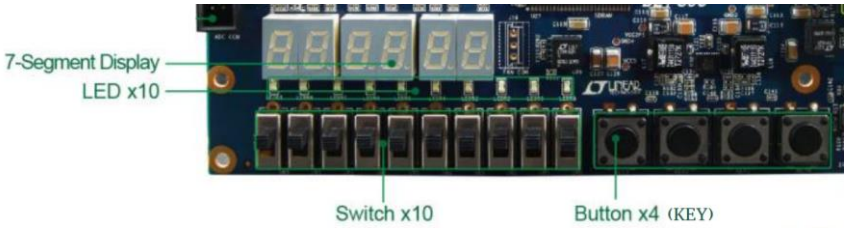
IP AXI4-lite avec I/O de la FPGA

Objectif :

Ce laboratoire a pour but de réaliser une IP avec une interface AXI4-lite et connectée sur le bus Lightweight HPS-to-FPGA. Cette IP doit permettre d'accéder à des I/O câblées sur la partie FPGA via des registres. Vous devrez analyser le fonctionnement du bus AXI4-lite afin de concevoir une IP personnalisée pour les besoins du laboratoire.

Spécifications :

L'objectif est d'interfacer à l'aide d'une IP AXI4-lite tous les I/O disponibles sur la FPGA, sans utiliser des composants PIO, soit les boutons (KEYs), les switchs (SW), les LEDs et les afficheurs 7 segments.



Votre IP AXI4-lite comprendra une constante 32 bits à l'offset 0x0 ainsi qu'un registre de test R/W à l'offset 0x4. Les offsets sont relatifs à l'adresse de base donnée à l'instance de l'IP dans Qsys.

Spécifications du programme:

Le but est d'allumer les LEDs selon l'état des boutons (KEYs) et interrupteurs (switch) disponibles. Les afficheurs 7 segments sont dépendants de la constante définie dans l'IP. La spécification du fonctionnement est la suivante :

- Appui sur KEY0 : l'états des switches est copiés sur les LEDs. Les afficheurs HEX5 à HEX0 affichent en hexadécimal les bits 23 à 0 de la constante définie dans l'IP.
- Appui sur KEY1 : l'états **inverses** des switches est copiés sur les LEDs. Les afficheurs HEX5 à HEX0 affichent en hexadécimal **l'inverse** des bits 23 à 0 de la constante définie dans l'IP.
- Appui sur KEY2 : l'affichage des LEDs et des afficheurs 7 segments subit une rotation à droite. Rotation d'un bit pour les LEDs, rotation d'un afficheur complet pour les afficheurs 7 segments.
- Appui sur KEY3 : l'affichage des LEDs et des afficheurs 7 segments subit une rotation à gauche. Rotation d'un bit pour les LEDs, rotation d'un afficheur complet pour les afficheurs 7 segments.

Dans une seconde partie, l'appui sur les boutons KEY2 ou KEY3 devra être géré à l'aide d'interruption vers le HPS (2ème partie).

- Votre design doit permettre de générer une interruption lors de l'activation (détection de flanc) d'un des 4 boutons. Vous devez prévoir les accès et les flags nécessaires pour gérer l'interruption. Il doit être possible d'activer/masquer l'interruption pour chaque bouton.

Documents à rendre :

Vous devrez rendre un rapport à l'issu de ce laboratoire contenant les explications sur les différentes étapes de la réalisation de votre système.

Vous devez également rendre une archive avec les sources du projet pour Quartus, Qsys et le programme C. Utiliser le Makefile fourni pour générer votre archive à rendre en tapant "make zip" dans un terminal.

Les fichiers sont à rendre sur Cyberlearn (Moodle HES-SO)

Travail demandé :

1ère partie : sans interruption

1) Récupérer l'archive du projet sur Moodle.
2) Etablir le plan d'adressage de votre IP. Réserver l'offset 0x0 pour une constante et l'offset 0x4 pour un registre de test R/W.
3) Concevoir l'interface puis compléter le fichier "axi4lite_slave.vhd" pour répondre aux spécifications du bus AXI4-lite et avec votre plan d'adressage.

   Le fichier est dans le répertoire: *project_folder*/IP_axi4lite_interface/src
4) Tester l'IP afin de valider son fonctionnement. Un testbench basique est disponible dans *project_folder*/IP_axi4lite_interface/tb, il teste uniquement la validité des accès en lecture et écriture, pas le contenu de ce qui est écrit ou lu. Vous pouvez le compléter pour réaliser des tests plus complets si nécessaire.
5) Dans le projet Qsys fourni, créer un nouveau composant (File → New Component) afin de rajouter votre IP à Qsys. Compléter les champs des différents onglets, et notamment :
   - Onglet "Files" : dans la partie Synthesis Files, rajouter le fichier "axi4lite_slave.vhd".
   - Onglet "Signals & Interfaces" : Ajouter une interface "AXI4Lite Slave", "Clock Input", "Reset Input" et "Conduit". Faire glisser les signaux dans l'interface lui correspondant, configurer les interfaces et les signaux associés. Supprimer l'interface Avalon créée par défaut.
6) Ajouter le composant fraîchement créé dans le système Qsys, réaliser les connexions, les exports nécessaires ainsi que l'adressage du composant.
7) Générer les fichiers HDL du projet Qsys. ATTENTION : toutes modifications de l'IP imposent de re-générer les fichiers HDL du projet Qsys avant de faire la synthèse avec Quartus, sous peine que les modifications ne soient pas prises en compte.
8) Adapter le top du projet (DE1_SoC_top.vhd).
9) Synthétiser et faire le placement-routage du projet.
10) Réaliser le code C pour que les LEDs et afficheurs s'allument de manière à respecter les spécifications données précédemment.
11) Créer un nouveau projet Altera Monitor Program.
12) Compiler le code et le tester sur la carte DE1-SoC.

2ème partie : avec interruption

13) Compléter le plan d'adressage pour prendre en compte le fonctionnement avec interruption.
14) Compléter le fichier "axi4lite_slave.vhd" (p*roject_folder*/IP_axi4lite_interface/src) pour générer une interruption lors de l'appui sur une des 4 KEYs et gérer les registres correspondants, le tout en accord avec le plan d'adressage du point 13.
15) Tester l'IP afin de valider son fonctionnement. Il est nécessaire de modifier le testbench si vous souhaitez l'utiliser dans cette partie.
16) Dans le projet Qsys, modifier le composant déjà créé en rajoutant notamment une interface "Interrupt Sender" et ses signaux correspondants.
17) Réaliser les connexions nécessaires dans Qsys.
18) Générer les fichiers HDL du projet Qsys.
19) Synthétiser et faire le placement routage du projet.
20) Compléter le code C pour que les LEDs et afficheurs s'allument de manière à respecter les spécifications du programme tout en utilisant une interruption pour les actions sur les boutons KEY2 et KEY3.
21) Modifier la configuration mémoire de votre projet "Altera Monitor Program" pour qu'il alloue une portion pour les vecteurs d'interruptions.
22) Compiler le code et le tester sur la carte DE1-SoC.