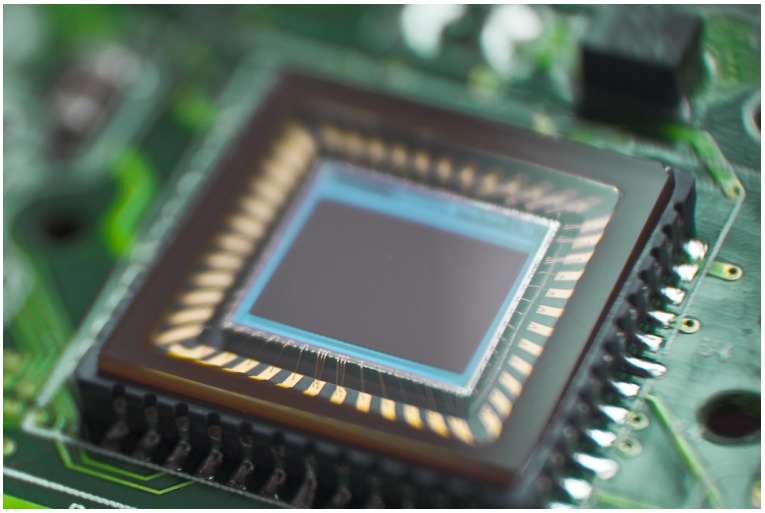
INSA
CENTRE VAL DE LOIRE

Cours d'électronique numérique

2ème Année



Enseignant référent : Julien BUSTILLO
julien.bustillo@insa-cvl.fr

Table des matières

0.1 Introduction et objectifs du cours

Il est fréquent en électronique d'opposer deux disciplines : le digital (ou numérique) à l'analogique. Le monde numérique constitue aujourd'hui une grande partie du monde que nous connaissons, telles que internet,les ordinateurs, les smartphones, etc... Le monde analogique est quant à lui de moins en moins présent, laissant la place au numérique quand cela est possible, mais reste encore présent dans les technologies radios (FM/AM) et dans les amplificateurs opérationnels.

Cependant le monde qui nous entoure est entièrement analogique et le numérique n'en est qu'une représentation. Cependant la simplicité d'utilisation du numérique dans les applications actuelles ont permis la croissance de ses parts de marché. Afin de passer du monde numérique au monde analogique, il est toujours nécessaire de passer par une étape interface qui est permise par les convertisseurs analogiques-numériques et numériques-analogiques.

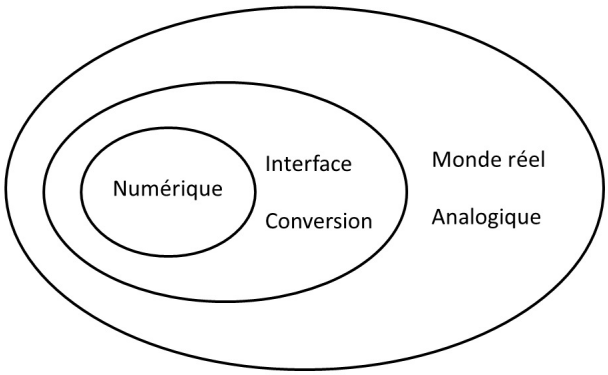


FIGURE 1 – Représentation des espaces analogiques et numériques ainsi que leurs interfaces

L'objectif de ce cours est l'étude des circuits logiques combinatoires, puis séquentiels. Combinatoire signifie que pour une réalisation donnée, chaque combinaison des entrées détermine une valeur bien définie de la sortie. Ainsi, ce type de montage peut toujours être représenté sous la forme d'un arbre et aucune boucle ne peut se former (cf. figure 2).

Séquentiel signifie au contraire que les valeurs de sortie ne dépendent pas uniquement de la valeurs des entrées mais aussi des états précédents du système. Ainsi, l'état de sortie dépendra de l'ordre d'arrivée des entrées. Dans ce cas là, des boucles peuvent être remarquées dans le circuit logique.

La première partie portera sur l'étude de circuits combinatoires et aura pour but la compréhension des systèmes combinatoires ainsi que leur synthèse.

La compréhension des-dits systèmes nécessite de savoir lire un schéma technologique, de mettre en équation les différentes variables ainsi que de connaître les différentes fonctions usuelles disponibles en circuits intégrés.

La synthèse nécessite les mêmes bases que la compréhension des circuits logiques

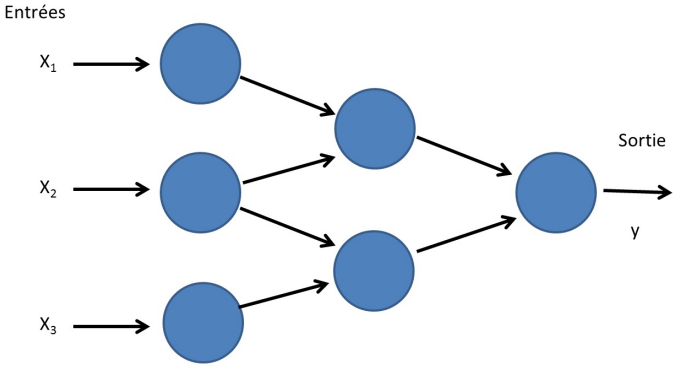


FIGURE 2 – Représentation sous la forme d'arbre d'une fonction logique

auxquelles s'ajoutent l'identification des variables nécessaires pour la réalisation et l'optimisation du système.

La seconde partie portera sur l'étude de circuits séquentiels et aura pour but la compréhension des circuits séquentiels et leurs réalisations.

Première partie

Logique combinatoire

Chapitre 1

Concept de Base

Les moyens théoriques disponibles pour l'étude des fonctions binaires résultent des travaux de Georges Boole (1854) repris et systématisés par Claude Shannon à partir de 1938. On parlera ainsi de variables Booléennes et d'algèbre de Boole

1.1 Variable Booléenne

On appelle variable booléenne (ou binaire) une variable susceptible de prendre uniquement deux valeurs représentant les deux états uniques de l'élément qu'elle représente. Voilà quelques exemples de variables booléennes :

1. Proposition : Vrai - Faux
2. Interrupteur : Ouvert - Fermé
3. Réponse : Oui - Non
4. Haut-Bas : H - L (High et Low en anglais)

En électronique les circuits traitant les variables binaires sont appelés circuits logiques. Les deux valeurs de la variable sont appelées "0" et "1". ce sont suivant les technologies des niveaux de tension (le plus courant) ou de courant, tel que $V_H > V_L$ et $I_H > I_L$. On définit alors deux logiques :

– V_H = "1" et V_L = "0" : logique positive
– V_H = "0" et V_L = "1" : logique négative.

Nous nous placerons,dans ce cours, toujours dans la première hypothèse (logique positive). D'après la définition d'une variable booléenne, nous avons :

– si A différent de 0, A = 1
– si A différent de 1, A = 0

Remarque : Pour chaque technologie, ce sont les règles d'interconnexion qui fixent les valeurs des niveaux haut et bas. Compte tenu de la dispersion sur les éléments constituants, le constructeur donne un intervalle de tension (ou de courant) pour le niveau bas et un intervalle de tension (ou de courant) pour le niveau haut. Cette souplesse sur les niveaux réels est un atout important des techniques digitales (ou numérique).

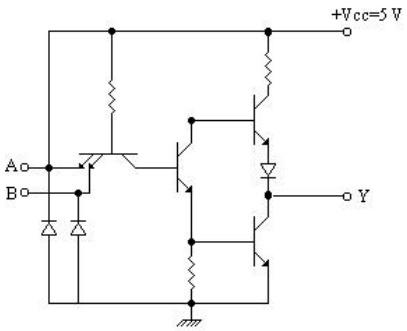


FIGURE 1.1 – Porte NAND TTL.

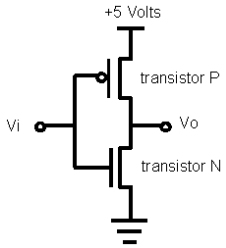


FIGURE 1.2 – Inverseur CMOS.

Remarque : Il n'y a pas forcément correspondance entre Vrai-Faux et "0"-"1". Il ne faut pas confondre (nous y reviendrons) l'action souhaitée et le moyen utilisé (le circuit).

1.2 Représentation des nombres

1.2.1 Virgule fixe-cas général

Définition

Tout nombre réel N peut être écrit de la façon suivante :

$$(N)_B = (E)_B.(F)_B \tag{1.1}$$

où E est la partie entière de N, F la partie décimale de N et B la base considérée. On peut aussi le mettre sous la forme d'un polynôme :

$$(N)_B = \sum_{-m}^{n} a_j B^j \tag{1.2}$$

où les a_j correspondent aux chiffre de la base (ou digits). Ce sont des éléments d'un ensemble qui en compte B (ou valeur de la base). La partie entière correspond aux exposants ayant une valeur nulle ou positive et la partie décimale correspond aux exposants négatifs.

Exemple de bases courantes :

– Décimal : $B = 10, a_j \in \{0, 1, ..., 9\}$

– Héxadécimal : $B = 16, a_j \in \{0, 1, ..., 9, A, B, C, D, E, F\}$
– Octal : $B = 8, a_j \in \{0, 1..., 7\}$
– Binaire : $B = 2, a_j \in \{0, 1\}$

Dans le cas du binaire, le digit est appelé le bit (Binary digIT).

Changement de base

Pour passer d'une base quelconque au décimal, on utilise directement la décomposition polynomiale, présentée précédemment, en remarquant que chaque digit est affecté d'un poids.

exemple :

$$195 = 1 * 10^2 + 9 * 10^1 + 5 * 10^0 \tag{1.3}$$

Dans le cas du binaire, il devient utile de connaitre les premières puissances de 2.

TABLE OF POWERS OF 2

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

FIGURE 1.3 – Puissance de 2

Exemple :

$$(0110110)_2 = 0 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 \quad (1.4)$$
$$= 2 + 4 + 16 + 32 = (54)_{10}$$

Pour passer du décimal à une base quelconque, on procède ainsi :
- on traite séparément la partie entière et la partie fractionnaire (ou décimale).
- pour la partie entière,on effectue des divisions successives par la valeur de la base en prenant les restes comme les valeurs des digits recherchés.
- pour la partie fractionnaire, on procède à des multiplications successives par la valeur de la base recherchée en gardant chaque partie entière résultante.

Exemples :

$$(195)_{10} = (C3)_{16} \quad (1.5)$$
$$= (11000011)_2$$

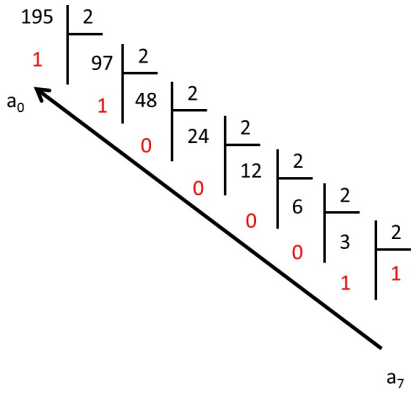


FIGURE 1.4 – Décomposition en binaire de $(195)_{10}$.

$$(0.37)_{10} = (0.0101...)_2 \quad (1.6)$$

1.2.2 Représentation des nombres négatifs par le complément à 2

En binaire pur, avec n chiffres, il n'y a que 2^n combinaisons possibles. Avec 8 bits, on peut ainsi représenter tous les entiers de 0 à $2^8 - 1 = 255$, avec 4 bits de 0 jusqu'à 15.

Soit un nombre binaire A composé de 4 bits :

$$A = a_3 a_2 a_1 a_0 \quad (1.7)$$

Définissons $\bar{A}$ tel que :

$$\bar{A} = \bar{a_3}\bar{a_2}\bar{a_1}\bar{a_0} \tag{1.8}$$

avec

$$\begin{aligned} \bar{a_n} &= 0 \text{ si } a_n = 1 \\ \bar{a_n} &= 1 \text{ si } a_n = 0 \end{aligned} \tag{1.9}$$

$\bar{A}$ s'appelle le complément à 1 de A.
Évaluons la la somme :

$$\begin{aligned} A + \bar{A} &= (a_3 + \bar{a_3})(a_2 + \bar{a_2})(a_1 + \bar{a_1})(a_0 + \bar{a_0}) \\ &= (1111)_2 = (15)_{10} \end{aligned} \tag{1.10}$$

Ajoutons 1 à cette somme :

$$\begin{aligned} A + \bar{A} + 1 = (10000)_2 &= (16)_{10} \text{ sur 5 bits} \\ &= (0)_{10} \text{ sur 4 bits} \end{aligned} \tag{1.11}$$

Nous avons donc trouvé un nombre $\bar{A} + 1$ qui, ajouté à A donne 0 à condition de tronquer le résultat à 4 bits. $\bar{A} + 1$ est la représentation choisie pour -A puisque $A + (-A) = 0$. Cette quantité s'appelle le complément à 2 de A.

Remarque : il s'agit en fait du complément à 2^n, le résultat étant généralisable quel que soit le nombre n de bits considéré.

Conséquences : Le bit de poids fort est le bit de signe (0 pour positif, 1 pour négatif).

1.2.3 Code BCD (Binary Coded Decimal)

Chaque chiffre décimal (unité, dizaine, centaine...), qui a une valeur comprise entre 0 et 9, est remplacé par son équivalent binaire sur 4 bits.

Exemple :

$$\begin{aligned} (247)_{10} &= (001001000111)_{BCD} \\ &= (247)_{16} \end{aligned} \tag{1.12}$$

On remarque que si ce système est simple, il a le gros inconvénient d'ignorer toutes les combinaisons de 4 bits comprises entre 10 et 15. Ainsi avec 8 bits, on ne peut représenter que les 100 nombres compris entre 0 et 99 et non les 256 premiers comme en binaire pur.

1.2.4 Code ASCII (American Code for Information Interchange)

C'est un code sur 7 bits (soit 128 possibilités) très utilisé par les systèmes informatiques. Il est encore appelé alphanumérique car il comporte le codage des lettres de l'alphabet, des nombres, puis des caractères de contrôle nécessaires à la transmission des messages (cf annexe).

1.3 Table de Vérité

Avant d'entamer la synthèse d'un problème combinatoire, il est souvent nécessaire de visualiser dans un tableau l'état de la sortie (ou des différentes sorties) pour toutes les combinaisons possibles des variables d'entrées (que l'on numérote suivant l'ordre binaire). Pour cela, il faut évidemment être capable d'identifier les variables d'entrée et de sortie.

Exemple 1 :

	Moyenne	note éliminatoire	Admis
0	F	F	F
1	F	V	F
2	V	F	V
3	V	V	F

Cette table visualise les deux équations suivantes :
- Admis SI moyenne ET PAS de note éliminatoire.
- Pas Admis Si note éliminatoire OU pas moyenne.

Exemple 2 :

$B = 0$ et $C = 0$ est impossible physiquement. $X = 0$ ou 1.

	A	B	C	S
0	0	0	0	X
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	X
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

La sortie est indifférente. Pour effectuer une réalisation de ce système, on pourra choisir l'état le plus avantageux. D'après cette table de vérité, on peut déduire que $S = 1$ SI $A = 1$ ET $C = 0$.

1.4 Opérateurs simples

Les exemples du paragraphe précédents on fait apparaître qu'une fonction booléenne peut se présenter sous la forme :

SI condition (sur les entrées) ALORS action (sur les sorties)

La condition sur les entrées fait appel suivant le cas aux conjonctions ET, OU et PAS.

Dans ce paragraphe, nous passons en revue les opérateurs simples et les circuits élémentaires qui leur correspondent.

1.4.1 Inverseur

A la variable A, on fait correspondre $\bar{A}$ telle que :

- $\bar{A} = 1$ si $A = 0$
- $\bar{A} = 0$ si $A = 1$

On a ainsi $\overline{\bar{A}} = A$

Schéma de la porte inverseuse :

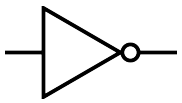


FIGURE 1.5 – Porte inverseuse.

1.4.2 Fonction ET

Cette fonction peut aussi être appelé produit logique ou intersection. S est actif SI A ET B sont actifs. On note $S = A.B$.

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

Le produit logique est :

- commutatif : $A.B = B.A$
- associatif : $(A.B).C = A.(B.C)$
- l'élément neutre est le 1 : $A.1 = 1.A = A$
- l'élément absorbant est le 0 : $A.0 = 0.A = 0$
- $A.\bar{A} = 0$

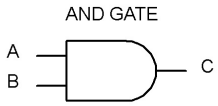


FIGURE 1.6 – Porte ET ou AND

Schéma de la porte ET (ou AND) :

En remarquant que $S = 0$ SI $A = 0$ OU $B = 0$, on peut obtenir une autre représentation basée sur des portes NOR.

1.4.3 Fonction OU inclusif

Cette fonction est encore appelée somme logique ou réunion.

Si est actif SI (A OU B) OU (A ET B) est actif. On note $S = A + B$.

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

La somme logique est :

– commutative : $A + B = B + A$

– associative : $(A + B) + C = A + (B + C)$

– l'élément neutre est le 0 : $A + 0 = 0 + A = A$

– l'élément absorbant est le 1 : $A + 1 = 1 + A = 1$

– $A + \bar{A} = 1$

Distributivité du produit par rapport à la somme :

$A.(B + C) = A.B + A.C$

Schéma de la porte OU (ou OR) :

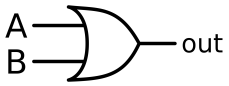


FIGURE 1.7 – Porte OU ou OR

En remarquant que $S = 0$ SI $A = 0$ OU $B = 0$, on peut obtenir une autre représentation à base de portes NAND.

1.4.4 Fonction NAND (NON-ET) et NOR (NON-OU)

En associant l'inverseur et les fonctions précédentes, on obtient les tables de vérité suivantes :

S = A NAND B

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

$$S = \overline{A.B} = A \uparrow B \tag{1.13}$$

S = A NOR B

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

$$S = \overline{A + B} = A \downarrow B \tag{1.14}$$

Schéma des portes NAND et NOR :

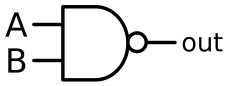


FIGURE 1.8 – Porte NAND

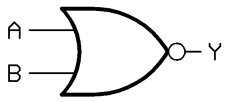


FIGURE 1.9 – Porte NOR

1.4.5 Fonction OU exclusif (XOR) et ET inclusif (EOR)

$S = 1$ SI A OU B est actif.

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

$$S = A \oplus B = \bar{A}.B + A.\bar{B} \tag{1.15}$$

De même, $S = 0$ (SI $A = 0$ ET $B = 0$) OU (SI $A = 1$ ET $B = 1$).

Donc

$$\bar{S} = \bar{A}.\bar{B} + A.B = A \odot B \tag{1.16}$$

Cette fonction est appelée fonction comparateur ou ET inclusif.
Schéma de la porte EOR :

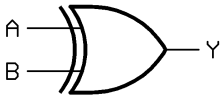


FIGURE 1.10 – Porte EOR ou XOR

1.5 Axiomatiques de l'algèbre de Boole

Les propriétés du ET et du OU permettent de démontrer les théorèmes suivant :
Théorème 1 : Absorption

$$A + A.B = A \text{ car } A + A.B = A.(1 + B) = A.1 = A \tag{1.17}$$

Théorème 2 : Implication

$$A.B = A \Longleftrightarrow A + B = B \tag{1.18}$$

Première implication :

$$B = B.1 = B.(1 + A) = B + B.A \tag{1.19}$$
$$\text{donc si } A.B = A, B = A + B$$

Deuxième implication :

$$A = A.(1 + B) = A + A.B = A.A + A.B = A(A + B) \tag{1.20}$$
$$\text{donc si } A + B = B, A = A.B$$

Théorème 3 : Adjacence

$$A.B + A.\bar{B} = A.(B + \bar{B}) = A \tag{1.21}$$

Théorème 4 : Consensus

$$A + \bar{A}.B = A.(B + \bar{B}) + \bar{A}.B = A.B + A.\bar{B} + \bar{A}.B \tag{1.22}$$
$$= B.(A + \bar{A}) + A.(B + \bar{B}) = A + B$$

Théorème 5 : Lois de De Morgan

Première loi :

"Le complément d'un produit est égal à la somme des compléments."

$$\overline{A.B.C....} = \bar{A} + \bar{B} + \bar{C}... \tag{1.23}$$

Deuxième loi :

"Le complément d'un somme est égal à le produit des compléments."

$$\overline{A + B + C + ...} = \bar{A}.\bar{B}.\bar{C}... \tag{1.24}$$

Ces deux lois ont déjà été vérifiées au niveau des tables de vérité des opérateurs simples. Cependant, on peut en donner une démonstration algébrique, ce qui constitue un bon exercice.

1.6 Formes canoniques et formes normales

Considérons une fonction de trois variables définie par sa table de vérité. Cette fonction est complète, c'est à dire qu'elle est définie par les $2^3 = 8$ combinaisons possibles des variables.

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

A partir d'une proposition de la forme : "F est vérifiée SI...ET...OU ", on peut écrire directement :

$$F = \bar{A}.\bar{B}.C + \bar{A}.B.C + A.\bar{B}.C \tag{1.25}$$

On peut ainsi exprimer une fonction sous la forme d'une forme de somme de produits. C'est la première forme canonique. Chaque terme produit de cette somme contient explicitement toutes les variables (complémentées ou non). On les appelle Minterm (le minterm est une fonction définie avec un seul "1"). Si on numérote les minterm en pondérant de manière arbitraire chaque variable, il est alors commode de définir la fonction avec la notation suivante :

$$F = R(1, 3, 5) \tag{1.26}$$

où R est la réunion. 1, 3, 5 représentent les minterm $m_1 = \bar{A}\bar{B}C$, $m_3 = \bar{A}BC$ et $m_5 = A\bar{B}C$.

Remarque : pour une fonction complète, comme cet exemple,

$$\bar{F} = R(\text{ autres minterm }) = R(0, 2, 4, 6, 7) \tag{1.27}$$

on a alors :

$$\bar{F} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC \tag{1.28}$$

En appliquant De Morgan à cette expression, nous avons alors :

$$F = \bar{\bar{F}} = (A + B + C).(A + \bar{B} + C).(\bar{A} + B + C).(\bar{A} + \bar{B} + C).(\bar{A} + \bar{B} + \bar{C}) \tag{1.29}$$

F est alors sous la forme de produit de somme. C'est la deuxième forme canonique. Chaque terme est dénommé Maxterm car il représente une fonction qui n'aurait qu'un seul "0".

Une représentation commode de la deuxième forme canonique qui suppose une numérotation des minterms est :

$$F = I(0, 2, 4, 6, 7) \tag{1.30}$$

où I est l'intersection.

Remarque : puisque la fonction est complète, on peut écrire à l'évidence que :

$$\bar{F} = I(1, 3, 5) \tag{1.31}$$

En utilisant les règles de l'algèbre booléenne, on peut transformer l'expression de F afin de réduire l'écriture de cette fonction :

$$\begin{aligned} F &= \bar{A}.\bar{B}.C + \bar{A}.B.C + A.\bar{B}.C = \bar{A}C(\bar{B} + B) + \bar{B}C(\bar{A} + A) \\ &= \bar{A}C + \bar{B}C \end{aligned} \tag{1.32}$$

Cette forme réduite est une somme de produit et est appelée première forme normale de F. On peut aussi simplifier la première forme canonique de $\bar{F}$. On obtient alors :

$$\bar{F} = AB + \bar{C} \Rightarrow F = C(\bar{A} + \bar{B}) \tag{1.33}$$

Cette écriture sous la forme de produit de sommes est appelée deuxième forme normale de F.

1.7 Analyse et synthèse - schémas technologiques

Il n'y a pas lieu d'opposer analyse et synthèse mais plutôt d'y voir des démarches complémentaires. Si l'on considère, en effet, un système particulier, il aura été mis au point par l'ingénieur de conception, puis il passera au stade de la fabrication, complété par le test de fonctionnement. Plus tard, il faudra procéder à des interventions de dépannages.

Les différentes personnes intervenant après la fabrication feront en réalité de l'analyse. Le document commun aux différents intervenants est le schéma technologique du système (et non les équations logiques). Le responsable de la conception qui doit l'établir doit tenir compte tout au long de son travail d'un certain nombre de principes (éventuellement codifiés par une norme).

Les différentes étapes de conception sont les suivantes :

1. identification des variables binaires et choix des mnémoniques (ou abréviations) appropriés.
2. établissement des tables de vérité et choix des niveaux actifs.
3. simplification des équations logiques.
4. Passage au schéma technologique avec optimisation.

Mnémoniques polarisés

Le choix du mnémonique suggère l'action souhaitée. La variable sera complémentée (ou L) ou non (H) suivant que son niveau actif est haut ou bas.

Exemples :

– $\overline{RAZ}$: remise à zéro, actif au niveau bas
– INIT : initialisation, actif au niveau haut
– LOAD(L) : chargement, actif au niveau bas

Symboles des portes

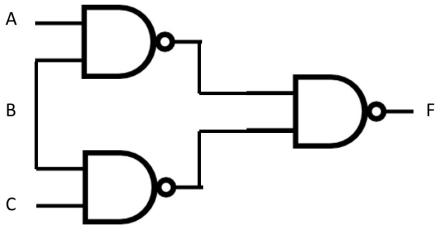


FIGURE 1.11 – Schéma technologique 1

Si nous comparons les deux schémas des figures 11 et 12 qui sont strictement équivalent du point de vue équation logique, nous constatons que seul le deuxième permet de reconnaitre instantanément la fonction F réalisée :

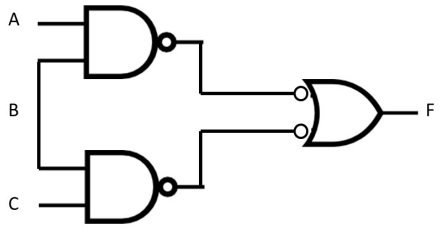


FIGURE 1.12 – Schéma technologique 2

$$F(H) = (AB + BC)(H) \tag{1.34}$$

Les niveau actif (H) sont aussi connus sans l'ambiguïté dans le premier schéma en ce qui concerne la sortie. Sauf impossibilité, une variable active au niveau bas est connectée à une entrée précédée d'un rond inverseur, et réciproquement.

Exemple de construction d'un schéma logique :

Soit à réaliser la synthèse de la fonction $\bar{F} = A\bar{B} + \bar{A}C$. Les entrées disponibles sont A,B et C (actives au niveau haut). La sortie $\bar{F}$ est active au niveau bas.

Premier schéma :

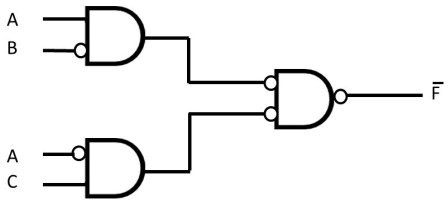


FIGURE 1.13 – Schéma technologique 1

On représente le plus simplement l'équation OU, ET en prenant pour la sortie un schéma avec "bulle" pour souligner qu'il s'agit d'une sortie active au niveau bas.

Deuxième schéma :

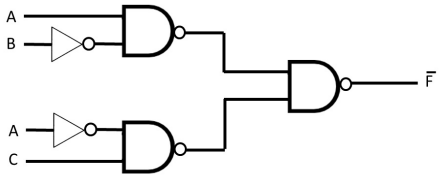


FIGURE 1.14 – Schéma technologique 2

Pour uniformiser le type de portes, on fait glisser les bulles intermédiaires et on place des inverseurs sur les entrées qui en nécessite.

Chapitre 2

Simplification des fonctions combinatoires

Dans le chapitre précédent, nous avons abordé les trois étapes fondamentales de la synthèse d'un problème combinatoire :

1. établissement d'une table de vérité. La fonction est reconnue combinatoire complète ou incomplète.
2. simplification de la fonction.
3. établissement d'un schéma technologique approprié.

La difficulté de l'optimisation de la réalisation est due à ce que pour une fonction de logique donnée, il y a un grand nombre de solutions possibles. Dans le cas de fonctions incomplètes, le nombre de solutions différentes envisageables constitue un obstacle de plus car de nombreux degrés de liberté sont ajoutés. Aussi, a-t-on besoin de quelques outils complémentaires pour effectuer la synthèse avec optimisation sans trop de difficulté.

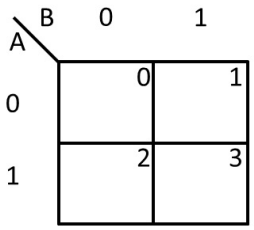
Dans ce chapitre, nous présentons deux méthodes de simplifications puissantes. La table de Karnaugh a l'avantage de visualiser les simplifications possibles et permet ainsi d'effectuer des choix décisifs pour l'obtention d'une bonne réalisation. La méthode des consensus est une méthode systématique qui s'applique bien à des fonctions à grand nombre de variables et peu de minterms.

Au niveau de l'optimisation des réalisations, nous prendrons comme critère (qui est un choix, mais qui pourrait être différent selon les besoins) le nombre de boitiers utilisés.

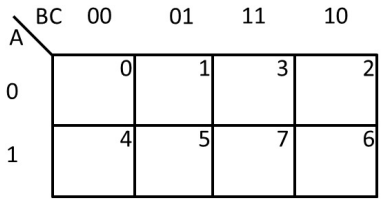
2.1 Simplification par table de Karnaugh

2.1.1 Construction de la table

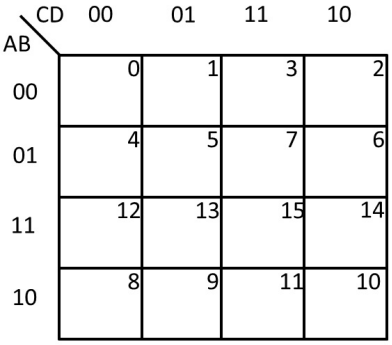
La table de Karnaugh (1953) est une forme particulière de la table de vérité. La succession des minterms y est ordonnée de façon à ce que l'on passe de l'un à l'autre par changement d'une seule variable. On dit que les minterms sont adjacents.



Minterms	Adjacence
0	1,2
1	0,3
2	0,3
3	1,2



Minterms	Adjacence
0	1,2,4
1	0,3,5
2	0,3,6
3	1,2,7
4	0,5,6
5	1,4,7
6	2,4,7
7	3,5,6



Minterms	Adjacence
0	1,2,4,8
1	0,3,5,9
2	0,3,6,10
3	1,2,7,11
4	0,5,6,12
5	1,4,7,13
6	2,4,7,14
7	3,5,6,15
8	0,9,10,12
9	1,8,11,13
10	2,8,11,14
11	3,9,10,15
12	4,8,13,14
13	5,9,12,15
14	6,10,12,15
15	7,11,13,14

Remarque : Les bords extrêmes sont adjacents en horizontale et en verticale. Pour un nombre de variables supérieur à 4, la représentation est moins évidente.

2.1.2 Utilisation de la table

Toutes les simplifications qui sont mises en évidence par la table sont des applications à plusieurs niveaux du théorème d'adjacence $A(B + \bar{B}) = A$.

Les différents cas de figure sont présentés ici sur des exemples.

Exemple à 2 variables : $F(A, B) = R(2, 3)$

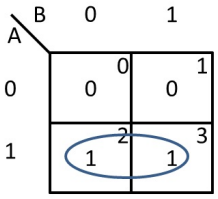


FIGURE 2.1 – Exemple avec 2 variables

$$F = A\bar{B} + AB = A \tag{2.1}$$

Le regroupement des deux minterms adjacents élimine la variable changeante, qui est B dans cet exemple.

Exemple à 3 variables : $F(A, B, C) = R(1, 3, 5)$

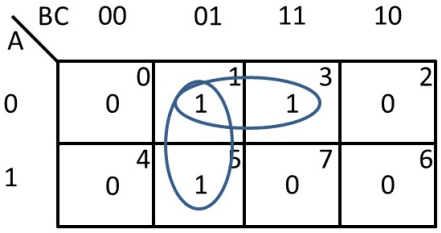


FIGURE 2.2 – Exemple avec 3 variables

Les minterms 1 et 3 sont adjacents. La fonction s'écrit donc partiellement $\bar{A}(\bar{B}C + BC) = \bar{A}C$. La variable B s'est éliminée dans ce regroupement. De même le regroupement des minterms 1 et 5 donne $\bar{B}C$. D'où $F = \bar{A}C + \bar{B}C$

Remarque : On peut, bien sur, obtenir la deuxième forme normale en considérant les regroupements des "0".

$$F = C(\bar{A} + \bar{B}) \tag{2.2}$$

C est obtenue en regroupant les minterms 0, 2, 4, 6. $\bar{A} + \bar{B}$ est obtenu par regroupement de 7 et 6.

Exemple à 4 variables : $F = R(3, 7, 8, 9, 10, 11, 12, 13, 15)$

on peut alors procéder aux regroupement suivant :

- $3, 7 \Longrightarrow AB\bar{D}$
- $15, 11 \Longrightarrow ABD$

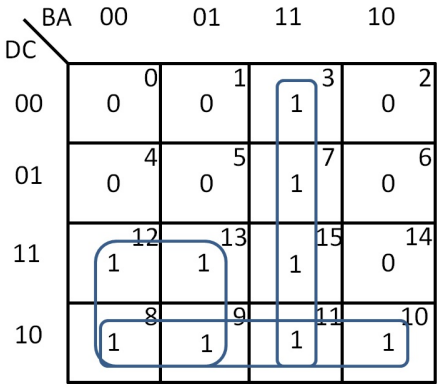


FIGURE 2.3 – Exemple avec 4 variables

– 3, 7, 15, 11 $\Longrightarrow AB$ (élimination de 2 variables par regroupement de 4 cases)
– 8, 9, 10, 11 $\Longrightarrow D\bar{C}$
– 8, 9, 12, 13 $\Longrightarrow D\bar{B}$

Finalement puisqu'on a couvert tous les minterms, on obtient la forme simplifiée de F, qui est la première forme normale :

$$F = AB + C\bar{D} + \bar{B}D \tag{2.3}$$

En recherchant à regrouper les "0", on obtient la deuxième forme normale de F :

$$F = (B + D).(A + D).(A + \bar{B} + \bar{C}) \tag{2.4}$$

2.1.3 Exposé de la méthode de lecture

1. Considérer d'abord les "1" isolés qui ne peuvent pas intervenir dans une simplification. Ce sont des implicants essentiels.

   Exemple :

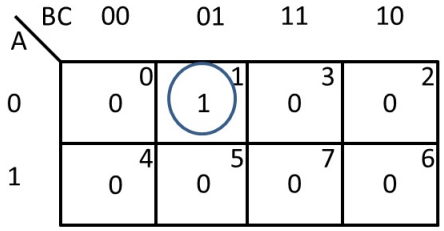


FIGURE 2.4 – "1" isolés

2. Chercher les minterms qui ne rentrent que dans la simplification d'une seule variable (groupe de de "1" adjacents). Entourer ces nouveaux implicants et recommencer.

   Exemple :

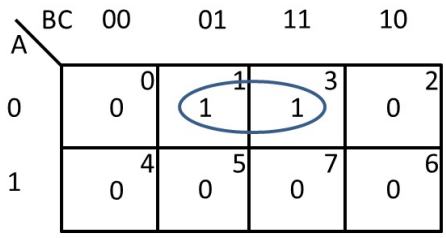


FIGURE 2.5 – regroupement de 2 variables

3. Chercher les minterms qui ne rentrent que dans la simplification de 2 variables (groupe de 4 "1" adjacents).

   Exemples :

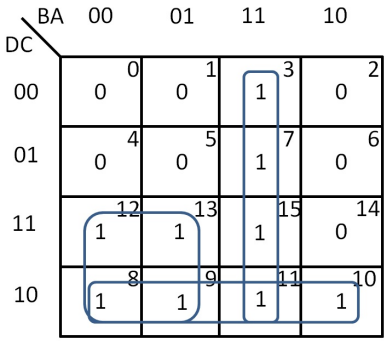


FIGURE 2.6 – Exemple avec 4 variables

4. recouvrir les minterms qui ne rentrent dans la simplification de 3 variables (groupe de 8 "1" adjacents).

   Exemples :

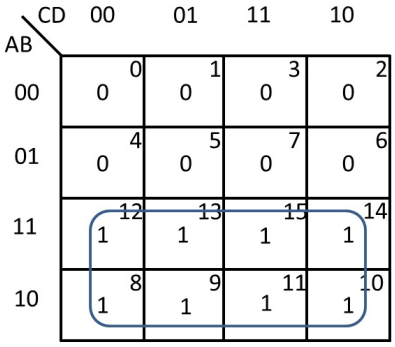


FIGURE 2.7 – Exemple avec 4 variables

2.1.4 Table de Karnaugh à 5 et 6 variables

Pour les fonctions de plus de 4 variables, la table de Karnaugh est d'un emploi moins souple car il faut considérer des symétries pour définir l'adjacence des minterms.

ABC \ DEF	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

FIGURE 2.8 – Table de Karnaugh à 6 variables

Ainsi dans la table à 6 variables présentée ci-dessus, 27 est adjacent à 11, 19,25 et 26 mais aussi à 31 (symétrie par rapport à l'axe vertical) et à 59 (symétrie par rapport à l'axe horizontal).

La méthode est toujours la même, on considère des implicants comportant 1, 2, 4, 8, 16... minterms.

2.1.5 Exemple d'une fonction incomplète de variables

3 états logiques sont alors présents : "0", "1" et ϕ qui peuvent indifféremment être un "1" ou un "0", soit :

$$F(A,B,C,D,E) = \tag{2.5}$$
$$R(0,2,3,4,8,11,18,19,20,25,27,28,29,\phi(5,6,9,10,12,14,17,26,30))$$

1. pas de "1" isolé
2. implicant à 2 termes : 25, 29 $\Longrightarrow A B \bar{D} E$
3. implicant à 4 termes : 4, 12, 28, 20 $\Longrightarrow C \bar{D} \bar{E}$
4. implicants à 8 termes :

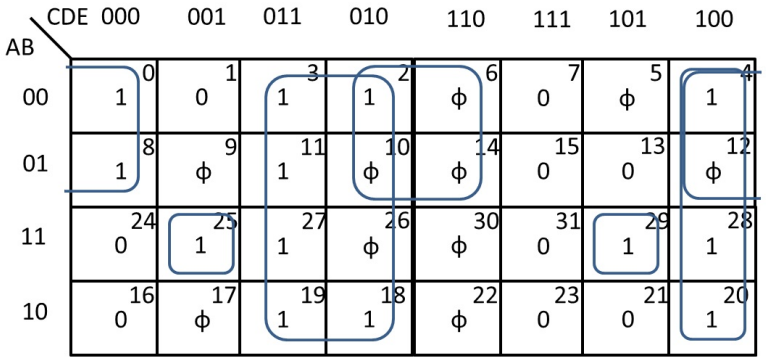


FIGURE 2.9 – Exemple d'une fonction incomplète de variables

– $3, 2, 11, 10, 27, 26, 19, 18 \Longrightarrow \bar{C}D$
– $0, 8, 4, 12, 2, 10, 6, 14 \Longrightarrow \bar{A}\bar{E}$

D'où $F = \bar{C}D + \bar{A}\bar{E} + C\bar{D}\bar{E} + AB\bar{D}E$

Remarque : Au niveau du circuit qui réalisera cette fonction, il n'y a plus d'états ϕ. Les ϕ recouverts donneront des "1", les autres donneront des "0".

2.2 Simplification par la méthode des consensus

2.2.1 Définition

Toute fonction combinatoire $F = f(A, B, C, D)$ peut se mettre sous la forme $F = A.G(B, C, D) + \bar{A}.H(B, C, D)$.

– G est le résidu de F par rapport à A.
– H est le résidu de F par rapport à $\bar{A}$.
– G.H est le consensus de F par rapport à A.

2.2.2 Théorème

Une fonction combinatoire reste identique à elle-même si on la réunit au consensus par rapport à l'une de ses variables.

$F = A.G(B, C, D) + \bar{A}.H(B, C, D) + G.H$

Exemple simple : $F = A + \bar{A}B$

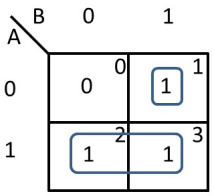


FIGURE 2.10 – Démonstration de l'adjacence sur le tableau de Karnaugh

Le consensus de F par rapport à A est B. Donc $F = A + \bar{A}B + B = A + B(1 + \bar{A}) = A + B$

La table de Karnaugh de cet exemple montre que la simplification provient de l'adjacence du terme AB contenu dans A avec le terme $\bar{A}B$.

On aurait pu en effet écrire :

$$F = A + \bar{A}B = A + A(B + \bar{B}) + \bar{A}B = A + AB + \bar{A}B + A\bar{B} \tag{2.6}$$
$$= A(\bar{B} + 1) + B(\bar{A} + A) = A + B$$

Remarquons que la réciproque est vraie : $F = A + \bar{A}B + B = A + \bar{A}B$. On peut donc supprimer le consensus.

2.2.3 Discussion de la méthode

Pour simplifier une fonction, on fait apparaitre tous les consensus possibles et on regarde s'ils simplifient ou non la fonction. On ne les garde que dans le cas affirmatif. En fin de simplification, on cherche de même si un des termes n'est pas lui même un consensus, auquel cas il peut lui-même disparaître.

L'avantage de cette méthode est d'être systématique. Elle peut s'appliquer à des fonctions incomplètes. Cependant, dans ce dernier cas, il faudra être vigilant et placer entre parenthèses les consensus optionnels provenant des ϕ.

Exemple : $F = ABDE + AB\bar{E} + AC\bar{D} + B\bar{C}\bar{D} + \bar{B}\bar{D}$

Variables	Résidus réduits	Consensus réduits	Fonction en cours de simplification
A	$BDE + B\bar{E} + C\bar{D}$	0	F
$\bar{A}$	0		
B	$ADE + \bar{C}\bar{D} + A\bar{E}$	$\bar{C}\bar{D} + A\bar{D}\bar{E}$	$F = ABDE + AB\bar{E} + \bar{C}\bar{D} + \bar{B}\bar{D} + AC\bar{D}$
$\bar{B}$	$\bar{D}$		
C	$A\bar{D}$	$A\bar{D}$	$F = ABDE + AB\bar{E} + \bar{C}\bar{D} + \bar{B}\bar{D} + A\bar{D}$
$\bar{C}$	$\bar{D}$		
D	ABE	$ABE + A\bar{C}BE$	$F = AB + A\bar{D} + \bar{C}\bar{D} + \bar{B}\bar{D}$
$\bar{D}$	$A + \bar{C} + \bar{B}$		
B	A	$A\bar{D}$	$F = AB + \bar{C}\bar{D} + \bar{B}\bar{D}$
$\bar{B}$	$\bar{D}$		

2.3 Simplification des fonctions à sorties multiples

Lorsqu'il est nécessaire de réaliser des fonctions distinctes des mêmes variables d'entrée, un principe d'optimisation est la mise en commun du maximum d'implicants.

Pour parvenir à ce but, en plus des tables de Karnaugh de chaque fonction, on constitue une table des implicants. Cela permet de choisir pour plusieurs fonctions des regroupements identiques.

Exemple :

- $F_1(A,B,C) = R(0,2,3,5,6)$
- $F_2(A,B,C) = R(1,2,3,4,7)$
- $F_3(A,B,C) = R(2,3,4,5,6)$

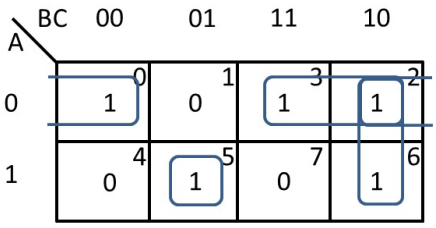


FIGURE 2.11 – Tableau de Karnaugh de F_1

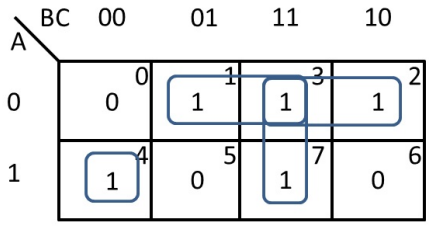


FIGURE 2.12 – Tableau de Karnaugh de F_2

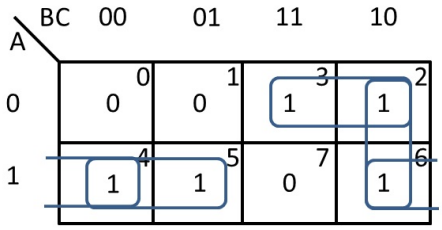


FIGURE 2.13 – Tableau de Karnaugh de F_3

Minterm	F_1	F_2	F_3
m_0	$\bar{A}\bar{B}\bar{C}/\bar{A}\bar{C}$		
m_1		$\bar{A}\bar{B}C/\bar{A}C$	
m_2	$\bar{A}B\bar{C}/\bar{A}\bar{C}/\bar{A}B/B\bar{C}$	$\bar{A}B\bar{C}/\bar{A}B$	$\bar{A}B\bar{C}/\bar{A}B/B\bar{C}$
m_3	$\bar{A}BC/\bar{A}B$	$\bar{A}BC/\bar{A}B/\bar{A}C/BC$	$\bar{A}BC/\bar{A}B$
m_4		$A\bar{B}\bar{C}$	$A\bar{B}\bar{C}/A\bar{C}/A\bar{B}$
m_5	$A\bar{B}C$		$A\bar{B}C/A\bar{B}$
m_6	$AB\bar{C}/B\bar{C}$		$AB\bar{C}/B\bar{C}/A\bar{C}$
m_7		ABC/BC	

Pour construire progressivement les expressions de F_1, F_2 et de F_3, on commence par examiner les lignes ne contenant qu'une seule colonne remplie, puis celles à 2 colonnes, etc... Cela donne sur cet exemple :

- 1 colonne :
  - $m_0 : F1 = \bar{A}\bar{C} + ...$
  - $m_1 : F2 = \bar{A}C + ...$
  - $m_7 : F2 = BC + ...$
- 2 colonnes :
  - $m_4 : F2 = A\bar{B}\bar{C} + ...$ et $F3 = A\bar{B}\bar{C} + ...$
  - $m_5 : F1 = A\bar{B}C + ...$ et $F3 = A\bar{B}C + ...$
  - $m_6 : F1 = B\bar{C} + ...$ et $F3 = B\bar{C} + ...$
- 3 colonnes :
  - $m_2 : F1 = \bar{A}B + ...$, $F2 = \bar{A}B + ...$ et $F3 = \bar{A}B + ...$
  - m_3 n'apporte aucune modification

Les fonctions optimisées sont donc :

- $F_1 = \bar{A}\bar{C} + A\bar{B}C + B\bar{C} + \bar{A}B$
- $F_2 = \bar{A}\bar{C} + A\bar{B}\bar{C} + \bar{A}B + BC$
- $F_3 = A\bar{B}\bar{C} + A\bar{B}C + B\bar{C} + \bar{A}B$

2.4 Réalisation à partir d'opérateurs simples

Le problème de l'établissement des schémas électroniques a déjà été abordé précédemment. Une réalisation de type OU de ET peut être déduite immédiatement de la première forme normale de la fonction. De même, une réalisation de type ET de OU traduit la deuxième forme normale de la fonction.

La technologie des circuits intégrés permet de réaliser plus facilement des portes NAND ou NOR que des portes ET et OU. De plus, ces opérateurs contiennent implicitement l'inverseur. Pour minimiser le nombre de boitiers, on cherche donc le plus souvent à utiliser ce type de portes logiques. Il convient de souligner cependant qu'il n'existe pas de méthode conduisant à LA solution unique optimum et qu'un même problème peut avoir un grand nombre de solutions présentant des avantages comparables. L'expérience de l'ingénieur en matière de synthèse est un élément primordial.

Dans ce paragraphe, nous présentons quelques cas fréquemment rencontrés et leurs solutions.

2.4.1 Somme de produits

$$F = AB + CD = \overline{\overline{AB}.\overline{CD}} \tag{2.7}$$

La réalisation de ces circuit s'effectue à l'aide de portes NAND.

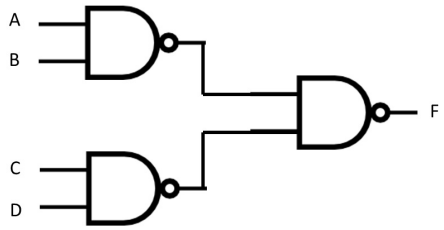


FIGURE 2.14 – Somme de Produits avec des portes NAND

2.4.2 Produit de sommes

$$F = (A + B).(C + D) = \overline{\overline{A + B} + \overline{C + D}} \tag{2.8}$$

La réalisation de ces circuit s'effectue à l'aide de portes NOR.

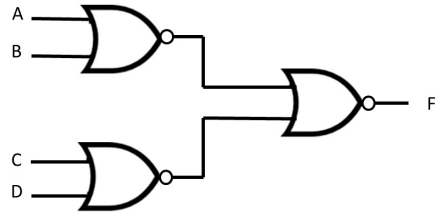


FIGURE 2.15 – Produit de sommes avec des portes NOR

2.4.3 Economie de l'inverseur

1/ par porte NAND

$$F = A.\bar{B} = A.(\bar{A} + \bar{B}) = A.\overline{A.B} \tag{2.9}$$

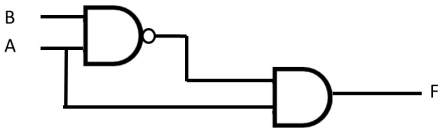


FIGURE 2.16 – Inverseur avec une porte NAND

Avantage : une seule porte NAND au lieu de 2 inverseurs dans le cas de la réalisation des deux fonctions.

Exemple : Création d'une fonction OU exclusif avec des portes NAND : (voir Fig. 2.17)

$$F = A + \bar{B} = A + \bar{A}.\bar{B} = A + \overline{A + B} \tag{2.10}$$

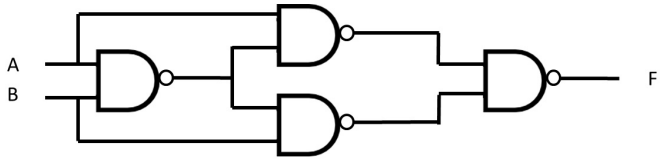


FIGURE 2.17 – Fonction XOR avec des portes NAND

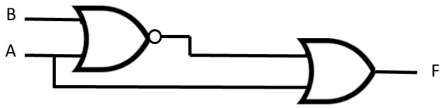


FIGURE 2.18 – Inverseur avec une porte NOR

2.5 Aléas de continuité

Lorsqu'un circuit a été synthétisé, il est absolument nécessaire de vérifier son fonctionnement dynamique. En effet, chaque porte logique étant affectée d'un retard de propagation, il peut en résulter l'apparition d'impulsions parasites en sortie. Les redondances introduites au niveau du recouvrement des minterms permettent d'éliminer ce genre d'aléas (en fait on rajoute le consensus).

Exemple 1 : considérons la fonction $F = A\overline{A} = 0$ réalisée avec un inverseur et une porte ET

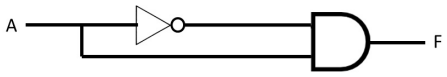


FIGURE 2.19 – Fonction F avec aléa

Supposons à $t = 0$, $A = 0$, $\overline{A} = 1$, $F = 0$.

Soit θ le retard de propagation de chaque porte, que l'on estimera identique.

A $t = 0$, A passe à 1, à $t = \theta$, $\overline{A} = 0$ et $F = 1$ et à t = 2 θ, F=0.

F reste donc à 1 pendant une durée θ. On a détecté un aléa. La solution pour le supprimer est de rajouter le terme $\overline{\overline{A}}$ et donc un inverseur.

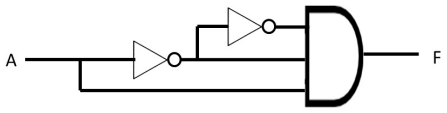


FIGURE 2.20 – Fonction F sans aléa

Exemple 2 : $F = AC + B\bar{C}$

Si $A = 1, B = 1, F = 1$ quelque que soit C. Ce cas est alors le même que dans l'exemple précédent. On rajoute donc le consensus afin d'éviter les aléas.

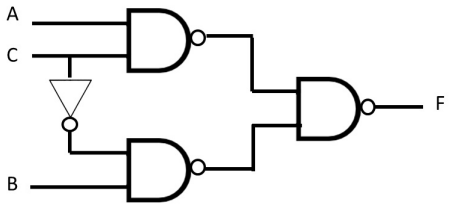


FIGURE 2.21 – Fonction F avec aléa

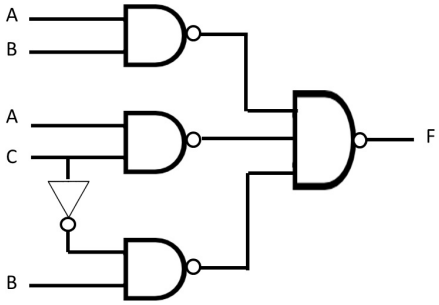


FIGURE 2.22 – Fonction F sans aléa

Chapitre 3

Circuits combinatoires complexes

Depuis vingt ans, les progrès de la technologie ont enrichi le catalogue des fabricants tant sur le plan des performances électriques que de la diversité et de la complexité des fonctions réalisées. On est passé des circuits à faibles densités d'intégration (SSI) comportant une dizaine de portes au maximum, au circuits à moyenne puis forte densité d'intégration (LSI et VLSI), pouvant comporter jusqu'à plusieurs milliards de portes. Il est donc nécessaire à l'ingénieur d'aujourd'hui de bien connaitre tous les circuits disponibles afin d'optimiser ses études. Dans le domaine combinatoire, ces circuits peuvent être très spécialisés (circuit arithmétique, encodeurs de priorité, convertisseurs de code, contrôleurs de parité...) ou d'un usage plus général (multiplexeurs, démultiplexeurs, mémoires mortes, circuit logiques programmables...)

Ce chapitre a pour but de poser les bases théoriques de quelques fonctions essentielles. Pour l'utilisation de circuits réels, on se reportera aux catalogues constructeurs.

3.1 Multiplexeurs, démultiplexeurs

3.1.1 Multiplexeur logique

Un multiplexeur logique est un circuit comportant N entrées et une sortie égale à un de ces entrées.

$$S = E_{adresse} \tag{3.1}$$

n entrées particulières permettent d'appliquer au circuit un mot d'adresse codé en binaire pur. cette adresse sélectionne la voie à transmettre.

En général, on a N = 2^n avec N = 2,4,8,16.

Remarque : Souvent une entrée supplémentaire de validation (strobe ou enable en anglais) forçant la sortie dans un état préfixé lorsqu'elle est inactive.

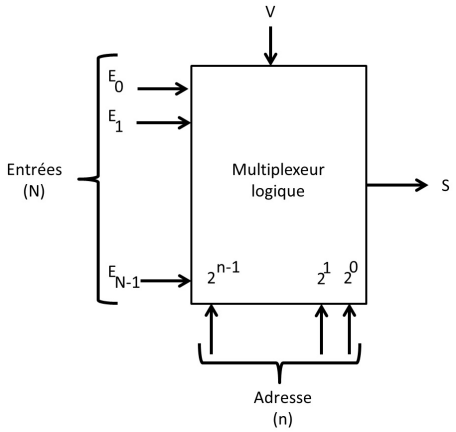


FIGURE 3.1 – Schéma générique d'un multiplexeur

Synthèse d'un multiplexeur à 4 entrées

La table de vérité est la suivante :

V	A_1	A_0	S
1	0	0	E_0
1	0	1	E_1
1	1	0	E_2
1	1	1	E_3
0	X	X	0

On en déduit l'équation logique de S :

$$S = (E_3 A_1 A_0 + E_2 A_1 \bar{A_0} + E_1 \bar{A_1} A_0 + E_0 \bar{A_1} \bar{A_0}).V \tag{3.2}$$

Ainsi que le schéma technologique correspondant :

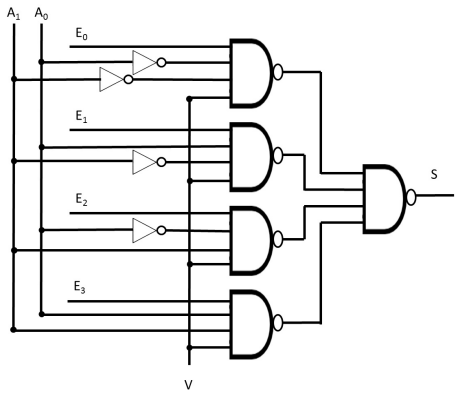


FIGURE 3.2 – Schéma technologique d'un multiplexeur

Réalisation de fonction combinatoires à l'aide de circuits multiplexeurs

L'exemple précédent nous a montré que l'équation logique du multiplexeur est de la forme suivante :

$$S = \sum_{i=0}^{N-1} E_i m_i \tag{3.3}$$

où E_i sont les différentes entrées (N) et m_i sont les minterms constitué avec les bits d'adresse (n). Il est ainsi très simple de réaliser toute fonction combinatoire ayant ces entrées connectées aux bits d'adresse par identification à la première forme canonique.

Exemple : soit la fonction $F(A, B, C, D) = ABD + AC\bar{D}$

On met cette fonction sous sa forme canonique :

$F = ABCD + AB\bar{C}D + ABC\bar{D} + A\bar{B}C\bar{D}$

$F = R(15, 13, 14, 10)$ en prenant les bits forts à gauche.

La réalisation à l'aide d'un multiplexeur à 16 entrées (n=4) est particulièrement simple. Les minterms présents dans le forme canonique seront activés par des "1" aux entrées correspondantes. Les minterms absents seront inhibés par des "0" aux entrées.

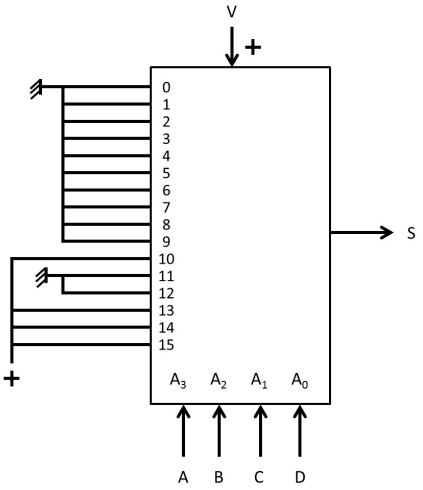


FIGURE 3.3 – Câblage de la fonction F avec un multiplexeur

Remarque : La réalisation précédente correspond à un cas général. Cependant l'exemple donné est assez particulier et permet d'importantes simplifications dans le schéma final :

1. La variable A peut se mettre en facteur
2. Les variables B et C n'apparaissent jamais complémentées.

1/ $F = A.R'(7, 5, 6, 2)$ donne une réalisation basée sur un multiplexeur à 8 entrées (boitier plus petit) comportant une entrée de validation (A).

2/ $BD + C\bar{D} = B(CD + \bar{C}D) + C\bar{D}$

ou $BD + C\bar{D} = BD + C(B\bar{D} + \bar{B}\bar{D})$

On divise encore le nombre d'entrée par 2.

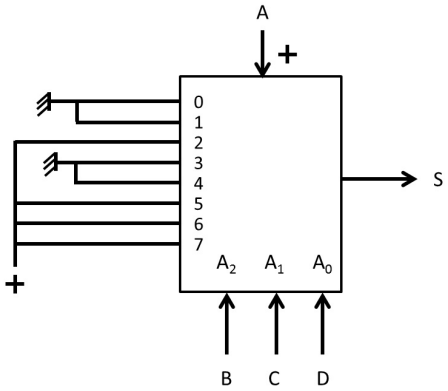


FIGURE 3.4 – Simplification 1

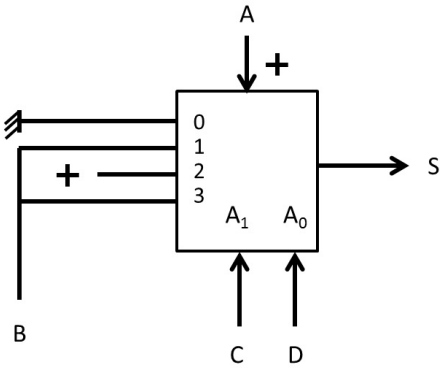


FIGURE 3.5 – Simplification 2

Augmentation du nombre d'entrées

On peut toujours décomposer en produit le nombre d'entrées. Le schéma technologique qui en résulte est constitué de deux couches de multiplexeurs, comme le montre l'exemple ci-dessous : $64 = 4 * 16$.

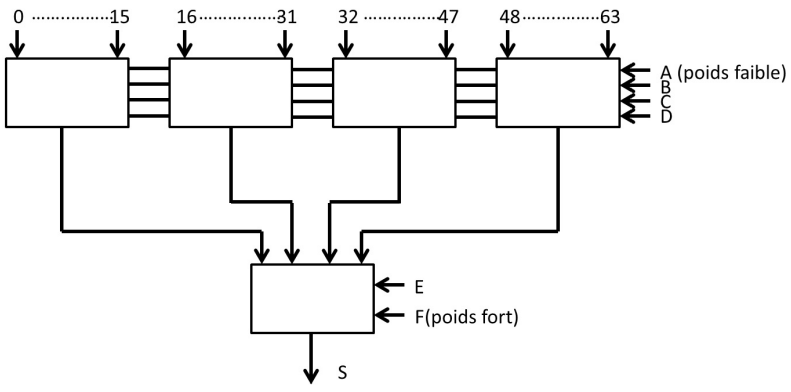


FIGURE 3.6 – Multiplexeur 64 entrées

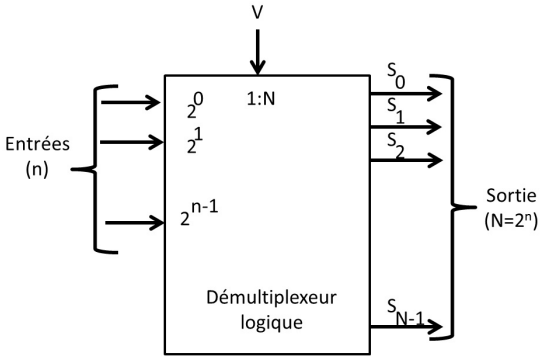


FIGURE 3.7 – Décodeur binaire

3.1.2 Décodeur binaire - démultiplexeur

Le décodeur binaire est un circuit comportant n entrées (adresse) et $N = 2^n$ sorties.

Lorsqu'une adresse est appliquée, la sortie d'indice correspondant devient active, toutes les autres sorties étant inactives. Ce circuit fait ainsi correspondre à un code binaire pur, un code à décalage.

On peut avoir une entrée supplémentaire de validation (strobe ou enable) qui dans son état complémentaire rend toutes les sorties simultanément inactives. On a alors affaire à un démultiplexeur : le niveau logique sur l'entrée validation est aiguillée vers la sortie repérée par le mot d'adresse appliqué.

Synthèse d'un démultiplexeur à 2 entrées - sortie active niveau bas

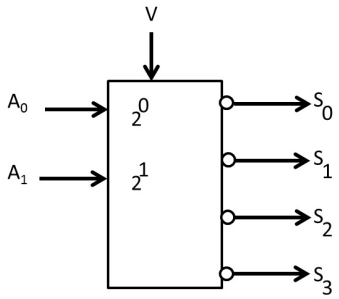


FIGURE 3.8 – Décodeur binaire à 4 sorties

V	A_1	A_0	S_0	S_1	S_2	S_3
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0
0	X	X	0	1	1	1

Equations logiques et schéma technologique

$$S_0 = \overline{\bar{A}_1 \bar{A}_0 V} \tag{3.4}$$

$$S_1 = \overline{\bar{A}_1 A_0 V} \tag{3.5}$$

$$S_2 = \overline{A_1 \bar{A}_0 V} \tag{3.6}$$

$$S_3 = \overline{A_1 A_0 V} \tag{3.7}$$

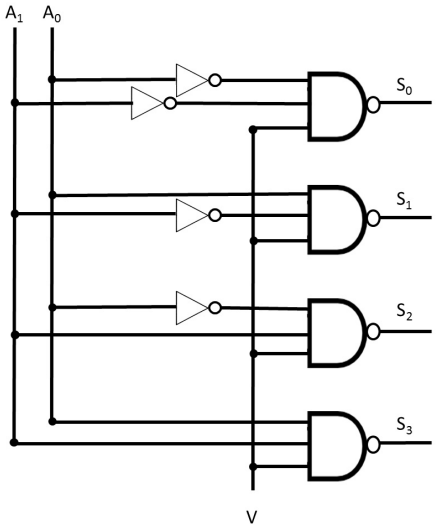


FIGURE 3.9 – Schéma technologique d'un décodeur binaire à 4 sorties

Réalisation de fonctions combinatoires à l'aide de circuits démultiplexeurs

L'exemple traité au paragraphe précédent nous a permis de démontrer que l'équation logique d'une sortie d'un codeur binaire à la forme suivante (pour un niveau actif haut) :

$$S_i = m_i \tag{3.8}$$

avec i l'adresse et m_i le minterms formé avec les entrées.

La première forme canonique d'une fonction combinatoire étant de la forme R(mi), on pourra toujours réaliser une telle fonction à l'aide d'un décodeur binaire et d'un circuit OU ayant autant d'entrée que la fonction de minterms.

Exemple : soit à réaliser la fonction $A \oplus B = \bar{A}B + A\bar{B}$ en utilisant le décodeur codé précédemment.

$$A \oplus B = R(1,2) = I(0,3) \tag{3.9}$$

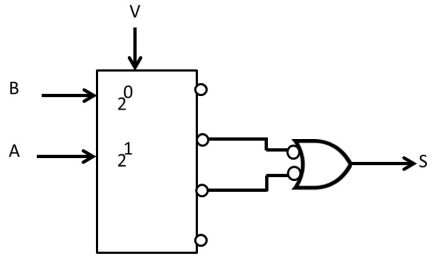


FIGURE 3.10 – Réalisation de la fonction F grâce à un décodeur binaire

Compte tenu du niveau bas du décodeur, on a deux solutions : avec une porte NAND on réalise la première forme canonique et avec une porte ET on réalise la deuxième forme canonique.

Remarque : Apparemment, comparativement aux réalisations comportant des multiplexeurs, celles basées sur l'emploi de décodeurs binaires n'ont aucun intérêt puisqu'on doit rajouter des boitiers de OU (ou NAND,ET...). Cependant, dans le cas de fonctions multiples des mêmes variables, on peut déjà trouver un bénéfice dans ce genre de réalisation. Les circuits ROM ou PLA qui intègre dans un même boitier le décodeur binaire et les fonctions OU, qui sont alors programmables par l'utilisateur, constituent une application directe de ce principe.

Augmentation du nombre d'entrées

Le nombre d'entrées peut être décomposé en produit, chaque sous multiple correspondant à un décodeur.

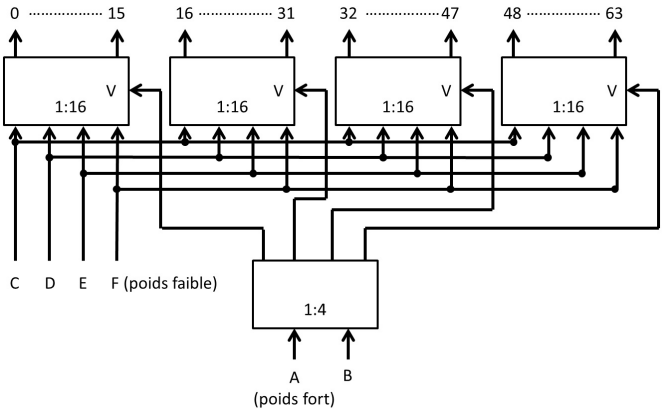


FIGURE 3.11 – Décodeur binaire à 64 sorties

3.2 Circuits arithmétiques

3.2.1 Comparateur à 4 bits

Soit deux mots binaires A et B

$$A = a_3 * 2^3 + a_2 * 2^2 + a_1 * 2 + a_0; B = b_3 * 2^3 + b_2 * 2^2 + b_1 * 2 + b_0 \quad (3.10)$$

On désire générer les trois fonctions suivantes

F = 1 si A sup B

G = 1 si A inf B

H = 1 si A = B

On peut remarquer que $G = \overline{HF}$. Il faut donc effectuer la synthèse des fonctions H et F.

Synthèse de H :

$H = 1$ si $a_3 = b_3$, $a_2 = b_2$, $a_1 = b_1$ et $a_0 = b_0$.

On peut donc en déduire que :

$$H = (a_3.b_3 + \bar{a_3}.\bar{b_3}).(a_2.b_2 + \bar{a_2}.\bar{b_2}).(a_1.b_1 + \bar{a_1}.\bar{b_1}).(a_0.b_0 + \bar{a_0}.\bar{b_0}) \quad (3.11)$$

On en déduit un schéma possible :

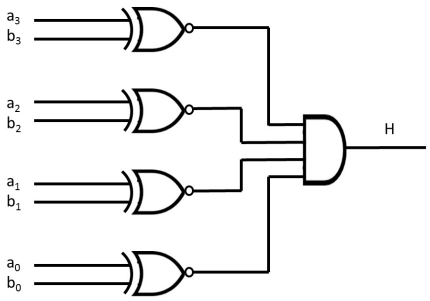


FIGURE 3.12 – Schéma technologique de la fonction H

Synthèse de F :

La méthode classique, consistant à faire une table de vérité de la fonction puis de la simplifier, s'avère particulièrement lourde voire impossible à appliquer. En effet, la fonction comporte 8 entrées et donc 256 combinaisons possibles. Il est donc préférable d'essayer d'obtenir directement une équation logique exploitable. En remarquant que $A > B$ si $a_3 > b_3$ OU si $a_3 = b_3$ ET $a_2 > b_2$ OU si..., on peut écrire :

$$F = a_3\bar{b_3} + \left(a_3.b_3 + \bar{a_3}\bar{b_3}\right)\left[a_2\bar{b_2} + \left(a_2.b_2 + \bar{a_2}\bar{b_2}\right).\left(a_1\bar{b_1} + \left(a_1.b_1 + \bar{a_1}\bar{b_1}\right)a_0\bar{b_0}\right)\right] \quad (3.12)$$

Cette équation peut s'écrire sous la forme d'une fonction récurrente :

$$F_n = a_n\bar{b_n} + \overline{(a_n \oplus b_n)}F_{n-1} \quad (3.13)$$

avec $F_0 = a_0\bar{b_0}$. On peut alors en déduire un schéma logique possible (non optimisé) :

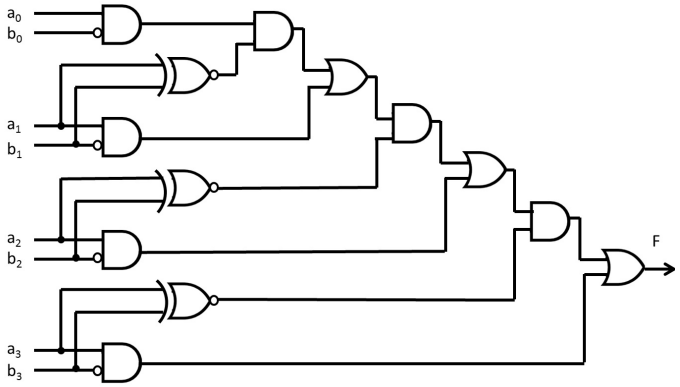


FIGURE 3.13 – Schéma technologique de la fonction F

3.2.2 Additionneur binaire

Additionneur à report série (Ripple carry)

Cet additionneur utilise un principe de récurrence. Pour construire un additionneur à n bits, on met en cascade n modules élémentaires appelés additionneur complet. La figure ci dessous représente le cas de l'addition de deux mots de 4 bits.

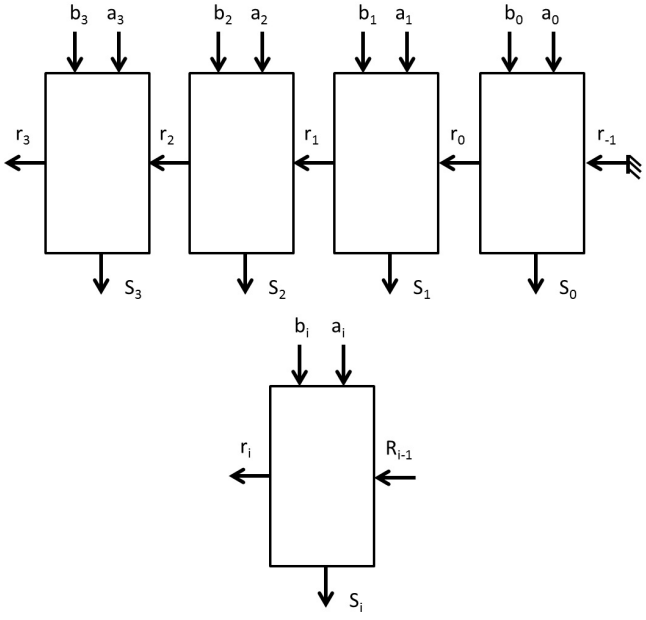


FIGURE 3.14 – Schéma générique d'un additionneur à report

Synthèse de l'additionneur complet :
Table de vérité : 3 entrée, 2 sorties

a_i	b_i	r_i	S_i	r_i
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

équation logique :

$$S_i = a_i\overline{(b_i + r_{i-1})} + \bar{a}_i(b_i \oplus r_{i-1}) = a_i \oplus b_i \oplus r_{i-1} \tag{3.14}$$

$$r_i = a_i b_i + a_i r_{i-1} + b_i r_{i-1} = a_i b_i + r_{i-1}(a_i \oplus b_i) \tag{3.15}$$

Schéma technologique (non optimisé) :

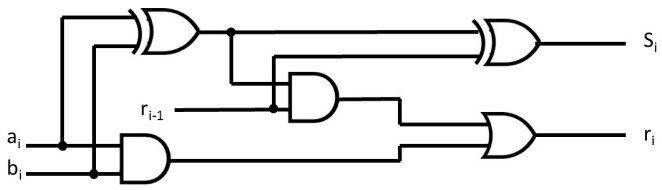


FIGURE 3.15 – Schéma technologique du module élémentaire d'un additionneur à report

Fonctionnement dynamique :

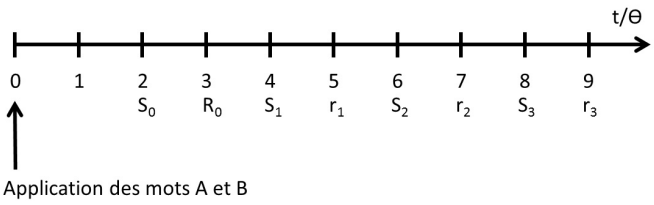


FIGURE 3.16 – Fonctionnement dynamique d'un additionneur à report

La propagation du report constitue un inconvénient majeur quant à la rapidité du calcul. Pour illustrer cela, supposons que chaque porte élémentaire du circuit est affecté d'un retard à la propagation θ. Lorsqu'on applique à l'additionneur complet simultanément ses trois entrées, il faudra 2θ pour avoir le résultat S_i stable en sortie et 3θ pour r_i. Pour un additionneur 4 bits, le régime transitoire pourra être schématisé par le diagramme suivant :

Ainsi plus on augmentera la capacité de l'additionneur, plus le temps de réponse de celui-ci sera long. De plus pendant toute la durée de ce temps de réponse, des valeurs parasites peuvent intervenir en sortie.

Si l'on veut minimiser cet inconvénient, il est nécessaire de changer le principe du circuit.

Additionneur à retenue anticipée (carry look-ahead)

Le principe est le suivant : on calcule chaque report directement à partir des entrées de rang inférieur.

$$r_i = a_i b_i + r_{i-1}(a_i + b_i) = g_i + P_i r_{i-1} \tag{3.16}$$

où P_i est le terme de propagation car si $r_{i-1} = 1$ alors $r_i = 1$ si $a_i = 1$ OU $b_i = 1$. g_i est le terme de génération car $r_i = 1$ si $a_i = 1$ ET $b_i = 1$.

On peut alors dessiner le module de base suivant :

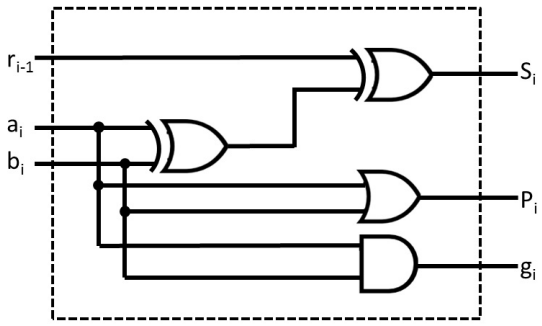


FIGURE 3.17 – Schéma technologique du module élémentaire d'un additionneur à retenue anticipée

Pour la réalisation d'un additionneur à 4 bits, il faudra associer 4 modules de base et y adjoindre un circuit combinatoire réalisant le calculs des reports appelé générateur de retenue anticipée.

Le schéma électrique du générateur de retenue anticipée met en évidence des temps de traversée identiques (2 couches à traverser) pour l'obtention des reports. Ainsi pour reprendre l'exemple déjà traité, si à $t = 0$ on applique simultanément A et B, à $t = \theta$ seront valides tous les P_i et g_i, à $t = 3\theta$ tous les r_i et $t = 4\theta$ tous les S_i. La fabrication parallèle des reports se traduit donc par une augmentation significative de la rapidité de l'additionneur.

On retrouve ce principe utilisé dans les unités arithmétiques et logiques (ALU).

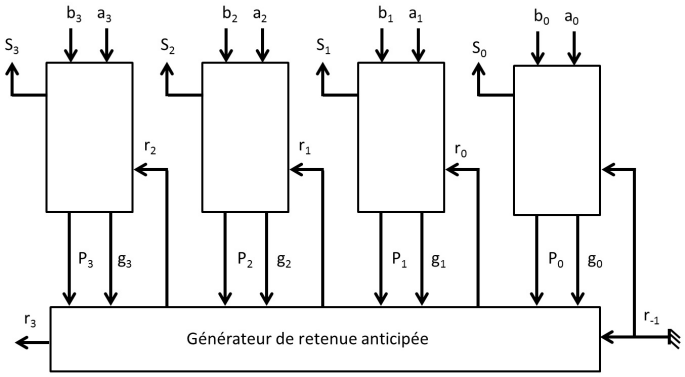


FIGURE 3.18 – Schéma générique d'un additionneur à retenue anticipée

Chapitre 4

Circuits combinatoires programmables

4.1 Simplification des fonction par la méthode de Quine-Mac Cluskey

Cette méthode offre l'avantage d'être facilement programmable. Elle est souvent à la base des logiciels de simplification de fonctions logiques très utilisés lorsque le nombre de variables est grand ou bien intégrés à des outils de programmation de circuits. Elle est basée sur le théorème d'adjacence ($A\bar{B} + AB = A$).

La méthode s'explique aisément sur un exemple classique et simple. Soit la fonction F de quatre variables exprimée sous sa première forme canonique :

$F(A, B, C, D) = R(0, 1, 2, 3, 4, 6, 7, 8, 9, 11, 15)$, les variables A, B, C et D étant arbitrairement pondérées de façon décroissante.

4.1.1 Recherche systématique des adjacences

On établit une liste une première liste en classant les termes selon le nombre de "1" qu'ils contiennent dans leur écriture binaire. On obtient une succession d'ensembles de termes regroupés selon une variable de comptage c.

liste 1

$c = 0$	0
$c = 1$	1,2,4,8
$c = 2$	3,6,9
$c = 3$	7,11
$c = 4$	15

Deux termes adjacents ne diffèrent que par un bit complémentaire. S'ils existent, ils se trouvent ainsi dans deux ensembles successifs.

On construit une deuxième liste en comparant successivement chaque terme d'un ensemble suivant. Si une adjacence est reconnue (différence des termes décimaux comparés égale à une puissance de 2), alors le bit représentant la variable d'adjacence est éliminé. En fait, on créera un masque de suppression associé au minterm). Ainsi pour l'exemple traité :

Liste 2

$c = 0$	0,1	000-
	0,2	00-0
	0,4	0-00
	0,8	-000
$c = 1$	1,3	00-1
	1,9	-001
	2,3	001-
	2,6	0-10
	4,6	01-0
	8,9	100-
$c = 2$	3,7	0-11
	3,11	-011
	6,7	011-
	9,11	10-1
$c = 3$	7,15	-111
	11,15	1-11

On remarque que cette liste est nettement plus longue que la première mais que le nombre d'ensembles pointés par c a diminué de 1. Ainsi par la suite, on va reprendre ce processus jusqu'à aucun événement ne se produise. Pour les adjacences suivantes, seuls sont comparés les termes avec le même masque.

Liste 3

$c = 0$	0,1,2,3	00–
	0,1,8,9	-00-
	0,2,1,3	00–
	0,4,2,6	0–0
	0,8,1,9	-00-
	0,2,4,6	0–0
$c = 1$	1,3,9,11	-0-1
	2,3,6,7	0-1-
$c = 2$	3,7,11,15	–11

On ne peut aller au delà. Toutes les adjacences possibles ayant été envisagées, on obtient un premier résultat brut :

$$F = \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{D} + \bar{B}D + \bar{A}C + CD \tag{4.1}$$

4.1.2 Elimination des redondances

La méthode ayant eu un point de départ arbitraire, elle a envisagé toutes les adjacences possibles (certainement trop) mais n'a pu s'assurer que chaque minterm n'avait besoin de n'être recouvert qu'une seule fois (au minimum). On va donc vérifier, à posteriori, ces recouvrements au moyen d'une grille de minterms.

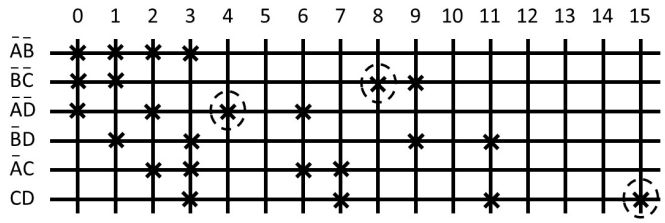


FIGURE 4.1 – Schéma technologique d'un décodeur binaire à 4 sorties

Ayant porté en abscisse les termes premiers de la fonction originelle et en ordonnées les implicants trouvés en première phase de simplification, on note d'une croix tous les termes que recouvrent ces implicants.

Puis, comme on le ferait sur une table de Karnaugh, on va chercher les termes les plus isolés et recouvrir progressivement (le plus rapidement possible) la fonction dans son ensemble. En observant les verticales de la grille, on trouve les minterms 4, 8 et 15 n'ayant qu'une possibilité de recouvrement, par respectivement $\bar{A}\bar{D}$, $\bar{B}\bar{C}$ et CD. Ces trois implicants recouvrent ensemble les minterms 0, 1, 2, 3, 4, 6, 7, 8, 9, 11, 15 et donc toute la fonction F. Les autres termes sont donc redondants et la méthode a atteint son point final.

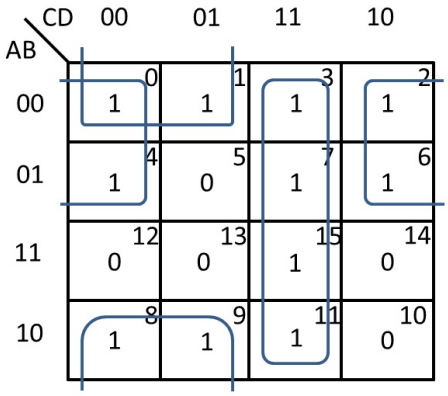


FIGURE 4.2 – Schéma technologique d'un décodeur binaire à 4 sorties

La fonction F simplifiée est donc :

$$F = \bar{A}\bar{D} + \bar{B}\bar{C} + CD \tag{4.2}$$

Il est intéressant de terminer cet exemple simple en observant cette même simplification sur une table de Karnaugh (cf.figure 2).

4.2 Les mémoires mortes

Ce sont des mémoires à lecture seule et à accès aléatoire encore appélées ROM (Read Only Memory). L'information est présente de façon permanente, "inscrite" dans le circuit, et est disponible dès que celui-ci est alimentée.

4.2.1 Schéma de principe

Une mémoire morte est constituée de :

- Une matrice à diodes programmables assurant une liaison ou pas entre lignes et colonnes.
- Un décodeur d'adresse de n fils commandant les 2^n lignes.
- Un circuit de génération des niveaux corrects des sorties.

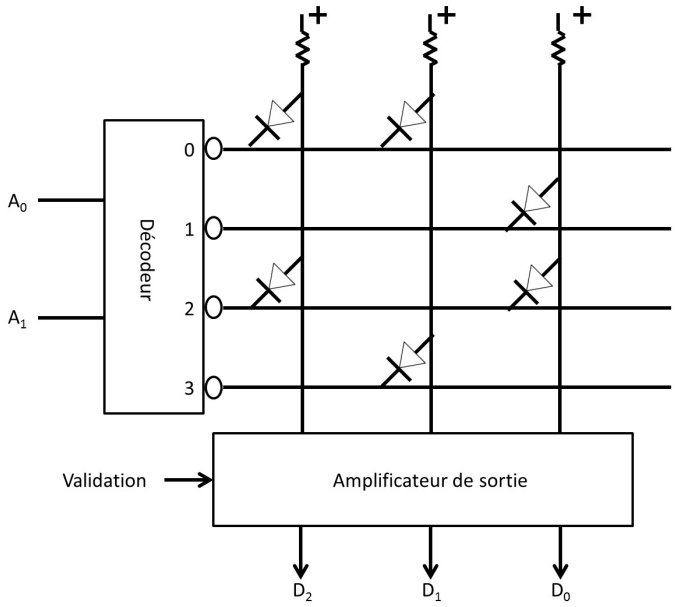


FIGURE 4.3 – Schéma technologique d'un décodeur binaire à 4 sorties

La figure illustre la réalisation d'une mémoire programmée de 4*3 bits. Lorsque une ligne est adressée via le décodeur, elle est portée à un niveau BAS tandis que toutes les autres sont au niveau HAUT. En conséquence, les diodes connectées à cette ligne conduisent et forcent les colonnes concernées aussi au niveau bas. On peut ainsi établir à propos de l'exemple la table de vérité suivante :

Adresses (	A_0	A_1	) Données (	D_2	D_1	D_0	)
	0	0		0	0	1	
	0	1		1	1	0	
	1	0		0	1	0	
	1	1		1	0	1	

Cette table est la *table de programmation* de la mémoire. Etant décrite par une table de vérité, chaque sortie correspond à une fonction combinatoire des entrées d'adresse.

$$D_2 = A_0 \tag{4.3}$$

$$D_1 = A_1\bar{A_0} + \bar{A_1}A_0 \tag{4.4}$$

$$D_0 = A_1A_0 + \bar{A_1}\bar{A_0} \tag{4.5}$$

Différents types technologiques

Ce qui caractérise une mémoire morte, c'est en premier lieu d'être constituée d'un réseau ET fixe (le décodeur qui a 2^n portes à n entrées) associé à un réseau OU programmable (liaison entre lignes et colonnes). L'opération d'inscription des données réalisant l'ensemble de ces OU est la programmation de la mémoire. Diverses astuces technologiques sont mises à profit pour cette opération et conduisent à distinguer entre plusieurs types de ROM.

- ROM : la table de programmation est envoyée au fabricant de circuit qui réalise pour la matrice un masque de métallisation correspondant. Il est alors en mesure d'assurer la production d'un grand nombre de circuits identiques. Cette manière de procéder n'est rentabilisée que par un grand nombre de circuits et exclut de ce fait la réalisation de prototypes.
- PROM (Programmable ROM) : On achète des circuits vierges pour lesquels toutes les possibilités de liaisons lignes-colonnes sont établies (dans l'exemple précédent, on ne lirait alors que des "0"). Les liaisons sont par exemple constituées d'une diode et d'un petit fusible en série. L'utilisateur a la possibilité par un adressage adéquat conjugué avec une impulsion électrique calibrée de faire fondre le fusible là où il veut faire apparaitre l'information complémentaire ("1" dans l'exemple). L'opération de programmation est évidemment grandement simplifiée si l'on utilise un appareil spécialisé : le programmateur de mémoire morte. Ce genre de circuit est très adapté au travail de prototypage. L'opération de programmation est cependant irréversible (sauf pour un oubli), chaque erreur de mise au point correspond à un coût.

– EPROM (erasable PROM) : il s'agit de mémoire programmable électriquement et effaçable par exposition à un rayonnement ultraviolet. A cet effet, le dessus du boîtier est une fenêtre en quartz qui laisse voir la puce.
– EEPROM ou E2PROM : ces EPROM sont effaçables électriquement, fonction qui peut ainsi être incorporée au programmateur.

4.2.2 Les réseaux logiques programmables

Philosophie

Lorsque le nombre n de variables augmente, le nombre de 2^n minterms qui forment le décodeur finissent par occuper une surface très importante. Or souvent (dans 90 % des cas), les fonctions combinatoires à réaliser ne comportent qu'un nombre limité de minterms (N«2^n).

Dans un réseau logique programmable, ou PAL (Programmable Array Logic), on trouve une zone ET programmable dans laquelle on va pouvoir implanter un certain nombre de produits limités, associée à une zone OU programmable pour former des sorties. Ceci correspond à une meilleure occupation de la surface du silicium

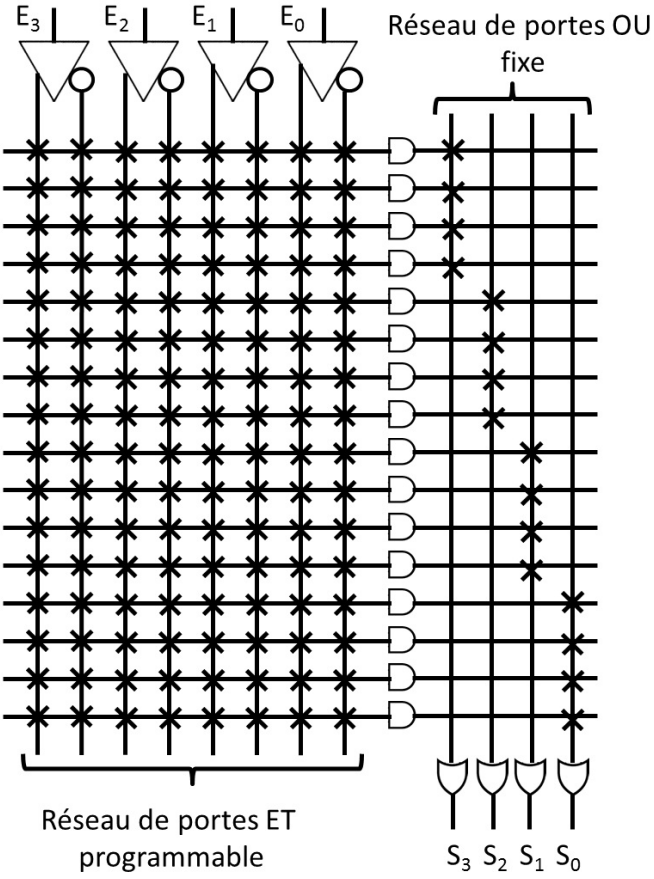


FIGURE 4.4 – Schéma technologique d'un décodeur binaire à 4 sorties

Utilisation

Concrètement un circuit PAL, qui utilise plus souvent la technologie fusible, le circuit EPLD, à effacement par ultraviolet, ou encore le FPGA, effaçable électriquement, se caractérisent par les trois termes A*B*C soit respectivement : nombre d'entrées, nombre de produits possibles, nombre de sorties.

Exemple : PAL_16L2 16 entrées, 8 produits et 2 sorties.

Les programmateurs de PAL comporte heureusement un logiciel de calcul des termes premiers de décomposition de fonctions multiples. Les équations logiques peuvent être rentrées sous forme quelconque a l'aide d'une syntaxe simple.

Deuxième partie

Logique séquentielle

Chapitre 5

Mémoires et bascules

5.1 Circuit séquentiels

Dans un circuit combinatoire, les valeurs des sorties ne dépendent que des valeurs des entrées, au temps de traversée près. La description du circuit peut se faire par une table de vérité, la structure du circuit est arborescente.

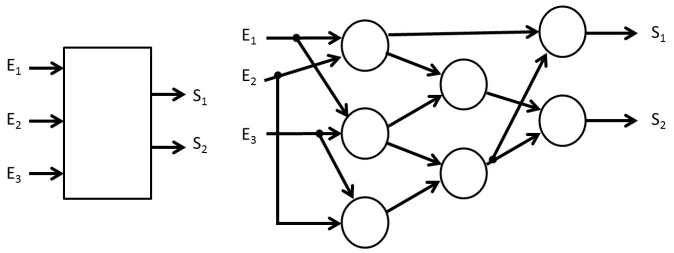


FIGURE 5.1 – Circuit combinatoire

Dans un circuit séquentiel, les sorties dépendront encore des entrées mais aussi de l'état précédent l'apparition de cette combinaison d'entrées. L'état peut être défini comme l'ensemble des variables du circuit stabilisé. En d'autres termes, les sorties dépendent de l'ordre d'arrivée des entrées.

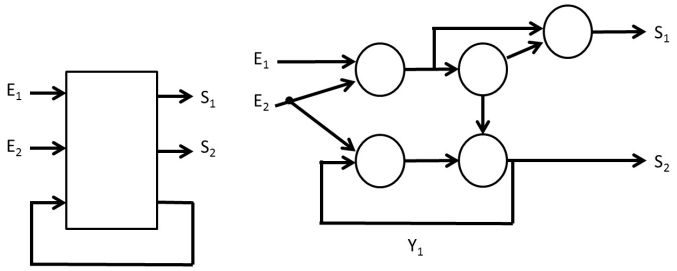


FIGURE 5.2 – Circuit séquentiel asynchrone

La méthode d'analyse ou de synthèse d'un circuit séquentiel consiste en premier lieu à le ramener à un circuit combinatoire. Pour cela, on considère en plus des variables d'en-

trée primaire des variables internes appelés "secondaires". L'existence de ces variables internes se traduit au niveau du schéma logique du circuit par la présence de boucles.

Lorsque les variables internes sont localisées, on peut alors "ouvrir les boucles" et traiter le problème combinatoire en se plaçant au moment précis de son changement d'état. On supposera un seul changement à la fois.

On appelle t_N l'instant de cette transition et des t_{N+1} l'instant correspondant à la réponse d'une variable contenant du temps de propagation inhérent au circuit de base.

Pour le circuit le plus général (de type asynchrone), on a à l'instant t_{N+1} des équations logiques du type :$y^i_{n+1} = f(E_1, E_2, ..., E_N, y_1, ..., y^i_n)$ pour une variable interne.

Les sorties peuvent être :

– soit elles-mêmes des variables internes.
– soit être fonction des variables internes seules.
– soit être fonction des variables internes et des sorties.

Un état stable est caractérisé par la relation $y_{n+1} = y_n$ appliquée à des instants extérieurs à la période de transition (dans ce cas là $y_{n+1} = \bar{y}_n$ correspondrait à une instabilité).

5.2 Mémoire Set-Reset

5.2.1 Principe

C'est le circuit séquentiel le plus simple et le plus fondamental.

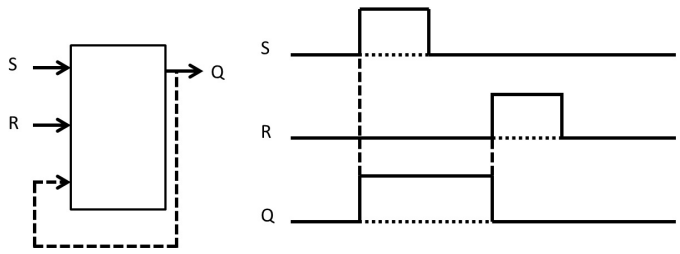


FIGURE 5.3 – Mémoire Set-Reset

Set (S) est l'entrée de mise à 1, Reset (R) est l'entrée de mise à 0. Il est interdit d'appliquer simultanément ces deux entrées (R.S = 0).

5.2.2 Equation

La figure 5.3 illustre par un diagramme temporel la relation entre entrées et sorties : Toutes les séquences ne sont pas représentées. En parcourant ce diagramme, on peut aisément constater la nature séquentielle du circuit. En effet, pour S=R=0, on voit que la sortie Q peut prendre soit la valeur "0", soit la valeur "1", ce qui n'est pas possible dans un circuit combinatoire. C'est cela qui constitue l'effet "mémoire". Pour écrire l'équation combinatoire de la sortie, on va introduire une variable interne permettant de différencier

les deux états pour S=R=0, donc ayant la valeur "0" pour Q = 0 et "1" pour Q=1, par exemple.

La variable interne peut donc alors être la sortie Q elle-même. L'équation du Set-Reset est donc une relation combinatoire de la forme :

$$Q_{n+1} = f(R, S, Q_n) \tag{5.1}$$

Le plus simple pour l'établir est de représenter tous les cas possibles au moyen d'une table de Karnaugh.

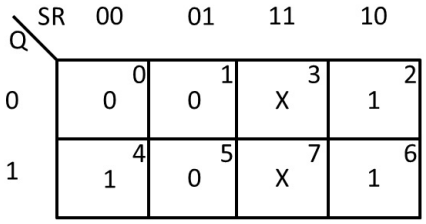


FIGURE 5.4 – Schéma technologique d'un décodeur binaire à 4 sorties

Selon les cas envisagées pour Q_{n+1} lorsque S=R=1 (qui est un cas impossible), on obtient quatre équations différentes, dont nous ne retiendrons que les trois suivantes :

1) mémoire à inscription prioritaire (S=R=1, Q=1)

$$Q_{n+1} = S + \bar{R}Q_n \tag{5.2}$$

Se réalise simplement avec des portes NAND.

2) mémoire à effacement prioritaire (S=R=1, Q=0)

$$Q_{n+1} = S\bar{R} + \bar{R}Q_n \tag{5.3}$$

Se réalise simplement avec des portes NOR.

3) mémoire P-Q (S=R=1, $Q_{n+1} = Q_n$)

$$Q_{n+1} = S\bar{R} + (S + \bar{R})Q_n \tag{5.4}$$

Ce circuit est plus complexe à réaliser et assez peu utilisé.

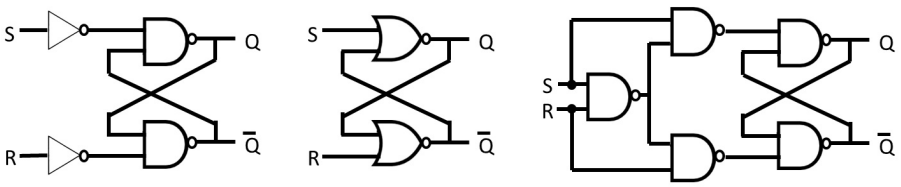


FIGURE 5.5 – Schéma technologique des mémoires Set-Reset

5.2.3 Contrainte temporelle

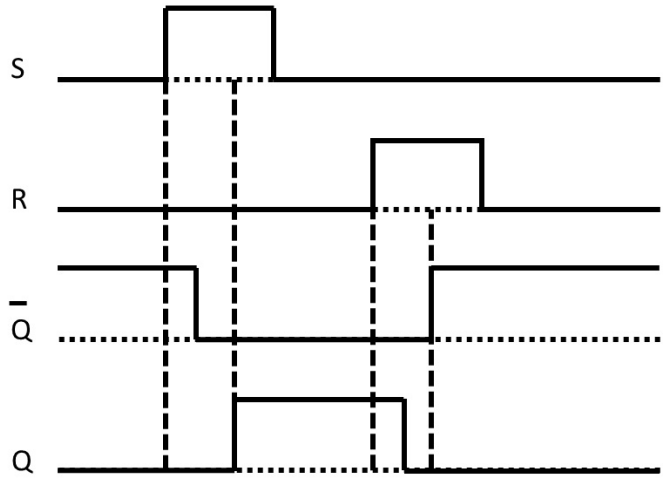


FIGURE 5.6 – Comportement dynamique de la mémoire Set-Reset

En supposant chaque porte de base affectée d'un retard à la propagation θ, on constate lors d'une commutation :

- qu'il faut 2θ pour que les 2 sorties soient stables.
- que les deux sorties se recouvrent pendant θ.
- que la sortie la plus rapide vaut "0" pour une réalisation NOR et "1" pour une réalisation NAND.
- qu'il faut 2θ pour que la sortie confirme l'entrée de commande. Ce temps est appelé "largeur minimum de l'impulsion d'entrée".

5.3 Mémoire RSH, mémoire D

5.3.1 Mémore RSH

Une mémoire RSH est une mémoire RS à laquelle on rajoute une entrée H (horloge) d'inhibition des entrées. Les commandes vraies du Set-Reset sont respectivement SH et RH. Le fonctionnement est donc le suivant :

- Tant que H=1, on a une bascule RS asynchrone.
- Si H=0, le Set-Reset est isolé, au repos en état mémoire.

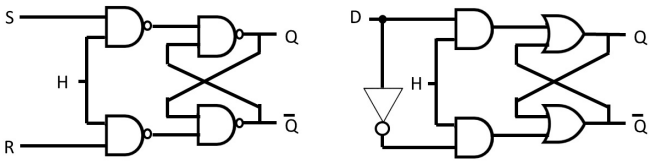


FIGURE 5.7 – Mémoires RSH et D

Si maintenant, on impose que les données d'entrée S et R ne changent que lorsque H=0, alors l'apparition de la sortie sera synchrone avec le front montant de H. Compte tenu de cette hypothèse restrictive on résumera la situation en disant que la mémoire fonctionne sur niveau. Son équation logique est :

$$Q_{n+1} = H(S\,\bar{Q}_n + \bar{R}Q_n) + \bar{H}Q_n \tag{5.5}$$

5.3.2 Mémoire D

Une mémoire D (Delay) est une mémoire RSH dans laquelle $S = \bar{R} = D$. Lorsque H = 1, on a Q = D et lorsque H=0, il y a mémorisation du dernier état au moment du front descendant de H. Ce fonctionnement correspond à l'équation logique suivante :

$$Q_{n+1} = HD + \bar{H}Q_n \tag{5.6}$$

On peut aussi définir la fonction X_Q, dite de commutation, qui est équivalente :

$$X_Q = H(D\bar{Q} + \bar{D}Q) \tag{5.7}$$

5.4 Principe des bascules

5.4.1 Diviseur par 2

Une bascule est un dispositif de mémorisation plus complexe que le précédent ayant la propriété de pouvoir fonctionner en diviseur par 2 de l'entrée de synchronisation H.

Ceci implique un fonctionnement de H sur front et non plus sur niveau. On écrira $Q_{n+1} = \bar{Q}_n H^*$ ou $X_Q = H^*$, où H^* décrit un front actif de H. Il pourra être omis dans les équations lorsqu'il n'y aura pas d'ambiguïté sur la source de synchronisation.

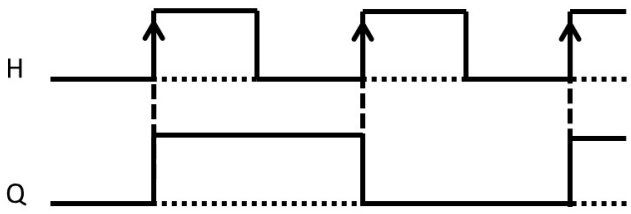


FIGURE 5.8 – Diviseur par 2

5.4.2 Bascule par une impulsion calibrée

Pour concevoir une bascule, on peut procéder de manière tout à fait intuitive et chercher à commander de façon alternée un circuit de sortie de type mémoire RS.

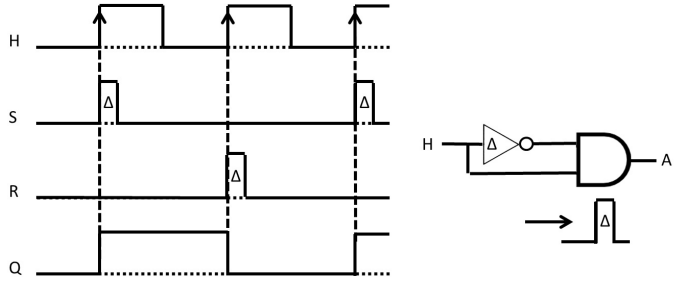


FIGURE 5.9 – Diagramme temporel incluant l'impulsion calibrée

La question est de savoir comment fabriquer une impulsion à chaque front de H. Une solution électronique (absurde d'un point de vue logique) consiste à générer un aléa statique en mettant en parralèle deux chemins différents d'un retard Δ. Il suffit d'aiguiller cette impulsion I en fonction de l'état de sortie $S = I\bar{Q}$ et $R = IQ$ et on aura un schéma de principe de la bascule.

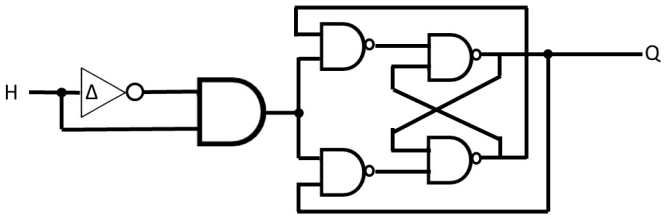


FIGURE 5.10 – Bascule à impulsion calibrée

L'analyse du fonctionnement dynamique de ce schéma nous indique que l'impulsion doit être calibrée à une largeur telle que $2\theta < \Delta < 3\theta$, en appelant θ le retard d'une porte de sortie. La première inégalité correspond à la commande correcte de la mémoire Set-Reset tandis que la seconde correspond à la limite au delà de laquelle il y aurait oscillation.

Ce principe n'est pas très bon car délicat à mettre en oeuvre. Son seul avantage est éventuellement au niveau de la faible surface de silicium du dispositif. C'est pour cette raison qu'on le trouve appliqué surtout dans de vieux circuits tels que 74112, 74113 ou 74114.

5.4.3 Bascules à deux variables internes

En considérant le signal d'entrée H et le signal de sortie Q, on peut chercher en introduisant des variables logiques supplémentaires à séparer tous les états. L'ajout de la variable Y déclenchant sur front descendant de H suffit à transformer le problème en combinatoire à deux variables internes Y et Q que l'on peut résoudre en utilisant une table de Karnaugh.

On obtient alors deux équations symétriques :

$$Y_{n+1} = \bar{H}Q_n + HY_n = HY_n + \bar{H}(QY_n + Q\bar{Y}_n) \tag{5.8}$$
$$Q_{n+1} = \bar{H}Q_n + H\bar{Y}_n = \bar{H}Q_n + H(\bar{Y}_nQ_n + \bar{Y}_n\bar{Q}_n)$$

Sous la deuxième forme, il est facile d'identifier les équations à celles de mémoires RSH et retrouver la solution évidente à priori :

– Pour Q, horloge H, mise à 1 $\bar{Y}$, mise à 0 Y.
– Pour Y, horloge $\bar{H}$, mise à 1 Q, mise à 0 $\bar{Q}$.

On obtient le schéma de type maître-esclave de la figure 11.

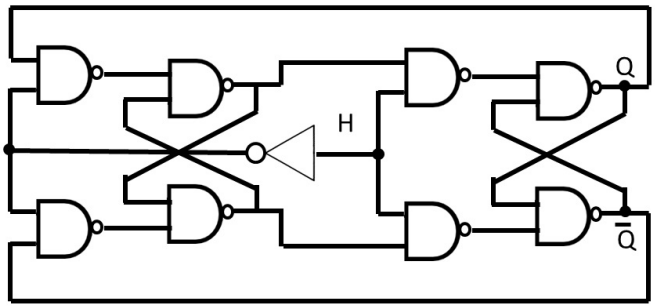


FIGURE 5.11 – schéma de type Maître-Esclave

5.5 Classification logique des bascules

5.5.1 Bascule S-R Maître-Esclave

Elle correspond aux équations suivantes équivalentes :

$$Q_{n+1} = H^*(S\,\bar{Q}_n + \bar{R}Q_n) \tag{5.9}$$

$$X_Q = H^*(S\,\bar{Q} + \bar{R}Q) \tag{5.10}$$

Le schéma est celui du paragraphe précédent sans la rétroaction sortie-entrée et correspond à deux mémoires RSH en cascade avec des horloges en opposition (figure 13).

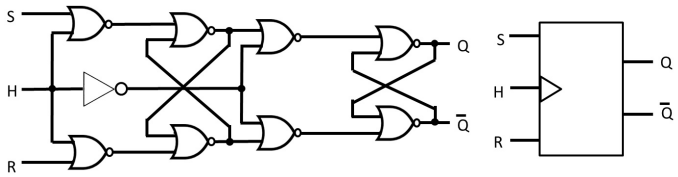


FIGURE 5.12 – RSH Maître-Esclave

Lorsque $H = 0$, il y a enregistrement des commandes d'entrée par la mémoire maître (sortie y du schéma) avec un fonctionnement sur niveau. Lorsque $H \rightarrow 1$, les entrées R

et S sont inhibées et le contenu du "maître" est recopié dans l'esclave. La sortie est ainsi synchronisée sur front montant de H.

5.5.2 Bascule JK

Il s'agit d'une bascule RS synchronisant sur front (J = S et K = R) avec en plus le fonctionnement en diviseur par 2 lorsque J = K = 1, selon la table de vérité réduite suivante :

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Les équations du dispositif sont :

$$Q_{n+1} = (J\overline{Q_n} + \bar{K}Q_n)H^* \tag{5.11}$$
$$X_Q = (J\overline{Q} + KQ_n)H^*$$

Il existe différentes conceptions et modes de fonctionnement des bascules JK, du plus simple au plus compliqué. On peut distinguer 3 types de montages.

JKH Maître-Esclave à simple inhibition

Le schéma est le même que pour la bascule RSH et le diviseur par 2 (figure 14). Les bascules 7476 et 7472 utilisent ce principe.

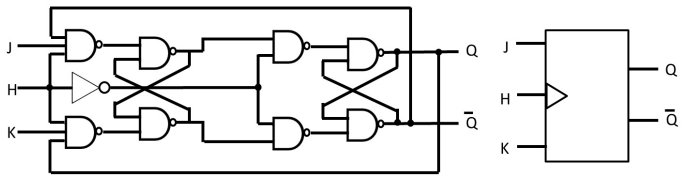


FIGURE 5.13 – Bascule JKH Maître-Esclave

JKH Maître-Esclave avec verrouillage des entrées

Dans ce type de schéma (avec verrouillage des entrées), les entrées sont automatiquement inhibées après un laps de temps Δ (en utilisant une impulsion calibrée). On fabrique en fait la fonction $H\bar{H} = 0$ qui devient 1 durant le recouvrement due à l'impulsion calibrée. La bascule de type 74111 utilise ce principe.

JKH à synchronisation par front (Edge-Triggered)

C'est le type le plus complet et le plus sûr à utiliser puisque le même front prend en compte les entrées et synchronise la sortie. Ce résultat est obtenu soit par impulsion calibrée (Exemple de la bascule 74112) soit de façon purement logique (Exemple de la bascule 74110).

5.5.3 Bascule T (Toggle)

Cette bascule permet soit la mémorisation d'un état (T=0), soit la fonction de diviseur par 2 (T=1). Sa fonction est la forme suivante :

$$Q_{n+1} = (T\bar{Q}_n + \bar{T}Q_n)H^* \tag{5.12}$$

5.5.4 Bascule D (Delay)

C'est la bascule la plus utilisée et elle est très performante. Son équation est la plus simple ($Q_{n+1} = DH^*$). Son schéma optimisé présenté à la figure 14 est utilisé dans la bascule 7474.

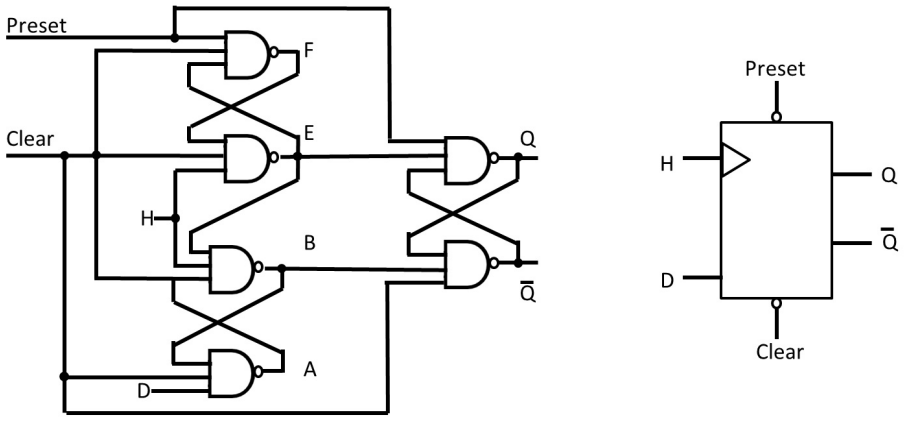


FIGURE 5.14 – Bascule D

Analyse succinte :

- Pour H = 0, E=B=1 la mémoire est au repos.
- Sur front montant de H, $E \rightarrow \bar{D}$ et $B \rightarrow D$ conduisent aux commutations $Q \rightarrow 1$ et $\bar{Q} \rightarrow 0$. D'autre part, le niveau 0 de E est inhibiteur pour les 2 portes qui pourraient transmettre un changement de D. Il y a donc aussi verrouillage de l'entrée et synchronisation par front.
- Les remises à zéro (Clear) et à un (Preset), actifs au niveau bas, sont de nature asynchrone (pas besoin de l'horloge) et sont donc prioritaires sur le fonctionnement synchrone (avec horloge). Il agissent par niveau et non pas sur front. Il est fortement

déconseillé de les activer de façon simultané, le fonctionnement de la bascule étant alors incertain.

L'équation complète d'une telle bascule est donc :

$$Q_{n+1} = DH^* + \overline{preset}.\bar{Q} + \overline{clear}.Q \tag{5.13}$$

avec $\overline{preset}.\overline{clear} = 0$.

5.6 Caractéristiques temporelles

5.6.1 Temps de propagation

Le temps de propagation est mesuré en utilisant comme référence le front actif de l'horloge (50% ou seuil précisé). La valeur tpHL est le temps pour la sortie passe au niveau bas. La valeur tpLH est le temps pour la sortie passe au niveau haut.

5.6.2 Temps de préconditionnement (Set-up)

C'est la durée minimale de présence des entrées avant le front actif d'horloge.

5.6.3 Temps de maintien (Hold)

C'est la durée minimale de présence des entrées avant le front actif d'horloge.

5.6.4 Autres caractéristiques :

Largeur minimum d'horloge
Durée du clear, du preset, etc...
Fréquence maximale d'horloge.

5.7 Conclusion

Lorsqu'on doit choisir un composant dans un catalogue, chaque détail de fonctionnement peut avoir son importance. Nous avons vu dans ce chapitre un grand nombre d'aspect de ce problème qui peuvent se combiner. On doit donc pour une bascule donnée savoir répondre à toutes les questions suivantes :

- les entrées sont elles synchrones ?
- les clear et preset sont ils synchrones ?
- y-a-t-il des entrées sur niveau ?
- y-a-t-il verrouillage ?

Chapitre 6

Registres et compteurs

6.1 Machines d'états finis

6.1.1 Classification

L'état d'une machine est la propriété de cette machine telle que la connaissance du vecteur d'entrée et de l'état à l'instant t = t_n détermine complètement le vecteur de sortie pour t = t_{n+1}. Dans le cas d'une machine asynchrone, ce sont les transitions d'entrée (au temps de réponse près) qui correspondent aux instant t_n. Dans le cas d'une machine synchrone, les instant t_n sont calés sur une horloge externe commune aux éléments mémorisant l'état. La conception synchrone est grandement simplificatrice puisqu'elle permet en quelque sorte de maitriser l'aspect temporel du circuit en échantillonnant les entrées à des moments précis. On pourra ainsi faire abstraction des retards, à condition d'accepter que la période d'horloge soit au minimum égale au plus grand retard possible de passage entre un état et le suivant, voire même plus comme cela sera expliqué dans la suite.

On peut distinguer plusieurs architectures (ou classes) de machines séquentielles :

Classe A ou machine de Mealy : les sorties sont fonctions des entrées présentes et de l'état de la machine.

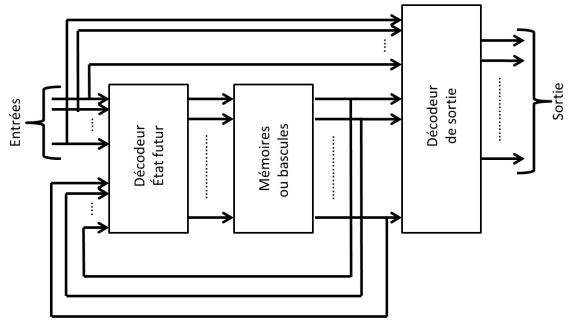


FIGURE 6.1 – Machine de Mealy

Classe B et C : machine de Moore. Les sorties sont strictement fonctions de l'état de la machine. En classe C, les sorties sont des variables internes

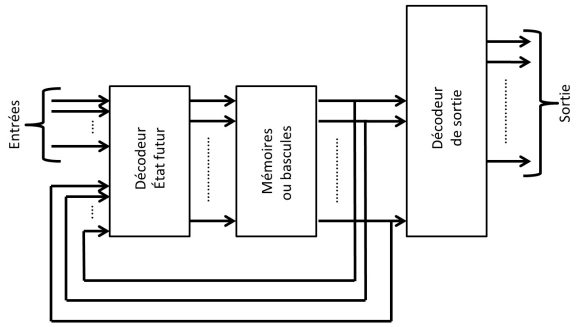


FIGURE 6.2 – Machine de Moore classe B

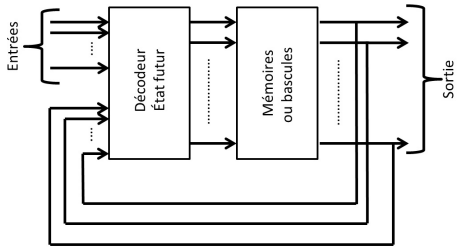


FIGURE 6.3 – Machine de Moore classe C

Remarques : Si on utilise des éléments mémoire avec une entrée d'horloge commune, les machines de classes B et C sont entièrement synchrones. Par contre en classe A, les sorties peuvent avoir un caractère asynchrone.

6.1.2 Diagramme d'état

Le diagramme d'état est un graphe décrivant sous forme géométrique le comportement d'un système.

normalisation

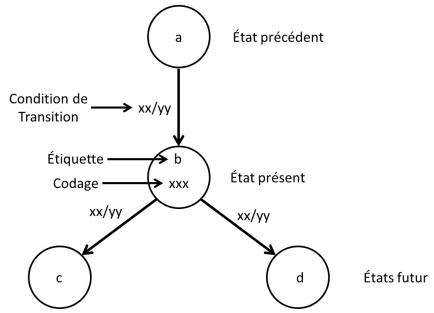


FIGURE 6.4 – Normalisation du graphe

Pour une machine de Moore, on fait l'économie d'écriture de la valeur de la sortie

pour chaque transition. Il suffit d'indiquer quand une sortie est affirmée à coté de l'état associé.

Exemple :

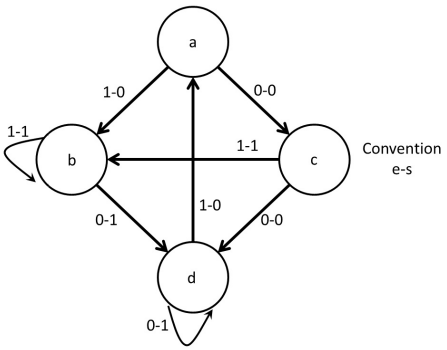


FIGURE 6.5 – Exemple de graphe

On peut reconnaitre une machine de Mealy sur la figure suivante. En effet, pour les états c et d, la sortie change en fonction de l'entrée, contrairement aux états a et b.

6.1.3 analyse d'un exemple

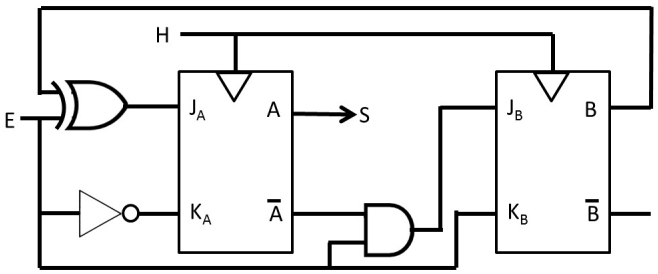


FIGURE 6.6 – Exemple de circuit synchrone

On extrait du schéma les équations suivantes :

$$A_{n+1} = J_A\bar{A} + \bar{K_A}A = E(A + \bar{B}) + \bar{E}\bar{A}B \tag{6.1}$$

$$B_{n+1} = J_B\bar{B} + \bar{K_B}B = E\bar{A}\bar{B} + \bar{E}B \tag{6.2}$$

On en déduit la table des états :

	Etat présent	Etat futur	
		$E = 0$	$E = 1$
E_0	00	00	11
E_1	01	11	00
E_2	10	00	10
E_3	11	01	10
	AB	AB	AB

que l'on peut représenter sous la forme d'un diagramme d'état.

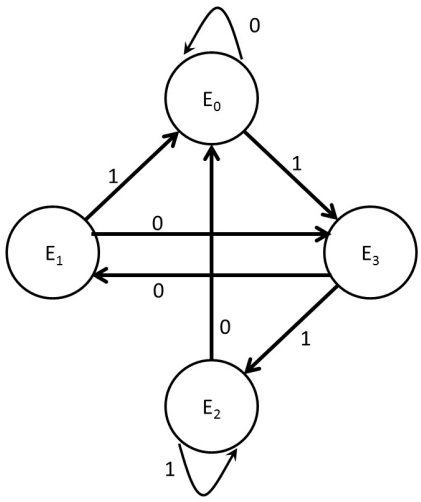


FIGURE 6.7 – Diagramme d'état

6.1.4 Conclusion

La méthode de synthèse d'une machine d'états finis synchrone consiste à :
- établir le diagramme d'état
- procéder au codage des états
- choisir le type des bascules
- établir les équations d'excitation de ces bascules en vue du schéma technologique.

6.2 Registres

Il s'agit d'association en parallèle de mémoires ou de bascules avec une horloge commune.

6.2.1 Latch ou mémoire tampon

L'élément de base est la bascule D. L'écriture se fait donc sur niveau de l'horloge (exemple de composant : 7475).

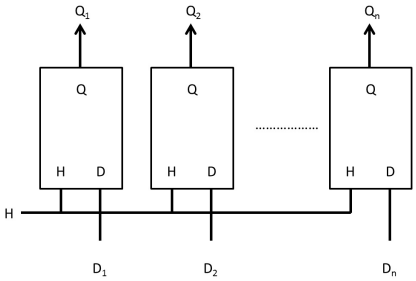


FIGURE 6.8 – Latch

6.2.2 Registre à décalage

A chaque front actif d'horloge, l'information mémorisée dans une cellule est transférée à l'élément de gauche (poids croissants) ou de droite (poids décroissants). Les éléments terminaux fournissent les entrées ou les sorties série.

Ceci ne peut être réalisé que par un circuit associant des bascules D en série (cf. figure 9). A chaque front actif de H, la sortie de chaque bascule recopie l'entrée de la bascule précédente. Le bon fonctionnement est assuré puisque le temps de propagation de chaque bascule est supérieur au temps de maintien des données.

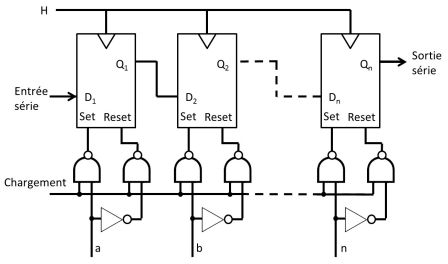


FIGURE 6.9 – Registre à décalage

Si les bascules constituantes du registre sont munies de remise à zéro et à 1 (synchrones ou asynchrones), il est intéressant et simple de créer la possibilité de chargement parallèle des données :

$$S = load.B \tag{6.3}$$

$$R = load.\bar{B} \tag{6.4}$$

où B représente un des bits du mot à charger. La possibilité de sortie parallèle n'est évidemment qu'une question de nombre de pattes disponibles du boitier.

Exemples : circuit 7475, 54198, 54195...

6.2.3 Compteurs synchrones

Définition

Ce sont les machines d'état les plus simples qu'il soit puisque leur nombre d'entrée est nul (mis à part l'horloge commune). Le compteur réalise une succession d'état de façon cyclique.

Le modulo de comptage est le nombre d'états utilisés : $0 < M \leq 2^n$ pour n bits. Le codage des états correspond au choix du nombre de bits : $n \geq log(M)/log(2)$. Il correspond aussi à l'assignement des variables d'états. Les codes les plus utilisés sont :

- Binaire pur ou naturel : 0 à 2^n.
- BCD (Binary Coded Decimal) pour la réalisation de décade. Il faut 4 bits pour 100, 8 pour 10000...
- Codes à décalage : La réalisation peut utiliser un registre à décalage. Cependant, ce genre de codage nécessite un assez grand nombre de bits par rapport au modulo. Le décalage du compteur peut par contre être grandement simplifié.
  - Exemple 1 : simple décalage : nécessite M bits pour un modulo M.
  - Exemple 2 : Compteur en annaux torsadés ou compteur Johnson : nécessite M/2 bits pour un compteur modulo M.
- Codes autocomplémentés à 9. Ils sont économiques pour la réalisation des décades réversibles (possibilité de comptage et de décomptage).
  - Exemple 1 : code Aïken : on prend les 5 premières et 5 dernières combinaisons de 4 bits.
  - Exemple 2 : code à excès de 3 : de manière générale n on enlève les (2^n-M)/2 états de début et de fin du code binaire pur.

Méthode de synthèse des compteurs synchrones

Une fois le modulo et le code de comptage établis, la méthode de synthèse consiste à l'écriture des équations d'excitation des bascules (états présents → états futurs), puis à identifier ces équations avec celles des bascules choisies comme éléments de base.

L'équation d'équation se présente sous la forme suivante :

$$Q_{n+1} = f_1 Q_n + f_2 \overline{Q_n} \tag{6.5}$$

Quant aux bascules courantes, elles ont pour équation :

$$\begin{aligned} Q_{n+1} &= D \\ Q_{n+1} &= J\bar{Q} + \bar{K}Q \\ Q_{n+1} &= T\bar{Q} + \bar{T}Q \end{aligned} \tag{6.6}$$

Le rapprochement de ces équations entraine les relations suivantes :

- $D = f_1 Q + f_2 \bar{Q}$
- $J = f_2$, $K = \overline{f_1}$, relation simple dans le cas d'une bascule JK. On a tout intérêt dans ce cas à simplifier J et K sur la table de Karnaugh de la fonction d'excitation. Pour cela, on partagera la table en 2 parties. Celle contenant $\bar{Q}$ permet de déduire J par regroupement des "1" , et K par le regroupements des "0".
- $T = \overline{f_1} Q + f_2 \bar{Q}$. Dans ce dernier cas, il faut regrouper les "1" de la partie $\bar{Q}$ et les "0" de la partie Q pour déduire T.

Compteur binaire pur

En portant les états futurs sur des tables de Karnaugh à quatre variables A, B, C, D par ordre décroissant (cf. figure 10), on obtient les équations suivantes :

$$\begin{aligned} A_{n+1} &= \bar{A}BCD + A\bar{C} + A\bar{D} + A\bar{B} \\ B_{n+1} &= \bar{B}CD + B(\bar{C} + \bar{D}) \\ C_{n+1} &= \bar{C}D + C\bar{D} \\ D_{n+1} &= D \end{aligned} \tag{6.7}$$

A_{n+1}

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

B_{n+1}

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

C_{n+1}

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

D_{n+1}

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

FIGURE 6.10 – Equation d'excitation d'un compteur binaire pur

La réalisation du compteur par 16 est particulièrement simple si on utilise des bascules T.

$$
\begin{aligned}
T_D &= 1 \\
T_C &= D \\
T_B &= CD \\
T_A &= BCD
\end{aligned}
\tag{6.8}
$$

La solution apparait comme évidente à posteriori et récurrente pour des modulos supérieurs.

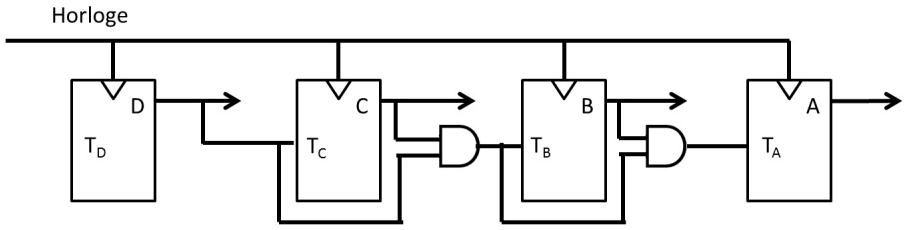


FIGURE 6.11 – compteur binaire pur modulo 16

Compteur BCD

Les 4 tables de Karnaugh des états futurs permettent d'établir les équations suivantes :

$$
\begin{aligned}
A_{n+1} &= BCD + A\bar{D} \\
B_{n+1} &= \bar{B}CD + B(\bar{C} + \bar{D}) \\
C_{n+1} &= \bar{A}\bar{C}D + C\bar{D} \\
D_{n+1} &= \bar{D}
\end{aligned}
\tag{6.9}
$$

L'utilisation des bascules JK conduit aux équations suivantes :

$$
\begin{aligned}
J_A &= BCD, & K_A &= D \\
J_B &= CD, & K_B &= CD \\
J_C &= \bar{A}D, & K_C &= D \\
J_D &= 1, & K_D &= 1
\end{aligned}
\tag{6.10}
$$

Pour une conception parfaite, il faut vérifier que les états non utilisés (de 10 à 15) ne conduisent pas à des cycles erronés mais au contraire permettrait au contraire une réinitialisation automatique du cycle normal. On vérifie cela sur les tables de Karnaugh.

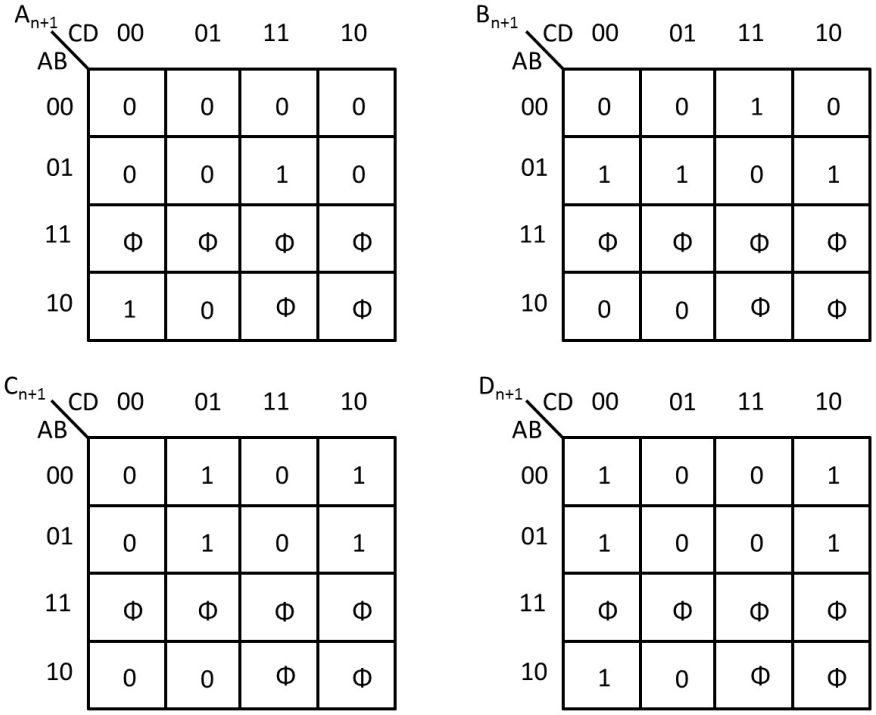


FIGURE 6.12 – Equation d'excitation d'un compteur BCD

Compteurs réversibles

Ce sont des registres capables d'opération d'incrémentation (comptage) et de décrémentation (décomptage). Ces deux opérations nécessitent soit deux horloges séparées soit une horloge et une commande de multiplexage des réseaux combinatoires assurant l'une ou l'autre fonction.

Pour en faire la synthèse, on commence par calculer pour chaque bascule la fonction comptage, puis la fonction de décomptage. Il suffit ensuite d'appliquer à l'entrée de chaque bascule une fonction de type :

$$Entree = \overline{UD} \cdot Comptage + UD \cdot Decomptage \tag{6.11}$$

UD (UP-Down) est le bit de commande du multiplexeur. Pour exemple, se reporter aux circuit 74190 et 74191.

Compteurs programmables

Comme on l'a vu pour des registres à décalages, il est très intéressant de pouvoir initialiser un compteur (ou un décompteur) par une commande de chargement, associée à un ensemble de bits en parallèles sur chaque bascule. Ceci est en général réalisé par action sur les entrées Set et Reset des bascules.

Mise en cascade de plusieurs étages - Reports

Les circuits intégrés on un nombre de bascules limité. De même, il n'est pas raisonnable de calculer un compteur à chaque changement de modulo. On peut facilement rendre modulaire un compteur 4 bits en lui adjoignant des entrées-sorties de report. Ainsi avec un compteur 4 bits, on pourra réaliser des compteurs 8, 12, 16 bits...

Le report correspond à la dernière position de comptage (ou de décomptage) du module.

Troisième partie

Introduction GRAFCET

ELEMENTS DE COURS SUR LE GRAFCET

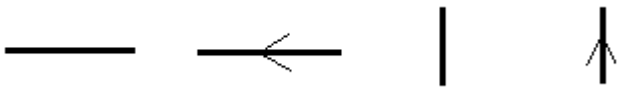
I - Introduction

Le Grafcet est un outil graphique de description des systèmes automatiques séquentiels. C'est un langage graphique clair, strict et sans ambiguïté, permettant par exemple au réalisateur de montrer au donneur d'ordre comment il a compris le cahier des charges. Langage graphique universel, indépendant (dans un premier temps) de la réalisation pratique, le grafcet permet de réaliser la partie commande d'un système automatique.

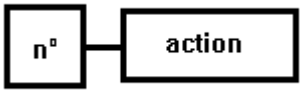
II - Définitions

Un Grafcet est composé d'étapes, de transitions et de liaisons.

Une LIAISON est un arc orienté (ne peut être parcouru que dans un sens). A une extrémité d'une liaison il y a UNE (et une seule) étape, à l'autre UNE transition. On la représente par un trait plein rectiligne, vertical ou horizontal. Une verticale est parcourue de haut en bas, sinon il faut le préciser par une flèche. Une horizontale est parcourue de gauche à droite, sinon il faut le préciser par une flèche.

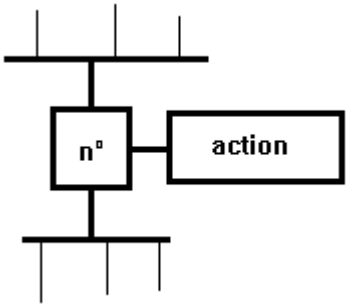


Une ETAPE correspond à une phase durant laquelle on effectue une ACTION pendant une certaine DUREE (même faible mais jamais nulle). L'action doit être stable, c'est-à-dire que l'on fait la même chose pendant toute la durée de l'étape, mais la notion d'action est assez large, en particulier composition de plusieurs actions, ou à l'opposé l'inaction (étape dite d'attente).



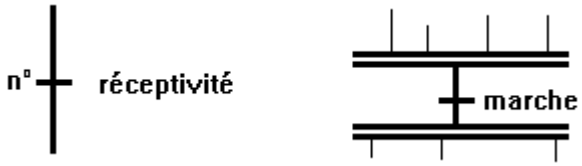
On représente chaque étape par un carré, l'action est représentée dans un rectangle à gauche, l'entrée se fait par le haut et la sortie par le bas. On numérote chaque étape par un entier positif, mais pas nécessairement croissant par pas de 1, il faut simplement que jamais deux étapes différentes n'aient le même numéro.

Si plusieurs liaisons arrivent sur une étape, pour plus de clarté on les fait arriver sur une barre horizontale, de même pour plusieurs liaisons partant de l'étape. Cette barre horizontale n'est pas une nouvelle entité du Grafcet, elle fait partie de l'étape, et ne représente qu'un "agrandissement" de la face supérieure (ou inférieure) de l'étape.



Une étape est dite active lorsqu'elle correspond à une phase "en fonctionnement", c'est-à-dire qu'elle effectue l'action qui lui est associée. On représente quelquefois une étape active à un instant donné en dessinant un point à l'intérieur.

Une TRANSITION est une condition de passage d'une étape à une autre. La condition est définie par une RECEPTIVITE qui est généralement une expression booléenne (c.-à.-d avec des ET et des OU) de l'état des CAPTEURS : elle ne peut être qu'une fonction logique (1 = Vrai ou 0 = Faux), sans notion de durée.



On représente une transition par un petit trait horizontal sur une liaison verticale. On note à droite la réceptivité. Dans le cas de plusieurs liaisons arrivant sur une transition, on les fait converger sur une grande double barre horizontale, qui n'est qu'une représentation du dessus de la transition. De même pour plusieurs liaisons partant sous une transition.

III - Règles d'évolution du grafcet

La modification de l'état de l'automatisme est appelée évolution, et est régie par 5 règles :

R1 : Les étapes INITIALES sont celles qui sont actives au début du fonctionnement. On les représente en doublant les côtés des symboles. On appelle début du fonctionnement le moment où le système n'a pas besoin de se souvenir de ce qui s'est passé auparavant (allumage du système, bouton RAZ ou "reset",...). Les étapes initiales sont souvent des étapes d'attente pour ne pas effectuer une action dangereuse par exemple à la fin d'une panne de secteur.

R2 : Une TRANSITION est soit validée, soit non validée . Elle est validée lorsque toutes les étapes immédiatement précédentes sont actives (toutes celles reliées directement à la double barre supérieure de la transition). Elle ne peut être FRANCHIE que lorsqu'elle est validée **et** que sa réceptivité est vraie. Elle est alors obligatoirement franchie.

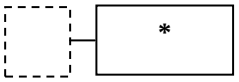
R3 : Le FRANCHISSEMENT d'une transition entraîne l'activation de TOUTES les étapes immédiatement suivantes et la désactivation de TOUTES les étapes immédiatement précédentes (TOUTES se limitant à 1 s'il n'y a pas de double barre).

R4 : Plusieurs transitions SIMULTANEMENT franchissables sont simultanément franchies (ou du moins toutes franchies dans un laps de temps négligeable pour le fonctionnement). La durée limite dépend du "temps de réponse" nécessaire à l'application (très différent entre un système de poursuite de missile et une ouverture de serre quand le soleil est suffisant).

R5 : Si une étape doit être à la fois activée et désactivée, elle RESTE active. Une temporisation ou un compteur actionnés par cette étape ne seraient pas réinitialisés. Cette règle est prévue pour lever toute ambiguïté dans certains cas particuliers qui pourraient se présenter : simultanéité.

IV - Les actions : cas particuliers

Action continue :

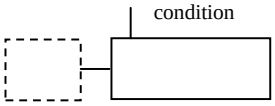


action continue avec libellé d'assignation

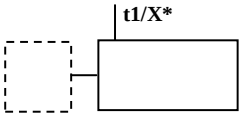
Si on souhaite qu'une action ait lieu de façon continue pendant plusieurs étapes successives, il suffit de répéter cette action à chacune des étapes concernées.

Action conditionnelle :

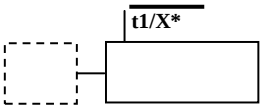
On peut faire en sorte qu'une action soit conditionnée par une variable (condition) logique. On la représente alors en ajoutant une barre verticale sur la partie supérieure du cadre, avec la condition logique.



Action temporisée :
La condition logique associée à une action peut être une temporisation, ce qui permet soit de limiter le temps pendant lequel elle est réalisée, soit de retarder son démarrage. Par exemple, une temporisation de durée 12 secondes lancée à l'étape no. 7 sera notée 12s / X7.



action retardée

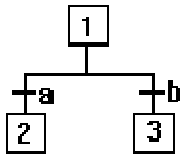


action limitée dans le temps

Remarque : Si on souhaite qu'une action démarre dès qu'une étape i est active, mais qu'elle ne soit réalisée que pendant une durée d donnée, il suffit d'affecter à la transition qui suit cette étape une réceptivité d / Xi (dans ce cas on n'utilise pas d'action conditionnelle).

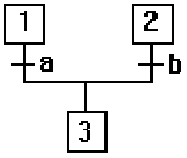
V - Configurations courantes

Divergence en OU :



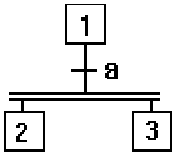
Si 1 active et si a seul, alors désactivation de 1 et activation de 2, 3 inchangé.
Si a et b puis 1 active alors désactivation 1, activation 2 et 3 quel que soit leur état précédent. (règle 4). Pour éviter toute ambiguïté, on fait en sorte que a et b ne puissent pas simultanément être à 1 : soit ceci est physiquement impossible, sinon il faut l'imposer en prenant pour les deux réceptivités des fonctions logiques qui s'excluent (par exemple a et $\bar{b}a$).

Convergence en OU :



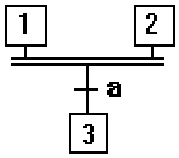
Si 1 active et a sans b, alors activation de 3 et désactivation de 1, 2 reste inchangé
Si 1 et 2 et a et b alors 3 seule active
On appelle BARRE DE OU la barre symbolisant les entrées / sorties multiples d'étapes.

Divergence en ET :



Si 1 active et si a, alors désactivation de 1 et activation de 2 et 3.

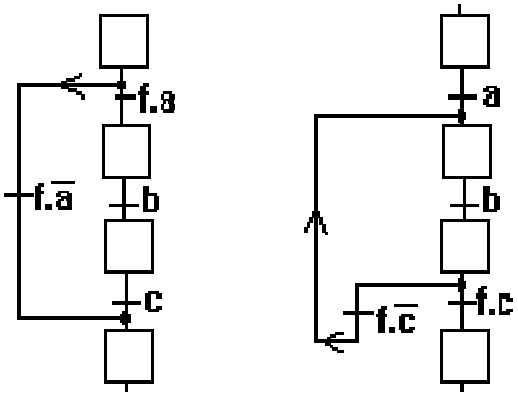
Convergence en ET :



Si 1 active seule et a alors aucun changement. Si 1 et 2 et a, alors activation de 3 et désactivation de 1 et 2. En pratique, on place généralement une étape d'attente (sans action) à la fin de chaque séquence précédant une convergence en ET, et on affecte une réceptivité toujours vraie (=1) à la transition qui suit la convergence.

On appelle couramment BARRE DE ET la double barre, mais attention ce n'est pas une entité à part, mais une partie d'une transition.

Détaillons également le saut avant (si a alors ...) et les boucles (répéter ... jusqu'à c). Ce sont les deux seules possibilités avec des OU: il ne peut y avoir de divergence en « «ou » après une transition.



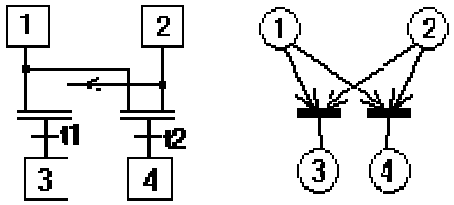
Saut de séquence Reprise de séquence

VI Exercices :

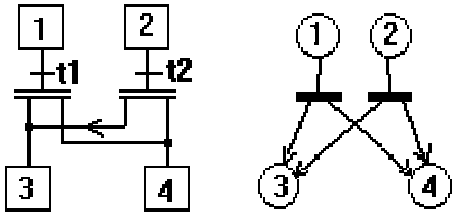
A- Quelques configurations plus complexes.

1- soient 4 étapes 1 à 4 et deux transitions de réceptivité t1 et t2. Construire la portion de Grafcet réalisant : quand 1 ET 2 actifs alors
si t1 passer en 3 (et désactiver 1 et 2),
si t2 passer en 4 (et désactiver 1 et 2),
sinon rester en 1 et 2

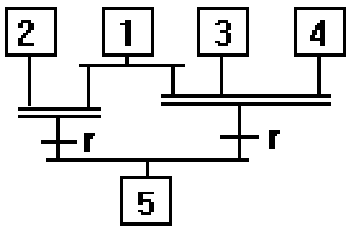
On trouve la solution facilement en analysant les cas d'évolution (quand franchit-on t1 ?). Il faut souligner que l'ajout d'une étape intermédiaire n'est pas une bonne solution car tout passage d'une étape dure un laps de temps (donc discontinuité sur les sorties = aléa technologique).



2 - Problème du même ordre : Quand (étape 1 et t1) OU (étape 2 et t2) alors passer en 3 ET 4:



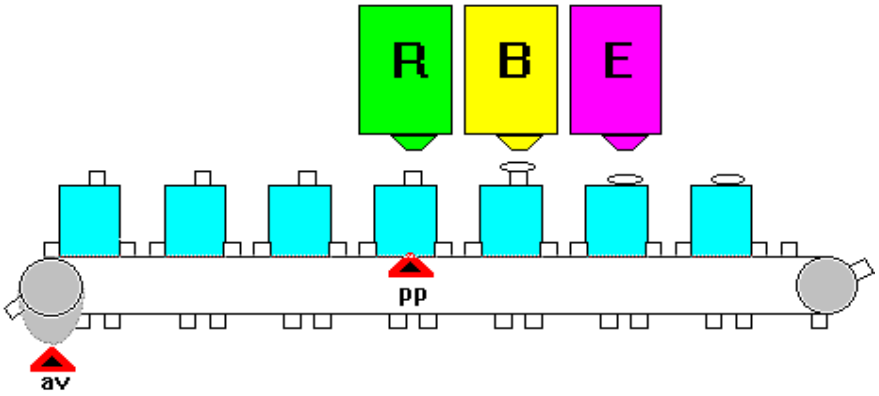
3 - si {étape 1 et [étape 2 ou (étapes 3 et 4)]} et réceptivité r alors activer l'étape 5 (et désactiver les autres).



B - travail à la chaîne

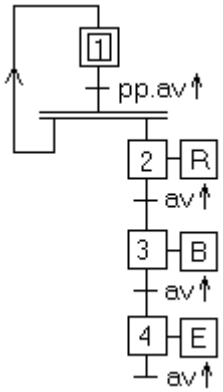
Soit une chaîne de remplissage de bidons d'huile. Un tapis roulant se déplaçant par saccades (cadencé par un système supposé externe à notre Grafcet, s'arrêtant à chaque nouvel appui de la came sur le capteur av) est alimenté manuellement (de temps en temps il manque des bidons). Trois postes sont prévus : remplissage (R), bouchage (B) et enfoncement (E).

Schéma du dispositif



Un seul capteur détecte la présence d'un bidon en début de chaîne : pp. On désire faire les 3 opérations simultanément, sauf s'il n'y a pas de bidon sous le poste. On suppose que le tapis est vide lors de l'initialisation.

Grafcet :



L'étape 1 est constamment active. La dernière transition est appelée "transition puits", mais il était possible de la relier à l'étape 1. En fonctionnement normal, toutes les étapes du Grafcet sont actives. Du point de vue commande, chaque opération comportera plusieurs étapes (R = descendre l'entonnoir, ouvrir le robinet,...) dont une seule sera active à la fois). Chaque activation représente un bidon dans le circuit.

Cette méthode utilise au mieux le séquencement du Grafcet, on peut maintenant rajouter des capteurs, mais qui n'auront pour fonction que de vérifier le bon fonctionnement du système. Dans tous les cas similaires, on utilisera cette démarche : faire le Grafcet pour une pièce seule, puis le modifier pour gérer l'ensemble des pièces, en vérifiant bien que jamais une même étape ne corresponde à 2 pièces, on décompose donc le système en tronçons et on ne laisse entrer dans un tronçon que s'il est libre. Exemples : atelier flexible (on suit la pièce pour chaque opération jusqu'au produit fini), montage (monter 2 pièces ensemble correspond à une convergence en ET : de 2 étapes actives on arrive à 1), chariots filo-guidés (si un tronçon est occupé, essayer de le contourner par une voie libre)...

VII- Traduction du grafcet en programme exécutable sur API :

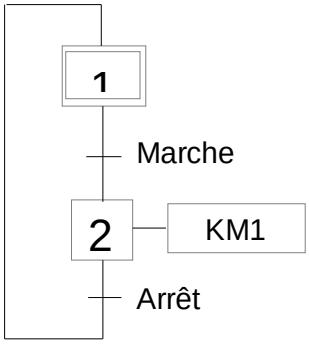
Une fois le grafcet de description du système réalisé, il faut pouvoir le traduire en programme sur Automate Programmable Industriel, quel que soit la marque et le type de celle-ci.
Les langages de programmation pour automates et systèmes programmables sont :

- deux langages littéraux : **IL** (Instruction List), c'est un langage de très bas niveau, comparable à l'assembleur et qui convient bien pour décrire de petites routines optimisées ; **ST (**Structured Text), c'est un langage structuré de haut niveau qui ressemble au Pascal avec des opérateurs de type Repeat, Until, For, While, If then, Else… ;
- trois langages graphiques : **LD** (Ladder Diagram), c'est le schéma à relais, langage d'origine des automaticiens ; **FBD** (Function Block Diagram), c'est un langage graphique où les opérateurs ou les fonctions sont représentés par des blocs reliés entre eux comme un schéma électronique et **SFC** (Sequentiel Fonction Chart) diagramme fonctionnel en séquences.

La norme CEI 61131-3 de 1993 définit ces langages, en termes de symboles, sémantique, variables, etc...

Le langage Ladder est commun à tous les API et est proche du schéma électrique, ce qui rend son utilisation aisé pour les automaticiens débutants ou confirmés.

Exemple :
Soit le grafcet suivant qui représente la mise en fonctionnement d'un moteur, par l'intermédiaire de son contacteur KM1 :



- La première étape consiste à affecter à chaque variable du grafcet une entrée ou sortie :
- On affecte également aux étapes des bits internes : Etape1->M1, Etape2->M2.
- On écrit les conditions de validation de chaque étape :

M1=
M2=

- On écrit enfin les équations des sorties :

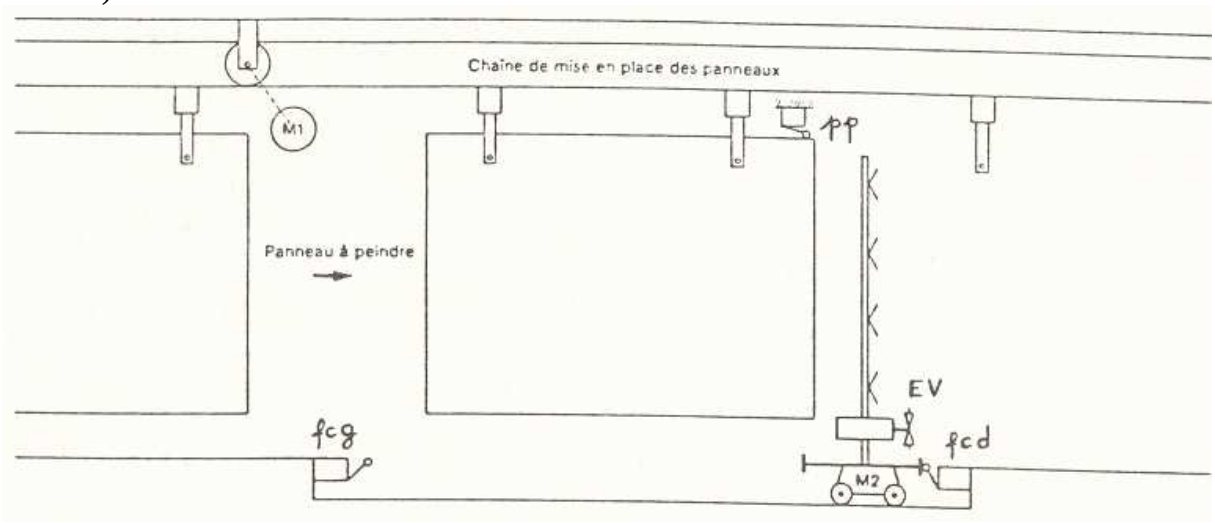
O1=
O2=

- On schématise le programme en langage Ladder :

TD N°6

Exercice 1 – Poste automatique de peinture

Il s'agit d'automatiser un poste de travail destiné à peindre des panneaux accrochés par des pattes de fixation à une chaîne de mise en place qui assure leur translation (cf. figure ci-dessous).



La chaîne de mise en place des panneaux est actionnée par un moteur électrique M1. Un chariot automoteur, actionné par un moteur électrique M2 à 2 sens de rotation (D et G), supporte un ensemble de peinture à 4 buses alimentées par une électrovanne EV.
Les positions extrêmes du chariot automoteur dans le poste de peinture sont repérés par les contacts de fin de courses fcg et fcd. La présence d'un panneau au poste de peinture est détectée par un capteur p.
Pour démarrer le cycle de peinture, il faut :

- qu'un panneau soit présent au poste de peinture,
- que le chariot automoteur soit à droite,
- appuyer sur le bouton « dcy ».

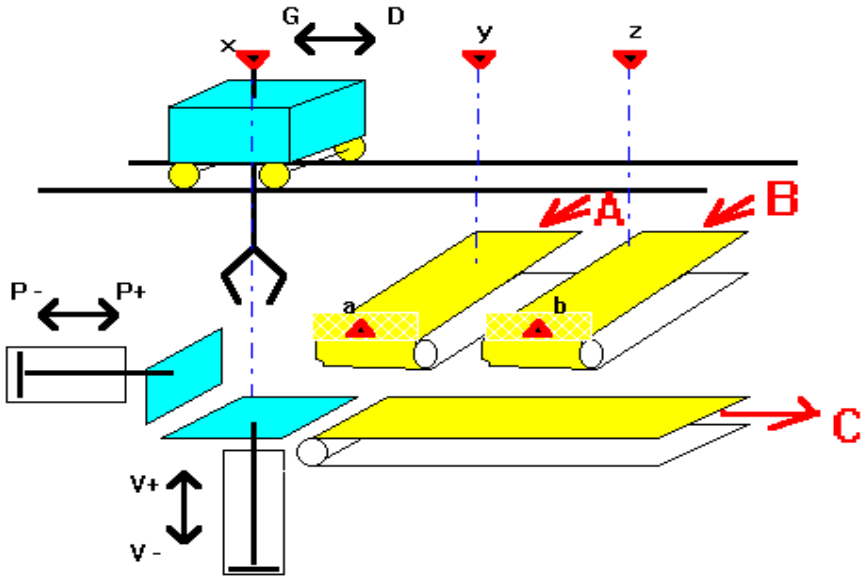
S'il n'y a pas de panneau ou si le chariot n'est pas en place, la commande dcy permet dans un 1er temps d'initialiser l'installation dans la position ci-dessus. Si un panneau est présent et le chariot automoteur est à droite, les buses commencent à projeter de la peinture. Cinq secondes après, le chariot part à gauche, va en fin de courses à gauche, puis revient à droite jusqu'à sa position initiale. Les buses arrêtent alors la projection de peinture et le panneau qui se trouve peint est évacué par la chaîne de mise en place jusqu'à ce qu'il soit sorti du poste de peinture.

a) Après lecture détaillée du sujet, établir la liste des entrées (capteurs, boutons…), puis celle des sorties (actions).
b) Etablir le GRAFCET.
c) Ecrire les équations des actions.
d) Modifier le GRAFCET de façon à avoir 2 modes de fonctionnement possibles (par un bouton « cpc ») : cpc = 1 → fonctionnement cycle par cycle, c-a-d qu'il faut appuyer sur dcy à chaque nouveau panneau ; cpc = 0 → fonctionnement cycles répétés.
e) On ajoute un bouton d'arrêt d'urgence (au). Que doit-on modifier ?

Exercice 2 – priorité

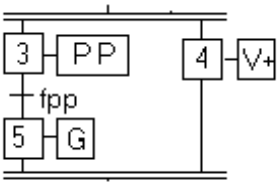
Soit un chariot se déplaçant sur deux rails (action D vers la droite, G vers la gauche). Il comporte une pince pouvant prendre une pièce (PP, fin quand fpp) s'il se trouve sur le tapis A (capteur y) et qu'une pièce est présente (capteur a) (idem en z si b). Puis il retourne en v, pose la pièce (action DP, fin quand fdp) sur le plateau qui aura été préalablement placé en position haute (fv+). Celui-ci descend (V-, jusqu'à fv-), un second vérin pousse la pièce (P+, fin quand fp+), puis le pousseur recule en fp-. Le tapis de sortie C est supposé toujours en mouvement. Les tapis A et B sont commandés par des systèmes non traités ici.

Schéma du dispositif



a) Effectuer d'abord un grafcet linéaire comprenant une seule voie d'arrivée A. On vérifiera que le système est dans son état initial (poussoir reculé, plateau en bas et chariot en v) avant de lancer le cycle.

b) L'améliorer en prévoyant le retour de l'actionneur « V+ », en temps masqué, en même temps que « PP puis G ».



c) Prévoir deux tapis d'alimentation A et B (en cas de pièces en a ET b, prendre celle en a). Exprimer l'équation de la réceptivité, après l'étape 2, qui permet de réaliser ce fonctionnement.

d) Modifier cette réceptivité pour prévoir une priorité tournante (en cas de conflit, prendre la voie qui n'a pas été servie la fois précédente) attention, si plusieurs pièces arrivent sur la même voie et aucune sur l'autre, ne pas bloquer le système. On réalisera un Grafcet séparé pour gérer les priorités :