Université de Bordeaux
Licence STS

ARCHITECTURE DES ORDINATEURS

TP : **07**

Construction d'une mémoire

L'outil que vous allez utiliser est le simulateur de circuits SimcirJS, dont une copie adaptée au cours est installée sur :

`http://dept-info.labri.fr/ENSEIGNEMENT/archi/circuits/blank.html`

(ce lien est disponible depuis le site du cours).

Exercice 1 : Bascule SR

La bascule SR est le moyen le plus élémentaire de stocker une information dans un circuit numérique. Elle consiste à « reboucler » les sorties de deux portes `NOR` l'une sur l'autre.

N.B. : En cas de doute, vous pouvez consulter les diapositives du cours pour retrouver le câblage de la bascule SR.

Question 1

Construisez une bascule SR au moyen de deux portes `NOR` (utilisez des « plots » pour que le câblage soit propre). Placez une `LED` sur la sortie `Q`, et deux `PushOn` sur chacune des entrées `R` (« *Reset* ») et `S` (« *Set* »). Vérifiez que le circuit mémorise bien la dernière action sur l'un des `PushOn`, y compris lorsqu'on appuie plusieurs fois sur le même.

Question 2

La bascule SR est instable lorsqu'on remet à zéro simultanément les entrées `R` et `S`. Pour le vérifier, supprimez l'un des `PushOn`, branchez les deux entrées `R` et `S` sur la sortie du bouton restant, et constatez ce qui se passe lorsque les deux entrées repassent à zéro exactement en même temps.

N.B. :

1. Pour que cet l'effet d'instabilité apparaisse, il faut induire un délai dans la propagation des signaux le long des boucles entre les deux portes NOR. Pour cela, utilisez des « plots » au sein de vos boucles ; ils ont l'intérêt supplémentaire de rendre votre câblage plus élégant.
2. Afin de ne pas consommer trop de CPU du serveur, arrêtez rapidement la simulation de votre circuit en déconnectant l'une des sorties des portes `NOR`.
3. Dans la « vraie vie », le circuit retombera rapidement dans un état stable, du fait des imperfections physiques lors de la fabrication (piste plus longue et/ou lente à traverser sur le silicium du fait des défauts de structure atomique, etc.). Cela dépendra de chaque circuit, et n'est pas prévisible sur le plan théorique.

Exercice 2 : Bascule D

La bascule D vise à corriger l'instabilité de la bascule SR, en garantissant que les entrées `R` et `S` ne reviendront jamais ensemble à zéro. Pour cela, les deux entrées seront toujours l'inverse l'une de l'autre.

Question 1

Transformez votre bascule SR en bascule D. Afin de la tester, branchez un `Toggle` sur l'entrée `D` de la bascule, et un `PushOn` sur l'entrée `Clk`, pour déclencher la mémorisation. Testez le comportement de votre circuit.

Question 2

En dessous de votre circuit, construisez-en un nouveau, qui utilise le composant `D`. Vérifiez que le fonctionnement des deux circuits est identique.

Exercice 3 : Registre

Un registre est un composant mémoire servant à stocker un ensemble de n valeurs binaires, pour former un « mot » de n bits. Ici, on prendra $n = 4$.

Question 1

Assemblez en parallèle quatre composants `D` pour former un registre permettant de stocker un mot de 4 bits entré sous la forme d'un bus (il vous faudra donc un `BusIn` pour séparer les 4 valeurs binaires à stocker dans chaque composant `D`). La valeur à mémoriser sera donnée au moyen d'un `RotaryEncoderBus` et affichée au moyen d'un `4bit7segBus`. La mémorisation s'effectuera toujours au moyen d'un `PushOn` qui activera l'entrée `Clk`.

Question 2

Au dessous du registre que vous avez créé « à la main », positionnez un composant `RegBus`, et rajoutez les composants vous permettant d'obtenir le même comportement que celui de votre circuit.

Question 3

Le composant `RegBus-E` diffère du composant `RegBus` par la présence d'une entrée supplémentaire `E`, pour « *enable* » : le registre ne mémorisera la valeur d'entrée que si cette entrée est à 1.

Remplacez votre circuit `RegBus` par un `RegBus-E`. L'entrée `Clk` pourra donc être branchée directement sur un oscillateur, et c'est l'entrée `E` qui sera branchée en sortie d'un `PushOn`. Vous obtenez un circuit qui « capture » à intervalles de temps régulier une valeur.

C'est un peu ce qui se produit dans le cadre du fonctionnement d'un processeur, où le registre de destination d'une instruction sera modifié à la fin de chaque instruction, au moment indiqué par l'horloge.

Exercice 4 : Mémoire

Une mémoire est un ensemble de cellules (ou « mots », qui sont ici des registres) dont la cellule actuellement active (en lecture et/ou en écriture) est indiquée par son adresse.
Dans cet exercice, il s'agit de construire une mémoire à 4 mots de 4 bits. Ces 4 mots sont donc numérotés sur 2 bits, de 00 à 11. Pour faciliter le paramétrage de l'adresse, au lieu de la coder avec un ensemble de `Toggle` représentant chacun un bit de l'adresse, on pourra utiliser en entrée un `RotaryEncoderBus`, dont seuls les deux bits de poids faible seront utilisés.

Fonctionnellement, votre circuit mémoire doit donc disposer des entrées et sorties suivantes :
- une sortie de données, sur 4 bits, fournissant le contenu de la cellule mémoire actuellement active, c'est-à-dire celle dont l'adresse est donnée en entrée ;
- une entrée d'adresse, sur 2 bits (pour des raisons de commodité, elle sera sur 4, mais seuls les deux bits de poids faible seront effectivement utilisés), codant l'adresse de la cellule mémoire actuellement active ;

— une entrée `Clk`, sur un bit, devant recevoir le signal d'horloge qui alimentera les registres ;
— une entrée de données, sur 4 bits, qui contiendra la valeur devant éventuellement être écrite dans la cellule mémoire active ;
— une entrée `W`, sur un bit, qui, lorsqu'elle sera activée, provoquera au prochain top d'horloge l'écriture de la valeur de donnée d'entrée dans la cellule mémoire actuellement active.

Le travail à réaliser est le suivant :
— Placez quatre `RegBus-E` au milieu de votre plan de travail, l'un au dessus de l'autre. Autour d'eux, à bonne distance, placez autant de plots (simples ou « x4 », selon les cas) que votre circuit mémoire possède d'entrées et de sorties. Les entrées seront à gauche, et la sortie à droite.
— Reliez la borne extérieure de chacun des plots aux composants nécessaires pour tester le fonctionnement de votre circuit mémoire :
  — la sortie, à un afficheur `4bit7segBus` ;
  — l'entrée d'horloge, à un `OSC` ;
  — l'entrée d'adresse, à un `RotaryEncoderBus` (dont seules les positions 0 à 3 seront donc pertinentes) ;
  — l'entrée de données, à un second `RotaryEncoderBus` ;
  — l'entrée d'activation d'écriture, à un `PushOn`.
— Au moyen d'un composant `MuxBus`, faites en sorte que le contenu de la cellule mémoire actuellement active soit celui fourni en sortie.
— Au moyen d'un composant `Demux`, faites en sorte que seule la cellule mémoire active puisse recevoir le signal d'activation d'écriture.
— Fournissez aux registres la donnée d'entrée et le signal d'horloge.
— Testez votre circuit, en écrivant des valeurs différentes dans chacune des cellules, et en les relisant, les re-modifiant, etc.