Implantation d'un double cœur

1) Problématique

Dans cet exemple on souhaite utiliser les deux cœurs de processeur de la partie PS du ZYNQ. Cette utilisation impose de nouvelles contraintes liées au démarrage du système PS, l'intercommunication entre les deux processeurs et l'accès à des ressources partagées. XILINX a prévu un mécanisme de multitraitement asymétrique afin de résoudre ces nouvelles contraintes

2) Mécanisme d'initialisation et de synchronisation des doubles cœurs.

Le ZYNQ-7000 possède deux processeurs CORTEX-A9 qui partage une mémoire commune et des périphériques. Le multitraitement asymétrique (Asymmetric multiprocessing (AMP)) est un mécanisme qui permet à la fois aux processeurs de commencer leurs propres OS ou des applications bare-metal avec la possibilité de coupler certaines applications via des ressources partagées.
Chacun des deux cœurs de processeur CPU0 et CPU1 sont configurés pour commencer ces propres applications cloisonnées. Durant cette séance les applications cloisonnées commencent sur le CPU0 qui est le maitre du système et est responsable de :

- ✓ L'initialisation du système
- ✓ Le contrôle et le démarrage du CPU1
- ✓ La communication avec le CPU1
- ✓ Le partage de l'UART avec le CPU1

L'application cloisonnée qui commencent sur le CPU1 est responsable de :

- ✓ La communication avec le CPU0
- ✓ L'interruption des services à partir d'un IPCORE présent dans la logique programmable PL générateur d'interruption
- ✓ Le partage de l'UART avec le CPU0

Le SOC ZYNQ et son système de traitement (PS) inclut des ressources qui sont privées à chaque CPU et des ressources partagées à la fois par les deux CPU

Les ressources privées peuvent être :

- ➢ La mémoire cache L1
- ➢ Les interruptions des périphériques privés
- ➢ L'unité de management de la mémoire
- ➢ Les timers privés

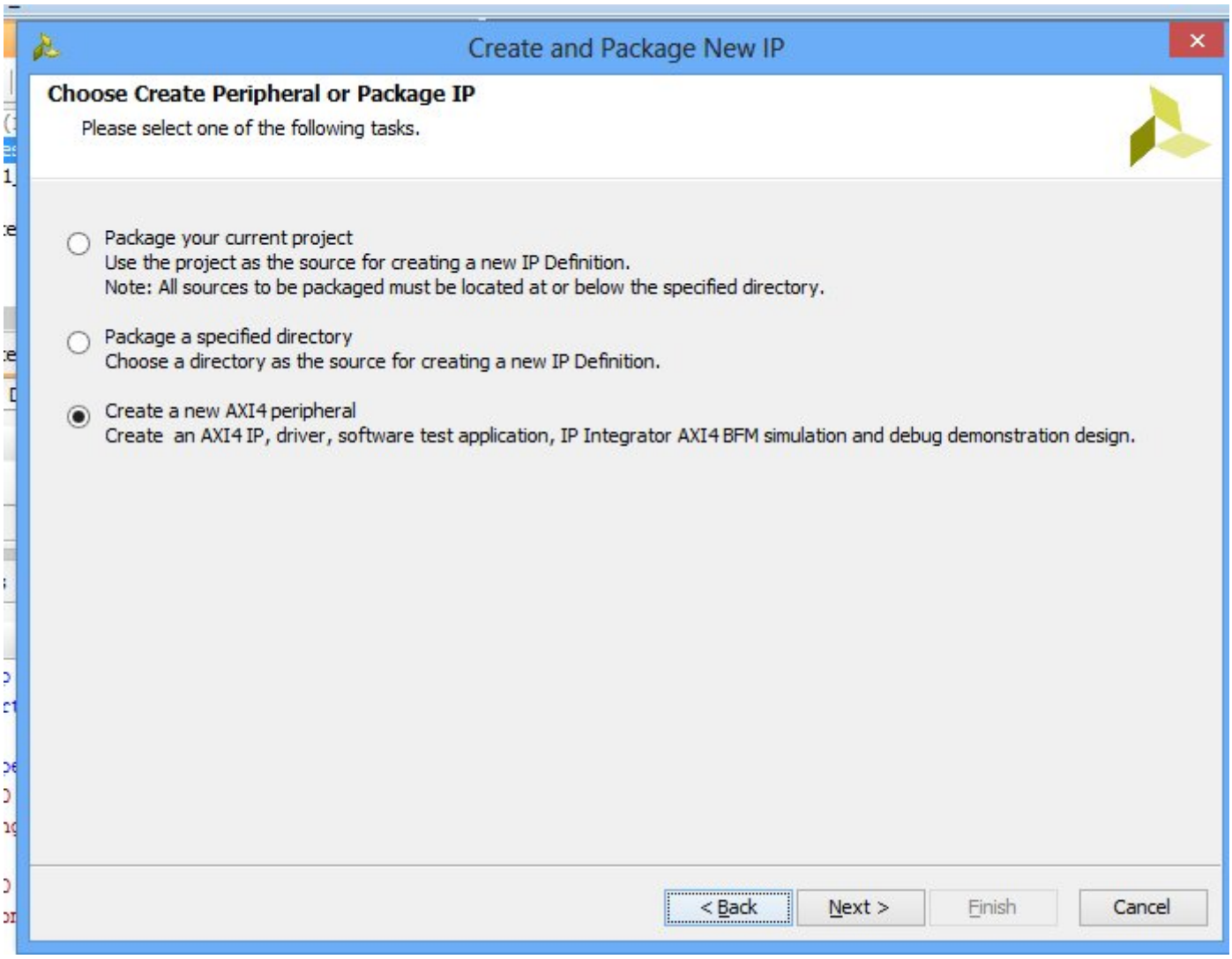
Les ressources partagées peuvent être :

- ➢ Le distributeur de contrôle d'interruption
- ➢ La mémoire DDR
- ➢ La mémoire sur circuit OCM On-chip memory (OCM)
- ➢ Les timers globaux
- ➢ La SCU et la mémoire cache de niveau 2
- ➢ L'UART0

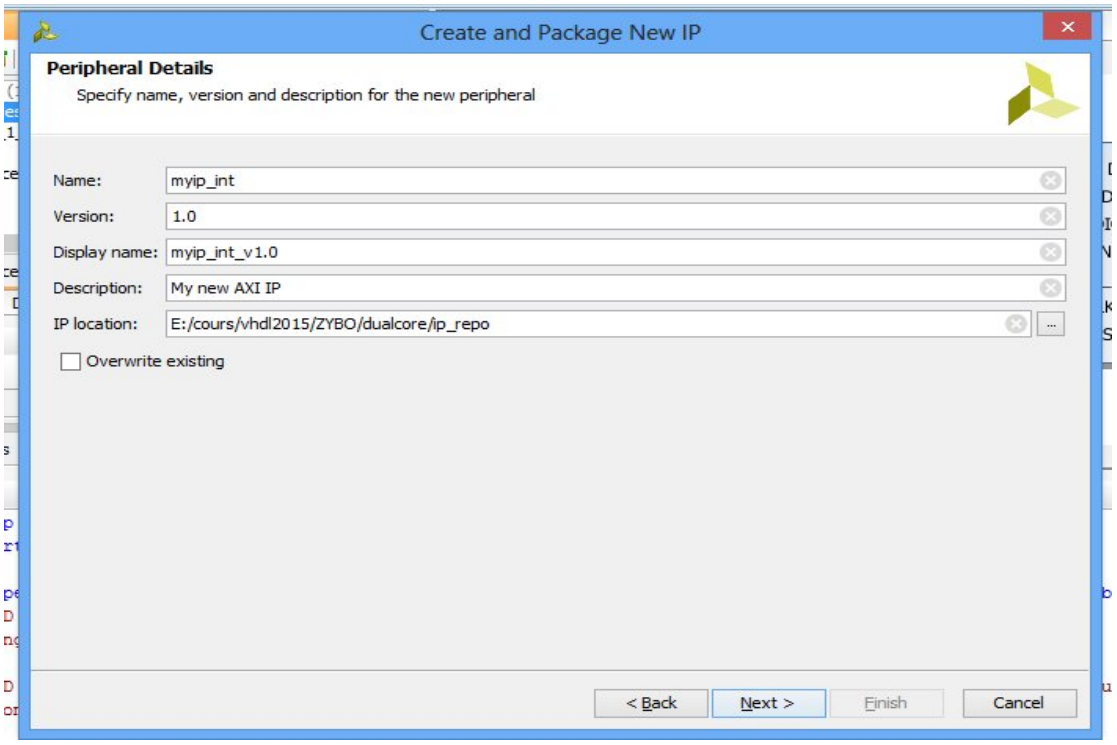
Dans cet exemple le CPU0 est traité comme le maitre et contrôle les ressources partagées. Si le CPU1 veut réquisitionner une ressource partagée, il fera une requête auprès du CPU0. Il laisse le CPU0 contrôler la ressource. Les applications cloisonnées qui commencent sur le CPU1 ont été modifié afin de limiter l'accès aux ressources partagées.
L'OCM est utilisée par les deux processeurs comme une mémoire partagée. Comparé à la mémoire DDR, l' OCM fournit de bien meilleures performances et des faible latences lors des accès.

3) L' Intégration matérielle

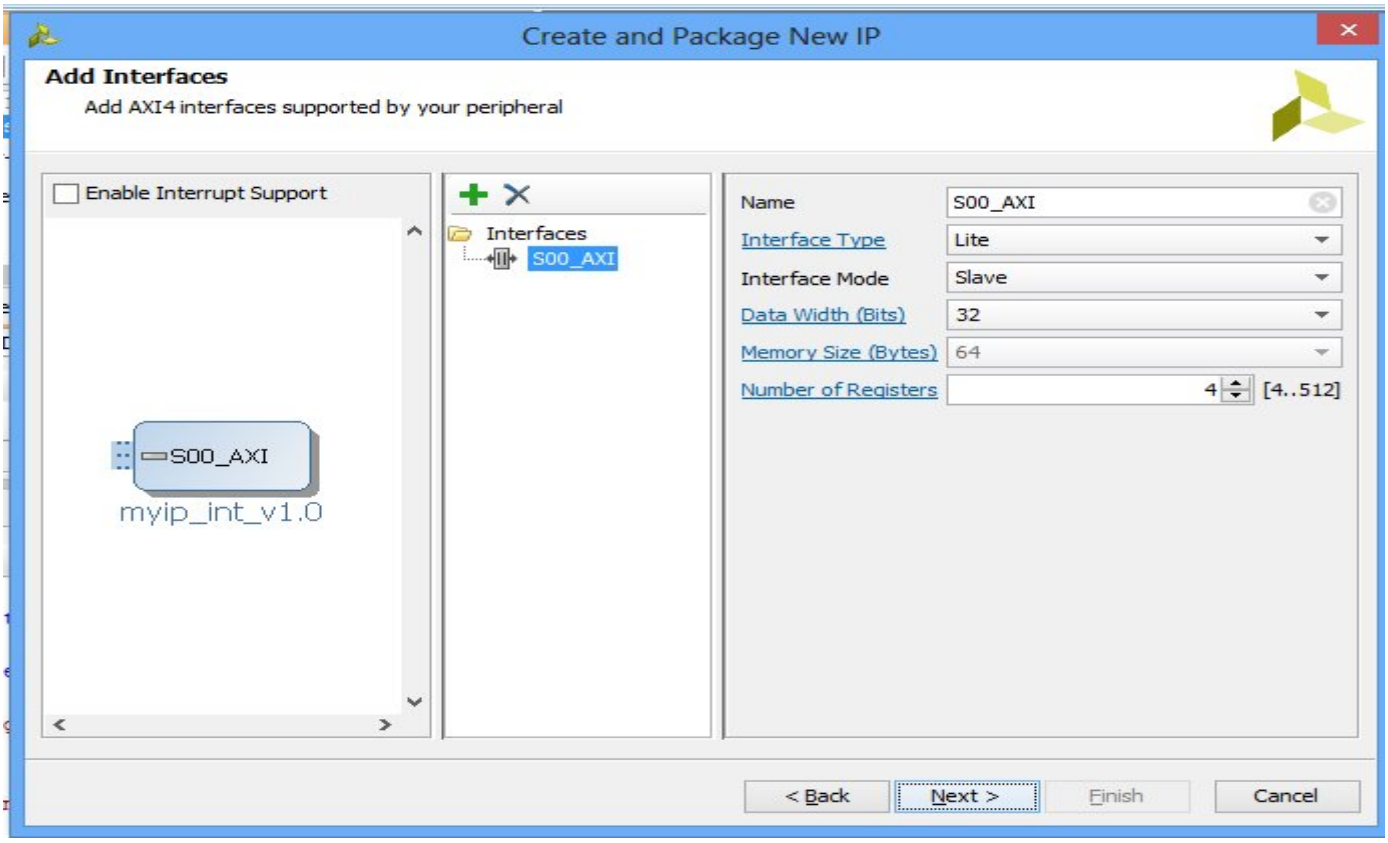
Créez Un projet avec le fichier de contrainte de la carte ZYBO. L'étape suivante va nous permettre d'implanter notre propre périphérique.

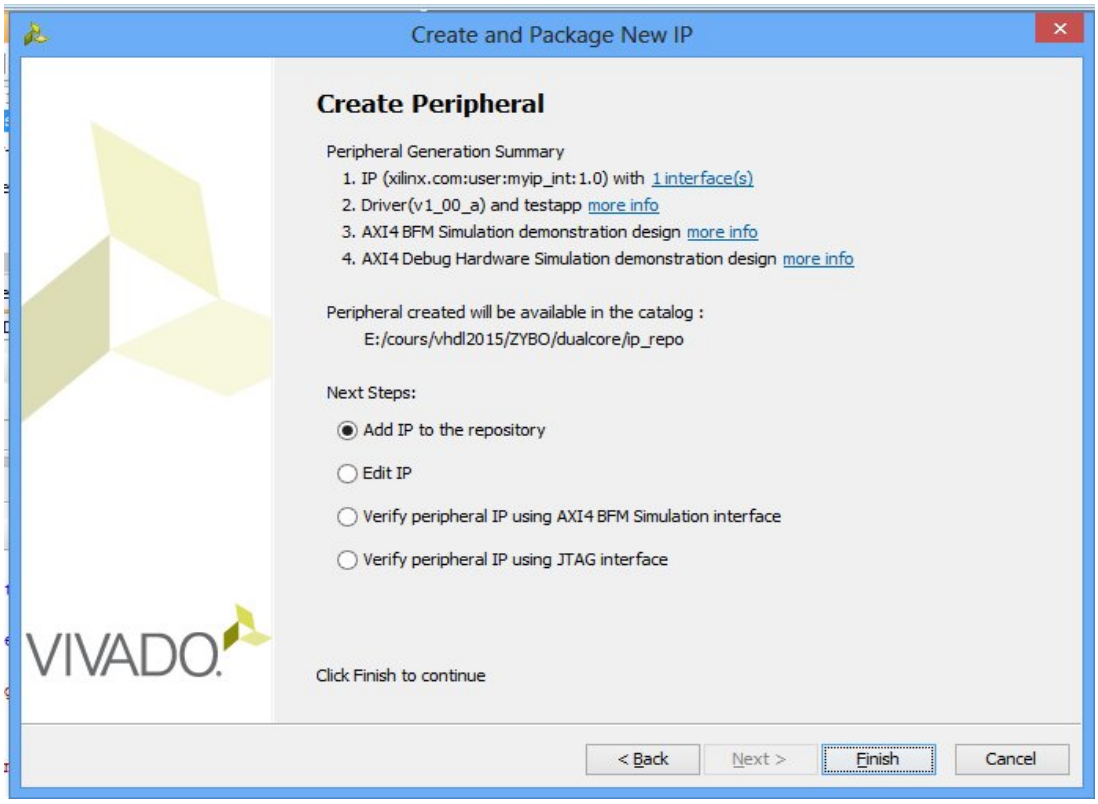


Notre périphérique sera nommé myip_int

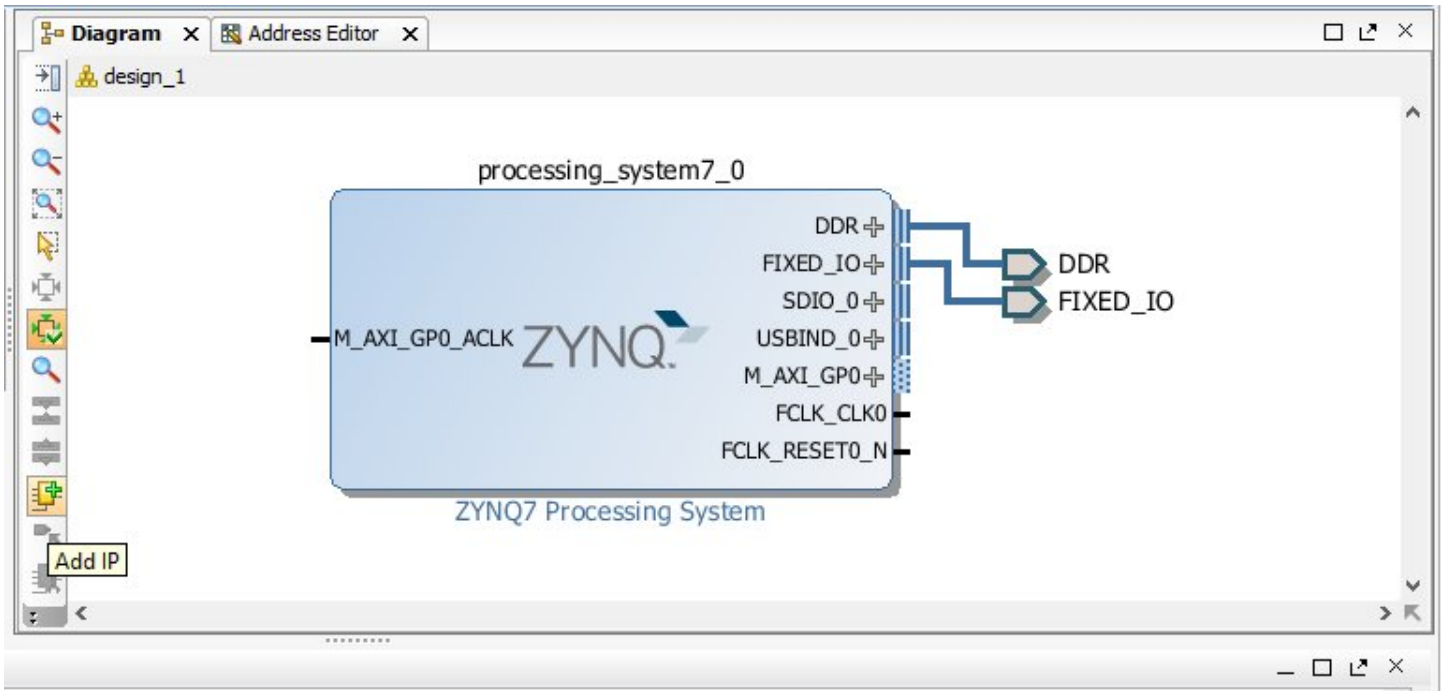


On choisit un bus du type AXI LITE

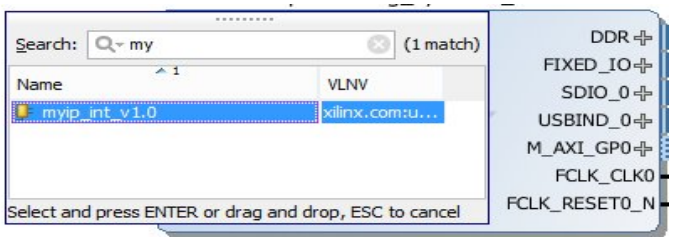


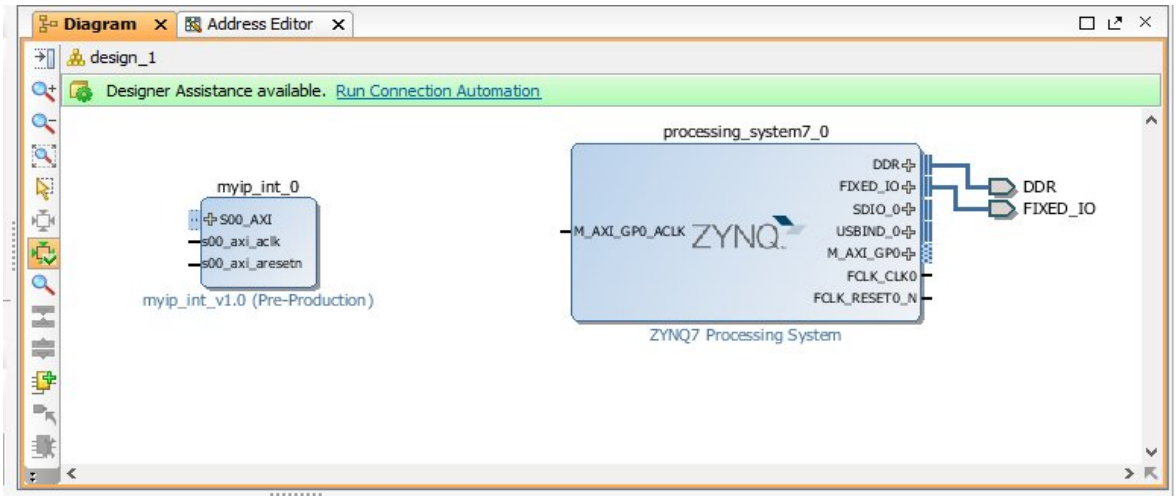


Créez un bloc design, Intégrez un PS, puis automatisez les connexions du PS

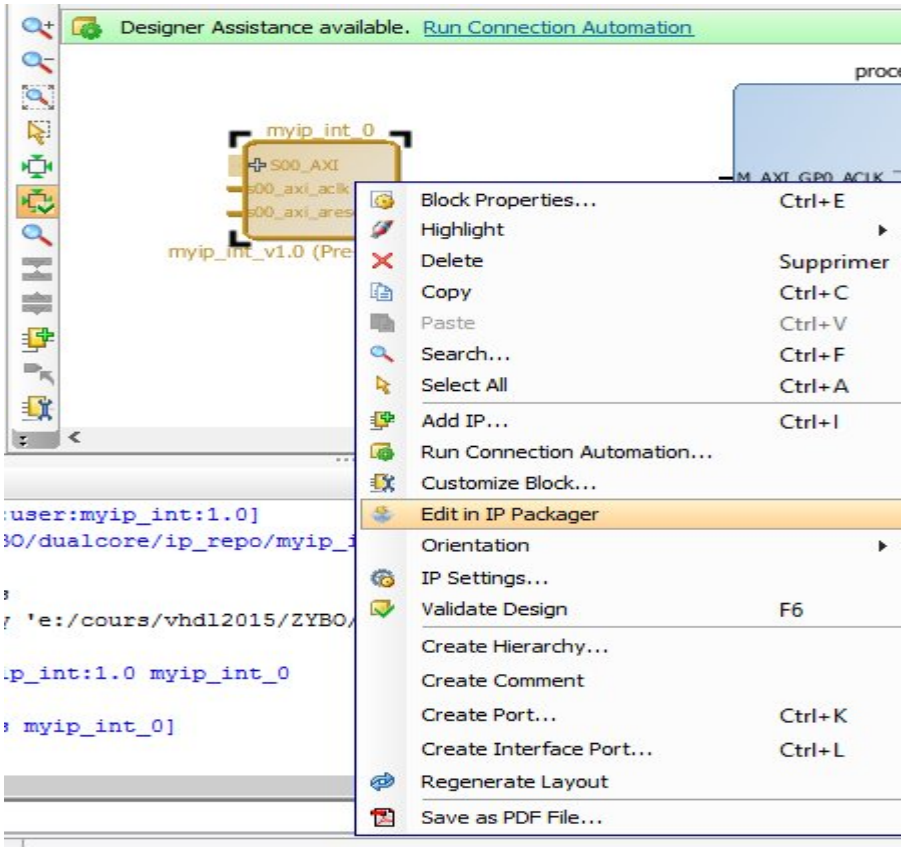


Intégrez le composant myip_int

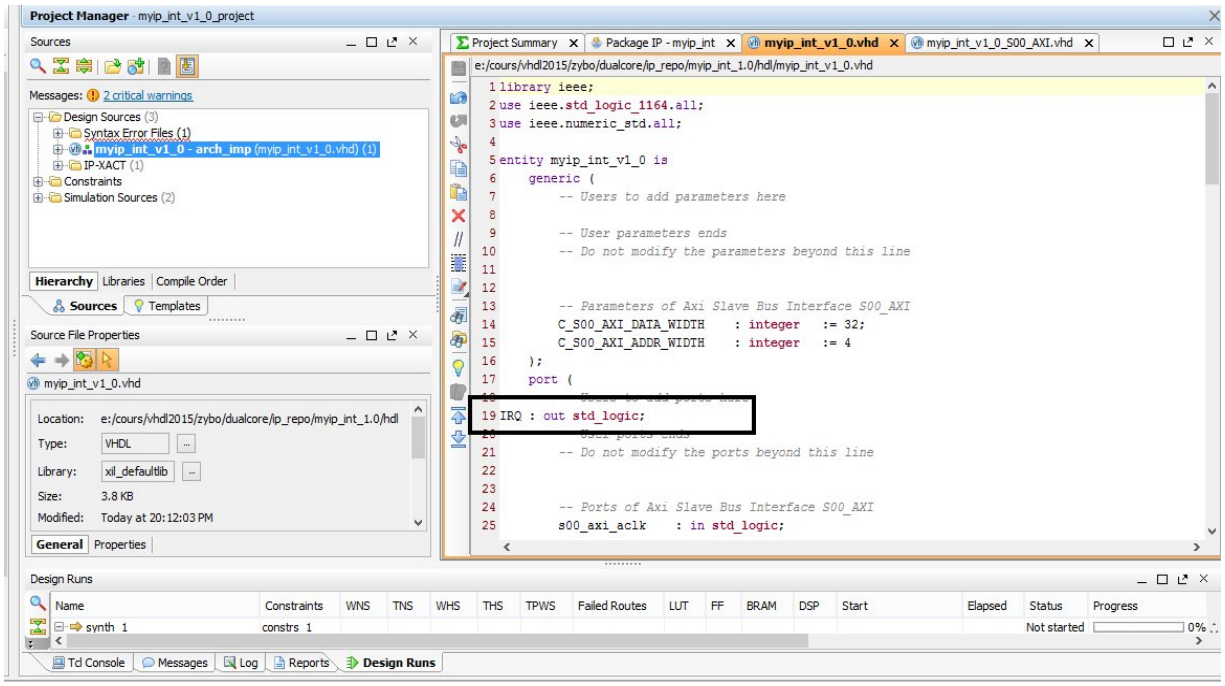


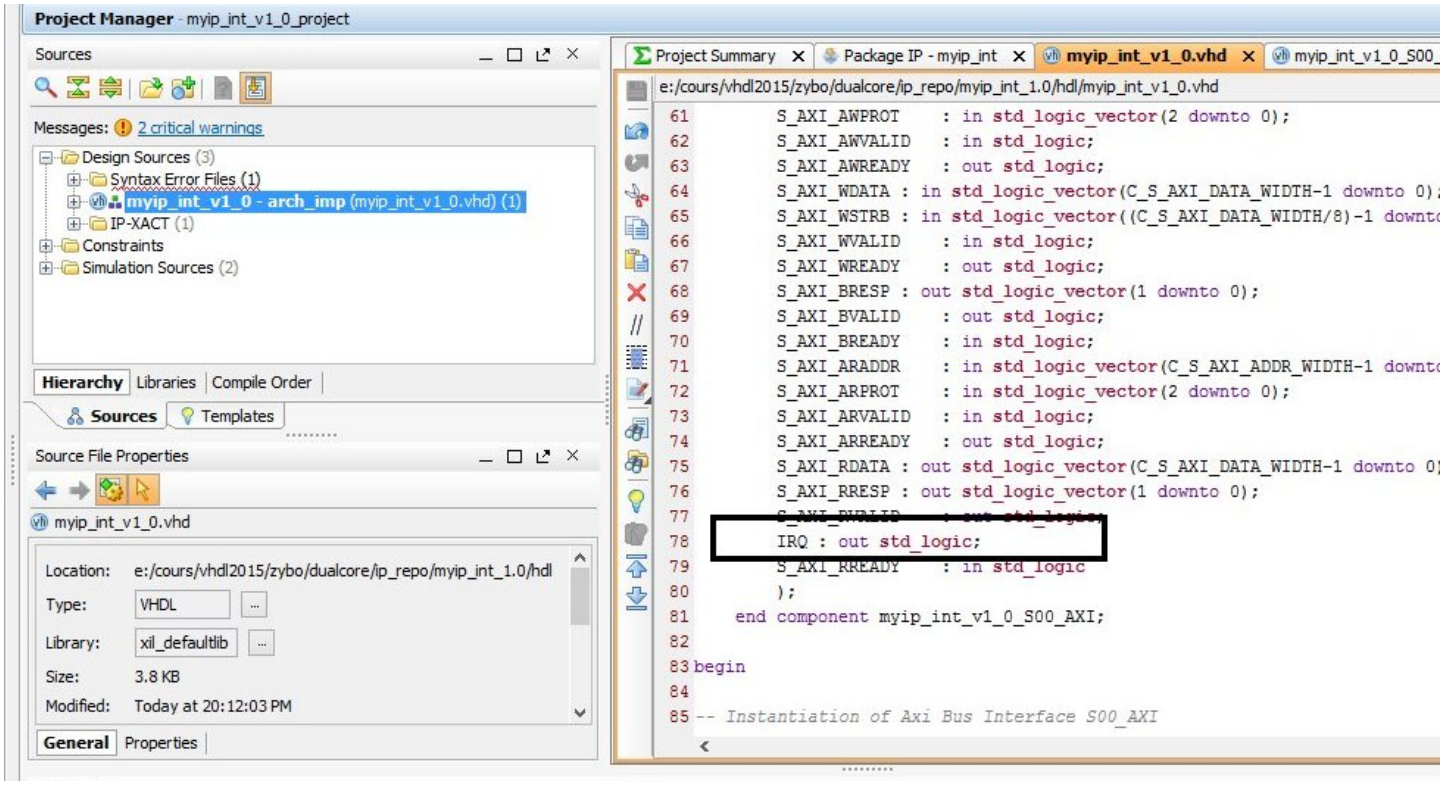


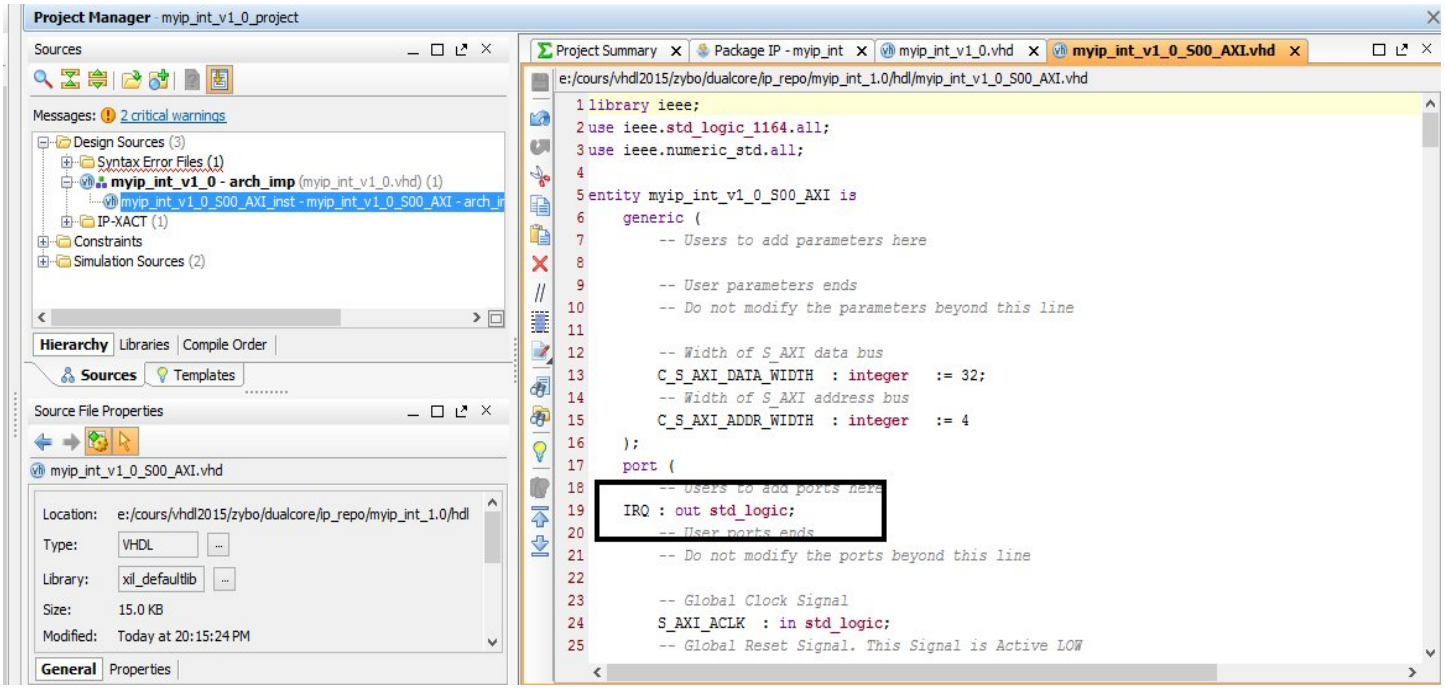
On va lancer IP packager pour compléter le fichier VHDL de notre périphérique



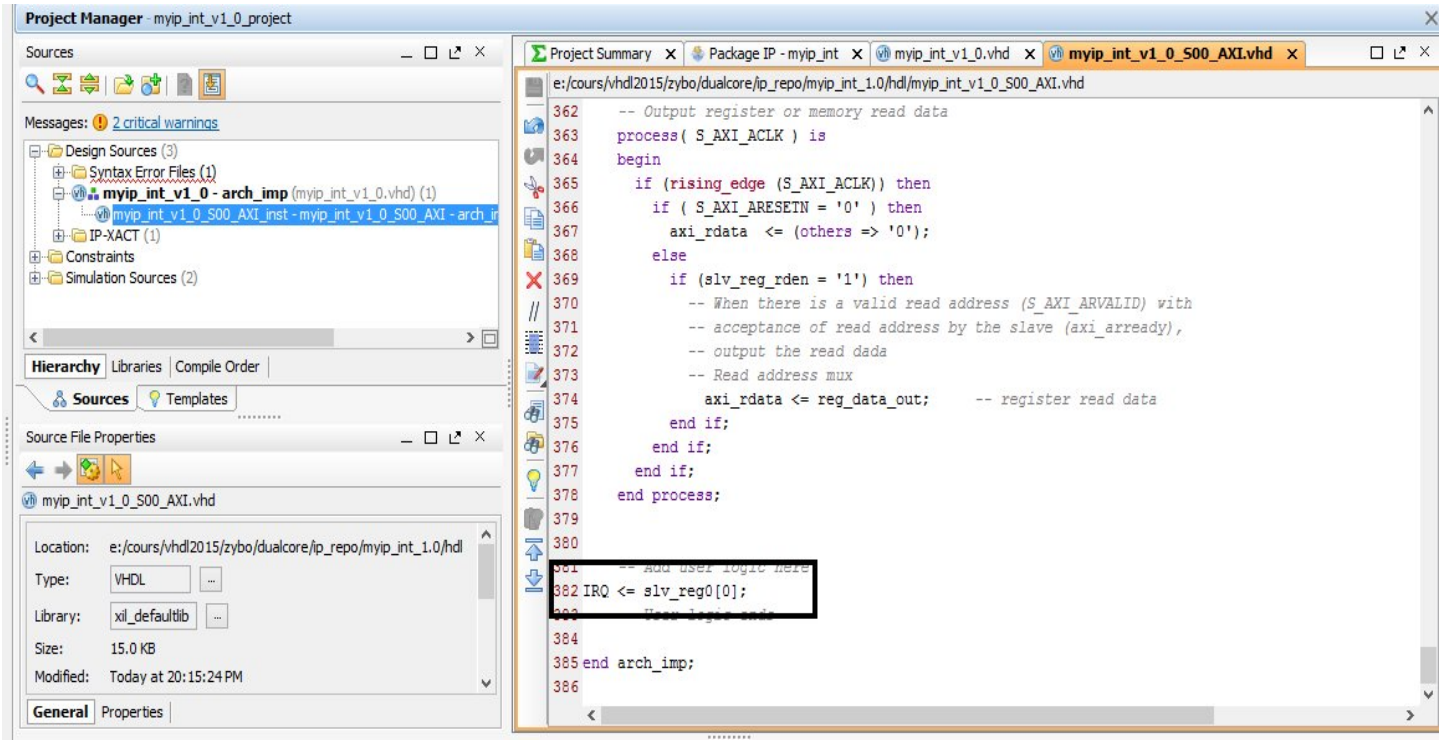
Déclarez une sortie du type IRQ



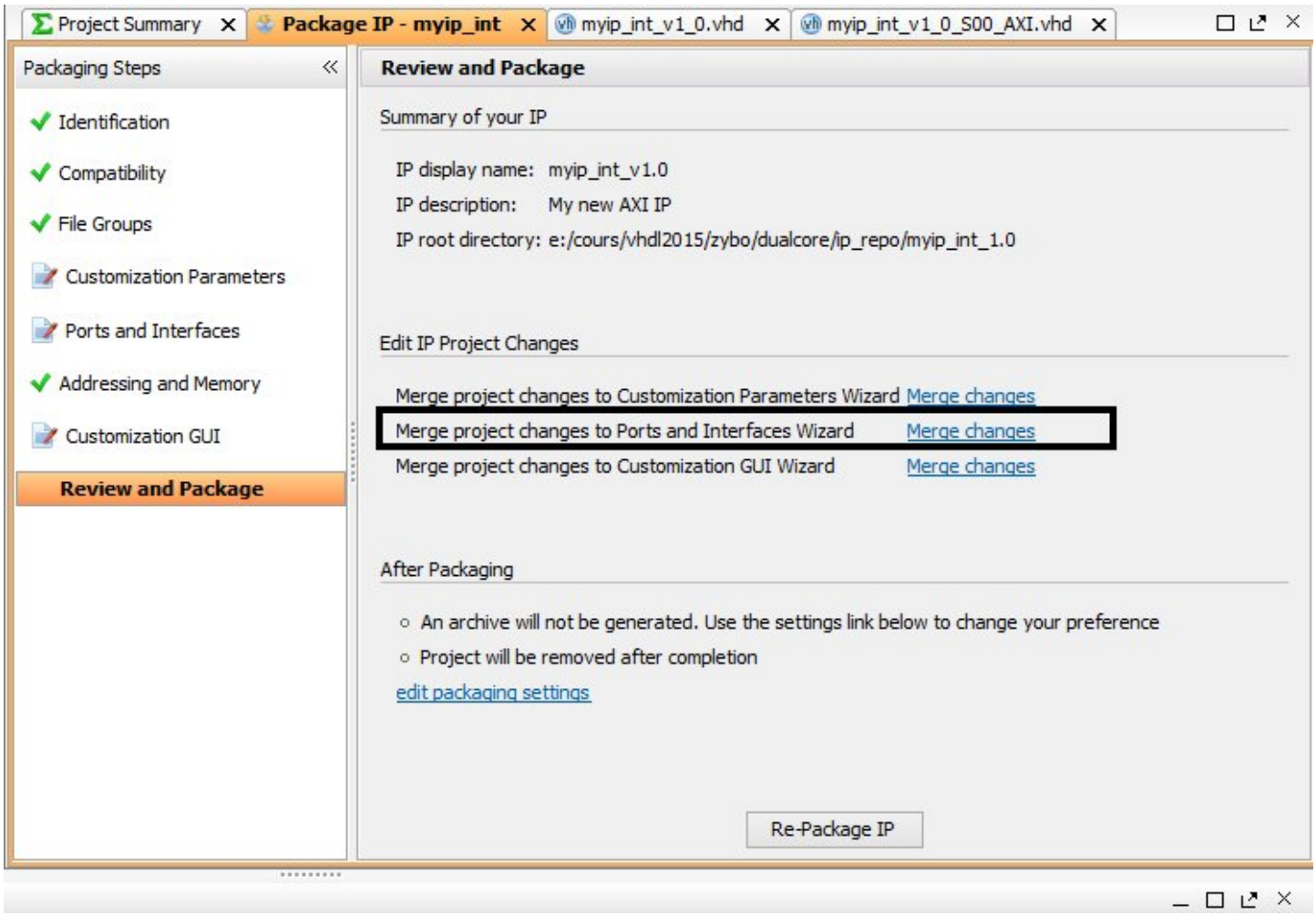




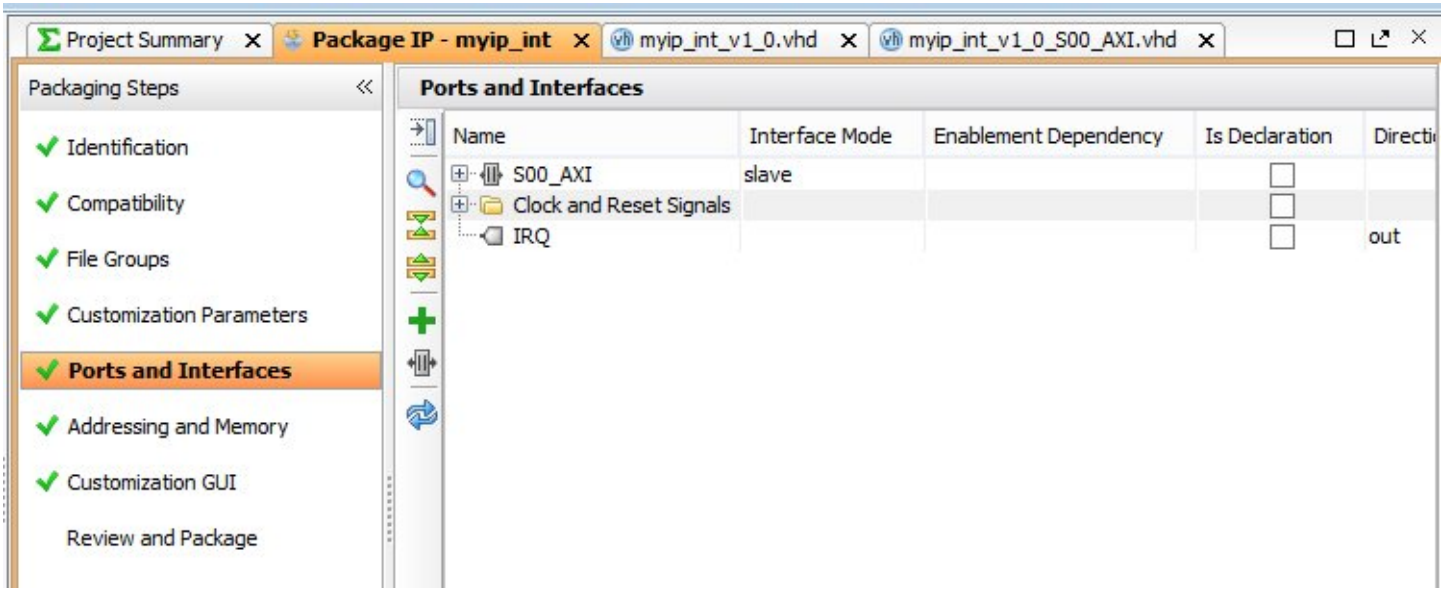
Définir l'instruction suivante la sortie IRQ sera l'image du bit slv_reg0(0)



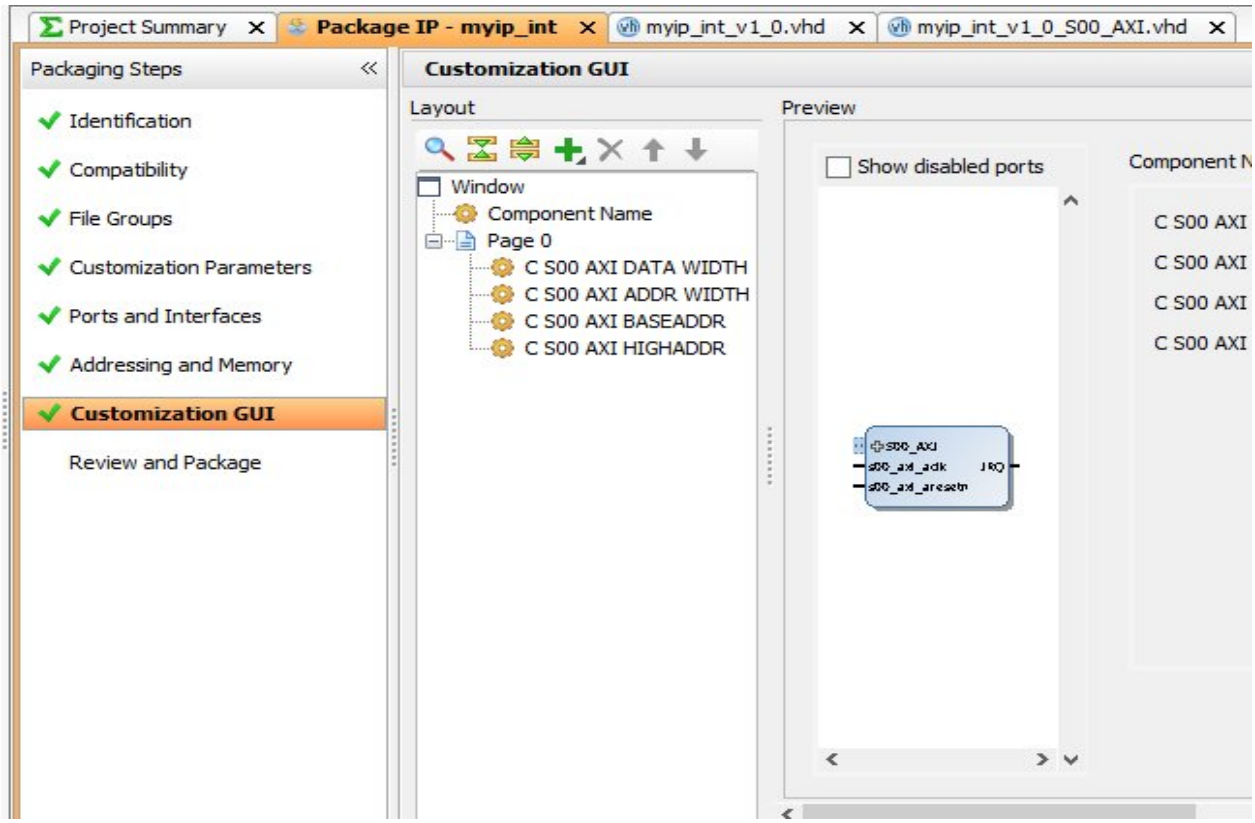
Prendre en compte les modifications en cliquant sur Merge project



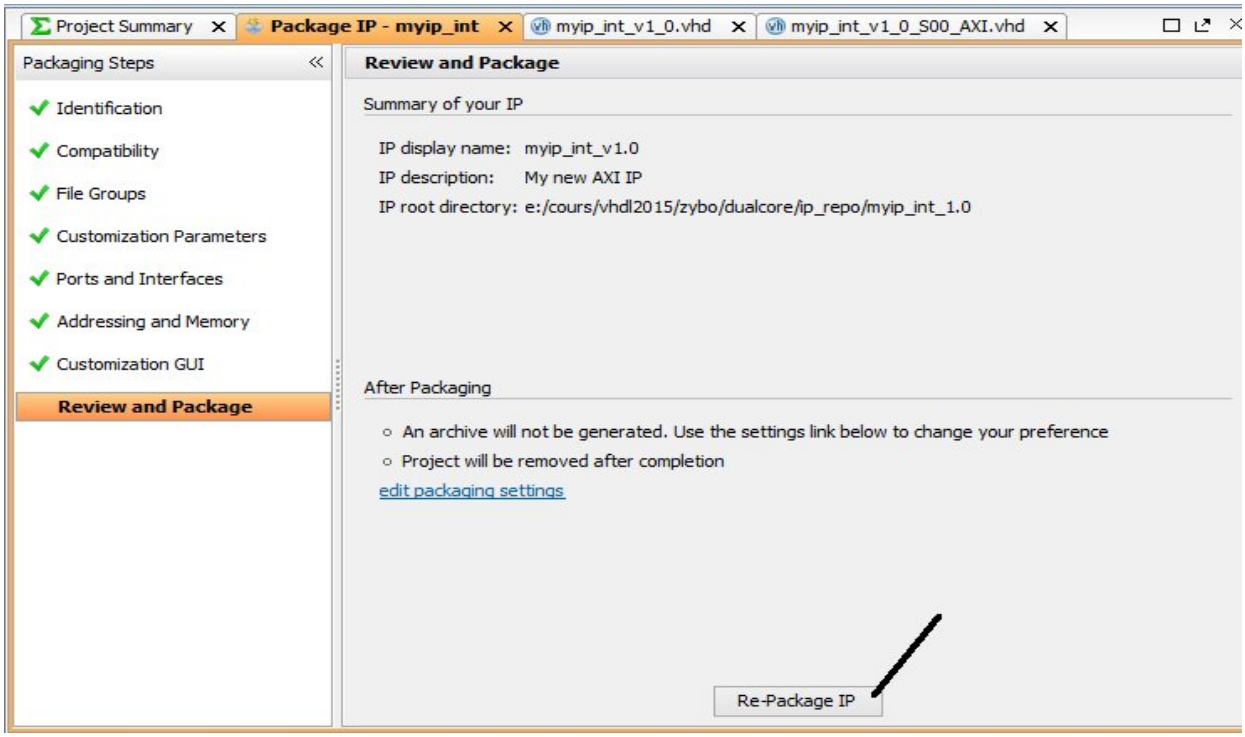
La sortie IRQ apparait



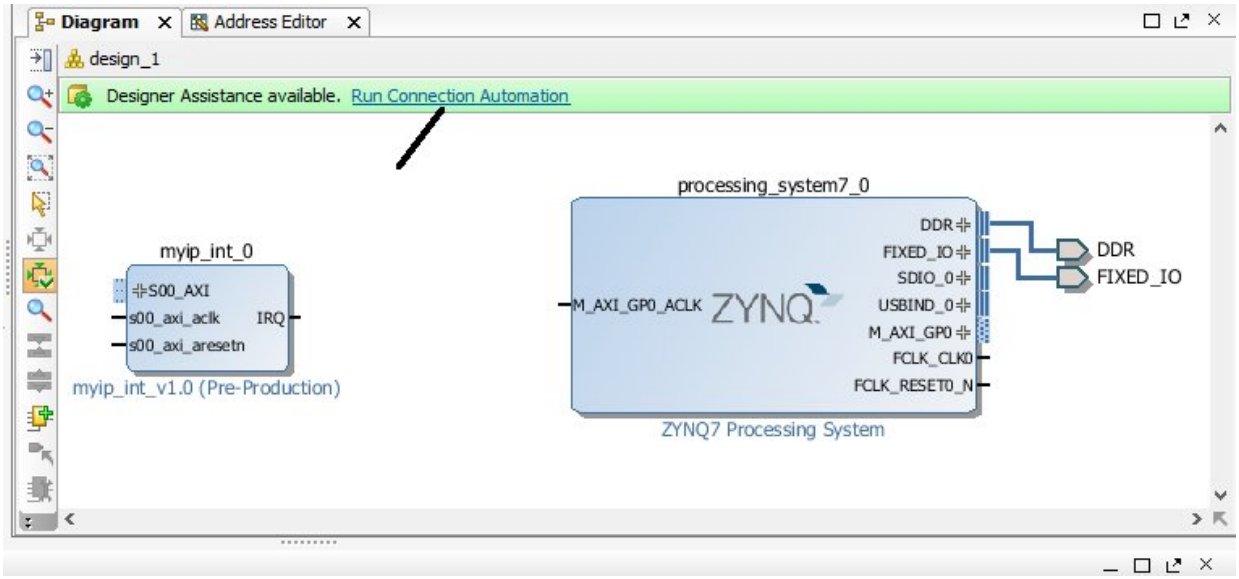
Le symbole avec son bus AXI lite et la sortie IRQ

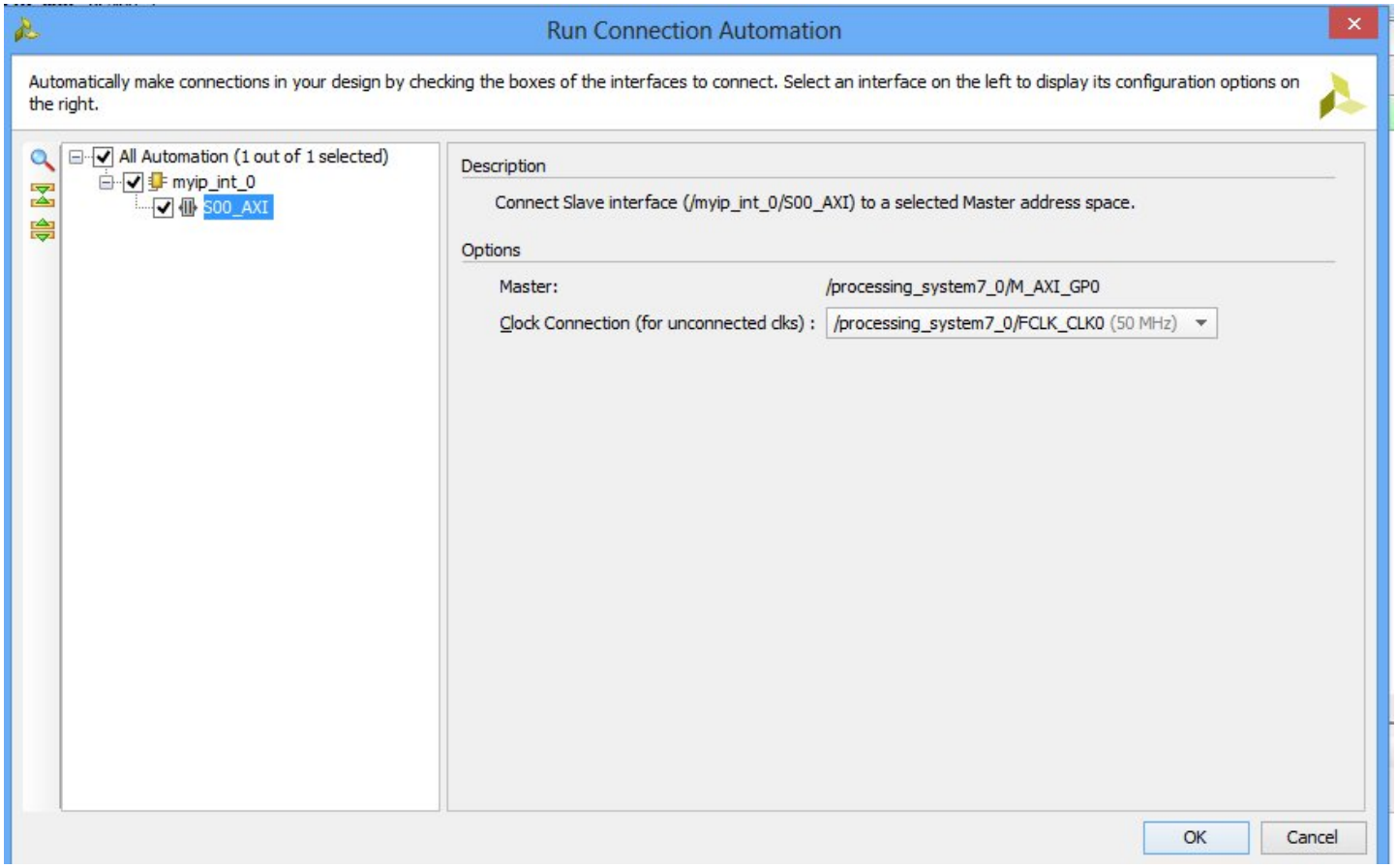


Relancez la package IP

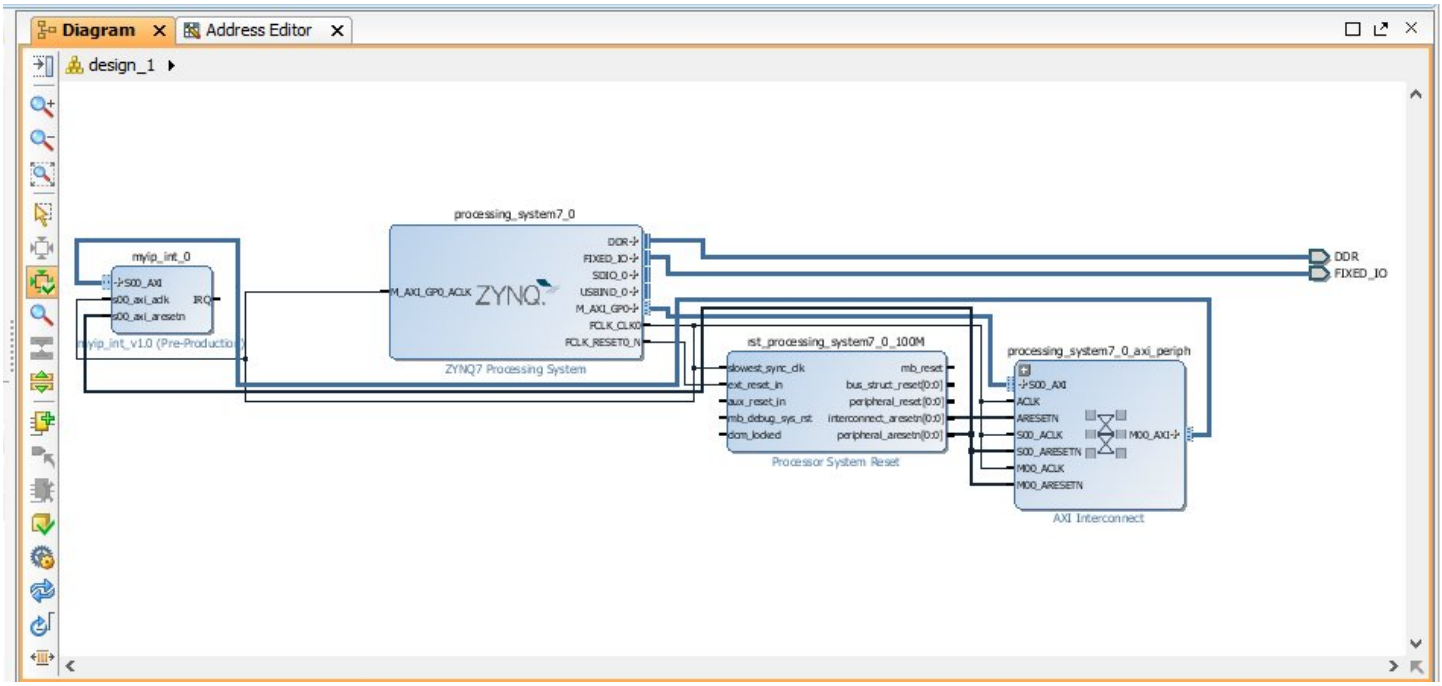


Continuez l'automatisation de la connexion

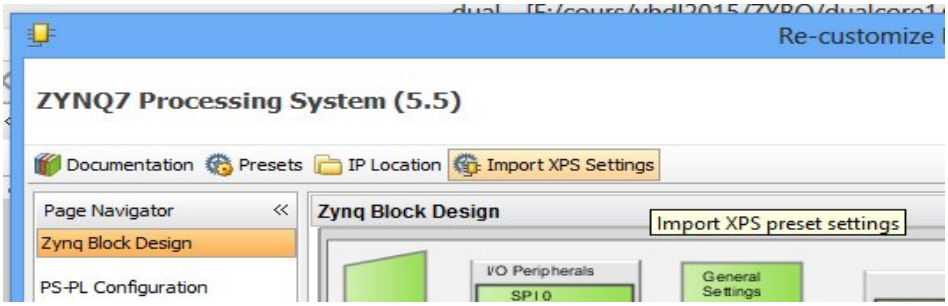


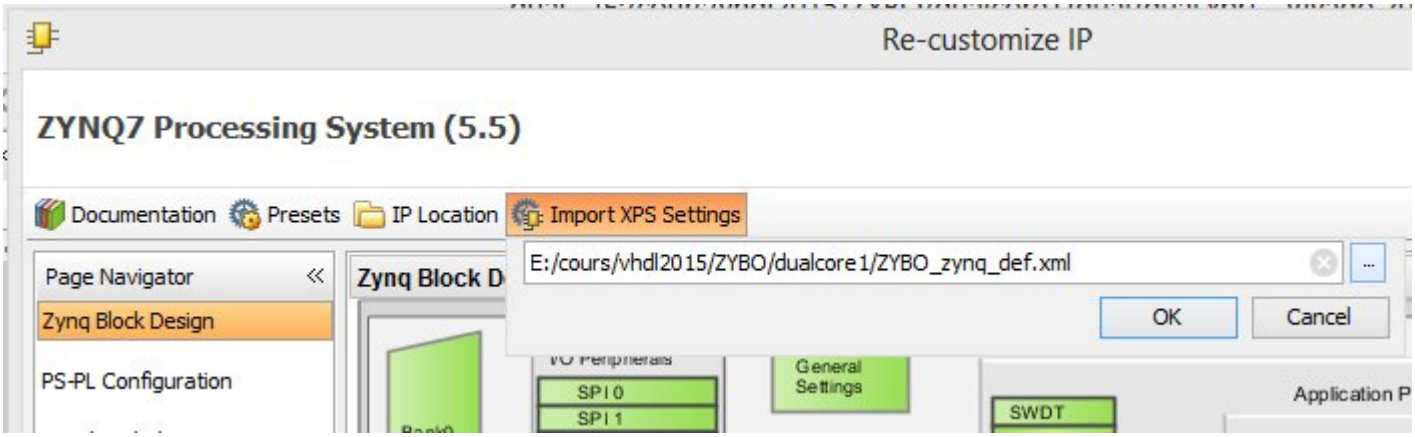


La connexion au bus AXI4 lite apparait

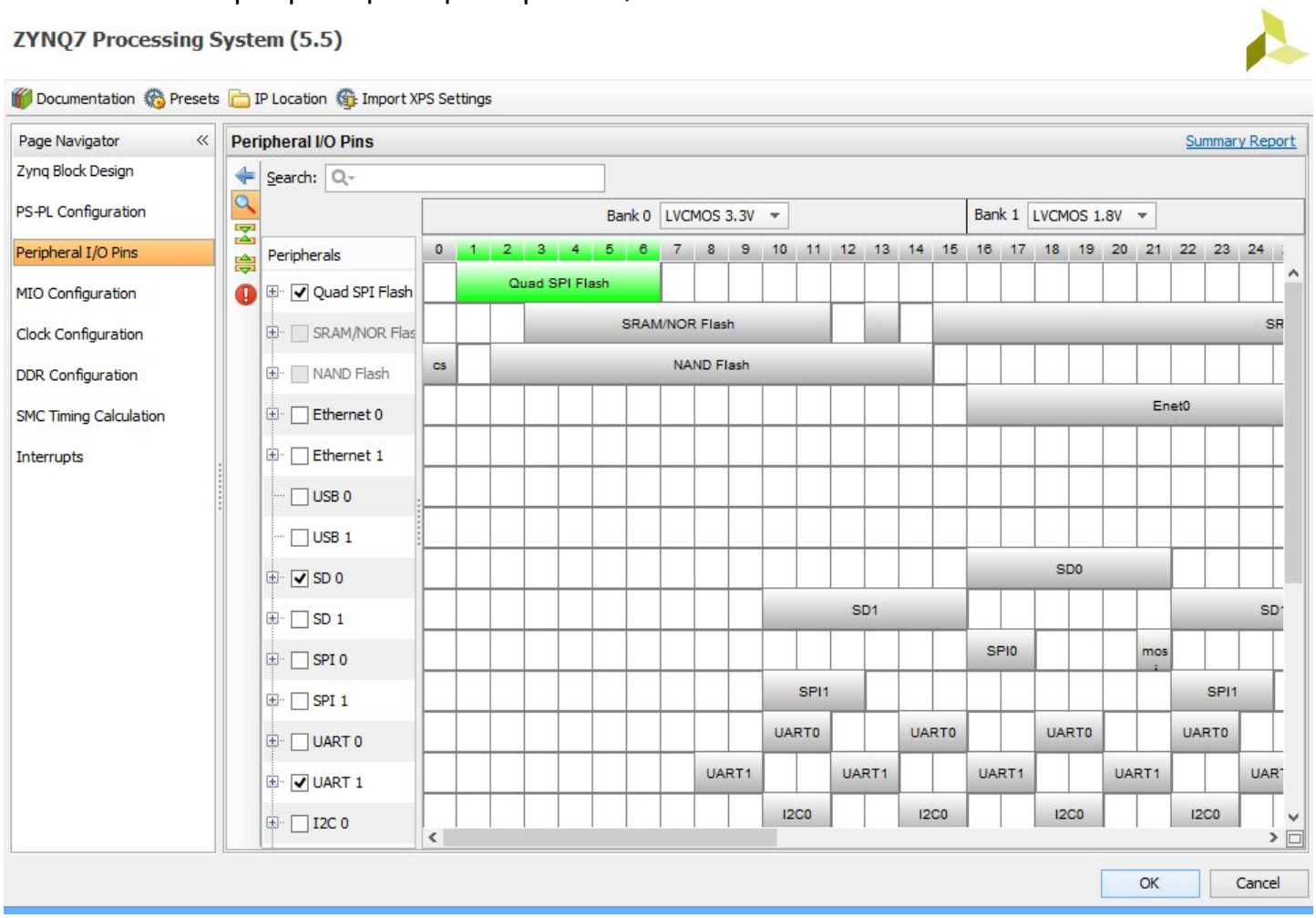


Double cliquez sur le PS et importez le fichier XPS de configuration du PS pour la carte ZYBO

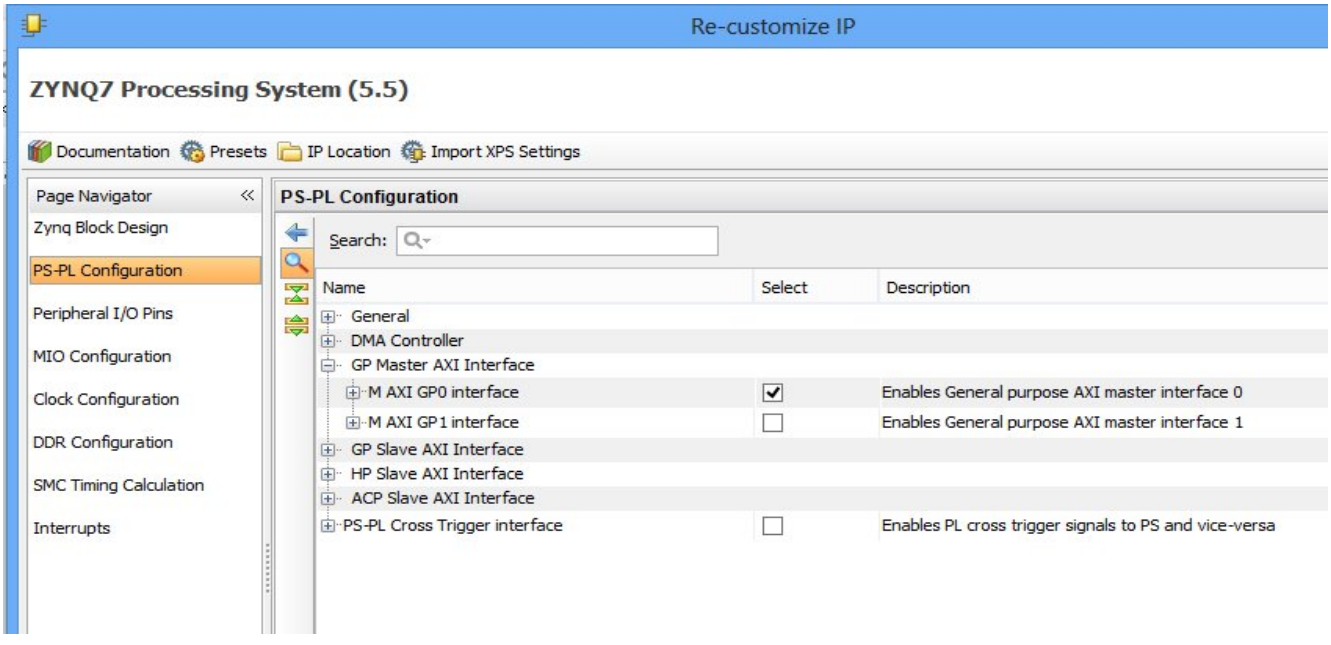


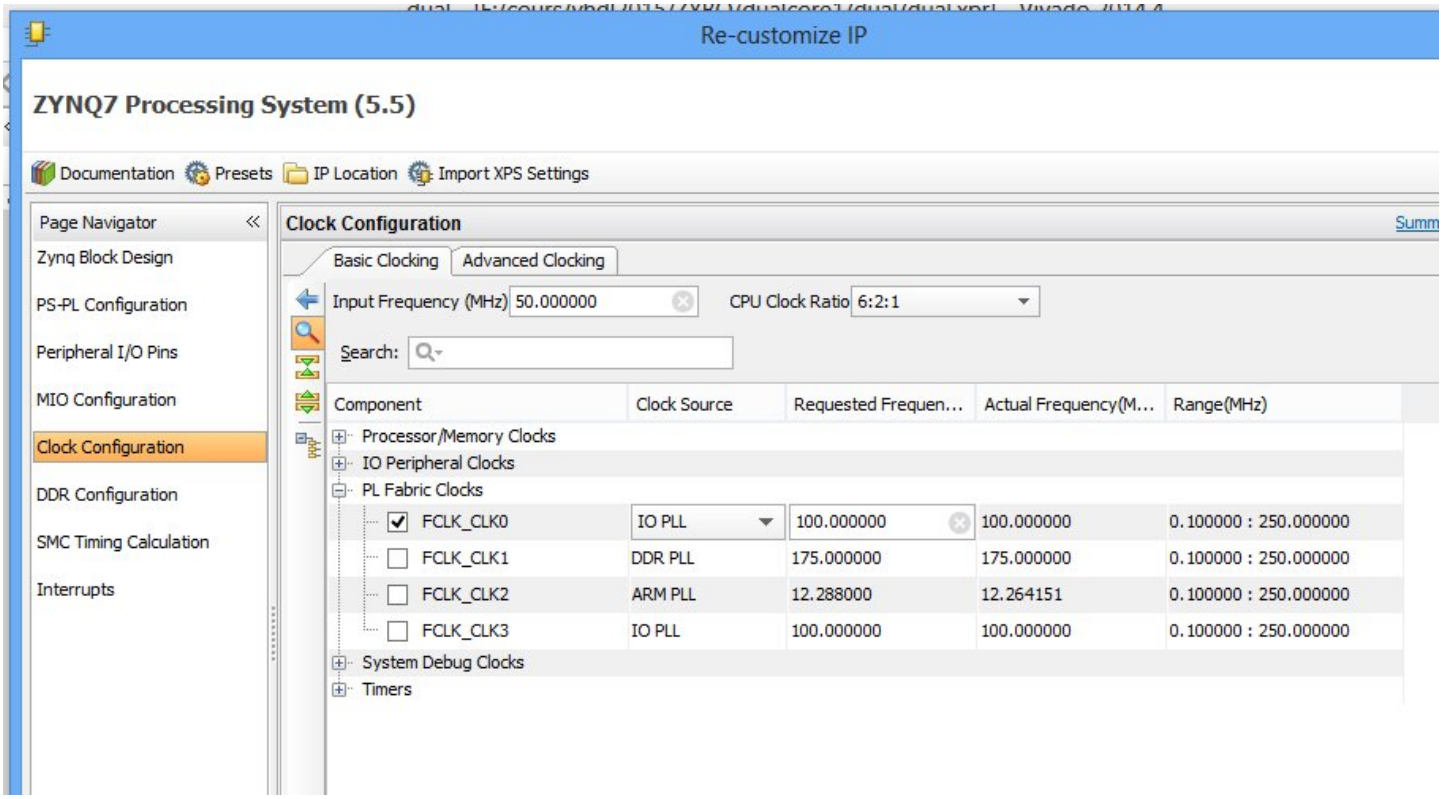


Sélectionnez les périphériques quad spi flash, SD0 et l'UART1

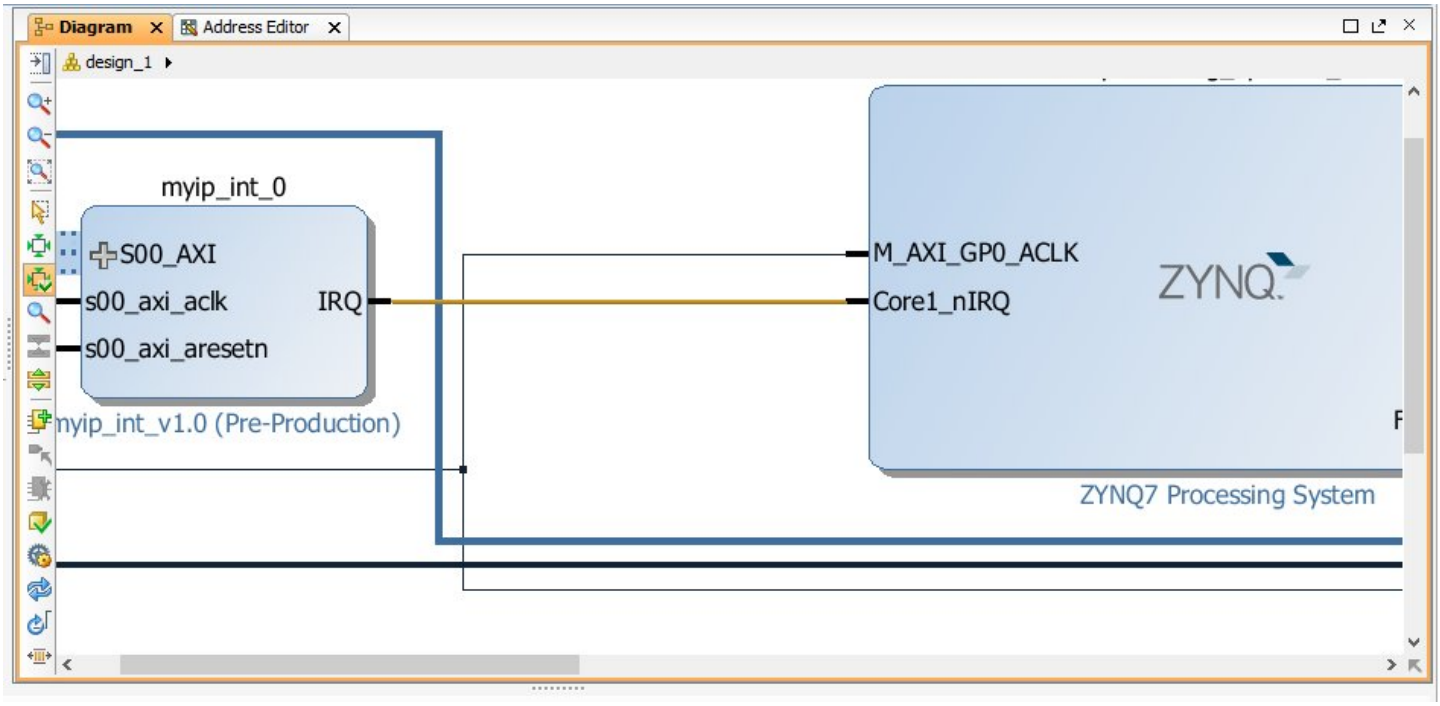


Configurez les éléments pour le dialogue avec le PL sur le bus AXI

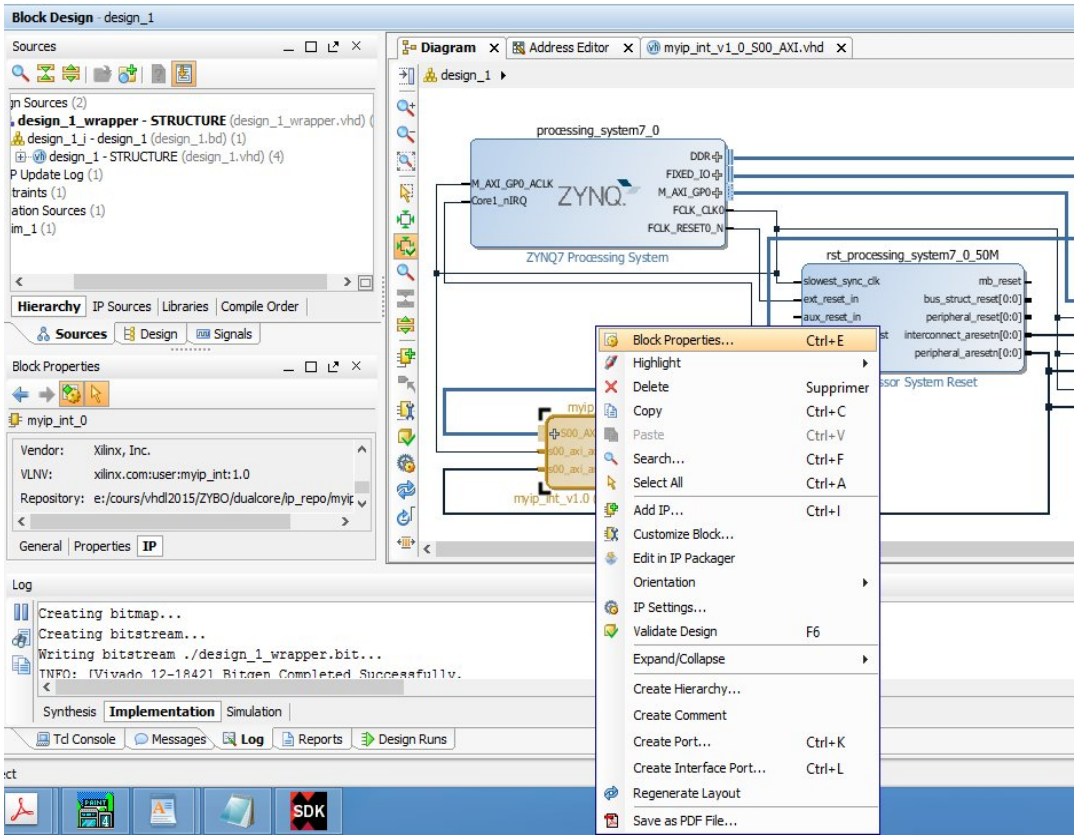




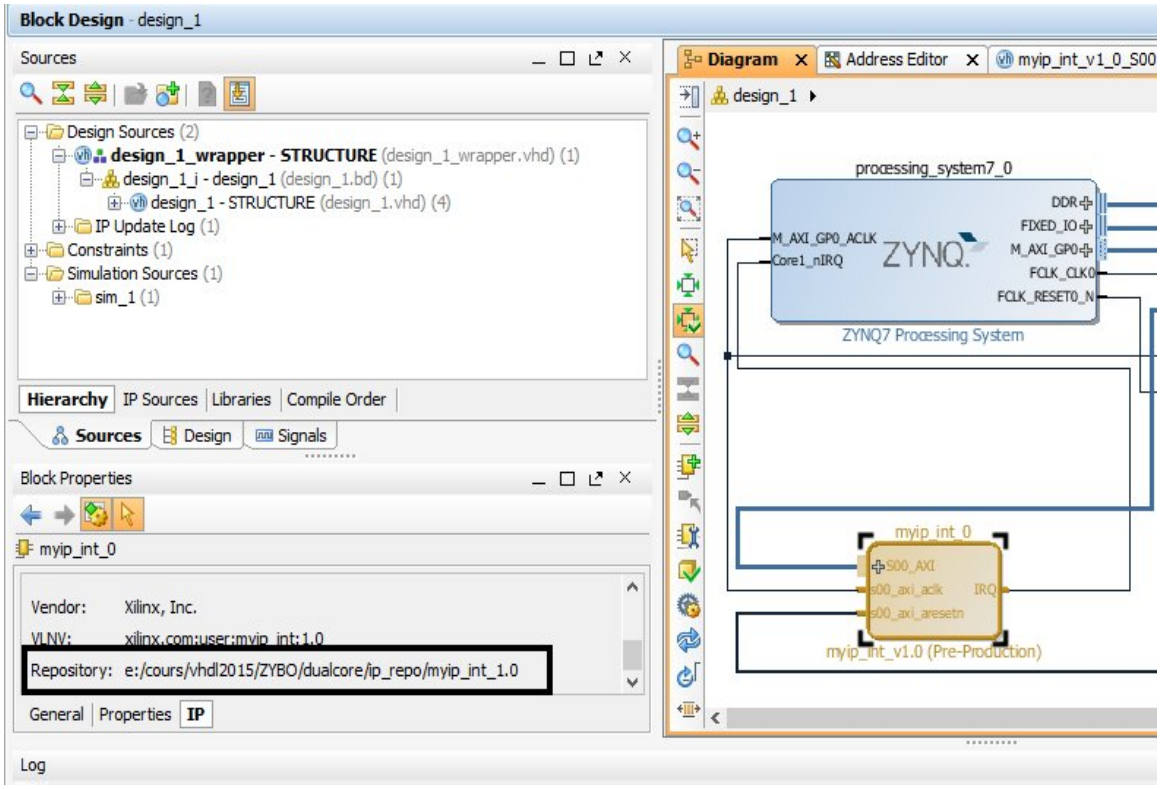
Réalisez la connexion matérielle pour l'interruption



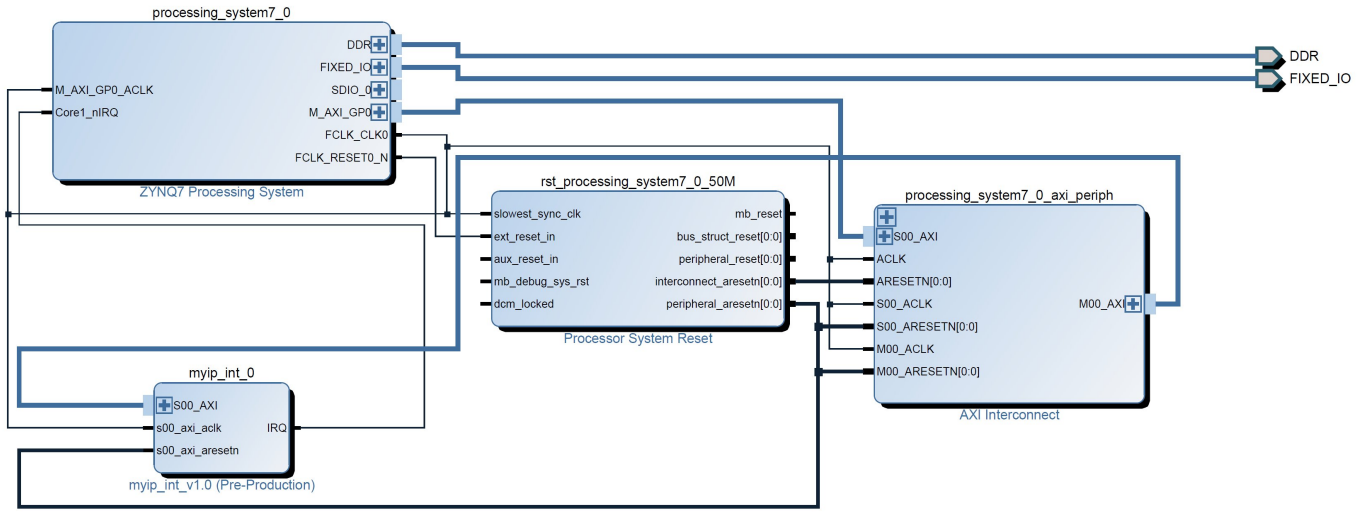
Visualisant les propriétés de l'IPCORE myip_int



L'encadré nous donne la position d'enregistrement des fichiers de l'IPCORE



Le schéma final est le suivant :



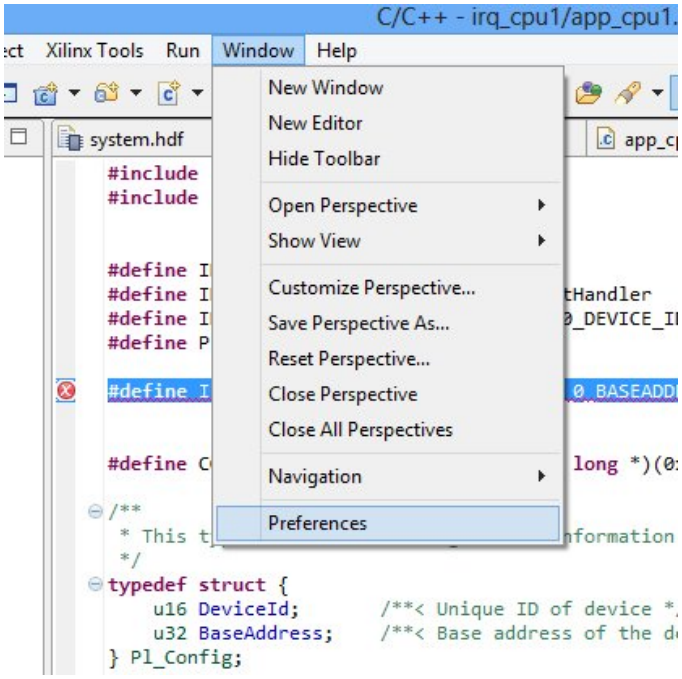
Dans ce système Le CPU0 du ZYNQ partage dans la mémoire OCM une case mémoire @ `0xFFFF0000 avec le CPU1.` Cette case est une sorte de flag ou de jeton.
Le CPU0 prend la ressource partagé UART1 lorsque le flag ou le jeton est à 0 et affiche
CPU0: Hello World CPU 0
Le CPU1 prend la ressource partagé UART1 lorsque le flag ou le jeton est à 1 et affiche
CPU1: Hello World CPU 1
Les deux cœurs se donnent alternativement le jeton afin que chacun puisse prendre la main sur l'UART1
On peut aussi utiliser le CPU0 pour générer une interruption et lancer la routine d'interruption du CPU1 qui redonnera par la suite la main au CPU0 grâce au jeton.

L'applicatif du CPU0 se comporte de la façon suivante :

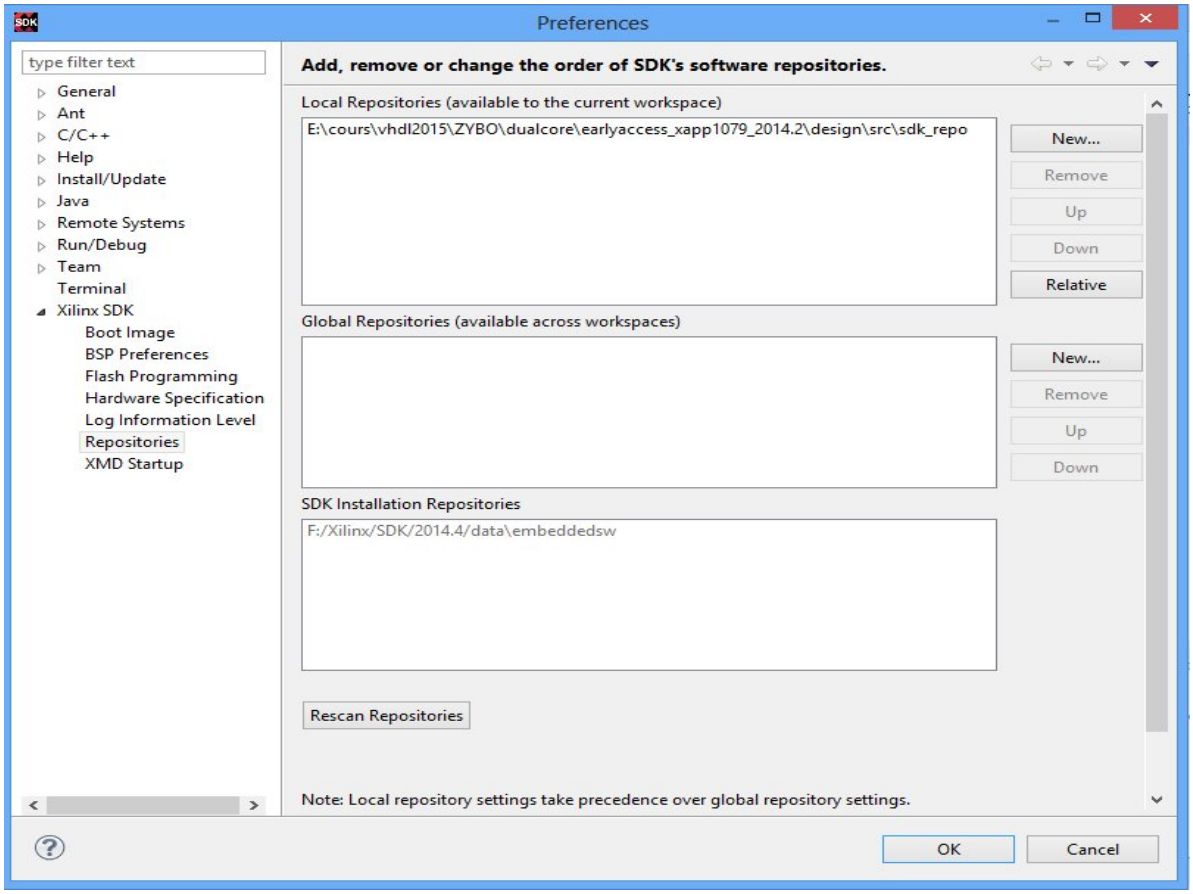
- Il configure le MMU afin de désactiver pour l'OCM dans la gamme d'adresse de `0xFFFF0000` à `0xFFFFFFFF`. Ainsi l'OCM existe à l'adresse `0x00000000` à `0x0002FFFF et de 0xFFFF0000` to `0xFFFFFFFF. Les 64KB supérieurs de l'OCM sont utilisé ainsi la mémoire cache est désactivée de 0xFFFF0000` to `0xFFFFFFFF`.
- Il initialise l'ICD
- Il démarre le CPU1
- Il affiche Hello world CPU O grâce à l'UART.
- Il prépositionne dans la mémoire OCM le jeton
- Il attend que le flag ou le jeton passe à 0

Le CPU0 démarre le CPU1 en écrivant la valeur de 0x00200000 à l'adresse 0xFFFFFFF0 et il démarre la commande SEV the Set Event (SEV). La commande SEV démarre le CPU1 qui lit la valeur 0x00200000 à l'adresse 0xFFFFFFF0 et alors démarre à l'adresse 0x00200000 de la RAM.
Le FSBL se charge de placer le code applicatif du CPU1 à l'adresse 0x00200000 de la mémoire RAM DDR3

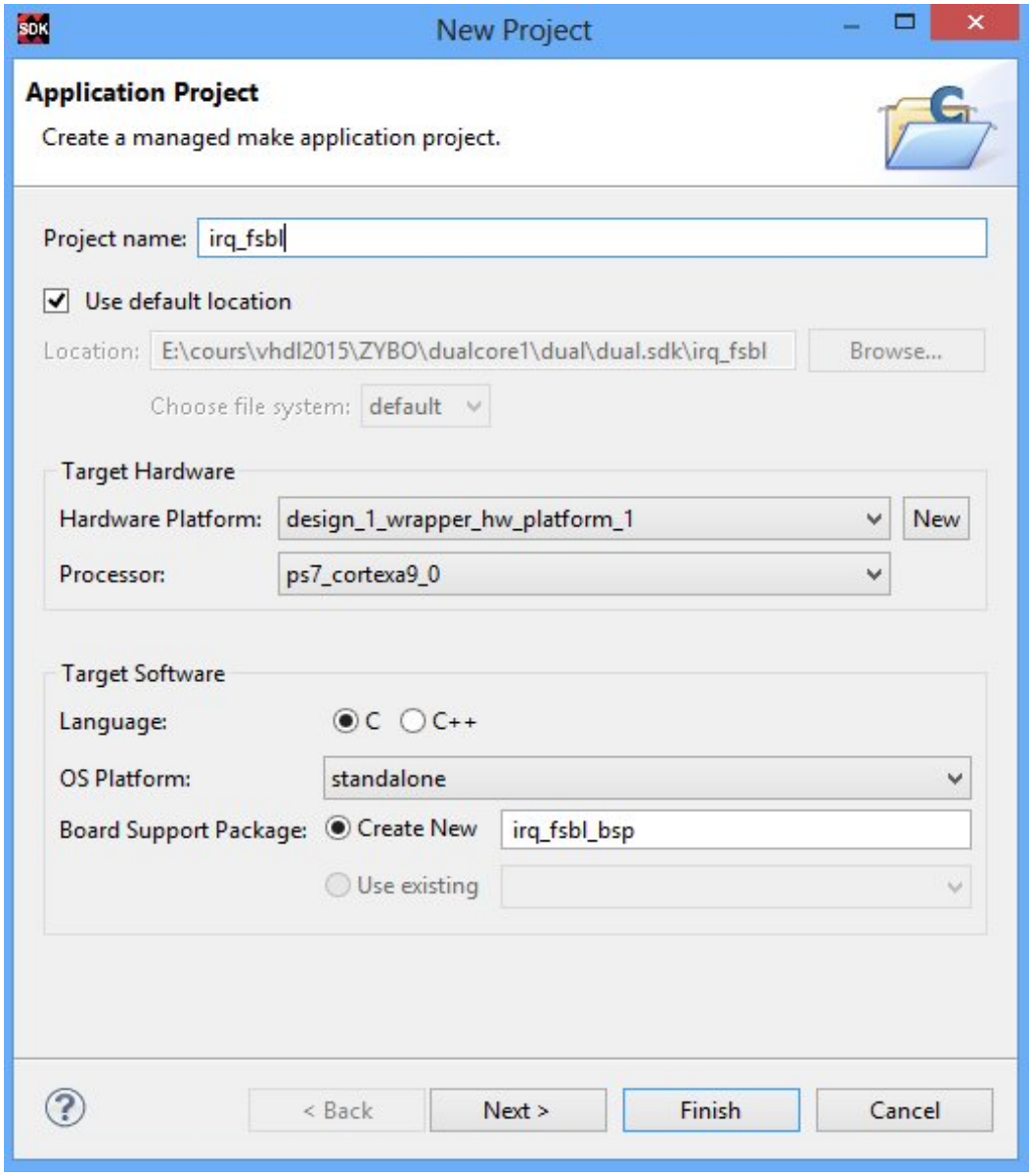
Lancez le logiciel SDK. On doit intégrer un nouveau BSP dans le repository, afin de gérer les doubles cœurs



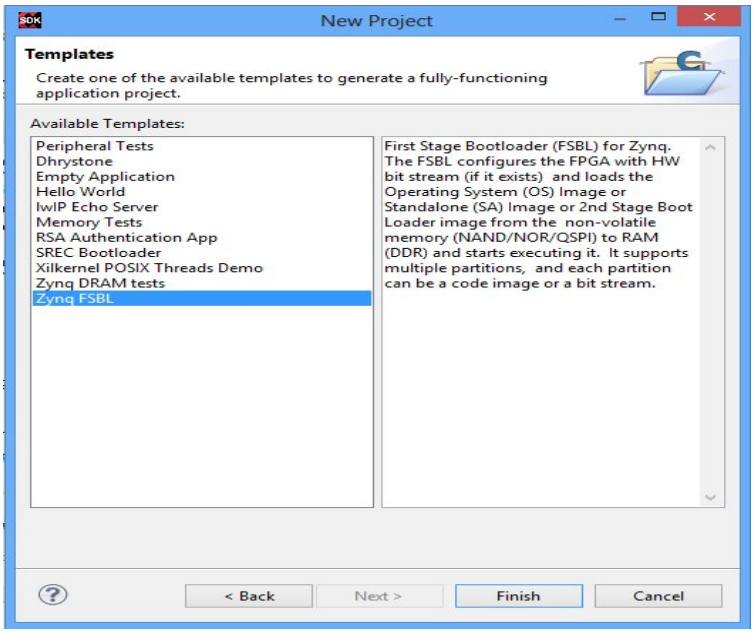
Choisir le BSP earlyaccess_xapp1079_1024 pour la suite cliquez sur apply



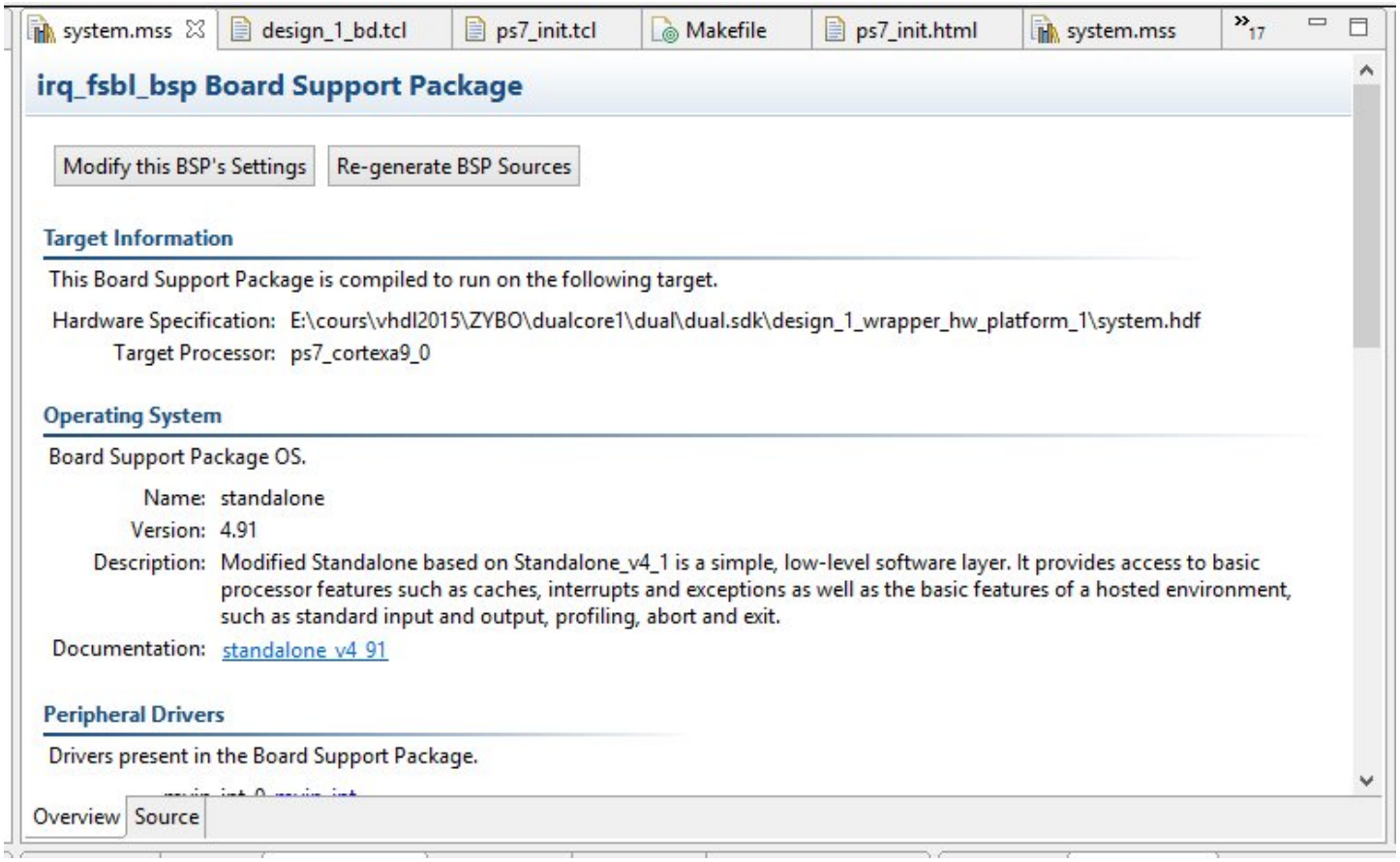
Créez une nouvelle application pour le FSBL



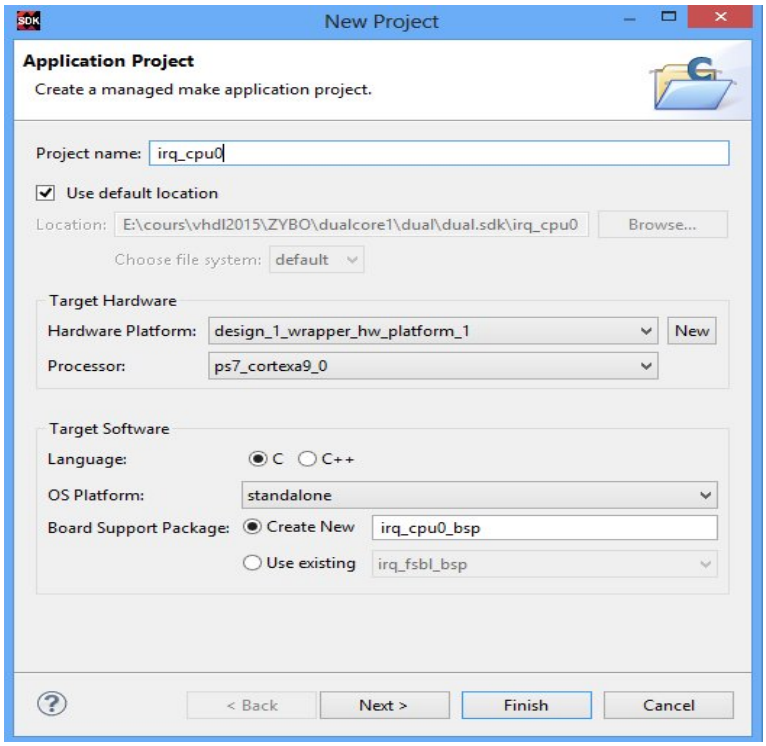
L'application sera du type FSBL



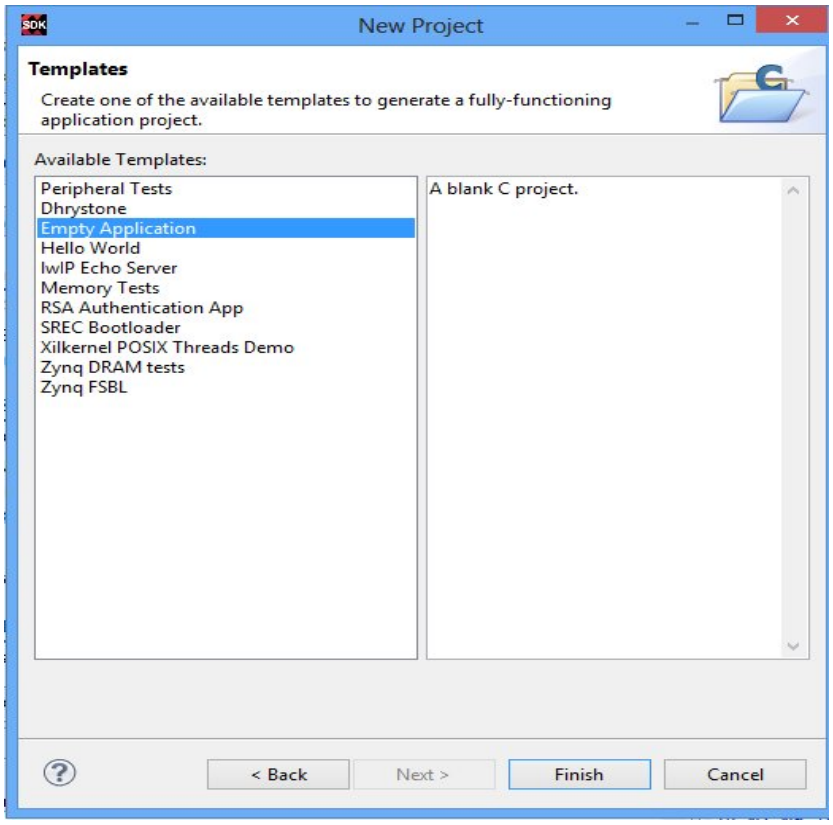
Le board support package est bien un standalone modifié de la version 4.1



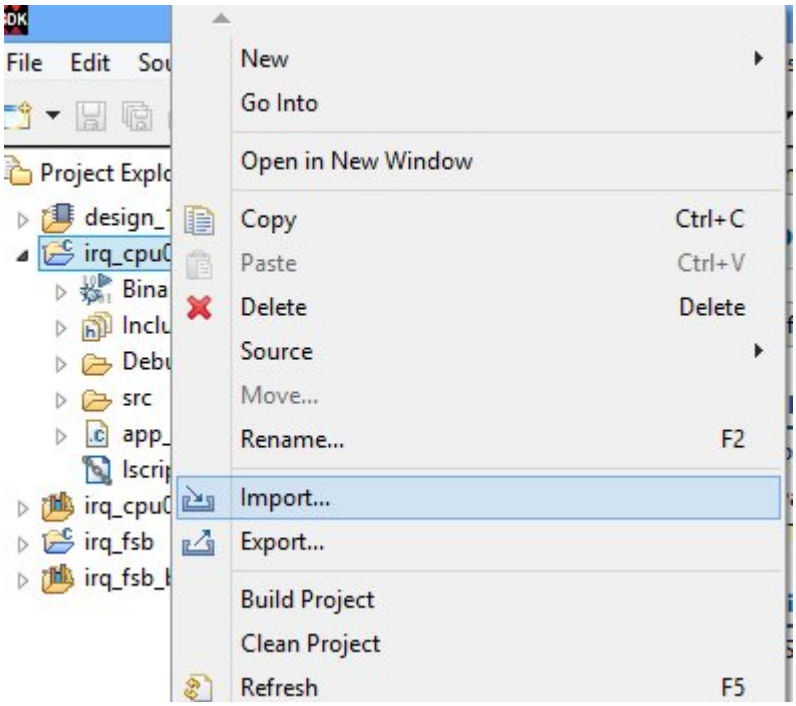
Créez une nouvelle application pour le cœur CPU0

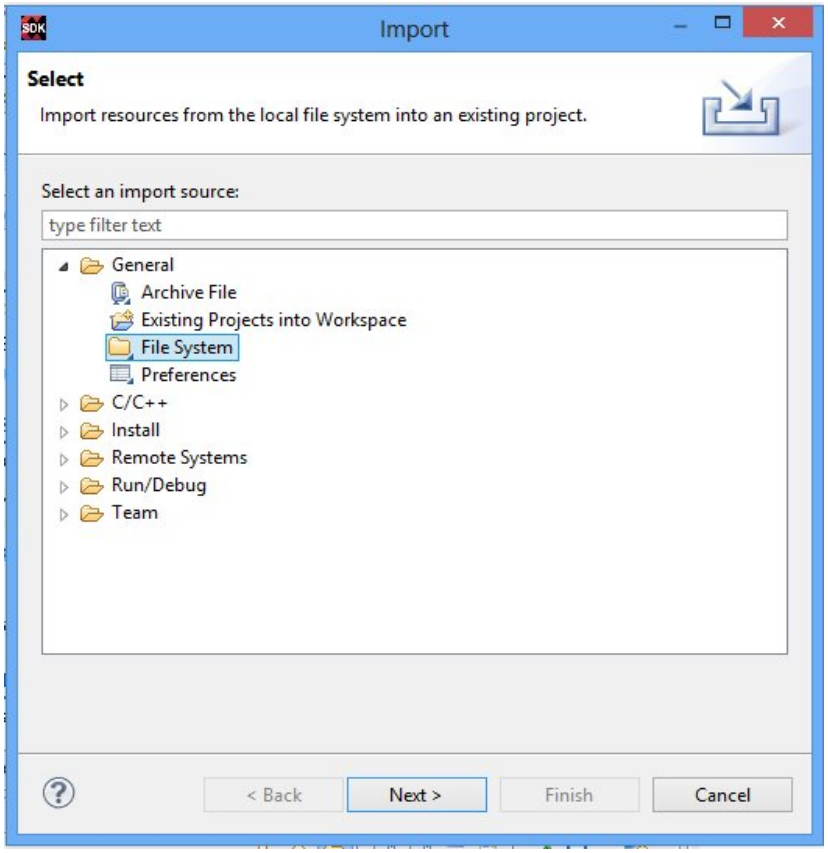


L'application sera vide on importera par la suite les fichiers sources

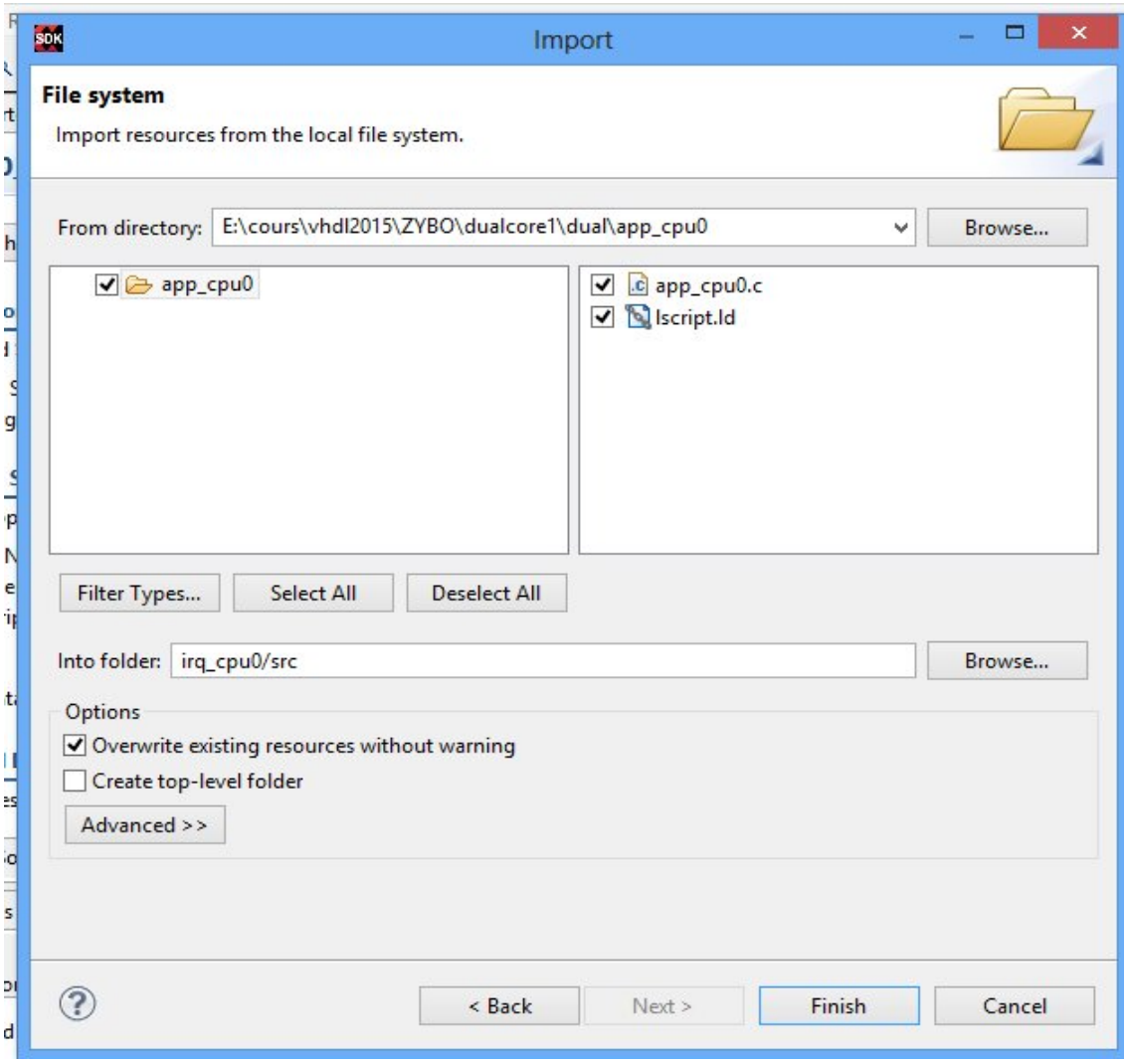


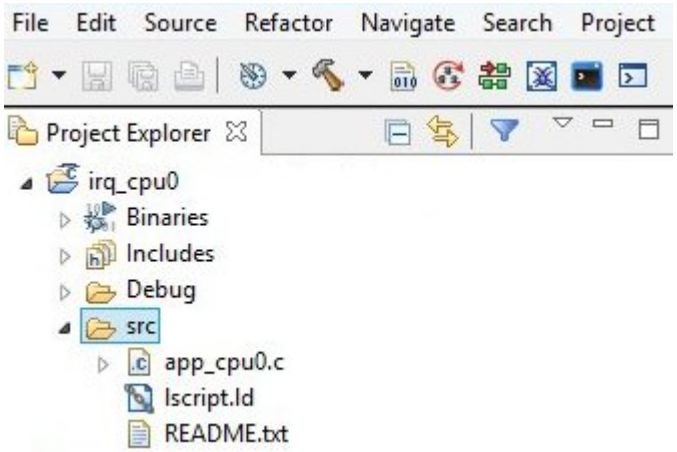
Cliquez droit sur irq_cpu0



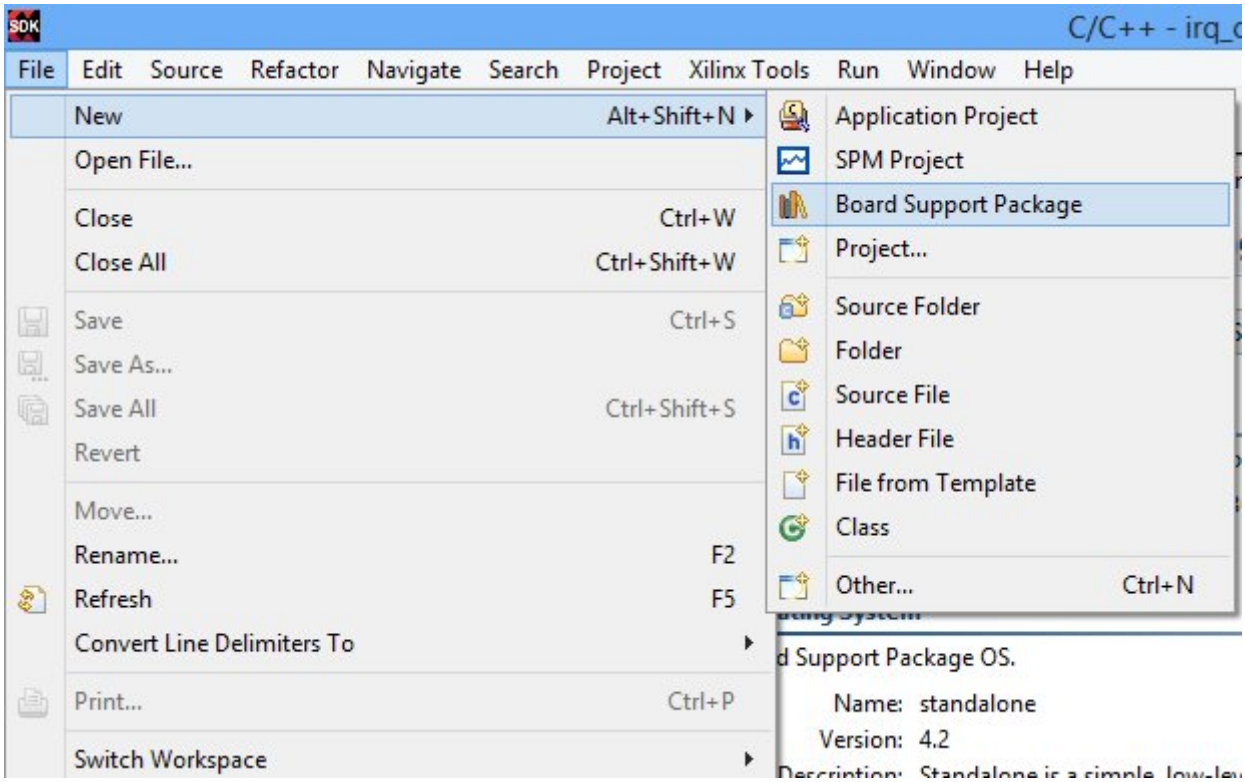


Importez les fichiers suivants pour le CPU0

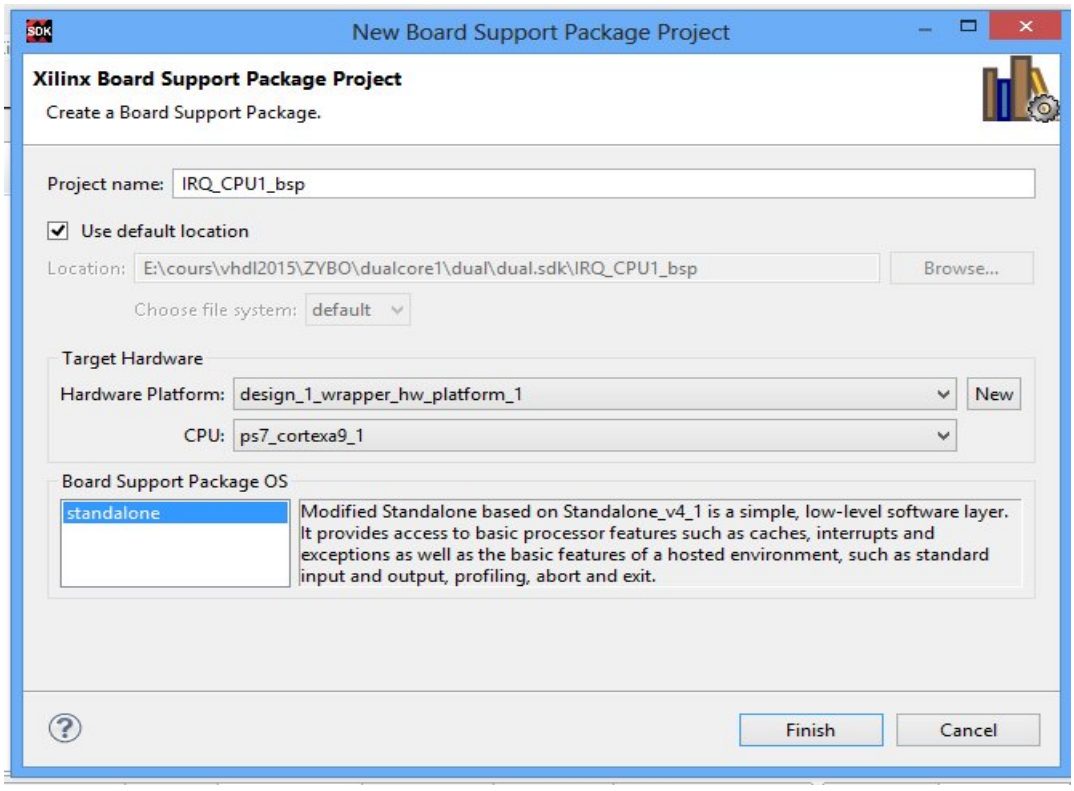




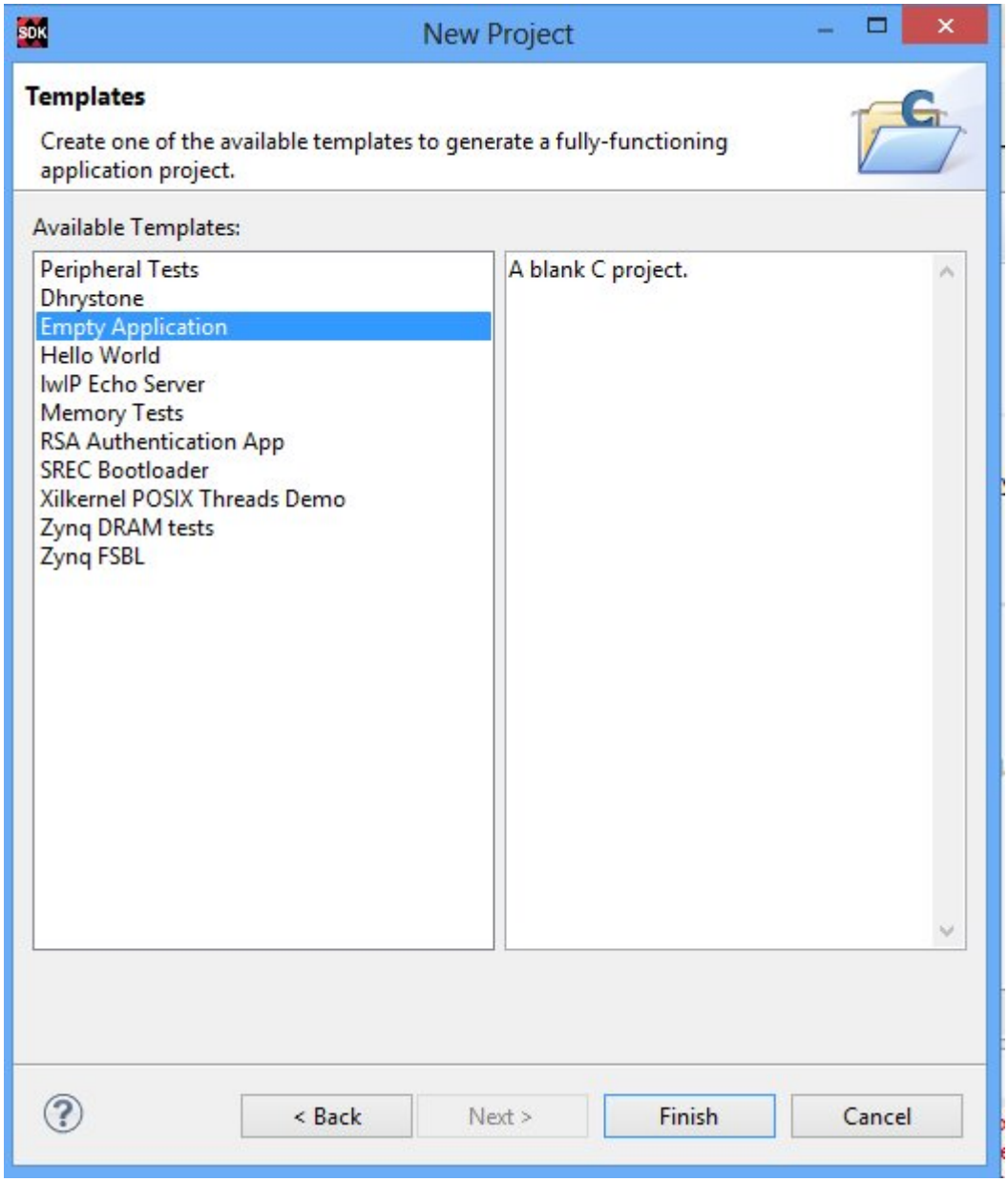
Créez un board support package pour le CPU1



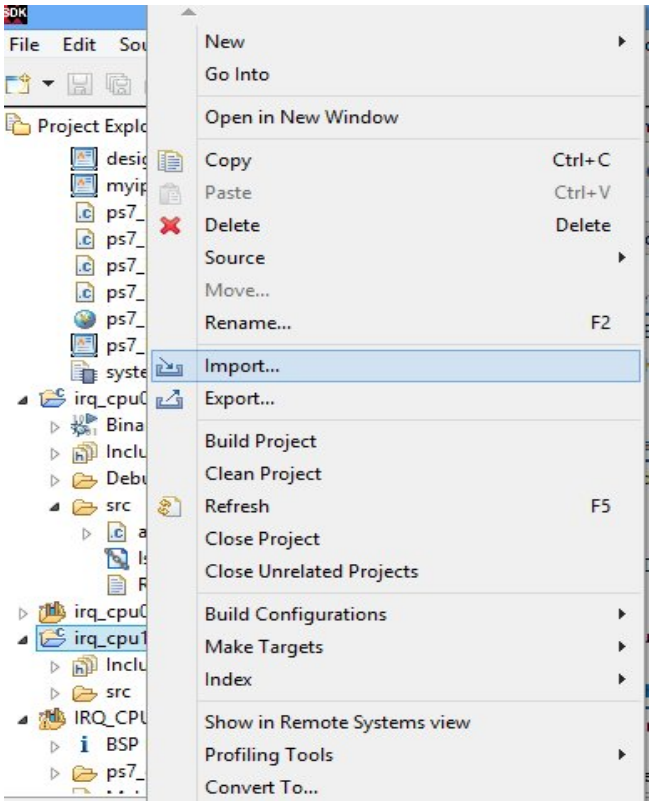
Donnez un nom au BSP et bien choisir le CPU1 ps7_cortex9_1

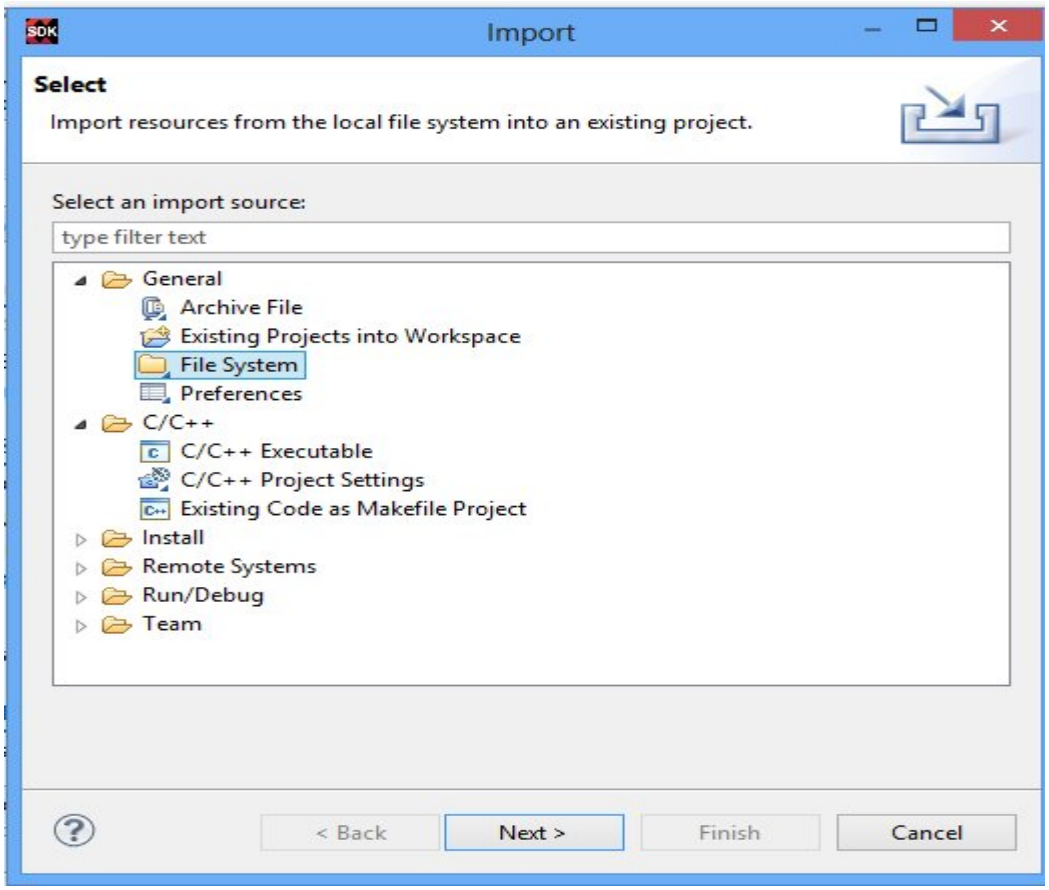


L'application sera vide on importera par la suite les fichiers sources

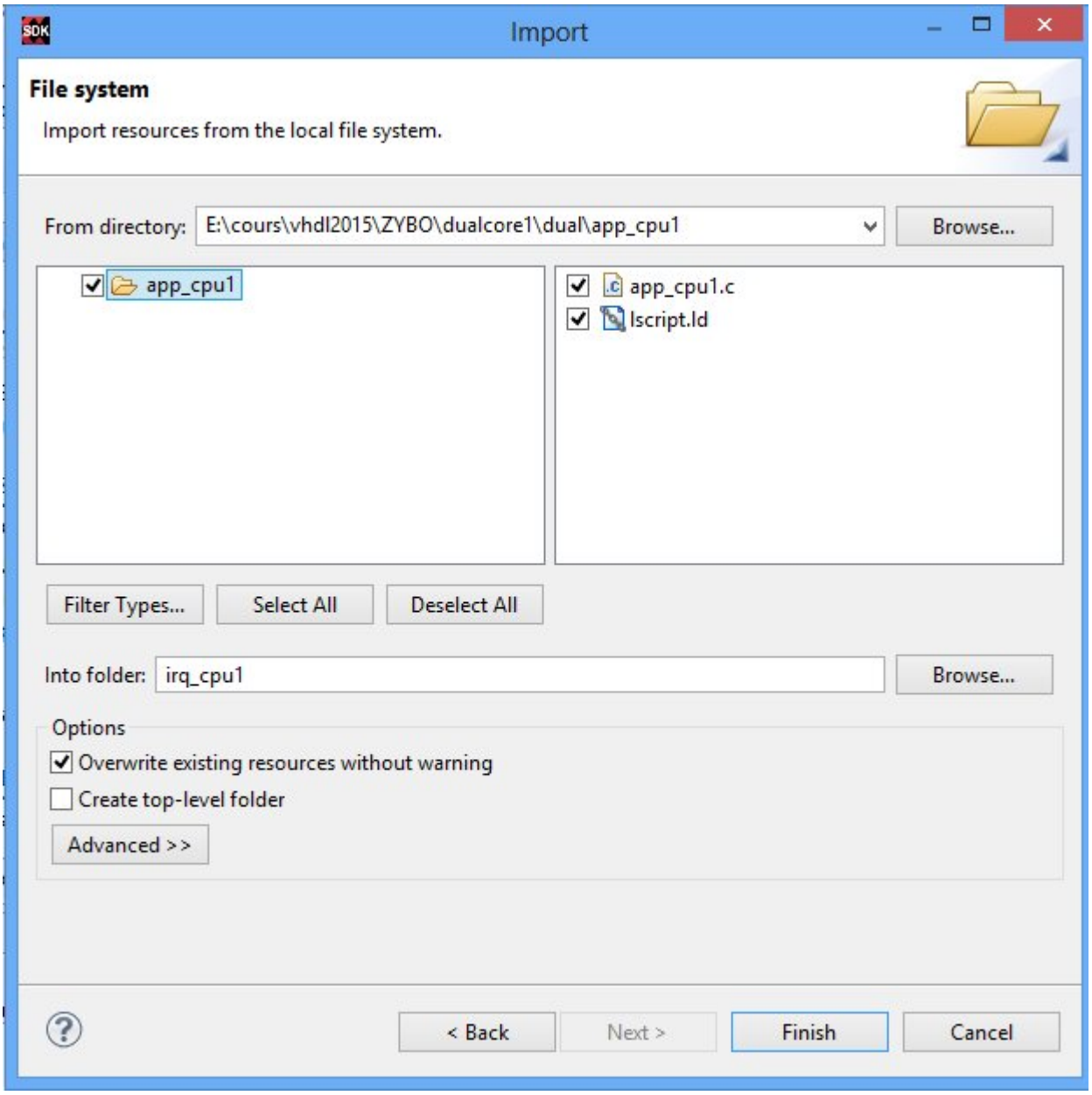


Cliquez droit sur irq_cpu1 puis import

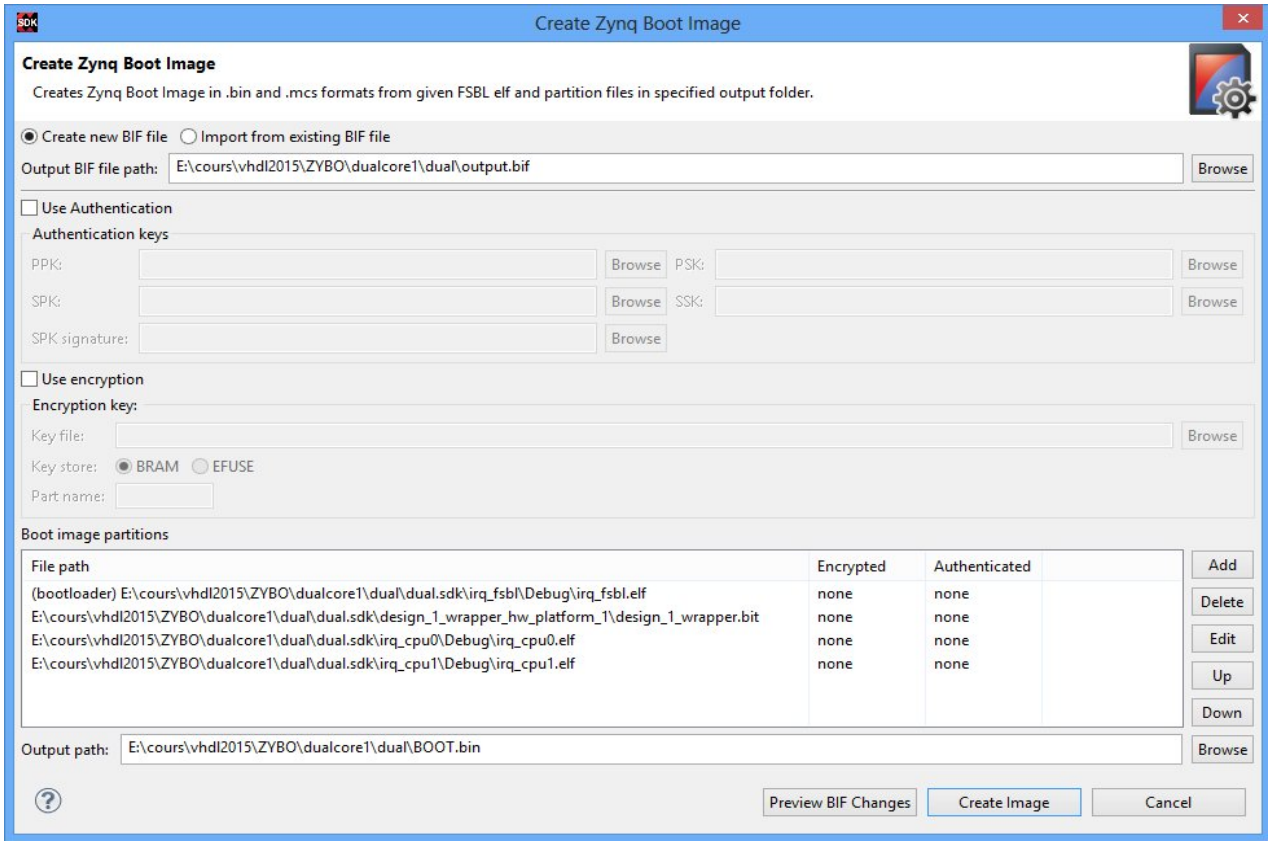




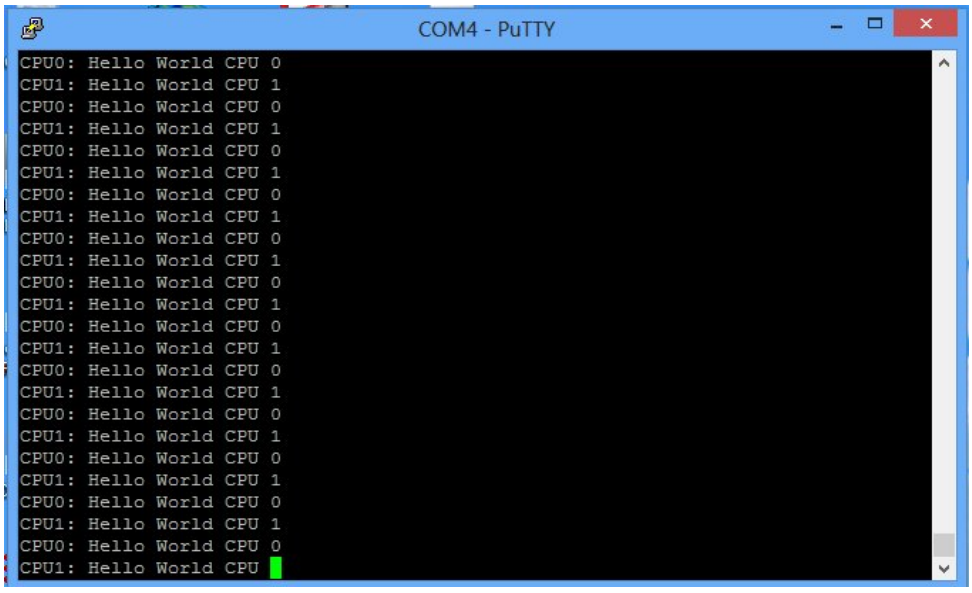
Choisissez les fichiers suivants :



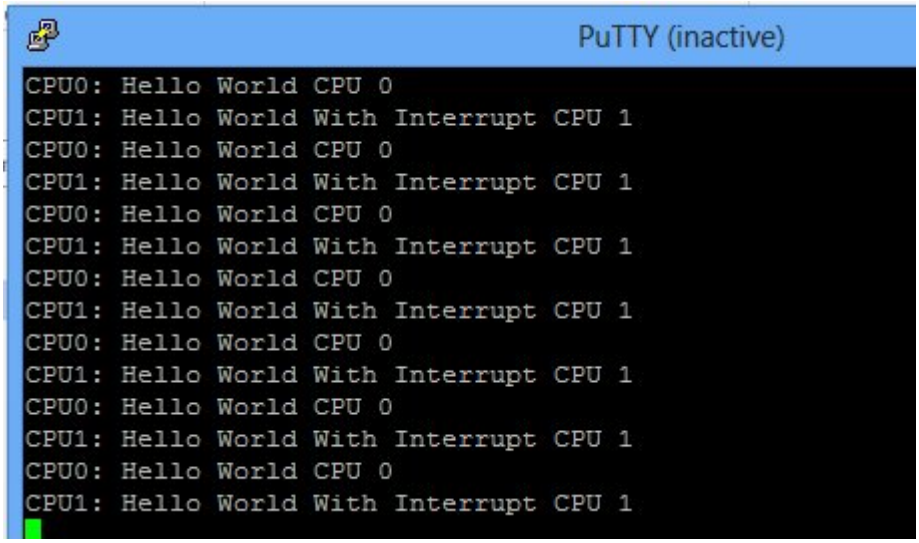
Afin de rendre l'application autonome on peut par la suite créer un Zynq Boot image pour la carte SD



La communication entre les deux processeurs se visualise sur le port console de la JTAG UART



En utilisant la gestion des interruptions par le CPU1, l'applicatif nous donne comme résultat :



Le terme Asymmetric multi-processing (AMP) peut être utilisé sur un système qui utilise de multiples cœurs qui peuvent être d'architecture différente. Chaque cœur peut commencer son propre OS. La communication entre les cœurs de CPU est facilitée grâce à l'utilisation de la mémoire partagée.

Le terme Symmetric multi-processing (SMP). Un simple OS se partage tous les cœurs afin de distribuer et de coordonner les différentes tâches entre elles. Comme avec l'AMP un espace de mémoire partagé est utilisé pour la communication entre les cœurs de CPU

