Logique Combinatoire
Circuits Combinatoires

Ce document a pour objectif de vous familiariser avec les portes logiques et la logique booléenne afin de comprendre comment les circuits combinatoires sont construits.

La logique combinatoire est un domaine des mathématiques visant à étudier les fonctions afin de les formaliser et supprimer le besoin des variables. On retrouve cette idée dans les langages de programmation fonctionnels. Plus concrètement en architecture, on utilise cette théorie mathématique pour simplifier les formules logiques représentants des circuits combinatoires. Ces circuits combinatoires (ou *combinational circuits* en anglais) sont des composants générant des valeurs en sortie selon les valeurs données en entrées : il s'agit simplement de fonctions calculant chaque bit de sortie à partir de formules basées sur les entrées.

La logique booléenne (liée à l'Algèbre de Boole) est le domaine mathématique permettant de définir et travailler avec les différents opérateurs logiques manipulant des valeurs binaires : *vrai* (1) et *faux* (0). Les portes logiques (ou *logic gate* en anglais) sont des composants élémentaires permettant de réaliser les circuits combinatoires. Plusieurs notations différentes permettent de représenter les portes logiques. Ce document présente les portes logiques ainsi que leurs notations européennes (les versions rectangulaires contenant un opérateur) et américaines (aussi appelées « symboles distinctifs », c'est-à-dire les versions arrondies sans opérateur).

Il faut principalement retenir que les circuits combinatoires permettent de déterminer l'état des bits en sortie du composant exclusivement à partir de l'état des bits en entrée du composant, et cela grâce aux portes logiques servant à appliquer concrètement la logique booléenne. Afin de limiter le nombre de portes logiques utilisées dans ces circuits combinatoires, des simplifications peuvent être appliquées aux formules logiques. D'autres techniques permettent également de réduire l'usage de certaines variables, voire, de détecter qu'elles sont inutiles.

1 Portes et Fonctions Logiques

Les *portes logiques* sont des composants en électronique numérique (l'électronique qui manipule des données numériques, donc des 0 et des 1) représentants les *fonctions logiques* de façon concrète. Par exemple, pour la *Porte Logique NON*, vous pouvez trouver comme dénomination la *Fonction Logique NON*. Ces portes, ou fonctions, logiques s'appuient directement sur la logique booléenne.

Par soucis de simplification, on omettra parfois le terme *logique* dans les expressions. Par exemple, pour la *Porte Logique XOR*, vous pouvez trouver comme dénomination abrégée la *Porte XOR*.

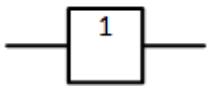
Les schémas européens et américains des portes logiques seront accompagnés de la formule présentant l'opérateur logique ainsi que la table de vérité associée.

1.1 Portes Logiques à une entrée

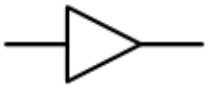
1.1.1 Porte Logique OUI / YES

Il s'agit d'une porte logique agissant exactement comme la fonction identité : l'état d'entrée est recopié en sortie. Cette porte logique s'appelle également un *tampon* (*buffer* en anglais).

Porte OUI/YES



Symbole européen



Symbole américain

$X = A$

A	X
0	0
1	1

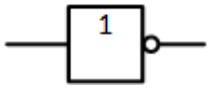
Cette porte logique ne présente que peu d'intérêt. Néanmoins, il existe des raisons techniques de s'en servir (comme réhausser un signal dans certains cas), voire, certains autres domaines indépendants de l'électronique s'appuient eux aussi sur les outils logiques et nécessitent l'équivalent d'une porte recopiant son état d'entrée.

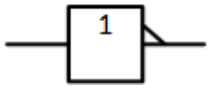
1.1.2 Porte Logique NON / NOT

Il s'agit d'une porte logique effectuant la négation de la valeur d'entrée, c'est-à-dire que l'on applique le complément à 1 sur le bit inséré en entrée. Cette porte logique s'appelle également un *inverseur* (*inverter* en anglais).

Porte NON/NOT

Symbole européen (1)





Symbole européen (2)

Symbole américain

$X = \overline{A}$

A	X
0	1
1	0

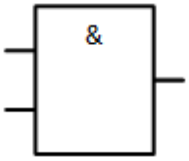
On notera que sur les schémas il s'agit simplement d'ajouter une bulle (ou une barre oblique sur certaines versions européennes) indiquant que l'on applique une inversion à la sortie de la porte logique.
Ainsi, la porte NON n'est qu'une porte OUI à laquelle on applique un inverseur : on effectue l'identité sur le bit de l'entrée, puis on applique le complément à 1.
Dans la représentation algébrique, on remarque qu'il s'agit d'ajouter une barre sur le bit que l'on veut inverser, tout comme dans les probabilités lorsque l'on exprime la négation d'une assertion.

1.2 Portes Logiques à deux entrées

1.2.1 Porte Logique ET / AND

Il s'agit d'une porte logique exigeant que les deux entrées soient simultanément activées pour que la sortie le soit.

Porte ET/AND



Symbole européen

Symbole américain

$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

On remarquera que le symbole algébrique correspondant est un point ($\cdot$) comme pour la multiplication. En effet, en multipliant les bits d'entrée, on obtient le bit de sortie.

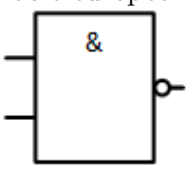
On peut interpréter cette porte de plusieurs manières. La plus évidente consiste à indiquer que les 2 entrées doivent être actives pour que la sortie le soit également (il faut avoir 2 validations).
Une autre en logique négative serait de dire qu'il ne faut aucun désaccord pour activer la sortie.

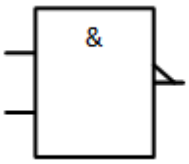
1.2.2 Porte Logique NON-ET / NAND

Il s'agit d'une porte logique effectuant la négation de la porte ET, il faut donc qu'une ou moins des entrées soit activée pour que la sortie le soit. On peut également lire cette porte comme : les deux entrées ne doivent pas être simultanément activées.

Porte NON-ET/NAND

Symbole européen (1)





Symbole européen (2)

Symbole américain

$X = \overline{A \cdot B}$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

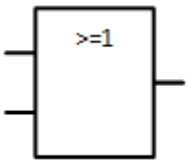
Cette porte peut s'utiliser pour s'assurer que les conditions en entrées ne sont jamais vérifiées ensemble. Au plus, une seule condition sera validée.

On peut utiliser cette porte lorsque l'on scrute des évènements qui ne doivent surtout pas se produire simultanément.

1.2.3 Porte Logique OU / OR

Il s'agit d'une porte logique exigeant qu'au moins une des deux entrées soit activée pour que la sortie le soit également.

Porte OU/OR



Symbole européen

Symbole américain

$X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

On remarquera que le symbole algébrique correspondant est un plus (+) comme pour l'addition. En effet, en additionnant les bits d'entrée, on obtient soit 0, soit une valeur supérieure à 0 qui active la sortie. Dans la représentation européenne, on remarque également cette condition ≥ 1 qui exige que la somme des entrées soit supérieure ou égale à 1.

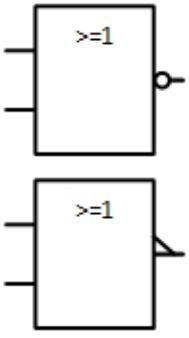
Cette porte peut s'utiliser dans le cas où au moins un évènement doit se produire pour activer la sortie. Ou encore, au moins une condition doit être résolue pour activer la sortie.

1.2.4 Porte Logique NON-OU / NOR

Il s'agit d'une porte logique effectuant la négation de la porte OU, il faut donc qu'aucune des entrées ne soit activée pour que la sortie le soit.

Porte NON-OU/NOR

Symbole européen (1)



Symbole européen (2)

Symbole américain

$X = \overline{A + B}$

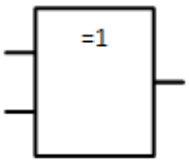
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Un cas d'usage de cette porte est simplement qu'absolument aucun évènement scruté ne doit se produire pour que la sortie soit activée.

1.2.5 Porte Logique OU Exclusif / XOR

Il s'agit d'une porte logique exigeant qu'au plus une des deux entrées soit activée pour que la sortie le soit également. On peut également lire cette porte comme : les deux entrées doivent être dans des états différents.

Porte OU Exclusif/XOR



Symbole européen

Porte OU Exclusif/XOR

Symbole américain

$X = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

On remarquera que le symbole algébrique correspondant est un plus entouré ($\oplus$), car il s'agit effectivement d'une addition, mais d'une addition spéciale. La condition exprimée est qu'une seule entrée soit active, mais ni plus ni moins. La représentation européenne exprime effectivement cela en indiquant comme condition = 1, c'est-à-dire que la somme des entrées doit absolument être strictement égale à 1.

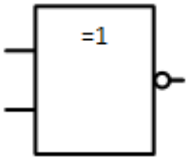
Parmi les autres usages possibles, il s'agirait de vérifier exactement une seule condition pour activer la sortie.

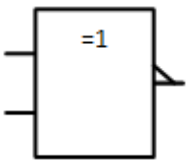
1.2.6 Porte Logique NON-OU Exclusif / NXOR

La porte logique NON-OU Exclusif, aussi appelée *coïncidence*, ou NXOR est la négation de la porte OU Exclusif, il faut donc qu'aucune ou les deux des entrées soient activées pour que la sortie le soit. On peut également lire cette porte comme : les deux entrées doivent être dans le même état. Cette porte logique s'appelle également une *coïncidence*.

Porte NON-OU Exclusif/NXOR

Symbole européen (1)





Symbole européen (2)

Symbole américain

$X = A \odot B = \overline{A \oplus B}$

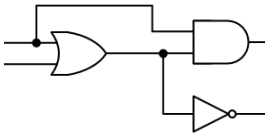
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

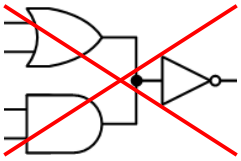
On remarquera que le symbole algébrique correspondant est un point entouré ($\odot$), car il s'agit effectivement d'une multiplication, mais d'une multiplication spéciale.

2 Circuits Logiques & Algèbre Booléenne

Les portes logiques peuvent se combiner ensemble.
Il y a néanmoins deux règles à respecter :

— On peut connecter à une entrée n'importe quelle autre entrée ou sortie
— On ne peut pas connecter une sortie à une autre sortie





On peut simplifier les circuits logiques (en réduisant le nombre de portes logiques) grâce à l'algèbre de Boole, et, comme nous le verrons plus tard, avec les tableaux de Karnaugh.

2.1 Principaux axiomes et théorèmes de l'algèbre Booléenne

L'algèbre Booléenne s'appuie essentiellement sur les trois opérateurs : NON, ET, OU

Priorité	$A + (B \cdot C) = A + B \cdot C$
NON	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A} \cdot A = 0$
ET	$A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$ $A \cdot B = B \cdot A$ $A \cdot A = A$ $A \cdot 1 = A$ $A \cdot 0 = 0$
OU	$A + (B + C) = (A + B) + C = A + B + C$ $A + B = B + A$ $A + A = A$ $A + 0 = A$ $A + 1 = 1$
OU EXCLUSIF	$A \oplus B = \overline{A} \cdot B + A \cdot \overline{B} = \overline{A} \oplus \overline{B}$ $\overline{A} \oplus \overline{B} = A \cdot B + \overline{A} \cdot \overline{B} = \overline{A} \oplus B = A \oplus \overline{B}$ $A \oplus (B \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$ $A \oplus B = B \oplus A$ $A \oplus 0 = A$ $A \oplus 1 = \overline{A}$ $A \oplus A = 0$ $A \oplus \overline{A} = 1$
Distributivité	$A \cdot (B + C) = A \cdot B + A \cdot C$ $A + B \cdot C = (A + B) \cdot (A + C)$
Théorèmes de De Morgan	$\overline{A \cdot B} = \overline{A} + \overline{B}$ $\overline{A + B} = \overline{A} \cdot \overline{B}$
Autres théorèmes	Théorème 1 : $A + A \cdot B = A$ Théorème 2 : $A + \overline{A} \cdot B = A + B$

(Merci David BOUCHET pour le superbe et très complet tableau !)

Il est possible de transformer un circuit de portes logiques en formule d'algèbre booléenne, et inversement, tant que les 2 règles sont respectées.
L'intérêt d'utiliser les axiomes et théorèmes est de pouvoir simplifier une formule, et donc, de produire un circuit beaucoup plus petit.

2.2 Formes canoniques

On peut transformer une expression booléenne de plusieurs manières grâce aux formules précédentes. Parmi ces formes, on trouve les *formes canoniques*. Les formes canoniques impliquent de détailler toutes les variables dans chaque terme de la somme ou du produit.

La première forme canonique est une somme de produits, tandis que la deuxième forme canonique est un produit de sommes.

2.2.1 Mintermes / Première forme canonique / Somme canonique de produits

La *première forme canonique* ou *somme canonique de produits* ou encore *somme des mintermes* est littéralement une somme dont chaque terme est constitué de toutes les variables manipulées multipliées entre elles. On a donc des OU dont les termes sont constitués de variables liées par des ET.

Un *minterme* est un produit de toutes les variables manipulées (complémentée ou non).

Le format est du type :

$$\sum X \cdot Y \cdot Z \;\Rightarrow\; (X \cdot Y \cdot Z) + (X \cdot \overline{Y} \cdot Z) + ...$$

Pour obtenir une somme canonique de produits à partir d'une somme quelconque, il suffit de multiplier à chaque terme incomplet les 2 états possibles de chaque variable manquante.

Par exemple, pour les variables A, B et C :

$A \cdot B \cdot C + \overline{A} \cdot B$

Le terme $\overline{A} \cdot B$ n'est pas un minterme, car il ne contient pas C.
On peut obtenir la somme canonique en multipliant le dernier terme par $(C + \overline{C})$:

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

Ainsi, la formule $A \cdot B \cdot C + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C}$ ne contient que des mintermes, et est la somme canonique de produits.

De même, si l'on avait une formule avec une quatrième variable D, il aurait fallu multiplier chaque terme de la somme où une ou des variables seraient manquantes par la ou les variables dans ses 2 états possibles.

On peut également retrouver une somme canonique de produits à partir d'une table de vérité (et inversement) : chaque terme de la somme correspond aux états *vrais* du résultat, sachant que chaque état inversé d'une variable correspond à un 0.

$$X = A \cdot B \cdot C + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C}$$

$$A \cdot B \cdot C \;\Rightarrow\; A = 1\ ,\ B = 1\ ,\ C = 1$$
$$\overline{A} \cdot B \cdot C \;\Rightarrow\; \overline{A} = 0\ ,\ B = 1\ ,\ C = 1$$
$$\overline{A} \cdot B \cdot \overline{C} \;\Rightarrow\; \overline{A} = 0\ ,\ B = 1\ ,\ \overline{C} = 0$$

A	B	C	X	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\leftarrow \overline{A} \cdot B \cdot \overline{C}$
0	1	1	1	$\leftarrow \overline{A} \cdot B \cdot C$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$\leftarrow A \cdot B \cdot C$

2.2.2 Maxtermes / Seconde forme canonique / Produit canonique de sommes

La *seconde forme canonique* ou *produit canonique de sommes* ou encore *produit des maxtermes* est littéralement un produit dont chaque terme est constitué de toutes les variables manipulées additionnées entre elles. On a donc des ET dont les termes sont constitués de variables liées par des OU.

Un *maxterme* est une somme de toutes les variables manipulées (complémentée ou non).

Le format est du type :

$$\prod X + Y + Z \;\Rightarrow\; (X + Y + Z) \cdot (X + \overline{Y} + Z) \cdot ...$$

Pour obtenir un produit canonique de sommes à partir d'un produit quelconque, il suffit d'additioner à chaque terme incomplet les 2 états possibles de chaque variable manquante.

Par exemple, pour les variables A, B et C :
$(A + B + C) \cdot (\overline{A} + B)$
Le terme $\overline{A} + B$ n'est pas un maxterme, car il ne contient pas C.
On peut obtenir le produit canonique en ajoutant au dernier terme $(C \cdot \overline{C})$:
$(A + B + C) \cdot (\overline{A} + B + \boldsymbol{(C + \overline{C})}) = (A + B + C) \cdot (\overline{A} + B + \boldsymbol{C}) \cdot (\overline{A} + B + \boldsymbol{\overline{C}})$

Ainsi, la formule $(A + B + C) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C})$ ne contient que des maxtermes, et est le produit canonique de sommes.
De même, si l'on avait une formule avec une quatrième variable D, il aurait fallu ajouter, à chaque terme de la somme où une ou des variables seraient manquantes, la ou les variables dans ses 2 états possibles.

On peut également retrouver un produit canonique de sommes à partir d'une table de vérité (et inversement) : chaque terme du produit correspond aux états *faux* du résultat, sachant que chaque état inversé d'une variable correspond à un 1.

$$X = (A + B + C) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C})$$
$$A + B + C \;\Rightarrow\; A = 0\ ,\ B = 0\ ,\ C = 0$$
$$\overline{A} + B + C \;\Rightarrow\; \overline{A} = 1\ ,\ B = 0\ ,\ C = 0$$
$$\overline{A} + B + \overline{C} \;\Rightarrow\; \overline{A} = 1\ ,\ B = 0\ ,\ \overline{C} = 1$$

A	**B**	**C**	**X**	
0	0	0	0	$\leftarrow A + B + C$
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	0	$\leftarrow \overline{A} + B + C$
1	0	1	0	$\leftarrow \overline{A} + B + \overline{C}$
1	1	0	1	
1	1	1	1	

Pour construire les maxtermes, il s'agit de suivre la logique inverse des mintermes.

On peut également le voir comme la recherche de l'état absorbant pour l'opération principale :

- dans les *mintermes*, c'est un OU/l'addition qui est l'opération principale, donc un état *vrai*/1 est absorbant : on va chercher tous les états absorbants pouvant renvoyer 1
- dans les *maxtermes*, c'est un ET/la multiplication qui est l'opération principale, donc un état *faux*/0 est absorbant : on va chercher tous les états absorbants pouvant renvoyer 0

De fait, passer d'une forme canonique à l'autre revient à sélectionner les résultats inverses, en appliquant également l'inverse sur l'état des variables.

2.3 Tableaux de Karnaugh

Pour simplifier les expressions booléennes, on doit s'appuyer sur les formules présentées plus tôt. Néanmoins, il existe une méthode graphique permettant de résoudre cela : les *tableaux de Karnaugh* ou *diagrammes de Karnaugh.*

À partir d'une table de vérité, on va construire un tableau de Karnaugh et y dessiner les plus grands groupes de 1 alignés sous forme de rectangles ou carrés. Ces regroupements permettent de déterminer quelles variables ou états sont constamment vérifiés et ne nécessitent donc pas d'être écrits dans la formule finale. La formule finale n'est donc quasiment jamais une forme canonique.

2.3.1 Construction d'une table de vérité

Pour construire un tableau de Karnaugh, on doit d'abord construire la table de vérité. Cette table s'obtient simplement en testant l'ensemble des combinaisons possibles des variables composant une formule.

$$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$$

A	B	C	X	
0	0	0	1	$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$
0	0	1	1	$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$
0	1	0	0	
0	1	1	1	$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$
1	0	0	1	$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$
1	0	1	0	
1	1	0	1	$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$
1	1	1	1	$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$

2.3.2 Transformation d'une table de vérité en tableau de Karnaugh

Une fois la table de vérité trouvée, il faut la transformer en tableau de Karnaugh en respectant le format des états des variables.

Chaque côté du tableau représente $\frac{N}{2}$ variables, et éventuellement $\frac{N}{2} - 1$ variables si leur nombre est impair. Par exemple, lorsque la formule concerne 4 variables, on va construire un tableau « carré » de 2 variables sur 2, tandis que si la formule concerne 3 variables, on va construire un tableau « rectangulaire » de 1 variable sur 2.

Les variables observées doivent être triées dans des états respectant strictement le code Gray : 00, 01, 11, 10, ...

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$X = A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C$$

La formule contenant 3 variables A, B et C donnera :

BC

	X	**00**	**01**	**11**	**10**	← Code Gray
A	**0**	1	1	1	0	
	1	1	0	1	1	

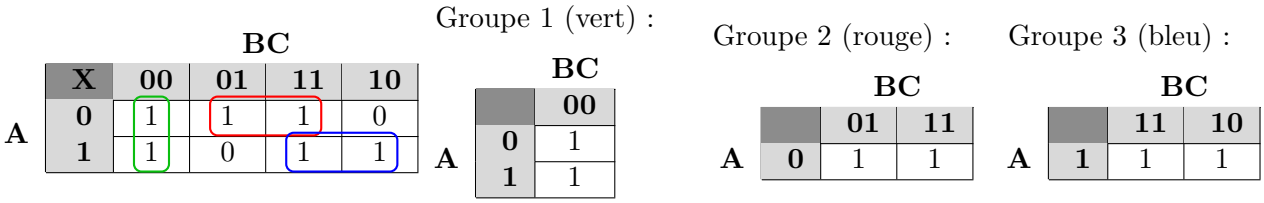
↑ Code Gray

2.3.3 Recherche de groupes dans le tableau de Karnaugh

Une fois le tableau construit, on peut rechercher les groupes de 1 et les encercler en respectant certaines règles :

— ensemble des 1 doivent être dans un ou plusieurs groupes
— Le nombre de cases dans un même groupe doit être une puissance de 2
— Les cercles doivent être de taille maximale
— Le nombre de cercles doit être minimum

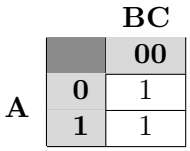
Les groupes peuvent partager quelques éléments, mais ils doivent rester de taille maximum sans englober intégralement un autre groupe. On peut disposer de groupes ne contenant qu'un seul élément.



Chaque groupe correspond à un terme de la formule optimisée. Chaque terme est un produit (un ET) de variables, et les termes sont sommés ensemble.
Une fois les groupes établis, il faut donc prendre chacun des groupes et le transformer en un terme en respectant ces règles :

— Une variable qui change d'état est ignorée/n'apparait pas dans le terme
— Une variable qui reste à 0 doit être retranscrite comme complémentée (avec une barre)
— Une variable qui reste à 1 doit être retranscrite normalement (sans barre)

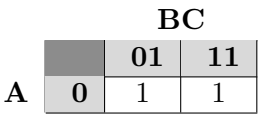
Groupe 1 (vert) :



— A varie entre 0 et 1
— B reste à 0
— C reste à 0

$\Rightarrow \overline{B} \cdot \overline{C}$

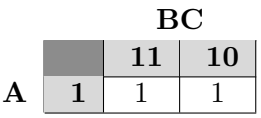
Groupe 2 (rouge) :



— A reste à 0
— B varie entre 0 et 1
— C reste à 1

$\Rightarrow \overline{A} \cdot C$

Groupe 3 (bleu) :



— A reste à 1
— B reste à 1
— C varie entre 0 et 1

$\Rightarrow A \cdot B$

On reconstruit donc la somme de produits suivante :

$$A \cdot B \ + \ \overline{B} \cdot \overline{C} \ + \ \overline{A} \cdot C$$

Ce qui correspond à une versions simplifiée de la formule initiale :

$$X \ = \ A \cdot B + \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot C \ = \ A \cdot B + \overline{B} \cdot \overline{C} + \overline{A} \cdot C$$

On peut donc en déduire un circuit logique simplifié en quantité de portes logiques (et donc une réduction de la quantité de composants requis et d'énergie pour les alimenter et les faire changer d'état).

Attention, le tableau de Karnaugh peut être construit dans n'importe quel ordre : $A \times BC = AB \times C = AC \times B$. Ceci implique que les « bords » du tableau se recoupent : on peut construire un carré à partir des valeurs dans les coins.

Ce document et ses illustrations ont été réalisés par Fabrice BOISSIER en novembre 2023.
Le contenu est très inspiré des supports de cours d'Anne-Sophie DUJARDIN et David BOUCHET.