Implémentation PL/PS pour Ultra96v2

# Installation de l'environnement

* Vivado 2019.2
* Vitis 2019.2

# Préparation Vivado

# Ajout d’un Board Definition File :

Télécharger le BDF : [lien de téléchargement](https://github.com/Avnet/bdf/tree/master/ultra96v2/1.0)

Coller le dossier dans : *Xilinx/Vivado/2019.2/data/boards/board\_files*

Téléchargement du Constraint File :

Aller sur la [page de téléchargement](https://www.element14.com/community/docs/DOC-92693/l/ultra96v2constraints190430zip) d’Avnet

Télécharger : Ultra96\_V2\_constraints\_190430.zip

Extraire le contenu de l’archive

# Réalisation partie PL

Création d’un nouveau projet :

*File > Project > New…*

*Next*

Choix du nom du projet et de la localisation

*Next*

Choisir *RTL Project*

*Next*

*Boards > Ultra96v2 Evaluation Platform*

*Next*

*Finish*

Ajout de Constraints :

*File* > *Add Sources* > *Add or create constraints*

N*ext*

*Add Files* > Choisir le fichier .xdc téléchargé précédemment

Cocher “*Copy constraints files into project”*

*Finish*

Réalisation du bloc design :

Flow Navigator > IP Integrator > Create Bloc Design

Ajout et connexion des différents blocs IP

Paramétrages des IP

Paramétrages du Zynq UltraScale+ MPSoC :

Définition des adresses des blocs IP slave :

Windows > Address Editor

Clic Droit sur les éléments non assignés > Assign Address

Choisir la valeur de Range

Une fois le design terminé, il faut maintenant compiler le projet pour qu’il puisse être utiliser par la partie PS.

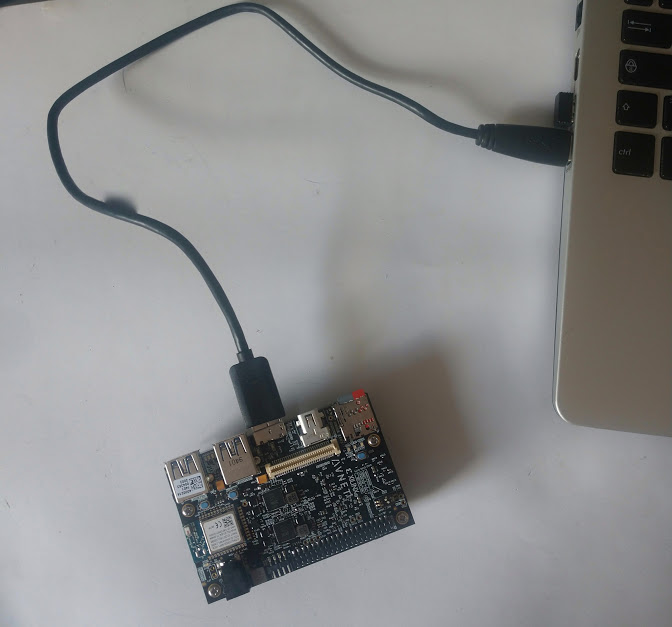
Exporter un fichier .xsa :

1. Diagram > Validate design (F6)
2. Sources > Design Sources > Clic droit sur files.bd > Create HDL Wrapper…
3. Flow Navigator > Synthesis > Run Synthesis >...> Run implementation >...> Generate bitstream
4. File > Export > Export Hardware > Cocher “Include bitstream” > OK

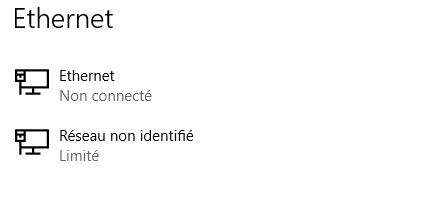
# Implémentation PS

Pour implémenter le design PL avec Pynq, la méthode utilisé est la suivante:

* Extraire du fichier .xsa les fichiers suivant: .bit .hwh et .tcl . Ces fichiers doivent avoir le même nom.
* Installer Pynq 2.5 pour l’Ultra96 sur la carte SD (image disponible [ici](http://bit.ly/2MMrXcS), md5sum: 3F8B5199C767EC64CAF414E3A3B15296). Et insérez la dans l’Ultra96.
* Brancher un câble USB entre l’Ultra96 et votre ordinateur:



* Sur votre ordinateur, un réseau filaire s’est créé:



* Vous pouvez vous connecter à l’adresse 192.168.3.1. Cette adresse vous permet d'accéder à l’interface de l’Ultra96.
* A partir de là deux choix sont possible:
  + utiliser un terminal (New→ Terminal): à partir de ce terminal, il est possible d’installer ROS (cf: [Installation ROS sur Pynq](https://docs.google.com/document/d/1yphL6LEXNzAzm-QwGpB4q3FQGsrOuFNRsUZvGM9AUf4/edit?usp=sharing)). Pour notre part, nous avons mis en place un svn pour développer à distance. Pour connecter l’Ultra96 au wifi, suivre le tuto sur la carte: common/wifi.ipynb.
  + coder sur les fichiers .ipynb. Ce type de fichier permet de développer rapidement un code en python en exécutant au fur et à mesure le code.
* Pour importer le hardware spécifique, il faut télécharger les trois fichiers (.hwh, .tcl et .bit) sur l’Ultra96.
* Grâce aux [librairies Overlay, MMIO et GPIO](https://pynq.readthedocs.io/en/v2.5.1/pynq_libraries.html) en python, il est possible de gérer les registres, pin…. Dans notre cas, pour l’uart, nous avons utiliser ce code [disponible sur git](https://github.com/parthpower/axi_uartlite_pynq).