

UNIVERSITÉ PAUL SABATIER

TOULOUSE III

M2 EEA

SME

Synthèse et mise en oeuvre des systèmes

Cahier de TP de VHDL

3 septembre 2022

# TP 1

## Initiation au VHDL

### 1 Introduction

Le but de ce TP est de prendre en main le langage VHDL afin de maîtriser ses fondamentaux. Ce TP a pour but aussi de prendre en main le logiciel Quartus qui permet de mettre en œuvre des solutions sur les FPGA Altera.

### 2 Logique combinatoire

#### 2.1 Porte ET

Voici la table de vérité d'une porte ET à deux entrées :

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 1 – Table de vérité d'une porte ET

- Créer un projet Quartus.
- Dessiner sur une feuille l'entité de la porte ET en faisant apparaître les entrées *A*, *B* et la sortie *S*.
- Décrire l'entité **and\_gate** de la porte ET en VHDL en utilisant l'entité que vous avez dessiné.
- Implémenter l'architecture **rtl\_and\_gate** de la porte ET en VHDL à l'aide de la table de vérité.
- Simuler et vérifier le bon fonctionnement de la porte ET grâce à la table de vérité.
- Connecter les entrées A et B aux boutons *KEY1* et *KEY0* et la sa sortie S à la LED *LEDR0*. Compiler, programmer et vérifier son fonctionnement.
- Pourquoi la LED *LEDR0* est allumée alors qu'aucun bouton n'est appuyé ?

## 2.2 Décodeur 7seg

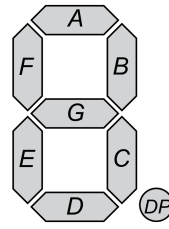


FIGURE 1 – Segments d'un afficheur

Un afficheur sept segments est utilisé pour afficher des informations vers un utilisateur en allumant une partie des segments. Par exemple, pour afficher le chiffre 2, on allume les segments  $A$ ,  $B$ ,  $D$ ,  $E$ ,  $G$ .

- On veut afficher sur l'afficheur sept segments, les chiffres allant de  $0_h$  à  $F_h$ . Combien de bits il faut en entrée et en sortie pour pouvoir représenter ces chiffres.

- Dessiner sur une feuille l'entité du décodeur sept segments en faisant apparaître l'entrée  $x$  qu'on veut decoder et la sortie  $y$  qui contient les segments qu'on veut allumer et éteindre en fonction l'entrée  $x$ . Faire apparaître les entrées/sorties vectorisées.

- Décrire la table de vérité de l'afficheur sept segments pour les chiffres allant de  $0_h$  à  $F_h$  à l'aide de l'entité que vous avez dessiné précédemment et de la figure 1.

- Décrire l'entité **seven\_segment** et l'architecture **rtl\_seven\_segment** de votre table de vérité en VHDL.

- Simuler et vérifier le bon comportement de votre table de vérité.

- Connecter les entrées de votre décodeur sept segments aux switchs ( $SWx$ ) et les sorties aux afficheurs sept segments ( $HEXx$ ) de la carte DE2. Tester et vérifier le bon fonctionnement.

- Expliquer pourquoi les bons segments sont éteints ? Instancier le composant *seven\_segment* à un module top level ou faire les modifications nécessaires au composant *seven\_segment* pour que les bons segments s'allument.

## 3 Logique séquentielle

### 3.1 Compteur

Un des blocs les plus utilisés en VHDL est le compteur. Ils sont très utiles dans plusieurs circonstances. Par exemple : compter des fronts, avoir des bases de temps, compter des événements ...

Voici la table de vérité d'un simple compteur :




$arst\_n$	$clk$	$srst$	$en$	$q$
0	*	*	*	0
1		1	*	0
1		0	1	$q + 1$
1		0	0	$q$

TABLE 2 – Table de vérité d'un compteur

- Dessiner sur une feuille l'entité du compteur en se basant sur la table de vérité 2. Pour la sortie  $q$  spécifier **N** comme taille du vecteur.

- Décrire l'entité **counter** et l'architecture **rtl\_counter** de cette table de vérité en VHDL en utilisant un *process* et en **respectant bien la priorité**. Contraindre le vecteur  $q$  sur **8 bits**.
- Simuler votre compteur et vérifier que votre implémentation respecte bien la table de vérité.
- Ajouter le paramètre générique  $N$  qui permettra de choisir la taille du vecteur  $q_n$  lorsqu'on instancie ce compteur.
- Simuler votre compteur pour  $N = 7$ . A quelle valeur votre compteur reboucle ? Pourquoi ?

### 3.2 Cascade de compteurs

On veut faire un compteur qui s'incrémente toutes les secondes. On veut visualiser le contenu de ce compteur sur un afficheur sept segments. Son contenu sera affiché en hexadécimal. Ce compteur devra compter de  $0_d$  à  $15_d$ .

- Dessiner sur une feuille l'entité du composant qui a été décrit dans l'énoncé. Ce composant aura pour rôle d'être un composant de top level. Il regroupera toutes les instances ainsi que la logique de la question suivante.

- Faire un schéma fonctionnel en faisant apparaître deux compteurs en cascades.

**NB : Il est interdit de connecter aux entrées horloges autres que des signaux horloges.**

- Implémenter votre schéma fonctionnel en VHDL.
- Simuler votre schéma fonctionnel. Pour éviter de simuler une seconde, faire incrémenter le compteur toutes les millisecondes au lieu de toutes les secondes.
- Intégrer un analyseur logique (Signal Tap Logic Analyzer) sur votre FPGA et visualiser le signal  $q$  des compteurs ainsi que le signal  $en$  du second compteur.
- Compiler et programmer. En utilisant l'analyseur logique, déclencher l'acquisition lorsque  $en$  vaut 1. Vérifier que  $en$  est à 1 que pendant un seul coup d'horloge.
- Ajouter une contrainte de temps à votre projet pour que le synthétiseur détermine la fréquence maximale de votre design. Quelle est la fréquence maximale que le synthétiseur a déterminée pour votre design ?

### 3.3 Machine à états

On veut modéliser un composant qui va commander un compteur.

Lorsqu'on appuie sur le bouton incrémenter, cela incrémentera le compteur de 1.

Lorsqu'on appuie sur le bouton décrémenter, cela décrémentera le compteur de 1.

Si les deux boutons sont appuyés au même temps, rien ne se passe.

Le contenu du compteur sera visualisé sur un afficheur sept-segments en hexadécimal.

- Dessiner sur une feuille l'entité du composant qui a été décrit dans l'énoncé.
- Quelle fonctionnalité faut-il ajouter au compteur de l'exercice 3.1 pour pouvoir répondre au besoin ci-dessus. Compléter la table de vérité 2. Ajouter cette fonctionnalité au compteur existant. Simuler le compteur.
- Modéliser la machine à états qui implémente la fonctionnalité d'un bouton. Créer le composant *fsm\_btn* et implémenter la machine à états que vous avez modélisé. Simuler le composant.
- Faire un schéma fonctionnel du problème en incluant le composant *fsm\_btn* et qui intègre les deux boutons. Implémenter votre schéma fonctionnel en VHDL. Simuler.
- Compiler et valider le bon fonctionnement sur la carte.
- Ajouter une contrainte de temps à votre projet pour que le synthétiseur détermine la fréquence maximale de votre design. Quelle est la fréquence maximale que le synthétiseur a déterminée pour votre design ?

## 4 Mini projet - Horloge

L'horloge n'affichera que les **heures** et les **minutes**. L'affichage des dizaine et unités des heures et des minutes se fera sur afficheur sept segments chacun.

L'horloge aura deux modes de fonctionnement. Un mode normal où l'heure compte normalement et un mode configuration où on pourra venir modifier les heures et les minutes.

Si on appuie sur *KEY1*, on rentre en mode configuration. L'horloge arrête de compter tant qu'on est dans le mode configuration.

Dans le mode configuration lorsqu'on appuie sur *KEY0* ou *KEY2*, on incrémentera ou décrémentera respectivement soit les heures soit les minutes.

Lorsqu'on entre en mode configuration, ce sont les minutes qui pourront être modifiées. Lors d'un nouvel appui sur *KEY1* et ce seront les heures qui pourront être modifiées. Un nouvel appui sur *KEY1*, on revient en mode normal.

- En utilisant les composants des exercices précédents et en modélisant des nouveaux, faire un schéma fonctionnel et les machines à états nécessaires pour modéliser l'horloge.

- Implémenter votre modélisation en VHDL.

- Valider votre implémentation en simulation.

- Valider votre implémentation sur votre carte.

- Ajouter une contrainte de temps à votre projet pour que le synthétiseur détermine la fréquence maximale de votre design. Quelle est la fréquence maximale que le synthétiseur a déterminée pour votre design ?

# TP 2

## Utilisation du processeur Nios

### 1 Introduction

Le but de ce TP est de prendre en main l'outil *Platform Designer* afin de pouvoir mettre en œuvre des solutions complexes et ciblées basé sur le processeur *Nios II* et le bus *Avalon*.

### 2 Construction d'un microcontrôleur personnalisé

A l'aide de l'outil *Platform Designer*, construire un microcontrôleur basé sur le processeur *Nios II* en ajoutant les périphériques suivants :

- Nios II
- On Chip RAM (où le code sera stocké)
- JTAG UART (pour avoir une console série + pouvoir debugger)
- PIO (pour pouvoir contrôler des Leds)

Générer le VHDL.

Connecter la sortie du périphérique *PIO* aux Leds de la carte DE0 Nano.

Compiler et téléverser la configuration du FPGA vers votre carte.

En utilisant le logiciel *Nios II Software Build Tools for Eclipse* créer un projet en prenant comme modèle de projet **Hello World Small**.

Sans toucher au code qui a été généré, compiler et téléverser vers le Nios. Vérifier que *"Hello from Nios II!"* s'affiche bien sur la console *Nios II Console*.

Écrire un code qui permet de faire un chenillard va et vient sur les Leds en utilisant le périphérique PIO.

### 3 Ajout d'un périphérique personnalisé au microcontrôleur

#### 3.1 PWM

On veut créer un composant VHDL qui permet de générer un signal PWM d'haute résolution sur une broche. Ce composant on veut l'interfacer avec le processeur NIOS II à l'aide du bus Avalon. Le PWM aura une résolution de **32 bits**.

- Faire un diagramme fonctionnel du composant PWM sans prendre en compte le NIOS ni le bus Avalon.

- Décrire l'entité et implémenter l'architecture en utilisant le diagramme fonctionnel.

- Simuler et vérifier que votre composant crée bien un signal PWM avec la fréquence et le rapport cyclique variable.

### 3.2 Interface avec le bus Avalon

- Créer un composant VHDL qui permet d'interfacer le composant PWM que vous venez créer, avec le processeur Nios II à l'aide du bus Avalon. La table d'adressage doit être la suivante :

31	24	23	16	15	8	7	0	
Duty								00h
Freq								01h

TABLE 2 – Table d'adressage du composant PWM sur le bus Avalon

- En utilisant l'outil *Platform Designer*, ajouter votre composant d'interface que vous venez de créer pour qu'il soit reconnu comme un périphérique Avalon.

- Intégrer ce composant au microcontrôleur que vous avez créé dans l'exercice 2 et exporter la sortie du composant PWM. Générer le VHDL du système.

- Affecter la sortie du PWM à une broche quelconque.

- Compiler et téléverser la nouvelle configuration du FPGA vers votre carte.

- Avec *Nios II Software Build Tools for Eclipse*, tester votre périphérique en changeant le rapport cyclique et la fréquence. Valider le fonctionnement à l'aide d'un oscilloscope.