Initiation au langage VHDL- Travaux Dirigés

Auteur : Hervé Le Provost
herve.le-provost@cea.fr

Préambule

Règles

Pour des raisons de cohérence et de facilité de suivi, Il est demandé :

- *De créer un répertoire pour chaque description VHDL (exercices 1, 2, 3, 4) portant le nom souligné au dessus de la représentation schématique du module à concevoir. Ce nom sera aussi celui de l'entité VHDL.*
- *d'utiliser les mêmes noms de ports que ceux notés sur la représentation schématique du module à concevoir.*
- *d'utiliser les mêmes noms de paramètres génériques que ceux notés sur la représentation schématique du module à concevoir.*

Exemple :

- *répertoire "decodeur",*
- *nom de l'entité decodeur",*
- *ports "i(2 downto 0)", "o(7 downto 0)".*

Ne pas utiliser de noms de signaux ou variables comportant des accentuations.
En en-tête de chaque description VHDL (entité + architecture), activer par défaut le package 'std_logic_1164' :

```
library ieee;
use ieee.std_logic_1164.all;
```

Pour les descriptions utilisant des fonctions de conversion ou des opérations d'addition ou de soustraction, activer les packages 'std_logic_arith ' et 'std_logic_unsigned' :

```
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

Utiliser de préférence les types std_logic, std_logic_vector, integer.

Outils

Le flot de développement conseillé est le suivant :

- *Lancer « Xilinx ISE 12.2i » et créer un nouveau projet portant le nom du module VHDL à concevoir (exemple : encodeur) et sélectionner la famille de composants spartan3A, le circuit « XC3S700A », le boitier « FG484 ».*
- *Ajouter le fichier de description VHDL au projet en prenant soin de décocher la case 'copy to project'*
- *Cliquez sur synthesise->'check syntax' pour vérifier la syntaxe de votre description et la corriger éventuellement.*
- *Cliquez sur synthesise->'view technology schematic', le schéma vous paraît-il cohérent avec votre description ?*
- *Si oui, ajouter au projet le fichier VHDL de test et lancer le simulateur depuis l'environnement « Xilinx ISE 12.2i ».*
- *Si votre description est validée par votre simulation, vous pouvez alors tester votre code sur la cible. Pour cela :*
  - *ajoutez à votre projet le fichier de contrainte ayant pour extension « .ucf » et pour nom le nom de votre entité (fichier fourni).*
  - *Cliquez sur « Generate Programming File » puis sur « Configure Target Device ->IMPACT »*

- *Associez le fichier « download.bit » au composant XC3S700A et cliquez sur « bypass » pour la PROM.*
- *Avec le bouton droit, cliquez sur le composant XC3S700A et sélectionnez l'ordre de programmation.*

Conseils

Pour chaque exercice, il est conseillé de :

- *Identifier les ports d'entrée, de sortie, d'entrée/sortie et écrire l'entité correspondante*
- *Ecrire l'architecture associée à l'entité en identifiant un éventuel signal d'horloge*

1. DECODEUR

Concevoir, simuler puis tester la description VHDL correspondant à la table de décodage représentée Tableau 1.

entrées (representation binaire)	**sorties (representation binaire)**
sw(2 downto 0)	**led(7 downto 0)**
000	00000001
001	00000010
010	00000100
011	00001000
100	00010000
101	00100000
110	01000000
111	10000000

Tableau 1 - table de décodage

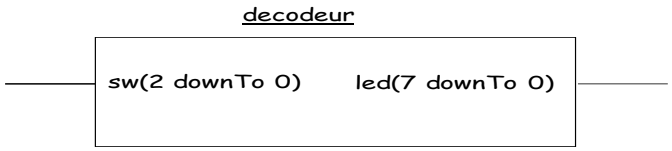

Figure 1- entrées/sorties du décodeur

2. DECODEUR SYNCHRONE

Rendre la sortie du décodeur décrit à l'exercice 1 synchrone d'une horloge 'clk_50MHz'. Pour cela, concevoir, simuler puis tester la description VHDL correspondant à la table de vérité représentée Tableau 2.

horloge	entrée (représentation binaire)	sortie (représentation binaire)
clk_50MHz	**sw(2 downto 0)**	**led_{n+1} (7 downto 0)**
0	x	led_n
1	x	led_n
↓	x	led_n
↑	000	00000001
↑	001	00000010
↑	010	00000100
↑	011	00001000
↑	100	00010000
↑	101	00100000
↑	110	01000000
↑	111	10000000

Tableau 2 – décodeur synchrone : table de décodage

Note :
x: non relevant
↑: front montant de l'horloge
↓: front descendant de l'horloge

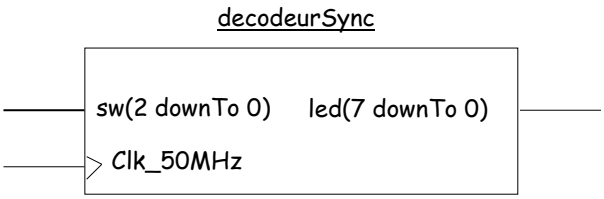

Figure 2 – entrées/sorties du décodeur synchrone

Quel problème potentiel peut apparaitre si le signal d'entrée n'est pas lui-même synchrone de l'horloge ?

3. MACHINE A ETATS D'ALLUMAGE DES LEDS

Soit un système de contrôle de l'allumage des leds. Il est modélisé par une machine d'états comportant les 4 états principaux « attente », « allume0 », « allume1 » et « allume2 » (Figure 3). Cette machine est synchrone de l'horloge 'clk_50MHz'. L'état de repos et d'initialisation par le signal 'reset' du système est l'état « attente ». Sur l'ordre du signal 'start' asynchrone de l'horloge ', la machine passe successivement par les états « allume0 », « allume1 », « allume2 ». Elle reste dans chacun des états pendant une durée « tempo/50MHz ». L'ordre de transition sur la Figure 3 est représenté par l'expiration d'un timer T.

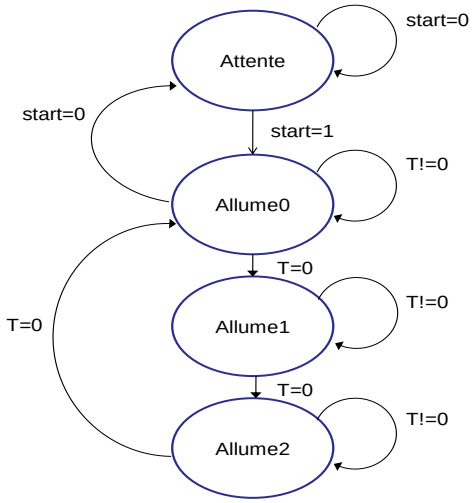

Figure 3 – Machine d'état d'allumage des leds

Les entrées/sorties du module sont représentées sur la Figure 4. Le temps d'attente dans chacun des états d'allumage est représenté par un paramètre générique « tempo ».

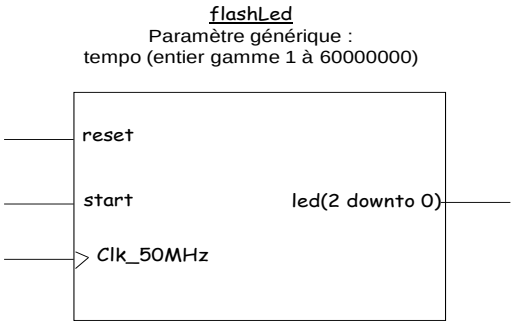

Figure 4 – entrées/sorties du module d'allumage des leds

Le Tableau 3 ci-dessous décrit l'état des sorties en fonction de l'état de la machine.

état	Sortie led(2 downto 0)
attente	000
allume0	001
allume1	010
allume2	100

Tableau 3 – valeurs des sorties en fonction de l'état de la machine

Ecrire, simuler puis tester le code VHDL correspondant à la machine d'état.

4. COMMANDE AVANCEE DES LEDS

Ecrire, simuler puis tester le code VHDL décrivant la structure représentée Figure 5.

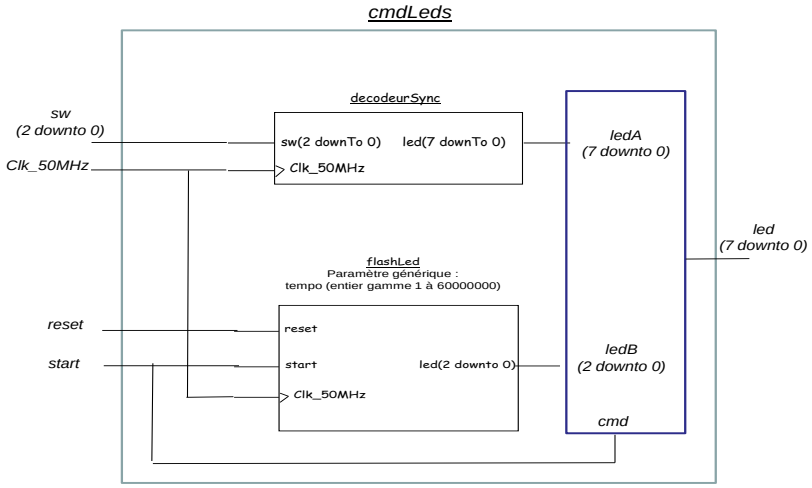

Figure 5 – schéma bloc du système de commande avancée des leds

cmd	Led(7 downto 0)
0	LedA(7 downto 0)
1	Led(2 down to 0)<=LedB(2 downto 0) Led(7 down to 3)<=b"00000"

Tableau 4- multiplexeur des sorties leds