Logique combinatoire

Kachouri Abdennaceur ENIS Département GE

Université Virtuelle de Tunis

2006

Introduction

Ce module porte sur les circuits logiques combinatoire.... Il couvre plus spécifiquement la représentation des nombres, l'algèbre de Boole et les fonctions élémentaires, la description et la simplification des fonctions logiques. On se concentre sur l'étude détaillée des circuits logiques combinatoires : multiplexeur, décodeur, mémoire, additionneur, unité arithmétique et logique, encodeur de priorité, générateur et vérificateur de parité, comparateur….

Le module s'adresse d'abord aux personnes qui s'intéressent à une connaissance de base aux circuits numériques et l'architecture des ordinateurs Il s'inscrit dans le programme du diplôme techniciens supérieures en génie électrique et informatique et aux élèves ingénieurs et. aux étudiants en maîtrise de physique. Module de base, aucune connaissance base ne requise pour comprendre ce cours.

Ce « Guide d'étude » a pour objectif de vous préparer à suivre le cours. Il définit en quelque sorte un mode d'emploi, non seulement pour le matériel didactique du cours, mais aussi pour le cheminement que vous devez adopter et les différentes exigences auxquelles vous devez répondre.

Bonne lecture et bon cours!

Le but de ce module est de se familiariser la circuits logique combinatoire. Plus spécifiquement, au terme de ce module, l'étudiant ou l'étudiante sera en mesure :

· De Maîtriser la représentation binaire des nombres signés non signés

· D'utiliser les règles de l'algèbre de Boole

· D'identifier les symboles des fonctions logiques élémentaires.

· De définir leurs tables de vérité.

· D'élaborer les équations ainsi que les chronogrammes d'une fonction combinatoire.

· De concevoir un logigramme à partir d'un équation logique

· Maîtriser la représentation et la simplification des fonctions logiques

· Étudier les différents types de circuits utilisant la logique combinatoire

---

Contenu du cours

Tableau 1 : Le contenu du module se compose de 3 chapitres subdivisés en 13 leçons [ou sections, parties, ...]**.**

Chapitre	Leçon	Résumé
1	1	Numération et codage -Les codes binaires pondérés, conversions, - Les codes binaires non pondérés
1	2	Arithmétique binaire - Représentations des nombres signés - Addition / soustraction des nombres non signés, - Addition / soustraction des nombres signés

2	3	Algèbre de BOOLE : les règles de l'algèbre de Boole
2	4	Algèbre de BOOLE 2 : - Représentation des variables et des fonctions
3	5	Fonctions élémentaires et circuits associés
3	6	Simplification des fonctions logiques
3	7	Propriétés et simplification des fonctions
4	8	Les circuits codeurs, décodeurs, transcodeurs
4	9	Les circuits multiplexeurs, démultiplexeurs
4	10	Les circuits d'addition, soustracteur
4	11	Unité Arithmétique et logique
5	12	Introduction sur PLA

5	13	Réalisation des fonctions logiques

Approche pédagogique

Ce cours… est conçu selon une approche pédagogique propre à la formation à distance. Le matériel didactique et la formule utilisée vous permettent d'adopter une démarche d'apprentissage autonome. Vous pouvez ainsi gérer votre temps d'étude et prendre en charge votre formation.

Toutefois, cette prise en charge est soutenue par la personne responsable de l'encadrement (le tuteur ou la tutrice), pendant toute le semestre. Sa tâche est de vous faciliter les conditions d'apprentissage et de vous aider dans votre démarche, de façon à ce que vous atteigniez les objectifs du cours. Il va de soi que le tuteur ou la tutrice ne donne pas les réponses des activités notées. Vous pouvez communiquer avec votre tuteur ou votre tutrice par le courrier électronique offert sur le site du cours ou en posant vos questions sur le forum. Votre tuteur ou votre tutrice y répondra à l'intérieur de 48 heures.

Haut de la page

Charge de travail et calendrier

Ce module est offert à distance sur une semestre de 13 semaines. Le volume de travail exigé pour l'étude du module et la réalisation des évaluations est de 2 heures par semestre. En moyenne, la charge de travail hebdomadaire est donc d'environ 1 heure. Certains leçons [ou sections ou…] **sont un peu plus longs à lire que d'autres, mais ils exigent moins de travail sous forme d'exercices. Un calendrier pédagogique détaillé est proposé au Tableau 3.**

Tableau 3: Calendrier pédagogique

Semaine	Module	Tâche	Envoi de l'évaluation

1	**1**	· **Lecture du guide pédagogique** · **Lecture de la leçon 1** · **Auto-évaluation**	
2	**1**	· **Lecture de la leçon 2** · **Auto-évaluation**	
3	**1**	· **Lecture de la leçon 3** · **Auto-évaluation**	
4	**1**	· **Lecture de la leçon 4** · **Auto-évaluation**	**Travail** **1 : date**
5	**2**	· **Lecture de la leçon 5** · **Auto-évaluation**	
6	**2**	· **Lecture de la leçon 6** · **Auto-évaluation**	

7	**2**	· **Lecture de la leçon 7** · **Auto-évaluation**	
8	**2**	· **Lecture de la leçon 8** · **Auto-évaluation**	**Travail** **2 : date**
9	**3**	· **Lecture de la leçon 9** · **Auto-évaluation**	
10	**3**	· **Lecture de la leçon 10** · **Auto-évaluation**	
11	**3**	· **Lecture de la leçon 11** · **Auto-évaluation**	
12	**3**	· **Lecture de la leçon 12** · **Auto-évaluation**	

13	3	· Lecture de la leçon 13 · Auto-évaluation	
14	1-2-3	· Révision	
15		· Examen final sous surveillance	Date de l'examen

Évaluation des apprentissages

L'auto-évaluation

Cette évaluation n'est pas notée. Elle est présentée sous forme d'activités d'intégration, de questions à répondre ou d'exercices à effectuer. Cette auto-évaluation met l'accent sur les points les plus importants de la matière. Le corrigé des exercices est disponible, mais nous vous suggérons de ne le consulter qu'après avoir complété les exercices. Ces derniers vous préparent aux évaluations notées.

Les travaux notés

Ces travaux visent à vérifier l'acquisition de vos connaissances et votre compétence à appliquer et à

transférer les notions étudiées à des situations concrètes. Le français utilisé dans vos travaux d'évaluation doit être correct. Un travail illisible, jugé irrecevable par votre professeur, vous sera retourné pour être refait. Vous devez obligatoirement réaliser et retourner *aux dates prévues* (voir la fiche calendrier) les travaux notés et passer l'examen final sous surveillance.

Examen sous surveillance

L'examen final sous surveillance porte sur toute la matière du cours et sera constitué de [Expliquez ici le type d'examen : questions objectives, à développement, études de cas, problèmes, etc.]**. L'utilisation des notes de cours et de la calculatrice sera autorisée** [ou non, selon le cas].

L'ensemble des évaluations notées compte pour 100 % de la note du cours. En voici, à titre d'exemple, un partage :

Évaluation notée	Pondération	Seuil de passage
Travail 1	X_1 %	
Travail 2	X_2 %	
Examen final	40 %	50 %
Total	100 %	60 %

Systèmes de numérotation et représentation binaire

Nombres et système de numération Les codes binaires pondérés

1.1- Terminologie:

1.1.1-Le système décimal:

Le système de numération le plus utilisé est le système décimal qui utilise les dix symboles suivants: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Le nombre de symbole utilisé dans un système s'appelle sa base. La base du système décimal est 10.

Dans un nombre, chaque terme est associé à une valeur. Cette valeur est égale à la base élevée à une puissance qui dépend de sa position du terme dans le nombre. D'une façon générale, chaque nombre doit être exprimé par les symboles qui le forme multiplier par des puissances croissantes de sa base comme le montrent les exemples suivants.

$$5871 = 1x10^{0}+7x10^{1}+8x10^{2}+5x10^{3}$$

$$562 = 2x10^{0}+ 6x10^{1}+ 5x10^{2}$$

D'une manière générale un nombre entier positif N s'exprime dans une base B, au moyen de B symboles a_i de la manière suivante:

$$N = a_{n-1}B^{n-1}+a_{n-2}B^{n-2}+ ....+a_1B^1+a_0B^0$$

Dans la pratique la base est sous-entendue et l'on se contente de juxtaposer les coefficients a_i et on écrit N de la manière suivante:

$$N = a_{n-1}a_{n-2}....a_1a_0$$

1.1.2- Le système binaire:

Dans le système binaire on a que deux symboles qui sont 0 et 1; donc la base du système binaire est 2. On donne dans le tableau ci dessous l'équivalence des nombres décimaux de 0 à 15 dans le système binaire.

Système décimal	Système binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

On vérifie facilement qu'on associe au nombre 1110 en base 2, couramment représenté par 1110/2, le nombre 14/10 en binaire. En effet:

1110 = 0x20 +1x21+1x22+1x23.= 0 + 2 + 4 + 8 = 14

Les symboles binaires sont souvent appelés bit, on dit par exemple que le nombre 1110/2 a 4 bits; par contraction de l'expression anglaise: BINARY DIGIT (chiffre binaire)

1.1.3- Le système octal:

Le système octal est formé par 8 symboles qui sont 0, 1, 2, 3, 4, 5, 6, 7; donc c'est système a base 8.

1.1.4- Le système duodécimal:

Le système duodécimal est formé par 12 symboles qui sont: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, α , β .

donc c'est système a base 12.

Base 12	0	1	2	3	4	5	6	7	8	9	α	β
Base 10	0	1	2	3	4	5	6	7	8	9	10	11

On associe au nombre $1\ \alpha\ \beta\ 2\ \alpha\ /_{12}$ le nombre $3346\ /_{10}$; en effet on a:

$1\beta\ 2\ \alpha\ /_{12} = \alpha\ .12^0+2.12^1+\ \beta\ .12^2+1.12^3= 3346/_{10}$.

1.1.5- Le système hexadécimal:

Le système hexadécimal est système a base 16. Il emploie 16 éléments qui sont :

Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

1.2- Système de comptage:

D'autres systèmes de numération sont fréquemment utilisés. Ces systèmes sont:

Base 1: Le comptage avec les doigts, cailloux, entailles.

Base 2: système binaire, logique symbolique, ordinateurs.

Base 5: système quinaire, Aztèques.

Base 7: notes musicales, jours de la semaine.

Base 8: système octal, ordinateur.

Base 10: système décimal adopté par l'Homme.

Base 12 : système duodécimal, gamme des notes et demi-tons, mois de l'année; heures.

Base 16: système hexadécimal, ordinateur.

Base 20: comptage sur les doits des mains et des pieds; Mayas.

Base 24: Heures du jour.

Base 60: degrés, minutes et secondes, savants Babylone.

1.3- Nombres binaires fractionnaires:

D'une façon générale pour un système de numération de base B, la partie fractionnaire d'un nombre est définie comme étant la somme des produits des symboles constituant ce nombre par des puissances négatives associe aux poids de ces symboles.

$N_F = 0,a_{-1}a_{-2}.....a_{-n}$

$N_F = a_{-1}xB^{-1}+a_{-2}B^{-2}+ ....+a_{-n}B^{n}$

La conversion binaire - décimale d'un nombre binaire fractionnaire est très simple.

Voici un exemple: $0.101/_2 = 1x2^{-1}+0x2^{-2}+1x2^{-3} = 0,625/10.$

1.4- Conversion entre système de numération à bases différentes:

1.4.1- Principe de changement de base:

Soit un entier N dans une base A et qui s'écrit dans une base B comme suit:

$N_B = a_{n-1}xB^{n-1}+a_{n-2}B^{n-2}+ ....+a_1B^1+a_0B^0$

Si nous divisons N par B nous obtenons un quotient Q_1 et un reste R_1 tels que:

$N_B= Q_1.B+R_1 = B\{a_{n-1}xB^{n-2}+a_{n-2}B^{n-3}+ ....+a_1\}+a_0$

Le reste R_1 représente le terme a_0

Si nous divisons Q_1 par B nous obtenons un quotient Q_2 et un reste R_2 tels que:

$Q_1=\{a_{n-1}xB^{n-3}+a_{n-2}B^{n-4}+ ....+a_2\}+a_1.$

Le reste R_2 représente le terme a_1 et ainsi de suite jusqu'on obtient le premier quotient inférieur à B. La division peut s'effectuer dans n'importe quel base, à condition de savoir compter facilement dans la base en question.

1.4.2- Conversion d'un entier décimal en son équivalant dans une base quelconque:

Pour convertir un nombre décimal à une autre base, on divise ce nombre par la base, puis on divise successivement les quotients obtenus par cette base jusqu'a ce que le dernier quotient soit zéro. L'équivalent du nombre est donné par les restes successifs. L'exemple ci dessous illustre la méthode pour la conversion de 200/10 en base 8.

		200 divisé par 8
reste **0 poids faible**	et le quotient est 25,	25 divisé par 8
reste **1**	et le quotient est 3,	3 divisé par 8
reste **3 poids fort**	et le quotient est **0**.	On arrête la division.

Le nombre recherché est **310**.

200/10 = 310/8

1.4.3- Conversion d'un nombre décimal en base B:

La méthode la plus générale de conversion d'un nombre décimal en un nombre à base quelconque B, est celle des divisions successives par B de la partie entière et les multiplications successives par B de la partie fractionnaire.

1.4.4- Conversions entre les systèmes à base 2, 8, 16:

- ***Conversions octal -- binaire:***

Pour convertir un nombre octal en binaire, il suffit de remplacer chaque chiffre par son équivalent à base 2, exprimé par 3 bits. On utilise la réciproque de cette propriété pour effectuer la conversion binaire octale, que le nombre soit entier ou fractionnaire.

Exemple 1: Donner l'équivalent binaire de 63_8.

6	3
----	----
110	011

$$63_8 = 110011/_2$$

Exemple 2: Donner l'équivalent binaire de $63,54/_8$

6 3, 5 4

110011,101100

$$63,54/_8 = 110011,101100/2$$

Exemple 3: Donner l'équivalent octal de $N = 1001110,1001/_2$

On regroupe les bits 3 par 3, de par et d'autre de la virgule, en complétant par zéro, si c'est nécessaire

N =1/001/110,100/100

N =116,44/$_8$

- ***Conversions Hexadécimale -- binaire:***

Pour convertir un nombre hexadécimal en binaire, il suffit de remplacer chaque chiffre par son équivalent à base 2, exprimé par 4 bits. On utilise la réciproque de cette propriété pour effectuer la conversion binaire hexadécimale, que le nombre soit entier ou fractionnaire.

Exemple 1: Convertir le nombre N = B7, A9

	B	7	A	9
équivalents décimaux	11	7	10	9
équivalents binaires	1011	0111	1010	1001

$$(B7,A9)_{16} = (10110111,10101001)_2$$

Exemple 2: Donner l'équivalent hexadécimal de N = $(11010111,1)_2$

On regroupe les bits 4 par 4, de par et d'autre de la virgule, en complétant par des zéros si c'est nécessaire:

$N = (1101/0111,1000)_2$

$N = (D7,8)_{16}$

Par la même méthode de conversion d'un nombre exprimé dans une base B à son équivalent dans le système décimal on associe le nombre 1AF/16 au nombre 431/10.

Arithmétique binaire

2.1- Introduction:

Le mécanisme des opérations appris et acquis en arithmétique décimale, peut être rigoureusement appliqué au système binaire. En particulier le maniement des retenues, quand elles existent, est identique.

Pour effectuer les opérations de soustraction ou de division de deux nombres binaires, il est nécessaire de comparer ces deux nombres. La procédure d'une comparaison de 2 nombres binaires, A et B, est la suivante:

On examine chaque bit en commençant par le rang le plus élevé :

*Si à ce rang, le nombre A présente un bit UN (1), alors que le nombre B présente un bit ZÉRO (0), le résultat de la comparaison est A > B.

* Si c'est le cas contraire, on en déduit que A < B.

* Si les 2 nombres présentent la même valeur (un bit UN (1) ou un bit ZÉRO (0)), on examine de la même façon, les bits immédiatement inférieur. Il est évident que si, rang après rang, les bits sont identiques, c'est que les 2 nombres sont égaux.

2.2- Addition:

Les règles de l'addition en l'arithmétique binaire sont résumées ci-dessous.

+	0	1
0	0	1
1	1	10

Table d'addition

Exemples

Ex 1: A = 10 et B = 1 on cherche S = A + B, S = 10+1 = 11

Ex 2: A = 11 et B = 1 on cherche S = A + B

1 **1** ⇐ **retenue de 1+1**

\+ **1**

100

S =100

Ex 3 : A = 101 et B = 111 calculer S = A + B

101 ⇐ **retenue de 1+1**

\+ 111

1100

S =1100

2.3- Multiplication:

Comme en numération décimale, la multiplication de deux nombres binaires se faire selon le procédé classique en utilisant la table de multiplication ci dessous. On obtient alors une suite de produits partiaux, décalés l'un par rapport à l'autre, d'un rang vers la gauche, et qui sont ensuite additionnés pour trouver le résultat de la multiplication.

X	0	1
0	0	0
1	0	1

Multiplication a.b

Exemples

Ex 1: Calculer le produit P = AB avec A = 110 et B = 101.

```
  110
x
  101
---------
  110
 000
110          produis partiels
--------------
11110     produit
```

Remarques

- La multiplication d'un nombre par 2, 4, 8... 2^k se fait en ajoutant 1,2, 3, ..k zéros à droite de ce nombre si c'est un entier, ou décalant la virgule de 1, 2 ou de 3...k rangs vers la droite, si le nombre est fractionnaire.

Arithmétique binaire

2.4- Soustraction:

2.4.1. Soustraction directe

2.4.1- Soustraction directe:

Il faut noter que, comme en arithmétique décimale, si on veut dans un rang, soustraire : 0 - 1 par exemple, on effectue 10 -1 et on retranche 1 de retenue au rang suivant. Cette opération de supprimer de rang immédiatement supérieur, qu'on avait momentanément ajouté pour faire l'opération.

-	0	1
0	0	1
1	1	0

Table de soustraction

Exemples:

Ex 1: Effectuer la soustraction de A - B avec A = 111 et B = 101.

```
  111
-
  101
------
  010
```

D = A - B = 111-101 = 010

Ex 2: Effectuer la soustraction de A - B avec A = 10 et B = 1

```
   10
-
    1
-----
```

01

D = 01

Ex 3: Effectuer la soustraction de A - B avec A = 11 et B = 101.

0 1 1

1 1 0 1

---

1 1 1 1 0

Remarques

On note que A< B donc la différence (A- B) est négative.

On remarque aussi, que on a une suite illimité de 1 (..1110)

- ***Transformation d'une soustraction en addition par le complément vrai:***

Si la différence est négative l'opération de soustraction devient très complexe puisqu'on ne sait pas quand il faut arrêter l'opération. Pour résoudre ce problème on transforme les soustractions en addition en prenant le complément vrai du nombre à soustraire.

Pour effectuer correctement ces opérations il faut fixer le nombre de bits. Par exemple nous allons fixer le nombre de bits à 4..

Ex 1: Effectuer la soustraction de A - B avec A = 0101 et B = 0011.

A - B = A + Opposé(B) = A + C_v B

C_v B = 1101, A + C_v B = 0101 + 1101

0 1 0 1

+

1 1 0 1

---

0 0 1 0

dans cet exemple, on prend que les quatre bis les mois significatifs, puisque nous avons fixé le nombre à 4 bits. On constate que le résultat est positif, puisque le dernier bit le plus gauche est égal à 0, et de valeur absolue égale à 2_{10} donc la différence est 0010 équivalent à +(2_{10}).

Ex 2: Effectuer la soustraction de B -A avec A = 0101 et B = 0011.

B - A = B+ Opposé(A) = B + C_v A

C_v A = 1011 , B + C_v A = 0011 + 1011

0 0 1 1

1 0 1 1

---

1 1 1 0

On constate que le résultat est sur 4 bits le dernier bit le plus à gauche est égale à 1 donc ce résultat est négatif. Pour chercher sa valeur absolue, il faut chercher le complément vrai du résultat.

C_v(résultat) = 0010 donc la différence de B - A est égale à (-2_{10}).

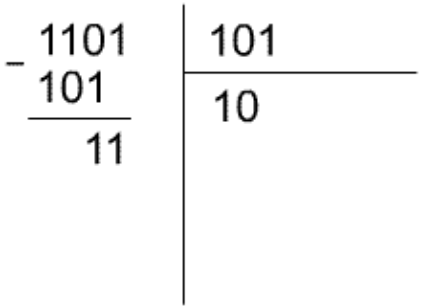
2.5- Division:

Les remarques faites de la multiplication binaire s'appliquent également à la division binaire. En effet, pour la division, il suffit donc de comparer les nombres en présence (le diviseur et le dividende) et d'écrire un bit UN au quotient si la soustraction est possible (dividende - diviseur >0), et un bit ZERO si elle ne l'est pas. C'est à dire que le diviseur ne peut être contenu qu'une seule fois le dividende ou pas du tout. Ces opérations sont effectuées à chaque reste partiel de la division.

Exemple: effectuer la division de A par B avec A = 1101 et B = 101

e l'est pas.

Exemple: effectuer la division de A par B avec A = 1101 et B = 101



Représentations binaires de l'information Les codes non pondérés

3.1-Les codes décimaux binaires:

3.1.1- Introduction:

Supposons que nous disposions de n chiffres de base B; on peut alors former B^n combinaisons différentes permettant de constituer B^n permutations. Chacune de ces permutations constitue un code valable pour la représentation des B^n nombres entiers. Toutefois, parmi tous ces codes, seul un petit nombre présentent des propriétés intéressantes que nous allons étudier.

L'opération qui établit une correspondance entre une donnée quelconque en une grandeur binaire s'appelle le **codage** (ou encodage). Le codage est une application bijective. L'application inverse est appelée le **décodage**. L'opération qui consiste à transposer une grandeur binaire, donnée dans un code A, en une autre grandeur binaire, équivalente dans un code B, s'appelle le **transcodage**.

3.1.2- Codes décimaux binaires pondérés DCB ou(Binary Coded Decimal) BCD:

Dans les ordinateurs on utilise les systèmes logiques, donc les données sont traitées sous forme binaire. Par exemple le nombre 25 en décimal devient 11001 en binaire. Cette représentation peut prendre un nombre considérable des bits pour les grands nombres décimaux. On rappelle que le système décimal est celui qui convient le mieux aux Hommes pour la représentation des nombres tandis que le système binaire est celui qui convient le mieux aux systèmes logiques. On peut imaginer des systèmes des numérations acceptables à la fois par l'homme et les machines. On peut prendre tout chiffre décimal par un groupe de 4 bits comme le montre le tableau Ce code est le système Décimal Codé en Binaire connu sous le DCB ou par les Anglo-saxons par le code **BCD**. En effet, avec ces codes, chaque chiffre du nombre décimal correspond à un groupe de 4 bits du nombre binaire que l'on obtient par conversion directe décimale-binaire, comme le montre l'exemple suivant :

La conversion du nombre 75 (décimal) en BDC 0111 1001

La conversion du nombre 154 (décimal) en BDC 0001 0101 0100

Le système de numération le plus utilisé est le système binaire naturel à 4 bits, pondéré, avec les poids 1, 2, 4, 8. Toutefois, il existe d'autres façons de pondérer les chiffres binaires. Quatre codes sont donnés dans le tableau ci dessous: **code 4321**, **code 4421**, **code 5221 code 5421**.

décimal	8421	4321	4421	5221	5421
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0001

2	0010	0010	0010	0010	0010
3	0011	0100	0011	0011	0011
4	0100	1000	0100	0110	0100
5	0101	1001	0101	0111	1000
6	0110	1010	0110	1001	1001
7	0111	1100	0111	1010	1010
8	1000	1101	1100	1011	1011
9	1001	1110	1101	1110	1100

3.1.3- Codes décimaux binaires symétrique ou codes auto - complémentaires:

Nous parlons que le but recherché ici est la simplicité de l'obtention le complément à 9 d'un nombre qui interviennent dans certaines méthodes de traitement des nombres négatifs. Nous rappelons la définition du complément à 9 d'un nombre N composé de k chiffres est égale à 10^k-1-N. Le **code EXCES de 3** et le code **AIKEN** sont fréquemment utilisés, ils sont :

- ***Codes EXCÈS(XS3):***

Codes EXCÈS (XS3) est un code décimal auto - complémentée, obtenu à partir des combinaisons du code BCD, auxquelles on ajoute systématiquement 3 (0011). Ce code a été créé pour permettre la réalisation simple des opérations de soustraction. En effet le complément à 9 d'un nombre s'obtienne en prenant le complément à 1 de chaque des chiffres du code binaire, c'est à dire en remplaçant les 0 par des 1 et réciproquement.

- ***Code AIKEN:***

Code AIKEN : C'est un code décimal pondéré auto - complémentaire, avec les poids des éléments binaires sont 2421. Par l'intermédiaire de ce code on peut avoir deux représentations du nombre 4 (1010 et 0100) et du nombre 6 (1100 et 0110)

décimal	Code Excès de 3	Code AIKEN
0	0011	0000
1	0100	0001
2	0101	0010
3	0110	0011

4	0111	0100
5	1000	0101
6	1001	0110
7	1010	0111
8	1011	1110
9	1100	1111

REPRÉSENTATION BINAIRE DE L'INFORMATION

3.2- Codes binaires purs, codes binaires réfléchis (Code Gray) Reflex:

- Définition distance entre deux nombres:

On appelle distance entre 2 combinaisons successives d'un code, le nombre de bits qui changent de l'une à l'autre. Par exemple la distance entre 1001 et 1010 est de 2, de même la distance entre 1011 et 1100 est de 3.

Les codes réfléchis sont dits : à distance unitaire, parce qu'il n'y a toujours qu'un seul bit qui change entre deux combinaisons successives.

Quand cette propriété est encore vérifiée entre la dernière et la première combinaison, on a affaire avec un code réfléchi cyclique.

Les codes réfléchis les plus utilisés sont les codes Gray (du nom de leur inventeur)

Le code binaire pur est le code qu'on a défini au paragraphe précédent.

- Première méthode:

Le code binaire réfléchi ou le code gray connu aussi par le code cyclique est très répondu. Il est fait d'une combinaison simple de 0 et de 1 réfléchie progressivement tandis que le système s'accroît. Les deux premiers nombres du code Gray sont le 0 et le 1 qui représentent le zéro et le un. On obtient les deux nombres suivant en réfléchissant la combinaison par rapport à un premier miroir et en ajoutant un 1 au digit de rang immédiatement supérieur (voir l'exemple ci dessous). On procède de la même manière pour retrouver les quatre nombres suivants puis 8 nombres puis les 16 nombres ainsi de suite avec des puissances de deux. En pratique, on ne dépasse pas 4 à 5 bits pour le codage en Gray. Le tableau ci dessous regroupe la représentation de code gray de 0 à 15.

0 0 0

1 0 1

---------- miroir

. 1 1 1 0 1 0

---------- deuxième miroir

1 0 0

1 0 1

1 1 1

1 1 0

Code décimal	Code binaire	Code gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

8	1000	1100
9	1001	1101

- ***Deuxième méthode:***

Il existe plusieurs moyens de construire un code Gray. En voici un qui prend, pour code de départ, un code binaire naturel à n bits:

Prenons par exemple la combinaison du code CBN à 4 bits, équivalente à (12)10.

1 1 0 0

$b_3\ b_2\ b_1\ b_0$

- On conserve, systématiquement, le bit le plus à gauche (ici le bit b3)

- Le bit suivant : b2 est:

  - conservé si le précédent b3=0

  - inversé si le précédent b3=1

- On poursuit le même raisonnement jusqu'au bit b0

CBN	1	1	0	0
	b3	**b2**	**b1**	**b0**
	b3 Conservé	**b2 inversé** **puisque b3=1**	**b1 inversé** **puisque b2 = 1**	**b0 conservé** **puisque b1=0**
Code Gray	1	0	1	0

- ***troisième méthode:***

Méthode par le calcul de la valeur de N' , exprimé en le code gray par rapport à N exprimé en CBN est:

N' = Partie entière ((N $\oplus$ 2.N) / 2)

REPRÉSENTATION BINAIRE DE L'INFORMATION

3.3- Données non numériques:

Les données non numériques correspondent aux caractères alphanumériques et aux caractères spéciaux, c'est à dire les lettres de l'alphabet (A, B C,...., Z), les chiffres (0, 1,...., 9) et le autres symboles (?, ! § ...etc).

Le codage est fait par un tableau de correspondance, propre à chaque code utilisé. Parmi le plus connus on peut citer les codes:

3.3.1 .BCD

3.3.2. ASCII

3.3.3. EBCDIC

3.3.1- BCD:

(**B**inary **C**oded **D**ecimal), un caractère est codé sur 6 bits;

3.3.2- ASCII:

(**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) codé sur 7 bits.

3.3.3- EBCDIC:

(**E**xtended **B**inary **C**oded **D**ecimal **I**nternal **C**ode) codé sur 8 bits.

3.4- *Exercice*

EXERCICE 1:

1- Convertir les nombres décimaux suivants en nombres binaires, octal puis hexadécimal:

12; 24; 192; 0,25; 17,15.

2- Convertir les nombres binaires en décimaux:

1011; 10110; 1011011.

EXERCICE 2:

Trouver la relation de récurrence pour convertir un nombre N de la base b à la base b^k et inversement.

Vérifier cette relation pour la conversion des nombres de l'exercice 1.

EXERCICE 3:

Ecrire les nombres décimaux suivants en binaire dans la représentation "module plus signe", avec un format de huit éléments binaires.

+48; -48; +17; -24; -15.

Retrouver ces valeurs en utilisant le complément à 2.

EXERCICE 4:

Ecrire les nombres décimaux suivants en BCD.

+48; -48; -157; +103; -124.

EXERCICE 5:

A l'aide des interrupteurs, réaliser les circuits électriques pour allumer une lampe qui fonctionne suivant les équations logiques suivantes:

L1 = A.B

L2 = A+B

L3 = A.B + A.C

L4 = A.B + C.D

Logique combinatoire	Auteur
Systèmes logiques combinatoires	Ecole polytechnique fédérale de Lausanne
La logique combinatoire	Robert Papanicola
Logique combinatoire	Rémi Lechartier
Cours sur la logique combinatoire	Gilles Bouvier et Génaël Valet
Les systèmes combinatoires	Yvan Crévits
Logique combinatoire avec des chronogrammes intéractifs en javascript	Nicolas Midoux
Encodage et Décodage	NeT_TroniquE
Multiplexage et Démultiplexage	NeT_TroniquE
Affichage Numérique	NeT_TroniquE
Additionneur & Soustracteur	netProblèMATHique
Circuits Combinatoires	Stéphane Martin

1- postulats et théorème de l'agébre de boole :

Georges BOOLE : Mathématicien britannique né à LINCOLN (1815-1864). Un des promoteurs de la logique mathématiques contemporaine. En 1854 Boole écrivit un article intitulé "Investigation parmi les lois de pensée", les mathématiques classiques quoique extrêmement utiles dans la recherche intellectuelle ne peuvent prendre en compte tous les aspects de la pensée, donc Boole a posé les fondement d'une nouvelle algèbre appela "Algèbre des classes". En 1938 la technique téléphonique atteignait un haut degré de complexité, et les méthodes de Boole furent soudainement redécouvertes grâce à un article publié par C.E. Shannon et intitulé " Analyse symbolique des circuits de commutation à relais". Shannon découvrit que l'algèbre des classes de Boole était un outil

puissant, qui permettait d'analyser et de représenter les circuits complexes, basés sur un fonctionnement à deux états.

1-Variable binaire

3.1.1- Définition

Une variable binaire est une variable qui ne peut prendre que deux états.

Exemple : On considère le schéma électrique suivant

L'interrupteur « a » peut être soit fermé, soit ouvert. Il possède donc deux états possibles de fonctionnement.

La lampe S possède également deux états possibles de fonctionnement qui sont éteinte, allumée.

On peut donc dire que les variables a et S sont des variables binaires.

Etats logiques

Les deux états que peuvent prendre une variable binaire sont appelés des états logiques.

Un état logique prendra la valeur binaire 0 ou 1

Dans l'exemple précédent, on peut écrire que lorsque :

L'interrupteur « a » est ouvert ® a = 0

L'interrupteur « a » est fermé ® a = 1

La lampe S est éteinte ® S = 0

La lampe S est allumée ® S = 1

On peut imaginer d'autres situations de variables binaires.

1- postulats et théorème de l'agébre de boole :

0.1.1.Postulats

L'algèbre de Boole est un ensemble quelconque d'éléments E, muni de Trios lois de composition suivantesL Addition booléenne, Produit booléen, Complémentation.

Addition booléenne	notée: + (ou V)	appelé OU
Produit booléen	noté : . (ou Λ)	appelé ET
Complémentation	noté : " $\bar{}$ "	appelé NON

Les éléments de E satisfont aux propriétés suivantes:

1.1.1. Inversions:

L'inverse de l'inverse d'une fonction est égal à la fonction elle-même.

$$\overline{\overline{A}} = A$$

1.1.2 Commutativité:

Les fonctions OU et ET sont commutables par rapport à chacune des variables d'entrées:

- Fonction OU: A **OU** B = B **OU** A, A **V** B = B **V** A, A **+** B = B **+** A
- Fonction ET: A **ET** B = B **ET** A, A Λ B = B Λ A, A **.** B = B **.** A

1.1.3 Associativité:

Les fonctions OU et ET sont associatives:

- **(A + B) + C = A + (B + C) = A + B + C**
- **(A . B) . C = A . (B . C) = A . B . C**

1.1.4 Distributivité:

La fonction ET est distributive par rapport à la fonction OU:

- **A . (B + C) = A . B + A . C**

La fonction OU est distributive par rapport à la fonction ET:

- **A + (B . C) = (A + B) . (A + C)**

1.1.5 Absorptions:

- **A . (A + B) = A**
- **A + (A . B) = A**

1.1.6 Association d'une fonction avec son complément:

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

1.1.7 Elements neutres:

- L'élément neutre pour l'addition est 0 : **A + 0 = A**
- L'élément neutre pour le produit est 1: **A . 1 = A**

1.1.8 Elément nul:

- L'élément nul pour l'addition est 1 : **A + 1 = 1**
- demostration $A+1=(A+1).1=(A+1).(A+\overline{A})=A+\overline{A}$
- L'élément nul pour le produit est 0 : **A . 0 = 0**

1.1.9 Idem potence ou relation d'une variable avec elle même:

- **A + A = A**
- **A . A = A**

1.1.10 Théorèmes de Morgan:

- Complémentation d'un produit logique :

Le complément d'un produit logique est égal à la somme logique des facteurs complémentés de ce produit.

- $\overline{A+B}=\overline{A}.\overline{B}$
- Complémentation d'une somme logique :

Le complément d'une somme logique est égal au produit des termes complémentés de cette somme.

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

Postulats et théorème de l'aLgébre de boole :

0.1.2.Théorèmes de l'algèbre de Boole:

- Lois d'absorption:

A +A.B = A

Démonstration : A + A.B = A.1 + AB = A . (1 +B) = A. 1 = A

A.(A+B) = A

Démonstration : A(A +B) = A.A + A.B = A + A.B = A

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

Démonstration : $A+\overline{A}B=(A+\overline{A}).(A+B)=1.(A+B)=A+B$

- Consensus:

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

BC est le consensus par rapport à A des termes A.B et $\overline{A}C$. On peut ajouter le consensus l'expression ne change pas.

Démonstration:

Postulats et théorème de l'aLgébre de boole :

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

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

Démonstration:

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

- ***Théorème de Morgan:***

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

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

Démonstration:

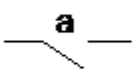
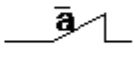
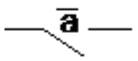
$$A+\overline{A}=1$$

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

2- La représentation symbolique des éléments de logique :

Schéma électrique à contacts :

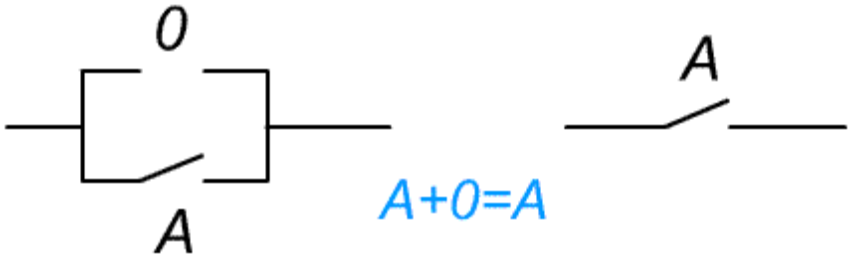
Pour mieux comprendre l'opération logique réalisée par un opérateur logique, nous représenterons le schéma électrique à contact dont le fonctionnement est équivalent. Un contact représente par ses deux positions les deux états d'une variable d'entrée.

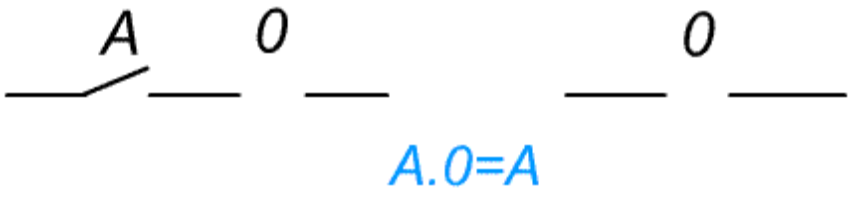
	a	$\bar{a}$
Etat logique	0	1
Contact au repos		
Etat logique	0	1
Contact au travail		

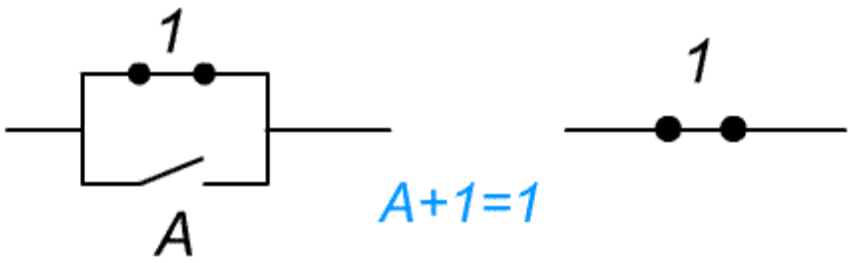
Par convention, nous représentons toujours des contacts au repos.

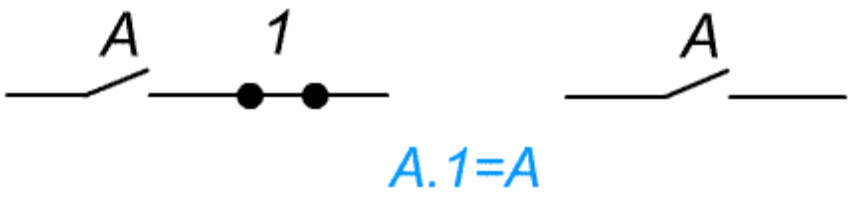
Exemples

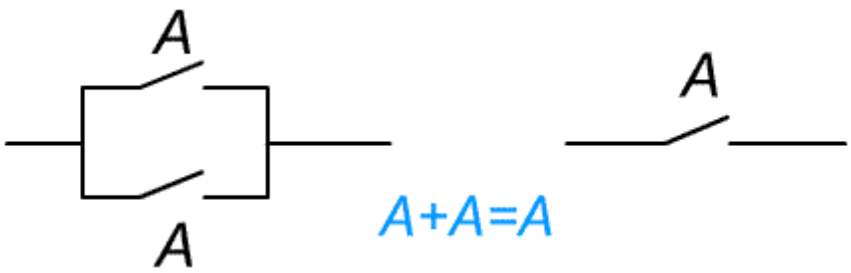
2- La représentation symbolique des éléments de logique :











Simulation Simulation

3- Fonctions élémentaires de la logique combinatoire:

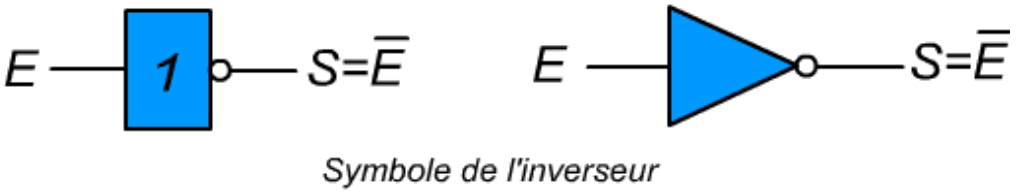
3.1-Fonctions simples:

Pour chacune des fonctions simples nous donnons la définition, le symbole associe et la table de vérité.

3.1.1.Fonction NON ou inverse:

La fonction non ou fonction complimentée ou fonction inverse

3- Fonctions élémentaires de la logique combinatoire :



Symbole de l'inverseur

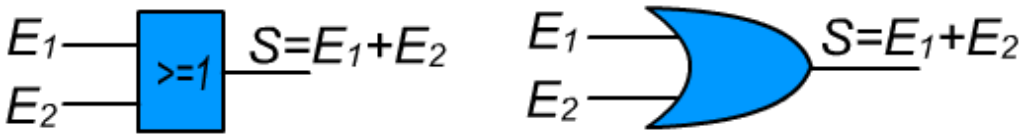
Simulation

Table de vérité

E	S
0	1
1	0

3.1.2.Fonction OU (OR):

Une affirmation apparaît en sortie si le signal est affirmatif sur au mois une des entrées. On représente l'opérateur OU à deux entrées comme le montre la figure ci dessous.



Symbole de l'opérateur OU

Simulation

Table de vérité

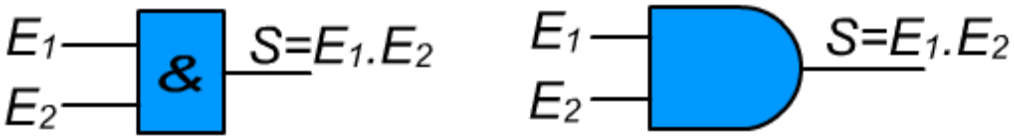
E_1	E_2	S
0	0	0
0	1	1
1	0	1
1	1	1

La sortie vaut 1, lorsque l'un au moins des signaux d'entrée E1 ou E2 vaut 1. L'équation Booléenne de l'opérateur Ou est alors:

S = E1 OU E2 = E1 V E2 = E1 + E2

3.1.3.Fonction ET (and):

Une affirmation apparaît en sortie lorsque chacun des signaux appliqués sur les bornes d'entrée est affirmatif.



Symbole de l'opérateur ET

Simulation

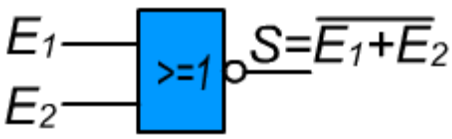
Table de vérité

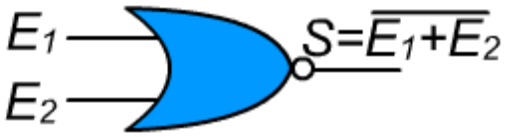
E_1	E_2	S
0	0	0
0	1	0
1	0	0
1	1	1

3.2-Fonctions logiques complètes:

3.2.1.Fonction OU - NON (Nor):

La fonction NOR est la fonction inverse de la fonction OR. La sortie vaut 0, lorsque l'un au moins des signaux d'entrée vaut 1, donc en déduit facilement sa table de vérité.





Symbole de l'opérateur NOR

Simulation

Table de vérité

E_1	E_2	S
0	0	1
0	1	0
1	0	0
1	1	0

L'opérateur NOR est un opérateur complet puisqu'on peut réaliser facilement toutes les fonctions simples

- réalisation de la fonction inverse:

relier les deux entrées ensembles

mettre toujours à 0 l'une des entrées

- réalisation de la fonction ET

Il suffit inverser les deux entrées par deux portes NOR montés en inverseur

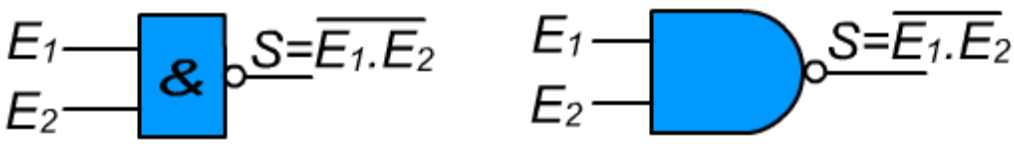
- réalisation de la fonction OU

Il suffit d'inverser la sortie de la porte NOR par une deuxième porte NOR

montée en inverseur.

3.2.2- Fonction ET -NON (NAND):

La fonction NAND est la fonction inverse de la fonction AND.



Symbole de l'opérateur NAND

Simulation

Table de vérité

E_1	E_2	S
0	0	1
0	1	1
1	0	1
1	1	0

L'opérateur NAND est un opérateur complet puisqu' on peut réaliser facilement toutes les fonctions simples

- réalisation de la fonction inverse:

relier les deux entrées ensembles

mettre toujours à 1 l'une des entrées

- réalisation de la fonction OU

Il suffit inverser les deux entrées par deux portes NAND montés en inverseur

- réalisation de la fonction ET

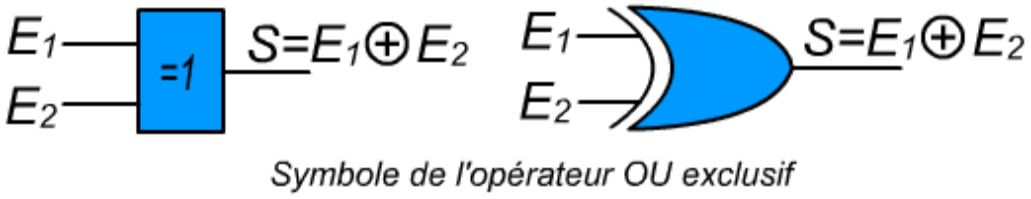
Il suffit d'inverser la sortie de la porte NAND par une deuxième porte NAND

montée en inverseur.

3.3-Fonctions complexes:

3.3.1- Fonction Ou exclusif:

La fonction OU exclusif est une fonction de deux variables uniquement qui prend la valeur 1 si une seule variable est égale à 1. Donc la fonction vaut 1 lorsque les valeurs de deux variables d'entrées sont différentes, c'est à dire (E1=0 et E2=1) ou (E1=1 et E2=0).



Symbole de l'opérateur OU exclusif

Simulation

Table de vérité

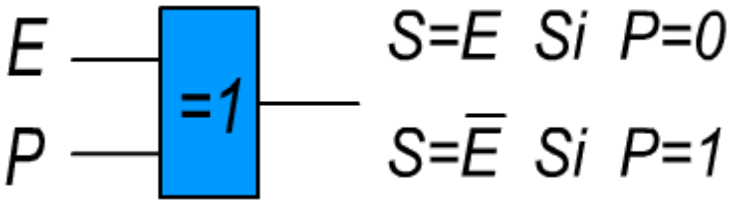
E_1	E_2	S
0	0	0
0	1	1
1	0	1
1	1	0

Les propriétés de la fonction ou exclusif sont:

- **commutativité: A⊕ B = B⊕ A**
- **associativité**: (A⊕ B)⊕ C =A⊕ (B⊕ C) = A⊕ B⊕ C
- **élément neutre 0**: A⊕ 0 = A

Le complément de la fonction OU exclusif est égal au Ou exclusif des variables dont l'une est complémentée: NON (A ⊕ B) = NON(B) ⊕ A = B ⊕ NON(A)

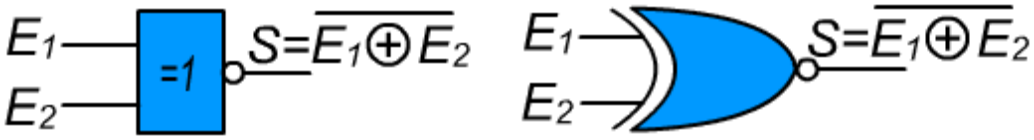
L'opérateur Ou exclusif est l'opérateur **programmable**, suivant la valeur de la commande P la relation entre la sortie et l'entrée est différente.



L'opérateur programmable suivant la valeur de P.

3.3.2- Le comparateur logique:

Le comparateur logique ou la fonction coïncidence ou identité est la fonction complémentaire de la fonction OU exclusif.



Symbole de l'opérateur comparateur logique

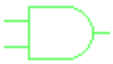
Table de vérité

E_1	E_2	S

3- Fonctions élémentaires de la logique combinatoire :

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

Les différentes portes logiques

SYMBOLE (Norme MILSTD 086B)	**NOM ET ÉQUATION**	**SYMBOLE** (notation française)
	INVERSEUR	
	ET s = a.b	
	OU s = a+b	
	NON-ET	

3- Fonctions élémentaires de la logique combinatoire :

	NON-OU	
	OU EXCLUSIF	
	NON-OU EXCLUSIF	

Des exemples d'exercices interactifs réalisés par JC Michel.free.fr

Exercice de simulation (Les différentes portes logiques)

Exercice de simulation (Ou exclusif)

Simulation

logiciel de simulation

dessiner interconnexion couper exécution

1- Dessiner le logigramme en Choisissant les portes logiques

2- Etablir les connexions

3- Lancer la simulation

Logique combinatoire	Auteur
Systèmes logiques combinatoires	Ecole polytechnique fédérale de Lausanne
La logique combinatoire	Robert Papanicola
Logique combinatoire	Rémi Lechartier
Cours sur la logique combinatoire	Gilles Bouvier et Génaël Valet
Les systèmes combinatoires	Yvan Crévits
Logique combinatoire avec des chronogrammes intéractifs en javascript	Nicolas Midoux
Encodage et Décodage	NeT_TroniquE
Multiplexage et Démultiplexage	NeT_TroniquE
Affichage Numérique	NeT_TroniquE
Additionneur & Soustracteur	netProblèMATHique
Circuits Combinatoires	Stéphane Martin

1- les Représentations d'une fonction logique

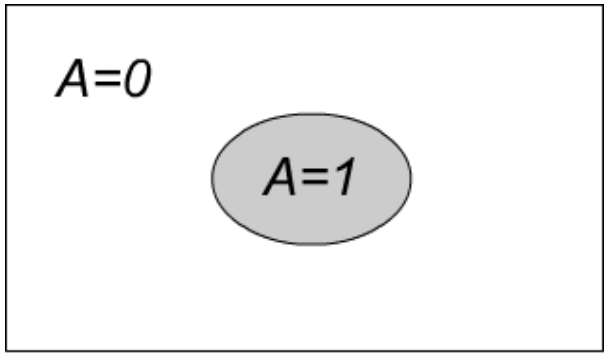
1.1- Variables et fonctions logiques:

Une variable logique ou binaire, que nous notons A, est une grandeur qui ne peut prendre que deux états notés 0 ou 1 et ne peut pas varier de façons continue.

Une fonction logique $F(x_{1,x2}...,x_{n)}$ de n variables $x_1x_2...x_n$ est une fonction qui ne prend, comme chacune des variables, que deus valeurs 0 ou 1.

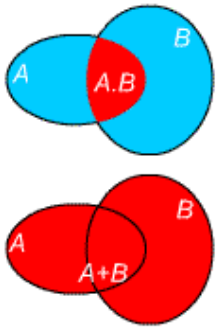
1.2- Représentation VENN-EULER:

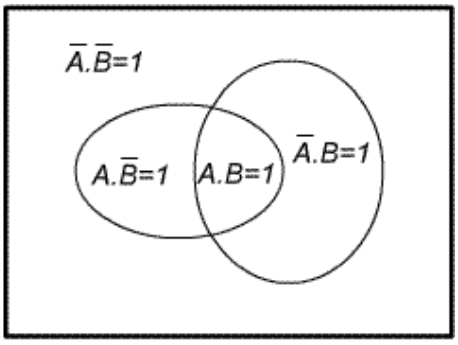
Le logicien anglais Venn a élaboré des diagrammes logiques. Il représente la variable par un domaine, l'espace extérieur à ce domaine est alors représentatif de la variable complémentée. Soit A une variable binaire, A peut prendre que l'un de deux états possibles que l'on représente par 0 ou 1. On convient que A prend la valeur 1 à l'intérieur du domaine et la valeur 0 à l'extérieur voir figure.



- ***Cas de deux variables A et B:***

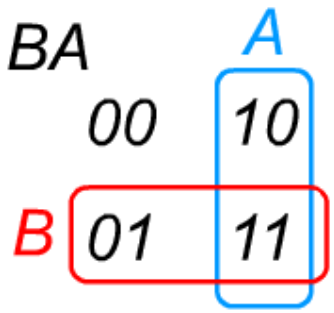
Dans le cas de deux variables on trace deux domaines l'un représente A et l'autre représente B. L'intersection des 2 domaines représente le produit logique A.B et la réunion des 2 domaines représente la somme logique A+B.



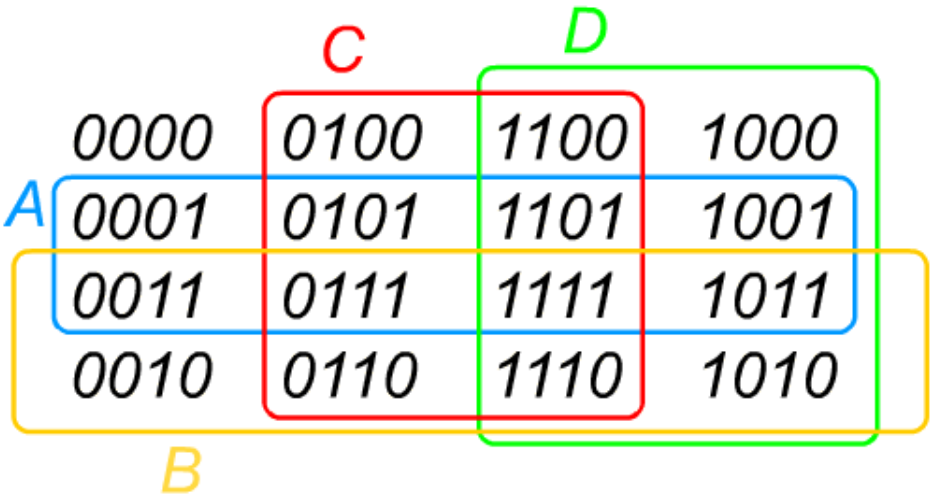


On peut encore représenter plus facilement le diagramme de Venn en correspondant à chaque zone du diagramme une ligne de la table de vérité. Pour avoir une représentation de cette fonction qu'il suffit d'indiquer dans chaque zone la valeur de la fonction (0 ou 1).

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



- ***Cas 4 variables:***



Le développement de cette représentation à l'univers booléen mène aux notions de min-termes et max-termes.

Un min-terme est donc représenté par un produit logique comportant tous les termes de base sans exception sous leur forme vraie ou leur forme complémentée.

Un max-terme est donc représenté par la somme logique comportant tous les termes de base sans exception sous leur forme vraie ou leur forme complémentée.

min-termes et max-termes

pour 3 variables

Valeur			*Min-termes*		*Max-termes*	
A	*B*	*C*	*Écriture*	*Notion*	*Écriture*	*Notion*

0	0	0	$\overline{A}.\overline{B}.\overline{C}$	m_0	$\overline{A}+\overline{B}+\overline{C}$	M_0
0	0	1	$\overline{A}.\overline{B}.C$	m_1	$\overline{A}+\overline{B}+C$	M_1
0	1	0	$\overline{A}.B.\overline{C}$	m_2	$\overline{A}+B+\overline{C}$	M_2
0	1	1	$\overline{A}.B.C$	m_3	$\overline{A}+B+C$	M_3
1	0	0	$A.\overline{B}.\overline{C}$	m_4	$A+\overline{B}+\overline{C}$	M_4
1	0	1	$A.\overline{B}.C$	m_5	$A+\overline{B}+C$	M_5
1	1	0	$A.B.\overline{C}$	m_6	$A+B+\overline{C}$	M_6
1	1	1	$A.B.C$	m_7	$A+B+C$	M_7

Selon les formules générales données ci dessous, le complément d'un min-terme est un max-terme et vice-versa.

$$\overline{m_i}=M_{2^n-1-i}$$

$$\overline{M_i}=m_{2^n-1-i}$$

- ou i est l'indice notant le rang de M ou m,

- 2^{n-1} est la valeur max de l'indice i

- n est le nombre de variables

On remarque que m_2 et M_5 sont complémentaire.

1.3- Représentation d'une fonction par son équation logique:

La représentation d'une fonction peut être définie par son équation logique par une suite de termes:

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

1.4- Représentation d'une fonction par sa table de vérité:

La représentation d'une fonction peut être définie par sa table de vérité:

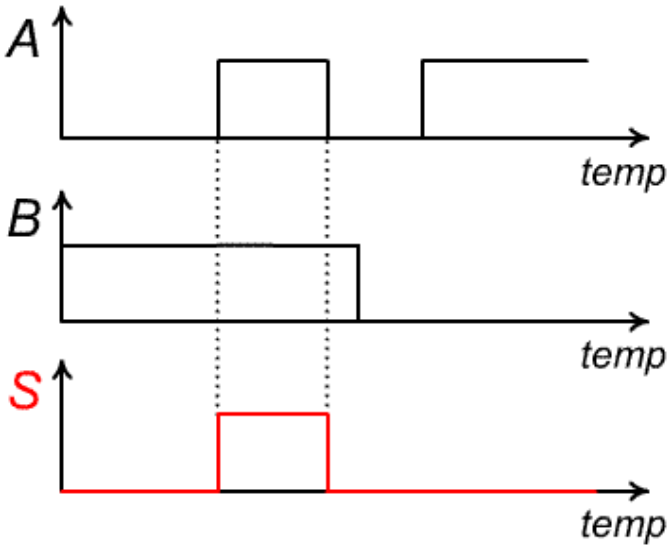
Décimale	d	c	b	a	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0

6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

1.5- Représentation d'une fonction par logigramme:

1.6- Représentation d'une fonction par chronogramme:

Le chronogramme : c'est le graphe de l'évolution temporelle des variables et des fonctions logiques.



Chronogramme d'une fonction ET

2- Représentation de veitch et karnaugh:

2.1- Représentation graphique:

Il s'agit d'une extension de celle de Venn. Dans ce diagramme on représente chaque zone du plan ci-dessous par un carré, sous la forme du tableau ci-dessous.

2.1.1- Cas de 2 variables:

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

Exemple représentation de la fonction OU exclusif:

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

B\A	0	1
0	0	1

1	1	0

2.1.2- Cas des 3 variables:

Veitch: Code binaire naturel

C\BA	00	01	10	11
0				
1				

Karnaugh: Code Gray

C\BA	00	01	11	10
0				
1				

Remarques:

Chaque case de la table de Karnaugh ou de Veitch représente une des 2^n combinaisons des n variables de la fonction. Cette combinaison est désignée par les coordonnées de la case.

La seule différence en le diagramme de Karnaugh par rapport au diagramme de Veitch est l'utilisation du code Gray. Lorsqu'on passe d'une case à la voisine dans le cas du diagramme de Karnaugh (les cases adjacentes), il n'y a qu'une seule variable qui change. En effet, chaque frontière de la table représente le changement d'une variable et

d'une seule. Il a lieu de noter l'adjacente des cases de la première et la dernière ligne, de même de la première et la dernière colonne.

2.1.3- Cas des 4 variables:

On n'utilise que le diagramme de Karnaugh

DC\AB	00	01	11	10
00				
01				
11				
10				

2.1.4- Cas des 5 variables:

Dans le cas de 5 variables, on a le choix entre représentation par un seul tableau ou bien par deux tableaux.

- Un seul tableau:

Dans ce cas on vérifie facilement les cases adjacentes comme pour les cas précédents.

ED\CBA	000	001	011	010	110	111	101	100
00								
01								

2- Représentation de veitch et karnaugh:

11								
10								

- ***Deux tableaux:***

ED\CB	00	01	11	10
00				
01				
11				
10				

Tableau A

ED\CB	00	01	11	10
00				
01				
11				
10				

Tableau $\overline{A}$

Entre les deux tableaux, les cases situées à des emplacements identiques sont adjacentes.

2- Représentation de veitch et karnaugh

2.2- *Représentation de karnaugh d'une fonction:*

La représentation d'une fonction peut être définie par:

- Par une suite de termes:

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

- Par sa table de vérité:

Décimale	*d*	*c*	*b*	*a*	*F*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1

2- Représentation de veitch et karnaugh:

12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

A chaque case de la table de Karnaugh, on associe le nombre décimal correspondant à son équivalent en code binaire naturel (DCBA). Numérotation décimale des cases dans le diagramme de karnaugh:

Par convention, le nombre décimal équivalent à une case associe au nombre binaire exprimé par DCBA est égale : $D.2^3+C.2^2+B.2^1+A.2^0$.

Exemples

Pour les entrées DCBA qui vaut respectivement 0000 on associe la case 0

Pour les entrées DCBA qui vaut respectivement 0111 on associe la case 7

Pour les entrées DCBA qui vaut respectivement 1110 on associe la case 14

DC\BA	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Remplissage de la table de Karnaugh d'une fonction F:

Nous remarquons que chaque case correspond à une combinaison des variables des entrées. On remplie chaque

2- Représentation de veitch et karnaugh:

case par la valeur de la fonction; c'est à dire, on remplie les cases associes à la fonction par 1 quand la valeur de la fonction est égale à 1, les autres cases sont remplies par 0.

DC\BA	00	01	11	10
00	**1** 0	**0** 1	**0** 3	**1** 2
01	**0** 4	**0** 5	**0** 7	**0** 6
11	**0** 12	**0** 13	**1** 15	**0** 14
10	**0** 8	**0** 9	**1** 11	**0** 10

L'expression de de la fonction logique F est:

$$F=\sum_4(0,2,11,15)$$

3- Expression d'une fonction:

3.1- Formes canoniques d'une fonction logique:

Une forme est dite canonique quand toutes les variables constituant le mot d'entrée apparaissent dans les termes exprimant la valeur de la fonction. Il existe deux formes canoniques pour une fonction donnée, appelée 1ére forme canonique et 2éme forme canonique.

3.2- Relation de Shannon dans le cas d'une variable:

Soit f(x) fonction de la variable booléenne; on peut écrire:

$$f(x)= x.f(1)+\bar{x}.f(0) \qquad (1)$$

En vérifie facilement que:

pour x=0 f(0) = 0.f(1) +1.f(0) =f(0)

pour x=1 f(1) = 1.f(1) + 0.f(0) = f(1)

Si la fonction f(x) est exprimée sous cette formule, somme de produits, $\sum(\Pi)$ on dit que la fonction est exprimée sous la première forme canonique. A partir de (1), on peut écrire:

$$\bar{f}(x)=x\,.\,\bar{f}(1)+\bar{x}\,.\,\bar{f}(0)$$

En applique le théorème de Morgan, il vient:

$$f(x)=(x+f(0)).(\bar{x}+f(1)) \qquad (2)$$

Si la fonction f(x) est exprimée sous cette formule sous cette forme, produit de sommes, $\Pi(\Sigma)$ on dit que la fonction est exprimée sous **la seconde forme canonique.**

3.3- Relation de Shannon dans le cas de deux variables:

Soit f(x,y) fonction de 2 variables booléennes indépendantes x et y; on peut écrire:

- ***première forme canonique:***

$$f(x,y) = x.f(1,y) + \bar{x}.f(0,y)$$

avec

$$f(1,y) = y.f(1,1) + \bar{y}.f(0,0)$$

$$f(0,y) = y.f(0,1) + \bar{y}.f(0,0)$$

$$f(x,y) = x.yf(1,1) + x\bar{y}.f(1,0) + \bar{x}.yf(0,1) + \bar{x}\bar{y}.f(0,0)$$

On voit donc qu'il est possible d'écrire une fonction sous la forme d'une somme de produits logiques. En remarquant qu'un terme n'existe que si la valeur correspondante de f(i,j) = 1.

- ***seconde forme canonique:***

$$f(x,y) = [x + f(0,y)].[\bar{x} + f(1,1)]$$

avec

$$f(0,y) = [y + f(0,0)].[\bar{y} + f(0,1)]$$

$$f(1,y) = [y + f(1,0)].[\bar{y} + f(1,1)]$$

d'ou

$$F(x,y) = \{x + [y + f(0,0)].[\bar{y} + f(0,1)]\}.\{\bar{x} + [y + f(1,0)].[\bar{y} + f(1,1)]\}$$

$$= [x + y + f(0,0)].[x + \bar{y} + f(0,1)].[\bar{x} + y + f(1,0)].[\bar{x} + \bar{y} + f(1,1)]$$

3- Expression d'une fonction:

Exemple:

x	y	f(x,y)
0	0	0
0	1	1
1	1	1
0	1	0

- ***première forme canonique:***

$$f(x,y)=\bar{x}.y.f(0,1)+x.\bar{y}.f(1,0)=\bar{x}.y+x.\bar{y}$$

- ***seconde forme canonique:***

$$\begin{aligned} f(x,y) &= [x+y+f(0,0)].[x+\bar{y}+f(0,1)].[\bar{x}+y+f(1,0)].[\bar{x}+\bar{y}+f(1,1)] \\ &= [x+y+0].[x+\bar{y}+1].[\bar{x}+y+1].[\bar{x}+\bar{y}+0] \\ &= [x+y].[\bar{x}+\bar{y}] \end{aligned}$$

Généralisation

Il faut développer par rapport aux 1 puis par rapport aux 0.

En pratique, on déduit directement l'expression de F à partir de la table de vérité comme suit:

3- Expression d'une fonction:

a)somme de produits

On considère que les lignes ouF= 1

Dans ces lignes

une variable égale à zéro, on fait correspondre son complément,

une variable égale à un, on fait correspondre elle-même .

Pour chaque ligne, on écrit le produits logique correspondant, puis on termine en faisant la somme de ces produits.

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

3- Expression d'une fonction:

1	1	0	1
1	1	1	0

$$F=\bar{x}.\bar{y}.z+\bar{x}.y.\bar{z}+x.\bar{y}.z+x.y.\bar{z}$$

Une forme apparaît dans les ouvrages est une représentation par association à chaque mi-terme sa valeur soit binaire, soit décimale.

Forme binaire :

F= 001+010+101+110

Pour simplifier la représentation des fonctions, on fait la conversion binaire décimale. On doit obligatoirement fixer l'ordre des variables x, y et z (c.a.d le poids des bits dans la base 2: le poids de x est 2, le poids de y est 1 et de poids de z est 0). Donc on peut écrire la fonction comme suite:

Forme décimale:

F= 1+2+5+6

$F = \sum 3(1,2,5,6)$

Notation plus condensées

$F = \Re (1,2,5,6)$

b) produit de sommes:

On considère que les lignes ouF= 0

Dans ces lignes

3- Expression d'une fonction:

une variable égale à zéro, on fait correspondre elle-même ,

une variable égale à un, on fait correspondre son complément.

Pour chaque ligne, on écrit la somme logique correspondant, puis on termine en faisant le produit de ces sommes.

x	y	z	F(x,y, z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

3- Expression d'une fonction:

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

La représentation par l'association décimale.

$$\prod_3(7,4,3,0)$$

Remarque:

En pratique, une série de circuits intégrés ne comprend qu'un nombre limité de différents types de portes. En général, seules les portes NAND et NOR sont utilisées.

- ***Porte NAND:***

Pour réaliser une fonction logique par des portes NAND , il faut de faire deux complémentations (négations) successives de la fonction exprimée sous la première forme canonique, ou sous la deuxième forme canonique.On note la porte NAND à deux entrées a et b par :

$$a/b=\overline{a.b}=\overline{a+b}$$

Exemples:

Ex1:
$$\begin{aligned} F1&=(a+b).(c+tp)\bar{d} \\ &=\overline{\overline{F1}} \\ &=\overline{\overline{(a+b).(c+tp)\bar{d}}} \\ &=\overline{\overline{(\bar{a}+\bar{b}).(\bar{c}+\overline{tp})\bar{d}}} \\ &=(\bar{a}/\bar{b})/(\bar{c}/(t/p))/\bar{d} \end{aligned}$$

Ex2:
$$\begin{aligned} F2&=\overline{\overline{a.b}+\overline{ctp}+\bar{d}} \\ &=a/b/c/t/p/\bar{d} \end{aligned}$$

- ***Porte NOR:***

3- Expression d'une fonction:

Pour réaliser une fonction logique par des portes NAND , il faut de faire deux complémentations (négations) successives de la fonction exprimée sous la première forme canonique, ou sous la deuxième forme canonique.On note la porte NOR à deux entrées a et b par:

$$a \downarrow b = \overline{a+b} = \overline{a}\,\overline{b}$$

Exemples:

Ex1: $F1=(a+b).(c+tp)\overline{d}=\overline{\overline{(a+b)}.\overline{(c+\overline{t}.\overline{p})}}\,\overline{d}=(a\downarrow b)\downarrow(c\downarrow(\overline{t}\downarrow\overline{p})\downarrow d$

Ex2: $F2=ab+ctp+\overline{d}=\overline{\overline{ab}+\overline{ctd}}+\overline{d}=(\overline{a}\downarrow\overline{b})\downarrow(\overline{c}\downarrow\overline{t}\downarrow\overline{p})\downarrow\overline{d}$

Représentation de Karnaugh d'une fonction:

La représentation d'une fonction peut être définie par:

- Par une suite de termes:

$$F=\overline{a}\,\overline{b}\,\overline{c}\,\overline{d}+a\,b\,\overline{c}.d+\overline{a}\,b\,\overline{c}\,\overline{d}+abcd$$

- Par sa table de vérité:

Décimale	*d*	*c*	*b*	*a*	*F*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1

3- Expression d'une fonction:

3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

$F = \sum_4(0,2,11,15)$ la première forme canonique

Table de Karnaugh de la fonction F

dc\ba	00	01	11	10

3- Expression d'une fonction:

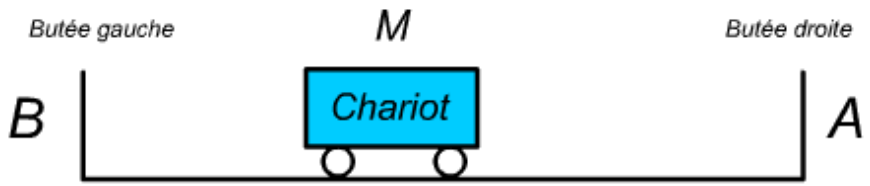
00	**1** 0	**0** 1	**0** 3	**1** 2
01	**0** 4	**0** 5	**0** 7	**0** 6
11	**0** 12	**0** 13	**1** 15	**1** 14
10	**0** 8	**0** 9	**0** 11	**0** 10

$F=\prod_4(1,3,4,5,6,7,8,9,10,12,13,14)$ la deuxième forme canonique

4- LES Fonctions f booléennes

Les fonctions fbooléennes sont des fonctions non définies pour certaines valeurs des variables d'entrées comme le montre l'exemple suivant:

Soit un chariot, animé par un moteur M à double sens de rotation(Droite, Gauche). Le chariot se déplace entre deux butées A et B en fonction de l'état du moteur M (marche ou arrêt). Déterminer l'équation logique du moteur M en fonction des boutons de fin de cours A et B.



M =1 quand le moteur fonctionne

M = 0 quand le chariot se trouve en A ou en B

M = 1 quand le chariot n'est pas en A ou en B

On se propose de chercher la fonction logique du moteur M.

Si le chariot se trouve en butée gauche c'est à A= 0 te B= 1 donc M=0

Si le chariot se trouve en butée droite c'est a A= 1 te B= 0 donc M=0

Si le chariot se trouve ni en butée gauche et ni en butée droite avec A= 0 et B= 0 donc M=1

Si le chariot se trouve à la fois en butée gauche et en butée droite avec A= 1 et B= 1, quelle est la valeur de M ? Comme cette situation n'est jamais réalisée on peut attribuer pour M n'importe quelle valeur 1 ou 0. Dans ce cas, il est préférable de mettre de ***f .*** Le ***f*** peut être un 0 et un 1.

B\A	0	1

0	1	0
1	0	***f***

En fonction des contraintes technologiques on attribue à ***f***

la valeur 0 ou 1

si ***f*** = 0 l'équation de $M=\overline{A}\,\overline{B}=\overline{A+B}$

si ***f*** = 1 l'équation de $M=A\otimes B$

7- SIMPLIFICATION DES FONCTIONS:

7.5-Simplification par les méthodes algébrique:

Cette méthode consiste à appliquer les principes de l'algèbre de Boole, et tout particulièrement

- en regroupant des termes à l'aide des identités remarquables

- En ajoutant des termes déjà existant

- En supprimant un terme superflu (consensus)

- en choisit par fois la fonction complémentée si elle a un minimum des termes.

Exemples:

Exercice 1:

$$
\begin{aligned}
F1 &= (\bar{a}+\bar{b}).(\bar{a}+b).(a+\bar{b}) \\
&= (\bar{a}+\bar{b}).(\bar{a}.a+\bar{a}\bar{b}+a.b+b\bar{b}) \\
&= (\bar{a}+\bar{b}).(\bar{a}\bar{b}+a.b) \\
&= \bar{a}\bar{a}\bar{b}+\bar{b}\bar{a}\bar{b}+\bar{a}.a.b+\bar{b}.a.b \\
&= \bar{a}\bar{b}
\end{aligned}
$$

Exercice 2:

$$F2=a.b.c+a.\overline{b}.c+a.b\overline{c}+a\overline{b}\overline{c}$$
$$=a.b(c+\overline{c})+a\overline{b}.(c+\overline{c})$$
$$=a.(b+\overline{b})$$
$$=a$$

Exercice 3:

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

Exercice 4:

$$F4=a.b+\overline{a}.c+\overline{b}+\overline{c}$$
$$=a.b+\overline{a}.c+bc+\overline{b.c}$$
$$=1$$

7.6-Simplification par la méthode de Karnaugh:

La méthode de Karnaugh pour la simplification d'une fonction logique est base sur la remarque suivante:

Considérons les 2 termes d'une somme logique:

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

Une variable disparaît par regroupement de deux termes qui contiennent les mêmes variables à l'exception d'une seule qui apparaît sous forme vraie dans un terme et sous forme complémentée dans l'autre. La variable qui disparaît est celle qui apparaissait sous les deux formes.

Remarque:

Il est possible de regrouper les cases par des puissances de 2. C'est à dire par 2, par 4, par 8, cet 2^k. Dans un tableau de Karnaugh de 4 variables les regroupements doivent être en ligne ou en colonne, ou en carré. Il faut utiliser tous les cases qui ont 1 au moins une fois, de même il est recommandé de chercher à regrouper le plus grand nombre possible de cases, car les simplifications obtenues sont plus importantes. Les regroupements peuvent s'entrecroiser et se superposer.

- Lorsque l'on regroupe 2 cases, on ramène à un seul terme les deux termes correspondants de l'équation booléenne. La variable qui change de valeur dans les deux cases n'y figure plus.
- Lorsque l'on regroupe 4 cases, on ramène à un seul terme les quatre termes correspondants de l'équation booléenne. Les deux variables qui changent de valeur dans les quatre cases n'y figurent plus
- Lorsque l'on regroupe 8 cases, on ramène à un seul terme les huit termes correspondants de l'équation booléenne. Les trois variables qui changent de valeur dans les huit cases n'y figurent plus.

Pour une fonction de n variables, un regroupement de 2k cases résultant de k simplifications successives correspond à un terme de (n-k) variables.

Exercice 1:

$$S1 = \bar{a}\bar{b}\bar{c}\bar{d} + a\bar{b}c\bar{d} + ab\bar{c}\bar{d} + a\bar{b}\bar{c}d + \bar{a}\bar{b}\bar{c}d + \bar{a}b\bar{c}d$$

DC\BA	00	01	11	10
00	1	0	0	0
01	0	1	1	0
11	0	1	0	0
10	1	0	0	1

$S1 = \bar{a}\bar{b}\bar{c} + a\bar{b}c + ac\bar{d} + \bar{a}\bar{c}d$

Exercice 2:

$$S2 = \bar{a}\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + ab\bar{c}\bar{d} + \bar{a}b\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}b\bar{c}d + ab\bar{c}d + a\bar{b}dc$$

DC\BA	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

$S1=\overline{c}$

Exercice 3:

$$S3=\overline{a}\overline{b}\overline{c}\overline{d}+a.\overline{b}\overline{c}\overline{d}+a\overline{b}.c\overline{d}+a.b.c.d+\overline{a}\overline{b}\overline{c}.d+a.b\overline{c}.d+a\overline{b}.d\overline{c}$$

DC\BA	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	0
10	1	0	1	1

$S1=\overline{a}.\overline{b}.\overline{c}+a.\overline{b}.\overline{d}+a.b.c+b.\overline{c}.d$

DC\BA	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	0
10	1	0	1	1

$S1=\overline{b}.\overline{c}.\overline{d}+a.c.\overline{d}+a.b.d+\overline{a}.\overline{c}.d$

Logiciel de simplification

http://www.puz.com/sw/karnaugh/index.htm

7.7-Simplification par la méthode de Quine Mc Cluskey:

Plus lourde à appliquer que celle de Karnaugh, cette méthode n'est en générale employée que quand le nombre des variables est important(> 5). Elle présente l'intérêt d'être systématique et donc programmable.

Méthode:

On écrit la fonction sous la 1ére forme canonique, on classe alors les monômes dans un tableau par groupes de combinaisons comportant le même nombre de 1 dans l'expression binaire. Ces groupes prennent le nom de classes. On dresse ainsi le tableau suivant:

Classe 0: C0 pas de 1: valeur numérique décimale: 0

classe 1: C1 un 1: valeur numérique décimale: 1,2,8,16 ...2k

classe 2: C2 deux 1: valeur numérique décimale: 3,5,6,9...

Règles de calcul:

- Un monôme d'une classe ne sera réductible qu'avec un monôme placé dans la classe précédente ou suivante (adjacente).
- un monôme ne sera réductible avec un autre placé dans une classe voisine que s'il existe entre les valeurs numériques décimales correspondantes une différence égale à 2i avec i ∈ N+.
- On établit alors un 2ème tableau dans lequel on regroupe également les monômes adjacents réunis en remplaçant dans leur expression binaire le chiffre supprimé par un tiret.
- On effectue alors une nouvelle réduction entre les monômes des classes du 2ème tableau ce qui donne, selon les mêmes principes, un 3éme tableau et ainsi de suite jusqu'à ce que toutes les réductions possibles aient été effectuées.
- il reste à grouper dans les différents tableaux les monômes sur lesquelles aucune réduction n'est possible.

Liens vers d'autres cours

Logique combinatoire	Auteur
Systèmes logiques combinatoires	Ecole polytechnique fédérale de Lausanne
La logique combinatoire	Robert Papanicola
Logique combinatoire	Rémi Lechartier
Cours sur la logique combinatoire	Gilles Bouvier et Génaël Valet
Les systèmes combinatoires	Yvan Crévits
Logique combinatoire avec des chronogrammes intéractifs en javascript	Nicolas Midoux
Encodage et Décodage	NeT_TroniquE
Multiplexage et Démultiplexage	NeT_TroniquE
Affichage Numérique	NeT_TroniquE
Additionneur & Soustracteur	netProblèMATHique
Circuits Combinatoires	Stéphane Martin

LE CODEUR DÉCODEUR TRANSCODEUR

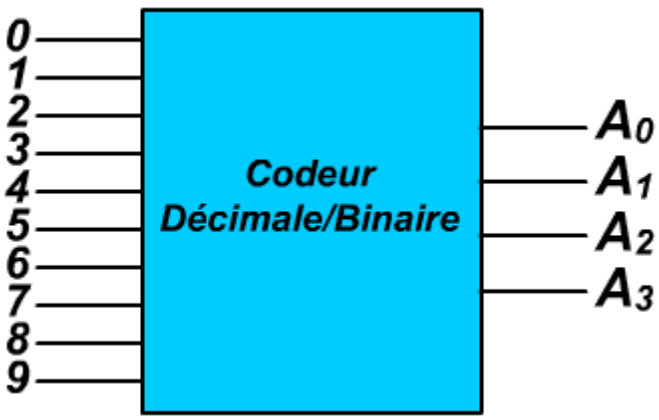
1.1-Le codeur:

1.1.1-Définition:

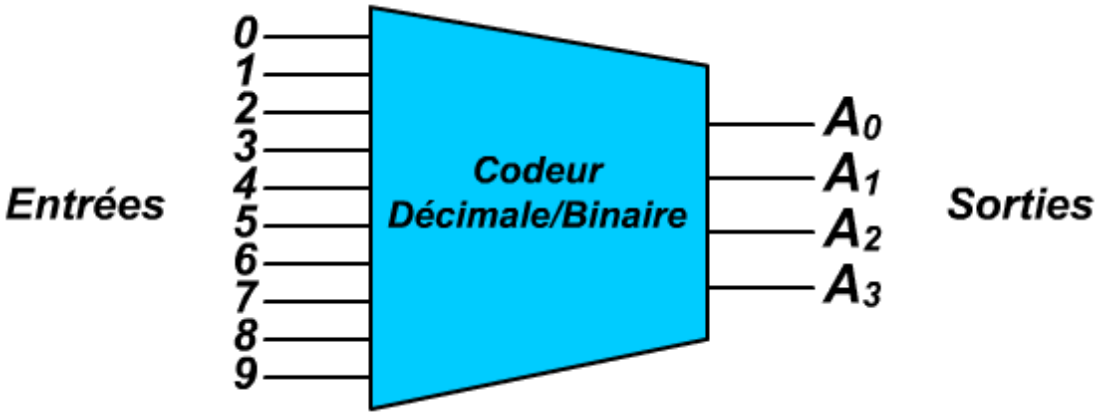
Un codeur est dispositif qui traduit les valeurs de ces entrées dans un code choisi. De façons générales, il y a n sorties et 2^n entrées, mais une seule entrée active à la fois.

1.1.2-Codeur décimal binaire:

Un codeur binaire traduire un certain nombre de chiffres décimaux en binaire.



Le codage des chiffres de 0 à 9 en binaire nécessite 10 entrées et 4 sorties. Le symbole couramment utilisé est le suivant:



Dans sa version la plus simple, un codeur est un ensemble de circuit OU. Soit la table de codage suivante:

Nombre décimale	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Les équations des sorties sous la 1 ère forme canonique:

$$A_0 = \sum (1,3,5,7,9)$$

$$A_1 = \sum (2,3,6,7)$$

$$A_2 = \sum (4,5,6,7)$$

$$A_3 = \sum (8,9)$$

1.1.3-Codeur prioritaire:

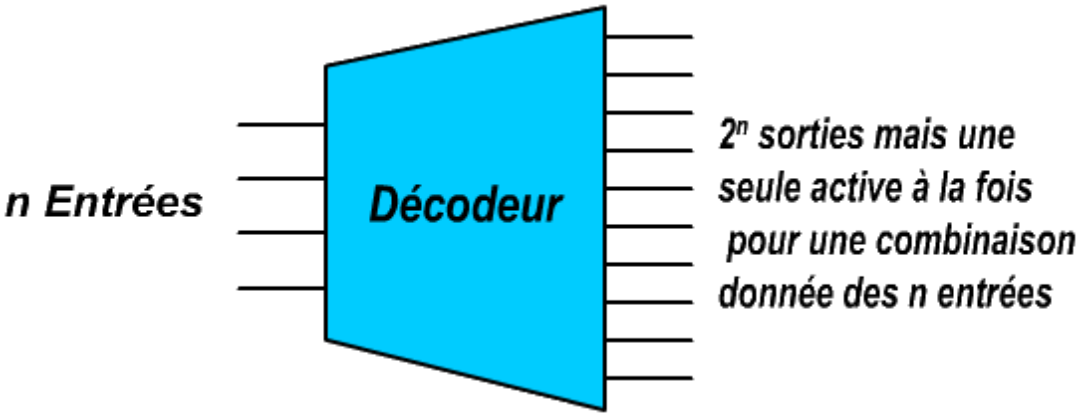
Le codeur prioritaire est un codeur binaire particulier dont voici les caractéristiques: Si maladroitement plusieurs entrées peuvent être actives en même temps, le codeur fera un choix parmi celles-ci. Il va coder le poids le plus élevé, en effet par exemple si on a appuyé en même temps sur les deux commandes N =1 et N = 4, le résultat codé est 101, ce qui ne correspond à aucune de deux combinaisons d'entrée. Un codeur prioritaire donne comme résultat 0100 qui correspond à N = 4. Le circuit intégré (74147) est un codeur prioritaire. Ce codeur regroupe à la fois les fonctions de bases qui sont les réunions de commande et les fonctions des conditions de priorités.

1-LE CODEUR DÉCODEUR TRANSCODEUR:

1.2-Décodeur:

1.2.1-définition:

Un décodeur est un dispositif qui effectue l'opération inverse du codeur c'est à dire pour n éléments en entrées on peut avoir 2^n combinaisons possibles en sortie, que l'on peut associer à un ensemble de 2^n éléments (chiffres, lettres , symboles...).



1.2.2-Réalisation d'un décodeur à deux entrées:

Un décodeur binaire à 2 entrées (E_1E_0) doit avoir 4 sorties ($2^2 = 4$), (S_0, S_1, S_2, S_3). Parfois, ce décodeur est appelé 1 parmi 4 (1/4)

La table de vérité du décodeur 1/4

E_1	E_0	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

Les équations des 4 sorties sont :

$$S0 = \overline{E}1.\overline{E}0$$
$$S1 = \overline{E}1.E0$$
$$S2 = E1.\overline{E}0$$
$$S3 = E1.E0$$

1.2.3-Mise en cascade de décodeurs:

En appliquant la méthode précédente, on peut réaliser un décodeur à plusieurs entrées. Toutefois dés que l'on cherche à réaliser un décodeur de plus de 3 entées, il est préférable d'adopter une structure en XY dite matrice de décodage. Cependant, compte tenu du nombre limité de connexions sur un circuit intégré, il est souvent utile de mettre en cascade les décodeurs pour permettre le décodage d'un grand nombre de combinaisons. L'utilisation d'une entrée supplémentaire permet ainsi la mise en cascade des décodeurs. Cette entrée est appelée entrée de **Validation,"V"** (Strobe :"S"). Si l'entrée de validation V = 1 le décodeur fonctionne normalement, par contre si V = 0 toutes les sorties du décodeur sont à zéro.

La table de vérité du décodeur 1/4 avec entrée de validation

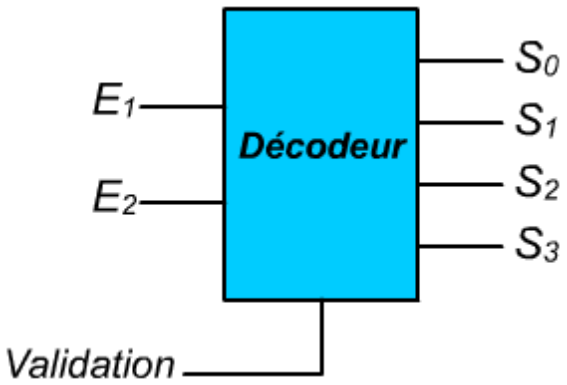
V	E_1	E_0	S_0	S_1	S_2	S_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Les équations logiques des 4 sorties en fonction des entrées, V, E_1et E_0 sont:

1-LE CODEUR DÉCODEUR TRANSCODEUR:

$$S0 = \overline{E}1.\overline{E}0.V$$
$$S1 = \overline{E}1.E0.V$$
$$S2 = E1.\overline{E}0.V$$
$$S3 = E1.E0.V$$

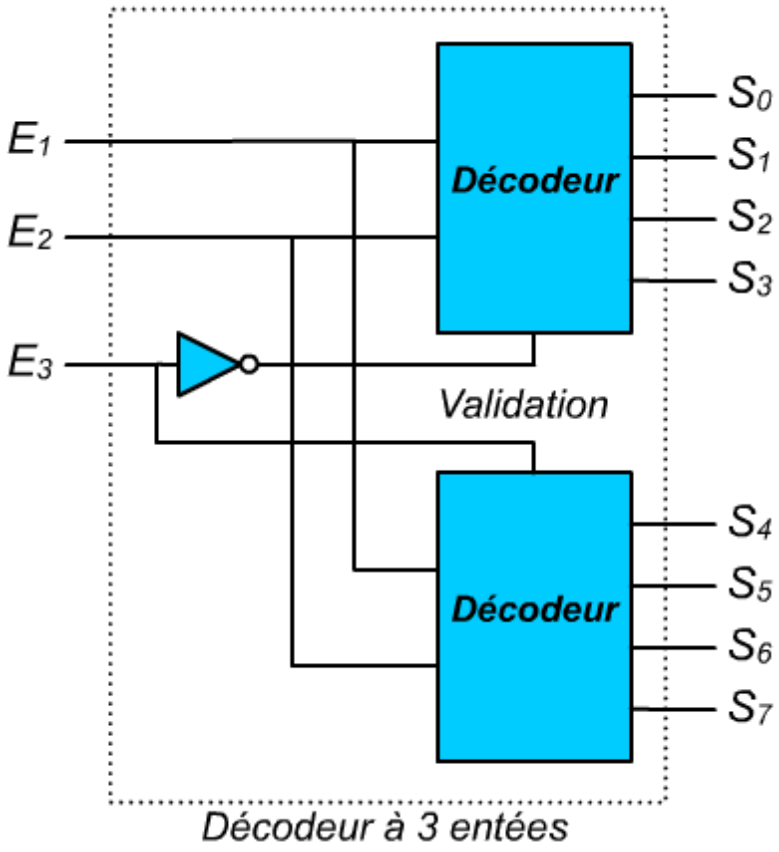
Symbole



Réaliser d'un décodeur à 3 entrées par des décodeurs à deux entrées:

Le principe de la mise en cascade des décodeurs consiste à utiliser l'entrées de validation comme entrée principale pour le décodage (E_3). En effet, l'entrée de validation permet de sélectionner ou de valider l'une de boîtier de décodeur (circuit décodeur) en fonction de sa valeur. Si E_3= 0 on sélectionne le décodeur du haut (N° 1) et on bloque le décodeur N°2, par contre si E_3= 1 on valide le décodeur N° 2 et on bloque le décodeur N°1.

1-LE CODEUR DÉCODEUR TRANSCODEUR:



Synthèse d'un décodeur BCD

Table de vérité:

Les Nombres Décimaux	a_3	a_2	a_1	a_0	Les sorties
0	0	0	0	0	S_0
1	0	0	0	1	S_1
2	0	0	1	0	S_2
3	0	0	1	1	S_3
4	0	1	0	0	S_4
5	0	1	0	1	S_5
6	0	1	1	0	S_6

1-LE CODEUR DÉCODEUR TRANSCODEUR:

7	0	1	1	1	S_7
8	1	0	0	0	S_8
9	1	0	0	1	S_9

Les équations logiques des 10 sorties:

On utilise le diagramme de Karnaugh pour trouver les expressions de 10 fonctions S_0 à S_9 en fonction des variables d'entrées(a_3...a_0). Sur le diagramme de Karnaugh il a des 6 cases vides puisqu' on arrête la conversion à 10.

Pour les cases vides on les remplit par des f.

Calculons par exemple S2

La table de vérité de S2

$a_3a_2\backslash a_1a_0$	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	ϕ	ϕ	ϕ	ϕ
10	0	0	ϕ	ϕ

$S2=\bar{a}2.a1\bar{a}0$

Les équations des autres sorties sont:

1-LE CODEUR DÉCODEUR TRANSCODEUR:

$S0=\bar{a}3\,\bar{a}2\,\bar{a}1\,\bar{a}0$

$S1=\bar{a}3\,\bar{a}2\,\bar{a}1.a0$

$S2=\bar{a}2.a1\,\bar{a}0$

$S3=\bar{a}2.a1.a0$

$S4=a2\,\bar{a}1\,\bar{a}0$

$S5=a2\,\bar{a}1.a0$

$S6=a2.a1\,\bar{a}0$

$S7=a2.a1.a0$

$S8=a3\,\bar{a}0$

$S9=a3.a0$

1.2.4-Applications:

1- Génération des fonctions logiques:

Soit la fonction F logique définie par sa table de vérité suivante:

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

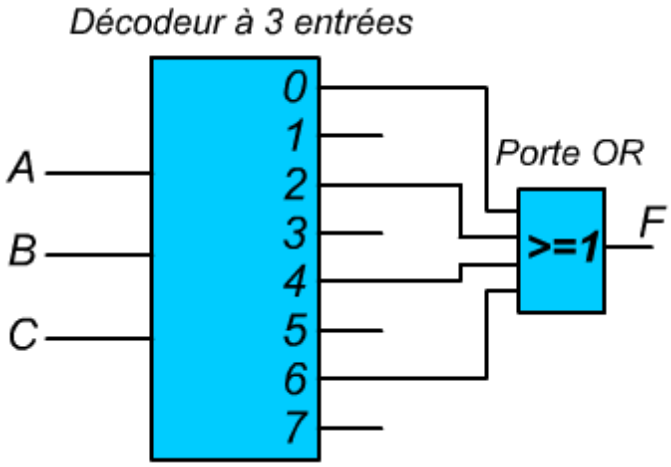
1-LE CODEUR DÉCODEUR TRANSCODEUR:

Réaliser cette fonction par un décodeur.

Solution:

On exprime F sous la première forme canonique (numérique), en associant les entrées A, B, C respectivement aux poids 4du code binaire pur, 2^2, 2^1, 2^0 **. On observe sur la table de vérité que la fonction est égale à 1 pour 0,2,4,6; donc :**

$F = \sum_3 (0,2,4,6)$. La fonction F est une fonction de 3 variables qui sont A, B, C. Pour réaliser cette fonction avec un décodeur à 3 entrées et à 8 sorties il suffit de faire la somme logique des sorties S_0, S_2, S_4, S_6 comme le montre la figue ci-dessous.



2-Adressage d'une mémoire

1-décodeur d'adresse

La représentation informatique d'une mémoire est celle d'une boite aux lettres. Chaque case est identifiée par un numéraux. Ce numéro est délivré par l' une des sorties d'un décodeur interne ayant par exemple n bits d'entrées. Ces n bits entrées sont appelées lignes d'adressage de la mémoire. donc dans une mémoire on trouve un décodeur d'adresse.

2-La sélection du boîtier (Chip Select)

Pour avoir une extension d'une zone mémoire on utilise des décodeurs pour sélectionner l'une des boîtiers de mémoire. La sélection du boîtier est assurée par le signal CS (Chip Select) délivré par le décodeur d'adressage.

3-Remarque

Il existe divers types de décodeurs, certains sont même propres à une application bien définie comme le SN 7447 qui set d'interface entre le BCD et afficheurs 7 segments, 7442, 74154, 74155.

8 -LE CODEUR DÉCODEUR TRANSCODEUR

8.3-Transcodeur :

8.3.1-Définition:

Le code binaire pur n'est pas universellement utilisé. Pour des applications données comme par exemple les transmissions des données numériques; qui font souvent appel à d'autres codes(Non Retour à Zéro (NRZ), Manchester Bipolaire etc..) . Un transcodeur est un dispositif permettant de convertir un nombre N du code 1 au code 2.

8.3.2-Exemples:

1-Transformation du code binaire pur à 4 bits en code Gray.

2-Faire la transformation inverse.

1- Le passage du code binaire en code Gray

La table de vérité:

Décimale	*Db*	*Cb*	*Bb*
0	0	0	0
1	0	0	0
2	0	0	1
3	0	0	1
4	0	1	0
5	0	1	0
6	0	1	1
7	0	1	1
8	1	0	0
9	1	0	0
10	1	0	1

11	1	0	1
12	1	1	0
13	1	1	0
14	1	1	1
15	1	1	0

La simplification des fonctions Ag, Bg, Cg et Dg par le tableau de Karnaugh est la suivante:

DbCb\BbAb	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

Ag = Ab⊕ Bb

DbCb\BbAb	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

Cg = Cb⊕ Db

2- Le passage inverse de code Gray en code binaire:

La simplification des fonctions Db, Cb, Bb , Ab par les tableaux de Kargnauh est la suivante:

DbCb\BbAb	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Ab = Dg⊕ Cg⊕ Bg⊕ Ag

DbCb\BbAb	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

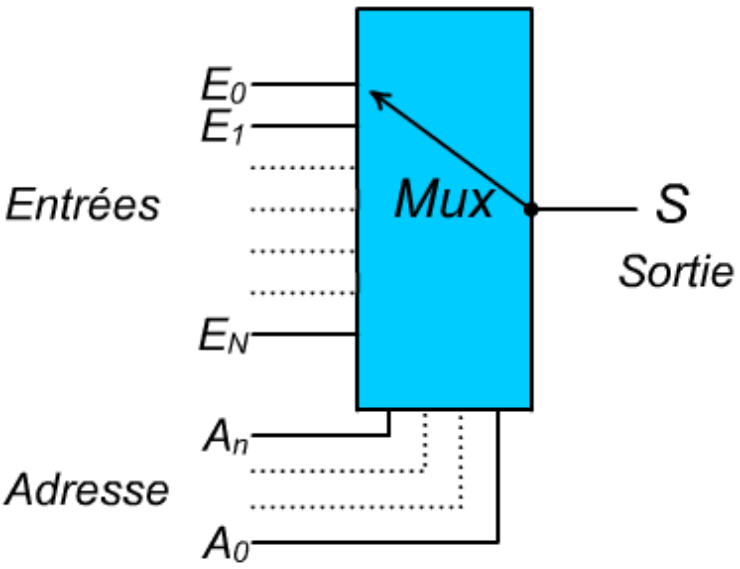
Cd = Cg⊕ Dg

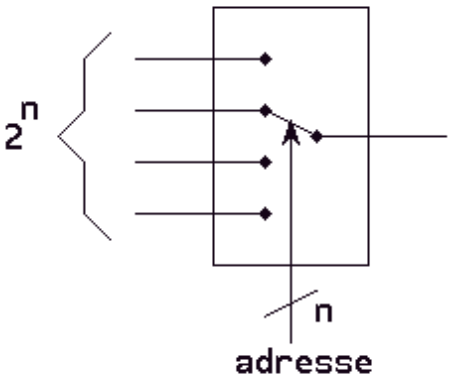
9- Multiplexeur Demultiplexeur:

9.1-multiplexeur:

9.1.1-Définition:

Un multiplexeur est un circuit combinatoire à N entrées d'information et une sortie unique. Cette sortie prend la valeur de l'une des entrées sélectionner par une adresse codée sur n bits (N = 2^n). Tout se passe comme s'il avait aiguillage de l'une des entrées vers la sortie.





9.1.2-Symbole d'un multiplexeur:

Exemple de réalisation:

Réaliser un multiplexeur à 4 entrées d'information E_0, E_1,E_2 E_4.

Pour sélectionner une entrée parmi les quatre, il faut 2 entrées d'adresse A_0, A_1.

Table de vérité:

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

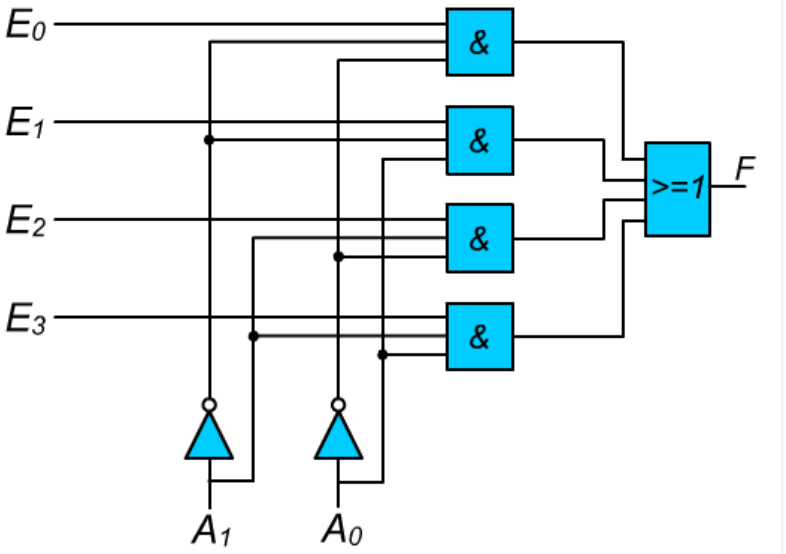
L'équation de sortie:

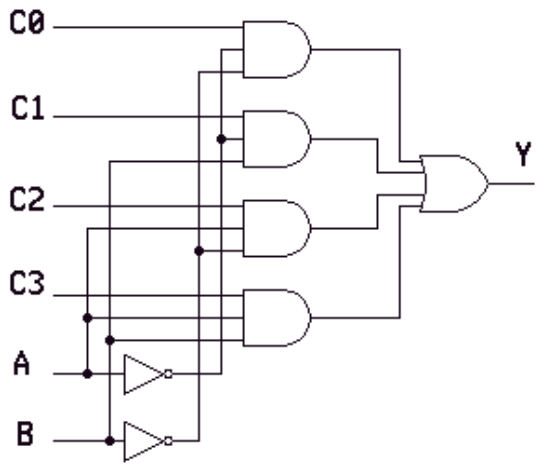
$$S=\overline{A}1.\overline{A}0E0+\overline{A}1.A0E1+A1.\overline{A}0E2+A1A0E3$$

Réalisation:

1 - réalisation Par des portes élémentaires:

9- Multiplexeur Demultiplexeur:



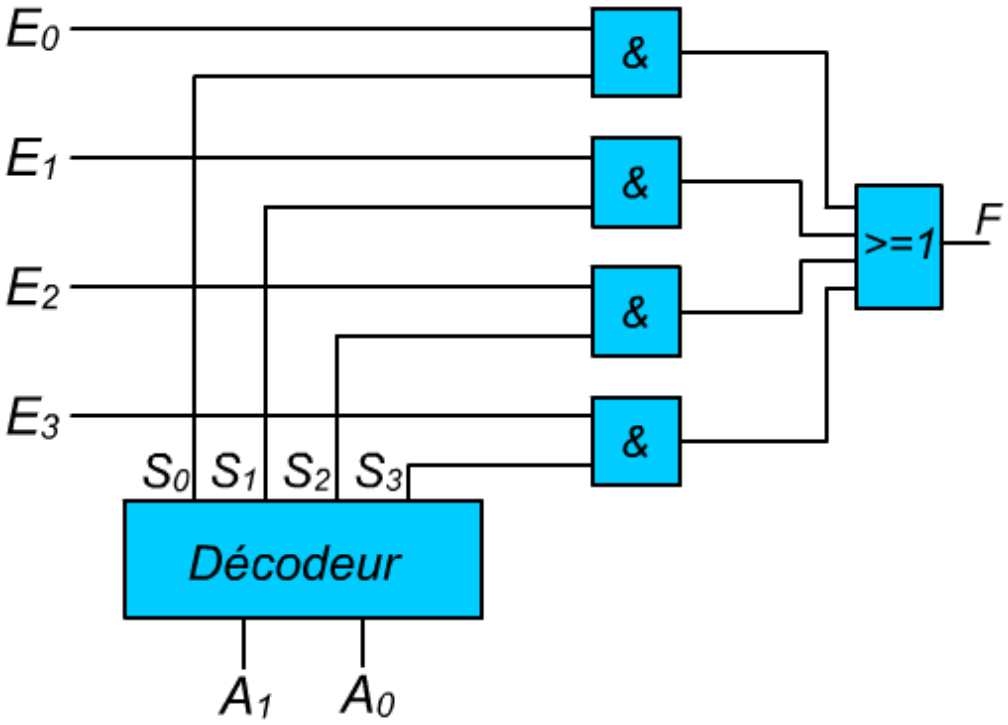


$$Y = \overline{A}.\overline{B}.C_0 + \overline{A}.B.C_1 + A.\overline{B}.C_2 + A.B.C_3$$

Mux 1/4 réalisé par des portes élémentaires.

2 - réalisation par un décodeur:

9- Multiplexeur Demultiplexeur:



Réalisation d'un Mux 1/4 par un décodeur et des portes logiques

3 - Réalisation par des multiplexeurs1/2:

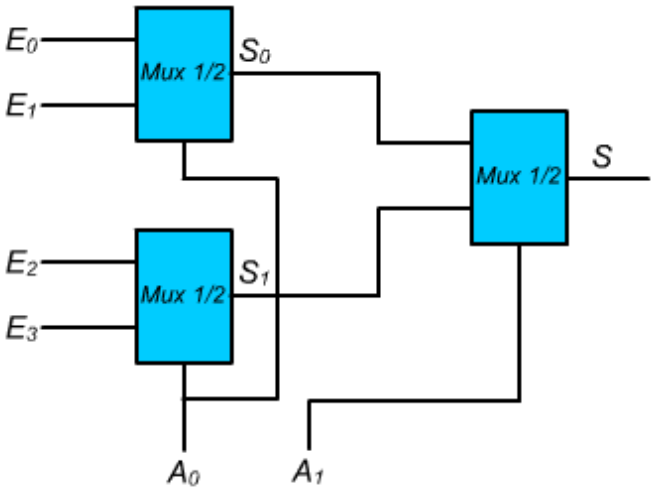
Pour réaliser un Mux 1/4, on peut utiliser trois Mux 1/2 comme le montre la figure ci dessous.

$$S = \overline{A}1S0 + A1S1$$

avec:

$$S0 = E0\overline{A}0 + E1.A0$$

$$S1 = E2\overline{A}0 + E3.A0$$



Mux 1/4 réalisé par 3 Mux 1/2.

S0 prend la valeur de E0 ou E1 en fonction de la valeur de A0, De même S1 prend la valeur de E2 ou E3 en fonction de la valeur de A0. L'aiguillage de S0 ou S1 vers la sortie dépend de la valeur de A1.

$$S = \overline{A}1S0 + A1S1$$

avec:

$$S0 = E0\overline{A}0 + E1.A0$$

$$S1 = E2\overline{A}0 + E3.A0$$

Multiplexeur de mots d'entrée

Un multiplexeur de mot travaille simultanément sur plusieurs bits. Il peut être considéré comme un ensemble de multiplexeurs de 1 parmi N fonctionnant en même temps avec la même adresse.

Dans cette famille de Mux on trouve par exemple:

Multiplexeur de 2 mots de 4 bits

Multiplexeur de 3 mots de 4 bits

Multiplexeur de 4 mots de 2 bits (54 / 74153)

Table de vérité du 74153:

A	*B*	*1G*	A_0	B_0	C_0	D_0	*2G*	A_1	B_1	C_1	D_1	*1Y*	*2Y*
X	X		X	X	X	X		X	X	X	X		
				X	X	X			X	X	X		
				X	X	X			X	X	X		
			X		X	X		X		X	X		
			X		X	X		X		X	X		
			X	X		X		X	X		X		
			X	X		X		X	X		X		
			X	X	X			X	X	X			
			X	X	X			X	X	X			

9.1.3-Application:

- **Génération de fonction:**

Pour réaliser une fonction logique par un Mux on effectue les opérations suivantes :
- On écrit l'équation de la fonction logique sous la première forme canonique (on détermine le nombre d'entrée).
- On écrit l'équation du Mux caractérisé par le nombre d'entrée d'adresse (nombre d'entrée d'adresse = nombre d'entées de la fonction -1).

Identification de deux équations

Prenons par exemple une fonction de 4 variables a b c d, qui s'écrit sous la première forme canonique:

$$F = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}.d + \bar{a}\bar{b}.c\,\bar{d} + ... + a.b.c.d$$

L'équation du Multiplexeur à 3 (4-1) entrées d'adresse est la suivante:

$$S = \bar{A}2.\bar{A}1.\bar{A}0.E0 + \bar{A}2.\bar{A}1.A0.E1 + \bar{A}2.A1.\bar{A}0.E2 + ... + A2.A1.A0.E7$$

On réalise les identifications:

- Un monôme abc avec un terme *A2A1A0* homologue
- La variable *d* avec les entrées *E0* à *E7* telle que:

  - Si le terme existe et dépend de *d*, on connectera d à l'entrée correspondante.
  - Si le terme existe et dépend de *d* bar, on connectera d bar à l'entrée correspondante.
  - Si le terme existe et ne dépend pas de *d*, on connectera l'entrée correspondante à 1.
  - Si le terme n'existe pas on connectera un 0 à l'entrée correspondante.

Exemple: réaliser la fonction $S1 = \bar{a}\bar{b}\bar{c} + a\bar{b}.c + a.c\,\bar{d} + \bar{a}\bar{c}.d$ Par un Mux.

On écrit S sous la première forme canonique. Les variables d, c, b et a sont affectées respectivement des poids du code binaire pur, c'est à dire 2^3, 2^2, 2^1, 2^0.

9- Multiplexeur Demultiplexeur:

$$S1=\bar{a}\bar{b}\bar{c}\bar{d}+a.\bar{b}.c.\bar{d}+a.b.c.\bar{d}+a\bar{b}.c.d+\bar{a}\bar{b}\bar{c}.d+\bar{a}.b.\bar{c}.d=\bar{d}$$

$$S = 0.\bar{d}+5.\bar{d}+7.\bar{d}+5d+0.d+2.d = 0+2.d+5+7.\bar{d}$$

On utilise un Mux à 3 entrées d'adresse et 8 entrées d'information (E0, E1, E2, E3, E4, E5, E6, E7).

E0 est connectée à 1 puisque le terme 0 est indépendant de *d* et $\bar{d}$

E1 est connectée à 0 puisque le terme 1 n'existe pas

E2 est connectée à 1 puisque le terme 2 existe et dépendant de *d*

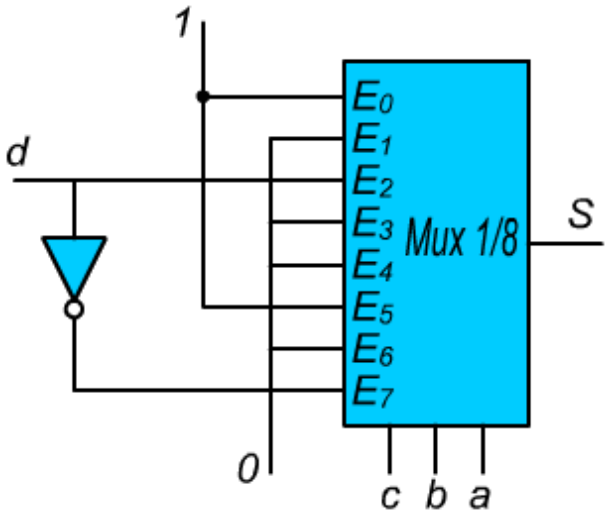
E3 est connectée à 0 puisque le terme 3 n'existe pas

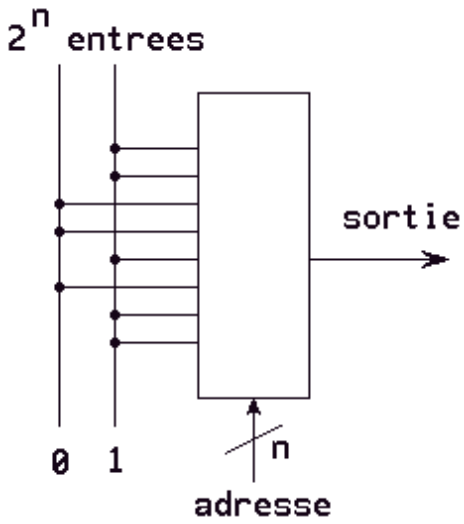
E4 est connectée à 0 puisque le terme 4 n'existe pas

E5 est connectée à 1 puisque le terme 5 est indépendant de *d* et $\bar{d}$

E6 est connectée à 0 puisque le terme 6 n'existe pas

E7 est connectée à 1 puisque le terme 7 existe et dépendant de $\bar{d}$



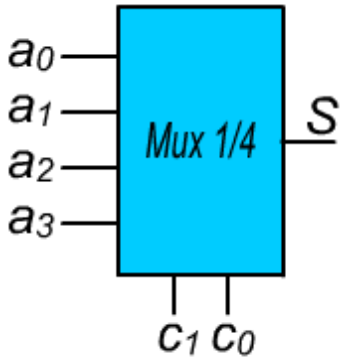


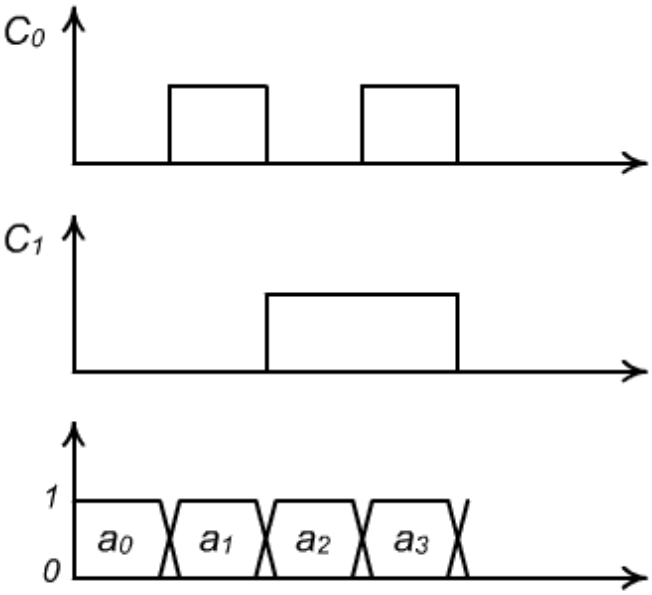
Connexion

Conversion parallèle série:

On rappelle qu à l'intérieur des unités de traitement ou des ordinateurs les calcules et les traitements se font sur des mots binaires des (4, 8, 12, 16, 32, 64bits). Ces mots sont disponibles en mode parallèle, c'est à dire chaque fil étant affecté à un élément binaire du mot. Pour transmettre les éléments binaires en série, c'est à dire les uns à la suite des autres sur un seul fil. Il faut réaliser une transformation Parallèle - série en utilisant un multiplexeur numérique associé à un compteur binaire. Pendant la 1ére top d'horloge on transmet le LSB bit de poids 0 2éme top bit de pois 1 ainsi de suite jusqu'on transmet le bit MSB.

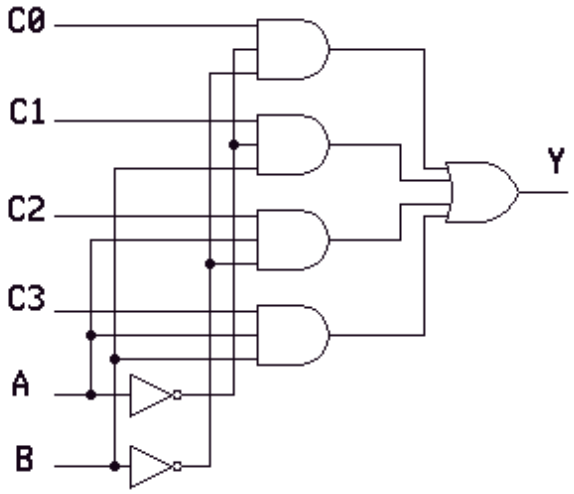
Soit l'exemple à convertir en mode série une information de 4 bits disponible en parallèle. Un compteur à 2 étages peut délivrer 4 adresses. Il est mis à zéro au début et chaque impulsion fait progresser de un le compteur donc l'adresse envoyée. La figue 1 montre une organisation possible. On recueille en sortie S l'information série, le premier bit présenté étant celui qui est connecté à l'entrée n° 0, le deuxième bit présenté étant celui qui est connecté à l'entrée n ° 2 et ainsi de suite.



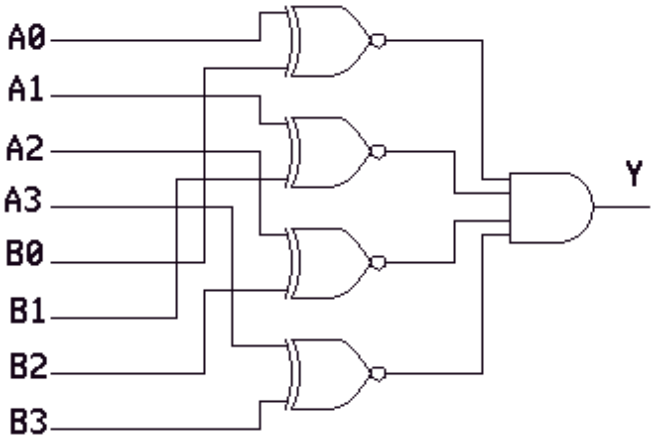


Chronogramme de la conversion parallèle série

9- Multiplexeur Demultiplexeur:



$$Y = \overline{A}.\overline{B}.C_0 + \overline{A}.B.C_1 + A.\overline{B}.C_2 + A.B.C_3$$



$$S = \overline{A_0 \oplus B_0} \; . \; \overline{A_1 \oplus B_1} \; . \; \overline{A_2 \oplus B_2} \; . \; \overline{A_3 \oplus B_3}$$

9- Multiplexeur demultiplexeur

9.2-Démultiplexeur:

9.2.1-Définition:

Un démultiplexeur (DMux) est un circuit combinatoire ayant une entre d'information n entrées d'adressage (affichage) et 2n sorties. Ce circuit réalise l'aiguillage d'information. La différence entre le Mux et le DMux réside dans le sens de circulation de l'information.

9.2.2-Réalisation d'un démultiplexeur à deux voies (sorties):

symbole:

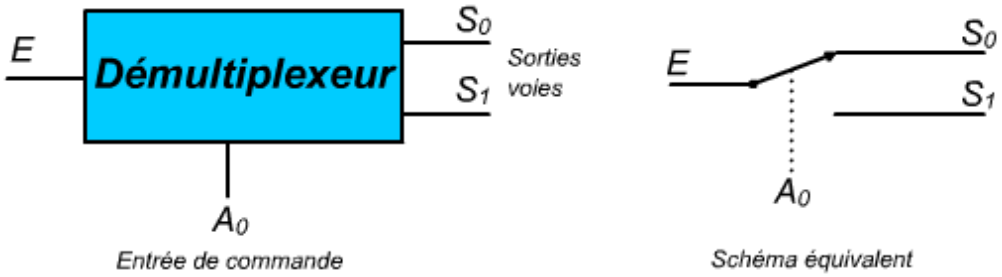


Table de vérité

Pour A0 = 0 , on a S0 = E

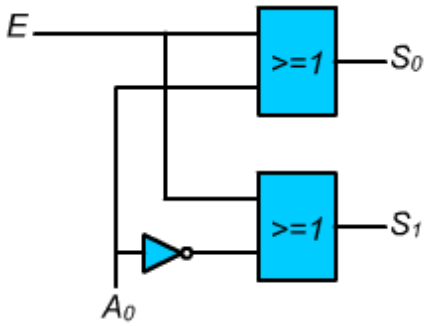
Pour A0 = 1 , on a S1 = E

La sortie non sélectionnée est à l'état 1.

E	A_0	S_0	S_1
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

9.2.3-Réalisation:

$$\overline{S0}=\overline{A0}.\overline{E}=\overline{A0+E}, \quad donc \quad S0=A0+E$$
$$\overline{S1}=A0.\overline{E}=\overline{E+\overline{A0}}, \quad donc \quad S1=E+\overline{A0}$$



Réalisation d'un démultiplexeur à deux voies

Table de vérité d'un multiplexeur à quatre voies

A_1	A_0	E	S_0	S_1	S_2	S_3
0	0	0	0	1	1	1
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	0
1	1	1	1	1	1	1

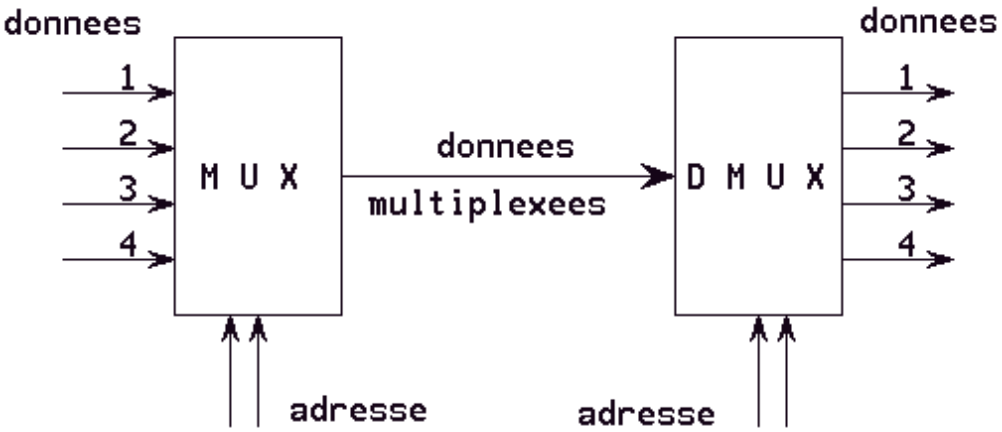
Toutes les cases hachurées ont pour valeur logique un 1,puisqu'on sait que les sorties non sélectionnées du démultiplexeur sont à l'état 1.

$$\overline{S}0 = \overline{A}0.\overline{A}1.\overline{E}$$ donc S_0 *est:*

$$S_0 = E + A_0 + A_1$$

9.2.4-Concentration d'un grand nombre de données:

Concentration de données et transmission parallèle



Logique combinatoire	Auteur
Systèmes logiques combinatoires	Ecole polytechnique fédérale de Lausanne
La logique combinatoire	Robert Papanicola
Logique combinatoire	Rémi Lechartier
Cours sur la logique combinatoire	Gilles Bouvier et Génaël Valet
Les systèmes combinatoires	Yvan Crévits
Logique combinatoire avec des chronogrammes intéractifs en javascript	Nicolas Midoux
Encodage et Décodage	NeT_TroniquE
Multiplexage et Démultiplexage	NeT_TroniquE
Affichage Numérique	NeT_TroniquE
Additionneur & Soustracteur	netProblèMATHique
Circuits Combinatoires	Stéphane Martin