POLYTECH®
DIJON

2024/2025

Architecture interne des ordinateurs

Pr. Dominique Ginhac
dginhac@u-bourgogne.fr

Dernière mise à jour le 21 Nov. 2024 par dG

Chapitre #01

Algèbre booléenne

https://ginhac.com/archi/01-logique.pdf

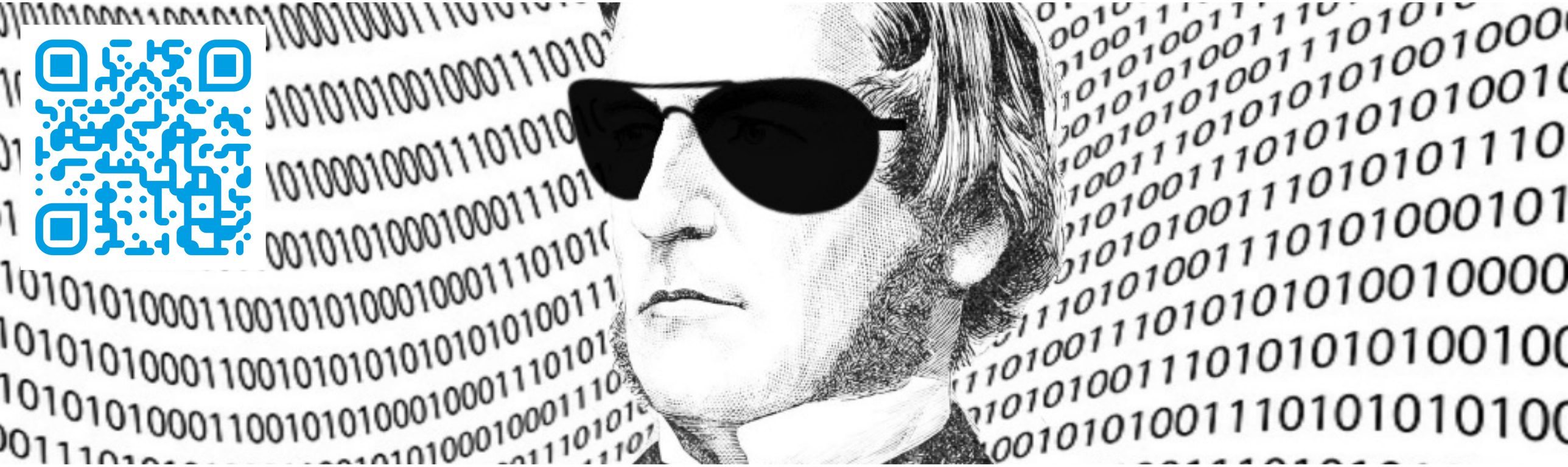


Photo par NeONBRAND sur Unsplash

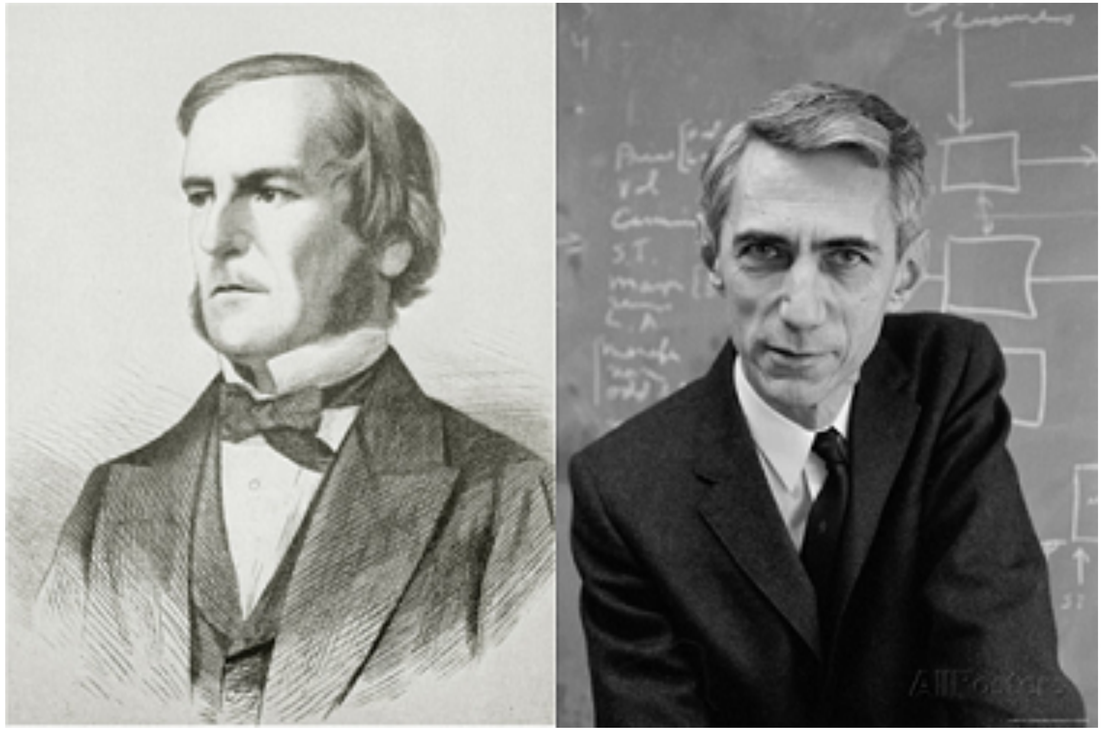
Introduire la logique booléenne, base de tout système numérique, reliant matériel et logiciel grâce à des opérations simples.

Enjoy!

Un peu d’**histoire**

En 1847, G. Boole (1815-1864) invente une algèbre pour traiter les variables binaires.

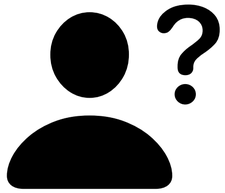
En 1938, C. Shannon (1916-2001) est le premier à appliquer l’algèbre de Boole à la conception de circuits de commutation téléphonique.



Une **définition**

L’algèbre ou logique booléenne est une branche des mathématiques qui étudie les relations entre des variables prenant uniquement 2 valeurs : **Vrai** (1, présent) et **Faux** (0, absent).

L'algèbre booléenne utilise 3 opérations de base : **et** (and), **ou** (or), **non** (not) et un ensemble de propriétés/théorèmes.



Quels liens entre l'**algèbre booléenne** et l'**architecture des ordinateurs** ?

1

Base de tous les **circuits numériques**

2

Socle pour la **conception des processeurs**

3

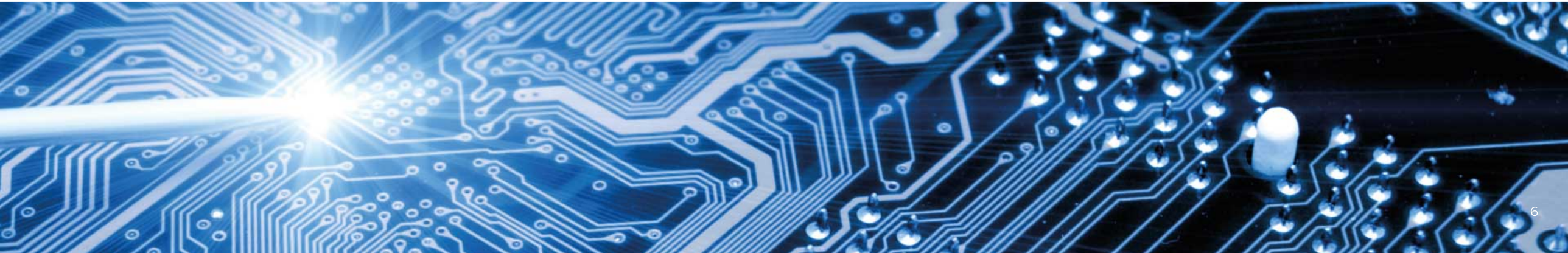
Pilier fondamental pour la **programmation d'algorithmes**

Pourquoi les **ordinateurs** sont-ils **“binaires”** ?

10110000000111100011010101011111101111000011111000001010111010110101

Un processeur est un circuit numérique constitué d’un ensemble de transistors qui agissent comme des interrupteurs (fermé/ouvert) commandés par des signaux électriques, soit au niveau bas (tension faible, 0V, faux), soit au niveau haut (tension élevée, 1 à 5V, vrai).

Les transistors sont regroupés en blocs pour former des opérations logiques plus ou moins complexes.



Les **opérateurs** de base

Porte OU (OR)

A	B	Y = A+B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

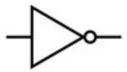
*Ex : "Une alarme sonne si un détecteur de fumée **OU** un détecteur de chaleur est activé."*

Porte ET (AND)

A	B	Y = A.B
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

*Ex : "Un feu vert s'allume si **ET** seulement si deux capteurs détectent un véhicule."*

Porte INV (NOT)



A	Y = $\overline{A}$
0	**1**
1	**0**

*Ex : "Un voyant s'allume lorsque la température **n'est pas** normale."*

Opérateurs logiques omniprésents dans de nombreux secteurs :

- Programmation conditionnelle : (Role = Admin OR Role = Manager) AND AccountActive.
- Recherche et filtrage : (Catégorie = "Ordinateur" AND Prix ≤ 1000) OR Promo=True
- Systèmes automatisés : CapteurPresence AND NOT (PorteOuverte)
- Gestion d'accès : (BadgeValide AND CodeCorrect) OR ApprobationAdmin
- Jeux vidéos : (Score > 1000 OR BonusActive) AND TempsRestant>0.
- ...

Les **portes** logiques **composées**

Porte NON OU (NOR) 

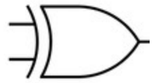
A	B	$Y = \overline{A+B}$
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Sortie vraie seulement si toutes les entrées sont fausses.

Porte NON ET (NAND) 

A	B	$Y = \overline{A.B}$
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Sortie fausse seulement si toutes les entrées sont vraies.

Porte Ou exclusif (XOR) 

A	B	$Y = A \oplus B$
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Sortie vraie seulement si une seule des entrées est vraie.

Portes logiques à 2 entrées extensibles à n entrées.

Portes logiques combinables pour créer des opérations complexes (arithmétique par exemple).

Portes logiques = briques élémentaires de l'électronique/informatique numérique

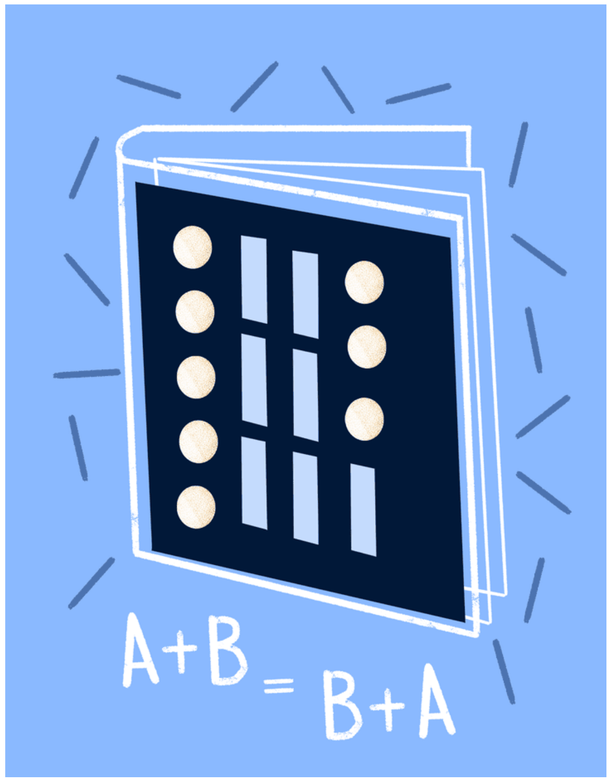
Les **Propriétés** de l'Algèbre de Boole

	OR	AND
Associativité	$(A + B) + C = A + (B + C) = A + B + C$	$(A \,.\, B) \,.\, C = A \,.\, (B \,.\, C) = A \,.\, B \,.\, C$
Commutativité	$A + B = B + A$	$A \,.\, B = B \,.\, A$
Priorité	$A + B \,.\, C = A + (B \,.\, C)$	
Distributivité	$A + (B \,.\, C) = (A + B) \,.\, (A + C)$ *Pas vrai en arithmétique*	$A \,.\, (B + C) = A \,.\, B + A \,.\, C$
Elément neutre	$A + 0 = A$	$A.1 = A$
Elément absorbant	$A + 1 = 1$	$A.0 = 0$
Complémentarité	$\overline{A} + A = 1$	$\overline{A} \,.\, A = 0$
Idempotence	$A + A = A$	$A \,.\, A = A$

Les **Théorèmes** de l'Algèbre de Boole

Théorème d'involution	$\overline{\overline{A}} = A$	
Théorème d'inclusion	$(A + B) \,.\, (A + \overline{B}) = A$	$A \,.\, B + A \,.\, \overline{B} = A$
Théorème d'allégement	$A + \overline{A} \,.\, B = A + B$	$A \,.\, (\overline{A} + B) = A \,.\, B$
Théorème d'absorption	$A + (A \,.\, B) = A$	$A \,.\, (A + B) = A$
Théorème de De Morgan	$\overline{A + B} = \overline{A} \,.\, \overline{B}$	$\overline{A \,.\, B} = \overline{A} + \overline{B}$

Fonctions **booléennes**



Une fonction booléenne est une application de $\{0,1\}^n$ vers $\{0,1\}$

$$(x_1, x_2, x_3, \ldots, x_n) \rightarrow f(x_1, x_2, x_3, \ldots, x_n) \in \{0,1\}$$

Elle peut être définie de 2 manières différentes :

1. Par une table de vérité
2. Par une équation logique

Fonctions **booléennes**

Soit $f(x, y, z) \in \{0,1\}$ $avec$ $x, y, z \in \{0,1\}$.

La table de vérité donne la valeur de f pour les 8 différentes combinaisons possibles : $f(0,0,1) = 1;\ f(0,1,1) =\ f(1,0,1) = 0,\ \ldots$

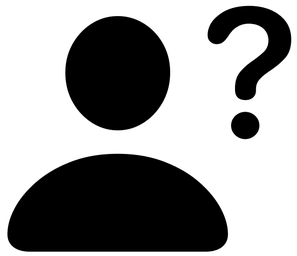
A partir de la table de vérité, on peut retrouver la forme canonique de la fonction sous la forme d'une somme de monômes pour lesquels $f(x, y, z) = 1$:

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

x	y	z	f
0	**0**	**0**	**1**
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	**1**	**1**	**1**

Comment simplifier la **FORME CANONIQUE** ?

$$f(x, y, z) = \bar{x} \,.\, \bar{y} \,.\, \bar{z} + \bar{x} \,.\, \bar{y} \,.\, z + \bar{x} \,.\, y \,.\, \bar{z} + x \,.\, y \,.\, z = ?$$



2 Méthodes

$$f(x, y, z) = \bar{x} \,.\, \bar{y} \,.\, \bar{z} + \bar{x} \,.\, \bar{y} \,.\, z + \bar{x} \,.\, y \,.\, \bar{z} + x \,.\, y \,.\, z = ?$$

① En utilisant les **propriétés/théorèmes** (factorisation, lois de De Morgan, ...)

② En utilisant la **table de vérité**

Réduction de la taille de la table de vérité

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Problème de taille de la table si le nombre de variables est grand (n>4) car le nombre de lignes de la table est égal à 2^n.

Choix d'une nouvelle représentation de la table sous la forme d'un tableau de 2^n cases avec une répartition des variables sur les 2 dimensions (ici xy en abscisse et z en ordonnée)

z \ xy	00	01	10	11
0	1	1	0	0
1	1	0	0	1

Réduction de la taille de la table de vérité

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Equivalence des 2 tables

xy

z \ f	00	01	10	11
0	1	1	0	0
1	1	0	0	1

x

yz \ f	0	1
00	0	0
01	1	0
10	1	0
11	0	1

Réduction de la taille de la table de vérité

Utilisation du code Gray

En code Gray (ou code binaire réfléchi), on inverse un seul bit à la fois pour passer d'une ligne ou colonne à la suivante. Le choix se fait sur le bit le plus à droite possible conduisant à un nouveau nombre.

xy

z \ f	00	01	10	11
0	1	1	0	0
1	1	0	0	1

xy

z \ f	00	01	11	10
0	1	1	0	0
1	1	0	1	0

Tableau de Karnaugh = table de vérité particulière destinée à faire apparaître visuellement les simplifications possibles.

Comment déterminer **L'ÉQUATION LOGIQUE** à partir de la **TABLE DE KARNAUGH ?**

1. Grouper les cases voisines égales à 1 par puissance de 2 (1, 2, 4, 8, ...).
2. Pour chaque groupe, éliminer les entrées qui changent de valeur.
3. Reconstituer l'équation logique sous la forme d'une somme des groupes simplifiés.

Détermination de l'équation logique

1. Grouper les cases voisines égales à 1 par puissance de 2 (1, 2, 4, 8, ...).
2. Pour chaque groupe, éliminer les entrées qui changent de valeur.
3. Reconstituer l'équation logique sous la forme d'une somme des groupes simplifiés.

xy

f	00	01	11	10
0	1	1	0	0
1	1	0	1	0

z

3 groupes

- x = 0, y = 0, z variable
- x = 0, y variable, z = 0
- x = 1, y = 1, z = 1

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

Un deuxième exemple

$$f(w,x,y,z) \in \{0,1\} \; avec \; w,x,y,z \in \{0,1\}$$

w	x	y	z	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	**0**	**1**	**1**	**1**
0	1	0	0	0
0	**1**	**0**	**1**	**1**
0	1	1	0	0
0	**1**	**1**	**1**	**1**

w	x	y	z	f
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	**0**	**1**	**1**	**1**
1	1	0	0	0
1	**1**	**0**	**1**	**1**
1	1	1	0	0
1	**1**	**1**	**1**	**1**

Un deuxième exemple

$$f(w,x,y,z) \in \{0,1\} \; avec \; w,x,y,z \in \{0,1\}$$

wx

yz	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	1	1
10	0	0	0	0

Toujours sélectionner les groupes
les plus grands possibles

w variable, x = 1, y variable, z = 1

w variable, x variable, y = 1, z = 1

$$f(w,x,y,z) = x \,.\, z + y \,.\, z = z \,.\, (x + y)$$

w inutilisé (voir table de vérité)

Un troisième exemple

$$f(w,x,y,z) \in \{0,1\} \; avec \; w,x,y,z \in \{0,1\}$$

wx

yz	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	1	0	0	1
10	0	1	1	0

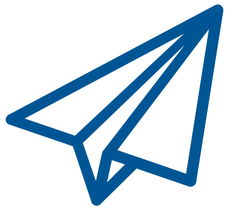
Utiliser la symétrie du code Gray pour sélectionner les groupes les plus grands

▢ + ▢ w variable, x = 1, y variable, z = 0

▢ + ▢ w variable, x = 0, y = 1, z = 1

$$f(w,x,y,z) = x \,.\, \bar{z} + \bar{x} \,.\, y \,.\, z$$

TAKE HOME MESSAGE



#1

Le Langage Universel des Ordinateurs

La logique booléenne est la base de tout système numérique, reliant matériel et logiciel grâce à des opérations binaires simples.

Des Concepts Simples pour des Systèmes Complexes

Avec seulement deux états (Vrai/Faux) et quelques opérateurs (AND, OR, NOT), on peut modéliser et optimiser des systèmes informatiques et électroniques.

Des Outils pour Simplifier et Optimiser

Les tables de vérité et les tableaux de Karnaugh permettent de visualiser, analyser et simplifier les expressions logiques, essentielles pour concevoir des circuits efficaces.

Questions

Photo by Alexander Andrews on Unsplash

Contacts

Pr. Dominique Ginhac

dginhac@u-bourgogne.fr

Retrouvez toutes les infos sur :

https://github.com/dginhac/esirem-archi

POLYTECH DIJON