

## N7\_SN\_1A

### Architecture des ordinateurs - Semestre 5

#### TD3 - Circuits séquentiels - Algorithmes cablés

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
```

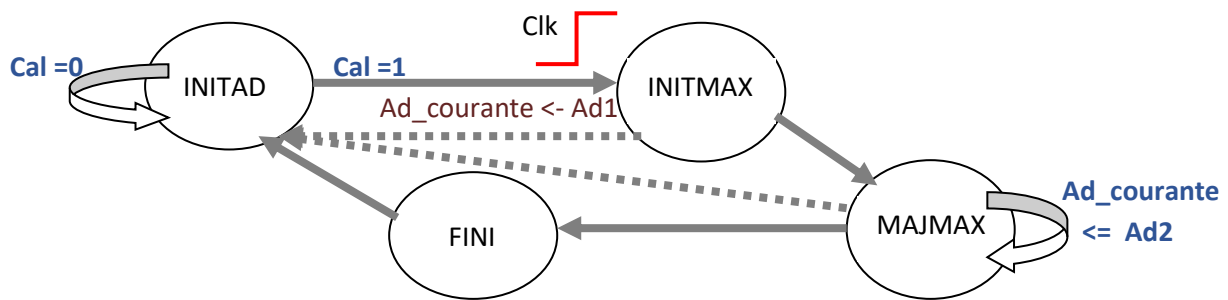
Analysons cet algorithme et listons les éléments dont nous avons besoin :

- 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 séquençement. 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 état « INITMAX », en préparant l'initialisation du compteur d'adresse  $Ad\_Courante \leftarrow Ad1$
- Au prochain top d'horloge, on quitte l'état « INITMAX » pour passer dans l'état « MAJMAX », en préparant les opérations suivantes :  
 $Max \leftarrow Tab(Ad\_Courante)$  ;  $Ad\_Max \leftarrow Ad\_Courante$  ;  $Ad\_courante \leftarrow Ad\_Courante + 1$   
 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  $Ad\_courante \leq Ad2$  (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  $Max \leftarrow Elément\_courant$  &  $Ad\_Max \leftarrow Ad\_courante$
- 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 gèrera le séquençement 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`

Compléter le module `etats_cal_max`. Il sera testé en TP3.

### B- Circuit cal max

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`. Il sera testé en TP, associé une ram (module prédéfini) : `$ram_aread_swrite(clk, wRam, adCour[7..0], inRam[31..0] : outRam[31..0])`.