

Département de génie informatique et génie logiciel

INF3500

Conception et réalisation de systèmes numériques

**Rapport de laboratoire #3**

**Banc d’essais**

|  |  |
| --- | --- |
| **Critères** | **Points** |
| Banc d’essai | /16 |
| Questions | /2 |
| Rapport : Présentation et qualité de la langue | /2 |
| Total | /20 |
|  |  |
|  |  |

**Soumis par :**

Alexandre Morinvil, #1897222

Nicolas Valenchon, #2032097

**Date :**

18 février 2020

# Table des matières

[Table des matières 2](#_Toc32838737)

[1 Objectifs 3](#_Toc32838738)

[2 Vérification par simulation 4](#_Toc32838739)

[2.1 Stratégie de simulation 4](#_Toc32838740)

[2.2 Description des résultats obtenus 4](#_Toc32838741)

[3 Réponses aux questions 5](#_Toc32838742)

[3.1 Question 1 5](#_Toc32838743)

[3.2 Question 2 5](#_Toc32838744)

[3.3 Question 3 5](#_Toc32838745)

[4 Références 5](#_Toc32838746)

# Objectifs

Les objectifs de ce laboratoire étaient de confirmer notre compréhension de la conception et l’utilisation de banc d’essai. Pour ce faire, le laboratoire a consisté à programmer un court logiciel générant des vecteurs de test spécifiques aux modules à vérifier. Ils sont ensuite utilisés pour concevoir des bancs d’essais afin de vérifier le fonctionnement des modules en VHDL. Ainsi, concrètement, ce laboratoire vise deux buts spécifiques, soient : apprendre à générer des vecteurs de test et comprendre l’importance des tests.

# Vérification par simulation

## Stratégie de simulation

Pour créer notre banc d’essai, nous avons décidé de prendre quelques cas particuliers, mais beaucoup de valeur aléatoire. Pour la fonction ch et maj, les cas particuliers sont toutes les combinaisons possibles si chaque entrée n’avait que deux valeurs possibles : 0 ou la valeur maximale, à savoir 232-1 (0xffffffff). On a ainsi 8 cas particuliers pour ces deux fonctions. Pour les fonctions sigma, on se contente de deux cas particuliers qui sont les valeurs extrêmes.

Pour améliorer notre générateur, nous aurions pu vérifier que deux valeurs n’étaient jamais identiques. Mais la probabilité que cela arrive étant très faible, cela ne nous a pas semblé nécessaire.

## Description des résultats obtenus

# Réponses aux questions

## Question 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?**

On pourrait calculer à la main certaines valeurs précise et vérifier si le résultat est bon. On peut aussi analyser des valeurs générées aléatoirement, et voir le résultat qui y est associé, et refaire les calculs associés.

## Question 2

**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?**

Pour automatiser cette tâche, il pourrait être intéressant de modifier directement depuis le logiciel développé les fichiers du banc d’essai. Il est aussi possible de faire cela au travers d’un script, mais le logiciel peut sans trop de difficulté réaliser cette tâche.

## Question 3

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

La génération de valeur aléatoire permet d’augmenter l’intervalle de confiance dans le module testé : si un grand ensemble de valeur, n’ayant aucun lien entre elles, fonctionne, la probabilité qu’il y ait un bug dans le module est faible. Cela permet aussi de tester des cas auxquels nous n’aurions pas pensé.

# Références