Logique combinatoire

SIN1 - Cours 2

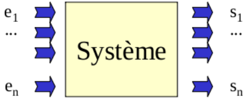
J. Villemejane - julien.villemejane@u-pec.fr

IUT Créteil-Vitry
Département GEII
Université Paris-Est Créteil

Année universitaire 2013-2014

Plan du cours

Notions de logique combinatoire

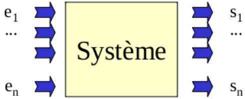


Les systèmes logiques traitent des **données binaires**, en entrée et en sortie.

La **caractéristique d'entrées-sorties** d'un système numérique est appelée **fonctions logiques**.

Lorsque la relation entre l'entrée et la sortie est directe (l'état du système n'intervenant pas), on parle alors d'un **système combinatoire**.

Notions de logique combinatoire

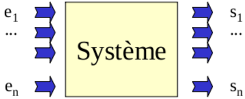


Les systèmes logiques traitent des **données binaires**, en entrée et en sortie.

La **caractéristique d'entrées-sorties** d'un système numérique est appelée **fonctions logiques**.

Lorsque la relation entre l'entrée et la sortie est directe (l'état du système n'intervenant pas), on parle alors d'un **système combinatoire**.

Notions de logique combinatoire

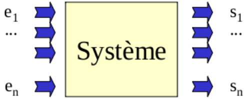


Les systèmes logiques traitent des **données binaires**, en entrée et en sortie.

La **caractéristique d'entrées-sorties** d'un système numérique est appelée **fonctions logiques**.

Lorsque la relation entre l'entrée et la sortie est directe (l'état du système n'intervenant pas), on parle alors d'un **système combinatoire**.

Notions de logique combinatoire



Les systèmes logiques traitent des **données binaires**, en entrée et en sortie.

La **caractéristique d'entrées-sorties** d'un système numérique est appelée **fonctions logiques**.

Lorsque la relation entre l'entrée et la sortie est directe (l'état du système n'intervenant pas), on parle alors d'un **système combinatoire**.

Notions de logique combinatoire

Les informations traitées par ces systèmes sont appelées des **variables logiques** ou **booléennes**.

Formalisation

Dans le monde industriel, on représente les assertions par des variables à 2 états : "*vrai*" ou "*faux*".
VRAI (ou TRUE, en anglais) sera associé à la valeur numérique "1"
FAUX (ou FALSE, en anglais) sera associé à la valeur numérique "0"

Ces niveaux logiques sont en pratique associés à des tensions.

Notions de logique combinatoire

Les informations traitées par ces systèmes sont appelées des **variables logiques** ou **booléennes**.

Formalisation

Dans le monde industriel, on représente les assertions par des variables à 2 états : "*vrai*" ou "*faux*".
VRAI (ou TRUE, en anglais) sera associé à la valeur numérique "1"
FAUX (ou FALSE, en anglais) sera associé à la valeur numérique "0"

Ces niveaux logiques sont en pratique associés à des tensions.

Représentation des systèmes combinatoires

Il existe plusieurs façons de représenter un système combinatoire :

- par son **équation logique** ;
- par sa **table de vérité** ;
- par son **logigramme**.

Equation logique

Une **équation logique** est une expression contenant des noms de **variables** espacées par des **opérateurs logiques** (ET, OU...)

Equation logique

$$S = a \cdot \bar{b} + \overline{a \cdot c}$$

Représentation des systèmes combinatoires

Il existe plusieurs façons de représenter un système combinatoire :

- par son **équation logique** ;
- par sa **table de vérité** ;
- par son **logigramme**.

Equation logique

Une **équation logique** est une expression contenant des noms de **variables** espacées par des **opérateurs logiques** (ET, OU...)

Equation logique

$$S = a \cdot \overline{b} + \overline{a \cdot c}$$

Représentation des systèmes combinatoires

Table de vérité

La **table de vérité** donne la **liste des valeurs de sortie** pour **toutes les combinaisons possibles de l'entrée**, classées selon l'ordre du code binaire naturel.

Table de vérité

Variables d'entrée

Variable de sortie

a	b	S
0	0	1
0	1	1
1	0	1
1	1	0

Evolution des variables d'entrée

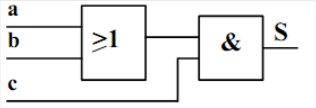
Evolution de la variable de sortie

Représentation des systèmes combinatoires

Logigramme

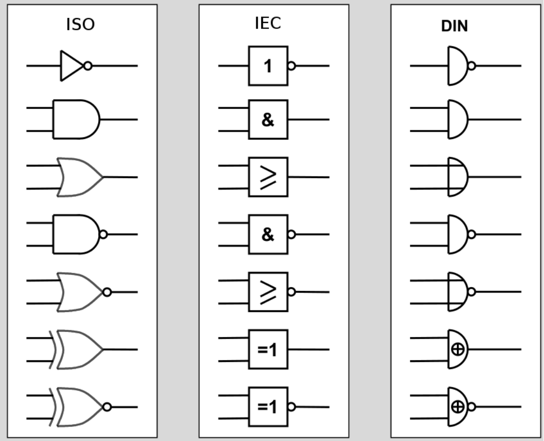
Le logigramme est un formalisme issu du monde électronique. Les expressions logiques sont traduites en un **câblage** reliant des symboles qui représentent les opérateurs.

Logigramme



Représentation des systèmes combinatoires

Il existe 3 normes en vigueur pour la dernière représentation



Algèbre de Boole

George BOOLE a défini, vers 1847, une algèbre qui s'applique à des fonctions logiques de variables logiques (appelées variables booléennes).

Toute fonction logique peut être réalisée à l'aide d'un petit nombre de fonctions logiques de base appelées **opérateurs logiques** ou **portes**.

Il existe deux types d'opérateurs :

- **Opérateurs fondamentaux** : NON, ET, OU
- **Opérateurs dérivés** : NON-ET, NON-OU, OU-EXC, NON-OU-EXC

Algèbre de Boole

George BOOLE a défini, vers 1847, une algèbre qui s'applique à des fonctions logiques de variables logiques (appelées variables booléennes).

Toute fonction logique peut être réalisée à l'aide d'un petit nombre de fonctions logiques de base appelées **opérateurs logiques** ou **portes**.

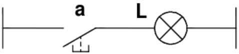
Il existe deux types d'opérateurs :

- **Opérateurs fondamentaux** : NON, ET, OU
- **Opérateurs dérivés** : NON-ET, NON-OU, OU-EXC, NON-OU-EXC

Algèbre de Boole
Opérateurs fondamentaux

Opérateur OUI

Schéma électrique :



Equation :

L = a

Table de vérité :

a	L
0	0
1	1

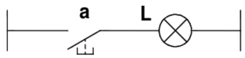
Symbole :

Opérateur NON

Algèbre de Boole
Opérateurs fondamentaux

Opérateur OUI

Schéma électrique :



Equation :

L = a

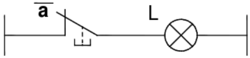
Table de vérité :

a	L
0	0
1	1

Symbole :

Opérateur NON

Schéma électrique :



Equation :

$L = \overline{a}$

Table de vérité :

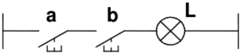
a	L
0	1
1	0

Symbole :

Algèbre de Boole
Opérateurs fondamentaux

Opérateur ET

Schéma électrique :



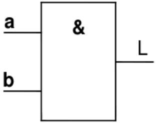
Equation :

L = a . b

Table de vérité :

a	b	L
0	0	0
0	1	0
1	0	0
1	1	1

Symbole :



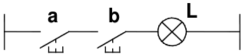
Opérateur OU

Algèbre de Boole

Opérateurs fondamentaux

Opérateur ET

Schéma électrique :



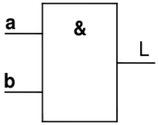
Equation :

L = a . b

Table de vérité :

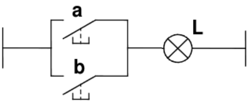
a	b	L
0	0	0
0	1	0
1	0	0
1	1	1

Symbole :



Opérateur OU

Schéma électrique :



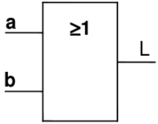
Equation :

L = a + b

Table de vérité :

a	b	L
0	0	0
0	1	1
1	0	1
1	1	1

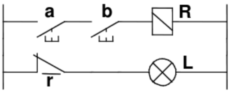
Symbole :



Algèbre de Boole
Opérateurs dérivés

Opérateur NON-ET

Schéma électrique :



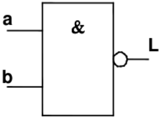
Equation :

$$L = \overline{a \,.\, b} = \overline{a} + \overline{b}$$

Table de vérité :

a	b	L
0	0	1
0	1	1
1	0	1
1	1	0

Symbole :



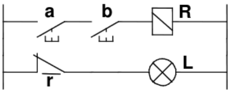
Opérateur NON-OU

Algèbre de Boole

Opérateurs dérivés

Opérateur NON-ET

Schéma électrique :



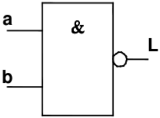
Equation :

$$L = \overline{a \,.\, b} = \overline{a} + \overline{b}$$

Table de vérité :

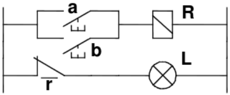
a	b	L
0	0	1
0	1	1
1	0	1
1	1	0

Symbole :



Opérateur NON-OU

Schéma électrique :



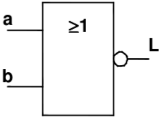
Equation :

$$L = \overline{a + b} = \overline{a} \,.\, \overline{b}$$

Table de vérité :

a	b	L
0	0	1
0	1	0
1	0	0
1	1	0

Symbole :

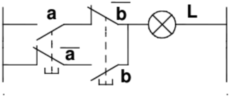


Algèbre de Boole

Opérateurs dérivés

Opérateur OU-EXCLUSIF

Schéma électrique :



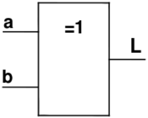
Equation :

$L = a \oplus b = \overline{a}\,.\,b + a\,.\,\overline{b}$

Table de vérité :

a	b	L
0	0	0
0	1	1
1	0	1
1	1	0

Symbole :

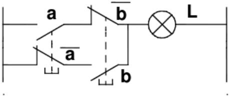


Opérateur NON-OU-EXCLUSIF

Algèbre de Boole
Opérateurs dérivés

Opérateur OU-EXCLUSIF

Schéma électrique :



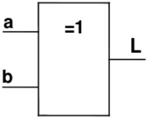
Equation :

$L = a \oplus b = \overline{a} \, . \, b + a \, . \, \overline{b}$

Table de vérité :

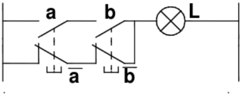
a	b	L
0	0	0
0	1	1
1	0	1
1	1	0

Symbole :



Opérateur NON-OU-EXCLUSIF

Schéma électrique :



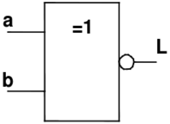
Equation :

$L = \overline{a} \, . \, \overline{b} + a \, . \, b$

Table de vérité :

a	b	L
0	0	1
0	1	0
1	0	0
1	1	1

Symbole :



Algèbre de Boole
Dualité

Chaque expression logique possède une expression **duale**.

L'expression duale est obtenue en échangeant les opérateurs et les valeurs logiques dans l'expression d'origine. Ainsi :

- l'opérateur $+$ devient $\cdot$;
- l'opérateur $\cdot$ devient $+$;
- la valeur logique "0" devient "1" ;
- la valeur logique "1" devient "0".

Par exemple, on a :

$$X + 0 = X$$

On a aussi :

$$X \cdot 1 = X$$

Ne pas confondre avec l'application du théorème de De Morgan.

Algèbre de Boole
Dualité

Chaque expression logique possède une expression **duale**.

L'expression duale est obtenue en échangeant les opérateurs et les valeurs logiques dans l'expression d'origine. Ainsi :

- l'opérateur $+$ devient $\cdot$;
- l'opérateur $\cdot$ devient $+$;
- la valeur logique "0" devient "1" ;
- la valeur logique "1" devient "0".

Par exemple, on a :

$$X + 0 = X$$

On a aussi :

$$X \cdot 1 = X$$

Ne pas confondre avec l'application du théorème de De Morgan.

Algèbre de Boole
Dualité

Chaque expression logique possède une expression **duale**.

L'expression duale est obtenue en échangeant les opérateurs et les valeurs logiques dans l'expression d'origine. Ainsi :

- l'opérateur $+$ devient $\cdot$;
- l'opérateur $\cdot$ devient $+$;
- la valeur logique "0" devient "1" ;
- la valeur logique "1" devient "0".

Par exemple, on a :

$$X + 0 = X$$

On a aussi :

$$X \cdot 1 = X$$

Ne pas confondre avec l'application du théorème de De Morgan.

Algèbre de Boole
Dualité

Chaque expression logique possède une expression **duale**.

L'expression duale est obtenue en échangeant les opérateurs et les valeurs logiques dans l'expression d'origine. Ainsi :

- l'opérateur $+$ devient $\cdot$;
- l'opérateur $\cdot$ devient $+$;
- la valeur logique "0" devient "1" ;
- la valeur logique "1" devient "0".

Par exemple, on a :

$$X + 0 = X$$

On a aussi :

$$X \cdot 1 = X$$

Ne pas confondre avec l'application du théorème de De Morgan.

Algèbre de Boole

Théorèmes de l'algèbre de Boole - Propriétés

Les colonnes gauche et droite représentent les formes duales.

Commutativité	
$X + Y = Y + X$	$X \cdot Y = Y \cdot X$

Associativité	
$(X + Y) + Z = X + (Y + Z) = X + Y + Z$	$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z$

Distributivité	
$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$	$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

Eléments neutres	
$X + 0 = X$	$X \cdot 1 = X$
$X + 1 = 1$	$X \cdot 0 = 0$

Complémentarité	
$X + \overline{X} = 1$	$X \cdot \overline{X} = 0$

Théorème d'idempotence	
$X + X = X$	$X \cdot X = X$

Algèbre de Boole

Théorèmes de l'algèbre de Boole - Théorèmes

Les colonnes gauche et droite représentent les formes duales.

Théorème d'involution

$$\overline{(\overline{X})} = X$$

Théorème de DeMorgan

$$\overline{(X + Y + Z + ...)} = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \cdot ... \qquad \overline{(X \cdot Y \cdot Z \cdot ...)} = \overline{X} + \overline{Y} + \overline{Z} + ...$$

$$\overline{f(X_1, X_2, X_3, ..., X_n, 0, 1, +, \cdot)} = f(\overline{X_1}, \overline{X_2}, \overline{X_3}, ..., \overline{X_n}, 1, 0, \cdot, +)$$

Théorème de multiplication et de factorisation

$$(X + Y) \cdot (\overline{X} + Z) = X \cdot Z + \overline{X} \cdot Y$$

$$X \cdot Y + X \cdot \overline{Z} = (X + Z) \cdot (\overline{X} + Y)$$

Théorème du consensus

$$X \cdot Y + Y \cdot Z + \overline{X} \cdot Z = X \cdot Y + \overline{X} \cdot Z$$

$$(X + Y) \cdot (Y + Z) \cdot (\overline{X} + Z) = (X + Y) \cdot (\overline{X} + Z)$$

Synthèse de circuits logiques

Synthèse

La **synthèse** de fonctions combinatoires consiste, à partir d'une table de vérité ou d'une expression booléenne, à spécifier les **opérateurs matériels** permettant l'implémentation de la table ou de l'expression correspondante dans un système réel.

Méthodes

- une **méthode analytique**, se basant sur les théorèmes de l'albèbre de Boole ;
- une **méthode graphique**, se basant sur l'utilisation de tableau de Karnaugh ;
- une **description comportementale**, à l'aide de langage de description matérielle de haut niveau (type VHDL ou Verilog).

Synthèse de circuits logiques

Synthèse

La **synthèse** de fonctions combinatoires consiste, à partir d'une table de vérité ou d'une expression booléenne, à spécifier les **opérateurs matériels** permettant l'implémentation de la table ou de l'expression correspondante dans un système réel.

Méthodes

- une **méthode analytique**, se basant sur les théorèmes de l'albèbre de Boole ;
- une **méthode graphique**, se basant sur l'utilisation de tableau de Karnaugh ;
- une **description comportementale**, à l'aide de langage de description matérielle de haut niveau (type VHDL ou Verilog).

Synthèse de circuits logiques

Synthèse graphique par tableau de Karnaugh

Les tables de Karnaugh sont une **méthode graphique** de simplification d'expression booléenne. Elle est basée sur l'application du théorème d'unification $(A \cdot (B + \overline{B}) = A)$.

Celle-ci s'applique à une fonction logique de la manière suivante : si

$$F(A, B, C, D) = G(A, C, D) \cdot B + G(A, C, D) \cdot \overline{B}$$

(où $G(A, C, D)$ est un certain produit), alors F ne dépend pas de B et peut s'écrire :

$$F(A, B, C, D) = G(A, C, D)$$

.

Synthèse de circuits logiques

Synthèse graphique par tableau de Karnaugh

Les tables de Karnaugh sont une **méthode graphique** de simplification d'expression booléenne. Elle est basée sur l'application du théorème d'unification ($A \cdot (B + \overline{B}) = A$).

Celle-ci s'applique à une fonction logique de la manière suivante : si

$$F(A, B, C, D) = G(A, C, D) \cdot B + G(A, C, D) \cdot \overline{B}$$

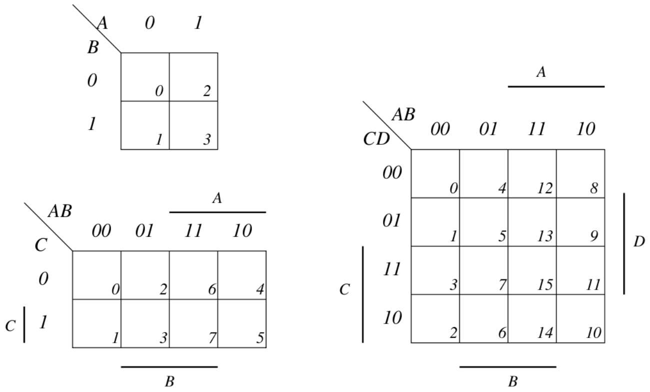
(où $G(A, C, D)$ est un certain produit), alors F ne dépend pas de B et peut s'écrire :

$$F(A, B, C, D) = G(A, C, D)$$

.

Synthèse de circuits logiques

Synthèse graphique par tableau de Karnaugh - Constitution d'un tableau de Karnaugh



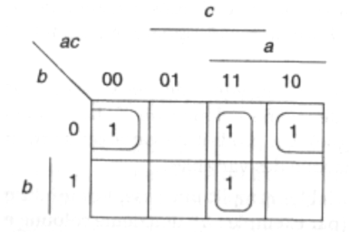
Utilisation du *code Gray* pour la numérotation des lignes.

Synthèse de circuits logiques

Synthèse graphique par tableau de Karnaugh - Simplification

Soit la fonction $Z(a,b,c)$ telle que :
$Z = \overline{a}\cdot\overline{b}\cdot\overline{c} + a\cdot\overline{b}\cdot c + a\cdot\overline{b}\cdot\overline{c} + a\cdot b\cdot c$

Soit la fonction W définit comme suit :

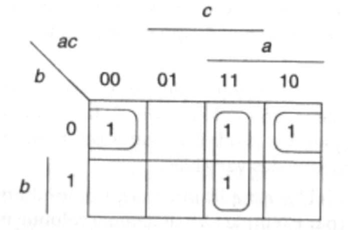


L'expression simplifiée est :
$Z(a,b,c) = a\cdot c + \overline{b}\cdot\overline{c}$

Synthèse de circuits logiques

Synthèse graphique par tableau de Karnaugh - Simplification

Soit la fonction $Z(a,b,c)$ telle que :
$Z = \overline{a}\cdot\overline{b}\cdot\overline{c} + a\cdot\overline{b}\cdot c + a\cdot\overline{b}\cdot\overline{c} + a\cdot b\cdot c$



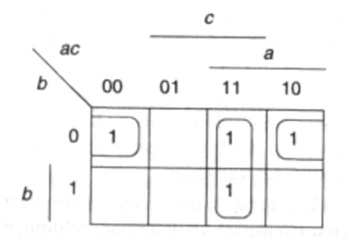
L'expression simplifiée est :
$Z(a,b,c) = a\cdot c + \overline{b}\cdot\overline{c}$

Soit la fonction W définit comme suit :

Synthèse de circuits logiques

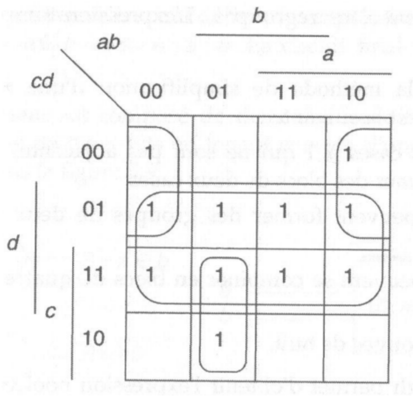
Synthèse graphique par tableau de Karnaugh - Simplification

Soit la fonction $Z(a,b,c)$ telle que :
$Z = \overline{a}\cdot\overline{b}\cdot\overline{c} + a\cdot\overline{b}\cdot c + a\cdot\overline{b}\cdot\overline{c} + a\cdot b\cdot c$



L'expression simplifiée est :
$Z(a,b,c) = a\cdot c + \overline{b}\cdot\overline{c}$

Soit la fonction W définit comme suit :



Synthèse de circuits logiques

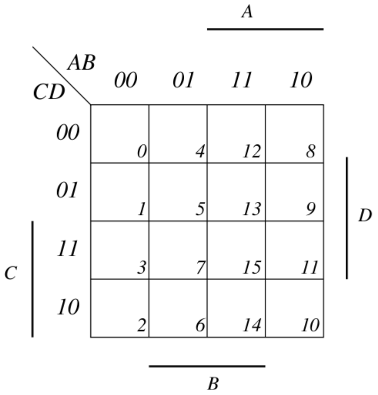
Synthèse graphique par tableau de Karnaugh - Exemple d'utilisation d'un tableau de Karnaugh

$$f(a,b,c,d) = a \cdot b \cdot \overline{d} + a \cdot \overline{b} \cdot \overline{d} + \overline{a} \cdot b \cdot c \cdot \overline{d} + a \cdot b \cdot c \cdot d + a \cdot \overline{b} \cdot c \cdot d$$

Synthèse de circuits logiques

Synthèse graphique par tableau de Karnaugh - Exemple d'utilisation d'un tableau de Karnaugh

$$f(a,b,c,d) = a \cdot b \cdot \overline{d} + a \cdot \overline{b} \cdot \overline{d} + \overline{a} \cdot b \cdot c \cdot \overline{d} + a \cdot b \cdot c \cdot d + a \cdot \overline{b} \cdot c \cdot d$$



Synthèse de circuits logiques

Tableau à cases indéfinies

Les problèmes réels conduisent souvent à des spécifications incomplètes des fonctions logiques. Des valeurs indéterminées (notées "X") se présentent dans le tableau. On peut alors adopter la valeur qui conduit à la meilleure simplification.

F	00	01	11	10
00	0	X	X	X
01	0	1	X	X
11	0	1	1	X
10	0	X	X	X

A (columns 11, 10); B (columns 01, 11); C (rows 11, 10); D (rows 01, 11)

Ainsi, la fonction F peut s'écrire : $F = B$.

Synthèse de circuits logiques

Tableau à cases indéfinies

Les problèmes réels conduisent souvent à des spécifications incomplètes des fonctions logiques. Des valeurs indéterminées (notées "X") se présentent dans le tableau. On peut alors adopter la valeur qui conduit à la meilleure simplification.

F	00	01	11	10
00	0	X	X	X
01	0	1	X	X
11	0	1	1	X
10	0	X	X	X

F	00	01	11	10
00	0	*1*	*1*	*0*
01	0	1	*1*	*0*
11	0	1	1	*0*
10	0	*1*	*1*	*0*

Ainsi, la fonction F peut s'écrire : $F = B$.

Synthèse de circuits logiques

Tableau à cases indéfinies

Les problèmes réels conduisent souvent à des spécifications incomplètes des fonctions logiques. Des valeurs indéterminées (notées "X") se présentent dans le tableau. On peut alors adopter la valeur qui conduit à la meilleure simplification.

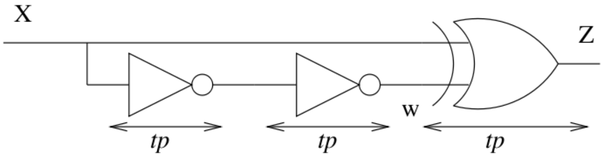
	F	00	01	11	10	
				A	A	
	00	0	X	X	X	
	01	0	1	X	X	D
C	11	0	1	1	X	D
C	10	0	X	X	X	
			B	B		

	F	00	01	11	10	
				A	A	
	00	0	*1*	*1*	*0*	
	01	0	1	*1*	*0*	D
C	11	0	1	1	*0*	D
C	10	0	*1*	*1*	*0*	
			B	B		

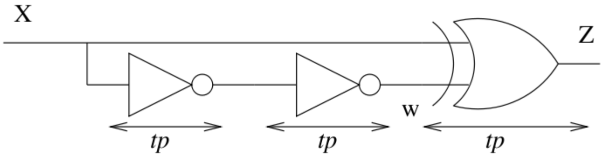
Ainsi, la fonction F peut s'écrire : $F = B$.

Temps de propagation et aléas

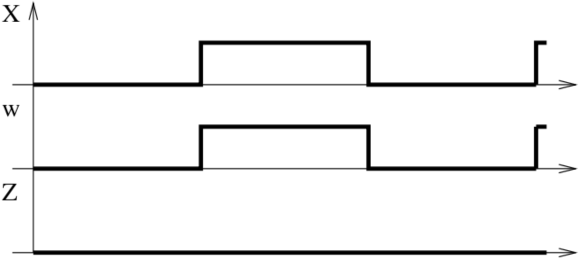
Temps de propagation et aléas



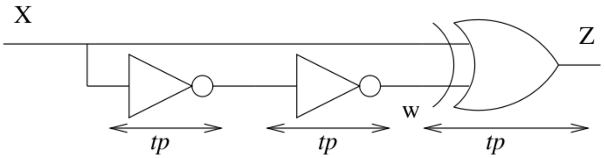
Temps de propagation et aléas



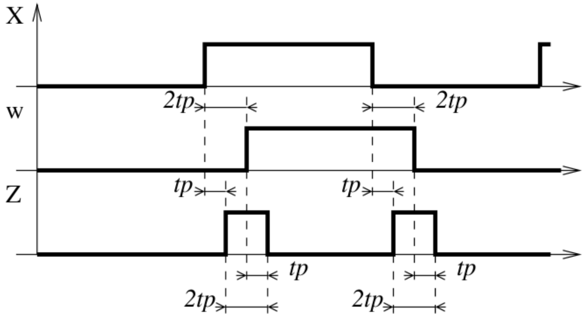
Chronogramme dans le cas où tp=0



Temps de propagation et aléas



Chronogramme dans le cas où tp = 125ns



Fonctions combinatoires standard

6 Fonctions combinatoires standard
- Multiplexeurs et démultiplexeurs
- Codeurs, décodeurs et transcodeurs
- Circuits arithmétiques

Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs

Comment relier **plusieurs sources** à **plusieurs destinataires** *via* **UN** seul câble ?

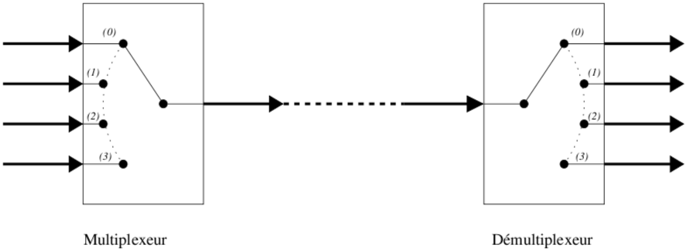
Les dénominations habituelles sont :

- un multiplexeur "1 parmi N" (N entrées et 1 sortie) ;
- un démultiplexeur à N voies (1 entrée et N sorties).

Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs

Comment relier **plusieurs sources** à **plusieurs destinataires** *via* **UN** seul câble ?



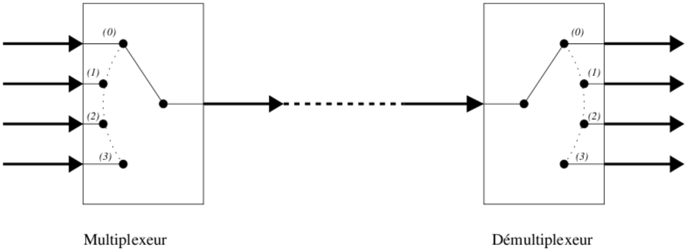
Les dénominations habituelles sont :

- un multiplexeur "1 parmi N" (N entrées et 1 sortie) ;
- un démultiplexeur à N voies (1 entrée et N sorties).

Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs

Comment relier **plusieurs sources** à **plusieurs destinataires** *via* **UN** seul câble ?

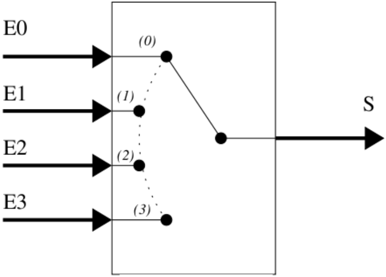


Les dénominations habituelles sont :

- un multiplexeur "1 parmi N" (N entrées et 1 sortie) ;
- un démultiplexeur à N voies (1 entrée et N sorties).

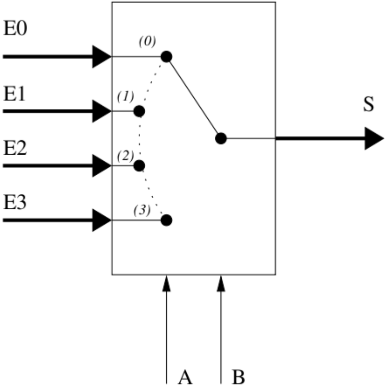
Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs - Multiplexeur pour 4 signaux binaires



Fonctions combinatoires standard

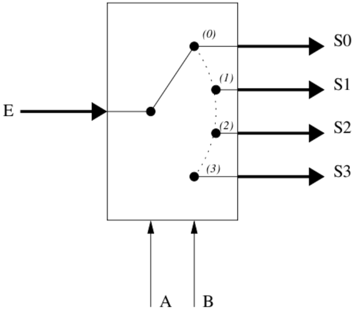
Multiplexeurs et démultiplexeurs - Multiplexeur pour 4 signaux binaires



N°	A	B	S
0	0	0	E0
1	0	1	E1
2	1	0	E2
3	1	1	E3

Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs - Démultiplexeur pour 4 signaux binaires

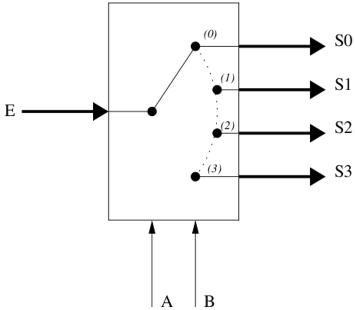


N°	A	B	S0	S1	S2	S3
0	0	0	E	0	0	0
1	0	1	0	E	0	0
2	1	0	0	0	E	0
3	1	1	0	0	0	E

La plupart des multiplexeurs/démultiplexeurs sont munis d'une entrée supplémentaire permettant d'activer ou non la fonction dans son ensemble. Lors de la désactivation, les sorties peuvent être placées soit à l'état inactif ("0") soit dans un état dit "haute impédance".

Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs - Démultiplexeur pour 4 signaux binaires

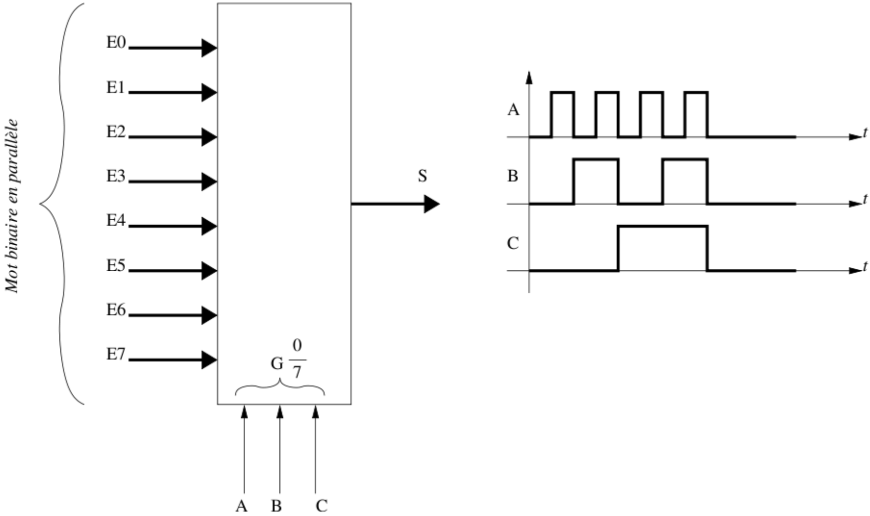


N°	A	B	S0	S1	S2	S3
0	0	0	E	0	0	0
1	0	1	0	E	0	0
2	1	0	0	0	E	0
3	1	1	0	0	0	E

La plupart des multiplexeurs/démultiplexeurs sont munis d'une entrée supplémentaire permettant d'activer ou non la fonction dans son ensemble. Lors de la désactivation, les sorties peuvent être placées soit à l'état inactif ("0") soit dans un état dit "haute impédance".

Fonctions combinatoires standard

Multiplexeurs et démultiplexeurs - Convertisseur parallèle/série



Fonctions combinatoires standard

Codeurs, décodeurs et transcodeurs

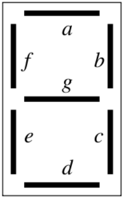
Ces dispositifs assurent la traduction d'une représentation de l'information vers une autre.

Codeurs

- Codeur binaire : possèdant N entrées, délivre sur ses sorties le code binaire du numéro de l'entrée activée.

Décodeurs

- Décodeur décimal : fonction inverse du codeur binaire (pour un nombre compris entre 0 et 9).
- Décodeur 7 segments.



d3	d2	d1	d0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1

Fonctions combinatoires standard

Circuits arithmétiques - Comparateur à 2 nombres entiers positifs

Soient deux nombres A et B, codés sur n bits, on souhaite alors disposer de signaux logiques $S_=$, $S_<$ et $S_>$ indiquant respectivement si $A = B$ ou $A < B$ ou $A > B$.

Des boitiers MSI à 4 bits cascadables existent (74xx85 ou 74xx688 par exemple). Ici l'étude sera faite pour des nombres A et B de 2 bits.

A		B		$S_=$	$S_<$	$S_>$
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à 1 bit (ou demi-additionneur)

La somme de deux nombres codés en binaire naturel sur un bit nécessite une représentation sur 2 bits (un bit de somme et un bit de retenue).

Ce circuit est souvent appelé demi-additionneur : il ne peut pas être utilisé dans une addition à plus d'un bit.

a	b	s	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s = \overline{a} \cdot b + a \cdot \overline{b} = a \oplus b$$

$$co = a \cdot b$$

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à 1 bit (ou demi-additionneur)

La somme de deux nombres codés en binaire naturel sur un bit nécessite une représentation sur 2 bits (un bit de somme et un bit de retenue).

Ce circuit est souvent appelé demi-additionneur : il ne peut pas être utilisé dans une addition à plus d'un bit.

a	b	s	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s = \overline{a} \cdot b + a \cdot \overline{b} = a \oplus b$$

$$co = a \cdot b$$

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à 1 bit (ou demi-additionneur)

La somme de deux nombres codés en binaire naturel sur un bit nécessite une représentation sur 2 bits (un bit de somme et un bit de retenue).

Ce circuit est souvent appelé demi-additionneur : il ne peut pas être utilisé dans une addition à plus d'un bit.

a	b	s	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s = \overline{a} \cdot b + a \cdot \overline{b} = a \oplus b$$

$$co = a \cdot b$$

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à plusieurs bits (ou additionneur complet) - 1/2

L'addition à plusieurs bits se réalise à la main avec le même algorithme que l'addition décimale.

retenues	1	1	1	1	
		1	1	0	1
+		1	0	1	1
	1	1	0	0	0

a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

L'addition de 2 signaux binaires doit alors traiter **3 opérandes** (a, b et la retenue ci. Elle doit toujours produire 2 résultats s et co.

$$s = \overline{a}\cdot\overline{b}\cdot ci + \overline{a}\cdot b\cdot\overline{ci} + a\cdot\overline{b}\cdot\overline{ci} + a\cdot b\cdot ci$$

$$co = a\cdot b + a\cdot ci + b\cdot ci$$

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à plusieurs bits (ou additionneur complet) - 1/2

L'addition à plusieurs bits se réalise à la main avec le même algorithme que l'addition décimale.

retenues	1	1	1	1	
		1	1	0	1
+		1	0	1	1
	1	1	0	0	0

L'addition de 2 signaux binaires doit alors traiter **3 opérandes** (a, b et la retenue ci. Elle doit toujours produire 2 résultats s et co.

a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$s = \overline{a} \cdot \overline{b} \cdot ci + \overline{a} \cdot b \cdot \overline{ci} + a \cdot \overline{b} \cdot \overline{ci} + a \cdot b \cdot ci$$

$$co = a \cdot b + a \cdot ci + b \cdot ci$$

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à plusieurs bits (ou additionneur complet) - 1/2

L'addition à plusieurs bits se réalise à la main avec le même algorithme que l'addition décimale.

retenues	1	1	1	1	
		1	1	0	1
+		1	0	1	1
	1	1	0	0	0

L'addition de 2 signaux binaires doit alors traiter **3 opérandes** (a, b et la retenue ci. Elle doit toujours produire 2 résultats s et co.

a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

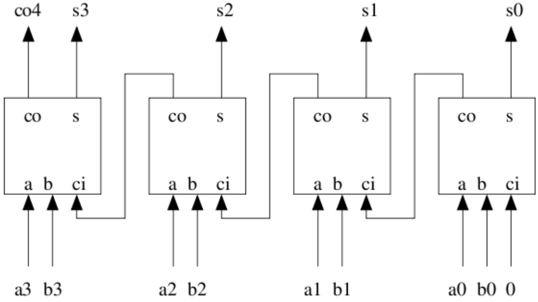
$$s = \overline{a}\cdot\overline{b}\cdot ci + \overline{a}\cdot b\cdot\overline{ci} + a\cdot\overline{b}\cdot\overline{ci} + a\cdot b\cdot ci$$

$$co = a\cdot b + a\cdot ci + b\cdot ci$$

Fonctions combinatoires standard

Circuits arithmétiques - Additionneur à plusieurs bits (ou additionneur complet) - 2/2

Les équations précédentes permettent de construire l'additionneur complet qui peut ensuite être utilisé comme élément de base d'un additionneur à plusieurs bits.



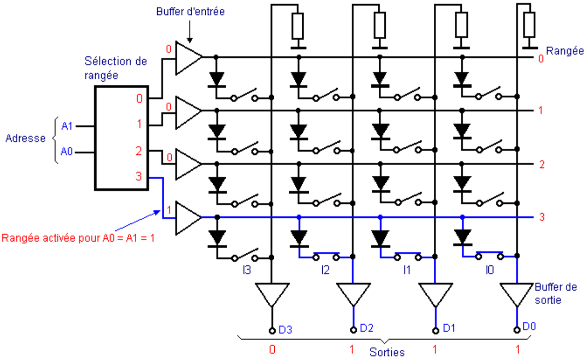
Des fonctions logiques combinatoires plus complexes

Les mémoires mortes (ROM)

- ROM (Read Only Memory) dont le contenu est défini lors de la fabrication ;
- PROM (Programmable Read Only Memory) sont programmables par l'utilisateur, mais une seule fois en raison du moyen de stockage, les données sont stockées par des fusibles ;
- EPROM (Erasable Programmable Read Only Memory) sont effaçables et programmables par l'utilisateur ;
- EEPROM (Electrically Erasable Programmable Read Only Memory) ;
- UVPROM (Ultra Violet Programmable Read Only Memory).

Des fonctions logiques combinatoires plus complexes

Les mémoires mortes (ROM) - Principe



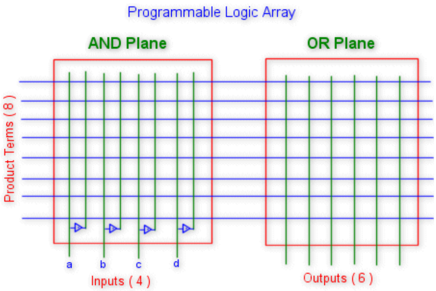
2 signaux (A_0 et A_1) permettant l'adressage de 2^2 mots de 4 bits (D_3, D_2, D_1, D_0).
Exemple : 32ko $= 2^{15}$ mots de 8 bits

Des fonctions logiques combinatoires plus complexes

Les circuits programmables

PLA = "Programmable Logic Array"

L'architecture de ces circuits est basée sur une des deux formes canoniques pour réaliser un composant *universel*.



CPLD = "Complex Programmable Logic Device"
FPGA = "Field Programmable Gate Array"