Compléments au labo 2 INF3610

1- Erratum

- Dans le code source fourni au début du laboratoire, une erreur s'est introduite. Il faut simplement supprimer la ligne 20 :

  #define VERIF_IRQ_ID XPAR_AXI_INTC_0_IRQ_GEN_1_IRQ_INTR

2- Addenda

- Pour le démarrage du projet, veillez à bien démarrer le software development kit de la **version 14.4** de xilinx. Le projet ne compilera pas avec la version 15.2
- Lorsque vous utilisez la fonction *xil_printf()* pour afficher des messages sur la console, soyez conscient que le buffer ne sera vidé que lorsqu'il rencontrera un symbole de fin de ligne (*n*). Je vous recommande donc de finir tous vos messages par ce symbole afin d'éviter tout problème.
- Lisez le paragraphe sur les interruptions dans la partie 5 du labo pour mieux comprendre le fonctionnement de celles-ci. Faites bien attention à ne pas confondre le contrôleur d'interruptions générique (GIC) dont l'instance dans le code est nommée *m_gic* **et** le contrôleur d'interruptions matériel ajouté sur le FPGA dont l'instance s'appelle *m_axi_intc*. Vous n'avez pas besoin de travailler avec le GIC puisqu'il est déjà intégralement configuré. Le seul contrôleur que vous devez manipuler est le axi_intc.

3- API linux

Nous vous avons fourni dans le code quelques scripts permettant de lancer automatiquement les différentes routines linux qui vous sont données. Toutefois il est possible que vous ayez besoin d'exécuter ces routines une par une lors d'un scénario de débogage. Voici donc la liste des applications linux et leur utilisation :

- *softuart.elf* :
  - usage *: /mnt/softuart.elf* (pas de paramètres)
  - description : cette routine permet au cœur linux de récupérer les chaînes de caractères produites pas le cœur µc lors de l'appel à la fonction *xil_printf().* Cette routine récupère les caractères un par un via les ports TX partagés. Elle affiche ensuite la chaîne dans la console linux lorsqu'elle rencontre un caractère de fin de ligne.

- Démarrez cette fonction en tâche de fond à chaque redémarrage de la carte car sans elle, vous ne verrez aucune trace provenant de µC. Cette routine est appelée dans le script *start.sh*

- *rwmem.elf* :
  - usage : */mnt/rwmem.elf ADDRESS_TO_READ/WRITE [VALUE_TO_WRITE]*
  - description : Cette application vous permet d'écrire ou de lire la mémoire partagée. Si vous voulez lire un emplacement, il suffit de ne pas renseigner le champ optionnel de paramètre (*VALUE_TO_WRITE*). Si vous voulez écrire, renseignez ce paramètre.
  - Cette tâche nous sert à déclencher les deux interruptions générées par les *IRQ_GEN* 0 et 1. Pour ce faire, il suffit d'écrire la valeur 1 aux adresses de base de ces deux modules matériels (resp. *0x78600000* et *0x78620000*). L'écriture sur *l'irq_gen_0* est assurée par la tâche *sendpacket.elf* dans le script *start.sh* tandis que l'écriture sur *l'irq_gen_1* est effectuée par le script *printstats.sh.*
  - Cette tâche sert aussi à démarrer le cœur µC lorsque l'on écrit la valeur 0x18000000 à l'adresse 0xFFFFFFF0. Vous pouvez donc démarrer le cœur µC à la main en faisant de même. Cette écriture est effectuée par *start.sh* avant de lancer *sendpacket.elf*.
- *sendpacket.elf* :
  - usage : */mnt/sendpacket.elf* (pas de paramètres)
  - description : Cette application génère de manière aléatoire les paquets qui sont ensuite envoyés vers le cœur µC. Ces paquets sont envoyés long par long (un paquet faisant 16 longs) via les ports RX partagés. Avant de commencer cette communication, la tâche *sendpacket.elf* écrit sur *l'irq_gen_0* pour déclencher l'interruption et démarrer la tâche de réception côté µC. Cette tâche est appelée par le script *start.sh* après l'appel à la tâche *softuart.elf.*

4- Aide sur le Débogage

L'énoncé vous signale qu'il est possible de lancer le débogage du code µC dans le Software Development Kit. Toutefois, beaucoup de paramètres peuvent influencer sur la procédure à suivre. Si vous voulez être sûr de réussir à lancer le débogage, suivez ces étapes :

1- Fermez l'invite XMD (si vous l'avez démarré depuis un shell) ou cliquez sur l'icône de redémarrage si vous avez démarré la console depuis le SDK.
2- Redémarrez la carte de développement. Il est possible que vous ayez à débrancher l'un des ports micro USB pour entièrement couper l'alimentation de la carte.
3- Relancez la console XMD avec la commande *connect arm hw –debugdevice cpunr 2*
4- Dans putty, lancez softuart.elf en tâche de fond (*/mnt/softuart.elf &*)

5- Démarrez le debug après vous être assurés que le débug précédent ait été éteint
6- Lancez *sendpacket.elf* une fois que le debug est démarré