ARCHITECTURE DES ORDINATEURS

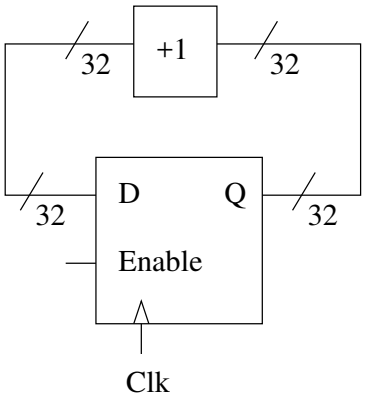
TD : **11**

Processeur complet

L'objectif de ce TD est d'enrichir un circuit simple jusqu'à obtenir un processeur complet.

Rappels

Repartons d'une variante du circuit de départ du TD précédent :



Ce n'est pas dessiné (pour simplifier), mais le schéma sous-entend un empilement de 32 bascules D, prenant chacune une des 32 entrées D et une des 32 sorties Q, tandis que les entrées *enable* et *Clk* sont répliquées sur toutes les bascules. L'entrée *enable* s'entend ainsi qu'elle a été construite au TD précédent : l'enregistrement de la valeur de D n'a lieu que si *enable* est à 1, la valeur précédente est conservée sinon.

En considérant le circuit entier, à quoi servirait un bouton que l'on mettrait sur l'entrée *enable* ?

Exercice 1 : Registres

Notre processeur n'a pour l'instant qu'un seul registre.

Ajoutez 2 autres bascules D pour avoir 3 registres en tout, et des multiplexeurs/démultiplexeurs/-codeurs/décodeurs, dont les entrées de contrôles *src* et *dst*, sur 2 bits, permettent ainsi de choisir quel registre est utilisé comme source de calcul, et quel registre est utilisé comme destination de calcul. Attention à ne pas écrire dans plusieurs registres à la fois !

Exercice 2 : Opérateur binaire

On veut pouvoir utiliser des opérateurs binaires tels que l'addition, ce qui nécessite 2 opérandes sources. Remplacez le module `+1` par un module `add`, et ajoutez ce qu'il faut pour apporter le deuxième opérande, *dst* étant réutilisé pour indiquer la deuxième opérande source.

Exercice 3 : ALU

Nous voulons que notre processeur effectue des calculs plus intéressants que seulement l'addition.

Adjoignez au module `add` un autre module `sub`, et ajoutez ce qu'il faut pour avoir une entrée de contrôle *op* permettant de choisir quel opération doit être faite.

Supposons que ces circuits de calcul possèdent une entrée *enable* qui permettent de choisir si le circuit fait réellement le calcul (et donc consomme de l'énergie). Comment en profiter ?

Exercice 4 : Et PC dans tout ça ?

Pour l'instant les entrées de contrôle *src*, *dst*, *op* sont des paramètres de notre circuit, à spécifier à la main. Il faudrait que cela soit automatique.

Ajoutez un module de mémoire prenant une adresse en entrée, et fournissant en sortie des valeurs *src*, *dst*, *op*, que vous pouvez donc brancher sur le reste de votre circuit.

Ajoutez un petit bout de circuit contenant un registre `PC` incrémenté de 6 à chaque cycle, et branchez sa valeur au module mémoire.

Nous avons ainsi un micro-processeur !

Pour aller plus loin...

Exercice 5 : Chargement de constante

Ajoutez au module mémoire une sortie *valC*.

Ajoutez une opération de chargement de constante dans un registre.

Exercice 6 : Branchement inconditionnel

Ajoutez une instruction de branchement inconditionnel : PC doit prendre la valeur `valC`, et les autres registres ne pas être modifés.

Exercice 7 : Branchement conditionnel

Ajoutez un registre de flags en sortie des modules de calcul.

Ajoutez un circuit `Bch` prenant en entrée la valeur du registre de flags et une entrée de commande sur 3 bits choisissant entre les tests *e, ne, l, le, g ge*, et indique en sortie si le branchement doit être pris.

Généraliser la boucle de branchement pour supporter le branchement conditionnel : PC doit prendre la valeur `valC` si et seulement si le circuit `Bch` retourne 1.