Dans les chapitres précédents, nous avons vu comment fabriquer des circuits relativement généraux. Il est maintenant temps de voir quelques circuits relativement simples, très utilisés. Ces circuits simples sont utilisés pour construire des circuits plus complexes, comme des processeurs, des mémoires, et bien d'autres. Les prochains chapitres vont se concentrer exclusivement sur ces circuits simples, mais courants. Nous allons donner quelques exemples de circuits assez fréquents dans un ordinateur et voir comment construire ceux-ci avec des portes logiques. Dans ce chapitre, nous allons nous concentrer sur quelques circuits, que j'ai décidé de regrouper sous le nom de **circuits de sélection** Les circuits que nous allons présenter sont utilisés dans les mémoires, ainsi que dans certains circuits de calcul. Il est important de bien mémoriser ces circuits, ainsi que la procédure pour les concevoir : nous en aurons besoin dans la suite du cours. Ils sont au nombre de quatre : le décodeur, l'encodeur, le multiplexeur et le démultiplexeur.

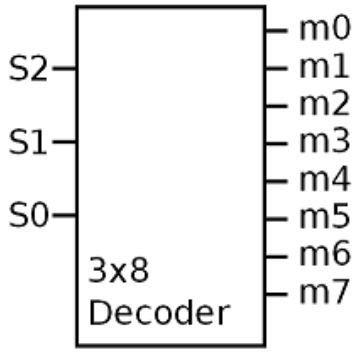
Décodeur

Nous allons Commencer avec le décodeur, un composant qui contient un grand nombre d'entrées et de sorties. Il y a une relation entre le nombre d'entrées et de sorties, qui tient à son fonctionnement : pour N entrées, un décodeur devra avoir 2^N sorties.

Interface

Pour résumer, un décodeur est un circuit :

- qui contient une entrée sur laquelle on envoie un nombre n de bits ;
- qui contient 2^n sorties ;
- où les sorties sont numérotées en partant de zéro ;
- où on ne peut sélectionner qu'une seule sortie à la fois : une seule sortie devra être placée à 1, et toutes les autres à zéro ;
- et où deux nombres d'entrée différents devront sélectionner des sorties différentes : la sortie de notre contrôleur qui sera mise à 1 sera différente pour deux nombres différents placés sur son entrée.

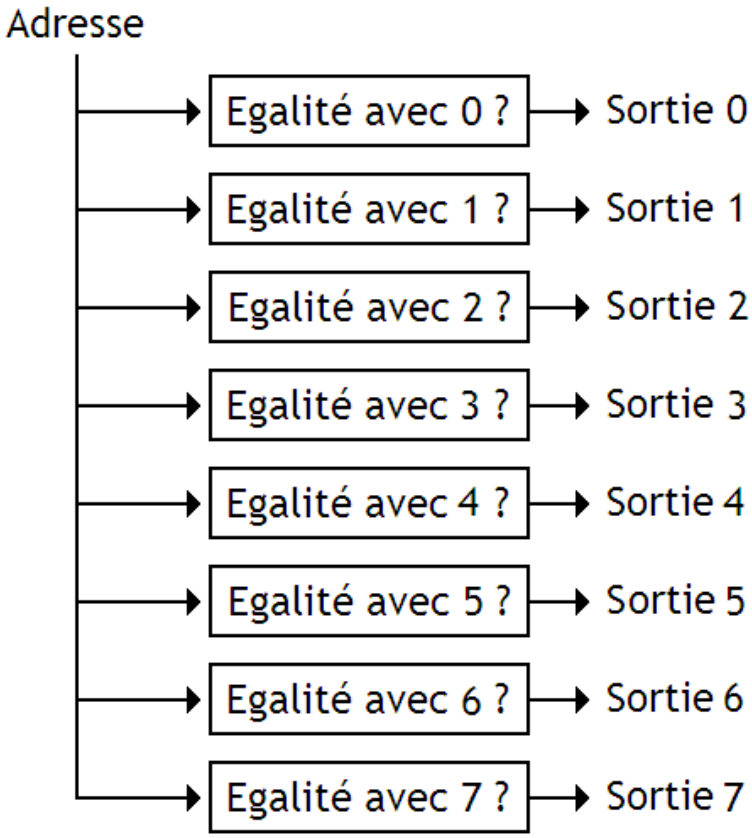


Le fonctionnement d'un décodeur est très simple :

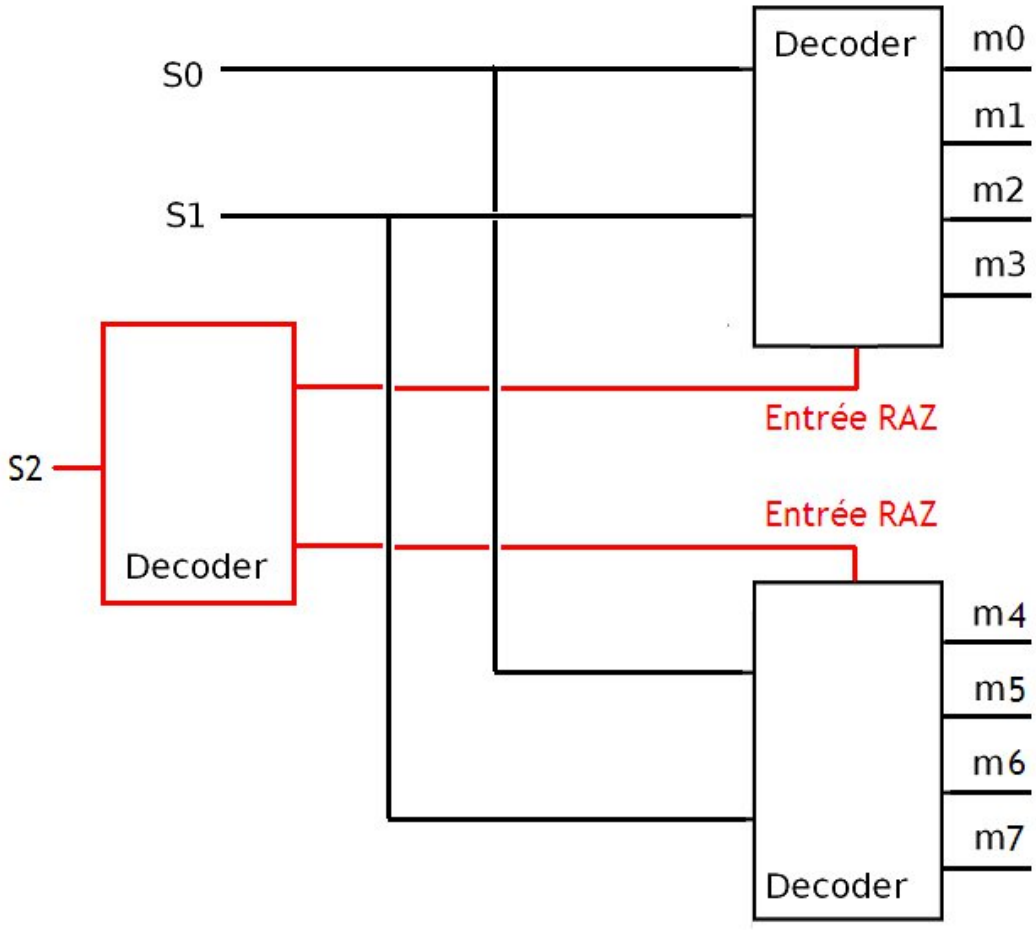
- il prend sur son entrée un nombre entier x codé en binaire ;
- positionne à 1 la sortie numérotée x ;
- et positionne à zéro toutes les autres sorties.

Conception interne

En réfléchissant bien, on sait qu'on peut déduire la sortie assez facilement en fonction de l'entrée : si l'entrée du décodeur vaut N, la sortie mise à 1 sera la sortie N. Bref, déduire quand mettre à 1 la sortie N est facile : il suffit de comparer l'entrée avec N. Si l'adresse vaut N, on envoie un 1 sur la sortie, et on envoie un zéro sinon. Pour cela, j'ai donc besoin d'un comparateur pour chaque sortie, et le tour est joué.

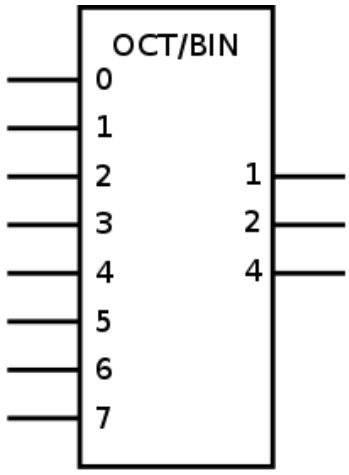


Comme autre méthode, on peut créer un décodeur en assemblant plusieurs décodeurs plus simples, nommés sous-décodeurs. Ces sous-décodeurs sont des décodeurs normaux, auxquels on a ajouté une entrée RAZ, qui permet de mettre à zéro toutes les sorties : si on met un 0 sur cette entrée, toutes les sorties passent à 0, alors que le décodeur fonctionne normalement sinon. Construire un décodeur demande suffisamment de sous-décodeurs pour combler toutes les sorties. Si on utilise des sous-décodeurs à n entrées, ceux-ci prendront en entrée les n bits de poids faible de l'entrée du décodeur que l'on souhaite construire (le décodeur final). Dans ces conditions, les n décodeurs auront une de leur sortie à 1. Pour que le décodeur final se comporte comme il faut, il faut désactiver tous les sous-décodeurs, sauf un avec l'entrée RAZ. Pour commander les n bits RAZ des sous-décodeurs, il suffit d'utiliser un décodeur qui est commandé par les bits de poids fort du décodeur final.

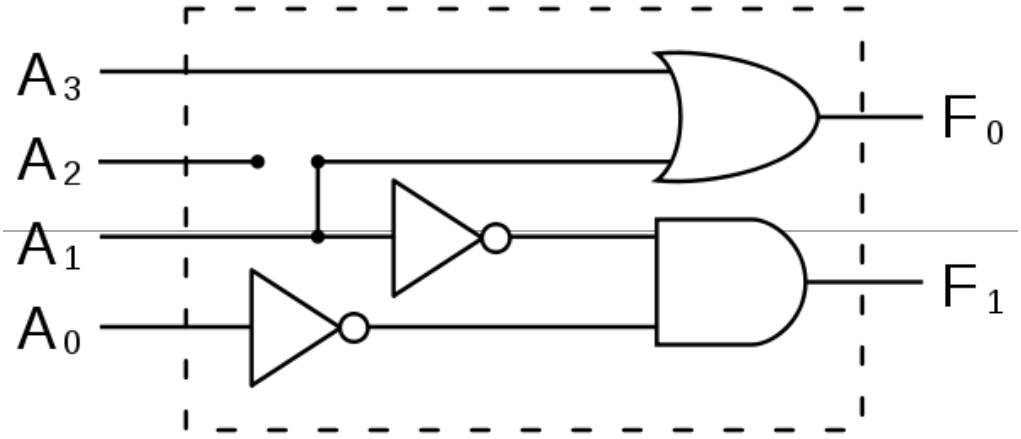


Encodeur

Il existe un circuit qui fait exactement l'inverse du décodeur : c'est l'**encodeur**. Ce circuit possède des entrées qui sont numérotées de 0 à N. En sortie, l'encodeur donne le numéro de l'entrée qui est à 1. Voici l'exemple d'un encodeur à 4 entrées et 2 sorties.

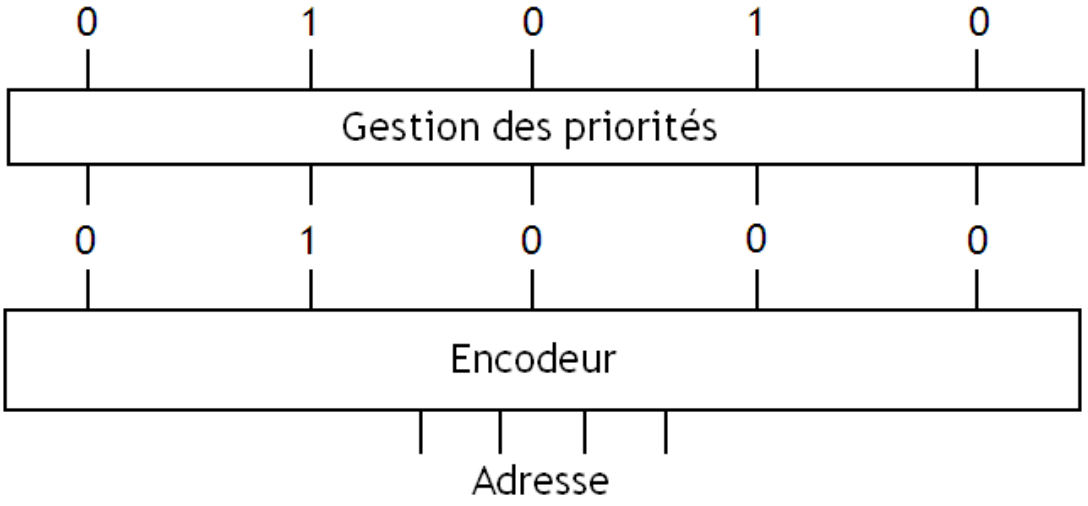


Encodeur à 8 entrées (et 3 sorties).

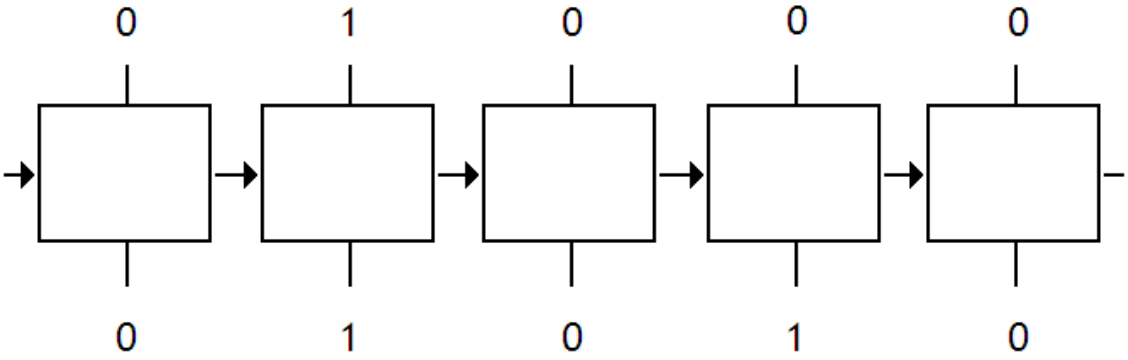


Encodeur à priorité

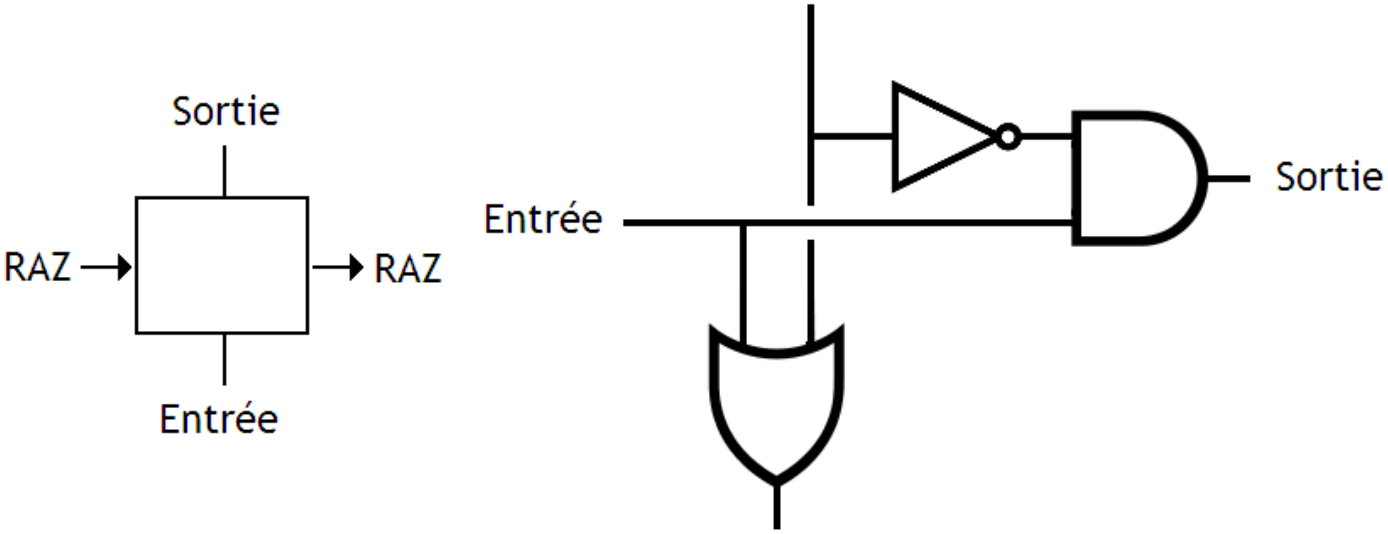
Si plusieurs entrées sont à 1, l'encodeur doit choisir une entrée. Dans la majorité des cas, l'encodeur est conçu pour choisir l'entrée qui a le numéro le plus petit : on parle alors d'**encodeur à priorité**. Fabriquer un tel priority encoder est assez simple : on peut tout simplement écrire sa table de vérité et en déduire le circuit directement. Mais il va de soit qu'effectuer de telles manipulations pour créer des circuits avec un grand nombre d'entrée n'est pas la meilleure des solutions. Il est possible de faire autrement. Première solution : on peut créer un encodeur à priorité en utilisant un encodeur normal, précédé d'un circuit qui se charge de sélectionner un seul des bits passé sur son entrée.



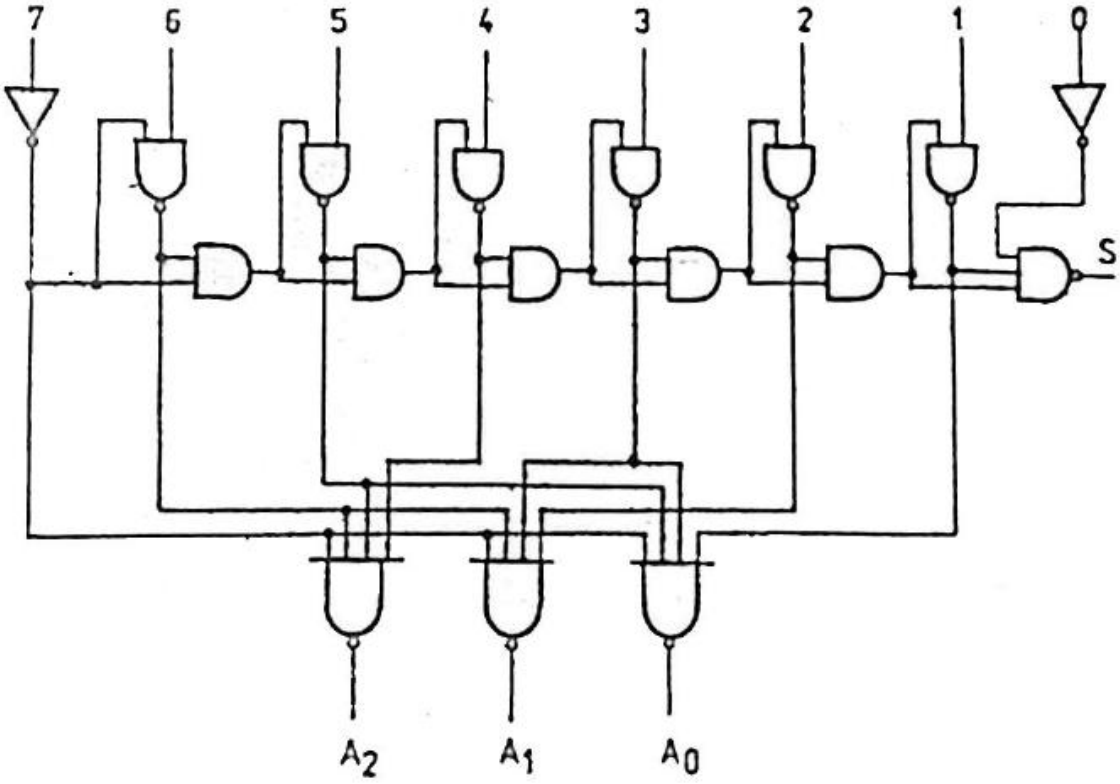
Le circuit de gestion des priorité est composé de petites briques de bases, reliées les unes à la suite des autres.



Ce circuit est basé sur un principe très simple : chaque brique de base va permettre de détecter le premier bit à 1 et va propager l'information aux briques de bases suivantes. Chaque brique de base va fonctionner comme suit : elle va regarder le signal d'entrée RAZ et va en déduire si elle doit recopier le bit passé en entrée ou le mettre à zéro. Si le bit RAZ vaut 1, alors la sortie sera mise à zéro automatiquement. Dans le cas contraire, le bit passé en entrée sera recopié. Si jamais le bit d'entrée vaut 1 ou que le signal d'entrée RAZ est à 1, alors le signal RAZ de sortie sera mit à 1. Ce circuit est constitué d'un paquet de Portes OU et NOR. Si vous cherchez à la concevoir à partir d'un table de vérité, vous obtiendrez ceci :

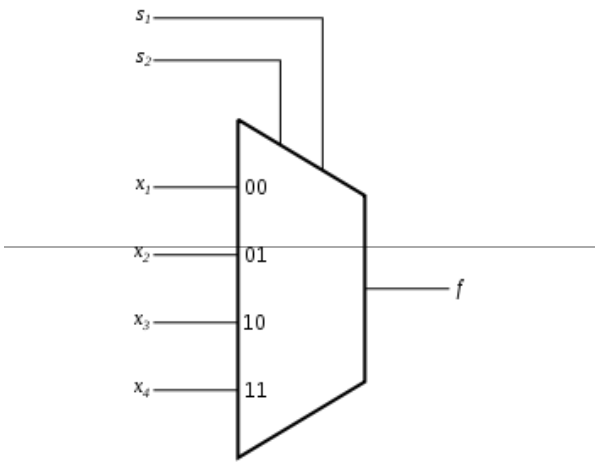


Le circuit complet d'un encodeur à priorité peut être déduit facilement à partir des raisonnements précédents. Après quelques simplifications, on peut obtenir le circuit suivant :

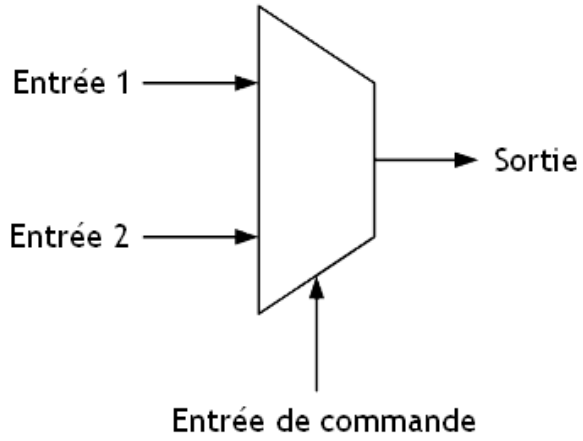


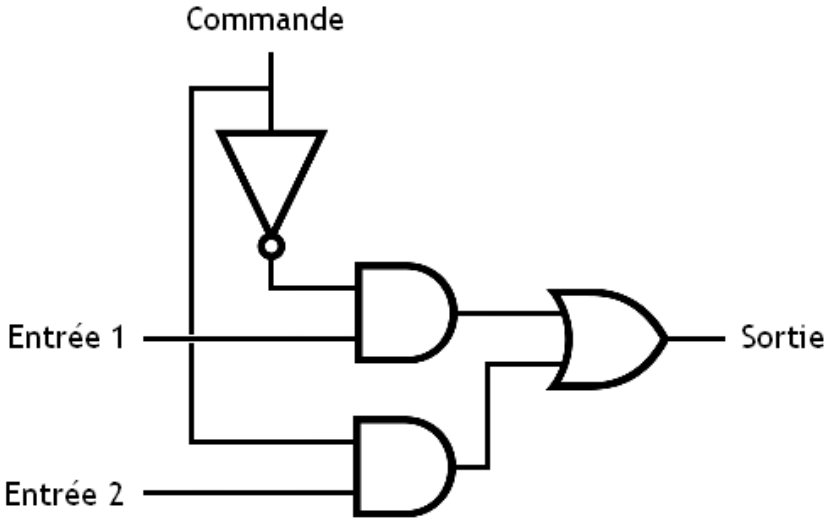
Multiplexeur

Les décodeurs ont des cousins : les multiplexeurs. Ces multiplexeurs sont des composants qui possèdent un nombre variable d'entrées et une sortie. Le rôle d'un multiplexeur est de recopier le contenu d'une des entrées sur sa sortie. Bien sûr, il faut bien choisir l'entrée qu'on veut recopier sur la sortie : pour cela, notre multiplexeur contient une entrée de commande qui permet de spécifier quelle entrée doit être recopiée.



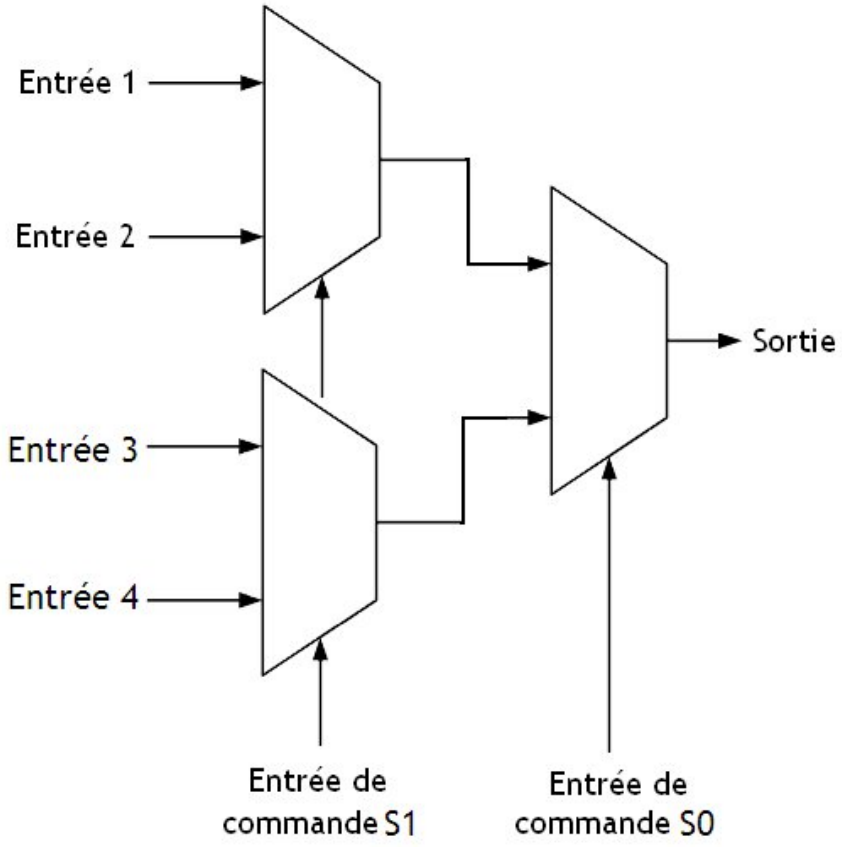
Le multiplexeur le plus simple est le multiplexeur à deux entrées et une sortie. Il est facile de le construire avec des portes logiques, dans les implémentations les plus simples. Sachez toutefois que les multiplexeurs utilisés dans nos ordinateurs ne sont pas forcément fabriqués avec des portes logiques. Ils sont fabriqués directement avec des transistors, afin de faire des économies.





Conception interne

On peut concevoir des multiplexeurs à plus de deux entrées en prenant deux multiplexeurs plus simples, et en ajoutant un multiplexeur 2 vers 1 sur leurs sorti respectives. Le multiplexeur final se contente de sélectionner une sortie parmi les deux sorties des multiplexeurs précédents, qui ont déjà effectué une sorte de présélection.

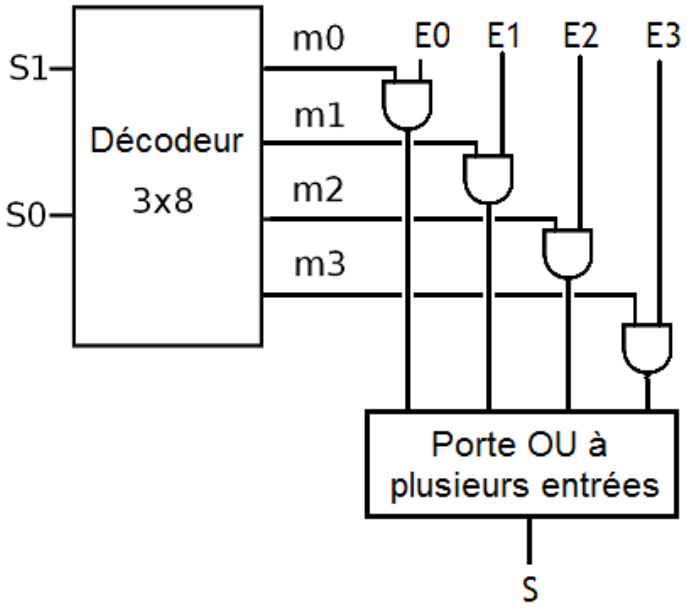


Il existe toutefois une manière bien plus simple pour créer des multiplexeurs : il suffit d'utiliser un décodeur, quelques portes OU, et quelques portes ET. L'idée est de :

- sélectionner l'entrée à recopier sur la sortie ;
- mettre les autres entrées à zéro ;
- faire un OU entre toutes les entrées : vu que toutes les entrées non-sélectionnées sont à zéro, la sortie de la porte OU aura la même valeu que l'entrée sélectionnée.

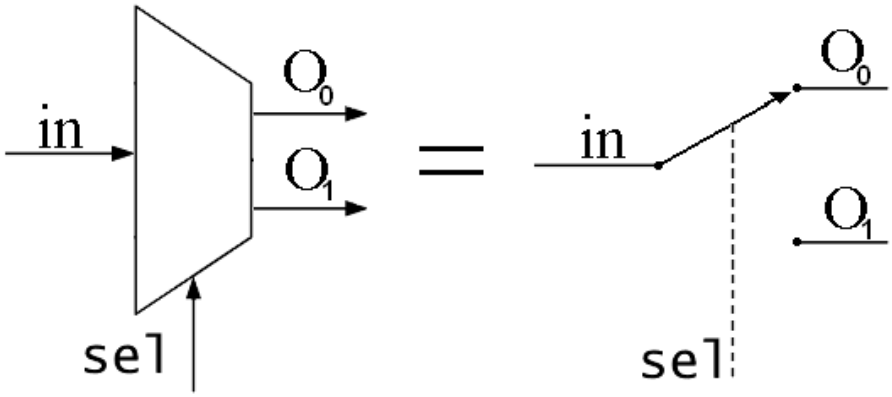
Pour sélectionner l'entrée adéquate du multiplexeur, on utilise un décodeur : si la sortie n du décodeur est à 1, alors l'entrée numéro n du multiplexeur sera recopiée sur sa sortie. Dans ces conditions, l'entrée de commande du multiplexeur correspond à l'entrée du décodeur. Pour mettre à zéro les entrées non-sélectionnées, on ajoute une porte ET par entrée : vu que a.0=0 et a.1=a, la porte ET

- recopie l'entrée du multiplexeur si le décodeur sort un 1 ;
- met à zéro l'entrée si le décodeur sort un 0.

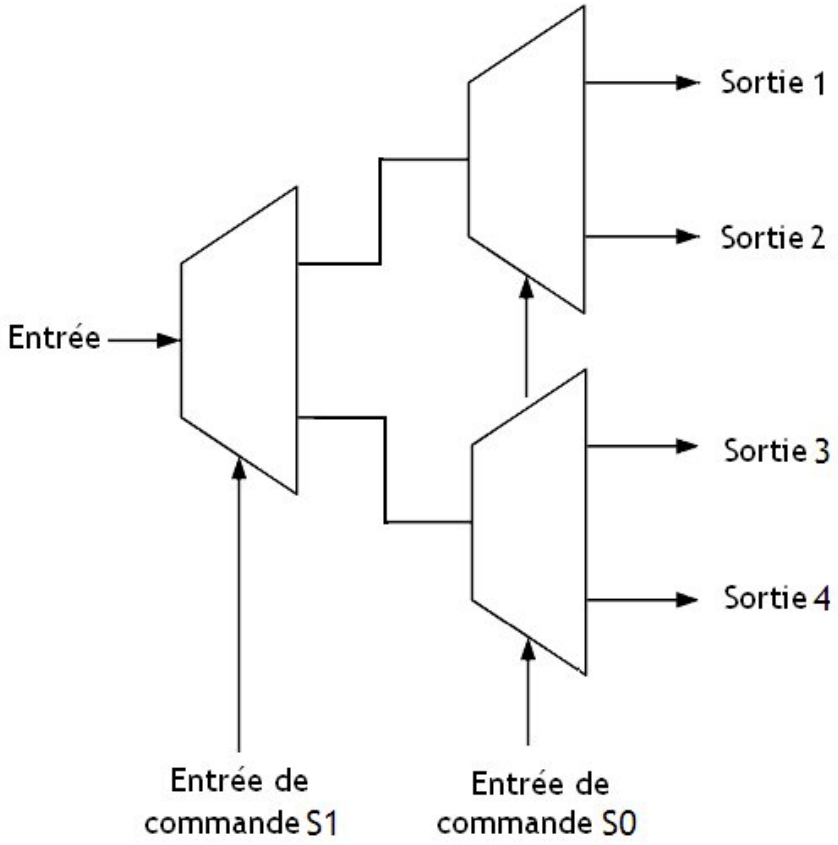


Démultiplexeur

Après avoir vu le multiplexeur, il est temps de voir de démultiplexeur. Comme le nom l'indique, le démultiplexeur fait l'exact inverse du multiplexeur. Son rôle est de recopier, sur une des sorties, ce qu'il y a sur l'entrée. Évidemment, la sortie sur laquelle on recopie l'entrée est choisie parmi toutes les entrées possibles. Pour cela, le démultiplexeur possède une entrée de commande.



Ce qui a été fait pour les multiplexeurs peut aussi s'adapter aux démultiplexeurs : il est possible de créer des démultiplexeurs en assemblant des démultiplexeurs 1 vers 2. Évidemment, le même principe s'applique à des démultiplexeurs plus complexes : il suffit de rajouter des couches.



Un démultiplexeur peut aussi se fabriquer en utilisant un décodeur et quelques portes ET. Pour résumer, un utilise un décodeur pour sélectionner la sortie sur laquelle recopier l'entrée. L'entrée doit alors : soit être recopiée si la sortie est sélectionnée, soit mise à zéro. Pour cela, on utilise une porte ET entre la sortie de sélection du décodeur et l'entrée.