INF3500 - Conception et réalisation de systèmes numériques

Labo 3 - Banc d'essais
1er février 2020

par Olivier Dion

Table des matières

1. Objectifs

Ce laboratoire sert à confirmer votre compréhension de la conception et l'utilisation de banc d'essai. Le but sera donc d'apprendre à générer des bancs d'essai pour des modules en **VHDL**. Pour ce faire, 2 objectifs sont à accomplir.

- Apprendre à générer des vecteurs de tests
- Comprendre l'importance des tests

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 !
- Dans le doute, consultez la **FAQ** du cours (`faq.pdf`).
- Écrivez vos rapports en `LaTeX`. La qualité de ceux-ci sera supérieure.

3. Contexte

Au dernier laboratoire, on vous a fourni un fichier `top-tb.vhd`. C'est à vous maintenant d'écrire vos tests. Une façon rapide pour les écrire est de générer les valeurs à l'aide d'un programme externe. Il suffit par la suite de copier les valeurs dans vos fichiers.

Évidemment, le programme qui génère ces valeurs doit être correctement écrit. Non seulement ça, mais le compilateur ou interpréteur de votre programme doit lui aussi être correctement écrit. Il en va de même de votre système d'exploitation. On peut remonter ainsi jusqu'aux transistors qui composent votre processeur par un certain manufacturier.

Ainsi, il existe une chaîne de confiance, de la production matérielle en passant par les librairies standards jusqu'à votre programme. Si vous ne faites pas confiance en un des maillons de cette chaîne, comment pouvez-vous faire confiance aux données produites ? Il existe un papier *On Trusting Trust* à ce sujet, écrit par Ken Thompson (co-inventeur de C, Unix, UTF-8, grep, Go, ...). Voir le fichier `ott.pdf` **si** vous êtes intéressé à en savoir plus.

Dans votre cas, on va faire confiance à cette chaîne, sauf à votre programme. C'est à vous de prouver que l'on peut faire confiance à votre programme et aux valeurs qu'il génère.

4. Banc d'essai

Vous devez pour cette partie, écrire des vecteurs de tests pour les 6 modules du laboratoire précédent. Le module du *pipeline* étant inutile, vous devez l'enlever de votre `top.vhd` et de vos tests.

4.1 Comment tester

Une façon de tester les modules est de générer des tables pour chaque module. Prenons le cas du module `ch`. Ce module comporte trois entrées et une sortie, toutes sur 32 bits. Ainsi, il faudrait 4 tables.

Il faut maintenant décider quelles valeurs tester. La fonction `ch` possède $(2^{32})^3$ permutations des entrées. Dans un vrai système, on pourrait décider de tester les 7.92E+28 permutations durant la fin de semaine et vérifier le résultat au retour, ou peut-être pas... Dans notre cas, on va choisir un sous-ensemble. Le plus important est de tester les cas spéciaux, par exemple `ch(x, x, x)`, ainsi qu'une plage de valeur. Cette dernière plage de valeur devrait être aléatoire pour s'assurer que ce n'est pas par chance que le module fonctionne.

En résumé, pour chaque module, vous devez tester 64 cas. Dans ces 64 cas, il devrait y avoir des cas spéciaux et des valeurs aléatoires.

4.2 Génération

À moins que vous vouliez écrire à la main et calculer dans votre tête des centaines de valeurs, il vaut mieux placer sa confiance en un programme que sur vos habiletés arithmétiques. Vous devez donc réécrire les 6 fonctions dans un langage de votre choix et générer les vecteurs de test.

Il est conseillé d'utiliser un langage proche de la machine, comme `C/C++` et non interpréter comme `Python`. Vous pourriez cependant utiliser une librairie en `Python` qui permet de faire des calculs binaires, par contre vous venez de rallonger la chaîne de confiance.

4.3 Aide

Voici des informations qui vont vous être utiles.

$$ROTR^n(x) = (x << (32 - n)) \vee (x >> n)$$
$$SHR^n(x) = (x >> n)$$

```c
#include <stdint.h>

typedef uint32_t u32;

/* Ceci est un exemple */
u32 ch(u32 x, u32 y, u32 z)
{
        return x * y * z;
}
```

FIGURE 4.1 – Comment utiliser un entier 32 bits en C

4.4 Livrables

- Votre code source du programme générateur avec le nom des autrices et leur matricule, avec optionnellement une licence en commentaire. Votre devez faire référence à votre programme dans votre rapport, mais nul besoin de l'expliquer si celui-ci est lisible. Vous devez aussi expliquer quels sont les cas spéciaux que vous avez choisis pour chaque module.
- Votre fichier de banc d'essai avec 64 tests différents pour chaque module (1 point par table). **NOTE !** Certaines de vos valeurs vont être testées pour vérifier que vous n'avez pas généré n'importe quoi.
- Des captures d'écran du chronogramme de votre simulation avec le nom des ports de vos modules seulement. Il ne faut pas voir les noms des signaux internes à vos modules ! Accompagner vos captures d'écran d'un numéro de figure, un titre et d'un court paragraphe résumant les figures.

5. Questions

Ces questions sont à répondre dans votre rapport.

a) (1) De quelle façon pourriez-vous déterminer que votre programme générateur de tables fonctionne ? En d'autres mots, pourquoi faites-vous confiance en votre programme pour générer les bonnes valeurs ?

b) (0.5) Copier/coller les valeurs générées est une tâche fatigante. De quelle(s) façon(s) pourrait-on faire un banc d'essai, sans avoir à copier/coller de nouvelles tables ?

c) (0.5) Quel(s) avantage(s) (autre que celui mentionné plus haut) offre la génération de valeurs de tests aléatoires ?

6. 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.

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

NOTE ! Il n'y a pas de description du système ou des ressources utilisées, ni de discussion. Seulement une introduction, une présentation de votre banc d'essai et les réponses aux questions.

7. Barème

Critères	**Points**
Partie 4 : Banc d'essai	16
Partie 5 : Questions	2
Rapport : Présentation et qualité de la langue	2
Total	20