



UNIVERSITÉ MOULAY ISMAÏL
FACULTÉ DES SCIENCES ET TECHNIQUES

ERRACHIDIA

SUPPORT DE COURS
Travaux Dirigés et Pratiques

Niveau : Licence Sciences de l'Ingénieur
Énergies Renouvelables

ÉLECTRONIQUE NUMÉRIQUE

Préparé par

M. ABDELKADER ELHANAOU
DOCTEUR DE L'UNIVERSITÉ IBN ZOHR
PROFESSEUR DE L'UNIVERSITÉ MOULAY ISMAÏL

Année Universitaire : 2020 / 2021

REMERCIEMENTS	iii
TABLE DES MATIÈRES	v
INTRODUCTION	1
I Cours	3
1 Introduction à l'électronique numérique	4
1.1 Généralités	4
1.1.1 Signal numérique VS signal analogique	4
1.1.2 Unité de base de l'information numérique	5
1.1.3 Conventions logiques	5
1.2 Encodage de l'information	6
1.2.1 Système binaire (BIN)	6
1.2.2 Borne supérieure représentable	7
1.2.3 Conversion BIN - DEC	7
1.2.4 Système hexadécimal (HEX)	8
1.3 Autres codes	9
1.3.1 Code BCD (Binary Coded Decimal)	9
1.3.2 Représentation en binaire des nombres relatifs	10
1.3.3 Flottants	12
1.4 Codes à chercher	14
2 Opérateurs combinatoires	15

2.1	Introduction	15
2.2	Opérateurs combinatoires fondamentaux	15
2.2.1	Opérateurs génériques	16
2.3	Propriétés fondamentales de l'algèbre de Boole	17
2.3.1	Conventions	17
2.3.2	Propriétés	17
2.4	Autres opérateurs logiques	18
2.4.1	Opérateurs logiques complémentaires : NON-ET, NON-OU	18
2.4.2	Opérateur logique : OU exclusif	19
2.4.3	Opérateur complémentaire du OU exclusif : XNOR	20
3	Circuits logiques combinatoires	24
3.1	Introduction	24
3.2	Outils théoriques pour les CLC	24
3.2.1	Première forme canonique	25
3.2.2	Seconde forme canonique	26
3.3	Minimisation logique	27
3.3.1	Minimisation algébrique	28
3.3.2	Minimisation utilisant le tableau de Karnaugh	28
3.4	Circuits usuels non arithmétiques	30
3.4.1	Multiplexeur	30
3.4.2	Multiplexeurs et génération de fonctions combinatoires	32
3.4.3	Décodeur	34
4	Circuits séquentiels	38
4.1	Introduction	38
4.1.1	Séquentialité?	38
4.1.2	Exemple : : détection d'une séquence	39
4.2	Éléments de mémoire	39
4.2.1	Cellules de base	39
4.2.2	Circuit bistable SR	40
4.2.3	Bistable SR avec signal d'activation	40
4.2.4	Bistable D	42
4.2.5	Bascule D de type maître esclave	42

TABLE DES MATIÈRES

4.2.6	Entrées asynchrones SET et RESET	47
4.2.7	Autres types de bascules : Bascule T	48
4.2.8	Autres : Bascule JK	48
4.3	Fonctions séquentielles STANDARD	49
4.3.1	Compteurs ou décompteurs	49
4.3.2	Registres	53
CONCLUSION ET PERSPECTIVES		ii
ANNEXE		ii

Depuis le début des années 2k, la technologie numérique est omniprésente dans la majeure partie des aspects de la société humaine. Une multitude de systèmes tels-que ordinateurs, téléphones, caméras, appareils électroménagers et médicaux, robots, est concernée par cette thématique. Les mini-drones et thermomètres à infrarouge ont été abondamment utilisés au Maroc pendant le confinement humain 2020. Ce support de Cours, Travaux Dirigés, et Pratiques d'électronique numérique, s'adresse aux étudiants de Licence des filières Sciences de l'Ingénieur, et Énergies Renouvelables. Il vise au travers de ses documents, programmes, et échanges pendant la durée de l'enseignement en présence, ou à proximité, à transmettre aux étudiants le savoir et savoir-faire des systèmes numériques, et les logiciels associés. En ce sens, le manuscrit fournit une couverture complète et ascendante des circuits digitaux.

En première partie (Cours), le manuel commence par une situation du cadre précis de l'électronique numérique, y compris la mention relative à l'aspect physique des systèmes digitaux linéaires. Nous portons dans ce chapitre d'initiation une attention particulière aux systèmes de numération nécessaires au codage de l'information.

Le deuxième chapitre est ensuite consacré aux opérateurs élémentaires, dits combinatoires. Ces derniers matérialisent en effet, des opérateurs bien connus de l'algèbre de **BOOLE**(Mathématicien anglais : 1815 – 1864). Pour permettre aux étudiants d'apprendre par l'exemple, l'additionneur *nbits* est traité à la fin de ce chapitre.

Le troisième chapitre rassemble un bagage de connaissances, et outils théoriques relatives aux systèmes logiques combinatoires (SLC). Nous aurons ici l'occasion d'évoquer les circuits usuels non-arithmétiques.

Enfin, dans **Le quatrième chapitre** sont étudiés les circuits logiques séquentiels. Ce chapitre couvre les éléments de mémoire ou Bascules, et la logique séquentielle qui inclue les compteurs, les registres, et les machines à états finis.

En seconde partie (Travaux dirigés), le manuscrit traite quelques séries d'exercices, avec corrigés succincts.

En dernière partie (Travaux pratiques), le livre débute par un premier TP d'initialisation au logiciel Quartus d'Altera. Ensuite s'enchainent d'autres TP relatifs à des applications illustrées au cours. Nous citons par là l'additionneur– soustracteur *nbits*, le détecteur de débordement de calculs (Over Flow), les circuits usuels non arithmétiques (Multiplexeurs, Décodeurs), les éléments de mémoires (bascules), les circuits logiques séquentiels tels-que les compteurs, les registres, ...

Première partie

Cours

Ce premier chapitre retrace les différentes notions de base du numérique. On y précise les aspects représentatifs de l'information, les conventions logiques, et définit les systèmes de numération les plus utilisés.

1.1 Généralités

1.1.1 Signal numérique VS signal analogique

En communication et traitement de l'information, deux techniques très générales sont distinguées : **analogiques** (ressemblance) et **numériques** (nombre).

Un signal analogique est défini comme une quantité continue (tension électrique p.ex.), qui correspond directement à l'information qu'il représente.

Pex. Un capteur de pression barométrique qui émet une tension électrique correspondant à la pression mesurée.

Un signal numérique est une fonction discontinue du temps. Il ne peut prendre qu'un nombre fini (généralement 2) de valeurs conventionnelles, sans rapport avec le contenu de l'information.

Pex. Un disque compact : une sorte de gravures en pointillé ou trous ovales, presque identiques, répartis de façon irrégulière sur des pistes quasi-circulaires.

Ces deux types de signaux sont présentés à la Figure 1.1.

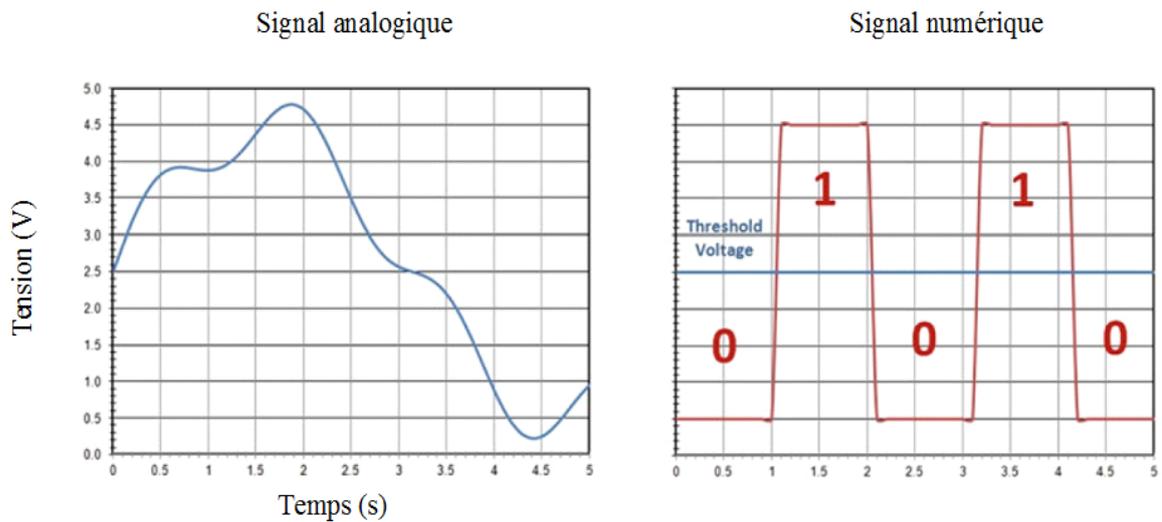


Figure. 1.1

1.1.2 Unité de base de l'information numérique

L'information numérique peut être représentée par un ensemble de quanta (quantité la plus petite) d'information. L'unité de base associée étant le bit (contraction de **B**inary **D**igit) ; c'est une variable binaire qui peut prendre 2 valeurs : 0 ou 1.

En électronique, la tension est une grandeur significative ; la d.d.p. U entre 2 points quelconques d'un circuit électrique, est exprimée par l'équation :

$$U = V_{point} - V_{reference} \quad (1.1)$$

Donc, un bit (0 ou 1) est spécifié par chaque équipotentielle.

1.1.3 Conventions logiques

Considérons le système numérique de la figure 1.2. e_i, s_j : 2 valeurs notées : **H** (High) et **L**

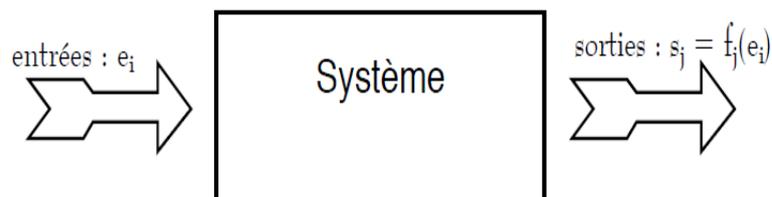


Figure. 1.2

(Low). Un tel circuit effectue comme missions générales : **les opérations d'Addition, Multiplica-**

tion, et Mémorisation de données. Dans ce cadre, les entrées et les sorties sont numériques.

La valeur d'un signal représente en général un chiffre en base 2 (0 ou 1), un état d'un opérateur (Actif ou non), ...

Une association entre variable électrique (H ou L), et le sens que l'on donne à cette valeur (p.ex. 0 ou 1), constitue une convention logique.

Une convention logique peut être **positive**, comme dans le cas du tableau 1.1 ; elle est dite **né-gative** dans le cas contraire.

Tension	Équivalent numérique	Variable logique	Ex. 1 : Lampe	Ex. 2 : Alarme
Niveau Haut	1	Vrai	Allumée	Activée
Niveau bas	0	Faux	Éteinte	Désactivée

Tableau. 1.1

1.2 Encodage de l'information

Coder une information consiste à lui associer un symbole ou une combinaison de symboles en vue de la conserver, la traiter, ou la transmettre.

En effet, les bits sont regroupés en paquets par 4, et le plus souvent par 8 (octet), ou des multiples de 8 (mots), respectant ainsi certaines règles de construction, ou codes. L'octet présenté à la figure 1.3, constitue l'unité pratique de base la plus commode, et la plus riche.

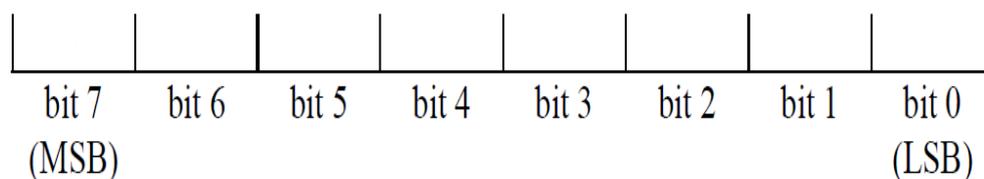


Figure. 1.3

1.2.1 Système binaire (BIN)

Comme en système décimal (DEC), il est possible d'exprimer un nombre N dans le système binaire sous la forme générale :

$$N = a_{m-1}R^{m-1} + a_{m-2}R^{m-2} + \dots + a_1R^1 + a_0R^0 \quad (1.2)$$

Où R est la base du système ; m le nombre de digits dont se compose la représentation ; les exposants $k = 0, \dots, m - 1$ représentent la position de chaque digit en commençant de la droite ; les symboles de l'alphabet qui représente les nombres sont contenus dans : $A = \{0, 1, \dots, R - 1\}$; les coefficients a_i sont des nombres qui correspondent aux valeurs des symboles de A .

- ◇ 1^{er} exemple : Système décimal (DEC) : Alphabet des symboles : $A = \{0, 1, \dots, 9\}$; $R = 10$; un nombre p.ex. $N = 2020$ est écrit dans le système DEC comme :

$$2020_{10} = 2 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 0 \cdot 10^0$$

- ◇ 2nd exemple : Système binaire, ou base deux (BIN) : Alphabet des symboles : $A = \{0, 1\}$; $R = 2$; un nombre p.ex. $N = 14$ écrit dans le système BIN signifie que :

$$14_{10} = 8 + 4 + 2 + 0 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 1110_2$$

1.2.2 Borne supérieure représentable

Soit N un nombre naturel de m bits. Il fournit au maximum 2^m combinaisons différentes ; autrement dit, les valeurs permises de N sont telles-que :

$$0 \leq N \leq N_{max} = 2^m - 1$$

Les valeurs couramment rencontrées pour m sont : 8 (octet), 16 (Entier court), et 32 (Entier long), de bornes supérieures respectives : 255 ; 65535 ; et 4294967295.

Re : Les m bits représentant la valeur maximale du nombre naturel sont tous égaux à un.

1.2.3 Conversion BIN - DEC

Dans le sens direct, la version décimale s'obtient directement en appliquant l'équation 1.2. En voici quelques exemples :

$$0101_2 = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5_{10}$$

$$1111_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 15_{10}$$

$$10001110_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 142_{10}$$

$$11111111_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 255_{10}$$

Dans le sens inverse, un nombre entier N avec $N \geq 2$ peut s'écrire sous la forme :

$$N = 2 \cdot q_0 + r_0 ; r_0 = 0, 1 \quad (1.3)$$

Où q_0 est le quotient ; r_0 le reste de la division de N par 2.

La divisions se répète si $q_0 \geq 2$:

$$q_0 = 2 \cdot q_1 + r_1 ; r_1 = 0, 1 \quad (1.4)$$

En substituant cette dernière équation dans la relation 1.3, on obtient :

$$N = 2 \cdot (2 \cdot q_1 + r_1) + r_0 \quad (1.5)$$

En poursuivant le processus si $q_1 \geq 2$, en écrivant :

$$N = 2 \cdot (2 \cdot (2 \cdot q_2 + r_2) + r_1) + r_0 = 2^3 \cdot q_2 + 2^2 \cdot r_2 + 2^1 \cdot r_1 + 2^0 \cdot r_0 \quad (1.6)$$

Enfin, nous obtenons par généralisation :

$$N = 2^k \cdot q_{k-1} + 2^{k-1} \cdot r_{k-1} + \dots + 2^1 \cdot r_1 + 2^0 \cdot r_0 \quad (1.7)$$

Les reports r_i à l'issue des divisions successives par deux représentent les bits du nombre N .

$$N_{10} = (q_{k-1}r_{k-1} \dots r_1r_0)_2$$

Pex. : Le tableau 1.2 montre que $46_{10} = 101110_2$.

résultats de $\div 2 : q_i$	Nombre	Reports	r_i
	$N_{10} = 46$	0	r_0
q_0	23	1	r_1
q_1	11	1	r_2
q_2	5	1	r_3
q_3	2	0	r_4
q_4	1	1	q_4

Tableau. 1.2

1.2.4 Système hexadécimal (HEX)

Quand le nombre de digits m est multiple de 4 (p.ex. 8 ; 16 ; 32 ; ...), il est souvent pratique d'écrire le nombre entier N en hexadécimal ($R = 16$). L'alphabet de ce système étant

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

Tableau. 1.3

$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$. La table de conversion 1.3 montre le passage entre systèmes DEC–BIN–HEX.

Étant donné que la base 16 est une puissance de 2, la correspondance entre un nombre écrit en BIN, et sa version HEX est simple ; la conversion BIN–HEX est immédiate.

Ex. :

$$10001111_2 = 1000_2 \ 1111_2 = 8F_{16} = 8F_H$$

$$11100_2 = 0001_2 \ 1100_2 = 1C_{16} = 1C_H$$

Dans ce cas, la conversion peut être également faite en menant des divisions successives par 16, comme dans l'exemple suivant :

$$655 = 28F_H$$

1.3 Autres codes

1.3.1 Code BCD (Binary Coded Decimal)

Tout le monde a pris l'habitude de compter en base 10, qui n'est pas une puissance de 2. Il n'y a donc pas de correspondance simple entre les systèmes BIN et DEC. Cette difficulté est parfois gênante en pratique. C'est pour cette raison que l'on rencontre parfois des **codes hybrides** :

le nombre est écrit en chiffres décimaux, et chaque chiffre est codé en binaire sur 4 bits. Le code le plus classique, appelé BCD 8421, et qui consiste à coder chaque chiffre décimal (0 à 9) en binaire naturel (0000 à 1001). Beaucoup de calculettes utilisent ce code pour faciliter les opérations d'affichage.

1.3.2 Représentation en binaire des nombres relatifs

1.3.2.1 Code Signe Valeur absolue

Habituellement, représenter un nombre signé N de m bits peut être selon un code qui sépare le signe et la valeur absolue : « Signe Valeur absolue » (1 bit pour le signe : le bit le plus à gauche ou MSB, et $m - 1$ bits pour la valeur absolue).

Considérons l'exemple d'un paquet de 4 bits, présenté à la figure 1.4. Les signes + et - sont

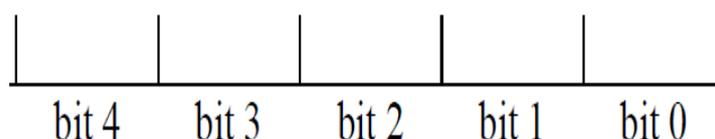


Figure. 1.4

codés respectivement avec 0 et 1, comme dans l'exemple de la table 1.4.

DEC	BIN	
4	0100	
3	0011	
2	0010	
1	0001	
0	0000	
-0	1000	!
-1	1001	
-2	1010	
-3	1011	
-4	1100	

Tableau. 1.4

Ce type de code n'est en fait jamais utilisé pour les entiers. En effet, l'arithmétique sous-jacente est compliquée (consultez TD). Dans ce code, comme dans d'autres, l'intervalle de définition d'un nombre relatif N , codé sur m bits est : $-2^{m-1} \leq N \leq 2^{m-1} - 1$.

1.3.2.2 Code complément à deux (CA2)

La construction du code CA2, sur m bits, se décline directement de la définition **MODULO** 2^m des nombres (Il s'agit d'une restriction à un sous-ensemble fini des opérations sur les entiers).

- ◇ Soit N entier ; Si $N \geq 0$ le code de N est son écriture en binaire naturel, éventuellement complété à gauche par des 0 ; p.ex. $N = +23$ est écrit dans le système BIN sur 8 bits comme :00010111
- ◇ Si $N \leq 0$ le code de N est l'écriture en binaire naturel de $2^m + N$, c'est -à-dire $2^m - |N|$; p.ex. $N = -23$ est écrit dans le système BIN sur 8 bits comme code BIN de 233 : 11101001.

Rem :

- ◇ Le bit de fort poids représente bien le bit de signe du nombre considéré ;
- ◇ Le calcul du code de l'opposé d'un entier quelconque est une conséquence de la définition de la représentation : $-N = 2^m - N$, Modulo 2^m ;
- ◇ **Astuce de calcul** : Comment obtenir rapidement l'expression binaire de l'opposé d'un nombre N dont on connaît le code ?

$$2^m - N = 2^m - 1 - N + 1$$

$$C_2(N) = C_1(N) + 1 = \overline{N} + 1$$

Le nouveau code $C_1(N)$ s'appelle **complément à UN** ou complément restreint de N , noté \overline{N} ; il est obtenu en remplaçant dans le code de N les 1 par des 0, et réciproquement.

Pex. : $23 = 00010111$; $-23 = \overline{23} + 1 = 11101000 + 1 = 11101001$ (Résultat précédent retrouvé !)

- ◇ Le code CA2 est le code utilisé pour représenter les nombres dans un ordinateur. Cependant, il complique les opérations de comparaisons et, plus généralement, celles qui font intervenir une relation d'ordre entre les nombres. On montre sur la figure 1.5 la relation qui lie les codes binaire naturel, et CA2 sur 8 bits.
- ◇ Un autre code qui permet de remédier à l'inconvénient du CA2 est le code binaire décalé. La formule qui génère le code binaire décalé sur m bits, d'un nombre N , tel-que $-2^{m-1} \leq N \leq 2^{m-1} - 1$, n'est autre que le code BIN de $N + 2^{m-1}$. On peut noter que le nombre 2^{m-1} a son MSB égal à 1, tous les autres chiffres étant nuls. On passe du code binaire décalé au code CA2 en inversant le bit de signe. La figure 1.6 présente le lien entre les codes BIN, et binaire décalé sur 8 bits. On remarque également que le code binaire décalé possède la

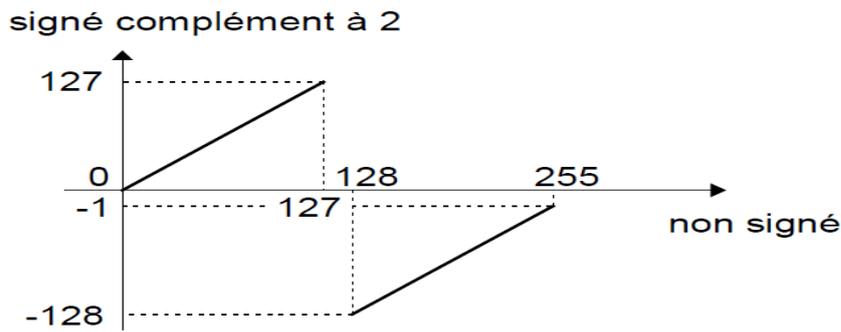


Figure. 1.5

même relation d'ordre que le code binaire naturel des nombres non signés. On le rencontre dans la représentation de l'exposant des nombres flottants (Voir ci-après).

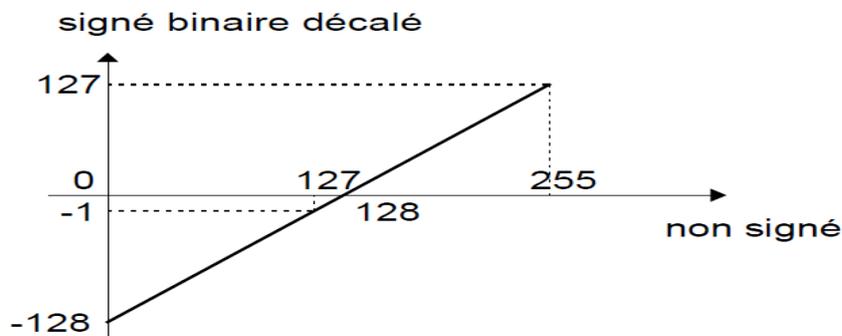


Figure. 1.6

1.3.2.3 Extension de signe

C'est l'opération qui consiste en l'augmentation de la taille ou la longueur du code (p.ex. passage de 8 à 16 bits). Elle se fait en complétant à gauche par le bit de signe.

Pex. : $-23 = 11111111\ 11101001$ (sur 16 bits) ; ce résultat es notablement différent de $00000000\ 11101001$, qui est le code de 233 dont l'existence est maintenant légale.

1.3.3 Flottants

1.3.3.1 Virgule flottante

Les nombres flottants permettent de représenter, de manière approchée, une partie des nombres réels. La valeur d'un réel ne peut être ni trop grande, ni trop précise. Cela dépend du nombre de bits utilisés (en général 32 ou 64 bits). C'est un domaine très complexe et il existe une norme, **IEEE-754**, pour que tout le monde obtienne les mêmes résultats. Le codage en binaire a la forme

présentée à la figure 1.7. C'est-à-dire que si X est un nombre flottant :

s : signe	e : exposant (signé)	zzz : partie fractionnaire de la mantisse (non signé)
-------------	------------------------	---

Figure. 1.7

$$X = (-1)^s \cdot 2^e \cdot 1,zzz \dots$$

où s est le signe de X , e est l'exposant entier signé, codé en binaire décalé et $zzz \dots$ est la partie fractionnaire de la valeur absolue de la mantisse. C'est, en binaire, la notation scientifique traditionnelle. Voici à la table 3.4, deux exemples de formats usuels.

nombre de bits	format binaire ($s + e + zzz \dots$)	valeur max	précision max
32	1 + 8 + 23	$2^{128} \approx 10^{38}$	$2^{-23} \approx 10^{-7}$
64	1 + 11 + 52	$2^{1024} \approx 10^{308}$	$2^{-52} \approx 10^{-15}$

Tableau. 1.5

Exa.1 : Nombre réel représenté en norme **IEEE**-754 simple précision, par : 0 10000000 10010001111010 \dots 0

- ◇ Le bit de signe est 0 : Le nombre est positif ;
- ◇ L'exposant est 10000000 ; il correspond à $128 = e + 127$, soit alors $e = 128 - 127 = 1$;
- ◇ La mantisse codée est 10010001111010 \dots 0 correspondant donc à : $1,10010001111010 \dots 0$
c'est-à dire : $1 + 2^{-1} + 2^{-4} + \dots \approx 3,14159 \approx \pi$
- ◇ $(3,14)_{10} = (11.001000111101)_2$

Exa.2 : Nombre réel $-118,625$ à représenter en norme **IEEE**-754 simple précision ?

- ◇ Le nombre est négatif, le bit de signe est 1 ; ;
- ◇ On convertit le nombre (sans le signe) en binaire,
on obtient : $1110110,101 = 1,110110101 \cdot 2^6$
- ◇ La mantisse est la partie après la virgule, remplie de 0 à droite pour obtenir 23 bits ; cela donne 11011010100000000000000
- ◇ L'exposant est égal à 6, et on doit le décaler puis le convertir en binaire : $6 + 127 = 133$ codé par 1000101 ;
- ◇ Au final $(-118,625)_{10}$ est codé par : 110001011101101010000000000000

Le calcul en virgule flottante est très coûteux en termes de circuit et de consommation (mais pas en fréquence de fonctionnement). Il est donc rarement utilisé en électronique numérique (comme en TNS d'ailleurs). On utilise plutôt le calcul en virgule fixe.

1.3.3.2 Virgule fixe

Le calcul avec des nombres en virgule fixe revient à faire tous les calculs avec des entiers en recadrant les résultats à l'aide d'une virgule fictive. Transposer un algorithme utilisant des flottants et le passer en virgule fixe est une tâche longue et délicate dans le cas général.

1.4 Codes à chercher

- ◇ Code GRAY : appelé aussi code binaire réfléchi ; une simple recherche vous montrera que ce code a la particularité que l'on passe d'une combinaison à la suivante en changeant la valeur **d'un seul chiffre binaire**.
- ◇ Codes détecteurs d'erreurs ;
- ◇ Code 1 parmi N , ou *ONE HOT* pour encoder les machines à états finis (FSM)
- ◇ Codes ALPHANUMÉRIQUES tels-que le code **ASCII** pour **American Standard Code for Information Interchange** ;
- ◇ ...

2.1 Introduction

Un système numérique aussi complexe qu'il soit, est construit de façon hiérarchique, comme un assemblage de boîtes noires, définies par leurs entrées et sorties. Tout en bas de cette hiérarchie, on trouve des opérateurs élémentaires, les briques ultimes au delà desquelles intervient l'électronicien qui les réalise avec des transistors. Certains de ces opérateurs élémentaires, dits combinatoires, sont la matérialisation, sous forme de circuits, ou de parties de circuits, des opérateurs (NON, ET, OU,...) bien connus de l'algèbre de **BOOLE** (Mathématicien anglais : 1815 – 1864). Les valeurs des sorties d'un tel opérateur, sont déterminées de façon univoque par les valeurs des entrées au même instant (à un temps de propagation près, bien sûr).

Sauf précision contraire, nous adopterons dans la suite une convention logique positive, qui associe le 0 binaire à la valeur logique **FAUX** et le 1 binaire à la valeur logique **VRAI**.

2.2 Opérateurs combinatoires fondamentaux

Pour chaque opérateur élémentaire, nous indiquerons le ou les symboles couramment rencontrés, puis nous en donnerons une description sous forme de table de vérité, fonction logique, équation ou expression algébrique montrant la fonction de chaque sortie en fonction des entrées,

...

2.2.1 Opérateurs génériques

Les opérateurs *NON* (*NOT*), *OU* (*OR*) et *ET* (*AND*) ont un rôle important dans la mesure où ils sont « génériques », autrement dit toute fonction combinatoire peut être exprimée à l'aide de ces opérateurs élémentaires.

2.2.1.1 Symboles

Bien que les symboles « rectangulaires » soient normalisés, la figure 2.1 présente les symboles curvilignes traditionnels, qu'on trouve dans la majorité des notices.

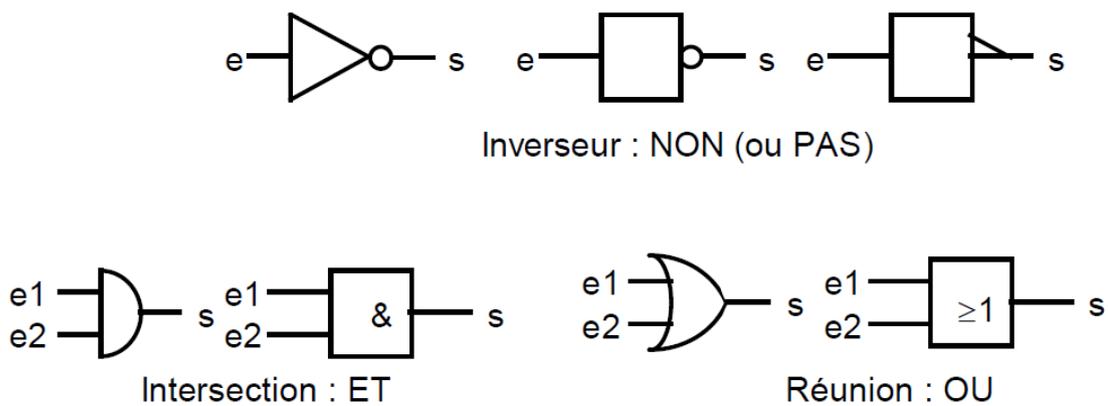


Figure. 2.1

L'assemblage de symboles élémentaires dans un « schéma » porte le nom de *logigramme* : schéma de câblage dans lequel on aurait oublié les masses et les alimentations des circuits.

2.2.1.2 Tables de vérité

Le fonctionnement d'un circuit logique peut être décrit par une table qui énumère les valeurs prises par la (ou les) sortie(s) en fonction des valeurs des variables d'entrée. Si le circuit a n entrées, alors on aura 2^n codes d'entrée possibles. Les tables peuvent être présentées sous forme linéaire ou, ce qui est souvent plus compact, sous forme de tableaux, conformément la figure 2.2.

2.2.1.3 Notations algébriques

- ◇ Opérateur *NON* : $s = \bar{e}$;
- ◇ Opérateur *OU* : $s = e_1 + e_2$;
- ◇ Opérateur *ET* : $s = e_1 \cdot e_2$.

e	s
0	1
1	0

NON

e1e2	s
00	0
01	0
10	0
11	1

	e2	0	1
e1	0	0	0
1	0	1	1

ET

e1e2	s
00	0
01	1
10	1
11	1

	e2	0	1
e1	0	0	1
1	1	1	1

OU

Figure. 2.2

2.3 Propriétés fondamentales de l'algèbre de Boole

L'algèbre de Boole fournit les moyens nécessaires pour calculer et interpréter l'information présentée sous forme binaire. Elle est définie par :

- ◇ Deux variables logiques, ou 2 états : $\{0, 1\}$;
- ◇ Opérations *OU* (somme logique), *ET* (multiplication logique), *NON* (négation) ;
- ◇ Opérateur *équivalence* : = (symétrique et transitif).

2.3.1 Conventions

- ◇ $X, Y, Z, X_1, X_2, \dots, X_n$ sont considérées des variables booléennes ;
- ◇ Les règles de priorités adoptées vont de la priorité la plus grande à la plus faible, c'est-à-dire de l'inversion au *OU*.

2.3.2 Propriétés

Les principales propriétés sont citées ci-après :

- ◇ **Loi de commutativité** : $X + Y = Y + X$, (Dual) $X \cdot Y = Y \cdot X$;
- ◇ **Loi d'associativité** : $(X + Y) + Z = X + (Y + Z) = X + Y + Z$, (Dual) $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z$;
- ◇ **Loi de double distributivité** : $(X + Y) \cdot Z = (X \cdot Z) + (Y \cdot Z)$, (Dual) $(X \cdot Y) + Z = (X + Z) \cdot (Y + Z)$;
- ◇ **Loi de De Morgan** : $\overline{X + Y} = \overline{X} \cdot \overline{Y}$, (Dual) $\overline{X \cdot Y} = \overline{X} + \overline{Y}$

Rem : Un opérateur, agissant sur 2 opérands, qui est associatif peut être généralisé à un nombre quelconque d'opérands. C'est le cas des opérateurs *ET* et *OU*, présentés à la figure 2.3.

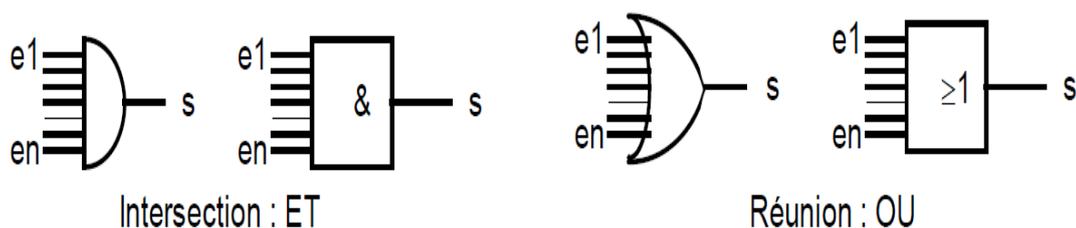


Figure. 2.3

2.4 Autres opérateurs logiques

2.4.1 Opérateurs logiques complémentaires : NON-ET, NON-OU

Les opérateurs *NON-ET* (NAND), et *NON-OU* (NOR), jouent un rôle particulier : ils contiennent chacun, éventuellement via les lois de De Morgan, les trois opérateurs génériques de la logique combinatoire *ET*, *OU* et *NON*.

2.4.1.1 Symboles et définitions

◇ **Non Et** : la figure 2.4 présente le symbole qui lui est associé.

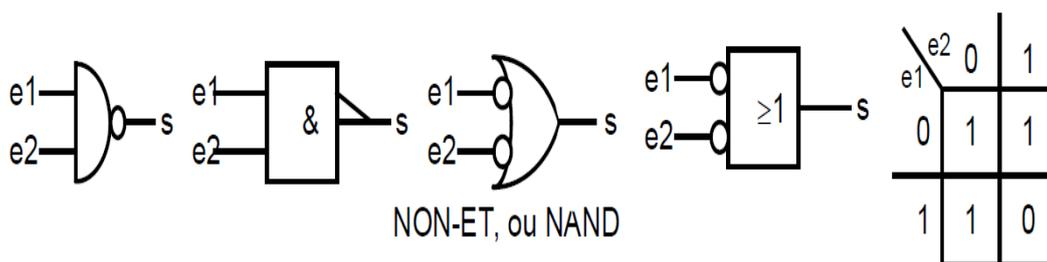


Figure. 2.4

◇ **Équations de NAND** : $s = \overline{e_1 \cdot e_2} = \overline{e_1} + \overline{e_2}$

◇ **Non Ou** : la figure 2.5 présente le symbole correspondant.

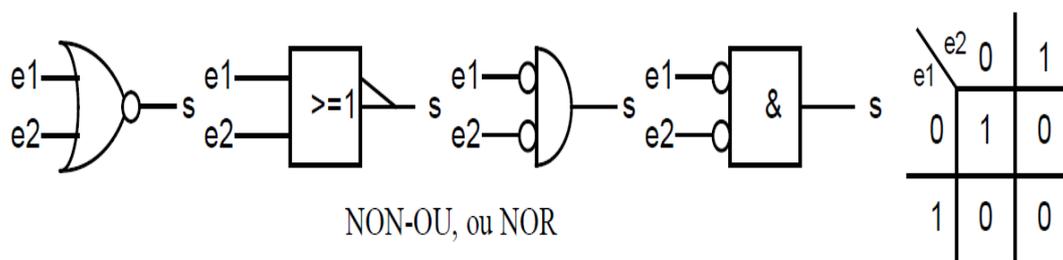


Figure. 2.5

◇ **Équations de NOR** : $s = \overline{e_1 + e_2} = \overline{e_1} \cdot \overline{e_2}$

Rem : *NAND* et *NOR* ne sont pas associatifs, ils ne sont donc pas généralisables, sans précaution, à un nombre quelconque d'entrées. Par contre on peut définir un opérateur qui est le complément du *ET* (resp. du *OU*) à $+s$ entrées comme un *NON-ET* (resp. *NON-OU*) généralisé.

2.4.2 Opérateur logique : OU exclusif

Opérateur aux multiples applications, le *OU EXCLUSIF* (*XOR*) est sans doute l'opérateur à deux opérandes le plus riche et le moins trivial. Il trouve ses applications dans les fonctions :

- ◇ *arithmétiques* : additionneurs, comparateurs et compteurs ;
- ◇ *de contrôle et de correction d'erreurs* ;
- ◇ Où l'on souhaite pouvoir programmer la convention logique ;
- ◇ de cryptage de l'information (p.ex. Sécurité des réseaux WiFi).

2.4.2.1 Symbole et définition

- ◇ **Ou exclusif** : la figure 2.6 présente le symbole qui lui est associé. On peut remarquer que cet opérateur prend la valeur 1 quand ses deux opérandes sont différents (somme modulo 2).

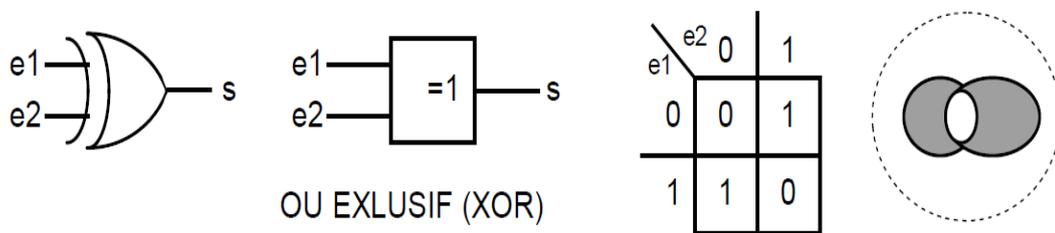


Figure. 2.6

- ◇ **Définition algébrique de XOR** : $s = e_1 \oplus e_2 = \overline{e_1} \cdot e_2 + \overline{e_2} \cdot e_1 = (e_1 + e_2) \cdot (\overline{e_1} + \overline{e_2})$
- ◇ Le *ou exclusif* possède les propriétés de l'addition : il est associatif, commutatif et a 0 comme élément neutre ; on peut donc le généraliser à un nombre quelconque d'opérandes d'entrée. Ainsi généralisé, l'opérateur devient une fonction qui vaut 1 quand il y a un nombre impair de 1 dans le mot d'entrée, d'où l'exemple de la figure 2.7, où la table de vérité concerne un opérateur à 4 opérandes.

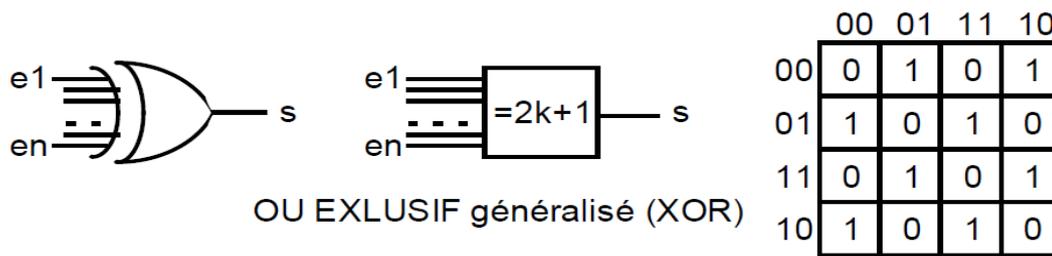


Figure. 2.7

2.4.3 Opérateur complémentaire du OU exclusif : XNOR

L'opérateur *OU INCLUSIF* a la particularité d'être obtenu, soit en complémentant la sortie de l'opérateur *XOR*, soit en complémentant l'une quelconque de ses entrées. On peut noter que cet opérateur prend la valeur 1 quand ses deux opérandes sont identiques.

2.4.3.1 Définition algébrique

◇ **Définition algébrique de XNOR :**

$$s = \overline{e_1 \oplus e_2} = \overline{e_1} \oplus e_2 = e_1 \oplus \overline{e_2} = e_1 \cdot e_2 + \overline{e_1} \cdot \overline{e_2} = (\overline{e_1} + e_2) \cdot (e_1 + \overline{e_2})$$

◇ Le *ou inclusif* indique l'identité entre les deux opérandes, d'où le nom parfois employé pour le désigner de « coïncidence ». Comme opérateur généralisé à un nombre quelconque d'opérandes, le complément du ou exclusif indique par un 1 qu'un nombre *pair* de ses opérandes vaut 1.

2.4.3.2 Application : addition en binaire

L'addition de 2 nombres *A* et *B* est réalisée en faisant la somme de 3 chiffres : les chiffres de rang *n* des 2 opérandes, et le report r_n issu de l'addition des chiffres de rang inférieur. Outre la somme, il faut également générer le report r_{n+1} pour l'étage suivant. En général, l'addition est régie par les équations suivantes :

$$0 + 0 = 0; \quad 0 + 1 = 1; \quad 1 + 0 = 1;$$

$$1 + 1 = 10;$$

Un tel opérateur schématisé sur la figure 2.8 est appelé *additionneur complet*. Le tableau 2.1 donne les valeurs des sorties en fonctions des différents codes d'entrées. **Les équations logiques** qui régissent l'additionneur 1 bit sont les suivantes :

$$r_{n+1} = b_n \cdot r_n + a_n \cdot r_n + a_n \cdot b_n$$

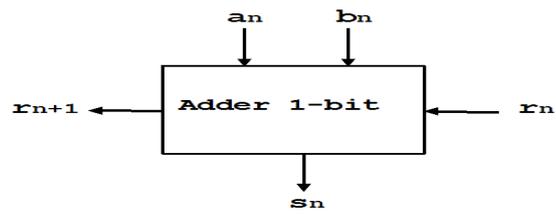


Figure. 2.8

a_n	b_n	r_n	r_{n+1}	s_n
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tableau. 2.1

$$s_n = \overline{a_n} \cdot \overline{b_n} \cdot r_n + \overline{a_n} \cdot b_n \cdot \overline{r_n} + a_n \cdot \overline{b_n} \cdot \overline{r_n} + a_n \cdot b_n \cdot r_n$$

La figure 2.9 montre le logigramme correspondant.

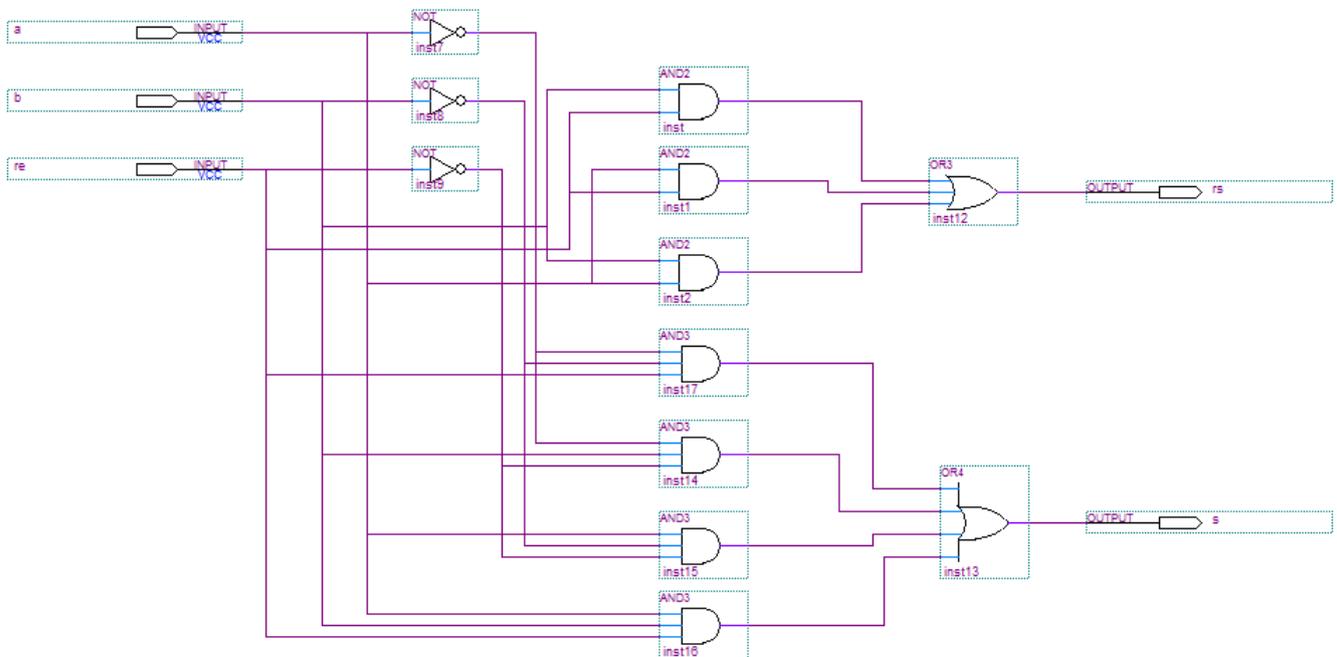


Figure. 2.9

En introduisant l'application de XOR en arithmétique, nous réécrivons **Les équations logiques** qui régissent l'additionneur 1 bit de la manière suivante :

◇

$$s_n = a_n \oplus b_n \oplus r_n$$

$$r_{n+1} = a_n \cdot b_n + (a_n + b_n) \cdot r_n$$

Ces équations permettent de représenter dans la figure 2.10, le logigramme le mieux adapté à l'additionneur 1 bit. La réalisation en fin de l'addition de deux nombres de taille n bits peut se

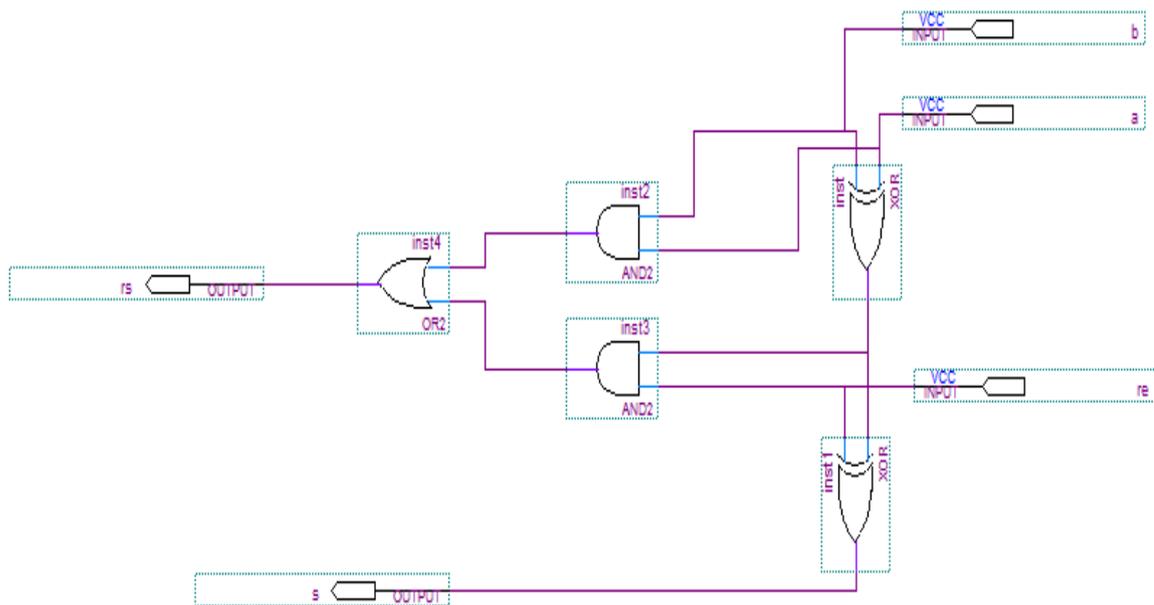


Figure. 2.10

faire en cascade des opérateurs précédents, on parle alors de « *propagation de retenue* ». La figure 2.11 présente un exemple d'additionneur 4 bit.

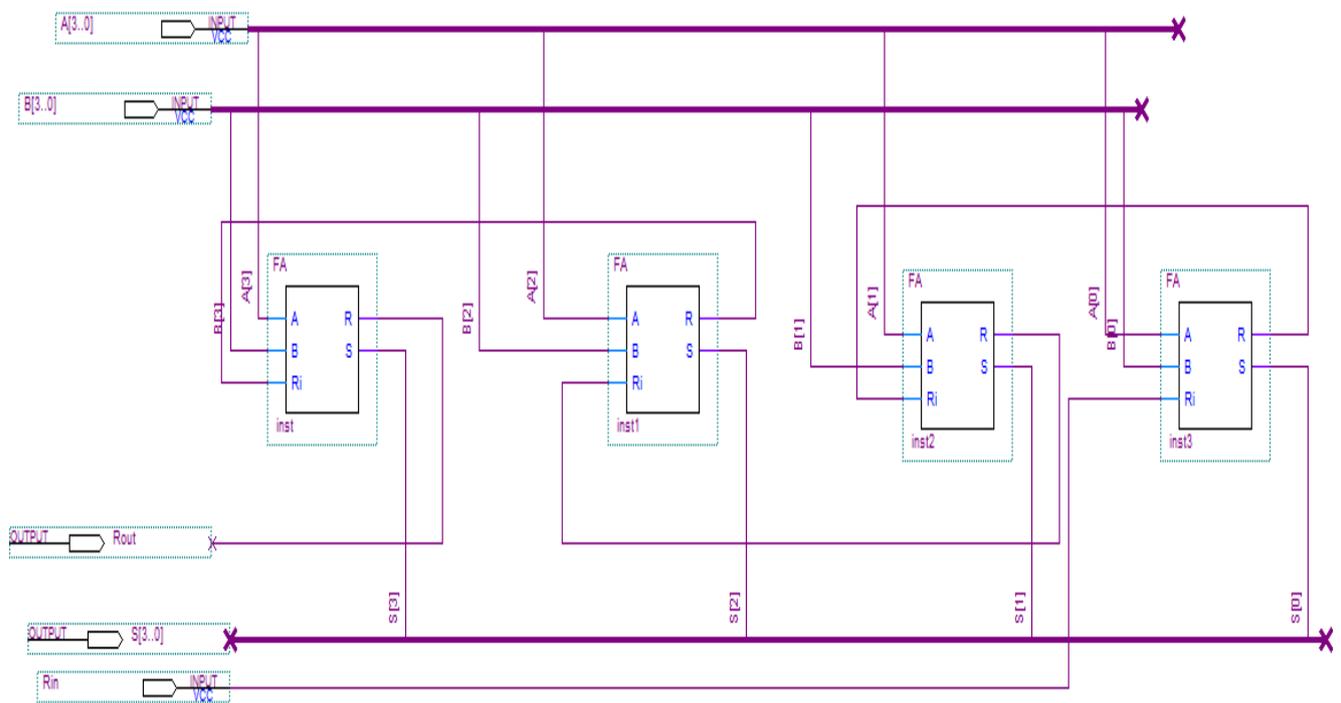


Figure. 2.11

3.1 Introduction

Le circuit logique combinatoire (CLC) est la première forme de circuits que nous aurons à étudier dans ce cours. Il est caractérisé par sa **propriété déterministe**, associant à tout code d'entrées **une et une seule combinaison** de sorties. Le comportement de tel système est régi par sa table de vérité ou (de manière équivalente) par sa fonction logique qui le décrit. Il s'agit ici de la relation qui unie les codes d'entrées aux combinaisons de sorties, et cela sans qu'on ait à se soucier de ce qui compose le circuit de l'intérieur. On parle alors de boîte noire pour séparer conceptuellement l'extérieur (entrées/sorties) du circuit de son intérieur (les portes logiques le composant).

L'étudiant doit assimiler et comprendre en profondeur tout le vocabulaire et notions présentés jusqu'ici.

3.2 Outils théoriques pour les CLC

La section suivante présente un contenu théorique d'importance, qui nous servira autant dans ce chapitre que dans le suivant. Nous utiliserons la représentation binaire des nombres qui fut introduite au chapitre 1. Le lecteur est donc invité à s'y référer au besoin.

On identifie, selon le principe de dualité propre à l'algèbre de Boole, deux formes **canoniques** pour exprimer une fonction logique quelconque :

- ◇ **Somme de Produits** : SOP $\sum \Pi$
- ◇ **Produit de Sommes** : POS $\Pi \sum$.

3.2.1 Première forme canonique

Les tables de vérité ont été introduites dans le chapitre précédent de manière relativement succincte, comme dans l'exemple du tableau 3.1. On tâche généralement de présenter l'ensemble

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Tableau. 3.1

des combinaisons d'entrées, et à les arranger par ordre croissant selon la représentation binaire des entiers telle que illustrée au tableau 3.2. Cette représentation nous permet de numérotter les

Ligne : i	X	Y	Minterms m_i
0	0	0	–
1	0	1	$m_1 = \bar{X} \cdot Y$
2	1	0	$m_2 = \bar{Y} \cdot X$
3	1	1	–

Tableau. 3.2

lignes de 0 à 3 pour une fonction de 2 variables, et plus généralement de 0 à $2^n - 1$ pour une fonction à n variables. La numérotation prendra toute son importance dans l'écriture des expressions canoniques qui suivent.

Une technique pour synthétiser directement un circuit logique à partir d'une table de vérité consiste à utiliser une somme canonique des produits basée sur des **minterms**. Un minterm désigné en utilisant la minuscule « m » avec le numéro de ligne en indice, est un terme de produit (c'est-à-dire une opération ET) qui sera **vrai** pour **un et un seul code** d'entrée. Le minterm doit contenir chaque variable d'entrée dans son expression. La figure 3.1 montre le logigramme correspondant.

On en déduit l'expression de F sous forme canonique disjonctive et son écriture condensée (moyen compact de décrire la fonction d'un circuit logique en listant simplement les lignes qui correspondent à une sortie de 1 dans la table de vérité) :

$$F = \bar{X} \cdot Y + \bar{Y} \cdot X = \sum m(1, 2) \quad (3.1)$$

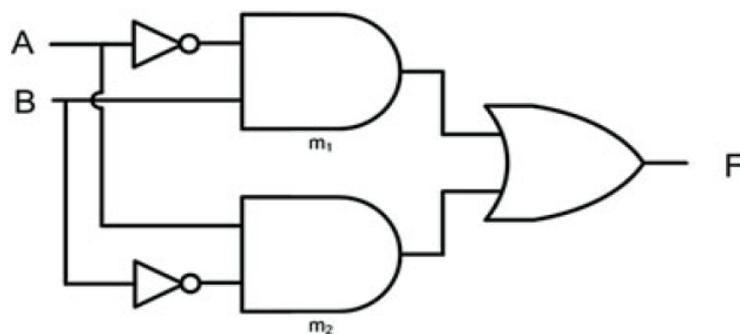


Figure. 3.1

De la première écriture (SOP), nous relevons que toute fonction peut être écrite sous la forme d'une somme des minterms m_i pour lesquels F vaut 1.

Cette expression est dite canonique disjonctive (en référence à la disjonction des termes opérée par l'opérateur logique (OU,+)). On l'appelle également **forme canonique** de somme de produits. Une somme de produits est sous forme canonique si toutes les variables apparaissent dans tous les termes des produits qui la composent. Cette écriture est dite canonique car elle est unique pour chaque fonction. Une expression canonique n'est cependant pas optimale ; nous verrons dans la suite qu'elle sert de base aux méthodes de simplification.

3.2.2 Seconde forme canonique

Une autre technique pour synthétiser directement un circuit logique à partir d'une table de vérité consiste à utiliser un produit canonique des sommes basée sur **maxterms**. Un maxterm, noté en utilisant la majuscule « M » avec le numéro de ligne en indice, est un terme de somme (c'est-à-dire une opération **OR**) qui sera **faux** pour un et un seul code d'entrée. Le tableau 3.3 présente cette seconde méthode.

Ligne : i	X	Y	Maxterms m_i
0	0	0	$M_0 = X + Y$
1	0	1	—
2	1	0	—
3	1	1	$M_3 = \bar{X} + \bar{Y}$

Tableau. 3.3

La figure 3.2 présente le logigramme correspondant.

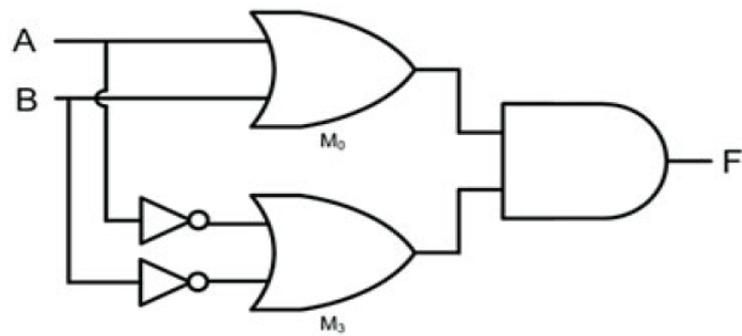


Figure. 3.2

On en déduit l'expression de F sous forme canonique conjonctive et son écriture condensée :

$$F = (X + Y) \cdot (\bar{X} + \bar{Y}) = \Pi M(0, 3) \quad (3.2)$$

De la seconde écriture (POS), nous relevons que toute fonction peut être écrite sous la forme d'un produit des maxterms M_i pour lesquels F vaut 0.

L'exemple illustré sur la figure 3.3 montre comment les écritures condensées (minterms et maxterms) produisent exactement la même fonction logique mais de manière complémentaire.

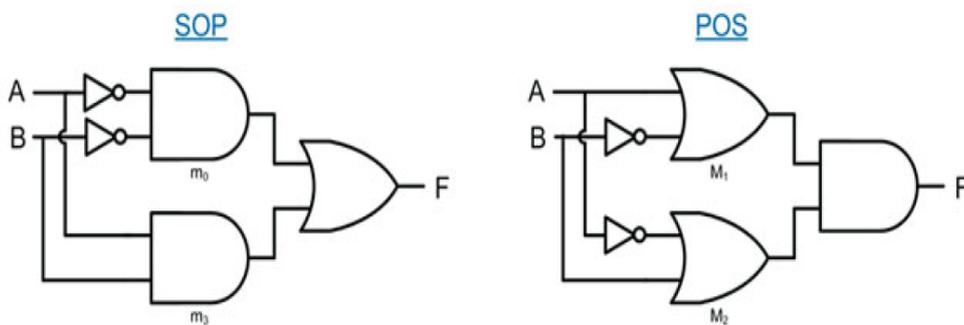


Figure. 3.3

Nous en déduisons qu'il y'a donc équivalence entre les deux formes.

3.3 Minimisation logique

Nous examinons maintenant comment réduire les expressions canoniques en des formes équivalentes qui utilisent moins de portes logiques. Cette minimisation est essentielle pour réduire la complexité de la logique avant la mise en œuvre dans des circuits réels.

3.3.1 Minimisation algébrique

Les expressions canoniques peuvent être réduites algébriquement en appliquant les théorèmes de l'algèbre de Boole (distributivité, théorèmes des compléments, d'identité, ...).

Pex. : On se donne la T.V. suivante (Tableau 3.4). On écrit :

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

Tableau. 3.4

$$F = \bar{X} \cdot Y + \bar{Y} \cdot X + X \cdot Y \quad (3.3)$$

En appliquant les propriétés de Boole, il vient :

$$F = X \cdot (Y + \bar{Y}) + \bar{X} \cdot Y$$

$$F = X + \bar{X} \cdot Y$$

Finalement,

$$F = X + Y$$

La méthode devient peu évidente si la fonction F est compliquée.

3.3.2 Minimisation utilisant le tableau de Karnaugh

Le tableau de Karnaugh est un moyen de représenter la table de vérité sous forme d'une grille bidimensionnelle. Ceci permet la minimisation logique grâce à un processus graphique (utilisant l'équation $X + \bar{X} = 1$). Les cellules voisines, ou adjacentes (qui ont 1 côté en commun) ne diffèrent que **d'un bit** dans leurs codes d'entrée (Code Gray). Un regroupement de cellules voisines ou termes mineurs, est appelé Impliquant ; il associe $2^m(1, 2, 4, 8, \dots)$ mintermes ; **la variable d'entrée qui change, disparaît**. 2 impliquants voisins s'associent pour former un nouveau impliquant comprenant 2 fois plus de termes mineurs.

Pex.1 : La solution minimale relative au T.K 3.5, est :

$$F = Y$$

$X \ Y \ Z$	0 0	0 1	1 1	1 0
0	0	0	1	1
1	0	0	1	1

Tableau. 3.5

Un impliquant qui ne peut être inclus dans un autre impliquant plus grand est dit impliquant 1^{er}.

La solution minimale est constituée uniquement d'impliquants 1^{ers}.

Pex.2 : La solution minimale relative au T.K 3.6, est :

$X \ Y \ Z \ T$	0 0	0 1	1 1	1 0
0 0	0	1	0	0
0 1	0	1	0	0
1 1	0	1	1	1
1 0	0	1	1	1

Tableau. 3.6

$$F = \bar{Z} \cdot T + X \cdot Z + (X \cdot T)$$

L'impliquant 1^{er} $X \cdot T$ n'est pas essentiel puisqu'il est intégralement compris dans les 2 autres impliquants 1^{ers}. Il s'agit d'une redondance d'information. Les impliquants 1^{ers} qui possèdent au moins 1 minterme qui n'est pas inclus dans un autre impliquant 1^{er} sont dits impliquants 1^{ers} essentiels. Ces derniers font toujours partie de la solution minimale. Au final,

$$F = \bar{Z} \cdot T + X \cdot Z$$

Pex.3 : La solution minimale relative au T.K 3.7, est :

$$F = \bar{T} + \bar{Z} \cdot Y + X \cdot \bar{Y} \cdot Z$$

Rem : Il arrive que la fonction de sortie F puisse prendre indifféremment la valeur 0 ou 1 pour certains états d'entrée. La sortie indifférente est symbolisée par : $F = \Phi$ (*Don't Happen ou Don't Care*).

$X \ Y \ Z \ T$				
	0 0	0 1	1 1	1 0
0 0	1	0	0	1
0 1	1	1	0	1
1 1	1	1	0	1
1 0	1	0	1	1

Tableau. 3.7

3.4 Circuits usuels non arithmétiques

Dans ce qui précède, nous avons synthétisé des circuits numériques en partant d'une table de vérité ou d'une fonction logique et en les réalisant à l'aide de portes logiques. Dans la pratique, de nombreuses fonctions de haut niveau sont réalisées par des circuits prédéfinis, communément appelés circuits usuels, dont le concepteur fait usage pour traduire immédiatement le comportement de son circuit. Les circuits usuels que nous présentons ici sont ceux qui ne sont pas nécessairement impliqués dans des opérations arithmétiques.

3.4.1 Multiplexeur

Le multiplexeur (souvent désigné par **mux**) est sans conteste le circuit usuel le plus utilisé. Le rôle du multiplexeur est d'acheminer au choix une entrée parmi +s. Le multiplexeur possède donc +s entrées et des signaux de contrôle permettant d'acheminer 1 de ces signaux vers sa sortie.

Le plus simple des multiplexeurs est le mux 2 vers 1, représenté dans la figure 3.4.

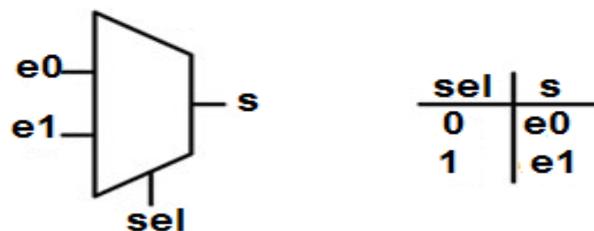


Figure. 3.4

Le logigramme qui correspond au mux 2 vers 1 est présenté sur la figure 3.5.

$D_{0,1}$: signaux de données ; C : signal de contrôle.

Le rapport de simulation sous Quartus, du résultat associé au mux 2 vers 1, est présenté sur la

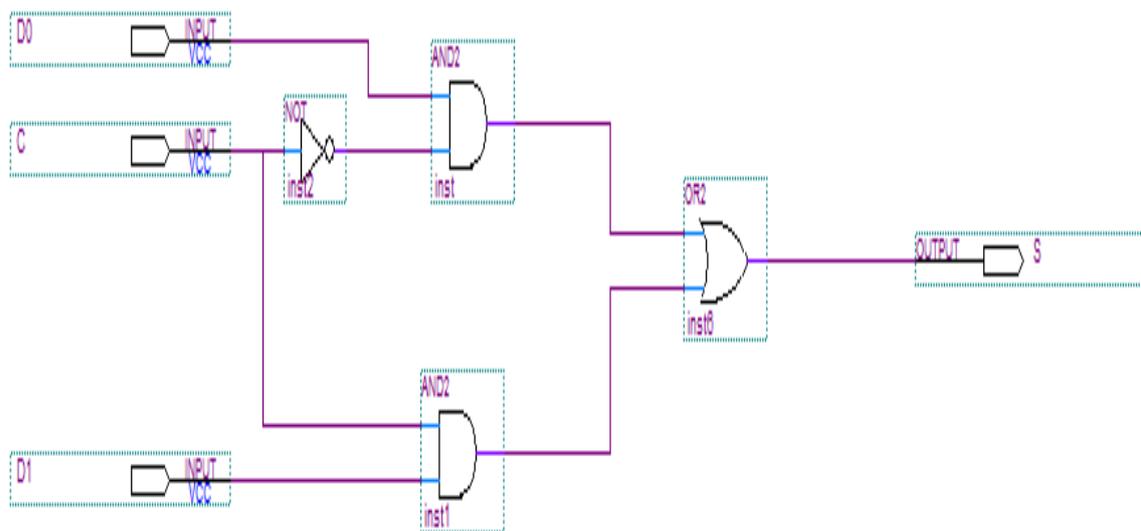


Figure. 3.5

figure 3.6.

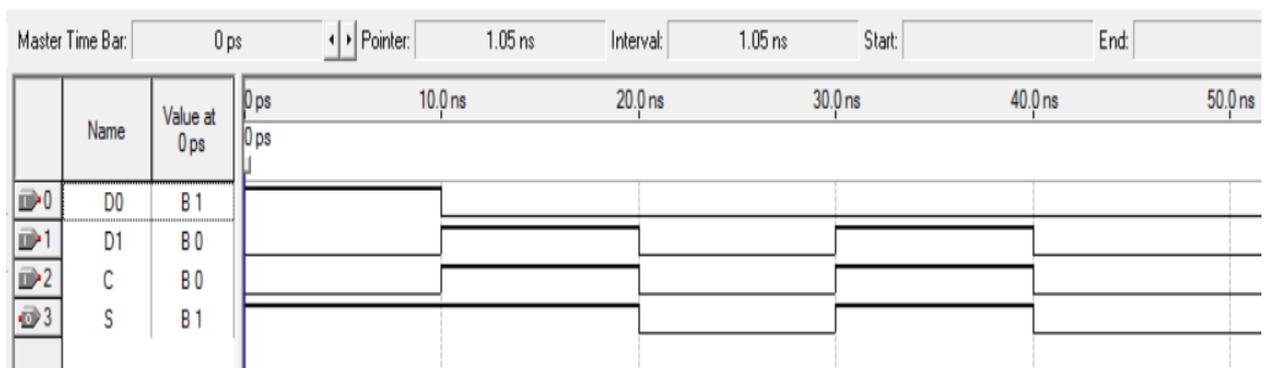


Figure. 3.6

De tels circuits sont principalement appliqués pour implémenter des **connexions contrôlables**, comme le montre la figure 3.7.

- ◊ $control = 1 \implies$: circuit A de données se connecte au circuit C
- ◊ $control = 0 \implies$: circuit B de données se connecte au circuit C

La figure 3.8 représente un autre exemple de multiplexeurs ; il s'agit du Mux 4 vers 1. Le multiplexeur 4 vers 1 possède 2 signaux de contrôle.

De même, nous présentons sur la figure 3.9 le rapport de simulation du résultat donné par le mux 4 vers 1.

Nous pouvons définir de la même façon des mux 8 vers 1 avec 3 signaux de contrôle, des

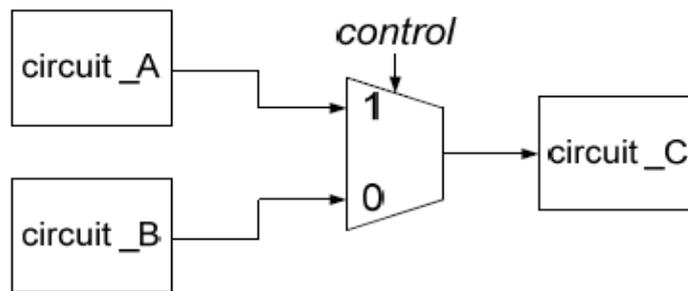


Figure. 3.7

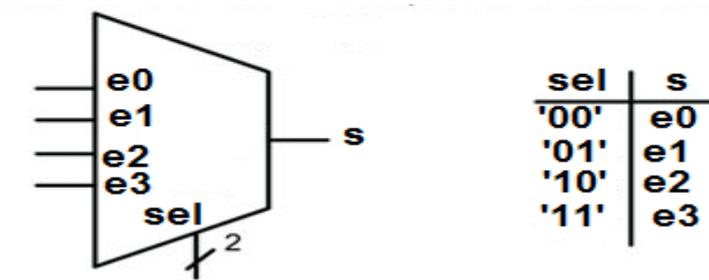


Figure. 3.8

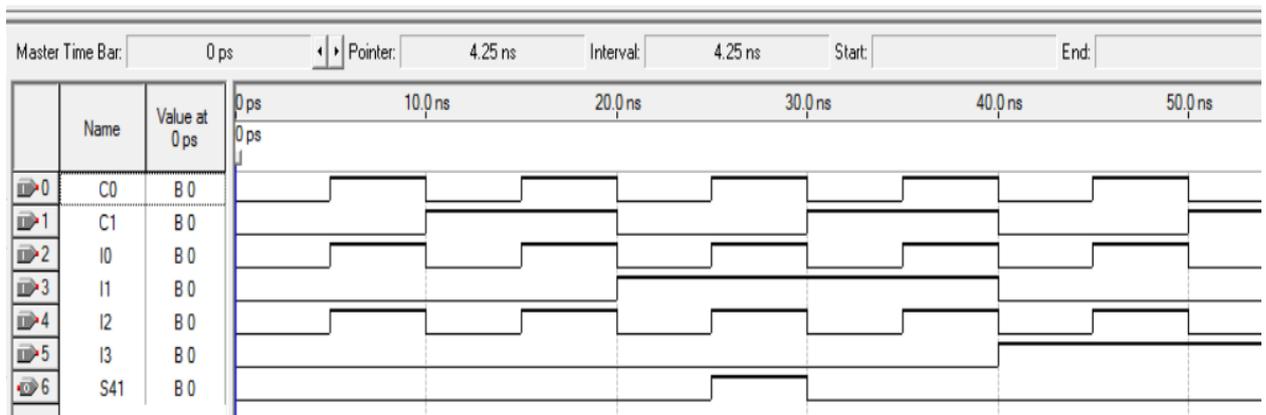


Figure. 3.9

mux 16 vers 1 avec 4 signaux de contrôle ...

D'autres types de multiplexeurs peuvent être également définis. La figure 3.10 montre un cas de multiplexeur 2 vers 1 à m bits. Il est contrôlé par le même signal.

3.4.2 Multiplexeurs et génération de fonctions combinatoires

La figure 3.11 montre un cas d'opération de multiplication, réalisée avec un mux 2 vers 1. La figure 3.12 présente le rapport de simulation de cette opération de multiplication. De même, la figure 3.13 montre un cas d'opération déterminant l'imparité, réalisée également avec un mux 2

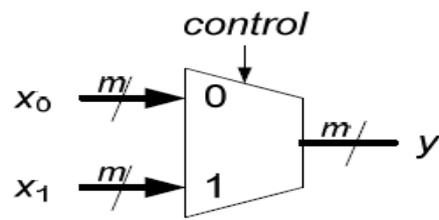


Figure. 3.10

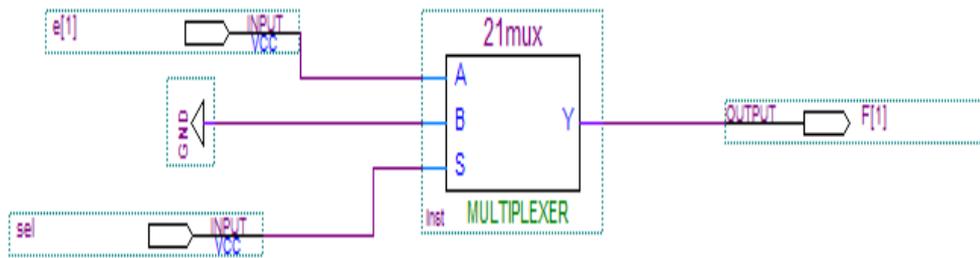


Figure. 3.11

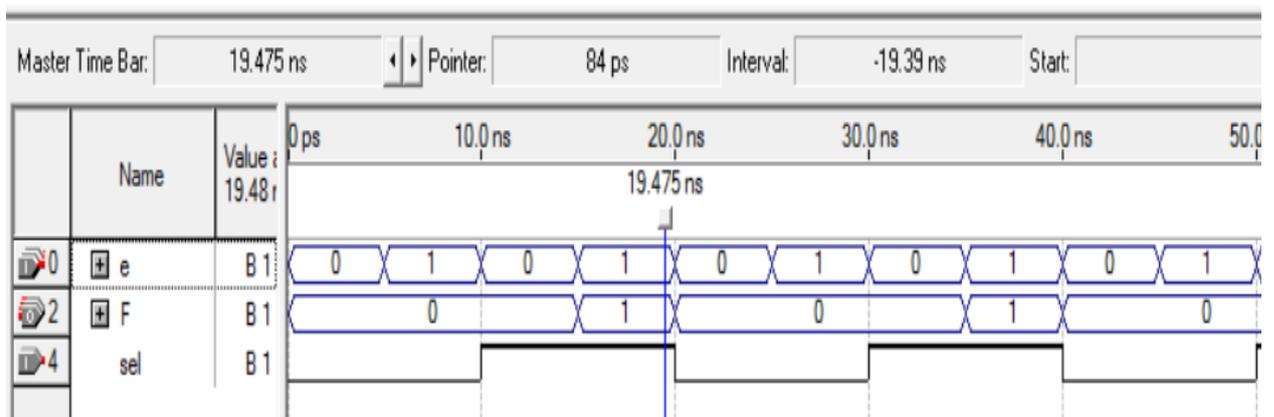


Figure. 3.12

vers 1. La figure 3.14 présente le rapport de simulation de cette mesure de l'imparité.

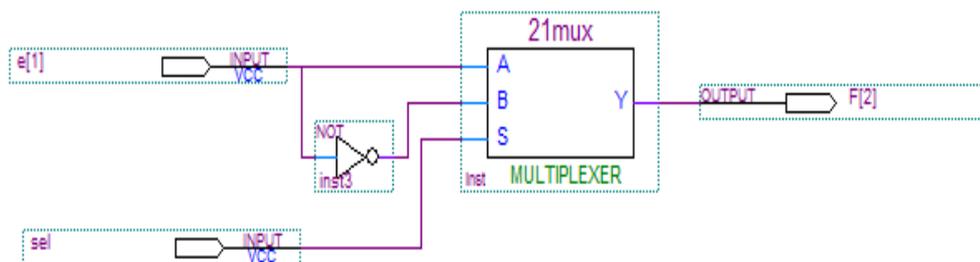


Figure. 3.13

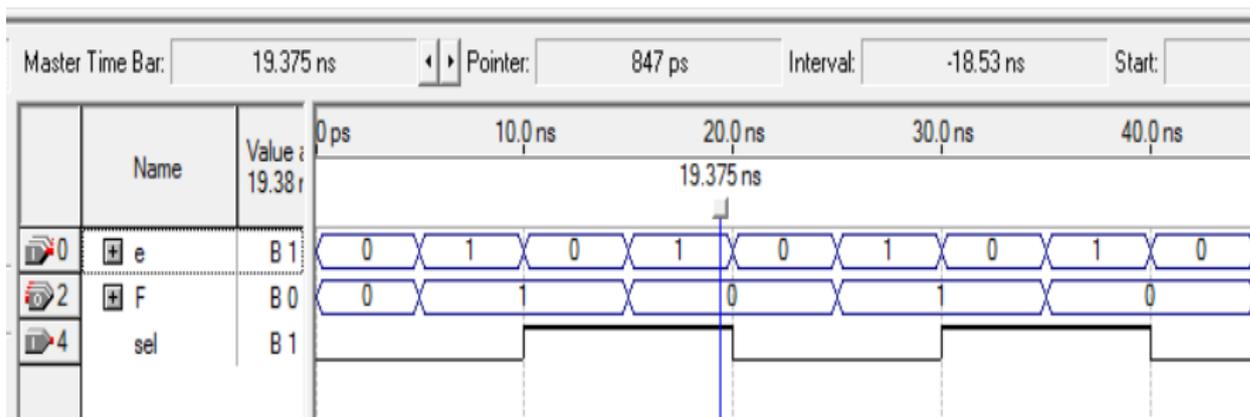


Figure. 3.14

$$F_1(e_1, e_2) = e_1 \cdot e_2 ; F_2(e_1, e_2) = e_1 \oplus e_2$$

Nous pouvons réaliser grâce aux circuits usuels, des fonctions à partir de T.V ; comme exemple l'expression logique suivante :

$$F_3(e_1, e_2, e_3) = \sum m(0, 1, 4, 7)$$

La figure 3.15 présente le circuit correspondant.

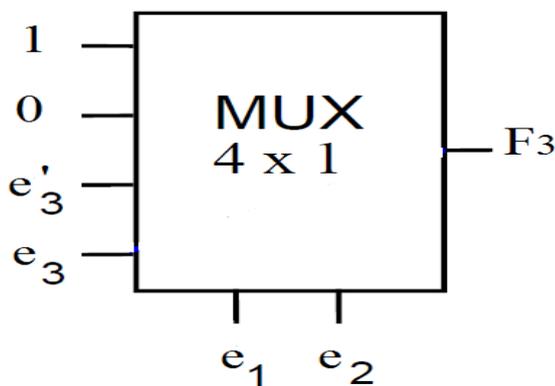


Figure. 3.15

3.4.3 Décodeur

Le démultiplexeur (**démux**) fonctionne de façon inverse à celle du multiplexeur. Il reçoit n signaux de contrôle $c_{n-1} \dots, c_1, c_0$, et 1 entrée e à acheminer vers l'une des 2^n sorties possibles $s_0, \dots, s_i, \dots, s_m$. En pratique, le démux est peu utilisé. On lui préfère un composant dérivé : le décodeur. L'entrée e ayant été fixée à 1 ; il reçoit n signaux de contrôle et donne 2^n sorties. La figure 3.16 présente le schéma général d'un décodeur.

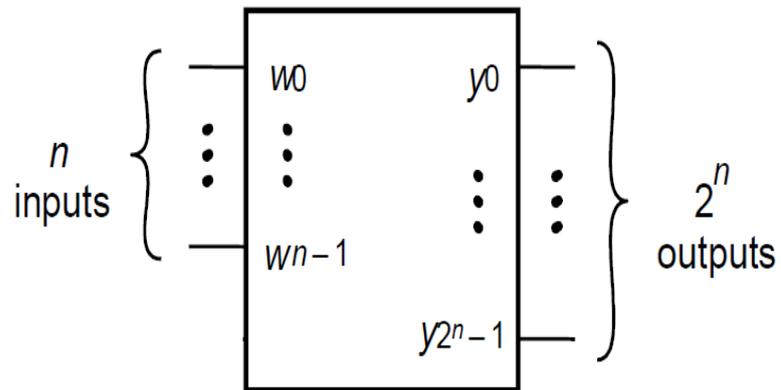


Figure. 3.16

La figure 3.17 présente un exemple de décodeurs ; il s'agit du décodeur 1×2 .

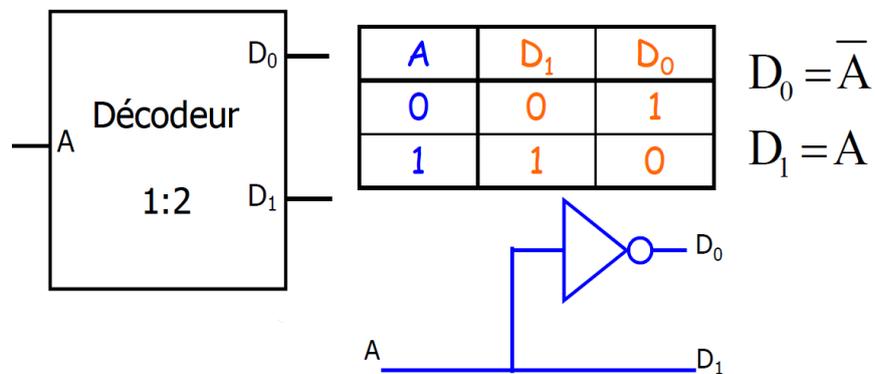


Figure. 3.17

Le circuit peut également être exploité pour réaliser des fonctions logiques en tenant compte du fait que les sorties correspondent aux minterms d'une expression logique.

Comme exemple, le circuit de la figure 3.18. Il présente un décodeur 2×4 , avec entrée **Enable (E)** ; E permet d'activer ou bien désactiver le système (Utile pour synthétiser de grands décodeurs).

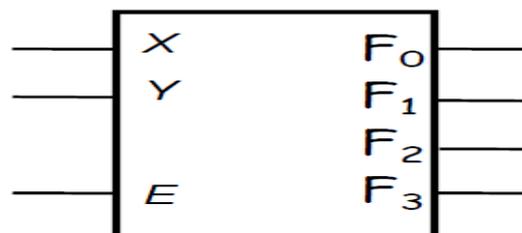


Figure. 3.18

Le schéma de la figure 3.19 montre les fonctions réalisées.

E	X	Y	F ₀	F ₁	F ₂	F ₃
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	d	d	0	0	0	0

Mintermes

- F₀=X'Y'E m₀
- F₁=X'YE m₁
- F₂=XY'E m₂
- F₃=XYE m₃

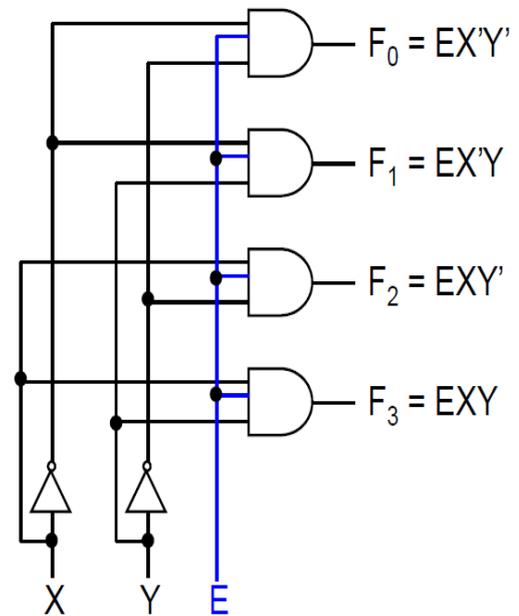


Figure. 3.19

Un autre exemple de fonction générée à partir de T.V, est présenté sur la figure 3.20.

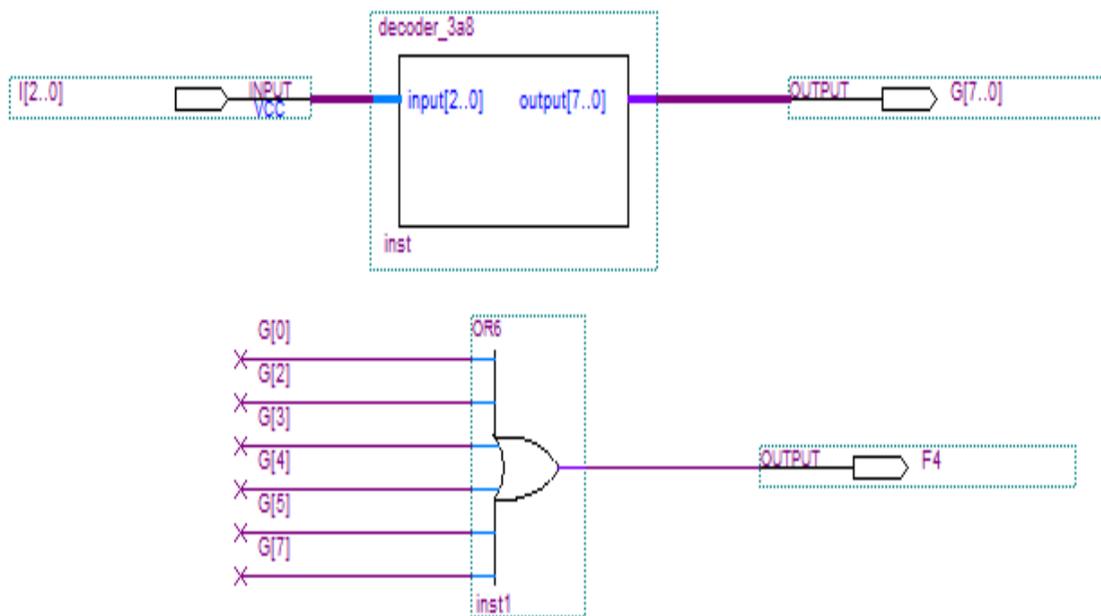


Figure. 3.20

$$F_4(I_0, I_1, I_2) = \Pi M(1, 6)$$

Le rapport de vérification est présenté sur la figure 3.21.

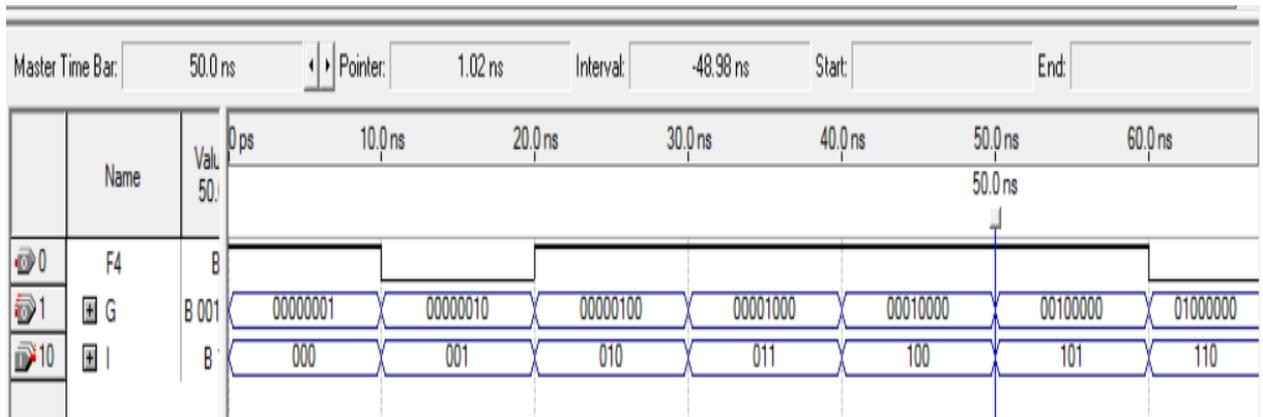


Figure. 3.21

Ce test est bien conforme à nos attentes.

4.1 Introduction

Vu la limitation du champ d'application des circuits logiques combinatoires, les circuits séquentiels ont pris de l'ampleur dans les systèmes numériques modernes. Récemment, une tendance de recherche s'avère notable dans le domaine de **création des systèmes sophistiqués et intelligents**.

Dans ce chapitre, on définit en premier lieu **les éléments de mémoire**, outils nécessaires pour construire en second lieu, les circuits séquentiels.

4.1.1 Séquentialité ?

La figure 4.1 montre le schéma général d'un circuit logique séquentiel. Le temps de consultation des données dans un tel système joue un rôle important dans son fonctionnement.

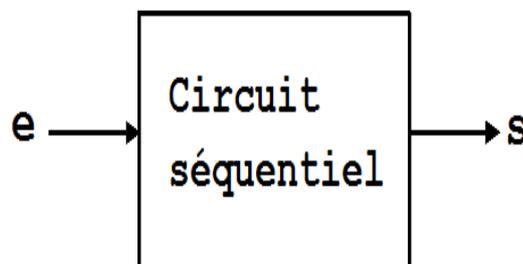


Figure. 4.1

En effet, les signaux de sorties $S(t)$ dépendent non seulement des signaux des entrées appliquées $e(t)$ au même instant t , mais aussi des informations traitées aux instants précédents

($t - 1, t - 2, \dots$). Les valeurs des signaux qui commandent alors un tel circuit sont fonction de la séquence des événements E_1, E_2, \dots, E_n . On parle de circuit séquentiel : Un nouveau événement (p.ex. E_1) n'est pas traité jusqu'à ce que cette séquence d'événements soit terminée. En conséquence, la mise en œuvre d'une Mémoire est nécessaire.

4.1.2 Exemple : : détection d'une séquence

- ▷ Système d'ouverture de porte avec code d'accès.
- ▷ Début
- ▷ Machine reçoit en entrée e une série de chiffres tapée sur un clavier numérique.
- ▷ Si la machine reçoit la bonne séquence de chiffres 0, 2, 3, 5, la porte s'ouvre grâce au signal de sortie.
- ▷ Si $e(t - 3) = 0$ ET $e(t - 2) = 2$ ET $e(t - 1) = 3$ ET $e = 5$ ALORS $s = 1$
- ▷ Si NON $s = 0$
- ▷ FIN.

4.2 Éléments de mémoire

4.2.1 Cellules de base

La figure 4.2 montre l'exemple d'un circuit logique séquentiel, où une rétroaction positive (boucle de retour, ou feed back), est appliquée entre la sortie et l'entrée.

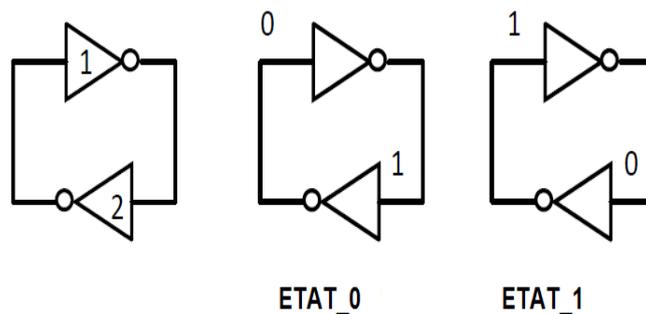


Figure. 4.2

- Le circuit enregistrera ou stockera un 0 logique sans être piloté par aucune autre entrée \Leftrightarrow ÉTAT 0.
- Il fonctionne en complémentaire : Il enregistre ou stocke un 1 logique sans être piloté par aucune autre entrée \Leftrightarrow ÉTAT 1.

Cette rétroaction positive est synonyme d'une **Base de Stockage**. De tels éléments à mémoire représentent le point de départ pour construire une grande variété de systèmes logiques séquentiels.

4.2.2 Circuit bistable SR

Étudions le circuit simple bouclé de la figure 4.3.

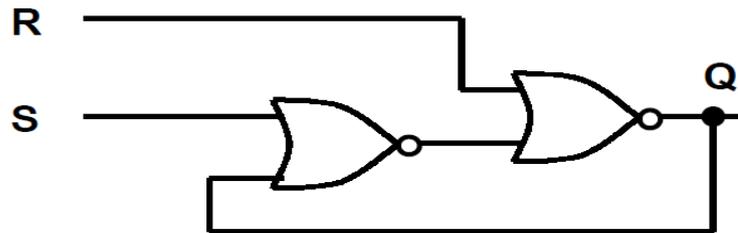


Figure. 4.3

On rappelle au tableau 4.1, la T.V. de la porte NOR.

e_1	e_2	S
0	0	1
0	1	0
1	0	0
1	1	0

Tableau. 4.1

Selon les deux entrées du circuit, +s cas sont à envisager.

- ◇ Si $R = 1$, le système est réinitialisé et enregistre un 0 à sa sortie Q ;
- ◇ Si $S = 1$, le système place un 1 à sa sortie Q (N'est possible que si $SR \neq 11$) ;
- ◇ Si $SR = 00$, aucune influence sur la sortie Q du système. Autrement-dit, la sortie garde son état passé : le système se souvient de son état antérieur. Il est stable, et mémorise la dernière valeur enregistrée.

Le tableau 4.2 résume le comportement du circuit de la bistable SR.

En définitive, une bistable SR est un élément à mémoire qui enregistre 1 seul bit.

4.2.3 Bistable SR avec signal d'activation

Pour commander le fonctionnement de la bistable SR, un signal d'activation périodique H (appelé horloge) est appliqué au circuit. La bistable fonctionne alors normalement quand H est

S	R	Q^+	\overline{Q}^+	Fonction
0	0	Q	\overline{Q}	Mémoire
0	1	0	1	RAZ
1	0	1	0	Mise à 1
1	1	0	0	Interdit (Instable)

Tableau. 4.2

actif ($H = 1$), et mémorise la dernière valeur enregistrée lorsque H est inactif ($H = 0$). Il suffit pour cela d'introduire des portes ET aux entrées, ce qui nous permet d'obtenir le système de la figure 4.4.

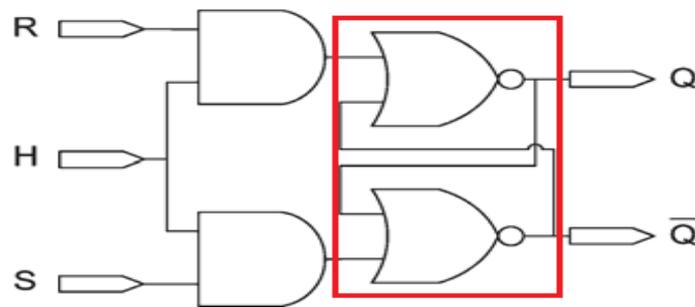


Figure. 4.4

Le tableau 4.3 présente la table de vérité rattachée au circuit de la bistable SR avec signal d'activation H. Comme déduction, il faut éviter la situation où $SR = 11$, mais uniquement dans

H	S	R	Q^+	\overline{Q}^+	Fonction
0	—	—	Q	\overline{Q}	Mémoire
1	0	0	Q	\overline{Q}	Mémoire
1	0	1	0	1	RAZ
1	1	0	1	0	Mise à 1
1	1	1	1	1	Interdit (Instable)

Tableau. 4.3

le cas où $H = 1$. L'utilisation du signal H peut être imaginée dans le cas d'un grand système où plusieurs bistables travaillent en interaction. On veut figer le système dans l'état où il se trouve. On pose alors $H = 0$, et toutes les mémoires demeurent au dernier état enregistré.

4.2.4 Bistable D

On part du constat que dans le cas de la bistable SR avec signal d'activation H, le cas $HSR = 100$ est **redondant** avec celui où $H = 0$. Il devient clair alors que les signaux S et R sont constamment **inverses** l'un de l'autre. La bistable D est un dérivé de la SR avec signal d'activation H. En effet, il suffit de réunir les signaux SR par un seul signal D qui correspond à la donnée que l'on veut écrire dans la mémoire de la bistable. On peut alors obtenir le circuit de la figure 4.5.

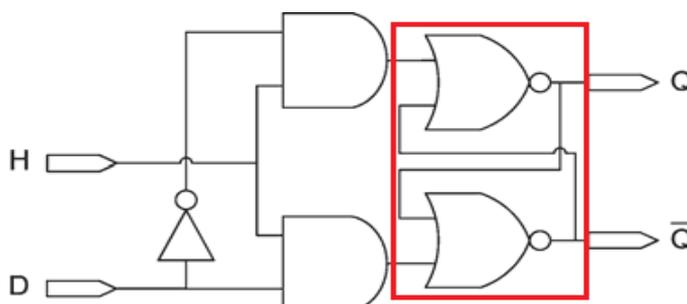


Figure. 4.5

Le tableau 4.4 montre la TV, qui résume le comportement de la bistable D.

H	D	Q	Q ⁺
0	-	-	Q
1	0	-	0
1	1	-	1

Tableau. 4.4

En définitive, la bistable D conserve sa valeur (sur le signal Q) lorsque $H = 0$; elle prend celle de D si $H = 1$ (**État futur si $H = 1$: $Q^+ = f(Q, D) = D$**).

La bistable D est **transparente** puisqu'elle correspond à un fil liant D à Q lorsque $H = 1$ (les délais de propagations sont mis à part).

Les figures 4.6, et 4.7 montrent un **résumé du comportement de la bistable D (Latch D ; Load = H)** .

4.2.5 Bascule D de type maître esclave

On pourrait vouloir disposer d'un circuit de mémoire qui fonctionne non pas sur la durée d'activation (fonctionnement transparent) du signal H mais à un moment précis : les fronts mon-

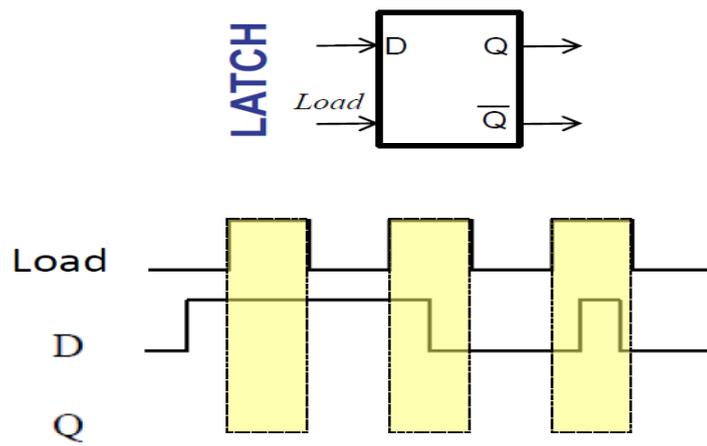


Figure. 4.6

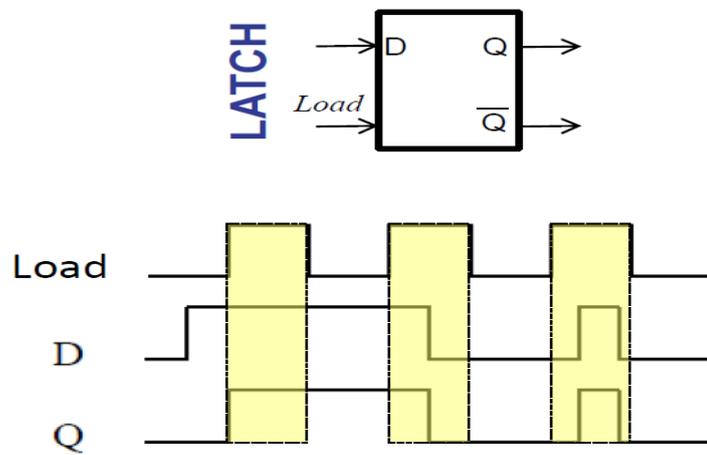


Figure. 4.7

tant 0 \rightarrow 1 ou descendant 1 \rightarrow 0 du signal d'horloge H (CLK) pourraient alors être utilisés pour indiquer ce changement, et ce conformément à la figure 4.8.

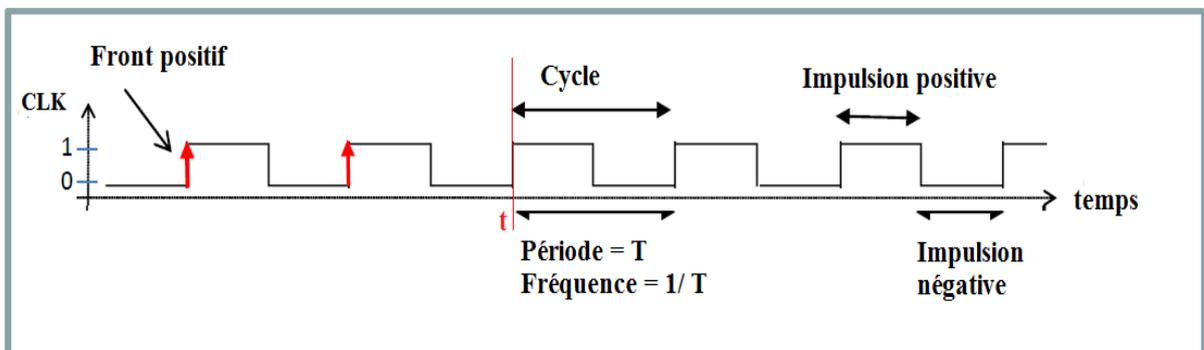


Figure. 4.8

La bascule D de type maître esclave utilise une bistable D pour réussir cette performance. L'idée est de mettre en cascade deux bistables qui vont fonctionner en conjonction. Pour ce faire, les deux bistables partagent le signal de commande CLK, à cette différence que la première (appelée Maître) reçoit le signal H_1 (Inverse de CLK), et la seconde (appelée esclave) reçoit le signal H_2 (CLK), conformément à la figure 4.9.

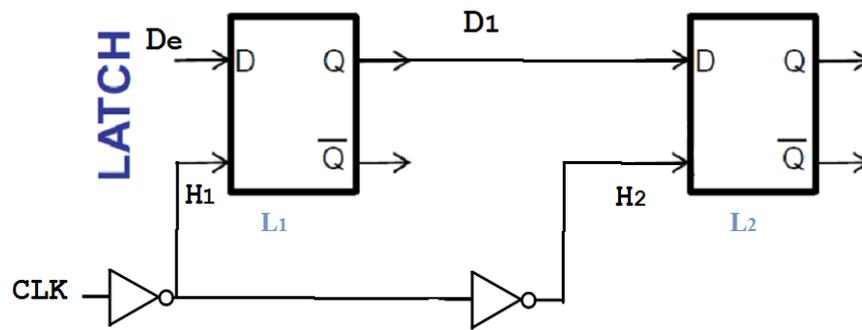


Figure. 4.9

Le système va passer par 2 phases de transition dans une période de d'Horloge CLK. Pour nous en convaincre, regardons les **chronogrammes** des figures 4.10, 4.11, et 4.12, montrant le comportement de la bascule D.

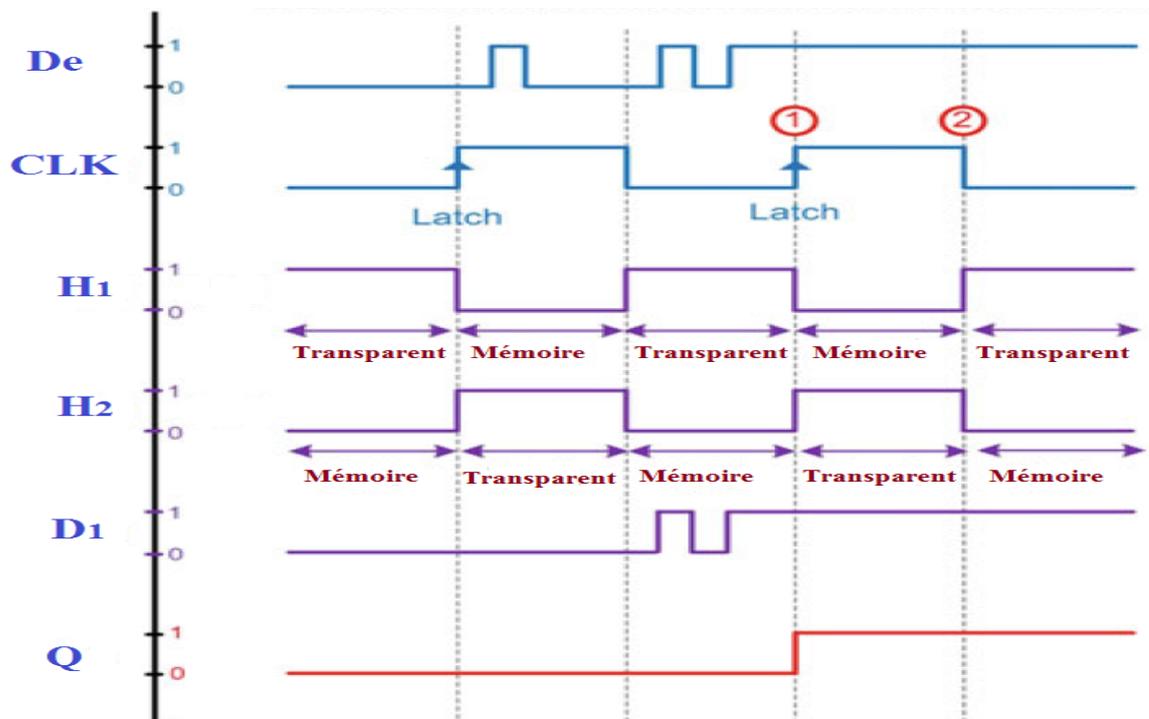


Figure. 4.10

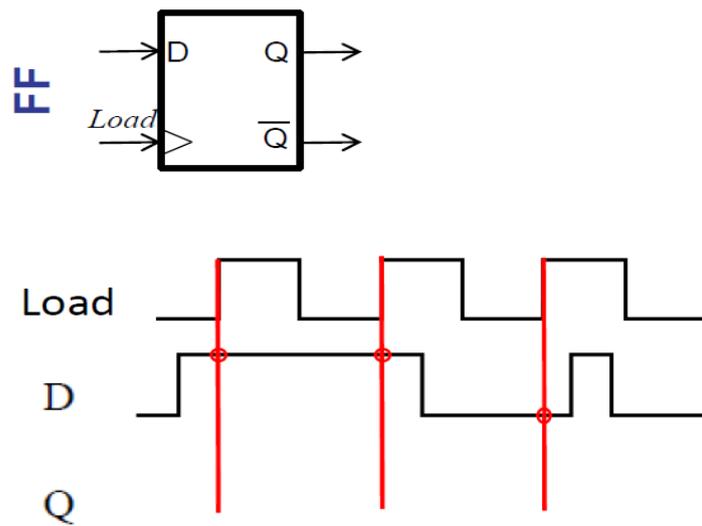


Figure. 4.11

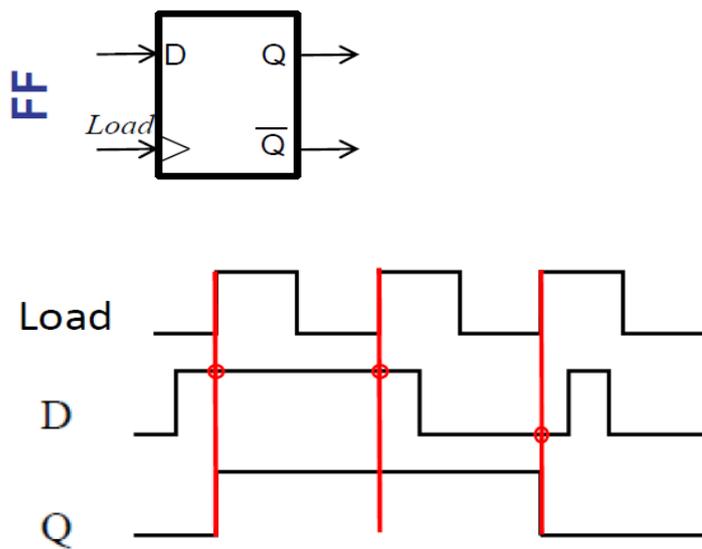


Figure. 4.12

Dans le chronogramme de la figure 4.10, D_1 suit exactement le comportement de D_e lorsque $H_1 = 1$. Si $H_1 = 0$, D_1 conserve la dernière valeur retenue (Celle de D_e précédant le front positif). Dans le même chronogramme, Q suit exactement le comportement de D_1 lorsque $H_2 = 1$ du fait que le signal d'horloge à l'entrée CLK est identique à H_2 . Lorsque $H_2 = 0$, la dernière valeur retenue est maintenue. Or rappelons que durant ce temps, le maître conserve la dernière valeur retenue sur le front montant. Ainsi, le système (la bascule) est sûr de conserver à sa sortie (et durant toute une période de CLK) la valeur du front positif.

Le chronogramme de la figure 4.12, illustre le fait que la sortie Q prend la valeur de D sur le front positif (indiqué par les traits rouges verticaux la figure 4.11). Notons en fin que le nom même de bascule fait référence au basculement (commutation) de la valeur de la sortie Q au moment précis où le signal d'horloge CLK passe de 0 à 1 (front positif). Cette bascule D qui réagit au front montant, est reportée à la figure 4.13 (c'est le triangle à l'entrée qui représente la sensibilité au front).

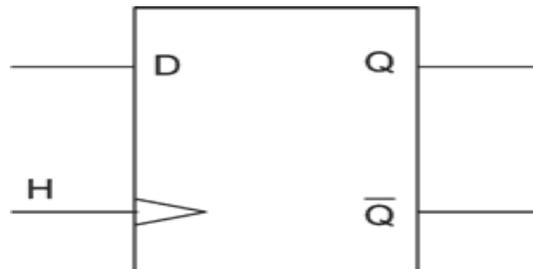


Figure. 4.13

Le tableau 4.5 présente la table de vérité de la bascule D agissant au front ascendant de l'horloge CLK. La flèche orientée vers le haut rend compte du front positif de l'horloge qui provoque le changement de valeur à la sortie.

CLK	D	Q^+
0	–	Q
1	–	Q
↑	0	0
↑	1	1

Tableau. 4.5

Remarque :

Il existe un autre modèle de bascule D qui n'est pas réalisé sur la base de la bistable D de type maître esclave. La figure 4.14 présente le symbole de la bascule D où le cercle d'inversion indique qu'il s'agit d'un front descendant.

Le tableau 4.6 donne la table de vérité de la bascule D dont la transition est faite selon le front négatif de l'horloge, et non pas sur le front positif.

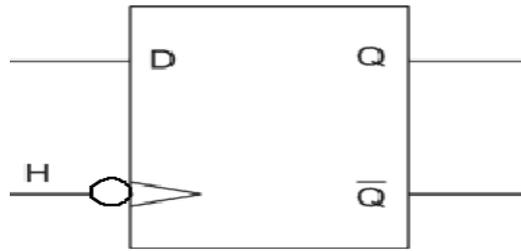


Figure. 4.14

CLK	D	Q ⁺
0	—	Q
1	—	Q
↓	0	0
↓	1	1

Tableau. 4.6

4.2.6 Entrées asynchrones SET et RESET

Reset, Set : 2 entrées asynchrones parmi d'autres, pouvant changer l'état de la bascule D, indépendamment de l'entrée synchrone. La figure 4.15 présente un cas d'exemple (Reset : Mettre l'état 0 ; Set : Mettre l'état 1).

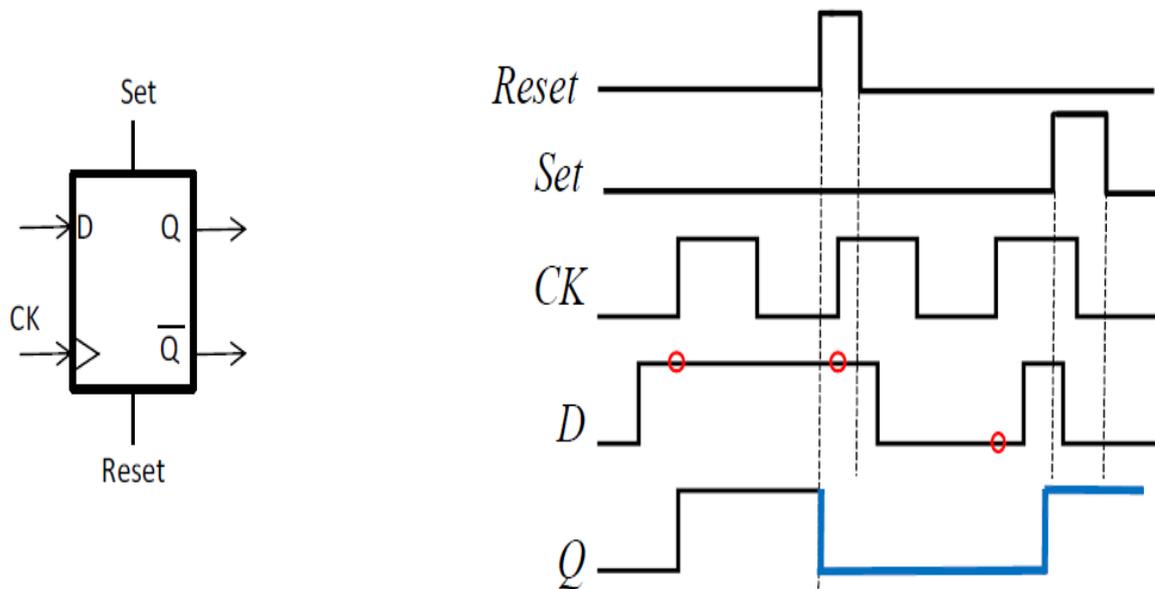


Figure. 4.15

4.2.7 Autres types de bascules : Bascule T

La bascule T est utile pour la conception de certains circuits séquentiels, comme nous aurons le loisir de le constater plus bas. C'est une bascule D à laquelle on ajoute une rétroaction depuis la sortie Q vers un XOR. La Figure 4.16 présente ce type de bascules.

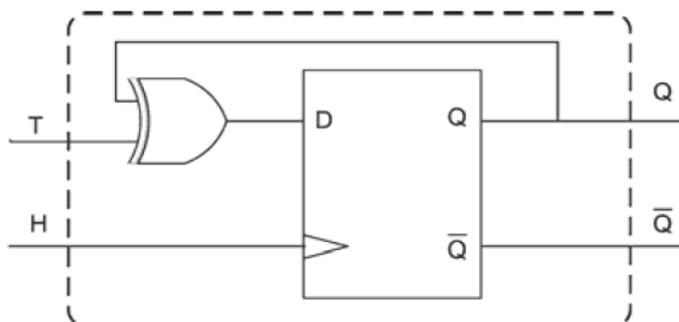


Figure. 4.16

La table de vérité associée à la bascule T, est reportée au tableau 4.7. Elle indique que l'entrée

H	T	Q^+
0	–	Q
1	–	Q
↑	0	Q
↑	1	\bar{Q}

Tableau. 4.7

T permet de changer ou conserver la valeur mémorisée par la bascule. Si $T = 0$, la valeur Q est maintenue au front montant. Si l'entrée $T = 1$, la valeur de Q est **inversée** au front montant.

4.2.8 Autres : Bascule JK

La Figure 4.17 présente le circuit de la bascule JK.

C'est une bascule relativement complexe. Contrairement aux bascules D et T, la bascule JK possède deux entrées. La table de vérité reportée au tableau 4.8 résume le comportement correspondant à la bascule JK. Son fonctionnement est à mi-chemin entre celui de la bistable SR et de la bascule T. La bascule JK possède un comportement proche de celui de la bascule T en cela qu'elle permet de conserver ou d'inverser la valeur de Q. Pour ce faire, il faut garder les deux entrées JK à la même valeur. Si $JK = 00$, la valeur de Q est conservée au front montant. Si les entrées $JK = 11$, l'entrée est inversée au front montant.

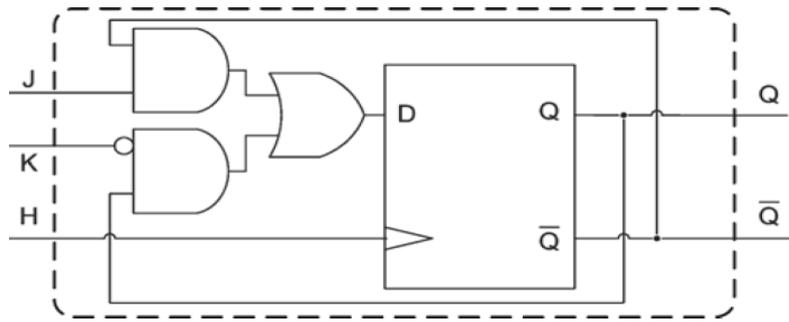


Figure. 4.17

H	J	K	Q^+
0	—	—	Q
1	—	—	Q
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	\overline{Q}

Tableau. 4.8

La bascule JK possède un comportement proche de celui de la bistable SR . Si on fixe les entrées $JK = 10$, on peut écrire 1 dans Q au front montant. À l'inverse, si les entrées $JK = 01$, on peut écrire 0 dans Q .

4.3 Fonctions séquentielles STANDARD

Les circuits numériques modernes sont obtenus en cascade plusieurs bascules. Nous allons tâcher d'étudier deux circuits logiques séquentiels importants :

- ▽ Les compteurs / décompteurs / diviseurs de fréquence ;
- ▽ Les registres de différents types.

4.3.1 Compteurs ou décompteurs

La fonction de comptage élémentaire consiste simplement à passer d'une valeur entière N à la valeur $N + 1$ (ou $N - 1$ s'il s'agit d'un décompteur), quand un ordre de comptage est actif. Le nombre N est codé sur m chiffres binaires. Comme m est fini (4, 8 ou 16 sont des valeurs courantes), l'ensemble des valeurs possibles pour le contenu du compteur est fini.

Quand N est égal au plus grand nombre possible, N_{max} , la valeur suivante est généralement 0.

Un compteur réel est donc toujours un compteur modulo $N_{max} + 1$; si $N_{max} = 2^m - 1$ il s'agit d'un compteur binaire. Notons en fin que les compteurs sont de deux types :

- ▽ Les compteurs **asynchrones** : pour lesquels les bascules commutent en asynchronisme avec le signal d'horloge **CLK**. Autrement dit, ce type de circuit numérique n'est pas commandé de l'extérieur ;
- △ Les compteurs **synchrones** : toutes les bascules reçoivent en même temps le signal d'une seule horloge **CLK**. Ceci permet une commutation au même instant des sorties de toutes ces bascules (Synchronisation).

La présence des états **transitoires** est le grand désavantage du 1^{er} type par rapport au 2nd.

Dans ce cours, une importance particulière est donnée au 2nd type de compteurs.

4.3.1.1 Compteur Modulo 2

C'est le compteur le plus simple (1 seul étage). La seule sortie Q_0 bascule à chaque front d'horloge **CLK** ($2^{nbreBascules} = 1 = 2$ ÉTATS). Le chronogramme de la Figure 4.18 décrit ce compteur.

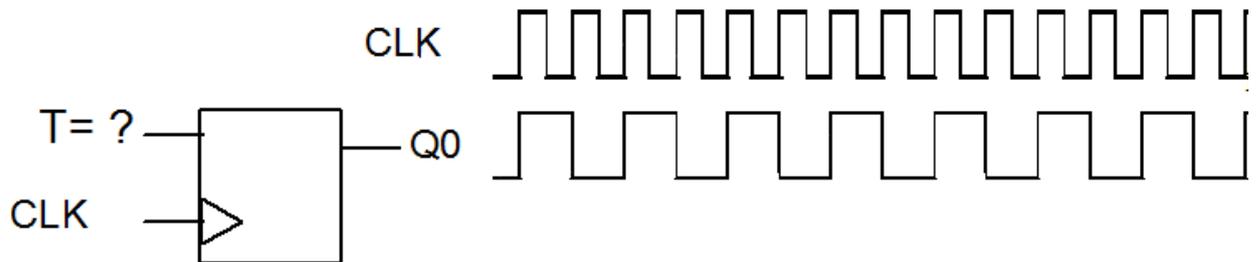


Figure. 4.18

On peut remarquer que la sortie Q_0 change à tous les coups d'horloge. Sachant que sur une bascule T (ou JK), il y a une inversion de la sortie pour $T = 1$ ($JK = 11$), on peut en déduire l'entrée de la bascule qui réalise ce compteur : $T = 1$.

- **Étape 0 de la procédure** : Graphe d'états montré dans la Figure 4.19.
- **Étape 1 de la procédure** : Table de transition d'états présentée dans le tableau 4.9.

État Présent	État Futur	CNT (Sortie)
C_0	C_1	" 0 "
C_1	C_0	" 1 "

Tableau. 4.9

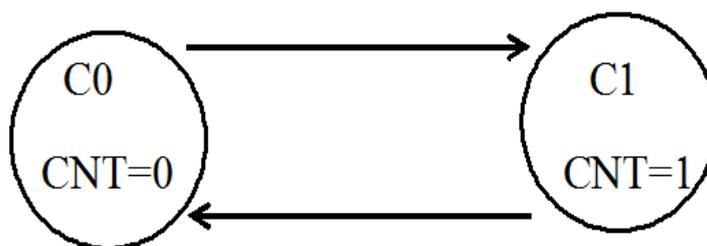


Figure. 4.19

- **Étape 2 de la procédure** : Encodage des états présentés dans le tableau 4.10 (C_0 et C_1 ont pour codes respectifs " 0 " et " 1 ").

État Présent	Etat Futur		Sortie
	$Q_{0,P}$	$Q_{0,F}$	CNT
C_0	0	C_1 1	0
C_1	1	C_0 0	1

Tableau. 4.10

- **Étape 3 de la procédure** : Détermination des entrées à l'instant t, des bascules (T, D, ou JK) pour conditionner l'évolution futur de l'état du système ; le tableau 4.11 montre cette étape.

État Présent	Etat Futur		Entrée Bascule T	Entrée Bascule D	Entrées Bascule JK	
	$Q_{0,P}$	$Q_{0,F}$	$T_{0,P}$	$D_{0,P}$	$J_{0,P}$	$K_{0,P}$
C_0	0	C_1 1	1	1	1	Φ
C_1	1	C_0 0	1	0	Φ	1

Tableau. 4.11

- **Étape 4 de la procédure** : Détermination des équations d'entrées de bascules (T, D, ou JK) : On tire facilement :

$$T = 1 ; D = \overline{Q_0} ; J K = 1 1$$

- **Étape 5** : Schéma du circuit, montré sur la Figure 4.20.

Exercice 1 : Que fait le circuit de la figure 4.21 (les bascules sont de type D) ?.

4.3.1.2 Compteurs Modulo 8, et modulo 16

Exercice 2 : Étendre le résultat obtenu dans l'exercice 1 afin de concevoir, en utilisant des bascules T, deux compteurs synchrones, Modulo 8 et 16.

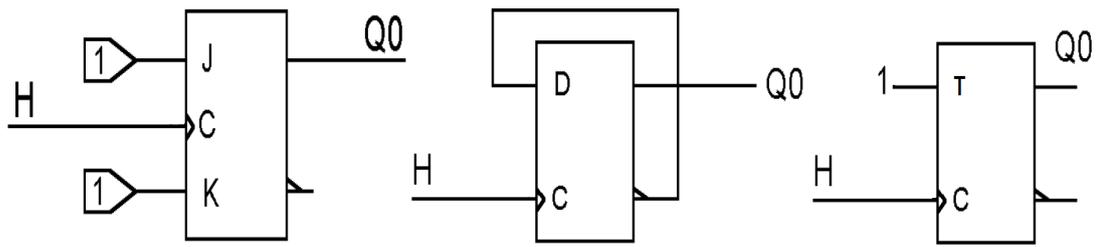


Figure. 4.20

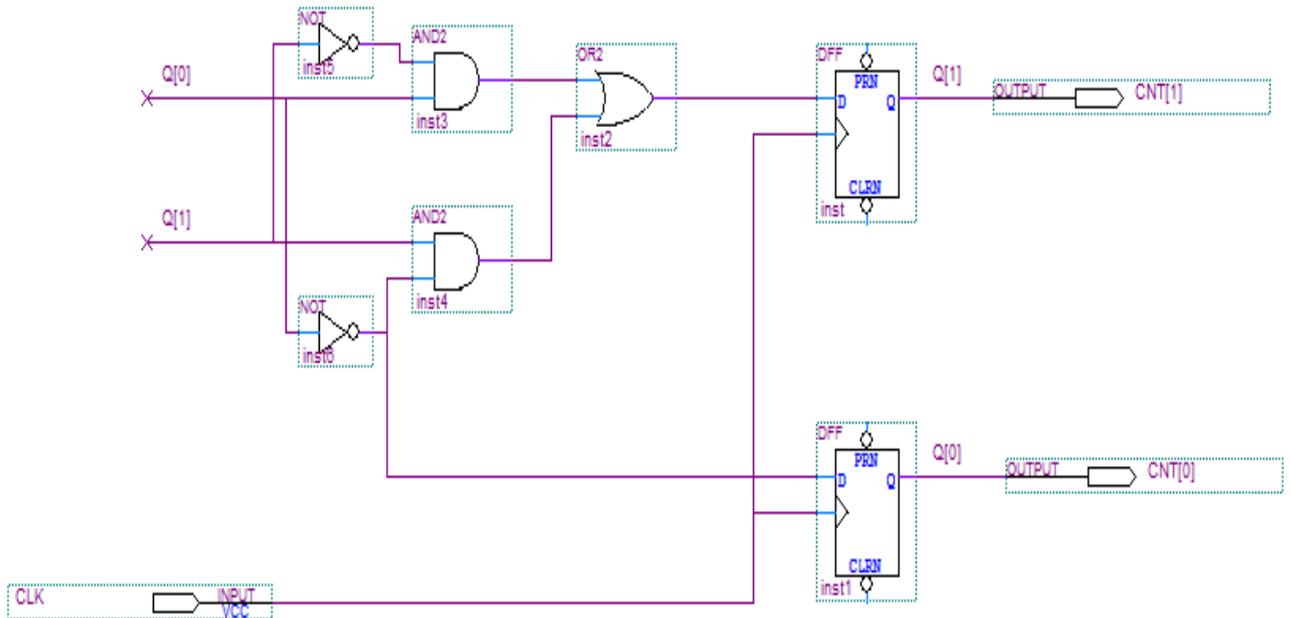


Figure. 4.21

Répondre en suivant la même démarche précédente (Simplifier le cas échéant, les équations d'entrées de bascules en utilisant les tableaux de Karnaugh).

Exercice 3 : Refaire l'exercice 2 en utilisant des bascules D.

Remarques :

- ▽ Les décompteurs sont obtenus en connectant les sorties sur $\overline{Q_i}$;
- ▽ Les compteurs modulo $\neq 2^m$ (p.ex. comteur décimal) se synthétisent en adaptant les équations des entrées des bascules avec la séquence des sorties en question.
- ▽ Un compteur joue le rôle de diviseur de la fréquence du signal d'horloge CLK. En effet, la fréquence F_l du signal de sortie Q_l d'un compteur est :

$$F_l = \frac{F_{CLK}}{2^l}$$

4.3.1.3 Autres types de compteurs

Il existe d'autres types de compteurs, nous pouvons citer :

- ▽ Les compteurs décimaux (code **DCB**) ;
- ▽ Les compteurs assurant des codes (**un seul actif : One hot**) : les codes correspondants sont les plus dilués ;
- ▽ Les **compteurs programmables** disposant d'entrées de commandes (Signaux d'entrée : *clk; reset; up; load, ...*), qui leur donnent bien d'autres fonctions que l'incrément d'un entier.
- ▽ ...

4.3.2 Registres

Un registre à décalage élémentaire est organisé de telle façon que l'entrée d'une bascule est connectée à la sortie de l'une de ses voisines, conformément à la figure 4.22.

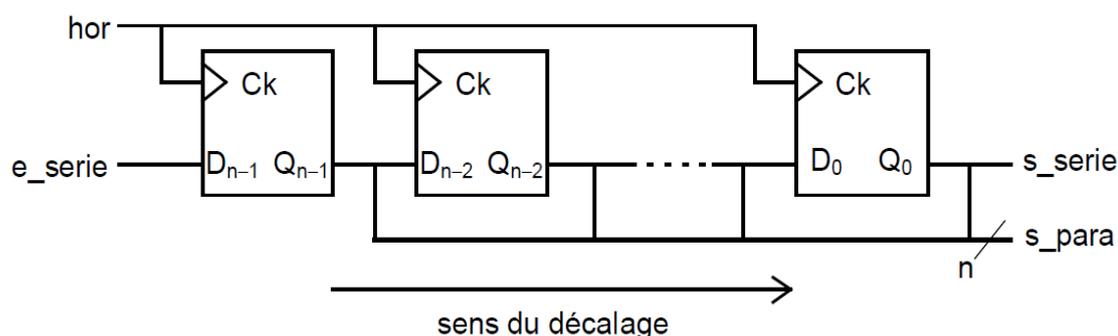


Figure. 4.22

Suivant le sens du décalage, lié à la façon dont on dessine le schéma, on parle de décalage à gauche ou à droite. La figure 4.23 montre un exemple de registre de décalage à gauche.

La figure 4.24 présente le rapport de simulation correspondant.

Si le contenu du registre représente un nombre codé en binaire, nous retrouvons que la fonction décalage est intimement liée aux opérations de multiplication et de division par 2.

La figure 4.25 montre un registre de chargement. Le registre *reg - mem - 4b* étant de mémorisation ; il est constitué de 4 bascules permettant la mémorisation d'un mot binaire (4 bits). On l'appelle aussi registre à entrées parallèles.

La figure 4.26 présente le rapport de simulation correspondant.

La figure 4.27 présente un modèle plus élaboré, formé d'une association de bascules permet-

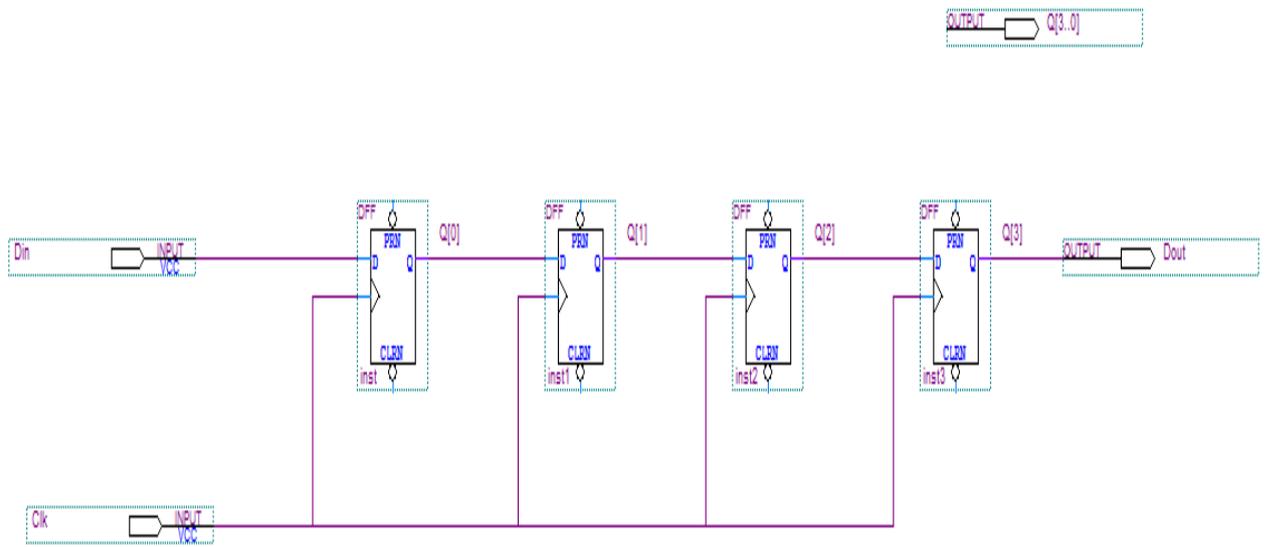


Figure. 4.23



Figure. 4.24

tant quatre modes de fonctionnement (Maintien, Décalage à droite ou à gauche, chargement parallèle). On y trouve une entrée de données en série, N entrées de chargement en parallèle et N sorties ; la N ème sortie peut servir de sortie série.

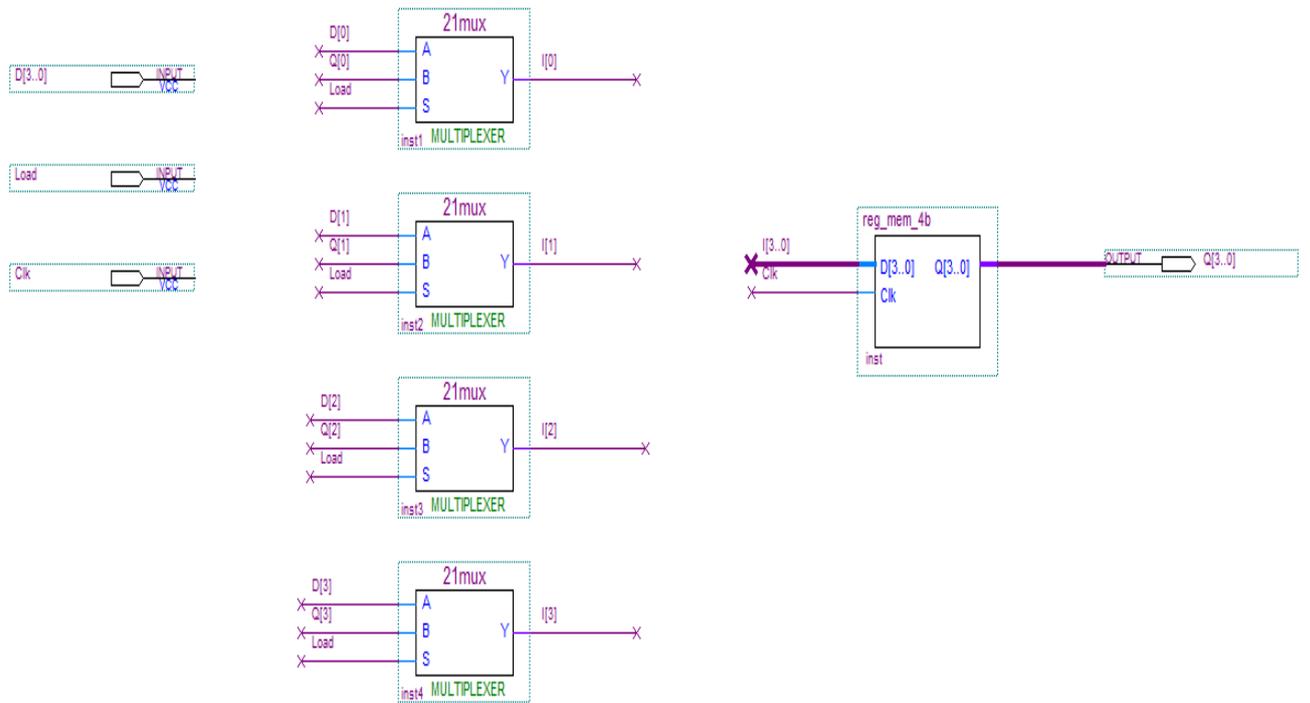


Figure. 4.25



Figure. 4.26

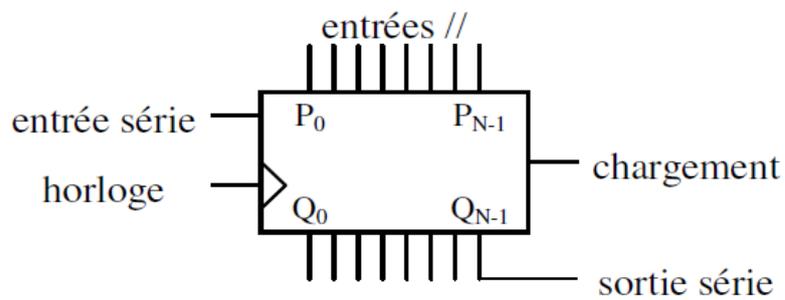


Figure. 4.27

