**Fonctions, portes logiques et algèbre booléenne**

**Définition :**

a**- Variable et fonction logiques** :

-Une variable logique est une variable qui peut prendre deux etats différents ;

soit la valeur « 0 «  soit la valeur » 1 » vrai ou faux (vrai 1 ,faux 0)

-ne fonction logique de variables est une fonction qui a deux etats 0 ou 1

**Opérateurs logiques élémentaires** :

-les opérateurs logiques élémentaires sont la base de la construction des fonctions logiques.

En algèbre classique, on a quatre opérateurs qui sont ; **( + ), (-) ; (x) ; ( : ) :addition,**

**soustraction, produit et quotient.** En alg&bre de Boole, ils sont trois seulement :

ET**(AND**) ; OU(**OR)** ; NON’**(NOW)**

**Exemple :**

 La fonction F est vraie si a est vraie ou si b et c sont vrais ou si d est faux s’écrit comme suit :

 **F= [a OU ( b ET c)OU NON d ]**

 **F= a+ bc + d̅**

**Opérateurs logiques complets ou universels**

On peut expliquer qu’il est possible de synthétiser les trois opérateurs de base avec un seul type d’opérateur que l’on appelle alors « opérateurs complets ou universels**.**

On distingue deux opérateurs universels, qui sont le **NON ET(NAND)** et le **NON OU (NOR).**

**a- NAND :**

**Y=a.b**

**Symbole**

 ****

 **Norme américaine Norme européenne(IEEE)**

**b- NOR :**

**Y=a+b**

**Symbole :**

****

 **Norme américaine Norme européenne (IEEE)**

**Ecriture des fonctions logiques :**

Une fonction logique définie par son expression logique, peut être représentée de plusieurs manières : par table de vérité, expression booléenne, tableau de Karnaugh etc…

**Table de vérité :**

Les combinaisons de n variables logique sont limitées à 2n du fait que les variables ne peuvent prendre que deux états.on peut ainsi représenter la fonction logique à l’aide d’un tableau appelé table de vérité.

Exemple :

Soit la fonction f(a,b)=a+b signifie : F=1 si a=1 ou b=1 ou a=b=1

|  |  |
| --- | --- |
| entrées | Sortie  |
| a b | F=a+b |
| 0 00 11 01 1 | 0 , F(0,0)=01 , F(0,1)=11, F(1,0)=11 F(1,1)=1 |

**Remarque :** la table de vérité comporte 2n lignes (une pour chaque combinaison de variables d’entrées).

**Tableau de Karnaugh** :

Le tableau de Karnaugh est une représentation de la table de vérité en deux dimensions .il comprend 2n cases.

**Exemple : 1**

Soit la fonction f(a,b)=a+b

n=2 variables donc 4cases

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

**Nota benné :**  plus de deux variables, le tableau doit être un volume. Pratiquement on conserve la représentation à deux dimensions, mais il faut respecter une règle qui est l’adjacence de deux lignes ou colonnes, c’est-à-dire on utilise le code Gray (code binaire réfléchi).

**Exemple du tableau à 4 variables :**

 Code Gray codeGray

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

**Fonctions élémentaires de bases :**

**Fonction, opérateur inverseur NON (NO) :**

**Expression**: si a est vrai, f(a) est faux : f(a)=a̅

**Table de vérité :**

 Entrée sortie

|  |  |
| --- | --- |
| a | f(a) |
| 01 | 10 |

**Symbole :**

 

 Symbole usuel (américain) symbole normalisé IEEE

**Schema electrique :**

 

**Fonction, opérateur ET (AND) :**

**Expression**: f(a,b) est vraie si a est vrai ET b est vrai ; f(a,b) =a.b

**Table de vérité** :

n=2 donc 22=4

|  |  |
| --- | --- |
| Entrées  | Sortie |
| a b | F=a.b |
| 0 00 11 01 1 | 0001 |

**Tableau de Karnaugh :**

|  |  |  |
| --- | --- | --- |
|  **a** **b** |  **0** |  **1** |
|  **0** | **0** | **0** |
|  **1** | **0** | **1** |

**Symbole :**

****

Norme américaineNorme européenne (IEEE)

**Schema electrique :**

****

**Fonction, opérateur (porte) OU (OR) :**

**Expression**: f(a,b) est vraie si a est vrai OU b est vrai Ou a et b sont vrais ; f(a,b)=a+b ☹fonction union).

**Table de vérité :**

|  |  |
| --- | --- |
| **Entrées**  | **Sortie** |
| **a b**  | **f=a+b** |
| **0 0****0 1****1 0****1 1** | **0****1****1****1** |

**Tableau de Karnaugh :**

|  |  |  |
| --- | --- | --- |
|  **a** **b** |  **0**  |  **1** |
|  **0** |  **0** |  **1** |
|  **1** |  **1** |  **1** |

**Symbole :**

 ****

 **Norme américaine Norme européenne (IEEE)**

**Schema electrique :**

 ****

**Propriétés des fonctions élémentaires :**

a- **Element neutre** :

-**OR** : a+0=a **0** : Elément neutre pour l’opérateur **OR**

**-AND : a.1=a 1 :** elément neutre pour le **NAND**

**b- Commutativité :**

**OR :** a+b = b+a

**AND :** a.b =b.a

c- **Ass :ociativité :**

**-OR : (a+b)+c = a+(b+c) =a+b+c**

**-AND : (a.b).c=a.(b.c)=a.b.c**

**c-Distributivité :**

 **a.(b+c) =a.b+a.c**

**d-Idempotence :**

**-OR : a+a =a**

**-AND : a.a =a**

**e-complément :**

**a̅̅ =a ; a+a̅ =1   a.a̅ =0**

**Théorèmes de base :**

Théoréme de De Morgan :

-le théoreme s’ecrit de la manière suivante :

f̅ **(a,b,….+,….) =f(a̅,b̅,...●) ,**c'’st à direle complément d'’ne fonction f de variables (a,b,c; ...) s'’btient en remplacants les variables par leur complément et les opérateurs AND (●) par les opérateurs OR (+) et inversement. Donc, on peut énoncé deux théorémes

1-le complément d’une somme logique est égal au produit des compléments :

 **a+ b = a̅ .b̅**

2-Le complément du produit logique est égal à la somme des compléments :

 a●b = a̅ + b̅

**Exemple:**

 - f =ab+c̅d+bd ₺ f̅ = (a̅ + b̅ ).( c+ d̅ ).( b̅ + d̅ )

**Dualité :**

une dualité demoeure vraie si l’on permute les fonctions OR par NAND et les « 0 » par les « 1 ».

 **Exemple :**

- a + a̅.b = a + b ₺ a . (a̅ + b ) = a.b

- a + 1 = 1 ₺ a̅ . 0 = 0

**Formes normales ou formes canoniques des fonctions :**

Les formes canoniques ou formes normales sont des expressions particulières de fonctions logiques,sous formes ET de OU (produits de sommes ou **max termes**) ou de OU de ET (somme de produits **min termes).** Dans chaque terme apparaissen**t toutes les variables.**

**Forme normale disjonctive (FND) :**

**1er theoréme de Chanon :**

Toute fonction logique peut se décomposer en un **OU** de deux **ET** logiqes :

 **f (a,b,c,. ) = a f(1,b,c,.) + a̅. f (0,b,c., )**

En utilisant successivement ce théorème on aboutit à la forme normale disjonctive :

 **f (a,b,c..) = ab f (1,1,c ) + a̅bf (0,1,c ) + ab̅ f (1,0,c...) + a̅b̅ f (0,0,c,…)**

la fonction s’ecrit alors comme **un OU** de toutes les combinaisons possibles **de n** variables pondérées par des « **0**» et « **1**» ( il existe donc deux termes)..les termes pondérés **« 0 » sont éliminés** et reste seulement les termes pondérés par des **« 1 »** . l’expression logique ou booléenne développement de la fonction suivant les « 1 » **appelé mintermes.**

**Exemple :**

Soit le tableau de Karnaugh suivant ; déterminer sa **FND (mintermes,ou forme canonique par rapport au « 1 »).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  **ab** **c** | **00** | **01** | **11** | **10** |
|  **0** | **0** | **0** | **0** | **1** |
|  **1** | **1** | **1** | **0** | **1** |

**f1 (a,b,c) = a̅b̅c̅ .0 +a̅bc̅ . 1 + abc̅ .0 + ab̅c̅ . 1 +a̅b̅c.1 +abc .0 +ab̅c.1**

Ce résultat est obtenu directement de la table de vérité ou du tableau de Karnaugh**.**

**Exemple 2 :**

 **f (a,b,c) = a̅b̅c̅ + a̅b̅c +a̅bc +ab̅c**

Dresser la table de vérité et le tableau de Karnaugh correspondants à cette fonction.

Sous sa forme normale, f s’ecrit :

**F(a,b,c) =a̅b̅c̅.f(0,0,0) + a̅b̅c.f(0,0,1)+ a̅bc̅.f(0,10)+ a̅bc.f(0,1,1) + ab̅c̅ .f(1,0,0) +ab̅c.f(1,0,1)+abc̅ .f(1,1,0)+abc.f(1,1,1)**

**avec:  f(0,0,0) = 1;  f(0,0n1)=1;  f(0,1,0)=0   f(0,1,1)=1**

 **f(1,0,0)=0 ; f(1,0,1)=1 ; f(1,1,0)=0 ; f(1,1,1)=0**

**D’où le tableau de Karnaugh est :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  **ab** **c** | **00** | **01** | **11** | **10** |
|  **0** | **1** | **0** | **0** | **0** |
|  **1** | **1** | **1** | **0** | **1** |

**Table de vérité :**

|  |  |
| --- | --- |
| **Entrées**  | **Sortie** |
| **a b c** |  |
| **0 0 0****0 0 1****0 1 0****0 1 1****1 0 0****1 0 1****1 1 0****1 1 1** | **1****1****0****1****0****1****0****0** |

Cela signifie qu’il faut mettre des « 1 » dans les cases ou les lignes définies par les combinaisons de variables présentent dans l’expression de la fonction «  f ».

**Forme normale conjonctive (FNC) ou 2eme forme canonique de la fonction ou max termes :**

**2eme théorème de Shannon** :

Toute fonction logique peut bse décomposer en un ET de deux OU logique :

**F(a,b,c,…) = [ a+ f(0,b,c,..)].[a̅ + f(1,b,c,…)]**

Cette forme est la duale ou le complément de la FND. L’utilisation successive de ce théorème, nous mène à la forme normale conjonctive.

**Exemple 1 :**

Soit le tableau de Karnaugh ci-dessous :

|  |
| --- |
|  ab 00 01 11 10 C  |
|  0 1 |  0  | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |

1er terme : a+b+c+f (0,0 ,0)

2eme terme a+b+c̅ +f(0,0,1)

3eme terme   a+b̅+c+ f(0,1,0)

4eme terme a+b̅+c̅= f(0,1,1)

5eme terme a̅+b+c+f(1,0,0)

6eme terme a̅+b=c̅+ f(1,0,1)

7eme terme a̅+b̅+c +f(1,1,0)

8eme terme a̅+b̅+c̅ +f(1,1,1)

 F'’a,b,c)=[(a+b=c)+0].[a+b+c̅)+1][(a+b̅+c) +0][(a+b̅+c̅)+1][(a̅+b+c)+1][(a̅+b+c̅)+1]

 [(a̅+b̅+c)+0][(a̅+b̅+c̅)+0]

les termes qui contiennent des "« "« sont éliminés;  donc on aura

f(a,b,c)= (a+b+c)(a+b̅+c)(a̅+b̅+c)(a̅+b̅+c̅)

Pratiquement, on prend les cases qui indiquent « 0 » et puis fait le ET des combinaisons de variables qui leur correspondent. Développement par rapport à « 0 ».

**Exemple 2 :**

Soit le tableau de Karnaugh suivant :

 ab

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  c | 00 | 01 | 11 | 10 |
|  0 | 1 | 0 | 1 | 0 |
|  1 | 0 | 1 | 0 | 0 |

Développement par rapport aux zéros « 0 »

**F(a,b,c)=(a+b+c̅)(a+b̅+c)(a̅+b̅+c̅)(a̅+b+c)(a̅+b+c̅)**

**Exemple 3:**

passage de la FNC au tableau de Karnaugh :

f(a,b,c) = (a+b+c)(a+b̅+c)(a̅+b̅+c)(a̅+b̅+c̅) s'’crit aussi comme suit:

f(a,b,c)= [(a+b+c)+f(0,0,0)][(a+b+c̅)+f(0,0,1)][(a+b̅+c)+f(0,1,0)][(a+b̅+c̅) +f(0,1,1)][(a̅+b+c)+f((1,1,0)][(a̅+b+c+f(1,0,0)][(a̅+b+c̅)+f(1,0,1)][(a̅+b̅+c)+f(1,1,0)][(a̅+b̅+c̅)+f(1,1,1)]

avec : **f(0,0,0) =0 ; f(0,0,1) =1 ; f(0,1 ,0) ; f(0,1,1) =1**

 **f(1,0,0) =1 ; f(1,0,1) =1 ; f(1,1,0) = 0 ; f(1,1,1) =0**

 d’où le tableau de Karnaugh sera comme suit :

 ab

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 |

**Simplification des fonctions :**

 **Définition :** la simplification d’une fonction, consiste à trouver, son expression la plus réduite possible, afin de minimiser le nombre d’opérateurs logiques, nécessaire à utiliser pour sa réalisation.

Deux méthodes de simplification sont possibles :

-Méthode algébrique ou analytique.

-Methode graphique ou tableau de Karnaugh.

**Simplification graphique :**

**Principes de base :**

Cette méthode se base sur l’utilisation du tableau de Karnaugh. Elle permet de montrer les regroupements du type :

1- ab+ab̅ = a(b+b̅) = a

2- abc+abc̅+ab̅c+ab̅c̅ =ab(c+c̅) +ab̅(c+c̅) =ab+ab̅ =a(b+b̅) =a

●on remarque que les regroupements dans les tableaux de Karnaugh ci-après correspondent au cas ou l'’n a 2; 4; 8; (2n) cases adjacentes qui sont simultanément égales à « 1 ».

**Adjacence des cases :**

\*Deux cases sont dites adjacentes, si, elles correspondent à des combinaisons qui diffèrent d’un seul bit. Cela est valable à l’intérieur du tableau, comme à ces bords.

**Régle :**

 La règle consiste à regrouper ces 2n cases pour trouver l’expression booléenne la plus réduite. Dans les cases regroupées, on élimine les variables qui changent d’état et on garde ceux qui ne changent pas d’état. Donc l’expression résultante ne doit contenir que les variables qui n’ont pas changer d’état.

**Exemple 1 :**

 Soit le tableau de Karnaugh suivant. Trouver l’équation réduite déduite du tableau ?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  ab c | 00 | 01 | 11 | 10 |
|  0 | 1 | 0 | 1 | 1 |
|  1 | 1 | 0 | 0 | 0 |

 a̅b̅ b- change d’etat

 a- ne change pas ac̅

 c - ne change pas

f = ac̅ + a̅b̅

**Exemple 2**

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

 a -change d’état exclut

 b -ne change pas d’état maintenue

 c change d’état exclut.

Donc l’expression réduite est : **f = b**

**Exemple 3 :**

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

 a change

 b ne change pas

 c change b̅ d̅ **d’où f = b̅ d̅**

 d ne change pas

**Remarque :1-** l’expression déduite d’un tableau de Karnaugh est la plus réduite c’est-à-dire

 Irréductible.

 **2**-Plus le nombre de cases regroupées est grand, plus on a moins de variables dans l’expression, c’est-à-dire un regroupement de 2n cases conduit à l’élimination de **« n »** variables.

**Simplification algébrique** :

Cette méthode se base sur l’utilisation des propriétés de l’algèbre de Boole et les théorèmes fondamentaux. Elle est moins systématique que la méthode graphique, mais peut parfois donner des résultats rapidement.

**a- utilisation des identités particuliéres :**

**-ab+ab̅ = a a.a = a a+a = a**

**-ab + a = a a.a̅ = 0 a + a̅ = 1**

**-a + a̅b = a + b a. 0 = 0 a + 0 = a**

**-(a + b)(a + b̅) =a a . 1 = a a + 1 = 1**

**-a (a + b) = a**

**-a ( a̅ + b) = ab**

**Exemple 1**

a- f = abc + abc̅ + ab̅cd = ab (c + c̅ ) + ab̅cd = ab + ab̅cd = a (b + b̅cd ) = **ab + acd**

**b-** ajout de terme existant ( utilisation de l'idempotence) du type   **a + a** ou bien **abc + abc** etc..

**Exemple 2 :**

 f=abc + a̅bc + ab̅c + abc̅

 = abc + a̅bc + abc + ab̅c +abc + abc̅

 = bc (a + a̅ ) + ac ( b =b̅ ) + ab (c + c̅ )

 **f = bc + ac + ab**

**Elimination des termes inclus :**

Exemple :

 Soit l’expression :

 F=ab + b̅c + ac

 F=ab + b̅c + ac (b + b̅ )

 =1 élément neutre du produit

Donc la fonction devient :

 f =ab + b̅c +abc + ab̅c = ab ( c+1) + b̅c ( a + 1) = ab.1 + b̅c.1 = **ab + b̅c**

**Combinaisons impossibles ou indéterminées :**

A certain moment des combinaisons particulières des variables ne peuvent pas se produire ou se réaliser, pour des raisons physiques ou technologiques, ces combinaisons sont utilisées pour simplifier les fonctions.

-le principe consiste à attribuer à ces combinaisons la valeur « 0 » ou « 1 »,selon ce qui arrange la simplification (on donne à ces combinaisons la valeur « x » ou « Փ ».

**Exemple :**

On cree la fonction f (a,b,c,d) que f =1 si et seulement si 1< N<5 avec N codé en BCD.

On a 4 bits pour coder les nombres de 0 à 9 ; or avec 4 bits on a jusqu’à 24 = 16 combinaisons.

**Table de vérité :**

|  |  |  |
| --- | --- | --- |
| N | a b c d | f |
| 0 | 0 0 0 0 | 0 |
| 1 | 0 0 0 1 | 0 |
| 2 | 0 0 1 0 | 1 |
| 3 | 0 0 1 1 | 1 |
| 4 | 0 1 0 0 | 1 |
| 5 | 0 1 0 1 | 0 |
| 6 | 0 1 1 0 | 0 |
| 7 | 0 1 1 1 | 0 |
| 8 | 1 0 0 0 | 0 |
| 9 | 1 0 0 1 | 0 |
| 10 | Ne peuvent  Se Produire( termes redondons) | 1ou 0 |
| 11 | Փ |
| 12 | Փ |
| 13 | Փ |
| 14 | Փ |
| 15 | Փ |

**Tableau de Karnaugh :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  **ab** **c** | **00** | **01** | **11** | **10** |
| **00** | **0** | **1** | **Փ** | **0** |
| **01** | **0** | **0** | **Փ** | **0** |
| **11** | **1** | **0** | **Փ** | **Փ** |
| **10** | **1** | **0** | **Փ** | **Փ** |

 **b̅ c bc̅d̅**

**1-si** on ne tient pas compte des etats 10 et 15 la fonctio**n f s'’crit   f = a̅b̅c + a̅bc̅d̅**

**2-** en utilisant les combinaisons impossibles affectées arbitrairement des valeurs **"« "« ou « 1 » (Փ).** on peut ecrire la fonction **f** de la mmanièresuivante**:**

**f=b̅c + bc̅d̅**

**Regle ou Remarque** on affecte les cases impossibles du symbole ( x ou Փ), ce qui veut dire que l'on peut choisir la valeur qui arrange la simplification le **0** ou le **1 pour ces cases.**

**Fonctions ,opérateurs complets ou universels :**

**Fonctions ,operateur NAND :**

a-Expression :

 f(a,b) est vraie si a ET b est faux : f(a,b) = a b

Table de vérité :

|  |  |
| --- | --- |
| Entrées  | sortie |
| a b | f |
| 0 00 11 01 1 | 1110 |

Remarque : c’est la table de vérité du produit opérateur AND complémentée à la sortie.

**Tableau de Karnaugh :**

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | **1** | 1 |
| 1 | 1 | 0 |

**Symbole :**

 

 Norme americaine Norme européenne (IEEE)

**Synthèse de l’inverseur NO (**NON) **avec NAND :**

**F= a b**

 **Si a = b f = a.a = a̅**

 **a a̅**

 **Si b = 1 f = a 1 = a̅**

 **a**

 **1  a̅**

 ** Y = a.b**

**Synthése du AND avec des NAND :**

F=a. b = ( a.b )

 A a b a.b = a.b

 b 

f=( a.b ) . 1 = ( a . b )

 a a b

 b  a.b

 1

**Synthése du OR avec des NAND :**

F= a + b = a̅ . b̅ = ( a . a )( b .b )

 a  a̅

 b  b̅  a +b

**Remarque :**

La synthèse du AND avec NAND on complémente deux fois la fonction.

La synthèse du OR avec NAND on complémente les entrées.

**Fonction ,opérateur ou porte NOR :**

a**- Expression** : f(a,b) est vraie si a OU b est faux : f(a,b) = a + b

b-**Table de vérité** :

|  |  |
| --- | --- |
| Entrées  | Sortie |
| a b | f |
| 0 00 11 01 1 | 1000 |

**c-Tableau de Karnaugh :**

|  |  |  |
| --- | --- | --- |
|  **a** **b** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |

**Symbole :**

 ****

 **Norme américaine Norme européenne (IEEE)**

**Synthèse de l’inverseur NON (NO) avec NOR :**

F= a + b

 Si a = b f = a + a = a̅

 a a̅

Si b = 1 f = a + 0 = a̅

 a

 0 a̅

**Synthése du AND avec des NOR :**

F=a b = ( a b ) = a̅ + b̅ = ( a + a ) + ( b + b ) =( a + 0 ) + (b + 0 )

**Remarque :**

- La synthèse du AND avec NOR, on complémente les entrées.

-La synthèse du OR avec NOR , on doit complémenter deux fois.

**Utilisation des NOR et des NAND pour la synthèse d’une fonction quelconque :**

Comme ce sont des opérateurs universels, ils peuvent donc, être utilisés indifféremment pour synthétiser une même fonction.

**Nota benné** : en pratique, le NAND est plus rapide que le NOR ce qui implique, qu’il est le plus utilisé.

**Regle de De Morgan graphique :**

 - On peut exprimer le théoreme de De Morgan directement sur l’expression graphique de la fonction.

**a**) - a + b = a̅ . b̅  a + b

 **b )-** a . b = a̅ + b̅  

 a. b

**Fonction, operateur ou porte XOR ( OU exclusif) :**

**XOR à deux entrées :**

**1-Expression** **:** f(a,b) est vraie si a est vrai ou b est vrai,mais pas les deux ;

 f(a,b) = a ⊕ b= a̅ b + ab̅

**Table de vérité**:

|  |  |
| --- | --- |
| Entrées  | Sortie |
| a b | f |
| 0 00 11 01 1 | 0110 |

D’après la table de vérité ,on a f = a **⊕ b =a̅b + ab̅**

**Tableau de Karnaugh :**

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

**Symbole :**

 

 Normes américaine Norme européenne ( IEEE)

**Remarque :** L’opérateur XOR est un détecteur de parité impaire.

**Opérateur XOR à plusieurs entrées :**

XOR(a,b,c,..) = 1 ; si le nombre de variables à 1 est impaire.

**Exemple à trois variables : f = a⊕ b ⊕ c**

**Table de vérité :**

|  |  |
| --- | --- |
| Entrées  | Sortie  |
| a b c |  |
| 0 0 00 0 10 1 00 1 11 0 01 0 11 1 01 1 1 | 01101001 |

**Tableau de Karnaugh :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  **ab**  **c** | **00** | **01** | **11** | **10** |
| **0** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **0** |

**Remarque :** la représentation d’un XOR en tableau de Karnaugh est un damier.

**Associativité :**

Le Xor posséde la propriété d’associativité :

**a⊕ b ⊕ c = (a⊕ b) ⊕ c = a⊕ ( b ⊕ c )**

**a⊕ b ⊕ c = a̅b̅c + a̅bc̅ + abc + ab̅c̅ = a̅ ( b̅c + bc̅ ) + a (bc + b̅c̅ )= a̅ ( b ⊕ c ) + a ( b ⊕ c )**

 **= a ⊕ ( b ⊕ c ).**

La symétrie permet d’effectuer toutes les permutation entre a, b ou c

  f = **a⊕ b ⊕ c**

 C

**Utilisation du XOR :**

 Pour la detection de l’inegalité de deux variables .

 **a⊕ b = 0 si a = b ; a⊕ b = 1 si a ≠ b**

**\*** Pour la détection de l’imparité. :

 **a⊕ b ⊕ c = 0 si le nombre de variables à 1 est pair**

 **a⊕ b ⊕ c = 1 si le nombre de variables à 1 est impair.**

**Opérateur programmable :**

 Le XOR est l’opérateur programmable le plus simple .avec les opérateurs AND , OR NO ,NOR et NAND , la relation entre sortie **f** et entrée ( a,b,c,…) est fixe ( figée ) .

Avec un XOR on peut jouer sur cette relation de facon à la modifier en fonction d’une entrée qui joue le role de commande.

**Principe :**

 Soit la table de vérité suivante :

|  |  |
| --- | --- |
| Entrées  | Sortie  |
| a b | f |
| 0 00 11 01 1 | 0110 |

 a

p 

f= 0 si a = 0 p =0 f = a si p = 0

f = 1 si a = 1 et p =0

f = 1 si a = 0 et p = 1 f = a̅ si p = 1

f = 0 si a = 1 et p = 1

**p :** est donc l’entrée de contrôle**, a** : peut etre la sortie d’autre opérateurs.

Exemple :

  **f**

 P

F =a.b si p = 0 ; f = a̅ b̅ si p = 1