REX Red Pitaya ADC-DAC démo & Echo Ethernet

Red Pitaya 125-14 STEMlabBoard

Fabrice Wiotte
fabrice.wiotte@univ-paris13.fr

Laboratoire de physique des lasers

Université **S**orbonne **P**aris **N**ord

Sommaire

Présentation de la Red STEMlab Board :

Introduction

La RedPitaya est une carte FPGA « bas coût », une carte de développement comme Arduino ou Raspberry Pi. La particularité de cette carte est sa capacité de gérer des signaux rapides jusqu'à 50MHz. La carte peut être utilisée rapidement avec les applications disponibles, ou peut être exploitée comme une plate-forme ouverte. Il est possible de l'adapter à un grand nombre de projets nécessitant un contrôle rapide et performant comme du traitement de signal ou de la radio logicielle.

C'est une plate-forme d'acquisition et de génération de signaux RF de la taille d'une carte de crédit. La carte est basée sur un système sur puce SOC — FPGA de la gamme Xilinx Zynq qui permet de combiner la capacité de programmation d'un cœur ARM Cortex-A9 à double cœur (LINUX embarqué).

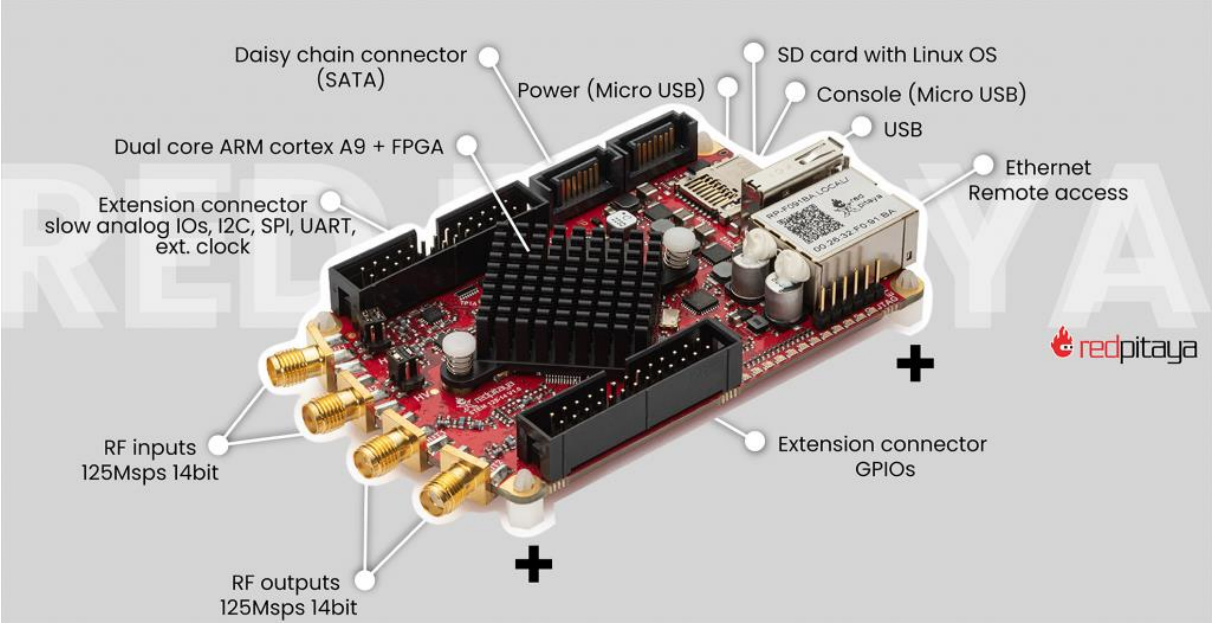
Schéma de principe et architecture

L'instrument est équipé de deux entrées et deux sorties analogiques RF (125 MS/s). Différentes résolutions sont proposées (10,14 ou 16 bits), quatre entrées et quatre sorties analogiques (100 kS/s), ainsi que de 16 ports entrés/sorties logiques universels. Coté connectivité, nous pouvons compter sur un port ETHERNET 1 Gbit, un port USB 2.0 et autre protocoles (I2C, SPI, UART), le tout peut être autonome grâce à la carte SD.

Pour le soft la carte est basée sur le système d'exploitation GNU/Linux. Possibilité d'intégration dans son propre système/produit, elle peut être programmée avec différents niveaux avec une variété d'interfaces logicielles, les langages de programmation sont : HDL/Verilog, C/C++, Python/Jupyter, et un serveur web intégré (NGINX) pour des interfaces web basées sur HTML/JavaScript. Elle peut être utilisé comme oscilloscope et générateur de signaux, spectre, analyseur de Bode, analyseur logique, compteur LCR*, streaming, SDR ou analyseur de réseau vectoriel* grâce à l'écosystème fourni par STEMlab.

On peut également utiliser PyRPL (Python Red Pitaya Lockbox), un progiciel open source en Python pour les expériences d'optique quantique contrôlées par FPGA avec des applications disponibles : Lock-in Amplifiers, PID...

STEMlab Board 125-14 :



Liens utiles sur le site Red Pitaya STEMlab :

https://redpitaya.com/stemlab-125-14/

https://redpitaya.com/rtd-iframe/?iframe=https://redpitaya.readthedocs.io/en/latest/developerGuide/hardware.html

https://redpitaya.com/applications-measurement-tool/fpga/

https://redpitaya.readthedocs.io/en/latest/developerGuide/software/build/fpga/fpga.html

-> Quick Start

-> Developers guide -> Hardware
-> Software

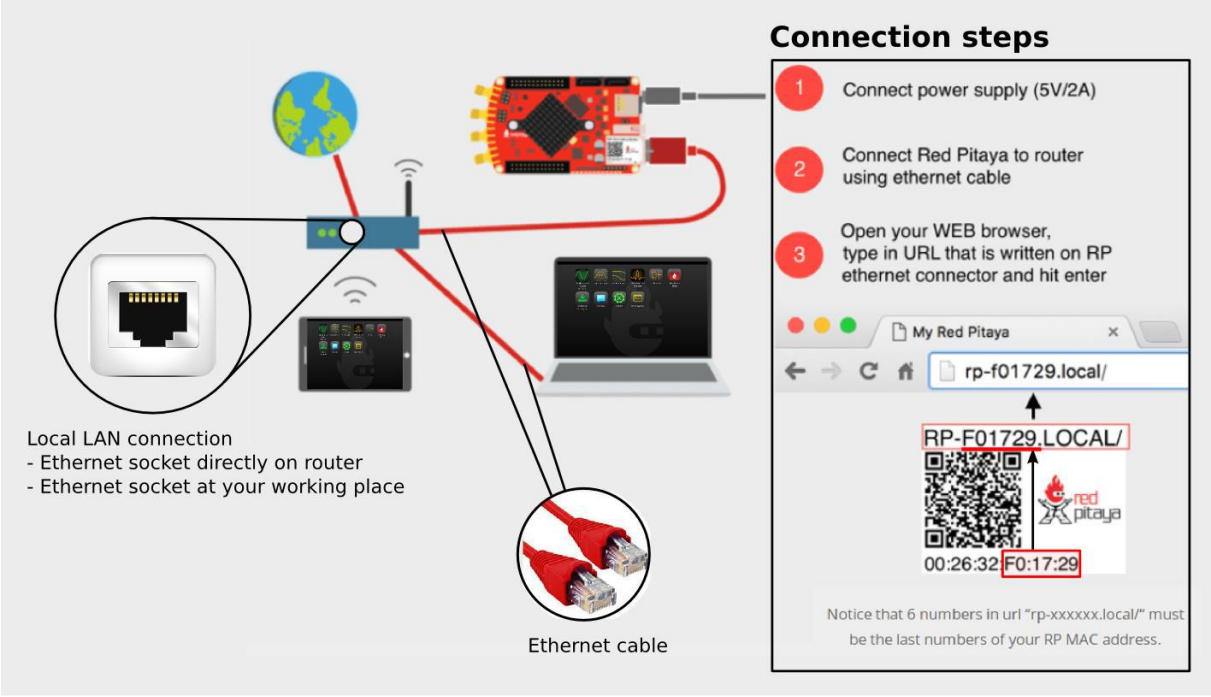
STEMlab Board 125-10 versus STEMlab Board 125-14:

		STEMlab 125-10	STEMlab 125-14
Basic	Processor	Dual Core ARM Cortex A9	Dual Core ARM Cortex A9
	FPGA	Xilinx Zynq 7010 SOC	Xilinx Zynq 7010 SOC
	RAM	256 MB (2 Gb)	512 MB (4 Gb)
	System memory	Micro SD up to 32 GB	Micro SD up to 32 GB
	Console connection	USB to serial converter required	micro USB
	Power connector	Micro USB	Micro USB
	Power connector	Micro USB	Micro USB
	Power consumption	5 V, 1,5 A max	5 V, 2 A max
Connectivity	Ethernet	1 Gbit	1 Gbit
	USB	USB 2.0	USB 2.0
	WiFi	requires WIFI dongle	requires WIFI dongle
	Synchronisation	-	Daisy chain connector (up to 500 Mbps)
RF inputs	RF input channels	2	2
	Sample rate	125 MS/s	125 MS/s
	ADC resolution	10 bit	14 bit
	Input impedance	1 MOhm / 10 pF	1 MOhm / 10 pF
	Full scale voltage range	+-20 V	+-20 V
	Absolute max. Input voltage range	30 V	30 V
	Input ESD protection	Yes	Yes
	Overload protection	Protection diodes	Protection diodes
RF outputs	RF output channels	2	2
	Sample rate	125 MS/s	125 MS/s
	DAC resolution	10 bit	14 bit
	Load impedance	50 Ohm	50 Ohm
	Voltage range	+-1 V	+-1 V
	Ouput slew rate	200 V/us	200 V/us
	Short circut protection	Yes	Yes
	Connector type	SMA	SMA
Extension connector	Digital IOs	16	16
	Analog inputs	4	4
	Analog inputs voltage range	0-3,5 V	0-3,5 V
	Sample rate	100 kS/s	100 kS/s
	Resolution	12 bit	12 bit
	Analog outputs	4	4
	Analog outputs voltage range	0-1,8 V	0-1,8 V
	Communication interfaces	I²C, SPI, UART	I²C, SPI, UART
	Dimensions	107 x 60 x 21 mm	107 x 60 x 21 mm

ADC : LTC2145-14

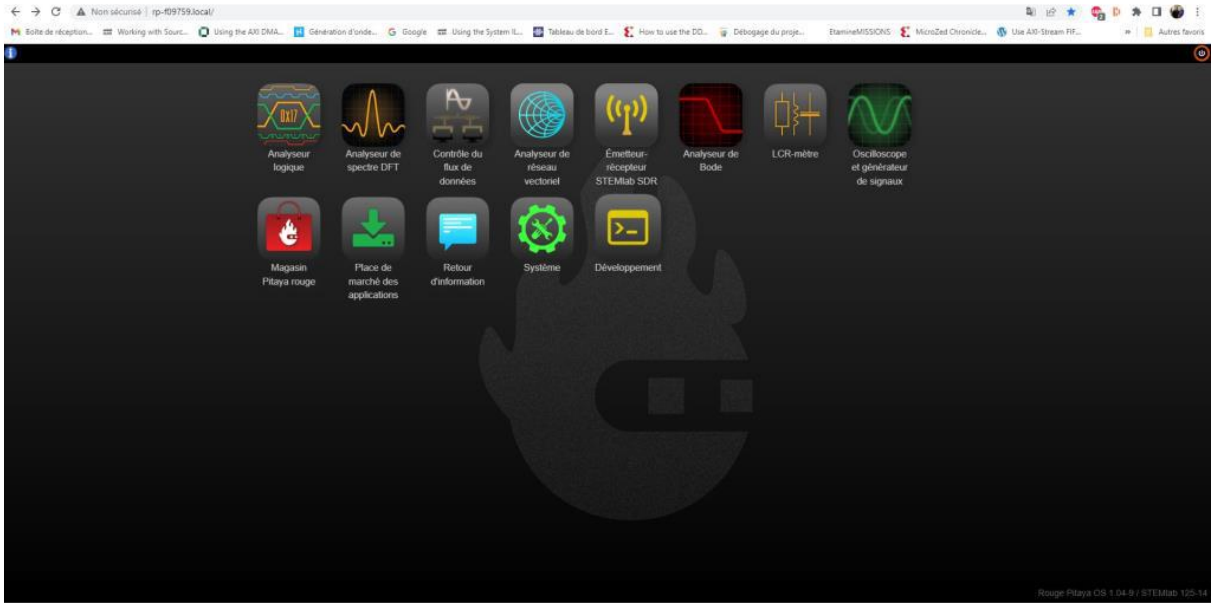
DAC LTC2145-14

STEMlab Board : Accès mode local ou adresse IP

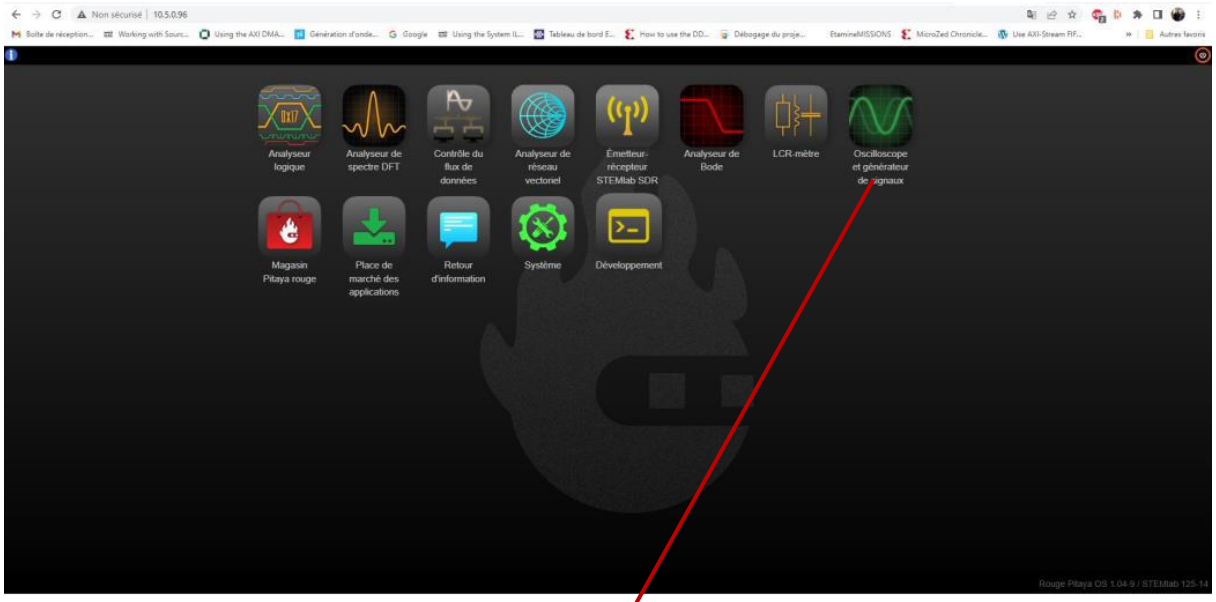


Ecosystème installé sur la carte SD :

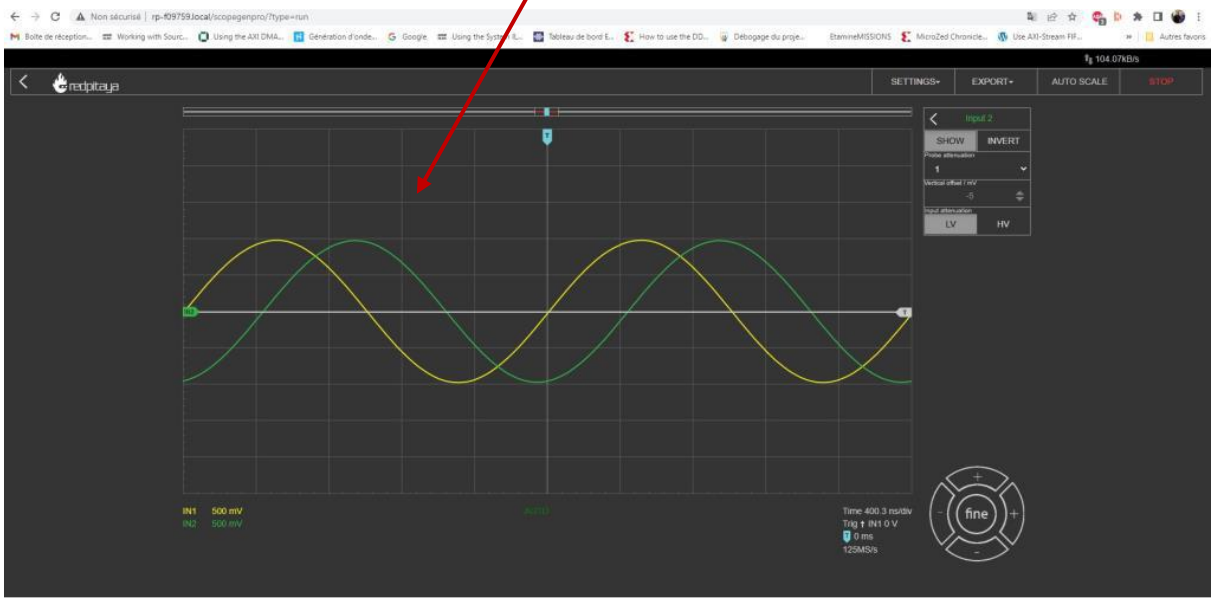
Accès local RP-F****LOCAL/



Accès distant adresse IP



Oscilloscope :



1. **Objectif du TP :**

Prendre en main et programmer la RedPitaya avec Vivado « bare metal »,

C'est à dire sans la carte SD insérée.

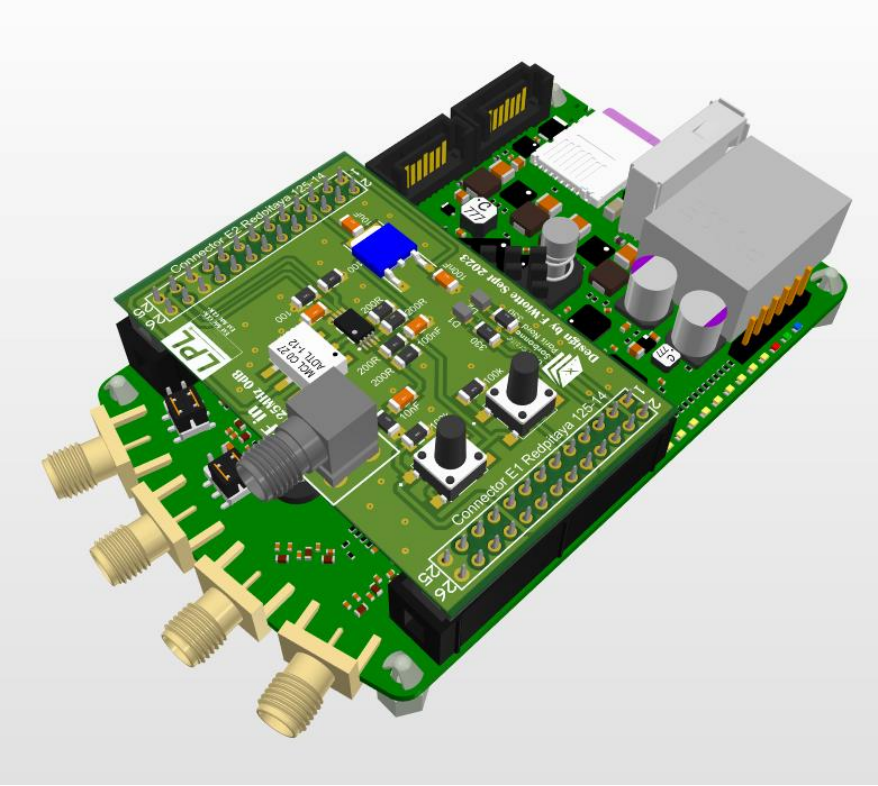
La Red Pitaya dispose de 2 interfaces de programmation :

1) Une prise micro USB avec une puce FTDI derrière (nommée CON) à l'arrière de la carte.
2) Une interface JTAG (pin header nommé JTAG) sur la face avant.

Nous allons utiliser un **câble USB-JTAG Xilinx HS2** pour :

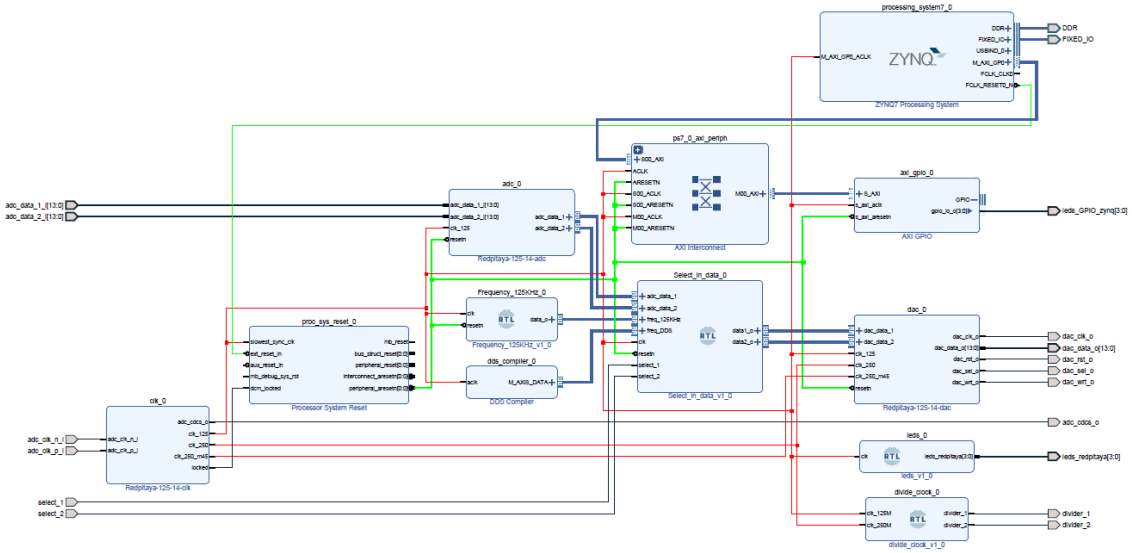
1) Programmer sous Vivado 2022_2 le HARDWARE- > ADC-DAC et IP maison
2) Programmer le SOFTWARE avec Vitis 2022_2-> Exemple Echo Ethernet.
3) Créer un fichier BOOT.bin sur carte SD pour la sauvegarde du programme.
4) Utiliser une carte fille développée pour la RedPitaya qui permet une entrée d'horloge externe pour la synchronisation.

Carte fille sur RedPitaya 125_14 with external Clock



3. Demo RedPitaya sous Vivado 2022_2

3.1 Bloc design du TP Demo RedPitaya sous Vivado 2022_2

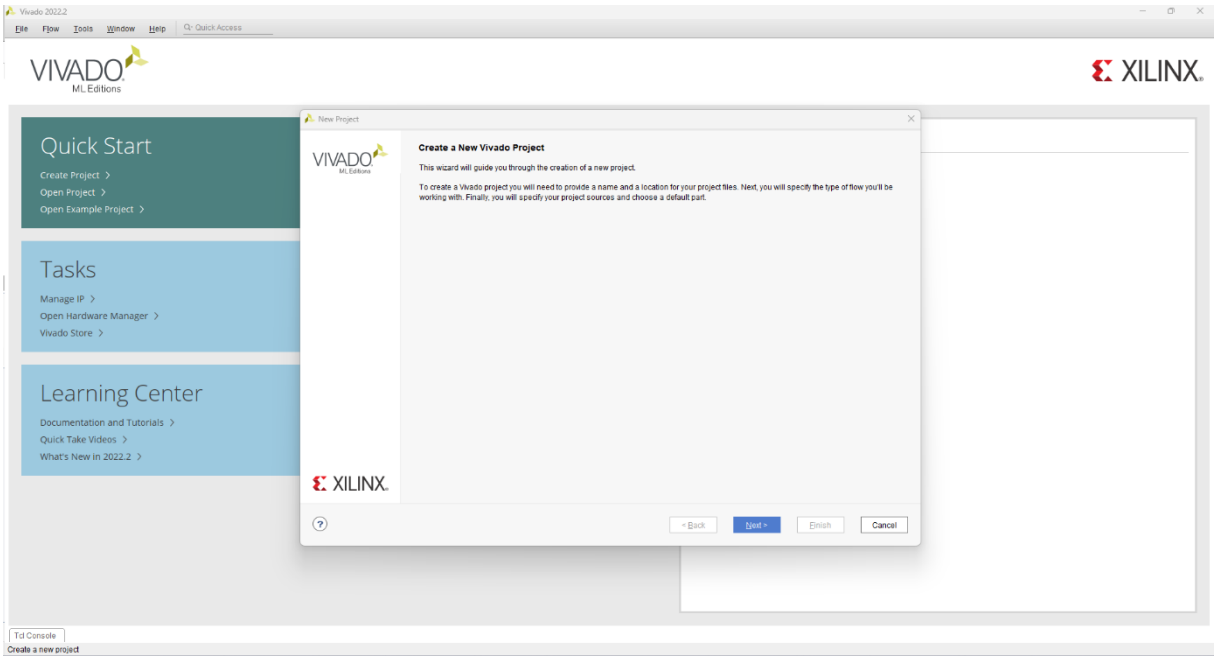


Vue d’ensemble Demo RedPitaya sous Vivado 2022_2

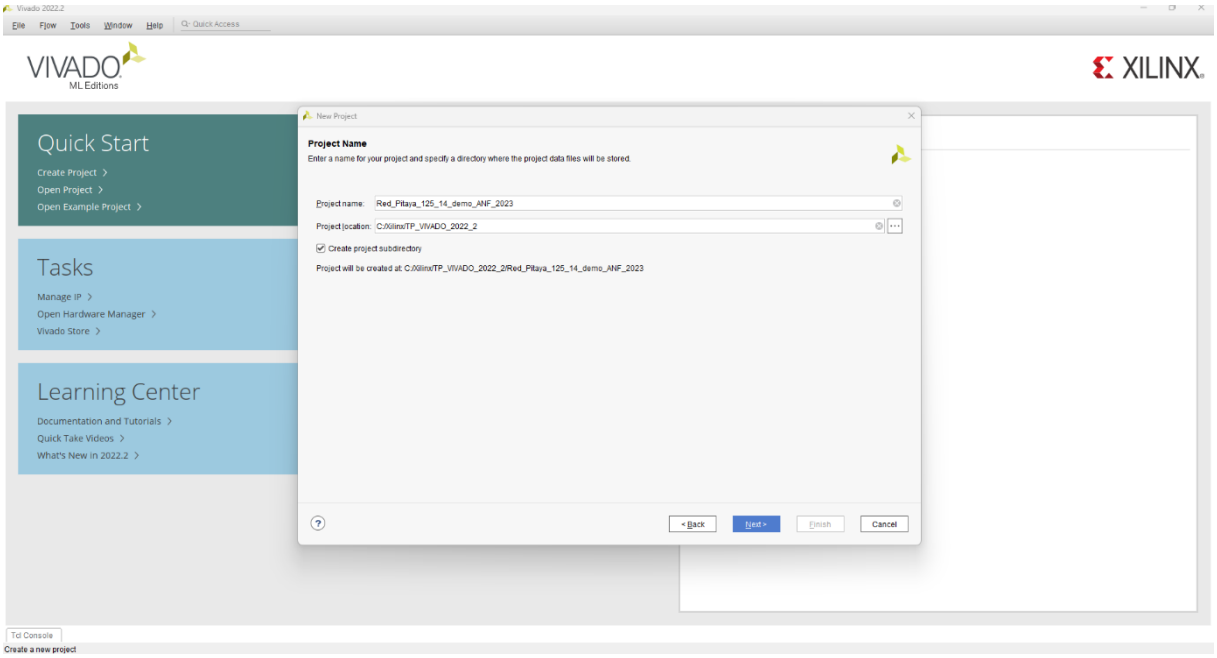
3.2 Environnement de développement Xilinx (VIVADO)

3.3 Création du projet sous Vivado_2022_2 :

Lancer Vivado 2022.2-> **File->Project->New->Create new Vivado project-> Next.**



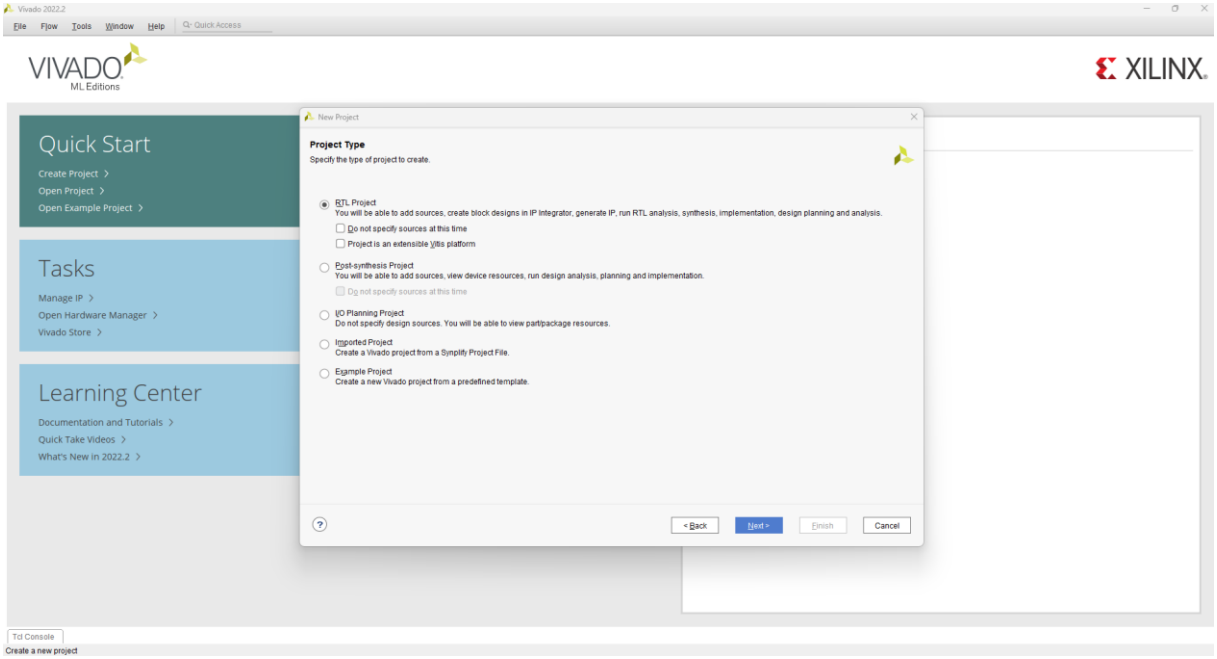
Create new Vivado project-> **project name-> Next.**



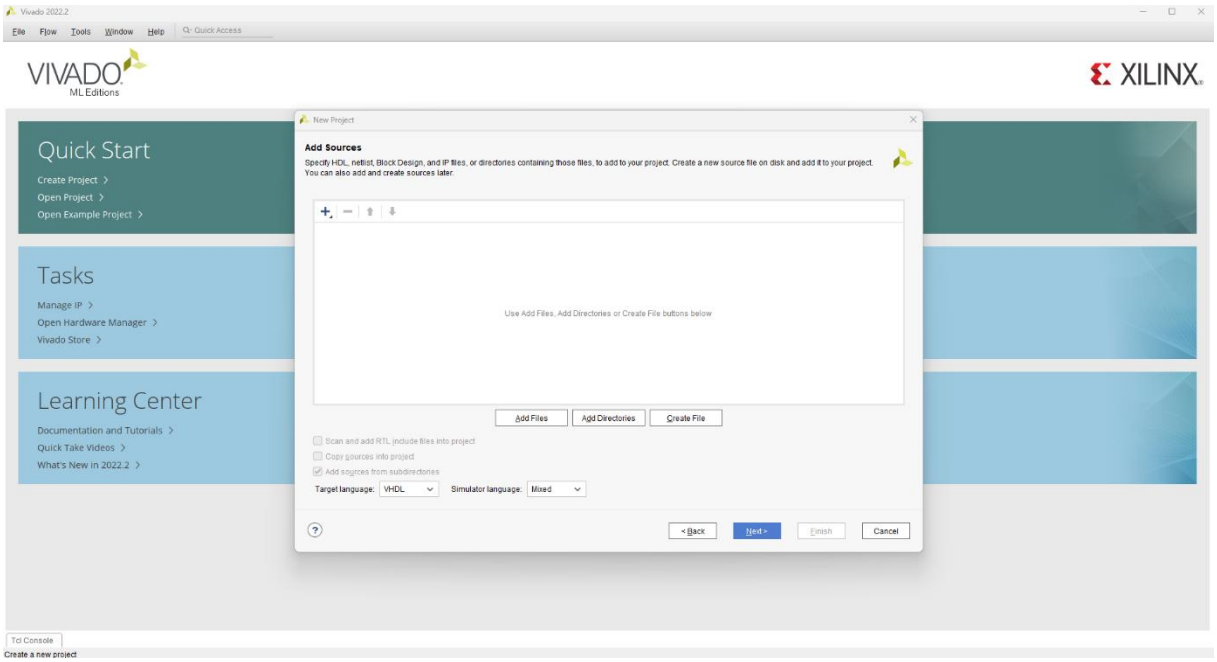
Ici je nomme le projet Red_Pitaya_125_14_demo_ANF_2023

Dans le répertoire Xilinx je créer un dossier-> TP_VIVADO_2022_2

Sélectionner « **RTL Project** » pour « **Projet Type** » puis « **Next** »



Dans **Add Sources et Add Constraint->** On ne spécifie rien**->** sélectionner**-> Next**

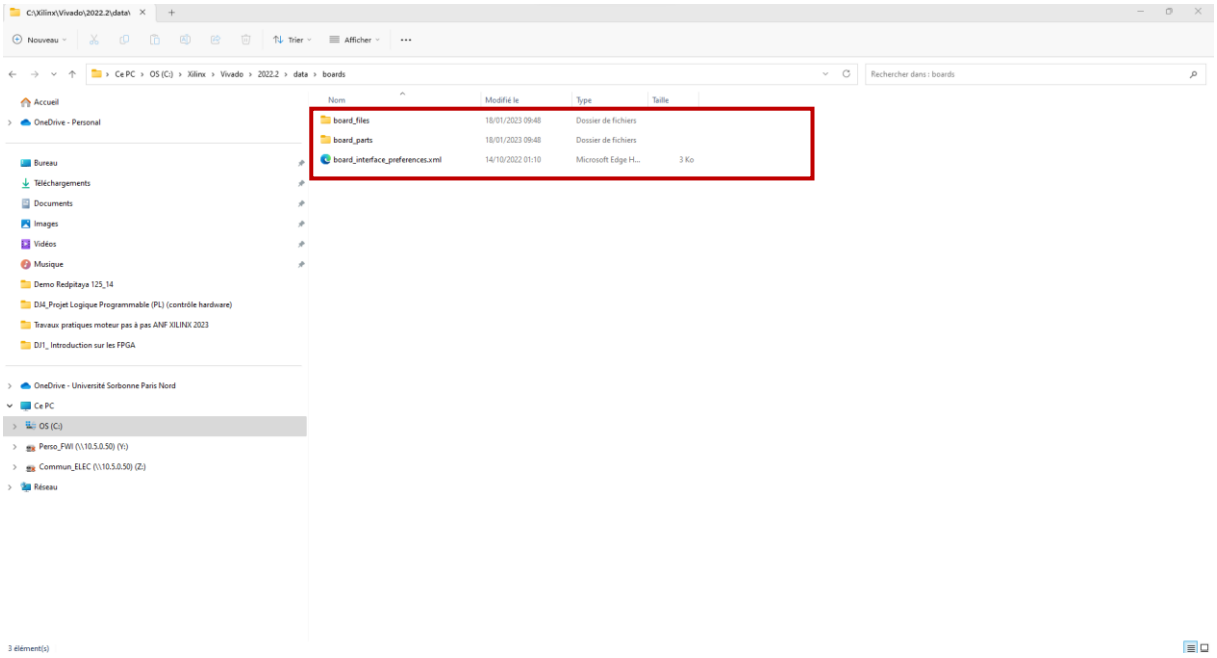


La board Redpitaya n'est pas présente par défaut dans Vivado :

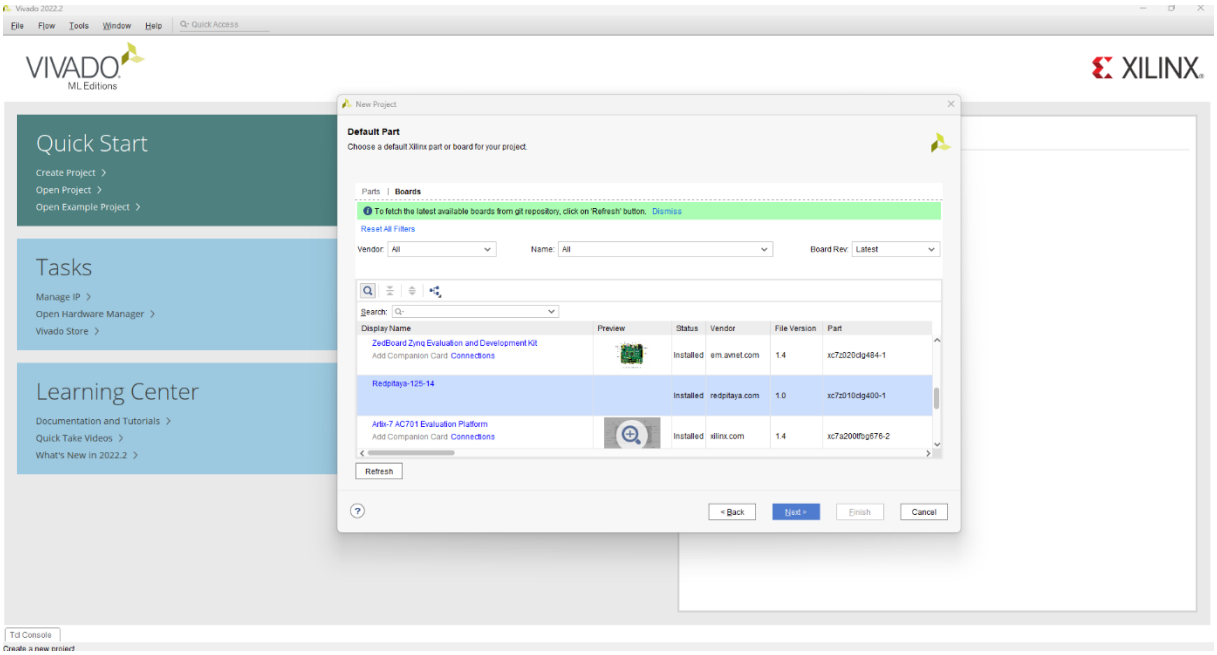
Allez sur le lien ci-dessous pour récupérer redpitaya-125-14/1.0

https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

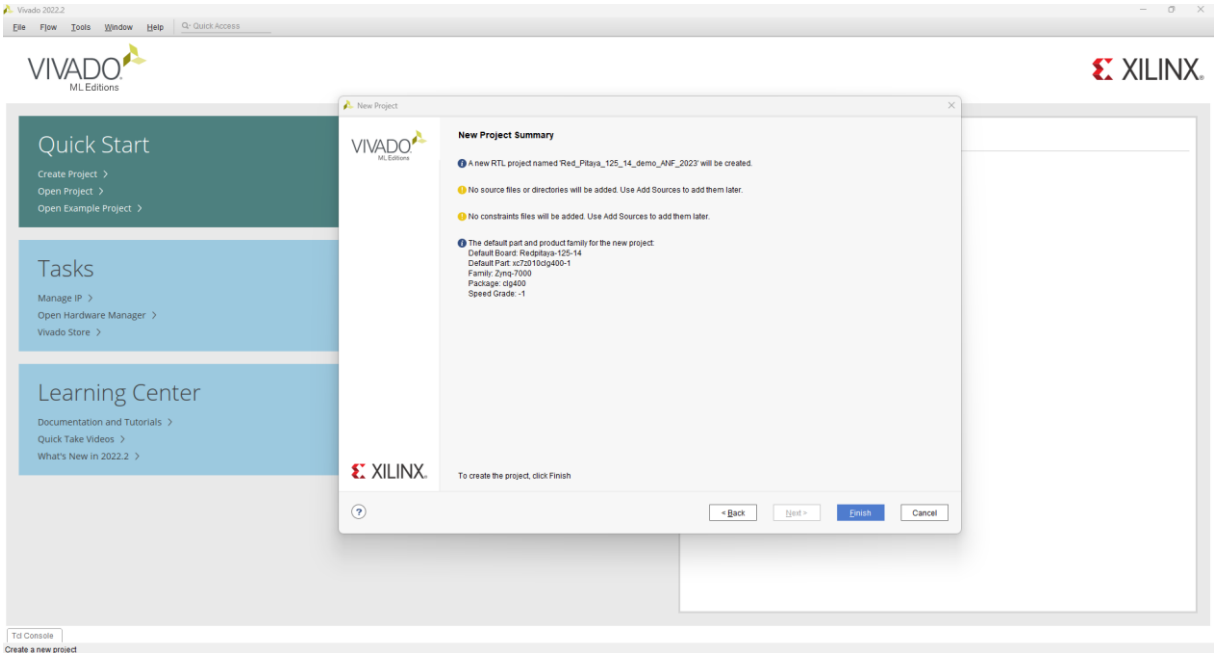
Copier /coller le dossier board_files/redpitaya-125-14/1.0 dont on a besoin dans C:\Xilinx\Vivado\2022.2\data\boards redémarrer Vivado.



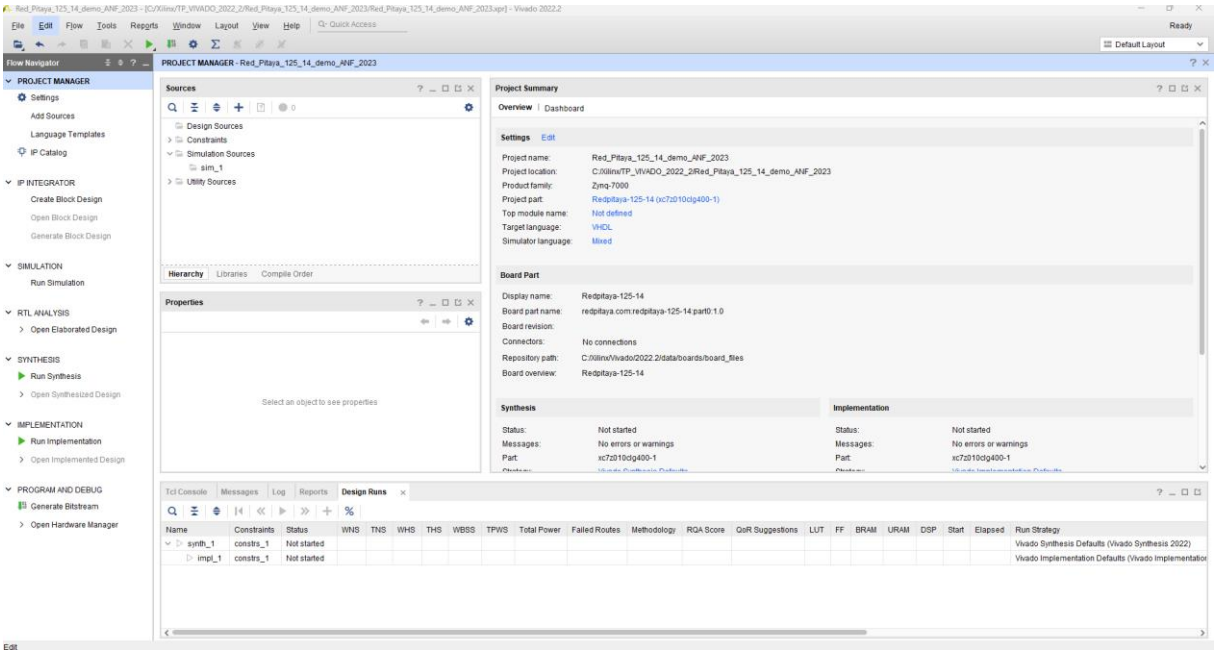
Dans la fenêtre « **Default Part** », sélectionner « **Boards** », puis redpitaya-125-14 dans l'onglet « Display Name ». Appuyer sur « Next »



Dans la fenêtre « **New Project Summary** », cliquez sur « **Finish** » :



L’environnement de Vivado est lancé.

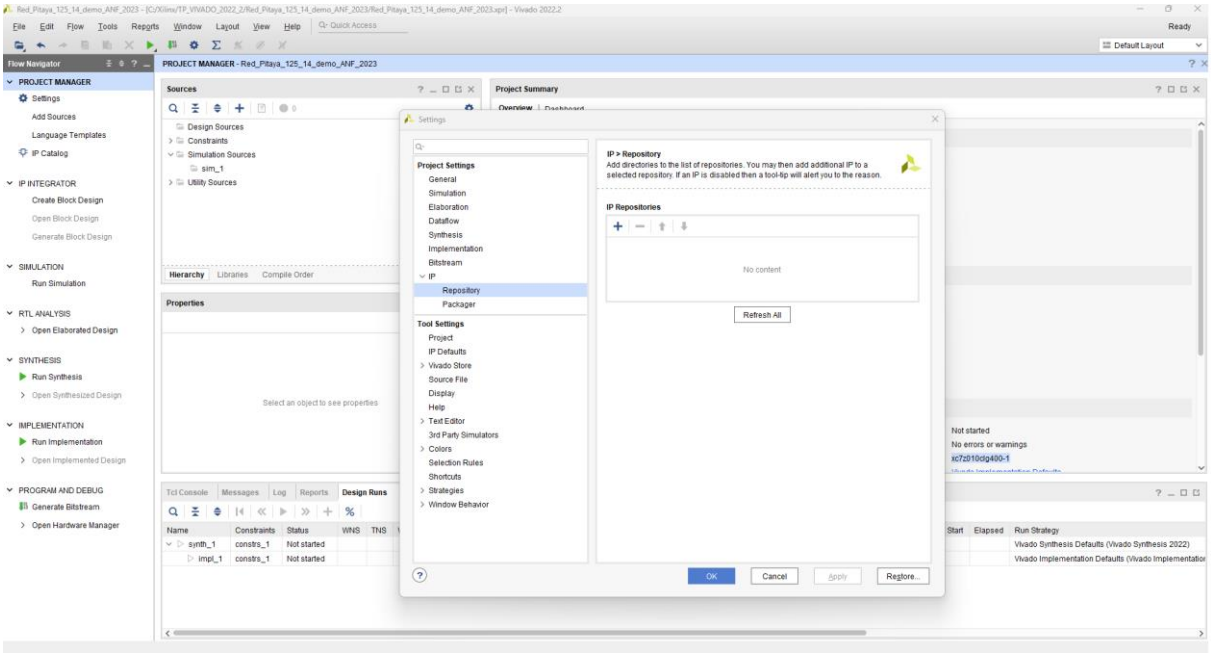


Nous allons récupérer la bibliothèque IP Redpitaya-125-14

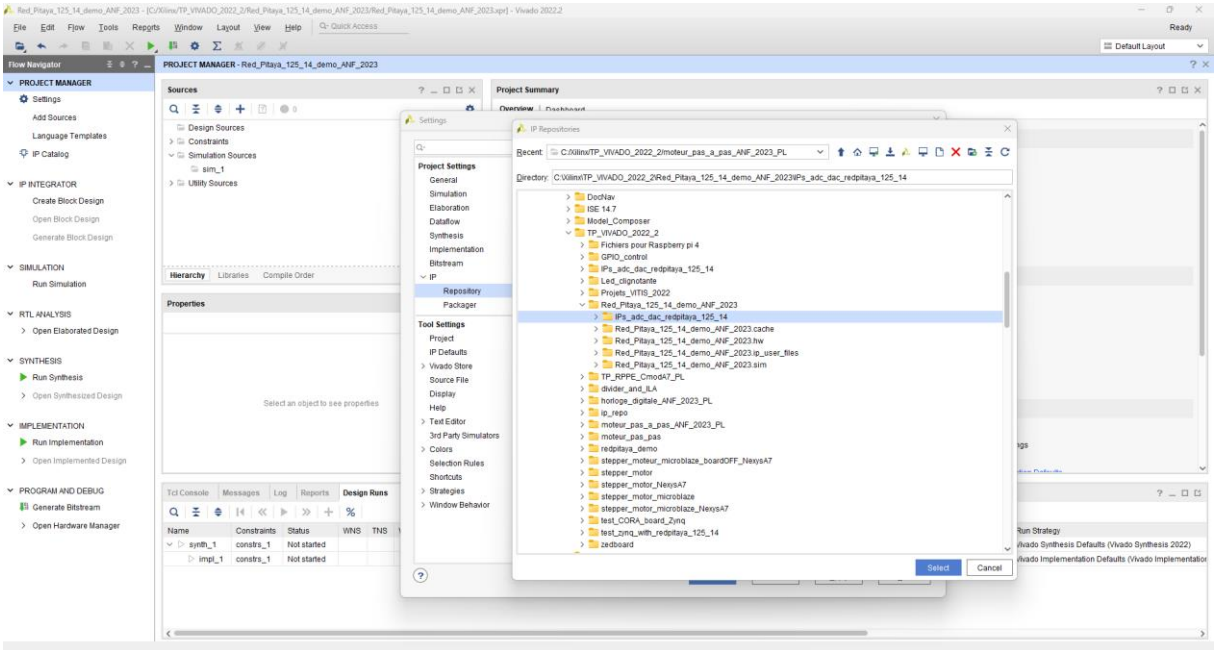
https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

Copier-coller le dossier IPs_adc_dac_redpitaya_125_14 dans le répertoire source de votre projet, pour moi c'est : C:\Xilinx\TP_VIVADO_2022_2\Red_Pitaya_125_14_demo_ANF_2023

Dans l'onglet **PROJECT MANAGER**-> **Settings** -> **IP Repository-> clic sur +-> ajouter le dossier :** IPs_adc_dac_redpitaya_125_14

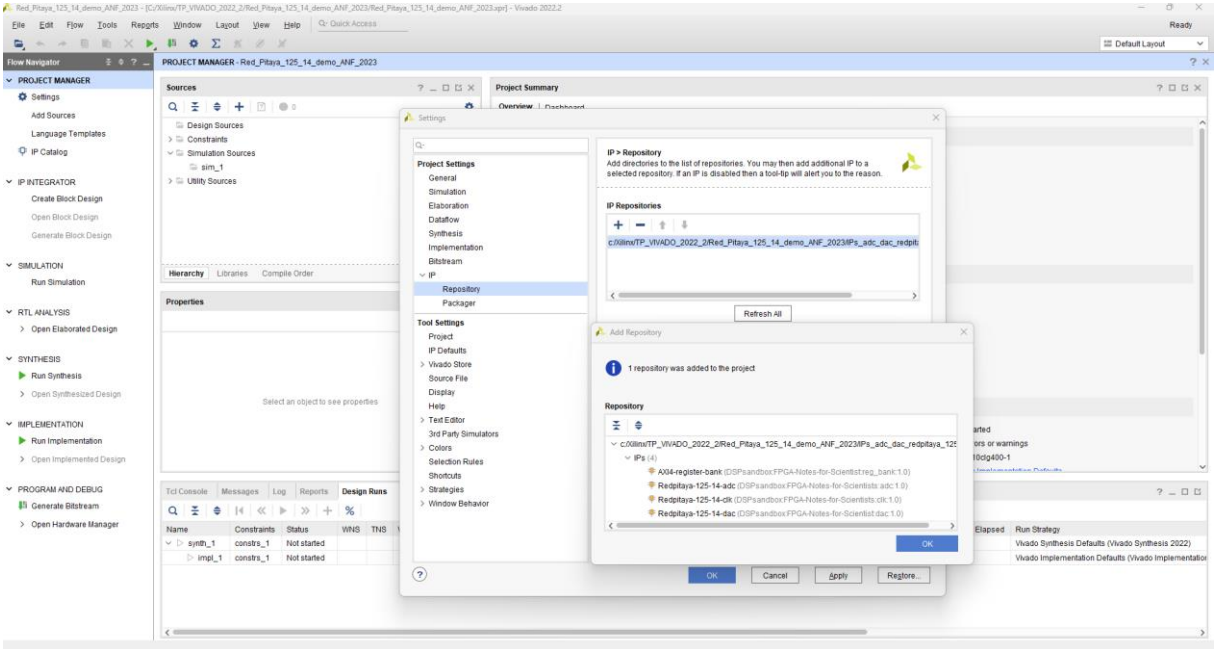


Dans l'onglet IP Repository-> Select-> IP_adc_dac_redpitaya_125_14-> **Select.**



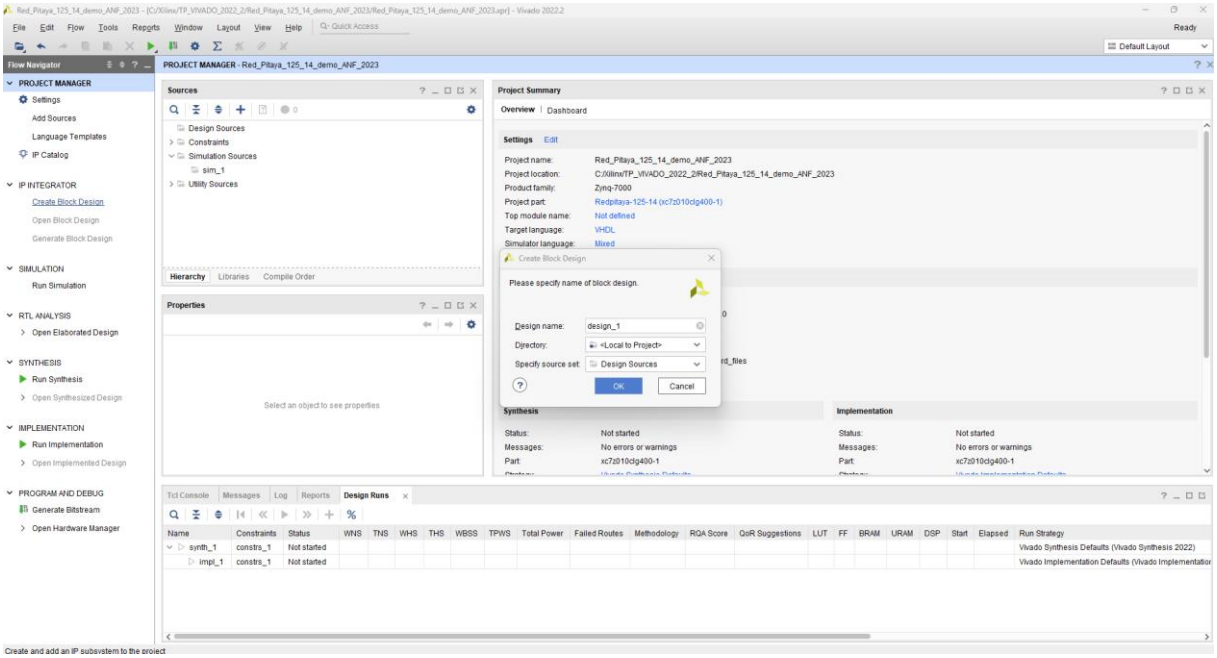
Dossier IPs_adc_dac_redpitaya_125_14 ajouté

Dans Settings-> sélectionner Apply puis **« OK »**

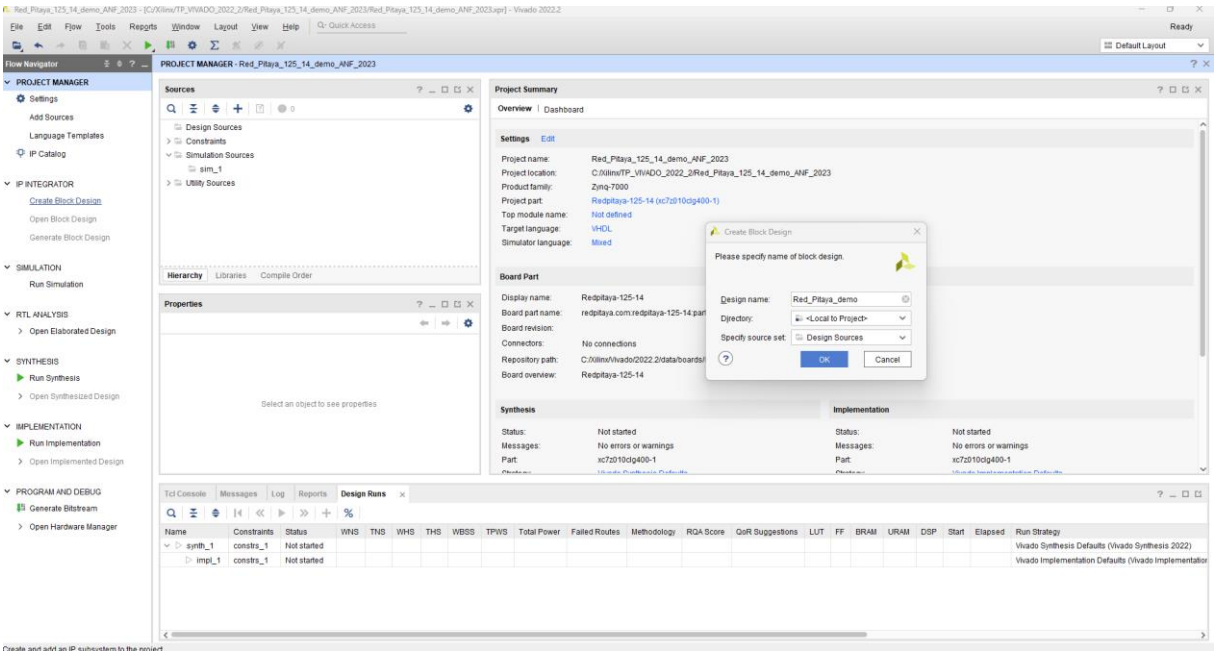


<u>Création du block design :</u>

Dans la fenêtre “**Flow Navigator** » de Vivado à gauche de l’écran, cliquez sur « Create Block Design » de la rubrique « IP Integrator », remplacer le nom donné par défaut au block design par Red_Pitaya_demo



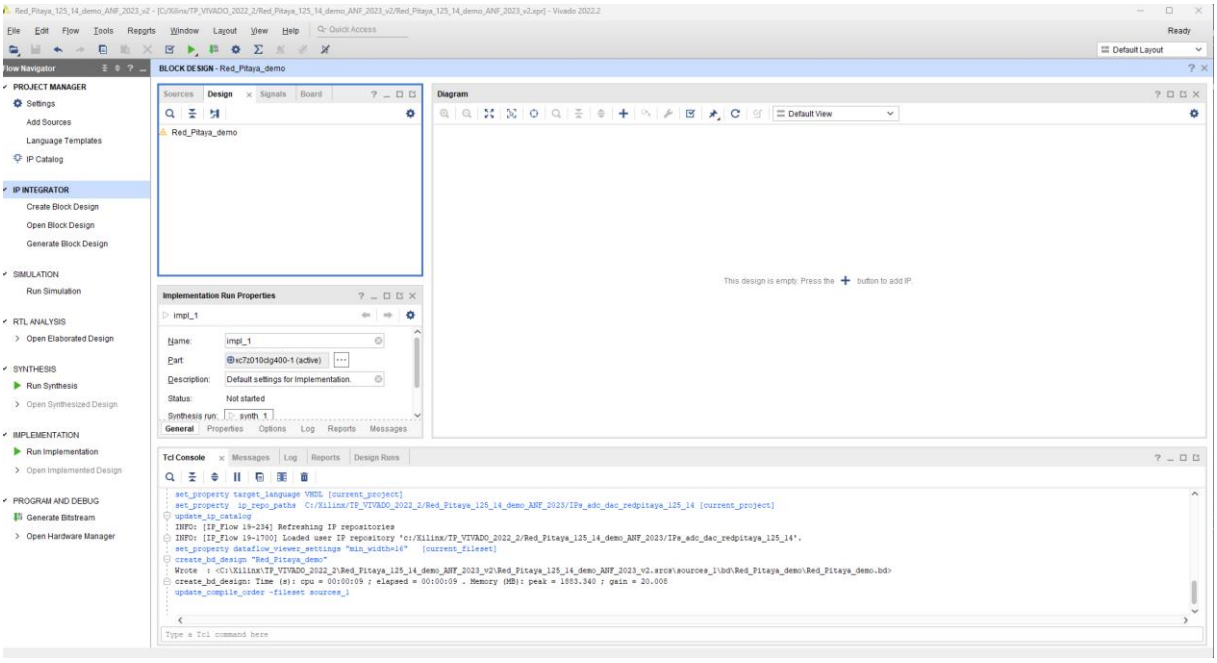
Puis « **OK** »



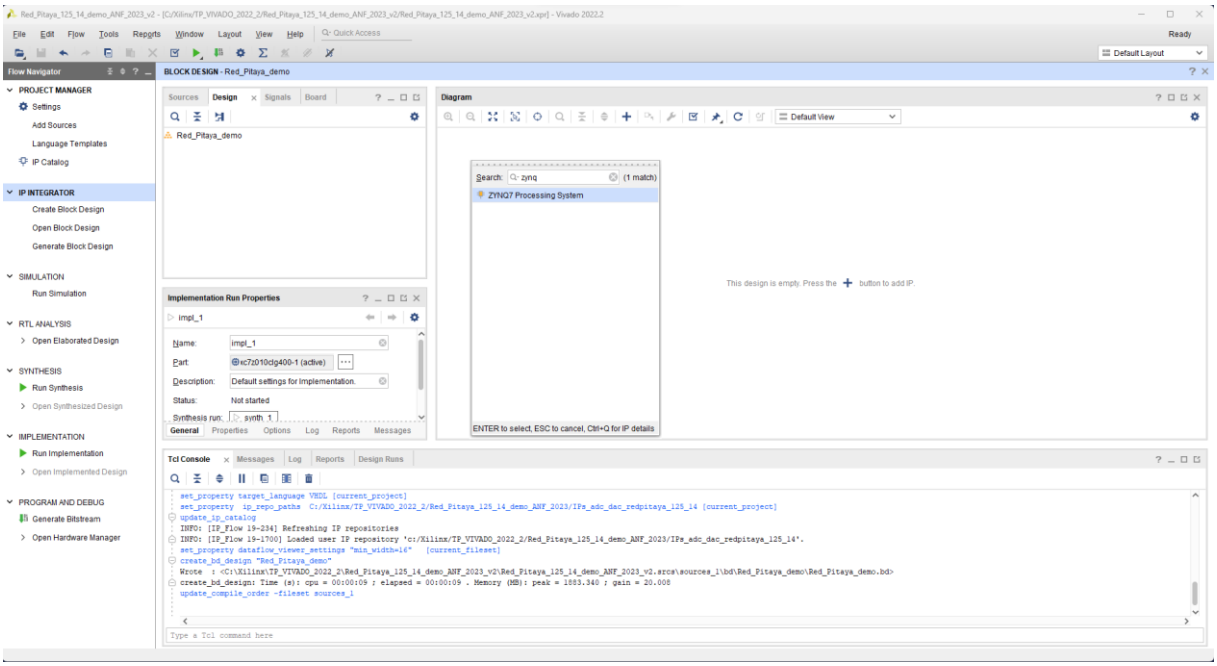
On va insérer les IP Processeur Zynq et Processeur Reset, ainsi que les IPs_adc_dac_redpitaya_125_14 sur le **Diagram** :

- **Le Processeur Zynq** (Système de traitement ZYNQ7)
- **Le Processeur Reset** (Réinitialisation du système du processeur)

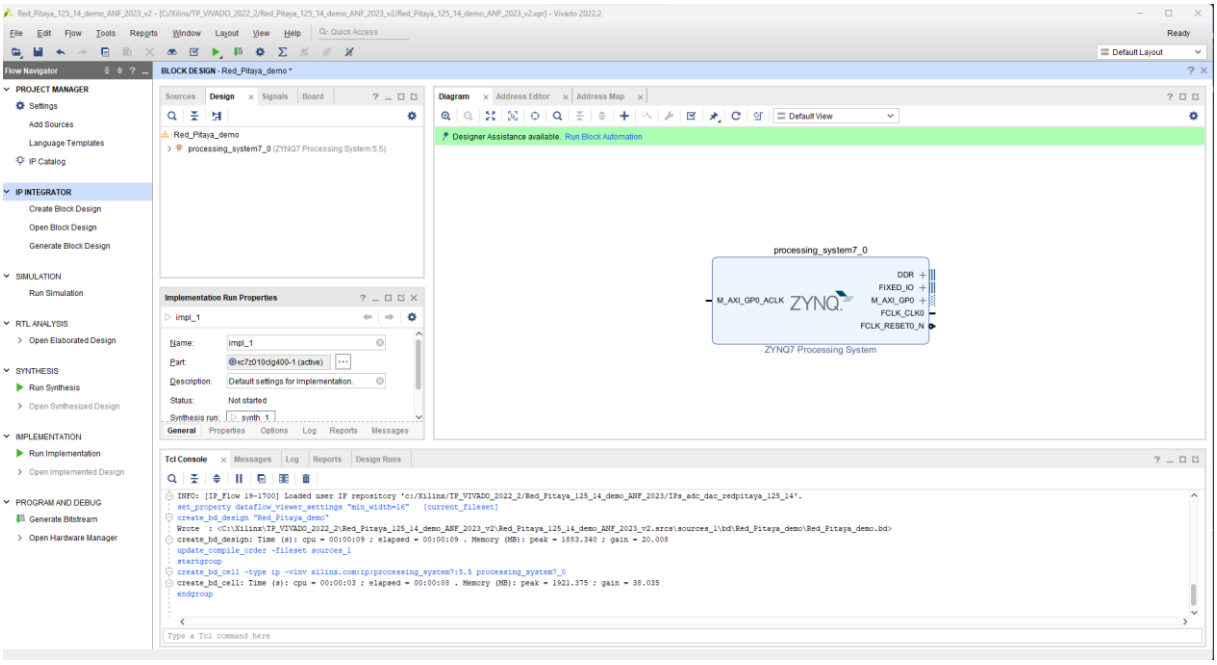
- AXIGPIO (L'AXI GPIO fournit une interface d'entrée/sortie à usage général) l'interface AXI)
- Redpitaya-125-14-clk
- Redpitaya-125-14-adc (les échantillons ADC 14 bits sont alignés MSB dans les interfaces de flux AXI 16 bits sortantes)
- Redpitaya-125-14-dac (les échantillons DAC 14 bits sont extraits du MSB des interfaces de flux AXI 16 bits entrantes)



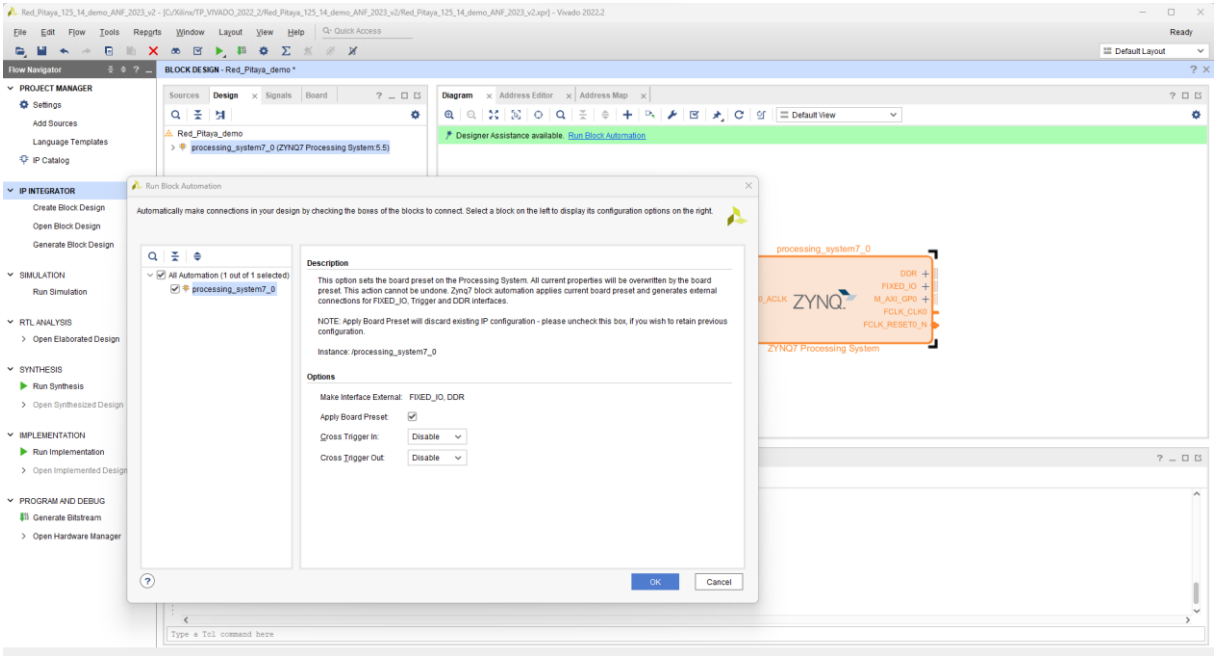
Dans **Diagram** -> clic sur + tapez dans la barre de recherche -> **Zynq**

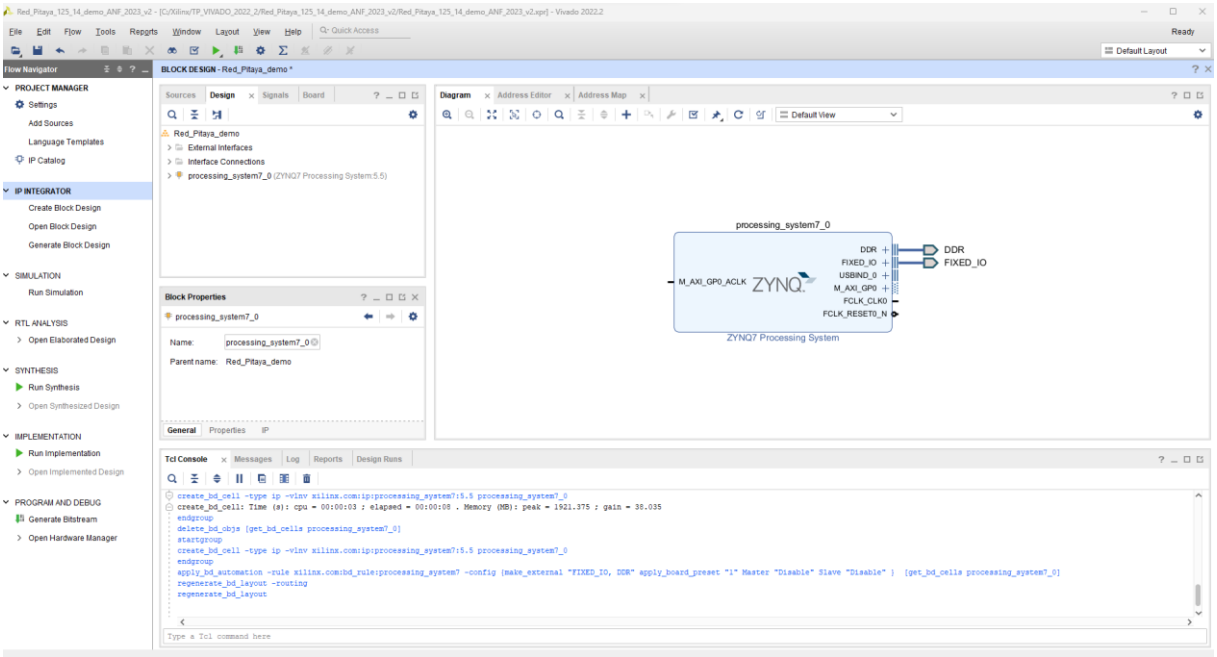


Touche **entrée** -> le processeur apparait sur le Diagram

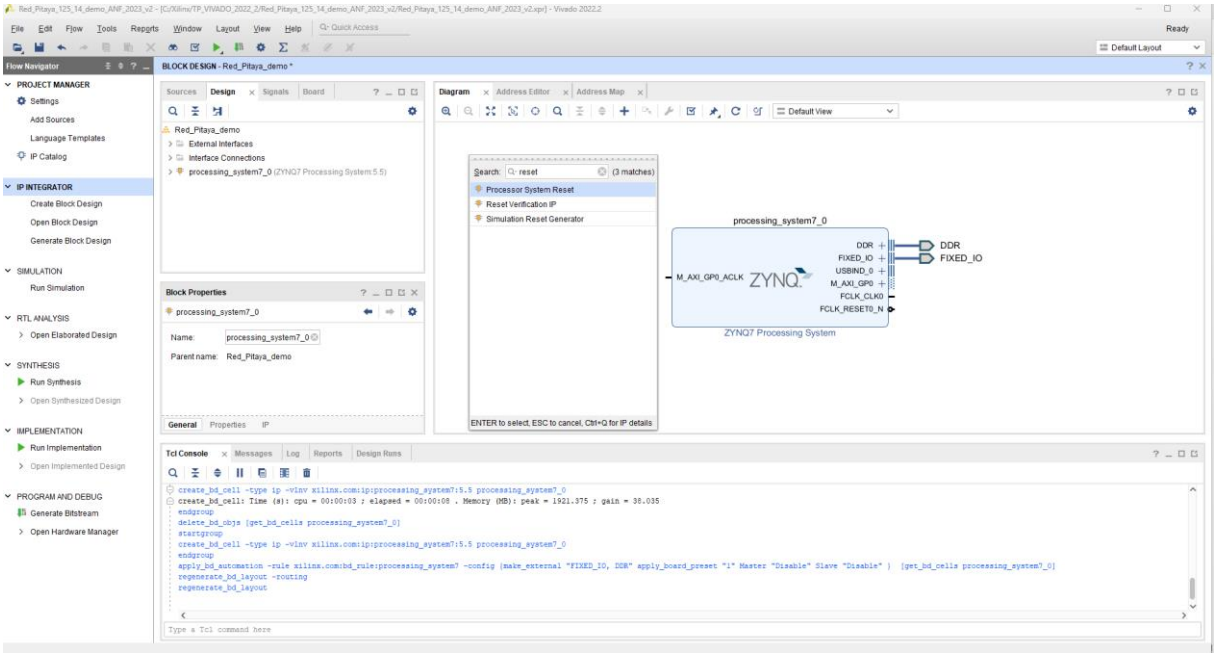


Clic sur **Run Block Automation->** puis **« OK »**

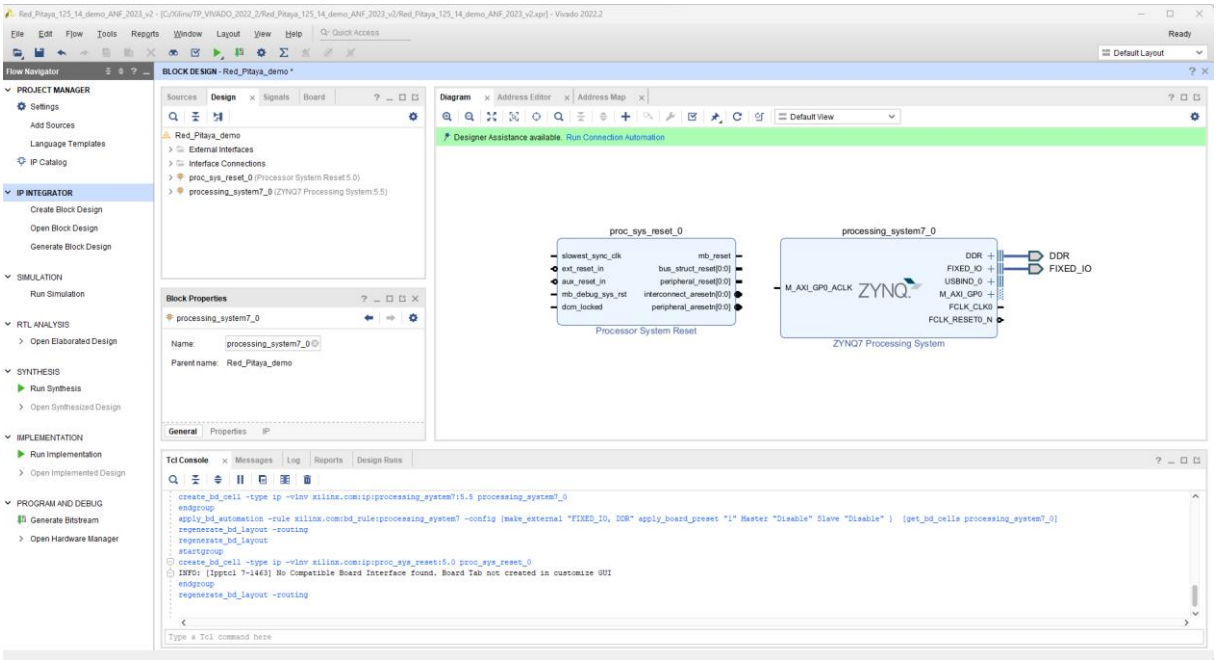




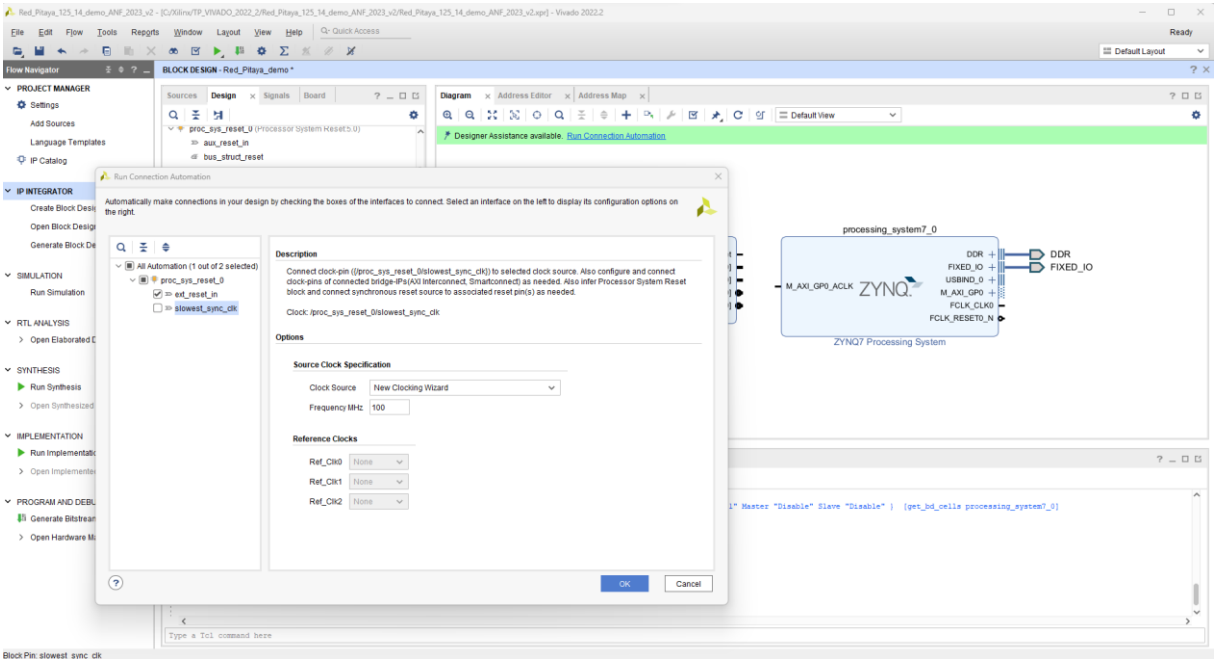
Dans **Diagram** -> clic sur + tapez dans la barre de recherche -> **processor system reset**

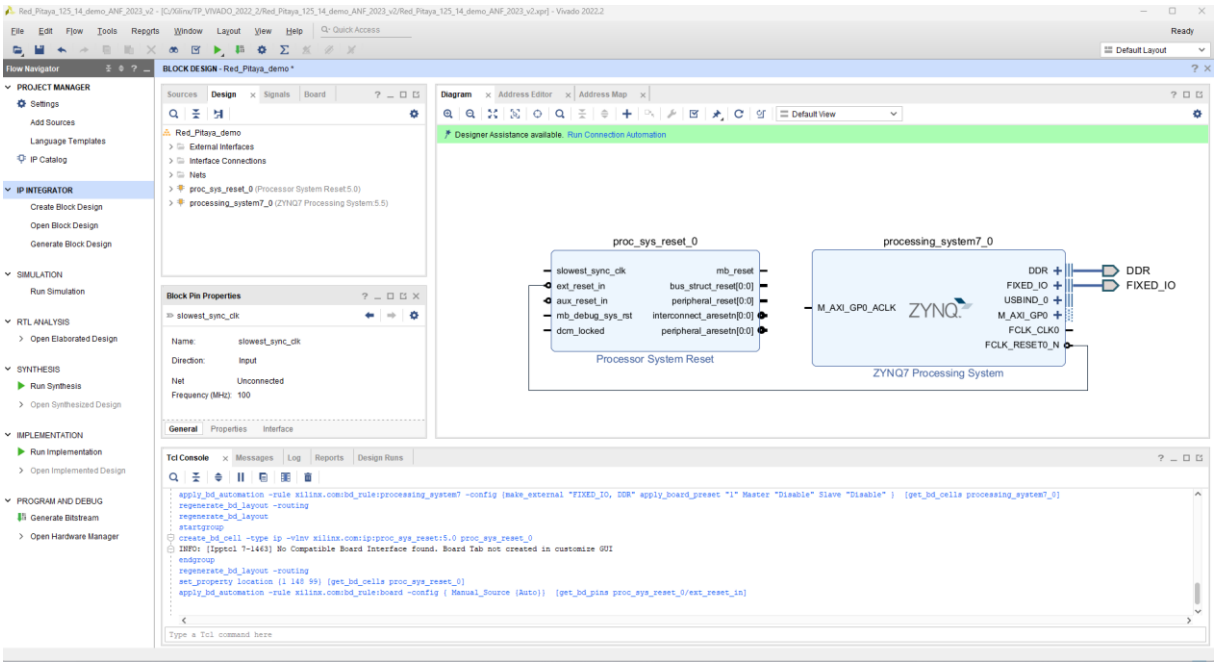


Touche **entrée** -> le processeur system reset apparait sur le Diagram

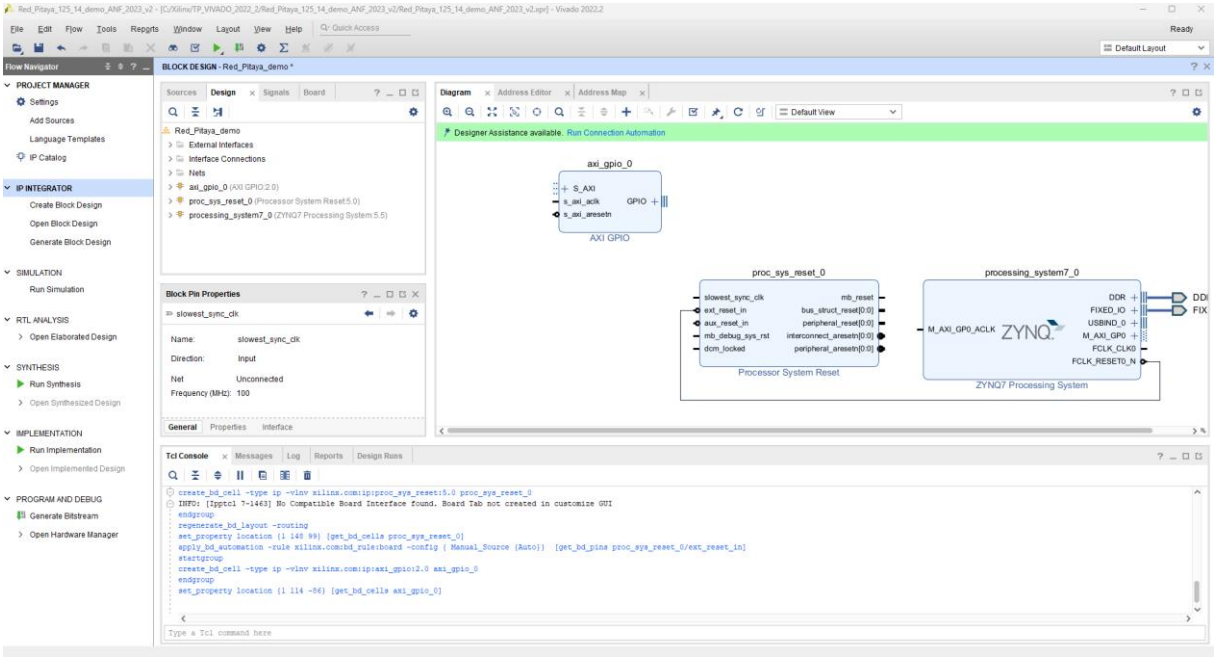


Dans **Run Connection Automation**-> sélectionnez **ext_reset_in** et **new Clocking Wizard->** puis **« OK »**

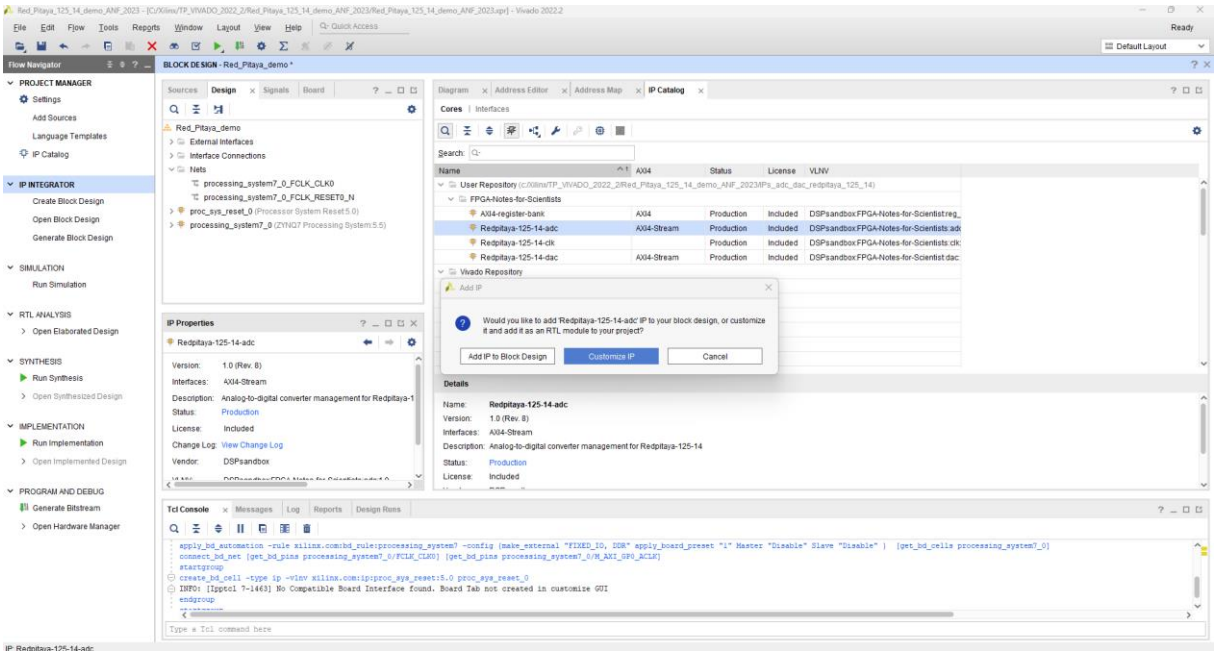




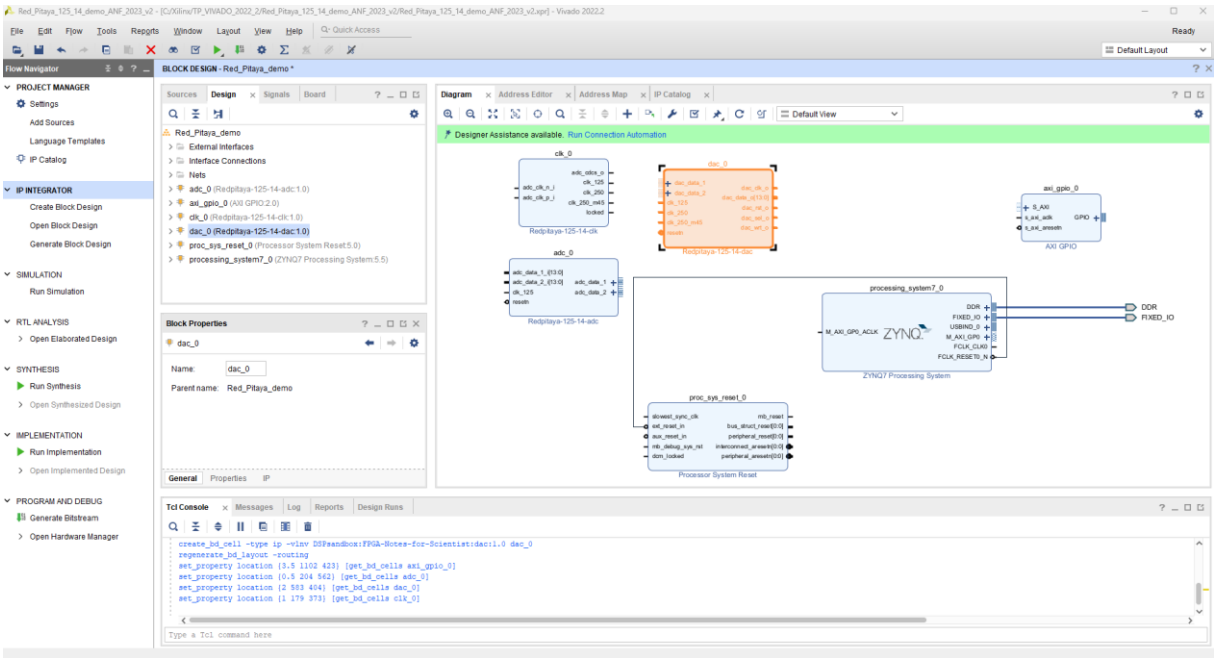
Dans **Diagram** -> clic sur + tapez dans la barre de recherche -> **AXI GPIO**



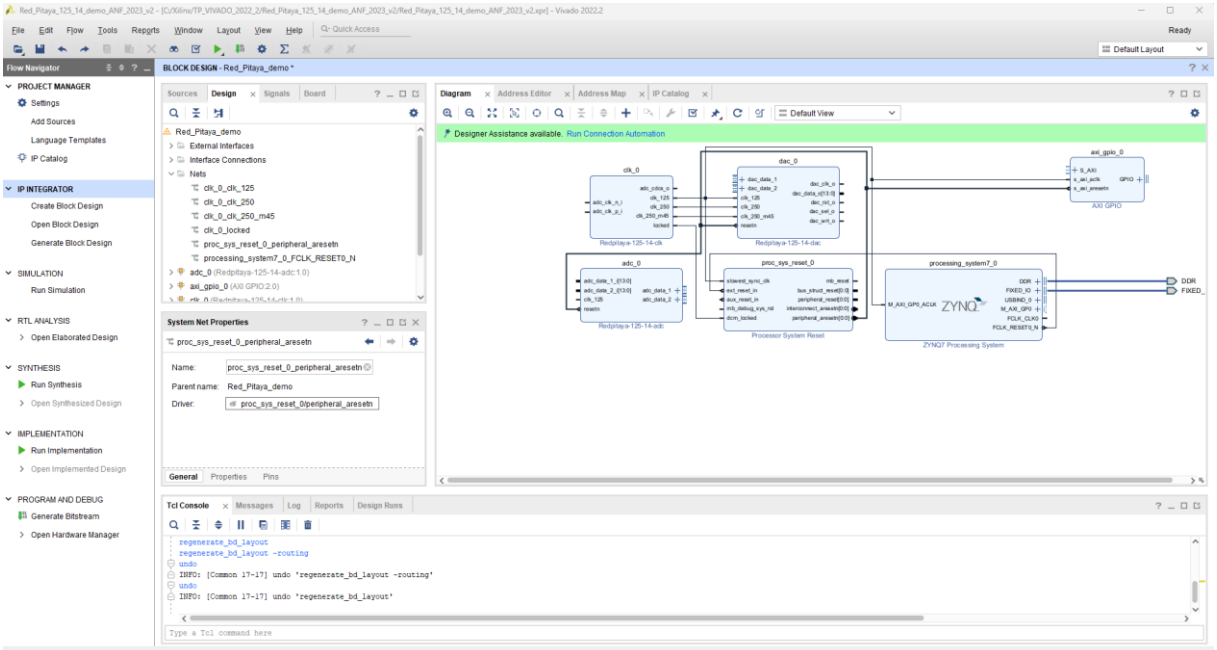
Dans **PROJECT MANAGER**-> **IP Catalog**-> sélectionnez **User_Repository-> Add IP to Block Design /** On ajoute les trois IP sur le Block Design



Les IP ADC DAC et CLK apparaissent

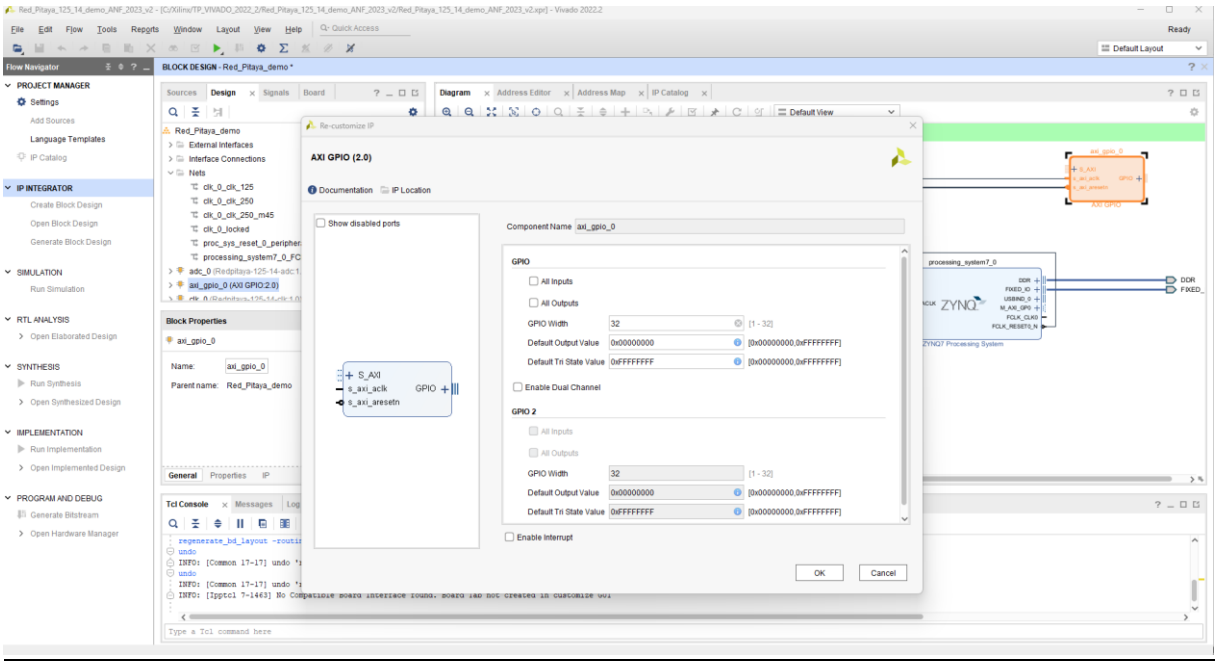


Nous utilisons *clk_125*, qui fonctionne à 125 MHz, pour piloter la logique principale de la conception. Tous les chemins de données de ce didacticiel seront synchrones avec cette horloge. Connectez les horloges et les réinitialisations comme indiqué dans le Diagram ci-dessous.

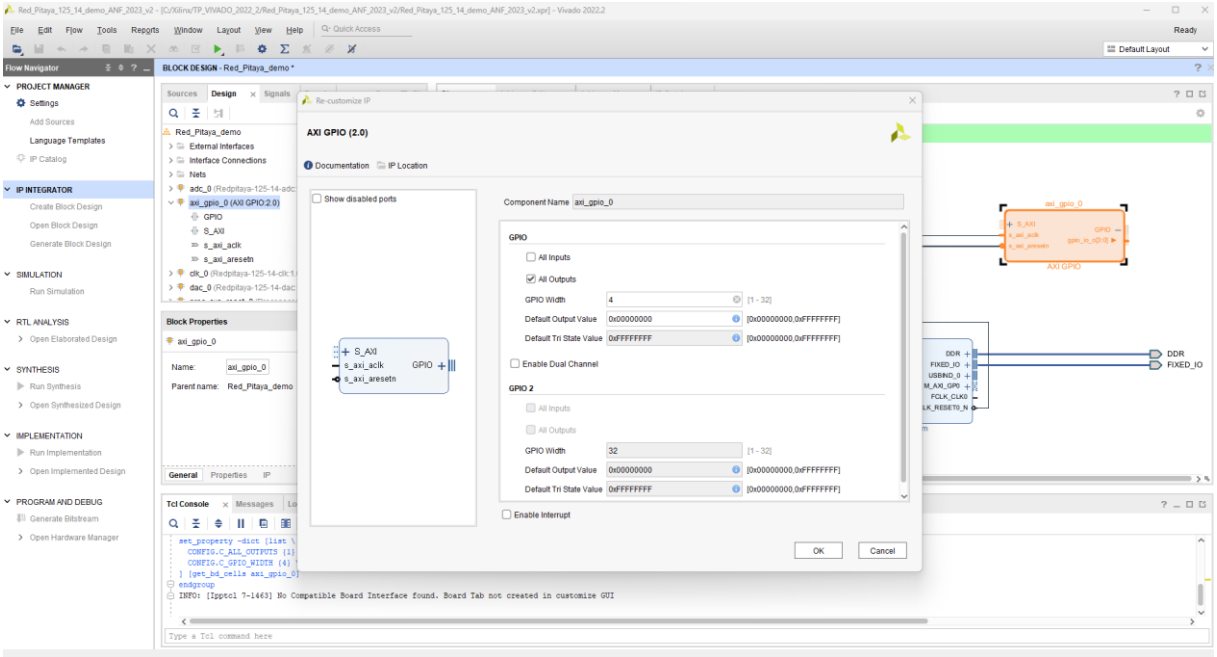


Configurez l'AXI GPIO pour avoir une sortie 4 bits qui correspond aux Leds de la RedPitaya que l'on va utiliser pour tester l'Echo Ethernet

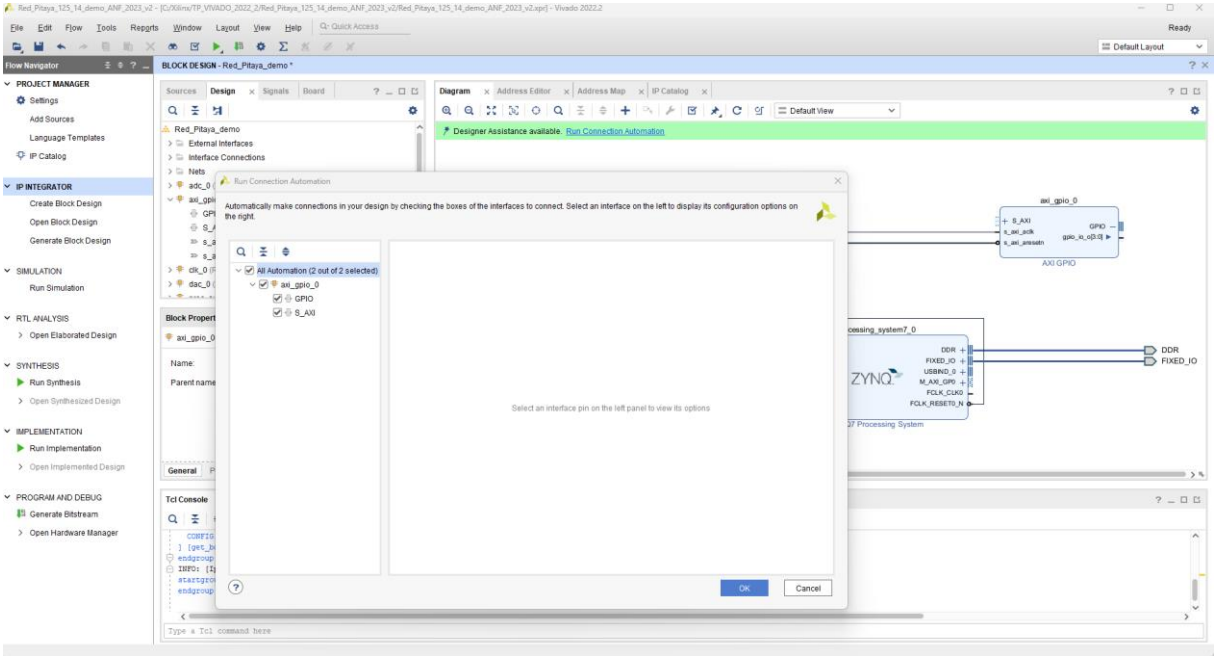
Double clic sur AXI_gpio_0



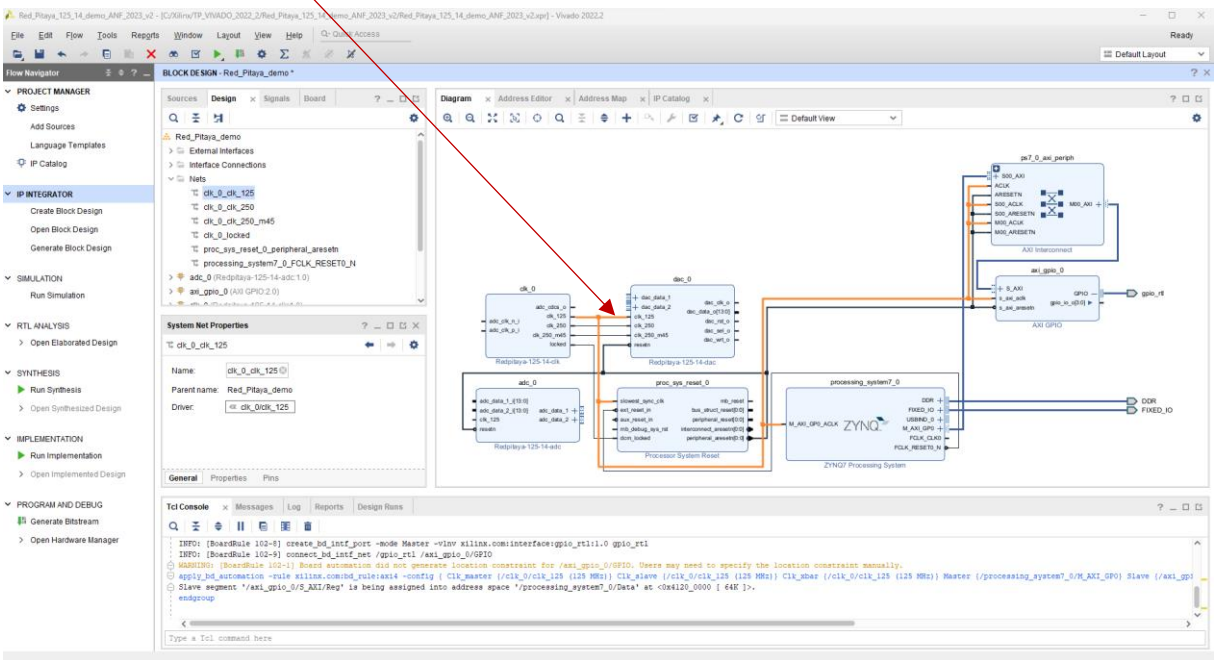
Sélectionnez GPIO et GPIO Width = 4 puis « OK »



Clic sur **Run Connection Automation->** puis **« OK »**



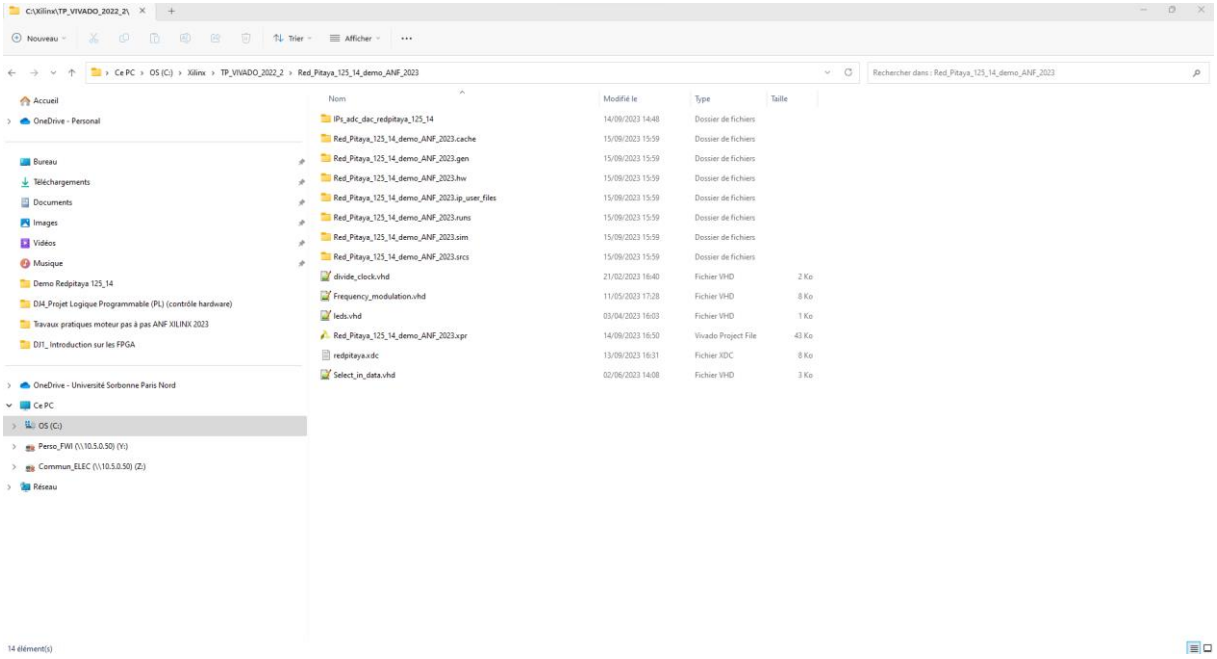
Il nous reste à câbler manuellement comme ci-dessous les lignes d'horloges.



Nous allons importer les IP « maison » du projet : récupérer **divide_clock.vhd**

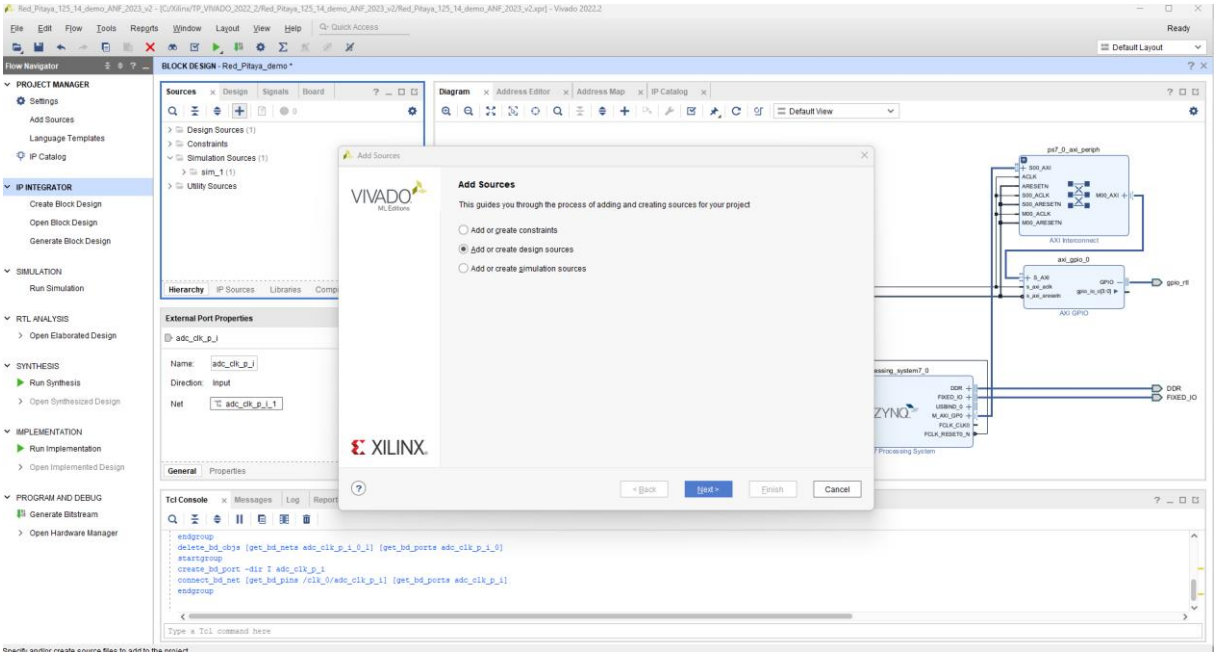
Frequency_modulation.vhd, **leds.vhd**, **Select_in_data.vhd a**vec le lien ci-dessous : https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

Copier-coller dans le répertoire du projet.

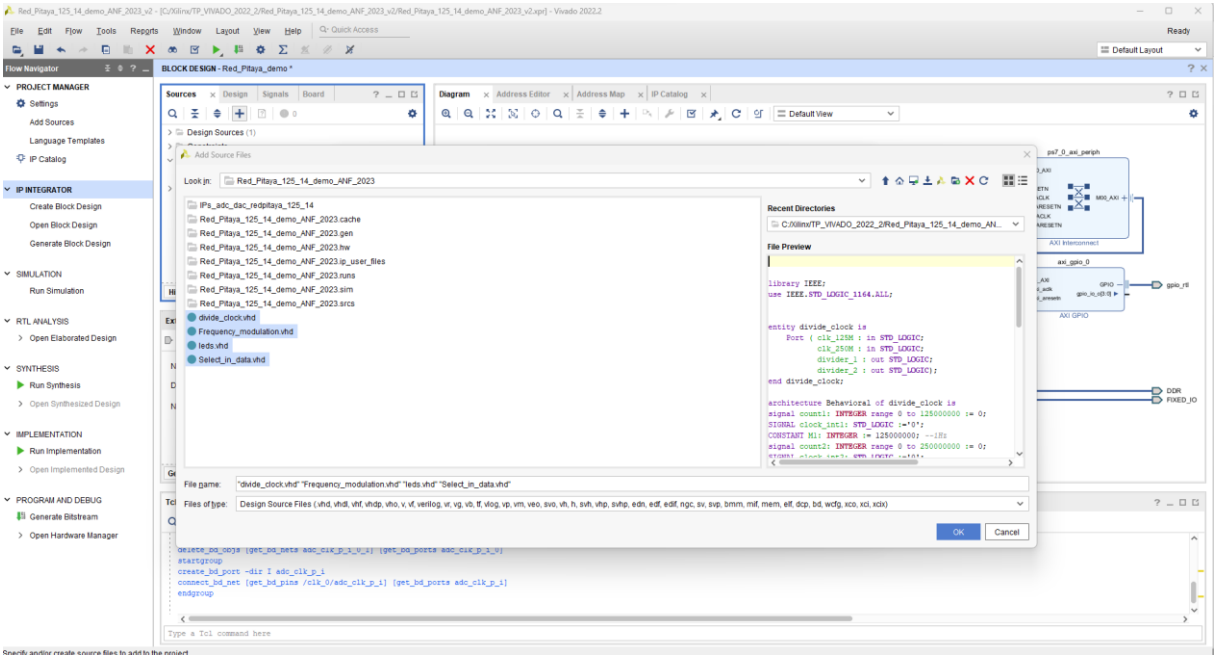


Nous allons les intégrer à notre projet dans le Diagram RedPitaya_demo

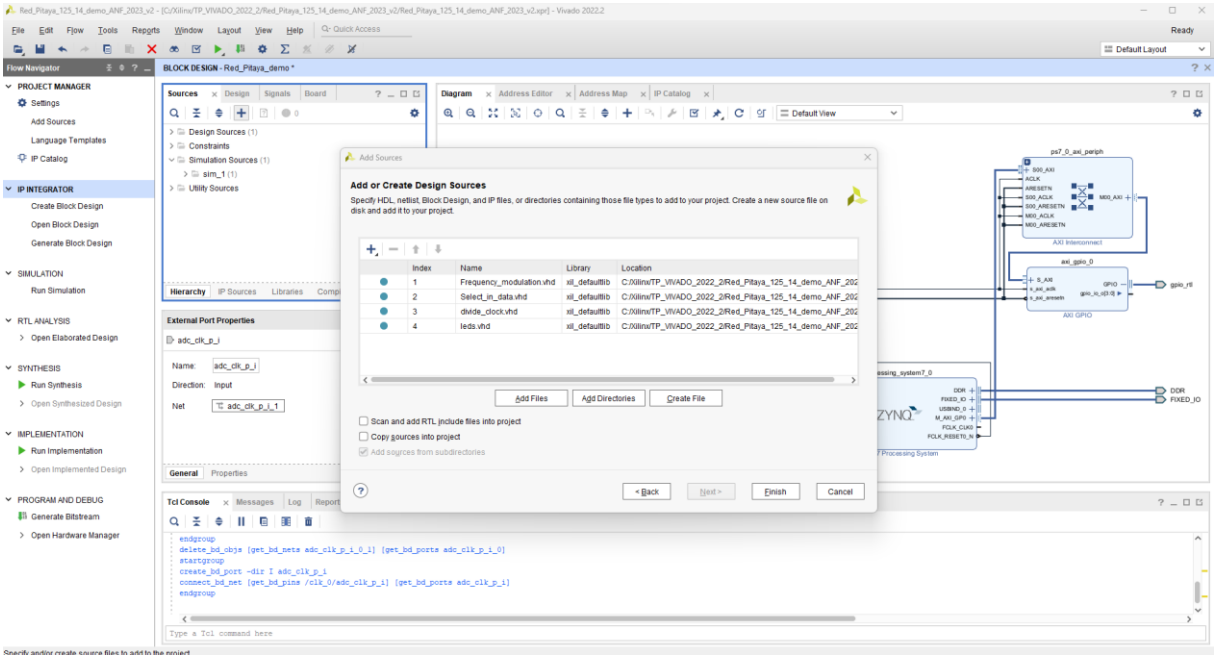
Dans **BLOCK DESIGN**-> clic sur **+** -> **Add or create sources.**



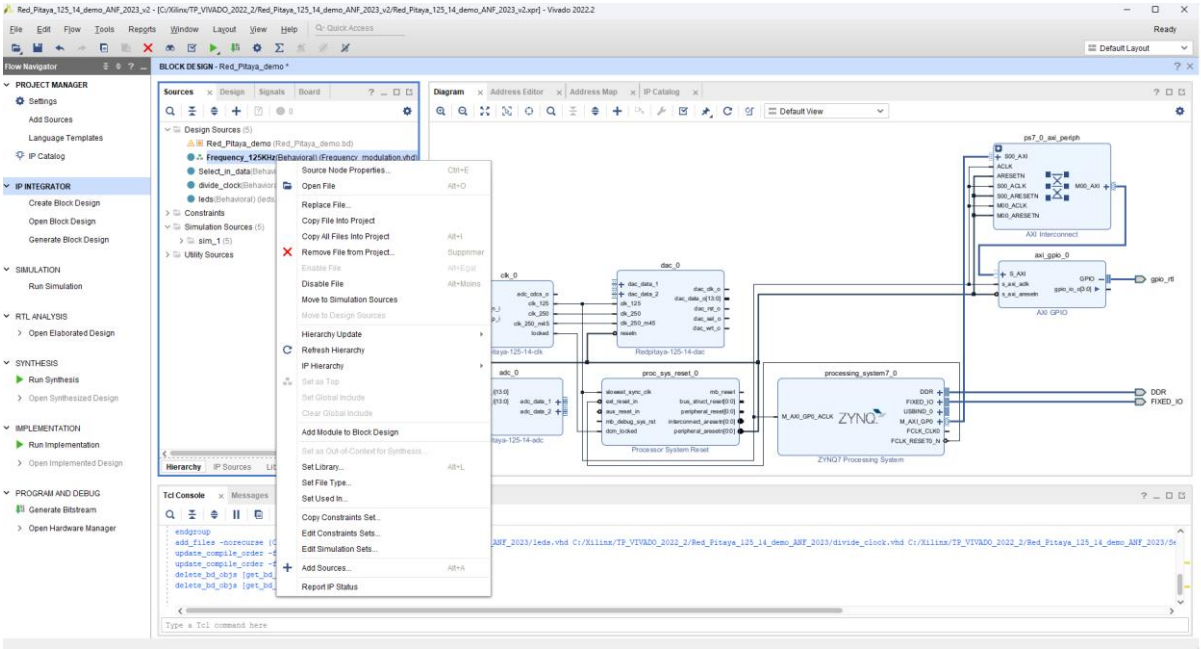
On sélectionne les fichiers VHDL puis « **OK** »



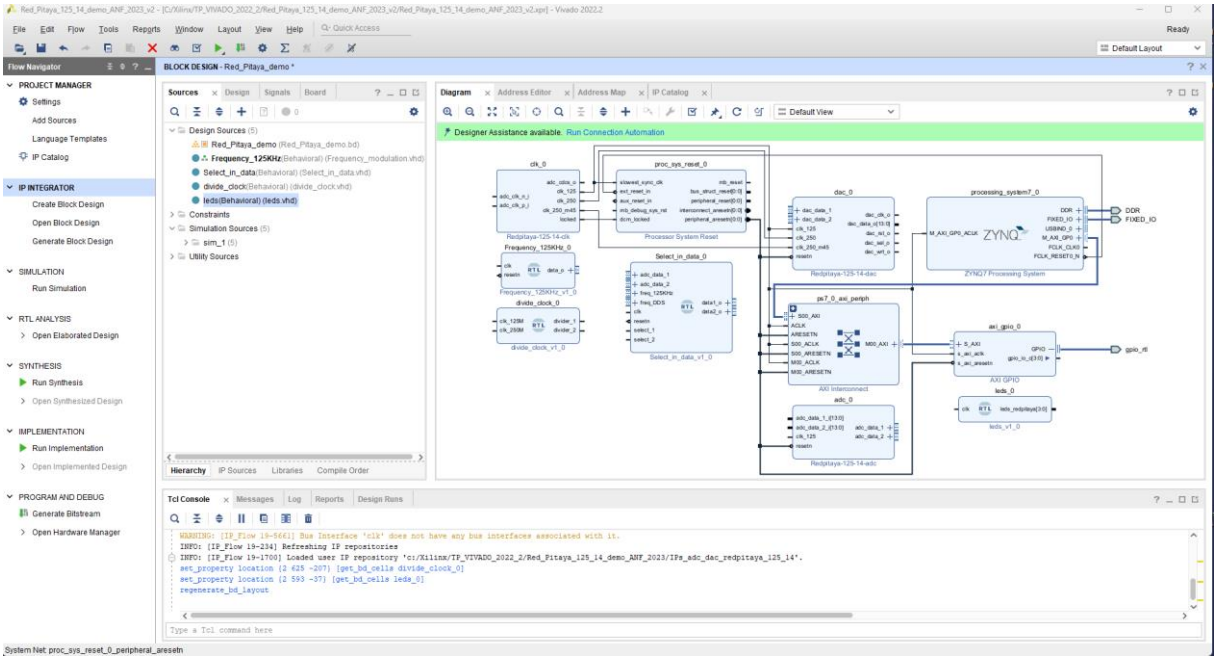
Puis « **Finish** »



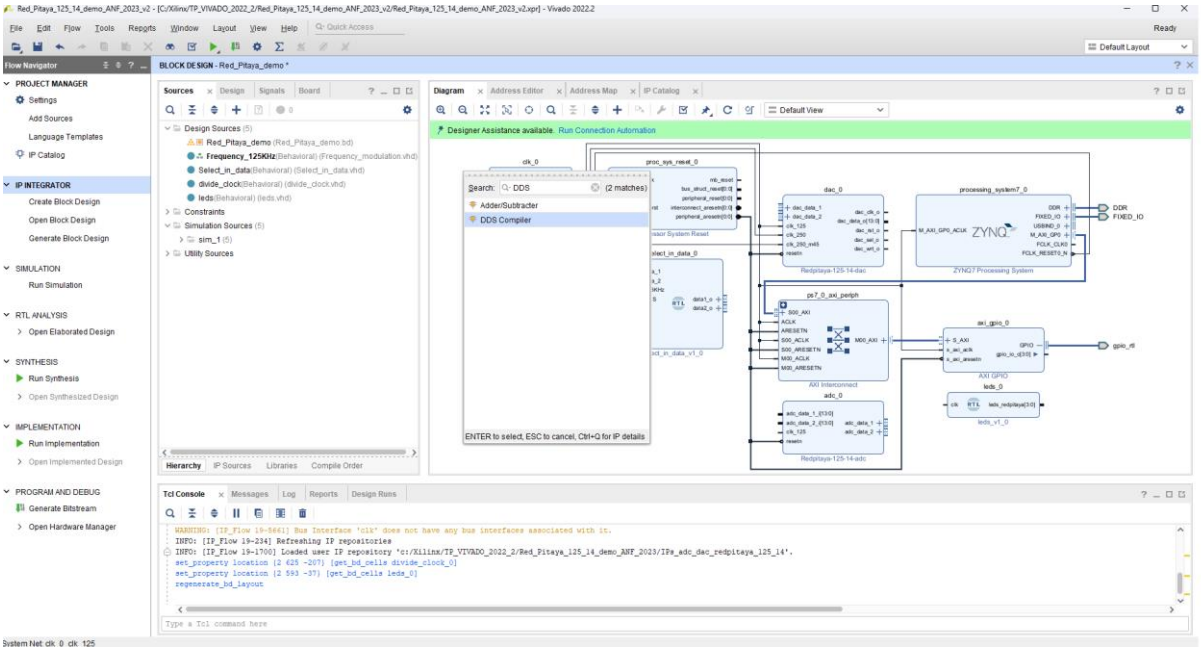
Clic droit sur chaque fichier VHDL puis-> **Add Module to Block Design**



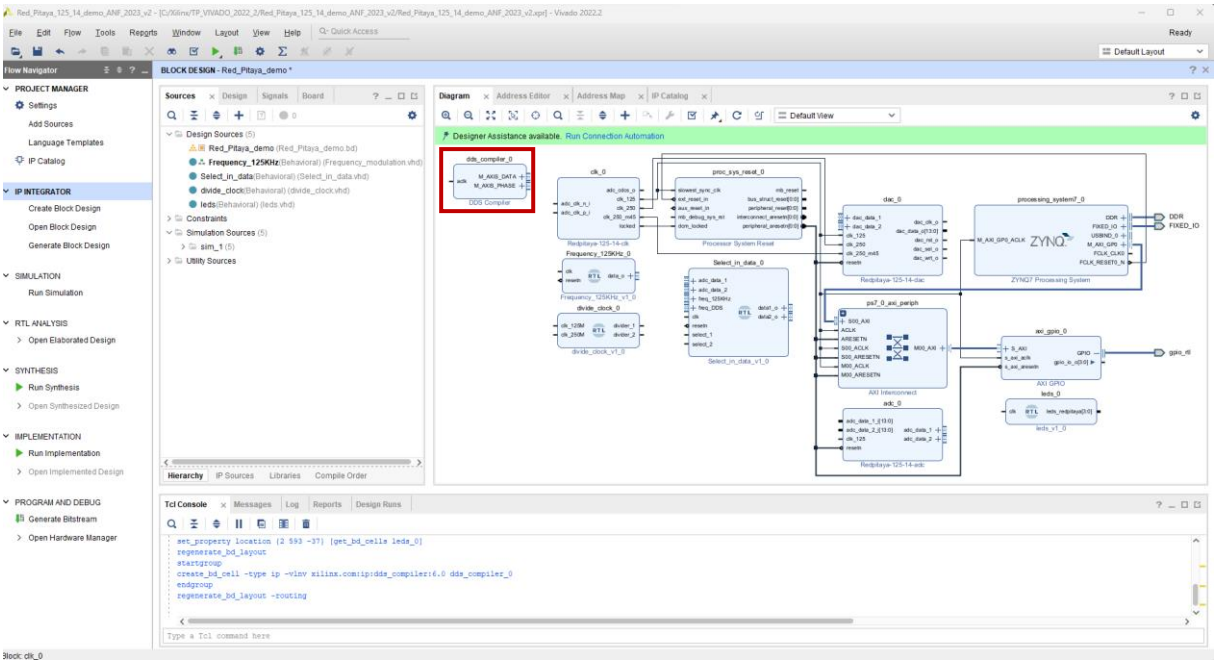
Les modules insérés sur le **Diagram** du projet



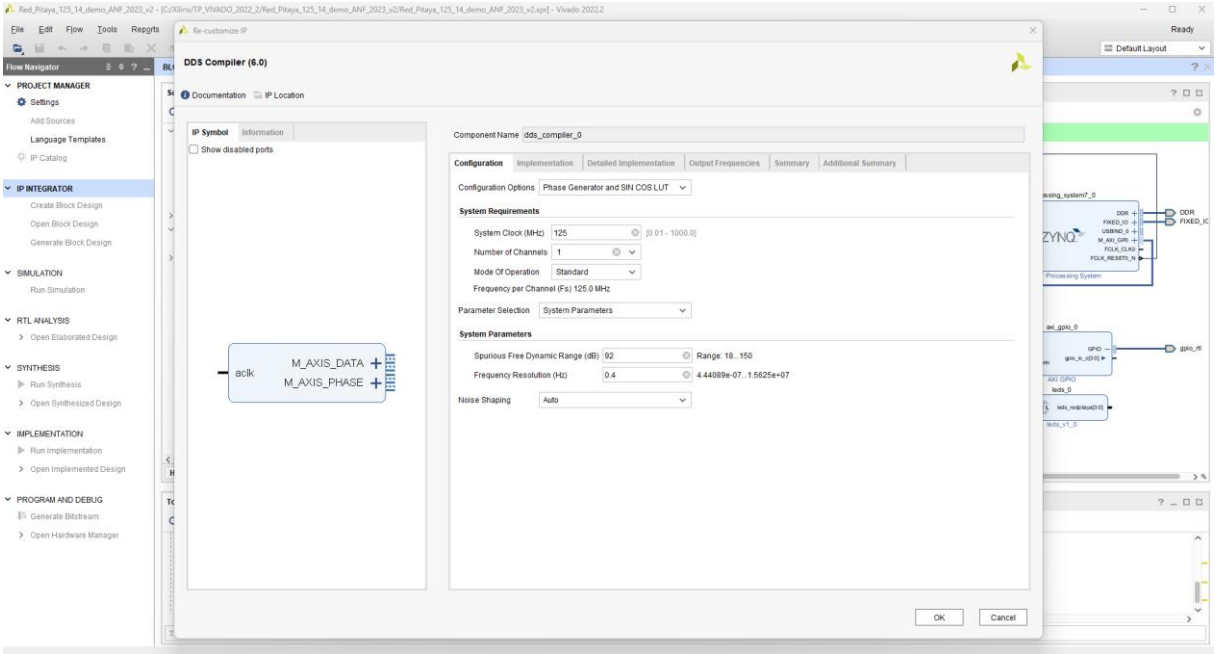
Nous allons ajouter une **IP Xilinx DDS** dans le projet : DDS Compiler puis-> touche **entrée**



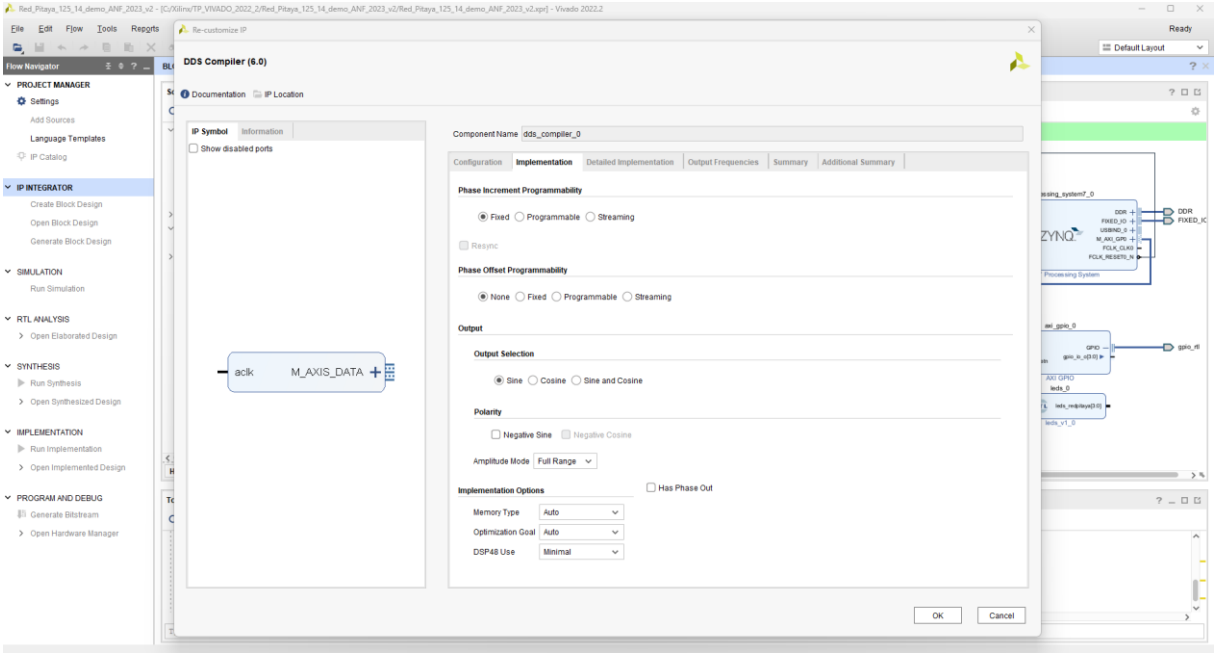
IP ajouté



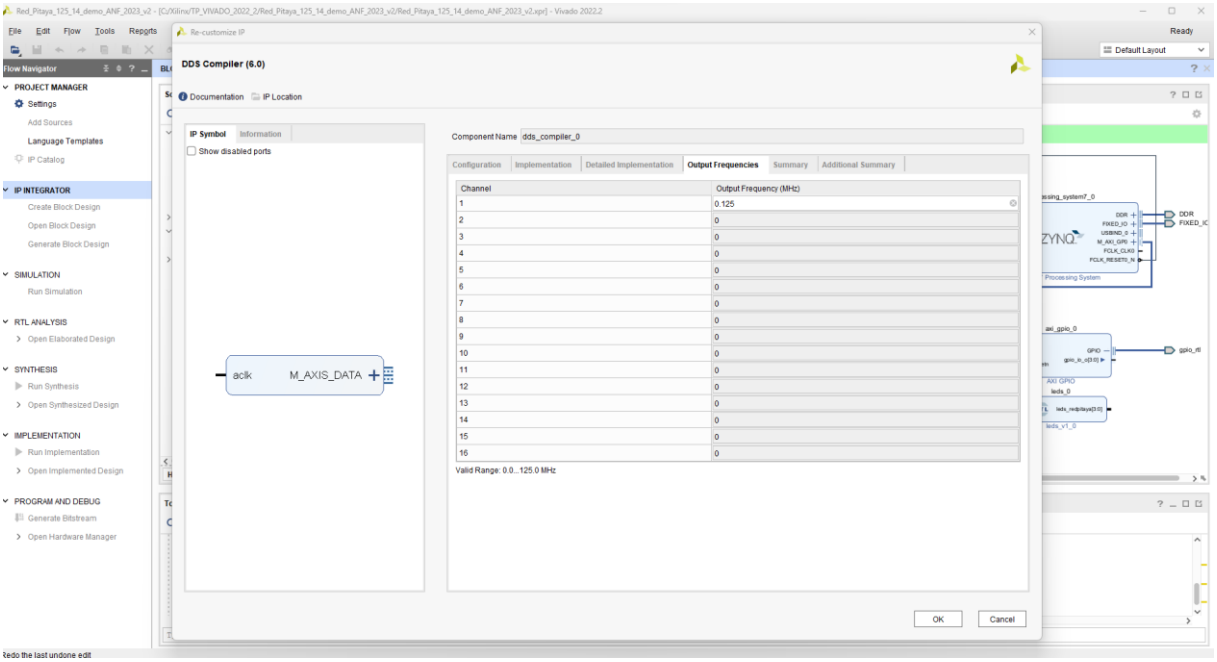
Double clic sur l'IP DDS Compiler : on configurer l'IP pour générer une modulation de fréquence de 125KHz-> dans **configuration**-> sélectionnez comme ci-dessous



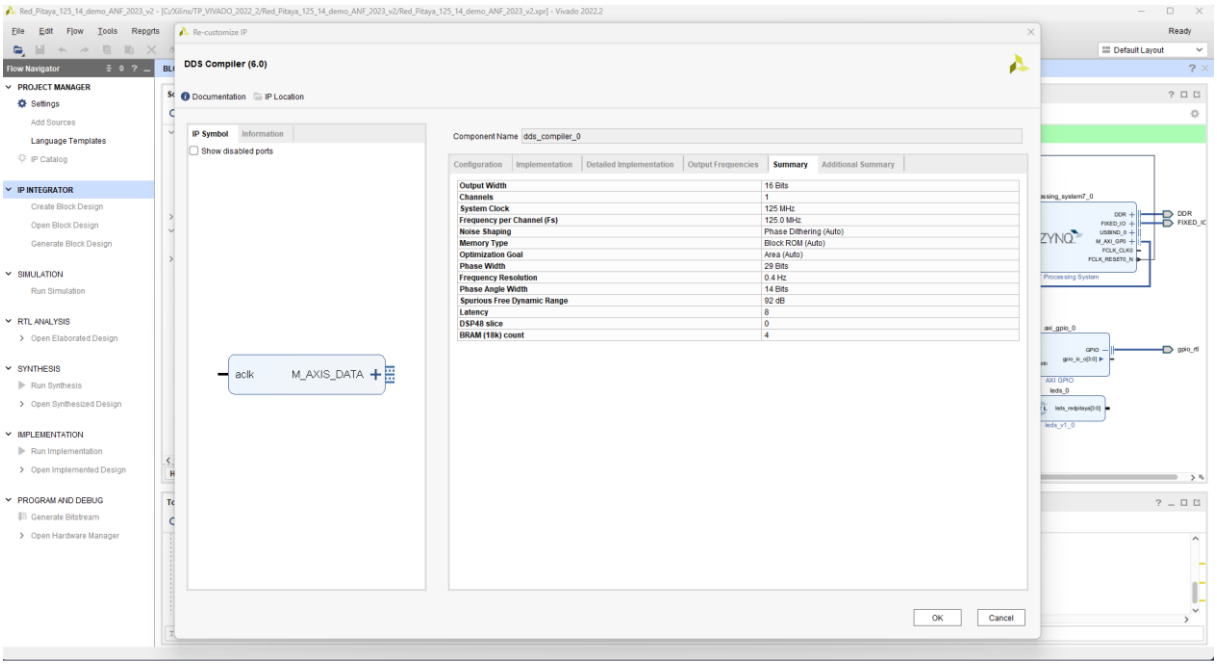
Dans **implementation**-> sélectionnez comme ci-dessous



Dans **Output Frequencies**-> sélectionnez comme ci-dessous



Dans **Summary**-> On vérifie que l'on génère un signal @125KHz sur 16bits



Pour finir il nous faut câbler manuellement les modules VHDL **insérés comme ci-dessous** : nous allons utiliser les ADC et DAC de la RedPitaya avec les IP « maison » développées en VHDL.

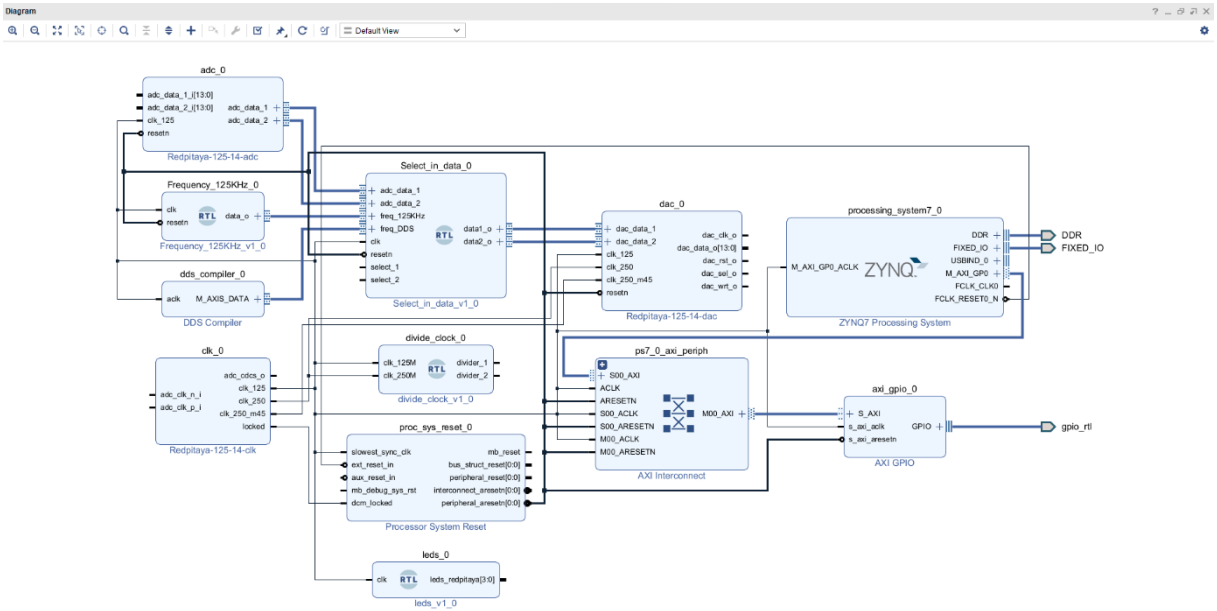
<u>Rôle des IP « maison » :</u>

Le module **select_in_data** : permet de sélectionner les signaux d'entrées des deux DAC. Deux BP sur la carte fille permettent la sélection des data-> Les signaux sont issus des ADC1 et ADC2 ou des IP internes **Frequency_modulation** et **DDS_Compiler.**

Le module **divide_clock** : permet de diviser l'horloge@125MHz vers 1Hz et visualisé sur des Leds de la carte fille.

Le module **Leds** : Gérer les Leds de la RedPitaya.

Relier les modules comme ci-dessous :



Enfin il faut câbler les ports d'entrées-sorties :

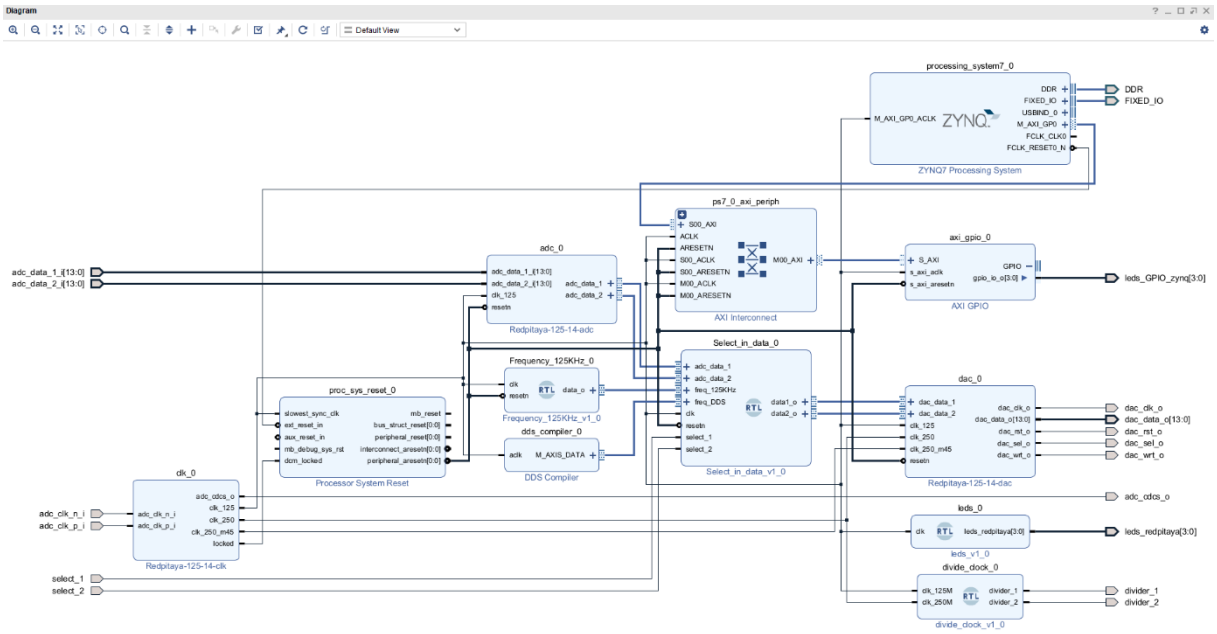
Par exemple on se place sur l'entrée adc_clk_n_i : clic droit puis

Create Port (Ctrl + K) ou **Make External** (ctrl + T).

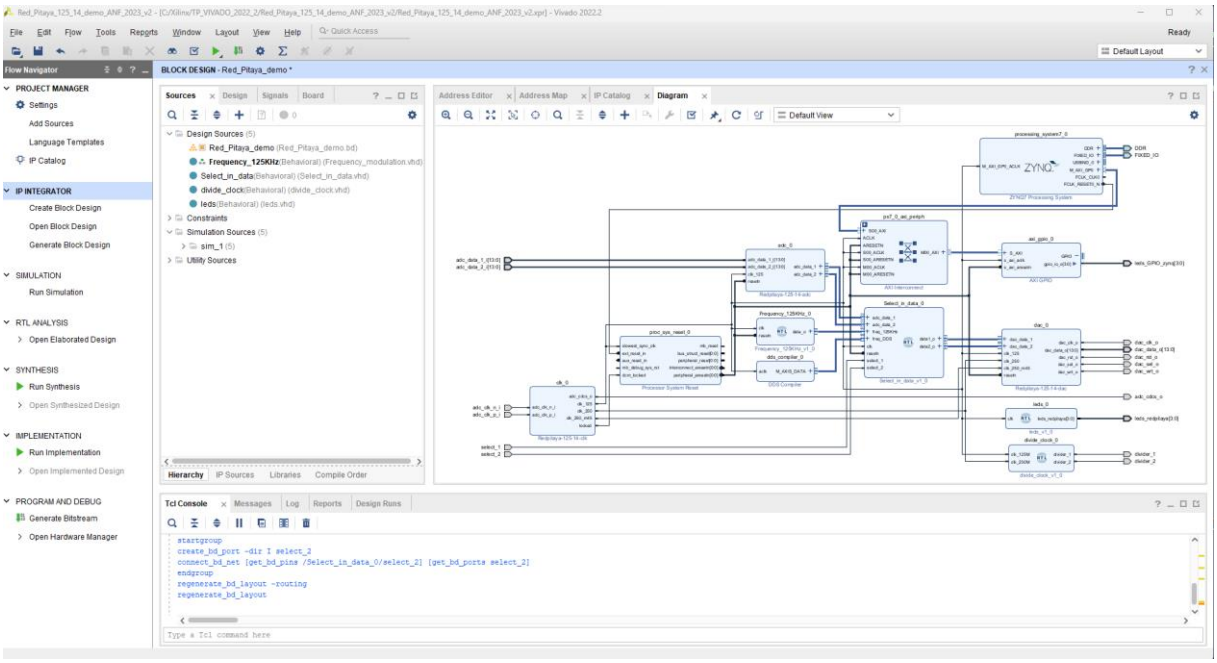
Attention seul **Create Port** vous permet de renommer la broche.

Faire de même pour toutes les autres entrées-sorties.

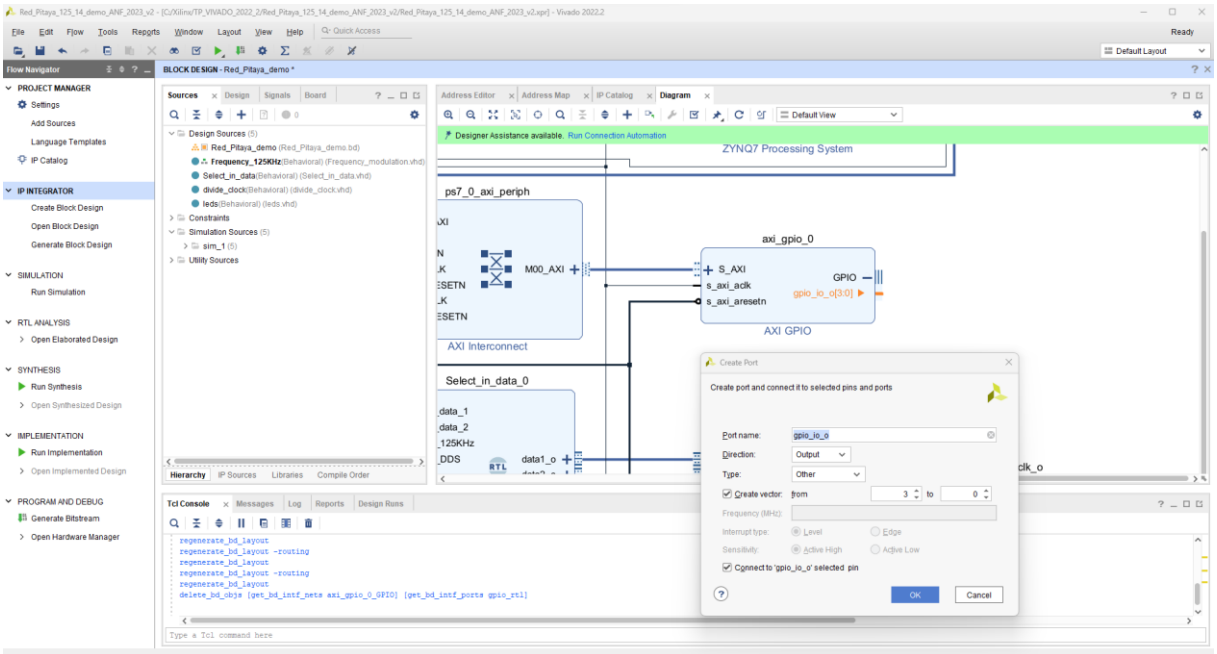
Zoom sur le **Diagram**



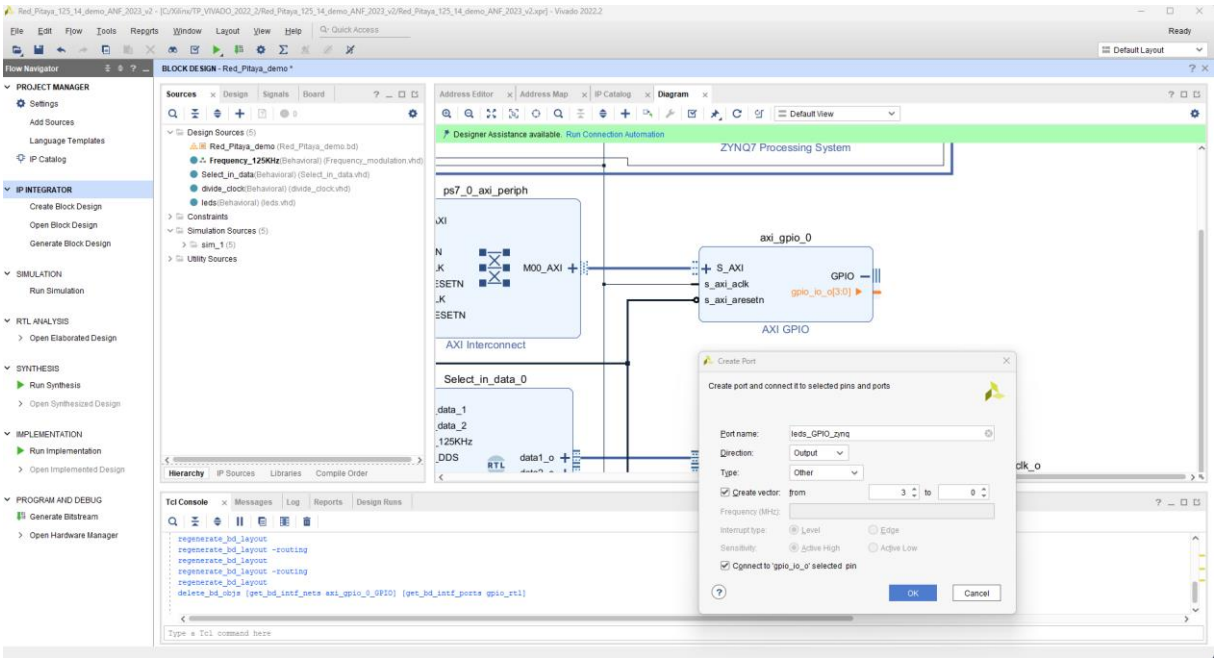
Vue complète Vivado :



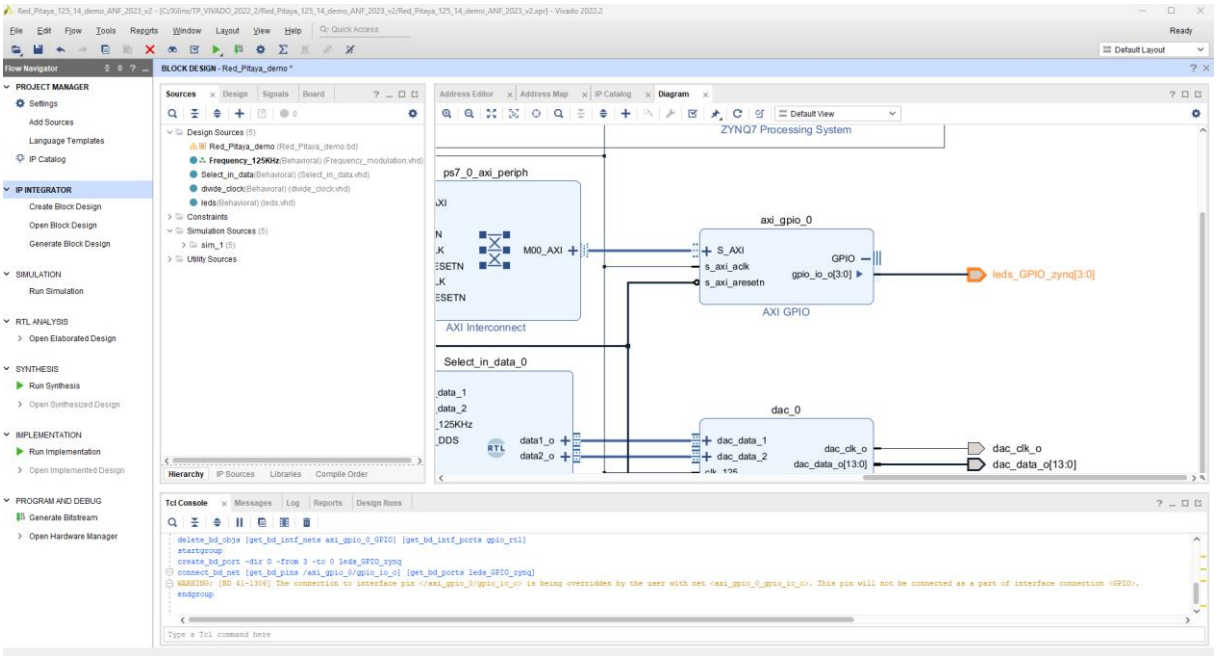
Pour finir mous allons modifier la sortie de axi_gpio_0 comme ci-dessous :



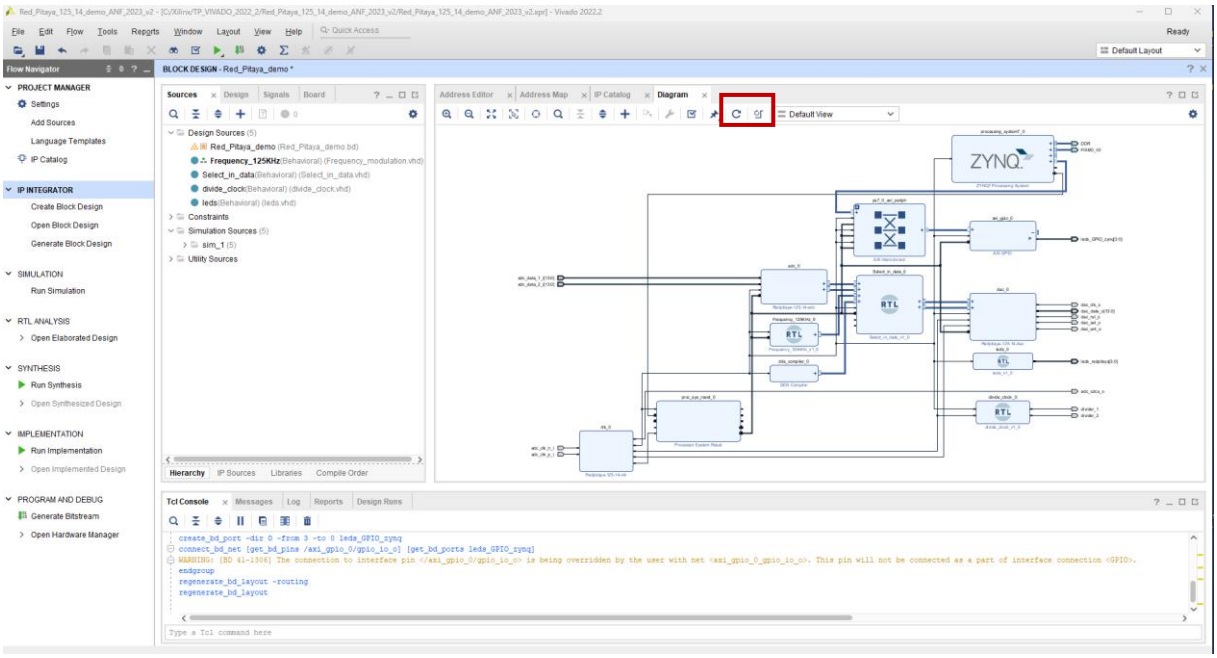
Renommer la sortie : leds_GPIO_zynq



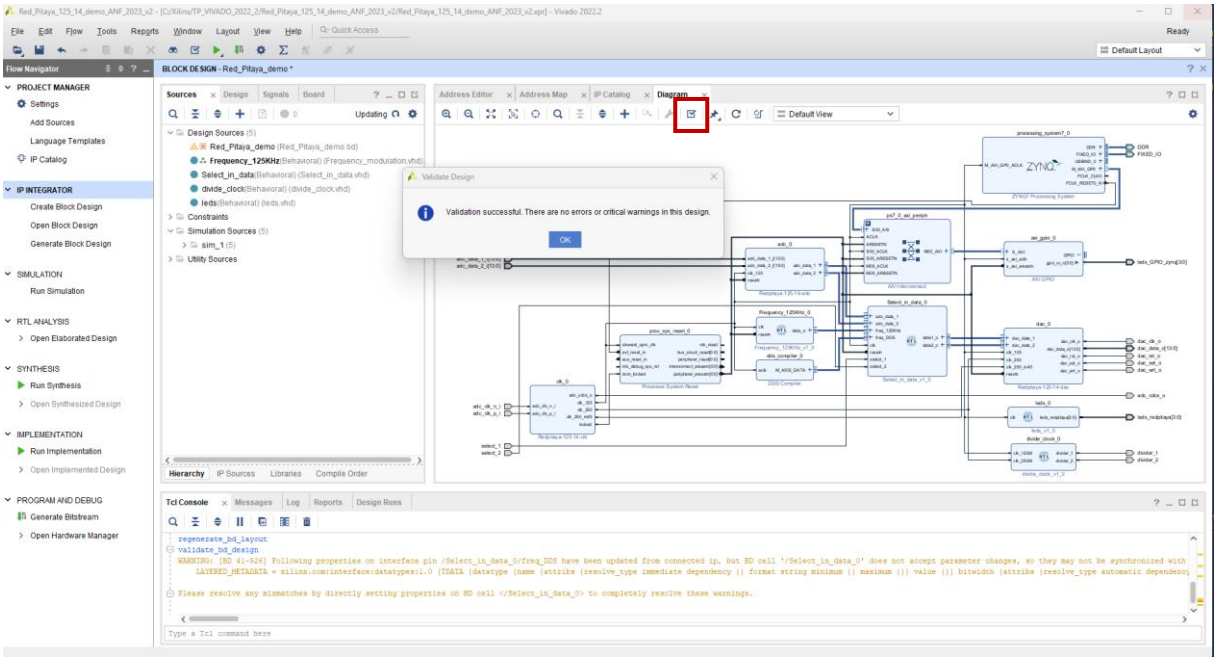
Le module AXI GPIO et sa sortie sur 4 bits



On agence automatiquement le Design

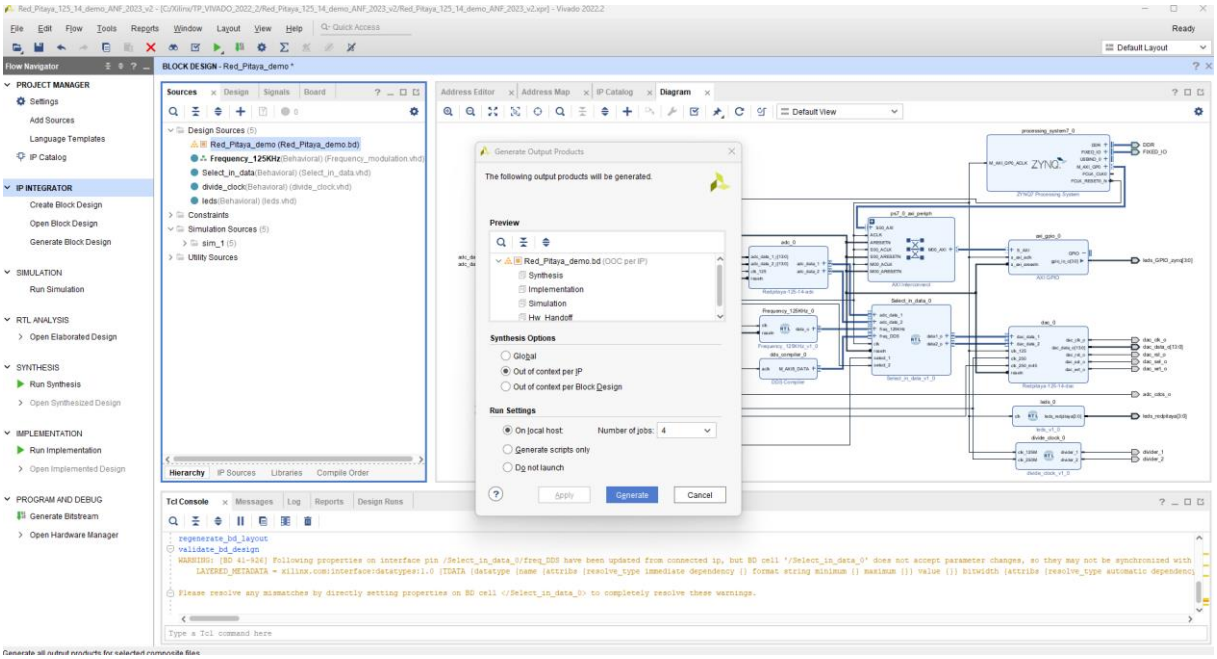


On valide le Diagram

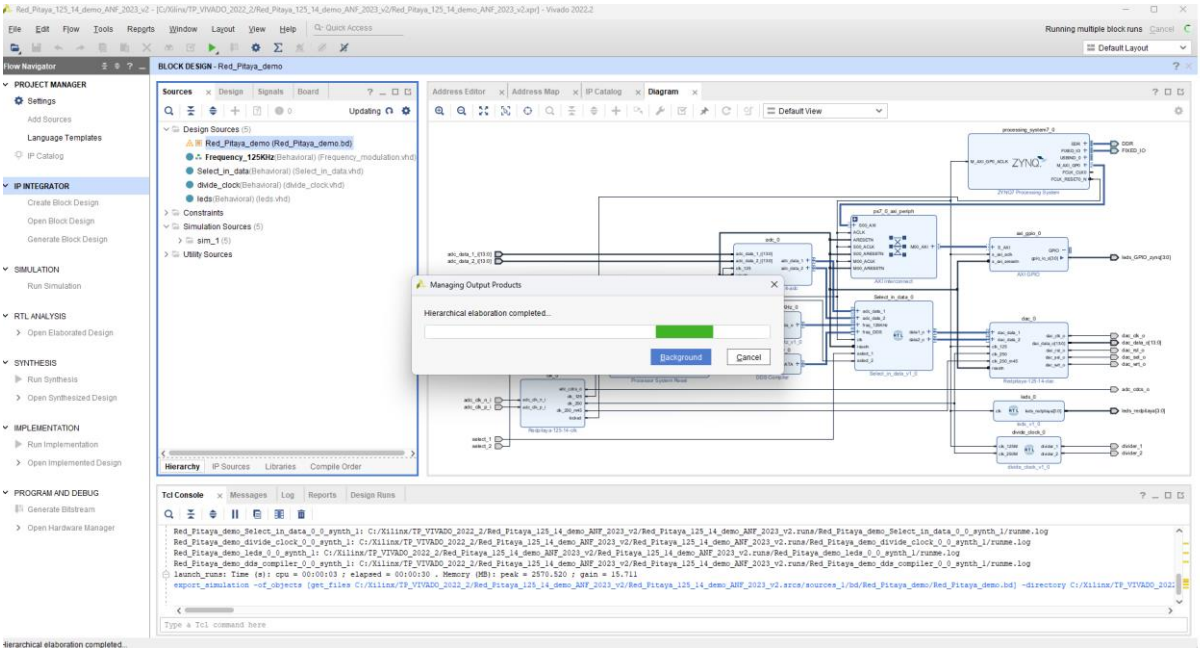


Création du HDL Wrapper :

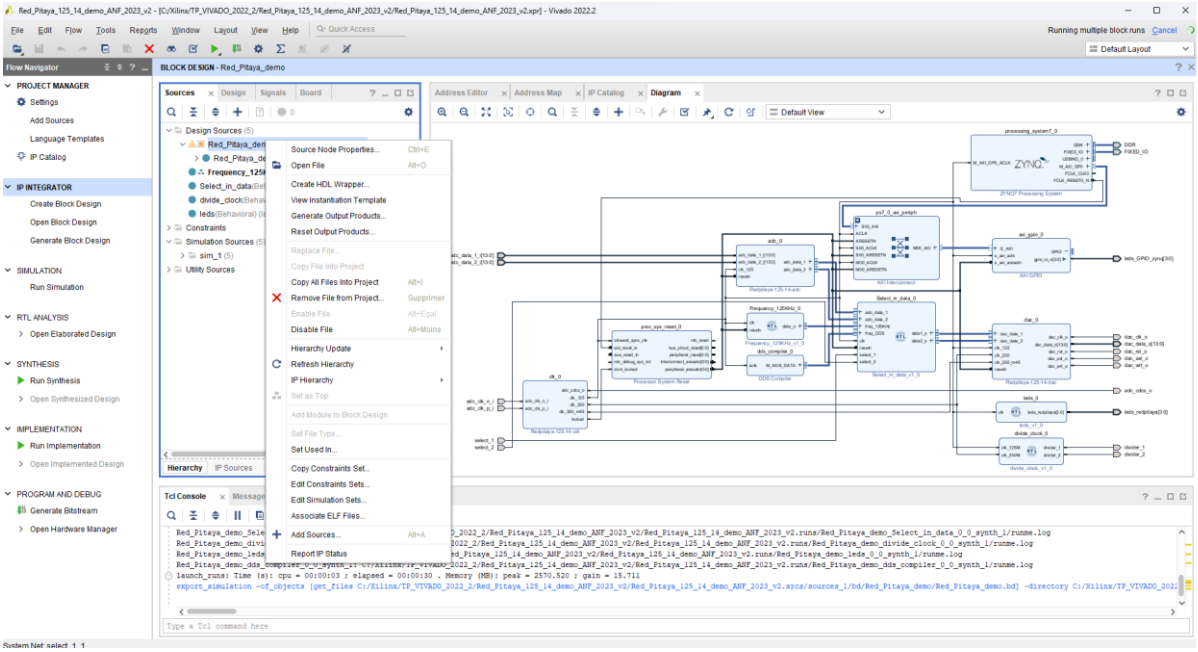
Dans l'onglet **Sources /Hierarchy** du BLOCK DESIGN « Red_Pitaya_demo », clic droit et sélectionnez « **Generate Outputs Products** », « **Generate** » puis « **OK** »



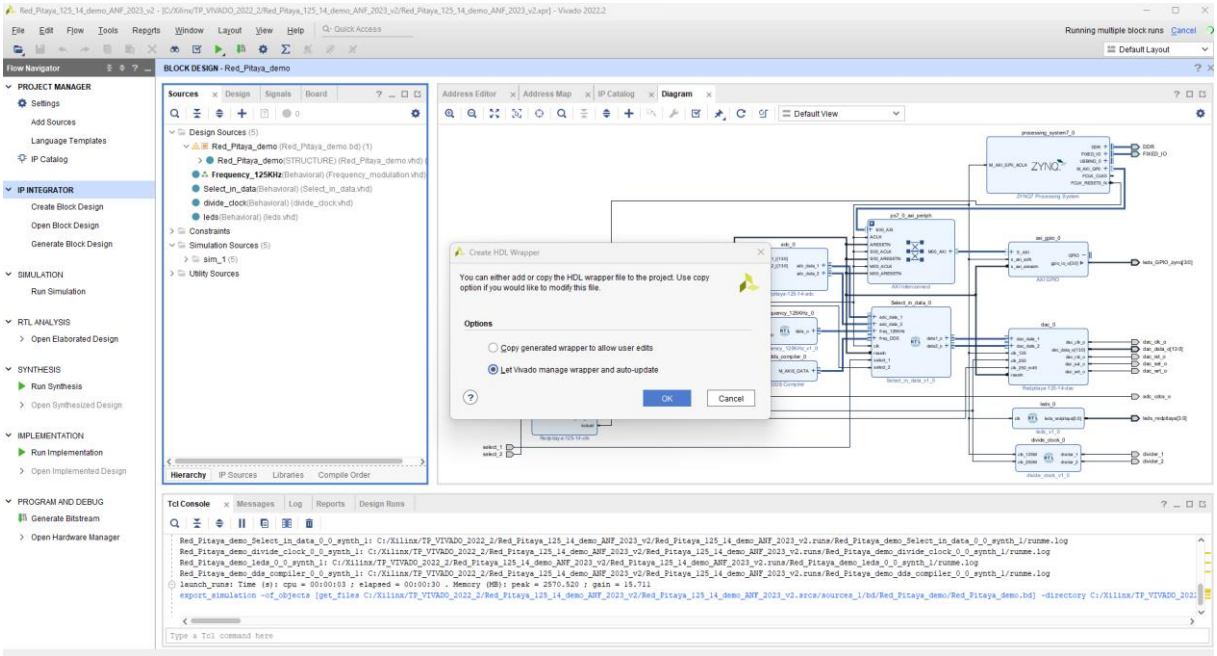
Generate Outputs Products …



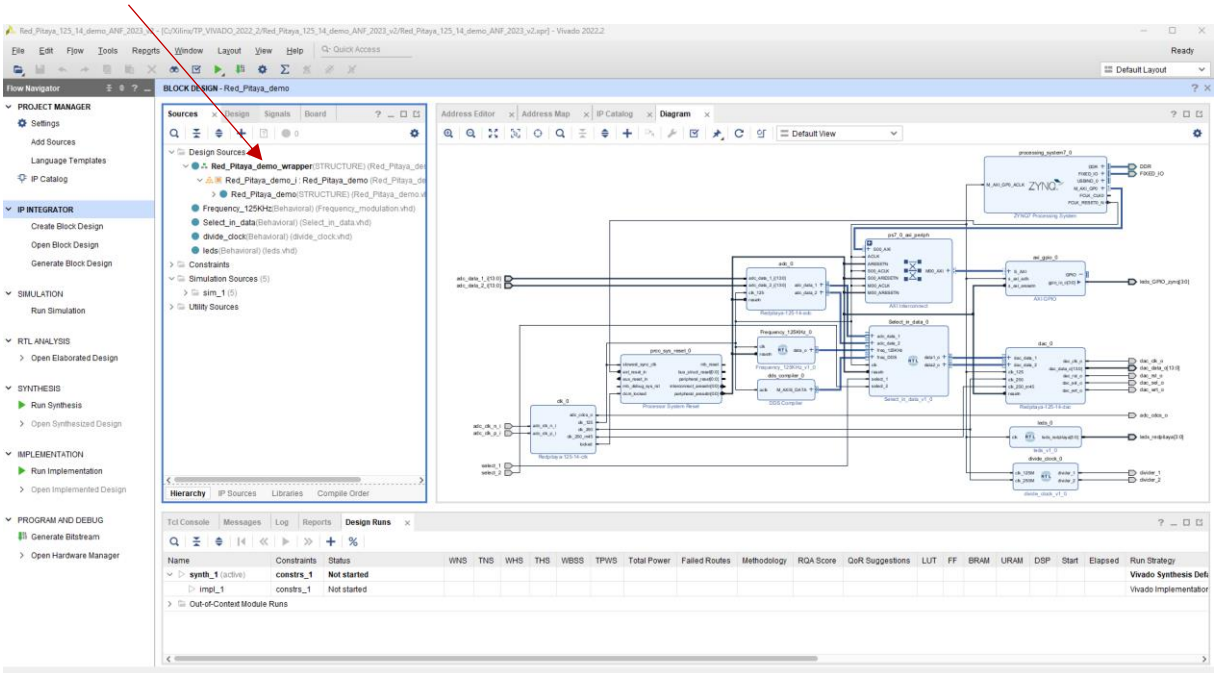
Dans l'onglet « Red_Pitaya_demo », clic droit et sélectionnez « Create **HDL Wrapper** » :



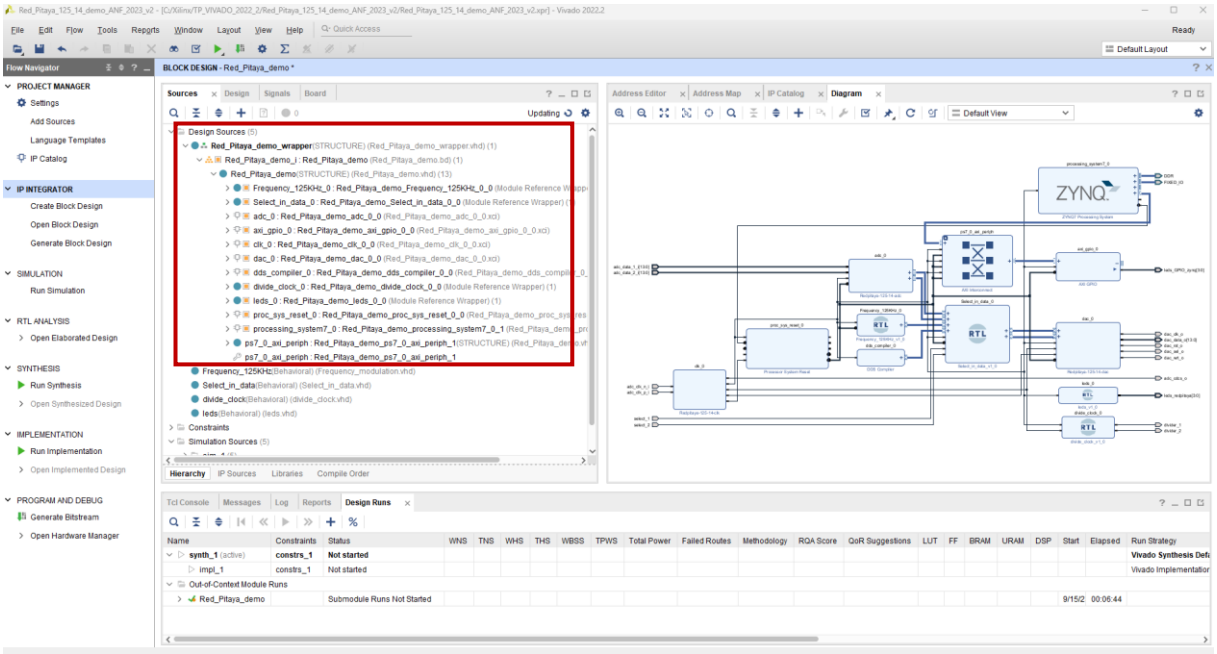
Let vivado manage wrapper… puis-> **OK.**



Wrapper crée !

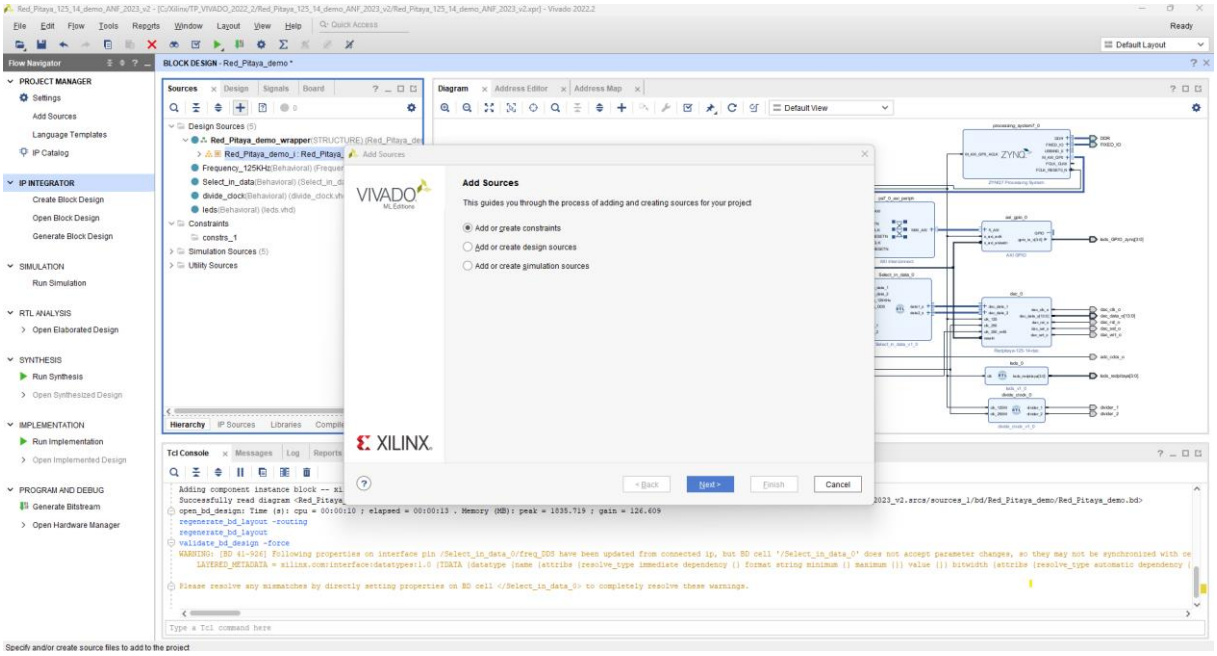


Arborescence complète !



Ajout ou création d'un fichier de contrainte :

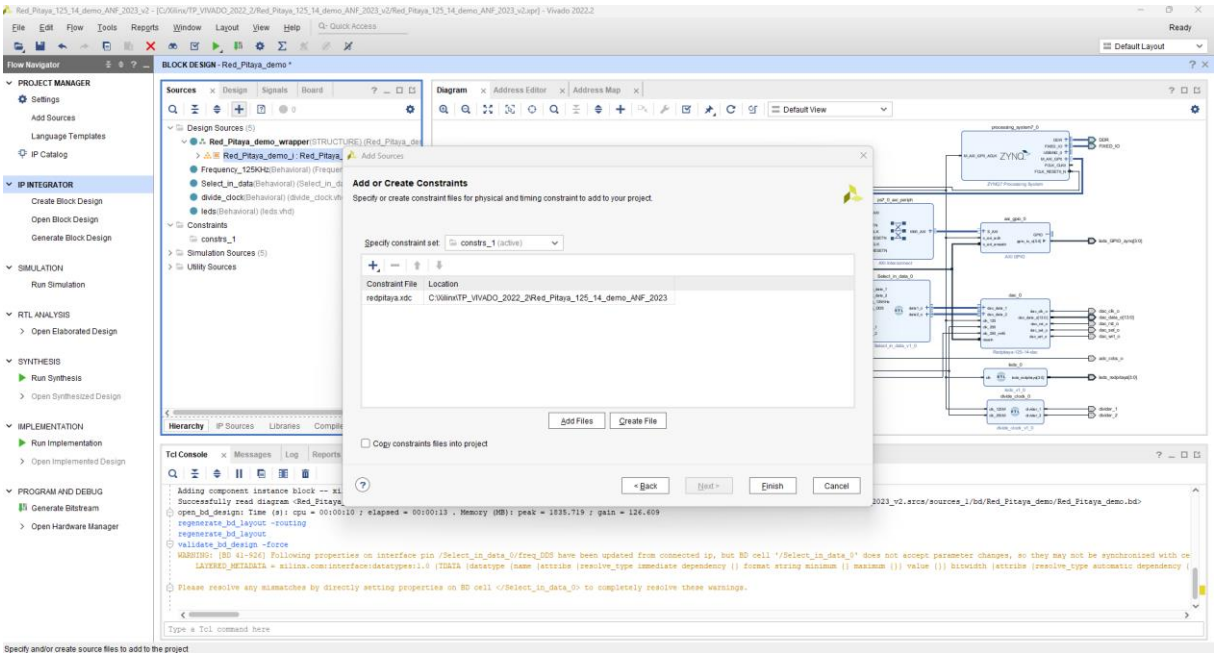
Dans **PROJECT MANAGER**-> Sélectionnez dans Add Sources-> « **Add or create constraints »-> Next>**



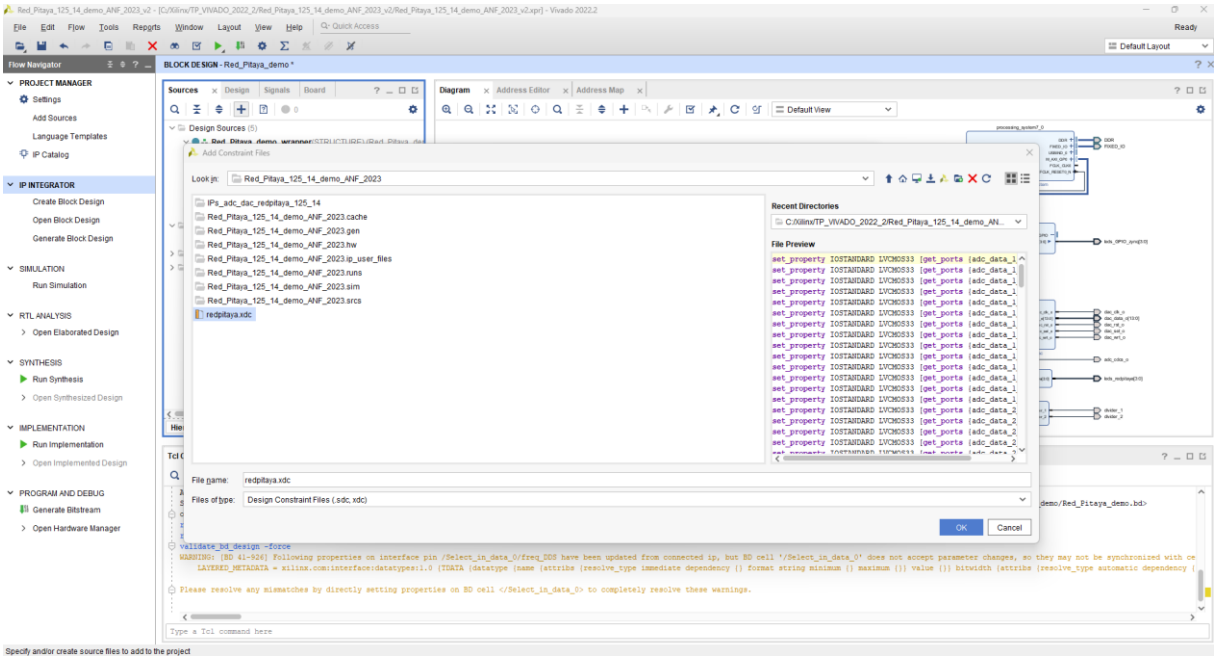
Nous allons récupérer le fichier de contrainte dans le dossier source du projet : CmodA7.xdc : Sélectionnez redpitaya.xdc puis « **OK** »

https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

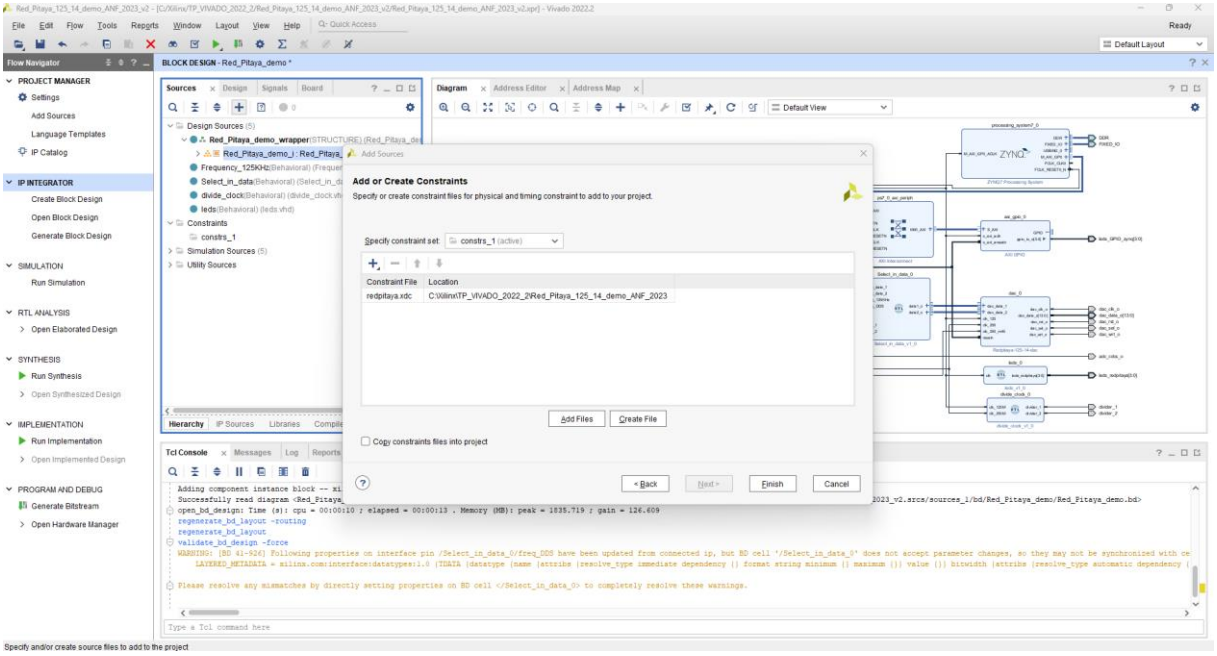
Add or Create Constraints-> **Add Files**-> **Next.**



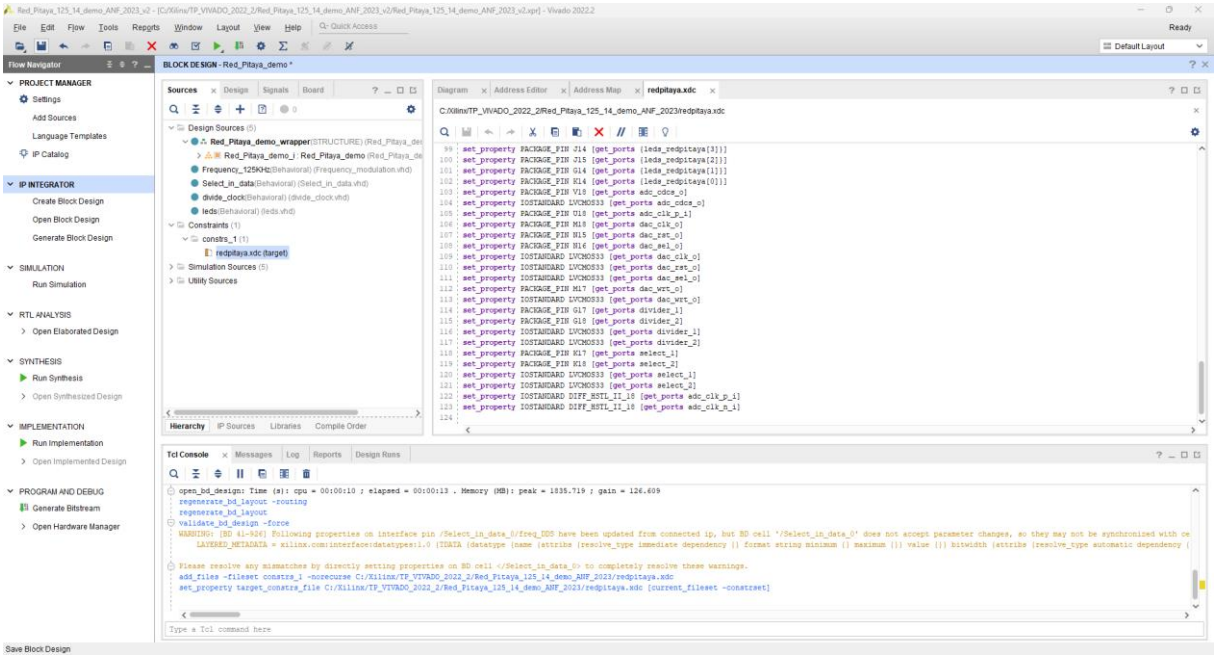
Sélectionnez redpitaya.xdc puis « **OK** »



Finish…

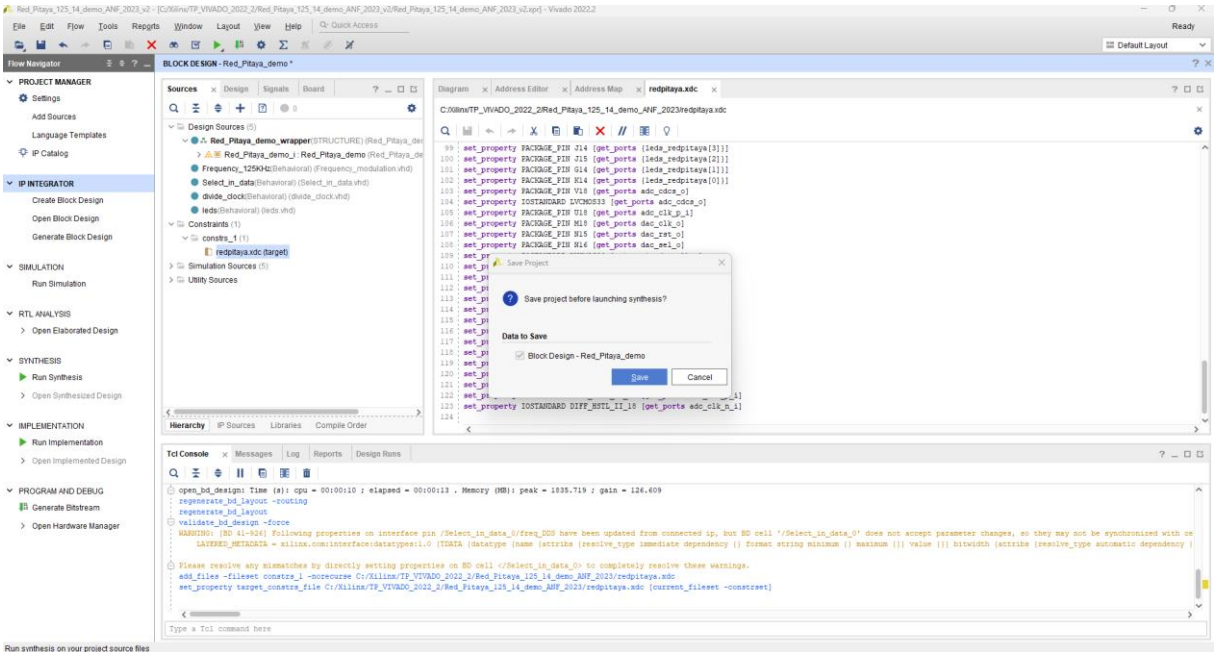


Le fichier de contrainte apparait dans **BLOCK DESIGN**-> sous l'onglet **Sources**-> **Constraints.** Clic droit sur le fichier de contrainte et définir comme la cible principale-> » **Set as target constraint file ».**

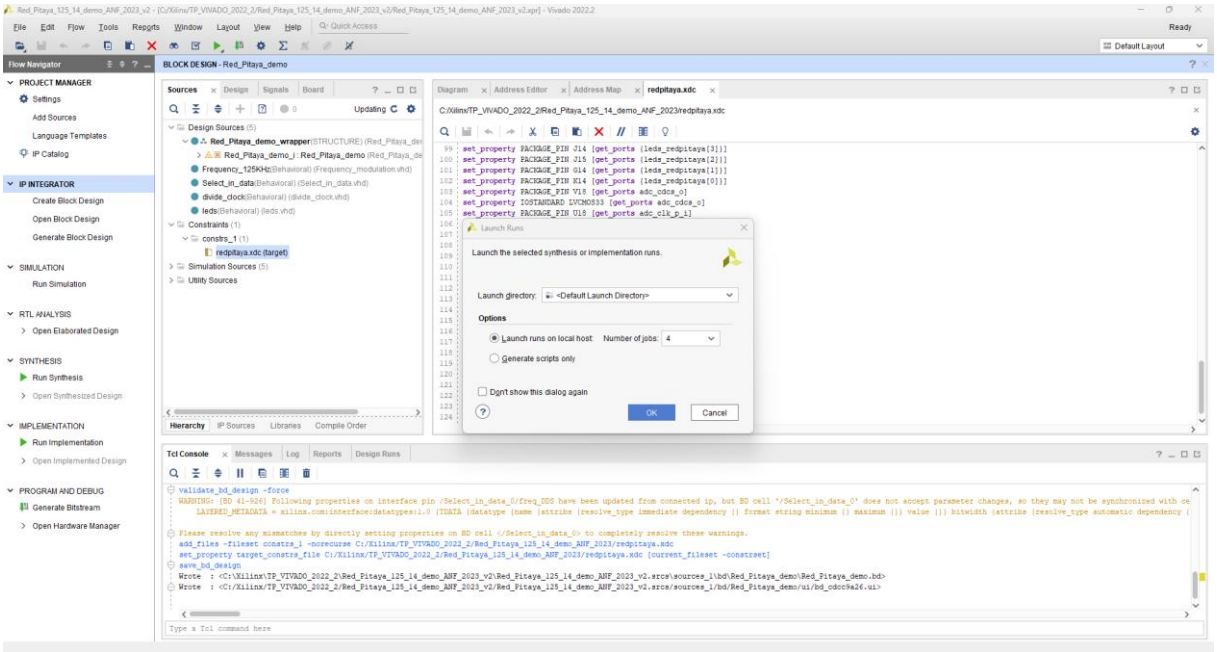


On lance la synthèse-> **touche F11**

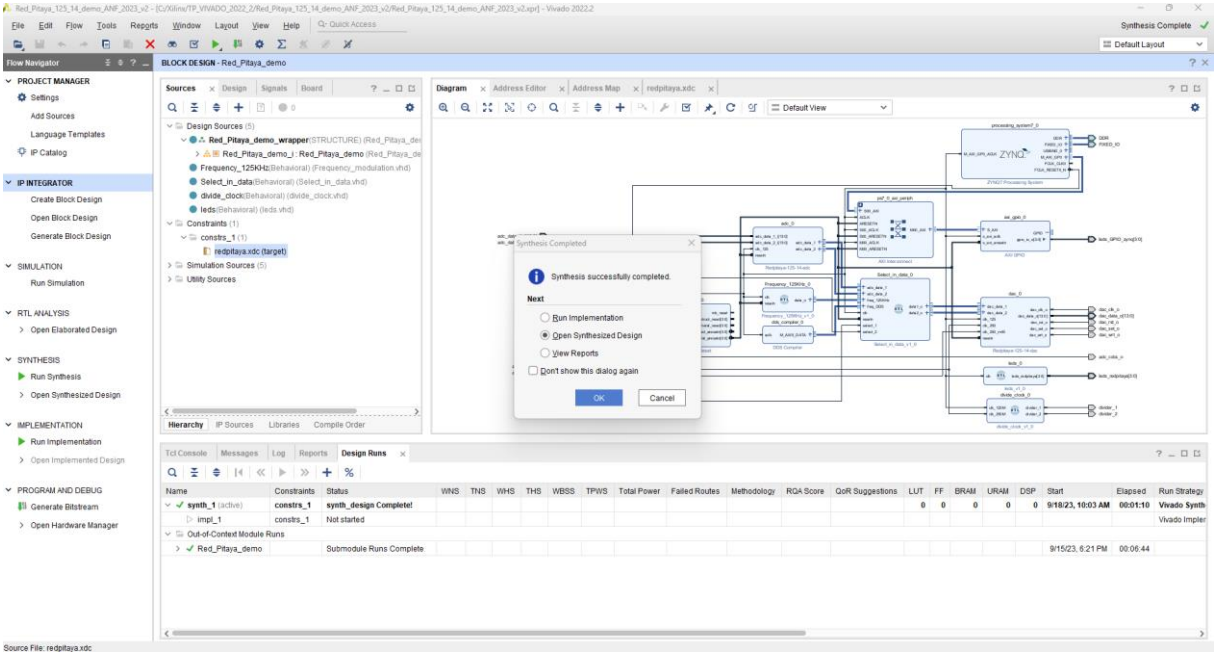
Save…



Clic sur « OK »

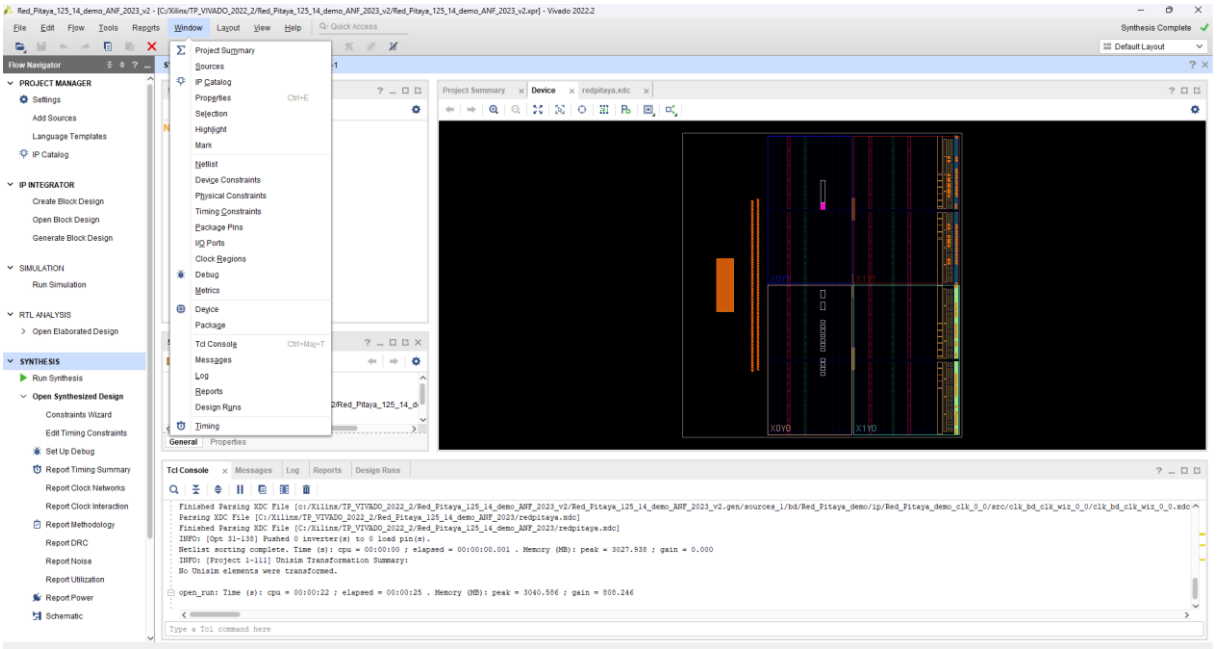


On ouvre l'implémentation :

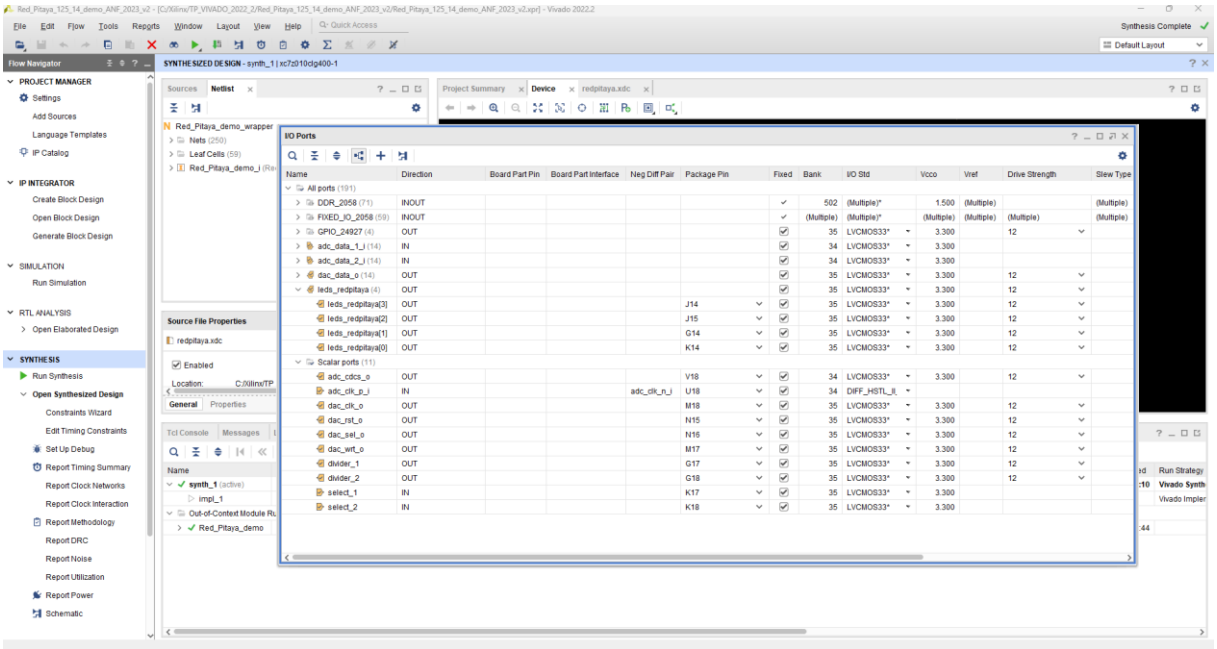


Implémentation…

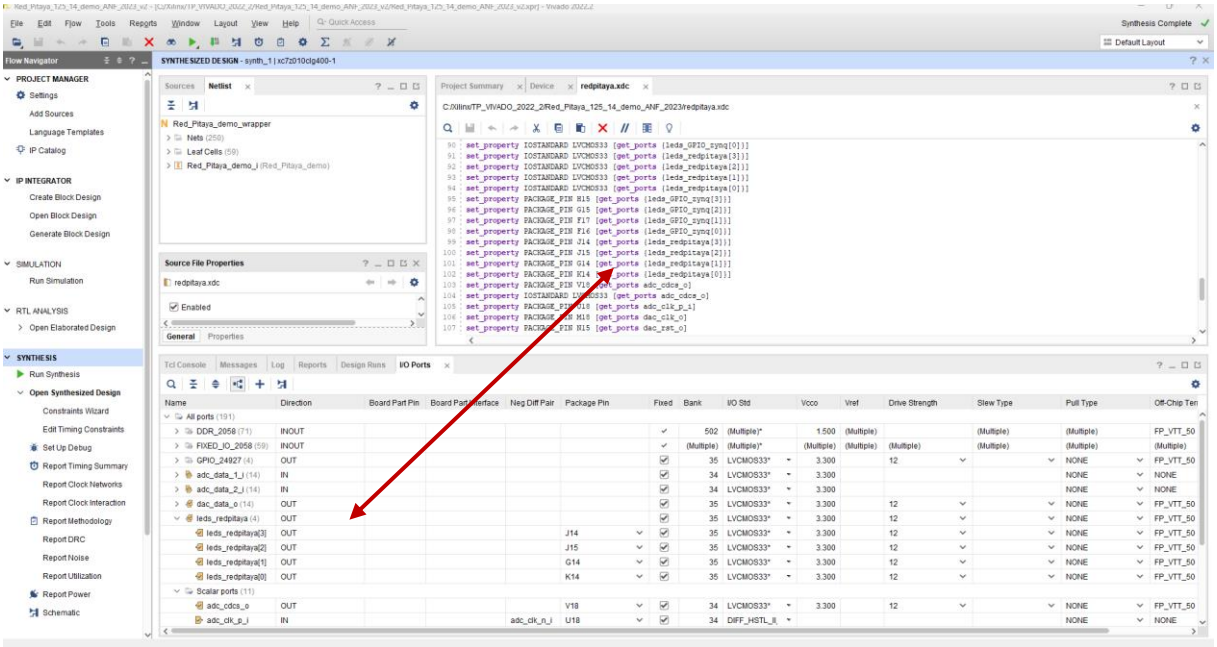
Dans l'onglet Windows-> I/O Ports



On vérifie et on édite si nécessaire les I/O :

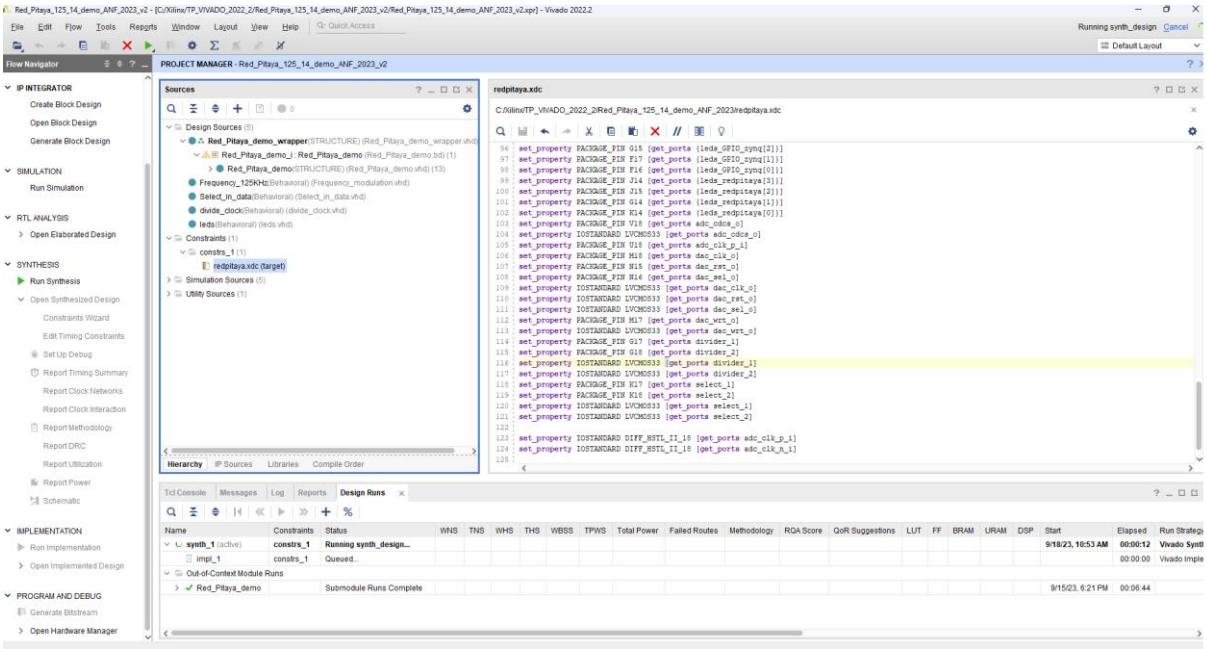


Vérification par rapport au fichier de contrainte redpitaya.xdc

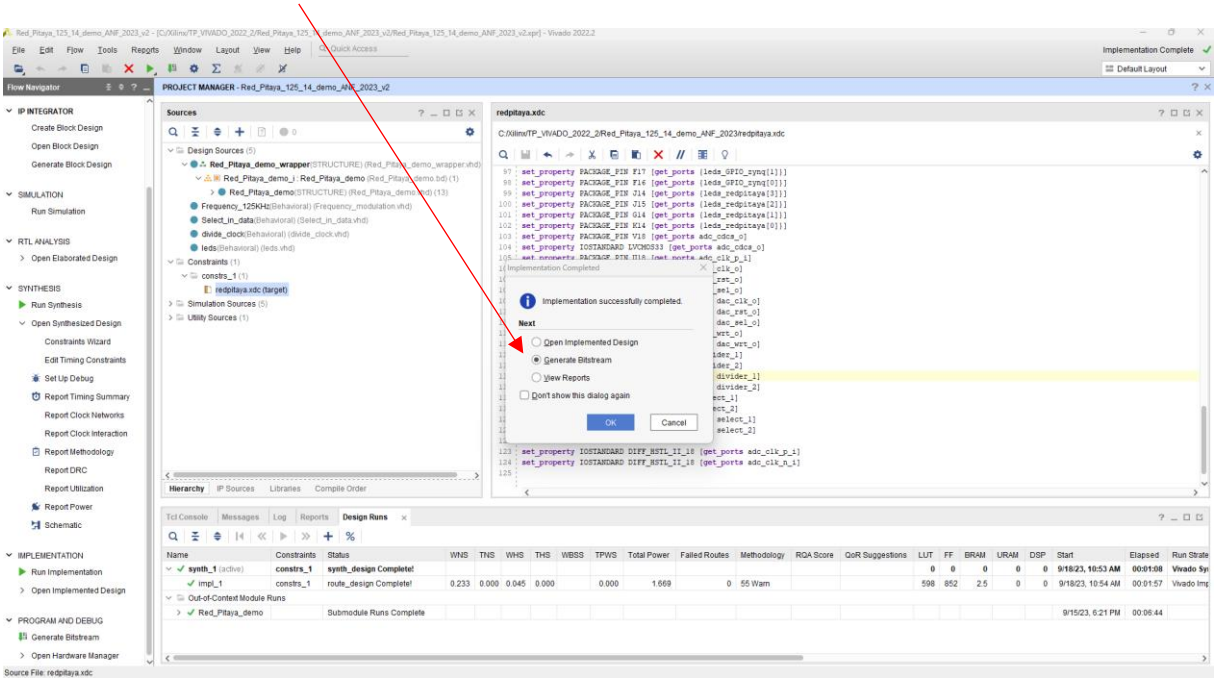


On fait une sauvegarde pour une mise à jour du fichier de contrainte : CmodA7.xdc-> **CTRL + S**

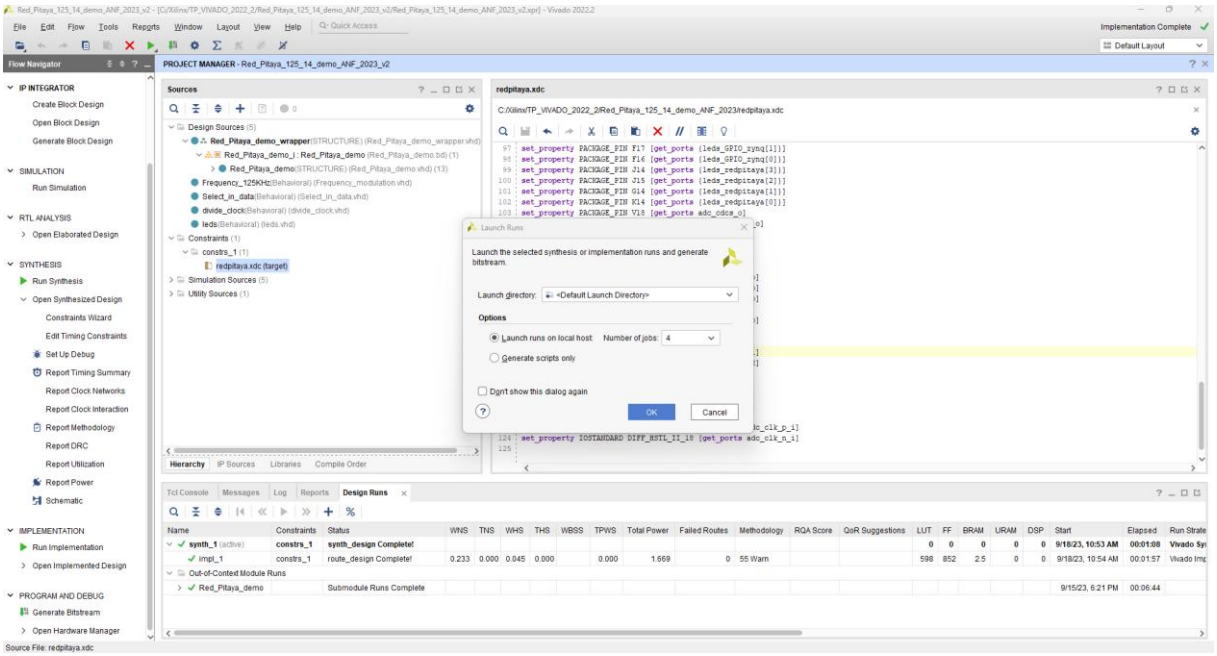
Dans Flow Navigator-> **Run Implementation**



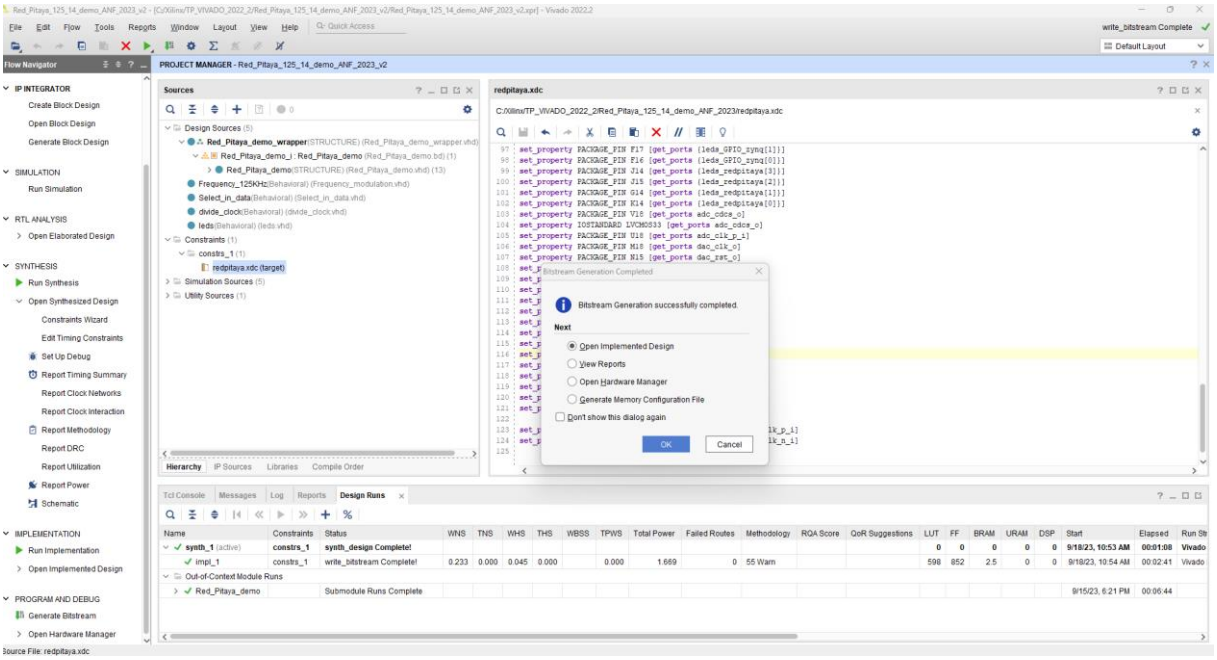
Générer le bit file pour programmer la cible FPGA partie HARDWARE



Launch Runs-> « **OK** »

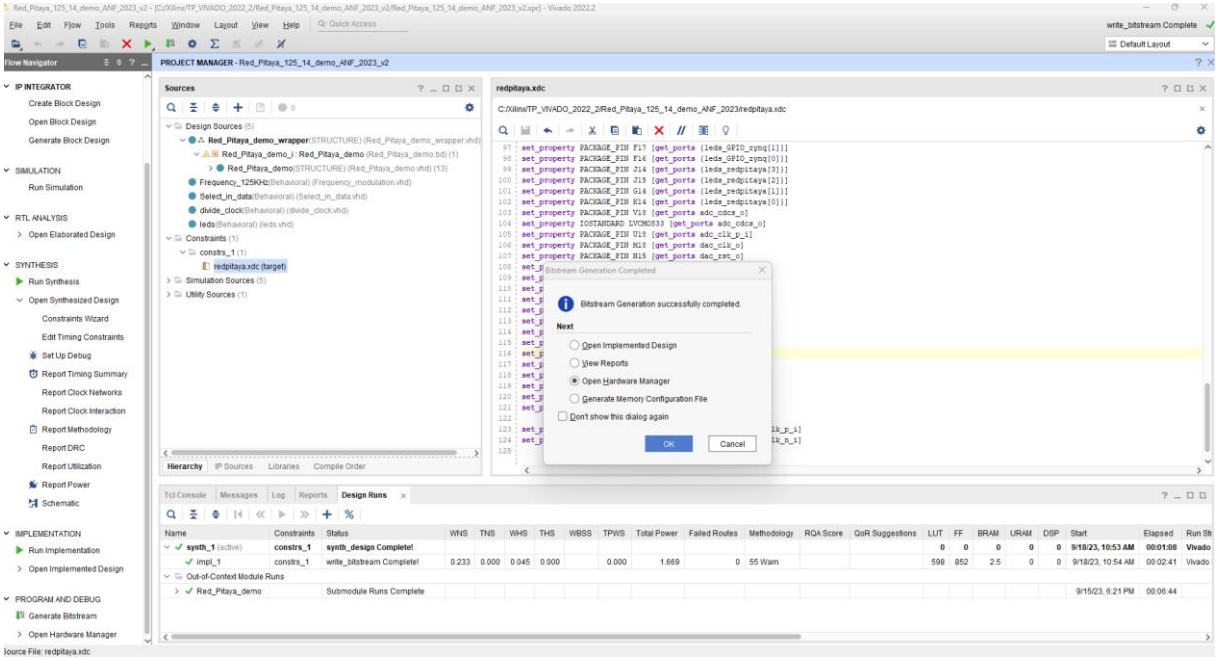


Bitstream OK

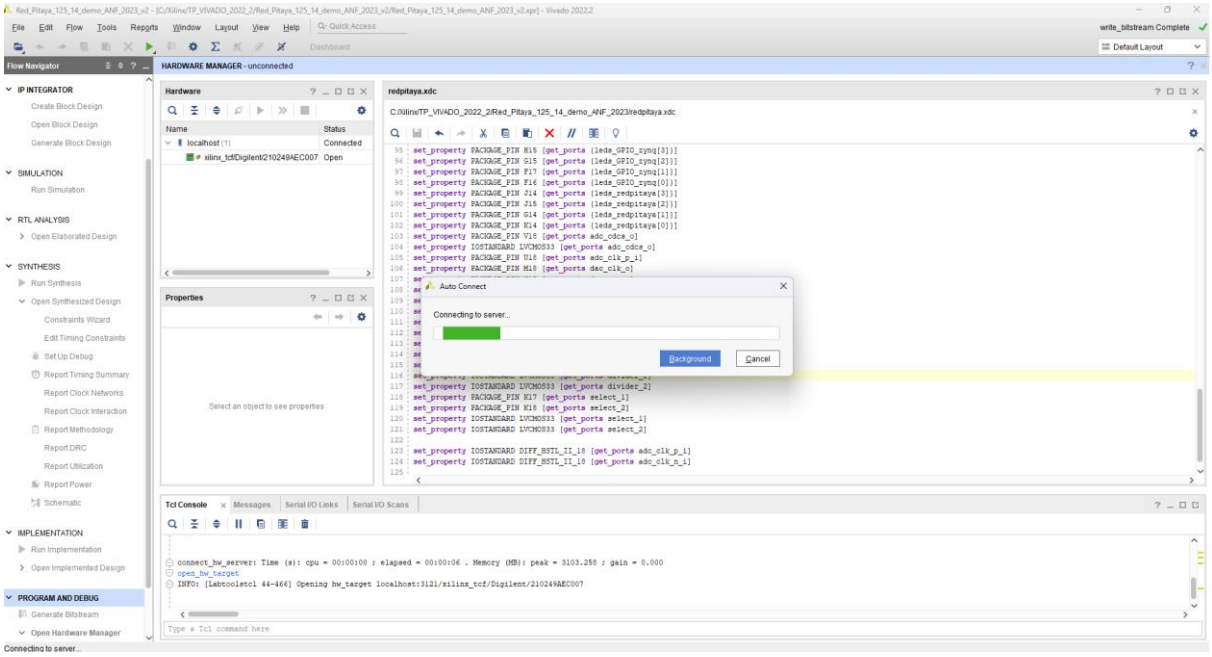


Dans Flow Navigator-> **Open Hardware Manager**-> **Open Target-> Auto connect.**

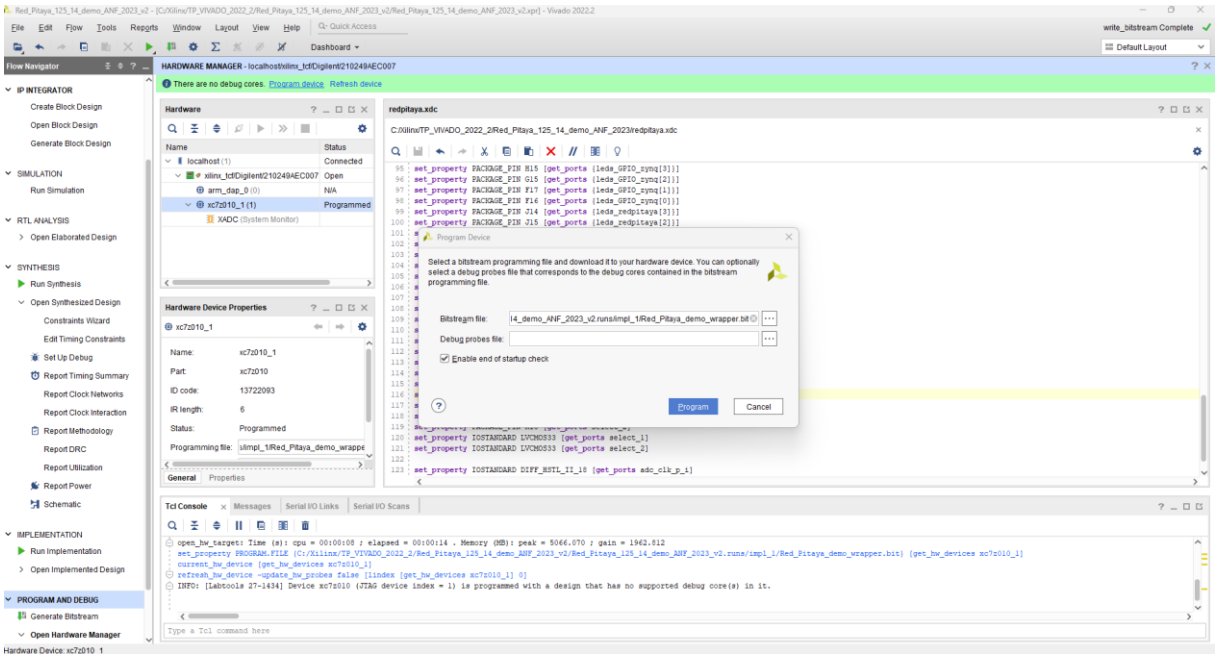
A ce stade on ne programme que le HARDWARE via le câble JTAG-USB.



Open Target-> Auto connect.

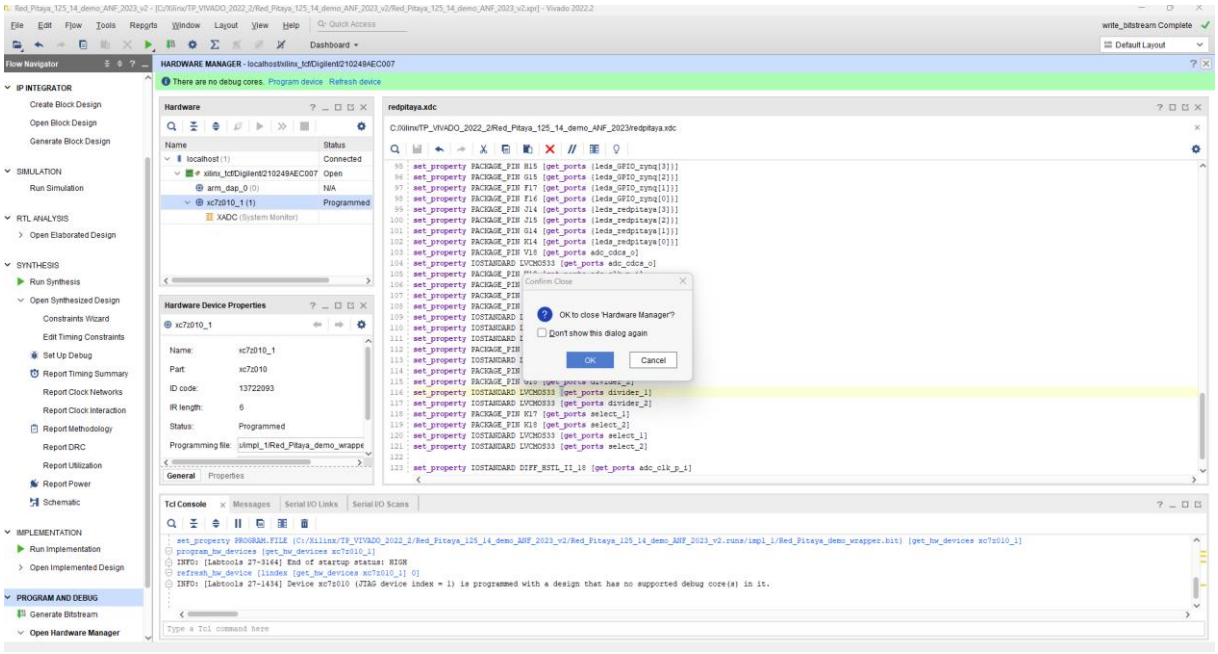


Hardware Manager-> Program Device->Program



Dans l'onglet Program Device vérifier le nom et le chemin du projet…

On ferme **HARDWARE MANAGER->** clic sur **« OK »**

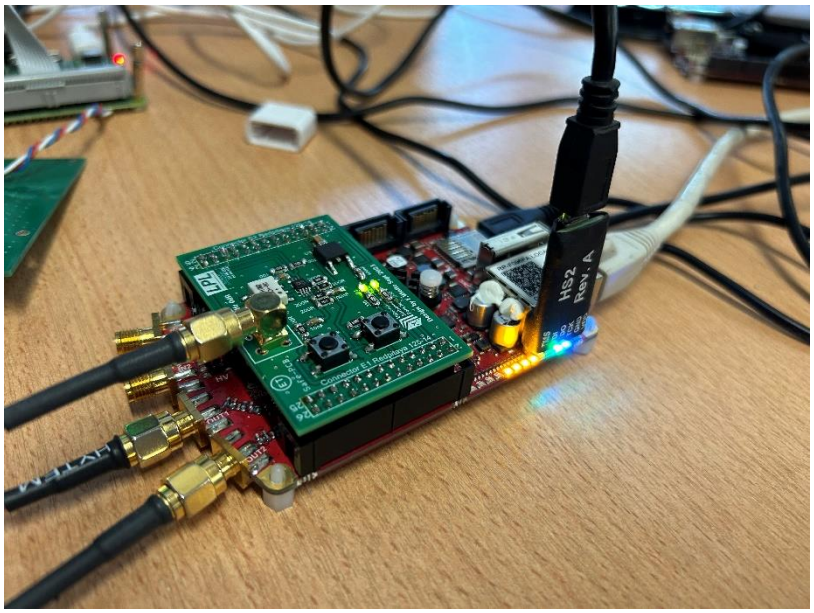


Important : Avec la programmation du bit file sur vivado :

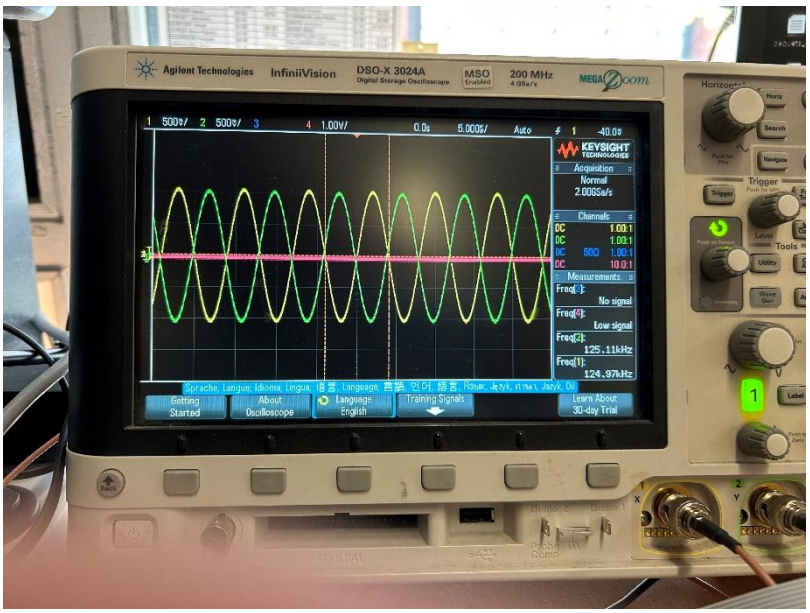
On peut vérifier sur la carte RedPitaya le fonctionnement des ADC-DAC avec les fréquences générées par l'IP DDS Compiler et l'IP maison frequency_modulation

en utilisant les boutons poussoirs. On visualise également l'état des leds (leds_redpitaya [3 :0]) cadencés@1Hz sur la carte fille, et les leds sur la carte RedPitaya (leds_GPIO_zynq [3 :0]) cadencés également @1Hz.

Fonctionnement de la carte Redpitaya 125_14 après chargement du programme

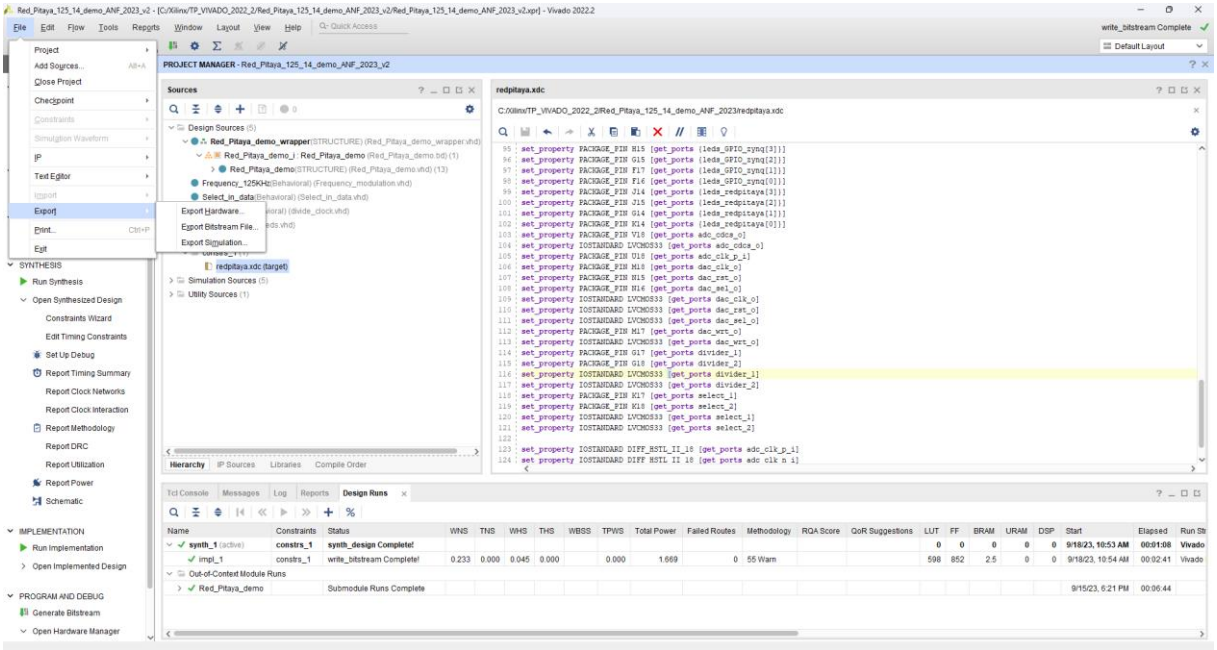


Fréquences générées par l'IP DDS et l'IP maison @125KHz

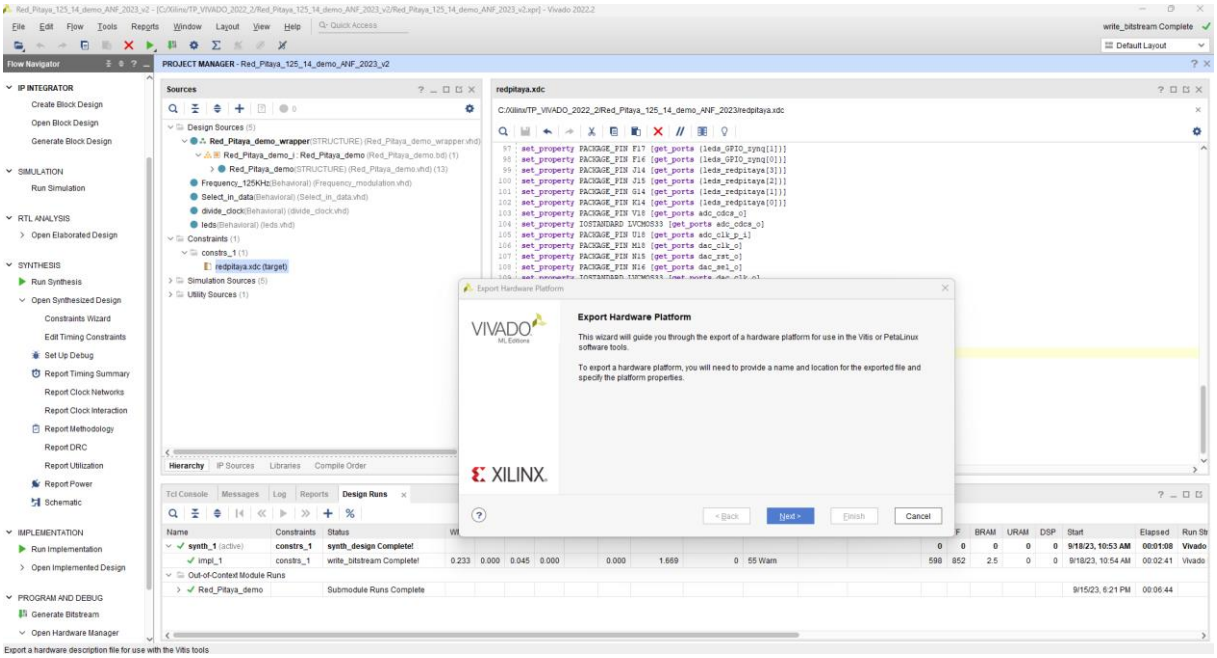


Exporter vers VITIS le projet VIVADO 2022.2

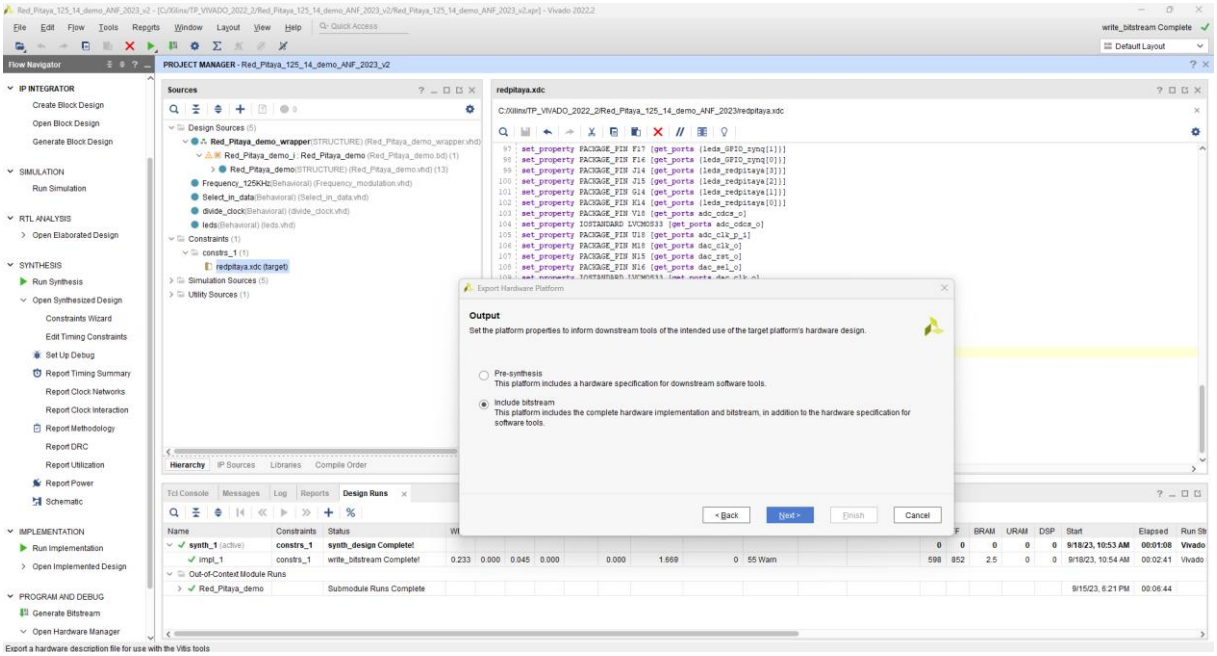
Menu principal-> **File**-> **Export**-> **Export Hardware**



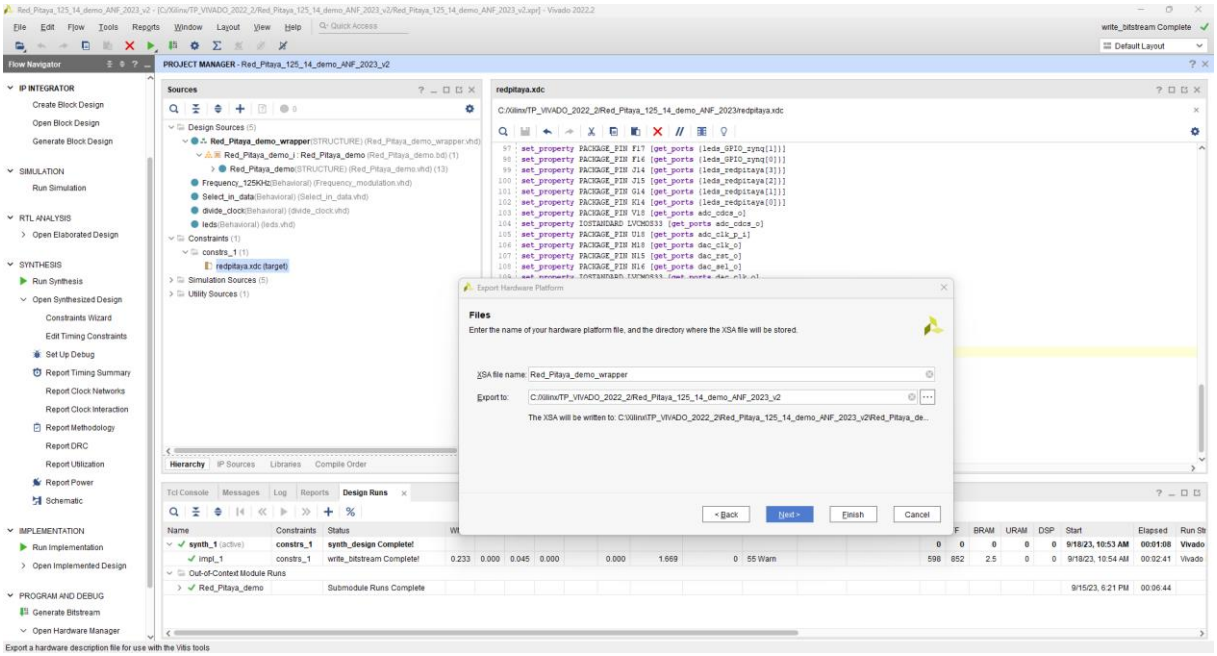
Next…



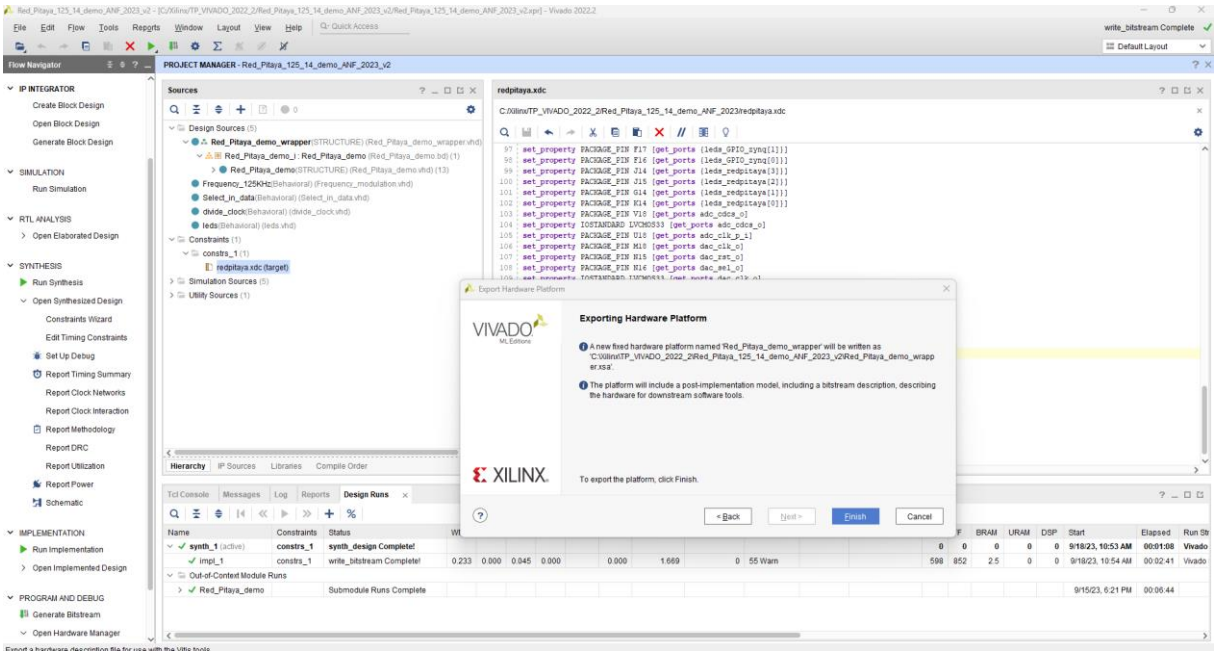
Export Hardware-> **Include bitstream-> Next.**


Export Hardware Platform
Output
Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.
Pre-synthesis
This platform includes a hardware specification for downstream software tools.
Include bitstream
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.
< Back
Next >
Finish
Cancel

Next…

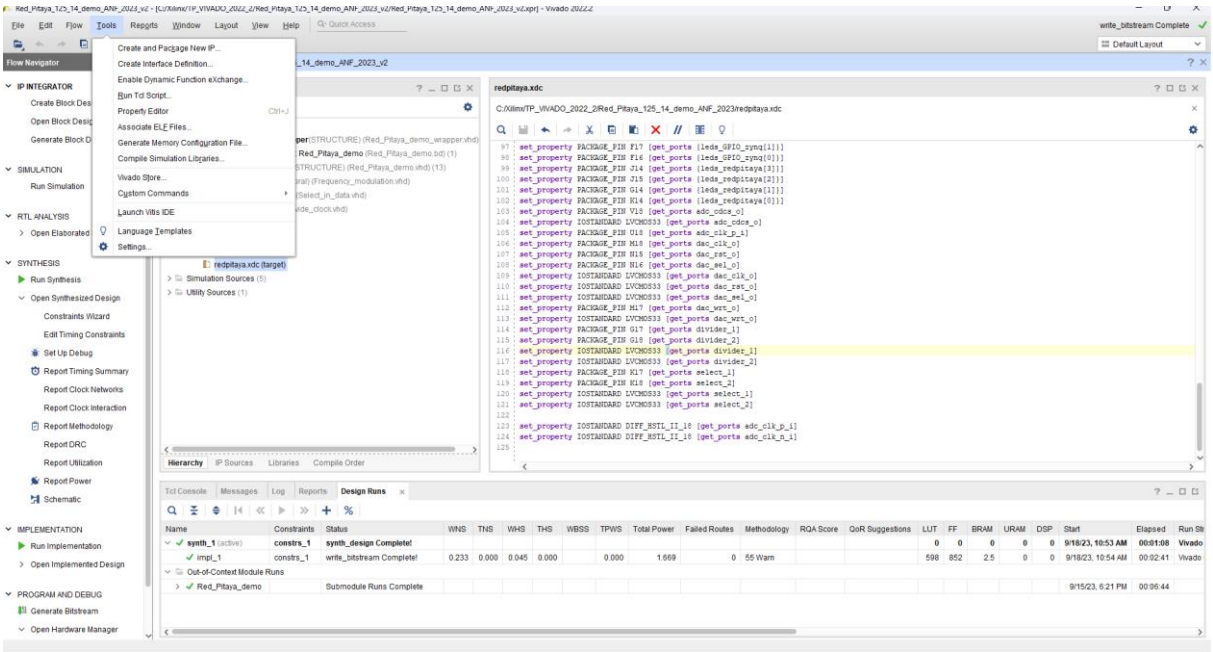

Export Hardware Platform
Files
Enter the name of your hardware platform file, and the directory where the XSA file will be stored.
XSA file name: Red_Pitaya_demo_wrapper
Export to: C:\Xilinx\TP_VIVADO_2022_2\Red_Pitaya_125_14_demo_ANF_2023_v2
The XSA will be written to: C:\Xilinx\TP_VIVADO_2022_2\Red_Pitaya_125_14_demo_ANF_2023_v2\Red_Pitaya_de...
< Back
Next >
Finish
Cancel

Finish

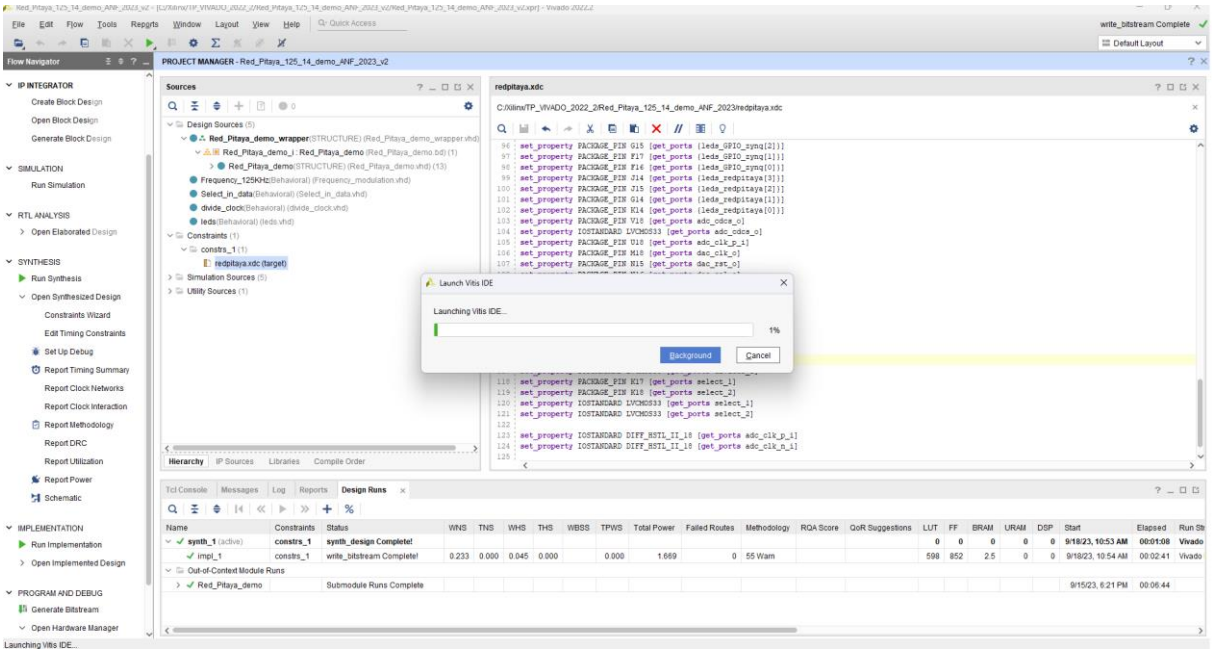


On lance VITIS 2022_2

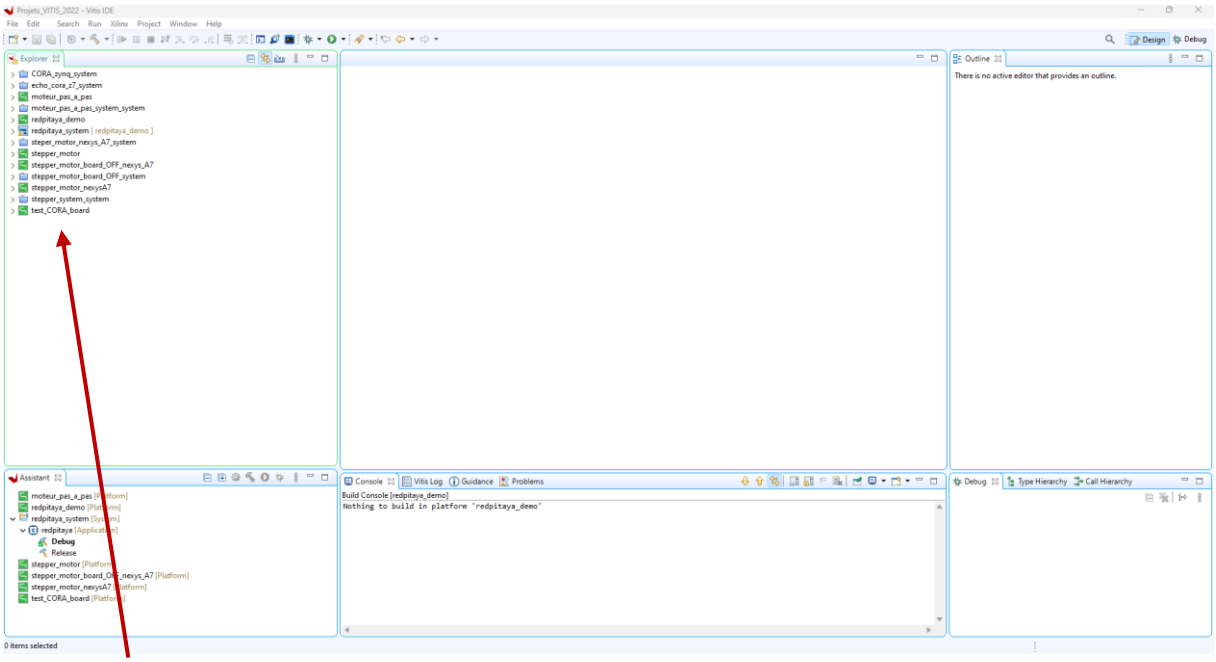
Menu Tools-> Launch Vitis IDE



Lauch Vitis...

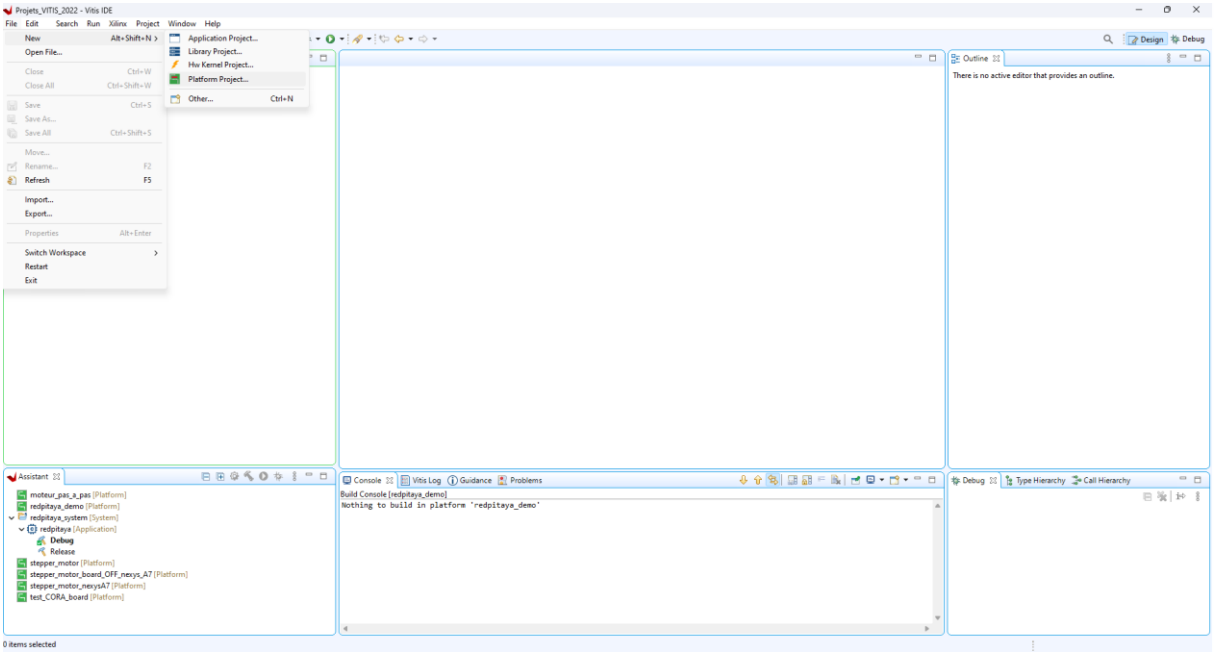


Environnement Vitis

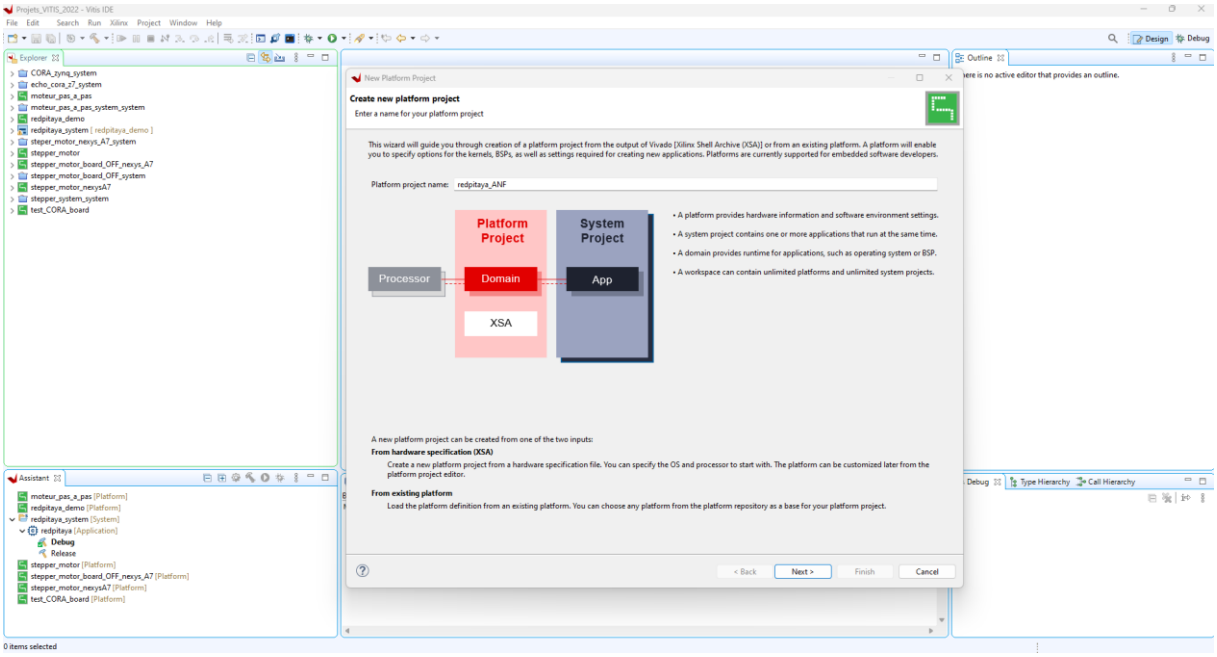


Projets déjà existants...

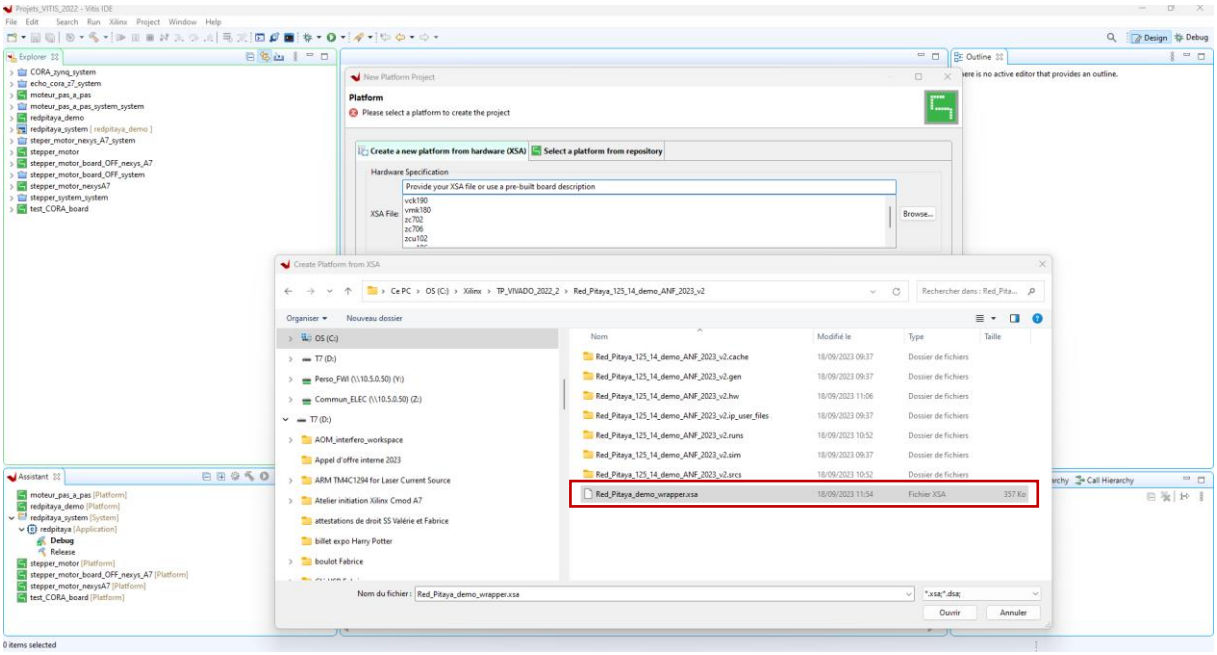
Dans Menu Principal->**File**->**New**-> **Platform Project**



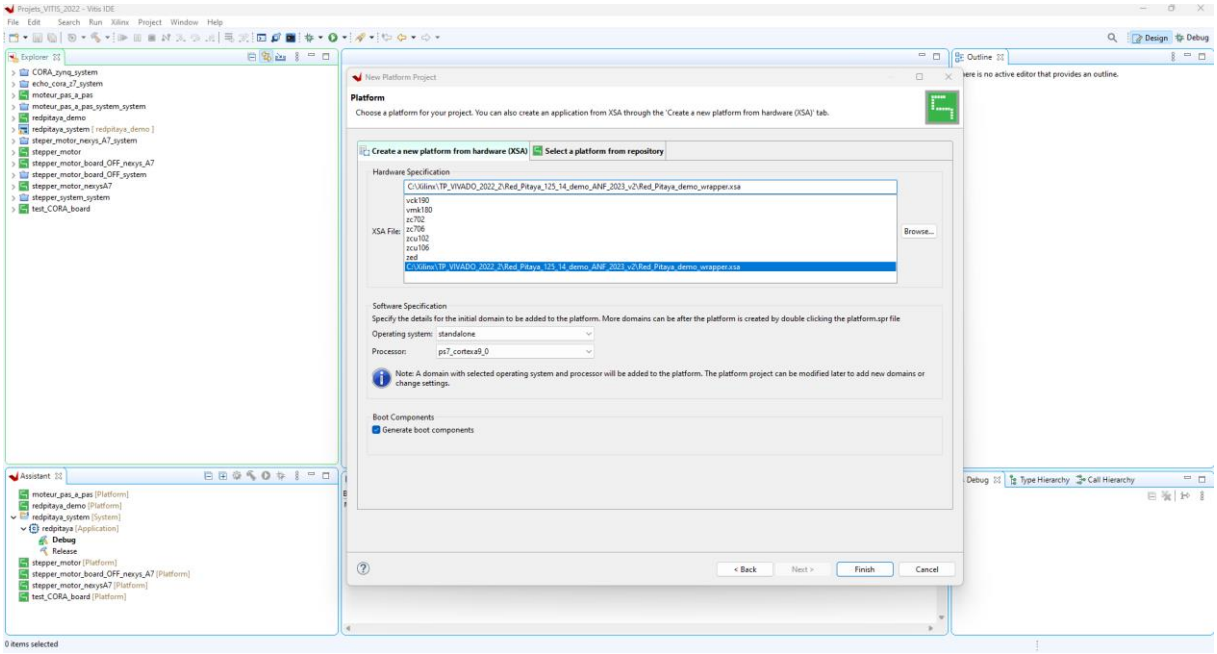
Rename the project-> **redpitaya_ANF-> Next.**



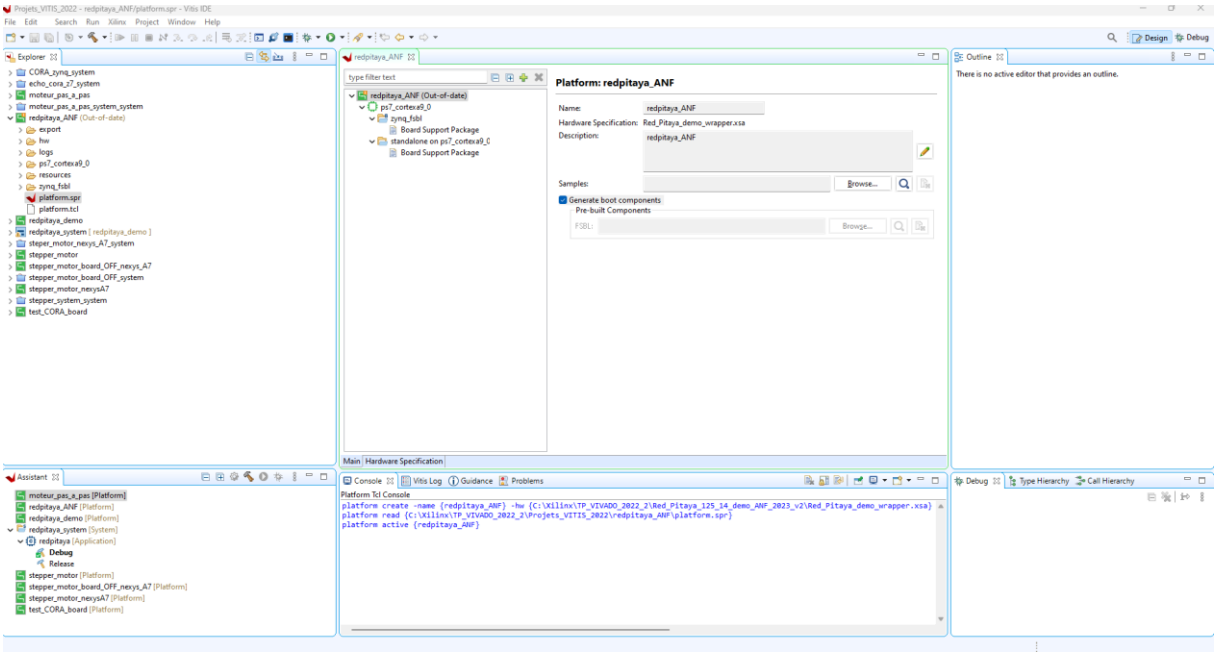
Dans Create a new platform hardware (XSA)-> Browse…



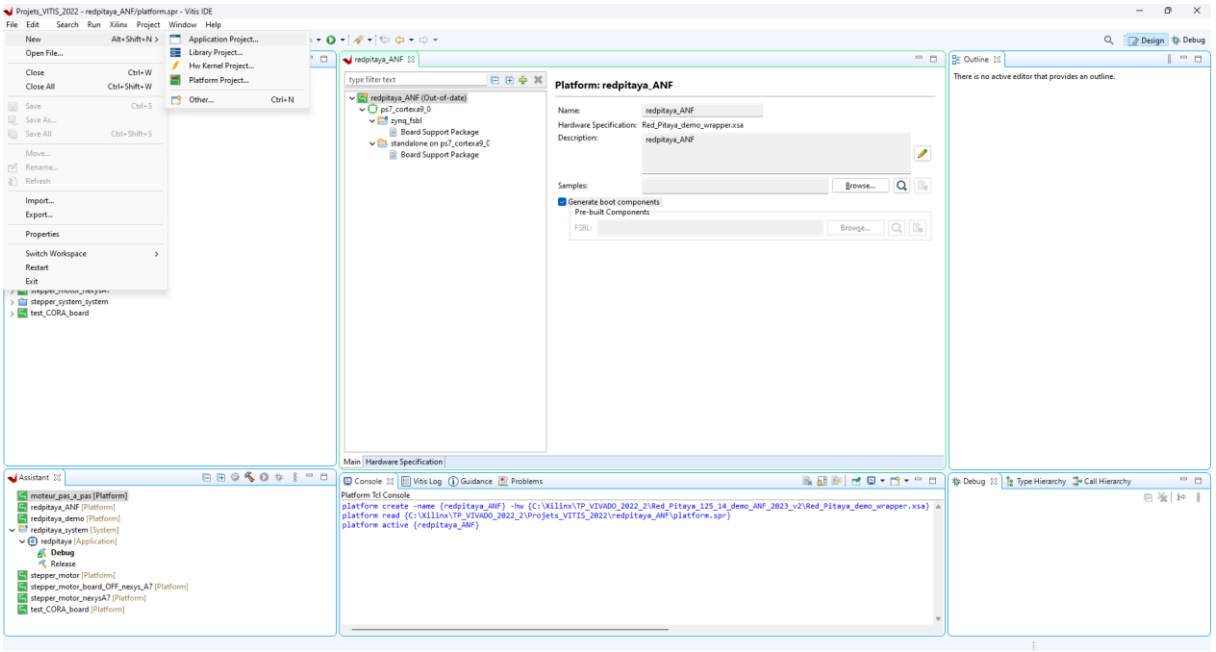
Clic sur-> **Finish**



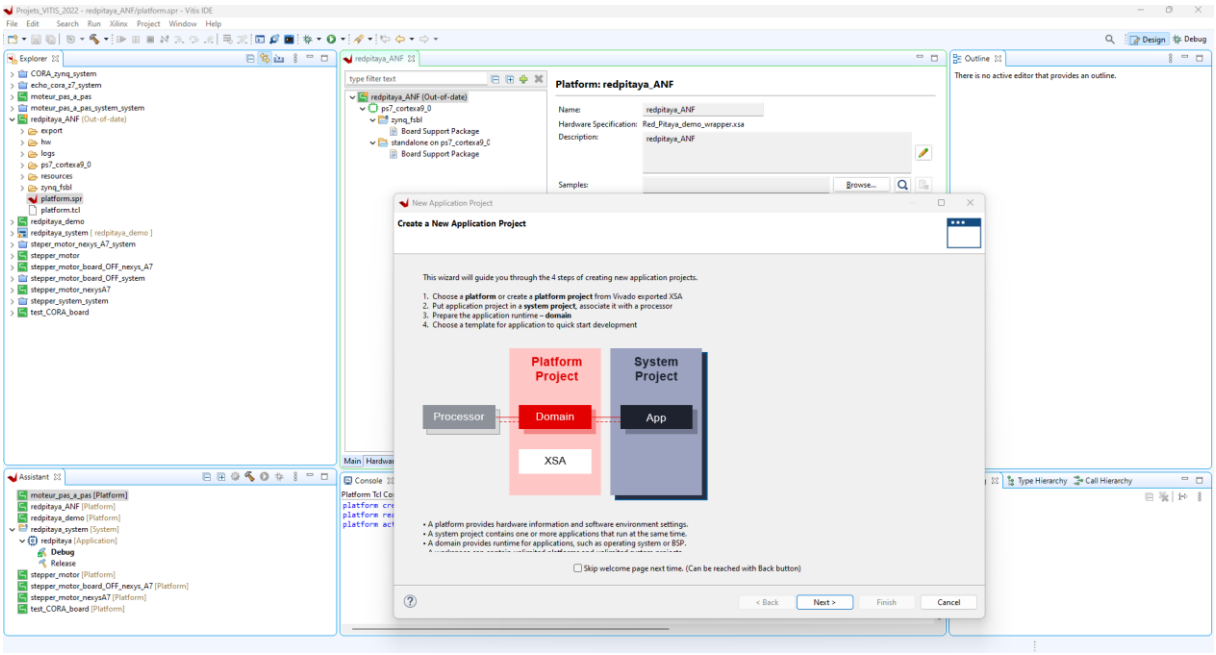
Platform : **redpitaya_ANF**



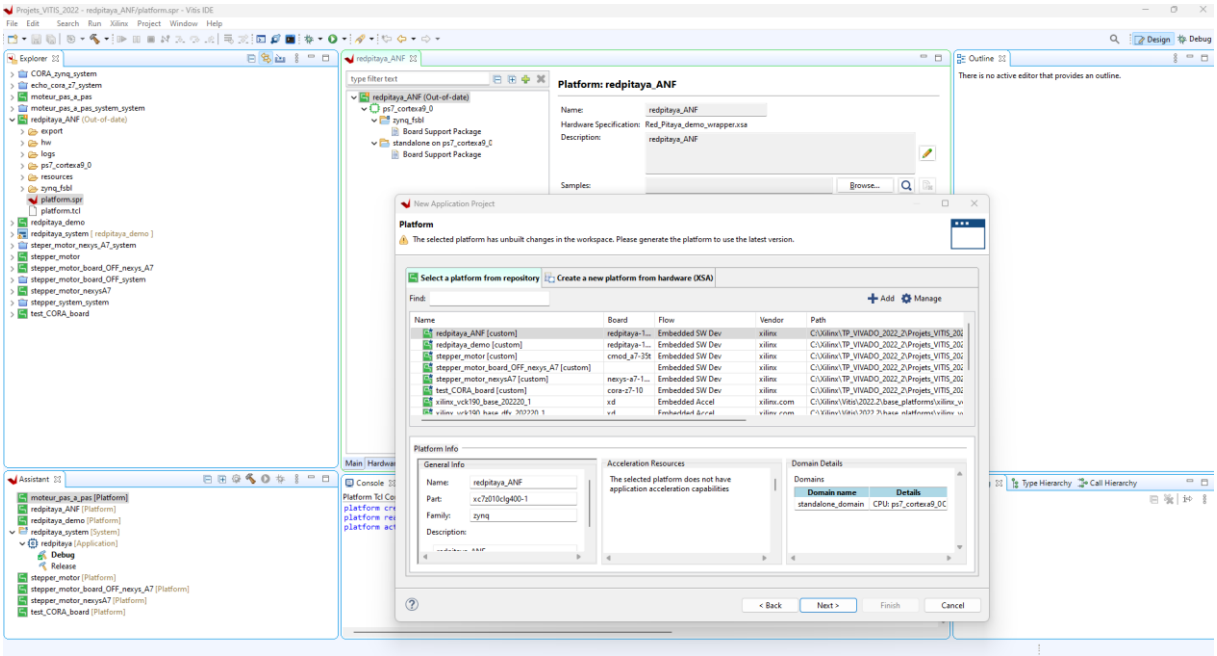
Menu principal->File-> New-> **Application Project**



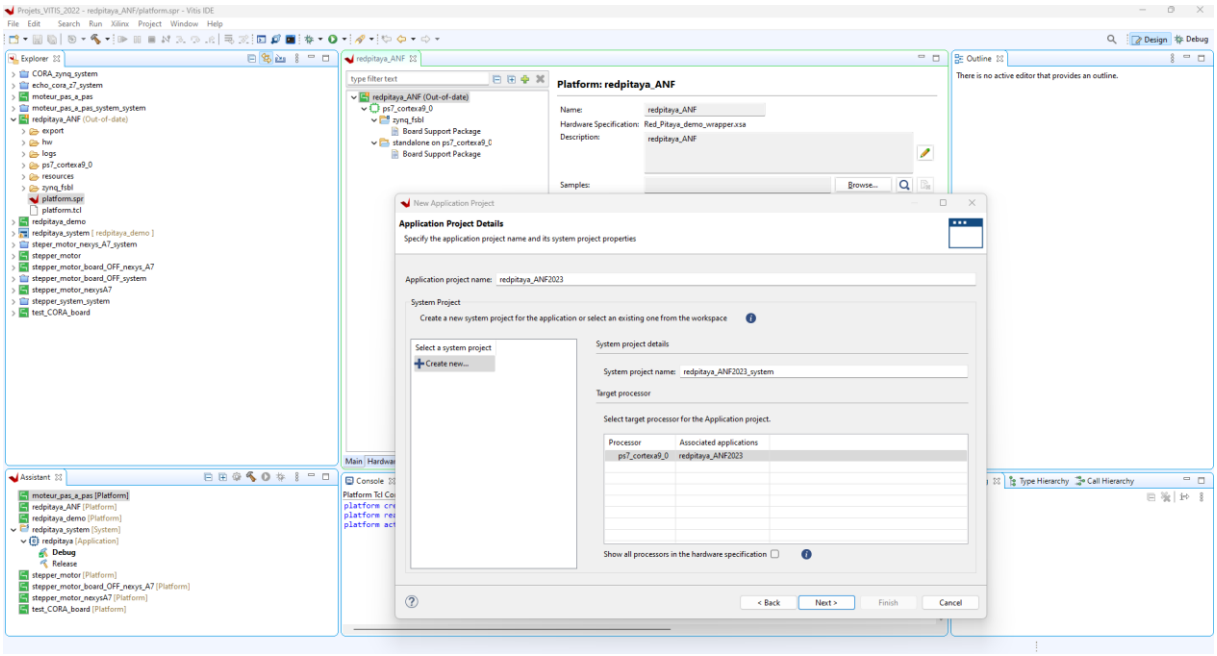
Next…



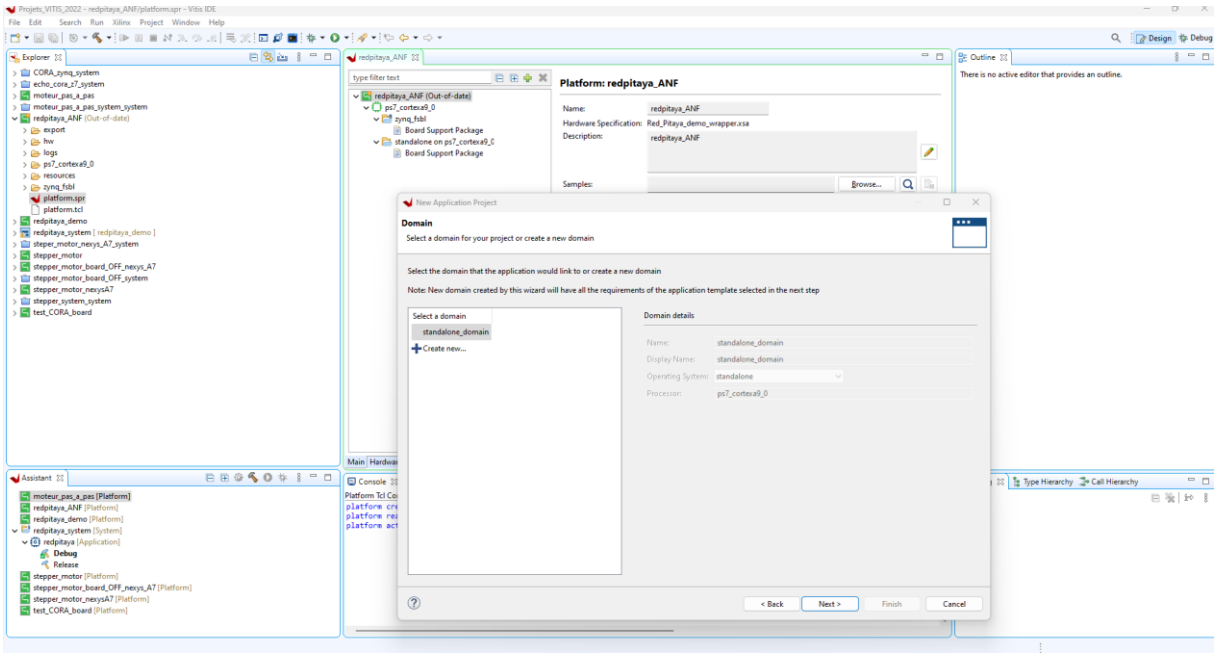
Select a **platform from repository-> Redpitaya_ANF [custom]-> Next.**



Application project name = **redpitaya_ANF2023-> Next**



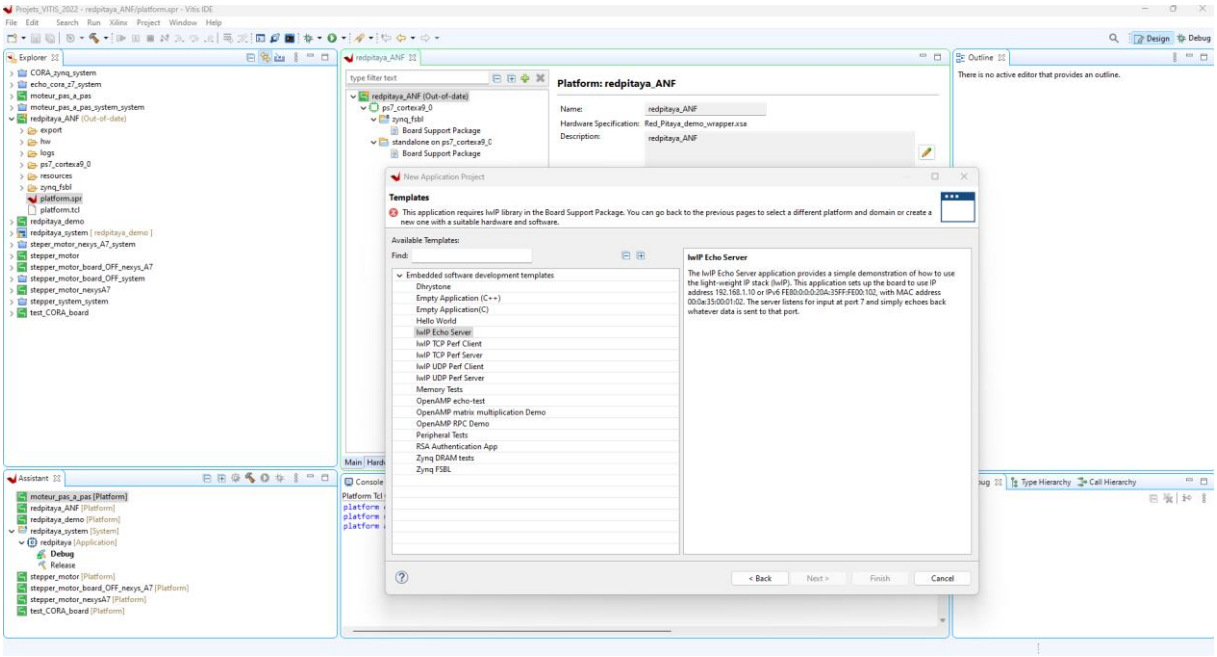
Domain = **standalone_domain-> Next**



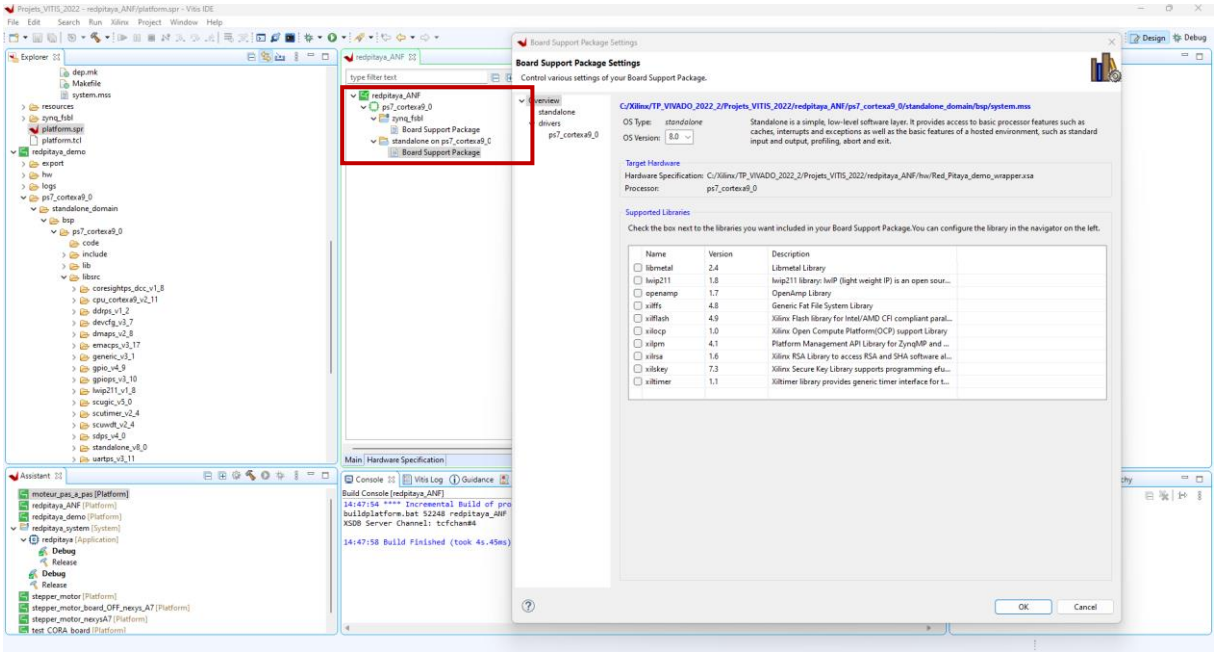
New Application Project-> LwIP Echo Server

La librairie n’est pas active par défaut dans le projet : on doit revenir dans le BSP

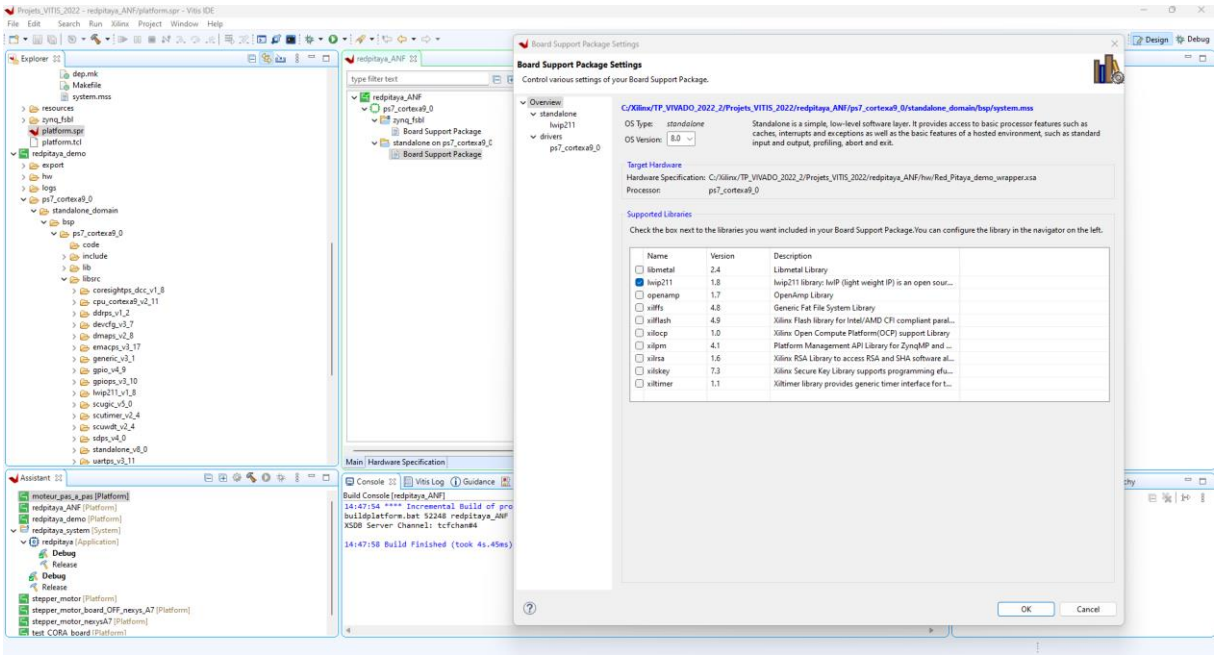
Pour activer lwIP Echo Server : donc faire **Cancel**

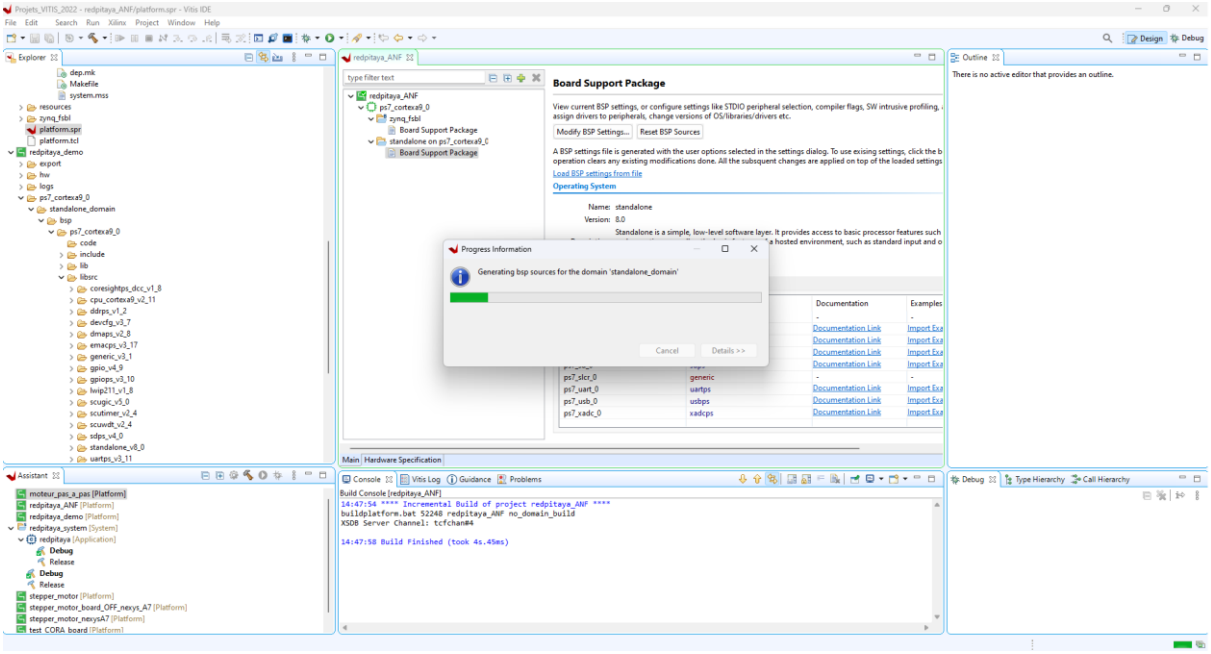


Double clic sur-> **platform.spr** sous la plateforme : redpitaya_ANF et sous **standalone_on_ps7_cortexa9_0- > Board Support Package Settings**

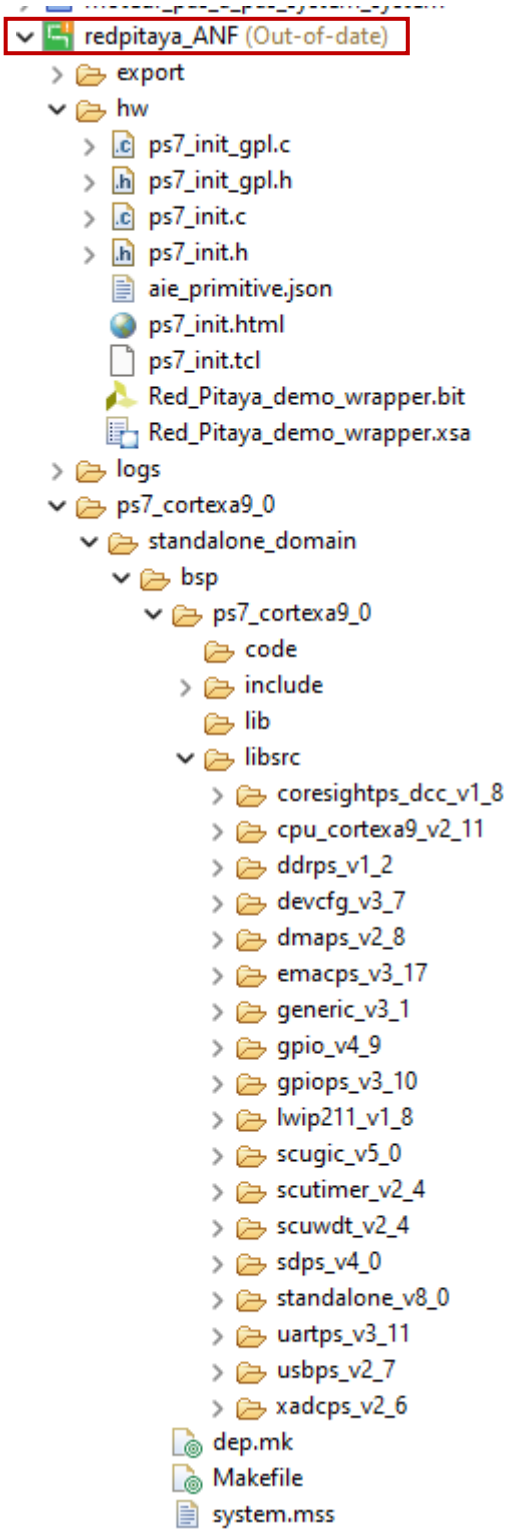


Selectionnez la librairie **lwip211** puis clic sur « **OK** »

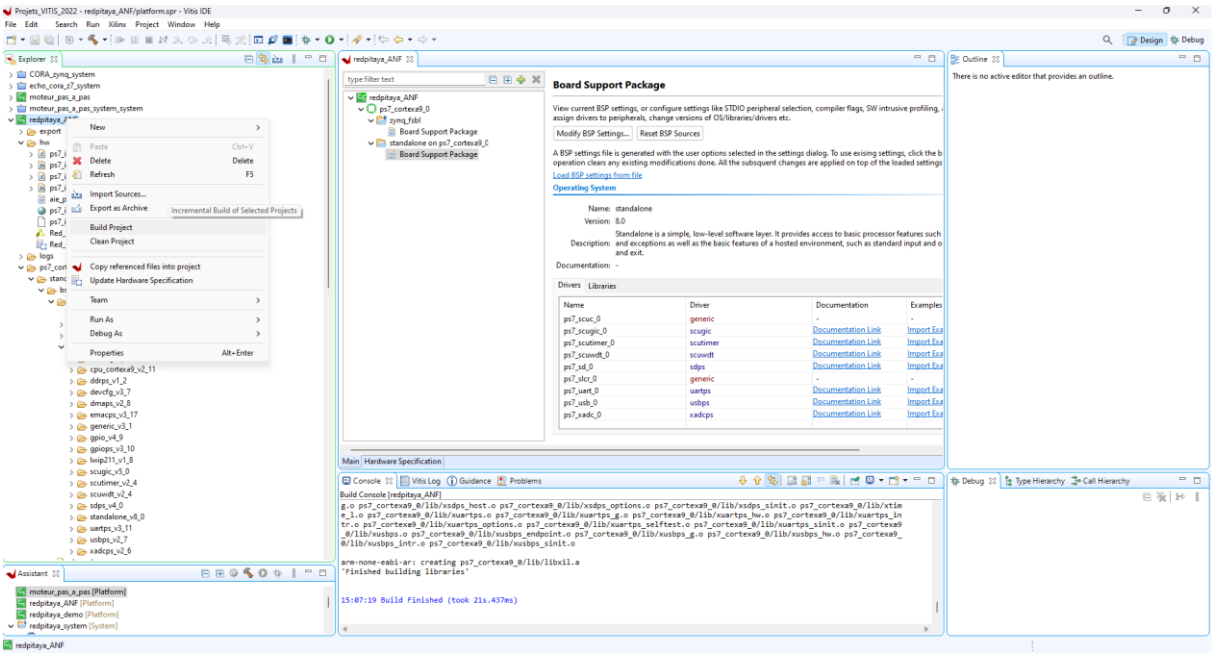




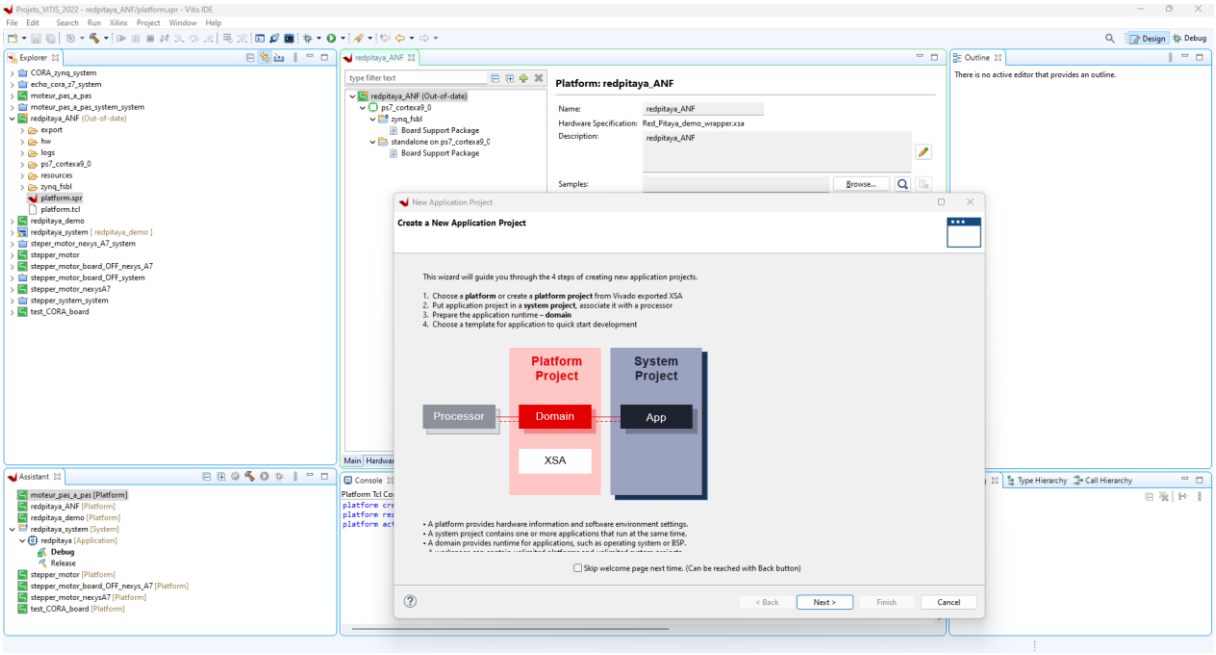
On retrouve installé la librairie **lwip211_v1_8**



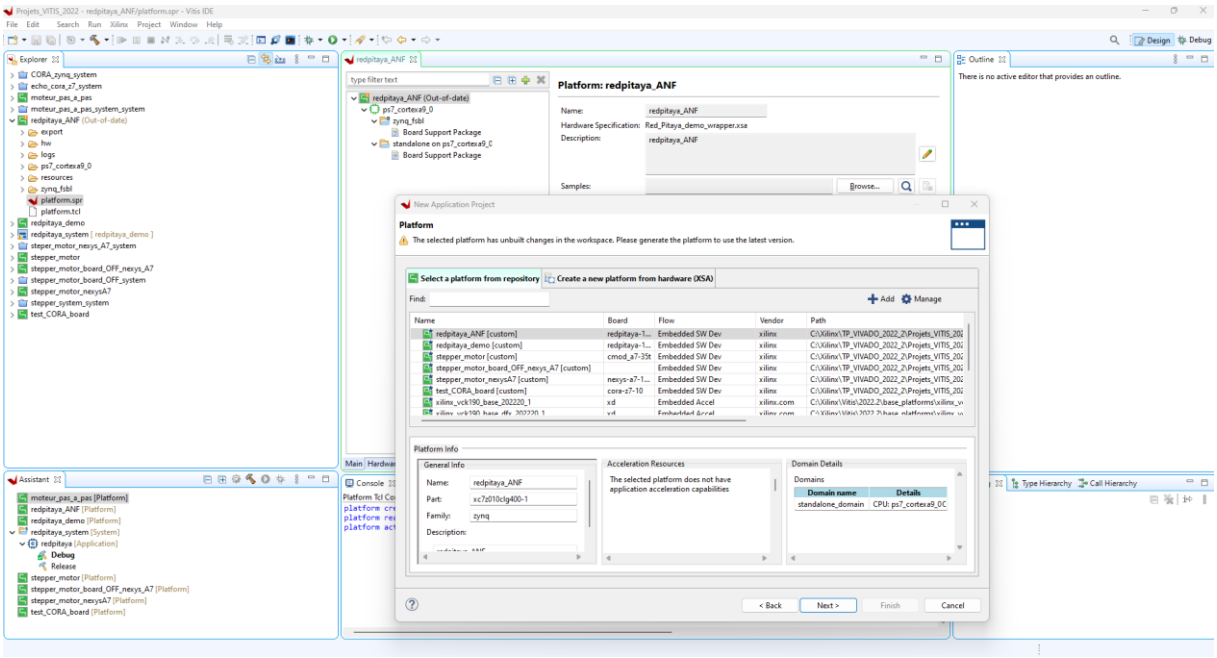
On enregistre **la platform redpitaya_ANF->** Clic droit sur **redpitaya_ANF-> Build Project**



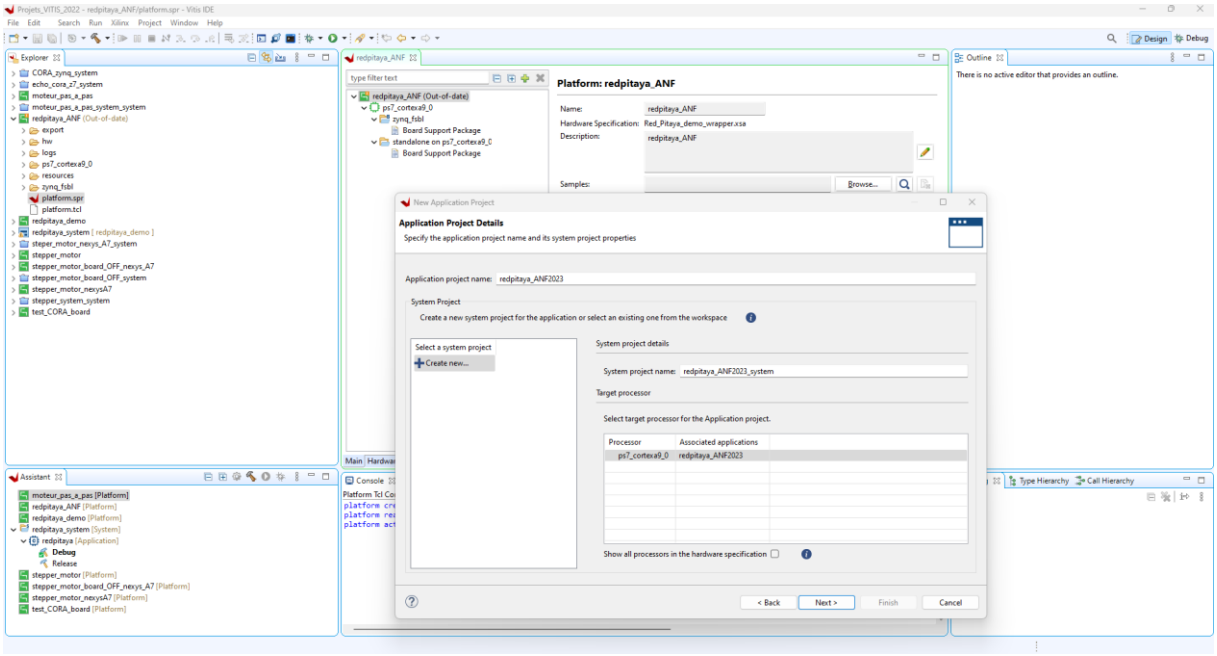
On reprend la précédente procédure : Menu principal->File-> New-> **Application Project**



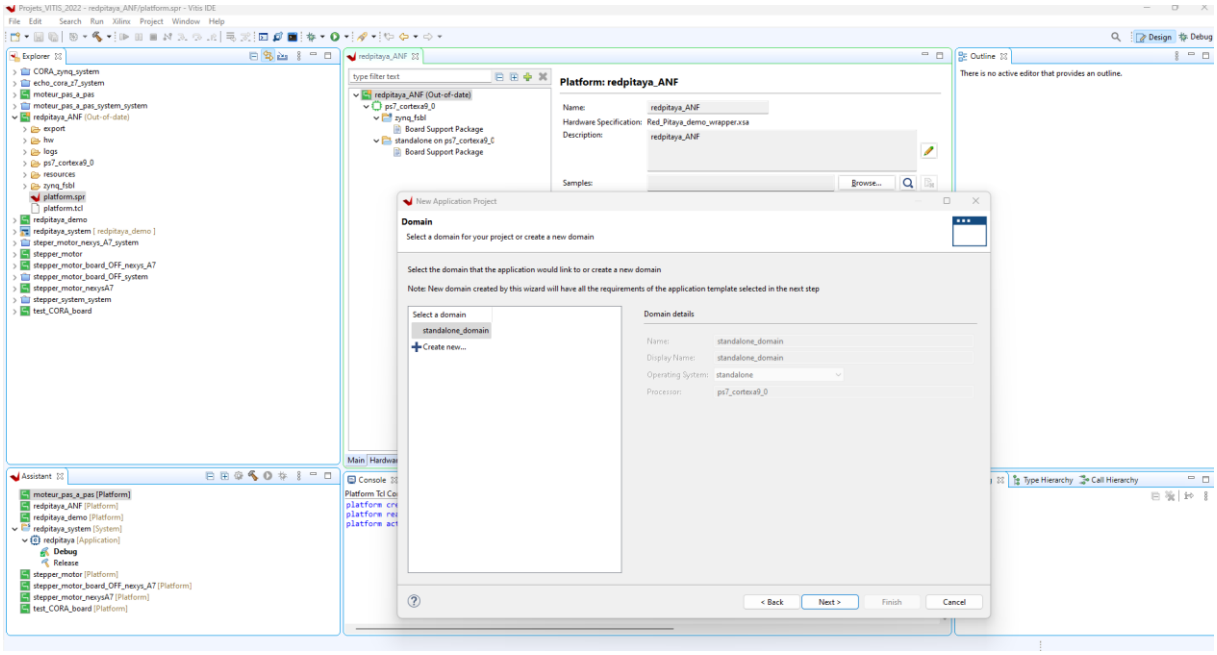
Select a **platform from repository-> Redpitaya_ANF [custom]-> Next.**



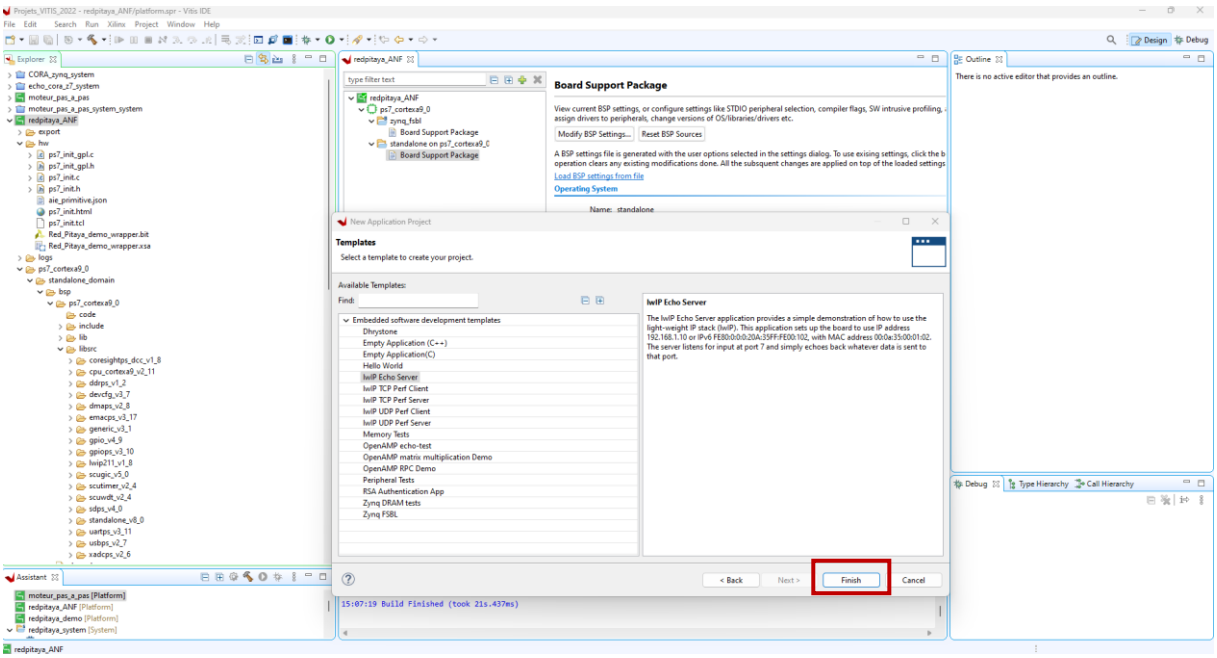
Application project name = **redpitaya_ANF2023-> Next**



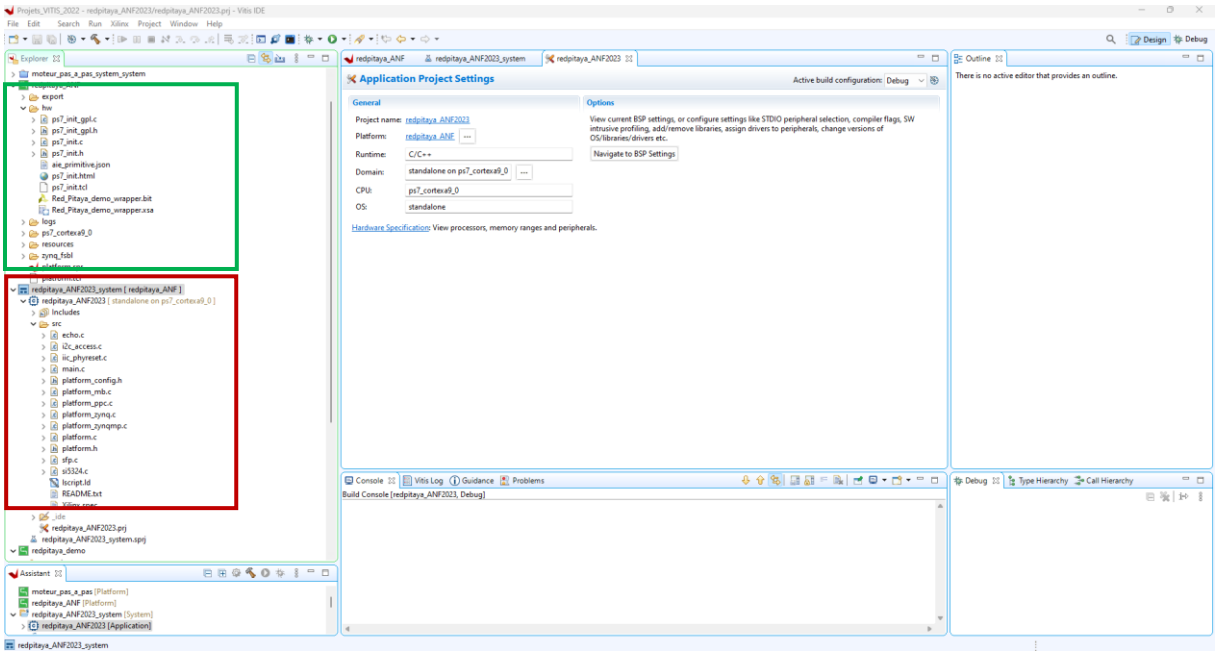
Domain = **standalone_domain-> Next**



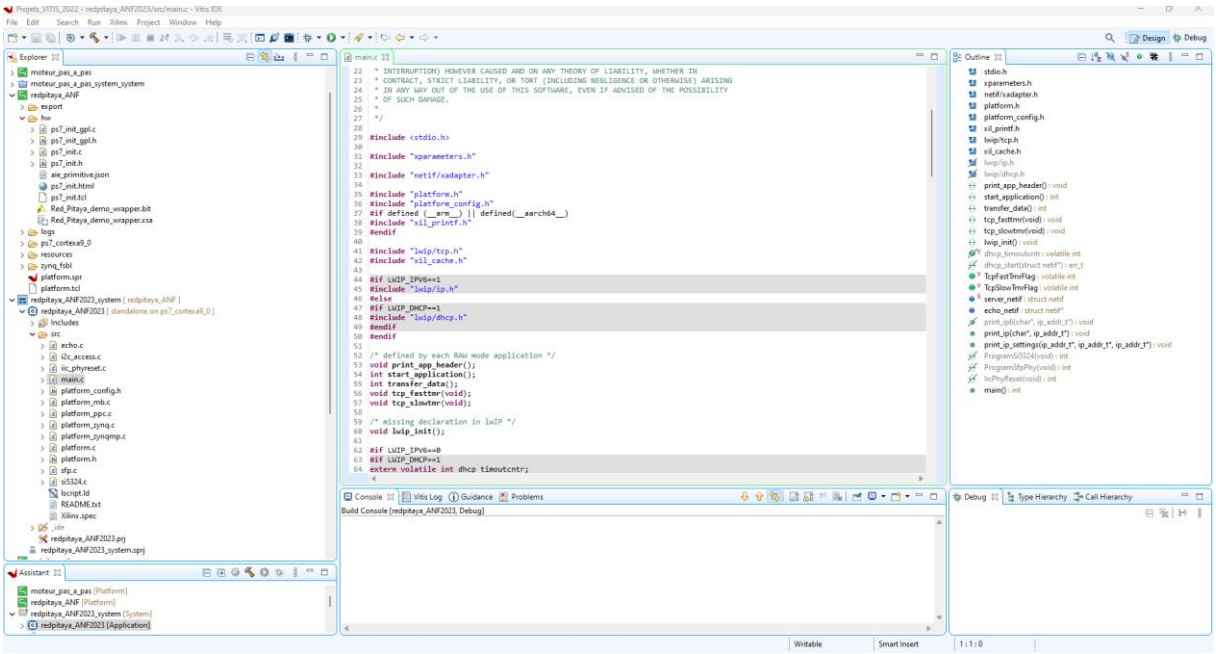
Sélectionnez **lwIP Echo Server** puis **Finish**



Le projet complet : En vert la platform HARDWARE en rouge la partie SOFTWARE Zynq avec l'exemple Echo Ethernet



On va modifier les fichiers sources du programme : main.c et echo.c et insérer deux nouveaux fichiers : leds_RGB.c Led_RGB.h au projet pour la gestion et la communication avec les leds de la RedPitaya en Ethernet.



Fermer VITIS 2022_2 !

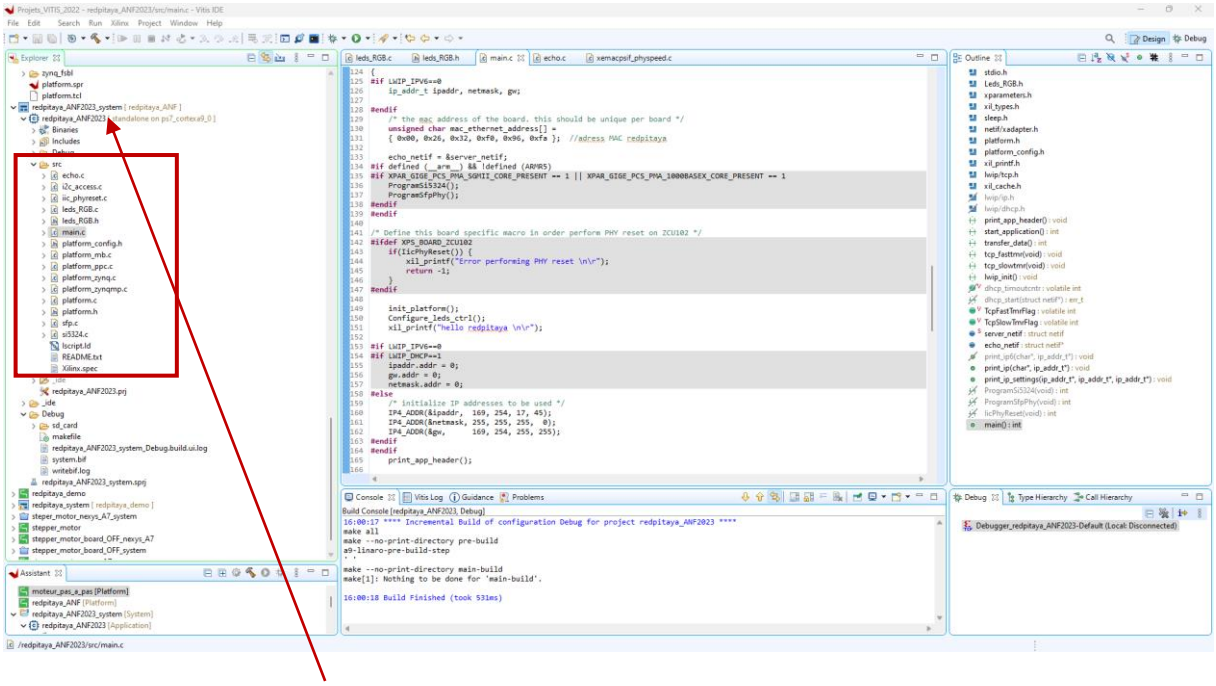
Télécharger les fichiers main.c, leds_RGB.c, leds.RGB.h, echo.c avec le lien

https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

Copier- coller dans le répertoire du projet Vitis les fichiers ci-dessus :

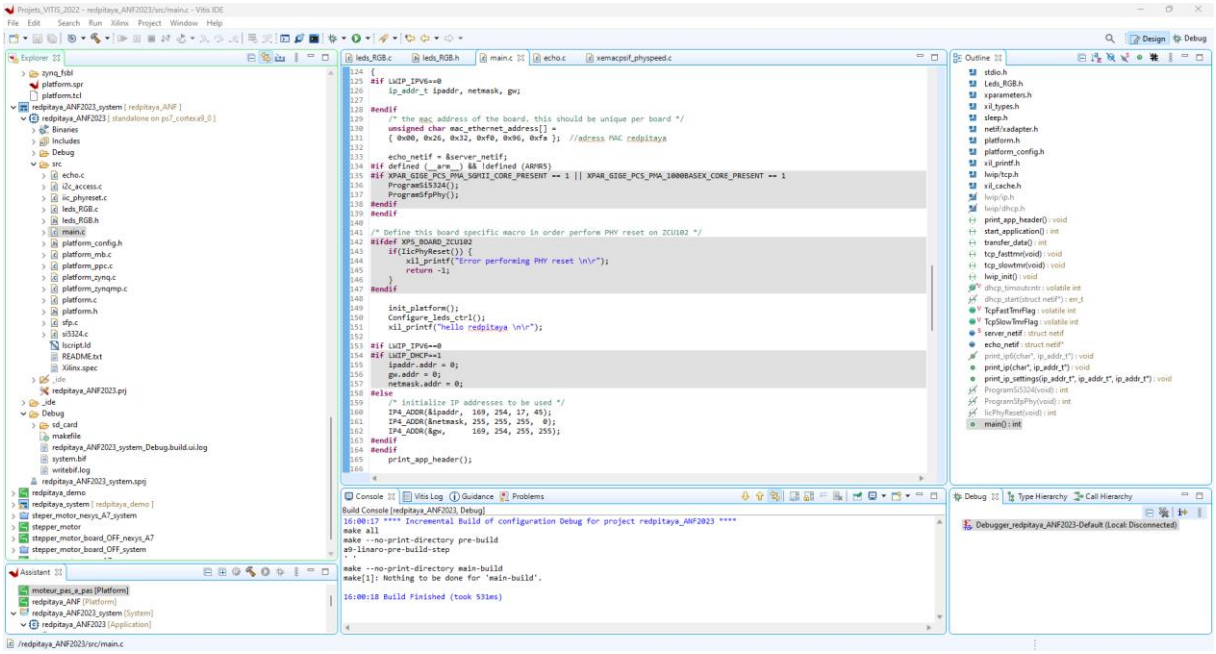
Le chemin d'accès devrait ressembler à ceci :

C:\Xilinx\TP_VIVADO_2022_2\Projets_VITIS_2022\redpitaya_ANF2023\src



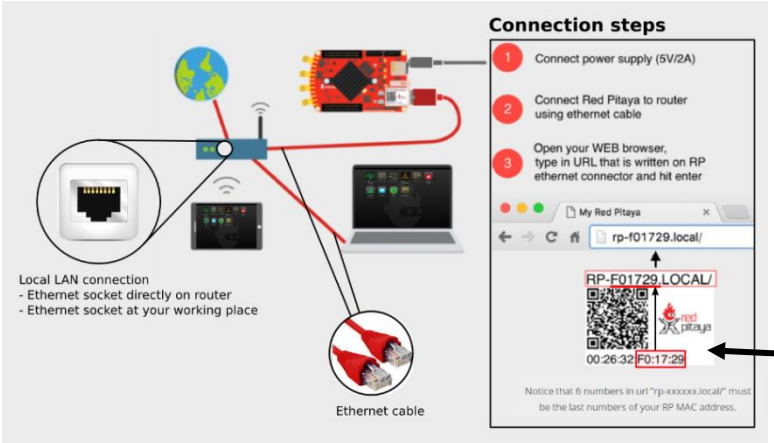
Sauvegarder: **Ctrl +SHIFT + S** puis clic droit sur redpitaya_ANF2023_system-> **Build Project**

Dans le main.c On vérifie l'adresse IP et l'adresse MAC de la redpitaya

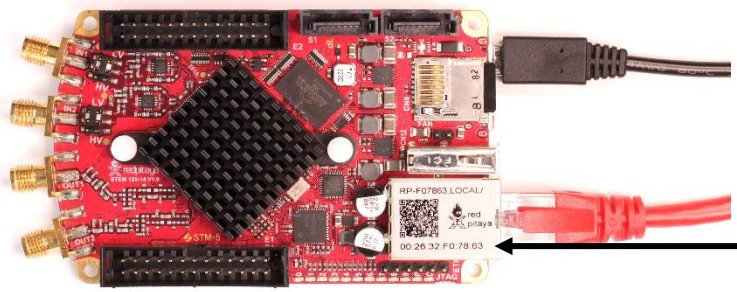


Pour retrouver l'adresse IP et MAC de la Redpitaya il faut la carte SD insérer et le système d'exploitation Red Pitaya préchargé :

https://redpitaya.readthedocs.io/en/latest/quickStart/SDcard/SDcard.html



Adresse MAC



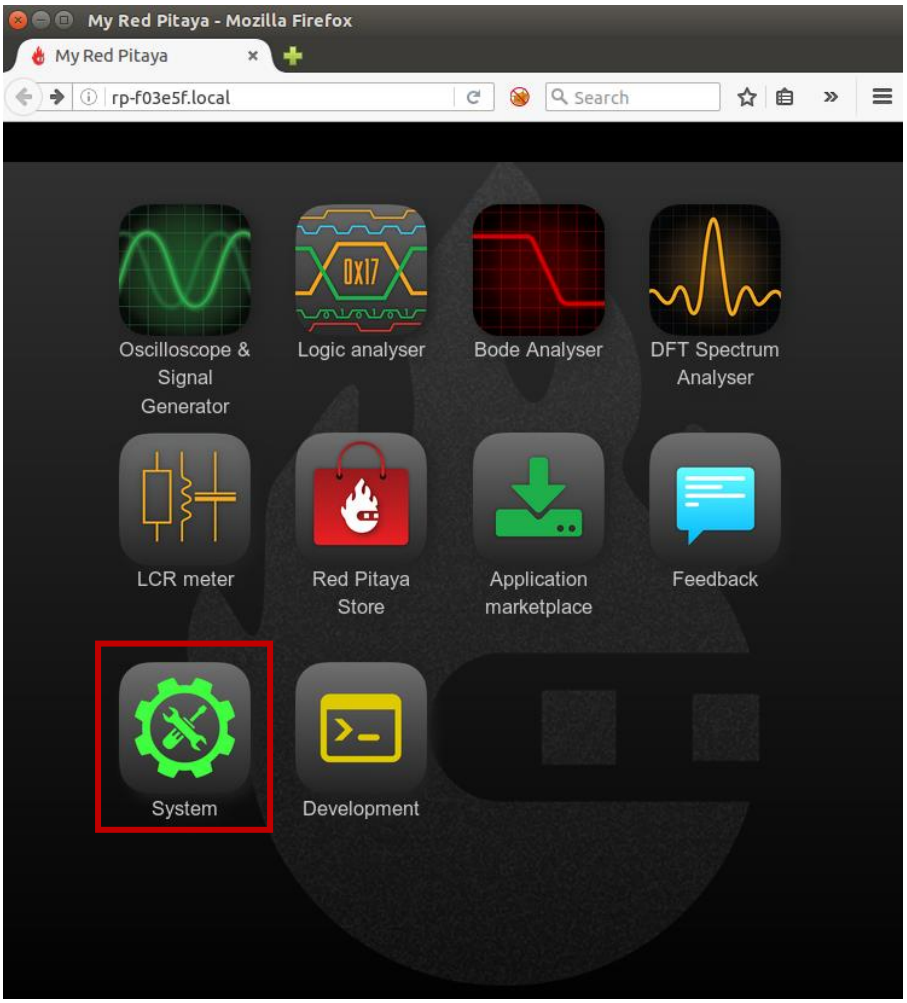
Procédure pour une connexion Ethernet directe :

Windows (le service Bonjour doit être installé pour Win 7/8)

1. Connectez le câble Ethernet et attendez env. 30 secondes
2. Ouvrez le navigateur Web et tapez **rp-xxxxxx.local/** dans le champ URL

Linux/Ubuntu

1. Ouvrez les paramètres réseau, modifiez la connexion et, pour le réseau LAN, sélectionnez Méthode **Partager avec d'autres ordinateurs** sous Paramètres IPv4.
2. Connectez le câble Ethernet et attendez env. 30 secondes
3. Ouvrez le navigateur Web et tapez **rp-xxxxxx.local/** dans le champ URL



Adresse IP dans System

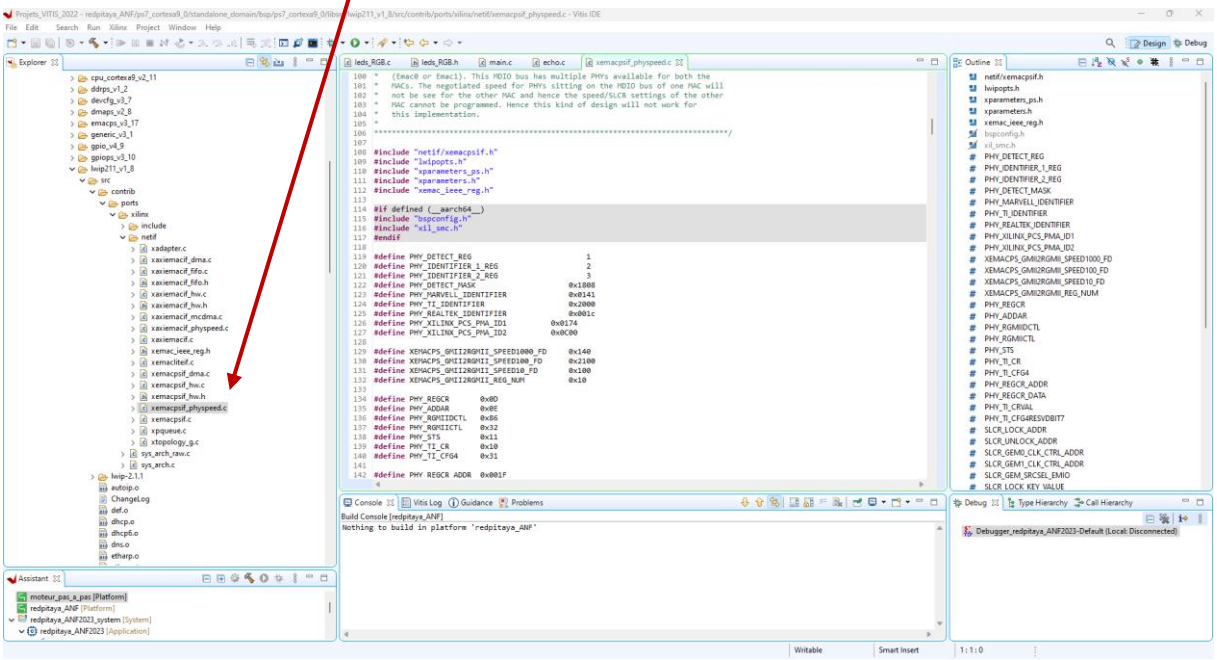
Une dernière chose est nécessaire pour la communication Ethernet avec la redpitaya et notre programmation Bare Metal, c'est de modifier le fichier xemacpsif_physpeed.c dans le BSP de la platform project Redpitaya**_ANF.**

Chemin d'accès du fichier xemacpsif_physpeed.c :

C:\Xilinx\TP_VIVADO_2022_2\Projets_VITIS_2022\redpitaya_ANF\ps7_cortexa9_0\standalone_domain\bsp\ps7_cortexa9_0\libsrc\lwip211_v1_8\src\contrib\ports\xilinx\netif\ **xemacpsif_physpeed.c**

Le contrôleur HARDWARE PHY LANTIQ sur la Redpitaya n'est pas inclus dans la librairie **lwip211_v1_8….** Gasp ! :(

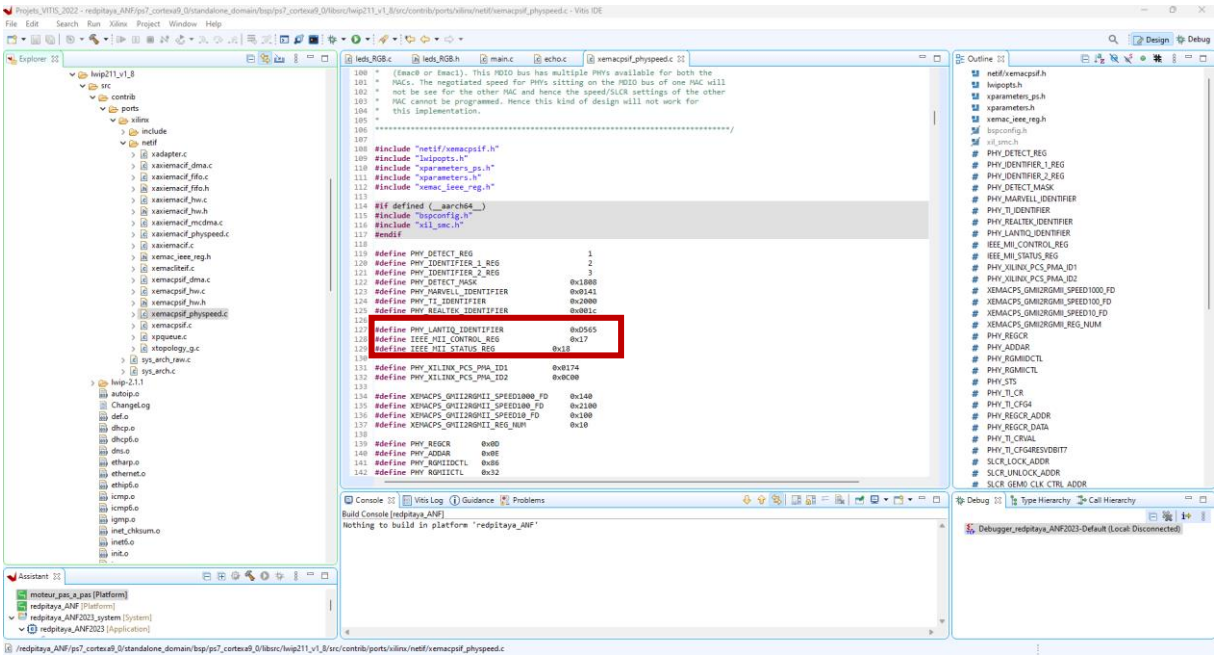
Ouvrir le fichier **xemacpsif_physpeed.c** dans Vitis comme ci-dessous :



Récupérer le fichier xemacpsif_physpeed.c sur *https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14*

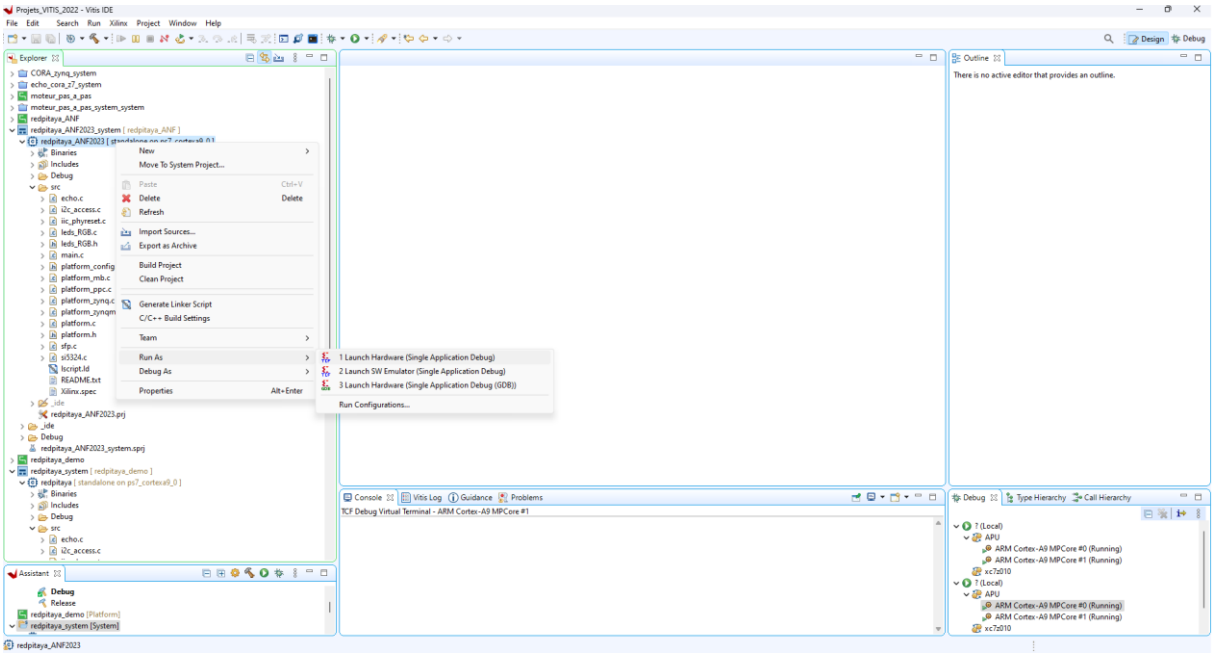
Effacer l'ancien contenu et copier- coller entièrement le nouveau dans le fichier ouvert sur VITIS ci-dessus.

Sauvegarder: **Ctrl +SHIFT + S** puis clic droit sur redpitaya_ANF-> **Build Project**

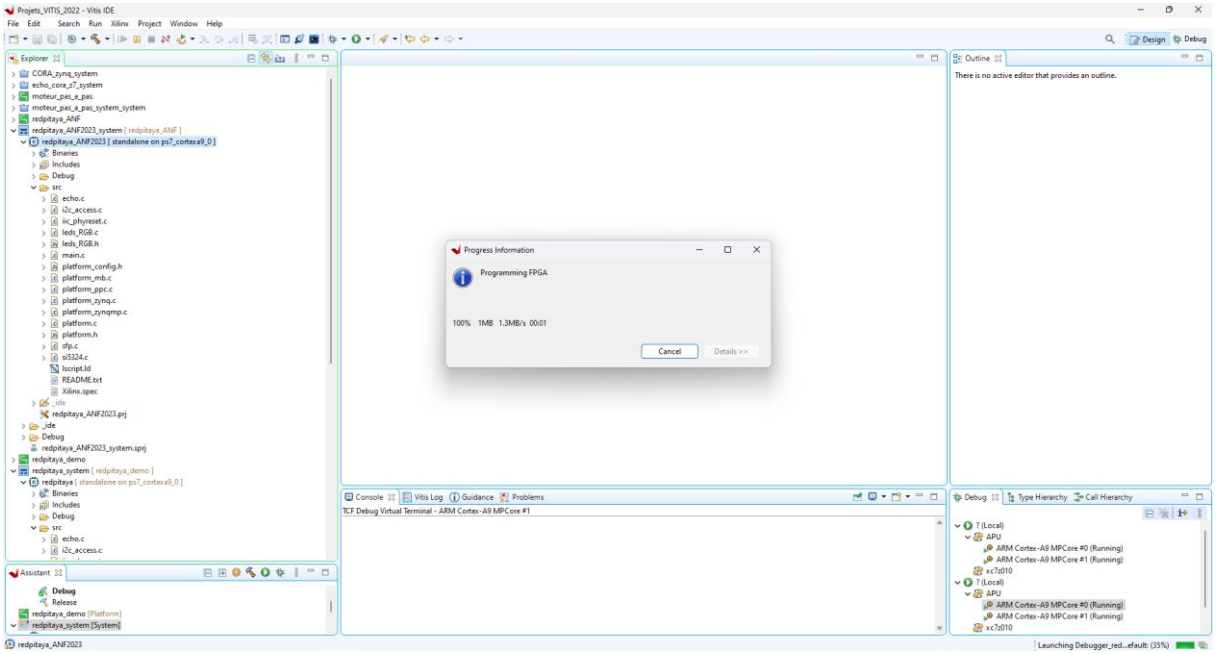


Dans menu Explorer de Vitis-> Clic droit sur **redpitaya_ANF2023 [standalone on ps7_cortexa9_0]**

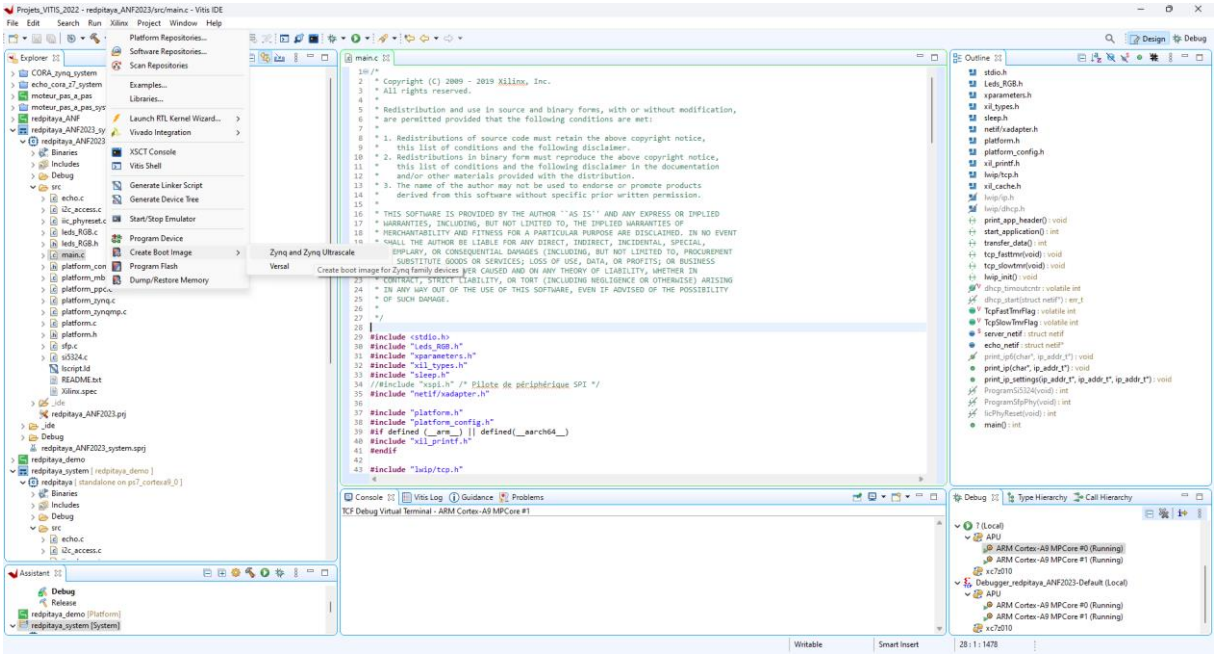
Run As-> Launch Hardware (single application debug)



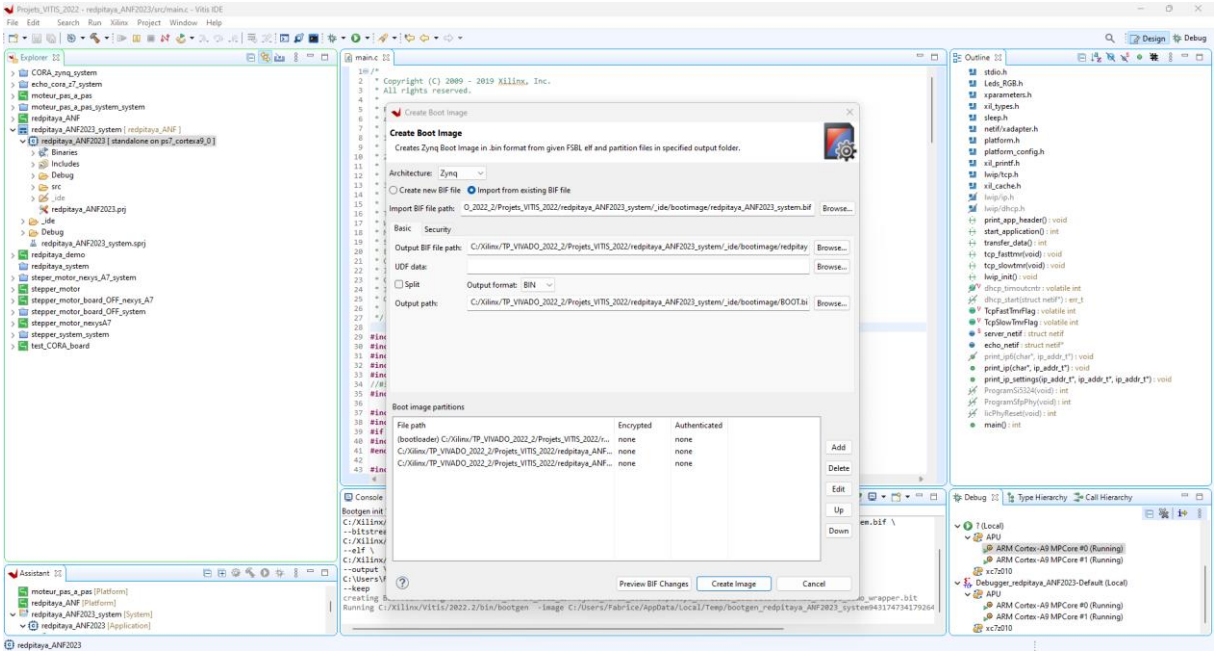
Launch Hardware...



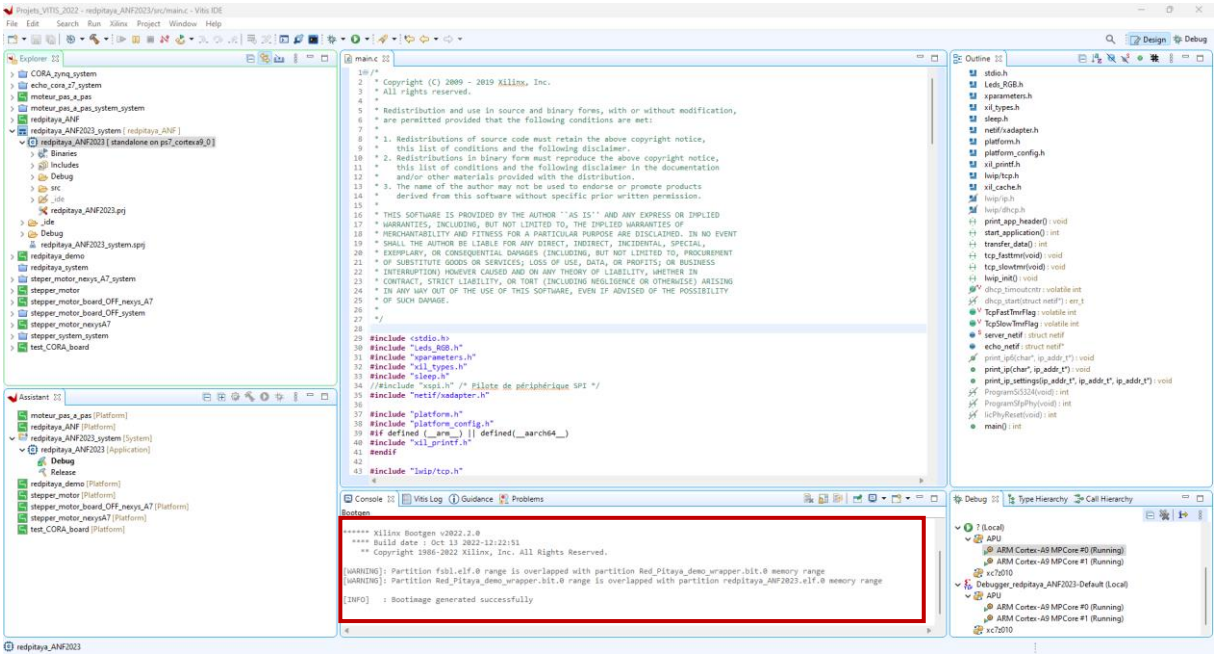
On va sauvegarder le projet complet sur une carte SD en faisant une image du projet : Dans le menu principal-> **Xilinx**-> **Create Boot Image-> Zynq**



C:/Xilinx/TP_VIVADO_2022_2/Projets_VITIS_2022/redpitaya_ANF2023_system/_ide/bootimage/redpitaya_ANF2023_system.bif



Dans Create Boot Image-> **Create Image**



Bootimage generated successfully!

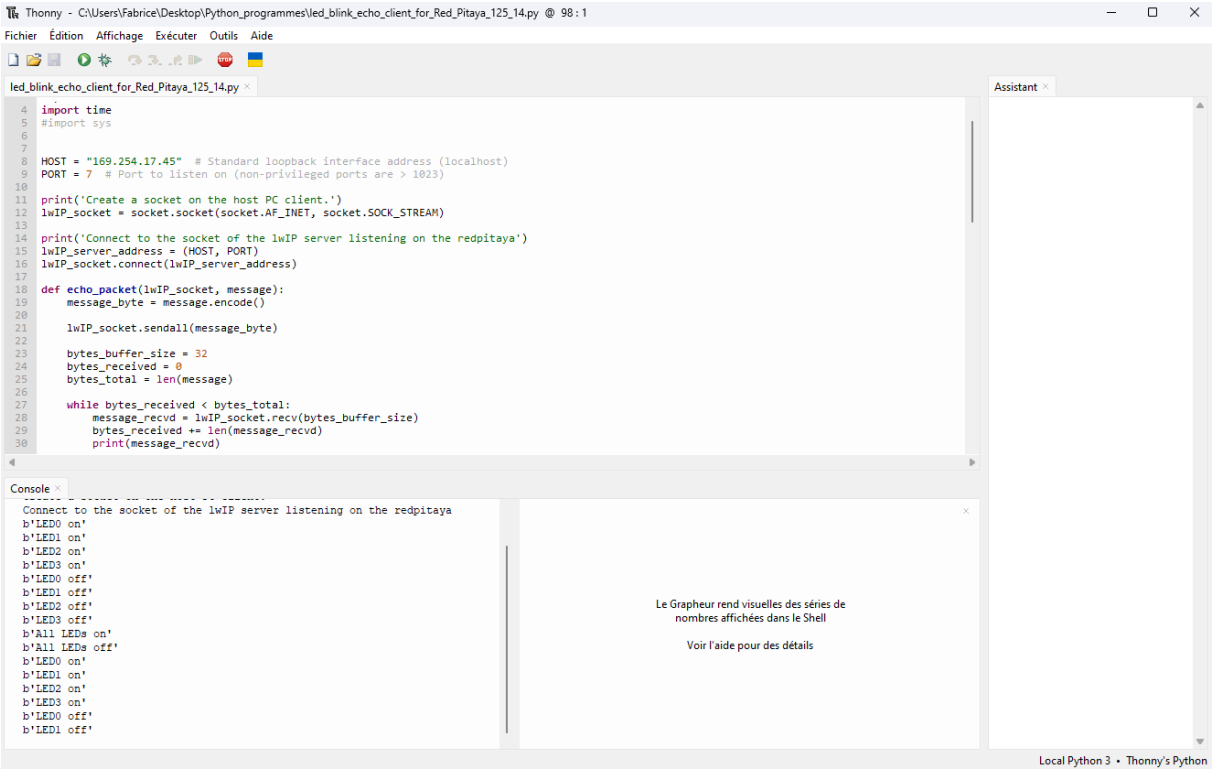
Fichier Boot dans:
C:\Xilinx\TP_VIVADO_2022_2\Projets_VITIS_2022\redpitaya_ANF2023_system_ide\bootimage**BOOT.bin**

Copier-coller le fichier **BOOT.bin** sur une micro-SD via un adaptateur SD to micro -SD et l'insérer dans l'emplacement micro-SD de la Redpitaya.

Redémarrer la Redpitaya et tester le programme.

Fin de la Démo ADC/DAC avec Echo Ethernet !

Script Python Echo ethernet




```
#-*- coding : utf-8-*-

import tkinter        # import de tkinter

import socket

import time

#import sys

HOST = "169.254.17.45"  # Standard loopback interface address (localhost)

PORT = 7  # Port to listen on (non-privileged ports are > 1023)

print('Create a socket on the host PC client.')

lwIP_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

print('Connect to the socket of the lwIP server listening on the redpitaya')

lwIP_server_address = (HOST, PORT)

lwIP_socket.connect(lwIP_server_address)


def   echo_packet(lwIP_socket, message):

        message_byte = message.encode()

       lwIP_socket.sendall(message_byte)
```

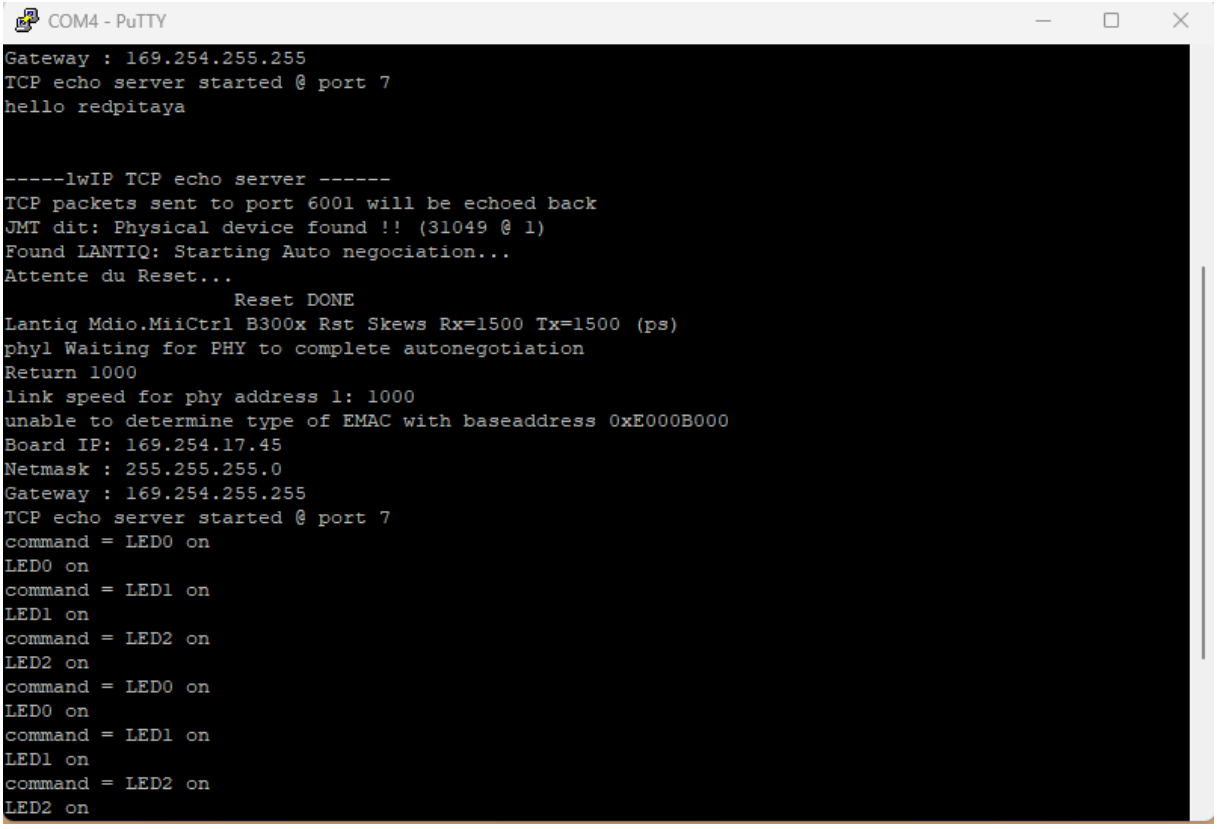
```
    bytes_buffer_size = 32
    bytes_received = 0
    bytes_total = len(message)
    while bytes_received < bytes_total:
        message_recvd = lwIP_socket.recv(bytes_buffer_size)
        bytes_received += len(message_recvd)
        print(message_recvd)

try:
  while True:
      message = 'LED0 on'
      echo_packet(lwIP_socket, message)
      time.sleep(0.5)
      message = 'LED1 on'
      echo_packet(lwIP_socket, message)
      time.sleep(0.5)
      message = 'LED2 on'
      echo_packet(lwIP_socket, message)
      time.sleep(0.5)
      message = 'LED3 on'
      echo_packet(lwIP_socket, message)
      time.sleep(0.5)
      message = 'LED0 off'
      echo_packet(lwIP_socket, message)
      time.sleep(0.5)
      message = 'LED1 off'
      echo_packet(lwIP_socket, message)
      time.sleep(0.5)
      message = 'LED2 off'
      echo_packet(lwIP_socket, message)
      time.sleep(1)
      message = 'LED3 off'
      echo_packet(lwIP_socket, message)
```

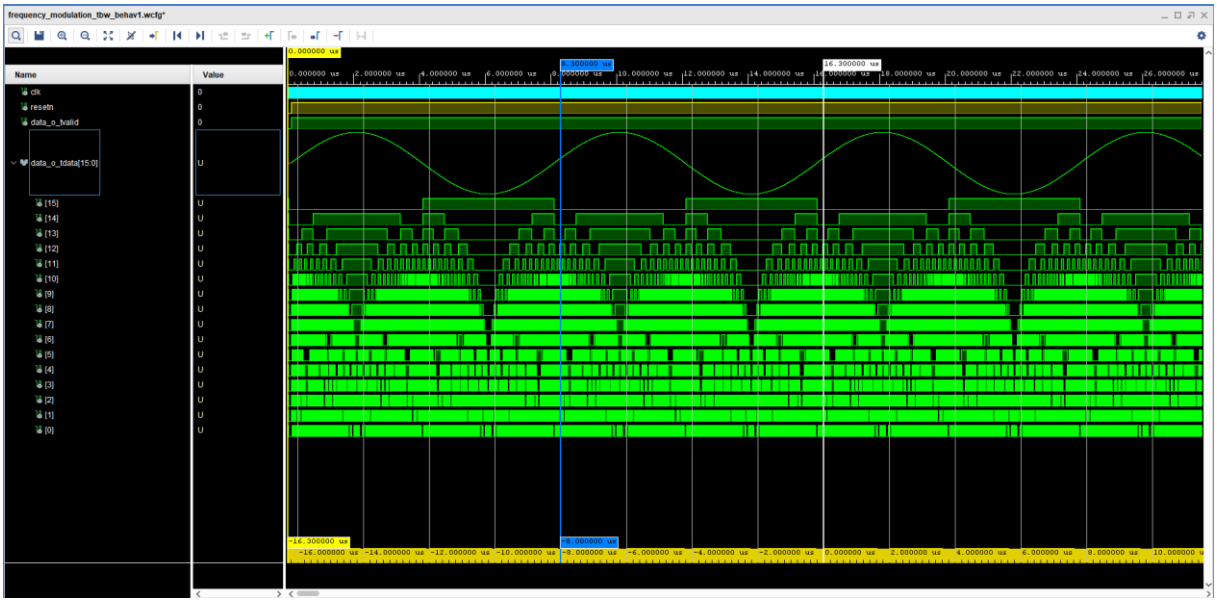
```
        time.sleep(1)
        message = 'All LEDs on'
        echo_packet(lwIP_socket, message)
        time.sleep(1)
        message = 'All LEDs off'
        echo_packet(lwIP_socket, message)
        time.sleep(1)

finally:
    print('Closing the socket from the host PC client side...')
    lwIP_socket.close()
```

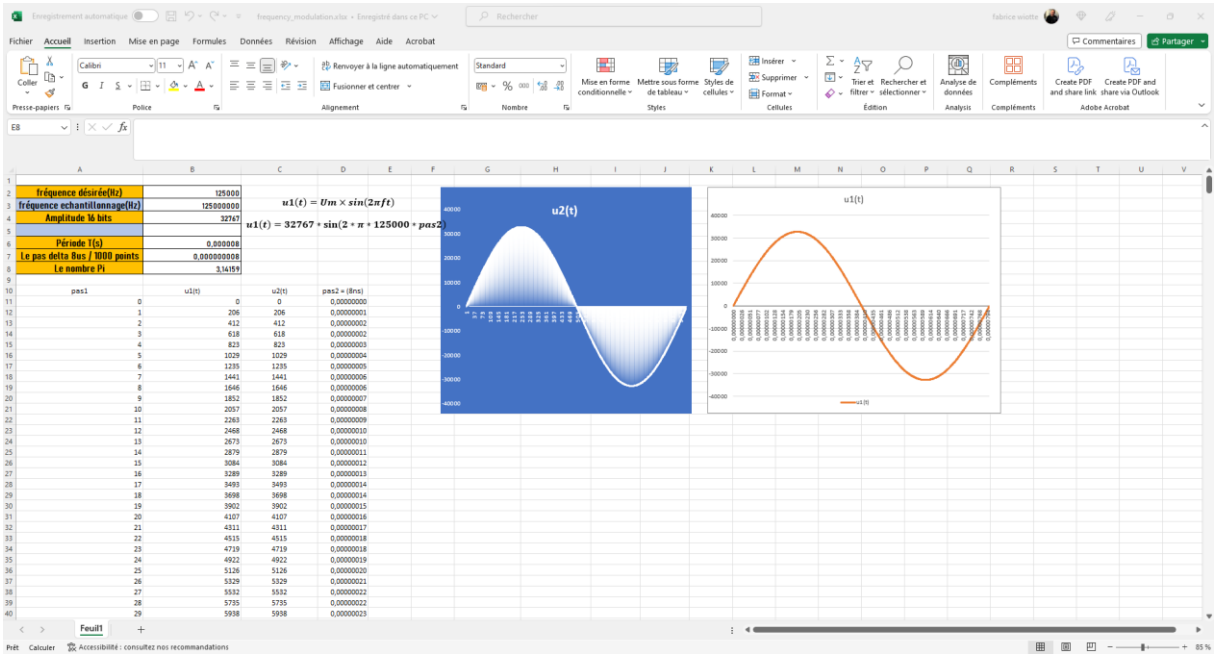
Ouvrir Putty et le port USB-série (COMx) correspondant à la carte Redpitaya détecté et visualiser les infos ci-dessous.



Test Bench module VHDL frequency_modulation@125KHz:



Calcul des points sur Excel :



Calcul des points sous Matlab :