Faculté de Technologie
Département Socle Commun ST
Module : *Logique Combinatoire et Séquentielle* (LCS)

Chapitre I : Systèmes de Numération et Codage.

Résumé

I. Introduction:

Un système de Numération permet de représenter un nombre N exprimé dans une base B quelconque (entière positive), écrit sous la forme :

$N = (a_n\ a_{n-1} \ldots a_1\ a_0\ a_{-1} \ldots a_{-m})_B$ qui correspond à la valeur dans la base 10: $N = \sum_{i=-m}^{n} a_i B^i$

Où : a_i : est un symbole représentant un chiffre entier de rang i, compris entre 0 et (B-1). Le poids d'un chiffre dépend de sa position dans le nombre.

Tel que : Si i=0 le chiffre correspondant est de poids le plus faible (LSB).
Si i=n le chiffre correspondant est de poids le plus fort (MSB).

Quelques bases des systèmes de numération sont représentées par le tableau suivant :

Système	Binaire	Octal	Décimal	Hexadécimal
Base	2	8	10	16
Chiffres	0,1	0,1,2,3,4,5,6,7	0,1,2,3,4,5,6,7,8,9	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

II. Système Binaire :

Le passage du système décimal au binaire s'appel le codage, l'inverse s'appel le décodage.

I.1. Conversion Binaire-Décimal :

$$(N)_{10} = a_n.2^n + a_{n-1}.2^{n-1} + ... + a_0.2^0 + a_{-1}.2^{-1} + a_{-2}.2^{-2} + ... + a_{-m}.2^{-m}$$

I.2. Conversion Décimal-Binaire :

1. **Conversion de la partie entière d'un nombre :**

a. Méthode des soustractions successives (petites valeurs):

- Déterminer les valeurs successives des poids 2^i (i = 0, ……,n);
- On cherche le plus grand multiple dans le nombre à convertir;
- On le retranche du nombre;
- On recommence avec le reste obtenu jusqu'à ce que l'on obtient un reste=0.

b. Méthode des divisions successives (grandes valeurs):

- Diviser le nombre décimal à convertir, successivement par "2";
- Conserver le reste jusqu'à ce que le résultat de la division soit égal à "0";
- Le nombre binaire sera la succession des restes, en commençant par le dernier.

2. **Conversion de la partie fractionnaire :**

On procède par des multiplications successives par "2" de la partie fractionnaire jusqu'à l'obtention d'un nombre entier et à chaque multiplication, on prend seulement la partie entière obtenue.

III. Relation entre la base 2 et les bases puissances de 2 ; $B=2^K$:

A faire des groupements de K bits en partant de la droite, puis convertir ces groupements au système de base $B = 2^K$. La conversion inverse se fait en convertissant chaque symbole à son équivalent binaire écrit sur K bits.

- Cas du système Octal $B = 8 = 2^3$; K=3 :
- Cas du système Hexadécimal : $B=16=2^4$ k=4:
- La conversion de l'hexadécimal à l'octal et inversement :

Hexadécimal $\longleftrightarrow$ Binaire $\longleftrightarrow$ Octal.

IV. Différentes représentations des nombres entiers signés :

Les systèmes numériques traitent aussi bien les nombres négatifs que ceux positifs, d'où la nécessité d'adopter une certaine convention selon les méthodes suivantes:

IV.1. Représentation signe-grandeur (Module et signe):

Consiste à ajouter un bit de signe à la représentation binaire de la valeur absolue du nombre. Ce bit a le poids le plus fort.

Si MSB = 0 ⟶ le nombre est positif.

Si MSB = 1 ⟶ le nombre est négatif.

Avec n bits, on peut représenter les nombres appartenant à la plage $-(2^{n-1}-1);+(2^{n-1}-1)$.

L'inconvénient majeur est qu'il existe deux représentations différentes pour le zéro.

IV.2. Représentation en complément restreint (complément à 1):

Le complément restreint est obtenu en complémentant chacun des bits du nombre et ainsi, la somme du nombre et de son complément sera égale à 2^n-1.

IV.3. Représentation en complément vrai (complément à 2).

C'est la représentation la plus utilisée. Il est égal au complément à 1 majoré de 1.

Avec des mots de n bits, on obtient 2^n valeurs différentes, de 0 à $2^{n-1}-1$ pour les valeurs positives, et de -1 à -2^{n-1} pour les valeurs négatives.

V. Différentes représentations des nombres réels :

- Représentation en virgule fixe.
- Représentation en virgule flottante.

1.Représentation en virgule fixe:

Un nombre fractionnaire en virgule fixe possède deux parties; Partie entière et partie fractionnaire. La virgule est toujours positionnée entre la partie entière et fractionnaire.

2. Représentation en virgule flottante :

La représentation en virgule flottante qui emploie la norme IEEE 754 est telle que:

- On considère les nombres décimaux comme : $N = (-1)^s .M.B^E$
  Avec : *s :* le signe, M: la mantisse, E: l'exposant.
- On normalise, en imposant à la mantisse d'être comprise entre 1 et 2;

La représentation en virgule flottante dans la norme IEEE 754 flottant sur 32 bits est:

b_{31}	b_{30}	b_{29}	…….	b_{24}	b_{23}	b_{22}	….....	b_1	b_0

Le bit b_{31} (signe mantisse): 1 pour une mantisse négative.

0 pour une mantisse positive.

b_{30} ……b_{23} : L'exposant (en excédent 127).

b_{22}…….b_0 : La mantisse qui vaut 1,xxx et on ne stocke que xxx sur b_{22}……b_1b_0.

VI. Les opérations arithmétiques en binaire :

VI.1. L'addition:

0+0=0 ; 1+0=1 ; 0+1=1 ; 1+1=0 avec une retenue 1.

VI.2. La soustraction :

0-0=0 ; 1-0=1 ; 0-1=1 avec un report de 1 ; 1-1=0.

VI.2.1. La soustraction en complément à 2 (c à 2) :

Elle permet de transformer une soustraction en binaire, en une addition et de pouvoir déterminer le signe du résultat. En représentant les valeurs négatives par leur c à 2 et le bit de signe étant égale à 1. La valeur absolue du résultat s'obtient en c à 2.

Le signe : Le signe est obtenu par addition de la retenue de l'addition des bits de poids le plus fort avec la somme des bits de signe. La retenue de cette somme est ignorée.

VI.2.2. La soustraction en complément à 1 (c à 1):

Les nombres négatifs sont remplacés par leurs c à 1 et précédés de 1. Le signe du résultat provient de l'addition de la retenue de la somme des MSB avec les bits de signe, puis la retenue de l'opération est additionnée aux LSB du résultat de l'addition des valeurs absolues. Si le résultat est négatif on prend le c à 1 de la somme des valeurs absolues.

VII. Le codage des nombres :

1. Les codes DCB :

Le nombre décimal est codé chiffre par chiffre et sa représentation binaire s'obtient en juxtaposant les combinaisons de quatre digits.

2. Code « excédent 3 » :

Il est formé de la même manière que le code DCB mais on ajoute systématiquement 3 à chaque chiffre.

3. Code GRAY : (Code cyclique progressif ou code réfléchi)

Dans ce code, un seul bit change de valeur lorsqu'on passe d'une combinaison à une autre (propriété d'adjacence).

4. Code Aïken :

Ce code ne permet que la représentation des chiffres décimaux de '0' à '9' par les 5 premiers et les 5 derniers chiffre du code binaire.

5. Codes 'p' parmi 'n' :

Chaque combinaison de n chiffres utilisée, contient un nombre 'p' de '1' et (n-p) de '0'. Ce sont des codes pondérés.

Décimal	DCB	Excédant 3	Aïken	Code '2' parmi '5'						GRAY
				Poids	7	4	2	1	0	
0	0000	0011	0000		1	1	0	0	0 (*)	0000
1	0001	0100	0001		0	0	0	1	1	0001
2	0010	0101	0010		0	0	1	0	1	0011
3	0011	0110	0011		0	0	1	1	0	0010
4	0100	0111	0100		0	1	0	0	1	0110
5	0101	1000	1011		0	1	0	1	0	0111
6	0110	1001	1100		0	1	1	0	0	0101
7	0111	1010	1101		1	0	0	0	1	0100
8	1000	1011	1110		1	0	0	1	0	1100
9	1001	1100	1111		1	0	1	0	0	1101
10										1111
11										1110
12										1010
13										1011
14										1001
15										1000

VIII. Les codes alphanumériques :

Un ordinateur est plus utile lorsqu'il peut traiter l'information non numérique: des lettres, des signes de ponctuations et de des caractères spéciaux. De ce fait il doit reconnaître des codes qui correspondent à des nombres, des lettres, des signes de ponctuations et des caractères spéciaux. Ces codes sont appelés alphanumériques.

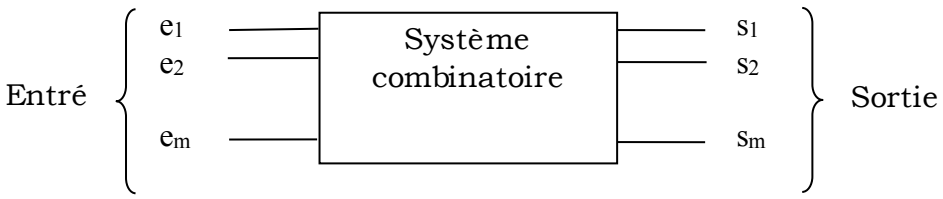
Le code alphanumérique le plus répandu est le code ASCII (American Standard Code for Information Interchange); ce code est sur 7 bits et permet de représenter 128 éléments codés. Il existe d'autre variantes du codes ASCII suivant la langue utilisée.

Chapitre II : Algèbre de Boole et logique Combinatoire

I. Algèbre de Boole et fonctions logiques :

I.1. définitions :

Logique combinatoire: Les sorties du système ne dépendent que de la combinaison des entrées. Une même combinaison d'entrée produit toujours le même effet (sortie).



$S_i = f(e_1, e_2, \ldots e_m) \quad i = 1, \ldots m$

L'Algèbre de Boole : C'est une algèbre à deux valeurs qui permet l'étude des systèmes logiques. Elle comporte :

- Les variables logiques,
- Les opérations logiques,
- Les fonctions logiques.

Une variable logique x ne peut prendre comme valeur que 0 ou 1. Elle permettra de caractériser l'état d'un élément logique.

L'élément logique : Est un système présentant 2 états stables différents. Exemples :

- Diode bloquée ou passante,
- Interrupteur ouvert ou fermé,

Une fonction logique dépend des variables logiques et sa valeur ne peut être que 0 ou 1.

I.2. les opérations logiques :

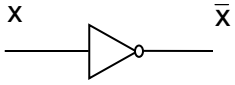
Opérations principales	Opérations auxiliaires
• NON (NOT)	• NON ET (NAND)
• ET (AND)	• NON OU (NOR)
• OU (OR)	• OU exclusif (XOR)
	• NON OU exclusif (XNOR)

a. Opération de complémentation (inversion logique NON): NON $x = \bar{x}$

La table de vérité :

x	$\bar{x}$
0	1
1	0

Symbolique :



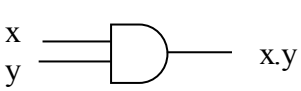
Le complément du complément d'une variable logique x est égale à la variable elle même: $\bar{\bar{x}} = x$

b. Multiplication logique (opération ET (AND)) :

La table de vérité :

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

Symbolique :

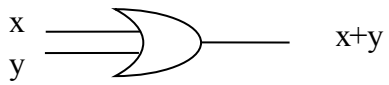


c. Addition logique (opération OU (OR)) :

Table de vérité :

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

Symbolique :

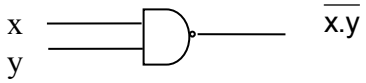


d. L'opération NON ET (NAND) :

Table de vérité :

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

Symbolique :

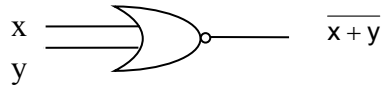


e. L'opération NON OU (NOR) :

Table de vérité :

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

Symbolique :



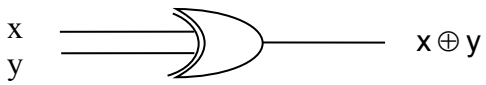
f. L'opération OU exclusif (XOR) :

$x\,\bar{y} + \bar{x}\,y = x \oplus y$

Table de vérité :

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

Symbolique :

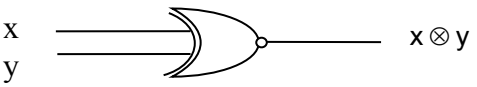


g. L'opération NON OU exclusif (XNOR) : Elle est définie par $x \otimes y = \overline{x \oplus y} = \bar{x}.\bar{y} + x.y$

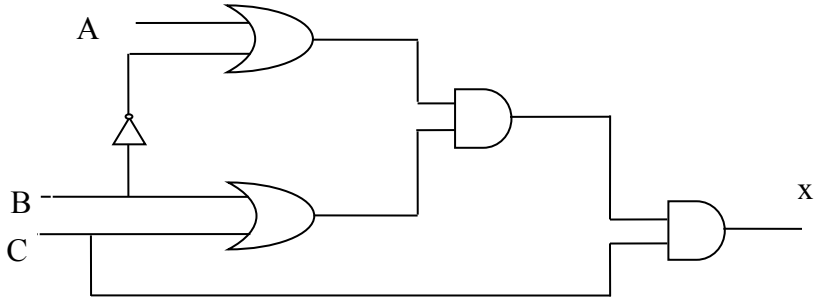
Table de vérité :

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

Symbolique :



Exemple : Donner l'expression logique de x. Simplifier cette expression.



Solution :

L'expression de x :

$$x = \left[\left(A + \overline{B}\right).\left(B + C\right)\right].B = \left[A.B + A.C + B.\overline{B} + \overline{B}.C\right].B = A.B + A.B.C = A.B\left(1 + C\right)$$

$x = A.B$

I.3. Théorème de De MORGAN :

1er théorème :

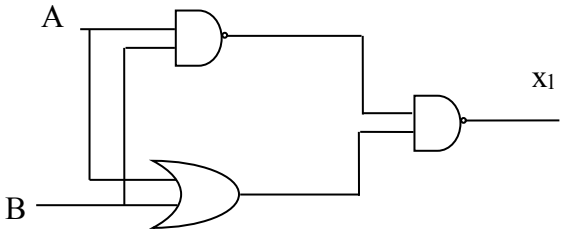
Le complément de la somme de variable logique x_i (i = 1, … n), est égal au produit des compléments $\overline{x}_i$ (i = 1, … n) de ces variables $\overline{\sum_{i=1}^{n} x_i} = \prod_{i=1}^{n} \overline{x}_i$ $\qquad \overline{x+y} = \overline{x}.\overline{y}$

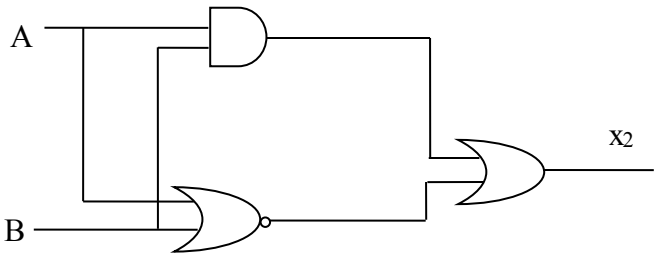
2ieme théorème :

Le complément d'un produit de variable logique x_i (i = 1, … n), est égal à la somme des compléments $\overline{x}_i$ (i = 1, … n) de ces variables $\overline{\prod_{i=1}^{n} x_i} = \sum_{i=1}^{n} \overline{x}_i$

Exemple :

En utilisant le Théorème de De MORGAN et les lois de l'algèbre de Boole, montrer que les circuits des deux figures sont équivalents :





Solution :

$$x_1 = \overline{\left(\overline{A.B}\right).\left(A + B\right)} = A.B + \left(\overline{A + B}\right)$$

$$x_2 = A.B + \left(\overline{A + B}\right)$$

$$\Rightarrow x_1 = x_2$$

I.4 Les fonctions logiques:

Une fonction logique est une expression dans laquelle les variables logiques sont liées entre elles par des opérations logiques. Elle est représentée par:

- Représentation par la table de vérité :

Pour une fonction à N variables, la table de vérité est constituée de 2^N lignes et (N+1) colonnes. La $(N+1)^{ième}$ colonne contient la valeurs de la fonction.

Exemple :

Représenter par une table de vérité la fonction: $F = A.B + A\overline{C} + \overline{B}.\overline{C} + A.\overline{B}.C$

Solution:

F est égale à 1 si l'un de ses termes est égale à 1. C'est-à-dire lorsque : A=1 et B=1 ou A=1 et C=0 ou B=0 et C=0 ou A=1 et B=0 et C=1.

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- Représentation par le tableau de KARNAUGH:

Le tableau de KARNAUGH est constitué par un nombre de cases égal au nombre de combinaisons possibles des variables intervenant dans la fonction. Le passage d'une case à une autre case symétrique ou adjacente doit entraîner le changement d'une seule variable.

- Tableau de Karnaugh à 2 variables A et B:

AB	00	01	11	10
	(0)	**(1)**	**(3)**	**(2)**

- Tableau de Karnaugh à 3 variables A, B et C:

C \ AB	00	01	11	10
0	**(0)**	**(2)**	**(6)**	**(4)**
1	**(1)**	**(3)**	**(7)**	**(5)**

- Tableau de Karnaugh à 4 variables A, B, C et D:

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

Exemple 1: Représenter sur un tableau de Karnaugh, la fonction:

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

Solution:

AB / C	00	01	11	10
0	0	0	1	1
1	0	0	1	1

Exemple 2: Représenter la table de vérité et le tableau de Karnaugh de la fonction:

$$F = B(A + C) + AC + D$$

Solution: F est à quatre variable, on aura besoin pour sa représentation d'une table de vérité de 16 lignes et 5 colonnes.

- Un tableau de Karnaugh de 16 cases.

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

AB / CD	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	1	1	1	1
10	0	1	1	1

- Les formes canoniques:
  - Première forme canonique:

On dit qu'une fonction est sous la première forme canonique, si elle est exprimé sous la forme d'une somme des combinaisons pour lesquelles elle vaut "1"; Chaque combinaison doit faire apparaître toutes les variables logiques. Chaque terme est appelé minterme.

Exemple : $F(A,B,C) = AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + ABC$

  - Deuxième forme canonique:

Ici la fonction est exprimée sous la forme d'un produit de sommes, comprenant toutes les variables ou leurs compléments. Chaque terme de cette deuxième forme est appelé maxterme.

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

Remarque: Le complément d'un minterme est un maxterme et inversement.

- Méthodes de calcul:
  - Première forme canonique:

Chaque terme de la somme est multiplié par la somme des variables manquantes et de leur complément.

  - Deuxième forme canonique:

On détermine d'abord l'expression de $\overline{F}$ sous sa 1ième forme canonique, ensuite on calcul son complément.

Exemple: Mettre sous la 1ère forme et la 2nd forme canonique la fonction suivante:

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

1ère forme: $F = A.C(B+\overline{B}) + B.\overline{C}(A+\overline{A}) = A.B.C + A.\overline{B}.C + A.B.\overline{C} + \overline{A}.B.\overline{C}$

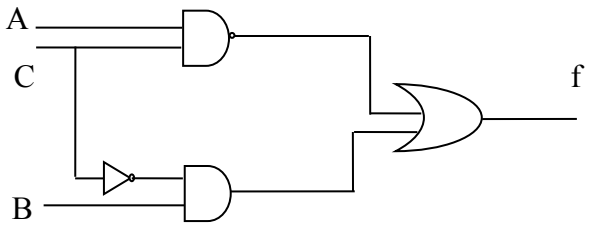
2nd forme:

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

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

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

d. Représentation par un logigramme:



I.5 Simplification des fonctions logiques:

La simplification d'une fonction revient à réduire le nombre de ses termes ou le nombre de variables dans un même terme. L'intérêt de cette réduction est de trouver l'écriture la plus simple pour une réalisation matérielle plus simple.

a. Réduction algébrique:

Exemple: Prenons la fonction écrite sous sa 1ère forme canonique suivante:

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

$$\Rightarrow f = (\bar{a}bc + abc) + (a\bar{b}c + abc) + (abc + abc)$$

$$= bc(a+\bar{a}) + ac(b+\bar{b}) + ab(c+\bar{c})$$

$$\Rightarrow f = bc + ac + ab$$

Cette méthode est hasardeuse et les manipulations deviennent très difficiles au-delà de trois variables.

b. Méthode de Karnaugh:

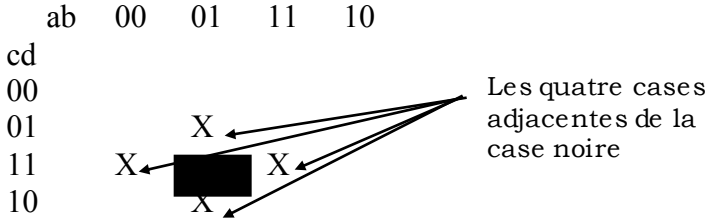
- Monômes adjacents et tableau de Karnaugh à 4 variables:

Pour simplifier une fonction on réunis des monômes adjacents (qui ne diffèrent que par une seule variable).

Exemple: $abc\bar{d} + abcd = abc(d+\bar{d}) = abc$

La méthode de KARNAUGH évite la recherche de ces monômes adjacents par un procédé algébrique. Donc le tableau doit être construit de telle sorte qu'entre une case et les cases adjacentes une seule variable change d'état.

Exemple:



- Simplification d'une fonction:

La simplification consiste, pour une fonction exprimée en somme canonique à rechercher des groupements de 2 cases, 4 cases et 8 cases adjacentes de façon à éliminer une, deux ou trois variables dans un groupe de monômes.

Les groupements doivent utiliser toutes les cases contenant un "1".

Exemple:

La fonction représentée par la table de vérité est sous sa forme canonique :

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

On peut faire 3 groupements dans le tableau:

1/ Les deux cases qui correspondent à:

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

2/ Groupement de quatre cases verticales:

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

3/ Groupement des 4 cases des 4 coins :

$$\bar{a}\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + \bar{a}\,\bar{b}\,c\,\bar{d} + a\,\bar{b}\,c\,\bar{d} = \bar{b}\,\bar{c}\,\bar{d}\left(a + \bar{a}\right) + \bar{b}\,c\bar{d}\left(a + \bar{a}\right) = \bar{b}\,\bar{c}\,\bar{d} + \bar{b}\,c\,\bar{d} = \bar{b}\,\bar{d}\left(c + \bar{c}\right) = \bar{c}\,\bar{d}$$

D'où :

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

a	b	c	d	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

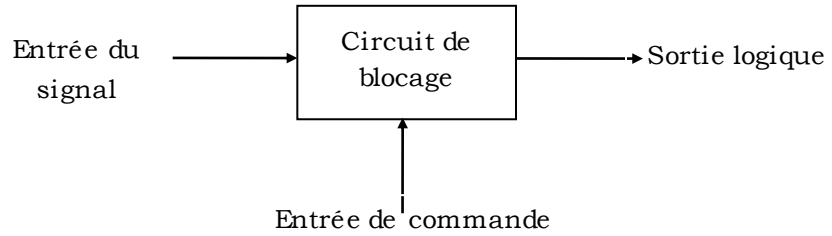
ab / cd	00	01	11	10
00	**1**	0	0	**1**
01	0	**1**	**1**	**1**
11	0	0	0	**1**
10	**1**	0	0	**1**

Chapitre III : Les circuits combinatoires.

1. la fonction de blocage
2. la fonction de comparaison
3. les fonctions arithmétiques (addition, soustraction)
4. les fonctions de codage, de décodage et de transcodage
5. la fonction d'aiguillage d'information (multiplexage) et démultiplexage

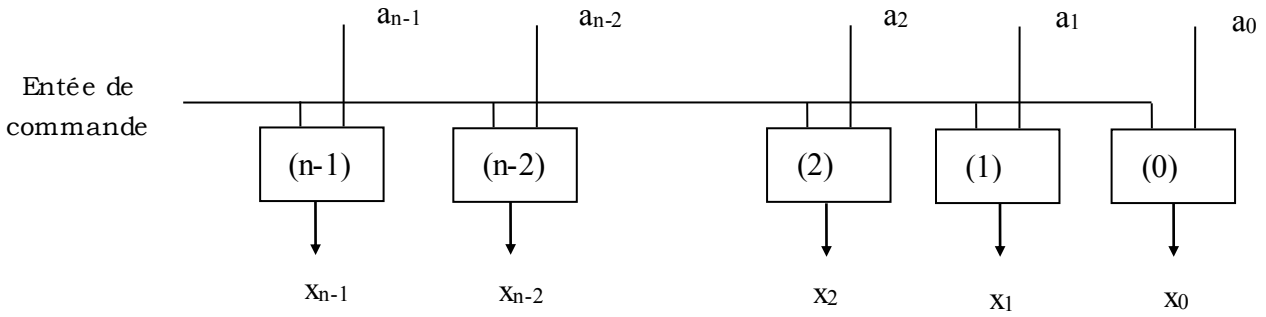
1. la fonction de blocage :

Le circuit associé à la fonction de blocage comporte à son entrée en plus d'une entrée logique, une entrée de commande qui valide ou bloque le passage du signal.



Remarque

Une information écrite sur n bits a_{n-1}, ..., a_2, a_1, a_0 commandée vers la sortie $x=x_{n-1}x_{n-2}.....x_2x_1x_0$, en utilisant la même entrée de commande pour tous les bits.



2. La fonction de comparaison :

Un comparateur de mots permet la comparaison d'un mot binaire A, avec un autre mot B de même langueur. Si les deux mots coïncident bit à bit, la sortie du comparateur est à '1'.

a/ cas d'un comparateur de mots d'un bit A_0 et B_0 :

A_0	B_0	S_0
0	0	1
0	1	0
1	0	0
1	1	1

$$S_0 = \overline{A_0}.\overline{B_0} + A_0.B_0 = \overline{A_0 \oplus B_0}$$

b/ Cas d'un comparateur de mots de deux bits A_1A_0 et B_1B_0 :

On compare A_1 avec B_1 et A_0 avec B_0. On désigne par S_1 le résultat de la 1iére comparaison et par S_0 celui de la seconde. La sortie du comparateur est à '1' si S_1 et S_2 sont les 2 à '1'.

S_1 (A_1=B_1)	S_0 (A_0=B_0)	S_2
0	0	0
0	1	0
1	0	0
1	1	1

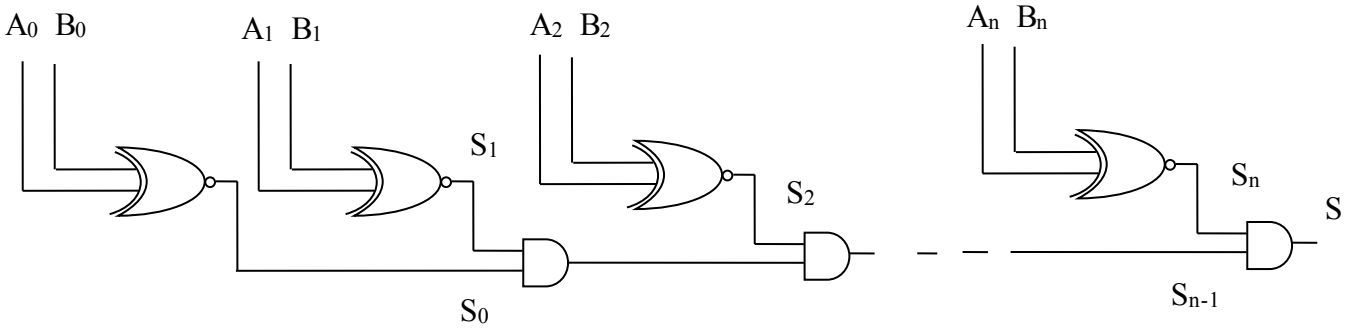
$$S_0 = S_1.S_2 = \overline{A_1 \oplus B_1}.\overline{A_0 \oplus B_0}$$

Cas général d'un comparateur de (n+1) bits

S_{n-1}	S_n	S
0	0	0
0	1	0
1	0	0
1	1	1

$$S = S_{n-1}.S_n$$

Le logigramme d'un tel système est donc le suivant



Comparateur de mots (n+1) bits

3. Les fonctions arithmétiques :

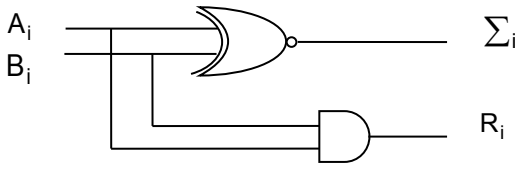
3.1 Demi-additionneur et additionneur complet :

Demi-additionneur :

Un demi-additionneur fait la somme de deux bits Ai et Bi de même poids

La table de vérité de ce circuit est la suivante :

A_i	B_i	$\sum_i$	R_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Logigramme d'un demi-addtionneur

$$\sum_i = \overline{A_i}.B_i + A_i.\overline{B_i} = A_i \oplus B_i$$

$$R_i = A_i.B_i$$

Additionneur complet :

L'additionneur complet fait l'addition de deux bits A_i et B_i ainsi que la retenue de l'étage précédent R_{i-1}

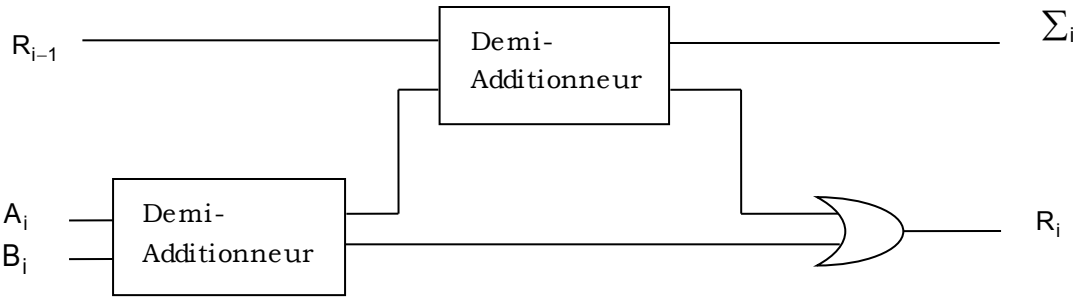
A_i	B_i	R_{i-1}	$\sum_i$	R_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\sum_i = \overline{A_i}.\overline{B_i}.R_{i-1} + \overline{A_i}.B_i.\overline{R_{i-1}} + A_i.\overline{B_i}.\overline{R_{i-1}} + A_i.B_i.R_{i-1}$$

$$\sum_i = \left(\overline{A_i}.\overline{B_i} + A_i.B_i\right)R_{i-1} + \left(\overline{A_i}.B_i + A_i.\overline{B_i}\right)\overline{R_{i-1}}$$
$$\sum_i = \left(\overline{A_i \oplus B_i}\right)R_{i-1} + \left(A_i \oplus B_i\right)\overline{R_{i-1}}$$
$$\sum_i = A_i \oplus B_i \oplus R_{i-1}$$

$$R_i = \overline{A_i}.B_i.R_{i-1} + A_i.\overline{B_i}.R_{i-1} + A_i.B_i.\overline{R_{i-1}} + A_i.B_i.R_{i-1}$$
$$R_i = \left(A_i \oplus B_i\right)R_{i-1} + A_i.B_i$$

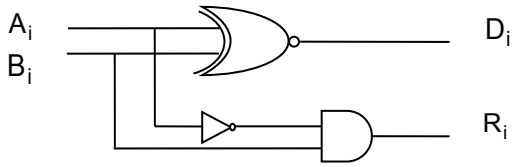
Donc, l'additionneur complet représente deux demi-additionneurs câbles en cascade.



Demi-soustracteur et soustracteur complet :

Demi- soustracteur fait la soustraction entre deux bits A_i et B_i de même poids.

A_i	B_i	D_i	Ri
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Demi-soustracteur

D_i : La différence et R_i : Le rapport.

$$D_i = \overline{A_i}.B_i + A_i.\overline{B_i} = A_i \oplus B_i$$
$$R_i = \overline{A_i}.B_i$$

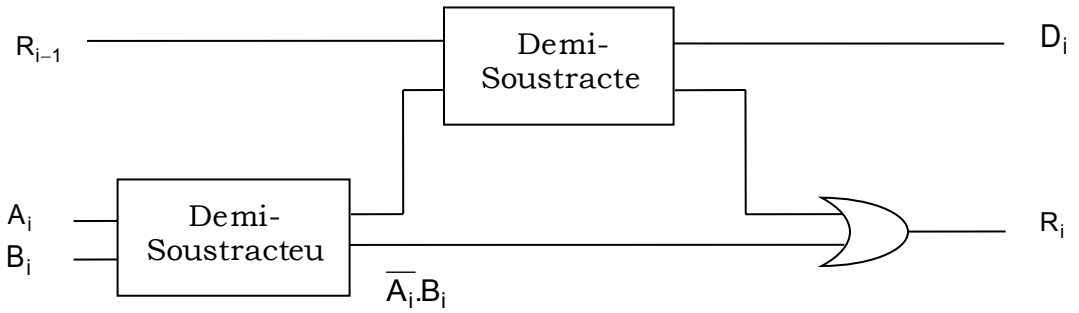
Soustracteur complet :

Le soustracteur complet fait la soustraction entre deux bits de même poids A_i, B_i et le rapport de l'étage précédent R_{i-1}.

A_i	B_i	R_{i-1}	D_i	R_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$R_i = \overline{A_i}.B_i + R_{i-1}.\left(\overline{A_i \oplus B_i}\right)$$

Le soustracteur complet est équivalent à 2 demi-soustracteurs mis en cascade.



$$D_i = \overline{A_i}.B_i.\overline{R_{i-1}} + A_i.B_i.R_{i-1} + \overline{A_i}.\overline{B_i}.R_{i-1} + A_i.\overline{B_i}.\overline{R_{i-1}}$$

$$D_i = A_i \oplus B_i \oplus R_{i-1}$$

Et

$$R_i = \overline{A_i}.\overline{B_i}.R_{i-1} + \overline{A_i}.B_i.\overline{R_{i-1}} + \overline{A_i}.B_i.R_{i-1} + A_i.B_i.R_{i-1}$$

Additionneur-soustracteur en complet à 2 :

On sait que la soustraction A-B en complet à 2, revient à faire :

$$A + (\text{en complet à } 2) = A + \overline{B} + 1$$

Un même circuit peut faire alors, l'addition et la soustraction de deux nombres binaires, suivant un signal de commande x.

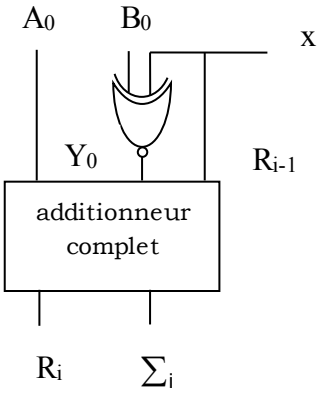
Si on pose que : pour x = 0 ; le circuit doit faire l'addition

Et pour x = 1 ; il doit faire la soustraction.

Alors les entrées du circuit réalisé avec des additionneurs, seront A et Y, avec Y = B si x = 0, Y = le complément de B si x = 1.

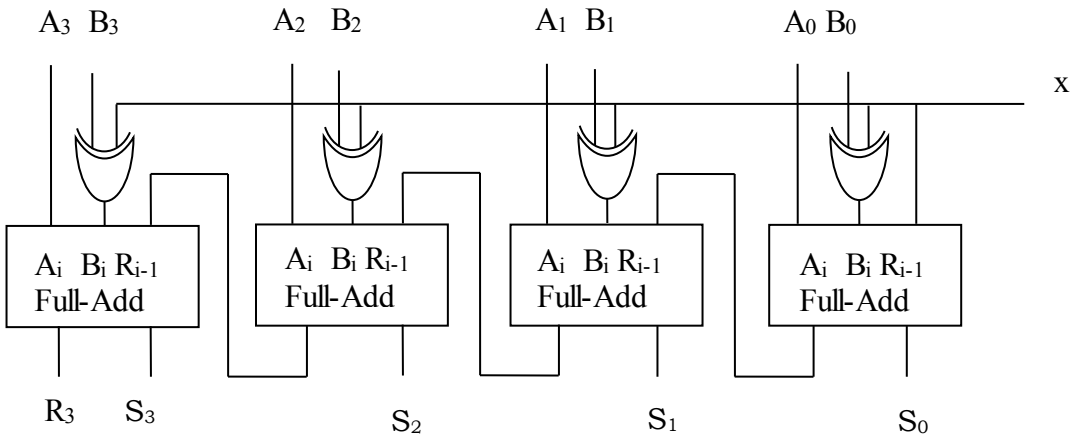
x	B_i	Y_i
0	0	0
0	1	1
1	0	1
1	1	0

$$Y_i = \overline{x}.B_i + x.\overline{B_i} = x_i \oplus B_i$$



L'entrée retenue, de l'additionneur de poids le plus faible, sera prise égale à x.

L'additionneur-soustracteur complément à 2, pour 2 nombres à 4 bits A et B est:

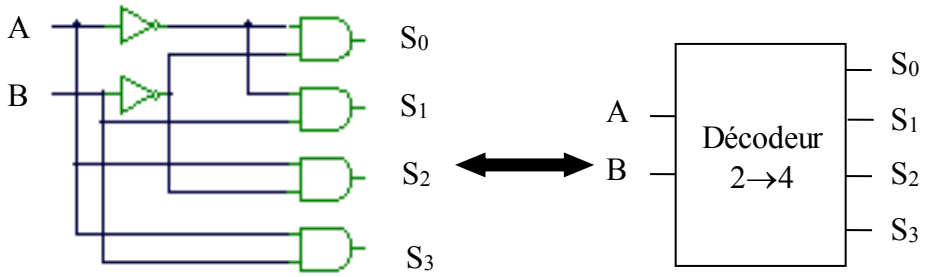


3.2. Les décodeurs et encodeurs :

Un décodeur est un circuit, qui à N entrées et 2^N sorties, dont une seule est active à la fois. En logique positive, la sortie active est à "1" et les autres sorties sont à "0" alors qu'en logique négative on a l'inverse.

Exemple: Un décodeur 2 → 4, fonctionnant en logique positive.

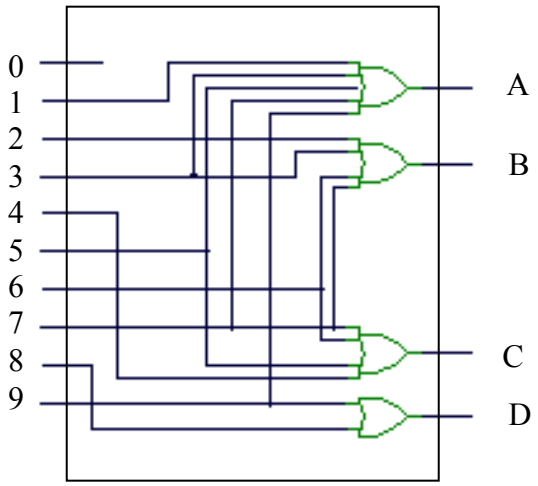
Entrées		Sorties			
A	B	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Le fonctionnement d'un encodeur est l'inverse de celui d'un décodeur. Il sert à générer une sortie codée en binaire ou en BCD, à partir d'une seule entrée active à la fois.

Le tableau de fonctionnement et le logigramme d'un encodeur décimal-BCD sont :

Entrées décimales	Sorties D C B A			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

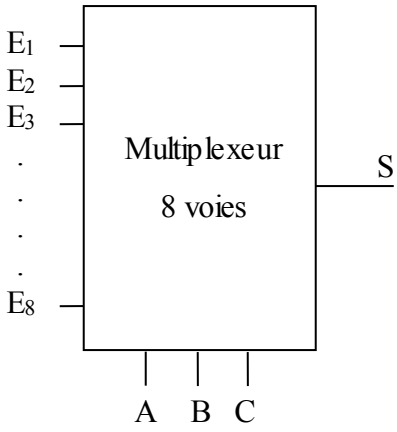


D=8+9 ; C=4+5+6+7 ; B=2+3+6+7 ; A=1+3+5+7+9.

3.3. Les multiplexeurs et démultiplexeurs :

Le multiplexage est la technique de sélection et de transmission de signaux émanant de plusieurs sources vers une seule voie. Un multiplexeur est un circuit qui a 2^N entrées dites d'information, N entrées adresses dites de commande et une seule sortie. L'état de la sortie indique la valeur de l'entrée d'information qui correspond à la valeur des entrées adresses.

Exemple: un multiplexeur à 8 entrées (E_i, i=1…,8) possède 3 lignes de commande (A, B, C) et une sortie S (Le MUX 74151). Le fonctionnement d'un tel circuit est défini par la table de vérité suivante:



A	B	C	S
0	0	0	E_1
0	0	1	E_2
0	1	0	E_3
0	1	1	E_4
1	0	0	E_5
1	0	1	E_6
1	1	0	E_7
1	1	1	E_8

$$S = \overline{A}\,\overline{B}\,\overline{C}E1 + \overline{A}\,\overline{B}CE2 + \overline{A}B\overline{C}E3 + \overline{A}BCE4 + A\overline{B}\,\overline{C}E5 + A\overline{B}CE6 + AB\overline{C}E7 + ABCE8$$

Un démultiplexeur fait la fonction inverse d'un multiplexeur. C'est un circuit qui possède une entrée E et 2^N sorties. Le transfert de l'entrée vers l'une des sorties se fait grâce à N signaux de commandes. La table de vérité suivante montre, le fonctionnement d'un démultiplexeur 4 voies.

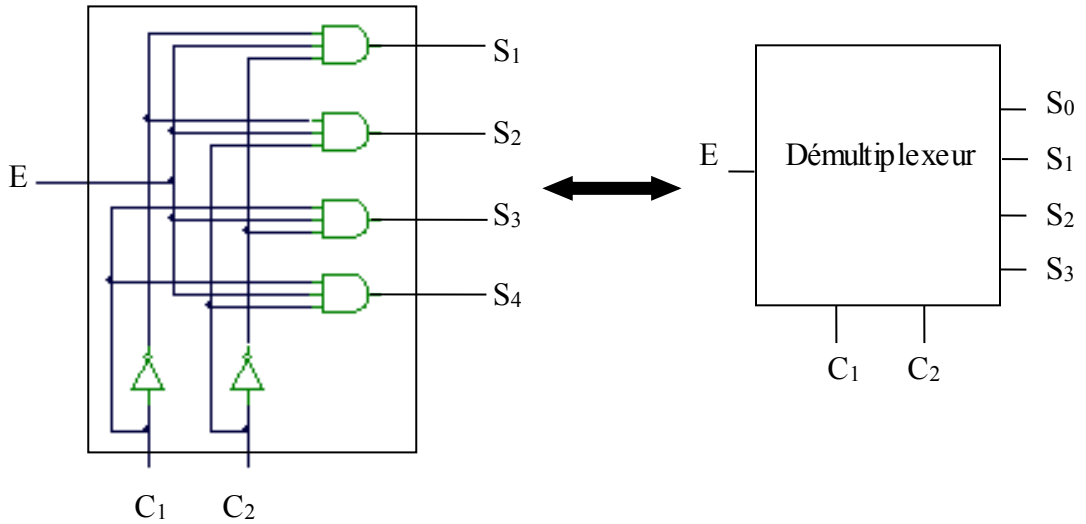
Table de vérité d'un démultiplexeur 4 voies:

C_1	C_2	S_1	S_2	S_3	S_4
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

Les expressions des sorties en fonction des entrées sont alors définies comme suit:

$$S_1 = \overline{C}_1.\overline{C}_2.E\,,\ S_2 = \overline{C}_1.C_2.E\,,\ S_3 = C_1.\overline{C}_2.E \quad \text{et} \quad S_4 = C_1.C_2.E$$

Ce démultiplexeur est représenté par la figure ci-dessous:



La plus grande utilité du multiplexeur et de son inverse le démultiplexeur se trouve dans les montages de transmission d'information, où il est question de transformation parallèle-série et inversement.

3.4. Les transcodeurs ou convertisseurs de code:

Un transcodeur permet la conversion d'un mot exprimé dans un code donné sur p bits, en son équivalent dans un autre code donné sur k bits.

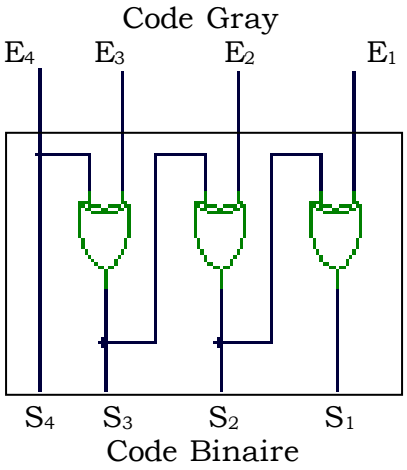
Le lien entre ces codes est donné par les fonctions logiques, calculées à partir de la table de vérité en utilisant les méthodes habituelles.

On peut trouver sur le marché des transcodeurs disponibles sous forme de circuits intégrés, comme par exemple le transcodeur excédant3-décimal (7442), le transcodeur décimal-excédant3 (7443) et le transcodeur excédant3-GRAY (7444).

Transcodeur Gray-Binaire:

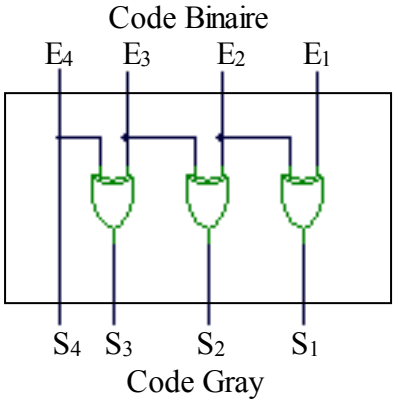
E_4	E_3	E_2	E_1	S_4	S_3	S_2	S_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Après simplification on obtient:
$S_4=E_4$, $S_3=E_3\oplus E_4$, $S_2=E_2\oplus E_3\oplus E_4$ et $S_1=E_1\oplus E_2\oplus E_3\oplus E_4$.
Ce qui donne le logigramme suivant:



Transcodeur Binaire-Gray:

De la même façon, en inversant les entrées/sorties on obtient le schéma suivant:



La table de vérité donne les fonctions logiques de sorties suivantes:
$S_4=E_4$, $S_3=E_3\oplus E_4$, $S_2=E_2\oplus E_3$, $S_1=E_1\oplus E_2$

Transcodeur Gray-Binaire/Binaire-Gray:

La conversion Gray-Binaire ou Binaire-Gray peut être facilement faite par un seul circuit commandé par un signal désignant le sens de conversion.
Ce circuit représente une combinaison des deux logigrammes obtenus précédemment avec des multiplexeurs, comme indiqué ici: