SOMMAIRE

Plan de cours

AUTOMATIQUE

Cours : Automatique

Profil : Génie Mécanique

Option : Toutes filières : Energétique, Electro-mécanique, Construction et Fabrication Mécanique

Niveau cible *:* Etudiants du niveau 3

Volume horaire *:* 22 h30 CI et 22 h30 TP

Coefficient : 2

Pré-requis :

- Notions mathématiques
- Informatique

Évaluation *:* 1 test, 1 DS et un examen final écrits.

But du cours :

Comprendre les principes de fonctionnement des circuits combinatoires et circuits séquentiels.

Ce plan est extrait à partir du livret du programme officiel des ISET, paru en juin 2004.

Chaque **objectif général** lui correspond des **objectifs spécifiques**.

OBJECTIFS GÉNÉRAUX		**CONDITION DE RÉALISATION DE LA PERFORMANCE**	**CRITÈRES D'ÉVALUATION DE LA PERFORMANCE**
OG1	Connaître les principes de base de la logique combinatoire	En se basant sur les pré-requis vus en secondaire, le cours et les exercices élaborés en classe, l'étudiant doit connaître les principes de base de la logique combinatoire.	L'étudiant doit appliquer ces principes de base sans commettre des erreurs.
OG2	Connaître les principes de base de la logique séquentielle.	En se basant sur les pré-requis vus en secondaire, le cours et les exercices élaborés en classe, l'étudiant doit connaître les principes de base de la logique séquentielle	L'étudiant doit appliquer ces principes de base sans commettre des erreurs.

Afin de mettre en oeuvre ces objectifs, une ou plusieurs **leçons** est nécessaire assistée généralement par un ou plusieurs **TP** mettant en pratique les notions mises à disposition pour l'étudiant dans une leçon. Un ou plusieurs **TD** est proposé pour permettre aux étudiants de mieux assimiler à travers des séries d'exercices le cours :

Énoncé de l'objectif général 1 : Connaître les principes de base de la logique combinatoire					
Codes	**Objectifs spécifiques**	**Eléments de contenu**	**Méthodologies et moyens**	**Evaluation**	**Durée**
OS1	Identifier les systèmes de numération	- Système binaire - Système hexadécimal - Applications	Exposé informel Tableau	Formative	1h30 CI
OS2	Identifier les codes	- Codes pondérés (binaire naturel) - Codes non pondérés (binaire réfléchi) - Codes alphanumériques - Applications	Exposé informel Tableau	Formative	1h30 CI
OS3	Reconnaître les fonctions logiques	- Les opérateurs logiques - Les fonctions logiques - Théorème de DEMORGAN - Applications	Exposé informel Tableau	Sommative	4h30 CI
OS4	Comprendre la simplification des fonctions logiques	- Simplifications algébriques des expressions logiques - Simplification graphique par tableau de KARNAUGH - Applications	Exposé informel Tableau	Sommative	4h30 CI
OS5	Reconnaître les circuits logiques combinatoires	- Codeur et décodeur - Multiplexeur et démultiplexeur - Comparateur et additionneur	Exposé informel Tableau	Sommative	4h30 CI

Moyens de mise en œuvre :

Leçon 1: Systèmes de numération et codes
TD 1

Leçon 2: Logique Booléenne
TD 2

Leçon 3: Simplification des fonctions logiques
TD 3

Leçon 4: Circuits logiques combinatoires
TD 4

Énoncé de l'objectif général 2 : Connaître les principes de base de la logique séquentielle					
Codes	**Objectifs spécifiques**	**Eléments de contenu**	**Méthodologies et moyens**	**Evaluation**	**Durée**
OS1	Simuler les fonctions logiques séquentielles	- Introduction aux systèmes séquentiels - Les bascules utiles RS, JK, D et T - Recherche des équations logiques	Exposé informel Tableau	Formative	2h30
OS2	Comprendre le principe de fonctionnement des circuits séquentiels	- Registre - Compteur	Exposé informel Tableau	Sommative	2h
OS3	Synthétiser les systèmes séquentiels avec des bascules	- Synthèse des systèmes synchrones et asynchrones avec des bascules	Exposé informel Tableau	Sommative	1h30

Moyens de mise en œuvre :

Leçon 5 : Système logique séquentiel
TD5

Leçon 6: Circuits logiques séquentiels
TD 6

Chapitre I

Systèmes de numération et codes

Objectifs

A la fin de la séance l'étudiant doit être capable de :

- Introduire la numération binaire
- Etudier les principaux codes utilisés dans les systèmes numériques

Pré requis

L'étudiant est supposé connaître :

- Les outils mathématiques
- Informatique

Durée

Deux séances de 1h30min

Eléments de contenu

- Les différentes bases de numération
- Conversion des bases
- Les opérations dans les bases
- Représentations par les codes

Chapitre I:

Systèmes de numération et codes

I Introduction

Aux premiers temps de l'informatique, toute la difficulté pour un être humain voulant « dialoguer » avec une machine informatique résidait dans le fait que le seul langage « parlé »par la machine était le langage binaire. Actuellement, le seul langage « parlé » par une machine informatique est toujours le langage binaire mais d'autres langages évolués sont apparus qui facilitent le dialogue homme-machine.

Dans le cadre particulier de ce cours, nous allons principalement nous intéresser aux passerelles existant entre 3 systèmes de numération: le système décimal (base10), le système binaire (base2) et le système hexadécimal (base 16).

II Les différents systèmes de numération

II.1 Système décimal

Ce système de numération, usuel dans la vie quotidienne, est construit à partir de 10 éléments de base constitués par les 10 chiffres universellement connus: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 : on dit ainsi que le système décimal est en base 10.

Tout nombre entier représenté par une suite de symboles :

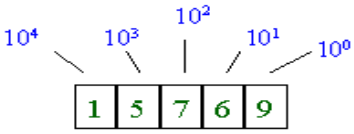
$$(N)_{10} = a_n a_{n-1} \ldots . a_1 a_0$$

Est donné sous forme d'une combinaison linéaire de puissance de 10 :

$$(N)_{10} = a_n \times 10^n + a_{n-1} \times 10^{.n-1} + \ldots . + a_1 \times 10^1 + a_0 \times 10^0$$

Avec $\begin{cases} a_0 \text{ est le chiffre de poids le plus faible} \\ a_n \text{ est le chiffre de poids le plus fort} \end{cases}$

Par exemple :

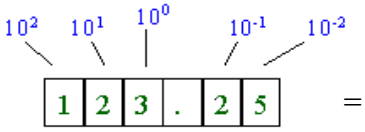


$= 10000 + 5000 + 700 + 60 + 9$

$\mathbf{(15769)}_{10} = 9 \times 10^0 + 6 \times 10^1 + 7 \times 10^2 + 5 \times 10^3 + 1 \times 10^4$

Ce qui est vrai pour les entiers est vrai aussi pour les réels, selon le même principe.

Par exemple :



$= 100 + 20 + 3 + 0{,}2 + 0{,}05$

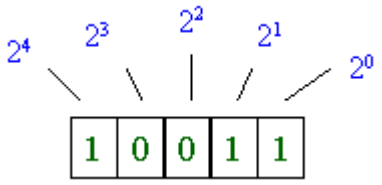
$\mathbf{123.35} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$

II.2 Système binaire

Le système binaire est construit à partir de 2 éléments : 0 et 1 (base 2). C'est la base '' naturelle'' dans le domaine de l'automatisme, de l'électronique et de l'informatique, en effet, travaillant à partir des données logiques 0 ou 1, il est normal d'utiliser la base 2 pour la représentation.

Ce système binaire est construit à partir de 2 éléments : 0 et 1 (base 2). N'importe quel chiffre binaire est décomposable en puissance de 2.

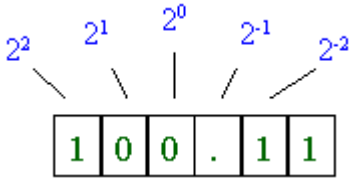
Par exemple :



10011 = 1 x 2^0 + 1 x 2^1 + 0 x 2^2 + 0 x 2^3 + 1 x 2^4

Ce qui est vrai pour les entiers est vrai aussi pour les réels, selon le même principe.

Par exemple :



100.11 = 1 x 2^2 + 1 x 2^{-1} + 1 x 2^{-2}

II.3 Système Octale

Ce système utilise 8 symbole : 0,1,2,3,4,5,6,7. Il n'est plus guère employé aujourd'hui, puisqu'il servait au codage des nombres dans les ordinateurs de première génération.

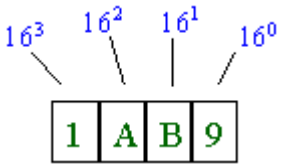
II.4 Système hexadécimal

Le développement des systèmes micro programmés (mini et micro-ordinateurs) a favorisé l'utilisation de système hexadécimal.

Ce système hexadécimal est construit à partir des 10 éléments de base du système décimal (0…9) + les 6 premières lettres de l'alphabet (A, B, C, D, E, F), chaque lettre étant équivalente à une grandeur décimale (A = 10, B = 11, C = 12, D = 13, E = 14 et F = 15).

N'importe quel chiffre hexadécimal est donc décomposable en puissance de 16.

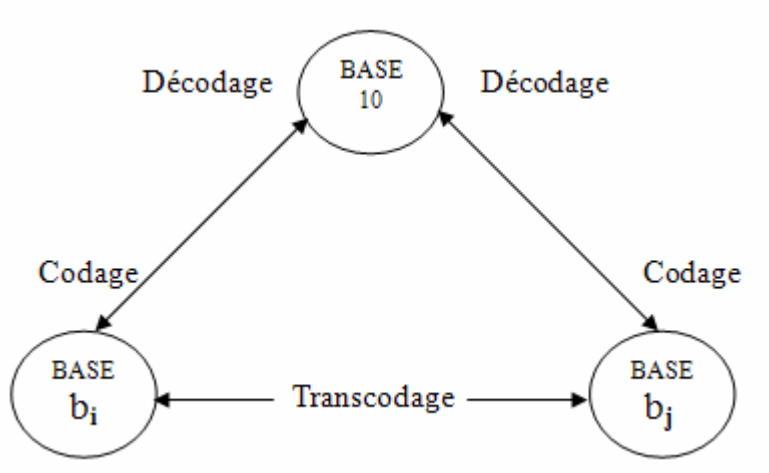
Par exemple :



1AB9 = 1 x 16^3 + A x 16^2 + B x 16^1 + 9 x 16^0

III Conversions des bases

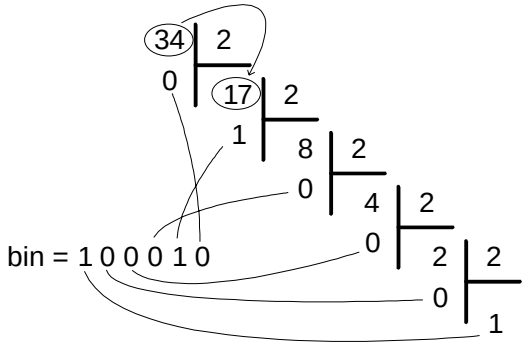
Il y a 3 types de conversion d'une quantité représentée dans une base bi en une autre base bj



III.1 Conversion du le système décimal ↔ système binaire

Le passage du système décimal vers le système binaire repose sur un ensemble de divisions successives par 2. Ce sont les restes de ces divisions qui, combinés, permettent de reconstituer l'équivalent en base 2 du nombre décimal :

Exemple n°1: $34_{(10)} = ?_{(2)}$



$34_{(10)} = 100010_{(2)}$

Conversion des nombres comportant une partie fractionnaire, exemple : $(462,4375)_{10} = (?)_2$

Pour résoudre cette conversion, on procède comme suit :

- Convertir la partie entière (462) comme s'est indiqué ci-dessus,
- Convertir la partie fractionnaire en faisant des multiplications successives par la base en conservant à chaque fois le chiffre devenant entier.

$(462)_{10} = (111001110)_2$

2^0	0	'0',4375
		* 2
2^{-1}	0	'0',8750
		* 2
2^{-2}	1	'1',75
		0,75
		* 2
2^{-3}	1	'1',5
		0,5
		* 2
2^{-4}	1	'1',0

Donc: $0,4375_{(10)} = 0,0111_{(2)}$

Donc $(462.4375)_{10} = (111001110.0111)_2$

Parfois en multipliant la partie décimale par la base, on n'arrive pas à convertir toute la partie entière. Ceci est du essentiellement au fait que le nombre à convertir n'a pas un équivalent exact dans la base B et sa partie décimale est cyclique.

Exemple : 0.15 x 2 = 0.3

0.3 x 2 = 0.6

0.6 x 2 = 1.2

0.2 x 2 = 0.4

0.4 x 2 = 0.8

0.8 x 2 = 1.5

0.6 x 2 = 1.2

$(0.15)_{10} = (0.001001\underline{1001}\ldots)_2$ on dit que $(0.15)_{10}$ est cyclique dans la base 2 de période 1001.

Pour passer du système binaire au système décimal, il suffit de sommer les différentes puissances de 2 en liaison avec chacun des chiffres 1 du nombre binaire.

Exemple n°1 : $100101_{(2)}$

Bits	1	0	0	1	0	1
Puissance	2^5	2^4	2^3	2^2	2^1	2^0
Pondération	32	0	0	4	0	1

Somme des pondérations: 32+4+1 = 37

Donc : $100101_{(2)} = 37_{(10)}$

Exemple n°2 : $1011{,}011_{(2)}$

Bits	1	0	1	1	0	1	1
Puissance	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Pondération	8	0	2	1	0	0,25	0,125

Somme des pondérations: 8+2+1+0,25+0,125 = 11,375

Donc : $1011{,}011_{(2)} = 11{,}375_{(10)}$

III.2 Conversions du système décimal ↔ système hexadécimal

Le passage du système hexadécimal vers le système décimal s'effectue selon le même principe que celui vu précédemment mais avec les puissances de 16.

$$1AB9 = 1 \times 16^3 + A \times 16^2 + B \times 16^1 + 9 \times 16^0$$
$$= 4096 + \mathbf{10} \times 256 + \mathbf{11} \times 16 + 9 \times 1$$
$$= 6841$$

Le passage du système décimal vers le système hexadécimal s'effectue, soit par des divisions successives par 16, soit par passage par la base 2.

Exemple: $469_{(10)} = ?_{(16)}$

Quotient	Reste (/16)
469	
29	5
1	13 (D)
0	1

Donc: $469_{(10)} = 1D5_{(16)}$

Exemples : $(4008)_{10} = (FA8)_{16}$, $(170)_{10} = (AA)_{16}$

Conversion des nombres comportant une partie fractionnaire, exemple : $(462{,}625)_{10} = (\ ?)_{16}$

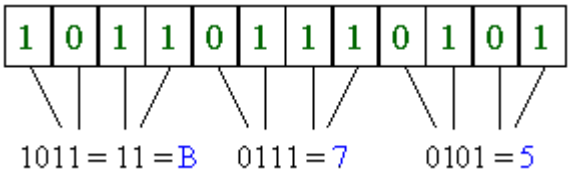
$(462)_{10} = (111001110)_2 = (\ 1CE)_{16}$

$0.625 \times 16 = \boxed{10}.00$

$(462{,}625)_{10} = (\ 1CE{,}A)_{16}$

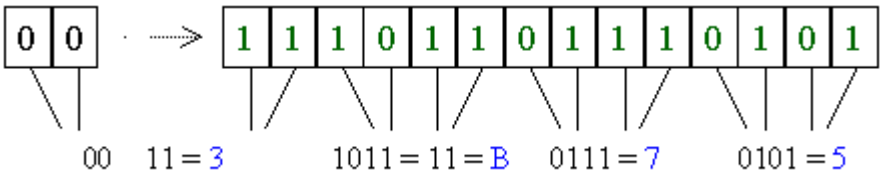
III.3 Conversions du système binaire ↔ système hexadécimal

Le passage du système binaire vers le système hexadécimal s'effectue selon le principe suivant: il suffit de faire des paquets de 4 bits, en partant du bit de poids faible et de donner l'équivalent hexadécimal de chaque paquet.



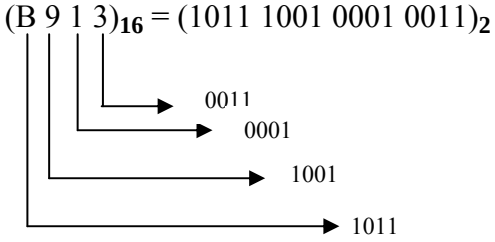
Donc **101101110101)$_{2}$ = B75)16**

Si le nombre binaire n'a pas un nombre de bits multiple de 4, il suffit de rajouter des « 0 » à gauche du nombre binaire afin d'obtenir un paquet de 4 bits: on procède ensuite de la même façon que ci-dessus.



Donc **11101101110101)$_{2}$ = 3B75)16**

Le passage de l'Hexadécimal vers Binaire est le processus directement inverse :



IV Les Opérations dans les bases

On procède de la même manière pour les opérations arithmétiques (addition, soustraction, multiplication et division) que celles de la base 10.

IV.1 Addition

Il faut faire la somme des chiffres dans la base 10 puis convertir dans la base B de travail en retenant les chiffres qui apparaissent à la gauche et en conservant le chiffre de l'extrémité droite.

- Exemple 1:

$$\begin{array}{r} 1\,0\,1\,1\,0\,0\,1 \\ +\quad 1\,1\,1\,0\,1 \\ \hline = (1\,1\,1\,0\,1\,1\,0)_2 \end{array}$$

IV.2 Soustraction

Exemple 1:

$$\begin{array}{r} B\,8\,6\,1 \\ -\quad C\,E\,F \\ \hline = (A\,B\,7\,2)_{16} \end{array}$$

IV.3 Multiplication

Exemple 1:

$$
\begin{array}{r}
1\,1\,0\,1\,1 \\
\times \quad 1\,1\,1\,0 \\
\hline
1\,1\,0\,1\,1 \\
1\,1\,0\,1\,1 \\
1\,1\,0\,1\,1 \\
\hline
= (1\,0\,1\,1\,1\,1\,0\,1)_{2}
\end{array}
$$

Exemple 2:

$$
\begin{array}{r}
7\;2\;A\;1 \\
\times \quad 2\;B\;3 \\
\hline
1\;5\;7\;E\;3 \\
4\;E\;C\;E\;B \\
E\;5\;4\;2 \\
\hline
= (1\,3\,5\,6\,8\,9\,3)_{16}
\end{array}
$$

V Représentation par les codes

V.1 Le code BCD

C'est un Code Binaire Décimal pour lequel chaque chiffre décimal est représenté sur 4 bits. Ce codage est destiné à l'affichage de valeurs décimales :

Base 10	Base BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Exemple :

$(125)_{10} = (\underline{0001}\ \ \underline{0010}\ \ \underline{0101})_{BCD}$

Ce codage ne permet aucun calcul, il est uniquement destiné à la saisie et à l'affichage de données.

V.2 Le code ASCII (American Standard Code for Informatique Interchange)

Le binaire permet de coder les nombres que les systèmes informatiques peuvent manipuler. Cependant, l'ordinateur doit aussi utiliser des caractères alphanumériques pour mémoriser et transmettre des textes. Pour coder ces caractères, on associe à chacun d'entre eux un code binaire, c'est le codage ASCII (American Standard Code for

Information Interchange). Chaque caractère est représenté sur 8 bits en utilisant le binaire naturel.

Exemple :

Le caractère A par exemple à pour code 65 soit 01000001 en binaire.

Le caractère f : 102

le point d'interrogation ? : 63

Le chiffre 2 : 50

A ⇨ $(65)_{10}$ ⇨ $(01000001)_{ASCII}$

[⇨ $(91)_{10}$ ⇨ $(01011011)_{ASCII}$

a ⇨ $(97)_{10}$ ⇨ $(01100001)_{ASCII}$

V.3 Le code binaire réfléchi (code GRAY)

Le code binaire réfléchi est utilisé pour simplifier des équations dans les tableaux de karnaugh. Le principe consiste à changer l'état d'un seul bit entre deux nombres consécutifs.

Comparaison entre le binaire et le binaire réfléchi

Décimal	Binaire naturel				Binaire réfléchi			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

V.4 Conversion Binaire naturel, Binaire réfléchi :

Il s'agit de faire une comparaison entre les bits b_{n+1} et b_n du binaire naturel, le résultat est b_n du binaire réfléchi qui vaut 0 si $b_{n+1}= b_n$ ou 1 si non. Le premier bit à gauche reste le même.

Exemples :

$(6)_{10} = (1\ 1\ 0)_{Bn}$

$= (1\ 0\ 1)_{Br}$

$(10)_{10} = (1\ 0\ 1\ 0)_{Bn}$

$= (1\ 1\ 1\ 1)_{Br}$

V.5 Conversion Binaire réfléchi, Binaire naturel :

On fait recours toujours à une comparaison entre les bits b_{n+1} du binaire naturel, et les bits b_n du binaire réfléchi. Le résultat est b_n du binaire naturel qui vaut 0 si $(b_{n+1})_n = (b_n)_r$ ou 1 si non. Le premier bit à gauche reste toujours le même.

$(10)_{10} = (1\ 1\ 1\ 1)_{Br}$

$=(1\ 0\ 1\ 0)_{Bn}$

$(12)_{10} = (1\ 0\ 1\ 0)_{Br}$

$=(1\ 1\ 0\ 0)_{Bn}$

Remarque : Deux nombres successifs du binaire réfléchi ne changent qu'un seul bit.

Chapitre II

Logique Booléenne

Objectifs

A la fin de la séance l'étudiant doit être capable de :

- Introduire à l'algèbre de Boole
- Reconnaître les fonctions logiques

Pré requis

L'étudiant est supposé connaître :

- Les outils mathématiques
- Les systèmes de numération et les codes

Durée

Trois séances de 1h30mn

Eléments de contenu

- Les opérateurs logiques de base
- Les opérateurs logiques élémentaires
- Les propriétés des opérateurs

Chapitre II:

Logique Booléenne

I Introduction

C'est la logique utilisée par les ordinateurs, les automates, les systèmes automatisés pour manipuler les données, les informations et les signaux reçus ou émis.

La logique binaire ou booléenne se repose sur une variable binaire (ou signal numérique) pouvant avoir uniquement deux valeurs qui sont 0 ou 1. Cette variable est employée pour traiter des phénomènes réduits à 2 états possibles: « ouvert ou fermé », « haut ou bas », « marche ou arrêt », « vrai ou faux », « tout ou rien ».

Au même titre que l'algèbre classique, on dispose d'opérateurs logique de base (ils sont au nombre de 3) pour établir des relations entre des événements binaires:

- l'addition logique symbolisée par le « + » (prononcé OU)
- le produit logique symbolisée par le «. » (prononcé ET)
- le complément ou l'inversion logique (prononcé NON)

II Définitions

II.1 Variable binaire

On appelle variable binaire (ou logique), une variable prenant ses valeurs dans l'ensemble {0,1}.

II.2 Equation logique

On appelle équation logique une combinaison de plusieurs variables logiques donnant l'état d'une variable dite de sortie associée.

II.3 Table de vérité

La table de vérité représente l'état de la variable de sortie pour chacune des combinaisons des n variables d'entrée (2^n lignes, n+1 colonne).

III Les opérateurs logiques

III.1 Addition logique (OU / OR)

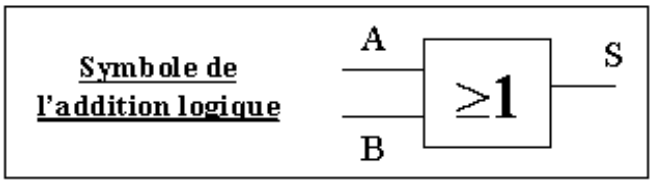
L'addition logique de 2 variables s'exprime de la façon suivante : **A+B**. La table de vérité d'un système intégrant 2 données d'entrée et 1 donnée de sortie est la suivante:

Table de vérité

Règle de l'addition logique:
*La donnée de sortie S prend l'état **1** lorsque **L'UNE AU MOINS** des données d'entrée vaut 1*

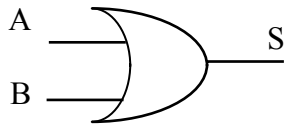
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Symbole de l'addition logique

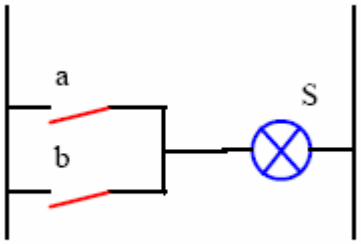


(Norme IEC : International Electrotechnical commission)

Symbole suivant la norme IEEE (Institue of Electrical ant Electronics Engineers)



Circuit électrique :



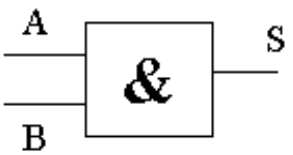
contacts en parallèle

III.2 Produit logique (ET / AND)

Le produit logique de 2 variables s'exprime de la façon suivante: **A .B**. La table de vérité d'un système intégrant 2 données d'entrée et 1 donnée de sortie est la suivante:

Règle du produit logique:
*La donnée de sortie S prend l 'état **1** lorsque **TOUTES** les données d 'entrée vaut 1*

Symbole du produit logique

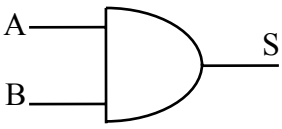


Norme IEC

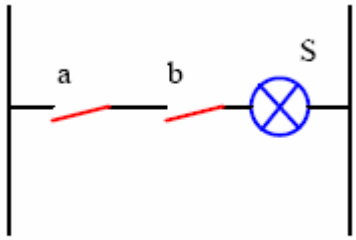
Table de vérité

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

Symbole suivant IEEE



Circuit électrique :



Contacts en série

III.3 Inversion logique (NON / NOT)

L'inversion logique d'une variable s'exprime de la façon suivante : $\overline{A}$. La table de vérité d'un système intégrant 1 donnée d'entrée et 1 donnée de sortie est la suivante:

Règle de l 'inversion logique:
Si la variable A vaut 1, son complément vaut 0 et vice-versa.

Symbole de l'inversion logique

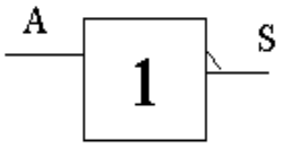
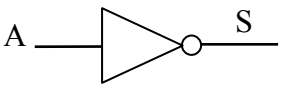


Table de vérité

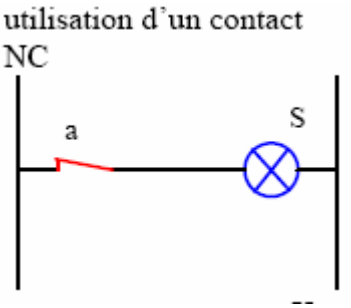
A	$S=\overline{A}$
0	1
1	0

Norme IEC

Symbole suivant IEEE



Circuit électrique :

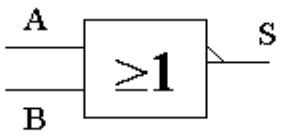


III.4 Opérateur NON-OU (NOR)

C'est tout simplement la fonction inverse de l'addition logique. Elle se note : $\overline{A+B}$

Règle de l'opérateur NON-OU:
*La donnée de sortie S prend l 'état **1** lorsque **TOUTES** les données d 'entrée valent 0*

Symbole de l'addition logique

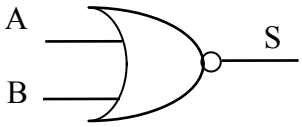


Norme IEC

Table de vérité

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

Symbole suivant IEEE



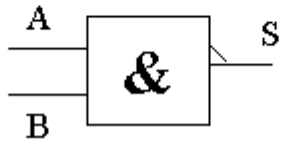
$S = \overline{a+b} = \bar{a}.\bar{b} = a \downarrow b$

III.5 Opérateur NON-ET (NAND)

C'est tout simplement la fonction inverse du produit logique. Elle se note : $\overline{A.B}$

Règle de l'opérateur NON-ET:
La donnée de sortie S prend l 'état ***0*** *lorsque* ***TOUTES*** *les données d 'entrée valent 1*

Symbole du produit logique



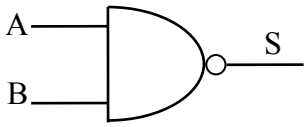
Norme IEC

Table de vérité

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

$S = \overline{a.b} = \overline{a} + \overline{b} = a \mid b$

Symbole suivant IEEE



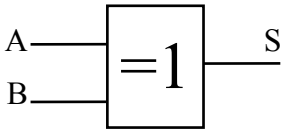
III.6 Opérateur OU Exclusif (XOR)

Cet opérateur se note $S = A \oplus B = A.B + A.B$

Règle de l'opérateur XOR

Cet opérateur logique binaire ne prend la valeur 1 que si une seule des entrées est à 1, son symbole est comme suit :

Symbole IEC :

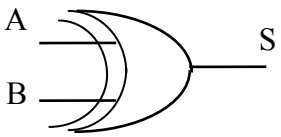


$S = A \oplus B$

Table de vérité

A	B	S= A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Symbole IEEE



Sur le plan pratique, les opérateurs logiques existent dans des technologies très diverses (portes électroniques, pneumatiques, etc…). Ils sont utilisés pour réaliser les équations logiques sous formes de circuits (exemple en électronique numériques : des circuits intégrés TTL, CMOS…).

IV Les propriétés des opérateurs

- Commutativité:

a ET b=b ET a $\Rightarrow$ a . b = b . a

a OU b=b OU a $\Rightarrow$ a + b = b + a

- Associativité:

a ET (b ET c) = (a ET b) ET c $\Rightarrow$ a . (b . c) = (a . b) . c

a OU (b OU c) = (a OU b) OU c $\Rightarrow$ a + (b + c) = (a + b) + c

- Distributivité:

a ET (b OU c) = a ET b OU a ET c $\Rightarrow$ a . (b + c) = a . b + a . c

a OU (b ET c) = (a OU b) ET (a OU c) $\Rightarrow$ a + (b . c) = (a + b) . (a + c)

- Propriétés de la somme:

a OU VRAI = VRAI $\Rightarrow$ a + 1 = 1

FAUX OU a = a OU FAUX = a $\Rightarrow$ 0 + a = a + 0 = a

a OU a = a $\Rightarrow$ a + a = a

a OU $\bar{a}$ = VRAI $\Rightarrow$ a + $\bar{a}$ = 1

- Propriétés du produit:

a ET VRAI = VRAI ET a = a $\Rightarrow$ a . 1 = 1 . a = a

a ET FAUX = FAUX $\Rightarrow$ a . 0 = 0

a ET a = a $\Rightarrow$ a . a = a

a ET ā = FAUX $\Rightarrow$ a . ā = 0

- Negation:

$$(\bar{a}) = a$$

- Propriétés combinées:

a ET (a OU b) = a $\Rightarrow$ a . (a + b) = a + a . b = a . (1 + b) = a

a OU (a ET b)=a $\Rightarrow$ a + (a . b) = a . (1 + b) = a

- De Morgan:

$$(\overline{aETb}) = \bar{a}OU\bar{b}; (\overline{a.b}) = \bar{a} + \bar{b}$$

$$(\overline{aOUb}) = \bar{a}ET\bar{b} \Rightarrow (\overline{a+b}) = \bar{a}.\bar{b}$$

Application

1- Démontrer si les opérateurs NOR et NAND sont associatifs :

$(a \mid b) \mid c \overset{?}{=} a \mid (b \mid c)$

$(a \downarrow b) \downarrow c = a \downarrow (b \downarrow c)$

2- Réaliser F =a | b | c avec des portes NAND à deux entrées.

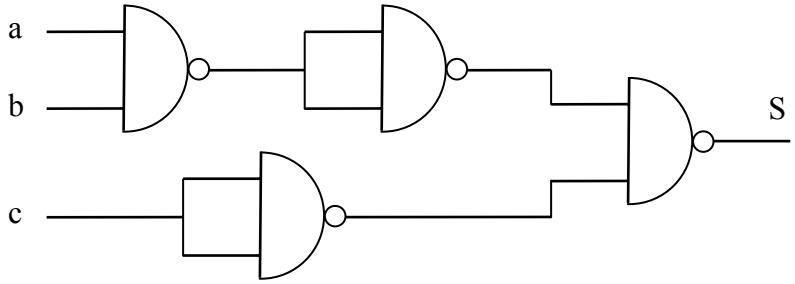
- *Réponses*

1-

- ✓ $(a \mid b) \mid c = \overline{(a\ b)} \mid c$
  $= \overline{\overline{(a\ b)} + \overline{c}}$
  $= a\ b + \overline{c}$
- ✓ $a \mid (b \mid c) = a \mid \overline{(b\ c)}$
  $= \overline{a} + \overline{\overline{(b\ c)}}$
  $= \overline{a} + b\ c$
  - ☞ $ab + \overline{c} \neq \overline{a} + b\,c$ **alors la fonction NAND n'est pas associative**
- ✓ $(a \downarrow b) \downarrow c = \overline{(a + b)} \downarrow c$
  $= \overline{\overline{(a + b)}\ \overline{c}}$
  $= (a + b)\overline{c}$
  $= a\,\overline{c} + b\,\overline{c}$
- ✓ $a \downarrow (b \downarrow c) = a \downarrow \overline{(b + c)}$
  $= \overline{a}\ \overline{(b + c)}$
  $= \overline{a}\,(b + c)$
  $= \overline{a}\,b + \overline{a}\,c$
  - ☞ $a\overline{c} + b\overline{c} \neq \overline{a}b + \overline{a}c$ **alors la fonction NOR n'est pas associative**

2- $a \mid b \mid c = \overline{a \cdot b} \cdot c$

$$= \overline{\overline{a} + \overline{b}} + \overline{c}$$
$$= \overline{\overline{a} \cdot \overline{b}} + \overline{c}$$
$$= \overline{a \cdot b} + \overline{c}$$
$$= \overline{\overline{(a \mid b) + c}}$$
$$= \overline{\overline{(a \mid b)} \cdot \overline{c}}$$
$$= \overline{(a \mid b)} \mid \overline{c}$$
$$= [(a \mid b) \mid (a \mid b)] \mid (c \mid c) = [(a \mid b) \mid 1] \mid (c \mid 1)$$



- **Remarques:**

Les portes **NAND** et **NOR** sont très usitées dans la réalisation des circuits logiques. Grâce aux lois de **De Morgan** il est possible de réaliser des systèmes logiques avec uniquement des portes **NAND** ou **NOR.** Pour cela les portes **NAND** et **NOR** sont appelées des **portes UNIVERSELLES**.

Règle1 : Pour trouver l'expression en **NAND** d'une fonction logique on fait son développement en somme de produit et l'on remplace les signes opératoires **OU** et **ET** par le signe opératoire **NAND**

Règle2 : Pour trouver l'expression en **NOR** d'une fonction logique on fait son développement en produit de somme et l'on remplace les signes opératoires **OU** et **ET** par le signe opératoire **NOR**

Chapitre III:

Simplification des fonctions logiques

Objectifs

A la fin de la séance l'étudiant doit être capable de :

- Comprendre la simplification algébrique
- Comprendre la simplification graphique
- Synthétiser des applications combinatoires

Pré requis

L'étudiant est supposé connaître :

- Les systèmes de numération et les codes
- Logique Booléenne

Durée

Trois séances de 1h30mn

Eléments de contenu

- Ecriture des fonctions logiques
- Simplification algébrique
- Simplification graphique

Chapitre III:

Simplification des fonctions logiques

I Écriture des fonctions logiques

I.1 Somme canonique de produits

Considérons trois variables booléennes x, y et z. A partir de ces trois variables nous pouvons construire huit produits logiques (ou min terme) Pi=0,7 faisant intervenir x ou son complément, y ou son complément et z ou son complément. Pour chacune des huit combinaisons Ci=0,7 (000, 001, 010, etc…) des variables x, y et z, nous pouvons calculer les valeurs de ces produits. Celles-ci sont rassemblées dans la table suivante. Chacun de ces produits ne prend la valeur 1 que pour une et une seule combinaison : Pi vaut 1 uniquement pour la combinaison Ci.

				P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
C_i	x	y	z	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}y\bar{z}$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	$xy\bar{z}$	xyz
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Pour toute fonction logique de trois variables x, y et z, nous pouvons écrire sa table de vérité, c'est-à-dire expliciter sa valeur pour chacune des huit combinaisons C_i.

Considérons, par exemple, la fonction F dont la table de vérité est donnée dans la table suivante :

C_i	x	y	z	F	$P_1 + P_3 + P_4$
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	0	0

Cette fonction F prend la valeur 1 pour la combinaison C_1 comme le produit P_1, la combinaison C_3 comme P_3 et la combinaison C_4 comme P_4. La fonction F prenant la valeur 0 pour toutes les autres combinaisons comme les produits P_1, P_3, P_4, nous pouvons donc écrire que F est égale à la fonction :

$$F = P_1 + P_3 + P_4$$

Nous pouvons vérifier cette identité dans la table 11. Nous pouvons donc exprimer F en fonction des variables x, y et z sous la forme :

$$\mathbf{F = \bar{x}\,\bar{y}\,z + \bar{x}\,y\,z + x\,\bar{y}\,\bar{z}}$$

Cette façon, très générale, d'écrire une fonction booléenne est appelée somme canonique de produits : C'est La **première forme canonique**

I.2 Produit canonique de sommes

Soient encore trois variables binaires **x, y** et **z**. Nous pouvons définir huit sommes logiques (ou max terme) $\mathbf{Si_{=0.7}}$ faisant intervenir **x** ou son complément, **y** ou son complément et **z** ou son complément. La table suivante donne les tables de vérité de ces sommes. Nous constatons que chacune de ces fonctions ne prend la valeur **0** que pour une et une seule combinaison.

				S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
C_i	x	y	z	$x+y+z$	$x+y+\bar{z}$	$x+\bar{y}+z$	$x+\bar{y}+\bar{z}$	$\bar{x}+y+z$	$\bar{x}+y+\bar{z}$	$\bar{x}+\bar{y}+z$	$\bar{x}+\bar{y}+\bar{z}$
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

Reprenons l'exemple précédent de la fonction F. Celle-ci vaut 0 pour les combinaisons C_0, C_2, C_5, C_6 et C_7 en même temps que S_0, S_2, S_5, S_6 et S_7. La fonction F peut donc être vue comme le produit logique de ces cinq sommes, ce qui est vérifié dans la table suivante. Nous pouvons donc exprimer la fonction F sous la forme suivante :

$$F = (x+y+z) \bullet (x+\bar{y}+z) \bullet (\bar{x}+y+\bar{z}) \bullet (\bar{x}+\bar{y}+z) \bullet (\bar{x}+\bar{y}+\bar{z})$$

Cette écriture est appelée produit canonique de sommes.

C_i	x	y	z	F	$S_0 \bullet S_2 \bullet S_5 \bullet S_6 \bullet S_7$
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	0	0

C'est **La seconde forme canonique**.

Application :

Soit l'équation logique suivante donnée sous forme canonique 1 :

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

1- Montrer que cette équation pouvait se simplifier à :

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

Soit une deuxième équation logique donnée sous forme canonique 2 :

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

2- Montrer que cette équation aussi pouvait se simplifier à :

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

Réponse :

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

2-

- $F = (AA + AB + A\overline{C} + BA + BB + B\overline{C} + CA + CB + C\overline{C})(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + \overline{C})$
- $F = (A + AB + A\overline{C} + AB + B + B\overline{C} + AC + BC + 0)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + \overline{C})$
- $F = (A + B)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + \overline{C})$
- $F = (AA + A\overline{B} + A\overline{C} + BA + B\overline{B} + B\overline{C})(\overline{A} + \overline{B} + \overline{C})$
- $F = (A + B\overline{C})(\overline{A} + \overline{B} + \overline{C})$
- $F = (A\overline{A} + A\overline{B} + A\overline{C} + B\overline{C}\,\overline{A} + B\overline{C}B + B\overline{C}\overline{C})$
- $F = (A\overline{B} + A\overline{C} + B\overline{C}) = A\overline{B} + B\overline{C}$

On utilisera ici $F = a + \overline{a}.b = (a + \overline{a}).(a + b) = a + b$

Donc une même équation logique peut être représenté sous ces deux formes canoniques.

I.3 Forme canonique avec NAND et NOR

La **troisième forme canonique** est celle qui présente les équations logiques sous une forme n'utilisant que des fonctions logiques NON-ET (NAND). Cette forme n'est utile que lorsque l'on utilise des circuits électroniques. Cette forme canonique s'obtient directement de la première forme canonique. Il suffit de faire une double négation de l'équation sous forme canonique 1.

Par exemple pour l'équation logique :

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

- 1) double négation : $F = \overline{\overline{A}B\overline{C} + AB\overline{C} + A\overline{B}\overline{C} + A\overline{B}C}$
- 2) théorème de De Morgan : $F = \overline{\overline{A}B\overline{C}} \bullet \overline{AB\overline{C}} \bullet \overline{A\overline{B}\overline{C}} \bullet \overline{A\overline{B}C}$

Cette dernière équation est sous forme canonique 3. La quatrième forme canonique est celle qui présente les équations logiques sous une forme n'utilisant que des fonctions logiques NON-OU (NOR). Cette forme non plus n'est utile que lorsque l'on utilise des circuits électroniques.

Cette forme canonique s'obtient directement de la seconde forme canonique. Il suffit de faire une double négation de l'équation sous forme canonique 2. Par exemple pour l'équation logique

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

Le processus est :

- 1) double négation : $F = \overline{(A + B + C)(A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + \overline{C})}$
- 2) théorème de De Morgan :

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

Cette dernière équation est sous forme canonique 4.

I.4 Forme canonique décimal

L'écriture des expressions logique a cet inconvénient d'être assez longue. Chaque min terme parmi les $\mathbf{2^n}$ de $\boldsymbol{n}$ variables correspond à un nombre représentant son ordre, c'est pourquoi on préfère parfois utiliser une écriture indiquant la liste classée des numéros des min termes de la première forme canonique.

- Exemple :

Soit :

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

peut aussi s'écrire $F(a,b,c,d) = \sum (0,6,10,15)$

II Technique de Simplification

Les tables de vérité permettent de trouver les équations logiques d'un système logique. Une simplification de ces équations est nécessaire pour diminuer le nombre et la nature des opérateurs et par la suite les équipements indispensables aux applications sans modifier le fonctionnement.

Deux techniques sont souvent utilisées pour simplifier les équations logiques, la méthode algébrique et la méthode du tableau de Karnaugh dés que le nombre des entrées est élevé.

II.1 Méthode Algébrique

La simplification est obtenue par calcul algébrique en utilisant :

-Les équations booléennes

-Les propriétés des fonctions logiques

-Les théorèmes de De Morgon

-La mise en facteurb

-La multiplication par 1 (car $X + \bar{X} = 1$)

-L'addition d'un terme nul « 0 » (car $\bar{X} \bullet \bar{X} = 0$)

Exemples :

$F = a + \bar{a}..b = (a + \bar{a})(a + b) = a + b$

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

$= c.(b + a.\bar{b}) + a.b.\bar{c} = c.(a + b) + a.b.\bar{c}$

$= a.c + b.(c + a.\bar{c}) = a.c + b.(a + c)$

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

$= a.(c + (a + c).b + 1) + b.\bar{c} + b.c$

$= a + b.(c + \bar{c}) = a + b$

II.2 Méthode graphique

II.2.1 Tableau de Karnaugh

C'est un tableau de 2n cases, n étant le nombre de variables.

Sur les lignes et colonnes, on place l'état des variables d'entrée codées en binaire.

Dans chacune des cases, on place l'état de la sortie pour les combinaisons d'entrée correspondante.

Les figures suivantes donnent la structure des tableaux de Karnaugh pour 2, 3, 4 et 5 variables. Pour 5 variables, deux représentations sont possibles. Le tableau de Karnaugh peut être traités comme deux tableaux 4x4 superposés ou un seul tableau de 4x8. Observez comment sont numérotées les lignes et les colonnes : d'une case à sa voisine une seule variable change d'état.

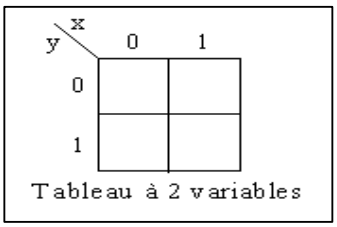


Tableau à 2 variables

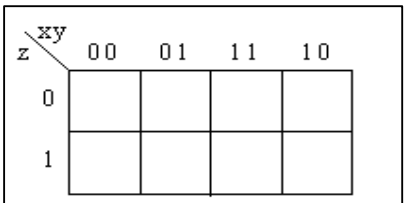


Tableau à 3 variables

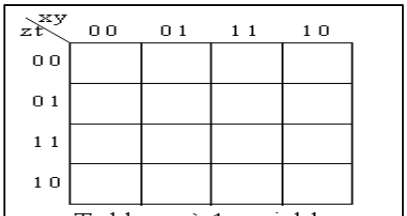


Tableau à 4 variables

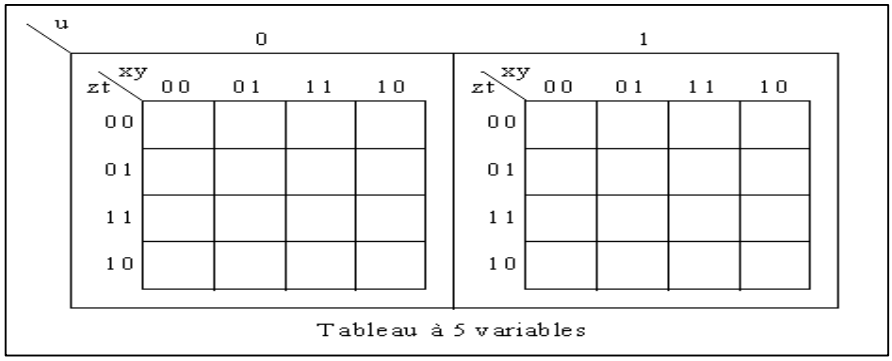


Tableau à 5 variables

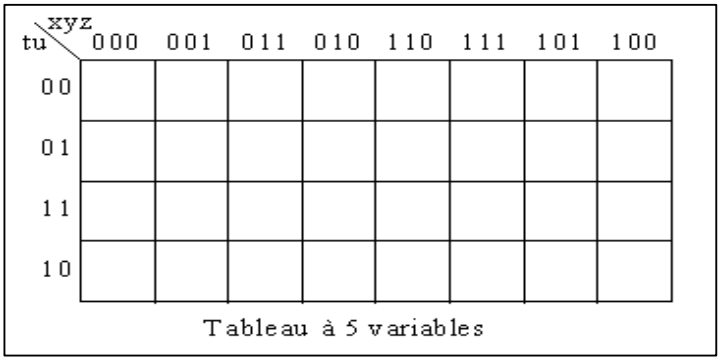
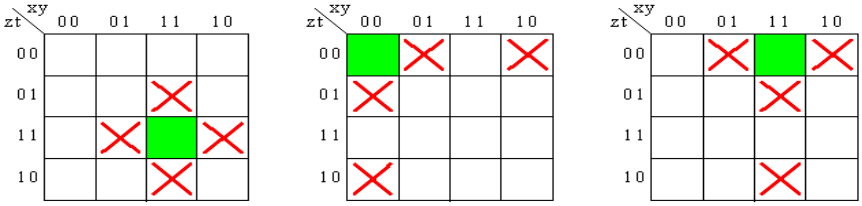


Tableau à 5 variables

Chaque case d'un tableau correspond au seul minterm prenant la valeur **1** pour la combinaison identifiée par la ligne et la colonne. Par exemple les trois cases coloriées dans les tableaux de la figure suivante correspondent respectivement aux produits suivants :

$$xyzt, \quad \bar{x}\bar{y}\bar{z}\bar{t} \quad et \quad xy\bar{z}\bar{t}$$

Il faut comprendre chaque ligne et chaque colonne comme une structure cyclique continue : chaque case a toujours quatre voisins qu'il faut éventuellement chercher à l'autre extrémité de la ligne ou de la colonne. Les tableaux suivants illustrent ce concept, les croix y matérialisent les voisins des cases coloriées :



$(xyzt)$ $(\bar{x}\bar{y}\bar{z}\bar{t})$ $(xy\bar{z}\bar{t})$

Dans le cas de la représentation en deux tableaux superposés chaque case a cinq voisins : les quatre dans le même plan et une dans l'autre plan.

II.2.2 Simplification d'équations à partir du tableau de Karnaugh

Le passage de la table de vérité au tableau de Karnaugh consiste à remplir chaque case avec la valeur de la fonction pour le produit correspondant. Il est possible de n'indiquer que les **1.**

La méthode de simplification par tableau de Karnaugh consiste à mettre en évidence, par un procédé graphique, tous les termes d'une fonction logique qui ne diffèrent que par l'état d'une seule variable (termes dits adjacents).

Pour cela on réalise des groupements de cases adjacentes. Ces groupements de cases doivent être de taille maximale (nombre de cases max.) et égale à un multiple de 2^n. On cesse d'effectuer les groupements lorsque tous les « 1 » appartiennent au moins à l'un d'eux.

On cherche à avoir le minimum de groupements, chaque groupement rassemblant le maximum de termes. Une même case peut intervenir dans plusieurs groupements car **C + C = C**. C'est le cas de la case jaune sur la figure suivante.

Pour les cases isolées on ne peut éliminer aucune variable. On conserve donc le produit caractérisant la case. L'expression logique finale est la réunion des groupements après élimination des variables qui changent d'état.

a- Regroupement des cases adjacentes:

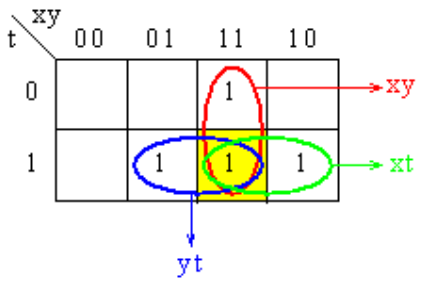
☞ ***2 cases:***

Exemple: tableau à 3variables

Prenons l'exemple d'une fonction F définie par la table de vérité suivante :

x	y	t	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

La figure suivante donne le tableau de Karnaugh correspondant :



Nous y observons trois groupements de deux termes, nous pouvons écrire pour la fonction :

$$F = xy + yt + xt$$

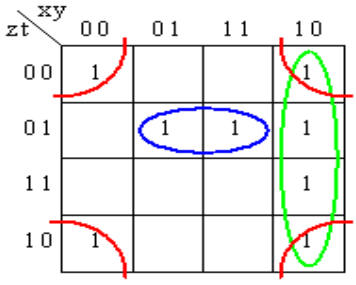
Règle: « La réunion de 2 cases adjacentes contenant '1' élimine la variable qui change d'état quand on passe d'une case à l'autre ».

☞ *4 cases:*

Considérons la fonction **F** de quatre variables **x**, **y**, **z** et **t** définie par la table de vérité suivante :

x	y	z	t	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

La figure suivante donne le tableau de Karnaugh équivalent. Sur cette figure nous avons également matérialisé les trois groupements possibles : deux groupements de quatre termes, dont un contenant les quatre coins, et un groupement de deux termes.



Cette méthode nous permet d'écrire : $F = x\bar{y} + \bar{y}\bar{t} + y\bar{z}t$

Règle: « 2 variables disparaissent quand on regroupe 4 cases adjacentes, on peut alors remplacer la somme des 4 cases par un seul terme produit qui ne comporte que les variables inchangées sur l'ensemble des 4 cases ».

☞ ***8 cases:***

- Exemple:

AB \ CD	$\bar{C}\,\bar{D}$ 00	$\bar{C}$ D 01	CD 11	C $\bar{D}$ 10
$\bar{A}\,\bar{B}$ 00	**1**			**1**
$\bar{A}$ B 01	**1**			**1**
AB 11	**1**			**1**
A $\bar{B}$ 10	**1**			**1**

$F = \bar{C}.\bar{D} + C.\bar{D} = \bar{D}$

Règle: « 3 variables disparaissent quand on regroupe 8 cases adjacentes, on peut alors remplacer la somme des 8 cases par un seul terme produit qui ne comporte que les variables inchangées sur l'ensemble des 8 cases ».

*** Remarques:**

- On ne peut regrouper que 2^n cases: 2, 4, 8, 16, ...
- On se limitera à des tableaux de 4 variables, pour résoudre par exemple un problème à 5 variables, on le décompose en 2 problèmes à 4 variables.
- Le symbole ∅ peut prendre indifféremment la valeur 0 ou 1; on remplace donc par 1 uniquement ceux qui permettent de simplifier une expression par regroupement.

☞ **Exemple:**

A B C	F(A,B,C)
0 0 0	∅
0 0 1	0
0 1 0	1
0 1 1	∅
1 0 0	0
1 0 1	0
1 1 0	∅
1 1 1	1

➔

AB \ C	0	1
00	∅	0
01	1	∅
11	∅	1
10	0	0

➔ F(A,B,C) = B

b- regroupement des Zéros

Il est également possible et c'est parfois plus facile, de regrouper les 0 de F et de considérer que nous étudions $\overline{F}$. Trouver l'équation de F nous amènerait simplement à considérer le complément de l'équation trouvée pour F.

AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}$ D 01	CD 11	C $\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	**0**	**1**	**1**	**1**
$\overline{A}$ B 01	**0**	**1**	**1**	**1**
AB 11	**0**	**1**	**1**	**1**
A $\overline{B}$ 10	**0**	**1**	**1**	**0**

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

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

a. Sortir l'équation simplifiée des tableaux suivants :

	00	01	11	10
00	0	1	1	0
01	0	0	1	1

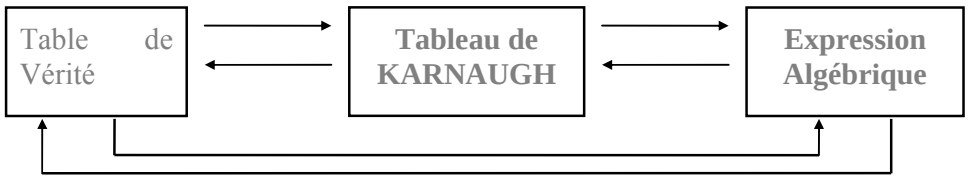
	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	0	1	1	0
10	0	1	1	0

	00	01	11	10
00	1	1	0	0
01	0	1	0	0
11	1	1	0	0
10	1	0	0	1

	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	0	0	1

Résumé: la synthèse d'un système combinatoire

* Différents aspects d'une fonction logique:



* Passage T.V. ==> T.K. ==> E.A.:

- Etape n°1: construire le tableau en repérant les lignes et les colonnes par les valeurs des combinaisons de variables.
- Etape n°2: transcrire les valeurs de la fonction dans les cases correspondantes.
- Etape n°3: chercher à effectuer des regroupements du plus grand nombre de '1' qui ont au moins un '1' qui n'a pas déjà été regroupé: 16 puis 8 puis 4 puis 2.
- Etape n°4: effectuer la somme logique de tous les termes produits des divers regroupements.

Chapitre IV

Circuits Logiques Combinatoires

Objectifs

A la fin de la séance l'étudiant doit être capable de :

- Reconnaître les circuits logiques combinatoires
- Utiliser les circuits combinatoires pour la réalisation des fonctions logiques.

Pré requis

L'étudiant est supposé connaître :

- Les systèmes de numération et les codes
- Logique Booléenne
- Simplification des fonctions logiques

Durée

Trois séances de 1h30mn

Eléments de contenu

- Codeurs et Décodeurs
- Multiplexeurs et Démultiplexeurs
- Comparateurs et Additionneurs

Chapitre IV

Circuits Logiques Combinatoires

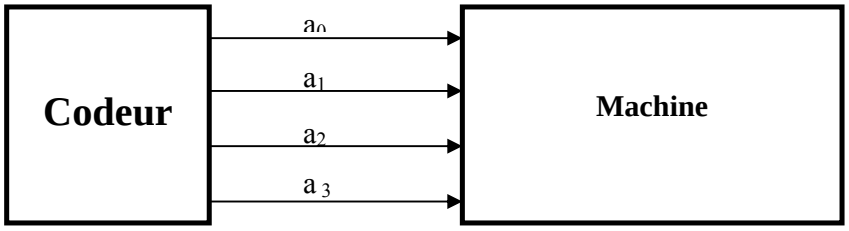
I Introduction

A un système combinatoire, correspond un circuit logique possédant un nombre n des entrées e_1, $e_{2, ...}$ e_n et un nombre p de sorties s_1, $s_{2, ...}$ s_p. A une combinaison d'entrées quelconque correspond une et une seule combinaison de sorties.

Nous allons étudier des systèmes logiques constitués avec des circuits de base qu'utilisent les différentes portes logiques.

II Les Codeurs

Un codeur est un circuit logique combinatoire qui reçoit un niveau valide de ces entrées convertit en une sortie codée (par exemple un **DCB** ou un nombre binaire).

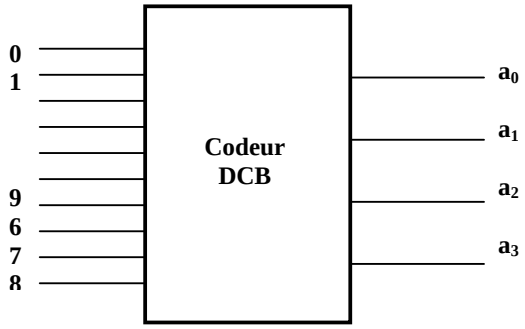


Ainsi, pour un codeur, lorsqu'une entrée (sur les N) est activée, les sorties affichent le numéro de l'entrée active dans le code binaire choisi (sur n bits), tel que:

$$2n-1 < N \leq 2n$$

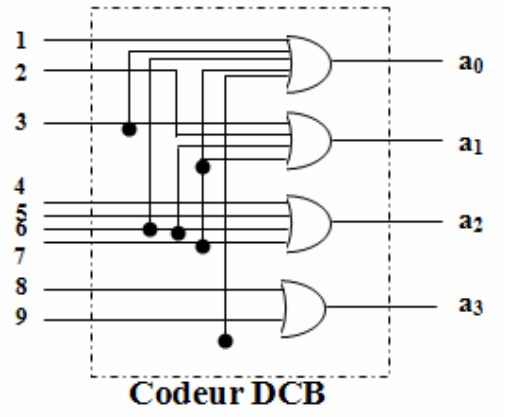
Exemple: codeur décimal vers binaire (10 entrées vers 4 sorties) Ex: si E5=1 et Ei=0 pour toutes les autres entrées, alors les sorties affichent (A3,A2,A1,A0)=(0,1,0,1).

- **Exemple : Codeur DCB**



Entrées	Sorties			
	a_3	a_2	a_1	a_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

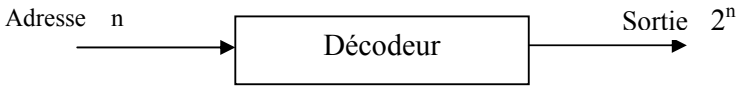
a_0 = 1 + 3 + 5 + 7 + 9
a_1 = 2 + 3 + 6 + 7
a_2 = 4 + 5 + 6 + 7
a_3 = 8 + 9



III Les Décodeurs

Un décodeur est un circuit combinatoire qui détecte la présence d'une combinaison spécifique de bits (code) à ses entrées et l'indique par un niveau spécifique de sortie.

Un décodeur possède n entrées et 2^n sorties. L'entrée n est dite entrée '' Adresses''.



Une seule sortie est active à la fois : 0 ou 1 suivant le circuit. Ce circuit permet de sélectionner une sortie parmi 2^n.

Exemple :

Décodeur **2** entrées **4** sorties : **1** sortie active à **1** parmi **4**.

Entrées Adresses B	A	Sorties S0	S1	S2	S3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$S0 = \bar{A}\bar{B}$$
$$S1 = A\bar{B}$$
$$S2 = \bar{A}B$$
$$S3 = AB$$

☞ **Remarques:**

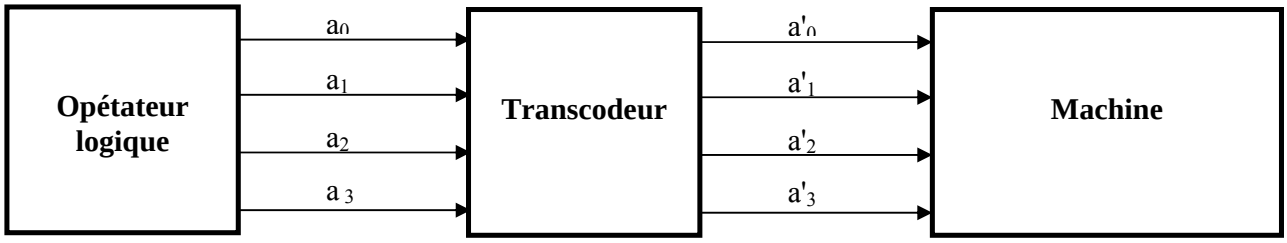
- On peut réaliser des décodeurs de taille quelconque par combinaisons des précédents en utilisant les entrées de validation.

Exemple: un circuit de décodage des combinaisons de 5 variables: 1 parmi 32, en utilisant 4 décodeurs 1 parmi 8 ou bien 2 décodeurs 1 parmi 16.

- Exemples d'utilisation des décodeurs: Système de mémoire d'un ordinateur: les décodeurs permettent de choisir un bloc mémoire parmi plusieurs blocs en réponse à une adresse.

IV Les Transcodeurs

Un transcodeur est un circuit logique combinatoire qui détermine à partir des données logiques leurs équivalents dans un autre code.
Il est généralement formé d'un décodeur en cascade avec un codeur.



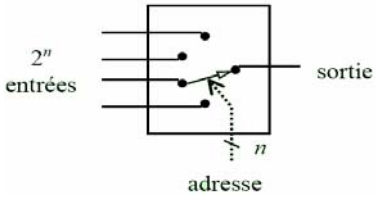
Exemple : Transcodeur binaire naturel - binaire réfléchi

V Les Multiplexeurs (MUX)

Pour transmettre des informations d'une station à une autre, il nous faut plusieurs lignes en parallèle. Ceci est difficile et très onéreux à réaliser lorsque les stations sont géographiquement éloignées l'une de l'autre. La solution est alors, transmettre en série sur une seule ligne, en utilisant à la station émettrice un Multiplexeur (convertisseur parallèle/série) et à la station réceptrice un Demiltiplexeur (convertisseur série/parallèle).

V.1 Définition:

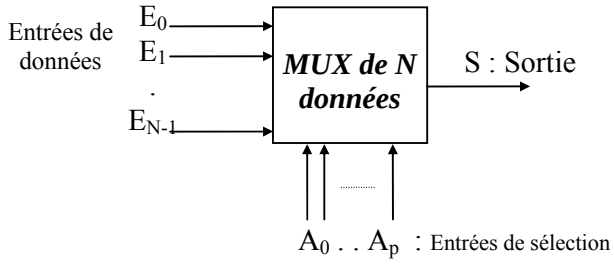
Un multiplexeur est un circuit logique combinatoire qui permet d'acheminer les informations numériques de plusieurs sources sur une ligne, afin de les transmettre vers une destination commune.



C'est un circuit logique qui permet de sélectionner une information logique parmi N informations:

- Les informations sont connectées à N entrées appelées « entrées de données ».

- Le choix de l'entrée se fait à partir d'un nombre P de variables appelées « variables de sélection ».
- Chaque combinaison des variables de sélection adresse l'une des entrées d'où: $\mathbf{N=2^P}$.



V.2 Multiplexeur à 2 entrées: « N=2 et P=1 »

Il permet d'aiguiller vers la sortie S, une voie d'information parmi 2 (E_0,E_1) suivant l'état d'une variable de sélection notée A_0.

**** Table de fonctionnement:***

A_0	Sortie S
0	Sélecteur de E_0
1	Sélecteur de E_1

****Symbole logique:***

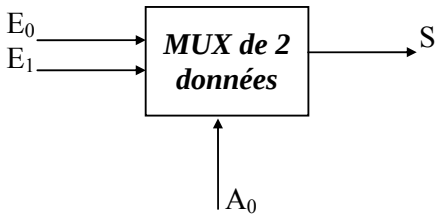


****Table de vérité:***

A_0	E_1	E_0	S
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

$$S(A_0,E_0,E_1) = \overline{A}_0.S(0,E_0,E_1)+A_0.S(1,E_0,E_1) = \overline{A}_0.E_0+A_0.E_1$$

V.3 Multiplexeur à 4 entrées: « N=4 et P=2 »

Il permet d'aiguiller vers la sortie S, une voie d'information parmi 4 (E_0,E_1,E_2,E_3) suivant l'état de 2 variables de sélection A_0A_1.

**** Table de fonctionnement:***

A_1	A_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

****Symbole logique:***

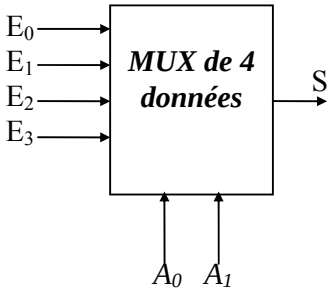
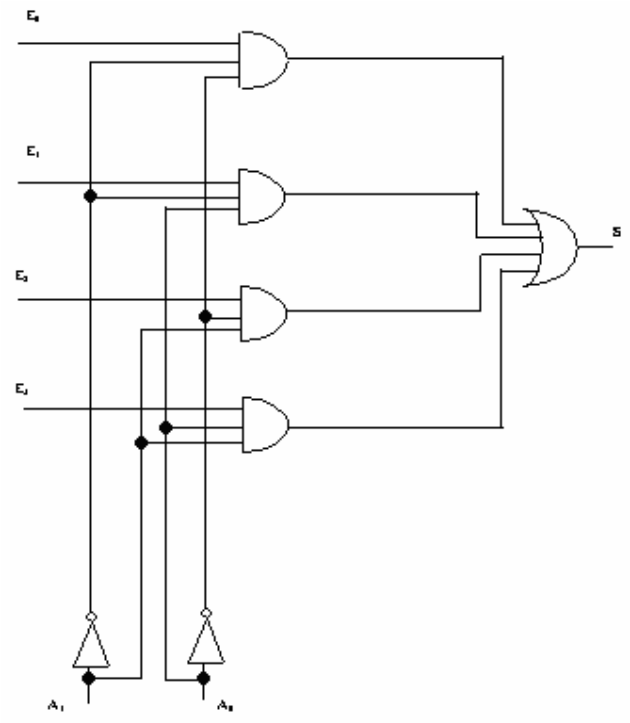


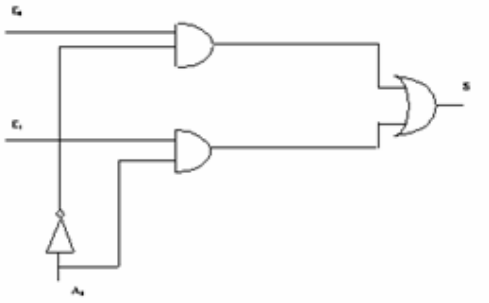
****Table de vérité:***

A_1A_0	E_3	E_2	E_1	E_0	S
0 0	X	X	X	0	0
0 0	X	X	X	1	1
0 1	X	X	0	X	0
0 1	X	X	1	X	1
1 0	X	0	X	X	0
1 0	X	1	X	X	1
1 1	0	X	X	X	0
1 1	1	X	X	X	1

$$S = \overline{A}_1.\ \overline{A}_0.E_0 + \overline{A}_1.A_0.E_1 + A_1.\ \overline{A}_0.E_2 + A_1.A_0.E_3$$

Q1 : Réaliser les schémas logiques des multiplexeurs à 2 entrées et à 4 entrées ?





Multiplexeurs à 8 voies d'entrées: (P = 3)
*** Exemple:** « le circuit 74LS151 à 8 entrées »

- Equation:

$Y= \overline{A}_2.\overline{A}_1.\overline{A}_0.E_0+\overline{A}_2.\overline{A}_1.A_0.E_1+\overline{A}_2.A_1.\overline{A}_0.E_2+\overline{A}_2.A_1.A_0.E_3+A_2.\overline{A}_1.\overline{A}_0.E_4+A_2.\overline{A}_1.A_0.E_5+A_2.A_1.\overline{A}_0.E_6+A_2.A_1.A_0.E_7$

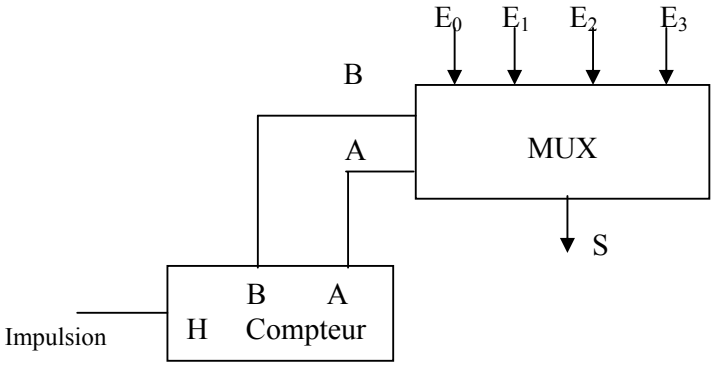
Exemples de multiplexeurs à 16 entrées: (P = 4)

- En technologie TTL: 74LS150
- En technologie CMOS: MC14067

V.4 Applications des multiplexeurs :

- Réaliser des aiguillages d'informations,
- Lire l'état d'une entrée en fonction de l'adresse n de l'entrée (1 parmi 2^n),
- Permettre la ''sérialisation'' d'une information, c'est à dire la transformation parallèle-série.

Exemple : principe de transformation parallèle-série



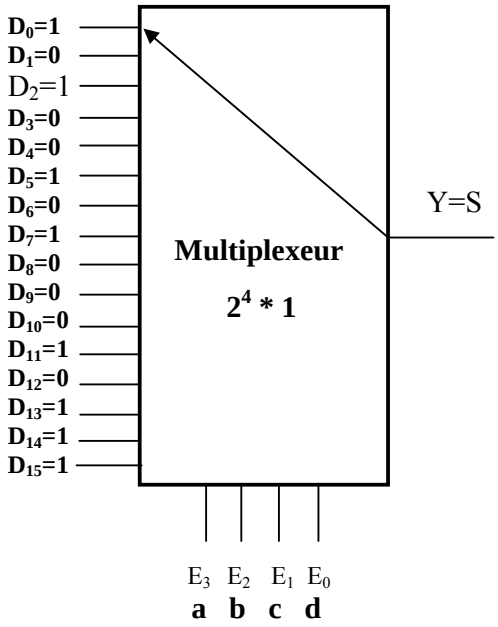
Le compteur à chaque impulsion d'horloge, délivre sur les entrée d'adresse A et B du Multiplexeur les informations (00,01,10,11,00…) ce qui implique sur la sortie, successivement les informations E_0, E_1,E_2, E_3, E_0….

V.5 Réalisation des fonctions logiques à l'aide des multiplexeurs

Soit la fonction $F_{(a, b, c, d)} = \Sigma (0, 2, 5, 7, 11, 13, 14, 15)$

En utilisant un Multiplexeur 16 vers 1 câbler cette fonction.

Décimal	**E_3=d**	**E_2=c**	**E_1=b**	**E_0=a**	**Y**	**S**
0	0	0	0	0	D_0	1
1	0	0	0	1	D_1	0
2	0	0	1	0	D_2	1
3	0	0	1	1	D_3	0
4	0	1	0	0	D_4	0
5	0	1	0	1	D_5	1
6	0	1	1	0	D_6	0
7	0	1	1	1	D_7	1
8	1	0	0	0	D_8	0
9	1	0	0	1	D_9	0
10	1	0	1	0	D_{10}	0
11	1	0	1	1	D_{11}	1
12	1	1	0	0	D_{12}	0
13	1	1	0	1	D_{13}	1
14	1	1	1	0	D_{14}	1
15	1	1	1	1	D_{15}	1



VI Les Démultiplexeurs : (DMUX)

VI.1 Définition:

Un démultiplexeur prend les données reçues sur une ligne et les distribue à un nombre donné des lignes de sortie.

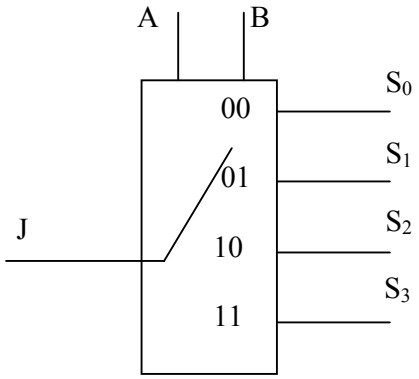
Un circuit démultiplexeur permet d'aiguiller la donnée présentée sur son entrée vers une seule destination parmi N connectées sur les N sorties du circuit. Le choix se fait à partir de P variables de sélection d'où: $N=2^P$.

⇨ C'est l'opération inverse du multiplexage.

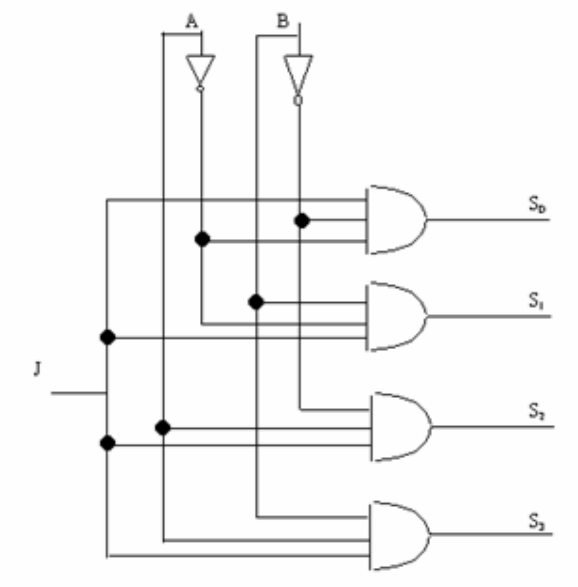
VI.2 Réalisation:

Le démultiplexage d'informations de « 1 bit » est réalisé pratiquement par les circuits décodeurs => appellation « décodeur/démultiplexeur »:

- L'entrée de la donnée du démultiplexeur est l'entrée de validation du circuit.
- Les entrées de sélection du démultiplexeur sont les entrées de données du circuit.



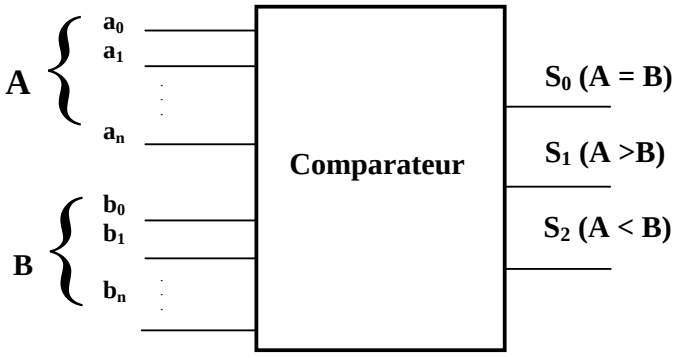
Q1 : Réaliser le schéma logique du démultiplexeur à 4 sorties ?



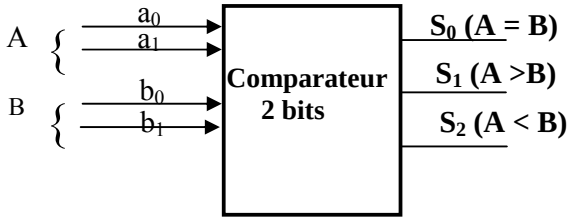
VII Les Comparateurs

C'est un circuit permettant de comparer 2 mots de n bits chacun en indiquant sur ses sorties S1, S2 et S3 si le premier mot est égal, plus grand ou plus petit que le second.

S_0 = 1 si A = B
S_1= 1 si A > B
S_2 = 1 si A <B



- **Exemple : Comparateur 2 bits**



b_1	b_0	a_1	a_0	S_0	S_1	S_2
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

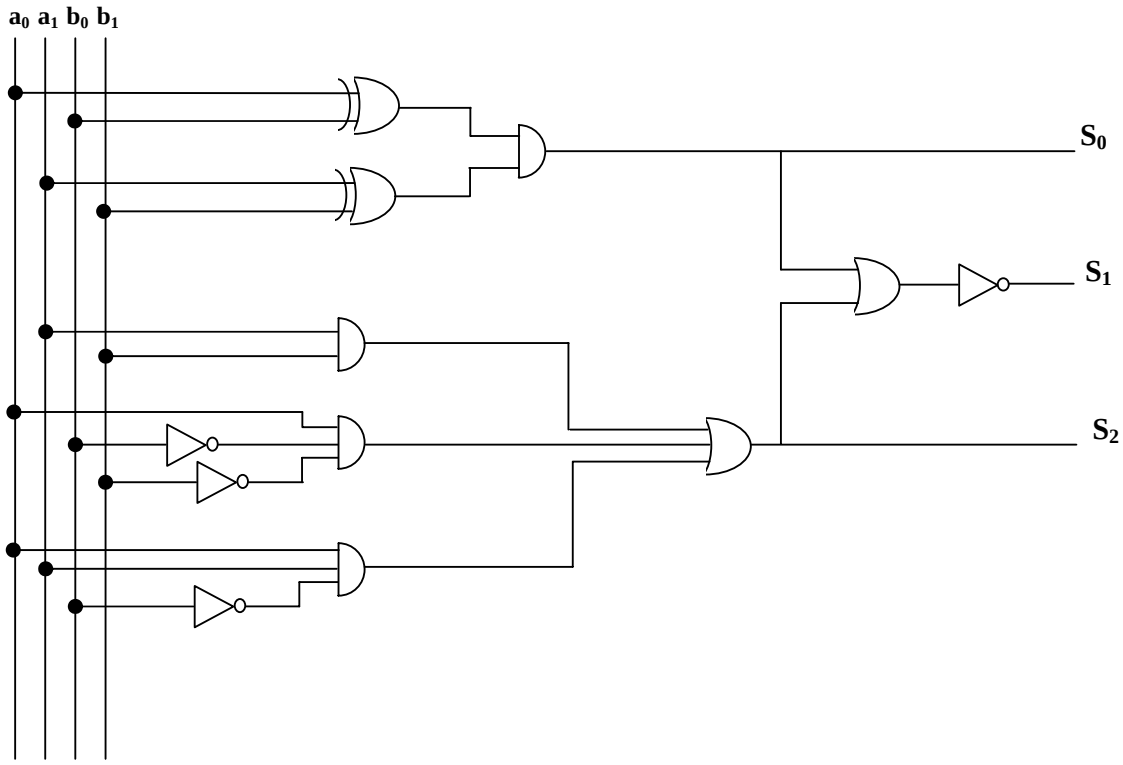
✓ **Equations**

$$S_0 = \overline{a_0} \cdot \overline{a_1} \cdot \overline{b_0} \cdot \overline{b_1} + a_0 \cdot \overline{a_1} \cdot b_0 \cdot \overline{b_1} + \overline{a_0} \cdot a_1 \cdot \overline{b_0} \cdot b_1 + a_0 \cdot a_1 \cdot b_0 \cdot b_1$$
$$= \overline{a_1} \cdot \overline{b_1} \cdot (\overline{a_0} \cdot \overline{b_0} + a_0 \cdot b_0) + a_1 \cdot b_1 \cdot (\overline{a_0} \cdot \overline{b_0} + a_1 \cdot b_0)$$
$$= \overline{(a_0 \oplus b_0)} \ \overline{(a_1 \oplus b_1)}$$

$$S_1 = a_0 \cdot \overline{a_1} \cdot \overline{b_0} \cdot \overline{b_1} + \overline{a_0} \cdot a_1 \cdot \overline{b_0} \cdot \overline{b_1} + a_0 \cdot a_1 \cdot \overline{b_0} \cdot \overline{b_1} + a_0 \cdot \overline{a_1} \cdot b_0 \cdot \overline{b_1}$$
$$+ a_0 \cdot a_1 \cdot b_0 \cdot \overline{b_1} + a_0 \cdot a_1 \cdot b_0 \cdot \overline{b_1}$$
$$= a_1 \cdot \overline{b_1} \cdot + a_0 \cdot b_0 \cdot \overline{b_1} + a_0 \cdot a_1 \cdot \overline{b_0}$$

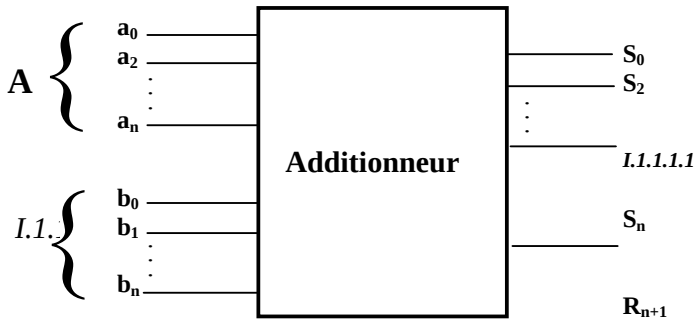
$$S_2 = \overline{S_0 + S_1}$$

✓ **Logigramme**



VIII Les Additionneurs

Un additionneur est un circuit combinatoire qui additionne dans sa sortie **S** deux éléments binaires **A** et **B**, tout en plaçant le retenue de l'opérateur à la sortie **R.** Ils sont d'une grande importance non seulement dans l'ordinateur mais aussi dans des nombreux systèmes traitant des données numériques.



En **base 2** l'addition de deux bits s'écrit :

$$\begin{cases} 0+0 &= 00 \\ 0+1 &= 01 \\ 1+0 &= 01 \\ 1+1 &= 10 \end{cases}$$

Comme en décimal, nous devons donc tenir compte d'une éventuelle retenue (carry). La figure suivante montre la décomposition de l'addition de deux nombres binaires de quatre bits.

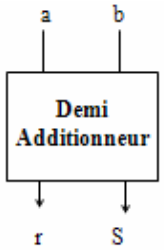
$A = a_3 a_2 a_1 a_0$
$B = b_3 b_2 b_1 b_0$

	r_3	r_2	r_1	r_0		*retenus*
		a_3	a_2	a_1	a_0	*nombre A*
+						
		b_3	b_2	b_1	b_0	*nombre B*
=	r_3	S_3	S_2	S_1	S_0	*Somme* $S = A + B$

L'addition des deux bits de bas poids (LSB : Least Significant Bit) $a_0 \, et \, b_0$, donne un résultat partiel S0 et une retenue r0. On forme ensuite la somme des deux bits $a_1 \, et \, b_1$ et de la retenue r_0. Nous obtenons un résultat partiel S_1 et une retenue r_1. Et ainsi de suite, nous obtenons un résultat sur quatre bits S et une retenue r_3.
Considérons la cellule symbolisée sur la figure suivante, comptant deux entrées $a \, et \, b$ les deux bits à sommer et deux sorties S le résultat de la somme et r la retenue.

VIII.1 Demi-Additionneur (2 bits)

Considérons la cellule symbolisée sur la figure suivante, comptant deux entrées $a \, et \, b$ les deux bits à sommer et deux sorties S le résultat de la somme et r la retenue.



✓ **Table de vérité :**

A	B	S	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

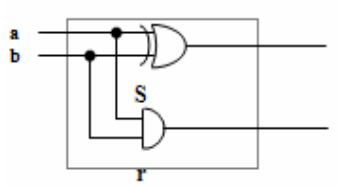
✓ Equations des sorties :

$$S = \bar{a}.b + a.\bar{b}$$
$$= a \oplus b$$

$$r = a.b$$

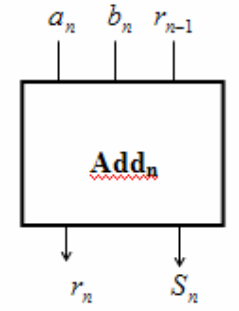
✓ **Logigramme :**

✓



VIII.2 Réalisation d'un Additionneur complet (2 bits)

Il faut en fait tenir compte de la retenue des bits de poids inférieurs, un circuit additionneur du **rang n** doit donc comporter trois entrées et deux sorties, comme représenté sur la figure suivante :



✓ Table de vérité :

a_n	b_n	r_{n-1}	S_n	r_n
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

✓ Equations des sorties :

$$\begin{aligned} S &= a \oplus b \oplus r_{n-1} \\ r_n &= ab + (a \oplus b) r_{n-1} \end{aligned}$$

✓ Logigramme :

Ce serait possible en combinant deux demi-additionneur comme présentés ci-dessous. En pratique pour minimiser le nombre de composants, ou de portes dans un circuit intégré, un tel additionneur est réalisé directement.

Chapitre V

Système Logique séquentiel

Objectifs

A la fin de la séance l'étudiant doit être capable de :

- Comprendre le fonctionnement des bascules
- Simuler les fonctions logiques séquentielles

Pré requis

L'étudiant est supposé connaître :

- Les systèmes de numération et les codes
- Logique Booléenne

Durée

Deux heures et 30 minutes

Eléments de contenu

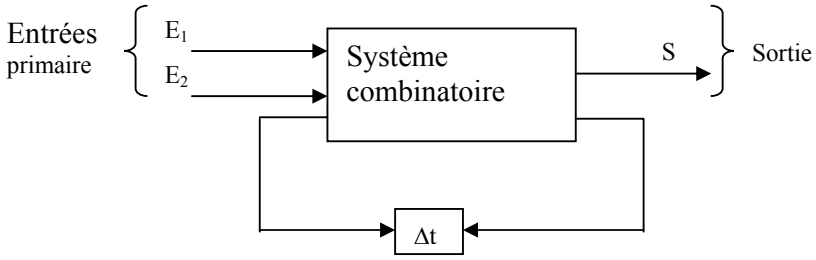
- Les différentes bascules
- Les équations logiques séquentielles

Chapitre V:

Système Logique séquentiel

I Introduction

Avec la logique combinatoire : la fonction de sortie à t ne dépend que des entrées à t, exemple : Un appui sur un bouton allume la lumière, un deuxième appui l'éteint.
Avec la logique séquentiel les sorties actuelles dépendent non seulement des états des entrées (capteurs, boutons poussoirs…) appelées entrées primaires mais encore des réactions provenant des états des précédentes appelées entrées secondaires : système possède une fonction mémoire.



Exemple : Appel d'un ascenseur ; on appuie sur le bouton, l'appel est enregistré et le voyant s'allume. Si on relâche le bouton, le voyant reste allumé, il y a donc mémorisation.

II Les bascules

Une bascule est un système séquentiel constitué par un ou deux entrée et une sortie notée Q et son complément Q. Leur rôle est de mémoriser une information élémentaire.
On distingue des bascules à fonctionnement synchrone et à fonctionnement asynchrones :

- **Synchrones : La** sortie d'un système à fonctionnement synchrone réagit à la modification des variables d'entrée uniquement quand un signal Horloge devient actif. (Il y a obligatoirement une horloge qui est considérée comme variable d'entrée).

- **Asynchrones :** La sortie d'un système à fonctionnement asynchrone répond immédiatement à toute modification des variables d'entrées primaires. (Pas d'horloge ou bien il y a une horloge mais la sortie peut changer indépendamment de l'horloge).

On étudiera quatre grand types de bascules : **RS**, **D**, **T** et **JK**.

II.1 Bascule RS

La bascule RS est une bascule asynchrone (sans entrée d'horloge). C'est la bascule élémentaire, qui constitue la base de tous les autres types de bascules. La bascule RS peut être réalisée avec des portes OU-NON (NOR) ou avec des portes ET-NON (NAND).

S : Set mise à 1 de Q

R : Reset mise à 0 de Q (effacement)

- Si on applique R S = 1 0 ; la sortie Q = 0 et $\overline{Q}$ = 1,
  R est l'entrée de mise à 0 de la bascule (RESET)
- Si on revient à R = 0 et toujours S = 0, la sortie reste Q = 0 et $\overline{Q}$ = 1;
- Si on applique S = 1 et R = 0, la sortie Q = 1 et $\overline{Q}$ = 0;
  S est l'entrée de mise à 1 de la bascule (SET)
- Si on revient à S = 0 et R = 0, la sortie Q = 1 c'est la valeur enregistrée dans la mémoire,
- Pour RS = 11 est interdite, le fonctionnement de la bascule est non spécifié ;

On ne peut pas prévoir le résultat, la porte la plus rapide impose son état (on doit donc tenir compte des temps de propagation); donc, c'est une configuration interdite (forme indéterminée).

Table de vérité

R	S	Q_n	Q_{n+1}	$/Q_{n+1}$	Description	
0	0	0	0	1	Etat précédent	
0	0	1	1	0	Etat précédent	
0	1	0	1	0	Enclenchement	
0	1	1	1	0	Maintient à 1	
1	0	0	0	1	Maintient à 0	
1	0	1	0	1	Déclenchement	
1	1	0	ϕ	ϕ	Indéterminé	Inutilisable
1	1	1	ϕ	ϕ	Indéterminé	

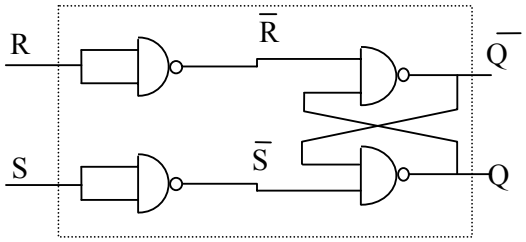
Equation

Q_n \ RS	00	01	11	10
0	0	1	ϕ	0
1	1	1	ϕ	0

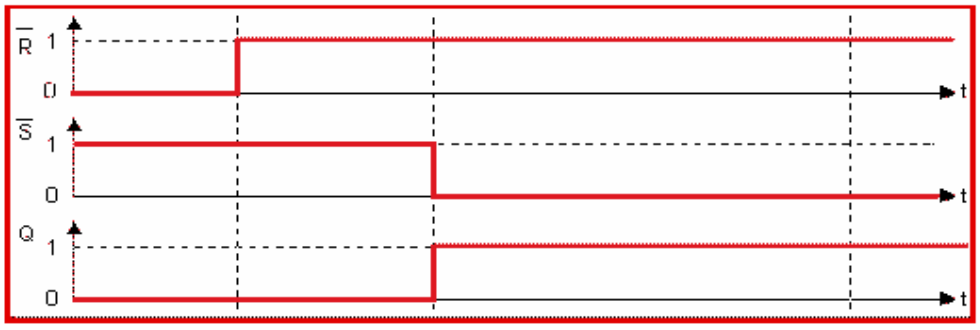
$$Q_{n+1} = Q_n.\overline{R} + S$$

Logigramme

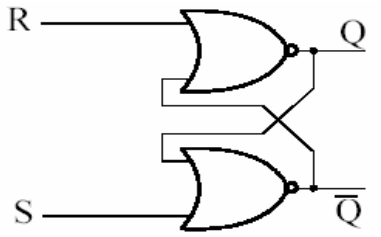
La bascule RS avec des portes **NAND**



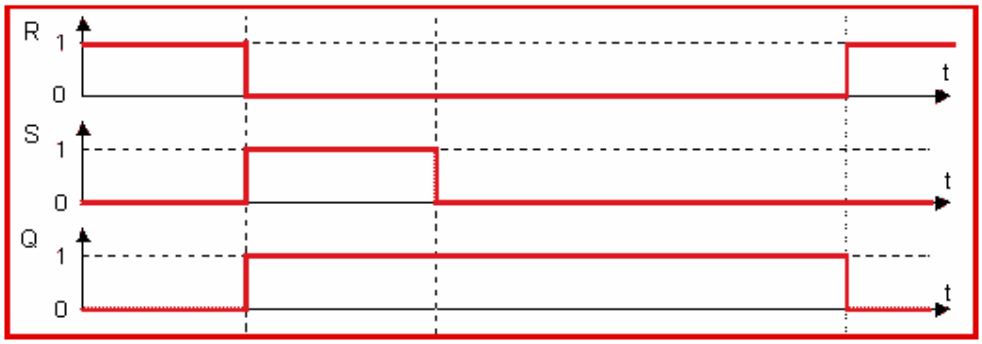
Chronogramme :



La bascule RS avec des portes **NOR** :



Chronogramme :



Exemple : Une alarme de voiture : le voleur ouvre la porte, l'alarme se met à sonner. Même si le voleur referme la porte, l'alarme doit continuer à sonner, il y a donc mémorisation. Seul le propriétaire de la voiture pourra arrêter l'alarme en appuyant sur un bouton spécial. La bascule utilisée ici est une bascule RS.

II.2 Bascule RSH

Fonctionnement: Les changements d'état de la sortie sont synchronisés sur le signal horloge H actif au niveau Haut (1): signal de type impulsion.

- Si H=0: les signaux R et S sont sans effet.
- Si H=1: réaction de la sortie suivant la combinaison (R,S).

==> H est par conséquent le signal de validation d'une bascule RS synchrone.

* La bascule RSH à commande sur front montant:

Entrées			Sorties	Opération sur Q
H	R	S	Q_{t+1}	(Fonction)
0	X	X	Q_t	mode mémoire
1	X	X	Q_t	mode mémoire
↑	1	0	0	mise à 0 sur front montant
↑	0	1	1	mise à 1 sur front montant
↑	1	1	1	opération interdite

II.3 Bascule D

La bascule D est une bascule synchrone (avec une entrée d'horloge) à une seule entrée de donnée : l'entrée **D** (*D=Data=Donnée*). Elle recopie avec un retard d'une période d'horloge l'unique signal appliqué à son entrée D.

Elle supprime la combinaison interdite de la bascule RS, en ne gardant que les 3 fonctions utiles :

- la mise à 0
- la mise à 1
- la mémorisation

Table de vérité

D	Q_n	Q_{n+1}	$/Q_{n+1}$	Description
0	0	0	1	Maintient à 0
0	1	0	1	Déclenchement
1	0	1	0	Enclenchement
1	1	1	0	Maintient à 1

La bascule D élimine la mémorisation et le cas indéterminé dans la bascule RS.

Equation

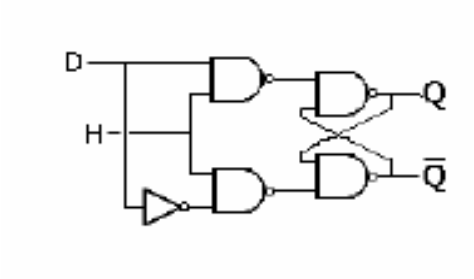
Q_n \ D	0	1
0	0	1
1	0	1

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

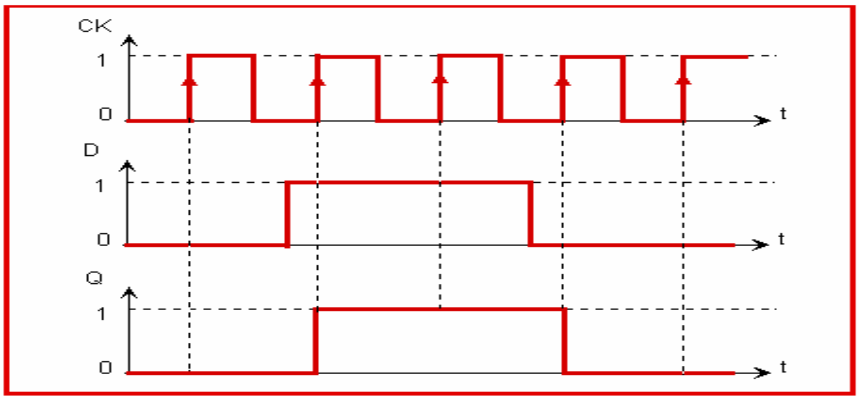
En mettant S = D et R = $\overline{D}$ dans l'équation de la bascule RS on aura :

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

Logigramme



Chronogramme :



II.4 Bascule JK

Contrairement à la bascule RS, condition J=K=1, ne donne pas lieu à une condition indéterminée, mais par contre la bascule passe à l'état opposée.

En fonction de la valeur appliquée sur les entrées J et K (4 combinaisons possibles), on choisit le mode de fonctionnement de la bascule parmi les 4 :

- Mémorisation
- Mise à 0
- Mise à 1
- Basculement

Table de vérité

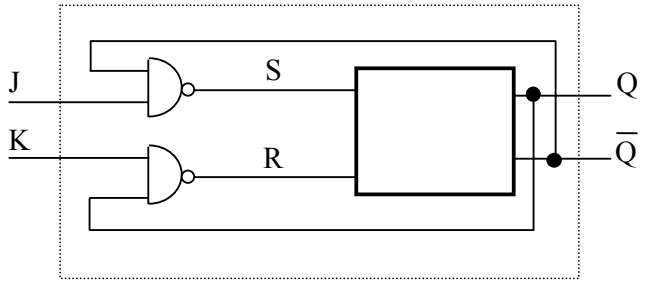
J	K	Qn	Qn+1	/Qn+1	Description
0	0	0	0	1	Etat précédent
0	0	1	1	0	Etat précédent
0	1	0	0	1	Maintient à 0
0	1	1	0	1	Déclenchement
1	0	0	1	0	Enclenchement
1	0	1	1	0	Maintient à 1
1	1	0	1	0	Enclenchement
1	1	1	0	1	Déclenchement

Equation

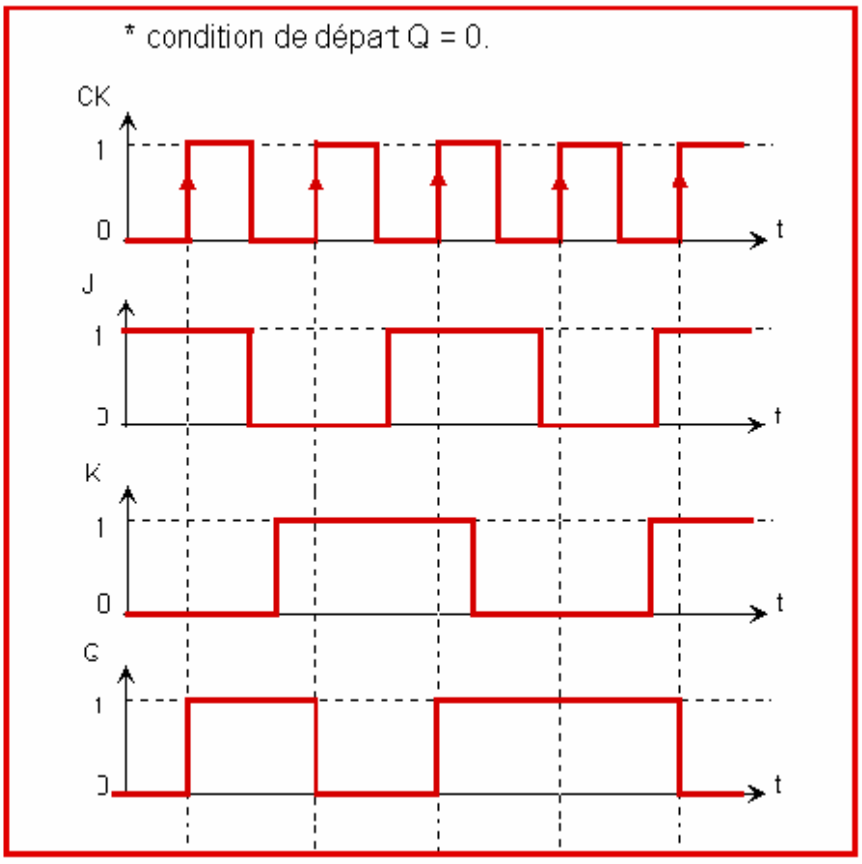
Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$Q_{n+1} = \overline{K}.Q_n + J.\overline{Q_n}$$

Logigramme



Chronogramme :



La sortie n'est activé qu'au front montant de l'horloge : J est l'entrée de mise à 1 et K celle pour la mise à 0.

II.5 Bascule T

On obtient la bascule T, en reliant les entrées J et K d'une bascule JK. A chaque impulsion d'entrée, la sortie est inversée. (*utile pour les compteurs*)

Table de vérité

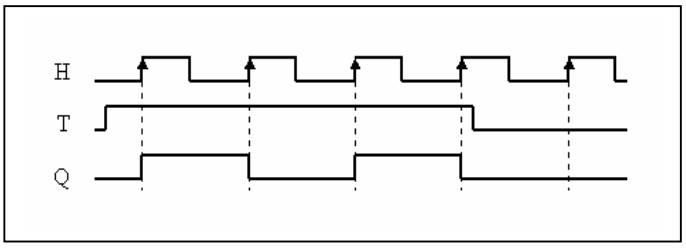
T	Qn	Qn+1	/Qn+1	Description
0	0	0	1	Maintient à 0
0	1	1	0	Maintient à 1
1	0	1	0	Enclenchement
1	1	0	1	Déclenchement

Equation

En remplaçant J et K par T dans l'équation de la bascule JK on obtient :

$$Q_{n+1} = \overline{T}.Q_n + T.\overline{Q}_n$$

Chronogramme :



Chapitre VI

Circuits Logiques Séquentiels : Registres et Compteurs

Objectifs

A la fin de la séance l'étudiant doit être capable de :

- Connaître les Compteurs et les registres
- Comprendre leurs fonctionnements
- Analyser les compteurs et les registres

Pré requis

L'étudiant est supposé connaître :

- Logique Booléenne
- Système logique séquentiel

Durée

Trois heures et 30 minutes

Eléments de contenu

- Les différents registres à décalage
- Les compteurs synchrones
- Les compteurs asynchrones

Chapitre VI:

Circuits Logiques Séquentiels :

Registres et Compteurs

I Introduction

Il s'agit maintenant d'examiner des dispositifs traitant plusieurs informations à la fois et comportant pour ce faire :

- un groupe des bascules alimentées par une horloge,
- un réseau combinatoire d'entrée qui élabore les commandes d'excitation des bascules,
- un réseau de sortie donnant l'état des bascules.

II Les registres

L'application la plus courante des bascules est la mémorisation des données binaires. Les données sont stockées dans des circuits numériques à base des bascules : ce sont les registres. Le chargement d'un registre est l'enregistrement des données dans le circuit.
L'opération fondamentale des registres est le transfert des données entre les circuits : écriture ou lecture.

II.1 Registres à décalage

Si on désire mémoriser une information comportant un trop grand nombre de bits, ou qu'elle doit être transportée à grande distance, on effectue '' un chargement série''.
L'information peut être chargée de deux manières dans ce type de registre :

- **Entrée parallèle (PI):** ou chargement parallèle
- **Entrée série (SI)**: ou chargement série

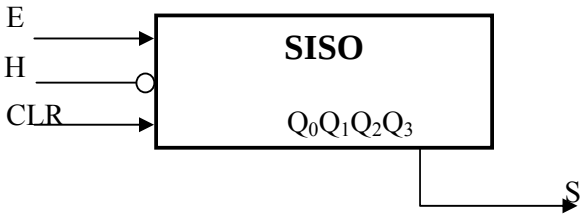
De même l'information peut être lue de deux manières

- **Sortie parallèle (PO) : lecture parallèle**
- **Sortie série (SO) : ou lecture série**

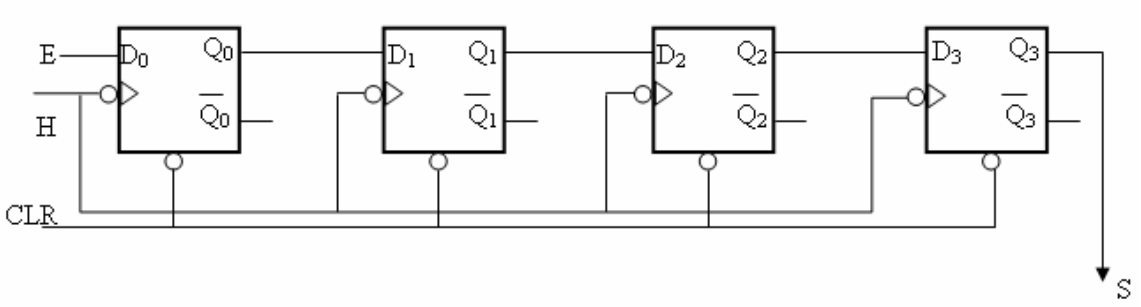
II.1.1 Registres à décalage SISO

Ce type de registre à décalage possède une seule entrée donnée binaire et une seule sortie. Donc, il est caractérisé par un transfert série des données vers la droite ou vers la gauche.

Le schéma fonctionnel:



Exemple : un registre à décalage de 4 bits s'utilise pour le stockage d'une information binaire de 4 bits. Le circuit est composé de 4 bascules D synchrones. Les entrées des bascules sont commandées par le même signal d'horloge. Les bascules fonctionnent au front montant d'horloge.



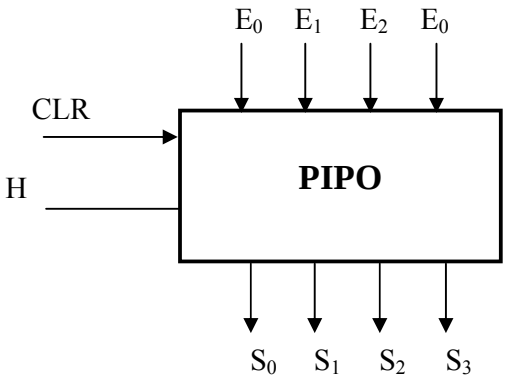
Remarque :

Un registre à décalage à droite et à gauche permet d'effectuer des multiplications et des divisions entiers par des puissances de 2. En effet, une multiplication par 2 est équivalente à un décalage vers la gauche et une division par 2 à un décalage vers la droite .

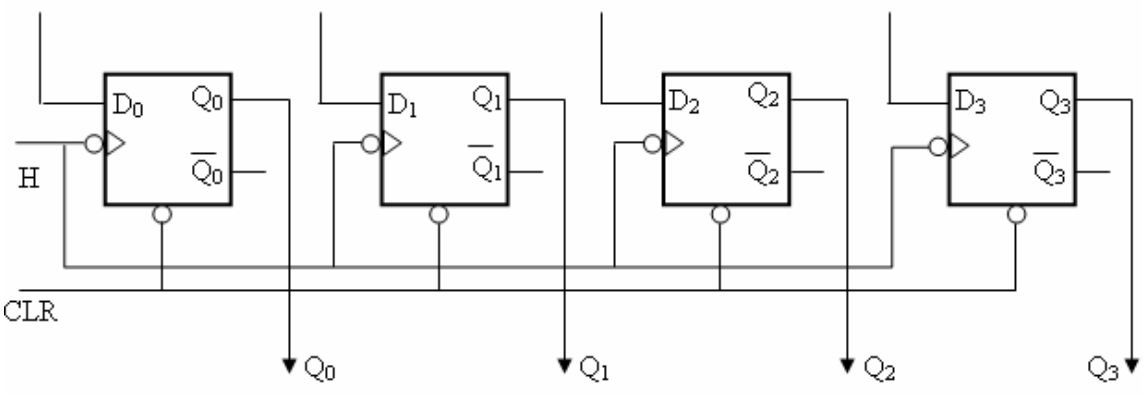
II.1.2 Registre à décalage PIPO

C'est un registre qui possède des entrées parallèles de données et des sorties parallèles.

Le schéma fonctionnel:



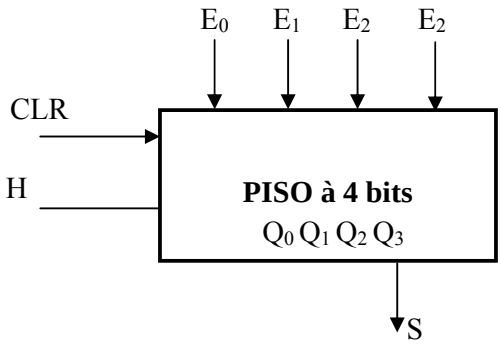
Exemple : le circuit suivant est un registre à entrées/sorties parallèles de 4 bits, constitué de 4 bascules D synchrones pouvant charger chacune un bit de données.



II.1.3 Registre à décalage PISO

Ce registre a plusieurs entrées des données et une seule sortie. Le chargement des données se fait donc en parallèle et le transfert à la sortie en série.

Le schéma fonctionnel:

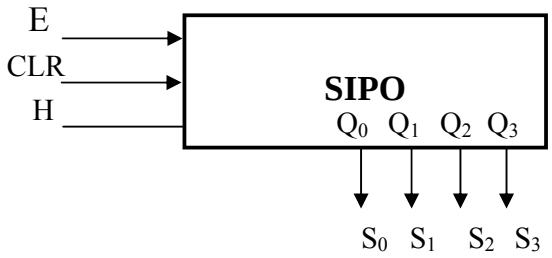


Exemple : Le circuit intégré 74165 est registre de 8 bits, il possède 8 entrées données (A,B,C,D,E,F,G,H) est une seule sortie (Q_H). La constitution interne utilise 8 bascules R-S et certaines portes logiques pour commander le circuit.

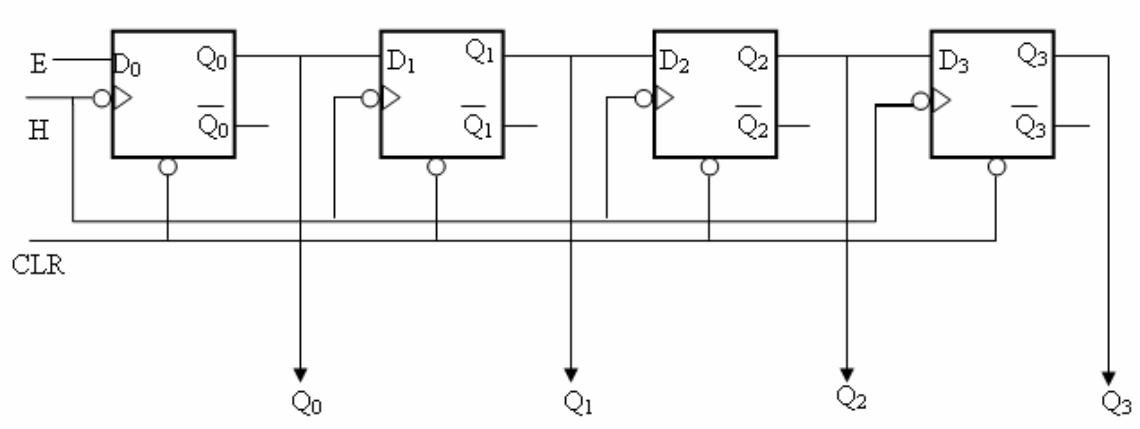
II.1.4 Registre à décalage SIPO

L'Entrée est écrite sur un seul bit. La sortie est lue plusieurs bits.

Le schéma fonctionnel:



Le schéma avec bascule D:



Exemple : le circuit intégré 74 164 est un registre de 8 bits. Il comporte deux entrées série A et B qui sont combinées dans une porte ET. La valeur de leur produit logique correspond à l'entrée principale.

Pour décaler la donnée série de la bascule Q_A à Q_H, il faut 8 impulsions d'horloge.

Si A et B sont au niveau 1→la donnée série est A = B = 1. La transmission de cette valeur aux entrée en fonction de chaque impulsion d'horloge : (A et B sont maintenues à 1).

Imp D'Hlge	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	Q_H
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0
..								
8	1	1	1	1	1	1	1	1

III Les Compteurs

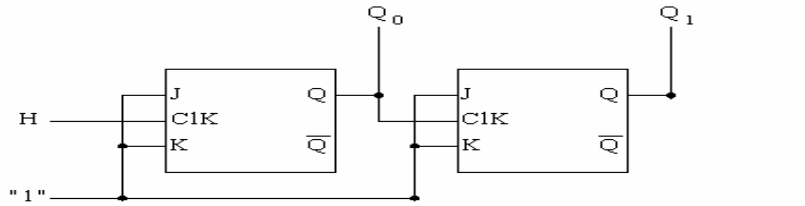
Un compteur est un ensemble de n bascules interconnectées par des portes logiques. Ainsi, ce sont des circuits séquentiels qui permettent de faire le comptage des impulsions en fonctions des états précédents. L'élément de base qui est la bascule, permet aussi de mémoriser les états précédents.

Capacité d'un compteur = N: C'est le nombre maximum d'impulsions qu'il peut enregistrer avant de revenir à son état initial. Un compteur à « n » bascules appelé COMPTEUR MODULO N = 2n, peut prendre N états: de 0 à N-1; la Nième impulsion remet le compteur à 0. On doit utiliser des portes logiques pour remettre à 0 le comptage et pouvoir compter de nouveau.

III.1 Compteurs asynchrones modulo N = 2n :

Le changement d'état des sorties n'est pas synchronisé par un seul signal d'horloge. En effet, la sortie de chaque bascule correspond à l'entrée d'horloge de la bascule suivante. Le nombre des bascules utilisées est égal au degré n de puissance de 2.

Exemple : compteur asynchrone modulo 4 : Le compteur comporte 2 bascules J-K à front descendent. La condition J=K=1 fait inverser l'état de sortie des bascules à chaque front descendant d'horloge.



Le signal d'horloge d'une bascule de rang i est le signal de sortie de la bascule de rang i-1.

Le tableau suivant comporte l'état des sorties Q0 et Q1 selon le nombre d'impulsions d'horloge :

Nombre d'impulsions	Q_1	Q_0
0	0	0
1	0	1
2	1	0
3	1	1
4	0	0
…	…	…

III.2 Compteurs asynchrones modulo N ≠ 2n

Ce type doit comporter un circuit de remise à 0 lorsqu'il atteint N impulsions. Le nombre des bascules nécessaires est (n), telle que $2^{n-1} < N < 2^n$.

On lui ajoute un asservissement de l'entrée Clear pour remettre le compteur à Zéro tous les N coups.

Exemple : compteur asynchrone modulo 10

Le circuit doit compter de 0000 jusqu'à 1001 en binaire. Puisque 10 se trouve entre 2^3 et 2^4, on doit utiliser donc 4 bascules. Le circuit de remise à 0 utilise les bornes Reset de chaque bascule.

La suite des nombres est :

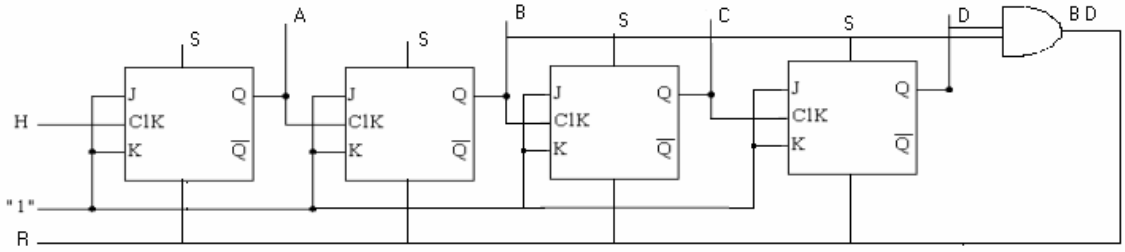
D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0

Etat des sorties qui remet à 0 le compteur

Cette dernière combinaison correspond donc à : D = 1, C = 0, B = 1 et A = 0. On tire la condition de remise à Zéro : D = B = 1.

Donc $R = D.B$

Le circuit logique est donné par la figure suivante :



Circuit Logique d'un compteur Asynchrone modulo 10

III.3 Compteurs synchrones

Appelés encore compteurs parallèles. Dans ce type, toutes les bascules composant le compteur reçoivent en même temps le signal d'horloge. Ceci exige l'existence d'un mécanisme qui sélectionne, à chaque impulsion d'horloge, laquelle des bascules qui doit changer d'état ou basculer. Pour cela, on utilise les entrées J et K des bascules pour réaliser ce mécanisme.

Exemple : Compteur synchrone modulo 16

On utilise 4 bascules J-K pour sortir le comptage sur 4 bits. Les nombres comptés sont de 0000 jusqu'à 1111. Dans ce cas, on doit tenir compte de l'état de chaque bascule pour chaque

impulsion d'horloge. Le tableau suivant nous montre les combinaisons des sorties des bascules :

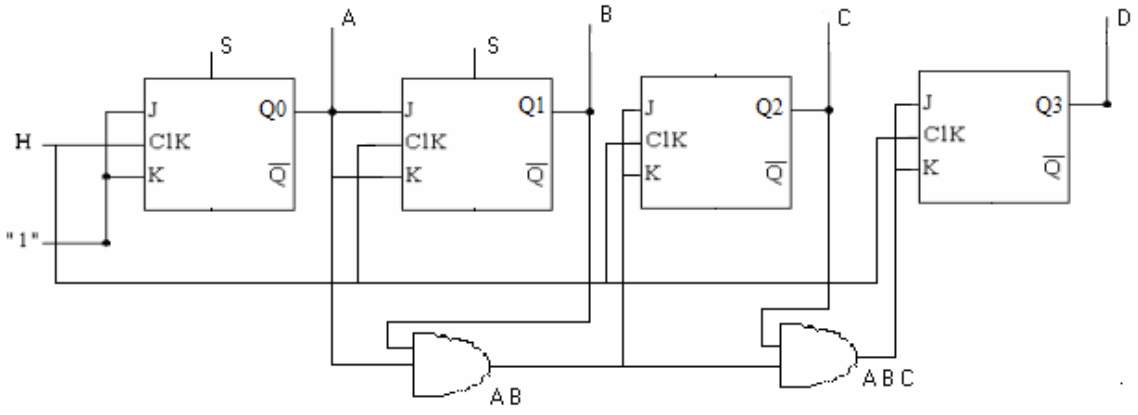
Nombre d'impulsions	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.

On constate que la bascule A doit basculer à chaque coup d'horloge. Il suffit donc de mettre J = K = 1 dans cette bascule.

Quand à la bascule B, elle ne change d'état qu'après deux coups d'horloge par rapport à la bascule A. On utilise alors la sortie complémentée de A comme entrée de J et K de la bascule B.

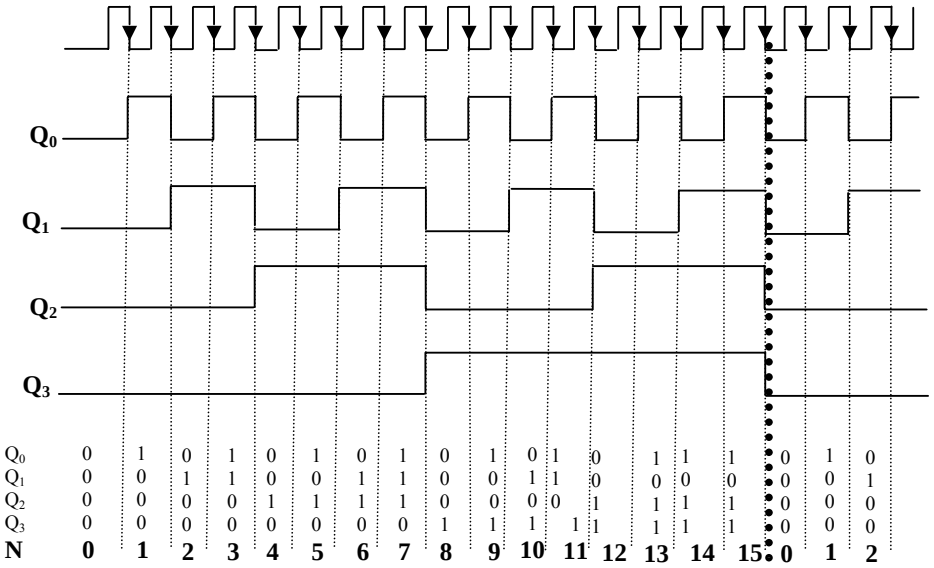
On rappelle que lorsque J=K=0, la bascule maintient l'état précédent. On procédera de la même façon pour les bascules C et D.

CLK étant le signal d'horloge.



Circuit Logique du Compteur synchrone modulo 16

Le chronogramme correspondant



http://maths-sciences.fr

LE SYSTÈME BINAIRE

I) Introduction

L'homme a toujours eu besoin de compter et il a inventé la numération décimale sur le modèle des dix doigts de nos mains. On pourrait toutefois noter que l'on a en fait 20 doigts (pied et main). On a aussi inventé la numération qui lui correspond, appelée numération vicésimale. Elle n'a pas eu le succès de la numération décimale mais on en a hérité quatre-vingt (au lieu d'octante ou huitante), quatre vingt dix (nonante).
L'écriture décimale nécessite l'existence de 10 chiffres (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Toute écriture dans un système supérieur à 10 nécessite la création de nouveaux idéogrammes. On les remplace souvent par des lettres.

Base 2	Base 3	Base 4	Base 5	Base 6	Base 7	Base 8	Base 9	**Base 10**	Base 11	Base 12	Base 13
0	0	0	0	0	0	0	0	**0**	0	0	0
1	1	1	1	1	1	1	1	**1**	1	1	1
10	2	2	2	2	2	2	2	**2**	2	2	2
11	10	3	3	3	3	3	3	**3**	3	3	3
100	11	10	4	4	4	4	4	**4**	4	4	4
101	12	11	10	5	5	5	5	**5**	5	5	5
110	20	12	11	10	6	6	6	**6**	6	6	6
111	21	13	12	11	10	7	7	**7**	7	7	7
1000	22	20	13	12	11	10	8	**8**	8	8	8
1001	100	21	14	13	12	11	10	**9**	9	9	9
1010	101	22	20	14	13	12	11	**10**	A	A	A
1011	102	23	21	15	14	13	12	**11**	10	B	B
1100	110	30	22	20	15	14	13	**12**	11	10	C
1101	111	31	23	21	16	15	14	**13**	12	11	10
1110	112	32	24	22	20	16	15	**14**	13	12	11
1111	120	33	30	23	21	17	16	**15**	14	13	12
10000	121	100	31	24	22	20	17	**16**	15	14	13
10001	122	101	32	25	23	21	18	**17**	16	15	14
10010	1000	102	33	30	24	22	20	**18**	17	16	15

Plus la base est importante et moins il faut de chiffre pour écrire un nombre. Exemple 1000 :

En	Ecriture
Base 2 (système binaire)	1 111 101 000
Base 3 (système ternaire)	1 101 001
Base 4 (système quaternaire)	33 220
Base 5 (système quinaire)	13 000
Base 6 (système sénaire)	4344
Base 7 (système septénaire)	2626
Base 8 (système octonaire)	1750
Base 9 (système nonaire)	1331
Base 10 (système décimal)	1000
Base 11 (système undécimal)	82A
Base 12 (système duodécimal)	6B4

II) Systèmes les plus courants

Le système duodécimal : Si le système décimal n'avait pas été universellement adopté, il aurait pu avoir un certain succès dans la mesure ou 12 a un plus grand nombre de diviseurs que 10.

Le système héxadécimal : (16 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Ce système est utilisé en informatique.

Le système sexagésimal : Ce système à base 60 fut élaboré par les babyloniens. Il est encore utilisé pour les mesures de temps et d'angle (heures, minutes, secondes).

Le système binaire : Ce système est très ancien et son existence en Chine remonterait au moins à 25 siècles avant J-C. Au XVII^{e} siècle, Leibniz essaiera de l'imposer sans succès. Ce système connaît son apogée avec l'apparition de l'électronique. Dans les transistors, 0 correspond à l'absence de courant et 1 au passage du courant.
Par convention un nombre élevé à la puissance 0 est égal à 1. Il en découle que tout nombre peut s'écrire sous la forme d'une somme de puissances de 2.

III) Convertir un nombre décimal en binaire

L'écriture binaire repose sur le fait que tout nombre peut s'écrire sous la forme d'une somme de puissances de 2.

1ère méthode

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$

⇨ Comment s'écrit 97 en nombre binaire ?

On commence par chercher la plus grande puissance de 2 contenue dans 97.
Il s'agit de $2^6 = 64$.
On soustrait 64 à 97. Il nous reste 33.
On cherche la plus grande puissance de 2 contenue dans 33.
Il s'agit de $2^5 = 32$.
On soustrait 32 à 33. Il reste 1.
La plus grande puissance de 2 contenue dans 1 est 2^0.
Il en résulte que $97 = \mathbf{1}\times 2^0 + \mathbf{0}\times 2^1 + \mathbf{0}\times 2^2 + \mathbf{0}\times 2^3 + \mathbf{0}\times 2^4 + \mathbf{1}\times 2^5 + \mathbf{1}\times 2^6$

L'écriture binaire de 97 est donc : **1 0 0 0 0 1 1**.

2ème méthode

⇨ Comment s'écrit 437 en nombre binaire ?

On prépare un tableau avec les puissances de 2 :

…	1024	512	256	128	64	32	16	8	4	2	1

On reconstitue le nombre décimal à convertir en plaçant des « 1 » dans les colonnes adéquates du tableau : 437 = 256 + 64 + 16 + 8 + 2 + 1

…	1024	512	256	128	64	32	16	8	4	2	1
	0	0	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**

L'écriture binaire de 437 est donc : **1 0 1 0 1 1 0 1 1**.

3ème méthode

Il existe un autre procédé plus rapide pour transférer un nombre du système décimal dans un autre système. Cette méthode consiste à diviser le nombre donné par la base tant que c'est possible. On rassemble ensuite les restes en partant de la fin et on obtient l'écriture dans la nouvelle base.

⇨ Comment s'écrit 193 en nombre binaire ?

193	2
13	96
1	16
	0

96	2
48	2
8	2
0	4
	0

24	2
12	2
6	2
0	3
	1

12	2
0	6
	0

6	2
0	3
	1

3	2
1	1

L'écriture binaire de 193 est donc : **1 1 0 0 0 0 0 1**.

IV) Convertir un nombre binaire en décimal

Soit 1011 le nombre binaire à convertir. Cette écriture est appelée écriture implicite. Pour trouver l'équivalent décimal il suffit d'employer l'écriture explicite.

1 0 1 1 correspond à : $\underbrace{1\times 2^3}_{8}+\underbrace{0\times 2^2}_{0}+\underbrace{1\times 2^1}_{2}+\underbrace{1\times 2^0}_{1}$ soit $1\ 0\ 1\ 1_2 = 11_{10}$

Remarque : quand on travaille simultanément dans différentes bases, il faut indiquer, en indice, la base dans laquelle est écrit chacun des nombres.

V) Utilisation

Nous l'avons dit en introduction que le système binaire trouve son utilité dans tous les domaines liés à l'informatique et l'électronique. Le langage binaire est utilisé pour tout transport d'information par voie électronique. Par exemple les lettres de notre alphabet sont codés par nos ordinateurs en binaire selon les codes ci-dessous :

A	**B**	**C**	**D**	**E**	**F**	**G**
01000001	01000010	01000011	01000100	01000101	01000110	01000111
H	**I**	**J**	**K**	**L**	**M**	**N**
01001000	01001001	01001010	01001011	01001100	01001101	01001110
O	**P**	**Q**	**R**	**S**	**T**	**U**
01001111	01010000	01010001	01010010	01010011	01010100	01010101
V	**W**	**X**	**Y**	**Z**		
01010110	01010111	01011000	01011001	01011010		