Conservatoire National des Arts et Métiers

Polycopié de cours

Electronique A4

Version provisoire du lundi 19 janvier 2004

Circuits numériques : 1ère partie

C.ALEXANDRE

Conservatoire National des Arts et Métiers

Polycopié de cours

Electronique A4

Version provisoire du lundi 19 janvier 2004

Circuits numériques : 2ème partie

C.ALEXANDRE

1. Logique combinatoire

1.1 De l'analogique au numérique

Une des principales évolutions de l'électronique depuis 1980 est le remplacement du traitement analogique des signaux par le traitement numérique. Depuis l'introduction du Compact Disc au début des années 1980, les chaînes de traitements du son et de l'image ont été numérisées de bout en bout. Bien que le monde qui nous entoure soit entièrement analogique, on préfère convertir le plus tôt possible les grandeurs analogiques en grandeurs numériques afin de bénéficier des importants avantages du traitement numérique du signal.

1.1.1 Grandeurs analogiques

Une grandeur analogique évolue de manière continue au cours du temps. Elle peut prendre toutes les valeurs comprises entre deux extrêmes : par exemple une tension de 12,341 volts. Toutes les grandeurs physiques qui nous entourent sont à priori analogiques :

- Tension et courant.
- Température.
- Dimension, distance, vitesse, pression, volume, …
- Son (onde sonore acoustique) et image (luminosité et couleurs).

Un capteur permet de transformer ces grandeurs en signaux électriques analogiques (tension ou bien courant). Par exemple, le microphone transforme une variation de pression en tension. On peut ensuite effectuer une opération quelconque sur ces signaux (mesure, enregistrement, amplification, …).

Exemple de signal analogique : le sillon gravé sur les « anciens disques microsillons » (33 tours ou 45 tours). Le capteur est la tête de lecture à aiguille et bobine magnétique, qui convertit alors cette gravure (le sillon) en signal électrique prêt à être amplifié.

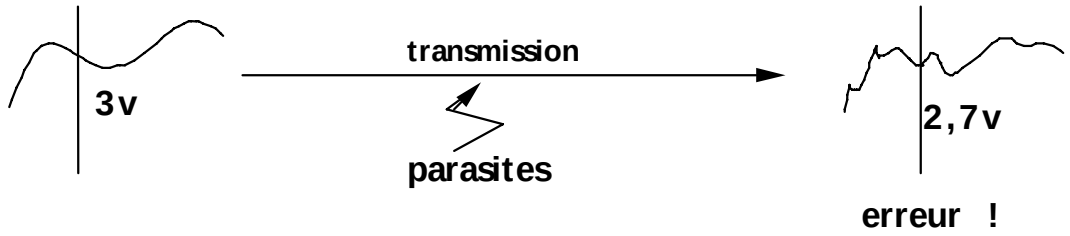
Exemple d'instruments ou d'appareils analogiques :

- Indicateur de vitesse à aiguille, thermomètre à mercure, …
- Haut-parleurs, microphones …
- Les magnétophones à cassettes.
- Les anciens électrophones.

- Les amplificateurs HiFi.

Quels sont les inconvénients des signaux analogiques ?

a) Sensibilité aux parasites lors d'une transmission.



Lorsqu'un signal est transmis, même sur une courte distance, il est soumis à l'influence de bruits de toutes sortes : bruits impulsifs de machines électriques dans le cas d'un environnement industriel, « ronflement 50Hz » dû à la proximité des lignes du secteur EDF, ...). A cause du bruit, une tension émise de 3 volts peut, après transmission, ne valoir que 2,7 volts. L'erreur est importante et peut être gênante.

b) Conservation des données.

L'enregistrement est relativement aisé (bandes magnétiques, gravure des microsillons), mais les données se dégradent progressivement dans le temps : par exemple, dégradation assez rapide de la qualité des anciens disques microsillons (craquements, souffle…), perte de qualité sonore sur les cassettes (pertes surtout de clarté dans les aiguës et bruit de fond plus important).

c) Capacité de traitement limité.

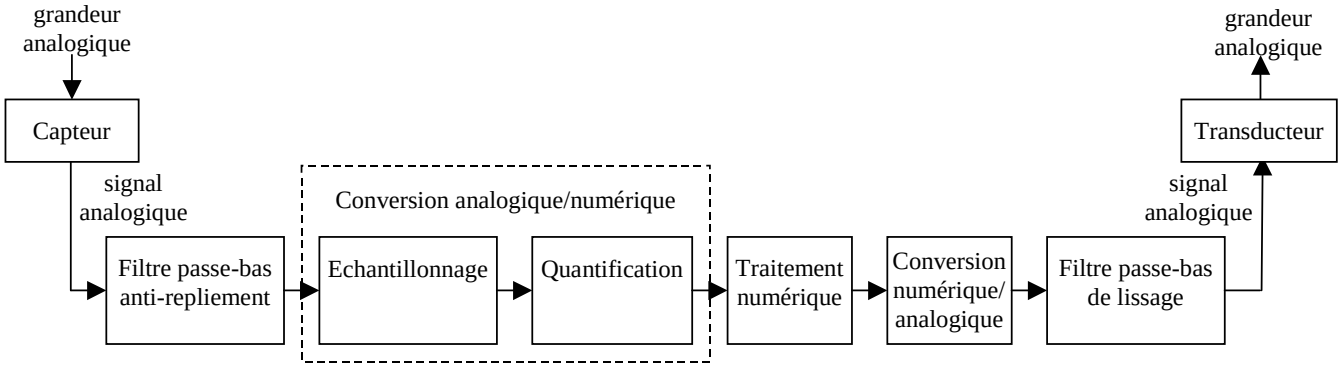
On ne peut réaliser en analogique que des traitements simples. Il faut passer en numérique pour bénéficier des capacités de traitement des ordinateurs même si on a réalisé des calculateurs analogiques dans les années 50-70.

Peut-on supprimer entièrement l'analogique ?

Non car le monde réel est analogique, et le signal analogique est l'**intermédiaire naturel** avec l'être humain. Un synthétiseur peut créer numériquement de la musique mais le signal sonore est forcément restitué par un haut-parleur sous forme d'ondes acoustiques analogiques.

1.1.2 Le numérique

Comment passe-t-on du monde des grandeurs analogiques au monde des grandeurs numériques ? On peut établir le schéma type d'une chaîne de traitement numérique.

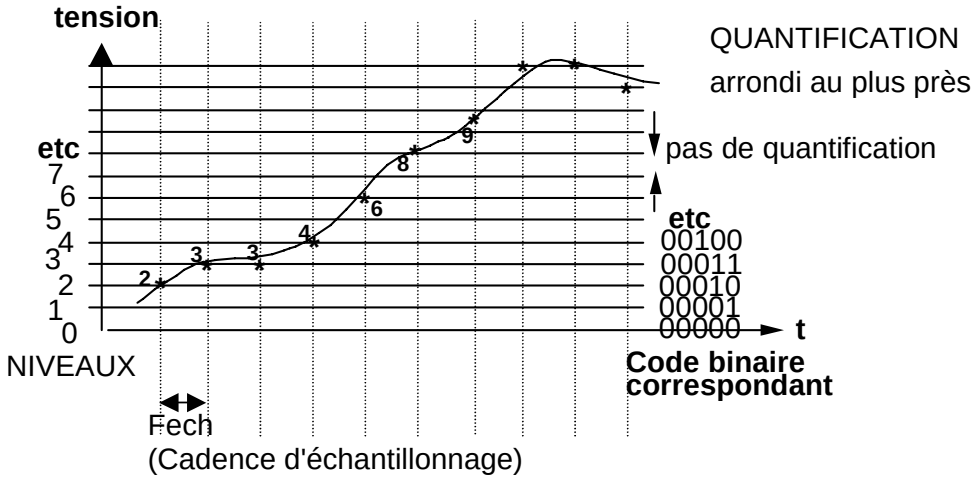


On trouve toujours dans une telle chaîne trois opérations principales :

- Le passage du signal analogique au signal numérique, c'est-à-dire le filtrage passe-bas anti-repliement, l'échantillonnage et la quantification. Les deux dernières étapes forment la conversion analogique/numérique.
- Le traitement numérique.
- Le passage du signal numérique au signal analogique, c'est-à-dire la conversion numérique/analogique et le filtrage passe-bas de lissage.

Etudions les trois opérations principales :

- ***L'échantillonnage*** du signal à une cadence régulière nommée Fech (fréquence ou cadence d'échantillonnage). Il faut que la fréquence d'échantillonnage soit suffisamment élevée pour pouvoir convertir fidèlement le signal. Ce signal est alors sous forme d'une suite d'échantillons, mais encore analogiques.



- ***La quantification :*** on partage l'étendue des variations du signal en N niveaux équidistants qui sont numérotés par exemple à partir de 0 : 0, 1, 2, 3, 4, …, N-2, N-1. L'intervalle entre chaque niveau se nomme le « pas de quantification ». On arrondit alors la valeur des échantillons précédents au niveau le plus proche (on commet donc une erreur de ± 1/2 du pas de quantification). Plus N est grand (plusieurs centaines ou plusieurs milliers), plus la précision sera élevée. En fait ceci n'est pas nouveau, c'est un peu finalement ce que l'on fait quand on mesure une longueur avec un mètre gradué et que l'on arrondit par exemple au millimètre, ou lorsque l'on donne l'heure à une minute près !

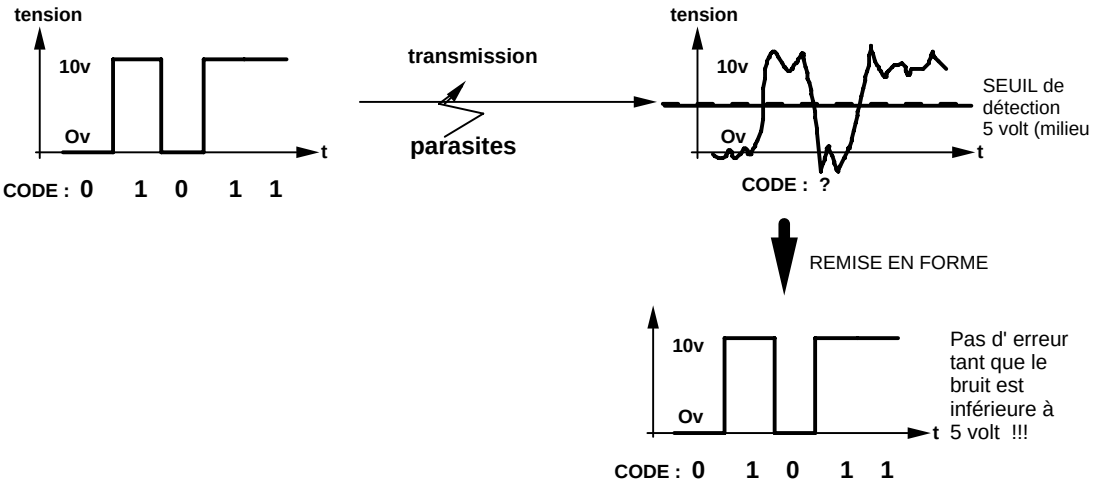
- ***Le passage en binaire :*** on fait correspondre à chaque niveau et valeur entière un CODE BINAIRE, écrit avec seulement des 0 et des 1. On nomme **bits ou éléments binaires** ces 0 et ces 1. Une valeur sera écrite avec un certain nombre de bits (8 bits pour des petits nombres, 16, 32, 64 bits ou plus pour des grands nombres ou des nombres fractionnaires). Exemple :

  Niveau 0 code 00000000
  Niveau 1 code 00000001
  Niveau 2 code 00000010
  …
  Niveau 255 code 11111111

La suite de ces 0 et 1 représente alors sous forme numérique le signal de départ. Exemple : avec 5 bits, nos échantillons 2, 3, 3, ... seront écrits 00010, 00011, 00011, … Un signal électrique peut facilement transporter cette suite de 0 et de 1, avec par exemple le 0 égal à 0 volt et le 1 égal à 10 volts. On peut facilement enregistrer ce signal binaire sur une bande magnétique, ou sur CDROM (par une suite de régions aimantées ou non, réfléchissantes ou non).

Quels sont les avantages du numérique ?

a) Protection contre les parasites lors d'une transmission.



Un seuil de détection permet une remise en forme, pratiquement sans perte d'information, sauf dans le cas de grosses perturbations. Il existe aussi des codes qui détectent et même corrigent les erreurs.

b) Mémorisation facile.

Il existe de nombreuses formes de mémoires à circuits intégrés (bascules, DRAM, Flash, …) pouvant stocker les deux états 0 et 1.

c) Stockage facile.

On enregistre le signal numérique sur bandes magnétiques, disquettes, disques durs, disques optiques, ...

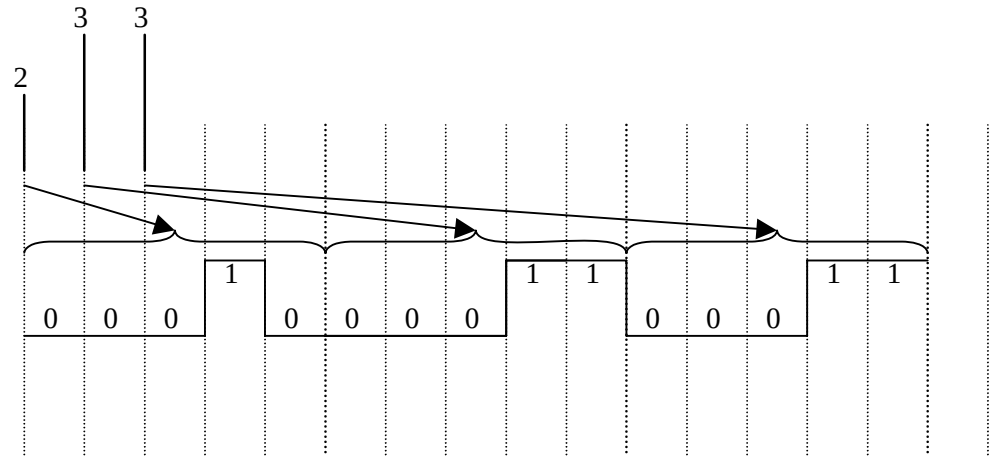
d) Capacité de traitement.

La capacité de traitement n’est limitée que par la puissance des ordinateurs. Par exemple, on sait faire aujourd’hui des films entièrement numériques avec des acteurs virtuels.

Quels sont les inconvénients du numérique ?

a) le temps de transmission sur un fil est plus élevé.

Soit à transmettre nos échantillons 2, 3, 3, … avec 5 bits par échantillon : la durée est alors multipliée par le nombre de bits.



Il faut donc accroître les vitesses de transmission (ce qui n'est pas un trop grand handicap vu les progrès des techniques numériques de nos jours).

b) Grand nombre de bits nécessaires pour le stockage en numérique.

Cela est surtout vrai si les grandeurs analogiques doivent être codées avec une bonne précision (sons et images de bonnes qualités). Heureusement, de gros progrès ont eu lieu en capacité de stockage (supports magnétiques ou optiques, mémoires des ordinateurs). De plus, des techniques de « compression » existent de nos jours comme MPEG.

Vu ses avantages qui dépassent largement ses inconvénients, le numérique est aujourd'hui partout :

- Dans les disques laser pour la haute qualité du son et la tenue dans le temps des informations.
- Dans les ordinateurs bien sûr.
- Dans les codes enregistrés sur les cartes de crédits, les cartes de transport, de parking, ...
- Dans la téléphonie mobile, la télé numérique, les radios FM RDS à synthétiseur de fréquence, …
- Dans les calculateurs de vols des avions, les automatismes de contrôle de tout processus industriel, …

Il y a en fait bien peu d'appareil non numérique aujourd'hui (même dans le matériel très grand public type cafetière, radio réveil, électroménager, jouet, …). Il n'y a guère que dans les hautes fréquences (au-delà de quelques dizaines de MHz) et dans l'électronique de puissance que l'analogique subsiste. L'électronique analogique est aussi **indispensable** pour

adapter le signal électrique du capteur à la conversion analogique/numérique ainsi que pour adapter le signal en sortie du convertisseur numérique/analogique aux caractéristiques du transducteur.

1.2 Introduction, variables et fonctions logiques

Un système numérique complexe est réalisé à partir d'un assemblage hiérarchique d'opérateurs logiques élémentaires réalisant des opérations simples sur des variables logiques. Ces variables logiques peuvent prendre les états : (**vrai, true**) ou bien (**faux, false,**). Vous connaissez déjà de nombreux systèmes physiques qui travaillent à partir de grandeurs ne pouvant prendre que deux états:

- interrupteur ouvert ou fermé,
- tension présente ou non,
- lampe allumée ou non, objet éclairé ou non,
- moteur en marche ou arrêté,
- affirmation vraie ou fausse,
- grandeur physique supérieure ou inférieure à un seuil fixé,
- grandeurs égales ou différentes.

Par convention, on associe généralement à l'état vrai d'une variable logique la valeur binaire 1 et la valeur 0 à l'état faux. C'est la logique positive. On peut aussi faire le contraire (c'est la logique négative) en associant la valeur 0 à l'état vrai et la valeur 1 à l'état faux. Dans ce cours, on travaillera toujours en logique positive, sauf mention contraire.

Electriquement, l'état vrai (ou la valeur 1) va être associé à un niveau de tension haut (la tension d'alimentation du montage V_{DD} en général) et l'état faux (valeur 0) va être associé à un niveau de tension bas (en général, la masse du montage GND). Les composants élémentaires du montage sont des paires de transistors MOS (logique CMOS) qui peuvent être vus comme des interrupteurs ouverts ou fermés.

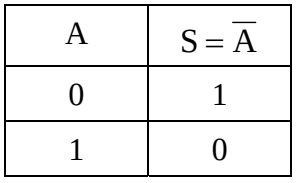
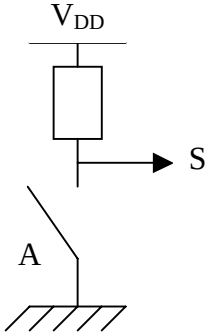
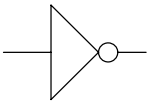
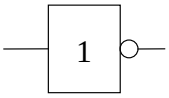
En logique, tout raisonnement est décomposé en une suite de propositions élémentaires qui sont vraies ou fausses. BOOLE, mathématicien britannique du XIX siècle a créé une algèbre qui codifie les règles (algèbre Booléenne) à l'aide de variables logiques ne pouvant prendre que deux états et d'opérations élémentaires portant sur une ou 2 variables.

L'algèbre de BOOLE ne traite que de la logique combinatoire, c'est à dire des circuits numériques dont la sortie ne dépend que de l'état présent des entrées (sans mémoire des états passés). A chaque opérateur logique booléen (NON, ET, OU, NON ET, NON OU, OU exclusif, NON OU exclusif), on va associer un circuit numérique combinatoire élémentaire. La logique séquentielle (avec mémoire) sera vue au chapitre suivant.

1.2.1 Les opérateurs fondamentaux

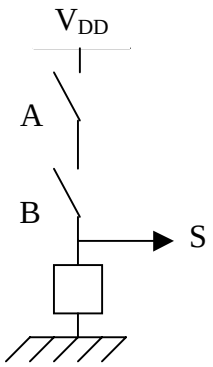
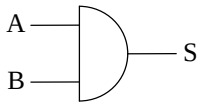
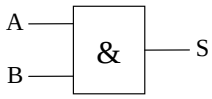
1.2.1.1 INV (NON)

L'opérateur d'inversion ne porte que sur une seule variable d'entrée. Si A est la variable d'entrée, S la variable de sortie vaut : $S = \overline{A}$ (on prononce A barre). Le tableau suivant résume l'action de cet opérateur. Dans ce chapitre, l'interrupteur ouvert vaut 0 et l'interrupteur fermé vaut 1.

Table de vérité	Montage	Symbole traditionnel	Symbole normalisé
			

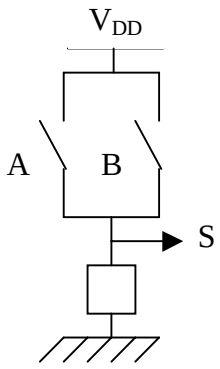
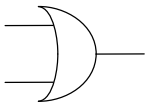
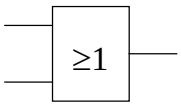
1.2.1.2 AND (ET)

L'opérateur AND (ET) porte sur deux variables d'entrée. Si A et B sont les variables d'entrée, alors S = A.B. S est vraie si A **ET** B sont vraies. L'opérateur AND est symbolisé par le point (**.**) comme la multiplication en mathématique (c'est d'ailleurs l'opération réalisée en binaire). On peut aussi voir cette fonction comme l'opérateur minimum (min) qui prend la plus petite des deux valeurs. Le tableau suivant résume l'action de cet opérateur.

Table de vérité	Montage	Symbole traditionnel	Symbole normalisé
<table><tr><th>A</th><th>B</th><th>S = A.B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>			

1.2.1.3 OR (OU)

L'opérateur OR (OU) porte sur deux variables d'entrée. Si A et B sont les variables d'entrée, alors S = A+B. S est vraie si A **OU** B sont vraies. L'opérateur OR est symbolisé par le plus (+) comme l'addition en mathématique. On peut voir cette fonction comme l'opérateur maximum (max) qui prend la plus grande des deux valeurs. Le tableau suivant résume l'action de cet opérateur.

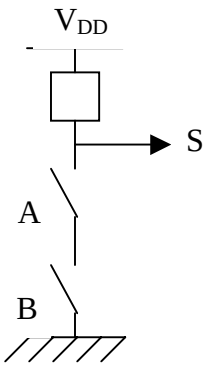
Table de vérité	Montage	Symbole traditionnel	Symbole normalisé
<table><tr><th>A</th><th>B</th><th>S = A+B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>			

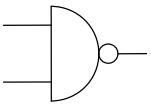
1.2.1.4 NAND (NON ET)

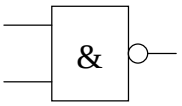
L'opérateur NAND (NON ET) porte sur deux variables d'entrée. Si A et B sont les variables d'entrée, alors $S = \overline{A.B}$. S est **fausse** si A **ET** B sont vraies. L'opérateur NAND est l'inverse de l'opérateur AND. Son symbole est le symbole du ET suivi d'une bulle qui matérialise l'inversion. Le tableau suivant résume l'action de cet opérateur.

Table de vérité	Montage	Symbole traditionnel	Symbole normalisé

A	B	$S = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0





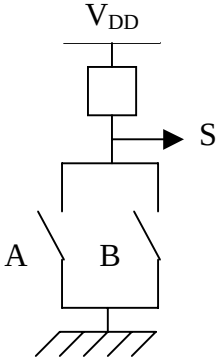


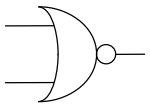
1.2.1.5 NOR (NON OU)

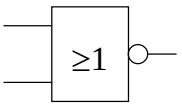
L'opérateur NOR (NON OU) porte sur deux variables d'entrée. Si A et B sont les variables d'entrée, alors $S = \overline{A + B}$. S est **fausse** si A **OU** B sont vraies. L'opérateur NOR est l'inverse de l'opérateur OR. Son symbole est le symbole du OU suivi d'une bulle qui matérialise l'inversion. Le tableau suivant résume l'action de cet opérateur.

Table de vérité	Montage	Symbole traditionnel	Symbole normalisé

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

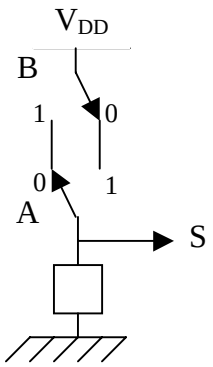
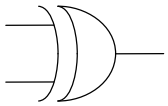
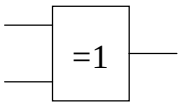






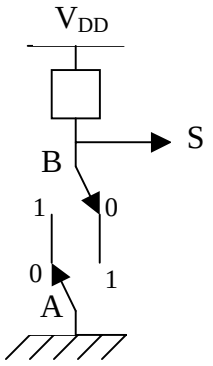
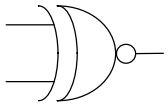
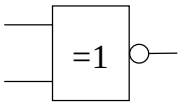
1.2.1.6 XOR (OU exclusif)

L'opérateur XOR (OU exclusif) n'est pas un opérateur de base car il peut être réalisé à l'aide des portes précédentes. Il porte sur deux variables d'entrée. Si A et B sont les variables d'entrée, alors $S = A \oplus B = A.\overline{B} + \overline{A}.B$. S est vraie si A est **différent de** B. L'opérateur XOR est symbolisé par un + entouré d'un cercle ($\oplus$) car il réalise l'addition en binaire, mais modulo 2. Le OU normal (inclusif) est vrai quand A et B sont vrai (1+1=1). Le OU exclusif exclut ce cas (d'où son nom). Le tableau suivant résume l'action de cet opérateur.

Table de vérité	Montage	Symbole traditionnel	Symbole normalisé
<table><tr><th>A</th><th>B</th><th>S = A ⊕ B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>			

1.2.1.7 XNOR (NON OU exclusif)

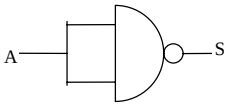
L'opérateur XNOR (NON OU exclusif) n'est pas non plus un opérateur de base. Il porte sur deux variables d'entrée. Si A et B sont les variables d'entrée, alors $S = \overline{A \oplus B} = \overline{A}.\overline{B} + A.B$. S est vraie si A **égale** B. L'opérateur XNOR est l'inverse de l'opérateur XOR. Son symbole est le symbole du XOR suivi d'une bulle qui matérialise l'inversion. Le tableau suivant résume l'action de cet opérateur.

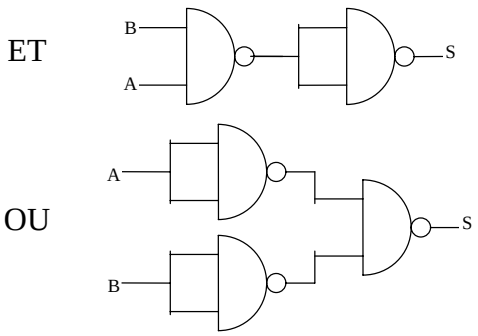
Table de vérité	Montage	Symbole traditionnel	Symbole normalisé
<table><tr><th>A</th><th>B</th><th>S = $\overline{A \oplus B}$</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>			

1.2.1.8 Portes universelles

Les NAND et les NOR sont des portes universelles car elles permettent de réaliser toutes les opérations logiques élémentaires. Par exemple avec des NAND, on peut réaliser les opérations :

inversion

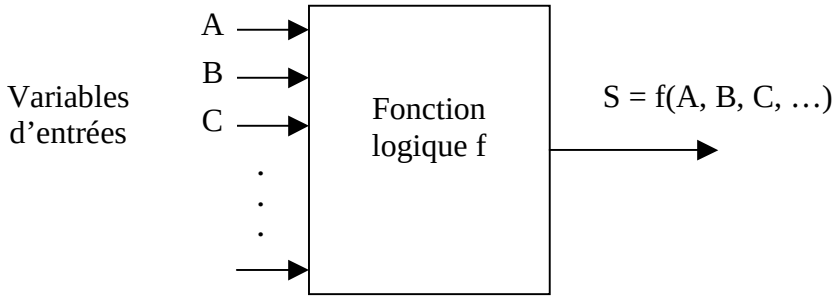




Le même genre de montage peut être réalisé avec des portes NOR. On verra plus tard que les portes NAND et NOR demandent le minimum de transistors pour être fabriquées et sont les plus rapides.

1.2.2 Algèbre de BOOLE

L'algèbre de BOOLE porte sur des variables logiques (qui ne peuvent prendre que deux états, vrai ou faux). Elle possède trois opérateurs booléens : NOT (NON), AND (ET), OR (OU). L'algèbre de BOOLE permet de réaliser des fonctions à l'aide de variables booléennes et des trois opérateurs de base. Le résultat obtenu est booléen, c'est-à-dire vrai ou faux.



Les lois fondamentales de l'algèbre de BOOLE se vérifient en écrivant les tables de vérités et en testant tous les cas possibles. Ces lois sont les suivantes :

Commutativité	A.B = B.A A+B = B+A
Associativité	A.(B.C) = (A.B).C A+(B+C) = (A+B)+C
Distributivité	ET sur OU : A.(B+C) = (A.B) + (A.C) OU sur ET : A+(B.C) = (A+B) . (A+C)

	ET sur ET : A.(B.C) = (A.B) . (A.C) OU sur OU : A+(B+C) = (A+B) + (A+C)
Idempotence	A.A = A A+A = A
Complémentarité	$A.\overline{A} = 0$ $A + \overline{A} = 1$
Identités remarquables	1.A = A 1+A=1 0.A = 0 0+A = A

A partir de ces propriétés, on démontre les relations de base suivantes :

$A.B + A.\overline{B} = A$ $(A + B).(A + \overline{B}) = A$
$A + A.B = A$ $A.(A + B) = A$
$A + \overline{A}.B = A + B$ $A.(\overline{A} + B) = A.B$

Le OU exclusif a des propriétés particulières. Il possède les propriétés de commutativité et d'associativité, mais n'est pas distributif par rapport au ET ou au OU. On a vu que :

$$A \oplus B = \overline{A}.B + A.\overline{B}$$

On a de plus

$$\overline{A \oplus B} = \overline{A} \oplus B = A \oplus \overline{B}$$

et

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

Le théorème de DE MORGAN complète les propriétés de l'algèbre de BOOLE. Il est indépendant du nombre de variables. Il s'énonce des deux manières suivantes :

1) La négation d'un produit de variables est égale à la somme des négations des variables. Par exemple :

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

$$\overline{(\overline{A+B}).(A.\overline{B})} = \overline{(\overline{A+B})} + \overline{(A.\overline{B})}$$

2) La négation d'une somme de variables est égale au produit des négations des variables. Par exemple :

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

$$\overline{(\overline{A+B})+(A.\overline{B})} = \overline{(\overline{A+B})}.\overline{(A.\overline{B})}$$

D'une manière plus générale, on peut dire que $\overline{f(A,B,C,etc,+,.)} = f(\overline{A},\overline{B},\overline{C},etc,.,+)$

1.2.3 Expression d'une fonction logique

Il existe deux méthodes pour exprimer une fonction logique : soit donner directement son équation logique (par exemple, $S = A + \overline{C} + \overline{A.B}.C$), soit utiliser une table de vérité. A un nombre fini N de variables d'entrée correspond 2^N combinaisons possibles. La table de vérité va donc indiquer la valeur de la fonction pour les 2^N valeurs possibles. Si par exemple S est une fonction de trois variables, il y aura 2^3 soit 8 combinaisons possibles. On va placer les trois variables dans un ordre arbitraire (à ne pas modifier ensuite !) A, B, C de gauche à droite par exemple et écrire les combinaisons dans l'ordre des entiers naturels (0, 1, 2, 3, …, 2^N-1). La table de vérité de S sera par exemple la suivante :

entrées	Sortie

Valeur entière	A	B	C	combinaison	S
0	0	0	0	$\overline{A}.\overline{B}.\overline{C}$	0
1	0	0	1	$\overline{A}.\overline{B}.C$	1
2	0	1	0	$\overline{A}.B.\overline{C}$	1
3	0	1	1	$\overline{A}.B.C$	1
4	1	0	0	$A.\overline{B}.\overline{C}$	0
5	1	0	1	$A.\overline{B}.C$	1
6	1	1	0	$A.B.\overline{C}$	0
7	1	1	1	A.B.C	0

Sous sa forme complète, l'équation logique de S se lit directement. C'est la somme du ET logique de chaque combinaison avec l'état de S correspondant. Ici, on obtient la <u>forme canonique complète</u> :

$$S = \overline{A}.\overline{B}.\overline{C}.0 + \overline{A}.\overline{B}.C.1 + \overline{A}.B.\overline{C}.1 + \overline{A}.B.C.1 + A.\overline{B}.\overline{C}.0 + A.\overline{B}.C.1 + A.B.\overline{C}.0 + A.B.C.0$$

On sait que 0.X = 0, donc on peut éliminer les termes qui valent 0. Cela nous donne :

$$S = \overline{A}.\overline{B}.C.1 + \overline{A}.B.\overline{C}.1 + \overline{A}.B.C.1 + A.\overline{B}.C.1$$

On sait aussi que 1.X = X, donc on peut écrire la <u>forme canonique abrégée</u> :

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

C'est simplement la somme de combinaisons pour lesquelles S vaut 1. On la note aussi $\sum(1,2,3,5)$.

Nous avons maintenant un ensemble de règles algébriques et nous savons exprimer une fonction logique. La question importante est : comment être sur que l'équation logique de la fonction est simplifiée au maximum, c'est-à-dire qu'elle va utiliser un nombre de portes minimum ?

1.2.4 Simplification des fonctions logiques

L'expression d'une fonction logique sous sa forme la plus simple n'est pas quelque chose d'évident. Trois méthodes sont utilisables :

- Le raisonnement. On cherche, à partir du problème à résoudre, l'expression la plus simple possible. Evidemment, cette méthode ne garantit pas un résultat optimal.
- La table de vérité et les propriétés de l'algèbre de BOOLE. C'est plus efficace, mais il est facile de rater une simplification, notamment quand la fonction est compliquée.
- La méthode graphique des tableaux de Karnaugh. C'est la méthode la plus efficace car elle garantit le bon résultat. Cette méthode est utilisée sous forme informatique dans tous les outils de CAO.

1.2.4.1 Simplification algébrique

Prenons un exemple. On a deux voyants A et B. On veut déclencher une alarme quand au moins un des deux voyants est allumé, c'est-à-dire : soit A allumé avec B éteint, soit B allumé avec A éteint, soit A et B allumés. Par le raisonnement, le lecteur attentif aura trouvé que S = A + B. Ecrivons la table de vérité :

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

On obtient donc :

$$S = \overline{A}.B + A\overline{B} + AB$$

D'où on tire (d'après les propriétés de l'algèbre de BOOLE) :

$$S = A(\overline{B} + B) + B\overline{A} = A.1 + B\overline{A} = A + B\overline{A} = A + B$$

On voit bien que cette méthode de simplification devient peu évidente quand la fonction est plus complexe. Voyons maintenant une méthode plus efficace.

1.2.4.2 Simplification par les tableaux de Karnaugh

Cette méthode permet :

- d'avoir pratiquement l'expression logique la plus simple pour une fonction F.
- de voir si la fonction $\overline{F}$ n'est pas plus simple (cela arrive).
- de trouver des termes communs pour un système à plusieurs sorties, dans le but de limiter le nombre de portes.
- de tenir compte de combinaisons de variables d'entrées qui ne sont jamais utilisées. On peut alors mettre 0 ou 1 en sortie afin d'obtenir l'écriture la plus simple.
- de voir un OU exclusif caché.

A la main, on traite 4 variables sans difficultés. Au maximum, on peut aller jusqu'à 5 voir 6 variables. En pratique, on utilise un programme informatique qui implémente généralement l'algorithme de QUINE-McCLUSKEY, qui reprend la méthode de Karnaugh, mais de manière systématique et non visuelle.

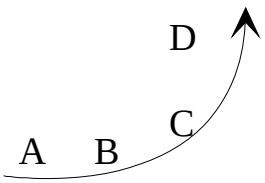
Définition : dans un code adjacent, seul un bit change d'une valeur à la valeur suivante. Exemple avec deux variables :

Code binaire naturel		Code GRAY	
A	B	A	B
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Les tableaux de Karnaugh sont une variante des tables de vérité. Ils sont organisés de telle façon que les termes à 1 (ou à 0) adjacents soient systématiquement regroupés dans des cases voisines, donc faciles à identifier visuellement. En effet, deux termes adjacents (qui ne différent que par une variable) peuvent se simplifier facilement : $A.B.C + A.B.\overline{C}$ = $AB(C+\overline{C})$ = AB. On représente les valeurs de la fonction dans un tableau aussi carré que possible, dans lequel chaque ligne et chaque colonne correspondent à une combinaison des variables d'entrée exprimée avec un code adjacent (le code GRAY en général).

Les traits gras correspondent à une zone où la variable vaut 1. Avec de l'habitude, il est inutile d'écrire les valeurs des variables, ces traits gras suffisent à la lecture.

Pour une écriture rapide à partir de la table de vérité, ou directement à partir du cahier des charges, ou à partir d'une équation déjà existante, on peut écrire les variables A, B, C, D dans l'ordre ci contre. Cela permet une lecture systématique.



La disposition des 1 et 0 dépend de l'ordre des variables et du code adjacent choisi, mais quelque soit l'ordre, l'équation finale obtenue reste la même.

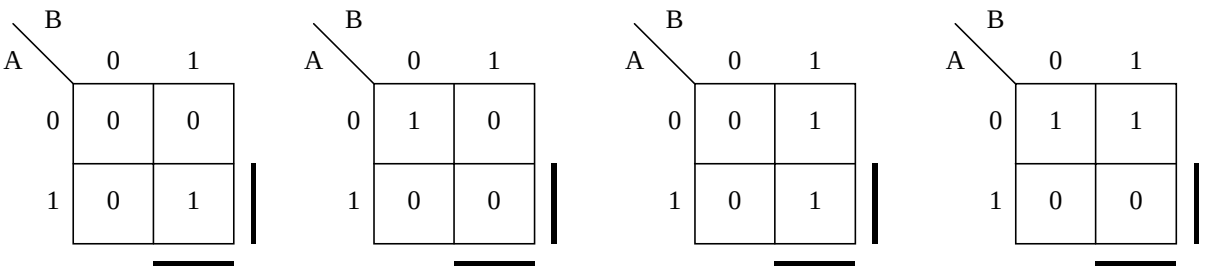
Pour la lecture et la simplification de l'expression, on cherche des paquets les plus gros possibles (de 1, 2, 4 ou 8 variables), en se rappelant que le code est aussi adjacent sur les bords (bord supérieur avec bord inférieur, bord gauche avec bord droit). On effectue une lecture par « intersection » en recherchant la ou les variables ne changeant pas pour le paquet. On ajoute alors les paquets. On obtient l'expression sous la forme d'une somme de produit. Une case peut être reprise dans plusieurs paquets.

<u>Exemple 1 : deux variables d'entrée A et B</u>

1 case = produit de 2 variables

2 cases = 1 variable

4 cases = 1 ou 0



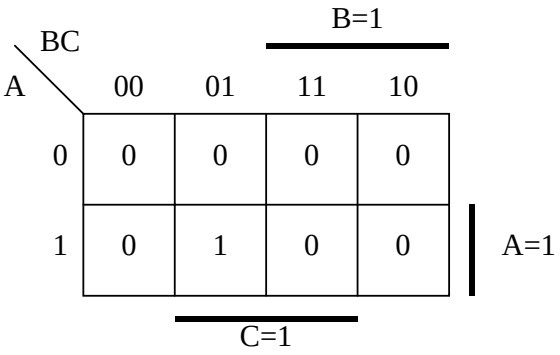
<u>Exemple 2 : trois variables d'entrée A, B et C</u>

1 case = produit de 3 variables

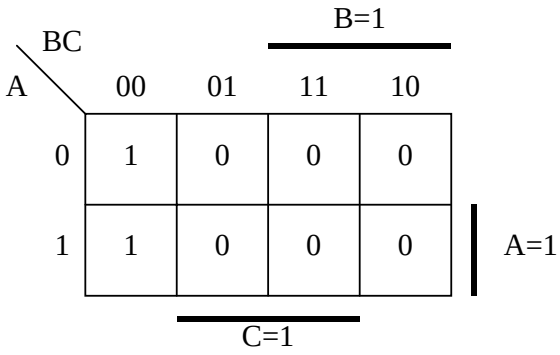
2 cases = produit de 2 variables

4 cases = 1 variable

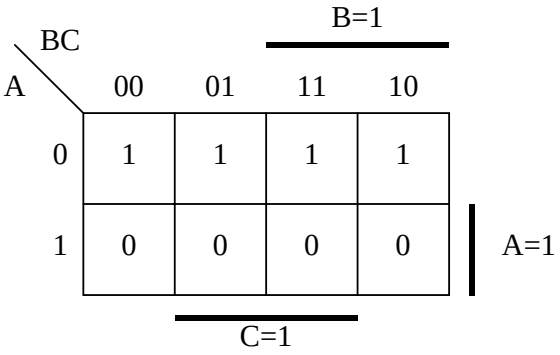
8 cases = 1 ou 0



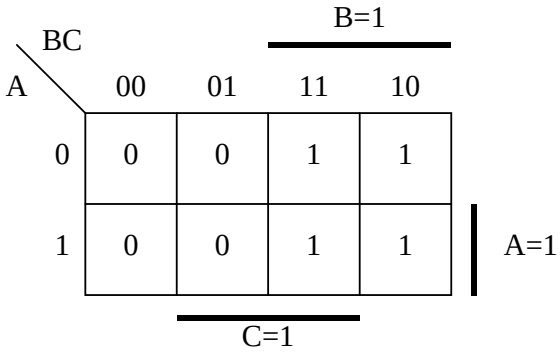
$S = A.\overline{B}.C$



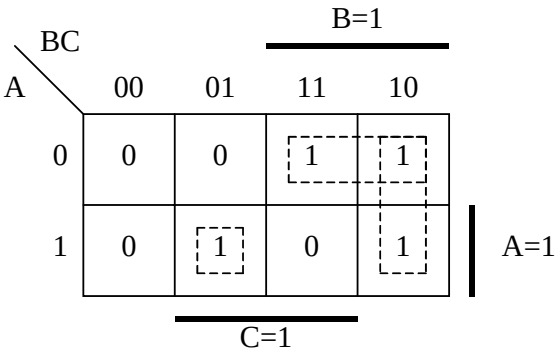
$S = \overline{B}.\overline{C}$



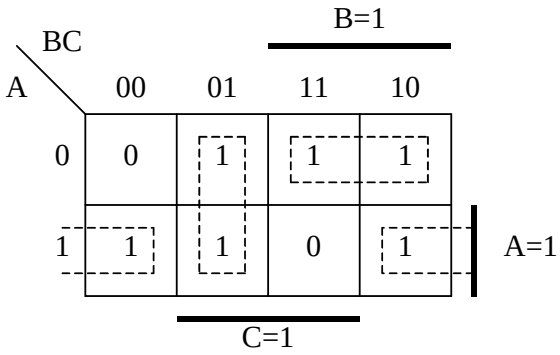
$S = \overline{A}$



S = B



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



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

Il y a parfois plusieurs solutions équivalentes.

Exemple 3 : quatre variables d'entrée A, B, C et D

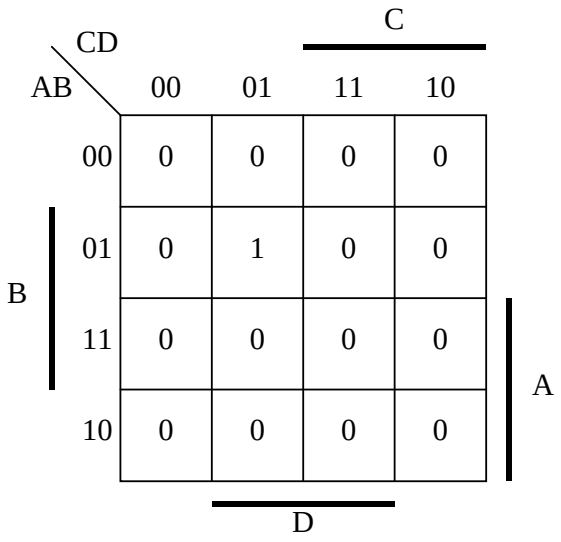
1 case = produit de 4 variables

2 cases = produit de 3 variables

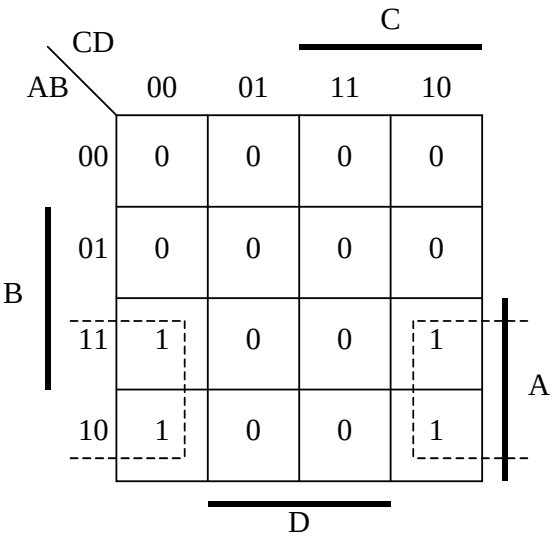
4 cases = produit de 2 variables

8 cases = 1 variable

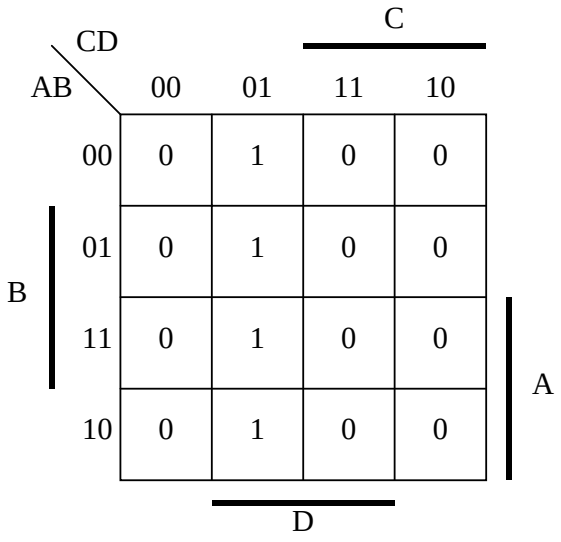
16 cases = 1 ou 0



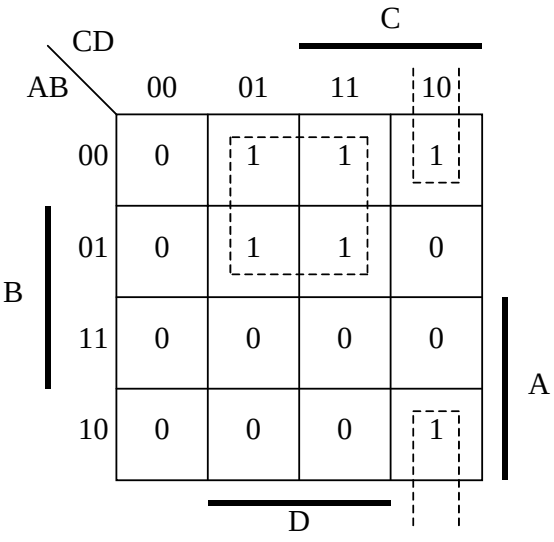
$S = \overline{A}.B.\overline{C}.D$



$S = A.\overline{D}$

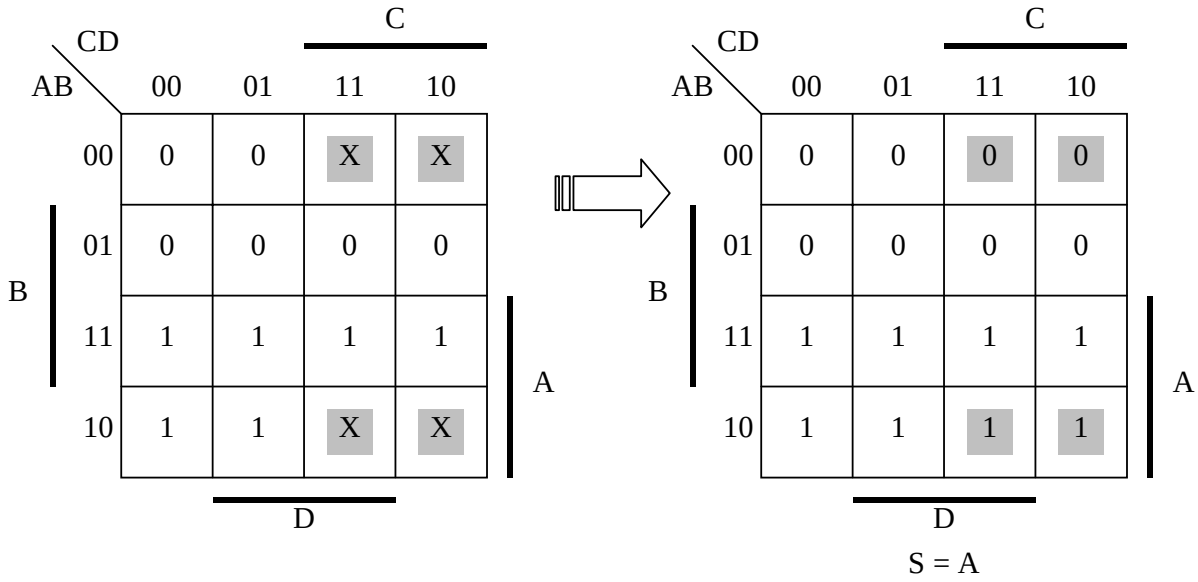


$S = \overline{C}.D$

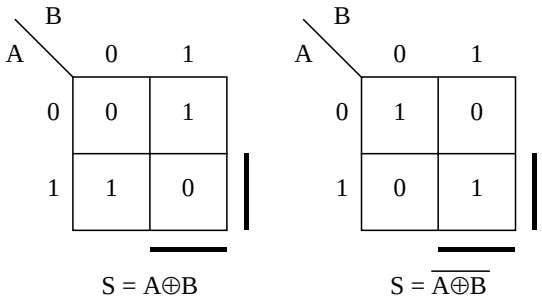


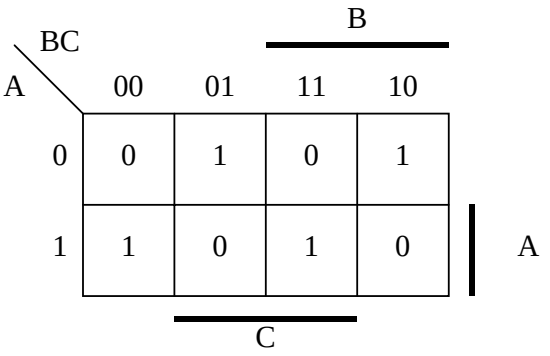
$S = \overline{A}.D+\overline{B}.C.\overline{D}$

Dans une fonction logique, il peut exister des états non utilisés. Ces combinaisons n'ont pas d'importance pour le problème à résoudre. On les appelle en anglais des états « don't care » (ne pas tenir compte) et on les symbolise par un X. Ces états X peuvent prendre la valeur la plus pratique pour simplifier la fonction. Exemple :

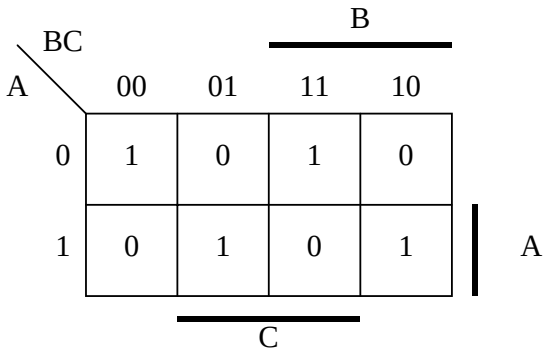


Il existe un cas particulier de fonction qu'il est parfois difficile à détecter, le OU exclusif (XOR) ou son complément. Il faut repérer des 1 disposés en quinconce. Pour voir s'il s'agit du XOR ou de son complément, il suffit alors de regarder la case « toutes entrées à 0 ». Si elle vaut 0, c'est un XOR, sinon c'est un XNOR.

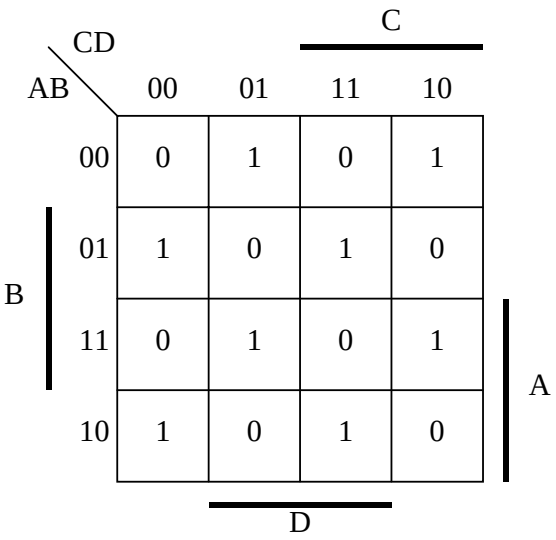




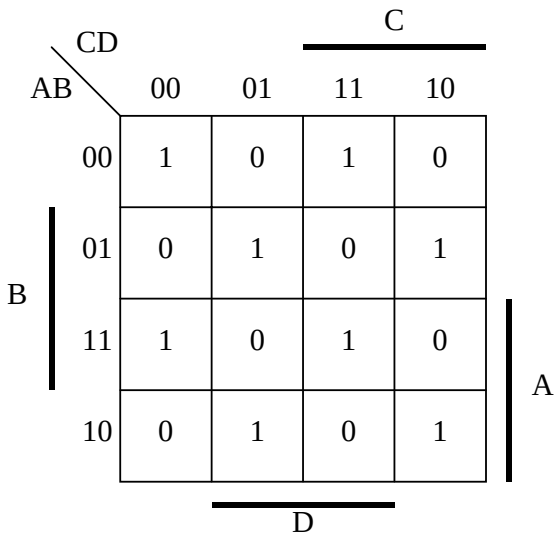
S = A⊕B⊕C



S = $\overline{A \oplus B \oplus C}$

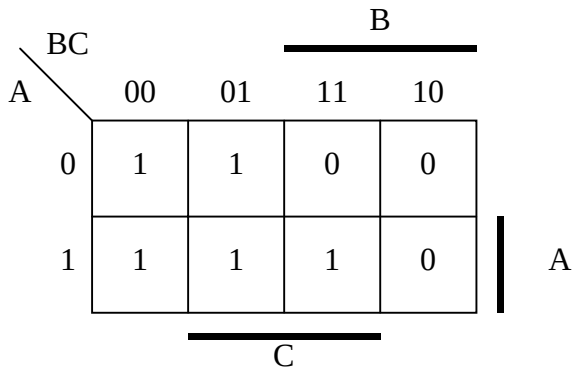


S = A⊕B⊕C⊕D



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

Il peut parfois être plus pratique de lire les 0 dans le tableau plutôt que les 1. On obtient alors $\overline{F}$ = somme de produits ce qui implique (par DE MORGAN) que F = produit de somme des variables inversées. Si par exemple on a :



En comptant les 1 : S = $\overline{B}$ + A.C

En comptant les 0 au lieu des 1, on obtient : $\overline{S} = B.\overline{C} + \overline{A}.B$. D'où on tire : $\overline{\overline{S}} = S = \overline{B.\overline{C} + \overline{A}.B}$ ce qui donne finalement : $S = (\overline{B} + C).(A + \overline{B})$. On peut utiliser la lecture directe sur les 0 quand il y a peu de 0 et beaucoup de 1.

1.3 Représentation des nombres : les codes

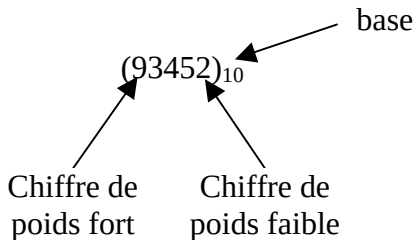
On regroupe souvent les bits par paquets afin de coder des nombres entiers, réels ou bien des caractères. Pour cela, un groupe de bits doit former un code. Il existe de très nombreux codes et nous allons voir maintenant les principaux.

1.3.1 Entiers naturels

1.3.1.1 Base d'un système de numération

Une base est constituée par l'ensemble des chiffres (ou caractères) différents qu'utilise le système de numération. Tout nombre peut se décomposer en fonction des puissances entières, positives ou négatives, de la base de numération. Il peut donc se mettre sous forme polynomiale. Prenons l'exemple de la base 10.

Dans la base décimale, un chiffre peut prendre 10 valeurs de 0 à 9. Un nombre est un polynôme en puissance de 10, 10^n avec n positif, nul ou négatif. Par exemple :



La forme polynomiale est :

$(93452)_{10}$ =	$9.10^4 +$	$3.10^3 +$	$4.10^2 +$	$5.10^1 +$	2.10^0
rang	4	3	2	1	0

Un autre exemple avec un nombre réel :

$(23,64)_{10}$ =	$2.10^1 +$	$3.10^0 +$	$6.10^{-1} +$	4.10^{-2}
rang	1	0	-1	-2

Le rang du chiffre d'un nombre représenté dans une base est l'exposant de la base associé à ce chiffre, dans la représentation polynomiale considérée.

Soit d'une manière générale, une base b contenant b caractères a_i tels que $a_i \in (0, 1, 2, \ldots, (b-1))$. La représentation polynomiale d'un nombre entier N est :

$N = \sum_{i=0}^{n} a_i.b^i$, n étant le rang du chiffre de poids fort.

Parmi toutes les bases possibles, 2 sont particulièrement intéressantes, les bases 2 et 16. La base 2 permet de traduire sous forme numérique l'état des variables booléennes (vrai = 1 / faux = 0 en logique positive). Les a_i peuvent donc valoir (0, 1) et la représentation polynomiale est :

$(10110)_2$ =	$1.2^4 +$	$0.2^3 +$	$1.2^2 +$	$1.2^1 +$	0.2^0
rang	4	3	2	1	0

Le bit le plus à gauche est le bit de poids fort (**MSB** : Most Significant Bit), le bit le plus à droite est le bit de poids faible (**LSB** : Least Significant Bit). Un chiffre en base 16 peut prendre 16 valeurs allant de 0 à 15. Il peut être codé avec 4 bits. Comme les chiffres s'arrêtent à 9 en décimal, on utilise les lettres a, b, c, d ,e et f pour représenter les derniers états.

décimal	binaire	hexadécimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8

9	1001	9
10	1010	a
11	1011	b
12	1100	c
13	1101	d
14	1110	e
15	1111	f

La représentation polynomiale en base hexadécimale (base 16) est :

$(a0c25)_{16} =$	$a.16^4 +$	$0.16^3 +$	$c.16^2 +$	$2.16^1 +$	5.16^0
rang	4	3	2	1	0

1.3.1.2 Changement de base

Une question importante est le passage d'une base à l'autre : le changement de base. Soit $(N)_{10}$, combien vaut $(N)_2$ ou $(N)_{16}$.

La conversion vers la base 10 est la plus simple. Il suffit de calculer la valeur du polynôme. En reprenant les exemples précédents, il est évident que (au moins avec une calculatrice) :

$$(10110)_2 = 1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0 = (22)_{10}$$

$$(a0c25)_{16} = a.16^4 + 0.16^3 + c.16^2 + 2.16^1 + 5.16^0 = (658469)_{10}$$

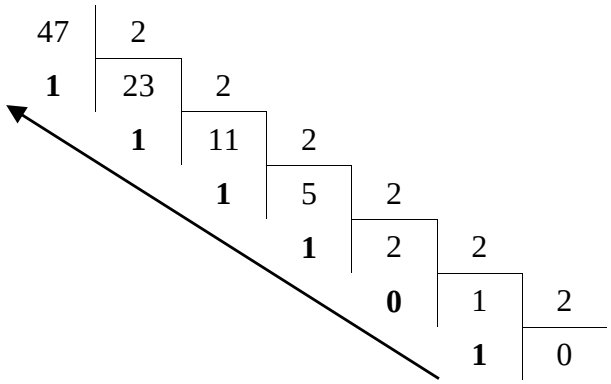
Remarque : il existe en binaire un regroupement très utile, l'**octet** (ou byte) qui est un groupe de 8 bits. Par exemple l'octet :

$$(10000001)_2 = 1*2^7 + 1*2^0 = (129)_{10}$$

La valeur d'un octet est comprise entre 0 et $(255)_{10}$. Pour raccourcir l'écriture, on utilise souvent la notation hexadécimale (base 16) en regroupant les bits par paquets de 4. Par exemple l'octet :

$$(10000001)_2 = (81)_{16}$$

Le changement de base décimal $\rightarrow$ binaire est moins évident. Il faut diviser le nombre à convertir par 2. On conserve le reste (qui vaut 0 ou 1) et on répète la division sur le quotient, jusqu'à ce que le quotient vaille 0. On écrit alors tous les restes à la suite, le premier reste obtenu étant le poids faible. Exemple :



On obtient :

$$(47)_{10} = (101111)_2$$

Le passage d'hexadécimal vers binaire et de binaire vers hexadécimal est plus simple. Dans le premier cas, il suffit de remplacer chaque chiffre hexadécimal par le groupe de 4 bits correspondant (voir tableau de conversion). Exemple :

$$(ab85)_{16} = (1010\ \ 1011\ \ 1000\ \ 0101)_2$$

Dans le deuxième cas, il suffit de regrouper les bits par paquet de 4 en partant du poids faible. Il faudra éventuellement rajouter des bits à 0 à gauche pour terminer le groupe de 4 bits de poids fort. Exemple :

$$(101001000010101111)_2 = (\mathbf{00}10\ \ 1001\ \ 0000\ \ 1010\ \ 1111)_2 = (290af)_{16}$$

Pour passer d'hexadécimal en décimal (et vice versa), vous pouvez passer par l'intermédiaire du binaire ou faire le calcul directement par la méthode des divisions successives.

1.3.2 Entiers signés

La méthode naturelle pour coder un nombre avec signe est d'utiliser la méthode traditionnelle de la base 10 : on indique le signe suivi de la valeur absolue du nombre. Ce type de code est connu sous le nom de « signe-valeur absolue ». En binaire, cela signifie que pour un nombre codé sur N bits, le bit de poids fort sert à coder le signe (0 = +, 1 = -) et les N-1 bits restant codent la valeur absolue du nombre. Par exemple :

$$(-16)_{10} = (1\ 10000)_2$$

Dans un tel code, la plage des valeurs que l'on peut coder (la dynamique) est :

$2^{N-1}-1$	0 11…11
…	…
2	0 00…10
1	0 00…01
0	0 00…00
-0	1 00…00
-1	1 00…01
-2	1 00…10
…	…
$-(2^{N-1}-1)$	1 11…11

Cette méthode simple pour l'être humain est plutôt compliquée à utiliser dans un ordinateur (ou un circuit numérique) car pour additionner ou soustraire deux nombres, il faut commencer par déterminer quel sera le signe du résultat ce qui oblige à commencer tout calcul par une comparaison qui fait intervenir les signes et les valeurs absolues des deux nombres. C'est la raison pour laquelle on utilisera plutôt le code complément à 2.

Le code complément à 2 est le code utilisé pour représenter les nombres entiers dans un ordinateur. Il n'est pas très parlant pour un être humain, mais il présente l'intérêt majeur de permettre une arithmétique simple (notamment, il n'y a plus de soustraction). La construction du code sur n bits découle directement de la définition modulo 2^n des nombres binaires. Etant donné un nombre A :

- Si $A \geq 0$, le code de A est l'écriture en binaire naturel de A, éventuellement complété à gauche par des 0. Par exemple, $A = (23)_{10}$, codé sur 8 bits s'écrit : $(00010111)_{ca2}$.
- Si $A < 0$, le code de A est l'écriture en binaire naturel de $2^n - |A|$ ou ce qui revient au même $2^n + A$. Par exemple, $A = (-23)_{10}$, codé sur 8 bits s'écrit : $(11101001)_{ca2}$ qui est la représentation en binaire naturel de $(256)_{10} - (23)_{10} = (233)_{10}$.

On remarquera que le bit de poids fort indique le signe (0 = +, 1 = -). Le calcul de l'opposé d'un nombre codé sur n bits est toujours : $-A = 2^n - A$ modulo 2^n. Par exemple -(-23) = 256 + 23 modulo 256 = 23. Il existe une astuce de calcul pour obtenir l'expression binaire de l'inverse d'un nombre dont on connaît le code :

- On remarque que $2^n - A = 2^n - 1 - A + 1$.
- On sait que $2^n - 1$ est le nombre dont tous les chiffres valent 1.
- On en déduit que $2^n - 1 - A$ est le nombre que l'on obtient en remplaçant dans le code de A les 1 par des 0 et réciproquement. Ce nouveau nombre s'appelle le complément à 1 de A et se note A_{ca1} ou $\overline{A}$. On en déduit finalement que :

$$-A = A_{ca1} + 1$$

Par exemple :

$$(23)_{10} = (00010111)_2 = (11101000)_{ca1}$$
$$-23 = (11101000)_{ca1} + 1 = (11101001)_{ca2}$$

Dans ce code, la dynamique est :

$2^{N-1}-1$	0 11…11
…	…

2	0 00…10
1	0 00…01
0	0 00…00
-1	1 11…11
-2	1 11…10
…	…
$-(2^{N-1})$	1 00…00

Un des avantages de ce code est que la soustraction n'existe plus. Au lieu de soustraire un nombre, il n'y qu'à additionner l'inverse de ce nombre. Par exemple, au lieu de calculer 50 – 23, on va pouvoir calculer 50 + (-23). Cela paraît bête, mais cela élimine la nécessité d'utiliser un soustracteur binaire. On n'utilise en conception logique que des additionneurs.

Un autre avantage du code complément à 2 est que si on effectue des additions successives, tant que le résultat final est dans la dynamique autorisée (pas de débordement du résultat final), on peut ignorer les débordements éventuels des résultats intermédiaires de calcul. Par exemple, on code avec 3 bits (dynamique de -4 à +3) et on additionne 2 + 2 - 3 ce qui doit donner 1 en base 10. Lors du premier calcul, 2 + 2 = 4, il y a un débordement mais le résultat final est quand même bon. En effet, $(010)_{CA2} + (010)_{CA2} = (100)_{CA2}$ qui vaut -4 (donc débordement) mais $(100)_{CA2} + (101)_{CA2} = (001)_{CA2}$ ce qui donne bien 1. Cela paraît magique, mais en fait, c'est une conséquence directe de l'arithmétique modulo 2^n.

Il y a malgré tout des inconvénients au code complément à 2 en ce qui concerne les comparaisons, les multiplications et les divisions, mais ses avantages l'emportent largement sur ses inconvénients.

1.3.3 Nombres réels (flottants)

Les nombres flottants permettent de représenter, de **manière approchée**, une partie des nombres réels. La valeur d'un réel ne peut être ni trop grande, ni trop précise. Cela dépend du nombre de bits utilisés (en général 32 ou 64 bits). C'est un domaine très complexe et il existe une norme, IEEE-754, pour que tout le monde obtienne les mêmes résultats. Le codage en binaire est de la forme :

signe	exposant (signé)	partie fractionnaire de la mantisse (non signé)

C’est-à-dire que si X est un nombre flottant :

$$X = (-1)^s * 2^e * 1,zzzzz\ldots$$

où s est le signe de X, e est l’exposant signé et zzzzz… est la partie fractionnaire de la mantisse. C’est, en binaire, la notation scientifique traditionnelle.

Voici quelques exemples de formats usuels :

nombre de bits	format	valeur max	précision max
32	1 + 8 + 23	$2^{128} \approx 10^{+38}$	$2^{-23} \approx 10^{-7}$
64	1 + 11 + 52	$2^{1024} \approx 10^{+308}$	$2^{-52} \approx 10^{-15}$
80	1 + 15 + 64	$2^{16384} \approx 10^{+4932}$	$2^{-64} \approx 10^{-19}$

1.3.4 Des codes particuliers

1.3.4.1 Le code BCD

Le code BCD (Binary Coded Decimal) permet de coder un nombre décimal en binaire. A chaque chiffre décimal, on fait correspondre un groupe de 4 bits comme pour la base hexadécimale. Mais ici, il n’y a pas de valeur supérieure à 9. Le code est le suivant :

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

On voit que c’est un code incomplet car il n’utilise que 10 combinaisons sur les 16 possibles. Ce code est utile quand on veut traiter des nombres décimaux en électronique numérique (par exemple pour dialoguer avec un être humain).

1.3.4.2 Le code Gray

Le code Gray est un code adjacent (ou un seul bit change quand on passe d'une valeur à la valeur suivante). On l'appelle aussi le code binaire réfléchi. On l'utilise dans les tableaux de Karnaugh mais aussi en conception numérique. Voici un exemple de code Gray sur 3 bits :

gray		
g_2	g_1	g_0
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

La construction peut se faire graphiquement par itération ou bien à partir du code en binaire naturel. Graphiquement :

on part de 0 / 1 puis on crée un axe de symétrie 0 / 1 (miroir) / 1 / 0

On ajoute le bit de poids fort en binaire naturel.

0 0
0 1
1 1
1 0

Pour prolonger le code (passer sur 3 bits), on re-crée un nouvel axe de symétrie sur les deux bits faibles, puis on ajoute un bit supplémentaire en binaire naturel :

0 0 0
0 0 1

$$\begin{array}{ccc} & & \overline{1} \\ 0 & 1 & 1 \\ 0 & \underline{1} & \underline{0} \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{array}$$

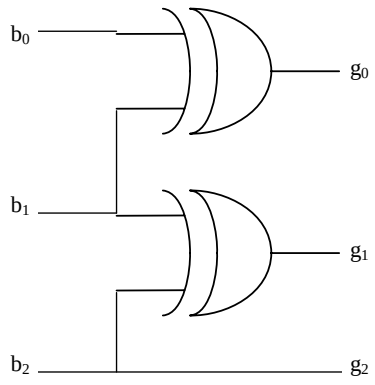
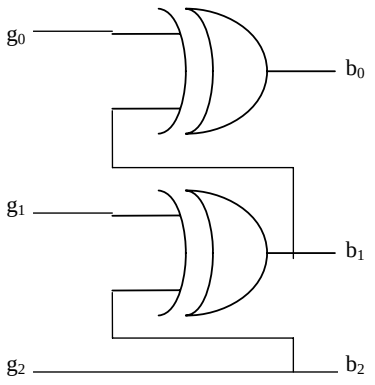
Et on recommence si on veut ajouter un bit supplémentaire.

En comparant le code binaire naturel et le code Gray sur 3 bits :

binaire		
b_2	b_1	b_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Gray		
g_2	g_1	g_0
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

On déduit les équations et les schémas suivants :

De binaire naturel vers code Gray	De code Gray vers binaire naturel
$g_0 = b_0 \oplus b_1$ $g_1 = b_1 \oplus b_2$ $g_2 = b_2$	$b_0 = g_0 \oplus b_1$ $b_1 = g_1 \oplus b_2$ $b_2 = g_2$
	

Sur un nombre quelconque de bits, on obtient les équations :

De binaire naturel vers code Gray	De code Gray vers binaire naturel
$g_i = b_i \text{ XOR } b_{i+1}$ pour $0 < i < N-2$ $g_{N-1} = b_{N-1}$	$b_i = g_i \text{ XOR } b_{i+1}$ pour $0 < i < N-2$ $g_{N-1} = b_{N-1}$

1.3.4.3 Le code Johnson

Le code Johnson est un autre exemple de code adjacent que l'on peut utiliser à la place du code Gray. En voici un exemple sur 3 bits :

Johnson		
b_2	b_1	b_0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0

On voit que c'est un code incomplet car il n'utilise que 6 combinaisons sur les 8 possibles.

1.3.4.4 Le code 1 parmi N

Le code 1 parmi N est très utilisé en conception numérique, notamment pour encoder les machines à états finis. Les Américains l'appellent le codage « one hot ». En voici un exemple sur 3 bits :

1 parmi 3		
b_2	b_1	b_0
0	0	1
0	1	0
1	0	0

On voit que c'est un code incomplet car il n'utilise que 3 combinaisons sur les 8 possibles.

1.3.4.5 Le code ASCII

Il n'y a pas que les nombres qui doivent être codé en binaire, mais aussi les caractères comme les lettres de l'alphabet, les signes de ponctuation, … Le code le plus connu et le plus utilisé est le code ASCII (American Standard Code for Information Interchange). Le code ASCII est un jeu normalisé de 128 caractères codés sur 7 bits, devenu un standard quasi universel. Il comporte tous les caractères alphanumériques non accentués et est lisible par pratiquement n'importe quelle machine. Ce sont les 8 premières lignes du tableau suivant.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	np	cr	so	si
1	dle	dc1	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del
8	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	np	cr	so	si
9	dle	dc1	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
a		¡	ć	£	ď	ě	ę	ğ	Í	©	ł			ŋ	ő	
b	°	±	²	³	ť	μ	¶	ů	ÿ	¹	ž	ż	ij	¡	¿	¿
c	À	Á	Â	Ã	Ä	Å	Æ	Ç	É	È	Ê	Ë	Ì	Í	Î	Ï
d	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	Ø	Ù	Ú	Û	Ü	Ý	Þ	β
e	à	á	â	ã	ä	à	æ	ç	è	é	ê	ë	ì	í	î	ï
f	ŏ	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Les 32 premiers codes sont utilisés comme caractères de contrôle pour représenter, par exemple, une fin de ligne ou une tabulation. Le code ASCII ne contient pas de caractères accentués et il a été complété par le code ISO-8859-1 (ou Latin 1). Les 128 premiers caractères correspondent au code ASCII, les 128 suivants aux caractères accentués et caractères spéciaux (voir les 8 dernières lignes du tableau).

Unicode est un jeu de caractères codé sur 16 bits (contre 7 ou 8 bits pour les standards actuels) qui permet le codage des caractères utilisés par toutes les langues du monde au sein d'une table unique. 16 bits permettent de coder 65 536 (2 puissance 16) caractères différents ce qui couvre largement les besoins en la matière. Les 256 premiers caractères d'Unicode correspondent au jeu ISO Latin 1.

1.3.4.6 Les codes détecteurs et/ou correcteurs d'erreurs

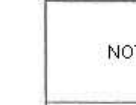
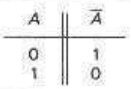
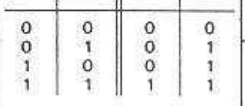
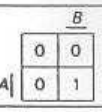
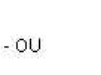
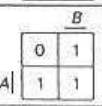
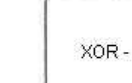
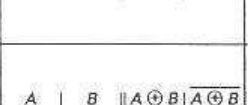
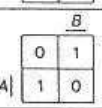
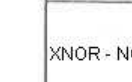
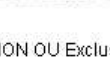
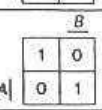
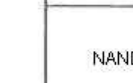
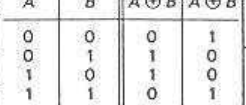
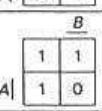
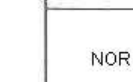
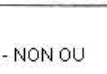
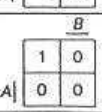
Si on craint que des erreurs se produisent dans un mot binaire (au cours d'une transmission par exemple), on rajoute au code des bits supplémentaires pour que le récepteur puisse détecter les erreurs et éventuellement les corriger. Par exemple, il y a un code détecteur et correcteur d'erreurs dans le train binaire enregistré sur un compact disque audio (code Reed-Solomon). Ces codes redondants sont un domaine d'étude complet de la théorie de l'information.

1.4 Circuits logiques combinatoires

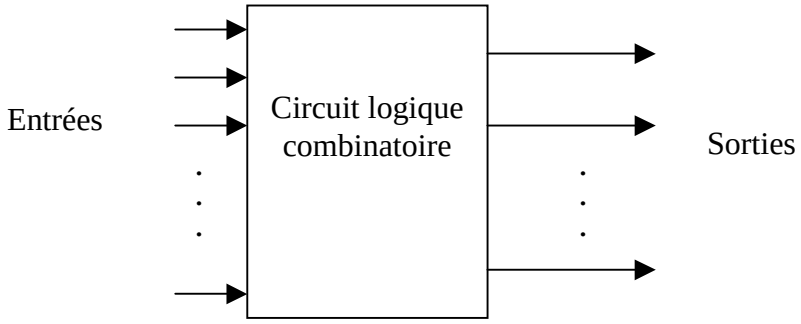
1.4.1 Circuits logiques fondamentaux

Il existe deux manières de dessiner ces opérateurs. La notation américaine est la plus utilisée (puisque les fabricants de composants sont souvent américains) et donc la plus connue mais la notation normalisée à l'avantage d'être plus facile à dessiner avec une table traçante. Le tableau suivant résume ce que nous avons déjà vu. Ces circuits logiques existent avec 2, 3 voir 4 entrées.

Tableau I. – Opérations logiques élémentaires.

Opération	Symbole usuel	Symbole normalisé	Table de vérité	Tableau de Karnaugh
NOT - INV				
AND - ET				
OR - OU				
XOR - OU Exclusif				
XNOR - NON OU Exclusif				
NAND - NON ET				
NOR - NON OU				

Les circuits logiques combinatoires sont des circuits constitués des portes ci-dessus fonctionnant simultanément et réalisant une ou plusieurs fonctions logiques.



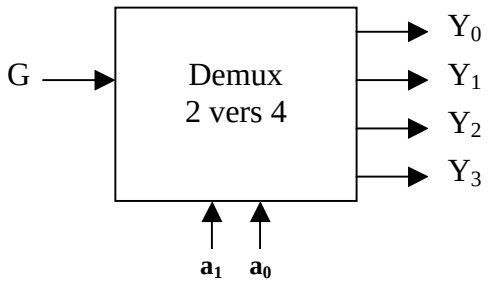
A une combinaison d'entrées (l'entrée) ne correspond qu'une seule combinaison de sorties (la sortie). La « sortie » apparaît après application de l' « entrée » avec un certain retard qui est le temps de propagation dans la logique interne. Ce temps est déterminé par la technologie utilisée, le nombre de portes traversées et la longueur des interconnections métalliques.

Les circuits combinatoires peuvent servir par exemple :

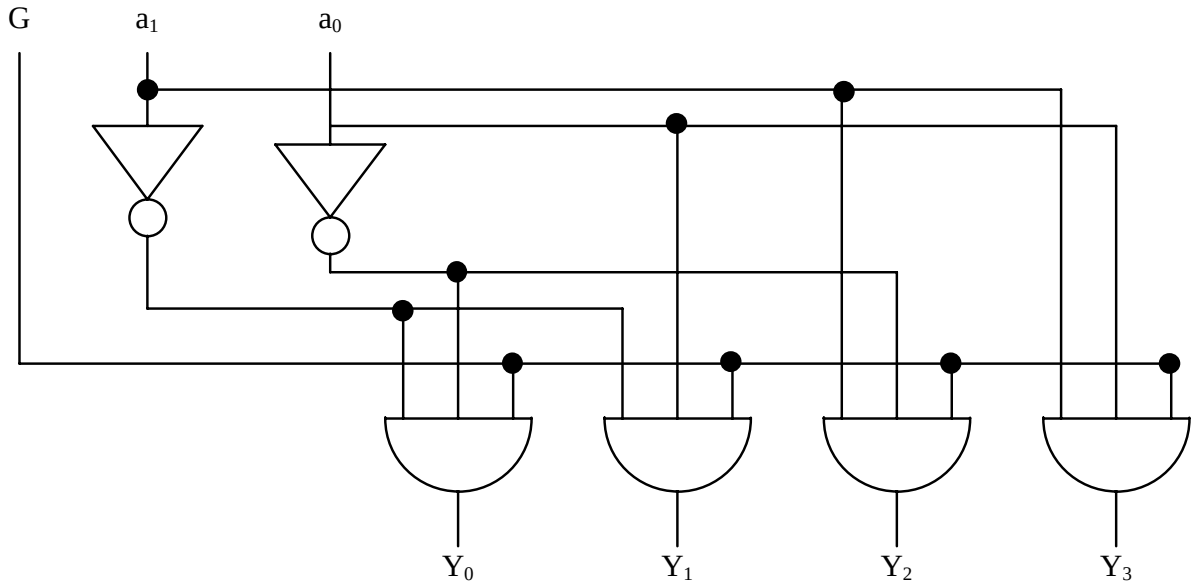
- à traduire des bits en chiffres représentant un nombre (ou des lettres ou un code particulier). On appelle ces circuits des codeurs (ou bien des décodeurs pour l'opération inverse). Par exemple, un codeur Gray ou bien BCD.
- à effectuer des opérations arithmétiques sur des nombres. Par exemple, un additionneur ou un multiplieur.
- à transmettre ou recevoir des informations sur une ligne unique de transmission (une ligne série), ce qui nécessite de transformer un nombre écrit sous forme parallèle en une suite de bits mis en série et vice-versa. C'est le rôle des circuits multiplexeur/démultiplexeur. Voici par exemple une transformation série/parallèle suivie d'une transformation parallèle/série :

1.4.2 Le démultiplexeur

C'est un circuit qui aiguille une entrée vers une sortie dont on donne l'adresse sous forme d'un nombre codé en binaire.



Le schéma fondamental d'un démultiplexeur est :



Sa table de vérité est égale à :

N	a_1	a_0	$Y_0 = \overline{a_0}.\overline{a_1}.G$	$Y_1 = a_0.\overline{a_1}.G$	$Y_2 = \overline{a_0}.a_1.G$	$Y_3 = a_0.a_1.G$
0	0	0	G	0	0	0
1	0	1	0	G	0	0
2	1	0	0	0	G	0
3	1	1	0	0	0	G

On obtient donc l'équation suivante : $Y_K(N) = G$ si $K = N$

$Y_K(N) = 0$ si $K \neq N$

Si G est une donnée, le circuit est un démultiplexeur (un aiguillage divergent). Son utilisation première est la conversion série/parallèle des données. Nous verrons plus tard qu'il peut aussi servir pour réaliser des fonctions combinatoires.

La limitation de ce type de circuit est le nombre de broches qui croit fortement avec le nombre de variables d'adresse N. Il évolue en $N + 2^N + 1$. Dans le TTL data book, le plus gros décodeur est un 4 vers 16 nécessitant un boîtier 24 broches. Un exemple typique de démultiplexeur est le SN74LS155, double démultiplexeur 2/4 avec entrée de validation (les entrées et les sorties pourvues d'une bulle sont actives à l'état bas) :

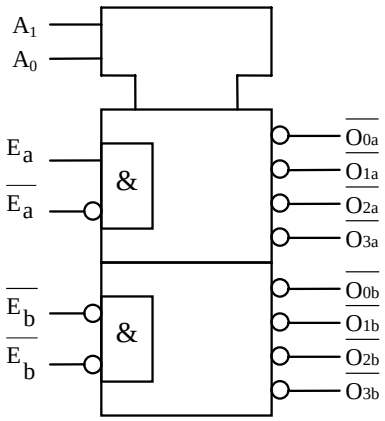
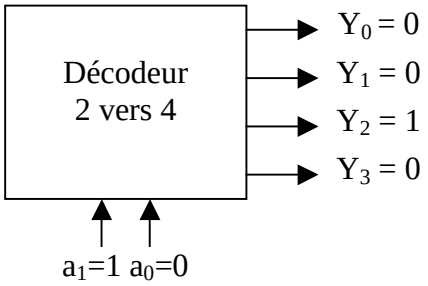


Table de vérité													
Adresses		Validation a		SORTIES a				Validation b		SORTIES b			
A_1	A_0	Ea	$\overline{E_a}$	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$	$\overline{E_b}$	$\overline{E_b}$	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$
X	X	0	X	1	1	1	1	1	X	1	1	1	1
X	X	X	1	1	1	1	1	X	1	1	1	1	1
0	0	1	0	0	1	1	1	0	0	0	1	1	1
0	1	1	0	1	0	1	1	0	0	1	0	1	1
1	0	1	0	1	1	0	1	0	0	1	1	0	1
0	1	1	0	1	1	1	0	0	0	1	1	1	0

X : l'entrée peut prendre n'importe quelle valeur

1.4.3 Le décodeur

Le décodeur est de la même famille que le démultiplexeur, mais avec l'entrée G qui vaut 1 en permanence. On trouve alors en sortie un 1 parmi des 0.



On appelle aussi ce circuit un décodeur d'adresse utilisé pour adresser les différentes lignes d'une mémoire. Sa table de vérité est égale à :

N	a_1	a_0	$Y_0 = \overline{a_0}.\overline{a_1}$	$Y_1 = a_0.\overline{a_1}$	$Y_2 = \overline{a_0}.a_1$	$Y_3 = a_0.a_1$
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

On obtient donc l'équation suivante : $Y_K(N) = 1$ si $K = N$

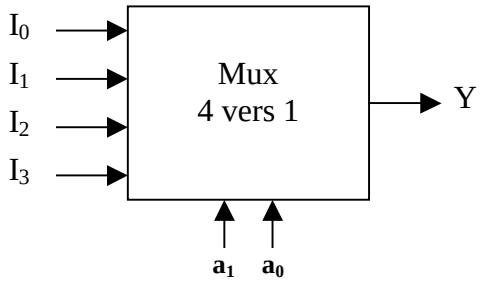
$Y_K(N) = 0$ si $K \neq N$

Comme pour le démultiplexeur, la limitation de ce type de circuit est le nombre de broches qui croit fortement avec le nombre de variables d'adresse N.

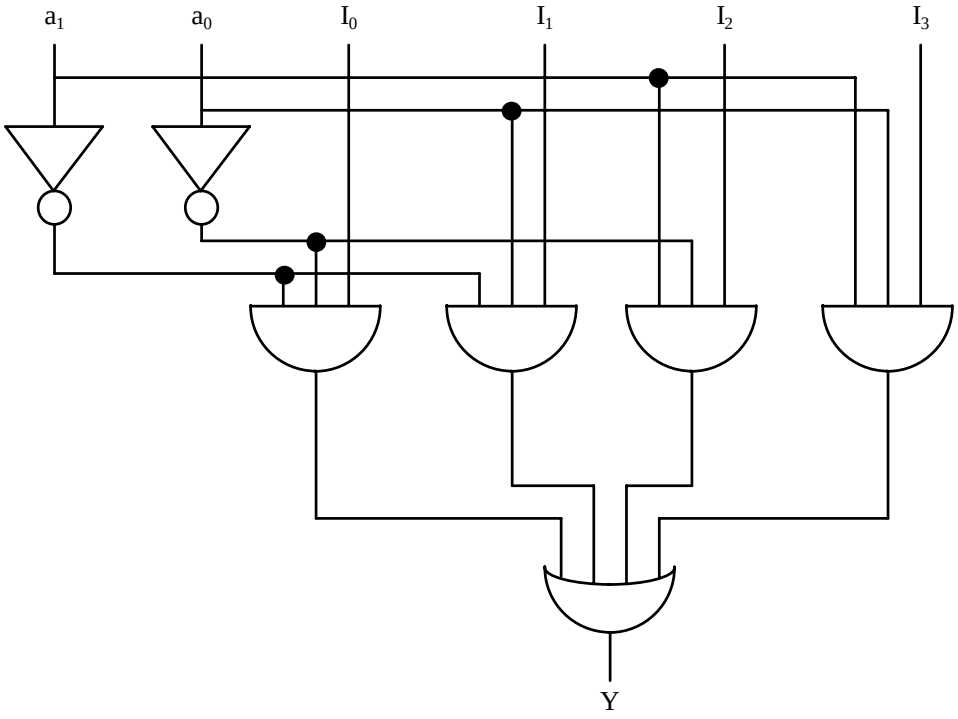
1.4.4 Le multiplexeur

Le multiplexeur est la fonction inverse du démultiplexeur. C'est un sélecteur de données ou aiguillage convergent. Il peut transformer une information apparaissant sous forme de n bits

en parallèle en une information se présentant sous forme de n bits en série. La voie d'entrée, sélectionnée par son adresse, est reliée à la sortie.



Le schéma logique est le suivant :



Son équation de sortie est égale à : $Y = I_0.\overline{a_1}.\overline{a_0} + I_1.\overline{a_1}.a_0 + I_2.a_1.\overline{a_0} + I_3.a_1.a_0$ avec la table de vérité :

	a_1	a_0	Y
0	0	0	I_0
1	0	1	I_1
2	1	0	I_2

3	1	1	I_3

Nous verrons aussi qu'un multiplexeur peut servir à réaliser des fonctions logiques quelconques. La limitation de ce type de circuit est toujours le nombre de broches qui croit avec le nombre de variables d'adresse N de la même manière que le démultiplexeur. Un exemple typique de multiplexeur est le SN74LS153, double multiplexeur 4 entrées avec signal d'activation :

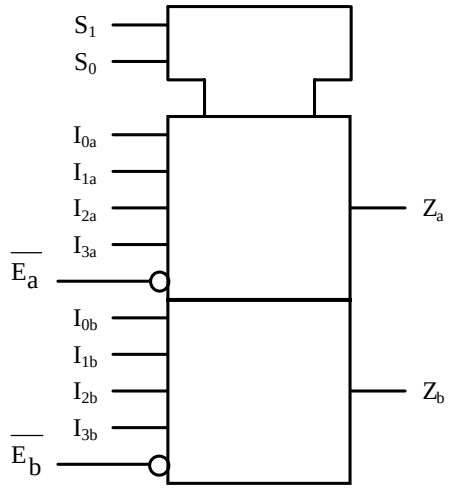
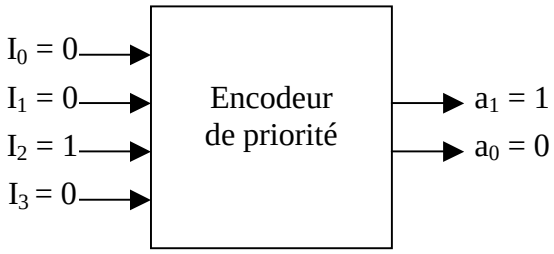


Table de vérité							
Sélection		Entrées (a ou b)					Sorties (a ou b)
S_1	S_0	$\overline{E}$	I_0	I_1	I_2	I_3	Z
X	X	1	X	X	X	X	0
0	0	0	0	X	X	X	0
0	0	0	1	X	X	X	1
0	1	0	X	0	X	X	0
0	1	0	X	1	X	X	1
1	0	0	X	X	0	X	0
1	0	0	X	X	1	X	1
1	1	0	X	X	X	0	0
1	1	0	X	X	X	1	1

1.4.5 L'encodeur de priorité

L'encodeur est la fonction inverse du décodeur. On met sur les entrées un 1 parmi des 0 et on obtient sur les sorties l'adresse de l'entrée à 1. La priorité ne sert que quand plusieurs entrées

sont à 1 en même temps. Le circuit donne alors l'adresse de l'entrée dont le rang est le plus élevé. Il y a donc priorité aux entrées de rang le plus élevé.



Le SN74LS348 est un exemple d'encodeur à 8 entrées. L'ordre de priorité va de l'entrée 7 vers l'entrée 0. Les sorties GS et EO et l'entrée EI servent à mettre en cascade plusieurs encodeurs. Les entrées et les sorties sont actives à 0.

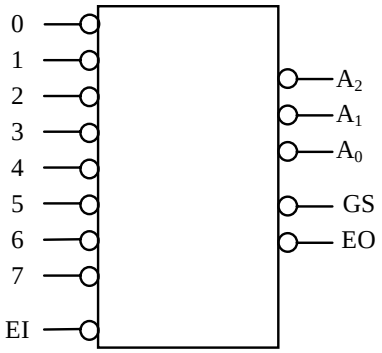


Table de vérité													
Entrées									Sorties				
EI	0	1	2	3	4	5	6	7	A_2	A_1	A_0	GS	EO
1	X	X	X	X	X	X	X	X	Z	Z	Z	1	1
0	1	1	1	1	1	1	1	1	Z	Z	Z	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

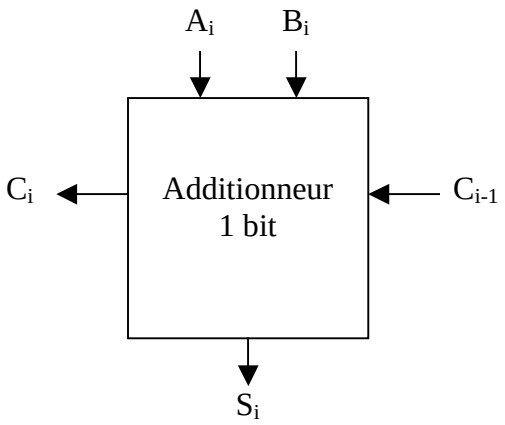
1.4.6 Les circuits arithmétiques

Les circuits arithmétiques effectuent des opérations binaires telles que l'addition, la multiplication, la comparaison ou combinent ces opérateurs de base au sein d'une UAL (Unité Arithmétique et Logique). Beaucoup de ces circuits sont aujourd'hui obsolètes et ne sont plus commercialisés en tant que composant discret mais la fonction existe toujours à l'intérieur des circuits comme les microprocesseurs ou les ASIC. Prenons l'exemple d'une simple addition en binaire naturel sur 4 bits :

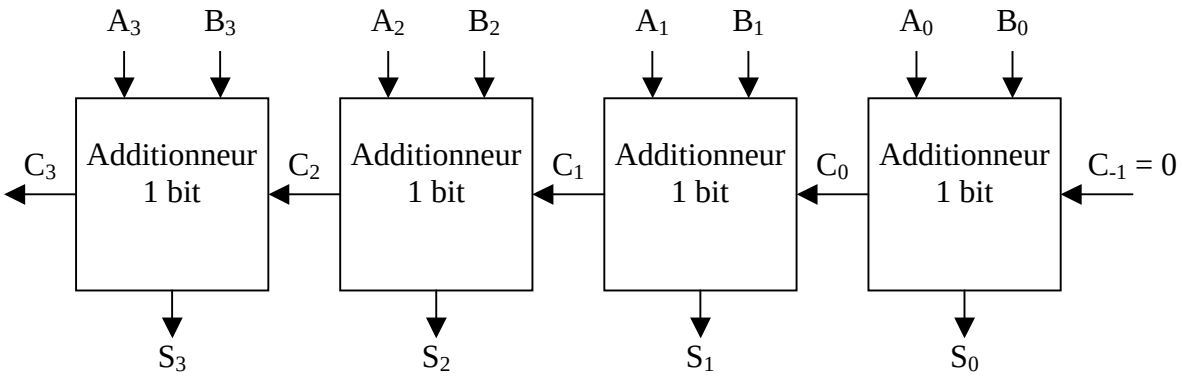
A	$(5)_{10}$	$(0101)_2$
B	$(3)_{10}$	$(0011)_2$
S	$(8)_{10}$	$(1000)_2$

1ère colonne : 1 + 1 = 0 avec une retenue sortante à 1 ($(2)_{10} = (10)_2$).
2ème colonne : la retenue rentrante vaut 1. 1 + 0 + 1 = 0 avec une retenue sortante à 1.
3ème colonne : la retenue rentrante vaut 1. 1 + 1 + 0 = 0 avec une retenue sortante à 1.
4ème colonne : la retenue rentrante vaut 1. 1 + 0 + 0 = 1.

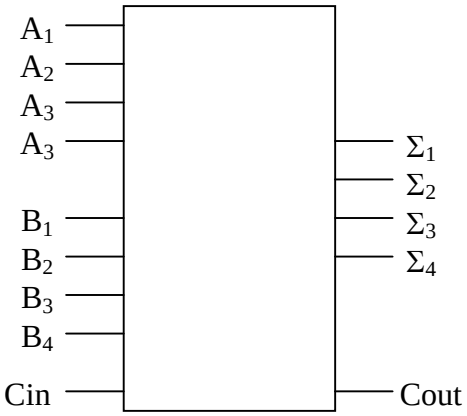
Sur les deux colonnes du milieu, il y a une retenue entrante et une retenue sortante. On peut donc définir les entrées-sorties d'un additionneur traitant le ième bit du calcul :



L'entrée C_{i-1} est la retenue entrante (Carry en anglais) et la sortie C_i est la retenue sortante. Vous verrez le schéma interne de cet additionneur en exercice dirigé. L'additionneur complet est construit de la manière suivante :



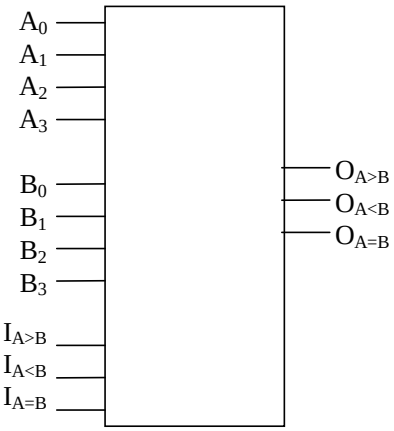
Vu de l'extérieur, un additionneur (ou un soustracteur, en CA2 c'est la même chose) se présente donc sous la forme suivante.



Ils effectuent l'addition (ou la soustraction) de deux nombres codés sur N bits. Le résultat est codé sur N bits avec une sortie supplémentaire pour la retenue. Il y a aussi une entrée de retenue pour pouvoir mettre en cascade les circuits. Sur 4 bits, L'opération décimale exacte réalisée est (le + est une addition, le . est une multiplication) :

$$Cin + (A_1+B_1)+2.(A_2+B_2)+ 4.(A_3+B_3)+8.(A_4+B_4) = \Sigma_1+2.\Sigma_2+4.\Sigma_3+8.\Sigma_4+16.Cout$$

Les comparateurs sont une autre grande famille de circuit arithmétique. Ils effectuent sur N bits, les opérations d'égalité, de supériorité ou d'infériorité. Les entrées $I_{A>B}$, $I_{A<B}$ et $I_{A=B}$ servent à mettre en cascade plusieurs circuits.



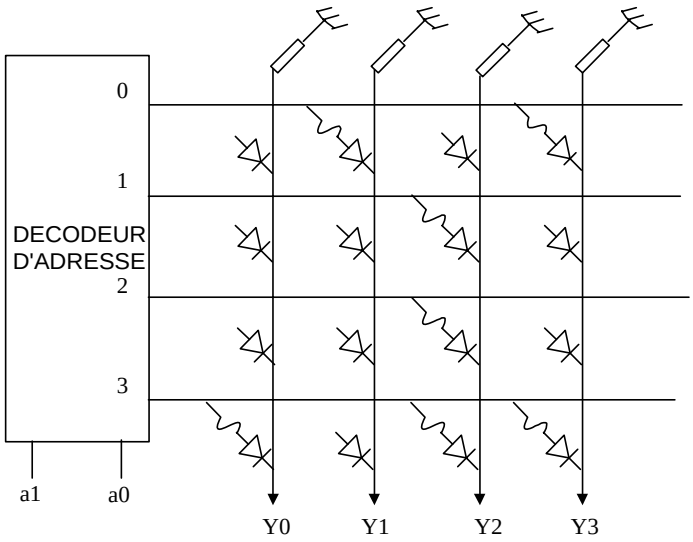
Par exemple, la table de vérité du SN74LS85 est :

Table de vérité									
A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	1	0	0
$A_3<B_3$	X	X	X	X	X	X	0	1	0
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	1	0	0
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	1	0	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	1	0	0	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	0	1	1	0

Il existe d'autres circuits comme le multiplieur ou le diviseur dont l'étude sort du cadre de ce cours.

1.4.7 Les mémoires

Les PROM sont des mémoires mortes programmables. Le schéma ci-dessous montre un exemple de PROM bipolaire à fusible.



L'adresse appliquée sur les entrées (a1 a0) provoque, via le décodeur d'adresses, la mise à 1 de la ligne d'adresse correspondante. Si le fusible existant entre cette ligne et la sortie Yn est intact, cette sortie vaut 1. Si le fusible a été claqué pendant la programmation, la sortie vaut 0 (à travers une résistance de pull-down). La programmation est réalisée avec un programmateur d'EPROM, le fichier contenant la configuration pouvant être au format JEDEC (Joint Electronics Design Engineering Council). Dans cet exemple, la PROM (4 mots de 4 bits) contient les données suivantes :

a1	a0	Y0	Y1	Y2	Y3
0	0	0	1	0	1
0	1	0	0	1	0
1	0	0	0	1	0
1	1	1	0	1	1

On n'utilise plus de PROM pour réaliser directement une fonction logique mais le principe, lui, demeure tout à fait d'actualité. Certains circuits, comme les circuits logiques programmables par exemple, utilisent en interne de mémoires pour réaliser des fonctions logiques.

1.4.8 Réalisation d'une fonction logique combinatoire

Ce paragraphe explique les différentes méthodes de réalisation de fonctions logiques avec des circuits standards. Rappelons tout d'abord les trois formes de fonction logique F sur un exemple avec la table de vérité suivante.

N	a2	a1	a0	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

1. La forme canonique complète est :

$$F = 1.\bar{a}_2.\bar{a}_1.\bar{a}_0 + 1.\bar{a}_2.\bar{a}_1.a_0 + 0.\bar{a}_2.a_1.\bar{a}_0 + 0.\bar{a}_2.a_1.a_0 + 0.a_2.\bar{a}_1.\bar{a}_0 + 1.a_2.\bar{a}_1.a_0 + 1.a_2.a_1.\bar{a}_0 + 0.a_2.a_1.a_0$$

2. La forme canonique abrégée est :

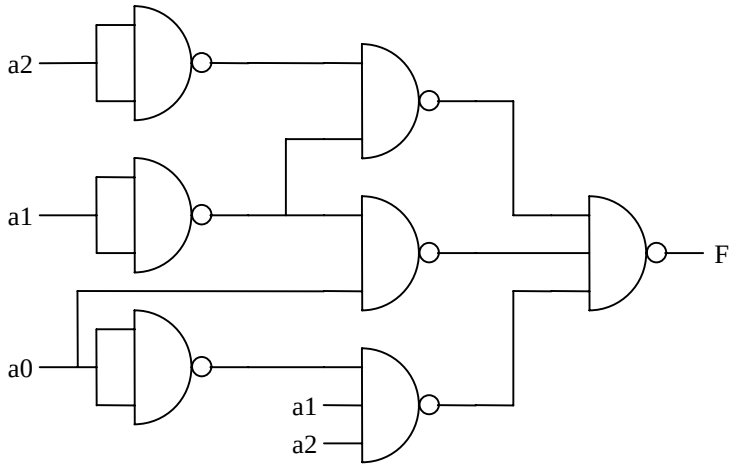
$$F = \bar{a}_2.\bar{a}_1.\bar{a}_0 + \bar{a}_2.\bar{a}_{.1}.a_0 + a_2.\bar{a}_1.a_0 + a_2.a_1.\bar{a}_0 = \Sigma(0, 1, 5, 6)$$

3. La forme simplifiée par le tableau de Karnaugh est :

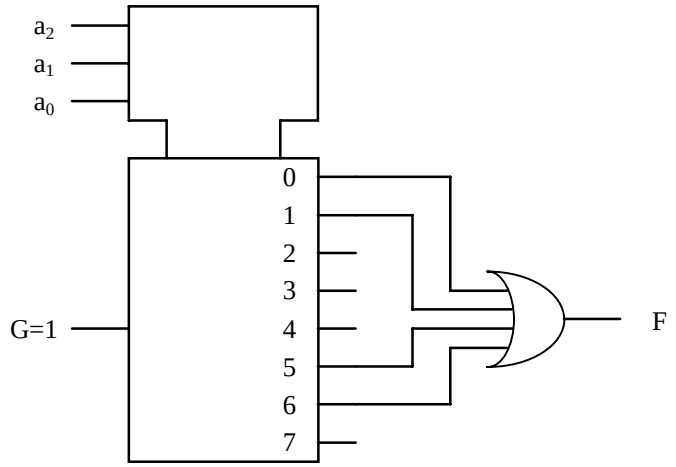
$$F = \bar{a}_2.\bar{a}_1 + \bar{a}_{.1}.a_0 + a_2.a_1.\bar{a}_0$$

A partir de ces trois formes et de la table de vérité, on peut réaliser la fonction logique F soit avec un multiplexeur, soit avec un décodeur, soit avec des opérateurs fondamentaux, soit avec une PROM.

- Avec la forme simplifiée par le tableau de Karnaugh et le théorème de De Morgan, on réalise F avec, par exemple, des portes NAND :



- Avec la forme canonique abrégée, on réalise F avec un décodeur et une porte OR :



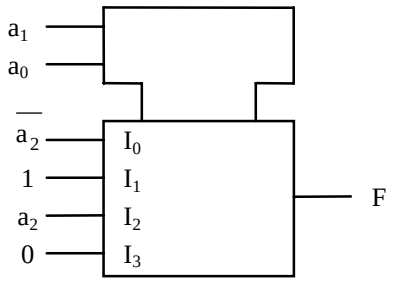
- Avec la forme canonique complète, on réalise F avec un multiplexeur. En effet, on a :

$$F = 1.\bar{a}_2.\bar{a}_1.\bar{a}_0 + 1.\bar{a}_2.\bar{a}_1.a_0 + 0.\bar{a}_2.a_1.\bar{a}_0 + 0.\bar{a}_2.a_1.a_0 + 0.a_2.\bar{a}_1.\bar{a}_0 + 1.a_2.\bar{a}_1.a_0 + 1.a_2.a_1.\bar{a}_0 + 0.a_2.a_1.a_0$$

D’où on tire :

$$F = \bar{a}_1.\bar{a}_0\left(1.\bar{a}_2 + 0.a_2\right) + \bar{a}_1.a_0\left(1.\bar{a}_2 + 1.a_2\right) + a_1.\bar{a}_0\left(0.\bar{a}_2 + 1.a_2\right) + a_1.a_0\left(0.\bar{a}_2 + 0.a_2\right)$$
$$= \bar{a}_1.\bar{a}_0.\bar{a}_2 + \bar{a}_1.a_0.1 + a_1.\bar{a}_0.a_2 + a_1.a_0.0$$

Comme, l'équation de sortie d'un multiplexeur est $F = I_0.\bar{a}_1.\bar{a}_0 + I_1.\bar{a}_1.a_0 + I_2.a_1.\bar{a}_0 + I_3.a_1.a_0$, on obtient en identifiant les deux expressions le schéma suivant :



CONCLUSION : Avec un multiplexeur à 2^n entrées I_k, on peut réaliser n'importe quelle fonction de n+1 variables. Les entrées I_k valent toujours : a_n, $\overline{a}_n$, 1 ou 0 (a_n étant la n+1 variable). La réalisation d'une fonction logique avec un multiplexeur est à la base du fonctionnement des FPGA.

- Avec la table de vérité, on réalise F avec une PROM. Dans notre exemple, il faut connecter $a_2a_1a_0$ sur les adresses de la mémoire et charger les 8 bits aux adresses 0 à 7. F est alors disponible sur le bit de donnée. La réalisation d'une fonction logique avec une PROM est à la base du fonctionnement des PAL et des EPLD.

Le choix du circuit ou des circuits dépend de la (ou des) fonction(s) logique(s) à réaliser. Chaque méthode a ses avantages et ses inconvénients. Il faut prendre en compte le nombre et le prix des boîtiers utilisés pour calculer le coût de la fonction et il faut déterminer le nombre de couches logiques traversées pour calculer sa vitesse. Les domaines d'utilisation sont les suivants :

⇒ Les portes élémentaires sont utilisées quand il y a peu de fonctions et de variables d'entrées.

⇒ Le décodeur est utilisé quand il y a peu de variables d'entrées (les boîtiers ont un nombre limité de broches) et beaucoup de sorties (on câble un OR par sortie).

⇒ Le multiplexeur ne permet de réaliser qu'une seule fonction (il n'a qu'une sortie) avec quelques variables d'entrées (une de plus que le décodeur à taille de boîtier équivalente). Il nécessite le câblage le plus simple et le plus petit boîtier.

⇒ La PROM est utilisée quand il y a beaucoup de variables d'entrées et quelques variables de sorties (8 ou 16 au maximum). Par exemple, une PROM 1024x8 (1024 octets) permet de réaliser 8 fonctions de 10 variables.

1.5 Caractéristiques temporelles

Nous étudierons ici les différents temps que l'on trouve dans les circuits combinatoires. Tous ces temps varient suivant les technologies (ECL, TTL, CMOS) de quelques dizaines de ps à environ 10 ns.

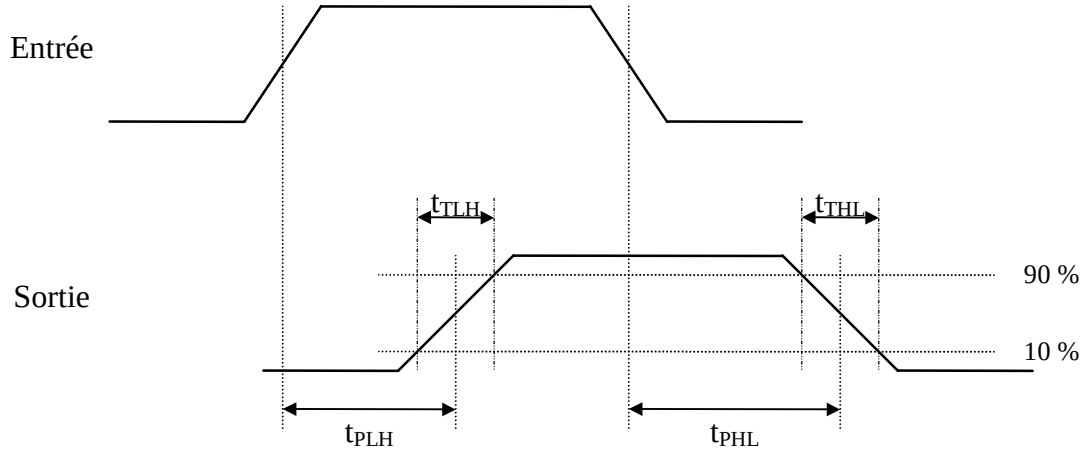
Dans les notices du constructeur, les temps min ou max (selon le cas) sont donnés pour le plus mauvais des circuits, et dans le cas le plus défavorable (circuit le plus chargé et à température maximale). Une étude faite avec ces temps fournit un montage fonctionnant dans 100% des cas. Les temps typiques sont moins contraignants mais ils ne représentent pas le cas le plus défavorable. Ils peuvent êtres pris pour des circuits peu chargés mais avec un plus grand risque.

1.5.1 Caractéristiques temporelles

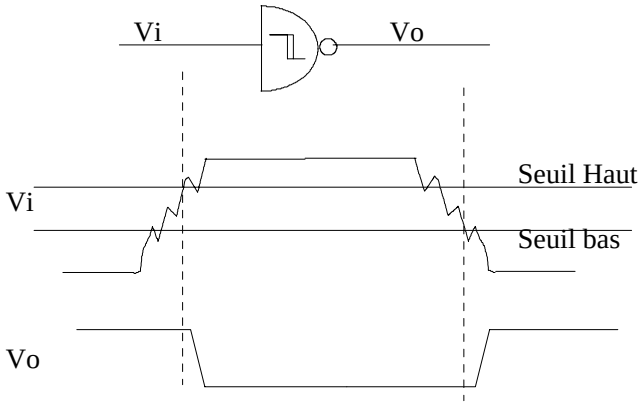
Considérons une porte logique. Les temps de transition et de propagation sont définis de la manière suivante :

- le temps de propagation est le retard entre les signaux d'entrée et de sortie. Il est causé par le temps de traversée des transistors et des autres composants formant le circuit logique. Il en existe deux types : t_{PLH} (la sortie passe de 0 à 1 « low to high ») et t_{PHL} (la sortie passe de 1 à 0 « high to low »).
- le temps de transition est le temps mis par une sortie pour changer d'état. Il est généralement pris entre 10 et 90 % du niveau maximum. Il en existe deux types : t_{TLH} (la sortie passe de 0 à 1) et t_{THL} (la sortie passe de 1 à 0). Ce temps est très dépendant de la charge (capacitive notamment) sur la sortie du circuit.

Le dessin suivant en donne une illustration pour une porte non-inverseuse :



A l'entrée d'un circuit logique dépourvu d'un déclencheur à seuil (trigger de Schmitt), on doit respecter un temps de transition maximum (t_{mmax}), sous peine de transitions parasites en sortie. Dans le cas de signaux à temps de transitions trop longs ou présentant des parasites, une porte pourvue d'un trigger permet de mettre en forme ces signaux pour qu'ils puissent attaquer correctement la logique :

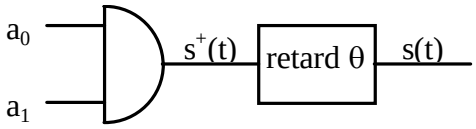


En sortie de circuit, les temps de transitions dépendent de la technologie employée mais pas du temps de montée du signal d'entrée, tant que celui-ci reste inférieur à 3 à 5 fois les temps de transitions usuels de cette technologie.

1.5.2 Etats transitoires

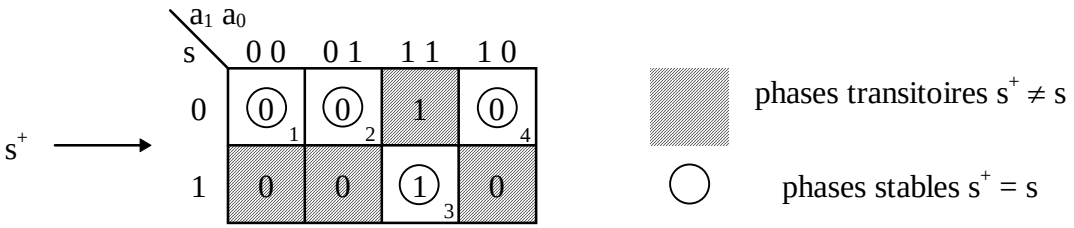
Le fonctionnement normal d'une carte logique combinatoire ne peut être déduit du fonctionnement des circuits qui la composent en supposant ces circuits idéaux (temps de transition nuls, temps de propagation nuls). Dans ce paragraphe, nous allons proposer une

méthode d'analyse des aléas de commutation en tenant compte des temps de propagation. Pour cela, nous allons définir le modèle asynchrone d'un circuit combinatoire :

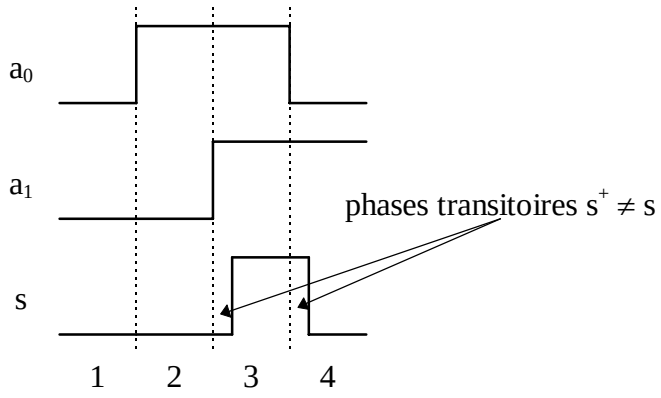


avec : $\theta = \sup(t_{PLH}, t_{PHL})$
$s(t) = s^+(t - \theta)$ ou
$s(t + \theta) = s^+(t)$

Il faut que la durée de l'impulsion active sur l'entrée d'un circuit combinatoire soit supérieure au retard θ pour que la sortie change d'état. Sinon, l'énergie de l'impulsion est insuffisante pour provoquer la commutation. Pour analyser le fonctionnement d'un circuit combinatoire à N entrées et une sortie, on détermine la table d'excitation du circuit. C'est un tableau de Karnaugh à N+1 variables (les entrées plus la sortie s) permettant de déterminer s^+. Quand s^+ est différent de s dans ce tableau (zone hachurée), nous sommes en présence d'un régime transitoire (s va changer après un temps θ). Quand s^+ est égal à s (valeur encerclée), nous sommes dans un état stable (s reste dans le même état).

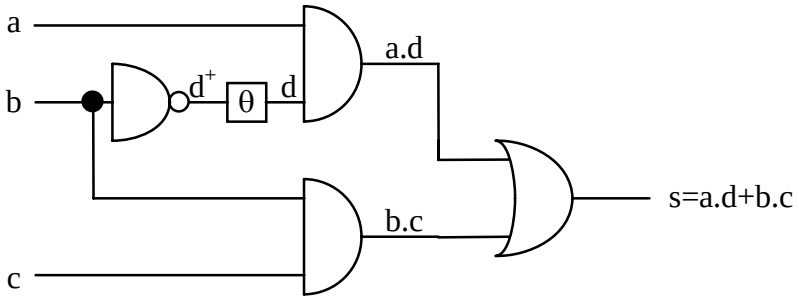


On se sert de cette table d'excitation pour établir le chronogramme du circuit à partir d'une séquence d'entrée quelconque (par exemple 0,0 ; 0,1 ; 1,1 ; 1,0) :

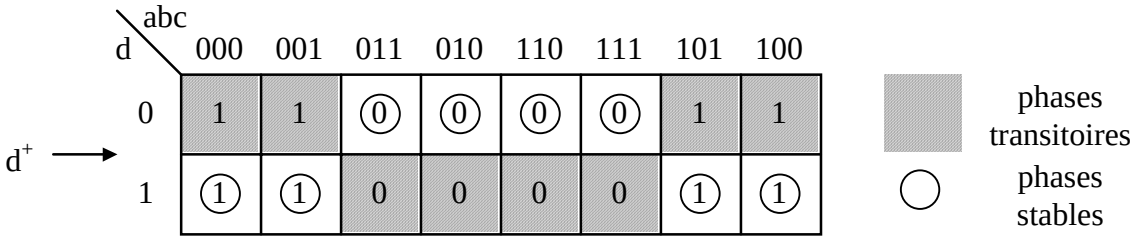


Prévision des aléas de commutation : un exemple

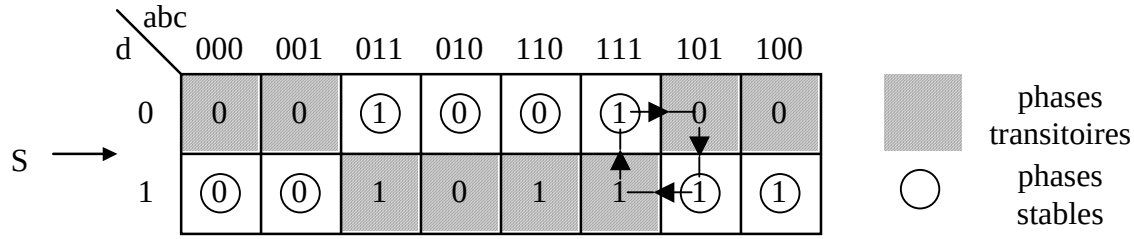
Soit le circuit suivant :



Pour la méthode d'analyse, toutes les portes sont à retard nul et un retard θ n'est ajouté que sur l'inverseur. La table d'excitation de d^+ en fonction de a, b, c, d est ($d^+ = \overline{b}$) :



On en déduit la table de vérité de la sortie du circuit :



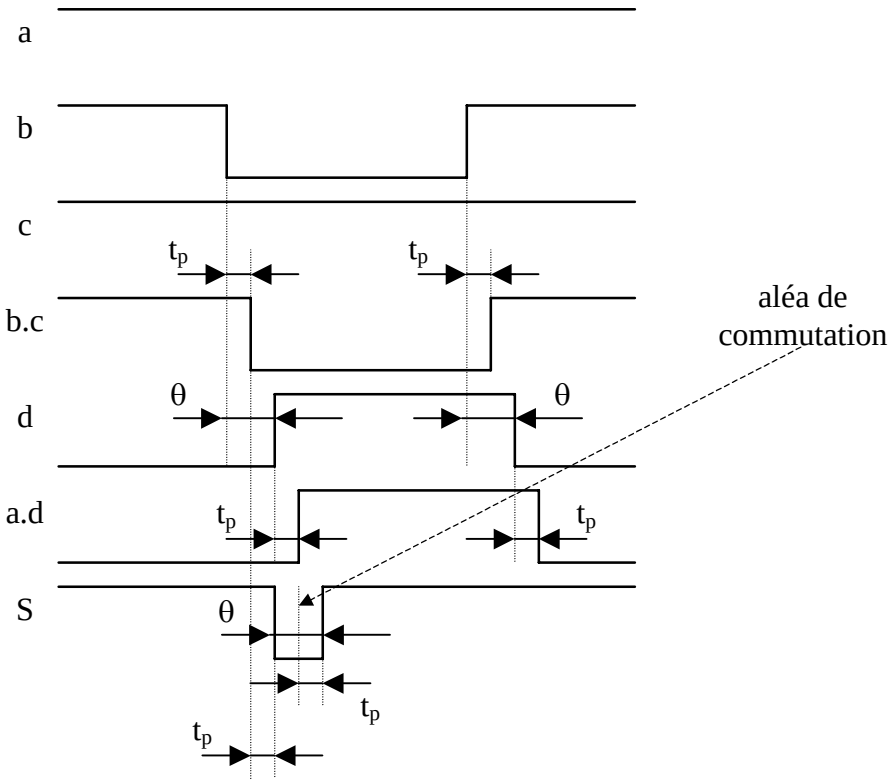
Etudions maintenant la transition b = 1, 0, 1 avec a et c qui restent en permanence à 1. On a alors l'évolution suivante en sortie :

abc = 111	S = 1	

abc = 101	Phase transitoire : S = 0 Etat final : S = 1.	Aléa de commutation
abc = 111	Phase transitoire : S = 1 Etat final : S = 1.	Pas d'aléa

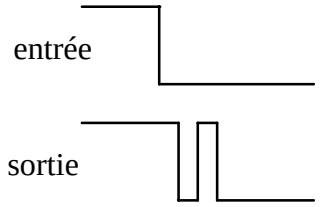
Sans utiliser cette méthode d'analyse, on aurait trouvé S = 1. En effet, cet aléa de commutation n'apparaît pas sur la table de vérité purement combinatoire de S. On nomme cette impulsion dont la durée ne dépend que du temps de propagation θ de l'inverseur, un « glitch ».

Dans l'analyse précédente, on a pris un temps de propagation nul pour les portes autres que l'inverseur afin de simplifier le calcul. Introduire un temps de propagation t_p sur ces portes ne change rien au résultat final comme nous le montre le chronogramme suivant. L'aléa est simplement décalé de $2.t_p$, mais sa durée reste égale à θ.



Quand un circuit logique combinatoire a la possibilité de produire cet état transitoire, on dit qu'il existe un « hazard » (que le glitch se produise ou non). Ces notions de glitch et de hazard ne s'appliquent que pour des circuits logiques combinatoires purs, c'est-à-dire sans rebouclage des sorties sur les entrées. Deux types de hazard existent :

- **Hazard statique.** Un hazard statique à 1 est la possibilité pour un circuit de produire un glitch à 0 alors que la sortie aurait normalement du rester à 1. C'est le cas dans l'exemple précédent. Un hazard statique à 0 est la possibilité pour un circuit de produire un glitch à 1 alors que la sortie aurait normalement du rester à 0.
- **Hazard dynamique.** Un hazard dynamique est la possibilité que la sortie d'un circuit change plus d'une fois pour un seul changement logique sur une entrée.



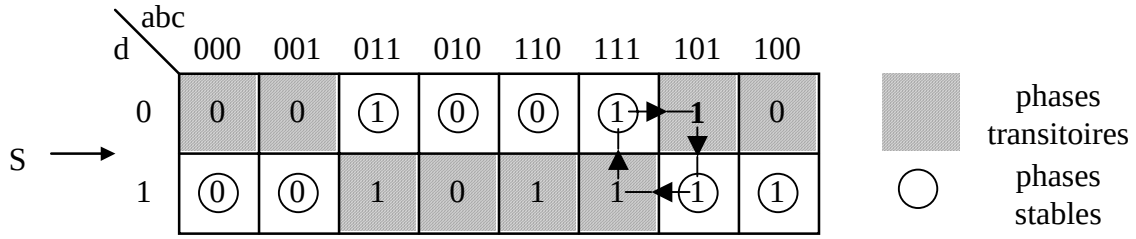
De multiples transitions sur une sortie peuvent se produire s'il y a plusieurs chemins logiques avec des retards différents entre l'entrée et la sortie qui changent.

Il est important de noter que le hazard statique ou dynamique est borné dans le temps. Une fois que le chemin combinatoire qui traverse le plus de portes logiques (**le chemin critique**) a été traversé, le signal est forcément stable en sortie. Le temps de traversée du chemin critique s'appelle **le temps critique**.

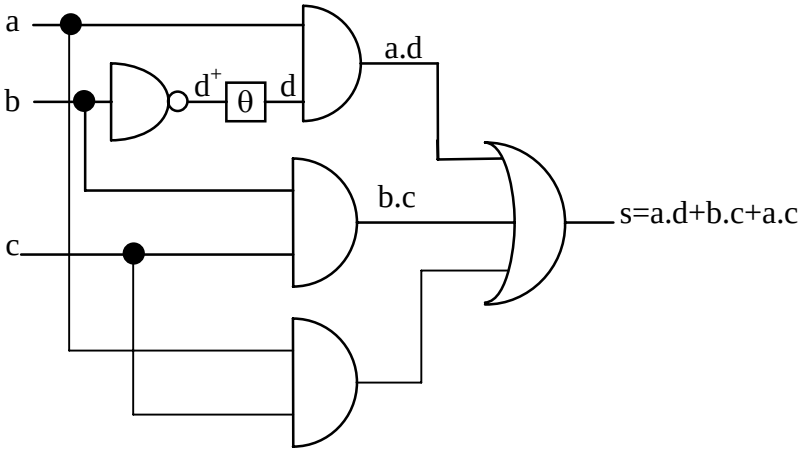
Le signal en sortie du circuit combinatoire est stable « temps critique » après le dernier changement sur les entrées. Il ne peut se produire aucun changement passé ce délai.

Pour la conception de circuits logiques séquentiels (avec rebouclage des sorties sur les entrées), il faut utiliser des circuits logiques combinatoires sans hazards (« hazard-free circuits »). Il est possible d'éliminer les hazards d'un circuit en utilisant la méthode d'analyse vue précédemment et en modifiant le tableau de karnaugh afin d'en éliminer les états transitoires. En effet, les phases transitoires du tableau de S peuvent être modifiées sans que

cela influe sur l'état stable de S. Cela ne change S que pendant les transitions. Dans notre exemple, il suffit de changer l'état abcd = 1010 en passant S de 0 à 1 pour supprimer le glitch.



Cela revient à modifier l'équation de S qui est maintenant S = a.d + b.c + a.c. La transition b = 1, 0, 1 avec a et c qui restent en permanence à 1 ne peut plus provoquer d'aléa de commutation puisque S vaut 1 dans tous les cas. Le schéma du circuit « hazard-free » est donc :



Remarque : il est important de noter que pour une porte logique élémentaire, il ne peut y avoir de glitch en sortie si une seule de ses entrées change à la fois.

1.6 Exercices

exercice 1.1

1 Rappeler la table de vérité des portes fondamentales OR, AND, NOR, NAND, XOR, XNOR.

2 Exprimer par une fonction logique que :

- Les variables A, B, C et D sont toutes égales à 1.
- Toutes les variables A, B, C et D sont nulles.
- Au moins l'une des variables A, B, C, D est égale à 1.
- Au moins l'une des variables A, B, C, D est égale à 0.
- Les variables A, B, C et D prennent les valeurs 0,1,1,0.

3 Simplifier les fonctions logiques suivantes :

$$F_1 = (A + \overline{B}).\overline{C} + A + \overline{B}.C$$

$$F_2 = A.C + B.\overline{C} + A.B$$

$$F_3 = \overline{A.B + \overline{A}.\overline{B}}$$

$$F_4 = A + B + \overline{A}.\overline{B}$$

4 Démontrer les relations suivantes :

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

$$(A + B).(A + \overline{B}) = A$$

$$A + A.B = A$$

$$A.(A + B) = A$$

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

$$A.(\overline{A} + B) = A.B$$

$$\overline{A \oplus B} = \overline{A} \oplus B = A \oplus \overline{B}$$

$$1 \oplus A = \overline{A}$$

$$0 \oplus A = A$$

5 Montrer que $\overline{A.C + B.\overline{C}} = \overline{A}.C + \overline{B}.\overline{C}$.

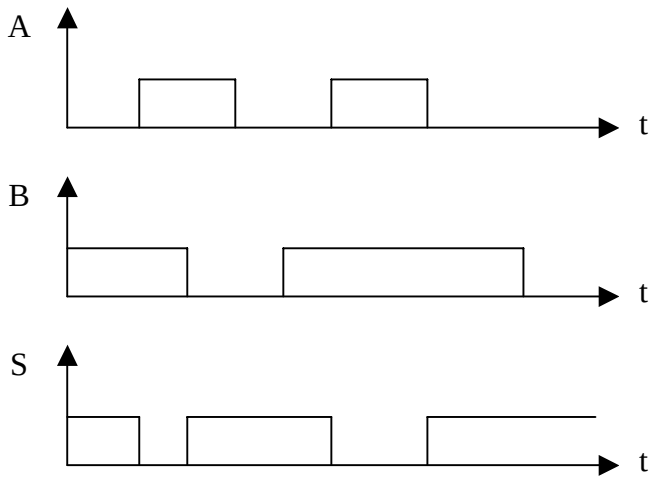
6 Calculer les compléments de :

$$(A + B).(\overline{A} + \overline{B})$$

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

$$\overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.(B.C + \overline{B}.\overline{C})$$

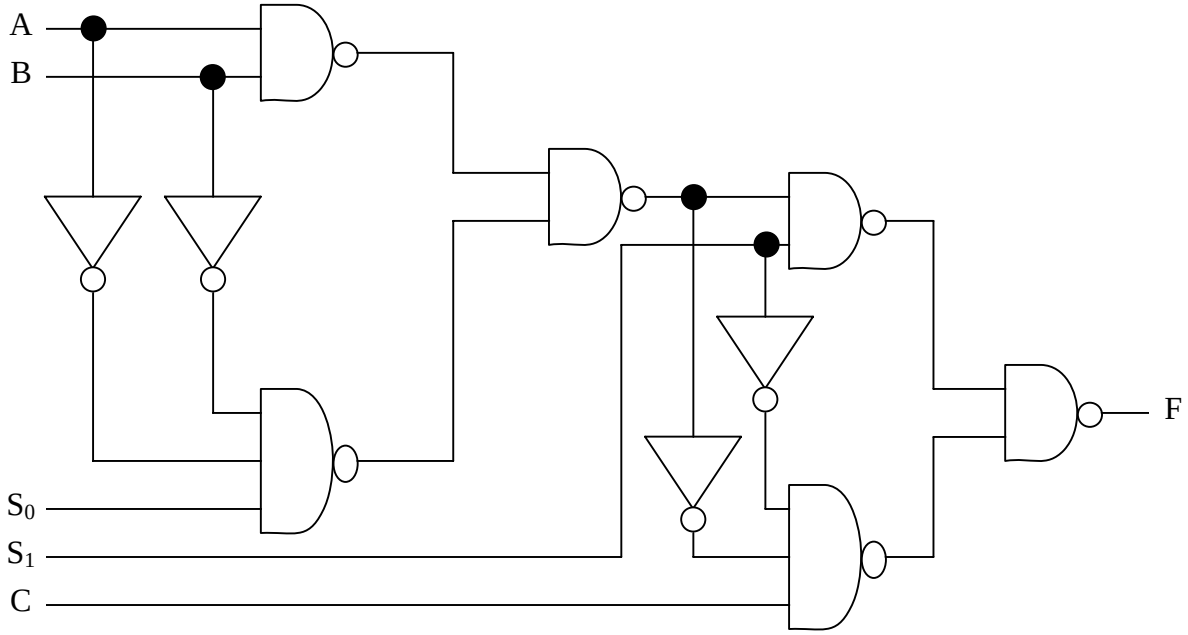
7 Réaliser un circuit logique ayant deux entrées A et B, une sortie S et respectant le chronogramme suivant :



8 Donner le schéma logique avec des porte OR, AND et NOT permettant de réaliser la fonction logique : $F = \overline{A}.C + \overline{B}.\overline{D}$

exercice 1.2

Considérons le circuit ci-dessous :



Ce circuit réalise la fonction F qui peut être contrôlée par les variables logiques S_0, S_1 et C. On veut analyser la variation de F en fonction des valeurs de S_0, S_1 et C et l'exprimer en fonction de A et B.

1. Etablir la forme générale de F en fonction des variables A, B, C, S_0, S_1. On appliquera le théorème de DE MORGAN pour obtenir F sous forme d'une somme de produits.

2. En déduire la table de vérité de F en fonction de S_0, S_1 et C.
3. En déduire enfin les valeurs respectives de S_0, S_1 et C pour que la fonction F représente $A.B$, $\overline{A.B}$, $A \oplus B$, $\overline{A \oplus B}$.

exercice 1.3

On désire réaliser un générateur de parité P basé sur le principe suivant. P vaut 1 quand dans un mot de 4 bits (D, C, B, A) le nombre de 1 est pair, sinon P vaut 0.

1. Etablir la table de vérité de cette fonction. On considérera que 0 fait partie des nombres dont la parité est paire.
2. Implanter cette fonction avec 3 ou exclusif et un inverseur.

exercice 1.4

Simplifier les fonctions suivantes en utilisant les tableaux de Karnaugh : $Y_1 = (A + B).\overline{A} + \overline{A}.\overline{B}$, $Y_2 = \overline{A}.\overline{B} + A.\overline{B} + \overline{A}.B$, $Y_3 = A + B + \overline{A}.\overline{B}$.

exercice 1.5

Simplifier les fonctions suivantes en utilisant les tableaux de Karnaugh. Les implanter ensuite avec des portes NAND, puis des portes NOR.

$$Y_1 = \overline{A}.\overline{B} + \overline{A}.\overline{C} + B.\overline{C}$$

$$Y_2 = A.B.\overline{C} + \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + \overline{A}.\overline{B}.C + \overline{A}.B.C$$

$$Y_3 = A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.C.\overline{D} + A.B.\overline{C}.D + A.B.C.D + \overline{A}.B.\overline{C}.D + \overline{A}.B.C.D + \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D}$$

$$Y_4 = \overline{B}.\overline{C} + A.\overline{D} + \overline{A}.\overline{B}.\overline{D} + \overline{A}.\overline{B}.C.D + A.\overline{B}.C.D$$

$$Y_5 = (A + B + \overline{C}).(A + \overline{B} + C).(\overline{A} + \overline{B} + C).(A + \overline{B} + \overline{C})$$

$$Y_6 = (\overline{B} + C).(A + \overline{C}).(\overline{A} + \overline{C})$$

$$Y_7 = (A + \overline{B}).(A + B + \overline{C}).(\overline{A} + \overline{B} + C).(\overline{A} + \overline{B} + \overline{C} + \overline{D}).(\overline{A} + \overline{B} + \overline{C} + D)$$

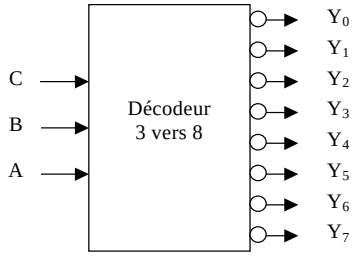
exercice 1.6

1. Quel est le nombre décimal le plus petit et le plus grand que l'on peut coder avec 4 bits, 8 bits, 16 bits, 32 bits, N bits en non signé et en CA2.
2. Convertissez $(1101101)_2$ en décimal.
3. Convertissez $(19)_{10}$ et $(45)_{10}$ et $(63)_{10}$ en binaire.

4. Convertissez $(11100101001010111)_2$ en hexadécimal.
5. Convertissez $(10A4)_{16}$ et $(CF8E)_{16}$ et $(9742)_{16}$ en binaire et en décimal.

exercice 1.7

On souhaite faire la synthèse d'un décodeur 3 vers 8 avec les sorties actives au niveau bas.



1. Etablir la table de vérité du circuit.
2. Déterminer les fonctions de sortie $Y_i = f(C, B, A)$.
3. Donner une implantation avec des portes NAND.
4. Comment faut-il modifier le schéma pour ajouter au montage une entrée de validation $\overline{V}$ telle que le circuit fonctionne normalement quand $\overline{V}=1$ et que toutes les sorties $Y_i = 1$ quand $\overline{V}=0$?

exercice 1.8

L'utilisation pour la transmission de données numériques d'un code de 4 bits comme le code BCD ne permet pas la détection d'une simple erreur d'inversion car le mot erroné obtenu appartient généralement au code utilisé. Par exemple, en BCD, le mot $(2)_{10} = (0010)_2$ transmis avec une inversion du troisième bit (généré par un parasite sur la ligne) sera pris pour un $(6)_{10} = (0\mathbf{1}10)_2$ à la réception. Afin de réduire les erreurs de ce type, on utilise un code de détection des erreurs comme le code « 2 de 5 » qui représente les dix combinaisons possibles de deux 1 dans un mot de 5 bits. Le codage obtenu est le suivant :

Nombre décimal		Code 2 de 5				
		b_4	b_3	b_2	b_1	b_0
0		0	0	0	1	1
1		1	1	0	0	0
2		1	0	1	0	0

3		0	1	1	0	0
4		1	0	0	1	0
5		0	1	0	1	0
6		0	0	1	1	0
7		1	0	0	0	1
8		0	1	0	0	1
9		0	0	1	0	1

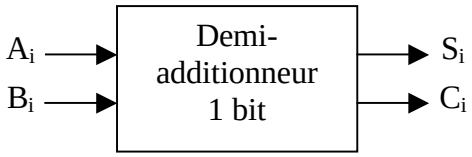
On se propose de réaliser un circuit combinatoire effectuant la conversion du binaire naturel en code « 2 de 5 ».

1. Compléter la table de vérité ci-dessus donnant le code en binaire nature a3 a2 a1 a0 correspondant à chaque nombre décimal.
2. Dresser le tableau de Karnaugh de chacune des fonctions b_4, b_3, b_2, b_1, b_0.
3. Déduire des tableaux précédents l'expression de chacune des fonctions et proposer une réalisation à base de fonctions logiques élémentaires.
4. Réaliser b0 sous une forme qui n'utilise que des portes NAND.

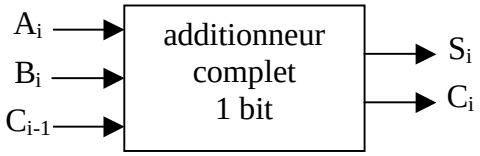
exercice 1.9

On cherche à réaliser un montage permettant d'effectuer l'addition ou la soustraction sur 1 bit avec retenue entrante et sortante.

1. Etablir la table de vérité et le schéma du demi-additionneur qui effectue l'opération $S_i = A_i + B_i$ et qui calcule la retenue sortante.



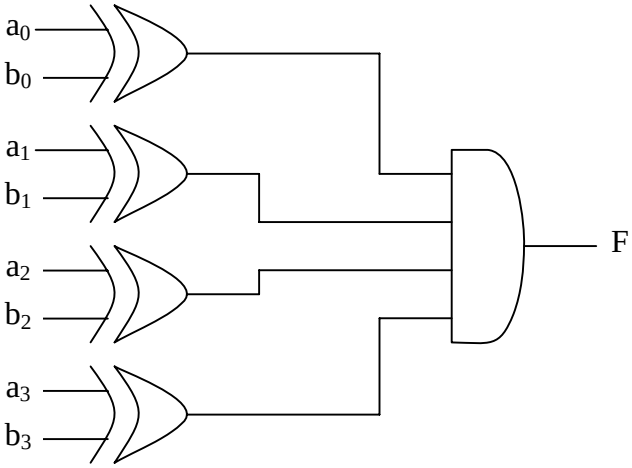
2. En déduire la table de vérité et le schéma de l'additionneur complet qui effectue l'opération $S_i = A_i + B_i + C_{i-1}$ et qui calcule la retenue sortante.



3. Même questions 1 et 2 pour le soustracteur.
4. En comparant l'étage additionneur et l'étage soustracteur 1 bit, proposer un montage unique qui, à l'aide d'une commande externe permet de réaliser soit l'addition, soit la soustraction.

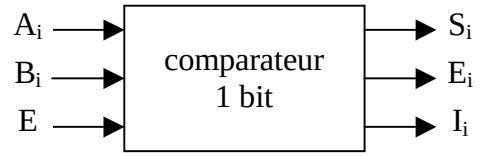
exercice 1.10

Analyser le circuit ci-dessous et définir son rôle.



exercice 1.11

1. Donner l'implantation d'un comparateur 1 bit pourvu d'une entrée E autorisant la comparaison. Si E = 0, toutes les sorties valent 0, sinon le fonctionnement est le suivant :



$S_i = 1$ si $a_i > b_i$, 0 sinon.

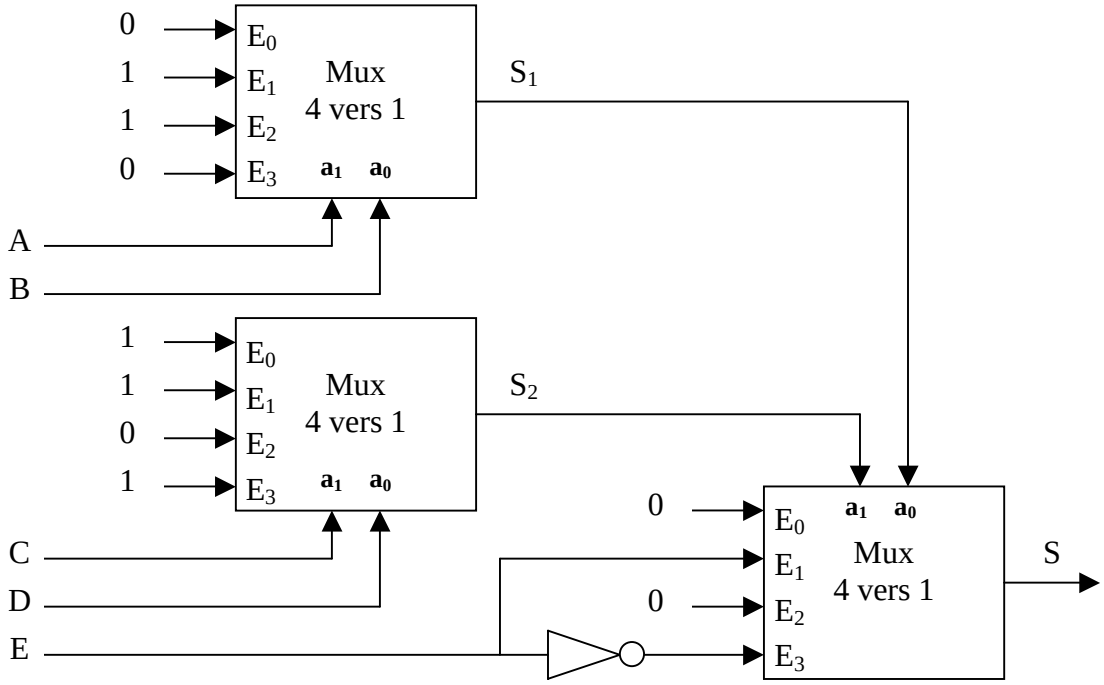
$E_i = 1$ si $a_i = b_i$, 0 sinon.

$I_i = 1$ si $a_i < b_i$, 0 sinon.

2. Donner l'implantation d'un comparateur de deux nombres A = $a_2a_1a_0$ et B=$b_2b_1b_0$ en utilisant des comparateurs 1 bit et des portes OR.

exercice 1.12

Donnez l'équation logique S=f(A, B, C, D, E) du montage suivant :



exercice 1.13

Réaliser un multiplexeur 32 vers 1 à partir de 4 multiplexeurs 8 vers 1 et d'un multiplexeur 4 vers 1.

exercice 1.14

Un appareil comporte trois cuves contenant de l'eau, du cassis et de la menthe. Trois boutons e, m, c commandant des électrovannes E, M, C permettent d'obtenir de l'eau pure, de la menthe à l'eau ou du cassis à l'eau. Une pièce P doit être introduite sauf pour l'eau pure qui est gratuite. Le déclenchement d'un bouton quelconque e, m, c ou l'introduction de la pièce

déclenche une temporisation. Si celle-ci arrive à son terme avant qu'un choix cohérent ait été fait, la pièce éventuellement introduite est rendue (fonction P de restitution). La pièce est également rendue en cas de fausse manœuvre.

1. Ecrire les équations logiques de commande des électrovannes E, M, C et la fonction de retour de la pièce P en fonction des variables e, m, c et p. On ne tiendra pas compte de la temporisation.
2. Simplifier les équations logiques à l'aide des tableaux de Karnaugh.
3. Réaliser la fonction à l'aide d'un décodeur 3 vers 8 et de portes logiques.
4. Réaliser la fonction à l'aide de double multiplexeurs 4 entrées.
5. Réaliser la fonction à l'aide d'une PROM 32x8.
6. Réaliser la fonction à l'aide de portes NAND.

exercice 1.15

Un système reçoit, codés en binaire naturel, des nombres compris entre 0 et 20 (inclus). Les digits d'entrée sont par poids d'ordre croissant A, B, C, D et E. Le système délivre trois informations S, T et U :

- La sortie S vaut 1 lorsqu'un nombre divisible par trois se présente à l'entrée.
- La sortie T vaut 1 lorsqu'un nombre divisible par cinq se présente à l'entrée.
- La sortie U vaut 1 lorsqu'un nombre divisible par sept se présente à l'entrée.

1. Donner la table de vérité du système.
2. Simplifier les équations logiques à l'aide des tableaux de Karnaugh.
3. Réaliser les fonctions T et U à l'aide de portes NAND.
4. Réaliser les fonctions S, T, U à l'aide de multiplexeurs 16 entrées.
5. Réaliser les fonctions S, T, U à l'aide d'une PROM 32x8.

exercice 1.16

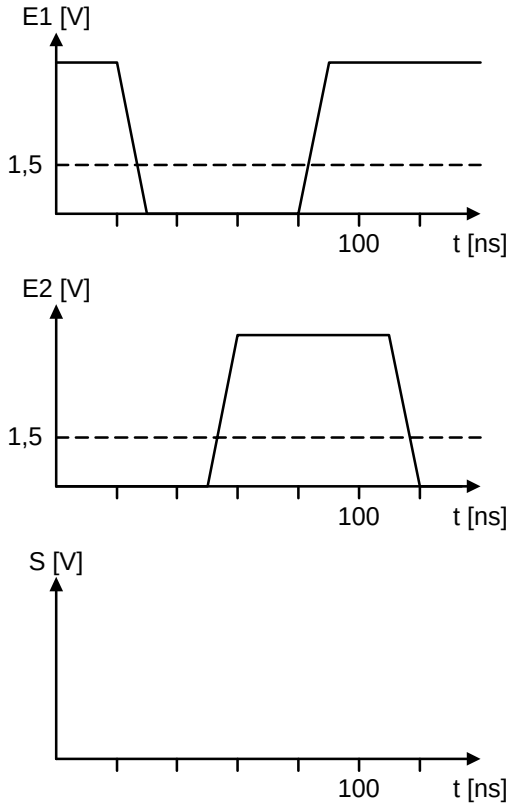
On souhaite réaliser un comparateur travaillant sur deux bits. Il possède deux entrées sur deux bits appelées AB et CD et 4 sorties : AB = CD (EQ), AB ≠ CD (NE), AB < CD (LT) et AB > CD (GT).

1. Donner la table de vérité du circuit.
2. Simplifier les équations logiques à l'aide des tableaux de Karnaugh.
3. Réaliser la fonction à l'aide de portes NAND.

4. Réaliser la fonction à l'aide de multiplexeurs 8 entrées.
5. Réaliser la fonction à l'aide d'un décodeur 4 vers 16 et de portes logiques.
6. Réaliser la fonction à l'aide d'une PROM 32x8.

exercice 1.17

Soit le circuit NAND SN74LS00 (voir les caractéristiques en annexe). Compléter le chronogramme suivant.



exercice 1.18

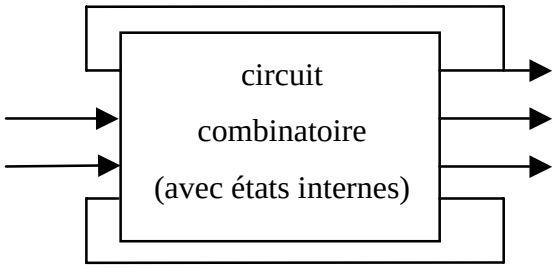
Soit un circuit inverseur avec trigger de Schmitt SN74LS14. Ses tensions de seuil de basculement typiques positives et négatives sont les suivantes : V_{T+} = 1,6 V, V_{T-} = 0,8 V.

1. Dessiner sa fonction de transfert Vs = f(Ve).
2. Définir et donner la valeur de son hystérésis.

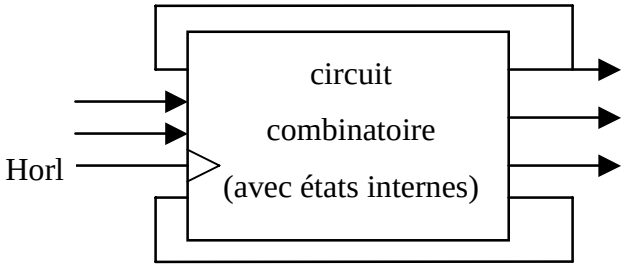
2. Logique séquentielle

Dans un circuit combinatoire, une sortie est uniquement fonction des entrées. Par contre, dans un circuit séquentiel, une sortie est une fonction des entrées mais aussi des sorties du circuit. Il y a rebouclage (rétroaction) des sorties sur les entrées. Cela signifie qu'un circuit séquentiel garde la mémoire des états passés. Il existe deux grandes catégories de circuit séquentiel :

- Le circuit séquentiel asynchrone. Les sorties du montage peuvent changer à tout moment dès qu'une ou plusieurs entrées changent après un temps de propagation qui peut être différent pour chaque sortie.

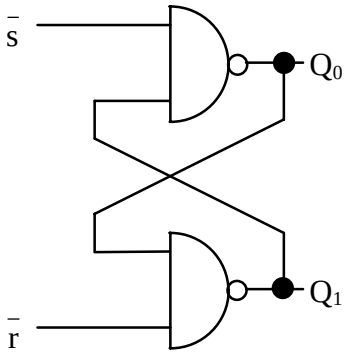


- Le circuit séquentiel synchrone. Le changement sur les sorties se produit après le changement d'état (front montant ou descendant) d'un signal maître, l'horloge. Les entrées servent à préparer le changement d'état, mais ne provoquent pas de changement des sorties. Tout changement d'état interne du montage est synchronisé sur le front actif de l'horloge.



2.1 Circuits séquentiels asynchrones

La forme la plus élémentaire de circuit séquentiel, que l'on appelle un latch (verrou), est la suivante :



$$Q_0 = \overline{\bar{s}.Q_1} = s + \overline{Q_1}\,,\ Q_1 = \overline{\bar{r}.Q_0} = r + \overline{Q_0}$$

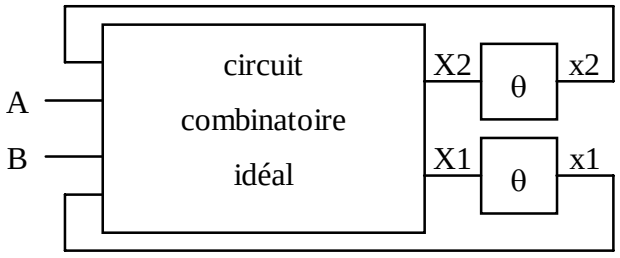
$$\Rightarrow Q_0 = s + \bar{r}.Q_0,\ Q_1 = r + \bar{s}.Q_1$$

Sa table de vérité est :

$\bar{s}$	$\bar{r}$	Q_0	Q_1
0	0	interdit	interdit
0	1	1	0
1	0	0	1
1	1	Q_0	Q_1

L'état 0,0 est interdit car il conduit à un état instable comme nous le verrons par la suite. Nous allons appliquer la méthode des tables d'excitation pour analyser dans le détail le fonctionnement de ce circuit.

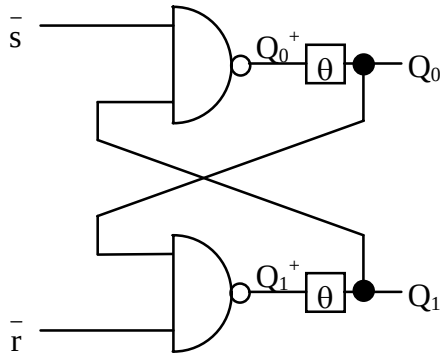
Tout circuit logique asynchrone comportant des boucles de réaction possède un fonctionnement séquentiel. On peut modéliser un circuit séquentiel en ajoutant des retards dans les boucles de réaction. Bien que les retards physiques, fonctions des temps de propagation à travers les portes élémentaires, ne soient pas égaux, on convient de symboliser la fonction « retard » par une même valeur θ pour l'étude des circuits asynchrones.



On définit :

- Les variables primaires A et B. Ce sont les entrées réelles du circuit.
- Les variables secondaires (ou variable interne) x1 et x2. Ce sont les sorties réinjectées sur l'entrée du circuit combinatoire idéal.
- Les variables d'excitation X1 et X2. Ce sont les sorties du circuit combinatoire idéal sur lesquelles a été placé un retard.

Dans le cas du latch, on obtient le schéma asynchrone équivalent :



$\bar{s}\,\bar{r}$ sont les variables primaires, $Q_0\ Q_1$ sont les variables secondaires et $Q_0^+\ Q_1^+$ sont les variables d'excitation. On établit la table d'excitation donnant $Q_0^+,Q_1^+=F(\bar{s},\bar{r},Q_0,Q_1)$ avec les équations combinatoires $Q_0^+ = \bar{s}.\overline{Q_1}$ et $Q_1^+ = \bar{r}.\overline{Q_0}$.

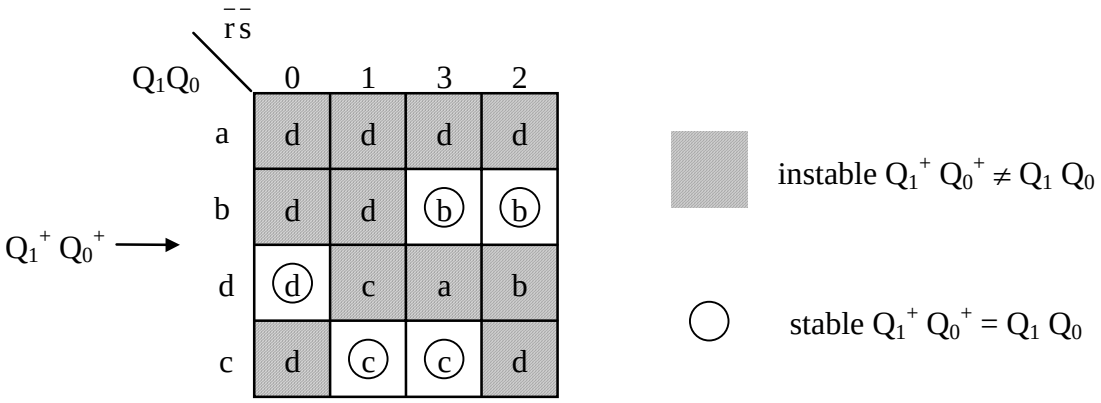
$Q_1^+ Q_0^+$; $Q_1 Q_0$ \ $\bar{r}\,\bar{s}$	0 0	0 1	1 1	1 0
00	11	11	11	11
01	11	11	01	01
11	11	10	00	01
10	11	10	10	11

Puis on change de notation pour obtenir une table d'états internes :

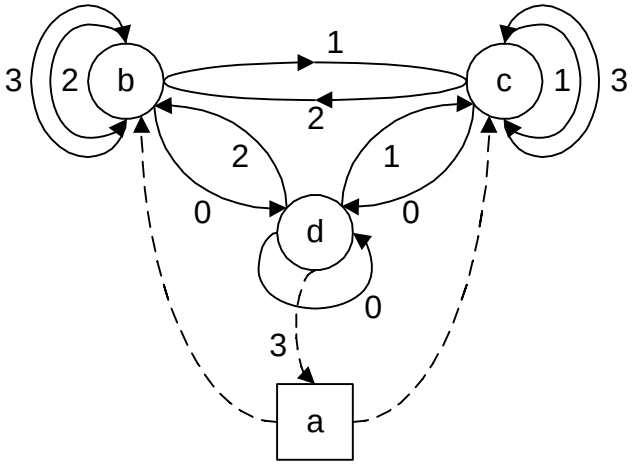
$\bar{s}$	$\bar{r}$	notation
0	0	0
0	1	1
1	0	2
1	1	3

Q_0	Q_1	notation
0	0	a
0	1	b
1	0	c
1	1	d

La table d'états internes obtenue est :



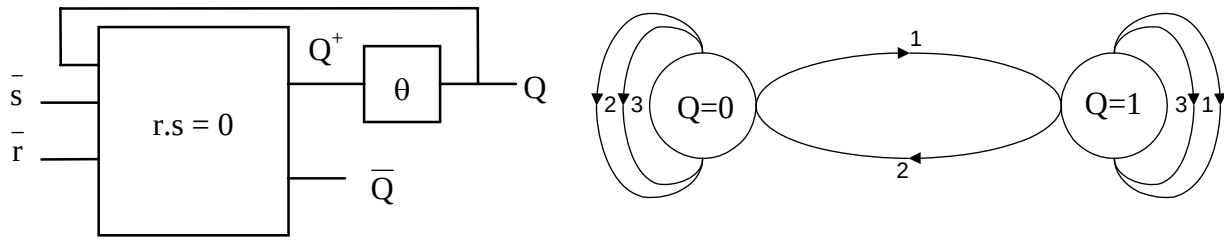
On voit que l'état interne (a) est toujours instable. A partir de l'état (b), on cherche sur la table des états internes les effets de toutes les commandes, puis on recommence pour les états (c) et (d). On obtient ainsi le graphe d'évolution du montage.



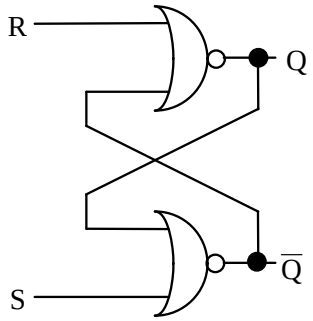
En (d), si $\bar{s}\,\bar{r}=3$, on va en (a), état instable. On a deux choix possibles : vers l'état b ou l'état c. L'état choisi dépendra de la vitesse des portes (des valeurs de θ), la porte la plus rapide passant à 0 en premier. Dans un circuit séquentiel, quand plusieurs changements d'états internes se produisent pour un seul changement sur les commandes, alors on dit qu'il se produit une « race condition ». Quand l'état final ne dépend pas de l'ordre des changements d'états internes, alors il s'agit d'une « noncritical race ». Quand l'état final du circuit dépend de l'ordre des changements d'états internes, alors il s'agit d'une « critical race ». C'est le cas pour le latch $\bar{s}\,\bar{r}$. Il faut toujours s'assurer qu'aucune « critical race » ne peut se produire dans ce type de circuit séquentiel avec rétroaction combinatoire car son comportement devient alors totalement imprédictible. L'interprétation du graphe est donc la suivante :

- L'état (d) est forcé par 0 et gardé en mémoire par 0.
- L'état (b) est forcé par 2 et gardé en mémoire par 3 ou 2.
- L'état (c) est forcé par 1 et gardé en mémoire par 3 ou 1.
- Les commandes sont équivalentes à :
  0. Commande interdite. On évite ainsi le problème dû à la commande 3 sur (d). On impose r.s = 0 à l'entrée de la bascule, ce qui supprime la « critical race ».
  1. Remise à un ⇒ état (c).
  2. Remise à zéro (RAZ) ⇒ état (b).
  3. Mémoire.

Le modèle final du latch valable si r.s = 0 et son graphe d'évolution sont :



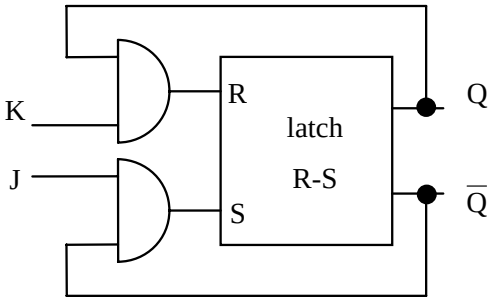
avec l'équation d'excitation : $Q^+ = s + \bar{r}.Q$. On peut réaliser une variante de ce latch avec des portes NOR, le latch RS :



L'équation ne change pas : $Q^+ = S + \overline{R}.Q$ avec R.S = 0. La table de vérité est :

R	S	Q^+	fonction
0	0	Q	mémoire
0	1	1	mise à 1
1	0	0	mise à 0
1	1	X	interdit

Afin de s'affranchir de l'état instable, on définit le latch JK de la manière suivante :



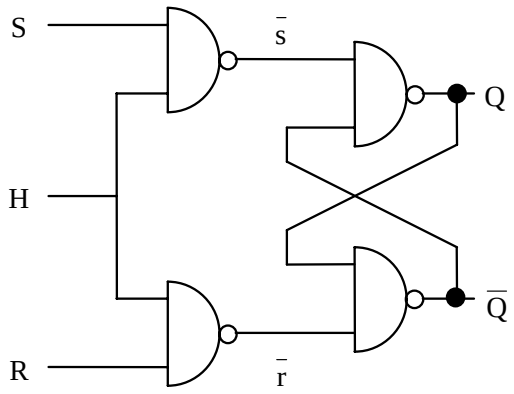
On garantit de cette manière que l'état R = 1, S = 1 ne se produira jamais en posant R = K.Q et $S = J.\overline{Q}$ ce qui nous donne l'équation de sortie suivante : $Q^+ = J.\overline{Q} + \overline{K}.Q$. La table de vérité vaut alors :

J	K	Q^+	fonction
0	0	Q	mémoire
0	1	0	mise à 0
1	0	1	mise à 1
1	1	$\overline{Q}$	inversion

L'état instable du latch RS s'est transformé en un état inversion.

2.2 Bistables synchronisés sur un niveau

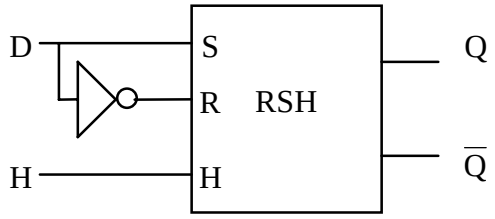
Nous allons maintenant essayer de synchroniser le fonctionnement du latch avec un signal d'horloge à l'aide du schéma :



En respectant r.s = 0, on a $Q^+ = s + \bar{r}.Q$, d'où on tire $Q^+ = S.H + \overline{R.H}.Q$ avec (R.H).(S.H) = R.S.H = 0. On obtient la table de vérité suivante :

R	S	H	Q_{n+1}	fonction
X	X	0	Q_n	mémoire
0	0	1	Q_n	mémoire
0	1	1	1	mise à 1
1	0	1	0	mise à 0
1	1	1	X	interdit

Quand H vaut 1, le bistable fonctionne normalement. Quand H vaut 0, il garde en mémoire l'état antérieur. On appelle ce circuit un bistable RSH. Il est synchronisé sur le niveau de l'horloge H. Il en existe une variante importante : le bistable DH.

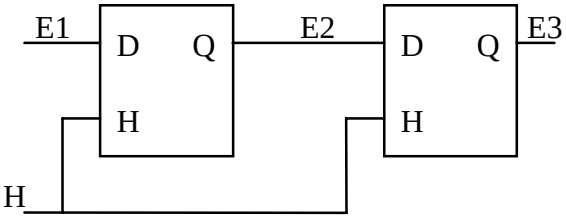


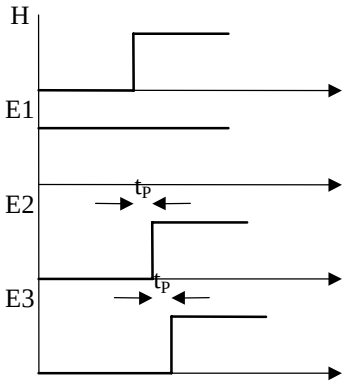
On pose S = D et R = $\overline{D}$ ce qui implique : D = 0 $\Rightarrow$ S = 0, R = 1 donc mise à 0 et D = 1 $\Rightarrow$ S = 1, R = 0 donc mise à 1. Ce bistable ne présente plus de combinaison interdite d'entrée car on a toujours R.S = 0. Il est appelé **latch transparent** et réalise la fonction $Q_{n+1} = \overline{H}.Q_n + H.D$:

- l'état copie, Q_{n+1}= D pour H = 1.
- l'état verrou, Q_{n+1}= Q_n pour H= 0.

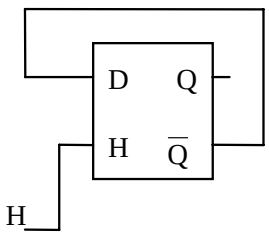
On n'exploite Q que pendant la phase verrou. C'est là, la principale limitation des bistables. En effet, les montages suivants sont interdits :

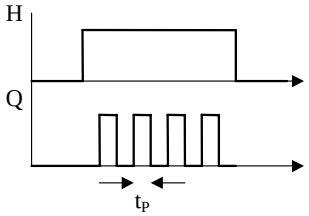
- Association synchrone en cascade. L'état à l'entrée de la chaîne se propage presque instantanément sur la sortie (après un temps de propagation t_p).





- Rétroaction. Le montage oscille.

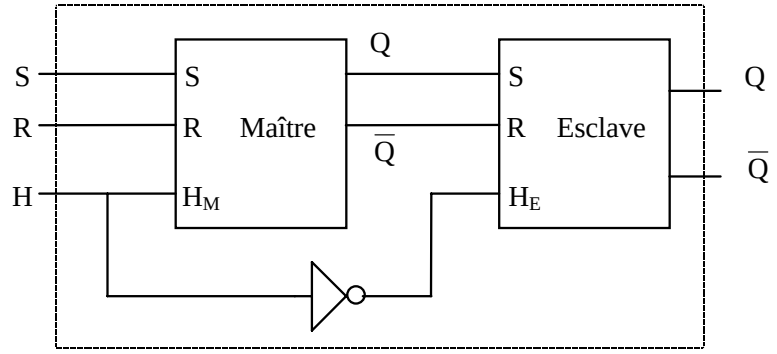




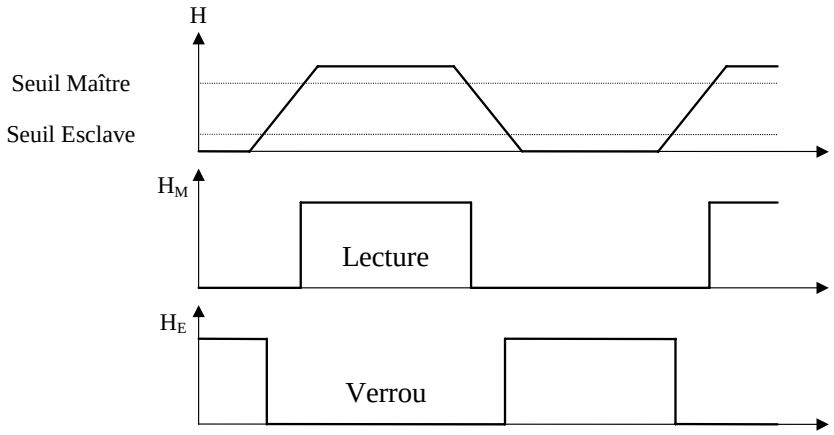
Les bistables sont encore utilisés, notamment les latches transparents, mais il est nécessaire de définir un nouveau circuit pour travailler en logique synchrone : c'est la bascule.

2.3 Bascules maître-esclave

Une bascule est un bistable qui ne change d'état qu'une seule fois par période d'horloge. Le premier montage de ce type a été la bascule maître-esclave qui n'est plus aujourd'hui utilisée. Sa structure est la suivante :



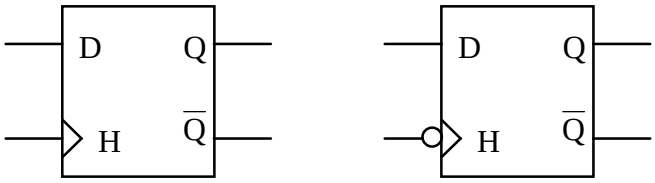
En réglant habilement les seuils de basculement d'horloge du bistable maître et du bistable esclave, on obtient le chronogramme suivant :



La commande SR est lue sur le niveau haut de H et elle est exécutée sur son front descendant. Ce type de bascule règle les deux problèmes vus au paragraphe précédent, l'association en cascade et la rétroaction. Toutefois, elle reste sensible aux parasites car la bascule maître accepte la commande pendant la totalité du niveau haut de l'horloge. On va donc définir un type de bascule qui n'accepte la commande que pendant le front actif de l'horloge.

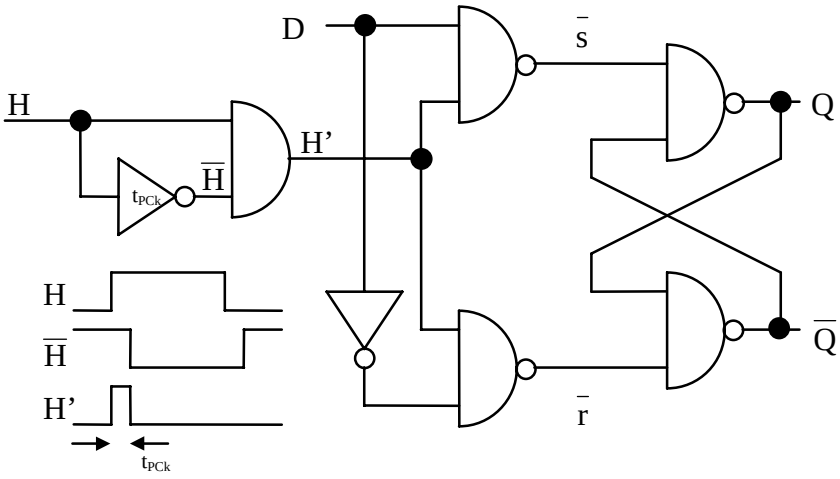
2.4 Bascules synchronisées sur un front

Ces bascules font l'acquisition de la donnée et réalisent la commande sur un front d'horloge. Ce sont les bascules actuellement utilisées pour la conception en logique synchrone. Elles peuvent être actives sur le front descendant ou sur le front montant de l'horloge.



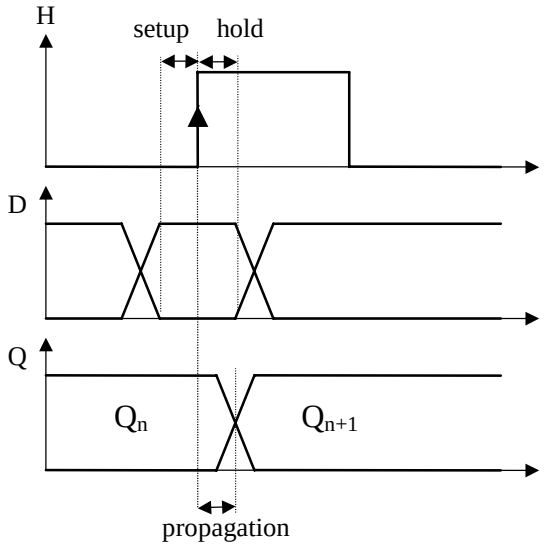
Front montant Front descendant

D'un point de vue purement théorique (il ne s'agit pas du montage utilisé réellement), on peut voir une bascule D commandée par un front montant de la manière suivante :



En dosant de manière adéquate le temps de propagation t_{PCK} de l'inverseur, on obtient en H' une impulsion juste assez large pour permettre le transfert de l'information D vers le latch RS.

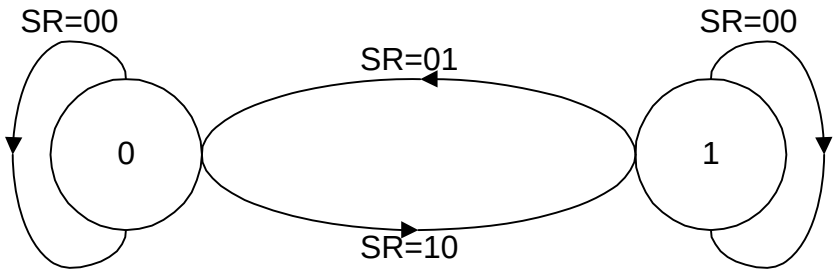
On voit bien avec ce montage que pour un fonctionnement correct de la bascule, la donnée doit être présente un certain temps (setup) avant le début de l'impulsion (le front actif) et rester stable un certain temps après (hold). La donnée en sortie ne peut être exploitée qu'après un temps de propagation.



2.5 Bascules usuelles

Il existe quatre types de bascules usuelles :

- La bascule RS. Son équation de sortie est $Q_{n+1} = S + \overline{R}.Q_n$ avec S.R = 0. Son graphe d'évolution est :



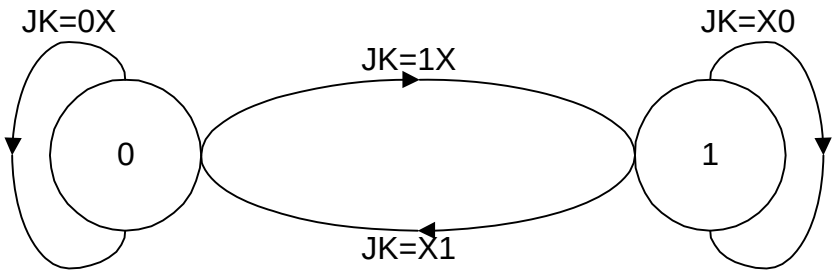
La bascule RS synchrone n'existe pas sous la forme de composant discret et n'est jamais utilisée directement dans un montage. Toutefois, sa structure se retrouve dans toutes les autres bascules. Sous la forme du latch RS, elle sert à réaliser des interrupteurs anti-rebonds. On obtient la table de vérité :

R	S	H	Q_{n+1}	fonction
0	0	↑	Q_n	mémoire
0	1	↑	1	mise à 1
1	0	↑	0	mise à 0
1	1	↑	X	interdit

- La bascule JK. Son équation de sortie vaut $Q_{n+1} = J.\overline{Q_n} + \overline{K}.Q_n$ ce qui donne la table de vérité :

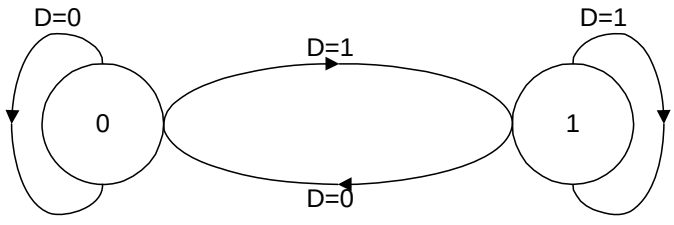
J	K	H	Q_{n+1}	fonction
0	0	↑	Q_n	mémoire
0	1	↑	0	mise à 0
1	0	↑	1	mise à 1
1	1	↑	$\overline{Q_n}$	$\overline{\text{mémoire}}$

Son graphe d'évolution est :



X est un état indifférent (don't care). Cette bascule permet la réalisation de montages ayant un minimum de portes combinatoires. Elle nécessite toutefois un câblage plus complexe qu'une bascule D car elle a deux entrées alors que la bascule D n'en a qu'une.

- La bascule D. Elle est obtenue en posant D = S = $\overline{R}$ avec une bascule RS ou D = J = $\overline{K}$ avec une bascule JK, ce qui donne l'équation Q_{n+1} = D et le graphe d'évolution :



Sa table de vérité est :

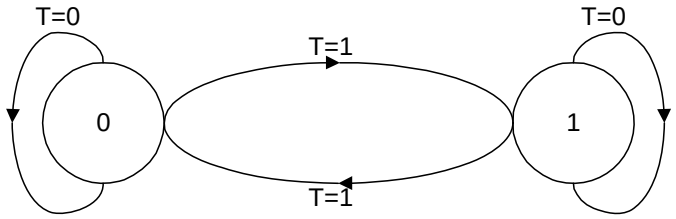
D	H	Q_{n+1}	fonction
0	↑	0	mise à 0
1	↑	1	mise à 1

Cette bascule existe sous forme de composant discret et permet la réalisation de montages nécessitant un minimum de câblage (car elle n'a qu'une entrée) mais plus de portes combinatoires qu'avec des bascules JK. Toutefois, comme le problème de la longueur du câblage est très important dans les circuits VLSI, **la bascule D est la seule utilisée** dans les circuits programmables et dans les ASIC.

- On peut noter une variante de la bascule JK, la bascule T pour laquelle on pose J = K = T (T pour toggle). Son équation est $Q_{n+1} = T \oplus Q_n$ avec la table de vérité :

T	H	Q_{n+1}	fonction
0	↑	Q_n	mémoire
1	↑	$\overline{Q_n}$	$\overline{\text{mémoire}}$

Son graphe d'évolution est :



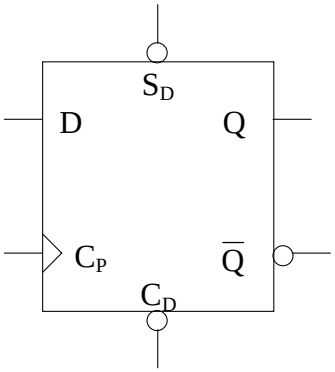
La bascule T n'existe pas sous forme de composant discret car elle est très facile à réaliser à partir d'une bascule JK ou d'une bascule D. Elle est particulièrement efficace dans la réalisation de compteurs binaires.

2.6 Caractéristiques temporelles des circuits séquentiels synchrones

Nous allons définir dans ce chapitre les intervalles de temps importants utilisés pour caractériser les circuits séquentiels synchrones.

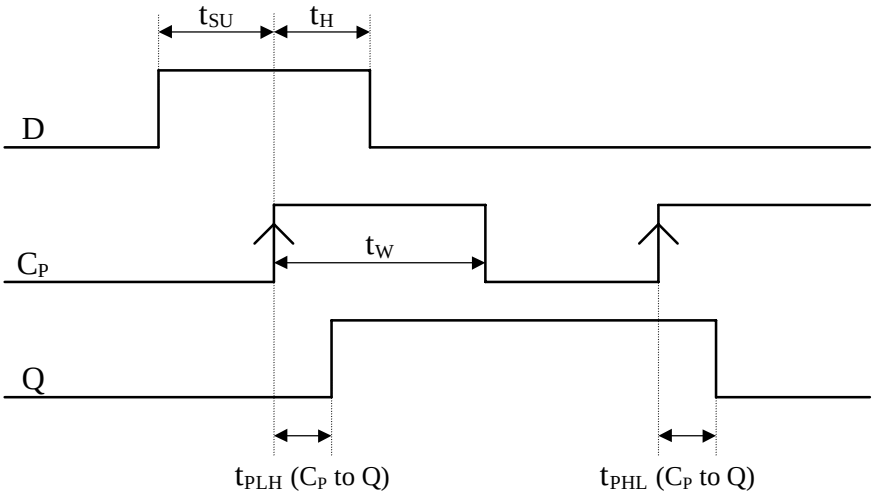
2.6.1 Définitions

Nous allons prendre l'exemple d'une bascule D de type SN74LS74 :

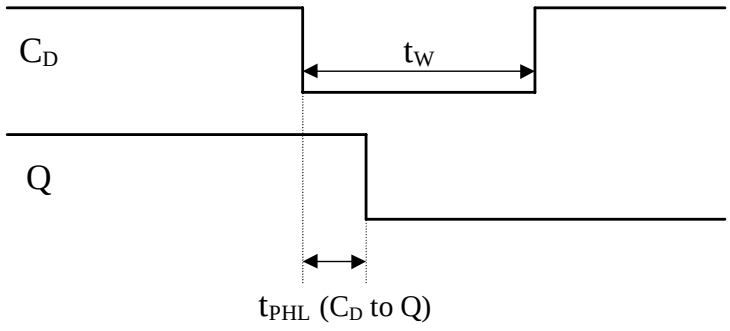


Les temps étudiés pour cette bascule se retrouveront (à quelques variantes près) dans pratiquement tous les autres circuits séquentiels. Les signaux à l'entrée d'un circuit séquentiel peuvent être classés en deux catégories :

- Les signaux à action synchrone. L'entrée D de la bascule est recopiée sur les sorties Q et $\overline{Q}$ après un temps de propagation t_{PLH} ou t_{PHL} au moment du front actif (ici le front montant) de l'horloge (notée C_K, C_P ou H). La donnée doit être présente sur l'entrée D un temps t_{SU} (setup time) avant le front actif et être maintenue un temps t_H (hold time) après ce front. L'impulsion active de l'horloge (ici l'impulsion positive) doit avoir une durée minimale t_W (width time) pour être prise en compte par la bascule.



- les signaux à action asynchrone. Les signaux de mise à 0 C_D (reset ou clear) et de mise à 1 S_D (set) ont une action immédiate. Ils ne sont pas synchronisés sur le front actif de l'horloge. Ces signaux sont actifs sur un niveau (ici le niveau 0). Tant que le niveau actif est maintenu, l'effet de la commande persiste. Le niveau actif de l'impulsion doit avoir une durée minimale t_W. Le dessin suivant donne un exemple de chronogramme pour le signal clear. Les temps de transitions ne sont pas représentés.

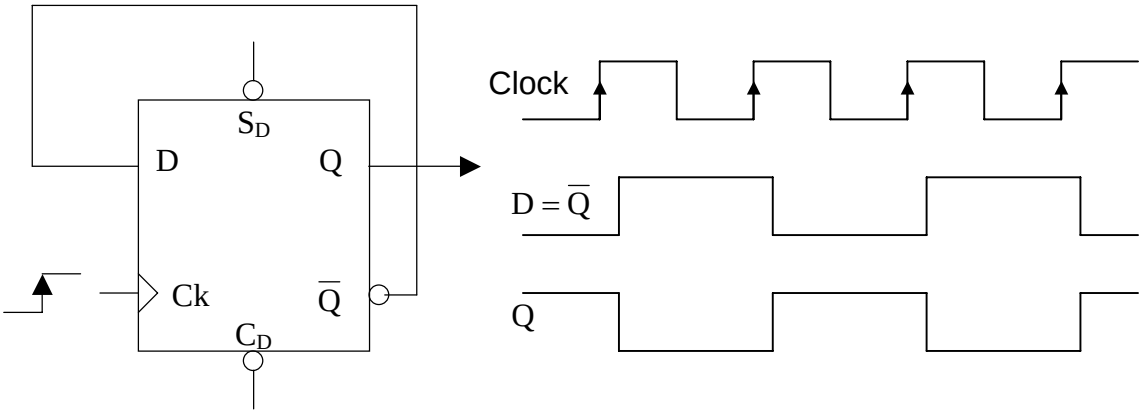


Afin de voir immédiatement comment agit un signal, il est pratique de respecter la notation suivante :

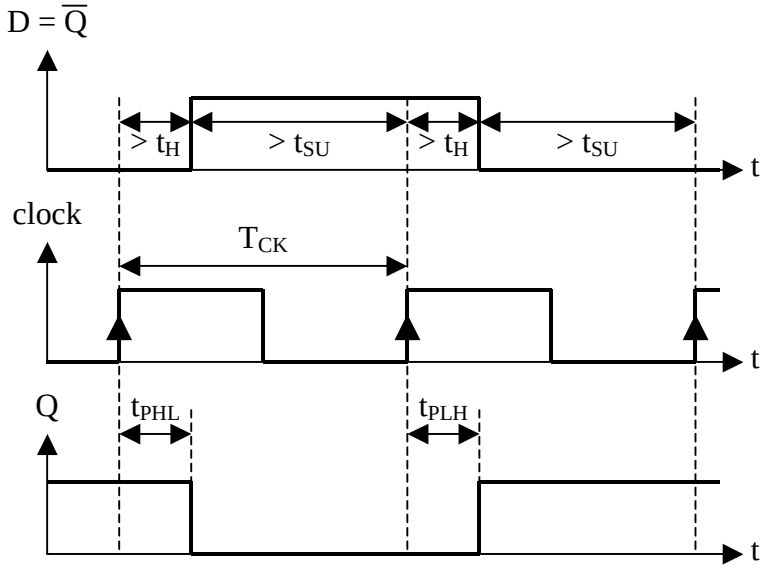
- le signal ayant un niveau actif à 0 est nommé $\overline{\text{SIGNAL}}$.
- le signal ayant un niveau actif à 1 est nommé SIGNAL.
- le signal réalisant deux fonctions (une sur chaque niveau) est nommé SIGNAL1/$\overline{\text{SIGNAL2}}$.

2.6.2 Calcul de la fréquence maximale d'horloge d'une bascule D

Il existe un autre paramètre utilisé pour caractériser un circuit séquentiel, la fréquence maximale de fonctionnement f_{MAX}. Ce paramètre est mesuré, dans le cas d'une bascule D, grâce au montage suivant :



Ce type de montage est appelé montage toggle car la sortie change d'état à chaque front actif d'horloge. la sortie Q est donc égale au signal d'horloge mais avec une fréquence divisée par 2. Agrandissons l'échelle des temps pour faire apparaître les différents timings de la bascule :



Pour que le montage fonctionne correctement, les paramètres du circuit doivent vérifier :

- $\mathbf{T_H < min(t_{PHL}, t_{PLH})}$. Cette relation ne dépend pas de la fréquence de l'horloge et elle est toujours vérifiée car on s'arrange pour que le temps de maintien de la bascule soit nul en fabrication. Cette relation n'intervient pas dans le calcul de la fréquence maximale du circuit.
- $\mathbf{T_{SU} < T_{CK} - max(t_{PHL}, t_{PLH})}$. Cette relation dépend de la fréquence d'horloge et elle permet de calculer la fréquence maximale du circuit. En effet, on remarque que pour un fonctionnement normal, la donnée D doit être présente t_{SU} avant le front montant de l'horloge. Or la donnée D est égale à la sortie $\overline{Q}$ qui apparaît un temps t_{PHL} ou t_{PLH} (C_P to

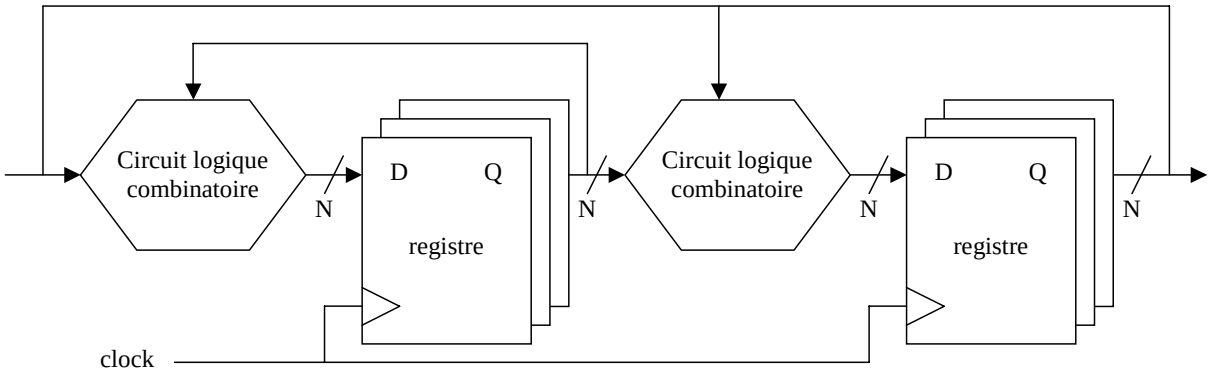
$\overline{Q}$) après le front montant de l'horloge. La période de l'horloge T_{ck} ne peut donc être inférieure à $t_{SU} + t_P$ ou t_p est le plus grand des temps de propagation clock to Q. Ce qui donne la relation : $T_{CK} > T_{SU} + \max(t_{PHL}, t_{PLH})$. La fréquence maximale de fonctionnement de la bascule (toggle rate) est donc définie par :

$$f_{max} < \frac{1}{t_{SU} + \max(t_{PHL}, t_{PLH})}$$

Hélas, la valeur de f_{max} n'est pas toujours prise dans ces conditions dans les feuilles de caractéristiques (data sheet) et la plus grande prudence s'impose dans l'exploitation de cette donnée. Une lecture attentive des caractéristiques constructeurs est nécessaire mais pas toujours suffisante pour déterminer la fréquence maximale de fonctionnement d'un circuit séquentiel synchrone.

2.6.3 Calcul de la fréquence maximale d'horloge dans le cas général

Dans le cas général, un circuit logique séquentiel synchrone peut toujours être mis sous la forme de registres (N bascules D fonctionnant en parallèle) reliées par de la logique combinatoire :



Certaines sorties de registres sont rebouclées sur des entrées. Il y a des entrées pures et des sorties pures. Dans un montage réel, le nombre de bascules peut s'élever à plusieurs dizaines ou centaines de milliers. Comment calcule-t-on la fréquence maximale de fonctionnement d'un tel « monstre ». En fait, c'est très simple. Il suffit de reprendre la formule vue précédemment et de lui ajouter le temps de propagation dans la logique combinatoire (t_{prop}). On obtient donc :

$$f_{max} < \frac{1}{t_{SU} + t_{prop} + \max(t_{PHL}, t_{PLH})}$$

ce qui traduit le fait que la donnée doit être stable t_{su} avant le front actif suivant de l'horloge. Toute la question est : quel temps de propagation doit-on choisir ? On ne peut pas le calculer à la main dans le cas général. Un outil de CAO va calculer tous les temps de propagation de toutes les sorties de bascules D vers toutes les entrées de bascules D. Le chemin le plus long, **le chemin critique**, va donner le temps le plus long, **le temps critique**. C'est ce temps qui va déterminer f_{max}.

$$f_{max} < \frac{1}{t_{SU} + t_{critique} + \max(t_{PHL}, t_{PLH})}$$

La fréquence la plus élevée d'un montage de ce type est égale à la fréquence maximale de fonctionnement d'une bascule D (qui ne dépend que de la technologie de fabrication utilisée pour construire la bascule). C'est le cas où le temps critique est nul, ce qui veut dire que le montage ne sert pas à grand chose étant donné qu'il ne comporte pas de logique combinatoire.

La fréquence maximale d'un montage réel est donc déterminée par le temps critique qui est fonction de la complexité du montage réalisé et du talent de l'ingénieur qui conçoit le circuit.

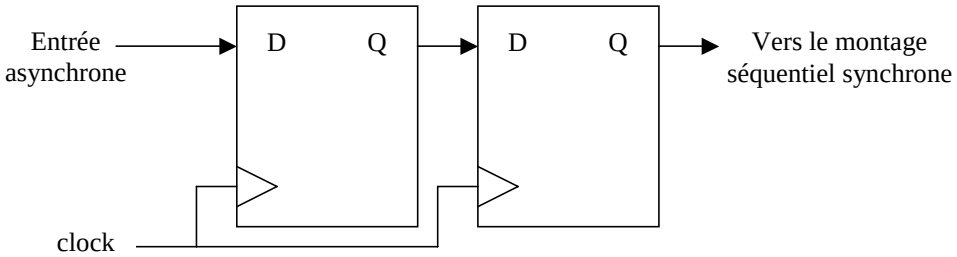
Il est important de noter que ce raisonnement ne vaut que pour un circuit logique séquentiel synchrone. C'est la première raison pour laquelle on n'utilise en pratique que ce type de montage dans les circuits logiques.

2.6.4 Métastabilité

Il reste un problème que nous avons soigneusement évité jusqu'à maintenant. Que se passe-t-il si le montage ne respecte pas le temps de setup, c'est-à-dire si la donnée n'est pas stable t_{su} avant le front actif de l'horloge ?
En fait, c'est un problème très courant que l'on peut rencontrer dans deux cas :

1. Dépassement de la fréquence maximale du montage. Il s'agit là d'une erreur d'utilisation du composant.
2. Entrées asynchrones. A partir du moment où le montage lit une donnée extérieure avec une bascule D, il y aura forcément une violation du temps de setup à un moment ou à un autre. Par exemple, dans le cas d'un bouton poussoir actionné par un opérateur humain ou bien d'un capteur qui indique un dépassement de trop plein ou encore d'une liaison série (RS-232) venant d'un ordinateur. On ne voit pas très bien comment un être humain pourrait être synchronisé avec l'horloge du montage.

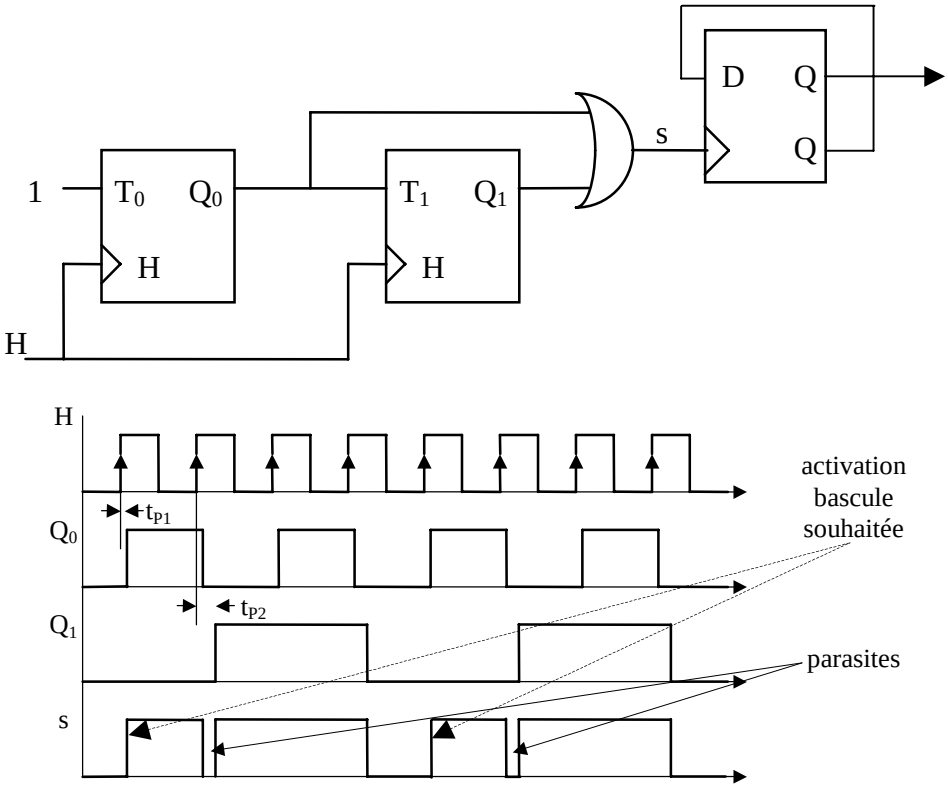
On peut faire beaucoup d'études savantes concernant le phénomène de métastabilité. Ce phénomène intervient quand la bascule hésite à changer d'état parce que la donnée n'est pas stable t_{su} avant le front actif de l'horloge. Elle se place alors dans un état intermédiaire entre le 0 et le 1, l'état « métastable ». En pratique, cela se traduit par un allongement du temps de propagation clock to Q de la bascule ; plus la violation du temps de setup est importante et plus le temps de propagation augmente. Bien sur, si la violation est trop importante le changement en sortie n'a pas lieu. Il est difficile de lutter contre la métastabilité car il s'agit d'un phénomène physique naturel inévitable. Une méthode simple et efficace consiste à effectuer une double (voir triple) synchronisation des entrées asynchrones d'un montage. Cela permet de minimiser l'importance du phénomène.



2.7 Règles de conception

2.7.1 Influence des aléas de commutation

Nous avons vu dans le chapitre sur la logique combinatoire que dans le cas général, il peut se produire un aléa de commutation (un glitch) en sortie de tout montage combinatoire sauf si celui-ci a fait l'objet d'une conception « hazard-free » extrêmement contraignante. Quelle va être l'influence de ce glitch dans un montage séquentiel ? Prenons un exemple simple :



Sur cet exemple, on voit bien l'impulsion parasite (le glitch) apparaître à la sortie de la porte OR. Il est dû à la différence de temps de propagation (clock to Q) entre les deux bascules T. Sa durée est indépendante de la fréquence de fonctionnement et totalement dépendante du processus de fabrication des bascules. Certains couples de bascules ne provoqueront pas de parasites, d'autres le feront. Si le signal S est utilisé pour attaquer l'entrée d'horloge d'une bascule, le glitch risque d'être vu comme un front supplémentaire et d'actionner la bascule. La seule solution efficace pour éviter ce genre de problème est d'utiliser la logique séquentielle synchrone.

2.7.2 Règles de conception synchrone

Les circuits séquentiels synchrones sont constitués de bascules synchronisées sur les mêmes fronts d'**une seule horloge** séparées par des couches de logiques combinatoires. Toutes les commandes sont prises en compte sur le front actif de l'horloge et les « hazards », s'il y en a, sont forcément terminés sur le front actif suivant de l'horloge (si bien entendu le montage respecte la fréquence maximale de fonctionnement). D'autre part, il ne peut y avoir de « race condition » puisqu'il n'y a pas de rétroactions combinatoires directes mais seulement sur les entrées des bascules. Le fonctionnement d'un circuit en logique synchrone est donc extrêmement fiable et portable d'une famille technologique à l'autre. De plus, sa vitesse

augmente linéairement avec l'accroissement de la vitesse des éléments qui le composent. L'amélioration de la densité d'intégration conduit donc automatiquement à une augmentation des performances.

Par construction, en logique séquentielle synchrone, les aléas de commutation peuvent être ignorés en toute sécurité car ils sont forcément terminés avant le front actif suivant de l'horloge. C'est la deuxième raison pour laquelle on n'utilise en pratique que ce type de montage dans les circuits logiques.

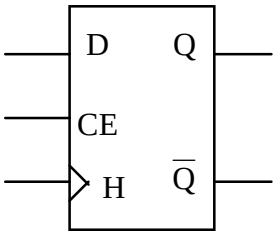
Toutefois, il est nécessaire pour éviter tout problème de respecter certaines règles :

1. Il ne faut pas attaquer une entrée d'horloge avec une combinaison de sorties de bascules. D'une manière plus générale, il faut traiter séparément le chemin des données et les lignes d'horloge.
2. Il ne faut pas utiliser les entrées asynchrones d'une bascule (entrée Clear et Preset par exemple) pour réaliser une réaction. Il est d'ailleurs conseillé de n'avoir que des entrées synchrones sur les bascules.

2.7.3 Le rôle du CE

Il reste toutefois un problème dont nous n'avons pas parlé. Si toutes les bascules sont activées par la même horloge, comment peut-on inhiber une bascule ou si vous préférez comment empêcher la copie de D sur Q sur le front actif de l'horloge ?

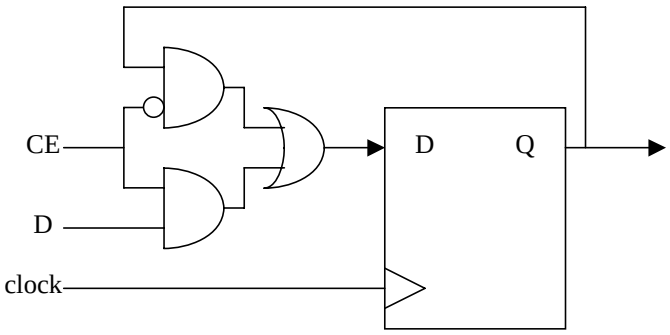
Il manque pour cela une entrée de validation sur nos bascules. C'est l'entrée CE ou « Chip Enable ». Sans cette broche supplémentaire, il est impossible de réaliser un montage séquentiel synchrone.



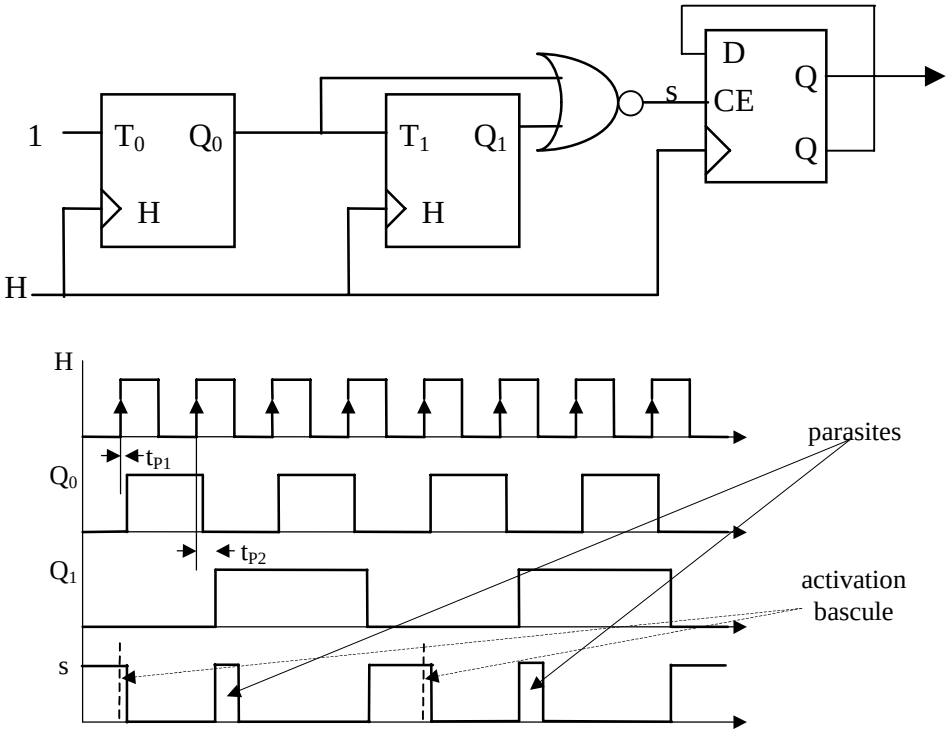
Le fonctionnement est très simple :

1. CE = 1 $\Rightarrow$ Q_{n+1} = D sur le front actif de l'horloge.
2. CE = 0 $\Rightarrow$ Q_{n+1} = Q_n sur le front actif de l'horloge.

En ce qui concerne la réalisation interne, il suffit d'ajouter un multiplexeur 2 vers 1 commandé par CE sur l'entrée D d'une bascule pour obtenir :



Nous pouvons maintenant reprendre et corriger le montage précédent :



L'équation logique de s a changé, mais la sortie du montage reste la même (vous pouvez vérifier que la bascule est activée au même moment). Les hazards sont toujours là, mais il ne nous gêne plus.

2.7.4 Asynchrone contre synchrone

En résumé, la logique séquentielle synchrone a les avantages suivants :

1. **Fiabilité** car les aléas de commutation n'interviennent pas dans le fonctionnement du montage.

2. **Calcul simple de la fréquence maximale de fonctionnement** du montage grâce à la formule suivante. Les outils de CAO savent calculer le temps critique du montage.

$$f_{max} < \frac{1}{t_{SU} + t_{critique} + \max(t_{PHL}, t_{PLH})}$$

3. **portabilité** d'une famille technologique à l'autre.
4. **La fréquence augmente linéairement** avec l'accroissement de la vitesse des éléments qui composent le montage. L'amélioration de la densité d'intégration conduit donc automatiquement à une augmentation des performances.

Comment définir la logique séquentielle asynchrone ? On la définit plutôt par opposition à la logique séquentielle synchrone. Tout montage qui ne respecte pas strictement les règles de conception synchrone est asynchrone. Pour le débutant, c'est hélas la solution qui vient généralement en premier. Elle est généralement caractérisée par l'absence d'horloge maîtresse et/ou par la présence de rebouclage séquentiel asynchrone. Lorsqu'elle est utilisée dans les règles, c'est-à-dire lorsqu'elle est conçue correctement et de manière fiable (*c'est-à-dire quasiment jamais sauf par des ingénieurs très expérimentés : il faut savoir que les outils de CAO marchent très mal en asynchrone, ce qui ne facilite pas la conception*), elle a les avantages suivants :

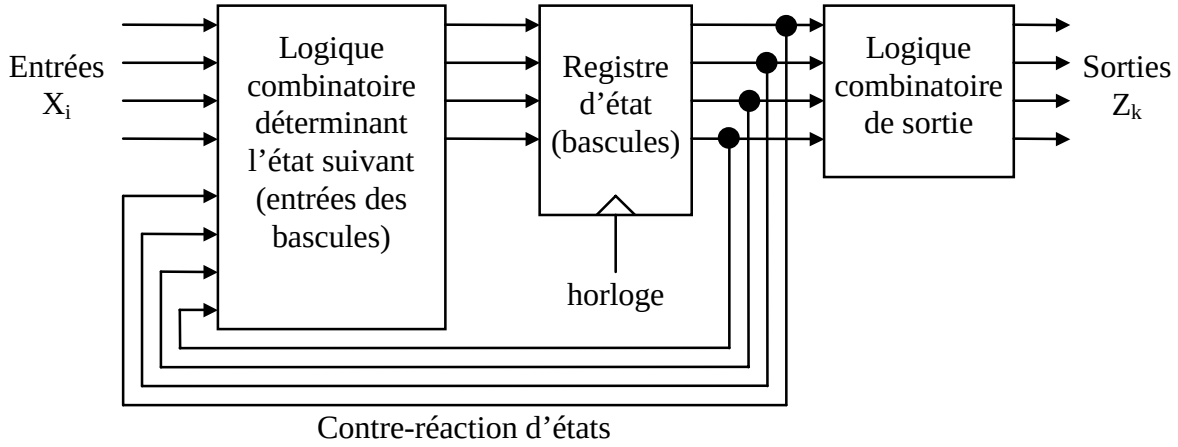
1. Fréquence de fonctionnement plus élevée qu'en logique synchrone.
2. Consommation plus faible qu'en logique synchrone à fréquence de fonctionnement identique.

Pour toutes ces raisons, depuis maintenant 20 ans, tous les montages correctement conçus utilisent la logique séquentielle synchrone.

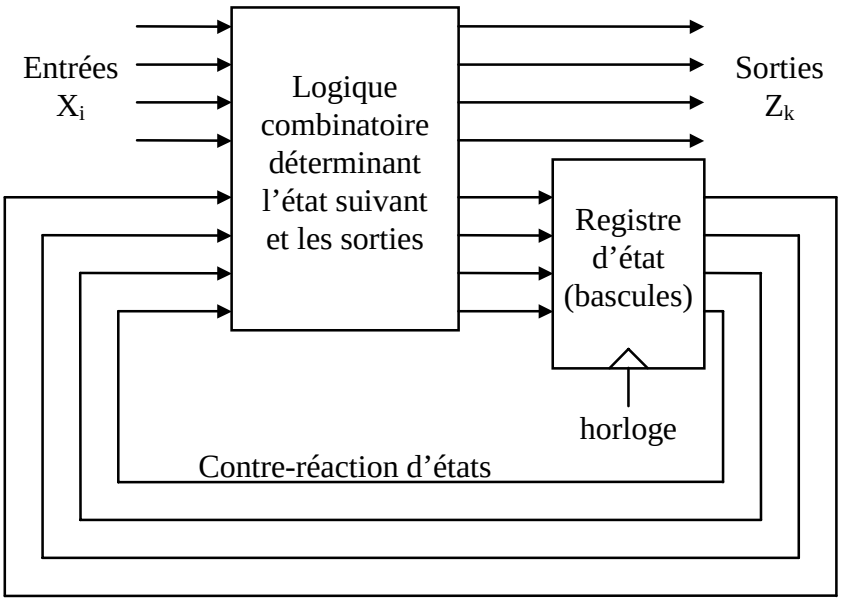
2.8 Machines d'états

Dans ce chapitre, nous allons examiner rapidement le type le plus important de circuit séquentiel : la machine d'état. Une machine d'état est ainsi appelée car la logique séquentielle qui l'implémente ne peut prendre qu'un nombre fini d'états possibles. Il existe deux familles de machines d'état :

- la machine de Moore. Les sorties dépendent seulement de l'état présent n. Un circuit combinatoire d'entrée détermine, à partir des entrées et de l'état présent n, les entrées des bascules (D principalement) du registre d'état permettant de réaliser l'état futur n+1. Les sorties sont une combinaison logique de la sortie du registre d'état et changent de manière synchrone avec le front actif de l'horloge.



- la machine de Mealy. Les sorties dépendent de l'état présent n mais aussi de la valeur des entrées. La sortie peut donc changer de manière asynchrone en fonction de la valeur des entrées. Il existe une variante synchrone de la machine de Mealy avec un registre placé sur les sorties et activé par l'horloge. Toutefois, la machine de Moore est plus adaptée pour réaliser une machine d'état totalement synchrone.

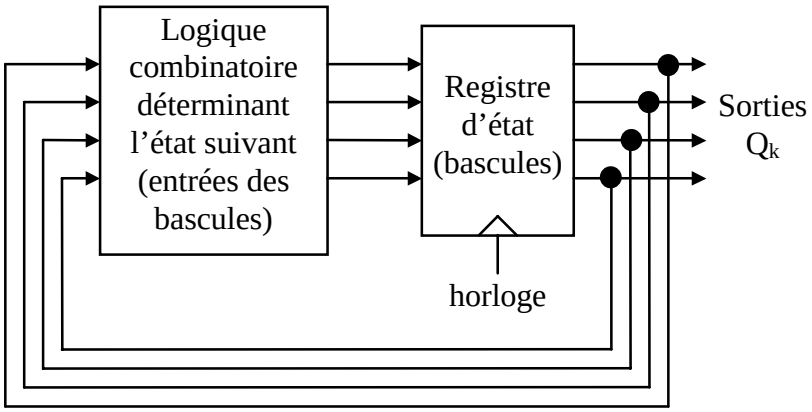


2.9 Les générateurs de séquences synchrones

2.9.1 Compteur, décompteur, générateur pseudoaléatoire

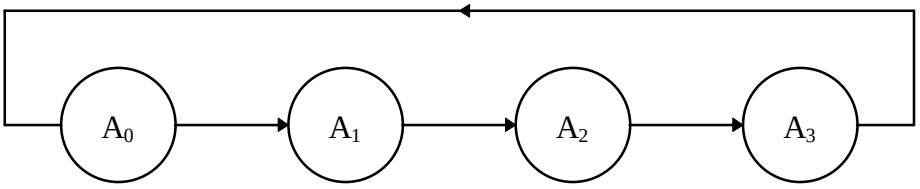
Les machines d'états servent à réaliser des automatismes complexes (contrôleur de feux tricolores par exemple) et ne nous intéressent pas directement. Par contre, nous allons maintenant étudier un cas particulier de machine d'état : le générateur de séquence synchrone. Il s'agit d'un cas simplifié de machine d'état de Moore dans laquelle :

- les sorties et les états internes sont identiques,
- la séquence est non-programmable,
- il n'y a pas d'entrées (il y a éventuellement des entrées de chargement, mais elles n'interviennent pas dans le déroulement de la séquence).



Contre-réaction d'états

Les générateurs de séquences synchrones sont composés de n bascules synchronisées par une horloge qui est la seule commande extérieure du circuit. On peut ainsi coder 2^n états différents et réaliser un graphe d'évolution du type :



Le cycle comporte au plus 2^n états qui se succèdent toujours dans le même ordre. On débute l'étude par le codage des états. On peut généralement les classer parmi les séquences du tableau ci-dessous (sur 3 bits par exemple). Pour répondre à un besoin spécifique, tout autre combinaison des sorties est possible et peut constituer un cycle de m états avec $m \leq 2^n$.

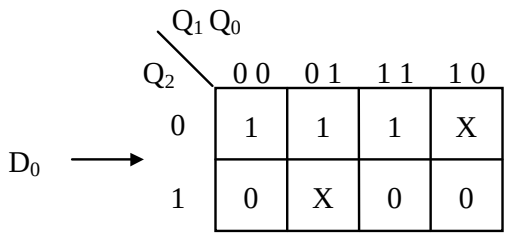
compteur cycle complet (modulo 8)										compteur cycle incomplet												
binaire naturel										binaire quelconque												
compteur				décompteur				gray			Johnson				1 parmi 3				pseudo-aléatoire			
Q_2	Q_1	Q_0		Q_2	Q_1	Q_0		Q_2	Q_1	Q_0		Q_2	Q_1	Q_0		Q_2	Q_1	Q_0		Q_2	Q_1	Q_0
0	0	0		1	1	1		0	0	0		0	0	0		0	0	1		0	0	1
0	0	1		1	1	0		0	0	1		0	0	1		0	1	0		1	0	0
0	1	0		1	0	1		0	1	1		0	1	1		1	0	0		1	1	0
0	1	1		1	0	0		0	1	0		1	1	1						1	1	1
1	0	0		0	1	1		1	1	0		1	1	0						0	1	1
1	0	1		0	1	0		1	1	1		1	0	0						1	0	1
1	1	0		0	0	1		1	0	1										0	1	0
1	1	1		0	0	0		1	0	0												

2.9.2 Cas particulier : les registres à décalages bouclés

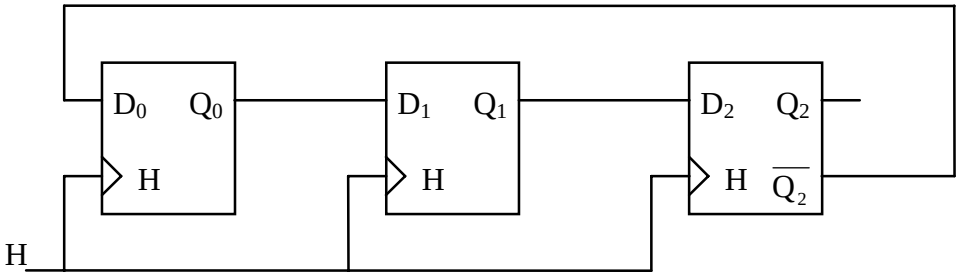
Une fois le codage des états déterminé, on regarde dans la séquence s'il y a un décalage temporel. Si tel est le cas, alors il y a une solution évidente à base de bascules D d'équation $Q_{n+1} = D$. Parmi les séquences précédentes, les compteurs Johnson et 1 parmi 3 peuvent être réalisés à partir de bascules D.

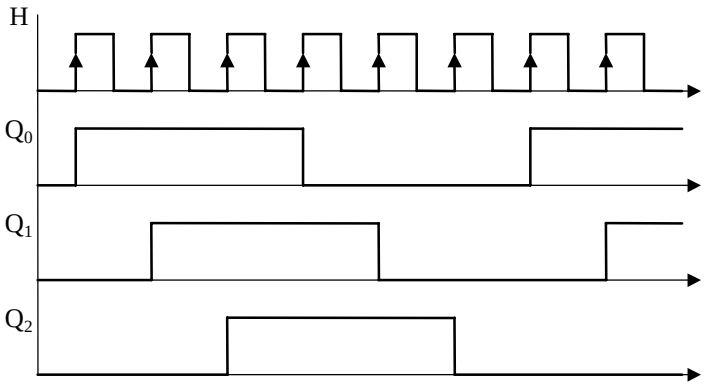
- Le compteur Johnson. On détermine à partir de la séquence d'états que $Q_1^{n+1} = D_1 = Q_0^n$, $Q_2^{n+1} = D_2 = Q_1^n$ et $Q_0^{n+1} = D_0 = f\left(Q_0^n, Q_1^n, Q_2^n\right)$. Il faut donc seulement chercher l'équation de D_0. Pour cela, on se demande quelle valeur mettre sur D_0 pour obtenir Q_0.

Q_2	Q_1	Q_0	D_0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	0
1	1	0	0
1	0	0	0



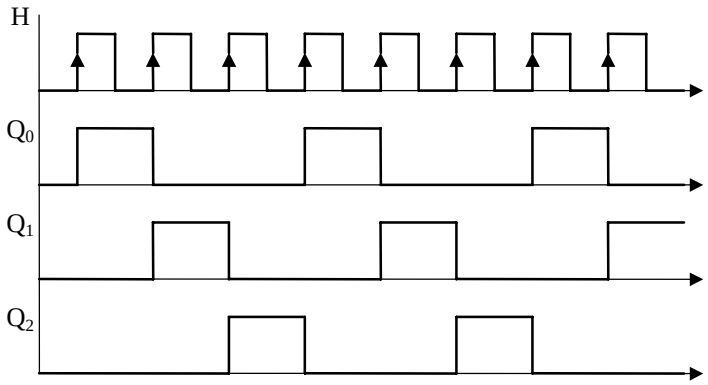
En simplifiant au maximum avec le tableau de Karnaugh, on obtient $D_0 = \overline{Q_2}$ ce qui donne le schéma et le chronogramme suivant :





Les bascules doivent être mises à 0 à la mise sous tension. On obtient des horloges de mêmes périodes, mais décalées en phase. Il s'agit d'un code jointif comme le code GRAY car il n'y a qu'une sortie qui change à chaque coup d'horloge.

- Le compteur 1 parmi 3. Le schéma est identique au précédent avec D_0 connecté sur Q_2 (au lieu de $\overline{Q_2}$), et il faut mettre à 1 une des bascules à la mise sous tension (et les autres à 0). On obtient des horloges décalées à temps jointifs.



2.9.3 Cas général : la méthode de la table d'états

Pour exposer cette méthode, nous allons traiter un exemple avec le compteur binaire 3 bits. Elle comprend 3 phases :

1. Détermination du tableau de Karnaugh donnant $Q_2^+Q_1^+Q_0^+$ en fonction de $Q_2Q_1Q_0$.

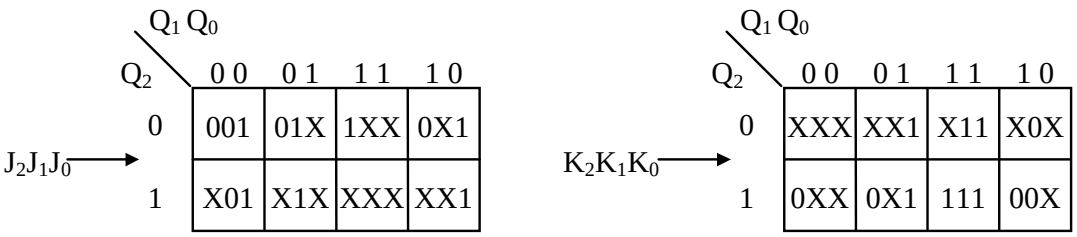
Q_2 \ Q_1Q_0	0 0	0 1	1 1	1 0
0	001	010	100	011
1	101	110	000	111

$Q_2^+Q_1^+Q_0^+ \longrightarrow$

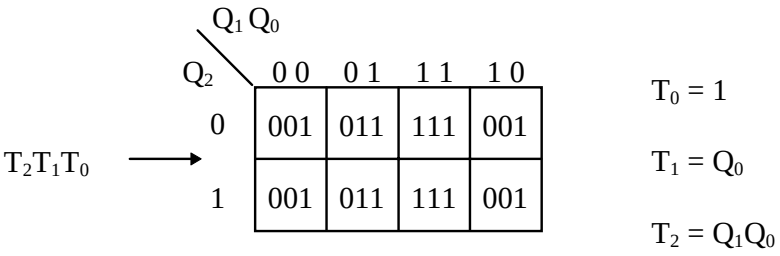
2. Détermination des tableaux de Karnaugh donnant les commandes nécessaires à l'obtention de $Q_2^+Q_1^+Q_0^+$.
   - Prenons l'exemple d'une réalisation avec des bascules D d'équation $Q^+=D$. On obtient directement du tableau précédent $Q_0^+ = D_0 = \overline{Q_0}$, $Q_1^+ = D_1 = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$ et $Q_2^+ = D_2 = \overline{Q_2}Q1Q_0 + Q_2\overline{Q_1} + Q_2\overline{Q_0}$.

   - Avec des bascules JK d'équation $Q^+ = J.\overline{Q} + \overline{K}.Q$, il faut poser :
     * $Q = 0 \Rightarrow J = Q^+$ quelque soit K (K = X).
     * $Q = 1 \Rightarrow \overline{K} = Q^+$ quelque soit J (J = X).

     D'où on tire les tableaux :

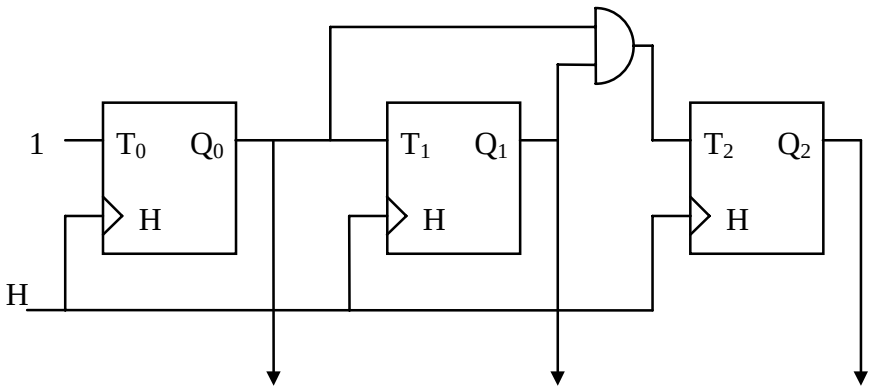


On peut ainsi déterminer les signaux J et K de chaque bascule. Dans le cas d'un compteur binaire, la solution à base de bascule T est la plus efficace, il suffit de poser J = K = T. On en tire $J_2J_1J_0 = K_2K_1K_0$ et un seul tableau de Karnaugh (en combinant les deux précédents).



On peut aussi, sans passer par les tables de vérité des bascules JK, déduire T de l'expression de Q^+ avec la relation $T = Q^+ \oplus Q$, ce qui nous donne exactement le tableau de Karnaugh précédent.

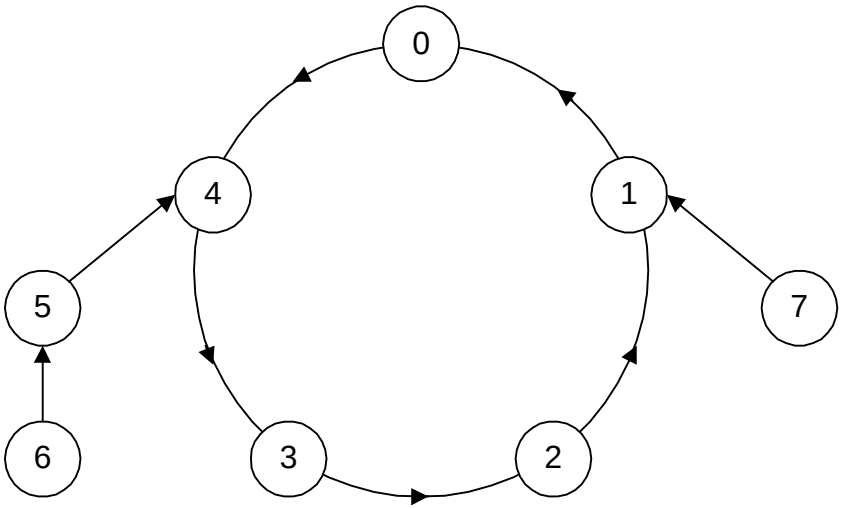
3. A partir des tableaux de la phase 2, élaboration du schéma.



2.9.4 Aléa dans les générateurs à cycle incomplet

Dans un générateur à cycle incomplet, tous les états possibles ne sont pas utilisés (compteur 1 parmi 3 par exemple). Quand on simplifie au maximum les équations à l'aide du tableau de

Karnaugh, le fonctionnement est correct tant que l'on reste dans le cycle principal. Toutefois, si le compteur s'initialise sur une valeur qui ne se trouve pas dans ce cycle, il peut travailler dans un cycle parasite ou rester sur un état bloqué. On peut avoir par exemple un décompteur 3 bits ayant pour cycle principal ($4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0 \rightarrow 4 \rightarrow ...$) et un cycle parasite ($5 \rightarrow 6 \rightarrow 5 \rightarrow ...$). Si le circuit prend la valeur 5 à la mise sous tension, il restera bloqué dans le cycle parasite. Il faut donc, lors de la simplification avec les tableaux de Karnaugh, prendre soin de connecter les cycles parasites éventuels ou les états bloqués avec le cycle principal comme dans le cas suivant :



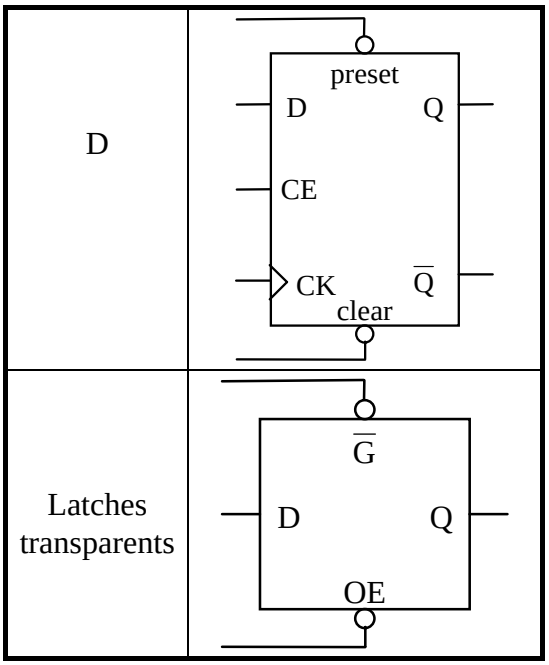
On voit bien qu'avec cette modification, le compteur rejoint le cycle principal quelque soit la valeur d'initialisation. On réalise ainsi un compteur auto-initialisé (self-starting counter).

2.10 Circuits logiques séquentiels

Ce paragraphe va traiter des circuits logiques séquentiels se trouvant dans le catalogue logique standard, le TTL data book. Quoique obsolètes, les fonctions de ce catalogue existent, à quelques variantes près, dans les autres technologies (notamment en CMOS) ainsi qu'en CAO.

2.10.1 Les bascules élémentaires

appellation	notation



2.10.2 Les compteurs

2.10.2.1 Introduction

A l'exception des cas particuliers où l'on réalise des compteurs en associant des bascules D ou JK, on utilise généralement des compteurs en circuits intégrés du commerce ou leurs équivalents dans les bibliothèques des outils de CAO. Nous allons étudier ici quelques compteurs appartenant au TTL data book, les techniques de base ainsi que les principales recommandations d'utilisation. Il existe deux familles de compteurs :

1. les compteurs synchrones : toutes les sorties changent d'état un temps $t_{P\ clock\ to\ Q}$ après le front actif de l'horloge.
2. les compteurs asynchrones : la sortie N change d'état un temps t_P après la sortie N-1, la sortie 0 change d'état un temps t_P après le front actif de l'horloge (aucune sortie ne change en même temps que les autres).

On trouve des compteurs binaires N bits (modulo 2^N) ou décimaux (par décades). Ils possèdent des broches supplémentaires telles que :

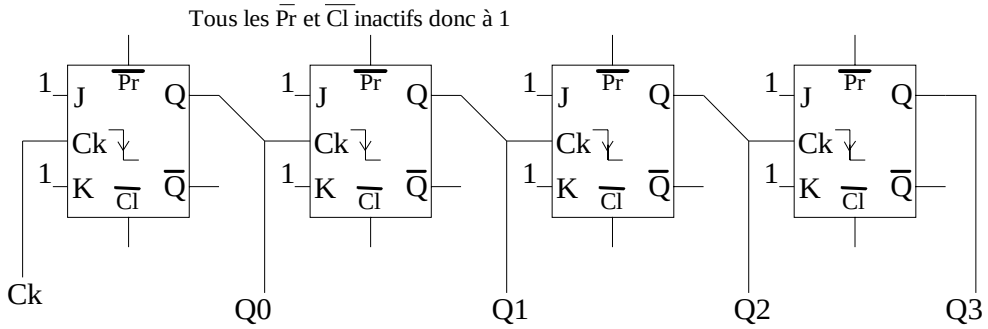
- l'entrée CLEAR ou remise à zéro de l'état du compteur.
- l'entrée LOAD ou chargement. Cette entrée permet de charger en parallèle une valeur dans le compteur, cette valeur devant être présente sur les entrées de chargement parallèle.
- l'entrée UP/DOWN activant le comptage ou le décomptage.
- l'entrée ENABLE ou validation autorisant ou bloquant le comptage.

- la sortie CARRY (aussi appelée CEO pour Chip Enable Output) qui indique le dépassement de capacité et sert à mettre les compteurs en cascade.

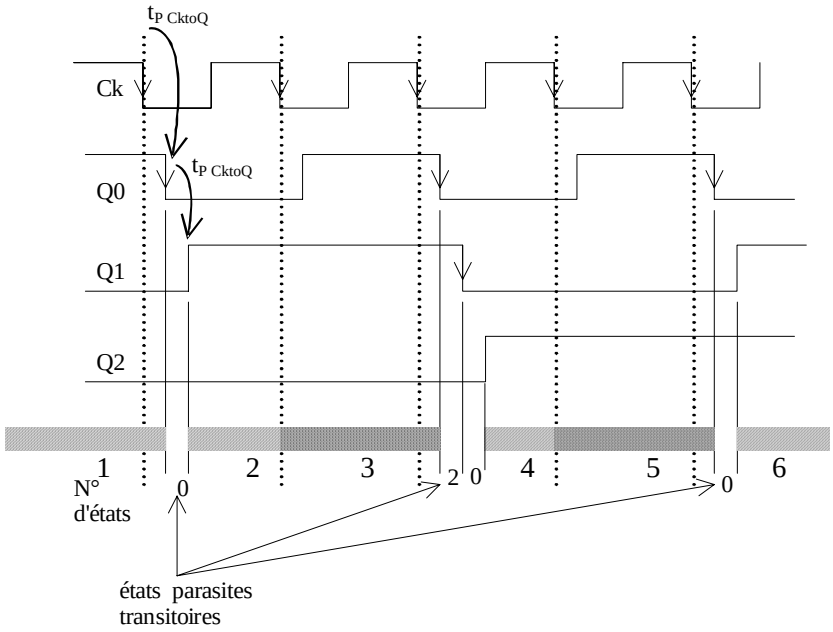
Les entrées sont actives à 0 ou à 1 et à action synchrone ou asynchrone selon les circuits. Il faut se reporter aux caractéristiques et aux chronogrammes donnés par le constructeur pour avoir plus de détails.

2.10.2.2 Compteurs Binaires Asynchrones

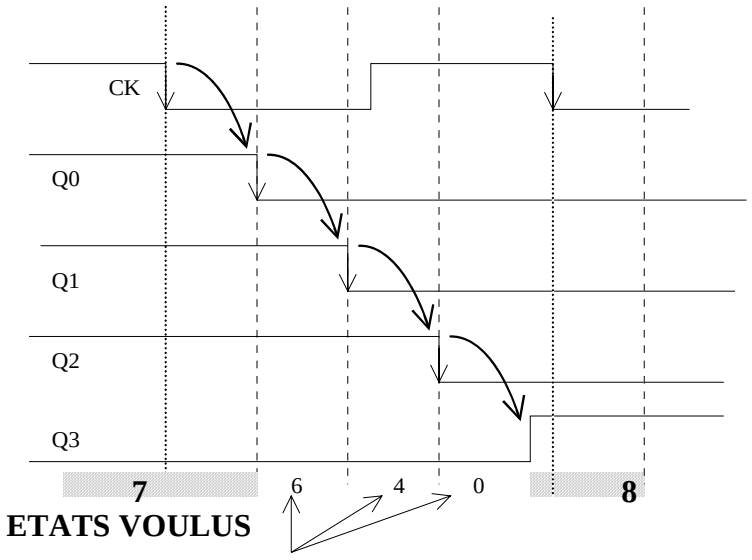
Les compteurs asynchrones, qui ne sont plus commercialisés, peuvent toujours être utilisés comme fonction dans un circuit logique. Leur principe de fonctionnement est le suivant :



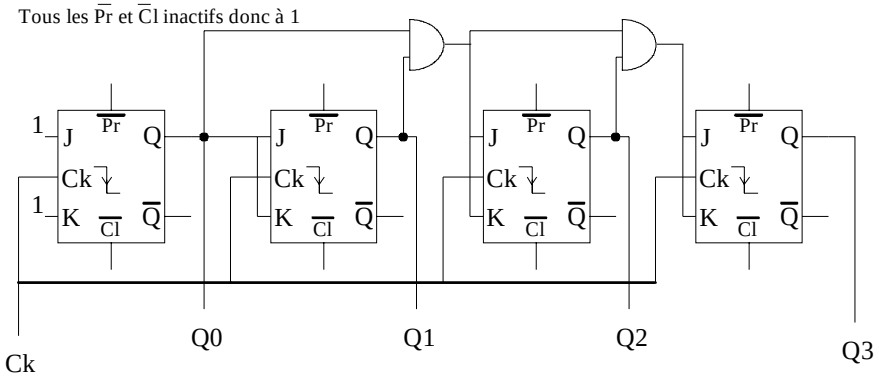
Cet exemple (74LS197) réalise un comptage de 0 à 15 puis retour à 0 puisque, pour chaque bascule JK, $Q_{n+1} = \overline{Q_n}$. Les temps de propagation $t_{P\ CktoQ}$ s'ajoutant, un dessin en haute fréquence montre des états parasites de commutation :



Si la fréquence augmente, les états parasites peuvent durer plus longtemps que les états réels du compteur. Toutefois, si on ne réalise qu'une division de fréquence, ils ne sont pas du tout gênants, et on obtient alors la structure de diviseur de fréquence la plus rapide car la fréquence maximale du compteur est égale à la fréquence maximale de la bascule de poids faible. De plus, ce montage consomme moins que son équivalent synchrone car chaque bascule fonctionne à la fréquence minimale possible pour réaliser la fonction. Par contre, on ne peut exploiter les états de sortie du compteur que quand ils sont stabilisés, ce qui en limite fortement l'usage. Il faudra compter sur un front et échantillonner les sorties par une bascule D active sur l'autre front (dans le cas où la somme des $t_{P\ ck\ to\ Q}$ ne dépasse pas la demi-période de l'horloge). Malgré cela, on atteint très rapidement les limites utilisables. Par exemple, si on augmente trop la fréquence d'un compteur 4 bits, on peut obtenir le chronogramme suivant :



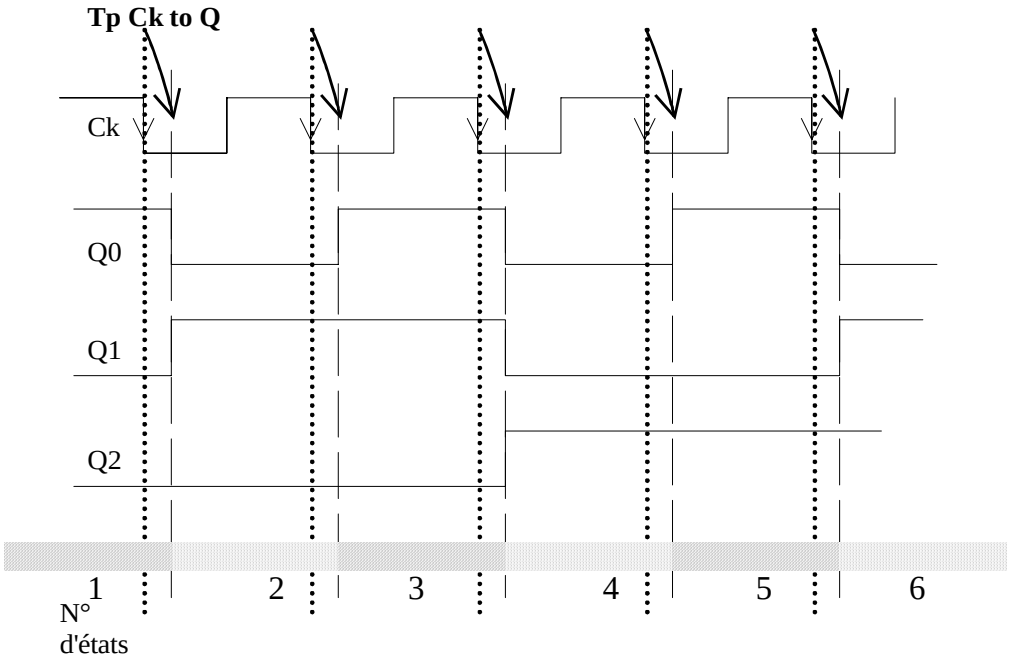
Comme il est impossible d'exploiter les états de sortie de ce compteur à cette fréquence, il faut passer en logique synchrone avec un schéma du type :



Toutes les sorties du compteur changent un temps $t_{P\ ck\ to\ Q}$ après le front actif de l'horloge. Il n'y a plus d'états parasites transitoires. Par contre, pour une simple division de fréquence, ils sont moins performants que les compteurs asynchrones car la fréquence maximale d'utilisation est plus faible :

$$f_{max} = \frac{1}{t_{P\ Ck\ to\ Q} + t_{setup} + t_{P\ logique\ combinatoire}}$$

Le chronogramme de sortie est donc :



Voyons maintenant quelques compteurs du commerce.

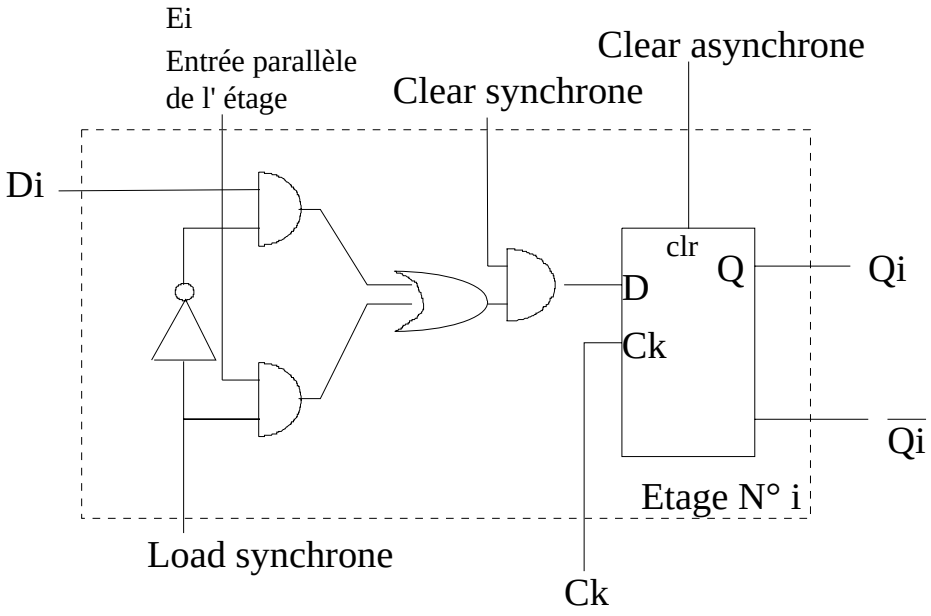
2.10.2.3 Quelques compteurs synchrones du commerce

Les compteurs synchrones LS160, 161, 162, 163 et LS190, 191, 192, 193 sont des compteurs 4 bits qui changent d'état sur le front montant de l'horloge. Le tableau suivant résume leurs caractéristiques :

	binaire	décimal	compteur	compteur-décompteur	chargement (load)	remise à zéro (raz)
160		*	*		synchrone	asynchrone
161	*		*		synchrone	asynchrone
162		*	*		synchrone	synchrone
163	*		*		synchrone	synchrone
190		*		ligne down/up	synchrone	
191	*			ligne down/up	synchrone	
192		*		deux horloges	asynchrone	asynchrone
193	*			deux horloges	asynchrone	asynchrone

Ils sont réalisés à partir de 4 bascules D, avec un rebouclage adéquat de Q sur D.

- Les chargements synchrones, remises à zéros asynchrones ou synchrones sont effectués par le circuit suivant :



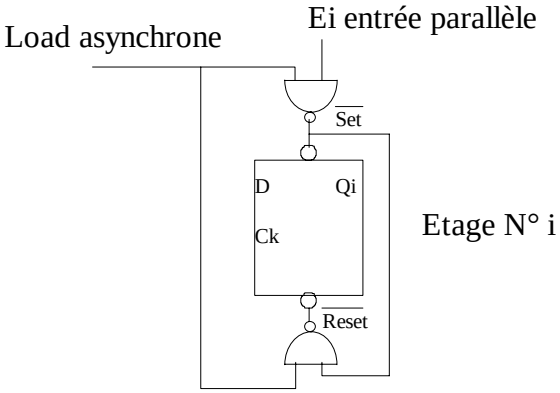
- Pour les chargements parallèles asynchrones, on agit sur les entrées Set et Reset des bascules D. L'équation logique de ces entrées s'obtient facilement : Set = Ei.Load et Reset = $\overline{Ei}$ .Load. Pour gagner une porte, on peut écrire :

$$Set = Ei \cdot Load$$

$$Reset = \overline{Ei} \cdot Load + Load \cdot \overline{Load} = Load\,(\overline{Ei} + \overline{Load})$$

$$D' \text{ ou } Reset = Load \cdot \overline{Set}$$

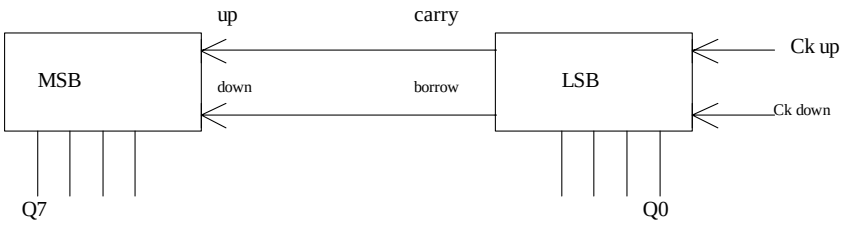
D'où le montage suivant pour chaque étage n° i :



2.10.2.4 Mise en cascade de compteurs

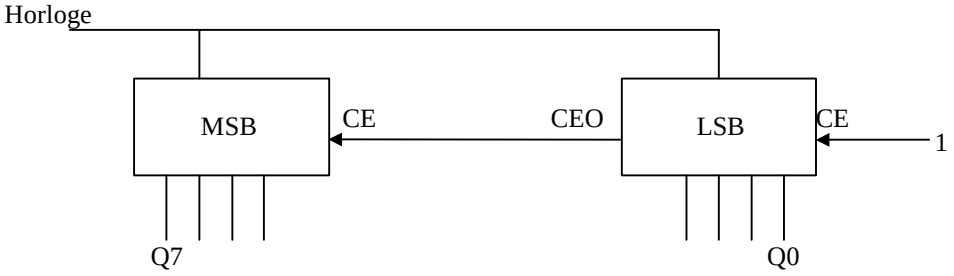
Suivant qu'ils possèdent ou non une entrée de validation, on met en cascade les compteurs de manière synchrone ou asynchrone.

2.10.2.4.1 mise en cascade série ou asynchrone



La sortie carry (borrow) actionne directement le comptage (décomptage) des poids forts. Elle sert d'horloge pour actionner le compteur de poids plus élevé. Le compteur n'est plus entièrement synchrone (seulement 4 bits par 4 bits), ce dont il faudra tenir compte pour éviter les aléas lors de la réalisation de compteurs modulo quelconque.

2.10.2.4.2 mise en cascade parallèle ou synchrone

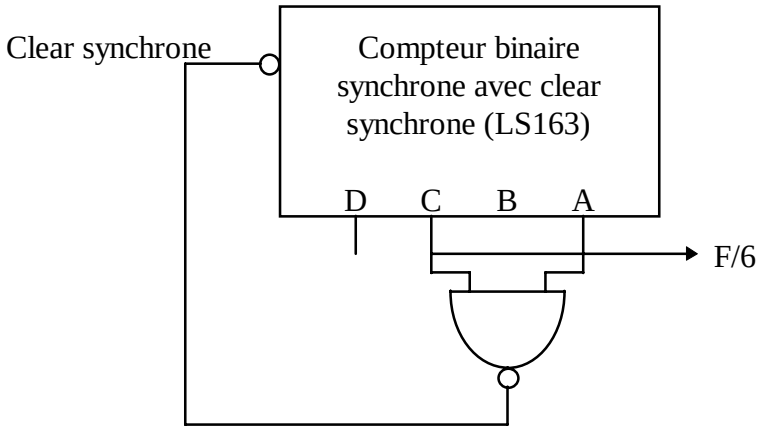


Le compteur reste ici totalement synchrone. La sortie CEO valide le compteur de poids plus élevé pour le coup d'horloge suivant.

2.10.2.5 Réalisation de compteurs modulo quelconque.

2.10.2.5.1 Action sur l'entrée Clear synchrone

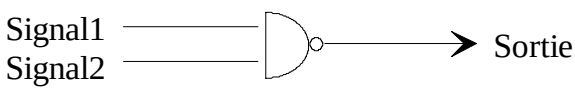
Lorsque l'état final du compteur est détecté, le front actif suivant de l'horloge remet à zéro le compteur. Une détection incomplète est possible et même souhaitable (nous le verrons plus loin pour limiter les aléas), elle consiste à ne détecter que les bits à 1. Dans l'exemple suivant utilisant un SN74LS163, on réalise un diviseur par 6 en détectant l'état 5, puis en réinitialisant le compteur au coup d'horloge suivant. Les états en sortie seront donc 0, 1, 2, 3, 4, 5, 0, 1...

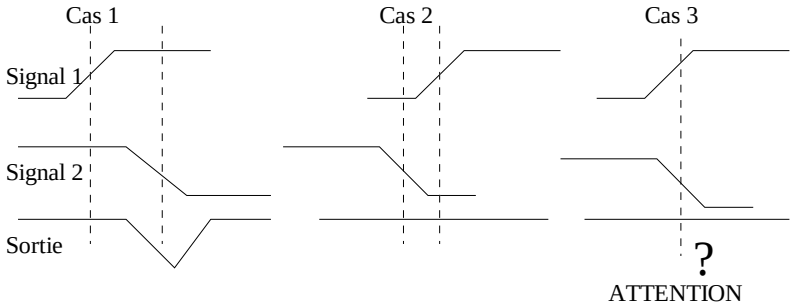


Bien que le compteur soit synchrone, de petites dispersions dans les $t_{P\ ck\text{-}to\text{-}Q}$ peuvent fournir un état parasite transitoire. Ces aléas ne sont pas forcément gênants, nous allons les étudier ci-dessous.

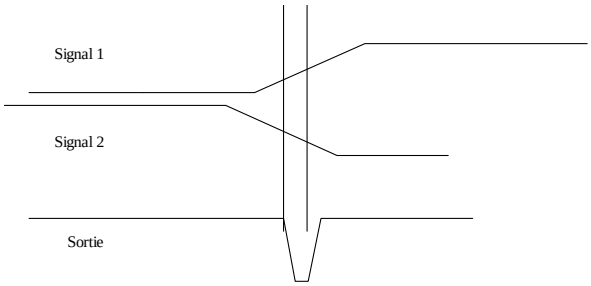
2.10.2.5.2 Rappel des cas possibles d'aléas

Un aléa peut survenir lorsque deux entrées d'une porte élémentaire changent d'état à peu près en même temps. Si les temps de transitions sont voisins du temps de traversée de la porte, on a trois possibilités :





Dans le cas particulier de cette NAND, le cas n°1 provoquera systématiquement un parasite à 0 car la transition 0-1 sur le signal 1 fait changer la sortie (avec signal 2 à 1). Par contre, le cas n°2 ne provoquera jamais d'aléa puisque la transition 1-0 sur le signal 2 ne fait pas changer la sortie (avec signal 1 à 0). Quand au cas n°3, tout dépendra des caractéristiques réelles de chaque porte. Certaines portes produiront un aléa, d'autres non. Si le temps de transition est beaucoup plus grand que le temps de traversée de la porte, alors l'aléa en sortie sera systématique.



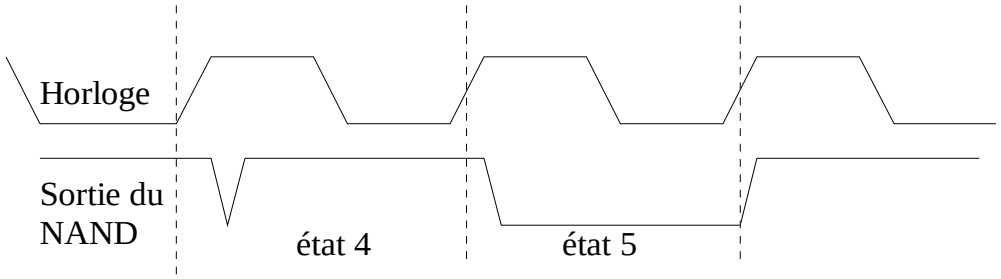
2.10.2.5.3 Influence de l'aléa

Reprenons l'exemple du diviseur par 6. L'aléa peut exister en sortie de la porte NAND détectant l'état 5, au passage de l'état 3 à l'état 4. Sur une entrée synchrone de remise à zéro, cet aléa n'a pas d'action car il ne se produit pas sur le front actif de l'horloge. Avec un clear synchrone, les aléas de commutation n'ont aucun effet. L'étude de ce type de montage s'en trouve grandement facilitée.

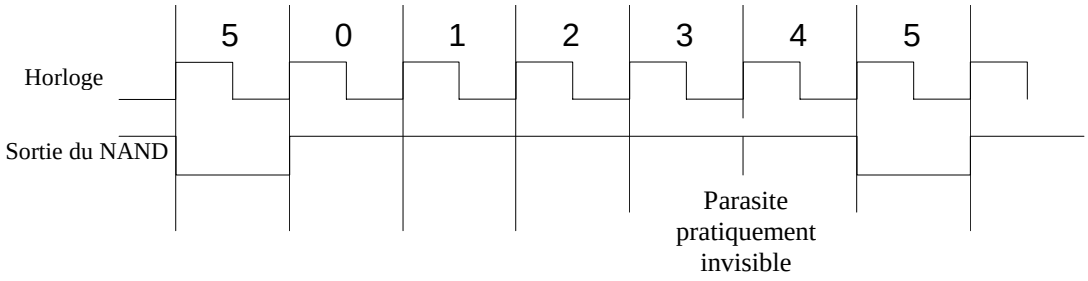
Par contre, si on utilise un clear asynchrone, alors l'aléa au passage de l'état 3 à l'état 4 peut remettre le compteur à 0. Cela dépendra de sa durée et de la rapidité du compteur. Il vaut donc mieux éviter l'utilisation d'un clear asynchrone pour réaliser un diviseur par N. Il faut toutefois noter que l'emploi d'une raz asynchrone avec un compteur asynchrone est possible,

puisque l’instant de variation des sorties est décalé et connu. L’étude de l’influence des aléas est dans ce cas beaucoup plus facile à prévoir.

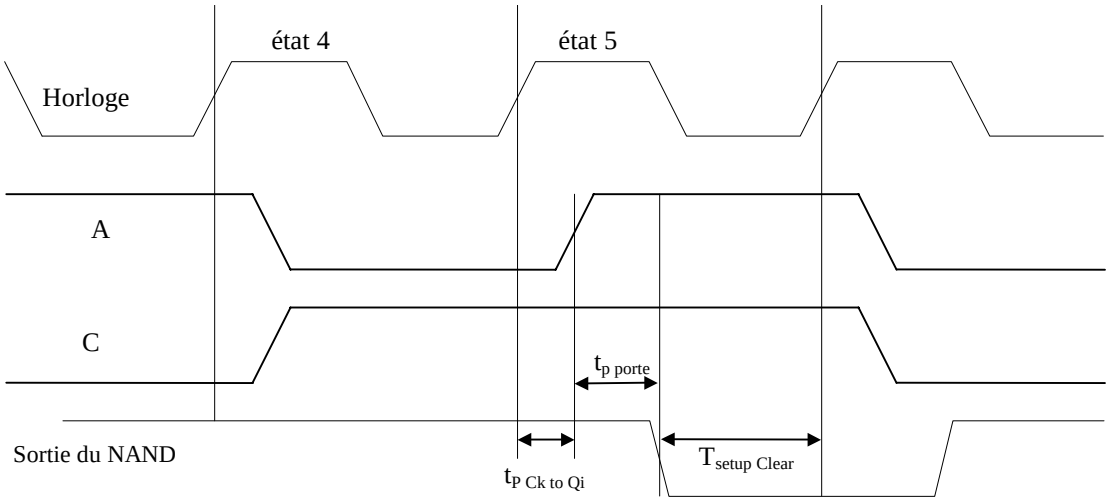
Comme nous l’avons déjà vu, il ne faut jamais utiliser comme horloge d'un compteur ou d’une bascule une combinaison logique de sorties d’un compteur. Des impulsions parasites (des glitches) se trouveront sur ce signal et seront prises pour des fronts d’horloge supplémentaires. De plus, la largeur de ces parasites dépendra uniquement des temps de propagation du compteur (clock to Q) et de la rapidité de la porte mais pas de la fréquence de fonctionnement. Si par exemple, on a le chronogramme suivant à la fréquence de 1 MHz :



A 1 kHz, on aura un parasite de même largeur mais il ne sera pratiquement plus visible quoique toujours actif.



La fréquence maximale de fonctionnement (pour l’exemple du diviseur par 6) est déterminée par le temps séparant le changement d’état en sortie de la NAND et le front actif suivant de l’horloge.



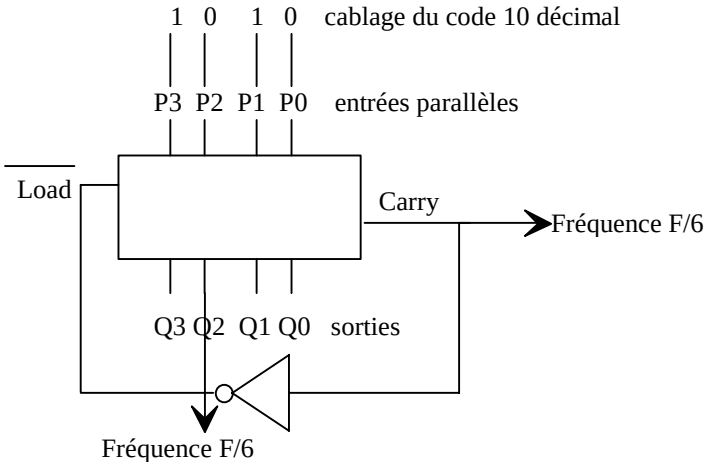
On déduit de ce chronogramme la fréquence maximale du compteur :

$$F_{max} = \frac{1}{t_{ck\ to\ Qi\ max} + t_{p\ porte\ max} + t_{setup\ Clear\ min}}$$

Au-delà de cette fréquence, le signal Clear arrive trop tard pour remettre à zéro le compteur, il apparaît un coup d'horloge supplémentaire dans le cycle (ou pire, un phénomène de métastabilité).

2.10.2.5.4 Action sur l'entrée LOAD synchrone

En mode comptage, au moment du passage de l'état 2^n-1 à l'état zéro, le compteur fournit un signal de retenue « carry ». On l'utilise pour charger une valeur qui servira de valeur initiale dans la séquence. L'exemple suivant reprend le diviseur par 6 mais avec en sortie les états 10, 11 ,12 ,13, 14, 15, 10, 11, …

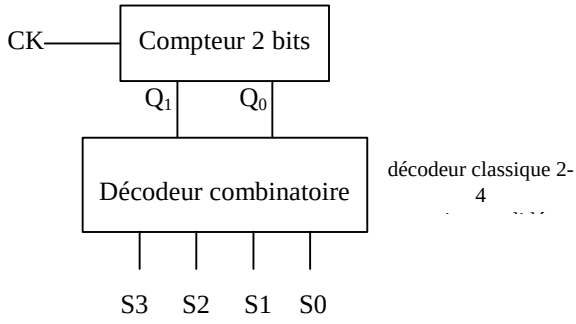


L'intérêt de ce type de montage est la réalisation simple de diviseurs programmables, un simple code 2^n - k sur les entrées de chargement permettant d'effectuer la division par k voulue.

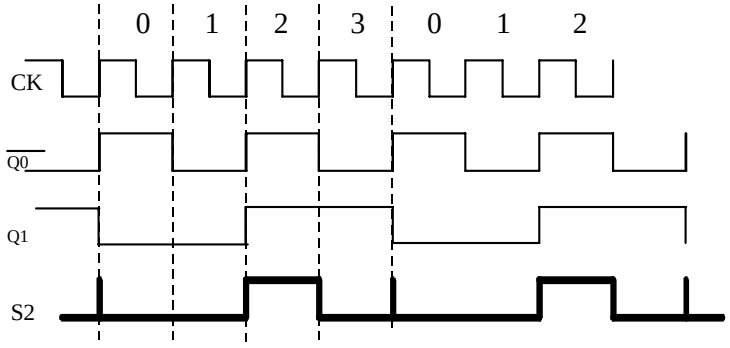
En mode décomptage, avec un compteur prévu à cet effet, le passage de 0 à 2^n -1 est indiqué par un signal de retenue "borrow". On peut donc réaliser un diviseur fonctionnant par décomptage, la présence du code k sur les entrées de chargement permettant d'effectuer une division par k +1. Dans le cas du diviseur par 6, une valeur 5 sur les entrées de chargement donne en sortie les états 0, 1, 2, 3, 4, 5, 0, 1, …

2.10.2.6 Exploitation des états de sortie d'un compteur.

Prenons par exemple un compteur deux bits comptant de 0 à 3. On veut créer, à l'aide d'un circuit combinatoire, 4 impulsions détectant respectivement les états 0, 1, 2 ,3.



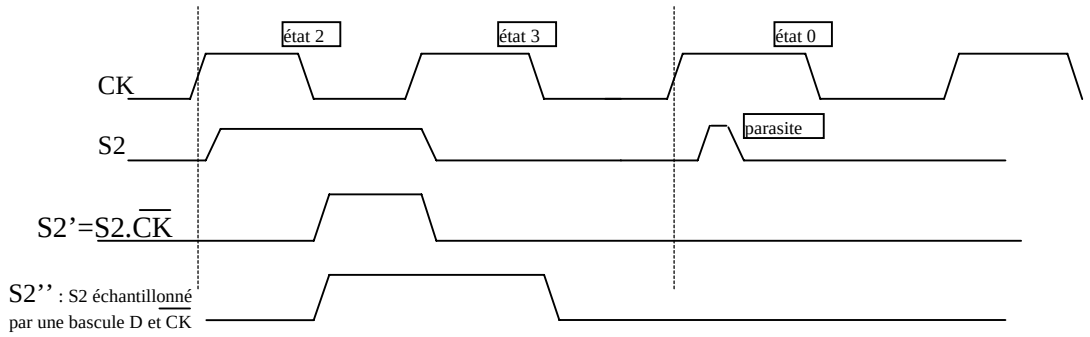
Etudions la sortie $S2 = Q1.\overline{Q0}$:



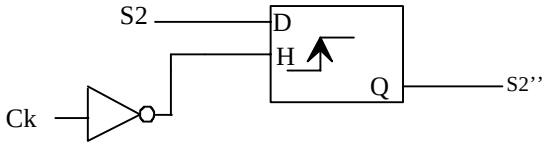
Lors du passage de l'état 3 à l'état 0, les signaux Q1 et $\overline{Q0}$ évoluant en sens inverse, un parasite est possible sur la sortie S2. Sa présence dépend du type de compteur (synchrone ou

asynchrone), de la technologie des circuits utilisés, et également de la façon dont est réalisée la fonction S2 à l'intérieur du décodeur. Ce parasite n'est pas gênant pour une exploitation synchrone. **Pour un usage asynchrone** (attaque d'une entrée horloge d'un compteur ou d'un registre), il faut absolument le supprimer. Il existe trois techniques possibles :

- Suppression d'aléas par l'utilisation d'un compteur de type Gray ou Johnson. Le décodeur combinatoire doit être modifié. Comme une seule sortie Q_n change à chaque coup d'horloge, aucun aléa sur les sorties S_n n'est possible si le décodeur est « hazard free » ce qui est généralement le cas (mais il faut quand même étudier de près le problème).
- Suppression d'aléas par validation avec l'horloge principale. On voit que le signal $S2' = S2.\overline{CK}$ est sans parasite et qu'il dure 1/2 période d'horloge. Son front montant se situe pratiquement au milieu de l'état décodé du compteur.



- Suppression d'aléas par échantillonnage avec une bascule D.

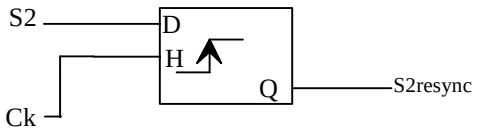


On obtient un signal S2'' ayant une période d'horloge pour largeur, sans parasite. Son niveau haut est cependant à cheval sur l'état détecté et l'état suivant, ce qui n'est pas forcément un inconvénient. On peut en effet exploiter son front montant situé au milieu de l'état détecté.

Toutes ces méthodes sont du domaine de la **bidouille** et l'**exploitation synchrone de l'état de sortie est fortement conseillée** sauf si vous ne pouvez pas faire autrement

(notamment si le retard entre le changement d'état et sa détection doit être inférieur à une période d'horloge).

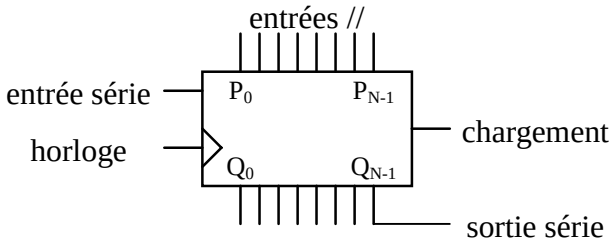
Le montage normal est bien sur :



2.10.3 Les registres à décalages

2.10.3.1 Définition

Un registre à décalage est une association en cascade de bascules D permettant de décaler à gauche ou à droite une série de bits de données. On y trouve généralement une entrée de données en série, N entrées de chargement en parallèle et N sorties. La Nième sortie peut servir de sortie série.



Il y a trois modes principaux d'utilisation :

1. L'entrée série avec sortie parallèle. Prenons un exemple avec un registre 4 bits et voyons l'évolution des données en sortie. A chaque coup d'horloge, la donnée sur l'entrée série est copiée sur l'étage 0 et la donnée se trouvant à l'étage i-1 est transférée à l'étage i. Ainsi, les données rentrées en série sont mises en parallèle sur les sorties.

entrée série	Ck	Q0	Q1	Q2	Q3
b0	↑	b0	0	0	0
b1	↑	b1	b0	0	0

b2	↑	b2	b1	b0	0
b3	↑	b3	b2	b1	b0
b4	↑	b4	b3	b2	b1

2. L'entrée parallèle avec sortie série. Prenons un exemple avec un registre 4 bits et voyons l'évolution des données en sortie. Sur le front actif de l'entrée de chargement, les données parallèles sont copiées sur les sorties. Ensuite, à chaque coup d'horloge, la donnée se trouvant à l'étage i-1 est transférée à l'étage i et un 0 est copié sur l'étage 0. On voit apparaître sur la sortie série Q3 les données parallèles mises en série.

chargement	Ck	Q0	Q1	Q2	Q3
1	↑	P0	P1	P2	P3
0	↑	0	P0	P1	P2
0	↑	0	0	P0	P1
0	↑	0	0	0	P0

3. L'entrée série avec sortie série. On reprend le tableau du mode 1, mais on considère les données sortant sur Q3. Ce sont les données de l'entrée série apparaissant après un retard de 4 coups d'horloge.

2.10.3.2 Applications

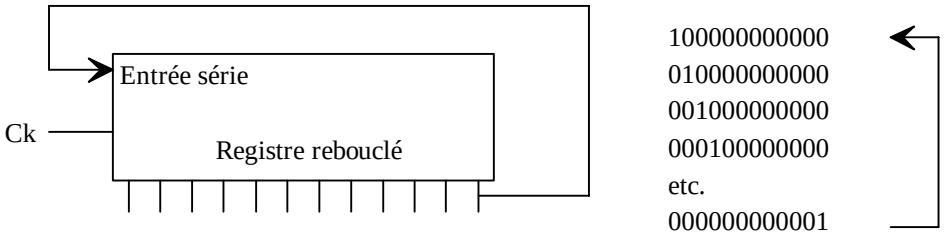
Les registres à décalage ont de nombreuses applications. Parmi celles-ci, on trouve :

- La conversion de données série-parallèle utilisée pour la transmission de données. On prend des bits en série et on les convertit (généralement sur un octet) en parallèle (exemple : le SN74LS164).

- La conversion de données parallèle-série utilisée pour la transmission de données. On prend des données binaires (généralement un octet) en parallèle et on les convertit en un train binaire série (exemple : le SN74LS165).

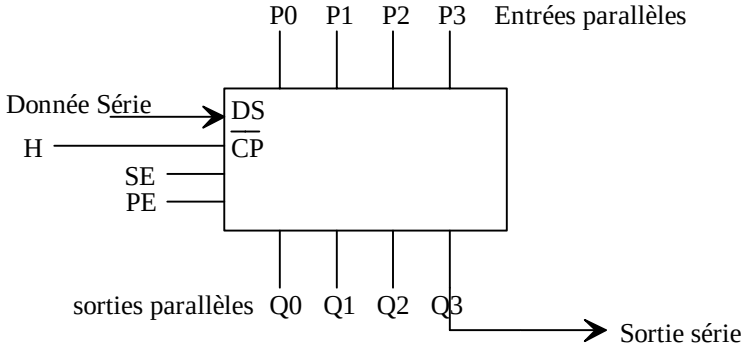
- Les opérations arithmétiques pour réaliser des multiplications et divisions (opérations utilisant le décalage). On entre et on sort alors en parallèle.

- Les compteurs en anneau (entrée série, sortie série). Il faut effectuer un chargement initial, puis à chaque coup d'horloge le contenu se décale en suivant une permutation circulaire. On peut réaliser ainsi des générateurs d'horloges décalées, des chenillards ...



2.10.3.3 Le SN74LS178, registre polyvalent de 4 bits

Finissons ce paragraphe avec un exemple de registre du TTL data book, le SN74LS178. C'est un registre à entrées parallèles, sorties parallèles, entrée et sortie série.



Le plus souvent, dans les registres à décalages, les bits à gauche ont les poids les plus faibles. Cette notation peut être gênante, mais c'est celle des circuits que l'on trouve dans le commerce. Les signaux SE et PE sélectionnent 3 types de fonctionnement possibles :

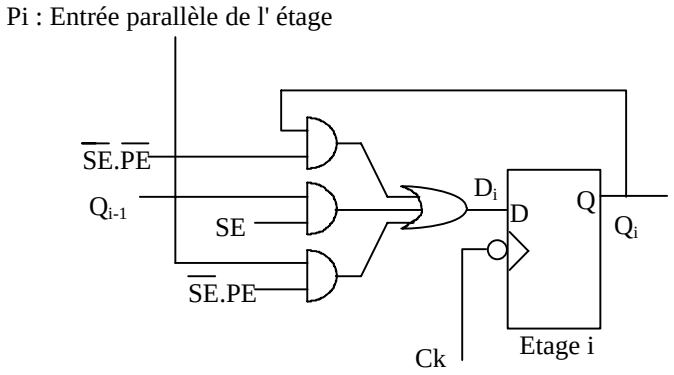
SE	PE	$\overline{CP}$	Action
H	X	↓	décalage : DS → Q0 → Q1 → Q2 → Q3
L	H	↓	chargement : Pi → Qi

L	L	X	pas d'action

Le registre est formé de 4 bascules D. D'après le tableau ci-dessus, on peut écrire immédiatement l'équation au niveau de l'étage numéro i :

$$D_i = SE.Q_{i-1} + \overline{SE}.PE.P_i + Q_i.\overline{SE}.\overline{PE}$$

D'où le schéma suivant pour l'étage i :



2.10.4 Les monostables

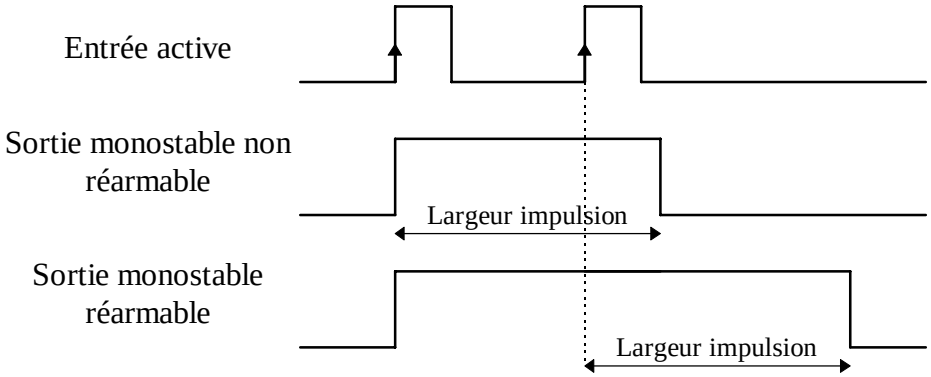
Un monostable est un circuit logique séquentiel qui délivre, sur le front actif d'un signal de commande, une impulsion de durée variable ajustée par un réseau RC. La durée de l'impulsion est approximativement égale à $0,7.R_{ext}.C_{ext}$. Son ordre de grandeur est compris entre une dizaine de nano-secondes et plusieurs secondes. Nous allons examiner deux types courants de monostables :

- SN74LS221 : double monostable non réarmable avec entrée trigger de Schmitt. Si un front actif se produit avant que l'impulsion de sortie ne revienne au repos, il n'est pas pris en compte. Sa table de vérité est la suivante :

entrées			sorties	
clear	A	B	Q	$\overline{Q}$

0	X	X	0	1
X	1	X	0	1
X	X	0	0	1
1	0	↑		
1	↓	1		
↑	0	1		

- SN74LS123 : double monostable réarmable. Si un front actif se produit avant que l'impulsion de sortie ne revienne au repos, sa durée est étendue de la valeur initiale. Sa table de vérité est identique à la précédente.



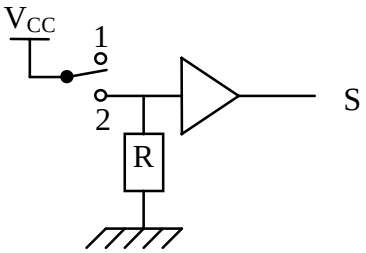
Les monostables ne sont plus utilisés aujourd'hui en conception des circuits logiques. Mais le principe est toujours intéressant, notamment sous sa forme séquentielle synchrone.

2.11 Exercices

exercice 2.1

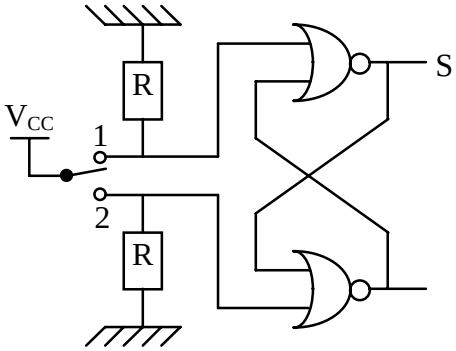
Un interrupteur manuel qui commute d'une ancienne position vers une nouvelle n'établit pas un contact franc avec la nouvelle position, mais il rebondit en l'air un certain nombre de fois avant de se stabiliser dessus. Nous allons voir dans cet exercice un montage dit anti-rebonds.

1. Soit le schéma suivant :



Donner le chronogramme de la sortie S quand l'interrupteur passe de la position 2 à la position 1.

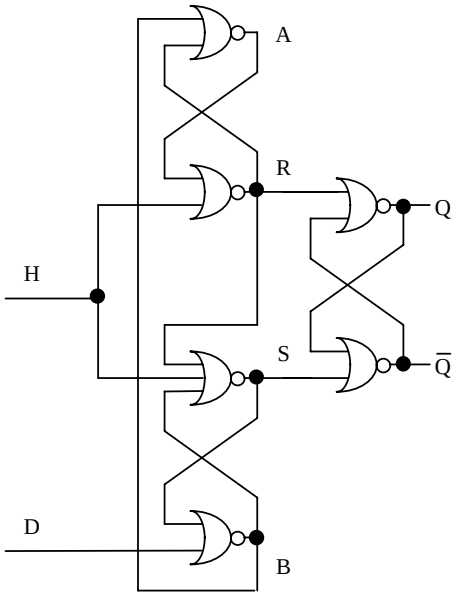
2. Même question quand il passe de 1 à 2.
3. On utilise maintenant le montage suivant :



Expliquer son fonctionnement. Quel est l'avantage par rapport au montage précédent ?

exercice 2.2

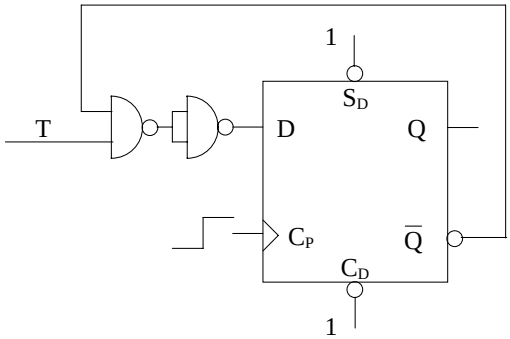
Soit le schéma suivant :



1. Indiquer l'état des points A, B, R, S, Q et $\overline{Q}$ quand H est à l'état haut, quand H passe de l'état haut à l'état bas, quand H est à l'état bas et quand H passe de l'état bas à l'état haut.
2. Quel type de bascule a-t-on réalisé ?

exercice 2.3

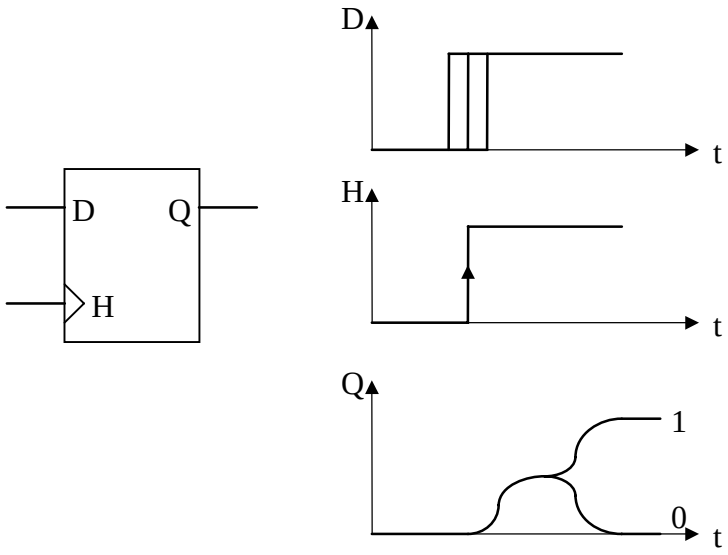
Soit le montage suivant composé d'une bascule D (SN74LS74) et de portes NAND (SN74LS00).



1. Dessiner l'allure du chronogramme en sortie de la bascule.
2. Rappeler la définition et donner la valeur des temps de setup, de hold et de propagation des circuits composant le montage.
3. Quelle est la fréquence maximale (dans le pire des cas) de fonctionnement de ce montage ?

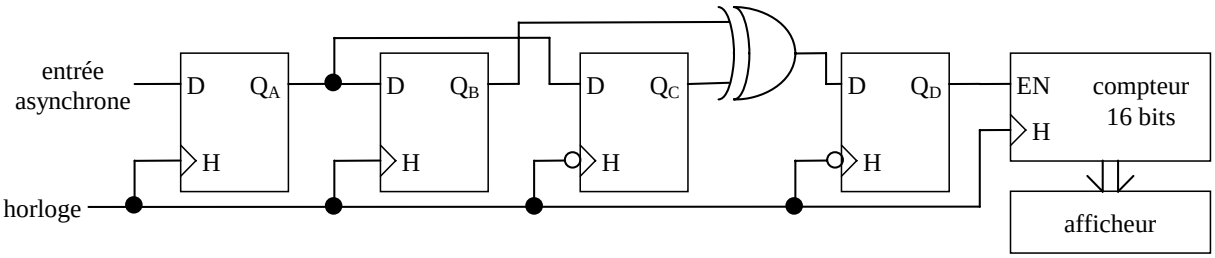
exercice 2.4

Que se passe-t-il si les temps de setup et de hold ne sont pas respectés quand un signal asynchrone est présent à l'entrée d'une bascule ? Sa sortie est alors dans un état indéterminé qui peut être un état intermédiaire compris entre 0 et 1, un état métastable. Ce phénomène est connu sous le nom de métastabilité. Le chronogramme suivant l'illustre.



Ce problème se pose systématiquement dès que l'entrée d'un système synchrone est asynchrone. Il y a alors forcément une chance pour que l'entrée asynchrone ne respecte pas les temps de setup et de hold d'une des bascules. La conséquence pratique de ce phénomène n'est pas l'apparition d'un état intermédiaire entre le 0 et le 1 logique. Au bout d'un certain temps, la bascule choisit un niveau et passe à 0 ou 1. C'est le temps au bout duquel elle basculera (si elle change d'état) qui est indéterminé. Ce qui est sûr, c'est qu'il est largement supérieur au temps de propagation de la bascule utilisée dans des conditions normales de fonctionnement.

Soit le montage suivant :



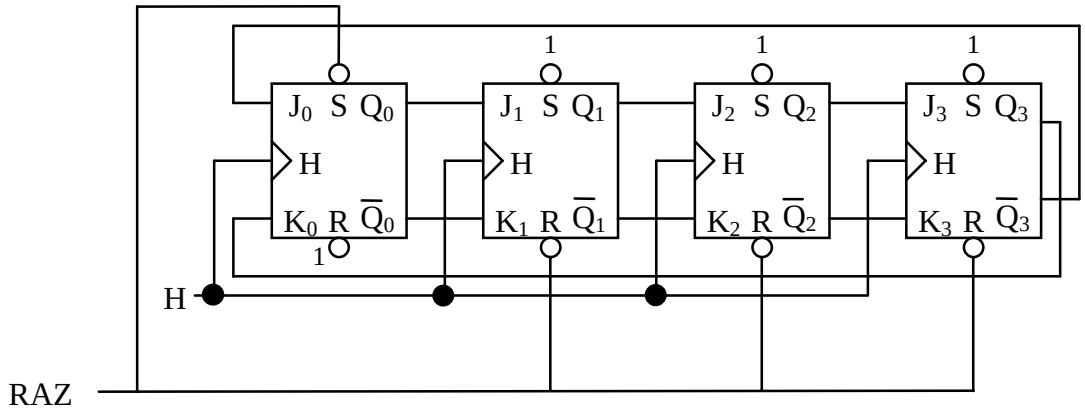
1. Expliquer, à l'aide des chronogrammes des différentes sorties, comment ce montage permet de quantifier le phénomène de métastabilité.
2. Le temps moyen entre deux erreurs d'acquisition (MTBF) ne peut être défini que de manière statistique. $MTBF = \frac{e^{K_2.t}}{F_1.F_2.K_1}$ avec F_2 = fréquence d'horloge, F_1 = fréquence moyenne du signal asynchrone, K_1 = constante = 0.1 ns, K_2 = constante dépendant de la

technologie de la bascule et t = temps de setup effectif. Cette formule n'est valable que si les deux fréquences sont indépendantes et sans corrélation.

On pose F_1 = 1 MHz, F_2 = 10 MHz et K_2 = 19.4 [1/ns]. Calculer le MTBF pour t = 0, 500 ps, 1 ns, 1.5 ns, 2 ns.

exercice 2.5

Soit le montage suivant :



La RAZ est asynchrone et active au niveau bas.

1. Dessiner le chronogramme des différentes sorties du montage en commençant par une RAZ.
2. Quelle est sa fonction ?

exercice 2.6

On se propose de synthétiser un décompteur synchrone modulo 5 (4, 3, 2, 1, 0, 4,...) codé en binaire naturel et réalisé avec des bascules JK commandées par les fronts montants de l'horloge H et des portes NAND. On désignera les sorties des bascules par Q_A, Q_B et Q_C où Q_A représente le bit de poids faible. Le mode de fonctionnement de chaque bascule est J = K.

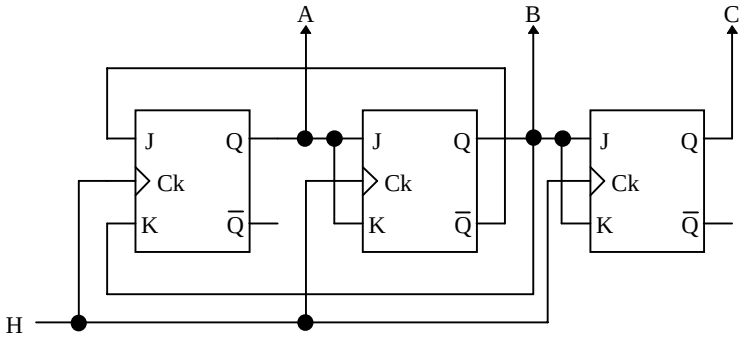
1. Remplir la table de vérité donnant Q_A^+, Q_B^+ et Q_C^+ en fonction de Q_A, Q_B et Q_C.
2. Donner les tableaux de Karnaugh des fonctions J_A, J_B et J_C sous la forme suivante :

Q_C \ Q_BQ_A	00	01	11	10
0				
1				

3. Donner les équations logiques simplifiées au maximum de J_A, J_B et J_C.
4. En déduire le logigramme du décompteur.
5. Que se passe-t-il si le décompteur est initialement chargé à l'état 5 ? Faire le graphe des états.
6. Modifier le schéma pour éliminer les cycles parasites et les états bloqués éventuels sans modifier le cycle principal.

exercice 2.7

On considère le circuit ci-dessous réalisé au moyen de bascules JK commandées par le front montant de l'horloge. La sortie est représentée par la valeur décimale correspondant au nombre binaire formé par les bits A, B et C où A est le bit de poids le plus faible.

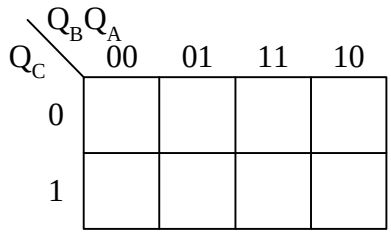


1. Etudier le fonctionnement du circuit. Etablir le graphe des états stables.
2. Ce système admet-il un ou plusieurs cycles parasites ? des états bloqués ?
3. Modifier le schéma pour éliminer les cycles parasites et les états bloqués éventuels sans modifier le cycle principal.

exercice 2.8

On souhaite réaliser un compteur 3 bits dont la séquence est la suivante : 0, 2, 3, 5, 6, 0, ... à l'aide de bascules T. On désignera les sorties des bascules par Q_A, Q_B et Q_C où Q_A représente le bit de poids faible.

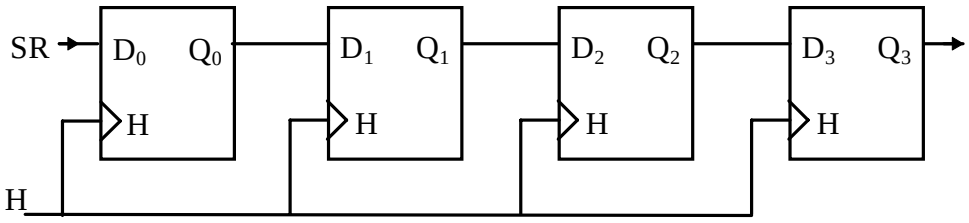
1. Remplir la table de vérité donnant Q_A^+, Q_B^+ et Q_C^+ en fonction de Q_A, Q_B et Q_C.
2. Donner les tableaux de Karnaugh des fonctions T_A, T_B et T_C sous la forme suivante :



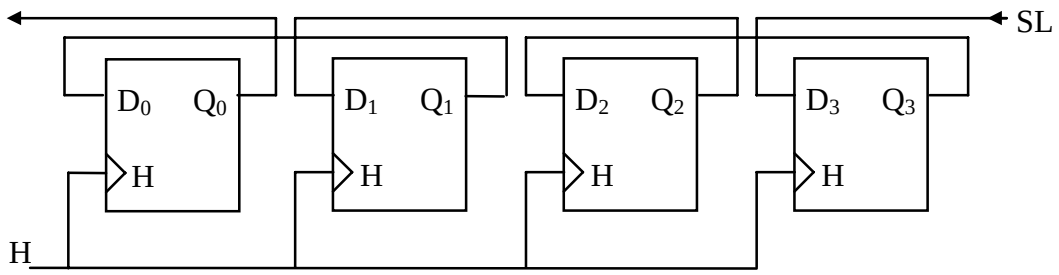
3. Donner les équations logiques simplifiées au maximum de T_A, T_B et T_C.
4. En déduire le logigramme du compteur.
5. Faire le graphe des états. Ce système admet-il un ou plusieurs cycles parasites ?
6. Modifier le schéma pour connecter l'état 7 à l'état 1 puis à l'état 3 (7→1→3) et l'état 4 à l'état 3 (4→3) sans modifier le cycle principal.

exercice 2.9

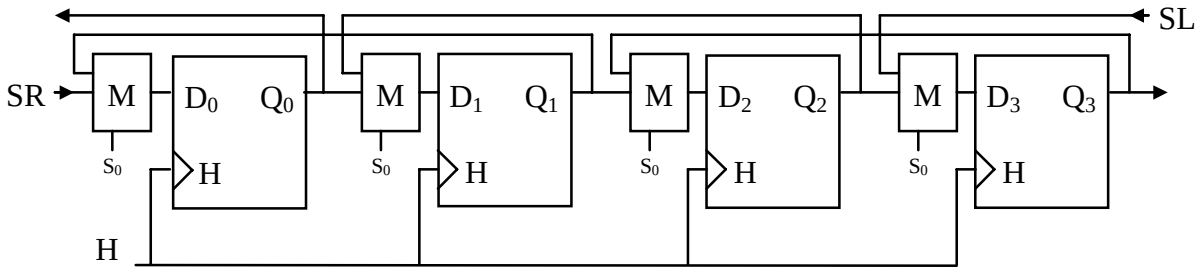
1. On considère le montage suivant. Quelle fonction a-t-on réalisée ?



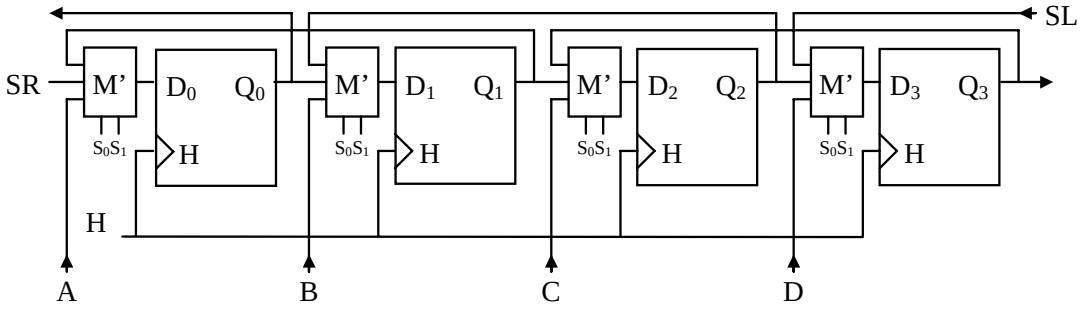
2. On considère le montage suivant. Quel fonction a-t-on réalisée ?



3. On combine les deux fonctions précédentes selon le schéma suivant. Effectuer la synthèse d'un opérateur M à l'aide de portes standards.



4. On désire ajouter une fonction de chargement du mot formé par A, B, C, D en Q_0, Q_1, Q_2, Q_3 selon le schéma suivant. Effectuer la synthèse d'un opérateur M' à l'aide de portes standards.



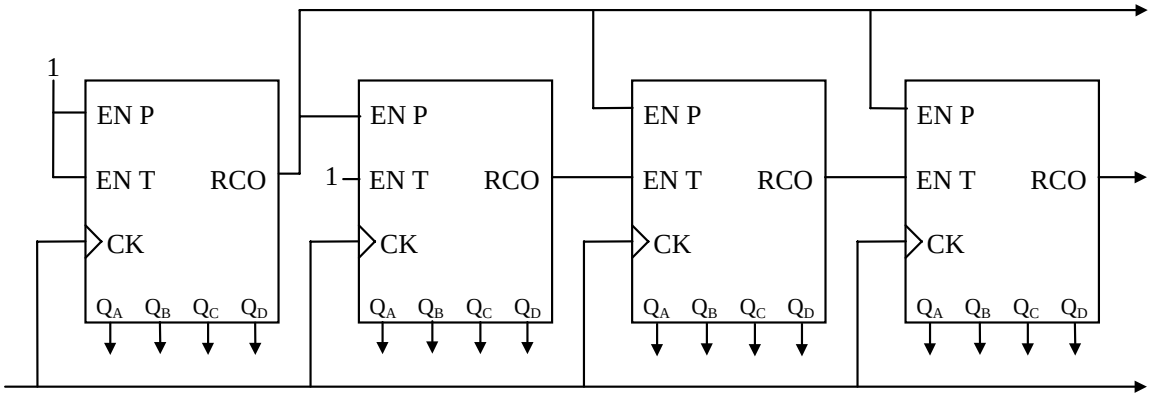
5. On désire ajouter au schéma précédent une fonction de validation EN. Proposer un schéma du nouveau montage ainsi que de la fonction M''.

Exercice 2.10

On étudie dans cet exercice la réalisation d'un compteur binaire ainsi que la meilleure manière d'associer des compteurs binaires en cascade. On n'utilisera que des portes combinatoires à deux entrées et des bascules D. On prendra comme référence les circuits SN74LS00 et SN74LS74.

1. Comment réalise-t-on une bascule T à partir d'une bascule D ? Quelle est sa fréquence maximale de fonctionnement ?
2. Comment réalise-t-on un compteur binaire 2 bits à partir de bascules T ? Quelle est sa fréquence maximale de fonctionnement ?
3. Même questions pour un compteur 3 bits, 4 bits, N bits. Conclusion.
4. On désire pouvoir associer en cascade ce compteur (4 bits). Quels types de signaux faut-il lui ajouter pour rendre cette association possible ? Quelle est la fréquence maximale de fonctionnement des circuits mis en cascade pour former un compteur 16 bits, 32 bits ?

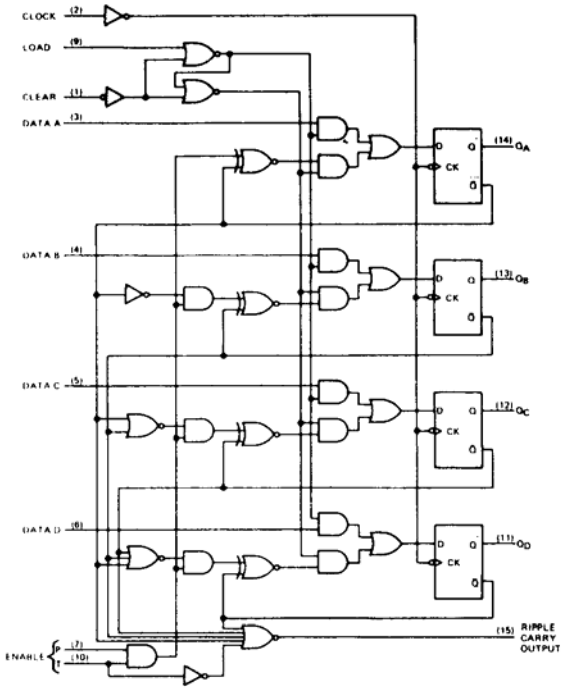
5. On souhaite améliorer la fréquence de fonctionnement maximale de ce montage. On va pour cela scinder le signal d'entrée de validation en deux signaux, EN P (enable parallel) et EN T (enable trickle). L'association a maintenant lieu selon le schéma suivant :



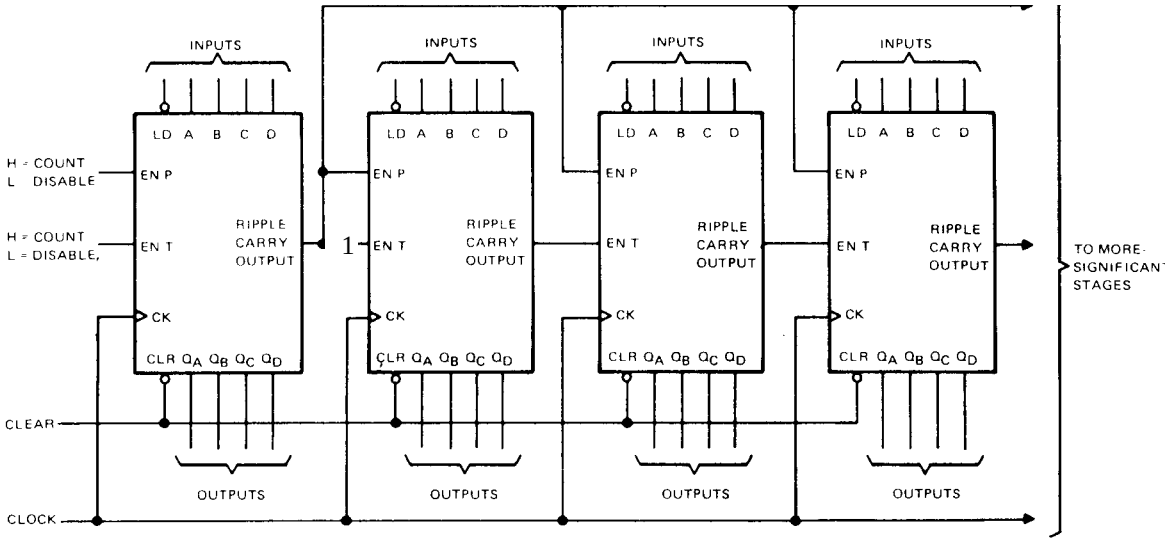
Proposer un schéma permettant de tirer parti des spécificités de la validation et de réaliser ce type d'association. Quelle est la fréquence maximale de fonctionnement des circuits mis en cascade pour former un compteur 16 bits, 32 bits ?

Exercice 2.11

Soit le schéma interne du compteur SN74LS163.



Ce circuit est conçu pour pouvoir être monté en cascade de la manière suivante :

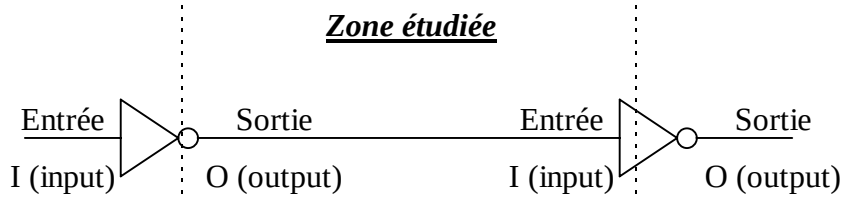


1. Comment réalise-t-on une bascule T à partir d'une bascule D ?
2. On pose load = 1, clear = 1 et enable P = enable T = 1. Supprimer du schéma de la bascule les portes inutiles et analyser le fonctionnement du montage. Pourquoi ne réalise-t-on pas directement des compteurs de grande taille avec cette méthode ? Quelle est alors la solution ?
3. On passe clear à 0. Analyser le fonctionnement du montage. La RAZ est-elle synchrone ou asynchrone ?
4. On passe clear à 1 et load à 0. Analyser le fonctionnement du montage. Le chargement est-il synchrone ou asynchrone ?
5. On passe load à 1 et enable P et T à 0. Analyser le fonctionnement du montage.
6. Supposons que l'on n'ait qu'un seul signal de validation. Quelle serait la limitation en fréquence de l'association en cascade (sur 16 bits et 32 bits) ?
7. Quel est alors l'avantage d'avoir deux signaux de validation ? Quelle est la limitation en fréquence de l'association en cascade (sur 16 bits et 32 bits) ?

3. Eléments de technologie des circuits logiques

3.1 Caractéristiques électriques des circuits logiques

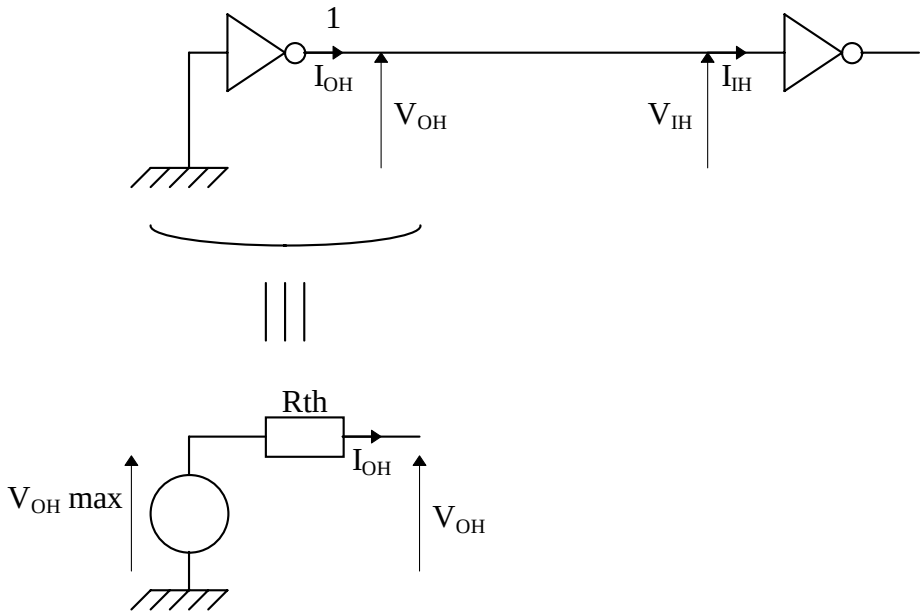
Nous allons maintenant étudier les principes de base utilisés pour caractériser électriquement les circuits logiques. Pour cela, nous allons considérer la zone située entre une sortie et une entrée de porte inverseuse.



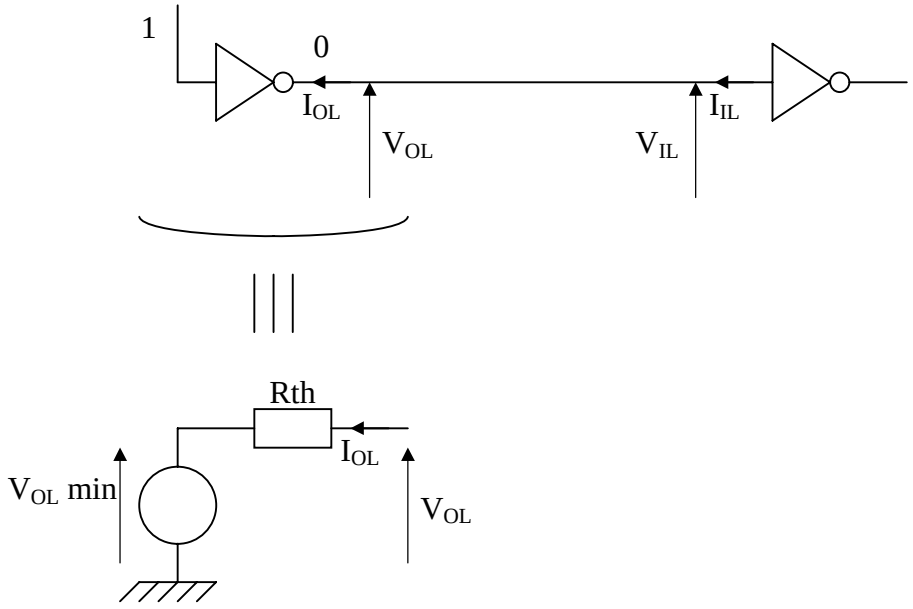
3.1.1 Tensions

Nous allons commencer par définir un schéma équivalent pour la sortie et l'entrée d'un circuit logique pour chaque niveau (les flèches définissent le sens réel des courants) :

- Une sortie à 1 est un générateur de tension. Sa tension à vide V_{OHmax} chute au fur et à mesure qu'on le charge. La valeur minimale permise de V_{OH} (V_{OHmin}) est atteinte lorsque la charge est maximale. Cette valeur ne doit pas être dépassée pour garantir le bon fonctionnement du circuit. On peut dessiner un schéma équivalent de Thévenin, dans lequel Eth représente V_{OH} à vide (ou peu chargé), et Rth la résistance interne. Rth varie un peu en fonction du courant de sortie I_{OH} (Rth est non linéaire) aussi ce schéma équivalent n'est qu'une approximation.

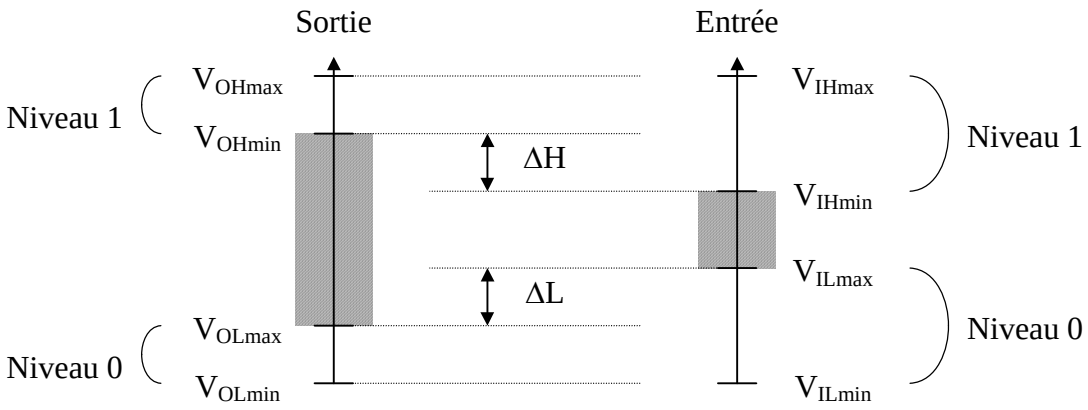


- Une sortie à zéro est un « récepteur ». La tension à ses bornes augmente au fur et à mesure qu'il absorbe du courant. La valeur maximale autorisée de V_{OL} (V_{OLmax}) est atteinte lorsque la charge est maximale. Cette valeur ne doit pas être dépassée pour garantir le bon fonctionnement du circuit. On peut dessiner un schéma de Thévenin très approché (valable en statique et pour des courants peu élevés).



- Les courants d'entrée peuvent être très différents selon le niveau. Un schéma équivalent général n'est pas possible. Pour certaines technologies (à base de transistors MOS), les courants d'entrée sont nuls en statique. Les courants d'entrée I_I sont généralement entrant au niveau haut et sortant au niveau bas. Ils ont obligatoirement des valeurs plus faibles que les courants de sortie équivalents. Si les tensions d'entrée dépassent les valeurs minimales et maximales autorisées, les courants d'entrée peuvent devenir très élevés (du fait des diodes de protection aux entrées). Le constructeur garantit les niveaux de tension suivants :
  $\Rightarrow$ si la tension à l'entrée de la porte est comprise entre V_{IHmin} et V_{IHmax}, la porte voit un niveau 1.
  $\Rightarrow$ si la tension à l'entrée de la porte est comprise entre V_{ILmin} et V_{ILmax}, la porte voit un niveau 0.

Compte tenu de ces schémas équivalents, les conditions de fonctionnement en tension d'un circuit logique connecté sur un autre circuit logique sont représentées par le schéma suivant :



Le niveau de sortie V_{OHmin} est toujours supérieur au niveau d'entrée V_{IHmin}. L'écart entre ces deux niveaux, $\Delta H = V_{OHmin} - V_{IHmin}$, est appelé marge de bruit à l'état haut. C'est le niveau de bruit maximal (crête) qui peut exister sur la sortie pour que l'entrée voie toujours un niveau 1. Le niveau de sortie V_{OLmax} est toujours inférieur au niveau d'entrée V_{ILmax}. L'écart entre ces deux niveaux, $\Delta L = V_{ILmax} - V_{OLmax}$, est appelé marge de bruit à l'état bas. C'est le niveau de bruit maximal (crête) qui peut exister sur la sortie pour que l'entrée voie toujours un niveau 0. Ce bruit est soit lié aux circuits numériques (ou analogiques dans le cas d'une carte mixte), soit ramené par l'alimentation ou la masse, soit capté par rayonnement.

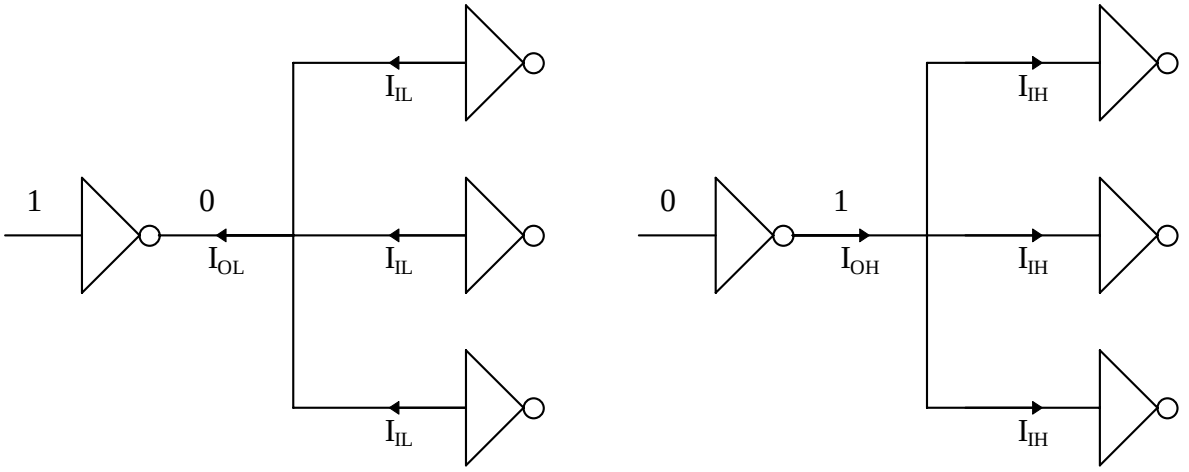
La zone hachurée doit, pour un circuit normal, être franchie rapidement (nécessité d'un temps de transition minimum), sous peine d'oscillations parasites. V_{IHmax} et V_{ILmin} sont des valeurs à ne pas trop dépasser sous peine de destruction ou de mauvais fonctionnement du circuit.

3.1.2 Courants

La charge maximale à la sortie de la porte, c'est-à-dire le nombre d'entrées de porte connectées sur cette sortie, doit être telle que V_{OH} reste supérieure à V_{OHmin}. On définit de cette manière la sortance qui est, dans le cas le plus défavorable (niveau haut ou bas selon le cas), le nombre maximum d'entrées de circuits pouvant être mis en sortie d'une porte. **Si on charge plus, le constructeur ne garantit plus à 100% le bon fonctionnement** avec la même immunité aux bruits. Cette garantie est donnée dans le cas le plus défavorable, c'est-à-dire pour une production de cartes logiques en grande série. Si la valeur de sortance n'est pas respectée, la probabilité d'avoir une panne aléatoire sous certaines conditions de tension d'alimentation ou de température n'est plus nulle avec le niveau de bruit indiqué. Plus la charge dépasse ce maximum et plus la probabilité d'avoir des ennuis augmente. En effet,

l'augmentation de la charge entraîne une diminution de l'immunité aux bruits (V_{OH} diminue et V_{OL} augmente) ainsi qu'une augmentation de la capacité parasite totale en sortie donc un allongement des temps de transition.

Le même raisonnement peut aussi être tenu pour les courants :

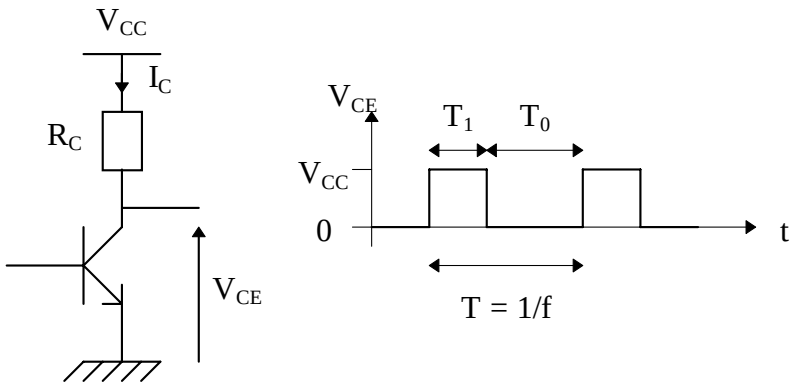


Le courant I_{OL} est égal à la somme des I_{IL} des portes connectées et ne doit pas dépasser I_{OLmax}. Le courant I_{OH} est égal à la somme des I_{IH} des portes connectées et ne doit pas dépasser I_{OHmax}. Le dépassement des valeurs maximales de courants de sortie correspond au dépassement des valeurs maximales (ou minimales) de V_{OH}. Concernant l'orientation des courants dans les documentations des constructeurs, on peut remarquer que très souvent, l'entrée est en convention récepteur et la sortie en convention générateur, pour les deux niveaux. On peut lire ainsi des valeurs négatives pour I_{OL} et pour I_{IL}. Le mieux est de connaître physiquement le sens du courant pour ne pas commettre d'erreur.

La sortance (fan out) est calculée en comparant les courants d'entrée et de sortie des portes au niveau haut et au niveau bas dans le cas le plus défavorable. Pour certaines familles (TTL notamment), une unité logique d'entrance (fan in) a été définie (I_{IH} = 40 µA, I_{IL} = 1,6 mA). Une entrance de 1 correspond à ces courants en entrée. Une porte ayant une sortance de 10 peut être chargée par 10 portes ayant une entrance de 1 ou 20 portes ayant une entrance de 0,5. Les constructeurs donnent souvent une sortance de 10 pour des portes de même famille technologique. Si la sortance d'une porte standard est insuffisante pour l'application, il faut utiliser une porte buffer qui permet d'attaquer un nombre de portes beaucoup plus important.

3.1.3 Puissance dissipée en fonction de la fréquence

La puissance dissipée en fonction de la fréquence peut être séparée en deux termes, la puissance statique (en continu ou en basse fréquence) et la puissance dynamique (au moment de la commutation). Illustrons ce phénomène sur un exemple. Soit le schéma très simplifié de la sortie d'un circuit logique :



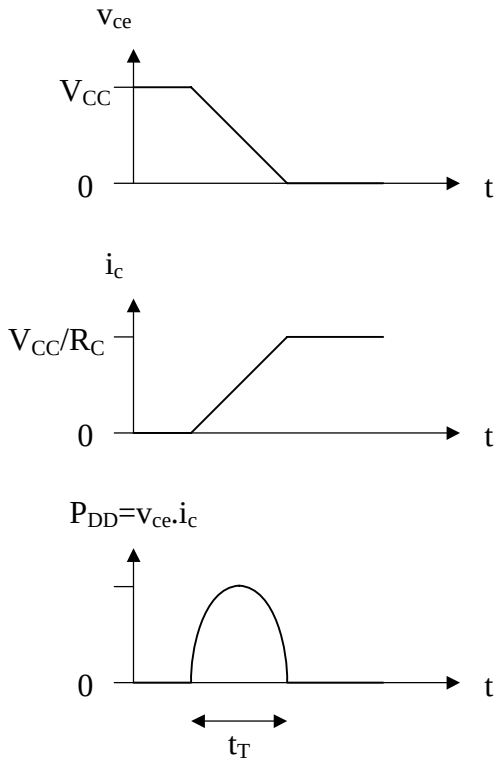
Nous allons supposer qu'à l'état 0, le transistor se comporte comme un court-circuit, et qu'à l'état 1, il soit équivalent à un circuit ouvert. La puissance statique dissipée à l'état 1 est alors nulle ($I_C = 0$), et la puissance statique dissipée à l'état 0 ($V_{CE} = 0$) vaut $P_{SD0} = \frac{V_{CC}^{\ 2}}{R_C}$. La puissance statique dissipée en basse fréquence est donc fonction du rapport cyclique du signal de sortie et vaut :

$$P_{SD} = P_{SD0} . \frac{T_0}{T}$$

Intéressons nous maintenant à ce qui se passe au moment de la commutation. Pendant la transition de durée t_T, le courant i_c et la tension v_{ce} sont non-nuls. L'énergie dissipée (surface hachurée sur la figure suivante) est égale à : $w = \int_{t_T} v_{ce} . i_c . dt$. En simplifiant dans cet exemple l'allure de v_{ce} et i_c, on obtient :

$$w = \int_{t_T} \left(V_{CC} - \frac{V_{CC} . t}{t_T} \right) . \left(\frac{V_{CC} . t}{R_C . t_T} \right) . dt = \frac{t_T . V_{CC}^{\ 2}}{6 . R_C}$$

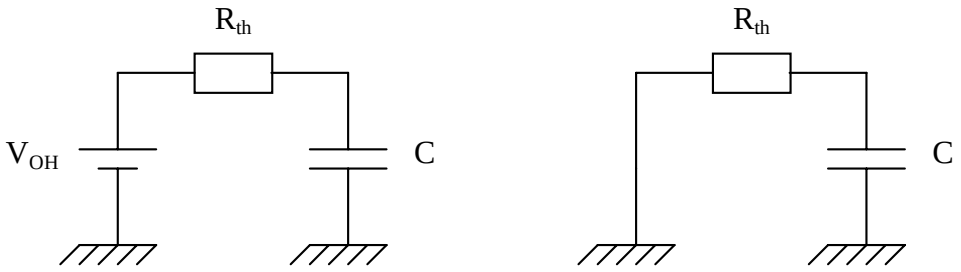
Ce qui nous donne le double sur une période. La puissance dynamique dissipée est donc égale à : $P_{DD} = 2.w.f = K.f$, c'est-à-dire proportionnelle à la fréquence.



La conclusion importante de cet exemple simplifié, valable pour tous les circuits logiques, est :

P dissipée = P statique (indépendante de f) + P dynamique (proportionnelle à f)

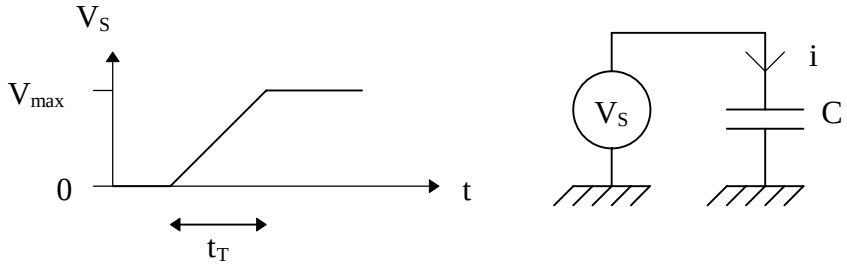
La puissance dissipée est aussi déterminée par la charge connectée sur la sortie du circuit logique. La résistance d'entrée d'un circuit logique est généralement très élevée et intervient peu dans les calculs. La charge vue par la sortie est généralement une capacité qui est la somme des capacités d'entrées des circuits connectés et de la capacité de la liaison entre la sortie et les entrées. En simplifiant le problème, on peut dessiner le schéma suivant :



L'énergie emmagasinée dans C pendant la transition $0 \rightarrow 1$ est égale à $w = \frac{1}{2}.CV_{max}^{2}$. Elle est aussi égale à l'énergie restituée pendant la transition $1 \rightarrow 0$. Donc, sur une période complète, $W_C = C.V_{max}^2$. La puissance dissipée pour charger et décharger C est égale à $P_{DDC} = C.V_{max}^2.f$. Ce phénomène rajoute un terme en K.f (proportionnel à la fréquence) à la puissance calculée précédemment et renforce donc l'importance de la puissance dynamique.

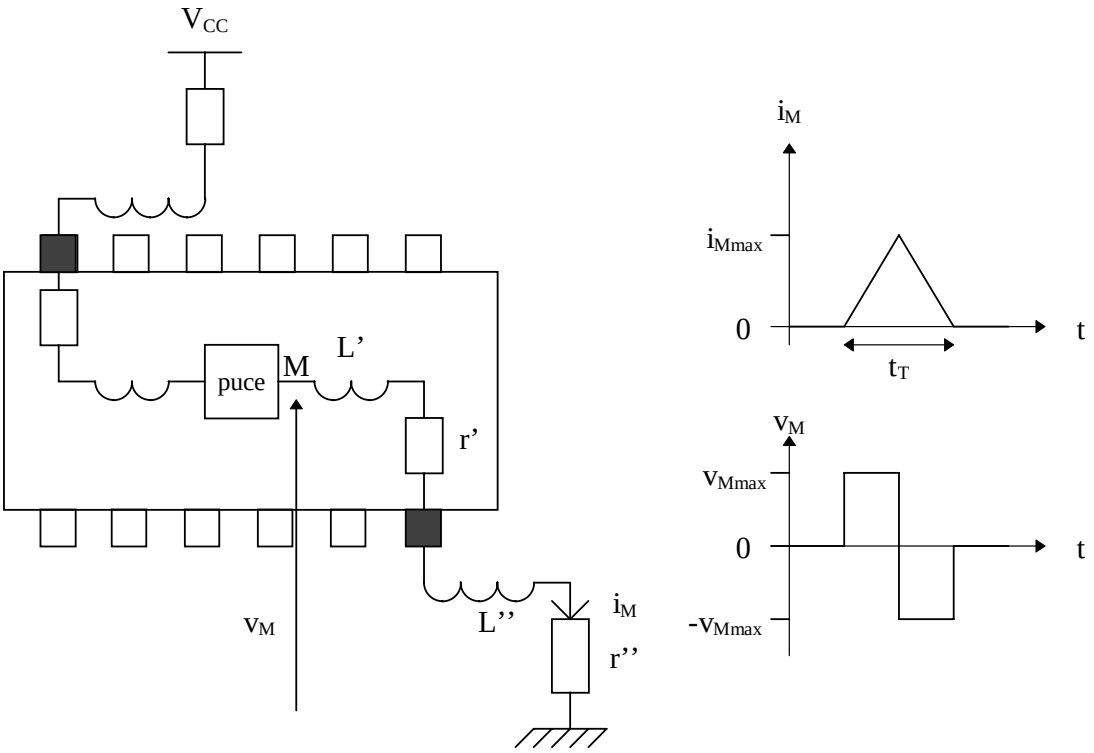
3.1.4 Nécessité d'un découplage en logique rapide

Nous allons maintenant étudier quelle est la quantité de courant nécessaire à la charge et à la décharge de la capacité précédente. Nous allons supposer, dans un calcul simplifié, que la charge se fait à courant constant avec une tension de la forme :

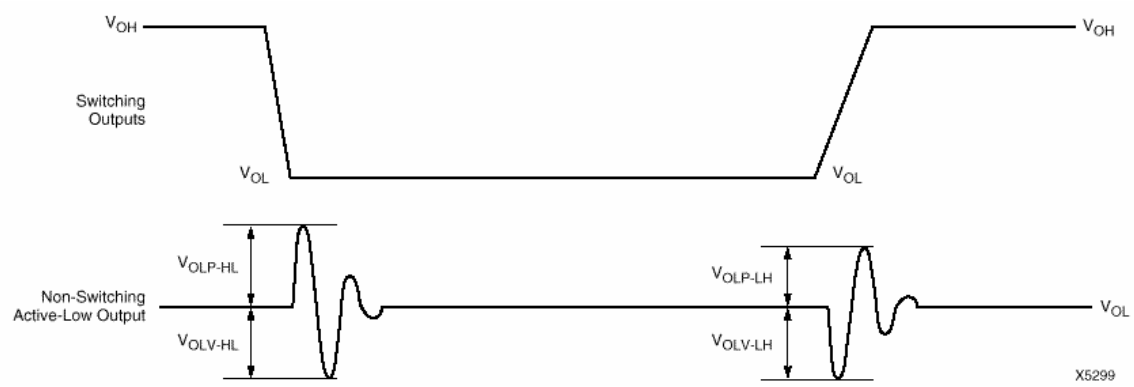


On a $i = C.\frac{dV_S}{dt}$, ce qui implique qu'au moment de la commutation, $i = C.\frac{V_{max}}{t_T}$ en supposant i constant durant la commutation. Par exemple, si C = 50 pF, V_{max} = 5V et t_T = 10 ns, on obtient i = 25 mA. Ce résultat doit être pris comme un ordre de grandeur valable quelle que soit la technologie. La conclusion à en tirer est que plus la technologie est rapide, et plus le courant à fournir est élevé, donc plus la puissance consommée est forte. D'autre part, à courant équivalent, plus l'excursion de tension V_{max} est élevée et plus le temps de transition augmente.

Lorsque plusieurs sorties commutent simultanément (SSO : Simultaneously Switching Outputs, cas d'un bus de données ou d'un bus d'adresses par exemple), il se produit un pic de courant élevé qui doit être fourni par l'alimentation du circuit. Or les liaisons vers V_{CC} et vers la masse d'un circuit présentent des résistances et des inductances parasites :



La masse vue par le circuit est le point M et on a $v_M = r.i_M + L.\frac{di_M}{dt}$ par rapport à la masse générale de la carte (avec L = L' + L'' et r = r' + r''). Si on reprend les valeurs de l'exemple précédent avec 10 sorties qui commutent simultanément et r = 0,01 Ω, L = 50 nH, i_{Mmax} = 250 mA et t_T = 10 ns, on a v_{Mmax} = 2,5 V ce qui est largement suffisant pour faire basculer une porte qui devrait rester à un niveau constant 0. Le phénomène réel provoque sur la masse une oscillation qui a la forme suivante :



On s'intéresse plus à la masse qu'à V_{CC}, non parce que les marges de bruit au niveau haut sont parfois supérieures aux marges de bruit au niveau bas, mais surtout parce que les potentiels des niveaux d'entrées et de sorties des circuits sont référencés par rapport à la masse. Avec les valeurs usuelles de r, seule l'inductance joue un rôle. Elle est due à l'inductance interne du boîtier et à l'inductance de la liaison entre le boîtier et la masse générale de la carte. On distingue deux effets possibles :

- Le pic positif peut poser un problème avec un signal quittant le circuit parasité et allant sur l'entrée d'une autre porte. Ce pic positif s'ajoute au niveau bas et, selon sa durée, peut être interprété comme une impulsion positive.

- Le pic négatif peut poser un problème avec un signal arrivant sur le circuit parasité. Ce pic négatif rend la masse du boîtier négative et diminue la marge de bruit à l'état bas. Si la tension au niveau bas à l'entrée du circuit n'est pas assez proche de 0, elle peut être interprétée comme une impulsion positive.

Afin de réduire ce phénomène, il faut respecter les règles suivantes en logique rapide :

- utiliser un circuit imprimé avec un plan de masse et un plan d'alimentation (si possible) connectés directement aux bornes des circuits intégrés. En l'absence de plan de masse, un maillage astucieux des lignes d'alimentation et de masse permet de le remplacer. Dans la mesure du possible, laisser le plan de masse intact, une faible modification (rangée de trous, fente) peut avoir un effet désastreux.

- Découpler tous les circuits intégrés rapides avec un ou plusieurs condensateurs placés au plus près du boîtier. Les condensateurs de découplage servent de réservoir de courant et fournissent au circuit rapide le pic de courant nécessaire à la commutation. La valeur du condensateur n'a pas à être élevée (10 nF suffit) mais il ne faut pas utiliser de condensateurs chimiques à cause de leur mauvais comportement en hautes fréquences ni de condensateurs ayant une inductance parasite trop élevée.

- Le choix du type de boîtier a son importance. En terme d'inductance parasite, les boîtiers PGA et DIP sont les plus mauvais, les boîtiers QFP sont les meilleurs et les boîtiers PLCC

se trouvent entre les deux (voir le §3.5 sur les boîtiers utilisés pour encapsuler les circuits logiques).

- Les autres règles à respecter (placement des entrées sensibles, répartition des sorties qui commutent sur le boîtier) dépendent du type de circuit. Il faut se reporter aux données constructeur.

3.2 Spécifications

En fonction de leur application, les circuits commercialisés ont les plages de fonctionnement en température ambiantes suivantes :

- gamme commerciale : 0 < T < 85 °C ou bien 0 < T < 70 °C,
- gamme industrielle : -40 °C < T < +100 °C,
- gamme militaire : -55 °C < T < +125 °C.

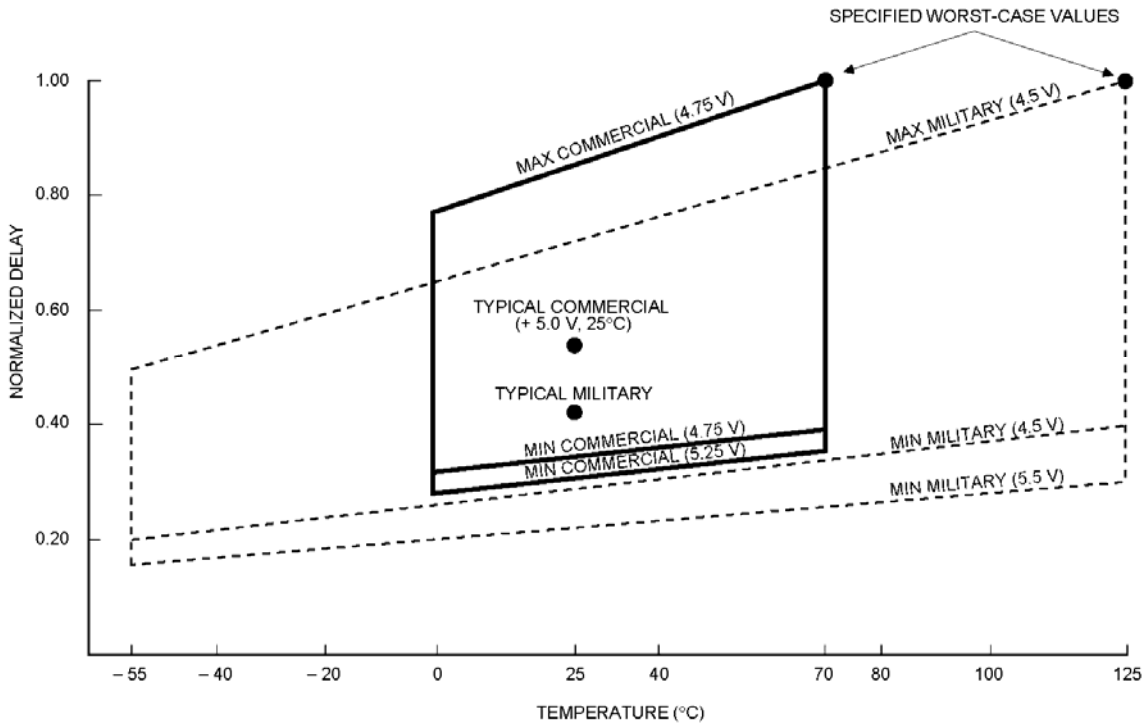
Les gammes militaires sont de moins en moins souvent commercialisées. On peut observer sur les feuilles de caractéristiques constructeur que les spécifications du circuit sont données en valeur minimale (min), typique (typ) et maximale (max). Ces valeurs min, typ et max représentent des phénomènes différents selon que l'on s'intéresse aux tensions et courants ou aux temps et fréquences :

- La tension d'alimentation du circuit peut varier entre V_{CCmin} et V_{CCmax}. Les valeurs min et max des tensions et courants d'entrée et de sortie sont spécifiées afin de respecter les niveaux de marge de bruit.

- Les temps et les fréquences min, typ et max correspondent à la dispersion des caractéristiques des composants sur une grande série. Cette dispersion est fonction de trois facteurs qui sont, par ordre d'importance :
  1. Les conditions de fabrication.
  2. La température de fonctionnement.
  3. La valeur de la tension d'alimentation.

L'écart entre valeur minimale et valeur maximale peut aller du simple au quadruple. Les temps max sont une garantie de fonctionnement dans le pire des cas. Une carte développée en

prenant en compte ces valeurs marchera dans 100% des cas. Si les valeurs typiques sont utilisées pour la conception, alors le fonctionnement n'est plus garanti. Le paramètre de fréquence maximale du circuit, fmax, doit être pris avec beaucoup de précaution. Il s'agit souvent de la fréquence maximale du circuit fonctionnant seul, sans charge. Vous trouverez sur la figure suivante un exemple de temps de propagation normalisé pour un circuit CMOS. Vous remarquerez le facteur 4 entre temps min et max ainsi que le facteur 2 entre typique et max.

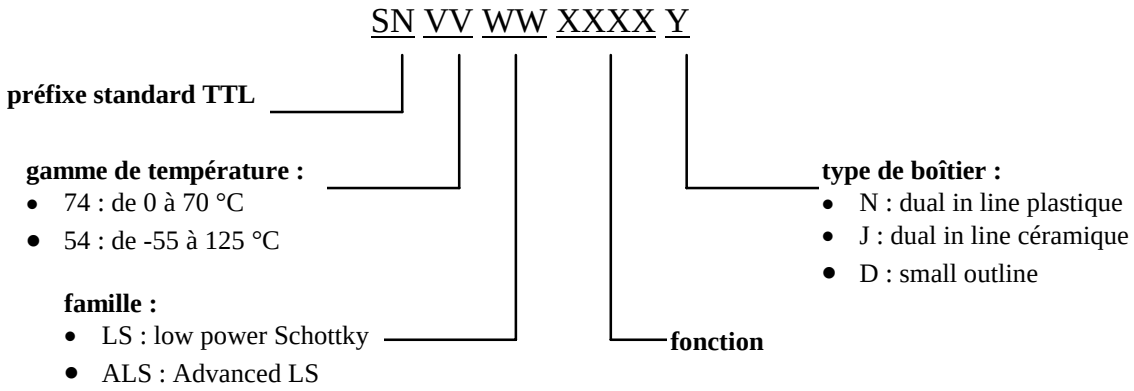


On peut noter les variations suivantes sur cet exemple (série commerciale) :

- $\frac{\Delta t_p}{\Delta T} = 0{,}35$ % par degré (à Vcc = 5 V).
- $\frac{\Delta t_p}{\Delta V_{CC}} = -2{,}5$ % par 100 mV (à T = 25 °).
- $\frac{\Delta t_p}{\Delta \text{fabrication}} = 145$ % entre minimum et maximum.

Il s'agit d'un ordre de grandeur tout à fait courant.

L’appellation du circuit donne un certain nombre de renseignements mais elle varie avec les fabricants de composants. L’exemple suivant est valable uniquement pour les circuits TTL :



Exemple : un boîtier SN54LS00N est un circuit combinatoire quadruple NAND à deux entrées en technologie LS et boîtier plastique dual in line. Sa gamme de température est comprise entre -55 °C < T < +125 °C.

3.3 Familles de circuits logiques

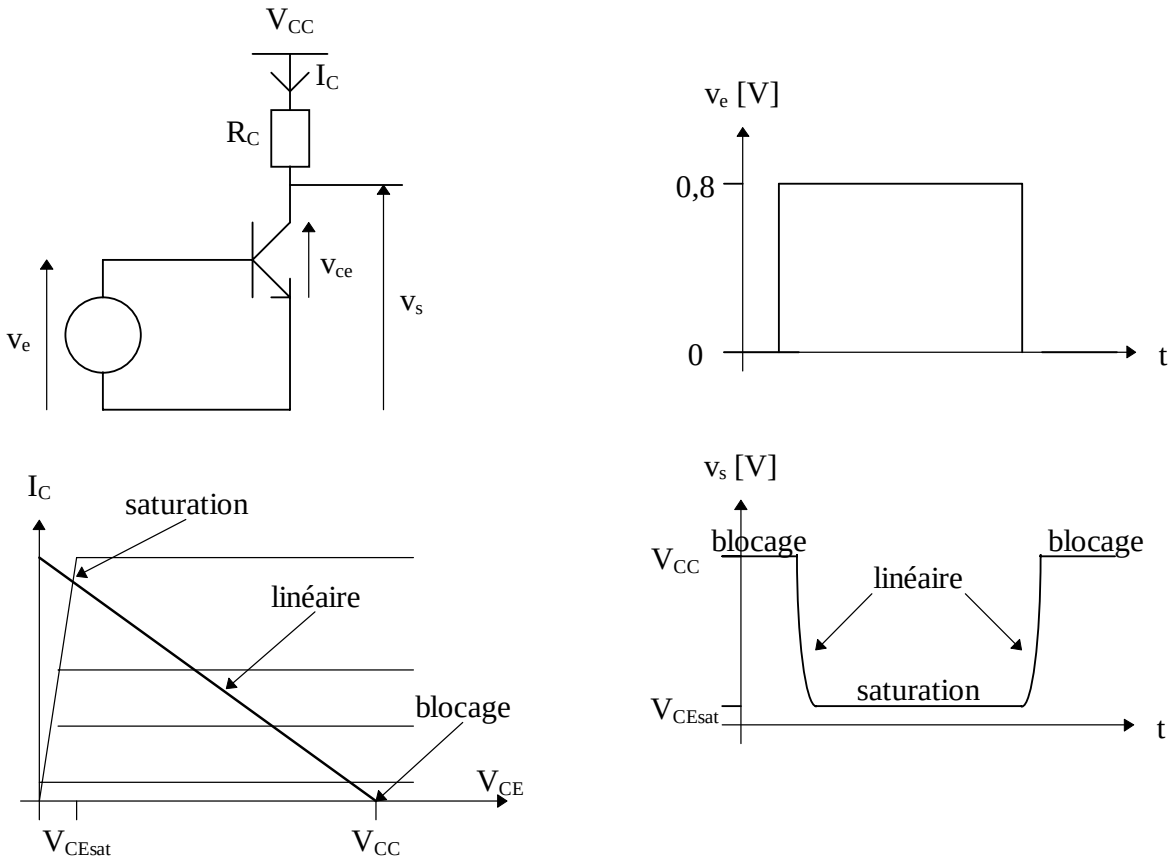
Quand un concepteur de cartes logiques doit développer un nouveau produit, il est bon qu’il examine l’ensemble des circuits commercialisés. Le choix de la technologie adaptée au besoin peut s’avérer périlleuse en raison de la grande diversité des produits existants sur le marché. Bien qu’il y ait beaucoup de familles logiques disponibles, elles peuvent être approximativement divisées en trois catégories : TTL (Transistor-Transistor Logic), CMOS (Complementary Metal-Oxyde Semiconductor logic) et ECL (Emitter-Coupled Logic). La TTL et l’ECL sont des technologies bipolaires alors que la CMOS est une technologie utilisant des transistors MOS complémentaires.

3.3.1 Technologie TTL

Le terme bipolaire se réfère à l’élément de base utilisé qui est le transistor bipolaire. Depuis le circuit originel TTL, de nombreuses améliorations ont été employées en vue notamment de réduire la consommation et d’augmenter la vitesse de cette technologie. La série L (Low power) pour réduire la consommation, la série S qui utilise des diodes Schottky pour augmenter la vitesse puis enfin la série LS qui combine les avantages des deux précédentes. Seule la TTL LS (Low power Schottky) est encore commercialisée aujourd’hui. Elle a évolué avec la série ALS (Advanced Low power Schottky) et la série FAST (Advanced Schottky).

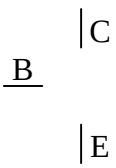
La vitesse élevée de la TTL et la basse consommation de la CMOS ont aussi été combinées dans la famille ABT (Advanced BiCMOS) qui utilise la TTL pour les entrées-sorties et la CMOS pour l'intérieur du circuit. Pour pouvoir étudier le fonctionnement d'une porte TTL, il faut d'abord voir le fonctionnement du transistor bipolaire en commutation.

3.3.1.1 Le transistor en commutation



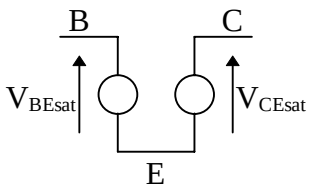
Le transistor bipolaire en commutation peut avoir trois modes de fonctionnement :

1. Le régime de blocage. V_{BE} et I_B sont nuls. Le courant collecteur I_C est peu différent de 0. Le transistor se comporte comme un circuit ouvert ($V_{BE} < 0.5$, $I_C = I_B = I_E = 0$). Son schéma équivalent est le suivant :



2. Le régime linéaire ou passant. C'est le mode de fonctionnement en amplificateur déjà étudié précédemment. $I_C = \beta.I_B$. $V_{BE} \approx 0,5$ V.

3. Le régime de saturation. Le courant de base est tel que le courant de collecteur est maximum. 0,5 V < V_{BE} < 0,8 V. $V_{CE} = V_{CEsat}$ = 0,2 V. La saturation commence quand I_C devient inférieur à $\beta.I_B$. Son schéma équivalent est le suivant :



Afin d'analyser un montage à base de transistor bipolaire fonctionnant en régime bloqué-saturé, on utilise la méthode suivante :

- on fait l'hypothèse la plus plausible sur l'état des transistors du montage (bloqué, saturé ou passant) compte tenu des tensions et courants estimés du montage.
- on les remplace dans le montage par leur schéma équivalent.
- on calcule les courants et les tensions dans le circuit.
- on calcule le V_{BE}, I_B et I_C pour chaque transistor et on vérifie les hypothèses de départ en utilisant les propriétés de chaque régime, c'est-à-dire :

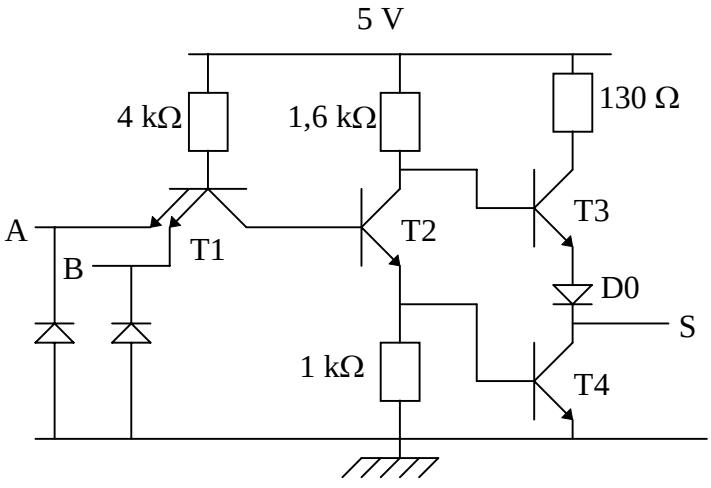
  $\Rightarrow$ Saturation : $I_C < \beta_{on}.I_B$.

  $\Rightarrow$ blocage : V_{BE} < 0,5 V ou $I_B = 0$.

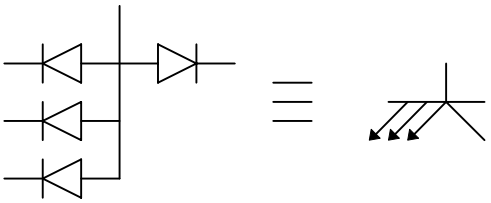
  $\Rightarrow$ passant : si on ne se trouve dans aucun des deux cas précédents.

3.3.1.2 La porte TTL totem pole

C'est la première structure vraiment efficace utilisée en logique TTL standard. Prenons l'exemple d'une porte NAND à deux entrées de type SN7400 :



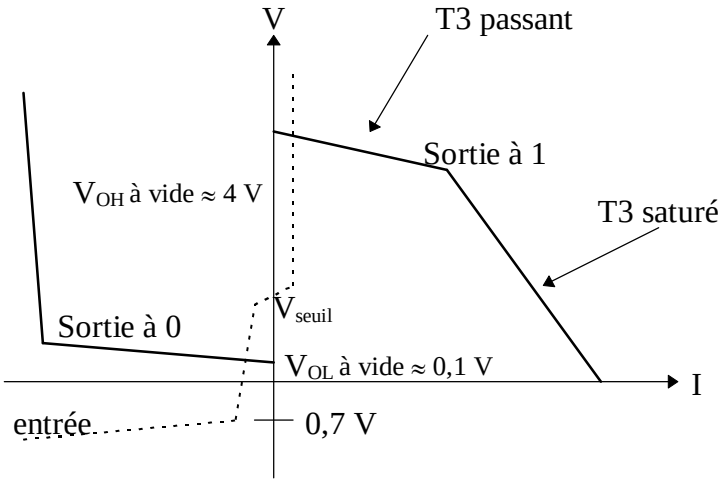
Le transistor T1 est un « transistor » multi-émetteur. Ce n'est pas vraiment un transistor, mais plutôt un aiguilleur de courant dont le schéma équivalent est, par exemple pour un multi-émetteur à trois entrées :



Le fonctionnement logique est le suivant :

- pour A et B au niveau 1, T1 aiguille du courant vers T2, T2 et T4 sont saturés. T3 est bloqué. S est au niveau 0 et peut recevoir du courant.
- pour A ou B au niveau 0, T1 n'aiguille plus de courant vers T2 mais vers l'entrée au niveau 0. T2 et T4 sont bloqués. La sortie S est au niveau 1 et peut fournir du courant. Suivant la quantité de courant sortant du montage, T3 est soit passant, soit saturé.

Les caractéristiques électriques d'entrée et de sortie sont les suivantes :



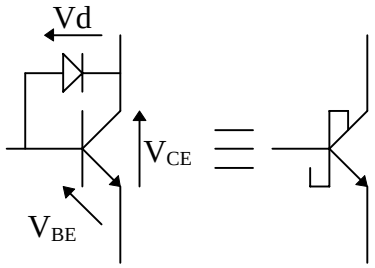
Une entrée en l'air peut être considérée comme un niveau 1 (le courant n'est aiguillé vers une entrée que si elle est à 0). Cela n'est vrai que pour la logique TTL. Toutefois, afin d'éviter des parasites, il est préférable de ne jamais laisser en l'air des entrées non-utilisées.

Il existe une forte dissymétrie des courants d'entrées au niveau haut et au niveau bas ($I_{IHmax} \approx 10\ \mu A$, $I_{ILmax} \approx 1$ mA). Si on veut imposer un niveau bas à l'entrée d'une porte avec une résistance connectée à la masse, il faut tenir compte du courant d'entrée pour que V_{ILmax} ne soit pas dépassée (quelques centaines d'Ohms en pratique). D'après les feuilles de caractéristiques, on relève pour une SN7400 ($V_{CC} = 5$ V):

en entrée	V_{IHmin}	=	2 V	en sortie	V_{OHmin}	=	2,4 V
	V_{ILmax}	=	0,8 V		V_{OLmax}	=	0,4 V
	I_{IHmax}	=	40 μA		I_{OHmax}	=	0,4 mA
	I_{ILmax}	=	1,6 mA		I_{OLmax}	=	16 mA

Les immunités au bruit valent donc : $\Delta H = V_{OHmin} - V_{IHmin} = 0{,}4$ V et $\Delta L = V_{ILmax} - V_{OLmax} = 0{,}4$ V. La sortance à l'état haut vaut $\frac{I_{OHmax}}{I_{IHmax}} = 10$ et la sortance à l'état bas vaut $\frac{I_{OLmax}}{I_{ILmax}} = 10$.

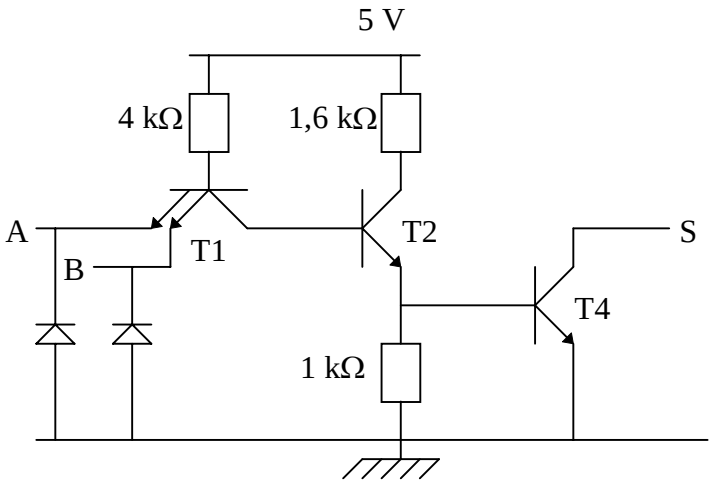
La limitation principale en terme de vitesse de cette technologie tient à la saturation profonde du transistor de sortie T4. Le temps nécessaire pour désaturer le transistor est trop important pour atteindre une vitesse élevée. Pour éviter ce phénomène, on fait appel à des transistors sur lesquels sont placés des diodes « Schottky » (diodes à jonction métal-semiconducteur très rapides et à faible tension de seuil $Vd \approx 0{,}2$ V). Ce sont les transistors « Schottky ».



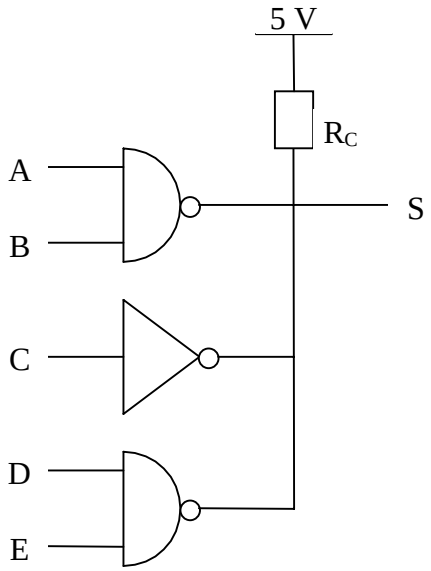
V_{CE} ne peut pas devenir inférieure à $V_{BE} - Vd \approx 0{,}5$ V, évitant ainsi la saturation. Cela donne la technologie S (pour Schottky) très rapide mais qui a pour inconvénient une consommation très élevée.

3.3.1.3 La porte TTL à collecteur ouvert

Il existe une variante de la porte précédente dite « à collecteur ouvert ». La charge du transistor de sortie ne se trouve plus dans le circuit mais à l'extérieur, sur la carte.

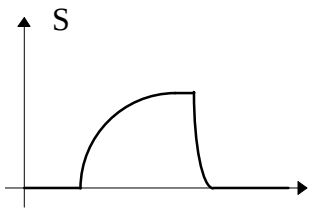


On peut ainsi relier plusieurs sorties pour réaliser des fonctions logiques avec une seule résistance de rappel R_C. Par exemple, on réalise la fonction $S = \overline{A.B}.\overline{C}.\overline{D.E} = \overline{A.B + C + D.E}$ avec le schéma suivant :



On réalise de cette manière une fonction ET câblée en logique positive (ou bien un OU en logique négative). On peut aussi utiliser la porte collecteur ouvert pour commander une diode électroluminescente (LED) ou un relais. Les principaux inconvénients de cette porte sont :

- la dissymétrie des temps de commutation : la charge étant généralement capacitive, le front montant est une charge de condensateur à travers R_C. Le front descendant est une décharge à travers le transistor de sortie qui absorbe beaucoup plus de courant que ne peut en fournir R_C.



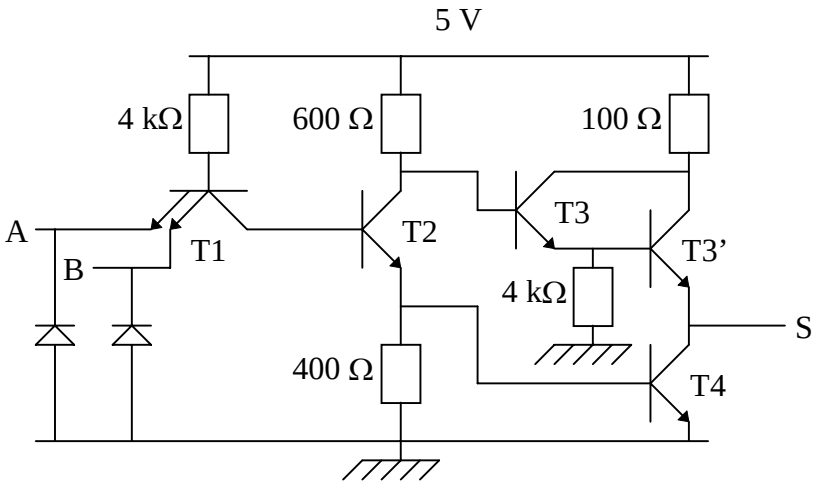
- la consommation de courant est élevée au niveau 0 (≈5 V sur R_C).

La valeur de R_{Cmin} est déterminée par $R_{Cmin} = \frac{V_{CC} - V_{OLmax}}{I_{OLmax}} = 287\ \Omega$. La valeur de R_{Cmax} est déterminée par la valeur maximale du front de montée souhaité.

3.3.1.4 La porte TTL avec buffer

Pour renforcer la sortance en vue de commander un plus grand nombre de portes, on remplace le transistor T4 de la porte totem pole par un montage Darlington qui augmente la quantité de courant disponible à l'état haut. Il existe deux familles de circuits à sortance améliorée :

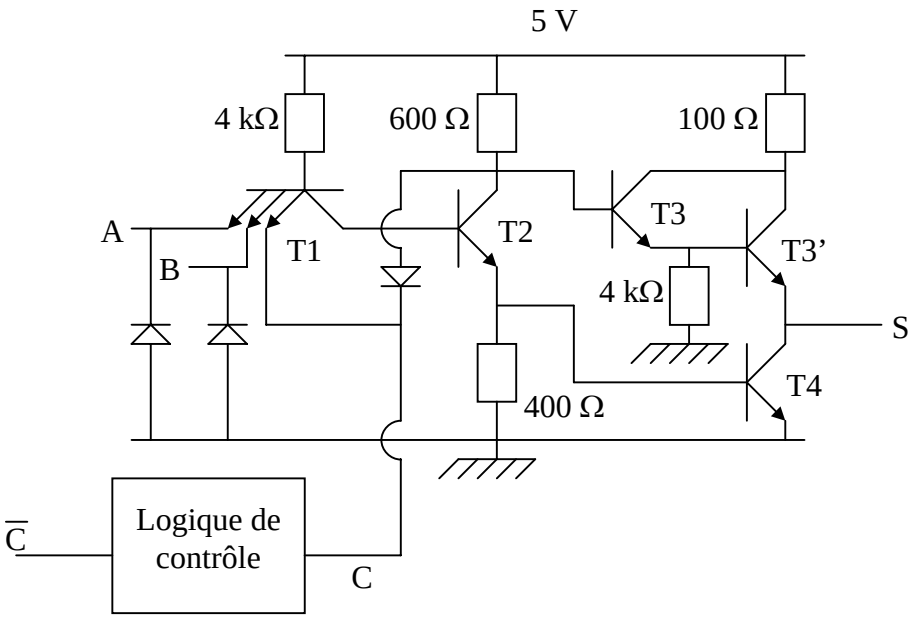
- Les drivers de ligne. Ils fournissent suffisamment de courant pour pouvoir attaquer une ligne 50 Ω adaptée.
- Les buffers. Ils fournissent suffisamment de courant pour pouvoir attaquer plusieurs dizaines de portes comme par exemple dans le cas d'une ligne d'horloge.



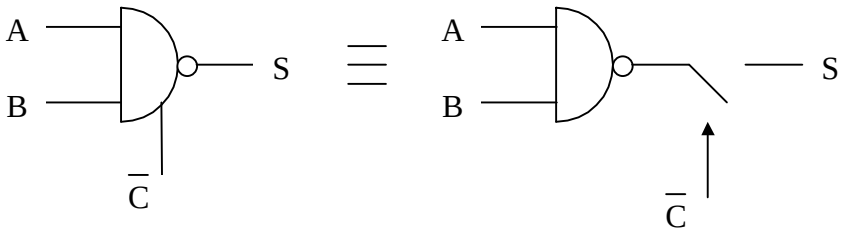
3.3.1.5 La porte TTL 3 états – notion de bus

3.3.1.5.1 La porte 3 états

Un signal de commande C est ajouté à la porte totem pole afin de mettre la sortie S à l'état haute impédance (S est en l'air). Quand C = 0, les transistors de sortie sont tous bloqués et la sortie est flottante. Quand C = 1, la porte NAND fonctionne normalement.



Le schéma logique équivalent est le suivant :

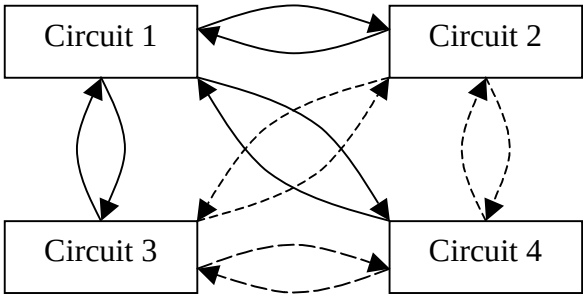


Quand la sortie S est à l'état haute impédance, on dit qu'elle est à l'état Z (ou HiZ).

3.3.1.5.2 notion de bus

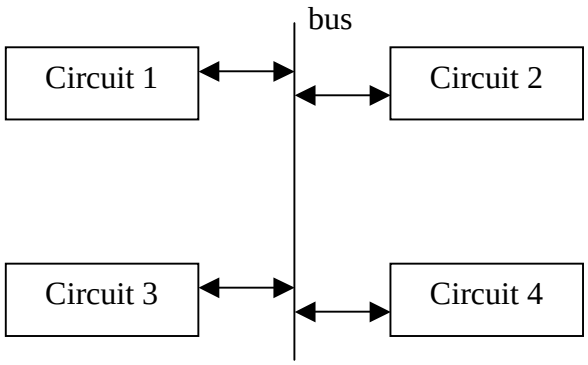
Une question fondamentale de l'électronique numérique est : comment faire dialoguer plusieurs circuits intégrés numériques entre eux de la manière la plus efficace possible ? Deux solutions sont possibles : la liaison point à point ou le bus.

1. **La liaison point à point**. Pour chaque liaison, on tire une paire de fils, un fil qui envoie des données, l'autre fil qui en reçoit. Chaque circuit étant relié indépendamment avec ses voisins, le nombre de fils augmente très rapidement. Voici un exemple avec 4 circuits.

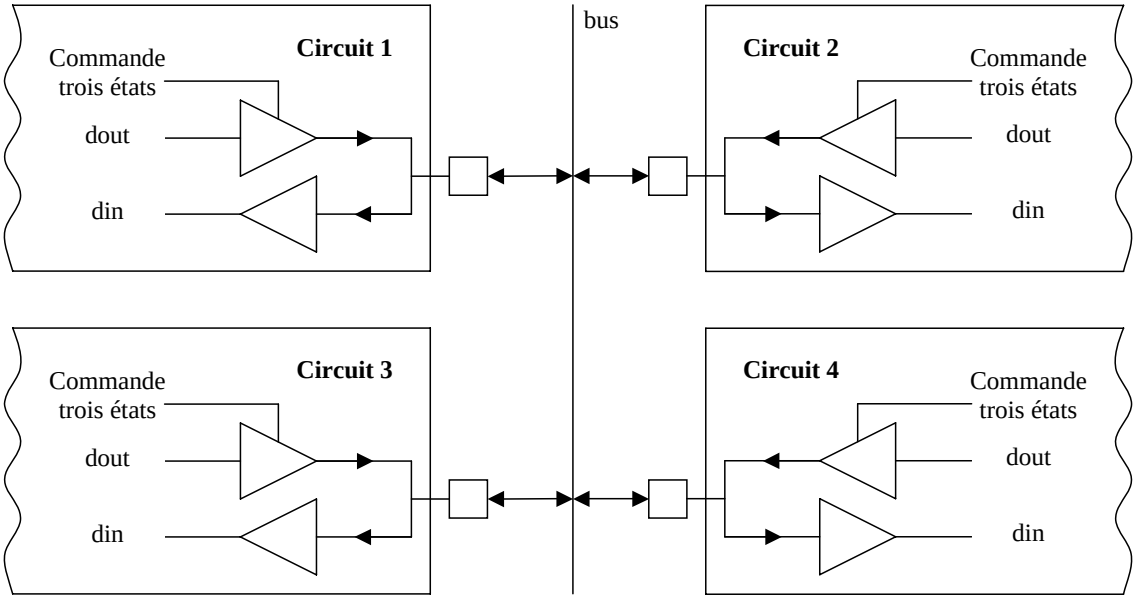


On peut compter 1 + 2 + 3 = 6 paires de fils pour assurer la liaison point à point. Dans le cas général, avec N circuits, on a 1 + 2 + 3 + 4 + …+ N-2 + N-1 liaisons ce qui donne $\frac{N^2 - N}{2}$ paires de fils pour N circuits. On voit que si on veut transmettre 32 bits en même temps, il faut 32 paires par liaison et le nombre de fils à tirer entre les circuits augmente comme 64 x (nombre de circuits)^2. Cette solution n'est pas valable sauf si on limite le nombre de circuits (5 ou 6 maximum) et le nombre de fils par paire (2 ou 4 au maximum). Mais le débit devient alors trop faible sauf si on augmente massivement la fréquence de transfert. C'est le principe retenu pour PCI-Express (ou 3GIO). La fréquence de transfert est égale à 2,5 GHz et il y a entre 1 et 16 paires différentielles par liaison. **C'est un bus série point à point, la bande passante n'est pas partagée entre les circuits.**

2. **Le bus**. Dans ce cas, tous les circuits sont branchés simultanément sur le même fil.



Tous les circuits ont la capacité de générer un état sur le bus ou bien de lire une valeur. Pour que cela soit possible, il faut que chaque broche soit bidirectionnelle, c'est-à-dire qu'elle fonctionne aussi bien en entrée qu'en sortie. Il faut donc connecter en parallèle un buffer d'entrée avec un buffer de sortie. Si on veut que l'entrée puisse lire une valeur envoyée par un autre circuit, il faut obligatoirement que le buffer de sortie cesse de générer un niveau sur le fil. La solution consiste à faire passer ce buffer à l'état haute impédance.



Quand un circuit veut envoyer un bit sur le bus, les autres circuits doivent mettre leur sortie à l'état Z et lire la valeur. Si deux circuits essayent de générer un niveau opposé en même temps, il y a conflit de bus et l'état devient indéterminé. Il y a donc forcément un maître de bus (par exemple, un microprocesseur) indiquant qui a le droit de générer sur le bus et qui force les autres circuits à passer à l'état Z. En pratique, **un bus est composé de N fils de même nature**. Un bus d'adresses par exemple qui sera unidirectionnel ou encore un bus de données qui sera lui bidirectionnel. Le bus est le principe retenu pour PCI-2.2 (c'est le bus PCI traditionnel). Le bus de données comprend 32 ou 64 fils et la fréquence de transfert est égale à 33 ou 66 MHz. **C'est un bus parallèle, la bande passante est partagée entre les circuits.**

La tendance actuelle est de passer du bus parallèle partagé PCI traditionnel à la liaison série point à point. En effet, le seul moyen d'augmenter les performances du bus PCI est d'accroître le nombre de fils (passer à 128 bits de données) ou d'augmenter la fréquence de transfert, ce qui complique considérablement la fabrication des circuits imprimés dans les deux cas. Ces problèmes disparaissent avec une liaison série qui peut monter plus facilement en fréquence (jusqu'à 10 GHz avec une paire de conducteur en cuivre sur un circuit imprimé).

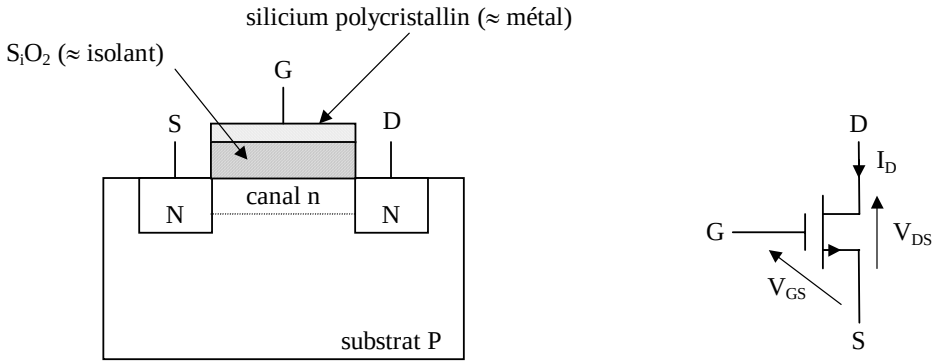
3.3.2 Technologies MOS et CMOS

L'avantage principal de la technologie CMOS sur la TTL est sa faible consommation statique et sa plus grande facilité d'intégration dans un circuit intégré. Son principal inconvénient à son lancement était sa très faible vitesse. De plus, sa fréquence de fonctionnement est

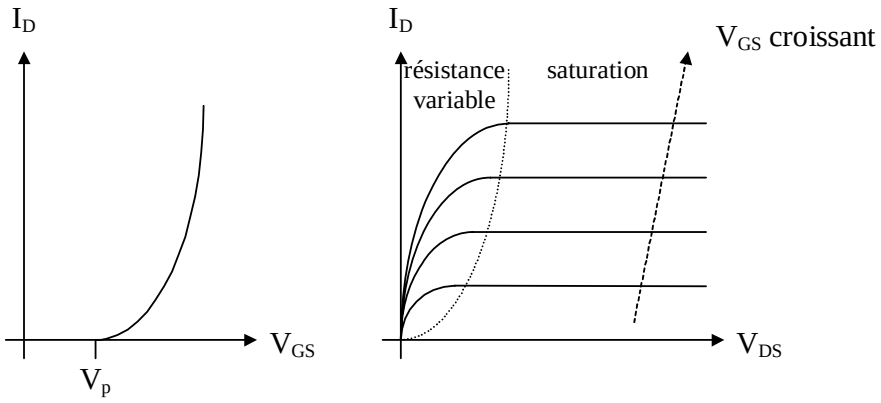
proportionnelle à sa tension d'alimentation ce qui la pénalisait fortement lors de la conception de cartes alimentées uniquement en 5 V. En effet, les premières familles pouvaient être alimentées entre 3 V et 18 V et elles étaient 3 à 4 fois plus rapides alimentées en 15 V qu'en 5 V. Depuis son introduction, cette technologie a subit de nombreuses améliorations et elle est aujourd'hui plus rapide que la TTL tout en gardant ses avantages initiaux. C'est pourquoi **la quasi-totalité des circuits intégrés sont développés aujourd'hui en logique CMOS** (à l'exception de certains convertisseurs analogique/numérique et de circuits très rapides). De la série classique MG (Metal-Gate CMOS), on est passé à la HC (High-speed CMOS) puis à la FACT (Advanced CMOS). La plus récente évolution consiste en la diminution de la tension d'alimentation. C'est par exemple la série LVC (Low-Voltage CMOS) qui est prévue pour fonctionner avec une alimentation 3,3 V avec des performances supérieures aux séries précédentes. Rappelons tout d'abord les caractéristiques d'un transistor MOS.

3.3.2.1 La logique MOS

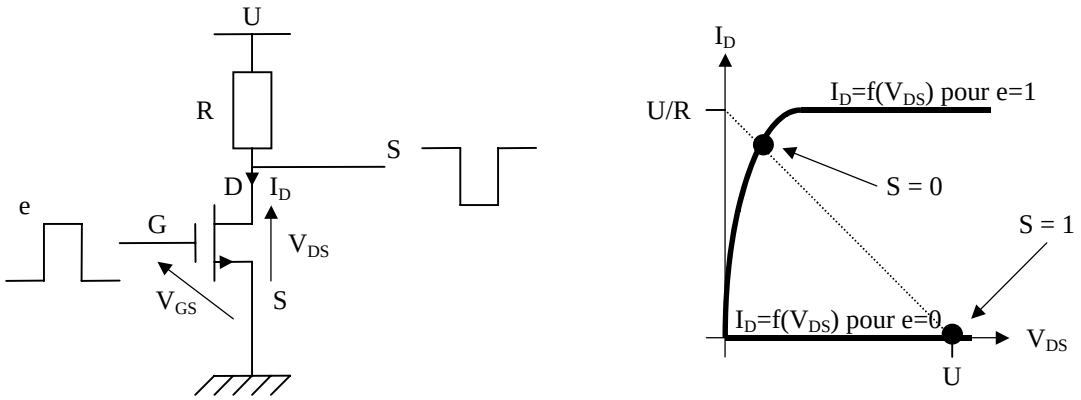
La structure d'un transistor MOS est rappelée ici. Nous ne travaillerons dans ce chapitre qu'avec des MOS à enrichissement (canal induit par V_{GS}).



Ses caractéristiques électriques sont :



La zone de saturation pour un transistor MOS équivaut à la zone linéaire (active) d'un transistor bipolaire, où I_D est un générateur commandé en tension. V_p est la tension de pincement. Quand $V_{GS} > V_P$, le courant de drain apparaît dans le MOS. Le schéma équivalent est donc un générateur de courant commandé par Vgs. La zone de résistance variable se situe au pied de la caractéristique, pour V_{DS} petit. Le schéma équivalent est une résistance variable Ron. On utilise cette zone pour réaliser des interrupteurs MOS analogiques. Les transistors MOS à canal N sont les plus utilisés car ils commutent plus vite que les canal P. La première application la plus simple et la plus évidente fut l'inverseur :

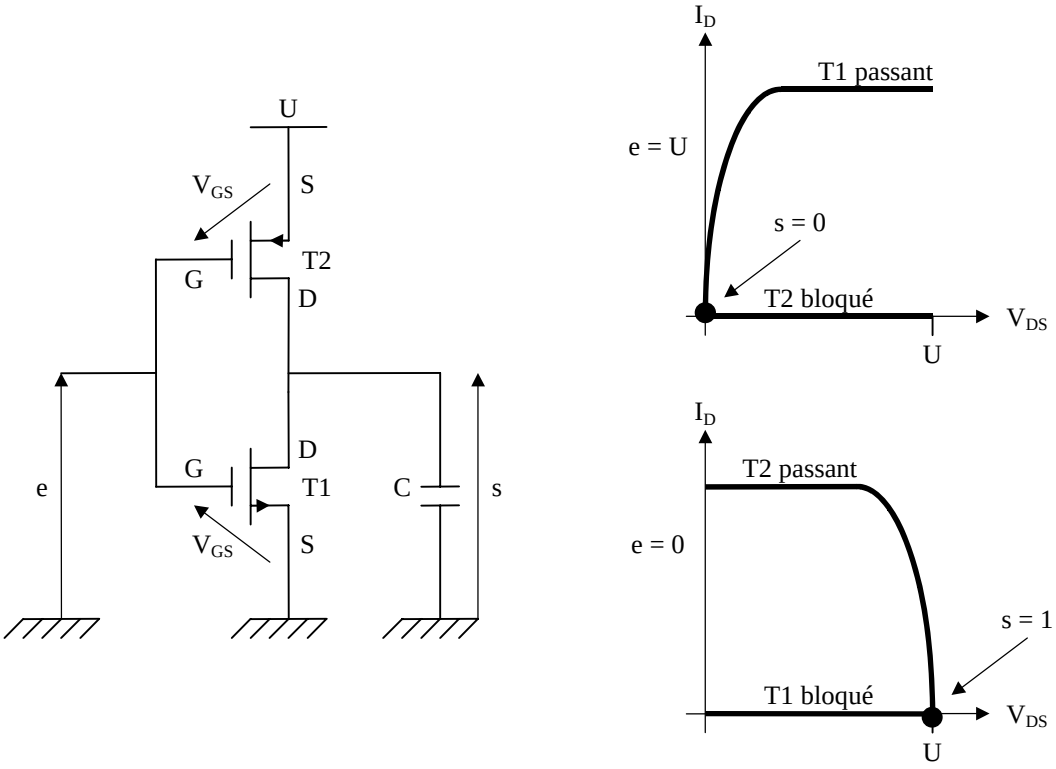


Les avantages de cette technologie sont le courant nul en entrée et la grande excursion de la tension de sortie (V_{OH} = U, V_{OL} est faible). Les principaux inconvénients sont la présence encombrante d'une résistance difficilement intégrable dans un circuit intégré et la consommation statique au niveau bas. Le remplacement de la résistance par un autre transistor MOS a été la principale évolution de la technologie NMOS (technologie du 80286 d'INTEL). Toutefois, la technologie CMOS l'a progressivement supplanté et elle n'est plus utilisée aujourd'hui.

3.3.2.2 La logique CMOS

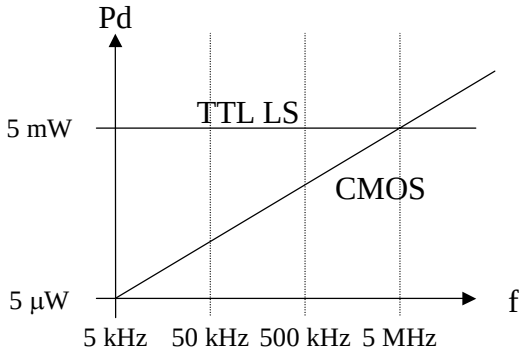
Les deux transistors MOS complémentaires utilisés sont à enrichissement. Les courants consommés aux niveaux haut et bas sont nuls. La consommation statique et basse fréquence est très faible (< au micro ampère). L'excursion de sortie est maximale (V_{OH} = U, V_{OL} = 0). Les deux transistors ayant le même I_{DSS}, les temps de commutation de la porte sont identiques pour les deux niveaux. On peut considérer que l'entrée d'une porte CMOS est une capacité pure de l'ordre de quelques pF. La sortie d'une porte ne voit donc qu'une capacité C représentant les capacités d'entrées des portes connectées plus la capacité de la ligne. Ce phénomène a une importance capitale et représente un des intérêts majeurs de cette

technologie. La vitesse de fonctionnement de la porte CMOS est quasiment déterminée par la valeur de cette capacité. Or, l'augmentation de la densité d'intégration a pour conséquence automatique la diminution de cette capacité de charge. Ainsi donc, de manière quasi mécanique, les performances de la logique CMOS s'améliorent au fur et à mesure de la progression de la technologie de réalisation des circuits intégrés. Aujourd'hui, les performances de la CMOS ont dépassé celles de la TTL qui a beaucoup moins évolué. De plus, l'intégration des transistors bipolaires est bien moins facile que l'intégration des transistors CMOS, la surface de silicium utilisée pour implanter un transistor CMOS étant environ 50 fois plus petite que la surface utilisée pour implanter un transistor bipolaire.

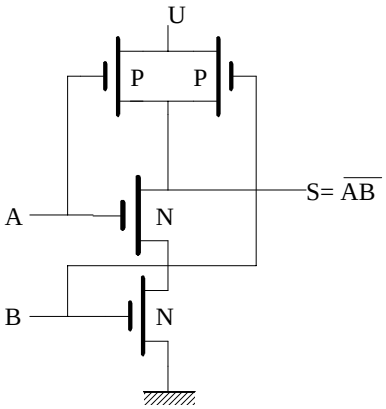


Il faut noter d'autre part que la vitesse de commutation est proportionnelle à la valeur de la tension d'alimentation. En effet, le courant disponible maximum vaut : $I_{Dmax} = I_{DSS}\left(\frac{U}{VP} - 1\right)^2$ et augmente donc proportionnellement à U^2, tandis que dans la formule approchée des temps de commutation, on a $t_T \approx \frac{C.U}{Imoyen}$. Comme Imoyen = $k.I_{Dmax}$, t_T diminue linéairement quand U augmente. **Les performances d'un circuit CMOS sont donc proportionnelles à U.**

Nous savons que la puissance dissipée Pd = Po(statique) + k.F. En statique, la consommation Po est nulle (ou presque), donc Pd est proportionnelle à F. Quand la fréquence augmente, la consommation d'un circuit CMOS s'élève beaucoup plus rapidement que pour un circuit TTL. Ainsi, un circuit CMOS proche de sa fréquence limite consomme autant (voir plus) qu'un circuit TTL-LS. Ce n'est qu'en basses fréquences que les consommations sont bien inférieures.



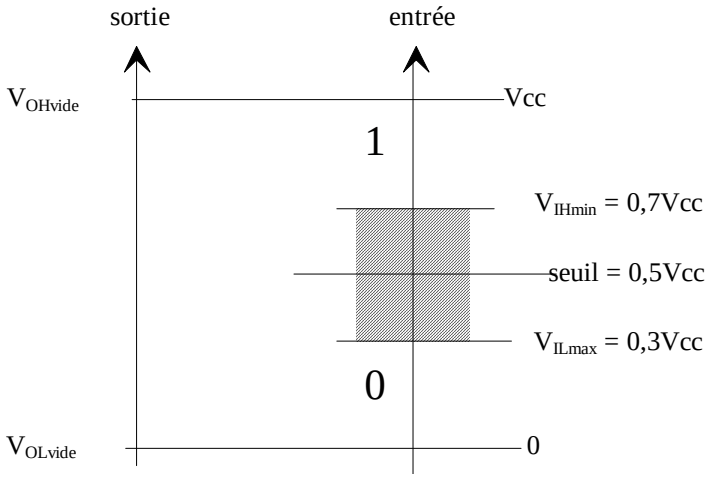
Voici un exemple de fonction logique NAND à base de transistors CMOS.



Tous les MOS réalisant les fonctions logiques internes ont de faibles I_{DSS}, la sortance est ensuite renforcée par des MOS à I_{DSS} plus élevé pour assurer des commutations rapides sur charges capacitives.

3.3.2.3 Caractéristiques électriques

Tant que les circuits CMOS restent chargés par d'autres circuits CMOS, les niveaux hauts et bas restent à V_{CC} et 0 (car les courants d'entrées sont nuls). Les immunités aux bruits sont élevées et proportionnelles à VCC : $\Delta H = \Delta L = 0{,}3.Vcc$. Il est interdit de laisser des entrées en l'air en CMOS, sinon le niveau peut prendre n'importe quelle valeur et peut même provoquer une oscillation en sortie.



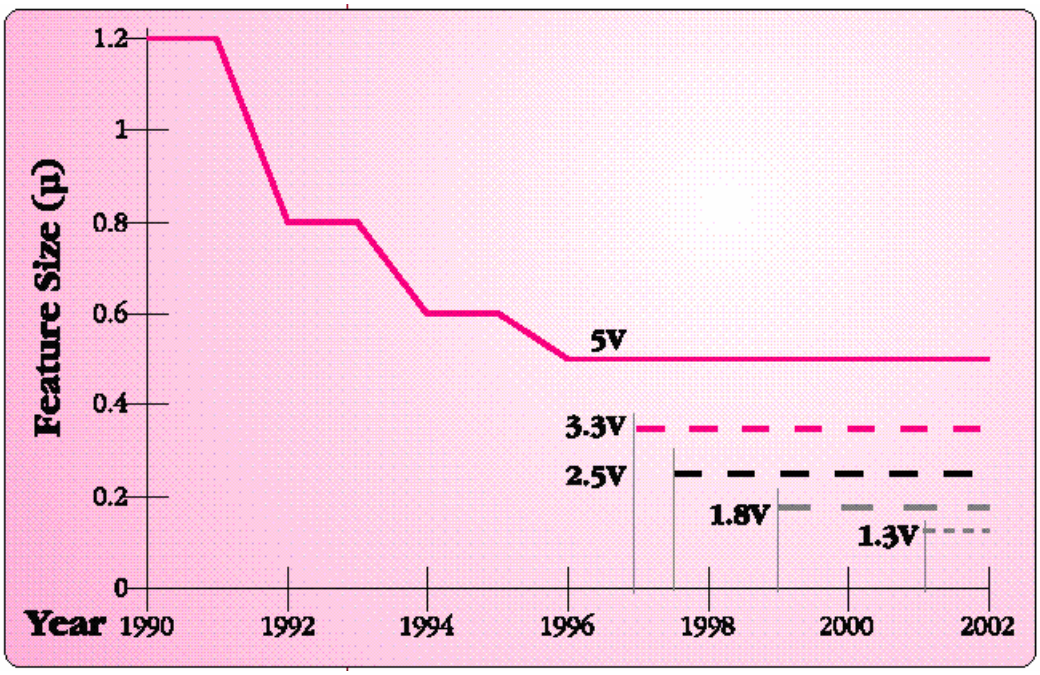
La sortance en CMOS est déterminée par le temps de propagation maximum du circuit et non par le rapport entre le courant de sortie et le courant d'entrée comme en TTL. Les constructeurs préconisent une sortance de 10 pour garantir les temps de propagation maximaux fournis. En effet, le temps de propagation du circuit est égal à la somme du temps de propagation interne plus le temps de transition en sortie. Or, c'est la valeur de la capacité de charge qui détermine ce temps de transition. Par exemple, pour une capacité d'entrée de porte égale à 5 pF, un circuit fournissant un courant moyen de 20 mA effectue des transitions de 12 V en 30 ns avec une charge de 10 portes. Le temps de propagation du circuit est donc fixé en pratique par la capacité de charge. On peut trouver dans les caractéristiques constructeurs les formules du temps de propagation en fonction de la capacité de charge. En basse fréquence, on peut souvent aller au-delà de la sortance limitée à 10, en sachant que les fronts vont se détériorer, ce qui n'est pas forcément gênant pour des circuits combinatoires.

Le mélange de circuits TTL et CMOS sur une carte doit être effectué avec précaution. Il y a deux cas (en 5V) :

- CMOS $\rightarrow$ TTL. Le constructeur fournit des valeurs maximales conseillées pour I_{OH} et I_{OL}. On en tient compte pour calculer la sortance sur des circuits TTL. La limitation pour les charges TTL vient du niveau bas. Si on dépasse I_{OLmax}, la valeur V_{OL} risque d'être trop près du seuil TTL ($\approx$ 1,4V).
- TTL $\rightarrow$ CMOS. Le V_{OHvide} est trop faible par rapport à la tension de seuil CMOS. La marge de bruit est alors diminuée surtout s'il y a des portes TTL parmi les portes CMOS. Il faut ajouter une résistance de pullup de 10 kΩ sur la sortie TTL pour remonter le niveau haut.

3.3.2.4 Tension d'alimentation et puissance

Trois facteurs sont essentiels en technologie CMOS : la longueur du canal du transistor MOS, la tension d'alimentation du circuit et sa puissance dissipée maximale. La longueur du canal est passée de 10 µm dans les années 70 à 0.5 µm vers 1995. Jusqu'à cette date, la tension d'alimentation est restée fixée à 5 V en règle générale (sauf pour des séries spéciales basse consommation). En dessous de 0.5 µm, le transistor MOS est incapable de supporter 5 V entre drain et source : il est donc obligatoire de baisser la tension d'alimentation (même sans tenir compte des problèmes de puissances dissipées). La figure suivante indique l'évolution de la tension d'alimentation en fonction de la longueur du canal :

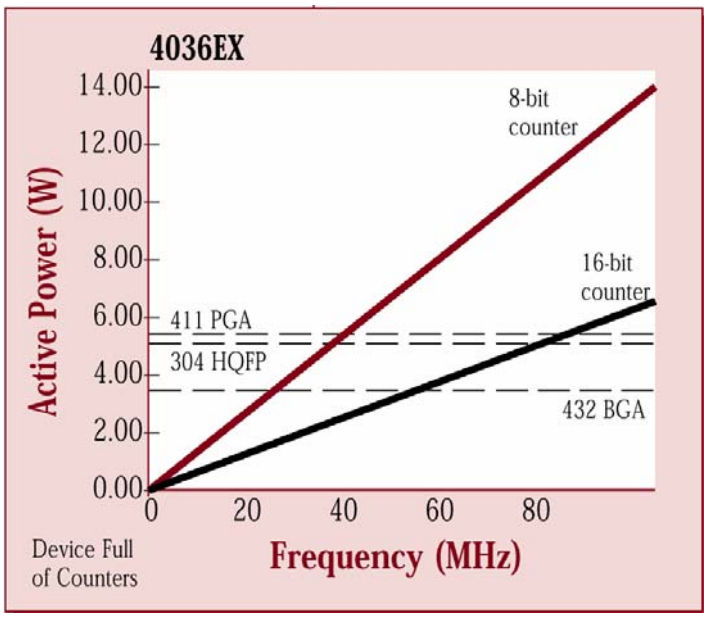


Cette chute de la tension d'alimentation a pour conséquence une réduction notable de la consommation du circuit (qui est proportionnelle à Vcc^2), ce qui est plutôt une bonne chose comme nous allons le voir. Elle a aussi pour conséquence une baisse des performances car la fréquence de fonctionnement d'un circuit CMOS est proportionnelle à Vcc. Malgré cela, la diminution de la longueur de canal augmente les performances du circuit (mais moins que si la tension d'alimentation restait la même). Le tableau suivant indique (en fréquence normalisée) la vitesse de fonctionnement dans 4 configurations :

	0.5 µm	0.35 µm
5 V	1	2 (en théorie)
3.3 V	0.66	1.3

On voit qu'en passant de 0.5 µm (5 V) à 0.35 µm (3.3 V), on gagne 30 % en vitesse et on diminue de 60 % la consommation.

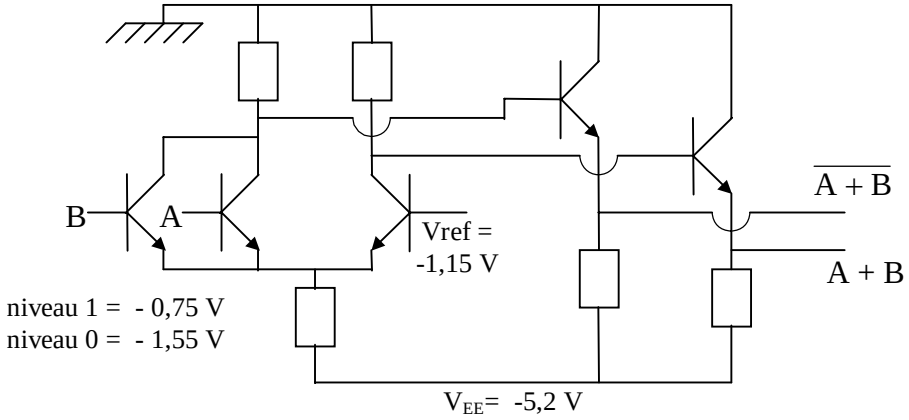
Il y a un deuxième problème important à prendre en compte en dessous (environ) de 0.5 µm. La puissance maximale dissipée par le circuit intégré peut être supérieure à celle dissipée par son boîtier. On peut donc réussir à détruire un circuit en fonctionnement normal pour la simple raison qu'il fonctionne trop rapidement. La figure suivante en est une illustration. On a pris un circuit logique programmable (XC4036EX de chez Xilinx) que l'on a entièrement rempli soit de compteur 8 bits, soit de compteurs 16 bits. Les traits en pointillés représentent la puissance maximale dissipée pour trois types de boîtiers.



On voit nettement qu'au-delà d'une certaine fréquence de fonctionnement (25 à 40 MHz en 8 bits), il est nécessaire d'utiliser un radiateur et/ou une ventilation pour augmenter la puissance dissipée par le boîtier sous peine de destruction du circuit intégré. Ce phénomène est bien connu dans le domaine des microprocesseurs avec l'apparition d'un ventilateur intégré dans le radiateur des Pentium d'Intel par exemple.

3.3.3 Technologie ECL

La logique ECL doit son nom à sa structure à base d'amplificateur différentiel. Les transistors bipolaires qui la composent restent constamment en régime linéaire (ni saturés, ni bloqués). C'est une logique à transistors non-saturés. C'est aujourd'hui encore la technologie la plus rapide au prix d'une consommation très élevée. Elle nécessite par ailleurs une alimentation négative. Cette technologie a elle aussi bien évolué depuis la série 100K des débuts (1975), la série 10KH (1981) jusqu'à la série récente série E-lite. Il faut toutefois signaler que la mise au point d'une carte en technologie ECL est beaucoup plus délicate qu'en TTL ou en CMOS.



Cette technologie est encore utilisée aujourd'hui pour les circuits logiques très rapides et notamment à base d'Arséniure de Gallium dans le domaine militaire.

3.3.4 Comparaison des technologies

Le tableau comparatif suivant essaye de synthétiser les caractéristiques typiques de ces différentes familles de circuits à partir des feuilles de caractéristiques constructeurs (data sheet). Les comparaisons sont difficiles à effectuer car ces caractéristiques ne sont pas mesurées dans les mêmes conditions. Il faut utiliser ce tableau pour comparer les caractéristiques générales de chaque famille et se reporter aux data sheet correspondantes pour étudier plus en détail les spécifications min/max qui sont plus proches de l'utilisation réelle du composant. Les paramètres utilisés dans ce tableau sont les suivants :

- **Vitesse**

Temps de propagation	t_{PLH} dans une porte OU de type 74LS32.
Fréquence maximale	Toggle Rate d'une bascule D de type 74LS74.
Temps de transition	t_{TLH} à la sortie d'une des deux portes précédentes.

- **Consommation :** la consommation est donnée au repos ou à une fréquence de fonctionnement de 1 MHz.

- **Tension d'alimentation :** ce sont les valeurs min et max de la tension d'alimentation.

- **Courant de sortie** : il s'agit du courant de sortie max à l'état bas.

- **Marge de bruit :** la marge de bruit est traditionnellement exprimée en pourcentage de l'excursion typique de la tension de sortie.

- **Boîtiers** : ce tableau concerne des circuits intégrés ayant un faible nombre de broches (catalogue MSI/LSI).

- **Fonctions dans la famille** : il s'agit du nombre de fonctions différentes commercialisées dans la famille.

- **Coût** : le prix relatif par porte en prenant comme référence la technologie ALS.

La signification des différents sigles utilisés est la suivante :

LS	Motorola Low power Schottky TTL
ALS	Texas Instrument Advanced Low power Schottky TTL
ABT	Philips Advanced BiCMOS Technology
FAST	Motorola Advanced Schottky TTL
MG	Motorola 14000 Series Metal Gate CMOS
HC	Motorola High-Speed Silicon Gate CMOS
FACT	Motorola Advanced CMOS
LVC	Philips Low-Voltage CMOS
10KH	Motorola 10KH Series ECL
E-lite	Motorola (ECLinPS Lite) Advanced ECL

Tableau comparatif des familles logiques

Paramètres commerciaux typiques à 25 °C	**familles logiques**									
	TTL/ABT				**CMOS**				**ECL**	
	LS	**ALS**	**ABT**	**FAST**	**MG** (Vcc=5V)	**HC**	**FACT**	**LVC**[1] (Vcc=3,3V)	**10KH**	**E-lite**
vitesse										
temps de propagation [ns]	14	9	2,3	4	160	10	6	3,3	1	0,25
fréquence maximal [MHz]	33	45	250	125	4	40	160	200	330	2800
temps de transition [ns]	6	4	2,5	4	100	7	3	3,5	1	0,25
consommation par porte										
statique [mW]	5	2,5	0,005	12,5	0,0006	0,001	0,003	0,0001	25	73
à 1 MHz [mW]	5	2,5	1	12,5	0,4	0,6	0,8	0,6	25	73
tension d'alimentation [V]	4,75 à 5,25	4,5 à 5,5	4,5 à 5,5	4,5 à 5,5	3 à 18	2 à 6	2 à 6	1,2 à 3,6	-4,5 à -5,5	-4,5 à -5,5
courant de sortie [mA]	8	8	20	20	1	4	24	24	charge 50 Ω	charge 50 Ω
marge de bruit										
état haut [%]	22	22	22	22	30	30	30	30	30	33
état bas [%]	10	10	10	10	30	30	30	30	30	33
boîtiers										
DIP	oui	oui	oui	oui	oui	oui	oui	non	oui	non
SO	oui	oui	oui	oui	oui	oui	oui	oui	non	oui
LCC	non	oui	non	oui	non	non	oui	non	oui	non
fonctions dans la famille	190	210	50	110	125	103	80	35	64	40
prix relatif par porte	0,9	1	1,6	1	0,9	0,9	1,4	1,8	2	32

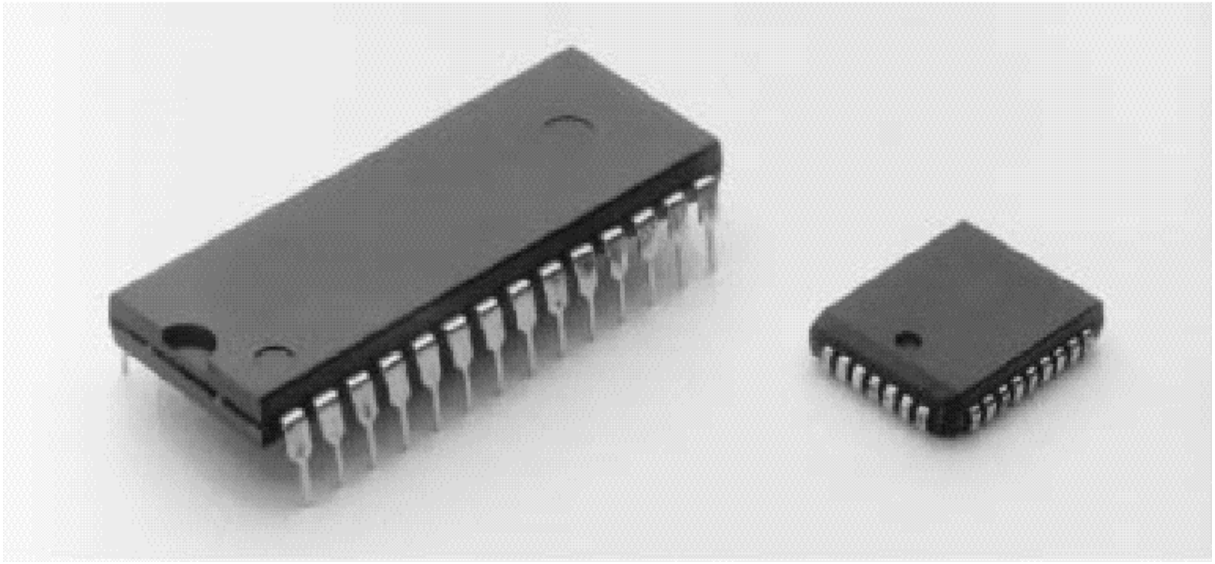
1. accepte les niveaux TTL en entrée.

3.4 Réalisation des circuits imprimés

Le choix des boîtiers de circuits intégrés est intimement lié à la réalisation du circuit imprimé. Il conditionne le coût de la carte et donc la rentabilité économique du produit. La réalisation de circuits imprimés (prototype ou grande série) est un métier à part entière qui englobe des disciplines telles que l'électronique et la chimie mais aussi la mécanique et la logistique. Il n'est pas question de faire ici un cours sur ce sujet, mais de donner à l'étudiant des notions de bases. La technologie traditionnelle des composants traversant le circuit imprimé (composants à piquer) était, au début des années 1980, la seule solution utilisée hors du Japon. A cette époque, elle atteint ses limites avec des boîtiers ayant 68 broches. La technologie des composants montés en surface « CMS » (ou SMT en anglais) s'imposa alors grâce à ses nombreux avantages :

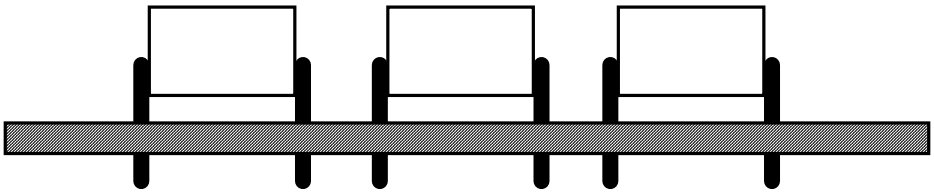
- densité d'intégration sur la carte plus élevée,
- possibilité de souder des composant des deux côté de la carte,
- poids réduit,
- meilleure possibilité d'automatisation des chaînes de production, donc baisse des coûts (environ 30 % avec un coût salarial occidental),
- réduction de la longueur des broches du boîtier et donc des éléments parasites,
- meilleure fiabilité.

La photo suivante illustre bien la différence entre deux boîtiers 28 broches :



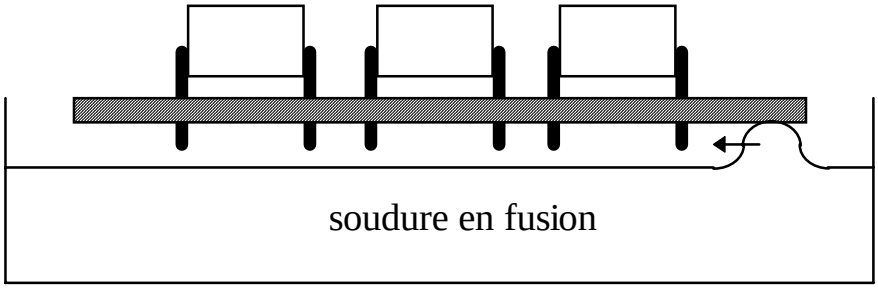
3.4.1 Circuits imprimés traditionnels

Le circuit imprimé est composé d'un mélange de résine époxy et de feuilles de cuivre. Les broches des composants à piquer traversent la carte grâce à des trous métallisés. Il peut y avoir des pistes en cuivre sur chaque face du circuit imprimé (on parle alors de circuit double face à trous métallisés) mais aussi plusieurs couches de pistes prises en sandwich à l'intérieur du circuit imprimé (on parle alors de circuits multicouches).



Une fois le circuit imprimé réalisé, les étapes suivantes sont nécessaires afin de réaliser la carte électronique :

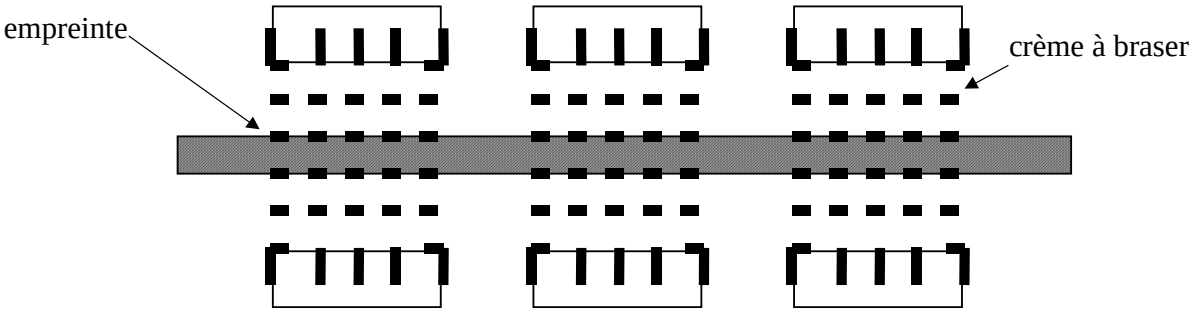
- l'insertion des composants : c'est une étape difficilement automatisable, aussi est-elle souvent réalisée à la main.
- la soudure à la vague : une vague de soudure en fusion vient lécher le coté du circuit sans composant. La soudure se dépose dans les trous métallisés et remonte vers l'autre face du circuit par capillarité. La vague peut se déplacer dans un seul sens (soudure simple vague) ou effectuer un aller retour (soudure double vague).



- le nettoyage : le circuit est ensuite nettoyé avec un solvant spécial que l'on appelle du flux.
- le test : les liaisons entre les composants et les fonctions électroniques sont ensuite testées pour vérifier le bon fonctionnement de la carte.

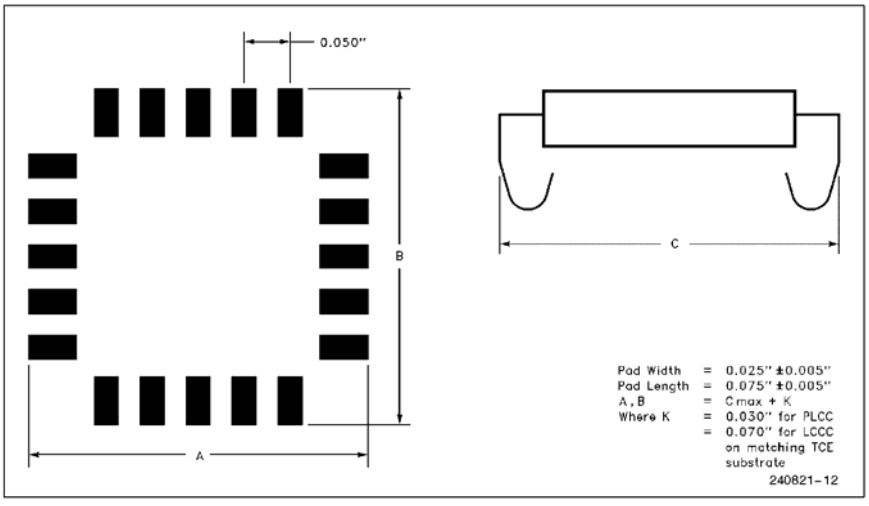
3.4.2 Circuits imprimés avec montage en surface

Le circuit imprimé a la même composition qu'au chapitre précédent, mais il n'y a pas de trous métallisés.



Après réalisation du circuit imprimé, les étapes suivantes sont nécessaires à la fabrication de la carte électronique :

- l'application de la crème à braser : c'est un mélange de soudure, de flux et d'adhésif. Cette crème doit être appliquée en petite quantité sur chacun des éléments de l'empreinte du CMS qui doit donc être connue de la machine qui réalise cette opération. L'absence de normalisation des boîtiers CMS complique singulièrement cette phase, chaque fabricant de composants ayant ses propres empreintes ainsi que ses propres tolérances.



  Le choix de la crème à braser et de la proportion des éléments qui la compose est aussi délicate. En effet, elle sert non seulement à souder le composant, mais aussi à le maintenir collé sur la carte pendant le processus de fabrication.

- le placement des composants : les petits boîtiers CMS sont livrés en bandes et les gros boîtiers CMS sont livrés en plateaux, ce qui facilite l'automatisation de cette étape.

- la refusion : la carte est chauffée afin de réaliser la fusion de la crème qui se trouve entre le composant et le circuit imprimé. Cette opération est effectuée dans une étuve grâce à un système de soudure en phase vapeur ou bien un système à infrarouge ou encore un système « hot bar ».
- si des composants sont soudés des deux cotés de la carte, il faut ajouter les étapes suivantes :
    - le retournement de la carte.
    - le placement des composants.
    - la refusion (on parle alors de double refusion).
- le nettoyage de la carte avec du flux.
- le test.

Il est parfois nécessaire de réaliser des cartes mélangeant les CMS et les composants à piquer car tous les boîtiers n'existent pas en CMS. Il faut alors effectuer les deux phases de fabrication à la suite, le soudage des CMS venant en premier. Il est déconseillé que la carte ait des composants actifs sur ses deux faces. La face qui sera au contact avec la vague de soudure peut toutefois comporter des composants CMS passifs. On évite généralement ce mélange de technologie car le surcoût dû aux deux phases de fabrication est trop élevé. Il est à noter que sous certaines conditions, il est possible de souder des boîtiers traversant avec de la crème à braser par refusion (opération dite « pin in past ») afin d'éliminer ce surcoût.

3.5 Le choix des boîtiers

3.5.1 Caractérisation d'un boîtier

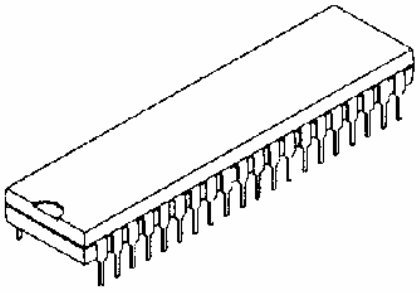
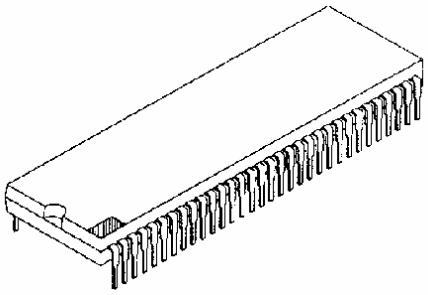
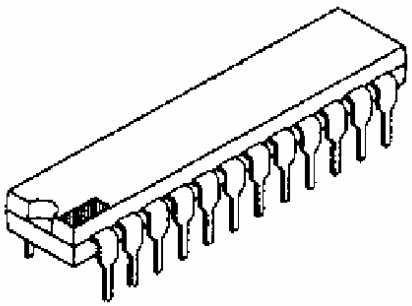
Sous ses apparences simples, le boîtier encapsulant la puce du circuit intégré possède des caractéristiques capitales pour le bon fonctionnement de la carte, mais aussi pour la rentabilité du produit :

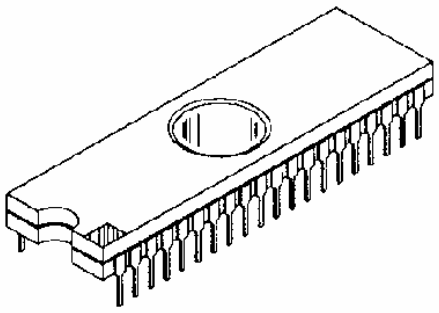
- le nombre de broches rapporté à la surface occupée.
- la hauteur.
- les capacités à évacuer la puissance dissipée par la puce. On parle de dissipation thermique. Les boîtiers céramiques (série militaire) sont progressivement abandonnés au profit des boîtiers plastiques (série industrielle ou commerciale).
- aptitude à l'automatisation de la réalisation du circuit imprimé.
- caractéristiques électriques.

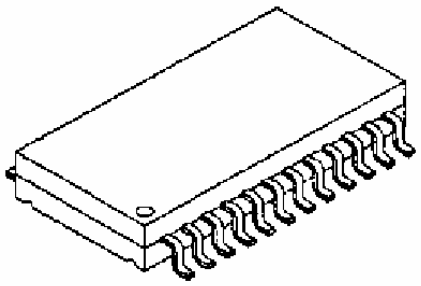
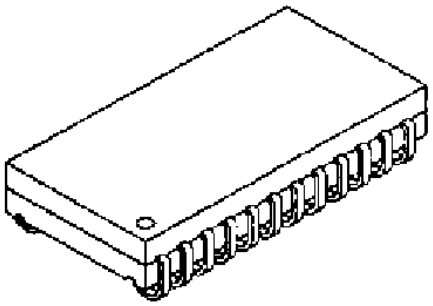
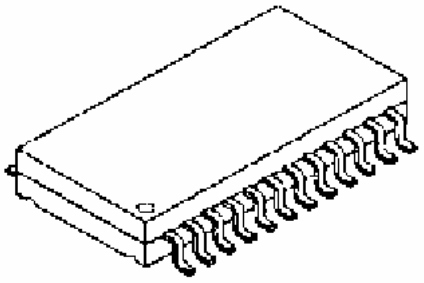
Les dimensions essentielles d'un boîtier sont la hauteur, la largeur, la longueur et le pas entre les broches (le pitch en anglais). Ces dimensions sont exprimées soit en millièmes de pouce (un mil = 0,0254 mm) soit en millimètres, ce qui ne simplifie pas les comparaisons ainsi que les conversions.

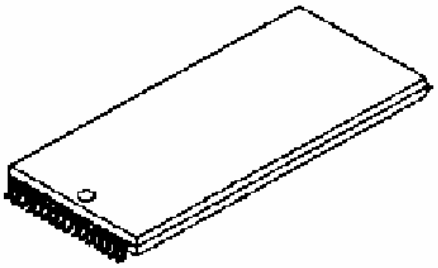
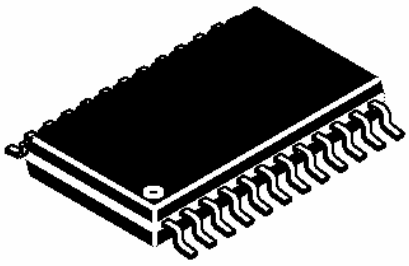
3.5.2 Les différents types de boîtiers

3.5.2.1 Les boîtiers à deux rangées de broches disposées aux extrémités

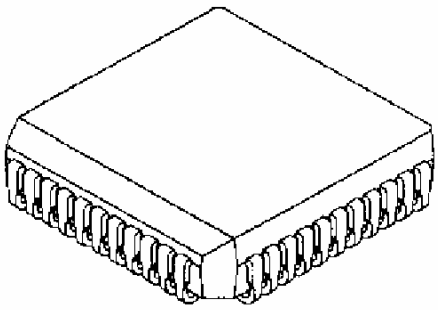
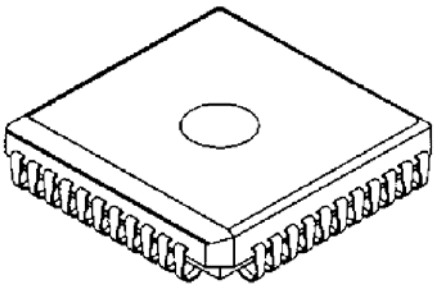
famille DIP (Dual In-line Package) de 8 à 64 broches			
boîtier	aspect	dimensions	application
DIP		pitch = 100 mil hauteur = 130 à 170 mil	circuits standards
SHRINK DIP		pitch = 70 mil hauteur = 130 à 170 mil	circuits standards
SKINNY DIP		pitch = 100 mil hauteur = 130 à 170 mil largeur divisée par 2	circuits standards

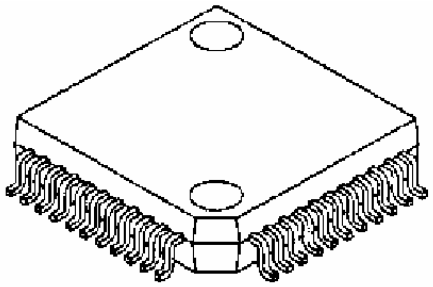
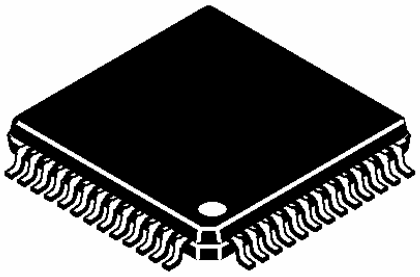
windowed CERDIP boîtier céramique avec fenêtre en mica		pitch = 100 mil hauteur = 100 à 170 mil	mémoire PROM effaçable aux UV

famille SOP (Small Outline Package) de 20 à 64 broches			
boîtier	aspect	dimensions	application
SOIC		pitch = 50 mil hauteur = 100 à 120 mil	circuits standards
SOJ		pitch = 50 mil hauteur = 110 mil	circuits standards
SSOP		pitch = 0,65 mm (26 mil) hauteur = 1,9 mm (75 mil)	circuits standards

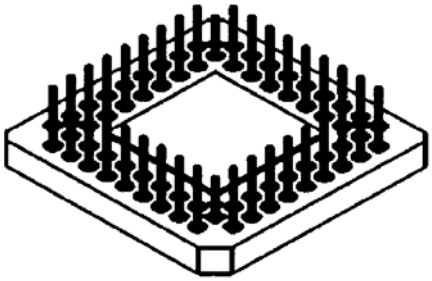
TSOP		pitch = 0,5 mm (20 mil) hauteur = 1,2 mm (47 mil)	mémoires
TSSOP		pitch = 0,65 mm (26 mil) hauteur = 1,2 mm (47 mil)	circuits standards

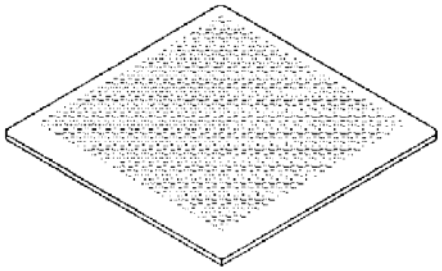
3.5.2.2 Les boîtiers à quatre rangées de broches disposées aux extrémités

famille PLCC (Plastic Leaded Chip Carrier) de 20 à 84 broches			
boîtier	aspect	dimensions	application
PLCC		pitch = 50 mil hauteur = 68 à 150 mil	circuits standards
windowed CERQUAD, boîtier LCC céramique avec fenêtre en mica		pitch = 50 mil hauteur = 150 à 190 mil	Circuits logiques programma-bles effaçable aux UV

famille QFP (Quad FlatPack) de 44 à 304 (ou plus) broches			
boîtier	aspect	dimensions	application
QFP		pitch = de 0,8 à 0,5 mm (20 mil) hauteur = 130 à 170 mil	circuits spécifiques, circuits périphériques micro-processeurs
TQFP		pitch = 26 mil hauteur = 67 mil	ASIC, périphériques micro-processeurs

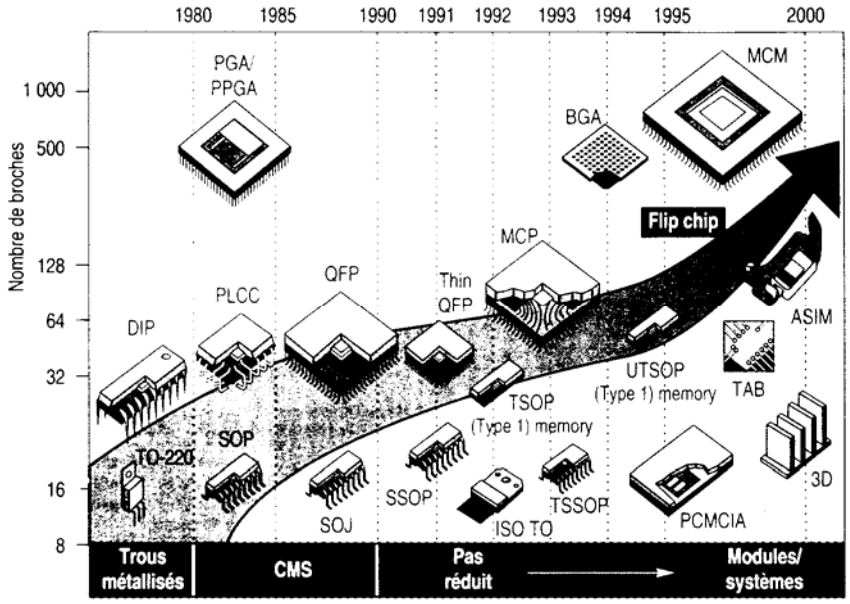
3.5.2.3 Les boîtiers ayant leurs broches disposées en dessous

famille PGA (Pin Grid Array) de 68 à 499 (ou plus) broches			
boîtier	aspect	dimensions	application
PGA	 (vue de dessous)	pitch = 100 mil hauteur = 120 mil	micro-processeurs, circuits spécifiques

famille BGA (Ball Grid Array) de 196 à 432 (ou plus) broches			
boîtier	aspect	dimensions	application
BGA	 (vue de dessous)  (vue de coté)	pitch = 50 mil hauteur = 50 à 80 mil	micro-processeurs, circuits spécifiques

3.5.3 L'évolution des boîtiers

L'évolution des boîtiers de circuits intégrés (package en anglais) est marquée par deux grandes tendances : l'augmentation du nombre de broches, en particulier pour les microprocesseurs et les circuits spécifiques (Asic en anglais) complexes, et la réduction des dimensions de tous les boîtiers. Ainsi, le pas entre broches ou pitch est passé des 100 mil typiques à 50 mil, voire à 20 mil pour les produits de pointe. La variété des technologies en présence s'étend chaque jour comme le montre la figure suivante (source National Semiconductor) :



Les boîtiers DIP (Dual-in-line package) traditionnels qui ont dominé les années 80 ne représentaient plus que 65 % du marché en 1990.

Les nombreuses variantes du boîtier DIP de base comprennent le boîtier SIP (Single-in-line package), le module SIMM (single-in-line memory modules) pour composants mémoires, ou le SDIP (Schrink DIP). Elles autorisent une diminution de la surface occupée sur le circuit imprimé. Mais le DIP est limité en pratique à 64 broches, en raison de la disposition périphérique de ses broches. Pour les circuits à grand nombre de broches tels que les microprocesseurs et les Asic complexes, les boîtiers Fakir ou PGA (Pin grid array) se sont donc imposés au cours des années 80. Caractérisés par la disposition surfacique de leurs broches sur le fond du boîtier, ils sont toujours destinés à un montage en trous traversants et se trouvent aujourd'hui en concurrence avec les boîtiers denses à broches en «ailes de mouette» pour montage en surface.

Un nouveau saut technologique est en effet intervenu entre-temps avec la généralisation de ces boîtiers à montage en surface (CMS ou SMT selon que l'on affectionne les sigles français ou américains). Connue depuis plus de trente ans, cette technologie n'était jusqu'alors mise en œuvre que par des fabricants japonais. Le premier boîtier à montage en surface fut le boîtier SO ou SOP (Small outline package) qui, avec un pas de 50 mil, a divisé par deux la surface occupée sur la carte par rapport au DIP traditionnel. Ses évolutions sont le SSOP (Shrink SOP) à pas de 25,6 mil (0,65 mm), le QSOP (Quarter sized outline package), qui est un SSOP à pas de 25 mil (0,635 mm), le TSSOP (Thin shrink SOP) à pas de 25,6 mil ou 20 mil (0,50 mm), dédiés aux cartes faible hauteur telles que celles employées dans les produits standards PCMCIA. Tous ces boîtiers, à nombre modéré de broches comme le DIP d'origine, sont dédiés aux mémoires, aux fonctions logiques et aux fonctions d'interface de bus.

La seconde génération de boîtiers CMS, apparue dans les années 80 pour les besoins des circuits à nombre de broches élevés, comprend le PLCC (Plastic leaded chip carrier) ou boîtier porteur de puces, et le QFP (Quad flat pack), caractérisé par un pas moyen de 0,50 mm (20 mil) et par des versions «fine pitch» de moins de 0,50 mm. Parmi les nombreuses variantes du QFP, on trouve le SQFP (Small quad flat pack), adapté aux problèmes de consommation grâce à son dissipateur thermique intégré, le RQFP à radiateur intégré, le TQFP (Thin quad flat pack) de faible hauteur...

Pour leur part, les années 90 furent caractérisées par l'apparition du boîtier BGA (Ball grid array), dont les contacts sont disposés en surface comme les PGA mais constitués de billes de soudures. Adoptée par des fabricants de semi-conducteurs comme IBM, Motorola, Matsushita ou VLSI Technology entre autres, cette technologie existe en trois versions : en céramique (CBGA), en plastique (PBGA) et sur bande (TBGA). Son usage s'est généralisé malgré des problèmes de testabilité et de réparabilité qui se sont longtemps posés aux fabricants. Aujourd'hui, le BGA a son rôle à jouer car, selon VLSI, il offre jusqu'à deux fois plus de connexions par unité de surface de circuit imprimé que le QFP et son coût par broche est beaucoup plus faible que celui des PGA.

En conséquence, pour les circuits à grand nombre de broches, le concepteur se trouve en présence de trois grandes familles de boîtiers : les PGA en technologie traversante, les QFP à montage en surface et à brochage périphérique, et les BGA. Le choix du boîtier a des répercussions sur les performances des circuits, notamment en technologie haute vitesse : ainsi, avec les PGA et les BGA, les différences de longueurs de pistes entre les plots d'E/S de la puce et les contacts sur le circuit imprimé peuvent varier de manière significative d'une broche à l'autre, posant des problèmes de décalage de signaux (skew) à haute vitesse. Avec les QFP, ce problème est moins critique mais il faut accorder plus d'importance à l'impact de l'inductance des broches sur l'intégrité du signal.

Mais le choix du boîtier a aussi des répercussions sur la fabrication du circuit imprimé. En augmentant le nombre de broches et en diminuant les pas, on diminue inexorablement la largeur des pistes autorisant une densité d'interconnexion donnée (exprimée en nombre de pistes autorisées entre pastilles). Cette largeur de piste est un facteur déterminant pour le coût du circuit imprimé. Pour un circuit grand public ou toute application hautes performances sensible aux coûts, le cahier des charges imposera communément une densité minimale de 5 pistes entre pastilles, tout en recommandant d'éviter de descendre en dessous d'une largeur de piste de 0,15 mm. Cela devient impossible avec les boîtiers de plus de 400 broches, qu'ils soient à technologie traversante ou CMS. Le concepteur peut choisir, pour rester dans une largeur de piste raisonnable, d'augmenter le nombre de couches signal du circuit imprimé, mais ceci se répercute alors sur la complexité, le poids et le prix du circuit. Les boîtiers de

type BGA sont théoriquement à même d'offrir une solution viable à ce problème. Le tableau suivant donne quelques valeurs concernant ces trois types de boîtiers :

	QFP			PGA			BGA		
Nombre de broches	**Pas [mm]**	**Côté du boîtier [mm]**	**Surface [mm^2]**	**Pas [mm]**	**Côté du boîtier [mm]**	**Surface [mm^2]**	**Pas [mm]**	**Côté du boîtier [mm]**	**Surface [mm^2]**
100	0,50	17	272	2,54	29	864	1,52	19	361
225	0,50	32	1032	2,54	42	1772	1,52	27	702
400	0,50	54	2916	2,54	55	3003	1,52	34	1156
625	0,50	82	6745	2,54	68	4556	1,52	42	1722
900	0,50	117	13572	2,54	80	6432	1,52	49	2401

3.5.4 Précautions à prendre

Il ne faut pas oublier les précautions à prendre pour manipuler et souder les composants CMOS afin d'éviter leur destruction par décharge électrostatique (« ESD ») :

- Il ne faut pas toucher les broches des composants sans être relié à la terre par le biais d'un bracelet antistatique.

- Les composants doivent se trouver dans un sac ou dans un étui antistatique pendant les manipulations et le stockage.

- Il faut utiliser un établi antistatique relié à la terre (ou au moins un bracelet) pour intervenir sur une carte.

3.6 Exercices

exercice 3.1

Soit le circuit NAND SN74LS00 (voir les caractéristiques en annexe).

1. Indiquer sur le schéma de la porte le sens des 4 courants définis dans le cours.
2. Pourquoi V_{OHmin} est-il toujours supérieur à V_{IHmin} et V_{OLmax} est-il toujours inférieur à V_{Ilmax} ?
3. Donner la définition des marges de bruit à l'état haut et à l'état bas et calculer ces valeurs.
4. Combien d'entrées de porte SN74LS00 peut-on connecter à la sortie d'une porte de même type ?

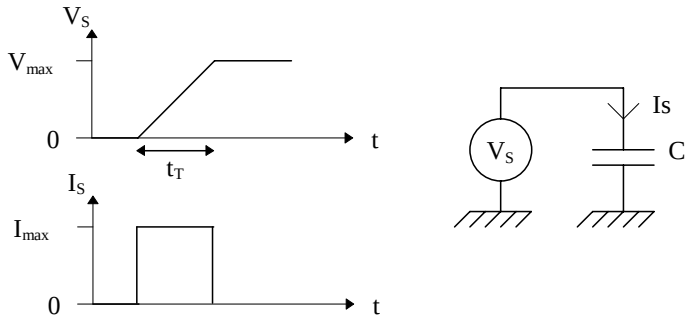
exercice 3.2

Reprenons la caractéristique de la page 133.

1. Retrouver les valeurs des trois variations $\frac{\Delta t_p}{\Delta T}$, $\frac{\Delta t_p}{\Delta V_{CC}}$, $\frac{\Delta t_p}{\Delta \text{fabrication}}$.
2. Vérifier l'écart entre délai minimum et délai maximum.

exercice 3.3

La charge vue par une sortie en logique CMOS est équivalente à un condensateur. Nous allons calculer le courant nécessaire pour charger ce condensateur à l'aide du modèle simplifié suivant :

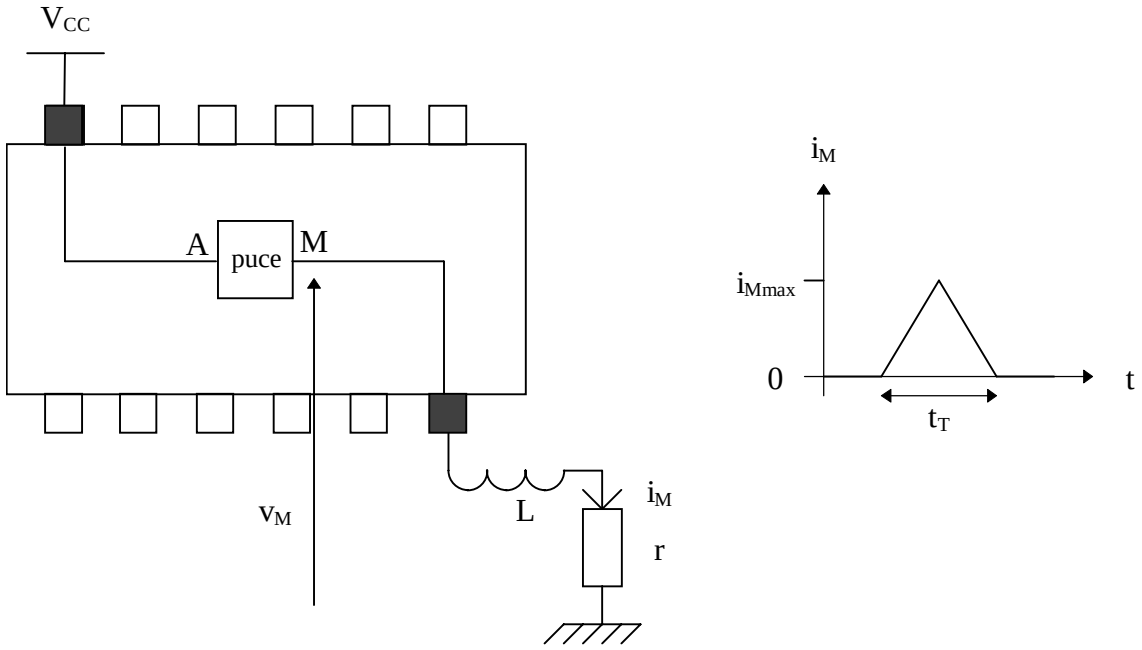


On a V_{max} = 4 V et C = 100 pF et on veut que t_T = 5 ns.

1. Calculer la valeur I_{max} du courant de sortie I_s.
2. Que se passe-t-il si le courant de sortie I_s est limité à 20 mA ?
3. Calculer la variation du courant d'alimentation lorsque 32 sorties du circuit changent d'état simultanément.

exercice 3.4

La figure suivante représente, de manière simplifiée la ligne d'alimentation d'un circuit intégré.

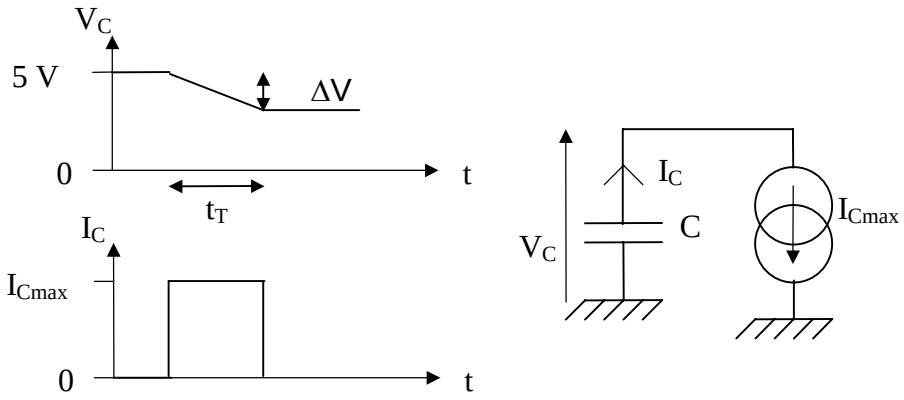


On suppose la variation maximale de courant d'alimentation égale à 2,5 A pendant une durée de 5 ns, avec r = 0,01 Ω et L = 2 nH.

1. Déterminer l'allure de V_M au moment de la commutation.
2. Quelle est l'influence de la variation de V_M sur une entrée du circuit ?
3. Quelle est l'influence de la variation de V_M sur une sortie du circuit ?
4. Pourquoi n'est-il pas nécessaire de s'occuper de la variation de V_A ?

exercice 3.5

Nous allons calculer la valeur du condensateur nécessaire pour découpler un circuit intégré. Ce condensateur, faiblement selfique, est placé au plus près du circuit afin de fournir le courant d'alimentation nécessaire pendant les commutations. On va pour ce calcul utiliser le modèle simplifié suivant :

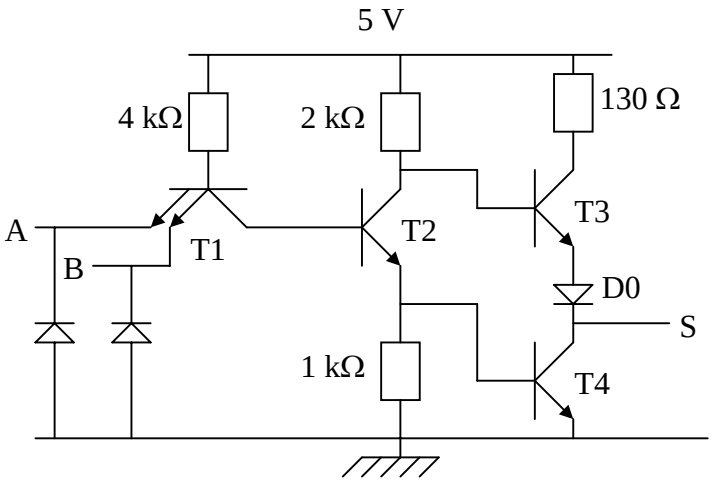


On suppose qu'au moment de la commutation, le condensateur de découplage C est déchargé par un courant constant de valeur 2,5 A pendant une durée de 5 ns.

1. On prend C = 10 nF. Calculer la valeur de la chute de tension ΔV aux bornes de C à la fin de la commutation.
2. La tension d'alimentation doit rester comprise entre 4,75 V et 5,25 V (série commerciale). Quelle valeur de condensateur de découplage faut-il alors choisir ?

exercice 3.6

Soit la porte TTL totem-pole suivante :

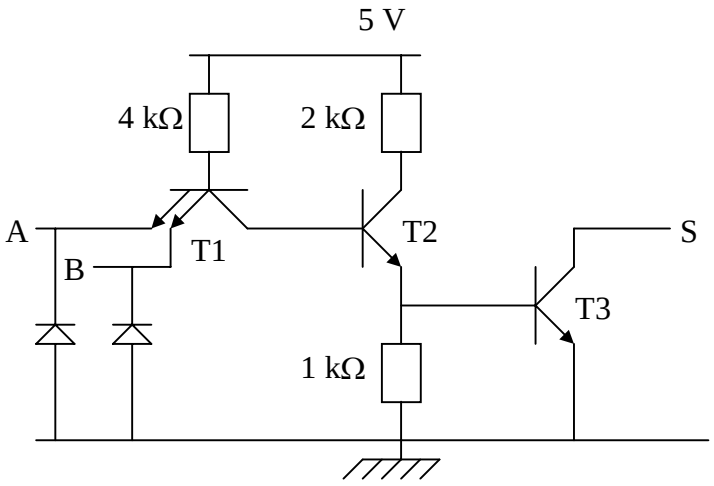


1. Expliquer, sans faire de calculs, le fonctionnement du montage.
2. Les entrées A et B sont à 0. Calculer les tensions et courants du montage et déterminer l'expression de V_S en fonction du courant de sortie du montage.
3. Les deux entrées A et B sont à 1. Calculer les tensions et courants du montage et déterminer l'expression de V_S en fonction du courant de sortie du montage.

4. Quel est le rôle des diodes à l'entrée du montage ?
5. Comparer les caractéristiques électriques obtenues à celles du circuit SN74LS00 données en annexe.

exercice 3.7

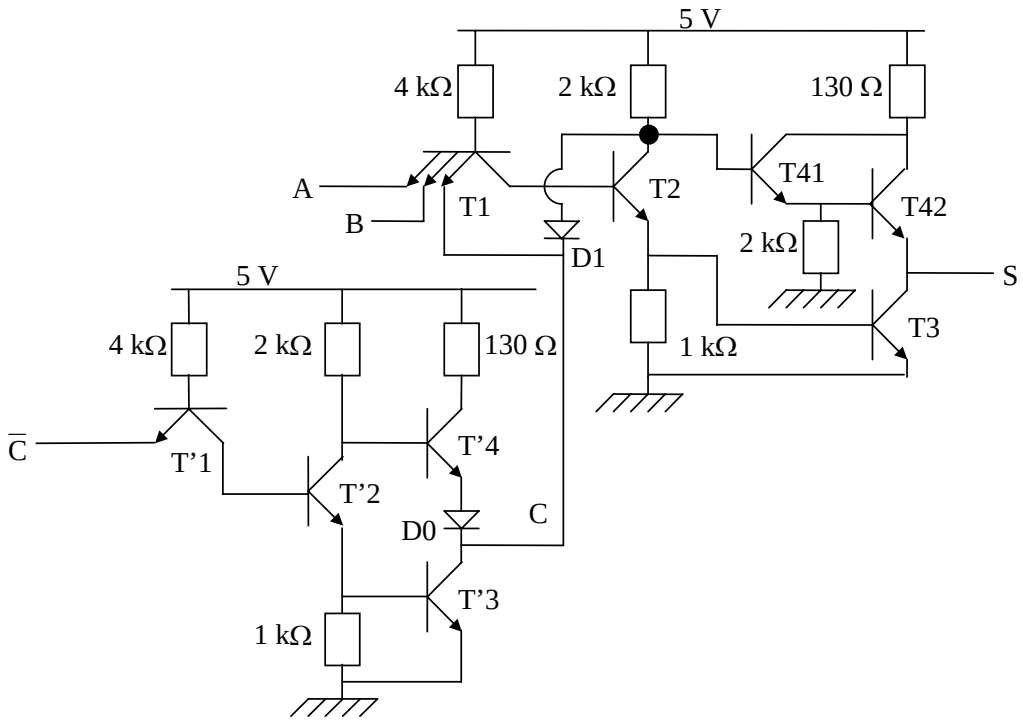
Soit la porte TTL collecteur ouvert suivante :



1. Expliquer, sans faire de calculs, le fonctionnement du montage.
2. Les entrées A et B sont à 0. Calculer les tensions et courants du montage.
3. Les deux entrées A et B sont à 1. Calculer les tensions et courants du montage et déterminer l'expression de V_S en fonction du courant de sortie du montage.
4. Calculer les valeurs minimales et maximales de la résistance de collecteur extérieure R_c en tenant compte des courants maximums possibles du montage.
5. Comparer les caractéristiques électriques obtenues à celles du circuit SN74LS00 données en annexe.
6. On désire commander une diode électroluminescente avec cette porte. Proposer un montage.

exercice 3.8

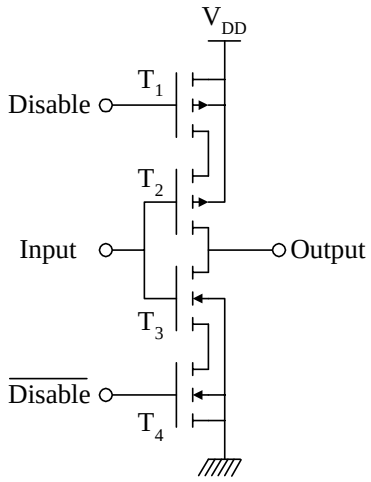
Soit la porte TTL trois états suivante :



1. Expliquer, sans faire de calculs, le fonctionnement du montage.
2. L'entrée $\overline{C}$ est au niveau bas. Calculer les tensions et courants du montage formé par les transistors T'1, T'2, T'3 et T'4 et déterminer l'expression de C en fonction du courant de sortie.
3. Les entrées A et B sont à 1. Calculer les tensions et courants du montage et déterminer l'expression de V_S en fonction du courant de sortie du montage.
4. L'entrée A passe à 0. Calculer les tensions et courants du montage et déterminer l'expression de V_S en fonction du courant de sortie du montage.
5. L'entrée $\overline{C}$ passe au niveau haut. Montrer que la sortie S passe en haute impédance.
6. Comparer les caractéristiques électriques obtenues à celles du circuit SN74LS00 données en annexe.

exercice 3.9

Soit le circuit suivant :

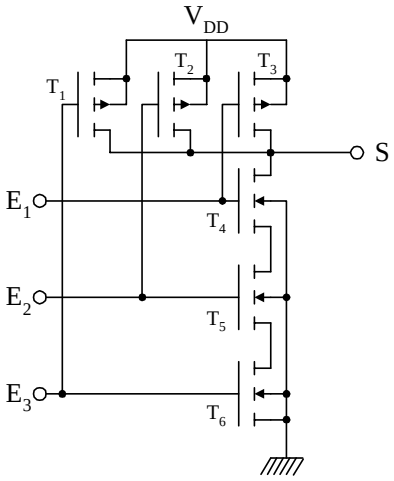


1. Expliquer le fonctionnement du montage transistor par transistor.
2. Remplir la table de vérité suivante (X ≡ état indifférent) :

Input	Disable	Output
1	0	
0	0	
X	1	

exercice 3.10

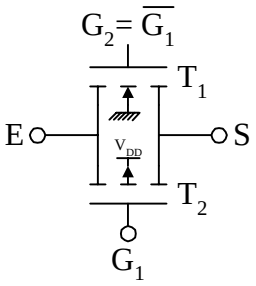
Soit le circuit suivant :



1. Expliquer le fonctionnement du montage transistor par transistor et donner sa table de vérité.
2. En déduire la fonction logique réalisée.

exercice 3.11

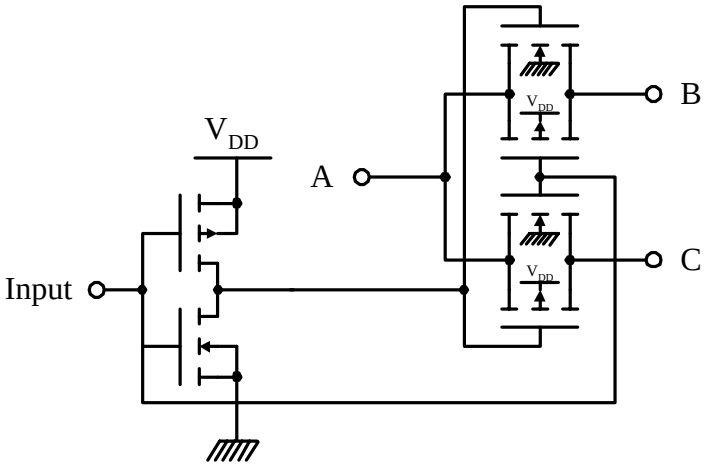
Soit le schéma de la porte de transmission CMOS suivant :



1. Expliquer le fonctionnement du montage transistor par transistor.
2. Remplir la table de vérité suivante :

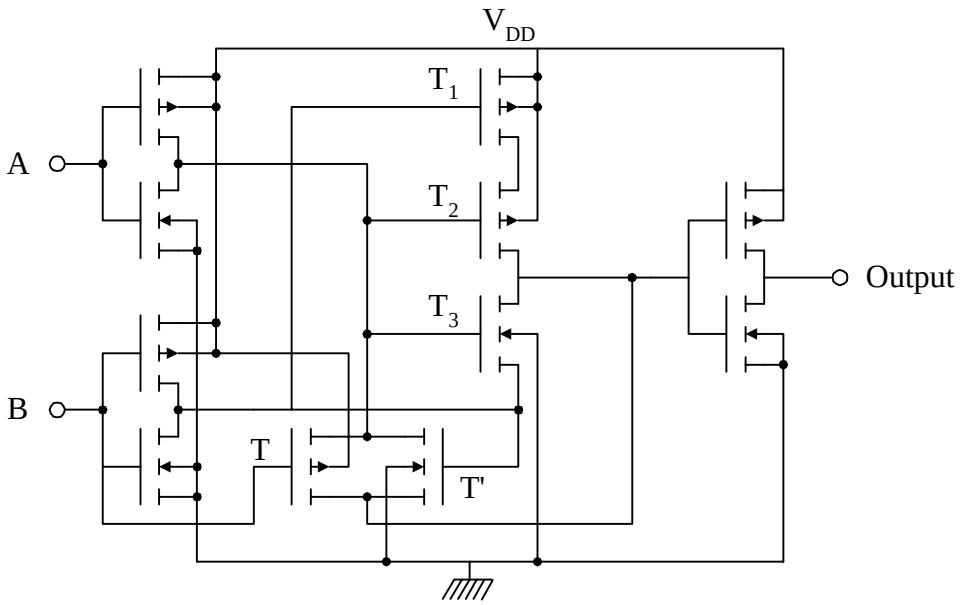
G_1	E	S
1	0	
1	1	
0	0	
0	1	

3. En déduire le fonctionnement du circuit suivant :



exercice 3.12

Soit le circuit suivant :



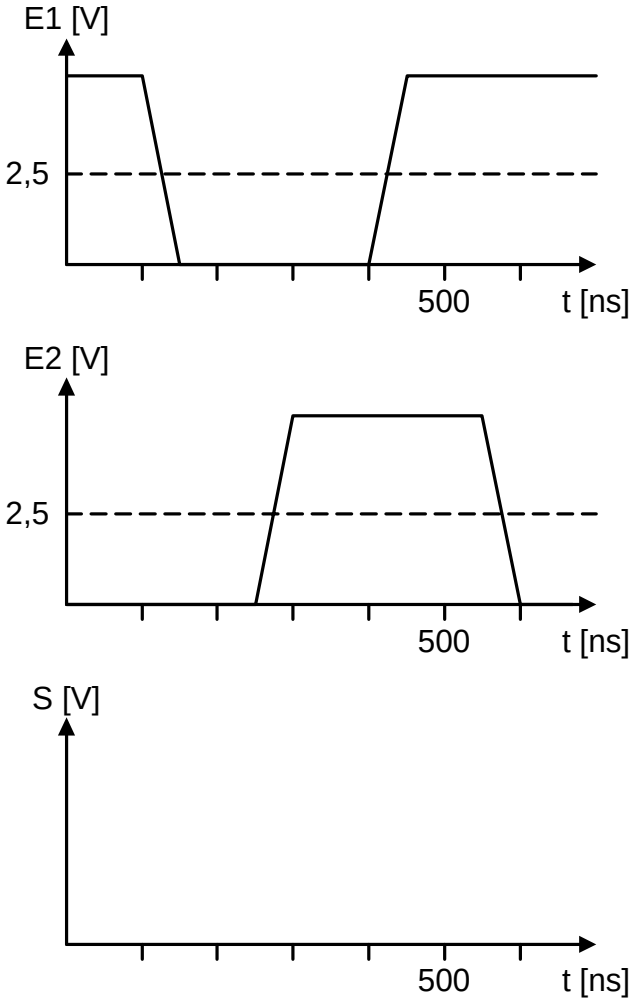
1. Expliquer le fonctionnement du montage transistor par transistor.
2. Remplir la table de vérité suivante :

B	A	Output
0	0	
0	1	
1	0	
1	1	

exercice 3.13

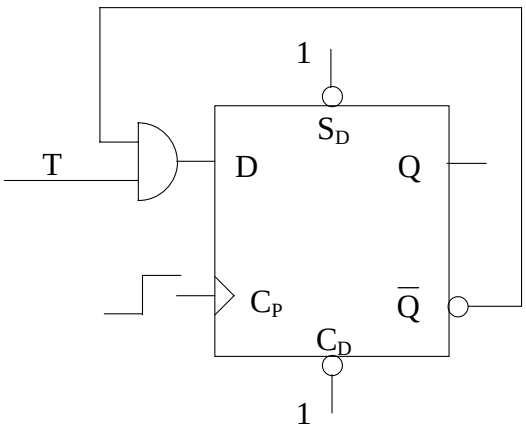
Soit le circuit NAND MC14011B alimenté en 5 V (voir les caractéristiques en annexe).

1. Indiquer sur le schéma de la porte le sens des 4 courants du tableau.
2. Pourquoi V_{OHmin} est-il toujours supérieur à V_{IHmin} et V_{OLmax} est-il toujours inférieur à V_{ILmax} ?
3. Donner la définition des marges de bruit à l'état haut et à l'état bas et calculer ces valeurs.
4. Combien d'entrées de porte MC14011B peut-on connecter à la sortie d'une porte de même type ?
5. Compléter le chronogramme suivant avec, connectée à la sortie de la porte, une charge capacitive de 10 pF, 50 pF et 100 pF.



exercice 3.14

Soit le montage suivant (alimenté en 5 V) composé d'une bascule D (MC14013B) et d'une porte AND (MC14081B).



1. Dessiner le chronogramme en sortie de la bascule.
2. Rappeler la définition et donner la valeur des temps de setup, de hold et de propagation des circuits composant le montage.
3. Quelle est la fréquence maximale (dans le pire des cas) de fonctionnement de ce montage ?
4. On a maintenant V_{CC} = 15 V. Quelle est la fréquence maximale (dans le pire des cas) de fonctionnement du montage ?
5. On revient à V_{CC} = 5 V. On connecte entre la sortie $\overline{Q}$ de la bascule D et la masse un condensateur C. Quelle est la fréquence maximale (dans le pire des cas) de fonctionnement du montage avec C = 50 pF et C = 100 pF ?

exercice 3.15

La loi d'Ohm thermique s'applique dans le cas des circuits intégrés numériques. Elle est définie par $T_J - T_A = R_{thj-a}.P_d$ avec T_J la température de jonction, T_A la température ambiante, R_{thj-a} la résistance thermique totale jonction $\rightarrow$ ambiance et P_d la puissance dissipée. Les données suivantes sont issues du data book Xilinx et donnent un exemple des caractéristiques thermiques d'un composant numérique :

- T_{jmax} = 125 °C (boîtier plastique) ou 150 °C (boîtier céramique),
- $0 < T_J < 85$ °C pour une série commerciale,
- quand $T_J > 85$ °C, les temps de propagation garantis par le constructeur augmentent de 0,35 % par degré supplémentaire jusqu'à T_{jmax},
- température de stockage : -65 °C $< T_A < +150$ °C,
- température de soudage : 260 °C pendant 10 secondes à une distance de 1,5 mm.
- Le tableau suivant donne des valeurs de résistance thermique totale (en °C/W) typiques pour différents types de boîtiers avec ou sans ventilation :

Boîtier	sans ventilation	ventilation 1,3 m/s	ventilation 2,5 m/s	ventilation 3,8 m/s
BG225	30	19	17	16
PG223	20	15	12	11
HQ240	12	9	7	6
PQ240	23	17	15	14
PC84	33	25	21	17

1. Tracer la courbe de puissance dissipée maximale en fonction de la température ambiante permettant d'obtenir les caractéristiques nominales du circuit pour un boîtier PLCC avec ventilation 1,3 m/s. Quelle puissance maximale peut-on dissiper à la température de 25 °C ?
2. Mêmes questions concernant un circuit fonctionnant en mode dégradé (température de jonction maximale).

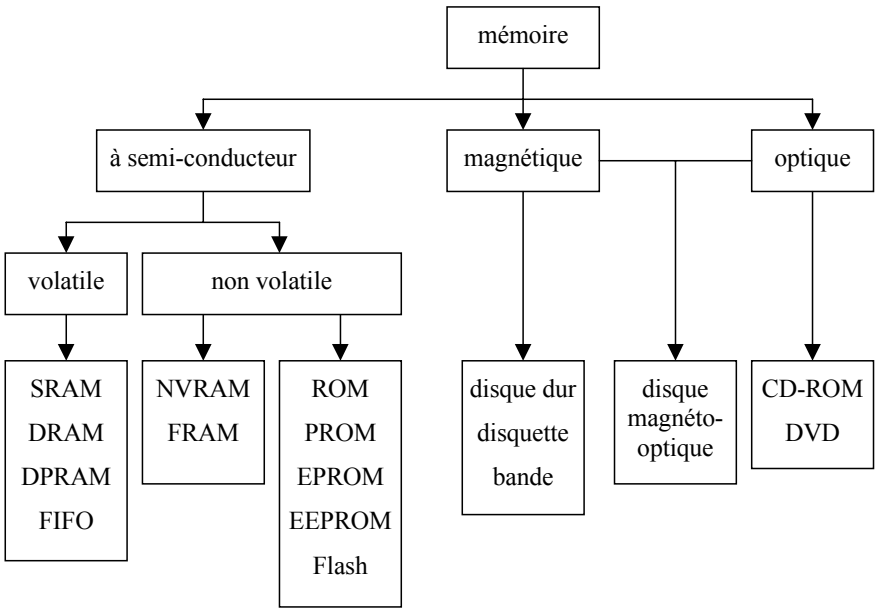
4. Les mémoires

Ce paragraphe va traiter des mémoires à semi-conducteurs utilisées principalement en tant que mémoires de stockage de type informatique ou plus rarement pour réaliser des fonctions logiques. Les circuits se trouvant à l'origine dans le TTL data book sont maintenant devenus obsolètes. A l'exception de quelques PROM bipolaire à fusibles, toutes les mémoires sont réalisées en technologie CMOS.

4.1 Généralités

4.1.1 Classification

Les mémoires servent à stocker de l'information numérique sous forme de bits (Binary digIT) pendant une certaine durée. Cette information peut être stockée sous forme de champs magnétiques rémanents (comme dans un disque dur par exemple), sous forme de modifications des propriétés optiques d'un matériau (comme dans un CD-ROM), sous forme d'une combinaison des deux effets précédents (comme dans un disque magnéto-optique) ou bien enfin sous forme de tensions et de courants dans un semi-conducteur. Ce dernier type de mémoire fait l'objet de ce paragraphe. L'organigramme suivant montre différentes mémoires existant à ce jour :



Il existe deux familles de mémoires à semi-conducteur :

- Les mémoires volatiles qui perdent l'information stockée en l'absence d'alimentation électrique. C'est la famille des RAM (Random Access Memory) ou mémoires à accès

sélectif (mémoires vives) ainsi nommées parce que l'on peut accéder directement à chaque élément de stockage de la mémoire (par opposition au registre à décalage qui est une mémoire à accès séquentiel). Il en existe de plusieurs types :

- ✓ les mémoires statiques ou SRAM (Static RAM) qui conservent l'information enregistrée tant que l'alimentation électrique est connectée sur le circuit,
- ✓ les mémoires dynamiques ou DRAM (Dynamic RAM) qui perdent l'information enregistrée au cours du temps et doivent être rafraîchies,
- ✓ les mémoires spéciales qui sont basées sur la technologie SRAM. Il existe les mémoires double port ou DPRAM (Dual Port RAM), les piles « premier entré premier sorti » FIFO (First In First Out) et bien d'autres de moindre importance.

- Les mémoires non volatiles conservent l'information stockée en l'absence d'alimentation électrique. C'est la famille des ROM (Read Only Memory) ou mémoires à lecture seule ou mémoires mortes. Elles peuvent être programmées une fois pour toute en usine (c'est le cas d'une ROM) ou bien programmable sur site. Dans ce cas, on trouve les circuits suivants :
  - ✓ les mémoires programmables une seule fois ou PROM (Programmable ROM),
  - ✓ les mémoires programmables électriquement et effaçables aux rayons ultraviolets ou EPROM (Erasable PROM),
  - ✓ les mémoires programmables et effaçables électriquement ou EEPROM (Electrically EPROM) ou bien EEPROM Flash.

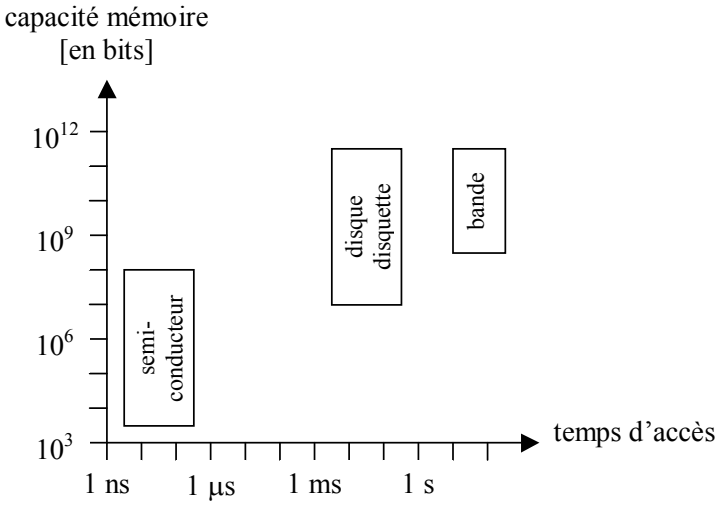
  Mais on trouve aussi certaines RAM non volatiles telles que :
  - ✓ les NVRAM (Non volatile RAM),
  - ✓ les mémoires ferroélectriques FRAM (Ferroelectric RAM).

Les mémoires possèdent deux caractéristiques principales, leur capacité et leur temps d'accès.

- La capacité d'une mémoire est définie par le nombre de bits d'information que l'on peut y stocker. Cette capacité s'exprime en Kbits ou K (1 kilobits = 2^{10} bits = 1024 bits) ou en Mbits ou MEG (1 Mbits = 2^{20} bits = 1048576 bits) et d'ici quelques années en Gbits (1 Gbits = 2^{30} bits = 1073741824 bits). Par exemple, une mémoire 256K a une capacité de 256x1024 bits. Le mot stocké dans une case de la mémoire n'est pas forcement 1 bit unique. Il peut aussi être codé sur 4, 8, 16 ou 32 bits. Par exemple, une mémoire 8 MEG x 8 a une capacité égale à 8 Mega-octets.

- Le temps d'accès d'une mémoire est l'intervalle de temps qui sépare la présentation de l'adresse de la donnée à l'entrée du circuit et l'apparition effective de cette donnée en sortie. Il s'agit du temps qu'il faut pour accéder à cette donnée.

Le diagramme suivant présente un classement des différentes familles de mémoires en fonction de leur capacité et de leur temps d'accès.



On choisit le type de mémoire en fonction de la quantité d'information à stocker, du prix de l'octet et du temps d'accès. Les mémoires centrales des ordinateurs sont à semi-conducteurs à cause de leur temps d'accès de quelques dizaines de nanosecondes. On stocke les informations à traiter sur des disques (optique, magnétique ou magnéto-optique) de bien plus grande capacité ayant des temps d'accès de l'ordre de quelques millisecondes (à quelques dizaines). La sauvegarde des disques et le stockage de longue durée utilisent des bandes magnétiques qui ont une bonne fiabilité mais un temps d'accès séquentiel de l'ordre de la minute.

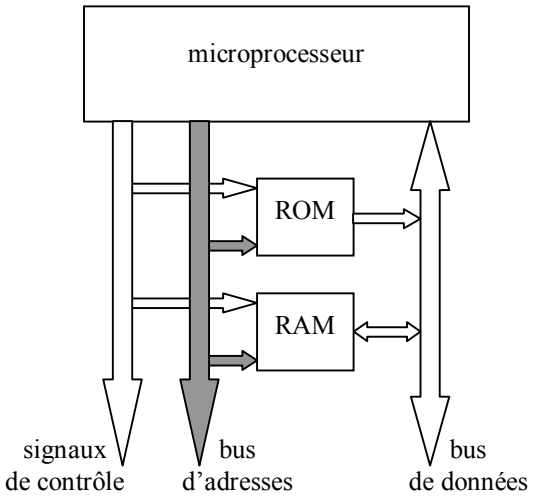
4.1.2 Principe d'un microprocesseur

Les micro-ordinateurs de type PC (Personnal Computer) sont les principaux consommateurs de mémoires (environ 50 % du marché des DRAM en 1997). Ils incorporent à la fois des mémoires mortes qui contiennent les logiciels de démarrage du microprocesseur et des mémoires vives où sont stockés les programmes utilisateurs. Comme le montre la figure suivante, le microprocesseur possède :

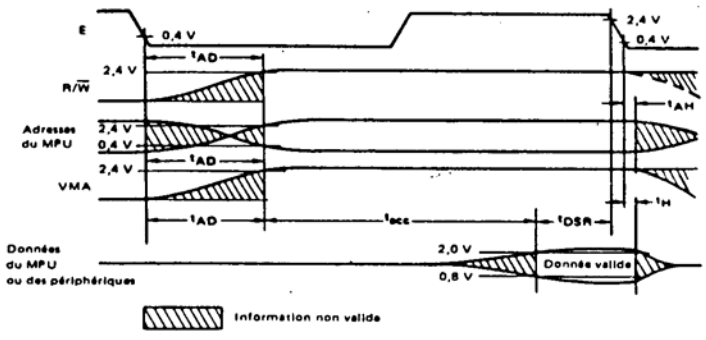
- un bus d'adresses. Un bus est le regroupement d'un ensemble de broches. Si on a, par exemple, 16 broches d'adresses numérotées A0, A1, A2, …, A15, on parle d'un bus

d'adresses sur 16 bits. Le bus d'adresses permet d'adresser une case mémoire particulière pour en extraire ou y écrire une donnée.

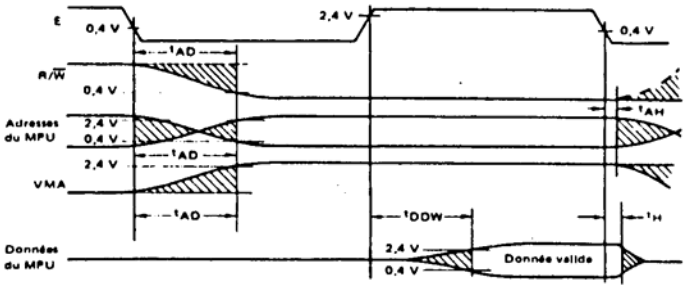
- un bus de données. Il permet de lire ou bien d'écrire une donnée (sur 16 bits par exemple) dans la mémoire.
- un bus de contrôle. Les signaux de contrôle permettent de donner des ordres aux boîtiers périphériques comme par exemple l'ordre de lecture ou bien d'écriture dans une mémoire.



Les deux chronogrammes suivants représentent les cycles de lecture et d'écriture d'un microprocesseur 6802 de Motorola. Il s'agit d'un modèle ancien quoique toujours commercialisé qui nous suffira pour comprendre les mécanismes de base des accès à la mémoire. On trouve sur ces chronogrammes un signal d'horloge E, un signal indiquant qu'une adresse valide se trouve sur le bus d'adresses VMA (Valid Memory Address), le signal de lecture/écriture $R/\overline{W}$ ainsi que les bus d'adresses et de données. La lecture des données en mémoire ($R/\overline{W}$ = 1) s'effectue de la manière suivante. Les adresses sont valides un temps t_{AD} après le front descendant de E. Les données en sortie de la mémoire doivent apparaître un temps t_{DSR} avant le front descendant suivant de E pour que le microprocesseur puissent les lire correctement.



Lorsque le microprocesseur écrit des données en mémoire ($R/\overline{W}$ = 0), les adresses apparaissent un temps t_{AD} après le front descendant de E puis les données apparaissent un temps t_{DDW} après le front montant de E et sont maintenues un temps t_H après le front descendant suivant de E.



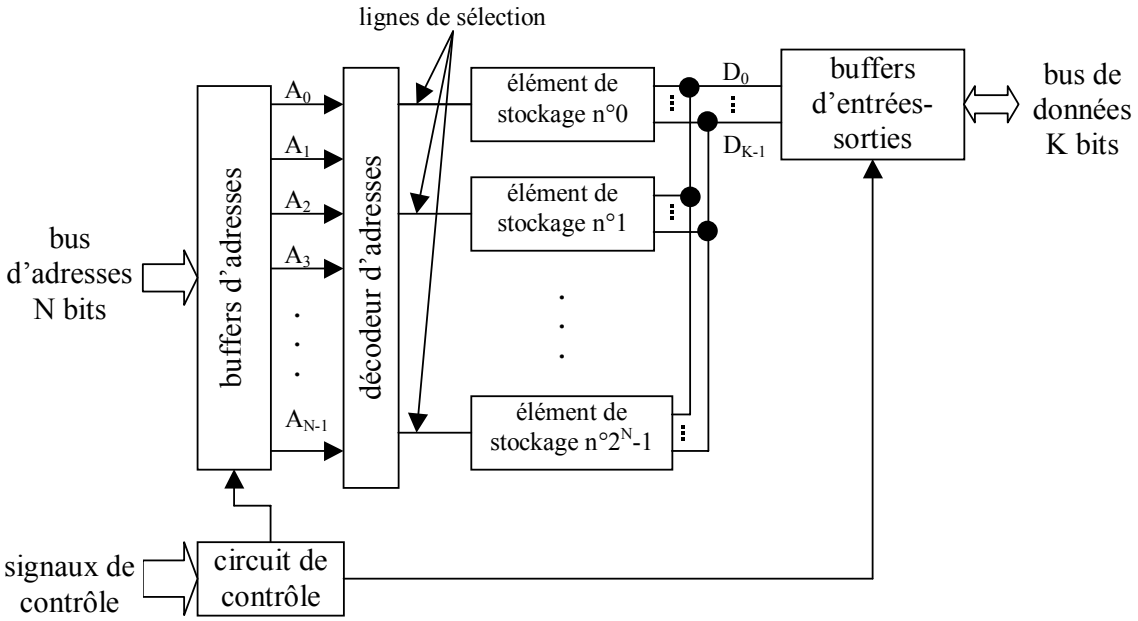
Le temps d'accès de la mémoire doit être choisi pour respecter les différents temps du microprocesseur. La structure du microprocesseur va déterminer en grande partie les différents signaux nécessaires à son bon fonctionnement.

4.1.3 Structure générale

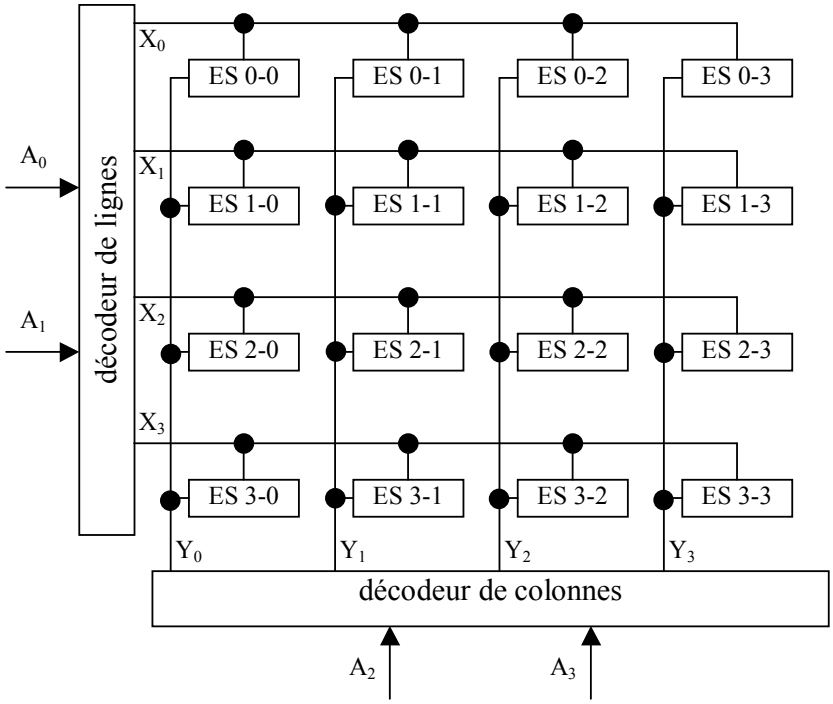
Une mémoire comprend 4 classes de broches :

1. Une ou plusieurs broches pour les données (nommées généralement D_0, D_1, …, D_{k-1}).
2. Un bus d'adresses. La capacité de la mémoire doit être égal à $2^{\text{largeur du bus}}$. Par exemple, une mémoire morte 256K nécessite 18 broches d'adresses car 2^{18} = 256K.
3. Une ou plusieurs broches pour sélectionner le boîtier. Ces broches sont généralement nommées CS0, CS1, … (Chip Select) ou bien CE0, CE1… (Chip Enable). Si le boîtier n'est pas sélectionné, le bus de données reste à l'état haute impédance.
4. Des broches de commande comme par exemple l'autorisation de lecture ou d'écriture. Ce signal peut exister sous la forme d'une seule broche, le signal de lecture/écriture $R/\overline{W}$ (Read/Write) qui vaut 1 en lecture et 0 en écriture, ou bien de deux broches, le signal d'autorisation d'écriture $\overline{WE}$ (Write Enable) et le signal d'autorisation de lecture $\overline{OE}$ (Output Enable).

La structure interne d'une mémoire est composée de trois parties (voir la figure suivante). Les circuits d'entrées-sorties et de contrôle (buffer d'entrées et de sorties, gestion de la sélection du boîtier et gestion des opérations de lecture/écriture), le décodeur d'adresses (qui permet à partir de l'adresse de sélectionner la bonne case mémoire) et la zone de stockage proprement dite (où sont effectivement stockées les informations binaires).

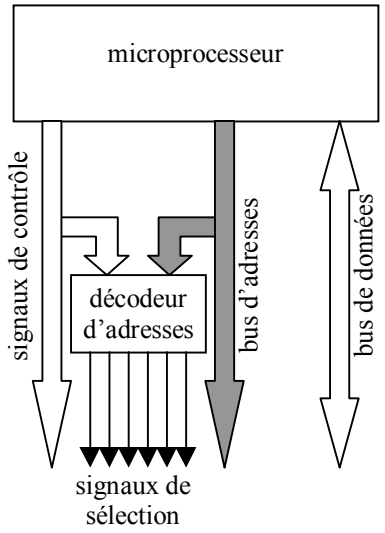


En fait, cette structure simplifiée n'est utilisée que pour les mémoires de petites tailles. Quand la capacité augmente, le nombre de portes utilisées pour réaliser le décodeur devient beaucoup trop élevé. On utilise donc, comme sur la figure suivante, une organisation matricielle pour ranger les éléments de stockage et deux décodeurs d'adresses : un décodeur de lignes X_n et un décodeur de colonnes Y_n. Nous verrons au §5.2 comment on accède aux données avec cette structure de mémoire (l'accès aux données n'est pas représenté sur cette figure).



4.1.4 Plan d'adressage

Il y a rarement un seul boîtier périphérique relié avec le microprocesseur. Chaque boîtier doit donc posséder son adresse ou plus généralement sa plage d'adresses dans le cas d'une mémoire. Elle doit être sélectionnée à l'aide des broches CS ou CE pour la zone mémoire qui lui est affectée. On utilise pour cela un décodeur d'adresses qui va générer les signaux permettant de sélectionner le bon boîtier à la bonne adresse, tous les autres boîtiers étant déselectionnés à ce moment.

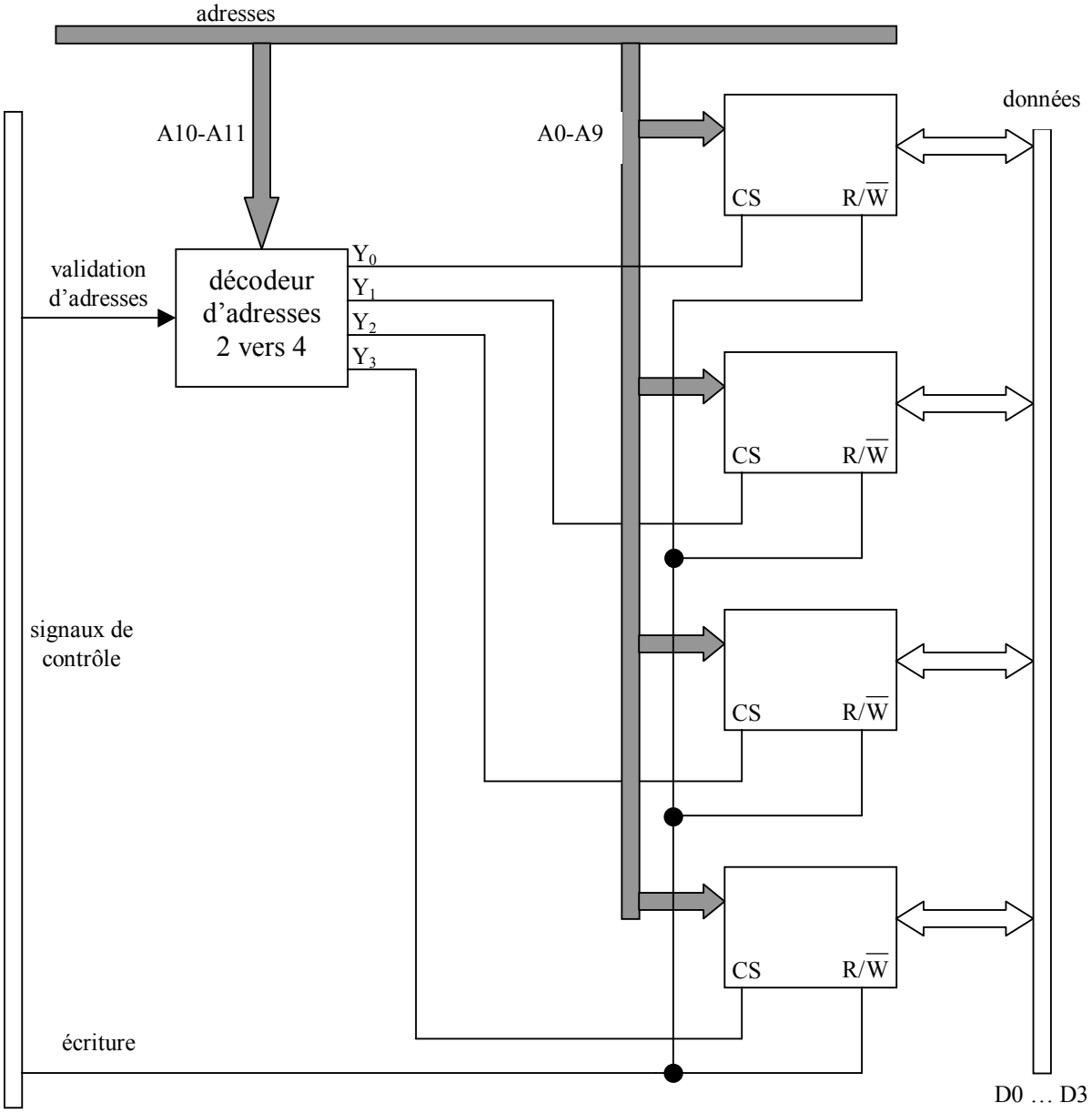


4.1.5 Expansion en capacité

Prenons l'exemple de l'extension de la capacité d'une mémoire RAM. On souhaite réaliser une mémoire de 4K x 4 bits à partir de mémoire 1K x 4 bits. Il faut monter en parallèle 4 boîtiers avec les lignes d'adresses A0-A9 et le signal d'autorisation de lecture/écriture mis en commun. L'entrée de sélection CS est issue d'un décodeur d'adresses qui fournit une zone d'adresses de 1K différente pour chaque boîtier selon le tableau suivant :

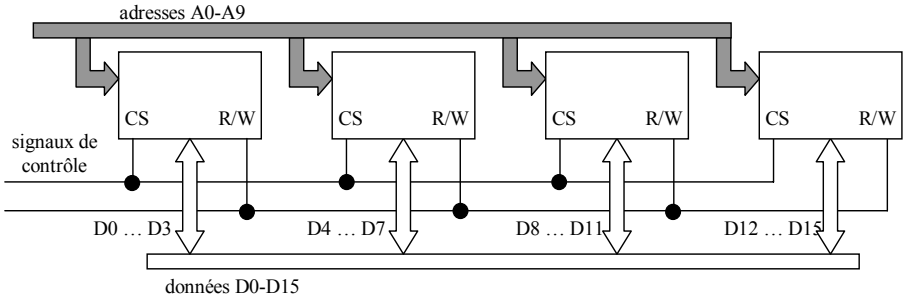
	VMA	A11	A10	A9-A0	zone d'adresses
Y_0	1	0	0	X	de 000000000000 à 001111111111
Y_1	1	0	1	X	de 010000000000 à 011111111111
Y_2	1	1	0	X	de 100000000000 à 101111111111
Y_3	1	1	1	X	de 110000000000 à 111111111111

Le schéma obtenu est alors le suivant :



4.1.6 Expansion de la largeur du bus de données

La largeur du bus de données d'un microprocesseur est généralement de 4, 8, 16, 32 ou 64 bits ce qui ne correspond pas forcément à la largeur du mot de données des mémoires disponibles. Si on veut augmenter la taille du mot d'une mémoire, il faut mettre en parallèle un certain nombre de boîtiers, les signaux d'adresses, de sélection et de lecture/écriture étant mis en commun. La largeur du mot est alors égale au nombre de bits de données de la mémoire utilisée multiplié par le nombre de boîtiers mis en parallèle. Dans l'exemple suivant, on réalise une mémoire 1K x 16 bits avec 4 boîtiers 1K x 4 bits.



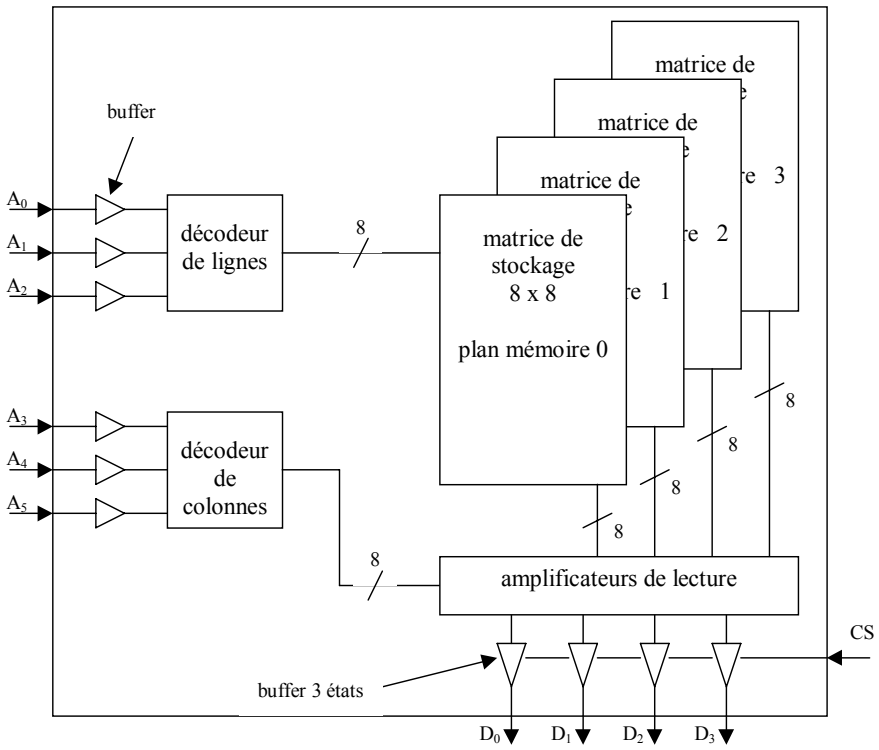
Les deux types d'expansion peuvent être utilisés simultanément pour étendre la capacité et la largeur du mot de données.

4.2 La famille des ROM

4.2.1 ROM et PROM

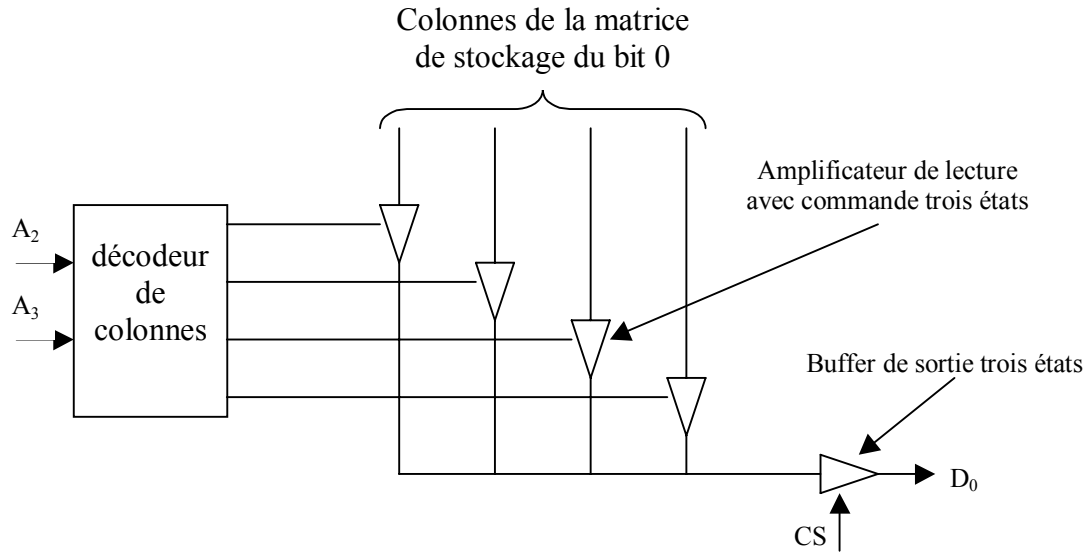
4.2.1.1 Principe général

Une ROM est une mémoire à lecture seule qui est définitivement programmée au moment de sa fabrication. Seule une très grande série permet d'en amortir le coût. Une PROM a l'avantage d'être programmable sur site par l'utilisateur à l'aide d'un programmateur. Cette solution est plus flexible, mais elle coûte plus chère pour la fabrication en série. L'architecture interne de ces deux mémoires est identique, seul l'élément de stockage diffère. Le schéma suivant nous montre la structure d'une ROM 64 x 4 bits.



La fonction des différentes parties est la suivante :

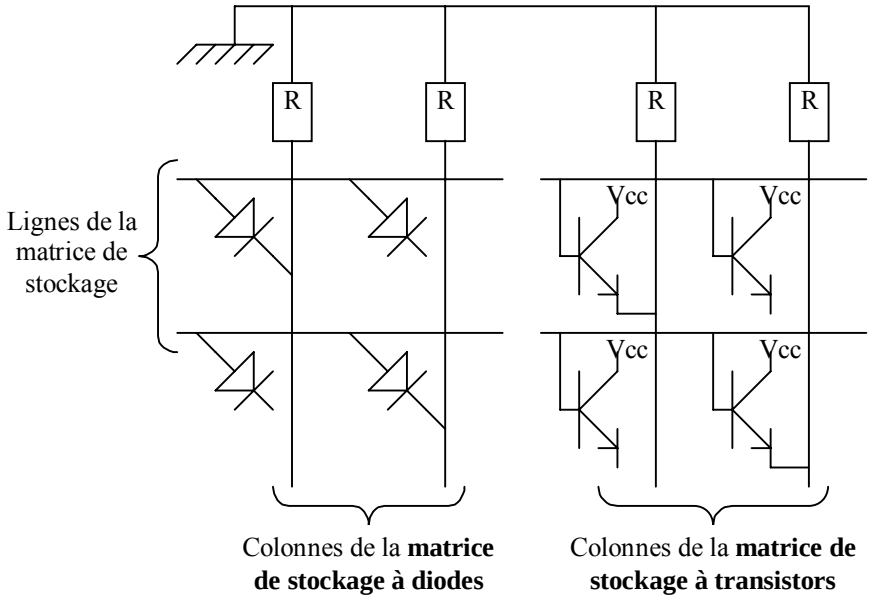
- Les buffers d'adresses servent à adapter les niveaux de tension du bus aux niveaux de tension requis dans la mémoire.
- Les décodeurs de lignes et de colonnes permettent de sélectionner l'élément de stockage.
- Les matrices de stockage réalisent la mémorisation proprement dite. Chaque plan mémoire correspond à un bit de donnée.
- Les buffers de données (trois états) fournissent les courants nécessaires pour attaquer le bus de données.
- Les amplificateurs de lecture servent à amplifier le signal (qui peut être assez faible) issu de l'élément de stockage pour l'amener à un niveau compatible avec les buffers de sortie. Le schéma suivant montre le détail des amplificateurs de lecture et du buffer de sortie pour un plan mémoire 4x4 :



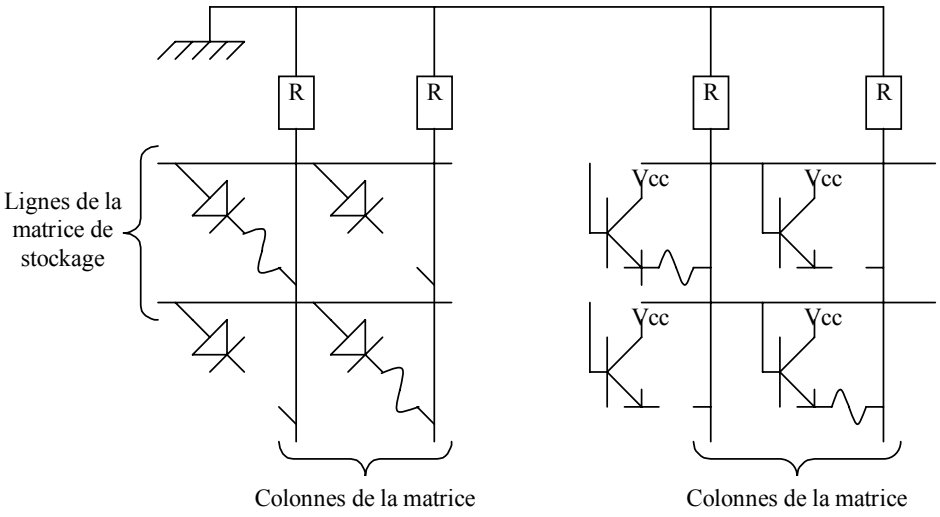
La fonction logique réalisée par l'ensemble formé du décodeur de colonnes, des amplificateurs de lecture et du buffer de sortie peut être vue comme un multiplexeur 4 vers 1 commandé par les adresses A_2 et A_3.

La figure suivante montre deux éléments de stockage utilisables pour réaliser une ROM : la diode et le transistor bipolaire. S'il correspond à une valeur 1, l'élément de stockage réalise la

connexion entre la ligne et la colonne correspondant à une adresse. Dans le cas de la diode, si la ligne vaut 1, alors la diode conduit et le niveau 1 se trouve appliqué sur la colonne. Si la cathode de la diode n'est pas reliée à la colonne, alors l'élément binaire mémorisé vaut 0. La connexion ou l'absence de connexion est réalisée par masque lors de la fabrication du circuit intégré. Dans les mémoires bipolaires, on remplace la diode par un transistor bipolaire. La ligne est alors reliée à la base et le collecteur à Vcc. Dans le cas d'un bit à 1, l'émetteur est relié à la colonne alors qu'il est en l'air dans le cas d'un bit à 0.



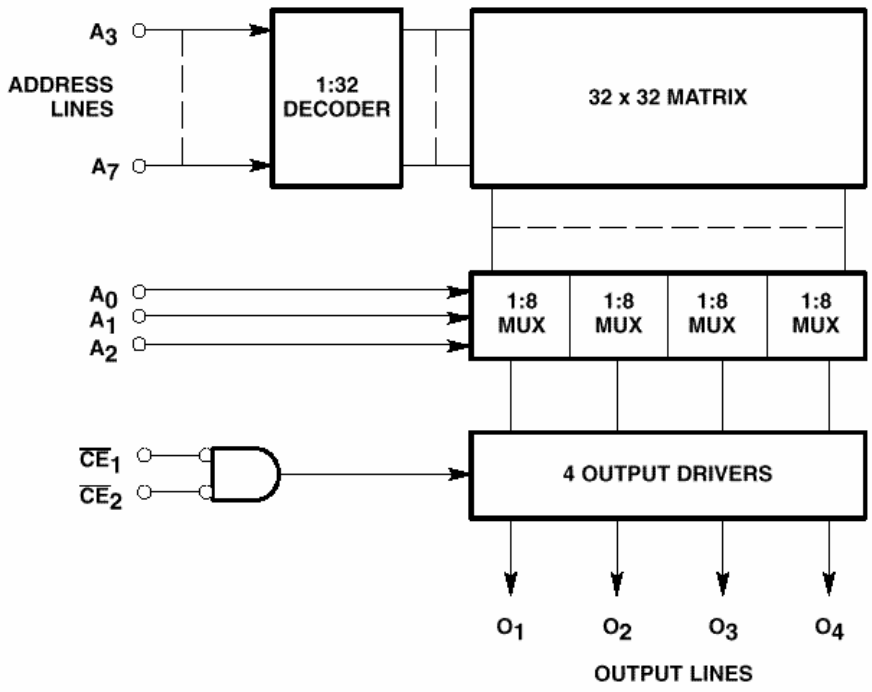
Les ROM sont aujourd'hui principalement réalisées en technologie CMOS (il suffit de remplacer le transistor bipolaire dans le dessin ci-dessus par un transistor MOSFET). Dans le cas de prototypes ou de petites séries, il n'est pas rentable de faire fabriquer une ROM chez un fondeur. On préfère alors utiliser une ROM programmable sur site ou PROM. L'architecture de la PROM est identique à celle de la ROM, sauf qu'un fusible réalise la connexion entre l'élément de stockage et la colonne. Pour programmer une PROM, on doit faire circuler un courant assez fort (20 à 30 mA) pour faire fondre les fusibles selon les valeurs à stocker. La programmation est effectuée à l'aide d'un programmateur de PROM qui vérifie aussi l'exactitude de la programmation. La figure suivante montre deux exemples d'éléments de stockage utilisables pour réaliser une PROM à fusibles : la diode et le transistor bipolaire.



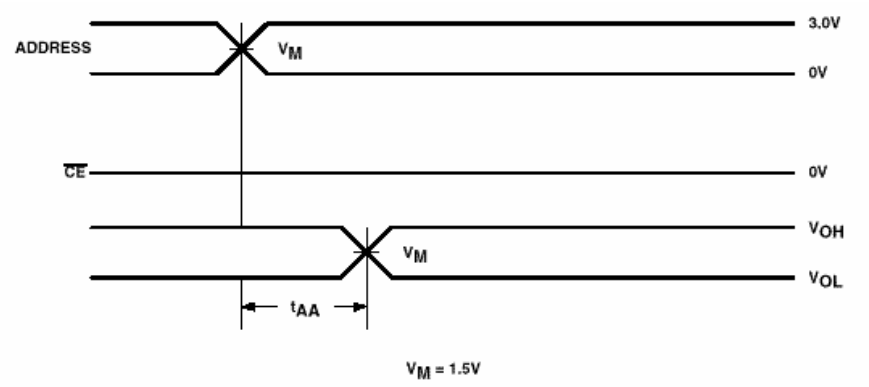
Les PROM bipolaires ou MOS sont aujourd'hui quasiment obsolètes. Leur technologie de fabrication n'évolue plus depuis au moins 10 ans et elles sont pratiquement remplacées par les PROM effaçables ou par les PAL (Programmable Array Logic). Par contre, les ROM sont toujours utilisées en technologie CMOS pour les grandes séries.

4.2.1.2 Exemple : la 82S129A de Philips

Philips est un des rares constructeurs a avoir encore à son catalogue des PROM bipolaires. La 82S129A est une PROM TTL bipolaire 256 x 4. Son temps d'accès est égal à 35 ns. Elle existe en boîtier DIP ou Flat Pack 16 broches. Sa consommation est supérieure à 100 mA. Son architecture interne est la suivante :



Le chronogramme suivant montre une opération de lecture en mode simplifié (avec $\overline{CE}$ en permanence à 0) :

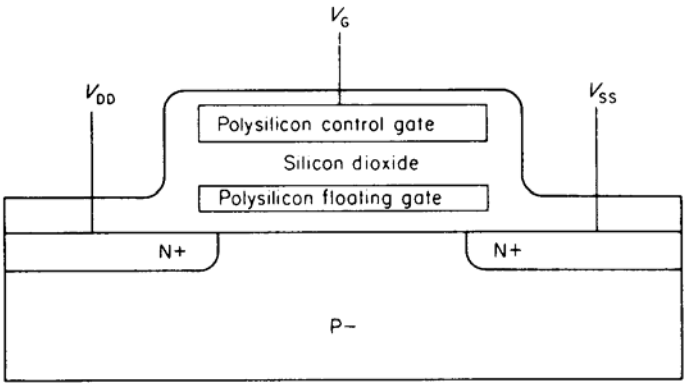


Le temps t_{AA} correspond au temps d'accès de la mémoire.

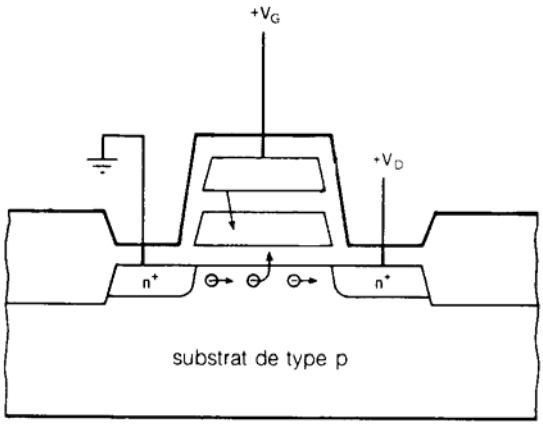
4.2.2 EPROM et OTP

4.2.2.1 Principe général

Le principal inconvénient de la PROM est sa programmation définitive. Son seul avantage sur la ROM est sa programmation sur site. L'idée de concevoir une ROM programmable plusieurs fois est apparue très rapidement. Dès 1971, Intel inventait le transistor FAMOS (Floating gate Avalanche injection MOS). Ce transistor permettait de réaliser les premières EPROM (Erasable PROM) programmables électriquement et effaçables aux rayons ultraviolets. La structure interne d'une EPROM est identique à celle d'une ROM, le transistor MOS de l'élément de stockage étant remplacé par un transistor FAMOS. Il y a aussi de la logique supplémentaire pour la programmation. La figure suivante montre que le transistor FAMOS possède deux grilles. La grille supérieure est utilisée pour la sélection et est connectée au décodeur de ligne. La grille inférieure entre la grille de sélection et le substrat est dite flottante car elle n'est reliée à rien. Elle est entièrement isolée par l'oxyde de silicium (SiO2).



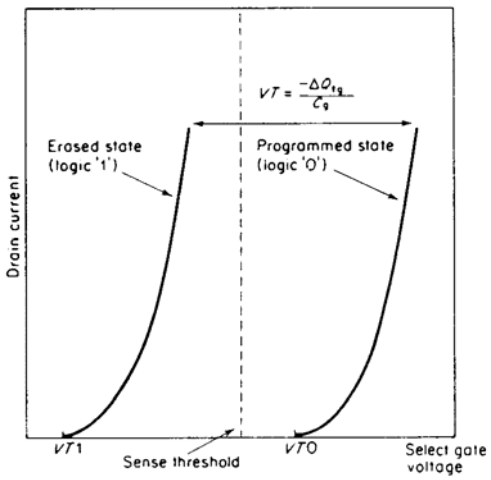
La programmation de la cellule (voir figure suivante) est réalisée en appliquant une tension positive élevée sur le drain et sur la grille de commande (≈ 12.5 V aujourd'hui) alors que la source est à la masse. Les électrons qui apparaissent alors dans le canal, soumis à un champ électrique grille-canal élevé passe à travers l'isolant dans la grille flottante par effet d'avalanche. On appelle ce phénomène l'injection d'électrons chauds car les électrons doivent avoir une énergie suffisante pour franchir la barrière d'énergie de 3.2 eV se trouvant à l'interface entre le substrat et l'isolant. Comme la grille flottante se charge négativement, elle tend à repousser les électrons et à mettre fin au transfert. Le processus de programmation est donc auto-limitatif et tend à s'arrêter de lui-même.



La cellule est entièrement effacée par une exposition aux rayons ultraviolets (longueur d'onde égale à 253.7 nm) qui accroissent l'énergie des électrons stockés dans la grille flottante jusqu'à ce qu'elle dépasse la barrière d'énergie de 3.2 eV existant entre la grille flottante et l'isolant. Les électrons quittent alors la grille qui redevient électriquement neutre, toutes les cellules étant programmées à l'état logique 1. Le boîtier d'une EPROM doit nécessairement posséder une fenêtre transparente en quartz pour laisser passer les rayons ultraviolets. L'effacement peut aussi être causé accidentellement par la lumière du jour ou la lumière électrique fluorescente. Pour éviter ce phénomène, un adhésif opaque doit être appliqué sur la fenêtre transparente après programmation. Le durée normale d'effacement avec un appareil du commerce (effaceur d'EPROM) est d'environ 30 minutes. La programmation d'une EPROM s'effectue à l'aide d'un appareil appelé programmateur d'EPROM.

La charge stockée sur la grille flottante lors de la programmation modifie la valeur de la tension de seuil V_T du transistor FAMOS. Dans l'état non programmé, la tension de seuil est faible et le transistor devient passant quand il est sélectionné. Dans l'état programmé avec stockage de charges dans la grille flottante, la charge négative stockée s'oppose à la création

de la couche d'électrons formant le canal. La tension de seuil est donc plus élevée et le transistor ne devient pas passant quand il est sélectionné. La figure suivante montre les courbes de courant de drain en fonction de la tension de grille pour un transistor FAMOS programmé (état logique 0) et non-programmé (état logique 1).



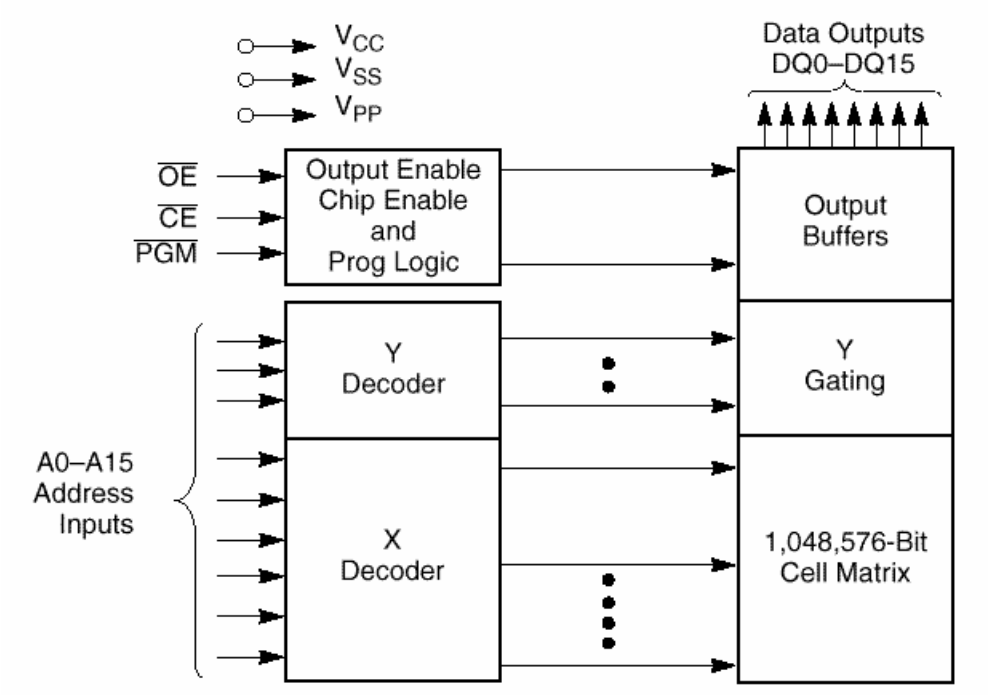
Quand une charge Q_{fg} est stockée dans la grille flottante, la tension de seuil est décalée de la quantité $\Delta V_T = \frac{Q_{fg}}{C_g}$ où C_g est la capacité entre les deux grilles. L'oxyde de silicium étant un très bon isolant, la grille flottante ne se décharge quasiment pas au cours du temps. La durée de rétention de l'information est supérieure à 10 ans pour une température de 70 °C. Le nombre maximum de cycles de programmation est supérieur à 1000. Le prix de revient de cette mémoire permanente, fiable et très répandue est assez faible puisqu'elle utilise une cellule de stockage à un seul transistor. C'est le boîtier la partie la plus coûteuse d'une EPROM à cause de la fenêtre en quartz qui nécessite obligatoirement un boîtier céramique dont le prix est bien supérieur à un boîtier plastique. C'est le prix de la reprogrammation. Cette fonctionnalité est très utile pendant la mise au point d'un système électronique, mais elle ne sert plus à rien lors de la production en série. Les fabricants d'EPROM ont donc créé les mémoires programmables une fois ou OTP (One Time Programming) qui sont strictement identiques aux EPROM, mais encapsulées dans un boîtier plastique qui les rend non effaçable. Cette approche présente de nombreux avantages :

- Baisse du prix de revient par rapport à une EPROM,
- Utilisation du même programmateur que pour une EPROM,
- Caractéristiques électriques identiques à celle de l'EPROM correspondante.

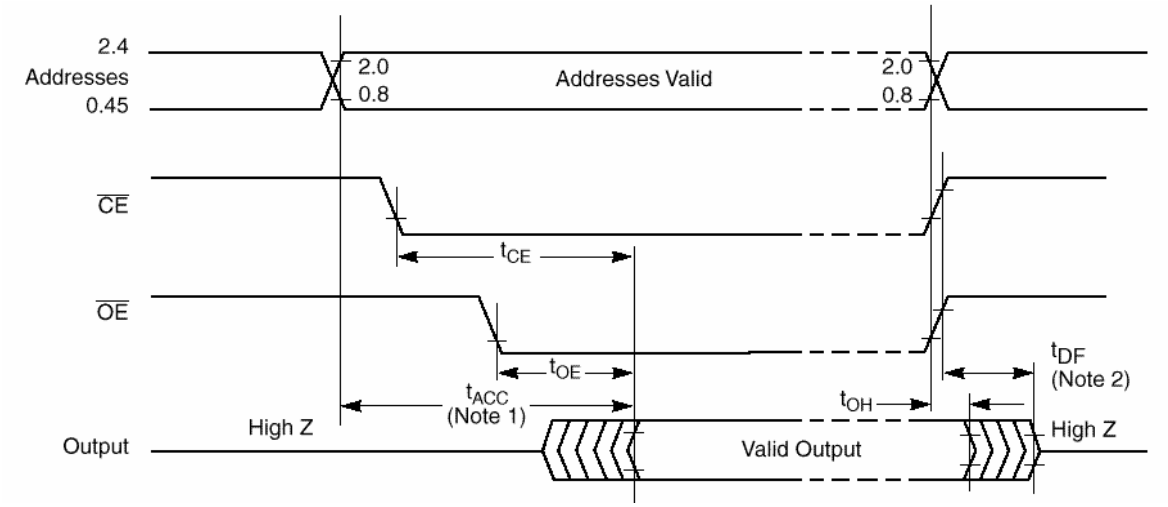
Tous les fabricants proposent les EPROM dans les deux versions UVPROM et OTP. Pour une fois, les fabricants se sont entendus lors de la création de cette famille de mémoire et ont adopté des brochages identiques et des appellations compatibles. Ainsi, une 2764 est une mémoire 64 Kbits (8K x 8) et une 27512 une 512 Kbits (64K x 8). Au-dessus de 1 Mbits, la situation se complique car les mémoires peuvent être organisées en mots de 8 ou 16 bits. Une 27010 devient alors une 128K x 8 alors qu'une 271024 est une 64K x 16.

4.2.2.2 Exemple : la 27C1024 d'AMD

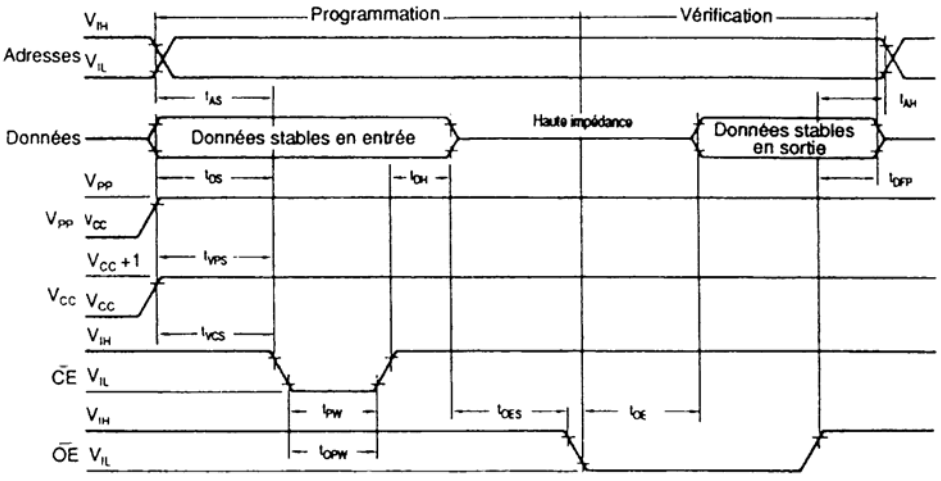
On trouve chez AMD une gamme d'EPROM allant de la 27C64 (8K x 8) à la 27C4096 (256K x 16) en UVPROM ou en OTP. L'AM27C1024 est une EPROM CMOS 64K x 16 alimentée en 5 V dont le temps d'accès est compris entre 55 ns et 250 ns. Elle existe en boîtier DIP 40 broches ou PLCC 44 broches. Sa consommation est égale à 30 mA en fonctionnement typique et descend à 100 µA en mode stand-by (boîtier non sélectionné). Le diagramme de blocs de cette mémoire est le suivant :



Le chronogramme suivant montre une opération de lecture (le temps t_{ACC} correspond au temps d'accès de la mémoire) :



Du fait de la technologie utilisée, la programmation d'une cellule EPROM n'est pas forcément bonne du premier coup. Elle doit être suivie d'une phase de vérification, puis recommencée en cas d'échec. Il y a donc un algorithme de programmation implanté dans le programmateur d'EPROM qui a la charge de garantir la validité de la programmation complète du composant. La durée minimale (toutes les écritures étant bonnes du premier coup) pour programmer entièrement la mémoire est égale à 8 secondes. Un cycle de programmation/vérification s'effectue en présentant le mot de 16 bits à programmer sur les sorties de l'EPROM, en mettant $\overline{PGM}$ à l'état bas et en respectant le chronogramme suivant :



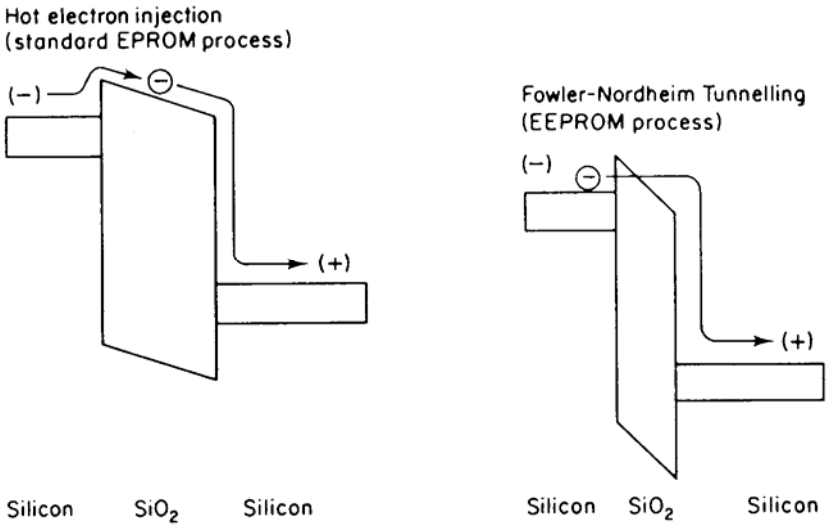
Une signature électronique permet d'identifier automatiquement le fabricant ainsi que le type de l'EPROM. Il suffit de mettre $\overline{OE}$ et $\overline{CE}$ à l'état bas, d'appliquer 12 V sur A9 et 0 V sur A0 pour obtenir 01 sur les données, ce qui correspond au code d'AMD. En appliquant ensuite

12 V sur A0, on lit le code du circuit 8C en sortie. Ces codes sont destinés à permettre la reconnaissance automatique du circuit par le programmateur d'EPROM.

4.2.3 EEPROM

4.2.3.1 Principe général

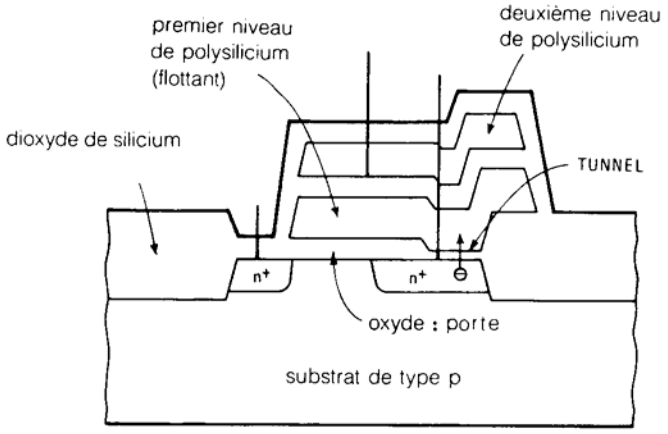
Le principal inconvénient de l'EPROM est d'être obliger de la retirer de l'équipement pour pouvoir l'effacer puis la reprogrammer. L'EEPROM (Electrically Erasable PROM) ou E^2PROM a été inventée pour pouvoir être reprogrammée dans l'équipement : c'est ce que l'on appelle l'ISP (In Situ Programming). Pour cela, le transistor FAMOS a été modifié pour pouvoir être effacé et programmé électriquement. Cette nouvelle cellule de stockage a été inventée par Intel au début des années 80 et s'appelle FLOTOX (FLOating gate Tunneling OXide). La charge et la décharge de la grille flottante utilisent maintenant l'effet tunnel de FOWLER-NORDHEIM. Dans la cellule FAMOS, on communiquait aux électrons se trouvant dans le canal une énergie suffisante pour passer par-dessus la barrière d'énergie de 3.2 eV existant entre la grille flottante et le substrat (voir figure de gauche ci-dessous). Dans la cellule FLOTOX, on utilise l'effet tunnel pour faire passer les électrons au travers de la barrière d'énergie (voir figure de droite ci-dessous).



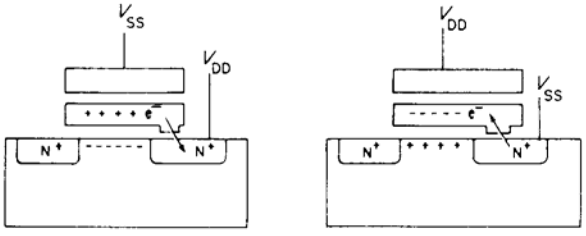
au-dessus de la barrière d'énergie au travers de la barrière d'énergie

On les appelle des électrons froids. L'effet tunnel à électrons froids est un effet dû à la mécanique quantique qui permet aux électrons de traverser la barrière de 3.2 eV alors que leur

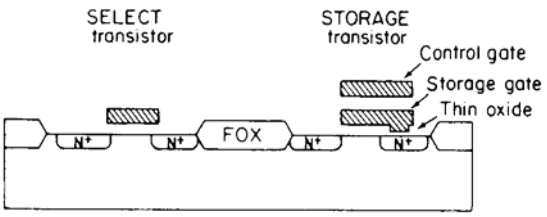
énergie est inférieure à cette barrière. Il ne se produit que si l'épaisseur de la zone à traverser est suffisamment faible. La figure ci-dessous montre la structure du transistor de stockage. On a toujours deux grilles, une grille flottante isolée dans l'oxyde de silicium et une grille de contrôle se trouvant juste au-dessus. L'épaisseur de SiO2 entre le drain et la grille flottante est suffisamment faible pour permettre l'effet tunnel.



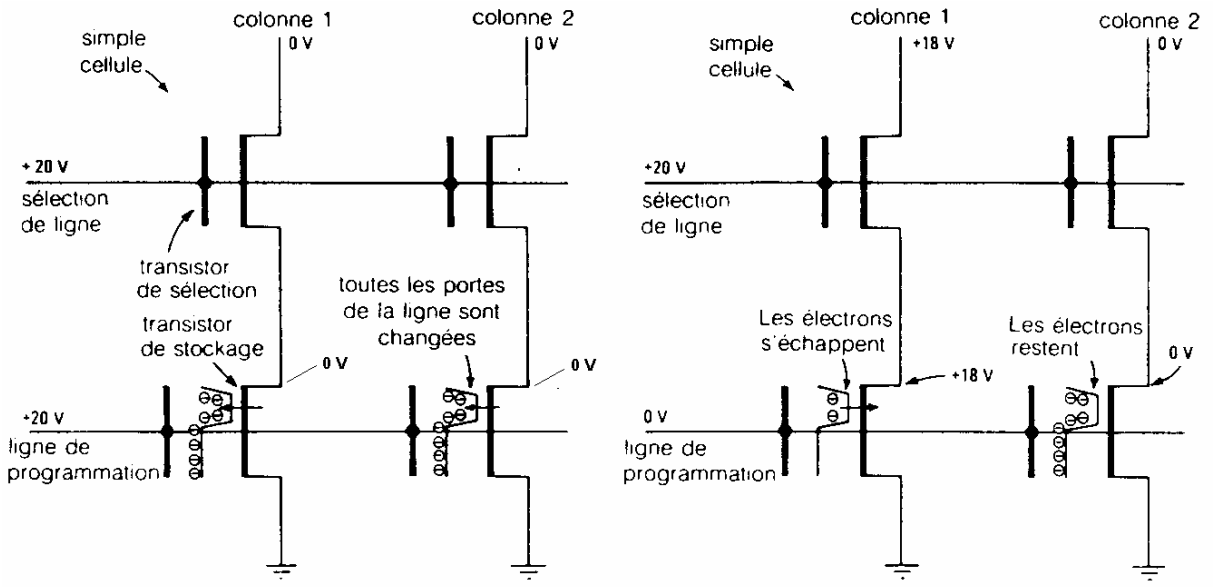
En mode écriture ou programmation (voir figure de gauche ci-dessous), la grille de contrôle est mise à la masse alors que le drain est connecté à une tension positive élevée. Les électrons quittent la grille flottante et passe dans le drain par effet tunnel. La grille se charge alors positivement et le transistor devient passant. Cela correspond à un état logique 0. En mode effacement (voir figure de droite ci-dessous), la grille de contrôle est mise à un potentiel élevé alors que le drain est à la masse. Les électrons passent du drain dans la grille flottante par effet tunnel. La grille se charge alors négativement et le canal dans le transistor disparaît. Cela correspond à un état logique 1.



En fait, à la différence du FAMOS, le transistor à structure FLOTOX n'a pas de grille de sélection, mais une grille de contrôle pour la programmation et l'effacement. On doit donc obligatoirement lui associer un transistor de sélection pour constituer une cellule de stockage comme dans la figure suivante :



On peut alors représenter sur la figure ci-dessous une partie de la matrice de stockage (une ligne, deux colonnes). On retrouve à gauche le mode effacement et à droite le mode programmation.



Cette mémoire peut être effacée octet pat octet et n'est alimentée qu'en 5 V, car les tensions élevées de programmation sont générées par des pompes de charges internes au circuit. La durée de rétention est identique à celle d'une EPROM et la mémoire supporte jusqu'à 100000 cycles de programmation ou d'écriture.

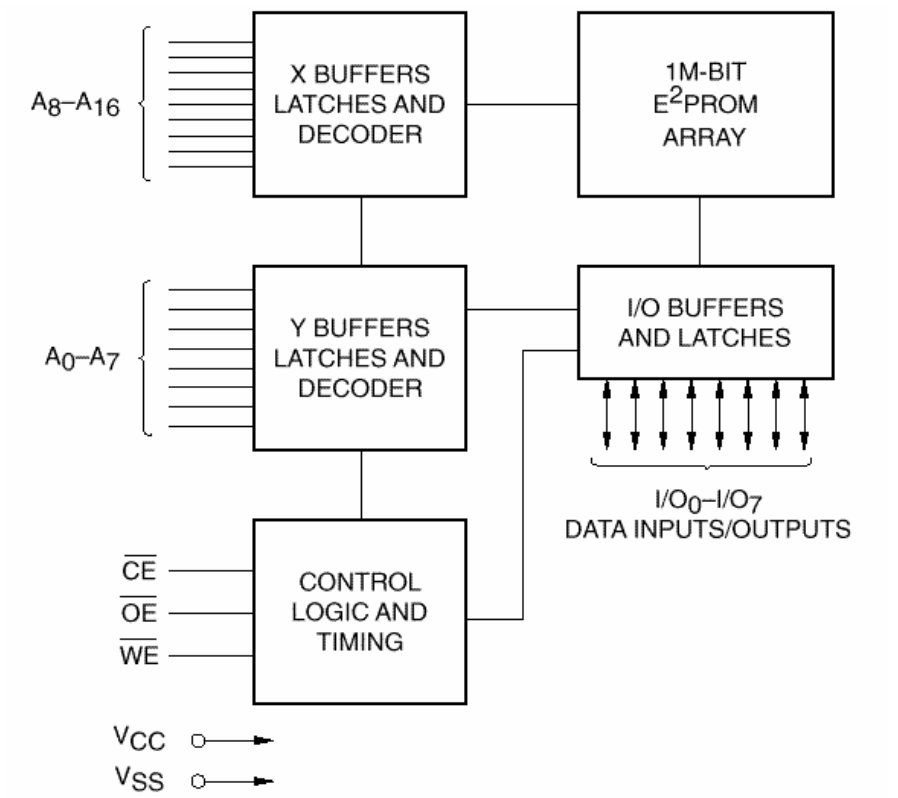
Cette famille de mémoire paraît parfaite puisqu'elle apporte de grandes améliorations à la famille des EPROM et peut être utilisée comme une RAM statique lente. Elle souffre cependant de deux défauts rédhibitoires :

- Le processus de fabrication de la fine couche d'oxyde entre le drain et la grille flottante est très délicat à réaliser.
- La cellule de stockage utilise deux transistors au lieu d'un seul pour l'EPROM, donc la surface de silicium utilisée est plus importante.

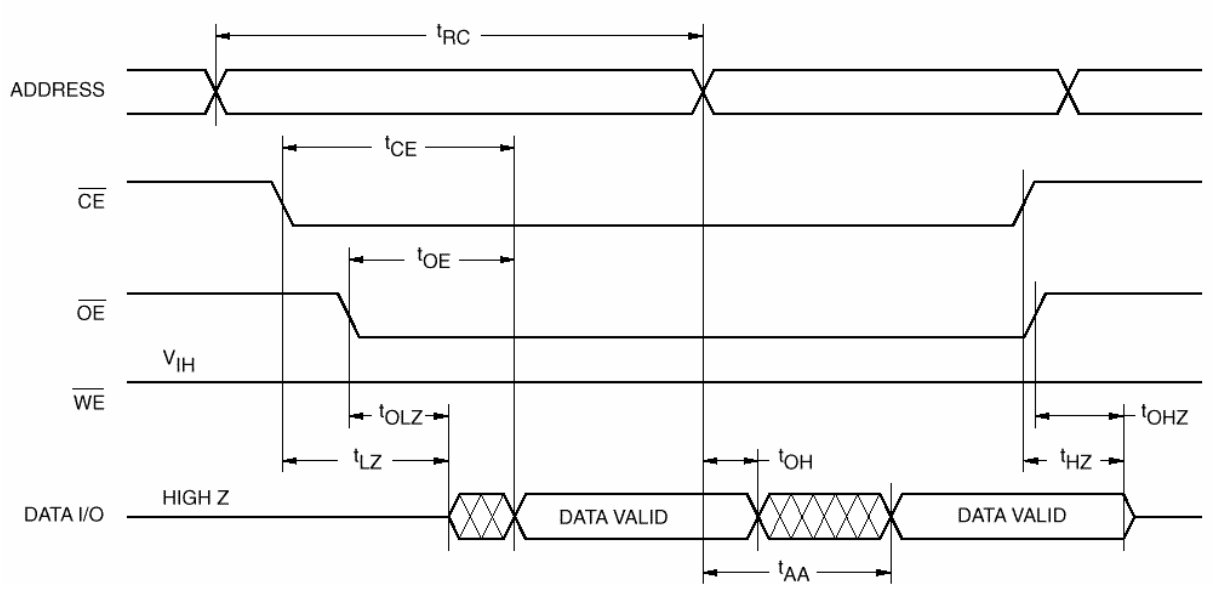
Pour ces deux raisons, le coût des E^{2}PROM est trop élevé et ne peut pas prétendre au remplacement des EPROM. Il ne reste plus que quelques fabricants spécialisés comme Xicor pour avoir un catalogue assez fourni dans ce domaine. Les E^{2}PROM existent soit sous forme parallèle comme les EPROM, soit sous forme série. On retrouve cette dernière famille, généralement de capacité assez réduite, dans tous les appareils où il est nécessaire de sauvegarder une configuration (sauvegarde de la programmation des chaînes d'un magnétoscope par exemple). Sous leur forme parallèle, elles sont remplacées par les mémoires flash qui sont beaucoup moins chères. Elles ont toutefois l'avantage, contrairement aux mémoires flash, d'être programmables octet par octet (il n'y a pas de phase d'effacement), d'être alimentées uniquement en 5 V et d'avoir un fonctionnement assez simple.

4.2.3.2 Exemple : la X28C010 de XICOR

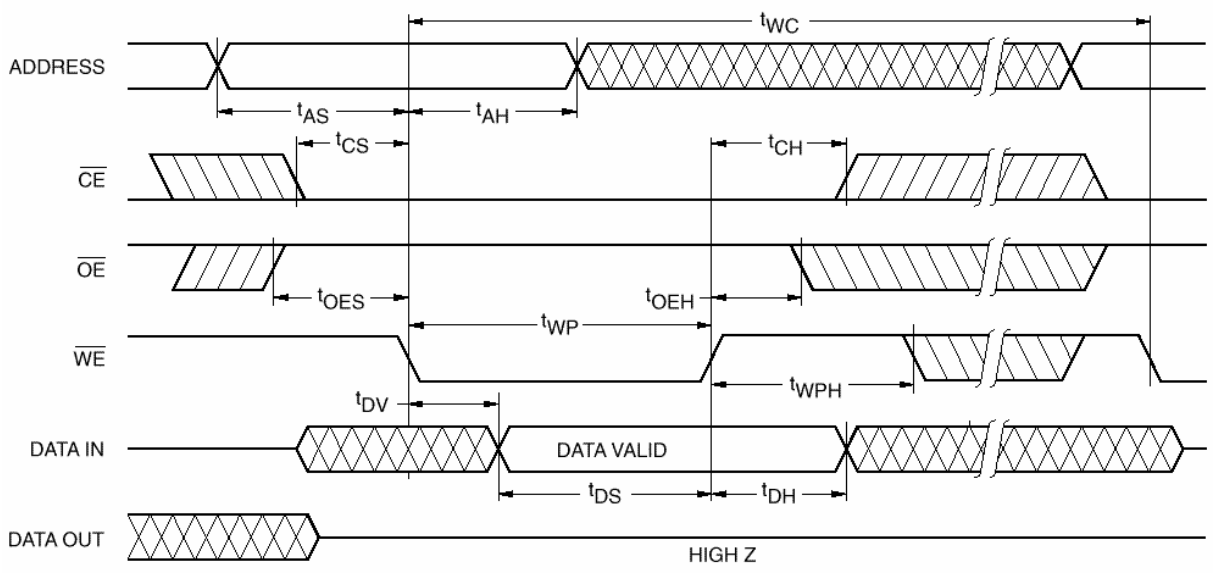
On trouve chez XICOR une gamme d'E^{2}PROM allant de la X2804 (512 x 8) à la X28C010 (128K x 8). La X28C010 est une E^{2}PROM CMOS 128K x 8 alimentée en 5 V dont le temps d'accès est compris entre 120 ns et 250 ns. Elle existe en boîtier DIP 32 broches, PLCC, TSOP ou PGA. Sa consommation est égale à 50 mA en fonctionnement typique et descend à 500 µA en mode stand-by. Le diagramme de blocs de cette mémoire est le suivant :



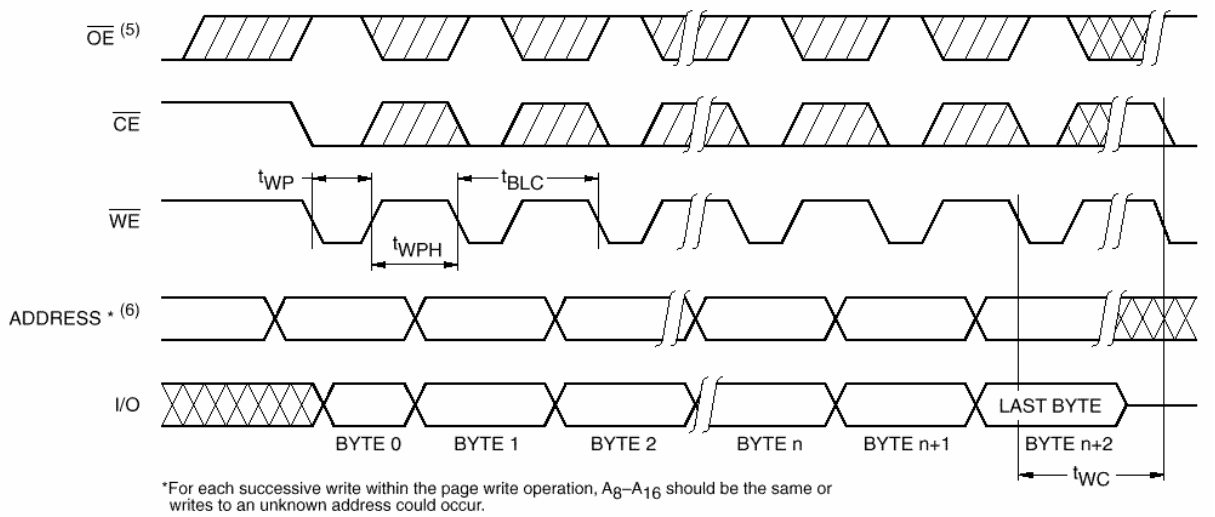
Le chronogramme suivant montre une opération de lecture (le temps t_{AA} correspond au temps d'accès de la mémoire) :



La programmation peut s'effectuer octet par octet. Il faut alors attendre 10 ms maximum entre deux écritures (temps t_{WC}). Le cycle d'écriture par octet est le suivant :



L'écriture peut aussi être réalisée en mode rafale en écrivant non plus octet par octet mais par page allant jusqu'à 256 octets. La durée totale d'écriture d'une page est égale à 10 ms maximum (40 µs max. par octet). La durée maximale nécessaire pour programmer entièrement la mémoire est alors égale à 5 secondes. Le chronogramme suivant montre un cycle d'écriture par page :

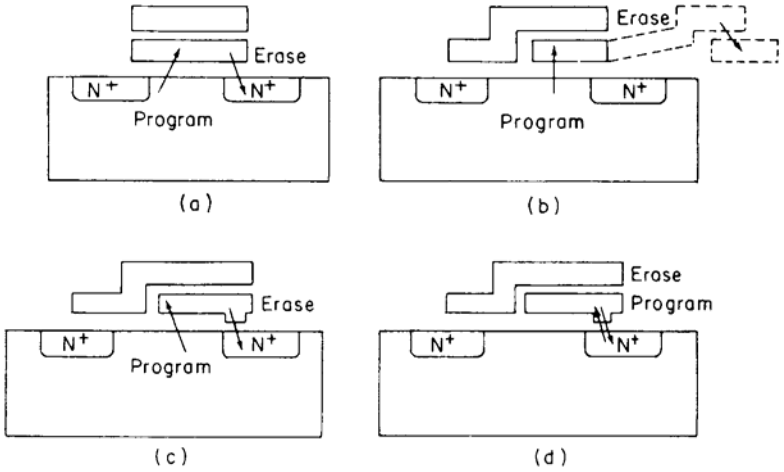


Cette mémoire possède aussi une protection logicielle contre l'écriture. Il suffit pour cela d'écrire AA à l'adresse 5555, puis 55 à l'adresse 2AAA et enfin A0 à l'adresse 5555 en mode d'écriture par page. La mémoire est alors protégée en écriture. Une séquence similaire permet de retirer la protection en écriture.

4.2.4 Flash EEPROM

4.2.4.1 Principe général

A la fin des années 1980, plusieurs sociétés ont mis au point des procédés permettant d'effacer électriquement la cellule EPROM. Le terme flash est venu du fait que ces mémoires n'étaient effaçables qu'en une seule fois dans leur totalité. Quoique ces mémoires soient rapidement devenues effaçables par section, le nom est resté. L'objectif de la cellule flash (qui fait partie de la famille des E^2PROM) est de supprimer le transistor de sélection de la cellule FLOTOX de façon à retrouver le coût d'une EPROM tout en gardant bien entendu la programmation et l'effacement électrique. Il n'existe pas une seule structure de cellule commune à tout les fabricants. La figure ci-dessous montre les principales cellules utilisées :

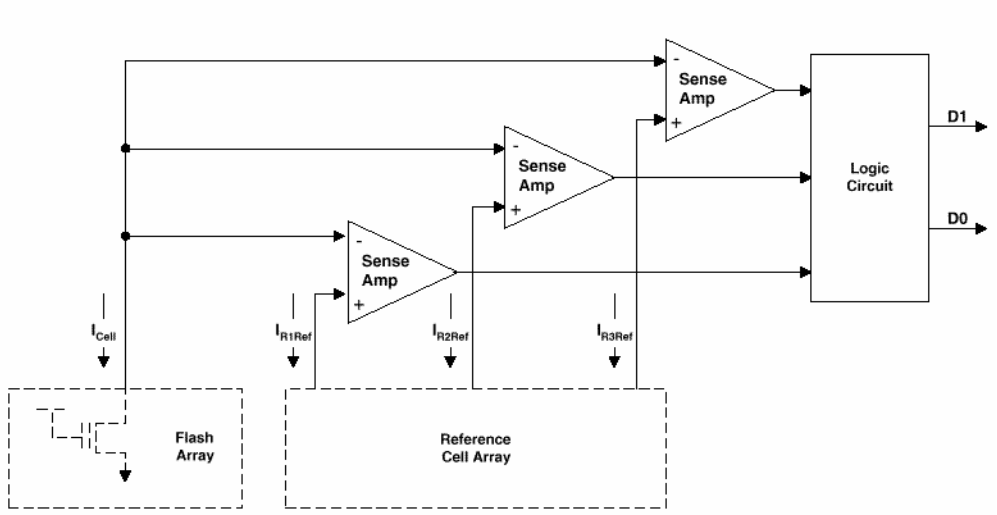


La cellule (a) (INTEL) est directement dérivée du transistor FAMOS, mais avec une plus fine épaisseur de SiO2 entre la grille flottante et le drain. La programmation s'effectue par avalanche et l'effacement par effet tunnel. Les trois autres cellules sont différents essais visant à intégrer le transistor de sélection de la cellule FLOTOX dans le transistor de stockage. La grille est coupée en deux, une moitié sert à la programmation, l'autre moitié sert à la sélection. On appelle cela le concept « split gate ». La cellule (b) (TOSHIBA) utilise l'effet d'avalanche pour la programmation et l'effacement, la cellule (c) (SEEQ) utilise l'effet d'avalanche pour la programmation et l'effet tunnel pour l'effacement. Ces trois cellules nécessitent une tension externe de 12 V pour la programmation alors que la cellule (d), programmée et effacée par effet tunnel, ne nécessite qu'une tension unique égale à 5 V (les autres tensions peuvent être générées en interne dans le circuit par des pompes de charges). Les mémoires flash ont la même durée de rétention de l'information que les EPROM (10 ans à 70 °C) et supporte jusqu'à 100000 cycles d'effacement (ou programmation).

Le tableau suivant indique le nombre de transistors nécessaires à la réalisation d'une cellule de stockage pour différentes familles de mémoires :

Features	Flash	DRAM	EEPROM	SRAM
Cell Components	1 Transistor	1 Transistor + 1 Capacitor	2 Transistor	4 Transistor + 2 Resistor
Cell Area (μm^2) [0.4μ lithography]	2.0	3.2	4.2	22
Chip Area (mm^2) (16-Mbit density)	61	98	107	59 (1-Mbit Density)
Read Speed (ns)	80 (5V) 120 (3V)	60	150	<60

La cellule flash a, comme la cellule FAMOS d'une EPROM, les plus petites dimensions. Comme le prix de la mémoire est directement proportionnel à la surface de silicium occupée, on comprend aisément l'intérêt énorme de la technologie flash pour les constructeurs (et leurs clients). De plus, INTEL a réussi à stocker 2 bits par cellule (voir figure suivante) sans trop modifier le transistor de stockage et parle d'atteindre trois bits par cellules sans modifier (ou fort peu) la surface de la puce.



Cette technologie, la StrataFlash permet dès aujourd'hui de doubler la densité des mémoires flash et de réaliser des densités égales à 64 Mbits comme pour les RAM dynamiques. Les principales applications des mémoires flash sont :

- Le remplacement des PROM dans les applications informatiques (même si le prix d'une flash est encore supérieur à celui d'une PROM).
- Le stockage de données. On utilise des modules flash pour réaliser de petits systèmes de stockage pour les appareils portables (exemple : les appareils photo numériques) ou bien dans les répondeurs téléphoniques pour stocker les messages ou encore dans les fax pour stocker les documents reçus.

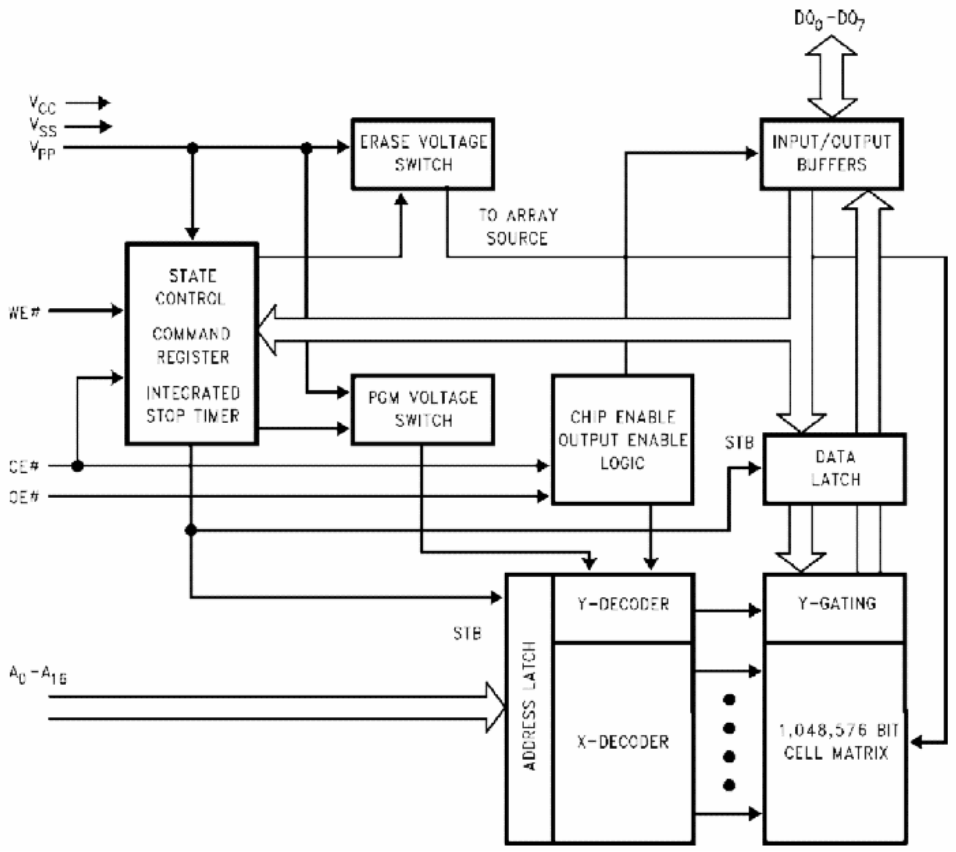
4.2.4.2 Exemple : la 28F010 d'INTEL

On trouve chez INTEL cinq familles de mémoires flash :

- La famille « bulk erase » qui s'efface en intégralité. La gamme va de la 28F010 (128K x 8) à la 28F020 (256K x 8).
- La famille « flash file » qui est découpée en sections de 64 Ko et s'efface par section. La gamme va de la 28F08SA (1M x 8) à la 28F032 (4M x 8 ou 2M x 16).

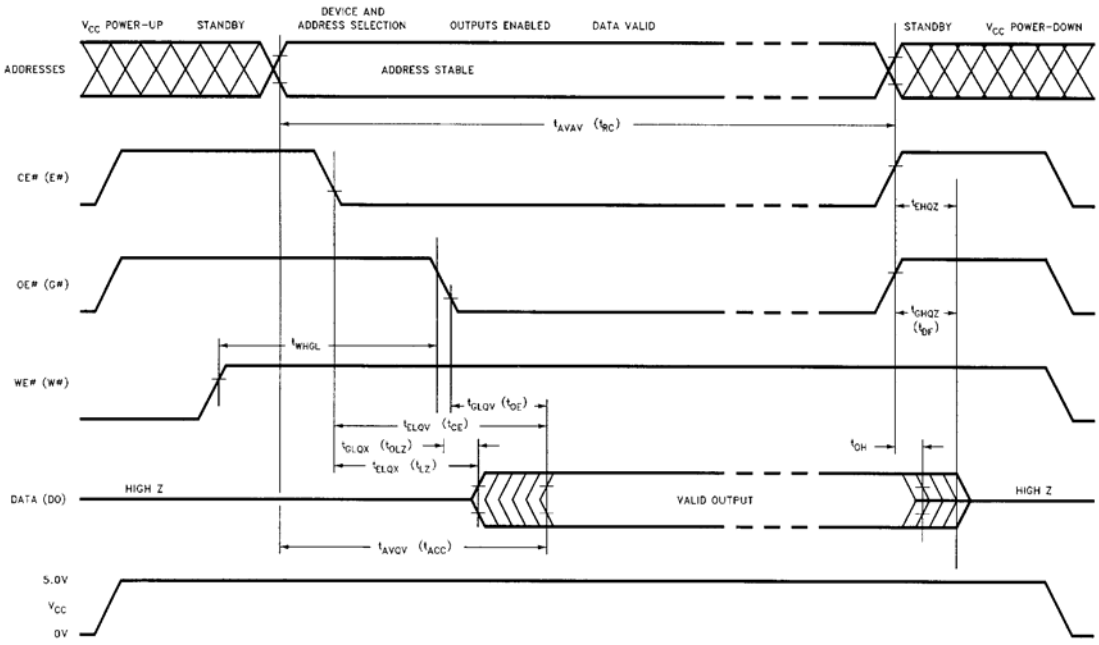
- La famille « boot block » qui possède une section spéciale bloquée en écriture pour contenir le programme de démarrage d'un microprocesseur. La gamme va de la 28F001BX (1M x 8) à la 28F160B3 (1M x 16).
- La famille « strataflash » qui contient 2 bits par cellule de stockage. La gamme va de la 28F320 (4M x 8 ou 2M x 16) à la 28F640 (8M x 8 ou 4M x 16).
- La famille « high performance fast flash » qui prétend se substituer à la DRAM pour l'exécution des programmes d'un microprocesseur. Il n'existe qu'un circuit, le 28F016X (2M x 8 ou 1M x 16).

La 28F010 est une mémoire flash CMOS 128K x 8 dont le temps d'accès est compris entre 65 et 200 ns, alimentée en 5 V mais nécessitant une tension externe de programmation égale à 12 V. Elle existe en boîtier 32 broches DIP, PLCC et TSOP. Sa consommation est égale à 10 mA en fonctionnement typique et descend à 50 µA en mode stand-by. Le diagramme de blocs de cette mémoire, plus complexe que pour les mémoires précédentes se trouve ci-dessous.



Le fonctionnement de cette mémoire est bien plus complexe que les EPROM et les E^2PROM. En effet, bien qu'elle puisse aussi être programmée sur un programmateur d'EPROM (ou d' E^2PROM), cette mémoire est conçue pour être programmée dans l'équipement. Lorsque la

tension appliquée sur la broche V_{PP} est égale à 0, la mémoire est en mode lecture et on a alors le chronogramme suivant où t_{AVQV} est le temps d'accès :

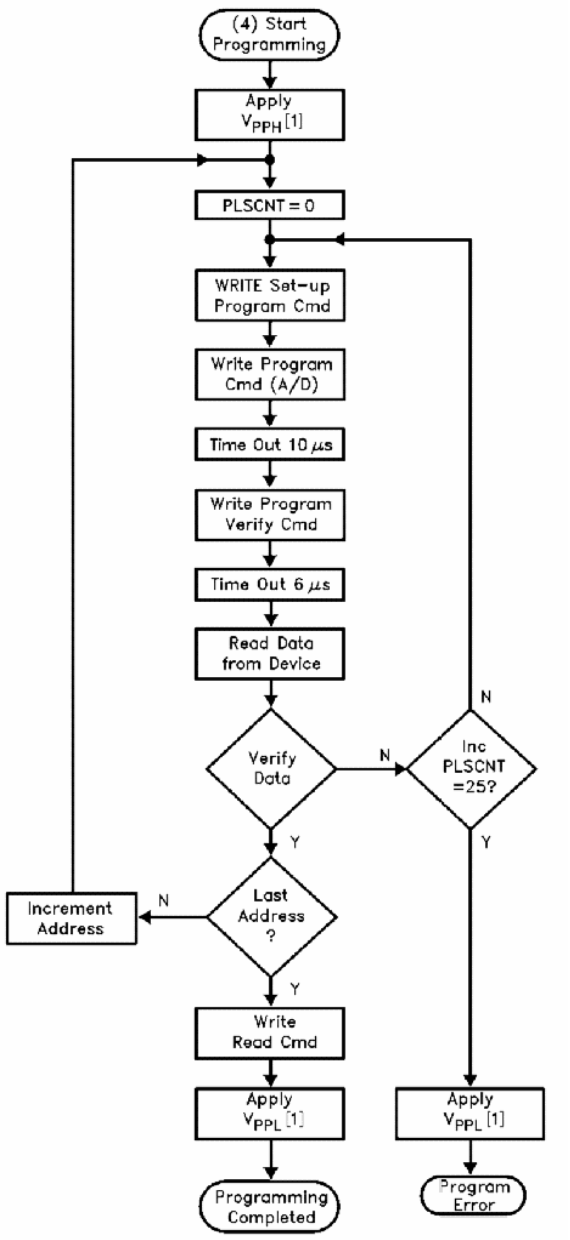


Lorsque la tension appliquée sur la broche V_{PP} est égale à la tension de programmation, on accède à un registre interne dans lequel on peut écrire des codes qui activent alors les différents modes de programmation et d'effacement (voir le tableau suivant).

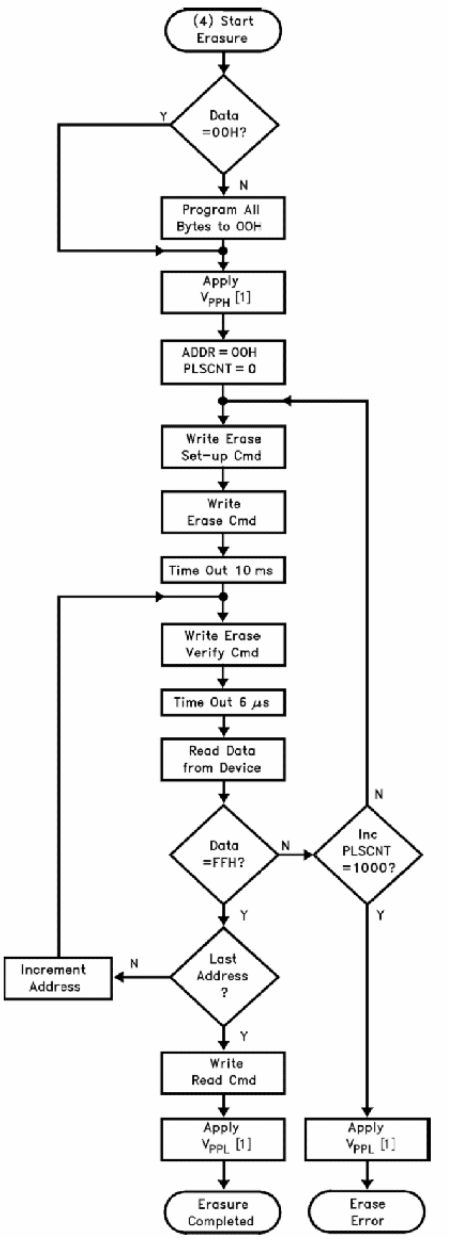
Command	Bus Cycles Req'd	First Bus Cycle			Second Bus Cycle		
		Operation(1)	Address(2)	Data(3)	Operation(1)	Address(2)	Data(3)
Read Memory	1	Write	X	00H			
Read Intelligent Identifier Codes(4)	3	Write	IA	90H	Read	IA	ID
Set-up Erase/Erase(5)	2	Write	X	20H	Write	X	20H
Erase Verify(5)	2	Write	EA	A0H	Read	X	EVD
Set-up Program/Program(6)	2	Write	X	40H	Write	PA	PD
Program Verify(6)	2	Write	X	C0H	Read	X	PVD
Reset(7)	2	Write	X	FFH	Write	X	FFH

On voit qu'il n'est pas nécessaire de prévoir une commutation de la haute tension de programmation sur la broche V_{PP} pour passer du mode lecture au mode programmation/effacement, puisqu'on peut lire la mémoire (code 00) lorsque V_{PP} est actif (c'est d'ailleurs le mode par défaut). Cela permet de simplifier le circuit imprimé où se trouve placé la mémoire flash.

Les modes de programmation et d'effacement sont plutôt complexes à mettre en œuvre. Pour les utiliser, il faut se servir d'algorithmes rapides dont les organigrammes sont les suivants :



Algorithme « quick pulse programming »



Algorithme « quick erase »

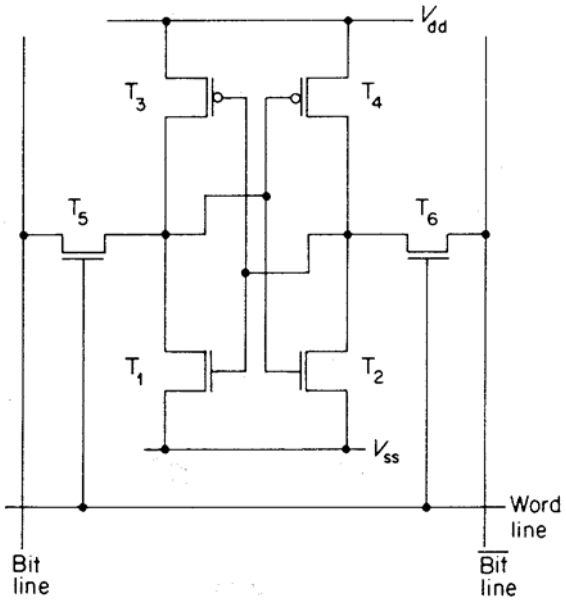
Ces algorithmes doivent être connus du programmateur d'EPROM. En cas de programmation dans l'équipement, c'est le rôle du microprocesseur (ou du microcontrôleur) qui pilote ces mémoires de générer les séquences nécessaires à l'aide d'un programme approprié. Cette mémoire flash incorpore, comme une EPROM, une signature électronique lui permettant d'être reconnue automatiquement par un programmateur.

4.3 La famille des RAM

4.3.1 RAM statique

4.3.1.1 Principe général

Il existe deux familles de cellules de stockage qui sont utilisées dans toutes les mémoires de type RAM : la cellule RAM dynamique qui doit être rafraîchie régulièrement pour garder la donnée en mémoire et la cellule RAM statique qui ne nécessite pas de rafraîchissement. Cette dernière est constituée de 6 transistors (voir figure ci-dessous). Les transistors T1 à T4 forment les deux inverseurs CMOS qui sont rebouclés pour constituer la bascule qui va mémoriser le bit. Les transistors T5 et T6 permettent de sélectionner la cellule de stockage afin de lire ou d'écrire un bit.



Cellule de stockage SRAM CMOS

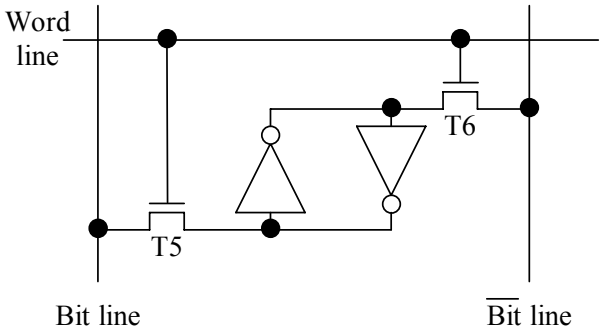
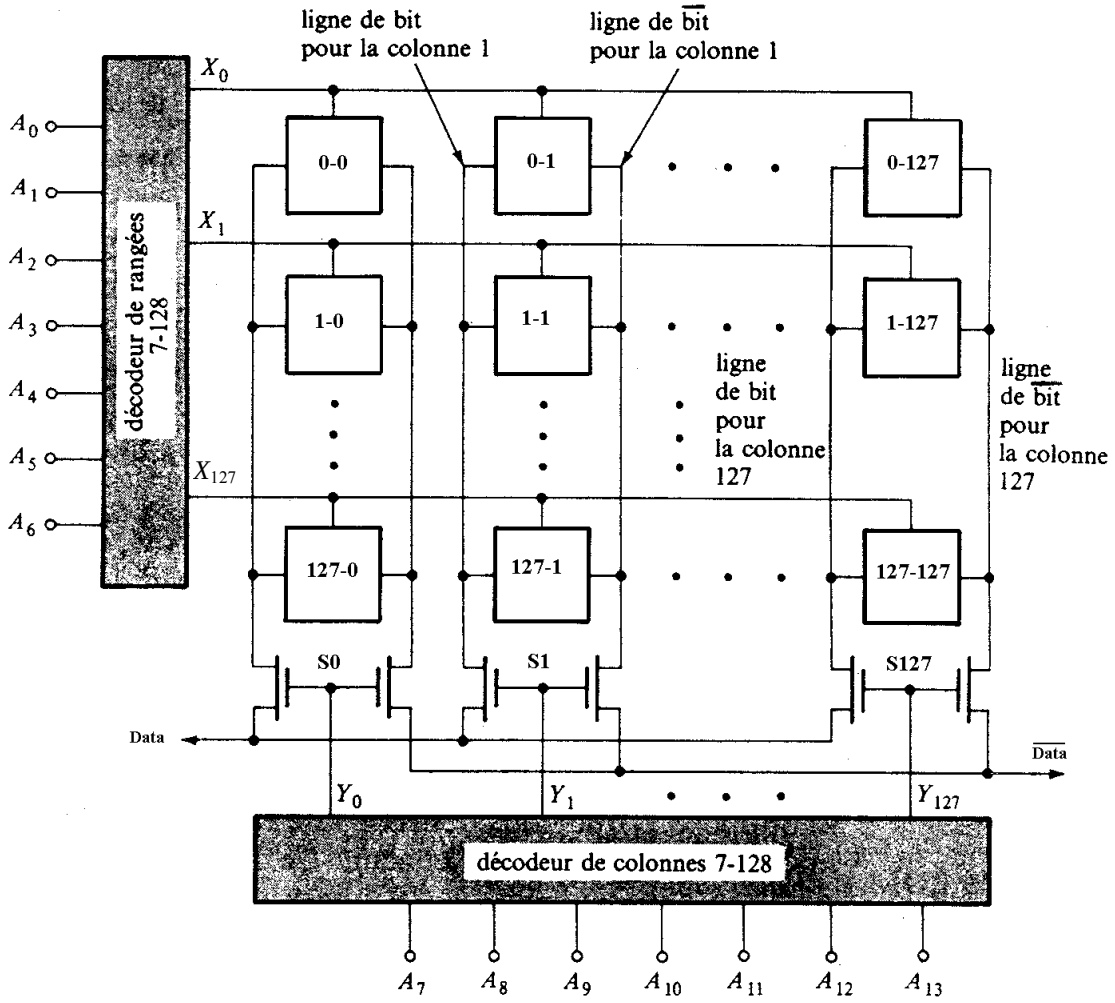


Schéma équivalent

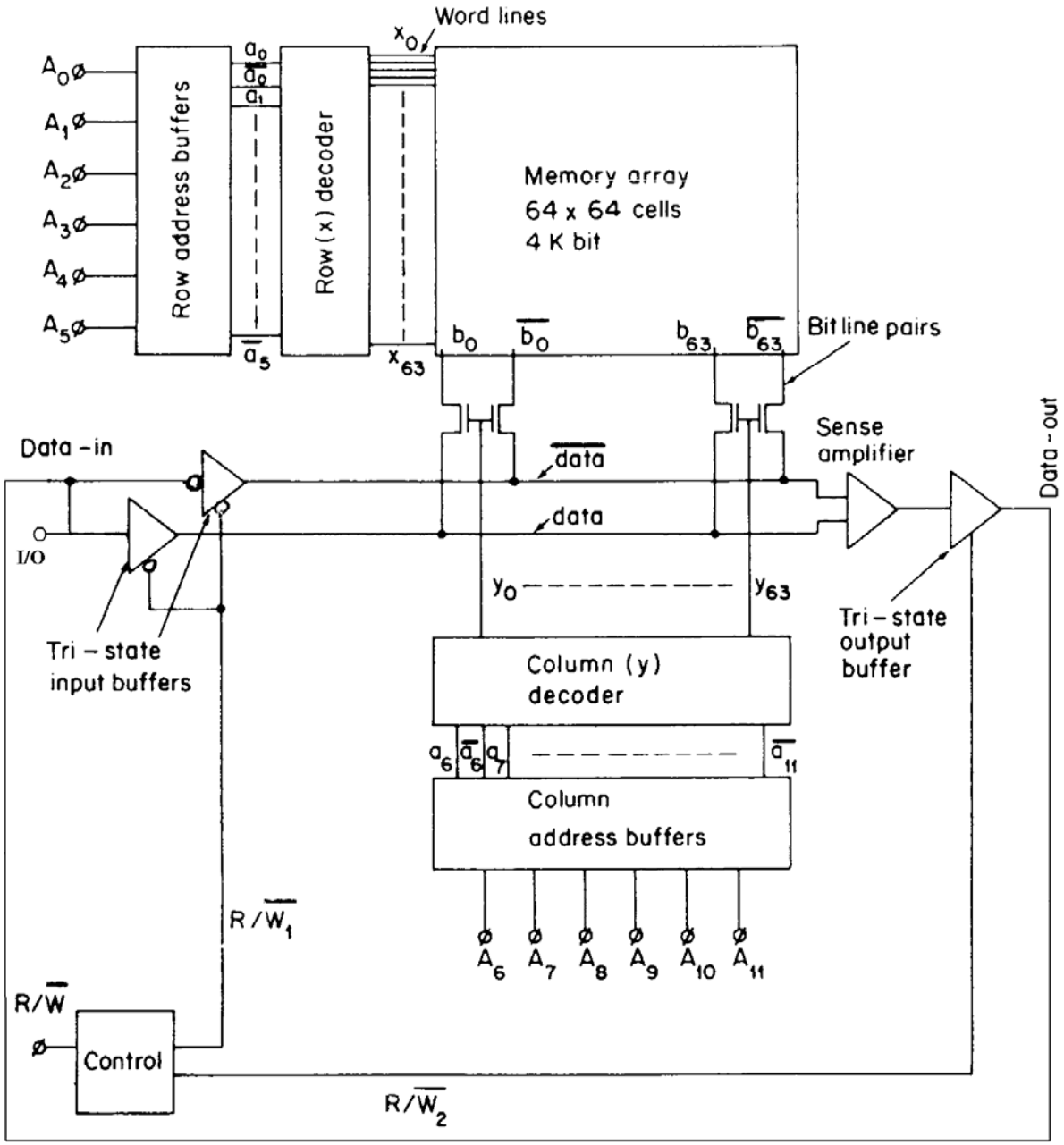
En se reportant au §5.1.3 sur la structure générale d'une mémoire, on voit que la rangée X_n s'appelle « word line » et la colonne Y_n s'appelle « bit line ». Les lignes bit et $\overline{bit}$ sont connectées à toutes les cellules d'une même colonne et la ligne word est reliée à toutes les cellules d'une même rangée. La figure suivante vous montre la structure simplifiée d'une RAM statique 16K x 1.



La paire de transistors MOS Sn permet de sélectionner la colonne n. La sélection d'une cellule se réalise donc en rendant passant cette paire de transistors ainsi que les deux transistors de sélection de rangée T5 et T6. Pour effectuer une lecture de la cellule sélectionnée, il suffit alors de lire le bit de donnée disponible sur les lignes bit et $\overline{bit}$. Quand on souhaite écrire un bit dans la cellule sélectionnée, il y a deux possibilités :

- Soit le bit de donnée à écrire est identique à celui mémorisé dans la bascule et rien ne change.
- Soit le bit de donnée à écrire est différent de celui mémorisé et il faut que la bascule change d'état. Supposons que bit = 1 ($\overline{bit}$ = 0). Cela implique que T3 et T2 sont passants alors que T1 et T4 sont ouverts. On veut écrire bit = 0 ($\overline{bit}$ = 1). Le niveau 0 est appliqué sur la grille de T2, T4 et T4 devient passant (et T2 ouvert). La grille de T1, T3 passe donc à 1 et T1 devient passant (et T3 ouvert). Lorsque l'on désélectionne la cellule, l'état est stable et reste mémorisé.

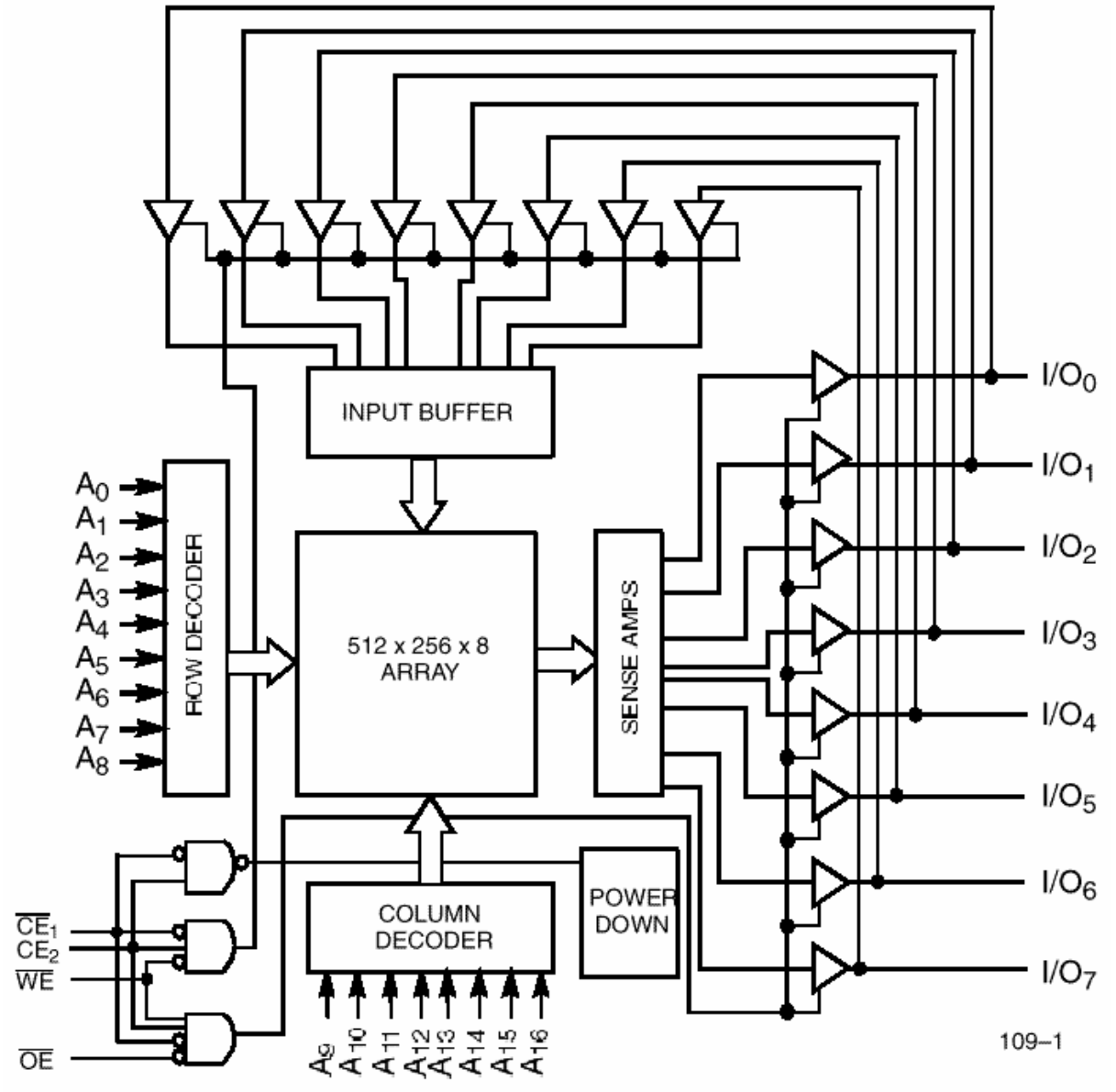
La figure suivante montre la structure d'une SRAM y compris les buffers d'écriture et les amplificateurs de lecture. La broche I/O est la broche d'entrée/sortie de la donnée.



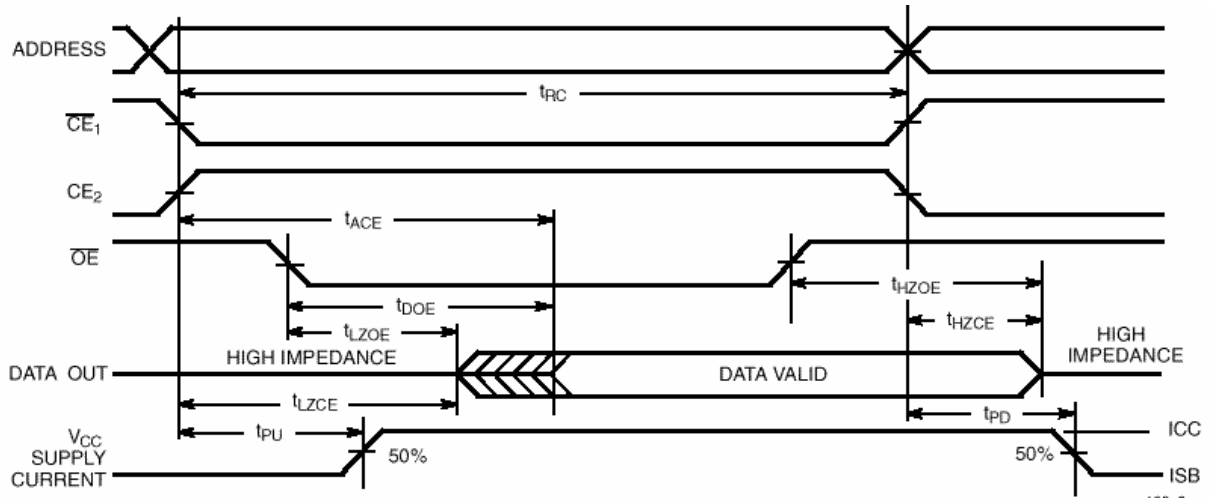
Le temps d'accès d'une mémoire SRAM est égal au temps de propagation dans les décodeurs d'adresses plus le temps de commutation de la bascule dans la cellule de stockage. C'est la plus rapide des mémoires avec des temps d'accès de l'ordre de 10 nanosecondes. Hélas, comme le montre le tableau du §5.2.4.1, c'est aussi la mémoire qui nécessite la plus grande surface de silicium par bit stocké. Elle est donc chère.

4.3.1.2 Exemple : la CY7C1009 de CYPRESS

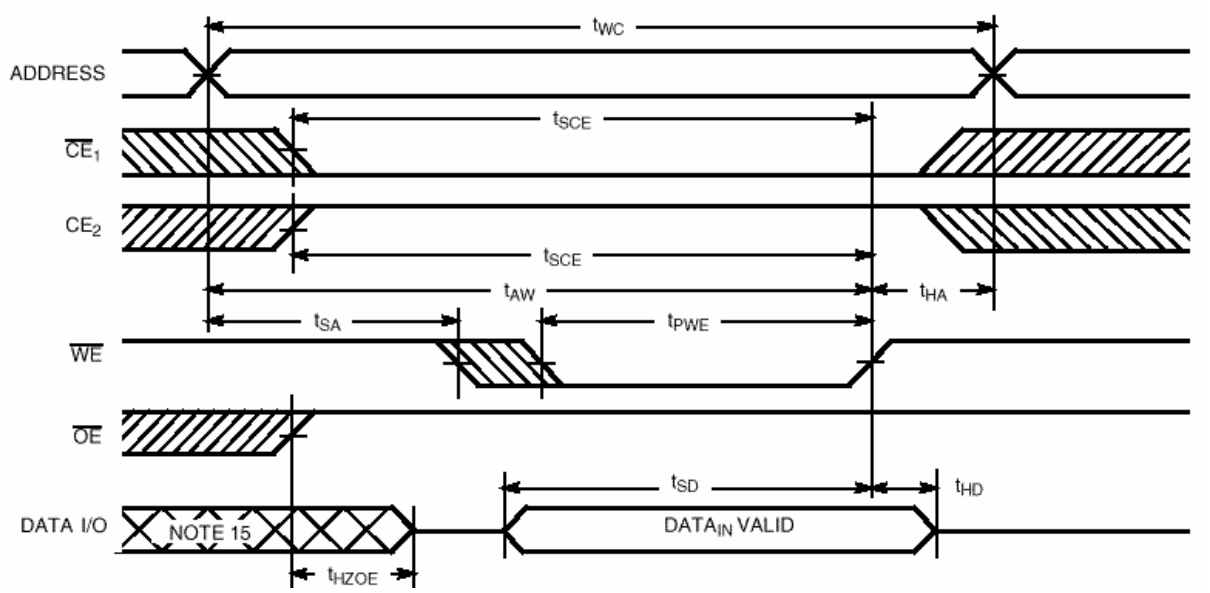
On trouve chez CYPRESS une gamme de SRAM allant de la CY7C122 (256 x 4) à la CY7C1049 (512K x 8). La CY7C1009 est une SRAM CMOS 128K x 8 alimentée en 5 V dont le temps d'accès est compris entre 10 et 35 ns. Elle existe en boîtier DIP, SOJ ou TSOP 32 broches. Sa consommation maximale est égale à 200 mA et descend à 10 mA en mode stand-by. Le diagramme de blocs de cette mémoire est le suivant :



De toutes les mémoires avec lecture/écriture, les SRAM sont les plus simples à utiliser. Le chronogramme suivant montre une opération de lecture (le temps t_{ACE} correspond au temps d'accès de la mémoire) :



Le chronogramme suivant montre un cycle d'écriture (le temps t_{WC} correspond à la durée d'un cycle complet d'écriture) :

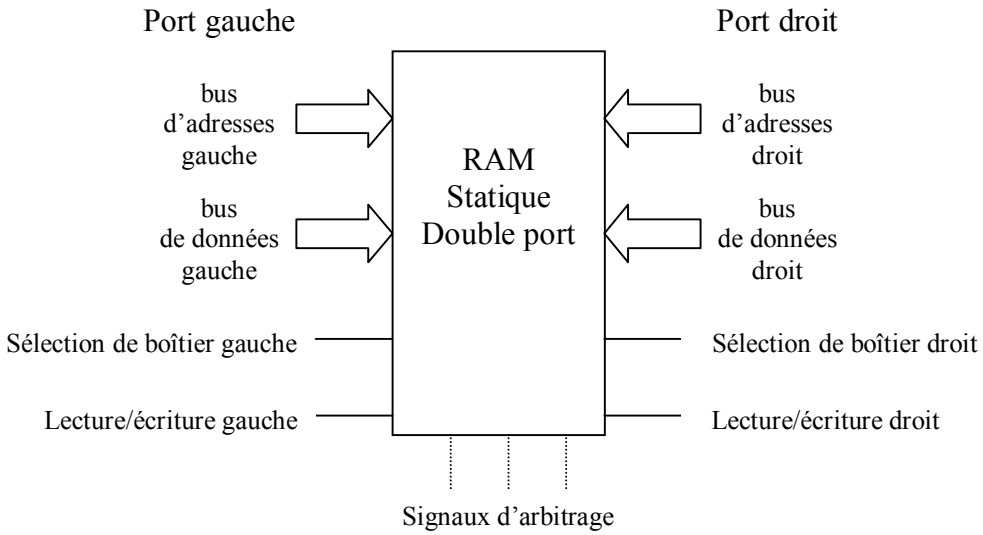


La plupart des SRAM sont bâties selon ce modèle où les signaux sont asynchrones les uns par rapport aux autres. Il existe aussi les SSRAM (SRAM synchrones) où tous les signaux sont pris en compte ou fournis sur le front actif d'une horloge. Ces mémoires synchrones sont généralement utilisées dans les systèmes les plus rapides. En effet, la synchronisation, si elle ne rend pas la mémoire plus rapide, permet d'atteindre beaucoup plus facilement la fréquence maximale de fonctionnement de la mémoire.

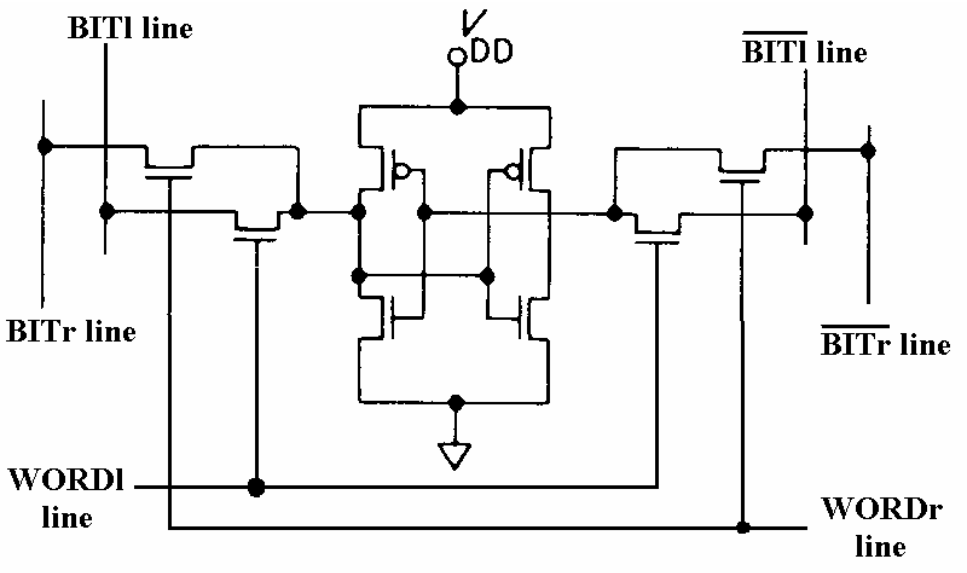
4.3.2 RAM statique double port

4.3.2.1 Principe général

Une SRAM double port a l'aspect extérieur suivant :



Deux ports indépendants, gauche et droit, permettent l'accès à la mémoire. Chaque côté peut être sélectionné indépendamment et utilisé de manière autonome. Les éventuels signaux d'arbitrage servent à gérer le cas où les deux côtés essaient d'écrire en même temps dans la même case mémoire ou bien quand un côté essaie de lire une case alors que l'autre côté est en train d'écrire dedans. Pour réaliser ce type d'architecture, la cellule RAM statique doit être modifiée de la manière suivante :

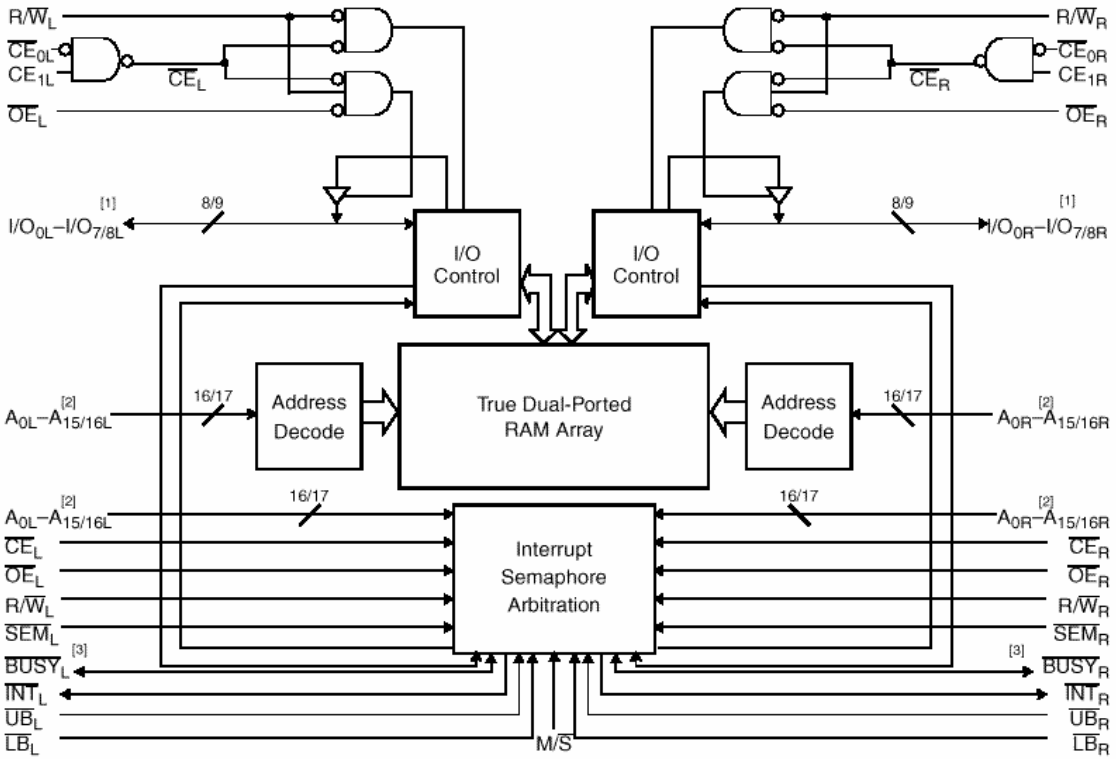


On a maintenant deux lignes de sélection de rangée, « WORDl line » (l : left) pour le port gauche et « WORDr line » (r : right) pour le port droit, et deux paires de lignes de sélection de colonne, « BITl line, $\overline{\text{BITl}}$ line » pour le port gauche et « BITr line, $\overline{\text{BITr}}$ line » pour le port droit. Chaque côté possèdant son propre décodeur d'adresses, on accéde indépendamment à la cellule de stockage par chaque port.

Cette mémoire de type SRAM est rapide (temps d'accès ~ 10 ns). Elle est utilisée soit pour faire communiquer deux systèmes ayant des vitesses de fonctionnement différentes, soit pour faire communiquer plusieurs processeurs entre eux dans un système multiprocesseur ou encore dans certaines applications vidéo.

4.3.2.2 Exemple : la CY7C009 de CYPRESS

On trouve chez CYPRESS une gamme de SRAM double port allant de la CY7C130 (1K x 8) à la CY7C019 (128K x 9). La CY7C009 est une SRAM double port CMOS 128K x 8 alimentée en 5 V dont le temps d'accès est compris entre 12 et 20 ns. Elle existe en boîtier QFP 100 broches. Sa consommation typique est égale à 180 mA et descend à 50 µA en mode stand-by. Elle possède deux ports indépendants plus des signaux d'arbitrage. Le diagramme de blocs de cette mémoire est le suivant :



Le nombre de broches (100) est plutôt élevé. Le tableau suivant donne la signification succincte des différents signaux :

- Lignes 1 à 5 : bus d'adresses, de données et de contrôle des deux ports.
- Lignes 6 et 7 : signaux d'interruptions et sémaphores servant à la communication port à port.
- Ligne 8 : signal d'arbitrage indiquant que la cellule de stockage est en cours d'utilisation.
- Ligne 9 : signal maître/esclave utilisé pour l'expansion du bus de données.

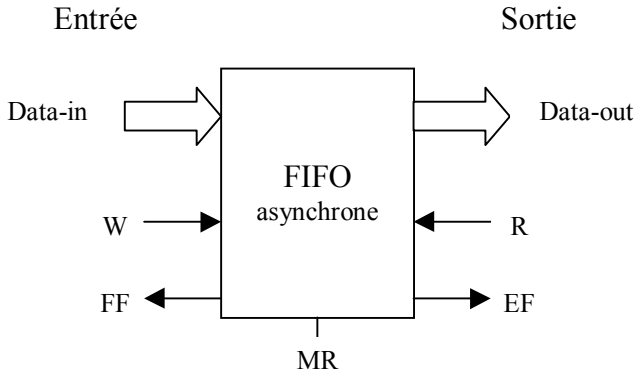
Ligne	Left Port	Right Port	Description
1	$\overline{CE}_{0L}$, CE_{1L}	$\overline{CE}_{R}$, CE_{1R}	Chip Enable ($\overline{CE}$ is Low when $\overline{CE}_0 \le V_{IL}$ and $CE_1 \ge V_{IH}$)
2	$R/\overline{W}_L$	$R/\overline{W}_R$	Read/Write Enable
3	$\overline{OE}_L$	$\overline{OE}_R$	Output Enable
4	A_{0L}–A_{16L}	A_{0R}–A_{16R}	Address (A_0–A_{15} for 64K devices and A_0–A_{16} for 128K devices)
5	I/O_{0L}–I/O_{8L}	I/O_{0R}–I/O_{8R}	Data Bus Input/Output (I/O_0–I/O_7 for x8 devices and I/O_0–I/O_8 for x9)
6	$\overline{SEM}_L$	$\overline{SEM}_R$	Semaphore Enable
7	$\overline{INT}_L$	$\overline{INT}_R$	Interrupt Flag
8	$\overline{BUSY}_L$	$\overline{BUSY}_R$	Busy Flag
9	$M/\overline{S}$		Master or Slave Select
10	V_{CC}		Power
11	GND		Ground
12	NC		No Connect

Les chronogrammes de lecture et d'écriture sont similaires à ceux d'une SRAM simple port. Les signaux de communication port à port, d'arbitrage et d'expansion compliquent considérablement l'utilisation de la mémoire.

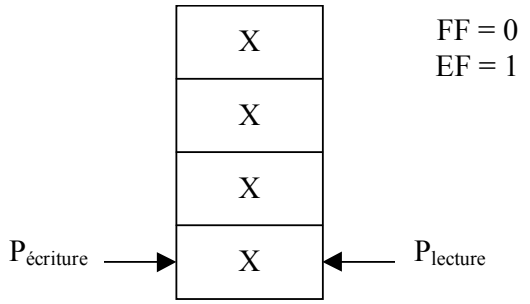
4.3.3 FIFO

4.3.3.1 Principe général

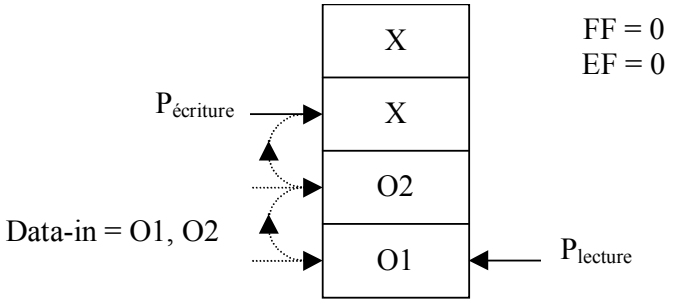
Le schéma suivant montre l'aspect extérieur d'une pile FIFO (First In, First Out) :



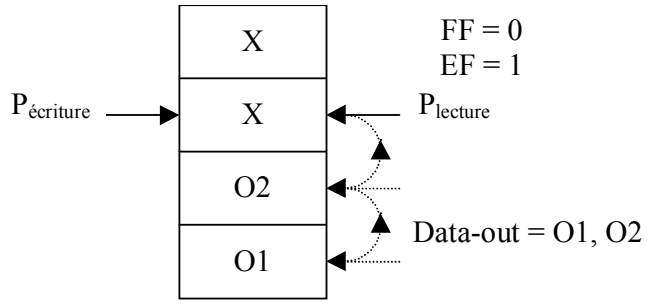
Il y a dans un FIFO deux bus de données, un servant pour l'écriture et l'autre servant pour la lecture. Un pointeur d'adresses écriture et un pointeur d'adresses lecture sont gérés en interne dans le circuit. Lorsque le signal W (Write) devient actif, la donnée data-in est copiée dans le FIFO et le pointeur d'écriture est incrémenté de 1. Lorsque le signal R (Read) devient actif, la donnée la plus ancienne mémorisée dans le FIFO est copiée sur data-out et le pointeur de lecture est incrémenté de 1. Le signal FF (FIFO Full) indique quand le FIFO est plein et que l'on ne peut plus y écrire de données, alors que le signal EF (Empty FIFO) indique quand le FIFO est vide et que l'on ne peut plus y lire de données. Le signal MR (Master Reset) initialise le circuit. Pour comprendre le fonctionnement de ce circuit, prenons l'exemple d'un FIFO 4 x 8. Le schéma suivant montre l'état de la mémoire, des signaux EF et FF et la position des pointeurs de lecture et d'écriture à l'initialisation (l'état X est non significatif). Les adresses sont croissantes vers le haut de la pile.



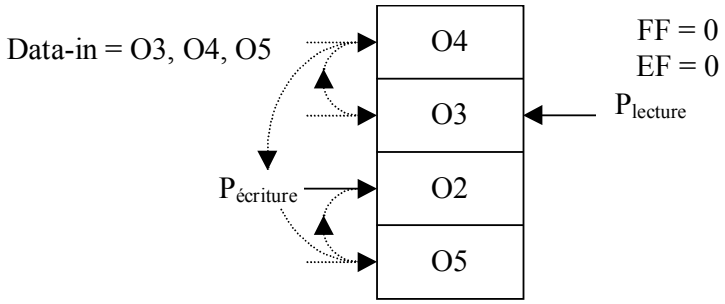
Le FIFO est vide. On voit que les pointeurs sont placés sur la première case mémoire disponible pour une lecture ou pour une écriture (même si la lecture ou l'écriture est impossible). Après écriture de deux octets O1 et O2, le pointeur d'écriture est incrémenté de 2 et les deux octets sont mis en mémoire :



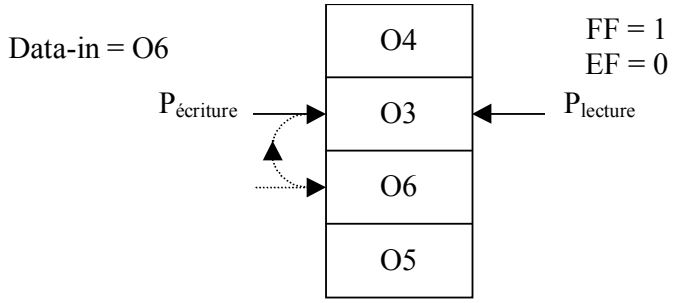
Comme le FIFO n'est plus vide, on peut lire les deux octets O1 puis O2. On comprend le nom de cette mémoire (premier entré – premier sorti) puisque le premier octet entré O1 est bien le premier octet lu. Le pointeur de lecture est incrémenté de 2 :



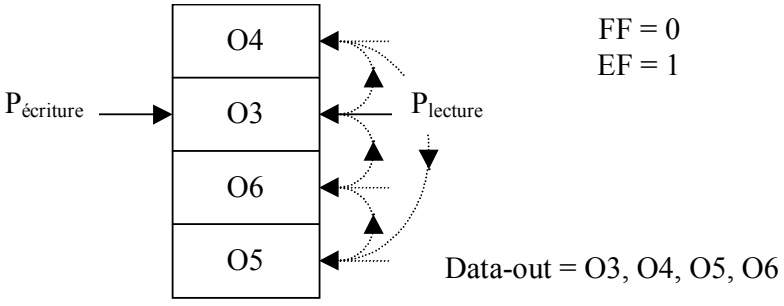
Le FIFO est à nouveau vide car le pointeur de lecture à rattrapé le pointeur d'écriture. Nous allons maintenant écrire 3 octets O3, O4 et O5 dans la mémoire. Cela est possible car les pointeurs de lecture et d'écriture repasse à 0 après avoir atteint leur valeur maximale.



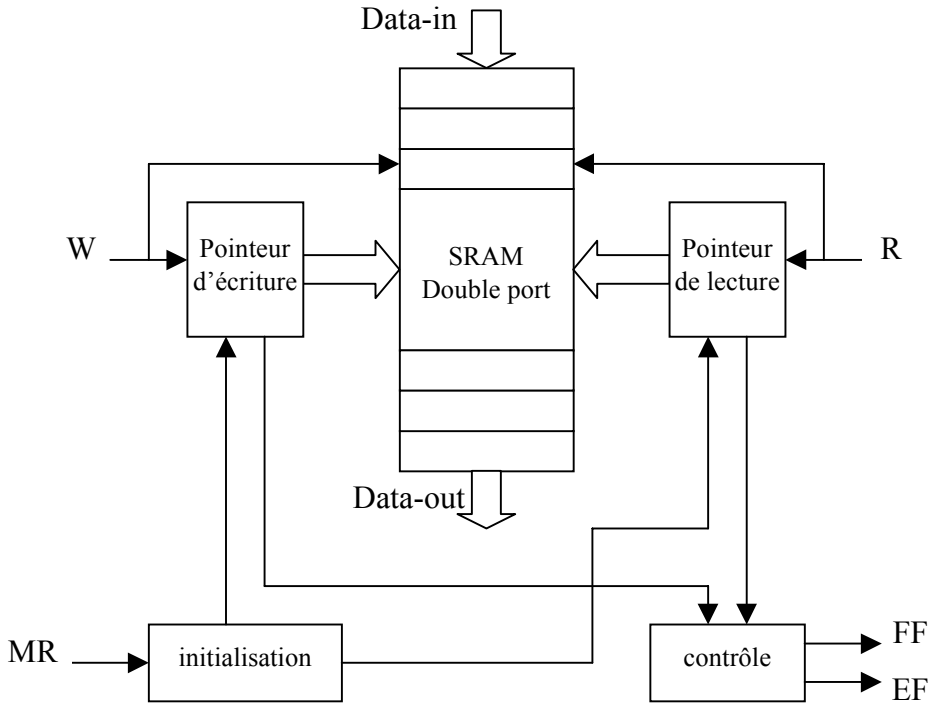
Avec une écriture de plus (O6), le FIFO est plein car le pointeur d'écriture a rattrapé le pointeur de lecture.



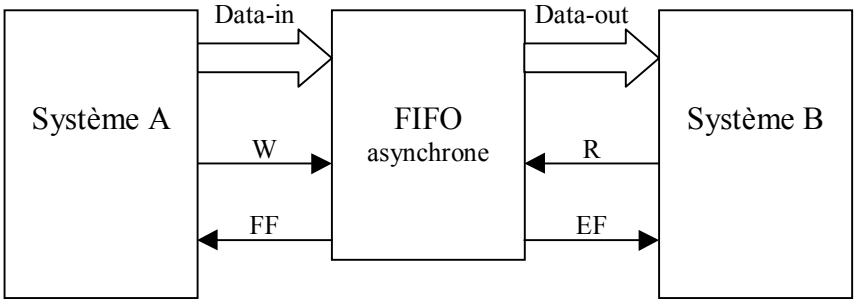
On peut maintenant effectuer 4 lectures et vider le FIFO



Un FIFO est généralement construit autour d'une SRAM double port, ce qui simplifie grandement la structure interne de ce genre de mémoire. Par construction, il n'y a jamais d'écriture et de lecture simultanées dans une même case mémoire puisque une case ne peut être lue qu'après avoir été écrite. L'arbitrage traditionnel dans une RAM double port n'a donc pas lieu d'être. L'écriture et la lecture dans le FIFO peuvent donc être simultanés et asynchrones. Le schéma suivant montre l'architecture interne simplifiée du circuit :



Un FIFO basé sur de la SRAM a un temps d'accès de l'ordre de 10 ns. Il est utilisé partout où il est nécessaire de faire communiquer deux systèmes électroniques ayant des fréquences de fonctionnement différentes. Le schéma suivant montre un exemple classique de communication entre deux systèmes asynchrones (un ordinateur et une imprimante par exemple) :



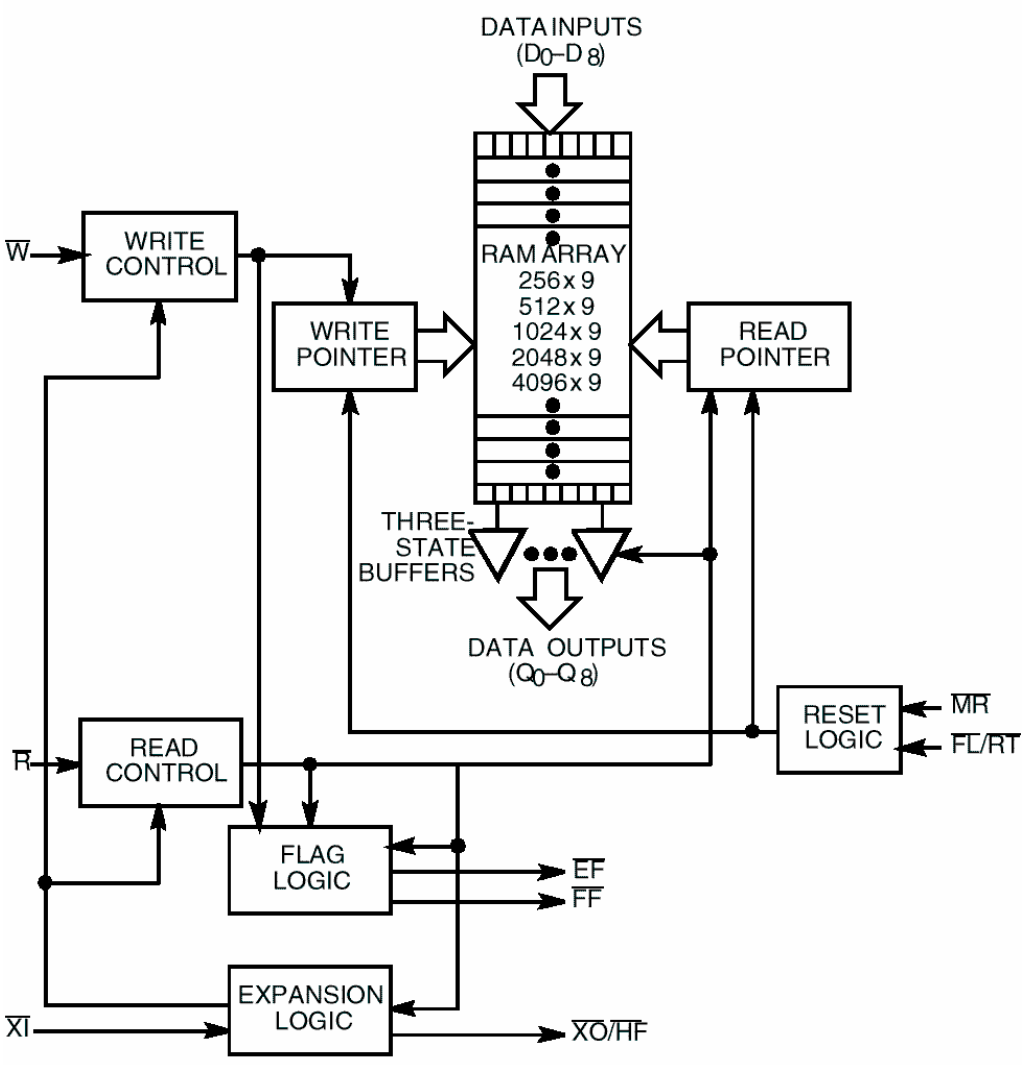
Comme pour les SRAM, il existe aussi des FIFO synchrones où les échanges sont cadencés par une horloge indépendante de chaque côté.

4.3.3.2 Exemple : le CY7C423 de CYPRESS

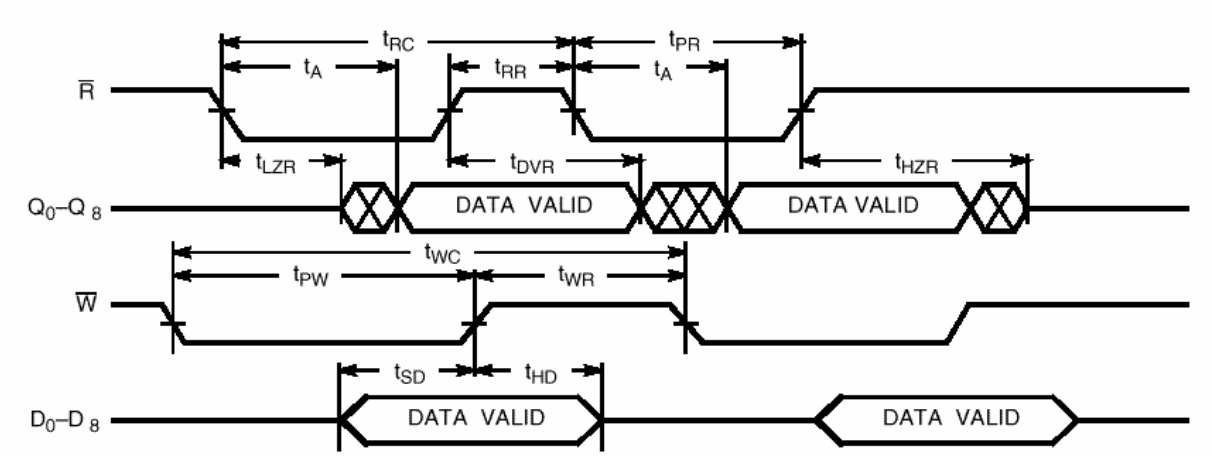
On trouve chez CYPRESS une gamme de FIFO très étendue :

- FIFO synchrones : du CY7C4421 (64 X 9) au CY7C4291 (128K x 9).
- FIFO asynchrones : du CY7C401 (64 X 4) au CY7C466A (64K x 9).

Le CY7C425 est un FIFO asynchrone (construit avec de la SRAM double port) CMOS 1K x 9 alimenté en 5 V dont le temps d'accès est compris entre 10 et 65 ns. Il existe en boîtier DIP, PLCC et QFP 32 broches. Sa consommation typique est égale à 35 mA et descend à 10 mA en mode stand-by. Le diagramme de blocs de cette mémoire est le suivant :



Elle comporte, outre les signaux définis au paragraphe précédent, un signal de demi-remplissage HF (Half FIFO), un signal de retransmission RT (ReTransmit) et deux signaux d'expansion XI et XO (Expansion In et Out). Le chronogramme suivant vous montre le fonctionnement normal du FIFO avec des écritures et des lectures asynchrones simultanées. t_A est le temps d'accès en lecture et t_{WC} le temps de cycle d'écriture.

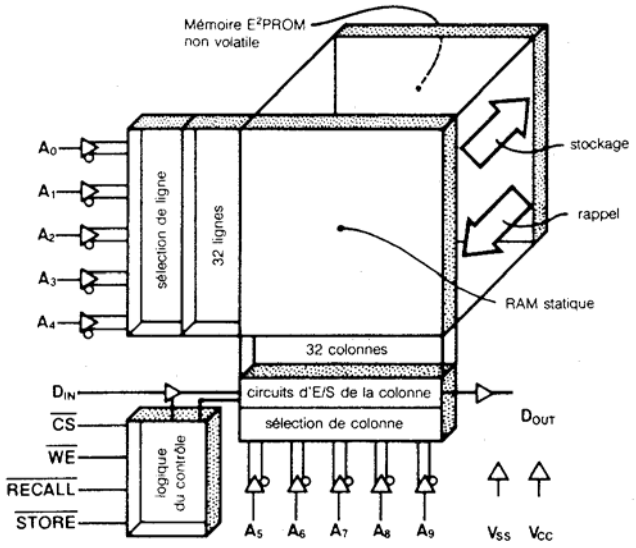


Comme pour les mémoires traditionnelles, grâce aux signaux $\overline{XI}$ et $\overline{XO}$, on peut réaliser une expansion en largeur du bus de données ainsi qu'une expansion en capacité.

4.3.4 RAM non-volatile

4.3.4.1 Principe général

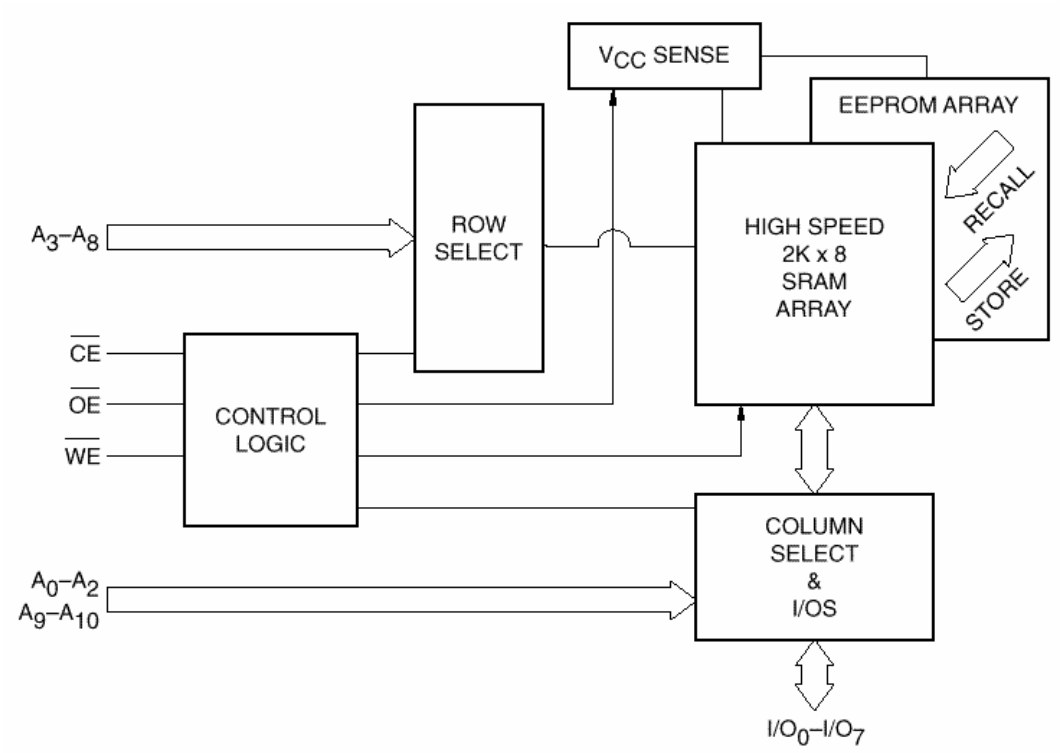
Ce type de mémoire combine un plan mémoire de travail SRAM et un plan mémoire de stockage E^2PROM. Le diagramme suivant montre la structure interne d'une NOVRAM (Non Volatile RAM) série :



On retrouve les signaux classiques d'une SRAM avec en plus $\overline{\text{STORE}}$ qui sert à stocker le contenu de la SRAM dans l'E^2PROM et $\overline{\text{RECALL}}$ qui fait l'opération inverse. L'opération de stockage est suffisamment rapide pour pouvoir être effectuée automatiquement en cas de coupure d'alimentation. Ce type de mémoire bénéficie de la vitesse d'écriture et de lecture de la SRAM et de l'aspect non volatile de l'E^2PROM. Son principal inconvénient est son prix. On utilise ce type de mémoire pour sauvegarder une configuration matérielle (configuration d'un port ou d'une carte PC par exemple).

4.3.4.2 Exemple : la X20C17 de XICOR

XICOR commercialise des NOVRAM série 256 bits et une famille de NOVRAM parallèle allant de la X22C10 (64 x4) à la X20C16 (2K x 8). La X20C17 est une NVRAM CMOS (2K x 8) qui détecte la chute de la tension d'alimentation et sauve alors automatiquement le contenu du plan SRAM dans le plan E^2PROM en une durée inférieure à 2,5 ms. A la mise sous tension, le contenu de l'E^2PROM est copié dans la SRAM. Le temps d'accès de cette mémoire (de la partie SRAM) est compris entre 35 et 55 ns. Elle n'existe qu'en boîtier DIP 24 broches et consomme typiquement 100 mA (250 µA en mode stand-by). Le diagramme de blocs de cette mémoire est le suivant :



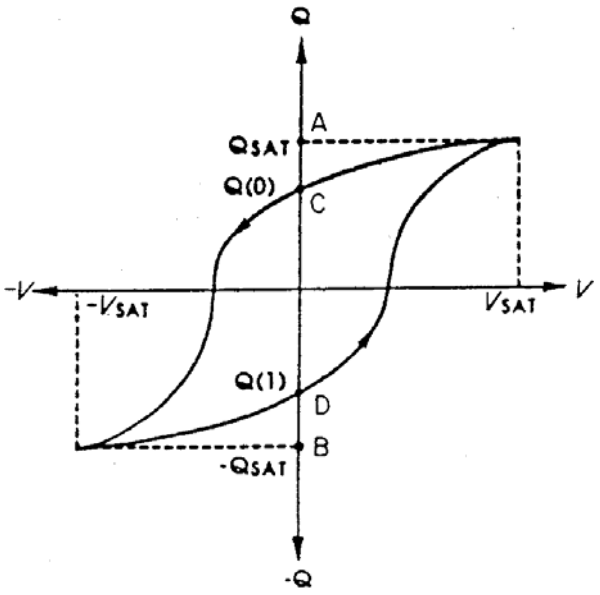
Son fonctionnement est strictement identique à celui d'une SRAM.

4.3.5 FRAM

4.3.5.1 Principe général

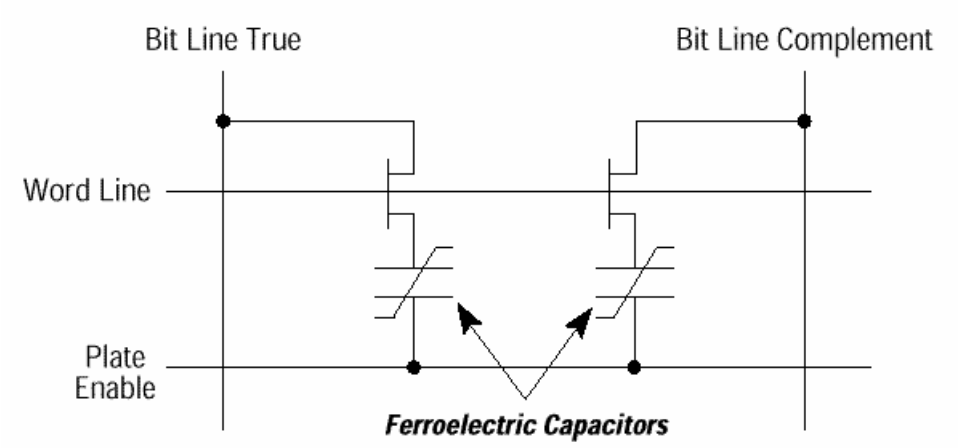
Un concept de RAM non volatile inhabituelle a été présenté par RAMTRON en 1988. La cellule de stockage de la RAM ferroélectrique est basée sur un condensateur constitué de deux plaques séparées par un matériau ferroélectrique. Ce diélectrique a deux propriétés intéressantes :

- Sa constante diélectrique étant 100 fois plus élevée que celle de l'oxyde de silicum (SiO2), c'est un meilleur isolant.
- Il existe un effet d'hysteresis (comme pour un matériau ferromagnétique, d'où son nom) dans le condensateur avec une charge stockée rémanente qui diffère selon la polarisation. C'est cette charge qui servira à stocker l'information. La figure suivante montre un exemple de courbe d'hysteresis de condensateur ferroélectrique soumis à une tension V :



Si la tension V atteint V_{SAT} puis est ensuite ramenée à 0, le condensateur conserve une charge positive Q(0) stockée. En faisant passer V à $-V_{SAT}$ puis à 0, la charge du condensateur devient négative et passe à Q(1). Si on sait détecter les charges stockées dans le condensateur, on peut réaliser une cellule mémoire avec ce type de condensateur.

La cellule de stockage utilisée par RAMTRON ressemble à une cellule de RAM statique avec deux condensateurs ferroélectriques polarisés de manière opposée en écriture.



La lecture s'effectue en polarisant les deux condensateurs dans le même sens, un amplificateur de lecture connecté sur les lignes bit mesurant la différence de charge entre les deux éléments. La lecture est donc destructive car elle implique le déchargement partiel des condensateurs de stockage. Comme pour les RAM dynamiques, la donnée mémorisée doit être régénérée à chaque lecture. Cette mémoire a comme caractéristiques principales :

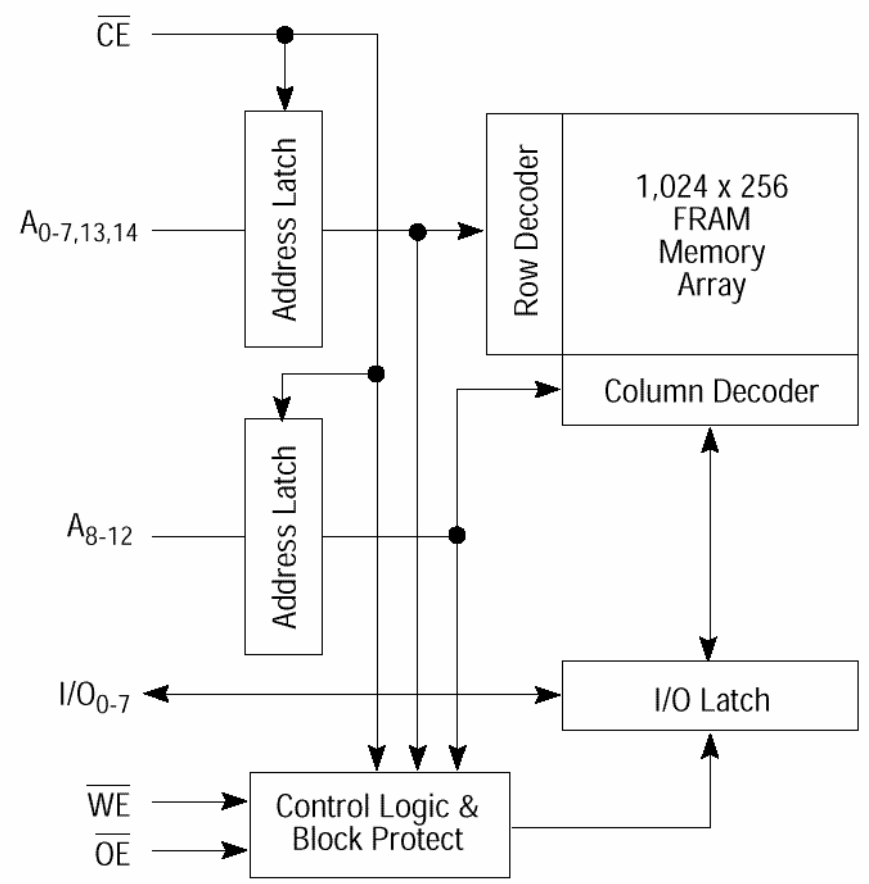
- Mise en œuvre identique à celle d'une SRAM.
- Temps d'accès en lecture de l'ordre de 200 nanosecondes.
- Temps de cycle d'écriture de l'ordre de 400 nansecondes.
- Consommation beaucoup plus faible qu'une E^2PROM ou une Flash.
- Une seule tension d'alimentation.
- Durée de rétention 10 ans.
- L'endurance n'est pas seulement limitée en écriture, mais aussi en lecture (de toute façon, une lecture implique une écriture). Par contre, le nombre de cycle de lecture/écriture est très élevé ($> 10^{10}$).
- Prix élevé.

Cette technologie peut a priori s'adapter à la fabrication de mémoires RAM dynamiques.

4.3.5.2 Exemple : la FM1808S de RAMTRON

RAMTRON commercialise une famille de FRAM série allant de la FM24C04 (512 x 8) à la FM25160 (2K x 8) et une famille de FRAM parallèle allant de la FM1208S (512 x 8) à la FM1808S (32K x 8). La FM1808S est une FRAM CMOS (32K x 8) alimentée en 3 V. Son temps d'accès en lecture est égal à 150 ns et son temps de cycle en écriture vaut 235 ns. Elle existe en boîtier SOP et TSOP 28 broches. Sa consommation typique est égale à 20 mA et

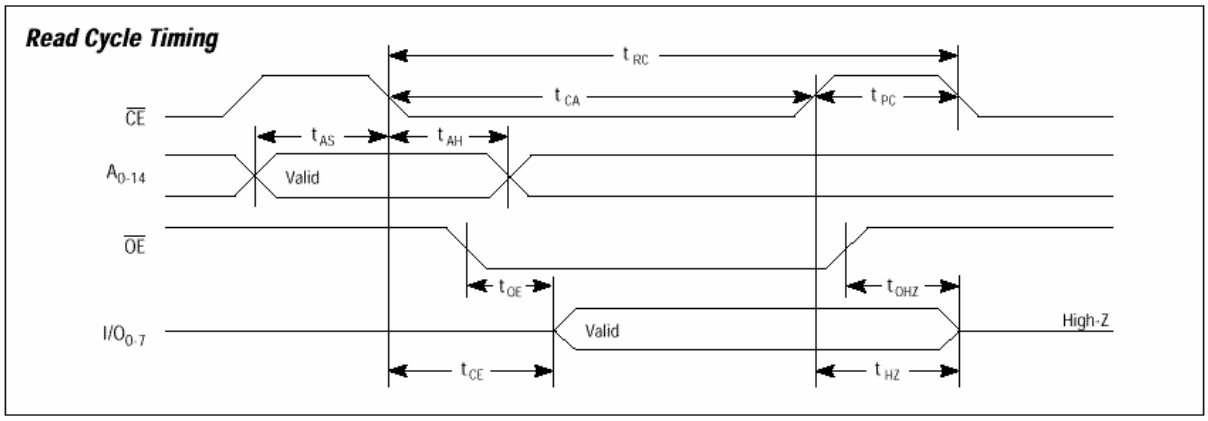
descend à 15 µA en mode stand-by. Elle est garantie pour 10^{12} cycles de lecture/écriture avec une durée de rétention égale à 10 ans. Le diagramme de blocs de cette mémoire est le suivant :

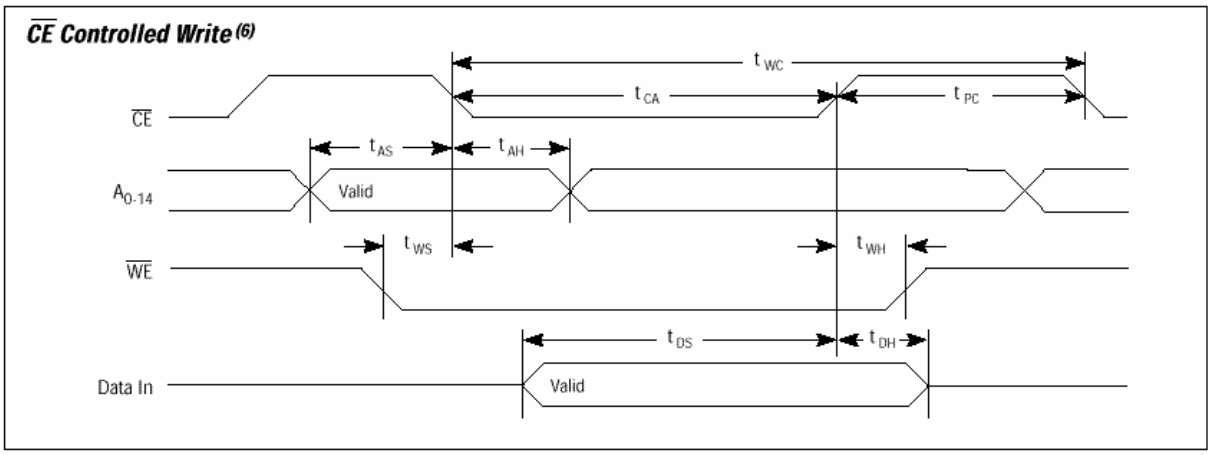


Il existe deux différences entre cette FRAM et une SRAM :

- Elle possède un système de protection en écriture logiciel,
- Les adresses sont mises en mémoire (latchées) sur le front descendant de $\overline{CE}$.

Les chronogrammes suivants montrent un cycle de lecture d'une FRAM (t_{CE} correspond au temps d'accès de la mémoire) ainsi qu'un cycle d'écriture. t_{WC} et t_{RC} correspondent aux durées des cycles d'écriture et de lecture.



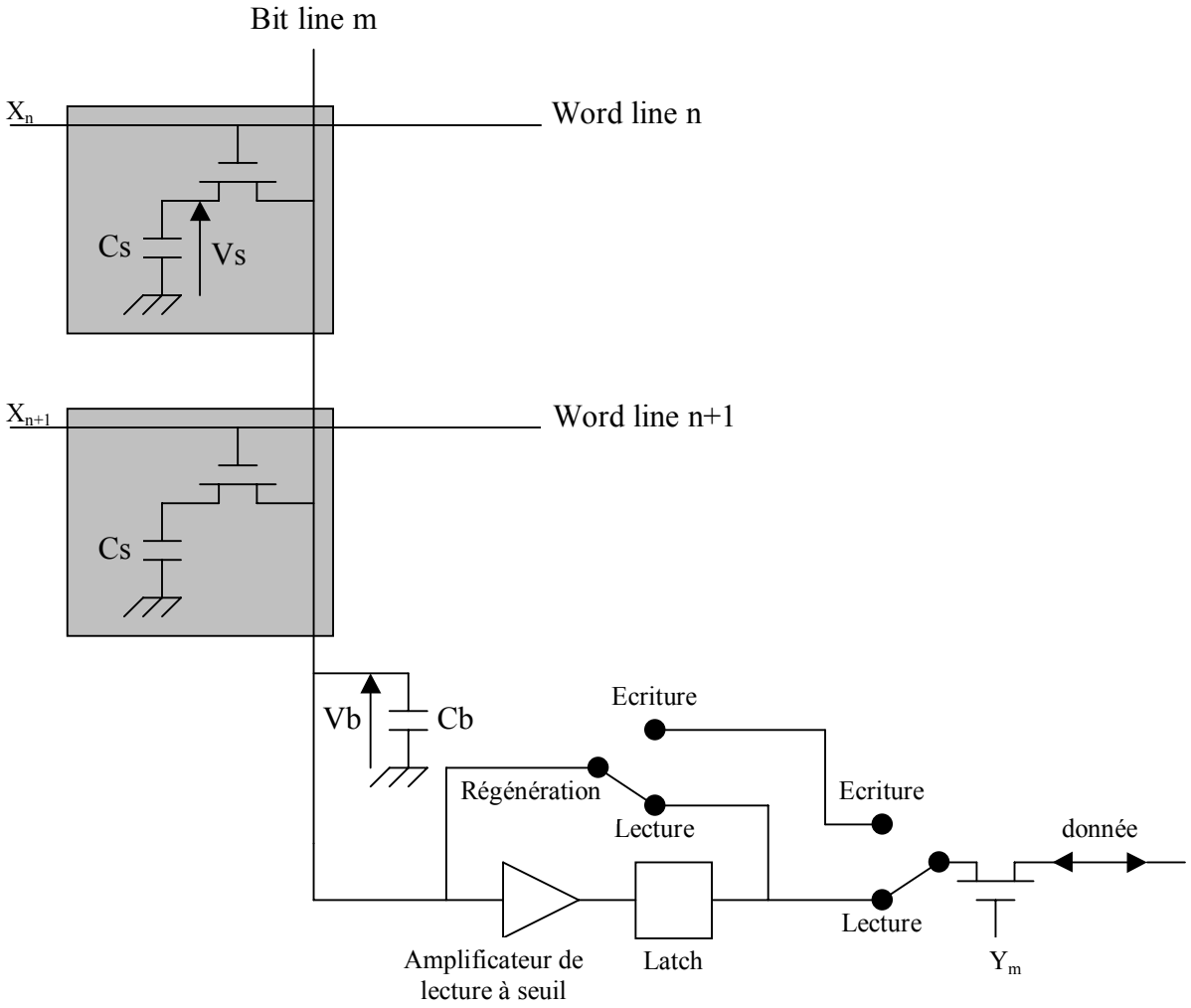


4.3.6 RAM dynamique

4.3.6.1 Modèles FPM et EDO

4.3.6.1.1 Principe général

Les RAM dynamiques (DRAM) forment le principal marché de mémoires et même le premier marché de l'industrie des semi-conducteurs. Les DRAM sont utilisées dans la mémoire centrale de tous les ordinateurs, ce qui représente un marché de masse. La notion de coût est donc ici essentielle. La structure simplifiée d'une portion de colonne de RAM dynamique apparaît sur la figure suivante. La cellule de stockage comprend un transistor de sélection de rangée en série avec un condensateur de stockage Cs. L'information est enregistrée sous forme d'une charge électrique contenue dans Cs d'une capacité d'environ 50 femtofarads (0.05 pF). La lecture de la cellule revient à connecter Cs sur la ligne de bit via le transistor de sélection de rangée et à lire la tension à ses bornes. On écrit dans la cellule en appliquant la tension de la ligne de bit sur Cs via le transistor de sélection.



Ce mode de fonctionnement pose deux problèmes :

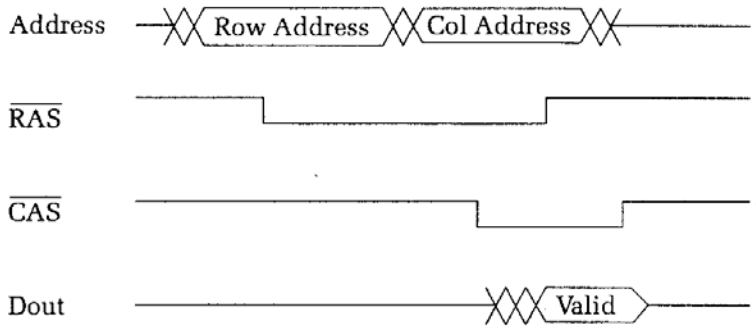
- La ligne de bit présente une capacité répartie Cb élevée (de l'ordre de 10 fois Cs) due au nombre élevé de cellules connectées sur la colonne. En mettant en parallèle Cb sur Cs, la tension Vb lue sur la ligne de bit vaut :

$$Vb = Vs.\frac{Cs}{Cs + Cb} \text{ (si Vb = 0 avant la lecture)}$$

Donc la tension lue est beaucoup plus faible (~ 10 fois) que la tension aux bornes de Cs. De plus, la lecture est destructive car Cs va se décharger dans Cb. Il faut donc régénérer l'information stockée à chaque lecture. L'amplificateur de lecture à seuil en bout de colonne doit donc être plutôt sensible et doit délivrer sur sa sortie le niveau logique stocké dans la cellule. Ce niveau doit ensuite pouvoir être réinjecté dans la cellule pour la régénération.

- L'oxyde de silicium (SiO2) utilisé comme diélectrique pour réaliser le condensateur Cs n'étant pas parfait, le condensateur va se décharger au cours du temps (en quelques millisecondes) même si on ne lit pas la donnée. Il faut donc rafraîchir régulièrement la donnée en la lisant puis en la réécrivant (ce qui correspond à une opération de lecture normale mais sans sortie sur le bus externe).

L'avantage de cette cellule de stockage est sa grande simplicité. Elle occupe une faible surface de silicium et donc son coût est faible. Les mémoires DRAM ont une très grande densité d'intégration et on sait faire aujourd'hui des mémoires atteignant 64 Mbits. Cela pose un problème supplémentaire car il faudrait 23 broches d'adresses pour adresser une mémoire 8M x 8, ce qui impliquerait des boîtiers de grande taille et donc un coût élevé (parce que le coût du boîtier est proportionnel au nombre de broches). On divise le nombre de broches d'adresses par deux en réalisant une opération de multiplexage. Au lieu d'envoyer toutes les adresses à la fois, on envoie d'abord l'adresse de ligne puis l'adresse de colonne. A cet effet, deux signaux supplémentaires ont été ajoutés par rapport à une SRAM, $\overline{RAS}$ (Row Address Strobe) et $\overline{CAS}$ (Column Address Strobe). Le chronogramme simplifié suivant montre le déroulement d'un cycle de lecture (le signal $\overline{WE}$ est à l'état haut pendant la durée du cycle) :

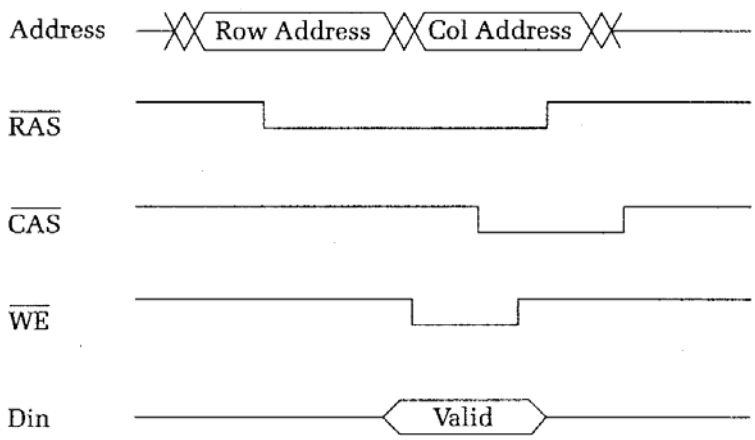


Il se compose des phases suivantes :

1. Présentation de l'adresse de ligne.
2. L'adresse de ligne est enregistrée par la mémoire sur le front descendant de $\overline{RAS}$. Toutes les cellules de la ligne sont lues puis mise en mémoire dans les latch de colonne.
3. Présentation de l'adresse de colonne.
4. L'adresse de colonne est enregistrée sur le front descendant de $\overline{CAS}$. La donnée apparaît en sortie après un temps de propagation.

5. Sur le front montant de $\overline{RAS}$, toutes les données de la ligne sélectionnée sont réécrites dans les cellules de stockage. C'est la régénération.
6. $\overline{CAS}$ remonte. La donnée repasse à l'état haute impédance après un temps de propagation.

Le chronogramme simplifié d'un cycle d'écriture se déroule de la manière suivante :

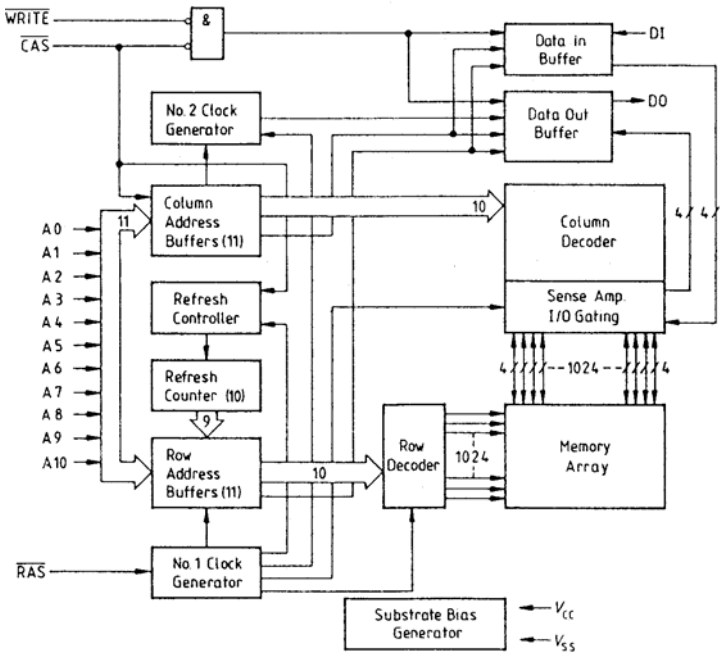


Les phases suivantes sont nécessaires pour écrire une donnée dans la mémoire :

1. Présentation de l'adresse de ligne.
2. L'adresse de ligne est enregistrée par la mémoire sur le front descendant de $\overline{RAS}$. Toutes les cellules de la ligne sont lues puis mise en mémoire dans les latch de colonne.
3. Présentation de l'adresse de colonne et de la donnée à écrire. Le signal d'entrée $\overline{WE}$ est mis à l'état bas.
4. L'adresse de colonne est enregistrée sur le front descendant de $\overline{CAS}$. La donnée Din présente sur le bus remplace la donnée mémorisée dans le latch de la colonne sélectionnée.
5. $\overline{WE}$ peut repasser à 1.
6. Sur le front montant de $\overline{RAS}$, toutes les données de la ligne sélectionnée sont réécrites dans les cellules de stockage.
7. $\overline{CAS}$ remonte. Le cycle d'écriture est terminé.

Il faut noter qu'une RAM dynamique ne comprend pas de signaux de sélection de boîtier. Ce sont $\overline{RAS}$ et $\overline{CAS}$ qui en tiennent lieu. Le schéma suivant montre la structure interne d'une

RAM dynamique 1M x 4. On voit immédiatement la grande complexité ce type de mémoire dont le fonctionnement est loin d'être trivial.



Nous avons parlé du mécanisme de régénération après lecture, mais nous n'avons pas encore abordé le problème du rafraîchissement. Il s'effectue par rangée entière et ressemble à un cycle de lecture simplifié. Il faut lire une rangée entière de cellules en interne, puis la réécrire. L'ensemble des rangées doit être rafraîchie périodiquement, une période durant de l'ordre de quelques dizaines millisecondes. Deux modes de rafraîchissement existent :

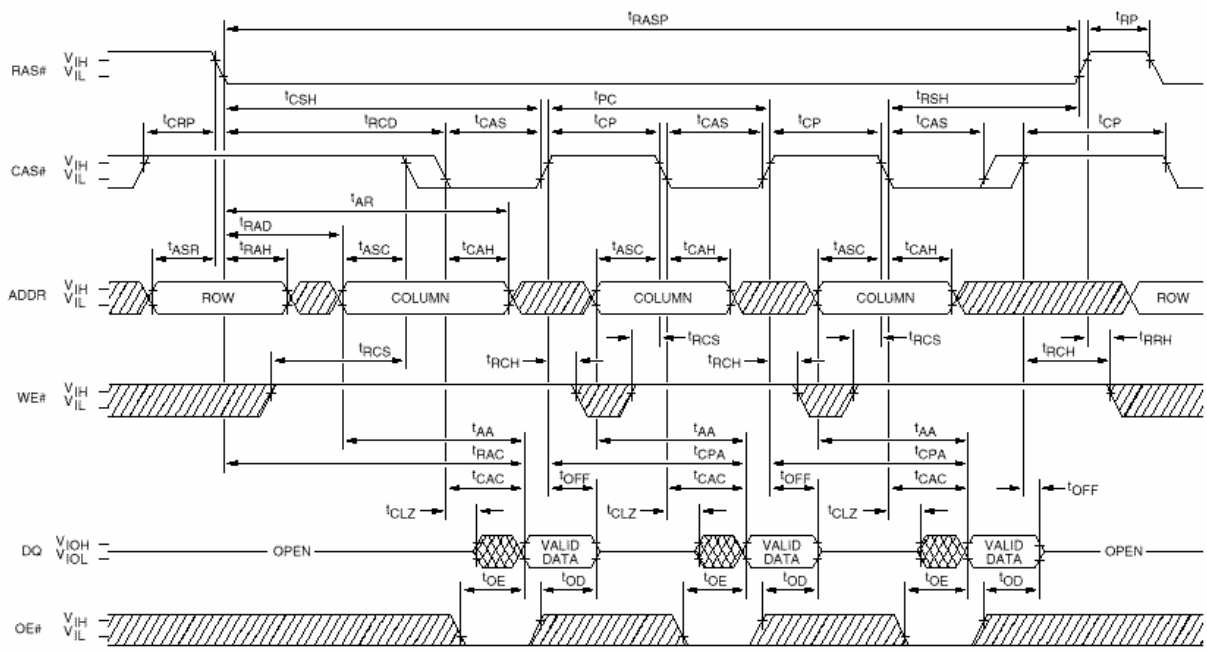
- RAS-only. Il faut présenter l'adresse de la ligne à rafraîchir, faire passer $\overline{\text{RAS}}$ à 0 pour lire la ligne puis faire remonter $\overline{\text{RAS}}$ à 1 pour la réécrire. Un dispositif extérieur à la mémoire doit gérer ce mode pour générer les bonnes adresses de ligne.
- Pour simplifier le contrôleur extérieur, le mode CAS-before-RAS a été créé. Un compteur d'adresses de ligne interne est ajouté dans la mémoire. Quand $\overline{\text{CAS}}$ passe à 0 avant $\overline{\text{RAS}}$, la mémoire reconnaît une commande de rafraîchissement, la ligne pointée par le compteur est lue puis réécrite et le compteur est automatiquement incrémenté.

Nous avons décrit les bases du fonctionnement d'une mémoire dynamique avec adressage ligne et colonne multiplexé, mais nous n'avons pas expliqué comment cette structure de mémoire permet d'améliorer les performances d'une DRAM. Le mode d'accès par page permet de réduire le temps d'accès de la mémoire quand on lit ou quand on écrit plusieurs données successives (mode rafale ou burst) dans une même ligne en tirant partie de

l'adressage en ligne et en colonne. En effet, le temps d'accès en lecture à la donnée stockée est déterminé par :

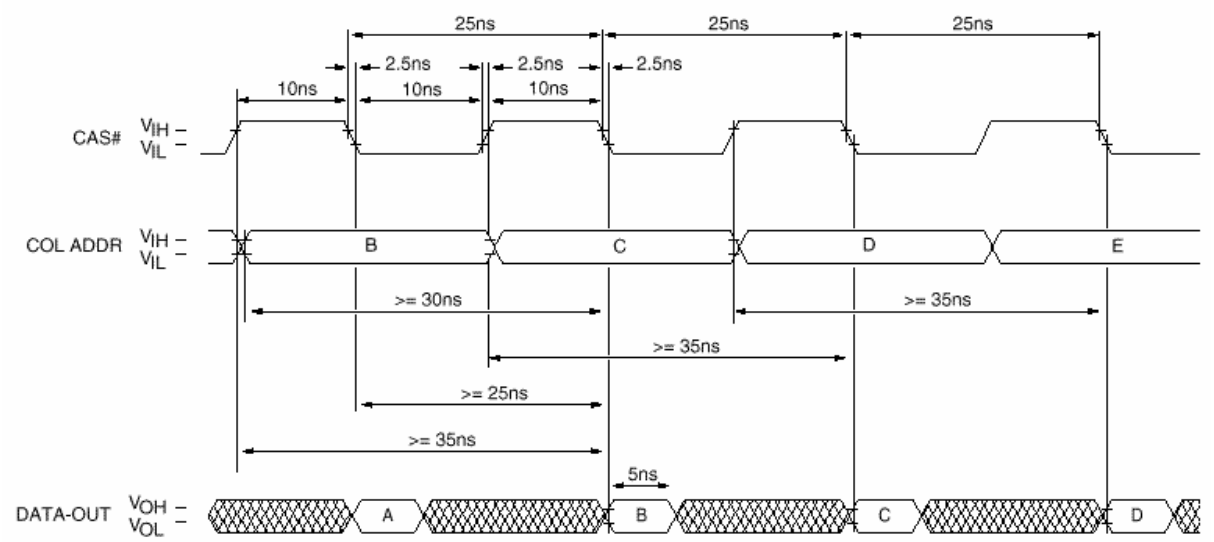
- le temps de propagation entre la présentation de l'adresse de ligne sur les broches du boîtier et la sélection effective de la ligne de la matrice de stockage plus le temps d'accès de la cellule de stockage, le temps de la lecture et le temps de mise en mémoire dans le latch : t_{PL}.
- le temps de propagation entre la présentation de l'adresse de colonne sur les broches du boîtier et la sortie de la donnée sur le bus t_{PC}.

Dans le mode d'accès par page rapide (FPM : Fast Page Mode), on accède à la première donnée en faisant passer $\overline{RAS}$ à 0, puis en faisant descendre $\overline{CAS}$ comme on l'a vu précédemment. Mais on peut accéder aux données suivantes sur la même ligne en activant seulement $\overline{CAS}$, le signal $\overline{RAS}$ restant à 0. A chaque front descendant de $\overline{CAS}$, la donnée pointée par la nouvelle adresse de colonne apparaît. Comme le temps de propagation de ligne t_{PL} n'existe plus, l'accès à la donnée est alors plus rapide. Sur le chronogramme suivant, le temps d'accès à la première donnée est t_{RAC} (ex : 50 ns) alors que le temps d'accès aux données suivantes est t_{CAC} (ex : 13 ns). D'autres modes d'accès rapide existe, mais le mode FPM est le plus utilisé.



Il existe une variante améliorée de la RAM FPM qui s'appelle RAM EDO (Extended Data-Out). Dans cette mémoire, la donnée ne passe pas à l'état haute impédance quand le signal

$\overline{CAS}$ remonte. Comme le montre le cycle de lecture suivant (DRAM EDO 60 ns), cela permet de faire remonter $\overline{CAS}$ et de commencer le cycle suivant avant que la donnée ne soit disponible sur le bus alors que dans une DRAM FPM, il faut attendre que la donnée soit disponible pour faire remonter $\overline{CAS}$ (voir chronogramme précédent).



La modification apportée à la DRAM FPM est mineure (une bascule D après les amplificateurs de lecture) et l'accès en mode page devient plus rapide d'environ 30 à 40 %, le temps d'accès à la première donnée d'une ligne restant identique. La DRAM EDO a remplacé la DRAM FPM dans les applications microinformatiques comme le PC par exemple avant d'être elle-même remplacée par la SDRAM.

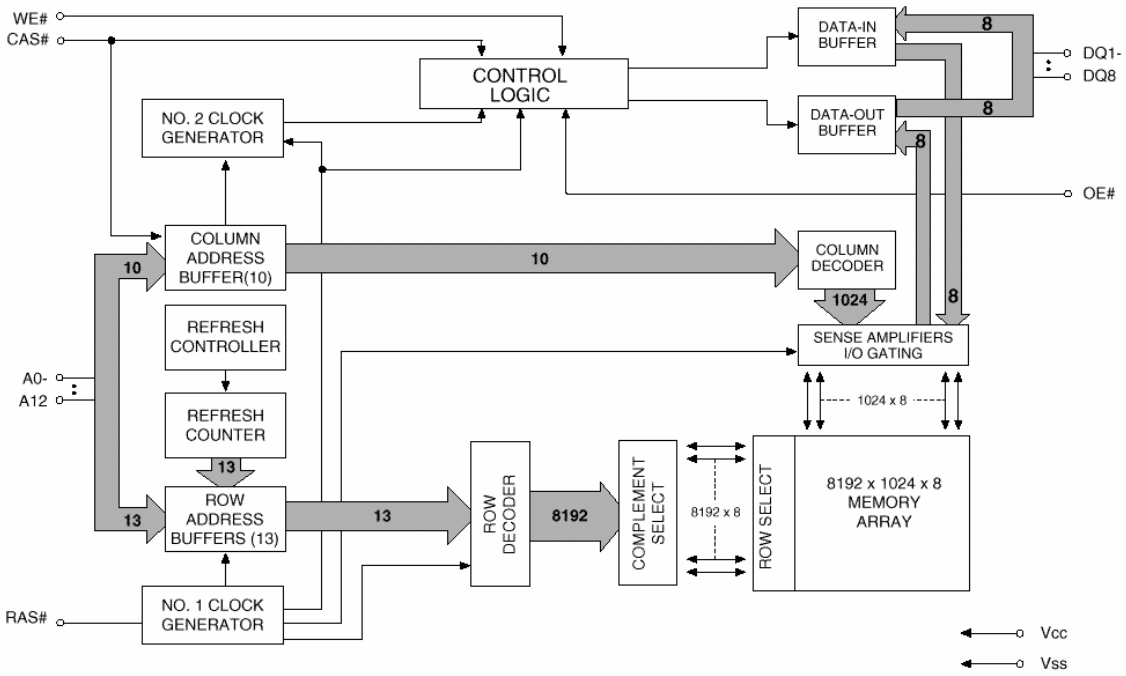
Etant donné la complexité d'utilisation d'une RAM dynamique, elle est rarement utilisée directement par un microprocesseur. Il existe un circuit contrôleur de mémoire dynamique qui sert d'interface entre le microprocesseur et les boîtiers mémoires. Ce contrôleur gère :

- Les extensions de capacité et de largeur de bus de donnée
- Les cycles $\overline{RAS}$ et $\overline{CAS}$ avec les conversions d'adresses,
- Les accès FPM,
- Le rafraîchissement.

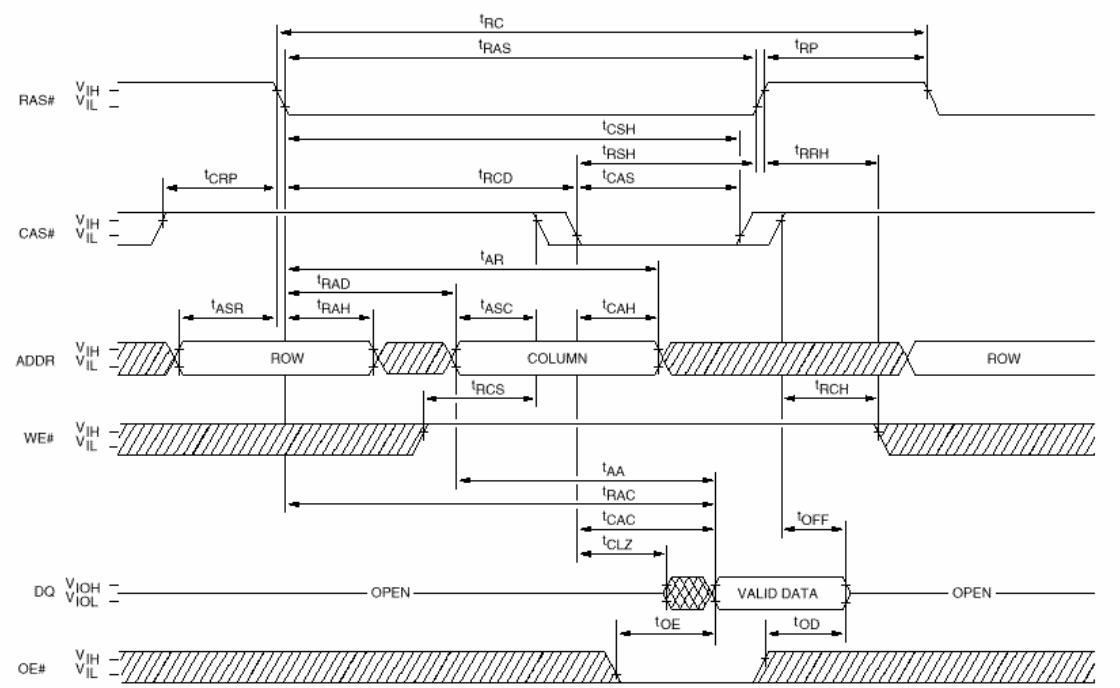
4.3.6.1.2 Exemple : la MT4LC8M8E1 de MICRON

MICRON commercialise une famille de DRAM allant de la MT4C16257 (256K x 16) à la MT4LC4M16F5 (4M x 16). La MT4LC8M8E1 est une DRAM CMOS (8M x 8) alimentée en

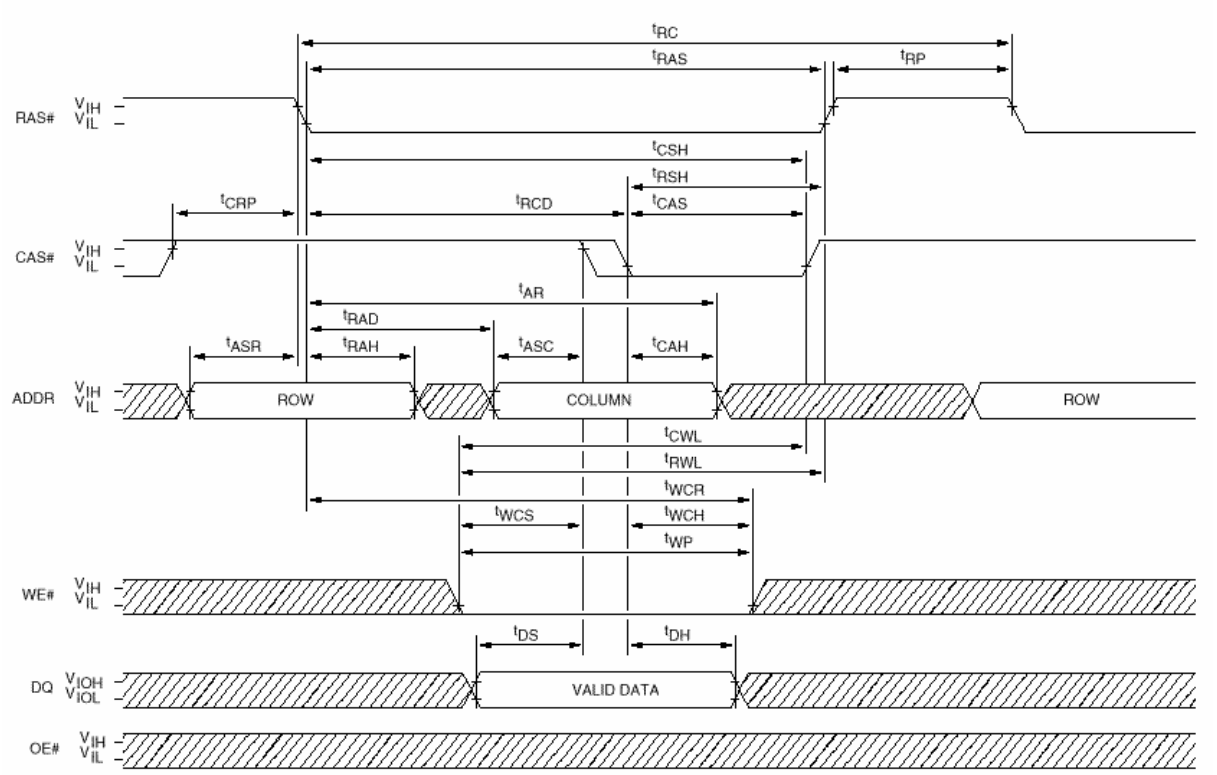
3,3 V. Son temps d'accès est égal à 50 ns ou 60 ns, elle existe en boîtier SOP et TSOP 32 broches. Sa consommation typique est égale à 135 mA et descend à 500 µA en mode stand-by. L'ensemble de ses 8192 lignes doit être rafraîchi toutes les 64 millisecondes. Cette mémoire est disponible soit en FPM, soit en EDO. Le diagramme de blocs de cette mémoire est le suivant :



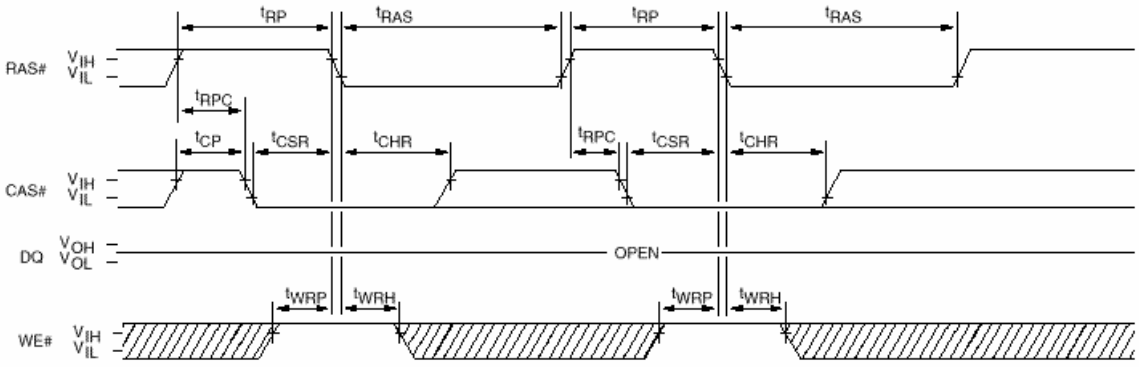
Le chronogramme suivant montre un cycle de lecture. T_{RAC} est le temps d'accès à la donnée et T_{RC} est le temps de cycle de lecture de la donnée.



Le chronogramme suivant montre un cycle d'écriture. T_{RC} est le temps de cycle d'écriture de la donnée.



Ce dernier chronogramme montre un cycle de rafraîchissement CBR.

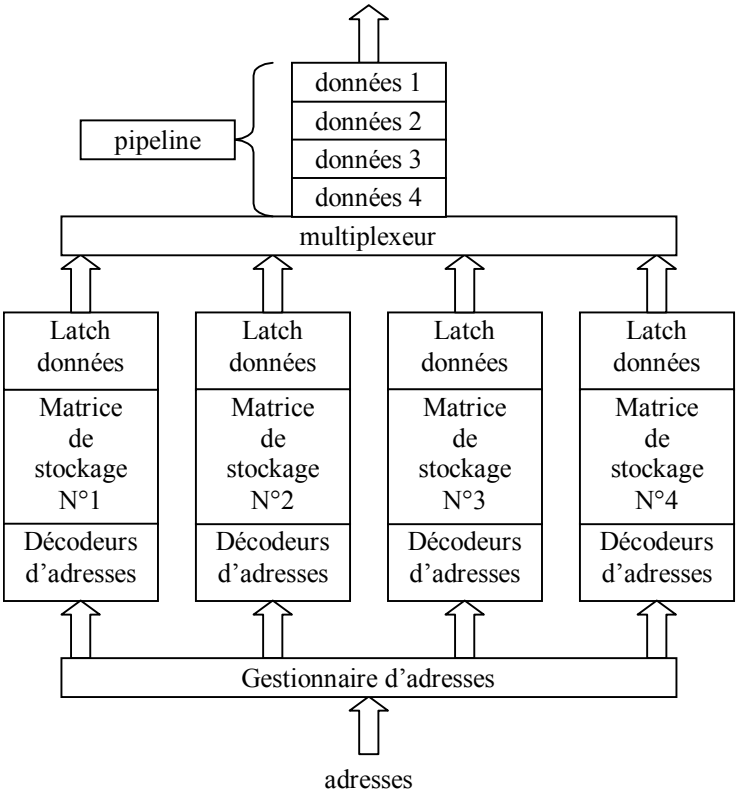


4.3.6.2 Evolution des DRAM

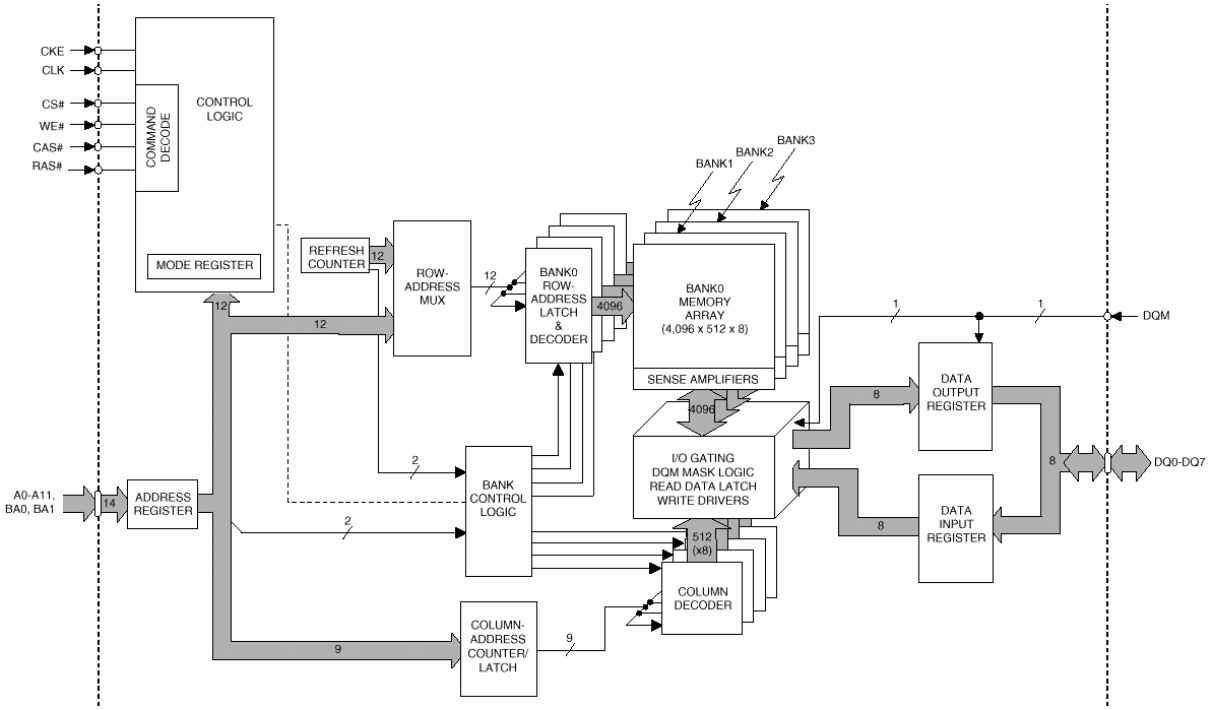
Les DRAM classiques FPM ou EDO ont atteint leurs limites en terme de temps d'accès en mode page. Deux solutions sont utilisées pour augmenter la fréquence de fonctionnement :

- La mémoire est rendue synchrone (tous les accès sont synchronisé avec une horloge).
- La structure interne est transformée pour mettre en œuvre un pipeline. Le principe est de faire fonctionner plusieurs plans mémoire en parallèle pour accélérer les accès.

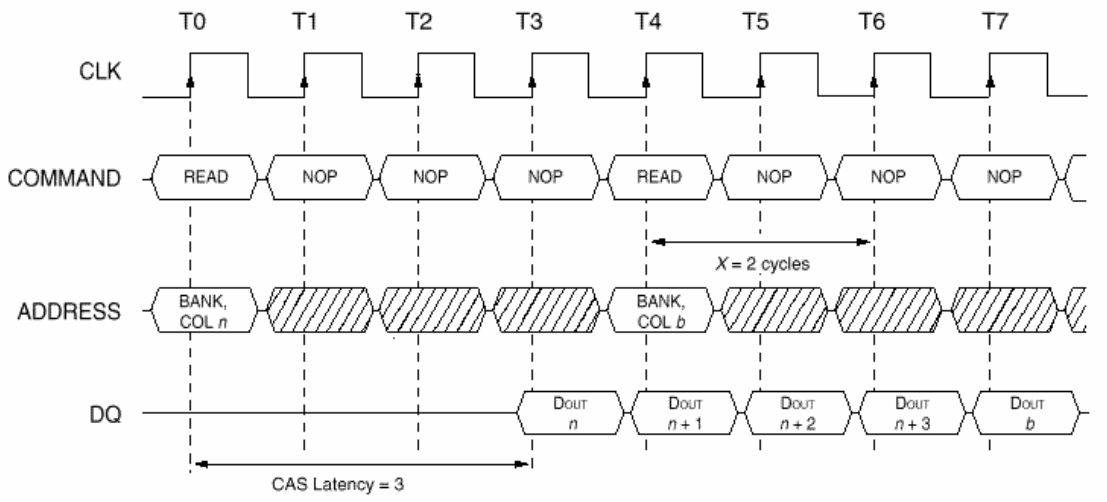
Le schéma de principe d'une architecture en pipeline est le suivant :



On trouve à l'intérieur de la mémoire 2 ou 4 sous-systèmes de mémorisation de type EDO ou FPM incluant une matrice de stockage ainsi que les décodeurs d'adresses de ligne et de colonne et les latch de données. Imaginons que l'on souhaite lire des données successives à partir de l'adresse N. Dans un premier temps, on va présenter l'adresse N sur la sous mémoire n°1 (SM1) et les adresses N+1, N+2 et N+3 sur les sous mémoires 2 (SM2), 3 (SM3) et 4 (SM4). Quand la première donnée est accessible sur SM1, elle entre dans le pipeline, le multiplexeur de sortie passe sur SM2 et l'adresse N+4 est présentée à SM1. Quand la deuxième donnée est accessible sur SM2, elle entre dans le pipeline, le multiplexeur passe sur SM3 et l'adresse N+5 est présentée à SM2. Quand la troisième donnée est accessible sur SM3, elle entre dans le pipeline, le multiplexeur passe sur SM4 et l'adresse N+6 est présentée à SM3 et ainsi de suite. Les sous-systèmes fonctionnent en parallèle, et on profite de la lecture dans une SM pour préparer l'accès dans les autres. Il faut noter que l'accès à la première donnée n'est pas plus rapide qu'avec une DRAM classique. Si vous rendez synchrone ce type de mémoire, vous obtenez une SDRAM (Synchronous DRAM). L'objectif est qu'à partir du moment où le pipeline est rempli, la mémoire accède à une nouvelle donnée à chaque coup d'horloge. Le diagramme de blocs suivant montre l'architecture interne d'une SDRAM MT48LC8M8A1 (8M x 8) de chez MICRON :

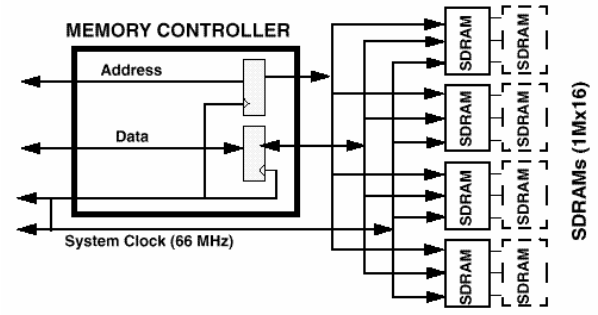


Ce type de mémoire est compliqué à utiliser et seul un contrôleur adapté permet d'en obtenir toutes les performances. Elle est utilisée à l'aide de mots de commande qui permettent de gérer ses différents modes de fonctionnement. Le chronogramme suivant montre un cycle de lecture simplifié. Pendant l'accès à la première donnée, un certain nombre de coup d'horloge est perdu : c'est le temps de latence. Les données suivantes sont obtenues au rythme de l'horloge.

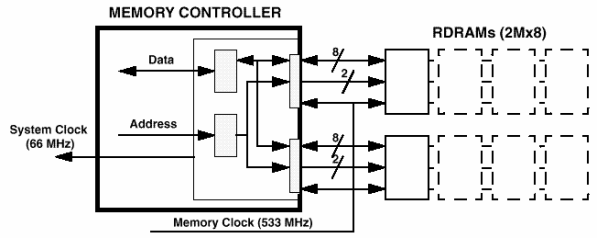


Cette mémoire fonctionne jusqu'à 125 MHz. Une évolution des SDRAM est la SDRAM DDR (Double Data Rate) qui fonctionne sur les fronts montant et descendant de l'horloge et permet de doubler la fréquence de fonctionnement.

Deux inconvénients des mémoires SDRAM sont le nombre de broches élevé qu'elles imposent au contrôleur (plus de 100) ainsi qu'une fréquence de travail encore limitée. Deux familles de DRAM sont donc apparues pour pallier ces défauts : les RDRAM (Rambus DRAM) et les SLDRAM (SyncLink DRAM). Leur principe est identique et repose sur un bus de communication étroit à très haut débit (> 500 Mo/s). Comme les SDRAM, elles sont synchrones et leur architecture interne est de type pipeline. Les deux schémas suivants montrent un plan mémoire de 16M x 8 réalisé avec des SDRAM et des RDRAM. Dans le premier cas, le contrôleur nécessite 120 broches alors que 62 broches suffisent dans le second. Il n'est pas évident de savoir aujourd'hui laquelle de ces deux technologies (RDRAM ou SDRAM DDR) sera utilisée dans les ordinateurs.



16M x 8 : SDRAM



16M x 8 : RDRAM

Il est à noter que certaines RAM statiques utilisent aussi les organisations en pipeline, le mode synchrone ainsi que les accès en rafales (burst). En effet, la fréquence de fonctionnement des microprocesseurs augmente bien plus rapidement que le temps d'accès des DRAM ne diminue. On sait faire aujourd'hui couramment des microprocesseurs fonctionnant à 600 MHz et aucune DRAM n'est capable de suivre une telle cadence. On insère donc entre le microprocesseur et la mémoire centrale une mémoire cache SRAM de petite taille (256 Ko à 1 Mo) qui sert à accélérer les échanges. Le microprocesseur va d'abord chercher (en écriture ou en lecture) les données dans le cache et n'accède à la DRAM que si la donnée ne s'y trouve pas. C'est dans ce domaine d'utilisation que l'on rencontre des mémoires SRAM synchrones fonctionnant par rafale et à architecture pipeline. Comme cette mémoire cache externe fonctionne à une fréquence inférieure ou égale à la fréquence interne du microprocesseur (par exemple à la moitié de la fréquence de fonctionnement pour un

Pentium III), il y a un premier niveau de mémoire cache de très petite taille (8 Ko à 64 Ko) à l'intérieur du microprocesseur qui lui fonctionne à la fréquence maximale du circuit.

Comme nous l'avons déjà vu, le microprocesseur accède en rafale aux données qui se trouvent généralement à des adresses consécutives. Le tableau suivant vous donne, à titre indicatif, le nombre de cycle d'horloge permettant d'accéder à la première donnée ainsi qu'aux trois suivantes pour différents types de mémoires :

type de mémoire	nombre de cycles en mode rafale
SRAM cache L2	2/1/1/1
DRAM FPM	5/3/3/3
DRAM EDO	5/2/2/2
SDRAM	5/1/1/1

4.4 Exercices

exercice 4.1

On dispose de mémoires mortes 16K x 4 ayant une entrée $\overline{CS}$.

1. Combien ces mémoires ont-elles de broches d'adresses ?
2. A quoi sert la broche $\overline{CS}$?
3. On souhaite réaliser un plan mémoire 32K x 8. Proposer un schéma de principe ainsi qu'une table d'adresses.

exercice 4.2

On dispose de mémoires mortes 8K x 8 ayant une entrée $\overline{CS}$.

1. Combien ces mémoires ont-elles de broches d'adresses ?
2. A quoi sert la broche $\overline{CS}$?
3. On souhaite réaliser une mémoire 32K x 16. Proposer un schéma de principe ainsi qu'une table d'adresses.

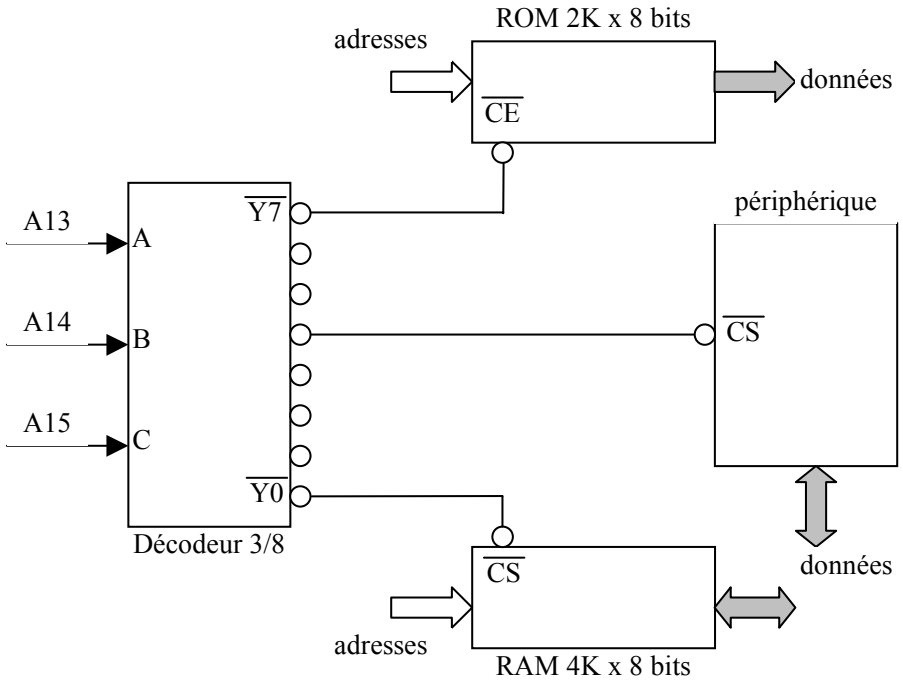
exercice 4.3

On travaille avec un bus d'adresses codé sur 16 bits. On dispose d'une ROM sélectionnée par $\overline{CS}$ câblé sur $\overline{A15}$ et d'une RAM sélectionnée par $\overline{CS}$ connectée à un NAND de A14, A13 et $\overline{A12}$.

1. Donner toutes les adresses sélectionnant la ROM.
2. Donner toutes les adresses sélectionnant la RAM.
3. Faire un tableau d'adresses. Sachant que l'on ne veut utiliser qu'un seul boîtier RAM et ROM, donner les adresses de ces éléments en recherchant la taille maximale possible en Kilo-octets. Existe-t-il des adresses images ?
4. Trouver les fonctions sélectionnant les zones libres et préciser leur taille.
5. On veut partager en 8 blocs successifs la zone débutant en 7000h. Donner le montage réalisant cette fonction en utilisant un décodeur 3 vers 8. On notera s0 à s7 les sorties du décodeur.

exercice 4.4

On travaille avec un bus d'adresses codé sur 16 bits. La figure suivante représente une partie du schéma d'une maquette à base de 6802.



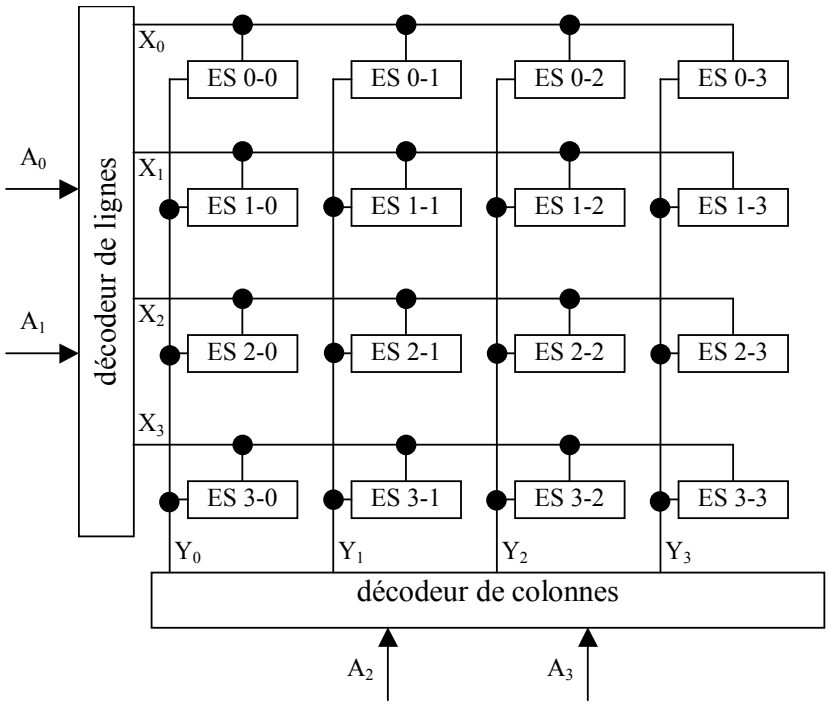
Le décodeur 3 vers 8 découpe l'espace adressé par le microprocesseur en 8 blocs d'adresses. Seules sont représentées sur le schéma les connexions concernant l'adressage et la sélection des boîtiers. Le type de périphérique n'a pas d'importance, seule son adresse de sélection nous intéresse.

1. Calculer la limite inférieure et supérieure ainsi que l'étendue de chaque bloc d'adresses que vous exprimerez en hexadécimal (faites un tableau).
2. Indiquez la plus petite et la plus grande adresse de la RAM et de la ROM. Que se passerait-il si on écrivait à l'adresse 1100h au lieu de 0100h ?
3. Quelles sont les adresses du périphérique.

exercice 4.5

1. Donner le schéma interne d'un décodeur 2 vers 4 avec et sans entrée G.
2. Combien de transistors CMOS sont nécessaires pour réaliser ce circuit (avec et sans G) ? Combien de transistors CMOS sont nécessaires pour réaliser un décodeur N vers 2^N (avec et sans G) ?
3. On souhaite réaliser une ROM 1M x 1. Calculer le nombre de transistor CMOS nécessaires pour réaliser une sélection d'adresses linéaire avec un décodeur unique. Comparer ce nombre avec le nombre de transistors nécessaire à la réalisation de la matrice de stockage.

4. Pour diminuer la taille du décodeur d'adresses, on utilise une autre méthode appelée sélection étagée. On utilise pour cela des décodeurs 4 vers 16 que l'on met en cascade. Proposer un schéma et calculer le nombre de transistors CMOS nécessaires. Comparer avec la question précédente et conclure.
5. On utilise maintenant une méthode différente appelée sélection matricielle. Les cellules de stockage sont maintenant organisées en une matrice 1024 x 1024 et les adresses sont séparées en deux, les adresses de ligne et les adresses de colonne. Le schéma suivant vous montre un exemple d'organisation matriciel 4 x 4.

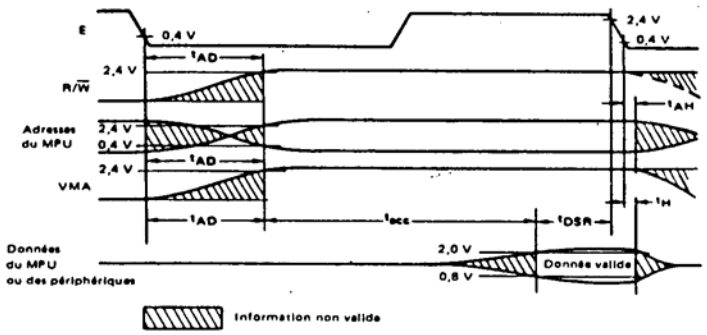
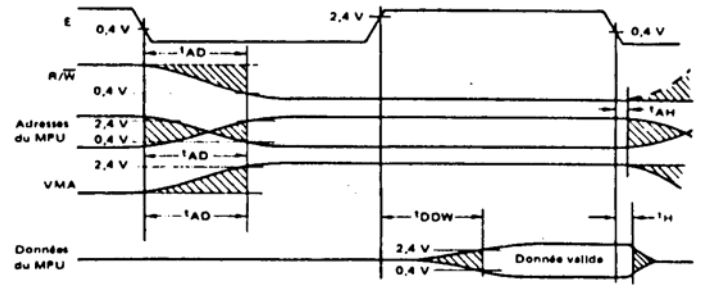


Calculer le nombre de transistor CMOS nécessaires pour réaliser les deux décodeurs. Comparer ce résultat avec ceux des questions 3 et 4. Est-il possible d'utiliser une sélection étagée dans ce type de montage ?

6. On utilise maintenant une matrice 4096 x 256. Combien de bits faut-il pour adresser X et Y. Quelle cellule est sélectionnée quand on présente l'adresse C35A9h à l'entrée de la mémoire ?

exercice 4.6

On travaille avec un microprocesseur de type 6802 fonctionnant à une fréquence égale à 1 MHz. Les cycles de lecture et d'écriture sont les suivants ($t_H = t_{AH} = 20$ ns) :

Cycle de lecture : t_{AD} max = 270 ns, t_{DSR} min = 100 ns,	
Cycle d'écriture : t_{AD} max = 270 ns, t_{DDW} max = 225 ns,	

On se place dans le cas le plus simple.

1. Expliquer la signification des temps t_{AD}, t_{DSR}, t_{DDW}, t_H et t_{AH}.
2. Donner le temps d'accès maximum autorisé par le microprocesseur pour une opération de lecture.
3. Donner le temps maximum alloué par le microprocesseur pour une opération d'écriture.
4. Les temps de maintien sont-ils difficiles à respecter ?

exercice 4.7

On souhaite réaliser un transcodeur binaire naturel – code de Gray sur 4 bits à l'aide d'une PROM.

1. Quelle taille de mémoire faut-il choisir ?
2. Proposer un schéma et donner le contenu de la mémoire.

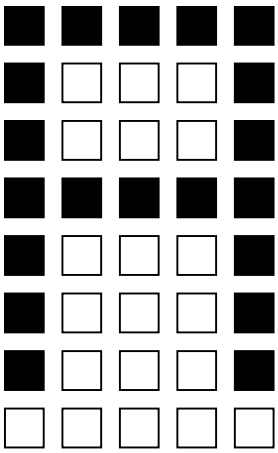
exercice 4.8

On souhaite réaliser un transcodeur binaire naturel – BCD (entrée sur 4 bits) à l'aide d'une PROM.

1. Quelle taille de mémoire faut-il choisir ?
2. Proposer un schéma et donner le contenu de la mémoire.

exercice 4.9

On souhaite réaliser un générateur de caractères pour un terminal d'ordinateur à l'aide d'une PROM. On place en entrée le code ASCII du caractère et on obtient sur plusieurs adresses consécutives les différents points de la matrice 5 x 8 représentant le caractère. Prenons l'exemple du A (code ASCII = 0100 0001). Sa matrice est la suivante :

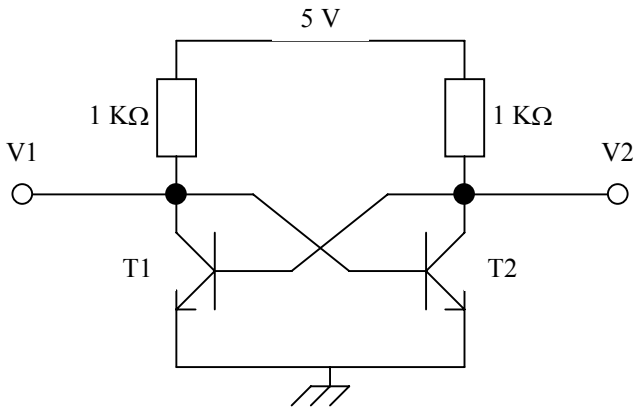


Les carrés noirs correspondent à des points allumés (valeur 1) et les carrés blancs à des points éteints (valeur 0).

1. Sachant que l'on cherche à afficher la partie basse de la table ASCII (128 valeurs), calculer la taille de la mémoire nécessaire à la génération.
2. Sur quelles adresses va-t-on présenter le code ASCII ? Sur quelles adresses va-t-on présenter la sélection de la ligne de balayage ? Donner les valeurs mises en mémoire pour la caractère A.

exercice 4.10

Soit le montage bistable à transistors bipolaires suivant (V_{CEsat} = 0.2 V, V_{BEsat} = 0.8 V, β=50) :



On suppose que le transistor T2 est saturé et le transistor T1 est bloqué.

1. Calculer les différentes tensions et courants de ce montage et vérifier l'hypothèse précédente. Cet état est-il stable.
2. On applique une tension de 0 V sur V2 et de 1 V sur V1. Que se passe-t-il ?
3. On applique une tension de 1 V sur V2 et de 0 V sur V1. Calculer les différentes tensions et courants du montage et déterminer l'état des transistors. Est–ce V1 ou V2 qui a fait basculer le montage ?
4. V1 et V2 sont maintenant déconnecté. Le montage est-il stable ? a-t-on réalisé une mémoire ?
5. Proposer une méthode permettant de détecter l'état stocké.

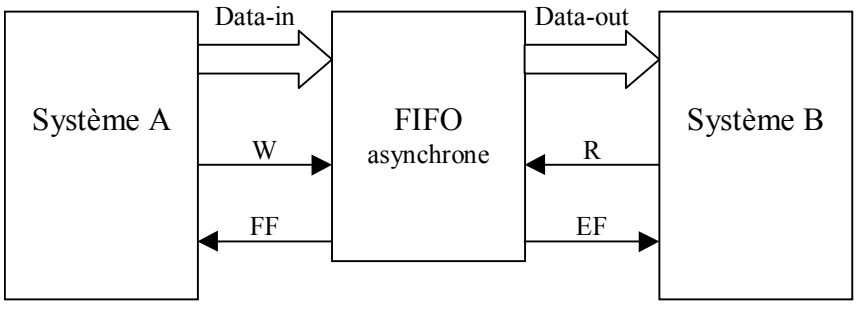
exercice 4.11

Dans une RAM dynamique, au moment de la lecture, tout se passe comme si on mettait en parallèle le condensateur parasite Cb sur le condensateur de stockage Cs. On suppose que le condensateur est entièrement déchargé au début de la lecture et qu'il n'y a pas de perte de charge dans le montage.

1. Ecrire la valeur de la charge stockée dans Cb et Cs à l'instant initial.
2. Ecrire la valeur de la charge stockée dans Cb et Cs à la fin de la lecture.
3. Retrouver la formule vue dans le cours exprimant la tension lue aux bornes de Cb à la fin de la lecture en fonction de la tension aux bornes de Cs à l'instant initial, de Cs et de Cb.

exercice 4.12

On cherche à faire communiquer deux systèmes ayant des fréquences de fonctionnement différentes selon le schéma :



Le FIFO asynchrone 4 x 8 possède une sortie FIFO plein (FF) et une sortie FIFO vide (EF).

Le montage respecte les propriétés suivantes :

- La donnée est écrite sur le front montant de W, elle est lue sur le front montant de R.

- La fréquence du système A est égale à 10 KHz alors que la fréquence du système B n'est que de 3.3 KHz (les deux horloges sont en phase).
- On suppose que la première donnée à transmettre vaut 0 et que les données suivantes sont incrémentées de 1 à chaque coup d'horloge.
- Le système A n'écrit pas dans le FIFO quand il est plein, le système B ne lit pas le FIFO quand il est vide.
- W et R sont des impulsions de courte durée déclenchées par le front montant des horloges.

1. Représenter l'état interne du FIFO ainsi que les différents chronogrammes pendant une transmission à partir de l'initialisation.
2. Quel est le débit réel de la transmission en régime établi ?

exercice 4.13

Soit la mémoire AM27C1024 dont la documentation se trouve à la page A-25.

1. Quelles sont les caractéristiques générales de ce circuit ?
2. Comment varie le courant d'alimentation avec la fréquence et la température ?
3. Que représentent les temps t_{ACC}, t_{CE}, t_{OE}, t_{DF} et t_{OH} ?
4. Comment efface-t-on cette mémoire ?
5. Comment programme-t-on cette mémoire ?

exercice 4.14

Soit la mémoire CY7C109 dont la documentation se trouve à la page A-37.

1. Quelles sont les caractéristiques générales de ce circuit ?
2. On s'intéresse au cycle de lecture n°2. Que représentent les temps t_{RC}, t_{ACE}, t_{DOE}, t_{HZOE} et t_{HZCE} ?
3. On s'intéresse au cycle d'écriture n°2. Que représentent les temps t_{WC}, t_{SCE}, t_{SA}, t_{AW}, t_{PWE}, t_{HA}, t_{SD} et t_{HD} ?