julien.braine@ens-lyon.fr,mateusz.skomra@ens-lyon.fr, florent.de-dinechin@insa-lyon.fr

ASR1, TD/TP : architecture de notre processeur, sur papier pour l'instant

Dans ce TD on prend une grande feuille de papier (A4 horizontal) et on va dessiner le plan de masse de notre processeur. Son chemin de données, quoi. Enfin bref l'équivalent des figures du chapitre 9 du poly.

Pour cette partie en papier, nous allons avoir besoin de gommer plusieurs fois : crayon à papier ou stylo gommable

Par ailleurs, plutot que tirer des fils dans tous les sens, utilisez massivement la convention que deux fils nommés pareil sont implicitement connectés (en Digital, on avait les tunnels).

Ensuite on attaquera éventuellement la réalisation en VHDL.

Exercice 1: Ce qui est pareil que d'hab :

A droite de votre feuille, placez l'interface avec la mémoire : MDI, MDO, MA, MW (memory write, le "enable" de la mémoire). Notez les tailles de tous ces fils (combien de bits).

Placez un registre de PC, un additionneur devant, et un multiplexeur devant l'additionneur pour ajouter soit 1, soit une autre valeur qui nous permettra de faire des sauts relatifs (instructions `j`, `jt` ou `jf`. Appelez le signal de contrôle de ce multiplexeur `jump`.

Si à ce stade vous ne savez pas d'ou va pouvoir sortir ce signal, c'est que la réponse est : de l'automate. (cette phrase sera à ressortir pour chaque signal – de chargement de registre, de commande de multiplexeur, etc. – dont vous ne savez pas d'ou il sort).

Tirez tous les fils, et y compris le fil qui va vers la mémoire. Mettez toutes les tailles dessus.

Quand vous avez fini, dites à voix haute "longue vie au PC" (le TDMan compte les exclamations et envoie le dernier qui n'a pas parlé au tableau).

Exercice 2: Ce qui est pareil que d'hab, 2 :

Dans le quart gauche inférieur de votre feuille, dessinez deux blocs :

— la boîte à registres,

— l'ALU.

Placez leurs entrées/sorties, en tout cas celles qu'il faut pour implémenter les instructions `add` et `sub` (attention, on a du code à 2 opérandes). Tirez les fils que vous pouvez.

Quand vous avez fini, dites à voix haute "c'est l'ALU de mon cœur" (le TDMan compte les exclamations et envoie le dernier qui n'a pas parlé au tableau).

Exercice 3: Dessinons l'intérieur de l'ALU : (vous avez droit à un additionneur, un décaleur, etc, autant de multiplexeurs que vous voulez, et autant de fonctions combinatoires à moins de 8 entrées que vous voulez).

En 10 mn car on rentrera dans le détail quand on écrira le VHDL.

Exercice 4: Cela se complique un peu :

Ce qui ne va pas être comme pour le processeur RISC vu en cours, c'est qu'une instruction n'est pas toujours en un mot mais parfois (le plus souvent) en plusieurs.

Le plus simple est de prévoir un certain nombre de registres de 4 bits que nous appellerons `RI0`, `RI1`, ... `RI`n (pour une certaine valeur de n à déterminer). Dessinez-les bien serrés en haut de votre dessin (là où il y a RI sur la fig. 9.2 du poly). Tirez les fils qui rentrent dans ces registres. Leur sortie sera (par abus de langage) appelée `RI0`, `RI1`, etc Donnez à chacun de ces registres un *load enable* que nous appellerons loadRI0, loadRI1, etc. D'où sortent ces signaux?

Quand vous avez fini, dites à voix haute "Sans nul doute, de l'automate"

C'est le moment de dessiner la boîte noire de l'automate dans un coin de votre feuille, avec ses entrées et ses sorties, en tout cas toutes celles qu'on a vu jusque là.

Exercice 5: Dessinons l'automate : OK, il est temps de se convaincre qu'on va savoir construire l'automate. Prenez une seconde feuille A4 horizontale (il est déconseillé d'utiliser le verso de la première).

Dessinez l'état initial, que nous appellerons *Fetch*. Son rôle est d'envoyer le PC vers le bus d'adresse. À la transition d'horloge qui termine cet état, la mémoire aura positionné le premier mioche d'instruction sur le bus de donnée, et il s'agira d'enregistrer cette information dans RI0. Quels sont donc les sorties de l'automate à activer dans cet état? Quelles sont les états suivants possibles?

Ensuite, on dispose du premier mioche d'instruction dans RI0, et l'automate va pouvoir décider si c'est tout pour cette instruction (`ret`), ou s'il faut encore charger deux mioches (`j`, `jt`, `jf`, `add`, `sub`, `copy`) ou trois (`not`, `and`, `or`, `xor`). Dessinez soigneusement toutes les patates correspondantes à ces trois cas de figure, avec les sorties de l'automate à activer.

Il ne reste plus qu'à dessiner les états de l'automate qui vont déclencher l'exécution de chaque instruction, et revenir au début du cycle de von Neumann (en vérifiant maniaquement que le PC a bien avancé du bon nombre de mioches). Faites-le pour `add`. Quand vous en êtes là, dites à voix haute "moi j'aime les spaghetti".

Là il y a un choix à faire. On peut faire un chemin dans l'automate pour chaque instruction différente : un pour `add`, un pour `sub`, ... On sera tranquille. Mais `add` et `sub` vont suivre des trajectoires tellement similaires dans l'automate qu'il vaut mieux les regrouper : dès lors que `RI0` entre dans l'ALU, celle-ci saura quelle instruction exécuter. Donc il est de bon ton de regrouper les instructions "similaires". Et là vient la question : `j` et `add` sont-ils similaires ? (ils sont tous deux encodés en trois mioches).

La réponse conservative est : dans le doute, faisons deux chemins. En effet, le rôle des états d'exécution est différent dans les deux cas : pour `add/sub`, il faut positionner le *enable* de la boite à registres ; pour `j` il faut positionner le multiplexeur sur l'entrée du PC. Toutefois il y a une réponse agressive qui est : toutes les instructions qui prennent le même nombre de cycle peuvent être fusionnées en un chemin de l'automate. En effet, la différence sur les signaux à positionner dépendra de l'instruction (`RI0` et `RI1`), qui est aussi disponible dans le chemin de données : chaque signal de contrôle peut être calculé par une fonction combinatoire dont les entrées sont l'état, `RI0` et `RI1`, et qu'on peut placer dans le chemin de donnée si cela nous chante.

Si vous avez compris, il y a sans doute un juste milieu que je vous laisse chercher avant que l'on ne vous le donne.

Completez maniaquement le chemin de données et l'automate sur le papier pour faire juste marcher `add`, `sub`, `xor`, et `j`.

Dites à voix haute "On ne va pas en vendre beaucoup".

Exercice 6: On n'a qu'un drapeau, autant l'implémenter : Ajoutez le registre 1 bit avec son enable, et ce qu'il faut pour faire tomber en marche `isequal`, `jt` et `jf`. A nouveau l'intérêt de cette question est de se poser la question suivante : est-ce que le test fait par `jt` se traduit en un branchement dans le dessin de patates, ou bien en quelques portes dans le chemin de donnée qui vont agir directement sur les signaux de contrôle sans que le chemin pris par l'automate ne change ?

Dites à voix haute "Chouette je vais faire du VHDL" et partez finir le TD7 en essayant de vous rapprocher de la construction du processeur.