

TP – FPGA1

SYSTEMES PROGRAMMABLES

1^{ERE} PARTIE – SYNTHÈSE VHDL SUR FPGA

Le double objectif de ce TP est de :

- Prendre en main la chaîne de conception **Xilinx Vivado** ainsi que la carte **FPGA Nexys4** sur laquelle nous travaillerons.
- De comprendre comment fonctionne l'outil de synthèse de Vivado et de quelle façon il va interpréter un code VHDL imprécis ou erroné.

I) Prise en main de la carte et des outils

1) Présentation de la carte Digilent Nexys4

Les cartes de développement **Nexys4** et **Nexys4-DDR** de Digilent disposent (notamment) des éléments suivants

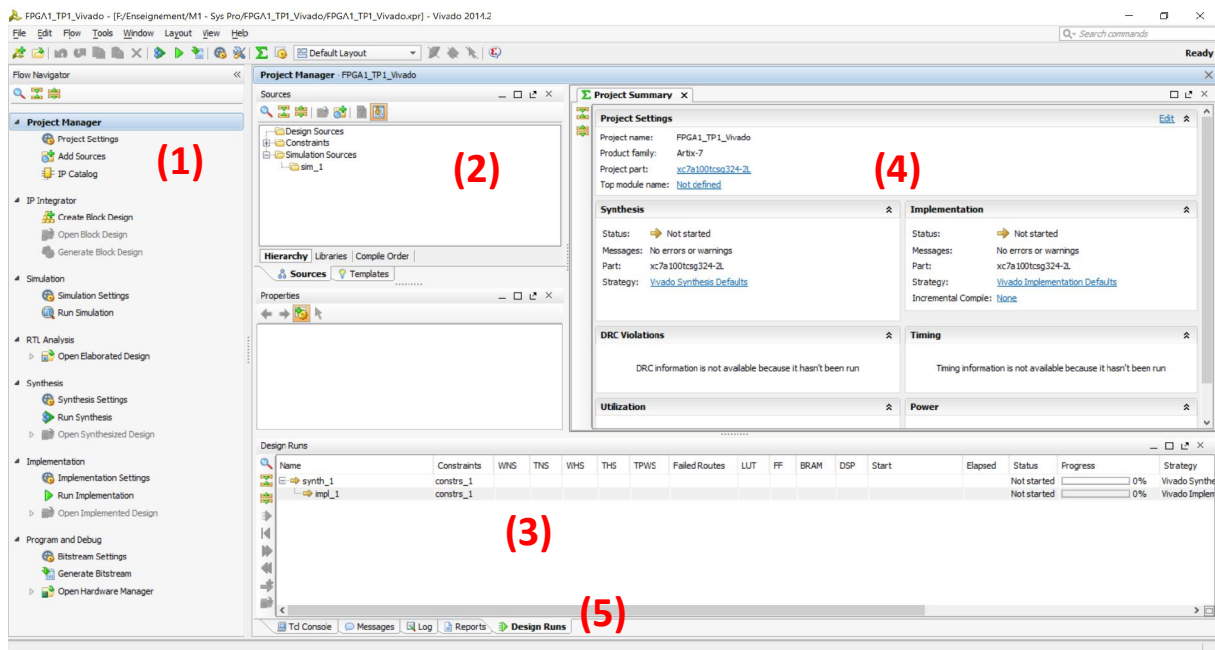
- **FPGA Xilinx Artix-7** : C'est le composant principal. Ce circuit reconfigurable nous permettra d'implémenter les systèmes numériques des TP à venir.
- **Oscillateur** : La carte comporte un oscillateur externe permettant de fournir au FPGA une horloge de fréquence 100 MHz.
- **Interrupteur ON/OFF** : Self-explanatory...
- **Connecteur USB** : L'USB permet d'alimenter la carte et de programmer le FPGA. Le connecteur sert également de port JTAG et UART qui seront utilisés par la suite.
- **Connecteurs PMOD** : ils permettent de connecter la carte Nexys4 à d'autres systèmes et d'amener des signaux d'entrées/sorties au FPGA.

2) Création d'un projet avec le logiciel Vivado

- Avant d'ouvrir le logiciel, recopier le répertoire **I:\denouletj\4I108\FPGA1_TP1** sur votre compte (disque H). Il contient tous les fichiers sources nécessaires au TP.
- Ouvrir le logiciel Xilinx Vivado. (**Menu Démarrer → Tous les Programmes → Xilinx Design Tools → Vivado 2014.2 → Vivado 2014.2**).
- Cliquer sur **Create New Project** puis **Next**. Choisir un nom pour le projet et choisir comme répertoire **C:/Users/1234567** (si 1234567 est votre numéro d'étudiant). Puis cliquer deux fois sur **Next**.
- Dans la fenêtre suivante pour choisir le FPGA, sélectionner les filtres suivants :
 - Family : **Artix-7**
 - Sub-Family : **Artix-7**
 - Package : **CSG324**
 - Speed : **-3**

Sélectionner le FPGA **xc7a100** puis cliquer sur **Next** puis **Finish**.

A l'écran s'affiche alors un ensemble de fenêtres similaire à la figure ci-dessous.



(1) **Flow Navigator** : Permet de démarrer les différentes phases du flot de conception, de la création des sources VHDL à la simulation, l'implémentation et jusqu'à la programmation du FPGA.

(2) **Sources** : Liste les différentes sources (VHDL, Testbenchs, Fichier de contraintes, etc...) du projet ainsi que leur hiérarchie.

- (3) **Console/Messages** : Indique les messages générés par Vivado. L'onglet **Reports** permet d'accéder aux rapports de synthèse et d'implémentation.
- (4) **Editeur** : Permet d'éditer les sources du projet ou de consulter les rapports d'implémentation.

3) Création d'un module VHDL

- Dans le **Flow Navigator**, cliquer sur **Add Sources**. Choisir **Add or Create Design Sources**, puis cliquer sur **Next**. Cliquer sur **Create File**, puis choisir **VHDL** comme type de fichier et donner comme nom **Test** au fichier. Cliquer sur **Finish**.
- Indiquer ensuite les entrées/sorties du module VHDL: 3 ports d'entrée appelés **SW2**, **SW1**, **SW0** et un port de sortie sur 3 bits appelé **LED(2 :0)**. Terminer la création de la source VHDL.
- Dans la fenêtre **Sources**, double cliquer sur **Test.vhd** pour ouvrir le fichier dans la fenêtre **Editeur**.
 - Vous constaterez que l'entité du module VHDL a déjà été pré-remplie conformément aux indications que vous avez fournies précédemment.
- Ecrire l'architecture du module **Test**.
 - La **LED(0)** prend la valeur de **SW0**.
 - La **LED(1)** prend la valeur de **SW1**.
 - La **LED(2)** prend la sortie d'un ET logique entre les 3 interrupteurs.

4) Testbench et simulation avec Modelsim

- Pour créer un fichier Testbench, cliquer à nouveau sur **Add Sources** et choisir **Add or Create Simulation Sources**. Choisir **Add Files**. Aller chercher le fichier **TB_Test.vhd** dans le répertoire **FPGA1_TP1**. Vérifier que l'option **Copy sources into project** est cochée puis cliquer sur **Finish**.
 - Dans la fenêtre **Sources**, dérouler le répertoire **Simulation Sources** puis **sim_1**. Vérifier que le fichier **TB_Test** est bien présent et que le fichier **Test** est bien associé au Testbench.
- Cliquer sur **Run Simulation** puis **Run Behavioral Simulation**.
 - Si jamais le code VHDL comporte des erreurs, corrigez-les à l'aide des informations de la **Console**.
 - Avec les commandes de zoom à gauche du chronogramme, visualiser l'ensemble de la simulation et vérifier le bon comportement de l'architecture.

5) Implémentation sur la carte FPGA

Pour implémenter l'architecture dans le FPGA, il faut ajouter un fichier de contraintes au projet. Ce fichier indique notamment sur quelles broches du circuit mapper les entrées/sorties du code VHDL.

- Cliquer sur **Add Sources** puis **Add or Create Constraints** pour ajouter le fichier **Test_Nexys4.xdc** (ou **Test_Nexys4DDR.xdc**) du répertoire **FPGA1_TP1**.
- Ouvrir le fichier **Test.xdc** en déroulant dans la fenêtre **Sources** le répertoire **Constraints**.
Noter la correspondance des noms de ports avec ceux du code VHDL.
- **IMPORTANT – LE FICHIER XDC EST CASE SENSITIVE A L'INVERSE DES FICHIERS VHDL**
- Cliquer sur **Run Synthesis**.
 - Cette procédure analyse les sources VHDL et les transforme en cellules logiques de base.
- Lorsque la synthèse est terminée, cliquer sur **Run Implementation**
 - Au cours de cette phase, Vivado va placer dans le FPGA les cellules logiques identifiées lors de la synthèse, conformément aux contraintes données par le fichier **XDC**. Les cellules placées sont ensuite routées les unes avec les autres.
- A la fin de l'implémentation, cliquer sur **Generate Bitsream** pour générer le fichier de configuration du FPGA.
- Cliquer ensuite sur **Open Hardware Manager** pour programmer le FPGA.
 - Connecter la carte au port USB du PC et allumer la carte avec l'interrupteur ON/OFF
 - Dans le bandeau vert en haut de l'écran, cliquer sur **Open a new hardware target**.
 - Cliquer trois fois sur **Next** puis **Finish**.
 - En bas du **Flow Navigator**, cliquer sur **Program Device** puis **Program**.
- Le FPGA est à présent programmé.
 - Vérifier en jouant sur les interrupteurs que les LEDs s'allument conformément au code VHDL.

II) Cas d'études – Synthèse VHDL.

Dans les parties suivantes, nous allons travailler avec plusieurs applications dont le code VHDL vous sera fourni. Nous allons voir en quoi ces architectures sont bien ou mal interprétées par l'outil de synthèse d'ISE et, le cas échéant, comment écrire un code qui soit parfaitement compréhensible et synthétisable par l'outil.

- L'architecture 1) propose deux compteurs imbriqués dont la valeur des poids forts est affichée sur les LEDs.
- Les architectures 2) 3) et 4) réalisent un système d'allumage/clignotement de LEDs selon les impulsions sur les boutons de la carte.

1) Compteurs Imbriqués

❖ Observation

- Reprendre le projet précédent en enlevant toutes les sources (sélectionner toutes les sources, puis cliquer avec le bouton droit et choisir **Remove File from project**)
- Ajouter au projet les fichiers **Test_CPT.vhd** et **Test_CPT_Nexys4.xdc** (ou **Test_CPT_Nexys4DDR.xdc**)
 - o Quelle est la fonctionnalité décrite dans le VHDL?
 - o A quel résultat peut-on s'attendre sur la carte ?
- Implémenter le design.
 - o Astuce : vous pouvez cliquer directement sur Generate Bitstream pour faire d'un trait toutes les étapes de l'implémentation. (Le flot s'arrêtera en cas d'erreur)
- Tester sur la carte. Que se passe-t-il ?

❖ Analyse

- Dans la console, cliquer sur l'onglet **Reports** et afficher le **Vivado Synthesis Report**. Dans la partie **Start Area Optimization**, analyser les messages **INFO** et **WARNING**.
 - o Que comprenez-vous ? Y a-t-il quelque chose qui vous semble anormal ?
 - o Y a-t-il des modules logiques qui sont supprimés lors de l'optimisation du design par l'outil de synthèse ?
 - o Cela peut-il expliquer le dysfonctionnement de l'architecture portée sur la carte ?
- Dans le Flow Navigator, dérouler la ligne **Synthesized Design** et cliquer sur **Schematic** pour générer un schéma RTL de l'architecture synthétisée.
 - o Est-ce que cela correspond au code VHDL de départ ?
 - o Faites le lien avec les messages de **WARNING** du rapport de synthèse.

- Aller dans le rapport de synthèse et afficher le message d'**INFO** portant sur le signal **Start**
 - o En étudiant le code VHDL, comment peut-on expliquer cela ?
 - o En déduire l'erreur qui s'est glissée dans le code VHDL

❖ Correction et vérification

- Une fois l'erreur corrigée, refaire une synthèse et ouvrir le rapport.
 - o Vérifier en fin de rapport qu'il n'y a plus de warnings.
 - o Recharger le schéma RTL de l'architecture synthétisée pour constater la différence
- Poursuivre l'implémentation et porter le design sur la carte. Valider le bon fonctionnement de l'architecture.

2) Compteur d'Impulsions

❖ Observation

- Retirer toutes les sources de votre projet et ajouter les fichiers **Impulse_Count.vhd** et **Impulse_Count_Nexys4.xdc** (ou **Impulse_Count_Nexys4DDR.xdc**)
 - o Quelle est la fonctionnalité décrite par le code VHDL ?
- Ajouter au projet le testbench **TB_Impulse_Count.vhd**.
 - o Analyser le Testbench pour savoir ce que l'on va effectuer dans la simulation.
 - o Simuler et vérifier le bon comportement de l'architecture.
- Implémenter et vérifier le bon comportement sur la carte. Que constatez-vous ?
 - o NB : il y a un phénomène de rebond sur les boutons de la carte qui peut entraîner plusieurs fronts du signal pour un seul et même appui sur le bouton.

❖ Analyse

- Regarder les warnings générés en synthèse et générer le **Schematic** du système synthétisé.
 - o Quelle différence y a-t-il entre la description VHDL et l'architecture synthétisée ?
- Dans le **Flow Navigator**, cliquer sur **Project Settings** et vérifier que le **Target Language** est bien VHDL (sinon, faire la modification).
- Relancer ensuite une simulation, mais en choisissant cette fois la **Post Synthesis Functional Simulation**.
 - o L'outil va simuler un fichier VHDL correspondant à l'architecture synthétisée par **Vivado**. Ce fichier, généré lors de la synthèse est consultable dans le simulateur en cliquant sur le module **UUT** dans la fenêtre **Scopes**.
 - o Ouvrir et regarder la structure du code. Comment est-il composé ?

- Comparer le chronogramme de la simulation post-synthèse avec la simulation pré-synthèse vue précédemment. Le comportement est-il différent ?
- Que constatez-vous entre le comportement de la simulation post synthèse et celui du design testé sur la carte ?.

❖ Correction et vérification

- Comment modifier le code VHDL pour faire en sorte que le compteur d'impulsions fonctionne correctement ?
 - Décrire cette nouvelle architecture en VHDL
 - Modifier le testbench en conséquence puis lancer une simulation pour valider fonctionnellement cette nouvelle description.
- Modifier le fichier de contraintes **XDC** de la façon suivante :
 - Pour la carte **Nexsy4**
 - Décommenter les lignes 8,9,10 sur le port d'horloge en faisant attention que le nom corresponde bien à celui de votre code VHDL.
 - Passer en commentaire la ligne 40 (***set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets Button_C_IBUF]***)
 - Pour la carte **Nexsy4DDR**
 - Décommenter les lignes 7,8 sur le port d'horloge en faisant attention que le nom corresponde bien à celui de votre code VHDL.
 - Passer en commentaire la ligne 29 (***set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets Button_C_IBUF]***)
- Implémenter votre design. Examiner le rapport de synthèse et le schéma RTL pour vérifier la bonne compréhension de votre code par l'outil, puis tester sur la carte.
- Comment gérer le rebond des boutons. Modifier le code VHDL pour cela et tester sur la carte.

3) Décodeur

❖ Observation

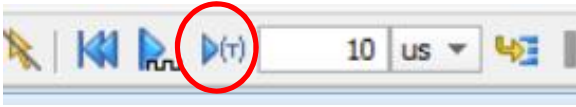
- Supprimer toutes les sources du projet et ajouter le code ***Selector.vhd***
 - Analyser le code pour comprendre quelle est la fonctionnalité du module.
- Ajouter le testbench ***TB_Selector.vhd*** et lancer une simulation
 - A quoi est due l'erreur de compilation? Corriger cette erreur
- Simuler et vérifier le bon fonctionnement du module ***Selector***.
 - Pour cela, vous pouvez cliquer avec le bouton droit dans le chronogramme sur le signal ***Limit*** pour changer sa base de représentation (***Radix***) en ***Hexadecimal***.
- Remettre dans le projet le fichier ***Impulse_Count.vhd***. Ajouter ensuite la source VHDL ***Impulse_Selector.vhd*** et le fichier de contraintes ***Impulse_Selector_Nexys4.xdc*** (ou ***Impulse_Selector_Nexys4DDR.xdc***). Ce code instancie les 2 modules VHDL ***Impulse Count*** et ***Selector*** en connectant les sorties du premier aux entrées du second.
 - Implémenter le design sur la carte
 - D'après le fichier ***XDC***, les LED de la carte afficheront les 16 bits de poids fort de ***Limit***. Donc en fonction de la valeur du compteur d'impulsions, on aura :
 - Count = 0,1,2 → LED = 0000 0000 0000 0000
 - Count = 3,4,5 → LED = 1001 0110 1110 0011
 - Count = 6,7,8,9 → LED = 1000 0111 1010 0001
 - Count > 9 → LED = 1111 1111 1111 1111
 - Une fois la carte programmée, appuyer sur le bouton Right. Le compteur étant initialisé à 0, les LED devraient normalement toutes s'éteindre. Que constatez-vous?

❖ Analyse et Correction

- Trouver la cause de l'erreur en vous aidant des rapports de synthèse.
- Modifier le code pour résoudre le problème.
- Implémenter à nouveau le design et vérifier le bon fonctionnement.

4) Machine à Etat

❖ Observation

- Ajouter le code **FSM.vhd** au projet. Ce code VHDL implémente une machine à états (**MAE**) qui prend en entrée deux informations :
 - o **Mode** qui contrôle le fonctionnement des LEDs (Allumées, éteintes, clignotement),
 - o **Seuil** qui fixe la fréquence de clignotement.
- Analyser le code, puis simuler la **MAE**.
 - o Pour cela, ajouter le testbench **TB_FSM.vhd**.
 - o Dans la fenêtre **Sources**, dans le répertoire **Simulation Sources**, cliquer avec le bouton droit sur **TB_FSM.vhd** et choisir **Set as Top**. Lancer le simulateur.
 - o Cliquer sur l'icône ci-dessous pour faire avancer la simulation de 10 µs.
 - o Vérifier le bon fonctionnement du code.
- Retirer **Impulse_Selector.vhd** des sources du projet et ajouter le code **Top.vhd** qui assemble les modules **Impulse_Count**, **Selector** et **FSM**.
 - o Les entrées **Seuil** et **Mode** de la **MAE** sont constituées à partir du vecteur **Limit** issu de **Selector**.
- Remplacer également **Impulse_Selector_Nexys4.xdc** par **Top_Nexys4.xdc** (ou **Impulse_Selector_Nexys4DDR.xdc** par **Top_Nexys4DDR.xdc**). Implémenter le design et tester sur la carte.
 - o Appuyer 3 fois sur le bouton Left puis 1 fois sur le bouton Right, ce qui doit entraîner un clignotement des **LEDs**. Que constatez-vous ?

❖ Analyse et Correction

- Ouvrir le rapport de synthèse et repérer les warnings..
 - o Quelle erreur dans le code a pu provoquer cela ?
- Corriger le code de **FSM.vhd** et implémenter à nouveau.
 - o Vérifier dans le rapport de synthèse que le ou les warnings ont disparu.
 - o Valider sur la carte le bon fonctionnement du design