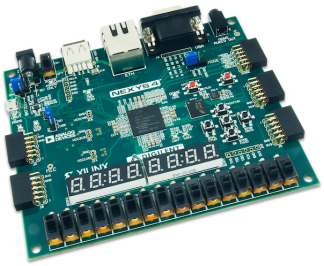
UPMC
SORBONNE UNIVERSITÉS

Université Pierre et Marie Curie

Rapport TP 1, 2 et 3

Systèmes programmables



KUOCH Johnny
JIANG Hongbo
2016 - 2017

Enseignant : M. DENOULET Julien

Fevrier 2017 — Mars 2017

1 re partie : Synthèse VHDL sur FPGA

I. Prise en main de la carte et des outils

Après avoir crée un nouveau projet, on ajoute le fichier source VHDL "test.vhd" au projet.

Les résultats de la simulation du module décrit par ce code VHDL se trouve dans le fichier "Simulation de test.png" dans le dossier Documents.

- Lorsque SW0 est actif, LED0 l'est aussi.
- Lorsque SW1 est actif, LED1 l'est aussi.
- Lorsque SW0 et SW1 sont actifs, LED0, LED1 et LED2 le sont aussi.

L'entité décrite par le code VHDL est bien le système attendu.

Afin de l'implémenter sur la carte, il faut ajouter un fichier de contrainte afin d'associer ses différentes entrées, aux interrupteurs de la carte et sa sortie, aux leds de la carte. Une fois le fichier ajouté, il faut lancer la synthèse, l'implémentation sur la carte et générer le flux de données pour configurer la carte.

II. Cas d'étude - Synthèse VHDL

1. Compteurs imbriqués

Il s'agit d'un système comportant deux compteurs qui s'incrémentent à chaque front d'horloge : un qui s'active sur un signal start et l'autre qui met le signal start à 0 lorsqu'il atteint sa valeur limite. En l'exécutant sur la carte et en associant les états logiques des bits du signal LED à des LEDs de la carte, on devrait les voir afficher la valeur des 4 derniers bits du compteur 2 : on utilise les 4 derniers bits pour diviser la fréquence de l'horloge principale.
En le testant sur la carte, on obtient bien ce qui était attendu à la différence que le compteur repart de 0 lorsqu'il a atteint sa valeur maximale alors qu'il devrait s'arrêter.
Lors de l'implémentation sur la carte du système décrit par le code VHDL, les lignes qui ne sont pas utilisées sont supprimées par Vivado pour optimiser l'implémentation.
D'après le message d'information, le signal start est toujours à 0 ce qui est normal car la valeur limite du compteur 1 est trop grande et n'est donc jamais atteinte : en effet, dans le code VHDL initial, elle est de 70 000 000 alors que le compteur ne peut compter que jusqu'à 20 000 000.

2. Compteur d'impulsion

La fonctionnalité décrite par le module VHDL est de compter des impulsions : lors de l'appui sur le bouton gauche, le compteur s'incrémente de 1 et lors de l'appui sur le bouton central, le compteur se décrémente de 1.
La simulation se déroule bien mais lors de la synthèse sur la carte, le programme ne fonctionne pas : le code ne peut pas être synthétisé car on essaye de synthétiser une bascule D synchronisée par deux horloges ce qui n'est pas possible.
Pour que le compteur d'impulsion fonctionne correctement, il faut modifier le code de telle sorte que chaque bascule D soit synchronisée par un signal d'horloge (voir le fichier impulse count.vhd se trouvant dans le dossier Sources/VHDL)
Afin de gérer le phénomène de rebond des boutons, il faut verrouiller le bouton après la réception du premier signal pendant un intervalle de temps raisonnable après lequel les rebonds se seront arrêtés.
Le fichier "Simulation impulse count.jpg" se trouvant dans le dossier Documents représente le chronogramme de la simulation de l'architecture de l'entité décrite par le code "impulse_count.vhd" pour

un compteur de verrouillage modulo 6.
Le compteur d'impulsion doit être initialisé par une mise à 1 du signal Reset avant de pouvoir compter. Ensuite, lors d'une impulsion sur le bouton central, le compteur d'impulsion s'incrémente d'une unité, le signal lock verrouille l'appui sur les boutons et le compteur de verrouillage démarre jusqu'à qu'il atteigne sa valeur finale et déverrouille l'appui sur les boutons. Le système peut alors compter à nouveau.

3. Décodeur

Le système décrit par le code VHDL du fichier "Selector.vhd" est un décodeur qui à partir d'un signal d'entrée de 2 bits permet de choisir le mode de clignotement des LEDs

2 e partie : Codesign Matériel / Logiciel

Le développement d'un système sur la carte avec le logiclel Vivado s'effectue en deux étapes :

— configuration de la plate-forme matérielle
— développement de la partie logicielle

Configuration de la plate-forme matérielle

La configuration de la plate-forme logicielle comprend les étapes suivantes :

— la création d'un nouveau projet au cours de laquelle on choisit la carte que l'on va utiliser (ici Nexys 4).
— la définition du langage de programmation (ici VHDL).
— la création d'un Block Design

Dans le Block Design, il faut commencer par instancier l'IP Microblaze, ensuite configurer le processeur en lançant **Run Block Automation**, choisir une mémoire locale de 32 Ko et autoriser les interruptions.
On configure ensuite l'horloge en double cliquant sur le module **Clocking Wizard** et en modifiant dans la section **Input Clock Information** dans l'onglet **Clocking Options**, le paramètre **Source** de la **Primary Clock** en **Single Ended Clock Capable Pin**.
On ajoute ensuite deux interfaces GPIO pour contrôler les LEDs, les interrupteurs et les boutons de la carte. On connecte ensuite les différents modules entre eux.

Il faut ensuite associer les différentes broches du FPGA aux signaux du système décrit précédemment en important le fichier "Nexys4DDR_Master.xdc". Ce fichier contenait initialement différentes commandes commentées.
La commande pour associer une broche à un signal est de la forme suivante :

```
set_property -dict  PACKAGE_PIN J15 IOSTANDARD LVCMOS33  [get_ports  sw_tri_i[0] ] ;
```

Cette instruction associe la broche J15 du package Pin au port GPIO sw_tri_i[0].

Il faut ensuite créer un HDL Wrapper et générer le Bitstream.

Développement de l'application logicielle

La première application logicielle que nous avons développé est une application qui allume une LED lorsque l'interrupteur qui lui est associé est activé. Le code commenté se trouve dans le fichier "test.c" du dossier C du dossier Sources.

Le code source en langage C et commenté de la seconde application se trouve dans le fichier "test2.c" dans le même dossier.

3 e partie : Conception d'IP pour le Microblaze

Création d’une IP contrôleur de LED

Le code du module VHDL "my_led.vhd" qui nous ait demandé d’écrire se trouve dans le dossier VHDL du dossier Sources. Le code étant simple, nous ne l’avons pas commenté. Il s’agit juste de connecter l’état des leds par groupe de 4 à celui des interrupteurs.