ENSEEIHT - $1^{ère}$ année Sciences du numérique
Contrôle d'architecture des ordinateurs - 11 janvier 2021
Durée : 1 heure - Tous documents autorisés

Comparaison d'entiers

En adoptant un raisonnement récursif, écrire un module SHDL *sup*3 qui prend en entrée trois entiers non signés A, B et C codés sur 8 bits et qui génère en sortie $min(A, B, C)$, le plus petit des trois. On pourra utiliser plusieurs instances du module ucmp8 écrit en TP, qui prend en entrée deux entiers non signé A et B codés sur 8 bits et retourne *sup* et *eq*. *sup* vaut 1 si $A > B$ et 0 sinon. *eq* vaut 1 si $A = B$ et 0 sinon.

Décompteur

On souhaite mettre en œuvre un décompteur. La figure 1 montre les signaux d'entrée et de sortie du circuit.

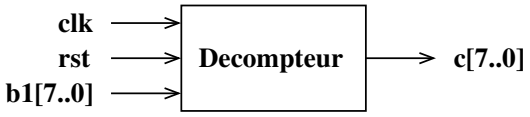

Figure 1: Circuit décompteur

Le circuit fonctionne de la manière suivante :

- l'entrée rst (reset) permet de mettre la sortie $c[7..0]$ à 0,
- à chaque front montant de l'horloge *clk*
  - si $init = 0$ et $decount = 0$, la valeur de la sortie est inchangée,
  - si $init = 1$, $c[7..0] \leftarrow b1[7..0]$
  - si $init = 0$ et $decount = 1$, la valeur de la sortie est décrémentée, i.e. $c[7..0] \leftarrow c[7..0] - 1$. Lorsque $c[7..0] = 00000000$, on obtient $c[7..0] = 11111111$

1. Pour le décomptage, donner la condition d'inversion du bit c[0], la condition d'inversion du bit c[1], la condition d'inversion du bit c[2]. En déduire la condition d'inversion du bit c[i], et écrire les quations shdl de la partie décomptage.
2. Compléter les équations précédentes avec la partie initialisation.
3. Expliquer comment votre solution laisse c[7..0] inchangé lorsque init=0 et decount=0.

Implantation d'un tri à bulles d'un tableau

Il existe de nombreux algorithmes pour trier un tableau d'entiers. En travaux pratiques, nous avons implanté l'algorithme de tri par sélections/échanges. Nous allons maintenant implanté l'algorithme de tri à bulles d'un tableau T de n éléments. L'algorithme est le suivant.

```
Pour i de n à 2 pas -1 faire
    Pour j de 1 à i-1 pas 1 faire
        Si T[j] > T[j+1] alors
            permuter (T[j],T[j+1]);
        Finsi;
    Finpour;
Finpour;
```

Cet algorithme peut être modélisé par la machine de Mealy de la figure 2. Sur ce graphe, les conditions pour franchir les transitions sont préfixées par *C:* et les actions à exécuter en franchissant les transitions sont préfixées par *A:*.

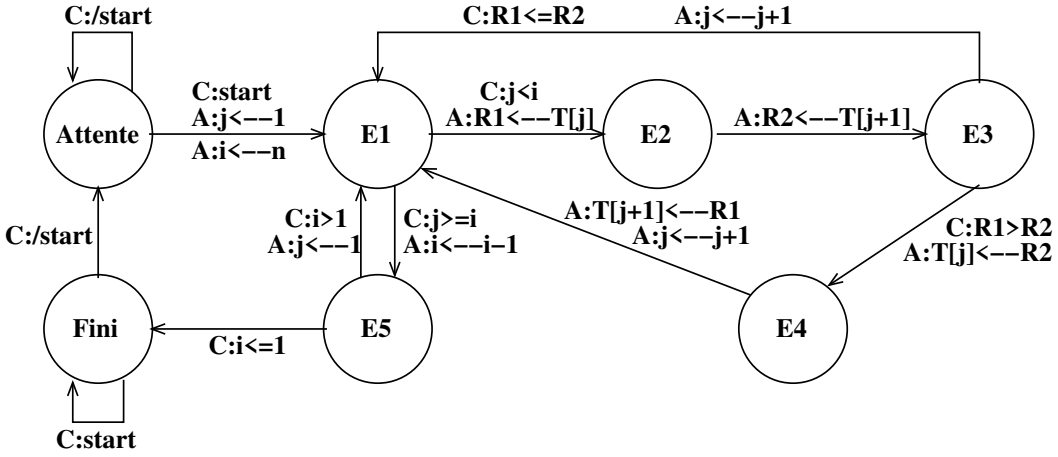

Figure 2: Machine de Mealy pour le tri à bulles

Nous allons implanter uniquement le sous-graphe contenant les tats E1, E2, E3 et E4, en restant dans l'état E1 lorsque $j >= i$.
On considèrera que les modules suivants existent (ils ne sont pas à écrire) :

- count8_b1(rst, clk, count, init, b1[7..0]: c[7..0]) : compteur modulo 256, initialisé à b1[7..0] lorsque $init = 1$, incrémenté lorsque $count = 1$,
- decount8_b1(rst, clk, decount, init, b1[7..0]: c[7..0]) : le décompteur écrit à l'exercice précédent,
- inc8(x[7..0],y[7..0]) : y[7..0]) = x[7..0]+1 à tout instant,
- ucmp32(a[31..0],b[31..0]:sup,eq) : comparateur non signé sur 32 bits écrit en TP,
- ucmp8(a[31..0],b[31..0]:sup,eq) : comparateur non signé sur 8 bits écrit en TP.

On utilisera une mémoire RAM, module prédéfini, qui a pour interface:

$ram_aread_swrite(clk, write, ad[7..0], data_in[31..0] : data_out[31..0])

et qui fonctionne de la façon suivante :

- lecture asynchrone : la sortie data_out indique à tout moment le contenu du mot mémoire présent ladresse ad,
- écriture synchrone: la donnée data_in est enregsitrée dans le mot mémoire à ladresse ad sur le front dhorloge clk lorsque write=1.

Ecrire le module permutations qui implante le sous-graphe contenant les tats E1, E2, E3 et E4.

module permutations (rst, clk, start, n[7..0] : E1, E2, E3, E4, JsupeqI)
// JsupeqI =1 lorsque j¿=i

Le tableau est stocké à partir de l'adresse 1.