Les fonctions logiques

Le fonctionnement des ordinateurs tout comme d'autres appareils électroniques repose sur l'emploi des circuits électroniques de logique binaire ou électronique numérique. Dans cette branche de l'électronique les signaux électriques ne peuvent prendre que deux états 0 ou 1. A l'opposé nous avons l'électronique analogique beaucoup plus complexe à manipuler, et dans la quelles les signaux peuvent ont la possibilité de prendre une infinité de valeurs à l'image des grandeurs physiques habituelle comme la température, la pression, le débit, la vitesse, etc.

Une fois la différence entre l'électronique analogique et l'électronique digitale ou numérique faite, il y a lieu de distinguer dans le numérique deux concepts différents d'approche, selon que le temps qui s'écoule est pris en compte ou non.

Dans la logique combinatoire, une fonction sera exécutée chaque fois qu'un ensemble de conditions sera réuni, et maintenue tant que ces conditions demeurent. Nous pouvons citer le cas de l'interrupteur qui une fois actionné maintient la lampe allumée jusqu'à ce qu'elle soit actionnée à nouveau.

A l'opposé, la logique séquentielle fait intervenir le temps, ou, en d'autres termes, la mémorisation. Nous pouvons citer comme la lumière commandée par bouton poussoir ou télérupteur. Quand vous appuyer sur le bouton poussoir le télérupteur se met en mode travail et allume la lampe. Même si vous enlevez votre doigt, la lampe reste allumée. Cela veut dire qu'il y a mémorisation de l'action antérieure.

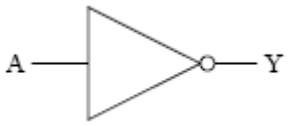
La logique élémentaire

Une des caractéristiques de l'électronique numérique de base est sa simplicité. Il n'existe en effet que trois circuits : le ET, le OU et le NON, les autres circuits étant des variantes de ces trois circuits.

L' Inverseur ou la porte NON

L'opération NON (NOT) a une seule entrée et une seule sortie. La sortie d'une fonction NON prend l'état 1 si et seulement si son entrée est dans l'état 0. La négation logique est symbolisée par un petit cercle.

Table de vérité, schéma et propriété de la fonction NON



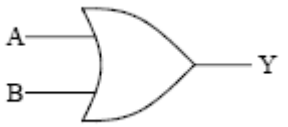
A	$Y = \overline{A}$
0	1
1	0

A''= A
A'+A=1
A'.A=0
A+(A'.B)= (A+A').(A+B) = A+B

La Porte OU (inclusif)

L'opération OU (OR), encore appelée addition logique (+) ou union, a au moins deux entrées. La sortie d'une fonction OU est dans l'état 1 si au moins une de ses entrées est dans l'état 1.

Table de vérité, schéma et propriété de la fonction OU



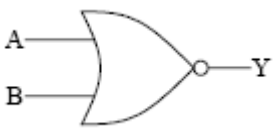
A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Il est facile de vérifier les propriétés suivantes de la fonction OU :

(A + B) + C = A + (B + C) = A + B + C	Associativité
A + B = B + A	Commutativité
A + A = A	Idempotence
A + 0 = A	Élément neutre
A + 1 = 1	

La porte NON OU ou NOR.

Une négation à la sortie d'une porte OU constitue une fonction NON OU (NOR : NOT OR)

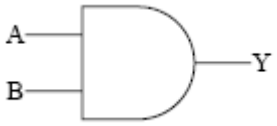


A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

La Porte ET

L'opération ET (AND), encore dénommée produit logique (.) ou intersection, a au moins deux entrées. La sortie d'une fonction AND est dans l'état 1 si et seulement si toutes ses entrées sont dans l'état 1.

Table de vérité, schéma et propriété de la fonction ET

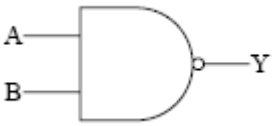


A	B	Y = A • B
0	0	0
0	1	0
1	0	0
1	1	1

(A • B) • C = A • (B • C) = A • B • C	Associativité
A • B = B • A	Commutativité
A • A = A	Idempotence
A • 1 = A	Élément neutre
A • 0 = 0	

La Portes NON ET ou NAND

Une porte NON ET (NAND : NOT AND) est constituée par un inverseur à la sortie d'une porte ET.

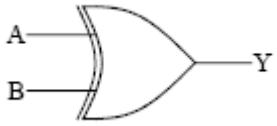


A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

Elle est vraie quand l'une des entrées est fausse

La porte XOR (Ou exclusif)

La porte ou exclusif est vraie quant les deux entrées sont de valeurs opposées.



A	B	Y = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

A xor B = (A+B).(A.B)' = A'B+AB' = (A+B).(A'+B')

Le théorème de Morgan

De Morgan a exprimé deux théorèmes qui peuvent se résumer sous la forme suivante :

$$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$$
$$\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$$

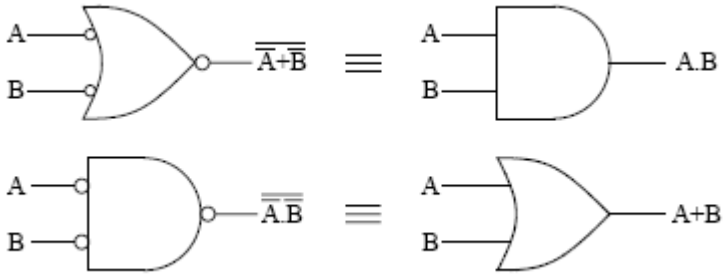
Pour vérifier le premier théorème nous remarquons que si toutes les entrées sont à 1 les deux membres de l'équation sont nuls. Par contre si une au moins des entrées est à 0 les deux membres de l'équation sont égaux à 1. Il y a donc égalité quels que soient les états des diverses entrées.

Le second théorème se vérifie de la même manière : si toutes les entrées sont à 0 les deux membres de l'équation sont à 1, par contre si au moins une des entrées est à 1 les deux expressions sont à 0.

Les théorèmes de De Morgan montrent qu'une fonction ET peut être fabriquée à partir des fonctions OU et NON. De même une fonction OU peut être obtenue à partir des fonctions ET et NON.

De même, à partir des théorèmes de De Morgan nous pouvons montrer qu'une porte ET en logique positive fonctionne comme une porte OU en logique négative et vice versa.

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



Le résumé des propriétés des portes logiques

OU	$(A + B) + C = A + (B + C) = A + B + C$ $A + B = B + A$ $A + A = A$ $A + 0 = A$ $A + 1 = 1$	Associativité Commutativité Idempotence Elément neutre
ET	$(A \bullet B) \bullet C = A \bullet (B \bullet C) = A \bullet B \bullet C$ $A \bullet B = B \bullet A$ $A \bullet A = A$ $A \bullet 1 = A$ $A \bullet 0 = 0$	Associativité Commutativité Idempotence Elément neutre
Distributivité	$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$ $A + (B \bullet C) = (A + B) \bullet (A + C)$	
NON	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A} \bullet A = 0$	
	$A + (A \bullet B) = A$ $A \bullet (A + B) = A$ $(A + B) \bullet (A + \overline{B}) = A$ $A + (\overline{A} \bullet B) = A + B$	
De Morgan	$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$ $\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$	
OU exclusif	$A \oplus B = (A + B) \bullet \overline{(A \bullet B)}$ $A \oplus B = (A \bullet \overline{B}) + (B \bullet \overline{A})$ $A \oplus B = \overline{(A \bullet B) + (\overline{A} \bullet \overline{B})}$ $A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$	

Écriture canonique d'une fonction logique

x	y	z	F
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

$$F = \bar{x}\,y\,z + x\,\bar{y}\,z + x\,y\,\bar{z} + x\,y\,z$$

Pour écrire la fonction canonique de cette table de vérité. Il faut considérer les valeurs vraies de la sortie F et faire la somme des produits.

Simplification de l'écriture des fonctions logiques

Simplifier une expression booléenne c'est lui trouver une forme plus condensée, faisant intervenir moins d'opérateurs et conduisant à une réalisation matérielle plus compacte. On peut simplifier une fonction par manipulation algébrique en utilisant une représentation graphique sous forme de tableaux à 2, 3, 4 et 5 variables : Les tableaux de Karnaugh

Simplification algébrique

Pour utiliser la méthode algébrique, il faut maitriser les identités remarquables sur les propriétés des fonctions logiques.

A.(B+C)=AB+AC Distributivité A+(BC) = (A+B).(A+C) Distributivité
A+AB = A A.(A+B) = A (A+B).(A+B') = A A+A'B = A+B

x	y	z	F
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

$$F = \bar{x}\,y\,z + x\,\bar{y}\,z + x\,y\,\bar{z} + x\,y\,z$$

$$\begin{aligned} F &= \bar{x}\,y\,z + x\,\bar{y}\,z + x\,y\,\bar{z} + x\,y\,z \\ &= (\bar{x}\,y\,z + x\,y\,z) + (x\,\bar{y}\,z + x\,y\,z) + (x\,y\,\bar{z} + x\,y\,z) \\ &= y\,z\,(\bar{x} + x) + x\,z\,(\bar{y} + y) + x\,y\,(\bar{z} + z) \\ &= x\,y + y\,z + z\,x \end{aligned}$$

Simplification par la méthode de Karnaugh

La méthode de Karnaugh est basée sur l'inspection visuelle de tableaux disposés de façon à ce que deux cases adjacentes en ligne et en colonne ne différent que par l'état d'une variable et d'une seule.

Si une fonction dépend de n variables, il ya 2^n possibilités de produits. Chacun de ces produits est représentées par une case dans un tableau. Il faut bien observer comment les tableaux sont numérotés. D'une case à sa voisine, une seule variable change d'état.

Partir de l'équation AB + AB' = A (B+B') = A

- Remplir les tables de vérité
- Faire l'équation canonique
- Dessiner le tableau de Karnaugh
- Remplir les cases du tableau de Karnaugh
- Regrouper les cellules adjacentes
- Simplifier les regroupements
- Retenir l'équation finale

Les combinaisons sont placées dans l'ordre du codage binaire réfléchi ou du codage Gray afin que les produits adjacents se trouvent dans des cases voisines ou dans les cases extrêmes.
On peut alors faire des regroupements de 2, 4, 8, etc.

Pour 5 variables, deux représentations sont possibles. Le tableau de Karnaugh peut être traités comme deux tableaux 4x4 superposés ou un seul tableau de 4x8. Observez comment sont numérotées les lignes et les colonnes : d'une case à sa voisine une seule variable change d'état.

Tableau à deux variables (n=2 , 2^n cases = 4 cases)

AB	A'	A
B'		
B		

AB	0	1
0	*x*	**x**
1		*x*

Chaque case a deux voisins adjacents

Tableau à trois variables (n=3, 2^n cases = 8 cases)

ABC	A' B'	A' B	A B	A B'
C'				
C				

ABC	00	01	11	10
0	*x*		*x*	**x**
1				*x*

Chaque case a trois voisins adjacents

Tableau à quatre variables (n=4, 2^n cases = 16 cases). 2 types de représentations

ABCD		A' B'	A' B	A B	A B'
C'	D'				
C'	D				
C	D				
C	D'				

ABCD	00	01	11	10
00	*x*		*x*	**x**
01				*x*
11				
10				*x*

Chaque case a 4 voisins adjacents

ABCD	A' B' C'	A' B' C	A' B C	A' B C'	A B C'	A B C	A B' C	A B' C'
D'								
D								

ABCD	000	001	011	010	110	111	101	100
0								
1								

Tableau à cinq variables (n=5, 2^n cases = 32 cases). 2 types de représentations

ABCDE		A' B' C'	A' B' C	A' B C	A' B C'	A B C'	A B C	A B' C	A B' C'
D'	E'								
D'	E								
D	E								
D	E'								

ABCDE	000	001	011	010	110	111	101	100
00							x	
01		x				x	x	x
11							x	
10								

Chaque case à 5 voisins. Pour les identifier, il suffit de replier le tableau par rapport à la ligne médiane qui se trouve entre le 010 et le 110

E'

ABCD		A' B'	A' B	A B	A B'
C'	D'	*x*		*x*	x
C'	D				*x*
C	D				
C	D'				*x*

E

ABCD		A' B'	A' B	A B	A B'
C'	D'				
C'	D				
C	D				
C	D'				

0

ABCD	00	01	11	10
00				
01				
11				
10				

1

ABCD	00	01	11	10
00				
01				
11				
10				

Le passage de la table de vérité au tableau de Karnaugh consiste à remplir chaque case avec leur valeur de la fonction pour le produit correspondant. Pour ne pas encombrer le tableau, il ne faut mettre que les 1.

La simplification consiste à rassembler les cases adjacentes contenant des 1 par groupe de 2, 4 et 8, ensuite les factoriser.

La logique combinatoire

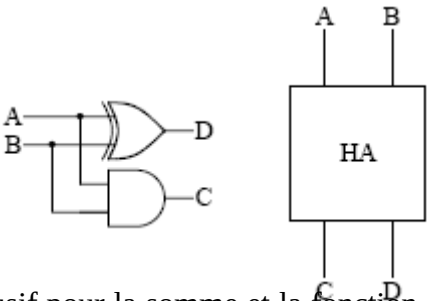
Le demi- additionneur

Pour additionner deux nombres, il faut d'abord additionner les 2 bits de poids faible a_0 et b_0, puis additionner les bits suivants sans oublier les retenues.
Un demi-additionneur ou Half Addr (HA) est un circuit qui additionne les 2 bits de poids faibles pour lesquelles la retenue propagée n'est pas prise en compte.

Table de vérité, forme canonique et schéma

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

$$\begin{cases} D = \overline{A}\,B + A\,\overline{B} \\ C = A\,B \end{cases} \qquad \begin{cases} D = A \oplus B \\ C = A\,B \end{cases}$$



Dans cette table de vérité, nous reconnaissons la fonction OU exclusif pour la somme et la fonction ET pour la retenue.

Additionneur complet d'1 bit

Pour additionner les bits de poids supérieur a_i, b_i $(0 < i < n)$ et de tenir compte de la retenue r_{i-1} propagée depuis le rand i-1, il faut recours à un Full Adder ou additionneur complet. Un Full Adder peut additionner 2 bits avec un report, soit trois entrées et deux sorties.

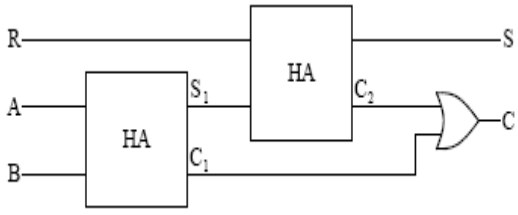
Table de vérité, forme canonique et schéma

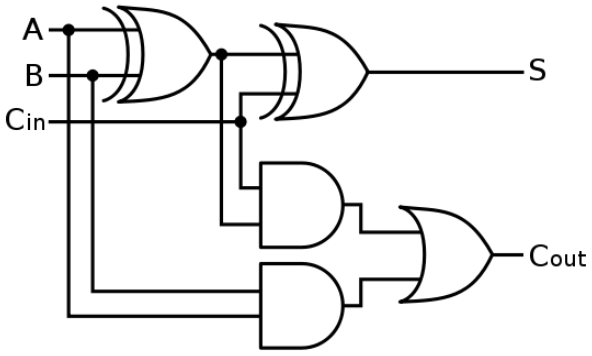
A	B	R	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S=A'B'R+A'BR'+AB'R'+ABR = R'(A xor B) + R (AB+A'B')
R'(A xor B) + R (AB+A'B') = R' (A xor B) + R (A xor B)' = R xor A xor B

C= R(A'B+AB')+ AB(R'+R) = R (A xor B)+ AB

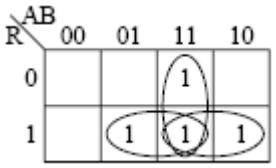
R= Cin (Carriage in) et C= Cout (Carriage out)





Simplification en utilisant la représentation de Karnaugh

$$\begin{cases} S = \overline{A}\,\overline{B}R + \overline{A}B\overline{R} + A\overline{B}\,\overline{R} + ABR \\ C = \overline{A}BR + A\overline{B}R + AB\overline{R} + ABR \end{cases}$$

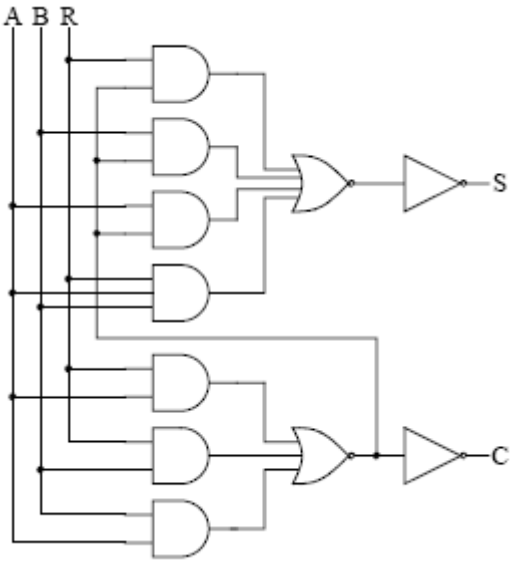


$$\begin{cases} A\overline{C} = A\overline{B}\,\overline{R} \\ B\overline{C} = \overline{A}B\overline{R} \\ R\overline{C} = \overline{A}\,\overline{B}R \end{cases} \Rightarrow (A+B+R)\overline{C} = A\overline{B}\,\overline{R} + \overline{A}B\overline{R} + \overline{A}\,\overline{B}R$$

$$C = AB + AR + BR$$

$$\overline{C} = \overline{A}\,\overline{B} + \overline{A}\,\overline{R} + \overline{B}\,\overline{R}$$

$$S = (A+B+R)\overline{C} + ABR$$



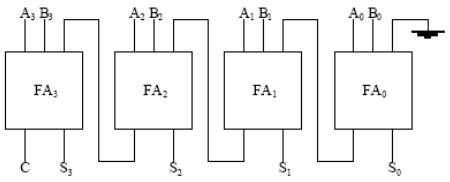
Additionneur complet de n bits ou Additionneur en parallèle (Binary Parallel Adder)

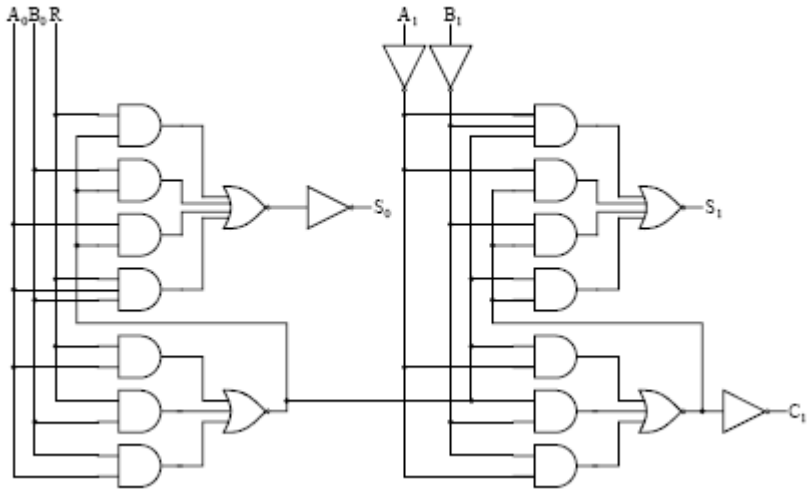
L'addition de nombres comptant plusieurs bits peut se faire en série (bit après bit) ou en parallèle (tous les bits simultanément).

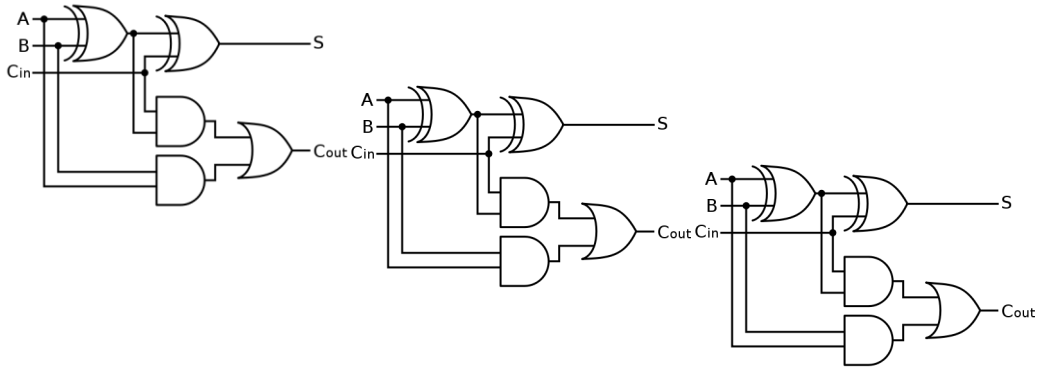
Nous allons prendre l'exemple d'un additionneur 4 bits comptant quatre "Full Adders", montés en parallèle ou en cascade. Chaque additionneur FA_i est affecté à l'addition des bits de poids i. L'entrée correspondant au report de retenue pour FA_0 est imposée à 0 (en logique positive). La retenue finale C indique un dépassement de capacité si elle est égale à 1.

Pour réaliser un additionneur complet de 2 nombres on met en cascade plusieurs FA de telle sorte que la retenu de sortie des bits moins significatifs (LSB) constituent le retenus d'entrée des bits plus significatifs (MSB).

	1	1	1	1		A3	A2	A1	A0
	0	1	1	1		B3	B2	B1	B0
1	0	1	1	0					







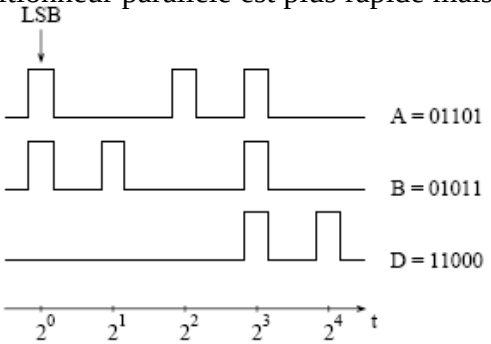
Si le dernier bit de retenue est égal à 1, il est signalé par un indicateur de carry mémorisé par un 1 noté C (Carry) dans un registre appelé le Processor Status Word (PSW) ou registre d'etat.

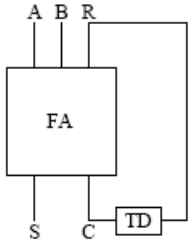
Si le résultat est aussi en dehors de l'intervalle des n bits, ce dépassement de capacité est aussi mémorisé dans le registre d'état du PSW par l'intermédiaire d'un bit 1 noté O (Overflow)

Addition séquentielle

Dans un additionneur séquentiel chacun des nombres A et B est représenté par un train d'impulsions synchrones par rapport à un signal d'horloge. L'ordre chronologique d'arrivée des impulsions correspond à l'ordre croissant des poids : le bit le moins significatif se présentant le premier. Ces impulsions sont injectées sur les deux lignes d'entrée d'un additionneur.
A chaque cycle d'horloge, la retenue provenant des bits de poids inférieurs doit être mémorisée (par exemple, à l'aide d'une bascule D qui sera étudiée dans le chapitre suivant).

Un additionneur parallèle est plus rapide mais nécessite plus de composants.





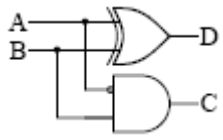
Le demi- soustracteur

Un demi-additionneur ou Half Substractor (HS) est un circuit qui peut faire la soustraction de 2 bits.

Table de vérité, forme canonique et schéma

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

$$\begin{cases} D = \overline{A}B + A\overline{B} = A \oplus B \\ C = \overline{A}B \end{cases}$$

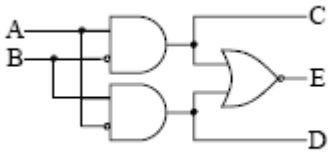


Le Comparateur

On rencontre très souvent la nécessité de comparer deux entiers (A = B, A > B ou A < B). Ecrivons la table de vérité correspondant à ces trois fonctions de comparaison de 2 bits. La fonction C doit être égale à 1 si et seulement si A > B, la fonction D si et seulement si A < B et la fonction E si et seulement si A = B. Ce qui nous donne :

A	B	C(A>B)	D(A<B)	E(A=B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$$\begin{cases} C = A\overline{B} \\ D = \overline{A}B \\ E = \overline{A \oplus B} = \overline{A\overline{B} + \overline{A}B} = \overline{C + D} \end{cases}$$



Circuit permetteant de comparer deux n

Le Contrôle de parité

La parité d'un mot binaire est définie comme la parité de la somme des bits, soit encore :
- parité paire (ou 0) : nombre pair de 1 dans le mot;
- parité impaire (ou 1) : nombre impair de 1 dans le mot.

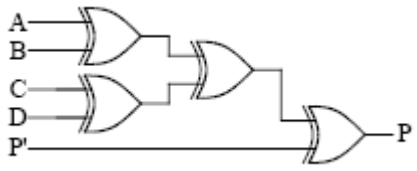
La fonction OU-exclusif donne la parité d'un sous-ensemble de deux bits. Le contrôle de parité est basé sur la constatation que le mot de n+1 bits formé en adjoignant à un mot de n bits son bit de parité est toujours de parité 0. La figure 15 représente le diagramme logique d'un générateur43 contrôleur de parité pour 4 bits. Si l'entrée P' est imposée à 0 ce circuit fonctionne comme générateur de parité : la sortie P représente la parité du mot composé par les bits A, B, C et D.

A	B	C	D	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Pas ce case adjacent, donc non simplifiable par Karnaugh.

P=A’B’C’D’+ A’B’CD+ A’BC’D+ A’BCD’+ AB’C’D+ AB’C’D+ ABC’D’+ ABCD

(A xor B) xor (C xor D) xor P’ = P



Un codeur

Un codeur permet de transmettre une information de la meilleure façon. Cette transmission est souvent plus simple, plus rapide et plus sécurisé. Les langues que nous utilisons sont des codes.
Le codeur fait correspondre à l'activation d'une entrée particulière une combinaison de bits en sortie. C'est un circuit à une entrée et n sorties.

Table de vérité d'un codeur BCD (Binary Code Decimal)

Entrée	a	b	c	d
E0	0	0	0	0
E1	0	0	0	1
E2	0	0	1	0
E3	0	0	1	1
E4	0	1	0	0
E5	0	1	0	1
E6	0	1	1	0
E7	0	1	1	1
E8	1	0	0	0
E9	1	0	0	1

0
1
2
3
4
5
6
7
8
9

a = E8 + E9 **b = E4+ E5+ E6 + E7** **c=E2+E3+E6+E7** **d=E1+E3+E5+E7+E9**

Le décodeur

Le décodeur réalise la fonction inverse du codeur. C'est un circuit logique comportant n entrée et 2 n sorties. Lorsque le signal de validation est actif, seule la sortie dont le numéro correspond à la valeur binaire affichée sur l'entrée est active. Toutes les autres sont inactives.

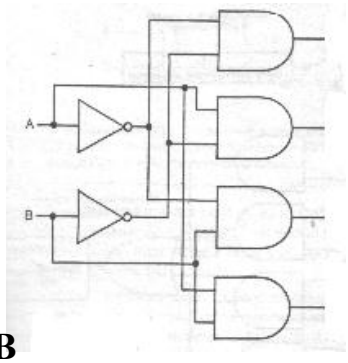
Nous pouvons citer le clavier d'une porte automatique ou le décodeur BCD/Afficheur 7 segments.

Le décodeur est un circuit très employé dans les microprocesseurs. Son rôle est de sélectionner entre autres, une adresse précise de mémoire parmi un lot important d'adresses différentes.

Supposez qu'il faille aller chercher dans une mémoire d'une capacité de 4096 mots, un mot donné situé à une adresse bien précise. Il n'est pas question d'adresser les 4096 mots différents contenus dans la mémoire. Cela nécessiterait 4096 fils. Pour résoudre ce problème il ne faut relier le microprocesseur que par 12 fils, car c'est un mot de 12 bit qui permet d'adresser 4096 place ($2^{12} = 4096$). Il nous faudra alors un décodeurs 12 entrées et 4096 sorties.

Décodeur 2 entrée et 4 sorties

A	B	S0	S1	S2	S3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



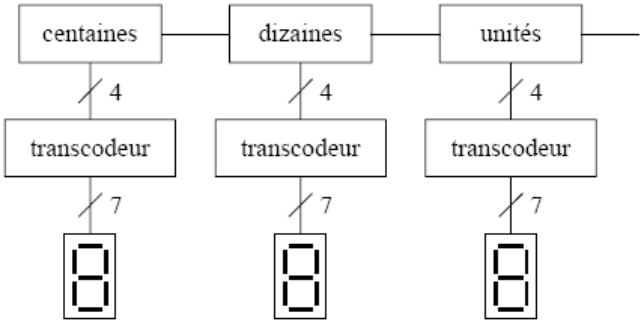
S0 = A'B' S1=A'B S2 = AB' S3=AB

Dans un système numérique les instructions, tout comme les nombres, sont transportées sous forme de mots binaires. Par exemple un mot de 4 bits peut permettre d'identifier 16 instructions différentes : l'information est codée. Très souvent l'équivalent d'un commutateur à 16 positions permet de sélectionner l'instruction correspondant à un code. Ce processus est appelé décodage. La fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes une seule sortie active parmi les $N = 2^n$ sorties possibles. A titre d'exemple, nous allons étudier le décodage de la représentation DCB des nombres

Représentation DCB (Décimale Codée Binaire)

Le code DCB (ou en anglais BCD : Binary Coded Decimal) transforme les nombres décimaux en remplaçant chacun des chiffres décimaux par 4 chiffres binaires. Cette représentation conserve donc la structure décimale : unités, dizaines, centaines, milliers, etc… Chaque chiffre est codé sur 4 bits.

Décimal	DCB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001



Par exemple le nombre décimal 294 sera codé en DCB : 0010 1001 0100. Ce type de codage permet, par exemple, de faciliter l'affichage en décimal du contenu d'un compteur. Pour ce faire on peut utiliser des tubes de Nixie, contenant 10 cathodes ayant chacune la forme d'un chiffre ou des afficheurs lumineux à sept segment

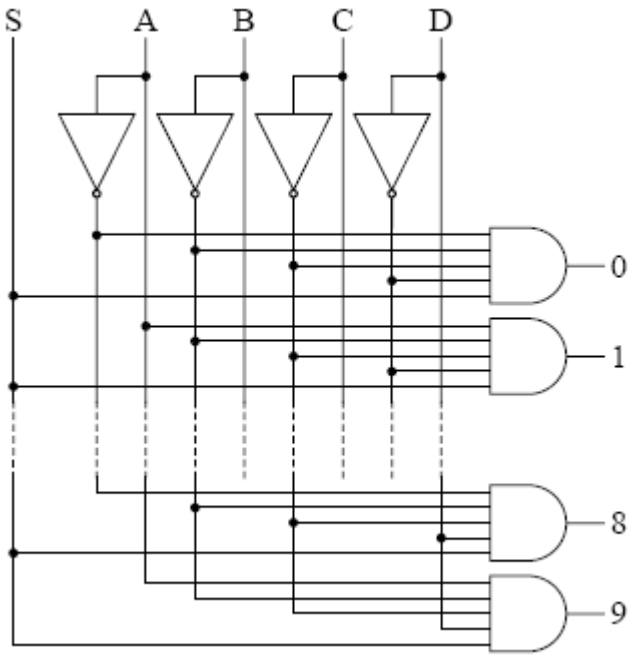
La fonction de chacun des transcodeurs est de positionner à 1 les lignes de sortie correspondant aux segments à allumer selon de code porté par les quatre lignes d'entrée. De manière générale, un transcodeur fait correspondre à un code A en entrée sur n lignes, un code B en sortie sur m lignes.

Nous allons étudier l'exemple d'un décodeur DCB-décimal. La table de vérité de ce décodeur est très simple :

D	C	B	A	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9
0	0	0	0	1									
0	0	0	1		1								
0	0	1	0			1							
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									1	
1	0	0	1										1

A chacune des lignes de sortie nous pouvons associer un produit prenant en compte chacune des quatre entrées ou leur complément. Ainsi la ligne 5 correspond à : A’ B C’D

D'autre part, on souhaite souvent n'activer les lignes de sortie qu'en présence d'un signal de commande global (strobe ou enable). Ce signal S est mis en coïncidence sur chacune des dix portes de sortie. Dans l'exemple suivant, si S est dans l'état 0 le décodeur est bloqué et tous les sorties sont également dans l'état 0.



Le transcodeur

Le transcodeur est un circuit qui permet de passer d'un code à un autre. Il comporte n entrée et n sorties. Le lecteur de code barre est un transcodeur.

Table de vérité pour passer du code binaire au code Gray

Le code Gray est fréquemment utilisé dans les capteurs angulaires ou de positionnement, mais aussi lorsque l'on désire une progression numérique binaire sans parasite transitoire.
Le code Gray sert également dans les tableaux de Karnaugh utilisés lors de la conception de circuits logiques.

Pour passer du code binaire au code Gray on utilise la formule suivante :

$$n = \frac{N \oplus 2N}{2}$$

n est le nombre convertit en Gray et N le nombre binaire.

A0	A1	A2		B0	B1	B2
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	1
0	1	1		0	1	0
1	0	0		1	1	0
1	0	1		1	1	1
1	1	0		1	0	1
1	1	1		1	0	0

B0=A0A1'A2'+A0A1'A2+A0A1A2'+A0A1A2
B1=A0'A1A2'+ A0'A1A2+A0A1'A2'+A0A1'A2
B2=A0'A1'A2+ A0'A1A2'+A0A1'A2+A0A1A2'

Un encodeur

Un encodeur est système qui comporte N lignes d'entré et n lignes de sorties. Lorsqu'une des lignes d'entré est activité l'encodeur fournit en sortie un mot de n bit correspondant au codage de l'information identifié par la ligne activée.

Table de vérité d'un encodeur transformant un nombre décimal en son équivalent en code BCD. Il comporte 10 entrées et 4 sorties

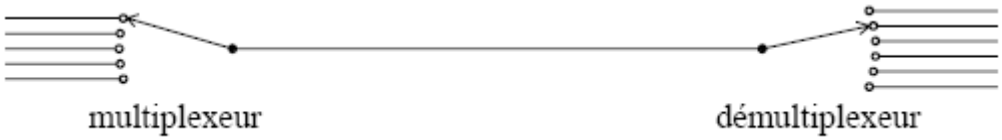
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9		S3	S2	S1	S0
1	0	0	0	0	0	0	0	0	0		0	0	0	0
0	1	0	0	0	0	0	0	0	0		0	0	0	1
0	0	1	0	0	0	0	0	0	0		0	0	1	0
0	0	0	1	0	0	0	0	0	0		0	0	1	1
0	0	0	0	1	0	0	0	0	0		0	1	0	0
0	0	0	0	0	1	0	0	0	0		0	1	0	1
0	0	0	0	0	0	1	0	0	0		0	1	1	0
0	0	0	0	0	0	0	1	0	0		0	1	1	1
0	0	0	0	0	0	0	0	1	0		1	0	0	0
0	0	0	0	0	0	0	0	0	1		1	0	0	1

S0 = E1+E3+E5+E7+E9
S1 = E2+E3+E6+E7
S2 = E4+E5+E6+E7
S3 = E8+E9

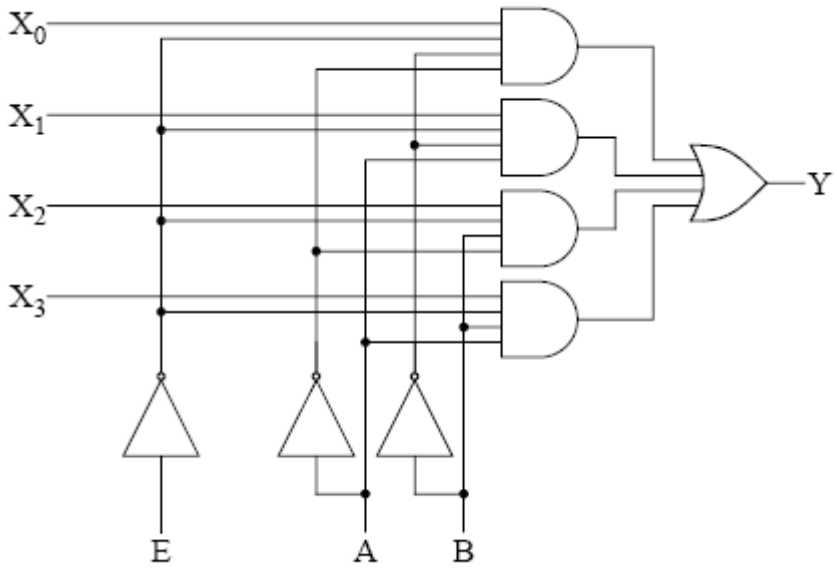
Le Multiplexeur

Le multiplexeur aussi très employé à pour rôle d'aiguiller sur une sortie unique un signal prélevé parmi plusieurs autres de nature différentes. C'est en quelque sorte un circuit qui réalise la fonction inverse d'un décodeur.
Le multiplexeur est un dispositif qui permet de transmettre sur une seule ligne des informations en provenance de plusieurs sources ou à destination de plusieurs cibles.

C'est un circuit comportant 2^n entrées d'informations, n entrées d'adresse et une seule sortie. Lorsque la validation est active la sortie prend l'état d'une entrée.



E Enable	B Commande	A Commande	E Entrée	S Sortie
0	0	0	X0	E'B'A'X0
0	0	1	X1	E'B'AX1
0	1	0	X2	E'BA'X2
0	1	1	X3	E'BAX3
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	0	

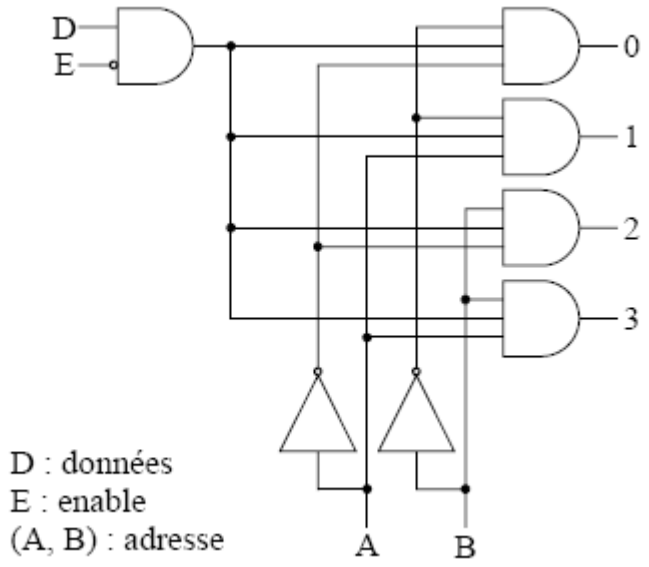


Démultiplexeur

C'est le circuit complémentaire du multiplexeur. Il comporte une entrée d'information, 2^n sorties d'information et n entrées d'adresse. Il met en relation cette entrée avec une sortie et une seule.

Pour pouvoir sélectionner cette sortie il faut également des lignes d'adressage : le code porté par ces lignes identifie la ligne de sortie à utiliser. Ce circuit est très proche d'un décodeur. Considérons un démultiplexeur avec quatre lignes de sortie. Il faut deux lignes d'adresse. Supposons que nous souhaitons également valider les données avec un signal de contrôle E (pour laisser par exemple le temps aux niveaux d'entrée de se stabiliser). Par convention nous choisissons de prendre en compte les données pour E = 0.

E	B	A	Y_0	Y_1	Y_2	Y_3	Produit
0	0	0	D	0	0	0	$\overline{A}\,\overline{B}\,\overline{E}\,D$
0	0	1	0	D	0	0	$A\,\overline{B}\,\overline{E}\,D$
0	1	0	0	0	D	0	$\overline{A}\,B\,\overline{E}\,D$
0	1	1	0	0	0	D	$A\,B\,\overline{E}\,D$
1	0	0	0	0	0	0	
1	0	1	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	0	0	



Comme pour l'additionneur, il est possible de faire des cascades de multiplexeur et de démultiplexeur pour avoir plusieurs entrées ou plusieurs sorties.

La logique séquentielle

Un système combinatoire est tel que l'état de ses sorties ne dépend que l'état des entrées. Il peut être donc représenté par une table de vérité ou un tableau de Karnaugh, pour chaque sortie. Il est possible donc d'écrire l'équation logique de chaque sortie, en fonction seulement des entreés.

Quant au système séquentiel, il dépend non seulement des entrées, mais aussi de l'état précédant du système. On retrouvera les mêmes états des entrés à plusieurs étapes, alors que els sorties seront différents. Il est donc impossible de représenter un tableau de Karnaugh.

Pour les circuits de logique séquentielle nous devons tenir compte de l'état du système. Ainsi les sorties dépendent des entrées mais également de l'état du système. Celui-ci dépend aussi des entrées. Si nous notons Q l'état d'un système séquentiel, X ses entrées et Y ses sorties, nous avons de manière générale :

$$\begin{cases} Q = f(X, Q) \\ Y = g(X, Q) \end{cases}$$

La logique séquentielle permet de réaliser des circuits dont le comportement est variable avec le temps. L'état d'un système constitue une mémoire du passé.

Lorsque les changements d'état des divers composants d'un circuit séquentiel se produisent à des instants qui dépendent des temps de réponse des autres composants et des temps de propagation des signaux on parle de logique séquentielle asynchrone. Cependant les retards peuvent ne pas être identiques pour toutes les variables binaires et conduire à certains aléas.

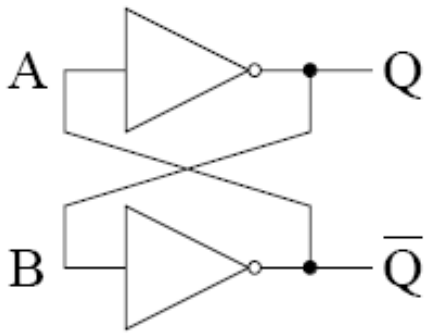
Ceux-ci peuvent être évités en synchronisant la séquence des diverses opérations logiques sur les signaux périodiques provenant d'une horloge. La logique séquentielle est alors dite synchrone : tous les changements d'état sont synchronisés sur un signal de contrôle.

Les éléments de base sont les bascules et une association de bascules nous permettront de construire des registres, des compteurs, de décompteurs et des diviseurs.

Nous allons étudier successivement la la bascule RS, RST, D, JK

Un latch ou verrou. Le cas de la bascule bistable

Une bascule (flip-flop) a pour rôle de mémoriser une information élémentaire. C'est une mémoire à 1 bit. Une bascule possède deux sorties complémentaires Q et Q. La mémorisation fait appel à un verrou (latch) ou système de blocage, dont le principe de rétro-action peut être représenté de la façon suivante :



Si (Q = 0)	(B= 0)	(Q= 1)	(A= 1)	(Q= 0)
Si (Q = 1)	(B= 1)	(Q= 0)	(A= 0)	(Q =1)

Une bascule ne peut donc être que dans deux états : "1" (Q = 1, Q = 0) et "0" (Q = 0, Q = 1) . Les interconnexions du verrou interdisent les deux autres combinaisons : Q = Q = 1 ou Q = Q = 0 .
Ce type de circuit, qui n'a que deux états stables possibles, est encore appelé circuit bistable.

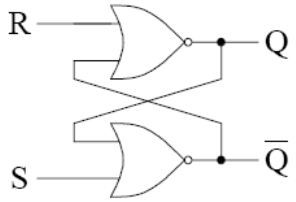
Un verrou permet de conserver un état, il nous faut maintenant savoir comment changer cet état.

Cette bascule a cependant deux inconvénients majeurs. Il est impossible de dire dans quel état il va se trouver à la mise sous tension. Ensuite si l'état se stabilise, il est impossible d'en sortir. S'il était possible de le piloter on aura réalisé une mémoire d'un bit.
Ce pilotage est possible en replaçant l'inverseur par un circuit logiquement équivalent, mais avec deux entrée (R et S) en utilisant les porte NOR ou NAND.

La bascule RS

La bistable RS est la première évolution du circuit mémoire à base d'inverseurs. Il a deux entrées de commande qui permettent de mettre le bistable dans l'un ou l'autre état : Set (Remise à Un) et Reset (Remise à Zéro). La valeur météorisée est accessible sur la sortie Q.

Les verrous les plus fréquemment rencontrés sont réalisés avec deux portes NOR ou NAND.
Considérons dans un premier temps le circuit suivant :



S	R	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Sorties inchangées
1	0	1	0	Set : Remise à Un : RAU
0	1	0	1	Reset : Remise à Zéro : RAZ
1	1	0	0	A proscrire

$$Q+ = R'S+(RS)'.Q$$

Si on applique S = 1 et R = 0 ou S = 0 et R = 1 on impose l'état de la sortie Q respectivement à 1 ou à 0, la sortie Q prenant la valeur complémentaire.

(S=1, R=0)	donne	(Q'=0, Q=1)	Set = Remise à Un RAU
(S=0, R=1)	donne	(Q'=1, Q=0)	Reset = Remise à Zéro : RAZ

Cet état se maintient lorsque les deux entrées retournent à 0.

(Q'=0, Q=1)	(S=0, R=0)	(Q'=0, Q=1)	Q reste Q et Q' reste Q'
(Q'=1, Q=0)	(S=0, R=0)	(Q'=1, Q=0)	Q reste Q et Q' reste Q'

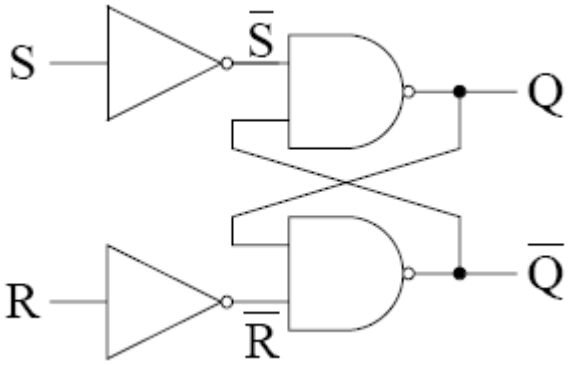
La configuration S = R = 1 est à proscrire car ici elle conduit à Q = Q' = 0, ce qui est inconsistant logiquement avec notre définition.

(Q'=0, Q=1)	(S=1, R=1)	(Q'=0, Q=0)	A proscrire

Mais surtout, lorsque R et S reviennent à 0, l'état Q= Q' étant incompatible avec les interconnexions, l'une de ces deux sorties va reprendre l'état 1, mais il est impossible de prédire laquelle : la configuration S = R = 1 conduit à une indétermination de l'état des sorties et est donc inutilisable.

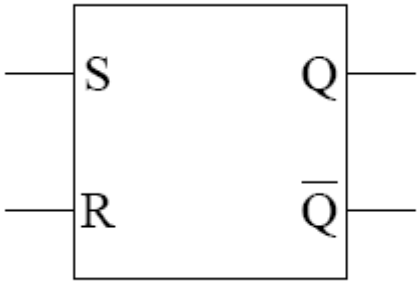
Le circuit est dit asynchrone car il n'est piloté par aucun signal d'horloge.

La bascule RS NAND



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

L'utilisation des deux inverseurs sur les lignes d'entrée nous permet de retrouver une table de vérité comparable à celle de la bascule RS précédente.



Cette bascule est toute fois un peu trop élémentaire, car elle souffre de deux défauts graves. Le premier défaut est l'impossibilité d'assurer la complémentarité de la sortie Q et Q' lorsque que les deux entrées sont à 1. Le second réside dans le fait que cette bascule est à tout instant sous la commande des entrées R et S. Si une impulsion parasite survient le désordre s'installe.

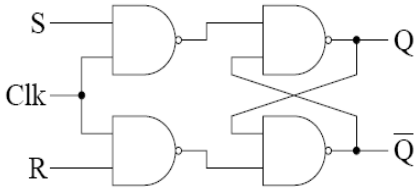
La RS à entrée en horloge (RST)

On a souvent besoin d'effecteur des changements d'état à des instants déterminés, rythmés par une horloge, signal carré prenant régulièrement les valeurs 0 et 1. On peut facilement modifier la bascule RS pour la rendre synchrone, c'est à dire de ne permettre de changement que lorsque le signal d'horloge vaut 1.

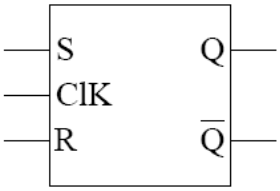
La bascule R.S.T. est une bascule pour laquelle les entrées S et R ne sont prises en compte qu'en coïncidence avec un signal de commande. Ce signal peut être fourni par une horloge, nous avons alors une bascule synchrone.

Cette bascule n'a pas le dernier inconvénient cité précédemment, c'est à dire la possibilité de changer d'état à tout moment.

Une entrée d'horloge constituée par deux porte ET assure le passage ou le blocage des ordres de basculement selon le niveau du signal d'horloge.



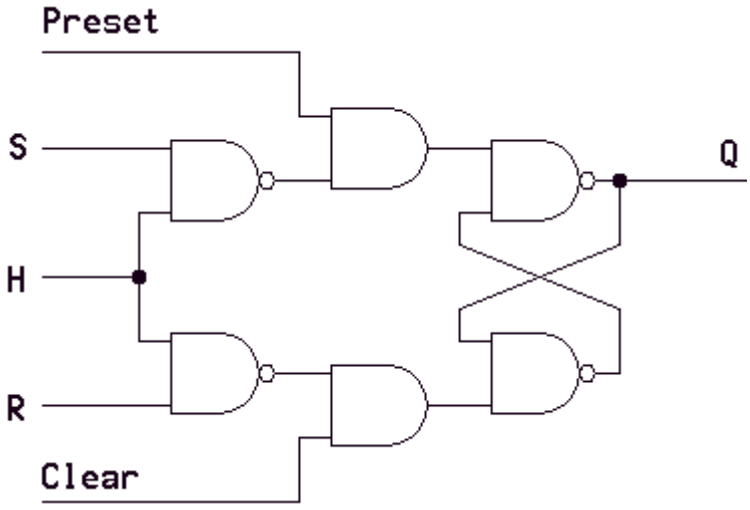
T-1		T	
S	R	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	0	1	0
1	1	?	?



Lorsque le signal de commande, noté ici Clk, est à 1 la bascule fonctionne comme indiqué précédemment et les sorties suivent les variations des entrées S et R. Par contre, lorsque le signal de commande est à 0, la bascule est bloquée : Q est indépendant des éventuels changements de S et R. L'état mémorisé correspond au dernier état avant le passage de la ligne de commande de 1 à 0.

De telles bascules sont souvent utilisées dans les circuits des microprocesseurs. Elles sont fréquemment regroupées par deux ou par quatre dans un même circuit intégré, avec une ligne commune pour l'horloge.

Bascule RST avec Preset et Clear

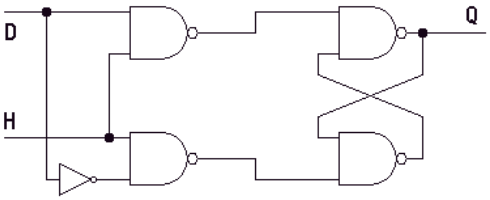


Il est également possible d'ajouter à la bascule RST des entrées permettant de forcer l'état de la bascule, et ceci même en l'absence de signal **H**

La bascule D

Pour les deux premières bascules il est impossible de maintenir Q et Q barre complémentaire si R et S sont tous égales à 1. Une façon de résoudre ce problème consiste à rendre complémentaire en permanence l'entrée R et S. Il ne subsiste alors qu'une seule entrée dénommée D (Data) répartie de façon complémentaire sur les deux portes ET et à l'aide d'un inverseur. S=1, R=0 si D=1 S=0 et R= 1 sir D=0.
Cette bascule permet d'éviter S=R=1 et S=R=0. Avec l'inverseur S et R sont toujours complémentaires.

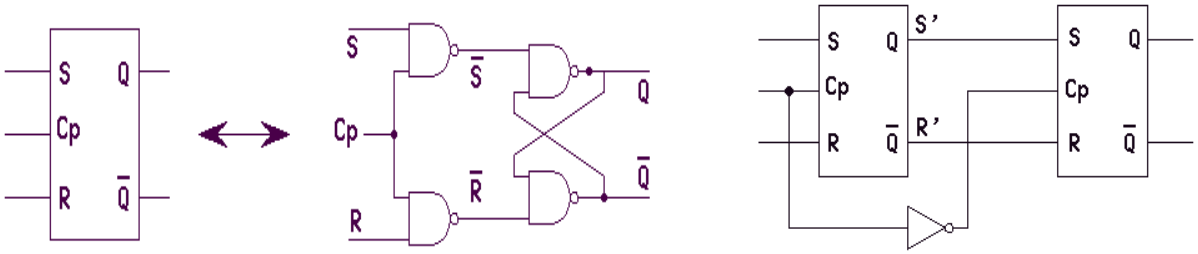
Ce type de bascule est très utilisé dans les mémoires intermédiaires situées à l'entrée des divers organes des microprocesseurs, là où l'information doit être recopié systématiquement avant qu'elle ne pénètre dans l'organe à qui elle est destinée.



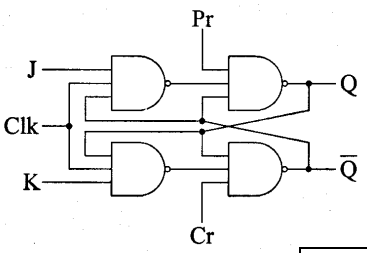
T-1	T
D	Q
0	0
1	1

Si H=0 quelque soit la valeur de D, Q et Q' restent inchangés.
Si H=1, D=1 Q=1 et Q'=0
Si H=1, D=0 Q=0 et Q'=1

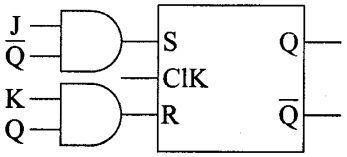
La bascule JK



L'introduction d'une entrée d'horloge sur une bascule n'a toutefois pas complètement supprimé le phénomène de déclenchement intempestif sur les signaux parasites d'entrée.
La bascule JK est constituée de deux bascules RST. Dont l'un est appelé maître et l'autre esclave.



$$\begin{cases} S = J\,\bar{Q} \\ R = K\,Q \end{cases}$$



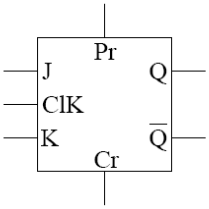
T-1		T
Jn	**Kn**	**Qn+1**
0	**0**	**Qn**
0	**1**	**0**
1	**0**	**1**
1	**1**	**Q'n**

J_n	K_n	Q_n	$\bar{Q}_n$	S	R	Q_{n+1}
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

Preset et Clear

Les entrées asynchrones (car à utiliser en absence de signal d'horloge) Pr (Preset) et Cr (Clear) permettent d'assigner l'état initial de la bascule, par exemple à la mise sous tension pour éviter tout aléa. En fonctionnement normal ces deux entrées doivent être maintenues à 1. Lorsque le signal d'horloge est à 0 nous avons la table de vérité suivante :

Pr	Cr	Q
1	1	Q
0	1	1
1	0	0

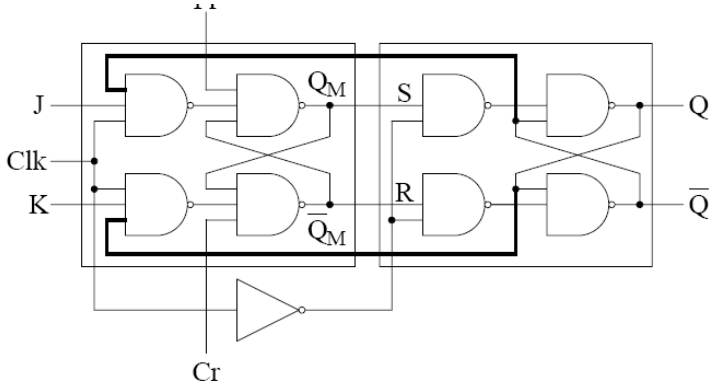


Bascule J-K Maître-Esclave

Jusqu'à présent nous avons construit les tables de vérité à partir de la logique combinatoire qui suppose que les entrées sont indépendantes des sorties. Or dans la bascule J-K nous avons introduit des connexions d'asservissement entre les entrées et les sorties. Ainsi supposons qu'avant le signal d'horloge nous avons J = K = 1 et Q = 0 . Lorsque le signal d'horloge passe à 1 la sortie Q devient 1. Ce changement intervient après un intervalle de temps ⊗t. Nous avons alors J = K = Q = 1. Nous voyons que la sortie Q doit alors revenir à 0. Ainsi la sortie Q va osciller entre 0 et 1 pendant toute la durée du signal d'horloge rendant le résultat ambigu.

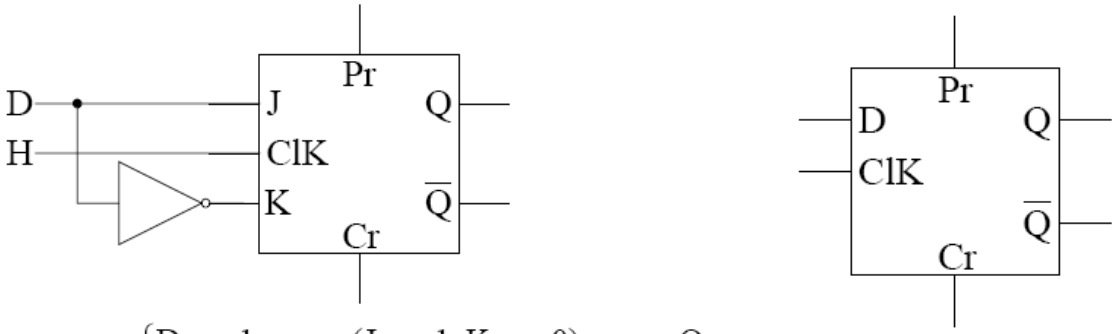
Pour éviter ce problème on monte deux bascules R-S en cascade en asservissant (traits épais) les entrées de la première (Maître) aux sorties de la seconde (Esclave). D'autre part, le signal d'horloge parvenant à l'esclave est inversé.

$$\begin{cases} (Q_M = 1, \overline{Q}_M = 0) \Rightarrow (S = 1, R = 0) \Rightarrow (Q_{n+1} = 1, \overline{Q}_{n+1} = 0) \\ (Q_M = 0, \overline{Q}_M = 1) \Rightarrow (S = 0, R = 1) \Rightarrow (Q_{n+1} = 0, \overline{Q}_{n+1} = 1) \end{cases}$$



Bascule JK – D

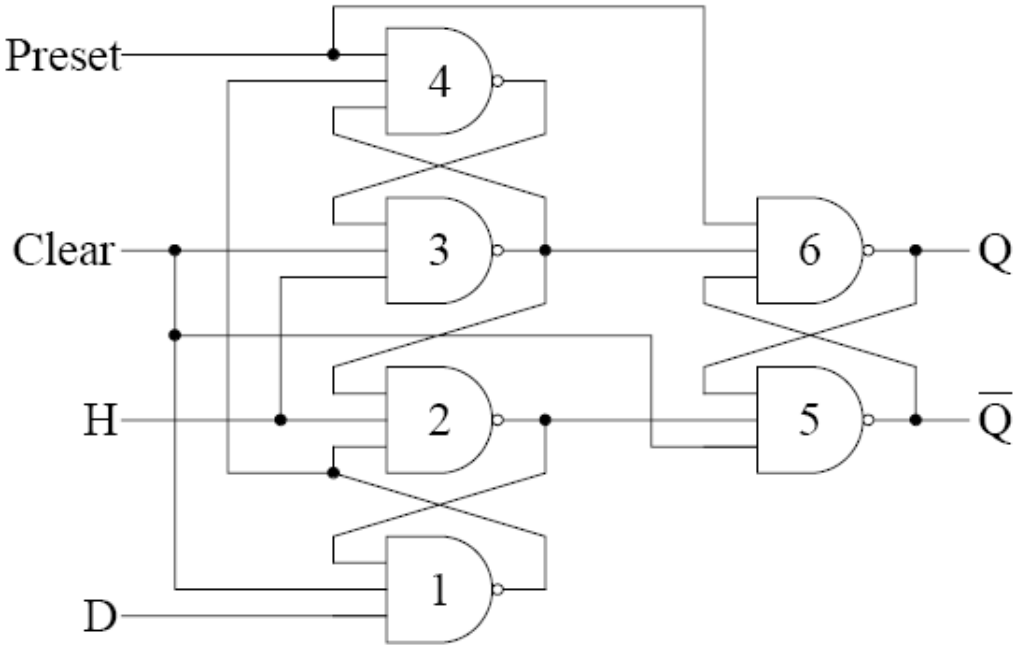
Une bascule D (Delay) est obtenue à partir d'une bascule J-K en envoyant simultanément une donnée sur l'entrée J et son inverse sur l'entrée K :



$$\begin{cases} D_n = 1 \Rightarrow (J_n = 1, K_n = 0) \Rightarrow Q_{n+1} = 1 \\ D_n = 0 \Rightarrow (J_n = 0, K_n = 1) \Rightarrow Q_{n+1} = 0 \end{cases}$$

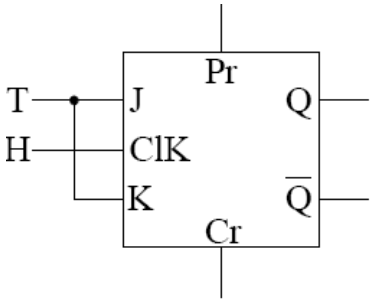
Ce qui peut se résumer par $Q_{n+1} = D_n$. Ainsi l'état de la bascule Q pendant l'intervalle n+1 est égal à la valeur de l'entrée D pendant l'intervalle n. Une bascule D agit comme une unité à retard pour laquelle la sortie suit l'entrée avec un cycle de retard.

Bascule D sur front montant



Bascule JKT

Nous constatons que si J = K = 1 alors $Q_{n+1} = Q_n$. L'état de la sortie est inversé à chaque cycle d'horloge. Une bascule T (Trigger) est obtenue à partir d'une bascule J-K en injectant le même état dans les entrées J et K .

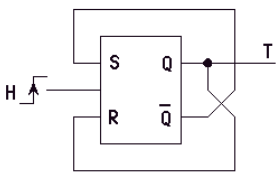


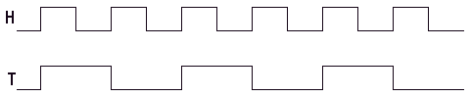
T_n	Q_{n+1}
1	$\overline{Q}_n$
0	Q_n

Le microprocesseur dans sa partie interne travaille en parallèle et a donc besoin de stocker les opérations intermédiaire sur des registres. Un registre peut être représenté par une bascule de type D.

De même que pour la logique combinatoire nous pouvons créer des additionneurs, des comparateurs et des multiplexeurs, etc. Pour la logique séquentielle nous pouvons créer des registres, des compteurs et des diviseurs de fréquence.

Bascule T





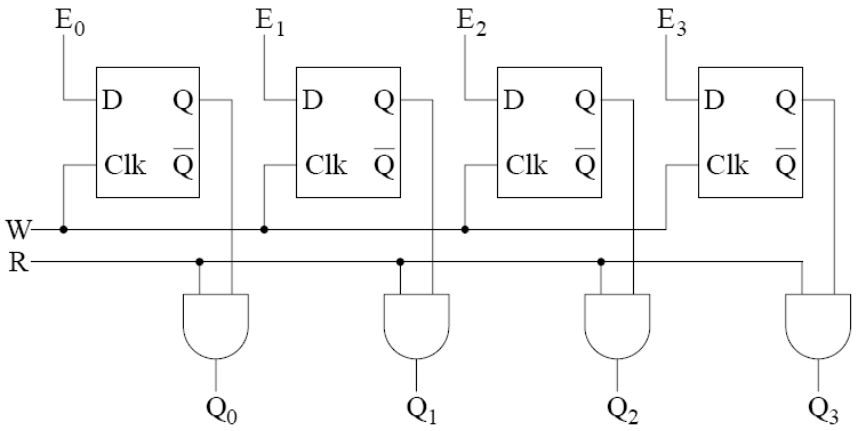
Cette bascule est un diviseur de fréquence. Elle obtenue en bouclant une bascule RS maitre-esclave

Les registres

Les registres sont des composants fondamentaux des microprocesseurs. Pou bien saisir une telle affirmation, il faut que vous vous rappeliez qu'un microprocesseur ne raite pas des signaux numériques limités à un seul bit, mais des mots à 4, 8, 16 ou 64 bits et ce, sur un mode parallèle, c'est-à-dire que tous les bits sont traités en même temps. Ce mode de traitement en parallèle est inhabituel à l'être humain qui travail souvent de façon sérielle.
Pendant le traitement parallèle, les résultats intermédiaires doivent être mémorisés pendant le temps que dure le traitement. C'est généralement à un registre qu'il incombe cette mémorisation de courte durée

Un registre est donc un élément de mémoire de stockage qui permet la mémorisation de n bits en parallèle. Il est constitué de n bascules en parallèle, mémorisant chacune un bit. L'information est emmagasinée sur un signal de commande et ensuite conservée et disponible en lecture.

Les bascules employés sont souvent de type D. Toutes les entrées d'horloge sont reliées ensemble. Ainsi chaque fois le signal de l'horloge est au niveau haut, les bascules enregistrent le mot numérique présent en ce moment sur les entrées D. Ce mot est alors transféré aux sorties jusqu'à ce qu'une nouvelle impulsion d'horloge enregistre le mot suivant.



En synchronisme avec le signal d'écriture W le registre mémorise les états des entrées E0, E1, E2 et E3. Ils sont conservés jusqu'au prochain signal de commande W. Dans cet exemple les états mémorisés peuvent être lus sur les sorties Q0, Q1, Q2 et Q3 en coïncidence avec un signal de validation R.

Registre à décalage

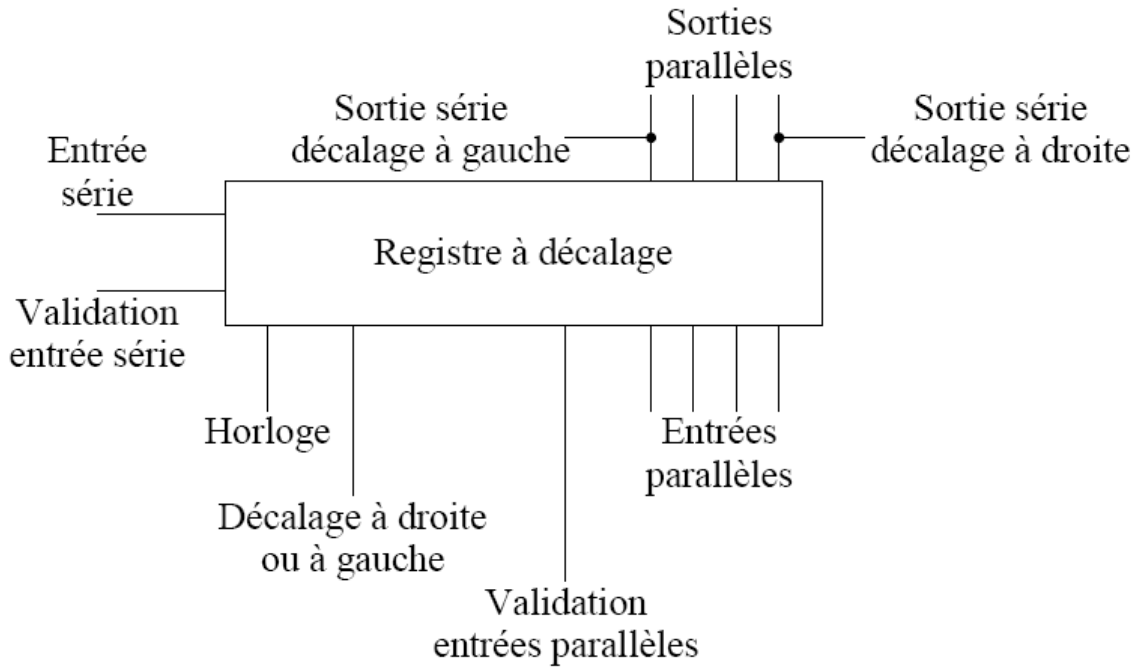
En plus de la mémorisation, ce registre décale les informations vers la droite ou vers la gauche. Un registre de décalage à quatre bits est représenté par quatre bascules D à déclanchement par front raccordés en cascade, la sortie Q de la première sur l'entré D de la seconde, et ainsi de suite.

Dans un registre à décalage les bascules sont interconnectées de façon à ce que l'état logique de la bascule de rang i puisse être transmis à la bascule de rang i+1 quand un signal d'horloge est appliqué à l'ensemble des bascules. L'information peut être chargée de deux manières dans ce type de registre.

- Entrée parallèle : comme dans le cas d'un registre de mémorisation. En général une porte d'inhibition est nécessaire pour éviter tout risque de décalage pendant le chargement parallèle.

- Entrée série : l'information est présentée séquentiellement bit après bit à l'entrée de la première bascule. A chaque signal d'horloge un nouveau bit est introduit pendant que ceux déjà mémorisés sont décalés d'un niveau dans le registre.

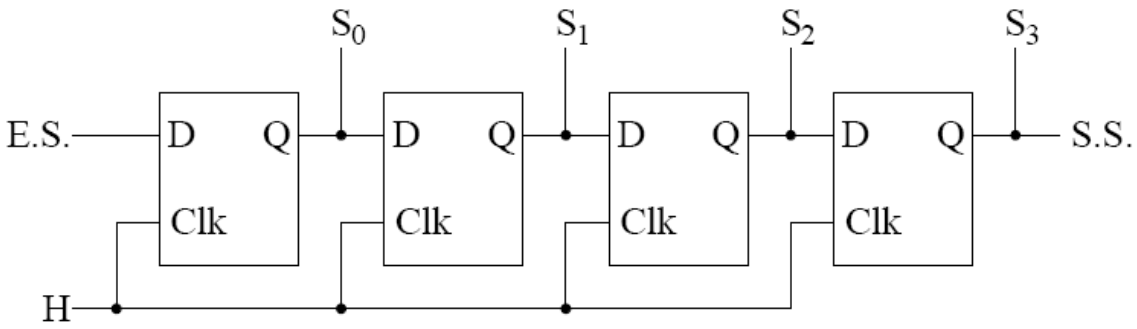
De même l'information peut être lue en série ou en parallèle. D'autre part, certains registres peuvent être capables de décaler à gauche et à droite. Un registre à décalage universel serait donc constitué des entrées, des sorties et des commandes suivantes :



Registre Entrée série - Sortie parallèle

La figure suivante donne un exemple de registre de 4 bits à entrée série et sortie parallèle réalisé avec des bascules D. Ce type de registre permet de transformer un codage temporel (succession des bits dans le temps) en un codage spatial (information stockée en mémoire statique).

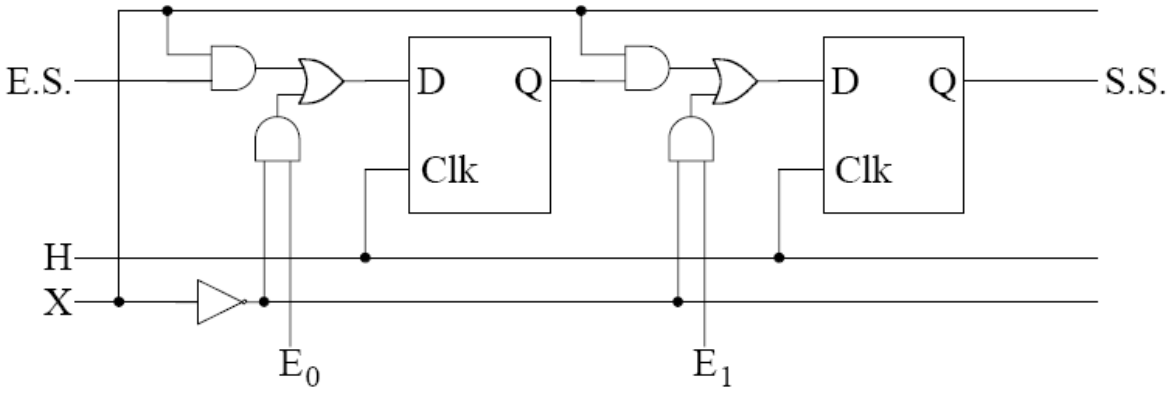
La sortie série peut également être utilisée. L'intérêt d'utilisation d'un registre à décalage en chargement et lecture série réside dans la possibilité d'avoir des fréquences d'horloge différentes au chargement et à la lecture. Le registre constitue alors un tampon.



Registre Entrée parallèle - sortie série

La figure suivante présente un exemple de registre à décalage à entrée parallèle ou série et sortie série. Si X = 1 l'entrée parallèle est inhibée et l'entrée série est validée. Si X = 0 l'entrée série est bloquée par contre le chargement par l'entrée parallèle est autorisé.

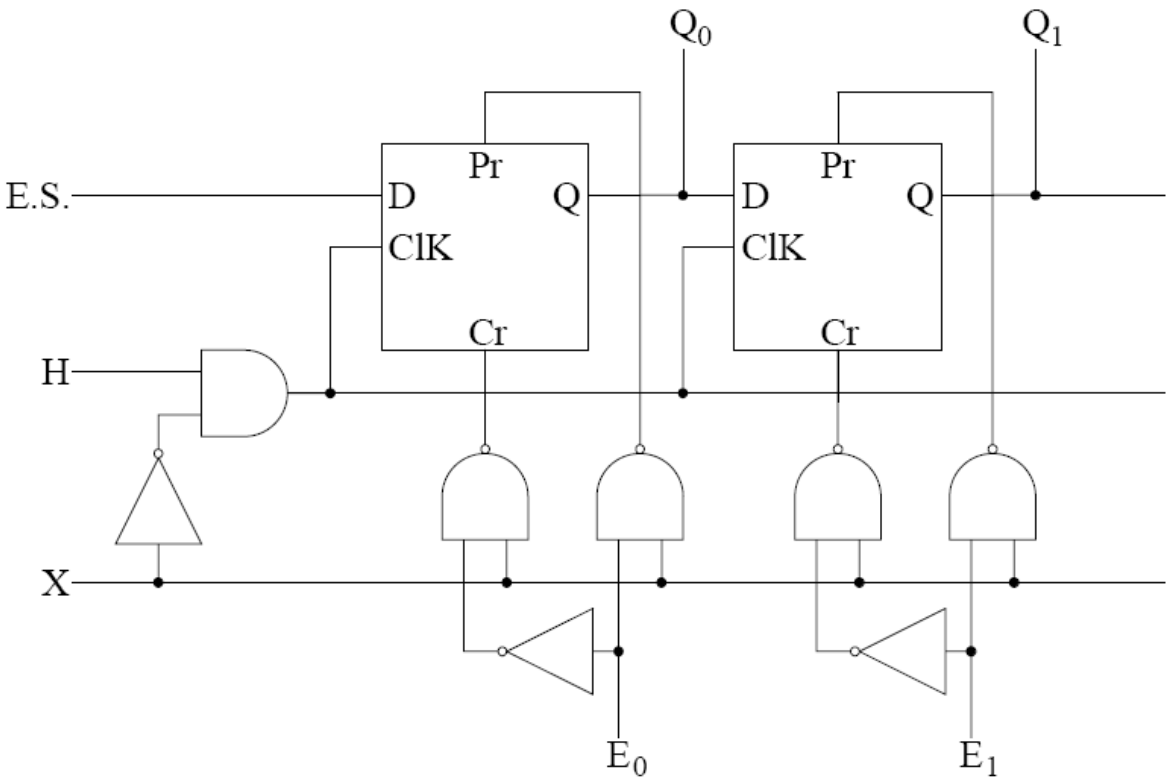
Un registre à décalage à entrée parallèle et sortie série transforme un codage spatial en codage temporel.



Registre à Entrée parallèle - Sortie parallèle

La figure suivante présente un exemple de registre à décalage avec entrées série et parallèle et sorties série et parallèle réalisé avec des bascules de type D.

La commande permet de sélectionner le mode de chargement et d'inhiber le signal d'horloge en cas de chargement parallèle. Si X = 0 nous avons Pr = Cr = 1, ce qui garantit le fonctionnement normal des bascules. Si X = 1 alors selon l'état de chacune des entrées nous avons :



$$\left.\begin{array}{l} E_i = 1 \Rightarrow (Pr = 0, Cr = 1) \Rightarrow Q_i = 1 \\ E_i = 0 \Rightarrow (Pr = 1, Cr = 0) \Rightarrow Q_i = 0 \end{array}\right\} \Rightarrow Q_i = E_i$$

Registre à décalage à droite et à gauche

La figure suivante présente un exemple de registre à décalage universel de 4 bits. Les diverses possibilités sont sélectionnées par les lignes commande S_0 et S_1. Considérons la ligne transportant le signal d'horloge aux bascules, elle est gouvernée par l'expression logique :

$$Clk = \overline{\overline{H} + \overline{S}_0 \bullet \overline{S}_1} = H \bullet (S_0 + S_1)$$

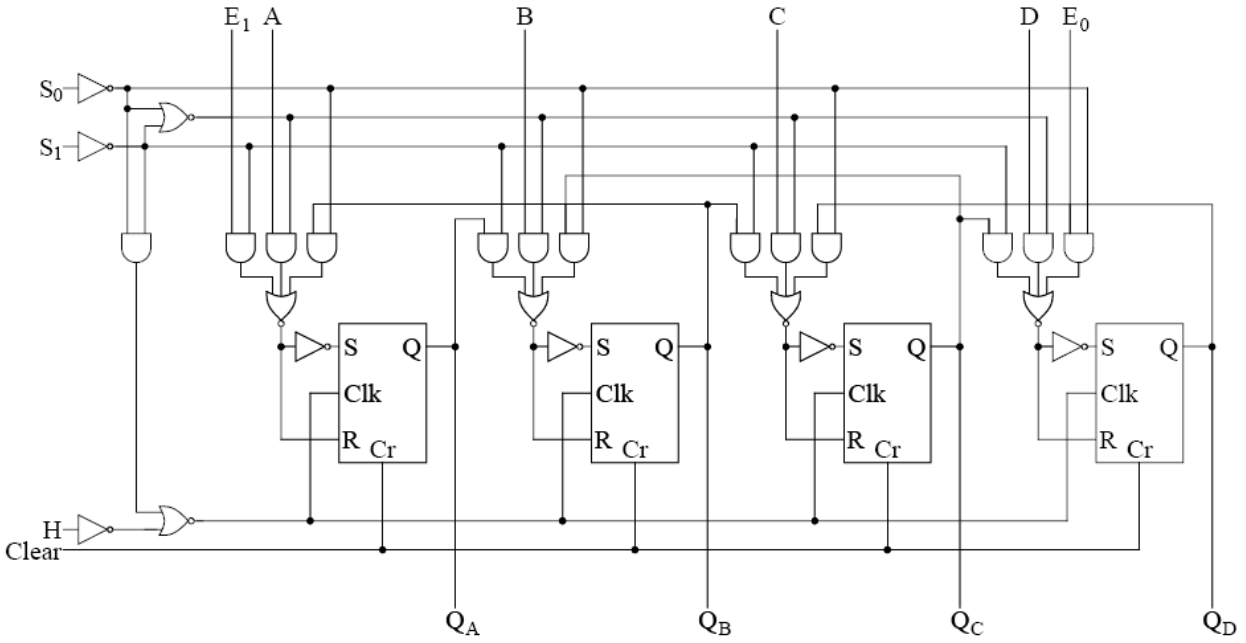
Le signal d'horloge sera donc inhibé si $S_0 = S_1 = 0$.
Pour sélectionner le chargement parallèle (entrées A, B, C et D) il faut :

$$\overline{\overline{S}_0 + \overline{S}_1} = S_0 \bullet S_1 = 1$$

C'est-à-dire $S_0 = S_1 = 1$. Le chargement se fera sur un signal d'horloge.
Pour sélectionner le décalage à droite (entrée E_1, sortie Q_D) il nous faut $S_0 = 1$ et $S_1 = 0$ et pour le décalage à gauche (entrée E_0, sortie Q_A) $S_0 = 0$ et $S_1 = 1$. Ce que nous pouvons résumer dans le tableau suivant :

S_0	S_1	Fonction
0	0	Registre bloqué
0	1	Décalage à gauche
1	0	Décalage à droite
1	1	Chargement parallèle

Un registre à décalage à droite et à gauche permet d'effectuer des multiplications et des divisions entières par des puissances de 2. En effet une multiplication par 2 est équivalente à un décalage vers la gauche et une division par 2 à un décalage vers la droite. Une multiplication par 2^n sera obtenue par n décalages à gauche et une division par 2^n par n décalages à droite.



Compteurs

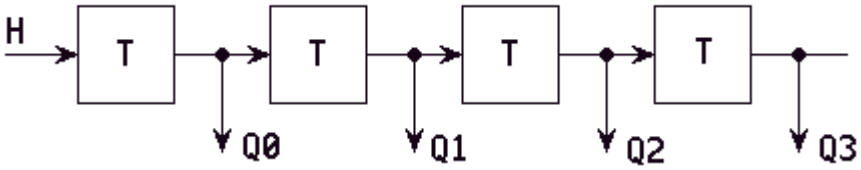
Un compteur est un ensemble de n bascules interconnectées par des portes logiques. Ils peuvent donc mémoriser des mots de n bits. Au rythme d'une horloge ils peuvent décrire une séquence déterminée c'est-à-dire occuper une suite d'états binaires. Il ne peut y avoir au maximum que 2^n combinaisons. Ces états restent stables et accessibles entre les impulsions d'horloge. Le nombre total N des combinaisons successives est appelé le modulo du compteur. On a $N \leq 2n$. Si $N < 2n$ un certain nombre d'états ne sont jamais utilisés.

H	$Q_3\ Q_2\ Q_1\ Q_0$
0	0000
1	1111
0	1111
1	1110
0	1110
1	1101
0	1101
1	1100
0	1100
1	1011
0	1011
1	1010
0	1010
1	1001
0	1001
1	1000
0	1000
1	0111
etc...	

Les compteurs binaires peuvent être classés en deux catégories :
- les compteurs asynchrones;
- les compteurs synchrones.

De plus on distingue les compteurs réversibles ou compteurs-décompteurs.

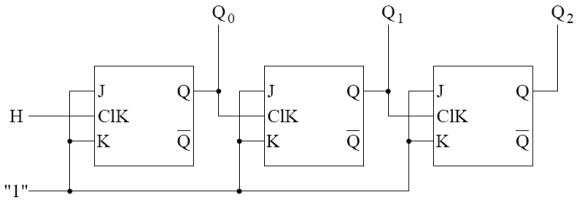
Compteurs asynchrones



Dans l'exemple ci-dessus, les sorties évolueront de la manière suivante au cours du temps, ce qui est bien un décompteur. Pour obtenir un compteur il suffit de complémenter toutes les sorties.
Ce système présente un inconvénient : lorsque plusieurs bascules commutent, elles le font en série, l'une après l'autre, et le bit de poids fort n'est mis à jour que lorsque tous les étages précédents ont fini de commutera: on parle de compteur asynchrone (ou ripple clock counter).

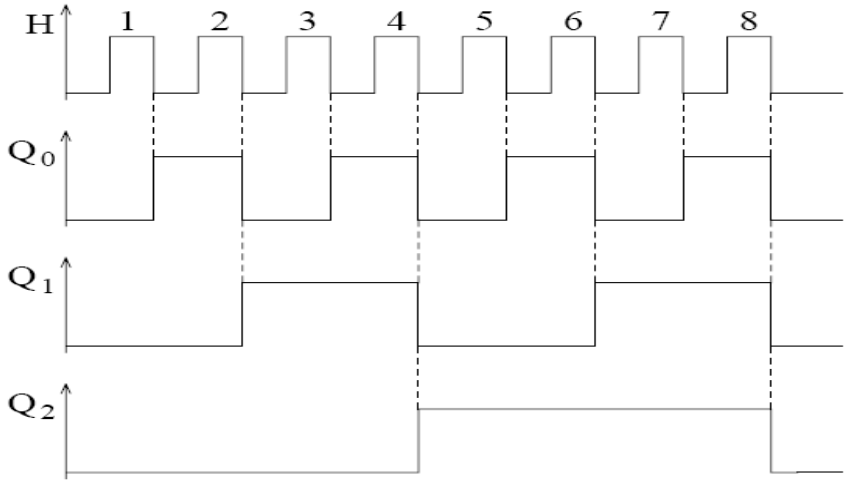
Un compteur asynchrone est constitué de n bascules J-K fonctionnant en mode T. Le signal d'horloge n'est reçu que par le premier étage (bascule LSB : Least Significant Bit). Pour chacune des autres bascules le signal d'horloge est fourni par une sortie de la bascule de rang immédiatement inférieur.

Considérons par exemple un compteur modulo 8 suivant le code binaire pur constitué de trois bascules J-K maîtres-esclaves.



Supposons les trois bascules à zéro à l'instant t = 0. Nous avons vu que pour une bascule maître-esclave la sortie change d'état juste après le passage du signal d'horloge de l'état 1 à l'état 0 (front descendant). L'évolution temporelle des trois sorties Q0, Q1 et Q2 par rapport aux impulsions d'horloge. La sortie Q0 bascule sur chaque front descendant du signal d'horloge. La sortie Q1 change d'état à chaque transition 1 vers 0 de la sortie

Q0. De même le basculement de la sortie Q2 est déclenché par une transition 1 vers 0 de la sortie Q1.



A partir de ce chronogramme nous pouvons écrire la liste des états successifs des trois sorties :

Impulsion	Q_2	Q_1	Q_0
état initial	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
8	0	0	0

Nous avons réalisé un compteur s'incrémentant d'une unité à chaque top d'horloge, avec un cycle de huit valeurs de 0 à 7 (modulo 8).

Nous constatons que les sorties Q_0, Q_1 et Q_2 fournissent des signaux périodiques de fréquences respectivement 2, 4 et 8 plus faibles. La division de fréquence est une des applications des compteurs.

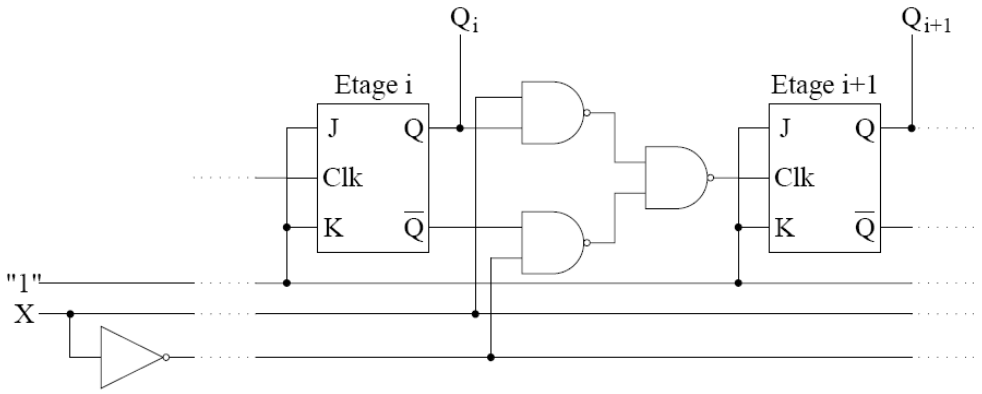
Compteur-décompteur asynchrone

Nous obtenons un compteur en déclenchant chaque bascule lorsque celle de rang immédiatement inférieur passe de l'état 1 à 0. Pour réaliser un décompteur il faut que le changement d'état d'une bascule intervienne lorsque la bascule de rang immédiatement inférieur passe de l'état 0 à 1. Pour cela il suffit d'utiliser la sortie Q de chaque bascule pour déclencher la suivante.

On réalise un compteur-décompteur en utilisant un multiplexeur 2 entrées - 1 sortie entre chaque étage pour sélectionner la sortie à utiliser.

Selon l'état de la ligne de commande X nous pouvons sélectionner le mode de comptage :

X = 1 ⇒ compteur;
X = 0 ⇒ décompteur.



Remise à Zéro et chargement d'un compteur

La figure suivante présente un exemple de montage permettant de remettre à zéro un compteur ou de le charger avec une valeur déterminée. Pour cela on utilise les entrées asynchrones des bascules. En fonctionnement normal du compteur nous devons avoir :

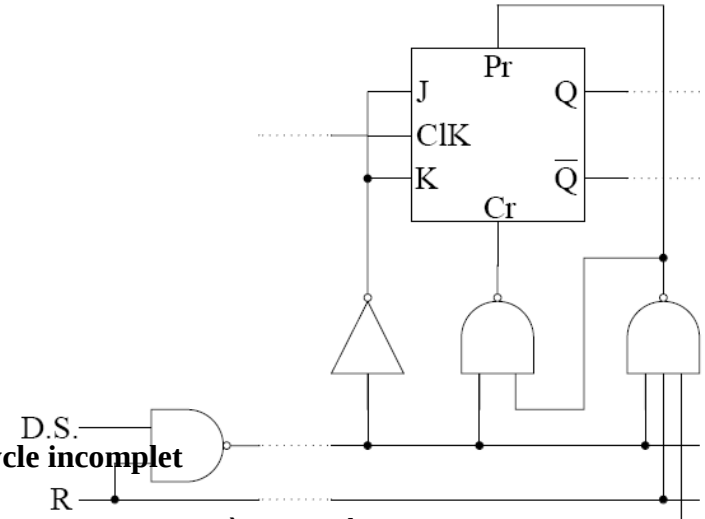
DS = R = 1. Nous avons alors : J = K = Pr = Cr = 1 sur chaque bascule du compteur.

Pour RAZ : R = 0
⇒ J = K = 0 Interdit tout basculement sur une impulsion du signal Clk;
⇒ (Pr = 1, Cr = 0) ⇒ Q = 0.

Chargement : (DS = 0, R = 1) ⇒ J = K = 0 Interdit tout basculement sur une impulsion du signal Clk;
D = 0 ⇒ (Pr = 1, Cr = 0) ⇒ Q = 0

D = 1 ⇒ (Pr = 0, Cr = 1) ⇒ Q = 1

Dans ces deux cas nous obtenons Q = D. Nous sommes donc capable de charger chaque bit du compteur avec une valeur donnée à présenter sur l'entrée D, donc d'initialiser le compteur.



Compteur à cycle incomplet

On peut souhaiter compter jusqu'à un nombre N qui ne soit pas une puissance de 2, par exemple 10 (système décimal). Pour cela on utilise un compteur de n bascules, tel que $2^n > N$. On lui ajoute un asservissement de l'entrée Clear pour remettre le compteur à zéro tous les N coups.

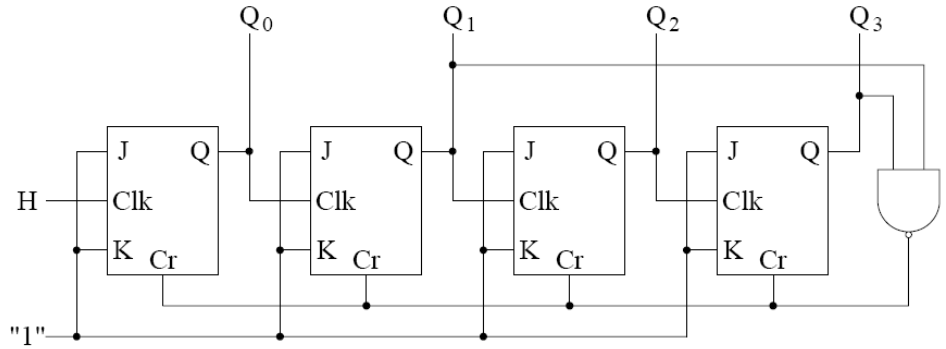
Considérons par exemple un compteur modulo 10. Nous voulons que l'entrée Clear soit à 0 lorsque le compteur atteint $10_{10} = 1010_2$. Pour cela nous pouvons écrire l'expression logique :

$$\overline{Cr} = Q_3 \bullet \overline{Q}_2 \bullet Q_1 \bullet \overline{Q}_0$$

En fait dans ce cas particulier nous pouvons simplifier cette relation logique en ne tenant compte que des sorties à 1 dans l'expression binaire de N. En effet il ne peut y avoir ambiguïté : toute combinaison contenant les mêmes sorties à 1 et au moins une autre à 1 correspond à un nombre plus grand que N et ne peut être rencontrée dans la séquence décrite par le compteur. Pour un compteur modulo 10 nous pouvons donc utiliser :

$$\overline{Cr} = Q_3 \bullet Q_1$$

ce qui nous conduit au schéma suivant :



Compteurs synchrones

Dans un compteur synchrone toutes les bascules reçoivent en parallèle le même signal d'horloge. Pour faire décrire au compteur une séquence déterminée il faut à chaque impulsion d'horloge définir

les entrées synchrones J et K. Pour cela on utilise la table de transition de la bascule J-K. Nous avons déjà remarqué que cette table peut se simplifier. En effet, pour chacune des quatre transitions possibles une seule des entrées J ou K est définie. Rien ne nous interdit donc de les mettre dans le même état, c'est-à-dire J = K, comme dans une bascule T.

Prenons l'exemple d'un compteur synchrone 3 bits fonctionnant selon le code binaire pur. Nous pouvons dresser un tableau précisant les valeurs des entrées J et K permettant d'obtenir chaque transition (passage d'une ligne à la suivante). Pour qu'une bascule change d'état il faut que ses deux entrées soient à 1.

# top	Q_2	Q_1	Q_0	$J_2 = K_2$	$J_1 = K_1$	$J_0 = K_0$
0	0	0	0	0	0	1
1	0	0	1	0	1	1
2	0	1	0	0	0	1
3	0	1	1	1	1	1
4	1	0	0	0	0	1
5	1	0	1	0	1	1
6	1	1	0	0	0	1
7	1	1	1	1	1	1
8	0	0	0			

Chaque ligne de cette table correspond à une même tranche de temps. Il est assez facile d'en déduire les expressions logiques reliant les entrées aux sorties :

$$\begin{cases} J_0 = K_0 = 1 \\ J_1 = K_1 = Q_0 \\ J_2 = K_2 = Q_0 \bullet Q_1 \end{cases}$$

De manière générale nous pouvons vérifier que les équations de commutation satisfont les relations de récurrence suivantes :

$$\begin{cases} J_0 = K_0 = 1 \\ J_i = K_i = Q_0 \bullet Q_1 \bullet \ldots \bullet Q_{i-1} \end{cases}$$

ou encore :

$$\begin{cases} J_0 = K_0 = 1 \\ J_i = K_i = J_{i-1} \bullet Q_{i-1} \end{cases}$$

Procédons de même pour réaliser un décompteur, nous écrivons la table des transitions recherchées :

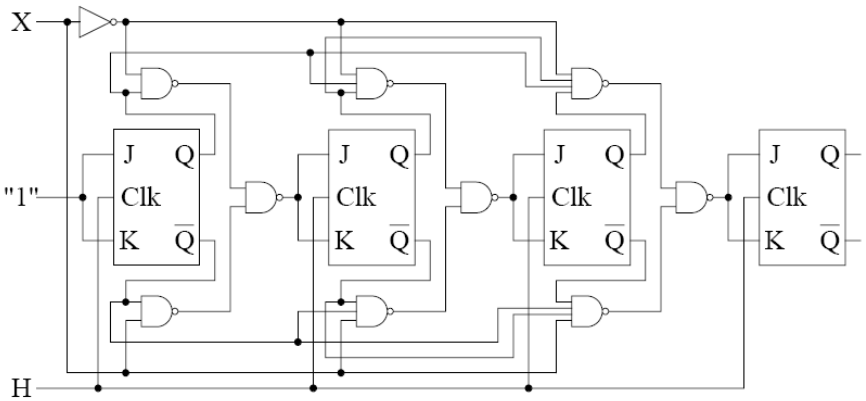
# top	Q_2	Q_1	Q_0	$J_2 = K_2$	$J_1 = K_1$	$J_0 = K_0$
0	1	1	1	1	1	1
1	1	1	0	0	0	1
2	1	0	1	0	1	1
3	1	0	0	0	0	1
4	0	1	1	1	1	1
5	0	1	0	0	0	1
6	0	0	1	0	1	1
7	0	0	0	0	0	1
8	0	0	0			

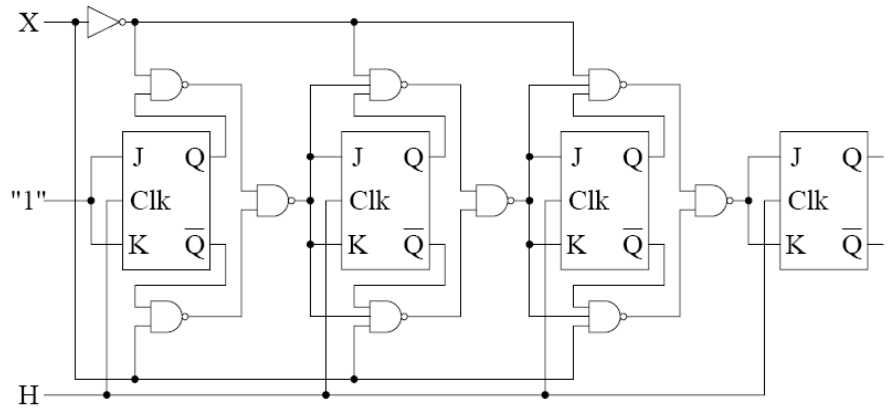
Nous en déduisons l'expression logique des entrées d'un décompteur :

$$\begin{cases} J_0 = K_0 = 1 \\ J_1 = K_1 = \overline{Q_0} \\ J_2 = K_2 = \overline{Q_0} \bullet \overline{Q_1} \end{cases}$$

Nous constatons que les équations de commutation sont identiques en utilisant cette fois les sorties complémentaires Q.
Aux deux manières d'exprimer les relations de récurrence des équations de commutation correspondent deux types de circuits. Le premier est dit à report parallèle, le second à report série. Dans le report série on utilise la fonction J_{i-1}. On évite ainsi des portes à multiples entrées. Par contre, il faut tenir compte du retard dans l'établissement de J_{i-1}. Il faut donc que la largeur des impulsions d'horloge soit assez grande et la vitesse maximum de fonctionnement sera plus faible que pour le report parallèle.

Les deux schémas correspondent respectivement à des compteurs-décompteurs
(X = 0 ⇒ compteur, X = 1 ⇒ décompteur).





L’unité centrale de traitement ou processeur

L'unité centrale de traitement (CPU : Central Processing Unit), encore dénommée processeur ou microprocesseur, est l'élément de l'ordinateur qui interprète et exécute les instructions d'un programme. C'est le cerveau de l'ordinateur. Mais on trouve aussi des processeurs, dits spécialisés, qui peuvent décharger l'unité centrale et assurer des tâches en parallèle. Ceci est très fréquent pour la gestion des entrées/sorties.

Un processeur est aujourd'hui un circuit électronique à très haute densité d'intégration (ULSI : Ultra Large Scale Integration), qui peut compter quelques dizaines de millions de transistors. Le premier circuit de ce type a été créé par Intel en 1971 : le 4004 conçu pour équiper des calculatrices. Il comptait alors 2300 transistors pour 46 instructions. La loi de Moore, formulée en 1965 par un des fondateurs de la compagnie Intel, qui prédit un doublement des capacités des processeurs tous les 18-24 mois, a jusqu'à présent été relativement bien suivie. Il ne s'agit pas seulement de l'augmentation de la fréquence de fonctionnement ou du nombre de transistors. Les concepteurs cherchent aussi à augmenter la quantité de traitement par cycle d'horloge.

Une unité centrale se compose d'au moins deux unités fonctionnelles : l'unité de commande et l'unité de calcul. A l'origine celle-ci s'identifiait à l'unité arithmétique et logique, chargée de l'exécution des opérations booléennes et des opérations arithmétiques (addition, soustraction, multiplication, division, comparaison, etc.) pour des entiers. En parallèle à cette unité, on peut trouver une unité de calcul sur les réels ainsi qu'une unité de traitement dédiée aux opérations multimédia (traitement des images et du son).

A côté de ces deux unités fonctionnelles on trouve une interface de gestion des communications sur le bus externe, ainsi qu'une mémoire cache. Celle-ci est baptisée de premier niveau car située à proximité immédiate du coeur du processeur. Mais depuis quelques années, les concepteurs ont été amenés à embarquer également la mémoire de second niveau. Ces caches peuvent être scindés pour séparer les instructions et les données (architecture Harvard).

L'unité de commande contient une unité chargée du décodage des instructions, une unité pour le calcul des adresses des données à traiter. On y trouve également le séquenceur qui contrôle le fonctionnement des circuits de l'unité de calcul nécessaires à l'exécution de chaque instruction.

L'unité centrale comprend un certain nombre de registres pour stocker des données à traiter, des résultats intermédiaires ou des informations de commande. Parmi ces registres certains servent pour les opérations arithmétiques ou logiques, d'autres ont des fonctions particulières comme le registre instruction (RI) qui contient l'instruction à exécuter, le compteur ordinal (CO) qui pointe sur la prochaine instruction ou un registre d'état (PSW : Processor Status Word) contenant des informations sur l'état du système (retenue, dépassement, etc.).

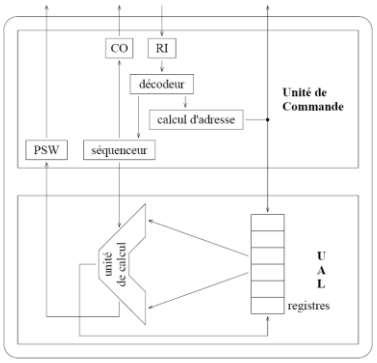
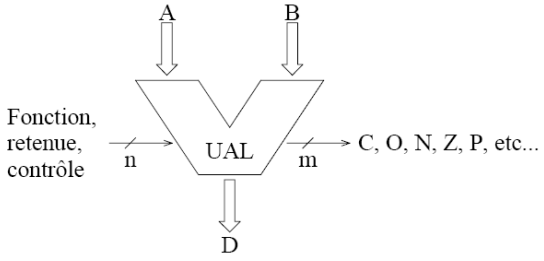


schéma général très simplifié de l'organisation de l'unité centrale sans tenir compte de la présence d'une mémoire cache et de l'interface avec le bus externe.

L'unité arithmétique et logique UAL.

Une unité arithmétique et logique est l'entité qui, dans un processeur, effectue toutes les opérations de traitement arithmétique et logique. L'AUL est constitué ce circuit de logique combinatoire à une échelle de complexité nettement supérieure. Elle doit être capable de faire toutes les opérations requises pour l'exécution d'un programme. Ces opérations sont toute fois très élémentaires. Il s'agit principalement de l'addition (la soustraction pouvant être fait par l'addition en complément de 2), du décalage (pour la multiplication et la division), des opérations logiques, des opérations de comparaison, des opérations de lecture et d'écriture de et vers la mémoire centrale. Le nombre d'opération ou instructions varient selon le processeur, mais tourne aux alentours de 100.
Le choix d'une opération est fait en fournissant à l'Alu le code d'une opération appelée code opératoire. Le cod opératoire d'une instruction du processeur Z80 est décrit sur 8 bits, celui du 68000 l'est sur 16 bits, ce qui signifie que le Z80 a (2^8) 256 instructions et le 68000 (2^{16}) 65535.

On peut cependant additionner deux nombre de 64 bit avec un microprocesseur de 32 bits, mais dans ce cas il faut découper l'opération.



Généralement l'instruction traite deux opérandes (A et B) et produit un résultat C= F(A,B).
L'UAL est un composant essentiel pour l'exécution d'une instruction, mais pour aboutir à un processeur, il faut la compléter par une unité capable d'enchainer automatiquement les instructions d'un programme. Cette unité est un automate appelé séquenceur ou unité de contrôle.

En plus, les résultats de calcul n'étant pas obtenu instantanément, il faudra les garder quelque part pour ensuite les utiliser pour continuer le calcul. Les mémoires qui permettent de stocker els résultats intermédiaires sont appelées registres. L'état des résultats obtenus sont consigné dans un indicateur que le processeur ou le programmeur peut utiliser. Ces états sont consignés dans un registre d'État.

Exemple d'un UAL d'un seul bit

La figure suivante montre l'exemple simplifié d'un UAL a un bit. Elle est capable d'effectuer le ET et le OU de deux bits, le non du second bit et la somme de deux bit avec une retenue d'entrée. Le choix parmi ces quatre opérations se fait via les deux lignes de commande actives f0 et f1.Suivant la valeur de ces deux bits, un décodeur active un des quatre lignes de sortie, sélectionne soit une des trois fonctions logiques ET, OU et NON, soit la retenue et la sortie de l'additionneur.
Pour concevoir une UAL plus large, il suffit de chainer plusieurs UAL de 1 bit, reliant la retenue de sortie d'un étage à la retenue d'entrée suivants.

En dehors de l'UAL, il existe des registres généraux qui sont mis à la disposition des programmeurs et des compilateurs et des registres spéciaux.

Les registres généraux ou banalisés

Ils permettent de limiter les accès à la mémoire, ce qui accélère l'exécution d'un programme. Ils peuvent conserver des informations utilisées fréquemment, desrésultats intermédiaires, etc. Ils sont accessibles au programmeur.

Registres d'indice ou d'index : (XR) Ils peuvent être utilisés comme des registres généraux mais ils ont une fonction spéciale utilisée pour l'adressage indexé. Dans ce cas l'adresse effective d'un opérande est obtenue en ajoutant le

contenu du registre d'index à l'adresse contenue dans l'instruction. Ce type d'adressage et de registre est très utile pour manipuler des tableaux. Le programmeur dispose alors d'instructions permettant l'incrémentation ou la décrémentation du registre d'index. En particulier les registres d'index peuvent être incrémentés ou décrémentés automatiquement après chaque utilisation. Dans certaines machines ces instructions sont applicables à tous les registres généraux, il n'y a alors pas de registre d'index spécifique.

L'accumulateur

Accumulateur (ACC) : L'accumulateur est un registre de l'unité arithmétique et logique. Il a de nombreuses fonctions. Il peut contenir un des deux opérandes avant l'exécution et recevoir le résultat après. Cela permet d'enchaîner des opérations. Il peut servir de registre tampon pour les opérations d'entrées/sorties : dans certaines machines c'est le seul registre par lequel on peut échanger des données directement avec la mémoire. Sa taille est égale à la longueur des mots en mémoire. Il possède souvent une extension (Q), pour les multiplications, décalages, divisions, etc. Le registre ACC est accessible au programmeur et très sollicité. Certaines machines possèdent plusieurs accumulateurs.

Le registre d'état.

Le registre d'état (State register ou Program Status Word (PSW) est tel que ses bits ne forment pas de valeur numérique mais servant d'indicateur (drapeau ou flag) sur l'état du processeur. Certains bits peuvent être positionnés par le programmeur pour demander un comportement particulier.
Les états sont les suivants :
Le bit Z (Zéro). Il est mis à 1 si le résultat est nul, sinon il est mis à 0
Le bit C (Carry).
Le bit N (Négative)
Le bit V (Overflow)
Le bit P (Parity)
La plupart des microprocesseurs ne se servent pas de la totalité des indicateurs. Les indicateurs jouent un rôle important car ils permettent au microprocesseur de prendre des décisions dans certaines situations, et de modifier le déroulement du programme en conséquence.

Registre de pointeur de pile

Lors de l'exécution du programme, le processeur doit parfois stocker de l'information en mémoire, non pas à la demande explicite du programmeur, amis afin d'assurer le bon fonctionnement des instructions. Exemple de l'appelle d'une fonction dans un programme en langage évolué. Le programme est dérouté pour faire appelle à la fonction puis reprend normalement là ou il s'était arrêté. (à l'inverse d'un saut qui le déroute définitivement à une autre adresse).
Il faut donc mémoriser la valeur du PC avant d'exécuter la fonction pour pouvoir reprendre le programme après l'appel.
Son rôle principal est de sauvegarder le nombre binaire contenu dans le compteur ordinal, chaque fois que le déroulement du programme effectue un saut.
A cette fin une structure de pile est implémenté en mémoire et un registre spécial du processeur appelé pointeur de pile ou SP (Stack Pointer) pointe sur le haut de la pile.

Le fonctionnement d'une pile est du type Dernier Entré Premier Sorti (LIFO : Last In First Out). Les deux principales opérations liées à la pile concernent l'ajout d'un élément dans la pile ou le retrait, souvent nomméesrespectivement PUSH et PULL. Lorsqu'une donnée est enregistrée dans la pile elle est placée à l'adresse qui suit celle du dernier mot stocké. Après l'opération le pointeur de pile est incrémenté.
Lorsque un mot est retiré de la pile il correspond à la dernière information qui y a été entrée. Après l'opération le pointeur est décrémenté. Une pile est réservée à l'usage de l'unité centrale, en particulier pour sauvegarder les registres et l'adresse de retour en cas d'interruption ou lors de l'appel d'une procédure. Le pointeur de pile est accessible au programmeur, ce qui est souvent source d'erreur. Certaines machines sont dotées de plusieurs pointeurs de piles.

Pour améliorer les performances d'un processeur il faut disposer du plus grand nombre de registres possible. On réduit ainsi les accès à la mémoire. De plus, il est préférable d'éviter de les spécialiser. On évite ainsi des transferts entre registres, par exemple pour calculer la valeur d'un indice et utiliser ensuite cet indice pour modifier une case d'un tableau.

Le registre d'adresse

Ce registre contient l'adresse servant à repérer l'emplacement mémoire que le microprocesseur désire utiliser. Au début du cycle de recherche et d'exécution, les contenus du registre d'adresse et du compteur ordinateur sont identiques : c'est l'adresse de l'instruction appelée. Mais contrairement au compteur ordinal, le registre d'adresse n'est pas automatiquement incrémenté pendant l'exécution de l'instruction. Il reste sous le contrôle du programme qui peut lui

demander par exemple de rester pointé sur une adresse donnée et modifier ainsi le contenu de l'emplacement de mémoire correspondante en fonction des résultats du traitement obtenu.

L'Unité de commande

L'unité de commande dirige le fonctionnement de tous les autres éléments de l'unité centrale en leur envoyant des signaux de commande. Les principaux éléments de l'unité de commande sont :

- le compteur ordinal (CO) : registre contenant l'adresse en mémoire où se trouve l'instruction à chercher;
- le registre instruction (RI) qui reçoit l'instruction qui doit être exécutée;
- le décodeur qui détermine l'opération à effectuer et les opérandes;
- le séquenceur qui génère les signaux de commande aux différents composants;
- l'horloge (interne ou externe) qui émet des impulsions permettant la synchronisation de tous les éléments de l'unité centrale.

Une horloge est un système logique, piloté par un oscillateur, qui émet périodiquement une série d'impulsions calibrées. Ces signaux périodiques constituent le cycle de base ou cycle machine. Nous avons déjà vu (dans le premier chapitre) les différentes phases de l'exécution d'une instruction. Un cycle d'instruction peut se décomposer en un cycle de recherche (instruction et opérandes) et un cycle d'exécution. On rencontre parfois le terme de cycle cpu pour indiquer le temps d'exécution de l'instruction la plus courte.

N'oublions cependant pas que les performances d'un ordinateur ne dépendent pas de la seule cadence de l'unité centrale. Elles dépendent également des mémoires et des bus, ainsi que de l'architecture, avec par exemple l'utilisation d'antémémoire (mémoire cache) pour anticiper les transferts des instructions et des données.

Le compteur ordinal

Le compteur ordinal est en quelque sorte le gestionnaire qui assure le bon déroulement d'un programme. C'est lui qui cherche, pendant l'exécution d'une instruction, l'emplacement en mémoire centrale de l'instruction suivante. Il anticipe en permanence.

Avant son exécution un programme est mis en mémoire, ses instructions étant placés les unes à la suite des autres. Chacune a donc une adresse précise qu'il faut envoyer au boitier mémoire lorsque que le processeur veut récupérer la dite instruction pour l'exécuter doit donc toujours savoir quelle est la prochaine instruction à exécuter et surtout quel est son adresse. Un registre spécial appelé Compteur Ordinal (CO) ou PC Programm Counter ou IP (Instruction Pointer) contient l'adresse en question.
Pour exécuter une instruction, le processeur commence par envoyer un ordre de lecture mémoire associé à la valeur contenue dans le PC avant de récupérer l'instruction. Le processeur incrémente alors le PC.
Le compteur ordinateur est connecté au bus d'adresse. Une fois récupéré dans la mémoire centrale, l'instruction est expédiée via le bus de données dans le registre d'instruction.

Le registre d'instruction

Lorsqu'une instruction est récupérée en mémoire pour être exécuté dans le processeur, elle est mémorisée dans un registre spécial, le registre d'instruction (RI). Le séquenceur utilise ce lieu de stockage pour disposer de l'instruction et des bits la composant pendant tout le temps de son exécution.
La longueur de ce registre d'instruction est la même que celle des mots du microprocesseur. La sortie de ce registre dirige ce qu'on appelle la logique de décodage et de séquencement qui est chargé de fournir au microprocesseur les divers signaux de commandes nécessaires à l'exécution de l'instruction placé dans le registre d'instruction, en temps voulu et en bon ordre entre les différents organes. Signaux d'écriture dans la mémoire, lecture de la mémoire etc. Ces opérations sont synchronisées par un signal d'horloge. Le séquenceur traite aussi les interruptions qui peuvent être demandé par différents organes.
Cette logique peut être câblée, mais dans la plupart des microprocesseurs elle est microprogrammé. Cela veut dire qu'à l'image du microprocesseur lui-même, elle suit une série de cycles de recherche et d'exécution qui lui sont propres, possède sa propre mémoire et son jeu d'instruction désigné sous le terme de microinstructions.
C'est le registre d'instruction qui, à chaque nouvelle instruction qu'il reçoit, fait démarrer le cycle de recherche et d'exécution des micro instruction de la logique de décodage et de sequencement.

Structures des instructions au niveau machine : format des instructions

Les ordinateurs sont capables d'effectuer un certain nombre d'opérations élémentaires.

Une instruction au niveau machine doit fournir à l'unité centrale toutes les informations nécessaires pour déclencher une telle opération élémentaire : type d'action, où trouver le ou les opérandes, où ranger le résultat, etc. C'est pourquoi une instruction comporte en général plusieurs champs ou groupes de bits. Le premier champ contient le code opération. Les autres champs peuvent comporter des données ou l'identification des opérandes. Sur certaines machines les instructions sont toutes de même longueur, sur d'autres cette longueur peut varier avec le code opération ou le mode d'adressage.

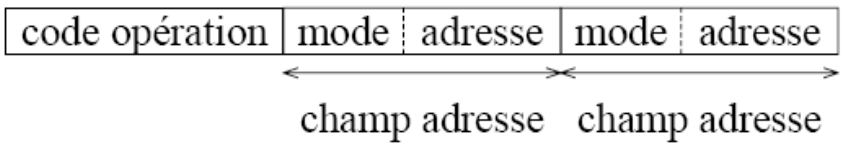
En général une instruction comprend au moins deux parties : l'opération elle-même et les données sur lesquelles elle porte. La première partie appelée code opération correspond au mnémoniques utilisé par l'assembleur (ADD, MOV, SUB, etc.) alors que le second doit être spécifié via des modes d'adressages qui permettent d'indiquer ou son les données (en mémoire, dans un registre, mise explicitement dans l'instruction..)
Chacune de ces deux parties intervient dans le code numérique final de l'instruction. Par exemple, les 8 premiers bits de l'instruction peuvent indiquer le code opération voulu tandis que les 24 autres bits décrivent les données. Ce découpage st le fait du constructeur lorsqu'il définit les jeux d'instruction de son processeur.

On distingue six groupes d'instructions :
- transferts de données : de mémoire à registre, de registre à registre, de registre à mémoire;
- opérations arithmétiques : addition, soustraction, multiplication et division;
- opérations logiques : ET, OU inclusif, NON, OU exclusif, etc.;
- contrôle de séquence : branchements conditionnels ou non, appel de procédure, etc.;
- entrées/sorties;
- manipulations diverses : décalage, conversion de format, permutation circulaire des bits, échange d'octets, incrémentation, etc.

Modes d'adressage

Un champ adresse peut permettre de référencer un registre ou un mot en mémoire. Il peut contenir le numéro du registre ou l'adresse effective du mot mais ce ne sont pas les seules manières d'identifier un opérande. Pour faciliter la programmation il existe de nombreux modes d'adressage.

Le mode est défini soit par le code opération lorsque celui-ci impose un type déterminé, soit par un code faisant partie du champ adresse.



Les exemples ci-dessous sont empruntés aux instructions de l'assembleur PDP-11.

Adressage implicite : Le code opération identifie automatiquement l'opérande, l'instruction ne peut porter que sur un registre particulier (par exemple test sur le signe du résultat d'une opération arithmétique : concerne le registre d'état PSW). Aucun champ adresse n'est nécessaire.

Adressage immédiat : La valeur de l'opérande est contenue dans le champ adresse si le nombre de bits dans ce champ est suffisant, sinon dans le mot suivant l'instruction.
MOV #100, R1
Après cette instruction le registre R1 contient la valeur 100.

Adressage registre : Le champ adresse contient le numéro du registre opérande.
CLR R3
Après cette instruction le contenu de R3 est nul.

Adressage direct : Le champ adresse de l'instruction (ou le mot suivant si le nombre de bits n'est pas suffisant) contient l'adresse effective de l'opérande.
100 : 250
MOV 100, R2

Après cette instruction le registre R2 contient le mot qui se situe à l'adresse 100 en mémoire, c'est-à-dire 250.

Adressage indirect : Le champ adresse (ou le mot suivant) contient l'adresse d'un pointeur : mot en mémoire qui contient l'adresse effective de l'opérande.
MOV (R1), R4
Après cette instruction R4 contient la valeur du mot dont l'adresse est contenue dans R1. Comme R1 vaut 100 on trouve 250 dans R4.

Adressage indexé : Ce mode d'adressage est très utile lorsqu'on travaille, par exemple, sur destableaux. Considérons un bloc de n mots consécutifs débutant à l'adresse A. Le kième mot se trouve à l'adresse A + (k − 1). Pour référencer ce mot il est possible d'utiliser un registre d'index.
L'adresse effective est calculée en additionnant le contenu de ce registre d'index à l'adresse qui se trouve dans le champ adresse de l'instruction. Sur certaines machines tous les registres généraux peuvent être utilisés comme registres d'index. La présence d'un registre d'index s'accompagne généralement de la possibilité d'incrémentation et décrémentation automatiques.
MOV R4, 100(R3)+
CLR 100(R3)

Avant la première opération R3 est nul, donc le contenu de R4 est transféré à l'adresse 100. Après le registre R3 est incrémenté. L'instruction suivante permet de mettre à zéro le contenu du mot à l'adresse suivante.

Adressage basé : L'adressage basé est comparable à l'adressage indexé mais cette fois l'adresse effective est obtenue en additionnant le contenu du registre de base au contenu du champ adresse de l'instruction. Ce mode d'adressage est utilisé par exemple en cas d'allocation dynamique de la mémoire : la position du programme en mémoire peut changer en fonction de la charge du système et il n'occupe pas toujours un espace contigu. Cette technique permet également de réduire le nombre de bits dans le champ adresse : le registre de base contient la première adresse d'un bloc de 2k mots et l'adresse (sur k bits) contenue dans l'instruction représente le déplacement à l'intérieur du bloc.

Adressage relatif : L'adresse effective est obtenue est additionnant le contenu du compteur ordinal au contenu du champ adresse de l'instruction. Ce type d'adressage est utilisé par exemple dans des instructions de branchement.
N'oublions pas que le calcul de l'adresse effective peut nécessiter quelques opérations (addition par exemple). L'utilisation de certains modes d'adressage sophistiqués (le 68020 de Motorola dispose par exemple d'une cinquantaine de modes d'adressage) peut donc augmenter letemps de traitement d'une instruction.

Le Séquenceur

Le séquenceur est un automate distribuant, selon un chronogramme précis, des signaux de commande aux diverses unités participant à l'exécution d'une instruction. Il peut être câblé ou microprogrammé.

Un séquenceur câblé est un circuit séquentiel complexe comprenant un sous-circuit pour chacune des instructions à commander. Ce sous-circuit est activé par le décodeur.

L'idée de la microprogrammation a été introduite par Maurice Wilkes en 1951. Il est en effet toujours possible de remplacer un circuit logique par un transcodeur ou une ROM.

Considérons un ensemble de n fonctions logiques dépendant de m variables logiques. Les valeurs
de ces fonctions pour les N = 2m combinaisons possibles peuvent être calculées (table de vérité)
et mémorisées sous forme de N mots de n bits. Ensuite en utilisant les m variables sous forme d'une adresse il est possible de restituer le résultat recherché.

De même pour reproduire une séquence d'opérations élémentaires il suffit d'un mot par "tranche" de temps. Cette série de mots constitue un microprogramme. Le code opération de l'instruction à exécuter peut être utilisé pour définir le pointeur sur la première microinstruction du microprogramme. En fonction du code opération le contenu d'un compteur est initialisé, puis celui-ci s'incrémente ensuite à chaque cycle d'horloge. La période de l'horloge utilisée à ce niveau peut être plus élevée que celle qui règle la cadence des autres éléments de l'unité centrale. Ce compteur sert à adresser une mémoire morte.

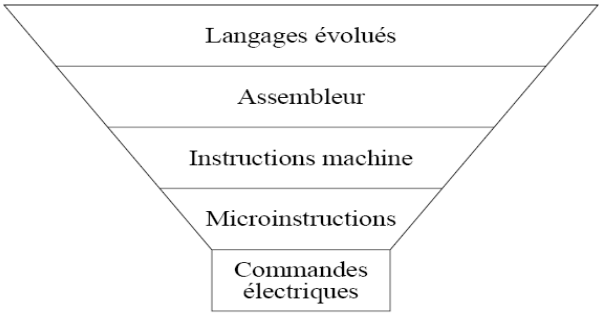
Le format des micro-instructions varie selon les machines. Wilkes a proposé des micro-instructions longues où chaque bit correspond à une ligne de commande. On parle alors de microprogrammation horizontale. A l'extrême une autre solution consiste à utiliser des micro-instructions compactes nécessitant un décodage avant la génération des signaux de commande. La microprogrammation est alors dite verticale. En microprogrammation horizontale la mémoire de commande comprend peu de micro-instructions, chacune comptant un grand nombre de bits. En microprogrammation verticale la longueur des micro-instructions est plus courte mais il y a un plus grand nombre de micro-instructions et il faut un décodage supplémentaire. Dans la pratique on rencontre des micro-instructions mixtes pour lesquelles certains bits agissent directement sur les lignes de commande associées alors que d'autres champs nécessitent un décodage. Un microprogramme peut également contenir des boucles, des tests et des ruptures de séquence. Le compteur est alors remplacé par un micro-séquenceur.

Intrinsèquement un séquenceur microprogrammé est plus lent qu'un séquenceur câblé.
L'avantage et les gains en performance d'un séquenceur microprogrammé résident dans la simplicité de sa conception et la souplesse de son utilisation. Il est ainsi possible d'offrir un jeu d'instructions très complexes. Une instruction peut donc être équivalente à plusieurs instructions d'une autre machine. On gagne alors sur le temps de transfert des instructions. Par ailleurs cela permet une plus grande souplesse aux compilateurs de haut niveau pour optimiser le code objet.

Il est également possible d'augmenter le nombre d'instructions sans augmenter la complexité, donc le coût, du processeur. Nous avons ici un premier exemple de l'imbrication du matériel et du logiciel dans la conception d'une architecture.

Pour programmer un ordinateur on utilise généralement des langages dits évolués ou de haut niveau : C, C++, Java, Basic, Fortran, Pascal, Ada, Assembleur, etc. Cependant l'unité centrale ne peut exploiter que les instructions machine : les codes binaires qui sont chargés dans le registre instruction.

Le terme de langage désigne un jeu d'instructions et de règles syntaxiques. A l'aide d'un langage évolué le programmeur écrit un code source. Celui-ci n'est pas directement exécutable par l'ordinateur. Il faut le traduire en code machine ou code objet. C'est le rôle des compilateurs ou assembleurs et des interpréteurs. Un interpréteur ne produit pas de code objet il traduit les instructions directement au fur et à mesure de l'exécution du programme.
La figure suivante schématise les différents niveaux de programmation. Lorsque l'utilisateur peut accéder au niveau de la microprogrammation la machine est dite microprogrammable.



Les mémoires

Définition

Les mémoires sont des dispositifs qui permettent d'enregistrer, de conserver et de restituer de l'information.

L'unité de base de la mémoire est le **bit (binary digit)** (0 ou 1) qui peut être regroupé pour donner un **octet** (byte) (8bits) ou un **mot** qui est un regroupement d'octets (8 bits, 16 bits, 32 bits, ...)

L'unité d'information adressable en mémoire sont :

un **KiloOctet** = 2^{10} octets = 1024 octets = 1 Ko
un **MegaOctet** = 2^{10} Ko = 1 Mo
un **GigaOctet** = 2^{10} Mo = 1 Go
un **TeraOctet** = 2^{10} Go = 1 To

Technologies et caractéristiques des mémoires

Du point de vue technologiques, les mémoires peuvent être électronique, optique ou magnétique et sont caractérisés par :

- la capacité
- le temps d'accès
- le débit
- et la volatilité

Elle se présente sous la forme de barrettes **SIMM** (Single In-line Memory Module) de formes plus ou moins longues suivant les différentes générations : d'abord adressées sur 8 bits puis sur 32 bits; les constructeurs fabriquèrent ensuite un modèle **EDO (Extended data Out)** plus rapide avec une cache implémentée. Les EDO on remplacé les FMP (Fast Mode Page)

Depuis peu, un nouveau type est apparu : la barrette **DIMM** (64 bits) équipant maintenant la plupart des PC à base de PENTIUM.

Performances des mémoires

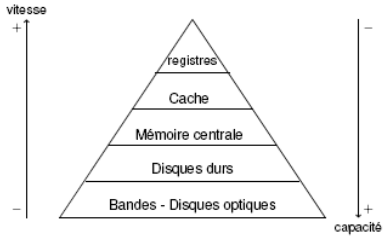
Le temps d'accès (t_a) est le temps qui sépare une demande de lecture/ écriture et sa réalisation
temps de cycle (t_c) est temps minimum entre deux accès à la mémoire.

$$t_a < t_c$$

Le débit ou **bande passante** (B) : nombre de bits maximum transmis par seconde en cas d'accès en temps uniforme au données $B = n / t_c$, n étant le nombre de bits transférés par cycle

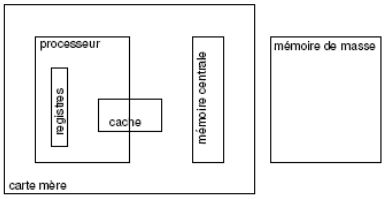
Mais pour des données consécutives, certaines architectures (entrelacement, etc.) et certains modes de lecture (rafales, etc.) peuvent permettre d'approcher la limite : $Bmax = n / t_a$,

Les catégories de mémoires



	vitesse (temps d'acces)	vitesse (debit)	capacite
registres	< 1 ns	> 50 Go/s	< 100 octets
cache	2 - 5 ns	5 - 20 Go/s	100 Ko - 1 Mo
mémoire centrale	20 ns	1 Go/s	256 Mo - 4 Go
disque dur	1-10 ms	300 Mo/s	50 Go - 500 Go

Localisation des mémoires



Méthodes d'accès

Accès séquentiel : pour accéder à une information, il faut parcourir toutes les informations qui la précèdent (bandes magnétiques)

Accès direct : chaque information possède une adresse propre, à laquelle on peut accéder directement (mémoire centrale de l'ordinateur)

Accès semi-séquentiel entre l'accès séquentiel et l'accès direct (disque dur) avec un accès direct au cylindre et un accès séquentiel au secteur sur un cylindre

Accès associatif : l'information est identifiée par sa clé. On accède à une information via sa clé (mémoire cache)

Types de mémoires

Les mémoires mortes

Sur la carte mère cohabite mémoires persistantes (mémoires mortes – ROM -Read Only Memory), leur contenu est figé (ou presque) et conserve en permanence même hors alimentation électrique. On les trouve à divers niveaux, un peu partout dans l'ordinateur, d'autres types de mémoires, non modifiées pendant l'utilisation de l'ordinateur :

On y trouve les programmes de vérifications des divers composants et périphériques, des tests à l'allumage, ainsi que ceux des périphériques qui sont configurés.

Une des mémoires mortes contient entre autres le BIOS (Basic Input/Output System) qui est le programme chargé de créer le lien entre le matériel et les logiciels. Ce programme permet, sans accéder au disque et donc au système d'exploitation, de gérer l'environnement physique de l'ordinateur (vidéo, lecteurs de disquettes, disque(s), mémoire) dont les paramètres sont sur PC en général stockés dans une petite mémoire à faible consommation alimentée en permanence par une batterie.

Le BIOS (Basic Input/Output System ou système de base de gestion des entrées/sorties) est chargé de gérer l'accès à la mémoire, au disque dur, à la carte vidéo, aux interfaces, etc.
C'est lui qui permet par exemple de gérer les **pistes**, **cylindres** et **têtes** d'un **disque dur** afin que les programmes puissent y lire et enregistrer des données sans trop se préoccuper de la manière dont elles seront rangées.

On peut avoir accès au BIOS au démarrage de la machine (boot) afin de déclarer un nouveau disque dur, de la mémoire en plus, ou simplement changer certains paramètres.

Attention: Le BIOS est contenu dans une ROM permanente mais on peut modifier certains paramètres qui sont contenus dans une SRAM (CMOS) dont le contenu reste même après coupure de courant grâce à une petite batterie. Accumulateur au cadmium-nickel, ou pile lithium, cette alimentation de secours préserve les paramétrages CMOS, ainsi que l'horloge en temps réel, qui continue de fonctionner même lorsque l'ordinateur est éteint ou débranché.

Cet autre type de mémoire pouvait être initialement lu mais non modifié. Il s'agit de la mémoire morte ou ROM (de l'anglais Read Only Memory). Tous les PC comportent de la mémoire de ce type. Elle contient les programmes de base permettant de contrôler les éléments matériels.

Ces programmes constituent le **BIOS du PC** (Basic Input Output System : système d'entrée-sortie de base). En fait, le BIOS est composé de plusieurs éléments, répartis dans différents composants. Le principal se trouve sur la carte mère. Il s'agit d'un boîtier DIP (boîtier de plastique noir possédant deux rangées de broches) inséré sur un support. Cette configuration permet de mettre à jour le BIOS en changeant le composant. En fait, il existe également un BIOS sur la carte d'interface de l'écran, ainsi que sur certaines autres cartes d'extension.

Les **ROM** sont fabriquées (on dit fondues) avec le programme qu'elles contiennent. Si cette technique est parfaitement adaptée à la production en masse de PC, elle rend difficile la mise au point des programmes qu'elle contient. Pendant la phase de développement, un programme doit être fréquemment modifié. S'il fallait, à chaque modification, produire des milliers de composants (ou du moins en payer le prix), il serait impossible de fabriquer des PC à prix abordable.

Aussi, on utilise un autre type de composant appelé **PROM** (**Programable ROM**). Pour simplifier, on peut dire que l'information est représentée dans une ROM par un élément conducteur pour la valeur 1 et un élément isolant pour la valeur 0. Une PROM est un composant dans lequel tous les bits valent 1. Lorsque l'on veut y introduire un programme, on place le composant sur un appareil spécial, puis on sélectionne les adresses devant contenir des 0 et on envoie une tension suffisante (12V) pour brûler l'élément conducteur (un fusible), qui prend ainsi la valeur 0.Évidemment, cette opération n'est pas réversible. Les PROM (*Programmable Read Only Memory*) ont été mises au point à la fin des années 70 par la firme*Texas Instruments*.

Les ROM ont petit à petit évolué de *mémoires mortes figées* à des mémoires programmables, puis reprogrammables.

Les EPROM (*Erasable Programmable Read Only Memory*) sont des PROM pouvant être effacées. Ces puces possèdent une vitre permettant de laisser passer des rayons ultra-violets. Lorsque la puce est en présence de rayons ultra-violets d'une certaine longueur d'onde, les fusibles sont reconstitués, c'est-à-dire que tous les bits de la mémoire sont à nouveau à 1. C'est pour cette raison que l'on qualifie ce type dePROM d'*effaçable*.

Les EEPROM (*Electrically Erasable read Only Memory*) sont aussi des PROM effaçables, mais contrairement aux EPROM, celles-ci peuvent être effacées par un simple courant électrique, c'est-à-dire qu'elles peuvent être effacées même lorsqu'elles sont en position dans l'ordinateur. Ces mémoires sont aussi appelées mémoires flash (ou *ROM flash*), et l'on qualifie de flashage l'action consistant à reprogrammer une EEPROM. Bios (“flashable”) - lecteur MP3

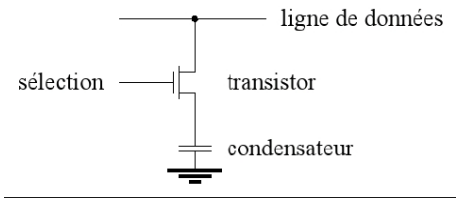
Le flashage du BIOS est donc une *mise à jour du BIOS par voie logicielle*, c'est-à-dire un remplacement de l'ancienne version du BIOS grâce à un programme. Par exemple, il peut être indispensable de flasher le BIOS pour qu’il accepte un nouveau disque dur, beaucoup plus gros que ce que le BIOS peut reconnaître.

La "shadow ROM".
La ROM d'un PC contient les programmes de base qui contrôlent son fonctionnement. Il s'agit donc de

Programmes utilisés avec une fréquence très élevée. Etant donné que les ROM sont beaucoup plus lentes que les mémoires de types RAM (une ROM a un temps d'accès de l'ordre de 150 ns tandis qu'une mémoire de type SDRAM a un temps d'accès d'environ 10 ns), les instructions contenues dans la ROM sont parfois copiées en RAM au démarrage, on parle alors de ***shadowing*** (en français cela pourrait se traduire par *ombrage*, mais on parle généralement de *mémoire fantôme*) ou de ***shadow ROM*** (ROM fantôme). Chaque fois qu'une fonction du BIOS doit être employée, elle est lue dans la RAM, qui est d'un accès beaucoup plus rapide.

Les mémoires vives

DRAM : Dynamic RAM
Dynamique : l'information doit être périodiquement rafraîchie réalisation : 1 bit = 1 transistor + 1 condensateur le condensateur stocke l'information doit être rafraichit régulièrement (pour conserver la valeur stockée dans le condensateur) cela ralentit la vitesse d'accès à la mémoire . Elle est peu couteuse. Elle sert beaucoup en vidéo, en particulier sur les cartes graphiques, et permet pour une résolution d'écran donnée d'afficher davantage de couleurs.



SDRAM (Synchronous DRAM) : qui est une DRAM dont l'accès est synchrone ; c'est a dire que chaque requête mémoire se fait en un seul cycle d'horloge.

EDO (Extended Data Output) : Elle est structurée comme la DRAM, à une petite différence près : un petit circuit a été ajouté , qui agit comme une minuscule zone de stockage ou tampon servant à sauvegarder les adresses. Ce tampon reçoit l'adresse de la prochaine donnée à lire ou à écrire avant même que la donnée précédemment lue ou écrite ait été traitée.

SRAM : Static RAM

Statique : l'information n'a pas besoin d'être rafraîchie. Pour sa réalisation : 1 bit = 4 transistors = 2 portes NOR, Bascule RS (ou D) qui stocke l'information. Elle est beaucoup plus **rapide et beaucoup plus chere** que la DRAM

VRAM (Vidéo RAM)

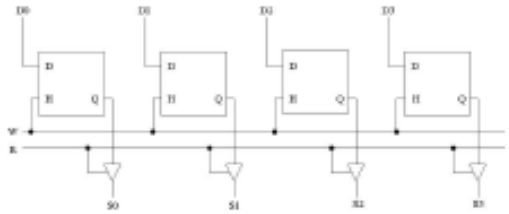
Cet autre type de mémoire est plus rapide que la DRAM car elle permet la lecture et l'écriture de données simultanément. Elle équipe les cartes vidéo (cette mémoire est également **volatile**).

Registres

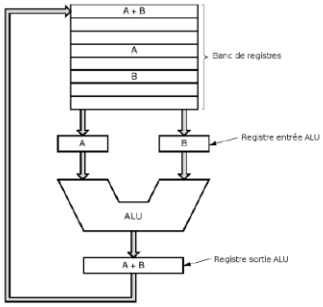
Mémoire de type SRAM intégrés au cœur du processeur un registre stocke les informations relatives à une instruction opérandes nécessaires à l'instruction résultats produits par l'instruction très peu nombreux (< 20) très rapides (cadences à la vitesse du processeur)

Réalisation :
Registre 1-bit = 1 bascule RS (ou D)
Registre n-bits = n bascules RS (ou D) en parallèle

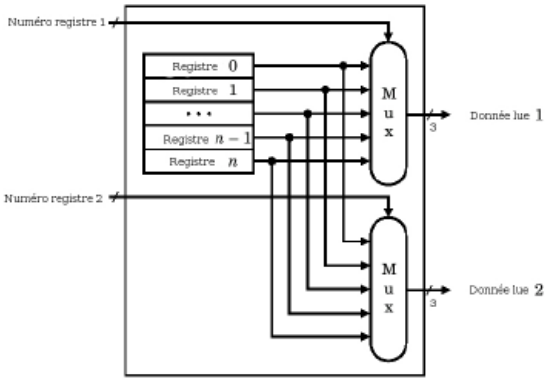
registre 4 bits

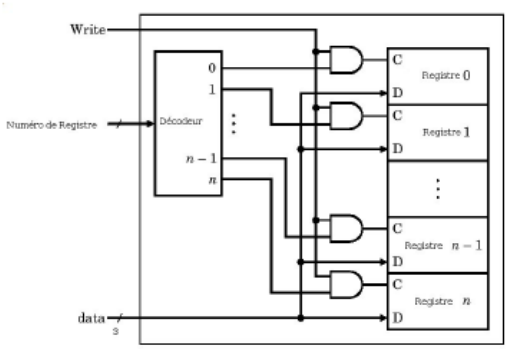


Fonctionnement en lecture



Fonctionnement en écriture





Organisation de la Mémoire centrale

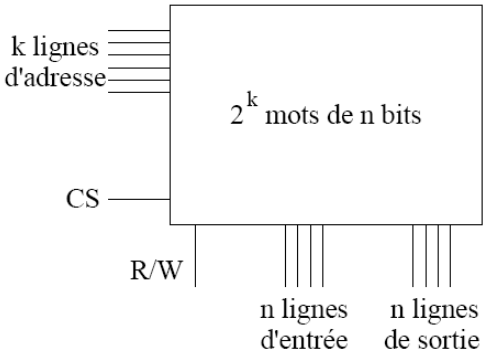
Mémoire unidimensionnelle : nombre de portes dans le décodeur trop important

Extérieurement, et en ne tenant compte que des signaux logiques, un bloc mémoire peut être représenté comme sur la figure suivante. Pour pouvoir identifier individuellement chaque mot on utilise k lignes d'adresse. La taille d'un bloc mémoire est donc 2^k, le premier mot se situant à l'adresse 0 et le dernier à l'adresse 2^k - 1.

Une ligne de commande (R/W) indique si la mémoire est accédée en écriture (l'information doit être mémorisée) ou en lecture (l'information doit être restituée).

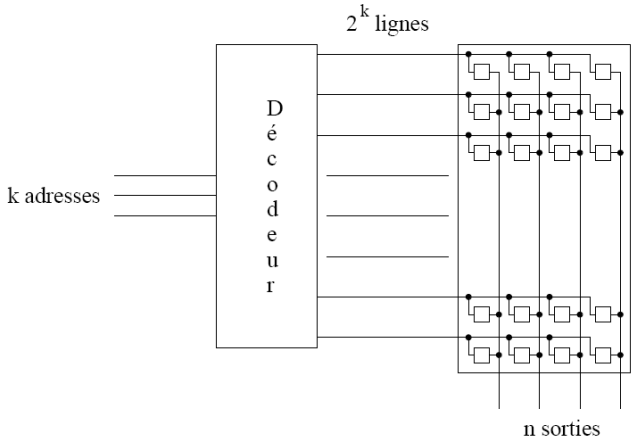
Sur ce schéma on distingue deux canaux de n lignes en entrée et en sortie, mais dans d'autres cas les accès en entrée et en sortie peuvent être confondus en un seul canal bidirectionnel.

Nous verrons l'intérêt de la ligne de validation ou de sélection du bloc (CS) un peu plus loin.



Adressage bidimensionnel ou matriciel

L'organisation des cellules à l'intérieur d'un bloc la plus simple à imaginer correspond au schéma suivant, chaque ligne correspond à un mot de n bits :

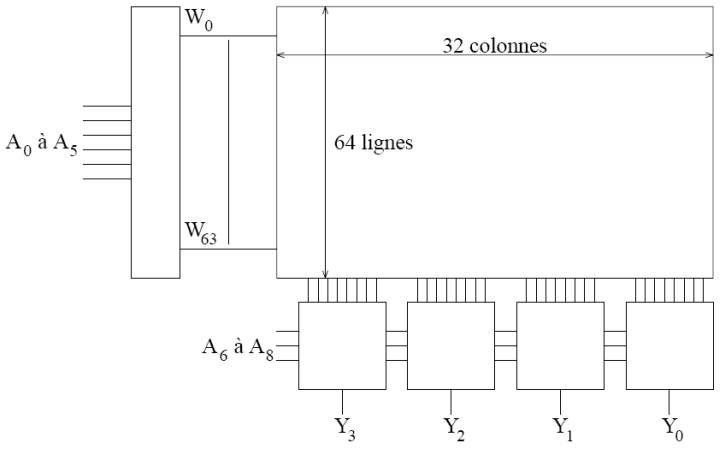


Pour simplifier la figure, chaque cellule y est matérialisée avec uniquement une ligne de sélection et une ligne de sortie. Si la ligne de sélection est à "0", la cellule est "isolée" de la sortie. Si la ligne de sélection est à "1", l'information mémorisée se retrouve sur la ligne de sortie.

La figure correspond au mécanisme de lecture, mais le principe est également valable en écriture. En fonction de l'adresse, le décodeur active une des 2^k lignes. Ainsi seules les cellules correspondant à l'adresse demandée sont sélectionnées et l'information mémorisée est alors disponible en sortie. Cette architecture très simple n'est pas la plus économique en terme de nombre de portes.

Considérons par exemple une mémoire contenant 512 mots de 4 bits soient 2048 bits. C'est-à-dire k = 9 et n = 4. Il faut 512 portes ET pour réaliser le décodeur. Une économie importante peut être obtenue en organisant la mémoire en une matrice de 64 lignes et 32 colonnes (2048 = 64 x 32). Chacune de ces 64 lignes peut être sélectionnée en utilisant 6 bits d'adresse.

Comme nous ne voulons que des mots de 4 bits, il faut encore utiliser 4 multiplexeurs chacun sélectionnant une ligne sur 8. Les 3 derniers bits d'adresse sont affectés à ces multiplexeurs. Cette
architecture est dénommée adressage X-Y ou bidimensionnel. Dans ce cas 64 portes ET sont nécessaires pour le décodeur et 9 portes (8 ET et 1 OU) pour chacun des multiplexeurs, soit au total 64 + (9 x 4) = 100 portes. Cela représente cinq fois moins de portes.



L'organisation matricielle des blocs mémoires permet également d'éviter des pistes trop longues pour la distribution des différents signaux aux cellules. Les constantes de temps et les pertes de charge sont ainsi réduites.

Nous n'avons considéré que deux cas de figure pour l'organisation matricielle du bloc mémoire : 512 lignes de 4 colonnes et 64 lignes de 32 colonnes. D'autres variantes sont évidemment possibles, pour certaines le nombre de portes nécessaires est résumé dans le tableau suivant. Nous constatons que le minimum se situe pour une organisation "carrée", pour laquelle les nombres de lignes et de colonnes sont égaux ou diffèrent d'un facteur 2.

nb lignes	nb colonnes	Décodeur	Multiplexeur	Total
512	4	512	0	512
256	8	256	3	268
128	16	128	5	148
64	32	64	9	100
32	64	32	17	100
16	128	16	33	148
8	256	8	65	268

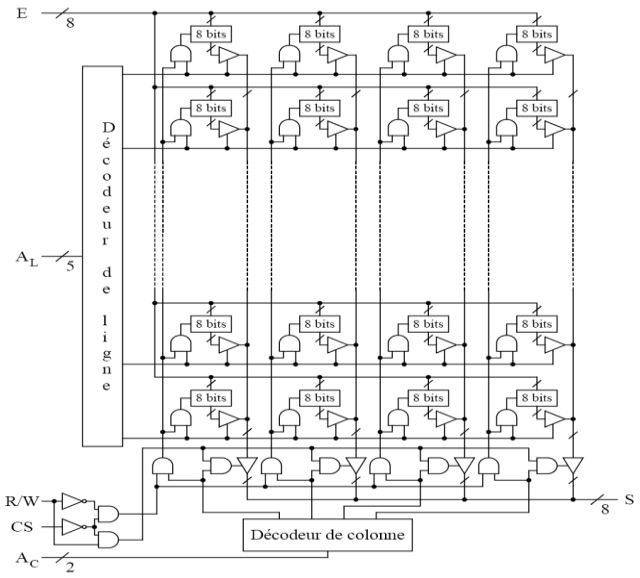
Très souvent, les blocs mémoires comportent autant de lignes que de colonnes. Les mêmes lignes d'adresse peuvent alors être utilisées pour identifier successivement la ligne puis la colonne. Cela permet de réduire le nombre de broches de connexion, donc l'encombrement et le coût des circuits. Cependant cela demande environ deux fois plus de temps par transmettre l'adresse complète. Par contre, si on cherche à accéder à des informations stockées dans une même ligne il peut être possible de définir une fois la ligne, puis pour chaque mot de n'avoir à envoyer que l'adresse de la colonne. Il faut alors au moins un signal de commande supplémentaire pour indiquer que l'adresse correspond à une ligne ou une colonne.

Parmi les caractéristiques d'une mémoire nous trouvons la capacité et le format. La capacité représente le nombre total de bits et le format correspond à la longueur des mots. Le nombre de bits d'adresse k définit le nombre total de mots de la mémoire, si n est le nombre de bits par mot, la capacité de la mémoire est donnée par :

Capacité = 2^k mots = 2^k x n bits
Cette capacité est exprimée en multiple de 1024 ou kilo. La table suivante résume la valeur des autres préfixes utilisés pour exprimer les capacités des mémoires :

Symbole	Préfixe	Capacité		
1 k	(kilo)	2^{10}	=	1024
1 M	(méga)	2^{20}	=	1048576
1 G	(giga)	2^{30}	=	1073741824
1 T	(tera)	2^{40}	=	1099511627776

La figure suivante, présente une organisation logique possible pour une mémoire de 128 mots de 8 bits. Ici chaque mot est stocké dans une case de 8 bits, tel un registre. Cette case reçoit en entrée 8 lignes de données et une ligne de chargement. Elle dispose de 8 lignes de sortie fournissant le contenu du registre. Chacune de ces lignes est commandée par une porte "3 états". Ces cases sont organisées en une matrice de 32 lignes et 4 colonnes. Les 7 bits d'adresse sont séparés en deux groupes, 5 bits pour identifier la ligne et 2 bits pour la colonne.



Le signal de sélection du boîtier (CS) valide les fonctions d'écriture et de lecture. Le boîtier est bloqué pour CS = 1. Lorsque CS = 0, une fonction d'écriture correspond à R/W = 0 et une fonction de lecture à R/W = 1.

En écriture, le mot à charger doit être présent sur l'entrée E du circuit. Ces données sont distribuées simultanément sur toutes les cases de 8 bits. La ligne désignée par l'adresse AL est à 1.

Le signal de chargement est transmis à la seule colonne identifiée par l'adresse AC. Seul le registre à l'intersection de cette ligne et de cette colonne est donc chargé.

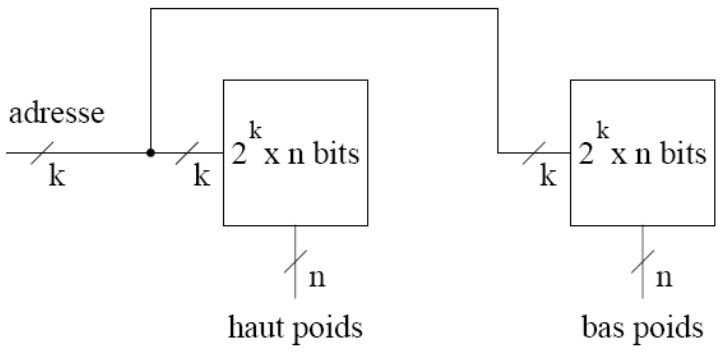
En lecture, les quatre cases de la ligne sélectionnée fournissent leur contenu sur les quatre bus verticaux. Une seule des quatre portes "3 états", au bas du schéma, est connectée à la sortie S du boîtier. Cette porte "3 états" fournit une amplification des signaux.

Assemblage de blocs mémoires

Les techniques d'intégration ne permettent pas d'obtenir des boîtiers ayant des capacités ou des formats suffisants pour toutes les applications. Il est alors nécessaire d'associer plusieurs boîtiers pour augmenter la longueur des mots ou le nombre de mots. L'association de plusieurs blocs peut permettre d'améliorer les performances temporelles de la mémoire en faisant fonctionner plusieurs blocs en parallèle.

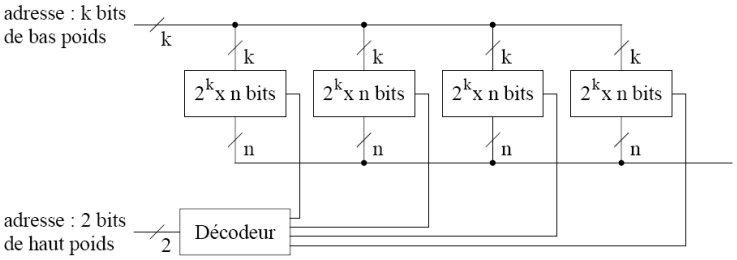
Augmentation de la longueur des mots

La figure suivante montre qu'il est aisé d'associer deux boîtiers de 2k mots de n bits pour obtenir un bloc de 2k mots de 2 n bits. L'adressage doit être appliqué simultanément aux deux circuits, l'un fournissant les n bits de bas poids et l'autre les n bits de haut poids.



Augmentation du nombre de mots

De même la figure suivante montre la réalisation d'un bloc de 4 x 2^k mots de n bits à l'aide de 4 boîtiers de 2^k x n bits. Il nous faut k+2 lignes d'adresse. Les k bits de bas poids de l'adresse sont appliqués simultanément sur les 4 boîtiers. Les deux bits de haut poids attaquent un décodeur à quatre sorties. Chacune de ces quatre lignes permet de sélectionner un boîtier (entrée de validation du boîtier : CS). Un seul boîtier est alors connecté aux lignes de sortie.



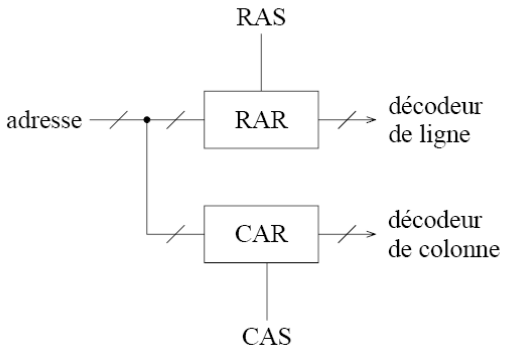
Réduction du nombre de broches

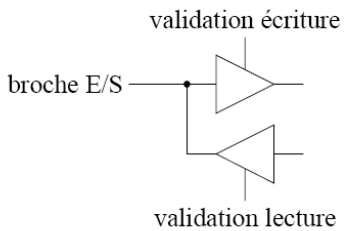
Les fabricants de mémoires cherchent à réduire le nombre de broches des boîtiers, pour augmenter la densité d'implantation sur les circuits imprimés. Deux méthodes peuvent être utilisées pour réduire le brochage des signaux logiques.

Il est possible de diviser par deux le nombre de broches pour l'adressage en chargeant successivement l'adresse de ligne puis l'adresse de colonne (fig. 10). La mémoire a alors une organisation "carrée", avec autant de lignes que de colonnes. Deux signaux (RAS : Row Address Strobe et CAS : Column Address Strobe) sont nécessaires pour charger ces deux adresses dans deux registres internes (RAR : Row Address Register et CAR : Column Address Register). Le contenu de ces registres est exploité respectivement par les décodeurs de ligne et de colonne.

Pour accéder à plusieurs mots d'une même ligne, il n'est pas nécessaire de charger l'adresse de la ligne à chaque requête, il suffit de modifier l'adresse de colonne, on parle alors d'accès en mode page.

On peut également tirer profit de ce qu'on n'effectue jamais une lecture et une écriture simultanément. Il est donc possible d'utiliser les mêmes broches en lecture et en écriture. Pour cela on fait appel à des porte "3 états".





Consommation d'une mémoire

Les constructeurs cherchent à limiter la consommation des boîtiers de mémoire qui peuvent être nombreux dans un système. Pour certaines mémoires on peut distinguer trois états avec chacun un niveau différent de consommation. **L'état actif** correspond au cas d'une opération de lecture ou d'écriture. La puissance dissipée dépend du débit.

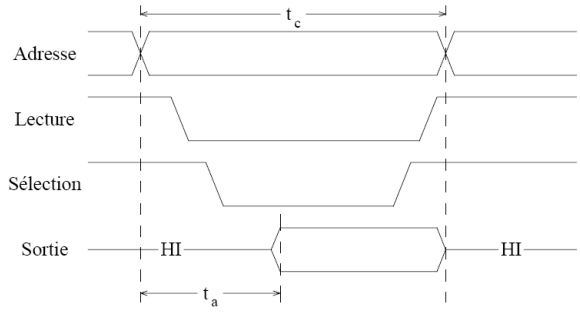
Dans l'état **inactif ou passif**, la mémoire alimentée normalement n'est pas sélectionnée. Les commandes de lecture et écriture envoyées sur un boîtier non sélectionné n'agissent pas. Les boîtiers qui consomment moins dans cet état sont dits "automatic power down".

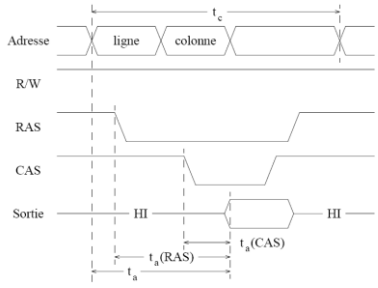
Dans l'état de **veille**, la mémoire est alimentée sous tension réduite avec des batteries. Elle ne peut être ni lue, ni écrite. Elle se contente de conserver son information : un circuit interne avec horloge et compteur permet un rafraîchissement automatique régulier de toutes les cellules sans intervention externe. Cet état permet de réduire la consommation.

Fonctionnement de la mémoire

Cycle de lecture

- établissement de l'adresse
- signal de lecture (R/W=0 par exemple)
- sélection du boitier (CS=0)
- Après un certain temps, l'information apparait sur la sortie et reste présente jusqu'à la fin du cycle.



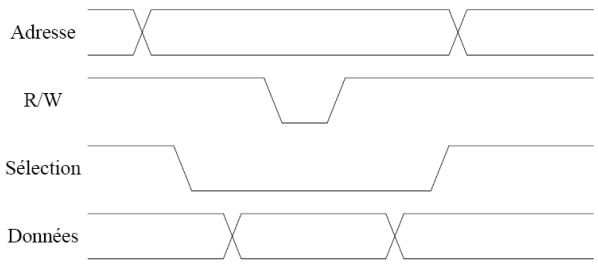


Il s'agit uniquement d'un chronogramme de principe. La polarité, la durée de stabilité de chaque signal, ainsi que les retards relatifs (ou phases) entre signaux sont précisés par le constructeur.

Lorsque les adresses de ligne et de colonne sont multiplexées celles-ci doivent être stabilisées avant les signaux de chargement RAS et CAS respectivement. Le temps d'accès ta est mesuré à partir du début de l'opération, mais on distingue deux autres temps d'accès mesurés à partir des fronts descendants des signaux RAS et CAS. Dans l'exemple illustré, la ligne R/W à 1 indique une fonction de lecture.

Cycle d'écriture

- Établissement de l'adresse
- Sélection du boitier (CS=0)
- Établissement de la donnee sur l'entrée
- Signal d'écriture (R/W=0 ci-dessous)



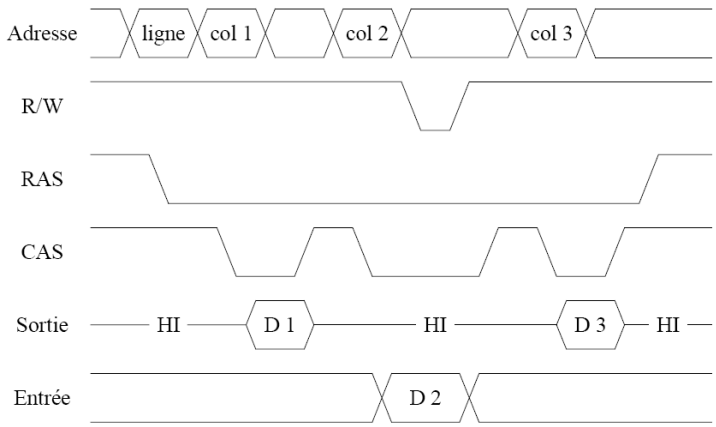
Cycle de lecture-modification-écriture

Ce cycle permet de lire une donnée à une certaine adresse, de modifier la donnée et de l'inscrire à la même adresse. Le même résultat pourrait être obtenu avec deux cycles successifs de lecture et d'écriture, mais plus lentement car il faut alors présenter l'adresse deux fois. Il faut que les lignes d'entrée et de sortie soient distinctes. La procédure consiste à :

- Établir l'adresse;
- Sélectionner le boîtier correspondant;
- Afficher la fonction lecture;
- La donnée apparaît sur la sortie au bout du temps d'accès;
- Le système modifie la donnée qu'il applique sur l'entrée;
- Établir une impulsion d'écriture.

Lecture ou écriture en mode page

Le mode page concerne les mémoires avec multiplexage des adresses ligne et colonne. Au cours de l'exécution d'un programme, l'accès aux informations (instructions ou données) se fait plus souvent à des adresses consécutives qu'à des adresses isolées. L'accès en mode page permet alors d'améliorer les temps d'accès. En effet lorsque l'adresse de la ligne (ou page) reste identique, il n'y a pas lieu de la présenter à nouveau. Dans un cycle en mode page, l'adresse de ligne est chargée une fois, chaque accès est ensuite défini par l'adresse de colonne. Le temps du premier accès est équivalent à celui d'un accès aléatoire. Il est appelé latence. Par contre les accès suivants sont plus rapides. La figure suivante illustre une séquence de trois accès comprenant une écriture intercalée entre deux lectures.



Rafraîchissement des mémoires dynamiques

Pour la plupart des mémoires dynamiques les adresses sont multiplexées. A chaque accès, en lecture ou en écriture, chaque bit de la ligne désignée est transféré en bas de colonne dans une bascule. Ce transfert est initié par le front descendant du signal RAS. Les opérations effectives de lecture ou d'écriture se font au niveau de ces registres. Ceux-ci sont ensuite chargés dans la ligne d'origine sur le front montant du signal RAS. En procédant ainsi on rafraîchit la ligne adressée à chaque accès. Cela ne garantit pas un rafraîchissement suffisant de l'ensemble de la mémoire. Il est possible de déclencher le même mécanisme de rafraîchissement d'une ligne sans faire aucunaccès. Il suffit de préciser l'adresse de la ligne à rafraîchir puis d'activer une impulsion d'une durée définie sur la ligne RAS. Par ailleurs les mémoires vives contiennent aussi un pointeur interne sur la prochaine ligne à rafraîchir. Il est utilisé en activant la ligne CAS avant la ligne RAS.

Le mécanisme de rafraîchissement des divers blocs mémoires est confié à un contrôleur. C'est également ce contrôleur qui se charge de multiplexer les adresses.

Protocoles d'échanges processeur-mémoire

Synchrone : au bout de k unités de temps, le processeur suppose que l'opération sur la mémoire a été réalisée (mot écrit en mémoire, mot lu disponible sur la sortie)

Asynchrone (handshaking) : processeur et mémoire s'échangent des informations de contrôle (**request/acknowledgment**)

Mémoire synchrone

Tous les accès étudiés jusqu'à présent sont de type asynchrone. Ils peuvent intervenir à tout instant et leur durée est fixée par les composants de la mémoire.

Ces dernières années le fossé entre la vitesse de fonctionnement des processeurs et la vitesse d'accès des mémoires n'a fait que croître. La mémoire constitue aujourd'hui le goulot d'étranglement des architectures. Les gains en performance sur la vitesse intrinsèque des composants sont relativement faibles. Les fabricants de mémoires jouent sur l'organisation logique des accès. Nous en avons déjà vu un exemple avec le mode page, dont de nombreuses variantes ont été mises en oeuvre.

Un autre mode permet de réduire le temps nécessaire à l'adressage des colonnes. Il s'agit du mode d'accès en rafale (burst) qui concerne plusieurs colonnes consécutives. Un compteur interne permet d'éviter le temps nécessaire à l'établissement des signaux d'adresse et à leur chargement. Il suffit donc de préciser initialement l'adresse de départ et le nombre de colonnes.

Dans un composant asynchrone l'incrémentation de ce compteur est provoqué par des cycles du signal CAS.

Mais il est également possible de faire appel à un signal d'horloge. Nous obtenons alors une mémoire synchrone (SDRAM pour Synchronous Dynamic Random Acces Memory). Cette solution a pour avantage de simplifier la gestion de la mémoire. Ce qui permet un léger gain de vitesse, mais aussi allège la conception du contrôleur. Il est ainsi possible d'avoir un accès à chaque cycle d'horloge. Par contre le temps d'accès pour une seule donnée n'est pas inférieur à celui d'une mémoire asynchrone. Il peut même être plus important.

Pour parvenir à suivre une fréquence intéressante les fabricants mettent également en oeuvre des techniques déjà rencontrées dans ce chapitre. Par exemple, à l'intérieur d'un même circuit la mémoire est organisée en deux blocs (ou plus) qui peuvent être entrelacés. Cela permet également l'ouverture simultanée de deux pages. Il est alors possible de passer d'une page à une autre sans ralentir la vitesse d'accès.

Gestion de la mémoire centrale

Au début de l'informatique les mémoires étaient chères et de petites tailles. A cette époque les programmeurs passaient l'essentiel de leur temps à optimiser leur programme en fonction de la taille mémoire. On a ensuite pu utiliser une mémoire secondaire : bandes, tambours et disques.

Le programmateur divisait alors son programme en un certain nombre de morceaux appelés branches (overlays), chaque branche étant de taille inférieure à celle de la mémoire. Pour exécuter un programme on chargeait la première branche puis la seconde et ainsi de suite. Le programmeur était totalement responsable de la découpe du programme et de la gestion de la migration des branches entre la mémoire principale et la mémoire secondaire.

En 1961 des chercheurs de Manchester ont proposé une méthode qui permettait de rendre transparent au programmeur le mécanisme de gestion des branches. Au début des années 1970 cette technique, dite de mémoire virtuelle, devint disponible sur la plupart des ordinateurs. L'idée préconisée par le groupe de Manchester consiste à distinguer les notions d'espace d'adressage et d'emplacement mémoire.

Optimisations

Mémoire synchrone (synchronisée avec le bus) : **SDRAM**

Pour les mémoires matriciels, **accès en mode page** : on charge
ligne et colonne, puis on ne change que les colonnes pour les accès suivants (localité des données) : **DRAM FPM**

Pour les mémoires matriciels, **accès en rafale (burst)** : on charge ligne et colonne ainsi que le nombre de données à lire ; incrémentation dans la mémoire des colonnes pour les accès
suivants (localité des donnees). **DDR-SDRAM**

Mémoire et erreurs

Du fait de sa nature “physique”, les informations en mémoire peuvent
comporter une ou des erreurs. Pour détecter et corriger, on ajoute des **bits de contrôle**. **bit de parité :** 1 bit supplémentaire (en plus des bits de données)
tel que le nombre de bits `a 1 est pair

mémoire ECC (Error Correction Coding) possède des bits supplémentaires pour d´ détecter et corriger le(s) bit(s) erroné(s).

La mémoire logique est la façon dont le processeur (ou le programmeur) voit la mémoire (physique).

La mémoire est définie comme un ensemble de N octets consécutifs dont la première adresse est 0 la dernière adresse est N-1

Adressage de la mémoire par des mots de : 8 (octet), 16, 32, 64, .. bits.

Un mot de 32 bits est constitué de 4 octets consécutifs

Pour un mot mémoire de 32 bits, il existe 2 fac¸ons de ranger les octets
qui le compose :
Le mot de poids fort est stocké en premier : **big-endian**

i	i+1	i+2	i+3
Octet 3	Octet 2	Octet 1	Octet 0

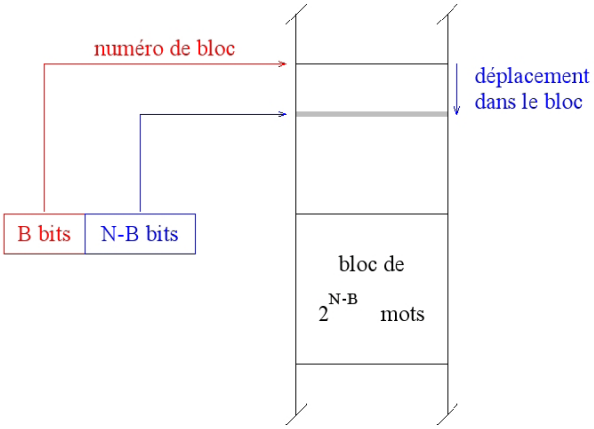
Le mot de poids fort est stocké en dernier : **little-endian**

i	i+1	i+2	i+3
Octet 0	Octet 1	Octet 2	Octet 3

Un mot mémoire ne peut commencer n’importe où
les mots de 16 bits commencent sur des adresses paires
les mots de 32 bits commencent sur des adresses multiples de 4

Segmentation de la mémoire : découpage logique de la mémoire en un certain certains nombres de blocs (ou segments). Une adresse est codée comme un numéro de blocs un déplacement dans le bloc (**offset**)

N bits d'adresses donne 2^N cases mémoire.
Les N bits sont séparés en deux :
B bits (de poids fort) pour le numéro de bloc
N-B bits (de poids faible) pour le déplacement dans le bloc
On a donc 2^B blocs ayant chacun 2^{N-B} cases.



Mémoire Virtuelle

Il existe une autre forme de mémoire appelée **mémoire virtuelle**. Il ne s'agit pas d'un type de composant particulier, mais d'une **technique** permettant de simuler à moindre coût la présence d'une quantité de mémoire importante.

La mémoire physique n'est qu'une partie de la mémoire disponible : le système permet l'utilisation de la mémoire de masse (disque durs) comme de la **mémoire virtuelle**.
Pagination de la mémoire virtuelle
Mécanisme de **swap**

Cache mémoire

Mémoire tampon entre la RAM et le processeur. Très rapide (plus que la RAM)

Notion de cache ou antémémoire. C'est une mémoire temporaire qui fait tampon entre deux différents types de mémoires permettant de stocker certaines informations utilisées fréquemment et d'y accéder plus rapidement.

Elle sont soit :

- entre les disques et la mémoire vive: cache disque
- entre la mémoire vive et les registres (ie le processeur): cache mémoire,

Types de cache :

Cache L1 :

Le processeur de votre ordinateur est très rapide, il peut tourner à des vitesses dépassant 2 Ghz. La mémoire RAM standard est plus rapide que la plupart des autres composantes de votre ordinateur, mais elle ne tient pas le coup face à des vitesses approchant celle du processeur. Ainsi les concepteurs de processeur (CPU) ont réservé un espace mémoire très rapide à l'intérieur même de la puce du processeur. C'est la cache L1, elle fonctionne à la même vitesse que le processeur. L'espace sur la puce du processeur coûte très chère, il n'est alors pas possible de réserver une grande quantité de mémoire pour la cache L1 (approximativement 1/1000 de la taille totale de la mémoire RAM de votre ordinateur). Puisque les programmes d'ordinateur effectuent beaucoup de boucles autour du même code, les instructions et données de la cache L1 sont exécutées souvent. Même une petite cache L1 peut donner un gain très appréciable en vitesse. Plus la vitesse du processeur est rapide par rapport à la mémoire principale RAM et plus la quantité de données sur laquelle vous travaillez est importante, plus grande devra être la cache L1.

Cache L2 :

La mémoire rapide situé dans le processeur (cache L1) est de petite taille, et la cache L1 ainsi que le processeur sont encore beaucoup plus rapides que la mémoire RAM (jusqu'à 50 fois plus rapide). Lorsque que les données ne sont pas dans la cache L1, le processeur doit aller les chercher dans la mémoire RAM, il y a alors un ralentissement notable. Le processeur doit alors attendre un long moment (par rapport à sa vitesse) pour que la mémoire RAM lui rende l'information, le processeur ne peut alors rien faire d'autre qu'attendre. Ainsi entre la cache rapide L1 (et le processeur) et la mémoire lente RAM, est insérée une deuxième cache, la cache niveau 2 ou L2. Cette cache est fabriquée à partir de mémoire rapide, mais relativement peu dispendieuse, appelée mémoire statique ou SRAM et est approximativement 10 fois plus rapide que la mémoire RAM standard. On peut ainsi se permettre 256k et même 1 ou 2 megabytes de SRAM. Le processeur travaille presque toujours sans arrêt, il fera probablement quelque chose qu'il a fait récemment (une boucle) et les données seront dans la cache L1 ; ou il possédera la logique qui lui aura permit de lire d'avance (prefetch) l'information située dans la cache L1 ; cependant les ordinateurs exécutent des centaines de millions d'instructions à la seconde ! 1% de ratés dans la cache L1 signifie que pendant un million de fois par seconde le processeur devra lire dans la mémoire RAM. Les ratés dans la cache L1 peuvent atteindre 10 à 20%, ainsi la cache L2 apporte un aide précieux. Lorsque le processeur ne trouve pas ce dont il a besoin dans la cache L1, il le trouvera la plupart du temps dans la cache L2. Le fait que la mémoire RAM soit plus lente que le processeur devient alors beaucoup moins significatif. Lorsqu'on entend parler de cache, il s'agit la plupart du temps de la cache L2. La majorité des ordinateurs personnels d'aujourd'hui possède les deux niveaux de cache (L1 et L2).

Cache L3 et L4 :

Certains systèmes sont tellement rapides qu'ils ont besoin de plusieurs niveaux de cache entre le processeur et la mémoire RAM. Chacun de ces niveaux possède le numéro suivant dans la séquence de numérotation des caches (L1, L2, L3, etc.).

Cache de disque dur :

De la même manière que le processeur est plus rapide que la mémoire RAM, celle-ci est beaucoup plus rapide que les disques durs. Une cache est utilisée entre le disque dur et la mémoire, lorsque l'ordinateur écrit sur le disque, les données sont placées dans la cache, elles sont alors écrites lentement (à la vitesse maximale du disque dur) alors que l'ordinateur peut s'occuper à faire autre chose. Lorsque l'ordinateur désire lire sur le disque, la cache peut avoir lu d'avance (prefetch) ou posséder des données lues auparavant, celles-ci sont alors tirées directement de la cache sans avoir à passer par le disque dur. Non seulement l'ordinateur peut-il lui même utiliser une partie de sa mémoire RAM comme cache entre lui et le disque dur (cache logicielle), les concepteurs de disques durs ont également ajouté une petite quantité de mémoire directement

sur les contrôleurs de disques durs comme cache (cache matérielle) et c'est la norme actuellement dans la fabrication des disques durs. Cependant les gens ont de la difficulté à différencier entre la cache logicielle et la cache matérielle du disque dur, soyez donc prudent lorsque vous utiliserez cette terminologie ; les caches font la même chose, mais différemment.

Cache de CD-ROM :

Les disques durs sont rapides comme l'éclair comparativement à la majorité des lecteurs CD-ROM. Des ingénieurs astucieux ont décidé d'utiliser le disque dur ou la mémoire RAM ou les deux à la fois pour accélérer les accès au CD-ROM. Ils fonctionnement comme les caches de disques durs et conservent les données temporairement sur le disque dur ou dans la mémoire RAM, jusqu'à ce que l'ordinateur ait besoin de lire. Rappelez-vous que les CD-ROM ne peuvent être que lus, il n'y donc pas de cache en écriture pour ceux-ci. (On peut écrire sur certains lecteurs CD, mais ils ne sont pas appelés CD-ROM, plutôt CD-R). Il y a également des caches de lecture d'avance (prefetch) directement sur les lecteurs CD-ROM.

En conclusion, une cache est placée entre deux composantes possédant des vitesses différentes. Quelquefois la cache est matérielle, mais plus souvent elle est logicielle. Les caches ne sont parfois utilisées que pour emmagasiner de l'information, elles peuvent posséder également leur propre logique leur permettant de se vider et lire d'avance (prefetch) ce que vous risquez d'avoir besoin plus tard. Merci aux caches, nos ordinateurs et leurs composantes fonctionnent beaucoup plus rapidement grâce à elles.

Le cours est complété par les exposés suivants qu'il faut lire pour la préparation de l'examen.