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

Ecole Supérieure en Sciences
Appliquées D'Alger (ESSAA)

DEPARTEMENT DE SECOND CYCLE – GENIE ELECTRIQUE
OPTION : TRACTION ELECTRIQUE

POLYCOPIE PEDAGOGIQUE

Pour l'obtention de

L'HABILITATION UNIVERSITAIRE

Spécialité : Electronique

Rédigé par

Dr. Mahmoud ARBID

Intitulé

COURS DE SYSTEMES NUMERIQUES 01

Année universitaire 2019/2020


بسم الله الرحمن الرحيم

© Copyright by Mahmoud ARBID 2019
All Rights Reserved

Table des matières

Page

Avant propos

Ce cours intitulé *Systèmes Numériques* 01, est conçu aux étudiants de deuxième année d'ingéniorat en électronique, électrotechnique et automatique. Et peut être utilisé par les étudiant ingénieurs en informatique. Le module et décortiqué en deux partie, à savoir la logique combinatoire et celle séquentielle, dont chaque partie est subdivisée en trois chapitres.

Le chapitre 1 est consacré aux systèmes logiques combinatoires, en commençant par la représentation numérique de l'information, le comportement des circuits combinatoires et la simplification des fonctions logiques, ainsi qu'une présentation de quelques systèmes séquentiels complexes.

Le chapitre 2 est dédié au comportement des circuits séquentiels, à savoir les bascules, les registres à décalage et les compteurs. Et on terminera par la synthèse des systèmes séquentiels asynchrones.

Le module en question comporte une charge hebdomadaire de 15 semaines, telle que, 01 cours de $01H30$, 01 TD de $01H30$ et 01 TP de $03H00$. Son évaluation finale sera effectuée de la manière suivante :

- Devoirs maison, Assiduité, et interrogation sur 30% ;
- Travaux pratiques sur 30% ;
- Examen final, noté sur 40%.

Première partie

Logique combinatoire

Chapitre 1

Représentation Numérique de l'information

1.1 Représentation des nombres

La représentation universelle utilisée dans notre vie quotidienne est la numérotation dite décimale. Elle contient des nombres constitués d'une série de chiffres et comprenant parfois un point ou une virgule.

Et puisque l'ordinateur est devenu une machine indispensable dans nos jours, alors il est parfois pratique d'utiliser d'autres bases que la base 10 ; les plus utilisées sont les bases 2, 8 et 16 qui conduisent à manipuler des nombres représentés en binaire, en octal ou en hexadécimal.

Dans un système de base k, il faudrait k symboles différents pour exprimer les chiffres de 0 à k-1.

Les nombres exprimés en octal constitués de huit chiffres : 0 1 2 3 4 5 6 7.

Pour exprimer un nombre en hexadécimal, on utilise en plus des dix chiffres décimaux les six premières lettres de l'alphabet romain. Donc, en hexadécimal les nombres sont exprimés avec les symboles : 0 1 2 3 4 5 6 7 8 9 A B C D E F.

1.2 Systèmes de numérations

La numération est une manière qui consiste à représenter un nombre A dans un système de base quelconque B : Décimal, Hexadécimal, Octal, Binaire, ...

$178{,}25_{10} = 1.10^2 + 7.10^1 + 8.10^0 + 2.10^{-1} + 5.10^{-2}$

$10111{,}01_2 = 1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2}$, d'où :

$N_B = a_{n-1}.B^{n-1} + a_{n-2}.B^{n-2} + ... + a_0.B^0 + a_{-1}.B^{-1} + ...$

Sachant que B représente la base, a_{n-1} le digit et B^{n-1} la puissance.

1.3 Conversion d'un système de numération

1.3.1 Conversion de la base B vers 10

Afin de convertir un nombre en base B vers une autre base 10, on multiplie chaque chiffre du nombre représenté en base B par la puissance de B correspondante, puis effectuer la somme de ses produits. L'example suivant montre une conversion de la base 2 vers la base 10 :

$$\begin{aligned} 11011{,}101_2 &= 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 + 1.2^{-1} + 0.2^{-2} + 1.2^{-3} \\ &= 16 + 8 + 0 + 2 + 1 + 0{,}5 + 0 + 0{,}125 \\ &= 27{,}62510 \end{aligned}$$

1.3.2 Conversion de la base 10 vers B

Il existe deux méthodes pour convertir un nombre en base B vers une autre base 10 :

1.3.2.1 $1^{ère}$ Méthode (Soustraction successive)

Une $1^{ère}$ méthode consiste à soustraire la plus grande puissance de B d'une manière successive, avec B = 2 :

$$\left.\begin{aligned} 363_{10} &= 2^8 + 7 \\ 107 &= 2^6 + 43 \\ 43 &= 2^5 + 11 \\ 11 &= 2^3 + 3 \\ 3 &= 2^1 + 1 \\ 1 &= 2^0 \end{aligned}\right\} \quad \begin{aligned} 363_{10} &= 2^8 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0 \\ &= 1\,0\,1\,1\,0\,1\,0\,1\,1_2 \end{aligned}$$

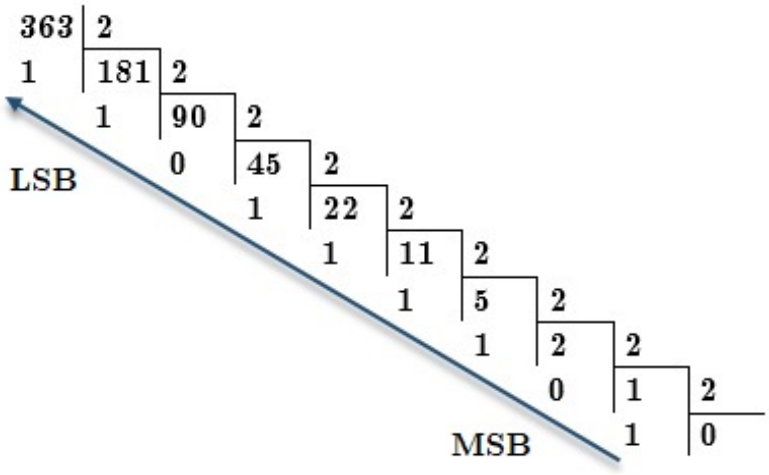
1.3.2.2 $2^{ème}$ Méthode (Division)

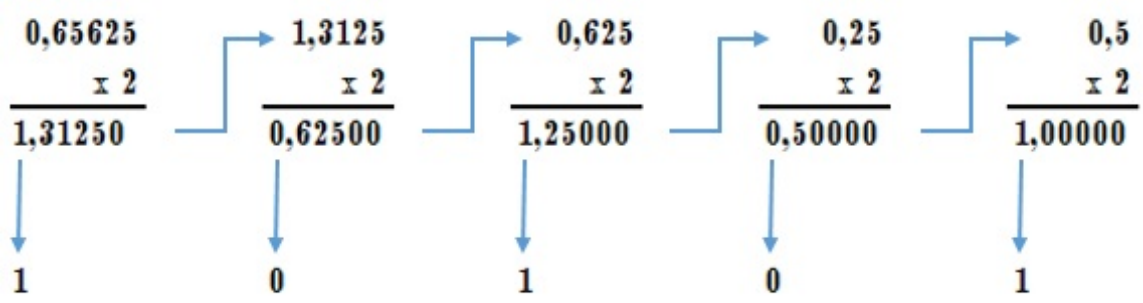
La $2^{ème}$ méthode consiste le nombre par la base B autant de fois que nécessaire pour obtenir le quotient nul, puis in écrit le reste dans l'ordre inverse où ils ont été obtenus, de la manière suivante : Il vient, $363_{10} = 101111011_2$.

NB : Le bit le plus à droite est appelé le bit le plus significatif ou LSB (Least Significant Bit), alors que celui le plus à gauche est appelé le bit le plus moins significatif ou MSB (Mosst Significant Bit).

Or, pour convertir un nombre décimal fractionnaire en base binaire, il suffit de multiplier le nombre en question par 2, et à chaque opération on garde le premier nombre après la virgule.

Exemple : $0{,}65625_{10} = 0{,}10101_2$





1.3.3 Généralisation de la conversion en puissance de 2

1.3.3.1 Conversion de la base 2^n vers la base n

La méthode consiste à convertir chaque chiffre du nombre 2^n en base 2, puis on juxtapose les résultats.

Exemple : $475_8 = ?_2$. B = 8 = 23 → 3 bits,

4	7	5
100	111	101

Donc, $475_8 = 100\ 111\ 101_2$.

1.3.3.2 Conversion de la base 2 vers la base 2^n

Dans la conversion inverse, on découpe le nombre en tranches de n éléments binaires chacun.

Exemple : $11\ 000\ 111\ 011\ 011\ 001_2 = ?_8$. B = 8 = 23 → 3 bits,

11	000	111	011	011	001
3	0	7	3	3	1

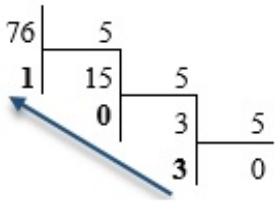
Donc, $11\ 000\ 111\ 011\ 011\ 001_2 = 307331_8$.

1.3.3.3 Conversion de la base 'i' vers la base 'j'

∘ Si 'i' et 'j' sont tous les deux des puissances de 2, on utilise la base 2 comme base de relais : $i \rightarrow j$

Exemple : $E8A_{16} = ?_8$,

E	8	A
1110	1000	1010



$E8A_{16} = 111\ 010\ 001\ 010_2 = ?_8$,

111	010	001	010
7	2	1	2

D'où, $E8A_{16} = 7212_8$.

∘ Si 'i' et 'j' ne sont pas de la puissance de 2, dans ce cas on utilise la base 10 comme base de relais.

Exemple : $136_7 = ?_5$

$136_7 = 1.7^2 + 3.7^1 + 6.7^0 = 76_{10}$.

$136_7 = 76_{10} = ?_5$.

D'où, $136_7 = 301_5$.

1.3.4 L'arithmétique binaire

Quand un calculateur effectue une opération arithmétique, deus contraintes interviennent :

1. Les circuits arithmétiques travaillent sur des nombres qui ont toujours le même format (longueur). Prenant l'exemple d'un codage de 8 bits, où le nombre **10111** doit être complété par des zéros : **000**10111 ;
2. Les circuits manipulent seulement deux nombres à la fois. Donc, pour obtenir $S = X + Y + Z$, on effectue : $S = X + Y$ puis $S = S + Z$.

1.3.4.1 Les opérations

Les mêmes règles de calculs s'appliquent dans un système de numération. L'arithmétique binaire ressemble à celle des nombres décimaux.

a	b	*Add.*	R_+	*Soust.*	R_-	*Prod.*	R_x
0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	0
1	0	1	0	1	0	0	0
1	1	0	1	0	0	1	0

1.4 Codage des entiers

Le code est une correspondance entre un ensemble des objets et un autre ensemble des symboles. Ce code traduit le langage humain en langage machine.

1.4.1 Les codes pondérés

Ce sont ceux qu'on utilise dans les opérations arithmétiques, on peut citer : le code binaire et ses dérivées (Octal, Hexadécimal, ...) et le code DCB (Décimal Codé Bonaire).

1.4.1.1 Codage binaire pur

C'est une représentation d'un nombre dans le système de numération binaire. D'où avec 8 bits on peut coder tous les entiers positifs dans un intervalle compris entre 0000 0000 et 1111 1111, c'est-à-dire entre 0 et 255. Donc, avec n bits on peut coder tous les entiers entre 0 et 2^n-1. Sachant que cette représentation n'est pas valable pour les nombres négatifs.

1.4.1.2 Codage DCB (Décimal Codé Binaire)

Cette représentation permet de coder chaque chiffre d'un nombre décimal sur 4 bits, comme suit :

1	0	4	9	7
0001	0000	0100	1001	0111

1.4.1.3 Codage binaire signé

Afin d'utiliser un additionneur comme soustracteur, il est impératif de représenter convenablement les nombres négatifs de la manière suivante :

$$A - B \ = A + (-B)$$

Dans une telle représentation, le bit le plus à gauche indique le signe. On utilise 0 pour indiquer un nombre positif et 1 pour nombre négatif. Dans ce cas et pour n bits, on ne peut représenter que les entiers compris entre –(2^n-1) et (2^n-1). Et on note que la représentation du 0 n'est pas unique.

Exemple : 11111111_2 représent -127 et 01101100_2 représente +108.

Soit à calculer -6+7

	0000 0111	*(+7)*
+	1000 0110	*(-6)*
=	1000 1101	*(-13)*

On constate que l'addition binaire n'est pas souvent valable pour les nombres négatifs en utilisant la représentation binaire signée si on ne prend pas en considération à part le bit signe.

1.4.1.4 Codage en complément à 1 (Complément restreint)

Les entiers positifs sont représentés comme celle du binaire signé. On obtient le complément à 1 (C1) d'un nombre entier x en inversant tous les bits de la représentation de x, c'est-à-dire remplacer les 0 par des 1 et inversement.

Exemple : $x = 010011_2 \rightarrow C1(x) = 101100_2$.

Si n est le nombre d'éléments binaires de x, alors :

$$x + C1(x) = 2^n - 1$$

$$\begin{array}{rrr} & x = & 1101_2 \\ + & C1(x) = & 0010_2 \\ \hline = & & 1111_2 \end{array} \text{, d'où}$$

$$-x = C1(x) + 1 - 2^n$$

Ce qui montre que le $C1(x) + 1$ représente le nombre négatif de x, à condition de ne pas en tenir compte du chiffre de poids 2^n, comme le montre l'exemple suivant pour $n = 4$:

$$\begin{array}{rr} x = +13 & ....000\ 1101_2 \\ C1(x) +1 & ....111\ 0011_2 \\ \hline & ....000\ 0000_2 \end{array}$$

1.4.1.5 Codage en complément à 2 (Complément vrai)

Les nombres positifs sont représentés de la même manière que celle du binaire. Afin de représenter les nombres négatifs, on part du complément à 1 puis on ajoute 1, appelé complément à 2 (C2).

$$-x = C1(x) + 1 = C2(x)$$

D'où avec n bits on peut coder les nombres compris entre -2^{n-1} et $(2^{n-1} - 1)$.

Exemple :

x=13_{10}=1101_2 → C2(x)=0010_2+1=0011_2. Or, C2(C2(x))=1100_2+1=1101_2=13_{10}.

- La représentation du nombre 0 est unique.
- La somme de 2 nombres de signes opposés est toujours correcte.
- La somme de 2 nombres de même signe n'est juste que si le résultat est de même signe.

Exemple :

	11111100	*(-4)*			01111110	*(+126)*			11000000	*(-64)*
+	00000110	*(+6)*		+	00000010	*(+2)*		+	10111111	*(-65)*
1 :	00000010	*(+2)*			10000000	*(-128)*		1 :	01111111	*(+127)*
	Avec retenue			Sans	retenue mais			Avec	retenue mais	
				avec	débordement			avec	débordement	

D'après les exemples ci-dessus, on peut constater que la retenue (Carry) n'est pas suffisante pour indiquer le débordement de calcul (Overflow). Ce dernier n peut être représenté correctement sur un nombre n de bits.

1.4.2 Les codes non pondérés

1.4.2.1 Codage excédent 3

On obtient ce code en décalant vers le haut de trois lignes le code DCB (voir la table ci-bas).

1.4.2.2 Codage binaire réfléchi (GRAY)

Dans ce code chaque deux thermes successifs ne diffèrent que par un seul bit, on dit que les termes sont adjacents.

Nombre	*Code DCB*	*Excédent 3*
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Nombre	*Code Binaire*	*Code GRAY*
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

1.5 Représentation des nombres réels

Les nombres réels sont utilisés dans les systèmes numériques car ils permettent une variété de calculs. Ils peuvent être représentés avec une virgule fixe ou flottante. La représentation en virgule fixe permet de coder une plage fixe de nombres et d'effectuer des calculs rapides, tandis que la codification de nombres de différents ordres de grandeur est plus facile avec une représentation en virgule flottante.

1.5.1 Virgule fixe

L'idée de la virgule fixe serait d'utiliser un entier et de déplacer ainsi sa virgule d'un nombre fixe de position, ce qui revient à multiplier la base par une puissance bien définie. Par exemple en base 10 avec un décalage de 3, 123.456 serait représenté par 123456 et 1.23456 par 1234. D'où on passe d'un entier à un réel par multiplication de 10^{-3}.

1.5.2 Virgule flottante

L'inconvénient de la représentation en virgule fixe est qu'on ne peut représenter des réels grands ou petits, comme par exemple la constante de Planck ($\hbar \approx 1,0545718 \times 10^{-34}\ J.s$), le nombre d'Avogadro ($N_A = 6,02214076 \times 10^{23}\ mol^{-1}$), ... D'où l'idée de faire varier la position de la virgule, d'où le nom de représentation à virgule flottante.

Cette représentation peut être considérée comme une notation scientifique pour les systèmes numériques. Un certain nombre de représentations en virgule flottante ont été proposées pour répondre aux exigences de nombreuses applications. Un nombre décimal N peut être quantifié et exprimé sous forme de virgule flottante comme suit :

$$N_{10} = (-1)^S.M.B^E$$

Où S est le bit de signe, M est la mantisse, B est la base et E est l'exposant. La mantisse est généralement normalisée et correspond à un nombre commençant par un chiffre différent de zéro, comme dans le cas des représentations numériques suivantes :

Nombre	*Représentation*
-1234.57_{10}	-1.23457×10^3
0.0000071539_{10}	$+7.1539 \times 10^{-6}$
100010100_2	1.00010100×2^8

Chapitre 2

Algèbre de Boole et fonctions logiques

2.1 Définition

La mise en œuvre pratique de systèmes logiques est, la plupart du temps, basée sur la technologie des circuits logiques électroniques. L'algèbre de base ressemble sous certains aspects à l'algèbre classique. C'est un ensemble de variables à deux états de valeur de vérité *vrai* (1) ou $faux$ (0), correspondant à successivement à deux valeurs de tension $0v$ et $5v$ et munie d'un nombre fini d'opérateurs : Non, Et, Ou, ...

La manipulation de ses variables dites booléennes en fonction de ses opérateurs donne aussi une des fonctions booléennes.

Exemple : Un circuit électronique ouvert est une variable booléenne *vraie* (1) et vice versa.

2.2 Opérations et fonctions de base de l'algèbre de Boole

2.2.1 Fonction à une seule variable

L'inverseur est un opérateur qui ne porte que sur une seule variable d'entrée, $S = \bar{A}$.

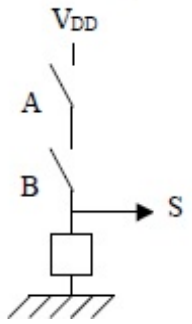
Table de vérité		Montage	Symbole traditionnel	Symbole normalisé
A	$S = \overline{A}$	V_{DD}, S, A		1
0	1			
1	0			

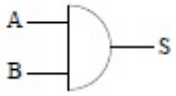
2.2.2 Fonction à deux variables

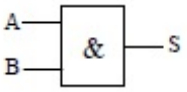
2.2.2.1 Opérateur ET (AND)

Si on prend A et B comme des variables d'entrée, alors $S = A.B$. S est vraie si A et B sont vraies.

Table de vérité			Montage	Symbole traditionnel	Symbole normalisé
A	B	S = A.B			
0	0	0			
0	1	0			
1	0	0			
1	1	1			



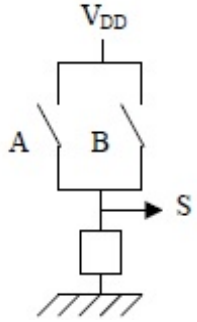


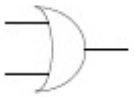


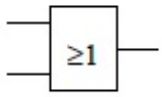
2.2.2.2 Opérateur OU (OR)

Si A et B sont les variables d'entrée, alors $S = A + B$. S est vraie si A ou B sont vraies.

Table de vérité			Montage	Symbole traditionnel	Symbole normalisé
A	B	S = A+B			
0	0	0			
0	1	1			
1	0	1			
1	1	1			



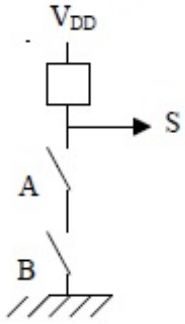


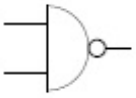


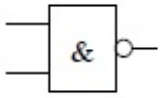
2.2.2.3 Opérateur Non ET (NAND)

Si A et B sont les variables d'entrée, alors $S = \overline{A.B}$. S est fausse si A et B sont vraies. Le $NAND$ est l'inverseur du AND.

Table de vérité			Montage	Symbole traditionnel	Symbole normalisé
A	B	S = $\overline{A.B}$			
0	0	1			
0	1	1			
1	0	1			
1	1	0			



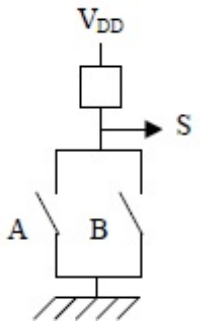


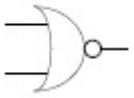


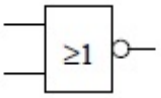
2.2.2.4 Opérateur Non OU (NOR)

Si A et B sont les variables d'entrée, alors $S = \overline{A + B}$. S est fausse si A ou B sont vraies. Le NOR est l'inverseur du OR.

Table de vérité			Montage	Symbole traditionnel	Symbole normalisé
A	B	$S = \overline{A+B}$			
0	0	1			
0	1	0			
1	0	0			
1	1	0			



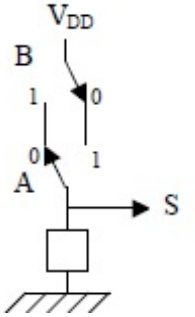


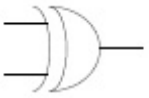


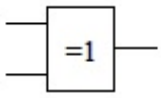
2.2.2.5 Opérateur OU Exclusif (XOR)

Cet opérateur n'est pas un opérateur de base car il peut être réalisé avec les portes décrites ci-dessus. Si A et B sont les variables d'entrée, alors $S = A \oplus B = A\overline{B} + \overline{A}B$. S est vraie si $A \neq B$.

Table de vérité			Montage	Symbole traditionnel	Symbole normalisé
A	B	$S = A \oplus B$			
0	0	0			
0	1	1			
1	0	1			
1	1	0			



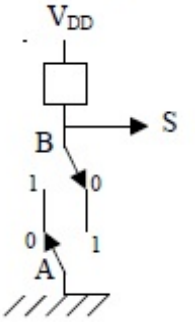


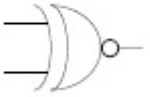


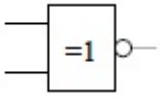
2.2.2.6 Opérateur Non OU Exclusif (XNOR)

Si A et B sont les variables d'entrée, alors $S = \overline{A \oplus B} = AB + \overline{A}\overline{B}$. S est vraie si $A = B$.

Table de vérité			Montage	Symbole traditionnel	Symbole normalisé
A	B	$S = \overline{A \oplus B}$			
0	0	1			
0	1	0			
1	0	0			
1	1	1			



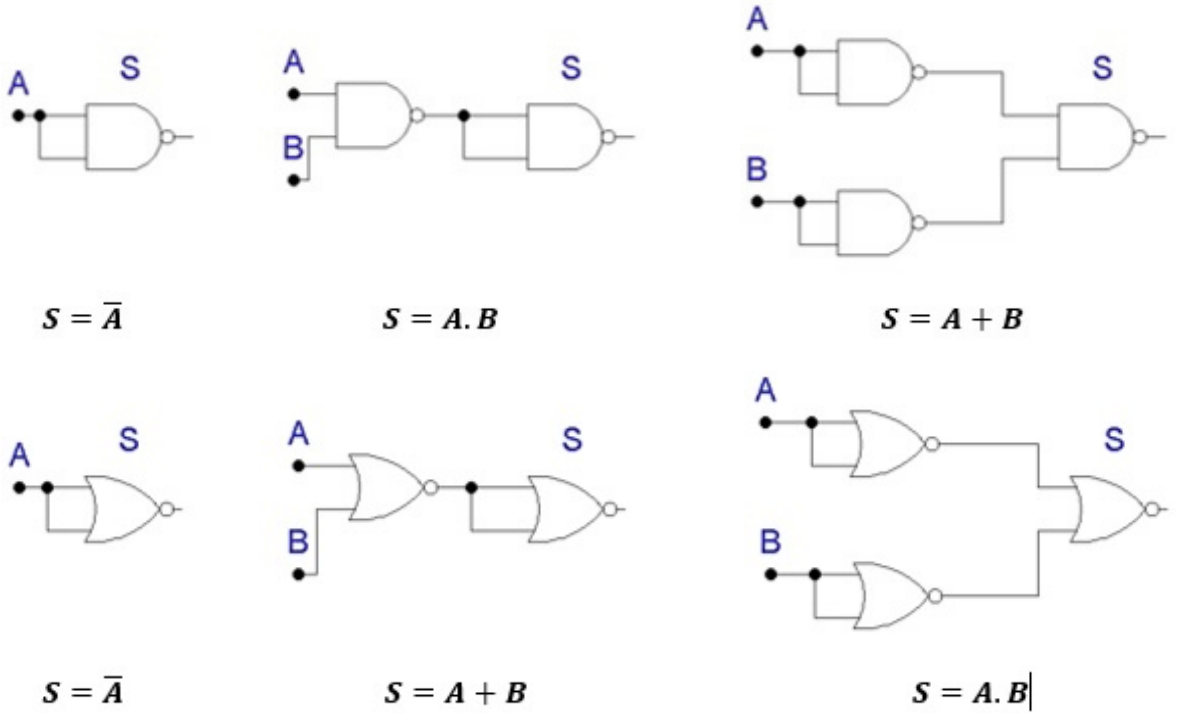




2.2.2.7 Les portes universelles

Le $NAND$ et le NOR sont toutes les deux des portes universelles car elles permettent de réaliser n'importe quel opérateur logique élémentaires.

Le même type de montage peut être réalisé avec des portes NOR.



2.3 Axiomes ou lois fondamentales de l'algèbre de Boole

Les lois fondamentales de l'algèbre de BOOLE pourraient être vérifiées avec des tables de vérités et en testant toutes les possibilités. Les lois sont les suivantes :

Commutativité	$A.B = B.A$
Associativié	$A.(B.C) = (A.B).C$ $A + (B + C) = (A + B) + C$
Distributivité	$A.(B + C) = A.B + A.C$ $A + (B.C) = (A + B).(A + C)$
Idempotence	$A.A = 1$ $A + A = A$
Complémentarité	$A.\overline{A} = 0$ $A + \overline{A} = 1$
Identités remarquables	$1.A = A$ $1 + A = 1$ $0.A = 0$ $0 + A = A$

2.4 Relations de base dans l'algèbre de Boole

1- $x.y + x.\overline{y} = x$ 1°- $(x + y).(x + \overline{y}) = x$

Preuve : $x.y + x.\overline{y} = x.(y + \overline{y}) = x.1 = x$

2- $x + x.y = x$ 2°- $x.(x + y) = x$

Preuve : $x + x.y = x.(1 + y) = x.1 = x$

3- $x + \overline{x}.y = x + y$ 3°- $x.(\overline{x} + y) = x.y$

Preuve : $x + \overline{x}.y = x + x.y + \overline{x}.y = x + y.(x + \overline{x}) = x + y.1 = x + y$

2.5 Théorèmes de MORGAN

Théorème 1 (01) *La négation d'un produit de variables est égale à la somme des négations :*

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

Théorème 2 (02) *La négation d'une somme de variables est égale au produit des négations :*

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

2.6 Représentation des fonctions booléennes

2.6.1 Mintermes / Maxtermes

∘ On appelle Minterme, le produit logique P_i de toutes les variables d'entrée intervenant dans une forme directe ou complémentée.

∘ On appelle Maxterme, la somme logique S_i de toutes les variables d'entrée intervenant dans une forme directe ou complémentée.

Exemple :

$F_1 = \underbrace{abc}_{P_0} + a\overline{b}c + ab\overline{c}$ $\qquad F_2 = \underbrace{(a + b + c)}_{S_0}.(a + \overline{b} + c).(a + \overline{b} + \overline{c}).$

2.6.2 Formes canoniques

Soit une fonction F, la $1^{ère}$ forme canonique de cette fonction est la réunion OU des Mintermes destinés aux 1 logiques de la fonction :

$$F(x_n, x_{n-1}, ...x_0) = \bigcup_{i=0} P_i$$

La $2^{ème}$ forme canonique de la fonction F est L'intersection ET des Mintermes destinés aux 0 logiques de la fonction :

$$F(x_n, x_{n-1}, ...x_0) = \bigcap_{i=0} S_i$$

Exemple : Soit une fonction F en fonction de trois variables booléennes a, b et c. La fonction est vraie si et seulement si la somme de ses variables d'entrées est un nombre pair. La table de vérité de F est donnée comme suit :

a	**b**	**c**	**F**	$\mathbf{P}_i$	$\mathbf{S}_i$	$\overline{F}$
0	0	0	1	$\overline{a}\overline{b}\overline{c}$	$a+b+c$	0
0	0	1	0	$\overline{a}\overline{b}c$	$a+b+\overline{c}$	1
0	1	0	0	$\overline{a}b\overline{c}$	$a+\overline{b}+c$	1
0	1	1	1	$\overline{a}bc$	$a+\overline{b}+\overline{c}$	0
1	0	0	0	$a\overline{b}\overline{c}$	$\overline{a}+b+c$	1
1	0	1	1	$a\overline{b}c$	$\overline{a}+b+\overline{c}$	0
1	1	0	1	$ab\overline{c}$	$\overline{a}+\overline{b}+c$	0
1	1	1	0	abc	$\overline{a}+\overline{b}+\overline{c}$	1

D$'$après la table de vérité, la fonction F peur s'écrire sous la $1^{ère}$ forme canonique comme suit :

$F(a,b,c) = \overline{a}\overline{b}\overline{c} + \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$

Afin de passer de la $1^{ère}$ forme à la $2^{ème}$ forme canonique, on prend en compte les combinaisons pour lesquelles F vaut 0.

$\overline{F}(a,b,c) = 1 = \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\overline{c} + abc$, donc :

$F(a,b,c) = 0 = \overline{\overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\overline{c} + abc}$, et d'après le théorème de MORGAN, il vient :

$F(a,b,c) = \overline{\overline{a}\overline{b}c}.\overline{\overline{a}b\overline{c}}.\overline{a\overline{b}\overline{c}}.\overline{abc}$, d'où

$F(a,b,c) = (a+b+\overline{c}).(a+\overline{b}+c).(\overline{a}+b+c).(\overline{a}+\overline{b}+\overline{c})$

2.6.3 La représentation numérique

Pour représenter numériquement la fonction F, on utilise directement l table de vérité :

$$F(a,b,c) = \sum(0,4,5,7)$$

$\sum$: représente le symbole de la somme logique. Les nombres entre parenthèse correspondent aux équivalents décimaux des combinaisons des variables d'entrée pour lesquelles la fonction vaut 1.

La représentation numérique équivalente est obtenue en prenant en compte les zéros de la fonction :

$$F(a,b,c) = \prod(1,2,3,6)$$

$\prod$: représente le symbole du produit logique. Les nombres entre parenthèse correspondent aux équivalents décimaux des combinaisons des variables d'entrée pour lesquelles la fonction vaut 0.

2.6.4 Représentation par le diagramme de Karnaugh

Cette représentation correspond à 2^n cases. Chaque case de la table est associée à une combinaison des variables d'entrée et elle comprend la valeur de la fonction F en fonction des variables d'entrée. Les diagrammes (tables) de Karnaugh pour n=2, 3 et 4 sont illustrés successivement comme suit :

a \ b	0	1
0		
1		

a \ bc	00	01	11	10
0				
1				

ab \ cd	00	01	11	10
00				
01				
11				
10				

Le diagramme de Karnaugh comprend une disposition particulière qui dit que deux cases voisines correspondent à des combinaisons adjacentes, d'où des combinaisons des diffèrent que par l'état d'une seule variable. Les cases du diagramme de pour lesquels la fonction vaux 0 sont laissées vides, sinon on met 1, comme illustré dans la table ci-bas.

Pour $n = 4$, les cases adjacentes de la combinaison 1000 sont 0000, 1100, 1010 et 1001.

Exemple : Soit $F(a, b, c, d) = \overline{a}bcd + a\overline{b}c + ab\overline{c} + b$

$\overline{a}bcd$: $abcd = 0111$ $\quad$ $ab\overline{c}$: $\forall d,\ abcd = 1001/1000$

$a\overline{b}c$: $\forall d,\ abcd = 1010/1011$ $\quad$ b : $\forall (a, c, d),\ ...$

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

2.6.5 Simplification des fonctions booléennes

Afin de simplifier une fonction à partir de la table de Karnaugh, on procède comme suit :

◦ Regrouper toutes les cases adjacentes pour lesquelles la fonction vaut 1 ;

◦ Le nombre de 1 dans les groupements doit être en puissance de 2 ;

◦ Eliminer les variables qui changent d'état et retenir ainsi celles qui conservent leur valeur lorsqu'on passe d'une case à l'autre.

Les doublets : Groupement de $2 = 2^1$.

Les quadruplets : Groupement de $4 = 2^2$.

Les octets : Groupement de $8 = 2^3$.

Remarque 3 *Une case peut être utilisée plusieurs fois, en raison de :* $a + a + a + ... + a = a$

a \ bc	00	01	11	10
0		1	1	
1				

a \ bc	00	01	11	10
0	1			1
1				

a \ bc	00	01	11	10
0				1
1				1

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

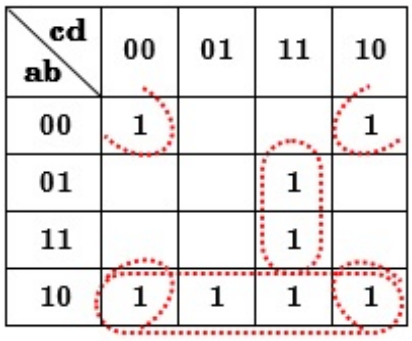
ab \ cd	00	01	11	10
00				
01	1	1	1	1
11				
10				

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

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

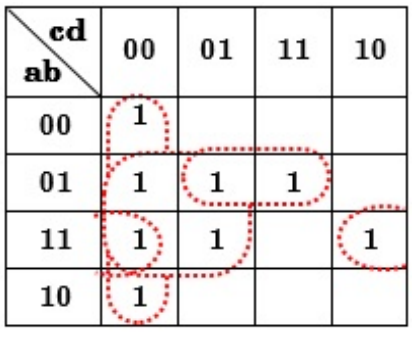
ab \ cd	00	01	11	10
00	1			1
01	1			1
11	1			1
10	1			1

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



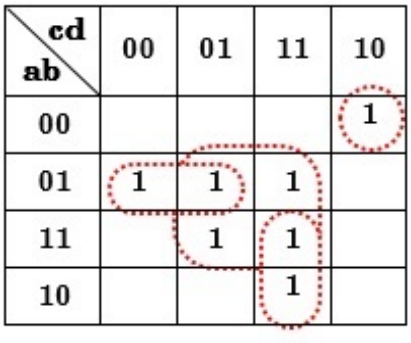
cd / ab	00	01	11	10
00	1			1
01			1	
11			1	
10	1	1	1	1

$F = a\bar{b} + \bar{b}\bar{d} + bcd$



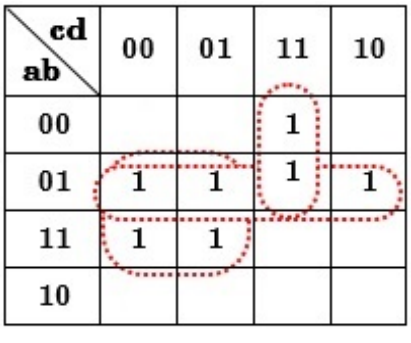
cd / ab	00	01	11	10
00	1			
01	1	1	1	
11	1	1		1
10	1			

$F = b\bar{c} + \bar{c}\bar{d} + \bar{a}bd + ab\bar{d}$



cd / ab	00	01	11	10
00				1
01	1	1	1	
11		1	1	
10			1	

$F = bd + \bar{a}b\bar{c} + acd + \bar{a}\bar{b}c\bar{d}$



cd / ab	00	01	11	10
00			1	
01	1	1	1	1
11	1	1		
10				

$F = b\bar{c} + \bar{a}b + \bar{a}cd$

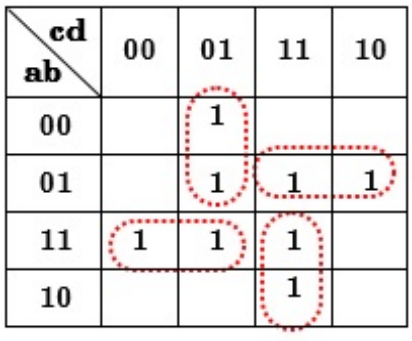
2.6.6 Les règles importantes pour lire un diagramme de Karnaugh

- Entourer tous les 1 impérativement ;
- Les cases vides (ou les 0) ne sont pas prises en considération ;
- Le 1 peut être entouré plusieurs fois.

Les exemples ci-bas montrent différents groupements d'une fonction $F(a, b, c, d)$:

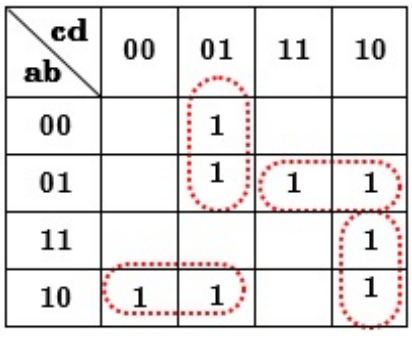
2.6.7 Les fonctions incomplètement définies

Dans ce type de fonctions qu'on verra plus en détail ultérieurement, nous allons placer des ϕ dans les cases correspondantes aux combinaisons interdites et les 1 dans les cases correspondantes à la fonction.



cd / ab	00	01	11	10
00		1		
01		1	1	1
11	1	1	1	
10			1	

$F = \bar{a}\bar{c}d + ab\bar{c} + acd + \bar{a}bc$



cd / ab	00	01	11	10
00		1		
01		1	1	1
11				1
10	1	1		1

$F = \bar{a}\bar{c}d + a\bar{b}\bar{c} + ac\bar{d} + \bar{a}bc$

Lors de la simplification, on va attribuer ϕ à la valeur 1 ou 0, afin d'obtenir les groupements les plus étendus. Voir l'exemple suivant :

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

$F = \bar{b}d + b\bar{c}$

2.6.8 Synthèse des systèmes combinatoires

La méthode à suivre afin d'extraire les équations d'un circuit combinatoire consiste à effectuer successivement les étapes suivantes :

∘ Extraire la table de vérité à partir du problème en question, c'est-à-dire :

– Déterminer le nombre n de variables d'entrée, d'où on aura 2^n combinaisons possibles ;

– Déterminer le nombre de variables de sorties ;

– Analyser chacune des combinaisons des de variables d'entrée.

∘ Simplifier séparément chaque fonction de sortie par le diagramme (table) de Karnaugh ;

∘ Etablir le schéma du circuit combinatoire à partir du système d'équation résultant.

Exemple :

Un distributeur de boisson permet de livrer aux consommateur de l'eau, la grenadine à l'eau et la menthe à l'eau. Mais il ne doit pas obtenir de : la grenadine seule, la menthe seule et la grenadine à la menthe.

La façade du distributeur comporte trois boutons :

$$e : \ L'Eau \qquad g : \ La\ Grenadine \qquad m : \ La\ Menthe$$

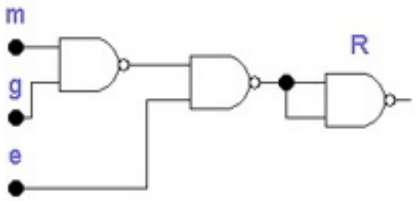
Déterminer la commande d'ouverture du robinet R du distributeur en déduire le logigramme correspondant en fontion seulement des portes $NAND$.

Solution :

∘ Table de vérité et simplification par le diagramme de Karnaugh (voir la figure ci-bas) :

e	m	g	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

e \ mg	00	01	11	10
00				
01	1	1		1



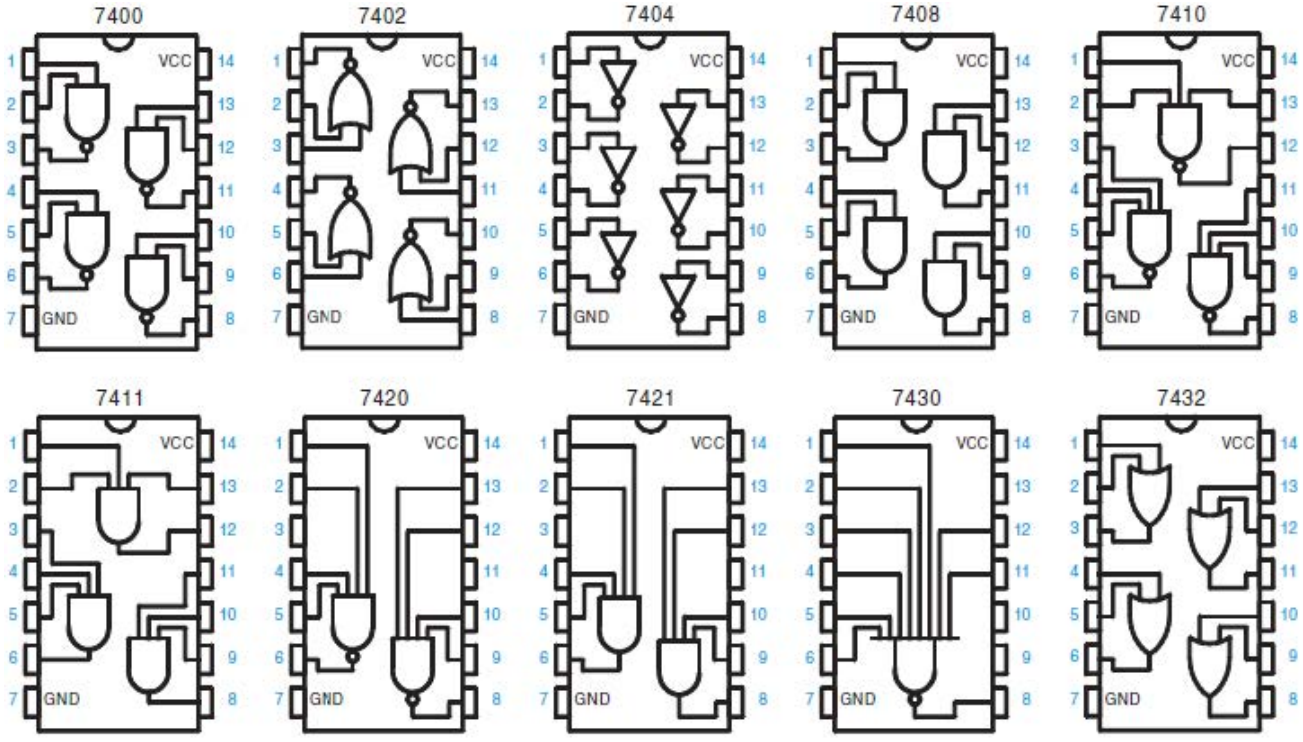
◦ Simplification par la table de Karnaugh :

$R = e.\overline{m} + e.\overline{g} \Longrightarrow R = e.(\overline{m} + \overline{g})$

◦ Logigramme en fonction des portes $NAND$:

$R = e.\overline{(\overline{m} + \overline{g})} = e.\overline{m.g} \Longrightarrow R = \overline{\overline{e.\overline{m.g}}}$

2.7 Quelques circuits intégrés logiques usuels



Chapitre 3

Circuits combinatoires usuels

3.1 Objectif

Dans certains cas, il est avantageux de concevoir des systèmes composés d'un ensemble de sous-systèmes identiques mis en cascade. Dans le présent chapitre nous examinerons les principaux circuits combinatoires usuels à savoir :

- Les circuits arithmétiques ;
- Les circuits de transcodage ;
- Les circuits d'aiguillage ;
- Systèmes séquentiels complexes.

3.2 Les circuits arithmétiques

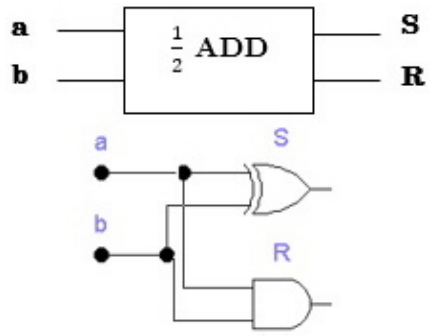
3.2.1 L'additionneur binaire

3.2.1.1 Le demi-additionneur ($\frac{1}{2}$ ADD)

Il représente une addition simple de deux bits a et b. On obtient donc une somme $S = a + b$ et une éventuelle retenue R. Et d'après la table de vérité illustrée ci-bas, on obtient les équations de S et R comme suit :

$$\begin{cases} S = a.\overline{b} + \overline{a}.b \\ R = a.b \end{cases} \Longrightarrow \begin{cases} S = a \oplus b \\ R = a.b \end{cases}$$

a	b	S	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



3.2.1.2 L'additionneur complet (ADD)

Considérons le cas de deux nombre binaires A et B de n bits chacun. La différence avec le demi-additionneur, c'est qu'il va y avoir une retenue précédente comme représenté ci-après :

$A = a_{n-1}a_{n-2}...a_i...a_1a_0$

$B = b_{n-1}b_{n-2}...b_i...b_1b_0$

	R_{n-1}	R_{n-2}	...	R_{i-1}	...	R_0	
		a_{n-1}	...	a_i	...	a_1	a_0
+		b_{n-1}	...	b_i	...	b_1	b_0
		S_{n-1}	...	S_i	...	S_1	S_0

La table de vérité et les diagrammes de Karnaugh correspondant à la somme et la retenue sont illustrés comme suit :

a_i	b_i	R_{i-1}	S_i	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

S_i

a_i \ b_iR_{i-1}	00	01	11	10
0		1		1
1	1		1	

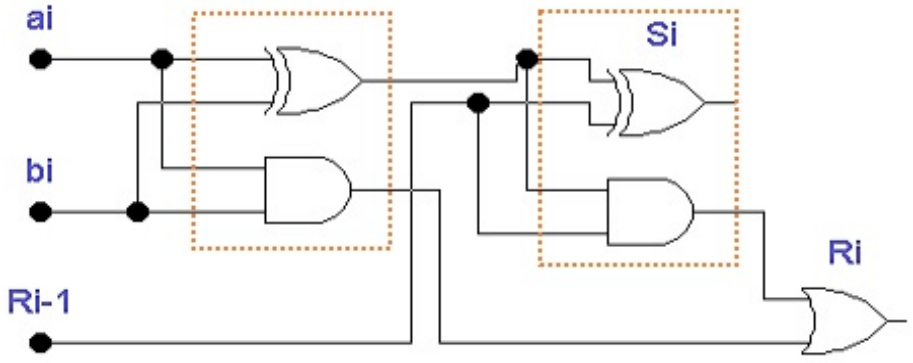
R_i

a_i \ b_iR_{i-1}	00	01	11	10
0			1	
1		1	1	1

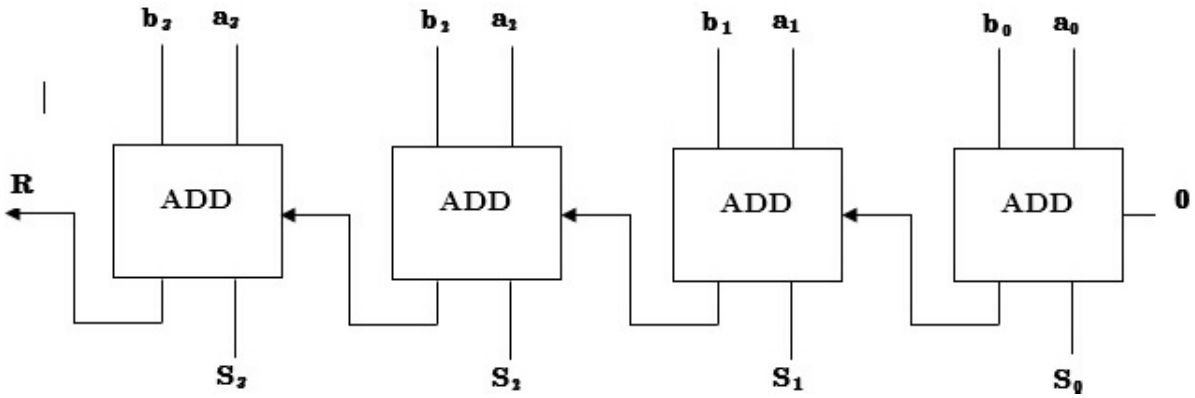
D'après la table de vérité, les formes canoniques de S_i et R_i sont données comme suit :

- $R_i = \overline{a_i}b_iR_{i-1} + a_i\overline{b_i}R_{i-1} + a_ib_i\overline{R_{i-1}} + a_ib_iR_{i-1} \implies R_i = R_{i-1}(\overline{a_i}b_i + a_i\overline{b_i}) + a_ib_i(R_{i-1} + \overline{R_{i-1}})$, il vient : $R_i = a_ib_i + R_{i-1}(\ a_i \oplus b_i)$
- $S_i = \overline{a_i}\,\overline{b_i}R_{i-1} + \overline{a_i}b_i\overline{R_{i-1}} + a_i\overline{b_i}\overline{R_{i-1}} + a_ib_iR_{i-1} \implies S_i = R_{i-1}(\overline{a_i}\,\overline{b_i} + a_ib_i) + \overline{R_{i-1}}(\overline{a_i}b_i + a_i\overline{b_i})$
  $\implies S_i = R_{i-1}(\overline{a_i \oplus b_i}) + \overline{R_{i-1}}(a_i \oplus b_i)$, il vient :
  $S_i = a_i \oplus b_i \oplus R_{i-1}$

D'où le logigramme,



Le dispositif qui réalise l'addition de deux nombres binaires A et B de n bits chacun est constitué d'un demi-additionneur au rang 0 et de $(n-1)$ additionneurs complets mis en cascade, comme le montre la figure suivante :



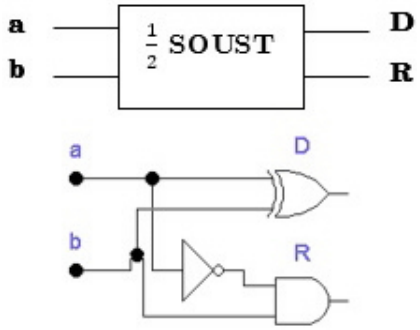
3.2.2 Le soustracteur binaire

3.2.2.1 Le demi-soustracteur ($\frac{1}{2}$ $SOUST$)

Ressemblant au cas de la demi-addition. C'est la soustraction simple de deux bits a et b. On obtient donc une différence $D = a - b$ et une éventuelle retenue R. Et d'après la table de vérité illustrée ci-bas, on obtient les équations de D et R comme suit :

$$\begin{cases} D = a.\overline{b} + \overline{a}.b \\ R = \overline{a}.b \end{cases} \Longrightarrow \begin{cases} D = a \oplus b \\ R = \overline{a}.b \end{cases}$$

a	b	D	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



3.2.2.2 Le soustracteur complet ($SOUST$)

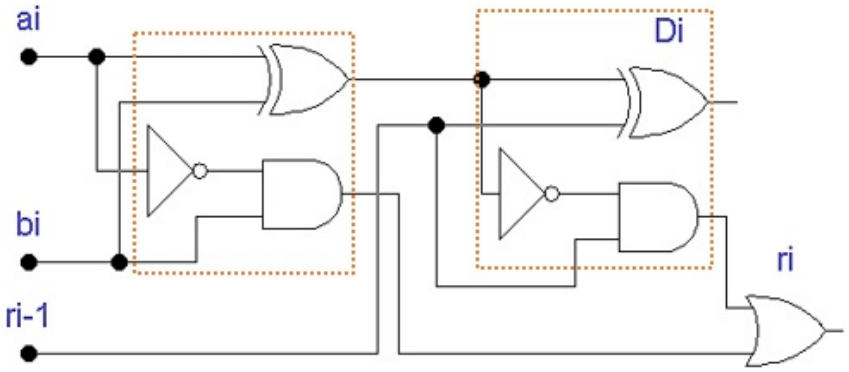
Considérons le cas de deux nombre binaires A et B de n bits chacun. Les résultats sont la différence $D_i = a_i - b_i - r_{i-1}$ et la retenue r_i, données comme suit :

a_i	b_i	r_{i-1}	D_i	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

D'après la table de vérité :

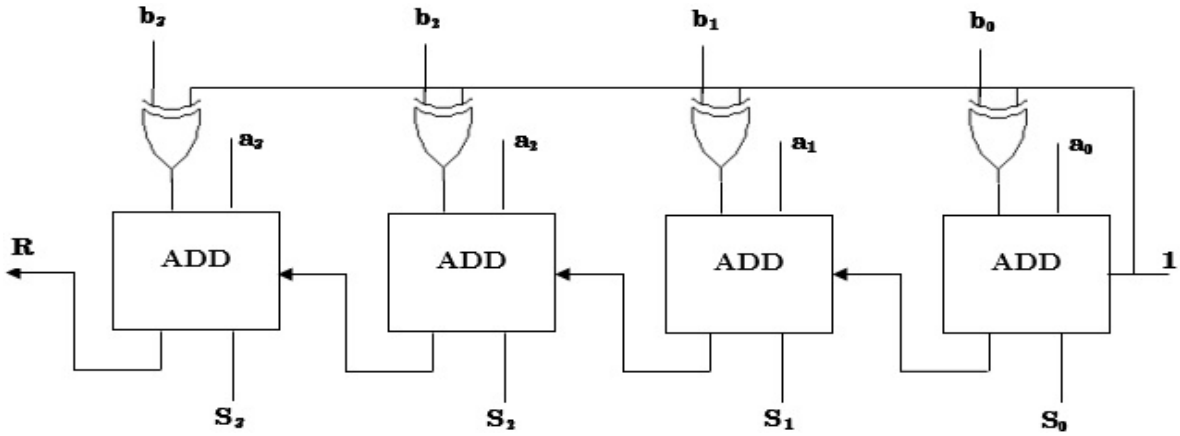
- $r_i = \overline{a_i}\overline{b_i}r_{i-1} + \overline{a_i}b_i\overline{r_{i-1}} + \overline{a_i}b_ir_{i-1} + a_ib_ir_{i-1} \implies r_i = a_i\overline{b_i}(\overline{r_{i-1}} + r_{i-1}) + r_{i-1}(\overline{a_i}\overline{b_i} + a_ib_i)$, il vient : $r_i = a_i\overline{b_i} + r_{i-1}(\overline{a_i \oplus b_i})$
- $D_i = S_i$ c'est-à-dire que : $D_i = a_i \oplus b_i \oplus R_{i-1}$

D'où le logigramme,



Cet additionneur est capable de traiter des nombres négatifs représentés en complément à deux comme indiqué dans le *chapitre* 01. Il reste à déterminer comment traiter la soustraction ?

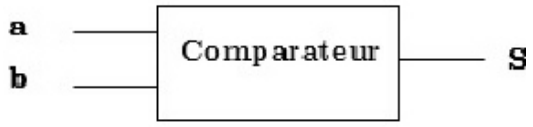
On remarque que l'expression $x - y$ est équivalente à celle $x + (-y)$. Au 1^{er} chapitre, on a vu, que pour avoir l'opposé d'un nombre il suffit d'inverser chacun de ses bits, puis on l'additionne à 1 au résultat. Il reste donc à calculer l'expression $x + \overline{y} + 1$. Afin calculer $\overline{y}$, on inverse chaque position de y avant d'arriver à l'additionneur, mais comment ajouter 1 par la suite ? Cette opération semble nécessiter encore un additionneur. Heureusement, nous avons une entrée (retenue précédente) de la position 0 qui n'est pas actuellement utilisée. Or donner la valeur 1 à cette entrée additionne 1 au résultat entier. Le circuit complet pour la soustraction est présenté sur la figure.



3.2.3 Les comparateurs

C'est un circuit à deux entrées d'informations binaires a et b en plus d'une sortie S, indiquant si ses deux informations sont égales ou non. Et d'après la table de vérité on en déduit que l'opérateur $XNOR$ représente le comparateur binaire élémentaire de deux bits.

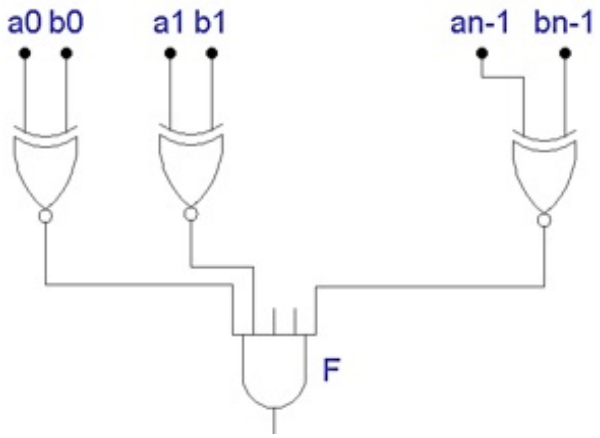
a	b	S
0	0	1
0	1	0
1	0	0
1	1	1



Or, si on considère que A et B sont deux informations de n bits chacun telle que :

$$\begin{cases} A = a_{n-1}a_{n-2}...a_i...a_1a_0 \\ B = b_{n-1}b_{n-2}...b_i...b_1b_0 \end{cases}$$

Afin de satisfaire l'égalité $A = B$, il est impératif que tous les bits des deux informations en question du même rang soient égaux, sinon $A \neq B$.



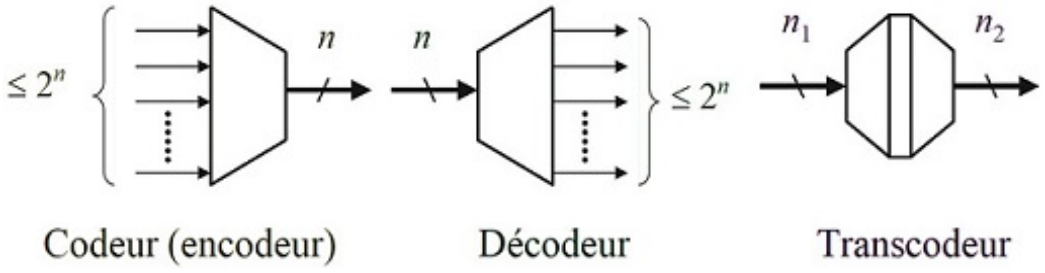
D'où $F = \overline{a_0 \oplus b_0} \cdot \overline{a_1 \oplus b_1} \ \ ... \ \ \overline{a_{n-1} \oplus b_{n-1}}$

3.3 Les circuits de transcodage

Un opérateur de transcodage est un circuit qui converti une information présente en entrée et exprimée dans un code A en son équivalent dans un autre code B, telle que :

$$\begin{cases} I_{cA} = A_1A_2A_3... \\ I_{cB} = B_1B_2B_3... \end{cases}$$

D'où il faut établir les fonctions suivantes :



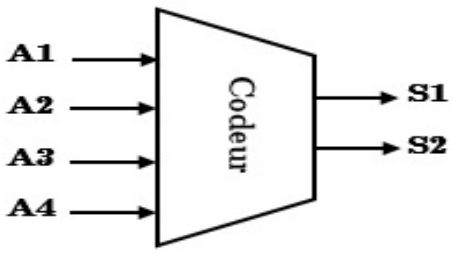
$$\begin{cases} B_1 = f(A_1, A_2, A_3, ...) \\ B_2 = f(A_1, A_2, A_3, ...) \\ B_3 = f(A_1, A_2, A_3, ...) \\ ... \\ ... \end{cases}$$

3.3.1 Le codeur (encodeur)

C'est un circuit logique qui possède 2^n entrées, dont une seule est activée à la fois en plus de n sorties.

Exemple : Codeur à 04 voies d'entrées et 02 bits de sortie.

Entrées				*Sorties*	
A_3	A_2	A_1	A_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



On pourrait obtenir les équations de sortie directement sans passer par la table de Karnaugh, comme suit :

$S_0 = 1$ si $A_1 = A_3 = 1 \implies S_0 = A_1 + A_3$

$S_1 = 1$ si $A_2 = A_3 = 1 \implies S_1 = A_2 + A_3$

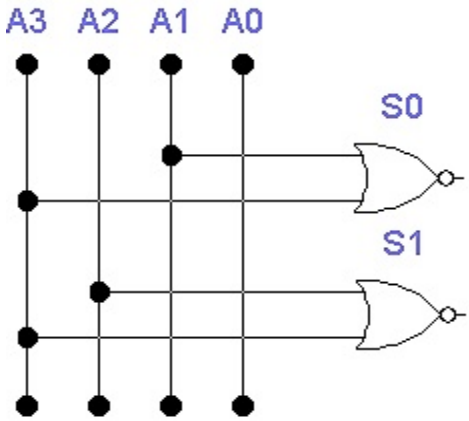
D'où le logigramme du codeur :

3.3.2 Le décodeur

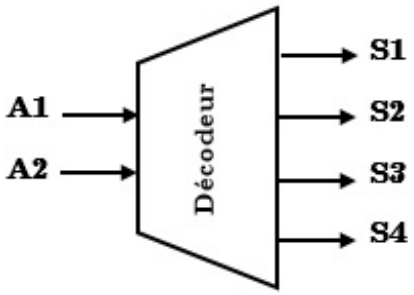
C'est un circuit combinatoire logique qui possède une entrée binaire codées sur n bits et une seule sortie active parmi les 2^n cas possibles.

Exemple : Décodeur 1 parmi 4.

On a besoin de deux bits à l'entrée afin d'activer les quatre voies de sortie en suivant la règle $2^2 = 4$.



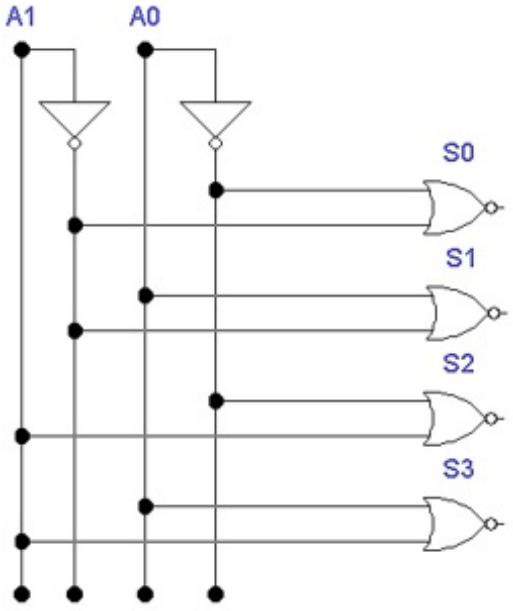
Entrées		*Sorties*			
A_1	A_0	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



On pourrait extraire les équations de sortie directement de la table de vérité, comme suit :

$S_0 = \overline{A_1}.\overline{A_0}$ $S_1 = \overline{A_1}.A_0$ $S_2 = A_1.\overline{A_0}$ $S_3 = A_1 A_0$

D'où le logigramme du décodeur :

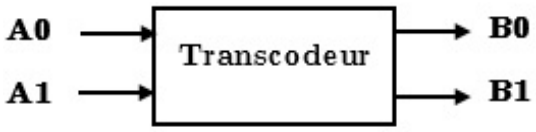


3.3.3 Le transcodeur

Un transcodeur fait correspondre une information A en entrée sous forme d'un code sur p lignes en une autre information B en sortie (généralement un autre code) sur m lignes.

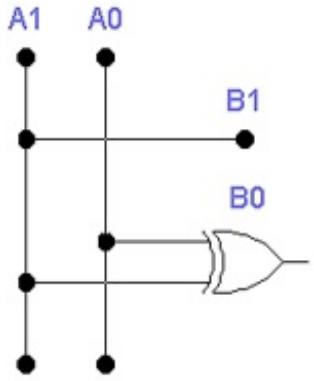
Exemple : A titre d'exemple nous allons étudier un transcodeur Binaire-Gray, qui permet de convertir un code binaire de 02 bits en un autre code Gray de 02 bits. Contrairement au codeur et au décodeur, on remarque que le nombre de lignes en entrée est le même qu'en sortie. D'où la règle de la puissance n'est pas vérifiée.

Entrées		*Sorties*	
A_1	A_0	B_1	B_0
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



Les équations de sortie et le logigramme sont données comme suit :

$$B_0 = A_1.\overline{A_0} + \overline{A_1}.A_0 = A_1 \oplus A_0$$
$$B_1 = A_1.\overline{A_0} + A_1.A_0 = A_1$$



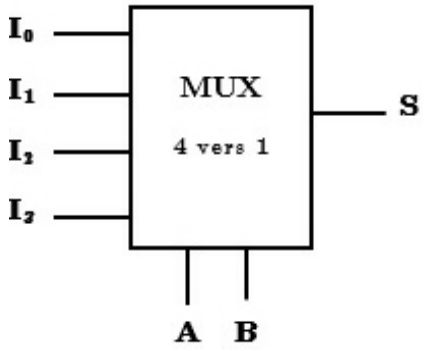
3.4 Les circuits d'aiguillage

3.4.1 Le multiplexeur (MUX)

C'est un circuit combinatoire qui permet traiter successivement les données présentes sur 2^n entrées d'information $I_0, I_1, \ldots$, commandées par n entrées d'adresse (ou entrées de sélection) $A, B, \ldots$, où ces entées sont dirigés l'une après l'autre vers une unique sortie S.Ainsi que la commande de validation V permet d'autoriser ou non le fonctionnement du multiplexeur, c'est-à-dire que si $V = 0$, $S = 0$ quels que soient les états des entrées de sélection. Le multiplexeur permet de choisir une entrée parmi d'autres, donc il est du type 2^n *vers* 1.

Exemple : On prend à titre d'exemple un MUX à 04 entrées ($4=2^2$ vers 1). Celui-ci nécessite 02 entrées de sélection A et B ($4=2^2$). La table de vérité correpondante est donnée comme suit :

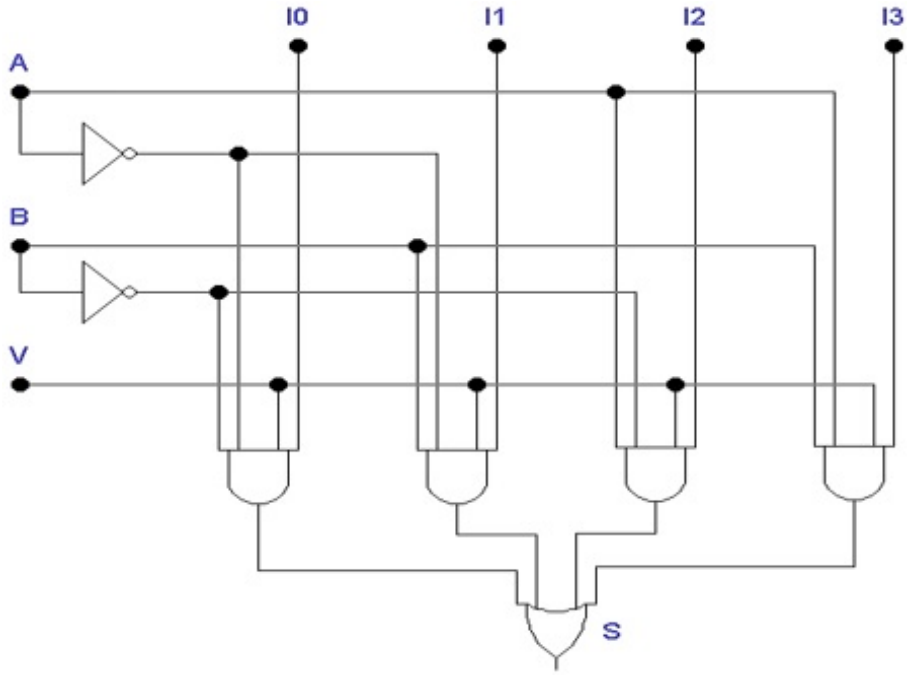
Validation	*Adresses*		*Sortie*
V	A	B	S
0	ϕ	ϕ	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3



On déduit l'expression de S :

$S = V.(\overline{A}.\overline{B}.I_0 + \overline{A}.B.I_1 + A.\overline{B}.I_2 + A.B.I_3)$

Ainsi le schéma du circuit est donné comme suit :

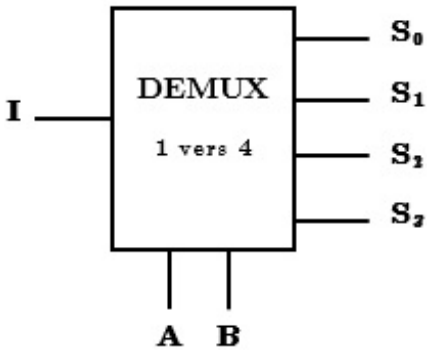


3.4.2 Le démultiplexeur ($DEMUX$)

C'est un circuit combinatoire à une entrée d'information I, n entrées d'adresse (entrées de sélection) A, B, ... et 2^n sorties S_0, S_1, ... Il réalise ainsi l'opération inverse du Multiplexeur. On dit qu'il est du type 1 *vers* 2^n.

Exemple : On prend l'exemple un $DEMUX$ à 04 sorties (1 vers 4=2^2). Il nécessite 02 entrées de sélection A et B (4=2^2). La table de vérité correpondante est donnée comme suit :

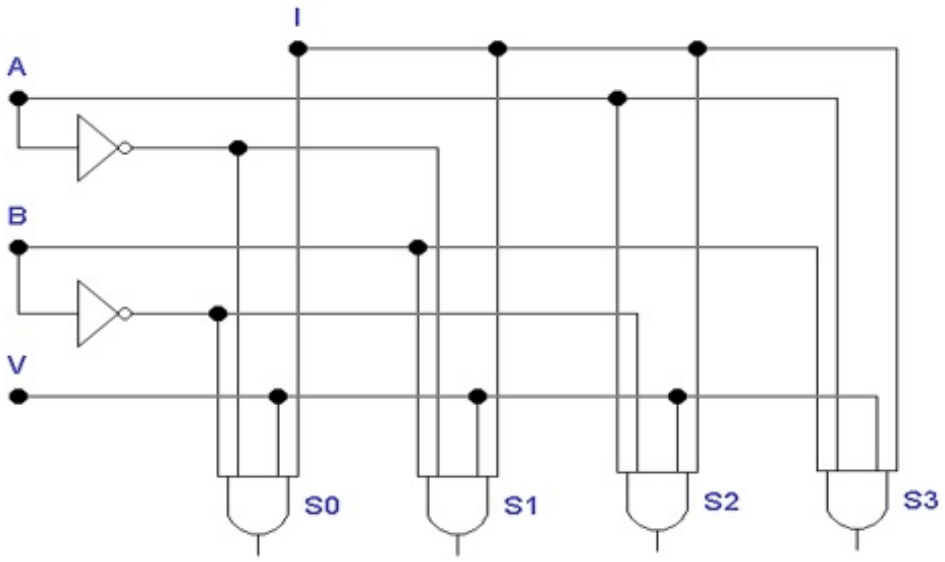
Validation	*Adresses*		*Sorties*			
V	A	B	S_0	S_1	S_2	S_3
0	ϕ	ϕ	0	0	0	0
1	0	0	I	0	0	0
1	0	1	0	I	0	0
1	1	0	0	0	I	0
1	1	1	0	0	0	I



On déduit les expressions des sorties :

$S_0 = V.I.\overline{A}.\overline{B}$ $S_1 = V.I.\overline{A}.B$ $S_2 = V.I.A.\overline{B}$ $S_3 = V.I.A.B$

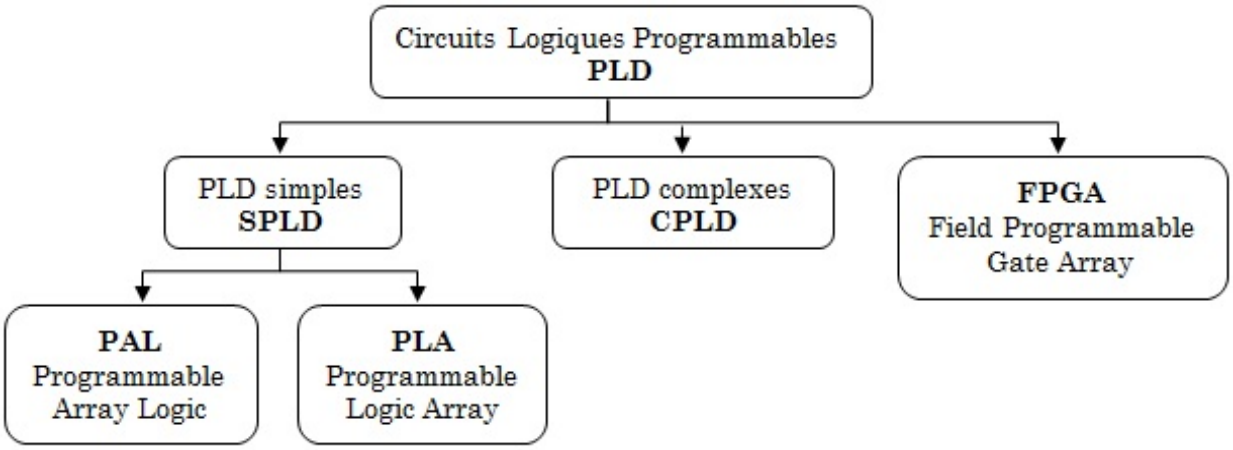
Le schéma du circuit est donné comme suit :



3.5 Circuits logiques programmables (PLD)

Les dispositifs logiques constituent l'une des trois catégories importantes de dispositifs utilisés pour construire des systèmes électroniques numériques, les dispositifs de mémoire et les microprocesseurs étant les deux autres. Des dispositifs de mémoire tels que ROM et RAM sont utilisés pour stocker des informations telles que les instructions logicielles d'un programme ou le contenu d'une base de données, et les microprocesseurs exécutent des instructions logicielles pour exécuter diverses fonctions, allant de l'exécution d'un programme de traitement de texte à la réalisation de tâches distantes.

La fonction à exécuter par un dispositif logique programmable n'est pas définie au moment de sa fabrication. Ces périphériques sont programmés par l'utilisateur pour effectuer une gamme de fonctions en fonction de la capacité logique et des autres fonctionnalités offertes par le périphérique. Il est à noter que les circuits FPGA seront représentés dans un autre module, nommé Systèmes Numériques 2.



3.5.1 Logique fixe ou logique programmable

Il existe deux grandes catégories de dispositifs logiques, à savoir les dispositifs à logique fixe et les dispositifs à logique programmable. Alors qu'un dispositif logique fixe tel qu'une porte logique, un multiplexeur ou une bascule remplit une fonction logique donnée au moment de la fabrication du dispositif, un dispositif logique programmable peut être configuré par l'utilisateur pour réaliser une grande variété de fonctions logiques. En termes de disposition schématique interne des deux types de dispositifs, les circuits ou blocs de construction et leurs interconnexions dans un dispositif à logique fixe sont permanents et ne peuvent plus être modifiés après la fabrication du dispositif. Or, un dispositif logique programmable offre à l'utilisateur une large gamme de capacités logiques en termes de blocs de construction numériques, que l'utilisateur peut configurer pour exécuter la fonction ou l'ensemble de fonctions souhaité.

Une telle configuration peut être modifiée ou modifiée autant de fois que nécessaire par l'utilisateur en reprogrammant l'appareil.

3.5.2 Les circuits logiques programmables simples (SPLD)

3.5.2.1 Structure des SPLD

Un SPLD se constitue essentiellement de deux partie, une matrice programmable d'une part, et une structure de sortie non programmable d'une autre part.

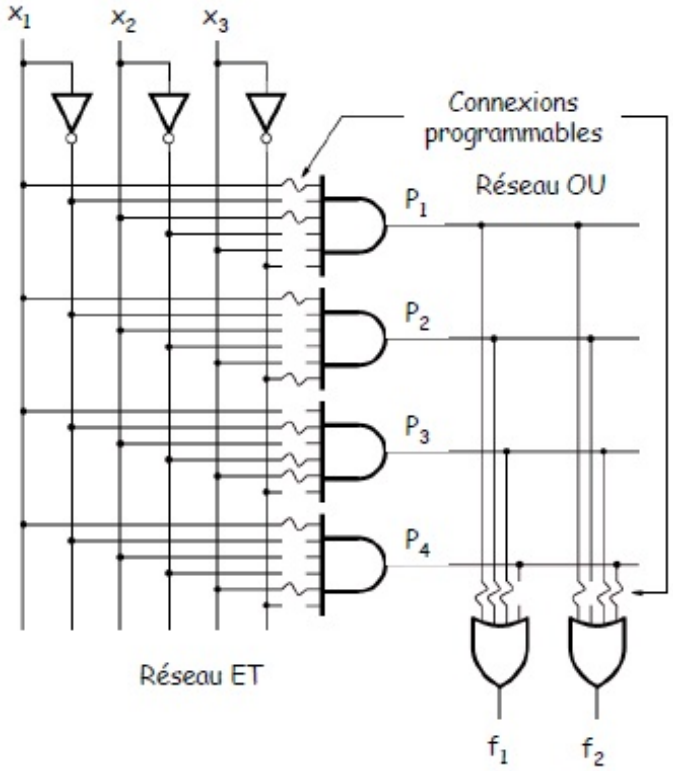
3.5.2.2 Les Programmable Logic Array (PLA)

L'idée de base d'un PLA, est que toutes les fonctions logiques peuvent être représentées sous une forme canonique (Exactement une somme de produits). Il est composé d'un ensemble de portes ET relié à un jeu de portes OU.

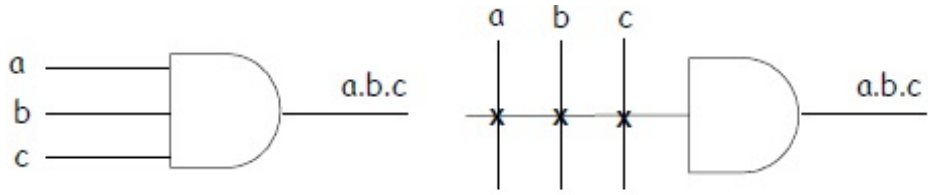
Un exemple de PLA avec 3 entrées (x_1, x_2 , x_3), 4 termes produits et 2 sorties est représenté sur la figure ci-bas. Chaque porte ET a 6 entrées correspondant à chaque variable x_i et à son complément

$\overline{x_i}$. Les connexions sur une porte *ET* sont programmables et une entrée connectée est représentée par le symbole $\sim$. Finalement on prend :

$$\begin{cases} f_1 = x_1x_2 + x_1\overline{x_3} + \overline{x_2}x_3 \\ f_2 = x_1x_2 + \overline{x_1x_2} + x_1x_3 \end{cases}$$



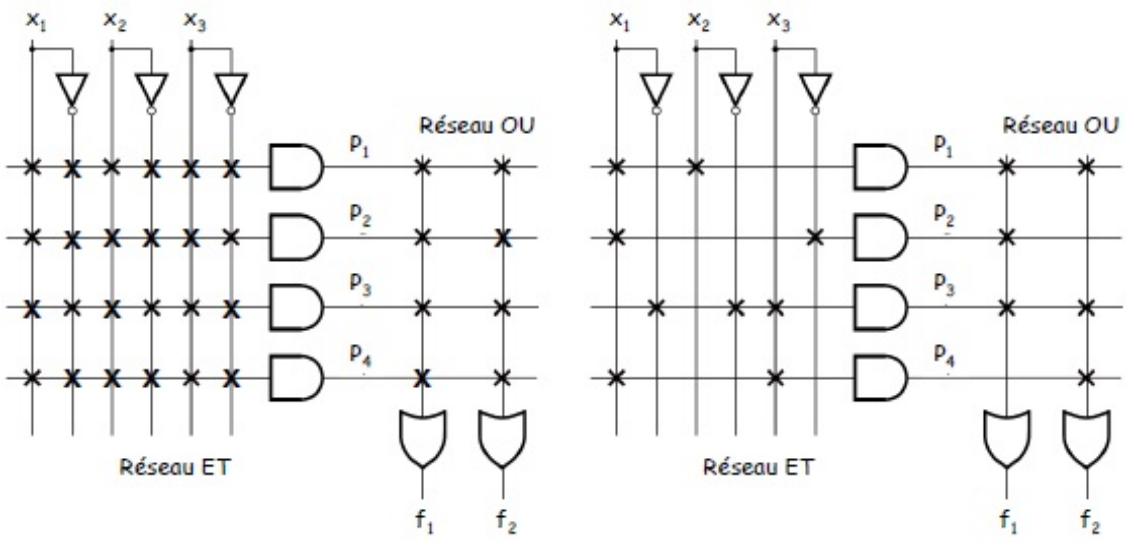
La figure suivantes montre comment schématiser un circuit PLA afin d'éviter les encombrements dans les connexions.



Et la prochaine figure montre une Représentations schématiques d'un PLA vierge (à gauche) et du PLA programmé (à droite).

3.5.2.3 Les Programmable Array Logic (PAL)

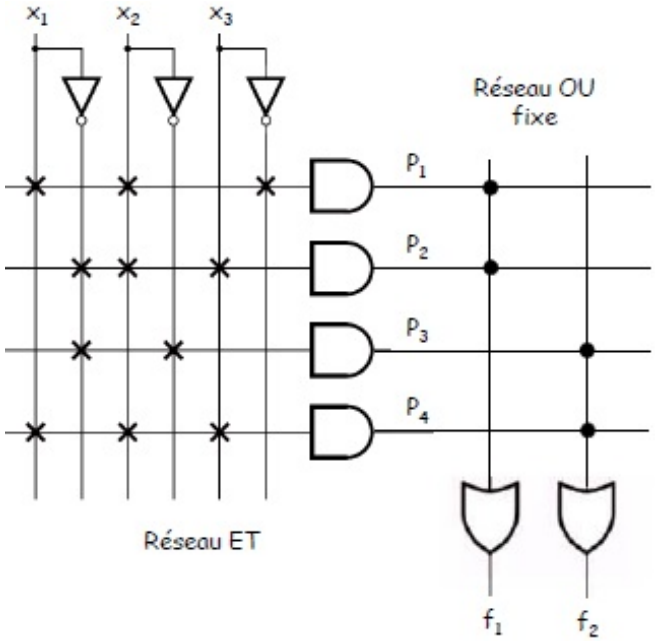
Dans les PLA les réseaux ET et OU sont programmables. Pour les constructeurs, les connexions programmables ont présentées deux difficultés majeures : elles étaient délicates à fabriquer correctement, et limitaient la vitesse des circuits implémentés sur le PLA. Ce qui a conduit au développement de composants similaires dans lesquels seul le réseau ET est programmable tandis que le réseau OU



reste fixe. Dont On parle de Programmable Array Logic (PAL), plus simples à fabriquer, plus rapides et moins chers que les PLA.

Un exemple de PAL avec 3 entrées, 4 termes de produits et 2 sorties est présenté sur la figure qui suit, avec :

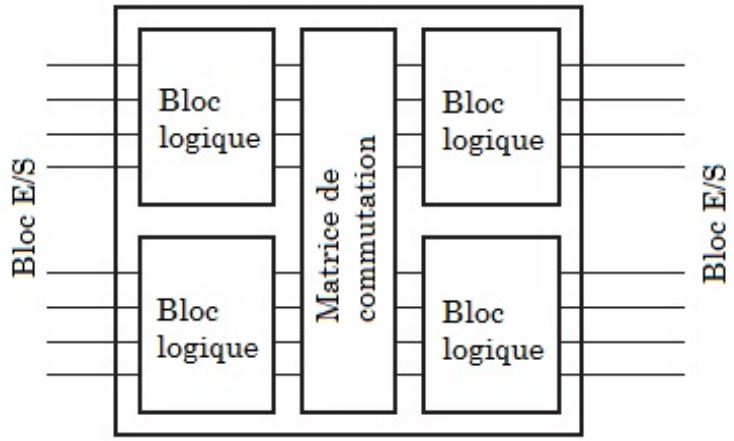
$$\begin{cases} f_1 = x_1 x_2 \overline{x_3} + \overline{x_1} x_2 x_3 \\ f_2 = \overline{x_1 x_2} + x_1 x_2 x_3 \end{cases}$$



Seulement, le PAL est beaucoup moins flexible que le PLA, d'où chaque porte OU ne peut sommer que 2 termes au lieu de 4 dans le PLA.

3.5.3 Les Complex Programmable Logic Device (CPLD)

Ou Dispositif logique programmable complexe Les dispositifs logiques programmables tels que les PLA, PAL et autres dispositifs de type PAL sont souvent regroupés dans une seule catégorie appelée dispositifs de logique programmable simple (SPLD) afin de les distinguer de ceux qui sont beaucoup plus complexes. Comme son nom l'indique, un dispositif de logique programmable complexe (CPLD) est beaucoup plus complexe que tous les dispositifs de logique programmable décrits jusqu'à présent. Un CPLD peut contenir des circuits équivalents à ceux de plusieurs périphériques PAL reliés entre eux par des interconnexions programmables. La suivante montre la structure interne d'un CPLD.



Chacun des quatre blocs logiques est équivalent à un PLD tel qu'un appareil PAL. Le nombre de blocs logiques dans un CPLD peut être supérieur ou inférieur à quatre. Chacun des blocs logiques contient des interconnexions programmables. Une matrice de commutation est utilisée pour les interconnexions entre blocs logiques. De plus, la matrice de commutation d'un CPLD peut être connectée complètement ou non. Alors que la complexité d'un périphérique PAL typique peut être de l'ordre de quelques centaines de portes logiques, un CPLD peut avoir une complexité équivalente à des dizaines de milliers de portes logiques. En outre, et en raison de leur consommation électrique relativement faible et de leur coût réduit, les CPLD constituent une solution idéale pour les applications portables alimentées par batterie, telles que les téléphones portables, les assistants numériques, etc. Un CPLD peut être programmé en utilisant un programmateur PAL ou en l'alimentant avec un flux de données série à partir d'un PC.

Deuxième partie

Logique séquentielle

Chapitre 4

Introduction aux circuits séquentiels

4.1 Du combinatoire au séquentiel

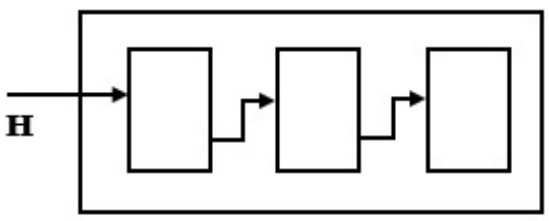
Dans les circuits combinatoires, la sortie ne dépend que des variables d'entrées. Par contre, dans les circuits séquentiels, une sortie dépend non seulement des variables d'entrées mais également des sorties du même circuit. Il y a une sorte de rétroaction des sorties sur les entrées. Cela signifie qu'un circuit séquentiel garde en mémoire la valeur des états précédents. Les systèmes numériques fonctionnent d'une manière asynchrone ou synchrone.Dans les systèmes asynchrones, les sorties des circuits logiques peuvent changer d'état chaque fois que l'une ou plusieurs des entrées changent aussi leurs états.

4.1.1 Circuits séquentiels asynchrones

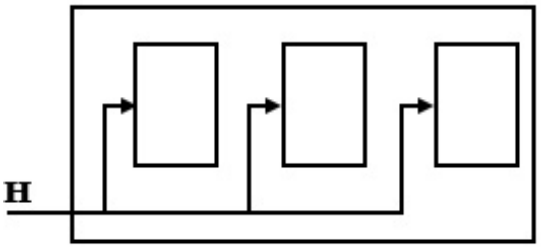
Un système asynchrone est généralement plus difficile à concevoir qu'un système synchrone. Or, il évolue sans ordre extérieur. Les modifications des entrées provoquent une réponse immédiate des états de sortie.

4.1.2 Circuits séquentiels synchrones

Un tel sysème n'évolue que sur un ordre extérieur. Ses variables d'entées, de sortie et internes sont commandées par un signal d'horloge extérieur au système. Et la sortie peut changer son état qu'à travers un signal d'horloge.



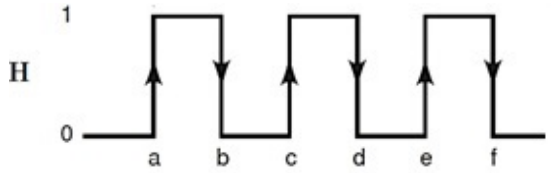
Système séquentiel asynchrone

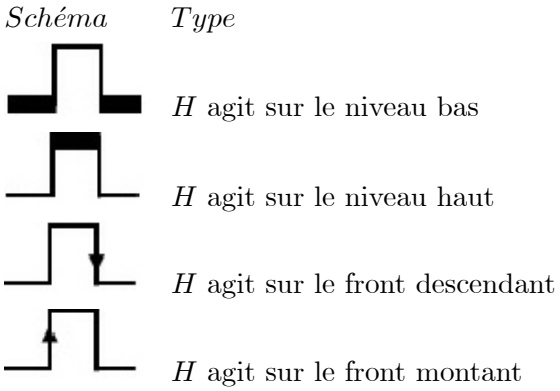


Système séquentiel synchrone

4.2 L'horloge (H) ou Clock (CLK)

L'horloge est généralement représentée par un train d'impulsions rectangulaire, comme illustré à la figure ci-bas. Le signal d'horloge est distribué à toutes les parties du système, alors que la plupart des sorties du système ne peuvent changer d'état que lorsque l'horloge effectue une transition. C'est-à-dire passer successivement de 0 à 1 et de 1 à 0 d'une manière périodique dans le temps.





Cependant, les ordinateurs dans nos jours sont synchrones, c'est-à-dire que les sorties de tous les circuits changent simultanément suivant le signal d'horloge.

Il existe deux types de synchronisation. Une synchronisation par niveau, où tous les changements d'état ont lieu pendant que l'horloge est à un niveau fixé, haut ou bas. Et une synchronisation par front d'impulsion, où tous les changements d'état se font sur un front d'impulsion fixé, montant $\uparrow$ ou descendant $\downarrow$.

4.3 Les bascules

4.3.1 Définition

La bascule est un circuit de base d'un système séquentiel. Elle possède deux état stables 0 et 1 sous forme de mémoire binaire élémentaire. Autrement dit, c'est un opérateur capable de changer d'état sur commande et de conserver ainsi son état jusqu'à la prochaine commande. On différencie les bascules à fonctionnement asynchrone et celles à fonctionnement synchrone. La différence principale entre les deux types de bascule est que la première n'a pas de signal d'horloge, alors que le second en a toujours un. Il est à noter qu'il existe quatre principaux types de bascules : SR, JK, D et T.

4.3.2 Bascule SR

Elle comporte deux entrées : S (Set), pour mise à 1 et R (Reset), pour remise à 0 et deux sortie Q et $\overline{Q}$. La table de vérité de la bascule en question est donnée comme suit :

Entrées			*Sortie*
R	S	Q	Q^+
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	ϕ
1	1	1	ϕ

$\Longrightarrow$

Entrées		*Sortie*
R	S	Q^+
0	0	Q (état précédent)
0	1	1
1	0	0
1	1	ϕ (état indéterminé)

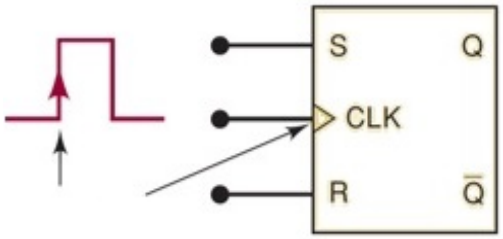
D'où la simplification par la table de Karnaugh :

RS / Q	00	01	11	10
0		1	ϕ	
1	1	1	ϕ	

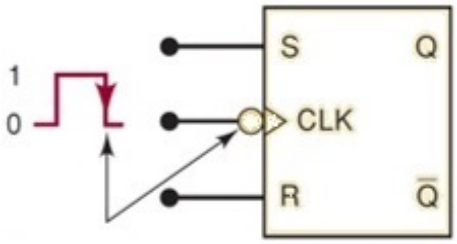
$$Q^+ = S + \overline{R}.Q \implies Q^+ = \overline{\overline{S + \overline{R}.Q}}$$
$$\implies Q^+ = \overline{\overline{S}.\overline{\overline{R}.Q}}$$

Le logigramme en fonction des portes $NAND$ seulemement est illustré comme suit :

La bascule SR doit être commandée ou synchronisée. La figure ci-dessous illustre le symbole logique d'une bascule SR déclenchée par le front montant (descendant) du signal d'horloge. Cela signifie que la bascule ne peut changer d'état que lorsqu'un signal appliqué à son entrée d'horloge effectue une transition de 0 à 1.

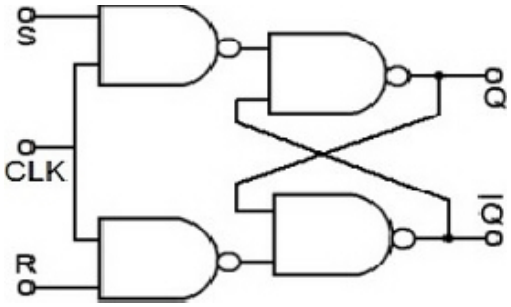


Bascule SR commandée par
le front montant de l'horloge



Bascule SR commandée par
le front descendant de l'horloge

Le schéma détaillé d'une bascule commandée par le front montant de l'horloge est illustré comme suit :



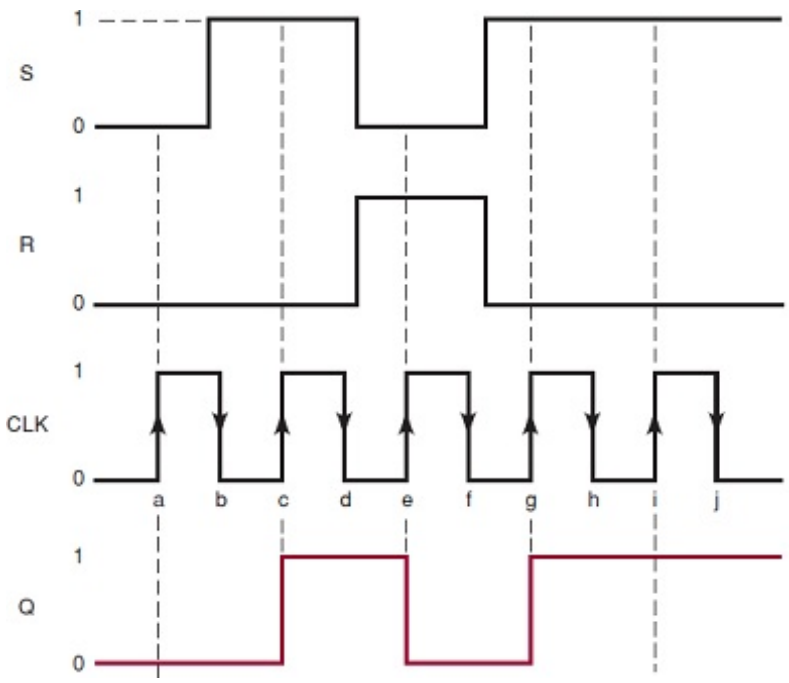
La table ci-bas montre comment répond la sortie pour les différentes combinaisons d'entrées S et R. La flèche vers le haut ($\uparrow$) indique que la bascule déclenchée par le front montant du signal d'horloge. Inversement, la flèche vers le bas ($\downarrow$) indique que la bascule déclenchée par le front descendant du même signal.

Entrées			*Sortie*
R	S	CLK	Q^+
0	0	$\uparrow\downarrow$	Q (état précédent)
0	1	$\uparrow\downarrow$	1
1	0	$\uparrow\downarrow$	0
1	1	$\uparrow\downarrow$	ϕ (état indéterminé)

Les chronogrammes de la figure suivante illustrent le fonctionnement de la bascule SR déclenchée par le front montant du signal d'horloge.

4.3.3 Bascule JK

La bascule JK est une bascule SR plus élaborée, qui est appelée aussi bascule universelle. Sauf pour la bascule SR, la sortie pour $(S, R) = (1, 1)$ est indéterminée, alors que dans la bascule JK, la sortie passera à son état opposé lors de la commande du signal d'horloge. C'est ce qu'on appelle le mode de fonctionnement à bascule. La table de vérité de la bascule en question est donnée comme suit :



Entrées			*Sortie*
J	K	Q	Q^+
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	1
1	1	1	0

$\Longrightarrow$

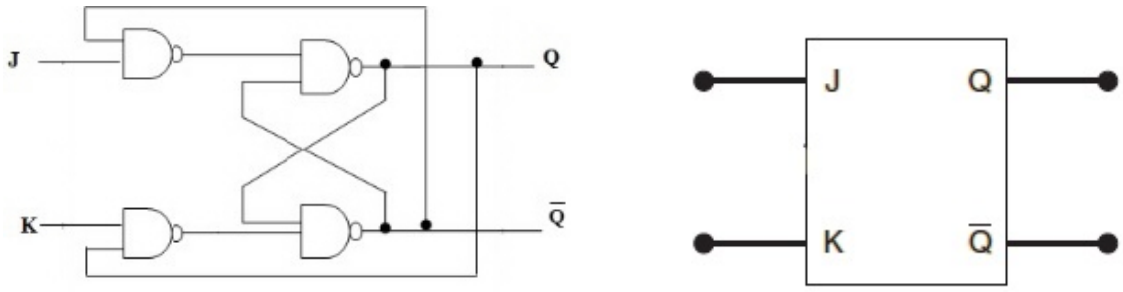
Entrées		*Sortie*
J	K	Q^+
0	0	Q (état précédent)
0	1	1
1	0	0
1	1	$\overline{Q}$ (basculement)

D'où la simplification par la table de Karnaugh :

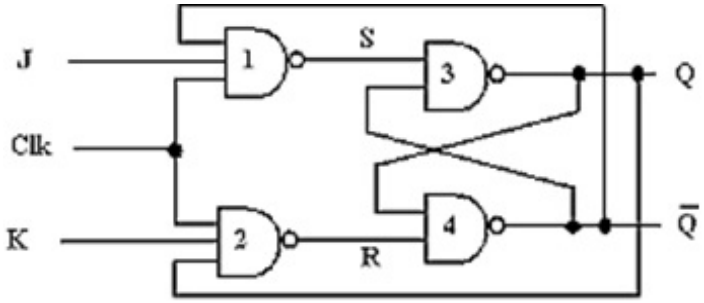
Q \ JK	00	01	11	10
0			1	1
1	1			1

$$Q^+ = J.\overline{Q} + \overline{K}.Q \implies Q^+ = \overline{\overline{J.\overline{Q} + \overline{K}.Q}}$$
$$\implies Q^+ = \overline{J.\overline{Q}.\overline{\overline{K}.Q}}$$

Le logigramme en fonction des portes $NAND$ seulement est illustré comme suit :



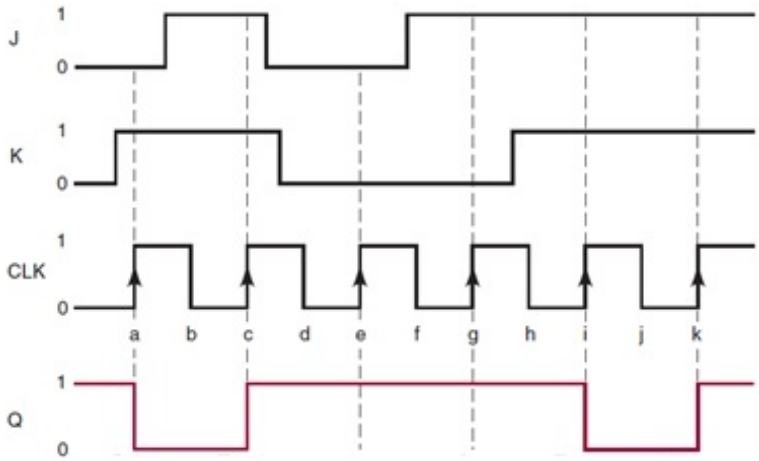
En utilisant la synchronisation de l'horloge :



La table ci-bas montre l'tat de sortie pour les différentes combinaisons d'entrées J et K.

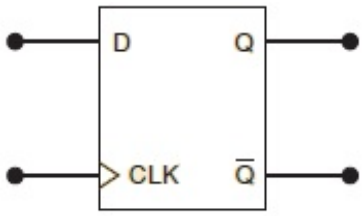
Entrées			*Sortie*
J	K	CLK	Q^+
0	0	$\uparrow\downarrow$	Q (état précédent)
0	1	$\uparrow\downarrow$	1
1	0	$\uparrow\downarrow$	0
1	1	$\uparrow\downarrow$	$\overline{Q}$ (basculement)

Les chronogrammes de la figure suivante illustrent le fonctionnement de la même bascule déclenchée par le front montant du signal d'horloge.



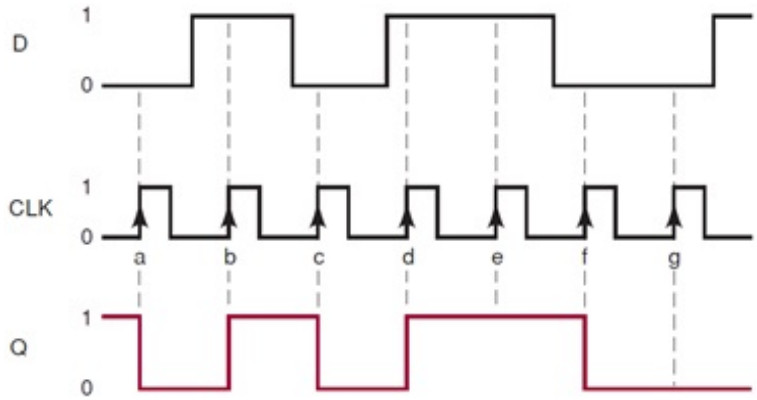
4.3.4 Bascule D

Entrée		*Sortie*
D	CLK	Q^+
0	$\uparrow\downarrow$	0
1	$\uparrow\downarrow$	1



D'après la table de vérité ; on conclut directement que : $Q^+ = D$

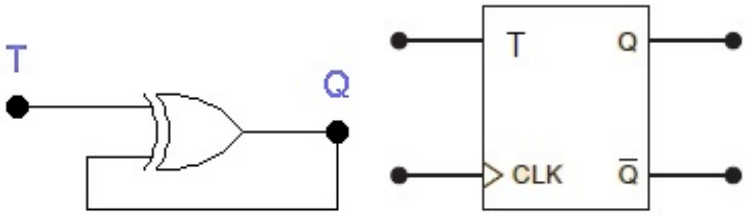
La figure ci-dessus montre le symbole et le tableau des fonctions d'une bascule D synchronisée et déclenchée par le front montant de l'horloge. Contrairement aux bascules SR et JK, cette bascule n'a qu'une seule entrée de commande synchrone, D, qui représente des données. Le fonctionnement de la bascule D est très simple : Q passera au même état que celui présent sur l'entrée D lors de la détection du front montant. En d'autres termes, le niveau présent en D sera stocké dans la bascule au. Les chronogrammes de la figure qui suit illustrent cette opération.



4.3.5 Bascule T

Une telle bascule dispose d'une seule entrée T et change son état à chaque impulsion d'horloge, son seul et principal fonctionnement est le basculement. La table de vérité est le schéma de la bascule sont illustrés comme suit :

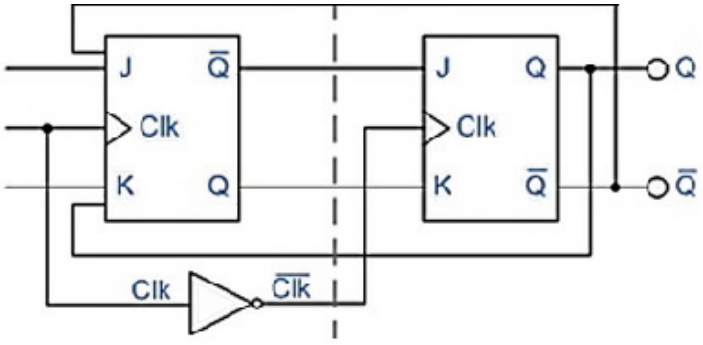
Entrées			*Sortie*
T	Q	CLK	Q^+
0	0	$\uparrow\downarrow$	0
0	1	$\uparrow\downarrow$	1
1	0	$\uparrow\downarrow$	1
1	1	$\uparrow\downarrow$	0



Où, $Q^+ = T \oplus Q$

4.3.6 Bascule maitre-esclave

La bascule maitre-esclave est représentée par la mise en cascade de deux bascules élémentaires, où la sortie de la première bascule (maitre) constitue l'entrée de la deuxième bascule (esclave), avec un même signal d'horloge, seulement il est inversé pour la bascule esclave. La figure qui suit montre une telle représentation avec deux bascule JK.



- Si CLK est activée au front montant $\uparrow$: la sortie de la bascule maitre prend l'état logique correspondant aux entrées synchrones J et K.
- Sinon (CLK est activée au front desendant $\downarrow$) : dans ce cas l'étage maitre est déconnecté des entrées de la bascule maitre et relié ainsi à l'étage esclave. Et c'est à ce dernier de délivrer l'information de sortie.

4.3.7 Réalisation des bascules synchrones SR, D et T en fonction de JK

Puisque la bascule JK est universelle, alors on pourrait réaliser les autres types en se basant sur la bascule en question à l'aide de leurs équations de sortie.

4.3.7.1 SR en fonction de JK

$$\begin{cases} Q^+_{RS} = S + \overline{R}.Q \\ Q^+_{JK} = J.\overline{Q} + \overline{K}.Q \end{cases}$$, on utilise un élement neutre,

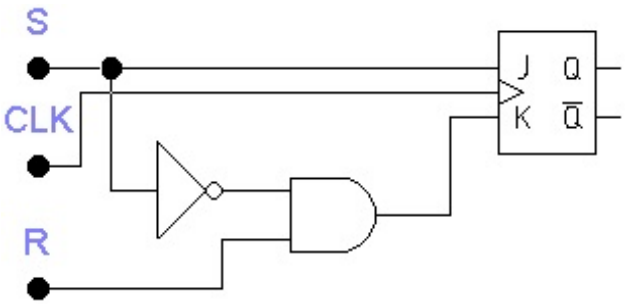
$$\begin{cases} Q^+_{RS} = S.(Q + \overline{Q}) + \overline{R}.Q \\ Q^+_{JK} = J.\overline{Q} + \overline{K}.Q \end{cases} \Longrightarrow \begin{cases} Q^+_{RS} = S.\overline{Q} + (S + \overline{R}).Q \\ Q^+_{JK} = J.\overline{Q} + \overline{K}.Q \end{cases}$$, avec $Q^+_{RS} = Q^+_{JK}$ il vient :

$$\begin{cases} \overline{K} = S + \overline{R} \\ J = S \end{cases} \Longrightarrow \begin{cases} K = \overline{S}.R \\ J = S \end{cases}$$

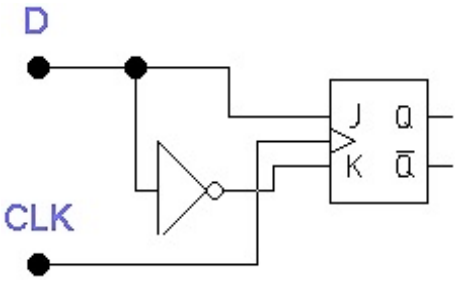
4.3.7.2 D en fonction de JK

$$\begin{cases} Q^+_{D} = D \\ Q^+_{JK} = J.\overline{Q} + \overline{K}.Q \end{cases}$$, on utilise un élement neutre,

$$\begin{cases} Q^+_{D} = D.(Q + \overline{Q}) \\ Q^+_{JK} = J.\overline{Q} + \overline{K}.Q \end{cases} \Longrightarrow \begin{cases} Q^+_{D} = D.\overline{Q} + D.Q \\ Q^+_{JK} = J.\overline{Q} + \overline{K}.Q \end{cases}$$, avec $Q^+_{D} = Q^+_{JK}$ il vient :



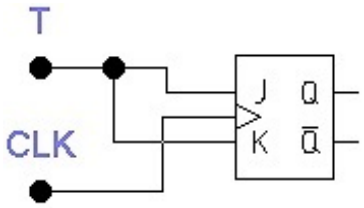
$$\begin{cases} \overline{K} = D \\ J = D \end{cases} \Longrightarrow \begin{cases} K = \overline{D} \\ J = D \end{cases}$$



4.3.7.3 T en fonction de JK

$$\begin{cases} Q_T^+ = T.\overline{Q} + \overline{T}.Q \\ Q_{JK}^+ = J.\overline{Q} + \overline{K}.Q \end{cases}$$, avec $Q_T^+ = Q_{JK}^+$ il vient :

$$\begin{cases} \overline{K} = \overline{T} \\ J = T \end{cases} \Longrightarrow \begin{cases} K = T \\ J = T \end{cases}$$



4.4 Les entrées asynchrones

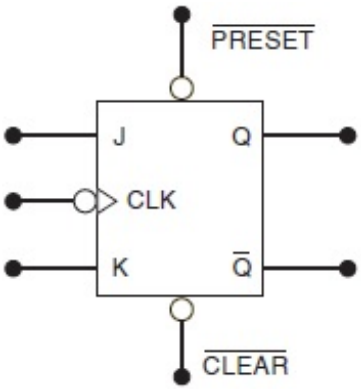
Pour les bascules synchronisées que nous avons vues, les entrées S, R, J, K, D et T sont appelées entrées de commande. Ces entrées sont également appelées entrées synchrones car leur effet sur la sortie de la bascule sont synchronisés avec l'Entrée CLK. Or, les entrées de commande synchrones doivent être utilisées avec un signal d'horloge pour déclencher la bascule.

La plupart des bascules synchronisés ont également une ou plusieurs entrées asynchrones qui fonctionnent indépendamment des entrées synchrones et de l'entrée de l'horloge. Ces entrées asynchrones peuvent être utilisées pour régler la bascule à l'état 1 ou l'effacer (réinitialiser) à l'état 0, à tout mo-

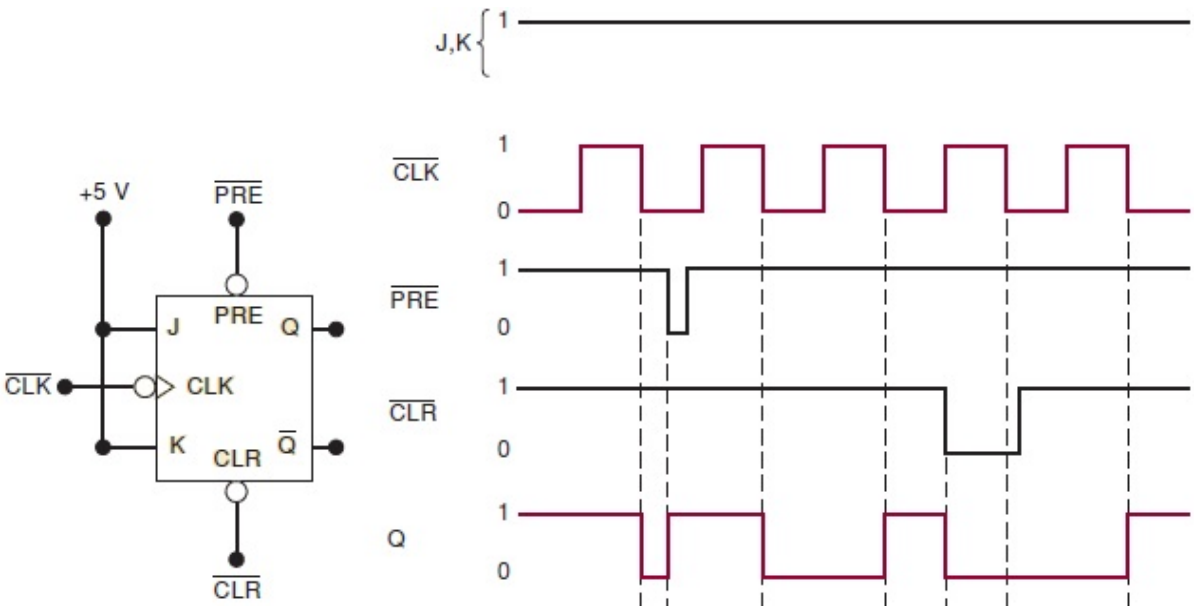
ment quelles que soient les conditions des autres entrées. Autrement dit, les entrées asynchrones sont des entrées de substitution, qui peuvent être utilisées pour remplacer toutes les autres entrées afin de forcer la bascule dans un état ou dans un autre.

La figure ci-dessous montre une bascule JK avec deux entrées asynchrones désignées par $\overline{PRESET}$ ($\overline{PRE}$) et $\overline{CLEAR}$ ($\overline{CLR}$). Ce sont des entrées actives au niveau bas, comme indiqué par les bulles sur la bascule. Le tableau des fonctions associées résume leur incidence sur la sortie de la bascule.

Entrées synch.			*Entrées asynch.*		*Sortie*
J	K	CLK	$\overline{PRE}$	$\overline{CLR}$	Q^+
0	0	↓	1	1	Q (pas de changement)
0	0	↓	1	1	0
0	1	↓	1	1	1
0	1	↓	1	1	$\overline{Q}$ (basculement)
x	x	x	1	1	Q (pas de changement)
x	x	x	1	0	0 (clear asynchrone)
x	x	x	0	1	1 (preset asynchrone)
x	x	x	0	0	ϕ (invalide)



La figure ci-dessous illustre les chronogrammes d'une bascule JK, avec $J = K = 1$ (5 volts) et possède des entrées asynchrones actives au niveau bas.



4.5 Les registres

4.5.1 Définition d'un registre

Un registre est un circuit séquentiel synchrone capable de stocker temporairement des informations binaires à l'aide des bascules. En effet les principales utilisations des registres sont la mémorisation, le décalage et le comptage.

4.5.1.1 La mémorisation

Un registre de mémorisation est un ensemble de mémoires synchrones ayant la même entrée d'horloge et qui permet de stocker momentanément une information à plusieurs bits. La table suivante présente la combinaison des entées des bascules qui réalisent une telle fonction.

Type de bascule	*Entrées*
RS	$R = S = 0$
JK	$J = K = 0$
	$J = \overline{K} = Q$
D	$D = Q$

4.5.1.2 Le décalage

Un registre à décalage permet de transmettre l'état du rang i vers d'une bascule vers une autre bascule de rang $i-1$ ou $i+1$, suivant le type de décalage : à droite ou à gauche.

4.5.1.3 Le comptage (Décomptage)

Un tel registre permet d'incrémenter ou décrémenter la valeur du registre après une impulsion d'horloge.

4.5.2 Les type de décalage

4.5.2.1 Décalage à gauche

L'information du registre est décalée de la droite vers la gauche. Autrement dit, à chaque impulsion d'horloge le contenu de la bascule de rang i est transmis à celle du rang $i+1$. Un tel décalage est équivalent à une multiplication par 2.

	S_4	S_3	S_2	S_1	S_0
Etat initial	1	0	0	1	1

Après imp. CLK	0	0	1	1	*ESG*

ESG : Entrée Série Gauche. S_0 : le bit le moin significatif. S_4 : le bit le plus significatif.

4.5.2.2 Décalage à gauche

L'information du registre est décalée de la gauche vers la droite. Autrement dit, à chaque impulsion d'horloge le contenu de la bascule de rang i est transmis à celle du rang $i-1$. Un tel décalage est équivalent à une division par 2.

	S_4	S_3	S_2	S_1	S_0
Etat initial	1	0	0	1	1
Après imp. CLK	*ESD*	1	0	0	1

ESD : Entrée Série Droite.

4.5.2.3 Décalage circulaire

Pour garder l'information la plus à droite (à gauche), on la relie à la sortie du même circuit, c'est-à-dire à la bascule la plus à gauche (à droite).

Décalage circulaire à gauche

	S_4	S_3	S_2	S_1	S_0
Etat initial	1	0	0	1	1
Après imp. CLK	0	0	1	1	1

Décalage circulaire à droite

	S_4	S_3	S_2	S_1	S_0
Etat initial	1	0	0	1	1
Après imp. CLK	1	1	0	0	1

4.6 Les compteurs

4.6.1 Définition

C'est un dispositif affichant certains états à partir s'une commande extérieure. Un compteur est constitué d'un ensemble de bascules où son état est défini par le nombre binaire formé par la sortie de ces bascules.

Caractéristiques de compteurs

4.6.2 Capacité de comptage

Nombre de bascule	*Capacité*	*Etats possibles*
01	Jusqu'à 1	0,1
02	Jusqu'à 3	00,01,10,11
03	Jusqu'à 7	000,001,...,111
n	Jusquà 2^n-1	-

4.6.3 Comptage / décomptage

Lorsque les états successifs du dispositif représentent un nombre croissant, on parle d'un compteur, sinon ça sera un décompteur.

4.6.4 Fonctionnement synchrone ou asynchrone

4.6.4.1 Compteur synchrone

Dans un tel compteur, l'impulsion de progression est appliquée directement à toutes les entrées d'horloges des bascules. Tandis qu'un circuit combinatoire calcul les fonctions agissant sur les entrées synchrones associées.

4.6.4.2 Compteur asynchrone

Dans la structure asynchrone, le déclenchement s'applique sur la $1^{ère}$ bascule de poids le plus faible. Les autres entrées d'horloge dépendent des sorties des autres bascules.

4.6.5 Type de fonctionnement

Si on considère un compteur constitué de n bascules, alors il possède 2^n états :

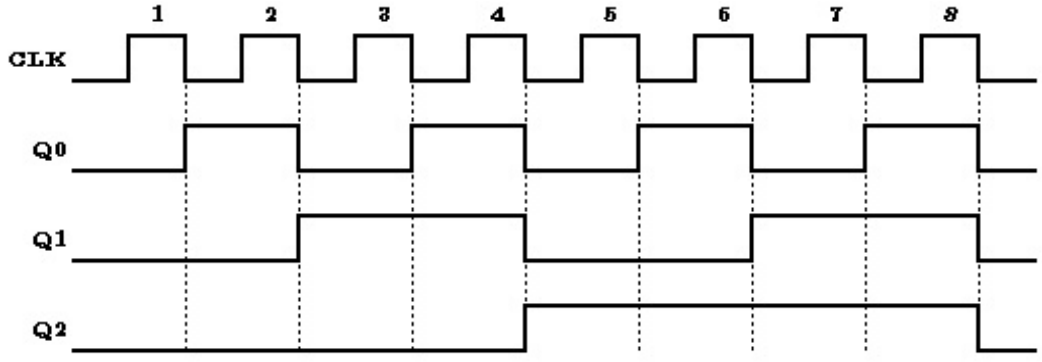
- Si $N = 2^n \Rightarrow$ Fonctionnement permanant : son cycle est complet ;
- Si $N < 2^n \Rightarrow$ Arrêt automatique : son cycle est incomplet.

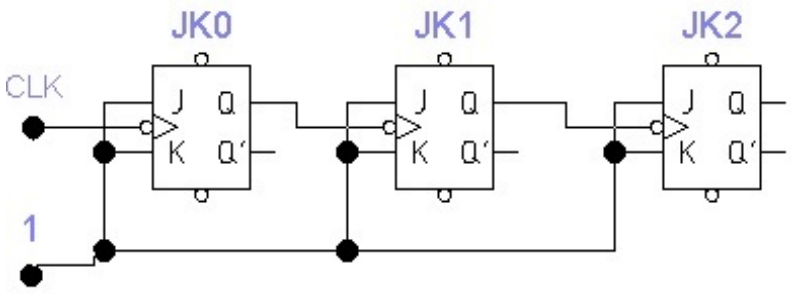
4.6.6 Synthèse des compteurs asynchrones

La synthèse des compteurs asynchrones est assez simple, elle se fait en étudiant, soit la table de vérité ou bien le chronogramme, sachant que l'état initial est $Q_2Q_1Q_0 = 000$.

Exemple : Concevoir un compteur asynchrone modulo 8, en utilisant des bascules JK, synchronisables sur le front descendant de l'horloge.

- Compteur modulo $8 \Rightarrow$ comptage de $0 : 1 : 7$ (000 à 111).
- $8 = 2^3 \Rightarrow$ 3 bascules JK.
- Le chronogramme des sorties est représenté comme suit :





Q_0 change d'état à chaque impulsion d'horloge active au font descendant, d'où $H_0 = H$: $Q_0^+ = \overline{Q_0} = J_0\overline{Q_0} + \overline{K_0}Q_0$, c'est-à-dire $(J_0, K_0) = (1, 1)$.

Q_1 change d'état à chaque impulsion d'horloge active au font descendant, d'où $H_1 = Q_0$: $Q_1^+ = \overline{Q_1}$, de même $(J_1, K_1) = (1, 1)$.

Q_0 change d'état à chaque impulsion d'horloge active au font descendant, d'où $H_2 = Q_1$: $Q_2^+ = \overline{Q_2}$ et $(J_2, K_2) = (1, 1)$.

– Logigramme du compteur :

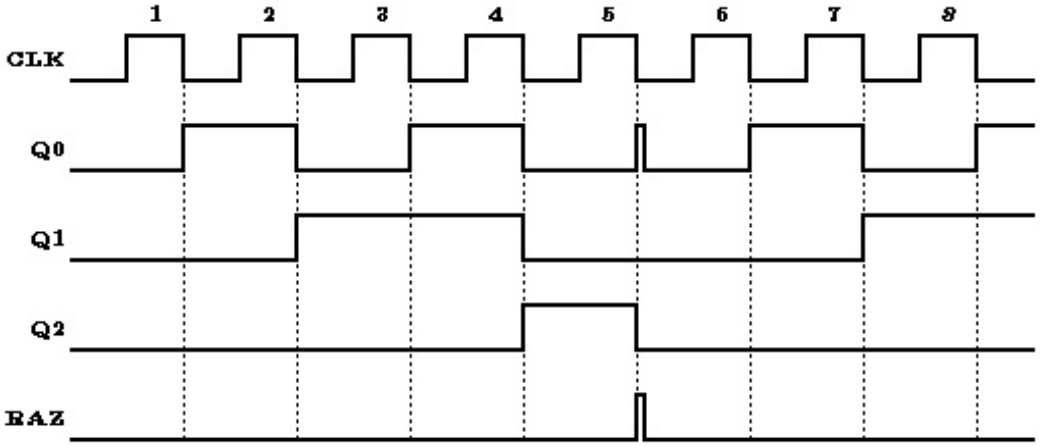
Exemple : Dans cet exemple, on va essayer de concevoir un compteur asynchrone, modulo 5 en binaire pure, à l'aide des bascules JK, sachant que l'entrées de remise à zéro (RAZ) est activée au niveau haut. A l'état initial $Q_2Q_1Q_0 = 000$.

– Compteur modulo 5 $\Rightarrow$ comptage de 0 : 1 : 4 (000 à 100).

– $2^2 < 5 < 2^3 \Rightarrow$ 3 bascules JK.

– L'état indésirable est bien $Q_2Q_1Q_0 = 101$. Donc, Q_2Q_0 doivent être reliées à RAZ.

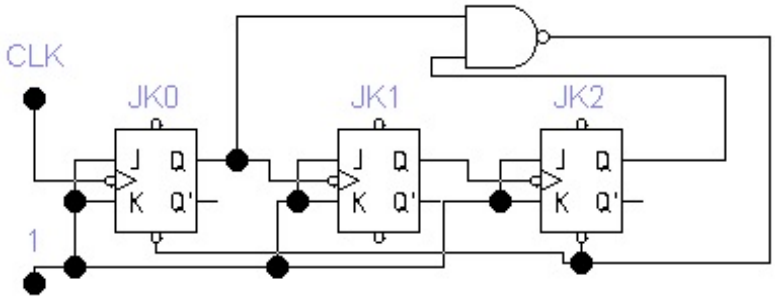
– chronogramme des sorties :

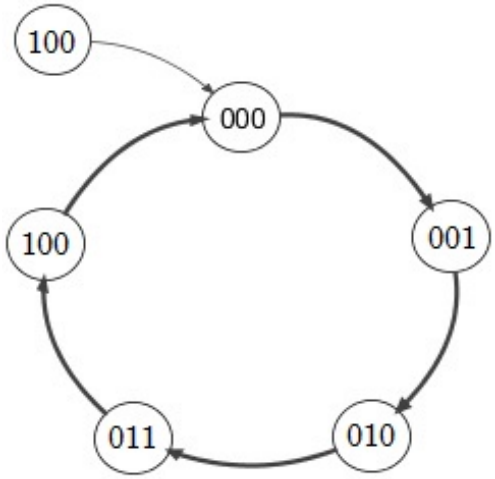


– Logigramme du compteur :

– Si l'état initial était $Q_2Q_1Q_0 = 100$, alors le diagramme de fluence sera donné par :

La suite cyclique est représentée en gras.





4.6.7 Synthèse des compteurs synchrones

Afin d'étudier un tel système, on doit suivre les points suivant :

1. On définit le nombre de bascules nécessaires pour réaliser le compteur, suivant la formule $N \leq 2^n$ (avec n nombre de bascules pour réaliser un compteur modulo N) ;
2. Ecrire la table des séquences successive des bascules A, B, C, ... ;
3. Etablir à partir du diagramme de Karnaught les équations de JK (RS, T, D), en fonction de Q_A, Q_B, ... ;
4. Réaliser le schéma du compteur en fonction des équations obtenues.

Exemple : Réalisation d'un compteur binaire synchrone modulo 8, à l'aide des bascules JK.

La synthèse du réseau logique peut être extraite à partir des déductions tirées directement de la table de vérité. sachant que :

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

$\Leftrightarrow$

Q	Q^+	J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	0	ϕ	0

Déc.	$C\ B\ A$	$C^+B^+A^+$	J_CK_C	J_BK_B	J_AK_A
0	000	001	0 ϕ	0 ϕ	1 ϕ
1	001	010	0 ϕ	1 ϕ	ϕ 1
2	010	011	0 ϕ	ϕ 0	1 ϕ
3	011	100	1 ϕ	ϕ 1	ϕ 1
4	100	101	ϕ 0	0 ϕ	1 ϕ
5	101	110	ϕ 0	1 ϕ	ϕ 1
6	110	111	ϕ 0	ϕ 0	1 ϕ
7	111	000	ϕ 1	ϕ 1	ϕ 1

Les fonctions des entrées des bascules D sont extraites à partir de la table de Karnaught, comme suit :

J_A	BA			
C	00	01	11	10
0	1	ϕ	ϕ	1
1	1	ϕ	ϕ	1

$J_A = 1$

K_A	BA			
C	00	01	11	10
0	ϕ	1	1	ϕ
1	ϕ	1	1	ϕ

$K_A = 1$

J_B	BA			
C	00	01	11	10
0		1	ϕ	ϕ
1		1	ϕ	ϕ

$J_B = A$

K_B	BA			
C	00	01	11	10
0	ϕ	ϕ	1	
1	ϕ	ϕ	1	

$K_B = A$

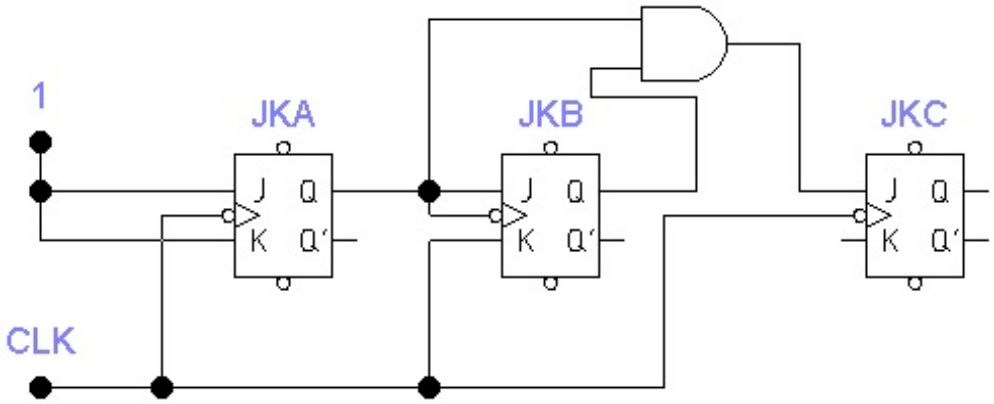
J_C	BA			
C	00	01	11	10
0			1	
1	ϕ	ϕ	ϕ	ϕ

$J_C = AB$

K_C	BA			
C	00	01	11	10
0	ϕ	ϕ	ϕ	ϕ
1			1	

$K_C = AB$

Schéma du compteur :



Exemple : Dans cet exemple, on va réaliser un compteur binaire synchrone modulo 10, à l'aide des bascules D.

La table de vérité qui comprend l'état présent, l'état suivant et les entrées des bascules est donnée par :

Déc.	$DCBA$	$D^+C^+B^+A^+$	$D_D D_C D_B D_A$
0	0000	0001	0001
1	0001	0010	0010
2	0010	0011	0011
3	0011	0100	0100
4	0100	0101	0101
5	0101	0110	0110
6	0110	0111	0111
7	0111	1000	1000
8	1000	1001	1001
9	1001	0000	0000

On sait que :

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

Les fonctions des entrées des bascules D sont extraites à partir de la table de Karnaught, comme suit :

D_D	BA			
DC	00	01	11	10
00				
01			1	
11	ϕ	ϕ	ϕ	ϕ
10	1		ϕ	ϕ

$D_D = \overline{A}D + ABC$

D_C	BA			
DC	00	01	11	10
00			1	
01	1	1		1
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

$D_C = AB \oplus C$

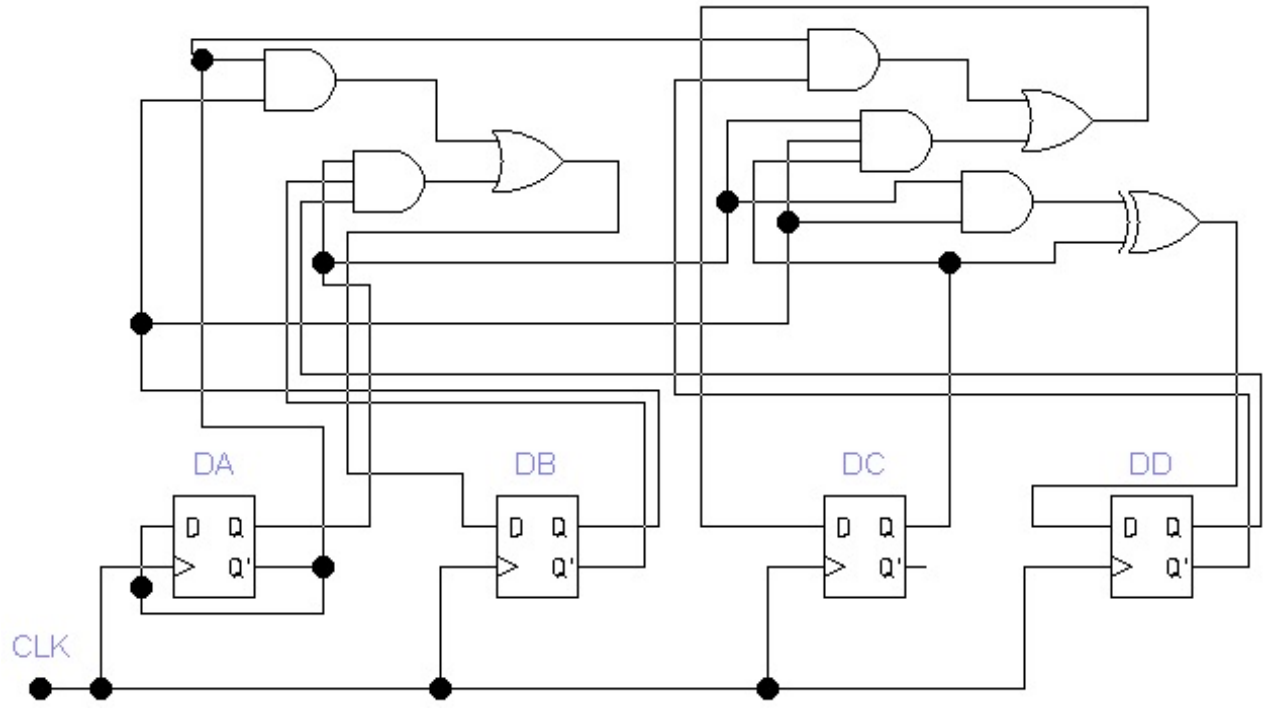
D_B	BA			
DC	00	01	11	10
00		1		1
01		1		1
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

$D_B = \overline{A}B + A\overline{B}\ \overline{D}$

D_A	BA			
DC	00	01	11	10
00	1			1
01	1			1
11	ϕ	ϕ	ϕ	ϕ
10	1		ϕ	ϕ

$D_A = \overline{A}$

Schéma du compteur :


DA
DB
DC
DD
D Q
Q'
CLK

Chapitre 5

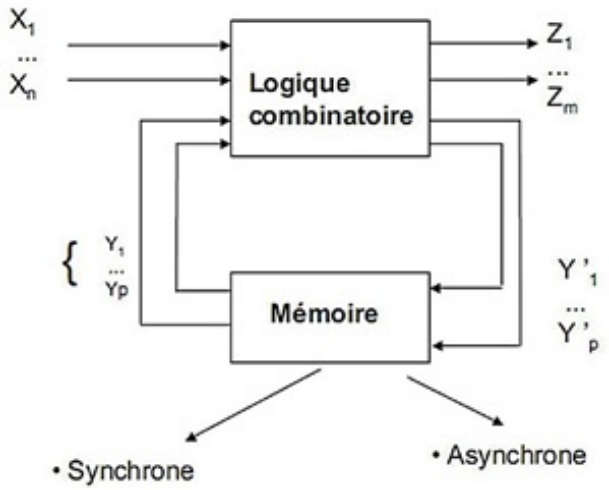
Machines à états finis

5.1 Définition

Le présent chapitre va nous servir comme introduction pour modéliser les systèmes séquentiels asynchrones et synchrone et choisir ainsi le type de machine nécessaire qu'il faut utiliser, afin d'entamer la synthèse par la suite.

Une machine à état finis (ou machine d'état tout court) est une description algorithmique qui modélise tout un circuit séquentiel par un vecteur d'entrée, un vecteur de sortie, en plus d'une séquence d'états définissant son comportement.

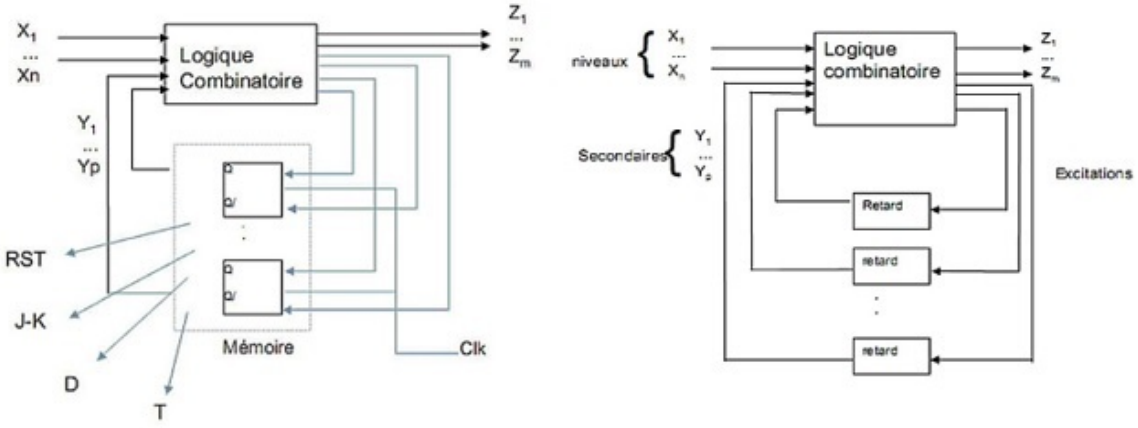
La machine (également appelée automate) va passer d'un état à un autre suivant les séquences d'entrée qu'elle reçoit. On attribue généralement à la machine un état de départ qui lui permet de débuter son fonctionnement à partir d'un point fixe.



5.2 Types des machines Séquentielles

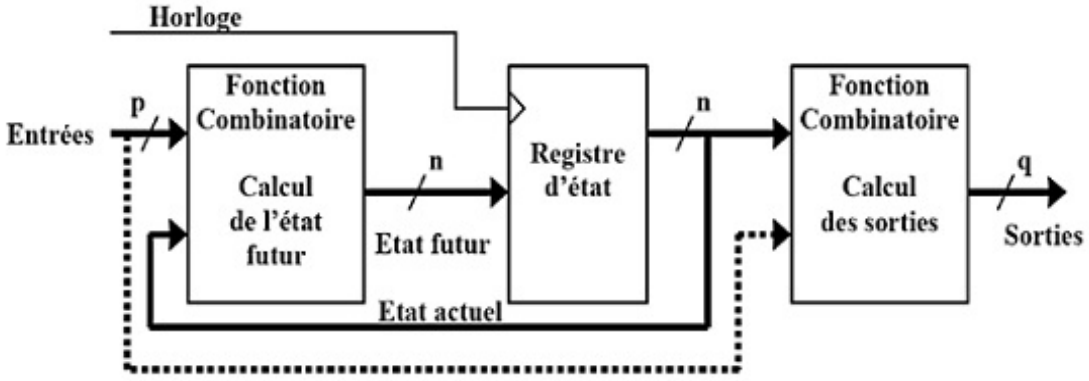
Dans le mode synchrone, les éléments de mémorisation sont bien des bascules. Les modifications d'état du système ne peuvent intervenir qu'à des instants bien précis déterminés par des signaux d'horloge.

Or, dans le mode asynchrone, la fonction de mémorisation est réalisée par des simples boucles de rétroaction. Donc l'évolution des états ne dépend que des modifications intervenant sur les entrées primaires (X_i) de la machine.



Remarque : La figure de gauche représente le modèle généralisé d'une machine séquentielle synchrone et celle de droite celui d'une machine séquentielle asynchrone.

5.3 Les machines synchrones à nombre d'états fini



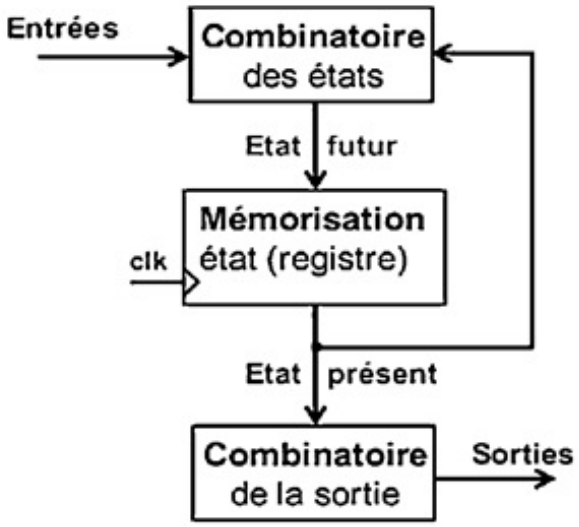
Une machine à états ou Finite State Machine (FSM) comme le montre la figure ci-haut, est un système dynamique, qui peut se trouver, dans une position parmi un nombre fini de positions possibles, à chaque instant. La machine en question traverse des cycles, en changeant éventuellement d'état lors des transitions actives de l'horloge.

5.4 Différentes Architectures de La FSM

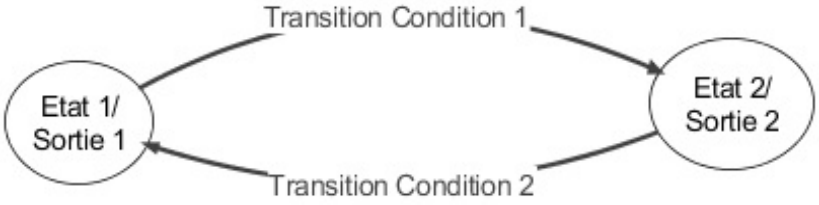
L'architecture générale d'une machine à états déjà présentée montre que, selon la façon dont les sorties dépendent des états et des commandes, il y'a 02 types de machines à états : la machine de Moore, ainsi que celle de Mealy.

5.4.1 Machine de Moore

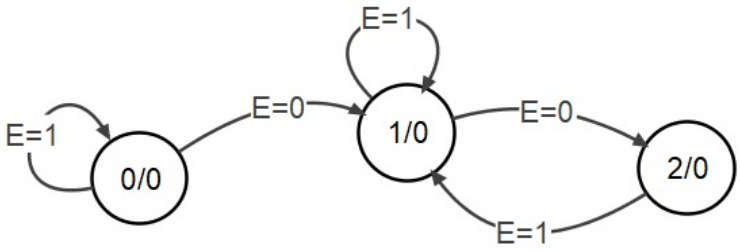
Les sorties d'une machine de Moore dépendent seulement de l'état présent (synchrones, elles changent sur un front d'horloge). On retrouve son état futur à partir des entrées et de l'état présent.



Le graph d'état est représenté comme le montre la figure ci-bas :

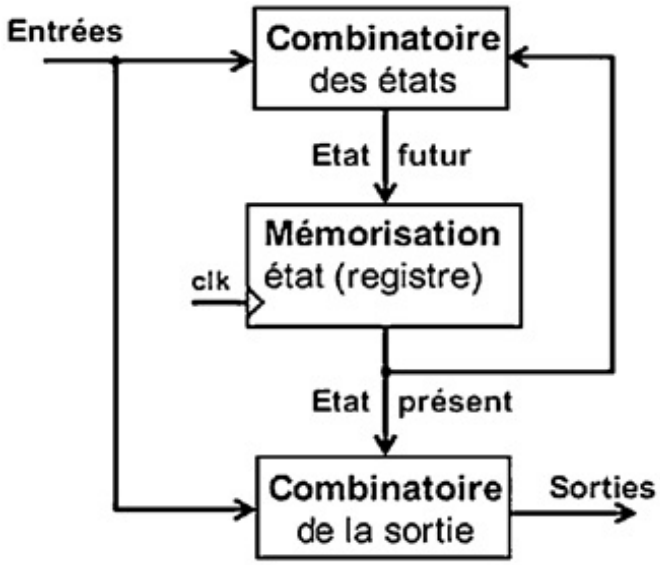


Exemple : Reconnaissance de la séquence *10* par la machine de Moore.

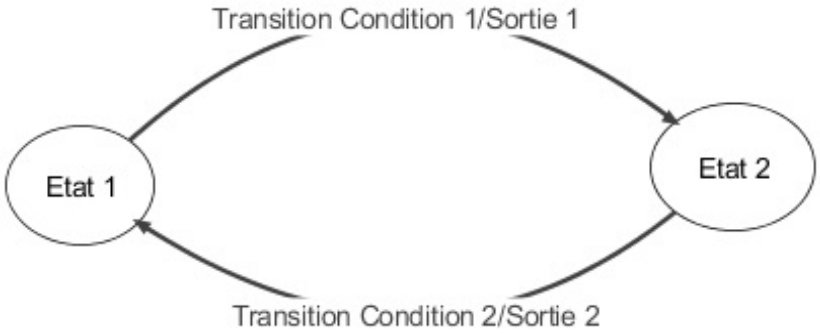


5.4.2 Machine de Mealy

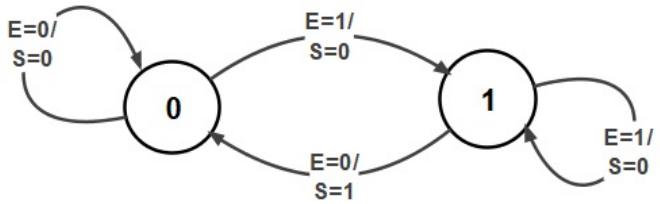
Les sorties d'une machine de Mealy dépendent de l'état présent en plus des entrées. Comme dans la machine de Moore, on retrouve son état futur à partir des entrées et de l'état présent. Mémorisation synchrone des états (càd sur un front d'horloge). La sortie dépend directement de l'entrée et indépendamment de l'horloge (Clk) d'où une sortie asynchrone. C'est-à-dire qu'elle comporte un nombre d'états plus réduit que celui d'une machine de Moore.



Le graph d'état est représenté comme le montre la figure ci-bas :



Exemple : Reconnaissance de la même séquence précédente *10* par la machine de Mealy.



5.5 Comparaison entre les deux machines

- A toute machine de Mealey Correspond une machine de Moore et réciproquement ;
- Une telle équivalence ne s'applique qu'aux séquences d'entrée et de sortie ;
- Les deux machines n'ont pas le même comportement temporel ;
- Dans une machine de Mealey, une variation des entrées entraîne immédiatement une variation des sorties. Alors que dans une machine de Moore, les variations des sorties n'interviennent qu'aux moments des variations d'état ;
- Les Machines de Mealey sont rapides, et économiques par rapport à celles de Moore (moins d'état). Or, Les machines de Moore sont plus simples à concevoir.

Chapitre 6

Les systèmes séquentiels asynchrones

6.1 Introduction

Contrairement aux circuits séquentiels synchrones, les circuits séquentiels asynchrones ne sont pas concernés par la référence de temps, c'est à dire que l'action sur les entrées est prise en compte dès leur changement d'état, car, Dans beaucoup de situations pratiques, les entrées du système à réaliser ne sont pas commandées par une horloge et sont donc susceptibles sujettes à modification à des instants divers.

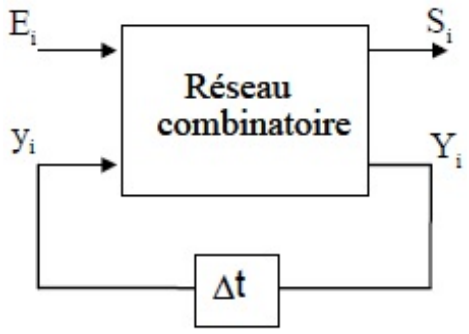
6.2 Structure d'un système séquentiel asynchrone

Dans le mode synchrone, on utilise des bascules comme éléments de mémorisation. Or, dans le mode asynchrone, la mémorisation est réalisée par des boucles de rétroaction. Le changement des états ne dépend donc que des modifications surgissant sur les entrées primaires (E_i) de la machine. La représentation d'un tel système est donné par la figure qui suit :

Δt est un élément virtuel permettant de modéliser le retard des signaux en logique combinatoire.

6.3 But d'un système séquentiel asynchrone

L'objectif d'un tel système est, à partir d'un cahier de charge, de réaliser par un coût minime, le montage correspondant. La réalisation du montage en question se fait par la recherche des équations du



système, ainsi que la détermination du nombre minimum de fonctions secondaires. Or, pour effectuer la synthèse d'un système séquentiel, on utilisera la méthode matricielle dite d'Hoffman.

Définitions

- Etat stable : On dit qu'un état est stable si le système reste inchangé tant qu'il garde les mêmes entrées qui correspondent au régime permanant. Autrement dit : $Y_i = y_i$ (i=1,2,3,...,k), avec k entier ;
- Etat transitoire : C'est l'état qui caractérise un système lors de son passage d'un état stable à un autre état stable, c'est à dire que $Y_i \neq y_i$ (i=1,2,3,...,k), avec k entier.

Remarques

- On notera un état stable avec un chiffre entouré par un cercle (①, ②, ...), et un état transitoire avec chiffre simple (1, 2, ...) ;
- On peut concevoir que l'état initial et l'état final d'un système soient le même.
- Le principe de la machine de Mealey ou Moore, dans le cas des systèmes séquentiels asynchrones n'existe plus. En effet, et de manière combinatoire, les états dépendent directement des entrées et les sorties dépendent des entrées, que ce soit directement ou indirectement en fonction des variables secondaires.

6.4 La synthèse par la méthode d'Hoffman

Le but principal de la synthèse des systèmes séquentiels asynchrones est la minimisation du nombre de variables secondaires (Y_i). D'où la réduction du nombre de fonctions du circuit combinatoire. Cette méthode se déroule suivant des étapes successives comme suit :

- Spécifier le graphe d'états ;
- Matrice primitive des états ;
- Déduction de la table des transitions primitive ;
- Eliminer des états équivalents par le graphe de fusionnement ou le polygone de liaison ;
- Déduire la table des états réduite ;
- Coder les états internes (variables secondaires) ;
- Synthèse des variables secondaires ;
- Extraire les équations logiques ainsi que le schéma en portes.

6.5 Commande d'un moteur

On prend l'exemple d'un système séquentiel correspondant à un moteur M, commandé par deux boutons poussoirs, a (arrêt) et m (marche). Notez bien qu'on ne peut actionner a et m simultanément.

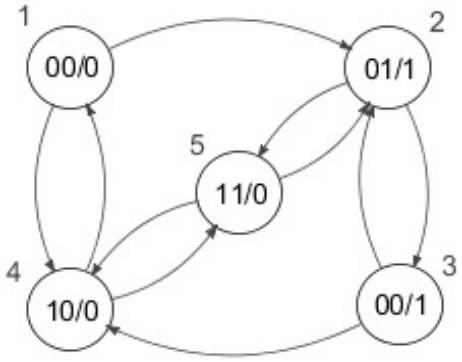
La traduction du cahier des charges est résumée dans la table correspondante aux entrées/sorties suivantes :

Entrées			*Sortie*
Action	a	m	M
Position repos	0	0	0
Activation de m	0	1	1
Relâchement de m	0	0	1
Activation de a	1	0	0
Relâchement de a	0	0	0
Activation de a et de m	1	1	0

Pour les variables d'entrées, nous avons deux boutons de commande, a et m et pour les fonctions de sorties, le moteur M (pour mise en marche).

6.5.1 Le graphe d'états

Ce diagramme représente chaque état par un cercle qui contient les valeurs des *entrées/sorties*. Le passage entre deux état s'effectue par une flèche allant de l'état présent à l'état futur.



6.5.2 Matrice primitive des états

C'est une matrice caractérisée par :

1. Le nombre de colonnes égal au nombre de combinaisons des variables d'entrées ;
2. Le nombre de lignes égal eu nombre des états stables du système ;
3. Les états stables figurent dans la colonne des combinaisons des variables d'entrées qu'il a engendrée ;
4. Les états transitoires figurent dans la même colonne que son état stable ; et la sortie sera représentée sur une colonne séparée avec autant de lignes que d'états stables.

5. Le passage d'un état à un autre est provoqué par le changement d'une seule variable à la fois, et les transitions impossibles sont négligées ;
6. Il n'y a qu'un seul état stable dans une ligne.

	$a\ m$				*Sortie*
Etat	00	01	11	10	M
1	①	2	-	4	0
2	3	②	5	-	1
3	③	2	-	4	1
4	1	-	5	④	0
5	-	2	⑤	4	0

Les états stables ① et ③ correspondent à la même combinaison des variables d'entrées (00), mais donnent des états de sorties différentes 0 et 1 respectivement. On ne peut donc les différenciés par une seule combinaison des variables d'entrées, alors on fait appel à une variable secondaire.

6.5.3 Polygone de liaison

Afin de faire les liaisons entre les sommets, on doit comparer les lignes de la matrice primitive, 2 par 2 et voir s'il y a compatibilité.

La règle de fusionnement de deux ou plusieurs lignes ne concerne que les états de mêmes sorties. Cette fusion ne pourrait se fait que si elle vérifie colonne par colonne les conditions indiquées dans la table suivante :

X	$-$	X	$-$
X	X	$-$	$-$
↓	↓	↓	↓
X	X	X	$-$

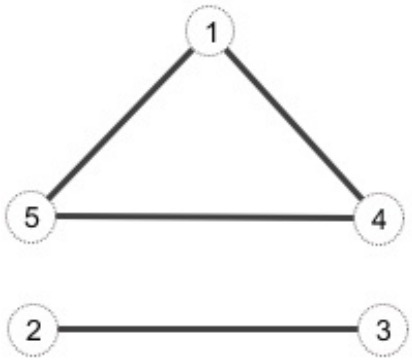
X : Etat de même chiffre, stable ou transitoire.

Les liaisons compatibles sont donc : (1,4), (1,5), (2,3), (2,5), (3,5) et (4,5). Le polygone de liaison sera représenté comme suit :

On peut retenir la solution : **(1,4,5)** et **(2,3)**.

6.5.4 Table d'états réduite

Le polygone de liaison permet d'obtenir le minimum de variables secondaires (table ci-bas à gauche). On introduit ainsi une variable secondaire y pour différencier les deux lignes (table ci-bas à droite), il vient :



	$a\ m$				*Sortie*
Etat	00	01	11	10	M
1,4,5	①	2	⑤	④	0
2,3	③	②	5	4	1

	$a\ m$				*Sortie*
y	00	01	11	10	M
0	①	2	⑤	④	0
1	③	②	5	4	1

6.5.5 Variables secondaires

La matrice contractée est remplie de la manière suivante :

- Tout état stable prend la valeur de la variable secondaire correspondante à la ligne où il est situé ;
- Tout état transitoire (instable), prend la même valeur que son état stable. il vient :

Y	$a\ m$			
y	00	01	11	10
0		1		
1	1	1		

$$Y = \overline{a}.(m + y)$$

6.5.6 Equation de la fonction de sortie S

Il existe une matrice par fonction de sortie, elle à la même disposition de la matrice contractée (réduite ou ordonnée). Elle est définie comme suit :

- Tout état stable repéré par un cercle, est remplacé par la valeur de la sortie qui lui correspond ;
- Pour l'état instable, la matrice contractée peut être remplie de deux manières :

  1. Avec condition : l'état transitoire prend la valeur de la sortie correspondante à l'état stable ;
  2. Sans condition : les états transitoires prendront la valeur indéterminée.

Y	$a\ m$			
y	00	01	11	10
0		ϕ		
1	1	1	ϕ	ϕ

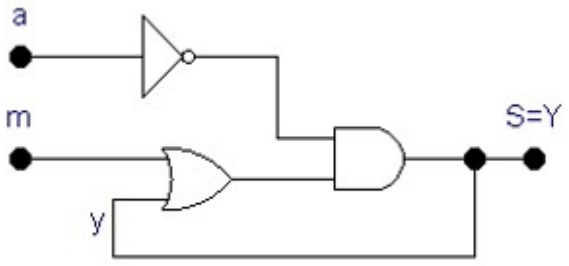
Etat instable sans condition

Y	$a\ m$			
y	00	01	11	10
0		1		
1	1	1		

Etat instable avec condition

En se basant sur le $2^{ème}$ cas (avec condition), l'équation de sortie S sera donnée par :

$S = Y = \overline{a}.(m + y)$. Alors le schéma de réalisation sera représenté comme suit :

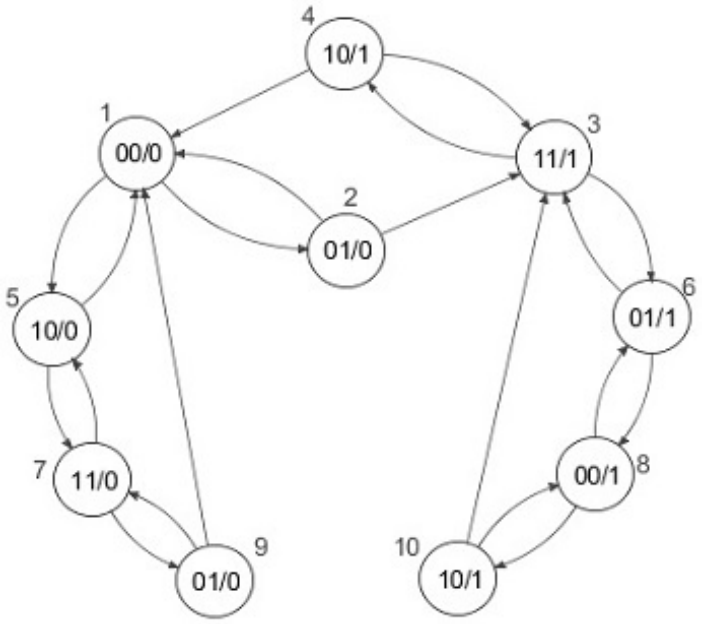


6.6 Ouverture d'une serrure électronique

Afin de bien éclaircir la synthèse d'un système séquentiel asynchrone, on va prendre un autre cas plus complexe.

Une serrure électronique comporte deux entrées x_1, x_2 et une sortie z, déclenchant l'ouverture. Cette dernière prend la valeur 1 à la fin de la séquence $x_1x_2 = 00 \rightarrow 01 \rightarrow 11$, et garde cette valeur, jusqu'à l'apparition de la séquence : $x_1x_2 = 11 \rightarrow 10 \rightarrow 00$. A la fin de celle-ci z reprend la valeur nulle.

Le graphe d'états

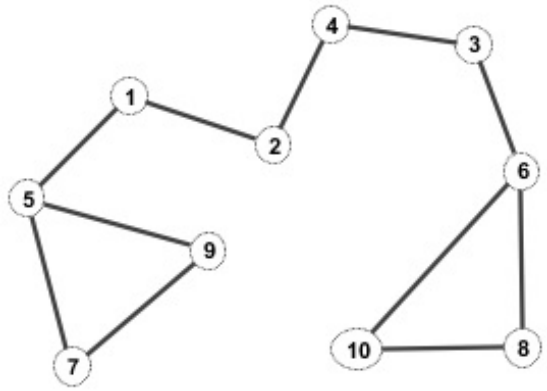


Matrice primitive des états

	$x_1\ x_2$				*Sortie*
Etat	00	01	11	10	z
1	①	2	-	5	0
2	1	②	3	-	0
3	-	6	③	4	1
4	1	-	3	④	1
5	1	-	7	⑤	0
6	8	⑥	3	-	1
7	-	9	⑦	5	0
8	⑧	6	-	10	1
9	1	⑨	7	-	0
10	8	-	3	⑩	1

Polygone de liaison

Le polygone de liaison sera représenté comme suit :



On peut retenir la solution : **(1,2), (3,4), (5,7,9)** et **(6,8,10).**

Table d'états réduite

Les états instables sont reportés sur les états stables qui ont même chiffre. On peut alors déduire la table réduite :

	x_1x_2				*Sortie*
Etat	00	01	11	10	z
1,2	①	②	3	5	0
3,4	1	6	③	④	1
5,7,9	1	⑨	⑦	⑤	0
6,8,10	⑧	⑥	3	⑩	1

$\Rightarrow\ VS\ \Rightarrow$

	x_1x_2				*Sortie*
y_1y_2	00	01	11	10	z
00	①	②	3	5	0
01	1	6	③	④	1
11	1	⑨	⑦	⑤	0
10	⑧	⑥	3	⑩	1

Variables secondaires

Le nombre de variables secondaires est déterminé à partir du nombre de lignes de la matrice contractée : 4 *lignes* $= 2^2$ *lignes* $\Rightarrow$ 2 *variables secondaires*.

La matrice contractée est remplie de la manière suivante :

	x_1x_2			
y_1y_2	00	01	11	10
00	⓪⓪	⓪⓪	01	11
01	00	10	⓪①	⓪①
11	00	⓪①	⓪①	⓪①
10	⓪⓪	⓪⓪	01	⓪⓪

En séparant les variables secondaires dans deux tables différentes, il vient :

Y_1	x_1x_2			
y_1y_2	00	01	11	10
00				1
01		1		
11		1	1	1
10	1	1		1

$Y_1 = \overline{x_1}y_1\overline{y_2} + \overline{x_1}x_2y_2 + x_1\overline{x_2}y_1 + x_1\overline{x_2}\ \overline{y_2}$

Y_2	x_1x_2			
y_1y_2	00	01	11	10
00			1	1
01			1	1
11		1	1	1
10			1	

$Y_2 = x_2y_1y_2 + x_1x_2\overline{y_2} + x_1y_2 + x_1\overline{y_1}\ \overline{y_2}$

Equation de la fonction de sortie z

L'état instable est choisir d'une manière conditionnelle, et la fonction de sortie est déduite de la table suivante :

z	x_1x_2			
y_1y_2	00	01	11	10
00			1	
01		1	1	1
11				
10	1	1	1	1

$z = y_1\overline{y_2} + \overline{y_1}y_2(x_1 + x_2) + \overline{y_1}x_1x_2$

Bibliographie

[1] Mosser., Granjon, Y. & Tanoh, J. (2008). *Sciences industrielles pour l'ingénieur 1re année MPSI-PCSI-PTSI*. Paris : Dunod.

[2] Strandh, R. & Durand, I. (2005). *Architecture de l'ordinateur : portes logiques, circuits combinatoires, arithmétique binaire, circuits séquentiels et mémoires. Exemple d'architecture*. Paris : Dunod.

[3] Ndjountche, T. (2016). *Digital electronics*. London, UK Hoboken, NJ : ISTE, Ltd. Wiley.

[4] Tocci, R., Widmer, N. & Moss, G. (2007). *Digital systems : principles and applications*. Upper Saddle River, N.J : Pearson Prentice Hall.

[5] Maini, A. (2007). Digital electronics : principles, devices and applications. Chichester, England Hoboken, NJ : John Wiley & Sons.