Université de Metz

U.F.R. SciF.A.

Cours de Logique et d'APIs

D.E.U.G. STPI

Année Universitaire 2002/2003

Y. Morère

Cette page est laissée blanche intentionnellement

Table des matières

Table des figures

Liste des tableaux

Cette page est laissée blanche intentionnellement

Chapitre 1

Représentation des nombres

1.1 Système de numération

1.1.1 Généralités

Le système de représentation des nombres actuels est une numération de position : un petit nombre de signe (ici 10 chiffres) est utilisé, mais ceux-ci acquièrent une signification particulière selon leur position dans l'écriture du nombre.

Ce système est fondé sur les propriétés de la division euclidienne : pour un nombre b, tout nombre N peut se décomposer de manière unique, sous la forme du polynôme suivant :

$$N = a_n.b^n + a_{n-1}.b^{n-1} + ... + a_{n-i}.b^{n-i} + ... + a_1.b^1 + a_0.b^0$$

avec $0 \leq a_i \leq b-1$, le nombre s'écrivant alors : $a_n a_{n-1}...a_i...a_1 a_0$.

On appelle le nombre b la base de numération. Un tel système nécessite b signe différent que l'on appellera chiffres.

Ainsi pour $b = 10$, 1264 représente le nombre $1.10^3 + 2.10^2 + 6.10^1 + 4.10^0$. Pour $b \leq 10$ on utilise les b premiers chiffres : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ Pour $b = 16$ on untilise la notation héxadécimale :0123456789$\mathbb{ABCDEF}$

1.1.2 Les valeurs de b

La base 10 est actuellement utilisée probablement parce que les hommes de toutes races ont 10 doigts et qu'ils ont commencé par compter sur leurs doigts.

À la suite des développement des calculateurs, on utilise beaucoup la base 2 puisque tout organe physique bistable peut être considéré comme un chiffre dans cette base. Ceci induit l'utilisation des bases 8 et 16 qui sont des puissances entières de 2 pour lesquelles il est assez aisé de passer de l'une à l'autre (bases multiples l'une de l'autre).

1.1.3 Le changement de base

Le problèmes est le suivant : on dispose d'un nombre N écrit dans une base b_1 noté $N_{(b_1)}$ et l'on souhaite trouver l'écriture de N dans une base b_2, soit $N_{(b_2)}$. Pour ce faire on dispose de moyens de calculs qui travaillent sur des nombres représentés dans une base b'. Trois cas peuvent donc se présenter à nous : $b' = b_1$, $b' = b_2$ b' différent de b_1 et de b_2.

1.1.3.1 Premier cas : $b' = b_1$

Il est possible d'utiliser la méthodes des divisions successives. On divise $N_{(b_1)}$ par b_2 exprimé dans b_1.
Par exemple : 1994_{10} à convertir en base 16, calculs dans la base 10.

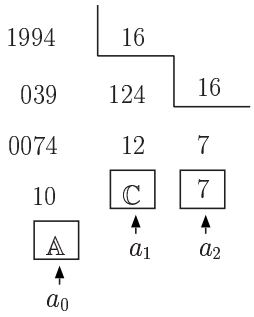


FIG. 1.1 – Conversion base 10 vers base 16

Par exemple : 1990 (10) donne 11 111 000 110 (2)

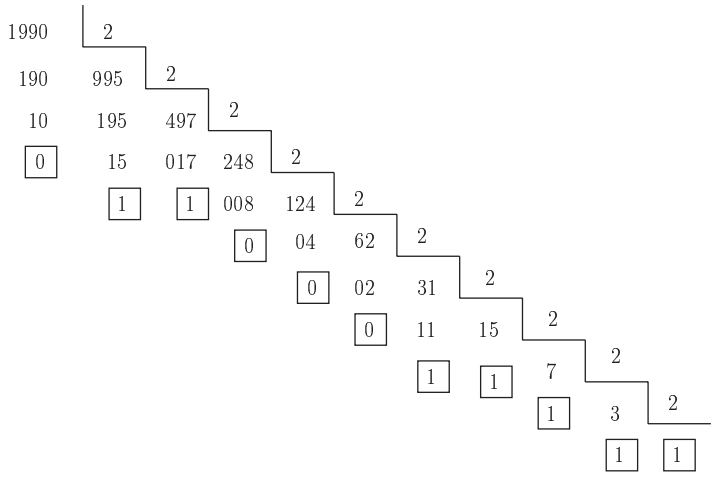


FIG. 1.2 – Conversion base 10 vers base 2

1.1.3.2 Deuxième cas : $b' = b_2$

On effectue le developpement en puissance de b_2, exprimé dans b_1.
Vérification de l'exemple précédent : $7\mathbb{C}\mathbb{A}_{16}$: 7.16^2+12(C dans b_2).16^1+10(A dans b_2).16^0 = $7.256 + 12.16 + 10 = 1994$
On peut également procéder par multiplications successives en utilisant une factorisation particulière du polynôme. On minimise alors les opérations à effectuer. $N_{(b_2)} = a_O + b_1(a_1 + b_1(a_2 + b_1(a_3... + b_1(a_{p-1} + b_1.a_p))...))$ $1994 = 10 + 16.(12 + 16.7)$

1.1.3.3 Troisième cas : b' différent de b_1 et b_2

Dans ce cas il faut procéder en deux fois en transitant par la base b' (deuxième cas puis premier cas).

1.1.4 Bases multiples l'une de l'autre

C'est le cas de la base 8 et 16 vis à vis de la base 2 ($8 = 2^3, 16 = 2^4$). Le passage d'une base à l'autre se fait simplement par regroupement de k éléments ou décomposition d'un élément en k éléments.

Exemple : $1\mathbb{F}6_{16} = ???_2$

1	$\mathbb{F}$	6
0001	1111	0110

$1637_8 = ???_2$

1	6	3	7
001	110	011	111

$1111000110_2 = ???_8$

001	110	011	111
1	7	0	6

$1111000110_2 = ???_{16}$

11	1100	0110
3	$\mathbb{C}$	6

En conséquence, il peut être plus efficace pour passer de la base 10 à la base 2 de passer d'abord en base 16 (conversion plus rapide) puis de convertir ensuite le nombre obtenu en base 2.

1.2 Représentation des nombres pour le traitement logique ou numérique

1.2.1 Représentations binaires

1.2.1.1 Traitement logique

Chacun des bits est disponible en parallèle. Le traitement s'effectue bit à bit au moyen de circuits standards.

1.2.1.2 Traitement numérique

La représentation binaire des nombres est alors une représentation formatée. Limitation au nombre au format fixe.

Le formatage est défini par une suite de mots de N bits, ou N est caractéristique de l'organisation de la mémoire de stockage ou du processeur de traitement des mots. La signification de chaque mot est fonction du type de variable mémorisée. Par exemple deux mots succéssifs peuvent représenter :

- ▷ un nombre entier en double précision,
- ▷ un nombre rationnel, le premier mot représentant la partie entière, la seconde partie décimale.

En général la représentation des nombres pour le traitement numérique doit être signée. Le signe est identifiée par le bit de poids le plus élevé du premier mot de la représentation.

Exemple application au cas des nombres relatifs, format défini par un mot unique de 16 bits en simple précision nombre positif.

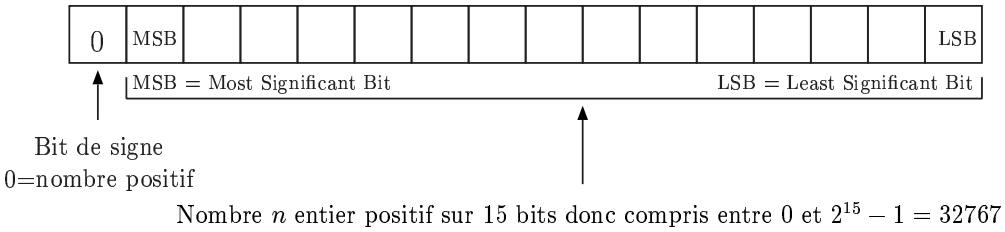


FIG. 1.3 – Nombre signé

Le nombre négatif en représentation complémentée à 2.

Le principe de la représentation complémentée à 2 est le suivant : on associe au nompbre négatif $-n$ (avec $n > 0$) le nombre binaire $bc2$ (appelé représentation binaire complémentée à 2) défini par l'équation logique suivante :

$$bc2 = \bar{n}_{(2)} + 1$$

avec n exprimé dans le format choisi.

Exemple : Soit à représenter le nombre $-n = -1988$

$$n_{(2)} = 0000\,0111\,1100\,1100$$

$$\bar{n}_{(2)} = 1111\,1000\,0011\,0011$$

$$bc2 = 1111\,1000\,0011\,0100$$

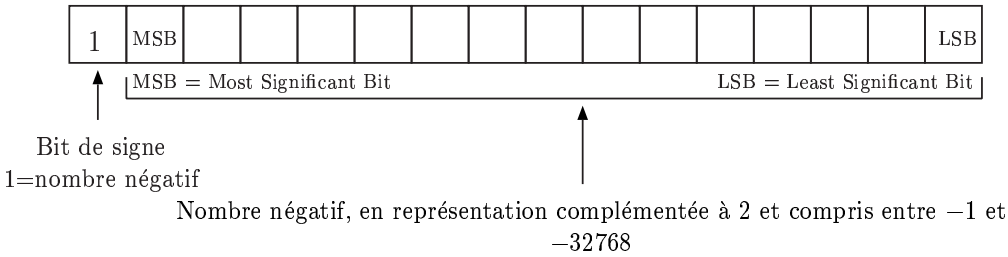


FIG. 1.4 – Nombre signé négatif

Pour -1 :

$$n_{(2)} = 0000\,0000\,0000\,0001$$

$$\bar{n}_{(2)} = 1111\,1111\,1111\,1110$$

$$bc2 = 1111\,1111\,1111\,1111$$

Pour -32768 :

$$n_{(2)} = 1000\,0000\,0000\,0000$$

$$\bar{n}_{(2)} = 0111\,1111\,1111\,1111$$

$$bc2 = 1000\,0000\,0000\,0000$$

L'intérêt de cette représentation complémentée est manifeste pour l'addition des nombres entiers en représentation binaire.

$-73 + 54 = -19$		10110111
	+	00110110
	=	11101101

Il s'agit bien d'un nombre négatif. On vérifie par l'opération réciproque, $n = \overline{(bc2 - 1)}$, que $n = \overline{11101100} = 00010011$ qui correspond bien au nombre 19.

$73 - 54 = 17$		01001001
	+	11001010
	$\boxed{1}$	00010011

$-73 - 54 = -127$		10110111
	+	11001010
	$\boxed{1}$	10000001
devient		11111111

($\overline{bc2 - 1} = 01111111$)

Les débordements de capacité n'ont donc pas du tout la même signification selon le signe des opérandes : le débordement n'est pas significatif dans le cas $73-54$, mais est significatif dans le cas de l'addition, des deux nombres négatifs.

1.2.2 Représentation binaire décimale (DCB ou BCD)

À chacun des chifres d'un nombre entier décimal on associe son expression binaire sur 4 bits. La représentation DCB d'un entier à m chiffres comporte donc $4.m$ bits. Par exemple 1989 s'exprime en DCB par 0001/1001/1000/1001

Cette représentation des nombres est particulièrement exploitée pour
▷ lire une entrée sur une roue codeuse,
▷ afficher un nombre sur un ensemble d'unités d'afficheurs 7 segments.

1.2.3 Code Gray ou code binaire réfléchi

La propriété fondamentale du code gray est de ne subir qu'une seul variation de digit (le digit est la valeur binaire, 0 ou 1, associé à un bit) d'un nombre entier au nombre entier suivant (d'un nombre n à $n + 1$ ou à $n - 1$).

Le tableau ci dessous montre la correspondance entre le code gray et le code binaire pour les 22 premiers nombres :

Nombre	Code binaire pur					Code gray				
	B4	B3	B2	B1	B0	G4	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	1
3	0	0	0	1	1	0	0	0	1	0
4	0	0	1	0	0	0	0	1	1	0
5	0	0	1	0	1	0	0	1	1	1
6	0	0	1	1	0	0	0	1	0	1
7	0	0	1	1	1	0	0	1	0	0
8	0	1	0	0	0	0	1	1	0	0
9	0	1	0	0	1	0	1	1	0	1
10	0	1	0	1	0	0	1	1	1	1
11	0	1	0	1	1	0	1	1	1	0
12	0	1	1	0	0	0	1	0	1	0
13	0	1	1	0	1	0	1	0	1	1
14	0	1	1	1	0	0	1	0	0	1
15	0	1	1	1	1	0	1	0	0	0
16	1	0	0	0	0	1	1	0	0	0
17	1	0	0	0	1	1	1	0	0	1
18	1	0	0	1	0	1	1	0	1	1
19	1	0	0	1	1	1	1	0	1	0
20	1	0	1	0	0	1	1	1	1	0
21	1	0	1	0	1	1	1	1	1	1
22	1	0	1	1	0	1	1	1	0	1
...			...					...		

La construction des colonnes de bits du code Gray est systématique et correspond à une alternance de "paquets" de bits de même digit.

$G0$: 1 "0" puis une alternance de paquets de deux (2^1) "1" puis deux "0", ...

$G1$: 2 (2^1) "0" puis une alternance de paquets de quatre (2^2) "1" et "0", ...

$G2$: 2^2 "0", puis paquets alternés de huit (2^3) "1" et "0", ...

$G3$: 2^3 "0", 16=2^4 "1", 2^4 "0", ...

...

Gn : 2^n "0", $2^{n+1}1$ "1", $2^{n+1}1$ "0"

1.2.3.1 Relation entre les codes : Transcodage Binaire - Gray

$Gn = Bn$

$G(n-1) = Bn \oplus B(n-1)$

....

$G2 = B3 \oplus B2$

$G1 = B2 \oplus B1$

$G0 = B1 \oplus B0$

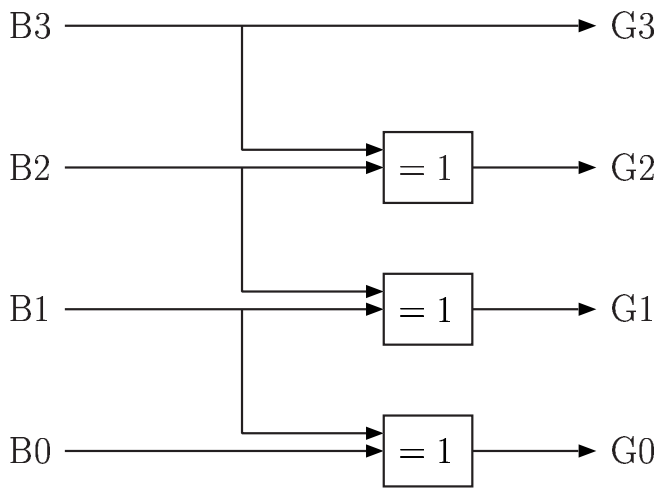


FIG. 1.5 – Circuit de transcodage Binaire - Gray

1.2.3.2 Applications industrielles

L'application type du code Gray est son utilisation pour le codage numérique binaire de codeurs multipistes. Il existe deux techonologies : opto-electronique (piste alternativement opaque et transparente), magnétique (piste alternativement magnétique ou non). Le nombre de piste du codeur caractérise sa résolution ou nombre de point par tour. 12 pistes = $2^{12} = 4096$

Pour les codeurs opto-electronique, la lecture est effectuée par un ensemble comprenant une diode LED émettrice et des photos récepteurs à transistors. Les pistes sont obtenues par photo gravure.

1.2.3.3 Application pout l'étude des systèmes logiques

L'application fondamentale du code Gray en logique est le codage utilisé pour la représentation en tableau de Karnaugh. Cette représentation est basée sur le principe d'adjacence combinatoire des cases adjacente du tableau. Ce principe implique impose un changement d'une et une seule variable d'un case à une case adjacente.

La notation de Girard n'est qu'une matérialisation graphique d'un codeur Gray multipiste linéaire.

1.2.4 Codes pondérés

Ces codes binaires pondérés sont basés sur une pondération des bits différente du code binaire naturel (pondération $2^n ... 8 - 4 - 2 - 1$).

Le code $5 - 2 - 2 - 1$ par exemple permet de coder les 11 premiers nombres sur 4 bits (pondération des 4 digits : 5 le poids le plus élevé puis respectivement 2, 2 et 1).

0=0+0+0+0	0000
1=0+0+0+1	0001
2=0+0+2+0	0010
3=0+0+2+1	0011
4=0+2+2+0	0110
5=5+0+0+0	1000
6=5+0+0+1	1001
7=5+0+2+0	1010
8=5+0+2+1	1011
9=5+2+2+0	1110
10=5+2+2+1	1111

Ce type de code n'est plus utilisé.

1.2.5 Codes décimaux-binaire symétriques

Des codes symétriques ont été utilisés pour la représentation des nombres décimaux. On peut citer par exemple les codes Aiken et Excédent3. Ces codes sont devenus désuets du fait du developpement des méthodes modernes de cryptage. L'intérêt de ces codes résidait dans leur plus grande facilité de traitement des nombres négatifs par rapport au binaire naturel.

	Représentation	Binaire Nat.	Aiken	Excédent3	Gray
	0000	0	0		0
	0001	1	1		1
	0010	2	2		3
	0011	3	3	0	2
	0100	4	4	1	7
	0101	5		2	6
	0110	6		3	4
axe de	0111	7		4	5
symétrie	1000	8		5	15
	1001	9		6	14
	1010	10		7	12
	1011	11	5	8	13
	1100	12	6	9	8
	1101	13	7		9
	1110	14	8		11
	1111	15	9		10

1.3 Codes détecteurs et correcteurs d'erreurs

Le principe de la correction d'erreur (erreur de codage ou erreur de transmission de l'information) consiste à rajouter une information surabondante à l'information minimale pour représenter le message à transmettre. Cette information se traduit par des bits supplémentaires par rapport au nombre qu'il faut pour coder l'ensemble des informations.

1.3.1 Correction par bit de parité ou *checksum*

Un bit supplémentaire, bit toujours de poids le plus élévé, permet une détection d'erreur unique dans la transcription du code (un et un seul changement d'état d'un bit, bit de parité inclus).

Le bit de parité est souvent impair, c'est à dire qu'il prend la valeur :
▷ 1 lorsque le nombre de bits à 1 du codage normal est pair ou nul,
▷ 0 lorsque le nombre de bits à 1 du codage normal est impair.
L'intérêt de la parité impaire est d'éliminer des combinaisons normales, la combinaison ne comprenant que des zéros, ce qui permet de distinguer la présence de l'absence d'un code.

1.3.2 Détection et correction d'erreur

Le principe consiste à accroître la distance entre deux combinaisons normales quelconques du code. Par définition la distance d'un code est le nombre minimal de changement d'état de bits nécessaires pour passer d'une combinaison à une autre.

Le principe général de génération d'un code avec possibilité de détection d'erreur s'appuie sur le concept de distance minimal D_m selon la formule suivante :

$$D_m = C + D + 1$$

avec D indiquant le niveau de détection d'erreur (1 pour erreur simple, 2 pour erreur double...), avec C indiquant le niveau dencorrection d'erreur (1 pour corriger une erreur simple, 2....).

Par exemple pour pouvoir détecter et corriger une erreur simple le distance du code doit être au moins égale à 3.

Cette page est laissée blanche intentionnellement

Chapitre 2

Systèmes binaires et algèbre de Boole

Actuellement, alors que les ordinateurs analogiques sont encore du domaine de la recherche, les informations traitées par les systèmes informatiques sont codées sous forme binaire. Un *système binaire* (signal, circuit, etc...) est un système qui ne peut exister que dans deux états autorisés. Diverses notations peuvent être utilisées pour représenter ces deux états :

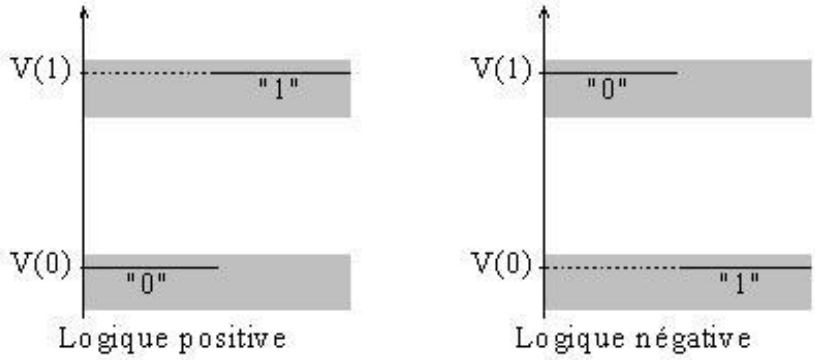
numérique :	1 et 0 (bit : *bi*nary digi*t*)
logique :	vrai et faux (true et false) oui et non (yes et no)
électronique :	ON et OFF haut et bas (HI et LO, H et L, H et B)

Pour étudier les fonctions de variables binaires on utilise une algèbre développée au XIX^{me} siècle par un mathématicien anglais : Georges Boole. Dans ce chapitre nous nous proposons de présenter les fonctions de base de l'algèbre booléenne ainsi que leurs représentations symboliques en électronique. Nous rappellerons également, sans prétendre à la rigueur mathématique, les quelques notions élémentaires nécessaires à l'étude des circuits électroniques.

L'algèbre de Boole concerne la logique des systèmes binaires. Une variable booléenne ne peut prendre que deux valeurs possibles 0 ou 1. En électronique les deux états d'une telle variable peuvent être associés à deux niveaux de tension : V(0) et V(1) pour les états 0 et 1 respectivement. On distingue les logiques positive et négative selon que V(1) > V(0) ou V(1) < V(0). Ce que nous pouvons résumer dans la table suivante donnant la signification logique des niveaux physiques :

Niveau	Logique positive	Logique négative
H	1	0
L	0	1

En pratique un niveau est défini par un domaine en tension ou en courant. Par exemple en technologie TTL, un niveau sera dit haut s'il est compris entre +2 V et +5 V et un niveau sera bas s'il est inférieur à +0.8 V.



2.1 Porte OU (inclusif)

L'opération OU (OR), encore appelée addition logique, a au moins deux entrées. La sortie d'une fonction OU est dans l'état 1 si au moins une de ses entrées est dans l'état 1. La fonction OU, notée +, est définie par la table de vérité suivante :

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

TAB. 2.4 – OU : table de vérité

Une porte OU à deux entrées est symbolisée de la manière suivante :

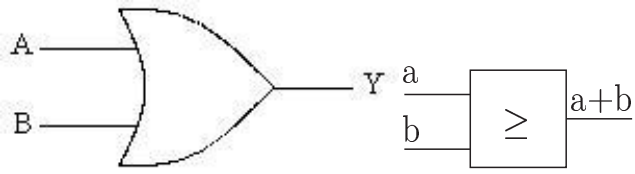


FIG. 2.1 – OU : symbole

Il est facile de vérifier les propriétés suivantes de la fonction OU :

(A + B) + C = A + (B + C) = A + B + C	Associativité
A + B = B + A	Commutativité
A + A = A	Idempotence
A + 0 = A	Elément neutre
A + 1 = 1	

2.2 Porte ET

L'opération ET (AND), encore dénommée produit logique ou intersection, a au moins deux entrées. La sortie d'une fonction AND est dans l'état 1 si et seulement si toutes ses entrées sont dans l'état 1. La fonction ET, notée •, est définie par la table de vérité suivante :

A	B	Y = A • B
0	0	0
0	1	0
1	0	0
1	1	1

TAB. 2.7 – ET : Table de vérité

Une porte ET à deux entrées est symbolisées de la manière suivante :

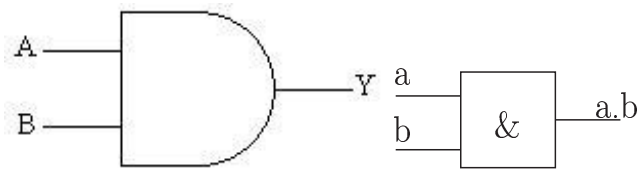


FIG. 2.2 – ET : Symbole

Il est facile de vérifier les propriétés suivantes de la fonction ET :

(A • B) • C = A • (B • C) = A • B • C	Associativité
A • B = B • A	Commutativité
A • A = A	Idempotence
A • 1 = A	Elément neutre
A • 0 = 0	

D'autre part, les opérations ET et OU sont distributives l'une par rapport à l'autre :

A • (B + C) = (A • B) + (A • C)
A + (B • C) = (A + B) • (A + C)

2.3 Inverseur : porte NON

L'opération NON (NOT) a une seule entrée et une seule sortie. La sortie d'une fonction NON prend l'état 1 si et seulement si son entrée est dans l'état 0. La négation logique est symbolisée par un petit cercle dessiné à l'endroit où une ligne en entrée ou en sortie rejoint un symbole logique. La table suivante donne la table de vérité correspondante.

A	$Y = \overline{A}$
0	1
1	0

TAB. 2.11 – NON : table de vérité

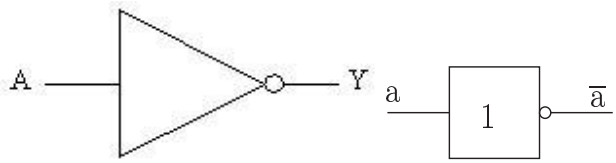


FIG. 2.3 – Inverseur

À partir des définitions des fonctions NON, OU et ET nous pouvons déduire :

$$\overline{\overline{A}} = A$$
$$\overline{A} + A = 1$$
$$\overline{A} \bullet A = 0$$
$$A + \left(\overline{A} \bullet B\right) = A + B$$

2.4 Théorèmes de De Morgan

De Morgan a exprimé deux théorèmes qui peuvent se résumer sous la forme suivante :

$$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$$

$$\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$$

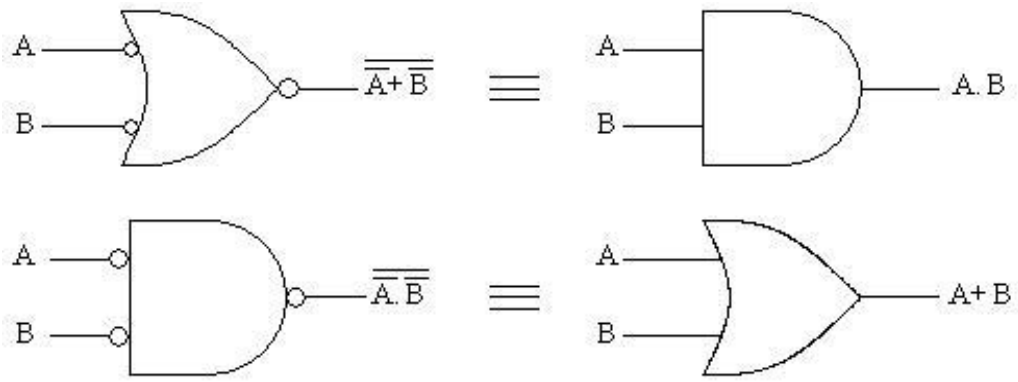
Pour vérifier le premier théorème nous remarquons que si toutes les entrées sont à 1 les deux membres de l'équation sont nuls. Par contre si une au moins des entrées est à 0 les deux membres de l'équation sont égaux à 1. Il y a donc égalité quels que soient les états des diverses entrées. Le second théorème se vérifie de la même manière : si toutes les entrées sont à 0 les deux membres de l'équation sont à 1, par contre si au moins une des entrées est à 1 les deux expressions sont à 0.

Les théorèmes de De Morgan montrent qu'une fonction ET peut être fabriquée à partir des fonctions OU et NON. De même une fonction OU peut être obtenue à partir des fonctions ET et NON. La figure montre la conversion d'une porte OU en porte ET et réciproquement, utilisant le fait que :

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

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

De même, à partir des théorèmes de De Morgan nous pouvons montrer qu'une porte ET en logique positive fonctionne comme une porte OU en logique négative et vice versa.



2.5 Portes NON ET et NON OU

Une porte NON ET (NAND : NOT AND) est constituée par un inverseur à la sortie d'une porte ET. Elle a pour table de vérité :

A	B	$\mathbf{Y = \overline{A \bullet B}}$
0	0	1
0	1	1
1	0	1
1	1	0

Tab. 2.16 – NON ET : table de vérité

et pour symbole :

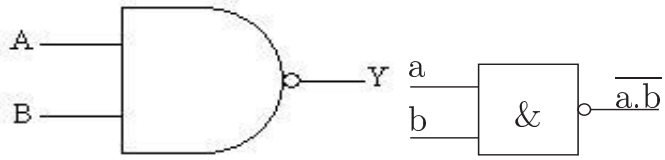


Fig. 2.4 – NON ET : symbole

Une négation à la sortie d'une porte OU constitue une fonction NON OU (NOR : NOT OR). Elle a pour table de vérité :

A	B	$\mathbf{Y = \overline{A + B}}$
0	0	1
0	1	0
1	0	0
1	1	0

Tab. 2.18 – NON OU : table de vérité

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

TAB. 2.20 – XOR : table de vérité

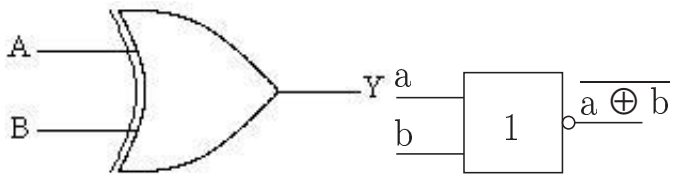


FIG. 2.6 – XOR : symbole

et pour symbole :

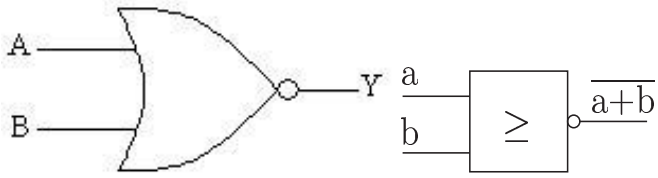
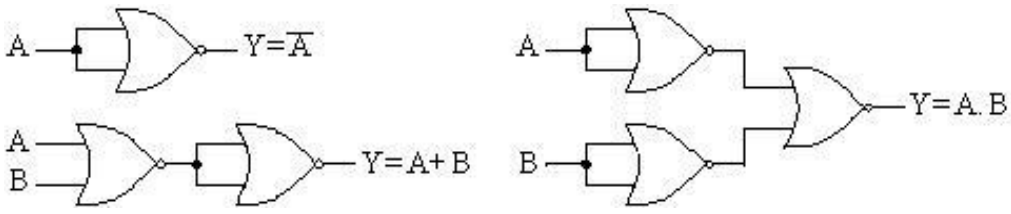


FIG. 2.5 – NON OU : symbole

Comme les transistors qui interviennent comme éléments de base des portes sont par essence des inverseurs, les portes NAND et NOR sont très usitées dans la réalisation des circuits logiques. Grâce aux lois de De Morgan il est possible de réaliser des systèmes logiques avec uniquement des portes NAND ou NOR. La figure suivante montre, par exemple, comment les portes NOT, OR et AND peuvent être obtenues à partir de portes NOR.



2.6 Porte OU exclusif

La sortie d'une fonction OU exclusif (XOR) à deux entrées est dans l'état 1 si une entrée et seulement une est dans l'état 1. Sa table de vérité est la suivante :

Elle a pour représentation symbolique :

Nous pouvons formuler de diverses manières la définition précédente : Y est égal à 1 si et seulement si A = 1 ou B = 1 mais pas simultanément. Ce que nous pouvons écrire :

$$A \oplus B = (A + B) \bullet \overline{(A \bullet B)}$$

Nous pouvons encore dire Y est égal à 1 si A = 1 et B = 0 ou si B = 1 et A = 0. Soit :

$$A \oplus B = (A \bullet \overline{B}) + (B \bullet \overline{A})$$

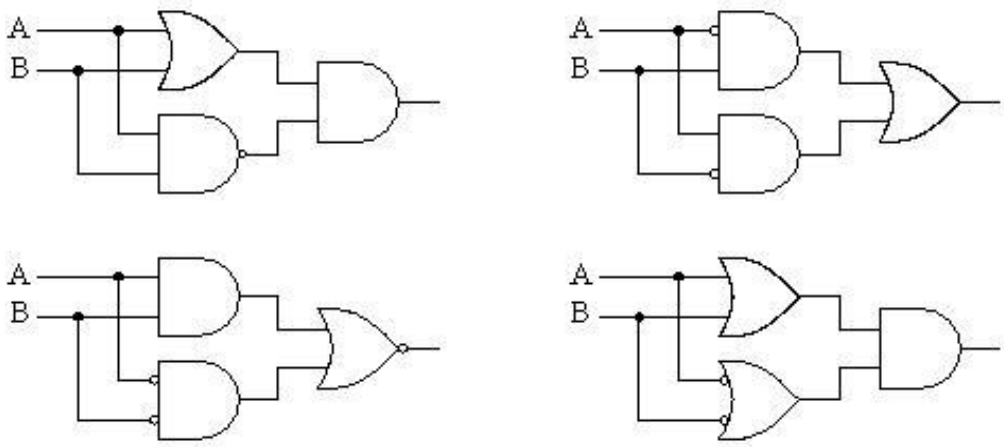
Une fonction XOR fournit un comparateur d'inégalité : Y ne vaut 1 que si A et B sont différents. Si A et B sont égaux à 1 ou si A et B sont égaux à 0 alors Y = 0. Ce qui s'écrit :

$$A \oplus B = \overline{(A \bullet B) + (\overline{A} \bullet \overline{B})}$$

Le complément du OU-exclusif correspond à un détecteur d'égalité. Nous avons encore la relation suivante qui peut être démontrée en utilisant les théorèmes de De Morgan :

$$A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$$

A ces quatre relations logiques correspondent quatre circuits réalisant la fonction XOR à partir de portes OR et AND :



2.7 Porte à Trois Etats

La porte «3 états», ou «3 tri-state», n'est pas une porte logique au sens strict. Elle est principalement utilisée pour connecter une sortie sur une ligne commune à plusieurs circuits (un bus par exemple). Elle remplace généralement une porte ET. En effet, la mise en parallèle sur une même ligne de plusieurs portes ET introduit des capacités parasites. Ceci augmente les constantes de temps et a pour effet de détériorer les fronts de montée et de descente des signaux. Cela peut perturber le fonctionnement d'un système. Une porte 3 états est schématisée de la manière suivante :

Son fonctionnement peut être résumé ainsi :

C	A	Y	sortie
1	0	0	faible impédance
1	1	1	faible impédance
0	X	0	haute impédance

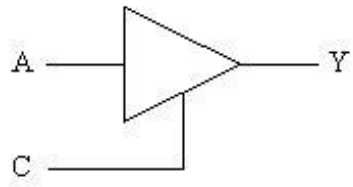


FIG. 2.7 – Porte à Trois Etats

Lorsque la commande C est à 0 l'impédance de sortie est très grande : pratiquement déconnectée. D'autre part, ces portes "3 états" fournissent une amplification de puissance.

2.8 Résumé des identités booléennes de base

Il est possible montrer que toute fonction booléenne d'un nombre quelconque de variables peut s'écrire avec les trois fonctions de base ET, OU et NON. Nous avons rassemblé dans la table suivante les relations de base de l'algèbre de Boole qui nous seront utiles par la suite.

OU	$(A + B) + C = A + (B + C) = A + B + C$
	$A + B = B + A$
	$A + A = A$
	$A + 0 = A$
	$A + 1 = 1$
ET	$(A \bullet B) \bullet C = A \bullet (B \bullet C) = A \bullet B \bullet C$
	$A \bullet B = B \bullet A$
	$A \bullet A = A$
	$A \bullet 1 = A$
	$A \bullet 0 = 0$
Distributivité	$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$
	$A + (B \bullet C) = (A + B) \bullet (A + C)$
NON	$\overline{\overline{A}} = A$
	$\overline{A} + A = 1$
	$\overline{A} \bullet A = 0$
	$A + (A \bullet B) = A$
	$A \bullet (A + B) = A$
	$(A + B) \bullet (A + \overline{B}) = A$
	$A + (\overline{A} \bullet B) = A + B$
De Morgan	$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$
	$\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$

OU exclusif	$A \oplus B = (A + B) \bullet \overline{(A \bullet B)}$ $A \oplus B = (A \bullet \overline{B}) + (B \bullet \overline{A})$ $A \oplus B = \overline{(A \bullet B) + (\overline{A} \bullet \overline{B})}$ $A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$

2.9 Ecritures canoniques d'une fonction logique

2.9.1 Somme canonique de produits

Considérons trois variables booléennes x, y et z. A partir de ces trois variables nous pouvons construire huit produits logiques (ou *minterms*) $P_{i=0,7}$ faisant intervenir x ou son complément, y ou son complément et z ou son complément. Pour chacune des huit combinaisons $C_{i=0,7}$ (000, 001, 010, etc...) des variables x, y et z, nous pouvons calculer les valeurs de ces produits. Celles-ci sont rassemblées dans la table suivante. Chacun de ces produits ne prend la valeur 1 que pour une et une seule combinaison : P_i vaut 1 uniquement pour la combinaison C_i.

				P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7		
C_i	x	y	z	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}y\bar{z}$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	$xy\bar{z}$	xyz		
0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0	1	0	0	1

te fonction logique de trois variables x, y et z, nous pouvons écrire sa table de
t-à-dire expliciter sa valeur pour chacune des huit combinaisons C_i. Considér
mple, la fonction F dont la table de vérité est donnée dans la table suivante

x	y	z	F	$P_1 + P_3 + P_4$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

nction F prend la valeur 1 pour la combinaison C_1 comme le produit P_1
son C_3 comme P_3 et la combinaison C_4 comme P_4. La fonction F prenan
pour toutes les autres combinaisons comme les produits P_1, P_3, P_4, nous pou
re que F est égale à la fonction :

$+ P_3 + P_4$

uvons vérifier cette identité dans la table 11. Nous pouvons donc exprimer l
des variables x, y et z sous la forme :

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

Cette façon, très générale, d'écrire une fonction booléenne est appelée somme canonique de produits.

2.9.2 Produit canonique de sommes

Soient encore trois variables binaires x, y et z. Nous pouvons définir huit sommes logiques des trois variables faisant intervenir x ou son complément, y ou son complément et z ou son complément. La table suivante donne les tables de vérité de ces sommes. Nous constatons que chacune de ces fonctions ne prend la valeur 0 que pour une et une seule combinaison.

				S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
C_i	x	y	z	$x+y+z$	$x+y+\bar{z}$	$x+\bar{y}+z$	$x+\bar{y}+\bar{z}$	$\bar{x}+y+z$	$\bar{x}+y+\bar{z}$	$\bar{x}+\bar{y}+z$	$\bar{x}+\bar{y}+\bar{z}$
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

Reprenons l'exemple précédent de la fonction F. Celle-ci vaut 0 pour les combinaisons C_0, C_2, C_5, C_6 et C_7en même temps que S_0, S_2, S_5, S_6 et S_7. La fonction F peut donc être vue comme le produit logique de ces cinq sommes, ce qui est vérifié dans la table suivante. Nous pouvons donc exprimer la fonction F sous la forme suivante :

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

Cette écriture est appelée *produit canonique de sommes*. Celle-ci est moins utilisée que la somme canonique de produits.

C_i	x	y	z	F	$S_0 \bullet S_2 \bullet S_5 \bullet S_6 \bullet S_7$
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	0	0

2.10 Simplification de l'écriture des fonctions logiques

2.10.1 Définitions et notations

Le tableau de Karnaugh d'une fonction est la représentation matricielle de ses valeurs 1 et 0en fonction des variables.

Pour une fonction à n variables le tableau associé comportera donc 2^n cases. Le codage des états des lignes et des colonnes du tableau est un codage binaire réfléchi (ou code GRAY) de sorte qu'une et une seul evariable varie d'une case du tableau à une case adjacente.

Exemple de notation classique pour un tableau à 5 variables

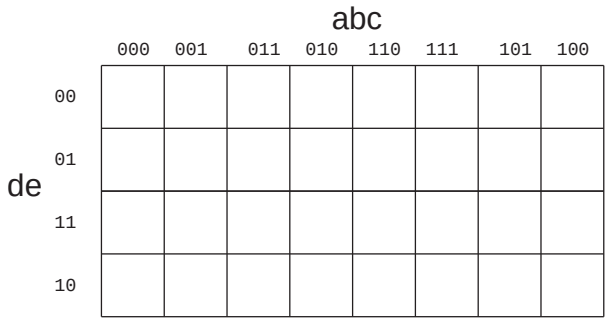


FIG. 2.8 – Tableau de Karnaugh à 5 variables

Notation de Girard qui ne fait figurer, sous forme d'une barre, que les états associés à la valeur 1 d'une variable.

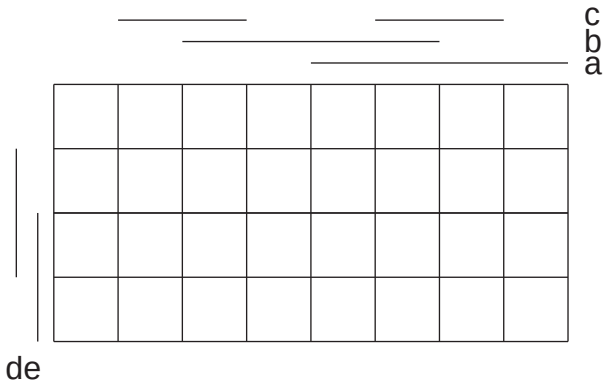


FIG. 2.9 – Tableau de Girard à 5 variables

2.10.2 États adjacents

Chaque état possède n adjacences (n nombre de variables). Chaque adjacence correspondant à un changement de valeur de l'une et une seule des n variables.

Le tableau suivant fait figurer les adjacences possibles (états adjacents) de deux états $E1$ et $E2$.

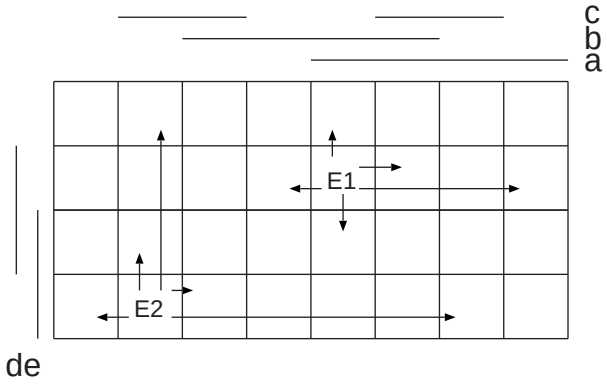


FIG. 2.10 – Tableau de Girard à 5 variables

2.10.3 Utilisation des tableaux de Karnaugh pour déterminer les fonctions logiques

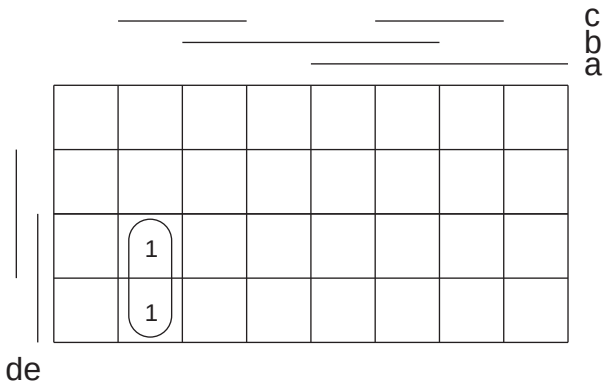
La méthode de recherche des expressions logiques minimales d'une fonction logique à partir des tableaux de Karnaugh, consiste à identifier les regroupements d'états où la fonction vaut 1 (regroupement ne comportant donc aucun zéro). Cette méthode fournit une forme canonique en OU.

Les règles pratiques de recherche sont les suivantes :

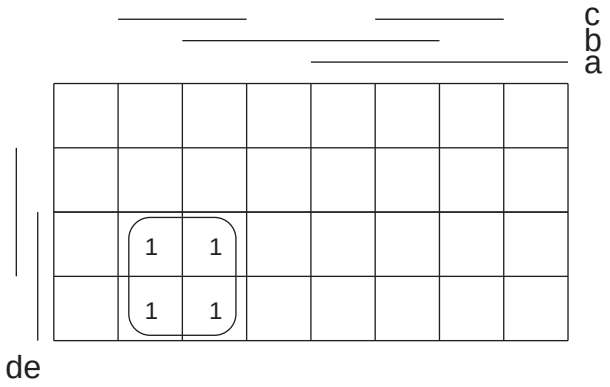
1. les regroupements ne doivent comporter que des 1,
2. un regroupement ne peut comporter que des états adjacents,
3. tout regroupement d'ordre p est un ensemble $m = 2^p$ états à 1. Le terme logique associé est un produit de degré $d = n - p$, s'exprimant donc avec d variables seulement.

   Il resulte de cette règle que toute association de k états où $k \neq 2^p$ ne peut correspondre à un regroupement associable à un terme logique.
4. dans un regroupement d'ordre p chaque états doit être adjacent à p autres états.

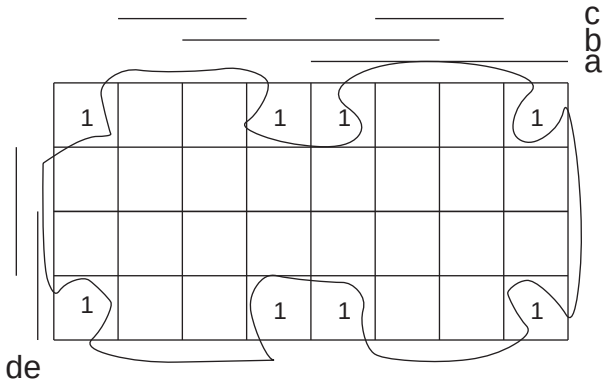
Illustration des règles Dans un tableau à 5 variables, un terme de degré $d = 5-1 = 4$ regroupe $m = 2^p = 2^1 = 2$ états.



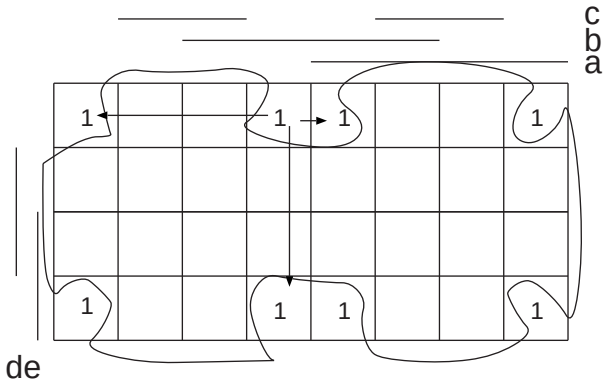
Dans un tableau à 5 variables, un terme de degré $d = 5-2 = 3$ regroupe $m = 2^p = 2^2 = 4$ états.



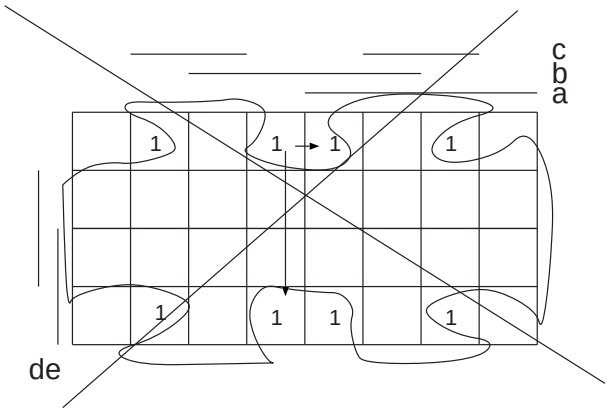
Dans un tableau à 5 variables, un terme de degré $d = 5-3 = 2$ regroupe $m = 2^p = 2^3 = 8$ états.



Règle 4 : vérifiée, chaque état est adjacent à 3 autres états dans la groupement ci-dessous d'ordre 3.



non vérifiée, l'association ci-dessous ne peut pas être un groupement au sens de Karnaugh.



2.10.4 Tableaux de Karnaugh

La méthode de simplification de Karnaugh repose sur l'identité :

$$(A \bullet B) + (A \bullet \overline{B}) = A \bullet (B + \overline{B}) = A$$

Elle est basée sur l'inspection visuelle de tableaux disposés de façon telle que les cases adjacentes en ligne et en colonne ne diffèrent que par l'état d'une variable et une seule.

Si une fonction dépend de n variables il y a 2^n produits possibles. Chacun de ces produits est représenté par une case dans un tableau. Les figures suivantes donnent la structure des tableaux de Karnaugh pour 2, 3, 4 et 5 variables. Pour 5 variables, deux représentations sont possibles. Le tableau de Karnaugh peut être traités comme deux tableaux 4x4 superposés ou un seul tableau de 4x8. Observez comment sont numérotées les lignes et les colonnes : d'une case à sa voisine une seule variable change d'état.

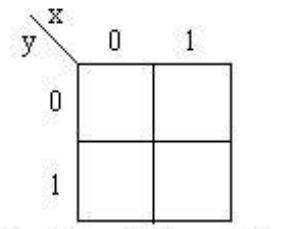


Tableau à 2 variables

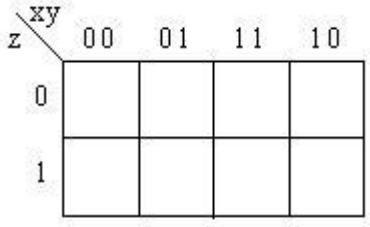


Tableau à 3 variables

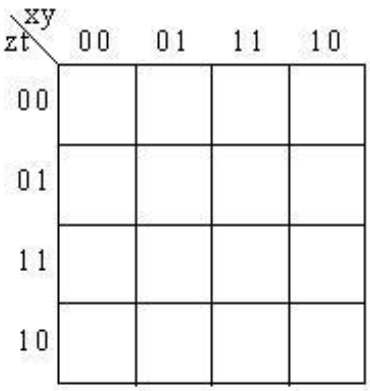


Tableau à 4 variables

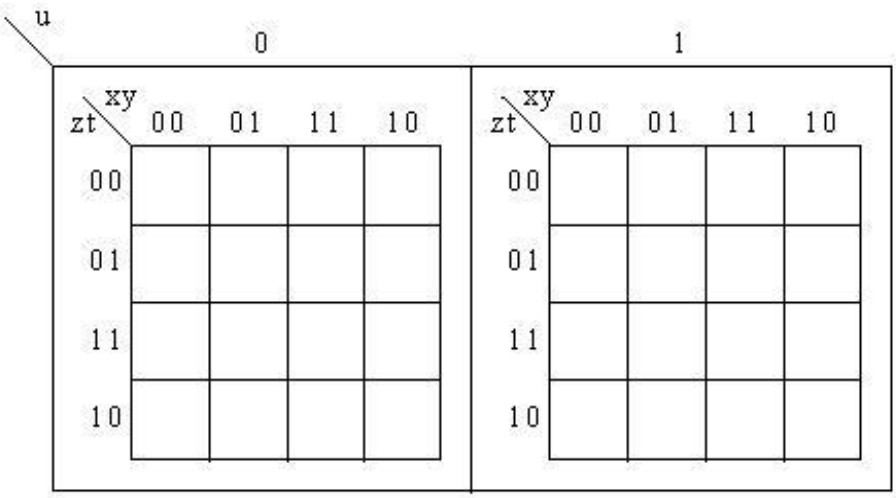


Tableau à 5 variables

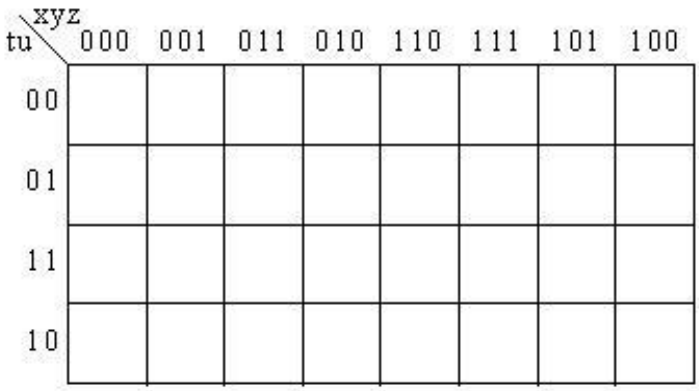
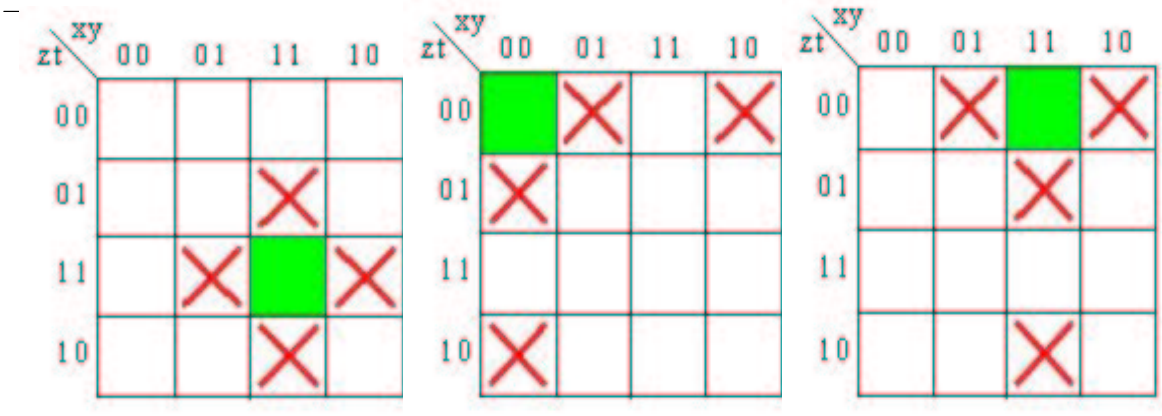


Tableau à 5 variables

Chaque case d'un tableau correspond au seul minterm prenant la valeur 1 pour la combinaison identifiée par la ligne et la colonne. Par exemple les trois cases coloriées dans les tableaux de la figure suivante correspondent respectivement aux produits suivants :

$$\mathbf{x\,y\,z\,t,}\quad \mathbf{\bar{x}\,\bar{y}\,\bar{z}\,\bar{t}}\quad \mathbf{et}\quad \mathbf{x\,y\,\bar{z}\,\bar{t}}$$

Il faut comprendre chaque ligne et chaque colonne comme une structure cyclique continue : chaque case a toujours quatre voisins qu'il faut éventuellement chercher à l'autre extrémité de la ligne ou de la colonne. Les tableaux suivants illustrent ce concept, les croix y matérialisent les voisins des cases coloriées :



Dans le cas de la représentation en deux tableaux superposés chaque case a cinq voisins : les quatre dans le même plan et une dans l'autre plan.

Le passage de la table de vérité au tableau de Karnaugh consiste à remplir chaque case avec la valeur de la fonction pour le produit correspondant. Il est possible de n'indiquer que les 1.

La méthode de simplification de Karnaugh consiste à rassembler les cases adjacentes contenant des 1 par groupes de 2, 4 ou 8 termes. Considérons en effet le groupement vertical de deux cases, en rouge, de la figure suivante. Il correspond à la somme de deux termes :

$$G = x\,y\,t + x\,y\,\bar{t}$$

Il est possible de factoriser le produit x y :

$$G = x\,y\,(t + \bar{t}) = x\,y$$

La variable t qui prend les deux valeurs 0 et 1 dans le groupement disparaît. Il ne reste que le produit des variables x et y, qui gardent ici la valeur 1.

Dans un groupement de deux termes on élimine donc la variable qui change d'état et on conserve le produit des variables qui ne changent pas. Dans un groupement de quatre on élimine les deux variables qui changent d'état. Dans un groupement de huit on élimine trois variables, etc...

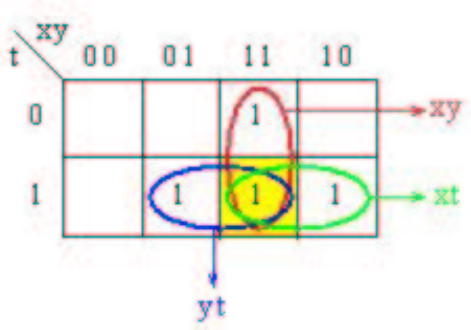
On cherche à avoir le minimum de groupements, chaque groupement rassemblant le maximum de termes. Une même case peut intervenir dans plusieurs groupements car C + C = C. C'est le cas de la case jaune sur la figure suivante.

Pour les cases isolées on ne peut éliminer aucune variable. On conserve donc le produit caractérisant la case. L'expression logique finale est la réunion des groupements après élimination des variables qui changent d'état.

Prenons l'exemple d'une fonction F définie par la table de vérité suivante :

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

La figure suivante donne le tableau de Karnaugh correspondant :



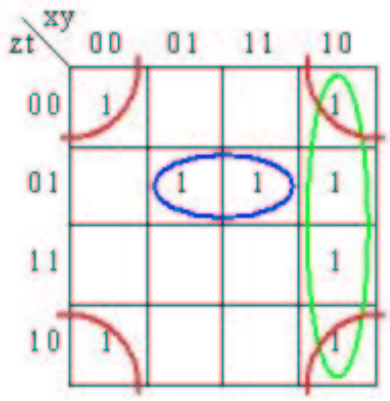
Nous y observons trois groupements de deux termes, nous pouvons écrire pour la fonction :

$$F = x\,y + y\,z + z\,x$$

Considérons une autre fonction F de quatre variables x, y, z et t définie par la table de vérité suivante :

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

La figure suivante donne le tableau de Karnaugh équivalent. Sur cette figure nous avons également matérialisé les trois groupements possibles : deux groupements de quatre termes, dont un contenant les quatre coins, et un groupement de deux termes.



Cette méthode nous permettent d'écrire :

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

Cette page est laissée blanche intentionnellement

Chapitre 3

Logique Combinatoire

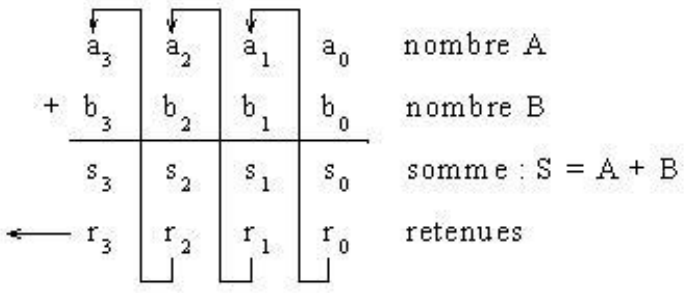
3.1 Addition binaire

3.1.1 Demi-additionneur

Addition et soustraction sont deux opérations arithmétiques de base. Commençons par l'addition de deux nombres binaires, la soustraction sera étudiée dans le prochain paragraphe. En base 2 l'addition de deux bits s'écrit :

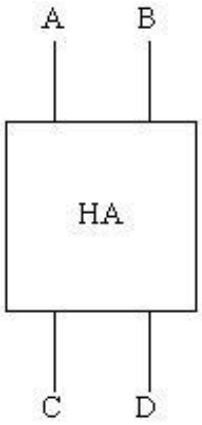
$$\begin{cases} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{cases}$$

Comme en décimal, nous devons donc tenir compte d'une éventuelle retenue (carry). La figure suivante montre la décomposition de l'addition de deux nombres binaires de quatre bits.



L'addition des deux bits de bas poids (LSB : *L*east *S*ignificant *B*it) a_0 et b_0, donne un résultat partiel s_0 et une retenue r_0. On forme ensuite la somme des deux bits a_1 et b_1 et de la retenue r_0. Nous obtenons un résultat partiel s_1 et une retenue r_1. Et ainsi de suite, nous obtenons un résultat sur quatre bits S et une retenue r_3.

Considérons la cellule symbolisée sur la figure suivante, comptant deux entrées A et B les deux bits à sommer et deux sorties D le résultat de la somme et C la retenue.



Ce circuit, qui permettrait d'effectuer l'addition des deux bits de plus bas poids est appelé demi-additionneur (Half-Adder). Ecrivons la table de vérité de celui-ci :

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

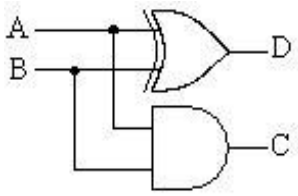
Si nous écrivons ces deux fonctions sous leur forme canonique il vient :

$$\begin{cases} D = \overline{A}\,B + A\,\overline{B} \\ C = A\,B \end{cases}$$

Nous reconnaissons pour la sortie D une fonction OU exclusif, donc :

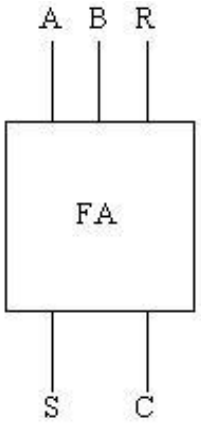
$$\begin{cases} D = A \oplus B \\ C = A\,B \end{cases}$$

Ce qui peut être réalisé par le circuit schématisé sur le logigramme suivant :

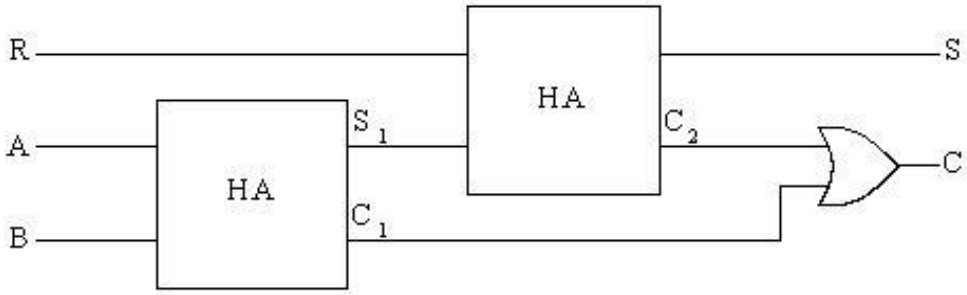


3.1.2 Additionneur

Il faut en fait tenir compte de la retenue des bits de poids inférieurs, un circuit additionneur doit donc comporter trois entrées et deux sorties, comme représenté sur la figure suivante :



Ce serait possible en combinant deux demi-additionneurs comme présenté ci-dessous. En pratique pour minimiser le nombre de composants, ou de portes dans un circuit intégré, un tel additionneur est réalisé directement.



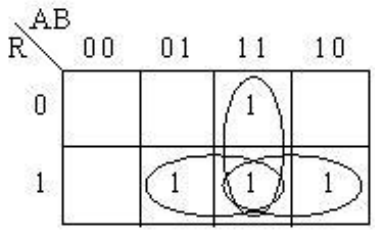
Les entrées A et B représentent les bits à additionner et R le report de la retenue de l'addition des bits de poids inférieurs. La sortie S représente le résultat de la somme et C la retenue. La table de vérité de ce circuit est la suivante :

A	B	R	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A partir de cette table nous pouvons écrire pour S et C les expressions booléennes suivantes :

$$\begin{cases} S = \overline{A}\,\overline{B}\,R + \overline{A}\,B\,\overline{R} + A\,\overline{B}\,\overline{R} + A\,B\,R \\ C = \overline{A}\,B\,R + A\,\overline{B}\,R + A\,B\,\overline{R} + A\,B\,R \end{cases}$$

Nous pouvons simplifier l'expression de C en utilisant un tableau de Karnaugh :



Nous en déduisons :

C = A B + A R + B R

Le bit de carry est égal à 1 si au moins deux des entrées sont à 1. D'autre part, nous pouvons remarquer qu'intervertir les 0 et les 1 dans la table 2 revient à permuter les lignes 1 et 8, 2 et 7, 3 et 6, 4 et 5. La table de vérité reste globalement invariante par inversion des entrées et des sorties, nous avons donc :

$$\overline{C} = \overline{A}\,\overline{B} + \overline{A}\,\overline{R} + \overline{B}\,\overline{R}$$

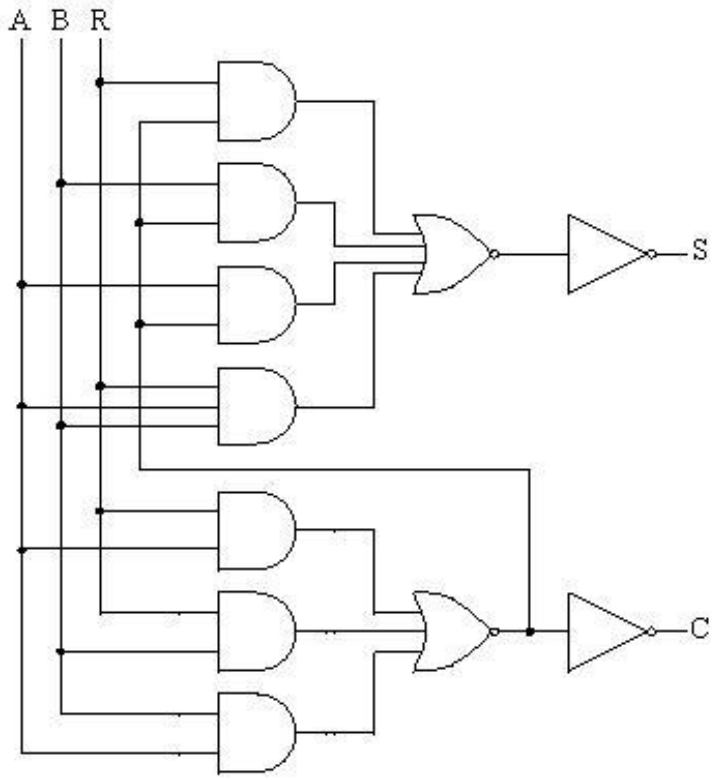
A partir de cette relation, nous pouvons écrire :

$$\begin{cases} A\,\overline{C} = A\,\overline{B}\,\overline{R} \\ B\,\overline{C} = \overline{A}\,B\,\overline{R} \\ R\,\overline{C} = \overline{A}\,\overline{B}\,R \end{cases} \Rightarrow (A+B+R)\,\overline{C} = A\,\overline{B}\,\overline{R} + \overline{A}\,B\,\overline{R} + \overline{A}\,\overline{B}\,R$$

Ce qui nous permet de réécrire l'expression de S :

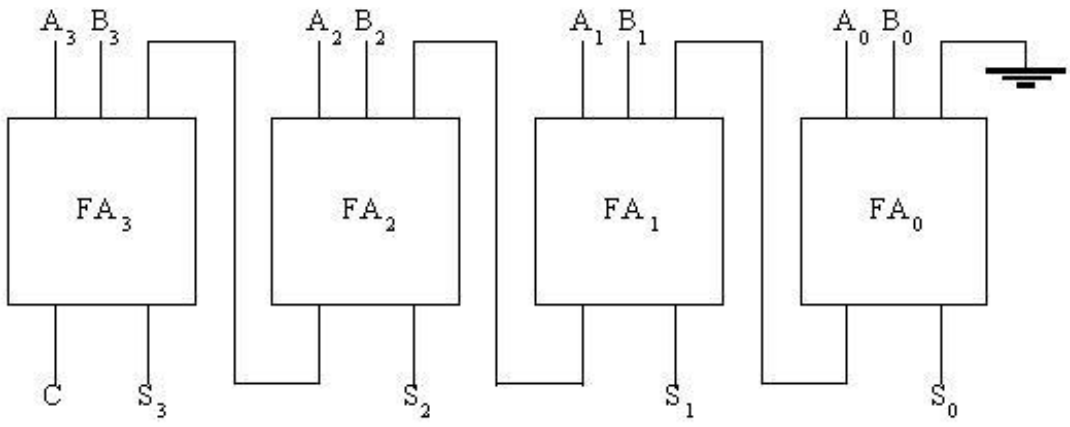
$$S = (A+B+R)\,\overline{C} + A\,B\,R$$

La figure suivante donne un exemple de réalisation d'un additionneur 1 bit basé sur deux *portes AOI* (AND OR INVERT), c'est-à-dire un ensemble de portes ET suivies d'une porte NON-OU.



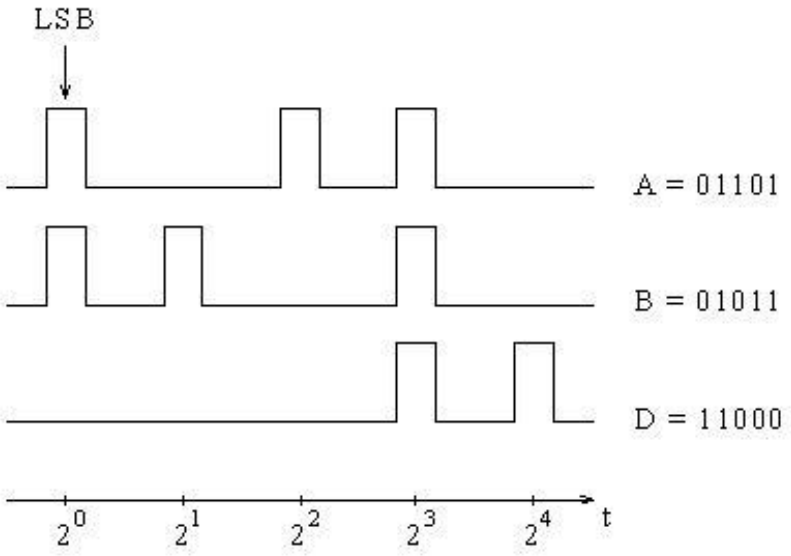
3.1.3 Addition en parallèle

L'addition de nombres comptant plusieurs bits peut se faire en série (bit après bit) ou en parallèle (tous les bits simultanément). La figure suivante montre l'exemple d'un additionneur 4 bits comptant quatre "Full Adders", montés en parallèle ou en cascade. Chaque additionneur FAi est affecté à l'addition des bits de poids i. L'entrée correspondant au report de retenue pour FA0 est imposée à 0 (en logique positive). La retenue finale C indique un dépassement de capacité si elle est égale à 1. Le temps d'établissement du résultat correspondant au temps de propagation des retenues au travers des diverses cellules. Si δ t est le temps réponse d'une cellule, la sortie S_0 et la retenue R_0 sont valables après un retard δ t, la sortie S_1 et la retenue R_1 ne sont correctes qu'après un retard 2 δ t, et ainsi de suite.

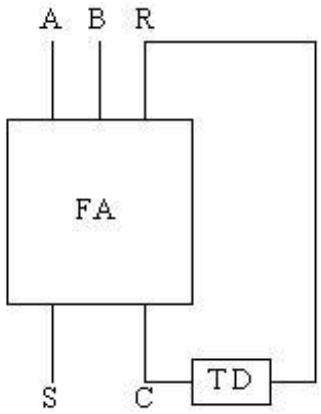


3.1.4 Addition séquentielle

Dans un additionneur séquentiel chacun des nombres A et B est représenté par un train d'impulsions synchrones par rapport à un signal d'horloge. L'ordre chronologique d'arrivée des impulsions correspond à l'ordre croissant des poids : le bit le moins significatif se présentant le premier.



Ces impulsions sont injectées sur les deux lignes d'entrée d'un additionneur. A chaque cycle d'horloge, la retenue provenant des bits de poids inférieurs doit être mémorisée (par exemple, à l'aide d'une bascule D qui sera étudiée dans le chapitre suivant).



Un additionneur parallèle est plus rapide mais nécessite plus de composants.

3.2 Soustraction

3.2.1 Demi-soustracteur

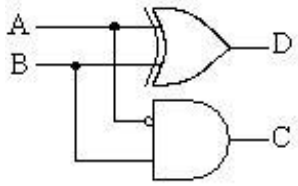
La table de vérité pour un demi-soustracteur (ne tenant pas compte d'une éventuelle retenue provenant des bits de poids inférieurs) est la suivante :

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

Où D représente le résultat de la soustraction A - B et C la retenue. Nous en déduisons les expressions logiques définissant D et C :

$$\begin{cases} D = \overline{A}\,B + A\,\overline{B} = A \oplus B \\ C = \overline{A}\,B \end{cases}$$

et le schéma correspondant :



Nous pourrions maintenant étudier un soustracteur prenant en compte la retenue. Nous allons plutôt tirer parti de certaines propriétés de la numération binaire pour traiter de la même manière l'addition et la soustraction.

3.2.2 Additionneur-soustracteur

Nous savons qu'avec un mot de n bits nous pouvons représenter un entier positif dont la valeur est comprise entre 0 et $2^n - 1$. Le complémentaire d'un mot de n bits est obtenu entre prenant le complément de chacun des n bits. Ainsi, si nous sommons un nombre et son complément nous obtenons un mot dont tous les bits sont à 1. C'est-à-dire :

$$A + \overline{A} = 2^n - 1$$

Attention : dans ce paragraphe le signe + représente l'opération addition et non la fonction logique OU. Nous pouvons encore écrire :

$$-A = \overline{A} + 1 - 2^n$$

Mais sur n bits l'entier 2^n est identique à 0 :

$$2^n \equiv 0 \quad (n\,bits)$$

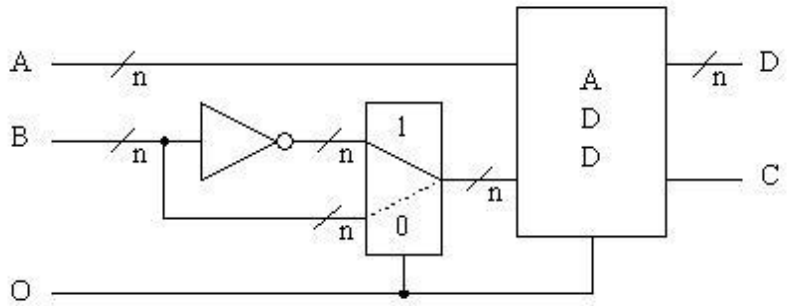
C'est-à-dire qu'il est possible d'écrire un nombre entier négatif comme le "*complément à 2*$-73 + 54 = -19$ de sa valeur absolue :

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

Nous reviendrons sur les divers codages des entiers signés plus tard. Nous pouvons utiliser cette propriété pour écrire la soustraction de deux mots de n bits sous la forme suivante :

$$A - B = A + \overline{B} + 1 - 2^n \equiv A + \overline{B} + 1 \quad \text{(n bits)}$$

Ce résultat conduit au schéma de principe présenté sur la figure suivante combinant les fonctions addition et soustraction. Celui-ci est basé sur l'emploi d'un additionneur n bits et d'un multiplexeur à deux lignes d'entrée. Nous étudierons ce type de circuit un peu plus loin dans ce chapitre. Selon le code opération O (0 pour une addition et 1 pour une soustraction) ce multiplexeur permet de sélectionner une des deux entrées, B ou son complémentaire. Le code opération est également injecté sur l'entrée report de retenue de l'additionneur. Pour simplifier le schéma et éviter de représenter n lignes de connexion parallèles, on ne matérialise qu'une seule ligne. Celle-ci est barrée et accompagnée d'une valeur qui indique le nombre réel de connexions.



3.3 Comparaison

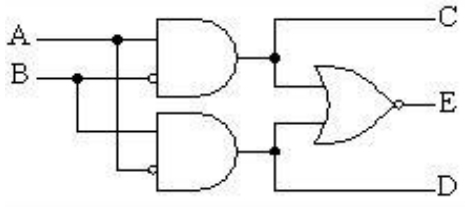
On rencontre très souvent la nécessité de comparer deux entiers (A = B, A > B ou A < B). Ecrivons la table de vérité correspondant à ces trois fonctions de comparaison de 2 bits. La fonction C doit être égale à 1 si et seulement si A > B, la fonction D si et seulement si A < B et la fonction E si et seulement si A = B. Ce qui nous donne :

A	B	C (A > B)	D (A < B)	E (A = B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Nous en déduisons les expressions logiques de C, D et E :

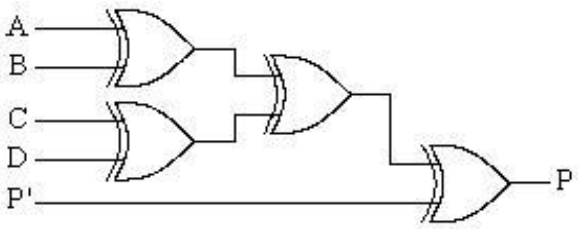
$$\begin{cases} C = A\,\overline{B} \\ D = \overline{A}\,B \\ E = \overline{A \oplus B} = \overline{A\,\overline{B} + \overline{A}\,B} = \overline{C + D} \end{cases}$$

La figure suivante présente le diagramme d'un bloc logique comparant deux bits A et B.

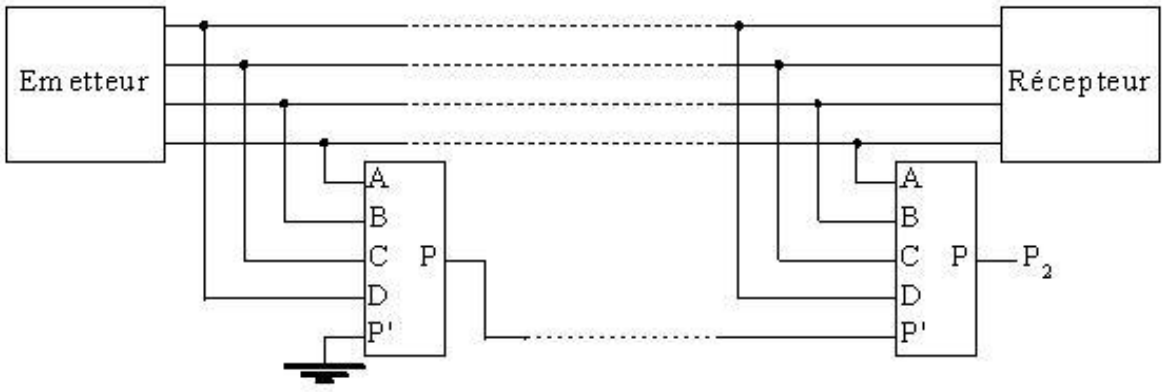


3.4 Contrôle de parité

La parité d'un mot binaire est définie comme la parité de la somme des bits, soit encore :
▷ parité paire (ou 0) : nombre pair de 1 dans le mot ;
▷ parité impaire (ou 1) : nombre impair de 1 dans le mot.
La fonction OU-exclusif donne la parité d'un sous-ensemble de deux bits. Le contrôle de parité est basé sur la constatation que le mot de n+1 bits formé en adjoignant à un mot de n bits son bit de parité est toujours de parité 0. La figure suivante représente le diagramme logique d'un générateur-contrôleur de parité pour 4 bits. Si l'entrée P' est imposée à 0 ce circuit fonctionne comme générateur de parité : la sortie P représente la parité du mot composé par les bits A, B, C et D.



Le contrôle de la parité est utilisé, par exemple, pour augmenter la fiabilité d'un système de transmission ou de stockage de données. La figure suivante montre l'utilisation du circuit précédent en générateur de parité du côté de l'émission et contrôleur de parité du côté de la réception. La sortie P2 doit être à 0 pour chaque mot transmis, sinon cela indique un problème de transmission.



Remarquons cependant que la parité ne permet de détecter qu'un nombre impair de bits en erreur dans un mot. Par ailleurs il ne permet pas corriger les erreurs détectées. Pour ce faire il faut utiliser des codes correcteurs d'erreur qui nécessitent plusieurs bits supplémentaires.

3.5 Décodage

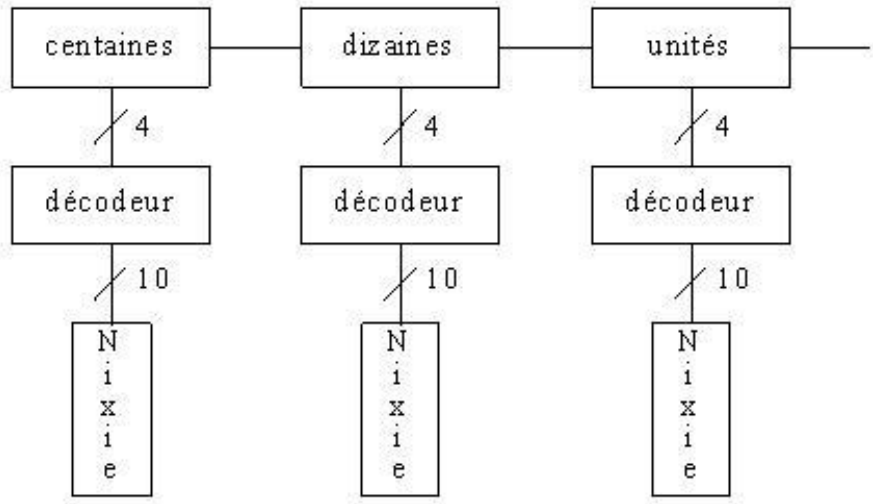
Dans un système numérique les instructions, tout comme les nombres, sont transportées sous forme de mots binaires. Par exemple un mot de 4 bits peut permettre d'identifier 16 instructions différentes : l'information est *codée*. Très souvent l'équivalent d'un commutateur à 16 positions permet de sélectionner l'instruction correspondant à un code. Ce processus est appelé *décodage*. La fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes une seule sortie active parmi les $N = 2^n$ sorties possibles. A titre d'exemple, nous allons étudier le décodage de la représentation DCB des nombres.

3.5.1 Représentation DCB (Décimale Codée Binaire)

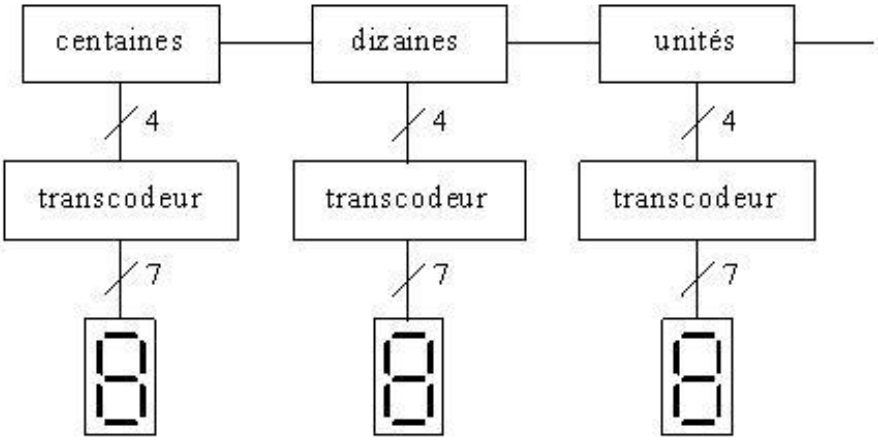
Le code DCB (ou en anglais BCD : *B*inary *C*oded *D*ecimal) transforme les nombres décimaux en remplaçant chacun des chiffres décimaux par 4 chiffres binaires. Cette représentation conserve donc la structure décimale : unités, dizaines, centaines, milliers, etc... Chaque chiffre est codé sur 4 bits selon :

Décimal	DCB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Par exemple le nombre décimal 294 sera codé en DCB : 0010 1001 0100. Ce type de codage permet, par exemple, de faciliter l'affichage en décimal du contenu d'un compteur. Pour ce faire on peut utiliser des tubes de Nixie, contenant 10 cathodes ayant chacune la forme d'un chiffre, ou des afficheurs lumineux à sept segment.



La fonction de chaque décodeur est d'activer une des dix lignes en sortie et une seule en fonction du code présent sur les quatre entrées. Par exemple, si ce code est égal à 5, la 6^{me} ligne de sortie est mise dans l'état 1 et le chiffre 5 est affiché par le tube de Nixie.



La fonction de chacun des *transcodeurs* est de positionner à 1 les lignes de sortie correspondant aux segments à allumer selon de code porté par les quatre lignes d'entrée. De manière générale, un transcodeur fait correspondre à un code A en entrée sur n lignes, un code B en sortie sur m lignes.

3.5.2 Décodeur DCB-décimal

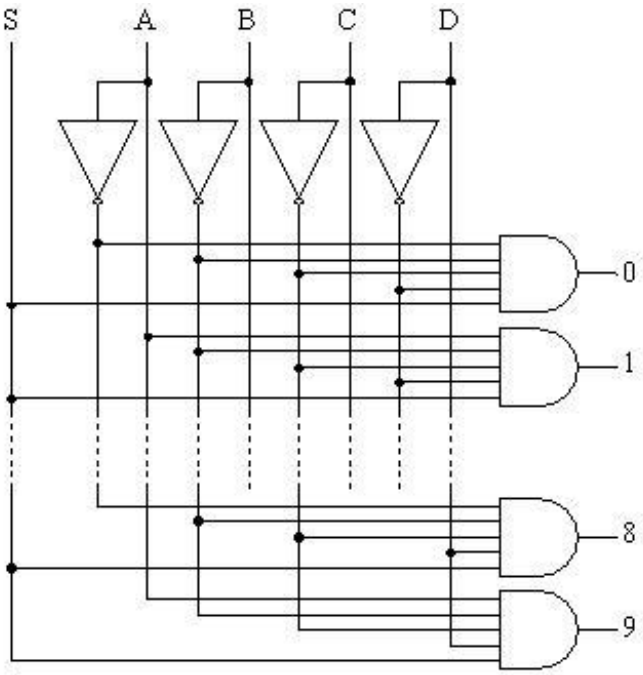
Nous allons étudier l'exemple d'un décodeur DCB-décimal. La table de vérité de ce décodeur est très simple :

D	C	B	A	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9
0	0	0	0	1									
0	0	0	1		1								
0	0	1	0			1							
0	0	1	1				1						
0	1	0	0					1					
0	1	0	1						1				
0	1	1	0							1			
0	1	1	1								1		
1	0	0	0									1	
1	0	0	1										1

A chacune des lignes de sortie nous pouvons associer un produit prenant en compte chacune des quatre entrées ou leur complément. Ainsi la ligne 5 correspond à :

$$A\,\overline{B}\,C\,\overline{D}$$

D'autre part, on souhaite souvent n'activer les lignes de sortie qu'en présence d'un signal de commande global (strobe ou enable). Ce signal S est mis en coïncidence sur chacune des dix portes de sortie. Dans l'exemple suivant, si S est dans l'état 0 le décodeur est bloqué et tous les sorties sont également dans l'état 0.



3.6 Multiplexage

Le multiplexage est un dispositif qui permet de transmettre sur une seule ligne des informations en provenance de plusieurs sources ou à destination de plusieurs cibles. La figure suivante en présente une analogie mécanique avec deux commutateurs à plusieurs positions. Choisir une ligne revient à définir l'angle du levier ou une adresse.

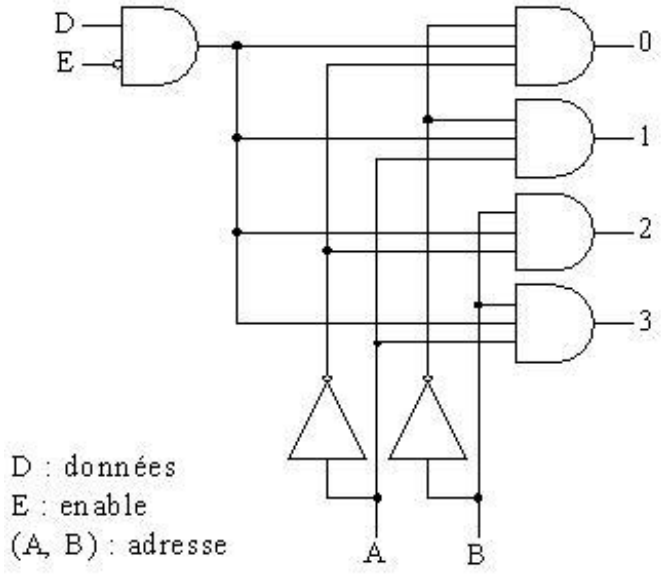
3.6.1 Démultiplexeur

Un démultiplexeur est un circuit comptant une entrée et N sorties et qui met en relation cette entrée avec une sortie et une seule. Pour pouvoir sélectionner cette sortie il faut également des lignes d'adressage : le code porté par ces lignes identifie la ligne de sortie à utiliser. Ce circuit est très proche d'un décodeur. Considérons un démultiplexeur avec quatre lignes de sortie. Il faut deux lignes d'adresse. Supposons que nous souhaitons également valider les données avec un signal de contrôle E (pour laisser par exemple le temps aux niveaux d'entrée de se stabiliser). Par convention nous choisissons de prendre en compte les données pour E = 0.

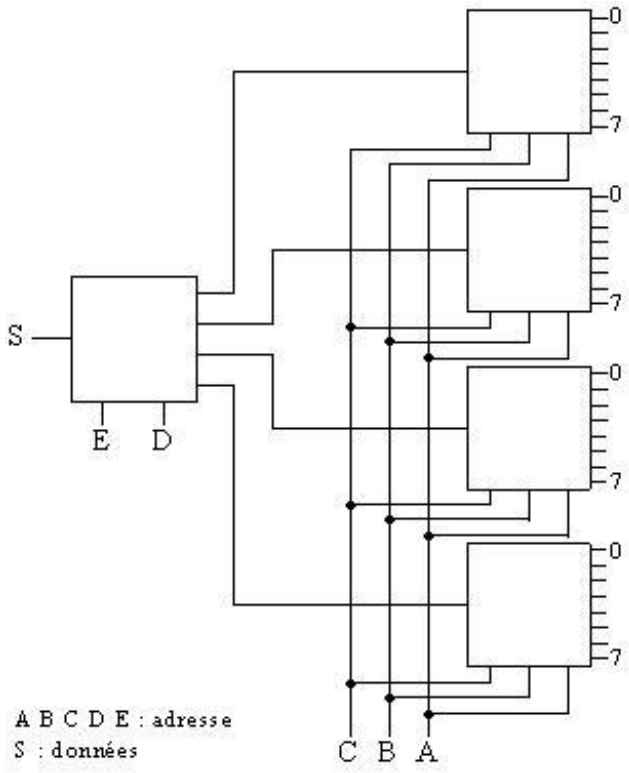
E	B	A	Y0	Y1	Y2	Y3	Produit
0	0	0	D	0	0	0	$\overline{A}\,\overline{B}\,\overline{E}\,D$
0	0	1	0	D	0	0	$A\,\overline{B}\,\overline{E}\,D$
0	1	0	0	0	D	0	$\overline{A}\,B\,\overline{E}\,D$
0	1	1	0	0	0	D	$A\,B\,\overline{E}\,D$

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

De cette table nous déduisons le logigramme suivant :



Il existe sous forme de circuits intégrés des démultiplexeurs avec 2, 4 ou 16 lignes de sortie. Pour constituer des démultiplexeurs d'ordre supérieur on peut être amené à cascader des démultiplexeurs. Par exemple un démultiplexeur avec 32 sorties peut être réalisé avec un "tronc" de 4 sorties et 4 "branches" de 8 sorties :



3.6.2 Multiplexeur

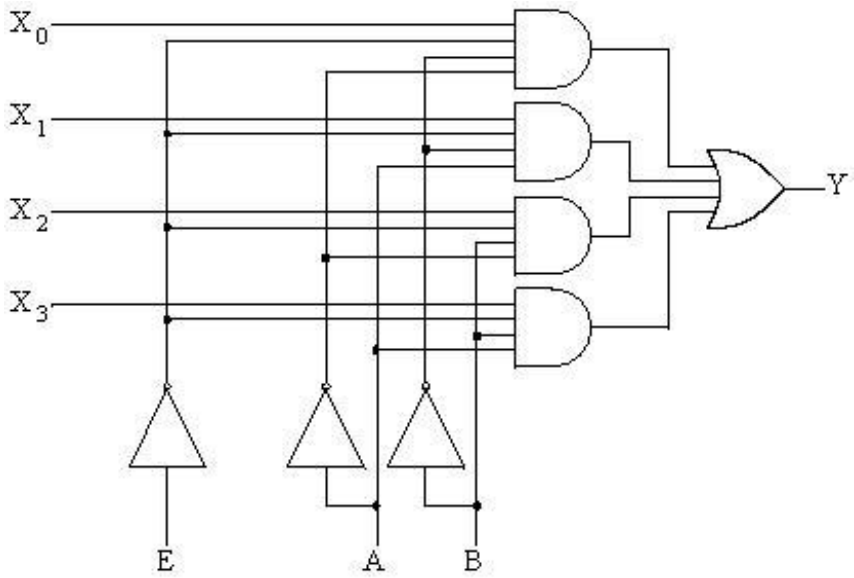
Un multiplexeur, réalise l'opération inverse. Il sélectionne une entrée parmi N et transmet l'information portée par cette ligne à un seul canal de sortie. Considérons un multiplexeur à quatre entrées, donc deux lignes d'adressage, et une ligne de validation. La table de vérité de ce circuit est donnée par la table suivante :

E	B	A	Y
0	0	0	X0
0	0	1	X1
0	1	0	X2
0	1	1	X3
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

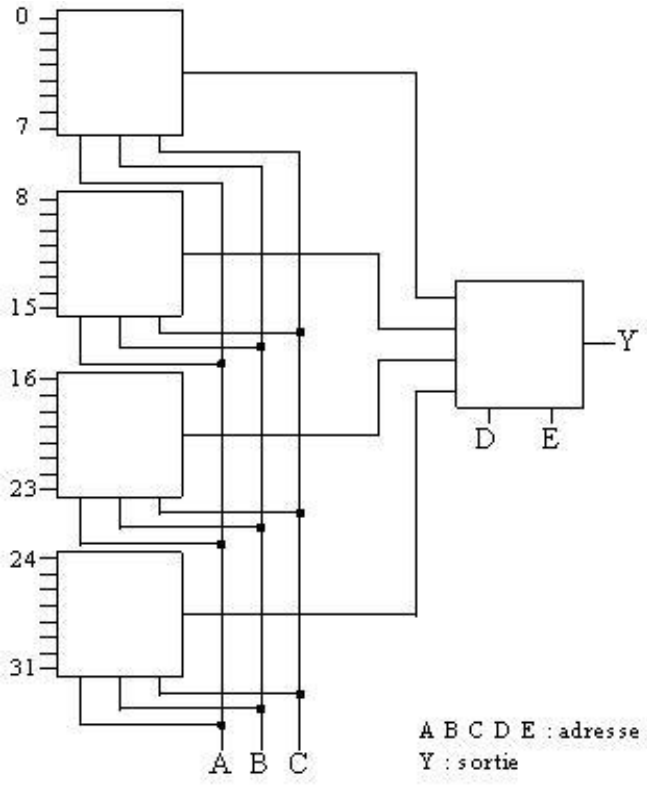
De cette table nous déduisons une expression logique pour la sortie :

$$Y = \overline{A}\,\overline{B}\,\overline{E}\,X_0 + A\,\overline{B}\,\overline{E}\,X_1 + \overline{A}\,B\,\overline{E}\,X_2 + A\,B\,\overline{E}\,X_3$$

Cette expression correspond au schéma suivant :



Tout comme pour les démultiplexeurs on peut cascader plusieurs multiplexeurs pour obtenir un multiplexeur d'ordre supérieur. La figure suivantemontre comment un multiplexeur à 32 entrées peut être réalisé à partir de quatre multiplexeurs à 8 entrées et d'un multiplexeur à 4 entrées.



3.6.3 Conversion parallèle-série

Considérons un mot de n bits (par exemple 4) présent en parallèle sur les entrées d'un multiplexeur :
▷ $X_0 \equiv$ bit correspondant à 2^0 ;
▷ $X_1 \equiv$ bit correspondant à 2^1 ;
▷ $X_2 \equiv$ bit correspondant à 2^2 ;
▷ $X_3 \equiv$ bit correspondant à 2^3.
Supposons que les lignes d'adresse A et B soient connectées aux sorties d'un compteur de période T, nous aurons en fonction du temps :

t	B	A	Y
[0,T]	0	0	X_0
[T,2T]	0	1	X_1
[2T,3T]	1	0	X_2
[3T,4T]	1	1	X_3
[4T,5T]	0	0	X_0

Les bits X_0, X_1, X_2 et X_3 se retrouvent en série dans le temps sur la sortie Y du multiplexeur.

3.7 Encodage

Nous venons d'étudier le principe du décodage, passons à l'opération inverse ou *encodage*. Un encodeur est un système qui comporte N lignes d'entrée et n lignes de sortie.

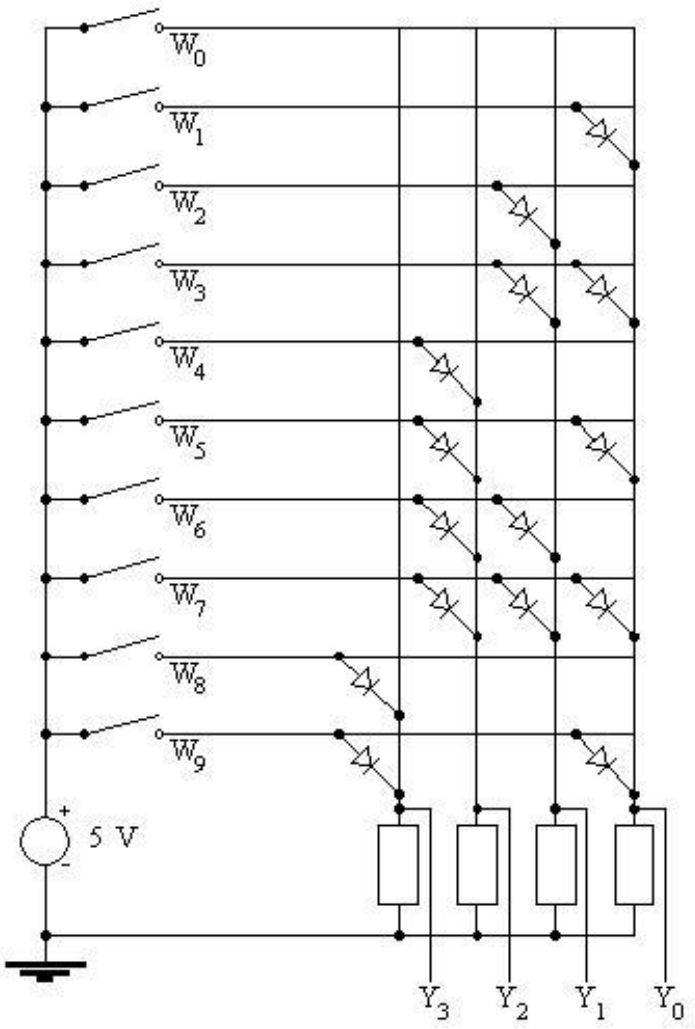
Lorsqu'une des lignes d'entrée est activée l'encodeur fournit en sortie un mot de n bits correspondant au codage de l'information identifiée par la ligne activée.

Considérons un encodeur transformant un nombre décimal en code DCB. Il comportera donc 10 entrées (0 à 9) et 4 sorties. Nous pouvons par exemple imaginer que chacune des dix lignes d'entrée peut être reliée à une touche d'un clavier. La table suivante correspond à la table de vérité de cet encodeur. A partir de cette table nous pouvons écrire les expressions logiques définissant les sorties à partir des entrées.

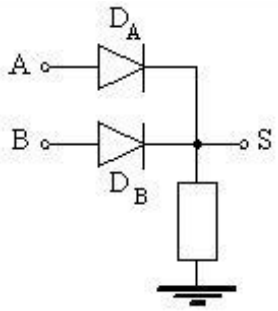
W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

$$\begin{cases} Y_0 = W_1 + W_3 + W_5 + W_7 + W_9 \\ Y_1 = W_2 + W_3 + W_6 + W_7 \\ Y_2 = W_4 + W_5 + W_6 + W_7 \\ Y_3 = W_8 + W_9 \end{cases}$$

En effet Y_0 est égal à 1 quand la ligne W_1 est dans l'état 1, ou la ligne W_3, ou la ligne W_5, ou la ligne W_7, ou la ligne W_9. La ligne Y_0 est nulle dans tous les autres cas. Il est possible de réaliser ces fonctions OU avec des diodes selon le montage de la figure suivante :



En effet considérons le circuit suivant :



A	B		S
0	0	D_A et D_B bloquées	0
+V	0	D_A passante et D_B bloquée	+V
0	+V	D_A bloquée et D_B passante	+V
+V	+V	D_A et D_B passantes	+V

Si nous traduisons la signification logique des niveaux haut et bas en logique positive, au circuit précédent correspond la table suivante. La fonction réalisée est donc un OU inclusif.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

La première figure représente un exemple de réalisation d'un encodeur DCB réalisé avec des diodes. Le bon fonctionnement de ce codeur suppose qu'une seule ligne d'entrée peut être dans l'état 1.

Par contre, si plusieurs entrées sont actives simultanément le résultat pourra ne pas avoir de signification. Par exemple, si les deux lignes W_7 et W_8 sont dans l'état 1 (frappe simultanée des deux touches), il en sera de même pour les quatre sorties. Pour éviter ce problème on utilise un *encodeur prioritaire*. Pour ce type de circuit si plusieurs lignes d'entrée sont actives simultanément le résultat correspondant à une seule parmi celles-ci est affiché en sortie. La règle peut être, par exemple, de mettre en sortie le code correspondant à la ligne d'entrée d'indice le plus élevé. Par exemple, si W_7 et W_8 sont dans l'état 1 l'encodeur prioritaire donne en sortie le code correspondant à W_8. La table de vérité correspondant à ce choix est donnée par la table suivante. Chaque croix indique que le code en sortie doit être indépendant de l'état de l'entrée concernée.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	0	0	0	1	0	0
X	X	X	X	X	1	0	0	0	0	0	1	0	1
X	X	X	X	X	X	1	0	0	0	0	1	1	0
X	X	X	X	X	X	X	1	0	0	0	1	1	1
X	X	X	X	X	X	X	X	1	0	1	0	0	0
X	X	X	X	X	X	X	X	X	1	1	0	0	1

Alors que les expressions logiques définissant les lignes de sortie Y_i ne dépendaient que des 1 dans la première table, il faut ici tenir compte des 0. Par exemple pour Y_0 nous avons :

$$Y_0 = W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_7\,\overline{W}_8\,\overline{W}_9 + W_9$$

Nous pouvons mettre le complémentaire de W_9 en facteur dans les quatre premiers termes, puis en utilisant l'identité :

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

il vient, après factorisation du complément de W_8 :

$$Y_0 = \overline{W}_8\,(W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7 + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7 + W_5\,\overline{W}_6\,\overline{W}_7 + W_7) + W_9$$

Soit encore :

$$Y_0 = \overline{W}_8\,(W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6 + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6 + W_5\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,((W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5 + W_3\,\overline{W}_4\,\overline{W}_5 + W_5)\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,((W_1\,\overline{W}_2\,\overline{W}_3\,\overline{W}_4 + W_3\,\overline{W}_4 + W_5)\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,(((W_1\,\overline{W}_2\,\overline{W}_3 + W_3)\,\overline{W}_4 + W_5)\,\overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\,(((W_1\,\overline{W}_2 + W_3)\,\overline{W}_4 + W_5)\,\overline{W}_6 + W_7) + W_9$$

Soit en réorganisant l'ordre des termes :

$$Y_0 = W_9 + \overline{W}_8\,(W_7 + \overline{W}_6\,(W_5 + \overline{W}_4\,(W_3 + W_1\,\overline{W}_2)))$$

Pour la ligne Y_1 nous avons :

$$\begin{aligned} Y_1 = \; & W_2\,\overline{W}_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 \; + W_3\,\overline{W}_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 \\ & + W_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 \qquad\qquad + W_7\,\overline{W}_8\,\overline{W}_9 \end{aligned}$$

Soit en factorisant :

$$Y_1 = \overline{W}_8\,\overline{W}_9\,(W_7 + \overline{W}_7\,(W_6 + \overline{W}_6\,(W_3 + W_2\,\overline{W}_3)\,\overline{W}_4\,\overline{W}_5))$$

En utilisant toujours la même identité nous pouvons simplifier cette expression, il vient en réordonnant les termes :

$$Y_1 = \overline{W}_9\,\overline{W}_8\,(W_7 + W_6 + \overline{W}_5\,\overline{W}_4\,(W_3 + W_2))$$

Pour Y_2 nous devons écrire :

$$Y_2 = W_4\,\overline{W}_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_5\,\overline{W}_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_6\,\overline{W}_7\,\overline{W}_8\,\overline{W}_9 + W_7\,\overline{W}_8\,\overline{W}_9$$

Soit encore :

$$Y_2 = \overline{W}_9\,\overline{W}_8\,(W_7 + \overline{W}_7\,(W_6 + \overline{W}_6\,(W_5 + \overline{W}_5\,W_4)))$$

En utilisant toujours la même identité il vient :

$$Y_2 = \overline{W}_9\,\overline{W}_8\,(W_7 + W_6 + W_5 + W_4)$$

Enfin pour Y_3 nous avons :

$$Y_3 = W_8\,\overline{W}_9 + W_9 = W_9 + W_8$$

Cette page est laissée blanche intentionnellement

Chapitre 4

Les Systèmes Automatisés de Production

4.1 Introduction

L'automatisation d'un processus industriel concerne tous les aspects de l'activité industrielle : production, assemblage, montage, stockage, contrôle, conditionnement, manutention, stockage, etc...

Un système automatisé de production à pour but de traiter une matière d'œuvre pour lui apporter une valeur ajoutée de façon reproductible et rentable.

4.2 Définition d'un Système Automatisé de Production

Un Système Automatisé de Production est un moyen d'assurer l'objetctif primordial d'une entreprise, la compétitivité de ses produits.

Un Système Automatisé de Production (S.A.P.) est plongé dans un environnement avec lequel un certain nombre de flux s'établissent :
▷ flux de produit(s) traité(s)
▷ flux d'énergie,
▷ flux d'informations,
▷ flux de nuisances,
▷ flux de déchets.

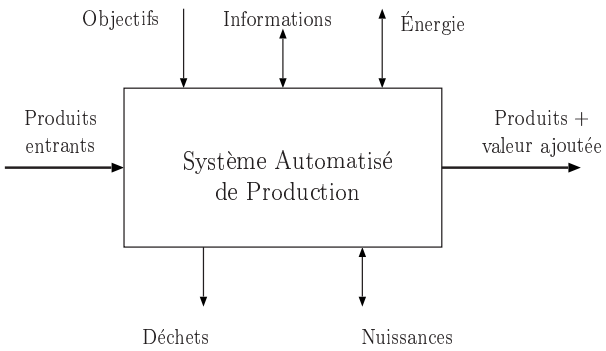


FIG. 4.1 – Flux d'un système automatisé de production

Ces flux lient le S.A.P. au sein d'un ensemble hieraechisé à d'autres S.A.P. afin de former un ensemble productif industriel (îlot, cellule ou ligne de fabrication).

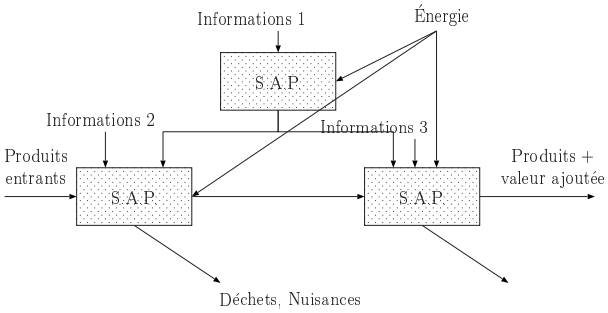


FIG. 4.2 – Flux de plusieurs S.A.P.

4.3 Typologie des système de production

4.3.1 Processus continu ou semi-continus

Un processus est dit continnu ou semi-continu lorsque les produits traités y sont introduits et extraits continuement en fonction du temps (raffinerie, aciéries, cimenteries...).

Un processus de nature continu qui doit être arrêté périodiquement est dit semi-continu.

4.3.2 Processus discontinu ou manufacturiers

Un processus est dit discontinu lorsque les produits traités, y sont introduits à l'unité ou par lots dissociables en unités (Usinage, assemblage).

4.3.3 Processus mixtes ou par lots

Ces processus associent un ou plusieurs processus de production continues avec des étapes de transformations discontinues.

4.3.4 Principales évolutions des systèmes de production

▷ Système non mécanisé : premiers systèmes apparus avec l'utilisation par l'homme d'un outil. L'homme apporte l'energie nécessaire au fonctionnement. Il assure aussi le commande.
▷ Système mécanisé : un système se mécanise lorsque l'on réduit la part d'énergie fournie par l'homme. L'énergie est fournie par d'autres sources et est dirigée vers des dispositifs appelés actionneurs. Entre la sources d'énergie et l'actionneur doit se trouver un élément destiné à canaliser le flux d'énergie et commandé par l'homme. L'homme conserve la commande du système. Energies principales : electrique, hydraulique, pneumatique. Actionneurs principaux : moteurs, vérins. Pré-actionneurs principaux : boutons poussoirs, interrupteurs, contacteurs, distributeurs, relais...
▷ Système automatisé : on remplace le savoir faire humain par des dispositifs dont l'ensemble forme la partie commande. Les sens de l'homme sont remplacés par des détecteurs et des capteurs. L'homme n'a plus qu'un rôle de surveillance. Certaines tâches restent manuelles et l'automatisation doit prendre en compte la spécificité du travail humain : assurer la dialogue homme machine, assurer la sécurité.

4.3.5 Objectifs d'un système de production

L'automatisation apparait comme un des moyens d'assurer la compétitivité. L'automatisation doit répondre à une grande variété d'objectifs.
▷ Gain en compétitivité : diminuer le coût de fabrication, réduire les stocks,
▷ Gain en qualité : réduire les délais, améliorer les performances, réduire les rebuts,
▷ Gain en flexibilité : améliorer l'opérabilité, convivialité,
▷ Gain en sureté : augmenter la sécurité et le disponibilité, réduire la pénibilité,
▷ Gain technique : impossibilité ou difficulté de conduite humaine.

4.4 Frontière d'un S.A.P.

La notion de système automatisé de production peut s'appliquer aussi bien à une machine isolée qu'à une unité de production, voire même à une usine ou un groupe d'usine.

Il est indispensable, avant toute analyse, de définir la frontière permettant d'isoler le système de production étudié de son milieu extérieur.

Cette démarche préliminaire permettra au concepteur d'identifier clairement les interactions du S.A.P. avec son environnement et donc de faciliter la spécification de ses fonctionnalités internes.

4.5 Structure d'un S.A.P.

D'une façon tout à fait générale, on peut décomposer fonctionnellement un S.A.P. en deux parties.

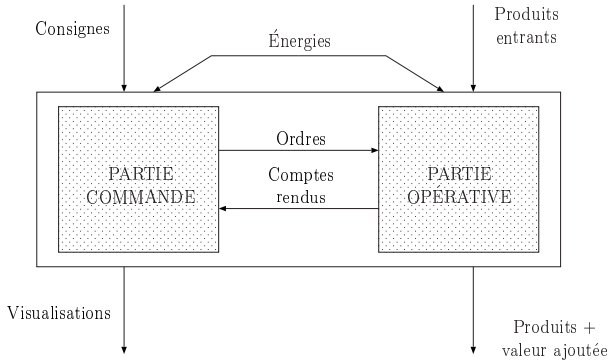


FIG. 4.3 – Décomposition fonctionnelle Partie Opérative/Partie Commande

4.5.1 La partie opérative (partie commandée)

C'est l'ensemble des dispositifs permettant d'apporter la valeur ajoutée. Elle met en œuvre un ensemble de processus physiques qui permettent la transformation des produits. Ces processus physiques nécessitent obligatoirement un apport d'énergie.

4.5.2 La partie commande (équipement de commande)

Automatiser la production consiste à transférer tout ou partie des tâches de coordination et des comandes auparavant exécutées par des opérateurs humains, dans un ensemble d'objets techniques appelé PARTIE COMMANDE.

La partie commande reproduit le savoir faire des concepteurs pour obtenir la suite des actions à effectuer sur les produits afin d'assurer la valeur ajoutée désirée. Pour ce faire, elle émet des ordres vers la Partie Opérative et en reçoit, en compte rendu, un ensemble d'informations.

Par ailleurs la Partie Commande est en interaction avec son milieu extérieur par des liaisons informationnelles avec l'environnement humain tout au long du cycle de vie du S.A.P. (première mise en œuvre, exploitation maintenance) et/ou avec d'autres Parties Commandes (via des réseaux de communication)

4.5.3 Fonctionnalité de la Partie Commande

La partie commande d'un système automatisé de production est destinée à traiter des informations afin de répondre aux fonctionnalités suivantes :
▷ gestion des entrées/sorties,
▷ traitement des équations combinatoires,
▷ traitement des fonctions de sécurité,
▷ traitement du séquentiel,
▷ fonction de régulation,
▷ commande d'axe(s) et asservissement,
▷ fonction de calculs,
▷ gestion d'outillage (usinages, montage, ...),
▷ contrôle de la qualité lié à la production,
▷ participation à la maintenance,

▷ suivi de la production,
▷ ...

4.5.4 Hiérarchie des communications

On distingue au sein d'une entreprise de production, 5 niveaux hiérarchisés (0 à 4), qui représente chacun des besoins bien spécifiques, tant en volume de données à transmettre qu'en temps de réponse nécessaire.

Le niveau 4 représente la gestion globale de l'usine avec planification générale et fait appel à l'informatique traditionnelle et concerne les services commerciaux et financier.

Le niveau 3 concerne la gestion de la production, les commandes, les approvisionnements, la centralisation des résultats de fabrication et provoque :
▷ l'établissement et le suivi des bilans relatifs à la quantité et la qualité des produits fabriqués.
▷ le suivi et la maintenance des installations.
Ces deux niveaux traitent et échangent des volumes de données très important. Les temps de communication sont rarement critique. Transfert de données en temps différés et parfois en temps réel.

Le niveau 2 concerne la commande centralisée d'un atelier, le contrôle d'une cellule. À ce niveau se fait la supervision et la gestion du changement de programme de fabrication par téléchargement des programmes dans les machines et automates de niveaux inférieurs. À ce niveau les volumes d'informations à échanger restent moindres que pour les niveaux 3 et 4 mais les temps de transferts doivent être très faibles afin de permettre un contrôle de processus en temps réels.

Le niveau 1 est celui de l'atelier, de la partie opérative. Les échanges d'informations s'effectuent entre automates placés en architecture distribuée permettant des fonctions de traitement réparties. Il faut assuré un transfert d'informations entre automates mais également mais également une gestion du temps pour synchroniser les différentes opérations de la partie opérative. Le nombre de station est relativement faible à ce niveau, elles échangent des volumes d'information également faibles mais avec des temps de communication garantis.

Le niveau 0 correspond aux échanges d'informations entre capteurs, pré-actionneurs, actionneurs et les automatismes de niveau 1 (automate, commande numérique). Les échanges de message sont très courts et s'effectuent entre un très petit nombre de stations mais avec des temps de réponse réduits.

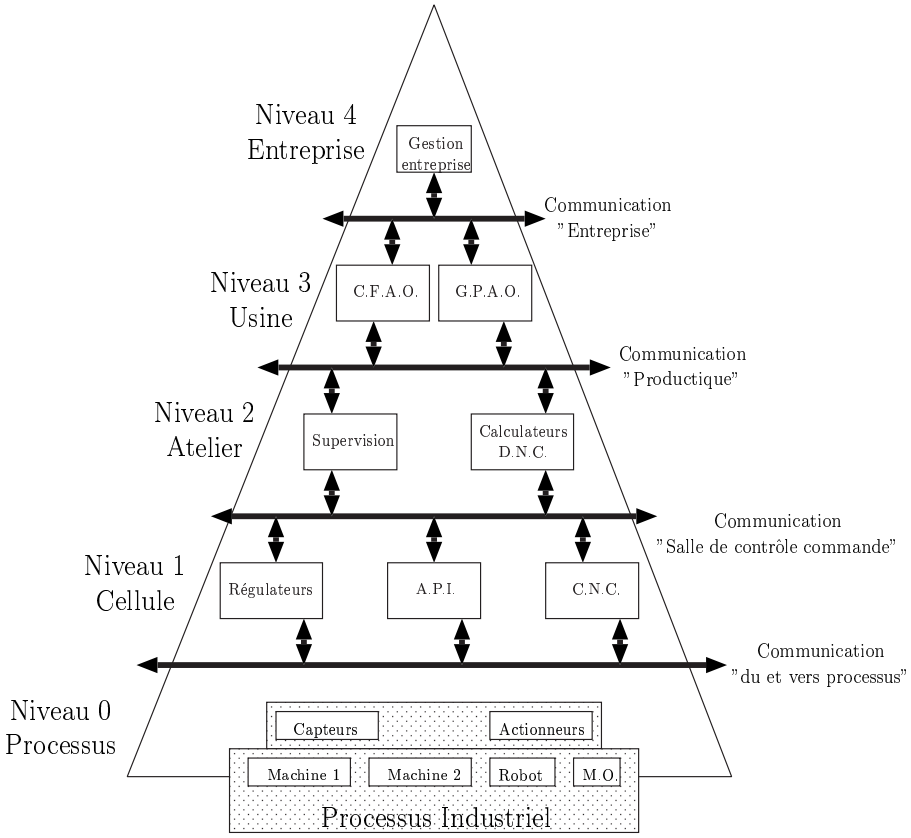


FIG. 4.4 – Hiérarchie des communications

Chapitre 5

Grafcet : Élements et structure de base

5.1 Historique

Le Grafcet : Graphe Fonctionnel de Commande Étape/Transition, est né en 1977 des travaux de l'AFCET (Association Française pour la Cybernétique Économique et Technique), en tant que synthèse théorique des différents outils existants à cette époque.

Mis sous sa forme actuelle par l'ADEPA, en 1979, normalisé en France en 1982, le Grafcet est aujourd'hui normalisé sur le plan internationnal sous l'appelation diagramme fonctionnel.

Depuis sa création le Grafcet est en perpétuel évolution, suite à différents travaux de recherche. Actuellement la norme est en cours de révision.

5.2 Domaine d'application de la norme internationale

La norme s'intitule : **Établissement des Diagrammes fonctionnels pour système de commandes**. Elle n'utilise pas le mot Grafcet, celui-ci est normalisé sous l'appelation Diagramme Fonctionnel.

Cette norme s'applique à l'établissement des descriptions de sla fonction et du comportement des systèmes de commandes, en établissant une représentation graphique indépendante de la réalisation technologique.

Le Grafcet est donc un outil indépendant de la technologie. Le Grafcet a été conçu pour décrire le fonctionnement de système logique. Mais il se révèle très performant dans l'étude et la conception de systèmes automatisés de production.

5.3 Éléments constitutifs d'un Grafcet

Un Grafcet est défini par un ensemble de symboles qui sont les étapes, les transitions et les liaisons orientées reliant les étapes aux transitions. Si ces symboles sont combinés suivant la manière préscrite, cet élément constitue la structure du Grafcet.

Cette structure constitue une représentation statique qui est interprétée par des actions qui peuvent être associées aux étapes et des réceptivités qui doivent être associées aux transitions.
À l'aide de cinq règles d'évolution, et de postulats temporels, le Grafcet prend alors un caractère dynamique correspondant aux évolutions dynamiques du processus.

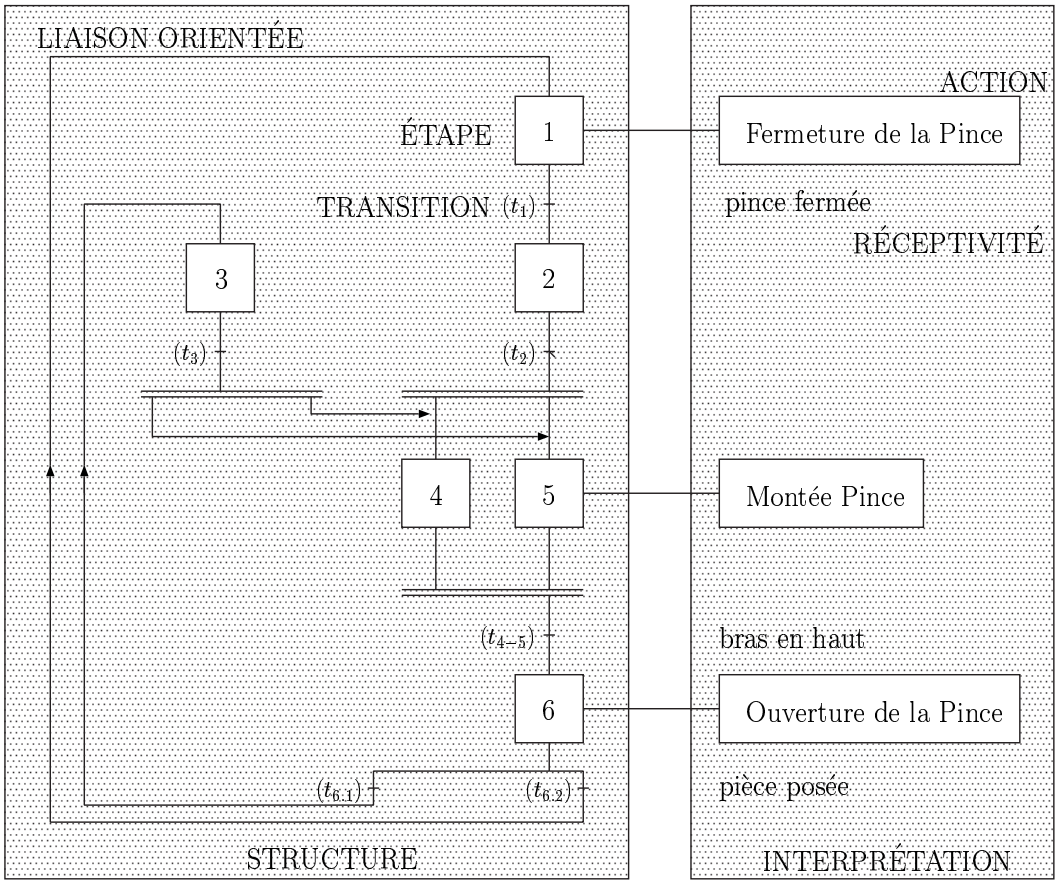


Fig. 5.1 – Élements constitutifs du Grafcet

Le Grafcet se compose d'une structure composée d'étapes, de transitions et de liaisons orientées et de son interprétation faite d'actions et de réceptivités. Il évoluera suivant cinq règles d'évolution et des postulats temporels.

5.4 La structure du Grafcet

5.4.1 les Étapes

5.4.1.1 Définition concernant les étapes

Une étape caractérise le comportement invariant d'une partie ou de la totalité du système isolé. À un instant donné et suivant l'évolution du système, une étape est soit **active** soit **inactive**. L'ensemble des étapes actives à un instant donné définit l'état du Grafcet à l'instant considéré. L'état actif ou inactif d'une étape peut être représenté par une variable booléenne notée X_i, i repère repère de l'étape, avec $X_i = 1$ quand l'étape est active et $X_i = 0$ quand l'étape est inactive.

5.4.1.2 Graphisme

L'étape se représente par un rectangle de rapport arbitraire, un carré étant recommandé (Norme CEI), par un carré (norme NF). Elle est identifiée par un repère alphanumérique.

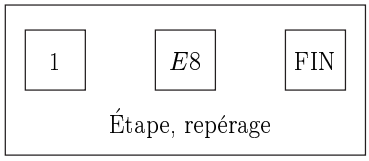


FIG. 5.2 – Élements constitutifs du Grafcet : l'Étape

L'Entrée est figurée à la partie supérieure et la sortie à la partie inférieure de chaque symbole d'étape.

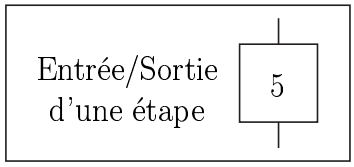


FIG. 5.3 – Élements constitutifs du Grafcet : l'Étape : entrée et sortie

Pour indiquer les étapes qui sont actives à un instant considéré, un point peut être placé dans la partie inférieure des symboles des étapes concernées.

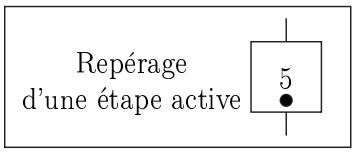


FIG. 5.4 – Élements constitutifs du Grafcet : l'Étape : repérage d'une étape active

5.4.1.3 Regroupement des liaisons orientées en amont et en aval des étapes

Plusieurs liaisons orientées (venant de différentes transistions), peuvent arriver à l'entrée d'une étape, dans ce cas elles sont regroupées en amont de l'étape à l'aide d'une représentation comme ci-dessous.

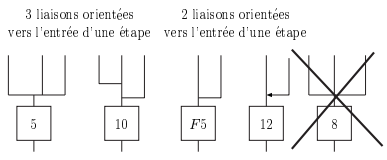


FIG. 5.5 – Élements constitutifs du Grafcet : l'Étape : regroupement des liaisons

De même plusieurs liaisons orientées peuvent partir de la sortie d'une étape pour aller à plusieurs transitions, le regroupement s'effectue en aval de l'étape à l'aide d'une représentation ci-dessous.

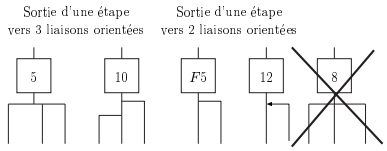


FIG. 5.6 – Élements constitutifs du Grafcet : l'Étape : regroupement des liaisons

Afin de ne pas faire d'erreur de graphisme il suffit de considérer pour l'étape un symbole général possédant autant de point d'entrées et de points de sorties que nécesaire. Les liaisons ne peuvent alors accéder qu'à ces points.

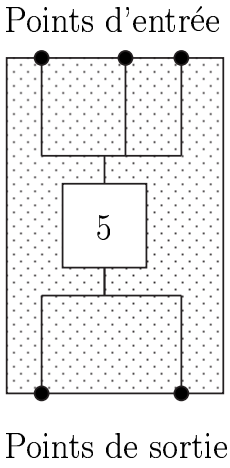


FIG. 5.7 – Élements constitutifs du Grafcet : l'Étape : représentation générale

5.4.2 Les transitions

5.4.2.1 Définition concernant les transitions

Une transition indique la possibilité d'évolution entre étapes. Cette évolution s'effectue par le **franchissement** de la transition ce qui provoque un changement d'activité des étapes (défini par les règles d'évolution).

5.4.2.2 Graphisme

Une transition est représentée par un trait perpendiculaire aux liaisons joignant deux étapes. Il n'y a toujours qu'une seule transition entre deux étapes, quels que soient les chemins parcourus.

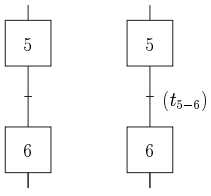


FIG. 5.8 – Élements constitutifs du Grafcet : la Transition : représentation générale

Pour faciliter la description du Grafcet, chaque transition peut être repérée de façon alphanumérique, entre parenthèses, à gauche du symbole de transition.

5.4.2.3 Regroupement des liaisons orientées en amont ou en aval des transitions

Lorsque plusieurs étapes sont reliées à une même transition, les liaisons orientées d'entrées et/ou de sortie de ces étapes sont regroupées en amont ou en aval par le symbole de synchronisation représenté par deux traits parallèles horizontaux.

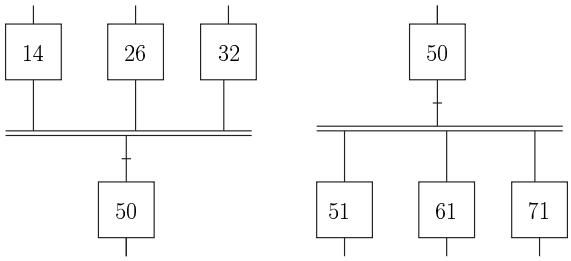


FIG. 5.9 – Élements constitutifs du Grafcet : la Transition : regroupement des liaisons orientées

Comme pour les étapes, il suffit de considérer un symbole général de la transition pour ne pas faire d'erreur de graphisme.

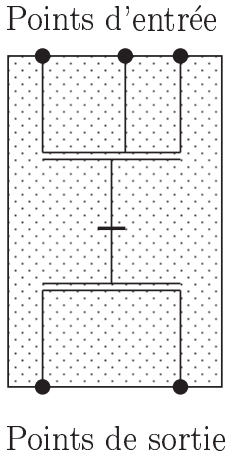


FIG. 5.10 – Élements constitutifs du Grafcet : la Transition : représentation généralisée

5.4.2.4 Validation d'une transition

À un instant donnée et suivant l'évolution du système, une transition est **validée** ou **non validée**. Elle est dite validée si toutes les étapes immédiatement précédentes reliées à cette transition sont actives.

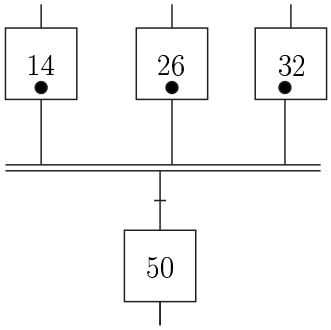


FIG. 5.11 – Élements constitutifs du Grafcet : la Transition : validation

5.4.3 Liaisons orientées

Les liaisons orientées sont des traits qui relient les étapes aux transitions et les transitions aux étapes. Elle indiquent les voies d'évolution du Grafcet.

Par convention, le sens d'évolution s'effectue toujours du haut vers le bas. Des flèches doivent être utilisées lorsque cette convention n'est pas respectée ou éventuellement lorsque leur présence peut apporter une meilleure compréhension sur le sens des évolutions.

Lorsqu'une liaison doit être interrompue, diagrammes complexes ou représentations sur plusieurs pages, des renvois sont utilisés pour assurer la continuité de la lecture. Des repères indiqueront pour chaque liaison l'étape ou la transition d'origine ou de destination ainsi que les numéros de pages ou de folios. Pour une meilleure lisibilité, il faut éviter de couper la liaison étape/transition de façon à mettre en évidence toutes les transitions validées par une même étape.

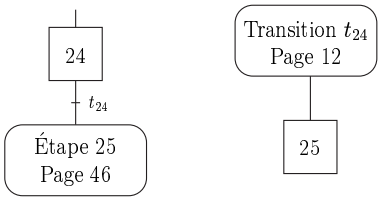


FIG. 5.12 – Élements constitutifs du Grafcet : la Liaison orientée

5.4.4 Analyse d'une structure grafcet

Pour analyser une structure grafcet, afin de déterminer si elle est juste vis à vis des règles de graphisme, définies précédemment, il faut décomposer en éléments de bases ce grafcet et vérifier que :

- ▷ l'alternance étape-transistion est respectée ;
- ▷ que chaque élément, étape ou transistion est correctement représenté et bien relié par ses points d'entrée et de sortie à des liaisons orientées.

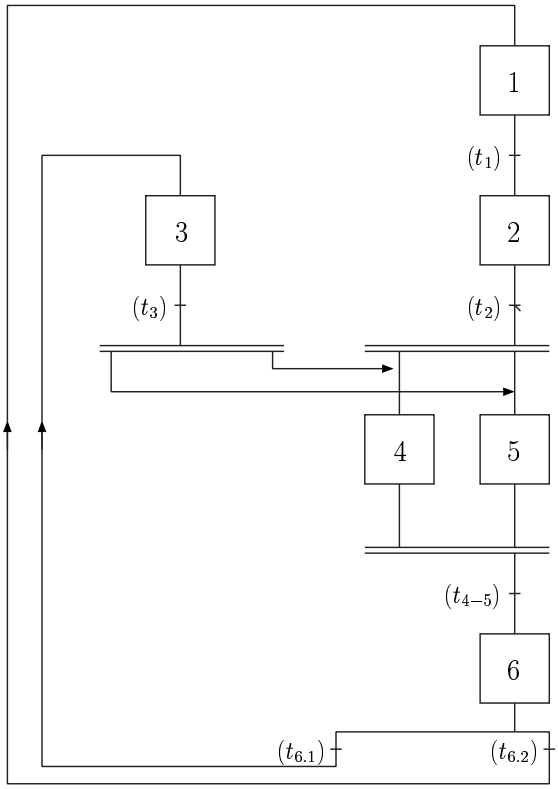


FIG. 5.13 – Analyse d'une structure grafcet

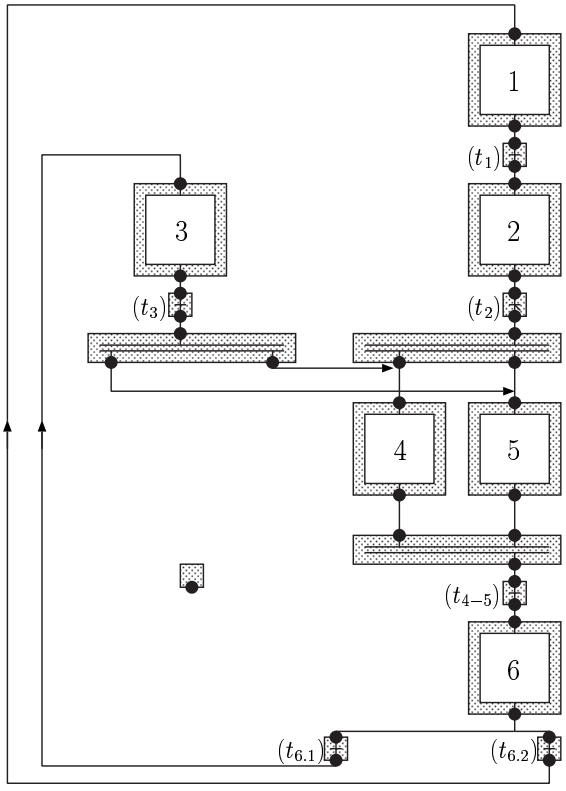


FIG. 5.14 – Analyse d'une structure grafcet

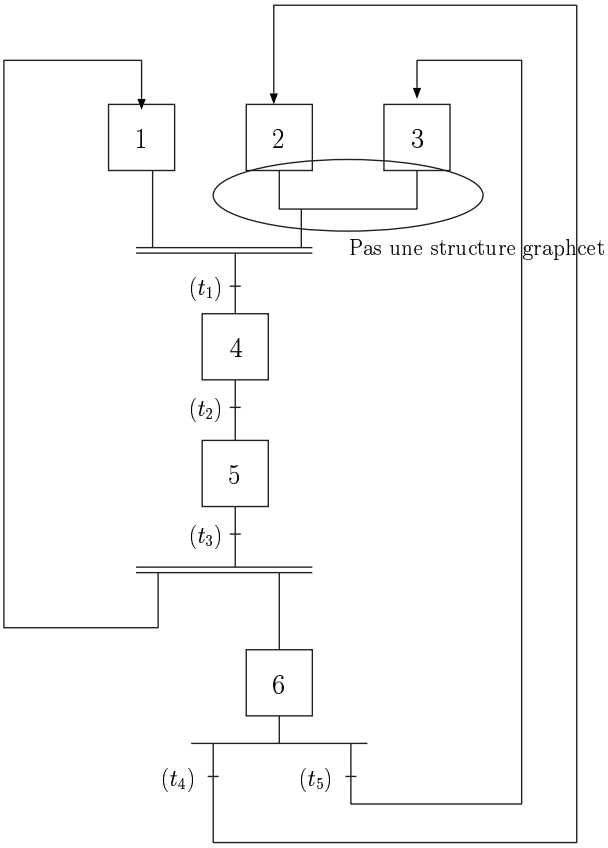


FIG. 5.15 – Analyse d'une structure grafcet : Erreur

5.5 Interprétation du grafcet

L'interprétation du grafcet est constituée des actions et des receptivités.

5.5.1 Les actions

Une action est une description de ce qui doit être effectué par la Partie Opérative (P.O.), la Partie Commande (P.C.) ou des organes extérieurs. Les actions correspondent à des émissions d'ordres.

Lorsqu'une étape est active les actions associées à l'étape sont exécutées.

Les actions sont décrites de façon littérale ou symbolique à l'intérieur d'un ou plusieurs rectangles de dimensions quelconques reliés par un tiret au symbole de l'étape à laquelle elles sont associées.

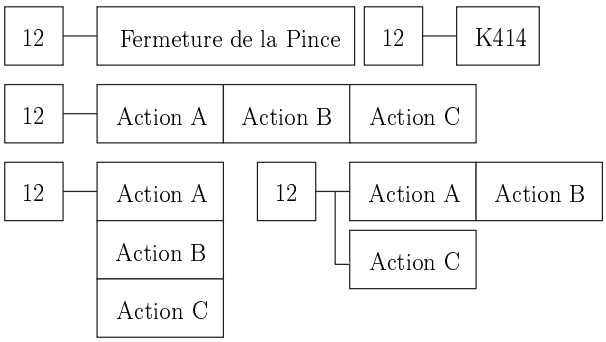


FIG. 5.16 – Interprétation du grafcet : Étapes

Une étape ne comportant aucune indication d'action peut correspondre à un comportement d'attente, d'auto-synchronisation entre plusieurs grafcets.

Plusieurs actions associées à une étape peuvent être disposées de différentes façons sans que cette représentation implique une priorité entre ces actions.

Lorsque les actions sont décrites de manières symboliques, un tableau de correspondance entre le nom symbolique utilisé et l'action qui lui correspond doit être établi.

K414	Fermeture de la Pince
K412	Ouverture de la Pince
K314	Avance Vérin 3
K312	Retour Vérin 3

FIG. 5.17 – Interprétation du grafcet : Tableau de correspondance

Des commentaires peuvent être utilisés pour apporter davantage de précision. Pour éviter toute confusion, ces commentaires sont écrits entre guillemets.

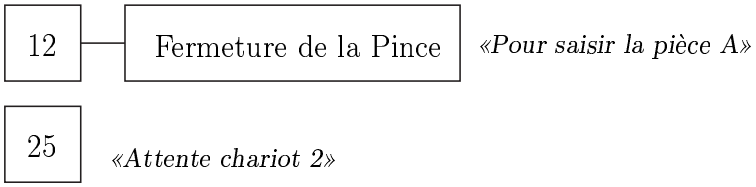


FIG. 5.18 – Interprétation du grafcet : Commentaires

5.5.2 Description détaillées des actions

La norme CEI/IEC 848 spécifie une représentation générale des ordres ou des actions qui doivent s'exécuter.

FIG. 5.19 – Interprétation du grafcet : Description des actions

La section «a» contient une (ou une combinaison de) lettre symbole décrivant les caractéristiques logiques d'association de la sortie à l'activité de l'étape.

▷ C : Action conditionnelle,
▷ D : Action retardée,
▷ L : Action limitée dans le temps,
▷ P : Action impulsionnelle,
▷ S : Action mémorisée.

La section «b» contient la déclaration symbolique ou littérale décrivant l'action.

La section «c» indique le repère de référence du signal de fin d'exécution correspondant. Les sections «a» et «c» ne sont spécifiées que si nécessaire. Si elle ne sont pas spécifiées, l'action correspondante est dite continue.

5.5.2.1 Action continue

Une action associée à une étape est dite continue lorsque sa durée d'exécution est identique à la durée d'activité de l'étape. Dans ce cas les sections «a» et «c» sont omises.

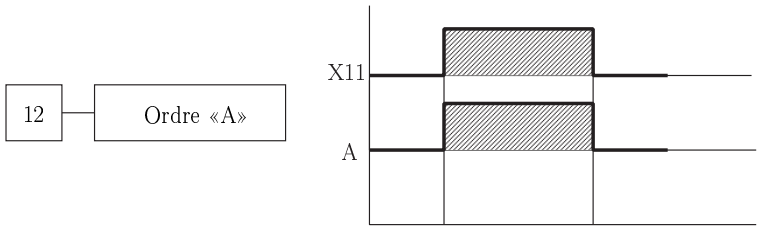


FIG. 5.20 – Interprétation du grafcet : Action continue

5.5.2.2 Action conditionnelle

Une action conditionelle est une action dont l'exécution est soumise à une condition logique. Cette condition peut être, une variable d'entrée, une variable d'étape ou un résultat booléen d'une combinaison de plusieurs de ces variables.

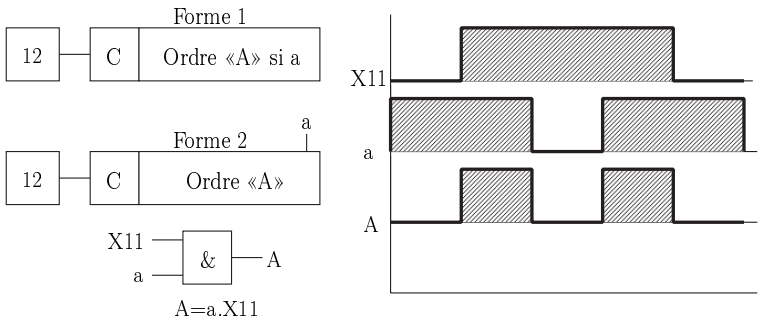


FIG. 5.21 – Interprétation du grafcet : Action conditionnelle

5.5.2.3 Action retardée ou limitée dans le temps

Les actions retardées ou limitées dans le temps sont des cas particuliers d'actions conditionnelles (le temps intervant comme condition logique). La lettre «D» (Delayed action) décrit une action retardée par rapport à l'activation de l'étape à laquelle elle est associée.

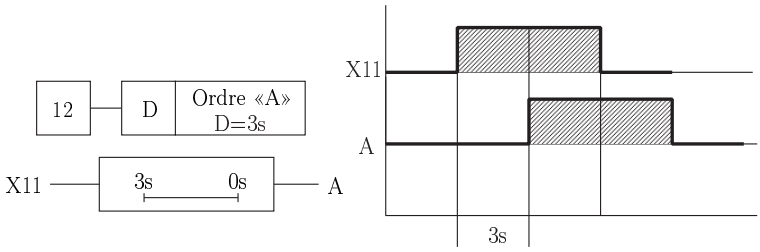


FIG. 5.22 – Interprétation du grafcet : Action retardée

La lettre «L» (Time Limited action) indique une action dont la durée est limitée dans le temps.

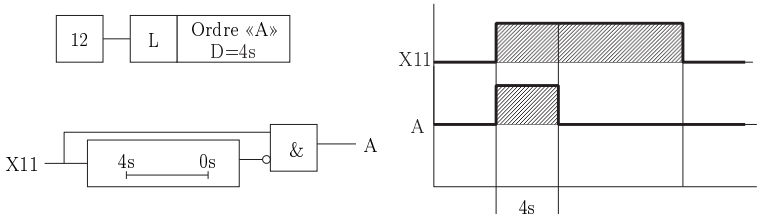


FIG. 5.23 – Interprétation du grafcet : Action limitée dans le temps

Dans les deux cas l'unité de temps retenue doit être spécifiée.

5.5.2.4 Action impulsionnelle

Les actions impulsionnelles sont des actions temporisées de très courtes durées dont la valeur est sans importance mais suffisante pour obtenir l'effet souhaité.

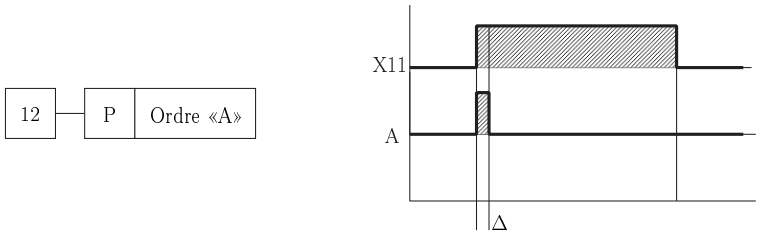


FIG. 5.24 – Interprétation du grafcet : Action impulsionnelle

5.5.2.5 Action mémorisée

Pour qu'une action reste maintenue lorsque l'étape qui l'a commandée vient d'être désactivée il faut utiliser une action mémorisée.

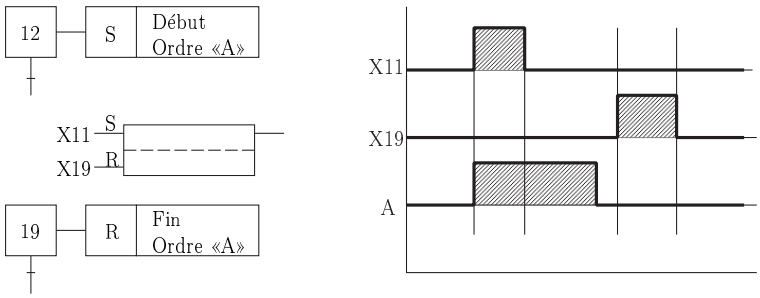


FIG. 5.25 – Interprétation du grafcet : Action mémorisée

La documentation UTE C03-191 propose de modifier cet ordre de la manière suivante :
▷ Début d'action mémorisée (SET)
▷ Fin d'action mémorisée (RESET)

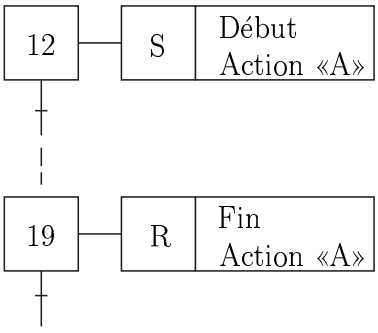


FIG. 5.26 – Interprétation du grafcet : Action mémorisée

5.5.3 Les receptivités

À chaque transition est associée une receptivité qui est une condition logique qui regroupe des informations logiques issues de la P.O., de la P.C. ou d'organes extérieurs.

La réceptivité est une fonction booléenne inscrite, de manière symbolique ou littérale, à droite de la transition.

FIG. 5.27 – Interprétation du grafcet : Receptivités

La notation «= 1» associée à une transition indique une receptivité toujours vrai. Nous verrons à l'aide des règles d'évolutions, qu'une receptivité = 0 n'a pas de sens car elle empêche toute évolution du grafcet.

5.5.3.1 Prise en compte du temps

Norme NFE L'indication du temps s'effectue par la notation «$t/i/q$» qui fixe, après la lettre «t», le repère de l'étape «i» déterminant l'origine du temps et ssa durée «q».

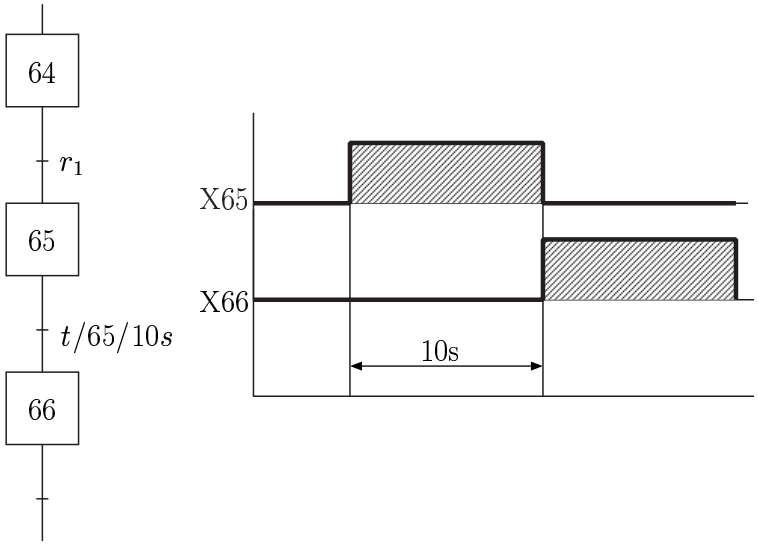


FIG. 5.28 – Interprétation du grafcet : Receptivités et temps

cette norme étant antérieure à la norme CEI, cette représentaion ne devrait plus être utilisée.

Norme CEI/IEC Comme les symboles graphiques sont permis, la dépendance du temps peut être mise en évidence par le symbole d'un opérateur binaire à retard. Dans le langage littéral et les expressions booléennes, la notation $t_1/a/t_2$ est recommandée. La conditiuon de transition de $t_1/a/t_2$ devient vrai après un retard t_1 référé au passage de l'état logique 0 à l'état logique 1 de la variable booléenne a, elle redevient fausse après un retard t_2 référé au passage de l'état logique 1 à l'état logique 0 de la variable booléenne a.

Si t_1 ou t_2 est égal à 0, les notations a/t_2 ou t_1/a sont à utiliser. t_1 et t_2 doivent être remplacés par leurs valeurs rélles.

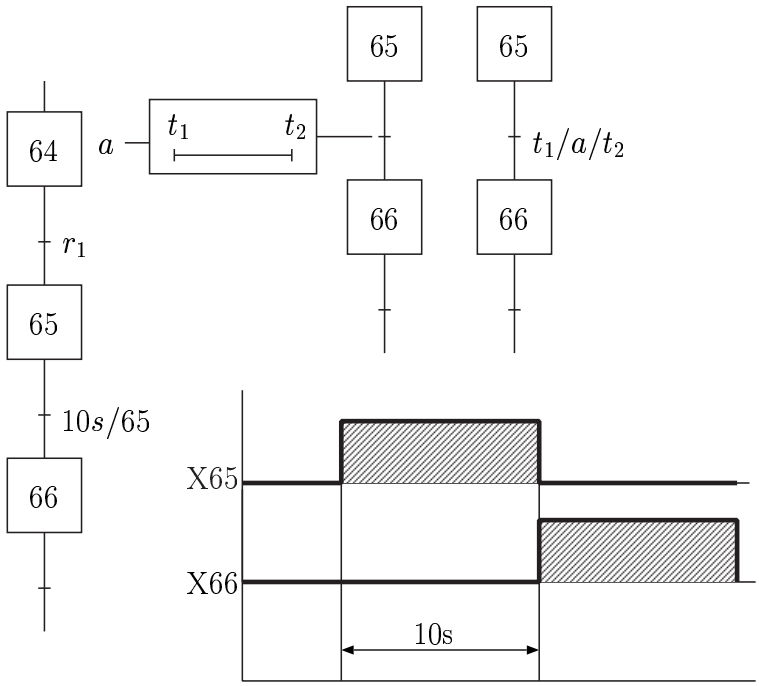


FIG. 5.29 – Interprétation du grafcet : Receptivités et temps

5.5.3.2 Prise en compte des changement d'états d'informations

La notation «↑ a», front montant de la variable a, indique que la receptivité est vrai (= 1) qu'à l'instant du passage de la variable a de 0 à 1. De même la notation «↓ a», front descendant de la variable a, indique que la receptivité est vrai (= 1) qu'à l'instant du passage de la variable a de 1 à 0.

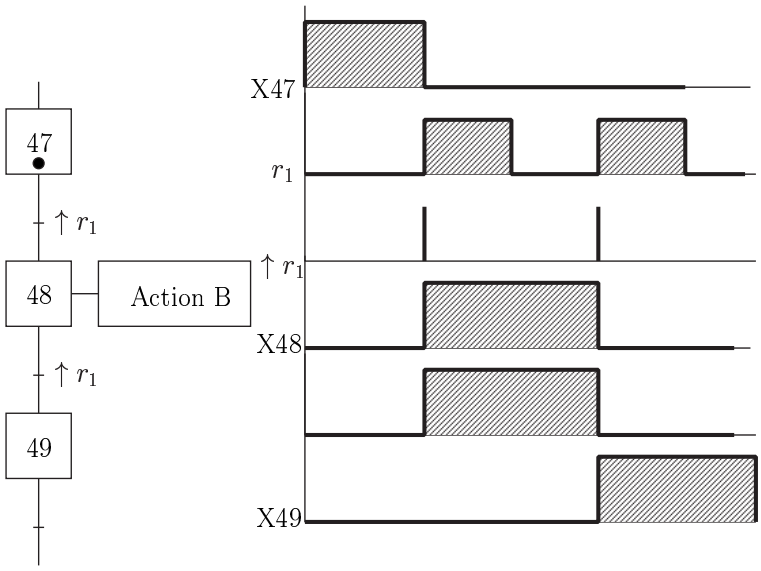


FIG. 5.30 – Interprétation du grafcet : Receptivités et fronts

5.6 Les règles d'évolutions du grafcet

Le Grafcet obéit à 5 règles.

5.6.1 Règle 1 : Situation Initiale

La situation initiale d'un grafcet caractérise le comportement initial de la partie commande vis à vis de la partie opérative., de l'opérateur et ou des éléments extérieurs. Elle correspond aux étapes actives (étapes initiales) au début du fonctionnement.

Une étape initiale se représente par un double carré, elle est activée inconditionnellement en début de cycle.

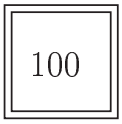


FIG. 5.31 – Règles d'évolution du grafcet : Étape initiale

La situation initiale traduit généralement un comportement d'attente, de repos.

5.6.2 Règle 2 : Franchissement d'une transition

Le franchissement d'une transition obéit à deux conditions
▷ la transition doit être validée,
▷ la réceptivité associée doit être vraie.
Lorsque la transition est franchissable, elle est obligatoirement franchie.

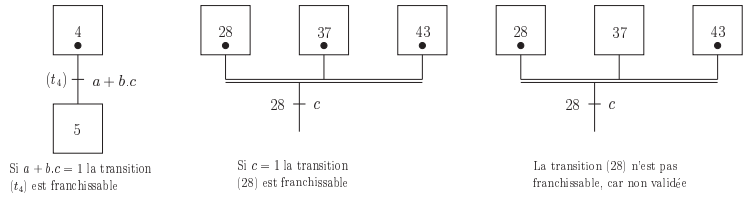


FIG. 5.32 – Règles d'évolution du grafcet : Franchissement d'une transition

5.6.3 Règle 3 : Évolution des étapes actives

Le franchissement d'une transition entraîne simultanément l'activation de toutes les étapes immédiatement suivantes, et la désactivation de toutes les étapes immédiatement précédentes.

5.6.3.1 Exemple 1

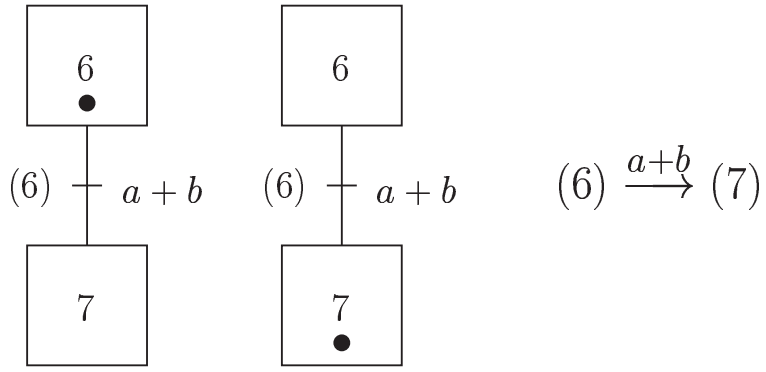


FIG. 5.33 – Règles d'évolution du grafcet : Évolutions des étapes actives

Lorsque $a + b = 0$ la transition (6) n'est pas franchissable, quand $a + b = 1$ la transition franchissables est immédiatement et obligatoirement franchie. Simultanément l'étape (6) est désactivée et l'étape (7) est activée.

5.6.3.2 Exemple 2

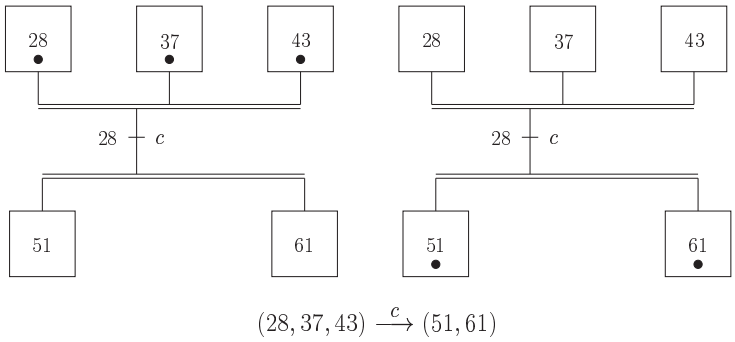


FIG. 5.34 – Règles d'évolution du grafcet : Évolutions des étapes actives

$c = 0$, la transition est validée (28,37 et 43 actives), mais non franchissable. Quand $c = 1$, simultanément les étapes 28, 37 et 43 sont désactivées et les étapes 51 et 61 sont activées.

5.6.4 Règle 4 : Évolution simultanées

Plusieurs transitions simultanément franchissables sont simultanément franchies.

5.6.4.1 Exemple 1

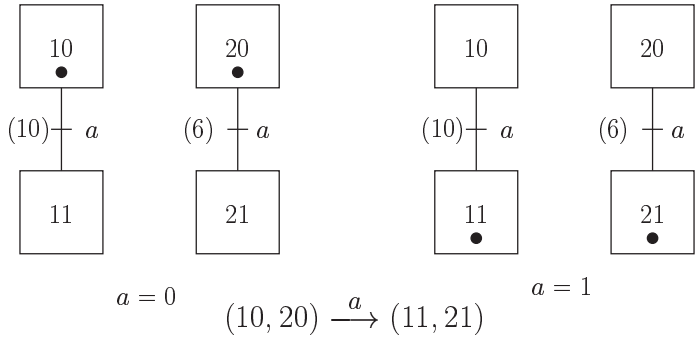


FIG. 5.35 – Règles d'évolution du grafcet : Évolutions simultanées

5.6.4.2 Exemple 2

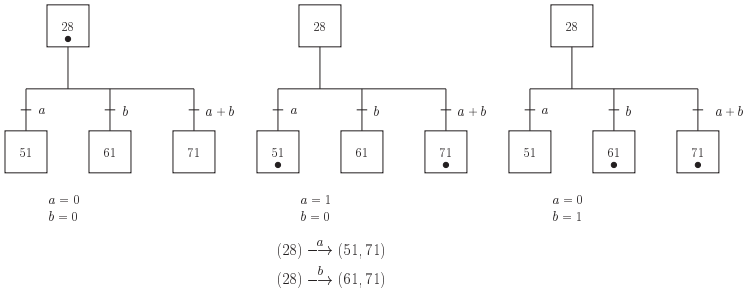
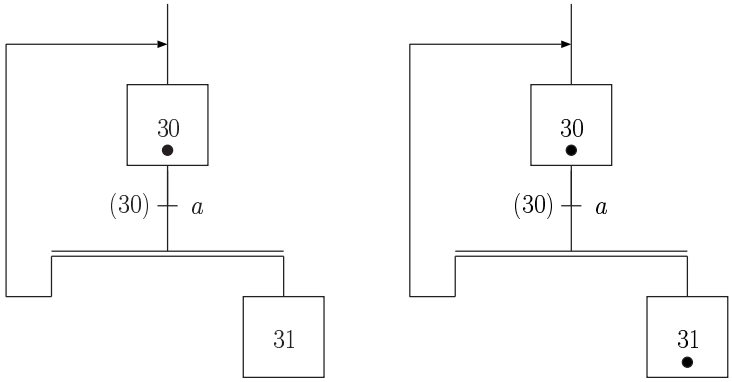


FIG. 5.36 – Règles d'évolution du grafcet : Évolutions simultanées

Une astérisque peut être utilisée pour indiquer des transistions simultanément franchissables.

5.6.5 Règle 5 : Activation et Désactivation simultanée d'une étape

Si au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste active.



$$(30) \xrightarrow{a} (30, 31)$$

FIG. 5.37 – Règles d'évolution du grafcet : Activation et Désactivation simultanée d'une étape

Remarque : la durée de franchissement d'une transition n'est pas nulle, mais peut être rendue aussi petite que l'on veut.

5.7 Structures de base

5.7.1 Séquence unique

Une séquence unique est composée d'une suite d'étapes qui seront activées les unes après les autres.

Dans cette structure, chaque étape est suivie par une seule transition, et chaque transition est validée par une seule étape.

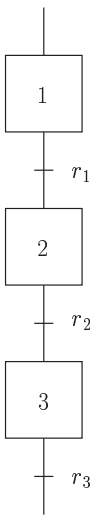


FIG. 5.38 – Structures de base : Séquence unique

5.7.2 Sélection de séquence

5.7.2.1 Début de sélection de séquence (divergence de sélection de séquence)

Une sélection de séquence est un choix d'évolution entre plusieurs séquences à partir d'une ou plusieurs étapes. Elle se représente graphiquement par autant de transitions validées en même temps qu'il peut y avoir d'évolutions possibles.

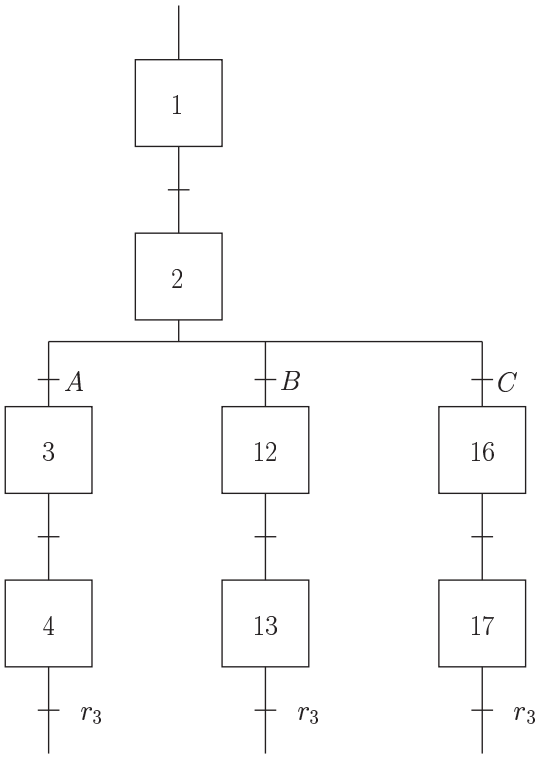


FIG. 5.39 – Structures de base : Sélection de séquences

D'après la structure proposée et les règles d'évolutions du grafcet, deux cas peuvent se présenter :

- ▷ dans tous les cas les séquences sont exclusives : on parle alors d'aiguillage (on ne peut évoluer que vers une seule séquence, comme un train ne va que vers une voie). Cette structure correspond à un OU exclusif, à partir d el'étape 2, on évolue vers 3 ou (exclusivement) vers 12 ou (exclusivement) vers 16.
- ▷ l'on peut partir de l'étape 2 évoluer vers deux (ou plus) séquences : on parle alors de parallélisme interprété (on évolue vers deux séquences en parallèle et ceci du aux réceptivités (deux ou plus réceptivités vraies) les réceptivités faisant parties de l'interprétation du grafcet, on utilise le terme parallélisme interprété). Cette structure correspond à un OU inclusif, à partir de l'étape 2 on évolue vers 2 ou 12, ou 16, c'est à dire que l'on peut évoluer vers $(3, 12)$ ou $(3, 16)$ ou $(3, 12, 16)$ par exemple suivant la valeur des réceptivités A, B, C.

La norme CEI, sans exclure le parallélisme interprété, ne le présente pas. Sous l'appellation sélection de séquence, elle sous entend sélection de séquences exclusives.

5.7.2.1.1 Séquences exclusives : aiguillage Il s'agit d'un cas particulier de sélection de séquence, où une seule séquence est sélectionnée. Dans ce cas, il est indispensable

que toutes les réceptivités associées aux transitions validées en même temps soient exclusives, c'est à dire ne pouvant être vraies simultanément.

Cette exclusion peut être :
▷ d'ordre physique (incompatibilité mécanique ou temporelle),
▷ d'ordre logique (dans l'écriture des réceptivités).
Le terme "aiguillage", souvent employé, illustre correctement cette structure.

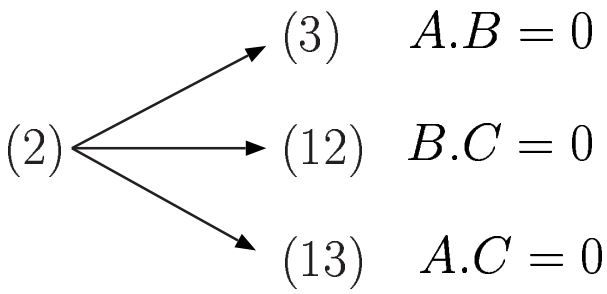


FIG. 5.40 – Structures de base : Aiguillage

5.7.2.1.2 parallélisme interprété (non normalisé) Le parallélisme interprété est un cas particulier des sélections de séquences dans lequel deux ou plusieurs séquences sont sélectionnées.

Les évolutions au sein de chaque séquence sélectionnée sont indépendantes.

Ce mode de fonctionnement doit être utilisé avec prudence car la plus grande difficulté réside dans la spécification correcte de la façon dont il se termine.

Les transitions susceptibles d'être simultanément franchies peuvent être repérées par des astérisques.

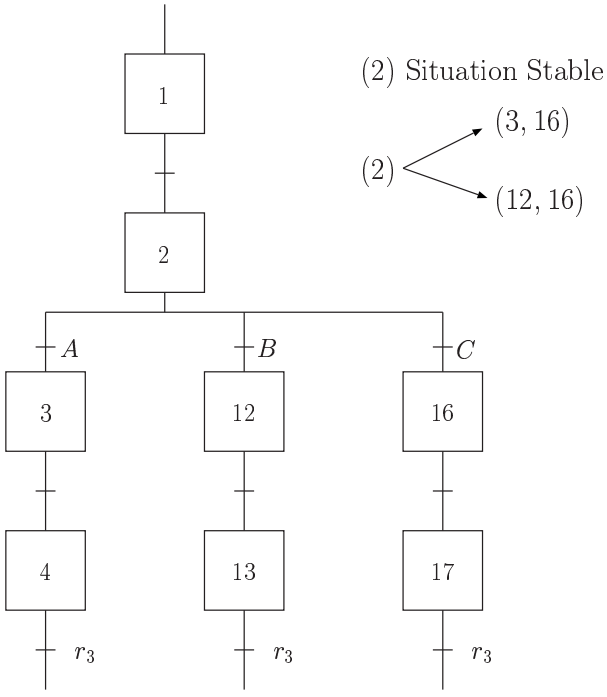


FIG. 5.41 – Structures de base : Parallélisme interprété

5.7.2.2 Fin de sélection de séquence (convergence de sélection de séquence)

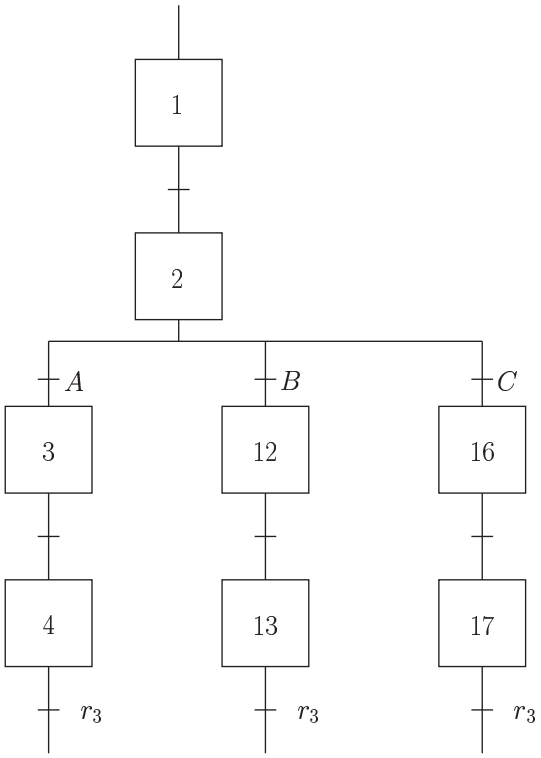


FIG. 5.42 – Structures de base : Fin de sélection de séquence

Paragraphe 4.6.2.2 de la norme CEI 848

5.7.2.3 Reprise de séquence (non normalisé)

La reprise de séquence est un cas particulier des séquences exclusives dans lequel une séquence peut être recommencée une ou plusieurs fois tant qu'une condition fixée n'est pas obtenue.

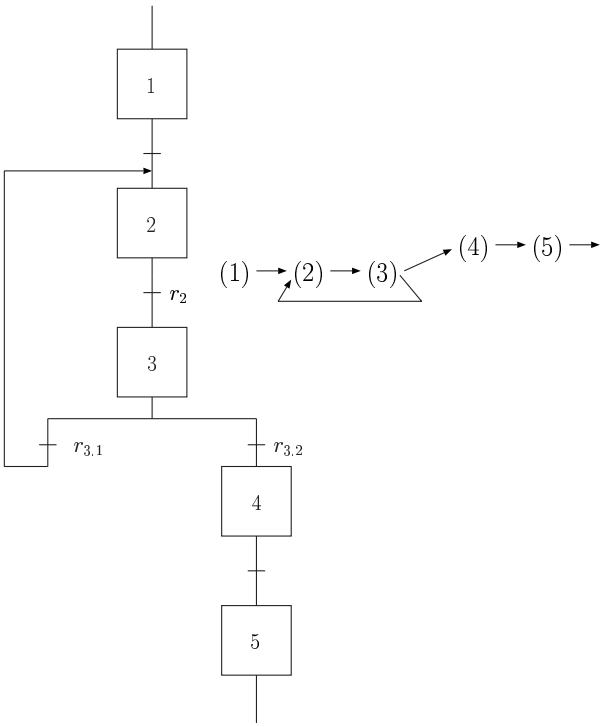


FIG. 5.43 – Structures de base : Reprise de séquence

5.7.2.4 Saut de séquence (non normalisé)

Le saut d'étape est un cas particulier des séquences exclusives dans lequel une séquence ne comporte pas d'étape. Cette séquence permet de sauter une ou plusieurs étapes.

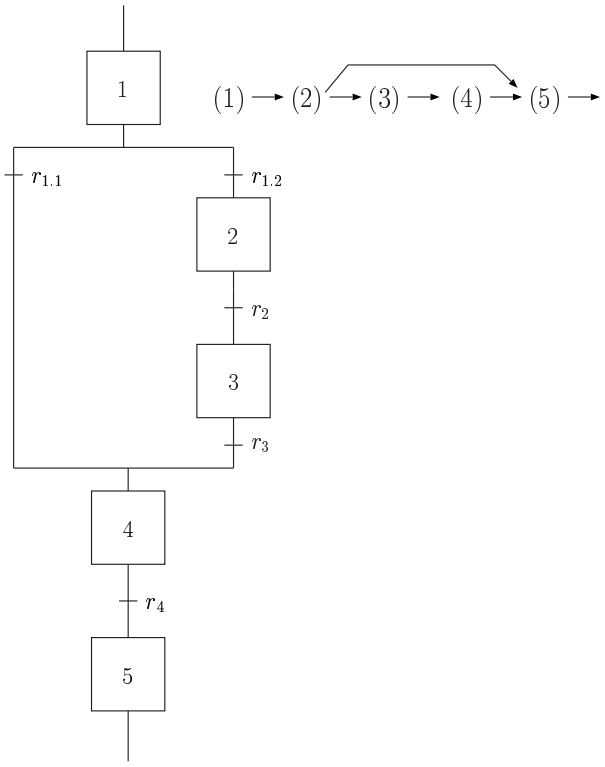


FIG. 5.44 – Structures de base : Saut de séquence

5.7.3 Séquences simultanées (mode parallèle)

5.7.3.1 Début de séquences simultanées (divergence de séquences simultanées)

On parle également de parallèlisme structural, car cette fois, le parallélisme est donné par la structure même du grafcet. De même que dans le parallélisme interprété, les évolutions au sein de chaque séquence sont indépendantes.

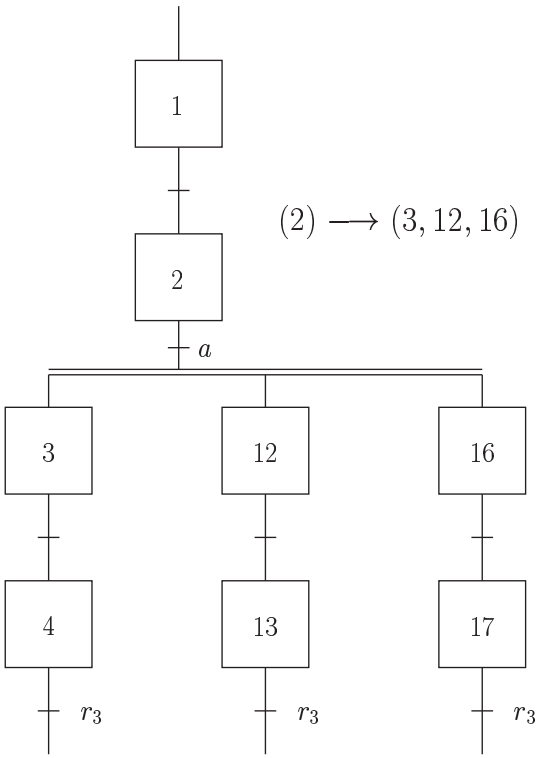


Fig. 5.45 – Structures de base : Séquences simultanées

5.7.3.2 Fin de séquences simultanées (convergence de séquences simultanées)

Lorsque le franchissement d'une transition conduit à activer simultanément plusieurs séquences (parallélisme structural), ces séquences évoluent de manière indépendante. Dans la plupart des cas il est nécessaire de les regrouper ensuite, en une ou plusieurs fois, vers une seule étape, pour en assurer la re-synchronisation. Pour réaliser cette condition il est prudent d'utiliser des étapes dites d'attente (suivant la, les technologies utilisées ces étapes peuvent être supprimées ou non).

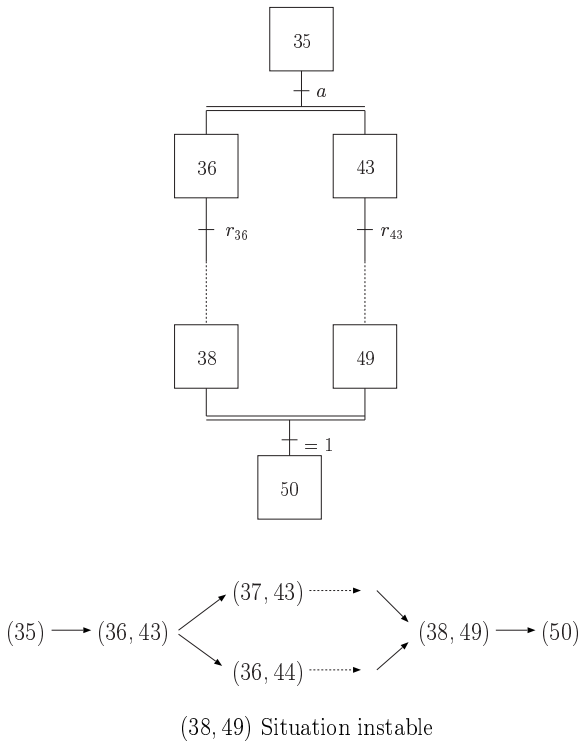


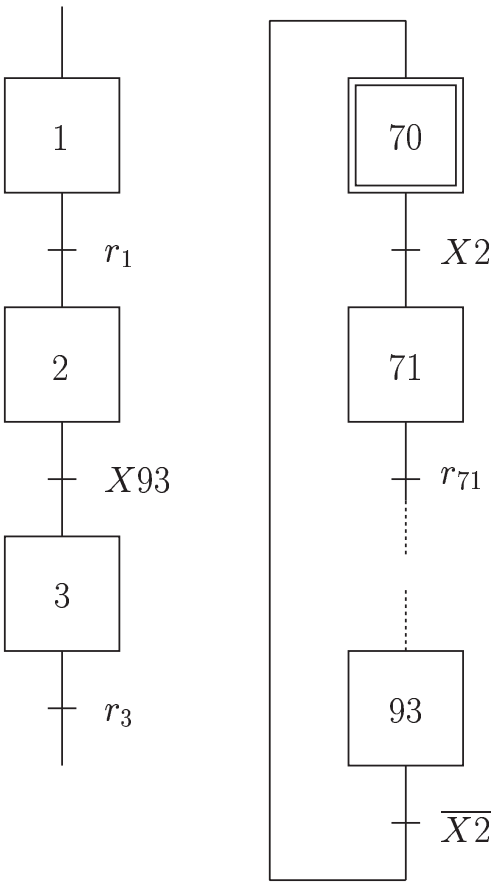
FIG. 5.46 – Structures de base : Séquences simultanées

5.7.4 Réutilisation d'une même séquence (sous-programme)

Cette structure très souvent employée, n'est pas présentée dans la norme CEI.

Lorsqu'une séquence est utilisée plusieurs fois, cette séquence peut être organisée sous une forme identique à celle d'un sous programme.

L'utilisation de la représentation en sous-programme demande une attention particulière pour le lancement et l'arrêt du sous-programme.



$$(1,70) \longrightarrow (2,70) \longrightarrow (2,71) \dashrightarrow$$
$$\dashrightarrow (2,93) \longrightarrow (3,93) \longrightarrow (3,70)$$

FIG. 5.47 – Structures de base : Réutilisation d'une même séquence

Cette page est laissée blanche intentionnellement

Chapitre 6

Grafcet : Mise en Œuvre

Il s'agit de définir précisément les éléments à prendre en compte dans la description afin de déterminer les entrées/sorties logiques et evènementielles qui permettront d'écrire les réceptivités et les action du grafcet. Il s'agit aussi de déterminer l'ensemble des ordres et comptes-rendus, c'est à dire l'ensemble des échanges entre la PC et la PO.

6.1 Mode de fonctionnement

Il est possible de décrire directement le fonctionnement complet d'un petit système, mais ceci devient inconcevable dès que le système se complique. Il est alors nécessaire d'adopter une démarche d'étude par segmentation du problème. Le GEMMA (Guide d'Ètude des Modes de Marche et d'Arrêt) fournit une méthode d'étude efficace en segmantant le problème en mode de fonctionnement et en les étudiant les uns après les autres. Dans un système automatisé, le GEMMA identifie les principaux modes de fonctionnements et l'ordre d'étude suivants :
▷ étude de la production normale,
▷ définition de l'arrêt en état initial,
▷ étude du lancement de production,
▷ étude de l'arrêt de production,
▷ étude des défaillances,
▷ étude de l'initialisation,
▷ étude d'autres modes : marches de vérification, de test...
Lors de la construction d'un grafcet, il s'agit de préciser quel mode de fonctionnement il décrit et de s'y tenir.

6.2 Construction de la structure

Suivant la complexités d'un système, différentes méthodes peuvent être utilisées.

6.2.1 Identification immédiate d'une structure grafcet

Sur des systèmes simples, on peut chercher à identifier parmi les strucures de base (séquence, auguillage, parallelisme, reprise de séquence...), laquelle convient ou si un assemblage judicieux de celles-ci permet de répondre au cahier des charges.

6.2.2 Analyse des comportements

Une liste des différents comportements de la P.O. étant dressé (sans doublon), l'un affecte une étape pour chacun d'entre eux puis l'on recherche pour chaque comportement quel doit être le comportement suivant. On peut donc construire le grafcet pas à pas.

Pour des problèmes simples, cette méthode permet de mettre en évidence les structures de base à utiliser.

6.2.3 Coordination des tâches

Lorsque le système est complexe, il est judicieux de le partitionner en sous-système, d'étudier chaque sous-système, puis de créer un grafcet de niveau supérieur chargé de synchroniser ces sous-systèmes. Cette méthode est décrite dans un paragraphe suivant.

6.3 Report des actions

Les variables de sorties déterminées précédemment sont alors reportées.

6.4 Report/détermination des réceptivités

Les réceptivités sont élaborées à partir des variables d'entrées et/ou des variables internes, ou bien elles sont déterminées en cherchant quelles sont les conditions nécessaires au passage d'un état à un autre.

6.5 État initial, situation initiale, lancement de production

Arrivé à ce stade de l'étude, il faut définir l'état initial du système : état dans lequel les systèmes est avant le passage en production normale. Cet état défini, permettra de préciser la situation initiale du grafcet décrivant la production normale. Pour lancer la production normale, l'expérience montre qu'il est plus performant de construire un grafcet de niveau supérieur (baptisé en général GC : grafcet de conduite) qui permetttra le lancement du grafcet de production normale (GN) et par le suite son arrêt.

6.6 Arrêt du système et retour à l'état initial

L'arrêt et le retour à l'état initial peuvent être délicat à étudier. En effet la demande d'arrêt peut arriver à tout moment du cycle, le cycle en cours doit se finir de telle manière que la partie opérative revienne à l'état initial, sur GPN cela se traduira par un ou plusieurs aiguillage de "sortie", dont certain peuvent comporter des séquences particulières afin de remettre la P.O. en état initial.

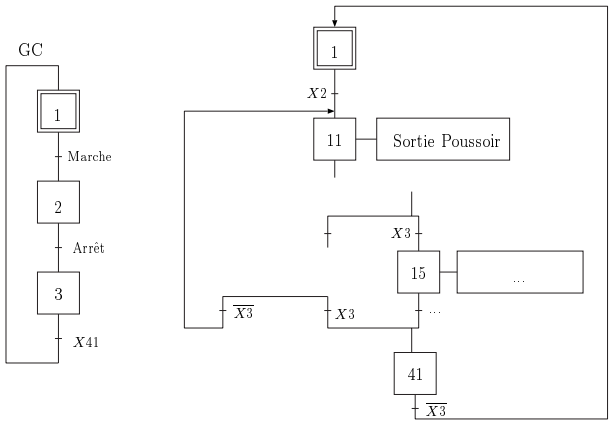


FIG. 6.1 – Construction de la structure : Ètat initial

6.7 Structure de coordination des tâches

6.7.1 Introduction

La complexité des systèmes automatisés nécessite lors de leur conception une approche méthodologique basée sur les démarches d'analyse descendante. Cette démarche commence par une description la plus générale possible du système, système décomposé en un minimum d'éléments. Parmi les différentes fonctions à assurer par un système automatisé de production, les fonctions opératives permettent l'obtention de la valeur ajoutée sur les produits. Les chois techniques, technologiques définissent un ensemble d'opérations fonctionnelles matérialisant ces fonctions. D'une démarche d'analyse descendante, une partition judicieuse de ces opérations en tâches permet une réduction des éléments d'étude et minimise ainsi la complexité de l'analyse.

6.7.2 Caractérisation d'une tâche

Une tâche est un regroupement fonctionnel d'un ensemble d'opérations pour lequel sont définis, vis à vis d'un critère de partition, un début ou lancement et une fin. Toute opération appartenant à une tâche est disjointe temporellement et/ou structurellement de toutes les opérations des autres tâches. Il n'existe aucune liaison entre une opération interne à une tâche et une opération interne à une autre tâche (autre que les liaisons définies par le lancment et la fin de tâche)

6.7.3 Représentation d'une tâche à l'aide du grafcet

La représentation d'une tâche en grafcet peut s'effectuer de différentes manières :
▷ Sous-programme, Normalisé
▷ tâche, non normalisé,
▷ Macro étape, normalisé
Actuellement le concept de macro étape est le plus répandu et le plus utilisé.

6.7.4 Méthodologie d'établissement d'un grafcet de coordination des tâches

1. Étape 1 : définition des différentes tâches et définitions des entrées/sorties,
2. Étape 2 : recherche des antériorités et des postériorités. Pour chaque tâche, on décrit les règles transitionnelles de début et de fin de tâche.
   ▷ Règle transitionnelle de début de tâche T_i, Si condition de début de tâche T_i réunies Alors faire tâche T_i
   ▷ Règle transitionnelle de fin de tâche T_i, Si condition de fin de tâche T_i réunies Alors autoriser tâche T_j
3. Étape 3 : élaboration de graphe associé à chaque tâche. À chaque tâche, l'on associe un graphe partiel construit de la manière suivante : La condition de lancement de la tâche T_i, exprimée par la fin de tâche T_j est représentée par une étape portant le numéro ji ou encore $j \rightarrow i$. L'état de fin d'exécution de la tâche T_i autorisant la tâche T_k va être représentée par une étape portant le numéro ik ou $i \rightarrow k$.
4. Étape 4 : élaboration du grafcet des coordination des tâches. Pour constituer le grafcet de coordination des tâches, l'on superpose les étapes de même repères des graphes associés aux différentes tâches. Si des étapes non fonctionnelles apparaissent, elles sont alors supprimées.