Laboratoire no 2 – Partie 1a

Création d'une plate-forme pour Zedboard avec Xilinx Vivado 2018.3

Dans ce laboratoire, vous allez réaliser le bloc du bas de la figure 3 (voir énoncé de laboratoire). Ce bloc est illustré à la figure 1 suivante.

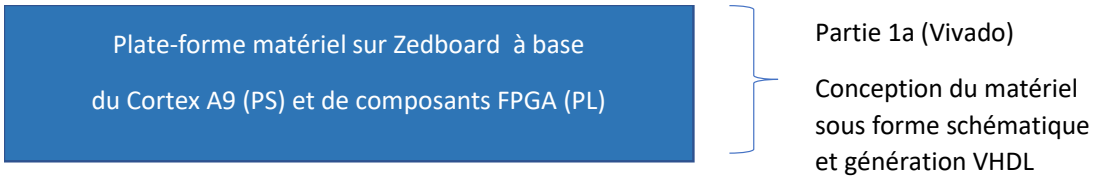


Figure 1 À concevoir dans la partie 1a

Plus précisément, vous allez créer avec l'aide de Vivado la plate-forme matérielle de la figure 2 qui vous servira dans la suite de ce laboratoire. Plus précisément, vous allez instanciés, puis connecter entre eux, un certains nombres de composants (IPs) et ainsi construire le schéma de la figure 3 (voir Annexe). À partir de ce schéma, vous allez générer (automatiquement) le VHDL puis le bitstream.

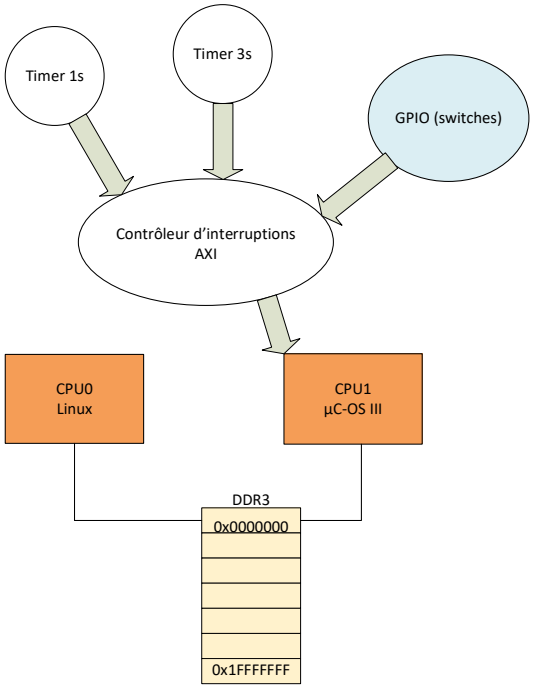
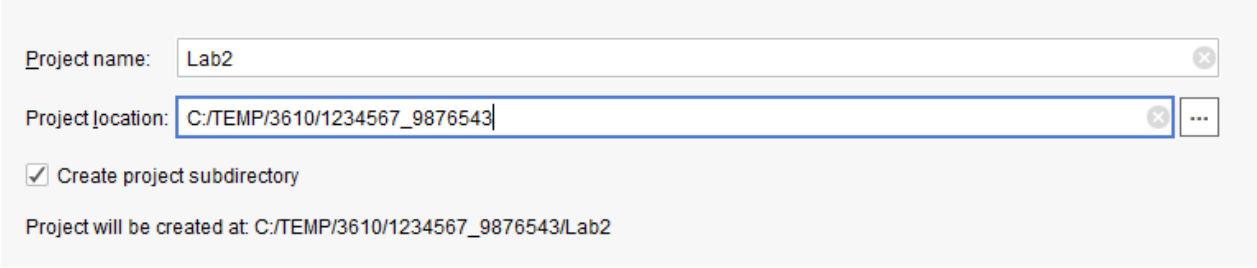


Figure 2. Plate-forme servant au laboratoire no 2

1. Ouvrez **Vivado 2018.3**

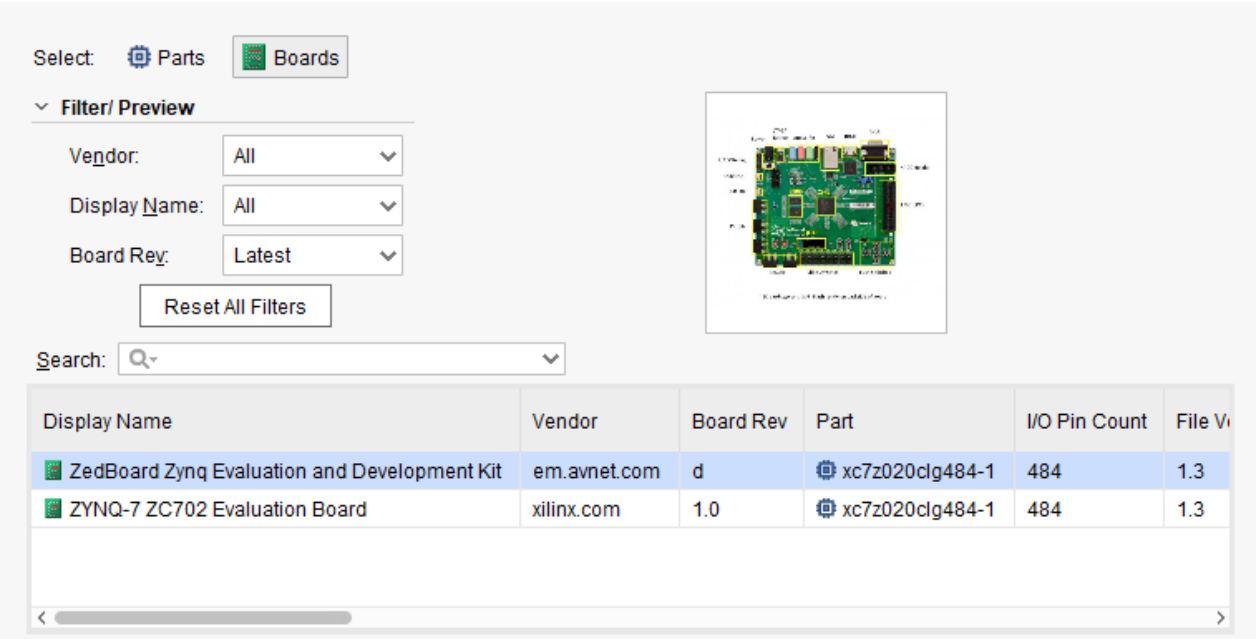
2. Créez un nouveau projet, nommé Lab2, dans **C:/Temp/3610/matricule1_matricule2**[1]



3. Créez un projet RTL, en cochant *Do not specify sources at this time*

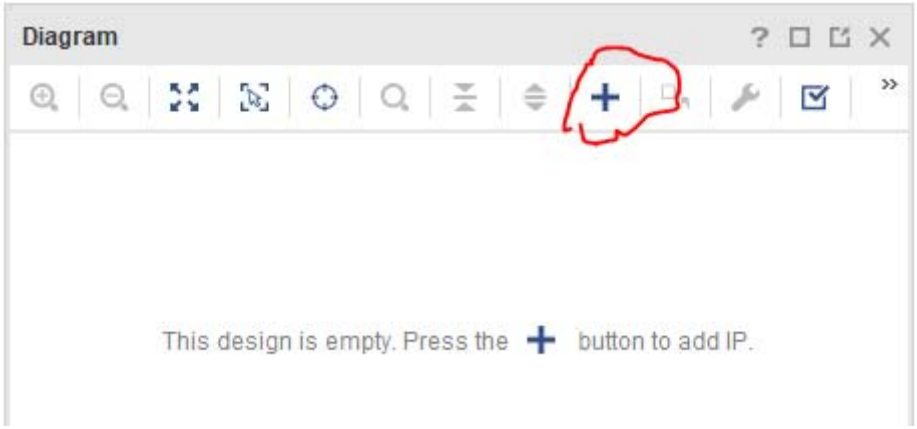
[1] Il est primordial de ne pas créer ce projet sur le réseau de Poly, aka sur votre bureau, dans *Mes documents*, ou n'importe où ailleurs dans X:\, afin de sauver votre temps et celui de votre chargé(e) de laboratoire, qui devra se déplacer lorsque vous aurez des problèmes pour vous dire exactement ce qui est écrit ici.

4. Ciblez le *Zedboard Zynq Evaluation and Development Kit* dans *Boards*



5. Dans le menu à gauche, cliquez sur *Create Block Design*, puis OK.

6. Le bouton + dans la nouvelle fenêtre *Diagram* vous permet maintenant d'ajouter ou d'activer des blocs au Zynq.

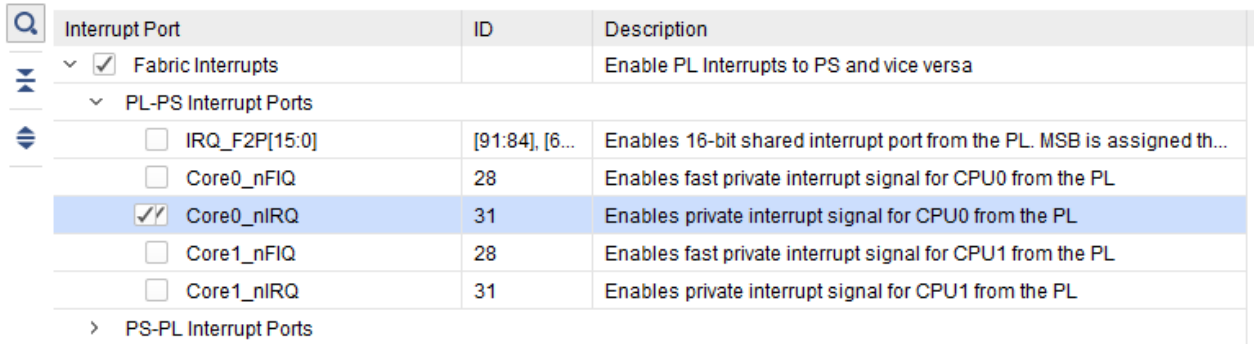


7. Ajoutez une instance de *Zynq7 Processing System*. Ceci permet d'activer le processeur ARM à deux cœurs[2] présents sur le Zedboard.

8. Cliquez sur *Run block automation* pour automatiser la connexion à la mémoire externe (les paramètres par défaut sont bons).

[2] Notez que bien que le processeur ait 2 cœurs, un seul (le coeur #0) sera utilisé plus loin pour faire fonctionner µC.

9. Double-cliquez sur l'instance créée, allez dans *Interrupts*, puis cochez *Fabric Interrupts* et *Core0_nIRQ* pour permettre aux blocs du FPGA de générer des interruptions sur le processeur.



10. Créez 2 instances de *Fixed Interval Timer* (FIT), en cliquant sur *Run block automation* après chaque ajout.

11. Les FIT sont des timers très simples, qui ne font que lever un signal sur leur sortie *Interrupt* après un compte préprogrammé de cycles, en boucle. Modifiez leur configuration pour que le timer #0 lève son interruption à toutes les secondes et que le timer #1 lève son interruption aux 3 secondes (sachant qu'à l'étape précédente, vous avez connecté le bloc à une horloge de 100 MHz).

12. Ajoutez un bloc *AXI GPIO*. Modifiez sa configuration pour que l'interface *GPIO* soit connectée sur les switches du Zedboard (*sws 8bits*) et l'interface *GPIO2* soit connectée aux LEDs du Zedboard. Cochez aussi l'option *Enable interrupt* puis automatisez la génération des connexions.

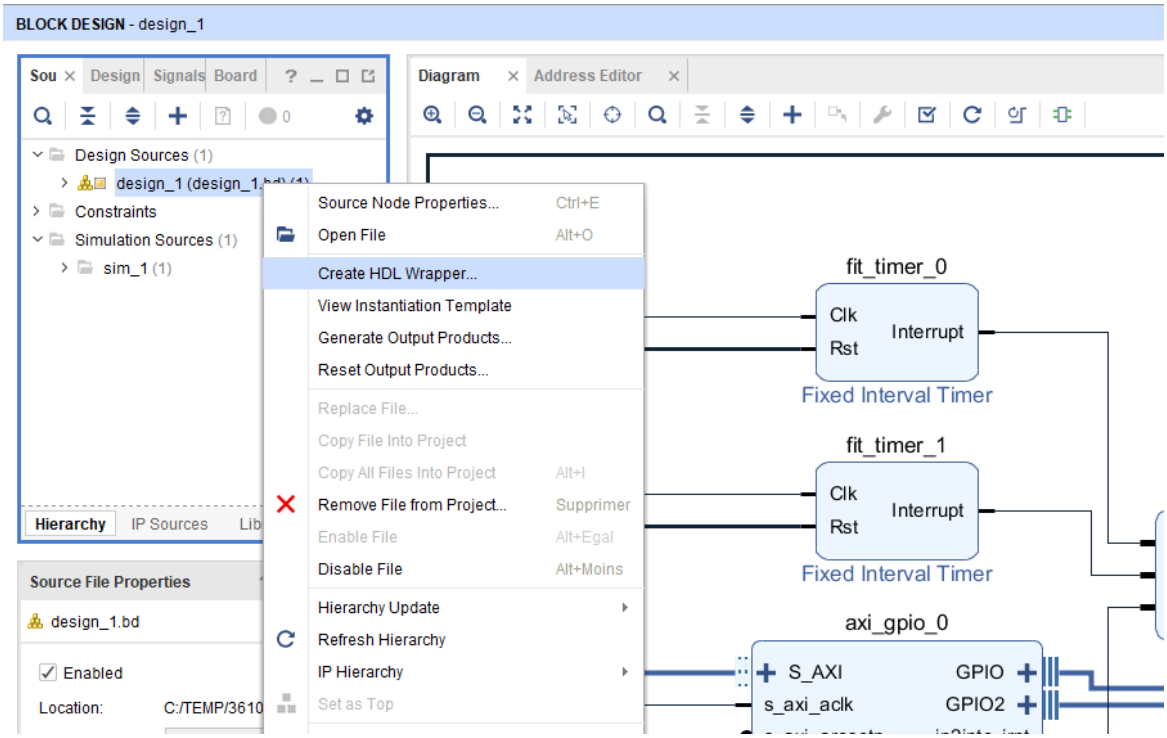
13. Ajoutez un bloc *AXI Interrupt Controller*. Tel qu'expliqué dans l'énoncé du laboratoire, ce contrôleur permettra de faire le multiplexage des 3 signaux d'interruption des blocs préalablement générés dans la seule interruption privée au cœur utilisé du processeur ARM. Connectez sa sortie *irq* (disponible en cliquant sur le **+** du signal *interrupt*) à l'entrée *Core0_nIRQ* du Zynq. Puis, automatisez la génération des connexions.

14. Ajoutez une instance de *Concat*[3] et modifiez là pour qu'elle ait 3 entrées. Connectez les sorties d'interruption des blocs FIT et GPIO aux entrées de cette instance, et sa sortie au port *intr* du contrôleur d'interruption AXI.

[3] Notez que *Concat* (sorte de MUX) n'est qu'un moyen visuel de connecter plusieurs sorties indépendantes à un port d'entrée à plusieurs bits.

15. Votre diagramme de blocs devrait maintenant ressembler à celui présenté en Annexe.

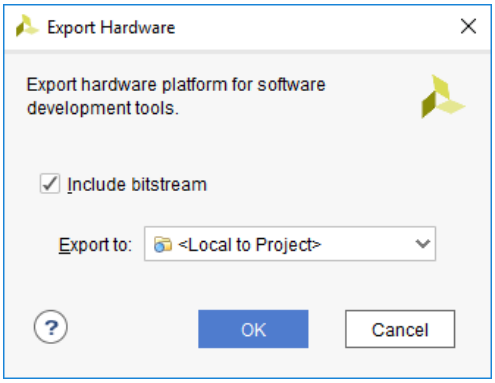
16. Dans le panneau *Source*, faites un clic droit sur votre design et cliquez sur *Create HDL Wrapper*, puis OK.



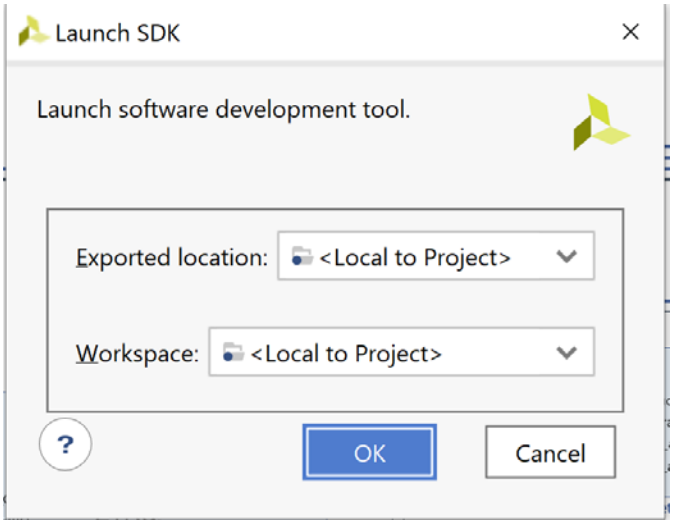
17. Sauvegardez, puis cliquez sur *Generate Bitstream* dans le panneau de gauche. Cliquez sur Yes lorsque Vivado vous informe qu'il doit préalablement exécuter la synthèse du système. Démarrez autant de *jobs* en parallèle que possible. Le tout prendra quelques minutes.

18. Fermez la fenêtre apparaissant à la fin de la génération correcte du bitstream.

19. Cliquez sur File->Export->Export Hardware et assurez-vous de cocher Include bitstream.



20. Si tout s'est bien passé, vous devriez maintenant pouvoir passer à la partie SDK (création du BSP et de l'application uC). Pour cela ouvrir Xilinx SDK à partir de Vivado afin que votre workspace soit bien initialisé et que les fichiers nécessaires soient importés : Faites donc File -> Launch SDK puis cliquez OK.



21. Vivado peut maintenant être fermé.

Annexe

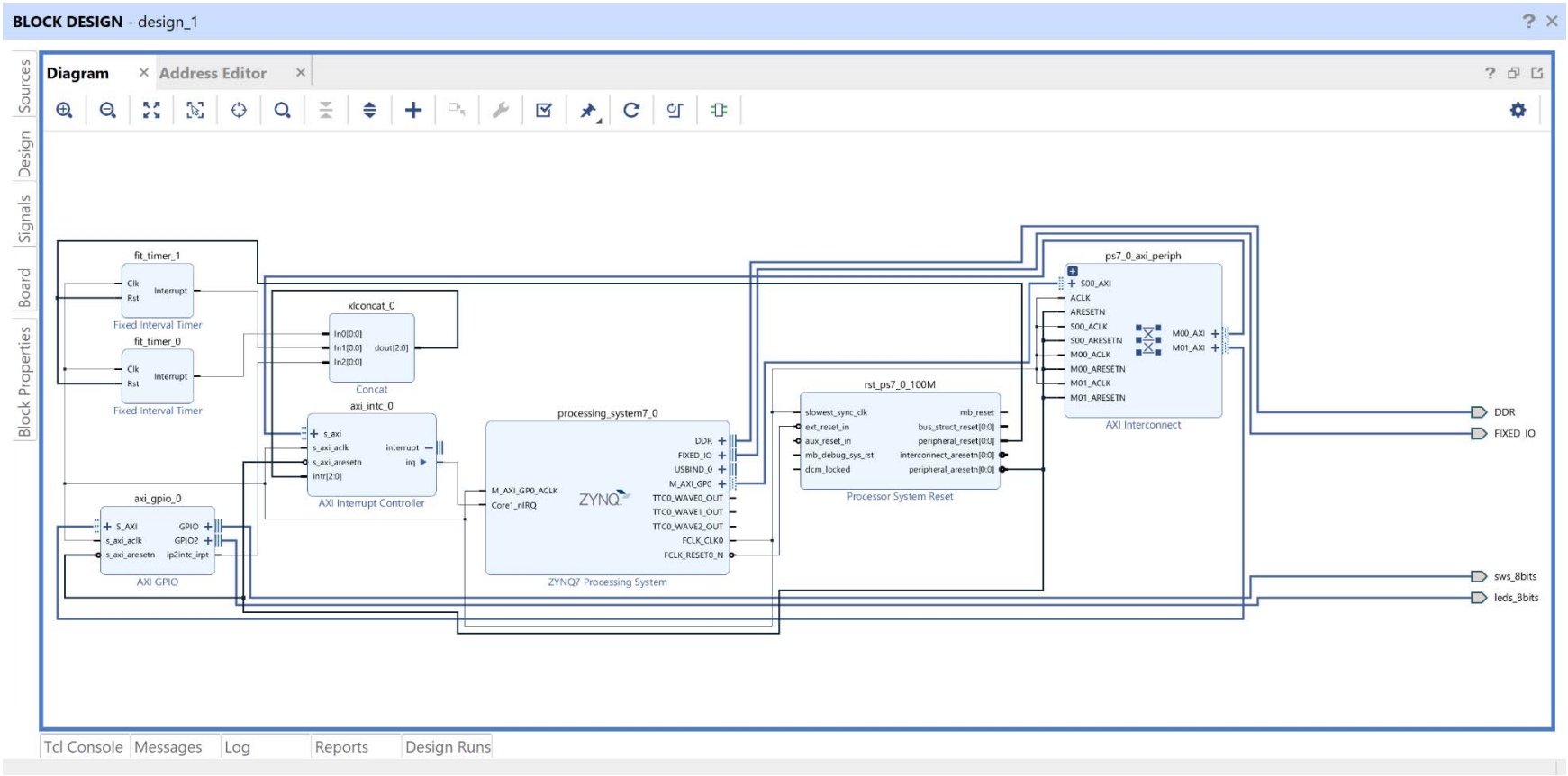


Figure 3. Schéma résultant sous Vivado représentant la plate-forme de la figure 2