HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Prof. Yann Thoma

Laboratoire CSF

semestre de printemps 2022 - 2023

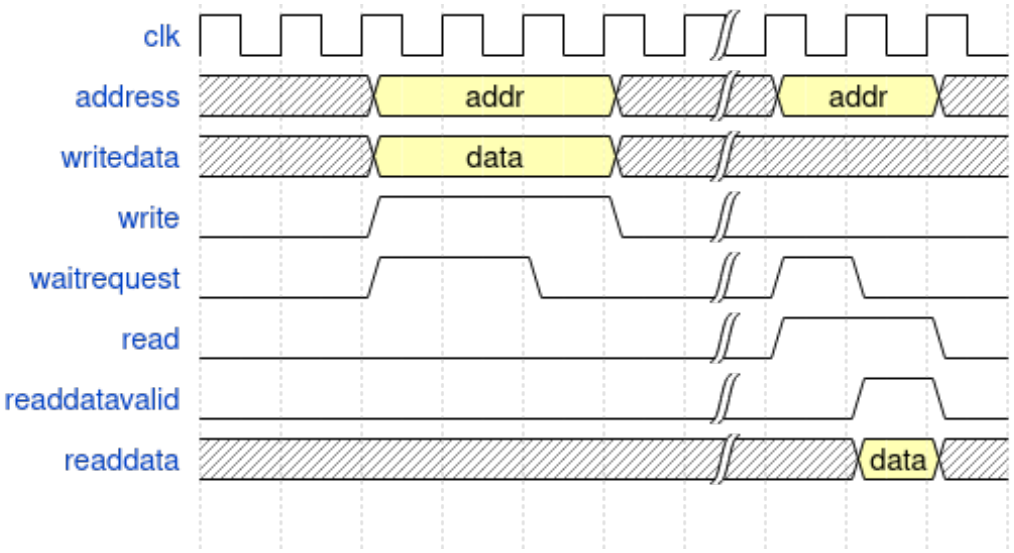
Réalisation d'un bloc simple sur bus avalon
Prise en main des outils de simulation, synthèse et placement-routage

Introduction

Ce premier labo vise la prise en main des outils de simulation et de synthèse, ainsi que la réalisation d'un composant relativement simple. Nous y réaliserons un composant sur bus avalon, le vérifierons, observerons le résultat en synthèse et le testerons sur carte.

Cahier des charges

Le composant à réaliser offre une interface sur bus Avalon. Ce bus permet des accès en lecture/écriture à des addresses définies, et respecte le comportement suivant :



Nous pouvons noter que le signal `waitrequest` permet au composant de faire attendre les lectures et écritures jusqu'à ce qu'il soit prêt. Il peut donc faire monter (ou non) ce signal, et son passage à 0 valide alors l'écriture ou la lecture.
⚠ Pour ce laboratoire, votre composant devra activer le waitrequest pendant au moins un cycle lors d'un accès en lecture ou écriture.
Notre composant offre sept éléments, accessibles via des mots de 32 bits, aux adresses suivantes :

Addresse	Description
0x0	Valeur fixe à définir
0x1	Compteur en lecture seule
0x2	Contrôle du compteur
0x3	Registre en lecture-écriture
0x4	Registre en lecture-écriture
0x5	Registre en lecture-écriture
0x6	Registre en lecture-écriture

Le premier registre doit juste fournir une valeur pré-définie. Il pourrait s'agir d'un code particulier ou un numéro de version.
Le deuxième est un compteur accessible en lecture seule.
Le troisième est un registre de contrôle pour le compteur. Y écrire la valeur 0x1 remet le compteur à 0, et y écrire la valeur 0x2 lance une incrémentation du compteur.
Les derniers sont quatre registres sur 32 bits, accessibles en lecture/écriture.

Etape 1

Votre composant peut être décomposé en une partie responsable du respect du protocole Avalon et une contenant les différents registres. Commencez par dessiner le schéma du système.

Etape 2

Les étudiants EEMs des groupes doivent décrire le composant en VHDL, chacun pour soi.
Les étudiants IEs sont là pour aider les EEMs à avancer et participent aux discussions. Ils sont notamment responsables de guider le développement pour la réalisation d'un byteenable, permettant de sélectionner les bytes à modifier lors des écritures.

Etape 3 (en parallèle de 2)

Il est nécessaire de pouvoir vérifier le bon fonctionnement du système. Les IEs ayant déjà acquis ce type de connaissances, ils doivent prendre en main la mise en place d'un testbench. Le choix technologique est libre (VHDL, SystemVerilog, vérification formelle).

Etape 4

Votre code fonctionnant en vérification, nous allons le synthétiser en utilisant le logiciel Quartus 18, d'Altera. Nous allons intégrer le système sur une carte DE1-SoC. Un logiciel en C, fourni, lance quelques commandes. Il est possible de le modifier pour disposer de plus de scénarios de tests.
Exploitez le projet Quartus fourni, et lancez une synthèse/placement/routage.
Observez les erreurs potentielles, ainsi que les warnings. L'analyse des warnings est toujours importante, car certains peuvent être d'une importance capitale (problème dans la liste de sensibilité, génération de latches, ...).

Vous devriez maintenant avoir la possibilité d'observer la quantité de ressources nécessaires à votre système. (Dans Analysis & Synthesis / Resource Utilization by Entity).

- Pour observer le schéma généré, sélectionnez *Tools/Netlist Viewers/RTL Viewer*. **Que voyez-vous ?**
- Pour observer le schéma généré, sélectionnez *Tools/Netlist Viewers/Technology Map Viewer (Post-Mapping)*. **Que voyez-vous ?**

- **Quelle est la différence entre les deux schémas ? (entre les deux vues).**

Etape 5

Maintenant que votre système est prêt, vous allez l'intégrer et la tester sur la FPGA.
Compilez (Synthèse, Placement & Routage cette fois-ci) votre projet.
Vérifiez la position des switches MSEL sur le dessous de la carte. MSEL[0..4] devrait valoir 01001. Regardez avec l'assistant si vous n'êtes pas sûr.
Dans le menu *programmer* il faut que la carte soit reconnue et sélectionnée, si ce n'est pas le cas "No Hardware" est affiché, il faut aller dans *Hardware Setup...* et chosir le hardware, normalement nommé DE-SoC. Une fois le hardware sélectionné, cliquez sur *Auto Detect* et sélectionnez le device **5CSEMA5** qui correspond à votre FPGA. Une fois le device sélectionné cliquez sur *Change File ...* et spécifiez le fichier .sof généré par votre compilation (dans output_files). Cochez la case *Program/Configure* et appuyez sur *Start*. La barre de progression en haut à droite vous signalera si tout s'est bien passé.

Pour tester votre système, exploitez le code C fourni. Pour ce faire, depuis une VM, exécutez la commande `make run`, qui compile, transfert le code, et l'exécute sur la carte. Testez puis faites valider votre intégration par le professeur ou l'assistant.
Les tests décrits dans le programme C ne sont pas réellement des tests. L'IE du groupe est responsable de mettre en place des tests automatiques plus pertinents.

Si la carte n'est pas reconnue

— Vérifiez que le câble USB est bien branché.
— Vérifiez que la carte de développement est allumée.
— Vérifiez que le matériel est reconnu par le pc avec `lsusb`. Il devrait y avoir une ligne avec un device **Altera**.
— Vérifiez que le service `JTAG` reconnaisse le matériel avec la commande `jtagconfig`, si rien n'est listé faite :

```
$ sudo killall -9 jtagd
$ sudo /opt/EDA/quartus16/quartus/bin/jtagd
```

Et vérifiez à nouveau avec :

```
$ jtagconfig
```

— Vérifiez la position des switches MSEL sur le dessous de la carte. MSEL[0..4] devrait valoir 01001. Regardez avec l'assistant si vous n'êtes pas sûr.
— Si rien ne fonctionne voir avec l'assistant.

A rendre

Rendez, sur cyberlearn, une archive par groupe contenant les différents codes, ainsi qu'un mini rapport expliquant vos choix, leas problèmes rencontrés et comment vous les avez résolus.