Master Informatique

M1 Spécialité SESI

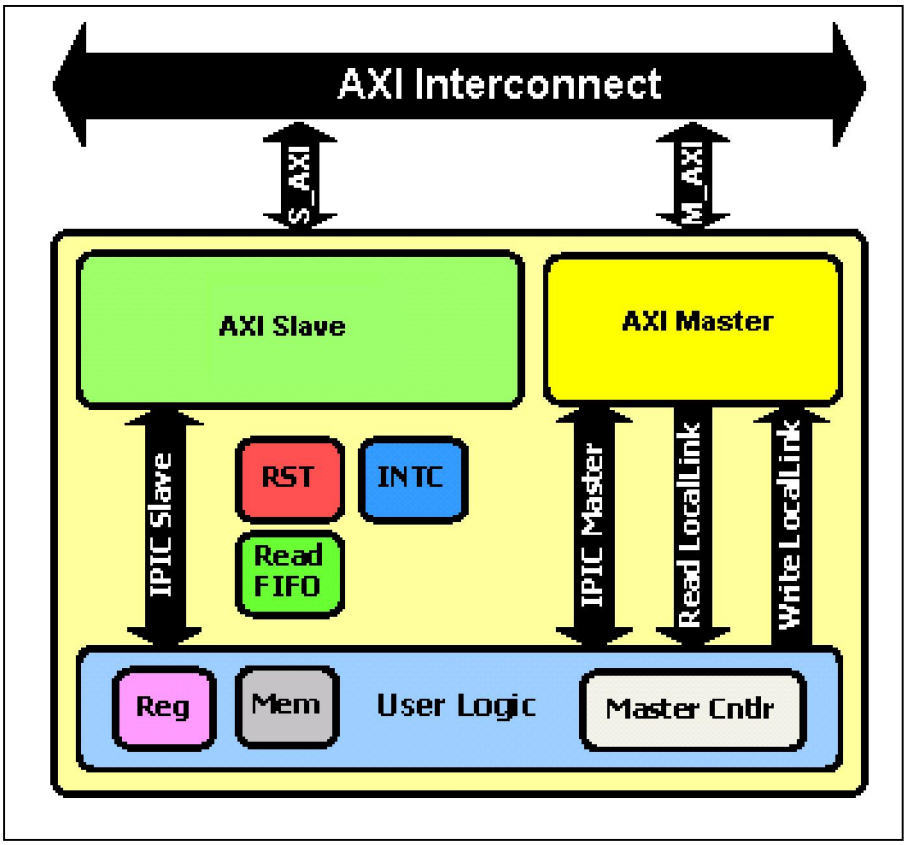
UE 4I108 - J.Denoulet

TP – FPGA1

SYSTEMES PROGRAMMABLES

3EME PARTIE – CONCEPTION D'IP POUR LE MICROBLAZE

L'objectif de ce TP est e réaliser une ***IP*** destinée à être connectée au ***Microblaze***, en remplacement du contrôleur de LED. Le système ainsi généré sera programmé pour réaliser quelques exemples d'applications



Comme pour la partie précédente, le développement sera réalisé grâce aux outils ***Xilinx***

- ***Vivado*** pour le développement de l'IP et la génération de la plate-forme matérielle
- ***SDK (Software Development Kit)*** pour le développement et l'exécution de l'application logicielle

1) Création d'une IP Contrôleur de LED

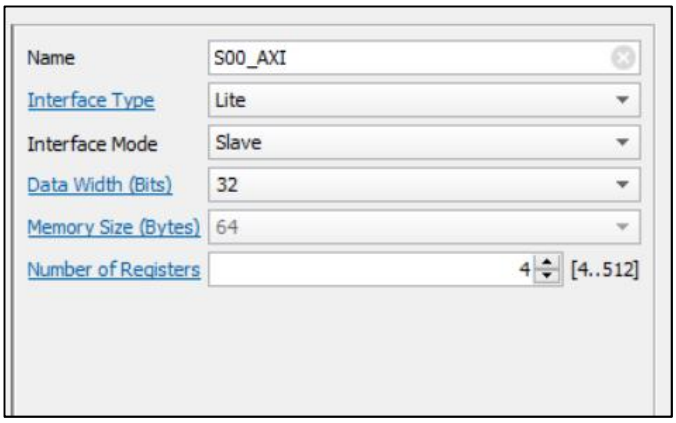
- Lancer ***Vivado*** et ouvrir le projet créé lors du TP précédent
  - Pour conserver le travail fait au TP2, cliquer sur le menu ***File*** puis ***Save Project As*** et donner un nouveau nom puis un nouveau répertoire pour votre projet (***TP3*** par exemple).

- Aller dans le menu ***Tools***, sélectionner ***Create and Package IP***.
  - Une fenêtre s'ouvre : cliquer sur ***Next*** puis dans la fenêtre suivante choisir ***Create a new AXI4 peripheral.***

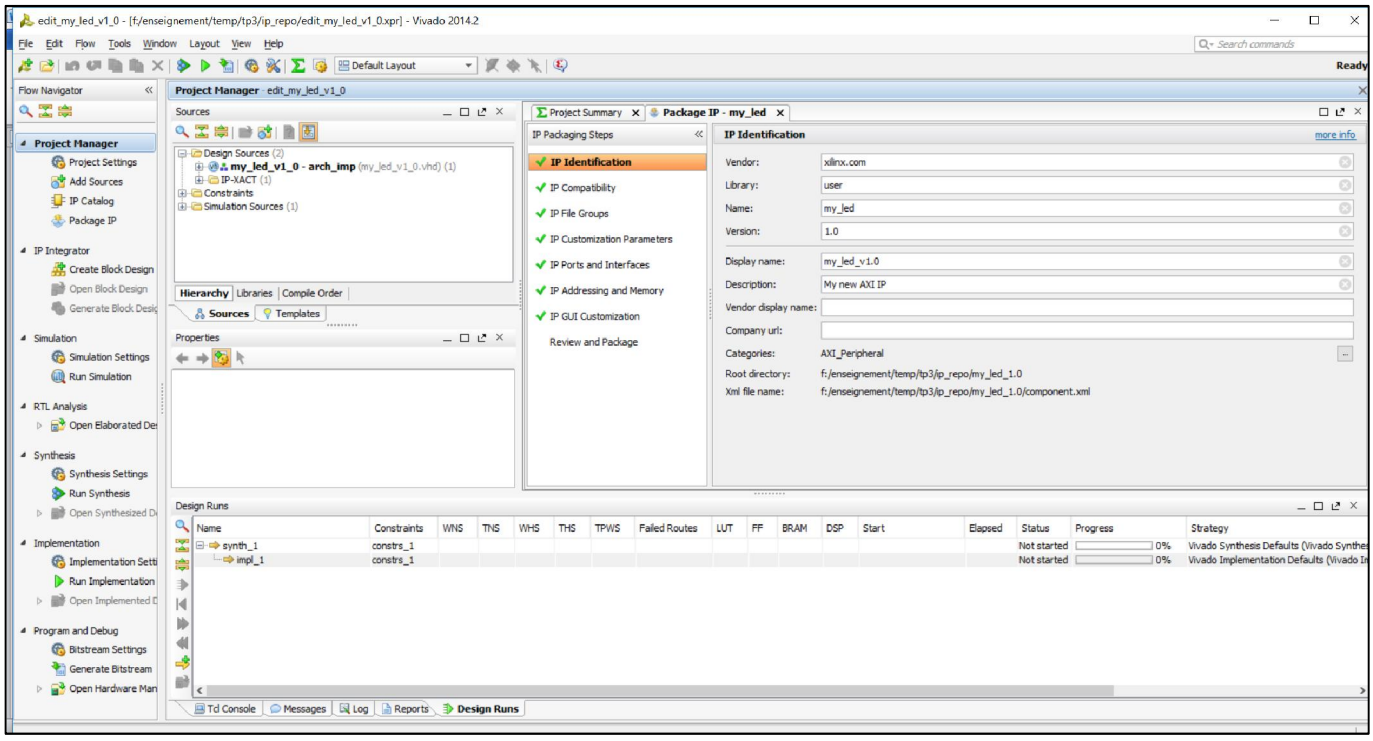
- Dans la fenêtre suivante, donner un nom (par exemple ***my_led***) et éventuellement une description à votre IP puis ***Next***

- Dans la fenêtre suivante, choisir comme interface bus ***AXI4-Lite*** puis cliquer sur ***Next***

- Dans la fenêtre suivante, vérifier que les paramètres sont bien ceux de la fenêtre ci-contre et cliquer sur ***Next***..
  - Noter le nombre de registres (4 étant le plus petit nombre possible)



- Dans la fenêtre suivante, terminer en cliquant sur ***Edit IP*** puis ***Finish***.
  - Une deuxième fenêtre ***Vivado*** s'ouvre alors sous la forme d'un projet vous permettant de créer ou modifier la fonctionnalité de votre ***IP***.

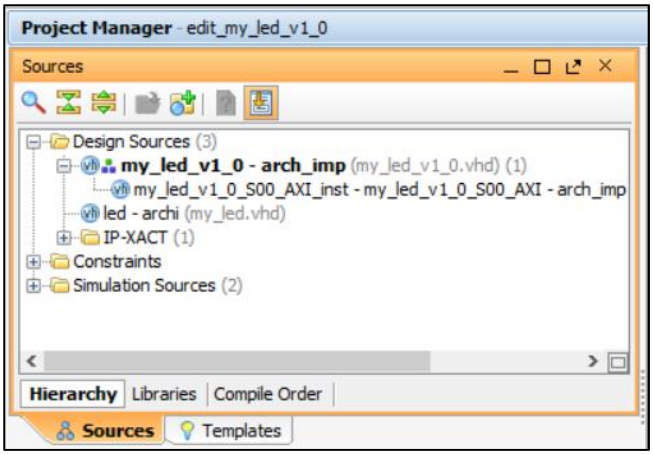


- Aller dans le menu ***File*** et cliquer sur ***New File*** et créer un fichier appelé ***my_led.vhd***

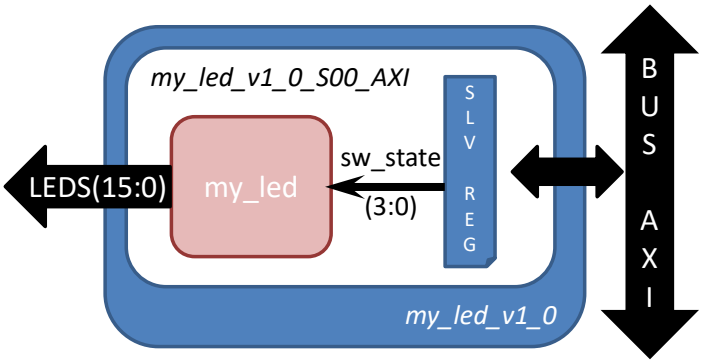
- Dans ce fichier décrire un module VHDL avec une entité appelée ***led*** et possédant :
  - 1 entrée sur 4 bits (***sw_state***)
  - 1 sortie sur 16 bits (***leds***)

  et qui allume 4 LED si un des bits d'entrée est à 1
  - Par exemple, si ***sw_state(0)*** = '1', on allume les ***LED(3 :0)***

- Sauvegarder le fichier à la racine de votre projet ***Vivado TP3*** et ajouter le fichier aux sources du projet.
  - La liste des sources de votre projet IP comprend à présent 3 sources VHDL
    - ***my_led_v1_0.vhd*** : Le fichier top de l'IP
    - ***my_led_v1_0_S00_AXI.vhd*** : Ce fichier implémente le protocole AXI pour que l'IP puisse être reliée au bus. Il doit aussi instancier la fonctionnalité de l'IP.
    - ***my_led.vhd*** : Implémente la fonctionnalité de l'IP.



- Il faut à présent modifier les codes VHDL pour :
  - Faire figurer en sortie de l'***IP*** les 16 bits de commande des ***LED***.
  - Instancier l'entité de ***my_led.vhd*** dans le fichier ***my_led_v1_0_S00_AXI.vhd***



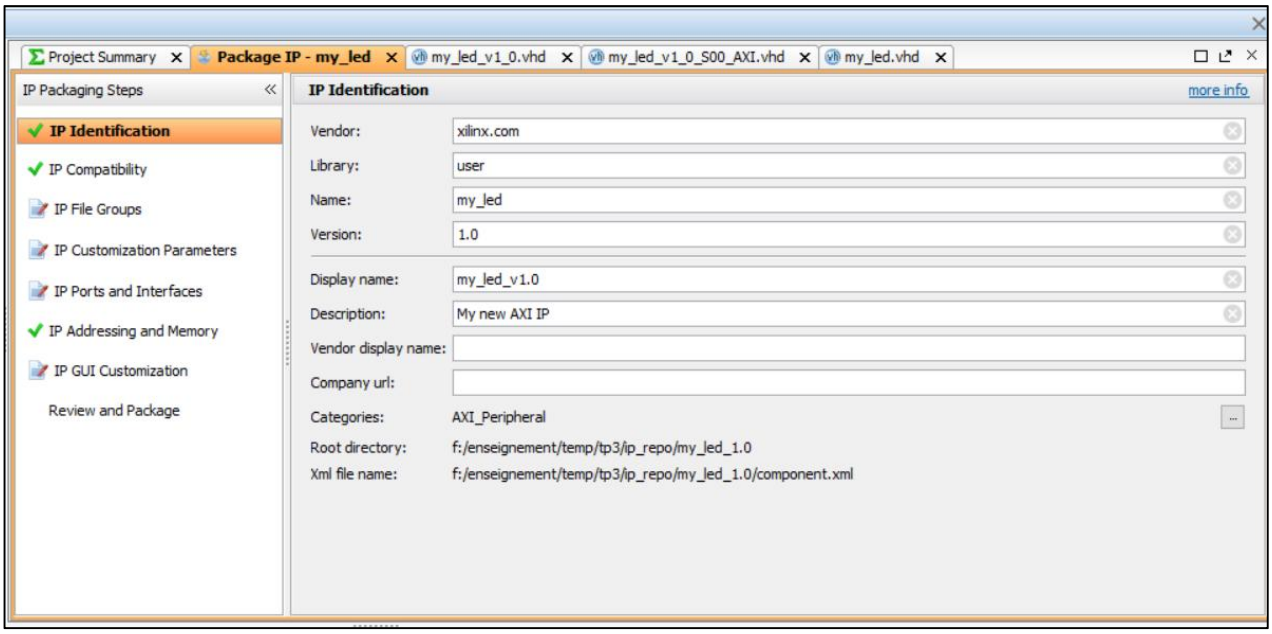
- L'architecture de l'***IP*** aura alors une structure semblable à celle de la figure ci-dessus.
  - ***Slv_reg*** correspond au nom des 4 registres de configuration de l'***IP***

- Pour cela, ouvrir d'abord le fichier ***my_led_v1_0.vhd.***
  - Dans l'entité, après la ligne `-- Users to add ports here`, ajouter :
    - `leds: out std_logic_vector(15 downto 0);`
  - Chercher la ligne `-- component declaration`. Dans la liste des ports, ajouter avant ***S_AXI_ACLK*** la ligne suivante :
    - `leds: out std_logic_vector(15 downto 0);`
  - Chercher la ligne `port map(` et ajouter dans la liste des instanciations la ligne suivante :
    - `leds => leds,`

- Ouvrir ensuite le fichier ***my_led_v1_0_S00_AXI.vhd.***
    - Dans l'entité, après la ligne `-- Users to add ports here`, ajouter :
        - `leds: out std_logic_vector(15 downto 0);`
    - Juste avant le `begin` de l'architecture, ajouter la ligne suivante :
        - `signal sw_state: std_logic_vector(3 downto 0);`
    - Chercher la ligne `-- Add user logic here` et ajouter la ligner suivante :
        - `L0: entity work.led port map (sw_state,leds);`
- Après sauvegarde des fichiers, l'entité ***led*** est à présent rattachée au module ***my_led_v1_0_S00_AXI***

- Dans le fichier ***my_led_v1_0_S00_AXI.vhd,*** chercher la déclaration des signaux ***slv_reg0,1,2,3***
    - Ces 4 signaux correspondent aux 4 registres de configuration de votre ***IP***.
    - Nous allons fixer pour l'***IP*** le comportement suivant :
        - Si le ***bit 0*** du ***slv_reg0*** est à 1, les ***LED(3 :0)*** sont allumées. Elles sont éteintes sinon.
        - Si le ***bit 1*** du ***slv_reg0*** est à 1, les ***LED(7 :4)*** sont allumées. Elles sont éteintes sinon.
        - Si le ***bit 0*** du ***slv_reg1*** est à 1, les ***LED(11 :8)*** sont allumées. Elles sont éteintes sinon.
        - Si le ***bit 1*** du ***slv_reg1*** est à 1, les ***LED(15 :12)*** sont allumées. Elles sont éteintes sinon.
    - Pour cela, ajouter dans l'architecture (dans la zone indiquée par le commentaire `– Add user logic here`) 4 instructions qui :
        - Affectent à ***sw_state(0)*** le ***bit 0*** du ***slv_reg0***
        - Affectent à ***sw_state(1)*** le ***bit 1*** du ***slv_reg0***
        - Affectent à ***sw_state(2)*** le ***bit 0*** du ***slv_reg1***
        - Affectent à ***sw_state(3)*** le ***bit 1*** du ***slv_reg1***

- Après sauvegarde des fichiers, cliquer sur l'onglet ***Package IP***.



- Cliquer sur ***IP File Groups*** puis sur ***Merge Changes from IP File Groups Wizard*** dans le bandeau jaune.
    - Faire de même pour les autres ***IP Packaging Steps*** jusqu'à ce qu'il n'y ait plus que des ✓ verts devant chaque ***Packaging Step***.
    - Cliquer alors sur ***Review and Package*** puis sur ***Repackage IP***.
        - ***Vivado*** ferme alors la fenêtre du projet de votre IP.

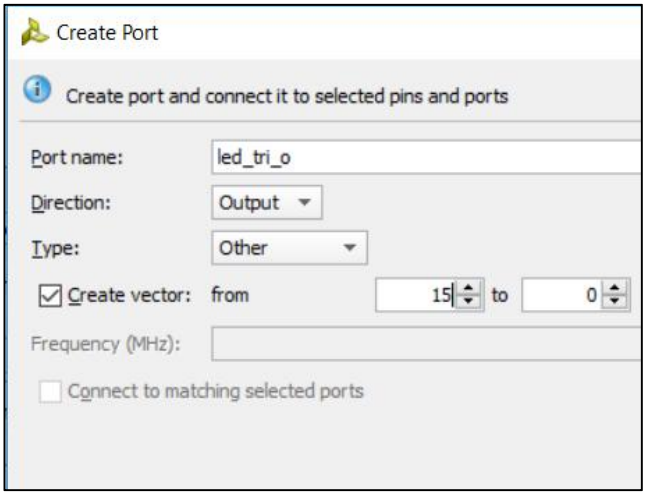
2) Intégration de l'IP au système Microblaze

- De retour dans le projet ***Vivado*** principal, ouvrir le ***Block Design*** contenant le système ***Microblaze***.
  - Nous allons à présent :
    - Retirer du système l'***IP GPIO led_switch***.
    - Modifier l'***IP GPIO boutons*** pour qu'elle gère à présent les interrupteurs
    - Ajouter notre ***IP my_led***.
    - Apporter quelques modifications supplémentaires pour finaliser le système.

- Dans le ***Diagram***, cliquer sur le bloc ***led_switch*** puis supprimer le module.
  - Supprimer également les connecteurs ***sw*** et ***led***.

- Cliquer sur le bloc ***boutons***, et dans la fenêtre ***Block Properties***, changer le nom du module en ***sw***.
  - Faire de même pour le connecteur ***boutons***.

- Cliquer avec le bouton droit sur le ***Diagram*** et choisir ***Add IP***. Chercher puis ajouter l'***IP my_led***.
  - Cliquer sur ***Run Connection Automation***. Vérifier que le paramètre ***Auto*** est sélectionné puis valider.
  - Cliquer sur le ***Diagram*** avec le bouton droit et choisir ***Create Port***. Fixer les caractéristiques du port comme sur la fenêtre ci-contre.
  - Connecter manuellement ce port à la sortie ***leds*** de votre ***IP my_led***.



NB : le nom ***led_tri_o*** est choisi pour ne pas avoir à modifier le fichier ***XDC***.

- Comme il ne reste plus qu'une source d'interruption (le bloc ***GPIO sw***), supprimer le bloc ***Concat*** du ***Diagram*** puis connecter manuellement la ligne d'interruption du bloc ***sw*** à l'entrée ***intr*** du ***contrôleur d'interruptions***.
  - Ne pas faire attention au fait que l'entrée ***intr*** soit sur 2 bits. Cela sera corrigé automatiquement par la suite.

- Double cliquer sur l'***IP sw*** et dans l'onglet ***IP Configuration***, changer la taille de 3 à 4 bits d'entrée.

- Ouvrir le fichier ***XDC***. Mettre en commentaire les assignations de broches correspondant aux ***boutons***, puisqu'ils ne sont plus utilisés.

- Cliquer sur ***Validate Design*** pour vérifier qu'il n'y a pas d'erreurs dans le ***Diagram***.
  - Si tout est correct, vous constaterez que l'entrée ***intr*** du ***contrôleur d'interruptions*** est passée sur 1 bit.

- Nous pouvons à présent implémenter la plate-forme matérielle et générer le bitstream du FPGA via la fenêtre ***Flow Navigator*** en cliquant sur ***Generate Block Design*** puis sur ***Generate Bitstream***
    - Si la synthèse révèle des erreurs dans le code VHDL de votre ***IP***, vous pouvez faire modifications en cliquant avec le bouton droit sur votre ***IP*** dans le ***Diagram*** et en choisissant ***Edit IP in Package***
        - Ne pas oublier de repackager l'***IP*** après chaque modification.

- Lorsque l'implémentation est terminée, s'il n'y a pas d'erreurs, exporter le système vers ***SDK***, afin que l'application logicielle puisse être développée correctement.
    - Dans le menu ***File***, cliquer sur ***Export*** puis ***Export Hardware***
    - Vérifier que l'export inclut le bitstream
    - Cliquer sur ***File*** puis ***Launch SDK***

3) Développement logiciel

- Dans SDK, un nouveau projet matériel a été créé : TP2_wrapper_hw_platform1 (si votre Block Design sous Vivado s'appelle TP2)
    - Ce projet correspond à votre nouvelle plate-forme matérielle avec votre IP my_led.

- Fermer à présent tous les projets utilisés précédemment pour ne garder que ce nouveau projet matériel.
    - Créer ensuite un nouveau projet logiciel (***TP3***) avec un projet BSP associé.

- Une fois les projets créés, aller dans le ***projet BSP*** au répertoire ***microblaze_0/include*** pour ouvrir le fichier ***my_led.h***
    - Ce fichier généré automatiquement est le driver de votre ***IP my_led***. Il contient une série de ***#define*** qui vont vous permettre d'utiliser des fonctions de lecture ou d'écriture des registres de votre IP.
        - `MY_LED_S00_AXI_SLV_REG0_OFFSET` vous permettra d'accéder au slv_reg0
        - `MY_LED_S00_AXI_SLV_REG1_OFFSET` vous permettra d'accéder au slv_reg1
        - Etc…

Fonction	Rôle
`MY_LED_mWriteReg``(BaseAddress, RegOffset, Data)`	Ecriture dans l'IP
`Data = `**`MY_LED_mReadReg`**`(BaseAddress, RegOffset)`	Lecture de l'IP

- Avec:
    - ***BaseAddress***: L'adresse de base de l'***IP***, que l'on peut retrouver dans ***xparameters.h***
    - ***RegOffset*** : L'adresse à ajouter à celle de base pour atteindre le registre souhaité. (Utiliser pour cela l'un des alias de type `MY_LED_S00_AXI_SLV_REG0_OFFSET`)
    - ***Data*** : Donnée lue ou à écrire. (de type integer)

- Créer un programme C dans votre projet logiciel puis écrire un programme qui lit l'état des 4 interrupteurs et qui allume les LED par blocs de 4 si les interrupteurs sont actifs. (cf. fonctionnement de l'***IP my_led*** page 4)
    - Ne pas oublier d'inclure le .h de votre driver pour pouvoir utiliser les fonctions ci-dessus.