Examen d'architecture des ordinateurs

1IN ENSEEIHT

26/01/2016

1. Arithmétique binaire (2 pts)

- sur 8 bits, effectuer la somme en binaire des 2 nombres hexadécimaux suivants : 0x7B et 0x43
- Interpréter ce calcul en termes d'arithmétique signée et d'arithmétique non signée. On précisera notamment les valeurs de C et V et on les expliquera

2. Opposé arithmétique (2 pts)

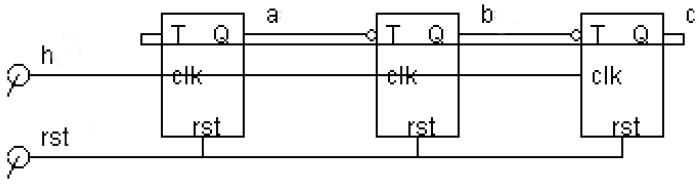
Poly.copié page 38 : « … l'opposé peut se calculer à partir de l'original de la façon suivante : le bit de poids faible est recopié tel quel ; les autres bits sont inversés seulement s'il existe un bit à 1 sur leur droite dans l'original. »

Concevoir un circuit *combinatoire* qui implémente cet algorithme, sous forme d'un module SHDL d'interface :

```
module oppose(e[3..0] : s[3..0])
```

3. Analyse d'un circuit séquentiel (5 pts)

On considère le circuit suivant :



a) Décrire ce circuit sous forme d'un module SHDL (1 pt)
b) Ce circuit est-il de type MOORE ou MEALY ? Justifier. Quels sont ses vecteurs d'entrées, d'état, de sorties ? (1 pt)
c) Enumérer tous les états possibles de ce circuit, puis donner une assignation possible et la table de transition associée (1 pt)
d) A partir de cette table de transitions, concevoir un circuit équivalent avec un plus petit nombre de bascules. On fournira un schéma ou un module SHDL (2 pts)

4. Synthèse d'un compteur (4 pts)

Concevoir un compteur / décompteur avec remise à zéro synchrone, d'interface suivant :

```
module count4ZD(rst, clk, sclr, down : s[3..0])
```

Lorsque `sclr`=1, le compteur se remet à zéro *au front d'horloge* (remise à zéro *synchrone*)

Le compteur compte dans l'ordre croissant si `down`=0, dans l'ordre décroissant si `down`=1

`sclr` est prioritaire sur `down`, c'est à dire que :

- `sclr`=1 et `down`=* : le compteur se remet à zéro au front d'horloge
- `sclr`=0 et `down`=0 : le compteur s'incrémente au front d'horloge
- `sclr`=0 et `down`=1 : le compteur se décrémente au front d'horloge

On fournira au choix un schéma ou le code SHDL correspondant.

5. Séquencement d'une instruction (2 pt)

On souhaite ajouter une instruction `cpmem [%rs1+%rs2], [%rd1+%rd2]`, qui permet de copier le contenu mémoire à l'adresse = %rs1+%rs2 dans l'adresse = %rd1+%rd2. Elle a pour code :

```
011.00000000.rs1.rs2.rd1.rd2
```

rs1, rs2, rd1, rd2 sont des champs de 5 bits qui représentent des numéros de registres. Par exemple, l'instruction `cpmem [%r1+%r2],[%r3+%r4]`, de code 011.00000000.00001.00010.00011.00100, copie le contenu de la case mémoire d'adresse %r1+%r2 dans la case mémoire d'adresse %r3+%r4

Dessiner le graphe de séquencement de cette instruction (étapes élémentaires nécessaires à son exécution) en indiquant l'opération réalisée à chaque transition sous forme textuelle.

Indiquer pour une transition au choix (sauf la première et la dernière) le détail des commandes de la micromachine nécessaires à l'exécution de l'opération correspondante.

6. Programmation de CRAPS (5 pts)

- Écrire un sous-programme `incr` qui incrémente les 2 chiffres d'un nombre décimal stocké dans (%r1-dizaine, %r2-unité). Ainsi à chaque appel, les valeurs de (%r1, %r2) seront successivement :
  (0, 0), (0, 1), …, (0, 9), (1, 0), (1, 1), …, (1, 9), (2, 0),…, …, (9, 9), (0, 0), etc.
  Ce sous-programme ne devra modifier aucun registre
- Écrire un sous-programme `aff` qui affiche sur les 2 afficheurs 7-segments de droite la valeur décimale codée dans (%r1, %r2). On supposera les afficheurs initialisés ; ce sous-programme ne devra modifier aucun registre
- Écrire un sous-programme `delay` qui attend durant environ 1s ; ce sous-programme ne devra modifier aucun registre
- Écrire un programme principal qui ne termine jamais et qui à chaque cycle répète les opérations suivantes :
  - Affichage du compteur et délai
  - Incrémentation du compteur

  On effectuera toutes les initialisations nécessaires.