Mise en place d'un AXI timer

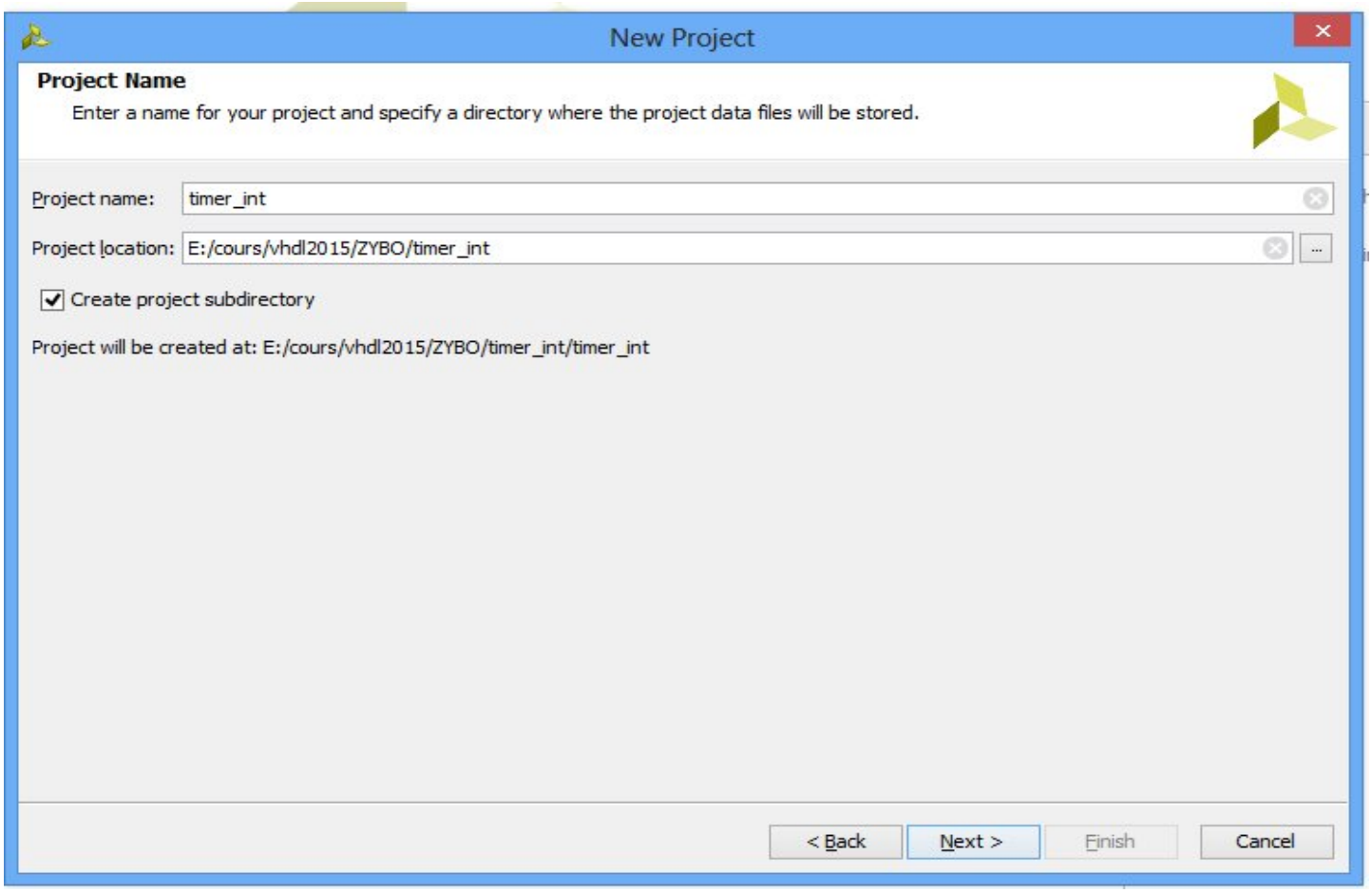
Ce tutorial permet d'allier à la fois de la logique programmable PL et le système de traitement PS. Le timer et le GPIO seront créés au sein d'IPCORE générique dans préfaçonné dans la logique programmable PL du FPGA.

1) Problématique

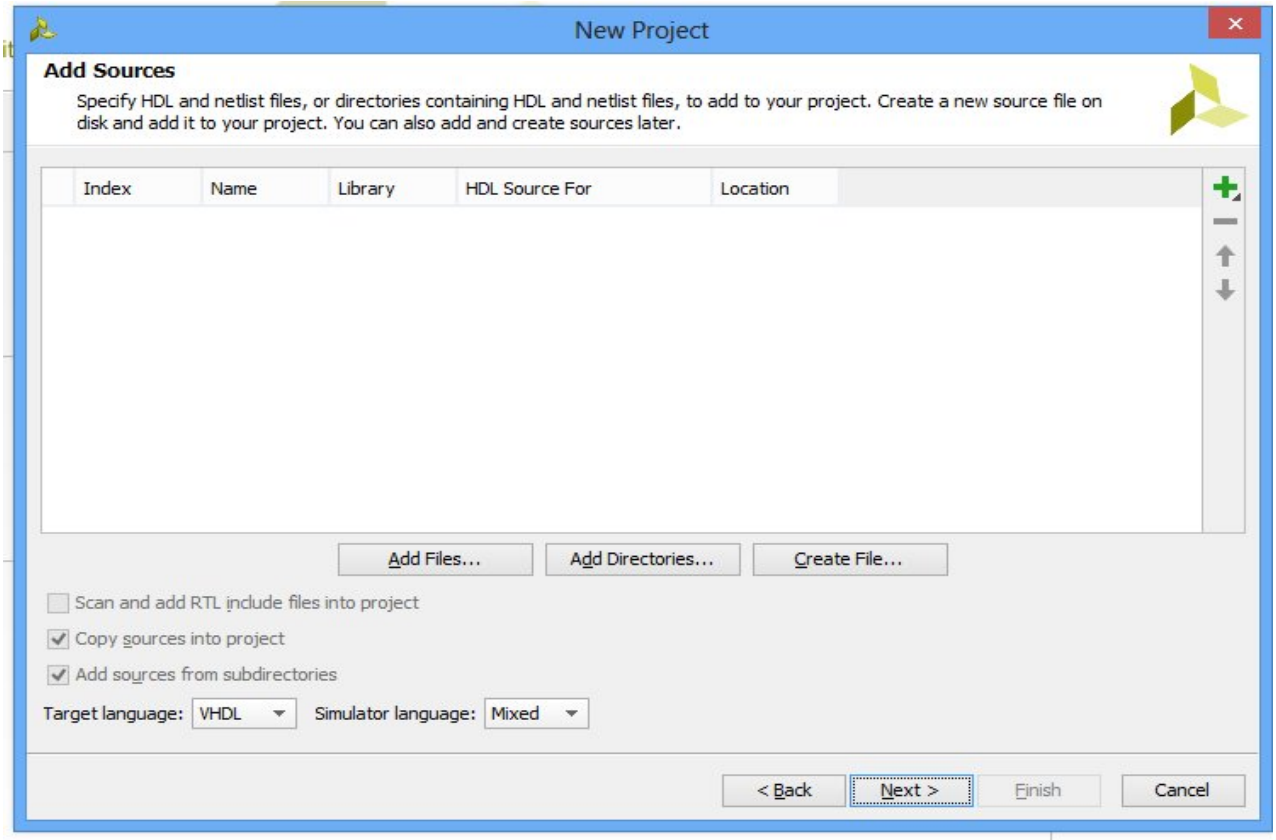
On souhaite faire clignoter une LED périodiquement grâce à un temporisateur.
La LED sera piloté grâce à un GPIO. La gestion du temps s'effectuera grâce à un mécanisme d'interruption. Le temporisateur ou timer générera une interruption à chaque fin de cycle de comptage.

2) Mise en place matérielle du projet

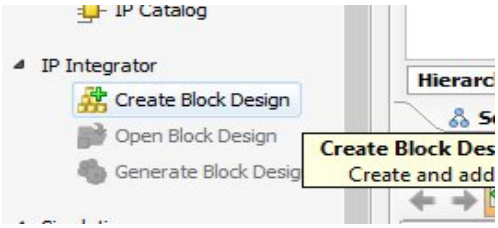
Créé un nouveau projet dans VIVADO



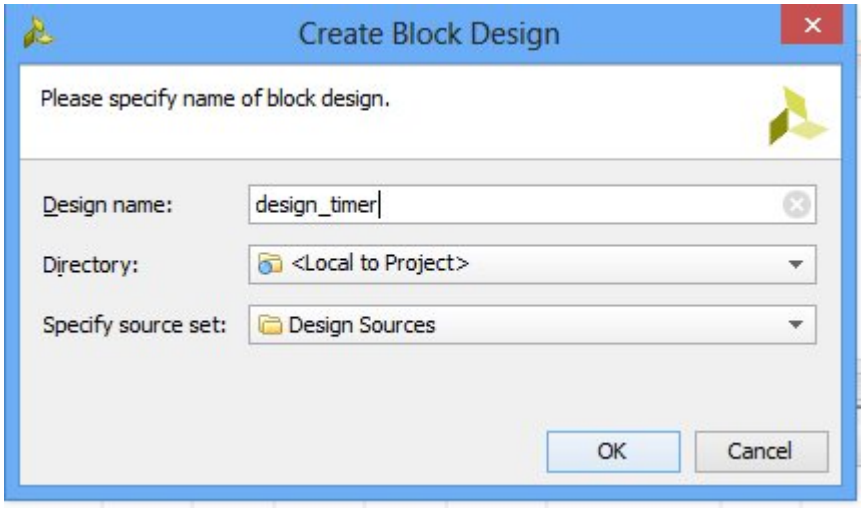
Aucun fichier source VHDL et d'IPCORE personnel a rajouté pour l'instant



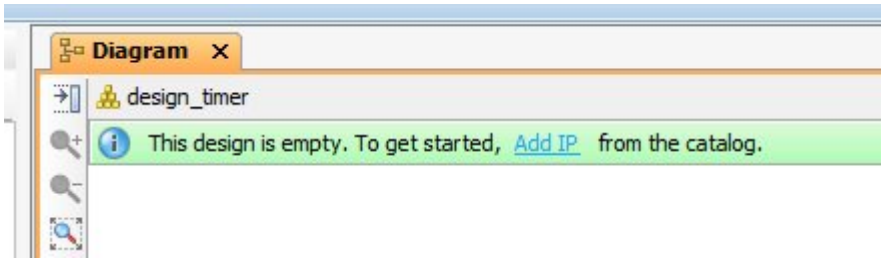
Créé un nouveau bloc design



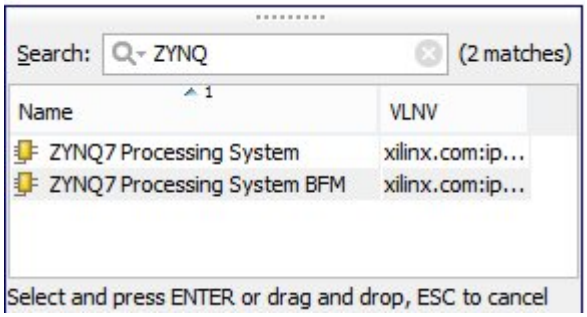
Donnez un nom au bloc design

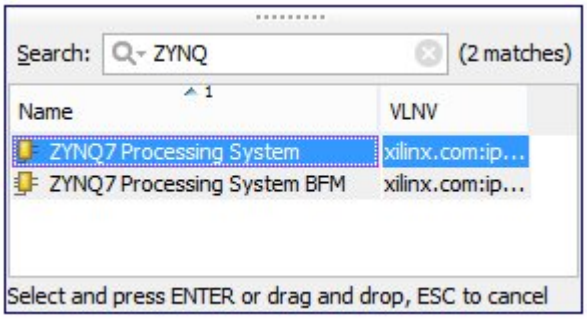


Cliquez sur add IP

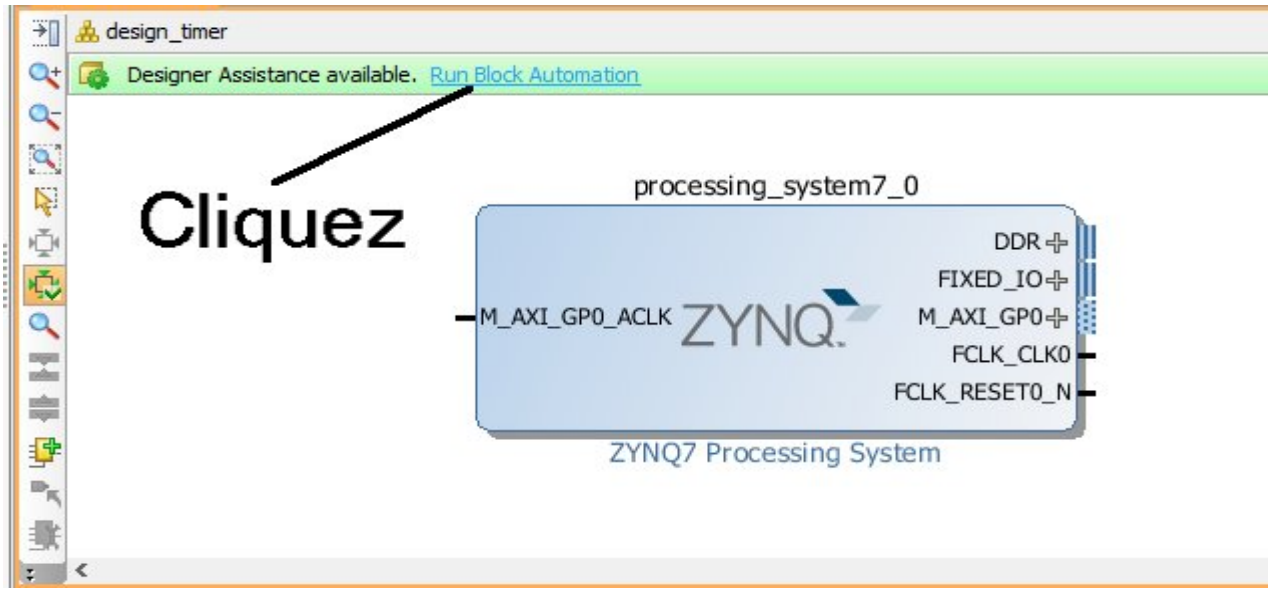


Choisir le système de traitement ZYNQ7



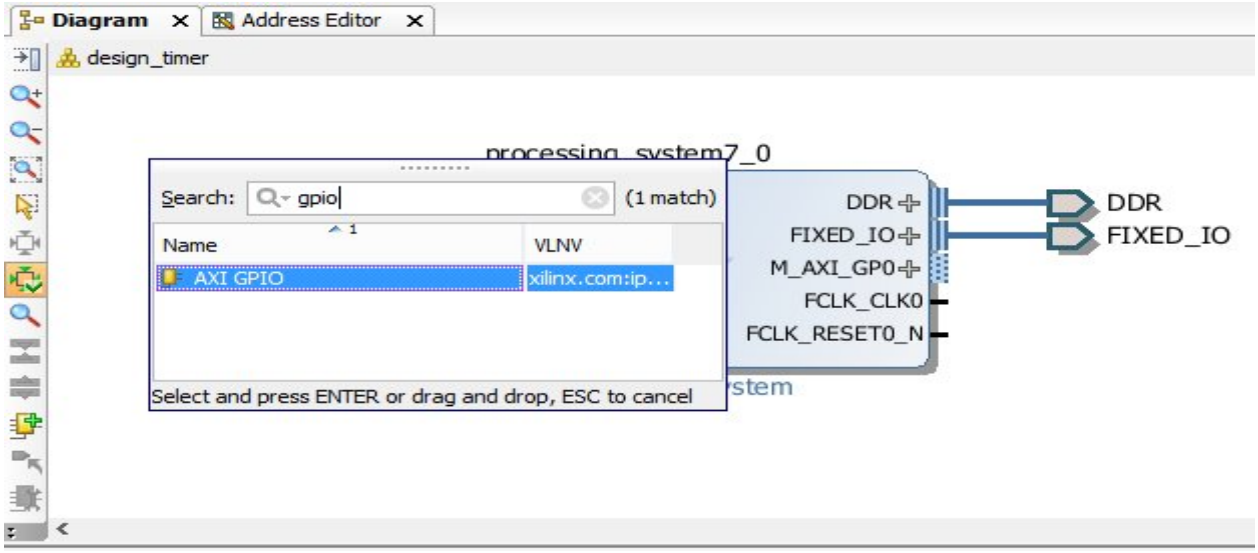


Automatisez la connexion du système de traitement avec la mémoire DDR3





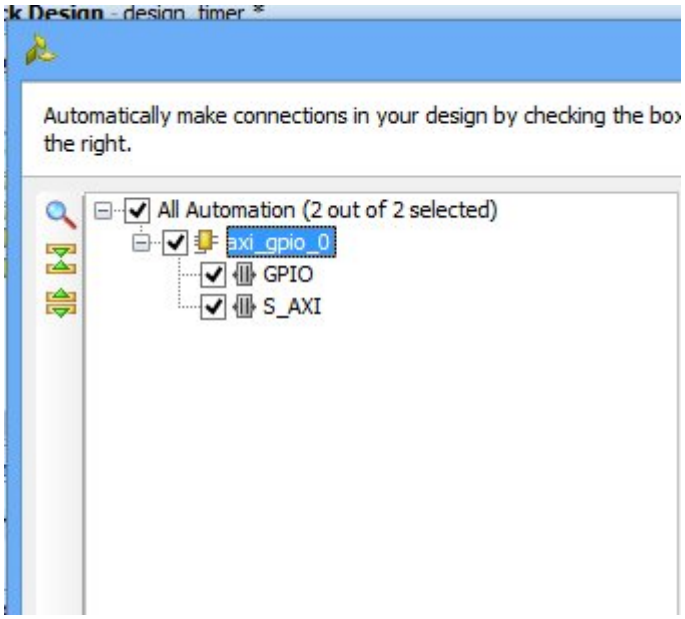
Rajoutez le bloc IP GPIO



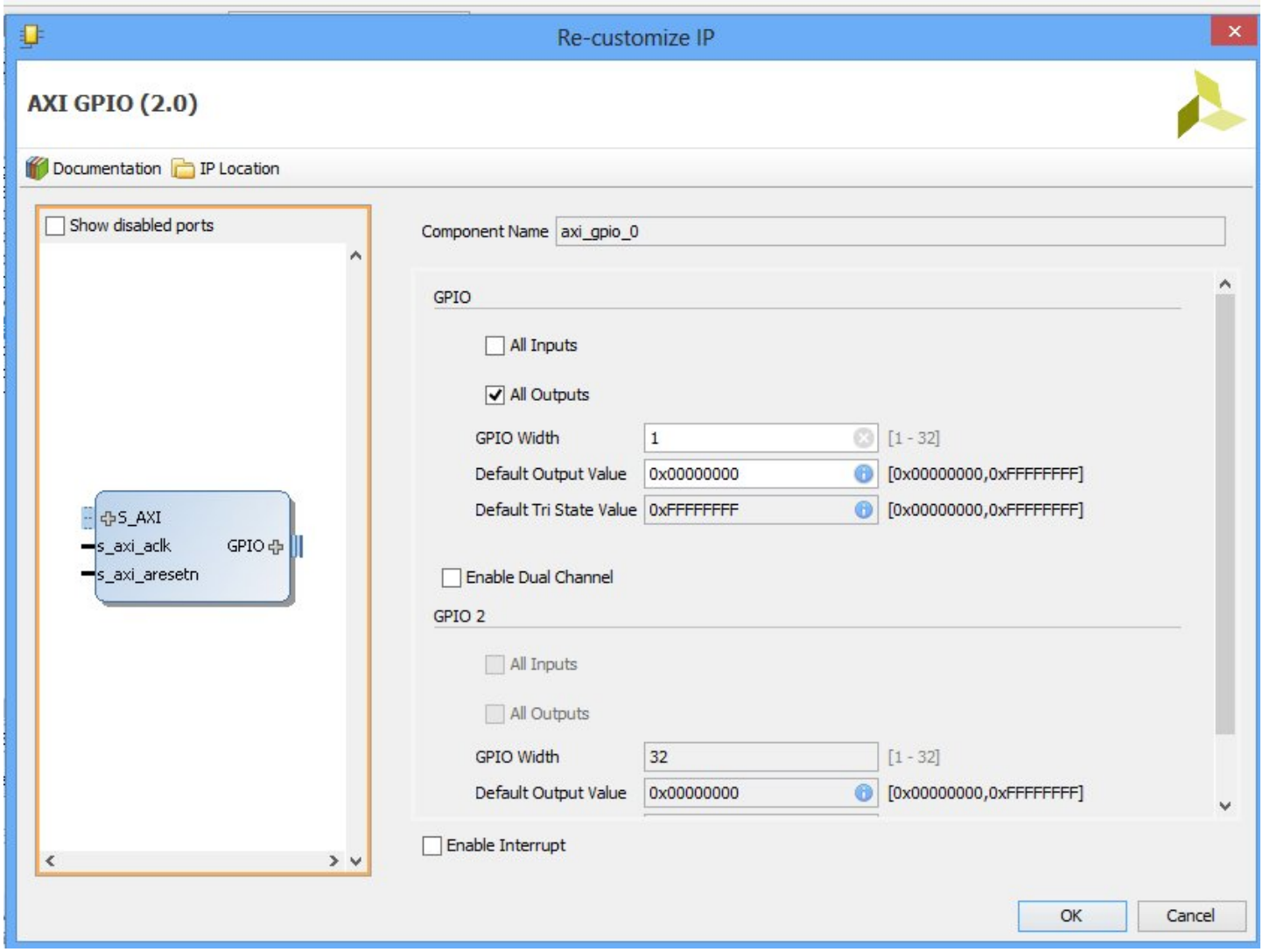
Mettre en place la connexion automatique de l'IPCORE AXI_GPIO avec le PS



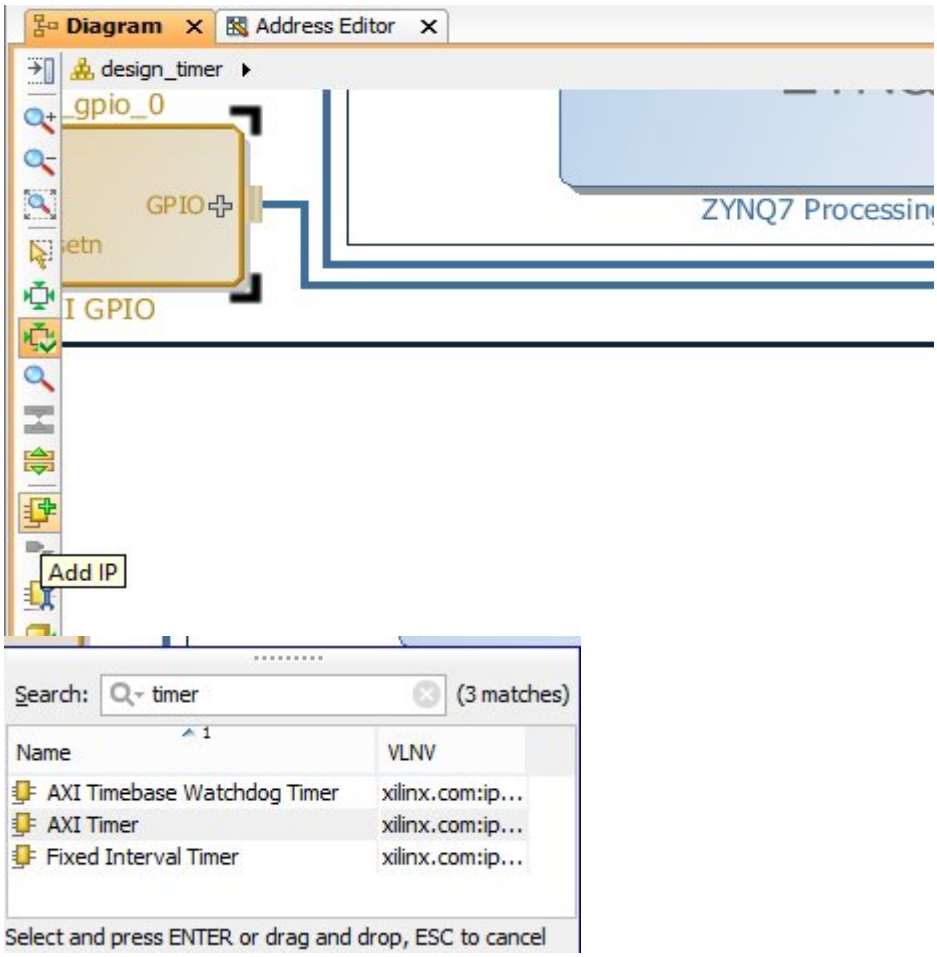
Le message suivant nous indique que l'IPCORE GPIO est connecté comme esclave sur le bus AXI vers le PS



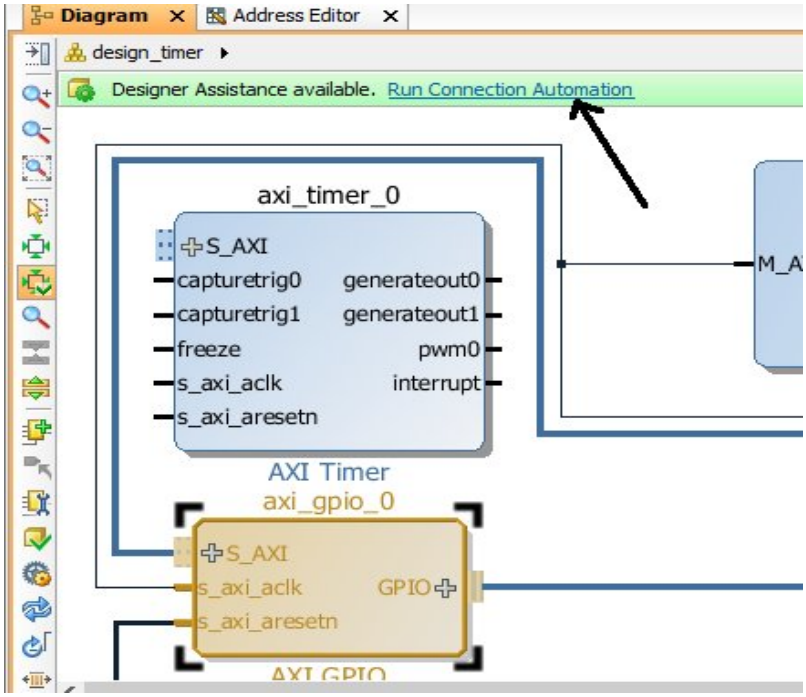
En double cliquant sur l'IPCORE AXI_GPIO on obtient ces caractéristiques. Le canal N°1 sera programmé en sortie et constitué que d'un seul bit

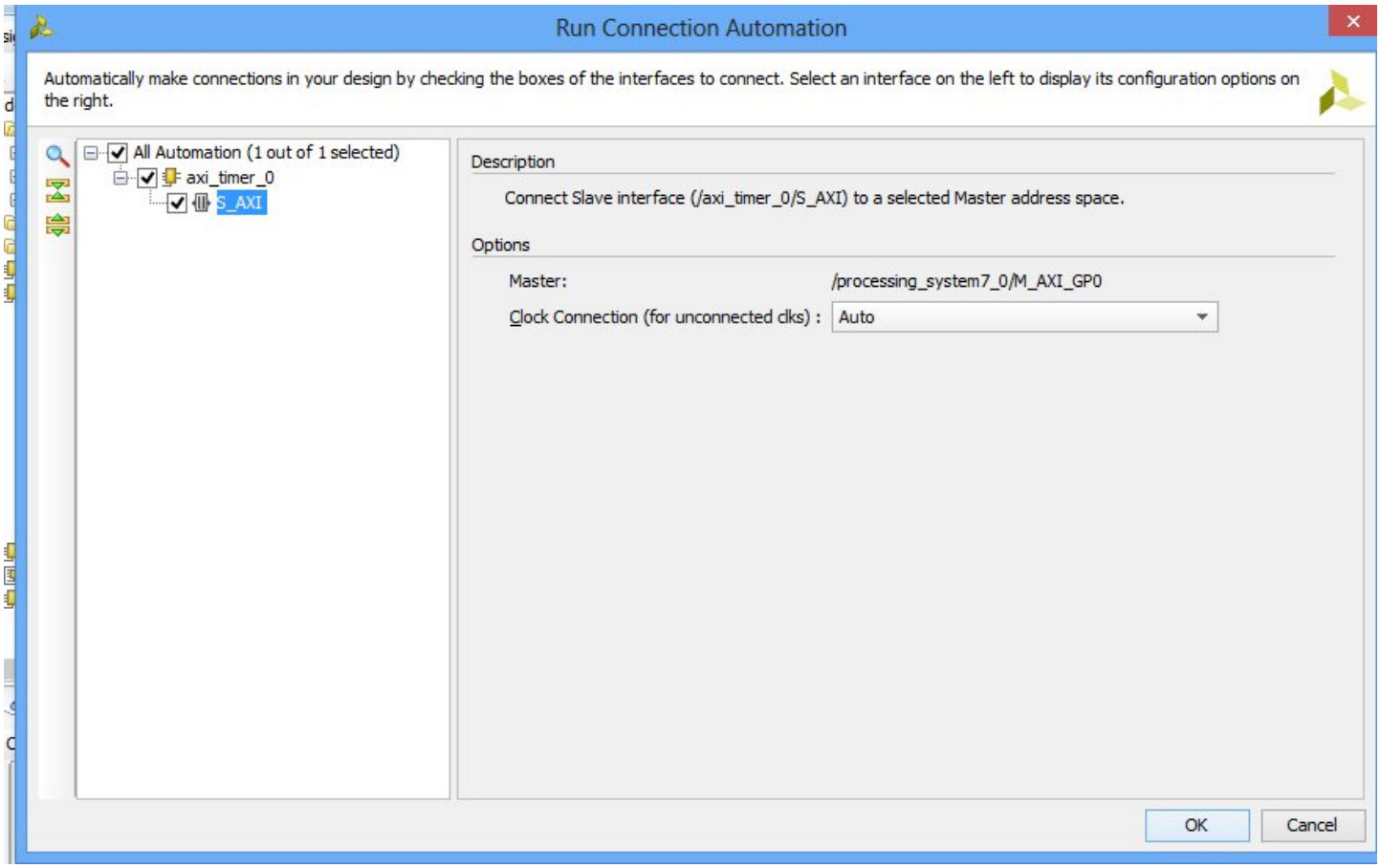


On additionne l'IPCORE Timer par la suite

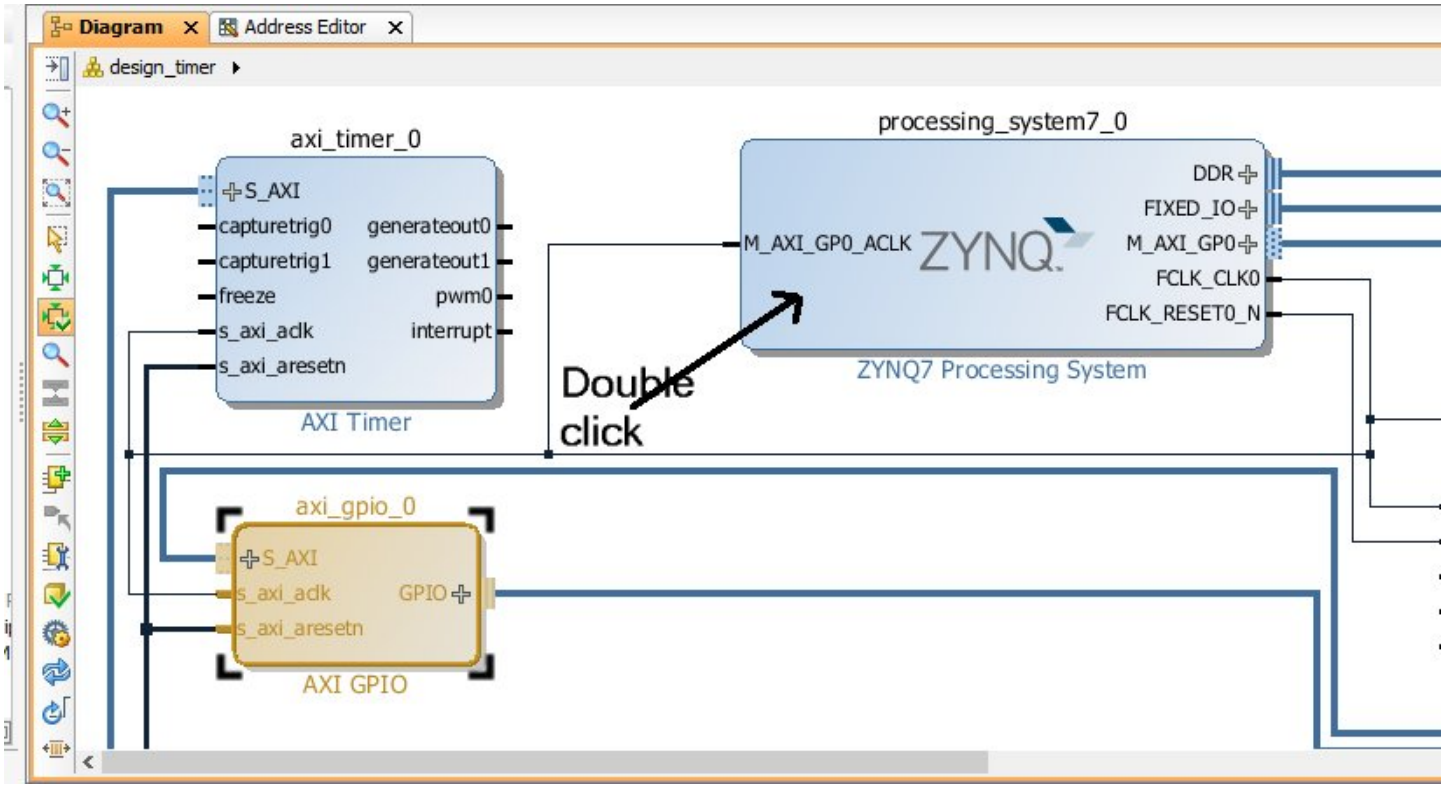


Mettre en place la connexion automatique de l'IPCORE AXI_Timer avec le PS

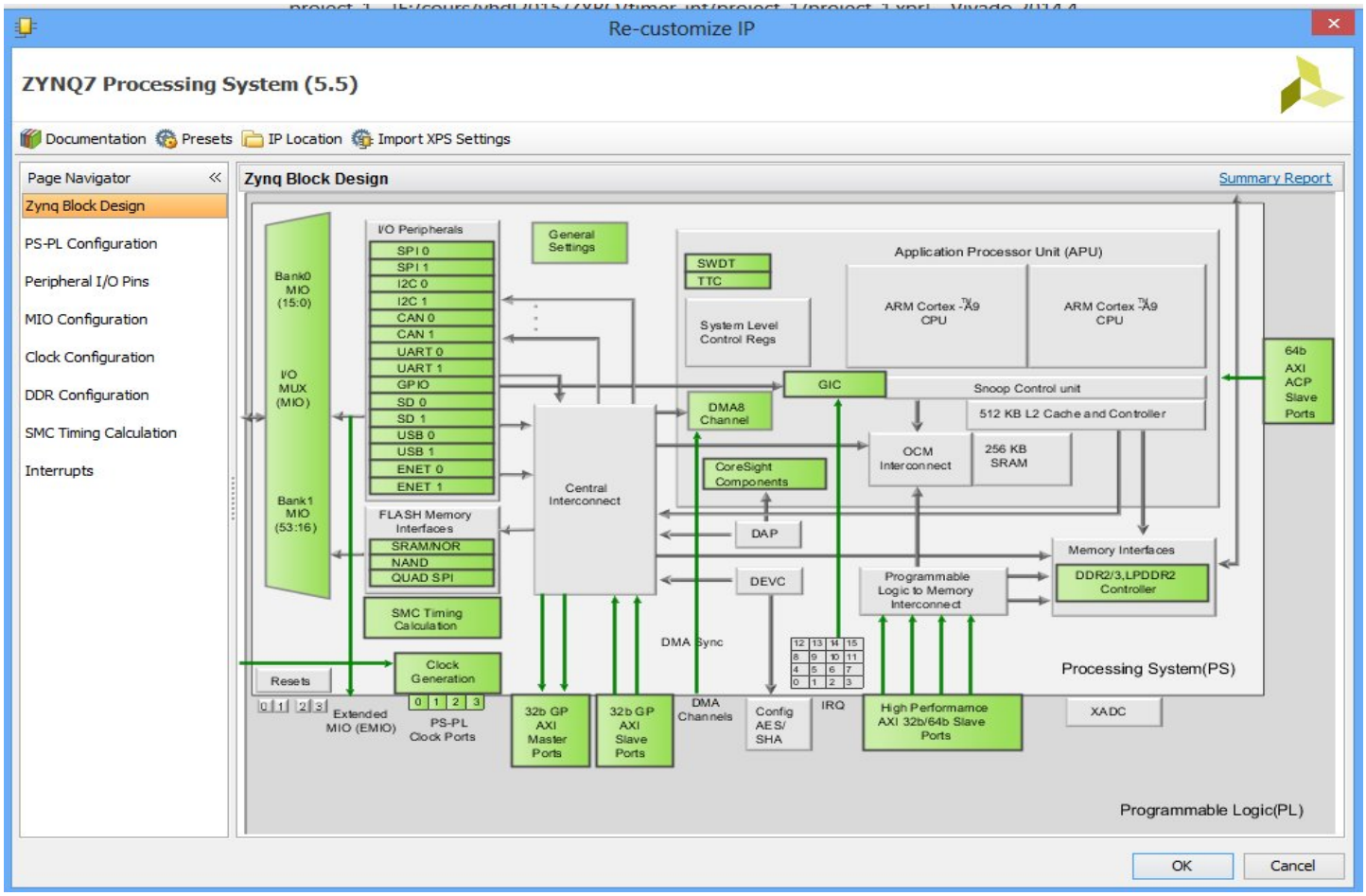




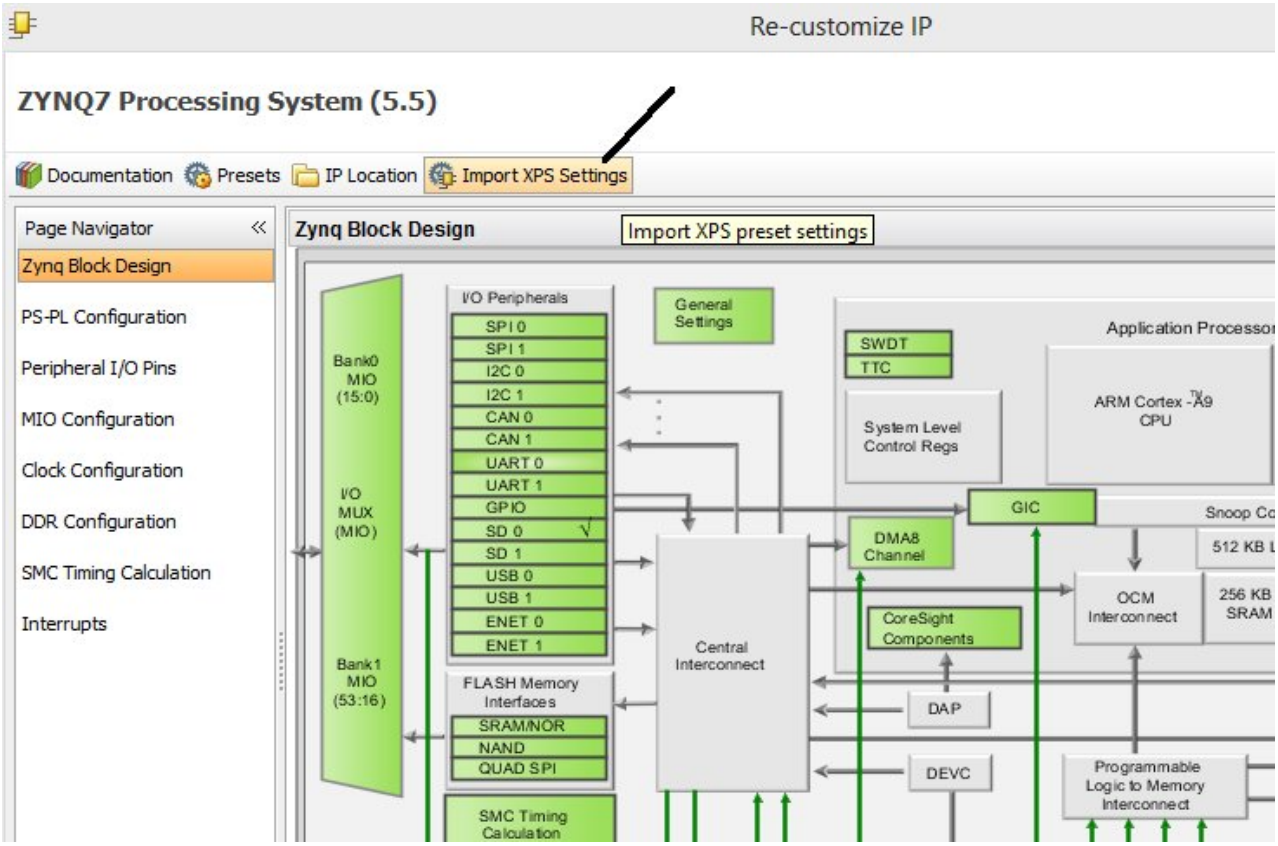
Double cliquez sur le système de traitement pour faire apparaitre ces propriétés



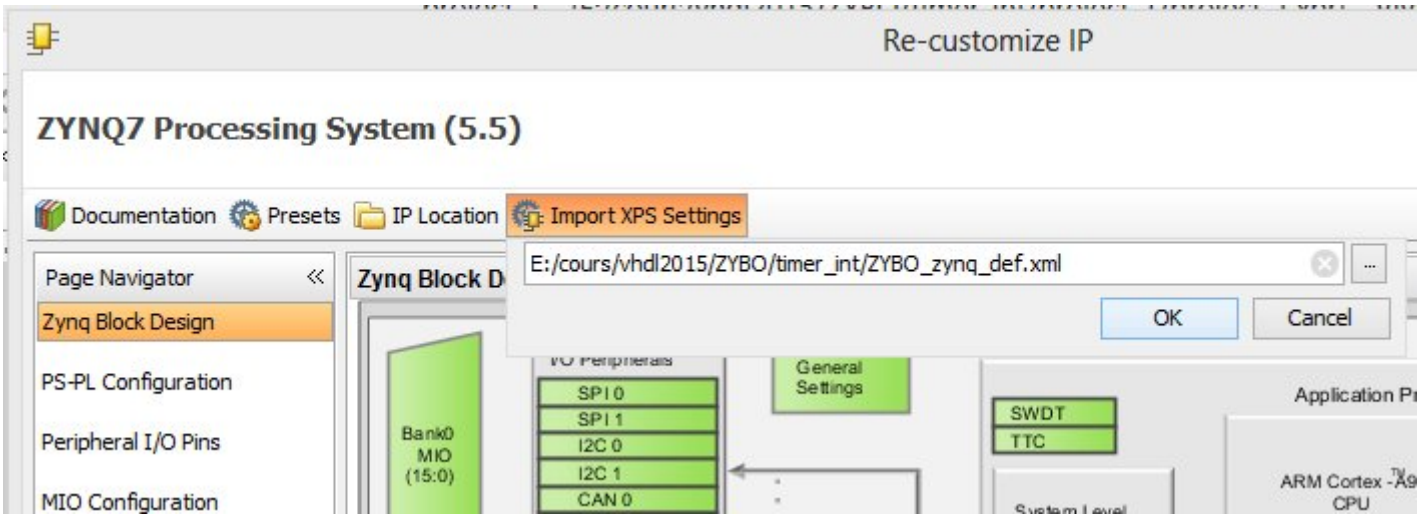
L'architecture interne du ZYNQ apparait :



La société digilent nous fournit **le fichier de configuration du PS** pour la carte ZYBO. Cette configuration doit être respectée car elle est liée à l'implantation matérielle des éléments sur la carte ZYBO

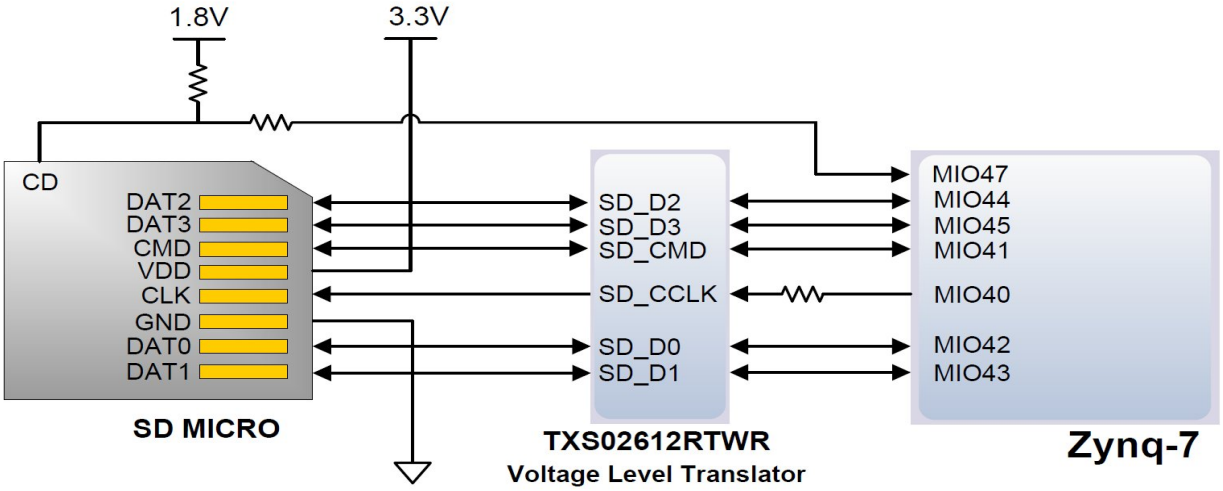


Donnez le chemin ou se situe ce fichier de configuration



La carte ZYBO dispose d'une SDCARD très utile pour avoir un système embarqué autonome. Cette SDCARD dispose d'un fichier BOOT.bin qui intègre un bootloader, le fichier bitstream de configuration de la logique programmable du FPGA et le fichier code machine de notre programme. Ce fichier code machine peut même incorporer un OS comme LINUX
Le slot Micro SD est connecté au ZYNQ grâce au port MIO du système de traitement PS. Le slot micro SD n'est pas connecté directement au ZYNQ mais par l'intermédiaire d'un translateur

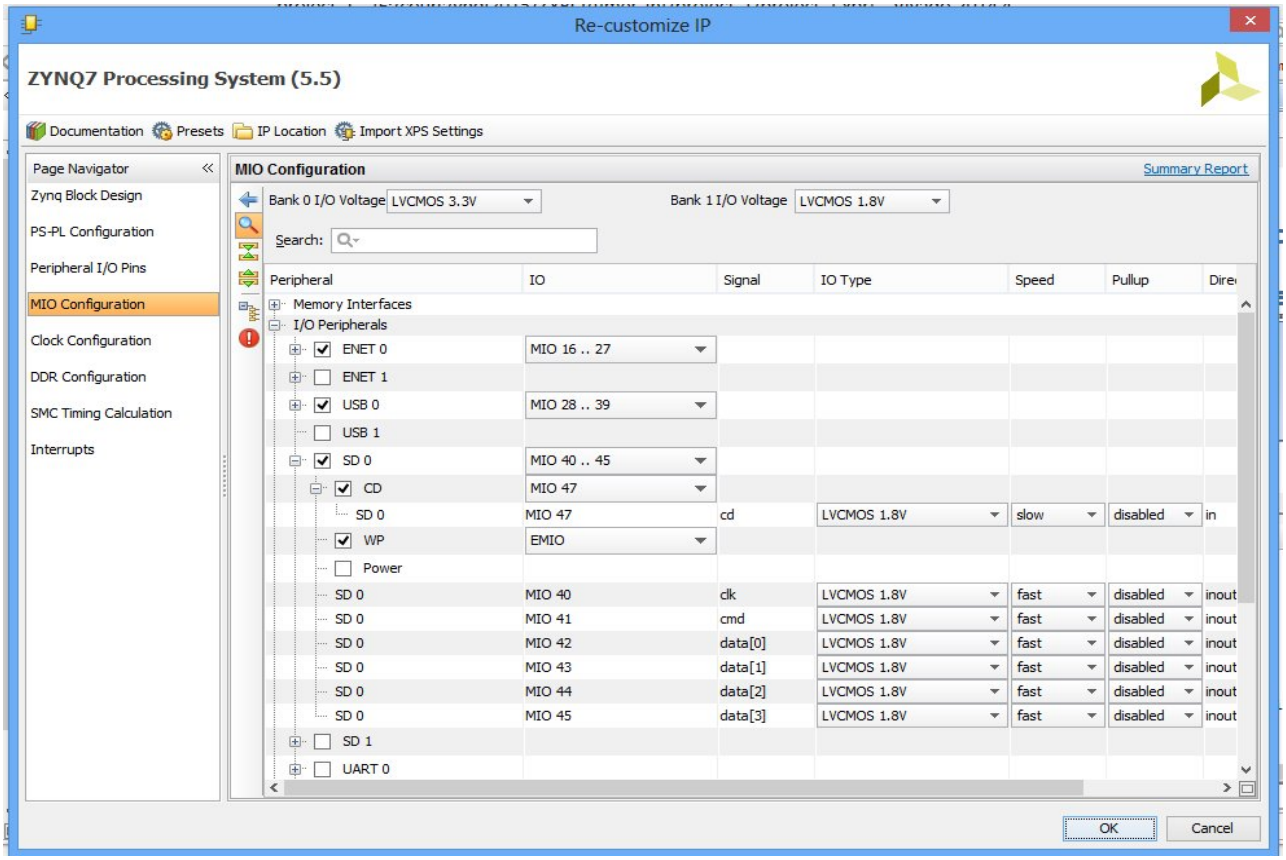
de niveau externe le TXS02612RTWR. Les broches MIO 40 à 47 font partie de la bank1 du FPGA et sont alimentés en 1.8V. La carte SD s'alimente en 3.3V le circuit TXS02612RTWR permet de faire la translation de niveau



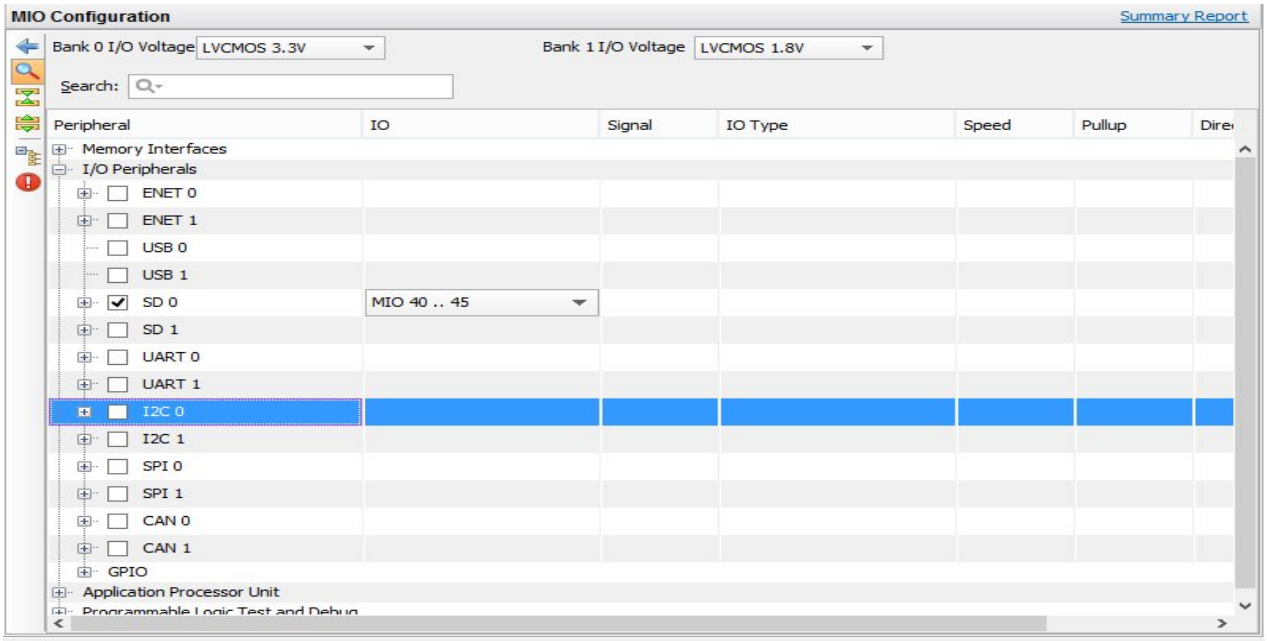
Le contrôleur de périphérique du ZYNQ-7 supporte un mode de transfert de 1 bit et 4 bits mais ne supporte pas le mode SPI. La connexion avec le slot micro-SD est précisée dans le tableau ci-dessous :

Signal Name	Description	Zynq Pin	SD Slot Pin
SD_D0	Data[0]	MIO42	7
SD_D1	Data[1]	MIO43	8
SD_D2	Data[2]	MIO44	1
SD_D3	Data[3]	MIO45	2
SD_CCLK	Clock	MIO40	5
SD_CMD	Command	MIO41	3
SD_CD	Card Detect	MIO47	9

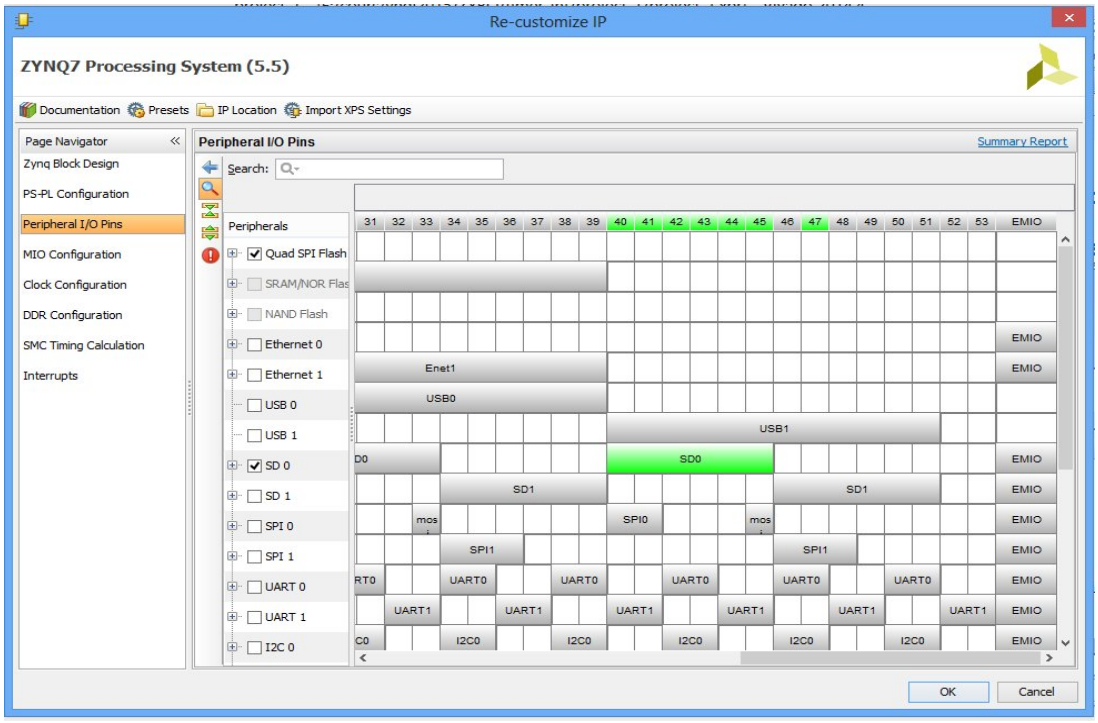
La configuration du PS et des périphériques se situe dans la rubrique configuration MIO. Le fichier de configuration .xml nous donne par défaut les périphériques qui sont sélectionnés.



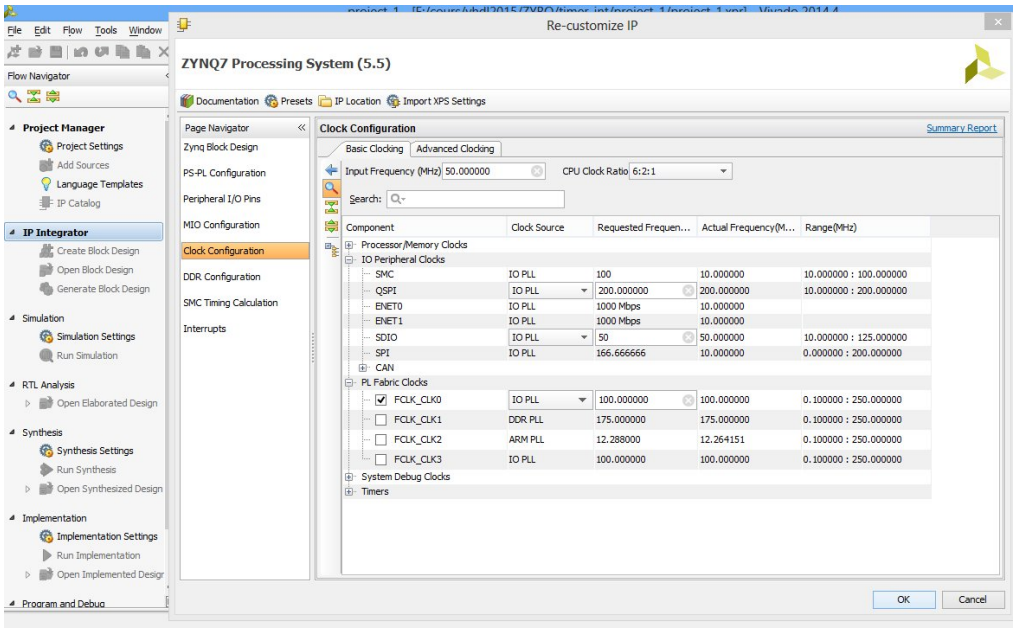
Certains périphériques du PS ne sont pas utiles pour notre application, on doit les décocher.



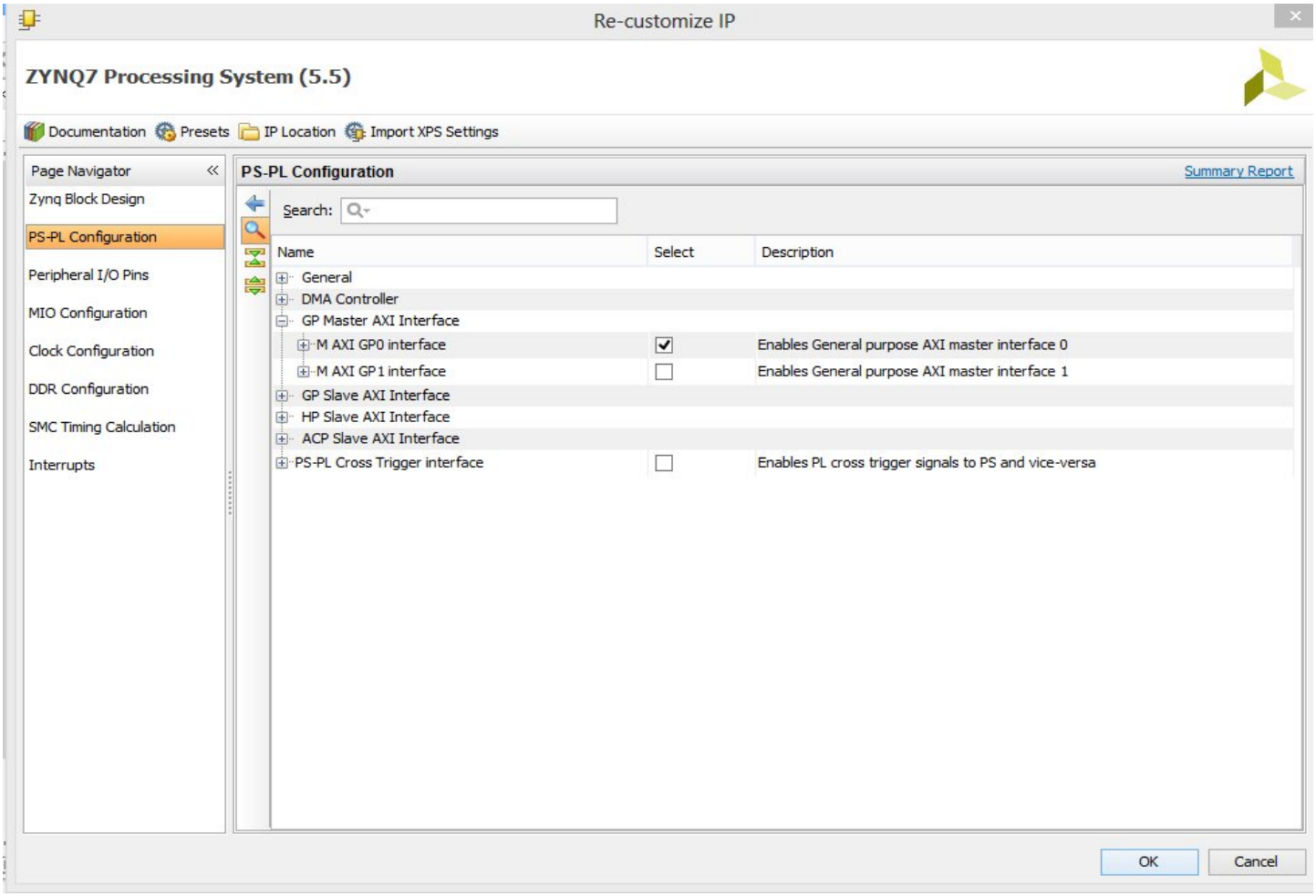
La rubrique peripheral I/O pins nous indique les ressources en entrées sorties qui sont monopolisées par les périphériques.



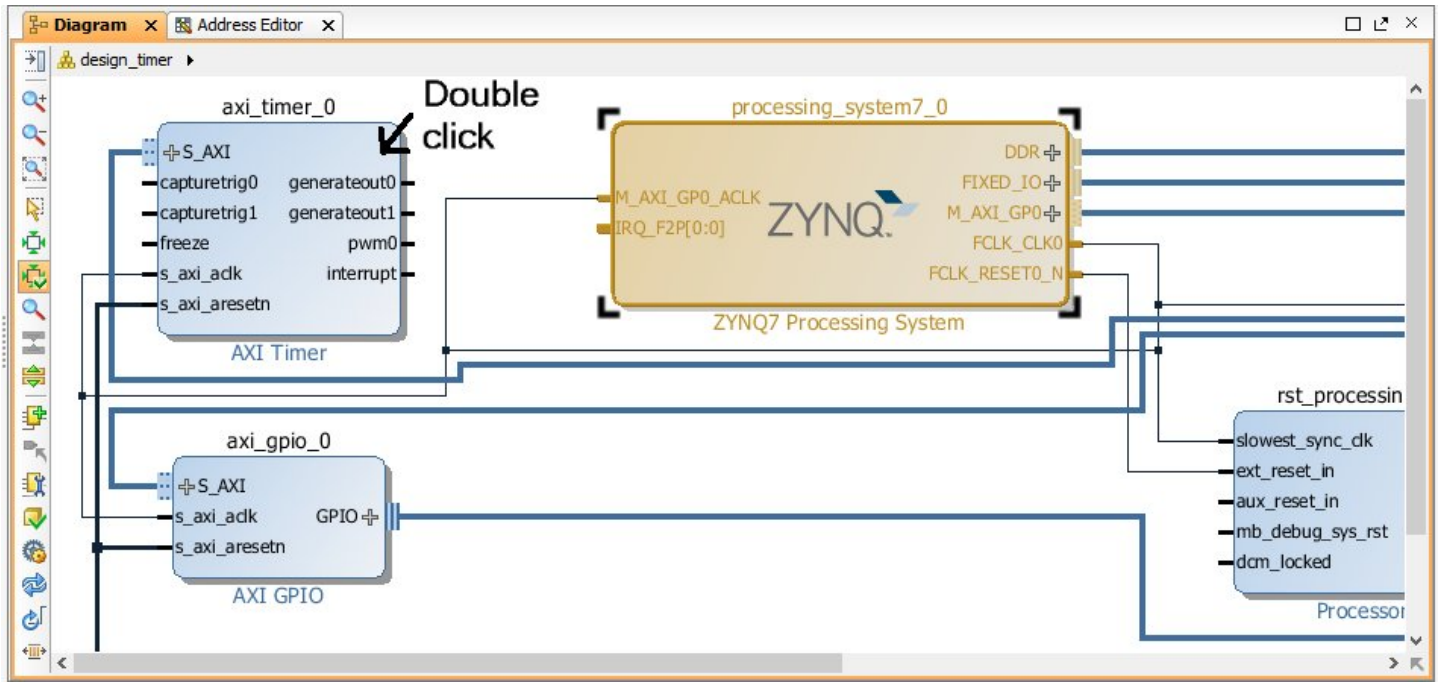
Notre timer générera une interruption à chaque fin de cycle de comptage. On doit configurer pour le PS le type d'interruption qu'il doit gérer. On prendra une interruption du type FIQ.
Pour que le PS puisse piloter les IPCORE ou les périphériques dans la logique programmable PL par le bus AXI, le PS doit être configuré.
Le contrôleur de bus AXI et une horloge doivent être sélectionnés.



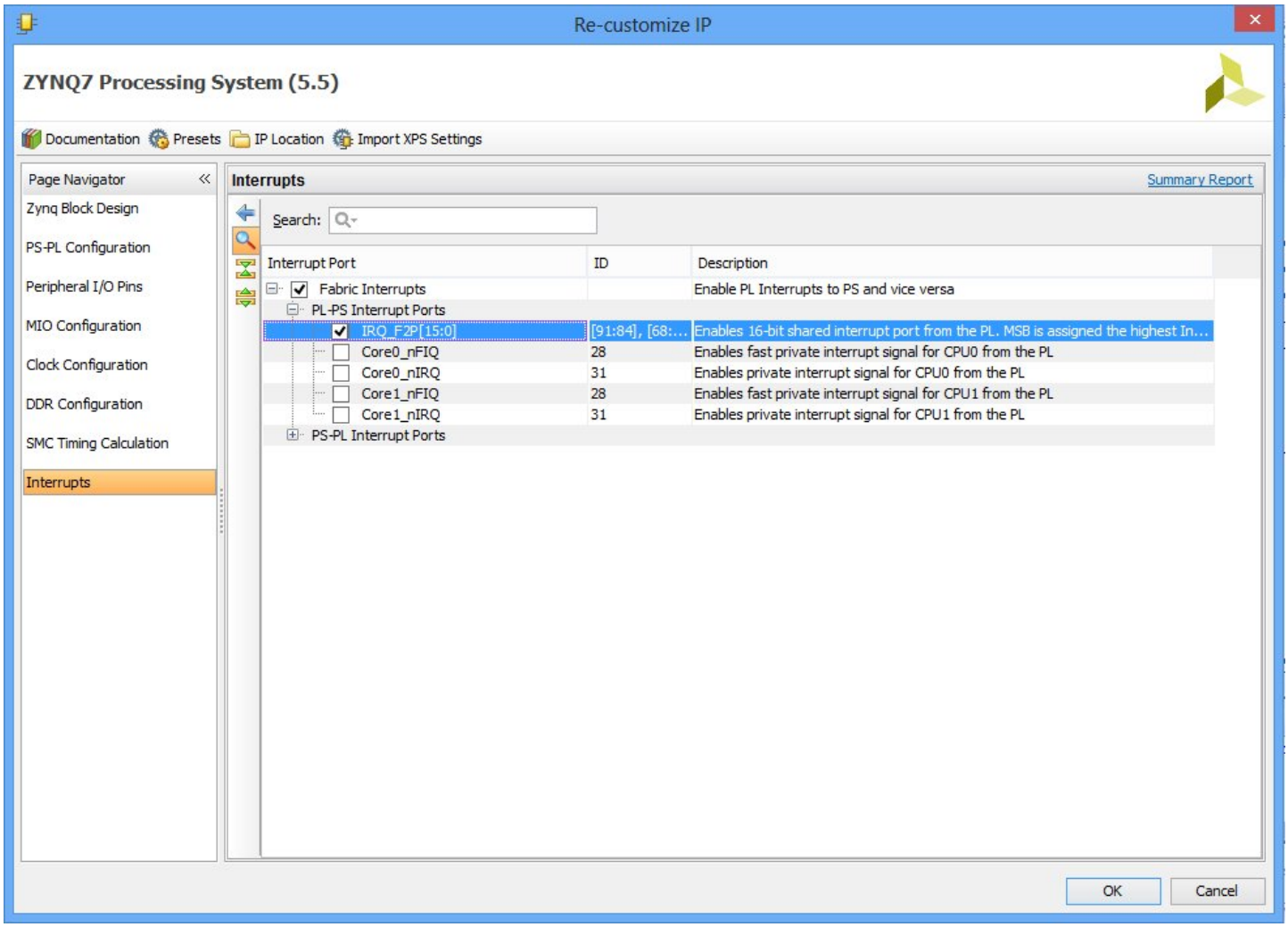
Sélectionnez l'interface qui nous permettra de connecter un contrôleur maitre bus AXI



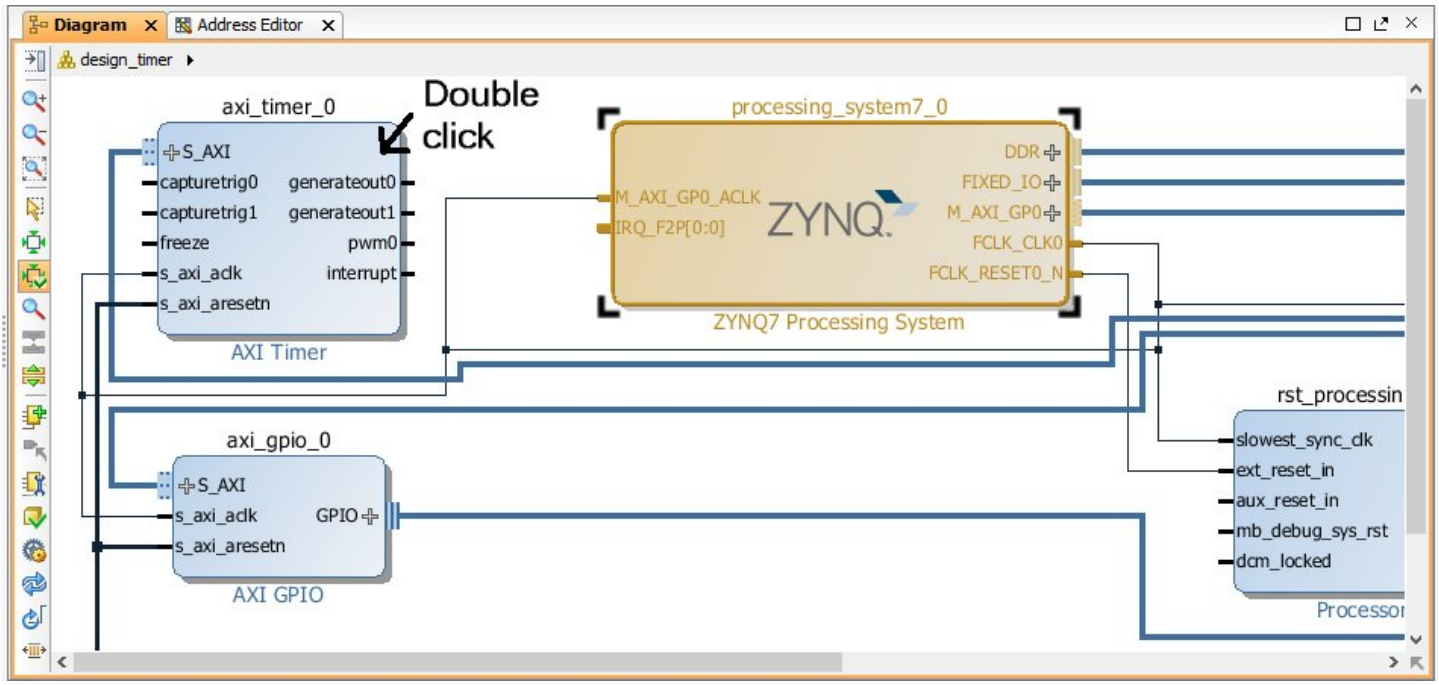
Le bus AXI est bien connecté à nos périphériques



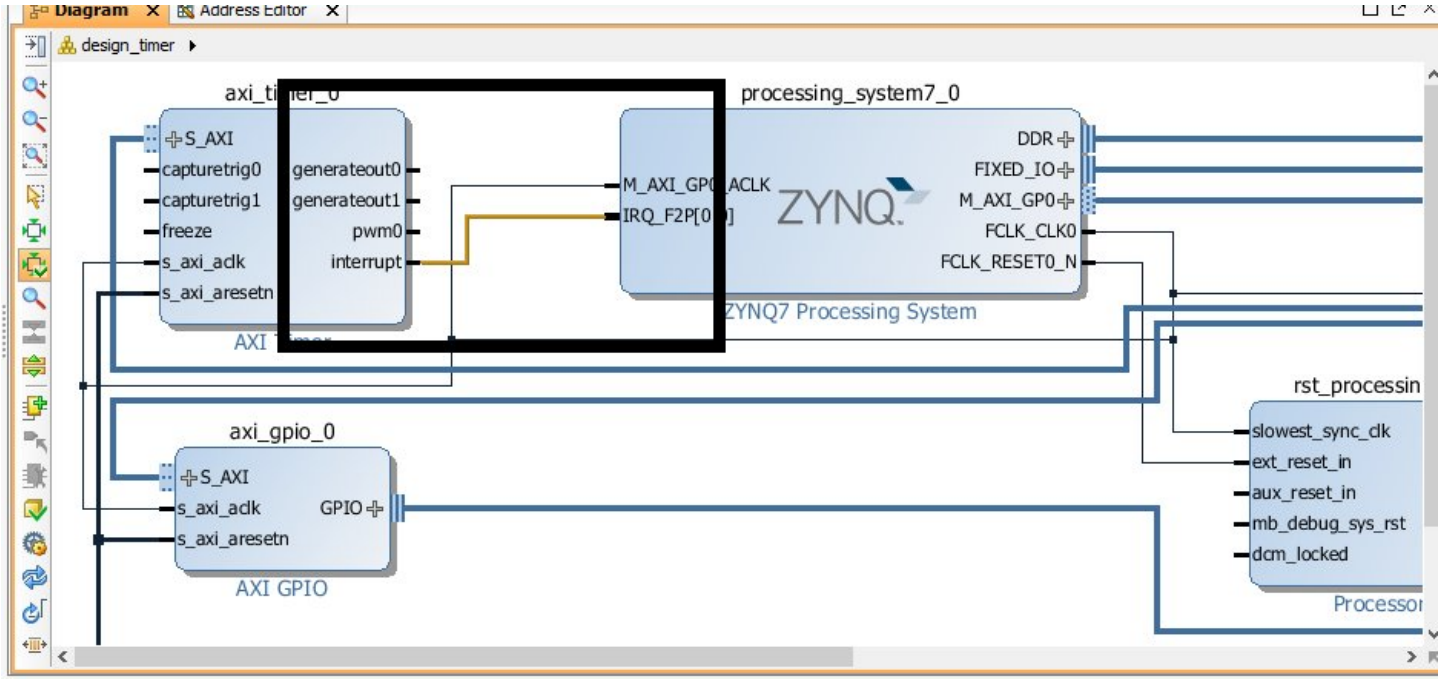
Les interruptions sont configurées dans ce menu



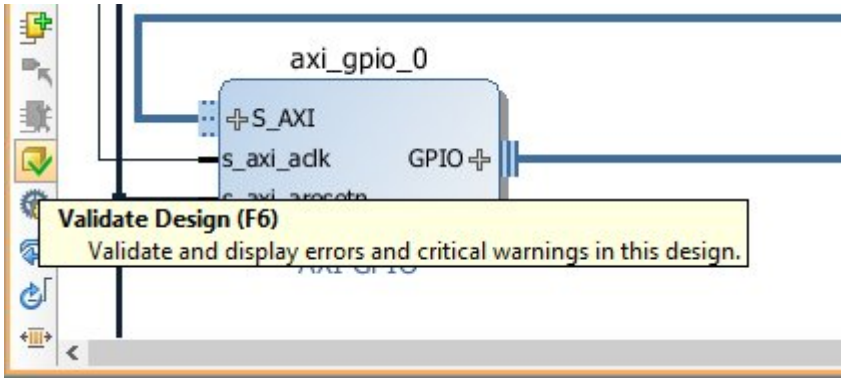
Pour le timer il faut aussi valider les interruptions



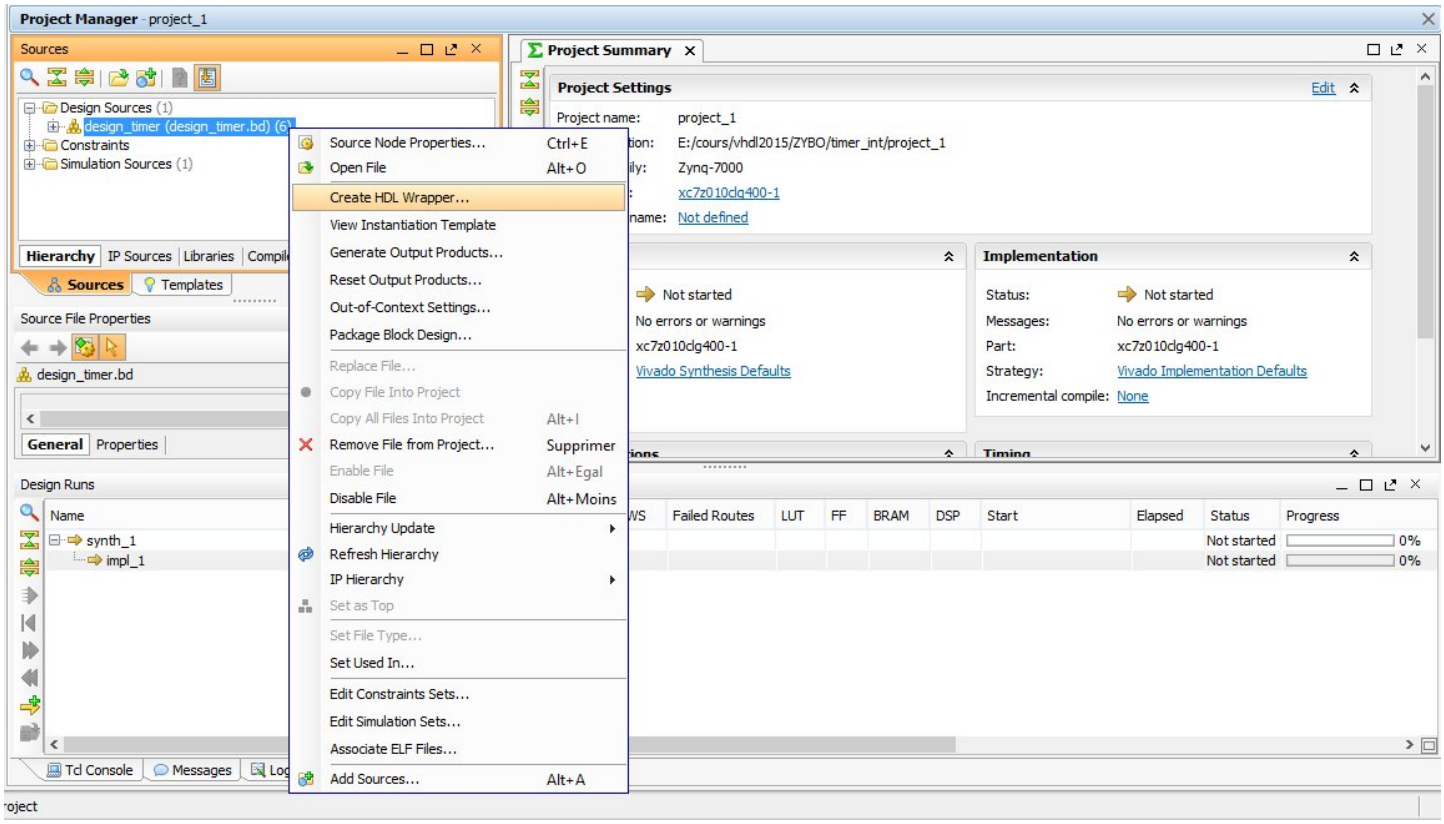
Pour le timer on doit connecter manuellement avec l'outil fil la sortie interrupt sur l'entrée IRQ



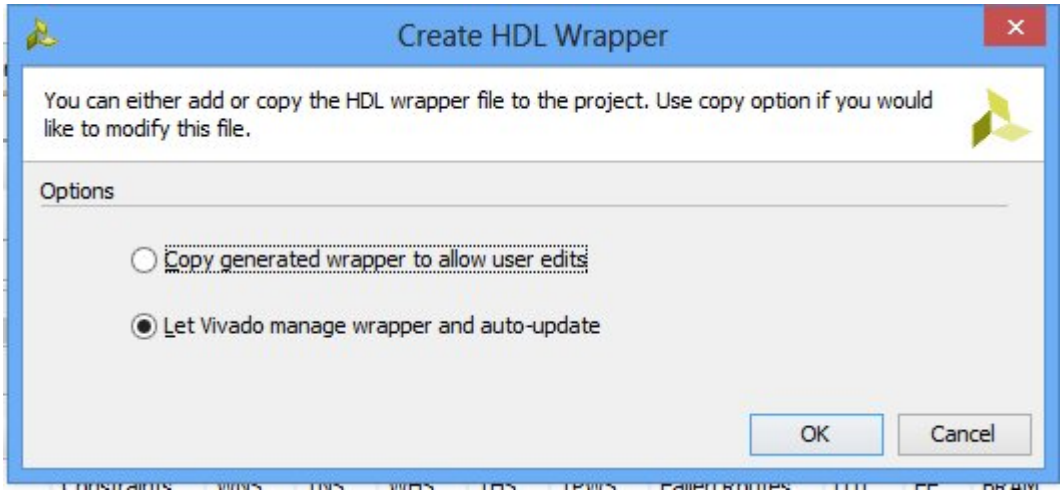
Pour vérifier la conformité des règles de dessin cliquez sur validate design



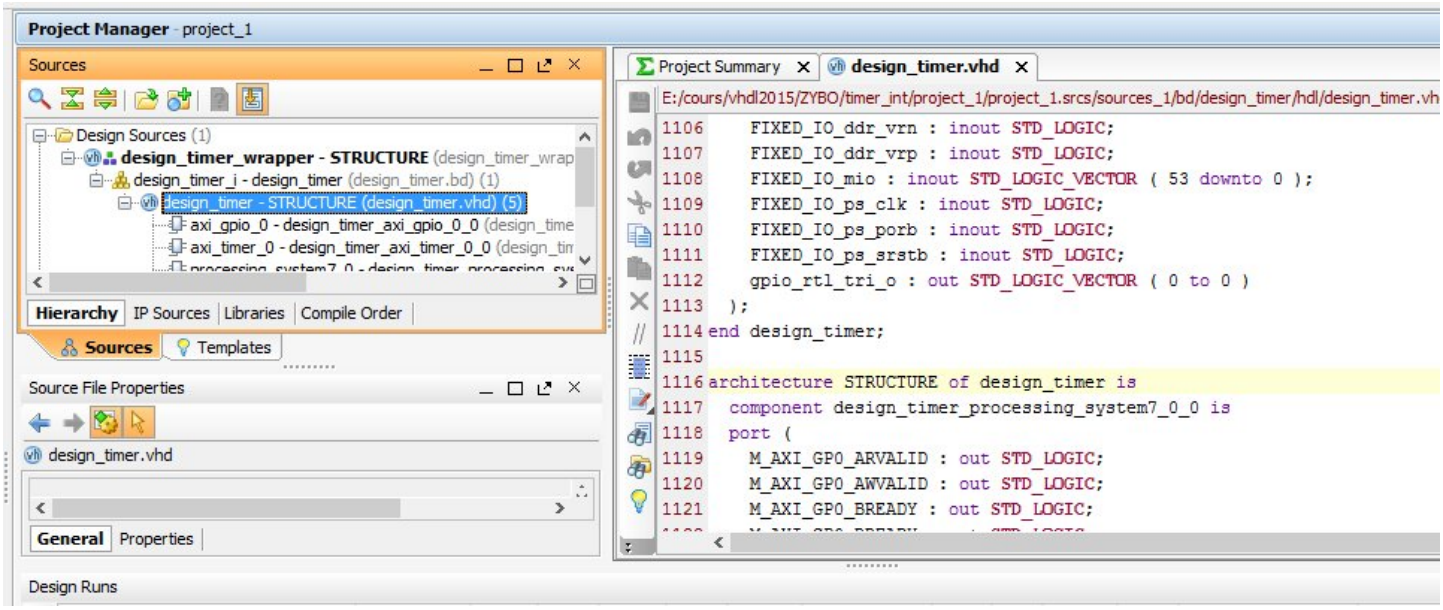
On doit créer un fichier VHDL de top level pour notre bloc design, cliquez droit sur le fichier design_timer.bd et lancez create HDL WRAPPER



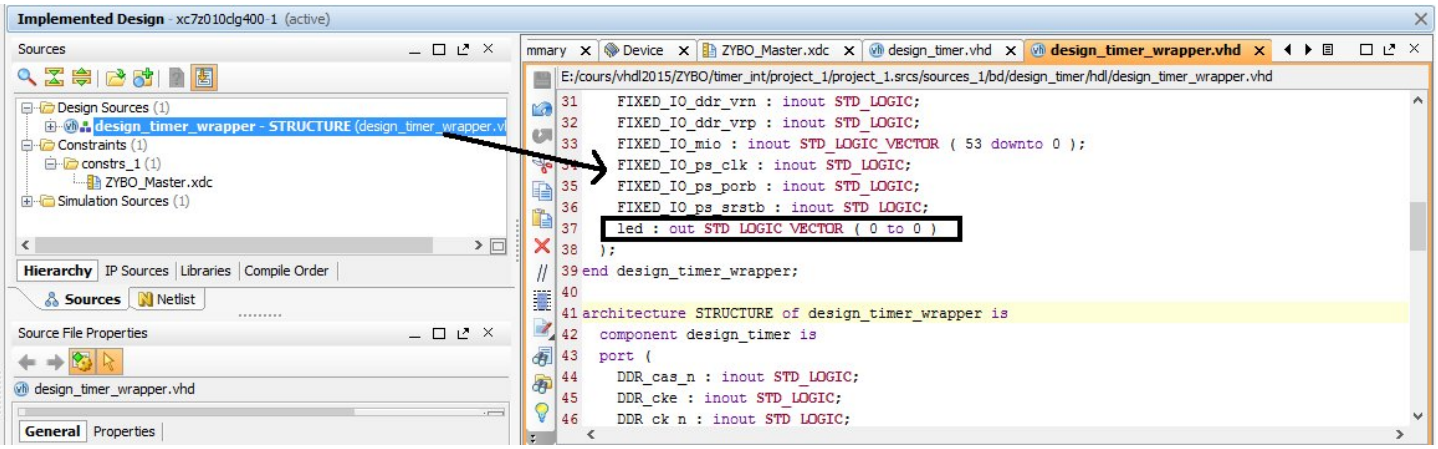
Laisser VIVADO gérer la création du fichier VHDL

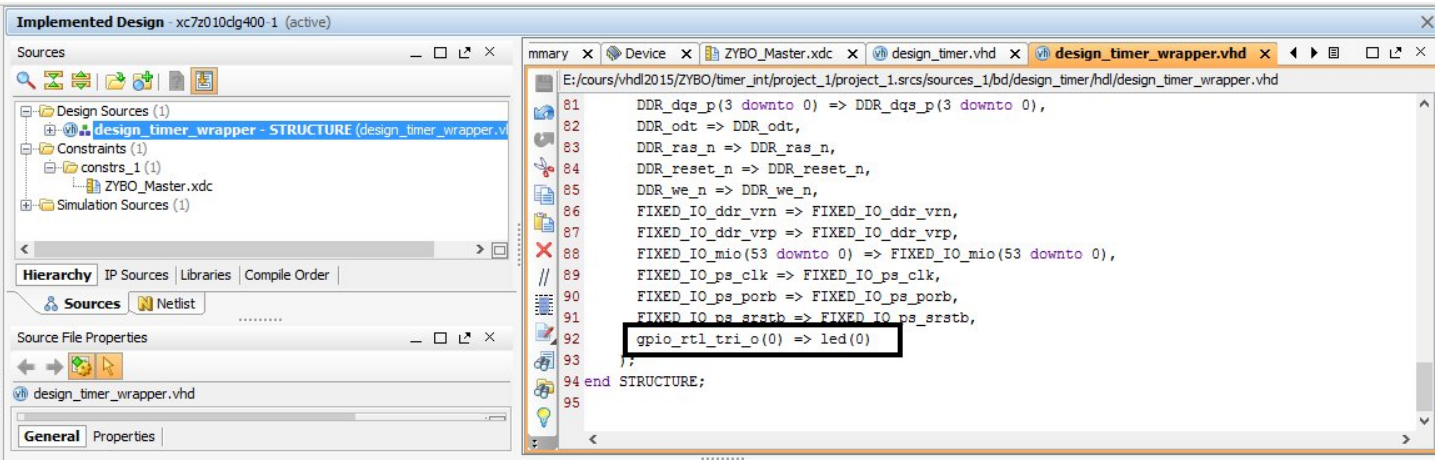


Deux fichiers VHDL sont créés un qui caractérise la connexion, la déclaration des différents signaux des éléments du bloc design, L'autre de plus haut niveau qui caractérise le composant Bloc design

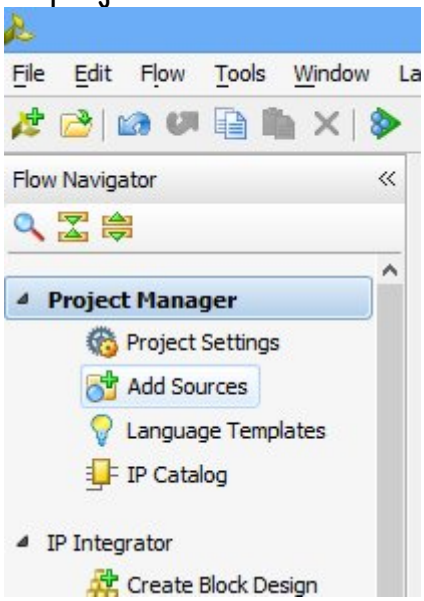


Remplacez dans le fichier VHDL de top level, la déclaration du signal gpio_rtl_tri_o par le signal LED de la façon suivante :

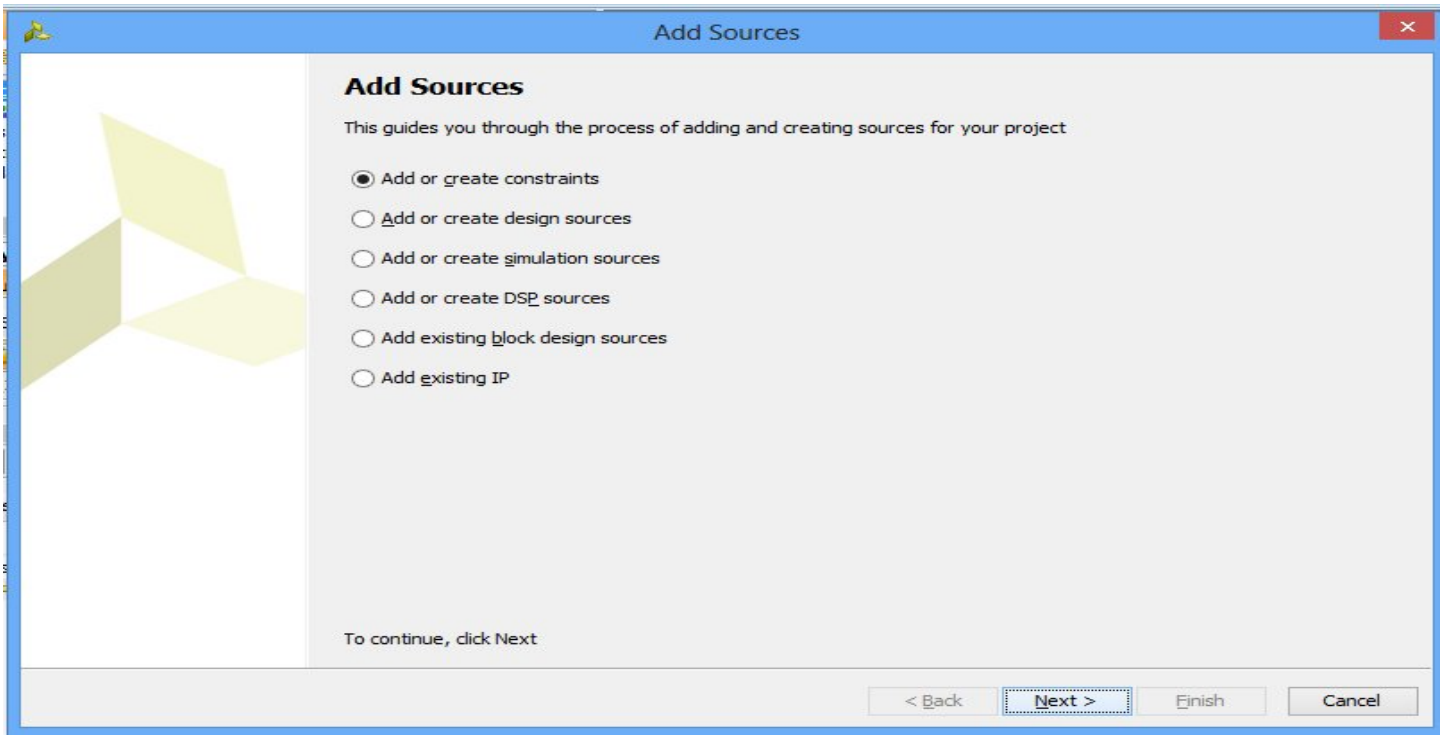




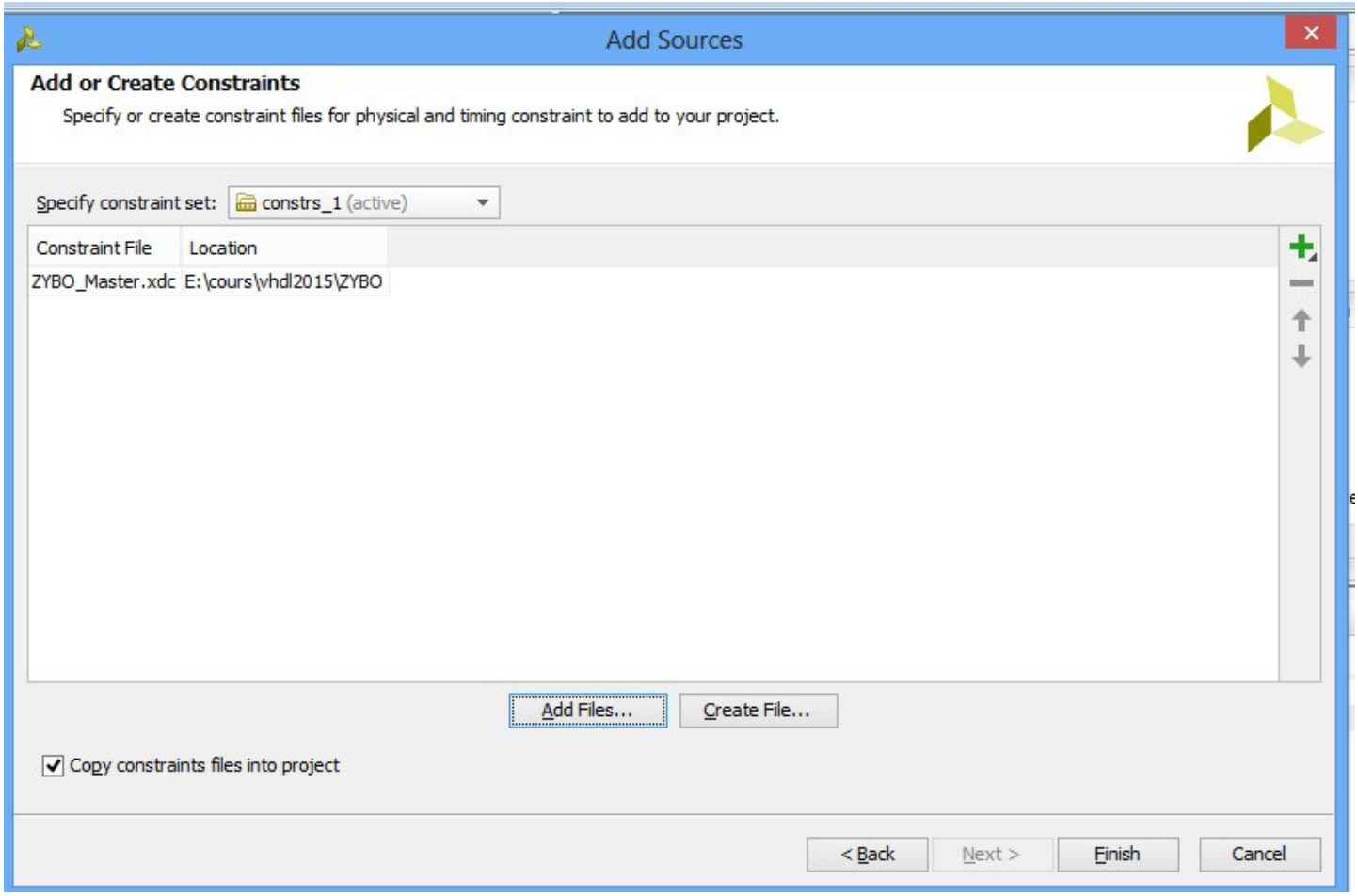
Additionnez le fichier de contrainte pour la carte zybo si cela n'a pas été fait lors de la création du projet



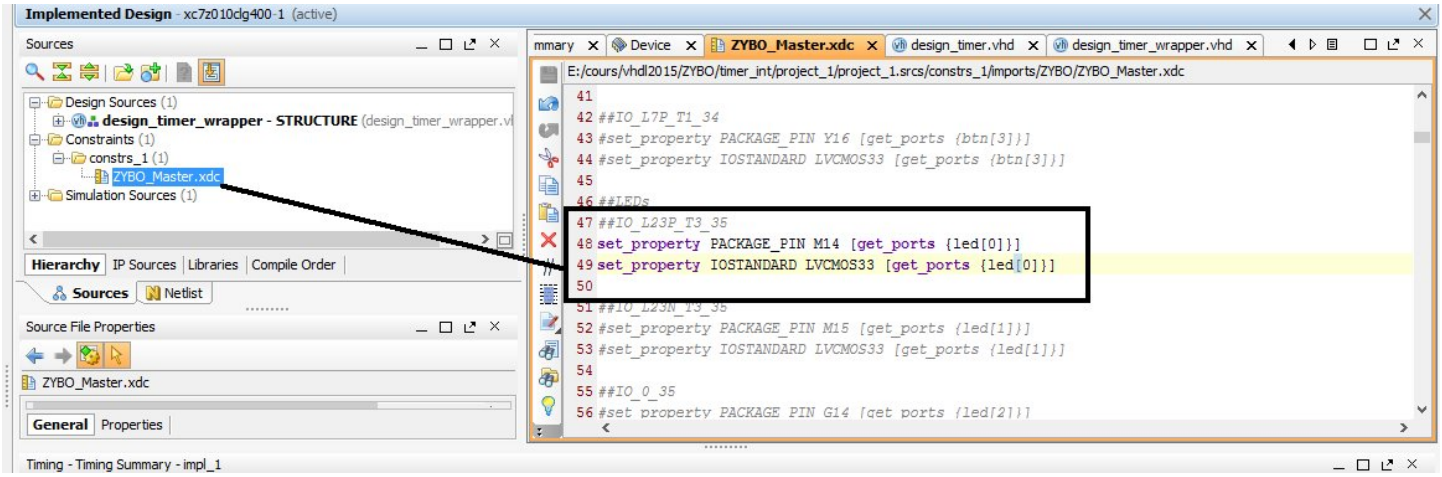
Choisir un fichier de type contrainte

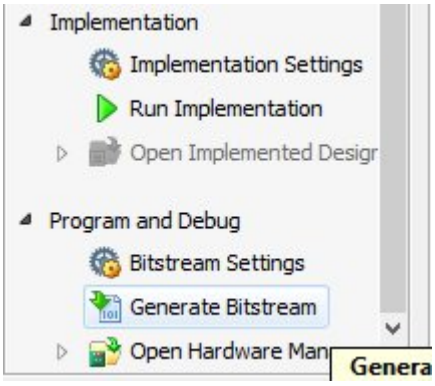


Fournir son chemin

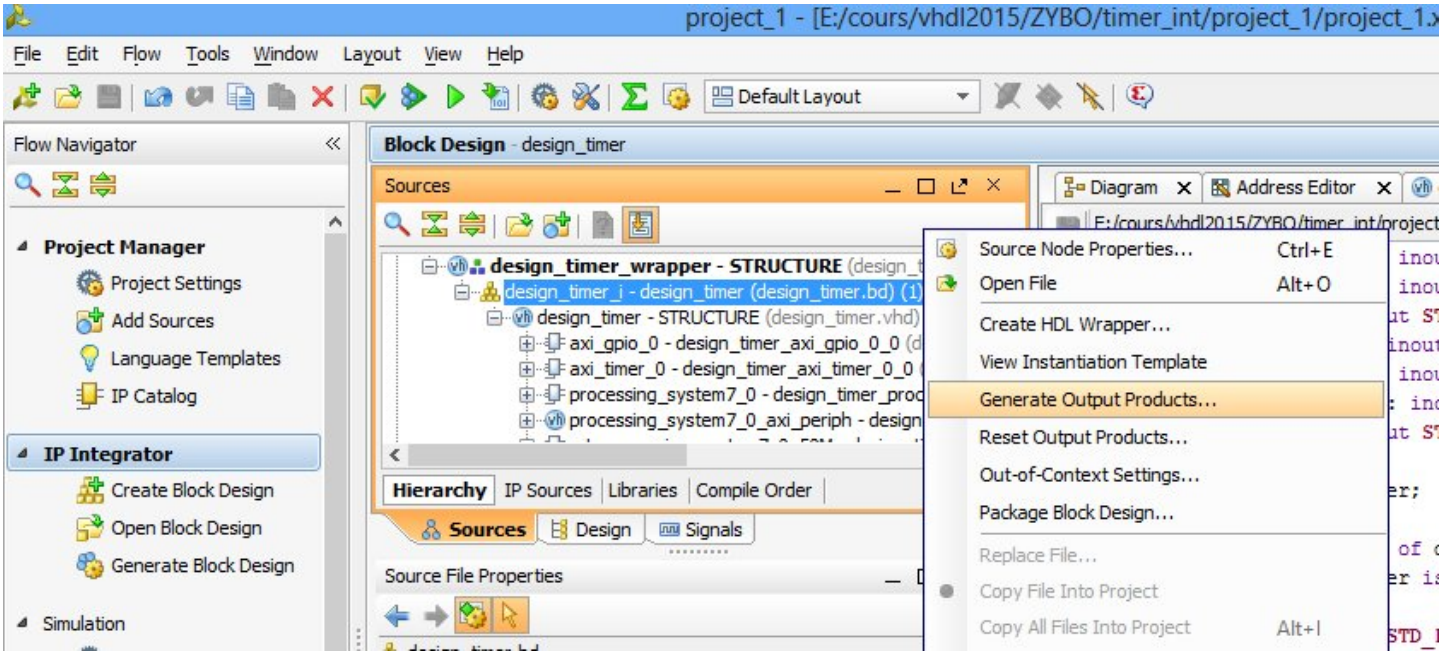


Validez la sortie de gisement M14 afin de pourvoir piloter la LED

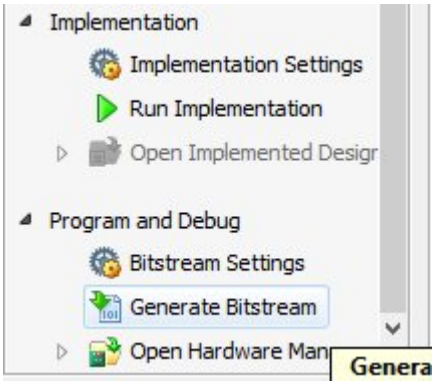




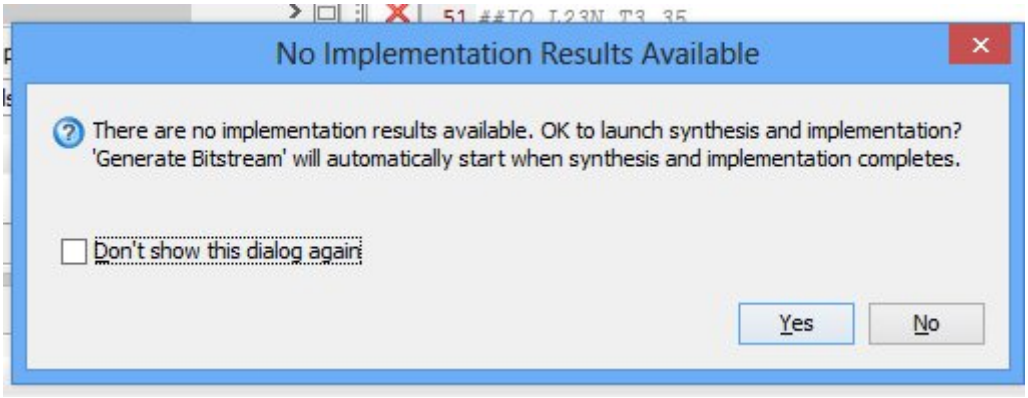
On peut générer par la suite pour le logiciel SDK, les éléments nécessaires
Pour la configuration du système de traitement PS



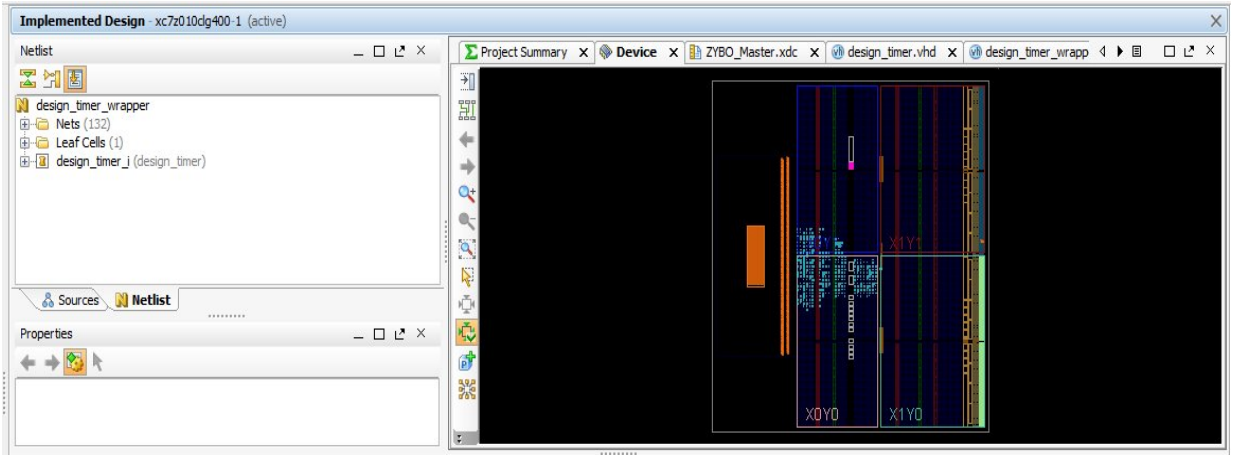
On poursuit par la création du bistream permettant la configuration de la logique programmable PL (AXI_timer et AXI_GPIO)



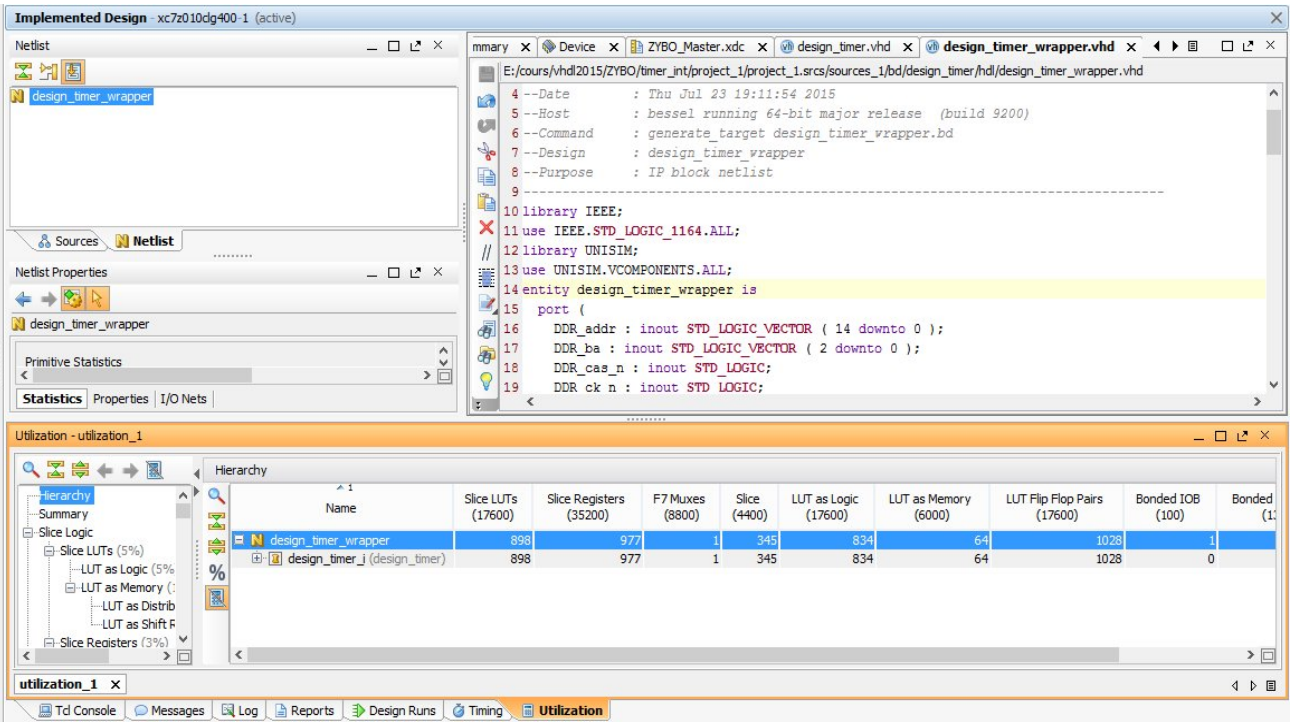
Vivado demande d'abord de synthétiser et d'implémenter le FPGA, répondre par yes



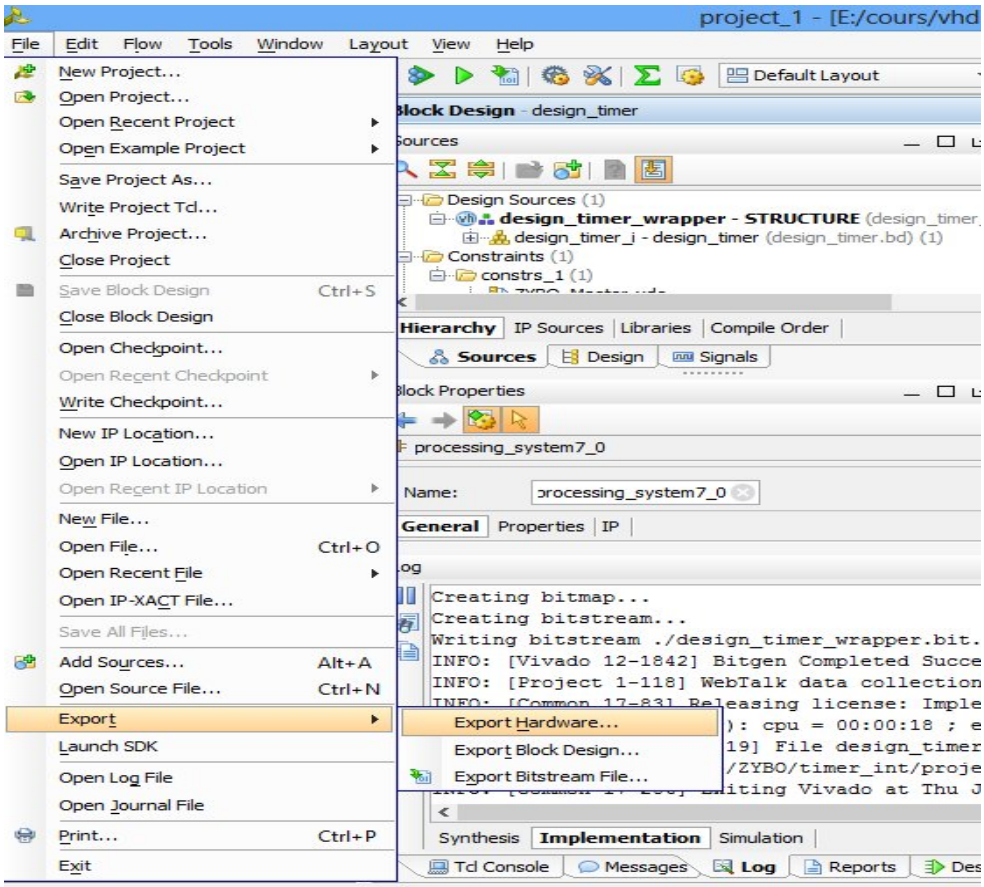
A la fin de la phase de synthèse on peut observer le résultat de cette implémentation



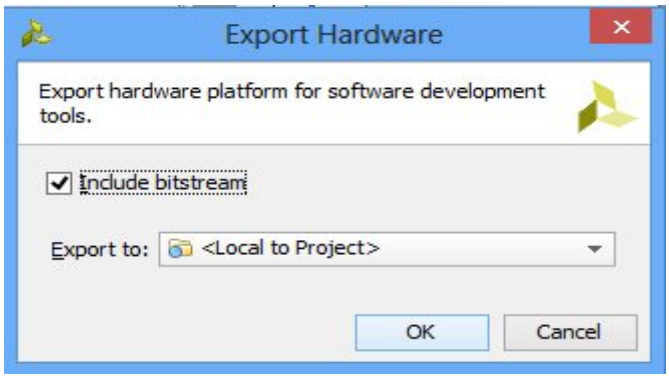
En cliquant sur hierarchy on observe les ressources matérielles monopolisées



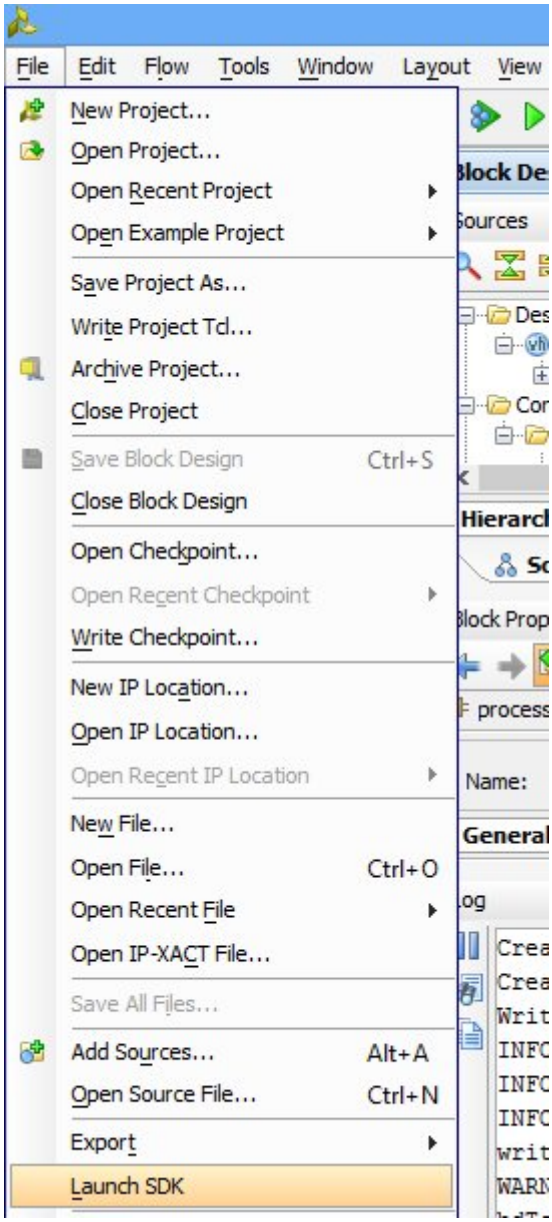
Nous reste à exporter le matériel afin de réaliser le board support Package pour la partie programmation et fournir le bitstream



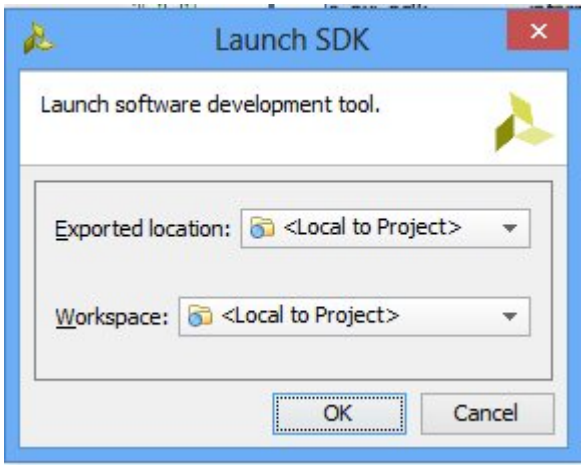
Le logiciel nous demande si on veut aussi exporter le bistream (cochez la case)



Le board design est ainsi réalisée on peut lancer le logiciel de développement logiciel pour le PS et ces processeurs ARM

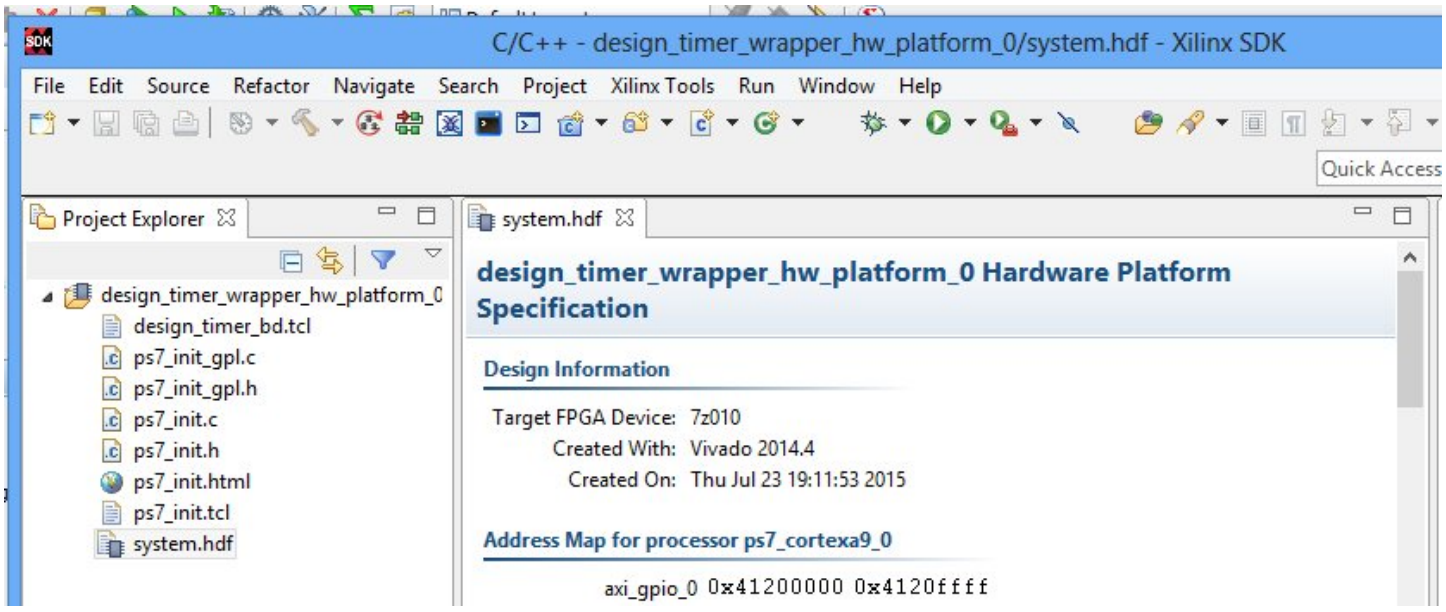


On vous demande si les éléments développés par la suite doivent rester locales à ce projet répondre oui

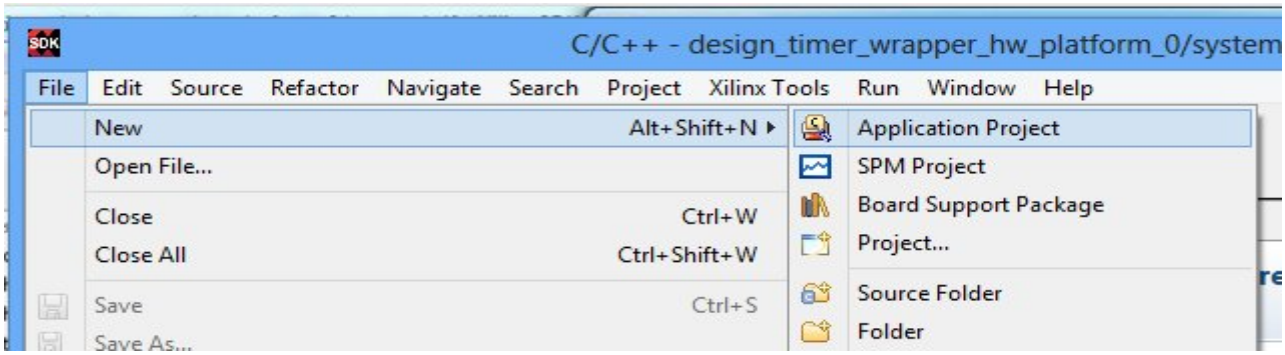


3) Développement logiciel pour la partie traitement PS

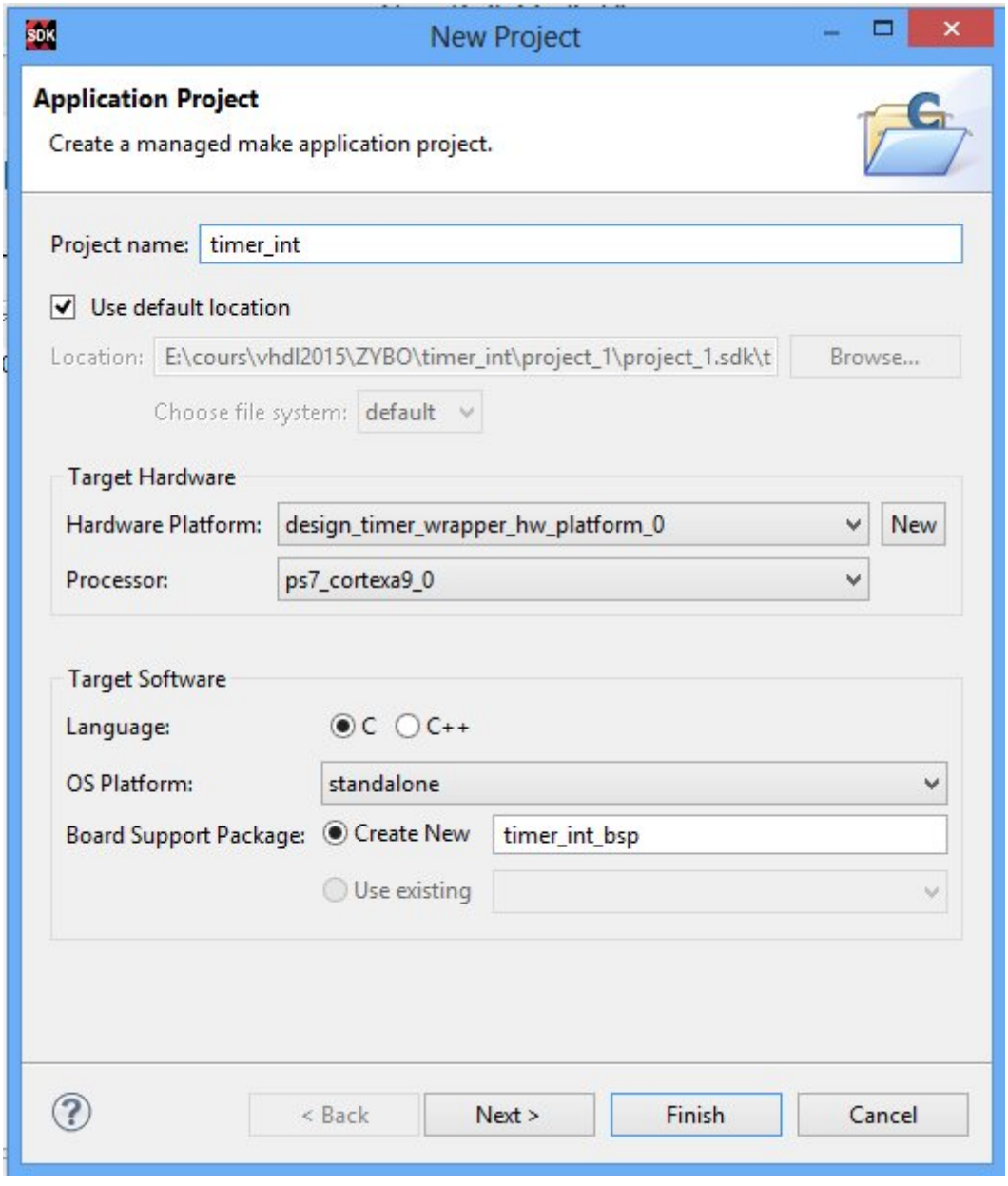
Le logiciel SDK apparait avec la configuration matérielle



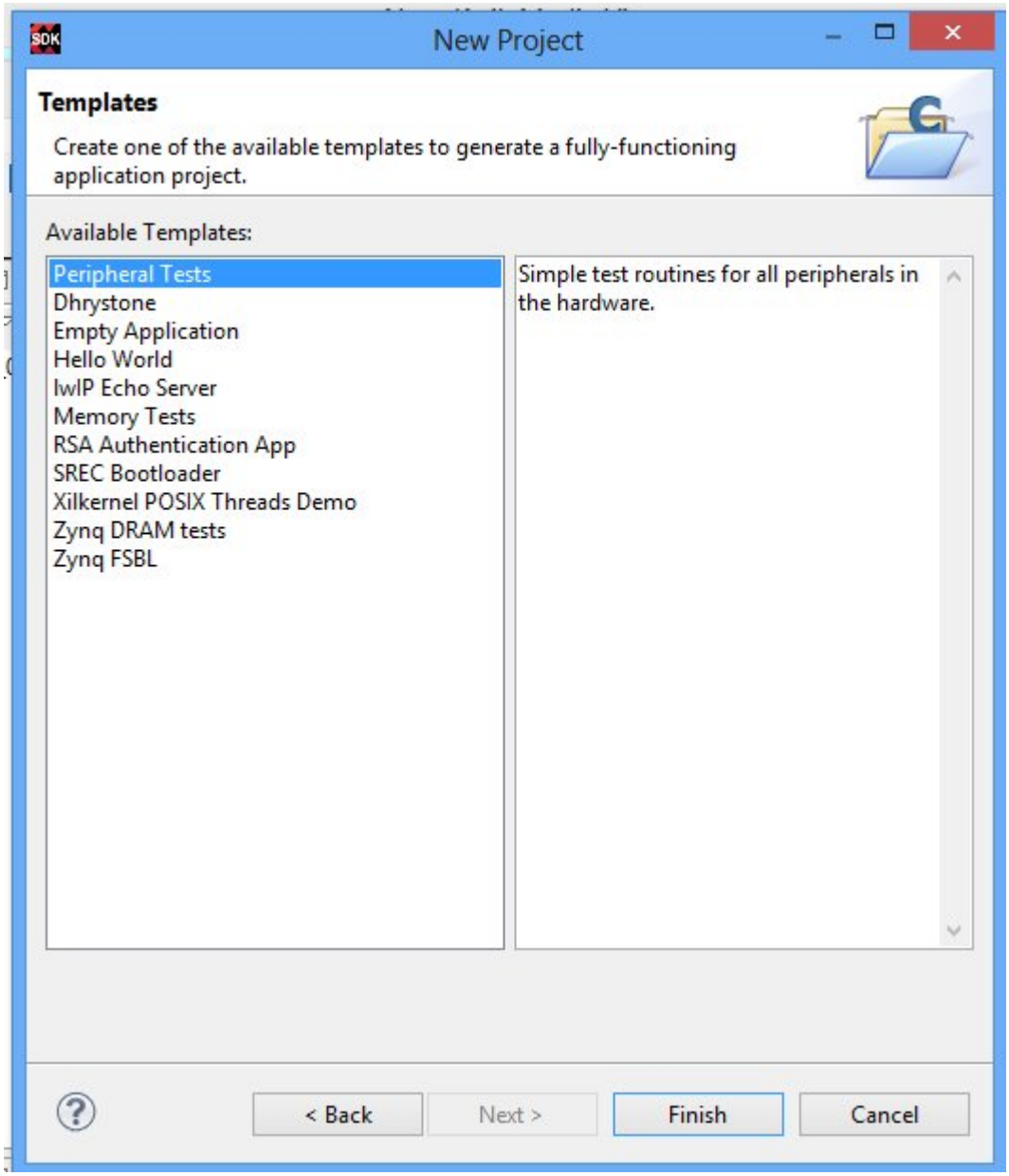
Créer un nouveau projet application



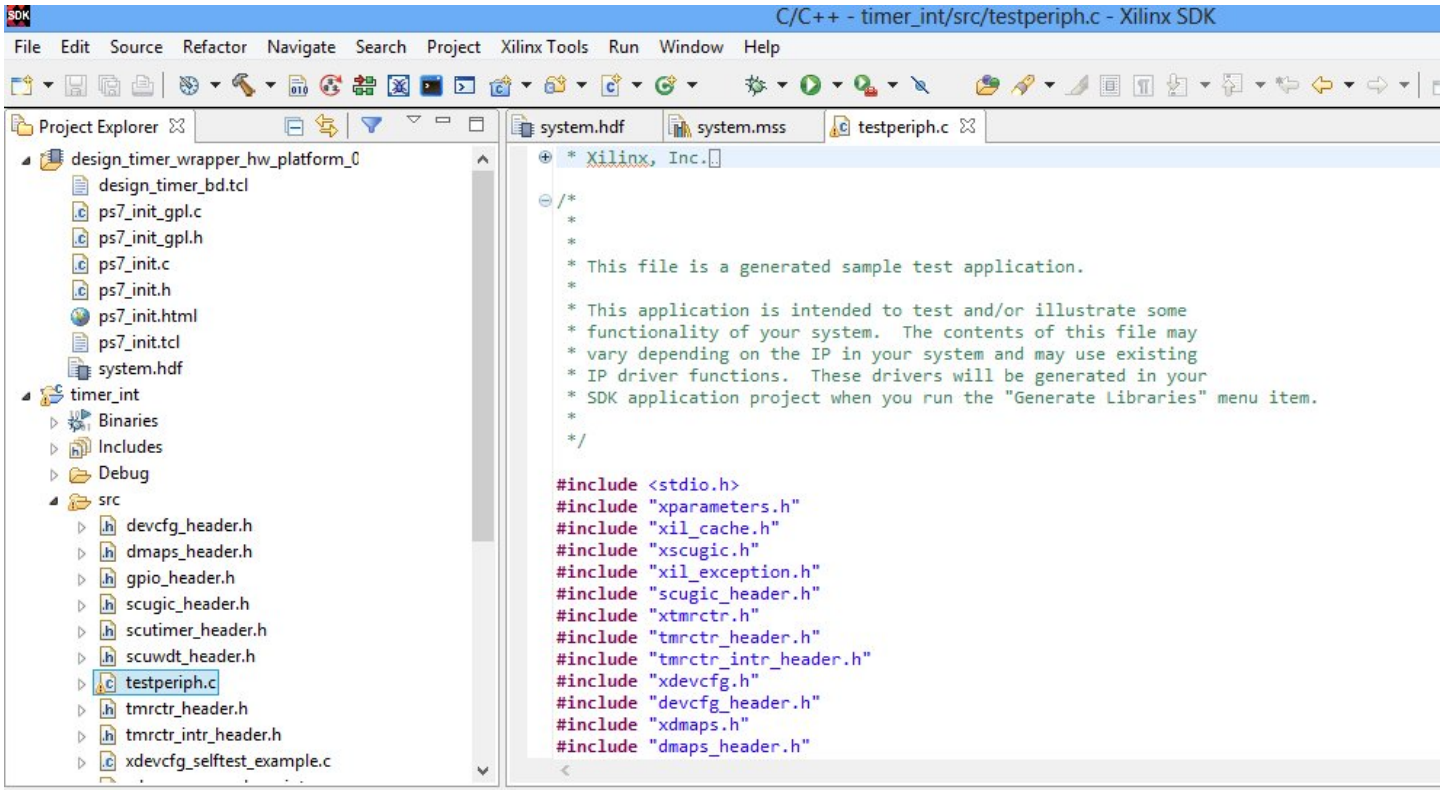
Donnez un nom à ce nouveau projet



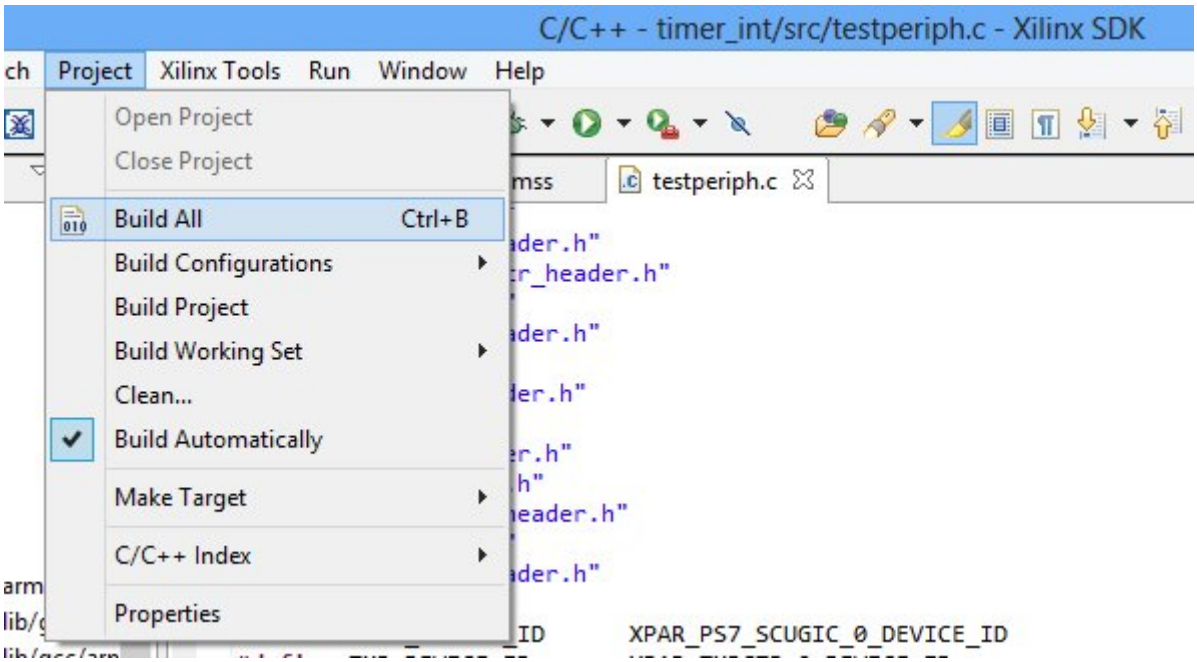
Choisir une application du type peripheral test



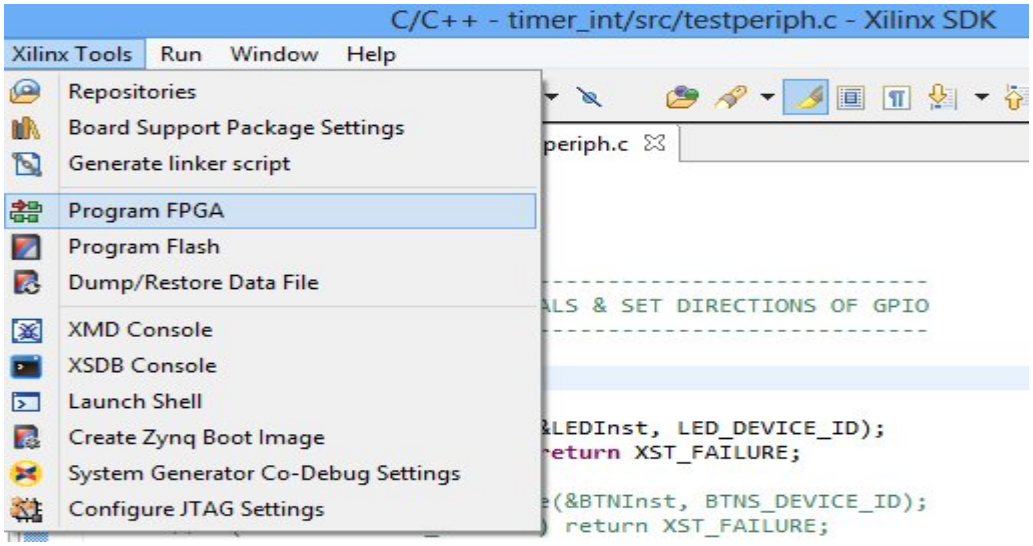
Modifiez le fichier source à votre convenance de façon à ce qu'il réponde à l'application



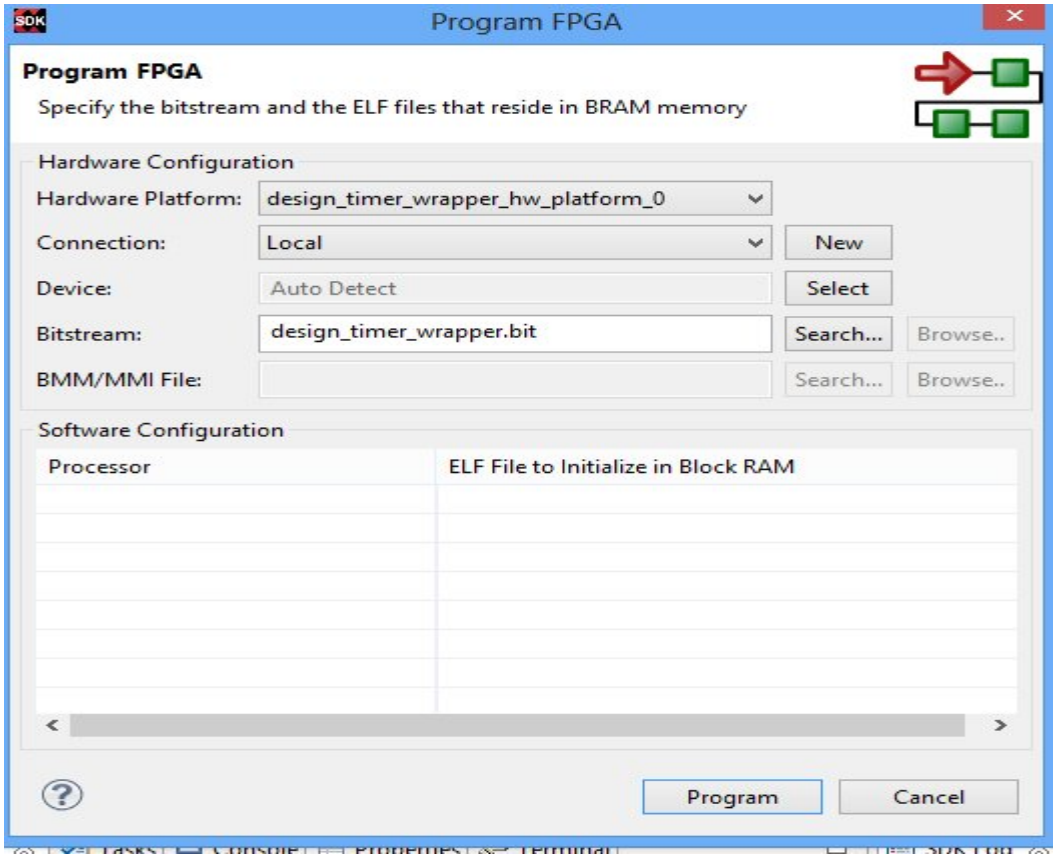
On peut relancer la compilation du fichier



Pour pouvoir développer logiciellement notre programme il faut au préalable programmer la logique programmable du FPGA



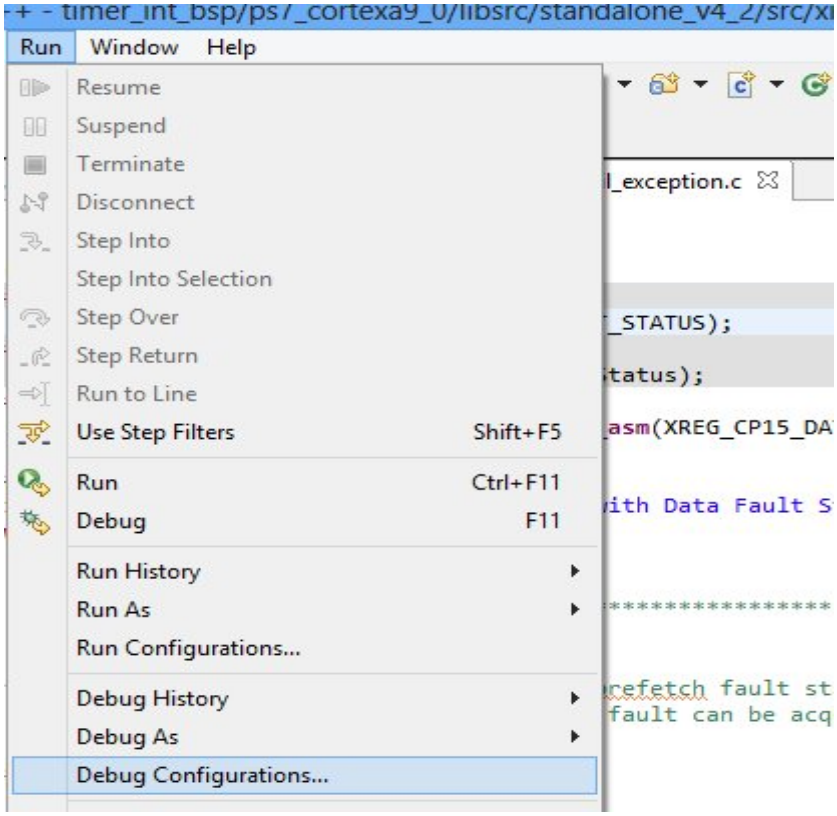
Sélectionnez le fichier bitstream



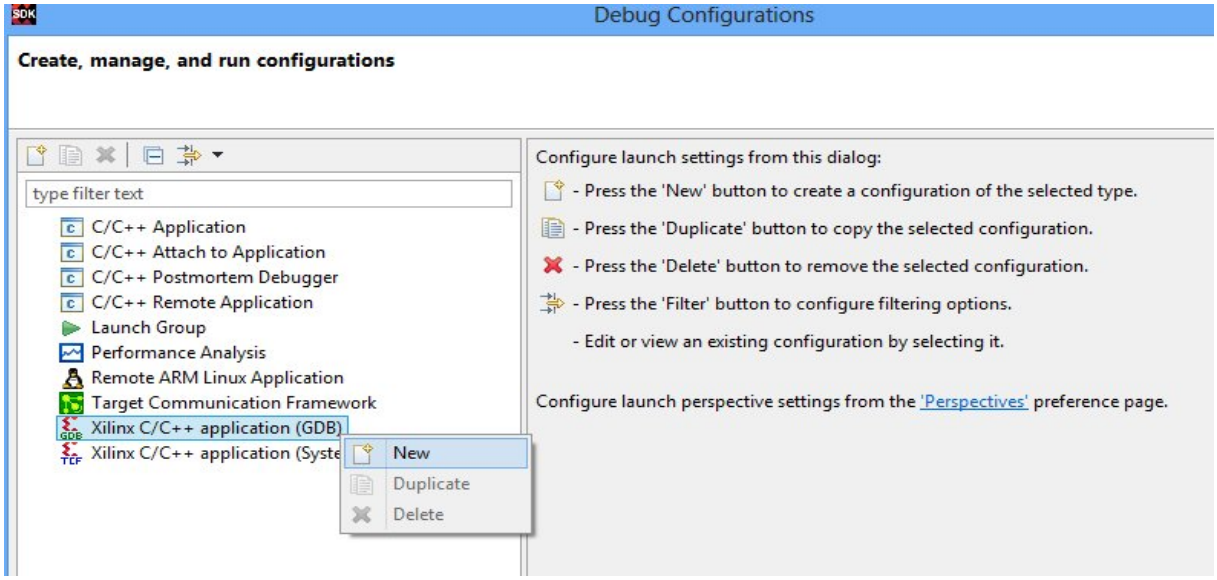
La programmation du PL est alors effective

Par la suite deux possibilités s'offrent à nous lancez directement le programme ou déboguer ce programme

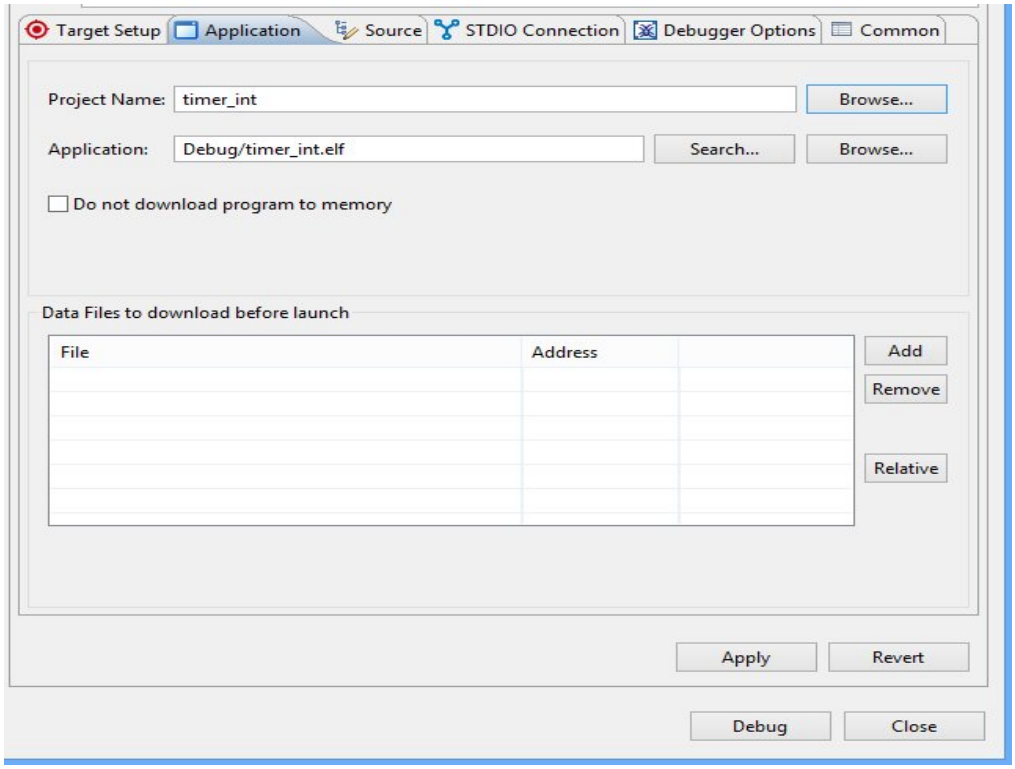
Le mode debuggage



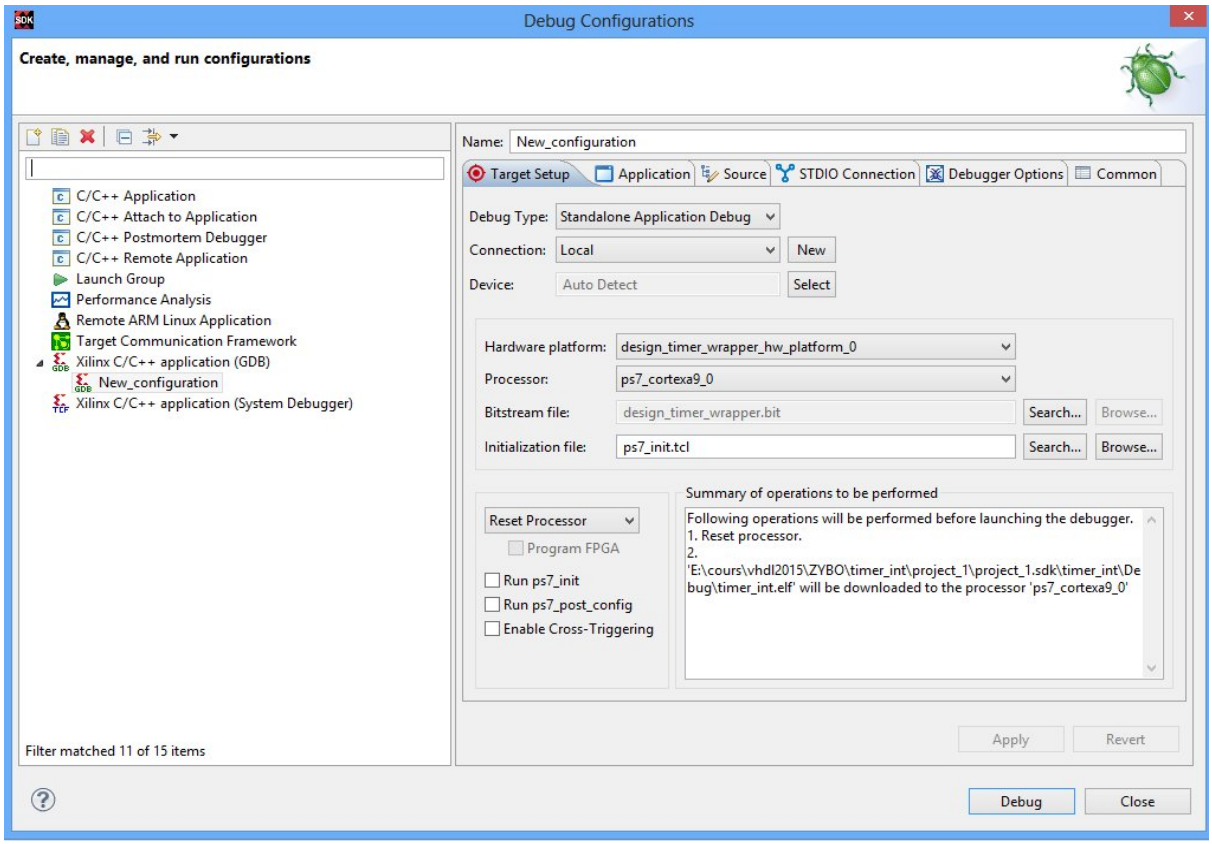
Créez une nouvelle configuration de debugage pour la carte ZYBO



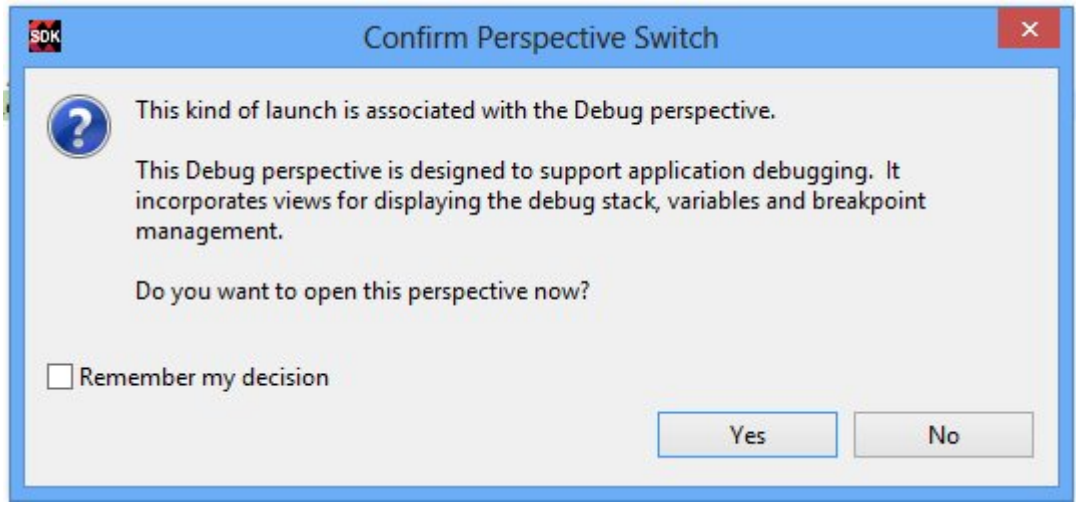
Sélectionnez notre applicatif



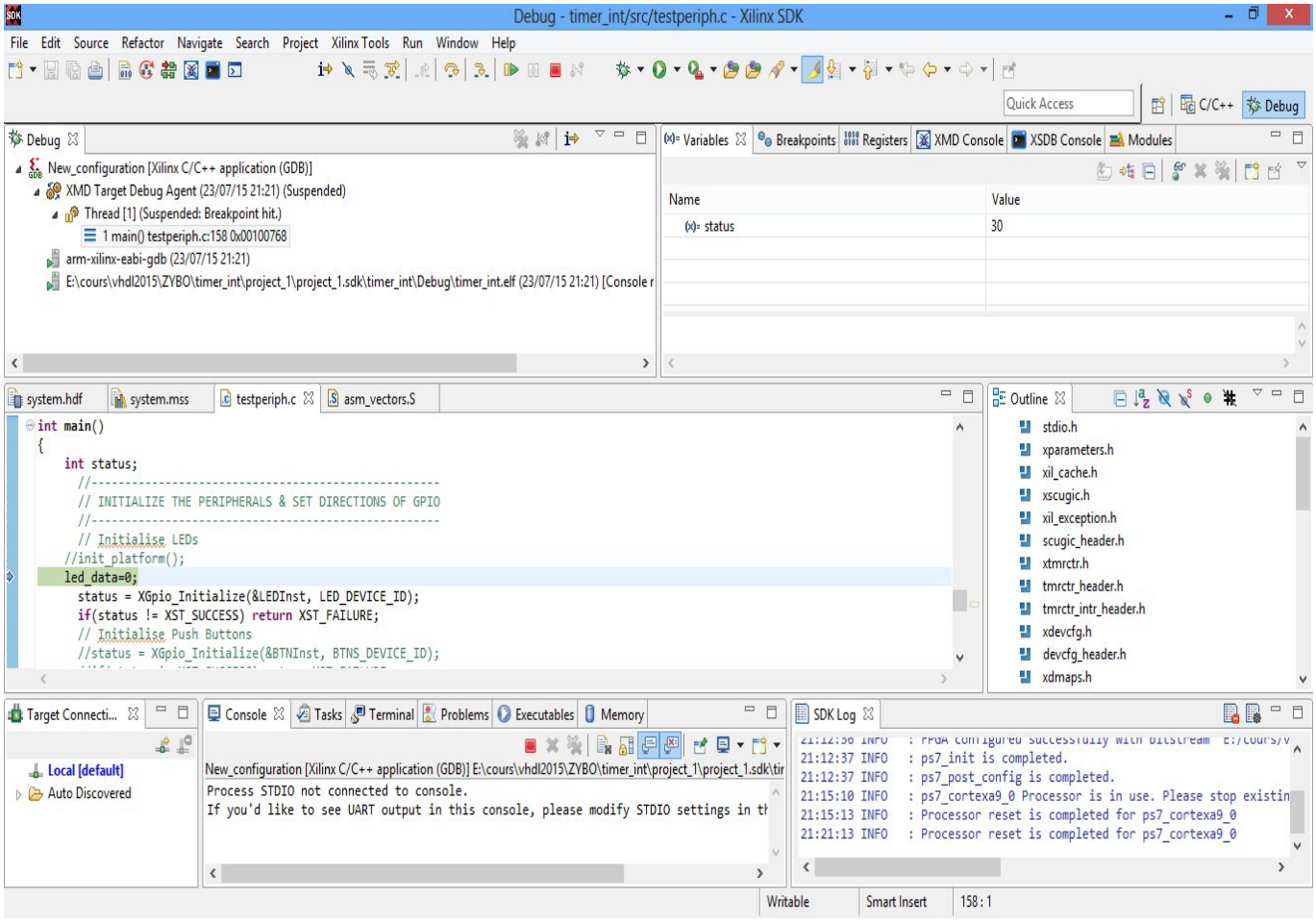
Notre configuration est la suivante



Répondre yes pour ouvrir la perspective



Le debugger se lance



Pour lancer le mode Run sans debugage cliquez droit sur le projet lié à notre applicatif timer_in puis run as → Launch on hardware

