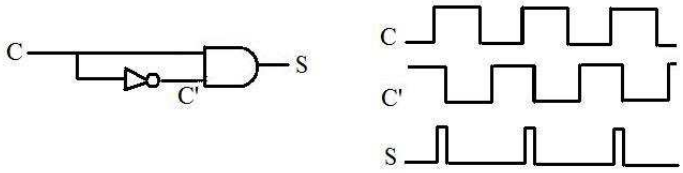
VIII- Circuits séquentiels. Mémoires

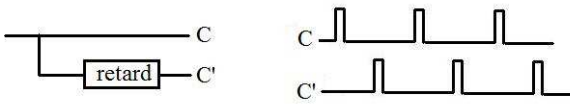
Maintenant le temps va intervenir. Nous avions déjà indiqué que la traversée d'une porte ne se faisait pas instantanément et qu'il fallait en tenir compte, notamment lors de la propagation de la retenue dans une addition. D'autre part, un certain timing dans la succession des évènements est indispensable, pour synchroniser les opérations et éviter à la machine de s'emmêler les pinceaux. C'est là qu'intervient l'horloge interne, qui délivre une alternance régulière de signaux 0 et 1, haut et bas, grâce à un oscillateur à quartz. En combinant l'horloge et le retard lors de la traversée d'une porte, on obtient de nouveaux circuits logiques, comme le détecteur de front montant, ou la division du cycle d'horloge en sous-cycles.

Détecteur de front montant

Faisons passer le signal d'horloge *C* dans un fil, avec une dérivation sur un deuxième fil qui fait traverser au signal une porte NON. Cela introduit un léger décalage entre les deux signaux *C* et *C'* obtenus. La mise en place d'une porte ET donne à la sortie *S* une succession de brèves impulsions qui se produisent lorsque le signal d'entrée passe de 0 à 1. Le circuit détecte cette transition qui correspond au front montant du signal d'horloge. Un tel procédé est notamment utilisé dans le conception des mémoires, comme nous le verrons dans la suite.



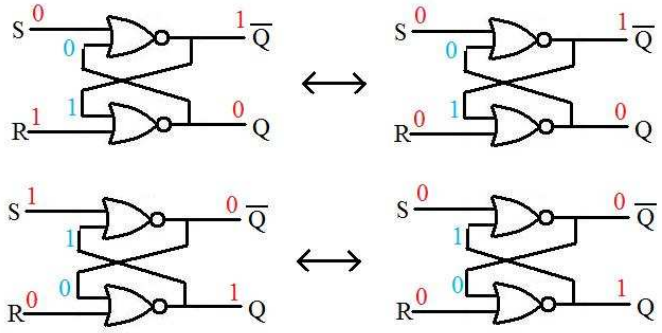
En introduisant une nouvelle dérivation dans ce signal à impulsions, avec un temps de retard plus important, on peut obtenir deux signaux décalés d'un demi-cycle d'horloge l'un par rapport à l'autre. Cela permet de gérer certains évènements deux fois plus rapidement, par rapport au cycle d'horloge initial.



Les circuits séquentiels que nous allons voir maintenant intègrent les notions de temps et de mémorisation. Ils permettent d'obtenir une sortie qui est fonction des entrées, mais aussi de ce qui s'est passé antérieurement. Tout se passe comme si le circuit gardait une mémoire du passé. C'est par exemple le cas d'un bouton poussoir qui allume une lampe lorsque l'on appuie dessus, puis qui l'éteint quand on appuie encore, puis qui la rallume quand on ré-appuie, … Sans mémorisation, cela ne serait pas possible. Les circuits logiques à mémoire que nous allons voir sont de deux sortes. Il s'agit soit de verrous (*latch* en anglais) permettant une forme rudimentaire de mémorisation, soit de bascules au nom évocateur de *flip flops* en anglais, qui sont plus pratiques pour de nombreuses applications, notamment la conception de mémoires dans l'ordinateur. Si la notion de temps est fondamentale dans les circuits séquentiels, l'horloge interne de l'ordinateur n'est pas pour autant concernée dans les cas les plus simples, notamment celui du verrou SR, que nous allons commencer par étudier.

1) Le verrou SR

Cette porte logique comporte deux entrées *S* pour *Set* et *R* pour *Reset*, ainsi que deux sorties dont on va voir que l'une est la négation de l'autre, soit *Q* et $\overline{Q}$. Dans ces conditions seule la sortie *Q* nous intéresse. D'autre part on considère comme impossible de faire *set* et *reset* en même temps, d'avoir à la fois *S* = *R* = 1. Le verrou *SR* comporte deux portes NON OU, et deux boucles de rétroaction qui font intervenir le temps. Un changement à l'entrée fait passer de la valeur *Q* qui existait jusque là à une nouvelle valeur de *Q*, que nous noterons *Q**.



Que va-t-il se passer ? Imaginons que le *Reset* ait lieu, à savoir *R* = 1 et *S* = 0. Le forçage de *R* à 1 provoque une valeur *Q* en sortie égale à 0, à cause de la deuxième porte NON OU où se trouve *R*. A son tour la première porte impose que $\overline{Q}$ = 1. Le fait de se mettre ensuite à *R* = 0 et *S* = 0 maintient la situation précédente, aucune contradiction ne se produisant. Il en est de même pour le *Set*, avec *S* = 1 et *R* = 0. La sortie *Q* doit passer à 1 et $\overline{Q}$ à 0. Le fait de remettre les pendules à 0 maintient la situation avec *Q* = 1 et $\overline{Q}$ = 0.

On en déduit le comportement du verrou SR :

S	R	comportement
0	0	pas de changement
0	1	Q* = 0
1	0	Q* =1
1	1	état interdit

Résumons ce qui se produit. Le *Reset* provoque le maintien de *Q* à 0 ou son passage à 0. Le *Set* provoque le maintien de *Q* à 1 ou son passage à 1. Et dans le cas *S* = *R* = 0, il y a conservation de la valeur de *Q*, ce qui constitue une mémorisation (ou un verrouillage) qui peut durer aussi longtemps qu'on le désire, notamment si la sortie *Q* est à 1, elle va le rester même si *S* passe à 1. Elle ne passera à 0 que si l'on fait *R* = 1 (avec *S* = 0).

Exercice 1: Construire la table de vérité du verrou SR avec en entrées *S*, *R* et l'ancienne valeur de *Q*, et en sortie la nouvelle valeur de *Q*, soit $\overline{Q}$. Construire le tableau de Karnaugh et en déduire l'équation du verrou SR.

Table de vérité

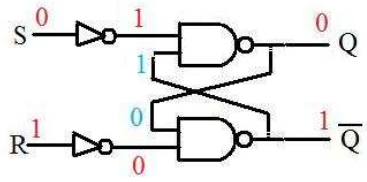
Q	S	R	Q*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	x

Tableau de Karnaugh

SR / Q	00	01	11	10
0	0	0	x	1
1	1	0	x	1

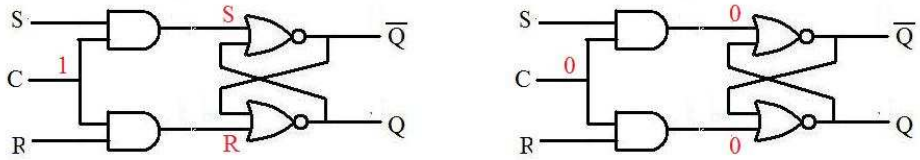
On en déduit l'équation : $Q^* = S\overline{R} + Q\overline{R}$ ou même $Q^* = S + Q\overline{R}$ si l'on considère que dans les cases x on peut mettre ce que l'on veut.

Exercice 2 : Construire le verrou SR en utilisant $\overline{R}, \overline{S}$ et des portes NON ET

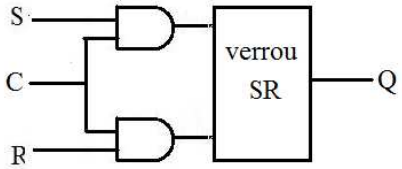


2) Le verrou SRH

Maintenant ajoutons un signal d'horloge *C* (*clock*) en entrée, avec sa succession régulière d'états 0 et 1. On fait en sorte que pour *C* = 1, le verrou SRH se comporte comme le verrou SR classique, et que lorsque *C* passe à 0, il se met en position mémoire, c'est-à-dire que *Q* reste dans l'état où il se trouvait avant le passage de *C* à 0.



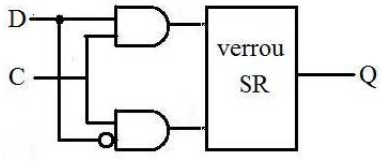
soit :



3) Le verrou D

C'est un dérivé du cas précédent, avec une seule entrée *D* au lieu de *S* et *R*, telle que $D = S$ et $\overline{D} = R$. On en déduit la table de vérité et le schéma électronique :

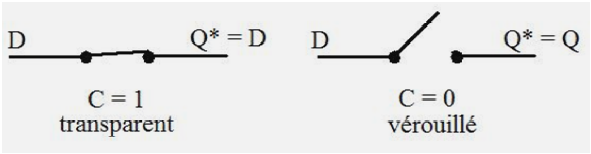
C	D	Q*
0	0	Q
0	1	Q
1	0	0
1	1	1



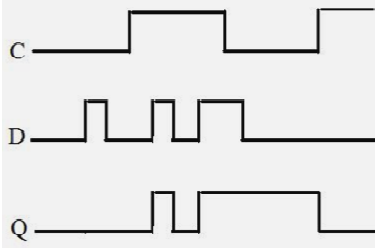
Il y a donc deux cas :

- Lorsque *C* = 1, la sortie *Q* reproduit l'entrée *D*, on dit que le circuit est transparent.
- Lorsque *C* passe à 0, *Q* conserve la valeur qu'elle avait avant, les variations possibles de *D* n'ont plus aucun effet, on dit que le circuit est verrouillé.

Ces deux comportements peuvent être schématisés par la présence d'un interrupteur fermé ou ouvert :

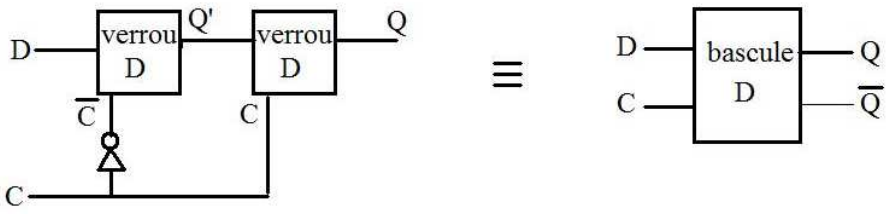


Remarquons que le cas interdit *S* = *R* = 1 du verrou SR ne se produit plus dans le verrou *D*.
Voici un exemple d'évolution dans le temps, où l'on suppose que *Q* est à 0 au départ :

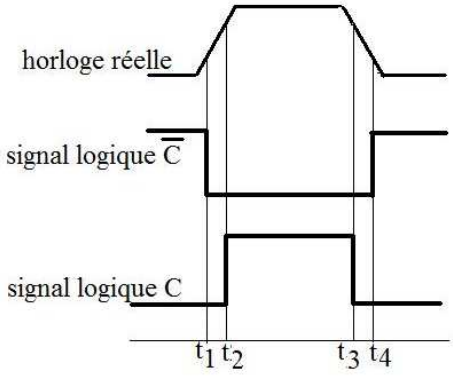


4) La bascule D, ou *flip flop* D

Prenons maintenant deux verrous D en succession, le premier étant appelé « maître » et le second « esclave ». Comme l'entrée du deuxième recopie la sortie du premier, l'esclave est effectivement asservi au maître. D'autre part l'horloge C synchronise les deux verrous mais cela est effectué en opposition, par le biais d'un inverseur. Le signal d'horloge arrivant sur le premier verrou est $\overline{C}$, tandis que celui sur le deuxième est ***C***.

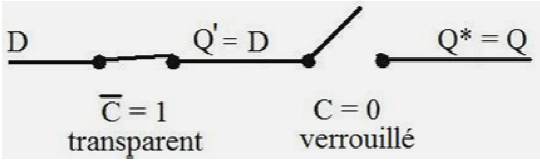


Reprenons la représentation du verrou D par un interrupteur fermé ou ouvert. La bascule comporte deux interrupteurs en série, et à cause des deux temps C et $\overline{C}$ inversés, un des interrupteurs est ouvert et l'autre fermé, quoi qu'il arrive, et rien ne passe d'une extrémité à l'autre de la bascule. L'entrée *D* n'a donc jamais aucune influence sur la sortie *Q*. C'est du moins ce qui se passerait idéalement, et la bascule D n'aurait aucun intérêt. Heureusement il n'en est pas ainsi. En fait l'horloge n'émet pas des impulsions parfaites. Son passage du niveau bas 0 au niveau haut 1, ou inversement, n'est pas instantané, il demande un court intervalle de temps, lors de la montée ou de la descente. C'est là que tout va se jouer pour la bascule D. On va considérer que l'inverseur qui crée $\overline{C}$ bascule de 1 à 0 légèrement avant les portes logiques du verrou D soumis à *C*, puisque l'esclave doit attendre le maître, et légèrement après lorsqu'il bascule de 0 à 1. Les signaux logiques $\overline{C}$ et C réagissent de la façon suivante par rapport au signal réel de l'horloge :

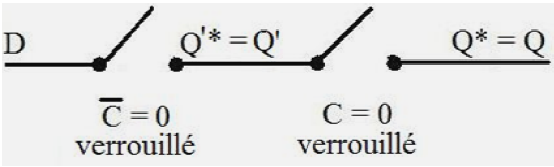


On pourra constater que pendant des laps de temps très courts, *C* et $\overline{C}$, normalement opposés, vont être égaux. Il s'ensuit les phénomènes suivants, suivant les intervalles de temps successifs :

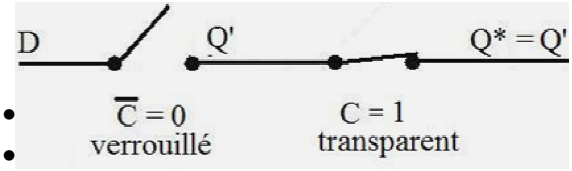
- Avant le temps t_1, le maître est transparent et l'esclave est verrouillé, la sortie *Q'* du maître a la valeur de l'entrée *D*, mais la sortie finale *Q* conserve son état :



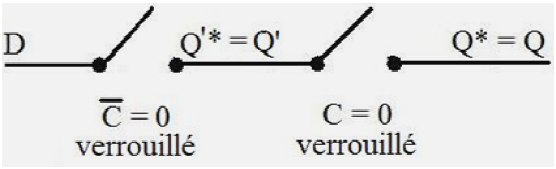
- Au temps t_1, et de même jusqu'au temps t_2 , le premier verrou se verrouille à son tour, *Q'* garde la valeur qu'il avait, celle de l'ancien *D*, et la sortie *Q* reste inchangée:



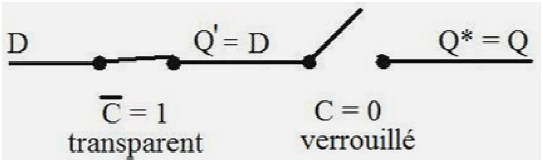
- Au temps t_2, et jusqu'au temps t_3, c'est là que tout va se passer ! La sortie *Q'* de la première porte ne change pas, elle garde la valeur qu'avait l'entrée *D* à l'instant t_1, mais maintenant elle traverse la deuxième porte, et la sortie *Q* prend dès l'instant t_3 cette valeur de *D* à t_1 :



- Au temps t_3, et pendant le court instant allant jusqu'à t_4, les deux signaux d'horloge sont à 0, et rien ne change.

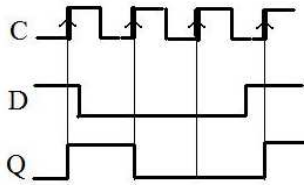


- A l'instant t_4 et jusqu'à la fin du cycle, la fermeture de la première porte fait passer *D* dans *Q'* mais la sortie finale *Q* ne change pas. La boucle est bouclée, et un nouveau cycle peut commencer.



Finalement, la valeur de l'entrée *D* n'est prise en compte que lors du front montant de l'horloge, et elle est transférée dans la sortie *Q*. Le basculement n'est possible qu'à ce moment-là. Entre deux fronts montants de l'horloge, la sortie *Q* ne change pas.

Voici un exemple d'évolution dans le temps, donnant *Q* à partir de *D* :

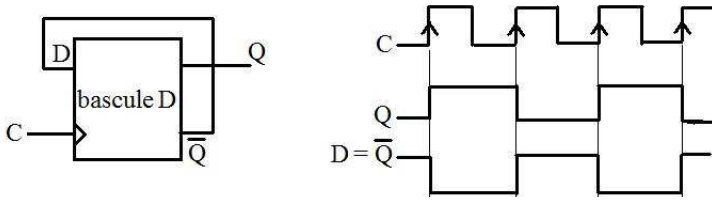


Les bascules D qui commutent sur le front montant du signal d'horloge sont dites *positive edge triggered*, il en existe d'autres qui réagissent sur le front descendant.

5) Quelques applications des bascules D

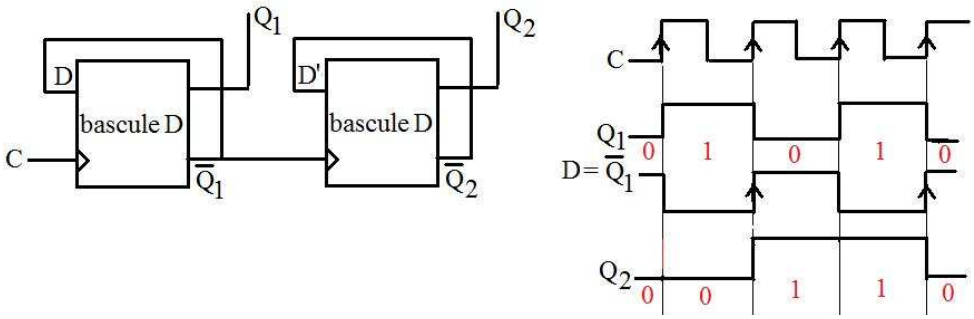
5-1) Conception d'un diviseur de fréquence par deux

Prenons une bascule D et envoyons sa sortie $\overline{Q}$ sur l'entrée *D*. Que va-t-il se passer ? Supposons qu'à l'instant initial *Q* soit à 0 (et $D = \overline{Q}$ à 1). Lors du premier front montant de l'horloge *C*, *Q* prend la valeur qu'avait *D*, soit 1, et $D = \overline{Q}$ passe à 0. Il en est de même jusqu'au front montant suivant, où *Q* prend la valeur 0 qu'avait D, et $\overline{Q}$ passe à 0. Et ainsi de suite de façon cyclique. La sotie *Q* est périodique, avec une période qui est le double de celle de l'horloge *C* (ou encore sa fréquence est la moitié de celle de *C*).



5-2) Compteur

Mettons maintenant deux bascules D diviseurs de fréquence en succession, où l'entrée d'horloge de la deuxième est la sortie $\overline{Q_1}$ de la première. La sortie Q_1 de la première bascule a une fréquence moitié de celle de l'entrée *D*. A son tour, l'entrée $\overline{Q_1}$ joue le rôle d'horloge pour la deuxième bascule et elle agit par son front montant, et divisant encore la fréquence par deux sur la sortie Q_2 de la deuxième bascule.



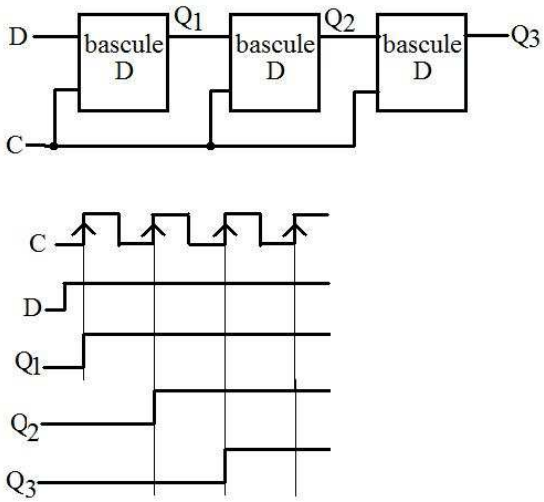
Finalement les deux sorties Q_2 Q_1 prennent les valeurs 00, 01, 10, 11, et cela de façon répétée. Le circuit se comporte comme un compteur modulo 4. Cela se généralise. En plaçant *n* bascules D en cascade, on obtient un compteur modulo 2^n. Un tel circuit est asynchrone, car s'il est synchronisé sur les impulsions de l'horloge pour la première bascule D, il ne l'est plus dans les suivantes, et de légers décalages en cascade peuvent finir par perturber le comptage.

Exercice : Construire un décompteur modulo 8. Par décompteur, on entend un circuit qui donne une succession décroissante de nombres, soit 111, 110, 101, 100, 011, 010, 001, 000, périodiquement répétée.

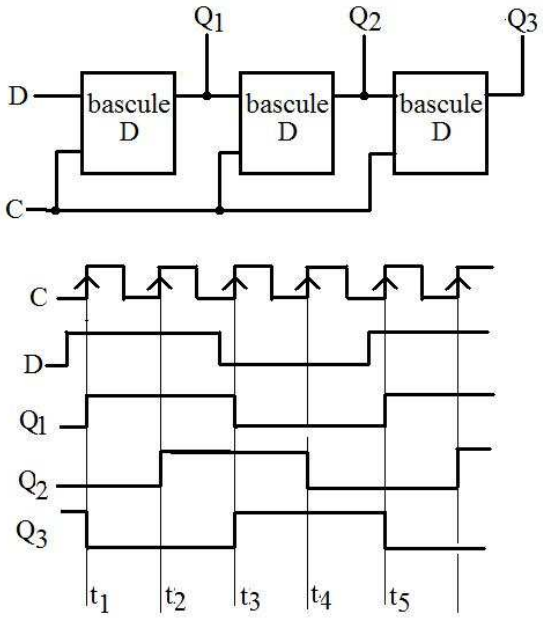
5-3) Registre à décalage

Plaçons plusieurs bascules D où chaque sortie est reliée à l'entrée de la suivante, avec le même signal d'horloge appliqué à toutes les bascules. On a alors un circuit synchrone. Supposons que la première entrée *D* et toutes les suivantes sont à 0. Puis faisons passer cette entrée *D* de la première bascule de 0 à 1 à un certain instant, celle-ci restant ensuite à 1 définitivement. Dès le premier front montant de l'horloge *C*, la sortie Q_1 de la première bascule passe à 1 et reste à 1 définitivement. Comme la sortie Q_1 devient l'entrée de la deuxième bascule, la sortie Q_2 de cette bascule passe à 1 définitivement dès le second front montant de l'horloge. A son tour la sortie Q_3 de la troisième bascule passe à 1 définitivement dès le troisième front d'horloge. Et ainsi de suite pour chaque bascule suivante, ce qui provoque finalement un retard pour la dernière sortie, par rapport à l'entrée

D initiale. Ce phénomène de temporisation est d'autant plus long qu'il y a plus de bascules en succession.



Exercice : Prendre comme précédemment trois bascules D en succession, et soumettons l'entrée *D* à un signal périodique, avec une période double de celle de l'horloge *C*. Les trois sorties des bascules sont aussi reliées à des lampes L_1, L_2, L_3 qui s'allument si la sortie vaut 1 et s'éteignent à 0. Que va-t-il se produire ?



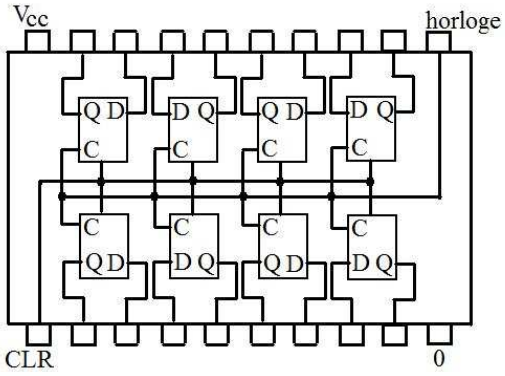
Après le premier passage à 1 de l'entrée *D*, la sortie Q_1 passe à 1 dès le premier front montant d'horloge, à l'instant t_1, puis repasse à 0 au troisième front t_3. A son tour, la sortie Q_2 de la deuxième bascule passe à 1 au deuxième front montant, à l'instant t_2, et retombe à 0 au quatrième t_4. Enfin la sortie Q_3 passe à 1 à l'instant t_3, tout cela se répétant ensuite de façon cyclique.

Ainsi, à l'instant t_1, la lampe L_1 s'allume, l'instant t_2, la lampe L_2 s'allume, à l'instant t_3, la lampe L_3 s'allume et L_1 s'éteint, à l'instant t_4, L_2 s'éteint, à l'instant t_5 L_1 s'allume et L_3 s'éteint, puis cela recommence.

5-4) Mémoire

Les bascules *flip flops* D, ainsi que d'autres types de *flip flops* que nous ne verrons pas ici, sont à la base de la constitution de certaines mémoires de l'ordinateur. Un premier modèle consiste à prendre plusieurs bascules réagissant au front montant d'un même signal d'horloge. Un tel modèle est synchrone. A chaque bascule est ajoutée une entrée *CLR* (*clear*). Mis à 0, ce signal *CLR* force la sortie *Q* de la bascule à 0. Un tel signal est notamment important lorsqu'il s'agit de choisir les conditions initiales de la sortie *Q*. Dans le cas présent, un seul signal *CLR* agit sur toutes les

bascules simultanément. On donne ci-dessous un schéma de circuit à huit bascules D, chacune ayant sa propre entrée *D* et sa propre sortie *Q*. Avec en plus l'horloge, le signal de commande *CLR* et l'alimentation (borne V_{cc} et 0), ce circuit de huit bits mémoires nécessite 20 broches.

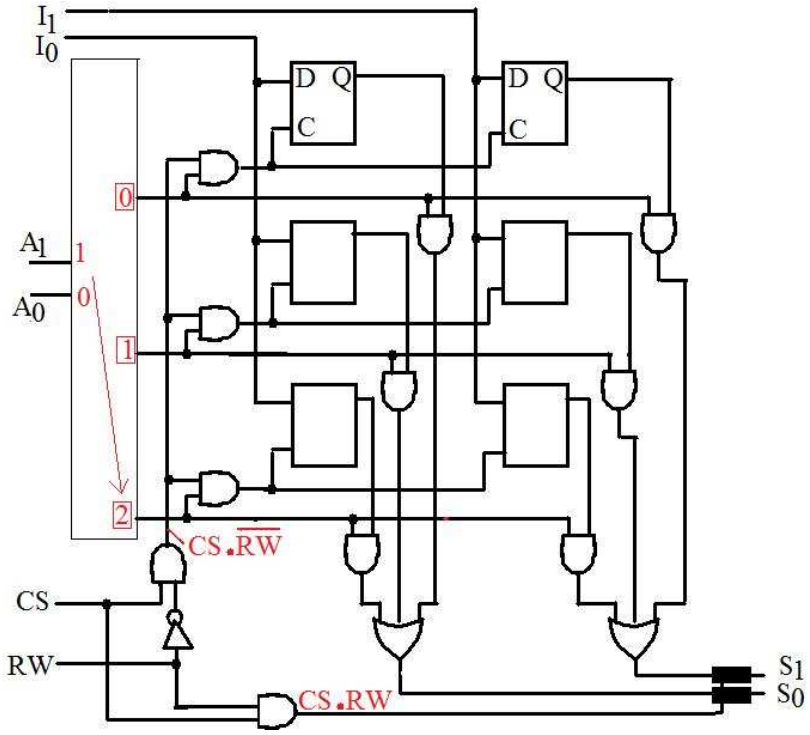


Il existe d'autres modèles de mémoires, de type rectangulaire ou lieu d'être linéaire comme le précédent, et par suite bien plus économes en nombre de broches. Maintenant la mémoire va être formée de mots mémoire de plusieurs bits chacun, ces mots ayant chacun une adresse. Nous avons dessiné ci-dessous un modèle très simplifié, organisé suivant un tableau rectangulaire de 3 lignes sur 2 colonnes, avec trois mots de deux bits chacun. Chaque élément élémentaire de mémoire de 1 bit est constituée par une bascule D. Précisons que les entrées *C* (d'horloge) de chaque bascule ne sont pas reliées à une horloge, le circuit étant dans le cas présent asynchrone. Les deux données arrivent par les entrées I_0 et I_1. L'adresse du mot mémoire concerné passe par les entrées A_0 et A_1 (ce qui permet d'obtenir quatre adresses, mais pour simplifier nous n'avons pris que trois mots). A cela s'ajoutent deux autres entrées : *CS* chargée de sélectionner ce circuit mémoire (car il peut en exister un grand nombre reliés les uns aux autres), et *RW* qui vaut 1 en cas de lecture et 0 en cas d'écriture dans la mémoire.[1]

Prenons le cas où $CS\ \overline{RW} = 1$: notre circuit a été sélectionné et on est en phase d'écriture. La donnée arrive par les fils A_0 et A_1. Son adresse est indiquée dans I_0 et I_1, et un décodeur se charge de mettre à 1 la ligne du mot ayant comme numéro en binaire $I_1\ I_0$ (sur le dessin c'est le mot numéro 2), toutes les autres lignes étant à 0. Une porte ET ayant en entrée la ligne de sélection concernée et l'ordre d'écriture fait sortir un 1 qui entre dans l'entrée d'horloge des deux bascules concernées. La donnée est ainsi mémorisée dans le mot d'adresse 2. Les autres mots ont au contraire leur entrée d'horloge mise à 0 et rien ne se passe, puis une nouvelle porte ET reliée au fil de sélection force un 0 en sortie. Cette partie finale du circuit n'a d'ailleurs pas d'importance dans cette phase d'écriture, puisque la donnée est bel et bien enregistrée dans son mot mémoire. Il suffit juste de placer ce que l'on appelle un *buffer* juste avant chacune des sorties S_0 et S_1. Ces *buffers* reçoivent comme signal de commande *CS* . *RW*, soit 0 dans le cas présent, ce qui signifie que chaque buffer agit comme un interrupteur ouvert, déconnectant les lignes de sortie de l'extérieur. Les données écrites en mémoire se font en zone protégée, sans rien envoyer à l'extérieur (ni un 0 ni un 1).

[1] En prenant un modèle un peu plus réaliste, avec quatre mots de trois bits chacun, on utilise 3 fils d'entrée pour les données, deux fils d'adresse, trois fils d'entrée de commande (CS, RW et OE (*output enable*) pour activer les sorties), plus trois fils de sortie, et deux fils pour l'alimentation, il suffit d'avoir un boîtier de 14 broches. Rappelons qu'avec le modèle présenté précédemment, avec ses huit bascules constituant chacune un mot mémoire d'un bit, il fallait vingt broches.

Prenons maintenant le cas d'une lecture, avec $CS\ \overline{RW} = 0$. Les fils d'écriture sont tous à 0, et le passage à travers les portes ET donne un 0 qui arrive sur toutes les entrées C des bascules. Aucun bascule n'est altérée pendant l'opération de lecture. Par contre c'est à la sortie *Q* des bascules que tout se passe. Le mot mémoire concerné a été sélectionné par le décodeur d'adresse. Le fil de sélection correspondant est à 1, ceux des autres mots restant à 0. Les deux portes ET du mot concerné font passer leurs bits au travers, puisque le fil de sélection est à 1, à la différence des autres mots, où les portes ET lâchent un 0. Les portes OU qui suivent laissent passer les bits du mot en cours de lecture, et ceux-ci arrivent dans les sorties S_0 et S_1, les buffers laissant passer le mot puisque dans le cas présent les signaux de commande des *buffers* sont à 1 (CS . RW = 1).



Référence bibliographique : Daniel Robert, *Electronique et informatique*, sur Internet.