Note de COURS
1 : Introduction aux circuits logiques

Algèbre Booléenne

- Base de la conception et de l'analyse des systèmes numériques. Il traite du cas où les variables prennent une seule des deux valeurs : **TRUE** (généralement représentée par le symbole «1») et **FALSE** (généralement représentée par le symbole «0»). Elle est aussi appelée algèbre booléenne à deux valeurs ou algèbre de commutation.
- Un circuit composé de commutateurs peut être représenté en termes d'équations algébriques booléennes. Les équations peuvent ensuite être manipulées sous la forme représentant le circuit le plus simple. Le circuit peut alors être immédiatement tiré des équations. Cette méthode est apparue pour la première fois dans : "A symbolic Analysis of Relay and Switching Circuits", Claude E. Shannon, Transactions of the AIEE, vol. 57, no. 12, Dec. 1938, pp. 713-721.

Opérations de base

- X et Y sont des variables booléennes. Les variables booléennes sont utilisées pour représenter les entrées ou les sorties d'un circuit numérique.

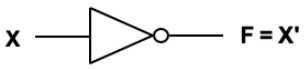
Opérations		Expression Booléenne
NOT	Négation logique	$X'(or \bar{X})$
AND	Conjonction logique	*X.Y*
OR	Disjonction logique	*X+Y*

Table de vérité et portes logiques

- **Table de vérité** : Table donnant les valeurs d'une fonction pour toutes les combinaisons possibles de ses arguments en entrée. S'il y a n entrée, il y a 2^n combinaisons possibles.
- **Portes logiques** : C'est un composant matériel qui produit une valeur logique (0 ou 1) selon l'état des entrées. Les fonctions booléennes peuvent être implémentées avec des portes logiques.

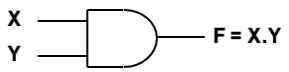
Porte NON

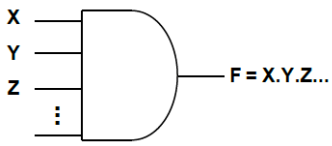
X	F=X'
0	1
1	0



Porte ET

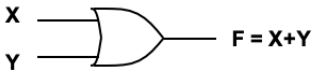
X	Y	F=X.Y
0	0	0
0	1	0
1	0	0
1	1	1

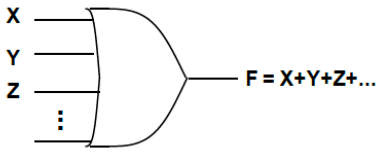




Porte OU

X	Y	F=X+Y
0	0	0
0	1	1
1	0	1
1	1	1





Axiomes :

0.0=0	*1.1=1*	*0.1=1.0=0*	$\bar{0} = 1$
1+1=1	*0+0=0*	*1+0=0+1=1*	$\bar{1} = 0$

Théorèmes :

Élément neutre	*X.1=X* *X+0=X*
Commutativité	*X.Y=Y.X* *X+Y=Y+X*
Complémentaire	$X.\bar{X}=0$ $X+\bar{X}=1$

Idempotence	*X.X=X* *X+X=X*
Élément absorbant	*X.0=0* *X+1=1*
Double négation	$\bar{\bar{X}}=X$
Associativité	*X.(Y.Z)=(X.Y).Z* *X+(Y+Z)=(X+Y)+Z*
Distributivité	*X.(Y+Z)=X.Y+X.Z* *X+(Y.Z)=(X+Y).(X+Z)*

Autres théorèmes

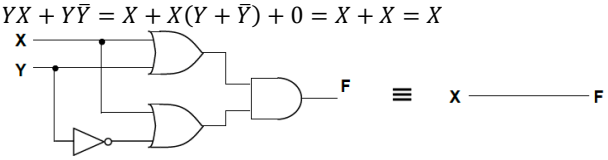
Absorption	*X.(X+Y)=X.X+X.Y=X+X.Y=X.(1+Y)=X* *X+X.Y=X.(1+Y)=X*
Adjacence	*X.Y+X.$\bar{Y}$=X* *(X+Y).(X+$\bar{Y}$)=X*
Consensus	*X.Y+$\bar{X}$.Z+Y.Z=X.Y+$\bar{X}$.Z* *(X+Y).($\bar{X}$+Z).(Y+Z)=(X+Y).($\bar{X}$+Z)* *Corollaire : (X+Y).($\bar{X}$+Z)=$\bar{X}$Y+XZ*
DeMorgan	$\overline{X.Y} = \bar{X} + \bar{Y}$ $\overline{(X+Y)} = \bar{X}.\bar{Y}$
Simplification	*X.($\bar{X}$+Y)=X.Y* *X+$\bar{X}$Y=X+Y*

Une application des théorèmes réside dans la simplification des fonctions booléennes et dans la réduction du nombre de portes logiques.

Exemple :
$F = (A + \bar{B}C + D + EF)(A + \bar{B}C + \overline{D + EF})$
$F = (X + Y)(X + \bar{Y})$ avec $X = A + \bar{B}C$ et $Y = D + EF$
$F = X = A + \bar{B}C$

Exemple :
$F = \overline{(\overline{X + Y})Z + X\bar{Y}Z}$
$F = \bar{X}\bar{Y}Z + X\bar{Y}Z$
$F = \bar{Y}Z(X + \bar{X}) = \bar{Y}Z = Y + \bar{Z}$

Exemple :
$F = (X + Y)(X + \bar{Y}) = XX + X\bar{Y} + YX + Y\bar{Y} = X + X(Y + \bar{Y}) + 0 = X + X = X$

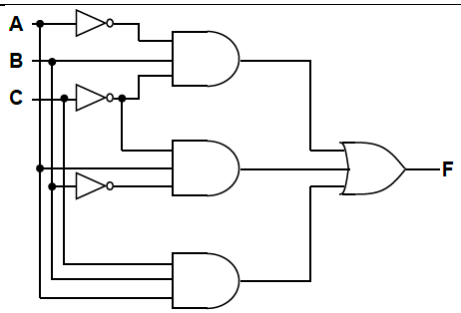


Constructions de fonctions booléennes à partir des tables de vérités

En utilisant les 1 :

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

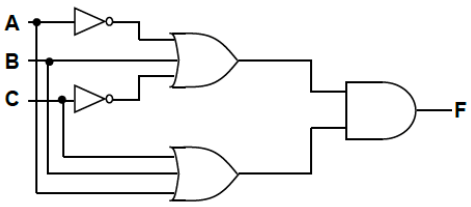
$F = \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$



En utilisant les 0 :

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

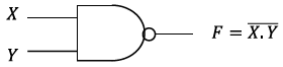
$F = (A + B + C)(\bar{A} + B + \bar{C})$



Autres portes logiques :

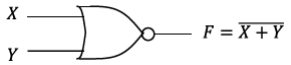
NAND

X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0



NOR

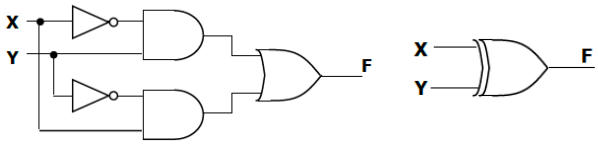
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0



XOR

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

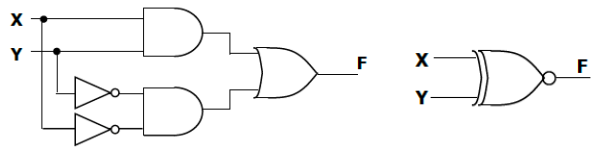
$F = \bar{X}Y + X\bar{Y} = X \oplus Y$



XNOR

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

$F = XY + \bar{X}\bar{Y} = \overline{X \oplus Y}$



Somme de Produit (SOP) et Produit de Somme (POS) en utilisant Minterm et Maxterm

- Maxterm : dans un terme somme où chaque variable est présente une fois exactement.
- Minterm : dans un terme produit où chaque variable est présente une fois exactement.
- Exemple à 3 variables :

	x_1	x_2	x_3	Minterms	Maxterms
0	0	0	0	$m_0 = \overline{x_1}\,\overline{x_2}\,\overline{x_3}$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x_1}\,\overline{x_2}x_3$	$M_1 = x_1 + x_2 + \overline{x_3}$
2	0	1	0	$m_2 = \overline{x_1}\,x_2\overline{x_3}$	$M_2 = x_1 + \overline{x_2} + x_3$
3	0	1	1	$m_3 = \overline{x_1}\,x_2x_3$	$M_3 = x_1 + \overline{x_2} + \overline{x_3}$
4	1	0	0	$m_4 = x_1\overline{x_2}\,\overline{x_3}$	$M_4 = \overline{x_1} + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x_2}x_3$	$M_5 = \overline{x_1} + x_2 + \overline{x_3}$
6	1	1	0	$m_6 = x_1x_2\overline{x_3}$	$M_6 = \overline{x_1} + \overline{x_2} + x_3$
7	1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \overline{x_1} + \overline{x_2} + \overline{x_3}$

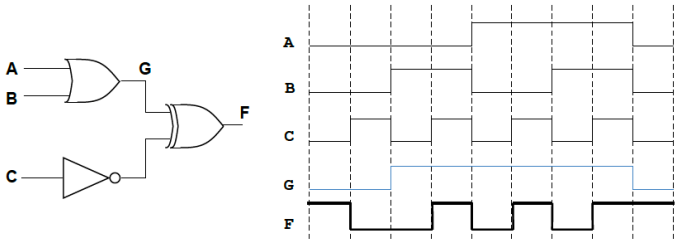
- Une fonction à *n* variables peut avoir jusqu'à 2^n minterms (notés m_0 à m_{2^n-1}) ou 2^n maxterms (notés M_0 à M_{2^n-1})
- $M_i = \overline{m_i}$
- Le nombres de fonctions différentes à *n* variables est de 2^{2^n}
- Une fonction peut s'exprimer comme la somme de minterm ou le produit de maxterms :
  - Lorsqu'un minterm est évalué à 1, cela implique que la fonction est évaluée à 1
    Exemple : $f(x,y) = xy + \bar{x}\bar{y}$. Dans ce cas, $f(x,y) = 1$ lorsque $xy = 00{,}11$. Donc, les minterms sont m_0 et m_3.
    $f(x,y) = m_0 + m_3$
  - Lorsqu'un maxterm est évalué à 0, cela implique que la fonction est évaluée à 0
    Exemple : $f(x,y) = (x + \bar{y})(\bar{x} + y)$. Dans ce cas, $f(x,y) = 0$ lorsque $xy = 10{,}01$. Donc, les maxterms sont M_1 et M_2.
    $f(x,y) = M_1.M_2$
- Une somme de produit (SOP) composée uniquement de minterms ou un produit de somme (POS) composé uniquement de maxterms est appelé forme canonique.
- Si un SOP (respectivement POS) contient des éléments qui ne sont pas des minterms (resp. maxterms), il n'est pas sous forme canonique.
  - Exemples : $F(x,y,z) = x.y.z + \bar{x}\bar{y}$; $F(x,y,z) = (x + y + z).(\bar{x} + \bar{y})$; $F(x,y,z) = xyz + x\bar{y}z + \bar{x}y\bar{z} + (\bar{x} + y + z)$

- Exemple :

X	Y	Z	F	Sum of Products
0	0	0	0	$F = \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + X\bar{Y}Z + XY\bar{Z}$
0	0	1	1	$F(X,Y,Z) = \sum(m_1, m_4, m_5, m_6).$
0	1	0	0	$F(X,Y,Z) = \sum m(1,4,5,6)$ Also: $\bar{F}(X,Y,Z) = \sum m(0,2,3,7)$
0	1	1	0	
1	0	0	1	Product of Sums
1	0	1	1	$F = (X+Y+Z)(X+\bar{Y}+Z)(X+\bar{Y}+\bar{Z})(\bar{X}+\bar{Y}+\bar{Z})$
1	1	0	1	$F(X,Y,Z) = \prod(M_0, M_2, M_3, M_7).$
1	1	1	0	$F(X,Y,Z) = \prod M(0,2,3,7)$ Also: $\bar{F}(X,Y,Z) = \prod M(1,4,5,6)$

- Remarqué que $F(X,Y,Z) = \sum m_{1,4,5,6} = \prod M_{0,2,3,7}$

Chronogramme



Conception numérique

- Lorsque l'on conçoit des circuits numériques, il faut définir les spécifications des circuits désirés.
- On utilise des variables booléennes pour représenter les états des entrées (ex : switchs, boutons) et des sorties (ex : LED, verrous)
- **Méthode basée sur la table de vérités :** méthode très simple
  - La relation entre les entrées et les sorties est définit à partir de la table de vérité
  - Les fonctions logiques sont ensuite générées et simplifiées. On peut esquisser le circuit résultant en utilisant les portes logiques.
  - Cette méthode, certes très simple, est très efficace pour des circuits avec un petit nombre d'entrée. Pour un grand nombre d'entrée, elle devient inutile car le nombre de possibilité croit exponentiellement avec le nombre d'entrée. (Ex : un circuit à 16 entrées nécessite une table de vérité de 65 536 lignes !)
  - **Exemple :** Fonction majorité (circuit qui renvoie la valeur la plus présente entre 3 valeurs)

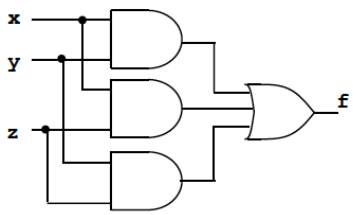
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$f = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$

$f = \bar{x}yz + x(\bar{y}z + y\bar{z} + yz) = \bar{x}yz + x(\bar{y}z + y)$
$f = \bar{x}yz + x(\bar{y} + y)(z + y)$

$f = \bar{x}yz + xz + xy = xz + y(x + \bar{x}z)$

$f = xz + y(x + \bar{x})(x + z) = xz + yx + yz$



Flot de conception de circuit FPGA avec les outils XILINX :

- **Spécification du circuit** : Le circuit est spécifié via un langage de description matérielle (HDL), un schéma ou une forme d'onde. Le processus de vérification de la syntaxe HDL des connexions schématiques est appelé Synthèse.
- **Simulation comportementale :** Il s'agit d'une étape cruciale. Votre circuit peut être « sans erreur » au niveau de la syntaxe, mais peut ne pas fonctionner comme prévu. Dans cette partie, des stimuli sont donnés en entrée du circuit et les sorties vérifiées par rapport aux sorties attendues. Lorsque les stimuli sont écrits en HDL, cela s'appelle un « banc d'essai ». Ce processus est très similaire à l'utilisation d'un générateur de signaux pour créer les entrées et d'un oscilloscope pour visualiser les sorties au fil du temps.
- **Positionnement physique** : Les entrées et les sorties du circuit sont mises en correspondance avec les composants du FPGA cible et de sa carte hôte (PCB). L'outils Xilinx Vivado a besoin d'un fichier appelé fichier de contraintes (.xdc)
- **Simulation temporelle** : La simulation comportementale simule uniquement le circuit de manière « logique », c'est-à-dire qu'elle ne prend pas en compte l'analogique et les effets électriques. La simulation temporelle prend en compte le retard qui existe entre les entrées et les sorties, et permet d'identifier les éventuels problèmes.
- **Implémentation** : Ensuite le FPGA est programmé à l'aide d'un fichier de configuration appelé « bitstream ».