Machines à états finis

Objectifs

L'objectif de ce travail est d'illustrer la description comportementale en langage VHDL de machines à états finis autonomes de type Moore ou Mealy.

Introduction

On rappelle qu'un automate à état fini (Finite State Machine) constitue un modèle d'architecture des circuits numériques pour lequel les combinaisons des sorties s'expriment en fonction des combinaisons d'entrée et d'état interne. L'évolution de l'état interne peut être libre (machines asynchrones) ou contrôler par une entrée d'horloge (machines synchrones). Seles. Selon que la ou les sorties dépendent ou non des entrées, on parle de machine de Mealy ou de machine de Moore. Dans le cas de circuits mettant en œuvre des transferts de données, une machine d'état est souvent utilisée comme description du fonctionnement d'une unité de contrôle du chemin de données. On parle également de séquenceur. Dans ce TP, on se propose d'illustrer la description de machines à états sous forme comportementale.

I Contrôleur d'une mémoire

Il s'agit à présent d'illustrer la description de machines à états finis de type Moore en VHDL au travers d'un circuit de contrôle d'une mémoire. Le diagramme de contexte du circuit (figure 3) montre que la mémoire à contrôler dispose de 2 signaux de commandes *oe* et *we*. selon que l'on veut lire ou écrire respectivement sur cette mémoire. Le rôle du circuit de contrôle est d'interfacer la mémoire en générant ces commandes en réponse aux signaux *ready* et *read/write* provenant d'un bus d'un système à base de microprocesseurs (bus AMBA par exemple)..

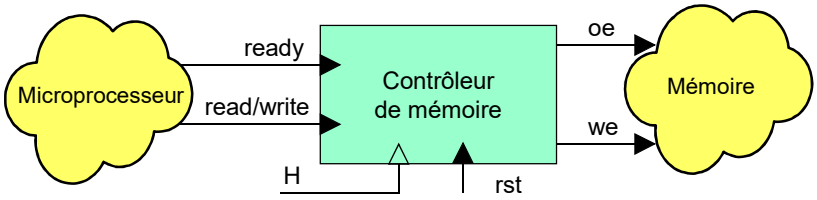

Figure 1 : Circuit de contrôle d'une mémoire.

Le principe de fonctionnement de cette interface esclave est le suivant :

Une nouvelle transaction vers la mémoire débute toujours avec l'affirmation du signal ready. Un cycle d'horloge plus tard, la valeur read_write détermine s'il s'agit d'un cycle de lecture (read_write=1) ou d'écriture (read_write=0). Un cycle se termine par une nouvelle affirmation du signal ready. Les sorties (output enable) oe et (write enable) we sont respectivement vraies (à '1') durant un cycle de lecture et un cycle d'écriture.

Préparation

A. A l'aide d'un process unique, synchrone à une horloge comprenant une instruction *case* permettant de spécifier l'état des sorties et les états suivants pour chaque état courant, décrivez le comportement reproduisant celui décrit par le diagramme à états finis de la Figure 2. L'état initial sera fixé sur *rst* vrai à bas.

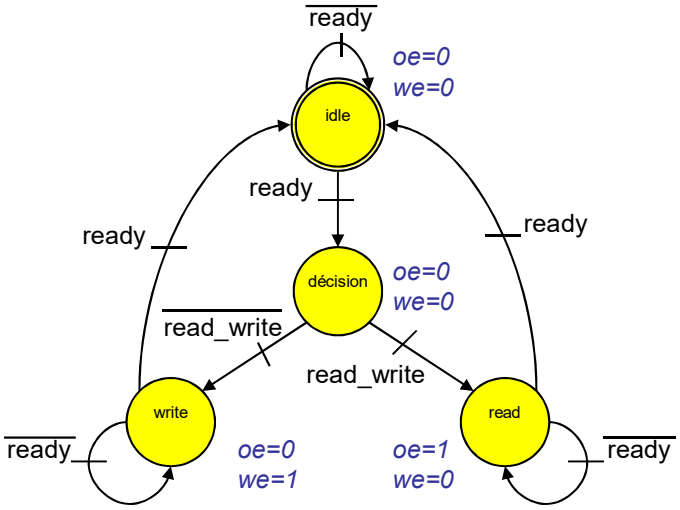

Figure 2 : Diagramme d'états de la machine de Moore.

B. Dans un second fichier, définissez un testbench.

C. A la suite de la première description, définissez une nouvelle architecture utilisant 1 process pour mémoriser l'état courant sur front d'horloge et un process combinatoire générateur des sorties et de l'état suivant.

D. Proposez finalement un nouveau couple entité-architecture à partir de la description à 2 process que vous modifierez pour permettre la gestion d'un mode rafale (burst). Dans ce mode, lorsque le signal *burst* est affirmé sur l'état *read,* le contrôleur effectue 4 accès successifs à des emplacements mémoires contigus en incrémentant automatiquement les 2 poids faibles de l'adresse (signal *adr*).

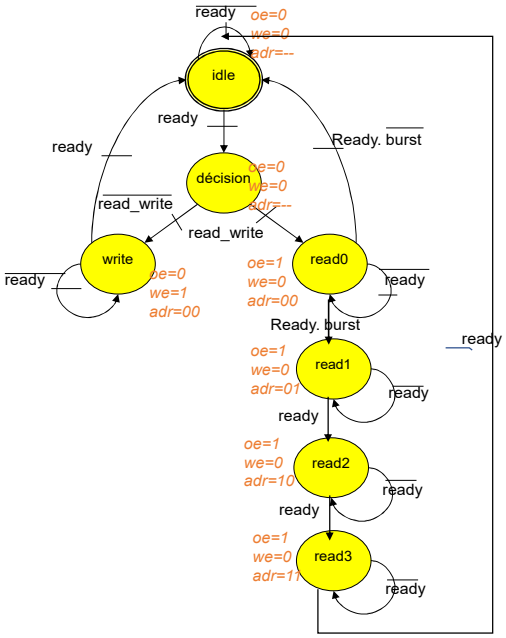

Figure 3: Diagramme d'états du controleur modifié

Travail en séance

Simulez chacune des architectures en faisant apparaitre la variable d'état interne. Dans le testbench, insérez une instruction de configuration ciblant l'architecture à valider Commentez soigneusement les résultats de votre simulation.

II Compteur modulo-3 up/down (Mealy)

On s'intéresse dans un premier temps à la validation fonctionnelle d'une machine de Mealy machine permettant le comptage ou le décomptage d'une séquence modulo-3. Comme le montre le diagramme ci-dessous, 3 états forment la machine. Les entrées du diagramme sont formées des combinaisons de E et S (dans l'ordre), *S* indiquant le sens (comptage ou décomptage) *E* autorisant ou non l'opération. La sortie *Z* fournie une retenue lors du modulo du compteur.

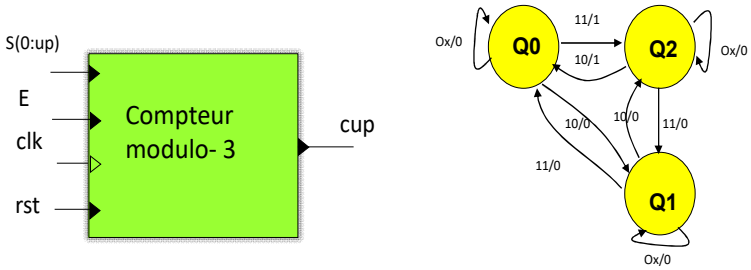

Figure 4: symbole et diagramme du compteur

Préparation:

A. Donnez une description de l'entité *compteur* en utilisant le type *bit* pour l'ensemble des signaux de l'interface.

B. Donnez une description de l'architecture du circuit à l'aide de deux processus concurrents.

C. Développez une entité et une architecture de test du circuit. On considérera 2 scénarii :

1. Dans le premier scénario, les entrées seront synchrones : une modification de l'entrée interviendra toujours peu de temps après le front d'horloge.
2. Dans un second scénario, les entrées ne sont plus synchrones : une entrée est modifiée par exemple, peu de temps avant le front d'horloge.

Travail en séance :

Testez les 2 scénarii. Utilisez la fonctionnalité *waveform compare* pour comparer les résultats et concluez.