1) Logique combinatoire

1. Algèbre de Boole

1.1. Vrai ou faux ?

Une variable logique (booléen) est une variable qui ne peut prendre que deux valeurs ; Vrai (1) ou Faux (0). C'est la base de tous les signaux numériques. Toutes les informations numériques (photos, images, sons...) peuvent être représentées par une suite de 0 et de 1.

1.2. Représentations des fonctions logiques

Une fonction logique (s = f(a,b,...)), qui associe la valeur d'une variable à celle d'autres variables, peut être représentée de plusieurs manières :

- Une représentation littérale.
- Des représentations utilisant un graphique.
- Des représentations utilisant un tableau.

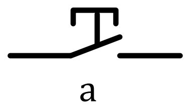
2. Représentations graphiques

2.1. Les contacts et les bobines

Les premières fonctions logiques ont été réalisées à l'aide de contacts et de bobines électriques. Chaque bobine pouvant contrôler plusieurs contacts. C'est tout naturellement que l'on peut utiliser les schémas électriques pour représenter des équations logiques.

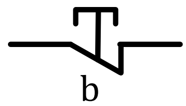
2.2. Le contact NO

Au repos (a=0), le contact NO est ouvert et ne laisse pas passer le courant. Quand on appuie dessus, au travail (a=1), le contact NO se ferme et laisse passer le courant. Le contact NO est un contact à fermeture.



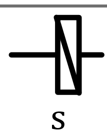
2.3. Le contact NF

Au repos (b=0), le contact NF est fermé et laisse passer le courant. Quand on appuie dessus, au travail (b=1), le contact NF s'ouvre et ne laisse plus passer le courant. Le contact NF est un contact à ouverture.



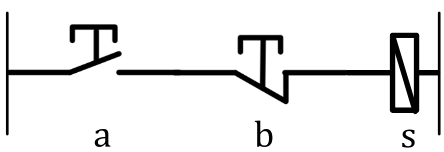
2.4. La bobine

Quand la bobine n'est pas alimentée, elle est au repos (s=0).
Quand la bobine est alimentée, elle est au travail (s=1).



2.5. Schéma électrique

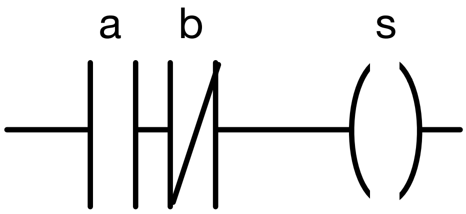
En utilisant le formalisme des électriciens, l'équation $s = a.\bar{b}$ devient :



2.6. Schémas à contacts (LADDER)

Les premières personnes amenées à programmer les automates industriels furent des électriciens. Les constructeurs ont développé un langage proche du schéma électrique pour aider l'adoption dans l'entreprise de leurs automates. C'est le langage dit LADDER ou schéma à contacts. La représentation des différents éléments a été simplifiée pour faciliter l'impression de ces schémas sur une imprimante.

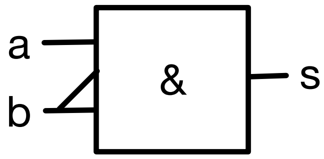
L'équation $s = a.\overline{b}$ devient :



2.7. Blocs fonctionnels

L'arrivée de l'électronique est des circuits intégrés s'est accompagnée de l'utilisation de composants réalisant les fonctions logiques de base. C'est tout naturellement qu'est apparue une représentation par blocs destinée aux jeunes électroniciens.

L'équation $s = a.\overline{b}$ devient :



3. Représentation en tableau

3.1. Table de vérité

On peut représenter une fonction logique à l'aide d'un tableau faisant correspondre à chaque combinaison des variables la valeur de la fonction correspondante. On appelle cela la table de vérité.

L'équation $s = a.\overline{b}$ devient :

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

3.2. Tableau de Karnaugh

Le tableau de Karnaugh est une représentation en deux dimensions d'une fonction logique. Il comprend 2^n cases.

L'équation $s = a.\overline{b}$ devient :

a\b	**0**	**1**
0	0	0
1	1	0

Au-delà de deux variables, les têtes de ligne et colonne se partageront les variables. Il faudra alors respecter une règle qui est de ne faire varier qu'une variable à la fois.

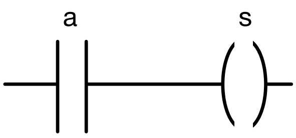
Cas de trois variables :

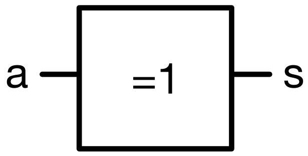
↓ Code de Gray ↓

a\bc	00	01	11	10
0				
1				

4. Les fonctions logiques de base

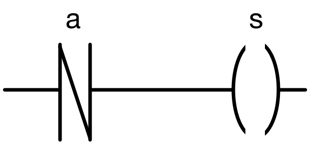
4.1. La fonction OUI

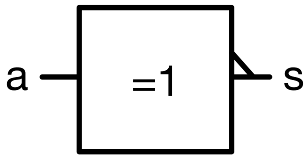




$s = a$

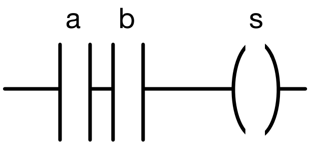
4.2. La fonction NON

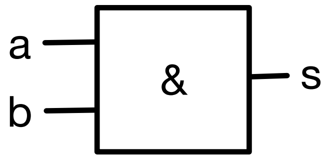




$s = \bar{a}$

4.3. La fonction ET

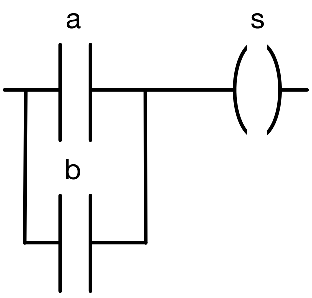


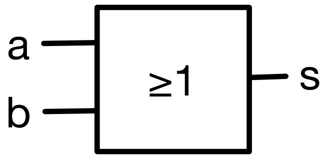


a\b	0	1
0	0	0
1	0	1

$s = a.b$

4.4. La fonction OU

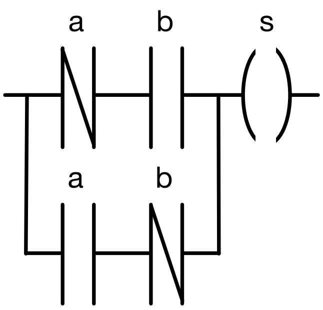


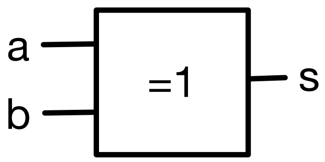


a\b	0	1
0	0	1
1	1	1

$s = a + b$

4.5. La fonction OU exclusif





a\b	0	1
0	0	1
1	1	0

$s = a \oplus b$

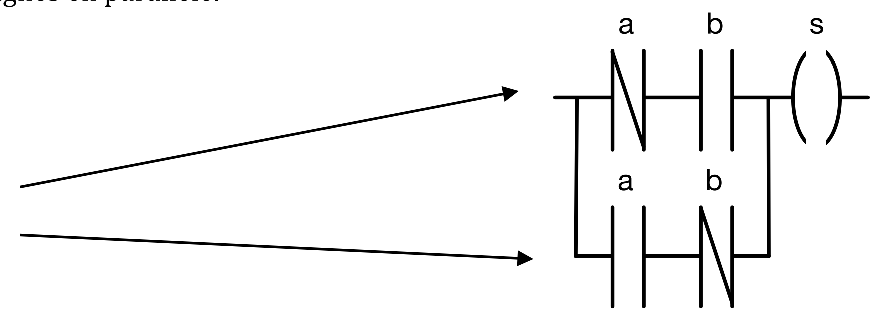
5. Conversion d'une table de vérité en schéma à contacts

Il existe une méthode très rapide pour traduire une table de vérité en schéma à contacts.

- Dans la table de vérité, ne considérer uniquement les lignes où la fonction est à 1.
- Dans chacune de ces lignes, il faut mettre en série tous les contacts, mettre des contacts NO si le contact est à 1
- Il suffit de mettre toutes les lignes en parallèle.

Exemple avec le OU-exclusif :

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



6. Simplifier une équation logique

6.1. Pourquoi

L'objectif du cours d'automatismes est la programmation d'API. Pour avoir un programme le plus maintenable possible et le plus évolutif. Il devra être facilement lisible. Pour cela, il faut limiter l'utilisation de contacts dans un programme, ce sera notre premier indice de qualité. Moins on utilise de contacts, meilleure sera notre programmation.

6.2. Réduction

- $a + \bar{a} = 1$
- $a.\bar{a} = 0$
- $a + a = a$
- $a.a = a$
- $a + 1 = 1$
- $a + 0 = a$
- $a.1 = a$
- $a.0 = 0$

6.3. Commutativité

- $a + b = b + a$
- a.b = b.a

6.4. Distributivité

- $(b + a).c \; = \; bc + ac$
- $b + ac = (b + a)(b + c)$

6.5. Egalités de De Morgan

- $\overline{a.b} = \bar{a} + \bar{b}$
- $\bar{a}.\bar{b} = \overline{a + b}$

6.6. Réflexion exclusive

$$a \oplus b = c \Leftrightarrow b \oplus c = a \Leftrightarrow c \oplus a = b$$

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

7. Simplification à l'aide du tableau de Karnaugh

7.1. Principe

Cette méthode repose sur l'utilisation des tableaux de Karnaugh. Elle consiste à mettre en évidence les simplifications vues précédemment.

7.2. Cases adjacentes

L'adresse d'une case du tableau de Karnaugh est composée de l'ensemble des bits ordonnés qui permettent sa situation. La première case en haut à gauche a l'adresse **0000**, la dernière case a l'adresse **1010**.

Dans un tableau de Karnaugh deux cases adjacentes correspondent à des adresses qui diffèrent d'un seul bit. Ceci est valable à l'intérieur du tableau, mais aussi sur ses bords : en passant du bord droit au bord gauche ou du haut au bas il y a adjacence. Ceci revient à dire que l'on peut considérer le tableau comme une sphère.

7.3. Règles

On ne regroupe que 2^n cases (2, 4, 8, 16…).

- Les regroupements sont forcément des rectangles ou carrés (compte tenu des permutations pour les bords). Pas de « L », pas de croix, …
- L'expression sera d'autant plus compacte que l'étendue des regroupements est grande. Pour un regroupement occupant la moitié du tableau il n'y a plus qu'une variable, pour le quart il reste deux variables, pour un regroupement de deux cases, il reste n-1 variables. D'une manière générale, un regroupement de 2i cases conduit à supprimer i variables.
- Il est inutile de regrouper des « 1 » qui ont tous déjà été regroupés par ailleurs. On parle alors de termes inclus.

<u>Exemple 1 :</u>

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

Dans le tableau ci-contre il y a trois 1, que l'on peut regrouper en deux groupes de deux 1.

Entre les cases **0001** et **0011** seul c varie, donc l'adresse de ces deux cases est **00x1**, soit $\bar{a}.\bar{b}.d$.

Entre les cases **0001** et **0101** seul b varie, donc l'adresse de ces deux cases est **0x01**, soit $\bar{a}.\bar{c}.d$.

La fonction logique de ce tableau est donc : $\bar{a}.\bar{b}.d + \bar{a}.\bar{c}.d$

<u>Exemple 2 :</u>

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

Dans le tableau ci-contre il y a six 1, que l'on peut regrouper en deux groupes, un groupe de deux 1 et un groupe de quatre 1.

Entre les cases **0100** et **0101** seul d varie, donc l'adresse de ces deux cases est **010x**, soit $\bar{a}.b.\bar{c}$.

Entre les cases **0010**, **0110**, **1110**, **1010**, a et b varient. L'adresse de ces quatre case est **xx10**, soit $c.\bar{d}$.

La fonction logique de ce tableau est donc : $\bar{a}.b.\bar{c} + c.\bar{d}$.

Exemple 3 :

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

Dans le tableau ci-contre il y a quatre 1, que l'on peut regrouper en un groupe de quatre 1.

Entre les cases **0000**, **0010**, **1000**, **1010**, seuls b et d ne varient pas.

L'adresse de ces quatre cases est **x0x0**, soit $\bar{b}.\bar{d}$.

La fonction logique de ce tableau est donc : $\bar{b}.\bar{d}$.

7.4. La fonction ⊕ cachée dans le tableau de Karnaugh

- Sélectionner un carré de quatre cases avec deux 1 en diagonale et deux 0.
- Déterminer le facteur sera devant la fonction ⊕ en notant les variables qui ne change pas.
- Le ⊕ portera sur les deux variables restantes.
- Pour déterminer si on a ⊕ ou $\overline{\oplus}$, il suffit de regarder si pour les cases à 1 les variables du ⊕ sont dans le même état ou non. Si les variables sont dans le même état, on est en présence d'un $\overline{\oplus}$, sinon on est en présence d'un ⊕.

<u>Exemple 1 :</u>

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

- Dans la zone sélectionnée, a et c ne varient pas et sont à 0. Le facteur sera : $\bar{a}.\bar{c}$
- Dans la première case avec un 1, b et d ont la même valeur 0. On est en présence d'un $\overline{\oplus}$.
- La solution est donc : $\bar{a}.\bar{c}.\overline{(b \oplus d)}$.

<u>Exemple 2 :</u>

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

- Le facteur est : $\bar{a}.d$.
- C'est un $\overline{\oplus}$.
- La solution = $\bar{a}.d.\overline{(b \oplus c)}$.

<u>Exemple 3 :</u>

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

- Le facteur est : $b.\bar{c}$.
- C'est un $\overline{\oplus}$.
- La solution =$b.\bar{c}.\overline{(a \oplus d)}$.

<u>Exemple 4 :</u>

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

- Le facteur est : $b.d$.
- C'est un ⊕.
- La solution =$b.d.(a \oplus c)$.