HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Yann Thoma

Laboratoire VSN

semestre de printemps 2017 - 2018

Vérification d'une ALU combinatoire

Composant à tester

Nous nous intéressons à tester une unité arithmétique et logique capable d'effectuer 8 opérations, sur 2 vecteurs de taille générique.

Un paramètre générique `SIZE` permet de définir la taille des opérandes.

Les entrées sont:

Nom	Taille	Description
`a_i`	SIZE	Premier opérande
`b_i`	SIZE	Deuxième opérande
`mode_i`	3	Mode d'opération
`c_o`	1	Retenue en sortie
`s_o`	SIZE	Résultat

Les résultats attendus sont les suivants:

mode_i	**s_o**	**c_o**
"000"	a_i+b_i	retenue
"001"	a_i-b_i	retenue
"010"	a_i **or** b_i	indéfini
"011"	a_i **and** b_i	indéfini
"100"	a_i	indéfini
"101"	b_i	indéfini
"110"	s_o(0)= '1' when a_i=b_i **else** '0'	indéfini
"111"	0	indéfini

Un deuxième paramètre générique `ERRNO` permet d'injecter articifiellement des erreurs dans le design. Il s'agit d'un entier qui offre le comportement suivant:

1. S'il est compris entre 0 et 15, le résultat est valide;
2. S'il est compris entre 16 et 24, le résultat n'est pas valide.

Ce paramètre générique vous permettra de valider votre banc de test, et l'essayant avec toutes les valeurs de `ERRNO`.

⚠ L'entité à tester ne doit pas être modifiée.

Exercice 1

Nous allons utiliser TbGenerator pour générer un banc de test initial. Le projet est disponible sur gitlab.com : `https://gitlab.com/reds-public/TbGenerator`, et est open source. La documentation se trouve ici: `http://tbgenerator.readthedocs.io/en/latest/`. Un repo git est déjà présent sur votre machine, dans le répertoire : `/home/master/TbGenerator`.

1. Utilisez TbGenerator pour vous générer un banc de test initial. Pour ce faire, suivez les étapes suivantes:
   (a) Dans un terminal, placez-vous dans le répertoire : `/home/master/TbGenerator`
   (b) Lancez la commande suivante pour récupérer la dernière version: `git pull`
   (c) Placez-vous ensuite dans le répertoire : `/home/master/TbGenerator/dev/src/gen_tb_vhdl/simple_tb`
   (d) Lancez le script ainsi : `python3 gen_simple_vhdl_tb.py --help`
   (e) Après avoir identifié la manière de lancer le script, exécutez-le avec comme fichier d'entrée l'ALU fournie. Attention, il faut spécifier le template `template_combi1` car le DUV est purement combinatoire. La commande devrait ressembler à

```
python3 gen_simple_vhdl_tb.py -t template_combi1 -i <folderlab>/code/src/alu.vhd -d <folderlab>/code
```

2. Observez les fichiers générés, et lancez une simulation avec QuestaSim. Pour ce faire, exécutez le fichier `scripts/sim.do` depuis le répertoire `comp`.
3. Modifiez le fichier `sim.do` de manière à grouper automatiquement les signaux `*_sti`, `*_obs`, et `*_s`.
4. Le banc de test fourni ne fait pas grand chose. Placez-y de quoi générer des entrées. Pour ce faire arrangez-vous pour disposer d'une procédure pour l'application des entrées du DUV.

Exercice 2

Le banc de test n'effectue aucune vérification.

1. Modifiez-le de manière à vérifier le bon fonctionnement du DUV, dans le processus existant.
2. Arrangez-vous pour disposer d'une procédure pour la vérification.
3. Utilisez le logger que vous avez développé précédemment.
4. Modifiez la constante `ERRNO`, dans le banc de test, pour observer le bon fonctionnement de votre banc de test.
5. Modifiez le banc de test de manière à ce que la vérification se fasse dans un processus dédié, commandé par un signal de synchronisation.

Exercice 3

Nous désirons valider l'unité arithmétique et logique avec des entiers de 16 bits. Ceci devrait poser quelques problèmes de temps d'exécution. Pour remédier à ça:

1. Modifiez votre banc de test de manière à tester des cas pertinents;
2. Ajoutez de l'aléatoire pour tester 1000 cas supplémentaires. A vous de choisir une manière de générer de l'aléatoire.