UNIVERSITE DE TUNIS

INSTITUT SUPERIEUR DE L'EDUCATION ET LA FORMATION CONTINUE

BARDO

DEPARTEMENT DES SCIENCES PHYSIQUES ET TECHNIQUES

COURS : ELECTRONIQUE NUMERIQUE

PROPOSE PAR : H . MECHERGUI

ANNEE SCOLAIRE : 2004

TABLE DE MATIERE

1- Introduction à l'électronique numérique 8]

Les images des grandeurs physiques récupérées en sortie des capteurs sont analogiques. Elles évoluent continûment en fonction du temps. Longtemps, elles ont été récupérées et traitées comme telles et elles le restent encore dans de nombreux systèmes (télévision pour quelques temps encore). Cependant, ce type de signal pose de nombreux problèmes (conception des systèmes délicate, sensibilité au bruit, stockage d'information moins performant…).

L'électronique numérique permet d'atténuer nombre de ces inconvénients. En effet, les valeurs des signaux étant quantifiées, ils sont moins sensibles au bruit (transmissions), le stockage d'informations est plus simple et plus fiable, on peut réaliser ou programmer des séquences évoluées complexes et enfin, l'intégration des composants est bien plus grande qu'en analogique (on n'a plus à intégrer des inductances ou des capacités notamment…). Dans ce qui suit, les concepts fondamentaux des systèmes logiques sont présentés

.

Variable continue (infinité de valeurs)

Quantité, représentée par un symbole, qui peut prendre une infinité de valeurs.

$y = ax^3 + bx^2 + c$ où

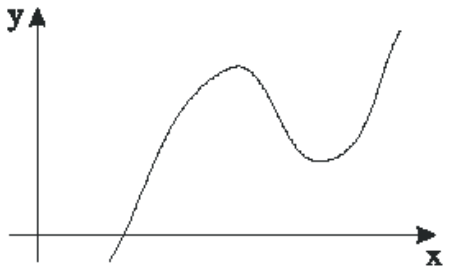
y est la fonction,

= est le symbole d'égalité,

x est la variable,

+ est un opérateur,

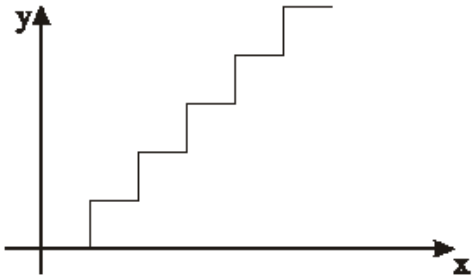
a, b et c sont des constantes



Variable discrète

Variable susceptible de prendre un nombre limité de valeurs prédéfinies et discontinues.

y = int(x) (partie entière de x)



Variable binaire

Variable discrète qui ne peut prendre que deux et seulement deux valeurs.

Variable logique

Variable binaire qui peut prendre deux états associés au caractère vrai ou faux d'un événement.

état Logique

Valeur attribuée à une variable logique. L'état d'une variable peut être vrai ou faux. On représente l'état vrai par "1" et l'état faux par "0". Une variable dans son état vrai est dite "active".

Opérateurs Logiques

Les opérateurs logiques de base sont ET, OU et NON.

Fonction Logique

Ensemble de variables logiques reliées par des opérateurs logiques. Une fonction logique ne peut prendre que deux valeurs: 0 ou 1.

Signal Logique
Quantité physique qui représente une variable logique dans l'un ou l'autre de ses deux états possibles.
Système Logique
Ensemble de composants qui effectuent des fonctions sur des signaux logiques dans le but de stocker, communiquer ou de transformer de l'information.

2- Codages des informations numériques [9]
2.1 les systèmes de numération
2.1.1 Numération décimale
Ce système de numération, usuel dans la vie quotidienne, dispose de dix symboles (les chiffres) :0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
On travaille alors en **base 1**0.

Exemple : $7239 = 7.10^3 + 2.10^2 + 3.10^1 + 9.10^0$

De manière général e, un nombre s'écri vant $N = a_{n-1}...a_1a_0$ (les *ai* représentent les *n* chi ffres) dans une base *B* (on dispose de *B* symboles) s'écrit

$$N = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \ldots + a_1B^1 + a_0B^0$$

On note alors $N = (a_{n-1}...a_1a_0)_B$. La base *B* est notée en indice, codée en décimal.

2.I.2. Numération binaire
La numération en base deux (ou **numération binaire**) utilise deux symboles 0 et 1. Cette base est très commode pour di sti nguer l es deux états l ogiques fondamentaux.
On écrit :

$(a_{n-1}a_{n-2}...a_1a_0)_2 = a_{n-1}.2^{n-1} + \ldots + a_0.2^0$ (expression de droite écrite dans la base 10 et $a_i \in \{0, 1\}$).

Exemple : $(4)_{10} = 1.2^2 + 0.2^1 + 0.2^0 = (100)_2$

Un nombre à *n* chiffres en base deux distingue 2^n états.

Un état binaire est appelé **bit** (contraction de *binary digi*t). Un bit prend les valeurs 0 ou 1.
Les puissances successives de 2 (1, 2, 4, 8, 16, 32, 64, 128, 256,...) sont appelées **poids binaire** s. En général, le poids du bit de rang *n* est 2^n **(attention, on commence toujours au rang** 0). Le bit de poids le plus fort est appelé MSB ***(****Most Significant Bi*t).
Le bit de poids le plus faible est appelé LSB ***(****Less Significant Bi*t).:

2.I.3. Numération octale
Ce système utilise 8 symboles : 0, 1, 2, 3, 4, 5, 6, 7. Il n'est plus guère employé aujourd'hui, puisqu'il servait au codage des nombres dans les ordinateurs de première génération.

$(N)_8 = a_{n-1}..a_0$, $(N)_{10} = a_{n-1}8^{n-1} + a_{n-2}8^{n-2} + \ldots + a_1 8^1 + a_0 8^0$

$472_o = 100\ 111\ 010_b$ (on peut vérifier que ça vaut 314_d). En effet, 100111010_b

$$\begin{array}{lccccccccc} = & 1\cdot2^8 + & 0\cdot2^7 + & 0\cdot2^6 + & 1\cdot2^5 + & 1\cdot2^4 + & 1\cdot2^3 + & 0\cdot2^2 + & 1\cdot2^1 + & 0\cdot2^0 \\ = & 256 + & 0 + & 0 + & 32 + & 16 + & 8 + & 0 + & 2 + & 0 \\ = & 314_d & & & & & & & & \end{array}$$

2.I.4. Numération hexadécimale
Le développement des systèmes microprogrammés (mini- et micro-ordinateurs) a favorisé l'utilisation de ce code. Il comporte 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

$(N)_{16} = a_{n-1}..a_0$, $(N)_{10} = a_{n-1}16^{n-1} + a_{n-2}16^{n-2} + \ldots + a_1 16^1 + a_0 16^0$

Exemple
$(AA)_{16} = AAH = \$AA = A.16^1+A.16^0 = 10.16+10.1 = (170)_{10}$

Un quartet, ou digit hexadécimal, évolue entre 0 et 15 (en base 10) soit 0 et F en hexadécimal.
L'assemblage de 2 quartets forme un **octet** qui varie de 0 à 255 (en décimal).
Pour indiquer la base 16, on peut la noter en indice suivant la manière générale. Mais dans la pratique on utilise une autre notation. On place le caractère $ (dollar) devant le nombre ou H derrière.

$3A5_h$ vaut $3x16^2 + 10x16^1 + 5x16^0 = 3x256 + 160 + 5 = 933_d$.

Transformer de l'hexadécimal en binaire est très simple : il suffit de remplacer chaque chiffre par sa valeur binaire sur quatre bits :

$3A5_h = 0011\ 1010\ 0101_b$ (on peut vérifier que ça vaut 933_d). En effet, 001110100101_b

=	0.2^{11} +	0.2^{10} +	1.2^9 +	1.2^8 +	1.2^7 +	0.2^6 +	1.2^5 +	0.2^4 +	0.2^3 +	1.2^2 +	0.2^1 +	1.2^0
=	0 +	0 +	512 +	256 +	128 +	0 +	32 +	0 +	0 +	4 +	0 +	1
=	933_d											

Ou :

=	$3_d x16^2 + 10_d x16 + 5_d$
=	3x256+ 10x16+ 5
=	933_d

Contrairement à ce que beaucoup de gens croient, aucune machine ne compte en hexadécimal. Elles travaillent toutes en binaire, et ne se servent de l'hexadécimal que pour dialoguer avec nous (nous nous trompons trop souvent dans de longues listes de 0 et 1).

3- Changements de base – Conversions [9]

3.1.. Conversion décimal vers binaire

Il existe plusieurs moyens d'effectuer une tel l e conversion :

- par soustractions successives des poids binaires dans la différence de l'étape précédente ;
- par divisions successives par 2 du dividende de l'étape précédente. Les restes correspondants

sont les bits consécutifs. C'est terminé au premier dividende nul (que l'on de compte pas).
bits forment un octet (BYTE). Mais plusieurs codifications sont envisageables.
101100_b représente :

	$1x2^6$ +	$0x2^5$ +	$1x2^4$ +	$1x2^3$ +	$0x2^2$ +	$0x2^1$ +	$1x2^0$
=	1x64 +	0x32 +	1x16 +	1x8 +	0x4 +	0x2 +	1x1
=	89_d						

à l'inverse, pour transformer 89d en binaire, on peut utiliser la méthode des divisions successives par 2 :
on divise successivement par 2 jusqu'à un résultat de 0, les restes successifs (de bas en haut) forment le nombre binaire.

Ex : 89_d :	89 :2=44	reste	1	LSB (Less Significant Bit)
	44 :2=22	reste	0	
	22 :2=11	reste	0	
	11 :2=5	reste	1	
	5 :2=2	reste	1	
	2 :2=1	reste	0	
	1 :2=0	reste	1	MSB (Most Significant Bit)

$89_d = 1011001_b$

3.2. Conversion décimal vers hexadécimal

On reprend les deux méthodes précédentes avec des poids hexadécimaux ou en divisant par 16.

3.3. Décomposition d'un nombre décimal en octal

Les mêmes principes s'appliquent aussi.

3.4. Toutes les conversions vers le décimal

Dans tous les cas, il n'y a rien de particulier à ajouter. Le principe de conversion est directement attaché à la manière dont on écrit un nombre dans une base donnée (Cf. définition).

$(N)_B = a_{n-1}.B^{n-1} + \ldots + a_0.B^0$ où B est codé en décimal

La conversion est réalisée automatiquement dans la mesure où le résultat est écrit directement dans la base dix.

5.5. Les conversions directes (sans passer par le décimal)

Dans les bases usuelles (2, 8 et 16) utilisées dans les systèmes numériques, les conversions peuvent
être réalisées par exploitation de propriétés particulières aux nombres de ces bases.

3.5.1. Binaire vers hexadécimal

Un nombre hexadécimal est « découpable » en quartets facilement codables en binaire. Donc, pour convertir du binaire en hexadécimal, on divise le nombre binaire en « tranches de quatre » en partant de la droite. Chacun des « paquets » est ensuite converti en hexadécimal. Cette méthode revient à fractionner en décompositions successives.

Exemple : $(110101110001)_2 = (\underline{1101}\ \underline{0111}\ \underline{0001})_2 = D71H$

Explication : la mise en paquet revient à effectuer une série de factorisations partielles de la base de destination. Ici c'est $16 = 2^4$. Les résidus constituent les chiffres de la conversion. Dans l'exemple précédent, cela donne :

$$\begin{aligned}(110101110001)_2 &= 1.2^{11} + 1.2^{10} + 0.2^9 + 1.2^8 + 0.2^7 + 1.2^6 + 1.2^5 + 1.2^4 + 0.2^3 + 0.2^2 + 0.2^1 + 1.2^0 \\ &= \{1.2^3+1.2^2+0.2^1+1.2^0\}.(2^4)^2 + \{0.2^3+1.2^2+1.2^1+1.2^0\}.2^4 + \{0.2^3+0.2^2+0.2^1+1.2^0\} \\ &= 13.16^2 + 7.16^1 + 1.16^0\end{aligned}$$

3.5.2. Hexadécimal vers binaire

C'est le processus directement inverse, on écrit chaque quartet sur 4 bits en complétant éventuellement avec des zéros.

Exemple : $BC34H = (1011[B]\ 1100[C]\ 0011[3]\ 0100[4])_2 = (1011\ 1100\ 0011\ 0100)_2$

3.5.3. Binaire vers octal et inversement

On reprend les mêmes principes, sachant que $8 = 2^3$ (en factorisant $8 = 2^3$).

4. Codage des nombres

Un code constitue une correspondance entre des symboles et des objets à désigner. Les codes sont pondérés : dans une base de travail donnée, la valeur d'un rang donné est un multiple par la base de celle du rang inférieur. D'autres codes ne sont pas pondérés, c'est à dire que la position d'écriture ne correspond pas à un poids des autres. Ils ne permettent d'effectuer d'opérations arithmétiques.

4.1. Codes pondérés

4.1.1. Code naturel

Le code binaire naturel et ses dérivés (octal et hexadécimal) répondent aux règles classiques de l'arithmétique des nombres positifs (on peut calculer).

4.1.2. Code décimal codé binaire (DCB)

Dans ce codage (BCD, *Binary Coded Decimal* en anglais), chaque digit décimal est écrit en binaire puis tous sont juxtaposés. Cette représentation est commode pour traiter les nombres dans le mode de représentation le plus adapté à l'opérateur humain (lors d'un affichage par exemple).

Exemple : $7239 = (0111\ 0010\ 0011\ 1001)_{DCB}. = (1110001000111)_2$.

Table des caractères de contrôle (00 à 31)

ASCII	Caract.	Signification	ASCII	Caract.	Signification
00	NUL	*null*, nul	16	DLE	*data link escape*, échap. liaison données
01	SOH	*start of heading*, début d'en-tête	17	DC1	*device control 1*, commande unité 1
02	STX	*start of text*, début de texte	18	DC2	*device control 2*, commande unité 2
03	ETX	*end of text*, fin de texte	19	DC3	*device control 3*, commande unité 3
04	EOT	*end of transmission*, fin de transmission	20	DC4	*device control 4*, commande unité 4
05	ENQ	*enquiry*, interrogation	21	NAK	*negative acknowledge*, acc. récep. nég.
06	ACK	*acknowledge*, accusé de réception	22	SYN	*synchronous idle*, inactif synchronisé
07	BEL	*bell*, sonnerie	23	ETB	*end of transmission block*, fin tran. bloc
08	BS	*backspace*, espacement arrière	24	CAN	*cancel*, annuler
09	HT	*horizontal tabulation*, tabulation horiz.	25	EM	*end of medium*, fin du support
10	LF	*line feed*, saut de ligne	26	SUB	*substitute*, substitut
11	VT	*vertical tabulation*, tabulation verticale	27	ESC	*escape*, échappement
12	FF	*form feed*, saut de page	28	FS	*file separator*, séparateur de fichiers
13	CR	*carriage return*, retour chariot	29	GS	*group separator*, séparateur de groupes
14	SO	*shift out*, hors code	30	RS	*record separator*, sép. d'enregistr.
15	SI	*shift in*, en code	31	US	*unit separator*, séparateur d'unités

Table des caractères imprimables (32 à 127) — ou table ASCII standard

ASCII	Caractère.
32	SP (*space*, espace)
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

ASCII	Caractère
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

ASCII	Caractère
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	DEL (*delete*, sup.)

Tableau 1 : codage ASCII.

5- Algèbre de Boole - Equations Logiques [11]

5.1. Algèbre de Boole et fonctions logiques élémentaires.

L'algèbre de Boole permet de travailler avec des variables ne pouvant prendre que 2 valeurs, 0 (état bas) ou 1 (état haut). Ces valeurs correspondent à des niveaux de tension électrique qui dépendent de la technologie employée. Par exemple, si on travaille entre 0 et 5 V, une tension comprise entre 0 et 2,5 V correspondra à 0 et une tension comprise entre 2,5 et 5V à 1. Les circuits logiques comprennent souvent plusieurs entrées pour une seule sortie. On appelle table de vérité le tableau donnant la valeur de la sortie pour toutes les combinaisons possibles en entrée.

5.1.1. Variable binaire

On appelle **variable binaire** (ou logique), une variable prenant ses valeurs dans l'ensemble {0, 1}.

Exemple : état d'un interrupteur, d'un bouton poussoir, la présence d'une tension,...

Soit ***a*** la variable associée à l'état d'un bouton poussoir, alors a = 0 (faux ou bas) signifie qu'il n'est

pas actionné, a = 1 (vrai ou haut) signifie qu'i l est actionné.

Opérateurs Logiques:

I.1. Fonctions logiques élémentaires.

Il faut noter que les fonctions dont nous allons parler peuvent être réaliser avec un nombre variable d'entrées. Pour simplifier, nous nous contenterons de donner les fonctions à une ou deux entrées (notées A et B). Le résultat de l'opération sera appelé x.

Voici les 3 types d'opérateurs élémentaires:

1. **ET** (AND)
2. **OU** (OR)
3. **NON** (NOT)

Les symboles:

1. ET => **.** (un point)
2. OU => **+** (un plus)
3. NON => une barre au dessus

axiomes

Pour qu'une algèbre puisse être dite de Boole, elle doit vérifier :

Commutativité	a+b=b+a	a.b=b.a
Associativité	(a+b)+c=a+(b+c)	(ab)c=a(bc)
Distributivité	a(b+c)=ab+ac	a+(bc)=(a+b)(a+c)
Eléments neutres	a+0=a	a.1=a
Complémentarité	$\bar{a}+a=1$	$\bar{a}.a=0$

théorèmes

Une algèbre de Boole vérifie les théorèmes suivants :

Idempotence	a+a=a	aa=a
Absorption	a+ab=a	a(a+b)=a
De Morgan	$\overline{a+b}=\bar{a}.\bar{b}$	$\overline{a.b}=\bar{a}+\bar{b}$
Elément neutre	a+1=1	a.0=0

5.I.2. Equation logique

On appelle **équation logique** une combi nai son de pl usi eurs vari abl es logiques donnant l'état d'une variable dite de sortie associée. Cette combinaison est réalisée à l'aide d'opérations logiques :

Soit *xi (i* ∈ [1, n]) les variables d'entrée. L'équation *A* = f(*xi*) définit l'état de la variable de sortie A.

5.I.3. Table de vérité

Tables de vérité

Une table de vérité est un tableau permettant de décrire toutes les possibilités de sorties en fonction de entrées. On place donc les variables d'entrées dans les colonnes de gauche en les faisant varier de telle façon à couvrir l'ensemble des possibilités. La colonne (ou les colonnes si la fonction a plusieurs sorties) de droite décrit la sortie.

Voici par exemple les tables de vérités des portes logiques:

Fonctions Logique	Entrées		Sortie
	B	A	S
AND $S = A \cdot B$	0	0	0
	0	1	0
	1	0	0
	1	1	1
OR $S = A + B$	0	0	0
	0	1	1
	1	0	1
	1	1	1
NAND $S = \overline{A \cdot B}$	0	0	1
	0	1	1
	1	0	1
	1	1	0
NOR $S = \overline{A + B}$	0	0	1
	0	1	0
	1	0	0
	1	1	0
XOR $S = A \oplus B$	0	0	0
	0	1	1
	1	0	1
	1	1	0
XNOR $S = \overline{A \oplus B}$	0	0	1
	0	1	0
	1	0	0
	1	1	1
INV $S = \overline{IN}$	0		1
	1		0

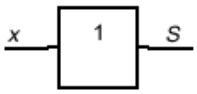
5.2. Les opérations logiques élémentaires

5.2.1. Opérateur OUI

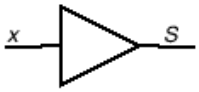
L'opération (ou opérateur) OUI est dite **unaire** (ne s'applique qu'à une seule opérande). Elle affecte à la variable de sortie l'état logique de la variable d'entrée.

Equation : x est la l'entrée, S la sortie : $S = x$.

Symbole (norme IEC[1])



Symbole : norme IEEE[2]



x	S
0	0
1	1

Table de vérité

Diagramme de Venn
(représentation ensembliste)

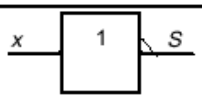
Remarque : les anglo-américains notent H (*High*) le niveau haut et L (*Low*) le niveau bas.

5.2.2. Opérateur NON

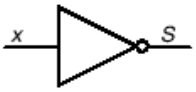
L'opération (ou opérateur) NON est la fonction unaire qui affecte à la variable de sortie l'état complémentaire de la variable d'entrée.

Equation : x est la l'entrée, S la sortie, $S = \overline{x}$ (prononcer « x barre »).

Symbole (norme IEC)



Symbole (norme IEEE)



x	S
0	1
1	0

Table de vérité

Diagramme de Venn

5.2.3. Opérateur ET

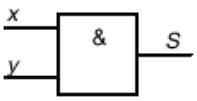
L'opération ET est le **produit logique**. Le signe est celui de la multiplication (un point), mais on lit « et ». C'est un opérateur **binaire** qui affecte à la variable de sortie l'état 1 si et seulement si les variables d'entrée sont à 1 simultanément.

Equation : x et y les entrées, S la sortie, $S = x.y = xy$.

On note aussi l'opération ET par un V retourné :

$x.y = x \wedge y$ (penser à l'intersection d'ensembles).

Symbole (norme IEC)



Symbole (norme IEEE)

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité

Diagramme de Venn

5.2.4. Opérateur OU

L'opération OU est la **somme logique**. Le signe est celui de l'addition (+), mais on lit « ou ». C'est un opérateur binaire qui affecte à la variable de sortie l'état 1 si et seulement si une variable d'entrée est à 1. Cette définition induit directement le symbole ≥1.

Equation : x et y les entrées, S la sortie, $S = x.y = xy$.

On note aussi l'opération OU par un V :

$x.y = x \vee y$ (penser à l'union d'ensembles).

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Table de vérité

Symbole (norme IEC)

Symbole (norme IEEE)

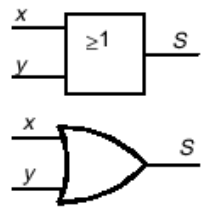


Diagramme de Venn

5.2..5. Remarques et compléments

Il est possible d'étendre la notion d'opération logique en utilisant des concepts plus « algébriques » :

- pour le NON logique : $\bar{x} = 1 - x$ avec $x \in \{0, 1\}$,
- pour le ET logique : $x.y$ = Min(x,y) avec $(x,y) \in \{0, 1\}\acute{}\{0, 1\}$,
- pour le OU logique : $x + y$ = Max(x,y) avec $(x,y) \in \{0, 1\}\acute{}\{0, 1\}$,

Ces notations sont aisément vérifiables à l'aide de tables de vérité.

6. Les opérations logiques induites[4]

6.1. L'opération NON ET ou NAND

Cette fonction logique est le résultat de l'association d'un NON et d'un ET. C'est un opérateur binaire qui affecte à la variable de sortie l'état 0 si et seulement si les variables d'entrée sont à 1 simultanément

Equation : x et y les entrées, S la sortie, $S = \overline{x.y}$.

On note aussi l'opération NAND par une flèche montante :

$S = \overline{x.y} = x \uparrow y$ (penser $\wedge$).

x	y	$x \uparrow y$
0	0	1
0	1	1
1	0	1
1	1	0

Table de vérité

Symbole (norme IEC)

Symbole (norme IEEE)

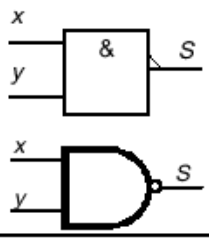


Diagramme de Venn

6.2. L'opération NON OU ou NOR

Cette fonction logique est le résultat de l'association d'un NON et d'un OU. C'est un opérateur binaire qui affecte à la variable de sortie l'état 1 si et seulement si les variables d'entrée sont à 0 simultanément.

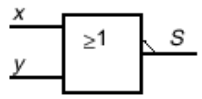
Equation : x et y les entrées, S la sortie, $S = \overline{x+y}$.

On note aussi l'opération NOR par une flèche descendante : $S = \overline{x+y} = x \downarrow y$ (penser $\vee$).

x	y	$x \downarrow y$
0	0	1
0	1	0
1	0	0
1	1	0

Table de vérité

Symbole (norme IEC)



Symbole (norme IEEE)

Diagramme de Venn

6.3. L'opération OU EXCLUSIF ou XOR

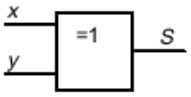
Cet opérateur logique binaire ne prend la valeur 1 que si une seule des entrées est à 1.

Equation : x et y les entrées, S la sortie, $S = x \oplus y$.

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité

Symbole (norme IEC)



Symbole (norme IEEE)

Diagramme de Venn

Généralisation

L'opérateur XOR se généralise à un ensemble de *n* variables d'entrée par la définition suivante :

La sortie vaut 1 si et seulement si le nombre d'entrées à 1 est impair.

Cet opérateur peut donc aisément faire fonction de contrôleur de parité (ou d'imparité).

7. Les expressions logiques et leur simplification [12]

Tous les opérateurs précédents permettent de combiner des variables pour en construire de nouvelles.

Exemple : $\left.\begin{array}{l} c = a + b.d \\ d = e + f \end{array}\right\} \Rightarrow c = a + b.(e + f)$

7.1. Propriétés

- Commutativité

  $a + b = b + a$ (commutativité de l'opération OU)

  $a.b = b.a$ (commutativité de l'opération ET)

- Associativité

  $a + (b + c) = (a + b) + c = a + b + c$ (associativité de l'opération OU)

  $(ab)c = a(bc) = abc$ (associativité de l'opération ET)

- Distributivité

  $(a + b).c = ac + bc$ (distributivité du produit logique sur la somme logique)

  $ab + c = (a + c).(b + c)$ (distributivité de la somme logique sur le produit logique)

Les parenthèses imposent une priorité supérieure.

7.2. Autres propriétés

a est une variable logique

$a+\overline{a} = 1$	$a + 0 = a$	$a + 1 = 1$	$a + a = a$
$a.\overline{a} = 0$	$a.0 = 0$	$a.1 = a$	$a.a = a$

7.3. Autres propriétés

a est une variable logique

$a+\overline{a} = 1$	$a + 0 = a$	$a + 1 = 1$	$a + a = a$
$a.\overline{a} = 0$	$a.0 = 0$	$a.1 = a$	$a.a = a$

7.4. Exemples

$f = a + \overline{a}.b = a + b$

$abc + \overline{a}bc + a\overline{b}c + ab\overline{c} = bc(a + \overline{a}) + a\overline{b}c + ab\overline{c} = c(b + a\overline{b}) + ab\overline{c} = ac + bc + ab\overline{c} = ab + bc + ac$

$(a + b).c + (a + c)ab + b\overline{c} + a = ac + bc + ab + abc + b\overline{c} + a = a(c + b + bc + 1) + bc + b\overline{c} = a + b$

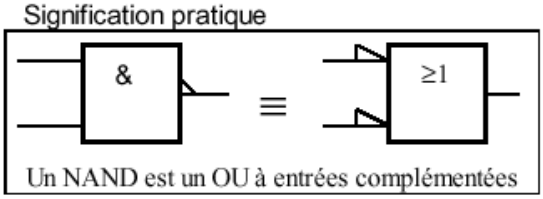
7.5. Théorèmes de De Morgan 3

But : exprimer les opérateurs ET, OU et NON exclusivement à l'aide d'opérateurs NOR seuls ou
NAND seuls. On dit que les opérateurs NOR et NAND sont **universels** ou **complet**s.

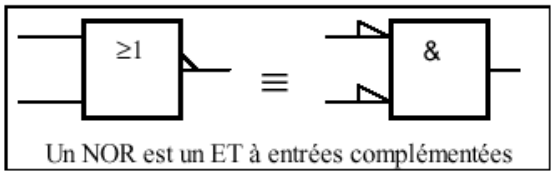
Premier théorème : $\overline{a + b} = \overline{a}\,\overline{b} \rightarrow$ généralisation $\boxed{\overline{\sum_{i=1}^{n} a_i} = \prod_{i=1}^{n} \overline{a_i}}$ où les les a_i sont les variables, $i \in [1,n]$.

Second théorème : $\overline{ab} = \overline{a} + \overline{b} \rightarrow$ généralisation $\boxed{\overline{\prod_{i=1}^{n} a_i} = \sum_{i=1}^{n} \overline{a_i}}$ où les les a_i sont les variables, $i \in [1,n]$.

Signification pratique



Un NAND est un OU à entrées complémentées



Un NOR est un ET à entrées complémentées

Expression des opérateurs de base à l'aide des seuls opérateurs universels.

- $\overline{a} = \overline{a.a} = a \uparrow a = \overline{a+a} = a \downarrow a$ — Opérateur NON réalisé avec un NAND et avec un NOR
- $a.b = \overline{\overline{a.b}} = \overline{\overline{a}+\overline{b}} = \overline{a} \downarrow \overline{b}$ — Trois opérateurs NOR (dont deux en NON)

  $a.b = \overline{\overline{a.b}} = \overline{a \uparrow b}$ — Deux opérateurs NAND (dont un en NON) *Cas trivial*
- $a+b = \overline{\overline{a+b}} = \overline{\overline{a}.\overline{b}} = \overline{a} \uparrow \overline{b}$ — Trois opérateurs NAND (dont deux en NON)

  $a+b = \overline{\overline{a+b}} = \overline{a \downarrow b}$ — Deux opérateurs NOR (dont un en NON) *Cas trivial*

7.6. Synthèse

Lors de la synthèse de circuits logiques, l'un des problème fréquemment rencontré consiste à réaliser une fonction logique à partir d'une table de vérité ou d'un cahier des charges. Ce résumé présente brièvement quelques méthodes permettant de résoudre ce problème.

Synthèse à partir de la table de vérité

En partant de la table de vérité, il est relativement aisé de définir la fonction logique. En effet, en raisonnant sur les valeurs «1» de la table, il suffit d'additionner (addition logique, donc fonction OU) les termes considérés.

L'expression des termes est définie par le produit (multiplication logique, donc fonction ET) des variables d'entrées ou de leur complément considérés à la valeur logique «1».

L'exemple ci-dessous illustre ceci pour la détection de majorité sur trois variables.

Exemple : fonction majorité à trois variables

La fonction de majorité utilisée ici est définie comme vrai (état «1») si au moins deux des trois variables d'entrées (c, b, a) sont à l'état «1».

La table de vérité est la suivante :

On en déduit la fonction y

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

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

8. Ecriture des fonctions booléennes [1],[13]

8.1. Définitions

On appelle **min terme** de *n* variables, un produit logique de ces dernières (complémentées ou non).

Avec *n* variables, on construit 2^n min termes, c'est-à-dire autant que de combinaisons possibles de *n* éléments prenant deux états.

Exemple : pour 2 variables *a* et *b*, voici les 4 mintermes : ab, $\overline{a}b$, $a\overline{b}$ et $\overline{a}\overline{b}$.

8.2. Première forme canonique

La **première forme canonique** d'une expression booléenne est composée d'une somme de min termes

exclusivement. Pour une expression donnée cette forme est unique.

Exemple :

Remarque : la somme de tous les min termes de *n* variables vaut toujours 1 puisqu'il existe toujours un min terme de *n* variables valant 1.

8.3. Seconde forme canonique

La **seconde forme canonique** d'une expression booléenne est composée d'un produit de max termes exclusivement. Pour une expression donnée cette forme est unique.

Exemple :

Remarque : Le produit de tous les max termes de *n* variables vaut toujours 0 puisqu'il existe toujours un max terme de *n* variables valant 0.

Pour changer de forme canonique on effectue d'une double complémentation (involution) de l'expression suivie de l'application de l'un des théorèmes de De Morgan.

8.4. Forme canonique décimale

L'écriture des expressions logique a cet inconvénient d'être assez longue. Chaque min terme parmi les

2^n de *n* variables correspond à un nombre représentant son ordre, c'est pourquoi on préfère parfois utiliser

une écriture indiquant la liste classée des numéros des mintermes de la première forme canonique.

Exemple : $F = abcd + \overline{a}bc\overline{d} + a\overline{b}c\overline{d} + \overline{a}\overline{b}\overline{c}\overline{d}$ peut aussi s'écrire $F(a, b, c, d) = \Sigma\ 0, 6, 10, 15.$

8.5. Extraction d'une équation logique à partir d'une table de vérité

Une table de vérité recense l'ensemble des états d'une sortie pour **toutes** les combinaisons possibles des variables d'entrée.

Pour trouver une expression sous la première forme canonique, on applique la méthode suivante :

- on définit les mintermes de *n* variables qui sont les expressions logiques bâties sur la combinaison de ces *n* variables ;
- chaque minterme est associé à l'une des combinaisons de la table de vérité (en conservant la correspondance 1 pour la variable et 0 pour la variable complémentée),
- tous les mintermes valant 1 sont sommés logiquement pour obtenir l'expression de la sortie.
- Les simplifications sont effectuées par les procédés de calcul algébrique.

$F_1 = ab + \bar{c} + c(\bar{a} + \bar{b})$

$F_2 = (a+b+c)(\bar{a}+b+c)+ab+bc$

$F_3 = (x\bar{y}+z)(x+\bar{y})z$

$F_4 = (\bar{a}b + a\bar{b})(ab+\bar{a}\bar{b})$

$F_5 = \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}d + a\bar{b}cd + \bar{a}\bar{b}\bar{c}\bar{d} + ab\bar{c}d + \bar{a}bcd + a\bar{b}\bar{c}\bar{d}$

Tracer les logigrammes des expressions logiques suivantes :

$S_1 = b\bar{c}\bar{d} + ab\bar{d} + \bar{a}bc\bar{d}$ $\qquad$ $S_2 = x + \bar{y}z + xy\bar{t} \oplus \bar{z}yt$

Chronogrammes

Dessiner la forme d'onde en sortie S du système logique de la ***Figure 1***.

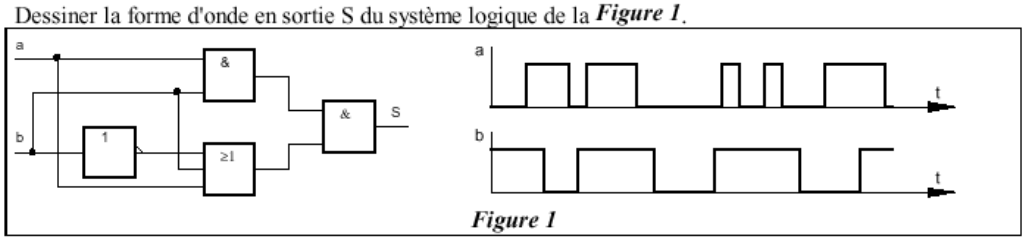


Figure 1

Trois interrupteurs a, b et c commandent l'allumage de deux lampes R et S suivant les conditions :

- dès qu'un ou plusieurs interrupteurs sont activés la lampe R s'allume,
- la lampe S ne doit s'allumer que si au moins deux interrupteurs sont activés.

Trouver les expressions de R et S et dessiner les logigrammes.

9- *Tables de Karnaugh [6]*

La fonction obtenue par la méthode précédente n'est pas obligatoirement optimale dans le sens qu'elle ne fait pas nécessairement intervenir un nombre minimum d'opérations logiques. Ce point est important pour toute réalisation (discrète, intégrée ou encore logiciel) car le nombre d'opérations logiques fixe la complexité du système, en nombred'éléments, en surface de silicium ou en temps de calcul. Il est donc intéressant de définir une méthode qui permette de simplifier l'expression obtenue au paragraphe 2. On peut simplifier une équation à l'aide de l'algèbre de Boole, mais il existe une méthode plus simple : les tables de Karnaugh. On se limitera à trois et quatre variables d'entrées.

9.1.Méthode

• On introduit la table de vérité dans une table de Karnaugh. Cette table est une représentation bidimentionnelle de la table de vérité. La figure 1.a illustre ceci pour les trois variables c, b et à. La figure 1.b illustre la table de Karnaugh pour les quatre variables d, c, b, et a.

c \ ba	00	01	11	10
0				
1				

Figure 1.a

dc \ ba	00	01	11	10
00				
01				
11				
10				

Figure 1.b

• On recherche ensuite visuellement les blocs (ensembles de cases adjacentes) de dimensions un, deux, quatre ou huit qui contiennent la valeur «1».
• On élimine tout bloc déjà contenu dans un ou plusieurs blocs plus grands.
• La fonction simplifiée se déduit de l'expression logique des blocs restants.
Remarque : Les blocs de dimension 1 s'expriment par le produit des 3 variables dans le cas d'un système à trois variables. Ces blocs s'expriment par le produit des quatre variables dans le cas d'un système à quatre variables.
Les blocs de dimension 2 s'expriment par le produit de 2 variables dans le cas d'un système à trois variables. Ces blocs s'expriment par le produit des trois variables dans le cas d'un système à quatre variables. Les blocs de dimension 4 s'expriment par l'une des trois variables dans le cas d'un système à trois variables. Ces blocs s'expriment par le produit de deux variables dans le cas d'un système à quatre variables.
Les blocs de dimension 8 s'expriment par l'une des quatre variables dans le cas d'un système à quatre variables.

Exemple : fonction majorité à trois variables
La table de vérité du paragraphe 9.1 permet d'écrire la table de Karnaugh de la figure 9.2 :

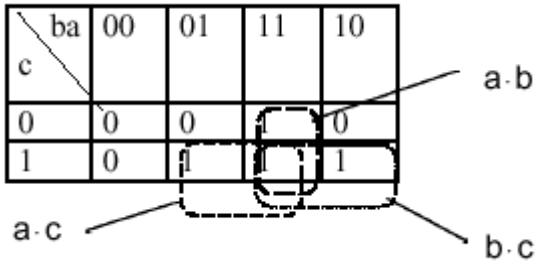


Figure 9.2
Remarque : Les blocs à une dimension n'ont pas été représentés pour ne pas surcharger la figure.
On en déduit la fonction simplifiée :
y =a b+ a c+b c
Cette expression nécessite 5 opérations logiques contre 11 pour l'expression précédente.

Exemple 2
on considère la fonction logique donnée par la table de vérité suivante:

A	B	C	D	x
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

sans aucune simplification, on voit que:

$$x = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D} + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.C.\overline{D} + A.\overline{B}.C.D + A.B.C.\overline{D} + A.B.C.D$$

Cette expression, lourde à gérer peut être simplifiée.
En utilisant la méthode de Karnaugh, on dresse le tableau suivant (on remarque l'ordre de rentrée des variables qui rend 2 à 2 adjacentes des cases ayant un côté commun ou situées au bord du tableau sur la même ligne ou la même colonne):

C D \ A B	0 0	0 1	1 1	1 0
0 0	1	0	0	1
0 1	0	0	0	0
1 1	0	0	1	1
1 0	1	0	1	1

On va alors chercher à faire des regroupements les plus importants possibles séries de cases 2 à 2 adjacentes comportant des 1 (qui peuvent être intégrés à plusieurs regroupements). Quand le système a quatre entrées, un regroupement de deux cases fait intervenir 3 variables, un regroupement de 4 en fait intervenir 2…etc.
Ici, on constate que l'on peut effectuer 2 regroupements de cases adjacentes intéressants (ils regroupent tous les deux quatre cases).

On a donc $x = \overline{B}.\overline{D} + A.C$ ce qui est beaucoup plus simple que la première expression proposée (le premier terme correspond au premier regroupement et l'autre au second regroupement).

premier regrouppement intéressant

C D \ A B	0 0	0 1	1 1	1 0
0 0	1	0	0	1
0 1	0	0	0	0
1 1	0	0	1	1
1 0	1	0	1	1

second regroupement intéressant

C D \ A B	0 0	0 1	1 1	1 0
0 0	1	0	0	1
0 1	0	0	0	0
1 1	0	0	1	1
1 0	1	0	1	1

exercice:
On donne la table de vérité suivante:

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

en simplifiant par la méthode de Karnaugh, on trouve $x = \overline{A}.B + B.\overline{C} + \overline{A}.C.D$.

9.2. Conclusion.

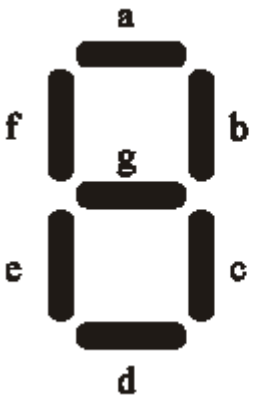
Il faut noter que ces problèmes sont désormais secondaires. En effet, l'utilisation d'un grand nombre de portes pour réaliser des fonctions logiques n'a plus lieux d'être depuis que l'on utilise des composants programmables. Lorsque l'on programme ces composants, la complexité de la relation entre les entrées et la sortie n'a pas une grande importance.

9.3. Conditions facultatives

Sous certaines conditions, il se peut qu'un système soit conçu avec des combinaisons d'entrées qui ne se présentent jamais ou qui sont sans intérêts. On désigne les états de sorties correspondantes par des conditions facultatives *(don't car*e) et on les considère comme des "1" ou des "0", selon ce qui est avantageux. On représente ces conditions facultatives dans les tables par le symbole "X" ou ∀"

Exemple

Les afficheurs à sept segments sont bien connus. Ils servent à afficher un chiffre allant de 0 à 9. Ce nombre est représenté sur 4 bits qui permettent de représenter les nombres de 0 à 15. Les combinaisons d'entrées de 10 à 15 sont des conditions facultatives puisqu'elles ne devraient jamais être présentes.



$$a = \sum(0,2,3,5,6,7,8,9) + \phi(10,11,12,13,14,15)$$

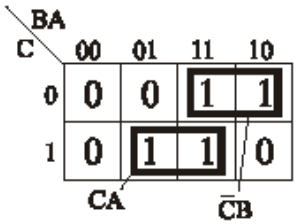
$$b = \sum(0,1,2,3,4,7,8,9) + \phi(10,11,12,13,14,15)$$

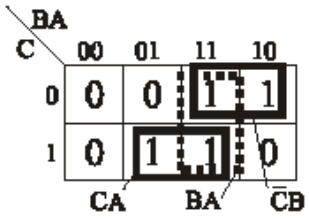
Impliquants redondants

Un impliquant est dit redondant si toutes les fenêtres qu'il couvre dans une table de Karnaugh sont déjà couvert par un autre impliquant. Ce terme peut être enlevé de l'équation sans changer la table de vérité. Sous certaines conditions, ce terme peut stabiliser le circuit en enlevant des erreurs momentanées *(glitch*s) de fonctionnement.

Exemple

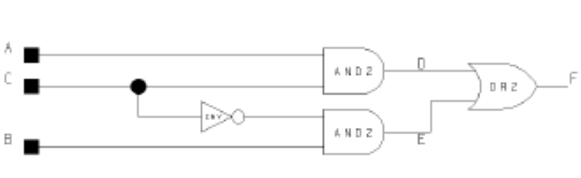
$$F = \overline{C}B\overline{A} + \overline{C}BA + CBA + C\overline{B}A$$

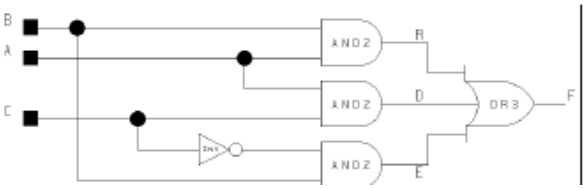




$$F = \overline{C}B + CA$$

$$F = \overline{C}B + CA + BA$$





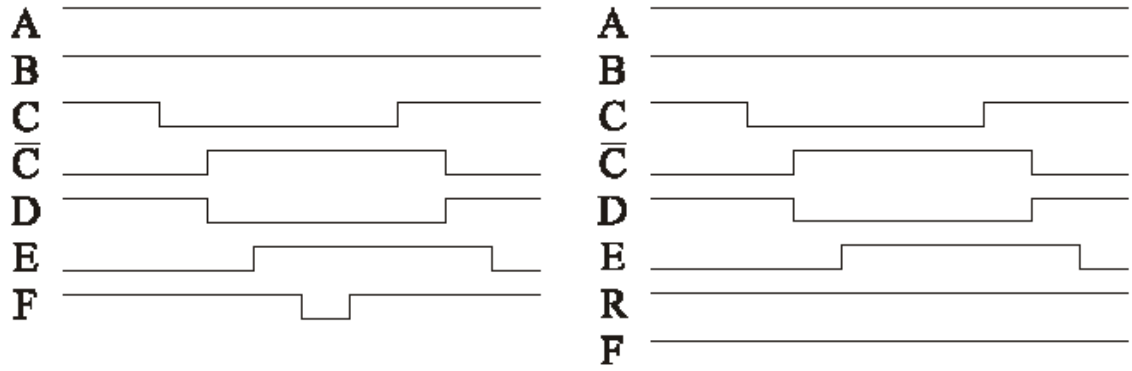
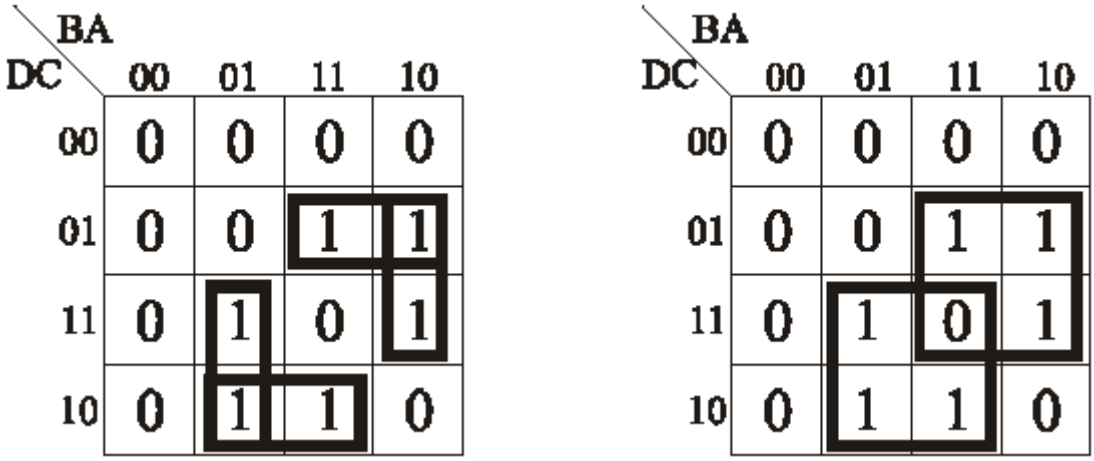


Figure 2.10: Effets d'un impliquant redondant.

Symétrie et pliage

Certaines configurations courantes des tables de Karnaugh permettent de simplifier les équations. La réduction d'équation en utilisant ces configurations, ou des techniques de pliage, est toutefois très intuitive et peut introduire des erreurs lorsqu'elles ne sont pas appliquées avec
attention. Il est préférable de maîtriser les tables de Karnaugh avant d'utiliser ces techniques et configurations.



Symétrie de la table de Karnaugh

$F = D\overline{B}A + D\overline{C}A + \overline{D}CB + CB\overline{A}$

$F = DA(\overline{B} + \overline{C}) + CB(\overline{D} + \overline{A})$

$F = DA\overline{BC} + CB\overline{DA}$

$F = DA \oplus BC$

$F = DA \oplus BC$

C \ BA	00	01	11	10
0	0	1	0	1
1	1	0	1	0

DC \ BA	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	0	0	0
10	0	0	0	0

Symétrie de la table de Karnaugh.

$F = A \oplus B \oplus C$

$F = \overline{D}(A \oplus B \oplus C)$

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	1	0	0	0
10	1	1	0	0

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	1	0	0	0
10	1	1	0	0

Symétrie de la table de Karnaugh.

$F = D\overline{BA} + D\overline{C}\overline{B} + \overline{DC}B + \overline{D}B\overline{A}$

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

$F = (\overline{A} + \overline{C})(B \oplus D)$

$F = \overline{AC}(B \oplus D)$

$F = \overline{AC}(\overline{B}D + B\overline{D})$

$F = \overline{AC}(B \oplus D)$

9.4. Quine/McCluskey

Avec les méthodes précédentes, les simplifications ont été obtenues de façon purement intuitive; rien ne nous assure que la fonction obtenue est réellement la plus simple que l'on peut obtenir. De plus, en l'absence d'algorithme bien défini, ces méthodes ne peuvent être implantées efficacement dans un logiciel.

La méthode Quine/McCluskey, constituée d'une procédure bien définie. garantit une simplification maximale de la fonction obtenue, sous forme de somme de produits. I1 n'existe aucune autre fonction équivalente contenant moins de termes. La méthode de Quine/McCluskey utilise un algorithme pour faire ressortir l'adjacence entre les termes.

La procédure à suivre est la suivante:

1. Mettre la fonction sous forme canonique.
2. Transformer les termes en nombres binaires.
3. Grouper ces nombres selon leur poids (nombre de "l").
4. Placer les nombres par ordre croissant à l'intérieur de chaque groupe.
5. Comparer chaque terme d'un groupe avec chaque terme du groupe suivant: deux termes n'ayant qu'un bit qui ne correspond pas génèrent un nouveau terme où le bit de différence est remplacé par un "X"; les nouveaux termes engendrés forment une liste de nouveaux nombres binaires groupés par poids.
6. Refaire l'étape 5 à partir de la nouvelle liste obtenue jusqu'à ce qu'aucune autre nouvelle liste ne soit générée.
7. Identifier les impliquants, c'est à dire les éléments qui n'ont pas été utilisés pour générer un élément de la nouvelle liste.
8. Identifier les impliquants essentiels, c'est à dire ceux dont la représentation est unique pour certaines solutions.
9. Vérifier si l'ensemble des impliquants essentiels représente toutes les solutions. Si oui, la solution minimale est trouvée. Si non, on doit ajouter un ou plusieurs autres impliquants afin de représenter toutes les solutions. Il n'existe aucune façon précise pour choisir les autres impliquants.

Exemple

On veut simplifier la fonct ion suivante:

$$S = \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\overline{C}D + AB\overline{C}D$$

Étape 1: La fonction est déjà sous forme canonique.

Étape 2: Transformation en nombres binaires.

$$S=0010+0100+0101+0110+0111+1001+1101$$

Étapes 3 et 4: Classification.

Poids 1	0010
	0100
Poids 2	0101
	0110
	1001
Poids 3	0111
	1101

Etapes 5 et 6: Comparaisons.

1	0010	a	0X10 (1-4)	01XX (b-f et c-d)
2	0100	b	010X (2-3)	
	------	c	01X0 (2-4)	
3	0101		-------	
4	0110	d	01X1 (3-6)	
5	1001	e	X101 (3-7)	
	------	f	011X (4-6)	
6	0111	g	1X01 (5-7)	
7	1101			

Étape 7: Identification des impliquants
Les termes qui n'ont jamais engendrés de nouveaux termes sont marqués d'un "*"

0010	0X10 *	01XX *
0100	010X	
------	01X0	
0101	-------	
0110	01X1	
1001	X101 *	
------	011X	
0111	1X01 *	
1101		

Étape 8 et 9: Identification des impliquants essentiels

	0010	0100	0101	0110	0111	1001	1101
0X10	[✓]			(✓)			
X101			✓				✓
1X01						[✓]	(✓)
01XX		[✓]	(✓)	(✓)	[✓]		

✓ indique que l'impliquant couvre le terme.

[✓] indique que l'impliquant est essentiel à ce terme.

(✓) indique un terme couvert par un impliquant essentiel.

Les impliquants essentiels sont: 0X10, 1X01 et 01XX. Puisqu'ils sont suffisants pour représenter toutes les solutions, la fonction simplifiée est:

$$S = \overline{A}C\overline{D} + A\overline{C}D + \overline{A}B$$

Exemple 2.9

On veut simplifier la fonct ion suivante:

$$S = \overline{A}BD + ABCD + A\overline{B}CD + ABC\overline{D} + A\overline{B}C\overline{D}$$

Étape 1: Mettre la fonction sous forme canonique.

$$S = \overline{A}BD(\overline{C} + C) + ABCD + A\overline{B}CD + ABC\overline{D} + A\overline{B}C\overline{D}$$

$$S = \overline{A}B\overline{C}D + \overline{A}BCD + ABCD + A\overline{B}CD + ABC\overline{D} + A\overline{B}C\overline{D}$$

Étape 2: Transformation en nombres binaires.

S=0101+0111+1111+1011+1110+1010

ðtapes 5, 6 et 7: Comparaisons et identification des impliquants.

Il est aussi possible d'indiquer un "✓" lorsque les termes sont utilisés. Les impliquants seront alors les termes sans "✓" .

0101 ✓	01X1	1X1X
1010 ✓	101X ✓	
------	1X10 ✓	
0111 ✓	------	
1011 ✓	X111	
1110 ✓	1X11 ✓	
------	111X ✓	
1111 ✓		

ðtape 8 et 9: Identification des impliquants essentiels

	0101	1010	0111	1011	1110	1111
01X1	[✓]		(✓)			
X111			✓			✓
1X1X		[✓]		[✓]	[✓]	(✓)

Les impliquants essentiels sont: 01X1 et 1X1X. Puisqu'ils sont suffisants pour représenter toutes les solutions, la fonction simplifiée est:

$$S = \overline{A}BD + AC$$

Exemple 2.10

On veut simplifier la fonct ion suivante. Notez les condit ions facultat ives.

$$F = \sum(2,4,7,12,10,15) + \phi\sum(6,9,11,14)$$

ðtape 1: La fonction est sous forme canonique.

ðtape 2: Transformation en nombres binaires.

$$F=0010+0100+0111+1100+1010+1111+\varnothing(0110+1001+1011+1110)$$

ðtapes 3 et 4: Classification.

ð partir de cette étape, les condit ions facultat ives sont traitées comme les autres termes.

Poids 1	0010
	0100
Poids 2	0110
	1001
	1010
	1100
Poids 3	0111
	1011
	1110
Poids 4	1111

Étapes 5, 6 et 7: Comparaisons et identification des impliquants.

0010 ✓	0X10 ✓	XX10
0100 ✓	X010 ✓	X1X0
------	01X0 ✓	--------
0110 ✓	X100 ✓	X11X
1001 ✓	-------	1X1X
1010 ✓	011X ✓	
1100 ✓	X110 ✓	
------	101X ✓	
0111 ✓	1X10 ✓	
1011 ✓	10X1	
1110 ✓	11X0 ✓	
------	-------	
1111 ✓	X111 ✓	
	1X11 ✓	
	111X ✓	

Étape 8 et 9: Identification des impliquants essentiels

Notez que seulement les termes essentiels sont inscrits. L'utilité des condit ions facultatives est terminée puisqu'elle se limite à permettre une plus grande simplification.

	0010	0100	0111	1100	1010	1111
10X1						
XX10	[✓]				(✓)	
X1X0		[✓]		[✓]		
X11X			[✓]			(✓)
1X1X					✓	✓

Les impliquants essentiels sont: XX10, X1X0 et X11X. Puisqu'ils sont suffisants pour représenter toutes les solutions, la fonction simplifiée est:

$$S = B\overline{A} + C\overline{A} + CB$$

Généralement, on utilisera:
. La méthode algébrique pour des fonctions à deux variables ou des fonctions de plus de deux variables mais comportant peu de termes sous forme canonique.
. La méthode de Karnaugh pour des fonctions de trois, quatre, cinq ou six variables.
. La méthode de Quine/McCluskey pour des fonctions à cinq variables ou.

9.5.Table de Karnaugh à variable inscrite

La table de Karnaugh à variable inscrite (KVI) est née de la difficulté de réduire les fonctions de plus de quatre variables en utilisant la table de Karnaugh normale. Cette méthode est cependant plus complexe et beaucoup d'expérience avec la table de Karnaugh est un atout majeur. Il est préférable d'avoir complété la série d'exercices sur Karnaugh avant de débuter KVI.
L'idée derrière KVI est d'ajouter un niveau d'abstraction sur la table de vérité de la fonct ion à réduire. Le résultat n'est pas seulement donné en terme de "0" et de "1" mais en fonction d'une, ou plusieurs, variable d'entrée. La table de vérité suivante est transformée afin d'inscrire la variable C.

A	B	C	F	
0	0	0	0	0
0	0	1	0	
0	1	0	1	$\overline{C}$
0	1	1	0	
1	0	0	0	C
1	0	1	1	
1	1	0	1	1
1	1	1	1	

A	B	F
0	0	0
0	1	$\overline{C}$
1	0	C
1	1	1

La KVI est ensuite construite de la même façon que la table de Karnaugh. La seule différence est que les fenêtres peuvent maintenant contenir des variables en plus des "0" et "1".

Les groupements se font maintenant entre termes identiques. Notez qu'il est possible de grouper un "C" avec un "1" car le "1" contient un $"C"\ (1 = C + C)$ Tout comme la Karnaugh ordinaire où

il était très important d'utiliser tous les "1", dans la KVI, il faut absolument utiliser toutes les combinaisons de la variable inscrite. Si un "C" a été groupé avec un "1", il faut utiliser le $"\overline{C}"$ qui fait partie du "1", soit comme $"\overline{C}"$ soit en utilisant le "1".

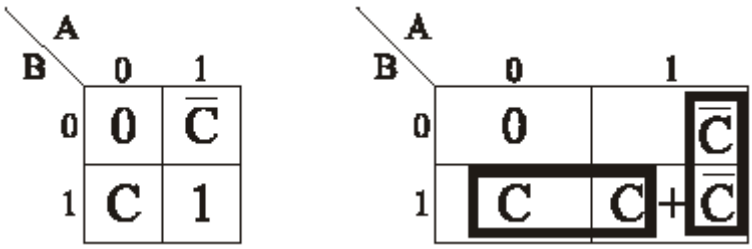


Figure 2.14: Table de Karnaugh à variable inscrite.

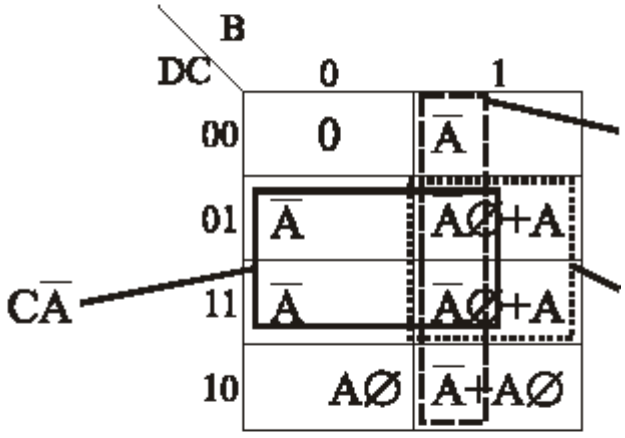
$F = CB + \overline{C}A$

Exemple

Simplifier la fonction suivante:

$$F(D,C,B,A) = \sum(2,4,7,12,10,15) + \phi\sum(6,9,11,14)$$

D	C	B	A	F	
0	0	0	0	0	
0	0	0	1	0	0
0	0	1	0	1	
0	0	1	1	0	$\overline{A}$
0	1	0	0	1	
0	1	0	1	0	$\overline{A}$
0	1	1	0	∅	
0	1	1	1	1	$A + \overline{A}\phi$
1	0	0	0	0	
1	0	0	1	∅	$A\phi$
1	0	1	0	1	
1	0	1	1	∅	$\overline{A} + A\phi$
1	1	0	0	1	
1	1	0	1	0	$\overline{A}$
1	1	0	1	∅	
1	1	1	1	1	$A + \overline{A}\phi$

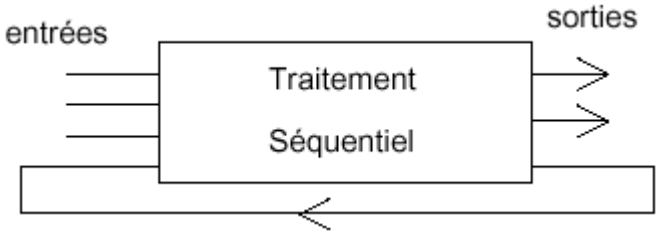


$$F = C\overline{A} + B\overline{A} + CB$$

10. SYSTEMES SEQUENTIELS [4]

10.1. Système séquentiel:

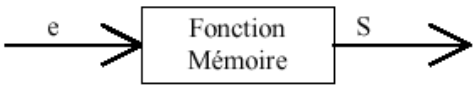
Un système est dit séquentiel, lorsque la ou les sorties dépendent de la combinaison des entrées et de l'état précédent des sorties.

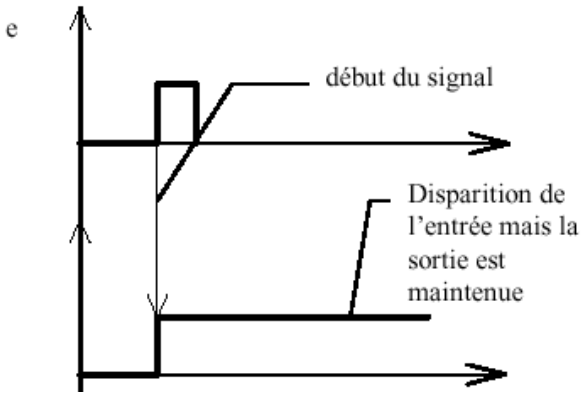


Une même cause (même combinaison des entrées) peut produire des effets différents.
le temps peut être une cause déclenchante.
L'effet peut persister si la cause disparaît.

10.2. Prise en compte du temps

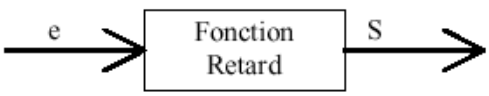
10.2.1. Fonction mémoire

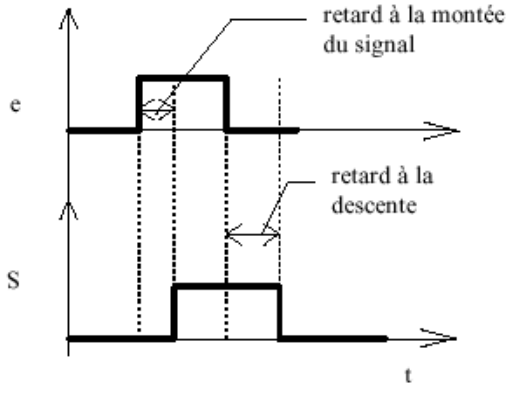




à l'apparition du signal e, la sortie change d'état, à la disparition du signal la sortie reste dans le même état.
Le maintien de la sortie est l'effet mémoire.

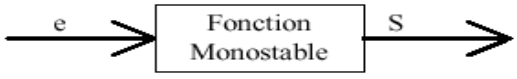
10.2.2. Fonctions retard (s), temporisation;

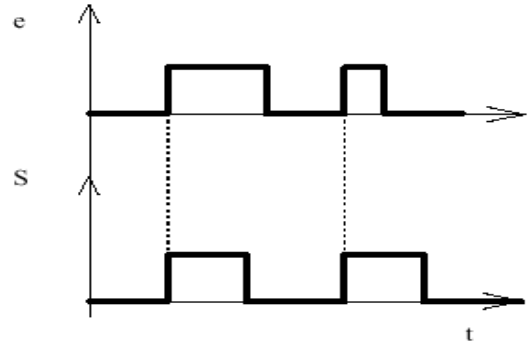




A l'apparition du signal e, la sortie S ne change d'état qu'au bout d'un certain temps t_1, à la disparition du signal la sortie reste dans le même état pendant le temps t_2.

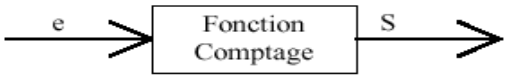
10.2.3. Fonction monostable;

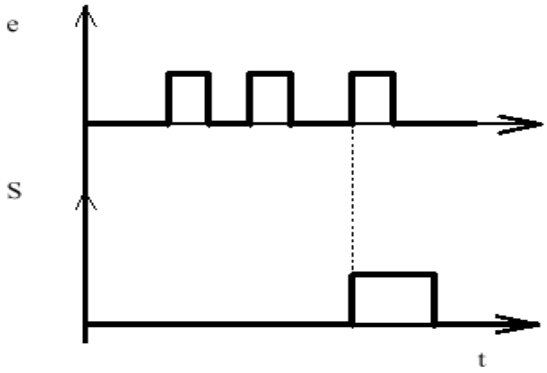




Quelle que soit la durée du signal d'entrée, la sortie a toujours la même durée.

10.2.4 .Fonction comptage

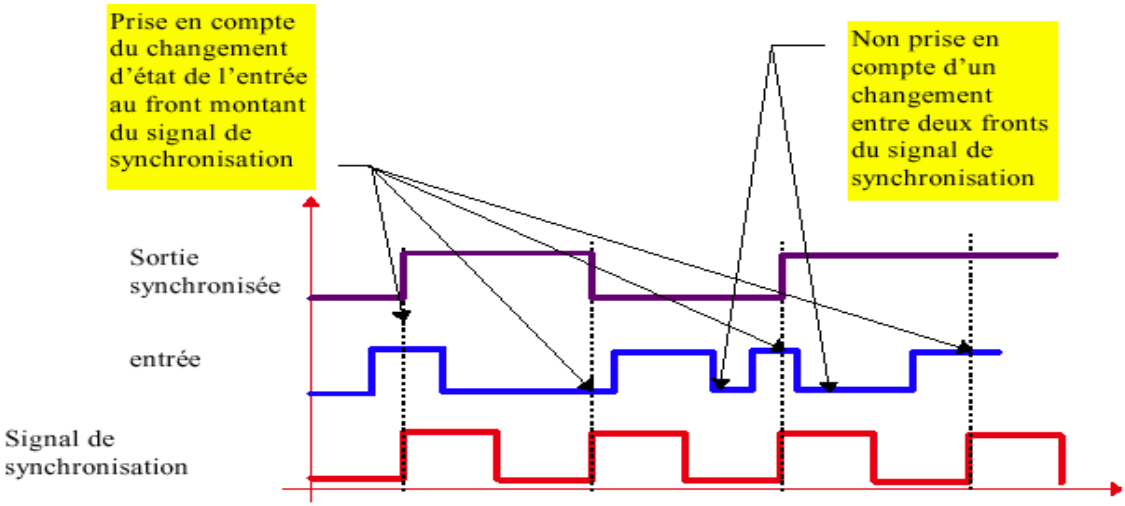




La sortie change d'état lorsque l'entrée a changée d'état le nombre de fois prédéfini. La sortie est indépendante de l'intervalle entre deux changements mais seulement du nombre de changement.

10.2.5. Fonctionnement synchrone ou asynchrone

Un fonctionnement est dit synchrone à un événement extérieur, lorsque la prise en compte de
l'évolution des entrées ne s'effectue qu'a des instants précis, un fonctionnement est dit
asynchrone lorsque cette prise en compte est effective dès le changement d'état.



Cette notion de synchronisation est surtout utilisée dans le fonctionnement des bascules et
constituants mémoires pour synchroniser plusieurs composant entre eux.

10.3. Fonction mémoire

La plupart des traitements ne sont pas uniquement combinatoires mais souvent séquentiels.
Dans un traitement séquentiel le système doit pouvoir mémoriser certaines valeurs pour
pouvoir les réutiliser.
Une bascule est un composant qui permet de réaliser la fonction Mémoire.

10.3.1. Fonction mémoire à effacement prioritaire

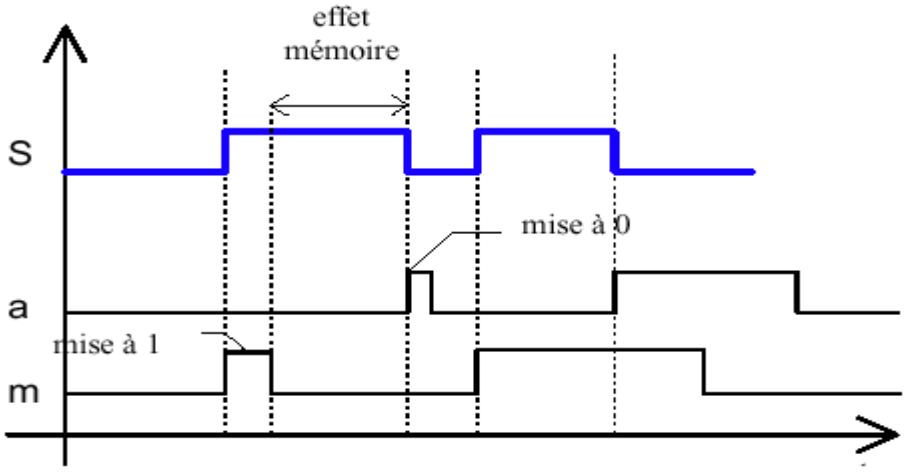
a) fonctionnement

Le système comporte deux entrées m (marche) et a (arrêt). un appui sur m active la sortie S, si on relâche l'entrée, la sortie reste à l'état 1, si on appuie sur a, quel que soit l'état de la sortie et de m, la sortie passe à l'état 0.

b) table de vérité

a	m		S
0	0		0
0	1		1
0	0		1
1	0		0
1	1		0

c) Chronogramme



d) Analyse du fonctionnement

On utilise pour décrire le fonctionnement une variable intermédiaire (variable d'état) *S t* ..qui

représente l'état de S. à l'instant précédent.

e) Table de vérité

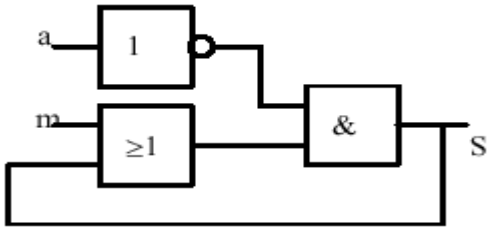
a	m	$S-\Delta t$	S
0	0	0	0
0	1	0	1
0	1	1	1
0	0	1	1
1	0	1	0
1	0	0	0
1	1	0	0
1	1	1	0

f) Equation

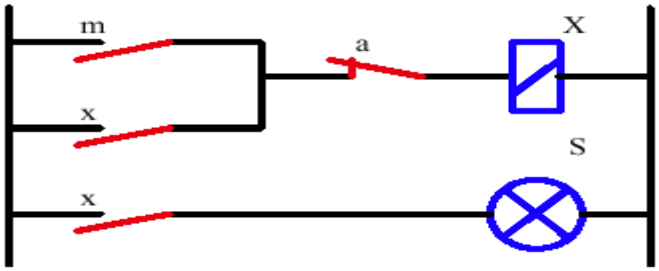
En fait si on suppose que (temps de réaction nul) on a .*t* =0 *S t S* .=.

S =(m+S).a

g) Schéma logique –



h) Schéma électrique, 'relais auto – maintenu



10.3.2.. Fonction mémoire à écriture prioritaire

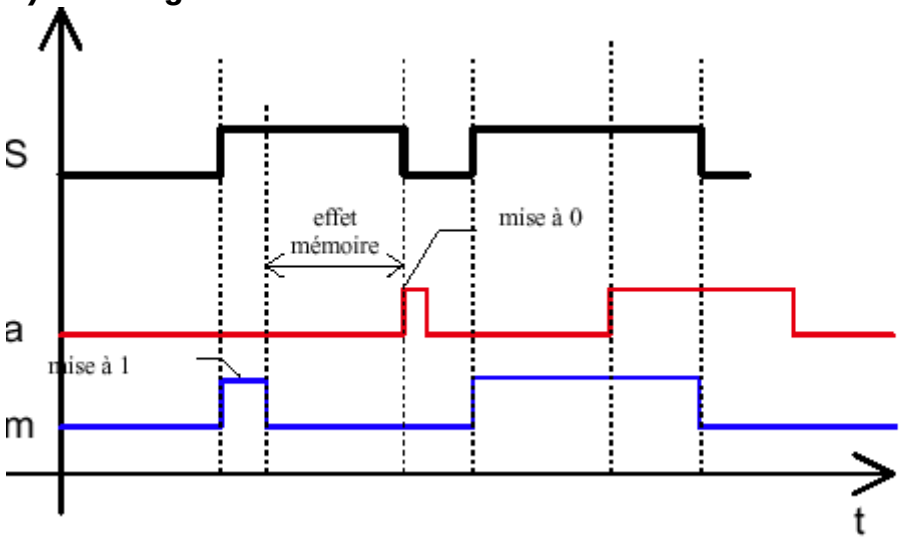
a) fonctionnement

le système comporte deux entrées m (marche) et a (arrêt). un appui sur m active la sortie S, si on relâche l'entrée, la sortie reste à l'état 1. Si on appuie sur a, quel que soit l'état de la sortie, la sortie passe à l'état 0 sauf si m=1, la sortie reste alors à 1.

b) table de vérité

a	m		S
0	O		O
0	1		1
0	0		1
1	0		0
1	1		1

c) Chronogramme



d) Analyse du fonctionnement

On utilise pour décrire le fonctionnement une variable intermédiaire *S t* ..qui représente l'état
précédent de S.

e) Table de vérité

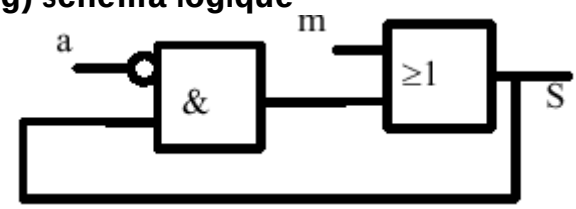
a	m	$S - \Delta t$	S
0	0	0	O
0	1	0	1
0	1	1	1
0	0	1	1
1	0	1	0
1	0	0	0
1	1	0	1
1	1	1	1

f) Equation

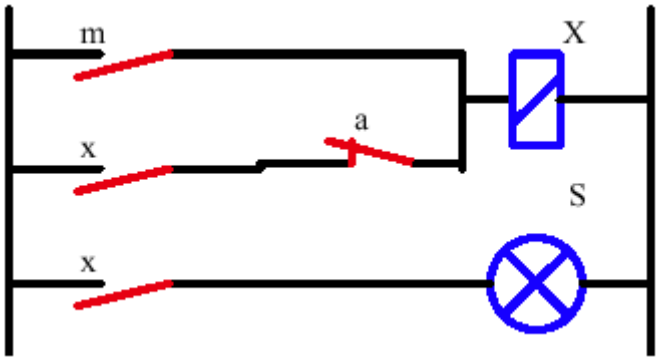
En fait si on suppose que (temps de réaction nul) on a . $S - \Delta t = S$

$$S = m + \bar{a}.S$$

g) schéma logique



h) Schéma électrique, (relais auto-maintenu)

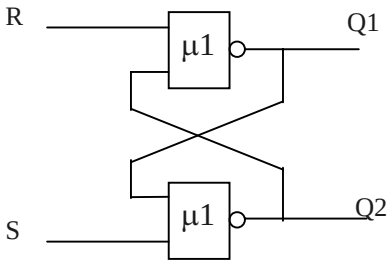


10.4. Etude des bascules.

La bascule est un circuit bistable pouvant prendre deux états logiques "0" ou "1" . L'état de la bascule peut être modifié en agissant sur une ou plusieurs entrées. Le nouvel état de la bascule dépend de l'état précédent, c'est l'élément de base des circuits séquentiels. La bascule peut conserver son état pendant une durée quelconque, elle peut donc être utilisée comme mémoire.

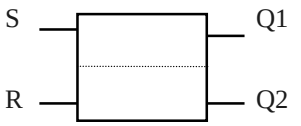
10.4.1. Bascules R S et $\overline{R}$ $\overline{S}$

a) Principe



R	S	Q1	Q2
0	0	x	$\overline{x}$
0	1	1	0
1	0	0	1
1	1	0	0

Interdit



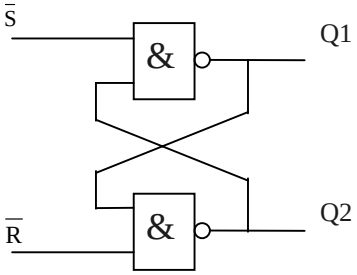
Niveau « 1 » actif

S : Set = mise à un. . .
R : Reset = mise à zéro. . .

Q1 est forcé à un par .S . .
Q2 est forcé à un par .R . .

Autre montage : Bascule $\overline{R}$ $\overline{S}$

Application : Anti-rebond.

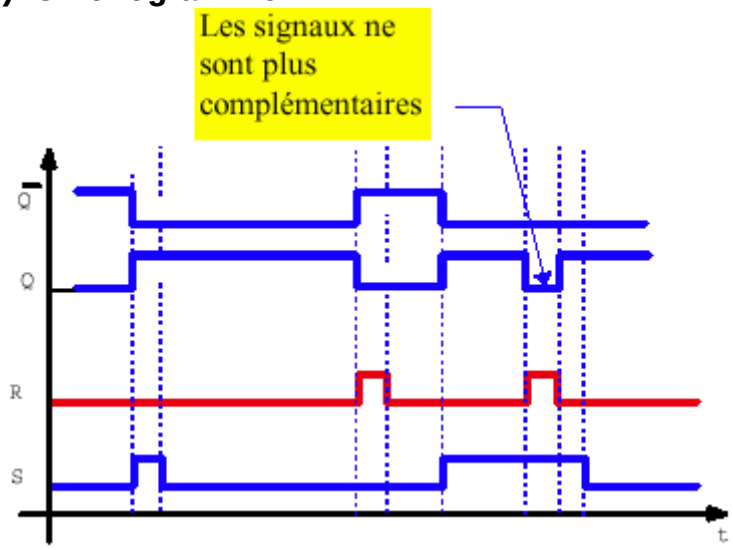


R	S	Q1	Q2
0	0	1	1
0	1	0	1
1	0	1	0
1	1	x	$\overline{x}$

Interdit

Niveau « 0 » actif

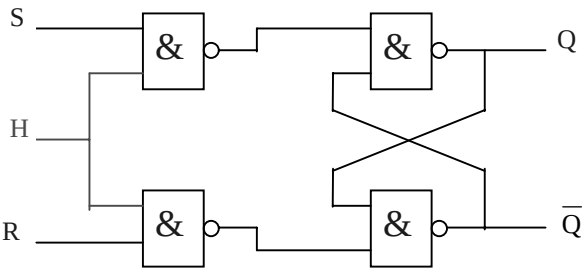
b) Chronogramme

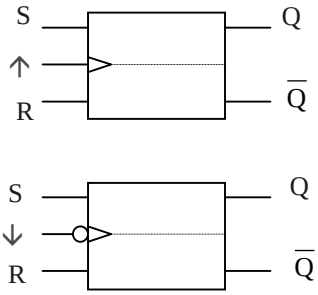


10.4.2. Bascule R S H (Bascule synchrone).

C'est une bascule R S dont la prise en compte de l'état des entrées est synchronisée par une impulsion d'horloge. Ceci permet d'éviter l'arrivée accidentelle de "zéro" sur R ou sur S.

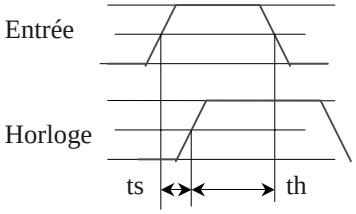
Lorsque H = .0. il y a mémorisation de l'état précédent.





Signal d'horloge: Une bascule synchronisée peut être déclenchée sur le front montant ↑ ou sur le front descendant ↓ de l'impulsion d'horloge.

De plus, afin d'obtenir un fonctionnement correct, le constructeur indique des temps à respecter.



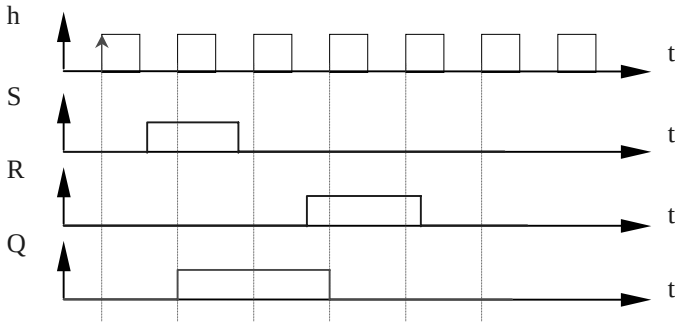
ts : temps de stabilisation .

th : temps de maintien (holding time)

Table de vérité

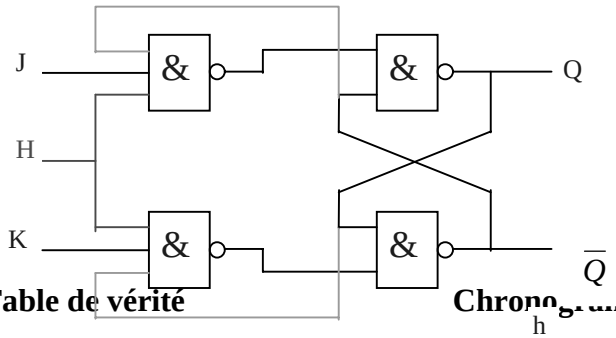
R	S	Q	$\overline{Q}$
0	0	x	$\overline{x}$
0	1	1	0
1	0	0	1
1	1	Interdit	

Chronogramme



10.4.3. Bascule J K synchrone.

La bascule J K synchrone (simple étage) est obtenue à partir d'une bascule R S H dont les sorties sont rebouclées sur les entrées. Ceci permet d'éliminer l'état indéterminé.



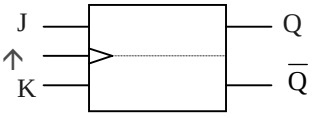
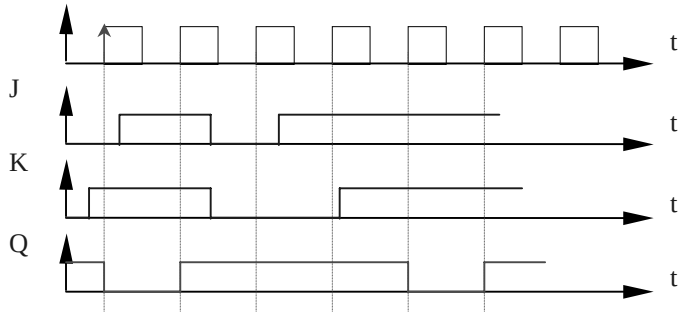


Table de vérité **Chronogramme**

h

R	S	Q	$\overline{Q}$
0	0	x	$\overline{x}$
0	1	1	0
1	0	0	1
1	1	$\overline{x}$	x



Remarque: Pour J = K = .1. , on dit que l'on est dans le mode basculement et l'on définit la bascule « T »(Toggle). Cette bascule passe à l'état opposé à chaque signal d'horloge.

Attention: Les montages que nous avons vus sont des montages de principe qui permettent de comprendre le fonctionnement mais ils ne répondent pas à l'exigence « déclenchement sur front ».
Les bascules déclenchées sur front possèdent un circuit détecteur de front qui permet leur basculement uniquement sur un front montant ou un front descendant.

10.4.4. Bascule D.

A) Bascule D synchrone.

Une bascule D est réalisée à partir d'une bascule R S ou J K dont les entrées sont reliées par un inverseur. Ceci impose donc que les entrées prennent des états complémentaires.

Réalisation:

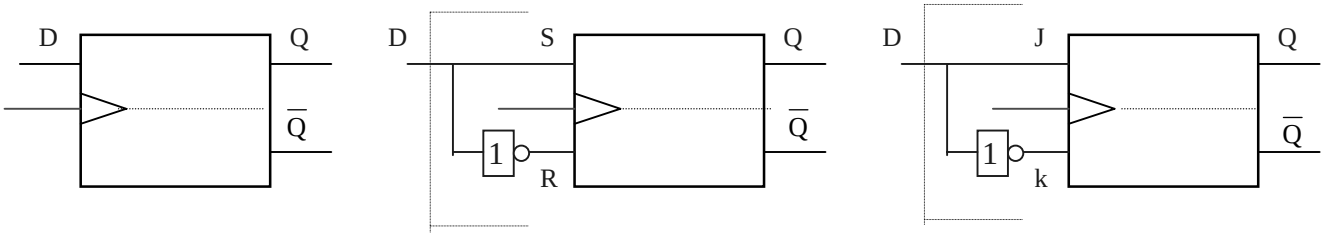
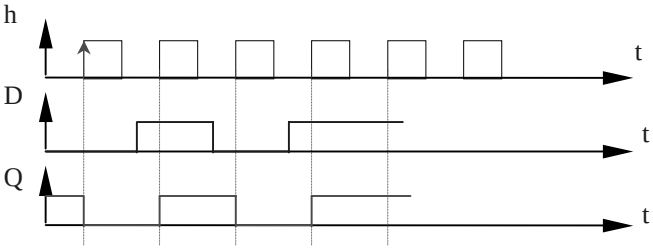


Table de vérité

D	Q
0	0
1	1

Chronogramme

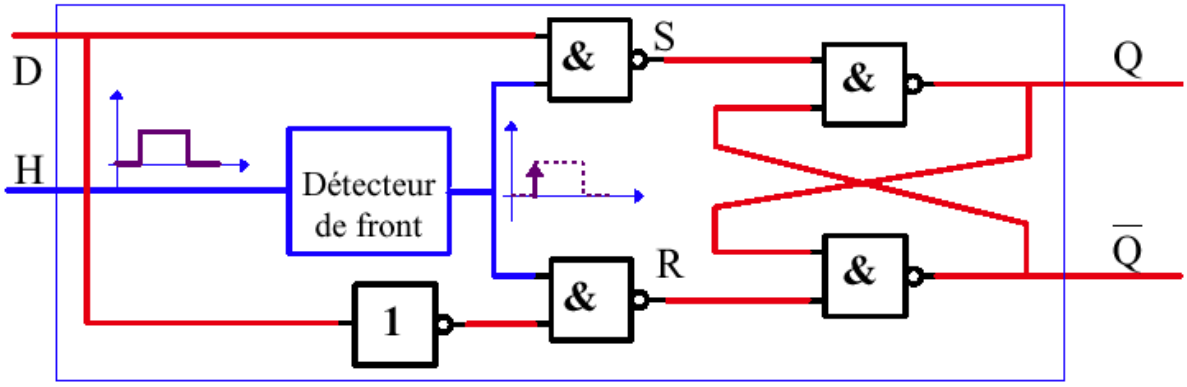


Utilisation: La sortie prend l'état de l'entrée D aprés l'impulsion d'horloge. Ceci permet par exemple de synchroniser le transfert de données en parallèle. (Voir codeur de clavier).

B) Bascule D à verrouillage (Latch).

a) Schéma interne

La structure interne d'une bascule synchrone comporte obligatoirement un détecteur de front,
qui permet de générer un signal de durée infime par rapport aux évolution extérieures, la prise
en compte de l'entrée D n'est réalisée qu'au front montant du signal d'horloge (de synchronisation).



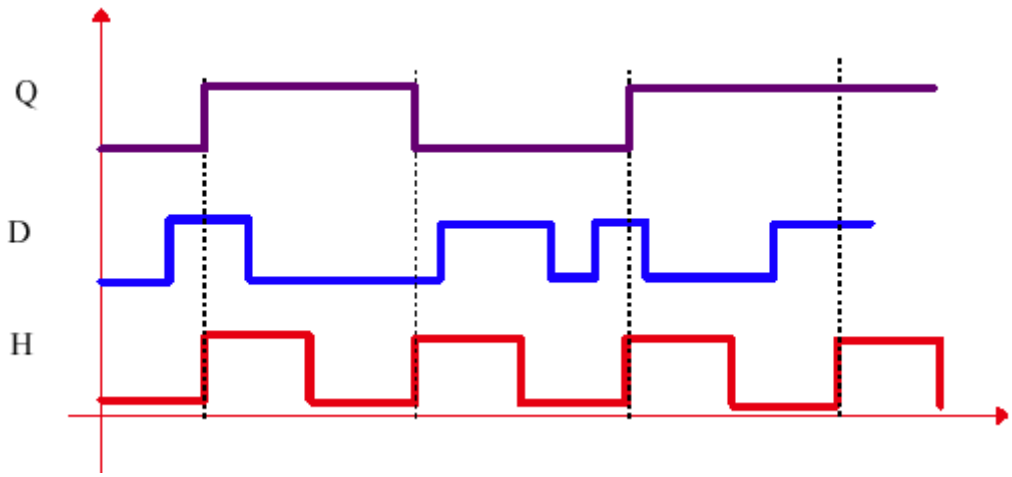
b) table de vérité (déclenchement sur front montant)

H	D	Q	$\overline{Q}$
↑	0	0	1
↑	1	1	0
0	x	Q_{n-1}	$\overline{Q_{n-1}}$
1	x	Q_{n-1}	$\overline{Q_{n-1}}$

Q_{n-1} :état précédent de Q

x : quel soit l'état

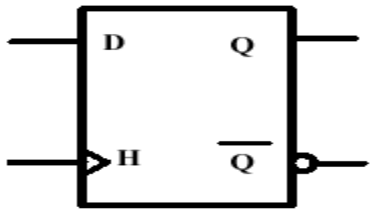
c) Chronogramme

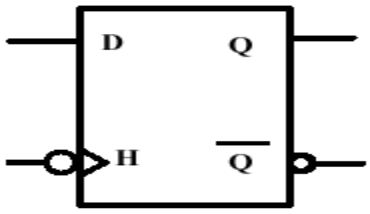


on constate :
entre deux fronts, le signal Q ne change pas d'état quelle que soit la valeur de D ;
un signal trop court de D n'est pas pris en compte .

d) Symbole

bascule D synchrone à déclenchement sur
front montant bascule D synchrone à déclenchement sur front descendant

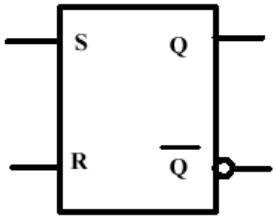




1. **Quelques bascules**

Désignation /symbole	Table de vérité	Fonctionnement/utilisation

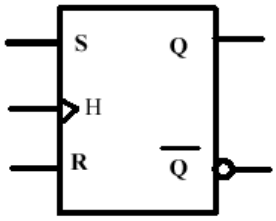
Bascule RS



S	R	Q_n	$\overline{Q_n}$
0	0	Q_{n-1}	$\overline{Q_{n-1}}$
0	1	0	1
1	0	1	0
1	1	interdit	interdit

l'utilisation principale est la mémorisation d'un signal, avec une possibilité de mise à 0(R) et de mise à 1 (S). L'état 1,1 (pour les bascule à base de NOR) est en général à éviter, perte de complémentarité des sorties

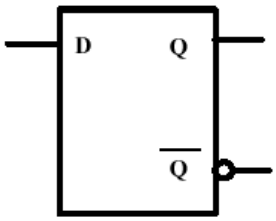
Bascule RS Synchrone



H	S	R	Q_n	$\overline{Q_n}$
↑	0	0	Q_{n-1}	$\overline{Q_{n-1}}$
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	interdit	interdit
1	x	x	Q_{n-1}	$\overline{Q_{n-1}}$
0	x	x	Q_{n-1}	$\overline{Q_{n-1}}$

idem précédente mais le changement d'état ne s'effectue qu'à l'instant du front montant de H

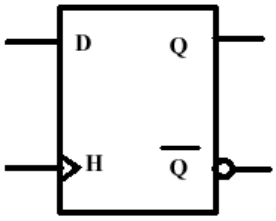
Bascule D



D	Q	$\overline{Q}$
0	0	1
1	1	0

L'entrée D est recopiée sur la sortie Q.

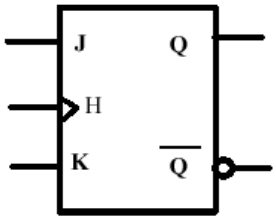
Bascule D synchrone sur front descendant



H	D	Q	$\overline{Q}$
↑	0	0	1
↑	1	1	0
0	x	Q_{n-1}	$\overline{Q_{n-1}}$
1	x	Q_{n-1}	$\overline{Q_{n-1}}$

Idem précédente, le changement d'état ne s'effectue qu'à chaque front montant de H

Bascule JK synchrone

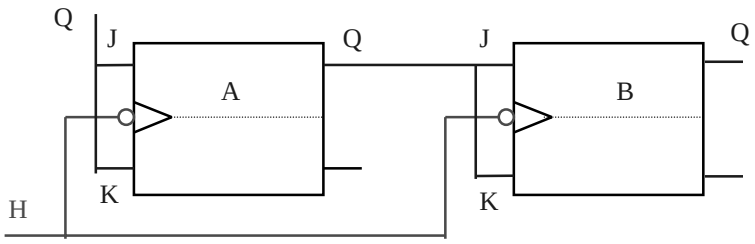


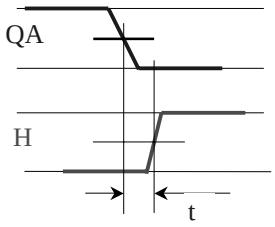
H	J	K	Q_n	$\overline{Q_n}$
↑	0	0	Q_{n-1}	$\overline{Q_{n-1}}$
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	$\overline{Q_{n-1}}$	Q_{n-1}
1	x	x	Q_{n-1}	$\overline{Q_{n-1}}$
0	x	x	Q_{n-1}	$\overline{Q_{n-1}}$

La bascule JK est une bascule universelle, elle se comporte comme une bascule RS, sans l'ambiguïté précédente, pour la combinaison 1-1, il y a inversion des sorties

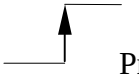
10.4.5. Bascule Maitre-Esclave.

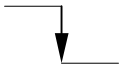
Problème: Les bascules synchrones nécessitent des états stables sur leurs entrées au moment de la transition du signal d'horloge, cela n'est pas toujours possible lorsque plusieurs bascules sont cablées entre elles (ex: en comptage) et l'on a des aléas de fonctionnement.



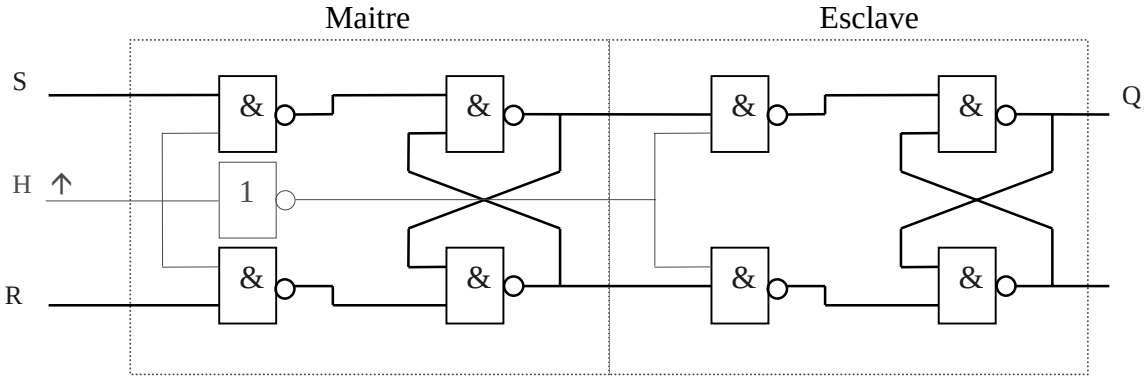


Solution: Il existe des bascules à 2 étages qui évoluent en 2 temps.

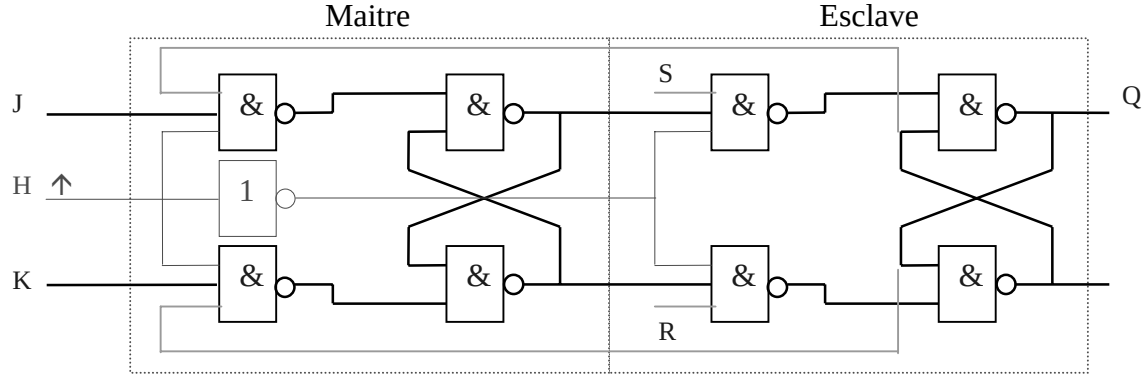
1er temps:  Verrouillage du 2ème étage
Prise en compte des entrées par le 1er étage

2ème temps  Verrouillage du 1er étage
Prise en compte des données par le 2ème étage

A) Bascule R S H Maitre-Esclave.



B) Bascule J K Maitre-Esclave.

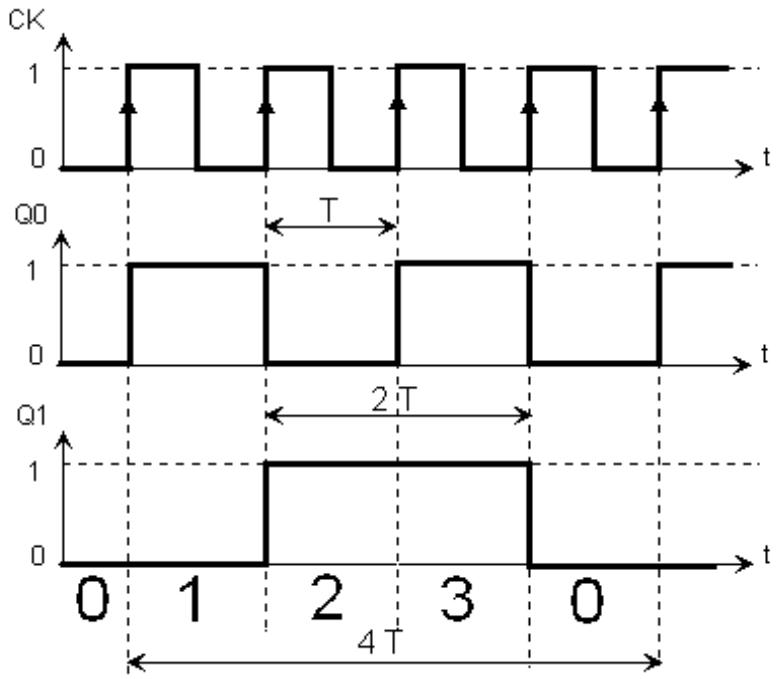


Remarque: En agissant directement sur la bascule esclave, on peut forcer à "1" ou "0" la sortie Q de la bascule J K R et S sont des entrées de forçage.

10.4.6.Etude du comptage.

Un compteur décimal correspond a un comptage BCD (Binary Coded Décimal : Décimal Codé Binaire) de 0000 (0) à 1001 (9). Quatre bascules associées fournissent un compteur modulo 16 en binaire, mais câblées en compteur BCD, on force, les sorties à 0, lorsque le cycle parvient à 10.

Exemple : Compteur modulo , soit 2^1 (compteur asynchrone à 2 bascules D).



Les bascules D sont câblées en diviseur par 2

A partir de bascules (D ou JK) : à fronts montants :

$\overline{Qn}$ **relié à CK, câblage en compteur**

$\overline{Qn}$

Qn relié à CK, câblage en décompteur

10.4.7. Comptage / décomptage asynchrone.(Le modulo 8 nécessite 3 bascules J K)

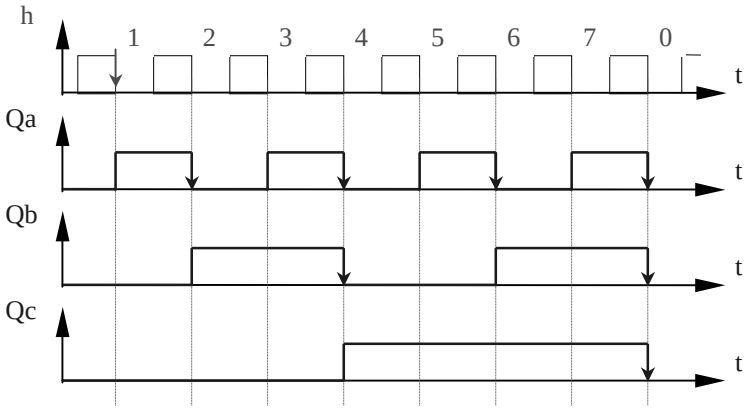
Exemple : décompteur modulo 8, soit 2^3 (décompteur asynchrone à 3 bascules JK).

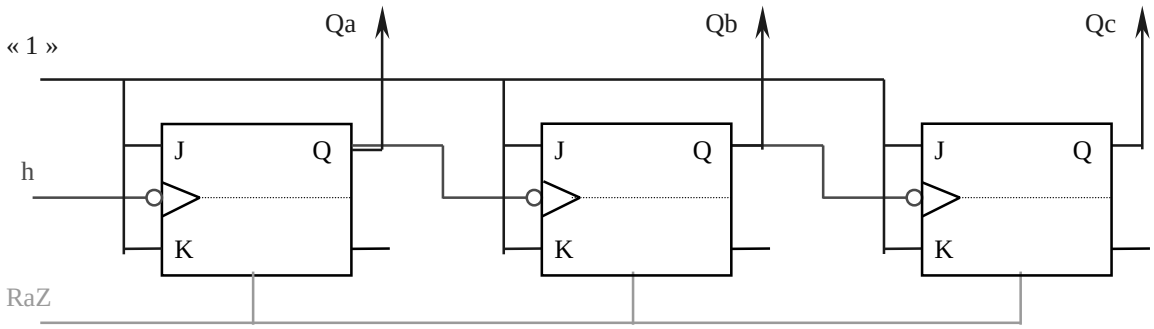
Les bascules JK, sont câblées en diviseur par 2, les entrées J et K, sont reliées à Vcc. Le changement d'état des bascules se fait en cascade, d'une bascule à l'autre. Cette structure est dite de type asynchrone.

Compteur modulo 8

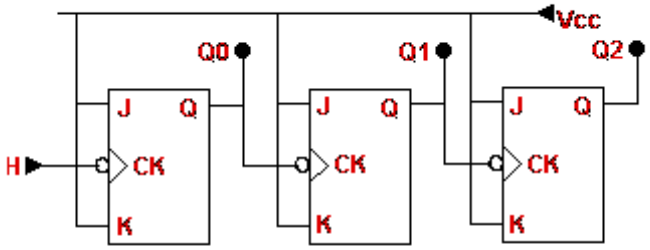
N	Qc	Qb	Qa
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Chronogrammes.

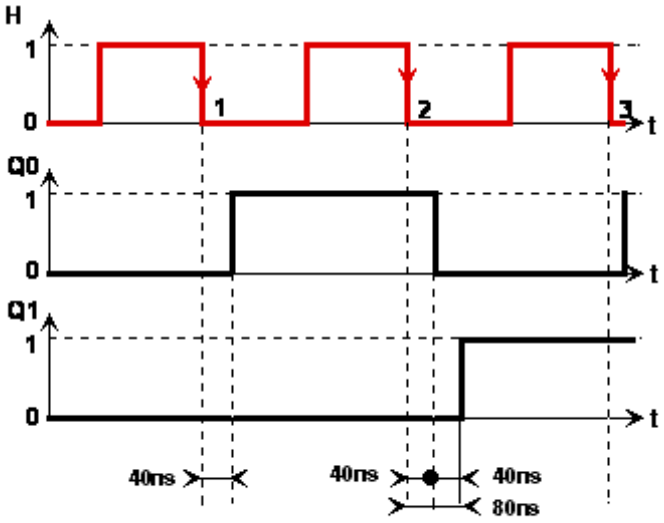




Structure Asynchrone

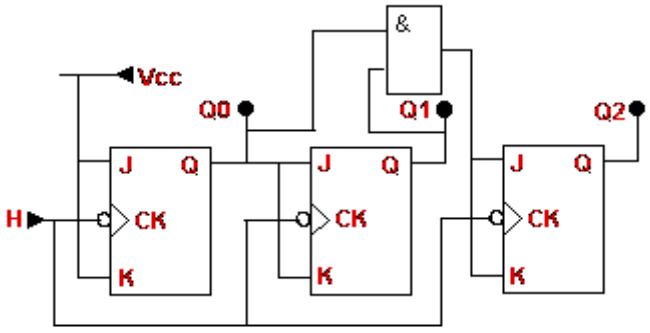


Asynchrone ou en cascade, chaque sortie d'une bascule est reliée à l'entrée d'horloge de la bascule suivante.

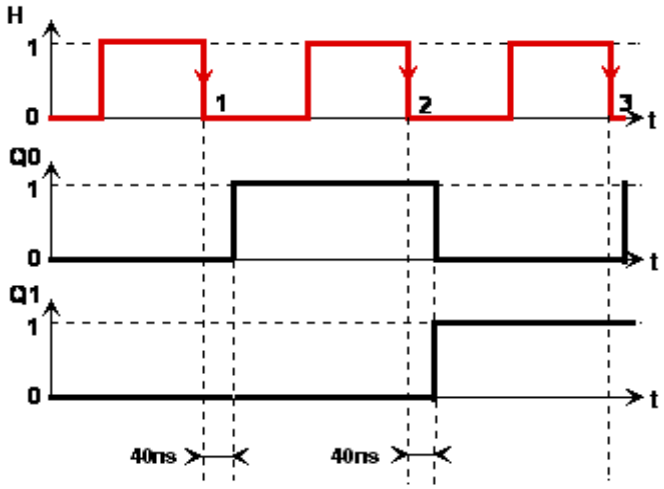


D'une bascule à l'autre les temps de propagation s'ajoutent

Structure synchrone



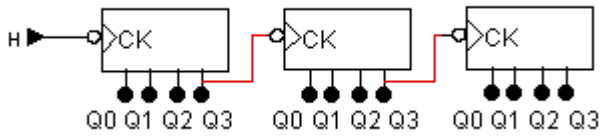
Synchrone, l'entrée d'horloge est commune à toutes les bascules.



Il n'y a pas d'accumulation de retard de propagation

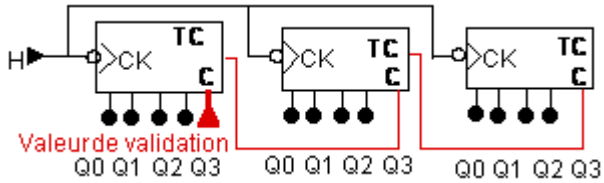
10.4.8. Mise en cascade des compteurs

a) Structure asynchrone



Pour la mise en cascade des compteurs asynchrones, il suffit de relier la sortie du dernier étage du compteur précédent sur l'entrée d'horloge du premier étage du compteur suivant.

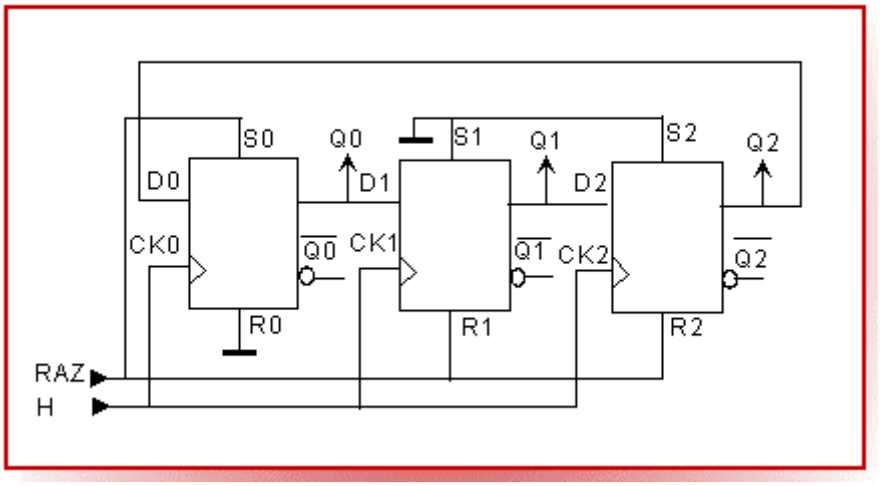
b) Struture synchrone

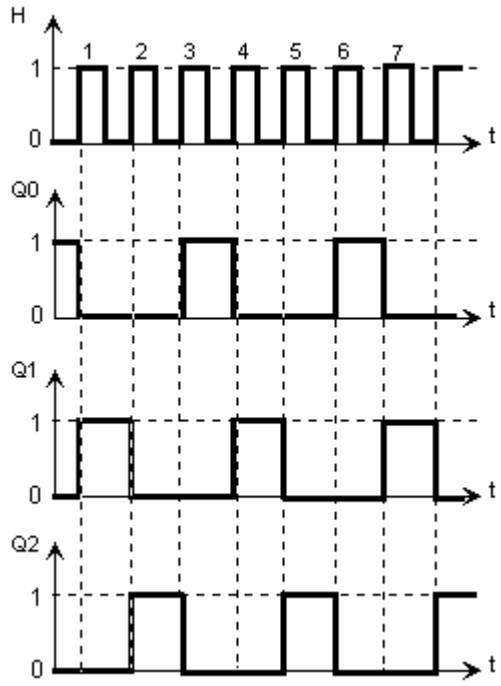


L'horloge est la même pour tous les boîtiers. La sortie de retenue TC, est cablée à l'entrée de validation C de deux boitiers adjacents.

Principe d'un compteur à anneau élémentaire
Le compteur circulaire le plus élémentaire qui soit est un registre à décalage réalisé au moyen de bascules D (ou de bascules JK).

Les bascules sont connectées de façon à ce que l'information soit décalé de gauche à droite. Il s'agit ici d'un compteur en anneau modulo 3.

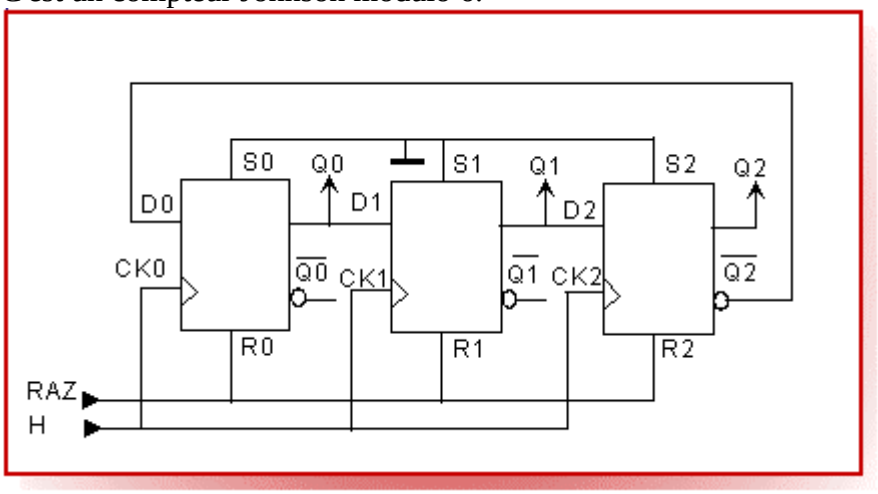


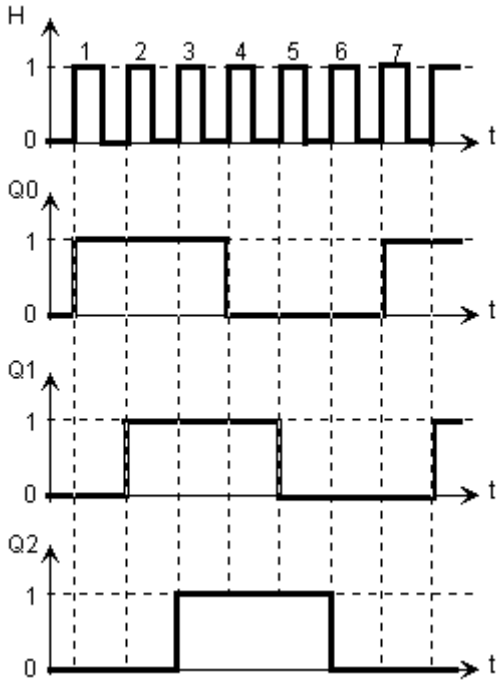


10.4.9. Principe d'un compteur Johnson

On construit le compteur Johnson exactement comme le compteur à anneau normal, sauf qu'on raccorde la sortie complémentée de la dernière bascule à l'entrée de la première bascule. Le compteur prend les six états suivants avant que ne se répète la séquence d'opération : 000, 100, 110, 111, 011 et 001.

C'est un compteur Johnson modulo 6.





a) Compteur asynchrone modulo 10.

En général, La réalisation d'un compteur modulo 10 se fait par la remise à zéro du compteur à la $10^{ème}$ impulsion.

Ce type de compteur présente donc pendant un court instant la combinaison 1 0 1 0 (10) sur ses sorties, le temps de la remise à zéro.

Afin d'éviter cet état intermédiaire on peut anticiper l'évolution des bascules au passage de
1 0 0 1 (9) à 0 0 0 0 (0) au lieu de 1 0 1 0.

Il faut donc: - Forcer le bit de poids fort à passer à zéro. . .
- Interdire au bit de poids 2^1 de passer à « 1 ». . . . . .

Montage

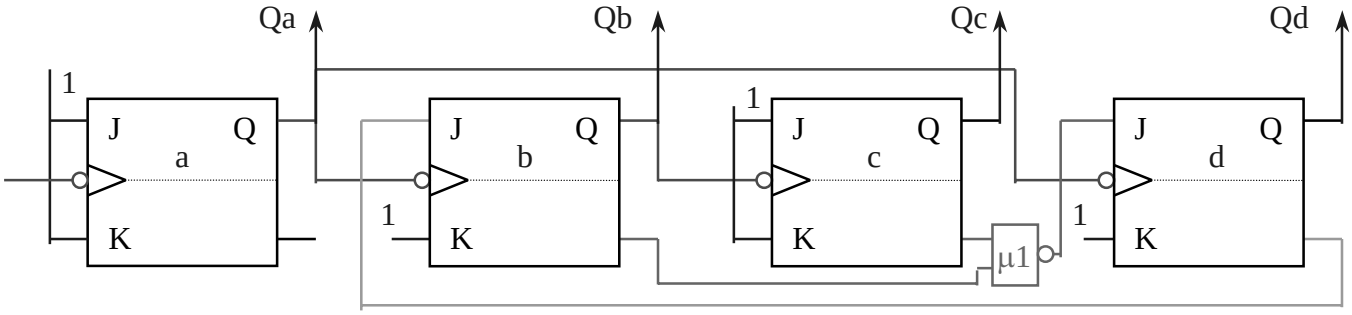
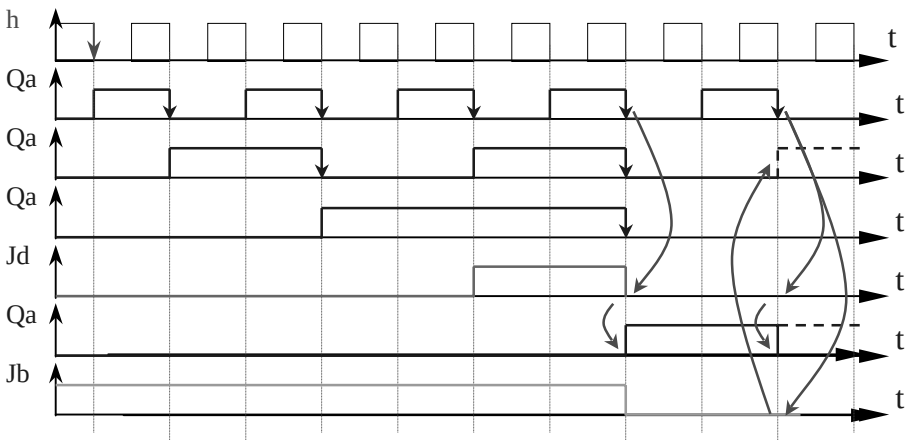


Table de vérité.

N	Qd	Qc	Qb	Qa
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	0	0

Chronogrammes.



b) Comptage en synchrone : Compteur modulo 8 synchrone.

L'impulsion d'horloge est appliquée simultanément à chaque bascule. Celles-ci évoluent en fonction des informations présentent sur leurs entrées J, K au moment ou apparait l'impulsion. Il faut donc prépositionner J et K à l'instant t pour obtenir le basculement désiré à l'instant t+1.

J	K		Q
0	0		Q
0	1		0
1	0		1
1	1		$\overline{Q}$

Qn	Q n-1
0 →	1
1 →	0
1 →	1
0 →	0

J	K
1	x
x	1
x	0
0	x

Si QC, QB, QA, sont les sorties de trois bascules on a la table de vérité suivante:

N		Qc	Qb	Qa		Jc	Kc		Jb	Kb		Ja	Ka
0		0	0	0		0	x		0	x		1	x
1		0	0	1		0	x		1	x		x	1
2		0	1	0		0	x		x	0		1	x
3		0	1	1		1	x		x	1		x	1
4		1	0	0		x	0		0	x		1	x
5		1	0	1		x	0		1	x		x	1
6		1	1	0		x	0		x	0		1	x
7		1	1	1		x	1		x	1		x	1

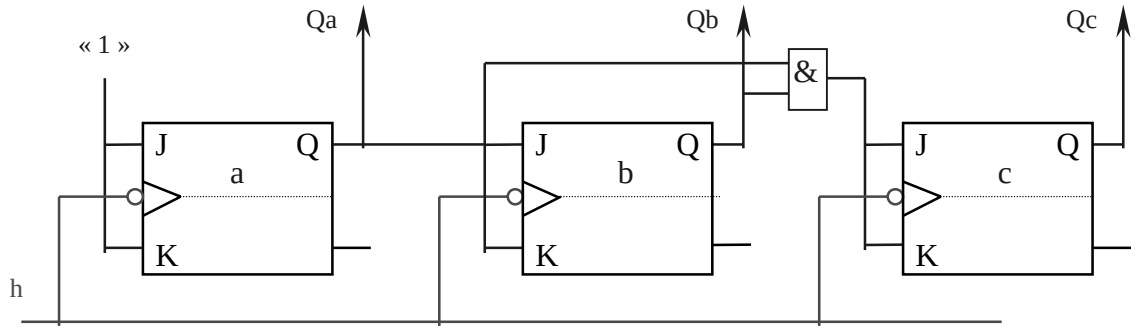
<u>Equations:</u> (des entrées J et K obtenues par KARNAUGH).

JA = KA = 1

JB = KB = Qa

JC = KC = Qa . Qb

Schéma:



c) Décompteur modulo 8 synchrone.

N		Qc	Qb	Qa		Jc	Kc		Jb	Kb		Ja	Ka
7		1	1	1		x	0		x	0		x	1
6		1	1	0		x	0		x	1		1	x
5		1	0	1		x	0		0	x		x	1
4		1	0	0		x	1		1	x		1	x
3		0	1	1		0	x		x	0		x	1
2		0	1	0		0	x		x	1		1	x
1		0	0	1		0	x		0	x		x	1
0		0	0	0		1	x		1	x		1	x

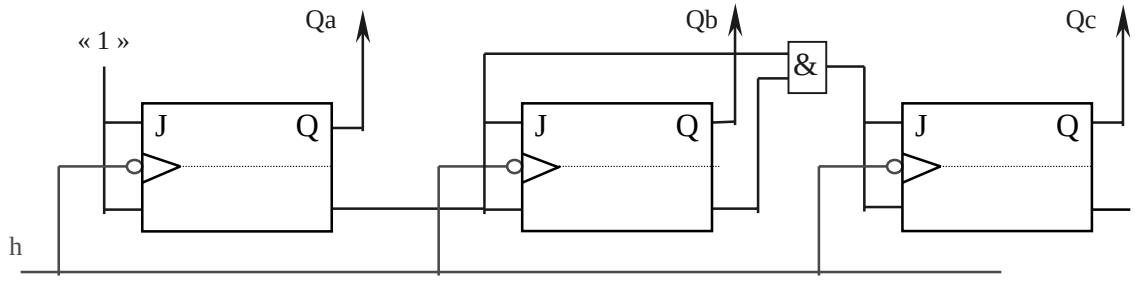
Equations: (des entrées J et K obtenues par KARNAUGH).

JA = KA = 1.

JB = KB = $\overline{Qa}$

JC = KC = $\overline{Qa} \cdot \overline{Qb}$

Schéma:



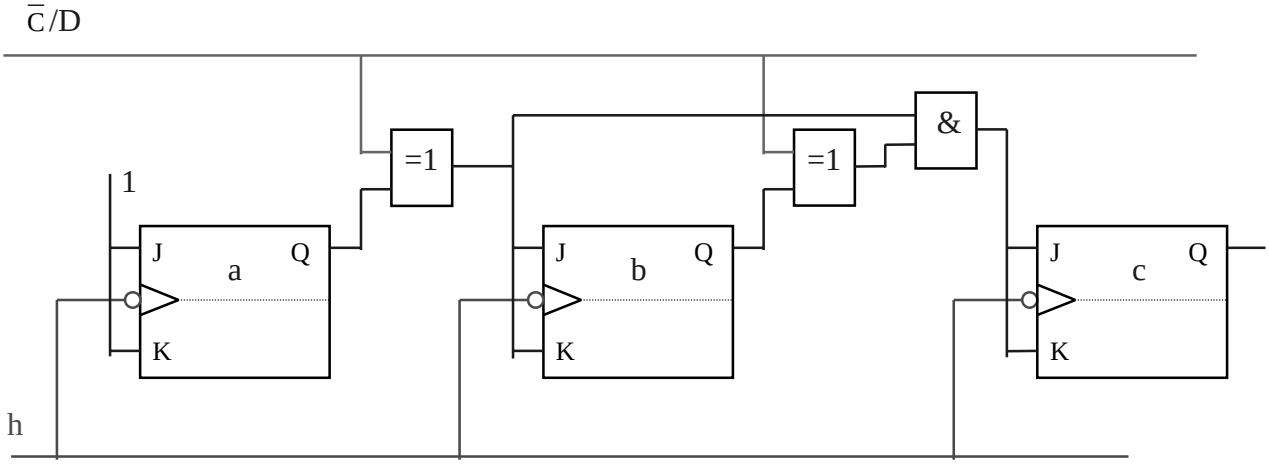
c) Compteur / Décompteur modulo 8 synchrone.

Analyse du problème En comptage C = .0. JB = KB = .Qa.

En décomptage C = .1. JB = KB = . $\overline{Qa}$.

d'ou JB = KB = $Qa \cdot \overline{C} + \overline{Q}a\ C = Qa \oplus C$

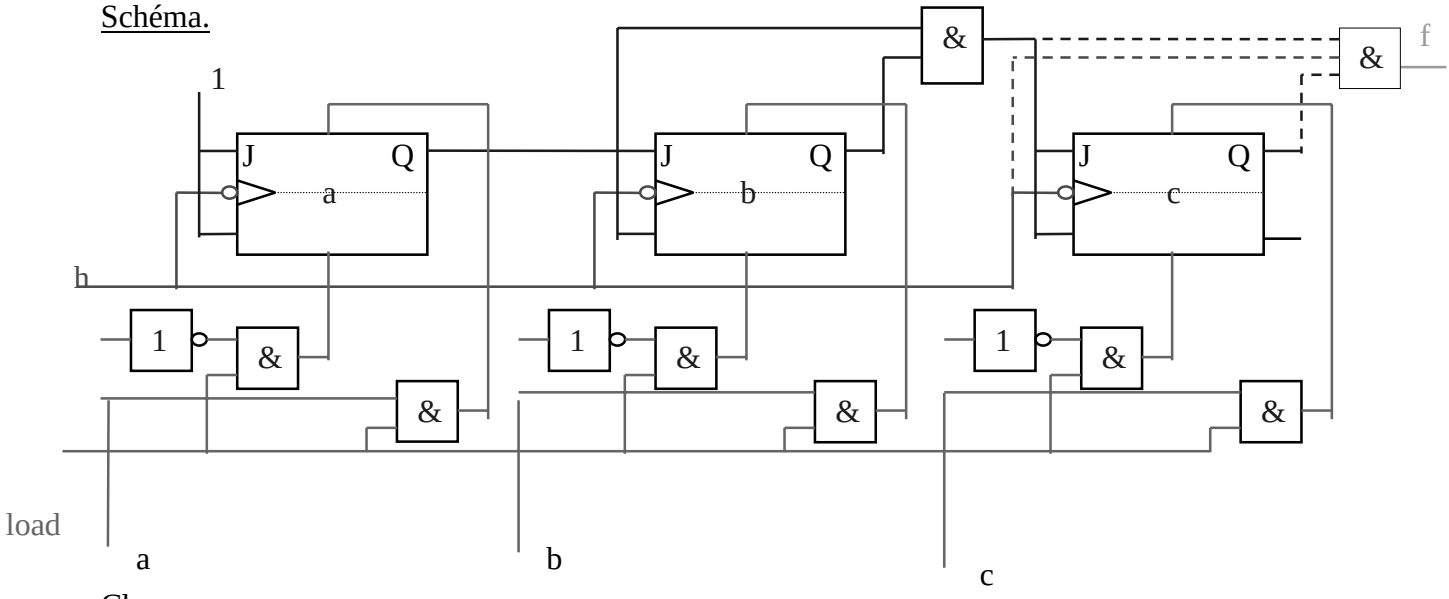
JC = KC = $Qa \cdot Qb \cdot \overline{C} + \overline{Q}a\ \overline{Q}b\ C = (Qa \oplus C)(Qb \oplus C)$



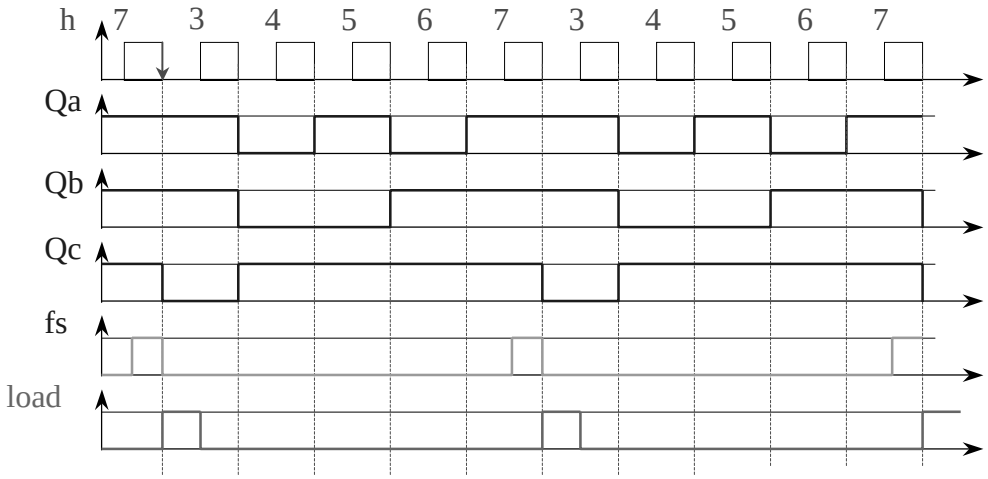
10.4.8. Compteur prépositionnable.

Si nous avons accés aux entrées de forçage S et R sur les bascules, nous pouvons initialiser (prépositionner) le compteur à une valeur autre que zéro. Une entrée de chargement (load) permet la prise en compte des entrées parallèles a,b,c …

Schéma.



Chronogrammes:



Application: Diviseur de fréquence différent de 2^n . (voir manipulation)

Applications sur le comptage asynchrone.

a) Tracer les chronogrammes des sorties Qa, Qb, Qc, Qd d'un compteur 74 LS 90 lorsqu'il est utilisé : en B C D;
- en biquinaire.

b) Réaliser le schéma complet et normalisé d'un compteur modulo 60 à l'aide des compteurs 74 LS 90 et 74 LS 92 avec sortie sur afficheurs.

c)réaliser l'étude d'un compteur-décompteur synchrone modulo 10.

10.4.10. Réalisation de chaine de comptage (x. 4 BITS)

Pour réaliser des compteurs de grande capacité, on connecte des compteurs 4 bits en cascade ce qui est rendu très aisé par la présence de fonctions particulières sur certains circuits.

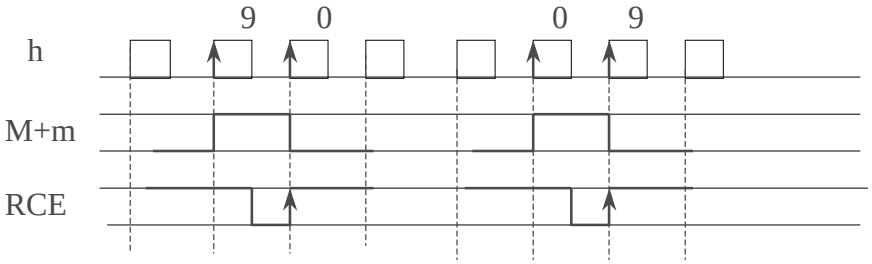
A) Etude du 74190 et du 74191:

Ces compteurs B C D synchrones possédent deux sorties particulières;

- RCE ou RCO : transmet le dixième front d'horloge pour la décade de poids supérieur .

- M + m : passe à 1 pour 9 en comptage et 0 en décomptage.

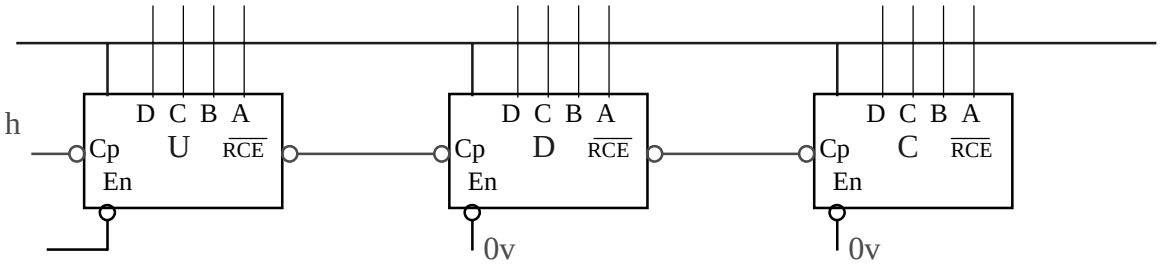
Chronogrammes:



Cette connexion peut être réalisée de 3 manières différentes.

1) Chaîne connectée en mode asynchrone

Seule la décade des unités (LSB) reçoit les impulsions à compter sur son entrée d'horloge (*CP*). Les autres décades reçoivent sur l'entrée *CP*, le signal de sortie $\overline{RCE}$ de la décade de poids inférieur. Toutes les décades sont en position de "fonctionnement autorisé", avec l'entrée $\overline{EN}$ à "0" sauf la premiere qui reçoit un ordre extérieur pour valider ou non le comptage. Enfin, toutes les décades reçoivent le signal de comptage/décomptage *UP/DOWN*.



Trois conditions sont nécessaires pour un bon fonctionnement.

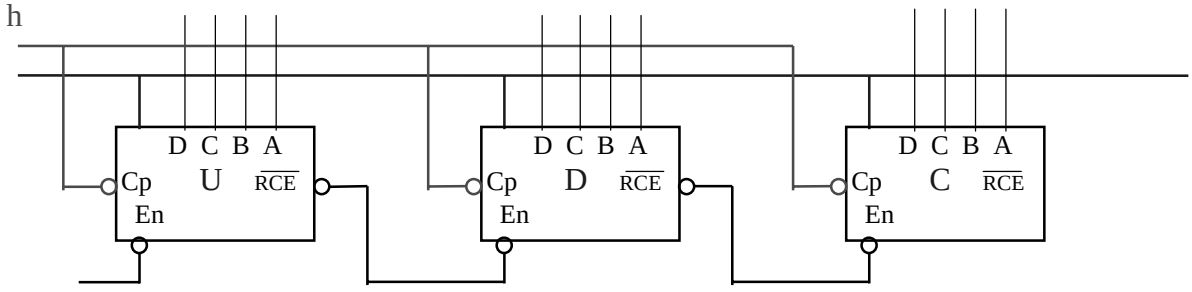
- l'état de l'entrée "*DOWN* / *UP* " ne doit pas être modifié quand le signal d'horloge est à zéro, car sinon la sortie *RCE* qui est conditionnée par l'état de *DOWN* / *UP* à travers la sortie *M+ m*, pourrait délivrer une impulsion parasite ;

- cet état de *DOWN* / *UP* ne doit pas être non plus modifié avant que le signal d'horloge ne se soit propagé jusqu'au dernier étage de la chaîne, sinon les derniers étages pourraient compter au lieu de décompter d'une impulsion (ou inversement) ;

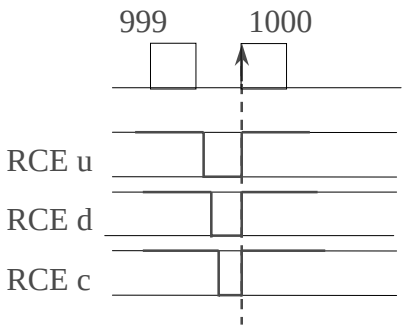
- la vitesse de fonctionnement est limitée par le temps de propagation du signal "horloge" à travers la chaine. La durée du signal d'horloge sera conditionnée par le temps de réponse des sorties $\overline{RCE}$, elle dépend aussi du nombre d'étages.

10.4.11. Chaîne connectée en mode synchrone (propagation en cascade)

Toutes les décades reçoivent simultanément l'impulsion de comptage (en *CP*). La premiere décade permet de bloquer le fonctionnement par son entrée $\overline{EN}$. La sortie $\overline{RCE}$ est utilisée pour permettre l'incrémentation de la décade de poids supérieur en validant cette-dernière par $\overline{EN}$.

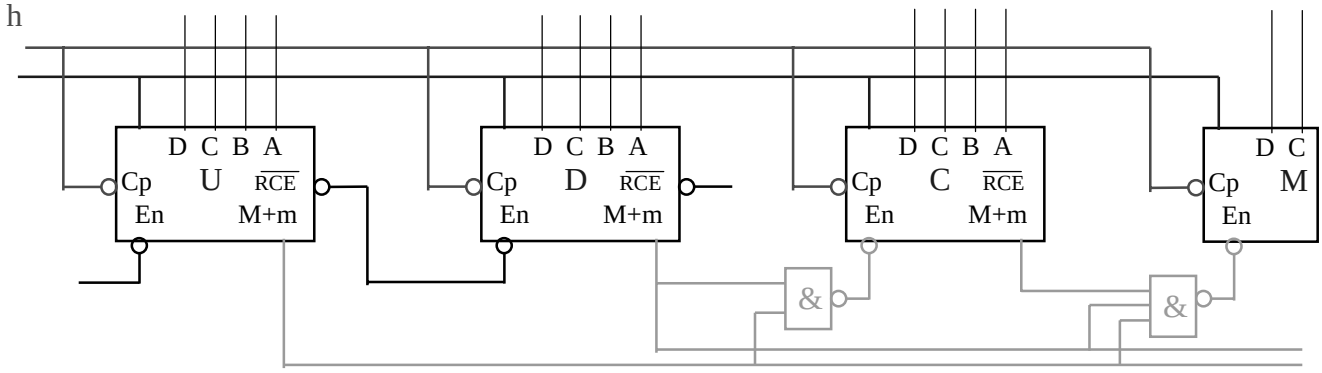


Comme une sortie $\overline{RCE}$ ne passe à zéro que lorsque l'entrée $\overline{EN}$ (c'est-à-dire ici la sortie $\overline{RCE}$ de la décade précédente) passe à zéro, il y a nécessairement un retard qui devient de plus en plus grand à mesure que le nombre d'étages croît. La fréquence de fonctionnement en sera donc limitée d'autant. On retrouve la limitation propre au report série.



10.4.12.Chaîne connectée en mode synchrone avec propagation anticipée

C'est l'organisation qui permet le fonctionnement le plus rapide, pour des bascules d'un type donné. Les impulsions de comptage sont envoyées simultanément à toutes les décades, ainsi que la commande *DOWN* / *UP*, mais l'autorisation de fonctionnement (condition sur l'entrée $\overline{EN}$) est élaborée en mettant en condition ET les états, exprimés par *M + m***,** de tous les étages précédents (report parallèle).



On constate que la premiere décade reçoit un ordre extérieur sur l'entrée $\overline{EN}$, par lequel on commande l'état de toute la chaîne (validation de fonctionnement) et l'autorisation de la décade de poids supérieur est simplement obtenue à partir de la sortie $\overline{RCE}$ de la premiere décade (la sortie $\overline{RCE}$ est conditionnée par l'état de *M + m*).

La fréquence maximale de fonctionnement de cette chaîne est donc seulement limitée par un seul temps de retard, quelque soit la longueur de cette chaîne, celui de *M + m* plus le temps de réponse d'une porte NON ET, soit au total typiquement 25 ns.

11. MISE EN FORME DES SIGNAUX [10]

11.1 Mise en forme des signaux.

a) Circuit anti-rebond RC

b) Circuit anti-rebond à bascule RS

c) RIGGER de SCHMITT

- Réalisation en circuit CMOS
- Réalisation à l'aide d'un AO.

11.2. Monostables

a) Réalisation à l'aide d'un AO

- Montage classique
- Montage dérivé de l’astable.

b) Réalisation à l'aide de circuit logiques CMOS.

- Avec des portes logiques
- Avec un circuit 4538

c) Réalisation à l'aide de circuit logiques TTL 74121

11.3. ASTABLES

a) Réalisation à l'aide d'un A.O

b) Réalisation à l'aide de porte logique CMOS et TTL

c) Oscillateur àcircuit intégré spécialisé NE 555

d) Oscillateur à quatrz

e) Oscillateur commandé en tension (VCO)

f) Oscillateur à asservissement de phase

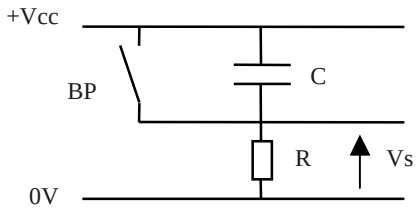
11.4. Montages Anti-Rebond

Le rebond des contacts est un problème qu'il faut absolument éliminer dès que l'on travaille avec des circuits logiques en comptage ou en logique programmée.

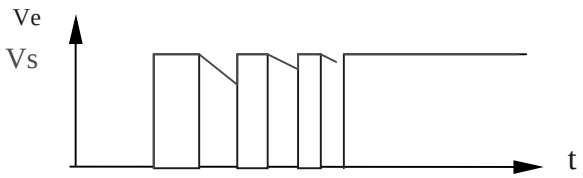
Plusieurs solutions permettent de résoudre le problème selon l'utilisation.

a) Circuit anti-rebond RC

Montage :



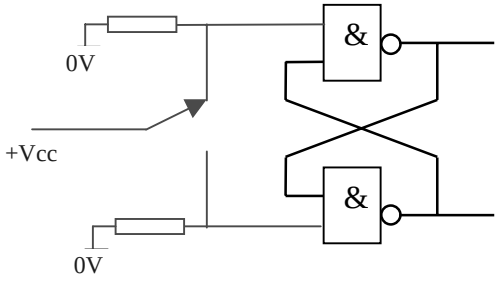
Oscillogramme :



Ce circuit a pour avantage d'être fiable et peu onéreux.

b) Circuit anti-rebond à bascule RS

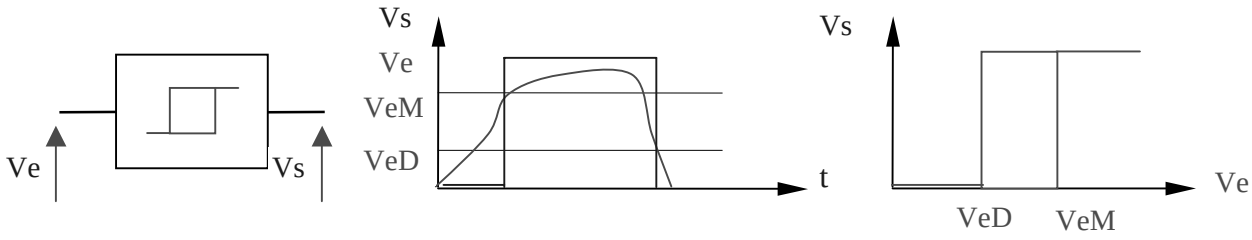
Montage :



Oscillogramme :

c) Circuit Trigger de SCHMITT

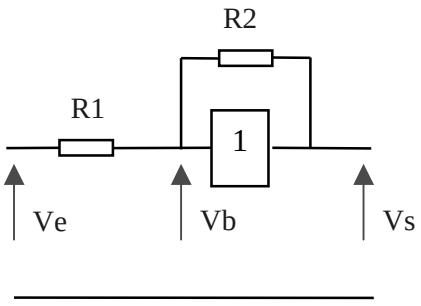
Principe :



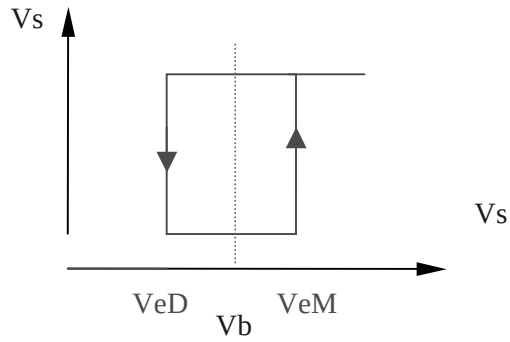
d) Réalisation à l'aide d'un circuit CMOS

Le basculement se fait pour Vb = $\boxed{\frac{VDD - VSS}{2} = \frac{Vcc}{2}}$

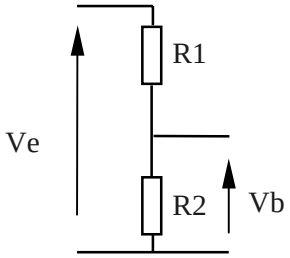
Montage :



Caractéristique de Transfert :



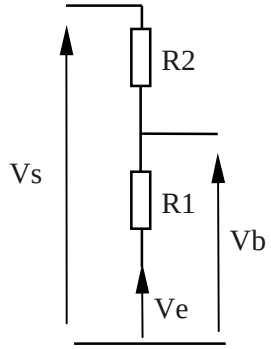
1er cas : Vs = 0 basculement pour Ve_M = ?



Vb = $\frac{VeR2}{R1 + R2}$ $\Rightarrow$ Ve = $Vb\left(\frac{R1 + R2}{R2}\right)$

Ve_M = $\boxed{\frac{Vcc}{2}\left(\frac{R2 + R1}{R2}\right)}$ > Vcc / 2

2ème cas : Vs = + Vcc basculement pour Ve_D = ?



Vb = $Ve + (Vs - Ve)\frac{R1}{R1 + R2}$

Ve_D = $\boxed{\frac{Vcc}{2}\left(\frac{R2 - R1}{R2}\right)}$ < Vcc / 2

AN1 : On donne R1 = 470 Ω R2 = 10 R1 . . V_{CC} = 12 V. . .

Ve_M = 6 x 11 / 10 = 6.6 V

Ve_D = 6 x 9 / 10 = 5.4 V

Symétrie par rapport à Vcc / 2 soit 6 V

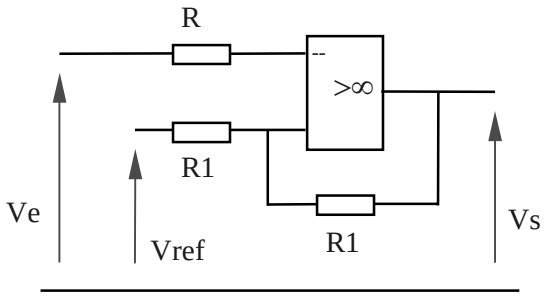
AN2 : On donne Ve_M = 10 V VeD = 2 v (symétrie par rapport à Vcc / 2)

R2 = 10 k Ω (R1 + R2) / R2 = 1.67 $\Rightarrow$ R1 = 6,7 k

E) Réalisation à l'aide d'un A.O

Il faut réaliser une contre réaction positive. Le point de basculement peut être réglable.

- Trigger de SCHMITT inverseur R = R1 // R2



1er cas : V_S = + Vsat.

$$VeD = \frac{VsatR1}{R1+R2} + \frac{VrefR2}{R1+R2}$$

2ème cas : V_S = -- Vsat

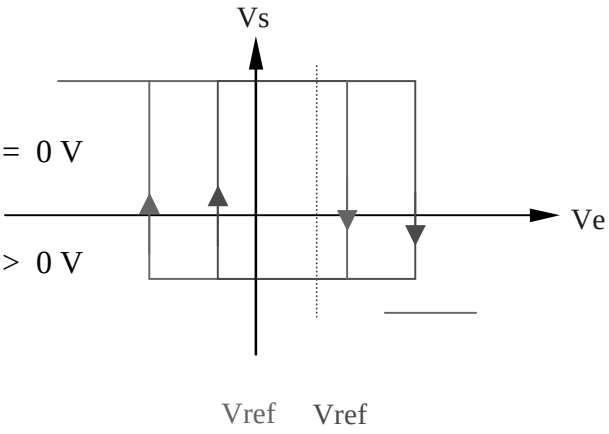
$$VeM = \frac{-VsatR1}{R1+R2} + \frac{VrefR2}{R1+R2}$$

Caractéristique de transfert pour Vref = O V

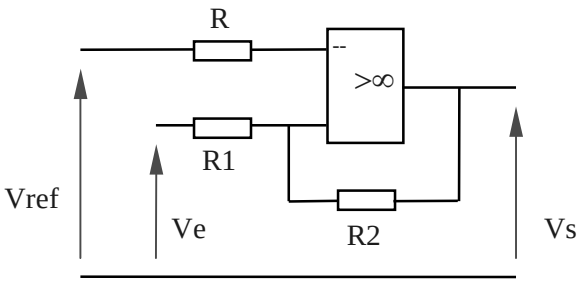
$$Ve_D = \frac{VsatR1}{R1+R2}$$

$$Ve_M = \frac{-VsatR1}{R1+R2}$$

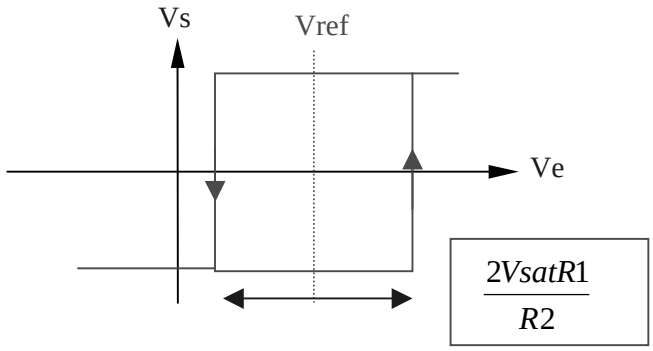
Largeur du cycle. $\frac{2VsatR1}{R1+R2}$



* Trigger de SCHMITT non inverseur



Caractéristique de transfert :



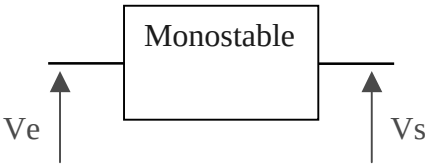
$$VeM = Vref \frac{(R1+R2)}{R2} + Vsat \frac{R1}{R2}$$

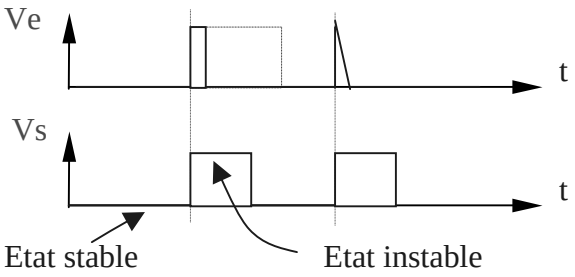
$$VeD = Vref \frac{(R1+R2)}{R2} - Vsat \frac{R1}{R2}$$

11.5. Etude des monostables

Les monostables sont des temporisateurs de courte durée qui permettent de :

- rendre un signal furtif exploitable ;
- créer un signal à partir d'un front.

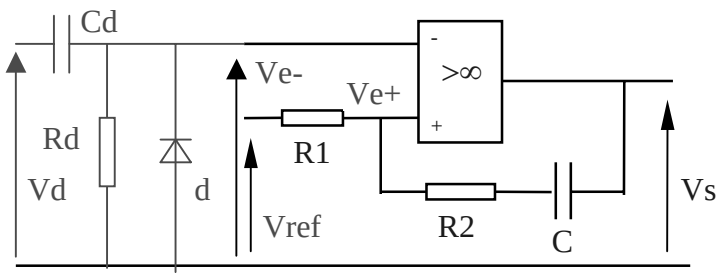




Ils peuvent être réalisés à l'aide de circuits analogiques (A.O) ou de circuits logiques (TTL-CMOS)

1) Réalisation à l'aide d'un A.O

a) Etude du montage classique

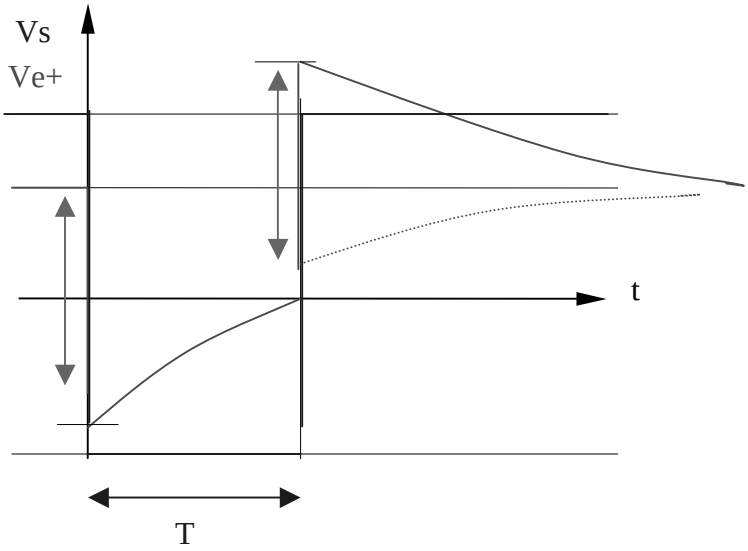


Déclenchement par un circuit dérivateur ⟶ Impulsion brève

Ve- > Vref

Vs passe de +Vsat à -Vsat

Chronogramme pour Ve+



Pente : ❶ = (R1 + R2)

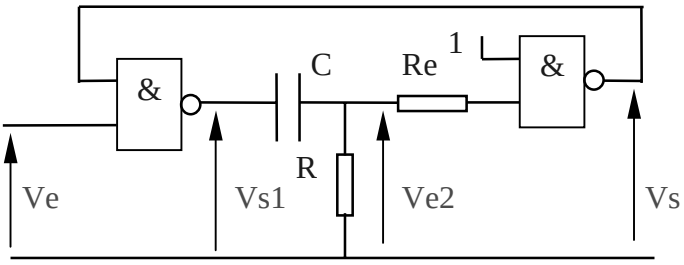
$$\frac{2VsatR1}{R1+R2}$$

$$T = (R1+R2)C\ln\left(\frac{2Vsat}{|Vref|}\frac{R1}{R1+R2}\right)$$

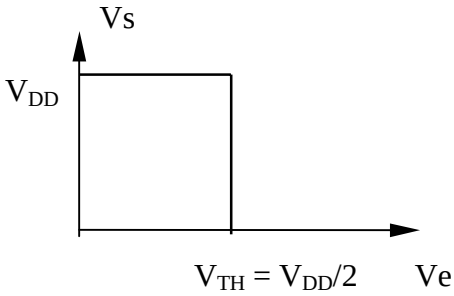
11.6. Réalisation à l'aide de circuits logiques.

a) Utilisation de portes logiques CMOS.

Montage :



caractéristique de transfert :



* Etat stable : à l'état stable, aucun courant ne circule dans le condensateur. Par conséquent :

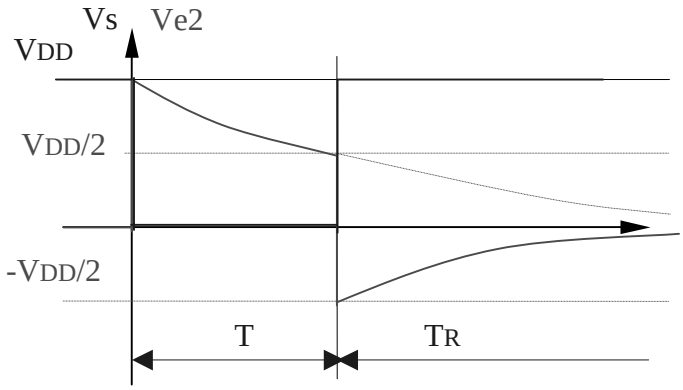
$Ve_2 =$ 0 V et donc Vs = « 1 »

- Fonctionnement : En pratique, Ve = V_{DD} au repos; le déclenchement est obtenu par Ve = 0V.

ce qui entraine : le passage à « 1 » de Vs1 et la charge du condensateur .

La charge dure jusqu'à ce que : Ve2 atteigne V_{TH}

* Chronogrammes :



Calcul de T :

$Ve2 = V_{DD}\, e^{-t/\tau}$ avec $\tau = R\,C$

à $t = T$ $Ve2 = V_{DD}/2$. .

$T = \tau \ln 2 = 0.69\ R\,C.$

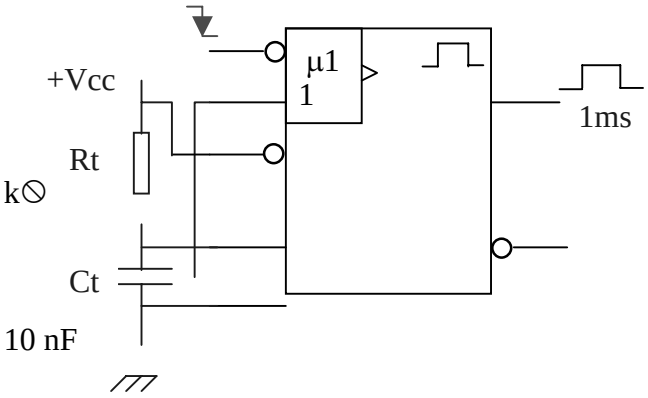
Note : Il faut tenir compte du temps de récupération avant de relancer le monostable.

$T_R =$ 3 à 5 τ avec $\tau = R\,C$ (temps mis par une exponentielle pour « atteindre » son asymptote).

b) Utilisation d'un circuit CMOS spécialisé 4538(voir doc)

Le 4538 est un double monostable-multivibrateur de précision.

Mise en oeuvre du 4538 pour réaliser un monostable de 1ms déclenché par un front descendant.



T = Rt Ct.

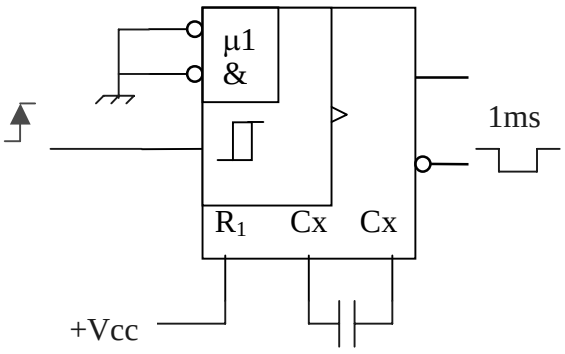
pour Ct = 0.1 µF on a Rt = 10

pour Rt = 100 kΩ on a Ct =

c) Utilisation d'un circuit TTL spécialisé 74121.

Le 74121 est un monostable-multivibrateur avec bascules de SCHMITT en entrées.

Mise en oeuvre du 74121 pour réaliser un monostable de 1ms à l'état bas déclenché sur front montant.



R = 2 kΩ

Tp = 0.7 Cx Rx

Cx = Tp / 0.7 Rx

Cx = 0.71 µF

11.7. Etude des Astables.

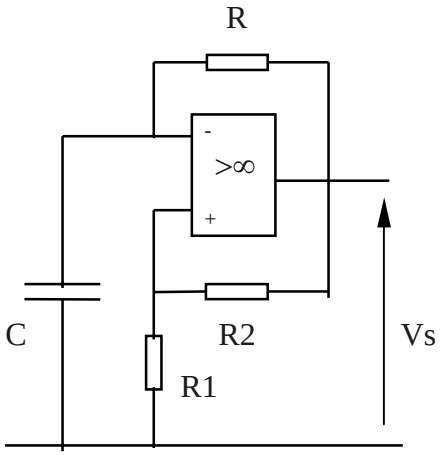
Les astables sont des circuits oscillant en permanence dont la sortie peut être utilisée comme :

Signal d'horloge dans les systèmes numériques.

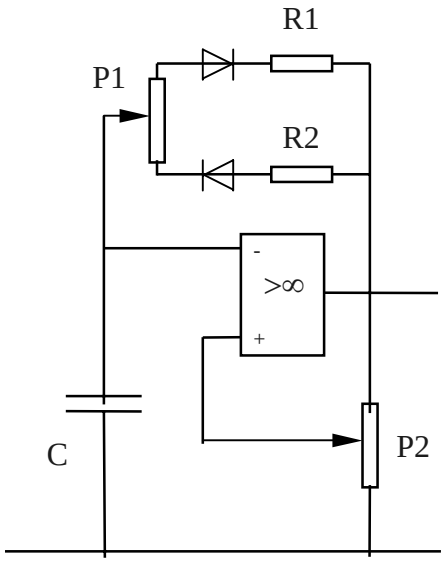
Ils peuvent être réalisés à partir de composants divers, en fonction de la précision et de la stabilité voulue.

a) Oscillateur à amplificateur-opérationnel.

Montages :



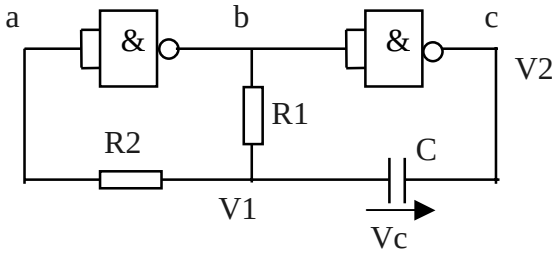
$$2\,R\,C\,\ln\,(1\,+\,2\,\frac{R1}{R2})$$



Dans ce montage, on peut régler la fréquence d'oscillation et le rapport cyclique indépendament ;
P1 agit sur le rapport cyclique ;
P2 agit sur la fréquence.

b) Oscillateur à portes logiques TTL ou CMOS.

Montage.



Vc = V2 – V1

donc V1 = V2 -Vc

Fonctionnement :

hypothèse de départ : a = 1 donc b = .0. donc c = .1. et C est déchargé.

La capacité se charge ce qui fait diminuer V_1 et impose en « a » un potentiel inférieur au seuil de basculement.

En TTL : V_{IL} = 0.8 V. En CMOS $V_{TH} = V_{DD} / 2$. .

a = 0 donc b = .1. et c = .0.

En TTL Vc = V_{OH} - V_{IL} = 1.6V. En CMOS Vc = $V_{DD} / 2$.

le changement d'état de c impose brusquement un potentiel en V_1

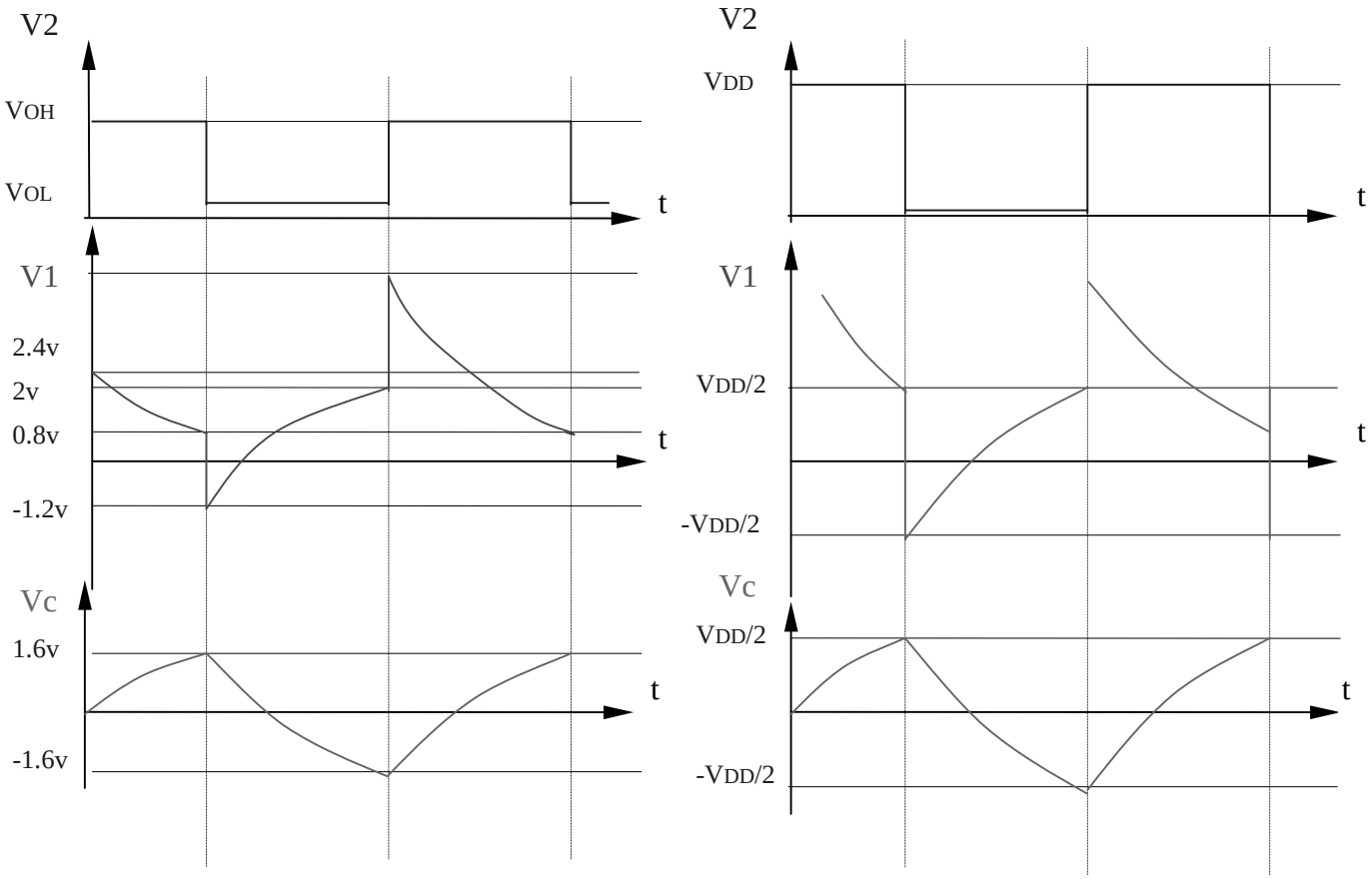
En TTL V_1 = -1,2V. En CMOS V_1 = - $V_{DD} / 2$.

La capacité se charge en sens inverse jusqu'à ce que V_1 atteigne :

. 2V . . en TTL . $V_{DD} / 2$. en CMOS

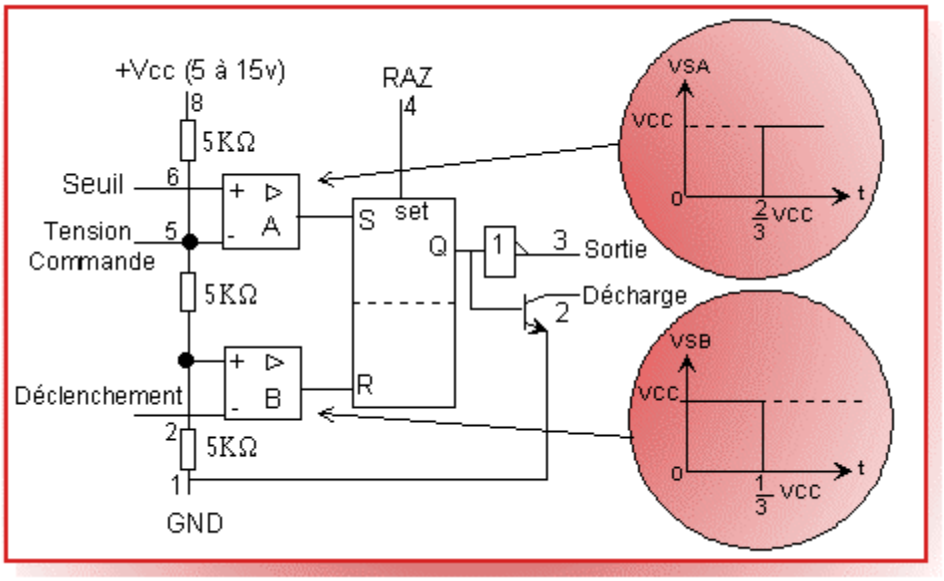
A ce moment le système bascule dans l'état d'origine et le cycle recommence.

Chronogrammes.

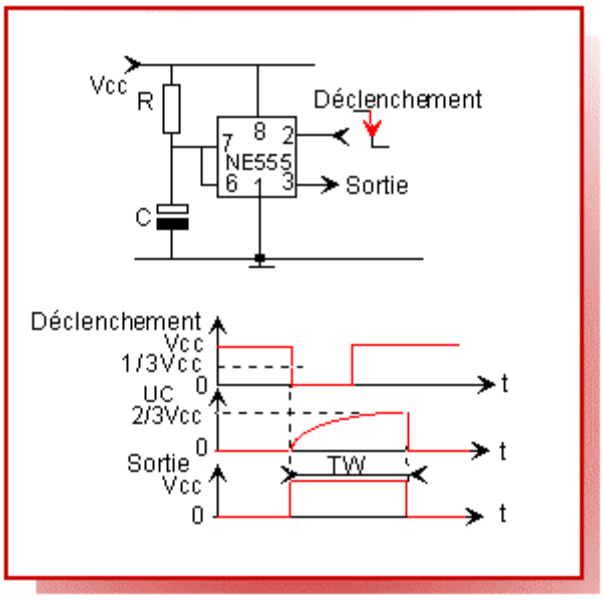


11.8. Oscillateur à circuit intégré spécialisé NE 555.

a) Schéma interne du NE555



b) Fonctionnement en monostable

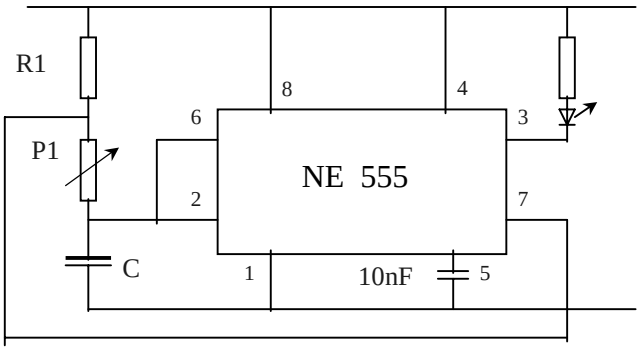


Principe :

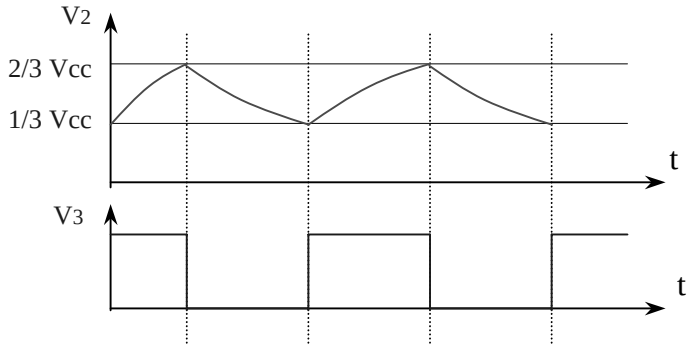
Chaque fois qu'une impulsion négative est envoyée sur l'entrée "Trigger", on obtient en sortie une tension calibrée de durée : $TW = R \cdot C \ln 3$

c) Fonctionnement en astable

Montage .



Fonctionnement.



A la charge de C1, la tension va augmenter jusqu'à 2/3 Vcc, la bascule passe à «1», d'où décharge de C1, au passage de 1/3 Vcc, la sortie change d'état, d'où nouvelle charge de C1, et le cycle recommence. La charge de C1 s'effectue à travers R1 et P1, sa décharge à travers P1.

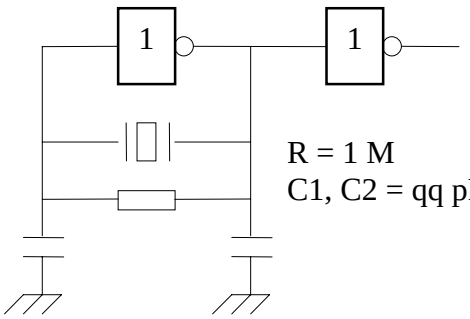
Valeur de la période T : $T = C1 . (R1 + 2. P1) \ln 2$

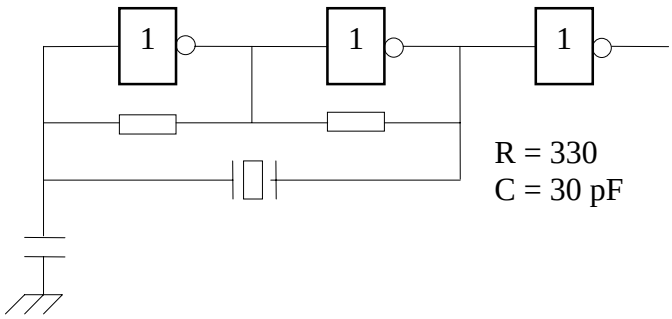
11.9. Oscillateur à Quartz.

Les oscillateurs à quartz ont une fréquence d'oscillation trés élevée et précise. Ils servent d'horloge pour les systèmes informatiques.

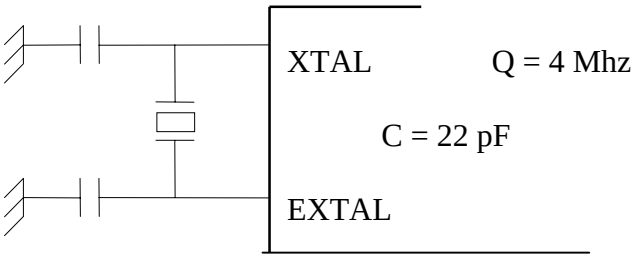
Principe: Soumise à un champ électrique, une mince plaquette de quartz oscille spontanément avec une trés grande précision. L'oscillation doit être entretenue par un circuit extérieur.

Montages : en CMOS en TTL





Pour le microprocesseur 6809.



La précision des quartz étant au minimum de 10^{-6} , les oscillateurs obtenus sont donc trés précis.

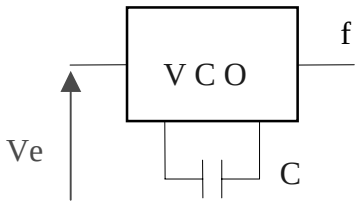
Il existe également des circuits intégrés comprenant un oscillateur Piezo-électrique.

11.10. Oscillateur contrôlé en tension.

La fréquence d'oscillation de certains oscillateur peut être controlée par une tension;

V C O : Voltage – Controled - Oscillator.

Il existe une relation (linéaire ou non) entre la fréquence de sortie et la tension de commande en entrée.



pour C donnée

f = k Ve si linéaire.

Utilisation : On trouve des convertisseurs de ce type dans les onduleurs.

11.11. Oscillateur à asservissement de phase.

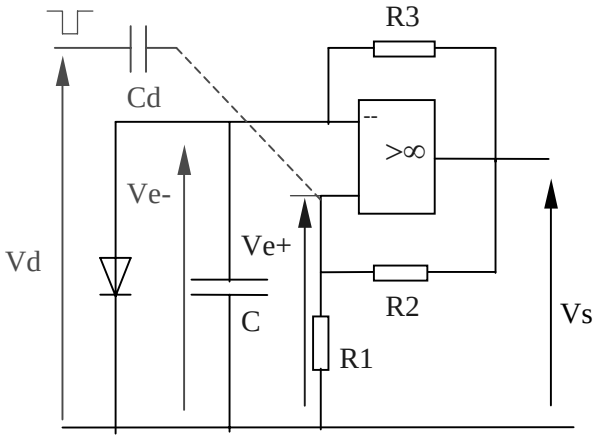
P L L : Phase locked Loop (boucle à verrouillage de phase).

Ce type de circuit permet d'obtenir une fréquence d'oscillation élevée synchronisée sur un signal de fréquence plus faible.

Mise en oeuvre d'un 4046 : On veut disposer d'une fréquence multiple de 25,6 khz et de 150, de plus cette fréquence doit être synchronisée sur le réseau 5O hz

Annexe1.

b) Montage dérivé de l'ASTABLE



Etat stable pour : VS = +Vsat

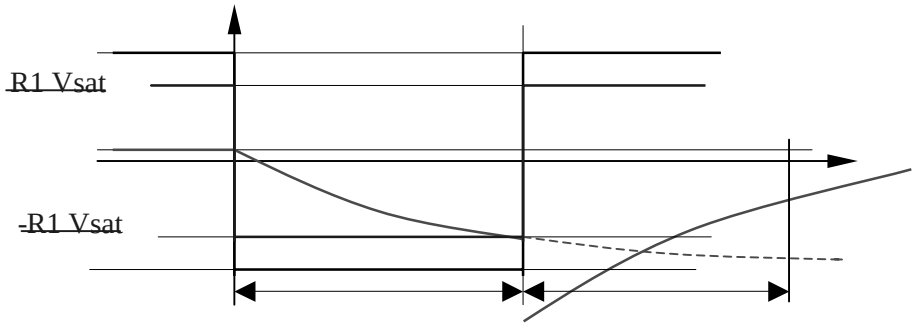
Déclenchement par un circuit dérivateur sur l'entrée e+

il faut : Ve+ < Ve- (φ 0.7 V)

d'ou $Vd < \left| \left(R1 / R1 + R2 \right) Vsat - Vo \right|$

$$T = R3\,C\,\ln\left[\left(1+\frac{R1}{R2}\right)\left(1+\frac{Vo}{Vsat}\right)\right]$$

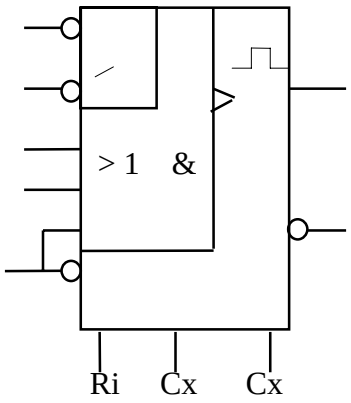
Chronogramme

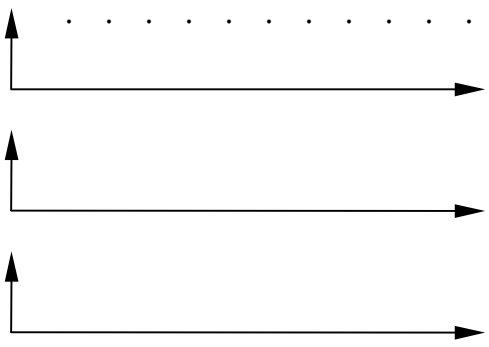


Remarque :

on peut prendre $R_1 = R_2 = R$ on a alors : $$T = R3\,C\,\ln\left[2+\left(\frac{2Vo}{Vsat}\right)\right]$$

Mise en oeuvre d'un 74122 en multivibrateur à 100 khz. . . . . .





12. Fonctions principales de la logique combinatoire [5]

Les composants utilisés jusqu'à maintenant (ET, OU, NON-ET, XOR, ...) faisaient partie de la catégorie SSI *(Small Scale Integratio*n). Principalement à cause de leur simplicité, ce sont
les premiers circuits intégrés à avoir été réalisés avec succès au tout début de l'ère électronique
moderne. Les progrès techniques réalisés en conception de circuits intégrés ont permis de concevoir des circuits un peu plus complexes permettant de réaliser des fonctions plus générales.
Ces circuits d'intégration moyenne (MSI – *Medium Scale Integratio*n) sont présentés dans cette section.

12.1. Le multiplexeur (MUX)

Le mult iplexeur est un système combinatoire ayant pour fonction de sélectionner une Parmi 2^n entrées et de la transmettre à la sortie. La sélection est faite à l'aide de *n* lignes d'adresse. La notation usuelle du MUX est: MUX 2^n à 1. Par exemple, un MUX 8 à 1 aura 3 lignes d'adresse. La figure 12.1 présente la forme générale d'un MUX.

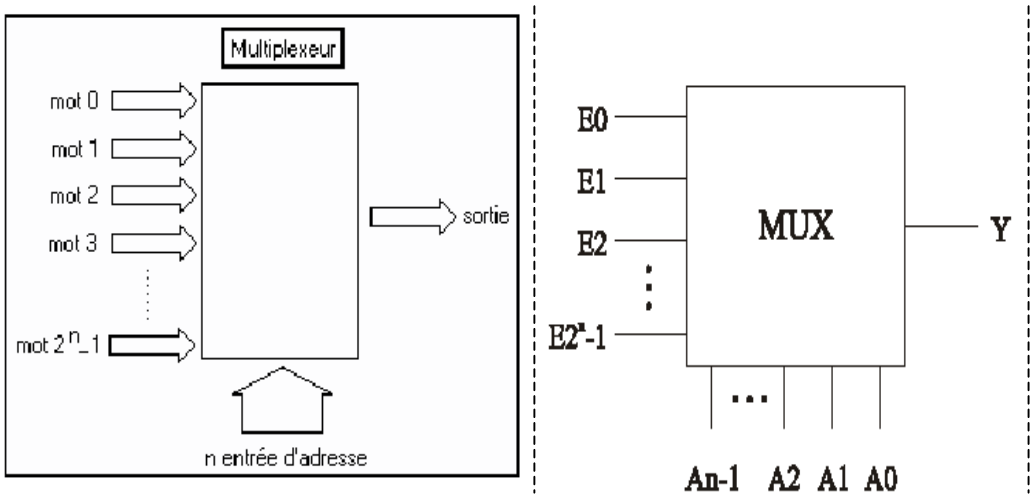
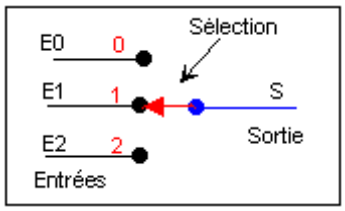


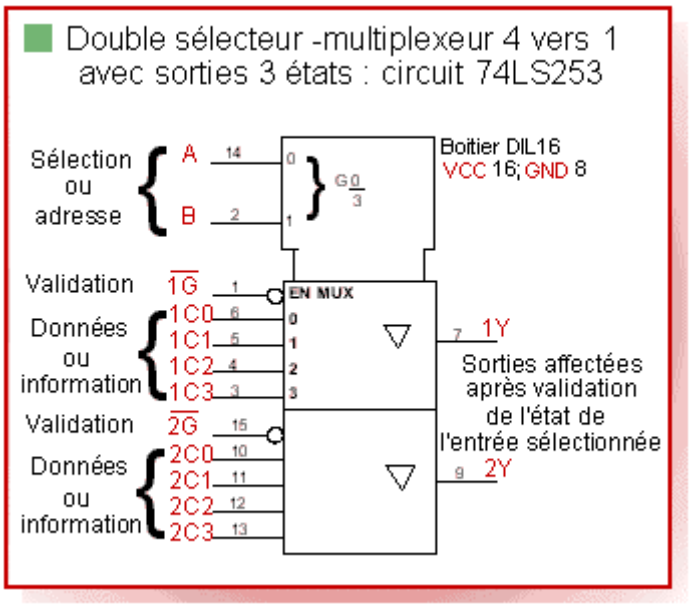
Schéma de principe d'un multiplexeur

Figure.12.1



Les mots d'entrée comportent autant de bits que la sortie.

En effet, le multiplexage consiste à envoyer sur une même ligne de transmission des informations provenant de sources différentes.

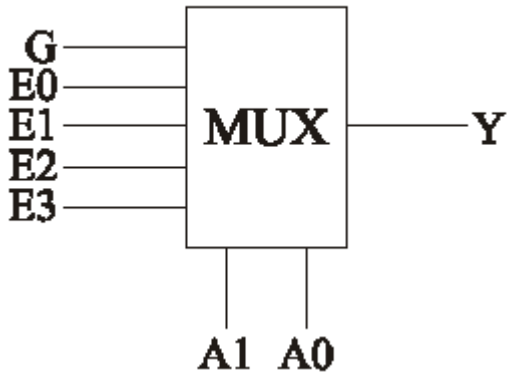


Condition d'emploi

deux entrées de sélection ou adresse, A et B, qui permettent de sélectionner une entrée parmi les quatre. deux entrées de validation /1G et /2G qui permettent chacune de valider l'entrée sélectionnée, c'est-à-dire d'en affecter son état sur sa sortie correspondante.

deux fois quatre entrées de données. deux sorties « 3 états » séparées 1Y et 2Y.

La figure 12.2 montre le composant 74153 qui contient deux MUX 4 à 1. Le signal supplémentaire *G (Strob*e) est un signal d'activation du composant. Si *G* est inactif, la sortie du MUX sera obligatoirement inactive.



G	A1	A0	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	E0
1	0	1	E1
1	1	0	E2
1	1	1	E3

Figure 12.2 : Caractéristiques du 74153.

12.2. Le démultiplexeur.

Le démultiplexeur (DEMUX)

Le démultiplexeur est un système combinatoire ayant pour fonction de transmettre une entrée vers une des 2 n sorties. La sélection est faite à l'aide de *n* lignes d'adresse et les sorties sont mutuellement exclusives. La notation usuelle du DEMUX est: DEMUX 1 à 2^n. Par exemple, un DEMUX 1 à 8 aura 3 lignes d'adresse. La figure 12.3 présente la forme générale d'un DEMUX.

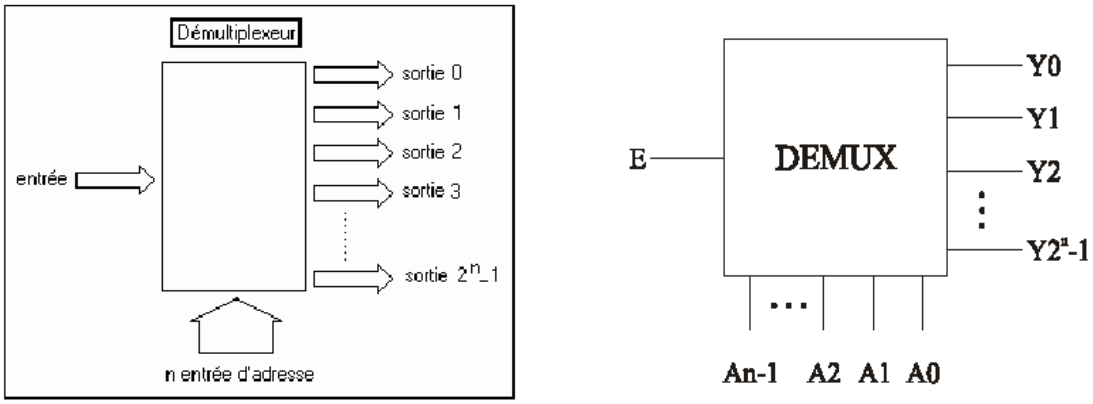
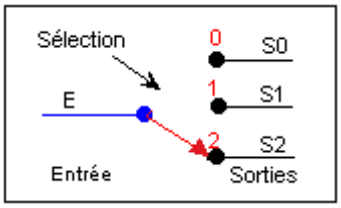
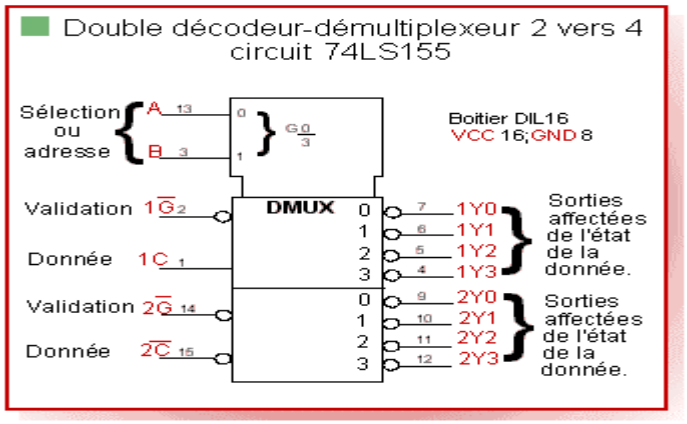


Figure 12.3: Représentation générale d'un démultiplexeur.

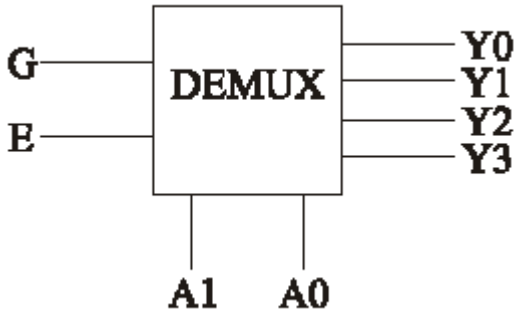


En effet, le démultiplexage consiste à répartir sur plusieurs lignes à des fins d'exploitations différentes, des informations qui arrivent en série sur une même ligne.



Ce démultiplexeur réalise l'aiguillage de deux fois une entrée, vers quatres sorties, d'où sa désignation de double démultiplexeur de deux fois une ligne vers quatre.

La figure 12.4 montre un DEMUX 1 à 4. Le signal supplémentaire *G (Strob*e) est un signal d'activation du composant. Si *G* est inactif, toutes les sorties du DEMUX seront obligatoirement inactives.



G	A1	A0	Y
0	0	0	Yn inactifs
0	0	1	Yn inactifs
0	1	0	Yn inactifs
0	1	1	Yn inactifs
1	0	0	Y0=E
1	0	1	Y1=E
1	1	0	Y2=E
1	1	1	Y3=E

Figure 12.4: Caractéristiques d'un démultiplexeur.

Ce circuit permet de transmettre un mot logique à un canal dont le numéro sera choisi par l'entrée d'adresse.

12.3. L'encodeur

L'encodeur est un système combinatoire ayant pour fonction de retourner l'index d'activation d'une parmi 2 n entrées. L'index d'activation est donné sur *n* lignes d'adresse. Lorsque plusieurs entrée sont activées, l'encodeur accorde la priorité à l'entrée dont l'index est supérieur.

La notation usuelle de l'encodeur est: encodeur 2 n à n. Par exemple, un encodeur 8 à 3 aura 8 entrées et 3 lignes d'adresse en sortie. La figure 11.5 présente la forme générale d'un encodeur.

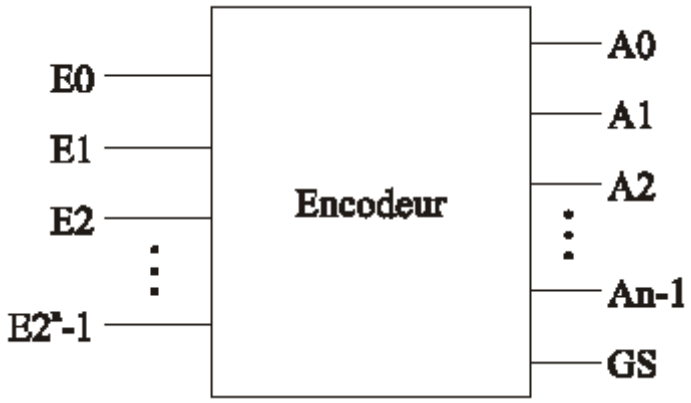
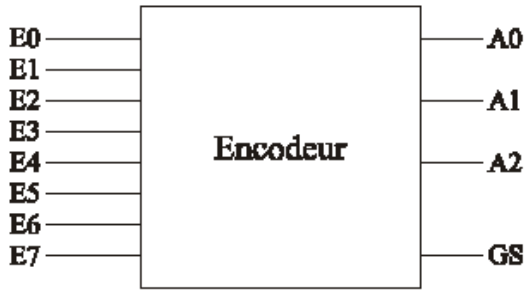


Figure 12.5 : Représentation générale d'un encodeur.

La figure 12.6 montre une simplification du composant 74148 qui contient un encodeur 8 à 3. Le signal supplémentaire *GS (Got Something*) est un signal qui indique qu'une des entrées est active dans le but de faire la différence entre l'entrée 0 active et lorsqu'aucune entrée n'est activée.



E7	E6	E5	E4	E3	E2	E1	E0	GS	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	X	1	0	0	1
0	0	0	0	0	1	X	X	1	0	1	0
0	0	0	0	1	X	X	X	1	0	1	1
0	0	0	1	X	X	X	X	1	1	0	0
0	0	1	X	X	X	X	X	1	1	0	1
0	1	X	X	X	X	X	X	1	1	1	0
1	X	X	X	X	X	X	X	1	1	1	1

Figure 12.6: Caractéristiques du 74148.

12. 4. Le décodeur

Le décodeur est un système combinatoire ayant pour fonction d'activer une des 2 n sorties. La sélection est faite à l'aide de *n* lignes d'adresse et les sorties sont mutuellement exclusives. La notation usuelle du décodeur est: décodeur 1 parmi 2 n . Le décodeur se comporte exactement comme un DEMUX avec son entrée toujours à 1. Par exemple, un décodeur 1 parmi 8 aura 3 lignes d'adresse. La figure 12.7 présente la forme générale d'un décodeur.

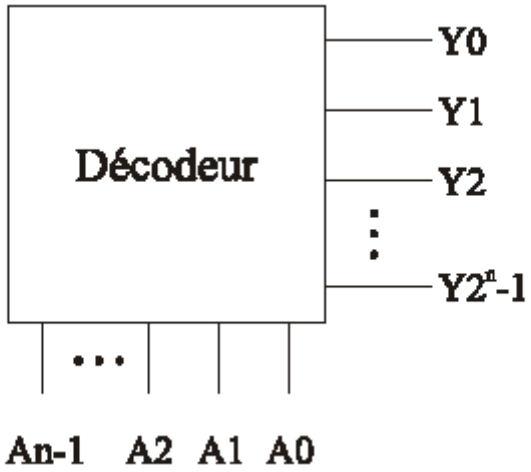
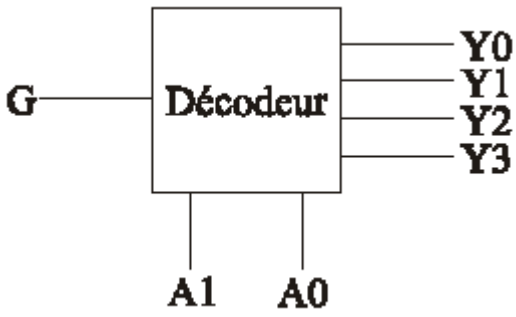


Figure 12.7: Représentation générale d'un décodeur.

La figure 12.8 montre un décodeur 1 parmi 4. Le signal supplémentaire *G (Strob*e) est un signal d'activation du composant. Si *G* est inactif, toutes les sorties du DEMUX seront obligatoirement inactives.



G	A1	A0	Y
0	0	0	Yn inactifs
0	0	1	Yn inactifs
0	1	0	Yn inactifs
0	1	1	Yn inactifs
1	0	0	Y0=1
1	0	1	Y1=1
1	1	0	Y2=1
1	1	1	Y3=1

Figure 12.8: Caractéristiques d'un décodeur 1 parmi 4.

12.5. Les mémoires

Il existe une très grande quantité de catégories de mémoires. Les plus connues sont les ROM *(Read Only Memor*y) et les RAM *(Random Access Memor*y). Une mémoire est un réseau combinatoire ayant pour fonction d'emmagasiner de l'information. La mémoire est une matrice de cellules capable d'emmagasiner une information unitaire (bit) qui a une adresse spécifique unique. Les cellules ont souvent une largeur *m* supérieure à 1. L'accès aux cellules se fait à l'aide de *n* bits d'adresse pour une mémoire d'une capacité de m•2 n bits. La différence entre les deux catégories principales est que les ROM ne peuvent être que lues tandis que les RAM peuvent être utilisées en écriture tout aussi bien qu'en lecture. La figure 12.9 montre le schéma général d'une ROM et d'une RAM. Le signal *OE (Output Enabl*e) permet l'activer les sorties. Si *OE* n'est pas actif, les sorties sont électriquement inactives, ce qui signifie qu'il n'y a ni un "0" ni un "1" de présent. Le signal *WE (Write Enabl*e) permet d'écrire des données dans une RAM. ◦ ce moment, le signal *OE* doit être inactif pour permettre aux données d'arriver à la mémoire sans conflits.
Même si les deux types de mémoires ne diffèrent que par le signal *W*E, le fonctionnement interne est différent.

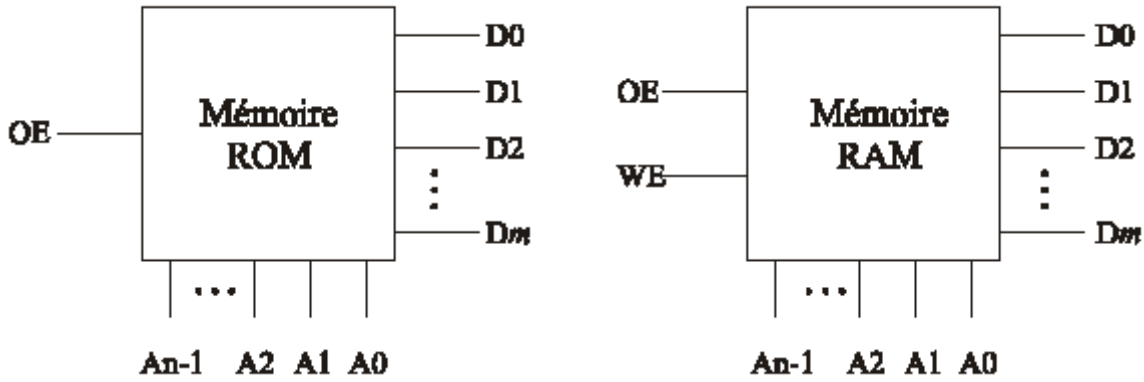


Figure 12.9: Représentation générale des mémoires ROM et RAM.

12.6. Les registres

Contrairement aux composants MSI vus jusqu'ici, les registres sont ne sont pas des systèmes combinatoires, ce sont des composants permettant d'emmagasiner de l'information. Pour une même combinaison d'entrées, ce type de composant peut produire des sorties différentes, selon l'information emmagasinée. Il existe deux types de registres: les bascules *(flip-flo*p)
est les bistables *(latch*s) qui diffèrent par leur signal d'activation. Les bistables ont un signal d'activation qui est basé sur un niveau logique tandis que le signal d'activation des bascules est sensible à une transition de niveau. Lorsque le signal d'activation d'une bistable est actif, tout ce qui est à l'entrée est transféré directement à la sortie. Lorsque le signal d'act ivat ion devient inactif, la sortie conserve sa valeur jusqu'à ce que le signal redevienne actif. Avec une bascule, le comportement est légèrement différent. L'information à l'entrée est copiée à la sortie sur une transition spécifique du signal d'activation, habituellement la transition du "0" vers le "1". La
figure 12.10 présente la forme générale d'une bistable et d'une bascule ainsi qu'un diagramme de temps pour chacun.

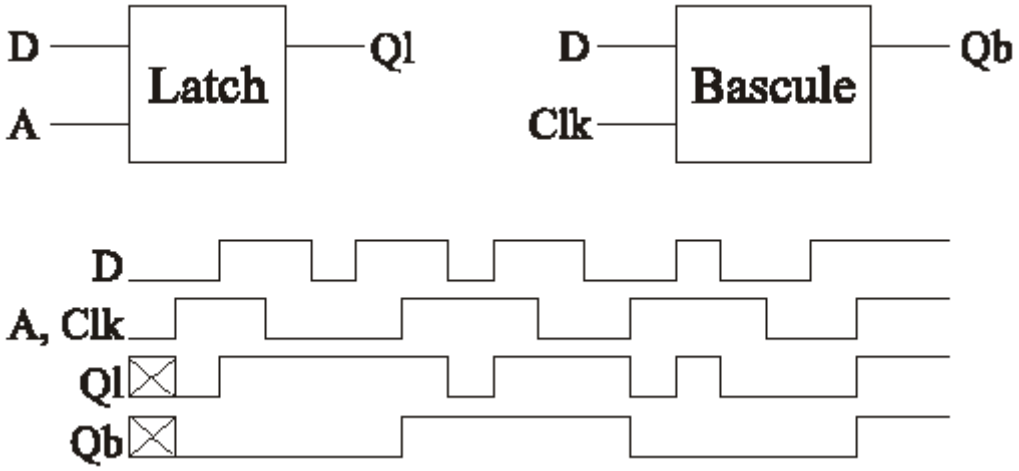


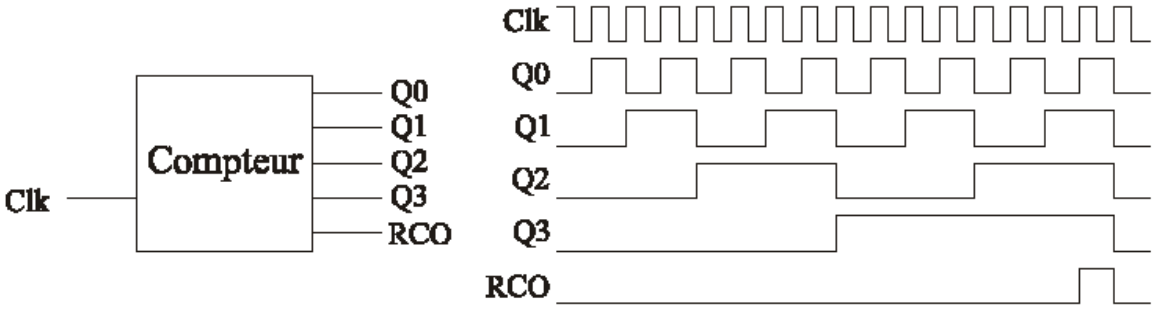
Figure.12.10

12.7 Les compteurs

Comme dans le cas des registres, les compteurs ne sont pas des systèmes combinatoires, ce sont des composants permettant d'emmagasiner et de traiter de l'information.

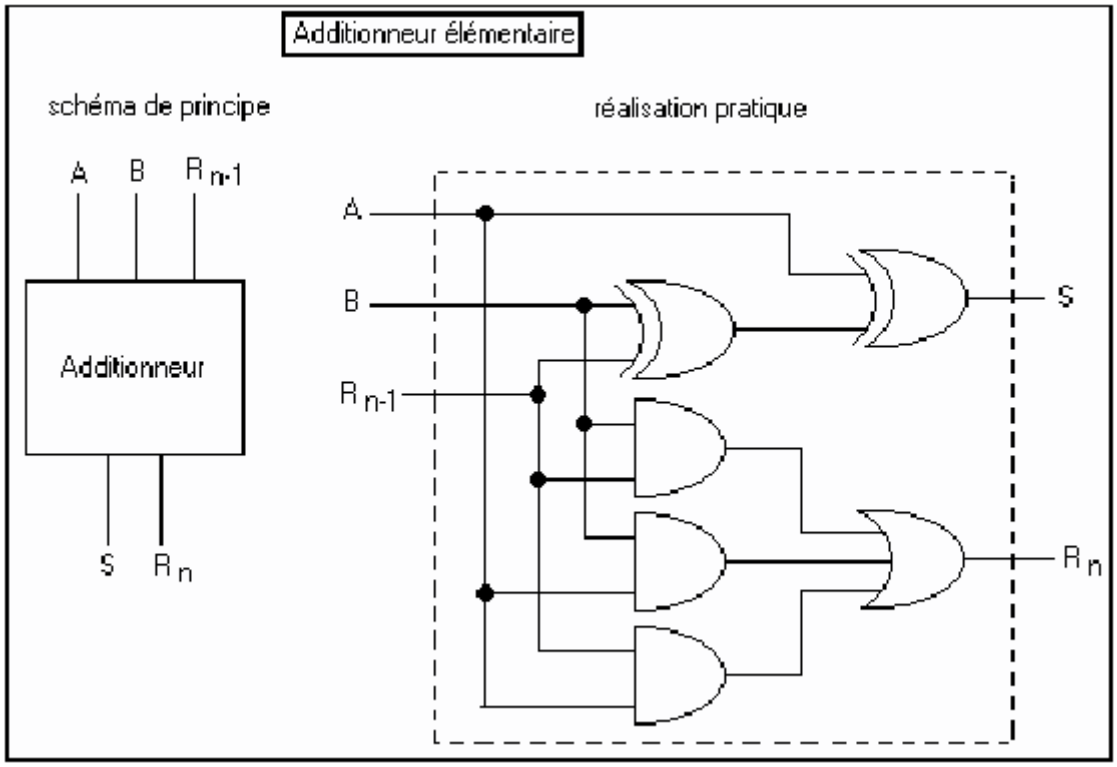
Pour une même combinaison d'entrées, ce type de composant peut produire des sorties différentes, selon l'information emmagasinée. Le compteur dispose de deux signaux importants:

le signal de compte est le résultat qui est représenté sur *n* bits. Sur une transition spécifique du signal de compte, habituellement la transition du "0" vers le "1", un incrément est de 1 est ajouté au résultat. Même si ma majorité des compteurs suit un ordre qui est binaire naturel, il est possible de trouver des compteurs avec d'autres séquences. Lorsque le compteur est rendu à la limite représentable de sa séquence, certains compteurs ont un signal *TCO (Terminal Count Output*) ou *RCO (Ripple Carry Output*) qui signifie qu'un débordement aura lieu au prochain incrément. La figure 1.10 présente la forme générale d'un compteur 4 bits ainsi qu'un diagramme de temps.

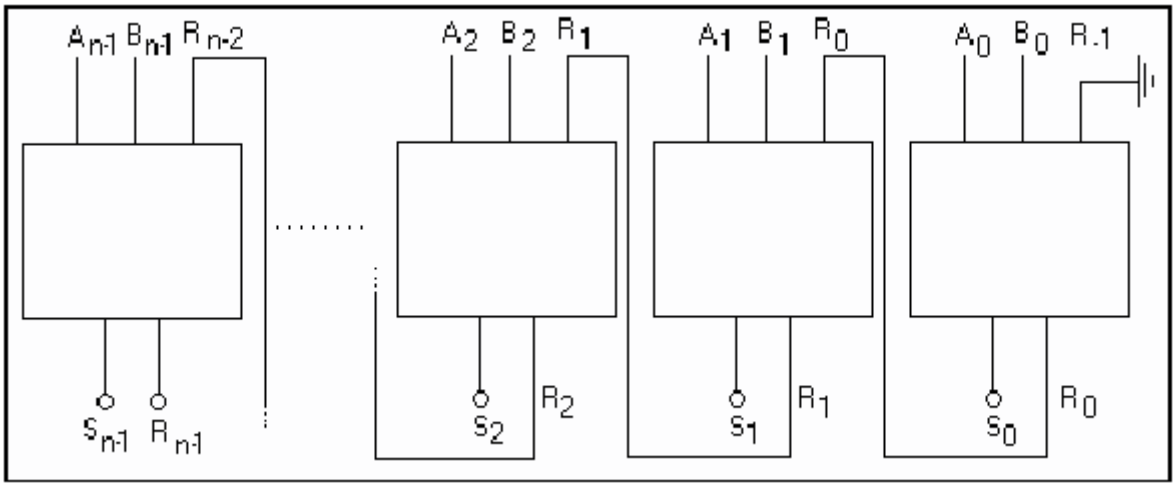


13 . Opérateurs arithmétiques et logiques.[14]
Un calculateur doit pouvoir effectuer toutes les opérations arithmétiques élémentaires (addition, soustraction, multiplication et division). En fait addition et soustraction suffisent, car la multiplication est une addition répétée et la division une soustraction répétée.

13.1. L'additionneur.
. Addition sur deux bits: Nous allons raisonner à partir de la structure élémentaire suivante



En sortie S, on récupère la somme de A et B. Rn-1 est l'entrée de retenue alors que Rn sera la retenue obtenue par addition de A,B et Rn-1.
Addition sur N bits:
On utilise N fois l'élément précédent en réalisant l'association suivante:



La sortie (S0,S1, …, Sn-1) est la somme de (A0, A1,…,An-1) avec (B0, B1,…,Bn-1). Il faut noter que la somme de 2 mots codés sur n bits ne se code pas forcément sur n bits, d'où la retenueRn-1 qui permet de donner le résultat sur n+1 bits.

13.2. Le soustracteur.

On va considérer que l'on réalise l'addition d'un nombre positif et d'un nombre négatif. Reste à définir un nombre négatif.

. Nombre négatif – complémentation à 2.

Considérons un nombre binaire de n bits K. On vérifie aisément que

$$K + \overline{K} + 1 = 2^n = 0 \quad \text{modulo } 2^n$$

On appelle complément à 2 le nombre logique

$$\overline{K} + 1 = 2^n - K = -K \quad \text{modulo } 2^n$$

K K 2 1 K .=.=+modulo 2 . Nombre signés.

Pour pouvoir représenter des nombres négatifs, on va adopter les conventions suivantes. Un bit représentera le signe (0 si le nombre est positif, 1 s'il est négatif), alors que les autres bits représenterons classiquement la valeur absolue pour un nombre positif et son complément à deux pour un nombre négatif.

Pour un nombre quelconque, on a donc

$A>0$ représenté par $0/a$

$A<0$ représenté par $1/\overline{a}+1=\overline{A}+1$

La complémentation à 2 donne bien le bon bit de signe (1 sur le bit de poids fort grâce à 2^n).

. Soustraction de A et B.

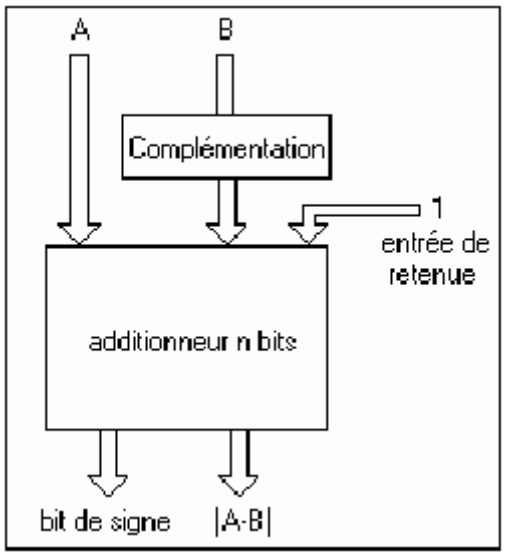
On va additionner A et le complément à 2 de B et considérer deux cas.

Si A>B, on a $A + \overline{B} + 1 = 0/a + 1/\overline{b} + 1 = 0/a + 1/2^n - b = 0/a - b$

Si A<B, on a $A + \overline{B} + 1 = 0/a + 1/\overline{b} + 1 = 0/a + 1/2^n - b = 1/\overline{b-a} + 1$ car a-b<0

. Le soustracteur est donc réalisé simplement au moyen d'un additionneur dans lequel la retenue d'entrée est forcée à 1 et d'un dispositif qui complémente le nombre à soustraire.

On réalise le soustracteur de la façon suivante:



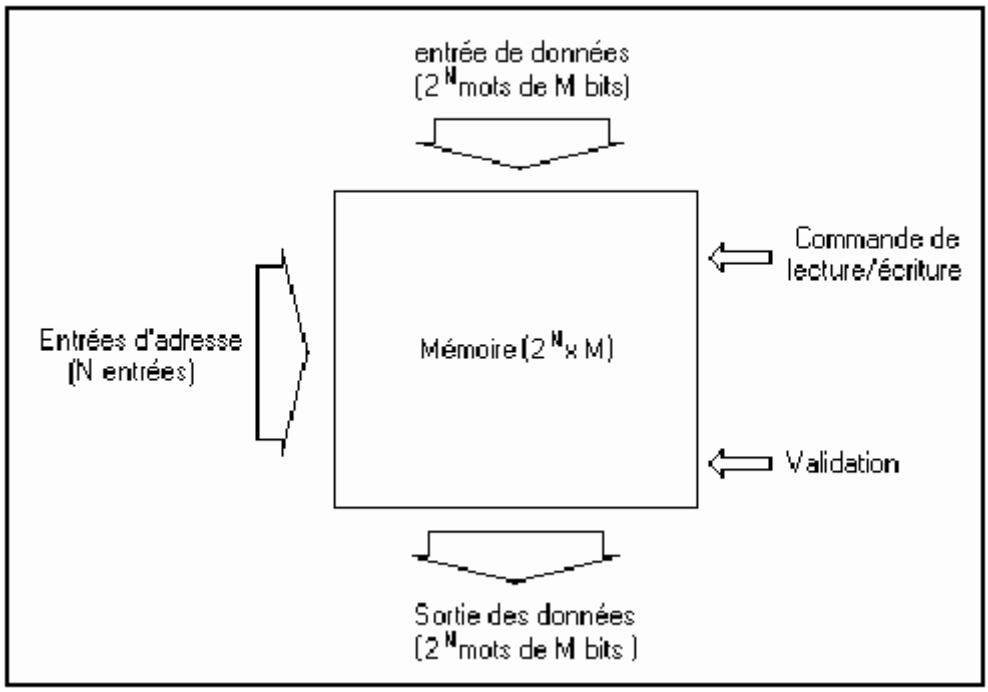
Suivant la valeur du bit de signe, on récupère directement le résultat si celui-ci est 0 et son complément à 2 s'il vaut 1.

13.3. Les mémoires mortes.

C'est l'un des éléments de base pour tout calculateur. C'est là que seront souvent stockées les données à traiter ainsi que le résultats du traitement. Les mémoires mortes vont conserver l'information, même lorsqu'on cessera de les alimenter (l'information est figée dans la structure). Il en existe plusieurs sortes (structures différentes, caractère effaçable ou non…etc…).

13.3.1 Structure générale d'une mémoire.

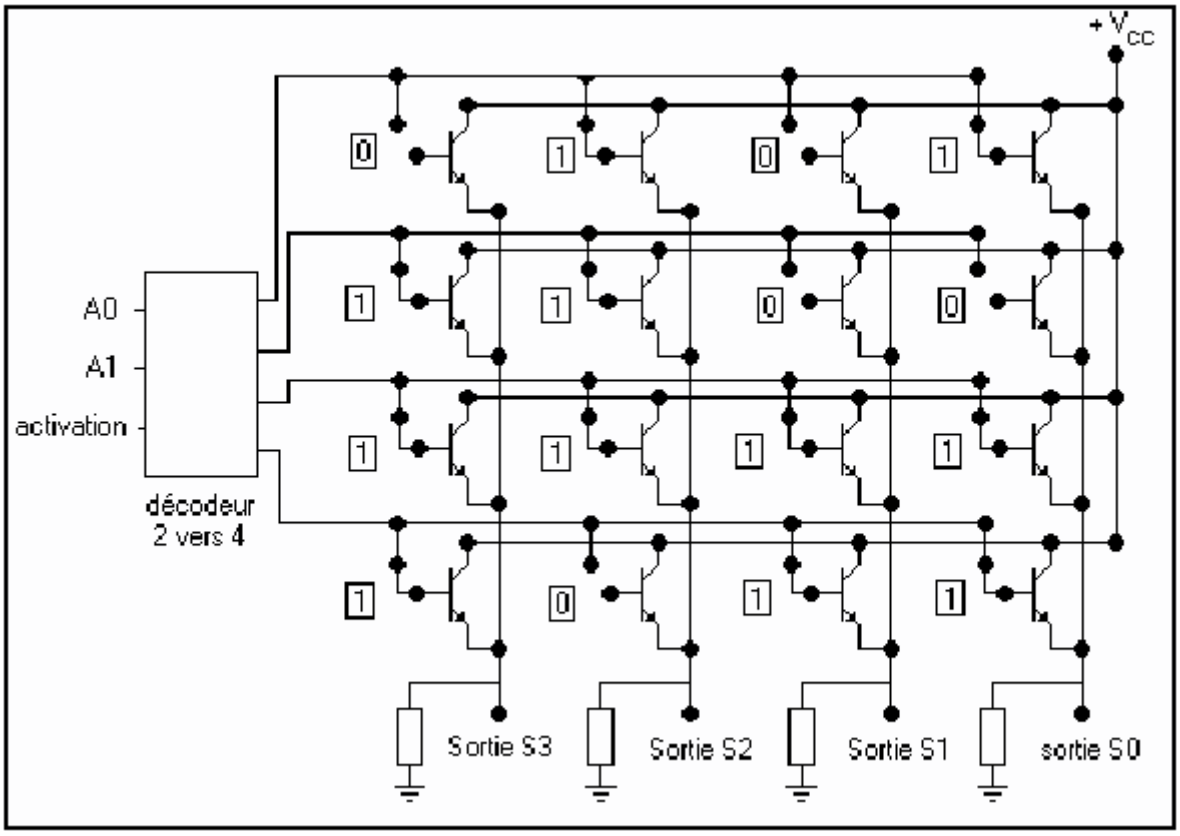
Nous allons prendre l'exemple d'une mémoire pouvant stocker 2^N mots de M bits (il faut N entrées d'adresse pour disposer de 2^N adresses possibles pour les mots). La structure générale pour une mémoire de cette capacité est la suivante:



Dans un premier temps, il faut pouvoir rentrer les données. Pour cela, il faut valider l'écriture et rentrer les mots aux endroits définis par les adresses (cases mémoires). Cette étape étant réalisée, on peut décider de lire les informations stockées. On valide alors la lecture et on récupère en sortie la case mémoire demandée en adresse.
L'entrée de validation permet de bloquer le fonctionnement de la mémoire. Celle-ci refuse alors de répondre aux sollicitations extérieures.
rq: Cette structure générale est valable pour les mémoires mortes (systèmes combinatoires) mais elle l'est aussi pour les mémoires vives (Systèmes séquentiels). C'est essentiellement la façon de stocker l'information dans les cases mémoires qui va différer.

13.3.2. Les ROM ("read only memory").

Ce sont des mémoires à semi-conducteurs qui permettent de stocker en permanence des données qui ne seront que très rarement modifiées par la suite. Les données sont rentrées dès la fabrication en utilisant un masque adapté (MROM) ou par programmation (PROM). Dans certains dispositifs, il est possible d'effacer les données rentrées pour les modifier (EPROM).
. A titre d'exemple, on donne la structure interne d'une MROM à 4 mots (2 entrées d'adresse de 4 bits) réalisée à base de transistors bipolaires. Une base non raccordée permet de mémoriser un 0 (transistor ouvert) alors qu'une base raccordée permet de mémoriser un 1 (transistor fermé). Le résultat en sortie est celui de la ligne validée par le décodeur. L'entrée d'activation permet de bloquer la sortie quelle que soient les adresses d'entrée.

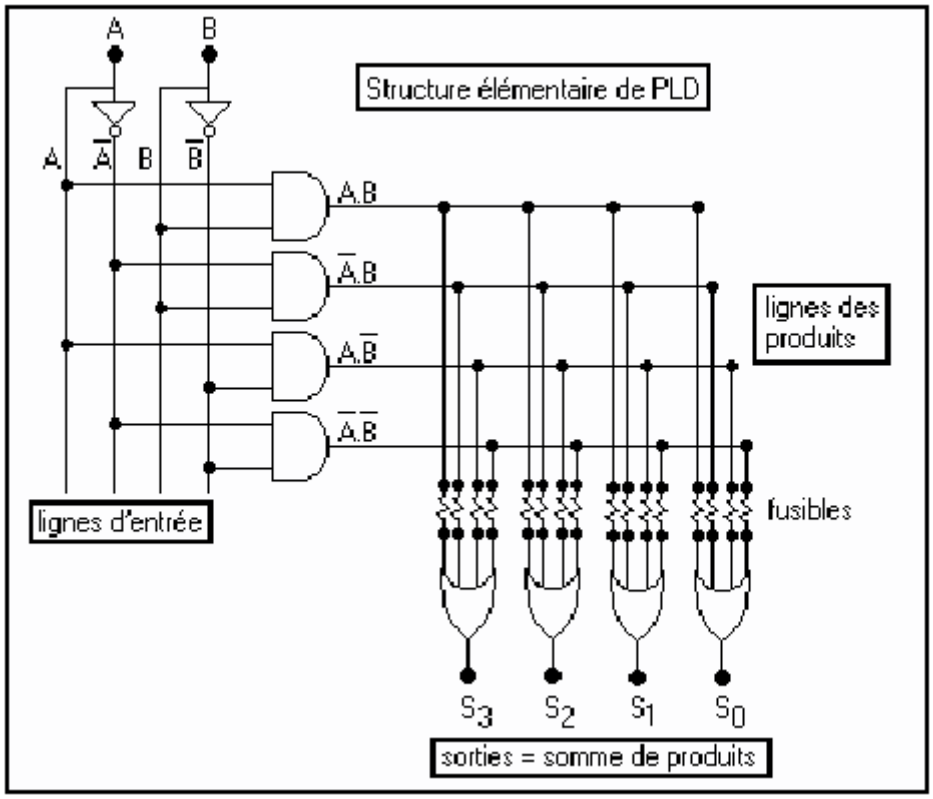


. Pour réaliser la même structure en PROM, la base est reliée par un fusible. Programmer va consister à griller ou non ce fusible afin de mémoriser un 1 ou un 0. Une fois le fusible grillé, on peut plus rien changer. Là encore, le circuit n'est plus effaçable.
. Dans les EPROM, les cellules de stockage sont réalisées à base de transistors à effet de champ munis d'une grille non reliée (flottante). Dans son état normal, le transistor est bloqué et la valeur stockée est un 1. Si on veut mémoriser un 0 à la place, on envoie des charges sur la grille en appliquant, au moyen d'un dispositif externe une forte tension (elles ne peuvent pas s'écouler puisque la grille n'est pas reliée). Ces charges vont rendre le transistor passant ce qui permettra de récupérer un 0 à la place du 1 initial. Si on veut changer les donnée rentrée, on peut ramener la mémoire à son état initial (des 1 partout) en la soumettant un un rayonnement UV qui va créer un photocourant entre la grille et le substrat (les charges ayant disparu, le transistor est à nouveau bloqué). Les nouvelles données sont rentrées comme précédemment.
. Application:
- stocker les points d'une forme de signal donnée (sinus, triangle…etc) pour un GBF.
- stockage de programme d'amorçage (pour aller chercher le système d'exploitation sur le disque dur d'un PC)

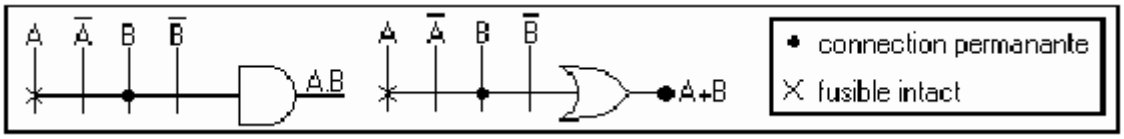
13.3.3. Les composants logiques programmables.
Nous avons vu qu'il existait des circuit combinatoires permettant de réaliser des fonctions complexes données (multiplexage, addition, etc…). On peut ainsi réaliser, au moyen d'un seul composant, ce qui demanderait d'utiliser un grand nombre de portes élémentaires.
Les composants logiques programmables (PLD), permettent de réaliser des applications complexes personnalisées (autres celles que nous venons d'énoncer), au moyen d'un seul composant. Ce dernier renferme un très grand nombre de transistors et de portes raccordés entre eux . Certaines liaisons sont des fusibles qui peuvent être fondus par l'intermédiaire d'un dispositif extérieur. On a donc bien un composant programmable.

De par leur structure standardisée (quel que soit la fonction à réaliser, on part du même élément), ces composants sont peu coûteux et offrent une grande souplesse dans la réalisation de circuits combinatoires (on réalise directement ce dont on a besoin). On les emploiera donc de préférence à des associations de portes discrètes qui sont techniquement fastidieuses à mettre ne oeuvre.

. Structure générale: La structure de base de tout circuit programmable se présente sous la forme suivante



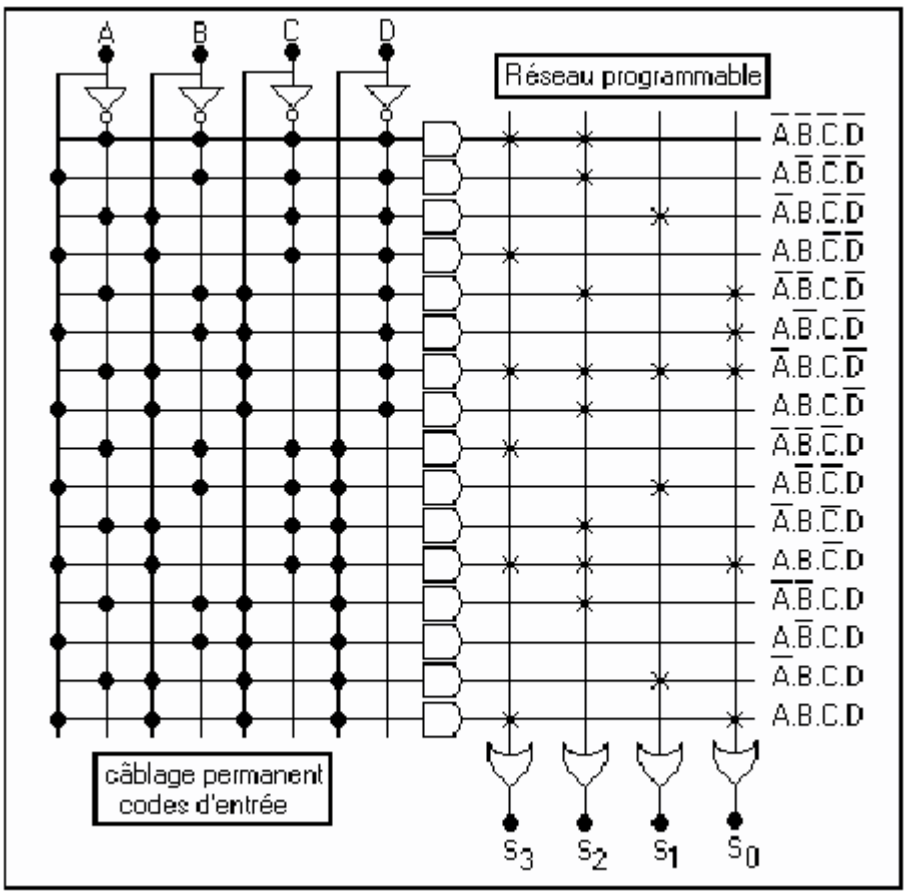
On observe un réseau de portes AND, un réseau de portes OR capable de fournir n'importe quelle combinaison des deux variables A et B ainsi que de leurs compléments.
On remarque les fusibles en entrée des portes OR. Suivant les fusibles que l'on choisit de griller, on obtiendra différentes les valeurs choisies en sortie.
Plus le nombre d'entrées sera grand, plus le circuits mettra en jeux un grand nombre de portes intégrées.
Par la suite, on utilisera le formalisme suivant, afin de simplifier les schéma de principe et de les rendre plus lisibles. On ne fait apparaître qu'une seule ligne d'entrée pour les portes sur laquelle on trouve toutes les données et on représente différemment les fusibles et les liaisons permanentes.



Nous allons maintenant nous intéresser aux différents types de circuits programmables.

. Les PROM:

Dans ce type de circuit, le réseau si tué en entrée des AND est figé (on a tous les codes d'entrée possibles). En revanche, l'entrée des OR est programmable.
Sur la figure suivante, on donne un exemple de circuit programmé réalisé avec une PROM permettant de sommer au maximum 16 produits de 4 bits.



En sortie, on obtient

$$S_0 = \overline{A}.\overline{B}.C.\overline{D} + A.\overline{B}.C.\overline{D} + \overline{A}.B.C.\overline{D} + A.B.\overline{C}.D + A.B.C.D$$

$$S_1 = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.C.\overline{D} + A.\overline{B}.\overline{C}.D + \overline{A}.B.C.D$$

$$S_2 = \overline{A}.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.B.C.\overline{D} + A.B.C.\overline{D} + \overline{A}.B.\overline{C}.D + A.B.\overline{C}.D + \overline{A}.\overline{B}.C.D$$

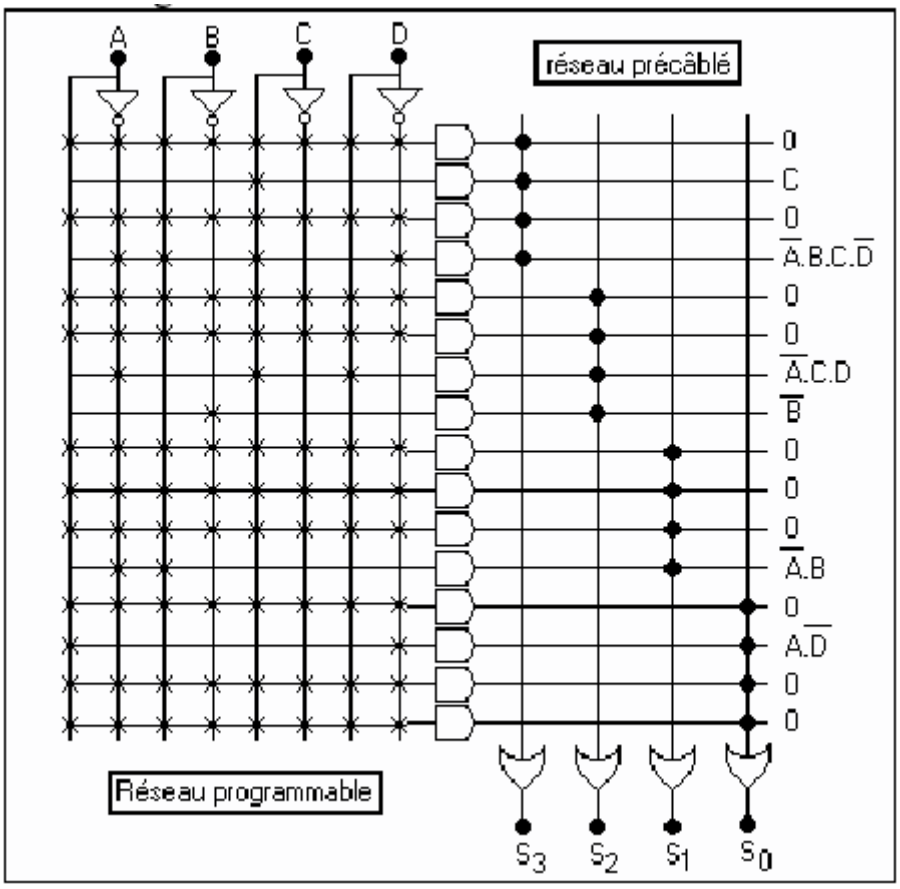
$$S_3 = \overline{A}.\overline{B}.\overline{C}.\overline{D} + A.B.\overline{C}.\overline{D} + \overline{A}.B.C.\overline{D} + \overline{A}.\overline{B}.\overline{C}.D + A.B.\overline{C}.D + A.B.C.D$$

Nous n'avons pas cherché à simplifier les expression, car cela n'a ici aucun intérêt à priori.

. Les PAL:

Le PAL ("programmable array logic") est constitué exactement comme le PROM, sauf que cette fois, le réseau précédent les AND sera programmable (et non précâblé), alors que le réseau précédent les OR sera précâblé.

La sortie qui nous intéresse se présente sous forme d'une somme de produits. Le réseau d'avant les AND permet de fabriquer les produits qui sont ensuite ajoutés par le réseau de OR. Dans le cas où l'on fait la somme, au maximum de 4 produits différents, on peut travailler avec le PAL présenté sur la figure suivante:



Sur l'exemple, les sorties sont

$$S_0 = A.\overline{D}$$

$$S_1 = \overline{A}.B$$

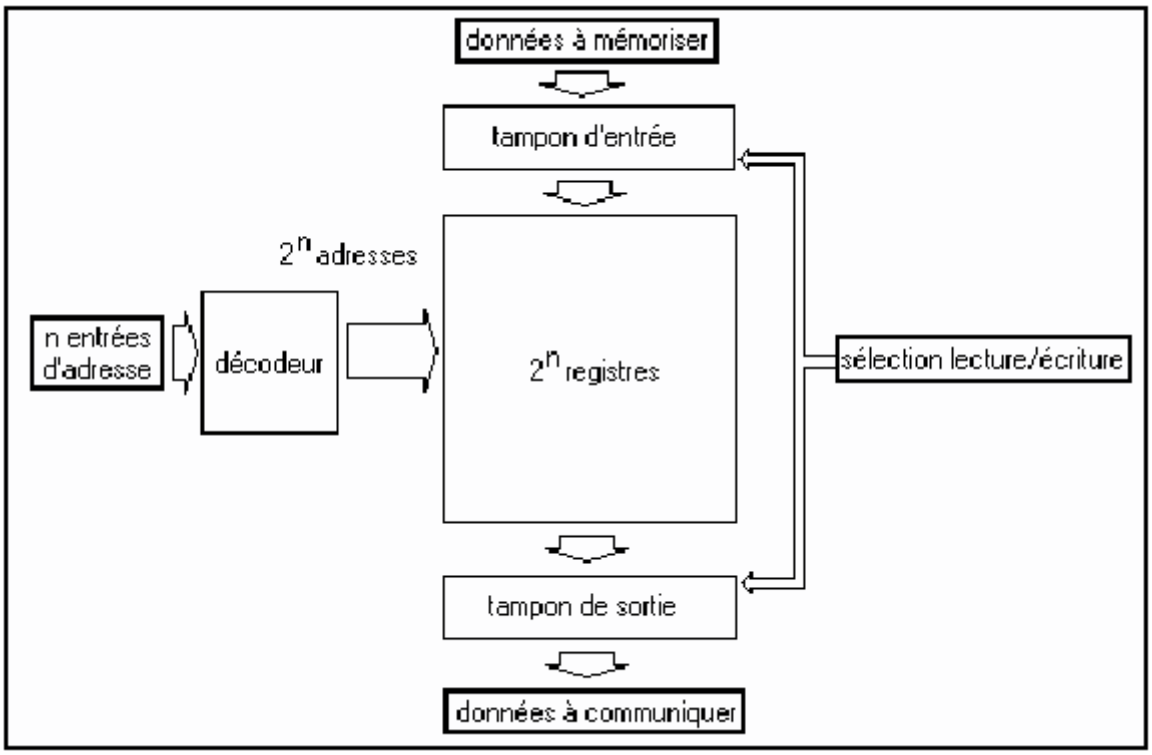
$$S_2 = \overline{A}.C.D + \overline{B}$$

$$S_3 = C + \overline{A}.B.C.\overline{D}$$

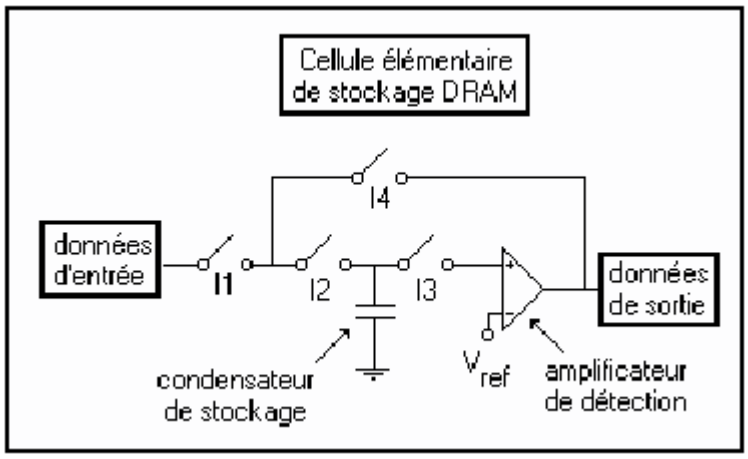
Les PLA ou FPLA ("field programmable array"):
La structure globale est la même que pour les PROM et les PAL, sauf que le réseau précédent les AND et celui précédent les OR sont tous les deux programmables. C'est donc l'élément qui confère la plus grande souplesse d'utilisation. Il est cependant plus complexe à fabriquer.
rq: Les éléments dont nous venons de parler ne sont pas effaçables, ce qui est préjudiciable en cas d'erreur de programmation. Il existe des PLD effaçables, appelés EPLD dont le principe de programmation et d'effacement est proche de celui des EPROM.

13.3.4. Les mémoires vives (RAM = "random access memory").

Lorsque nous nous sommes intéressé, dans un paragraphe précédent, aux mémoires mortes (ROM), nous avons vu que ces circuits combinatoires stockaient une relation entre entrées et sorties, mais pas de bits d'information. Tout était figé, et le passé du circuit n'apparaissait pas. L'intérêt principal des mémoires vives est qu'on peut y lire ou y écrire rapidement des informations. Leur principal inconvénient est qu'elles perdent leurs information si on cesse de les alimenter. Cependant, elles consomment très peu d'énergie et on peut les alimenter longtemps avec une simple pile.
Ces mémoires sont notamment utilisées dans les ordinateurs pour stocker temporairement programmes et données.
Leur structure de base est proche des mémoires mortes, sauf que cette fois, l'information est stockée dans des registres, que l'on peut lire, ou dans lesquels on peut écrire rapidement. Il s'agit cette fois vraiment de mémoire et non d'information figées stockées une fois pour toute comme dans les mémoires mortes.
Leur structure peut donc se présenter sous la forme suivante:



Il existe plusieurs sortes de RAM.
. Les mémoires vives statiques (SRAM) dans lesquelles l'information est stockée au moyen de registres formés de bascules. Elles sont très rapides, mais prennent de la place ce qui limite la capacité de stockage. On les trouves dans les appareils domestiques associées à des microcontrôleurs (pas besoin de stocker beaucoup de données), ou dans les oscilloscopes numériques ou les micro-ordinateurs (besoin de rapidité) notamment
. Les mémoires vives dynamiques (DRAM) dans lesquelles l'information est stockée sous forme de charges électriques dans des capacités réalisées à partir de transistors MOS (capacités de quelques pF). En raison de l'épuisement inévitable des charges, il est nécessaire de procéder périodiquement à un rafraîchissement, sous peine de perdre les données.
La cellule de base se compose de la façon suivante:



Lors de l'enregistrement des données, I1 et I2 sont fermés et les autres interrupteurs ouverts. Un 1 logique en entrée charge la capacité alors qu'un 0 logique la décharge. Une fois la donné rentrée, tous les interrupteurs sont ouverts pour déconnecter la capacité du circuit. Pour lire les données, on ferme I2, I3 et I4 alors que I1 reste ouvert. On récupère en sortie la comparaison de la tension de la capacité avec un niveau de référence qui donne 0 ou 5 V. On constate que la capacité est reliée à la sortie lors de la lecture ce qui fait que l'information est rafraîchie à chaque fois qu'elle est lue.
Ces mémoires sont plus complexes et moins rapides que des SRAM, mais elle consomment beaucoup moins d'énergie et permettent un stockage de plus forte capacité (la cellule de stockage occupe environ 4 fois moins de place que son équivalent statique). On les retrouve dans les micro-ordinateurs en raison de leur faible consommation et de leur grande capacité de stockage.

Familles TTL				
Description	Notation	Propagation (ns/porte)	Consommation (mW/porte)	Fanout
Standard	74*xxx*	10	10	10
Low-Power	74LSxxx	33	1	20
High Speed	74H*xxx*	6	22	10
Schottky	74S*xxx*	3	19	10
Low-Power Schottky	74LS*xxx*	9.5	2	20
Advanced Schottky	74AS*xxx*	1.7	8	40
Advanced LP Schottky	74ALS*xxx*	4	1.2	20
Fast	74F*xxx*	3	4	33
Familles CMOS				
Série A	40*xxx*A	25	10	50
Série B	40*xxx*B	60	10	50
Série UB	40*xxx*UB	30	7	50
C	74C*xxx*	50	5	50
HCT	74HCT*xxx*	8	10	50
FACT	74AC*xxx*	5	2	50
HC	74HC*xxx*	8	1	50
AHCT	74AHCT*xxx*	7	.12	50
HCTLS	74AHCTLS*xxx*	10	.12	50
Autres Familles				
RTL		12	16	5
DTL		30	12	5
MDTL		30	10	5
ECL 10K	101*xxx*, 105*xxx*	2	25	83
ECL 10K	102*xxx*, 106*xxx*	1.5	25	54
ECL 10K	108*xxx*	1-2.5	2.3	63
ECL 10KH	10 KH*xxx*	1	25	83
ECL 100K	1001*xxx*	.75	40	

Bibliographie

[1] Tocci Winder, Digital Systems principle and Applications, printice Hall,1998.
[2] siwww.epfl.ch/LSI2001/teaching/physiciens/lecon13/lecon13.html(logique)
[3] ftp://ftp.disccip.crd.p.acc-caen.fr/discip/crgelec/Cours/combinatoire.doc

[4]http://perso.wanadoo.fr/papanicola/sciences_indus/automatique/logique/3-%20Syst%E8messequentiels.pdf

[5]http://www.ac-nancy-metz.fr/pres-etab/lycom/electro/Electro-cours/multiplexa

[6] http://perso.wanadoo.fr/tsite/circuits.htme

[7]http://www.ac-nancy-metz.fr/pres-b/lycom/electro/Electro-cours/sequentielle.htm

[8]http://www.physiens-cachan.fr/pagregp/enseignement/elec/electronique/numerique.pdf(introduvction

[9] ftp://ftp.discip.crdp.ac-caen.fr/discip/elec/Cours/boole.doc(codag)

[10] p://ftp.discip.crp.ac-aen.fr/discip/crgelec/Cours/signaux.doc(mise en forme des signaux)

[11] http://perso.wandoo.fr/michel.hubin/physique/logique/chap_log1.ht(algebre de boole)

[12]http://www.eisi.hesbe.ch/dem/Mediatheque/microinformatique/Cours/LogiqueBooléenne.PDF(intro boole)

[13] http://perso.club-internet.fr/valetg/prof/eleon/logique/logique.htm(logique)

[14]http://www.ac-nancy-metz.fr/pretab/lycom/electro/Electro-cours/memoires.htm(memoire)