

Bases des systèmes Numériques

A. Oumnad

Chapitre I -- Systèmes de Numération.....	3
I.1 -- Système décimal.....	3
I.2 -- Système Octal.....	3
I.3 -- Système Hexadécimal.....	3
I.4 -- Système Binaire Naturel.....	3
I.5 -- Changement de base.....	4
I.5.1 -- Décimal → Binaire.....	4
I.5.2 -- Décimal → Octal.....	4
I.5.3 -- Décimal → Hexadécimal.....	5
I.5.4 -- Octal ↔ binaire.....	5
I.5.5 -- Hexadécimal ↔ binaire.....	5
I.6 -- Opérations dans le système Binaire naturel.....	6
I.6.1 -- Représentation des nombres négatifs.....	6
I.6.2 -- La multiplication.....	7
Chapitre II -- NOTIONS SUR les CODES.....	8
II.1 -- Codes décimaux.....	8
II.2 -- Codes réfléchis.....	9
II.2.1 -- Code de Gray.....	9
Chapitre III -- VARIABLES ET FONCTIONS BOOLEENNE.....	10
III.1 -- Algèbre de Boole.....	10
III.2 -- Variable booléenne.....	10
III.3 -- Fonction booléenne.....	10
III.4 -- Opérateur Logiques.....	11
III.4.1 -- Opérateur ET (AND).....	11
III.4.2 -- Opérateur OU (OR).....	11
III.4.3 -- Opérateur NON-ET (NAND).....	11
III.4.4 -- Opérateur OU EXCLUSIF (XOR).....	11
III.4.5 -- Analogie.....	11
III.5 -- Identités remarquables.....	12
III.6 -- Théorème de Morgan.....	13
III.7 -- Forme canonique.....	13
III.8 -- Théorème de la disjonction.....	13
III.9 -- NAND : Opérateur universel.....	14
III.9.1 -- Règle des fantômes :.....	15
Chapitre IV -- Simplification des fonctions logique.....	15
IV.1 -- Méthode Karnaugh (1953).....	15
IV.1.1 -- Diagramme de Karnaugh.....	15
IV.1.2 -- Remplissage de la table de Karnaugh à partir de la table de vérité.....	16
IV.1.3 -- Remplissage de la table de Karnaugh à partir d'une fonction.....	17
IV.1.4 -- Lecture du digramme de Karnaugh.....	18
IV.1.5 -- Code Interdit ou quand la valeur compte peu.....	19
IV.2 -- Exemple 1 : Convertisseur Binair naturel vers code de Gray.....	20
IV.3 -- Exemple 2 : Synthèse d'un système logique.....	21

CHAPITRE I -- SYSTEMES DE NUMERATION

I.1 -- Système décimal

Le système décimal est le système à base 10. Pour écrire un nombre dans ce système, on a besoin de 10 symboles qu'on a l'habitude d'appeler chiffres. L'écriture implique l'application d'un principe de position, tout chiffre placé à gauche d'un autre représente des unités 10 fois plus fortes que cet autre, cette unité est dite le poids du chiffre. Le poids du dernier chiffre à droite avant la virgule est $10^0=1$, c'est le chiffre des unités. A droite de la virgule, le poids est divisé par 10 chaque fois qu'on avance d'un chiffre vers la droite.

$$5289.76 = 5.10^3 + 2.10^2 + 8.10^1 + 9.10^0 + 7.10^{-1} + 6.10^{-2} +$$

I.2 -- Système Octal

C'est le système de base 8. L'écriture d'un nombre dans ce système nécessite 8 chiffres, on utilise les chiffres de 0 à 7. Dans cette base les poids sont :

...	32768	4096	512	64	8	1
...	8^5	8^4	8^3	8^2	8^1	8^0

$$235_8 = 5.8^0 + 3.8^1 + 2.8^2 = 157_{10}$$

$$2743_8 = 3.8^0 + 4.8^1 + 7.8^2 + 2.8^3 = 1507_{10}$$

Voici quelques opérations dans le système Octal :

5	7	1535	2743	235
<u>+4</u>	<u>+6</u>	<u>+2627</u>	<u>+5326</u>	<u>x 3</u>
11	15	4364	10271	723

I.3 -- Système Hexadécimal

C'est le système de base 16. L'écriture d'un nombre dans ce système nécessite 16 chiffres, on utilise les chiffres de 0 à 9 plus les lettres A, B, C, D, E et F.

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

$$6EA_{16} = (10.16^0 + 14.16^1 + 6.16^2)_{10} = 1770_{10}$$

A4B9

+3FE6

E49F

I.4 -- Système Binaire Naturel

C'est le système à base 2. Il utilise deux chiffres 0 et 1.

32	16	8	4	2	1	
2^5	2^4	2^3	2^2	2^1	2^0	
1	0	1	1	0	1	$= (32+8+4+1)_{10} = 45_{10}$

Ce système est très utilisé car facilement matérialisable. Les deux chiffres 0 et 1 peuvent être mis en évidence techniquement par plusieurs façons :

- Passage ou non d'un courant électrique dans un composant.
- Etat d'un commutateur (transistor ou autre) ouvert ou fermé.
- Valeur d'une tension à la sortie d'un composant, ex: +5V = "1" et 0V = "0"

$$100111_2 = (1+2+4+32)_{10} = 39_{10}.$$

$$11111_2 = (1+2+4+8+16)_{10} = 31_{10}.$$

I.5 -- Changement de base

I.5.1 -- Décimal → Binaire

$$175_{10} = ?_2$$

On cherche la plus grande puissance de 2 contenues dans 175, c'est $128=2^7$.

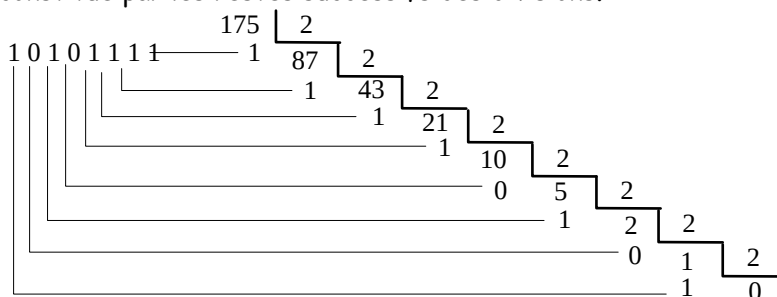
$$175 = 128(2^7) + 47, \text{ On refait la même chose avec } 47$$

$$47 = 32(2^5) + 15, \text{ de la même façon on obtient}$$

$$15 = 8 + 4 + 2 + 1, \text{ Donc}$$

$$175_{10} = 2^7 + 2^5 + 2^3 + 2^2 + 2^1 + 2^0 = 10101111_2$$

Cette conversion peut se faire d'une façon systématique à l'aide d'une division successive par 2. Le résultat en binaire est constitué par les restes successifs des divisions.



$$346_{10} = 101011010_2$$

$$233_{10} = 11101001_2$$

$$84_{10} = 1010100_2$$

$$42_{10} = 101010_2$$

Pour les chiffres à droite de la virgule, on procède d'une façon similaire, sachant que les poids à droite de la virgule sont 0.5, 0.25, 0.125 ... La division en échelle se transforme en une multiplication en échelle qui s'arrête quand le résultat est nul. IL arrive qu'on ne tombe jamais sur un résultat nul et la multiplication ne s'arrête jamais. Dans ce cas on est libre d'arrêter dès qu'on juge que le nombre de chiffres après la virgule est suffisant.

$$\text{exemple : } (175.34)_{10} = (?)_2$$

$$(175)_{10} = (10101111)_2$$

$$0.34 \times 2 = \underline{0}.68, 0.68 \times 2 = \underline{1}.36, 0.36 \times 2 = \underline{0}.72, 0.72 \times 2 = \underline{1}.44, 0.44 \times 2 = \underline{0}.88$$

$$0.34 = 0.01010$$

$$((175)_{10}) = (10101111.01010)_2$$

I.5.2 -- Décimal → Octal

$$318_{10} = ?$$

- On cherche la plus grande puissance de 8 (le plus grand poids) contenue dans 318. C'est 64.

$$\text{On aura donc } 318_{10} = cba_8 \text{ t.q. } 318 = a.1 + b.8 + c.64$$

$$\begin{array}{r} 318 \\ 62 \overline{) 64} \\ \underline{4} \end{array}$$

- On cherche combien de fois 318 contient de 64 :

$$\text{On a donc } 318 = 4 * 64 + 62 \quad (c=4)$$

- On refait la même opération avec 62 On considérant successivement tous les poids inférieurs au poids max. (64 pour cet exemple). Le poids juste inférieur est 8, On obtient $62 = 7 * 8 + 6 \quad (b=7)$

- Le poids inférieur est 1. On a donc $6 = 6 * 1 + 0$ ($a=6$).
On s'arrête car le reste est 0.

$$318_{10} = 476_8$$

$$1223_{10} = ?_8$$

$$1223 = 2 * 512 + 199$$

$$199 = 3 * 64 + 7$$

$$7 = 0 * 8 + 7$$

$$7 = 7 * 1 + 0$$

$$1223_{10} = 2307_8$$

Une deuxième méthode pour convertir un nombre décimal vers une base quelconque B consiste à faire une division successive par B autant de fois que cela est nécessaire jusqu'à obtenir un **quotient** nul. On écrit alors les restes où ils ont été obtenus dans l'ordre inverse

I.5.3 -- Décimal → Hexadécimal

$$16617_{10} = 40E9_{16}$$

$$5000_{10} = 1388_{16}$$

$$\begin{array}{r}
 16617 \div 16 = 1038 \text{ (reste } 9) \\
 1038 \div 16 = 64 \text{ (reste } 14) \\
 64 \div 16 = 4 \text{ (reste } 0) \\
 4 \div 16 = 0 \text{ (reste } 4)
 \end{array}$$

$$\begin{array}{r}
 5000 \div 16 = 312 \text{ (reste } 8) \\
 312 \div 16 = 19 \text{ (reste } 8) \\
 19 \div 16 = 1 \text{ (reste } 3) \\
 1 \div 16 = 0 \text{ (reste } 1)
 \end{array}$$

I.5.4 -- Octal ↔ binaire

Octal → binaire : On écrit chaque chiffre octal est écrit sur 3 bits

$$235_8 = 010\ 011\ 101_2$$

$$753_8 = 111\ 101\ 011_2$$

Binaire → octal : Le nombre binaire est découpé en tronçon de 3 bits en commençant de la droite (LSB), et chaque 3bits seront écrit en octal.

$$1100111101100_2 = 1\ 100\ 111\ 101\ 100_2 = 14754_8$$

I.5.5 -- Hexadécimal ↔ binaire

Hexadécimal → binaire : On écrit chaque chiffre hexadécimal sur 4 bits

$$235_{16} = 0010\ 0011\ 0101_2$$

$$753_{16} = 0111\ 0101\ 0011_2$$

$$8975_{16} = 1000\ 1001\ 0111\ 0101_2$$

$$8E97A_{16} = 1000\ 1110\ 1001\ 0111\ 1010_2$$

Binaire → hexadécimal : Le nombre binaire est découpé en tronçon de 4 bits en commençant de la droite (LSB), et chaque 4 bits seront écrits en hexadécimal.

$$1100111101100_2 = 1\ 1001\ 1110\ 1100_2 = 19EC_{16}$$

$$1111011110011011_2 = 1\ 1110\ 1111\ 1001\ 1011_2 = 1EF9B_{16}$$

$$\begin{array}{r}
 1223 \div 8 = 152 \text{ (reste } 7) \\
 152 \div 8 = 19 \text{ (reste } 0) \\
 19 \div 8 = 2 \text{ (reste } 3) \\
 2 \div 8 = 0 \text{ (reste } 2)
 \end{array}$$

I.6 -- Opérations dans le système Binaire naturel

Les choses se passent exactement comme on a l'habitude de le faire en base 10, seulement ici nous n'avons que 2 chiffres 0 et 1:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 1 = 0 \text{ et on retient } 1$$

$$1 + 1 + 1 = 1 \text{ et on retient } 1$$

$$\begin{array}{r} 101 \\ + 11 \\ \hline 1000 \end{array} \quad \begin{array}{r} 111 \\ + 101 \\ \hline 1100 \end{array}$$

Une contrainte très importante va intervenir dès qu'il s'agit de faire des opérations par des machines car celles ci travaillent avec des registres de tailles fixes donc traitent des nombres qui ont toujours le même nombre de bits. Prenons l'exemple d'une machine qui **travaille sur 4 bits, elle peut représenter $16=2^4$ nombres différents**. Deux possibilités s'offrent à nous :

- Les 16 nombres seront considérés comme des entiers non signés. On aura donc 16 nombres positifs allant de 0 à 15
- Les 16 nombres seront considérés comme des nombres signés. On aura donc 8 nombres positifs (moitié) allant de 0 à 7 et 8 nombres négatifs allant de -8 à -1.

Machine n bits $\equiv$ 2^n Nombres différents		
Nombres non signés	Nombres signés	
2^n Nombres	$2^n / 2$ Négatifs	$2^n / 2$ Positifs
0 $\rightarrow$ $2^n - 1$	-1 $\rightarrow$ $- 2^n / 2$	0 $\rightarrow$ $2^n/2 - 1$

Remarquons au passage que $2^n/2 = 2^{n-1}$

I.6.1 -- Représentation des nombres négatifs

Reprenons l'exemple d'une machine 4 bits, si on ajoute deux nombres et le résultat est supérieur à 15, il écrit normalement sur 5 bits est la machine n'a que 4 bits donc le 5^{ème} sera perdu, et le résultat sera diminué de $16_{10}=10000_2$

$$\begin{array}{r} 0101 \text{ (5)} \\ + 0011 \text{ (3)} \\ \hline 1000 \text{ (8)} \end{array} \quad \begin{array}{r} 0111 \text{ (7)} \\ + 1101 \text{ (13)} \\ \hline 0100 \text{ (4)} \end{array}$$

On s'aperçoit finalement qu'une machine 4 bits travaille modulo $16=2^4$, c'est à dire que pour cette machine, on a $16 = 0$. Donc pour cette machine l'opposé d'un nombre N est -N tel que $N + (-N) = 0 = 16$ soit $-N = 16 - N$.

On peut écrire $-N = 15 + 1 - N = 15 - N + 1$.

L'Opération $15 - N$ s'appelle le complément à 1 du nombre N car elle revient à complémenter (par rapport à 1) un par un tous les bits du nombre N.

Nombre	0101 (5)	1111 (15)
Son complément à 1	1010 (10=15-5)	0000 (15-15=0)

Finalement pour obtenir l'opposé d'un nombre N, il suffit de complémenter tous ses bits puis de rajouter 1, cette opération s'appelle le complément à 2 de N.

Si on note $\bar{N}$ le complément à 1 de N on a : $-N = \bar{N} + 1$

$$\begin{array}{r} -7 = \bar{7} + 1 = 1001 \\ \begin{array}{r} 7 \quad 0111 \\ \bar{7} \quad 1000 \\ + 1 \\ \hline \end{array} \end{array}$$

1001

Une machine 4 bits peut donc représenter les nombres suivants

Négatifs		Positifs	
-1	1111	0	0000
-2	1110	1	0001
-3	1101	2	0010
-4	1100	3	0011
-5	1011	4	0100
-6	1010	5	0101
-7	1001	6	0110
-8	1000	7	0111

Remarquons que pour les nombres positifs, le MSB est égal à 0 et pour les nombres négatifs le MSB est égal à 1. C'est ainsi qu'on peut tester si le résultat d'une opération est juste. La règle est simple :

- la somme de 2 nombres positifs doit donner un nombre positif
- la somme de 2 nombres négatifs doit donner un nombre négatif

L'erreur se produit quand il y a débordement, c.a.d quand le résultat dépasse le nombre maximum que la machine peut représenter soit +7 pour les nombres positifs ou -8 pour les nombres négatifs. Une erreur peut se produire quand on ajoute des nombres de même signe. Il y a erreur quand le signe du résultat est différent de celui des deux opérandes

0011 (3)	0101 (5)	0010 (2)	1110 (-2)	1011 (-5)
<u>+0100 (4)</u>	<u>+0110 (6)</u>	<u>+1001 (-7)</u>	<u>+1101 (-3)</u>	<u>+1100 (-4)</u>
0111 (7)	erreur	1011 (-5)	1011 (-5)	erreur

I.6.2 -- La multiplication

Regardons d'abord comment on fait une multiplication en base 2 sur un aspect mathématique pur sans limitation sur le nombre de bits. Remarquons au passage qu'ici on considère que tous les nombres sont positifs car le concept de représentation des nombres négatifs par le complément à 2 n'est valable que si on travaille avec un nombre de bit fixe.

$$\begin{array}{r}
 1011 \quad (11) \\
 *1101 \quad (13) \\
 \hline
 1011 \\
 1011 \\
 1011 \\
 1011 \\
 \hline
 10001111 \quad (143)
 \end{array}$$

Regardons maintenant comment on doit procéder si on travaille avec des nombres signés sur un nombre de bit fixe :

On commence à procéder de la même façon sauf que l'addition doit se faire sur 4 bits seulement, les bits de gauches seront ignorés.

On vérifiera que le résultat est juste en vérifiant les règles suivantes :

- Le produit de deux nombres de même signe doit être positif
- Le produit de deux nombres de signes différents doit être négatif

$$\begin{array}{r}
 0010 \quad (2) \\
 *0011 \quad (3) \\
 \hline
 0010 \\
 0010 \\
 \hline
 0110 \quad (6)
 \end{array}
 \quad
 \begin{array}{r}
 1110 \quad (-2) \\
 *1101 \quad (-3) \\
 \hline
 1110 \\
 1110 \bullet \\
 1110 \\
 \hline
 0110 \quad (6)
 \end{array}
 \quad
 \begin{array}{r}
 1110 \quad (-2) \\
 *0100 \quad (4) \\
 \hline
 1110 \bullet \bullet \\
 1000 \quad (-8)
 \end{array}
 \quad
 \begin{array}{r}
 0010 \quad (2) \\
 *0101 \quad (5) \\
 \hline
 0010 \\
 0010 \bullet \\
 \hline
 1010 \quad (-6: \text{erreur})
 \end{array}$$

CHAPITRE II -- NOTIONS SUR LES CODES

Les systèmes utilisés précédemment constituent des systèmes naturels ou codes naturels ou encore codes pondérés. Ils sont caractérisés par le fait que le poids du chiffre de rang i est B fois celui du rang $i-1$, B étant la base du système.

Il existe d'autres codes qui possèdent des avantages particuliers pour des utilisations particulières, on peut citer :

- L'utilisation d'un code particulier peut rendre le traitement d'un message plus ou moins économique du point de vue temps de traitement ou encombrement en mémoire ou encore en nombre de composant nécessaire pour effectuer le traitement.

- Dans de nombreux cas on peut améliorer l'efficacité d'un système de communication en utilisant des codes détecteurs d'erreurs ou encore des codes correcteurs d'erreurs ...

II.1 -- Codes décimaux

Dans la vie courante le système de numération le plus fréquent étant le décimal, il est souvent utile de garder d'un nombre une représentation décimale, tout en travaillant en binaire. On réalise ce tour de force simplement en convertissant chaque chiffre de la représentation décimale en un des nombreux codes binaires. Les chiffres utilisés varient de 0 à 9, nous aurons besoin d'au moins quatre bits par chiffre. Voici quelques exemples de codes décimaux résumés dans le tableau suivant:

	BCD	Exces 3	Aiken	Johnson	Biquinaire	2 parmi 5
0	0000	0011	0000	00000	01 00001	11000
1	0001	0100	0001	00001	01 00010	00011
2	0010	0101	0010	00011	01 00100	00101
3	0011	0110	0011	00111	01 01000	00110
4	0100	0111	0100	01111	01 10000	01001
5	0101	1000	1011	11111	10 00001	01010
6	0110	1001	1100	11110	10 00010	01100
7	0111	1010	1101	11100	10 00100	10001
8	1000	1011	1110	11000	10 01000	10010
9	1001	1100	1111	10000	10 10000	10100

Le plus courant est le code **DCB** (Décimal codé en binaire), en anglais **BCD** (*Binary Coded decimal*), chaque chiffre du système décimal est codé en binaire naturel sur 4 bits.

Décimal	Binaire (naturel)	BCD
7956	1111100010100	0111 1001 0101 0110
9801	10011001001001	1001 1000 0000 0001

Ce système est très utilisé pour les systèmes d'affichage sur afficheurs 7 segments. Pour afficher le nombre 9801 par exemple, au lieu d'utiliser son code binaire naturel, on utilise le code BCD et chaque afficheur reçoit les 4 bits correspondant à un chiffre

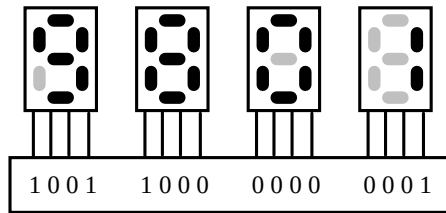


fig. : affiche à du code BCD

Le code Exces 3 ou excédent 3 est obtenu en décalant le binaire naturel de 3 lignes vers le haut. Ce code est intéressant pour effectuer des soustractions car le complément à 9 d'un chiffre s'obtient en inversant chaque bit. On ramène la soustraction à une addition. (Je trouve que ce code est inutilement compliqué, alors que la soustraction à l'aide de la représentation des nombres négatifs par le complément à deux est bien plus simple)

II.2 -- Codes réfléchis

On dit qu'un code est continu au sens large si dans la table de vérité qui le définit, les états successifs sont adjacents, c'est à dire que quand on passe de l'un à l'autre, il y a un seul chiffre qui change.

Un code est continu au sens stricte si en plus le premier et le dernier état sont adjacents.

Un code réfléchi est un code naturel dont on a renversé le sens de variation par endroits afin de le rendre continu. On renverse une période sur deux en commençant par la deuxième. Le tableau ci-dessous illustre le code ternaire (base 3) réfléchi.

Décimal	Ternaire	Ternaire réfléchi	
0	000	0	0
1	001	0	1
2	002	0	2
3	010	0	1
4	011	0	1
5	012	0	1
6	020	0	2
7	021	0	2
8	022	0	2
9	100	1	2
10	101	1	2
11	102	1	2
12	110	1	1
13	111	1	1
14	112	1	1
15	120	1	0
16	121	1	0
17	122	1	0
18	200	2	0
19	201	2	0
20	202	2	0
21	210	2	1
22	211	2	1
23	212	2	1
24	220	2	2
25	221	2	2
26	222	2	2

II.2.1 -- Code de Gray

decim	Binaire naturel	Code de Gray			
	dcba	D	C	B	A
0	0000	0	0	0	0
1	0001	0	0	0	1
2	0010	0	0	1	1
3	0011	0	0	1	0
4	0100	0	1	1	0
5	0101	0	1	0	1
6	0110	0	1	0	0
7	0111	0	1	1	0
8	1000	1	1	0	0
9	1001	1	1	0	1
10	1010	1	1	1	1
11	1011	1	1	1	0
12	1100	1	0	1	0
13	1101	1	0	1	1
14	1110	1	0	0	1
15	1111	1	0	0	0

Le code de Gray est le code binaire réfléchi, c'est un cas très important des codes continu. Il est très fréquemment utilisé notamment sur les tables de Karnaugh pour simplifier les fonctions logiques. Remarquons que le code de Gray est continu au sens stricte, 0 est adjacent avec 15

Comme on le voit sur la figure, la table de Karnaugh est une table de vérité représentée d'une façon particulière. Les adresses des cases sont représentées en code de Gray de sorte à ce que toutes les cases qui ont un coté commun soient adjacentes. Les cases sur les extrémités opposées sont aussi adjacentes, on peut vérifier les adjacentes suivantes : 0-3, 7-4, 8-11, 15-12, 0-15, 1-14, 2-13, 3-12.

La table de Karnaugh peut être utilisée pour fabriquer des codes continus au sens large ou au sens stricte.

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

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

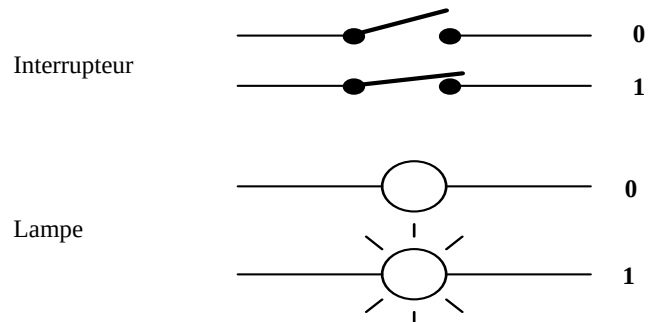
A	0	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0
B	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0
C	0	0	0	0	1	1	0	0	1	1	1	1	0	0	1	1
D	0	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0

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

CHAPITRE III -- VARIABLES ET FONCTIONS BOOLEENNE

III.1 -- Algèbre de Boole

C'est l'algèbre des grandeurs qui ne peuvent prendre que deux valeurs, 0 ou 1. L'algèbre de Boole est bien adaptée à l'étude des systèmes qui ne peuvent occuper que deux états. La correspondance entre l'état du système et la valeur 0 ou 1 est conventionnelle :



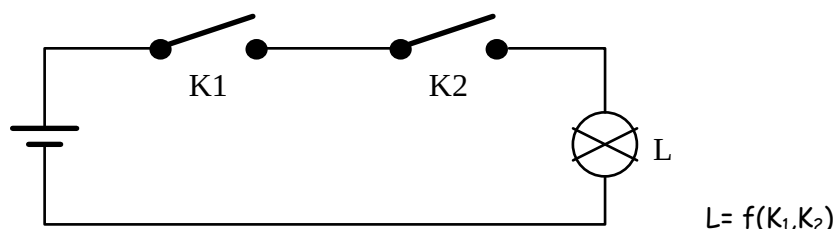
III.2 -- Variable booléenne

C'est une grandeur qui ne peut prendre que deux valeurs 0 ou 1.

III.3 -- Fonction booléenne

C'est une fonction de une ou plusieurs variables booléennes, ne pouvant prendre elle-même qu'une des deux valeurs 0 ou 1. Pour la définir, il faut préciser sa valeur pour toutes les combinaisons possible. Cela peut se faire de plusieurs façons :

- Par la donnée d'une figure illustrant le fonctionnement d'un système. Cela ne peut se faire que dans les cas simples.



- Par une définition écrite ou cahier des charges :
La lampe L s'allume lorsque les interrupteurs K_1 et K_2 sont fermés tous les deux. Elle est éteinte dans tous les autres cas.
- Par la donnée de la table de vérité

K_1	K_2	L
0	0	0
0	1	0
1	0	0
1	1	1

Le doublet des deux variables (K_1, K_2) peut prendre 2^2 valeurs distinctes. Dans le cas général de n variables, il y aura 2^n configurations possibles

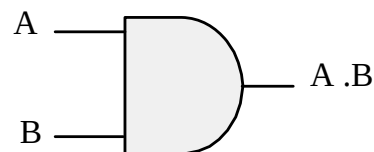
Les variables K_1 et K_2 sont dites les entrées du système, L est dite la sortie du système.

III.4 -- Opérateur Logiques

On peut montrer que toute fonction booléenne peut se synthétiser à partir d'un nombre très réduit de fonctions de deux variables ou l'extension à plusieurs variables de ces mêmes fonctions. Ces fonctions sont souvent appelées **Opérateurs**.

III.4.1 -- Opérateur ET (AND)

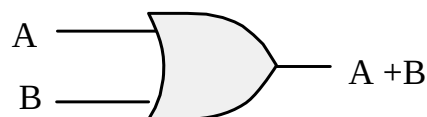
A	B	$A.B$
0	0	0
0	1	0
1	0	0
1	1	1



$A.B$ est vrai si et seulement si A est vraie **et** B est vraie

III.4.2 -- Opérateur OU (OR)

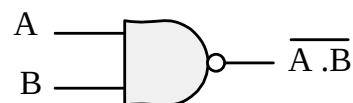
A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1



$A+B$ est vraie si au moins une des deux variables A **ou** B est vraie

III.4.3 -- Opérateur NON-ET (NAND)

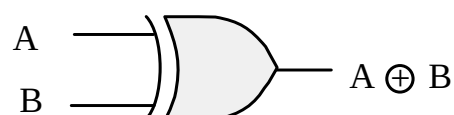
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



C'est le complément de l'opérateur ET. C'est l'opérateur le plus couramment utilisé dans la pratique.

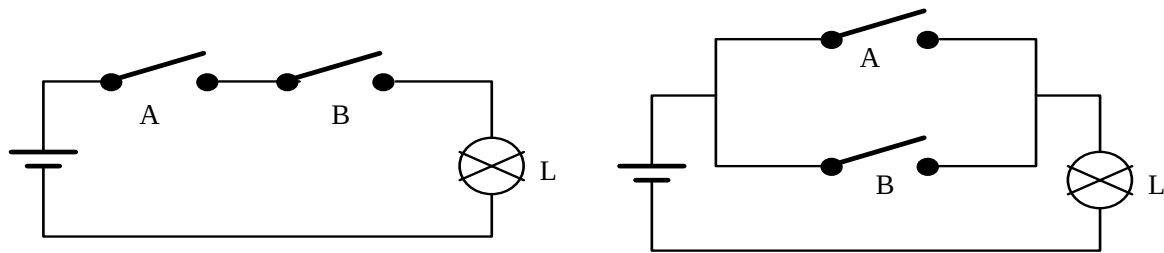
III.4.4 -- Opérateur OU EXCLUSIF (XOR)

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



$A \oplus B$ est vraie si une des deux variables A **ou** B est vraie mais pas les deux à la fois.

III.4.5 -- Analogie



Le schéma ci-dessus met en évidence l'analogie entre la mise en parallèle ou en série de contacteurs électriques et les opérateurs logiques AND et OR.

Pour le schéma de gauche, les deux contacteurs sont en série, on a $L = A \cdot B$

Pour le schéma de droite, les deux contacteurs sont en parallèle, on a $L = A + B$

III.5 -- Identités remarquables

$A + 0 = A$	$A \cdot 0 = 0$
$A + 1 = 1$	$A \cdot 1 = A$
$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$

- IDEMPOTANCE

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

- COMMUTATIVITE

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

- ASSOCIATIVITE

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

- ABSORPTION

$A \cdot (A + B) = A$

en effet $A \cdot (A + B) = AA + AB = A + AB = A \cdot (1 + B) = A$

- DISTRIBUTIVITE

- est distributive par rapport à $+$
- $+$ est distributive par rapport à $\cdot$

$A \cdot (B + C) = AB + AC$
$A + B \cdot C = (A + B) \cdot (A + C)$

$$\begin{aligned}
 (A + B) \cdot (A + C) &= A \cdot (A + C) + B \cdot (A + C) \\
 &= A + BA + BC \\
 &= A \cdot (1 + B) + BC \\
 &= A + BC
 \end{aligned}$$

• AUTRES IDENTITES

$$\begin{aligned} A.B + A.\bar{B} &= A \\ (A+B).(A+\bar{B}) &= A \end{aligned}$$

En effet : $A.B + A.\bar{B} = A.(B+\bar{B}) = A.1 = A$
 $(A+B).(A+\bar{B}) = A+(B.\bar{B}) = A+0 = A$

$$A + \bar{A}B = A + B$$

En effet : $A + \bar{A}B = (A + \bar{A}).(A + B) = 1.(A + B) = A + B$

III.6 -- Théorème de Morgan

Rappelons une fois de plus la table de vérité des opérateurs ET et OU. Ces tables peuvent se lire d'une façon différente de ce que nous avons l'habitude de faire :

- $A+B$ est fausse si et seulement si A et fausse **et** B est fausse
 - $A.B$ est fausse si et seulement si A et fausse **ou** B est fausse
- Ces deux phrases peuvent se traduire algébriquement par :

A	B	A+B	A.B
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

$$\begin{aligned} \overline{A+B} &= \bar{A}.\bar{B} \\ \overline{A.B} &= \bar{A} + \bar{B} \end{aligned}$$

III.7 -- Forme canonique

Soit la fonction S définie par la table de vérité ci-contre :

Cette table de vérité peut s'écrire de la façon algébrique suivante :

$$S(A,B,C) = A\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C + \bar{A}BC + ABC$$

A droite du signe $=$, on a ce qu'on peut appeler une expression logique, c'est un peut l'équivalent des polynômes qu'on a l'habitude d'écrire.

Il y a plusieurs façons différentes mais équivalentes pour représenter la fonction S . On peut vérifier sur la table qu'on a aussi $S = BC + AB + AC + \bar{A}C$.

La première expression de S est particulière, en effet, dans chaque **monôme**, figurent toutes les variables. Chaque monôme s'appelle **Minterme** de la fonction et la somme des mintermes s'appelle **forme canonique somme**.

La forme canonique produit revient à écrire l'expression de $\bar{S}$:

$$\bar{S}(A,B,C) = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C}$$

$$S(A,B,C) = \overline{\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C}}$$

$$S(A,B,C) = (A+B+C).(\bar{A}+B+C).(A+\bar{B}+C)$$

CBA	S(A,B,C)
000	0
001	0
010	0
011	1
100	1
101	1
110	1
111	1

III.8 -- Théorème de la disjonction

$$A.B = 0 \Rightarrow A \oplus B = A + B$$

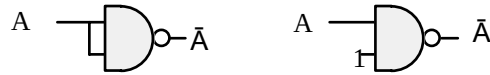
En effet, la forme canonique somme de l'opérateur OU est :

$$A+B = A\bar{B} + \bar{A}B + AB = A\bar{B} + \bar{A}B = A \oplus B \text{ si } A.B=0$$

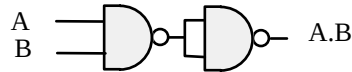
III.9 -- NAND : Opérateur universel

D'après la forme canonique, Nous pouvons exprimer toute fonction à l'aide de 3 opérateurs, AND, OR et INVERT (complémenteur). Voyons comment on peut faire pour n'utiliser que l'opérateur NAND pour exprimer ces fonctions.

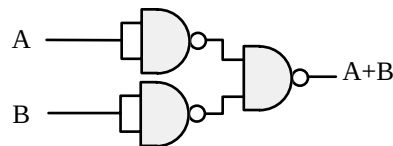
- INVERT :



- AND

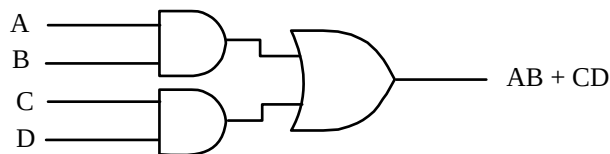


- OR

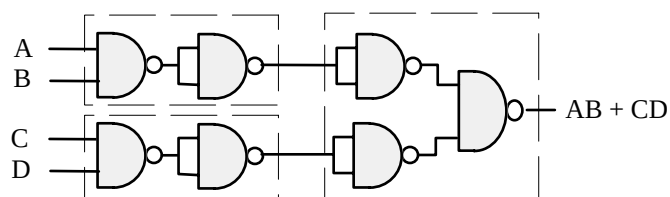


Méthode pratique :

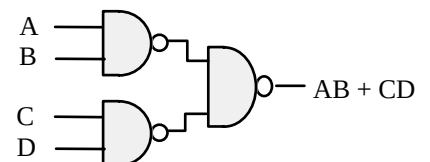
Si on écrit la forme canonique d'une fonction à deux variables, on obtient une expression de la forme : $AB + CD + \dots$ dont le schéma est le suivant :



Synthétisons chaque opérateur avec des NAND



Deux inverseurs qui se suivent peuvent bien sur être supprimés, on obtient le schéma suivant qui revient finalement à remplacer les AND et les OR par des NAND. Ceci n'est évidemment pas possible dans toutes les situations, la règle suivante, dite règles des fantômes en précise les conditions.

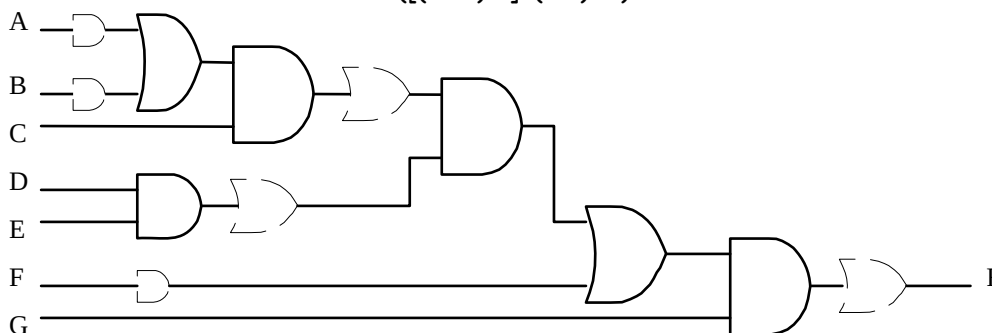


III.9.1 -- Règle des fantômes :

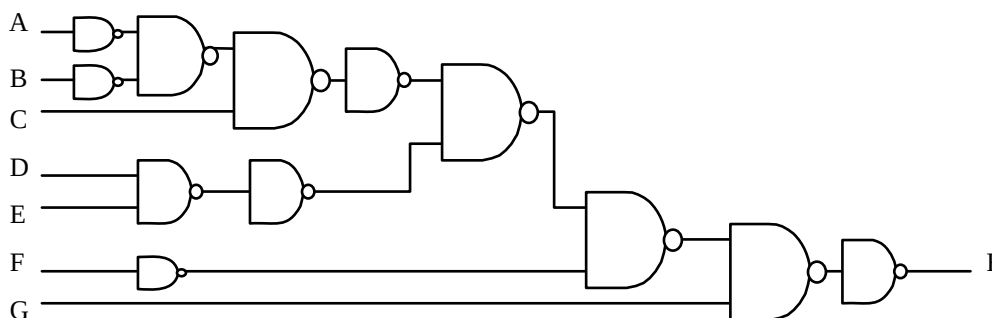
Un schéma AND-OR peut se transformer en un schéma NAND à condition qu'en suivant n'importe quel chemin entre une entrée et une sortie, on rencontre alternativement un AND et un OR. Le premier étant obligatoirement un AND et le dernier obligatoirement un OR. Si cette condition n'est pas vérifiée, on peut y remédier en intercalant des opérateurs A.A ou A+A aux endroits adéquats. Ceux ci ne changent rien au fonctionnement d'origine. On les appelle les opérateurs fantômes

Exemple : Pour être le plus général possible, on va prendre une expression qui n'est même pas sous forme canonique :

$$F = [(A+B).C]. (DE+G).H$$



Les opérateurs en trait découpé sont les opérateurs fantômes, ils sont rajoutés pour respecter la règle. Tous les opérateurs sont ensuite transformés en NAND.

**CHAPITRE IV -- SIMPLIFICATION DES FONCTIONS LOGIQUE****IV.1 -- Méthode Karnaugh (1953)**

Il s'agit de chercher la fonction la plus simple d'une fonction booléenne. Ceci bien sur dans le but d'avoir une réalisation avec un nombre minimum d'opérateurs.

Plusieurs méthodes existent, la plus simple est la méthode de Karnaugh (1953)

IV.1.1 -- Diagramme de Karnaugh

C'est la table de vérité disposée d'une manière particulière. La voici pour 2, 3 puis 4 variables:

A \ B	0	1
0		
1		

A \ BC	00	01	11	10
0				
1				

AB \ CD	00	01	11	10
00				
01				
11				
10				

Chaque case correspond à une ligne de la table de vérité, Les "coordonnées de la case représente l'adresse de la case. Elles sont représentées en code de Gray. Deux cases sont appelées adjacentes dans le diagramme de Karnaugh si elles ont un coté commun.

Sur le diagramme si contre, les cases K et L, M et N, P et Q, Q et R sont respectivement adjacentes. La propriété caractéristique du diagramme de Karnaugh est que les adresse de deux cases adjacentes sont des nombres adjacents, quand on passe de l'un à l'autre il n'y a qu'un bit qui change :

$K \leftrightarrow 0001$ $P \leftrightarrow 1100$ $M \leftrightarrow 1111$ $Q \leftrightarrow 1000$

$L \leftrightarrow 0101$ $Q \leftrightarrow 1000$ $N \leftrightarrow 1110$ $R \leftrightarrow 1001$

Remarquons que les cases P et N ainsi que K et R ont des adresses adjacentes :

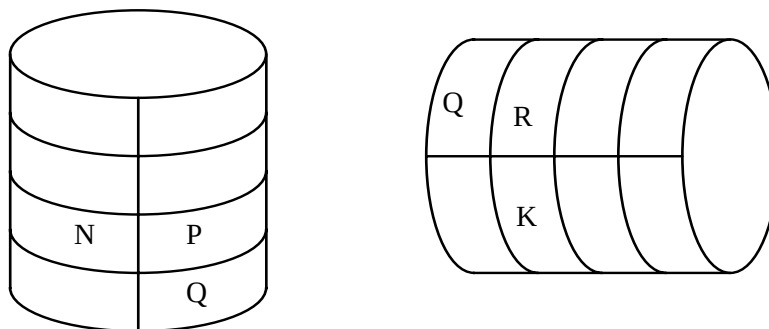
$P \leftrightarrow 1100$ $K \leftrightarrow 0001$

$N \leftrightarrow 1110$ $R \leftrightarrow 1001$

AB \ CD	00	01	11	10
	00	01	11	10
00		K		
01		L		
11	P		M	N
10	Q	R		

On généralise la notion d'adjacence au niveau du diagramme en disant que deux cases sont adjacentes quand leurs adresses le sont. Ainsi les cases de l'extrémité droite sont adjacentes à celles de l'extrémité gauche et les case de l'extrémité supérieure sont adjacentes à celles de l'extrémité inférieure.

Cela se passe comme si on enroulait la feuille de papier sur laquelle est dessiné le diagramme de Karnaugh d'abord horizontalement puis verticalement.



IV.1.2 -- Remplissage de la table de Karnaugh à partir de la table de vérité

Cela ne pose aucun problème à condition de ne pas changer le "poids logique" de chaque variable en passant de la table de vérité à la table de Karnaugh. Les tableaux ci-dessous montrent la correspondance entre les lignes de la table de vérité et les cases de la table de Karnaugh selon la disposition des variables ABCD dans les cases adresses.

	D	C	B	A	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0

	10	11	12	13	14	15
10	1	0	1	0	0	0
11	1	0	1	1	1	0
12	1	1	0	0	0	0
13	1	1	0	1	1	1
14	1	1	1	0	0	0
15	1	1	1	1	1	1

AB \ CD	00	01	11	10
00	0	8	12	4

01	2	10	14	6
11	3	11	15	7
10	1	9	13	5

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
----	---	---	----	----

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

Voici différentes dispositions de la table de Karnaugh de la fonction F

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

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

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

IV.1.3 -- Remplissage de la table de Karnaugh à partir d'une fonction.

Cela ne pose aucun problème si la fonction est définie par sa forme canonique, il suffit alors de placer des 1 dans les cases correspondantes aux monômes où la fonction est vraie et des 0 ailleurs. Pour une fonction de 4 variables, chaque monôme contient les 4 variables (complémentées ou non).

Dans le cas d'une fonction représentée sous forme non canonique, le nombre de 1 correspondant à chaque monôme dépend du nombre de variable figurant dans le monôme, ou plus précisément du complément du nombre de variable du monôme par rapport au nombre de variable de la fonction.

Soit la fonction suivante : $G(A,B,C,D) = \bar{A}BC\bar{D} + \bar{B}CD + AC + \bar{B}$

- Pour le monôme $\bar{A}BC\bar{D}$ correspond un 1 dans la case $ABCD=0110$
- Le monôme $\bar{B}CD$ est vrai pour $BCD=011$, $A=0$ et $BCD=011$, $A=1$. Il y aura donc un 1 dans la case $ABCD=0011$ et un autre dans la case $ABCD=1011$.
- De la même façon, pour le monôme AC , il y aura un 1 dans toutes les case où on a $AC=11$ soit $ABCD=1010$ ou 1011 ou 1110 ou 1111 .
- Pour le monôme $\bar{B}$, il y aura un 1 partout où on a $B=0$.

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

On remarque donc que si n est le nombre de variables de la fonction est m est le nombre de variable qui constituent un monôme, le nombre de 1 correspondant à ce monôme dans le diagramme de Karnaugh est de 2^{n-m} .

IV.1.4 -- Lecture du digramme de Karnaugh

Soit la fonction H définie par la table ci-contre. Nous savons que plusieurs 1 peuvent correspondre à un seul monôme, mais ce n'est pas toujours facile de trouver quel est ce monôme. Commençons par écrire un monôme complet pour chaque 1 :

$$H = \bar{A}\bar{B}\bar{C}D + \bar{A}BCD$$

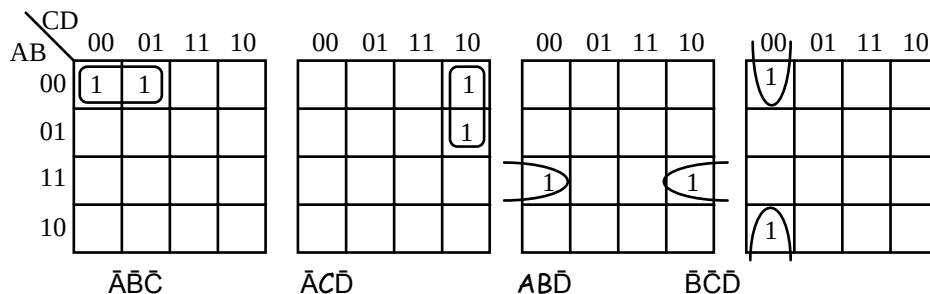
Cette expression peut être simplifiée

$$H = \bar{A}BD.(C+\bar{C}) = \bar{A}BD$$

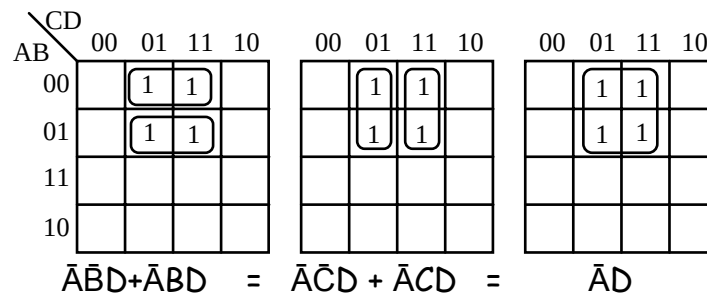
Nous nous baserons donc sur la règle $XY+X\bar{Y}=X$ pour faire les simplifications. C'est la variable qui change quand on passe d'un monôme à l'autre qui ne figure pas dans le monôme résultant.

Sur le diagramme cela se concrétise comme suit :

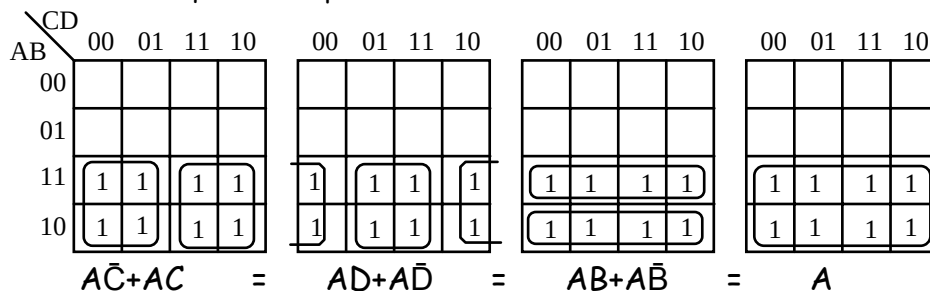
- Deux cases adjacentes groupées donne un rectangle à deux cases auquel correspond un monôme de 3 variables



- Deux rectangles à deux cases peuvent à leur tour être adjacents, ils seront groupés en un rectangle de 4 cases auquel correspond un monôme à 2 variables.



- De la même manière, 2 rectangles de 4 cases peuvent à leur tour être adjacents, ils seront groupés en un rectangle de 8 cases auquel correspond un monôme à une variable.



- De la même manière, 2 rectangles de 8 cases peuvent à leur tour être groupés en un rectangle de 16 cases, dans ce cas la fonction est toujours vraie, **fonction** = 1 et on n'a même pas besoin de dresser un diagramme de Karnaugh.

Reprenons la fonction G citée précédemment et partons du diagramme de Karnaugh pour arriver à l'expression logique la plus simple. On essaye de faire le plus grand rectangle possible car on sait maintenant que plus le rectangle, plus le monôme qui lui correspond est 'petit'. L'expression la plus simple de la fonction G est donc : $G = \bar{B} + AC + C\bar{D}$

		CD			
AB		00	01	11	10
	00	1	1	1	1
	01				1
	11			1	1
	10	1	1	1	1

Règle de groupement :

Pour obtenir l'expression la plus simple d'une fonction booléenne donnée par son diagramme de Karnaugh, il faut respecter les points suivants :

- Tous les 1 **doivent** être entourés
- Les 0 **ne doivent pas** être entourés
- Il faut faire les plus grands groupements possibles
- Les x peuvent être entourés ou non selon le besoin
- Les 1 et les x peuvent être entourés plusieurs fois
- On s'arrête dès que tous les 1 sont entourés

IV.1.5 -- Code Interdit ou quand la valeur compte peu

Étudions le cas suivant : On désire maintenir la température d'une salle de travail entre 17°C et 20°C . Nous nous procurons deux capteurs de température ayant chacun une sortie logique que nous appellerons C_{17} et C_{20} . Chaque capteur fonctionne de la manière suivante :

$C_{17} = 0$ si la température est $< T_{17}$, $C_{17} = 1$ si la température est $> T_{17}$

$C_{20} = 0$ si la température est $< T_{20}$, $C_{20} = 1$ si la température est $> T_{20}$

On va donc essayer de construire un système logique qui délivre une alarme S chaque fois que la température sort de l'intervalle $[17^{\circ}, 20^{\circ}]$. Ce qui revient à chercher la fonction S des deux variables C_{17} et C_{20} , $S = f(C_{17}, C_{20})$.

$T^{\circ}\text{C}$

	C_{17}	C_{20}	S
20	1	1	1
17	1	0	0
	0	0	1

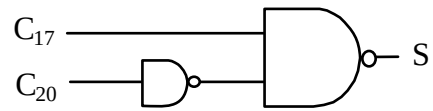
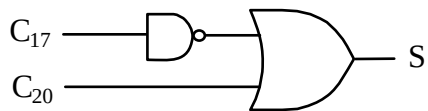
Construisons la table de vérité

Cette table de vérité n'est pas complète car avec deux variables on doit avoir 4 lignes dans la table de vérité. Il manque la ligne $C_{17} C_{20} = 01$ qui correspond à une situation où la température est inférieure à 17°C et en même temps supérieure à 17°C , chose évidemment impossible, d'où le nom de code interdit. Pour ce qui concerne la sortie S correspondant à $C_{17} C_{20} = 01$, on peut mettre ce qu'on veut du moment que cette situation ne se présentera pas de toute façon. On met alors un x ou tout autre symbole pour préciser que la valeur de la fonction à cet endroit ne compte pas, on peut prendre $S=0$ ou $S=1$ indifféremment.

Dans la table de Karnaugh, les x peuvent être entourés ou non selon les besoins. Si un x est entouré, cela veut dire qu'on a choisi la valeur 1 pour la fonction. S'il n'est pas entouré, cela veut dire qu'on a choisi la valeur 0. Les x seront entourés chaque fois que cela peut servir à agrandir un groupement contenant des 1. Il est évident qu'il ne faut pas constituer des groupements ne contenant que des x. On obtient :

		C_{20}	
C_{17}		0	1
	0	1	x
	1	0	1

$$S = \bar{C}_{17} + C_{20}$$



IV.2 -- Exemple 1 : Convertisseur Binair naturel vers code de Gray

D	C	B	A	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

On voit sur la table de vérité que $G_3=D$.

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

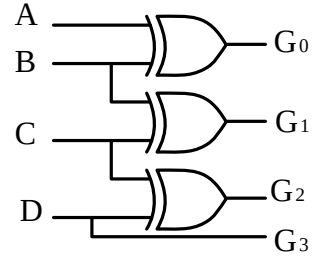
$$G_2 = C\bar{D} + \bar{C}D = C \oplus D$$

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

$$G_1 = B\bar{C} + \bar{B}C = B \oplus C$$

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

$$G_0 = A\bar{B} + \bar{A}B = A \oplus B$$



IV.3 -- Exemple 2 : Synthèse d'un système logique

Pour transporter le sable d'une position A vers une position B, on utilise le système suivant;

Un chariot peut se déplacer sur un rail de chemin de fer sous l'action d'un moteur électrique M ($M=0 \Rightarrow$ moteur arrêté, $M=1 \Rightarrow$ moteur en marche). Le déplacement du chariot se fait toujours dans le même sens. Deux capteurs A et B permettent de déterminer la position du chariot ($A=1 \Rightarrow$ chariot en position A, $A=0 \Rightarrow$ le chariot n'est pas en position A, le capteur B fonctionne de la même façon que A).

Un capteur P placé sur le chariot permet de savoir si le chariot est plein ou vide, ($P=1 \Rightarrow$ chariot plein, $P=0 \Rightarrow$ chariot vide).

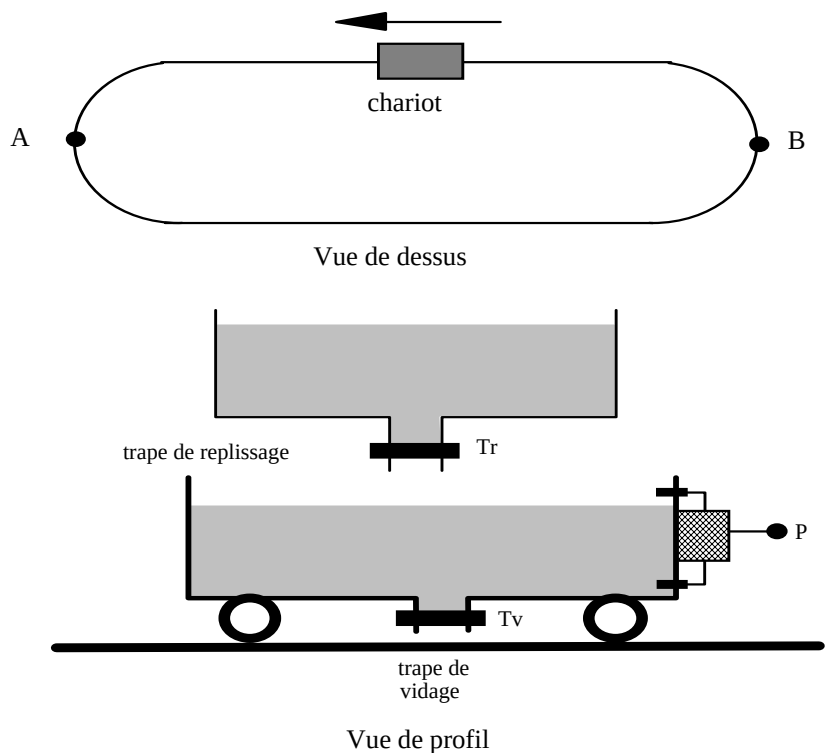
Le chariot peut être rempli à l'aide d'une trappe qu'on peut actionner avec la commande Tr ($Tr=1 \Rightarrow$ trappe ouverte, $Tr=0 \Rightarrow$ trappe fermée). Par abus de langage, cette trappe sera appelée trappe Tr.

Le chariot peut être vidé à l'aide d'une trappe qu'on peut actionner par la commande Tv ($Tv = 1 \Rightarrow$ trappe ouverte, $Tv = 0 \Rightarrow$ trappe fermée). Par abus de langage, cette trappe sera appelée trappe Tv.

Le fonctionnement se fait de la manière suivante :

Le chariot vide arrive en A ($A=1, P=0$), le moteur M s'arrête, la trappe de remplissage Tr s'ouvre. Une fois le chariot plein, Tr se ferme et le chariot démarre. Quand le chariot arrive en B, il s'arrête et la trappe Tv s'ouvre. Quand le chariot est vide, la trappe Tv se ferme et le chariot démarre. Quand il arrive en A le cycle recommence identique à lui-même.

Faire l'étude du système qui permet de générer les commandes M, Tr et Tv à partir des entrées A, B et P.



A	B	P	M	Tr	Tv	Obsevation
0	0	0	1	0	0	Vide, se déplace vers A, trapes fermées
0	0	1	1	0	0	Plein, se déplace vers B, trapes fermées
0	1	0	1	0	0	Vide en B, il faut démarrer
0	1	1	0	0	1	Plein en B, il faut s'arreter et vider
1	0	0	0	1	0	Vide en A il faut remplir
1	0	1	1	0	0	Plein en A, il faut partir, trapes fermées
1	1	0	x	x	x	Ipossible, ne peut être en A et en B
1	1	1	x	x	x	Ipossible, ne peut être en A et en B

$$M = \bar{B}P + \bar{A}\bar{P}$$

$$Tr = A\bar{P}$$

$$Tv = BP$$