Algèbre de Boole, circuits logiques

Vincent Risch, septembre 2006, révision mai 2014

I.U.T., Aix-Marseille Université

Plan

- Circuits combinatoires → *dispositifs de calcul*
  - Algèbres de Boole : rappels
  - Circuits élémentaires
  - Circuits utiles : décodeurs, multiplexeurs, démultiplexeurs, additionneur
- Circuits séquentiels → *dispositifs de mémorisation* : on verra (vaguement) plus tard...

Algèbres de Boole

- Définition *algèbrique* : treillis distributif complémenté

Algèbres de Boole

- Définition *algèbrique* : treillis distributif complémenté
- Définition *axiomatique* : ensemble contenant $\{0, 1\}$ et sur lequel sont définies deux opérations binaires, $\wedge$ et $\vee$, et une opération unaire, $\overline{}$, vérifiant un ensemble défini d'axiomes

Algèbres de Boole

- Définition *algèbrique* : treillis distributif complémenté
- Définition *axiomatique* : ensemble contenant $\{0, 1\}$ et sur lequel sont définies deux opérations binaires, $\wedge$ et $\vee$, et une opération unaire, $\overline{}$, vérifiant un ensemble défini d'axiomes
- Lien entre les deux définitions :

$$x \leq y \Leftrightarrow x \wedge y = x \Leftrightarrow x \vee y = y$$

où $\wedge$ et $\vee$ désignent les bornes inférieures du treillis et 0 et 1 les éléments nuls et universels

Axiomes

$$
\begin{array}{ll}
x \wedge \overline{x} = 0 & x \vee \overline{x} = 1 \\
x \wedge 0 = 0 & x \vee 1 = 1 \\
x \wedge 1 = x & x \vee 0 = x \\
x \wedge x = x & x \vee x = x \\
x \wedge y = y \wedge x & x \vee y = y \vee x \\
x \wedge (y \wedge z) = (x \wedge y) \wedge z & x \vee (y \vee z) = (x \vee y) \vee z \\
x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) & x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z \\
x \wedge (x \vee y) = x & x \vee (x \wedge y) = x
\end{array}
$$

$\rightarrow$ approche axiomatique = logique propositionnelle
$\rightarrow$ architecture : on note traditionnellement $\wedge$ par “.”, et $\vee$ par “$+$”

Fonctions booléennes

Fonctions sur $\{0, 1\}$ définies à l'aide des opérateurs booléens :

- une *table de vérité* peut suffire à leur étude ;

Fonctions booléennes

Fonctions sur $\{0, 1\}$ définies à l'aide des opérateurs booléens :

- une *table de vérité* peut suffire à leur étude ;
- toute fonction booléenne peut se représenter sous forme *canonique*,

Fonctions booléennes

Fonctions sur $\{0, 1\}$ définies à l'aide des opérateurs booléens :

- une *table de vérité* peut suffire à leur étude ;
- toute fonction booléenne peut se représenter sous forme *canonique*,
  - *conjonctive* : “produits” de “sommes” de toutes ses variables, par ex : $f(x, y) = (x + \overline{y}).(\overline{x} + y)$

Fonctions booléennes

Fonctions sur $\{0, 1\}$ définies à l'aide des opérateurs booléens :

- une *table de vérité* peut suffire à leur étude ;
- toute fonction booléenne peut se représenter sous forme *canonique*,
  - *conjonctive* : "produits" de "sommes" de toutes ses variables, par ex : $f(x, y) = (x + \overline{y}).(\overline{x} + y)$
  - *disjonctive* : "sommes" de "produits" de toutes ses variables, par ex : $f(x, y, z) = x.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.z$

Fonctions booléennes

Fonctions sur $\{0,1\}$ définies à l'aide des opérateurs booléens :

- une *table de vérité* peut suffire à leur étude ;
- toute fonction booléenne peut se représenter sous forme *canonique*,
  - *conjonctive* : "produits" de "sommes" de toutes ses variables, par ex : $f(x,y) = (x+\overline{y}).(\overline{x}+y)$
  - *disjonctive* : "sommes" de "produits" de toutes ses variables, par ex : $f(x,y,z) = x.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.z$
- toute fonction booléenne peut être *simplifiée* de façon à minimiser (en général pas de façon unique) le nombre d'occurences de chaque opérateur.

Exemple

$$f(x,y,z) = (x \to y) \oplus (y \to z) \oplus (z \to x)$$

x	y	z	$x \to y$	$y \to z$	$z \to x$	$(x \to y) \oplus (y \to z)$	$f(x,y,z)$
0	0	0	1	1	1	0	1
0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0
0	1	1	1	1	0	0	0
1	0	0	0	1	1	1	0
1	0	1	0	1	1	1	0
1	1	0	1	0	1	1	0
1	1	1	1	1	1	0	1

On obtient donc : $f(x,y,z) = \overline{x}.\overline{y}.\overline{z} + x.y.z$

Simplification des fonctions booléennes

→ *Identifier toutes les factorisations possibles par des tautologies*

- Simplification algébrique
- Tables de Karnaugh
- Algorithme de Nelson
- Algorithme de Quine–McCluskey
- Méthode des consensus
- ...

Simplification algébrique

Simplification à la main, exemple :

$$
\begin{aligned}
f(x_1, x_2, x_3, x_4) &= \overline{x_1}.\overline{x_2}.x_3 + x_1.\overline{x_2}.x_3 + x_2.x_3 + x_4 \\
&= (\overline{x_1} + x_1).\overline{x_2}.x_3 + x_2.x_3 + x_4 \\
&= \overline{x_2}.x_3 + x_2.x_3 + x_4 \\
&= (\overline{x_2} + x_2).x_3 + x4 \\
&= x_3 + x_4
\end{aligned}
$$

Tables de Karnaugh

- Représentation graphique sous forme normale disjonctive

Tables de Karnaugh

- Représentation graphique sous forme normale disjonctive
- Utilise le binaire réfléchi pour le repérage des factorisations

Tables de Karnaugh

- Représentation graphique sous forme normale disjonctive
- Utilise le binaire réfléchi pour le repérage des factorisations
- Table à deux variables :

$x_2 \backslash x_1$	$\overline{x_1}$	x_1
$\overline{x_2}$	$\overline{x_1}.\overline{x_2}$	$x_1\overline{x_2}$
x_2	$\overline{x_1}.x_2$	$x_1.x_2$

Tables de Karnaugh

- Représentation graphique sous forme normale disjonctive
- Utilise le binaire réfléchi pour le repérage des factorisations
- Table à deux variables :

$x_2 \backslash x_1$	$\overline{x_1}$	x_1
$\overline{x_2}$	$\overline{x_1}.\overline{x_2}$	$x_1\overline{x_2}$
x_2	$\overline{x_1}.x_2$	$x_1.x_2$

$$f(x_1, x_2) = x_1\overline{x_2} + x_1.x_2$$

Tables de Karnaugh

- Représentation graphique sous forme normale disjonctive
- Utilise le binaire réfléchi pour le repérage des factorisations
- Table à deux variables :

$x_2 \backslash x_1$	$\overline{x_1}$	x_1
$\overline{x_2}$	$\overline{x_1}.\overline{x_2}$	$x_1\overline{x_2}$
x_2	$\overline{x_1}.x_2$	$x_1.x_2$

$$f(x_1, x_2) = x_1\overline{x_2} + x_1.x_2 = (\overline{x_2} + x_2).x_1 = x_1$$

Karnaugh : stratégie de construction

- Ordre binaire réfléchi (*permet les factorisations et “ferme” la table sur elle-même)*

Karnaugh : stratégie de construction

- Ordre binaire réfléchi (*permet les factorisations et “ferme” la table sur elle-même)*
- Regroupements de 1 par multiples de puissances de 2 (2, 4, 8, 16, $\ldots$)

Karnaugh : stratégie de construction

- Ordre binaire réfléchi (*permet les factorisations et “ferme” la table sur elle-même)*
- Regroupements de 1 par multiples de puissances de 2 (2, 4, 8, 16, . . .)
- Les regroupements doivent être réalisés de telle sorte qu’un maximum de valeurs 1 soient englobées dans un minimum de regroupements (*réaliser les plus gros regroupements possibles, et en plus petit nombre possible*)

Karnaugh : stratégie de construction

- Ordre binaire réfléchi (*permet les factorisations et "ferme" la table sur elle-même)*
- Regroupements de 1 par multiples de puissances de 2 (2, 4, 8, 16, . . .)
- Les regroupements doivent être réalisés de telle sorte qu'un maximum de valeurs 1 soient englobées dans un minimum de regroupements (*réaliser les plus gros regroupements possibles, et en plus petit nombre possible*)

$\rightarrow$ *Pas forcément "une" simplification unique. . .*

Exemple de Table de Karnaugh

$$f(x_1,x_2,x_3,x_4) = \overline{x_1}.\overline{x_2}.\overline{x_3}.\overline{x_4} + \overline{x_1}.\overline{x_2}.x_3.x_4 + \overline{x_1}.\overline{x_2}.x_3.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.x_4 + x_1.x_2.\overline{x_3}.\overline{x_4} + x_1.x_2.\overline{x_3}.x_4 + x_1.\overline{x_2}.\overline{x_3}.\overline{x_4} + x_1.\overline{x_2}.x_3.\overline{x_4}$$

Exemple de Table de Karnaugh

$$
\begin{aligned}
f(x_1, x_2, x_3, x_4) = \quad & \overline{x_1}.\overline{x_2}.\overline{x_3}.\overline{x_4} + \overline{x_1}.\overline{x_2}.x_3.x_4 + \overline{x_1}.\overline{x_2}.x_3.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.\overline{x_4} + \\
& \overline{x_1}.x_2.\overline{x_3}.x_4 + x_1.x_2.\overline{x_3}.\overline{x_4} + x_1.x_2.\overline{x_3}.x_4 + x_1.\overline{x_2}.\overline{x_3}.\overline{x_4} + \\
& x_1.\overline{x_2}.x_3.\overline{x_4}
\end{aligned}
$$

$x_3\ x_4$ \ $x_1\ x_2$	00	01	11	10
00	1		1	1
01	1	1		
11	1	1		
10	1			1

Exemple de Table de Karnaugh

$$f(x_1, x_2, x_3, x_4) = \overline{x_1}.\overline{x_2}.\overline{x_3}.\overline{x_4} + \overline{x_1}.\overline{x_2}.x_3.x_4 + \overline{x_1}.\overline{x_2}.x_3.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.x_4 + x_1.x_2.\overline{x_3}.\overline{x_4} + x_1.x_2.\overline{x_3}.x_4 + x_1.\overline{x_2}.\overline{x_3}.\overline{x_4} + x_1.\overline{x_2}.x_3.\overline{x_4}$$

$x_3\ x_4$ \ $x_1\ x_2$	00	01	11	10
00	1		1	1
01	1	1		
11	1	1		
10	1			1

$$f(x_1, x_2, x_3, x_4) = \overline{x_2}.\overline{x_4} + \overline{x_1}.x_4 + x_1.\overline{x_3}.\overline{x_4}$$

Exemple de Table de Karnaugh

$$f(x_1, x_2, x_3, x_4) = \overline{x_1}.\overline{x_2}.\overline{x_3}.\overline{x_4} + \overline{x_1}.\overline{x_2}.x_3.x_4 + \overline{x_1}.\overline{x_2}.x_3.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.\overline{x_4} + \overline{x_1}.x_2.\overline{x_3}.x_4 + x_1.x_2.\overline{x_3}.\overline{x_4} + x_1.x_2.\overline{x_3}.x_4 + x_1.\overline{x_2}.\overline{x_3}.\overline{x_4} + x_1.\overline{x_2}.x_3.\overline{x_4}$$

$x_3\ x_4$ \ $x_1\ x_2$	00	01	11	10
00	1		1	1
01	1	1		
11	1	1		
10	1			1

$f(x_1, x_2, x_3, x_4) = \overline{x_2}.\overline{x_4} + \overline{x_1}.x_4 + x_1.\overline{x_3}.\overline{x_4}$ $+\ \overline{x_1}.\overline{x_2}$: NON !

- Chacun des opérateurs booléens est implémentable électroniquement à l’aide de *transistors* ;

Circuits logiques

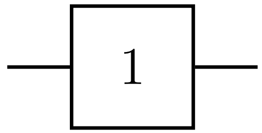
- Chacun des opérateurs booléens est implémentable électroniquement à l'aide de *transistors* ;
- Il est donc possible de d'implémenter n'importe quelle fonction booléenne par un circuit ;

Circuits logiques

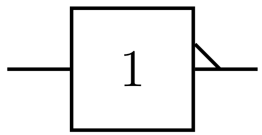
- Chacun des opérateurs booléens est implémentable électroniquement à l’aide de *transistors* ;
- Il est donc possible de d’implémenter n’importe quelle fonction booléenne par un circuit ;
- la représentation symbolique d’un circuit complexe se fait conventionnellement à l’aide de *portes logiques* dont chacune est associée à un opérateur booléen.

ANSI/IEEE Std 91-1984

Porte *IDENTITÉ*

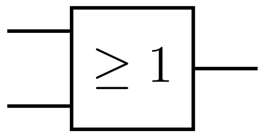


Porte *NON*

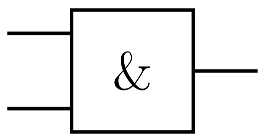


ou

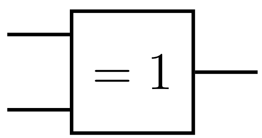
Porte *OU*



Porte *ET*



Porte *OU-EXCL*



- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.

Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction :

Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction :
  1. spécification de la fonction logique

Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction :
  1. spécification de la fonction logique
  2. tables de vérité

Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction :
  1. spécification de la fonction logique
  2. tables de vérité
  3. optimisation

Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction :
  1. spécification de la fonction logique
  2. tables de vérité
  3. optimisation
  4. tracé

Exemple

On considère la fonction logique suivante :

- $f(x, y, z) = \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z}$
- La table de vérité correspondante est :

x	y	z	$f(x, y, z)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Exemple : simplification algébrique

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

Exemple : simplification algébrique

$$
\begin{aligned}
f(x,y,z) &= \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} \\
&= \overline{x}.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + \overline{x}.y.\overline{z} + \\
&\quad x.\overline{y}.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} + x.y.\overline{z}
\end{aligned}
$$

Exemple : simplification algébrique

$$
\begin{aligned}
f(x,y,z) &= \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} \\
&= \overline{x}.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + \overline{x}.y.\overline{z} + \\
&\quad x.\overline{y}.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} + x.y.\overline{z} \\
&= \overline{x}.(\overline{y} + y).\overline{z} + (\overline{x} + x).\overline{y}.\overline{z} + \\
&\quad x.\overline{y}.(\overline{z} + z) + (\overline{x} + x).y.\overline{z} + x.(y + \overline{y}).\overline{z}
\end{aligned}
$$

Exemple : simplification algébrique

$$
\begin{aligned}
f(x,y,z) &= \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} \\
&= \overline{x}.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + \overline{x}.y.\overline{z} + \\
&\quad x.\overline{y}.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} + x.y.\overline{z} \\
&= \overline{x}.(\overline{y} + y).\overline{z} + (\overline{x} + x).\overline{y}.\overline{z} + \\
&\quad x.\overline{y}.(\overline{z} + z) + (\overline{x} + x).y.\overline{z} + x.(y + \overline{y}).\overline{z} \\
&= \overline{x}.\overline{z} + \overline{y}.\overline{z} + x.\overline{y} + y.\overline{z} + x.\overline{z}
\end{aligned}
$$

Exemple : simplification algébrique

$$
\begin{aligned}
f(x,y,z) &= \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} \\
&= \overline{x}.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + \overline{x}.y.\overline{z} + \\
&\quad x.\overline{y}.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} + x.y.\overline{z} \\
&= \overline{x}.(\overline{y} + y).\overline{z} + (\overline{x} + x).\overline{y}.\overline{z} + \\
&\quad x.\overline{y}.(\overline{z} + z) + (\overline{x} + x).y.\overline{z} + x.(y + \overline{y}).\overline{z} \\
&= \overline{x}.\overline{z} + \overline{y}.\overline{z} + x.\overline{y} + y.\overline{z} + x.\overline{z} \\
&= (x + \overline{x}).\overline{z} + (\overline{y} + y).\overline{z} + x.\overline{y}
\end{aligned}
$$

Exemple : simplification algébrique

$$
\begin{aligned}
f(x,y,z) &= \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} \\
&= \overline{x}.\overline{y}.\overline{z} + \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + \overline{x}.y.\overline{z} + \\
&\quad x.\overline{y}.\overline{z} + x.\overline{y}.\overline{z} + x.\overline{y}.z + x.y.\overline{z} + x.y.\overline{z} \\
&= \overline{x}.(\overline{y} + y).\overline{z} + (\overline{x} + x).\overline{y}.\overline{z} + \\
&\quad x.\overline{y}.(\overline{z} + z) + (\overline{x} + x).y.\overline{z} + x.(y + \overline{y}).\overline{z} \\
&= \overline{x}.\overline{z} + \overline{y}.\overline{z} + x.\overline{y} + y.\overline{z} + x.\overline{z} \\
&= (x + \overline{x}).\overline{z} + (\overline{y} + y).\overline{z} + x.\overline{y} \\
&= x.\overline{y} + \overline{z}
\end{aligned}
$$

Exemple : table de Karnaugh

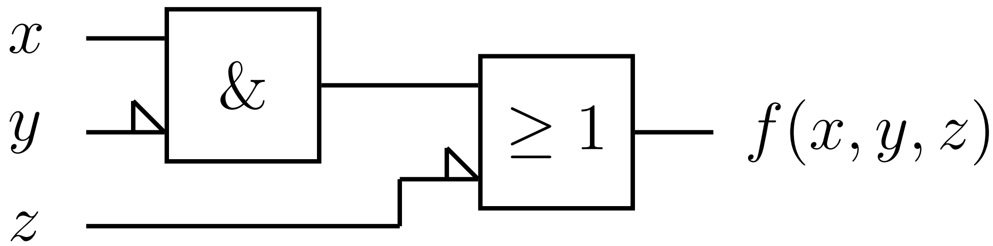
$z \backslash x\,y$	00	01	11	10
0	1	1	1	1
1				1

Exemple : table de Karnaugh

z \ $x\ y$	00	01	11	10
0	1	1	1	1
1				1

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

Exemple : circuit logique



Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.

Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction : (1) spécification de la fonction logique, (2) tables de vérité, (3) optimisation, (4) tracé.

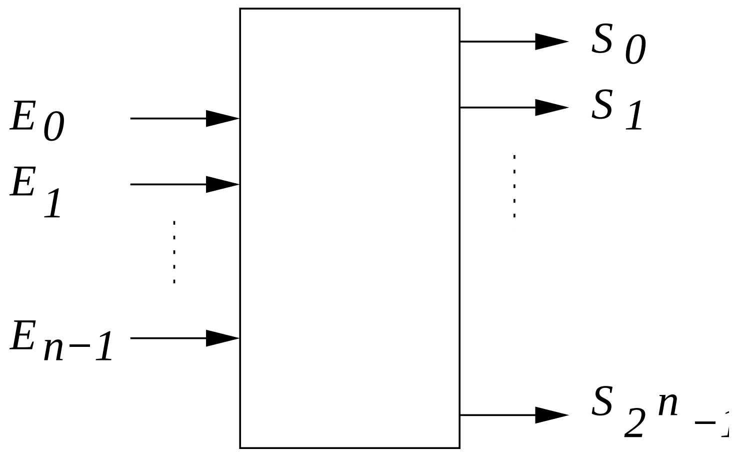
Circuits combinatoires

- Circuit combinatoire : circuit logique dont l'état des sorties ne dépend *que* des valeurs assignées aux variables d'entrée.
- Etapes de construction : (1) spécification de la fonction logique, (2) tables de vérité, (3) optimisation, (4) tracé.
- Quelques circuits fondamentaux dans la conception d'un ordinateur : décodeurs, multiplexeurs, démultiplexeurs, PLA, additionneurs. . .

Quelques circuits fondamentaux

1. Décodeurs
2. Multiplexeurs
3. Démultiplexeurs
4. Additionneurs

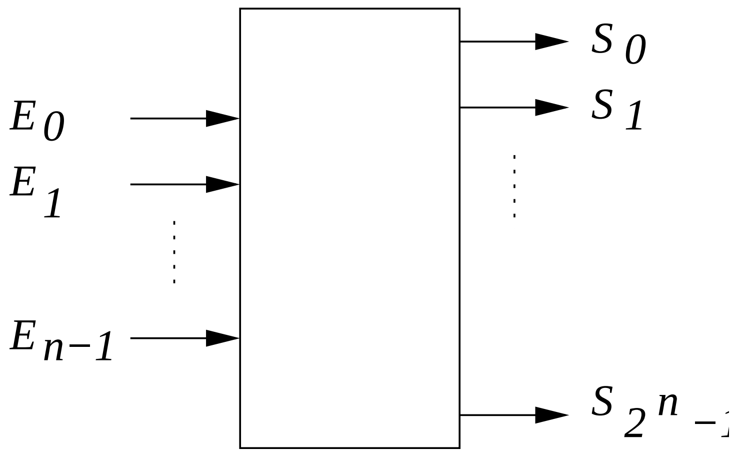
Décodeurs



Un décodeur est un circuit comportant :

- n entrées ;

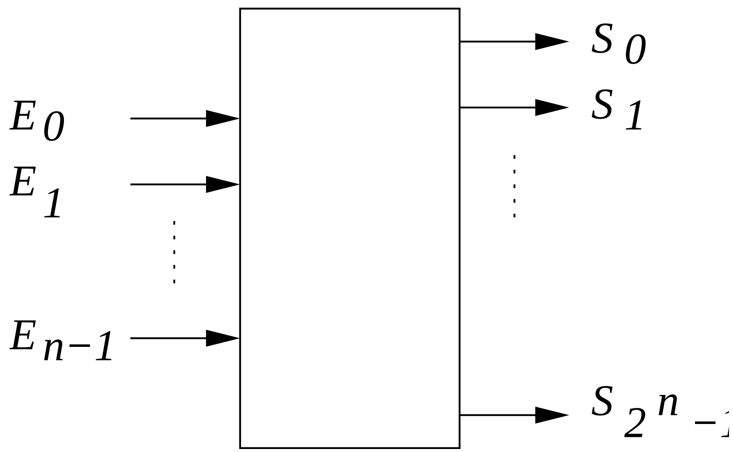
Décodeurs



Un décodeur est un circuit comportant :

- n entrées ;
- 2^n sorties, dont une toujours à 1 ;

Décodeurs



Un décodeur est un circuit comportant :

- n entrées ;
- 2^n sorties, dont une toujours à 1 ;
- telle qu'elle est celle dont le numéro est l'entier codé par la configuration binaire des entrées.

Décodeurs

Autrement dit : *Un décodeur permet de sélectionner une sortie* $S_i, 0 \leq i \leq 2^n - 1$, *avec* $i =$ *valeur*$_{10}(E_n, \ldots, E_0)$.

Exemple : Décodeur 3–8.

Soient $E_2 = 1, E_1 = 1, E_0 = 0$, c'est-à-dire 110. L'entier codé par $E_2E_1E_0$ est 6, par conséquent la sortie S_6 est sélectionnée, elle vaut 1.

Spécification d'un décodeur 2–4

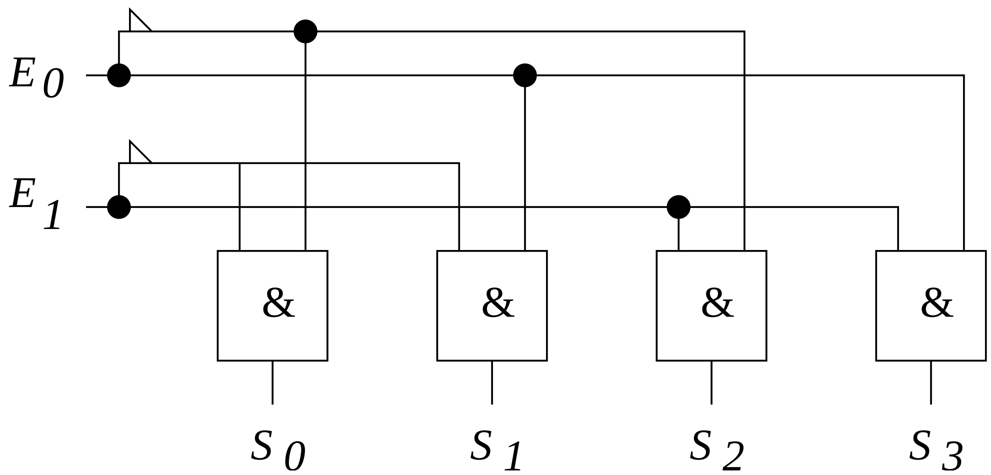
Deux entrées E_1 et E_0, quatre sorties S_3, S_2, S_1, S_0 telles que la sortie dont le numéro d'indice de la sortie est codé par la configuration binaire des entrées E_1E_0 est mise à 1.

E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

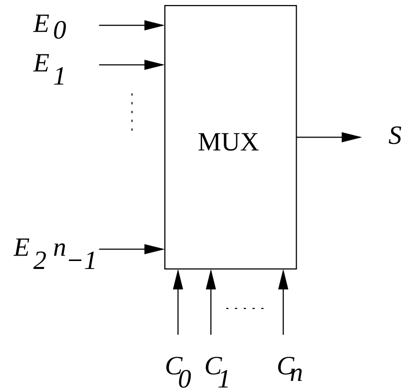
Equations booléennes des sorties :

$$S_0 = \overline{E_1}.\overline{E_0}, S_1 = \overline{E_1}.E_0, S_2 = E_1.\overline{E_0}, S_3 = E_1.E_0$$

Décodeur 2–4 : circuit



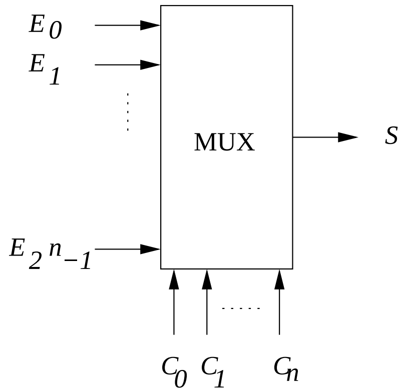
Multiplexeur



Un multiplexeur est un circuit comprenant :

- n fils de contrôle $C_n, C_{n-1}, \ldots, C_0$;

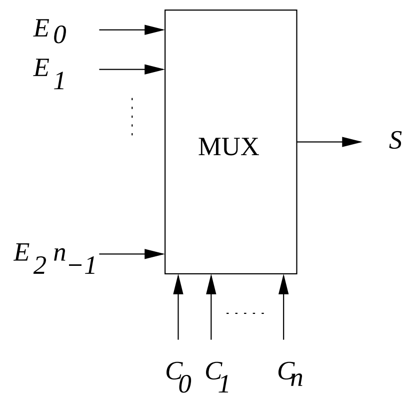
Multiplexeur



Un multiplexeur est un circuit comprenant :

- n fils de contrôle $C_n, C_{n-1}, \ldots, C_0$;
- 2^n entrées ;

Multiplexeur



Un multiplexeur est un circuit comprenant :

- n fils de contrôle $C_n, C_{n-1}, \ldots, C_0$;
- 2^n entrées ;
- Une sortie S qui prend la valeur de l'une des entrées, de telle sorte que la configuration binaire des n fils de contrôle code le numéro d'indice de cette entrée.

Multiplexeur

Autrement dit : *un multiplexeur permet de connecter une entrée $E_i, 0 \leq i \leq 2^n - 1$, à la sortie S de telle sorte que $i =$ valeur$_{10}(C_n, \ldots, C_0)$*.

Exemple : Multiplexeur 8–1.

Soient les trois variables de contrôle $C_2 = 1, C_1 = 1, C_0 = 0$, c'est-à-dire 110. L'entier codé par $C_2C_1C_0$ est 6, par conséquent la sortie S prend la valeur (1 ou 0) qui se trouve sur l'entrée sélectionnée E_6.

Spécification d'un multiplexeur 4–1

- Parce que les entrées sont mutuellement exclusives, il est possible de construire une table de vérité simplifiée, appelée *table de fonctionnement*, dont la colonne de sortie, plutôt que de contenir des 1 et des 0, contient les variables d'entrées.

Spécification d'un multiplexeur 4–1

- Parce que les entrées sont mutuellement exclusives, il est possible de construire une table de vérité simplifiée, appelée *table de fonctionnement*, dont la colonne de sortie, plutôt que de contenir des 1 et des 0, contient les variables d'entrées.
- La table en question contient donc deux variables de contrôle C_1 et C_0, et une sortie S qui prend la valeur de l'une des quatre entrées E_3, E_2, E_1, E_0 quand la configuration binaire C_1C_0 code le numéro d'indice de la dite entrée.

Spécification d'un multiplexeur 4–1

- Parce que les entrées sont mutuellement exclusives, il est possible de construire une table de vérité simplifiée, appelée *table de fonctionnement*, dont la colonne de sortie, plutôt que de contenir des 1 et des 0, contient les variables d'entrées.
- La table en question contient donc deux variables de contrôle C_1 et C_0, et une sortie S qui prend la valeur de l'une des quatre entrées E_3, E_2, E_1, E_0 quand la configuration binaire C_1C_0 code le numéro d'indice de la dite entrée.
- Il existe une table de vérité "classique" comportant une colonne assciée à E_0, une à E_1, une à E_2, et une à E_3, ainsi que les colonnes associées à C_1 et C_0 et la colonne de sortie S... Mais cette table "masque" la connexion de la sortie à l'une des entrées en fonction des variables de contrôle), et il est plus laborieux d'en extraire la fonction booléenne.

Spécification d'un multiplexeur 4–1

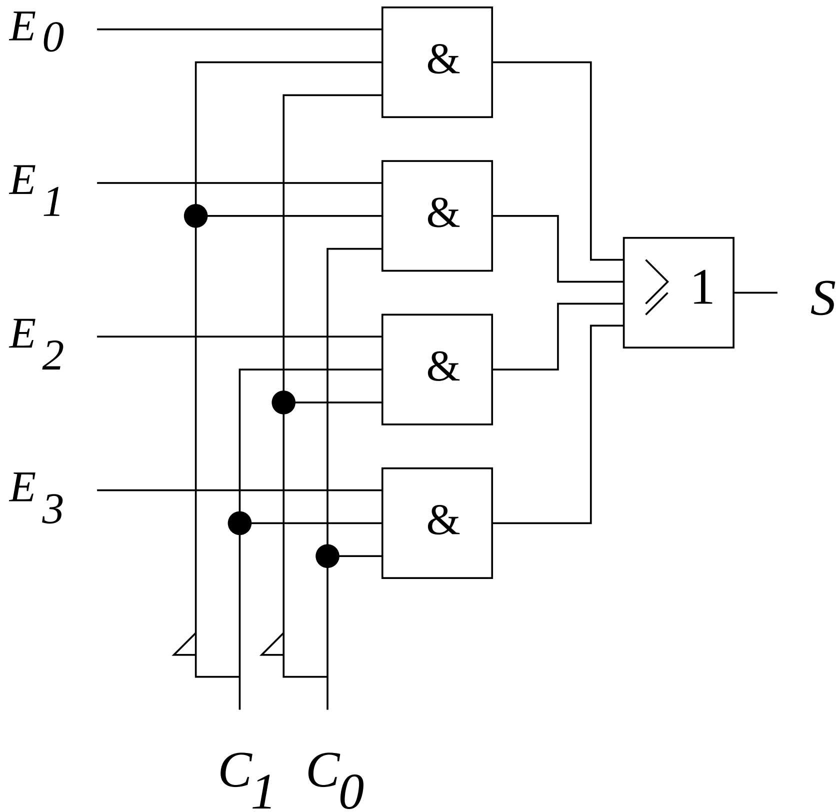
Table de fonctionnement :

C_1	C_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

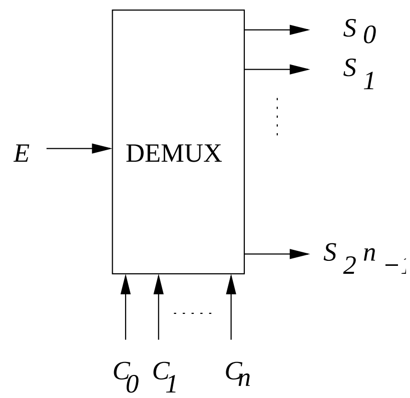
L'équation boléenne de sortie est donc :

$$S = \overline{C_1}.\overline{C_0}.E_0 + \overline{C_1}.C_0.E_1 + C_1.\overline{C_0}.E_2 + C_1.C_0.E_3$$

Multiplexeur 4–1 : circuit

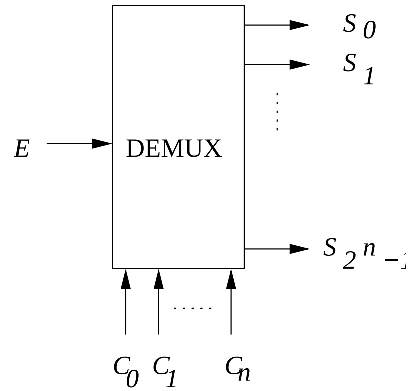


Démultiplexeur



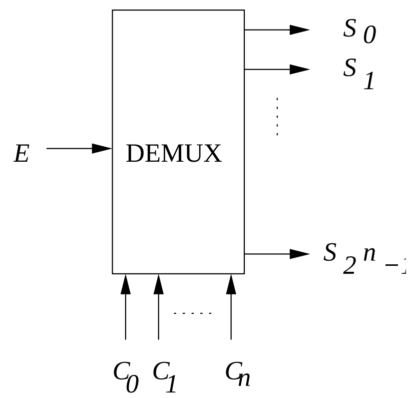
- n fils de contrôle $C_n, C_{n-1}, \ldots, C_0$;

Démultiplexeur



- n fils de contrôle $C_n, C_{n-1}, \ldots, C_0$;
- 2^n sorties $S_0, \ldots S_{2^n-1}$;

Démultiplexeur



- n fils de contrôle $C_n, C_{n-1}, \ldots, C_0$;
- 2^n sorties $S_0, \ldots S_{2^n-1}$;
- Une entrée E qui donne sa valeur à l'une des sorties, de telle sorte que la configuration binaire des n fils de contrôle code le numéro d'indice de cette sortie.

Démultiplexeur

Autrement dit : *un démultiplexeur permet d'aiguiller l'entrée E vers la sortie $S_i, 0 \leq i \leq 2^n - 1$, de telle sorte que $i =$ valeur$_{10}(C_n, \ldots, C_0)$*.

Exemple : Démultiplexeur 1–8.

Soient les trois variables de contrôle $C_2 = 1, C_1 = 1, C_0 = 0$, c'est-à-dire 110. L'entier codé par $C_2C_1C_0$ est 6, par conséquent la sortie S_6 est sélectionnée. Elle prend la valeur (1 ou 0) qui se trouve sur l'entrée E.

Spécification d'un démultiplexeur 1–4

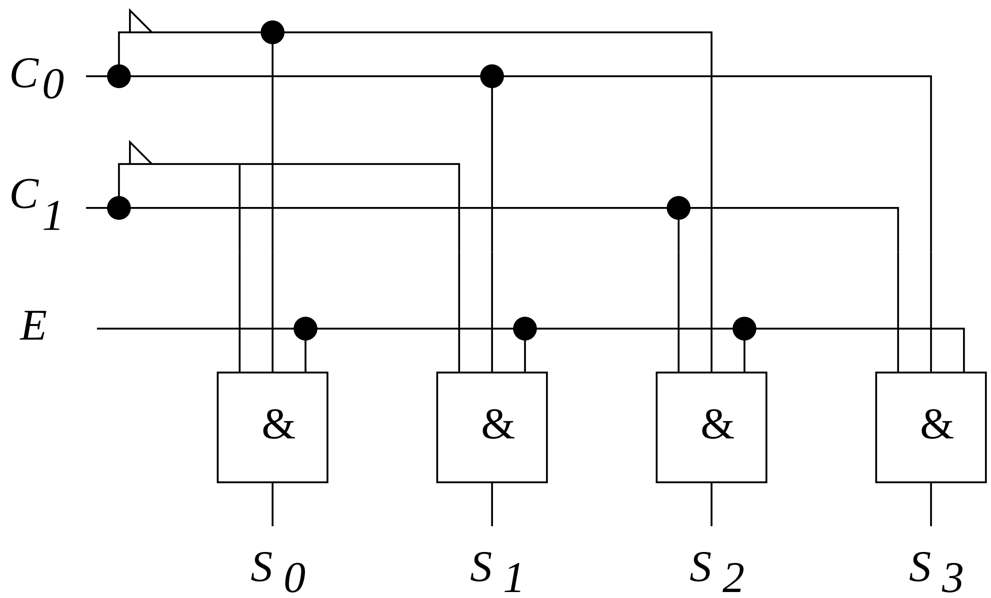
Table de fonctionnement :

C_1	C_0	S_0	S_1	S_2	S_3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

Les équations booléennes de sortie sont donc :

$S_0 = \overline{C_1}.\overline{C_0}.E, S_1 = \overline{C_1}.C_0.E, S_2 = C_1.\overline{C_0}.E, S_3 = C_1.C_0.E$

Démultiplexeur 1–4 : circuit



Demi-additionneur

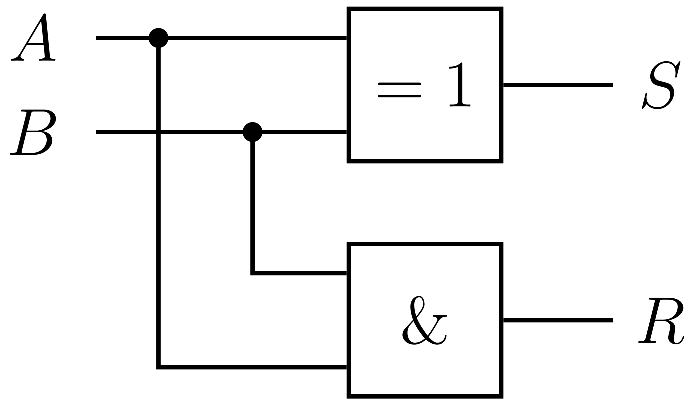
Circuit qui réalise l'addition de deux bits. La somme s'obtient sur deux bits, S pour le poids faible, R pour le poids fort (la retenue).

A	B	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\begin{aligned} S &= A.\overline{B} + \overline{A}.B \\ &= A \oplus B \end{aligned}$$

$$R = A.B$$

Demi-additionneur



Additionneur

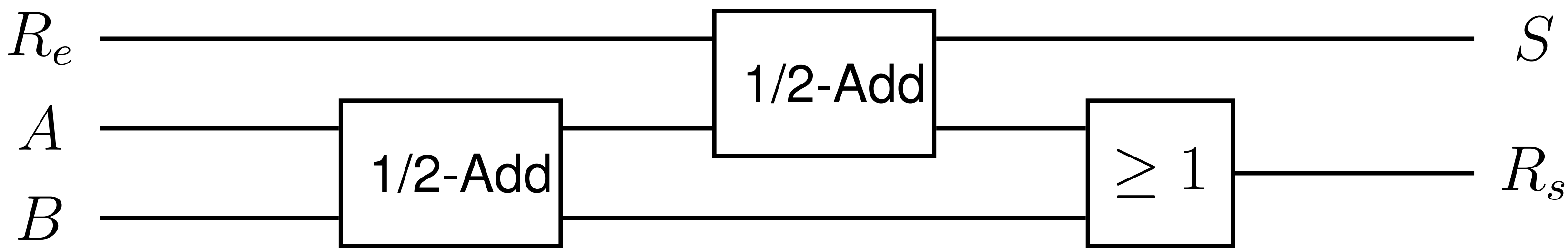
Circuit qui réalise la somme de deux bits en générant une retenue de sortie R_s et en tenant compte d'une retenue R_e en entrée.

A	B	R_e	R_s	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

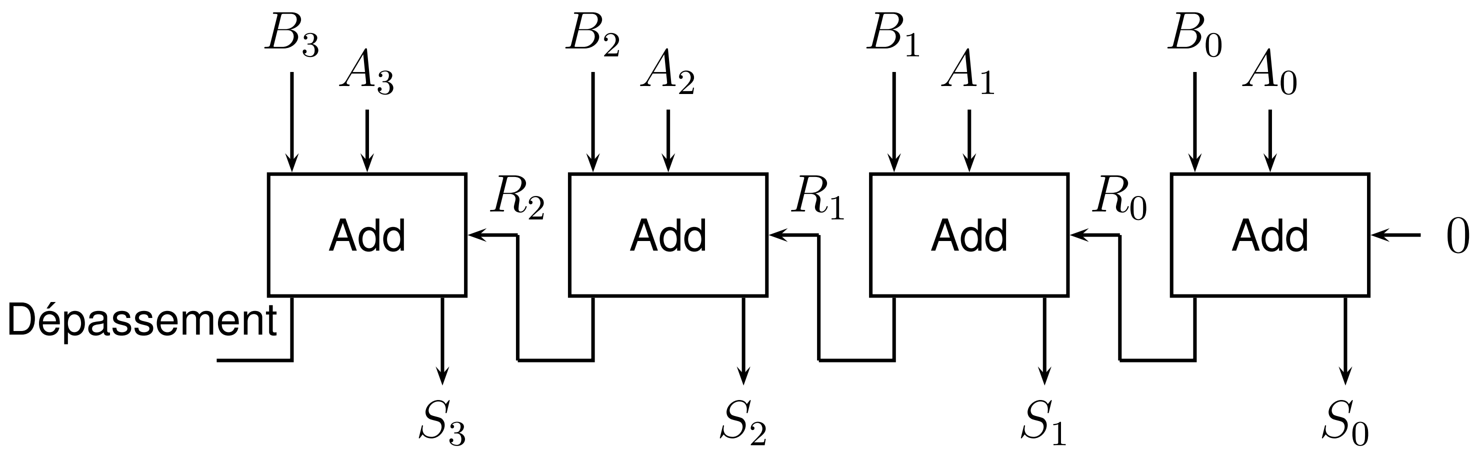
$$\begin{aligned}
S &= \overline{A}.\overline{B}.R_e + \overline{A}.B.\overline{R_e} + A.\overline{B}.\overline{R_e} + A.B.R_e \\
&= \overline{R_e}.(\overline{A}.B + A.\overline{B}) + R_e.(\overline{A}.\overline{B} + A.B) \\
&= \overline{R_e}.(A \oplus B) + R_e.(\overline{A \oplus B}) \\
&= R_e \oplus A \oplus B
\end{aligned}$$

$$\begin{aligned}
R_s &= \overline{A}.B.R_e + A.\overline{B}.R_e + A.B.\overline{R_e} + A.B.R_e \\
&= \underbrace{(A \oplus B).R_e}_{R_1} + \underbrace{A.B}_{R_2}
\end{aligned}$$

Additionneur



Additionneur 4 bits



Epilogue

Très schématiquement :

un ordinateur est un ensemble de circuits logiques dédiés (additionneurs, décodeurs, multiplexeurs. . .) reliés entre eux et ayant accés à des éléments de mémorisation.