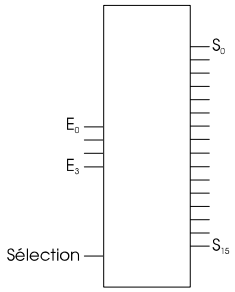
LES CIRCUITS LOGIQUES

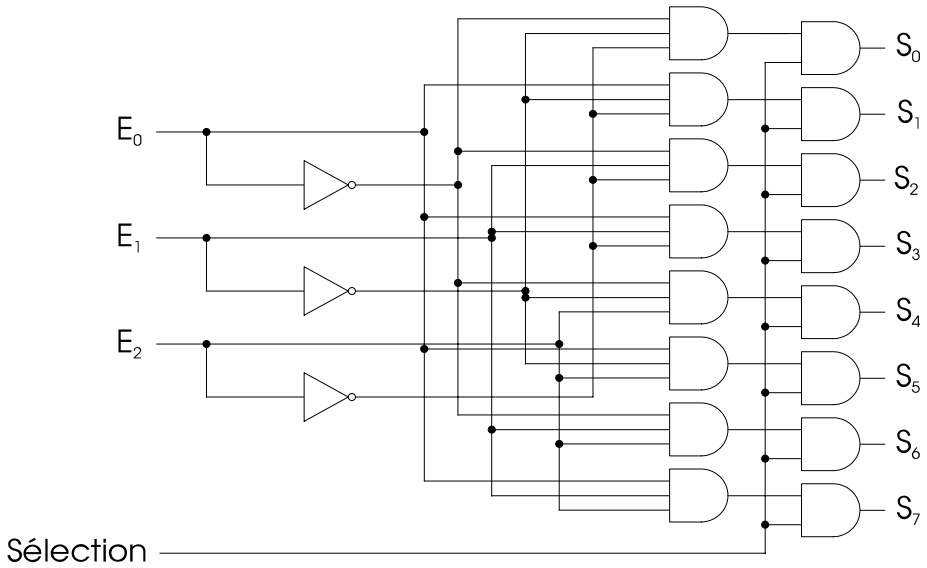
Ce chapitre explique comment les portes sont utilisées pour construire des circuits remplissant des fonctions logiques. Prenez patience : ces circuits seront utilisés par la suite pour former des ensembles "utiles".

DECODEUR

Reprenons notre joueur du chapitre "Les portes". Désormais, ce joyeux luron dispose de 16 drapeaux, numérotés de 0 à 15. On lui donne un nombre de 0 à 15. Son rôle est de lever le drapeau correspondant, en maintenant tous les autres baissés.
C'est ce que fait le décodeur, qui se présente fonctionnellement de la manière suivante :



Les 4 fils d'entrée E0-E3 représentent, suivant leurs tensions respectives (Vcc pour 1, 0V pour 0), un nombre sur 4 bits, codant donc une valeur de 0 à 15. Le rôle de la puce, quand elle est sélectionnée (c'est-à-dire que l'entrée *Sélection* est portée à la tension Vcc), consiste à ne mettre qu'un seul fil S_i à 1 (tous les autres étant à 0) : celui désigné par E. Quand le circuit n'est pas sélectionné, toutes ses sorties sont à 0.
Par exemple, si E = 1001, tous les fils S_0 à S_{15} seront à 0, sauf S_9 qui sera à 1 (1001b = 9d). Le câblage, limité ici à un décodeur 8 bits pour des raisons de lisibilité, est le suivant :



Son principe est le suivant :

- Pour chaque entrée E_i, on élabore $\overline{E_i}$ à partir d'un inverseur.
- Une première rangée de portes ET est installée, chaque porte ET étant connectée aux bons E_i ou $\overline{E_i}$, de manière à ce que chacune d'entre elles mette sa sortie à 1 pour une seule valeur possible de E, et que l'ensemble des portes ET couvrent les huit valeurs possibles de E. Par exemple, la troisième porte ET en

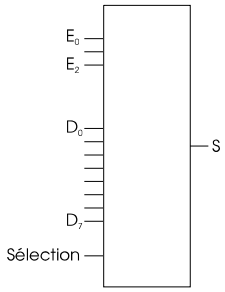
partant du haut aura sa sortie à 1 pour $\overline{E_2}E_1\overline{E_0}$ = 111, c'est-à-dire $E_2E_1E_0$ = 010. Ainsi, pour E=000, seule la première porte ET sera à 1, pour E=001 ce sera la deuxième, pour E=010 ce sera la troisième, etc… jusqu'à E=111.

- La deuxième rangée de portes ET consiste :
  - à mettre forcément à 0 les sorties si le circuit n'est pas sélectionné (0 ET X=0).
  - à transmettre l'état des portes ET de la première rangée si le circuit est sélectionné, c'est-à-dire à mettre la bonne sortie à 1 (1 ET X=X).

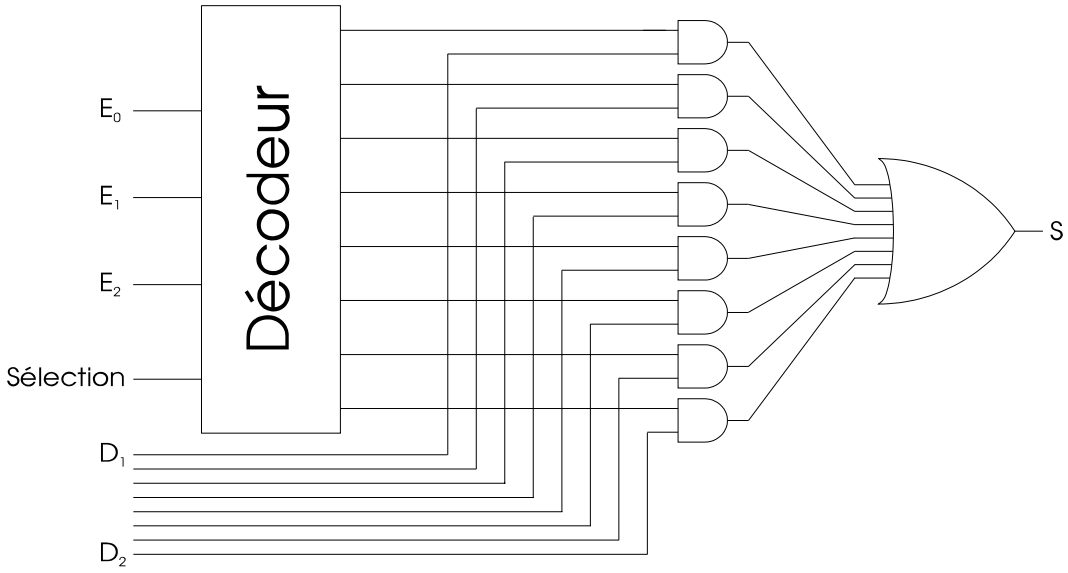
Ce circuit peut servir à sélectionner un composant parmi d'autres, tous étant affectés à un numéro d'ordre. Il existe bien sûr des décodeurs 4, 8, 16, 32... bits.

MULTIPLEXEUR

Notre joueur est maintenant standardiste avisé. Le standard téléphonique comporte 8 lignes. On demande au joueur un numéro de poste (entre 0 et 7) et il nous transfère la ligne (ce standard est un peu dépassé et ne sait transmettre qu'une ligne à la fois), c'est-à-dire que la sortie prend la valeur de la ligne sélectionnée, quelle que soient ses valeurs successives.
Ceci est le travail d'un multiplexeur, qui se présente fonctionnellement de la manière suivante (multiplexeur 8 bits) :



Quand le multiplexeur est sélectionné, la sortie S prend la valeur de l'entrée D_E indiquée par le nombre E codé par E0-E2. Par exemple, si E0-E2 = 101b = 5d, S = D_5 (que D_5 soit à 0 ou 1, constant ou variable).
Quand le multiplexeur n'est pas sélectionné, S = 0.
Le câblage utilise un décodeur :



Le décodeur permet de choisir la porte ET qui laissera passer l'information D_E voulue (1 ET X = X), les autres portes ET ne générant alors que des 0 (0 ET X = 0). Comme toutes les sorties des ET sont à 0 sauf éventuellement une, la sortie S prend la valeur de la porte ET sélectionnée (X OU 0 OU 0 OU 0… = X).

Ce circuit sert à choisir une valeur (portant sur 1 bit) parmi plusieurs disponibles. Pour choisir parmi plusieurs valeurs de plusieurs bits (par exemple, choisir un octet parmi plusieurs), il suffit d'utiliser des multiplexeurs en parallèle, en leur donnant à chacun le même numéro d'entrée E. Le premier multiplexeur s'occupera de choisir le bon bit0 de chacun des octets, le deuxième le bon bit1 de chacun des octets, etc… Le rassemblement physique des sorties des 8 multiplexeurs reconstituera l'octet choisi complet.

Il existe bien sûr des multiplexeurs 4 (E sur 2 bits, D sur 4 bits), 8 (E sur 3 bits, D sur 8 bits), 16 (E sur 4 bits, D sur 16 bits), 32 (E sur 5 bits, D sur 32 bits) bits…

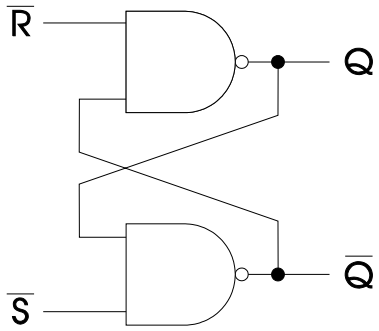
BASCULES

Notre joueur retrouve maintenant ses jetons : on fait défiler sous ses yeux une suite continue de jetons, certains étant blancs, les autres noirs. Là, on peut lui demander plusieurs choses :

Bascule D "latch"

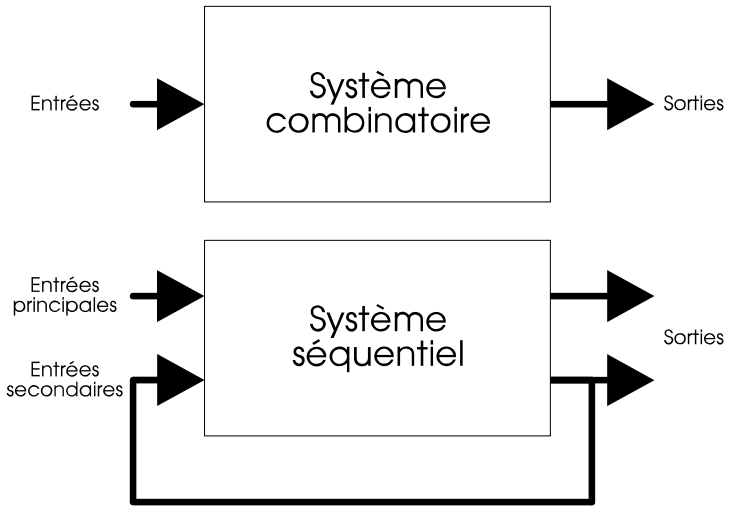
Il sort systématiquement le même jeton que celui qui passe devant lui. Mais quand on lève la main, il en reste au dernier jeton sorti, et ce aussi longtemps que notre main reste levée.

Bascule RS



$\overline{R}$	$\overline{S}$	Q_t	$\overline{Q}_t$
0	0		
1	0	0	1
0	1	1	0
1	1	Q_{t-1}	$\overline{Q}_{t-1}$

Q_{t-1} signifie : "valeur qu'avait Q à l'instant t-1" (par rapport à l'instant t, qui est maintenant), c'est-à-dire "valeur de Q avant la dernière modification des entrées".
Tous les circuits que nous avons vus jusqu'ici étaient combinatoires : leurs sorties étaient fonction de leurs entrées. Ce circuit est un circuit séquentiel : ses sorties sont fonction de ses entrées, mais également de ses sorties (car les valeurs de sortie sont reprises en entrée des portes).

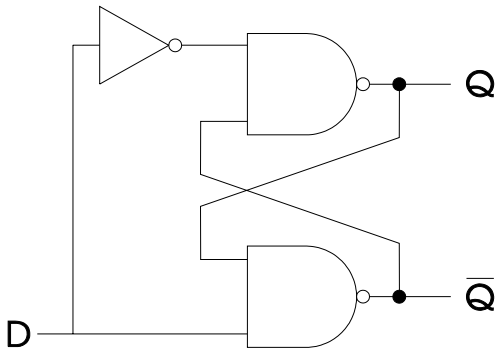


Le cas $\overline{S} = \overline{R} = 0$ ne nous intéresse pas, car il ne nous fourni pas deux sorties opposées. On cherchera donc à éviter cette configuration. Néanmoins, on voit que cette bascule présente l'intérêt de savoir conserver une donnée disparue :

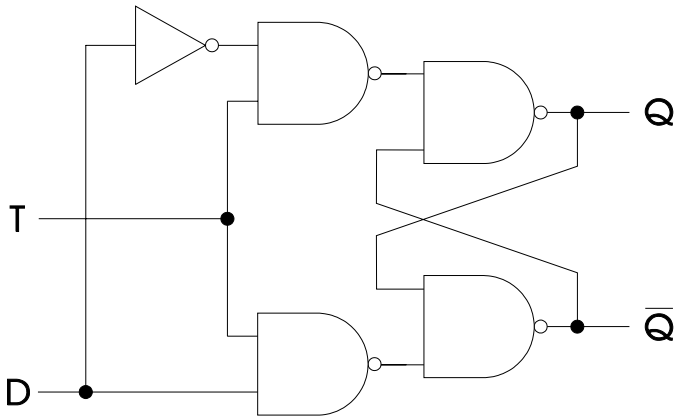
- lorsque soit $\overline{R}$, soit $\overline{S}$, est à 1, Q prend la valeur de $\overline{S}$
- puis lorsque tout revient à 1, Q conserve son ancienne valeur (le "top" du joueur).

Amélioration

Afin d'éviter l'état $\overline{S} = \overline{R} = 0$, nous modifions légèrement le schéma :



On a perdu par contre le cas intéressant : $\overline{R} = \overline{S} = 1$. Nous allons le recréer :



Ainsi, l'inverseur nous garantit de ne pas sombrer dans un état $\overline{S} = \overline{R} = 0$, et les portes NAND supplémentaires nous permettent de mémoriser l'état précédent :

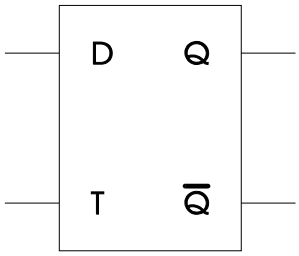
- quand T=0, les deux portes NET supplémentaires recréent le cas $\overline{S} = \overline{R} = 1$.

- quand T=1, les deux portes NAND supplémentaires laissent passer l'information D, moyennant inversion (mais cela n'est pas gênant, le circuit étant symétrique).

L'utilité de cette bascule, appelée *bascule D latch*, est donc la suivante :

- lorsque T est à 1, ~~Q~~ suit la valeur de D (D comme Donnée) : le joueur sort le même jeton que celui qui passe devant lui.
- lorsque T est à 0, ~~Q~~ conserve la dernière valeur de D, quelles que soient les nouvelles valeurs prises par D : nous avons levé la main.

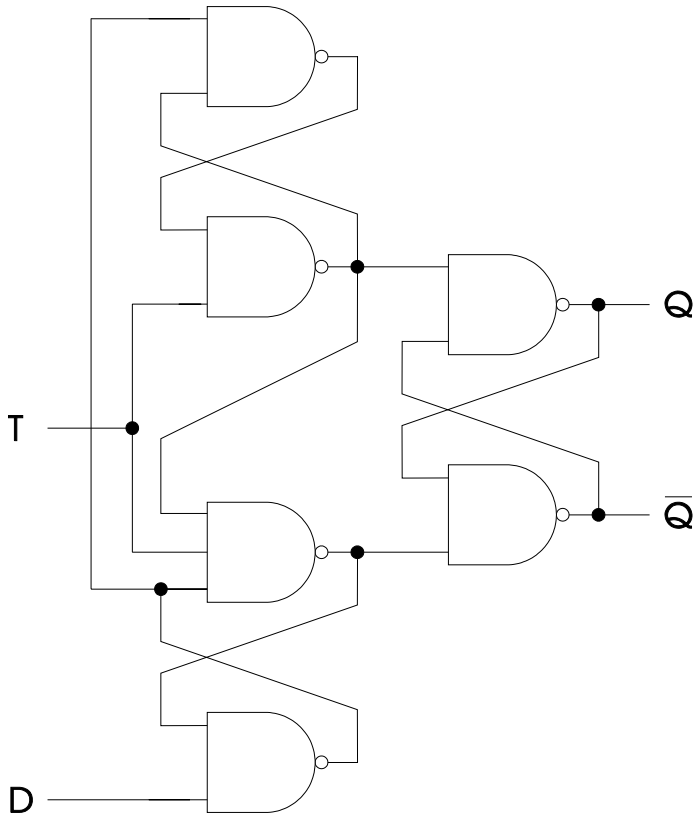
Sa symbolique est la suivante :



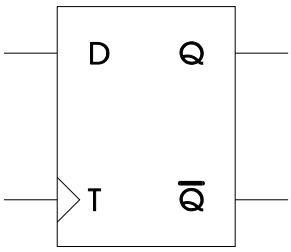
Bascule D "flip-flop"

Quand on lui dit "top", il sort un jeton de la même couleur que celui qui passait devant ses yeux au moment du top. Puis il attend le prochain "top".

Il est possible d'enrichir la bascule précédente : en effet, il se peut que nous ayons à enregistrer des valeurs de D dans des délais très brefs. Notamment, dans le cas ci-dessus, il ne faudrait pas que D varie avant que T ne revienne à 0, sinon la valeur mémorisée serait la dernière de D, hors nous voulons celle que D avait à l'instant ou T passait de 0 à 1. L'idéal serait que l'enregistrement de D dans Q ne s'effectue que quand T **passe** de 0 à 1 (c'est-à-dire sur un front montant de T), et que Q reste invariant sinon. C'est l'amélioration rendue par le montage ci-dessous, basé sur 3 bascules D "primitives" comme celle vue ci-dessus :



Ceci constitue ce qu'on appelle la *bascule D flip-flop*. Sa symbolique est la suivante :



Quelle que soit la bascule, l'entrée T (comme Trigger = déclencheur) peut s'appeler CLK (comme CLocK = horloge) ou encore H (comme Horloge). Une bascule (latch ou flip-flop) peut également comporter une entrée R (comme Reset = remise à zéro), qui force Q à 0 et $\overline{Q}$ à 1 (toujours de manière synchronisée avec T).

En montant huit de ces bascules en parallèle, on réalise une bascule permettant de traiter un octet entier.

Une bascule est un élément de mémoire. On peut ainsi l'utiliser dans des circuits séquentiels, ce qui nous permettra de réaliser des circuits évoluant dans le temps. En effet, pour qu'un circuit séquentiel ait un fonctionnement maîtrisé, ses entrées secondaires doivent pouvoir avoir le temps d'être traitées sans que ses sorties soient modifiées par ce traitement. Il faut donc temporiser la modification des sorties pendant ce temps : c'est ce que sait faire une bascule.

COMPTEUR#

HORLOGE

Notre joueur ne nous demande plus rien : avec la régularité d'un métronome, il présente un jeton blanc, puis un noir, puis un blanc, puis un noir, etc… en prenant soin de respecter une cadence rigoureusement constante.
L'horloge réalise le même travail avec des 0 et des 1 : sa sortie prend successivement les valeurs 0 et 1, à une fréquence définie par conception. L'ordre de grandeur de ces fréquences peut varier de quelques dizaines à quelques milliers de mégahertz (1 MHz = 1 mégahertz = 1 million d'allers-retours par seconde).
La construction de ce composant est simple, mais fait appel au phénomène de résonance en électronique analogique pour respecter une fréquence déterminée (à l'aide d'un quartz). Cet aspect demandant un long développement, pour une application limitée dans cet ouvrage à l'horloge, il ne fera pas l'objet de précisions particulière.

L'horloge est utilisée pour :

- synchroniser différents circuits : en sélectionnant pour deux circuits le signal d'une même horloge, leur fonctionnement est simultané
- cadencer des fonctionnements successifs de circuits
- compter le temps : connaissant la fréquence de l'horloge, on compte le temps en comptant ses changements de valeur.