INF3500 - Conception et réalisation de systèmes numériques

Labo 1 - Introduction
8 janvier 2020

par Olivier Dion

Table des matières

1. Objectifs

Ce laboratoire sert d'introduction au cours INF3500. Son but est de vous introduire à votre environnement de travail. Pour ce faire, 3 objectifs sont à accomplir.

- Se familiariser avec l'outil **Vivado**
- Implémenter un simple module
- Effectuer la simulation d'un module
- Faire la synthèse et l'implémentation d'un module

Vous serez évalué sur ces objectifs ainsi que votre compréhension de l'exercice.

2. Conseils

Avant de commencer, voici quelques conseils.

- Assurez-vous de ne pas travailler dans un dossier temporaire !
- Utiliser **Git** (*GitBash* sur votre poste de travail) pour sauvegarder l'historique de vos travaux. Indexer seulement les fichiers code sources et autre fichier **non binaire**.
- Écrivez toujours le nom des autrices et matricules étudiant aux débuts de vos fichiers en commentaire. Ajoutez optionnellement une licence pour votre code source.
- La synthèse et l'implémentation sont des procédés assez longs. Imaginez un programme `C++` qui compile de 10 à 30 minutes. Relisez-vous pour vous sauver du temps !

3. Introduction à Vivado

Ouvrez le fichier `guide-vivado11.pdf` et faites le tutoriel. Il est impératif de bien suivre chaque étape à la lettre et de bien les comprendre. Vous allez faire cela pour le reste de la session !

4. Implémentation d'un module

Dans le tutoriel précédent, vous avez implémenté un additionneur à 3 bits. Pour vous assurer que vous avez bien compris comment un module fonctionne, vous allez maintenant implémenter la fonction ***ch**oose*. On va voir dans le prochain labo l'utilité de cette fonction.

La fonction **ch** est :

$$ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

où ch, x, y et z sont sur 1 bit (type `std_logic`).

Pour rappel,

Symbole	VHDL
$\neg$	not
$\vee$	or
$\wedge$	and
$\oplus$	xor

Vous allez donc créer une nouvelle entité appelée **ch** au lieu de **add3bits**. Inspirez-vous du code fourni dans `add3bits.vhd`. Cependant, enlevez les énoncés et signaux superflus et adaptez les ports d'entrées et de sorties !

NOTE ! Dans le même fichier, vous devriez avoir les 4 sections suivantes :

1. Licence (optionnelle), noms + matricules, description du fichier
2. Importations des librairies
3. Déclaration de l'entité
4. Architecture de l'entité

Livrable

Votre fichier `ch.vhd` avec les 4 sections demandées.

5. Simulation

Vous allez maintenant faire la simulation de votre module. Assurez-vous que votre nouveau module soit bien le **top** module. Sinon la simulation va se faire sur l'additionneur 3 bits ! Par la suite, jouez un peu avec le chronogramme pour voir si les signaux ont du sens.

Une fois que vous êtes certain que tout marche bien, pour chaque combinaisons possible (8) des signaux `x`, `y` et `z`, prenez une capture d'écran du chronogramme (pas de fenêtres de Windows), incluant le nom des signaux ainsi que l'échelle du temps.

Livrable

Captures d'écran que vous allez inclure directement dans votre rapport. Ne pas mettre dans le zip !

6. Synthèse et implémentation

Pour cette partie, vous allez créer un nouveau fichier de contraintes nommé `ch.xdc`. Inspirez-vous de `add3bits.xdc`. Vous pouvez cependant vous amuser à aller chercher des commutateurs plus loin sur la carte. Il suffit de lire les codes en blanc sur la carte pour savoir leur numéro.

Après vous êtes assurer que votre module est bien le **top** module, faites la synthèse puis l'implémentation de votre module. Générez ensuite un bitstream que vous allez télécharger sur votre carte FPGA.

Vérifiez que la carte FPGA produit bel et bien la bonne sortie en fonction des entrées. Aidez-vous d'une table de vérité au besoin.

Demandez par la suite à ce que l'on vienne vous évaluer.

Livrable

Votre fichier de contraintes nommé `ch.xdc` ainsi que votre fichier bitstream nommé `ch.bit`.

7. Question

Cette question est à répondre dans votre rapport.

a) La fonction **ch** est le préfix de ***ch****oose*. Expliquez de façon concrète et concise pour quelle raison cette fonction est appelée ainsi. Vous pouvez donner des exemples. INDICE ! Faites une table de vérité.

8. Rapport

Vous devez écrire un rapport selon les directives dans le fichier `rapport.pdf`.

Seuls les fichiers de type `pdf`, `DjVu` et `ps` sont acceptés pour le rapport. Aucun fichier de type Word (doc, docx) n'est accepté.

Pour ce rapport, vous n'avez **pas** à parler ou à remettre

- les ressources utilisées et la performance
- une description générale de votre système
- le fichier VHDL généré par vos sources .bde (diagrammes)
- la discussion

Assurez-vous d'inclure tous les fichiers listés dans les sections `Livrable`.

9. Barème

Critères	**Points**
Partie 4 : Conception du module	6
Partie 5 : Simulation	6
Partie 6 : Synthèse et implémentation	5
Partie 7 : Question	1
Rapport : Présentation et qualité de la langue	2
Total	20