TP3

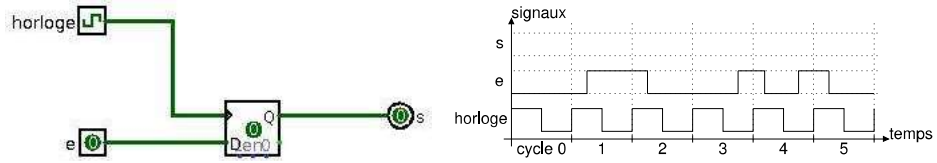
Le but de ce TP est de se familiariser avec certains circuits séquentiels, qui seront utilisés par la suite afin d'implanter dans Logisim une « calculatrice un peu programmable. » Dans le même but, vous devrez également fabriquer une unité arithmétique et logique (ALU, de l'autralien *Arithmetical and Logical Unit*).

3.1. Bascules et registres

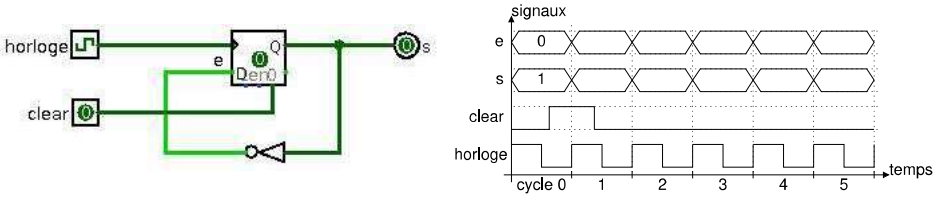
Exercice 3.1.1 : Bascules

Une bascule (composant `D Flip-flop` dans le répertoire `Memory` de la bibliothèque) est une mémoire 1 bit. On ne travaille qu'avec des bascules régies par le front montant de l'horloge : attribut `Trigger=Rising Edge`. Une bascule reçoit un signal d'horloge, et présente une entrée et une sortie : *l'entrée est mémorisée à la fin d'un cycle sur le front montant de l'horloge, et le bit mémorisé est maintenu sur la sortie pendant tout le cycle suivant.*

1) Commencez par brancher une bascule de Logisim comme indiquée ci-dessous. Servez-vous de ce circuit pour compléter le chronogramme fourni.



2) Expérimentez maintenant avec le circuit suivant. Notez que l'entrée `clear` de la bascule permet de la remettre à zéro à tout instan. Complétez le chronogramme.

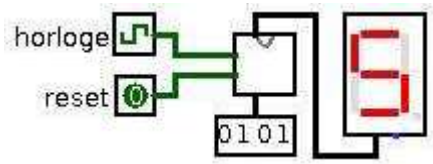


Exercice 3.1.2 : Registres

Un registre permet de mémoriser un mot binaire : par exemple, un registre 4 bits mémorise un mot de 4 bits. Les registres sont fabriqués à l'aide de bascules. Ainsi, *le mot placé en entrée du registre est mémorisée à la fin d'un cycle, et le mot mémorisé est maintenu sur la sortie pendant tout le cycle suivant.*

1) A l'aide de 4 bascules, implantez un circuit `reg4`, réalisant un registre 4 bits. Votre circuit présentera en entrée : un mot de 4 bits à mémoriser, le signal d'horloge, un signal `reset` permettant de remettre le contenu du registre à zéro. En sortie, votre registre présentera simplement le mot mémorisé sur 4 bits.

2) Dans le circuit principal, instanciez `reg4`, et testez-le dans le circuit représenté ci-dessous (le composant utilisé pour afficher le contenu du registre est un `Hex Digit Display` dans le répertoire `Input/Output`).



Exercice 3.1.3 : Compteurs

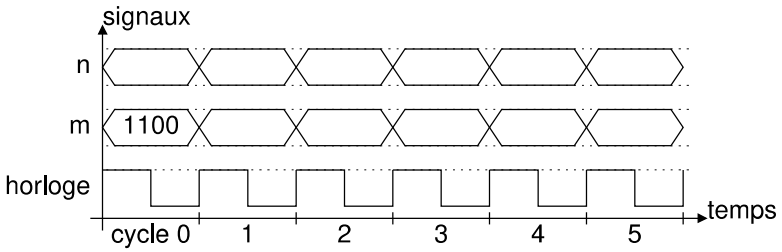
Un compteur est un circuit séquentiel qui reçoit un signal d'horloge, et un signal `reset` permettant de le remettre à zéro ; il présente une seule sortie sur k bits, formant un entier naturel n, qui est la valeur du compteur :

— le passage de `reset` à 1 remet la valeur n du compteur à 0,

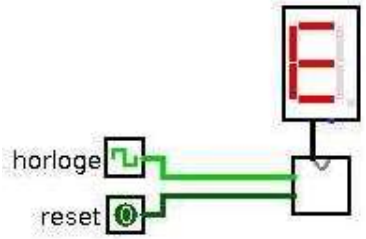
— quand `reset` = 0, la valeur du compteur est incrémentée à chaque fin de cycle d'horloge [1].

1) A l'aide de votre registre `reg4`, et d'un additionneur de la bibliothèque de Logisim, implantez le circuit d'un compteur 4 bits appelé `cmpt4`.

2) Dans le circuit `cmpt4`, on appelle m la valeur de l'entier en entrée de `reg4`, et n la valeur en sortie. Complétez le chronogramme suivant.



3) Testez votre circuit `cmpt4`, en l'utilisant dans un circuit ressemblant à celui ci-dessous.



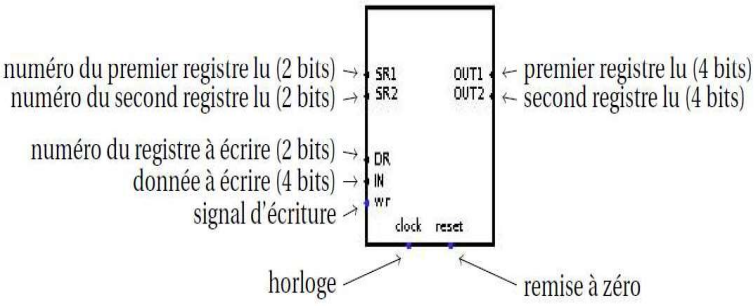
3.2. Banc de registres et mémoire

Un processeur doit lire et écrire des registres, et il est pratique de les regrouper dans un composant appelé *banc de registres* : ces registres recevront les résultats intermédiaires produits lors de exécution d'un programme.

Les registres sont rapides, mais assez coûteux [2] : les processeurs ne contiennent qu'un nombre réduit de registres. Par contre, ils peuvent utiliser une mémoire RAM (*Random Access Memory*) beaucoup plus grande et moins coûteuse, pour lire des programmes et des données, et écrire des résultats. Dans la suite, on s'intéresse par commodité à un autre type de mémoire, la mémoire ROM (*Read Only Memory*) qui ne peut qu'être lue.

Exercice 3.2.1 : Banc de registres

On étudie un un banc de 4 registres 4 bits, numérotés de 0 à 3 : ce banc peut lire deux registres et d'écrire un registre. Pour cela, il comporte les entrées `SR1`, `SR2`, `DR`, `IN` et le signal `wr`, et les ports de sortie `OUT1` et `OUT2`.



Le comportement du circuit est le suivant :
— le contenu du registre de numéro `SR1` est toujours présent sur la sortie `OUT1`,
— le contenu du registre de numéro `SR2` est toujours présent sur la sortie `OUT2`,
— si `wr` = 1, alors la donnée présente sur `IN` est écrite dans le registre numéro `DR` en fin de cycle,
— si `wr` = 0, alors aucun registre n'est écrit.

Téléchargez le fichier `regfile4.circ` qui contient un banc de 4 registres 4 bits (`regfile4`).

1. Il y a pleins d'autres subtilités, notamment ce qui arrive au compteur lorsqu'il atteint sa valeur maximale ; pour l'instant, on suppose que le compteur fonctionne modulo 2^k : si sur un cycle il est à $2^k - 1$, au cycle suivant le compteur retombe à 0.
2. On parle ici du coût monétaire par bit d'information

1) Dans le circuit `regfile4`, comment le contenu des registres à lire sont ils envoyés sur `OUT1` et `OUT2` ?
2) En cas d'écriture, comment la donnée présente sur `IN` est elle envoyée dans le registre numéro `DR` ?
3) Réalisez dans `main` un circuit pour tester `regfile4`.
4) Que se passe-t-il si on écrit et on lit dans le même registre en même temps ?

Exercice 3.2.2 : Mémoire ROM

Une mémoire ROM de 2^k mots de n bits est un tableau comportant 2^k cases contenant chacune un mot k bits. Chaque case est identifiée par son adresse, qui est son indice dans le tableau. Le circuit présente :
— une entrée `A` sur k bits, qui est l'adresse de la case que l'on veut lire,
— une sortie `D` sur n bits, qui est la donnée contenue dans la case d'adresse `A`.
Une ROM peut être utilisée pour contenir un programme, ou bien des données fixées une fois pour toute.

Dans Logisim, le composant `ROM` se trouve dans le répertoire `Memory` de la bibliothèque. Vous pouvez modifier son contenu de différentes façon (voir l'aide), mais un circuit ne peut que la lire, pas y écrire.

Téléchargez le fichier `chaine.circ`, et ouvrez : le circuit contient une mémoire ROM de 32 mots de 7 bits. Chaque case de cette mémoire contient le code ASCII d'un caractère, sur 7 bits. Le circuit comporte aussi un écran, qui affichera les caractères que vous lui enverrez sur son entrée à chaque cycle d'horloge.

1) Utilisez un compteur [3] de façon à parcourir le contenu de la ROM, et à placer successivement sur sa sortie chacun des caractères : à chaque cycle d'horloge, un caractère est lu sur la sortie `D` de la ROM.
2) Modifiez le circuit pour qu'à chaque cycle le caractère lu soit affiché sur l'écran.
3) Complétez le circuit pour que le bouton `reset` remette (aussi) le compteur à zéro.
4) Faites s'afficher tout le message philosophique contenu dans la mémoire en actionnant le signal d'horloge. Quel message est produit ?

3.3. Une unité arithmétique et logique

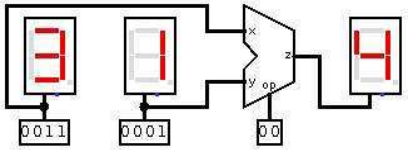
On veut mettre au point une unité arithmétique et logique (ALU) pour notre calculatrice : il s'agit d'un circuit qui prend en entrée deux entiers `x` et `y` codés sur n bits, et produit en sortie un résultat `z`, également sur n bits. Un port d'entrée supplémentaire `op` sur k bits permet de sélectionner l'opération dont on veut obtenir le résultat : k bits permettent de choisir parmi 2^k opérations.

Exercice 3.3.1 : ALU 4 bits

Dans cet exercice, les opérandes en entrée de l'ALU sont sur 4 bits ($n = 4$), et on souhaite qu'elle puisse effectuer les 4 opérations usuelles sur les entiers naturels : addition, soustraction, multiplication et division. L'opération à effectuer est déterminée par le codage suivant :

op	z
00	x + y
01	x − y
10	x × y
11	x ÷ y

1) Implantez un circuit `alu4`, réalisant l'ALU demandée. Pour les opérations, utilisez les composants proposés dans le répertoire `Arithmetic` de Logsim. En outre, vous utiliserez un composant `Multiplexer` judicieusement configuré pour sélectionner le bon résultat en sortie de l'ALU.



2) Testez votre composant `alu4` dans un circuit ayant l'allure de celui ci-dessous [4].

3. Utilisez un compteur de la bibliothèque (`Counter` dans le répertoire `Memory` ; voir l'aide de Logisim pour le configurer).
4. Votre ALU aura la forme d'un rectangle, ce qui est très bien ! Dans le circuit représenté ici, l'auteur s'est « amusé » à donner une forme un peu standard à son ALU : ne perdez pas de temps avec ça !