Chapitre 1: Algèbre binaire

Pour manipuler les expressions binaires on utilise l'algèbre binaire comme outil mathématique. Cette algèbre est un cas particulier de l'algèbre de Boole.

1.1 Définition et axiomes

Une algèbre est un ensemble E muni de deux opérations T et T'.
L'algèbre de Boole est une algèbre vérifiant les axiomes suivants:

- T et T' : Lois de composition interne
- T et T' commutatives
- T et T' associatives
- T et T' distributives l'une par rapport à l'autre
- T et T' possèdent un élément neutre (**u** pour T et **n** pour T').
- Chaque élément x de E possède un élément symétrique (noté /x)

1.2 Théorèmes de l'algèbre de Boole

T1: Idempotence	$x \, T \, x = x$	;	$x \, T' \, x = x$
T2:	$x \, T \, \mathbf{n} = \mathbf{n}$	;	$x \, T' \, \mathbf{u} = \mathbf{u}$
T3: Absorption	$x \, T \, (x \, T' \, y) = x$	;	$x \, T' \, (x \, T \, y) = x$
T4: De Morgan	$\overline{x \, T \, y} = \overline{x} \, T' \, \overline{y}$	;	$\overline{x \, T' \, y} = \overline{x} \, T \, \overline{y}$
T5:	$\overline{\mathbf{u}} = \mathbf{n}$	;	$\overline{\mathbf{n}} = \mathbf{u}$
T6:	$\overline{\overline{x}} = x$		

1.3 Algèbre binaire

1.3.1 Définition

C'est une algèbre de Boole, dont l'ensemble E ne possède que deux éléments. Ces éléments ne peuvent être que **u** et **n**.
Une interprétation

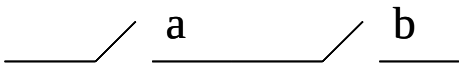
- L'ensemble E représente les états d'un interrupteur (fermé, ouvert).
- L'opération T entre deux variables représente la mise en série des interrupteurs correspondants.
- L'opération T' sur deux variables représente la mise en parallèle des interrupteurs correspondants

1.3.2 Opérateur T

–Appellation: ET, AND, conjonction, produit logique
–Notation: rien, . , & , ∧
On utilisera surtout le "." ou rien.
–Table des valeurs

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

–Circuit



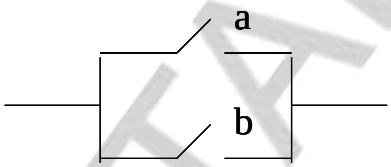
- Symboles

1.3.3 Opérateur T'

–Appellation: OU, OR, disjonction, somme logique
–Notation: +, v (On utilisera souvent "+")
–Table des valeurs

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

–Circuit



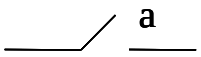
- Symboles

1.3.4 Opérateur OUI

–Appellation: Identité
–Expression: y = a
–Table des valeurs

a	y
0	0
1	1

–Circuit



Il s'agit d'un interrupteur normalement ouvert.

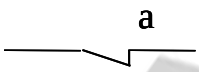
- Symboles

1.3.5 Opérateur NON

–Appellation: PAS, NOT, inversion, complémentation
–Expression: $y = \overline{a}$
–Table des valeurs

a	y
0	1
1	0

–Circuit



Il s'agit d'un interrupteur normalement fermé.

- Symboles

1.3.6 Opérateurs composés

a- OU exclusif

- ✓ Appellation: XOR, antivalence
- ✓ Expression: $a \oplus b = a\,\overline{b} + \overline{a}\,b$
- ✓ Table des valeurs

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

La sortie prend la valeur 1 uniquement lorsque les deux entrées sont différentes.

- ✓ Symboles

- ✓ Circuit

b- NOR exclusif

- ✓ Appellation: XNOR, équivalence
- ✓ Expression: $a \oplus b = a\ b + \overline{a}\ b$
- ✓ Table des valeurs

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

La sortie prend la valeur 1 uniquement lorsque les deux entrées sont égales.

- ✓ Symboles

- ✓ Circuit

c- NAND

- ✓ Appellation: NON ET, NOT AND
- ✓ Notation : $|$ $\overline{\quad}$
- ✓ Expression: $a \mid b = \overline{a\ b}$
- ✓ Table des valeurs

a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

- ✓ Symboles

- ✓ Circuit

d- NOR

- ✓ Appellation: NON OU, NOT OR
- ✓ Notation : ↓
- ✓ Expression: $a \downarrow b = \overline{a + b}$
- ✓ Table des valeurs

a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

- ✓ Symboles

- ✓ Circuit

1.3.7 Expressions binaires

Ce sont des expressions construites à base des opérateurs élémentaires et composés vus précédemment.

Exemple

$y = (a + b)(1 \oplus d) + /c\,(a + /d + /e)$

Règle de priorité

Priorité de AND devant OR.

1.3.8 Complémentation

–Règle de complémentation

1	0	x	$\overline{x}$	.	+	≡	⊕
0	1	$\overline{x}$	x	+	.	⊕	≡

–Règle de Shannon

En appliquant les règles de complémentation à une expression, on obtient son expression complémentaire (Règle de De Morgan généralisée).

–Exemple

Soit $y = (a + b)(1 \oplus d) + /c\,(a + /d + /e)$; Chercher $/y$ en appliquant la règle de Shannon.

Il faut commencer par mettre des parenthèses pour les produits figurant dans cette expression.

$$y = [(a + b)(1 \oplus d)] + [/c\,(a + /d + /e)]$$

On obtient :

$$/y = [/a./b + (0 \equiv /d)] \,.\, [\, c + (/a \,.\, d \,.\, e)]$$

1.3.9 Résumé des axiomes et théorèmes de l'algèbre binaire

–Commutativité

$a \,.\, b = b \,.\, a$

$a + b = b + a$

–Associativité

$a \,.\, (b \,.\, c) = (a \,.\, b) \,.\, c$

$a + (b + c) = (a + b) + c$

–Distributivité

$a \,.\, (b + c) = a \,.\, b + a \,.\, c$

$a + b \,.\, c = (a + b) \,.\, (a + c)$

–Éléments neutres

$a \,.\, 1 = a$

$a + 0 = a$

$a \,.\, /a = 0$

$a + /a = 1$

$a \,.\, 0 = 0$

$a + 1 = 1$

–Idempotence

$a \,.\, a = a$

$a + a = a$

–Absorption

$a \,.\, (a + b) = a$

$a + (a \,.\, b) = a$

–Adsorption

$a \,.\, (/a + b) = a.\, b$

$a + (/a \,.\, b) = a + b$

–De Morgan

$/(a \,.\, b) = /a + /b$

$/(a + b) = /a \,.\, /b$

Ces règles sont utiles pour manipuler les expressions et les fonctions binaires.

Chapitre 2: Représentation des nombres

2.1 Généralités

- ✓ Pour quantifier un objet quelconque on utilise des symboles appelés nombres.
- ✓ Pour écrire ces nombres on utilise un système de numération.
- ✓ Un système de numération est caractérisé par des symboles appelés chiffres.
- ✓ La quantité des chiffres correspond à la base de numération.
- ✓ Le nombre est représenté alors par une série de chiffres. A chaque chiffre correspond un poids qui dépend de la position de ce chiffre.

Exemple

Soit une base b contenant les chiffres : $s_0, s_1, s_2, \dots, s_{b-1}$

Un nombre N s'écrit dans la base b comme suit:

$$N = (a_n \dots a_1 a_0 , a_{-1} a_{-2} \dots a_{-m})_b \qquad \text{Avec } a_i \in \{s_0, s_1, s_2, \dots, s_{b-1}\}$$

La signification de cette écriture est:

$$N = a_n b^n + \dots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \dots + a_{-m} b^{-m}$$

Remarques

Cette écriture s'appelle forme polynomiale.

Les chiffres avant la virgule forment la partie entière, ceux après la virgule forment la partie fractionnaire de N.

2.2 Les principales bases

- La base 2 ou système de numération binaire

La base est b = 2. Les chiffres sont 0 et 1 et s'appellent bits ou chiffres binaires.

Un nombre N s'écrit dans la base 2 sous forme polynomiale:

$$N = a_n 2^n + \dots + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + a_{-2} 2^{-2} + \dots + a_{-m} 2^{-m} \qquad \text{Avec } a_i \in \{0 , 1\}$$

a_n s'appelle MSB. a_{-m} s'appelle LSB.

Exemple

Soit $N_2 = 1001{,}011$; trouver N_{10}.

Utilité

C'est la base utilisée dans les circuits électroniques numériques.

- La base 8 ou système de numération octale

La base est b = 8. Les chiffres sont 0,1,2,3,4,5,6,7.

Un nombre N s'écrit dans la base 8 sous forme polynomiale:

$$N = a_n 8^n + \dots + a_1 8^1 + a_0 8^0 + a_{-1} 8^{-1} + a_{-2} 8^{-2} + \dots + a_{-m} 8^{-m} \qquad \text{Avec } a_i \in \{0,1,2,3,4,5,6,7\}$$

Exemple

Soit $N_8 = 52{,}02$; trouver N_{10}.

Utilité

Cette base est utile pour compresser l'écriture de la base 2.

Attention

Les chiffres 8 et 9 n'ont pas de sens ici.

- La base 10 ou système de numération décimale

La base est b = 10. Les chiffres sont 0,1,2,3,4,5,6,7,8,9.

Un nombre N s'écrit dans la base 10 sous forme polynomiale:

$$N = a_n 10^n + \dots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + a_{-2} 10^{-2} + \dots + a_{-m} 10^{-m}$$

$$\text{Avec } a_i \in \{0,1,2,3,4,5,6,7,8,9\}$$

Utilité

Ce système de numération est utilisé dans la vie courante.

- La base 16 ou système de numération hexadécimale

La base est b = 16. Les chiffres sont: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
Un nombre N s'écrit dans la base 16 sous forme polynomiale:

$$N = a_n 16^n + \ldots + a_1 16^1 + a_0 16^0 + a_{-1} 16^{-1} + a_{-2} 16^{-2} + \ldots + a_{-m} 16^{-m}$$

Avec $a_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Exemple

Soit $N_{16} = 24{,}01$; trouver N_{10}.

Utilité

Cette base est utile pour compresser l'écriture de la base 2.

2.3 Passage d'une base à une autre

2.3.1 Passage d'une base b à la base 10

On applique directement l'écriture polynomiale.

Exemple: (Voir les exemples précédents)

2.3.2 Passage de la base 10 à une base b

On traite les parties entière (PE) et fractionnaire (PF) séparément.

- Partie entière

On divise le nombre décimal successivement par b. Les restes obtenus forment les chiffres de l'écriture de ce nombre dans la base b.

Exemples

- Soit $N_{10} = 86$; trouver N_7.
- Soit $N_{10} = 183$; trouver N_{13}.

- Partie fractionnaire

On la multiplie successivement par b. Les chiffres obtenus à gauche de la virgule forment le nombre dans la base b.

Exemples

- Soit $N_{10} = 0{,}47$; trouver N_7.
- Soit $N_{10} = 0{,}8$; trouver N_{13}.

- Nombre quelconque

On juxtapose la partie entière et la partie fractionnaire:

$N_b = PE_b , PF_b$

Exemples

- Soit $N_{10} = 86{,}47$; trouver N_7.
- Soit $N_{10} = 183{,}8$; trouver N_{13}.

2.3.3 Passage d'une base b à une base b'

- Règle générale

On divise la partie entière et on multiplie la partie fractionnaire successivement par b'.

- Inconvénient: Il faut faire les opérations dans le système de numération b (difficile).
- Mieux

On passe par la base 10.

$$\begin{tikzcd} N_b \arrow[rr] \arrow[dr] & & N_{b'} \\ & N_{10} \arrow[ur] & \end{tikzcd}$$

- Exceptions

Pour passer de la base 2 aux bases 8 et 16 ou inversement, il y a une méthode plus simple.

2.3.4 Passage de la base 2 à la base 8

- Règle

On regroupe les bits par 3 à partir de la virgule. On écrit ensuite chaque groupe en décimal.

- Exemples

N_2 = 1100011,1001101 ; Trouver N_8.

N_2 = 1011010101,11 ; Trouver N_8.

2.3.5 Passage de la base 2 à la base 16

- Règle

On regroupe les bits par 4 à partir de la virgule. On écrit ensuite chaque groupe en hexadécimal.

- Exemples

N_2 = 1100011,1001101 ; Trouver N_{16}.

N_2 = 1011010101,11 ; Trouver N_{16}.

2.3.6 Passage de la base 8 à la base 2

- Règle

On éclate les chiffres octaux en trois bits. Les zéros de la partie entière figurant complètement à gauche et ceux de la partie fractionnaire figurant complètement à droite peuvent être éliminés.

- Exemple

N_8 = 273,154 ; Trouver N_2.

2.3.7 Passage de la base 16 à la base 2

- Règle

On éclate les chiffres héxadécimaux en quatre bits.

- Exemple

N_{16} = 2B,1C ; Trouver N_2.

2.4 Représentation des nombres signés

Il y a deux possibilités.

2.4.1 Représentation module plus signe

- Règle

On ajoute un bit de signe S: 0 pour le signe + , 1 pour le signe -

Forme générale

Signe	*Module*		
S	MSB	……..	LSB

Exemples

N_{10} = +18 ; Trouver N_2

N_{10} = - 53 ; Trouver N_2

- Nombre maximal et nombre minimal

Si on utilise (n+1) bits (n pour le module et 1 pour le signe), on a alors :

$N_{max} = + (2^n-1)$

$N_{min} = - (2^n-1)$

- Exemple : n = 3 ; $-7 \leq N \leq +7$

Nombres positifs:			Nombre négatifs:		
0111	,	7	1111	,	-7
0110	,	6	1110	,	-6
0101	,	5	1101	,	-5
0100	,	4	1100	,	-4
0011	,	3	1011	,	-3
0010	,	2	1010	,	-2
0001	,	1	1001	,	-1
0000	,	0	1000	,	-0

- Avantage
  - Simplicité
- Inconvénients
  - Le zéro possède deux représentations.
  - Elle ne convient pas pour les opérations arithmétiques.

    Exemple

    (- 1) + (+3) = +2

    (1001) + (0011) = 1100 ($\Rightarrow$ - 4 $\neq$ +2)

2.4.2 Représentation avec le complément

- Complément à 1 d'un nombre N

Il est obtenu en inversant chaque bit de N. On le note: $\overline{N}$

Exemple

N = 100110 ; $\overline{N}$ = 011001

- Complément à 2 d'un nombre N

Il correspond à $\overline{N}$ + 1.

Exemple: N = 100110 ; $\overline{N}$ + 1 = 011010

- Règle

On a: (- N) = $\overline{N}$ + 1

On utilise cette règle pour représenter les nombres signés.

- Exemple: n = 3 (signe non compris) ;

Nombres positifs:			Nombre négatifs:		
0111	,	7		,	-7
0110	,	6		,	-6
0101	,	5		,	-5
0100	,	4		,	-4
0011	,	3	1101	,	-3
0010	,	2	1110	,	-2
0001	,	1	1111	,	-1
0000	,	0	0000	,	-0

Le nombre 1000 existe aussi. Il correspond à (– 8).

Car: (-8) + 1 = -7

1000 + 0001 = 1001

- Nombre maximal et nombre minimal

Si on utilise (n+1) bits (n pour le module et 1 pour le signe),

N_{max} = + (2^n-1)

N_{min} = - (2^n)

- Remarques
  - On a: -(-N) = N (pour N $\neq$ - 2^n)
  - Cette représentation est souvent utilisée, car elle facilite le calcul arithmétique.

2.5 Représentation des nombres à virgule

On distingue 2 représentations

• Représentation en virgule fixe

• Représentation en virgule flottante

2.5.1 Représentation en virgule fixe

La virgule garde une position fixe.

On suppose que le nombre a n chiffres et que la virgule se trouve au rang k (après k chiffres à partir de la droite).

$$N = a_{n-1-k} \dots a_0 , a_{-1} \dots a_{-k}$$

Cette représentation convient seulement pour l'addition.

2.5.2 Représentation en virgule flottante

La virgule est systématiquement placée à gauche du nombre.

Celui-ci s'écrit:

$$0,\dots\dots \times 10^{\cdots}$$

Dans une base b, le nombre N s'écrit:

$$N = 0,\boldsymbol{M} \times b^{\boldsymbol{E}}$$

$M = M_1 M_2 \dots M_m$ s'appelle mantisse.

$E = E_{p-1} \dots E_1 E_0$ s'appelle exposant.

M et E sont des entiers positifs ou négatifs.

Dans la représentation normalisée, le chiffre à droite de la virgule est différent de 0.

Exemples

M = 0,3521 : normalisée

M = 0,003521 : non normalisée

M = 46,521 : non normalisée

Le plus grand nombre positif

$N_{max} = +0,11\dots1 \times 2^{(+11\dots1)}$

$N_{max} = 2\,(\exp(2\exp(p)-1))$

Le plus petit nombre positif

$N_{min} = +0,10\dots0 \times 2^{(-11\dots1)}$

$N_{min} = 2\,(\exp(-2\exp(p))$

Exemple: p = 7

$N_{max} = 1,7 \,.\, 10^{38}$;

$N_{min} = 3 \,.\, 10^{-39}$

2.6 Opérations en binaire

2.6.1 Addition

▪ Demi-additionneur (HA)

Il fait l'addition de deux nombres à 1 bit (1BHA) et génère deux bits qui représentent la somme et la retenue (carry).

Schéma bloc du 1BHA

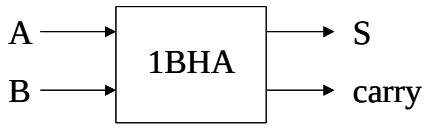


Table décrivant la fonction du 1BHA

A	B	S	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Additionneur complet (FA)

Il tient compte de la retenue (carry_in) provenant de la somme des chiffres de la colonne précédente.

Schéma bloc du FA à 1 bit

Table décrivant la fonction du 1BFA

A	B	c_in	S	c_out
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

- Addition de deux nombres quelconques

$S = A + B$; $A = a_{n-1}\ldots a_1 a_0$; $B = b_{n-1}\ldots b_1 b_0$

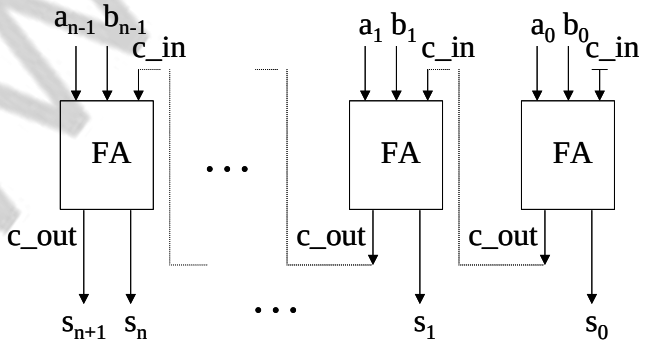
Méthode

A et B sont représentés en virgule fixe. La méthode est semblable à celle du système décimal.

Exemple: 1011101 + 110010,101

Circuit

Il faut mettre n FA en cascade.



2.6.2 Soustraction

- Demi-soustracteur (HS)

Il fait la soustraction de deux nombres à 1bit (1BHS) et génère deux bits qui représentent la somme et la retenue (carry).

Schéma bloc du HS à 1 bit

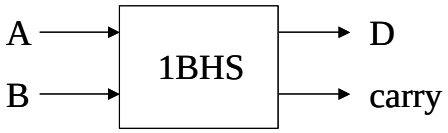


Table décrivant la fonction du 1BHS

A	B	D	carry
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- Soustracteur complet (FS)

Il tient compte de la retenue (carry_in) provenant de la soustraction des chiffres de la colonne précédente.

Schéma bloc du FS à 1 bit

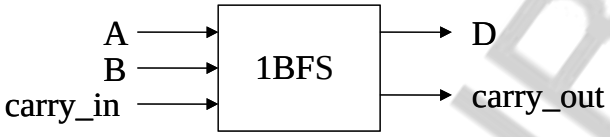


Table décrivant la fonction du 1BFS

A	B	c_in	D	c_out
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- Soustraction de deux nombres quelconques

$S = A - B$; $A = a_{n-1}\ldots a_1a_0$ et $B = b_{n-1}\ldots b_1b_0$

A et B sont représentés en virgule fixe. Ils peuvent être des nombres fractionnaires et/ou signés.

1er cas: Représentation non signée

✓Méthode I

On traite les nombres A et B bit par bit en appliquant pour chaque bit la fonction du HS (comme dans la base 10).

Exemples

Calculer:

101101,101 – 10,1101

10100 - 110101

✓Méthode II

On ramène l'opération de soustraction à une opération d'addition. B représente le complément à 1 de B.

Règle

1er cas: A > B ; Le résultat est positif.

On calcule A + $\overline{B}$. Il y a un 1 qui déborde à gauche. Ce 1 on l'ajoute à la somme obtenue:

$$A - B = A + \overline{B} + 1$$

2ème cas: A < B ; Le résultat est négatif.

On calcule A + $\overline{B}$. Il n'y a pas de 1 qui déborde à gauche. Il faut alors complémenter à 1 la somme obtenue:

$$A - B = \overline{A + \overline{B}}$$

Exemples

Calculer:

101101,101 – 10,1101

10100 – 110101

2ème cas: Représentation signée en complément à 2

Méthode

On a A – B = A + (-B)

On ajoute alors à A le complément à 2 de B.

Exemples

Calculer: (pour n = 8 avec signe)

101101,101 – 10,1101

10100 – 110101

Remarques

- Dans les circuits on utilise souvent la dernière méthode, car elle permet d'utiliser le FA pour l'addition et la soustraction.
- Il y a des incertitudes dues aux erreurs de conversion et aux troncatures.

  Exemple

  0,4 = 0,011001100110...

  Avec une précision de 3 chiffres, on a:

  0,4 ≈ 0,011 ; or 0,011 = 0,375
- Si on dépasse l'intervalle de définition des nombres, on obtient des résultats non logiques (overflow).

  Exemple

  On suppose que n = 4 (avec signe).

  7 + 3 = 0111 + 0011 = 1010 = -6 !!!

  La cause est le fait que 7 + 3 = +10 nécessite 5 chiffres pour sa représentation signée.
- Pour détecter un overflow, on compare le signe des opérandes avec celui du résultat.
- Il faut toujours savoir le nombre maximal de bits pour les nombres utilisés.

2.6.3 Autres opérations

Toutes les autres opérations (x, /, sin, log,…) peuvent être ramenées à une suite d'opérations d'addition.

Remarque

Pour x et / on utilise généralement la représentation en virgule flottante.

2.7 Codes binaires

Il y a deux représentations possibles d'une grandeur: Représentation analogique et représentation digitale (numérique).

Pour représenter numériquement un nombre on utilise un code.
Coder (ou encoder) un nombre ou symbole en binaire consiste à lui faire correspondre une suite de 0 et 1 appelée code. Une suite de 0 et 1 s'appelle aussi combinaison binaire.
Combinaison binaire à 4 bits = nibble
Combinaison binaire à 8 bits = byte ou octet
Combinaison binaire à 16 bits = word ou mot
Combinaison binaire à n bits = string ou chaîne

2.7.1 Code binaire pur

Appelé aussi code binaire naturel ou code binaire positionnel. Il correspond à l'écriture du nombre à coder en base 2.
Exemple: 49 ⟶ 110001

2.7.2 Codes BCD (BCD = décimal codé binaire)

On garde l'écriture décimale, mais on code chaque chiffre en binaire.

2.7.2.1 Codes BCD à 4 bits

Chaque chiffre X du nombre donné est codé sur 4 bits.

$X = x_3x_2x_1x_0$

- Codes pondérés

La valeur du chiffre X est donnée par:

$X = x_3A + x_2B + x_1C + x_0 D$ A, B, C, D s'appellent poids du code.

Le code est dit pondéré.
Exemple
Si on prend le code binaire pur, les poids sont: 8,4,2,1. On dit qu'on a un code pondéré 8421.
Autres codes pondérés: 7421, 2421, 4311, 5121

- Codes non pondérés

Le code le plus connu: Code à excès de 3 (Excess 3)

Chiffre décimal	Code à excès de 3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Avantage
Le complément d'un chiffre décimal X par rapport à 9 est obtenu en inversant chaque bit du chiffre : $C_9(X) = C_1(x3)\ C_1(x2)\ C_1(x1)\ C_1(x0)$

2.7.2.2 Codes BCD à 5 bits

Le code le plus connu: Code 2 sur 5

Chiffre décimal	Code 2 sur 5
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

2.7.3 Codes à distance unité

Quand on passe d'une combinaison à la combinaison suivante, le code ne change que d'un bit.
Règle de construction: (Voir cours)
Exemple : Le code le plus connu: Code de Gray

Nombre décimal	Code de Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

2.7.4 Autres codes

Code ASCII: C'est un code BCD sur 8 bits. (Voir cours)
Code 7-segments : (Voir cours)

2.7.5 Utilité des différents codes (Voir cours)

2.8 Représentation physique des signaux binaires

Les signaux binaires correspondent physiquement à deux états d'une tension.

Faible tension (Niveau bas) ⟶ 0 Tension élevée (Niveau haut) ⟶ 1

Exemples: Technologie TTL (Référence 74LS…):

Niveau bas: 0V … 0,8V Niveau haut: 2,5V … 5V

Pour avoir ces deux niveaux on a besoin d'un interrupteur électronique: C'est le transistor.

Chapitre 3 : Fonctions binaires

3.1 Rappels

Les combinaisons binaires de n bits sont les éléments de l'ensemble $\{0,1\}^n$. Si on a n bits, il y a alors 2^n combinaisons binaires différentes.

a- Définition d'une fonction binaire

Une fonction binaire à n variables associe aux combinaisons binaires de n bits la valeur 0 ou 1. Les n bits s'appellent variables binaires. C'est donc une application de l'ensemble $\{0,1\}^n$ sur l'ensemble {0,1}.

Remarque

L'ensemble des fonctions binaires muni des opérations . et + est une algèbre de Boole.

$f.g(x) = f(x).g(x)$; $f+g(x) = f(x) + g(x)$;

On peut donc appliquer les axiomes et les théorèmes de l'algèbre de Boole pour ces fonctions.

b- Représentation des combinaisons binaires

On utilise la représentation octale.

Exemple

X = (0,1,0,1,0,0,1,1,1,0,1,0,1) = 5165

On écrit le nombre octal comme indice de X : X_{5165}

Soit par exemple : f(0,1,0,1,0,0,1,1,1,0,1,0,1) = 1 ; on écrit alors : $f(X_{5165}) = 1$

Remarque

On ordonne les combinaisons binaires suivant l'ordre croissant de l'indice.

3.2 Représentation des fonctions binaires

Il y a plusieurs façons de représenter une fonction binaire.

3.2.1 Table de vérité

Elle possède deux colonnes, une pour les combinaisons binaires (entrées) et une pour la fonction binaire (sortie). La table de vérité d'une fonction à n variables possède 2^n lignes.

	Xj	f(Xj)
j =	0	…
	1	…
	2	…
	3	…
	…	…
	…	…
	2^n-1	…

$X = (x_n, \ldots, x_i, \ldots, x_2, x_1)$

Exemple

j	x_3	x_2	x_1	$y=g(X_j)$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

3.2.2 Ensembles de combinaisons

Définitions

- Un point nul d'une fonction f est une combinaison X_i telle que $f(X_i) = 0$.
- Un point non nul d'une fonction f est une combinaison X_i telle que $f(X_i) = 1$.
- Z(f) est l'ensemble des points nuls de f ; $Z(f) = \{...,Z_i,...,Z_k,...\}$
- U(f) est l'ensemble des points non nuls de f ; $U(f) = \{...,U_l,...,U_m,...\}$

Les indices de Z_i et U_i représentent le code octal des combinaisons correspondantes.

Représentation

Pour représenter une fonction f, il suffit de spécifier U(f) ou Z(f).

Exemple

Pour y = g(X) de la page 1, on a : $Z(g) = \{0,1,2,4,7\}$; $U(g) = \{3,5,6\}$

3.2.3 Vecteur caractéristique

Il représente le vecteur binaire associé à la dernière colonne de la table de vérité.

Exemple

Pour y = g(X) de la page 1, on a : g = 150 ; Autres écritures: g_{150} ; g^{150}

3.2.4 Expression booléenne

La fonction est décrite sous forme d'une expression logique.

y = f(X) = E E: Expression booléenne

Exemple

y = f(d,c,b,a) = (d + /c a + /d a) b + /a d (/x signifie l'inverse de x, c'est-à-dire $\overline{x}$)

3.2.5 Tables de Karnaugh

On les appelle aussi tableaux de Karnaugh.

a- Construction des tables (Voir cours)

b- Représentation de f

On porte dans la case j la valeur de $f(X_j)$.

c- Symétrie

Deux cases sont symétriques, si elles le sont par rapport à la ligne de degré de symétrie maximal se trouvant entre les deux cases.

Les combinaisons binaires relatives à deux cases symétriques ne diffèrent que dans une seule position.

3.2.6 Logigrammes

C'est un circuit à base de portes logiques représentant la fonction binaire donnée (voir TP).

3.2.7 Utilité des différentes représentations des fonctions binaires

Table de vérité

Elle permet de passer du cahier de charges à une description formelle de la fonction logique.

Ensembles de combinaisons et vecteur caractéristique

Ces deux possibilités décrivent la table de vérité de manière compacte.

Table de Karnaugh

Elle sert à simplifier les fonctions binaires.

Expression booléenne

Elle permet de tracer le logigramme.

Logigramme

Il sert pour la réalisation de la fonction.

3.3 Théorèmes généraux

3.3.1 Théorème fondamental de l'algèbre des fonctions binaires

3.3.1.1 Définitions

On considère une fonction à n variables $y = f(x_n, \dots, x_2, x_1)$.

a- Minterme

C'est un produit logique contenant toutes les variables de la fonction. On l'appelle aussi produit fondamental.

Exemple

$y = f(d,c,b,a)$

d /c b a est un minterme.

/d /b a n'est pas un minterme.

b- Maxterme

C'est une somme logique contenant toutes les variables de la fonction. On l'appelle aussi somme fondamentale.

Exemple

$y = f(d,c,b,a)$

/d+c+b+/a est un maxterme.

d+/c a n'est pas un maxterme.

c- Remarques

- Un minterme est = 1 pour une seule combinaison X_i.
  On désigne un minterme par m_i, i étant l'indice de la combinaison X_i qui rend le minterme =1.
- Un maxterme est = 0 pour une seule combinaison X_i.
  On désigne un maxterme par M_j, j étant l'indice de la combinaison X_i qui annule le minterme.

Exemples

(Voir cours)

z = f(d,c,b,a) ; /d+c+b+/a = ... ; d /c b a = ... ;

Mintermes	Maxtermes	x_3	x_2	x_1	$y=g(X_j)$
		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	0

3.3.1.2 Théorème fondamental

a- Forme disjonctive ou (SOP)

Toute fonction binaire peut s'écrire sous forme de la somme de tous les mintermes correspondant aux 1 de cette fonction.

$$\boxed{y = \sum_{\forall i \mid f(X_i)=1} m_i}$$ (1ère forme canonique)

b- Forme conjonctive ou (POS)

Toute fonction binaire peut s'écrire sous forme du produit de tous les maxtermes correspondant aux 0 de cette fonction.

$$\boxed{y = \prod_{\forall\, i \mid f(X_i)=0} M_i}$$ (2ème forme canonique)

Intérêt (Voir cours)
Exemple (Voir cours)

3.3.2 Théorème de Shannon

On peut développer une fonction binaire f par rapport à une de ses variables x_i:

$$\begin{aligned} f(x_n,\ldots,x_i,\ldots,x_1) &= x_i \,.\, f(x_n,\ldots,1,\ldots,x_1) + \\ &\quad \bar{x}_i \,.\, f(x_n,\ldots,0,\ldots,x_1) \\ &= x_i \,.\, f\big|_{x_i=1} + \bar{x}_i \,.\, f\big|_{x_i=0} \end{aligned}$$

Remarque
Si on développe f par rapport à toutes ses variables indépendantes x_i, on obtient la 1ère forme canonique.
Intérêt (Voir cours)
Exemple
Développer la fonction suivante par rapport à la variable c :
y = d /c + c b /a + /b
(Voir cours)

3.4 Fonctions incomplètement spécifiées

Parfois une fonction n'est pas spécifiée pour certaines combinaisons X_i. Ceci est dû à:

- Ces Xi ne se présentent jamais.
- L'action du système sous Xi est indifférente ou inhibée.

Exemple
(Voir cours)

a- Définitions

- Une fonction binaire à n variables est incomplètement spécifiées, si son ensemble de définition est un sous-ensemble se $\{0,1\}^n$.
- Les combinaisons X_i pour lesquelles la fonction n'est pas définie s'appellent redondances (R).

b- Représentation
Comme les fonctions complètement spécifiées. On écrit "X" pour les redondances de f.
Exemple (Voir cours)

3.5 Opérateurs de bases des fonctions binaires

Toute fonction binaire peut s'exprimer à l'aide des groupes d'opérateurs suivants:

- NOT, AND, OR (à cause du théorème fondamental)
- NOT, AND
- NOT, OR
- AND, XOR
- OR, XOR
- NAND
- NOR

3.6 Blocs de combinaisons d'une fonction

3.6.1 Définitions et représentation vectorielle

Un bloc (ou cube) B est un ensemble de combinaisons X_i des variables de la fonction.

Exemple

Soit y = f(x_5, x_4, x_3, x_2, x_1)

$$\left.\begin{matrix}(0,1,0,0,1)\\(0,1,1,0,1)\\(1,1,0,0,1)\\(1,1,1,0,1)\end{matrix}\right\} \Rightarrow B = (-,1,-,0,1) = (x_5, x_4, x_3, x_2, x_1)$$

- Les tirets signifient ici *'tous les cas possibles'*.
- x_5 et x_3 : Variables libres du bloc.
- x_4,x_2,x_1 : Variables fixes du bloc.
- (-,1,-,0,1) : Ecriture vectorielle de B.

Règle

Un bloc à r variables libres contient 2^r combinaisons.

Remarque

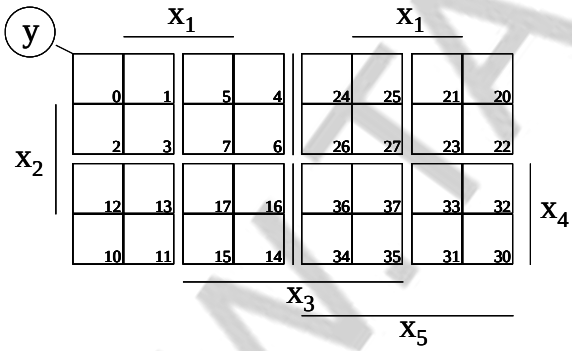
Si $\forall X_i \in B$, on a:

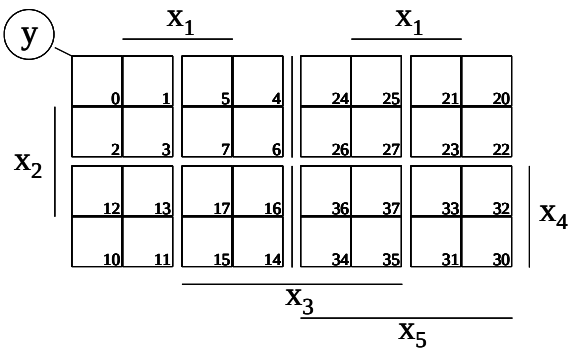
- $f(X_i) = 0$, B est dit bloc nul de la fonction f.
- $f(X_i) = 1$, B est dit bloc non nul de f.

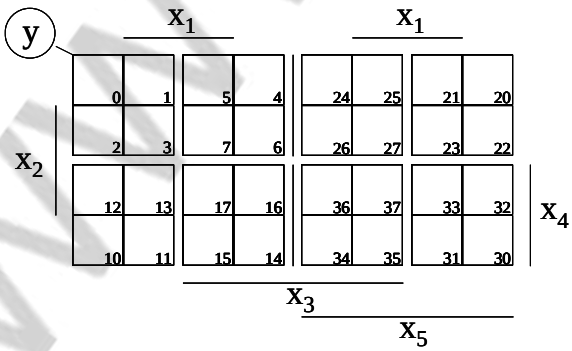
Signification graphique d'un bloc

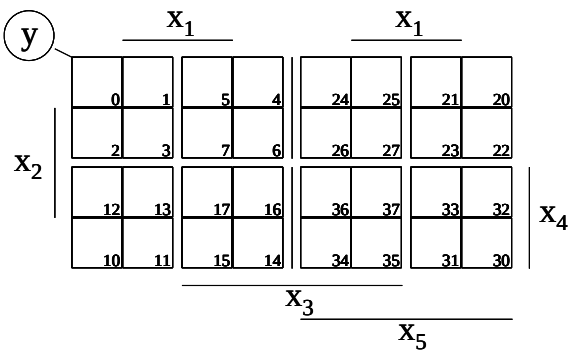
Dans la table de Karnaugh, un bloc est un ensemble de cases qui sont symétriques par rapport à l'axe le plus fort qui passe entre ces cases. Cette symétrie concerne aussi bien la direction horizontale que la direction verticale.

Exemples









3.6.2 Termes associés à un bloc

On peut représenter un bloc sous forme d'une expression en lui associant un terme.
On peut associer à un bloc:

- un produit appelé terme conjonctif w
- une somme appelée terme disjonctif W

Règle

- Si $x_i = 0$ dans B , on écrit $/x_i$ dans w , x_i dans W
- Si $x_i = 1$ dans B , on écrit x_i dans w , $/x_i$ dans W
- Si $x_i = -$ dans B , x_i est alors absente dans w et W

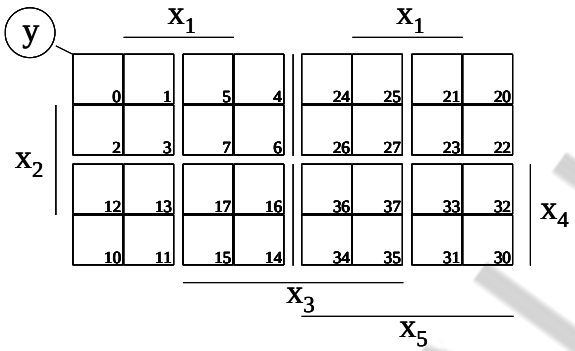
Exemple B = (x_5,x_4,x_3,x_2,x_1) = (-,1,-,0,1)
w = $x_4\ /x_2\ x_1$; W = $/x_4 + x_2 + /x_1$

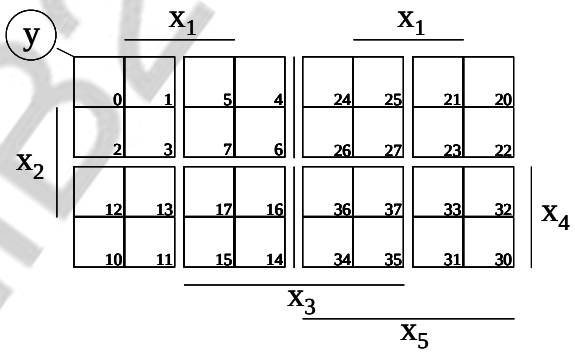
On peut obtenir le terme directement à partir de la table de Karnaugh selon la règle suivante:

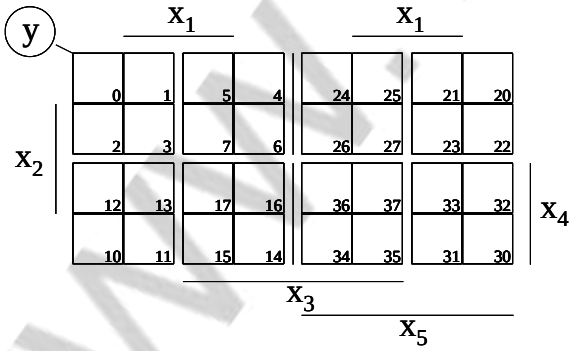
Règle

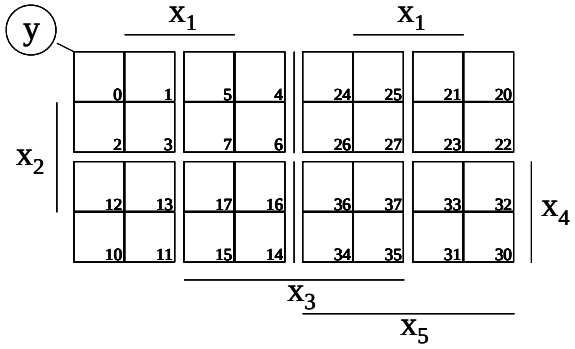
- Si B est en face du segment de x_i, on écrit x_i dans w et $/x_i$ dans W.
- Si B n'est pas en face du segment de x_i, on écrit $/x_i$ dans w et x_i dans W.
- Si B n'est que partiellement en face de x_i, x_i est alors absente dans w et W.

Exemples









Blocs maximaux

Un bloc (nul ou non nul) est dit maximal, s'il n'est pas contenu dans aucun autre bloc de même type.

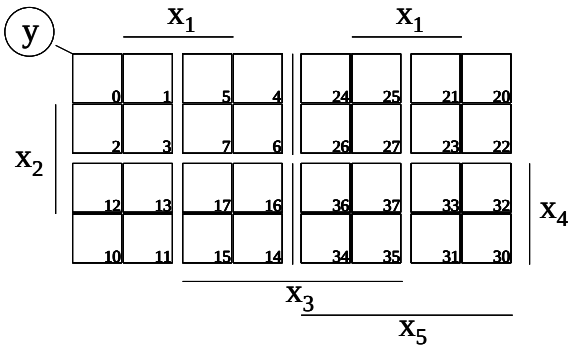
Termes premiers

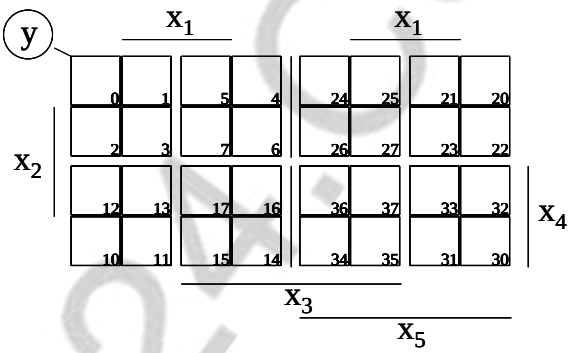
Ils sont les termes associés aux blocs maximaux.

Notation p : Terme premier conjonctif

P : Terme premier disjonctif

Exemples





3.7 Simplification des fonctions

Avant de passer à la réalisation d'une fonction, il faut la simplifier afin de réduire le coût de son circuit.

Définition d'une fonction de coût

Le coût d'une fonction correspond au nombre total des entrées des portes réalisant cette fonction.

Exemple

y = AB + /CD/E + /A

Coût(y) = 8

Remarque

Il y a plusieurs méthodes pour simplifier une fonction. On distingue les méthodes graphiques et les méthodes algorithmiques.

On se limitera au premier type de méthodes.

3.7.1 Méthode graphique pour la simplification des fonctions

Règle

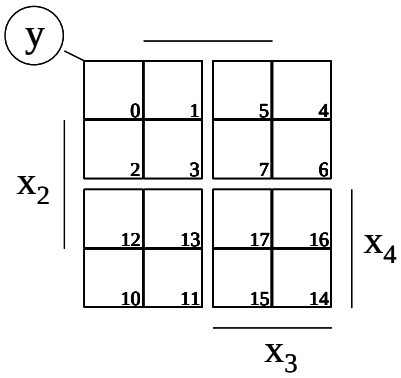
- Forme disjonctive (SOP)
  Une fonction peut s'écrire sous forme de somme des termes premiers conjonctifs p_i qui sont strictement nécessaires pour couvrir tous les '1' de la fonction.
- Forme conjonctive (POS)
  Une fonction peut s'écrire sous forme de produit des termes premiers disjonctifs P_i qui sont strictement nécessaires pour couvrir tous les '0' de la fonction.

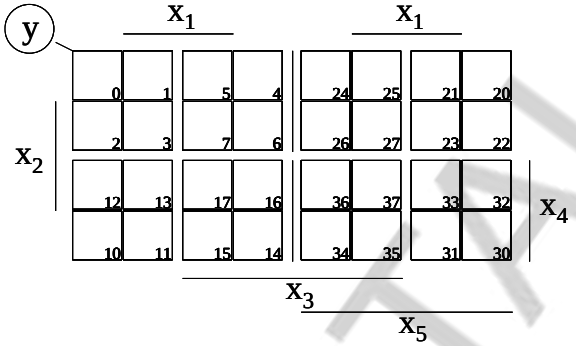
Méthode

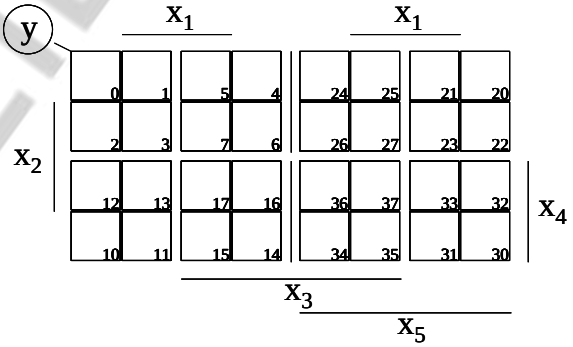
- On cherche un minimum de blocs qui couvre tous les '1' de la fonction (dans le cas de SOP) ou tous les '0'de la fonction (dans le cas de POS). Ceci permet de minimiser le nombre de termes dans l'expression de la fonction.
- Il faut que les blocs soient maximaux. Pour cela on peut exploiter les redondances pour augmenter la taille de ces blocs. Plus le bloc est grand, plus le terme correspondant sera simple.

3.7.1 Exemples

(Voir cours)



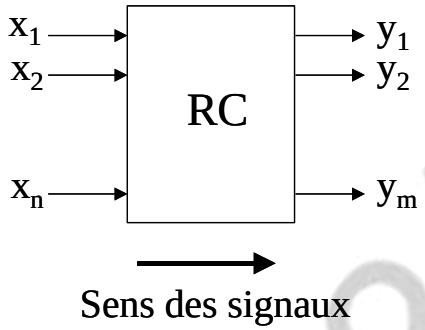




Chapitre 4 : Conception et réalisation des réseaux combinatoires

Les réseaux (circuits ou systèmes) combinatoires (RC) sont des circuits réalisant des fonctions binaires.
Caractéristiques des RC :

- La sortie ne dépend que des valeurs instantanées des entrées.
- Les signaux vont tous dans le même sens (de l'entrée vers la sortie)



4.1 RC à portes AND/OR

(Voir Chapitre 3)

4.2 RC à portes NAND ou NOR

Parfois on réalise les RC avec des portes NAND ou NOR au lieu de AND et OR.
Les méthodes de conception restent les mêmes. On utilise des règles de transformation des structures AND/OR en structures NAND ou NOR.

4.2.1 Méthode algébrique

- On inverse deux fois l'expression de la fonction.
- On applique le théorème de De Morgan pour la 1ère inversion.

Exemple (Voir cours)

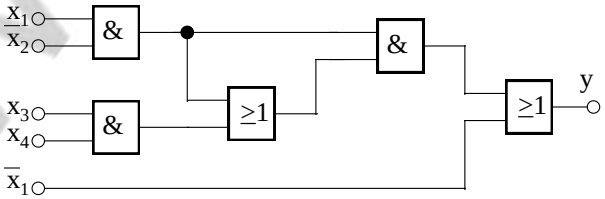
$y = a.b + /c.d.e$

4.2.2 Méthode schématique

Numérotation des niveaux d'un RC

- On attribue le numéro "1" à toute porte qui n'alimente pas une autre porte.
- On attribue le numéro i+1 à toute porte qui alimente au moins une porte de niveau i et des portes de niveau k < i.

Exemple (Voir cours)



Règles de transformation

Règle A: (Forme disjonctive vers NAND ; Forme conjonctive vers NOR)

- *Remplacer les portes AND et OR par des portes NAND <NOR>*

Règle B: (Forme conjonctive vers NAND ; Forme disjonctive vers NOR)

- *Remplacer les portes AND et OR par des portes NOR <NAND>*
- *Ajouter une porte NOT à la sortie*

Règle C:

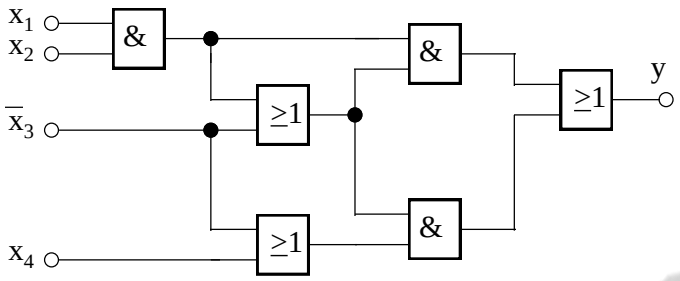
- *Inverser les variables indépendantes dans le RC transformé alimentant une porte de niveau impair*

Remarque

Ces règles ne sont pas applicables s'il y a des portes qui sont liées à d'autres portes de même type. Si ce cas se présente, il faut insérer entre ces portes des portes de type différent.

Exemple

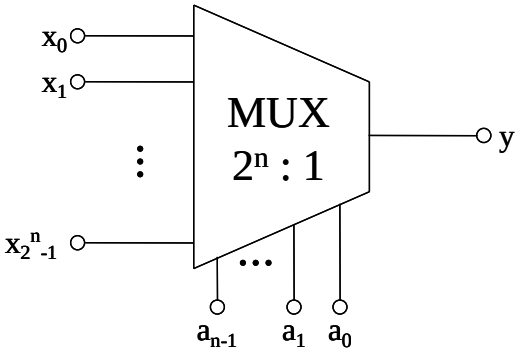
Transformer le circuit suivant en un RC à portes NAND et en un RC à portes NOR :



4.3 RC à multiplexeurs

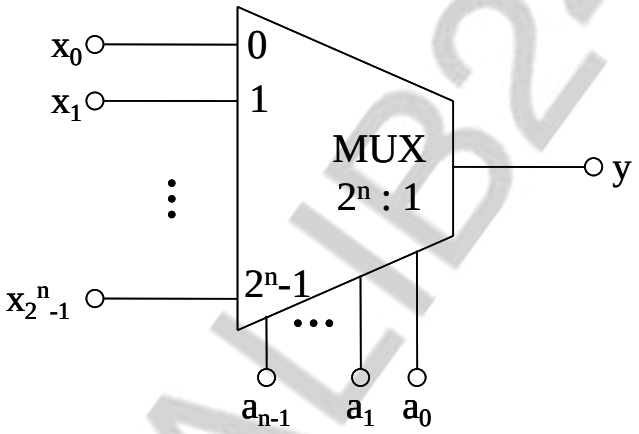
4.3.1 Généralités

Un multiplexeur est un aiguilleur de données. Il possède 2^n entrées de donnée x_i et n entrées d'adresse a_i. La sortie affiche la valeur de l'entrée x_i adressée par a_i.



Exemple Multiplexeur 2:1

Expression générale d'un multiplexeur



$y = A_0 x_0 + A_1 x_1 + \ldots + A_{2^n-1} x_{2^n-1}$

A_k : Combinaison de l'adresse

x_i : Entrée adressée par A_i.

4.3.2 Réalisation de RC dans des MUX

1ère méthode

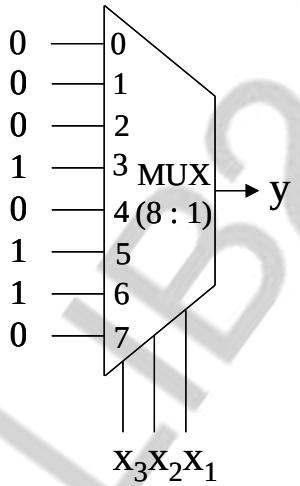
Elle est basée sur la table de vérité.

On câble les entrées x_i du MUX avec les valeurs de la fonction.

Exemple

$y = g(x_3, x_2, x_1)$

j	x_3	x_2	x_1	$y = g(X_j)$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



Remarque

Cette méthode n'est pas optimale.

2ème méthode

Elle est basée sur le théorème de Shannon.

Rappel Théorème de Shannon

Développement d'une fonction par rapport à une variable x_i:

$$f(x_{n-1}, \dots, x_0) = x_i f\Big|_{x_i=1} + \bar{x}_i f\Big|_{x_i=0}$$

1er cas: RC contenant des MUX et AND/OR

Algorithme

1ère étape: On choisit la ou les variables indépendantes, qui adresseront le ou les multiplexeurs.

2ème étape: On développe la fonction donnée par rapport à ces variables. Les fonctions restantes demeurent inchangées.

Exemple

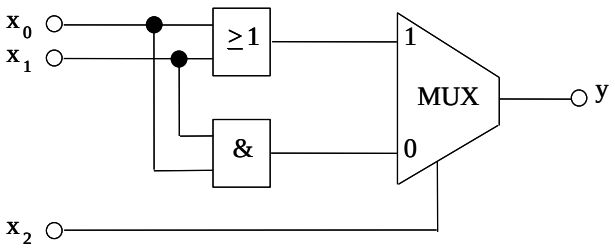
Réaliser la fonction suivante à l'aide de fonctions AND, OR et MUX :

$$f(x_2, x_1, x_0) = \bar{x}_2 x_1 x_0 + x_2 x_0 + x_2 x_1$$

- On choisit x_2 comme variable d'adresse.
- On développe y par rapport à x_2.

$$f(x_2, x_1, x_0) = x_2(x_0 + x_1) + \bar{x}_2(x_0 . x_1)$$

On obtient alors le circuit suivant:



2ème cas: RC contenant un seul multiplexeur

Algorithme

1ère étape: On choisit une variable indépendante, qui adresse le multiplexeur.

2ème étape: On développe la fonction donnée par rapport à cette variable. On utilise la distributivité pour faire apparaître l'expression du MUX.

3ème étape: S'il reste d'autres fonctions, on répète les étapes précédentes, si non fin.

Exemple

$$f(x_2, x_1, x_0) = \bar{x}_2 x_1 x_0 + x_2 x_0 + x_2 x_1$$

Réaliser cette fonction à l'aide d'un MUX.

- On choisit x_2 comme variable d'adresse.
- On développe y par rapport à x_2.

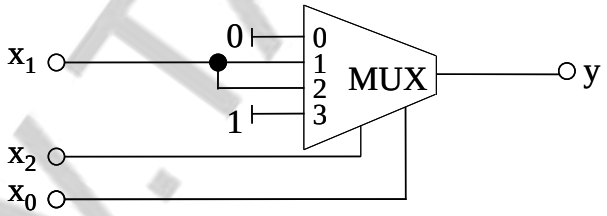
$$f(x_2, x_1, x_0) = x_2(x_0 + x_1) + \bar{x}_2(x_0 . x_1)$$

- Revenir à la première étape
- On choisit x_0 comme variable d'adresse.

(Voir cours)

y =

On obtient le circuit suivant :



3ème cas: RC contenant plusieurs multiplexeurs

Algorithme

Comme l'algorithme du 2ème cas, sauf qu'ici on ne développe que les fonctions résiduelles mais pas la fonction tout entière

Exemple

$$f(x_2, x_1, x_0) = \bar{x}_2 x_1 x_0 + x_2 x_0 + x_2 x_1$$

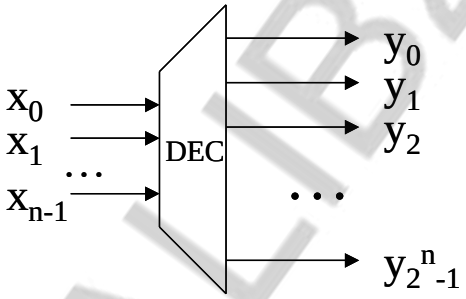
Réaliser cette fonction à l'aide de plusieurs multiplexeurs 2:1.

(Voir cours)

4.4 RC à décodeurs

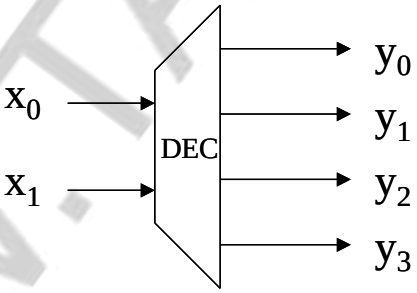
4.4.1 Décodeurs

Un décodeur est un réseau combinatoire qui transforme le code binaire naturel en code 'un parmi n'.



Exemple

Décodeur à deux entrées:



4.4.2 Réalisation des RC avec décodeur

Méthode

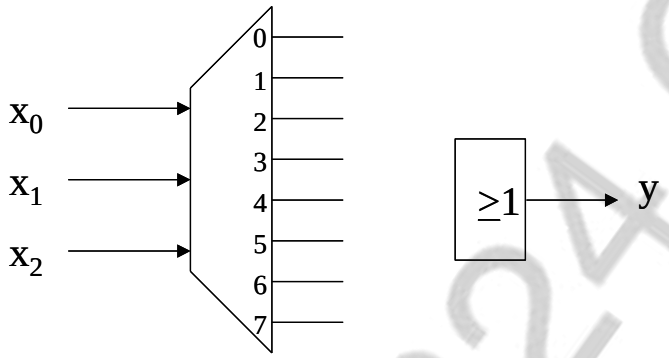
Les entrées du décodeur sont les variables d'entrée. Ses sorties correspondent aux combinaisons des variables d'entrée.

Il suffit de relier les sorties du décodeur correspondant à U(f) à une porte OR dont la sortie donnera la fonction du RC.

Exemple

On considère la fonction $y = g(x_3,x_2,x_1)$ décrite par la table de vérité suivante :

j	x_3	x_2	x_1	$y = g(X_j)$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



4.5 RC à mémoires

Une mémoire sert à mémoriser des données. On distingue les mémoires vives et les mémoires mortes.

–Mémoires mortes

Elles gardent les données même si on coupe l'alimentation.

Exemples ROM, PROM, EPROM, EEPROM, ispROM

–Mémoires vives

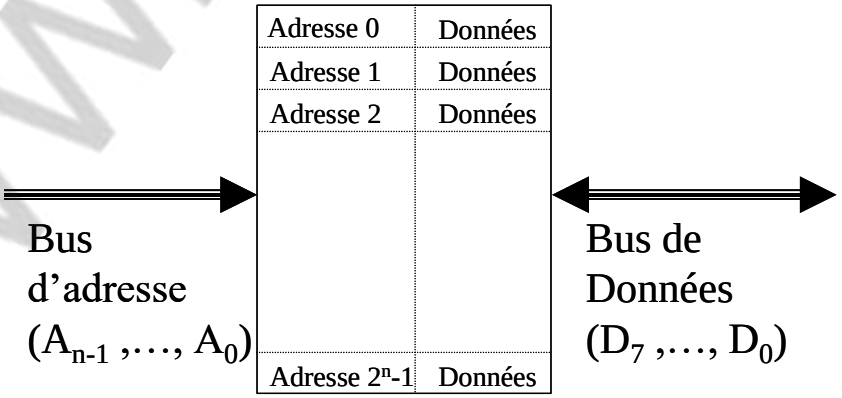
Elles gardent les données tant qu'elles sont branchées à l'alimentation.

Exemple RAM

Pour la réalisation des RC on utilise des mémoires mortes.

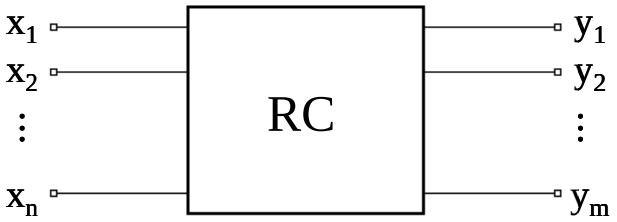
Structure d'une mémoire

Les données sont organisées en des mots généralement de 8 bits (D_8,D_7,...,D_2,D_1). Ces mots sont adressés par des variables d'adresse (A_n,...,A_1,A_0).



Cette structure peut réaliser des RC à plusieurs sorties (jusqu'à 8). Chaque bit D_i réalise une sortie du RC. Les variables d'adresse A_i réalisent les variables indépendantes.

Forme générale d'un RC



La table de vérité du RC a la forme suivante

x_{n-1}	…	x_1	x_0	y_{m-1}	…	y_1	y_0
0		0	0	-		-	-
0		0	1	-		-	-
	…				…		
1		1	1	-		-	-

Disposition des données dans une ROM

A_{n-1}		…	A_0	D_7		…	D_0
		…				…	
		…				…	
		…				…	
		…				…	

La comparaison avec la table de vérité du RC donne:

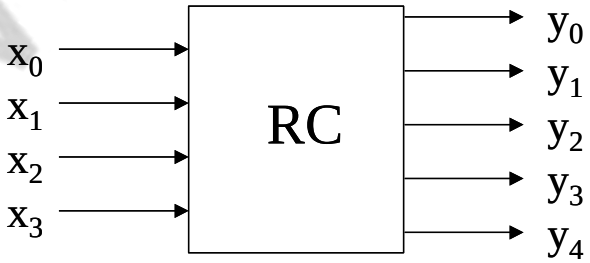
$A_i = x_i$ et $D_i = y_i$

Résultat

Il suffit donc de relier les entrées du RC au bus d'adresse et ses sorties au bus de données.

Exemple

Réaliser un RC qui transforme le code BCD-8421 en code BCD deux sur cinq à l'aide d'une EPROM 2716.



Sa table de vérité est :

x_3	x_2	x_1	x_0	y_4	y_3	y_2	y_1	y_0

0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	1
0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0
0	1	1	1	1	0	0	0	1
1	0	0	0	1	0	0	1	0
1	0	0	1	1	0	1	0	0

Attribution des signaux aux pins de l'EPROM

Signaux d'entrée:

$A_0=x_0$; $A_1=x_1$; $A_2=x_2$; $A_3=x_3$; $A_i=0$, i=4,5,...,10

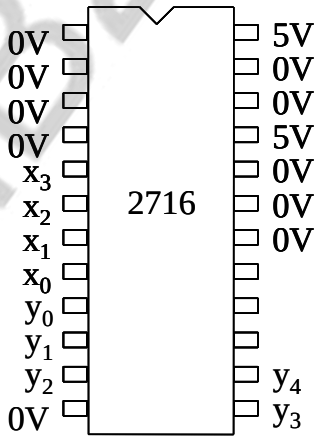
Signaux de sortie:

$D_0= y_0$; $D_1= y_1$; $D_2= y_2$; $D_3= y_3$; $D_4= y_4$; $D_i= 0$, i=5,6,7

Programme de l'EPROM (Voir cours)

Adresse	Donnée

Brochage du composant



Chapitre 5 : Circuit séquentiels

5.1 Généralités

• Réseaux combinatoires et séquentiels

Les réseaux combinatoires sont caractérisés par le fait que les signaux vont tous dans le même sens (de l'entrée vers la sortie).

Il y a des circuits dans lesquels on trouve des rebouclages (feed-back). Ici des signaux vont de la sortie vers l'entrée. Ces circuits s'appellent réseaux séquentiels et ont un grand intérêt pratique.

Parmi les éléments essentiels des réseaux séquentiels on trouve les bascules.

• Effet du rebouclage

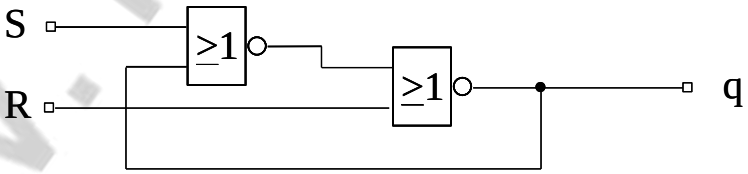
Exemples: (A compléter dans le cours)

La sortie q est instable.

Ce dernier circuit donne une sortie stable mais elle n'est pas contrôlable, c'est-à-dire on ne peut pas choisir la valeur de la sortie q. Il faut ajouter des entrées pour contrôler le circuit et définir les valeurs de sa sortie. On obtient alors une bascule. Une bascule est donc une mémoire. Elle sert à mémoriser un bit.

5.2 Types de bascule

5.2.1 Bascule RS asynchrone



Avec les entrées R et S on peut faire passer la sortie q à 0 ou à 1.

Reset : Passage à 0 (R=1)

Set : Passage à 1 (S=1)

Autre représentation:

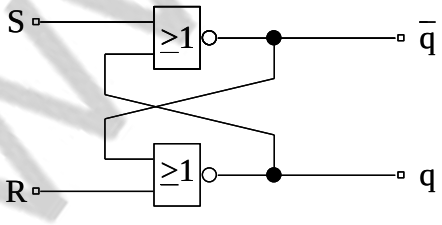


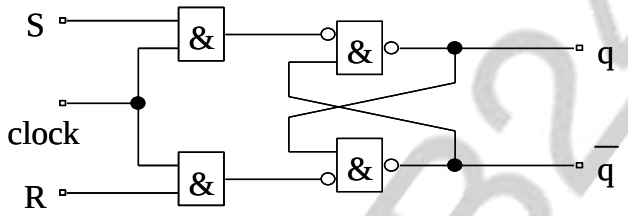
Table de vérité de la bascule RS

R	S	q^+
0	0	
0	1	
1	0	
1	1	

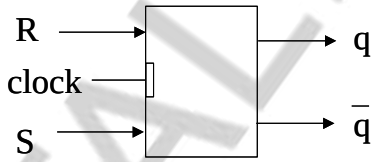
Remarques
• Le cas RS=11 pose des problèmes. Dans ce cas, on a q = /q, ce qui n'est pas logique. Pour cette raison il ne faut jamais attaquer les entrées d'une bascule RS avec la combinaison 11.
• La bascule RS précédente génère la sortie q dès que R et/ou S reçoive une nouvelle valeur. On dit qu'elle est asynchrone.

5.2.2 Bascule RS synchrone

Une bascule RS synchrone possède un autre signal d'entrée appelé horloge (clock). La sortie n'est alors modifiée que si cette horloge est active.
Remarque
L'horloge est généralement un signal carré.
L'horloge peut agir soit avec son niveau (0 ou 1) soit avec son front (montant ou descendant).
Dans le premier cas, on dit que la bascule est sensible au niveau (level sensitive).
Dans le deuxième cas, on dit que la bascule est sensible au front (edge sensitive).
Bascule RS synchrone sensible au niveau



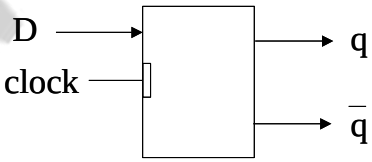
La bascule n'est sensible à R et S que si l'horloge est à 1.
Symbole de la bascule RS sensible au niveau



Remarque
Le cas de la bascule sensible au front sera vu ultérieurement (voir bascule D).

5.2.3 Bascule D synchrone sensible au niveau

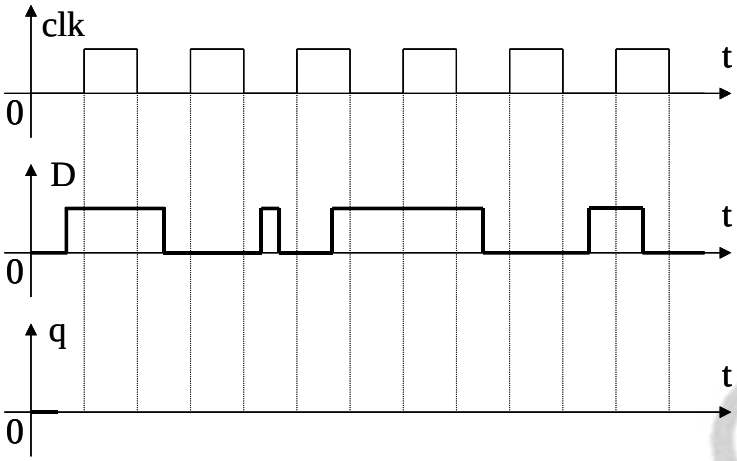
On l'appelle aussi latch ou verrou.
– Schéma-bloc:



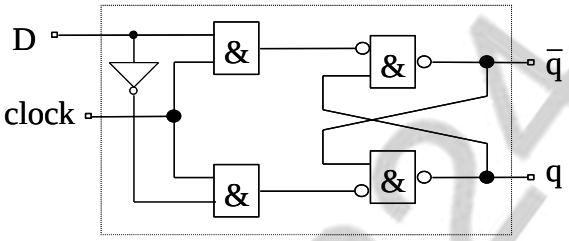
– Table de vérité du latch:

D	q^+
0	
1	

q ne prend la nouvelle valeur que si l'horloge clock a le niveau actif.
– Exemple:
(Voir page suivante)



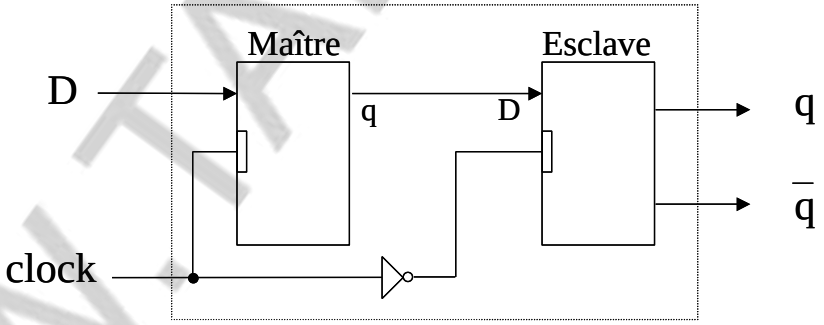
– Circuit:



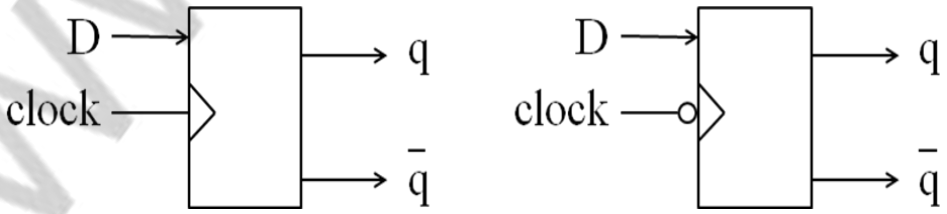
– Durée d'impulsion de l'horloge:
Il faut qu'elle soit supérieure à une valeur minimale donnée par le constructeur (en général plusieurs ns).

5.2.4 Bascule D synchrone sensible au front

On préfère avoir des bascules qui sont sensibles aux fronts de l'horloge.
On utilise alors deux bascules sensibles au niveau montées en maître-esclave.



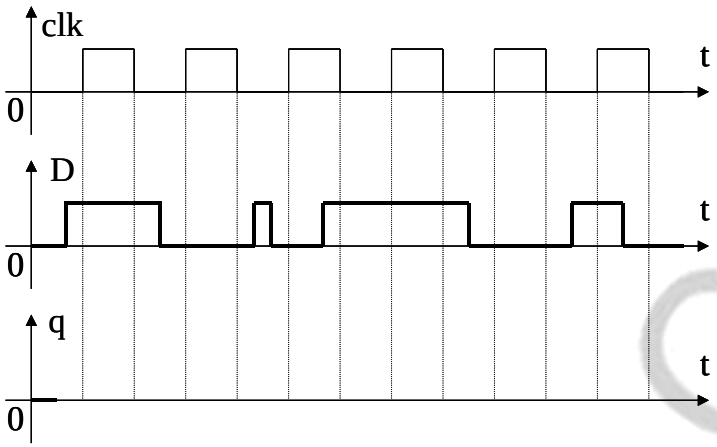
Symboles



Bascule D sensible au front montant Bascule D sensible au front descendant

Remarques
• Une bascule qui est sensible au front s'appelle flip-flop.
• Par défaut une bascule est considérée comme sensible aux fronts.

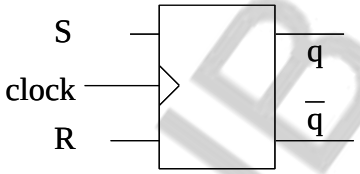
– Exemple:



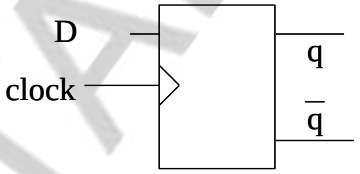
5.2.5 <u>Bascule T</u> (Toggle)

C'est une bascule qui change d'état à chaque front actif de l'horloge, sinon elle mémorise l'ancienne valeur de q.

– Circuit à base de bascule RS: (A compléter)

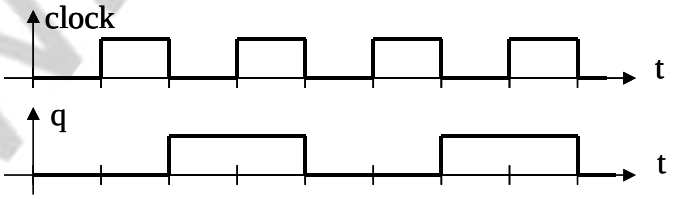


– Circuit à base de bascule D: (A compléter)



–Chronogramme:

On suppose que le front descendant est le front actif.



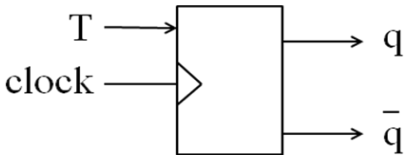
On constate que la fréquence de q est la moitié de celle de l'horloge. Une bascule T peut donc être utilisée comme <u>diviseur de fréquences</u>.

– Bascule T avec entrée de validation:

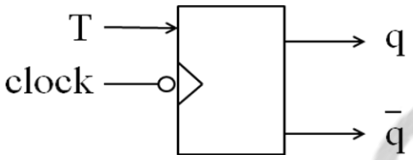
On trouve souvent des bascules T possédant une entrée de validation T qui autorise le basculement avec T=1 et l'interdit avec T=0. Sa table de vérité est:

T	q^+
0	
1	

Son symbole est:



Bascule T sensible au front montant

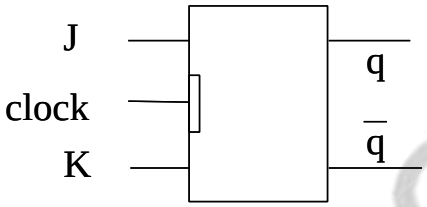


Bascule T sensible au front descendant

5.2.6 Bascule JK

On se limitera à l'étude de la bascule JK synchrone sensible au niveau.

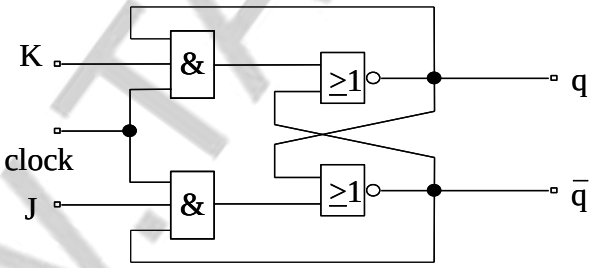
– Symbole:



– Table de vérité:

J	K	q^+
0	0	
0	1	
1	0	
1	1	

– Circuit:



5.2.7 Utilité des différents types de bascules

Les différents types de bascules sont classés ci-dessous suivant leur importance:

- ✓ Bascule D: Très simple et facile à utiliser
- ✓ Bascule JK: Complète mais complexe
- ✓ Bascule T: Simple mais difficile à utiliser
- ✓ Bascule RS: Simple mais difficile à utiliser

5.3 Circuits à bascules

On utilise les bascules dans les circuits (systèmes ou réseaux) séquentiels. Dans ce chapitre on étudiera que des circuits séquentiels simples. Dans le chapitre 2, on procédera à l'étude générale des systèmes séquentiels.

5.3.1 Définitions

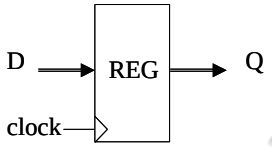
– Réseaux séquentiels :
Les réseaux séquentiels sont constitués de bascules et de portes logiques.
– Séquenceur :
Les réseaux séquentiels s'appellent aussi séquenceur.
– Etat d'un circuit séquentiel :
Il est donné par les valeurs des sorties des bascules.
– Circuit séquentiel synchrone :
Ses bascules sont synchronisées par une horloge qui peut être sensible au niveau ou au front. Mais on préfère les bascules sensibles au front, car les moments de changement d'état sont connus de manière plus précise.

5.3.2 Registres

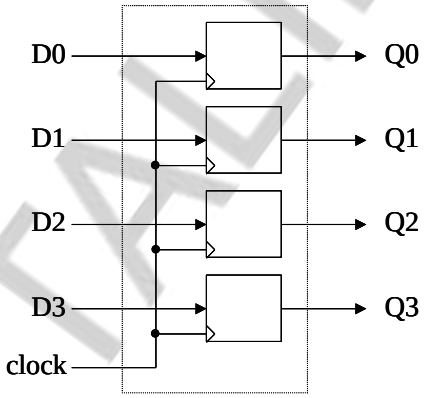
Un registre est un circuit séquentiel qui sert à mémoriser des données.

5.3.2.1 Registres à entrées parallèles

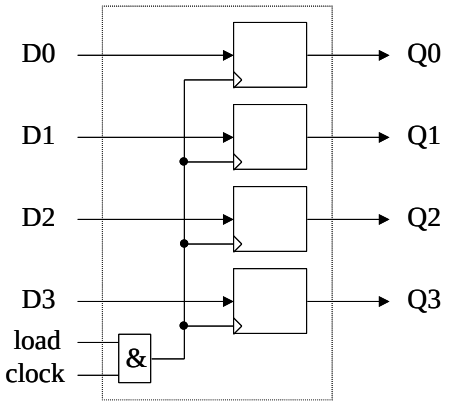
– Schéma-bloc :



– Structure interne :
On utilise des bascules D. Chaque bit sera mémorisé dans une bascule (souvent de type D).

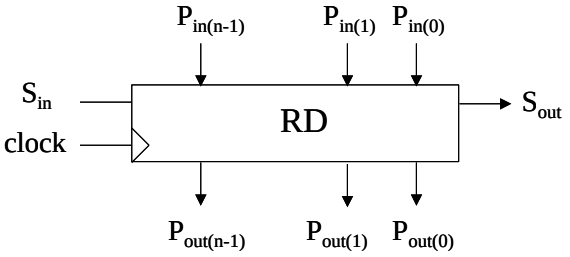


Un tel registre s'appelle Parallel In Parallel Out (PIPO).
– Registre avec entrée load :
Cette entrée permet de charger ou de ne pas charger le registre. Tant que load est à 1, le registre charge les valeurs d'entrée à chaque front actif de l'horloge. Si load=0, le registre mémorise les anciennes données.



5.3.2.2 Registres à décalage

– Schéma-bloc général:



S_{in} : Entrée série
S_{out} : Sortie série
$P_{in(n-1)}\ldots P_{in(1)}\ P_{in(0)}$: Entrée parallèle
$P_{out(n-1)}\ldots P_{out(1)}\ P_{out(0)}$: Sortie parallèle

– Types de registres à décalage:

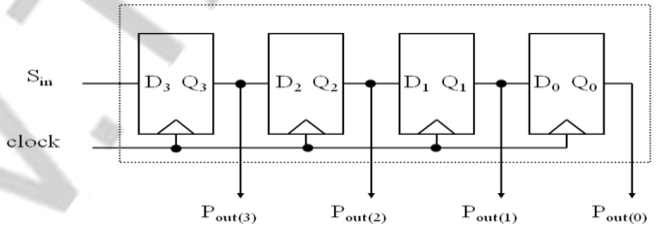
Entrée série et sortie série: SISO
Application : Mémoire série de type 'premier entré premier sorti' (FIFO)
Entrée parallèle et sortie parallèle: PIPO
Application : Mémoire parallèle
Entrée série et sortie parallèle: SIPO
Application : Convertisseur série-parallèle
Entrée parallèle et sortie série: PISO
Application : Convertisseur parallèle-série

– Signaux de commande:

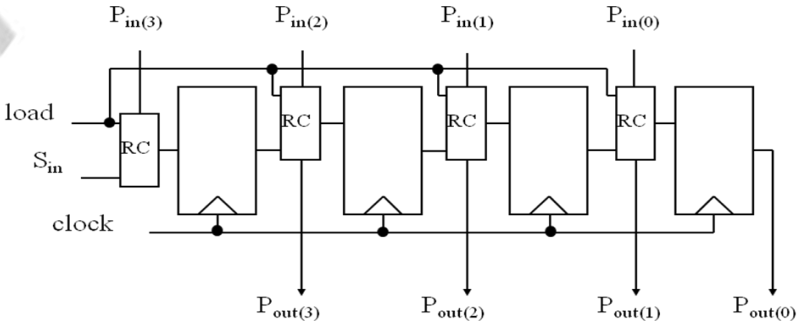
- Remise à zéro (clear):
  - Synchrone: Avec l'horloge
  - Asynchrone: Indépendamment de l'horloge
- Chargement (load):
  - Synchrone: Avec l'horloge
  - Asynchrone: Indépendamment de l'horloge
- Décalage (shift):
  - A droite
  - A gauche

– Exemples:

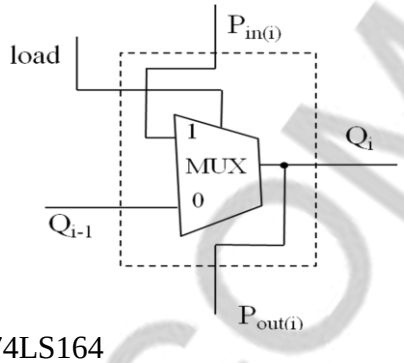
• Principe du SIPO à 4 bits



• Principe du PIPO/SIPO à 4 bits

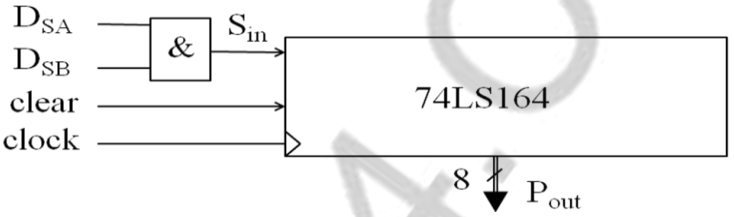


Le circuit combinatoire RC possède la structure suivante :



– Exemple de circuit intégré contenant un registre à décalage: 74LS164

C'est un SIPO à 8 bits. Il y a 2 bits D_{SA} et D_{SB} dont le produit forme l'entrée série. Il possède un clear asynchrone et une horloge active au front



5.3.3 Compteurs

– Définitions:

C'est un circuit séquentiel qui assure la fonction de comptage.

Exemple: 0 – 1 – 2 – 3 – 4 -…

Les valeurs fournies par le compteur forment la séquence de comptage. Une séquence de comptage peut ne pas commencer par 0. La longueur de la séquence de comptage s'appelle le modulo. Le passage de la valeur N à la valeur N+1 s'appelle incrémentation.

Un compteur peut aussi compter dans le sens inverse. On parle de décomptage. L'inverse de l'incrémentation est la décrémentation.

– Caractéristiques:

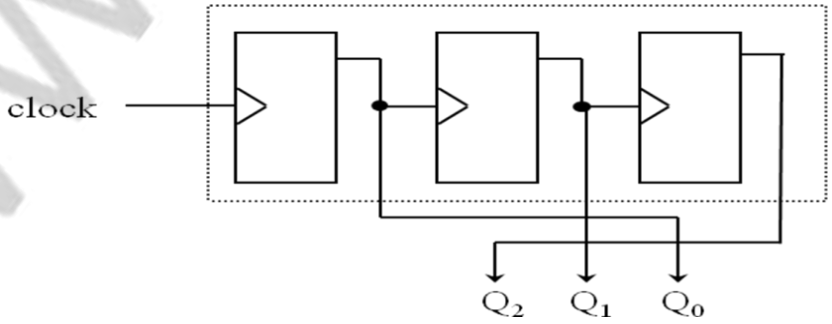
- ✓Modulo:
  - Compteur décimal : Modulo 10
  - Compteur héxadécimal: Modulo 16
  - Compteur binaire à n bits: Modulo 2^n
- ✓Remise à 0 (clear, reset):
  - Synchrone ou asynchrone
- ✓Chargement (load):
  - Synchrone ou asynchrone
- ✓Codage des sorties:
  - Code binaire naturel, Code 1 parmi n, …

– Types de compteurs:

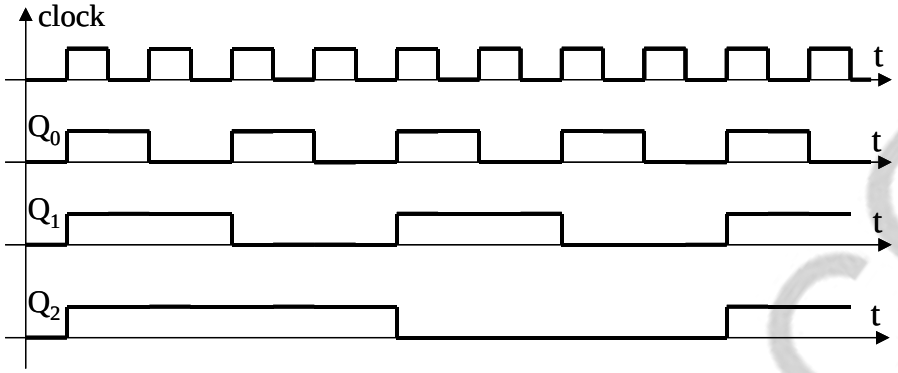
- ✓Asynchrone (Ripple):

Les bits changent de valeurs l'un après l'autre (pas simultanément).

Exemple: Décompteur à bascules T

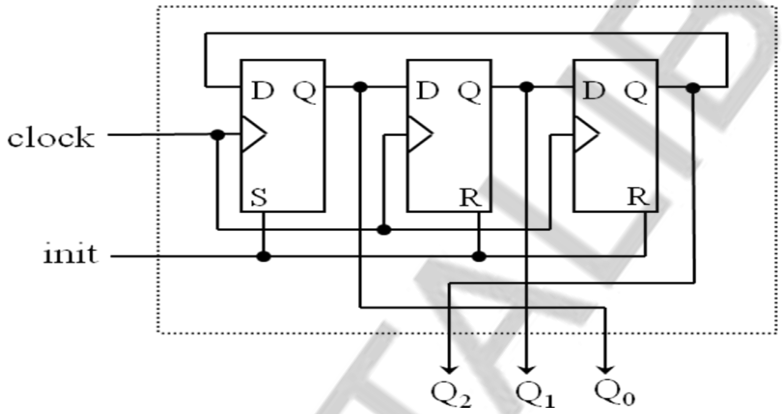


Les signaux de sortie de ce décompteur évoluent comme le montre la figure suivante :



✓Synchrone:
Les sorties sont générées quasi-simultanément.
✓En anneau:
Il fait circuler un '1' ou un '0'(code un parmi n, one-hot, one-cold).
Le nombre de bascules est égal au modulo.
Exemple: Compteur modulo 3 en anneau à bascules D

Circuit

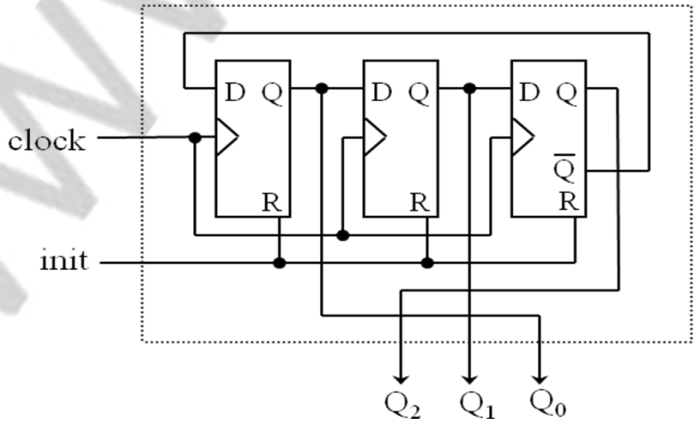


Séquence de comptage

Q_2	Q_1	Q_0
0	0	1
0	1	0
1	0	0
0	0	1
…		

✓Compteur de Johnson:
Il ressemble au compteur en anneau, mais il ne nécessite que la moitié du nombre de bascules. Si on a N bascules, le compteur sera modulo 2N. La séquence commence par 0…00. On reboucle l'inverse du MSB vers l'entrée de la bascule qui fournit le LSB.
Exemple: Compteur de Johnson modulo 6 à bascules D

Circuit



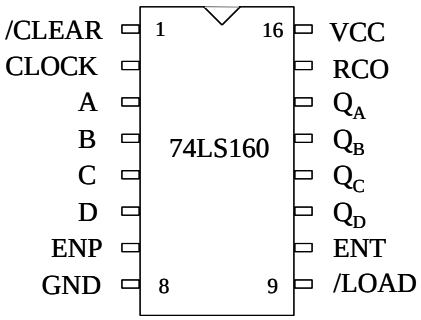
Séquence de comptage

Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0
…		

✓Compteur à séquence quelconque:
(Voir prochain chapitre)

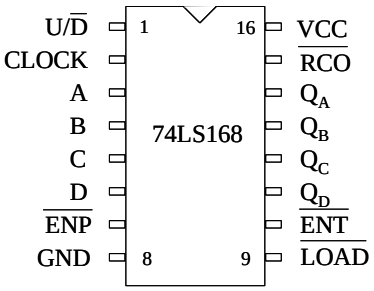
– Exemples de compteurs:

✓74LS160:



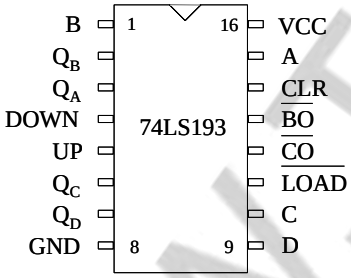
C'est un compteur décimal à 4 bits avec clear asynchrone et load synchrone.

✓74LS168:



C'est un compteur/décompteur décimal à 4 bts avec load synchrone.

✓74LS193:



C'est un compteur/décompteur héxadécimal à 4 bits avec clear asynchrone et load asynchrone.

– Exercices:

a- Câbler le compteur 74LS161 pour qu'il réalise la séquence de comptage suivante:
– 4 – 5 – 6 – 7 – 8 – 3 – 4 – 5 – …

b- Refaire la question précédente pour le circuit 74LS193.