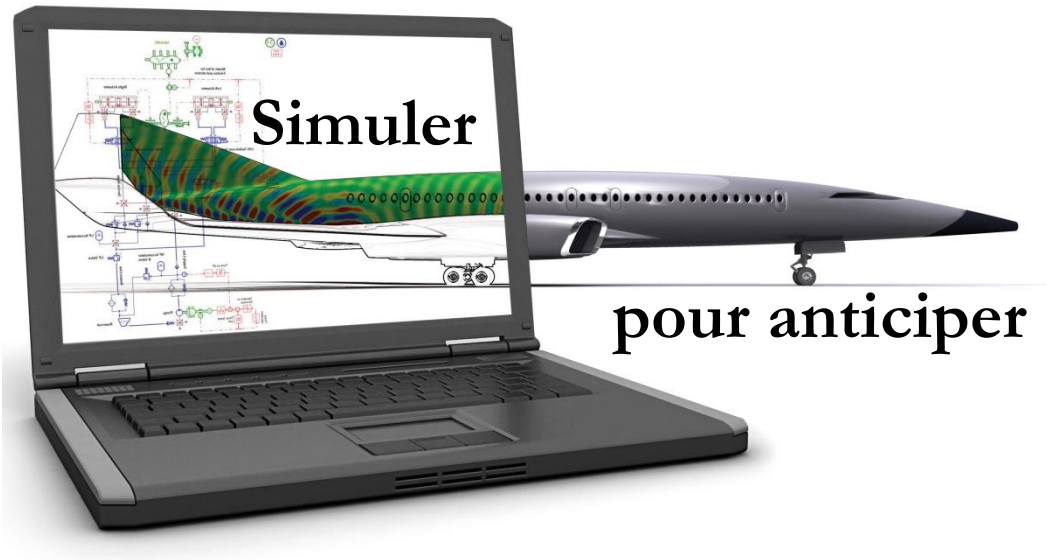
SYSTEMES à évènements discrets COMBINATOIRES

Objectifs - extraits du référentiel de 1ère et 2nde année (MP et PSI) pour les systèmes à évènements discrets

Compétence visée	*Savoir-faire associé*
Appréhender les analyses fonctionnelle et structurelle	*Interpréter tout ou partie de l'évolution temporelle d'un système à partir des diagrammes de séquence et d'états*
Proposer un modèle de connaissance et de comportement	*Exprimer un fonctionnement par des équations logiques (ET, OU, NON)*
	Représenter tout ou partie de l'évolution temporelle sous forme de chronogramme
	Décrire et compléter un algorithme représenté sous forme graphique



Sciences Industrielles de l'Ingénieur
1ère année de CPGE
Lycée Claude Fauriel

1 - Variables binaires:

De nombreux composants utilisés en automatisme ne peuvent normalement prendre que deux états différents: lampe allumée ou éteinte, bouton-poussoir actionné ou relâché, moteur tournant ou à l'arrêt, vérin pneumatique sorti ou rentré...

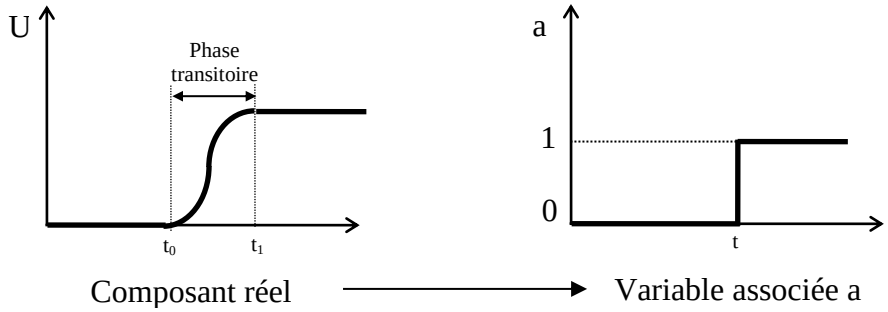
A chacun de ces composants, on peut associer une variable **binaire** ou **logique** qui ne peut prendre que deux valeurs notées *0* ou *1* (*vrai* ou *faux*, *oui* ou *non*). La convention habituellement utilisée est d'associer l'état *1* de la variable à la situation "actionnée", "activée", "en travail" ou "en énergie" du composant. L'état *0* est alors associé à la situation "relâchée", "désactivée", "au repos" ou "hors énergie" du composant.

Exemple : voyant lumineux allumé $\Rightarrow v = 1$
voyant lumineux éteint $\Rightarrow v = 0$

Remarques:

* le comportement "Tout ou Rien" (TOR) ne correspond qu'au comportement normalement prévu en régime stabilisé et en l'absence de tout dysfonctionnement.

* l'association d'une variable binaire à un composant ne peut pas rendre compte des états transitoires apparaissant entre deux états stables. C'est donc une simplification du comportement réel.



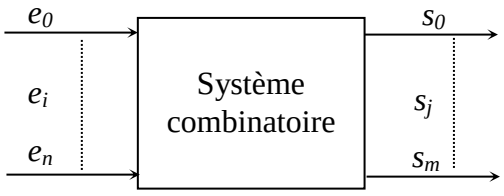
Exemple : les Automates Programmables Industriels (API) sont des appareils très utilisés dans la réalisation de commandes de systèmes automatisés. Par l'intermédiaire de modules d'entrées TOR, ils reçoivent des informations sous forme de signaux électriques. L'API travaille donc sur une variable binaire associée à chaque entrée. Il importe donc que le constructeur définisse parfaitement les paramètres reconnus par le module d'entrée comme étant caractéristiques des niveaux 0 et 1.

niveau 0 : U < 7 Volts et I < 3 mA
niveau 1 : 15 V < U < 30 V et 5 mA < I < 7 mA

Le constructeur de l'API ne garantit rien pour tout signal de valeur intermédiaire.

2 – Système combinatoire:

Les entrées e_i et les sorties s_j d'un système combinatoire sont des variables binaires.



$\forall t,\ s_j = f(e_i)$
f = fonction logique indépendante du temps
Pour traiter de tels systèmes, on utilise l'algèbre de Boole (mathématicien anglais 1815-1864).

3 – Fonctions logiques fondamentales:

On définit une algèbre binaire sur l'ensemble $\mathcal{E} = \{0,1\}$

3.1 – Fonction **OUI** :

$\forall$ a $\in \mathcal{E}$, OUI(a) = a

table de vérité:

a	OUI(a)
0	0
1	1

3.2 – Fonction **NON** (NOT) :notée "-"

$\forall$ a $\in \mathcal{E}$, a = 0 $\Leftrightarrow$ $\bar{a}$ = 1

$\bar{a}$ (lire *a barre*) est le "complément" de a: $\bar{\bar{a}}$ = a

a	$\bar{a}$
0	1
1	0

3.3 – Fonction **OU** (OR) : somme logique notée "+"

$\forall$ (a,b) $\in \mathcal{E}x\mathcal{E}$, a + b = 1 $\Leftrightarrow$ l'un au moins des deux éléments a ou b vaut 1

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	1

Remarque: si a et b valent simultanément 1, alors a+b = 1 $\Rightarrow$ on appelle aussi cette fonction le "OU inclusif" par opposition au "OU exclusif" ou "XOR" noté $\oplus$:

$\forall$ (a,b) $\in \mathcal{E}x\mathcal{E}$, a $\oplus$ b = 1 $\Leftrightarrow$ un seul des deux éléments a ou b vaut 1

a	b	a$\oplus$b
0	0	0
0	1	1
1	0	1
1	1	0

3.4 – Fonction **ET** (AND) : produit logique notée "."

$\forall$ (a,b) $\in \mathcal{E}x\mathcal{E}$, a . b = 1 $\Leftrightarrow$ a et b valent simultanément 1

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

✍ *Etablir la table de vérité de la fonction* $f = a\bar{b} + \bar{a}b$

a	b	$a\bar{b}$	$\bar{a}b$	f
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

$\bar{a}\,b$

$a\,\bar{b}$

$= \bar{a}\,b + a\,\bar{b} = a \oplus b$

Remarque: on peut trouver la forme canonique d'une fonction logique (somme de produits des variables ou de leur complément) à partir de la table de vérité.

3.5 – Propriétés et théorèmes: (✍ *à vérifier avec les tables de vérité*)

commutativité	$a + b = b + a$	$a\,b = b\,a$
distributivité	$a + (b\,c) = (a + b)(a + c)$	$a\,(b + c) = (a\,b) + (a\,c)$
élément neutre	$a + 0 = a$	$a \cdot 1 = a$
complémentarité	$a + \bar{a} = 1$	$a \cdot \bar{a} = 0$
idempotence	$a + a = a$	$a \cdot a = a$
élément absorbant	$a + 1 = 1$	$a \cdot 0 = 0$
absorption	$a + a\,b = a$	$a\,(a + b) = a$
associativité	$a + (b + c) = (a + b) + c = a + b + c$	$a\,(b\,c) = (a\,b)\,c = a\,b\,c$
Théorèmes de De Morgan	$\overline{a+b} = \bar{a}.\bar{b}$	$\overline{a.b} = \bar{a} + \bar{b}$

✍ *Déterminer* $\overline{a \oplus b}$ *de deux façons différentes.*

✍ *Simplifier les expressions suivantes:*

$f_1 = \bar{a}b + a$ $\quad f_2 = \bar{a}\,\bar{c}\,\bar{d} + \bar{a}\,b\,\bar{c} + a\,b\,\bar{c} + \bar{c}\,d\,\bar{a}$ $\quad f_3 = c\bar{d}b + \bar{c}d\bar{a}b + b\bar{c} + abc + \bar{a}bcd$

3.6 – Fonction **NAND** (NON ET) :

a NAND b = $\overline{\mathbf{a\,.\,b}}$ = $\bar{\mathbf{a}} + \bar{\mathbf{b}}$

3.7 – Fonction **NOR** (NON OU) :

a NOR b = $\overline{\mathbf{a + b}}$ = $\bar{\mathbf{a}}\,.\,\bar{\mathbf{b}}$

Ces deux fonctions sont universelles: elles permettent d'écrire n'importe qu'elle autre fonction.

✍ *Ecrire* $\bar{a}$, $a+b$, $a.b$ *à l'aide la fonction NAND. Idem avec NOR.*

4 – Tableaux de Karnaugh (sans doute hors programme):

Chaque case du tableau correspond à une ligne de la table de vérité d'une fonction logique: une fonction à n variables est donc représentée par un tableau à 2^n cases agencé de telle façon qu'une seule variable change de valeur quand on passe d'une case à une case adjacente.

$n = 2$

b \ a	0	1
0	0	1
1	0	0

$f_4 = a\,\bar{b}$

$n = 3$

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

$f_5 = \bar{a}\,b\,\bar{c} + a\,b\,\bar{c} = b\bar{c}$

$n = 4$

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

$f_6 = a\bar{b}cd + abcd + ab\bar{c}d + a\bar{b}\bar{c}d$
$= ad$

Les tableaux de Karnaugh permettent de représenter des fonctions logiques et de les simplifier en regroupant des cases adjacentes contenant des 1(en nombre égal à une puissance de 2).

✍ *Simplifier* $f_7 = \bar{a}b\bar{c}+ab\bar{c}+\bar{a}\bar{b}c+a\bar{b}c$, f_2 *et* f_3.

Remarque1: dans un tableau à *n* variables
- 1 case correspond au produit des *n* variables ou de leur complément
- 2 cases adjacentes correspondent au produit de *n*-1 variables
- $2^2 = 4$ cases adjacentes correspondent au produit de *n*-2 variables
- …
- 2^p cases adjacentes correspondent au produit de *n-p* variables.

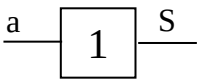
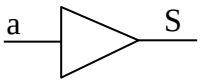
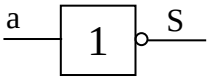
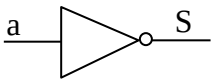
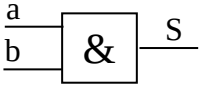
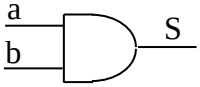
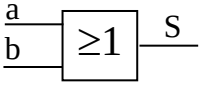
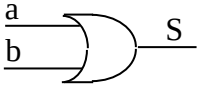
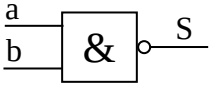
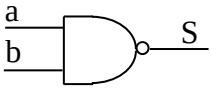
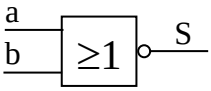
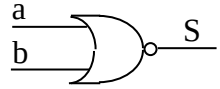
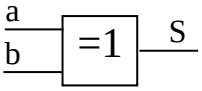
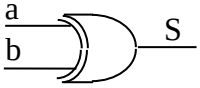
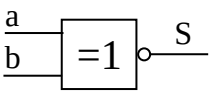
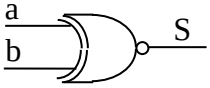
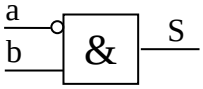
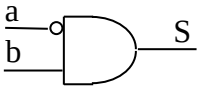
Remarque2: si un tableau contient peu de 0, il peut être intéressant de regrouper les 0 (au lieu des 1) et on trouve alors $\bar{f}$.

✍ *Déterminer l'équation logique de la fonction* f_8 *donnée par le tableau de Karnaugh:*

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

5 – Représentations des fonctions logiques (sans doute hors programme):

5.1 – Représentation symbolique des fonctions élémentaires (logigramme):

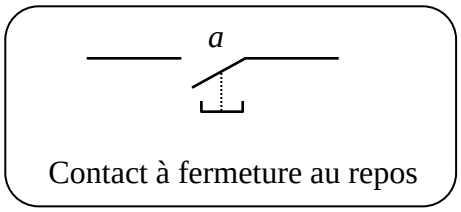
Opérateur	Fonction logique	Norme IEC (internationale)	Norme américaine
OUI	S = a		
NON	$S = \bar{a}$		
ET	S = a b		
OU	S = a + b		
NAND	$S = \overline{a\,b} = \bar{a}+\bar{b}$		
NOR	$S = \overline{a+b} = \bar{a}\,\bar{b}$		
XOR	$S = a \oplus b$		
IDENTITE	$S = \overline{a \oplus b}$		
INHIBITION	$S = \bar{a}\,b$		

✍ *Représenter le logigramme (ou schéma logique) des fonctions suivantes:*

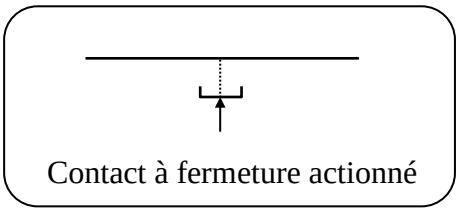
$f_9 = (\bar{c}\, b + a)(c\,\bar{a} + b)$ *et* $f_{10} = \overline{\bar{a} + \bar{b}} + \bar{c}\, d$

5.2 – Représentation électrique:

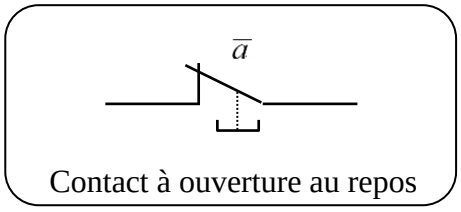
5.2.1 – Contact à fermeture: c'est un contact électrique (interrupteur) qui est normalement ouvert (au repos) et qui se ferme lorsqu'il est actionné. On le désigne généralement par des lettres minuscules *a*, *b*…



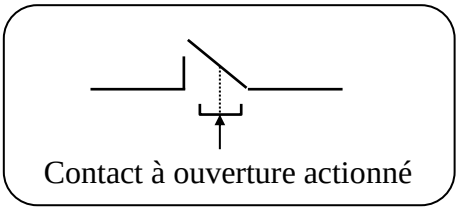
Contact à fermeture au repos


Contact à fermeture actionné

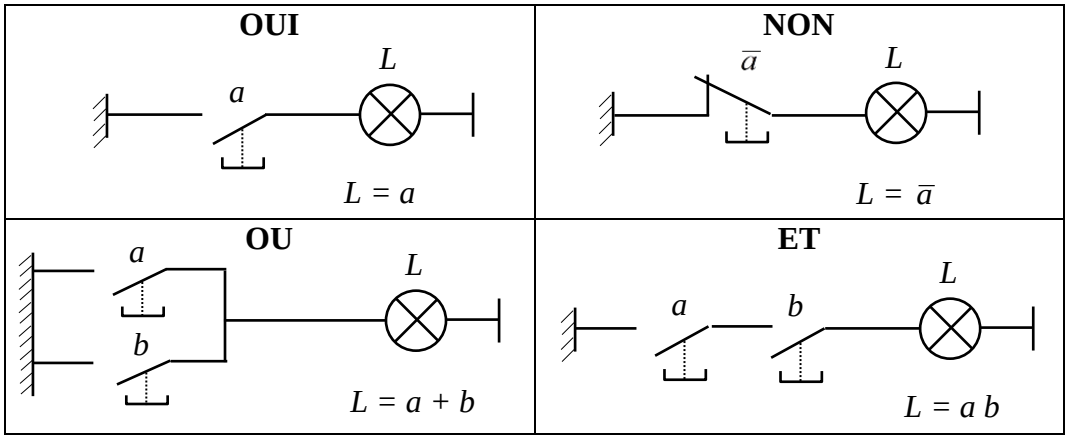
5.2.2 – Contact à ouverture: c'est un contact électrique qui est normalement fermé (au repos) et qui s'ouvre lorsqu'il est actionné. On le désigne généralement par $\bar{a}$, $\bar{b}$ …



Contact à ouverture au repos

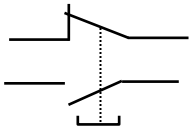

Contact à ouverture actionné

5.2.3 – Représentation des fonctions élémentaires:



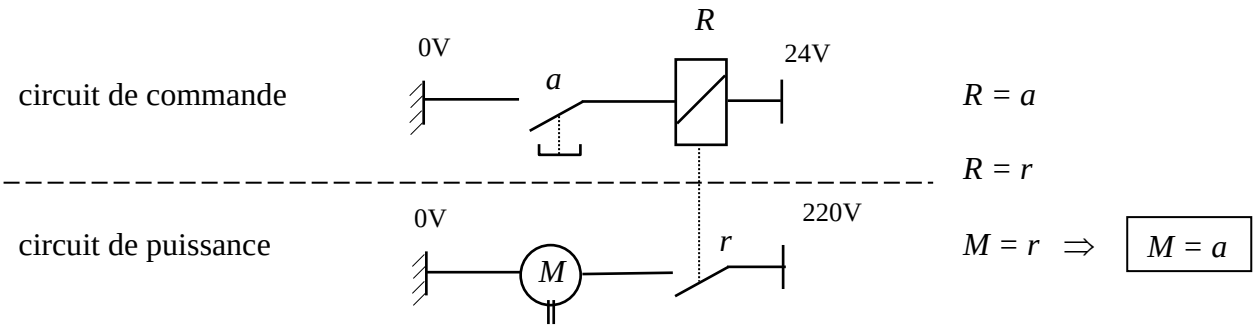
✍ *Représenter les fonctions NAND, NOR, XOR et IDENTITE avec des schémas à contacts.*

NOTA: on peut représenter *a* et $\bar{a}$ en même temps:

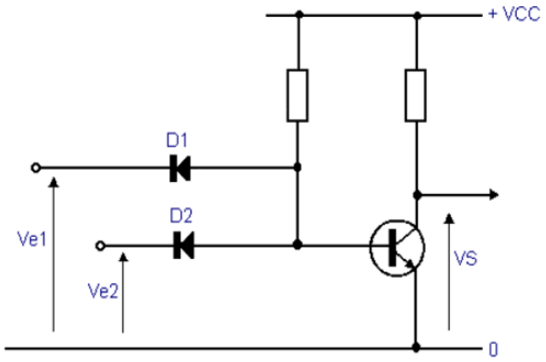


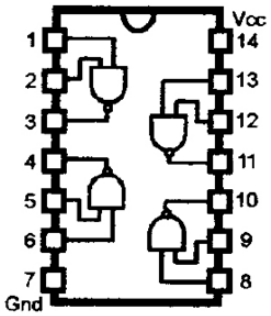
5.2.4 – Relais électrique:

Un relais (ou contacteur à relais) est constitué d'un circuit de commande comportant une bobine d'excitation électromagnétique *R* (pilotant un contact *r*) et d'un circuit d'utilisation: il permet ainsi de dissocier les deux circuits.

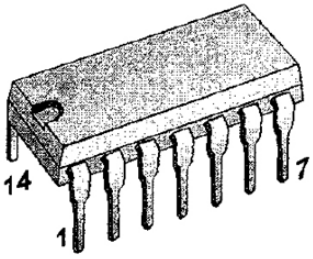


5.3 – Logique électronique : exemple de NAND



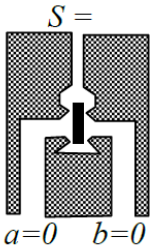


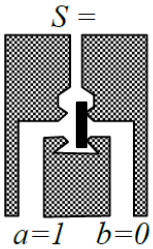
Circuit intégré (TTL)

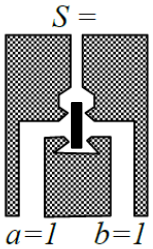


5.4 – Logique pneumatique:

5.4.1 Fonction ….

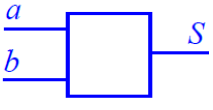




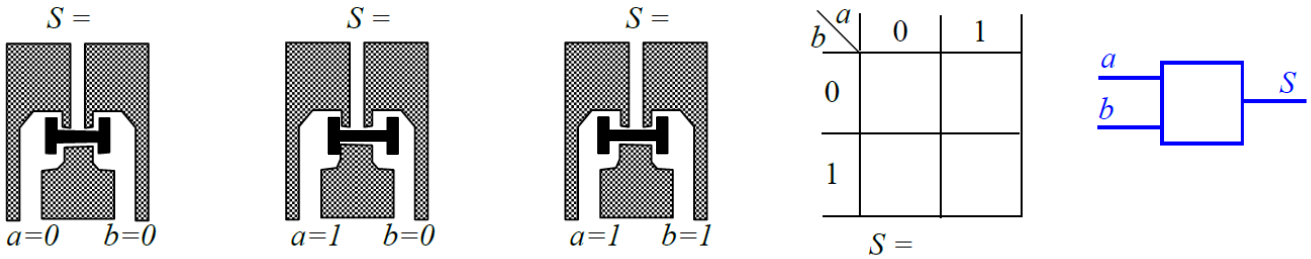


b \ a	0	1
0		
1		

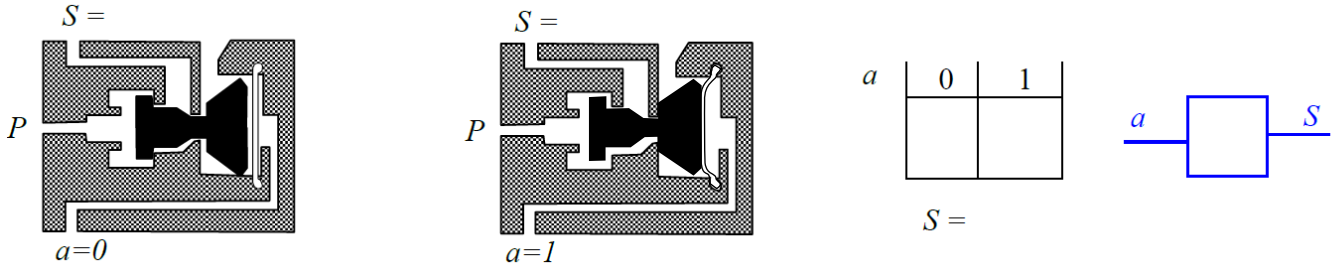
S =



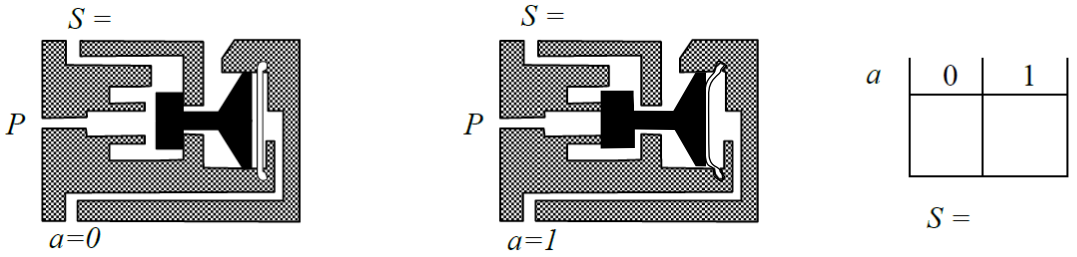
5.4.2-Fonction ….



5.4.3-Fonction ….



5.4.4-Fonction ….



5.4.5- Fonction ….

6 - Codages:

6.1 - Systèmes de numération:

Représentation d'un nombre N en base b: $N_{(b)} = a_n b^n + a_{n-1} b^{n-1} + ... + a_0 b^0$

b^i représente le "poids" du bit n°i.

- système binaire : b = 2 Base 2 = {0,1} = {a_i}
- système décimal : b = 10 Base 10 = {0,1,2,3,4,5,6,7,8,9}
- système hexadécimal : b = 16 Base 16 = {0,1,2,3...,9,A,B,C,D,E,F}

Exemple : $(29)_{10} = 1.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 \longrightarrow (11101)_2$

$16 + 8 + 4 + 0 + 1$

$= 1.16^1 + 13.16^0 \longrightarrow (1D)_{16}$

Le système binaire est celui des systèmes informatiques qui travaillent avec des données logiques 0 et 1.

L'écriture des nombres dans cette base donnant un grand nombre de chiffres, les informaticiens lui préfèrent le système hexadécimal qui permet de contracter les informations en paquets de 4 bits.

Dans l'exemple ci-dessus : $(0001\ 1101)_2 = (1\ D)_{16}$ Notations : *29 = %11101 = $1D*

6.2 - Codage de l'information :

Le traitement automatique de l'information nécessite un système de codage de cette information: gestion de marchandises par codes à barre, tri du courrier, commande numérique des machines-outils...

6.2.1 - Code binaire naturel:

	2^3	2^2	2^1	2^0
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

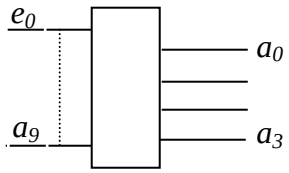
6.2.2 - Code binaire réfléchi ou code Gray:

0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1

Un seul bit change d'état à chaque incrémentation.
L'intérêt de ce code réside dans des applications d'incrémentation où il minimise le risque d'erreur.

6.2.3 - Code ASCII (*American Standard Code For Information Interchange*):
Il permet de coder 256 caractères sur 8 bits.
Exemple : A est codé 65 en décimal , %01000001 en binaire, $41 en hexadécimal.

6.3 - Codeur : il s'agit de convertir des informations données par l'opérateur en informations compréhensibles par la machine.



Exemple : codeur décimal qui transforme un chiffre de 0 à 9 (= entrée e_i) en nombre binaire donc codé sur 4 bits ($a_3\ a_2\ a_1\ a_0$) .

Entrées *Sorties*

	a_3	a_2	a_1	a_0
e_0	0	0	0	0
e_1	0	0	0	1
e_2	0	0	1	0
e_3	0	0	1	1
e_4	0	1	0	0
e_5	0	1	0	1
e_6	0	1	1	0
e_7	0	1	1	1
e_8	1	0	0	0
e_9	1	0	0	1

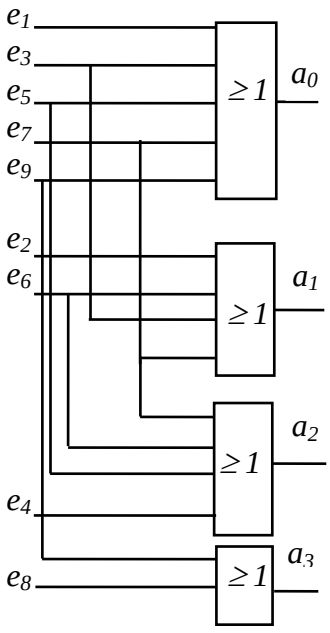
table de vérité

$a_0 = e_1 + e_3 + e_5 + e_7 + e_9$

$a_1 = e_2 + e_3 + e_6 + e_7$

$a_2 = e_4 + e_5 + e_6 + e_7$

$a_3 = e_8 + e_9$



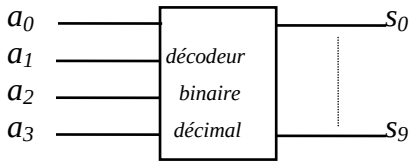
Logigramme du codeur

6.4 - Décodeur : après traitement de l'information, il faut décoder le résultat.

Exemple : décoder un résultat codé sur 4 bits en chiffre décimal

Entrées				*Sorties*
a_3	a_2	a_1	a_0	
0	0	0	0	s_0
0	0	0	1	s_1
0	0	1	0	s_2
0	0	1	1	s_3
0	1	0	0	s_4
0	1	0	1	s_5
0	1	1	0	s_6
0	1	1	1	s_7
1	0	0	0	s_8
1	0	0	1	s_9

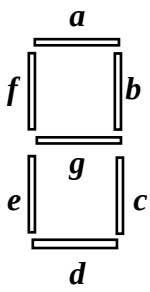
table de vérité



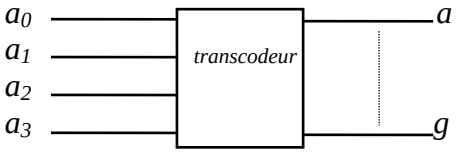
$s_7 = \overline{a_3}\, a_2\, a_1\, a_0$

6.5 - Transcodeur : c'est l'opération qui consiste à passer d'un code à un autre.

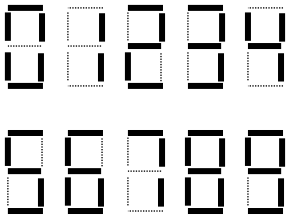
Exemple: un calcul numérique étant fait sur 4 bits a_i, on veut présenter le résultat sur un afficheur à 7 segments. Donner l'équation logique de chaque segment.



Afficheur 7 segments



4 entrées *7 sorties*



N	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

a	b	c	d	e	f	g
1	1	1	1	1	1	0
0	1	1	0	0	0	0
1	1	0	1	1	0	1
1	1	1	1	0	0	1
0	1	1	0	0	1	1
1	0	1	1	0	1	1
1	0	1	1	1	1	1
1	1	1	0	0	0	0
1	1	1	1	1	1	1
1	1	1	1	0	1	1

Tableau de Karnaugh du segment e:

a_3a_2 \ a_1a_0	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

$e = a_1\,\overline{a}_0 + \overline{a}_0\,\overline{a}_2$

$\overline{e} = a_0 + \overline{a}_1\,a_2$

$\overline{\overline{e}} = e$

Tableau de Karnaugh du segment a:

a_3a_2 \ a_1a_0	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$a = a_1 + a_3 + a_0\,a_2 + \overline{a}_0\,\overline{a}_2$

$\overline{a} = \overline{a}_1\,a_0\,\overline{a}_3\,\overline{a}_2 + \overline{a}_1\,\overline{a}_0\,a_2$

$\overline{\overline{a}} \neq a$

Les cases sans 0 ou 1 sont dans un état "indifférent" qui peut être mis à 1 pour simplifier