N7_SN_1A

Architecture des ordinateurs - Semestre 5

TP3 - Circuits séquentiels - Algorithmes cablés

Le travail fait dans ce TP servira pour le BE-TP4. Bien lire le sujet de ce BE à la fin de ce TP.

Rappel du TD3 : Nous nous intéressons à l'algorithme de tri par sélection, et nous allons essayer de l'implanter dans un circuit logique. Pour cela nous commencerons par réaliser et tester le circuit qui calcule le max et l'indice du max d'un tableau.

1- Calcul de Max et de Ad_Max de Tab(Ad1..Ad2)

Commençons par adapter l'algorithme de manière à le rendre plus proche des notions manipulées dans les couches basses : adresse, registres, etc. L'accès à un élément du tableau se fait par son adresse en mémoire, et on utilisera donc Tab(adresse) au lieu de Tab(i).

```
Ad_courante <- Ad1
Max <- Tab(Ad_courante)
Ad_Max <- Ad_courante
Ad_courante <- Ad_courante + 1
Tant Que Ad_Couante <= Ad2 Faire
    Si Tab(Ad_courante) > Max Alors
        Max <- Tab(Ad_Courante)
        Ad_Max <- Ad_courante
    Fin Si
    Ad_courante <- Ad_courante + 1
Fin TantQue
```

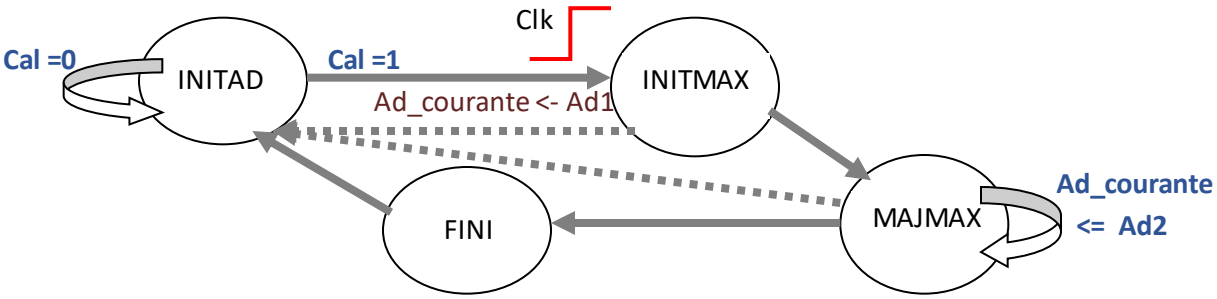
Analysons cet algorithme et listons les éléments dont nous avons besoin :

- Pour stocker le tableau, nous utiliserons une mémoire RAM avec des adresses codées sur 8 bits (256 mots), et des mots codés sur 32 bits.
- Pour conserver Ad_Max, nous aurons besoin d'un registre 8 bits (reg8 fait en TP2)
- Pour conserver le Max, nous aurons besoin d'un registre 32 bits (extension de reg8)
- Pour gérer Ad_Courante, nous avons besoin d'un circuit qui permet d'initialiser Ad_Courante avec Ad1, d'incrémenter Ad_Courante, et de s'arrêter lorsque Ad_courante atteint Ad2.

L'exécution de cet algorithme passe par plusieurs étapes successives, et nécessite par conséquent d'utiliser un circuit séquentiel qui permet de séparer les différentes actions dans le temps et de gérer leur séquencement. Il s'agit donc d'un circuit séquentiel composé d'un certain nombre d'états, et qui fonctionne selon le schéma suivant :

- A chaque top d'horloge, une transition s'effectue selon les conditions en cours, soit sur le même état soit vers un état différent,
- A chaque top d'horloge, donc à chaque transition, des opérations sont effectuées : les entrées de ces opérations sont préparées avant le top d'horloge, et le résultat est validé (enregistré) sur le top d'horloge.

Pour notre algorithme de calcul du Max, notre circuit peut être représenté par le graphe d'états suivant (les noms des états seront majuscule pour faire la différence avec les autres signaux) :



- A l'entrée dans l'algorithme, on se trouve dans un état que l'on appellera « INITAD», et on y restera tant que l'ordre de calcul n'est pas donné (on utilisera une entrée appelée « cal »)
- Lorsque « cal » est mis à 1, on quitte l'état « INITAD » pour passer dans un nouvel l'état « INITMAX », en préparant l'initialisation du compteur d'adresse Ad_Courante <- Ad1
- Au prochain top d'horloge, on quitte l'état « INITMAX » pour passer dans l'état « MAJMAX », en préparant les opérations suivantes :
  Max <- Tab(Ad_Courante) ; Ad_Max <- Ad_Courante; Ad_courante <- Ad_Courante + 1
  Il est important ici de noter que les 3 opérations ne sont pas interdépendantes, sinon elles ne pourraient pas être exécutées et validées sur un même cycle d'horloge.
- On boucle dans l'état « MAJMAX » tant que Ad_courante <= Ad2 (on utilisera un signal appelé « finTab » produit par un circuit ucmp8 fait en TP1. A chaque cycle d'horloge, on réalisera les opérations suivantes :
  Si Elément_courant > Max Alors Max <- Elément_courant & Ad_Max <- Ad_courante
- Lorsque Ad_courante atteint Ad2 (signal finTab=1), on passe dans un nouvel état qu'on appellera « FINI » et on y restera jusqu'à ce que l'entrée cal soit remise à 0. Dans ce cas on passe de nouveau à l'état « INITAD » pour recommencer un nouveau cycle de calcul.

A- Circuit etats_cal_max

Pour ne pas avoir à traiter toutes les difficultés en même temps, on adoptera une approche modulaire. On commencera donc par construire et tester le module qui gérera le séquencement décrit dessus : etats_cal_max(rst, clk, cal, finTab : INITAD, INITMAX, MAJMAX, FINI)

Et pour simplifier cette réalisation, on utilisera une bascule par état (bascule D). On peut donc écrire :
INIT := /cal on clk set when rst // set pour que INIT soit initialisé à 1 au démarrage

Réaliser le module etats_cal_max, le tester en vérifiant que :

- Après initialisation, l'état INITAD est à 1 et tous les autres états sont à 0
- L'on reste dans l'état INITAD tant que cal est à 0, quel que soit le nombre de tops d'horloge
- L'on passe de INITAD à INITMAX lorsque cal est à 1 au prochain top d'horloge
- L'on passe de INITMAX à MAJMAX
- L'on reste dans MAJMAX tant que finTab est 0
- L'on passe de MAJMAX à FINI lorsque finTab est 1
- L'on reste dans FINI tant que cal est 1
- L'on passe de FINI à INITAD lorsque cal est à 0

B- Circuit cal_max

Nous allons réaliser le module cal_max(rst, clk, cal, ad1[7..0], ad2[7..0], elemCour[31..0] : adCour[7..0], max[31..0], adMax[7..0], INITAD, INITMAX, MAJMAX, FINI)

A chaque étape, le module cal_max fournit l'adresse de l'élément courant dans adCour, et prend en entrée la valeur de l'élément courant dans elemCour.

Pour gérer Ad_Cour, nous avons besoin d'un circuit qui permet d'initialiser Ad_Cour avec ad1, d'incrémenter ad_Cour, et de s'arrêter lorsque Ad_Cour atteint ad2. Ce circuit peut avoir l'interface suivante : count8_b1_b2 (rst, clk, init, count, ad1[7..0], ad2[7..0] : adCour[7..0])

Mais on peut contrôler la dernière action d'arrêt lorsque Ad_Cour atteint ad2 à l'extérieur de ce module (en agissant sur l'entrée count : mise de count à 0 lorsque adCour atteint ad2), et utiliser un module plus simple (similaire au count4_init fait en TP2) :
count8_init (rst, clk, init, count, ad1[7..0] : adCour[7..0])

Réaliser et tester ce module (count8_init ou count8_b1_b2 au choix) ***sans utiliser l'entrée « en » des bascules.***

Réaliser le module cal_max. Tester et valider le module avec le fichier cal_max.test fourni.
Attention : il ne faut pas modifier l'interface du module (dessus) et respecter la casse.

C- Circuit max_tab

Pour réaliser le calcul du max d'un tableau, on va coupler cal_max avec une mémoire RAM (module prédéfini) : $ram_aread_swrite(clk, write, adr[7..0], datIn[31..0] : dataOut[31..0])

Cette RAM est un ensemble de 256 mots de 32 bits chacun. Chaque mot est référencé par une adresse (numéro) comprise entre 0 et 255.

Le fonctionnement de cette RAM est le suivant (similaire à celui des registres) :

- ***Lecture asynchrone*** : Le contenu du mot dont l'adresse est présentée sur l'entrée adr est immédiatement visible sur la sortie outRam
- ***Ecriture synchrone*** : pour écrire une donnée à une adresse donnée, il faut :
  - Présenter cette adresse sur l'entrée adr
  - Présenter la donnée écrire sur l'entrée dataIn
  - Mettre le signal write à 1

  L'écriture se fait sur le front de l'horloge clk

Réaliser le module max_tab.
Tester et valider ce module : Initialiser la RAM avec le fichier tableau.ram fourni en cliquant sur le bouton jaune visible à droite lorsqu'on active le simulateur, puis sur le bouton parcourir de « MEMORY CONTENTS ». Tester les cas suivants (en agissant sur les valeurs de ad1 et ad2 :

- Le max est le premier élément du tableau
- Le max est le dernier élément du tableau
- Le max n'est ni le premier ni le dernier élément du tableau