Ministère de l'Enseignement Supérieur et
de la Recherche Scientifique
****
Direction Générale des Etudes Technologiques
****
Institut Supérieur des Etudes Technologiques de
Djerba
****

Support de Cours

Systèmes logiques 1

Élaboré par :

Slah MHAYA

(Technologue ISET de DJERBA)

Public cible :

Classes de 1ère année

Licence Génie Électrique

Version 2015

Avant propos

Suivi des versions du support

Version	Date	Rédigé Par	Raison
1.0	Sep 2013	Slah MHAYA	Enseignement de la matière
2.0	Sep 2015	Slah MHAYA	Révision et Amélioration globale du support Création de la 2ème version du support

Pré requis

Notions élémentaires de l'algèbre binaire

Objectifs généraux

A l'issue de ce cours, l'étudiant(e) sera capable de :

- Faire une conversion entre les différentes bases d'un système de numération.
- Traiter des opérations arithmétiques et faire des calculs dans les différentes bases.
- Comprendre et appliquer l'ensemble de théorèmes de l'algèbre de Boole.
- Simplification des expressions logiques algébriquement et graphiquement (tableau de KARNAUGH)
- Connaître les différentes fonctions intégrées de la logique combinatoire (Codage, décodage, transcodage, circuits de transfert d'informations, circuits arithmétiques)

Niveau cible

Génie Électrique (Licence 1)

Volume horaire

- 1h 30 de cours intégré.

  Soit en total : 22,5h

Moyens pédagogiques

- Support de cours papier.
- Séries de travaux dirigés.
- Sujets de contrôle continu.

Evaluation

- Coefficient : 2
- Devoir de contrôle : 32%
- Note non présentielle (devoirs à la maison) : 20%
- Devoir de synthèse: 48%

Table de matières

Liste des figures

Chapitre : 1

Systèmes de Numération

Objectifs

Général

- Comprendre la notion de systèmes de numération
- Maitriser les règles de représentations des systèmes de numération.

Spécifiques

- Etre capable de faire une conversation entre les différentes bases.
- Traiter des opérations arithmétiques et faire des calculs dans des bases appropriées.
- Faire le codage des entiers naturels et des entiers signés en complément à 2
- Connaitre différents systèmes de codage

Plan du chapitre

I. Rappel : systèmes de numérations

II. Formule mathématique de conversion d'un nombre d'une base quelconque dans la base décimale

III. Conversion d'un nombre de la base décimale vers une base B

IV. Conversion d'une base quelconque vers une base quelconque

V. Opérations arithmétiques dans les bases

VI. Codage des entiers

VII. Les systèmes de codage

Volume horaire

4 heures et demi

1 Rappel : Systèmes de numérations

On appelle système de numération un ensemble fini de symboles plus une stratégie de représentation qui nous permet de donner une représentation d'un nombre dans le système en question. Cet ensemble fini de symboles est appelé *la base du système de numération.*

Exemple :

⇨ *La base décimale :* C'est la base du système de numération décimal noté B_{IO}.

B_{10} = { 0 ; l ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 }

1298 est un nombre exprimé dans ce système décimal. On note $(1298)_{10}$ ou 1298.

⇨ *La base binaire :* C'est la base du système de numération binaire noté :

B_2 = { 0 ; 1 }

$(10)_2$; $(101)_2$; $(111)_2$; $(12) \notin B_2$.

⇨ *La base octale* : C'est la base du système de numération octal noté :

B_8 = { 0 ; l ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 }.

$(271)_8 \in B_8$; $(309)_8 \notin B_8$.

⇨ *La base hexadécimale :* C'est la base du système de numération

Hexadécimal noté :

B_{16} = {0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; A ; B ; C ; D ; E ; F}

$(A)_{16}$ représente une quantité de dix unités

$(B)_{16}$ représente une quantité de onze unités

$(C)_{16}$ représente une quantité de douze unités

$(D)_{16}$ représente une quantité de treize unités

$(E)_{16}$ représente une quantité de quatorze unités

$(F)_{16}$ représente une quantité de quinze unités

Remarque importante :

La base d'un système B_n contient n symboles associés respectivement à n quantités de valeurs respectives 0; 1; 2; 3;...; n-1.

Exercice :

Donner les éléments des bases 6, 4, 7. Donner un nombre dans chaque base.

Solution :

$B_6 = \{ 0 , 1 , 2 , 3 , 4 , 5 \}$; $(125)_6 \in B_6$

$B_7 = \{ 0 , 1 , 2 , 3 , 4 , 5 , 6 \}$; $(1703)_7 \in B_7$

$B_4 = \{ 0 , 1 , 2 , 3 \}$; $(103)_4 \in B_4$

2 Formule mathématique de conversion d'un nombre d'une base quelconque dans la base décimale

Soit $N = (\sim_p \sim_{p-1} ..... \sim_1 \sim_0)$ un nombre dans une base B. alors $0 \leq \sim_i < B$. L'objectif est de trouver une écriture de N dans la base 10. **Exemple :**

$(1252)_{10} = 2 * 10° + 5 * 10^1 + 2 * 10^2 + 1 * 10^3$

Si on désigne par B la valeur 10 de la base alors on aura :

$(1252)_{10} = 2*B^0+5*B^1+2*B^2+1*B^3$

Généralisation:

Soit B une base et $0 \leq \sim_i < B$ pour $0 \leq i \leq p$. On a la formule générale de conversion vers la base décimale suivante :

$$(\sim_p \sim_{p-1} ..... \sim_1 \sim_0)_B = (\sim_p * B^p + \sim_{p-1} * B^{p-1} + ... + \sim_1 * B^1 + \sim_0 * B^0)_{10}$$

Remarque :

Les $\sim_i$ et B sont convertis aussi vers la base 10.

Exercice:

Convertir en base 10 le nombre $(32)_4$.

Solution

En utilisant la formule générale

$$(32)_4 = 3 * 4^1 + 2 * 4^0 = (14)_{10}$$

3 Conversion d'un nombre de la base décimale vers une base B

Soit N un nombre décimal et B une base.

Objectif :

Trouver $0 \leq \sim_i < B$ pour $0 \leq i \leq p$ tel que $(N)_{10} = (\sim_p \sim_{p-1} ..... \sim_1 \sim_0)_B$.

On veut écrire N sous la forme :

$N = (\sim_p * B^p + \sim_{p-1} * B^{p-1} + ... + \sim_1 * B^1 + \sim_0 * B^0)_B$

Et ceci dans le but de déduire les $\sim_i$ avec $0 \leq i \leq p$ comme représentation de N dans la base B.

Solution:

$(N)_{10} = (\sim_p \sim_{p-1} ..... \sim_1 \sim_0)_B$

On a $N = \sim_0 + \sim_1 * B^1 + ... + \sim_{p-1} * B^{p-1} + \sim_p * B^p$

$= \sim_0 + B(\sim_1 + \sim_2 * B^1 + ...+ \sim_{p-1} * B^{p-2} + \sim_p * B^{p-1})$

$N = \sim_0 + BQ_1$ avec $0 \leq \sim_{i0} < B$ et $<B$ et $0 \leq Q_1 < N$

D'où Q_1 est le quotient de la division euclidienne de N par B et $\sim_0$ est le reste de cette division.

$Q_1 = \sim_0 + \sim_2 * B^1 + ... + \sim_{p-1} * B^{p-2} + \sim_p * B^{p-1}$

$Q_1 = \sim_0 + B(\sim_2 + \sim_3 * B^1 + ... + \sim_{p-1} * B^{p-3} + \sim_p * B^{p-2})$

$Q_1 = \sim_1 + BQ_2$ avec $0 \leq \sim_{i1} < B$ et $<B$ et $0 \leq Q_2 < N$

D'où Q_2 est le quotient de la division euclidienne de Q_1 par B et $\sim_{i1}$ est le reste de cette division.

La condition d'arrêt est que le quotient de la division euclidienne soit nul.

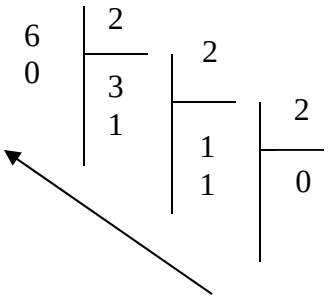
Remarque:

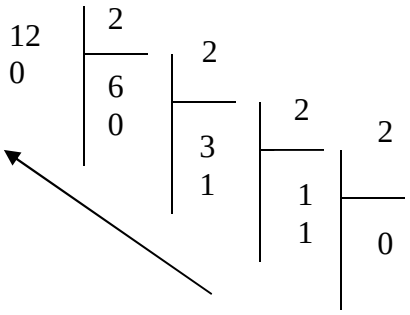
Cette méthode est reconnue sous le nom de la division successive par B.

Application :

Conversion d'un nombre de la base décimale vers la base binaire B

Convertir en binaire les nombres : 6 et 12





$(6)_{10} = (110)_2$ $(12)_{10} = (1100)_2$

Base décimale	Base binaire	Base octale	Base hexadécimale
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Exercice :

$(125)_{10} = (\ ?\ )_3$

Solution :

$(125)_{10} = (22102)_3$

4 Conversion d'une base quelconque vers une base quelconque

Soient B_I et B_{II} deux bases :

Pour convertir un nombre de la base B_I vers la base B_{II} on peut procéder comme suit :

Convertir le nombre de B_I vers B_{10} (base décimale)

Méthode : formule générale

Convertir le résultat obtenu de la base décimale B_{10} vers B_{II}

Méthode : division successive par B_{II}

4.1 Conversion de la base binaire vers la base octale

Soit (N) = $(r_p\, r_{p-1}$ ..... $r_1\, r_0)_2$ tel que $0 \le r_j \le 1$

Objectif :

Trouver $\sim_q \sim_{q-1} \ldots\ldots \sim_1 \sim_0$ tel que $0 \le \alpha_i \le 7$ pour $0 \le i \le q$

Et $(r_p\, r_{p-1}$ ..... $r_1\, r_0)_2 = (\sim_q \sim_{q-1} \ldots\ldots \sim_1 \sim_0)_8$

Méthode :

$8 = 2^3$ => Regrouper par groupe de ***3 les*** r_i à commencer à partir de r_0

Convertir le groupement $(r_{j+2} r_{j+}1 r_j)$ —> La base 8

Le premier groupement sera α_0

Exemple:

$(101011)_2 = (5\ 3)_8$ (regroupement de 3 bits)

$(011)_2 = (3)_{10} = (3)_8$

$(101)_2 = (5)_8$

4.2 Conversion de la base octale vers la base binaire

Soit $(r_p r_{p-1}$ ..... $r_1 r_o)_8$ tel que $0 \le r_j \le 7$ pour $0 \le j \le p$

Objectif :

Trouver $\sim_q \sim_{q-1} \ldots\ldots \sim_1 \sim_0$ tel que $0 \le \alpha_j \le 7$ pour $0 \le i \le q$

Et $(r_p r_{p-1}$ ..... $r_1 r_o)_8 = (\sim_q \sim_{q-1} \ldots\ldots \sim_1 \sim_0)_2$

Méthode :

C'est l'inverse de la précédente :

Pour chaque j, $0 \le j \le q$; Convertir r_j *vers* La base 2 sur 3 *bits*

Exemple :

$(010001)_2 = (21)_8$

4.3 Conversion de la base binaire vers la base Hexadécimale

Soit N =$(r_p r_{p-1}$..... $r_1 r_0)_2$ tel que $0 \le r_j \le 1$ pour $0 \le j \le p$

Objectif :

Trouver $\sim_p \sim_{p-1} \ldots\ldots \sim_1 \sim_0$ tel que $0 \le \alpha_i \le F$ pour $0 \le i \le p$

Et $(r_p r_{p-1}$..... $r_1 r_0)_2 = (\sim_p \sim_{p-1} \ldots\ldots \sim_1 \sim_0)_{16}$

Méthode :

$16 = 2^4$ => Regrouper les r_i en groupe de 4 à commencer par r_o

Exemple :

$(00111101)2 = (3D)_{16}$ (regroupement de 4 bits)

$(0000010011100101)_2 = (04E5)_{16}$ (regroupement de 4 bits)

4.4 Conversion de la base hexadécimale vers la base binaire

Soit $(r_p r_{p-1} \ldots\ldots r_1 r_0)_{16}$

Objectif :

Trouver $\sim_q \sim_{q-1} \ldots\ldots \sim_1 \sim_0$ tel que $0 \leq \alpha_i \leq F$ pour $0 \leq i \leq q$

Et $(r_p r_{p-1} \ldots\ldots r_1 r_0)_{16} = (\sim_q \sim_{q-1} \ldots\ldots \sim_1 \sim_0)_2$

Méthode ;

C'est l'inverse de la précédente :

Convertir chaque $0 \leq r_j \leq F$ pour $0 \leq j \leq p$ *vers* La base 2 sur *4 bits*

$(r_p r_{p-i} \ldots\ldots r_i r_0)_{16}$ Convertir vers B_2 sur 4 bits

Exemple :

Convertir *vers* B_2

$(FA)_{16} = (11111010)_2$

4.5 Conversion de la base Hexadécimale à la base octale

Méthode :

Soit $N \in B_{16}$

$(N)_{16} \longrightarrow (N')_2$ (conversion de chaque chiffre sur 4 bits)
$(N')_2 \longrightarrow (N'')_8$ (Regroupement par 3 bits

Exemple :

$(FA)_{16} = (11111010)_2 = (372)_8$

5 Opérations arithmétiques

Les opérations arithmétiques s'effectuent en base quelconque b avec les mêmes méthodes qu'en base 10. Une retenue ou un report apparait lorsque l'on atteint ou dépasse la valeur b de la base.

Exemple 1 : additionner les nombres $(110010111)_2$ et $(1010011)_2$

```
    1  1 1 1     Retenues
  1 1 0 0 1 0 1 1 1   Premier terme
+   1 0 1 0 0 1 1     Second terme
  ---------------
  1 1 1 1 0 1 0 1 0   Somme
```

Exemple 2 : Soustraire les nombres $(524)_8$, $(263)_8$

```
   1        l'emprunt de la 2ème colonne
   5 2 4    Premier terme
 - 2 6 3    Second terme
 -------
   2 4 1    Différence
```

Exemple 3 : Multiplier les nombres $(2A)_{16}$, $(1E)_{16}$

```
 2  8       Retenues
    2 A     Premier terme
 *  1 E     Second terme
 -------
   2 4 C
 + 2 A .
 -------
   4 E C    Resultat
```

Exemple 4 : Diviser les nombres $(1111010)_2$, $(1011)_{16}$

```
  1 1 1 1 0 1 0 | 1 0 1 1
- 1 0 1 1       | 1 0 1 1
  ---------     |
    1 0 0 0     |
-   0 0 0 0     |
  -----------   |
    1 0 0 0 1   |
-     1 0 1 1   |
    -----------
        1 1 0 0
-       1 0 1 1
      -------
        reste : 1
```

6 Codage des entiers :

6.1 Représentation d'un entier naturel (non signé)

Un entier naturel est un entier positif ou nul, coder cet entier revient à le convertir en binaire et d'utiliser un nombre des bits suffisant pour le représenter. D'une manière générale un codage sur **n** bits permet de représenter $\mathbf{2^n}$ nombres naturels dont la valeur est comprise entre **0** et $\mathbf{2^n-1}$.

Exemple : Pour coder des nombres naturels compris entre 0 et 7, il faut utiliser 3 bits car (2^3=8 positions).

6.2 Représentation d'un entier avec signe (signé)

Un entier signé est un nombre qui peut être positif ou négatif. Il faut le coder de telle façon que l'on puisse savoir s'il s'agit d'un nombre positif ou négatif, et de plus il faut conserver les règles d'addition (le nombre + son négatif = 0).

Pour coder cet entier, on réserve le bit de poids le plus fort (le bit le plus à gauche) pour le signe, il prend la valeur **0** pour le signe positif et **1** pour le signe négatif. Ce qui implique que la plus grande valeur codée avec n bit est $\mathbf{2^{n-1}-1}$, par contre la plus petite valeur est $\mathbf{-2^{n-1}}$**.** Le nombre des entiers signés codés sur n bit est égal à $\mathbf{2^n}$

Exemple : si n=4 le nombre le plus grand sera 0111 (7 en base décimale).

6.3 Principe de complément à deux (complément vrai)

Le complément vrai ou complément à deux est utilisé pour coder les entiers signés, il suffit donc de :

– **Coder** le nombre en binaire (base 2) comme un entier naturel sur n-1 bits,

– **Complémenter à un** chaque bit en l'inversant, c'est-à-dire que l'on remplace les zéros par des 1 et vice-versa)

– **ajouter** 1 à ce complément

Pour vérifier le codage, il suffit d'additionner le nombre et son complément à deux, il faut donc que le résultat d'addition soit égal à 0.

Exemple On désire coder la valeur -5 sur 8 bits par le complément à deux. Il suffit :

– D'écrire 5 en binaire sur 8 bits : 00000101

– De complémenter à 1 : 11111010

– D'ajouter 1: 11111011

– La représentation binaire de -5 sur 8 bits est donc 11111011.

Le bit de poids fort est 1, on a donc bien un nombre négatif

Si on ajoute 5 et -5 (00000101 et 11111011) on obtient 0 (avec une retenue de 1)

7 Les systèmes de codage

7.1 Introduction

Pour pouvoir traiter l'information dans l'ordinateur, il faut que cette dernière soit codée en binaire. Pour cela, on trouve plusieurs systèmes de codage en plus du système binaire naturel déjà vu au début de ce chapitre.

7.2 Codage BCD (Binary Coded Décimal)

Ce code conserve les avantages du système Décimal et du code binaire. Il est utilisé par les machines à calculer.

On fait correspondre à chaque caractère du système décimal un mot du code binaire de 4 bits, on a alors :

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

Exemple : conversion décimale BCD

$(19)_{10} = (00011001)_{BCD}$

7.3 Code Gray ou Binaire réfléchi

Ce système de codage est très important pour la simplification des fonctions logique qu'on verra dans les prochains chapitres. Il se présente comme suit :

Décimal	Code Gray sur 3 bits		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Inversion

Symétrie

- **Propriétés de la table de conversion**

  ↳ 2 codes Gray successifs se diffèrent par l'état d'un seul bit.

  ↳2 codes Gray symétriques par rapport à un axe de symétrie se diffèrent par l'état d'un seul bit

  Exemple :

  Deux codes voisines : $(3)_{10} = (010)_{Gray}$

  $(4)_{10} = (110)_{Gray}$

  Deux codes symétriques : $(7)_{10} = (100)_{Gray}$

  $(4)_{10} = (110)_{Gray}$

- **Autre représentation du code Gray : le tableau de Karnaugh**

  Le tableau de karnaugh est un tableau dont les lignes et les colonnes sont codés en code Gray, le numéro décimal de la case d'un tableau a comme équivalent en code Gray celui formé par le code Gray de la ligne suivit du code Gray de la colonne.

  Exemple :

CD / AB	00	01	11	10
00	0	1	2	3
01	7	6	5	4
11	8	9	10	11
10	15	14	13	12

↳Le code Gray de la case numéro 6 est 0101

$$\text{d'où } (6)_{10} = (0101)_{Gray}$$

↳Le cade Gray de la case numéro 13 est 1011

$$\text{d'où } (13)\ 10 = (1011)_{Gray}$$

Remarque :

On retrouve bien les propriétés du code Gray :

↳Les codes Gray de deux cases symétriques par rapport à un axe de symétrie se diffèrent par l'état d'un seul bit.

Exemple : les cases de 7 et 4

$(7)_{10} = (100)_{Gray}$

$(4)_{10} = (110)_{Gray}$

↳De même les codes Gray de deux cases successives (voisine en lignes) se diffèrent par l'état d'un seul bit.

Exemple : les cases 1 et 2

$(1)_{10} = (001)_{Gray}$

$(2)_{10} = (011)_{Gray}$

↳En plus deux cases voisines en colonnes se diffèrent en code Gray par l'état d'un seul bit.

Exemple : les cases 0 et 7

$(0)_{10} = (000)_{Gray}$ $(7)_{10} = (100)_{Gray}$

Chapitre : 2

Algèbre de Boole

Objectifs

Général

- Être capable de calculer et de simuler des fonctions logiques.

Spécifiques

- Savoir exprimer une fonction logique d'un système à logique binaire.
- Savoir les trois opérations de base de l'algèbre de Boole et leurs différentes propriétés.
- Comprendre et appliquer l'ensemble de théorèmes de l'algèbre de Boole.

Plan du chapitre

Volume horaire

4 heures et demi

1 Définition

« L'algèbre de Boole est un ensemble de variables à deux états de vérités : 1 (vrai) et 0 (faux), manipuler par un nombre limité d'opérateurs : et, ou, non. ». Il contient un ensemble de théorèmes mathématiques qui précisent les fondements théoriques de la logique binaire ou booléenne.

2 Fonction logique

2.1 Définition

C'est une expression logique (de valeur 0 ou 1) qui combine un ensemble de variables booléennes à l'aide des opérateurs logiques ou, et, non.

2.2 Présentation

Une fonction logique peut être présentée par :

2.2.1 Une table de vérité :

C’est une table qui décrit toutes les combinaisons des entrées et la valeur de la fonction (sortie) pour chaque entrée.

Exemple :

x	**y**	**z**	**F**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2.2.2 Le tableau de Karnaugh

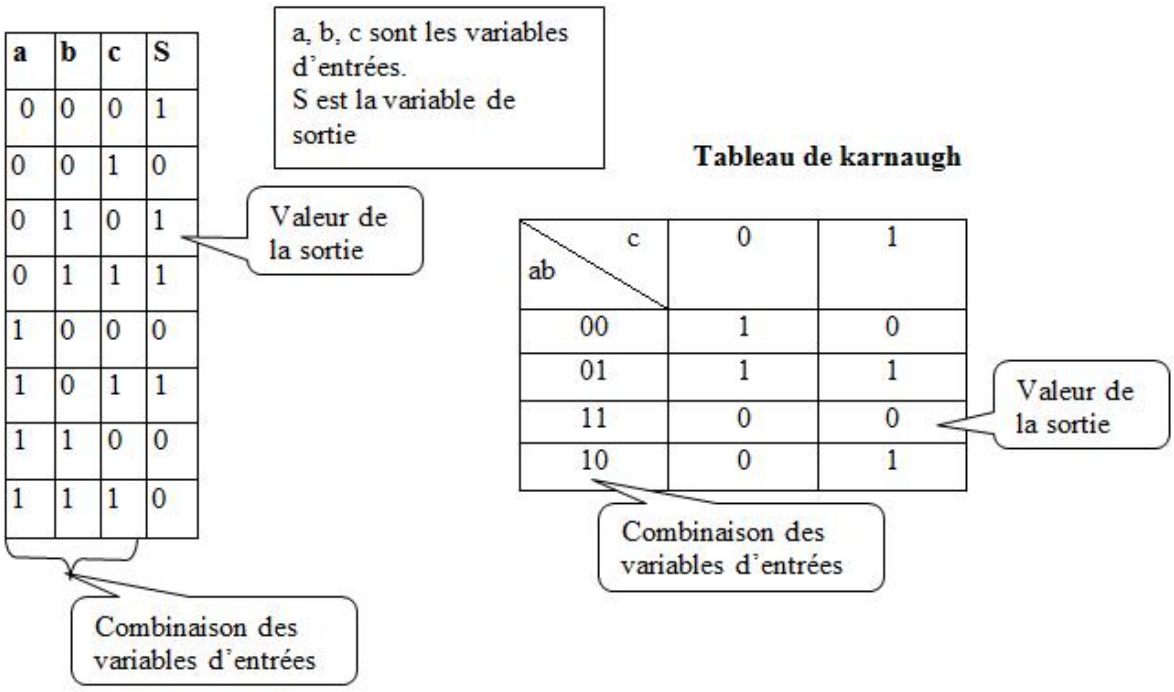
II s'agit de dresser un tableau de Karnaugh où les entrées de la fonction sont représentées par les numéros des cases et ses sorties par leur contenu.

Exemple l: le tableau de Karnaugh de la fonction précédente est :

x \ yz	**00**	**01**	**11**	**10**
0	0	0	1	0
1	1	1	1	1

Exemple2:

Table de vérité



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

ab \ c	0	1
00	1	0
01	1	1
11	0	0
10	0	1

Remarque importante :

Dans la majorité des cas la sortie d'une fonction est soit 0 ou 1. Mais dans certains cas, pour certaines fonctions, la sortie peut être *indifférente (elle peut être considérée comme un 1 ou un 0)* pour une ou plusieurs combinaison d'entrées. On la note dans ce cas par "X".

2.2.3 Formes Canoniques :

C'est une équation qui permet de localiser directement chaque case du tableau de Karnaugh comportant un « 1 » logique ou un « 0 » logique. On distingue principalement deux formes canoniques qui sont :

↳ *Première forme canonique : Somme de Produit :*

Considérant la table de vérité ou le tableau de Karnaugh de la fonction logique. A chaque 1 logique de la *variable de sortie,* on fait correspondre *le produit des n variables d'entrées.* Dans ce produit, *chaque variable* sera sous *forme normale* si elle est à 1 et sous forme *complémentée* si elle est à 0. L'expression de la fonction sera la somme des produits élémentaires ainsi formés.

Remarque: on peut dans la définition d'une fonction logique, donner seulement les combinaisons des entrées pour lesquelles la fonction sera à 1 logique.

Exemple :
F = 1 si (a, b, c) = (0,1,1) ou (1,1,1) ou (1,0,0) ou (1,0,1)
Si on note n = $(abc)_{10}$ alors F vaut 1 si et seulement si n = 3 ou 7 ou 4 ou 5.
On écrit alors f (a,b,c) = (3,4,5,7)

Application :

Exemple N° 1 : Etablir l'équation logique du système S (a,b,c) = (0,1,2,6,7).

Table de vérité :

a	**b**	**c**	**S**
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

L'équation de la fonction sous la 1ère forme canonique

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

Exemple N° 2 :

- Soit la forme canonique d'une fonction logique définie comme suit :

  f(a,b,c) = 1 si et seulement si (a,b,c) $\in$ {(1,0,1); (0,0,1) ;(1,1,1) }

Ecrire f sous forme algébrique

$f = a..\bar{b}.c + \bar{a}..\bar{b}.c + a.b.c$

↳*Deuxième forme canonique: Produit de Somme:*

Considérant la table de vérité ou le tableau de Karnaugh de la fonction logique. A chaque **0** logique de la *variable de* ***sortie,*** on fait correspondre ***la somme des n variables d'entrées.*** Dans cette somme, *chaque variable* sera sous *forme normale* si elle est à « *0* » et sous forme *complimentée* si elle est à « *1* ».

L'expression de la fonction sera le ***produit des sommes*** élémentaires ainsi formés.

Remarque: on peut dans la définition d'une fonction logique, donner seulement les combinaisons des entrées pour lesquelles la fonction sera à 0 logique.

Exemple: pour la même fonction de l'exemple 1, établir l'équation logique du système f (a,b,c) = (0,1,2,6,7).

Table de vérité :

a	**b**	**c**	**F**
0	0	0	1
0	0	1	1
0	1	0	1
0	*1*	*1*	0
1	*0*	*0*	0
1	*0*	*1*	0
1	1	*0*	1
1	1	1	1

L'équation de cette fonction peut être aussi: $F = \left(a + \bar{b} + \bar{c}\right).\left(\bar{a} + b + c\right).\left(\bar{a} + b + \bar{c}\right)$

Exemple :

F (a, b, c) = 0 SSI (a, b, c) = {6, 3, 5}

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

Exercice :

Transformer sous la première forme canonique la fonction suivante:

Soit f = a b c + a b + a c + b

F = 1 si et seulement si f (a, b, c) = { (1, 0, 1); (1, 1, x); (1, x, l); (x, l, x) }

F (a, b, c) = {(1, 0, 1); (1, 1, 0); (1, 1, 1); (0, 1, 0); (0; 1; 1)}

On peut alors déduire l'équation de f sous forme canonique.

3 Les opérations de l'algèbre de Boole

3.1 L'addition logique notée "+"

3.1.1 Définition :

L'addition logique applique de fonctionnement de l'opérateur "ou" comme suit :

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

3.1.2 Propriétés :

Les propriétés de cette opération sont :

Commutativité :

Soient x et y deux variables booléennes :
On a x + y = y + x

Démonstration de la commutativité :

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

⇨ **x + y = y + x**

Associativité :

Soient x, y et z trois variables booléennes :

On a : x + (y + z) = (x + y) + z

Démonstration de l'associativité :

x	**y**	**z**	**(z+y)**	**x+(y+z)**	**(x+y)**	**(x+y)+z**
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1

0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

⇨ x + (y + z) = (x + y) + z

↳ **L'invariance :**

Soit x une variable booléenne :

x + x = x

Démonstration de l'invariance :

x	**x**	**x+x**
0	0	0
1	1	1

3.2 L'opération de multiplication logique "ET" noté "."

3.2.1 Spécification :

Elle applique la logique de l'opérateur « ET » avec vrai =1 et faux =0

3.2.2 Propriétés

↳ **Commutativité** :
Soient x , y deux variables Booléennes x.y = yx

Démonstration :

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

⇨ x. y = y. x

↳ **L'associativité** :

Soient x, y et z trois variables Booléennes x. (y . z) = (x . y). z

Démonstration :

x	y	z	x.y	(x.y).z	y.z	(y.z).x
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

⇨ (x . y) z = x .(y . z)

↳ **L'invariance** :

Soit x une variable booléenne :

x . x = x

Démonstration :

x	x	x . x
0	0	0
1	1	1

3.3 Autres propriétés

3.3.1 La distributivité de "." sur "+"

Soient x, y et z trois variables booléennes

x.(y +z) = x.y + x.z

Démonstration :

x	y	z	y + z	x.(y + z)	x.y	x . z	x.y+x .z
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

⇨ x.(y +z) = x.y + x.z

3.3.2 Les identités remarquables

Quel que soit x variable Booléenne, on a :

- $x.\bar{x} = 0$
- $x + \bar{x} = 1$
- $x \,.\, 1 = x$
- $x + l = l$
- $x + 0 = x$
- $x \,.\, 0 = 0$

3.4 Théorèmes de l'algèbre de Boole

- *Absorption* : x + x y = x

Quel que soit x, y variables Booléennes x (l+y) = x.l = x d'où : x + x y = x

- *Allégement* ; $x + \bar{x}.y = x + y$

Quel que soit x, y variables Booléennes

x (1+ y) + $\bar{x}$ y = x + xy + $\bar{x}$ y = x + y(x + $\bar{x}$) = x + y , d'où : x + $\bar{x}$ y = x + y

- *Théorème de Morgan :*

Quel que soit x, y variables ou expressions Booléennes :

$\overline{x + y} = \bar{x} + \bar{y}$ (Transformation d'une somme en produit)

$\overline{x.y} = \bar{x} + \bar{y}$ (Transformation d'un produit en somme)

Autrement dit :

$$\overline{f(x).g(y)} = \overline{f(x)} + \overline{g(y)}$$

et $$\overline{f(x) + g(y)} = \overline{f(x)}.\overline{g(y)}$$

Exemple : S = $\overline{\overline{x.y} + \overline{y.z}}$ = xy . $\overline{\overline{y.z}}$ = xy.(y + $\bar{z}$) = xy+xy $\bar{z}$ = xy (l+ $\bar{z}$) = xy d'où S = x y

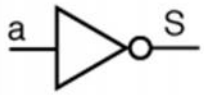
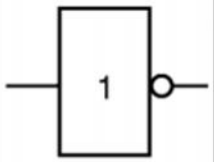
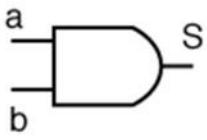
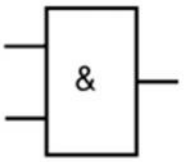
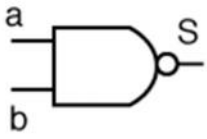
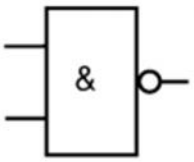
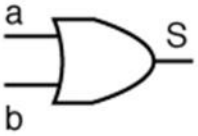
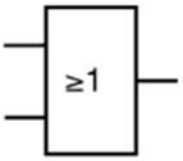
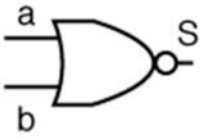
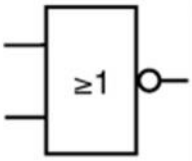
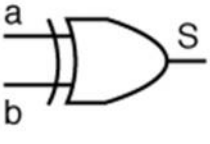
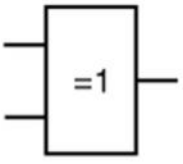
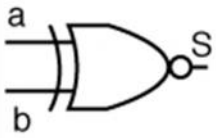
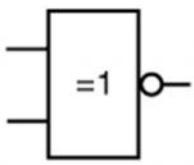
4 Les portes logiques

4.1 Définition:

Les portes logiques sont des circuits électroniques (électriques ou pneumatiques) qui appliquent les fonctions des opérateurs logiques de base Et, Ou, Non. Ceci avec l'attribution au 0 logique, une tension au voisinage de 0 v et le 1 logique une tension au voisinage de 5v.

4.2 Symboles :

Le tableau suivant présente les symboles des portes logiques standards avec leur table de vérité :

FONCTION	SYMBOLES		TABLES DE VERITE
	International	Français	
NON			a S 0 1 1 0
ET			a b S 0 0 0 0 1 0 1 0 0 1 1 1
NAND			a b S 0 0 1 0 1 1 1 0 1 1 1 0
OU			a b S 0 0 0 0 1 1 1 0 1 1 1 1
NOR			a b S 0 0 1 0 1 0 1 0 0 1 1 0
OU Exclusif			a b S 0 0 0 0 1 1 1 0 1 1 1 0
NOR Exclusif			a b S 0 0 1 0 1 0 1 0 0 1 1 1

Exercice :

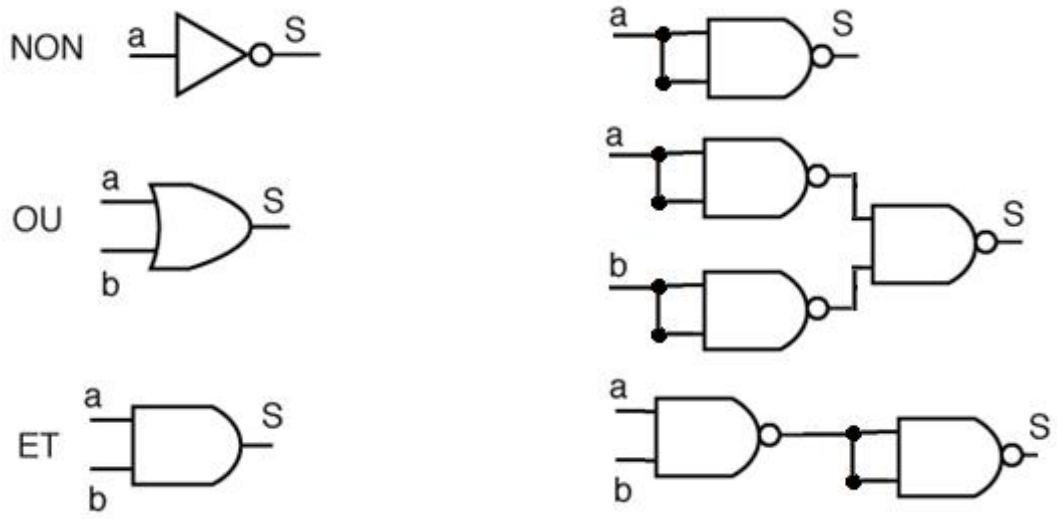
Les portes logiques NAND et NOR sont appelées universelles, car avec elles seules on peut réaliser toutes les autres portes logiques.

1- A l'aide des portes NAND uniquement réaliser les trois portes logiques de bases : NON, OU, ET

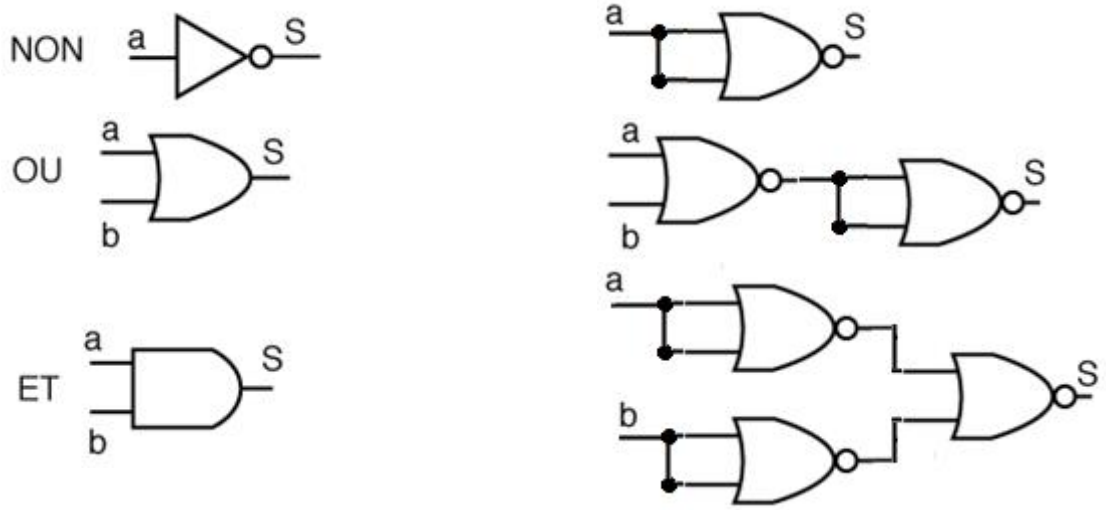
2- A l'aide des portes NOR uniquement réaliser les trois portes logiques de bases : NON, OU, ET

Solution :

1- Construction des portes NON, OU, ET à l'aide de portes NAND



2- Construction des portes NON, OU, ET à l'aide de portes NOR



Chapitre : 3

Simplification des fonctions logiques

Objectifs

Généraux

- Comprendre le pourquoi de la simplification logique
- Savoir simplifier une fonction logique.

Spécifiques

- Simplifier une fonction à l'aide des propriétés de l'algèbre de Boole
- Simplifier une fonction à l'aide du tableau de Karnaugh

Plan du chapitre

I. Problématique

II. Simplification des fonctions logiques

III. Application

Volume horaire

4 heures et demie

1 Problématique

Soit l'équation d'un circuit logique **S =** $\overline{\overline{x.y} + \overline{y}.z}$ **(1)**

A l'aide des théorèmes de l'algèbre de boule, on peut écrire

S = xy.(y + $\overline{z}$)

S = xy + xy $\overline{z}$ **(2)**

S = x y **(3)**

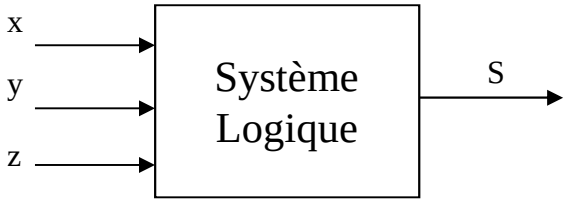


Figure 1 : Schéma générale d'un système logique

Conclusion

Le même système qui fournit une sortie S en fonction des valeurs des entrées x, y, z peut être réalisé de trois manières différentes :

(1) —> a pour coût :

- deux portes Et à deux entrées
- trois inverseurs
- une porte ou

(2) —> a pour coût :

- trois portes ET à deux entrées
- une porte ou à deux entrées
- un inverseur

(3) —> a pour coût :

- une porte ET à deux entrées

D'où la nécessité de simplifier au maximum la fonction logique d'un circuit afin de minimiser son coût.

2 Simplification des fonctions logiques

2.1 Définition :

On appelle forme minimale d'une expression logique l'expression sous forme réduite (somme de produit) qui comporte :

- Le nombre minimal de terme.
- Le nombre minimal de variable dans chaque terme.

On dispose de plusieurs outils de simplification de fonction logique dont on va citer les plus importants.

2.2 Simplification algébrique

Dans cette première méthode, on se base essentiellement sur les théorèmes de l'algèbre de Boole pour simplifier les expressions logiques.

Malheureusement, il n'est pas toujours facile de savoir quel théorème faut-il évoquer pour obtenir la simplification minimale.

Exemples : simplifier les fonctions suivantes

$F_1 = \bar{a}bc + abc + ab\bar{c} + \bar{a}b\bar{c}$

$F_2 = ab + \bar{a}b + a\bar{b}$

$F_3 = abc + ab\bar{c} + \bar{a}b\bar{c} + ab\bar{c}$

Solution :

$F_1 = \bar{a}bc + abc + ab\bar{c} + \bar{a}b\bar{c} = \bar{a}bc + ab\left(c + \bar{c}\right) + \bar{a}b\bar{c}$

$F_1 = \bar{a}b\left(c + \bar{c}\right) + ab = \bar{a}b + ab = b\left(a + \bar{a}\right) = \mathrm{b}$

$F_2 = ab + \bar{a}b + a\bar{b}$

$F_2 = b\left(a + \bar{a}\right) + a\bar{b}$

$F_2 = b + \bar{b}a$

$F_2 = b + a$ (d'après théorème d'allégement)

$F_3 = abc + ab\bar{c} + \bar{a}b\bar{c} + ab\bar{c}$

$F_3 = ab\left(c + \bar{c}\right) + b\bar{c}\left(a + \bar{a}\right)$

$F_3 = ab + b\bar{c}$

2.3 Simplification à l'aide du tableau de Karnaugh

2.3.1 Rappel: Caractéristiques du tableau de karnaugh

La caractéristique principale du tableau de karnaugh est que ses cases adjacentes horizontalement ou verticalement correspondent à des combinaisons de variables d'entrées qui se diffèrent par l'état d'une seule variable (code GRAY). De même pour des cases symétriques par rapport à un axe de symétrie vertical ou horizontal du tableau.

2.3.2 Notion de regroupement dans un tableau de Karnaugh

On peut simplifier une fonction logique représentée par un tableau de karnaugh en effectuant des regroupements de 2, 4, 8, 16, ... cases adjacentes remplies toutes avec des 1 logiques. Ceci va nous permettre de simplifier 1 ou 2 ou 4 ou plusieurs variables logiques. D'une manière générale, pour une fonction de n variables, un regroupement de 2^k cases nous donnera une équation de (n-k) variables.

2.3.3 Le processus de simplification

Les étapes de la démarche à suivre pour simplifier l'expression logique d'une fonction représentée par un tableau de Karnaugh sont les suivantes:

- Dresser le tableau de Karnaugh de la fonction et repérer les 1 adjacents
- Pointer sur une case contenant un 1 logique.
- Chercher un groupement maximal recouvrant le 1 désigné.
- L'expression du groupement est le produit des variables qui ne changent pas d'état dans les lignes formants le groupement, par les variables qui ne changent pas d'état dans les colonnes formants le groupement
- La même opération doit être faite avec toute case remplie de 1 logique non regroupé.
- S'arrêter lorsque tous les points vrais appartiennent au moins à un groupement
- Faire la somme des regroupements obtenus pour obtenir l'expression de la fonction.

Exemples :

• Regroupement de doublets

Le regroupement de deux cases adjacentes, verticalement ou horizontalement, ou symétriques remplies des 1 logiques simplifie une variable dans l'expression de la fonction.

ab \ c	0	1
00	1	0
01	1	1
11	0	0
10	0	0

L'expression canonique de cette fonction est : $F = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}bc$

L'expression réduite (simplifiée de la fonction): $F = \bar{a}\bar{c} + \bar{a}b$

• Regroupement de quartets

Un groupement de 4 cases adjacentes ou symétriques remplies des 1 logiques va simplifier 2 variables dans l'expression canonique de la fonction logique.

Exemple 1:

c \ ab	00	01	11	10
0	**0**	**0**	**0**	**0**
1	**1**	**1**	**1**	**1**

La forme canonique de F est :
$F = \bar{a}\bar{b}c + \bar{a}bc + abc + a\bar{b}c$

L'expression de F simplifiée est : $F = c$.

Exemple2 :

ab \ cd	**0 0**	**0 1**	**1 1**	**1 0**
0 0	0	0	0	0
0 1	1	0	0	1
1 1	1	0	0	1
1 0	0	0	0	0

L'expression canonique de F est :

$F = \bar{a}bc\bar{d} + abc\bar{d} + \bar{a}bcd + abcd$

L'expression de F simplifiée est :

$F = b\bar{d}$

- **Regroupement d'octets**

Un groupement de 8 cases adjacentes ou symétriques remplies des 1 logiques va simplifier 3 variables logiques dans l'expression canonique de la fonction logique.

Exemple :

ab \ cd	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

L'expression canonique de F est :

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

L'expression de f simplifiée est :

$F = \bar{d}$

Exercice:

Donner l'expression simplifiée de la fonction F représentée par son tableau de Karnaugh suivant:

ab \ cd	00	01	11	10
00	0	0	0	1
01	0	1	1	0
11	0	1	1	0
10	0	0	1	0

L'équation simplifiée est :

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

3 Application

3.1 Énoncé :

Trois interrupteurs I_1, I_2 et I_3 commandent le démarrage de deux moteurs M_1 et M_2 selon les conditions suivantes (lorsqu'un interrupteur est fermé, $I_i = 1$) **:**

- Le moteur M_1 ne doit démarrer *que si au moins deux interrupteurs* sont fermés
- Le moteur M_2 démarre *dès qu'un ou plusieurs interrupteurs sont activés.*

1. Donner la table de vérité régissante le fonctionnement du système.
2. Simplifier les expressions logiques des sorties en utilisant la méthode graphique basée sur la notion du tableau du Karnaugh.
3. Réaliser le logigramme adéquat en utilisant quelques portes logiques

3.2 Correction :

- Table de vérité :

I_3	I_2	I_1	M_2	M_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

- Tableaux de karnaugh :

I_3 \ I_2I_1	00	01	11	10
0	0	1	1	1
1	1	1	1	1

$M_2 = I_3 + I_2 + I_1$

I_3 \ I_2I_1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$M_1 = I_3 I_2 + I_2 I_1 + I_3 I_1$

- Logigramme des sorties :

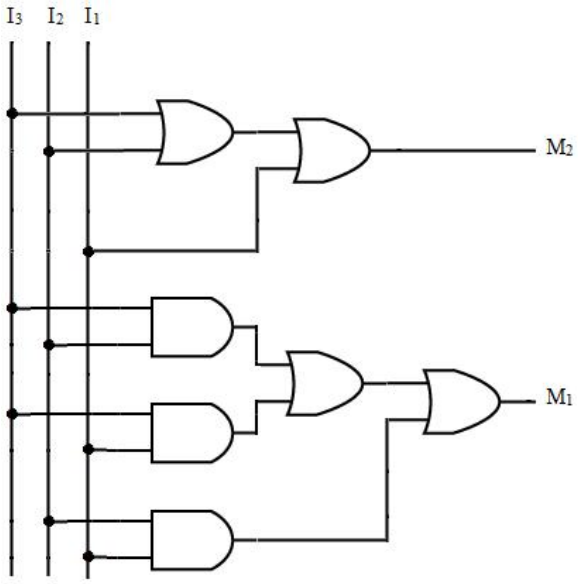


Figure 2 : logigramme des sorties des moteurs

Chapitre : 4

Les circuits combinatoires standards

Objectifs

Généraux

- Comprendre et maitriser les circuits combinatoires standards.

Spécifiques

- Connaitre les circuits de codages.
- Connaitre les circuits d'aiguillages.

Plan du chapitre

Volume horaire

4 heures et demi

1 Introduction

Les circuits de transformation des codes ou de codage font la transposition des données d'un code à un autre. Ils jouent le rôle d'interprète entre l'homme et la machine (codeur) entre la machine et l'homme (décodeur) entre machine et machine (transcodeur).

2 Les circuits de codage

2.1 Le décodeur

2.1.1 Description

Un décodeur est un circuit logique qui établit la correspondance entre un code d'entrée binaire de n bits et m lignes de sortie ($m \leq 2^n$). Pour chacune des combinaisons possibles des entrées une seule ligne de sortie est validée.

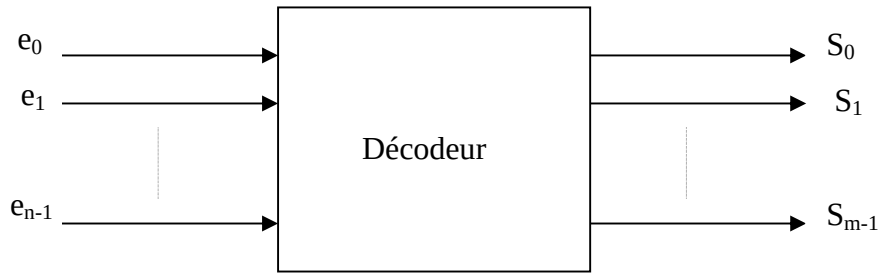


Figure 3 : Schéma générale d'un décodeur

Pour une combinaison binaire de n entrées => une seule ligne sera mise à 1

Remarque :

Certains décodeurs n'utilisent pas toute la gamme de 2^n codes d'entrée possible mais seulement un sous-ensemble de ceux-ci. Ils sont alors souvent conçus de façon à ce que les codes inutilisés n'activent aucune sortie lorsqu'ils se présentent à l'entrée du décodeur (décodeur BCD-décimal possède 4 bits d'entrées et 10 sorties).

2.1.2 Exemples d'application :

a) Décodeur 1 parmi 8 :

C'est un circuit combinatoire à trois entrées et $2^3 = 8$ sorties

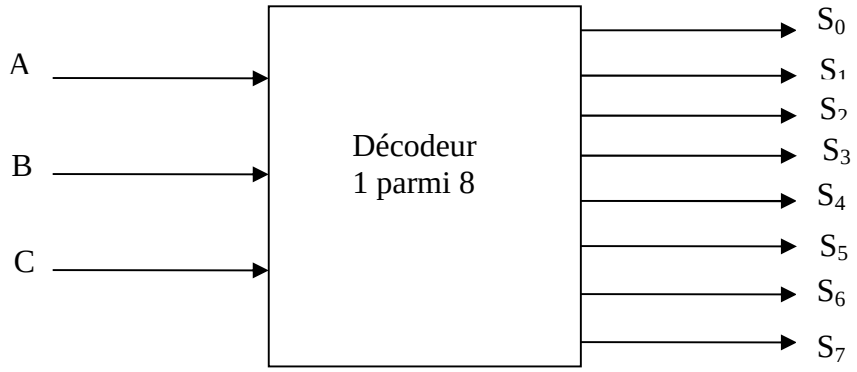


Figure 4 : Décodeur 1 parmi 8

La sortie activée est celle qui porte le rang de la valeur des entrées (A est la valeur de plus fort poids).

Question :

Etablir la table de vérité du circuit.

Donner l'équation simplifiée de chaque sortie et établir le logigramme du circuit.

A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$S_0 = \overline{A}\overline{B}\overline{C}$; $S_1 = \overline{A}\overline{B}C$; $S_2 = \overline{A}B\overline{C}$; $S_3 = \overline{A}BC$; $S_4 = A\overline{B}\overline{C}$; $S_5 = A\overline{B}C$; $S_6 = AB\overline{C}$; $S_7 = ABC$

Le logigramme des sorties du décodeur 1 parmi 8 :

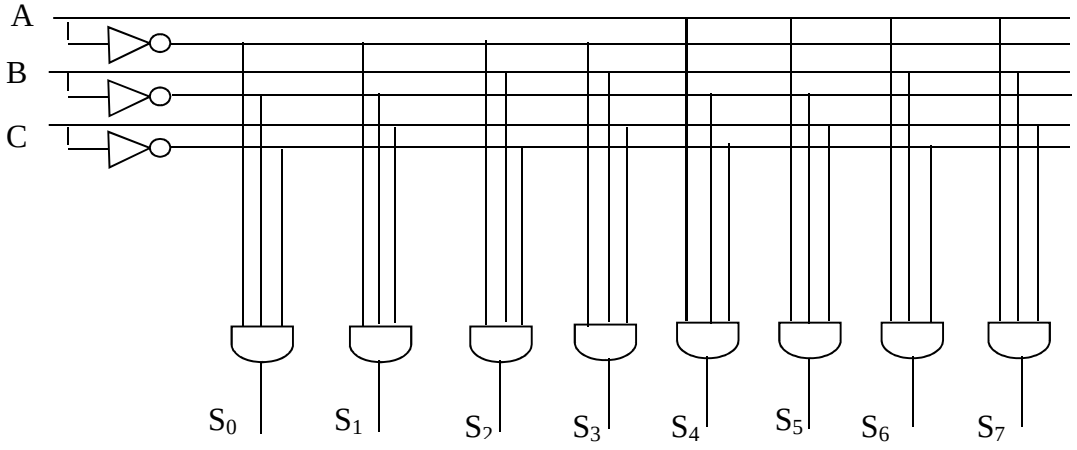


Figure 5 : Logigramme d'un décodeur 1 parmi 8

b) Décodeur DCB - Décimal (1 parmi 10)

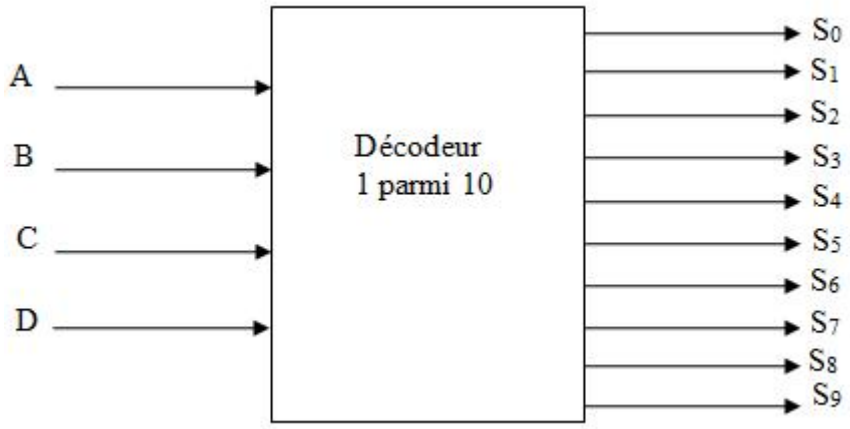


Figure 6 : Décodeur 1 parmi 10

(Si) est activée si la valeur i est présente en binaire en entrée et elle est valide.

Si i est non valide => aucune sortie n'est activée.

Question :

Dressez la table de vérité de ce circuit et déduire les équations logiques simplifiées des différentes sorties.

Solution :

A	B	C	D	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0

Les équations simplifiées des sorties de ce circuit sont :

$S_0 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ $S_1 = \overline{A}\,\overline{B}\,\overline{C}D$ $S_2 = \overline{A}\,\overline{B}C\,\overline{D}$ $S_3 = \overline{A}\,\overline{B}CD$ $S_4 = \overline{A}B\,\overline{C}\,\overline{D}$

$S_5 = \overline{A}BC\overline{D}$ $S_6 = \overline{A}BC\overline{D}$ $S_7 = \overline{A}BCD$ $S_8 = A\overline{B}\overline{C}\overline{D}$ $S_9 = A\overline{B}\overline{C}D$

c) Réalisation de fonctions logiques :

A l'aide d'un décodeur approprié et des portes logiques, réaliser la fonction logique suivante :

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

Solution:

D'après l'équation de la fonction F, nous aurons besoin d'un décodeur à trois entrées (A, B et C) ; donc on utilisera un décodeur 1 parmi 8.

Par identification avec les sorties d'un tel décodeur, on peut conclure que :

$F = S_2 + S_0 + S_6 + S_7$

Finalement, on obtient le logigramme de la fonction F suivant :

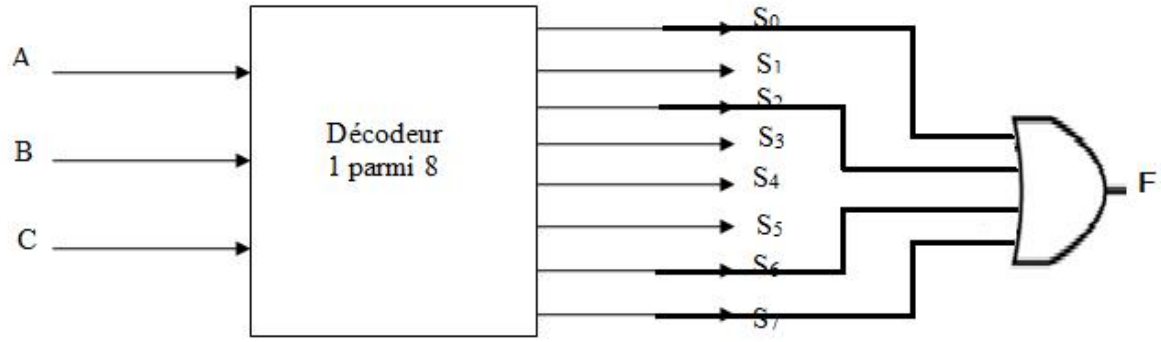


Figure 7: logigramme de la fonction F

2.2 Le codeur :

2.2.1 Description :

C'est un dispositif qui effectue l'opération inverse du décodeur: Une seule entrée parmi **m** est activée à la fois, ce qui correspond à un nombre binaire en sortie. On l'appelle aussi encodeur.

2.2.2 Exemples d'application :

a) Codeur 4 vers 2 :

C'est un codeur qui possède 4 entrées et deux sorties. Pour chaque entrée activée, son code binaire est affichée sur les sorties.

Question :

Dressez la table de vérité de ce circuit et déduire les équations logiques simplifiées des différentes sorties.

Solution :

- Table de vérité :

A_3	A_2	A_1	A_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- Equations de soties :

$S_1 = A_3 + A_2$
$S_0 = A_1 + A_3$

- Logigramme :

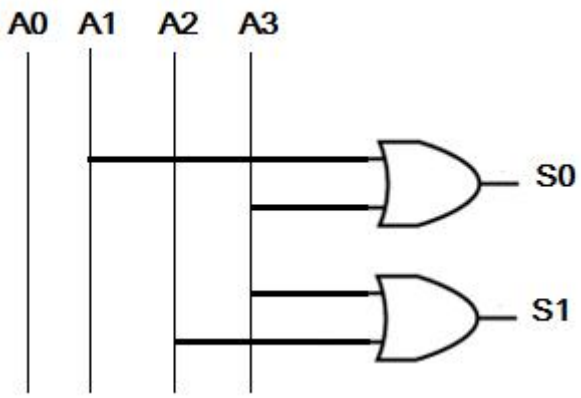


Figure 8: logigramme du codeur 4 vers 2

b) Codeur de priorité :

L'activation en simultané des entrées A_1 et A_2 du codeur de l'exercice précédent, fera passer les sorties S_1S_0 au nombre 11 qui ne correspond pas au code de l'une ou de l'autre des entrées activés. C'est plutôt le code qui représente l'activation de A_3.

Pour résoudre ce problème on utilise un codeur de priorité qui choisit le plus grand nombre lorsque plusieurs entrées sont activées à la fois. Exemple lorsque A1 et A_2 sont activées simultanément S_1S_0 sera égale à 10 ce qui représente l'activation de A_2

Exemple : Codeur de priorité 8 vers 3

Dresser la table de vérité du codeur :

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	S_2	S_1	S_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

2.3 Le transcodeur :

2.3.1 Description :

C'est un circuit combinatoire qui se diffère du décodeur par le fait que plusieurs de ses sorties peuvent être actives simultanément. Alors que pour un décodeur une seule des sorties peut être activée à la fois.

Le transcodeur est appelé aussi convertisseur de codes. En effet, il permet de passer d'un code en entrée E de n bits à un code en sortie S de m bits.

2.3.2 Exemple d'application :

Réaliser un transcodeur binaire/Gray à trois bits :

- Dresser sa table de vérité
- Donner les équations de ses sorties

Solution :

A	**B**	**C**	**X**	**Y**	**Z**
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

$X = A$; $Y = A \oplus B$; $Z = B \oplus C$

3 Les circuits d'aiguillage :

3.1 Le multiplexeur

3.1.1 Description :

Un multiplexeur est un circuit qui a pour rôle de faire circuler sur une seule voie les informations provenant de plusieurs sources.

D'une façon générale, un multiplexeur possède **n** entrées de commandes (d'adresses ou de sélection) qui permettent de sélectionner l'une des 2^n entrées de données possibles et de l'envoyer vers l'unique sortie.

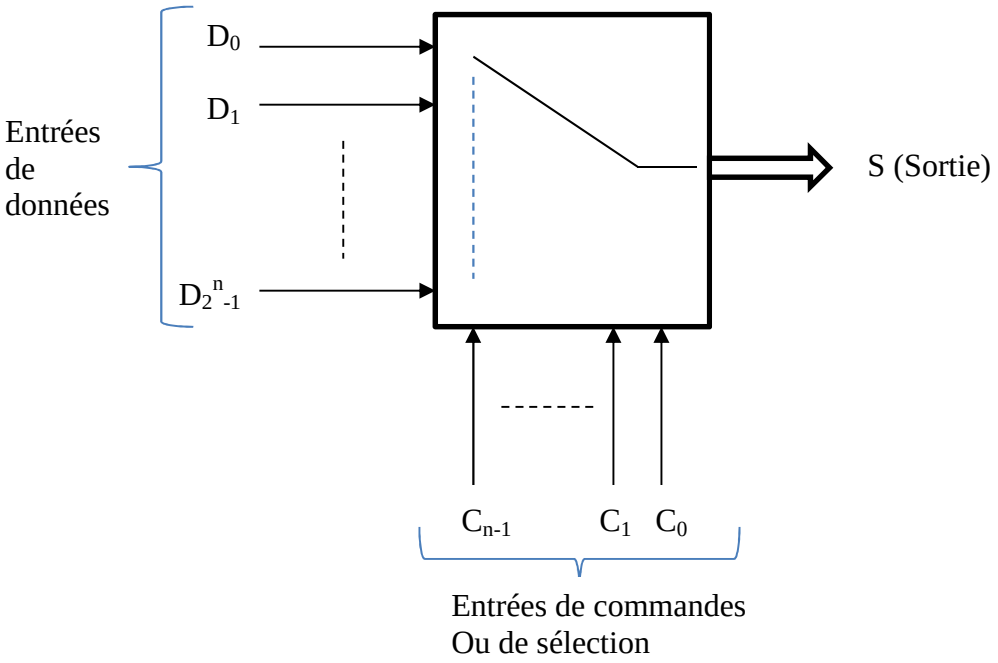


Figure 9 : Schéma générale d'un multiplexeur

Remarque :

- *Les* entrées de données peuvent avoir une taille m >= 1

- La sortie S du multiplexeur doit avoir la même taille que les données.

3.1.2 Exemples d'application

Exemple 1 :

1°/ Dresser la table de vérité d'un multiplexeur à 3 entrées de sélections et des entrées de données sur 1bit.

2°/ Etablir l'équation simplifiée de la sortie Z.

3°/ Etablir le logigramme de la sortie avec des portes logiques de votre choix.

Solution :

Un multiplexeur à trois entrées de commandes (I_2, I_1 et I_0) possède 8 entrées de données (D_0…D_7)

- Table de vérité simplifiée :

I_2	I_1	I_0	Z
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

- Equation de la sortie :

$$Z = \overline{I_2}\,\overline{I_1}\,\overline{I_0}D_0 + \overline{I_2}\,\overline{I_1}I_0D_1 + \overline{I_2}I_1\overline{I_0}D_2 + \overline{I_2}I_1I_0D_3 + I_2\overline{I_1}\,\overline{I_0}D_4 + I_2\overline{I_1}I_0D_5 + I_2I_1\overline{I_0}D_6 + I_2I_1I_0D_7$$

- Logigramme de la sortie :

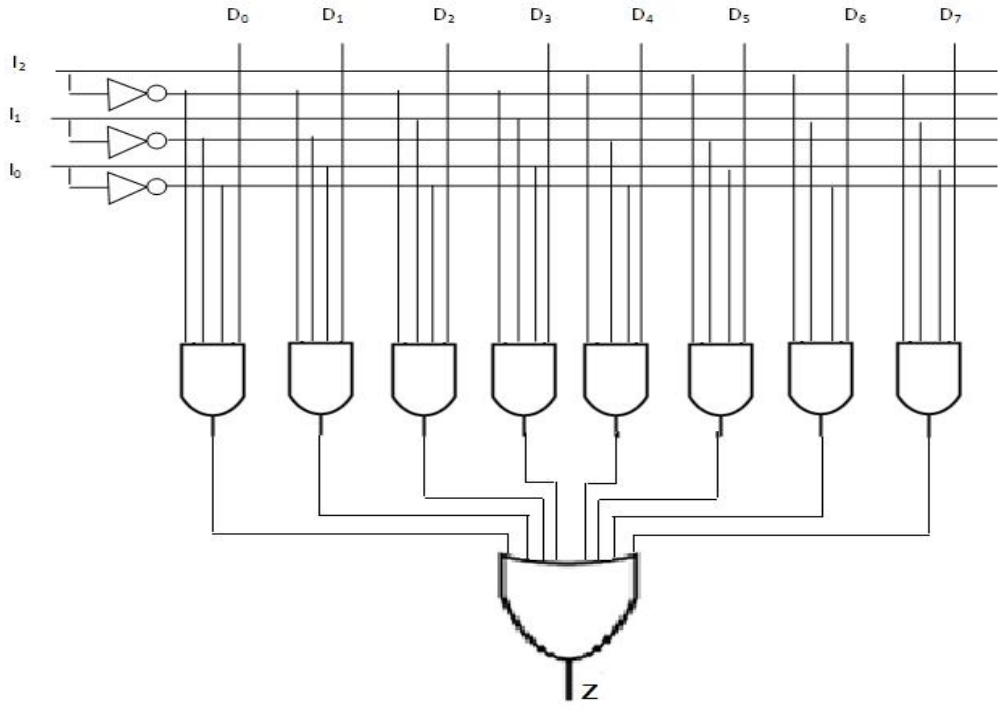


Figure 10 : Logigramme de la sortie du multiplexeur à 3 entrées d'adresses

Exemple 2 :

On donne la fonction suivante F (a, b,c) = $\bar{a}bc + a\bar{c} + abc$.

Réaliser cette fonction à l'aide d'un seul multiplexeur à 3 entrées de sélection.

Solution :

F est la sortie du multiplexeur et a, b et c sont les entrées de sélection. D'où la forme générale de F :

F = $\bar{a}\bar{b}\bar{c}\,D_0 + \bar{a}\bar{b}c\,D_1 + \bar{a}b\bar{c}\,D_2 + \bar{a}bc\,D_3 + a\bar{b}\bar{c}\,D_4 + a\bar{b}c\,D_5 + ab\bar{c}\,D_6 + abc\,D_7$ **(1)**

Pour avoir F (a, b,c) = $\bar{a}bc + a\bar{c} + abc$

Il faut tout d'abord l'écrire sous sa forme canonique:

F (a, b,c) = $\bar{a}bc + a\bar{c}(b + \bar{b}) + abc = \bar{a}bc + ab\bar{c} + a\bar{b}\bar{c} + abc$ **(2)**

Par identification entres les équations (1) et (2) de F, on conclut que :

D_3, D_4, D_6, D_7 doivent être à 1 et D_0, D_1, D_2, D_5 doivent être à 0.

Finalement, on obtient la solution suivante :

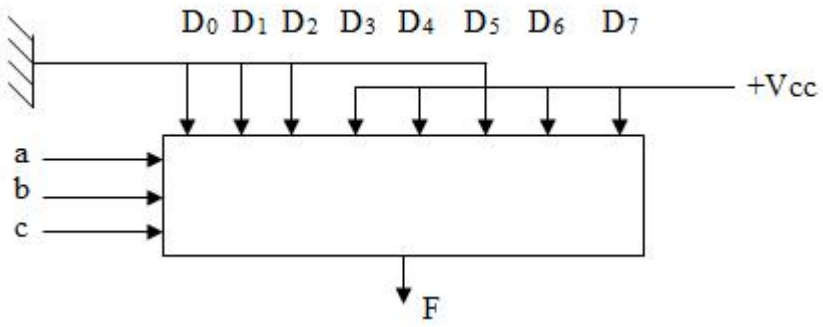


Figure 11 : Réalisation de la fonction F

3.2 Le démultiplexeur

3.2.1 Description

Le démultiplexeur dispose de n entrées de sélection, d'une entrée de données et 2^n sorties.

La donnée sera aiguillée vers la sortie dont le rang est composé par les entrées de sélections.

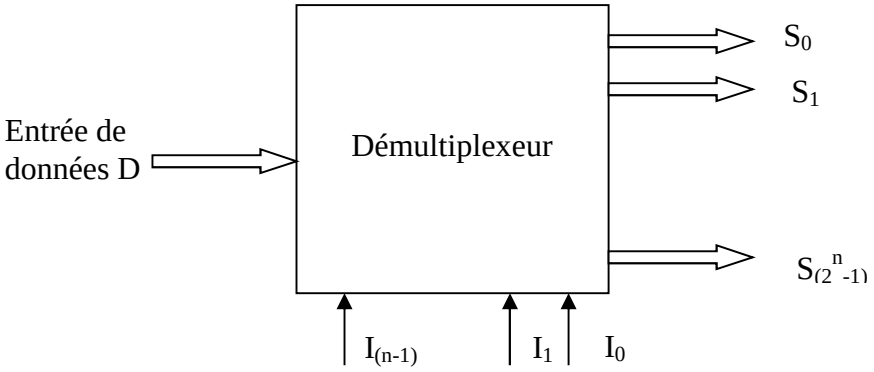


Figure 12 : Schéma générale d'un démultiplexeur

S_i = D si et seulement si $(I_{n-i}...I_o)$ = i

3.2.2 Exemples d'application :

Dresser la table de vérité d'un démultiplexeur à deux entrées de sélection

Etablir les équations de ses sorties et construire son logigramme

Solution:

Un démultiplexeur à deux entrées de sélection possède quatre sorties

- Table de vérité:

I_1	I_0	S_3	S_2	S_1	S_0
0	0	0	0	0	D
0	1	0	0	D	0
1	0	0	D	0	0
1	1	D	0	0	0

- Equations des sorties :

$$S_0 = \overline{I_1}\,\overline{I_0}D$$

$$S_1 = \overline{I_1} I_0 D$$

$$S_2 = I_1 \overline{I_0} D$$

$$S_3 = I_1 I_0 D$$

- Logigramme :

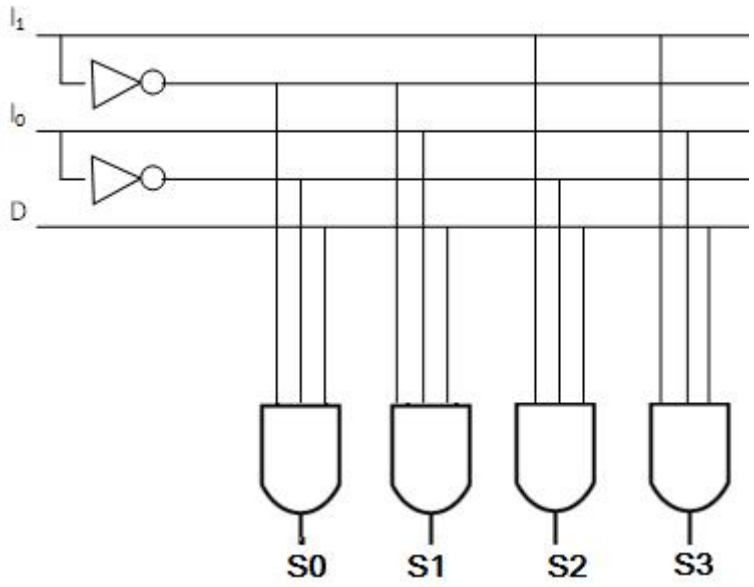


Figure 13 : Logigramme du démultiplexeur

Chapitre : 5

Les circuits arithmétiques

Objectifs

Généraux

- Assimiler et manipuler les circuits arithmétiques.

Spécifiques

- Construire des circuits d'addition de deux nombres binaires.
- Construire des circuits de soustraction de deux nombres binaires.
- Construire des circuits de comparaison de deux nombres binaires.

Plan du chapitre

I. Les additionneurs

II. Les soustracteurs

III. Les comparateurs

Volume horaire

4 heures et demie

1 Objectif:

Dans ce chapitre, nous allons réaliser des circuits combinatoires qui permettent d'établir les opérations d'addition, de soustraction et de comparaison de deux nombres binaires.

2 L'additionneur

Un additionneur est un circuit combinatoire qui présente la structure suivante :

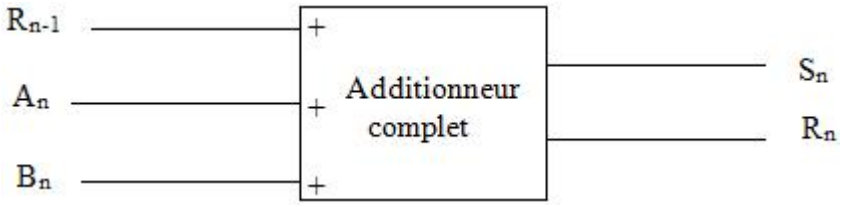


Figure 14 : Schéma d'un additionneur complet

Où:

➔ A_n et B_n sont les deux bits du rang *n* à additionner

➔ R_{n-1} est une retenue de l'étage précédent qui doit être prise en considération dans l'addition.

➔S_n est le résultat de l'opération d'addition du rang *n*

➔R_n est la retenue provoquée par l'addition et renvoyée vers l'étage suivant

2.1 Rappel

Les opérations d'additions de base sont :

0 + 0 = 0

0 + 1 = 1

1 + 1 = 0 avec une retenue = 1

1 + 1 + 1 = 1 avec une retenue = 1

Retenu précédent

Application 1:

Etablir la table de vérité et le tableau de karnaugh d'un additionneur complet 1 bit (AC : élémentaire). Donner le logigramme de cet additionneur à l'aide des portes logiques de votre choix.

Solution :

- La 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

- Tableau de Karnaugh de S_n :

R_{n-1} \ A_nB_n	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S_n = \overline{A_n}B_n\overline{R_{n-1}} + A_n\overline{B_n}\overline{R_{n-1}} + \overline{A_n}\overline{B_n}R_{n-1} + A_nB_nR_{n-1}$$

$$= (A_n\overline{B_n} + \overline{A_n}B)\,\overline{R_{n-1}} + (\overline{A_n}\overline{B_n} + A_nB_n)\,R_{n-1}$$

$$= (A_n \oplus B_n)\,R_{n-1} + (A_n \oplus B_n)\,R_{n-1}$$

$$S_n = (A_n \oplus B_n)\,R_{n-1} + (A_n \oplus B_n)\,R_{n-1}$$

$$S_n = (A_n \oplus B_n) \oplus R_{n-1}$$

- Tableau de Karnaugh de R_n :

R_{n-1} \ A_nB_n	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$R_n = A_n R_{n-1} + B_n R_{n-1} + A_n B_n$

$R_n = R_{n-1} (A_n + B_n) + A_n B_n$

- Logigramme :

Pour minimiser le nombre de portes logiques, nous allons écrire R_n sous la forme :

$$R_n = R_{n-1}(A_n \oplus B_n) + A_n B_n$$

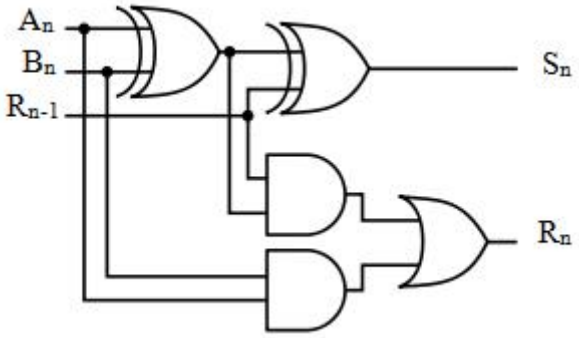


Figure 15 : logigramme additionneur complet 1 bit

Application 2:

A l'aide de l'additionneur complet 1 bit de l'exercice précédent, réaliser un additionneur complet 2 bits (permet de faire l'addition de deux nombres chacun composé de deux bits).

Solution :

Pour réaliser cet additionneur, on aura besoin de deux additionneurs 1 bit ; l'un pour l'addition du rang 0 et l'autre pour l'addition du rang 1 et la retenue du rang 0

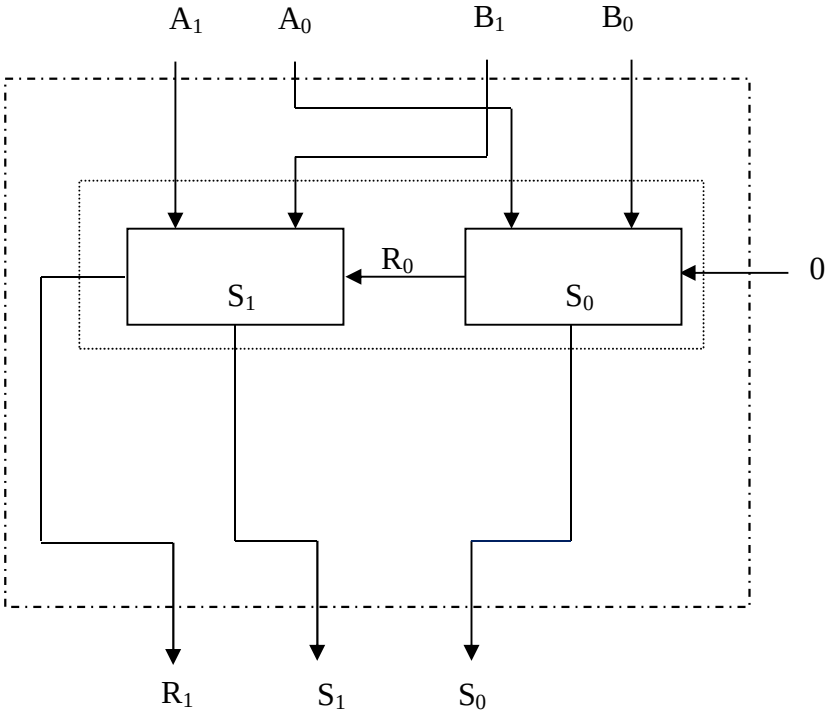


Figure 16: additionneur 2 bits

3 Le soustracteur

C'est un circuit combinatoire qui réalise la soustraction de deux nombres binaires. Il présente la structure suivante :

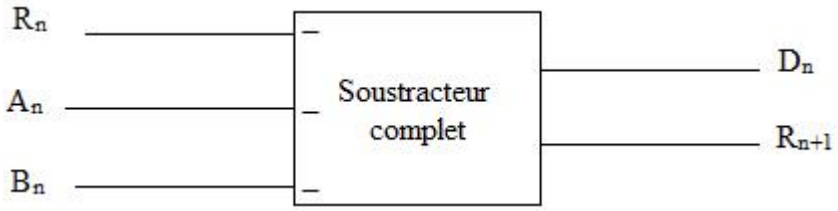


Figure 17: Schéma de principe d'un soustracteur complet

Où:

➔ A_n et B_n sont les deux bits du rang *n* à soustraire

➔ R_n est une retenue engendrée de l'étage précédent qui doit être prise en considération dans la soustraction.

➔D_n est le résultat de l'opération de soustraction du rang *n*

➔R_{n+1} est la retenue renvoyée vers l'étage suivant

Rappel :

Les opérations de soustraction de base sont :

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = 1 avec une retenue = 1

Exercice:

Etablir la table de vérité et le tableau de karnaugh d'un soustracteur 1 bit.

Solution:

- Table de vérité

R_n	A_n	B_n	D_n	R_{n+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Pour remplir la colonne de la sortie D_n, pour chaque ligne de la table de vérité il faut appliquer l'équation suivante :

$D_n = A_n - (B_n + R_n)$

Si l'opération est impossible et qu'il faut emprunter 1 pour la réaliser ($A_n < (B_n + R_n)$), alors R_{n+1} prend systématiquement 1.

- Tableau de Karnaugh de D_n:

R_n \ $A_n B_n$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$D_n = \overline{A_n} B_n \overline{R_{n-1}} + A_n \overline{B_n} \overline{R_{n-1}} + \overline{A_n} \overline{B_n} R_{n-1} + A_n B_n R_{n-1}$$

$$= (A_n \overline{B_n} + \overline{A_n} B) \overline{R_{n-1}} + (\overline{A_n} \overline{B_n} + A_n B_n) R_{n-1}$$

$$= (A_n \oplus B_n) R_{n-1} + (A_n \oplus Bn) R_{n-1}$$

$$D_n = (A_n \oplus B_n) R_{n-1} + (A_n \oplus B_n) R_{n-1}$$

$$= (A_n \oplus B_n) \oplus R_{n-1}$$

- Tableau de Karnaugh de R_n:

R_n \ $A_n B_n$	00	01	11	10
0	0	0	0	1
1	1	0	1	1

$$R_{n+1} = \overline{A_n} B_n + \overline{A_n} R_n + B_n R_n$$

$$R_{n+1} = R_n \left(\overline{A_n} + B_n \right) + \overline{A_n} B_n$$

- Logigramme :

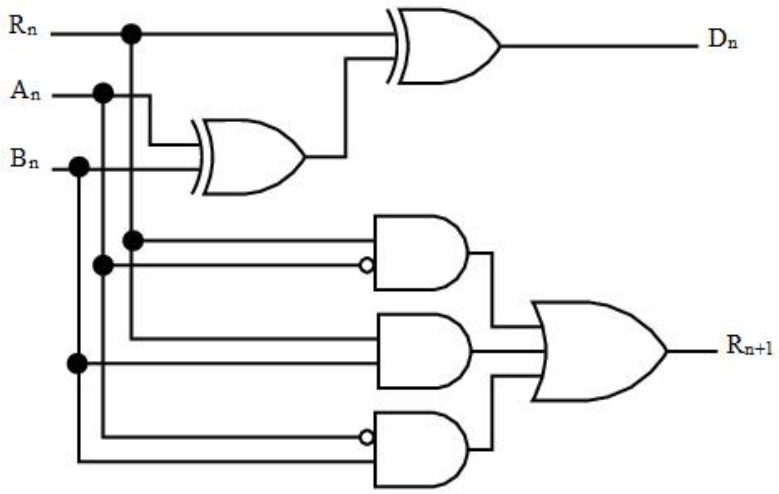


Figure 18: logigramme soustracteur 1 bit

4 Les comparateurs

Ce sont des circuits combinatoires standards qui servent pour la comparaison de deux nombres binaires.

4.1 Principe de la comparaison

Soit $A = A_n A_{n-1} A_{n-2} ..... A_1 A_0$

$B = B_n B_{n-1} B_{n-2} ..... B_1 B_0$

Deux nombres binaires à comparer. Le processus commence par la comparaison des bits de poids fort:

Si $A_n > B_n$ alors $A > B$

Si $A_n < B_n$ alors $A < B$

Si An = Bn alors il faut passer à la comparaison de A_{n-1} et B_{n-1}

On a alors besoin d'un comparateur élémentaire 2 bits

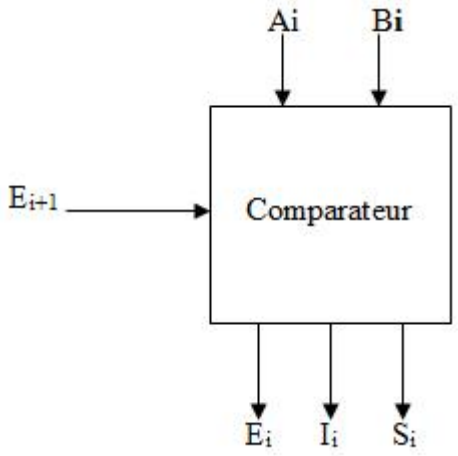


Figure 19 : Schéma d'un comparateur

➔ Si $E_{i+1} = 1$ alors la comparaison de A_i et B_i se fait normalement.

➔ Si $E_{i+1} = 0$ alors toutes les sorties seront à zéro, le résultat de la comparaison est déjà donné par la comparaison des bits précédents.

➔ Si $A_i > B_i$ alors $E_i = I_i = 0$ et $S_i = 1$.

➔ Si $A_i < B_i$ alors $E_i = S_i = 0$ et $I_i = 1$.

➔ Si $A_i = B_i$ alors $E_i = 1$ et $I_i = S_i = 0$.

Exercice :

Etablir la table de vérité de ce comparateur et donner les équations de ses sorties avec leurs câblages.

Solution :

L'entrée E_{i+1} joue le rôle d'une entrée de validation pour le circuit. Si elle est égale à 0 le circuit reste bloqué.

E_{i+1}	A_i	B_i	S_i	I_i	E_i
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	0	1

les équations des sorties :

$$E_i = E_{i+1}\overline{A_i}\,\overline{B_i} + E_{i+1}A_iB_i$$

$$= E_{i+1}(\overline{A_i}\,\overline{B} + A_iB_i) = E_{i+1}\,(A_i \otimes B_i)$$

$$I_i = E_{i+1}\overline{A_i}B_i$$

$$S_i = E_{i+1}A_i\overline{B_i}$$

- Logigramme :

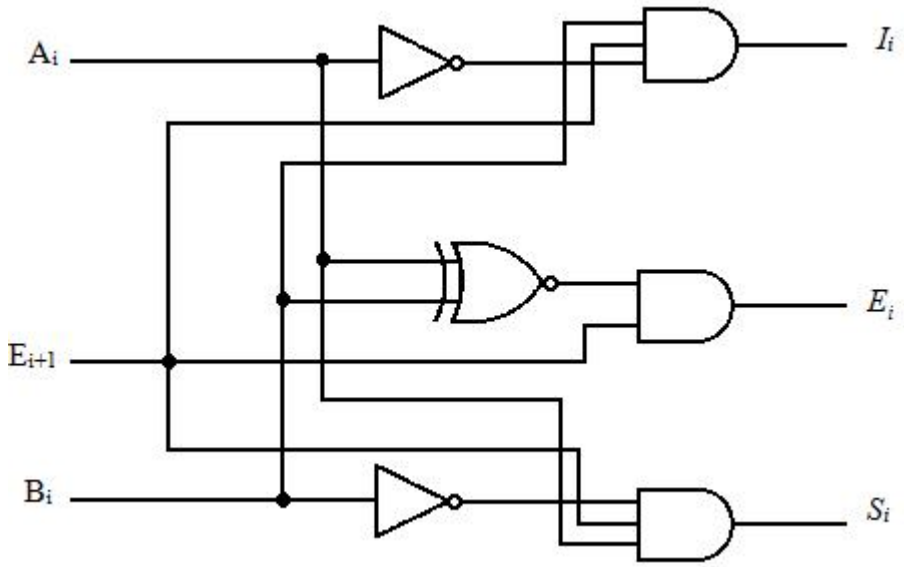


Figure 20 : Logigramme d'un comparateur élémentaire

4.2 Comparaison en cascade

Pour comparer deux nombres binaires sur plusieurs bits (supérieur ou égale à 2). On peut utiliser des comparateurs élémentaires montés en cascade.

Exercice :

Elaborer le câblage d'un comparateur de nombre binaire sur deux bits à base des comparateurs élémentaires

Solution :

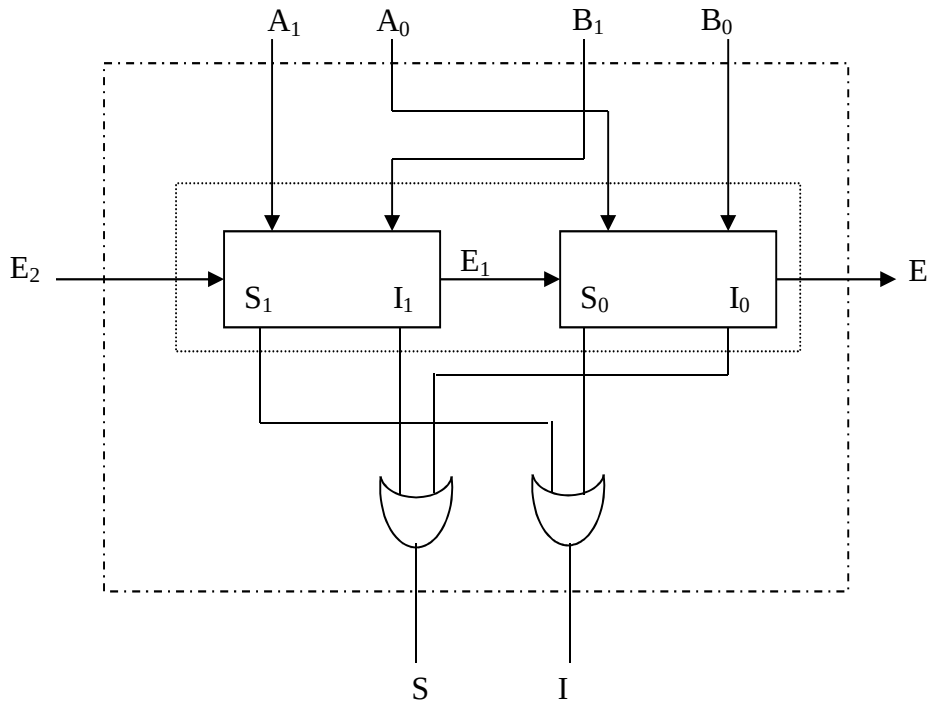


Figure 21 : Comparateur en cascade

Bibliographie

[1] Thomas L.Floyd, Systèmes Numériques, Reynold Goulet inc 9ème édition,2013

[2] Paolo Zanella, Architecture et technologie des ordinateurs, Dunod 3ème édition, 2002

Webographie

[3] http://www.electronique-et-informatique.fr, les comparateurs binaires, Daniel Robert

[4] https://sen.enst.fr, les opérateurs arithmétiques, TÉLÉCOM PARISTECH

[5] http://www.mongosukulu.com, arithmétique binaire opérations et circuits, Abdouramani Dadjé

[6] http://www.gecif.net, La fonction multiplexage–démultiplexage, Jean-Christophe MICHEL