efrei PARIS SUD Ingénieurs en mouvement | esigetel PARIS SUD École d'ingénieur des sciences du numérique

Cours Information numérique

L1, 2015-2016

Isabelle Sirot

Chapitre 1 : La représentation de l'information : codage numérique

La plupart des phénomènes que nous percevons apparaissent comme analogique, ils varient de façon continue. Par exemple quand le soleil chauffe, nous percevons que la température augmente progressivement de façon continue. Un signal numérique est discontinu et passe par des valeurs fixes quantifiées. Quand nous appuyons sur un interrupteur, la lumière est soit allumée soit éteinte. Ils sont utilisés dans les ordinateurs et dans les circuits d'électronique numérique

Les signaux numériques présentes de nombreux avantages, ils sont plus faciles à utiliser et à traiter, ils prennent moins de place et peuvent être compressés en supprimant des informations redondantes. Ils résistent mieux aux signaux ou bruits parasites

Le passage d'un monde à un autre se fait très facilement avec des Convertisseurs Numérique Analogique (CNA) et Analogique Numérique (CAN).

1) Les systèmes de numération

Nous comptons en base 10 ; nous utilisons 10 symboles (0,…,9). Le nombre de symbole définit la base de numération.

En informatique ou en électronique numérique:
Nous écrivons en base 2 : base binaire ; il y a donc deux symboles, deux chiffres (0, 1).
La base octale (base 8) a huit symboles (0,..,7).
La base hexadécimal (base 16) a seize symboles (0…, A, B, C, D, E, F) où A =10, B=11, C=12, D=13, E=14, F=15.

Selon un vieux traité, le binaire aurait été utilisé la première fois en l'an trois mille avant notre ère sur l'ordre de l'empereur Fou-Hi. Le yin et le yang chinois en seraient la première manifestation, le yang correspondrait à 1, le yin à 0. En 1650 avant notre ère, les Egyptiens découvrent qu'une progression binaire par doublement successifs permet d'effectuer des multiplications.

En Europe, le plus ancien document binaire découvert date de 1600, il est du à un astronome géographe anglais Thomas Hariot, c'est une table de valeur binaire. Il faut attendre Leibnitz (1646-1716) pour imaginer le binaire comme mode de calcul mais cette théorie reste sans applications.

A partir 1930, on essaye d'imaginer des dispositifs de calculs plus simples. De nombreux théoriciens français anglais et américains travaillent sur le calcul binaire. Les plus connus sont l'américain Claude Shannon et l'Anglais Alan Turing.

Dans n'importe quel système de numération, Lorsque nous écrivons un nombre, la position des chiffres détermine leurs poids. Nous utilisons la forme polynomiale :

$$N_B = \left(a_{n-1}a_{n-2}\ldots a_1 a_0 a_{-1}a_{-2}\right)_B$$
$$a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \ldots + a_1B^1 + a_0B^0 + a_{-1}B^{-1}a_{-2}B^{-2} + \ldots.$$

Où B est la base utilisée et ($a_{n-1}\ldots a_1\ a_0\ a_{-1}a_{-2}$) les chiffres du nombre N.

Par exemple 1789 = 1*1000+7*100+8*10+9*1
1789= $1*10^3+7*10^2+8*10^1+9*10^0$

2) Conversion d'un système de numération en un autre

2.1) Passage d'une base B vers la base 10

Nous utilisons la forme polynomiale.

Exemple (passage de la base 2 à la base 10)

$(1011011,11)_2 = 1*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2}$
$= 64 + 16 + 8 + 2 + 1 + 0,5 + 0,25 = 91,75$

$(1110011111,011100)_2 = 2^9 + 2^8 + 2^7 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + 2^{-2} + 2^{-3} + 2^{-4}$
$= 927 + (1/4 + 1/8 + 1/16) = 927,4375$

2.2) Passage de la base 10 vers une base B

Il existe deux méthodes.

2.1.1 Méthode des divisions et multiplications successives

Pour la partie entière du nombre, le passage de la base 10 à la base 2 se fait par divisions successives par deux jusqu'à ce que le quotient soit nul, nous écrivons les restes dans l'ordre inverse où ils ont été obtenus. Pour la partie fractionnaire, nous effectuons des multiplications par deux autant de fois que cela est nécessaire pour obtenir la précision voulue ou pour que la conversion tombe juste.

Exemple : 45 = $(101101)_2$

45 / 2= 22 reste 1
22 / 2 =11 reste 0
11 / 2 = 5 reste 1
5 / 2 = 2 reste 1
2 / 2 =1 reste 0
1 / 2 = 0 reste 1

↑ Lecture

Exemple : 0,40 = 0,0110011

Pour obtenir 0,40 avec une précision qui approche celle de la base 10, il faut multiplier 7 fois pas deux car

$2^{-6} > 1/100 > 2^{-7}$

$2^{-7} = 1/128 \approx 1/100$

0,4 *2 = **0**,8	0
0,8*2 = **1**,6 (1,6-1)	1
0,6 *2 = **1**,2 (1,2-1)	1
0,2*2 = **0**,4	0
0 ,4 *2 = **0**,8	0
0,8*2 = **1**,6 (1,6-1)	1
0,6 *2 = **1**,2 (1,2-1)	1

↓ Lecture

45, 40 = 101101,0110011

2.1.2 Méthode des puissances de B

Nous soustrayions successivement la plus grande puissance de B. Cette méthode avec un peu d'entraînement est beaucoup plus rapide que la première. Elle est toutefois plus difficile à mettre en œuvre notamment quand nous avons des nombres fractionnaires compliqués.

Exemple (Passage de la base 10 à la base 2)

$38,125 = 32 + 4 + 2 + 0,125 = 2^5 + 2^2 + 2^1 + 2^{-3} = 100110,001$

2.3) Passage de la base 2 vers une base de forme contractée de la base 2 et Inversement

Les bases 8 et 16 sont en fait des formes contractées de la base 2 car 8= 2^3 et 16= 2^4. Le passage de la base 2 à la base 8 se fait immédiatement en groupant les chiffres binaires par 3. Le passage de la base 2 à la base 16 en groupant les chiffres binaires par 4.

Exemple (Passage de base 2 à la base 8 et à la base 16)

$(101100)_2$= (101 100)$_2$= $(54)_8$
$(1010011100)_2$= (0010 1001 1100)$_2$ =$(29C)_{16}$

Inversement, chaque chiffre écrit en base 8 ou (base 16) est exprimé sur 3 (ou 4) chiffres binaires.

Exemple : (Passage de base 8 et la base 16 à la base 2)

$(56)_8$= (101 110)$_2$=$(101110)_2$
$(A,CA)_{16}$= (1010, 1100 1010)$_2$= $(1010,11001010)_2$

3) Représentation des nombres signés

Les nombres doivent respecter un certain format, ils doivent être écrits sur n bits. Il existe deux représentations différentes.

3.1) Représentation module plus signe

L'élément binaire le plus à gauche, le bit de poids fort du nombre MSB (Most Significant Bit), représente le signe. Il est à zéro si le nombre est positif. Il est à 1 si le nombre est négatif.

1.1.1.1.1.1.1.1

1.1.1.1.1.1.1.2 Exemple

Signe	Valeur absolue	Nombre signé sur 8 bits	Nombre décimal
1	101	10000101	-5
1	1000	10001000	-8
0	101	00000101	5
0	1000	00001000	8

Dans cette représentation, il est simple de connaître si un nombre est positif ou négatif, il suffit de regarder le bit le plus à gauche. Il est également simple de

changer le signe du nombre. Par contre il est impossible d'effectuer des opérations d'addition ou de soustraction, c'est pour cela que cette représentation est peu utilisée et que nous préférons la représentation suivante.

3.2) Représentation en complément à deux

C'est la représentation standard dans les ordinateurs pour exprimer les nombres négatifs.

La démonstration faite en cours ne sera pas reprise dans ce document.

Le passage d'un entier positif à un entier négatif (et inversement) se fait en complémentant le nombre bit à bit puis en lui ajoutant 1. Le codage en complément à deux n'est possible que si on définit un format pour les nombres, ce format doit être respecté pour chaque calcul et représentation des nombres.

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

Nous définissons des nombres sur n bits. sur lesquels est fait le codage. La représentation en complément à deux est possible car nous complémentons sa valeur à 2^n. La représentation du nombre se fait modulo 2^{n-1} et une retenue non contenue dans les n bits est ignorée.

Nombre décimal	Nombre représenté en complément à deux sur huit bits
-5	11111011
-8	11111000
+5	00000101
+8	00001000

Le nombre N en complément à deux ne doit pas sortir de l'intervalle

$$-2^{n-1} \leq N \leq 2^{n-1} - 1$$

Pour un nombre sur 16 bits, tout nombre entier doit être contenu dans l'intervalle.

$$(-32768)_{10} \leq N \leq (32767)_{10}$$

3.2) Arithmétique binaire

Les circuits doivent travailler avec des nombres qui ont toujours le même format. Si nous manipulons des nombres trop grands pour le format, nous obtenons un dépassement de capacité, le résultat est faux.

Les mêmes règles de calcul s'appliquent dans tous les systèmes de numération.

Exemple

	00111101	$(45)_{16}$
+	00110100	$+(57)_{16}$
	01110001	$(9C)_{16}$

Nous ramenons la soustraction à l'addition d'un nombre négatif représenté en complément à deux.

Exemple

Soit les nombres suivants exprimés en base 10 sur 8 bits en notation en complément à deux.

$+45 = (00101101)_2$
$-45 = (11010011)_2$
$+22 = (00010110)_2$
$-22 = (11101010)_2$

On effectue les opérations suivantes :

$45+22 = (00101101)_2 + (00010110)_2 = (01000011)_2$
$(01000011)_2 = 67$

$45-22 = (00101101)_2 + (11101010)_2 = (00010111)_2$
$(00010111)_2 = 23$

122+45 : hors format sur 8 bits le nombre maximum représentable est +127

$-45-22 = (11010011)_2 + (11101010)_2 = (10111101)_2$
$(10111101)_2 = -67$

Chapitre 2 : Algèbre de Boole

Georges Boole, philosophe et mathématicien anglais du 19ème siècle est l'auteur d'une théorie sur l'art de construire un raisonnement logique au moyen de propositions qui ont une seule réponse OUI (VRAI) ou NON (FAUX). L'ensemble des opérations formelles appliquées à ces propositions forme une structure mathématique appelée algèbre de Boole. A son époque, il s'agissait de développement purement théorique car on ignorait l'importance qu'allait prendre cette algèbre avec l'informatique. Les concepts de la logique booléenne ont été ensuite appliqués aux circuits électroniques par Claude Shannon (1916-2001).

Cette algèbre est en effet applicable à l'étude des systèmes possédant deux états s'excluant mutuellement. Nous associons OUI à 1 et NON à zéro ou vice versa. L'algèbre binaire qui en découle est à la base de la mise en œuvre de tous les Systèmes Numériques (Ordinateurs, Système de communication numérique) qui travaillent à partir de quelques opérations élémentaires répétées un grand nombre de fois.

1) Algèbre Binaire

C'est un cas particulier de l'algèbre de Boole. Nous travaillions sur un ensemble E qui ne comporte que deux éléments : 0 et 1.

1.1) La fonction NON, complément

Il affecte à la variable de sortie l'état complémentaire de la variable d'entrée.

$F(x) = \bar{x}$

x	F(x)
0	1
1	0

1.2) La fonction OU, somme logique

L'opérateur OU représente la somme logique (ne pas confondre avec la somme arithmétique) ou l'union logique. Sa table de vérité pour deux variables est la suivante.

F(A, B) = A+B

A	B	A.B
0	0	0
0	1	1
1	0	1
1	1	1

1.3) La fonction ET , produit logique

L'opérateur ET représente le produit logique ou l'intersection logique. Sa table de vérité pour deux variables est la suivante.

$F(A,B) = A.B$

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

2) Propriétés et théorèmes

- Commutativité
  A+B = B+A
  A.B=B.A

- Associativité
  A+(B+C)=(A+B)+C=A+B+C
  A.(B.C)= (A.B).C=A.B.C

- Éléments neutres
  A+0=A
  A.1=A

- Double distributivité
  A.(B+C)=AB+AC
  A+(B.C)=(A+B).(A+C)

- Idempotence
  A.A.A…..A=A

A+A+A+….+A=A

- Propriétés de la fonction complément

$$\overline{\overline{A}} = A$$
$$A + \overline{A} = 1$$
$$A \cdot \overline{A} = 0$$

- Absorption

A+A.B = A

A.(A+B) = A

- Absorption du complément

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

- Théorème des consensus

$$A.B + \overline{A}.C + B.C = A.B + \overline{A}.C$$
$$(A+B).(\overline{A}+C).(B+C) = (A+B).(\overline{A}+C)$$

- Principe de dualité

Toute expression logique vraie demeure vraie si on remplace les + par des . , les 0 par des 1 et les 1 par des 0.

- ***Théorème de Morgan***

Pour deux variables A et B

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$
$$\overline{A.B} = \overline{A} + \overline{B}$$

Cette propriété est généralisable à n variables.

Exemple de simplifications de fonctions par la méthode algébrique :

$$S_1 = (A + \overline{B})(\overline{A} + B)(\overline{A} + \overline{B})$$
$$S_1 = (A+\overline{B})(\overline{A}+B)(\overline{A}+\overline{B}) = (A.\overline{A}+\overline{B}.\overline{A}+A.B+\overline{B}.B).(\overline{A}+\overline{B}) = (0+\overline{B}.\overline{A}+A.B+0).(\overline{A}+\overline{B})$$
$$= (\overline{B}.\overline{A}+A.B).(\overline{A}+\overline{B}) = \overline{B}.\overline{A}.\overline{A}+A.B.\overline{A}+\overline{B}.\overline{A}.\overline{B}+A.B.\overline{B} = \overline{B}.0+A.\overline{A}.B+\overline{A}.\overline{B}.\overline{B}+A.0$$
$$= A.\overline{A}.B+\overline{A}.\overline{B} = 0.B+\overline{A}.\overline{B} = \overline{A}.\overline{B}$$

3) Fonctions logiques

3.1 Définition

Une fonction logique Booléenne se présente comme une association d'opérations algébriques sur un ensemble de variables logiques. Une fonction logique Booléenne peut se présenter comme une association de sommes et de produits logiques.

Si l'expression est une somme de produits la forme est dite **disjonctive**.

$$f(A,B,C) = \overline{A}BC + A\,\overline{C} + A\,\overline{B}C$$

Si l'expression est un produit de sommes la forme est dite **conjonctive**.

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

Une fonction est dite sous forme **normale** ou **canonique** si chaque terme contient toutes les variables. Par exemple

$$f(A,B,C) = \overline{A}BC + A\,\overline{C} + A\,\overline{B}C$$

Cette expression n'est pas une forme canonique car le deuxième terme ne contient pas la variable B. Pour obtenir la forme canonique, nous multiplions le terme $A\overline{C}$ par $(B+\overline{B})$.

$$f(A,B,C) = \overline{A}BC + AB\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$$

3.2 Représentation des fonctions logiques

Les fonctions logiques peuvent être représentées de différentes façons. Nous venons de voir ses représentations algébriques. Toutes les représentations sont équivalentes.

a) Table de vérité

Une fonction f de n variables est entièrement décrite par l'énoncé de l'ensemble des combinaisons des variables d'entrées et de la fonction associée à chaque combinaison. Cet énoncé prend généralement la forme d'un tableau à n+1 colonnes et 2^n lignes, Dans chaque ligne il y a une combinaison des variables et la valeur correspondante de la fonction.

A	B	C	F(A,B,C)
0	0	0	F(0,0,0)

0	0	1	F(0,0,1)
0	1	0	F(0,1,0)
0	1	1	F(0,1,1)
1	0	0	F(1,0,0)
1	0	1	F(1,0,1)
1	1	0	F(1,1,0)
1	1	1	F(1,1,1)

Exemple

On définit la fonction logique f (A, B, C) = 1 si $(A, B, C)_2 > 5$

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

La lecture sur les 1, donne la forme canonique disjonctive, la lecture sur les 0 donne la forme canonique conjonctive.

Exemple de lecture sur les 1 (tableau précédent), les variables à 1 sont exprimées sous forme normale, les variables à 0 sont exprimées sous formes complémentées.

$$F(A,B,C) = AB\overline{C} + ABC$$

b) Expression numérique

Pour simplifier la représentation de la fonction, nous pouvons l'exprimer sous forme numérique. Si nous reprenons l'exemple précédent la fonction logique f (A, B, C) = 1 si
$(A, B, C)_2 > 5$, $f(A,B,C) = \Sigma\ (6,7)$

3.3 Simplification des fonctions logiques

Les fonctions binaires sont réalisées avec des circuits électroniques pour lesquels nous cherchons toujours à réduire le coût, le temps de traversée des portes (augmenter la vitesse du circuit), et la consommation d'énergie.

Nous cherchons donc à représenter la fonction avec un minimum de termes car une fonction simplifiée utilisera moins de circuits. Elle sera exécutée, plus rapidement, à un moindre coût et consommera moins d'énergie.

a) Méthode algébrique

On utilise les différents théorèmes et propriétés de l'algèbre de Boole (voir chapitre précédent et exercices).

Exemple : Simplifier par les méthodes algébriques les fonctions suivantes

$$F_1 = [A.\overline{B}(C + BD) + \overline{A}.\overline{B}]C$$
$$F_2 = \overline{A}.BC + A\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.\overline{B}.C + ABC$$
$$F_3 = \overline{\overline{AB} + \overline{AC}} + \overline{A}.\overline{B}.C$$
$$F_4 = BD + B(D + E) + \overline{D}(D + F)$$

Calculons :

$$F_1 = [A.\overline{B}(C + BD) + \overline{A}.\overline{B}]C = [A.\overline{B}C + A.\overline{B}BD + \overline{A}.\overline{B}]C = [A.\overline{B}C + A.0.D + \overline{A}.\overline{B}]C$$
$$= [A.\overline{B}C + 0 + \overline{A}.\overline{B}]C = [A.\overline{B}C + \overline{A}.\overline{B}]C = A.\overline{B}CC + \overline{A}.\overline{B}C = A.\overline{B}C + \overline{A}.\overline{B}C$$
$$= (A + \overline{A}.).\overline{B}C = .\overline{B}.C$$

$$F_2 = \overline{A}.BC + A\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.\overline{B}.C + ABC = \overline{A}.BC + (A + \overline{A})\overline{B}.\overline{C} + A.\overline{B}.C + ABC$$
$$= \overline{A}.BC + \overline{B}.\overline{C} + A.\overline{B}.C + ABC = (\overline{A} + A).BC + \overline{B}.\overline{C} + A.\overline{B}.C = BC + \overline{B}.\overline{C} + A.\overline{B}.C$$
$$= BC + \overline{B}.(\overline{C} + A.C) = BC + \overline{B}.(\overline{C} + A) = .B.C + A.\overline{B} + \overline{B}.\overline{C}$$

$$F_3 = \overline{\overline{AB} + \overline{AC}} + \overline{A}.\overline{B}.C = \overline{\overline{AB}}.\overline{\overline{AC}} + \overline{A}.\overline{B}.C = (\overline{A} + \overline{B}).(\overline{A} + \overline{C}) + \overline{A}.\overline{B}.C$$
$$= \overline{A}.\overline{A} + \overline{B}.\overline{A} + .\overline{A}.\overline{C} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C = \overline{A}. + \overline{B}.\overline{A} + .\overline{A}.\overline{C} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C$$
$$= \overline{A}. + .\overline{A}.\overline{C} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C = \overline{A}. + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C = \overline{A}. + \overline{B}.\overline{C}$$

$$F_4 = BD + B(D + E) + \overline{D}(D + F) = BD + BD + BE + \overline{D}D + \overline{D}F = BD + BE + \overline{D}D + \overline{D}F$$
$$= BD + BE + \overline{D}F$$

b) Méthode de Karnaugh

Les tableaux de Karnaugh sont des représentations de la table de vérité de la fonction. Ils sont construits pour que les termes logiquement adjacents soient géométriquement adjacents. La condition d'adjacence est réalisée en utilisant le code GRAY. Deux combinaisons sont entre elles adjacentes si une seule variable change d'état entre deux combinaisons.

Règles de simplification :

Pour une fonction de n variables, le tableau à 2^n cases. A chaque case est associée un produit de variables égales à 1 ou à 0 pour la combinaison. Le passage d'une case à une autre se fait par changement d'une seule variable à la fois. Les cases sont repérées par le code Gray ou code binaire réfléchi.
Code de Gray pour deux variables : 00, 01, 11,10
Code de Gray pour trois variables : 000,001,011,010,110,111,101,100

Nous regroupons 2^n cases adjacentes de 1 (n = 0,1,2, ..). Les cases ayant un coté commun sont adjacentes ainsi que les cases extrêmes.

L'expression simplifiée sera constituée des variables qui ne changent pas d'état sur les 2^n cases regroupées.

Remarque importante : La simplification peut ne pas être unique

Exemple pour quatre variables

$$F(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} + A\,\overline{B}\,\overline{C}\,\overline{D} + A\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,D$$

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

La colonne à 1 correspond à CD = 00
Les quatre coins B=0 C=0
Le groupe de 2 dans la première ligne correspond à A=0,B=0, C=0

$$F(A,B,C,D) = \overline{C}\,\overline{D} + \overline{B}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}$$

Exemple pour cinq variables

Simplifier la fonction suivante.

f(A, B, C, D, E)= Σ (0,4,8,12,13,15,16, 17, 19, 23,29,31)

CDE / AB	000	001	011	010	110	111	101	100
00	1							1
01	1					1	1	1
11						1	1	
10	1	1	1			1		

Il y a deux groupes de quatre variables : $\overline{A}\,\overline{D}\,\overline{E}$ correspondant aux cases 0,4,8,12 et $B\,C\,E$
correspondant aux cases 13,15,31,29.
Il y a deux groupes de deux variables : $A\,\overline{B}\,\overline{C}\,\overline{D}$ correspondant aux cases 16 et 17 et $A\overline{B}\,D\,E$ correspondant aux cases 19 et 23.

$$f(A,B,C,D,E) = \overline{A}\,\overline{D}\,\overline{E} + B\,C\,E + A\,\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,D\,E$$

Chapitre 3 : Les opérateurs élémentaires

Ces circuits peuvent être utilisés pour effectuer une synthèse combinatoire. Une synthèse combinatoire est la traduction d'une fonction logique à partir d'un cahier des charges en un schéma. Diverses méthodes de synthèse sont possibles ; elles diffèrent sur la forme de la fonction utilisée (canonique ou simplifiée), sur le type d'opérateurs ou de CI choisi, sur la technique de découpage fonctionnel employée.

1) Les opérateurs intégrés fondamentaux NON OU ET

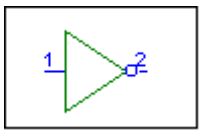
Comme nous l'avons vu précédemment, l'algèbre de Boole s'appuie sur les trois opérations élémentaires ET OU NON

1.1 L'opérateur NON

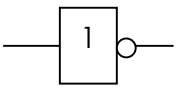
Il est également appelé inverseur. Il affecte à la variable de sortie l'état complémentaire de la variable d'entrée.

$$F(x) = \overline{x}$$

Il existe deux types de symbole pour représenter cet opérateur. Le symbole européen qui est normalisé et le symbole américain également très utilisé dans les simulateurs notamment.



Symbole Américain



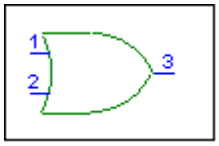
Symbole Européen

1.2 L'opérateur OU

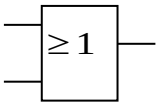
L'opérateur OU représente la somme logique ou l'union logique.

$$F(A,B) = A + B$$

Cet opérateur porte un nombre n de variable d'entrée. Dans la pratique la porte ne peut excéder sept entrées logiques. Les symboles de cette porte sont



Symbole Américain



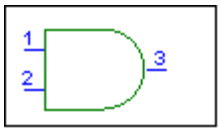
Symbole Européen

1.3 L'opérateur ET

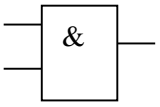
L'opérateur ET représente le produit logique ou l'intersection logique.

$$F(A,B) = A.B$$

Cet opérateur porte un nombre n de variables d'entrée. Dans la pratique la porte ne peut excéder sept entrées logiques. Les symboles de cette porte sont :



Symbole Américain



Symbole Européen

2) Les opérateurs logiques induits

Ces opérateurs sont obtenus en combinant entre eux les opérateurs précédents.

2.1 L'opérateur NON OU (NOR)

Cet opérateur est obtenu en associant un NON en sortie d'un OU. Il affecte à la variable de sortie la valeur zéro si une des variables d'entrée est à 1. L'opération NON OU est notée :

$$F(A,B) = A \downarrow B = \overline{A + B}$$

Sa table de vérité pour deux variables est la suivante.

A	B	F(A,B)
0	0	1
0	1	0
1	0	0
1	1	0

Propriétés de cette fonction :

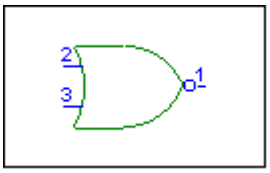
-Commutativité,

-Le NOR n'est pas associatif.

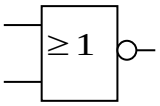
$$A \downarrow 0 = \overline{A + 0} = \overline{A}$$

$$A \downarrow A = \overline{A + A} = \overline{A}$$

Les symboles des opérateurs sont :



Symbole Américain



Symbole Européen

Dans la pratique la porte ne peut excéder sept entrées logiques.

2.1 L'opérateur NON ET (NAND)

Cet opérateur est obtenu en associant un NON en sortie d'un ET. Il affecte à la variable de sortie la valeur 1 si une des variables d'entrée est à 0. L'opération NAND

$$F(A,B) = A/B = \overline{A.B}$$

est notée :

Sa table de vérité pour deux variables est la suivante.

A	B	F(A,B)
0	0	1
0	1	1
1	0	1
1	1	0

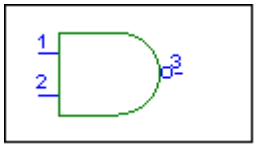
Propriétés de cette fonction :

- Commutativité

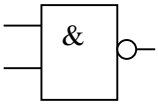
- Le NAND n'est pas associatif.

$$\overline{A.A} = \overline{A}$$
$$\overline{A.1} = \overline{A}$$

Symboles utilisés



Symbole Américain



Symbole Européen

Dans la pratique la porte ne peut excéder sept entrées logiques.

2.3 L'opérateur OU EXCLUSIF (XOR)

L'opérateur OU EXCLUSIF met à 1 la variable de sortie si une et une seule des deux variables d'entrée a la valeur 1. Pour n variables : l'opérateur OU EXCLUSIF affecte un 1 en sortie quand les variables d'entrée à 1 sont en nombre impair.

L'opération XOR est notée :

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

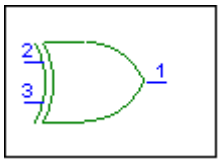
Sa table de vérité pour deux variables est la suivante.

A	B	F(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

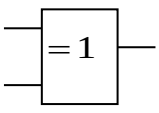
Propriétés de cette fonction :

- Commutativité
- Associativité
- Détecteur d'inégalité
- Détecteur d'imparité
- Élément neutre 0

Symboles utilisés



Symbole Américain



Symbole Européen

La porte logique OU EXCLUSIF ne possède que deux entrées.

2.3 L'opérateur NON OU EXCLUSIF (XNOR)

L'opérateur NON OU EXCLUSIF est la négation du OU EXCLUSIF. Pour deux variables, il met à 1 la variable de sortie si les deux variables d'entrée sont égales.
Pour n variables : l'opérateur XNOR affecte un 1 en sortie quand les variables d'entrée à 1 sont en nombre pair.

L'opération XNOR est notée :

$$F(A,B) = AB + \overline{A}\ \overline{B} = \overline{A \oplus B}$$

Sa table de vérité pour deux variables est la suivante.

A	B	F(A,B)
0	0	1
0	1	0
1	0	0
1	1	1

Propriétés de cette fonction :

- Commutativité
- Associativité.
- Détecteur d'égalité
- Détecteur de parité

Le symbole du XNOR est construit à partir du XOR avec une inversion (Rond ou Barre en sortie)

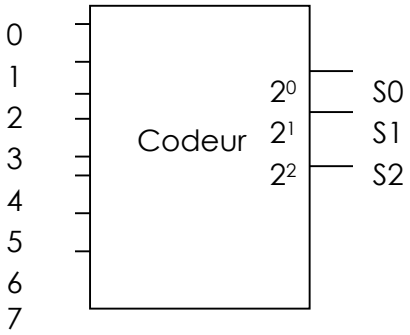
3) Fonctions de logiques combinatoires

3.1) Les circuits de transcodage

Ce mot désigne l'ensemble des codeurs, décodeurs et convertisseurs de code. Ces circuits transforment une information présente à leurs entrées (code 1) en la même information mais sous une forme différente en sortie (code 2).

3.1.1) Les Codeurs

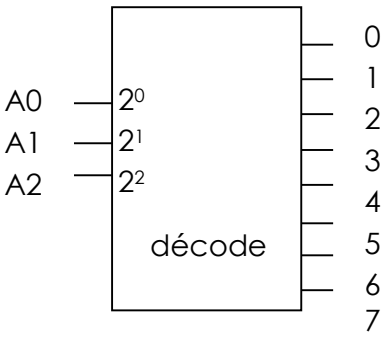
C'est un circuit qui possède 2^n entrées et n sorties. Ce circuit code en binaire le numéro décimal de l'entrée activée.



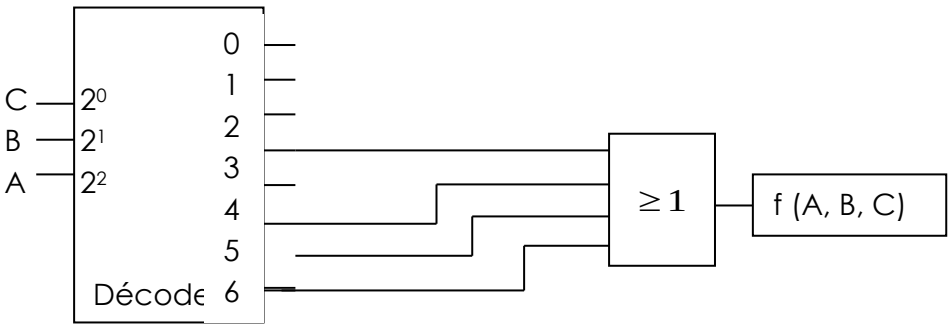
Par exemple si l'entrée 6 est active, le mot de sortie S2 S1 S0 = 110

3.1.2) Les Décodeurs

C'est un circuit qui possède n entrées et 2^n sorties. Ce circuit code en décimal l'entrée codée en binaire.



Exemple 1 : A2 A1 A0 = 0 1 1, la sortie 3 est active. En logique positive elle est à 1.
Exemple 2 : On cherche à réaliser la fonction f (A, B, C) = Σ (3, 5, 6, 7) avec un décodeur et une porte logique.



3.1.3) Les Transcodeurs (ou convertisseurs de codes)

Ces circuits convertissent un code quelconque en un autre code à l'exception des codes binaires et décimaux.

3.2) Les circuits d'aiguillage

3.2.1) Les multiplexeurs

Ce circuit est utilisé pour réaliser une conversion parallèle série.

C'est un circuit qui possède

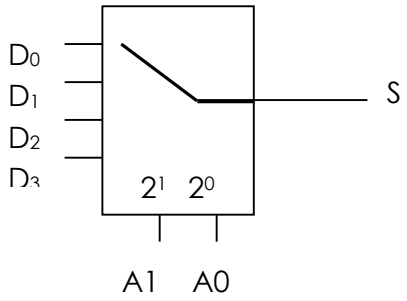
n entrées commandes $A_{n-1}…A_0$

2^n entrées de données $d_0 …d_{2n-1}$

1 sortie S

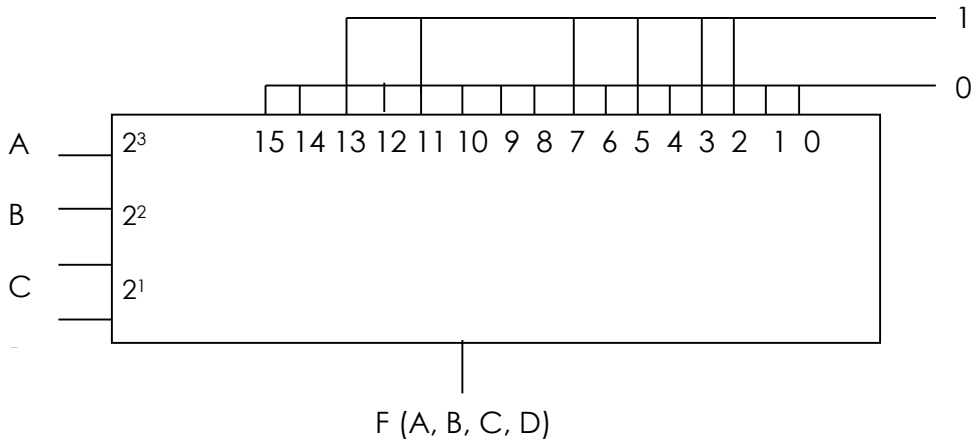
Un multiplexeur est nommé par rapport à ces entrées de données. On parle de multiplexeur 8 vers 1 (Pour 8 entrées données 3 entrées adresses et une sortie). Le multiplexeur peut être vu comme un interrupteur commandé par les entrées adresses. Selon la valeur donnée des entrées adresses une des entrées données sera connectée en sortie.

Supposons un multiplexeur à deux entrées adresses A1 et A0, pour chaque combinaison binaire de A1 A0 une entrée donnée sera connectée en sortie. Par exemple pour A1 A0 = 10, l'entrée D2 sera connectée en sortie.



Exemple d'application d'un multiplexeur : générateur de fonction
Soit la fonction F(A, B, C, D) = Σ(2, 3, 5, 7, 11, 13).

1) On cherche à réaliser cette fonction avec un multiplexeur à quatre entrées adresses.



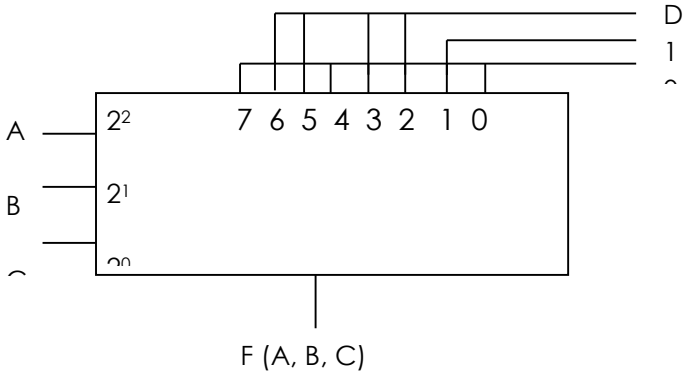
2) On cherche à réaliser cette fonction avec un multiplexeur à trois entrées adresses.

$F(A, B, C, D) = \Sigma\ (2, 3, 5, 7, 11, 13) = \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}CD + AB\overline{C}D$

$= \overline{A}\overline{B}C(\overline{D} + D) + \overline{A}B\overline{C}(D) + \overline{A}BC(D) + A\overline{B}C(D) + AB\overline{C}(D)$

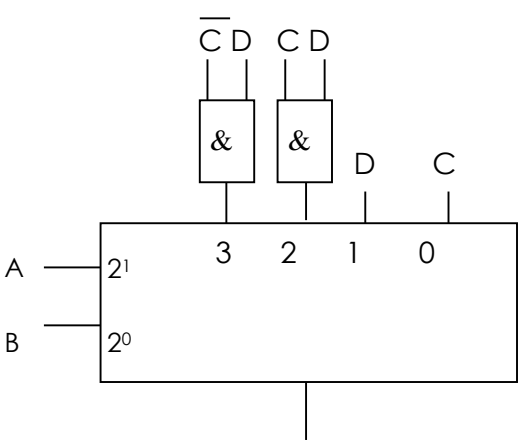
$= \overline{A}\overline{B}C(1) + \overline{A}B\overline{C}(D) + \overline{A}BC(D) + A\overline{B}C(D) + AB\overline{C}(D)$

Les termes non spécifiés dans l'équation sont à 0.



3) On cherche à réaliser cette fonction avec une multiplexeur à deux entrées adresses.

$F(A, B, C, D) = \overline{A}\overline{B}\,C + \overline{A}B\,(D) + A\overline{B}\,CD + AB\,\overline{C}D$



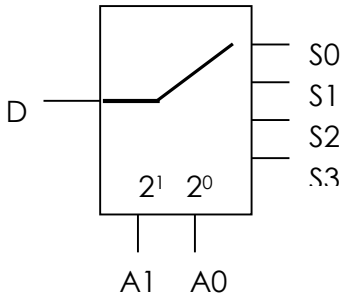
3.2.2) Les démultiplexeurs

Ce circuit réalise l'opération inverse du multiplexeur. Il réalise une conversion série parallèle des données.

Un démultiplexeur est un circuit qui possède
n entrées commandes
2^n sorties
1 entrée donnée

Un démultiplexeur recopie l'entrée D sur la sortie correspondante à la valeur de commande. Une sortie non sélectionnée à la valeur zéro.

Lorsque l'entrée vaut toujours 1, un démultiplexeur devient un décodeur.
Pour un démultiplexeur à 2 entrées adresses si on met par exemple A_1A_0 = 11 la sortie 3 est connecté à D



4) Les circuits arithmétiques

4.1) Les Comparateurs

4.1.1) Les Comparateurs d'égalité

Deux mots A (A_3 A_2 A_1A_0) et B (B_3 B_2 B_1B_0) sont égaux si les éléments binaires de même poids sont égaux deux à deux, c'est à dire si $A_3 = B_3$ et $A_2 = B_2$ et $A_1 = B_1$ et $A_0 = B_0$

La fonction XNOR permet de détecter l'égalité de deux éléments binaires.

$$f(A = B) = \left(\overline{A_3 \oplus B_3}\right).\left(\overline{A_2 \oplus B_2}\right).\left(\overline{A_1 \oplus B_1}\right).\left(\overline{A_0 \oplus B_0}\right)$$

4.1.2) Les Comparateurs

On souhaite réaliser un comparateur complet de deux mots de quatre bits.

Un comparateur de deux éléments binaires a_0 et b_0 fonctionne de la façon suivante :

S supérieur est égal 1 si $a_0 > b_0$
I inférieur est égal à 1 si $a_0 < b_0$
E=1 si les deux éléments binaires sont égaux.

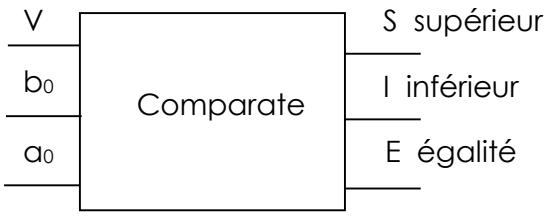
Sa table de vérité est la suivante :

a_0	b_0	S	I	E
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Les équations des sorties sont:

$$S = a_0\overline{b_0}\ ,\ I = \overline{a_0}b_0\ ,\ E = \overline{a_0 \oplus b_0}$$

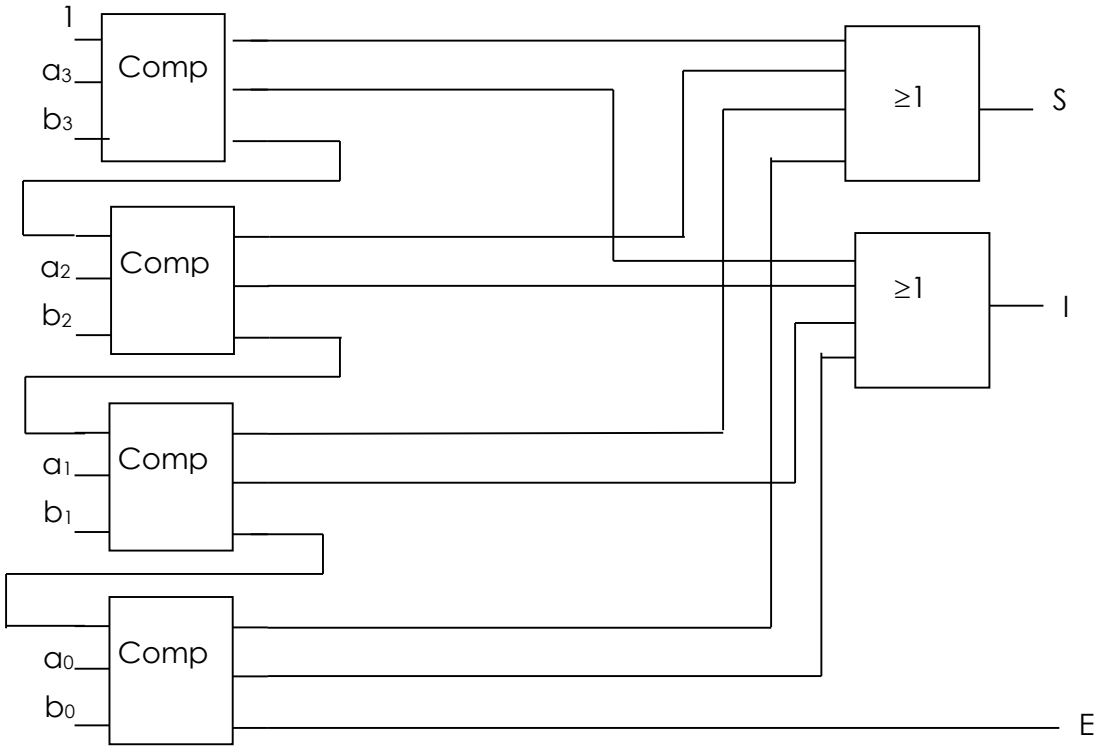
Le comparateur peut être schématisé par le circuit suivant, l'entrée V est une entrée de validation, le comparateur fonctionne si V est égal à 1 sinon toutes les sorties sont égales à zéro.



Nous cherchons maintenant à réaliser un comparateur en cascade de deux mots de quatre bits A (a_0, a_1, a_2, a_3) et B(b_0, b_1, b_2, b_3) à l'aide de comparateurs de 1 bit et de quelques portes logiques. La sortie S (supérieur) est vraie si S_3 est à 1 ou si S_2 est à 1 ou si S_1 est à 1 ou si S_0 est à 1. Le principe est le même pour la sortie I. Nous commençons la comparaison par les bits de poids forts car si S_3 = 1, ce n'est pas la peine de comparer les éléments binaires de poids plus faibles, le mot A sera supérieur au mot B.

Pour l'égalité, comme nous l'avons vu au paragraphe précédent que les éléments binaires doivent être égaux deux à deux, l'entrée de validation permet la détection d'égalité des éléments binaires de poids supérieurs.

Le schéma suivant représente ce comparateur.



Le résultat de la comparaison apparaît après un temps lié aux nombres de cellules traversées. Pour pallier à cet inconvénient, il faudrait utiliser une structure parallèle.

4.2) Les Additionneurs

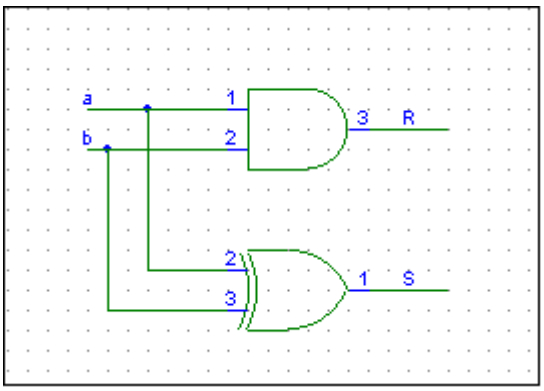
4.2.1) Demi additionneur

La table de vérité d'un additionneur de deux éléments binaires est la suivante :

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

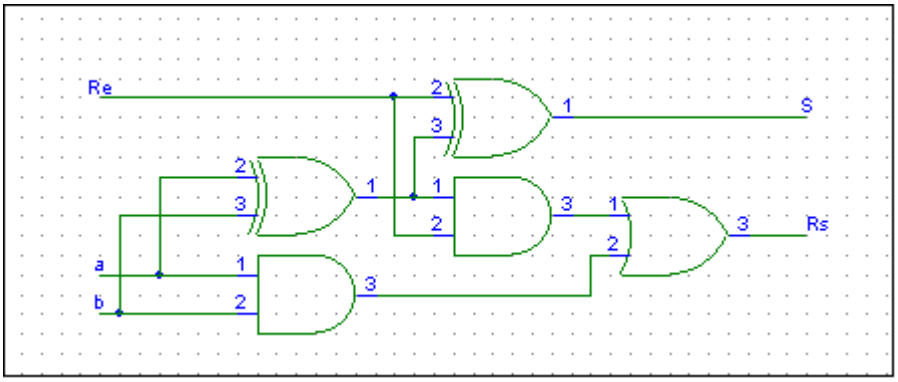
La sortie R représente la retenue, la sortie S représente la somme.
$S = a \oplus b,\ R = a.b$

Le schéma est le suivant :



4.2.2) Additionneur complet

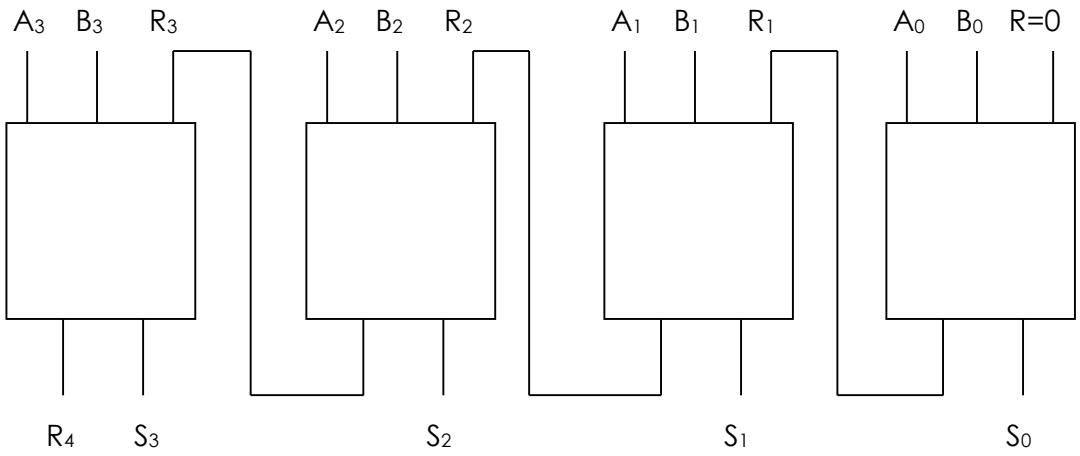
Un additionneur complet effectue la somme de trois éléments binaires a, b et r_e la retenue entrante et a deux sorties S la somme et Rs la retenue sortante. Le schéma d'un tel circuit est le suivant.



On peut représenter ce circuit sous la forme de la boite noire suivante

4.2.3) Additionneur de deux mots à propagation de retenue

L'addition de deux mots de n bits nécessite n additionneurs. La retenue se propage des éléments binaires de poids faibles vers les éléments binaires de poids forts. Le schéma suivant présente un additionneur de mots de quatre bits.

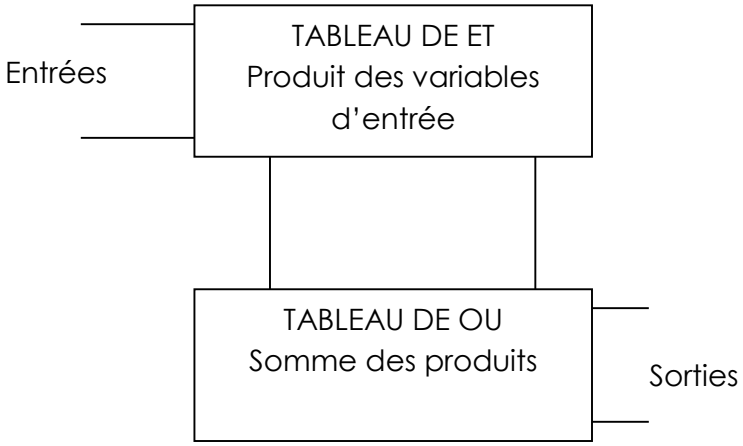


Cette architecture est intéressante d'un point de vue matériel car elle est répétitive. Par contre le résultat obtenu dépend du nombre d'additionneurs donc des mots à additionner. La retenue R_1 est délivrée après la première addition et ainsi de suite. Pour éliminer cet inconvénient une seconde technique consiste à calculer toutes les retenues en parallèle directement à partir des données sans même calculer les sommes partielles. Ce circuit que nous ne présenterons pas est appelé additionneur à retenue anticipée (voir projet).

5) Introduction aux éléments logiques programmables.

Toute fonction logique peut être écrite comme une somme de produits de variables logiques (Forme canonique disjonctive). La structure permettant l'implantation de fonctions de variables logiques combine un tableau de ET et un tableau de OU.

Les variables d'entrées sont disponibles sous forme directe ou complémentée.



Les éléments logiques programmables sont classés selon leur architecture.

Les Mémoires mortes programmables (PROM)

Une mémoire PROM (Programmable read-only memory) contient un tableau de ET non programmable connecté à un tableau de OU programmable. Les PROMs sont généralement utilisée comme mémoire et non comme circuit logique.

Les PLA (Programmable logic array)

Un PLA contient un tableau de ET programmable et un tableau de OU programmable. La complexité de ce circuit est trop importante, elle entraîne des retards inhérents.

Les PAL (Programmable array logic)

Les PALs contiennent un tableau de ET programmable et un tableau de OU fixe et une logique de sortie. Les PALs sont à programmation unique.

Les GAL (Generic array logic)

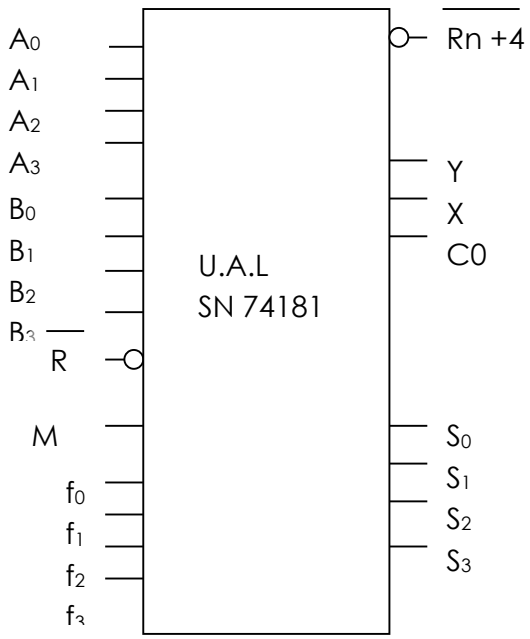
Ces éléments logiques programmables sont les plus récents. Ils contiennent un tableau de ET programmable et un tableau de OU fixe avec une logique de sortie programmable.
Contrairement au PAL, les GAL sont reprogrammables. La configuration de sortie est programmable.

6)Unité Arithmétique et Logique (U.A.L) ou A.L.U. (Arithmetic Logic Unit)

Ce circuit est utilisé dans presque tous les systèmes de calcul. C'est un opérateur capable d'effectuer un ensemble d'opérations logiques (ET, OU, NON, NOR, NAND, XOR, XNOR …) et d'opérations arithmétiques (Addition, Soustraction, Incrémentation ….). Les U.A.L sont caractérisées par la taille des opérandes traitées et par le nombre d'opérations différentes exécutées.

On peut utiliser ces circuits pour faire évoluer les séquences d'un automate programmable. En compliquant un peu l'U.A.L on aboutit au microprocesseur qui regroupe sur la même pastille de silicium une U.A.L et une unité de commande qui synchronise les différentes commandes du processeur en fonction des ordres données. Cette unité de commande est formée de circuits combinatoires et séquentiels.

Exemple de l'U.A.L SN 74181



Deux mots de quatre bits sont en entrées A (A_3, A_2, A_1, A_0) et B (B_3, B_2, B_1, B_0). Le mot de sortie est S (S_3,S_2,S_1,S_0). Rn est la retenue (de l'étage précédent) en entrée, Rn+4

est la retenue de sortie. Le résultat de l'opération est donc contenu par Rn+4, S_3, S_2, S_1, S_0.

M effectue la sélection entre le mode logique et arithmétique (M=0 les opérations sont arithmétiques, M=1 les opérations sont logiques). Les opérations arithmétiques ou logiques sont choisies par les signaux f_3, f_2,f_1, f_0. Seize opérations sont donc possibles en logique et seize opérations sont possibles en arithmétique. Les signaux X et Y servent à générer et propager les retenues en additionneur soustracteur. C0 est à 1 quand les quatre sorties S sont à 1. Les retenues entrantes et sortantes indiquées avec un rond en entrée sont active sur niveau bas.