



HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Hes·so

Haute Ecole Spécialisée
de Suisse occidentale

Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Laboratoire 2 : Utilisation des I/O et des interruptions entre la partie FPGA et le HPS

Département : Technologies de l'Information et de la Communication (TIC)
Unité d'enseignement : System on chip on FPGA (SOCF)

Auteurs : Pierrick Muller
Professeur : Alberto Dassati, Etienne Messerli
Assistant : Sébastien Masle, Sydney Hauke
Classe : SOCF-1-A
Date : 2 avril 2020

Table des matières

1	Introduction	2
1.1	Objectifs	2
1.2	Spécifications sans les interruptions	2
1.3	Spécifications avec les interruptions	3
2	Analyse	3
2.1	Partie 1	3
2.2	Partie 2	3
2.3	Partie 3	3
2.4	Adress Map finale	4
3	Réalisation et implémentation	4
3.1	Ajout des PIOs	5
3.2	Activation des interruptions dans Qsys	5
3.3	Programme pour les features sans interruption	5
3.4	Activation des registres du GIC et de la FPGA dans le code	5
3.5	Routine d'interruption	5
4	Simulation et tests	5
5	Conclusion	5
6	Signatures	5

1 Introduction

1.1 Objectifs

Ce laboratoire a pour but d'accéder à des I/O câblées sur la partie FPGA. Il s'agira d'ajouter des blocs PIO pour interfacer ces I/O sur le HPS. Vous devrez comprendre comment ajouter des IP disponibles dans Qsys pour construire des interfaces permettant d'accéder aux I/O de la FPGA depuis le HPS. Dans un deuxième temps, vous utiliserez les interruptions du HPS, qui seront générées par un PIO que vous utilisez déjà. Vous devrez comprendre le mécanisme des interruptions sur la Cyclone V SoC afin de bien gérer la gestion de celles-ci.

1.2 Spécifications sans les interruptions

Le but est d'allumer les LEDs et les afficheurs 7 segments selon l'état des boutons (KEY) et interrupteurs (switch) disponibles. La spécification du fonctionnement est la suivante :

- Appui sur KEY0 : les LEDs s'allument selon la position des différents switches. Les afficheurs HEX0 et HEX1 traduisent en hexadécimal les valeurs représentées par les LED3 à LED0 et LED7 à LED4 respectivement. Les afficheurs

HEX2 et HEX3 affichent 1 lorsque la LED8 et respectivement la LED9 sont allumées, 0 sinon.

- Appui sur KEY1 : les LEDs s'allument selon la position inverse des différents switches. Les afficheurs HEX0 et HEX1 traduisent en hexadécimal les valeurs représentées par les LED3 à LED0 et LED7 à LED4 respectivement. Les afficheurs HEX2 et HEX3 affichent 1 lorsque la LED8 et respectivement la LED9 sont allumées, 0 sinon.

1.3 Spécifications avec les interruptions

Complétez votre travail précédent avec les spécifications suivantes :
L'appui sur KEY2 ou KEY3 génère une interruption permettant de réaliser les fonctions suivantes :

- Appui sur KEY2 : l'affichage des LEDs et des afficheurs 7 segments subit une rotation à droite, les afficheurs 7 segments ne reflètent plus les valeurs des LEDs.
- Appui sur KEY3 : l'affichage des LEDs et des afficheurs 7 segments subit une rotation à gauche, les afficheurs 7 segments ne reflètent plus les valeurs des LEDs.

2 Analyse

2.1 Partie 1

Cette partie était principalement une marche à suivre. Il n'y avait pas réellement d'analyse à effectuer sur cette partie du laboratoire. Le but était de suivre la marche à suivre et de comprendre les actions que nous réalisions.

2.2 Partie 2

Cette partie nous demandais d'ajouter la gestion des leds et des afficheurs 7 segments en fonction des switches et sans gérer pour le moment les interruptions (Key2 et Key3). Ce qui est intéressant dans cette partie, c'est l'ajout des Keys et des afficheurs 7 segments en tant que PIO, en se servant de l'expérience reçue lors de la réalisation de la première partie.

2.3 Partie 3

Cette partie demandais un peu plus d'analyses que les deux parties précédentes. Il n'y avait pas de PIO à ajouter, par contre la gestion des interruptions devait être mise en place pour chacun des différents éléments du système. Ainsi, il fallait prendre en compte le CPU, le GIC et les PIO.

Pour ce qui concerne le CPU, la configuration des vecteurs d'interruptions était

fournie, mais la routine d'interruption devait être implémentée.

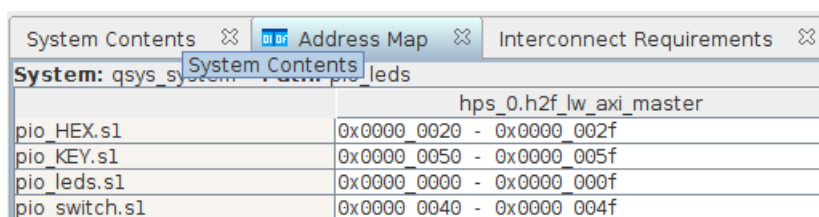
Concernant le GIC, je me suis inspiré des exemples de codes fournis dans le document "Gic_Altera_Manuel_short.pdf", et j'ai dû comprendre comment configurer le GIC. Voici les registres du GIC qui ont dû être configurés ou qui ont été utilisés lors de la routine d'interruption :

- ICCICR : Permet d'activer la transmission depuis le CPU interface jusqu'au CPU lui correspondant. C'est un enable.
- ICCPMR : Permet de set la priorité minimale pour qu'une interruption soit transmise au CPU.
- ICCIAR : Lors d'une interruption, contient l'ID de l'interruption en question.
- ICCEOIR : Permet au processeur de clear l'interrupt en écrivant l'ID correspondant à l'intérieur
- ICDDCR : Permet d'activer le distributor. C'est un enable.
- ICDISER : Permet d'activer une interruption (unmask)
- ICDIPTR : Permet de définir vers quel CPU doit être envoyé l'interruption.

A cela s'ajoutait deux registre , KEYS_INTERRUPT_ENABLE et KEY_INTERRUPT_REGISTER, qui permettait d'activer les interruptions des keys dans la FPGA et de récupérer ainsi que nettoyer lesdites interruptions.

Nous parlerons des valeurs qui ont été attribué à chaque registres dans la partie implémentation. Les adresses des registres sont basés sur les calculs fournis dans la documentation en fonction du numéro d'interruption 72, qui correspond au numéro d'interruption 0 de la FPGA (Nous verrons dans la partie implémentation d'ou vient ce numéro).

2.4 Adress Map finale



System: qsys_system	hps_0.h2f_lw_axi_master
pio_HEX.s1	0x0000_0020 - 0x0000_002f
pio_KEY.s1	0x0000_0050 - 0x0000_005f
pio_leds.s1	0x0000_0000 - 0x0000_000f
pio_switch.s1	0x0000_0040 - 0x0000_004f

FIGURE 1 – Laboratoire 2 : Address map

Le choix des offsets ci-dessus vient à la base d'un problème de compréhension. J'avais choisi ces offsets en fonction des adresses correspondantes dans le memory layout fournit dans le document "DE1-SoC_Computer_ARM.pdf". Si cela n'apporte pas de problème en tant que tel, il faut tout de même noter que j'aurais pu en choisir d'autres, en faisant en sorte qu'ils soient contiguë par exemple.

3 Réalisation et implémentation

J'ai choisis de ne pas présenter l'implémentation faite partie par partie, mais de présenter l'implémentation finale car elle contient les éléments correspondants à

chacune des parties.

3.1 Ajout des PIOs

3.2 Activation des interruptions dans Qsys

3.3 Programme pour les features sans interruption

3.4 Activation des registres du GIC et de la FPGA dans le code

3.5 Routine d'interruption

4 Simulation et tests

5 Conclusion

6 Signatures

Yverdon-les-Bains le 2 avril 2020

Pierrick Muller

Table des figures

1	Laboratoire 2 : Address map	4
---	---------------------------------------	---