N7_SN_1A

Architecture des ordinateurs - Semestre 5

TP2

ATTENTION : Pour pouvoir utiliser les fichiers de test fournis, il faut respecter l'interface des modules (noms des signaux en entrée et en sortie) telle qu'elle est indiquée dans le sujet.

Exercice 1

Pour coder les différents états d'un circuit séquentiel, on utilisera principalement des bascules D et des bascules T, dont le fonctionnement est régi par les équations suivantes :

- Bascule D (Delay) : l'état X (=sortie) à l'instant n+1 et égal à l'entrée D à l'instant n.
  X := D on clk reset when rst enabled when en // en est facultatif, égal à 1 par défaut
- Bascule T (Trigger) : l'état Xn+1 est égal à Xn si l'entrée T=0, et = /Xn si T=1
  X := /T*X + T*/X on clk reset when rst enabled when en

Ecrire un module test_bascules qui permet de tester le fonctionnement des bascules D et T.

Exercice 2

Un registre est un bloc d'un certain nombre de bits (8, 16, 32, 64, …) qui sert à mémoriser des informations utiles à l'exécution des instructions dans un processeur : opérandes, adresses, etc. L'entrée du registre y est mémorisée sur le top d'horloge.

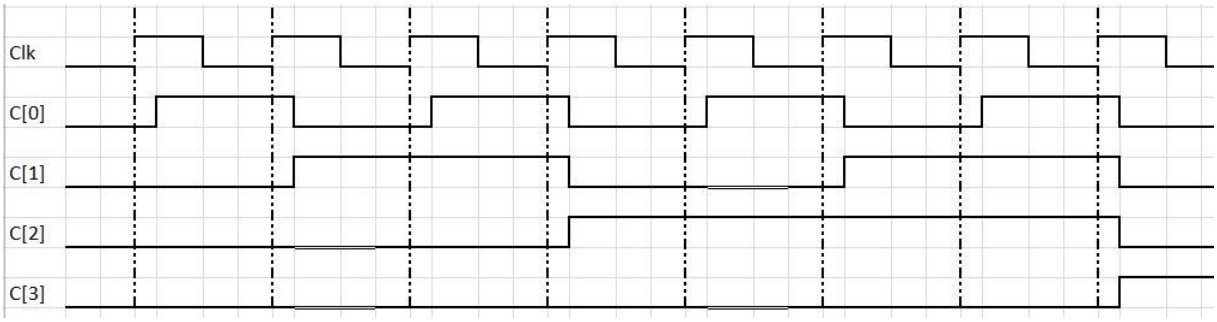
- Ecrire et tester le module reg8_D (rst, clk, en, e[7..0] : sr[7..0]) en utilisant des bascules D
- Ecrire et tester le module reg8_T (rst, clk, en, e[7..0] : sr[7..0]) en utilisant des bascules T
- valider avec le fichier de test fourni

Exercice 3

Ecrire et tester le module count4 (rst, clk, en : c[3..0]) qui compte de façon cyclique : 0, 1, 2, …, 15, 0, … lorsque l'entrée en = 1. Valider avec le fichier de test fourni.

Exercice 4

On souhaite simplifier count4 de manière à n'avoir aucune logique combinatoire pour le calcul des entrées T[i], quitte à contourner la règle qui prescrit que toutes les bascules doivent partager la même horloge. En vous inspirant du chronogramme dessous, écrire et tester le module clock4 (rst, clk, en : c[3..0]) qui réalise cet objectif.



Exercice 5

On souhaite transformer count4 de manière à pouvoir l'initialiser avec une valeur donnée en entrée lorsqu'un ordre d'initialisation est activé (entrée init=1). Ecrire et tester le circuit :
Count4_init (rst, clk, en, init, e[3..0] : c[3..0]) qui est initialisé avec la valeur e[3..0] lorsque inti=1, et compte de façon cyclique 0, 1, 2, …, 15, 0, … lorsque init=0.

Ecrire et tester ce module. En vous inspirant du fichier de test coun4.tst, écrire un fichier de test complet (count_init4.tst) pour ce module (**en mettre une copie dans votre espace shdl**).

Exercice 6

Réaliser et tester le circuit count4_pair_impair (rst, clk, p : c[3..0]) qui fonctionne de la façon suivante :
P = 1 : passe de la valeur courante à la première valeur paire supérieure
P = 0 : passe de la valeur courante à la première valeur impaire supérieure

Ecrire et tester ce module (fichier de test fourni).

Pour aller plus loin

7- Ecrire et tester le module count4_b1_b2 qui fonctionne de la façon suivante (sans utiliser l'entrée en des bascules) :
- initialisé avec b1 quand init=1 et count=0,
- compte quand init=0, count=1, et compeur < b2
- n'est pas modifié lorsque init = 0 et count = 0, et lorsque init = 1 et count = 4

8- En utilisant le module count4_init, réaliser et tester le circuit count4_par blocs (rst, clk, en : c[3..0]) qui compte de la façon suivante : 0, 1, 2, 3, 8, 9, 10, 11, 0, …

9- En utilisant le module count4_init, réaliser et tester le circuit count_5_10 (rst, clk, en : c[3..0]) qui compte de 5 jusqu'à 10, puis s'arrête.