SYSTEMES LOGIQUES 1

I. REPRESENTATION DE l'INFORMATION (CODAGE).

1. « Historique ».

- ✓ 35000-20000 ans av JC Des entailles sur des os
- ✓ 6000 ans av JC Des objets utilisés pour compter des dizaines et des centaines
- ✓ Grec et Romain Utilisation des chiffres romains (pb pour faire des calculs)
- ✓ 5 ème siècle Utilisation du zéro pour positionner les chiffres (en Inde)
- ✓ 12 ème siècle Utilisation du zéro en Europe

2. Code binaire « naturel ».

Le système de numérotation communément utilisé est le système décimal.
Les systèmes automatisés (et informatiques) utilisent le système binaire naturel (0 ou 1).

Base 10 (Décimal)	Base 2 (Binaire)	Base 16 (Hexadécimal)
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9

Base 10 (Décimal)	Base 2 (Binaire)	Base 16 (Hexadécimal)
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13

Ecriture des nombres > 16

$(142)_{10} = 1.10^2 + 4.10^1 + 2.10^0 = 100 + 40 + 2$

$(9B)_{16} = 9.16^1 + 11.16^0 = 144 + 11 = (155)_{10}$

$(101100)_2 = 1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 0.2^0 = 32 + 8 + 4 = 44$

Remarque. $(101100)_2 = (00101100)_2 = (2C)_{16} = 2.16^1 + 12.16^0 = 44$

Pour passer de base 10 en base 2 (et vis versa) il est intéressant d'utiliser la base 16.

Changement de base (exemples).

$(268)_{10}$ = en base 2 et 16 = $\_.16^2 + \_.16^1 + \_.16^0$ on cherche les _

$(268)_{10} = 1.16^2 + 0.16^1 + 12.16^0 = (10C)_{16} = (000100001100)_2 = (100001100)_2$

$(613)_{10}$ = en base 2 et 16 = $\_.16^2 + \_.16^1 + \_.16^0$ on cherche les _

$(613)_{10} = 2.16^2 + 6.16^1 + 5.16^0 = (265)_{16} = (001001100101)_2 = (1001100101)_2$

Remarque : Le code binaire naturel est un code pondéré, le décodage se fait en calculant directement le nombre, par ex : $(1010)_2 = 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 8 + 2 = 10$

3. Code binaire « réfléchi » (ou code « Gray »).

Dans le code binaire naturel, entre deux positions successives, plusieurs variables peuvent changer d'état, ce qui peut provoquer des erreurs, car le changement ne se fait jamais exactement en même temps. Exemple : entre 7 et 8, (0111) et (1000).

Le code binaire réfléchi est construit afin que deux configurations successives ne diffèrent que par le changement d'état d'une seule variable.

Ce code, comme son nom l'indique, contient des axes de symétrie dans sa construction.

Les 2 premiers codes : Symétrie : On ajoute des 0 d'un coté et des 1 de l'autre pour obtenir les 4 codes :

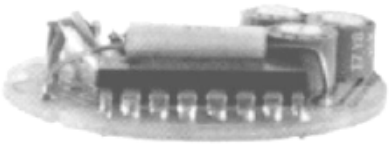
0	0	00
1	1	01
	1	11
	0	10

Symétrie : On ajoute des 0 d'un coté et des 1 de l'autre pour obtenir les 8 codes :

00	000
01	001
11	011
10	010
10	110
11	111
01	101
00	100

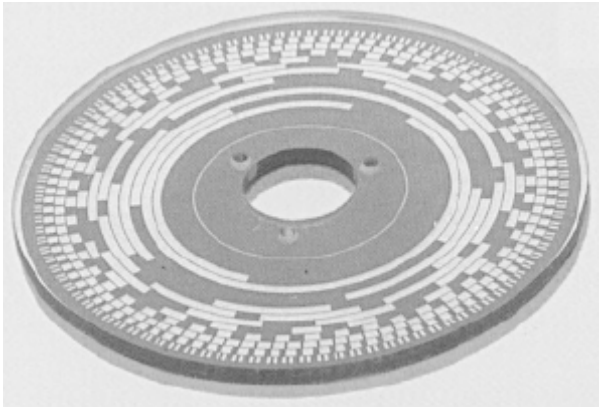
Ce code est utilisé pour les capteurs numériques de position car il permet d'éviter les confusions entre deux positions.

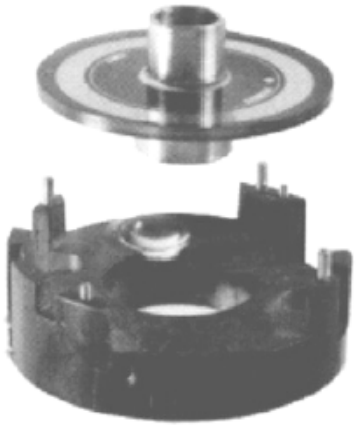
Le « codeur » numérique de position angulaire



Tous les codeurs de ce type sont construits autour d'un disque divisé en « n » pistes concentriques.

Chaque piste comporte des « cases » qui peuvent être laissées transparentes ou rendues opaques.





II. TRANSCODAGE.

Le transcodage consiste à passer d'un code à un autre.

Exemple.

Passage du code binaire réfléchi (code Gray) au code binaire naturel.

On cherche à exprimer (abc) en fonction de (xyz).

Décimal N	Binaire naturel (a b c)	Binaire réfléchi (x y z)
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

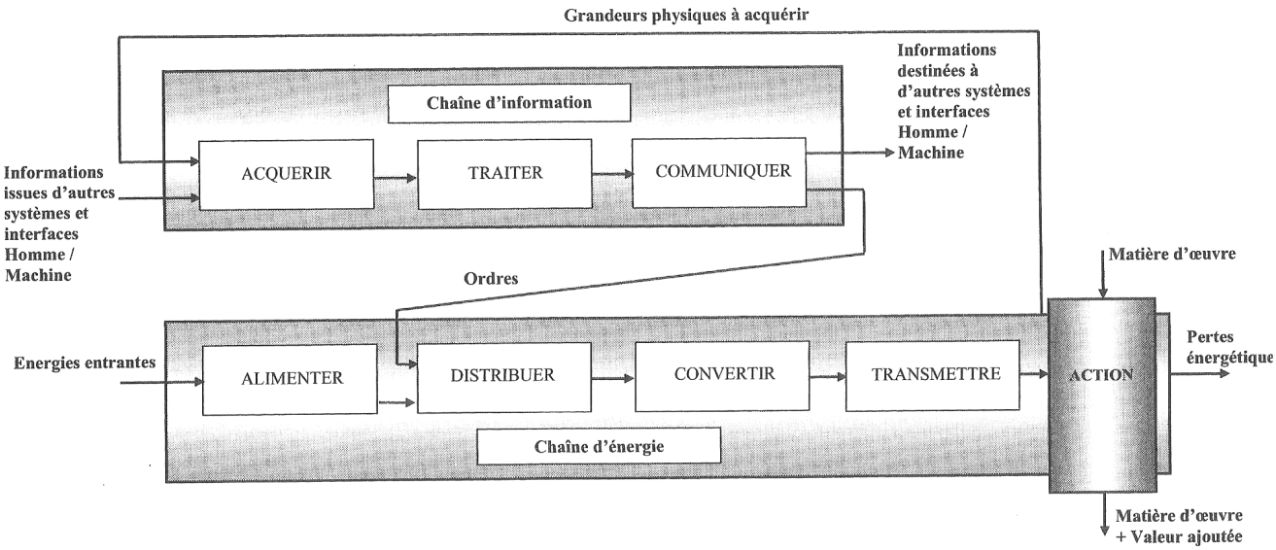
On trouve après observation attentive :

- ✓ a=1 lorsque x=1 → $a = x$
- ✓ b=1 lorsque x=1 et y=0 ou lorsque x=0 et y=1 → $b = x.\overline{y} + \overline{x}.y = x \oplus y$

  Remarque : On utilise les fonctions ET, OU, NON et OU EXCLUSIF que l'on va voir en détail dans la suite du cours.
- ✓ c=1 lorsque 1 entrée seulement ou les 3 entrées sont à 1 → $c = x \oplus y \oplus z$

III. VARIABLE LOGIQUE.

Rappel : structure d'une chaine fonctionnelle d'un système automatisé.



Les ordres et les informations peuvent être :

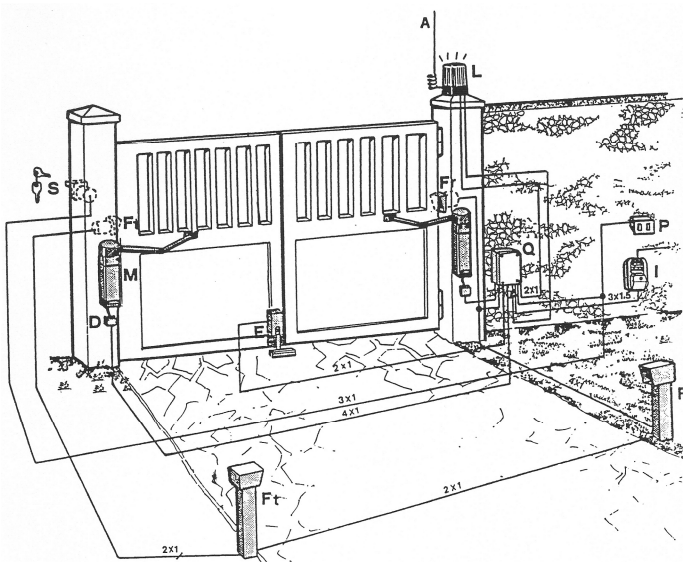
- ✓ Analogique (par exemple une tension variable)
- ✓ Logique (0 ou 1, vrai ou faux)
- ✓ Numérique

Exemple de système automatisé :

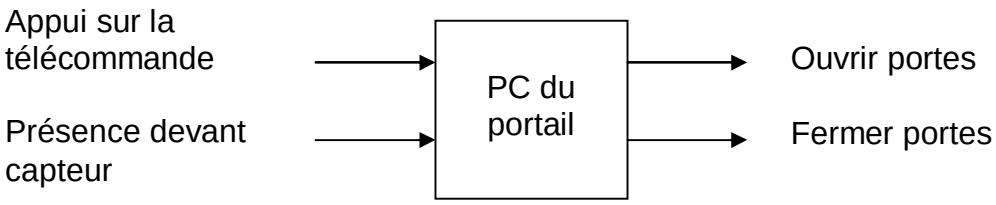
Le portail automatisé.

Pour simplifier, on s'intéresse aux éléments suivants :

- ✓ Les 2 portes
- ✓ Les 2 moteurs
- ✓ La télécommande
- ✓ La cellule photo électrique



Entrées/sorties de la Partie commande (PC) du portail : (les entrées sont les informations, les sorties sont les ordres)



Les entrées et les sorties sont sous la forme tout ou rien (1 ou 0) (vrai ou faux), on les appelle des variables logiques.

L'objet de ce chapitre est de modéliser le fonctionnement des PC

IV. ALGEBRE DE BOOLE.

Il permet de traiter des problèmes de logique.

1. Opérateurs logiques de base.

Opérateur égalité

Equation logique : S = a

Table de vérité

a	S
0	0
1	1

Opérateur ET

Equation logique : S = a.b

Table de vérité

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

Opérateur OU

Table de vérité

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

Equation logique : S = a + b

Opérateur NON

Table de vérité

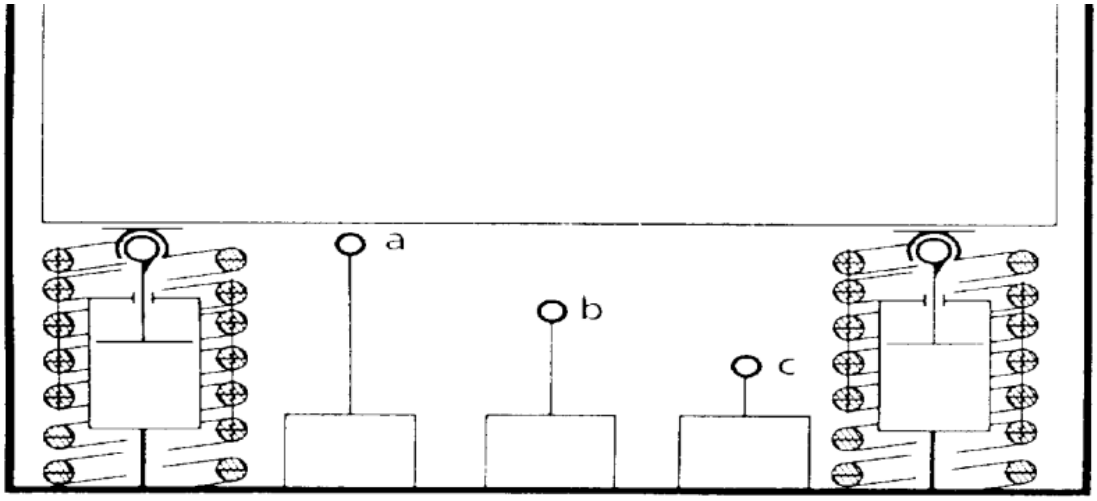
a	S
0	1
1	0

Equation logique : $S = \overline{a}$

2. Exemple de problème de logique. Etude d'un monte charge.

Un monte-charge doit permettre le levage de masses comprises entre 20 et 80 kg. Pour cela, il comporte une plate-forme reposant sur des ressorts.

Selon l'importance des charges à soulever, trois contacts réglables sont mis en circuit. Les contacts passent à 1 lorsque qu'ils sont en contact avec la cuve du monte-charge.



Cahier des charges

- ✓ À vide, aucun des contacts n'est activé. le monte-charge peut fonctionner.
- ✓ Pour des charges comprises entre 5 et 20 kg, le monte-charge ne peut fonctionner. Le contact « a » est actionné.
- ✓ Pour les charges comprises entre 20 et 80 kg, le monte-charge doit fonctionner. « a » et « b » sont actionnés.
- ✓ Pour des charges supérieures à 80 kg, le monte-charge ne peut fonctionner. Les contacts « a », « b » et « c » sont actionnés.

Problème posé : Il faut modéliser le comportement attendu en équations afin de réaliser la partie commande.

Question : *Déterminer l'équation booléenne de la sortie S assurant l'autorisation de fonctionnement de ce monte-charge*.

Le monte-charge doit fonctionner (S passe à 1) à vide (cas a=0 **et** b=0 **et** c=0) **ou** pour les charges comprises entre 20 et 80 kg (cas a=1 **et** b=1 **et** c=0).

On peut écrire l'équation de la sortie $S = \overline{a}.\overline{b}.\overline{c} + a.b.\overline{c}$

3. Propriétés de l'algèbre de BOOLE.

Commutativité	$a.b = b.a$	$a + b = b + a$
Associativité	$a.(b.c) = (a.b).c$	$a + (b + c) = (a + b) + c$
Distributivité	$a.(b + c) = a.b + a.c$	$a + (b.c) = (a + b).(a + c)$
Eléments neutres	$a.1 = a$	$a + 0 = a$
Elément absorbant	$a.0 = 0$	$a + 1 = 1$
Complément	$a.\overline{a} = 0$	$a + \overline{a} = 1$
Idempotence	$a.a = a$	$a + a = a$
Théorème de De Morgan	$\overline{a.b} = \overline{a} + \overline{b}$	$\overline{a + b} = \overline{a}.\overline{b}$

Exemples d'utilisation de l'algèbre de Boole pour simplifier des expressions logiques :

$$(a + b).(a + c) = a.a + a.c + a.b + b.c = a.(1 + c + b) + b.c = a + b.c$$

$$(c + b).c = c.c + b.c = c + b.c = c.(1 + b) = c$$

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

V. SIMPLIFICATION DE FONCTION.

Une expression combinatoire représente une fonction booléenne.

Pour n variables d'entrées, il existe une infinité d'expressions combinatoires (certaines sont équivalentes).

On recherche la forme la plus simple possible d'une expression combinatoire.
Le but est de réaliser une fonction en utilisant le moins d'opérateurs logiques possibles.

Méthode algébrique : On écrit les produits par ordre alphabétique afin de les comparer plus facilement et on utilise les propriétés de l'algèbre de Boole.

Exemples : $S1 = a + b + \overline{a}.b = a + b.(1 + \overline{a}) = a + b$

$$S2 = \overline{a}.(\overline{b}.\overline{c} + \overline{b}.c + b.c) = \overline{a}.(\overline{b}.(\overline{c} + c) + b.c) = \overline{a}.(\overline{b} + b.c)$$

Remarque : Certaines simplifications n'apparaissent pas, par exemple

$S3 = a + \overline{a}.b = a + \overline{a}.b + a.b$ (on rajoute $a.b$) $a + a.b = a.(1+b) = a$

$S3 = a + (\overline{a} + a).b = a + b$

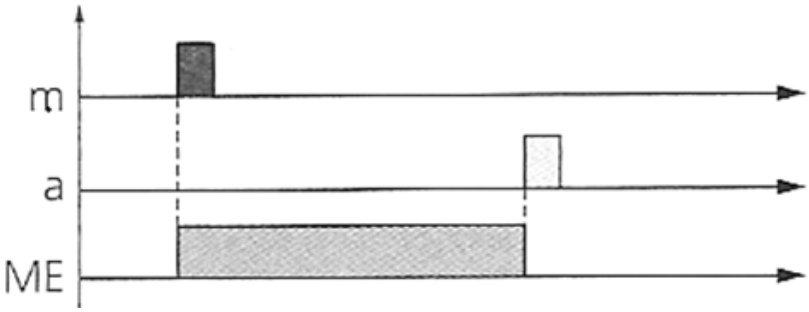
On peut donc reprendre $S2 = \overline{a}.(\overline{b} + c)$

VI. REPRESENTATION D'UNE FONCTION LOGIQUE

On peut représenter une fonction logique par :

- ✓ Une phrase (S est vrai si a et b sont vrais)
- ✓ Une équation logique (S = a.b)
- ✓ Une table de vérité.
- ✓ Un tableau de Karnaugh.
- ✓ Un chronogramme.
- ✓ Un schéma électrique.
- ✓ Un logigramme.

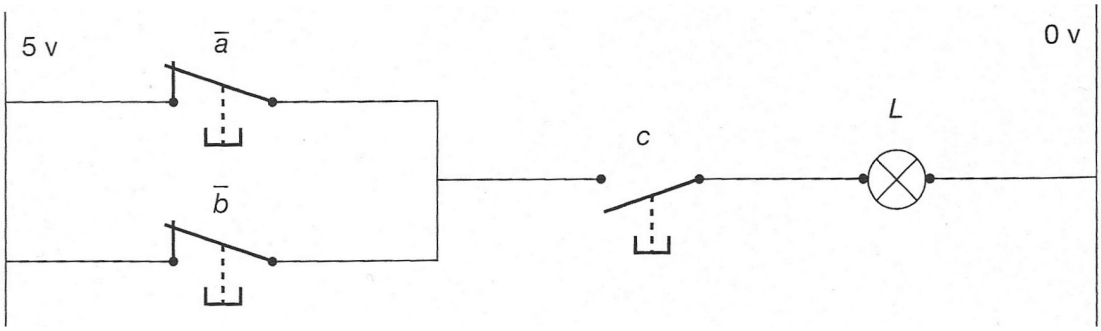
1. Chronogramme.



2. Schémas électriques.

Ils sont réalisés par des contacts électriques commandés manuellement (poussoir) ou électriquement (relais).

Exemple 1.



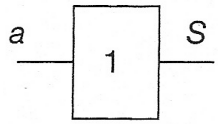
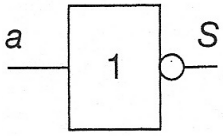
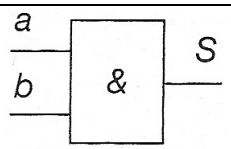
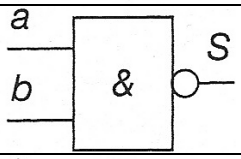
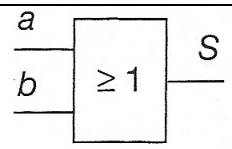
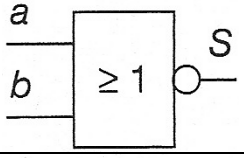
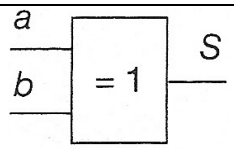
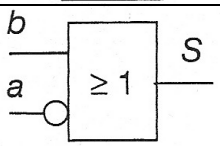
On distingue deux sortes de contacts : « à ouverture » (Non) » et « à fermeture » (Oui).

La fonction représentée est : $L = c.(\overline{a} + \overline{b})$

- ✓ La fonction « ET» est réalisée par des contacts en série.
- ✓ La fonction « OU » est réalisée par des contacts en parallèles.

3. Logigrammes.

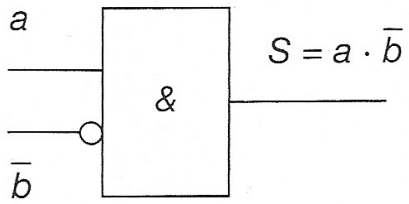
Un logigramme est une association d'opérateurs logiques décrivant une équation logique.
Liste (non exhaustive) des opérateurs logiques :

Symbole	Equation	Opérateur	Symbole	Equation	Opérateur
	S = a	Identité		$S = \overline{a}$	Non
	S = a.b	ET		$S = \overline{a.b}$	NAND
	S = a + b	OU		$S = \overline{a+b}$	NOR
	$S = a \oplus b$	OU exclusif		$S = \overline{a} + b$	a implique b

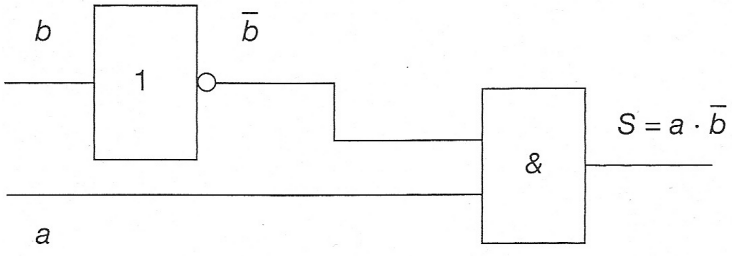
Le logigramme d'une fonction logique n'est pas unique. Il dépend des contraintes technologiques imposées.

Exemple : On veut réaliser la fonction $S = a.\overline{b}$

Solution 1 : En utilisant toutes les fonctions logiques disponibles



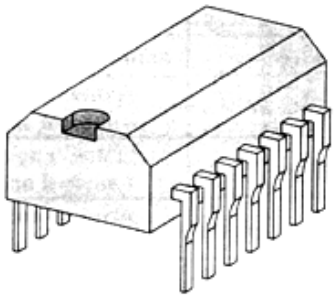
Solution 2 : En utilisant uniquement les fonctions de base ET, OU, NON

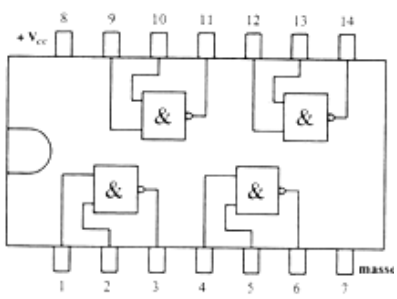


Dans certains cas, on se voit imposer l'utilisation unique des cellules NOR ou NAND

En effet, toute fonction peut être réalisée en utilisant uniquement des cellules NOR ou NAND (cellules dites universelles).

Cela permet de réaliser une fonction logique en utilisant qu'un seul type de cellules.





Il faut alors réorganiser la fonction (en utilisant le théorème de De Morgan) pour faire apparaître que des NOR ou que des NAND.