le schéma représentatif du démultiplexeur est illustré par la figure suivante :



Exemple 2.18 Démultiplexeur 1 vers 4

C'est un démultiplexeur à 4 (2^2) sorties (S_0, S_1, S_2 et S_3), qui nécessite 2 entrées de commande (C_0 et C_1) et une seule entrée(E).

Le fonctionnement du démultiplexeur est donné par la table de vérité simplifiée ($E =$

$0 \Rightarrow$ aucune information) suivante :

	C_0	C_1	S_0	S_1	S_2	S_3
$E = 1$	0	0	1	0	0	0
	0	1	0	1	0	0
	1	0	0	0	1	0
	1	1	0	0	0	1

Les équations caractéristiques des diverse sorties en prenant les "1" de la table de vérité précédente :

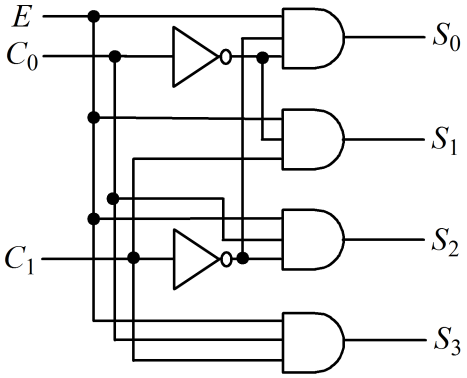
$$S_0 = \overline{C_0} \cdot \overline{C_1} \cdot E$$

$$S_1 = \overline{C_0} \cdot C_1 \cdot E$$

$$S_2 = C_0 \cdot \overline{C_1} \cdot E$$

$$S_3 = C_0 \cdot C_1 \cdot E$$

Le logigramme du démultiplexeur 4 vers 1 est représenté par le figure ci après :



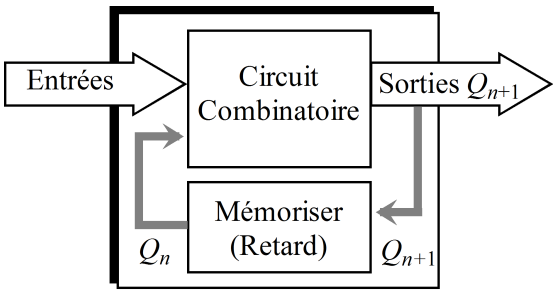
Chapitre 3

Circuits logiques séquentiels

3.1 Introduction

On a vu que dans un système combinatoire les sorties dépendent uniquement des entrées, pour les mêmes entrées on a toujours les mêmes sorties. Dans un système séquentiel, les sorties dépendent non seulement des entrées mais aussi des sorties précédentes (*le système mémorise les sorties précédentes*), donc pour les mêmes entrées on n'a pas toujours les mêmes sorties.

Tout circuit séquentiel est constitué d'un circuit combinatoire couplé à une mémoire comme indiqué par la figure suivante :



Circuit séquentiel

Dans la table de vérité on trouve, en plus des entrées, les sorties à l'état précédent.

$$Q_{n+1} = f(\text{Entrées},Q_n)$$

On distingue deux types de systèmes logiques séquentiels :

1. Les circuits séquentiels asynchrones, dans lesquels les sorties évoluent dès qu'il y a un changement sur l'une des entrées ;
2. Les circuits séquentiels synchrones, dans lesquels les sorties ne peuvent évoluer que si un signal d'horloge est actif.

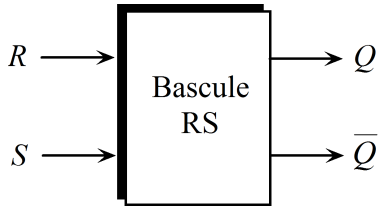
Les bascules sont les circuits logiques de base de la logique séquentielle, elles possèdent deux sorties complémentaires Q et $\overline{Q}$. On note Q_{n+1} la sortie à l'état actuel et Q_n la sortie

à l'état précédent (mémorisée). Il existe des bascules asynchrones (sans horloge) et des bascules synchrones (avec horloge).

3.2 Bascules asynchrones

3.2.1 Bascule RS

Une bascule RS comporte deux entrées : Une entrée R (Reset) de mise à zéro et une entrée S (Set) de mise à un. Schématisé sous la forme suivante :



Son fonctionnement se résume ainsi :

R	S	Q_{n+1}	Etat
0	0	Q_n	maintient (état mémoire)
0	1	1	mise à 1
1	0	0	mise à 0
1	1	$\emptyset$	indeterminé (cas interdit)

La table de vérité est donc :

R	S	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	$\emptyset$
1	1	1	$\emptyset$

Les équations logiques correspondant,

– En utilisant les mintermes :

R \ $S\,Q_n$	0 0	0 1	1 1	1 0
0	0	1	1	1
1	0	0	$\varnothing$	$\varnothing$

G_2 G_1

$$Q_{n+1} = G_1 + G_2 = S + R \cdot \overline{Q_n}$$

R \ $S\,Q_n$	0 0	0 1	1 1	1 0
0	0	1	1	1
1	0	0	∅	∅

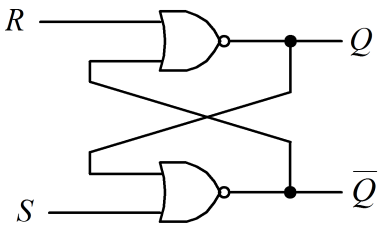
G_2 G_1

– En utilisant les maxtermes :

$$Q_{n+1} = G_1 \cdot G_2 = \overline{R} \cdot (S + Q_n)$$

Le logigramme avec uniquement les portes NOR :

$$\begin{aligned} Q_{n+1} &= \overline{R} \cdot (S + Q_n) \\ &= \overline{\overline{\overline{R} \cdot (S + Q_n)}} \\ &= \overline{\overline{\overline{R}} + \overline{(S + Q_n)}} \\ &= \overline{R} + \overline{(S + Q_n)} \end{aligned}$$



3.2.2 Bascule JK

La bascule JK vient prendre en charge le cas indéterminé de la bascule RS, son fonctionnement est défini par la table suivante:

J	K	Q_{n+1}	Etat
0	0	Q_n	maintient (état mémoire)
0	1	0	mise à 0
1	0	1	mise à 1
1	1	$\overline{Q_n}$	basculement (changement)

Sa table de vérité est alors :

J	K	Q_n	Q_{n+1}
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	1
1	1	1	0

J \ KQ_n	0 0	0 1	1 1	1 0
0	0	1	0	0
1	1	1	0	1

G_2 G_1

L'équation logique simplifiée par le tableau de Karnaugh en prenant les 1 :

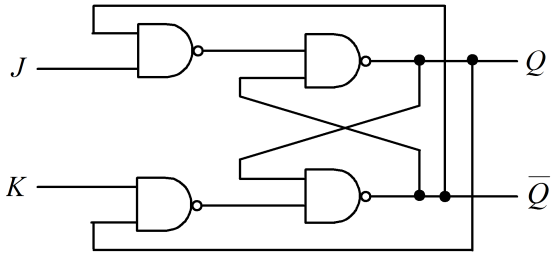
$$Q_{n+1} = G_1 + G_2 = J \cdot \overline{Q_n} + \overline{K} \cdot Q_n$$

Si on prend les 0, on aura :

$$Q_{n+1} = (J + Q_n) \cdot (\overline{K} + \overline{Q_n}) \Longrightarrow \overline{Q_{n+1}} = \overline{J} \cdot \overline{Q_n} + K \cdot Q_n$$

Le logigramme utilisant seulement les portes NAND :

$$\begin{aligned} Q_{n+1} &= J \cdot \overline{Q_n} + \overline{K} \cdot Q_n \\ &= J \cdot \overline{Q_n} + \overline{K} \cdot Q_n + Q_n \cdot \overline{Q_n} \\ &= J \cdot \overline{Q_n} + Q_n \cdot (\overline{K} + \overline{Q_n}) \\ &= \overline{\overline{J \cdot \overline{Q_n} + Q_n \cdot (\overline{K} + \overline{Q_n})}} \\ &= \overline{\overline{J \cdot \overline{Q_n}} \cdot \overline{Q_n \cdot (\overline{K} + \overline{Q_n})}} \\ &= \overline{\overline{J \cdot \overline{Q_n}} \cdot \overline{Q_n} \cdot \overline{(\overline{K} \cdot Q_n)}} \end{aligned}$$



3.2.3 Bascule D

La bascule D possède une seule entrée. Son principe de fonctionnement est donné par les tables suivantes :

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

$\Rightarrow$

D	Q_{n+1}
0	0
1	1

$\Rightarrow Q_{n+1} = D$

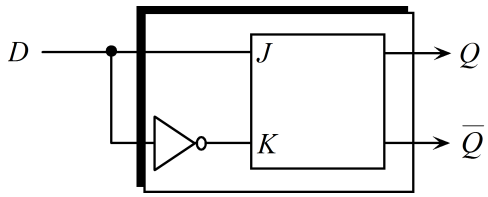
Le logigramme correspondant :

$$Q_{n+1} = D \cdot (Q_n + \overline{Q_n}) = D \cdot Q_n + D \cdot \overline{Q_n}$$

Par identification à l'équation de la bascule JK :

$$Q_{n+1} = J \cdot \overline{Q_n} + \overline{K} \cdot Q_n \Rightarrow \begin{cases} \text{J} = \text{D} \\ \text{K} = \overline{D} \end{cases}$$

Par suite, le schéma logique de la bascule D est le suivant :



Sous cette forme, la bascule D n'a pas grand intérêt puisqu'il s'agit d'un bloc fonctionnel qui ne fait que recopier son entrée, en permanence. On verra plus loin que son utilité apparaît avec le signal "d'horloge" (version synchrone).

3.2.4 Bascule T

La bascule T tire son nom du terme anglais 'toggle'. Si son entrée T est active, elle bascule à chaque impulsion d'horloge d'où son nom. Si son entrée T est inactive, elle conserve son état , comme indiqué par la table de fonctionnement suivante :

T	Q_n	Q_{n+1}	Etat
0	0	0	maintient
0	1	1	maintient
1	0	1	basculement
1	1	0	basculement

$\Rightarrow$

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

$\Rightarrow Q_{n+1} = \overline{T} \cdot Q_n + T \cdot \overline{Q_n}$

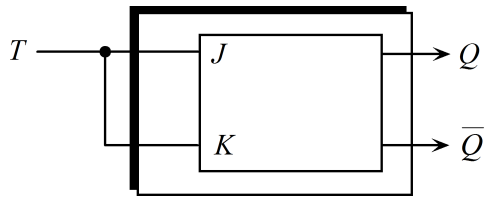
L'équation caractéristique est :

$$Q_{n+1} = T \cdot \overline{Q_n} + \overline{T} \cdot Q_n = T \oplus Q_n$$

Par identification à l'équation de la bascule JK, on obtient :

$$Q_{n+1} = J \cdot \overline{Q_n} + \overline{K} \cdot Q_n \Rightarrow \begin{cases} \text{J} = \text{T} \\ \text{K} = \text{T} \end{cases}$$

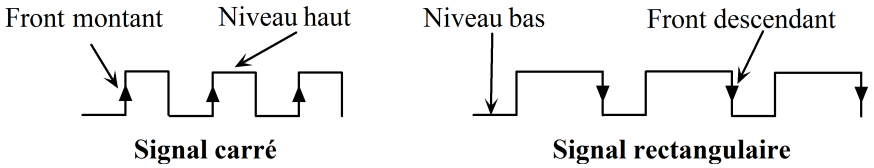
Le logigramme de la bascule T est :



3.3 Horloge et bascules synchrones

3.3.1 Horloge

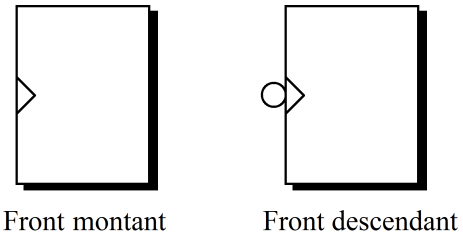
L'horloge génère des impulsions périodiques, qui se présentent sous forme de signaux carrés ou rectangulaires.



Les bascules sont conçues pour changer d'état, soit sur front montant, soit sur front descendant du signal d'horloge, et restent stables entre deux impulsions successives.

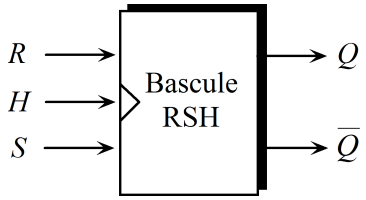
- Le passage du niveau bas (état 0) au niveau haut (état 1) est appelé : front montant ;
- Le passage du niveau haut au niveau bas est appelé : front descendant.

La représentation de l'entrée d'horloge et telle qu'illustrée par la figure suivante :



3.3.2 Bascule RS synchrone (RST ou RSH)

On rajoute une entrée d'horloge H (validation) à la bascule RS. Schématisé pour le cas front montant sous la forme suivante :



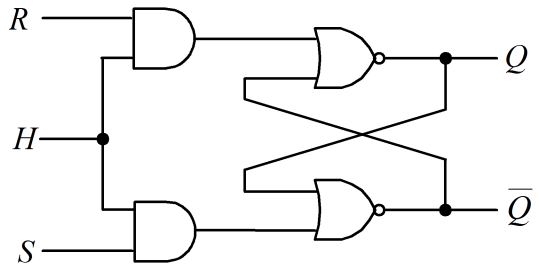
Pour son fonctionnement :

$$\begin{cases} \text{Si H=0} & \forall R \text{ et } \forall S \Rightarrow Q_{n+1} = Q_n \text{ mémorisation } (\forall \text{ quoi qu'il soit}) ; \\ \text{Si H=1} & \Rightarrow \text{retournement au fonctionnement normal de la bascule RS.} \end{cases}$$

Son fonctionnement se résume ainsi :

H	R	S	Q_{n+1}	Etat
0	$\forall$	$\forall$	Q_n	maintient (mémorisation)
1	0	0	Q_n	maintient (mémorisation)
	0	1	1	mise à 1
	1	0	0	mise à 0
	1	1	$\emptyset$	indeterminé (cas interdit)

Le logigramme correspondant :



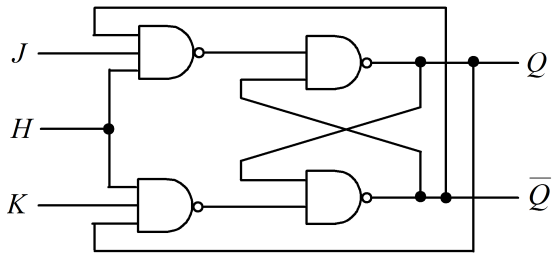
3.3.3 Bascule JK synchrone (JKT ou JKH)

De même on peut avoir une bascule JK qui fonctionne avec un signal d'horloge.

Pour son fonctionnement :

$$\begin{cases} \text{Si H=0} \quad \forall J \text{ et } \forall K \Rightarrow Q_{n+1} = Q_n \text{ mémorisation ;} \\ \text{Si H=1} \quad \Rightarrow \text{retournement au fonctionnement normal de la bascule JK.} \end{cases}$$

Le logigramme de JKH utilisant seulement les portes NAND est le même que celui de JK multiplié par l'entrée H, comme l'indique le schéma suivant :



La limitation de la JKT est quand $J = 1$ et $K = 1$: La sortie oscille en 0 et 1 pendant toute la durée de l'état haut du signal d'horloge, pour cela on utilise une bascule JK avec déclenchement sur front (montant ou descendant).

Une autre solution consiste à utiliser une bascule JK Maître-Esclave.

3.3.4 Bascule D à verrou (D latch)

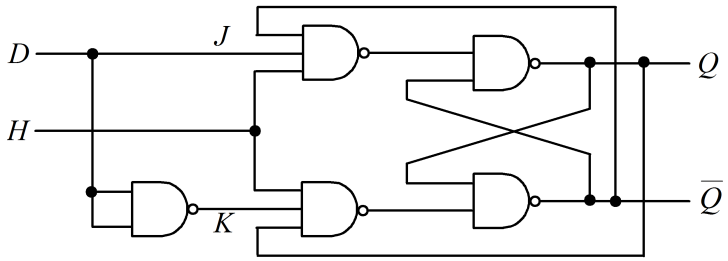
La bascule recopie l'entrée D en sortie Q quand l'horloge est active $H = 1$. Quand l'horloge est inactive $H = 0$, la bascule garde l'état précédent. Sur niveau 1 (haut) on peut

écrire :

H	Q_{n+1}
0	Q_n
1	D

$$\Rightarrow Q_{n+1} = \overline{H} \cdot Q_n + H \cdot D$$

Le schéma logique de la bascule 'D latch' à partir d'une bascule JKH (utilisant les portes NAND uniquement) est le suivant :

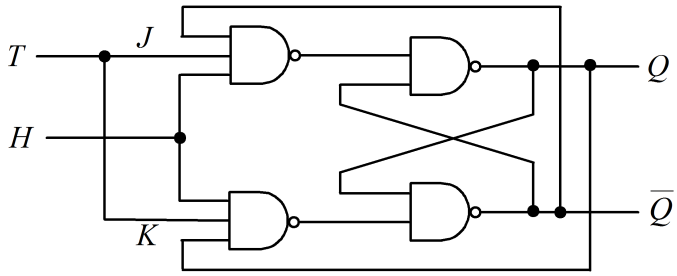


La bascule 'D latch' est très utilisée dans les compteurs synchrones.

3.3.5 Bascule T synchrone

En rajoutant un signal d'horloge à la bascule T, la sortie bascule (change d'état) dans le cas où l'horloge est active $H = 1$ ainsi que $T = 1$.

Le logigramme de la bascule T synchrone à partir d'une bascule JKH (avec des portes NAND uniquement) est :

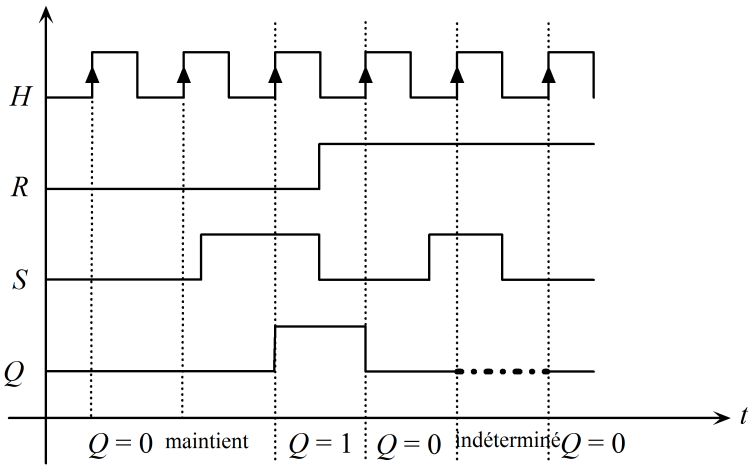


3.4 Synchronisation sur front et exemples de chronogrammes

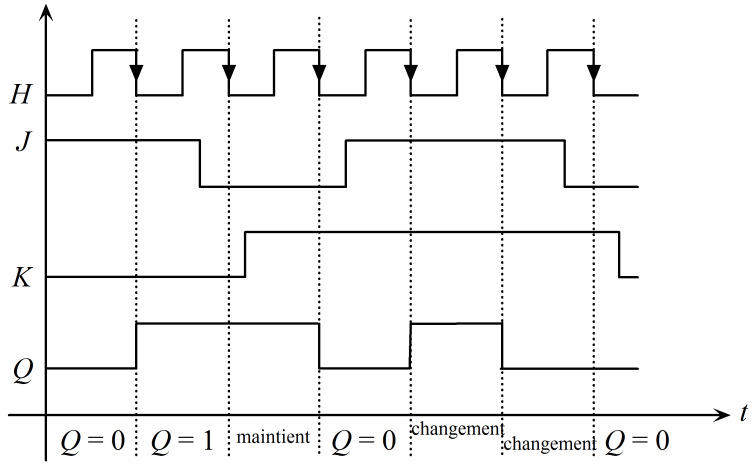
Les bascules synchrones sur front changent d'état uniquement sur un front du signal d'horloge ; en dehors de ces fronts, elle fonctionne en mémoire.

Ce mode de fonctionnement protège d'éventuels parasites sur les entrées car les entrées ne sont prises en compte que pendant la durée d'un front, qui est très courte.

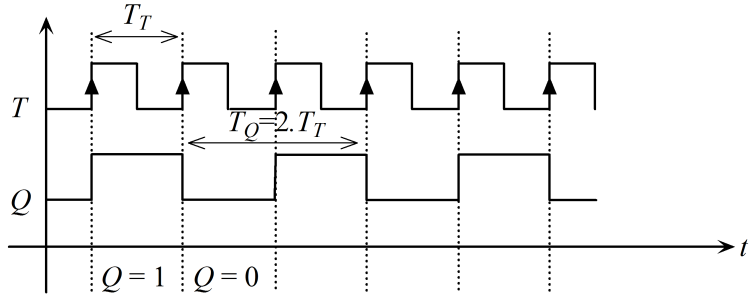
Exemple 3.1 Le chronogramme de fonctionnement d'une bascule RSH sur front montant, avec à l'état initial $Q = 0$ est illustré par la figure suivante :



Exemple 3.2 Le chronogramme suivant est un exemple de fonctionnement d'une bascule JKH sur front descendant, avec au départ $Q = 0$:



Exemple 3.3 Le chronogramme de fonctionnement d'une bascule T sur front montant, avec initialement $Q = 0$:

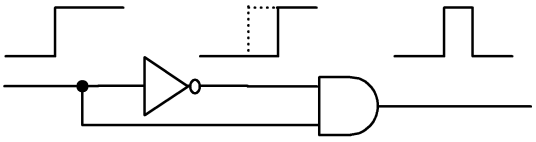


On remarque que la fréquence du signal en sortie (Q) est divisée par deux par rapport

à celle de la bascule T ($f_Q = f_T/2$), puisque la période de Q représente le double de la période de la bascule T ($T_Q = 2 \cdot f_T$).

Détection des fronts

Le circuit suivant permet de générer une impulsion de courte durée, par exploitation du temps de propagation des portes logiques :



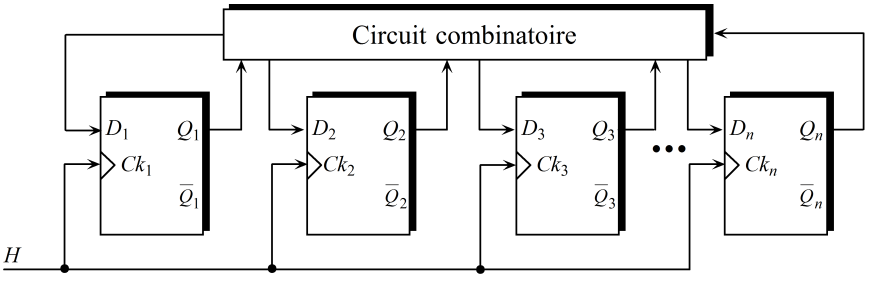
3.5 Compteurs

3.5.1 Définition

Un compteur est un circuit séquentiel constitué d'un circuit combinatoire et d'une succession de n bascules décrivant au rythme d'une horloge un cycle de comptage d'un maximum de 2^n combinaisons.

3.5.2 Compteur synchrone

Un compteur est dit synchrone, si toutes les bascules sont déclenchées en même temps par le même signal d'horloge. La figure suivante illustre le schéma d'un compteur synchrone utilisant des bascules D à front montant.



Exemples de réalisation

Exemple 3.4 Compteur modulo 8 à cycle complet, en utilisant des bascules JK :

Soit un compteur synchrone progressif (progressif : passe de la valeur m à $m+1$ dans le sens croissant) de trois bascules de types JK ($8 = 2^3 \Rightarrow 3$ bascules) à front descendant.

En se basant sur la table d'excitation de la bascule JK suivante :

Q	Q^+	J	K
0	0	0	$\emptyset$
0	1	1	$\emptyset$
1	0	$\emptyset$	0
1	1	$\emptyset$	0

Où : Q et Q^+ représentent l'état présent et l'état futur respectivement.

On obtient la table d'implication du compteur tel que :

Q_C	Q_B	Q_A	Q_C^+	Q_B^+	Q_A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	$\emptyset$	0	$\emptyset$	1	$\emptyset$
0	0	1	0	1	0	0	$\emptyset$	1	$\emptyset$	$\emptyset$	1
0	1	0	0	1	1	0	$\emptyset$	$\emptyset$	0	1	$\emptyset$
0	1	1	1	0	0	1	$\emptyset$	$\emptyset$	1	$\emptyset$	1
1	0	0	1	0	1	$\emptyset$	0	0	$\emptyset$	1	$\emptyset$
1	0	1	1	1	0	$\emptyset$	0	1	$\emptyset$	$\emptyset$	1
1	1	0	1	1	1	$\emptyset$	0	$\emptyset$	0	1	$\emptyset$
1	1	1	0	0	0	$\emptyset$	1	$\emptyset$	1	$\emptyset$	1

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	0	0	1	0
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

$J_C = Q_BQ_A$

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	0	0	1	0

$K_C = Q_BQ_A$

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	0	1	$\emptyset$	$\emptyset$
1	0	1	$\emptyset$	$\emptyset$

$J_B = Q_A$

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	$\emptyset$	$\emptyset$	1	0
1	$\emptyset$	$\emptyset$	1	0

$K_B = Q_A$

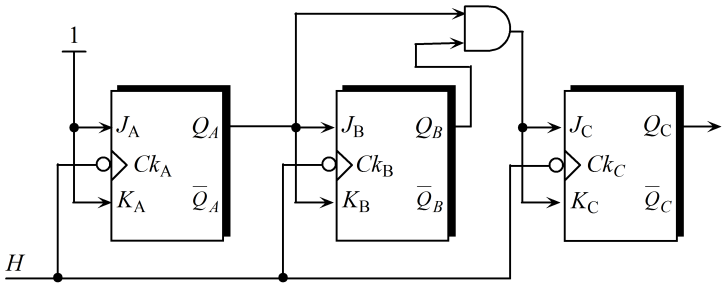
Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	1	$\emptyset$	$\emptyset$	1
1	1	$\emptyset$	$\emptyset$	1

$J_A = 1$

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	$\emptyset$	1	1	$\emptyset$
1	$\emptyset$	1	1	$\emptyset$

$K_A = 1$

Le logigramme correspondant au compteur à front descendant est donné par la figure.



Exemple 3.5 Compteur modulo 5 (à cycle incomplet) en utilisant des bascules T :

Le compteur modulo 5 compte de 0 à 4. Sachant que $2^2 < 5 < 2^3 \Rightarrow 3$ bascules. La table d'excitation de la bascule T est :

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

En se basant sur cette dernière, on complète la table d'implication du compteur synchrone progressif et régulier à cycle incomplet modulo 5.

Q_C	Q_B	Q_A	Q_C^+	Q_B^+	Q_A^+	T_C	T_B	T_A
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	∅	∅	∅	∅	∅	∅
1	1	1	∅	∅	∅	∅	∅	∅

Les équations logiques sont :

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	∅	∅	∅

$T_C = Q_BQ_A$

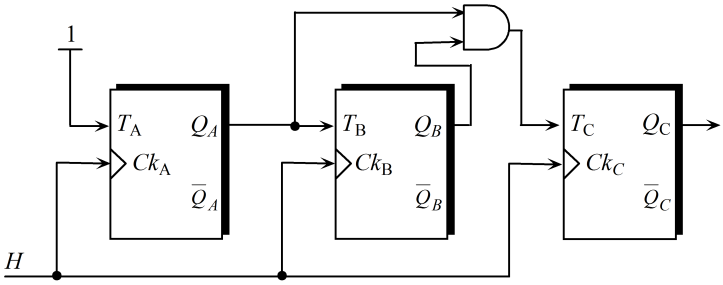
Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	0	1	1	0
1	0	∅	∅	∅

$T_B = Q_A$

Le logigramme correspondant au compteur synchrone modulo 5 à front montant est :

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	1	1	1	1
1	1	∅	∅	∅

$T_A = 1$



3.5.3 Compteur asynchrone

Le compteur est dit asynchrone, si le signal d'horloge est appliqué seulement à la première bascule, et l'état de chaque bascule est fonction des états des bascules précédentes.

Exemples de réalisation

Exemple 3.6 Compteur asynchrone progressif et régulier modulo 8 (de 0 à 7) :

$8 = 2^3 \Rightarrow 3$ bascules. Soit 3 bascules JK à front descendant. La table de vérité est :

$N°$	Q_C	Q_B	Q_A	Q_C^+	Q_B^+	Q_A^+	$N°+$
0 ⇒	0	0	0	0	0	1	1
1	0	0	1	0	1	0	⇒ 2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4 ⇒	1	0	0	1	0	1	⇒ 5
5	1	0	1	1	1	0	6
6 ⇒	1	1	0	1	1	1	7
7	1	1	1	0	0	0	⇒ 0

Le chronogramme de fonctionnement du compteur asynchrone progressif et régulier modulo 8 (à cycle complet) est le suivant :

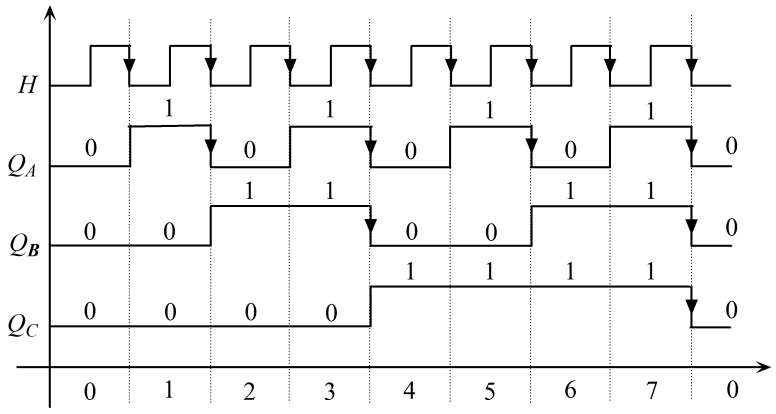
Fonctionnement :

– H passe de 1 à 0 (front descendant) $\Rightarrow Q_A$ change d'état, $Q_A^+ = \overline{Q_A}$ (complémentation, basculement). Pour la bascule JK et d'après sa table d'excitation, on aura :

$$Q_A^+ = \overline{Q_A} \Rightarrow J_A = 1 \text{ et } K_A = 1$$

donc

$$\begin{cases} CK_A = H \\ J_A = K_A = 1 \end{cases}$$



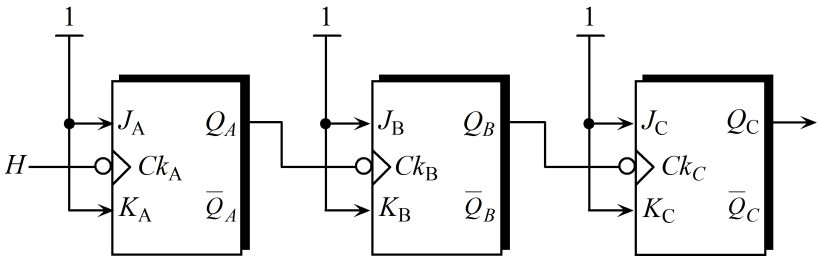
– Q_A passe de 1 à 0 $\Rightarrow Q_B^+ = \overline{Q_B}$. De même que pour Q_A, on obtient :

$$\begin{cases} CK_B = Q_A \\ J_B = K_B = 1 \end{cases}$$

– Q_B passe de 1 à 0 $\Rightarrow Q_C^+ = \overline{Q_C}$. Par suite, on trouve :

$$\begin{cases} CK_C = Q_B \\ J_C = K_C = 1 \end{cases}$$

Le schéma du compteur asynchrone régulier modulo 8 est le suivant :



Exemple 3.7 Compteur asynchrone progressif et régulier modulo 6 (de 0 à 5) :

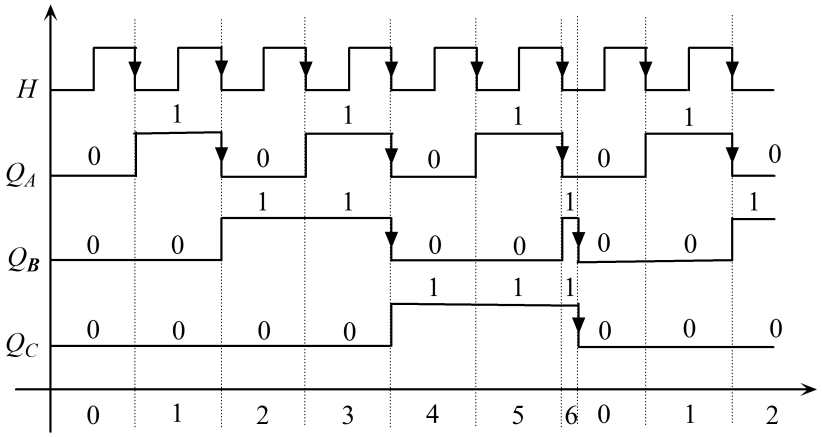
$2^2 < 6 < 2^3 \Rightarrow 3$ bascules. Le compteur modulo 5 réalise un compte de 0 à 5 (de 000 à 101 en binaire), arrivé à 5, le comptage doit être interrompu pour recommencer de 0. On doit, donc, remettre toutes les bascules à 0 après l'apparition de 5. Pour cela, on utilise les entrées asynchrone de remise à zéro (clr).

La table de vérité d'implication séquentielle est :

$N°$	Q_C	Q_B	Q_A		F
0	0	0	0		0
1	0	0	1		0
2	0	1	0		0
3	0	1	1		0
4	1	0	0		0
5	1	0	1		0
6	1	1	0	$\rightarrow$ 0 0 0	1

$Q_CQ_BQ_A = 110$ est un état temporaire existant pendant une durée très courte (état indésirable).

Le chronogramme de fonctionnement du compteur asynchrone à front descendant, progressif et régulier modulo 6 (à cycle incomplet) est le suivant :

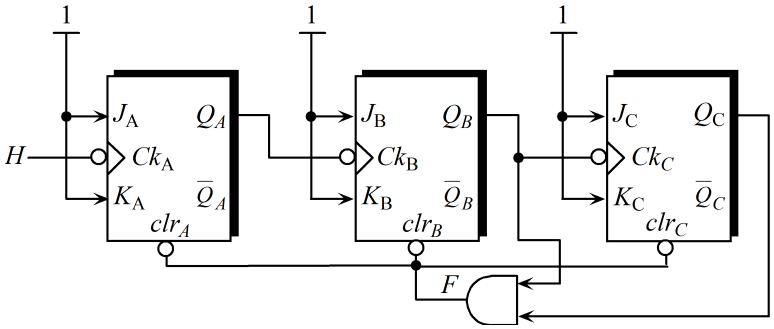


L'équation de l'entrée de remise à zéro (entrée de forçage à 0) est $F = Q_CQ_B$, tel que :

Q_C \ Q_BQ_A	0 0	0 1	1 1	1 0
0	0	0	0	0
1	0	0	∅	1

$F = Q_CQ_B$

Le schéma du compteur asynchrone régulier modulo 6, en considérant les bascule JK, est le suivant :



Exemple 3.8 Compteur asynchrone progressif, incomplet et irrégulier :

Le compteur réalise le compte suivant : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13.

- On force l'état 10 vers 11 ;
- On force l'état 12 vers 13 ;
- On force l'état 14 vers 0.

La table de vérité d'implication séquentielle correspondant au compteur à 4 bascules est :

$N°$	Q_D	Q_C	Q_B	Q_A	F_1	F_2
0	0	0	0	0	0	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	0	1	1	0	0
4	0	1	0	0	0	0
5	0	1	0	1	0	0
6	0	1	1	0	0	0
7	0	1	1	1	0	0
8	1	0	0	0	0	0
9	1	0	0	1	0	0
10	1	0	1	0	1	$\emptyset$
11	1	0	1	1	0	0
12	1	1	0	0	1	$\emptyset$
13	1	1	0	1	0	0
14	1	1	1	0	$\emptyset$	1

L'équation de l'entrée de forçage de 10 à 11 et de 12 à 13 est $F_1 = Q_D Q_B \overline{Q_A} + Q_D Q_C \overline{Q_A}$ obtenu à partir du tableau de Karnaugh suivant :

$Q_D Q_C$ \ $Q_B Q_A$	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	0	0	0	0
11	1	0	$\emptyset$	$\emptyset$
1 0	0	0	0	1

$Q_D Q_C \overline{Q_A}$ $Q_D Q_B \overline{Q_A}$

L'équation logique de forçage de 14 à 0 est $F_2 = Q_D Q_C \overline{Q_A}$, tel que :

$Q_D Q_C$ \ $Q_B Q_A$	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	0	0	0	0
11	$\emptyset$	0	$\emptyset$	1
1 0	0	0	0	$\emptyset$

$Q_D Q_B \overline{Q_A}$

3.6 Registres

3.6.1 Définition

Un registre est un circuit séquentiel synchrone, constitué de n bascules permettant de stocker temporairement un mot (une information) binaire de n bits dans l'objectif de son transfert dans un autre circuit (pour affichage, mémorisation, traitement, etc.).

3.6.2 Registre à décalage

Il s'agit d'un circuit permettant de décaler le mot (l'information) binaire bit par bit, soit à gauche, soit à droite.

Registre à décalage à gauche

Son objectif est de reproduire à la sortie de la bascule de rang i l'état logique de la sortie de la bascule de rang $i+1$, tout en appliquant le même signal d'horloge à toutes les bascules.

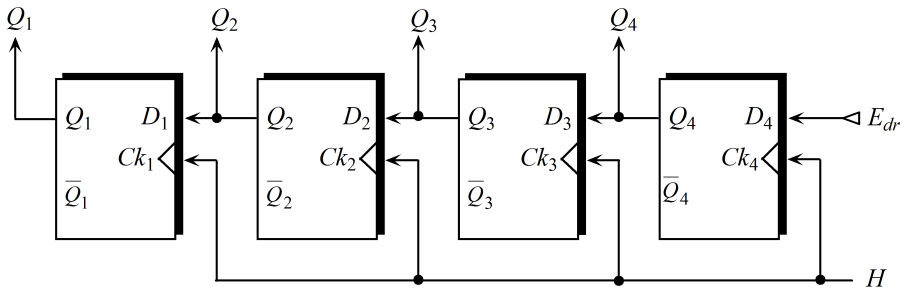
Il s'agit d'un registre possédant une seule entrée à droite E_{dr} et n sorties (Q_1, Q_2,$\cdots$, Q_n).

Exemple 3.9 Registre à décalage à gauche à 4 bascules D :

Selon la caractéristique de décalage à gauche $Q_i^+ = Q_{i+1}$ et de la caractéristique de la bascule D $Q_i^+ = D_i$, on déduit que $D_i = Q_{i+1}$, donc les équations logiques correspondant sont :

$$\begin{cases} D_1 = Q_2 \\ D_2 = Q_3 \\ D_3 = Q_4 \\ D_4 = E_{dr} \end{cases}$$

Le schéma correspondant au registre à décalage à gauche à 4 bascules D à front montant est donné par la figure suivante :



Registre à décalage à droite

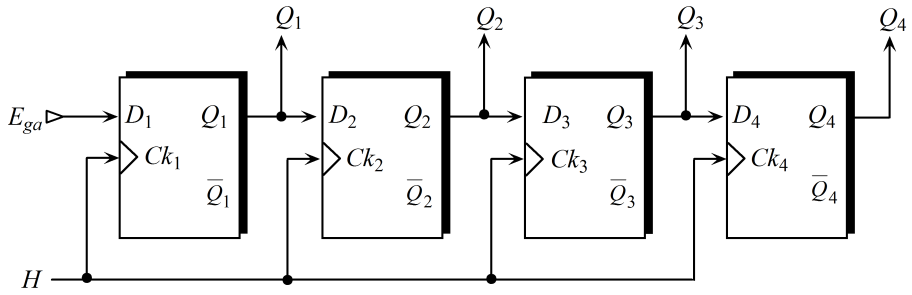
De même que pour le registre précédent, celui ci reproduit à la sortie de la bascule de rang $i+1$ l'état logique de la sortie de la bascule de rang i. C'est un registre possédant une entrée à gauche E_{ga} et n sorties $(Q_1, Q_2, \cdots, Q_n)$.

Exemple 3.10 Registre à décalage à droite formé de 4 bascules de type D :

Selon la caractéristique de décalage à droite $Q_i = Q_{i+1}^+$ et sachant que la caractéristique de la bascule D $Q_{i+1}^+ = D_{i+1}$, on déduit que $D_{i+1} = Q_i$, alors :

$$\begin{cases} D_4 = Q_3 \\ D_3 = Q_2 \\ D_2 = Q_1 \\ D_1 = E_{ga} \end{cases}$$

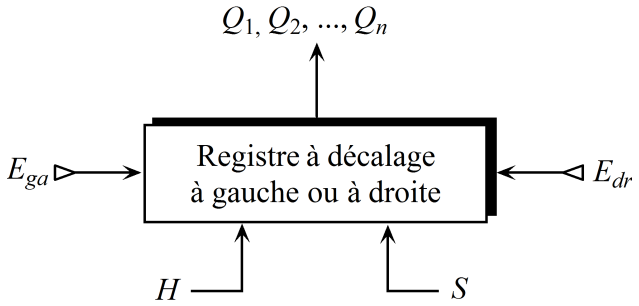
Le schéma du registre à décalage à droite à 4 bascules D à front montant est représenté par la figure suivante :



Registre à décalage à gauche ou à droite

Il s'agit de la composition des deux premiers registres, en rajoutant en entrée de sélection S offrant la possibilité de choisir le sens de décalage.

Le schéma représentatif de ce type de registre est illustré par la figure suivante :



Exemple 3.11 Registre à décalage à gauche ou à droite à 4 bascules de type D :

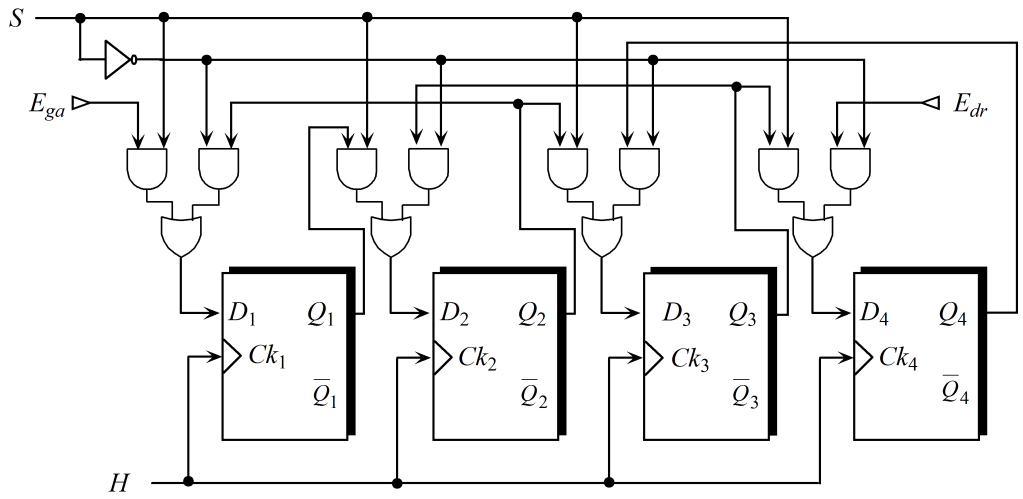
Ce genre de registre est caractérisé par l'entrée de sélection logique S, tel que :

1. $S = 0 \Rightarrow$ décalage à gauche ;
2. $S = 1 \Rightarrow$ décalage à droite.

D'après les équations logiques déjà établies pour les registres à décalages à gauche et à droite, respectivement, à 4 bascules D, on trouve :

$$\begin{cases} D_1 = \overline{S}Q_2 + SE_{ga} \\ D_2 = \overline{S}Q_3 + SQ_1 \\ D_3 = \overline{S}Q_4 + SQ_2 \\ D_4 = \overline{S}E_{dr} + SQ_3 \end{cases}$$

Le schéma logique correspondant au registre à gauche ou à droite formé de 4 bascules D à front montant est le suivant :



Bibliographie

[Ale04] C. Alexandre, *Circuits numériques*, Polycopié de cours électronique, Conservatoire national des arts et métiers, France, 2004.

[Bel10] Mc. Belaid et collectif, *Logique combinatoire et séquentielle*, Presses de Mitidja, Baraki, Alger, Algérie, 2010.

[Gin87] M. Gindre, *Electronique numérique*, Mc Graw Hill, Paris, France, 1987.

[Kle06] W. Kleitz, *Digital electronics with VHDL*, Pearson Education Hall, New Jersey, USA, 2006.

[Lil86] H. Lilen, *Cours pratique de logique pour microprocesseurs*, Ed Radio, Paris, France, 1986.

[Mer05] J. J. Mercier, *Bit par bit : numération, binaire, logique combinatoire*, Ellipses Editions Marketing, Paris, France, 2005.

[Mer06] J. J. Mercier, *Computers 2, séquence après séquence, logique séquentielle*, Ellipses Editions Marketing, Paris, France, 2006.

[Mok10] M. K. Mokhtari, I. Caid *De l'algèbre de boole aux circuits numériques, cours et applications*, Office des Publications Universitaires, Alger, Algérie, 2010.

[Sba13] M. Sbaï, *Electronique numérique, logique combinatoire et composants numérique*, Ellipses Editions Marketing, Paris, France, 2013.

[Tac88] A. Tachet, *Automatisme, Tome 1, logique combinatoire*, Office des Publications Universitaires, Alger, Algérie, 1988.