

CHAPITRE I

Le binaire, l'algèbre de BOOLE et les fonctions logiques.

Table des Matières

| | |
|--|-----------|
| LE BINAIRE. ----- | 2 |
| PRINCIPE DES CHANGEMENTS DE BASES. ----- | 2 |
| LES CODES BINAIRES. ----- | 5 |
| <i>Le binaire naturel.</i> ----- | 5 |
| Exemple : ----- | 6 |
| <i>Le code GRAY.</i> ----- | 6 |
| <i>Le code DCB.</i> ----- | 7 |
| LES OPÉRATIONS EN BINAIRE. ----- | 8 |
| <i>L'addition.</i> ----- | 8 |
| Exemple----- | 8 |
| <i>La multiplication</i> ----- | 8 |
| <i>La soustraction.</i> ----- | 9 |
| <i>La division.</i> ----- | 10 |
| REPRÉSENTATION DES NOMBRES RÉELS. ----- | 11 |
| <i>Traduction des nombres fractionnaires.</i> ----- | 11 |
| <i>Les nombres signés.</i> ----- | 13 |
| L'ajout d'un bit. ----- | 13 |
| Le complément à 1. ----- | 13 |
| Le complément à 2 ----- | 14 |
| LES BASES 8 ET 16. ----- | 17 |
| L'ALGÈBRE DE BOOLE. ----- | 18 |
| LES TABLES DE VÉRITÉ. ----- | 18 |
| L'OPÉRATEUR NON (OPÉRATEUR DE COMPLÉMENTATION).----- | 18 |
| <i>Présentation algébrique.</i> ----- | 18 |
| <i>Table de vérité.</i> ----- | 19 |
| L'OPÉRATEUR ET. ----- | 19 |
| L'OPÉRATEUR OU. ----- | 20 |
| COMBINAISON DE FONCTIONS LOGIQUES. ----- | 21 |
| REPRÉSENTATION GRAPHIQUES. ----- | 21 |
| LES OUTILS BOOLÉENS. ----- | 22 |
| TABLE DE VÉRITÉS ET FONCTIONS ALGÈBRIQUES. ----- | 22 |
| LE THÉORÈME DE DE MORGAN. ----- | 23 |
| LES TABLEAUX DE KARNAUGH. ----- | 24 |

Le binaire.

Il y a quelques années vos enseignants vous ont appris l'arithmétique, mais préalablement, ils vous ont tous présentés la définition de la base 10. En effet, il faut bien se rendre compte que la base conditionne tous les éléments de raisonnement et de calcul.

Or si la base 10 est votre élément habituel pour la bonne et simple raison que nous n'avons que 10 doigts, il faut savoir qu'il n'en a pas toujours été ainsi, c'est Charlemagne (vers 800 après JC) qui a officialisé la base 10 en prenant le modèle arabe (sans le zéro qui est un élément très récent).

Avant, les Mésopotamiens étaient des fervents de la base 12 puisque pour eux le poing était un élément de comptage. Et jusqu'à la révolution française, il faut bien se rendre compte que notre système monétaire été fondé sur la base 12.

Pour vous qui souhaitez devenir des électroniciens, la base 10 est une rareté. Nous utilisons essentiellement des ordinateurs ou des composants électroniques utilisant des bits (bit est l'abréviation de "Binary unit" et se traduit par élément binaire). Or un bit ne connaît que 2 valeurs "1" ou "0" (vrai ou faux).

On voit donc qu'il est impossible d'utiliser une bonne vieille base 10 puisque l'électronique numérique est prévue sur le modèle de la base 2. On définit donc une nouvelle méthode de calcul le binaire. Mais comme le binaire possède l'énorme défaut de ne pas présenter de formes courtes (il faut 10 signes "1" et "0" pour exprimer un nombre de l'ordre de 1000 en base 10), on utilise des développements du binaire que sont l'octal (base 8) ou l'hexadécimal (base 16).

Principe des changements de bases.

Imaginons pour commencer un nombre N en base A. Si l'on souhaite l'exprimer en base B, il va falloir effectuer une opération de changement de base.

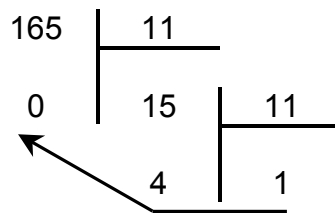
En base A, on divise le nombre N par B de façon successive jusqu'à ce que le quotient de la division soit inférieur à B. Puis on lit le résultat en revenant en arrière. Par exemple si A est la base 10 et B la base 7 (après tout pourquoi pas), pour N=165, on change de base de la façon suivante :

$$\begin{array}{r}
 165 \quad | \quad 7 \\
 \hline
 4 \quad | \quad 23 \quad | \quad 7 \\
 \hline
 \quad \quad | \quad 2 \quad | \quad 3 \\
 \hline
 \end{array}$$

←

Ainsi 165 en base 7 se traduit par 324.

Envisageons maintenant un autre exemple, en utilisant les mêmes valeurs de A et de N, on peut prendre une base B supérieure la base A, par exemple B=11.



165 en base 11 se traduit par 40.

Bien entendu, les plus perspicaces d'entre-vous voient tout de suite le piège que j'ai caché subrepticement, j'ai pris A=10 tandis que justement, c'est pour A différent de 10 que se posent des problèmes.

En effet, rares sont ceux qui maîtrisent assez bien les bases différentes de 10. Donc faire des calculs en base 11 ou 7, c'est assez complexe. Heureusement, il y a une autre solution pour effectuer ces calculs, cette méthode, c'est le passage par la base 10. En effet, une fois traduit en base 10, tout devient simple.

Posons-nous donc la question suivante, comment traduire de la base B vers la base 10 un nombre quelconque.

Par exemple : convertissons 140 de la base 11 vers la base 10. La méthode la plus simple est de développer 140 en base 11 comme un polynôme dont la variable est la base :

$$\begin{aligned}
 &= a_n \cdot x^n + \dots + a_1 \cdot x + a_0 \\
 (140)_{11} &= 1 \cdot (11)^2 + 4 \cdot (11) + 0 \\
 &= 121 + 44 + 0 \\
 &= 165
 \end{aligned}$$

On retrouve bien le nombre précédemment traduit.

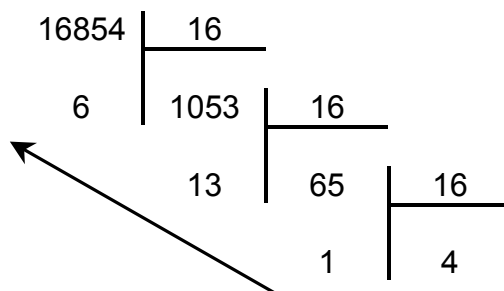
Reprenons un autre exemple en convertissant 324 de la base 7 vers la base 10.

$$\begin{aligned}
 (324)_7 &= 3 \cdot (7)^2 + 2 \cdot 7 + 4 \\
 &= 3 \cdot 49 + 14 + 4 \\
 &= 147 + 14 + 4 \\
 &= 165
 \end{aligned}$$

Le seul problème lié à cette méthode de calcul, c'est sa complexité pour des nombres relativement grands. Mais heureusement, cela ne pose pas de problème pour un utilisateur de calculatrice.

Mais il existe aussi un autre problème, comment exprimer un nombre en base B supérieure à 10, quels chiffres utilise-t-on pour compléter la table. Ainsi en base 10, les chiffres utilisables sont 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Pour une base supérieure à 10, on ajoute des lettres. Ainsi, la base 16 se traduit par l'utilisation des chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F. Où A vaut 10, B vaut 11, etc.

Par exemple convertissons 16854 de la base 10 vers la base 16



La traduction de $(16854)_{10}$ est donc $(41D6)_{16}$

Exercices.

En essayant de ne pas utiliser de calculatrice, complétez les équations suivantes :

La première ligne est un exemple. Utilisez une feuille de brouillon pour effectuer vos calculs. En respectant scrupuleusement les consignes précédentes, vous vous préparez d'une bonne manière aux futurs DS de numérique.

| Enoncé | Résultat | Base | Correction |
|-------------------|--|------|-----------------|
| $(127)_{10}$ | $127/2=63$ reste 1; $63/2=31$ reste 1; $31/2=15$ reste 1; $15/2=7$ reste 1; $7/2=3$ reste 1; $3/2=1$ reste 1 | 2 | $(1111111)_2$ |
| $(127)_{10}$ | | 8 | $(177)_8$ |
| $(127)_{10}$ | | 16 | $(7F)_{16}$ |
| $(7FE)_{16}$ | | 10 | $(2046)_{10}$ |
| $(7654)_8$ | | 10 | $(4012)_{10}$ |
| $(10100110101)_2$ | | 10 | $(1333)_{10}$ |
| $(127)_{16}$ | | 2 | $(100100111)_2$ |
| $(127)_{16}$ | | 8 | $(447)_8$ |

Les codes binaires.

Le binaire est un mode de représentation des chiffres, adapté aux composants électroniques. Toutefois, on a créé différentes façons de coder une information en binaire. Vous connaissez maintenant tous le binaire naturel (celui que vous obtenez par vos opérations de changement de base).

Mais récapitulons un peu nos connaissances en binaire naturel et trouvons des méthodes pour connaître, sans gros calculs, la valeur décimale d'un nombre binaire.

Le binaire naturel.

Le binaire naturel est la représentation la plus commune du binaire, c'est en tout cas, le mode de représentation que vous utilisez le plus.

Il est basé sur la forme polynomiale de 2, (là vous devez vous demander si je n'ai pas pété les plombs, mais bon, désolé, c'est comme ça que ça s'appelle). Alors cette forme polynomiale, vous savez ce que c'est, puisque pour la connaître, il suffit de partir de 1 et de multiplier à chaque fois le nombre obtenu par la dernière multiplication par 2 (facile). De tête, un étudiant lambda doit être capable de me donner le 12^{ème} terme de cette série géométrique :

| | | | | | | | | | | | | | |
|-----------------|------|------|------|-----|-----|-----|----|----|----|---|---|---|---|
| Puissance | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Valeur décimale | 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Par cette méthode, il est facile de traduire un nombre binaire en un nombre décimal. Par exemple si l'on prend le code 1111111110, on peut le traduire par :

| | | | | | | | | | | | | | |
|-----------------|--------------------------------------|------|------|-----|-----|-----|----|----|----|---|---|---|---|
| Puissance | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Valeur décimale | 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Nombre binaire | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Nombre décimal | 1024+512+256+128+64+32+16+8+4+2=2046 | | | | | | | | | | | | |

Voilà comment traduire rapidement un nombre binaire en un nombre décimal. Ainsi, en extrapolant, et en partant du fait que vous connaissez parfaitement les 12 premières puissances de 2, un nombre décimal se décompose rapidement en binaire par soustractions successives.

Exemple :

1000 est compris entre 1024 et 512 donc on lui soustrait 512, il reste alors 488 auquel pour la même raison on peut soustraire 256, puis 128, puis 64 puis 32, il reste alors 8, on le soustrait. On peut donc dire que 1000 s'exprime en binaire de la façon suivante : 1111101000.

Mais ce binaire a un gros défaut, il fait régulièrement changer plus de 2 bits en même temps, ce qui en électronique est impossible. On doit donc se résoudre à utiliser une autre forme qui ne fait varier qu'un seul bit à la fois, c'est le code GRAY (ou binaire réfléchi)

Le code GRAY.

Le code GRAY pallie efficacement à l'un des plus gros problèmes de l'électronique : la non-simultanéité.

Ainsi, si l'on récapitule les 16 premiers nombres du binaire naturel, on se rend compte que des changements simultanés apparaissent à foison ! Le code GRAY présenté à côté est lui par contre tout à fait libre de ces défauts.

| Binaire Naturel | | | | Changements simultanés | Code GRAY | | | | Réflexions |
|-----------------|---|---|---|---|-----------|---|---|---|----------------|
| 3 | 2 | 1 | 0 | | 3 | 2 | 1 | 0 | |
| 0 | 0 | 0 | 0 | <div style="display: flex; justify-content: center; align-items: center; gap: 10px;"> <div style="text-align: center;">← 2</div> <div style="text-align: center;">← 3</div> <div style="text-align: center;">← 2</div> <div style="text-align: center;">← 4</div> <div style="text-align: center;">← 2</div> <div style="text-align: center;">← 3</div> <div style="text-align: center;">← 2</div> </div> | 0 | 0 | 0 | 0 | Bit 0 |
| 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 1 | Bits 1 et 0 |
| 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 0 | Bit 0 |
| 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | |
| 0 | 1 | 1 | 0 | | 0 | 1 | 0 | 1 | Bits 2, 1 et 0 |
| 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 0 | |
| 1 | 0 | 0 | 0 | | 1 | 1 | 0 | 0 | Bit 0 |
| 1 | 0 | 0 | 1 | | 1 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | Bits 1 et 0 |
| 1 | 0 | 1 | 1 | | 1 | 1 | 1 | 0 | |
| 1 | 1 | 0 | 0 | | 1 | 0 | 1 | 0 | Bit 0 |
| 1 | 1 | 0 | 1 | | 1 | 0 | 1 | 1 | |
| 1 | 1 | 1 | 0 | | 1 | 0 | 0 | 1 | Bit 0 |
| 1 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | |

Comme présenté par le tableau, le code GRAY est réalisé par des réflexions successives. Cela lui permet de ne faire varier pour passer d'un nombre à l'autre qu'un seul bit. Ce système de comptage est utilisé pour certains calculs en binaire que nous étudierons plus loin. Il sert aussi à certains systèmes de positionnement par systèmes optiques.

Le code DCB.

Le code DCB (Décimal Codé Binaire) est une méthode de représentation du code décimal en binaire.

Il est pratiquement exclusivement utilisé dans l'affichage des données en provenance d'instruments de mesures.

Ainsi, le codage se fait par décomposition en polynômes du nombre décimal, puis par traduction des coefficients de ce polynôme en binaire.

$$(N)_{10} = a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$$

on traduit en binaire: $(a_n)_{10} \rightarrow (A_n)_2$

$$(N)_{DCB} = A_n x^n + \dots + A_2 x^2 + A_1 x + A_0$$

Exemple.

En prenant $N=185$, sa traduction en DCB est réalisée par traduction successive de ses 3 coefficients (1, 8 et 5) en binaire.

| | | | |
|-------------|---------|---------|---------|
| $N_{10} =$ | 1 | 8 | 5 |
| $N_{DCB} =$ | 0 0 0 1 | 1 0 0 0 | 0 1 0 1 |

Soit, sous une forme un peu plus compressée :

$$N_{DCB} = 1\ 1000\ 0101$$

Exercices.

Traduisez les nombres suivants (toujours sans utiliser de calculatrice).

| Enoncé | Résultat | Code | Correction |
|-------------------|----------|-----------------|-------------|
| $(19)_{10}$ | | Binaire naturel | 10011 |
| $(19)_{10}$ | | Code GRAY | 11011 |
| $(567)_{10}$ | | Binaire naturel | 1000110111 |
| $(567)_{10}$ | | Code DCB | 10101100111 |
| $(10110101110)_2$ | | Décimal | 2902 |

La division.

La division binaire est le reflet exact de la division décimale. On utilise une nouvelle fois les mêmes méthodes, et les mêmes propriétés s'appliquent.

Exemple.

| Division décimale | | Division binaire | |
|---|--|---|---|
| $\begin{array}{r} 165 \\ - 11 \\ \hline 55 \\ - 55 \\ \hline 0 \end{array}$ | $\begin{array}{r} 11 \\ \hline 15 \end{array}$ | $\begin{array}{r} 10100101 \\ - 0000 \\ \hline 101000 \\ - 1011 \\ \hline 10011 \\ - 1011 \\ \hline 10000 \\ - 1011 \\ \hline 01011 \\ - 1011 \\ \hline 0000 \end{array}$ | $\begin{array}{r} 1011 \\ \hline 01111 \end{array}$ |

Sur ces deux derniers exemples, les plus perspicaces d'entre-vous ont du se dire que le hasard faisait bien les choses puisque les résultats de la soustraction et de la division étaient respectivement positif et entier. Je ne vous cache pas qu'il n'est pas rare (en binaire comme en décimal) de trouver, soit des nombres à virgule (ensembles \exists , Θ ou 3), soit des nombres négatifs (ensembles ρ , \exists , Θ ou 3).

Exercices.

Effectuez les opérations suivantes sans vous aider d'une calculatrice.

| Enoncé | Résultat | Correction |
|----------------------------|----------|------------|
| $1111101+1110101=$ | | 11110010 |
| $10001011+1110101=$ | | 10000000 |
| $1111101-1110101=$ | | 1000 |
| $10001011-1110101=$ | | 10110 |
| $11001 \times 10001=$ | | 110101001 |
| $100111 \times 10101=$ | | 1100110011 |
| $1110101 \div 1101=$ | | 1001 |
| $1110001111101 \div 1101=$ | | 1000110001 |

Représentation des nombres réels.

Les étudiants les plus avisés ont dû se rendre compte que jusqu'à présent, nous n'avons parlé de binaire qu'en terme de traduction des termes de l'ensemble \mathbb{Z} (ensemble des entiers naturels), or l'ensemble \mathbb{Z} n'est pas le seul que nous avons l'habitude d'utiliser. Pour pouvoir efficacement remplacer le code décimal, il faut trouver un moyen de passer le binaire vers l'ensemble le plus complet, \mathbb{R} l'ensemble des réels.

Traduction des nombres fractionnaires.

L'objectif de ce paragraphe est donc de chercher à intégrer les nombres à virgule dans le système de binaire naturel. Encore une fois, il existe plusieurs méthodes pour traduire un nombre décimal à virgule en binaire, mais nous allons pour l'instant nous limiter à la forme la plus élémentaire.

Prenons un exemple pour appuyer notre démonstration. Soit $N=0,3$, comment traduire N en binaire. La méthode la plus simple c'est de multiplier par 2 le nombre N et en sortir la partie entière. Puis multiplier à nouveau par 2 la partie décimale et ainsi de suite.

| Résultat | Calculs |
|----------|----------------------|
| 0 | $0,3 \times 2 = 0,6$ |
| , | |
| 0 | $0,6 - 0 = 0,6$ |
| | $0,6 \times 2 = 1,2$ |
| 1 | $1,2 - 1 = 0,2$ |
| | $0,2 \times 2 = 0,4$ |
| 0 | $0,4 - 0 = 0,4$ |
| | $0,4 \times 2 = 0,8$ |
| 0 | $0,8 - 0 = 0,8$ |
| | $0,8 \times 2 = 1,6$ |
| 1 | |

Etc.

La traduction de $0,3$ en décimal donne donc $0,01001\dots$ en binaire. On peut maintenant s'amuser à traduire en sens inverse ce résultat. La question est donc comment doit-on considérer une puissance négative dans un polynôme.

On a comme polynôme binaire :

$$\begin{aligned}(N)_2 &= a_p 2^p + \dots + a_0, a_{-1} 2^{-1} + a_{-2} 2^{-2} + \dots + a_{-n} 2^{-n} \\ &= 2^{-2} + 2^{-5} + \dots = \frac{1}{2^2} + \frac{1}{2^5} + \dots = \frac{1}{4} + \frac{1}{32} + \dots \\ &= 0,25 + 0,03125 + \dots = 0,28125 + \dots\end{aligned}$$

Comme vous l'avez sans doute remarqué, 0,3 ne peut pas se traduire directement en binaire, c'est à dire que l'on n'arrive à trouver qu'une forme approchée avec une certaine précision. Il convient donc de définir comment on traduit la précision à partir de la base 10, vers le binaire.

La précision est calculable au moyen d'une formule mathématique quelque peu complexe. Toutefois, et bien que je ne vous demande d'en retenir que le résultat, je vous fournis ici sa démonstration :

$$\begin{aligned}2^{-m} &\leq 10^{-n} \\ \Leftrightarrow \text{Log } 2^{-m} &\leq \text{Log } 10^{-n} \\ \Leftrightarrow -m \text{Log } 2 &\leq -n \text{Log } 10\end{aligned}$$

Or $\text{Log } 10=1$
On trouve donc

$$\begin{aligned}m \text{Log } 2 &\geq n \\ \Leftrightarrow m &\geq \frac{n}{\text{Log } 2} \\ \Leftrightarrow m &\geq n \cdot 3,32\end{aligned}$$

Par exemple si l'on souhaite traduire 0,3 en binaire avec une précision de 10^{-4} , on trouve une précision binaire de $m \geq 4 \cdot 3,32$ d'où $m \geq 13,28$.

On prendra donc $m=14$ d'où la traduction de 0,3 en binaire :

$$(0,3)_{10} = (0,0100110011001)_2$$

Cette série impressionnante de "0" et de "1" permet de faire en sorte que le résultat de la conversion du nombre binaire en décimal soit compris dans une fourchette allant de 0,3001 à 0,2999. Le nombre binaire obtenu plus haut a pour conversion décimale 0.29992675...

On verra plus tard une autre représentation des nombres à virgule, la représentation à virgule flottante.

Les nombres signés.

L'introduction d'un signe en binaire n'est pas une chose simple, en effet, la notion de + ou de - est difficilement compréhensible par un composant électronique. Le fait même d'inclure un signe a fait l'objet de nombreuses analyses. On définit ainsi 3 méthodes que nous allons analyser.

L'ajout d'un bit.

Cette méthode consiste à ajouter un bit au début du nombre binaire, ce bit étant à 0 si le nombre est positif et à 1 s'il est négatif. Cette méthode n'est pas exploitable mathématiquement pour 2 raisons :

- Il y a 2 représentations pour 0 (par exemple pour un nombre de 4 bits avec un signe en 5^{ème} bit : 10000 et 00000).
- On ne peut pas effectuer les opérations mathématiques classiques directement.

Par exemple la représentation d'un nombre sur 4 bits (par exemple 12) se code en positif 01100 (+12 codé sur 5 bits) et en négatif 11100 (-12 codé sur 5 bits). Or la somme de +12 et de -12 donne 01000 (toujours sur 5 bits) ou 101000 si l'on code sur 6 bits. Ce résultat est dans les 2 cas différents de 0.

Le complément à 1.

Cette méthode consiste à traiter le signe dans le nombre. Le signe étant le bit de poids fort. Prenons un exemple, soit un nombre relatif codé sur 5 bits. Si ce nombre est positif, on a le bit de poids fort à 0. La représentation du nombre +12 sur 5 bits est donc 0 (bit de signe) 1100 (valeur), la représentation de -12 est alors 1 (bit de signe) 0011 (valeur) c'est à dire l'inverse bit à bit de +12 en remplaçant tous les 0 par des 1 et tous les 1 par des 0.

L'inversion binaire s'appelant la complémentation, on trouve ici le nom de notre méthode.

Ainsi tout nombre négatif commence par un 1 (comme pour la méthode précédente) mais cette fois ci, on peut effectuer des calculs puisque $-12+12$ donne $10011+01100=11111$ qui est le complément de 00000 qui vaut 0 en décimal.

Un autre exemple de calcul nous permet de nous rendre compte de cette capacité, ajoutons 1 à -12 (ce qui doit donner -11) : $-12+1$ se code $10011+1=10100$. Ce résultat est négatif (le bit de poids fort est à 1), pour en connaître la valeur absolue, il faut le complémenter d'où 10100 devient 01011 soit 11. Le résultat de notre addition est donc -11.

Toutefois, ici encore, on a 2 représentations pour la valeur 0, on a d'une part $0+=00000$ (codé sur 5 bits) et $0-=11111$ (toujours codé sur 5 bits). Ceci n'étant pas convenable, on a développé une autre méthode.

Le complément à 2

Cette méthode est la seule utilisable mathématiquement, Elle permet une utilisation des nombres signés avec une représentation unique du zéro et la possibilité d'effectuer des calculs.

Son principe est simple, la représentation négative d'un nombre est réalisée par la formule suivante :

$$-N = \bar{N} + 1$$

où \bar{N} est le complément de N.

Par exemple convertissons 17 en binaire, $(17)_{10} = (10001)_2$. Ce nombre est une valeur absolue. La valeur +17 s'écrit 010001 en la codant sur 6 bits. Par la méthode du complément à 2 on peut écrire que -17 se traduit par le complément de 010001 +1 soit $101110+1=101111$ toujours sur 6 bits.

Attention, l'utilisation de nombres signés impose des restrictions très importantes, on ne peut en effet pas utiliser à tort et à travers les formes complémentées. Avant d'effectuer une quelconque opération, il convient de définir la taille de l'espace de travail, c'est à dire le nombre de bits utilisés. Par exemple si l'on utilise une représentation signée sur 8 bits de -17, son code devient 11101111, c'est à dire que l'on a réalisé une extension de signe.

| | | | | | | | | |
|---------------------|---|---|---|---|---|---|---|---|
| -17 codé sur 6 bits | | | 1 | 0 | 1 | 1 | 1 | 1 |
| -17 codé sur 7 bits | | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| -17 codé sur 8 bits | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

etc.

Extension de signe

On utilisera cette règle et on apprendra à en maîtriser les effets dans les exemples détaillés plus loin dans ce paragraphe. Mais commençons par récapituler un peu les principes du code complément à 2. Et démarrons par une énumération du code complément à 2 des nombres binaires exprimés sur 4 bits.

| Valeur décimale | Valeur binaire | | | |
|-----------------|----------------|---|---|---|
| +7 | 0 | 1 | 1 | 1 |
| +6 | 0 | 1 | 1 | 0 |
| +5 | 0 | 1 | 0 | 1 |
| +4 | 0 | 1 | 0 | 0 |
| +3 | 0 | 0 | 1 | 1 |
| +2 | 0 | 0 | 1 | 0 |
| +1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

| Valeur décimale | Valeur binaire | | | |
|-----------------|----------------|---|---|---|
| -1 | 1 | 1 | 1 | 1 |
| -2 | 1 | 1 | 1 | 0 |
| -3 | 1 | 1 | 0 | 1 |
| -4 | 1 | 1 | 0 | 0 |
| -5 | 1 | 0 | 1 | 1 |
| -6 | 1 | 0 | 1 | 0 |
| -7 | 1 | 0 | 0 | 1 |
| -8 | 1 | 0 | 0 | 0 |

Sur ce tableau, on peut se rendre compte du rapport entre le nombre de bits utilisés et les valeurs entre lesquelles évoluent les nombres. Pour un code à n bits, on peut aller de $+(2^{n-1}-1)$ à -2^n .

On peut maintenant s'entraîner à effectuer des calculs à l'aide du code complément à 2. Ainsi, on tentera de définir les cas qui peuvent poser des problèmes et ceux qui au contraire n'en posent pas.

Commençons par l'addition.

| | | | |
|--------------------------|------|--------|--------------------------|
| 0 1 1 0 0 1 0 0 | 100 | 50 | 0 0 1 1 0 0 1 0 |
| + 0 1 0 1 1 0 1 0 | +90 | +(-75) | + 1 0 1 1 0 1 0 1 |
| 1 0 1 1 1 1 1 0 | -66 | -25 | 1 1 1 0 0 1 1 1 |
| | | | |
| 1 0 1 1 0 1 0 1 | -75 | -66 | 1 0 1 1 1 1 1 0 |
| + 0 1 1 0 0 1 0 0 | +100 | +(-75) | + 1 0 1 1 0 1 0 1 |
| 1 0 0 0 1 1 0 0 1 | 25 | 115 | 1 0 1 1 1 0 0 1 1 |

Légende :



Calcul aboutissant à un résultat faux
Bit non pris en compte

Analysons nos résultats, on réalise des calculs sur 8 bits, on ne doit donc tenir compte que des 8 premiers termes, la présence d'un 9^{ème} terme n'est pas en soit une erreur, seule la présence d'un 9^{ème} terme contenant une information sur le signe pose problème.

Ces additions nous permettent donc de voir que l'addition de 2 nombres de même signe peut entraîner un dépassement, c'est à dire un résultat dont le signe n'est pas valable. Par contre un calcul avec des nombres de signe opposés ne pose pas de problème.

Passons maintenant à la soustraction.

| | | | |
|--------------------------|------|--------|--------------------------|
| 0 1 1 0 0 1 0 0 | 100 | 100 | 0 1 1 0 0 1 0 0 |
| - 0 1 0 1 1 0 1 0 | -90 | -(-75) | - 1 0 1 1 0 1 0 1 |
| 0 0 0 0 1 0 1 0 | 10 | -81 | 1 0 1 0 1 1 1 1 |
| | | | |
| 1 0 1 1 0 1 0 1 | -75 | -66 | 1 0 1 1 1 1 1 0 |
| - 0 1 1 0 0 1 0 0 | -100 | -(-75) | - 1 0 1 1 0 1 0 1 |
| 1 0 1 0 1 0 0 0 1 | 81 | 9 | 1 0 0 0 0 1 0 0 1 |

Légende :



Calcul aboutissant à un résultat faux
Bit non pris en compte

On retrouve pour les soustractions les mêmes problèmes que pour l'addition. Seule la soustraction de 2 termes de même signe ne pose pas de problèmes. La soustraction de 2 termes de signes opposés peut engendrer un dépassement.

Remarque : Pour connaître la valeur absolue d'un nombre négatif, il suffit d'appliquer la formule de conversion déjà étudiée, c'est à dire le complémenter puis lui ajouter 1.

Remarque: On peut aussi multiplier ou diviser des nombres signés si l'on ne dépasse pas la taille maximum définie pour le calcul.

Exercices.

Essayez de répondre aux questions qui vous sont posées sans vous servir de calculatrice.

| Enoncé | Résultat | Correction |
|--|----------|---|
| Codez 0,124 en binaire naturel avec une précision de 10^{-5} | | 0,0001111110 11111 <i>soit 0,12399...</i> |
| Codez 0,01 en binaire naturel avec une précision de 10^{-3} | | 0,000000101 <i>soit 0,009...</i> |
| Donnez la valeur binaire de -31 en complément à 2 sur 6 bits | | 31 → 011111 -31 → 100001 |
| Donnez la valeur binaire de -112 en complément à 2 sur 8 bits | | 112 → 01110000 -112 → 10010000 |

Effectuez les opérations suivantes et entourez les résultats valables

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0 \\ +\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0 \\ \hline \end{array}$$

$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1 \\ +\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \\ \hline \end{array}$$

0 1 1 1 0 1 1 0

Réponse Réponse

1 0 0 0 0 1 1 1

$$\begin{array}{r} 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1 \\ -\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0 \\ \hline \end{array}$$

$$\begin{array}{r} 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1 \\ -\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1 \\ \hline \end{array}$$

0 1 0 1 0 0 1 1

Réponse Réponse

1 1 0 1 1 0 0 0

Donnez la valeur décimale des nombres suivants (sur 10 bits).

| Enoncé | Résultat | Correction |
|------------|----------|------------|
| 1010001101 | | -371 |
| 1110100011 | | -93 |

Ajoutez les 2 nombres signés suivants sur 10 bits.

Premier nombre : 0100100100 (exprimé sur 10 bits)

Second nombre : 110110 (exprimé sur 6 bits)

Réponse : $0100100100 + 111110110 = 0100011010$

Les bases 8 et 16.

On a pu voir que le binaire est particulièrement fastidieux à utiliser, en effet, le nombre de bits utilisés est souvent trop important pour permettre une lecture simple des valeurs. Aussi, on a décidé de s'intéresser à des formes condensées du binaire. Ces bases sont l'octal et l'hexadécimal.

Attention, que vous utilisiez l'octal ou l'hexadécimal, vous travaillez en binaire. Les règles du binaire s'appliquent donc à l'octal et à l'hexadécimal.

Pour passer du binaire à l'octal ou à l'hexadécimal, il suffit de décomposer le code binaire en paquets de 3 bits pour l'octal (octal veut dire base 8 or 8 c'est 2^3 , donc un chiffre d'octal c'est 3 bits) ou de 4 bits pour l'hexadécimal (hexadécimal veut dire base 16 or 16 c'est 2^4 , donc un chiffre d'hexadécimal c'est 4 bits), en partant du bit de poids le plus faible.

Prenons un exemple :

| | | | | | | | |
|---------|-----|-------|-------|-------|-------|-------|-------|
| binaire | 1 0 | 1 0 0 | 1 0 1 | 1 1 0 | 1 0 1 | 0 1 1 | 1 1 0 |
| octal | 2 | 4 | 5 | 6 | 5 | 3 | 6 |

| | | | | | |
|---------|---------|---------|---------|---------|---------|
| binaire | 1 0 1 0 | 0 1 0 1 | 1 1 0 1 | 0 1 0 1 | 1 1 1 0 |
| hexa. | A | 6 | D | 6 | E |

En hexadécimal, on a des chiffres qui peuvent dépasser 10 (on va jusqu'à 15), il faut donc prévoir des chiffres au-dessus de 10. Pour cela, on utilise les premières lettres de l'alphabet. Ainsi, on a les codes suivants :

| | | | | | | | | | | | | | | | | |
|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| binaire | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| hexadécimal | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Exercices.

Convertissez les termes suivants de binaire ou en hexadécimal.

| Enoncé | Résultat | Correction |
|---------------------|----------|--------------------------|
| 0101101011010001101 | | 2D68D |
| AFE59 | | 101011111110 01011001 |

Effectuez en hexadécimal les opérations sur 12 bits signés, encadrez les réponses valables.

$$\begin{array}{r}
 \text{F F E} \\
 + 0 \text{ A B} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 0 1 2 \\
 + 7 \text{ C F} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 2 \text{ F D} \\
 - 3 \text{ E 8} \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 \text{F A 7} \\
 - 0 6 5 \\
 \hline
 \end{array}$$

Réponse **0 A 9** 8 0 E **F 1 5** **F 4 2**

L'algèbre de BOOLE.

Comme vous connaissez maintenant parfaitement l'arithmétique binaire, il convient de s'intéresser à l'algèbre. Développé suite aux travaux du mathématicien anglais George BOOLE, sur la logique binaire, l'algèbre de BOOLE est la forme d'algèbre la plus adaptée au binaire. Elle est composée des opérateurs classiques que nous avons déjà vu, mais aussi de trois nouveaux opérateurs qui n'ont pas d'équivalent simple en base 10.

Les opérateurs booléens jouent sur une notion de rapport logique entre les termes binaires. Un bit peut être vrai ou faux selon sa valeur ainsi dans un cas de logique positive un bit à 1 est considéré comme vrai et un bit à 0 comme faux (et le contraire en logique négative).

Ainsi on peut effectuer des opérations logiques entre des termes binaires. Et pour cela M. BOOLE a défini 3 opérateurs logiques. Mais avant de les détailler, il convient de parler de l'outil le plus adapté à l'algèbre booléenne, les tables de vérité.

Les tables de vérité.

Une table de vérité c'est l'énumération de toutes les combinaisons des variables (dans un groupe de colonnes à gauche du tableau) tandis que la ou les colonnes de droites sont obtenues par application des opérateurs booléens. On verra plus loin quelques exemples de tables de vérités.

L'opérateur NON (opérateur de complémentation).

Présentation algébrique.

L'opérateur de complémentation, vous l'avez déjà utilisé dans les paragraphes précédents en particulier pour le calcul des nombres signés puisque l'utilisation du code complément à 2 nécessite la complémentation du terme binaire.

On a donc défini une représentation de cet opérateur. Il s'agit d'une barre horizontale au-dessus de la variable algébrique.

Le complément de la variable binaire a est \bar{a} .

Pour $a = 0$ on a $\bar{a} = 1$

et pour $a = 1$ on a $\bar{a} = 0$.

(Remarque $\overline{\bar{a}} = a$).

Table de vérité.

| a | \bar{a} |
|-----|-----------|
| 0 | 1 |
| 1 | 0 |

L'opérateur ET.

L'opérateur logique ET effectue une opération sur au moins 2 termes binaires. Il correspond à l'opérateur d'intersection entre 2 ensembles. On peut bien sur présenter sa forme algébrique qui est :

$$a \bullet b = c$$

Le \bullet est l'opérateur logique ET.

Mais la représentation la plus explicite est le format table de vérité.

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Table de vérité de la fonction ET.

On remarquera que les équations suivantes sont des solutions simplificatrices.

$$a \cdot 0 = 0 \text{ mais aussi } a \cdot 1 = a$$

$$a \cdot a = a \text{ mais aussi } a \cdot \bar{a} = 0$$

Expliquons donc les règles de cet opérateur. Pour que A ET B soit vrai, il faut que et A et B soient tous les 2 vrais. Si l'un des deux est faux, le résultat ne peut être que faux. Un peu comme ce petit jeu qui consiste à trouver le bon chemin en questionnant deux personnes, l'une qui dit toujours la vérité, l'autre qui ment toujours.

La solution logique consiste à demander à l'un des 2 (n'importe lequel) de demander à l'autre la bonne direction. Que l'on pose cette question au menteur ou à celui qui dit la vérité, inmanquablement le résultat sera faux puisqu'au l'un aura dit la vérité (ou aura répété le mensonge de l'autre) et l'autre aura menti (ou aura déformé la réalité de l'autre).

On peut aussi avoir plus de 2 entrées. Par exemple, si on a une fonction ET à 3 entrées, on peut la décomposer en 2 fonctions ET à 2 entrées :

| a | b | d | c | d | e |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

| a | b | c | e |
|---|---|---|---|
| 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 | 0 |
| 1 | 1 | 1 | 1 |

L'opérateur OU.

L'opérateur logique OU effectue une opération sur au moins 2 termes binaires. Il correspond à l'opérateur d'union entre 2 ensembles. On peut bien sur présenter la forme algébrique qui est :

$$a + b = c$$

Le + est l'opérateur logique OU

Mais la représentation la plus explicite est le format table de vérité.

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Table de vérité de la fonction OU.

On remarquera que les équations suivantes sont des solutions simplificatrices.

$$a + 0 = a \quad \text{mais aussi} \quad a + 1 = 1$$

$$a + a = a \quad \text{mais aussi} \quad a + \bar{a} = 1$$

Expliquons donc les règles de cet opérateur. Pour que A OU B soit vrai, il faut que soit A, soit B soient vrais. Pour que le résultat soit faux, il faut que A et B soient faux. A partir du moment où au moins l'un des 2 est vrai, le résultat est vrai.

Maintenant pour une fonction logique OU à 4 entrées, on peut réaliser cette fonction en utilisant 3 portes OU à 2 entrées. On obtient alors les tables de vérité suivantes :

| a | b | e | c | d | f | e | f | g |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| a | b | c | d | g |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Combinaison de fonctions logiques.

On peut combiner plusieurs fonctions logiques pour réaliser des fonctions plus complexes. Par exemple la fonction NON ET est composée de la fonction ET suivie de la fonction NON. Cela donne donc la table de vérité suivante :

| | | | | | | | |
|----|---|---|-----|---|--------|---|---|
| a | b | c | c | d | a | b | d |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| ET | | | NON | | NON ET | | |

D'où l'équation booléenne $d = \overline{a \cdot b}$

On peut aussi réaliser la fonction NON OU qui est composée de la fonction OU suivie de la fonction NON. Cela donne la table de vérité suivante :

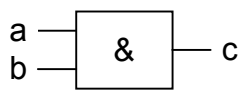
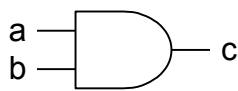
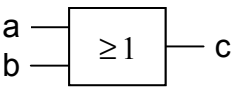
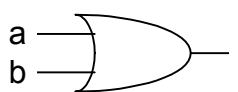
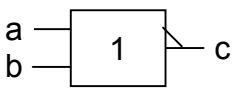
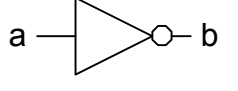
| | | | | | | | |
|----|---|---|-----|---|--------|---|---|
| a | b | c | c | d | a | b | d |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| OU | | | NON | | NON OU | | |


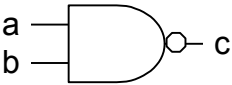
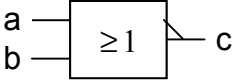
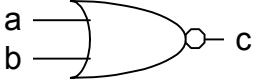
D'où l'équation booléenne $d = \overline{a + b}$

La combinaison de fonction impose de définir des règles permettant de simplifier des équations complexes pour atteindre une décomposition en éléments simples. Mais intéressons-nous à la représentation graphique des fonction logiques.

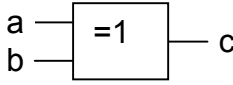
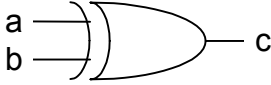
Représentation graphiques.

On a défini une forme de représentation graphique des fonctions logiques booléennes. Cette représentation utilise 2 modèles, l'un français, l'autre américain. Si en France, le seul modèle théoriquement utilisable est français, la réalité est plutôt une utilisation de la représentation américaine.

| Fonction | Modèle Français | Modèle Américain |
|----------|---|---|
| ET |  |  |
| OU |  |  |
| NON |  |  |

| | | |
|--------|---|---|
| NON ET |  |  |
| NON OU |  |  |

Il existe aussi une autre fonction, la fonction addition ou OU EXCLUSIF, qui peu s'exprimer grâce à l'équation booléenne $c = a \cdot \bar{b} + \bar{a} \cdot b$. Sa table de vérité et sa représentation en norme française et américaine est donc :

| <table border="1"> <thead> <tr> <th>a</th> <th>b</th> <th>c</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table> | a | b | c | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | b | c | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | |

Les outils booléens.

L'algèbre de BOOLE est comme on l'a vu et comme on le verra plus loin l'outil par excellence de la logique binaire. Toutefois, il n'est qu'une représentation algébrique des termes binaires. On utilise donc des outils, comme les tables de vérités, pour simplifier les calculs en algèbre binaire.

Table de vérités et fonctions algébriques.

La table de vérité est la méthode la plus simple pour représenter des équations booléennes, un peu comme le tracé d'une courbe lorsque l'on travail avec les équations classiques des mathématiques. Il convient donc d'apprendre à passer d'une équation booléenne à la table de vérité équivalente, et inversement.

Commençons par utiliser des équations pour créer des tables de vérités. Une équation booléenne code tous les cas où la sortie est à l'état actif ("1" si la sortie est en logique vraie, "0" en logique fausse ou complémentée).

Ainsi la fonction logique $\bar{a} + \bar{b} \cdot c = d$ signifie que $d=1$ si $a=0$ et $d=1$ si on a à la fois $b=0$ et $c=1$.

Si l'on réalise un tableau avec les entrées a, b et c, et en sortie la variable d, on obtient alors la table de vérité de la fonction, comme présenté si contre.

| a | b | c | d |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

On peut constater 2 choses à partir de ce tableau, d'une part, la fonction OU sépare les différents cas où la sortie est à l'état actif.

D'autre part, on peut aussi se rendre compte que si un élément n'est pas présent dans un des membres de la fonction, cela veut dire qu'il n'intervient pas. Par exemple, quand on écrit $\bar{b} \cdot c$, cela signifie que quelque soit la valeur de a, il suffit que b val 0 et c, 1 pour que le résultat soit à 1

Inversement, à partir d'une table de vérité, on peut trouver une forme de l'équation booléenne qui en est le fondement.

Ainsi, à partir de la table de vérité ci dessous, on peut écrire :

| a | b | c | d |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

d est égal à zéro dans 2 cas, et à 1 dans 6 cas, on va donc étudier \bar{d} (c'est à dire d=0). On a d=0 pour a=0, b=0 et c=1 ou pour a=1, b=0 et c=1. Cela donne donc $\bar{d} = \bar{a} \cdot \bar{b} \cdot c + a \cdot \bar{b} \cdot c$. On peut simplifier cette équation en mettant en facteur $\bar{b} \cdot c$. On peut donc écrire $\bar{d} = \bar{b} \cdot c \cdot (a + \bar{a})$. Or $a + \bar{a}$ vaut 1 donc on peut écrire $\bar{d} = \bar{b} \cdot c$.

Ceci dit, on cherche d et pas \bar{d} .

Le théorème de DE MORGAN.

Le théorème de DE MORGAN permet de créer des liens entre les formes complémentées et des formes plus simples.

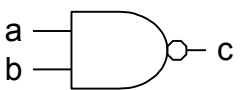
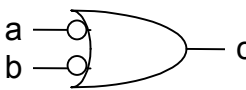
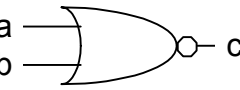
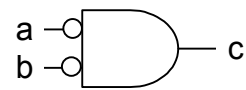
Analysons un peu ce théorème :

$$\overline{a+b} = c, \text{ soit en appliquant DE MORGAN : } \overline{a+b} = \bar{a} \cdot \bar{b} = c$$

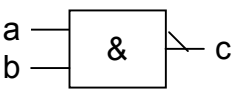
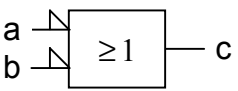
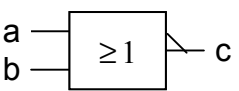
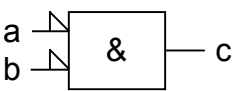
$$\overline{a \cdot b} = c, \text{ soit en appliquant DE MORGAN : } \overline{a \cdot b} = \bar{a} + \bar{b} = c$$

Cette règle permet de trouver des formes nouvelles pour les fonctions NON ET et NON OU. Effectivement, si on utilise les 2 équations précédentes, on peut se rendre compte que $\overline{a+b} = \bar{a} \cdot \bar{b}$. D'où les nouvelles représentations suivantes.

Norme américaine

| | | |
|--------|---|---|
| NON ET |  |  |
| NON OU |  |  |

Norme française

| | | |
|--------|---|--|
| NON ET |  |  |
| NON OU |  |  |

En utilisant le théorème de DE MORGAN, on arrive à simplifier quelque peu les équations booléennes. Par exemple, si l'on revient sur le paragraphe traitant du passage des tables de vérité vers les équations booléennes, on a obtenu $\bar{d} = \bar{b} \cdot c$ donc en utilisant le théorème on trouve $d = \overline{\bar{b} \cdot c} = \bar{\bar{b}} + \bar{c} = b + \bar{c}$.

Il n'en reste pas moins que le théorème de DE MORGAN ne permet pas forcément de traiter rapidement des cas compliqués. On est donc contraint d'utiliser une autre méthode dont la puissance est bien supérieure, le tableau de KARNAUGH.

Les tableaux de KARNAUGH.

Le tableau de KARNAUGH est l'outil par excellence pour la simplification des équations booléennes. Ces tableaux sont réalisés en plaçant en abscisse et en ordonnée les variables exprimées en code GRAY, et dans les cases ainsi créées les valeurs de la variable dont on cherche l'équation.

On doit ensuite encadrer l'ensemble des termes à 1 consécutifs (ou à 0) du tableau en n'utilisant que des groupes dont le nombre de 1 (ou de 0) est une puissance de 2 et en essayant de les prendre le plus gros possible.

Mais prenons plutôt un exemple.

| a | b | c | d |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

| | | bc | | | |
|---|---|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| a | 0 | 0 | 1 | 1 | 0 |
| | 1 | 0 | 1 | 0 | 0 |

En utilisant la table de vérité, on peut écrire :

$$d = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot c$$

Ce que l'on peut simplifier par :

$$d = \bar{a} \cdot c \cdot (\bar{b} + b) + a \cdot \bar{b} \cdot c = \bar{a} \cdot c + a \cdot \bar{b} \cdot c$$

C'est la forme la plus simplifiée que l'on puisse obtenir par cette méthode.

Par contre, pour obtenir les termes qui permettent de construire l'équation liée au tableau de KARNAUGH, il faut regarder un à un les encadrements que nous avons réalisés.

On peut ainsi se rendre compte par exemple, que pour le bloc vertical, $b=0$, $c=1$ et a pouvant être soit à 1 soit à 0. On en déduit que l'équation de ce bloc est $\bar{b} \cdot c$. De même pour le bloc horizontal, on voit que $a=0$, $c=1$ et b pouvant être soit à 1 soit à 0. Donc on peut écrire que l'équation de ce bloc est $\bar{a} \cdot c$.

Ainsi le tableau de KARNAUGH donne pour équation $d = \bar{b} \cdot c + \bar{a} \cdot c$. Ce qui est une forme plus intéressante puisque contenant moins de termes que celle obtenue par la table de vérité.

Mais prenons tout de suite un autre exemple.

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

D'ou le tableau de KARNAUGH suivant.

| | | cd | | | |
|----|----|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| ab | e | 00 | 01 | 11 | 10 |
| | 00 | 1 | 0 | 0 | 1 |
| | 01 | 1 | 1 | 1 | 1 |
| | 11 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 0 | 1 | |

$$e = \bar{b} \cdot \bar{d} + \bar{a} \cdot b + b \cdot c \cdot d$$

On peut constater que :

- Primo les 2 groupes réalisés en pointillés sont inutiles puisqu'ils n'apportent rien de nouveau (les termes qu'ils entourent sont déjà inclus dans d'autres groupes).
- Secundo les termes qui forment les 4 angles peuvent être inclus dans un même groupe puisque les extrémités gauche et droite, ainsi que haute et basse se rejoignent. Donc les 4 coins se rejoignent.

Enfin voyons un dernier exemple avant la fin de ce chapitre.

| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

| | | de | | | |
|-----|-----|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| f | 000 | 1 | 0 | 0 | 1 |
| | 001 | 0 | 1 | 1 | 0 |
| 011 | 011 | 0 | 0 | 1 | 0 |
| | 010 | 1 | 1 | 1 | 1 |
| abc | 110 | 0 | 1 | 1 | 0 |
| | 111 | 1 | 0 | 1 | 1 |
| 101 | 101 | 1 | 1 | 1 | 1 |
| | 100 | 0 | 0 | 0 | 0 |

$$f = \bar{a} \cdot \bar{c} \cdot \bar{e} + \bar{b} \cdot c \cdot e + c \cdot d \cdot e + b \cdot \bar{c} \cdot e + a \cdot c \cdot \bar{e}$$

Ce qui peut se simplifier par

$$f = \bar{e} \cdot (\bar{a} \oplus c) + e \cdot (b \oplus c) + c \cdot d \cdot e$$

$$\text{Rem : } \overline{a \oplus c} = a \cdot c + \bar{a} \cdot \bar{c}$$

On peut remarquer dans ce tableau que j'ai représenté en traits plus foncés des axes de symétrie qui servent à recréer des blocs. Ainsi les blocs délimités par de gros pointillés sont obtenus par utilisation de l'axe de symétrie centrale (en face des lettres abc).

Cet avantage a bien sur un défaut, on ne peut pas traverser un axe de symétrie sans justement la respecter. Ainsi la partie au dessus de l'axe de symétrie doit être égale à celle au dessous, tout en respectant la règle qui veut que le nombre d'éléments doit être le plus grand possible en restant une puissance de 2.

Exercices.

A partir de la table de vérité suivante, remplissez le tableau de KARNAUGH et donnez l'équation de la fonction. Représentez en suite la fonction en utilisant la norme américaine.

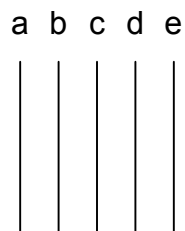
| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | | | |
|----|----|-----|-----|-----|-----|-----|-----|-----|-----|
| | | cde | | | | | | | |
| | | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| ab | 00 | | | | | | | | |
| | 01 | | | | | | | | |
| | 11 | | | | | | | | |
| | 10 | | | | | | | | |

f=

Correction : $f = be + \bar{b} \cdot d \cdot \bar{e}$

Représentation :



————— f