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

Université A. Mira de BEJAIA

Faculté de Technologie

Département de Génie Electrique.

Cours :

Electronique Numérique

Dr. TAFININE Farid
Maître de Conférences classe B

Année universitaire 2016-2017

TABLE DES MATIERES

Introduction générale

Ce polycopie est le support écrit du cours « Electronique Numérique : logique combinatoire et séquentielle» destiné aux étudiants de la deuxième année LMD Electronique, Automatique et Télécommunication du département Génie Electrique, Faculté de Technologie de l'université A/Mira de Béjaia.

Il s'articulera autour de sept chapitres. Le programme de ce cours a été conçu en s'appuyant sur diverses références : des ouvrages reconnus dans la discipline, mais aussi et surtout des ressources en ligne.

Le chapitre 1 fait l'objet de la présentation de l'ensemble des bases de numérotation : base binaire, octale, décimale, hexadécimale. En plus, il explique les règles de passage entre les bases et l'ensemble des opérations arithmétiques effectuées en binaire.

Le chapitre 2 est consacré à la représentation des systèmes combinatoires à savoir les différentes fonctions logiques (AND, OR, XOR,…), ainsi que les propriétés et théorèmes de l'algèbre de BOOLE.

Le chapitre 3 présente les quatre formes canoniques des fonctions logiques, suivi par les deux méthodes les plus utilisées dans la simplification des fonctions logiques (la méthode algébrique et la méthode de karnaugh).

Le chapitre 4 introduit d'une manière détaillée l'ensemble de circuits combinatoires de base qui sont : les circuits arithmétiques, les circuits d'aiguillage, les transcodeurs, les décodeurs, les comparateurs ainsi que les afficheurs à 7 segments.

Le chapitre 5 traite la logique séquentielle par l'introduction de la notion du temps, l'élément de base dans ce chapitre est la bascule. Les différents types de bascules ont été exposés, on cite : la bascule RS, D, JK, T, Maitre-Esclave, …

Le chapitre 6 est dédié aux compteurs binaires, deux familles de compteurs ont été étudiés : les compteurs asynchrones et les compteurs synchrones.

Le dernier chapitre aborde la notion des registres, deux types de registres ont été étudiés : les registres de mémorisation et les registres à décalage avec ses différentes structures.

Enfin, ce support de cours se clôturera par une conclusion générale et une liste de références bibliographiques suivi par l'ensemble des séries de travaux dirigés.

Chapitre 1 : Systèmes de numérotation

I. Introduction

La numération binaire sert comme base de calcul pour l'étude des circuits en Electronique numérique. C'est pourquoi, avant l'étude des systèmes numériques combinatoires et séquentiels, il convient de rappeler les bases du calcul binaire, ainsi que les différentes bases existantes en pratique.

En général un nombre N dans une base b s'écrit selon la formule générale suivante :

$$N = a_n b^n + a_{n-1} b^{n-1} + ... + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-n} b^{-n} \quad \textbf{(1)}$$

$$N = (a_n, a_{n-1}, ..., a_1, a_0, a_{-1}, ..., a_{-n})_b$$

Les a_i sont les chiffres exprimés dans la base b.

Dans la base b : $0 \leq a_i < b$

Si : b =10, $0 \leq a_i < 10$

Exemple : $834 = 8 . 10^2 + 3 . 10^1 + 4 . 10^0$

II. Les différents systèmes de numération

Base 2 :
Dans cette base on parle de BIT (BInary digiT), elle ne possède que deux chiffres : 0 et 1.
Exemple : $(1011)_2 = 1 . 2^3 + 0 . 2^2 + 1 . 2^1 + 1. 2^0 = 8 + 0 + 2 + 1 = 11.$

$1110{,}101 = 1 . 2^3 + 1 . 2^2 + 1 . 2^1 + 0. 2^0 + 1 . 2^{-1} + 0 . 2^{-2} + 1. 2^{-3} = 8 + 4 + 2 + 0 + 0.5 + 0 + 0.125 = 14.625.$

Base 8 :
Cette base possède 8 coefficients a_i : (0, 1, 2, 3, 4, 5 ,6, 7), un nombre N dans cette base s'écrit :

$$N = \sum_{i=-n}^{+n} a_i . 8^i .$$

Exemple : $(140)_8 = 1 . 8^2 + 4. 8^1 + 0 . 8^0 = 64 + 32 + 0 = 96.$

L'intérêt de ce système est que la base 8 est une puissance de 2 ($8 = 2^3$), donc les poids sont aussi des puissances de 2. Chaque symbole de la base 8 est exprimé sur 3 éléments binaires.

Base 16 ou hexadécimale :

Cette base contient 15 symboles, les coefficients a_i ont pour valeurs :

0, 1, 2, 3, …….9, A, B, C, D, E, F.

Chaque symbole de la base hexadécimale est exprimé en binaire sur 4 bits.

Exemple : $(A4)_{16} = 10. 16^1 + 4 . 16^0 = 164.$

III Règles de passage entre les bases

III.1 Base B *vers base 10*

La règle de conversion est basée sur l'équation (1) càd : multiplication par la base B.

Exemple : $(62, 1)_8 = 6. 8^1 + 2. 8^0 + 1. 8^{-1} = 48 + 2 + 0.125 = 50,125$

III.2 Base 10 vers base B

La règle de conversion est basée cette fois sur la division par la base *B* autant de fois dans le but d'obtenir un quotient nul. Ensuite on écrit les restes de la division dans l'ordre inverse de celui dans lequel ils ont été obtenus. Pour la partie fractionnaire on multiplie par la base *B* jusqu'à obtenir un résultat nul ou la précision souhaitée.

Exemple : (15,125)

15 / 2 = 7 reste 1	↑ sens	0,125 x 2 = 0,25 → 0	
7 / 2 = 3 reste 1	de	0,25 x 2 = 0,5 → 0	sens
3 / 2 = 1 reste 1	lecture	0,5 x 2 = 1,0 → 1	de
1 / 2 = 0 reste 1			↓ lecture

Nous obtenons l'écriture de 15.125 en base 2 : $(1111, 001)_2$

III.3 Base 8 vers base 2

Chaque symbole de la base 8 peut être représenté par 3 BITs, selon la relation de puissance : $8= 2^3$.

Exemple : $(45,07)_8 = \left(\underbrace{100}\ \underbrace{101}, \underbrace{000}\underbrace{111} \right)_2$

Et vis-versa càd : Chaque 3 BITs de la base 2 peut être représenté par un seul symbole de la base 8.

III.4 Base 16 vers base 2

Chaque symbole de la base hexadécimale peut être représenté par 4 BITs, selon la relation de puissance : $16= 2^4$.

Exemple : $(C9, E7)_{16} = \left(\underbrace{1100}\ \underbrace{1001}, \underbrace{1110}\underbrace{0111} \right)_2$

Et vis-versa càd : Chaque 4 BITs de la base 2 peut être représenté par un seul symbole de la base 16.

IV Opérations Arithmétiques en binaire

IV. 1. Addition

L'addition en binaire se fait avec les mêmes règles qu'en décimal. Une retenue sera générer lorsque la somme de deux bits de même poids dépasse la valeur binaire : 1. Cette retenue est reportée sur le bit de poids plus fort suivant, comme est indiqué dans le tableau suivant :

a	b	somme	retenu
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

IV. 2. Soustraction

La soustraction en binaire s'effectue comme en décimal. La différence réside lorsque la quantité à soustraire est supérieure à la quantité dont on soustrait, on emprunte la base qui est égal à 2.

L'opération est indiquée dans le tableau suivant :

a	b	Différence	emprunte
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

IV. 3. La multiplication

L'opération de la multiplication en binaire est illustrée dans le tableau suivant :

a	b	a x b
0	0	0
0	1	0
1	0	0
1	1	1

On remarque que lorsque le bit du multiplieur est nul, le résultat de la multiplication est nul

IV. 4. La division

La division en binaire ressemble à celle en décimal. Elle fait appelle à l'opération de soustraction en binaire ainsi que du décalage, le tableau ci dessous illustre l'opération :

a	b	a / b
0	0	impossible
0	1	0
1	0	impossible
1	1	1

V. Les compléments en binaire

V.1 Le complément Restreint (CR)

Il s'appel aussi le complément à 1. En base 2 il est obtenu en inversant chaque bit du nombre donné.

Exemple : $(1001)_2$ son CR est : $(0110)_2$

V.2 Le complément vrai (CV)

Il s'appel aussi le complément à 2. En base 2 il est obtenu en rajoutant un 1 au complément restreint (CR) du nombre donné.

Exemple : $(1001)_2$ son CV est : $(0110)_2 + 1 = (0111)_2$

VI. Les codes numériques

VI.1 Les codes numériques pondérés

C'est un code dont le rang de chaque bit possède un poids donné, on distingue :

VI.1.1 Le code binaire pur

Le code binaire pur est un code pondéré par des puissances de 2. Pour trouver la valeur du nombre en décimal, il suffit d'additionner (les poids x bits).

Exemple :

$(1011)_2$

Le premier bit a un poids $2^3=8$

Le deuxième bit a un poids $2^2=4$

Le troisième bit a un poids $2^1=2$

Le quatrième bit a un poids $2^0=1$

Donc : 8x1+ 4x0 + 2x1 + 1x1 = 11 en décimal.

VI.1.2 Le code DCB ou BCD (8421)

Dans le code DCB **(Décimal Codé en Binaire)**, chaque chiffre en décimal (0, 1, . . . , 9) est codé en binaire sur 4 bits. C'est un code pondéré avec les poids 8421.

Exemple :

$(97)_{10}= (1001\ 0111)_{DCB}$

$(2015)_{10}= (0010\ 0000\ 0001\ 0101)_{DCB}$

VI.1.3 Code binaire d'Aiken (2421)

Le code Aiken est un code pondéré par les poids suivants « 2421», Il est obtenu comme suit :

– les nombres de 0 à 4 on code en binaire pur ;
– les nombres de 5 à 9 on ajoute 6 et puis on code en binaire pur.

Exemple :

0 est codé : 0000

1 est codé : 0001

2 est codé : 0010

3 est codé : 0011

4 est codé : 0100

5 est codé : 1011

6 est codé : 1100

7 est codé : 1101

8 est codé : 1110

9 est codé : 1111

VI.2 Les codes numériques non pondérés

VI.2.1 Le code de Gray (binaire réfléchi)

Le code Gray, appelé aussi dans certaines littératures : « le code binaire réfléchi », est à un code dit « cyclique ». Ce qui signifie : pour passer d'un nombre au nombre suivant un seul bit change d'état. Il est appelé aussi « code à termes adjacents », il est fortement utilisé dans les tableaux de Karnaugh, mais il ne convient pas pour les opérations arithmétiques.

La séquence sur quatre bits est obtenue par 4 symétries miroir, d'où l'appellation : *binaire réfléchi.*

0 est codé	0000	8 est codé	1100
1 est codé	0001	9 est codé	1101
2 est codé	0011	10 est codé	1111
3 est codé	0010	11 est codé	1110
4 est codé	0110	12 est codé	1010
5 est codé	0111	13 est codé	1011
6 est codé	0101	14 est codé	1001
7 est codé	0100	15 est codé	1000

A/ Règle de conversion du code binaire pur vers le Gray

La règle du passage du code binaire pur vers le code binaire réfléchi est la suivante :

On prend le nombre binaire on fait un décalage d'un bit à gauche, puis on supprime le bit de décalage et on fait l'addition sans retenu.

Exemple :

13 en binaire s'écrit $(1101)_2$

$$\begin{array}{rl} & 1\ 1\ 0\ 1 \\ \textit{somme sans retenue} \oplus & 1\ 1\ 0\ 1 \\ \hline & \end{array}$$

$$(1\ 0\ 1\ 1)_{gray}$$

B/ **Règle de conversion du code Gray vers le binaire pur**

La règle du passage du code Gray vers le binaire pur est la suivante : On prend le bit du poids fort du nombre en gray, on fait l'addition sans retenu avec le bit qui vient juste derrière et on gardera le résultat et ainsi de suite avec l'ensemble des bits.

Exemple :

14 en gray s'écrit 1001

1001

1110 *somme sans retenue* $\oplus$

Le résultat de l'écriture de 14 en binaire pur est : $(1110)_2$

VI.2.2 Le code à excédant 3 (XS3)

Le code à excédant trois consiste à prendre chaque chiffre décimal, à lui rajouter 3, puis à coder le résultat obtenu en binaire pur.

La séquence des nombres décimales codés en excédant trois est la suivante :

0 est codé 0011	5 est codé 1000
1 est codé 0100	6 est codé 1001
2 est codé 0101	7 est codé 1010
3 est codé 0110	8 est codé 1011
4 est codé 0111	9 est codé 1100

VII. Les codes alphanumériques

Ce type de codes servent à coder les objets (chiffres, lettres, symboles, signes de ponctuations, caractères spéciaux …). Le code le plus utilisé est le code ASCII (pour American Standard Code for Information Interchange) ou le code télégraphique international N5. Il est composé de 7 bits et un bit de parité, on l'utilise en télécommunication et en transmission.

Exemple :

La lettre U est codé 1010101

La lettre F est codé 1000110

La lettre K est codé 1001011

Réciproquement le code correspondant à :

1000011 est la lettre C

0110011 est le chiffre 3

1111011 est le caractère {

NB :

- Si le nombre de bits à 1 présent dans le code est pair, on met le bit de parité à 0.
- Si le nombre de bits à 1 présent dans le code est impair, on met le bit de parité à 1.

Chapitre 2 : Représentations des systèmes combinatoires

I. Introduction

Dans le domaine de l'électronique numérique, on rencontre toujours deux notions de base : la variable binaire et la fonction binaire. Par définition :

La variable binaire ; c'est une variable qui prend deux valeurs : 0 ou 1 (vrai ou faux) et on parle de la logique toute ou rien.

La fonction binaire ; c'est une fonction de variables binaires, elle ne peut prendre que deux valeurs 0 ou 1 (vrai ou faux).

II. Les Fonctions logiques

II.1 Fonction d'une seule variable

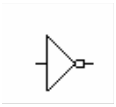
L'opérateur d'inversion ne porte que sur une seule variable d'entrée. Si x est la variable d'entrée, S la variable de sortie vaut : S= $\bar{x}$

On parle de la fonction « d'inversion », appelée porte logique NON

$x \rightarrow \bar{x}$

$0 \rightarrow 1$

$1 \rightarrow 0$

Son symbole est : 

II.2 Fonction de deux variables

A. La fonction OU /OR

L'opérateur OR (OU) porte sur deux variables d'entrée. Si x et y sont les variables d'entrée, alors S = x+y. S est vraie si x **OU y** sont vraies. L'opérateur OR est symbolisé par le plus (+) comme l'addition en mathématique. Sa table de vérité est la suivante :

x	y	x+ y
0	0	0
0	1	1
1	0	1
1	1	1

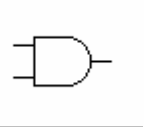
Son symbole est :

B. La fonction ET /AND

L'opérateur AND (ET) porte sur deux variables d'entrée. Si x et y sont les variables d'entrée, alors S = x. y est vraie si x **ET** y sont vraies. L'opérateur AND est symbolisé par le point (**.**) comme la multiplication en mathématique. Sa table de vérité est la suivante :

x	y	x.y
0	0	0
0	1	0
1	0	0
1	1	1

Son symbole est :

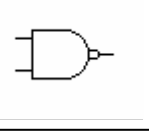


C. La fonction Non ET /NAND

L'opérateur NAND est l'inverse de l'opérateur AND. Son symbole est le symbole du ET suivi d'une bulle qui matérialise l'inversion. Sa table de vérité est la suivante :

x	y	$\overline{x.y}$
0	0	1
0	1	1
1	0	1
1	1	0

Son symbole est :

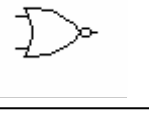


D. La fonction Non OU /NOR

L'opérateur NOR est l'inverse de l'opérateur OR. Son symbole est le symbole du OU suivi d'une bulle qui matérialise l'inversion. Sa table de vérité est la suivante :

x	y	$\overline{x + y}$
0	0	1
0	1	0
1	0	0
1	1	0

Son symbole est :

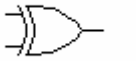


E. La fonction XOR / Ou Exculusif

La fonction XOR est vraie : si x ou y est vrai, mais pas simultanément, càd : x est **différent de** y. L'opérateur XOR est symbolisé par un + entouré d'un cercle ($\oplus$), il réalise l'addition en binaire sans retenu, sa table de vérité et la suivante :

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Son symbole est :



F. La fonction coïncidence :

La fonction **coïncidence** est vraie : si les deux variables se coïncident, c'est l'inverse de la fonction XOR. Son symbole est le symbole du XOR suivi d'une bulle qui indique l'inversion. Sa table de vérité et la suivante :

x	y	$x \otimes y$
0	0	1
0	1	0
1	0	0
1	1	1

Son symbole est :

Pour étudier ces systèmes et leur logique de fonctionnement, il est nécessaire au préalable de maîtriser l'algèbre spécifique qui en définit les règles : l'algèbre de BOOLE.

III. Algèbre de BOOLE

III.1 Théorèmes fondamentaux

a. Opérateur OU

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

b. Opérateur ET

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

III.2 Propriétés

- La commutativité :

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

- L'associativité :

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

$A . (B . C) = (A . B) . C = A . B . C$

-La distributivité :

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

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

III.3. Théorème d'absorption

$A + (A . B) = A$

$A . (A + B) = A$

III. 4. Théorème d'allongement

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

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

III.5. Théorème de De Morgan

C'est une des propriétés les plus importantes des fonctions logiques. Il est défini par les deux relations suivantes :

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

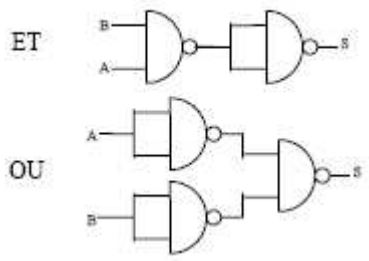
$$\overline{A . B} = \overline{A} + \overline{B}$$

Ainsi le complément d'une fonction logique sera obtenu en remplaçant les variables par leur complément, les signes + par des • et les signes • par des +.

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

- NAND et NOR sont des portes universelles, car elles permettent de réaliser toutes les opérations logiques élémentaires.

Exemple :



Chapitre 3 : Les formes canoniques des fonctions logiques et leurs simplifications

I. Introduction

Il existe deux méthodes pour exprimer une fonction logique : soit donner directement son équation logique, soit utiliser une table de vérité. Pour un nombre fini N de variables d'entrée correspond 2^N combinaisons possibles des variables d'entrée.

Par définition la table de vérité d'une fonction c'est un tableau qui représente l'état de sortie en fonction de toutes les combinaisons des variables d'entrée.

Exemple : soit une fonction $F_{(A, B, C)}$ dont la table de vérité est :

A B C	F	
0 0 0	0	
0 0 1	1	→ $\overline{A}\,\overline{B}\,C$
0 1 0	1	→ $\overline{A}\,B\,\overline{C}$
0 1 1	0	
1 0 0	1	→ $A\,\overline{B}\,\overline{C}$
1 0 1	0	
1 1 0	1	→ $A\,B\,\overline{C}$
1 1 1	0	

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

II. Les quatre formes canoniques d'une fonction logique

L'expression précédente de F possède la forme d'une somme de produits de variables, appelée somme de produits ou somme de monômes. Elle représente la première forme canonique.

$F = \sum \min termes$

On la note aussi : F= $\sum$ 1, 2, 4, 6

Alors $\overline{F} = \sum 0,3,5,7$

On complémente à $2^{n-1} = 2^{3-1} = 7$ (avec n nombre de variables, dans notre exemple n = 3)

On obtient : $F = \prod 7,4,2,0$

On déduit l'autre expression de la fonction F :

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

Cette forme est un produit de sommes de variables, appelé produit de somme ou produit de *maxterme*, elle représente la deuxième forme canonique.

F= $\prod \max terme$.

Récapitulation

Une fonction booléenne peut être exprimée sous quatre formes canoniques :

- la première forme canonique : *sommes de produits* : Σ(Π)

- la deuxième forme canonique : *produits de sommes* : Π(Σ)

- la troisième forme canonique : *l'écriture avec l'opérateur Nand uniquement*

- la quatrième forme canonique *: l'écriture avec l'opérateur NOR uniquement*

III. Simplification des fonctions booléennes

Dans le but de réaliser un circuit logique :

- à faible coût
- dans les plus brefs délais
- avec moins d'erreur
- avec peu de composants

L'opérateur doit simplifier au maximum ces fonctions algébriques, plusieurs techniques on été développées. On cite deux d'entre elles :

III.1. Simplification Algébrique

Cette méthode est basée sur les propriétés et théorème de l'algèbre de BOOLE (Chapitre 2). La méthode algébrique est toujours possible, mais elle repose sur une démarche intuitive qui dépend de l'habileté et de l'expérience (la méthodologie de la simplification algébrique n'est pas unique).

III.2. Simplification par la méthode de Karnaugh

La méthode de Karnaugh est l'une des différentes méthodes permettant d'exprimer une fonction booléenne sous sa forme réduite. La méthode repose sur la recherche des simplifications possibles entre les différents termes d'une expression. Elle est basée sur les termes ou les cases adjacentes.

a. Tables de Karnaugh

Le tableau de Karnaugh est une variante de la table de vérité. Il s'agit d'un tableau à double entrée dans lequel chaque combinaison des variables d'entrée est associée à une case qui contient la valeur de la fonction. Cependant, la disposition des cases est telle que deux cases contiguës correspondent à des combinaisons adjacentes des variables d'entrée, c'est à dire des combinaisons ne différant que par la complémentation d'une seule variable.

En effet, deux termes adjacents (qui ne différent que par une variable) peuvent se simplifier facilement, exemple : $A.B.C + A.\overline{B}.C = AC$.

On représente les valeurs de la fonction dans un tableau, dans lequel chaque ligne et chaque colonne correspondent à une combinaison des variables d'entrée exprimée avec le code GRAY.

Exemple 1 : deux variables d'entrée A et B, n = 2, nous avons $2^n = 4$ cases :

B \ A	0	1
0		
1		

Exemple 2 : trois variables d'entrée A, B et C, n = 3, nous avons $2^n = 8$ cases :

C \ BA	0 0	0 1	1 1	1 0
0				
1				

Exemple 3 : quatre variables d'entrée A, B, C et D, n = 4, nous avons $2^n = 16$ cases :

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

b. Règles de simplification

Etape 1 : Ecrire la fonction F sous sa première forme canonique

Etape 2 : Mettre des 1 dans les cases correspondantes aux mintermes présents.

Etape 3 : Réunir en plus grands boucles (groupements) possibles les **cases adjacentes** contenant la valeur **1.** Ces boucles seront réalisés par association en **puissances de 2** des cases adjacentes (on commence par les plus grandes boucles possibles : groupes de 8 cases, puis de 4 cases et de 2 cases sinon une seule case isolée).

NB : il faut chercher à minimiser le nombre de boucles et chercher à réaliser les plus grands groupements possibles car plus le groupement est important, moins le terme correspondant contient de variables.

Etape 4 : Eliminer la variable où les variables qui apparaissent avec leur complément dans une boucle.

Etape 5 : Additionner logiquement le résultat de chaque boucle. Finalement on obtient l'expression simplifiée sous la forme d'une somme de produit.

NB :

- Dans le tableau de Karnaugh, les cases sur le bord supérieur sont adjacentes avec ceux du bord inférieur de même pour le bord gauche avec le bord droit.

- Une case isolée contenant un 1 peut être commune à plusieurs groupements, càd : on peut utiliser une case plus d'une fois car : X=X+X+…+X

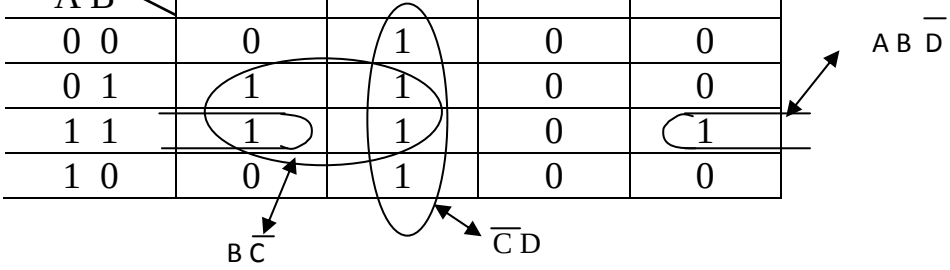
- Toutes les cases contenant 1 doivent être entourées.

- Dans un tableau de Karnaugh à 16 cases ou à 4 variables :

- ✓ une case isolée correspond à un terme de 4 variables,
- ✓ une boucle de 2 cases correspond à un terme de 3 variables,
- ✓ une boucle de 4 cases correspond à un terme de 2 variables,
- ✓ une boucle de 8 cases correspond à un terme d'une seule variable.

Exemple : F(A, B, C, D) = Σ 1, 4, 5, 9, 12, 13, 14

A B \ CD	0 0	0 1	1 1	1 0
0 0	0	1	0	0
0 1	1	1	0	0
1 1	1	1	0	1
1 0	0	1	0	0



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

IV. Cas des fonctions incomplètement définies

Une fonction logique est dite incomplètement définie si sa valeur est indifférente ou non spécifiée ou elle correspond à une combinaison impossible des variables d'entrée.

La question : peut-on simplifier une telle fonction par la méthode de karnaugh ?

Dans la table de karnaugh les cas indéfinis sont désignés par un des symboles suivant : X ou ϕ

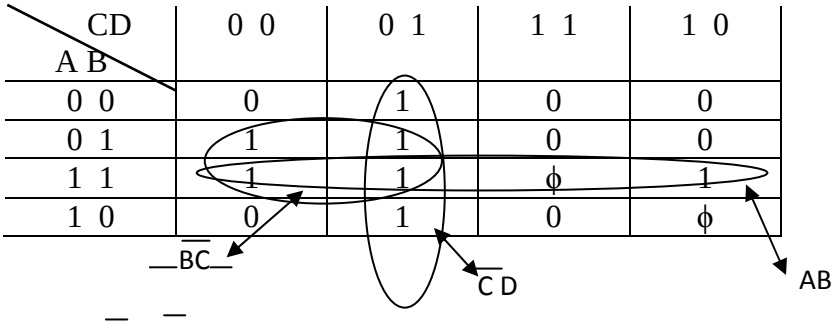
La méthode de simplification des fonctions incomplètement définies est réalisée selon la règle définie précédemment, reste le cas des cases qui ont pour valeur ϕ :

- ϕ prend la valeur 1 si un groupement plus important de cases adjacentes pourra se faire,

- ϕ prend la valeur 0 s'il n'a aucun rôle dans la simplification.

Exemple : F (A, B, C, D) = **Σ** 1, 4, 5, 9, 12, 13, 14 et ϕ pour : 10, 15

A B \ CD	0 0	0 1	1 1	1 0
0 0	0	1	0	0
0 1	1	1	0	0
1 1	1	1	ϕ	1
1 0	0	1	0	ϕ



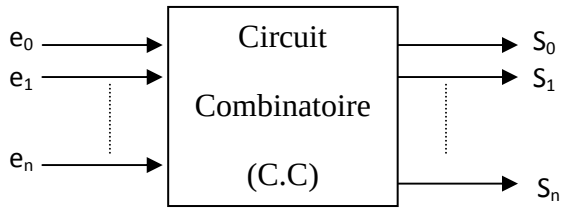
$F = AB + B\overline{C} + \overline{C}D$

Chapitre 4 : Les circuits combinatoires

I. Introduction

Les circuits logiques combinatoires sont des circuits constitués de portes logiques, dont les sorties dépendent uniquement de la combinaison des variables entrées. En d'autres termes :

- La sortie ne dépend que des entrées : $S_i = f(E_i)$,
- A chaque combinaison des variables d'entrée, nous auront une sortie fixe,
- Ce type de circuit ne possède pas de mémoire (la notion du temps n'intervienne pas),
- Ce type de circuit peut réaliser une ou plusieurs fonctions logiques à un instant donné.



II. Les circuits d'Arithmétiques

II.1. Additionneur

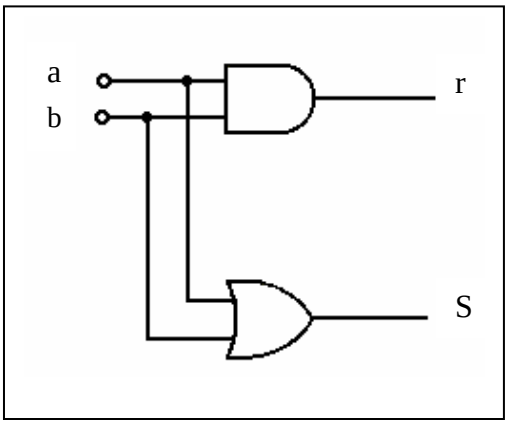
C'est un circuit combinatoire qui peut réaliser l'addition de deux nombres à n bits. Cette opération génère deux sorties : la somme S et la retenue R.

A. Demi-additionneur :

a	b	s	r
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**

On obtient :

$$\begin{cases} s = a \oplus b \\ r = a.b \end{cases}$$



B. Additionneur Complet (ADC) à 1 bit

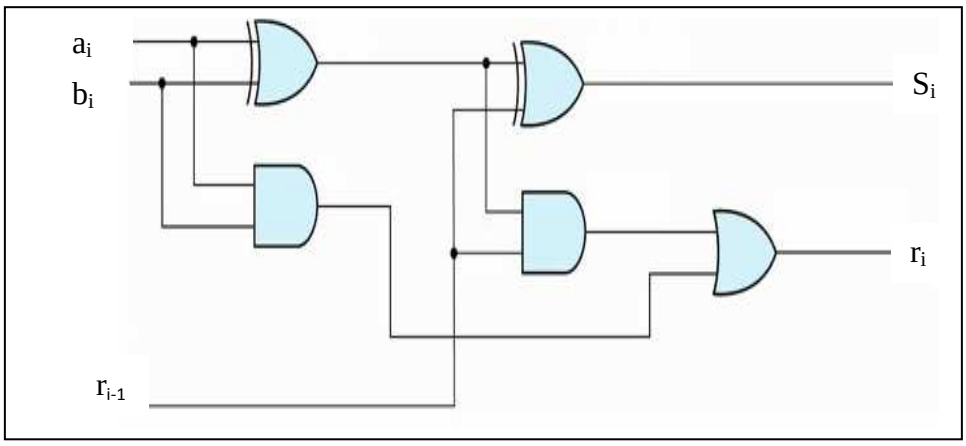
Cette opération dépend de a_i, b_i et du résultat de la dernière somme (r_{i-1}). L'entrée r_{i-1} est la retenue entrante et la sortie r_i est la retenue sortante.

L'opération est indiquée dans le tableau ci-dessous, selon l'équation : $s = a_i + b_i + r_{i-1}$

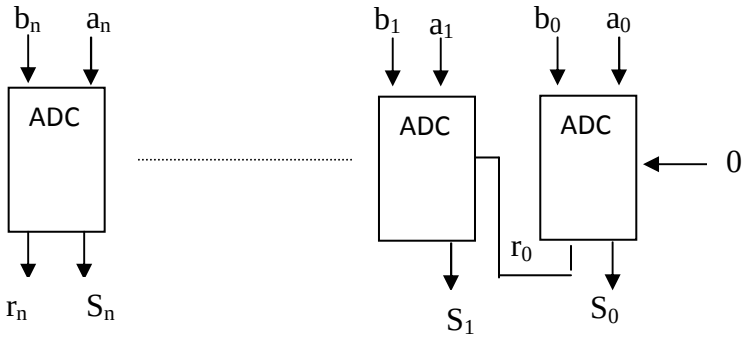
a_i b_i r_{i-1}	S_i r_i
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1

Apres simplification on obtient l'expression des deux sorties :

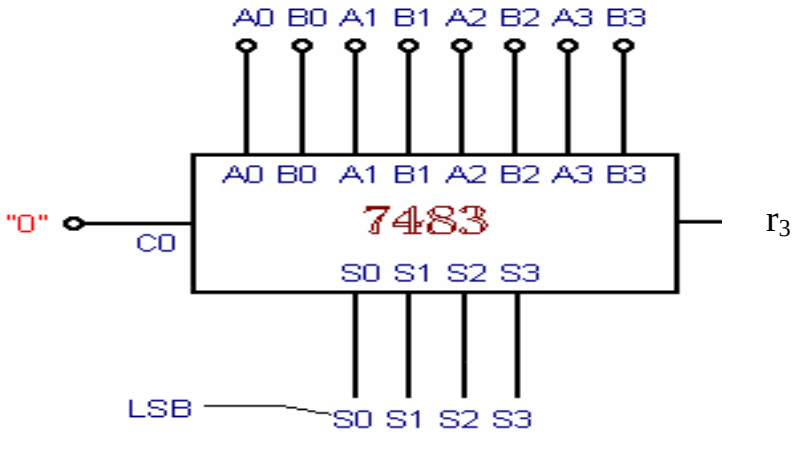
$$\begin{cases} S_i = a_i \oplus b_i \oplus r_{i-1} \\ r_i = r_{i-1}(a_i \oplus b_i) + a_i b_i \end{cases}$$



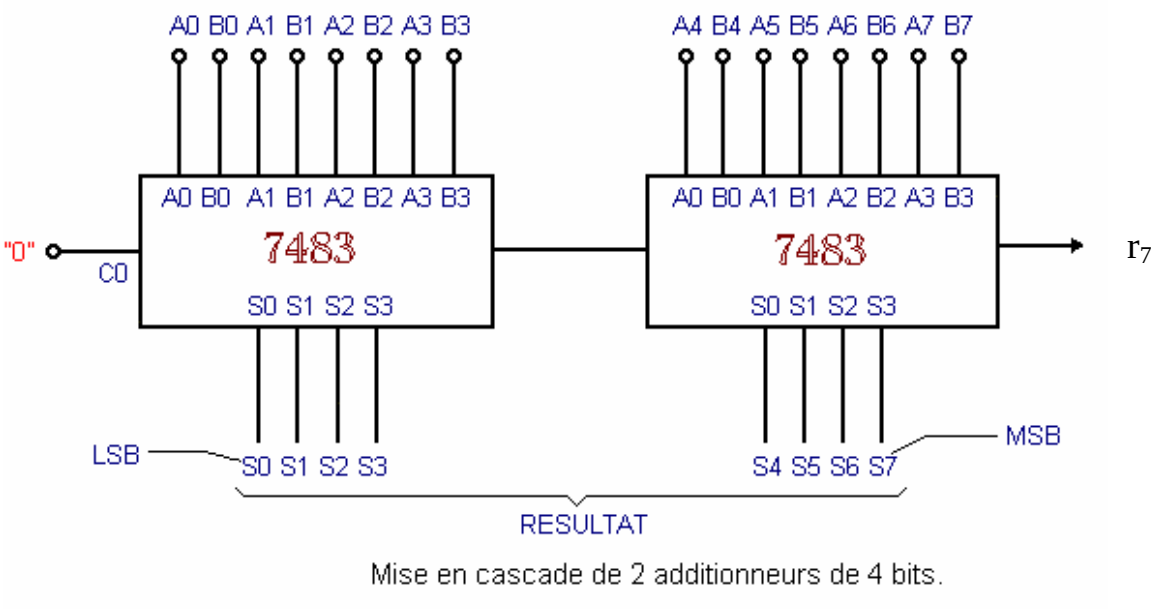
On généralise à n bits :



En pratique, on rencontre un Circuit Intégré CI 7483 qui réalise l'opération d'addition de deux nombres binaires à 4 bits (voit figure ci-dessous)



On généralise à 8 bits (et ainsi de suite) on aura :



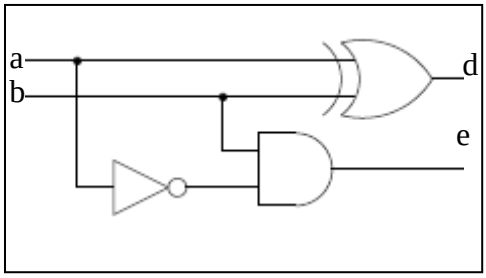
Mise en cascade de 2 additionneurs de 4 bits.

II.2. Soustracteur

Le soustracteur est un circuit combinatoire qui peut faire la soustraction de deux nombres à n bits. Cette opération génère deux sorties : la différence **d** et l'emprunte **e**.

A. Demi-soustracteur :

a	b	d	e
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



On obtient : $d = a \oplus b$

$e = \overline{a}.b$

B. Soustracteur Complet à 1 bit

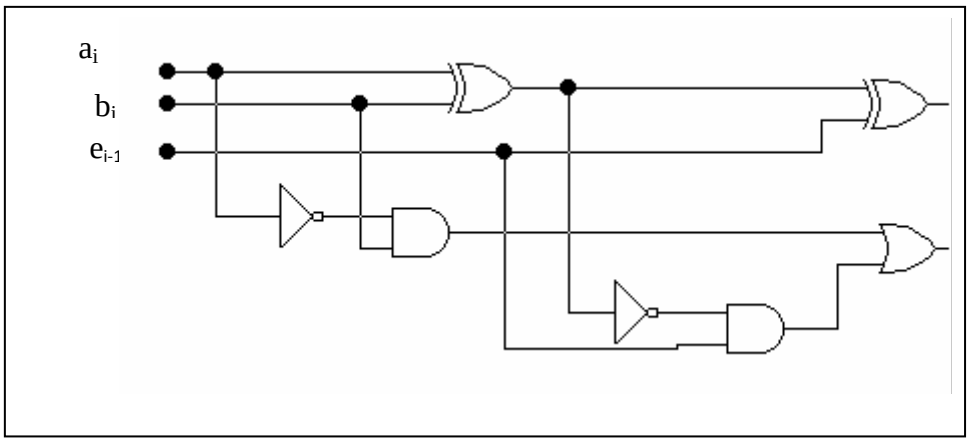
Cette opération dépend de a_i, b_i et du résultat de la dernière opération. L'entrée e_{i-1} est l'emprunte entrante et la sortie e_i est l'emprunte générée.

L'opération est indiquée dans le tableau ci-dessous selon l'équation : $d= a_i - b_i - e_{i-1}$:

a_i b_i e_{i-1}	d_i e_i
0 0 0	0 0
0 0 1	1 1
0 1 0	1 1
0 1 1	0 1
1 0 0	1 0
1 0 1	0 0
1 1 0	0 0
1 1 1	1 1

Apres simplification, on obtient l'expression des deux sorties :

$$\begin{cases} D_i = a_i \oplus b_i \oplus e_{i-1} \\ e_i = e_{i-1}.\overline{(a_i \oplus b_i)} + \overline{a_i} b_i \end{cases}$$



III. Les Transcodeurs

Le transcodeur est un circuit combinatoire qui permet le passage de l'écriture d'un nombre dans un code donné vers un autre code. Plusieurs circuits existent en pratique, dans ce qui suit on prend un exemple d'un transcodeur Gray-→ binaire à 3 bits.

g_2	g_1	g_0	b_2	b_1	b_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	1	0	1	0
0	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1
1	0	1	1	1	0
1	0	0	1	1	1

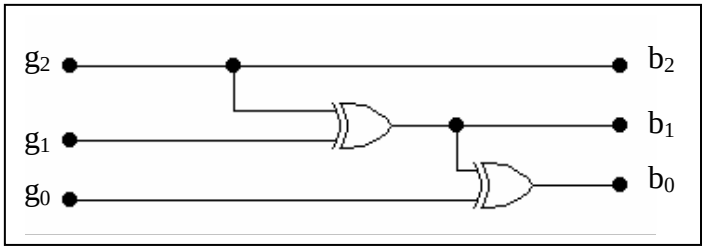
On utilise la méthode de karnaugh pour simplifier les équations des b_i, on obtient :

$$\begin{cases} b_2 = g_2 \\ b_1 = g_2 \oplus g_1 \\ b_0 = g_2 \oplus g_1 \oplus g_0 \end{cases}$$

La formule générale d'un transcodeur Gray-→ binaire à n bits est :

$$\begin{cases} b_n = g_n \\ b_i = g_n \oplus g_{n-1} \oplus .... \oplus g_i \end{cases}$$

Le logigramme de ce transcodeur est illustré sur la figure suivante :



IV. Les Décodeurs

Un décodeur est un circuit combinatoire qui délivre une information lorsque la combinaison des variables binaires est représentative du mot code choisi. Comme la combinaison des variables binaires à l'entrée du décodeur est unique, il ne peut exister qu'une seule sortie dans un état actif (1), les autres seront à zéro (0).

On distingue plusieurs décodeur 1parmi n ou (1/n), on cite : décodeur 1/4, 1/8 et 1/16.

Dans ce qui suit on montre le fonctionnement du Décodeur ¼ ;

C'est un circuit qui possède : 4 sorties : S3, S2, S1, S0.

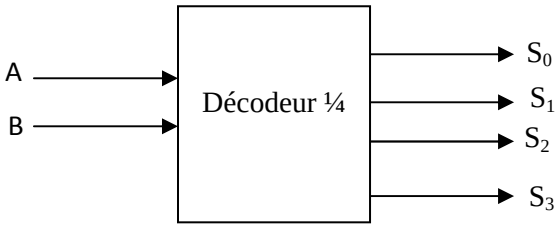
2 entrées : A, B

Sa table de vérité est la suivante :

A	B		S_3	S_2	S_1	S_0
0	0		0	0	0	1
0	1		0	0	1	0
1	0		0	1	0	0
1	1		1	0	0	0

Les équations du décodeur ¼ sont :

$$\begin{cases} S_0 = \overline{A}\ \overline{B} \\ S_1 = \overline{A}\ B \\ S_2 = A\ \overline{B} \\ S_3 = A\ B \end{cases}$$



Application des décodeurs : - Adressage des cases mémoires

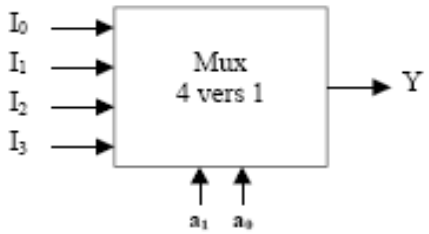
- Génération des fonctions logiques

V. Circuit d'aiguillage

V.1 Le Multiplexeur (MX)

C'est un sélecteur de données ou aiguillage convergent (dit transformation parallèle / série). Il peut transformer une information apparaissant sous forme de n bits en parallèle en une information se présentant sous forme de n bits en série. La voie d'entrée, sélectionnée par son adresse, est reliée à la sortie.

Sa représentation est illustrée sur le schéma ci-dessous :



Les entrées a_1 et a_0 sont appelées les entrées d'adresses, les entrées I_3 , ..., I_0 sont appelées entrées d'informations.

Dans tout type de (Mx), la relation entre ces deux type d'entrées est la suivante :

$$\textit{Nombre d'entrées d'informations} = 2^{\textit{nombre entrées d'adresses}}$$

Pour ce type de (MX : 4 vers 1) nous avons :

Si : a_1=0 , a_0= 0, la sortie Y = I_0

Si : a_1=0 , a_0= 1, la sortie Y = I_1

Si : a_1=1 , a_0= 0, la sortie Y = I_2

Si : a_1=1 , a_0= 1, la sortie Y = I_3

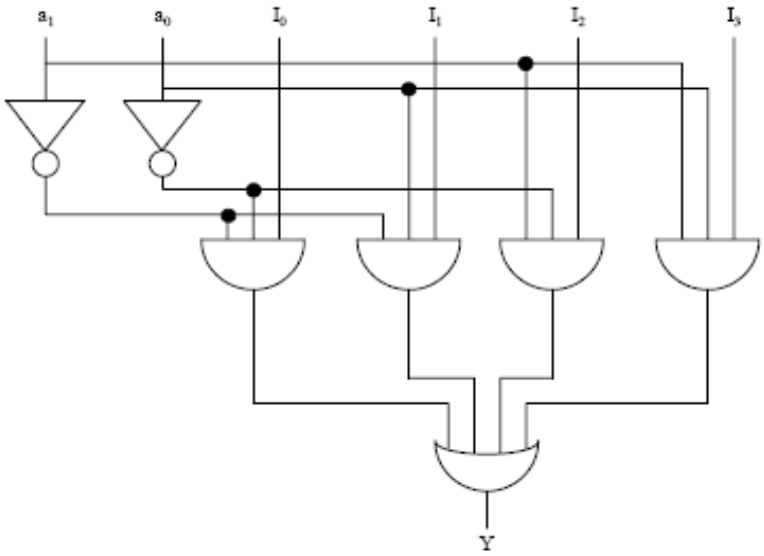
A partir de là, on tire l'équation de sortie de ce (MX) :

$Y= \overline{a_1}\,\overline{a_0}\,I_0 + \overline{a_1}\,a_0 I_1 + a_1\,\overline{a_0}\,I_2 + a_1\,a_0 I_3$

Application pratique des (MX)

Exemple d'utilisation de ce type de circuit : c'est la génération de fonction logique, si nous avons une fonction à n variables on utilise un (MX) à 2^n entrées d'informations. Les entrées de sélection sont alors les variables de la fonction.

Le logigramme du (MX) est le suivant :

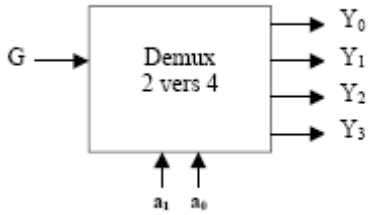


V.2 Le Démultiplexeur (DMX)

C'est un circuit qui aiguille une voie en entrée vers plusieurs voies en sortie, c'est une transformation série/ parallèle.

La différence entre le (MX) et le (DMX) réside dans le sens de la circulation de l'information.

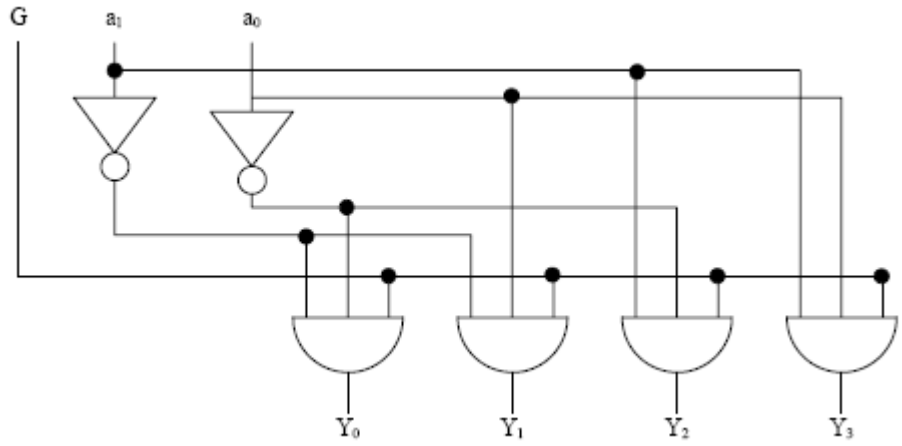
Le schéma fondamental d'un démultiplexeur à 2 entrées d'adresses est illustré ci-dessous :



Les équations de sortie sont :

$$Y_0 = \overline{a_1}\,\overline{a_0}\,G$$
$$Y_1 = \overline{a_1}\,a_0\,G$$
$$Y_2 = a_1\,\overline{a_0}\,G$$
$$Y_3 = a_1\,a_0\,G$$

Le logigramme de ce DMX est le suivant :



Application des DMX : C'est dans le domaine des transmissions, à l'émission on utilise un MX et à la réception on récupère l'information à l'aide d'un DMX.

VI. Les Comparateurs

C'est un circuit combinatoire qui compare deux nombres binaires A et B puis détecte si A est supérieur ou bien inférieur à B, de même il détecte l'égalité.

Table de vérité d'un comparateur à 1 bit :

a	b	S	E	I
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$S = a\,\bar{b}$

$E = a \otimes b$

$I = \bar{a}\,b$

on généralise pour un comparateur à 2 bits A (a_1, a_0) et B (b_1,b_0)

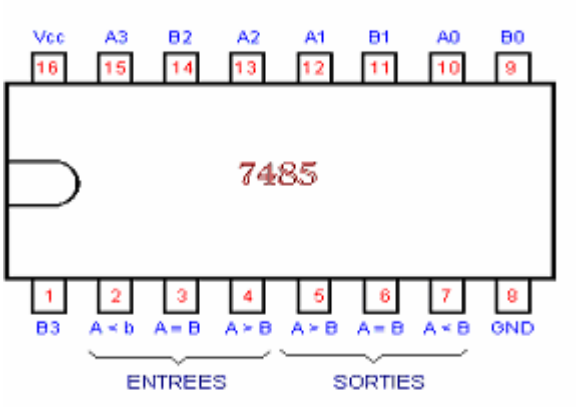
$S = a_1\overline{b_1} + (a_1 \otimes b_1)\,a_0\overline{b_0}$

$E = (a_1 \otimes b_1)(a_0 \otimes b_0)$

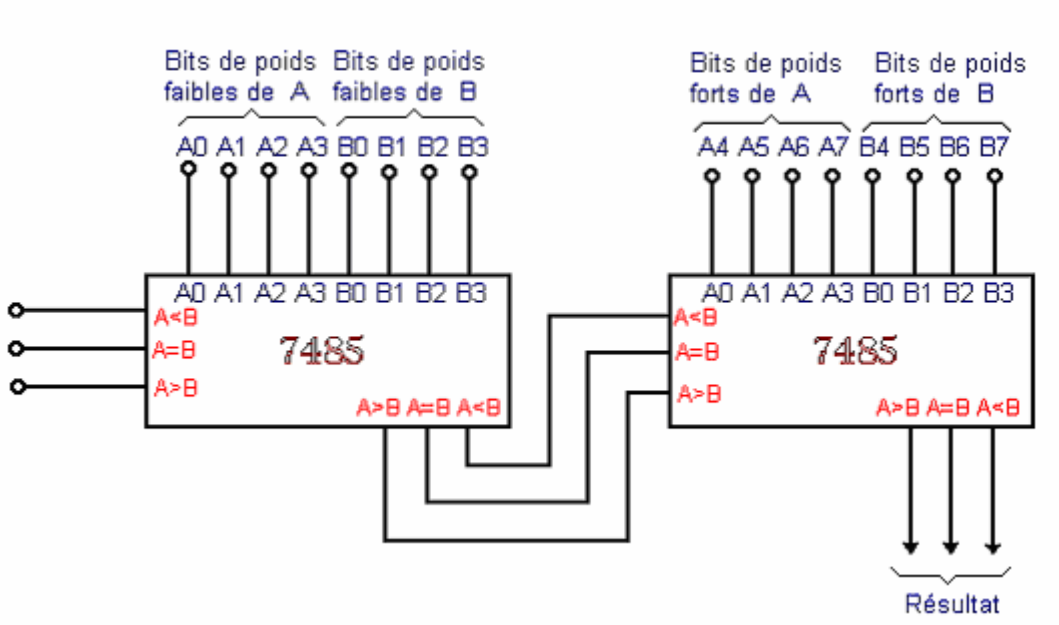
$I = \overline{a_1}\,b_1 + (a_1 \otimes b_1)\,\overline{a_0}\,b_0$

Comparateur Intégré à 4 bits CI 7485

Le circuit intégré 7485 est un comparateur de deux nombres de 4 bits. De plus, il dispose de 3 entrées notées A = B, A > B et A < B qui autorisent la mise en cascade de plusieurs circuits comparateurs du même type. Ainsi, on peut comparer des nombres de 8, 12, 16 bits....



Pour comparer deux nombres de 8 bits, il suffit de mettre en série deux comparateurs 7485 et de relier la sortie A = B du premier comparateur à la même entrée du second et de faire la même chose avec les sorties A > B et A < B. comme sont indiquées à la figure suivante :



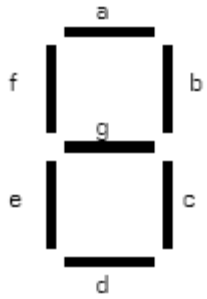
VII. Afficheur à 7 segments ou décodeur à 7 segments

On appel décodeur 7 segments, le dispositif de transcodage permettant de passer du code binaire ou DCB au code d'affichage du chiffre : le décodage est visuel.

Soit : a, b, c, d, e, f et g les variables correspondants aux 7 segments, si une variable est au niveau actif (1), le segment qui lui correspond sera allumé, inversement si une sortie est à l'état logique 0, le segment correspondant est éteint.

La synthèse d'un décodeur à 7 segment s'effectue comme pour un décodeur classique : il y'aura en effet 7 fonctions à déduire et à simplifier en fonction des variables d'entrée.

Les segments sont répartis comme c'est indiqué dans la figure ci-dessous :



L'affichage des symboles sur l'afficheur 7 segments est comme suit :

Table de vérité de l'afficheur à 7 segments :

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	0	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

Application des afficheurs à 7 segments

Les afficheurs 7 segments sont très présent sur les calculatrices et les montres à affichage numérique, les appareils de mesure et en électroménager et dans la plus part des installations industriels, …

Chapitre 5 : Les Bascules

I. Introduction

Dans certaines réalisations, les sorties des systèmes doivent tenir compte d'états antérieurs, ces systèmes intègrent une variable supplémentaire qui est : le temps, ils sont appelés ***systèmes séquentiels***. Dans leur réalisation nous devrons inclure des circuits capables de conserver en mémoire les informations relatives à l'évolution du système.

A l'inverse des circuits combinatoires, les sorties d'un circuit séquentiel sont fonction des valeurs des entrées à l'instant présent et aussi des sorties précédentes c.à.d. du temps (S_i= F(e_i, t)).

Pour réaliser ces systèmes, nous utiliserons des éléments qui mémorisent les états passés, ces opérateurs sont appelés *les bascules*.

Dans la pratique on rencontre plusieurs types de Bascules, les plus utilisées sont : la bascule RS, JK, D et T. Chaque bascule possède sa propre table de vérité et sa propre équation caractéristique.

II. Les différents types de bascules

1. La bascule RS

C'est une bascule qui possède deux sorties (Q *et* $\overline{Q}$) et deux entrées R (Reset : mise à zéro) et S (Set : mise à 1) :

Table de vérité : (avec Q^+ :C'est l'état futur, et Q : état passé)

Si, S=0, R= 0, La bascule garde sont état passé (Q^+= Q)

Si, S=0, R= 1 La bascule passe à zéro (Q^+= 0)

Si, S=1, R= 0 La bascule passe à un (Q^+= 1)

Si, S=1, R= 1 Combinaison interdite ((Q^+= Φ)

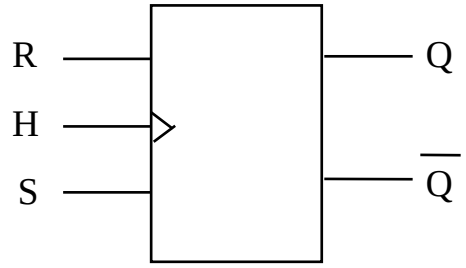
La table de vérité réduite est illustrée ci-contre :

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Interdit

Puis on tire la table de vérité :

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	Φ
1	1	1	Φ

Schéma symbolique



NB : H c'est l'entrée du signal d'horloge dans le cas des bascules synchrones

.

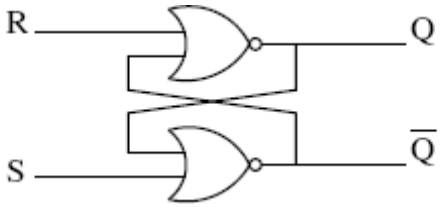
Equation caractéristique

D'après la table de vérité de la bascule RS, nous avons :

$$Q^+ = \overline{S}\,\overline{R}\;Q + S\,\overline{R}\,\overline{Q} + S\,\overline{R}\,Q = \overline{S}\,\overline{R}\;Q + S\,\overline{R}\;= \overline{R}(\,S + \overline{S}\,Q) = \overline{R}(\,S + Q)$$

$$Q^+ = \overline{R}(\,S + Q) = \overline{\overline{R}(\,S + Q)} = \overline{R + (\,\overline{S + Q})}$$

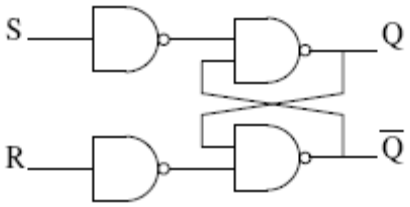
Le logigramme de la bascule RS avec des portes NOR est le suivant :



De la même manière, on exprime l'équation de la bascule RS avec des Nand uniquement et on obtient :

$$Q^+ = \overline{\overline{S}\,(\,\overline{\overline{R}\,Q})}$$

Le logigramme de la bascule RS avec des portes Nand est le suivant :



2. La bascule D (Data)

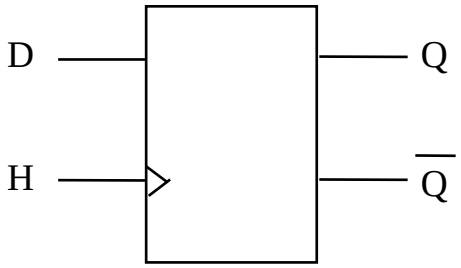
C'est une bascule qui possède deux sorties (Q *et* $\overline{Q}$) et une seule entrée D (Data). Elle est obtenue à partir de la bascule RS en posant : S=D et R = $\overline{D}$ (c-à-d, on rajoute un inverseur entre les entrées S et R de la bascule RS), ça table de vérité est la suivante :

Table de vérité :

D	Q^+
0	0
1	1

On dit que cette bascule recopie en sortie, la valeur présentée sur l'entrée D (elle joue le rôle d'une mémoire).

Schéma symbolique



Equation caractéristique

$$Q^+ = D$$

3. La bascule JK

C'est une bascule qui ressemble à la bascule RS. Elle possède deux sorties (Q *et* $\overline{Q}$) et deux entrées J et K. Elle diffère de la bascule RS dans la dernière combinaison lorsque (J=1 et K=1) la sortie : $Q^+ = \overline{Q}$

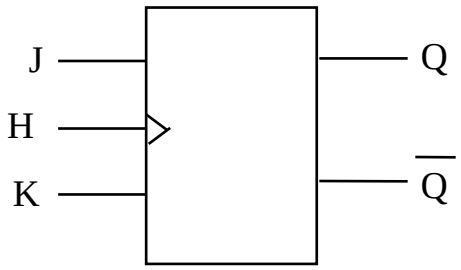
Table de vérité réduite :

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Ou :

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

Schéma symbolique



Equation caractéristique

$$Q^+ = J\,\overline{Q} + \overline{K}\,Q$$

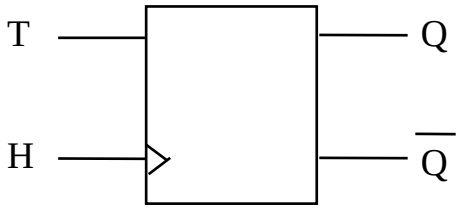
4. La bascule T

C'est une bascule qui ressemble à la bascule D, elle possède une seule entrée (T). elle s'appel bascule T pour signifié Trigger. La bascule T s'obtient à partir d'une bascule JK dont on a relié les entrées J et K entre elles (J=K=T).

Table de vérité

T	Q^+
0	Q
1	$\overline{Q}$

Schéma symbolique



Equation caractéristique

$$Q^+ = \overline{T}\,Q + T\,\overline{Q} = T \oplus Q$$

III. Commande des bascules :

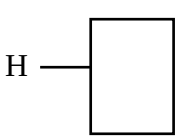
On distingue deux types de Bascules :

a. Les bascules qui fonctionnent sans signal d'horloge dites : bascules asynchrones.
b. Les bascules qui fonctionnent avec signal d'horloge dites : bascules synchrones.

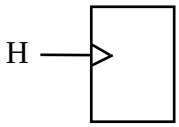
Dans cette catégorie, on différencie trois types de bascules synchrones :

i. bascule synchrone à niveau d'Horloge : la bascule prend en considération ses entrées lorsque H = 1, elle inhibe ses entrées si H = 0, (bascules Latchs).
ii. bascule synchrone à front Montant d'Horloge : la bascule prend en considération ses entrées lorsqu'elle reçoit un front montant du signal d'horloge ⌐↑ (passage de : $0 \rightarrow 1$)
iii. bascule synchrone à front Descendant d'Horloge : la bascule prend en considération ses entrées lorsqu'elle reçoit un front descendant du signal d'horloge ↓¬ (passage de : $1 \rightarrow 0$)

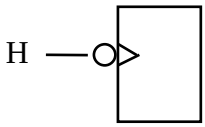
La représentation conventionnelle du mode d'activation de ces signaux :



H actif sur niveau d'Horloge



H actif sur front montant ⌐↑



H actif sur front descendant ↓¬

IV. Les Entrées de forçage

En plus des entrées synchrones (RS, JK, D, …), chaque bascule possède 2 entrées de forçage (R et S). Les entrées de forçage ont pour but de prépositionner la sortie d'une bascule à une valeur désirée (1 ou 0), soit lors de l'initialisation, soit en cours de fonctionnement.

Leur action est indépendante :

- des entrées synchrones,
- de la valeur de la sortie,
- du signal d'horloge.

Les entrées de forçage ont deux particularités essentielles :

i. Elles sont prioritaires devant toutes les autres entrées de la bascule.
ii. Leur effet est immédiat.

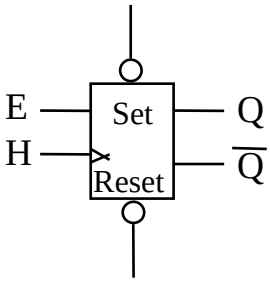
Chaque entrée de forçage est dédiée à une action :

- l'entrée Set provoque le forçage à 1 de la sortie de la bascule Q,
- l'entrée Reset provoque le forçage à 0 de la sortie de la bascule Q.

Dès qu'une entrée de forçage est activée, la sortie Q de la bascule prend instantanément la valeur correspondante.

Le mode d'activation est exprimé par la table de vérité suivante.

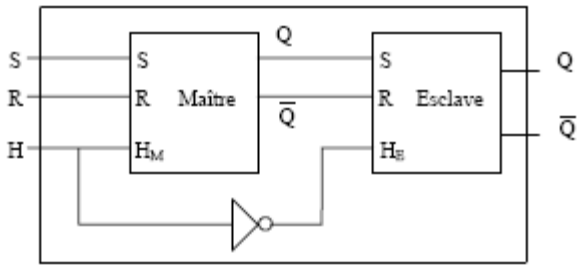
NB : Les entrées de forçage sont actives sur le niveau 0



E	H	Set	Reset	Q^+
X	X	0	0	INTERDIT
X	X	0	1	1
X	X	1	0	0
X	X	1	1	$Q^+ = f_{(Q, E)}$

V. La bascule Maître-Esclave ou Master/Slave (M/S)

La bascule (M/S) est constituée par la mise en cascade de deux bascules de même type, la bascule d'entrée est appelée le maître, la bascule de sortie, qui recopie l'état du maître, est appelée l'esclave. La structure générale d'une telle bascule est illustrée sur la figure ci-dessous :



Le maître : il fait l'enregistrement ou l'acquisition de l'information puis il ordonne ;

L'esclave : il exécute et il fait l'affichage de l'information

Le temps qui sépare l'affichage et l'enregistrement est égal à la demi-période du signal d'horloge, càd : si le *maître* fait l'acquisition sur front montant, l'*esclave* fait l'affichage sur front descendant $(H_m = \overline{H_e})$.

VI. Table de transition des bascules

Comme dans le cas de table de vérité, chaque bascule possède sa propre table de transition. Par définition, la table de transition d'une bascule définie les valeurs des entrées synchrones qui ont provoqués le basculement.

a. Table de transition d'une bascule RS

Q	Q^+	R	S
0	0	Φ	0
0	1	0	1
1	0	1	0
1	1	0	Φ

b. Table de transition d'une bascule JK

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

c. Table de transition d'une bascule D

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

d. Table de transition d'une bascule T

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

Chapitre 6 : Les Compteurs binaires

I. Introduction

Un compteur est un circuit séquentiel composé de plusieurs bascules connectés entre-elles, afin de réaliser une séquence chronologique dans un ordre croissant ou décroissant.

- Si toutes les bascules ont le même signal d'horloge, le compteur obtenu est dit s*ynchrone.*
- Si chaque bascule a son propre signal d'horloge, on parle de compteur *asynchrone.*

Les compteurs synchrones sont basés sur les tables de transition des bascules, par contre les compteurs asynchrones sont basés sur les diviseurs de fréquence.

- La sortie d'un compteur à n bascules évolue de 0 à 2^n-1, puis repasse à 0, soit un cycle de 2^n états différents.

II. Diviseur de fréquence par 2

C'est un circuit qui donne $Q^+ = \overline{Q}$ à chaque front actif d'horloge.

- Pour une bascule JK, il suffit de mettre j = k = 1,
- Pour une bascule D, il suffit de mettre D = $\overline{Q}$,
- Pour une bascule RS, il suffit de mettre S = $\overline{Q}$, R= Q,
- Pour une bascule T, il suffit de mettre T =1,

III. Structure d'un compteur asynchrone

La structure d'un compteur asynchrone est basée sur la mise en cascade de plusieurs diviseurs de fréquence par 2.

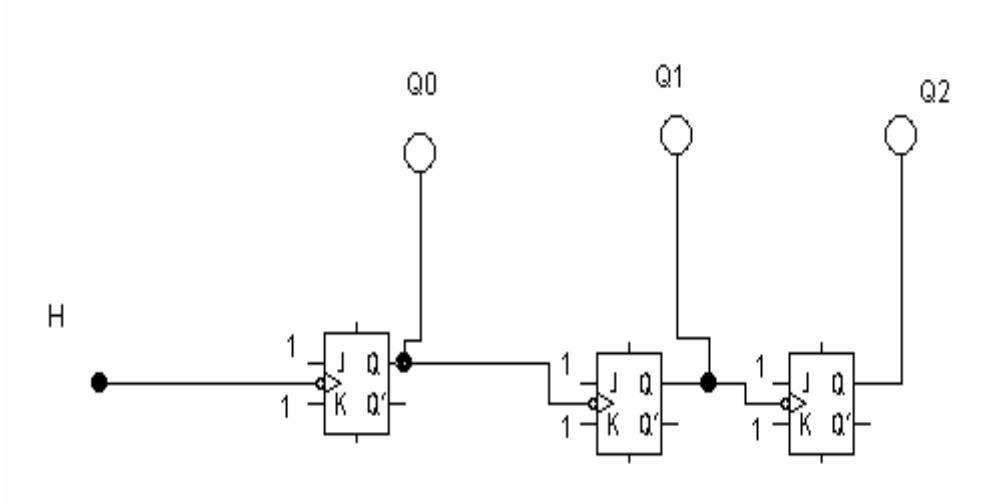
- Si le compteur parcoure tout ses états, on dit que c'est un compteur à *cycle complet.*
- Si le compteur saute quelques états, on dit que c'est un compteur à *cycle incomplet.*

Le nombre d'état = $2^{nombre\ de\ bascule}$, donc avec deux bascules nous obtenons les états suivants : 00, 01, 10, 11.

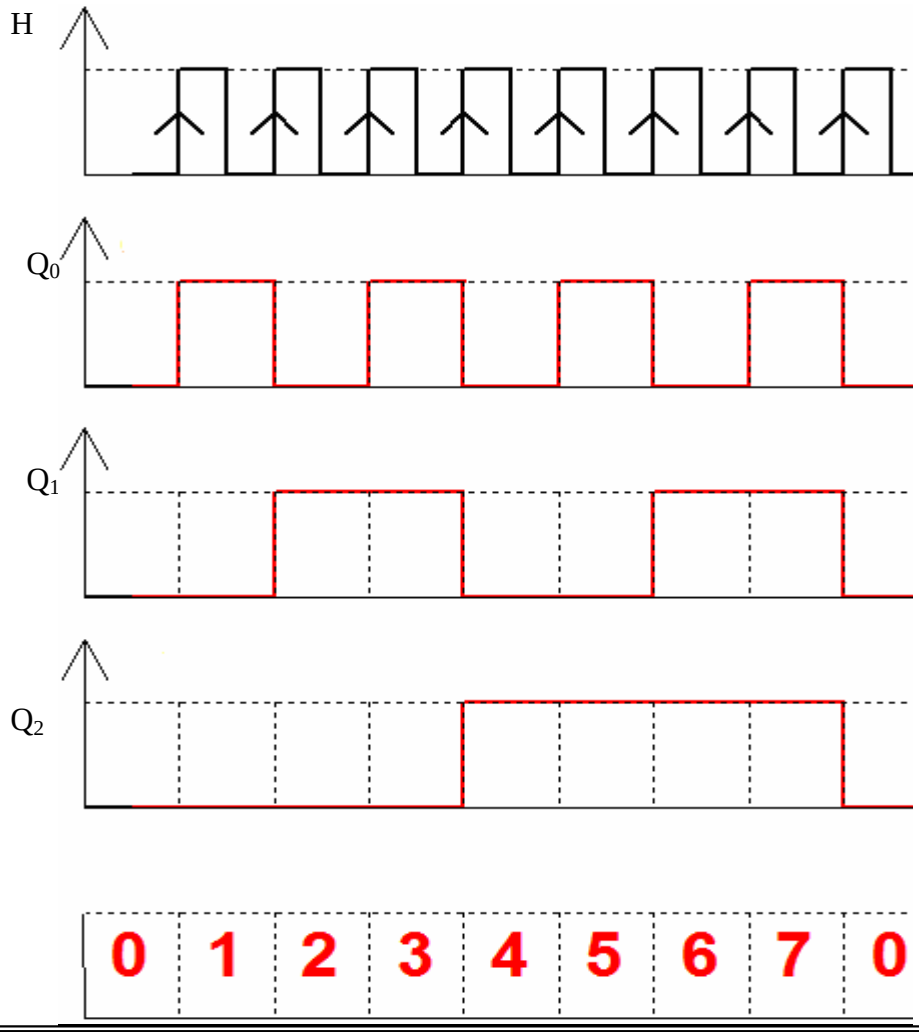
III.1 Compteur asynchrone à cycle complet

Le principe est que :

1. Chaque bascule va être câblée pour avoir un diviseur de fréquence par 2.
2. H_0 de la 1ère bascule est le signal d'horloge lui-même (H_0=H)
3. le signal d'horloge de la i^{ème} bascule $H_i = Q_{i-1}$ pour les bascules à fronts descendants et $H_i = \overline{Q_{i-1}}$ pour les bascules à fronts montants.



Le chronogramme de ce compteur est le suivant :



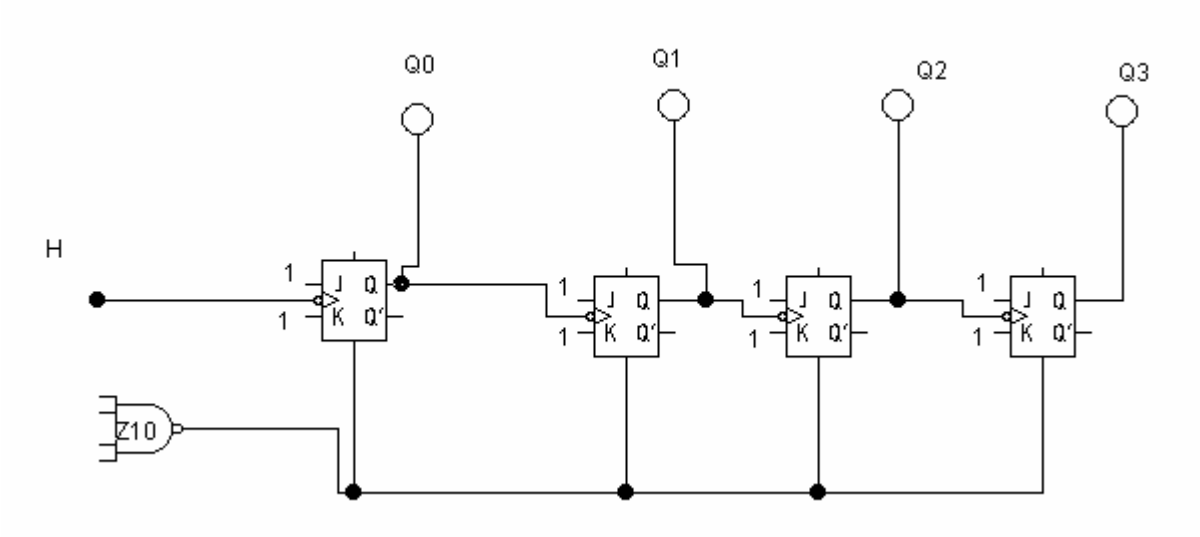
Remarque : si on affiche le résultat sur les sorties : $\overline{Q_{i-1}}$ des bascules JK, on obtient un décompteur, c.à.d : un circuit qui donne la séquence : 111, 110, 101, 100, 011, 010, 001, 000.

III.2 Compteur asynchrone à cycle Incomplet

Pour illustrer la synthèse d'un compteur à cycle incomplet on prend l'exemple suivant :

Exemple : Synthèse d'un compteur Modulo 10

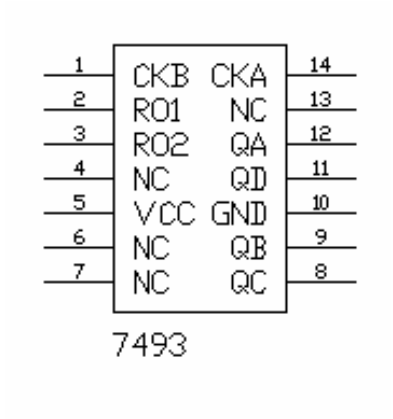
- Modulo 10 veut dire 10 états, et le compteur fait le cycle : $0 \rightarrow 9$.
- 9 s'écrit sur 4 bits, donc nous avons besoin de 4 bascules.
- Lorsque on détecte l'état 10, il faut forcer le compteur à zéro, on dit que l'état 10 est **fugitif** (noté : Z_{10}), donc on doit utiliser les entrées de forçage des bascules (voir Chapitre 5).
- L'équation de forçage est la suivante : 10 s'écrit 1010 donc $Z_{10} = Q_3 \overline{Q_2}\, Q_1 \overline{Q_0}$
- Lorsque le compteur détecte l'état 10, toutes les bascules vont être forcées à 0, pour cela on doit utiliser les entrées Reset (R) et comme elles fonctionnent avec des zéros, alors on doit inverser l'équation de forçage Z_{10}, on utilisant une porte Nand à 4 entrées. Le circuit final est le suivant :



IV. Compteur Asynchrone Intégré CI (7493)

A fin d'éviter certaines erreurs de câblage et de forçage, un CI 7493 a été conçu pour réaliser des séquences à cycle complet et même incomplet.

Il est illustré sur la figure suivante :



Avec : Q_A Q_B Q_C Q_D c'est les sorties du compteur.

CKA et CKB c'est les entrées d'horloge

R01 et R02 c'est les entrées de forçage dans le cas des compteurs à cycle incomplet.

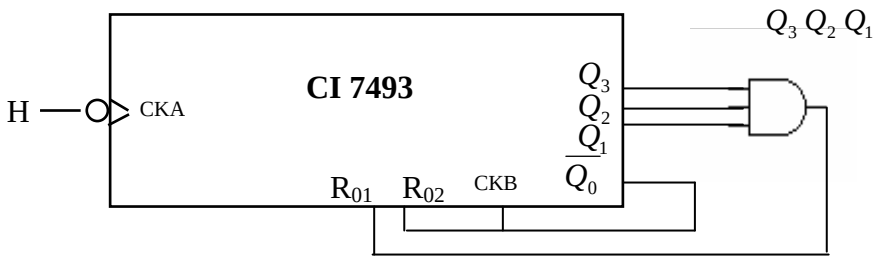
Vcc : Alimentation (+15V)

GND : Masse

NC : non connecté

Exemple d'utilisation :

Avec ce CI intégré, on réalise un compteur asynchrone à cycle incomplet modulo 14, sachant que la fonction de forçage est $Z_{14} = Q_3 Q_2 Q_1 \overline{Q_0}$ (car 14 s'écrit en binaire 1110), le branchement est illustré sur la figure suivante :



NB : La fréquence maximale du signal d'horloge H acceptée par le CI 7493 est autour de 16Mhz en technologie TTL.

V. Les limites du comptage asynchrone

Les compteurs asynchrones, faciles à constituer et ils sont très utilisés en pratique mais leur emploie trouve des limites :

- Le temps de propagation augmente avec le nombre d'étages (commande des bascules, les unes après les autres suite à la structure en cascade du compteur), ce qui entraîne une limite avec la fréquence d'utilisation
- Difficulté à passer du mode comptage au décomptage.
- Des aléas aléatoires lors d'existence de plus d'une fonction de forçage dans une séquence donnée.

La solution aux limitations signalées peut se trouver dans le recours aux compteurs synchrones.

VI. Structure d'un compteur Synchrone

La synthèse de ce type de compteur est basé sur la recherche des équations d'entrée de chaque bascule en fonction des sorties précédentes des autres bascules en utilisant les tables de transition.

Un compteur synchrone impose que toutes les bascules changent en même temps. Pour cela, il faut que toutes les bascules soient connectées avec une même entrée d'horloge.

VI.1 Méthode de synthèse d'un compteur synchrone à cycle complet

Exemple : Compteur synchrone modulo 8 avec les bascules JK.

Nous avons besoin de 3 bascules (2^3= 8 états) et on cherche les équations : $J_i, K_i = f(Q_j)$,

sachant que la table de transition d'une bascule JK (voir chapitre 5) est :

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

Donc la séquence est la suivante :

Q_2 Q_1 Q_0	Q_2^+ Q_1^+ Q_0^+		J_2 K_2	J_1 K_1	J_0 K_0
0 0 0	0 0 1		0 Φ	0 Φ	1 Φ
0 0 1	0 1 0		0 Φ	1 Φ	Φ 1
0 1 0	0 1 1		0 Φ	Φ 0	1 Φ
0 1 1	1 0 0		1 Φ	Φ 1	Φ 1
1 0 0	1 0 1		Φ 0	0 Φ	1 Φ
1 0 1	1 1 0		Φ 0	1 Φ	Φ 1
1 1 0	1 1 1		Φ 0	Φ 0	1 Φ
1 1 1	0 0 0		Φ 1	Φ 1	Φ 1

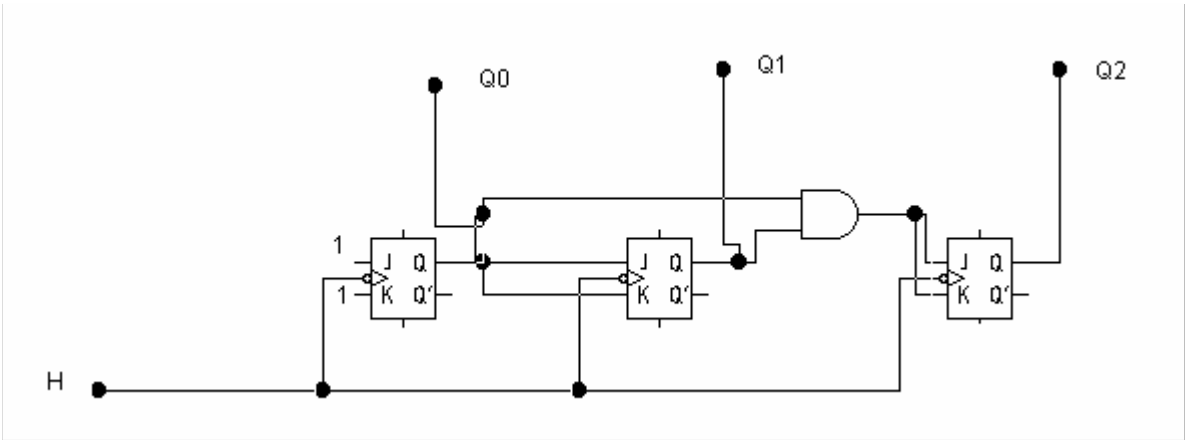
On remarque que : $J_0 = K_0 = 1$

En utilisant la méthode de karnaugh pour la simplification des équations de J_2 K_2 et J_1 K_1 on obtient :

$J_1 = K_1 = Q_0$

$J_2 = K_2 = Q_1 Q_0$

Le circuit final de ce compteur est le suivant :



VI.2 Méthode de synthèse d'un compteur synchrone à cycle Incomplet

Exemple : Compteur **synchrone modulo 6 avec des bascules JK.**

Q_2 Q_1 Q_0	Q_2^+ Q_1^+ Q_0^+		J_2 K_2	J_1 K_1	J_0 K_0
0 0 0	0 0 1		0 Φ	0 Φ	1 Φ
0 0 1	0 1 0		0 Φ	1 Φ	Φ 1
0 1 0	0 1 1		0 Φ	Φ 0	1 Φ
0 1 1	1 0 0		1 Φ	Φ 1	Φ 1
1 0 0	1 0 1		Φ 0	0 Φ	1 Φ
1 0 1	0 0 0		Φ 1	0 Φ	Φ 1

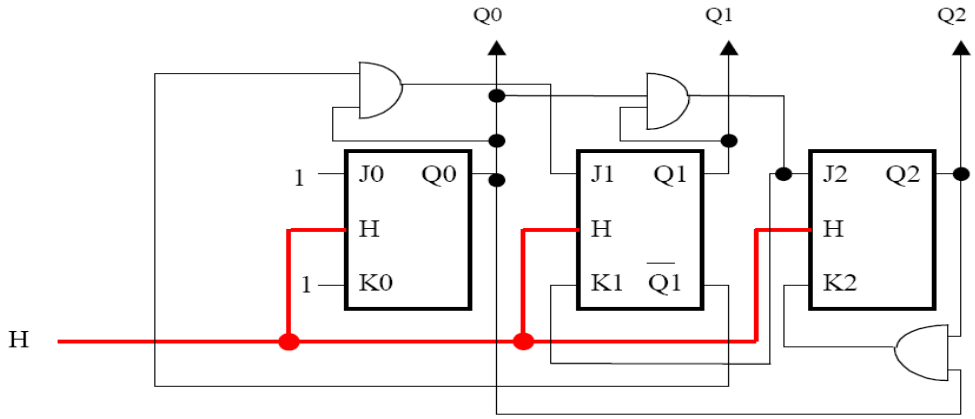
On remarque que : $J_0 = K_0 = 1$

Utilisant la méthode de karnaugh pour la simplification des équations de J_2 K_2 et J_1 K_1 on obtient :

$J_1 = \overline{Q_1}\ Q_0$, $K_1 = Q_1 Q_0$

$J_2 = Q_1 Q_0$, $K_2 = Q_2 Q_0$

Le circuit final de ce compteur est le suivant :



VII. Les compteurs synchrones particuliers

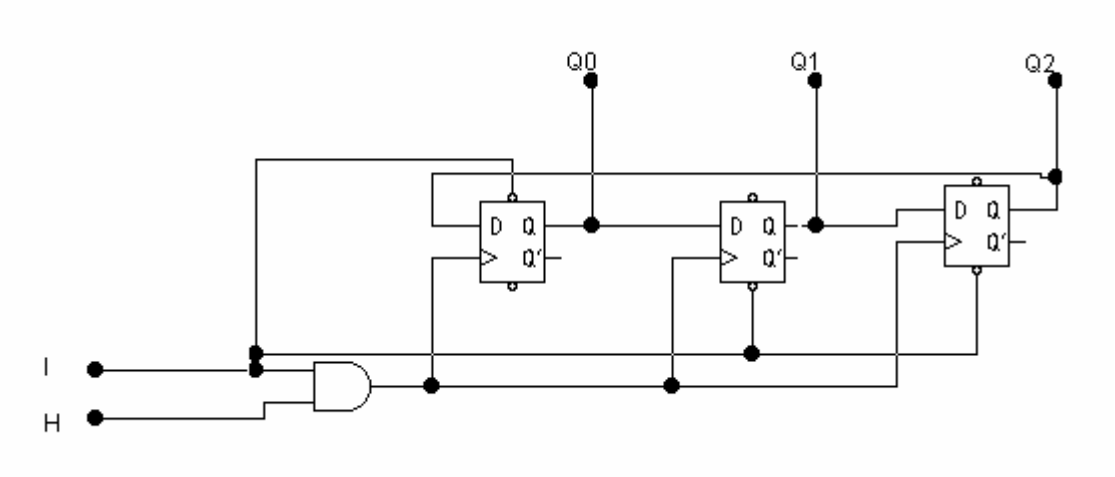
VII.1 Compteur en Anneau

Un compteur en anneau est un compteur circulaire, caractérisé par un bouclage de la sortie de la dernière bascule à l'entrée de la première bascule. En général, nous utilisons les bascules D. La caractéristique essentielle de ce type de compteur c'est la présence d'une seule bascule à 1 les autres sont à zéro. Les impulsions d'horloge jouent le rôle d'impulsion de décalage.

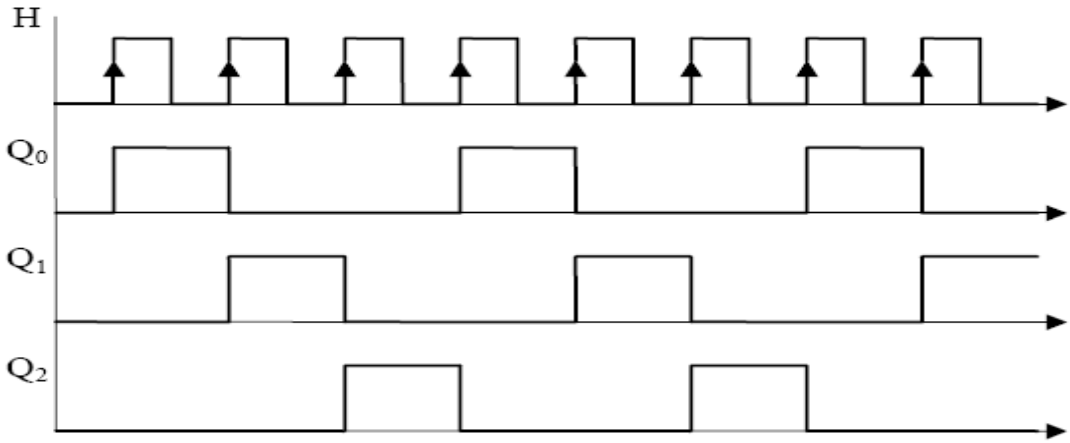
Le Problème principal dans ce type de compteur est qu'à la mise sous tension on' a pas forcément la séquence désiré. Il convient de forcer la première bascule à 1et les autres à 0 en rajoutant une porte AND et une entrée d'initialisation (I) comme l'indique le schéma suivant :

I = 0, initialisation, Q_0= 1, $Q_1 = Q_2$ =0

I= 1, fonctionnement normal.



Le chronogramme est :



Avec n bascules on aura n états, avec 3 bascules nous avons les états suivants : 001, 010, 100.

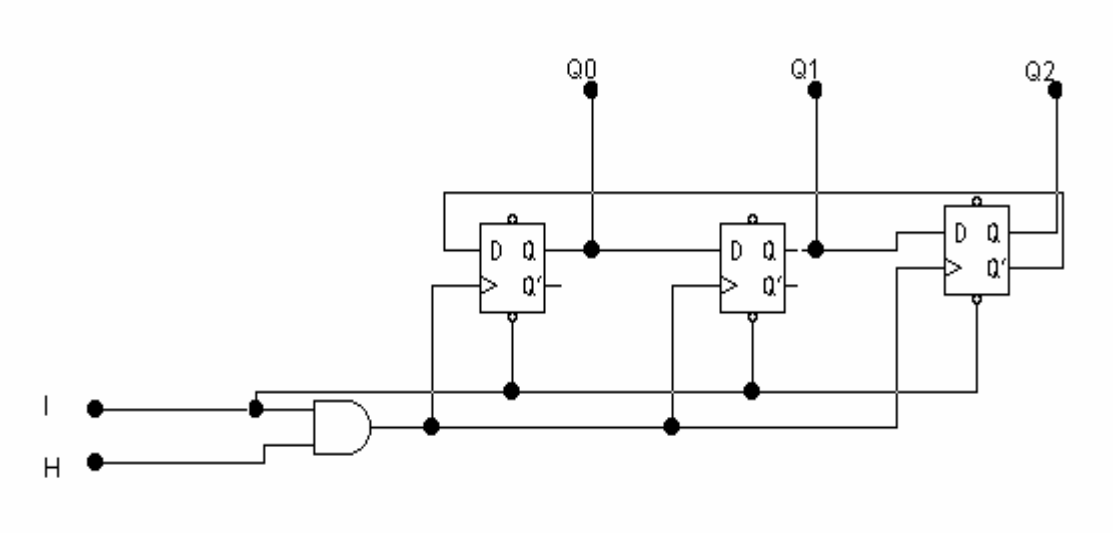
VII.2 Compteur de Johnson

Un compteur de johnson est un compteur circulaire, caractérisé par un bouclage de la sortie inversé $\overline{Q}$ de la dernière bascule à l'entrée de la première bascule. À la mise sous tension on a pas forcément toutes les bascules à 0. Il convient de les forcer toutes à 0 en rajoutant une porte AND et une entrée d'initialisation (I) comme l'indique le schéma suivant :

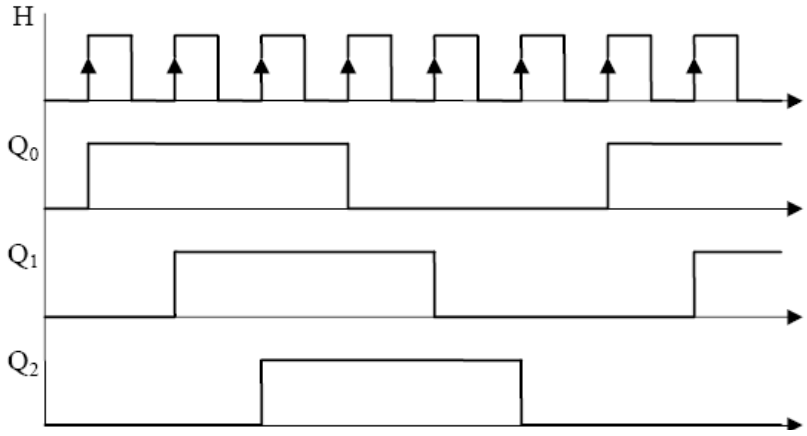
I = 0, initialisation, $Q_0 = Q_1 = Q_2 = 0$

I= 1, fonctionnement normal.

Le circuit du compteur est le suivant :



Le chronogramme est illustré sur la figure ci-dessous :



Avec n bascules on aura 2n états, avec 3 bascules nous avons les états suivants : 000, 001, 011, 111, 110, 100.

VIII. Compteur synchrone Intégré CI 74192 à 4 Bits :

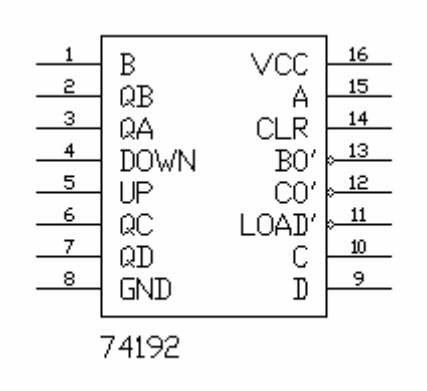
VIII.1 Structure

Le CI 74192 est un compteur/ décompteur à 4 bits et à chargement.

Le chargement se fait avec un zéro sur l'entrée LOAD.

- Si le chargement = 0, les sorties recopient les entrées c.à.d : $Q_A = A, Q_B = B$, $Q_C = C$, $Q_D = D$.

- Si le chargement = 1, nous aurons un compteur ou un décompteur selon l'entrée d'horloge choisie (UP/ DOWN), le comptage ou le décomptage commence à partir de la valeur rechargée.



L'entrée Clear sert à l'initialisation du compteur c.à.d la remise à zéro : $Q_A = 0$, $Q_B = 0$, $Q_C = 0$, $Q_D = 0$.

VIII.2 Utilisation du CI 74192 pour la réalisation des compteurs à cycle incomplet :

Exemple modulo 10 :

Lorsqu'on détecte l'état 10 (1010), on force le compteur à zéro en reliant la fonction de forçage $Z10 = Q_3 \overline{Q_2}\, Q_1 \overline{Q_0}$ vers l'entrée Clear du CI.

VIII.3 Applications du CI 74192

Le compteur numérique intégré est utilisé dans les fonctions de :

- Comptage d'événements,
- Mesure de fréquence, de période,
- Division de fréquence,
- Génération de séquences d'événements.

Chapitre 7 : Les Registres

I. Introduction

Un registre est un circuit séquentiel composé de *n* bascules synchrones commandées par le même signal d'horloge H, permettant de décaler à gauche ou à droite une série de bits de données. En pratique on distingue deux types de registres : les registres de *mémorisation* ou *tampon* et les registres *à décalage.*

Dans la première famille, les *n* bascules sont connectés d'une manière à mémoriser de façon temporaire un ensemble d'informations binaires.

Dans la deuxième famille, les *n* bascules sont connectées d'une façon que l'état logique de la bascule du rang (*i*) sera transmis à la bascule du rang (*i+1*) quand le front actif est appliqué aux entrées d'horloge de ses *n* bascules.

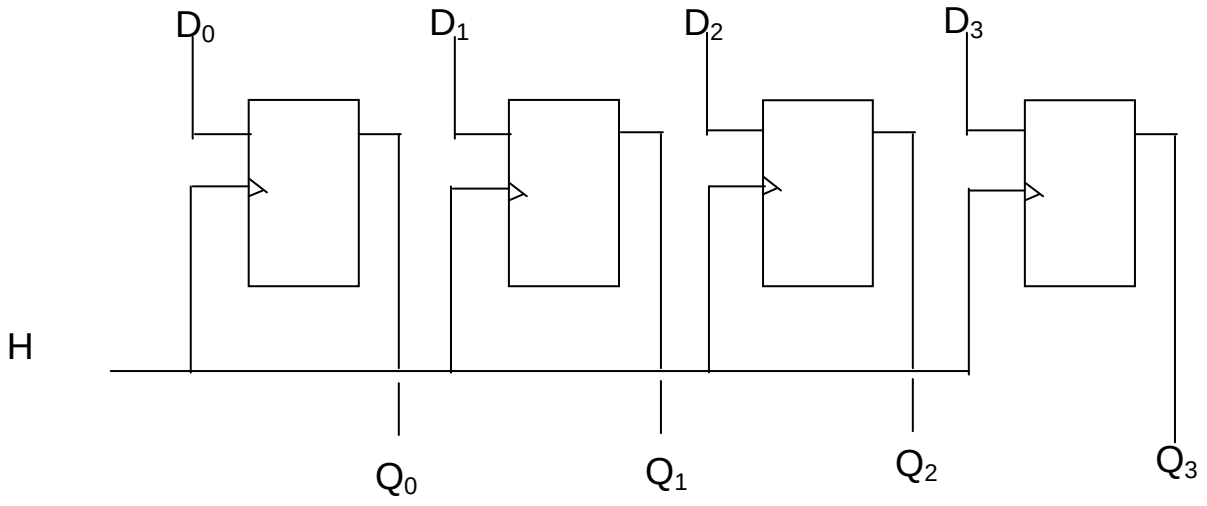
On y trouve généralement une entrée de données en série, N entrées de chargement en parallèle et N sorties. La Nième sortie peut servir de sortie série.

II. Registre de mémorisation

Un registre de mémorisation de n bits permet de mémoriser une information binaire durant une période du signal d'horloge, les *n* bascules sont indépendantes. Pour le fonctionnement on agit sur les entrées synchrones des bascules :

- Dans le cas des bascules D, on pose : $Di = Ei$
- Dans le cas des bascules JK, on pose : $Ji = Ei$ et $K = \overline{E_i}$

Exemple : Registre de mémorisation à 4 bits avec les bascules D



Les informations présentées en entrée (D_i) sont transférées en sortie (Q_i) lors de l'apparition du front actif d'horloge à l'instant (t) et restant mémorisées jusqu'au front actif d'horloge suivant à l'instant (t + T), même si durant cette période (T) les entrées (D_i) changent de valeur.

III Registre à décalage

Le problème consiste à trouver un circuit qui permet le transfert d'une valeur de la bascule du rang *(i)* vers une autre bascule du rang (*i+1*). Plusieurs architectures sont alors possibles pour effectuer cette opération, on cite 4 architectures de base :

- Registre entrée série sortie série
- Registre entrée série sortie parallèle
- Registre entrée parallèle sortie série
- Registre entrée parallèle sortie parallèle

Dans un registre à décalage *à droite* l'entrée de chaque bascule est reliée à la sortie de la bascule précédente. Seule la première bascule est reliée à l'entrée série E. A chaque front actif d'horloge, la valeur initiale de E est décalée d'un bit vers la droite (Application : division binaire par 2).

III.1 Entrée série

L'information (0,1) se présente bit après bit à l'entrée de la première bascule pour y être introduite à chaque impulsion d'horloge, il faut donc N impulsions d'horloge pour introduire une information de N bits.

III.2 Entrées parallèles

L'information de n bits est introduite simultanément par un seul front actif de l'horloge, chaque bascule doit être commandée séparément. Donc nous avons accès aux étages intermédiaires.

III.3 Sortie série

Les n bits de l'information passent dans le registre, elles sortent bit à bit au rythme du signal d'horloge par la sortie de la dernière bascule.

III.4 Sorties parallèles

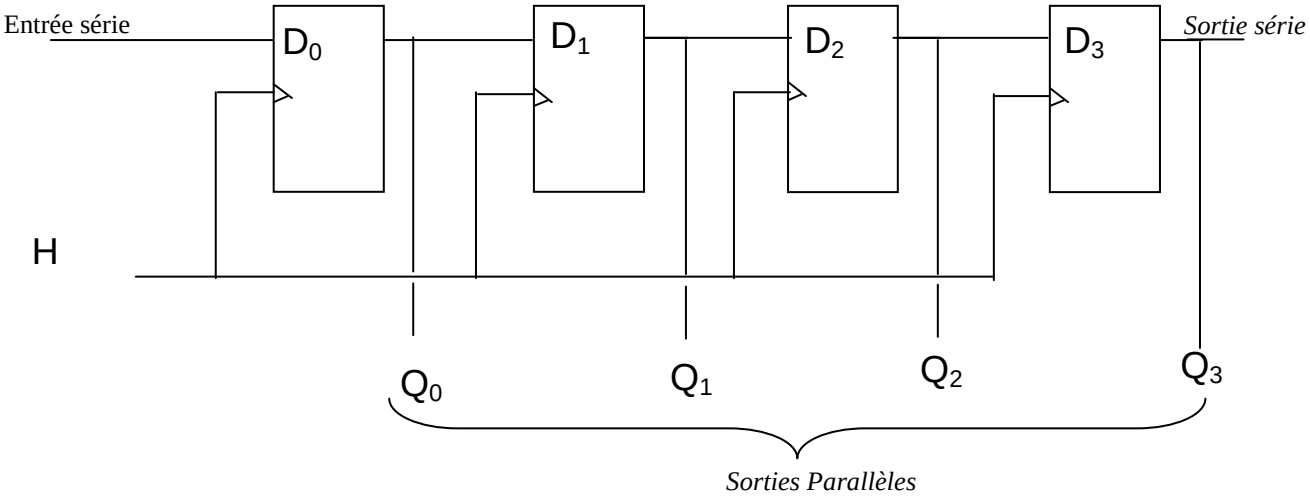
L'information de n bits peut être recueilli simultanément par les sorties de chacune des bascules

IV. Registre à décalage entrée série et sortie série – parallèle

Les registres de type série-série ou série parallèle, dans lesquels les informations sont décalées d'une bascule vers la suivante au rythme des impulsions d'une horloge, sont généralement réalisés avec des bascules D.

Le schéma de principe d'un registre à décalage (vers la droite) avec entrée série est présenté sur la figure ci-dessous. Au fur et à mesure des impulsions d'horloge, les données présentes sur l'entrée série sont transférées sur les différentes bascules. Dans ces conditions, la sortie *Qi* recopie, au moment de l'impulsion d'horloge, la valeur présente à son entrée.

La structure générale de ce registre avec les bascules D est :



V. Registre à chargement parallèle

La réalisation de ce type de compteur est basée sur l'utilisation des entrées de forçage R (reset) et S (set) de chaque bascule.

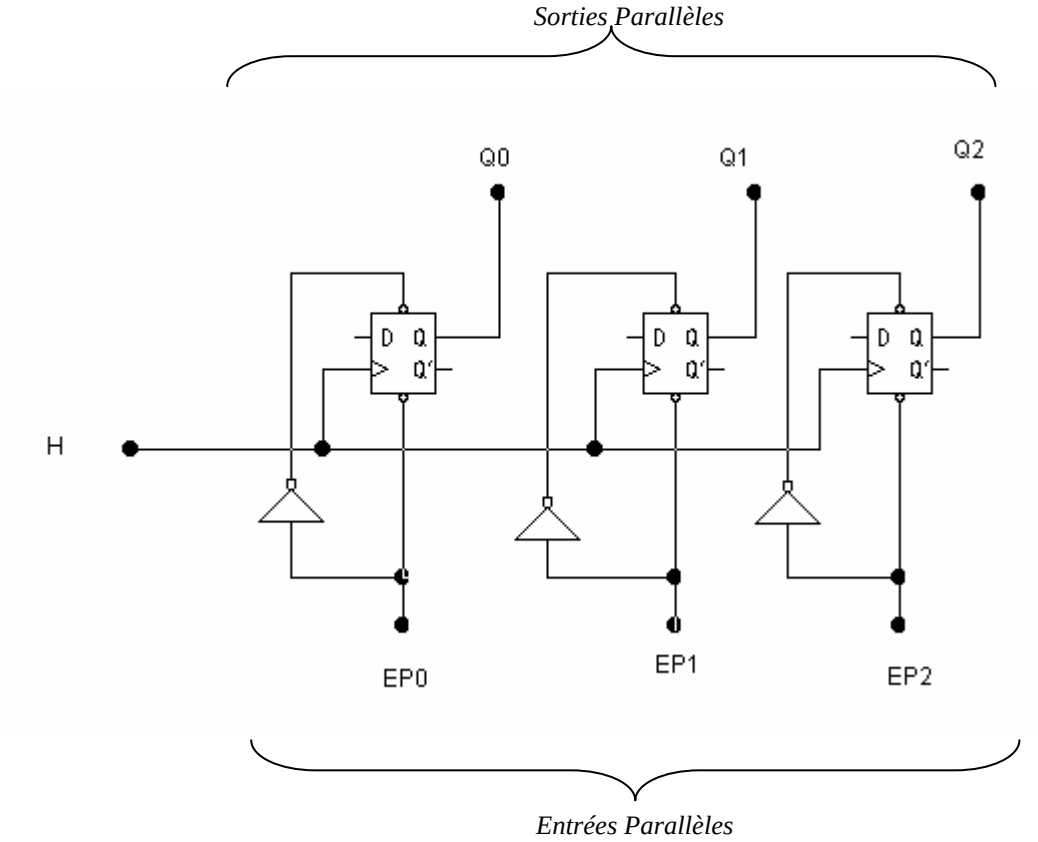
Soit E_p entrée parallèle :

Si E_p =0, on doit afficher un 0 sur la sortie de la bascule, donc on doit forcer la bascule à zéro, c.à.d : il faut mettre un zéro sur l'entrée R.

Si E_p =1, on doit afficher un 1 sur la sortie de la bascule, donc on doit forcer la bascule à un, c.à.d : il faut mettre un zéro sur l'entrée S.

E_p	*R*	*S*
0	*0*	*1*
1	*1*	*0*

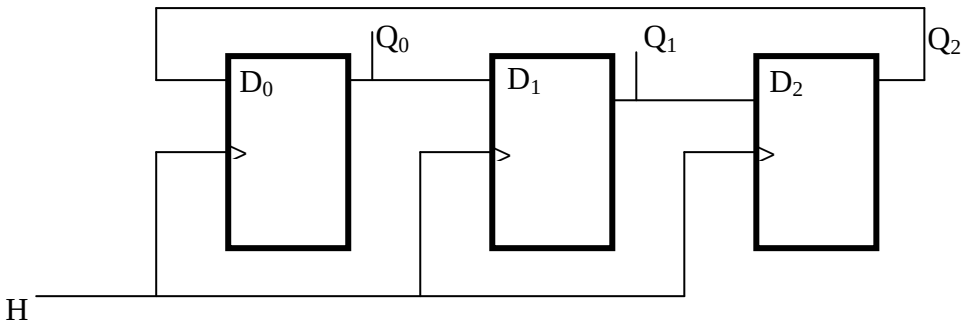
Alors : $S = \overline{E_p}$ et $R = E_p$



Dans ce type de registre, on ne trouve pas le décalage des bits de la gauche vers la droite et vis versa, on trouve uniquement un chargement de données en parallèle.

VI. Registre à rotation (ou circulaire)

C'est un registre qui possède une boucle de retour de la sortie vers l'entrée, on peut suivre le mouvement des bits via les sorties parallèles des bascules.



VII. Registre universel

C’est un registre qui possède une entrée de validation (s’appel Mode M). Son rôle est de sélectionner le mode de fonctionnement :

M= 0, Entrée série

M= 1, Entrées parallèles.

En sortie nous aurons une sortie série et plusieurs sorties parallèles. Il s’appel registre universel, car il permet de fonctionner en 4 modes possibles : Es/ Ss , Es/ Sp, Ep/ Ss, Ep/ Sp.

VIII. Applications des registres

Les registres à décalage ont de nombreuses applications. On cite :

- La mise en mémoire,
- La conversion de données série-parallèle utilisée dans la transmission de données. On prend des bits en série et on les convertit (généralement sur un octet) en parallèle (exemple : le SN74LS164).
- La conversion de données parallèle-série utilisée aussi en transmission de données. On prend des données binaires (généralement un octet) en parallèle et on les convertit en un train binaire série (exemple : le SN74LS165).
- Les opérations arithmétiques pour réaliser des multiplications et divisions (opérations utilisant le décalage).

CONCLUSION GENERALE

Dans ce support de cours, nous avons présenté une partie très importante du domaine de l'Electronique à savoir la logique combinatoire et séquentielle (LCS). Elle concerne l'introduction au domaine du numérique (la logique de toute ou rien). C'est un module essentiel dans la formation de l'ensemble des étudiants de diverses spécialités à savoir : Electronique, Automatique, Télécommunication, Electrotechnique,…

Grace à ses notions de base l'étudiant peut comprendre et assimiler d'autres futures modules dans son cursus de licence ou de Master telque : Architecture numériques avancées, les microprocesseurs et les microcontrôleurs, les circuits DSP, FPGA, …

Dans la partie logique combinatoire où la sortie du système ne dépend que de ses entrées, nous avons présenté l'ensemble des portes logiques et la plupart des circuits combinatoires de base.

Dans la partie logique séquentielle où la sortie dépend de ses entrées et de la sortie précédente (le temps intervient), nous avons donné les différents types de bascules et les circuits réalisables avec ces dernières, tels que les compteurs asynchrones et les compteurs synchrones, ainsi que les registres.

Ce support de cours se termine par l'ensemble de séries de Travaux Dirigés.

TD N° 1 :

SYSTEMES DE NUMEROTATION.

Ex 01 :

Exprimer dans les bases 2, 8 et 16 les nombres décimaux suivants : 35 ; 52 ; 12,25 ; 13,17.

Ex 02 :

Exprimer dans la base 10 les nombres suivants : $(1011,1011)_2$; $(567,2)_8$; $(FF1)_{16}$; $(5078)_8$.

Ex 03 :

Convertir en octal et en hexadécimal les nombres décimaux suivants : 666 ; 8430 ; 2570.

Ex 04 :

Déterminer l'équivalent décimal des nombres binaires suivants : 10,011011 ; 1,101101 ; 1,0001011.

Ex 05 :

Convertir les nombres décimaux suivants en héxadécimal puis directement en binaire : 831 ; 8797.

Ex 06 :

Convertir en DCB les nombres décimaux suivants : 4679 ; 12345 ; 18,18.

Ex 07 :

Convertir en décimal les nombres DCB suivants : $(1000111001,1)_{DCB}$; $(110010,010011)_{DCB}$.

Ex 08 :

Convertir en Gray les nombres suivants : $(10110)_2$; $(10110)_8$; $(FE1)_{16}$; $(46)_{10}$.

Ex 09 :

Convertir en décimal les nombres Gray suivants : (10110) ; (111111111111) ; (110101000).

Ex 10 :

Effectuer les additions suivantes en binaire :

1011,1101	110111,01	101111101
+ 110,1011	+ 110011,11	+ 111111011

Ex 11 :

Effectuer les soustractions suivantes en binaire :

1011,1101	110111,01	1110000
- 110,1011	- 110011,11	- 1101111

Ex 12 :

Effectuer les multiplications suivantes en binaire :

11011,01	10111,11
× 101,01	× 110

Ex 13 :

Effectuer les divisions suivantes en binaire :

(11011 ÷ 11) ; (10001110,10 ÷ 10111,11).

TD N° 2 :

Simplification et représentation des systèmes combinatoires.

Ex 01 :

A l'aide des propriétés de l'algèbre de Boole, démontrez que :

1) $\overline{\overline{a}} = a$.
2) $a + \overline{a}b = a + b$.
3) (a +b) (a +c) = a + b c.
4) $a + \overline{a}b + \overline{a}\overline{b}c + \overline{a}\overline{b}\overline{c}d + ... = a + b + c + d + ...$
5) $a + \overline{a}b + \overline{a}\,\overline{b}\,c + \overline{a}\,\overline{b}\,\overline{c}\,d = \overline{a}$.
6) $xy + xz + z\overline{y} = xy + z\overline{y}$.

Ex 02 :

Simplifier à l'aide des propriétés de l'algèbre de Boole, les expressions des fonctions logiques suivantes :

F1 = $a + abc + \overline{a}bc + \overline{a}b + ad + a\overline{d}$

F2 = $a\ b\ c + \overline{a}\ c$

F3 = $(a+b)(\overline{a} + \overline{b})$

F4 = $abc + a\overline{b}c + \overline{a}$

F5= $(\overline{a} + b)(a + b + d)\overline{d}$

Ex 03 :

Réaliser à l'aide des portes NAND puis à l'aide des portes NOR les fonctions logiques suivantes :

$Q = bc + ac + ab$ $\qquad$ $R = \overline{x} + yz$

$S = x \oplus y$ $\qquad$ $T = x \otimes y$

Ex 04 :

Ecrire les fonctions suivantes sous les 2 formes canoniques :

A) La somme des produits.
B) Le produit de la somme.

F1 = $a\ b + b\ c + a\ c$

F2 = $a + b\ c + \overline{b}\ d\ c$

F3 = $(a + b)(\overline{a} + b + d)$

F4 = $a\ (b\ +\ c\ )$

Ex 05 :

Simplifier les fonctions logiques à l'aide de la table de Karnaugh :

F1(a, b, c, d) = $\overline{a}\,\overline{c}\,d + \overline{a}\,cb + \overline{b}\,\overline{c}\,d + \overline{a}\,\overline{b}\,cd$

F2(a, b, c) = $\sum 0, 1, 3$

F3(a, b, c) = $\sum 0, 3, 4, 6, 7$

F4(a, b, c, d) = $\sum 5, 7, 13, 15$

F5(a, b, c, d) = $\sum 0, 2, 5, 6, 9, 11, 13, 14$

F6(a, b, c, d) = $\sum 0, 5, 9, 10$ et ϕ pour 2, 3, 8, 15.

F7(a, b, c, d) = $\sum 0, 3, 4, 5, 10$ et ϕ pour 1, 2, 6, 7, 8, 14, 15.

TD N° 3 :

Synthèse des systèmes combinatoires.

Ex 01 : *Commande d'une serrure.*

4 personnes A, B, C et D ont accès à un coffre.

- A ouvre le coffre en présence de B ou bien de C et D.
- B ouvre le coffre en présence de C et D.

Soit S la fonction qui donne l'état de la serrure (S = 1, pour serrure ouverte).

1- Donner la table de vérité de la serrure S.
2- Donner l'expression de la sortie S.
3- Simplifier l'expression de S avec le tableau de Karnaugh.
4- Tracer le logigramme de la serrure S à l'aide des portes NAND uniquement.

Ex 02 : *Générateur de parité*

Réalisez un circuit logique combinatoire ayant 4 entrées : a, b, c et d et une sortie S telle que :

S = 1 , si le nombre d'entrées au niveau haut est pair.
S = 0, sinon.

1- En utilisant les portes logiques.
2- En utilisant que 3 portes Ou-Exclusif à 2 entrées et une porte logique.

Ex 03 : *Commande de lampes.*

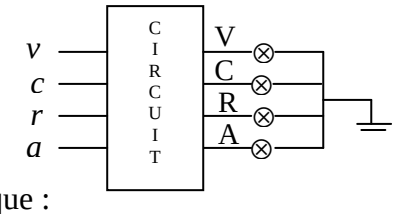
Trois interrupteurs A, B, C commandant l'allumage de 2 lampes R et S suivant les conditions suivantes :

- ♦ Dès qu'un ou plusieurs interrupteurs sont activés la lampe R doit s'allumer.
- ♦ La lampe S ne doit s'allumer que si au moins 2 interrupteurs sont activés.

Calculer les expressions des fonctions binaires R et S et dessiner le logigramme à l'aide de portes Non-Et uniquement.

Ex 04 : *Commande de phare d'un véhicule*

On dispose sur un véhicule de commandes (interrupteurs) *v, c, r* et *a* indépendants permettant la mise sous tension :



- ❖ Des vielleuses V.
- ❖ Des feux de croisements C (2 phares).
- ❖ Deux feux de route R (2 phares).
- ❖ Des anti-brouillard A (2 phares).

Les vielleuses V n'étant pas comptées comme des phares, il est précisé que :

- ➢ Quatre phares ne peuvent être allumés simultanément.
- ➢ Les feux R ont la priorité sur les feux A.
- ➢ Les feux de croisement C ont la haute priorité.
- ➢ Les veilleuses peuvent être allumées seules et que l 'allumage de A ou R ou C entraîne l'allumage de V.

1. Donner la table de vérité de V, R, A et C.
2. Etablir les tables de Karnaugh correspondant aux sorties V, C, R et A.
3. En déduire les expressions simplifiées des fonctions logiques de V, C, R et A en fonction des variables *v, c, r* et *a*
4. Réalisez le logigramme de ce dispositif de commande.

TD N° 4 :

Additionneurs & Soustracteurs.

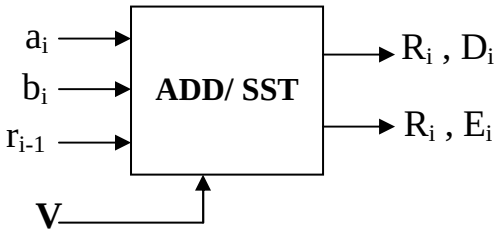
Ex 01 :

- Donner le schéma d'un :
  - Demi-additionneur ;
  - Additionneur ;
  - Demi-soustracteur ;
  - Soustracteur ;

Ex 02 :

A partir des résultats de l'exercice (01), établir le logigramme d'un additionneur-soustracteur complet avec une entrée de commande V tel que :

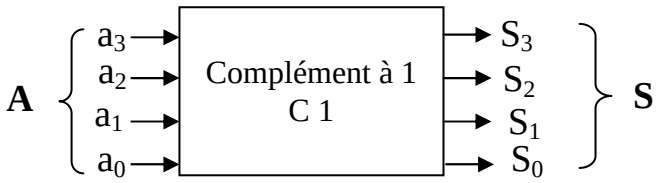
V = 0 ⟹ Addition.
V = 1 ⟹ Soustraction.



Ex 03 :

- Réaliser un circuit ***complémentateur à 1*** d'un nombre A à 4 bits tel que :

Si V=0 S = Complément à 1 de A.
Si V=1 S =A.



- Déduire le logigramme du circuit ***complémentateur à 2.***

Ex 04 :

Dans la représentation des nombres entiers ***(non signés)*** sur 8 bits, on donne :
$A = (1011\ 0111)_2$; $B = (1001\ 0101)_2$

1 - Trouver le complément à 1 (Complément restreint : ***CR***) de A et B.
2 - Trouver le complément à 2 (Complément vrai : ***CV***) de A et B.
3 - Effectuer l'opération **(A – B)** ; et **[A+ CV (B)]**, et puis conclure.

Ex 05 :

Soit A et B deux nombres : **$A = (F5)_{16}$ et $B = (6A)_{16}$.**
Si A et B sont considérés comme étant deux nombres signés représentés en **complément à deux** codés sur **8 bits** :

A- Donner les équivalents décimaux de A et B.
B - Faites les opérations : S1 = A+B ; S2 = A - B.
C - Donner les équivalents décimaux des résultats S1 et S2.

TD N° 5 :

Décodeurs & Transcodeurs.

Ex 01 :

- Donner le schéma d'un décodeur ¼ avec entrée de validation ***(V)*** à l'aide de portes ***NAND*** uniquement.

Ex 02 :

- Réaliser un décodeur **1/ 8** à l'aide de décodeurs **¼.**
- Réaliser un décodeur **1/ 8** à l'aide de décodeurs **1/16.**
- Réaliser un décodeur **1/16** à l'aide de décodeurs **¼.**

Ex 03 :

- Réaliser un additionneur 1 bit à l'aide de décodeur **¼** et de portes OR.
- Réaliser un additionneur complet à l'aide de décodeur **1/ 8** et de portes OR.

Ex 04 :

- Réaliser les fonctions suivantes à l'aide de décodeurs :
  $F_1(x,y) = x \oplus y$;
  $F_2(x,y) = x \, O \, y$;
  $F_3(a, b, c) = \sum 0, 3, 4, 6.$
  $F_4(a, b, c, d) = \sum 0, 1, 2, 4, 5, 8.$

Ex 05 :

- Donner le schéma d'un transcodeur :
  Gray ⟹ Binaire naturel.
  Binaire naturel ⟹ Gray.

Ex 06 :

- Réaliser un transcodeur 4 bits : DCB ⟹ DCBXS3.

(Sachant que le DCBXS3 c'est le DCB mais en ajoutant + 3).

- ✓ Etablir les équations simplifiées.
- ✓ Tracer son logigramme.

TD N° 6 :

MX, DMX & Comparateur.

Ex 01 :

- Réaliser un multiplexeur à 4 entrées d'informations à l'aide de portes logiques.
- Réaliser un démultiplexeur à 2 entrées d'adresses à l'aide de portes logiques.

Ex 02 :

- Réaliser les Fonctions suivantes à l'aide de multiplexeur (MX) :

  F_1 (a, b, c) = $\overline{a}\,\overline{b}\,\overline{c} + \overline{a}\,b\,c + a\,\overline{b}\,\overline{c} + a\,b\,\overline{c}$

  F_2(a, b, c, d) = $\sum$ 0, 1, 2, 4, 5, 8.

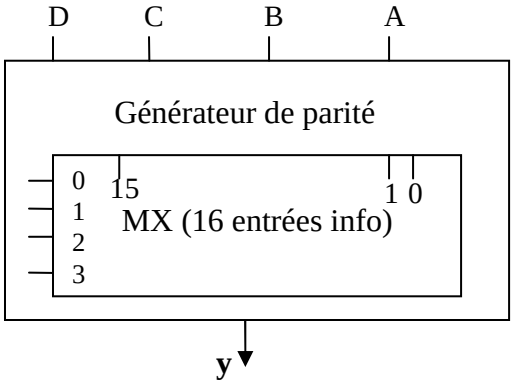
Ex 03 :

- Réaliser la fonction ci-dessous à l'aide :
  1. Multiplexeur à 3 entrées d'adresses.
  2. Multiplexeur à 2 entrées d'adresses seulement.

Avec : F = $AC + \overline{A}\,\overline{B} + AB$

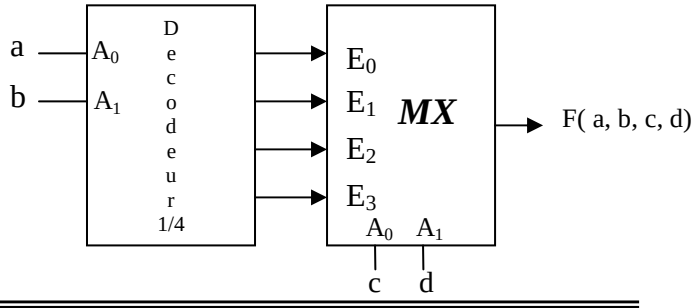
Ex 04 :

Soit un multiplexeur à 16 entrées d'informations. Utiliser ce Mx pour construire un dispositif qui analyse un mot de 4 bits appliqué aux entrées : A, B, C, D et qui détermine si le nombre de bits sur ABCD est pair (sortie Y=0), ou impair (sortie Y =1).
Faire un schéma logique de ce contrôle de parité.



Ex 05 :

Etant donné le schéma ci-contre, établir l'expression de la fonction logique F(a, b ,c, d). Considérez que A1 et A0 sont respectivement le poids fort et le poids faible pour le décodeur et le multiplexeur.



Ex 06 :

- Concevoir un comparateur de 2 bits (A= $a_1\,a_0$, B= $b_1 b_0$) à l'aide de portes logiques.

Ex 07:

- Concevoir un comparateur de 5 bits à l'aide de :
  A. 2 circuits intégrés (CI) 7485.
  B. 1 circuit intégrés (CI) 7485 et des portes logiques.

TD N°7 :

Les Bascules.

Ex 01 :

Réaliser une Bascule **RS** à l'aide des portes NOR, puis à l'aide des portes NAND.

Ex 02 :

Réaliser une Bascule **RST** (c'est une Bascule RS avec signal d'horloge T) à l'aide des portes NAND uniquement.

Ex 03 :

Réaliser une Bascule **RST Maître-Esclave** à l'aide des portes NAND uniquement.

Ex 04 :

Réaliser :

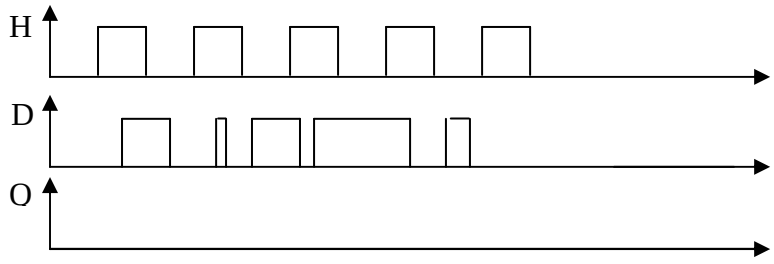
- ✓ Une Bascule **RS** à l'aide de Bascule : **JK, D**.
- ✓ Une Bascule **JK** à l'aide de Bascule : **RS, D**.
- ✓ Une Bascule **D** à l'aide de Bascule : **RS, JK**.

Ex 05 :

Réaliser un diviseur de fréquence par 2 à l'aide des Bascules :
RS, **JK** et **D**.

Ex 06 :

Soit une Bascule D active sur fronts montants du signal d'horloge H et attaquée par le signal suivant sur son entrée D :



A- Tracer le chronogramme du signal Q.

B- Même question dans le cas d'une Bascule D active aux fronts descendants du signal d'horloge H.

Ex 07 :

Soit une Bascule AB active sur le front descendant du signal d'horloge H et ayant une sortie :

$$Q^{+} = B + (A \oplus Q).$$

A- Donner la table de transition de cette Bascule.
B- Réaliser à l'aide de cette Bascule un diviseur de fréquence par 2.
C- Réaliser la Bascule AB à l'aide de la Bascule RS et D.

TD N° 8 :

Les Compteurs binaires.

Ex 01 :
Etudier le schéma d'un compteur/decompteur asynchrone en utilisant des bascules JK, actives sur front montant ou descendant de l'horloge.

Ex 02 :
Donner le schéma d'un compteur asynchrone modulo 8 réalisé à l'aide de bascule JK, RS, D actives sur front descendant de l'horloge.

Ex 03 :
Donner le schéma d'un compteur asynchrone modulo 10 réalisé à l'aide de bascule JK, RS, D actives sur front descendant de l'horloge.

Ex 04 :
Donner le schéma d'un compteur asynchrone ayant pour cycle : 0, 1, 2, 6, 7, 8 réalisé à l'aide de bascule JK active sur front montant de l'horloge.

Ex 05 :
Donner le schéma d'un compteur asynchrone ayant pour cycle : 0, 2, 4, 8 réalisé à l'aide de 2 bascules D uniquement et un circuit combinatoire.

Ex 06 :
Donner le schéma d'un compteur synchrone modulo 8 réalisé à l'aide de bascule JK, RS, D.

Ex 07 :
Donner le schéma d'un compteur synchrone modulo 7 réalisé à l'aide de bascule D actives sur front montant de l'horloge.

Ex 08 :
Donner le schéma d'un compteur synchrone ayant pour cycle : 0, 2, 1, 4 réalisé à l'aide de bascule D, actives sur front descendant de l'horloge, selon deux méthodes :

a. Avec 3 bascules
b. Avec 2 bascules uniquement et un circuit combinatoire.

TD N° 9 :

Les registres à Décalage.

Ex 01 :

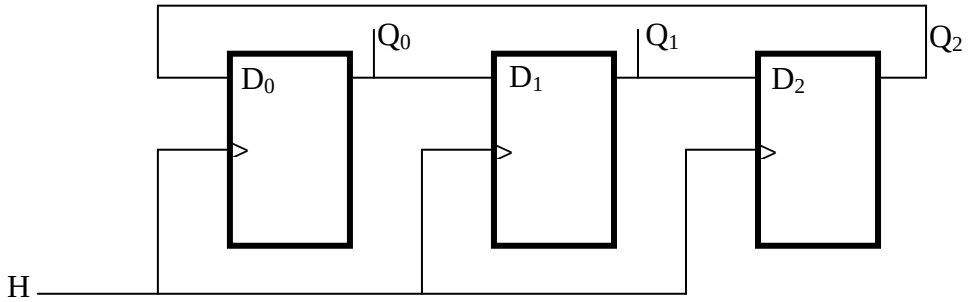
Réalisez un registre à décalage **à droite** à 3 bits avec :

- ✓ Entrée série.
- ✓ Sortie série.

A l'aide des bascules D actives sur le front montant de l'horloge H.

Ex 02 :

Analyser le fonctionnement du circuit séquentiel suivant :



Et déduire le type de ce circuit.

Ex 03 :

Réaliser un registre à décalage à **droite avec rotation** sur 3 bits à base de bascule RS actives sur le front montant de l'horloge H.

Ex 04 :

Etablir le circuit d'un registre à décalage à droite ayant 4 bits à entrée série ou parallèle suivant la commande :

- M=0, entrée série.
- M=1, entrée parallèle.

Ce registre sera conçu avec des bascules D actives sur le front montant de l'horloge H.

Ex 05 :

Même question que l'exo 04 mais, mais en utilisant les entrées de forçage (R et S) des Bascules D.

Références Bibliographiques

[1] : Jean Letocha : « Introduction aux circuits logiques », Mc Graw-Hill, 1985

[2] : Jean-Paul Vabre (Auteur) Jean-Claude Lafont (Auteur) : « Cours et problèmes d'électronique numérique » Ellipses 1998.

[3] : Roger L Tokheim : « Techniques numériques cours et problèmes», Mc Graw-Hill 1994.

[4] : Noël Richard : « Électronique numérique et séquentielle» Sciences Sup, Dunod 2009.

[5] : Rached Tourki : « Electronique numérique Cours et exercices corrigés », Dunod 2005.

[6] : C. ALEXANDRE : « Circuits numériques : 1ère partie » 2004 http://easytp.cnam.fr/alexandre/index_fichiers/support/ele015_cours_vol1.pdf

[7] : Mouloud Sbai : « Logique Combinatoire & Composants Numériques Cours & Exercices Corrigés », Ellipses 2013.

[8] Marcel Gindre : « Logique combinatoire et technologie, cours et exercices », Hermès 1987.

[9] :http://www.larmand.fr/fichiers/Ancien_site/enseigne/ressources/techno/bourse%20cours/COURS/Electronique%20numerique%20cabl%C3%A9e.pdf

[10] : http://www.gecif.net/articles/genie_electrique/cours/livre_electronique_numerique.pdf

[11] :http://thierryperisse.free.fr/documents/electronique_numerique/CNED/POLY%20CNED%201.pdf

[12] :http://www.estusmba.ac.ma/benbrahim/ENSA/Electronique%20num%C3%A9rique/semestre1-ELN-Num%C3%A9rique.pdf