4ème Ecole Technologique du réseau des Electroniciens

Detection synchrone

Principe, structure et applications

Fabrice Wiotte
fabrice.wiotte@univ-paris13.fr

Avec le soutien de la **M**ission pour les **I**nitiatives **T**ransverses et **I**nterdisciplinaires
et le réseau national des électroniciens du CNRS

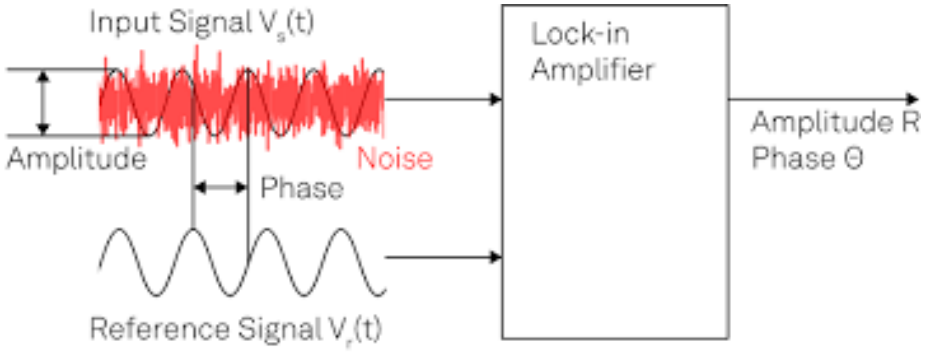
- Lock-in Amplifier (amplificateur à détection synchrone)
- Rappel sur le principe d'une détection synchrone
- Structure
- Application : démodulation d'amplitude AM
- Exemple d'utilisation
- Démodulation synchrone numérique : carte Red Pitaya
- Lien vers prise en main VIVADO 2022.2 et VITIS 2022.2

Historique

Lock-in Amplifier (**amplificateur à verrouillage**)

Les amplificateurs lock-in (détection synchrones) ont été inventés dans les années 1930 et commercialisés au milieu du 20e siècle en tant qu'instruments électriques capables d'extraire les amplitudes et les phases des signaux dans des environnements extrêmement bruyants.

1



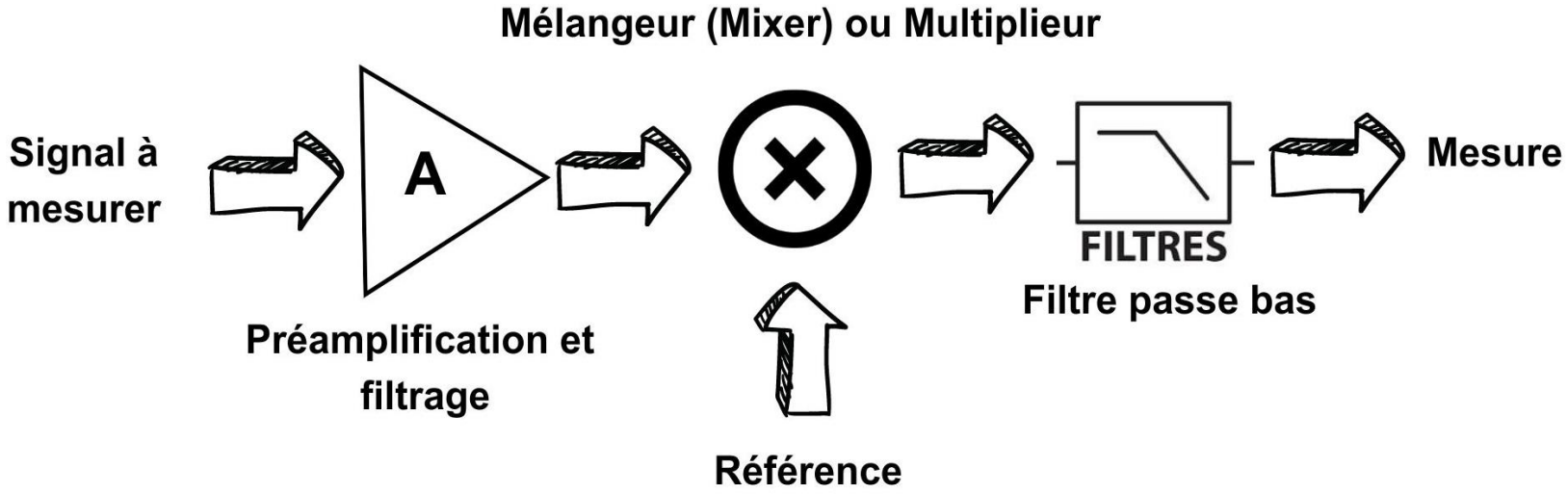
Lock-In Amplifier 4 MHz - SR 865A

SRS Stanford Research Systems

La détection synchrone peut porter le nom de détecteur de quadrature de phase, démodulateur I/Q ou détecteur cohérent. ***La détection synchrone est une méthode de mesure très classique en physique.***

Principe

La détection synchrone ou Lock-in Amplifier est une technique de traitement du signal hétérodyne permettant d'extraire des signaux de faible amplitude dans un bande étroite, noyés dans du bruit important, à large bande. Il s'agit essentiellement d'un **multiplicateur analogique** ou **numérique** suivi d'un **filtre passe-bas**.



Pratiquement, un amplificateur lock-in convertit le signal haute fréquence en une *composante continue ou à très basse fréquence.* On *multiplie le signal à mesurer par un signal sinusoïdal de référence de fréquence proche* de celle de la fréquence moyenne à détecter. En général une amplification est nécessaire après le filtre passe-bas.

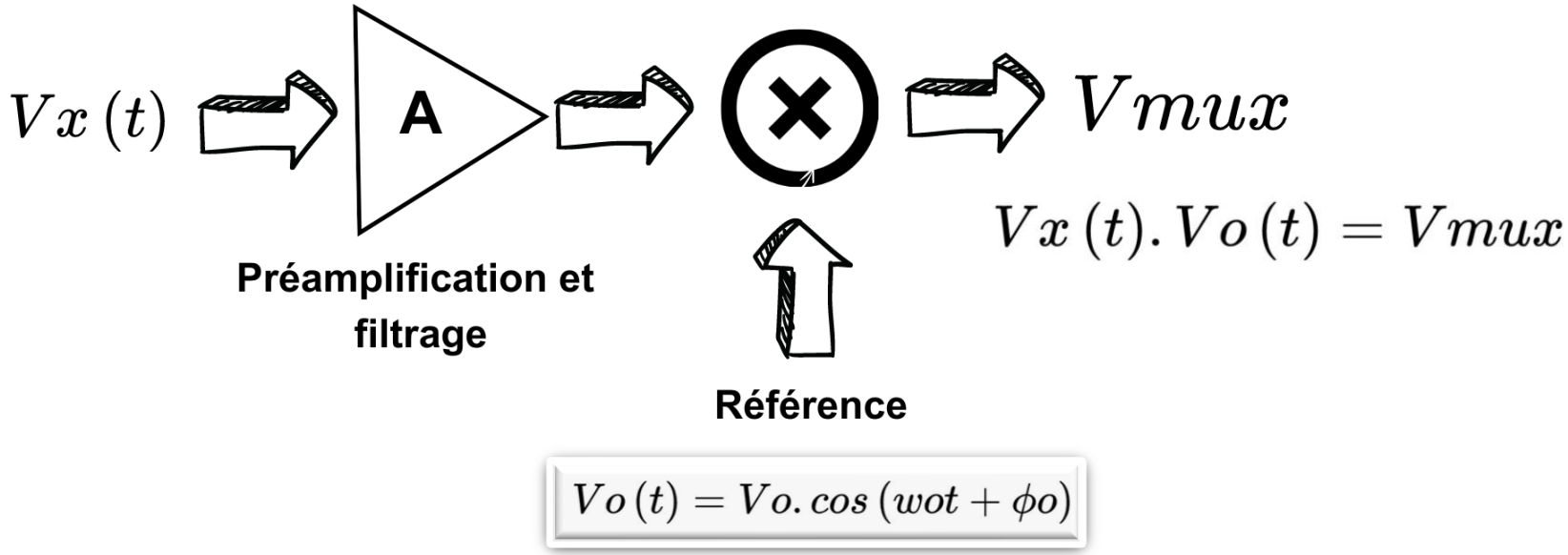
Structure

Le mélangeur

$$Vx\,(t).Vo\,(t) = Vx.Vo \div 2\,(cos\,(2\pi\,(\underbrace{fx + fo}_{}))t + (\phi x + \phi o)) + cos\,(2\pi\,(\underbrace{fx - fo}_{}))t + (\phi x - \phi o)))$$

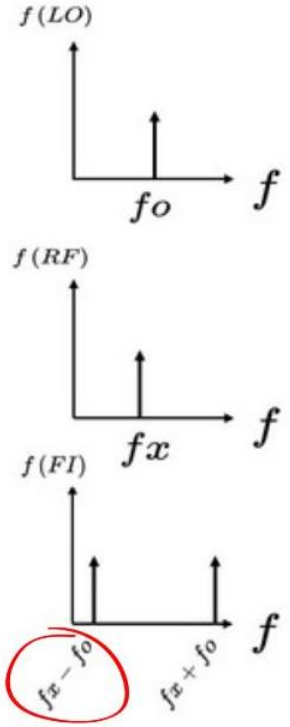
$$Vx\,(t) = Vx.cos\,(wxt + \phi x)$$

Mélangeur (Mixer) ou Multiplieur



$$Vx\,(t).Vo\,(t) = Vmux$$

$$Vo\,(t) = Vo.cos\,(wot + \phi o)$$



Sortie Mélangeur

La sortie du mélangeur contient à la fois les ***fréquences somme et différence des deux signaux d'entrée***.

Structure

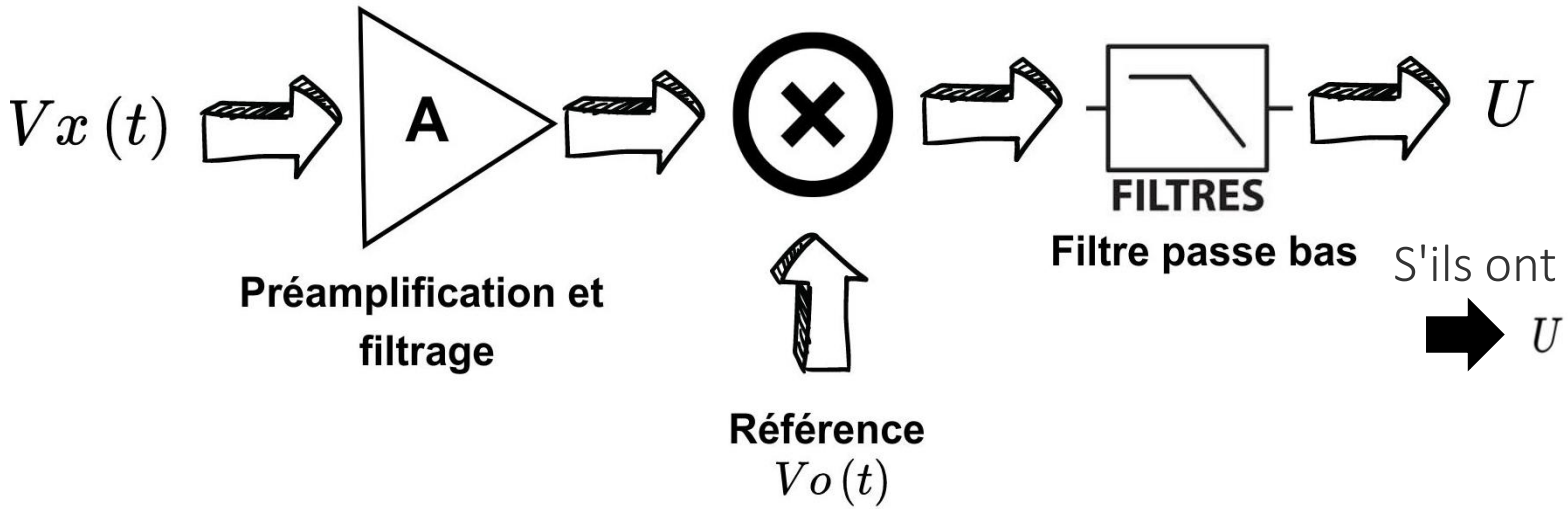
Le mélangeur + filtre passe bas (down conversion)

Après filtrage passe-bas et si $wx = wo$:

$$Vx\,(t) = Vx.\,cos\,(wxt + \phi x)$$

Si Vx (t) et Vo (t) ont la même phase

➡ $U = kVxVo \div 2$



S'ils ont des phases différentes

➡ $U = kVxVo \div 2.cos\,(\phi x - \Phi o)$

$$Vo\,(t) = Vo.\,cos\,(wot + \phi o)$$

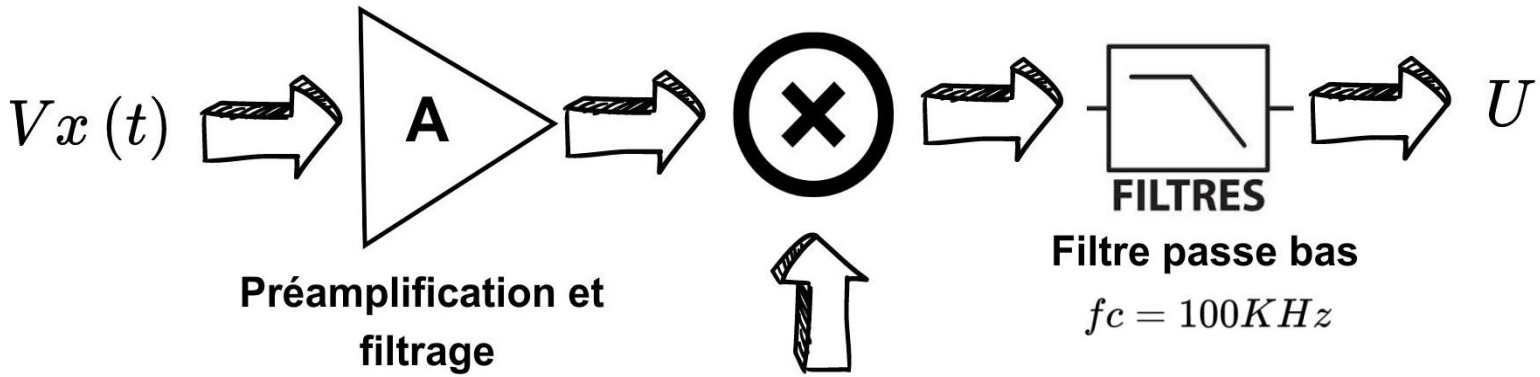
Structure

Mesure avec un générateur 2 voies synchrones et wx = wo

$fx = 10MHz - 3dBm$ et $fo = 10MHz - 7dBm$

$$Vx(t) = Vx.cos(wxt + \phi x)$$

Mélangeur (Mixer) ou Multiplieur



Préamplification et filtrage

Filtre passe bas

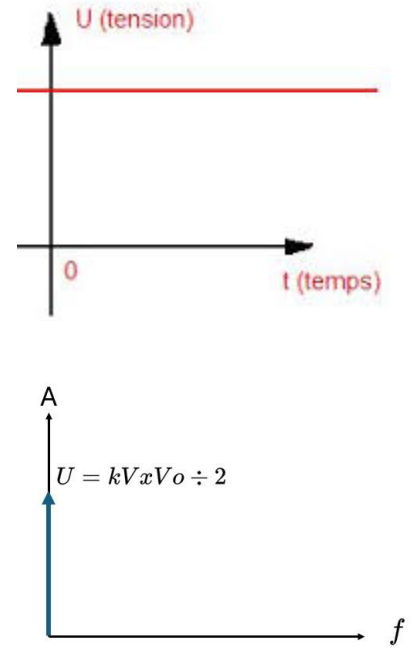
$fc = 100KHz$

Référence

$Vo(t)$

Avec un générateur 2 voies synchrones pour Vx(t) et Vo(t)

$$Vo(t) = Vo.cos(wot + \phi o)$$



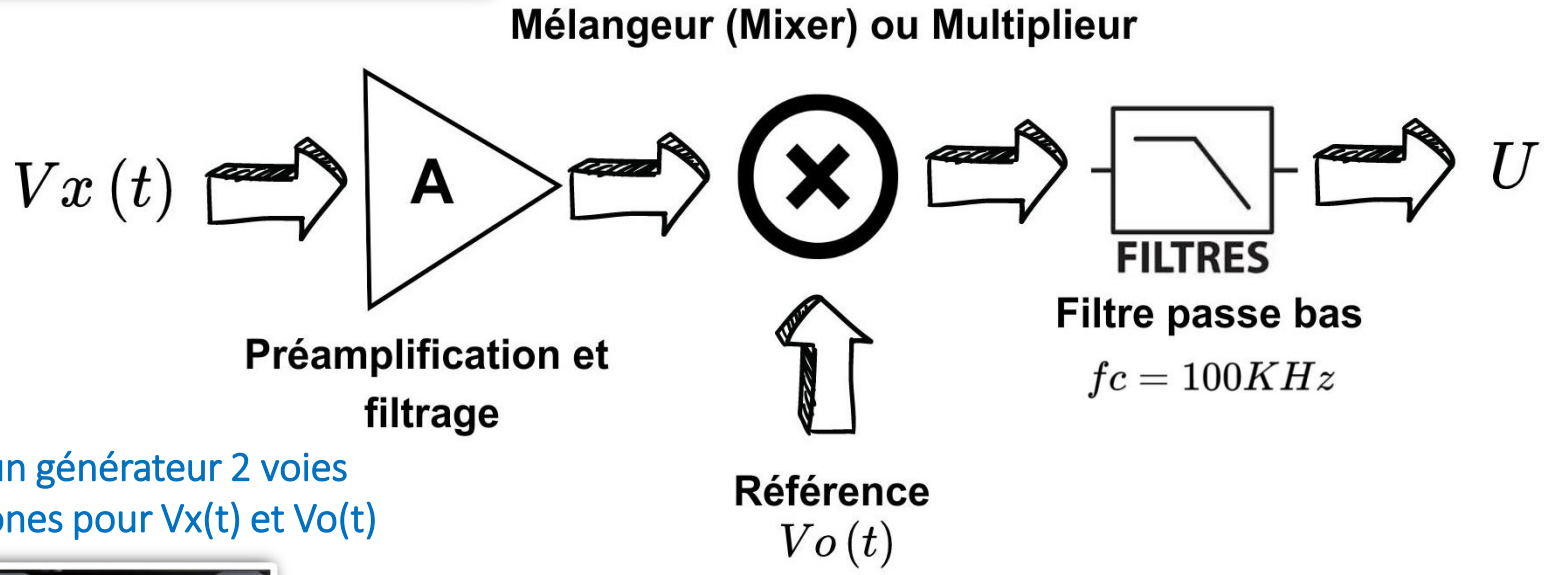
$$U \neq 0 \rightarrow si \rightarrow \phi x - \phi o \neq \pi \div 2$$

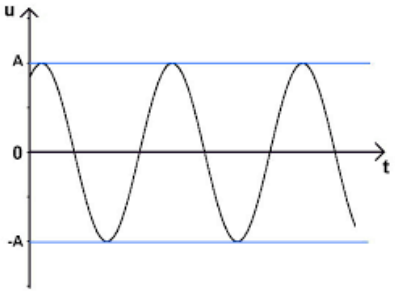
Structure

Mesure avec un générateur 2 voies synchrones et wx ≠ wo

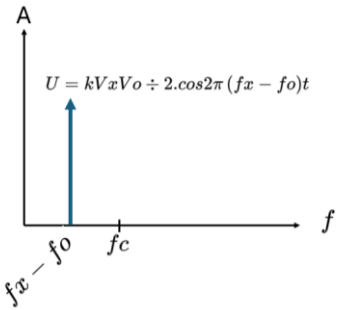
$fx = 10.02MHz - 3dBm$ et $fo = 10MHz - 7dBm$

$$Vx(t) = Vx.cos(wxt + \phi x)$$





$fx - fo = 20KHz$



Avec un générateur 2 voies synchrones pour Vx(t) et Vo(t)

$$Vo(t) = Vo.cos(wot + \phi o)$$

Application

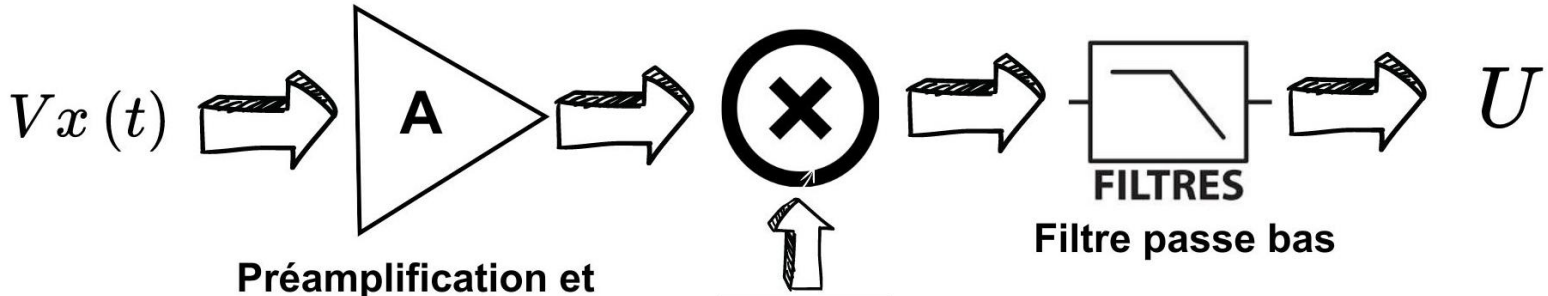
démodulation d'amplitude AM

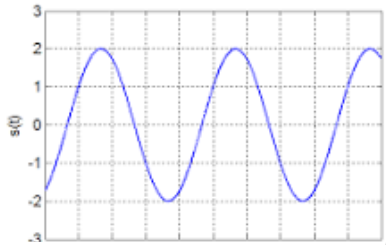
$$Vx(t) = Vo(1 + m.cos\Omega t).cos(wot + \phi x)$$

Signal modulé avec porteuse @10MHz

Mélangeur (Mixer) ou Multiplieur

Signal modulant





$fc = 100KHz$

+ amplification (G=+10)

$fo = 10MHz - 7dBm$

$Vx(t) = Vo(1 + m.cos\Omega t).cos(wot + \phi x)$

Modulation AM 10KHz

A = +/-20mV

$U = mkVo^2 \div 2.cos(\Omega t)$

Après filtrage et suppression du DC

$fm = 10KHz$

Porteuse générée à partir du signal modulé

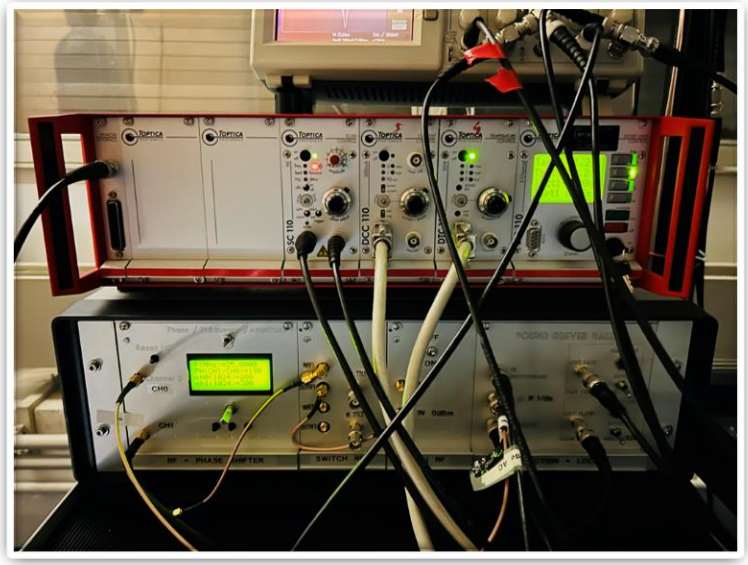
$$Vo(t) = Vo.cos(wot + \phi o)$$

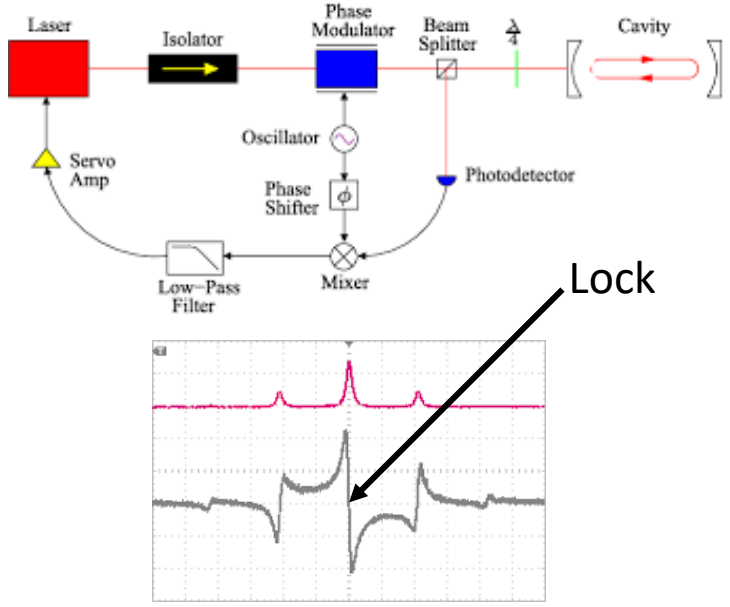
Dans une démodulation d'amplitude AM, on multiplie le signal AM par un signal sinusoïdal *cohérent en phase avec la porteuse.*
Générateur deux voies synchrones. *Très bon rapport signal/bruit.*

Exemple d'utilisation au LPL

Détection synchrone – démodulation synchrone

Lock-in Amplifier
DDS & phase shifter
Développement LPL
Lock-in Amplifier



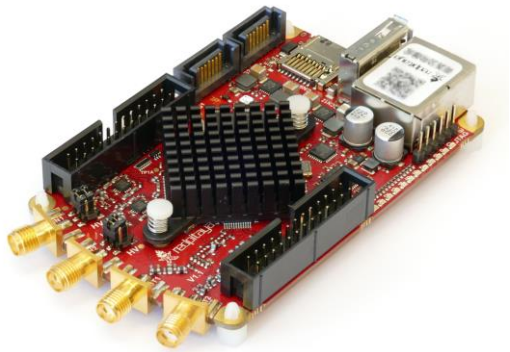


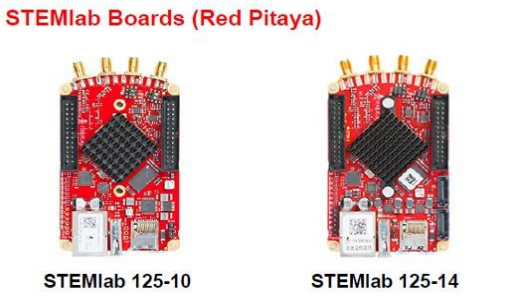
La technique Pound-Drever-Hall (**PDH**) est une approche largement utilisée pour stabiliser la fréquence de la lumière émise par un laser au moyen d'un verrouillage dans une cavité stable et ce grâce **à une démodulation synchrone.**

Le modulateur de phase électro-optique est piloté par un signal sinusoïdal et imprime des bandes latérales sur le laser. On démodule le signal issu du photodétecteur grâce à la détection synchrone, le signal d'erreur Issu de la démodulation corrige le courant du laser au travers d'un correcteur PI ou PID.

Détection synchrone numérique

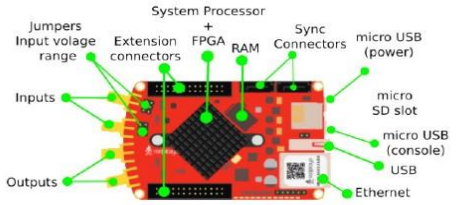
Sur carte Red Pitaya 125-14
Option horloge externe





STEMlab 125-10 vs. STEMlab 125-14 (originally Red Pitaya V1.1)

STEMlab is available in two versions and both offer the same functions and features with the difference in technical specification of high-frequency inputs and outputs, RAM capacity some other differences.
They are addressed to target different groups and/or needs. Where STEMlab 125-14 has 14 bit input/output channels for highly accurate measurement results in professional environment, STEMlab 125-10 has 10 bit input/output channels and is perfect for universities, students and makers.



Caractéristiques

		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

← LTC2145-14

← DAC1401D125

Lien pour la présentation de la carte Red Pitaya avec des exemples de mise en œuvre sur VIVADO 2022.2 et VITIS 2022.2
https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

Détection synchrone numérique

Caractéristiques

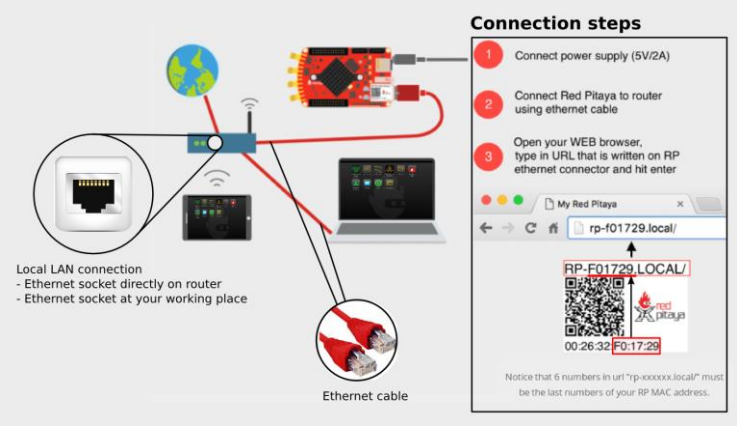
- ✓ Plate-forme d'acquisition et de génération de signaux RF de la taille d'une carte de crédit
- ✓ Connectivité Ethernet
- ✓ SoC Xilinx (processeur et FPGA)
- ✓ Deux entrées analogiques rapides et deux sorties
- ✓ Possibilité d'intégration dans son propre système/produit
- ✓ Code source du logiciel ouvert disponible, fonctionne avec Linux ou PC Windows
- ✓ Peut être utilisé comme oscilloscope et générateur de signaux, spectre, analyseur de Bode, analyseur logique, compteur LCR*, streaming, SDR ou analyseur de réseau vectoriel* →applications disponibles
- ✓ Peut être contrôlé à distance à l'aide de LabVIEW, MATLAB, Python ou Scilab
- ✓ Peut-être reprogrammé pour répondre à des besoins personnalisés -> JTAG
- ✓ Pris en charge par un marché d'applications avec plusieurs applications gratuites disponibles

Détection synchrone numérique

Caractéristiques

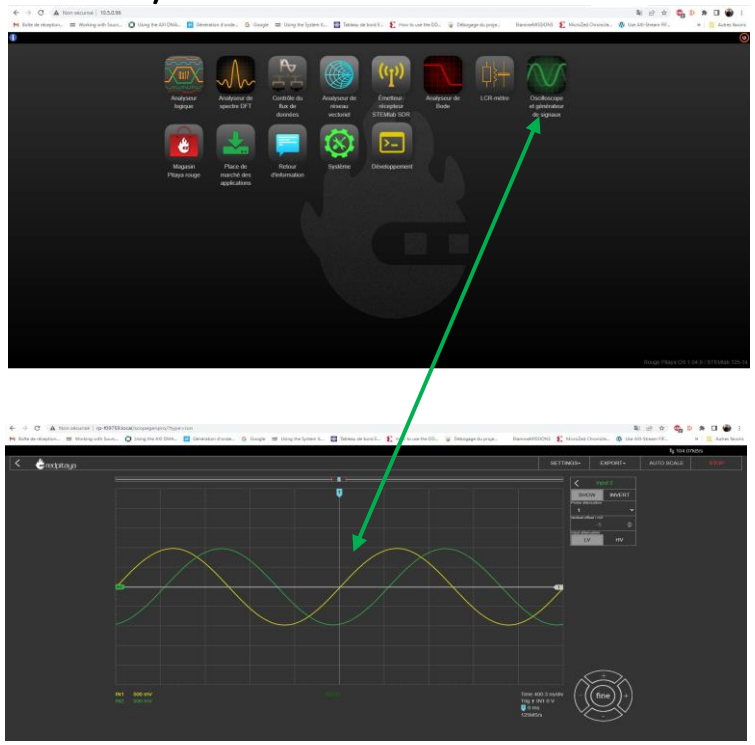
Écosystème et applications du Red Pitaya

Accès mode local ou adresse IP



Accès local RP-F****LOCAL ou accès distant adresse IP

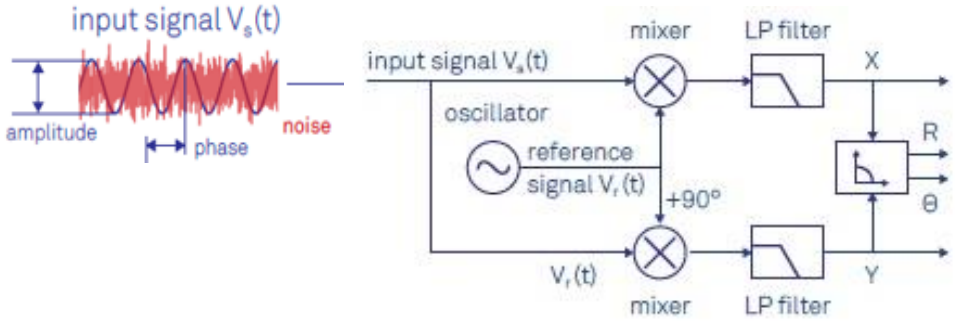
Ecosystème installé sur la carte SD



https://github.com/RedPitaya/RedPitaya

Réalisation de la détection synchrone avec VIVADO sur carte Red Pitaya 125-14

Si l'on souhaite effectuer une démodulation qui ne dépende pas de la phase existante ϕ entre la fréquence de modulation et de démodulation, il est possible d'utiliser ce montage :



R et Θ sont respectivement, l'amplitude et la phase du signal de sortie

12

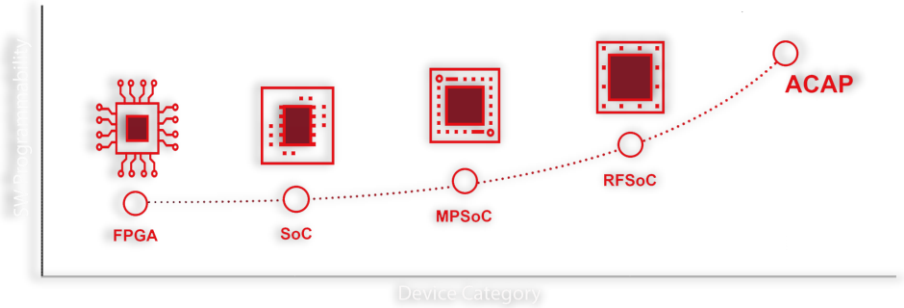
R et Θ définis par les équations suivantes : $R = X2 + Y2$; $\theta = atan2(Y, X)$ signal d'entrée et la référence (en phase et déphasé) sont mélangés puis filtrés pour obtenir les informations en amplitude et en phase.

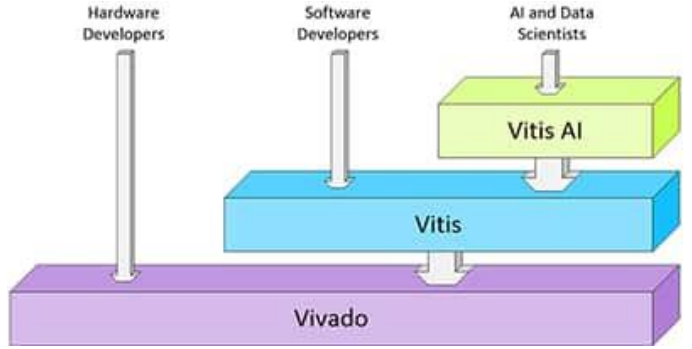
Pour réaliser la détection synchrone conformément au montage ci-dessus on va programmer le FPGA de la carte Red Pitaya en **Bare Metal** (sans la carte SD insérée) et réaliser différentes fonctions :

- Génération d'un signal de référence et son déphasage
- Multiplication de deux signaux
- Filtrage

Présentation de VIVADO et de VITIS

Vivado Design Suite de Xilinx a fait ses débuts en 2012 en tant qu'environnement de conception intégré pour le développement embarqué de FPGA. La plateforme Vitis plus récente inclut également Vivado, mais permet également à un plus large éventail de nouveaux développeurs de concevoir du matériel.





ISE XPS VITIS VIVADO SDK

VIVADO

➔ **langages de description matériel** : Verilog et VHDL

XILINX VITIS

➔ **langages C et C++**

https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html
https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vitis.html

Détection synchrone numérique

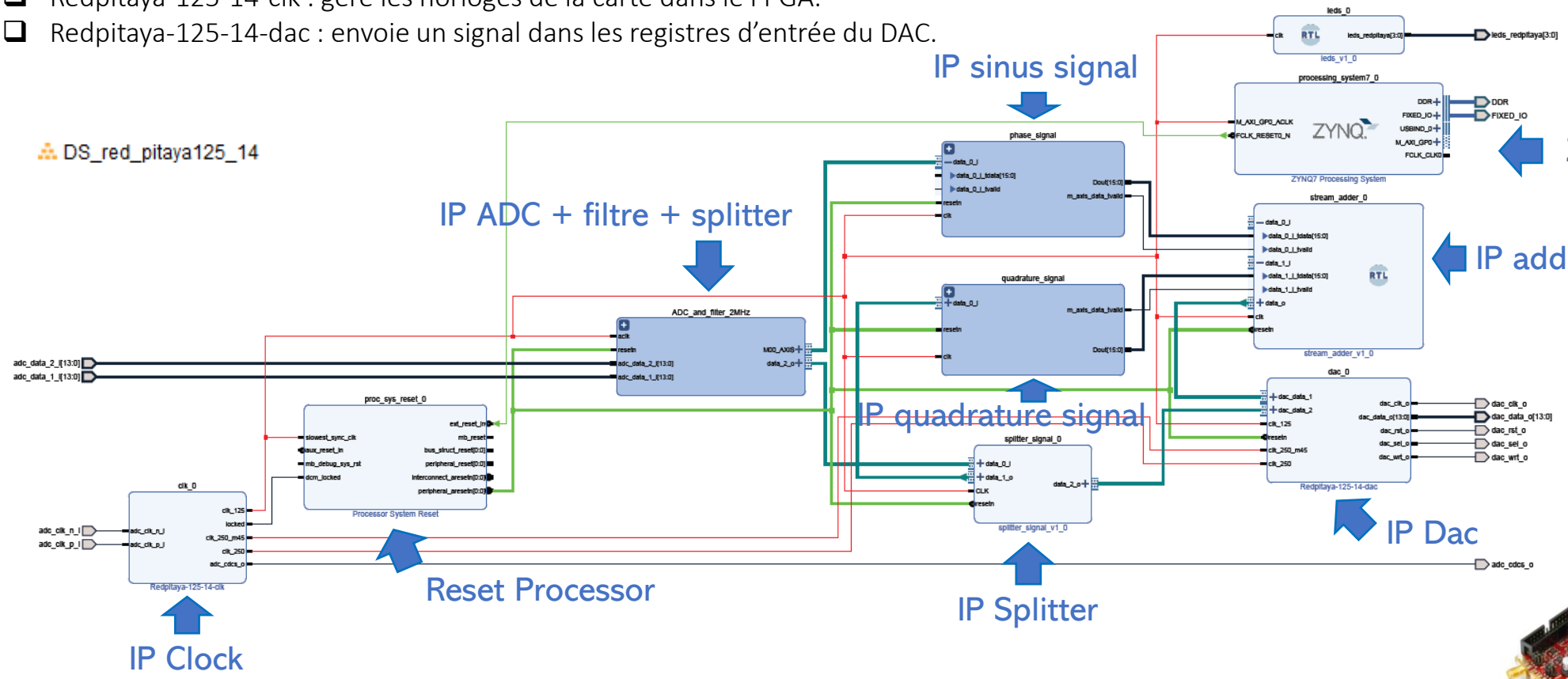
Sur carte Red Pitaya 125-14

14

Projet Vivado 2022.2 sur la Red Pitaya:

Le “Block Design” ci-dessous est composé de blocs réalisant des fonctions particulières.

- ❑ Redpitaya-125-14-adc -> récupère les signaux en sortie de l’ADC dans le FPGA.
- ❑ Redpitaya-125-14-clk : gère les horloges de la carte dans le FPGA.
- ❑ Redpitaya-125-14-dac : envoie un signal dans les registres d’entrée du DAC.



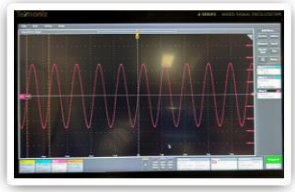
Des blocs fonctionnels de bases pour la Red Pitaya sont disponibles sur un dépôt GitHub.

Détection synchrone numérique

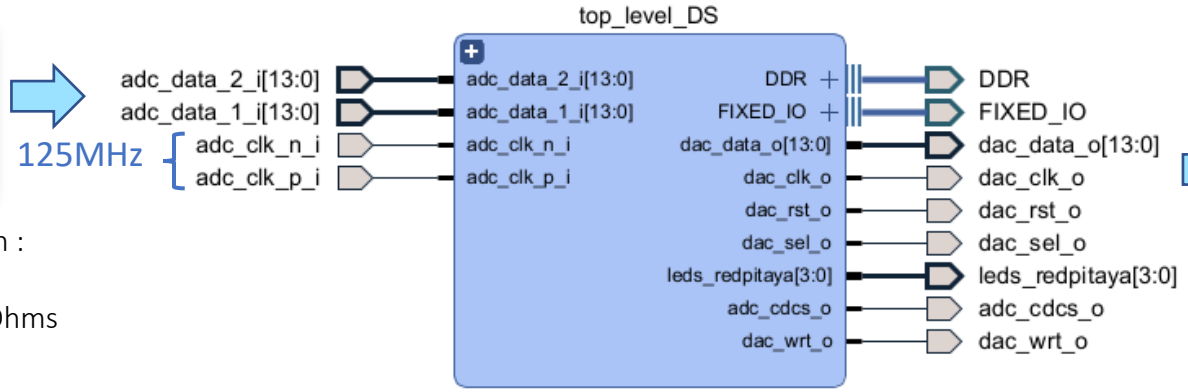
Sur carte Red Pitaya 125-14

Projet Vivado sur la Red Pitaya:

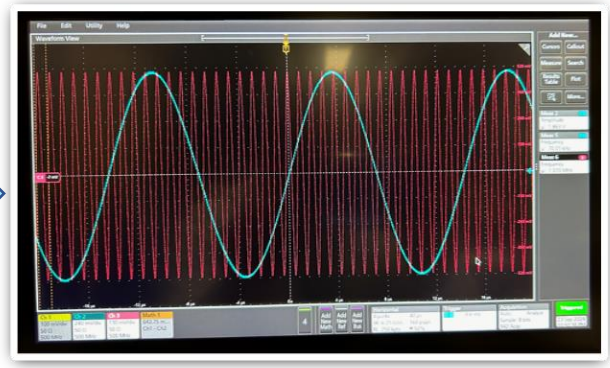
Signaux entrées-sorties



Générateur de fonction :
adc_data_1_i[13:0]
1.07MHz 1Vpp sur 50 Ohms



En bleu démodulation = 70KHz
Dac_data_1_i[13:0]
1.07MHz 1Vpp sur 50 Ohms



En rouge recopie adc_data_1_i[13:0]
1.07MHz 1Vpp sur 50 Ohms

DAC entrelacé

Signal de référence en phase et quadrature généré en interne du FPGA =1MHz

Détection synchrone numérique

Sur carte Red Pitaya 125-14

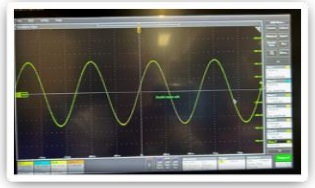
IP ADC

AXI4-Stream Broadcaster:

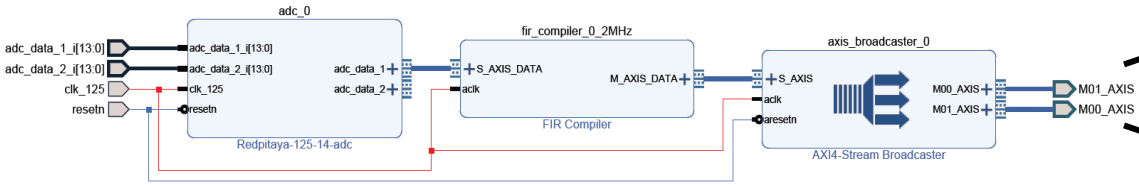
Cette IP est particulièrement pratique pour déboguer.

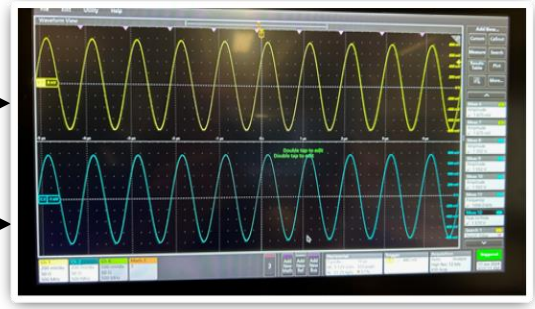
Elle permet de passer d'un seul bus de données d'un signal à plusieurs bus de données de ce même signal.

ADC_and_filter_2MHz



Générateur de fonction :
adc_data_1_i[13:0]
1MHz 1Vpp sur 50 Ohms





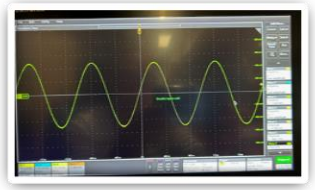
M00_AXIS = = 1MHz 1,6Vpp sur 50 Ohms
M01_AXIS = = 1MHz 1,6Vpp sur 50 Ohms

Détection synchrone numérique

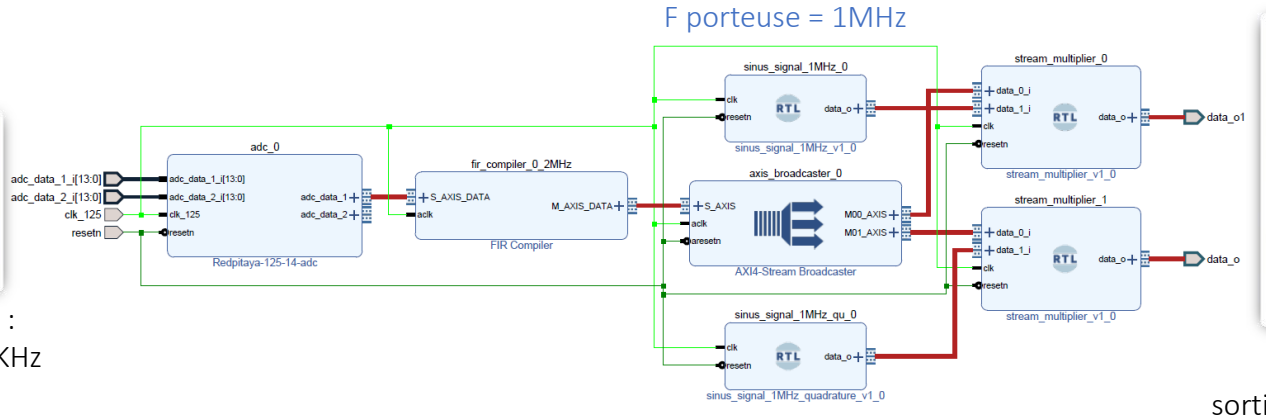
Sur carte Red Pitaya 125-14

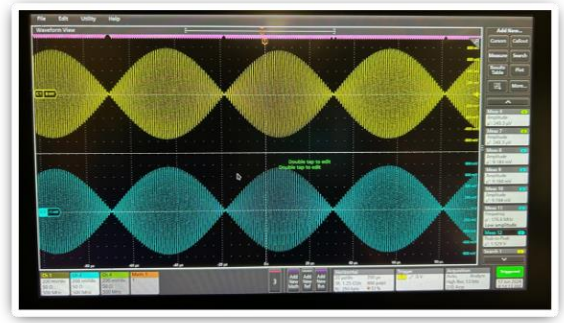
IP sinus signal & quadrature signal

L'IP Stream multiplier: Cette IP réalise le produit de deux nombres signés.



Générateur de fonction :
adc_data_1_i[13:0] = 10KHz





Modulation d'amplitude AM
sortie Multiplieur avec F(in) adc_data_1_i =10KHz
Porteuse @1MHz

Le taux de modulation, noté m est caractéristique du modulateur. Il représente l'amplitude du signal modulé par rapport à l'amplitude de la ***porteuse ici m =1***, modulation AM à porteuse supprimée (pas d'offset DC).

Détection synchrone numérique

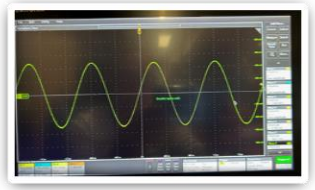
Sur carte Red Pitaya 125-14

Création d'une IP

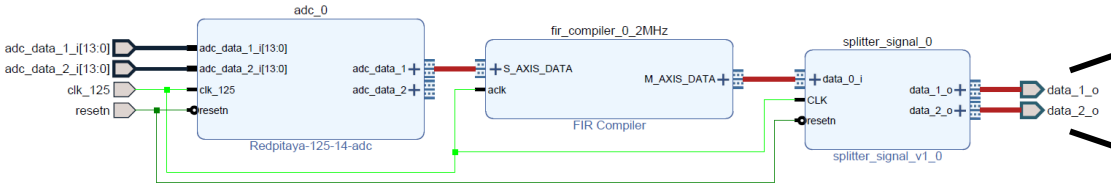
Création d'une IP pour Vivado à l'aide des interfaces streaming AXI :

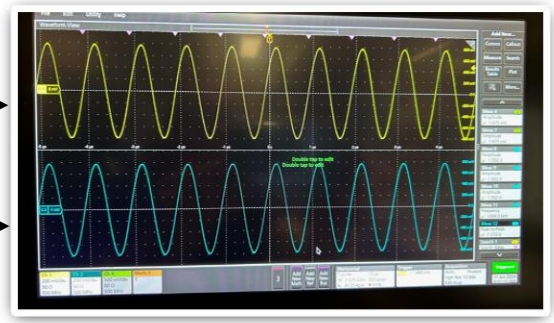
On peut également créer une IP (Intellectual Property) dans un répertoire dédié pour ses IP et intégrer les fichiers HDL par exemple remplacer l'IP ***Xilinx axis_broadcaster*** par une IP maison nommée ***splitter_signal .***

ADC_and_filter_2MHz



Générateur de fonction :
adc_data_1_i[13:0]
1MHz 970mVpp sur 50 Ohms





M00_AXIS = = 1MHz 1,6Vpp sur 50 Ohms
M01_AXIS = = 1MHz 1,6Vpp sur 50 Ohms

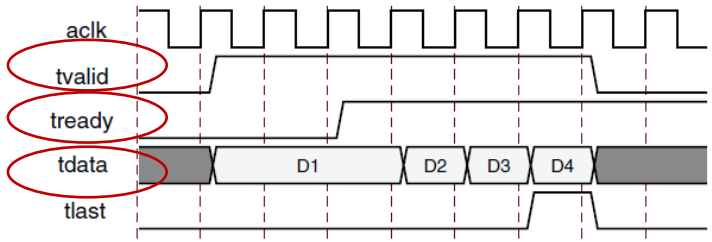
Création de l'IP splitter_signal

Détection synchrone numérique

Sur carte Red Pitaya 125-14

Création d'une IP pour Vivado à l'aide des interfaces de streaming AXI :
On créer un fichier VHDL pour le module splitter_signal en précisant les ports d'entrées-sorties en adéquation avec le standard AXI4-Stream interfaces.

Figure: Transfert de données dans un canal AXI4-Stream




```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity splitter_signal is
    Generic (data_width : integer := 16);
    Port
         ( CLK : in STD_LOGIC;
           resetn : in STD_LOGIC;
           data_0_i_tdata : in STD_LOGIC_VECTOR (data_width - 1 downto 0);
           data_0_i_tvalid : in STD_LOGIC;
           data_1_o_tdata :  out STD_LOGIC_VECTOR(data_width - 1 downto 0);
           data_1_o_tvalid : out STD_LOGIC;
           data_2_o_tdata :  out STD_LOGIC_VECTOR(data_width - 1 downto 0);
           data_2_o_tvalid : out std_logic
         );

end splitter_signal;
```

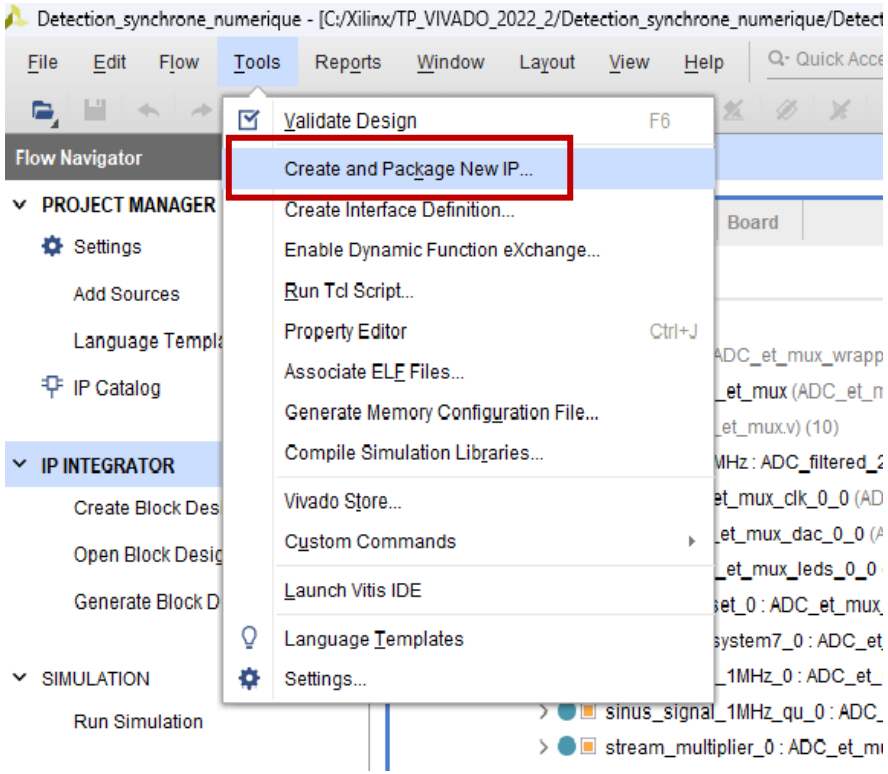
On créer le code VHDL splitter_signal en précisant les ports d'entrées-sorties et on créer un répertoire IP_directory en y insérant splitter_signal.vhd.

Détection synchrone numérique

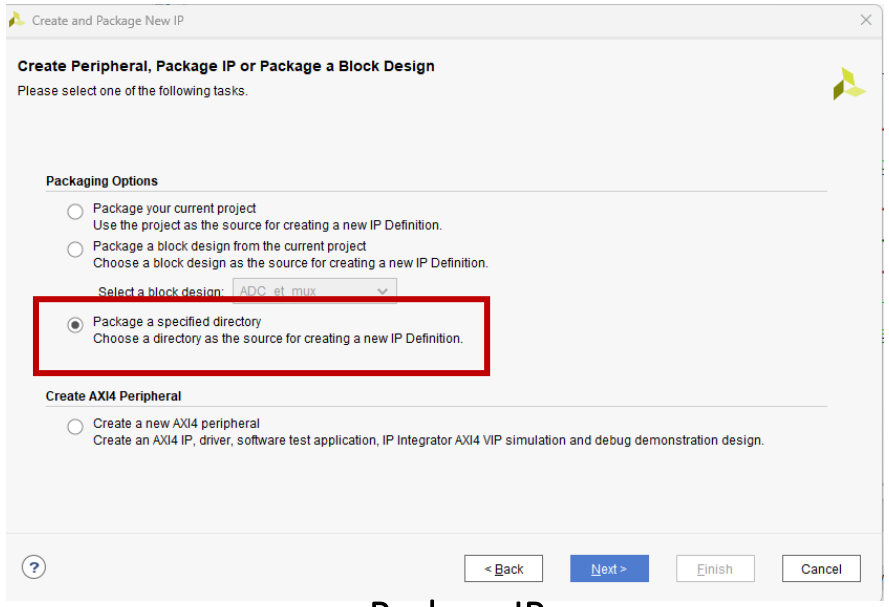
Sur carte Red Pitaya 125-14

Création d'une IP pour Vivado à l'aide des interfaces de streaming AXI : **Create IP**

Tools-> Create and Package New IP -> Next ->



Package a specified directory -> Next



Package IP

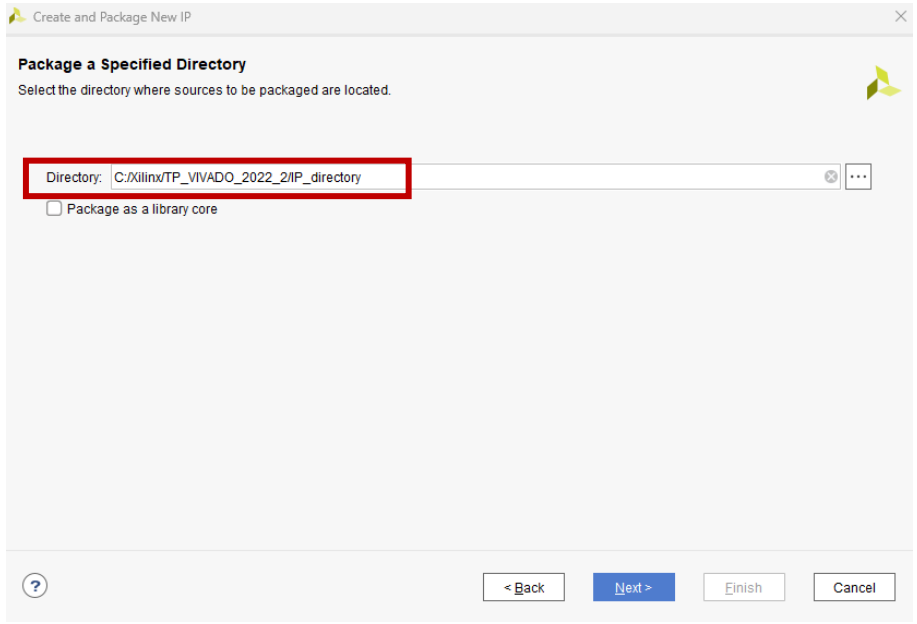
Détection synchrone numérique

Sur carte Red Pitaya 125-14

Création d'une IP pour Vivado à l'aide des interfaces de streaming AXI : **Directory**

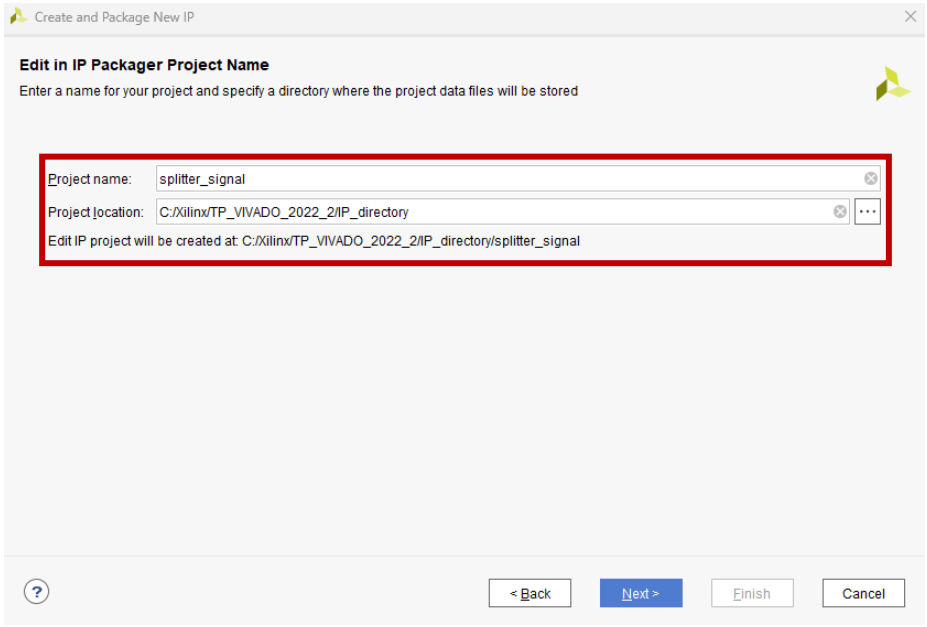
21

Next->



Package in a directory

Next -> and Finish



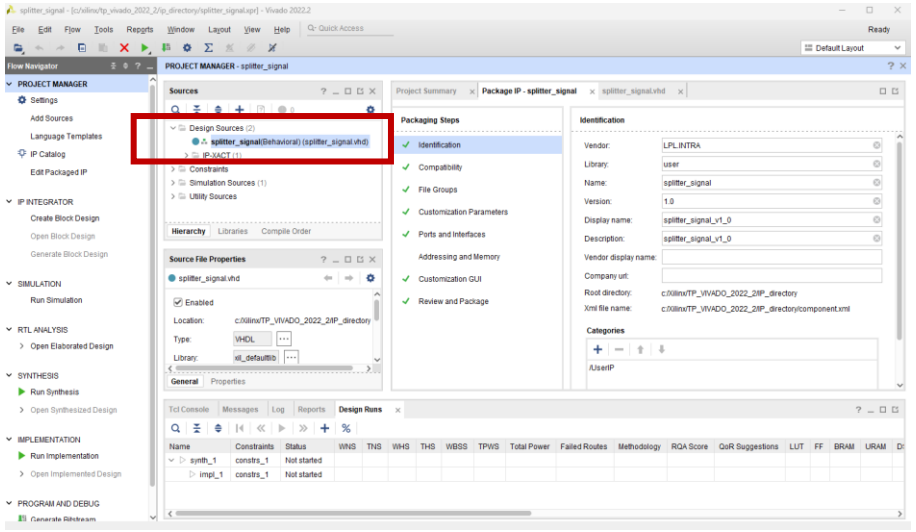
Name of IP

Détection synchrone numérique

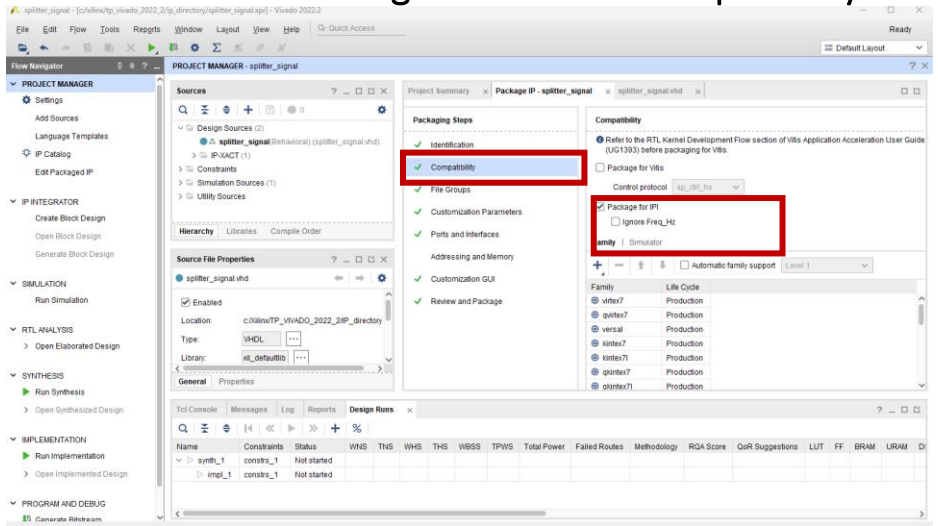
Sur carte Red Pitaya 125-14

Création d'une IP pour Vivado à l'aide des interfaces de streaming AXI : **Edition dans l'IP**

une nouvelle fenêtre Vivado s'ouvre.



Sélectionnez Package for IPI dans Compatibility

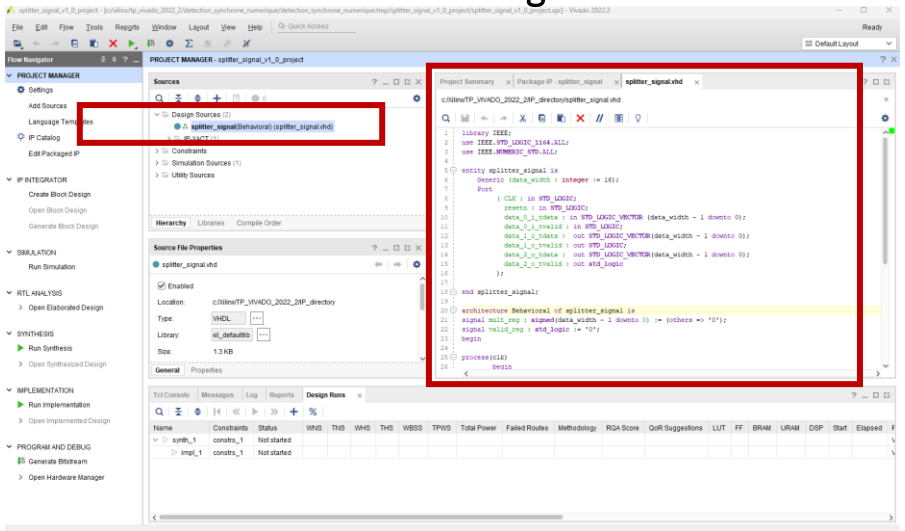


Détection synchrone numérique

Sur carte Red Pitaya 125-14

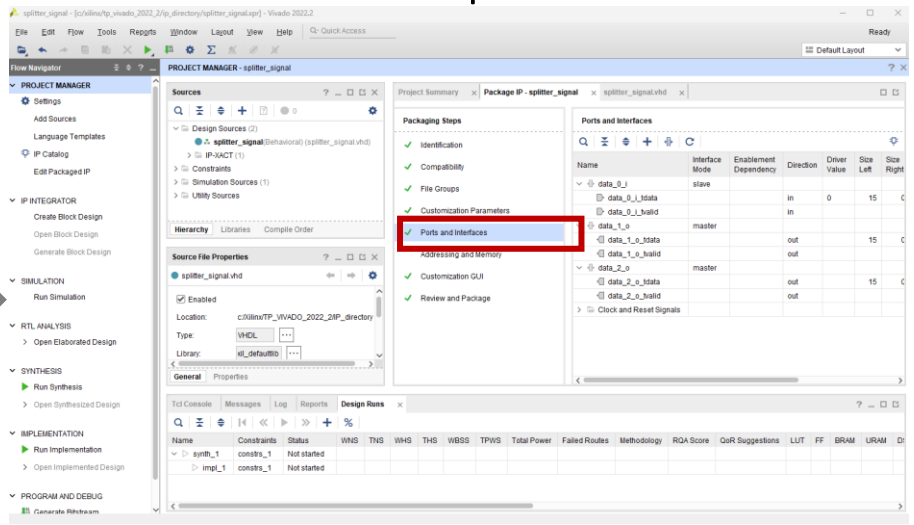
IP pour Vivado à l'aide des interfaces de streaming AXI : **Procédure dans Vivado**

Code VHDL dans Design Sources



Vérification du code VHDL

vérifications des ports et interfaces



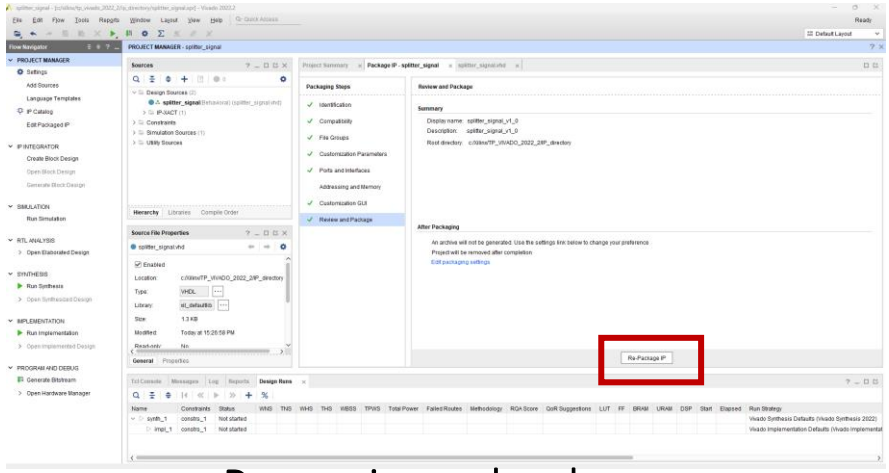
Ports and interfaces

Détection synchrone numérique

Sur carte Red Pitaya 125-14

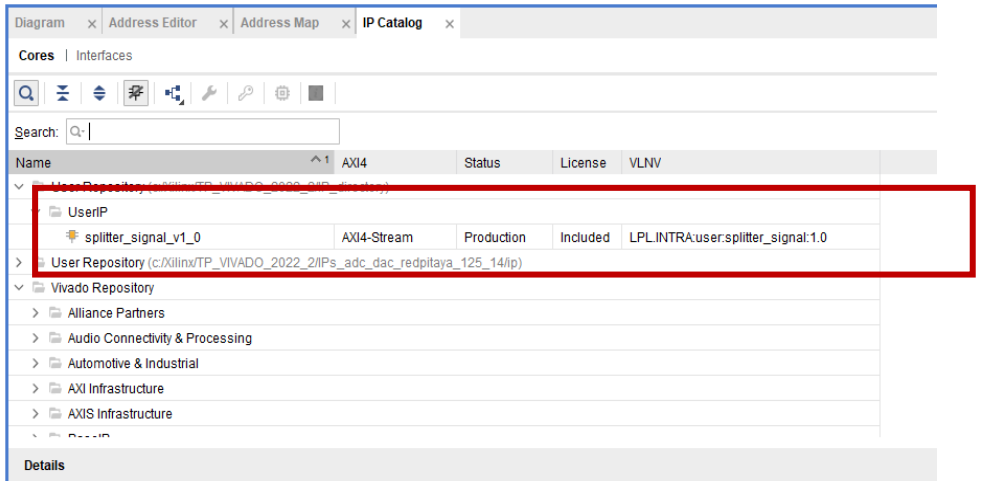
IP pour Vivado à l'aide des interfaces de streaming AXI :

Package IP or Re-Package IP



Dans review and package

IP Catalog -> UserIP



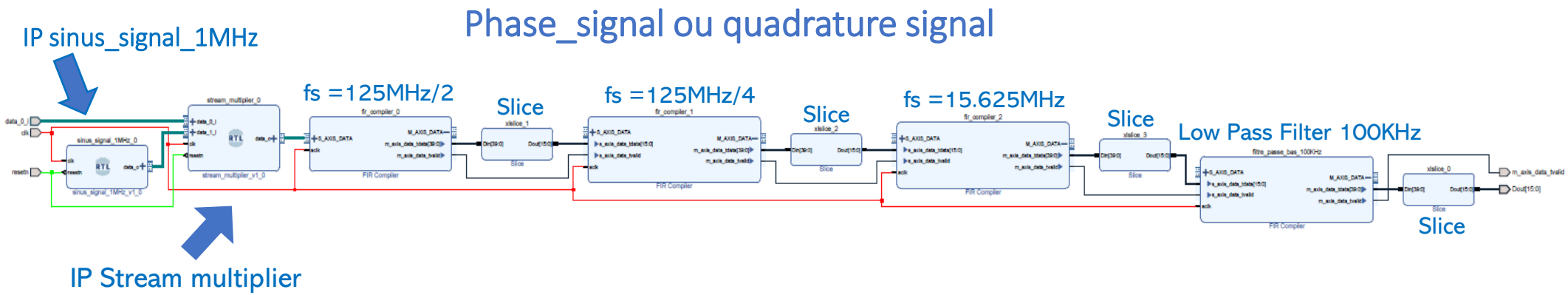
Détection synchrone numérique

Sur carte Red Pitaya 125-14

Génération du signal de référence en phase et quadrature avec filtrage & décimation:

Créer une IP dans laquelle le programme parcourt un tableau de coefficient correspondant à une sinusoïde à la fréquence souhaitée ou utiliser une DDS (IP synthétiseur numérique direct) pour générer le signal.

On va générer un signal de modulation à 1MHz en phase et quadrature.



L'IP Stream multiplier: Cette IP réalise le produit de deux nombres signés

L'IP sinus_signal_1MHz : Cette IP réalise le signal de modulation (référence) ici à 1MHz

L'IP Slice : Cette IP extrait des bits d'un signal de bus

L'IP Fir_compiler : permet de réaliser un filtre FIR passe-bas, passe-haut, passe-bande, une décimation, etc...

Détection synchrone numérique

Sur carte Red Pitaya 125-14

Génération du signal de référence en phase et quadrature :

```
i = 0:1:999;   % nombre  de points par période
Fs = 125000000;  % fréquence d'échantillonage
F = 1000000;       % fréquence du signal
Y = 32767*sin(2*pi*(F/FS)*i); % signal

% z ne  prend aucun chiffre apres la virgule.
Z = sprintf('%.0f,',1000*(sin(2*pi*(F/Fs)*i)))
```

```
i = 0:1:999;   % nombre  de points par période
Fs = 125000000;  % fréquence d'échantillonage
F = 1000000;       % fréquence du signal
Y = 32767*sin(2*pi*(F/FS)*i + pi/2);% signal

% z ne  prend aucun chiffre apres la virgule.
Z = sprintf('%.0f,',1000*(sin(2*pi*(F/Fs)*i)))
```

+pi/2

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.std_logic_unsigned.ALL;

entity sinus_signal_1MHz is
    Generic (data_width : integer := 16);
    Port ( clk : in STD_LOGIC;
           resetn : in STD_LOGIC;
           data_o_tdata :  out STD_LOGIC_VECTOR(data_width - 1 downto 0);
           data_o_tvalid : out STD_LOGIC);
end sinus_signal_1MHz;
```


L'IP sinus_signal_1MHz et L'IP sinus_signal_1MHz_quadrature : est calculé par exemple à partir de Matlab ou d'Excel avec 1250 points. On créer un fichier VHDL sinus_signal_1MHz et on le complète.

Détection synchrone numérique

Sur carte Red Pitaya 125-14

Filtre FIR Passe-Bas:

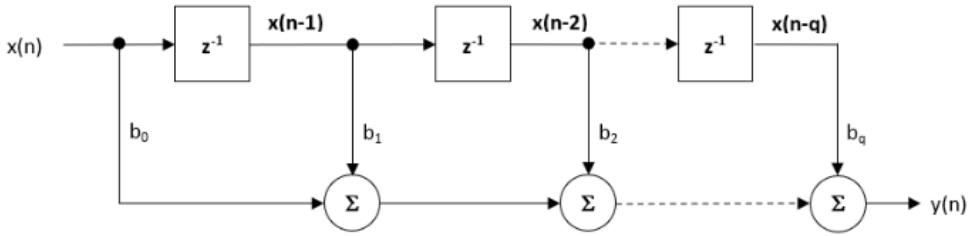
Il existe deux types de filtres numériques, les filtres à réponse impulsionnelle finie (RIF/FIR) et les filtres à réponse impulsionnelle infinie (RII/IIR). Les filtres FIR sont toujours stables, avec une phase linéaire, mais ont plus de coefficients qu'un filtre IIR qui n'a pas de phase linéaire. Les filtres IIR ont par ailleurs une réponse plus rapide et moins coûteuse en ressources, car ils ont moins de coefficients.

Rappelons l'équation d'un filtre FIR :

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

N = nombre de coefficients (ordre du filtre)
hk= Coefficients de la fonction de transfert du filtre

Implémentation :



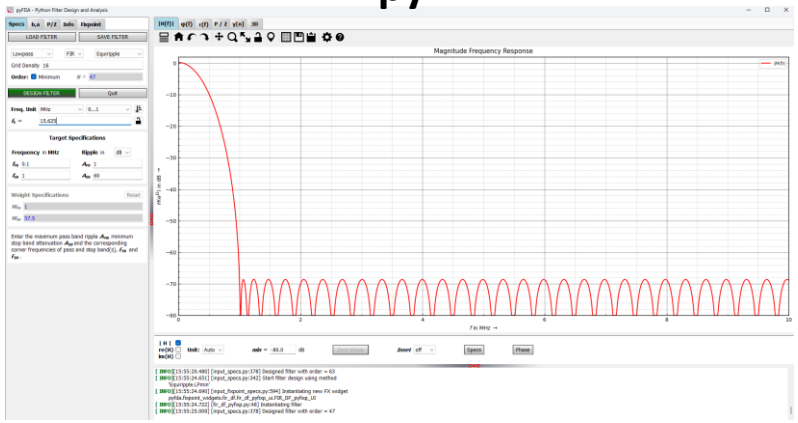
Détection synchrone numérique

Sur carte Red Pitaya 125-14

Filtre FIR Passe-Bas:

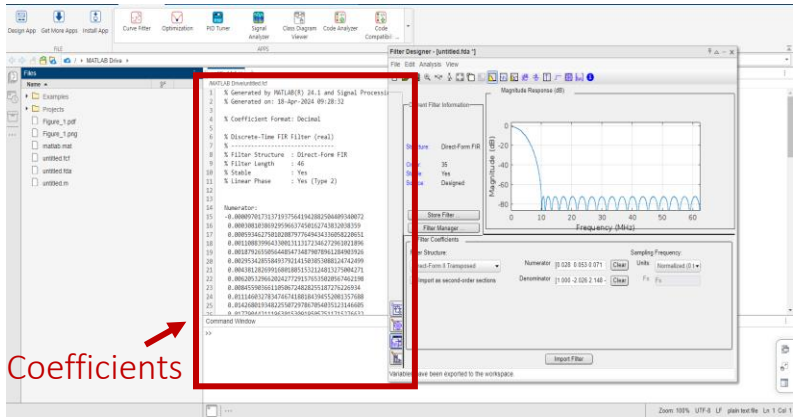
On va utilisé une ***IP Xilinx (FIR Compiler)*** permettant de réaliser un filtre FIR passe-bas. Les coefficients du filtre peuvent être trouvés à l'aide de différents logiciels (MATLAB, GNU Octave, Scilab, etc.). Dans notre cas, pour calculer les coefficients du filtre, nous utilisons **pyFDA open source** et son application **(Python Filter Design and Analysis).**

pyFDA



Application pyFDA (Python Filter Design and Analysis).

MATLAB



Application Filter Designer de Matlab

IP fir_compiler : Cette IP réalisera le filtre Passe-bas avec un nombre de coefficients réduit grâce aux filtres demi-bandes en amont de l'IP FIR.

Détection synchrone numérique

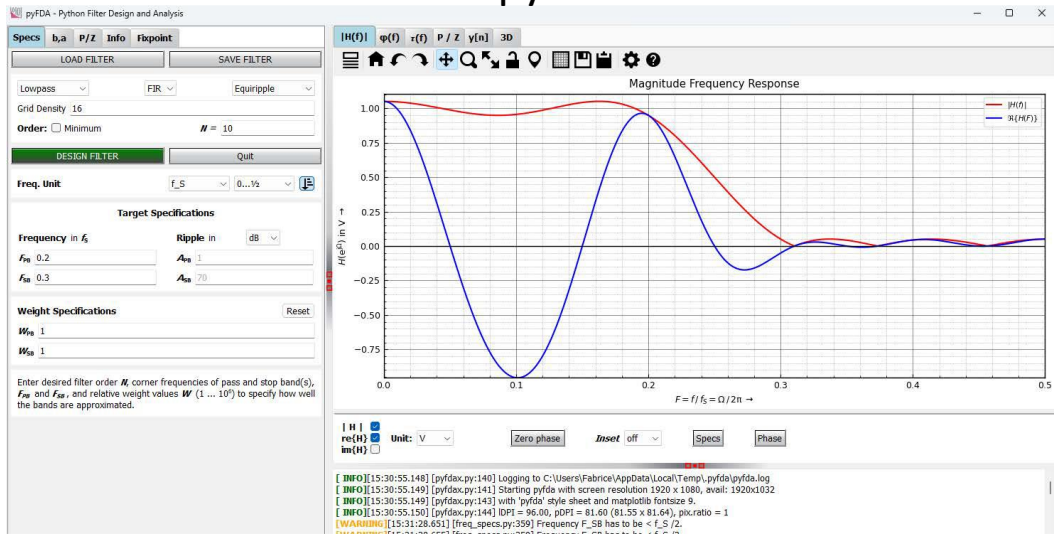
Sur carte Red Pitaya 125-14

Filtre demi-bande pour la décimation: "sous-échantillonner"

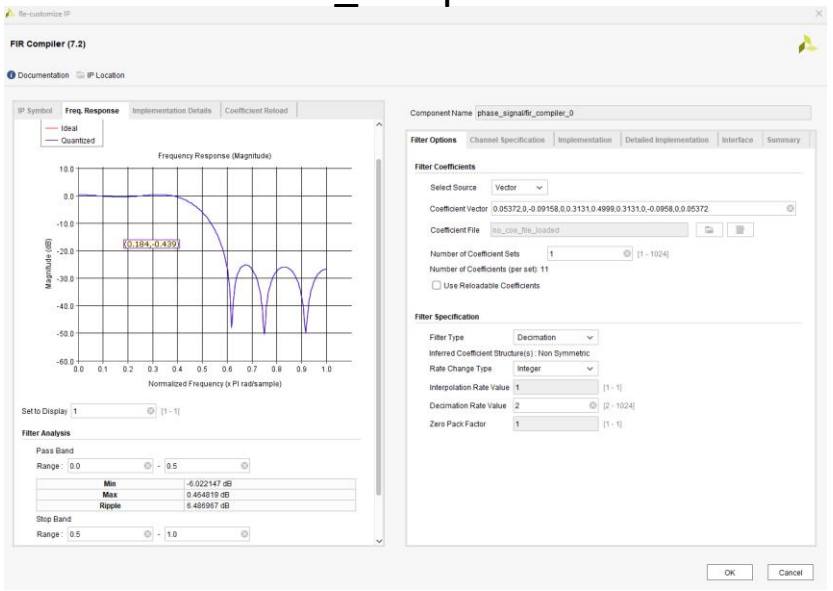
On utilise ***fir_compiler sous Vivado*** pour paramétrer un filtre demi-bande. On réduit la cadence/2 à chaque fois que l'on intercale un filtre demi-bande. Les coefficients sont à calculer avec Matlab ou différent logiciel tel que **PyFDA** comme ci-dessous.

Important : plus il y a de coefficient et plus la bande passante est réduite.

pyFDA



IP fir_compiler



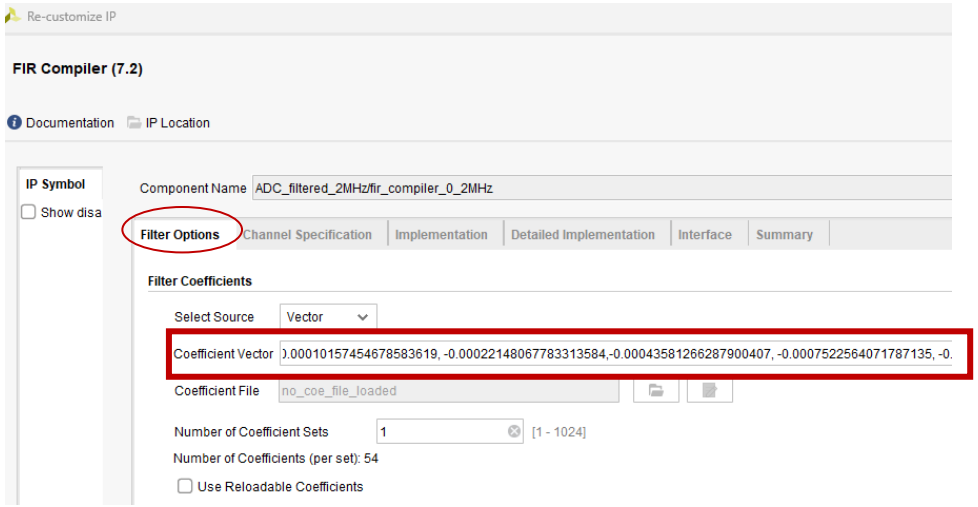
Cette IP permet de réduire la cadence de la fréquence d'échantillonnage, celle-ci est à 125MHz. C'est nécessaire pour limiter le nombre des coefficients dans le calcul des filtres FIR.

Détection synchrone numérique

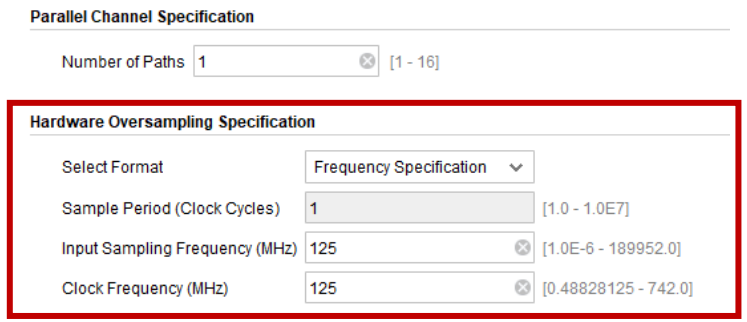
Sur carte Red Pitaya 125-14

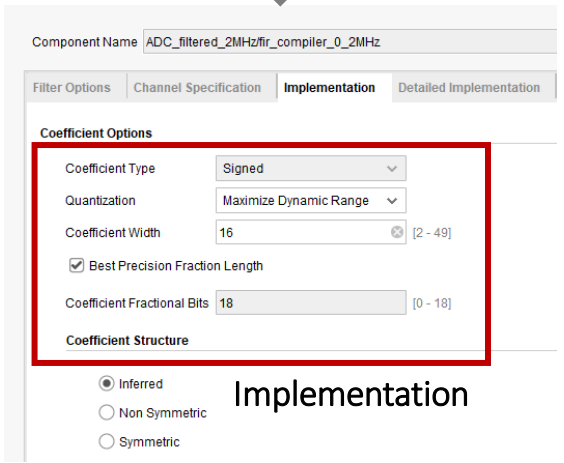
Exemple du Filtre FIR Passe-Bas 2MHz:

Rentrer les coefficients du filtre obtenu par pyFDA ou Matlab dans l'onglet coefficient vector de fir_compiler en séparant les uns des autres par des virgules.



Channel Specification





Implementation

Procédure :

1. Rentrer les coefficients du filtre obtenu par pyFDA ou Matlab
2. définir la fréquence d'échantillonnage à l'entrée du filtre FIR
3. définir le format des données et la taille du bus IN/OUT, ici 16 bits

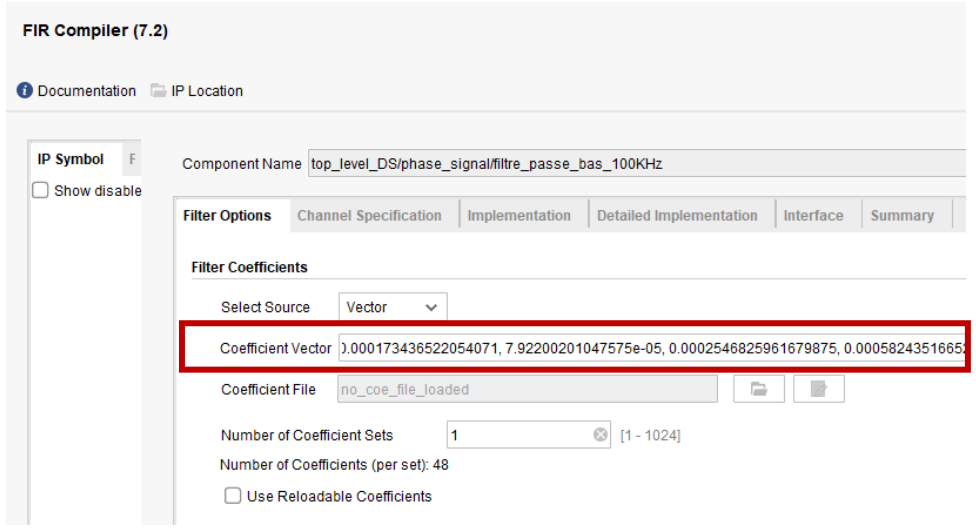
Maximize Dynamic Range, les coefficients vont être retravaillés pour s'intégrer dans la taille de bus que vous aurez déterminé.

Détection synchrone numérique

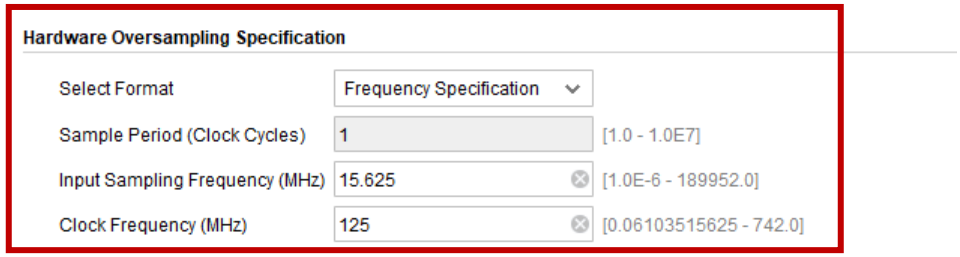
Sur carte Red Pitaya 125-14

Exemple du Filtre FIR Passe-Bas 100KHz:

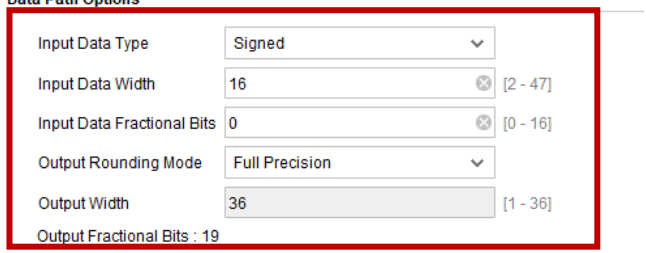
Rentrer les coefficients du filtre obtenu par pyFDA ou Matlab dans l'onglet coefficient vector de fir_compiler en séparant les uns des autres par des virgules.



Channel Specification



Implementation



Procédure :

1. Rentrer les coefficients du filtre obtenu par pyFDA ou Matlab
2. définir la fréquence d'échantillonnage à l'entrée du filtre FIR
3. définir le format des données et la taille du bus IN/OUT, ici 16 bits

Maximize Dynamic Range, les coefficients vont être retravaillés pour s'intégrer dans la taille de bus que vous aurez déterminé.

Détection synchrone numérique

Sur carte Red Pitaya 125-14

Understanding Timing Reports

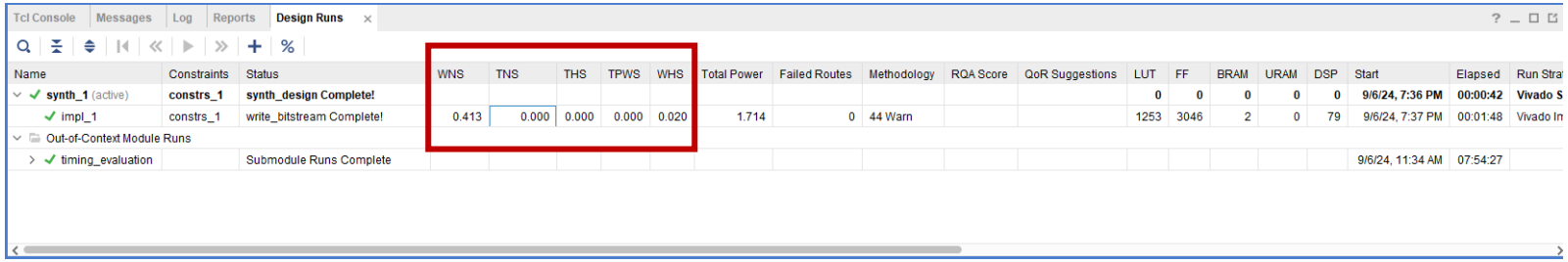
Les mesures de temps ci-dessous reflètent le score de temps de conception. Les nombres doivent être positifs pour respecter le temps. Ceci est utile comme mesure de la quantité de travail restant à faire pour respecter le timing.

WNS est la marge négative totale de votre chemin critique (dans le pire des cas).

TNS est une marge négative totale et est la somme de toutes les marges négatives.

Le « Total Negative Slack (**TNS**) » est la somme du (réel) jeu négatif dans votre conception.

Si c'est 0, alors la conception respecte le timing.

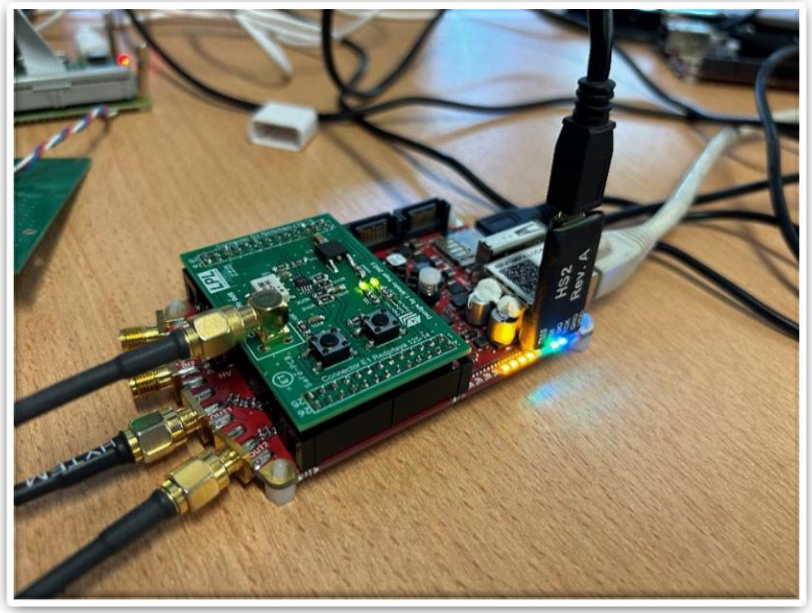


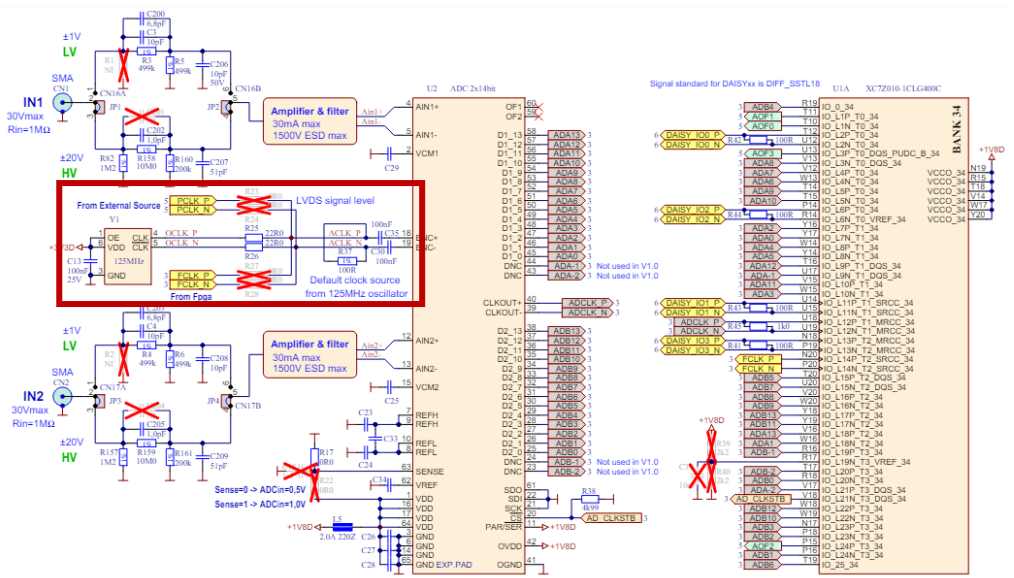
Détection synchrone numérique

Sur carte Red Pitaya 125-14

Horloge externe STEMlab 125-14 option external clock :

Cette version du STEMlab est un standard STEMlab 125-14 qui a été modifié de telle manière que l'ADC, le DAC et l'horloge FPGA peuvent être fournis par une source d'horloge externe. Une horloge externe doit être connectée aux broches Ext ADC CLK- et ADC CLK+.

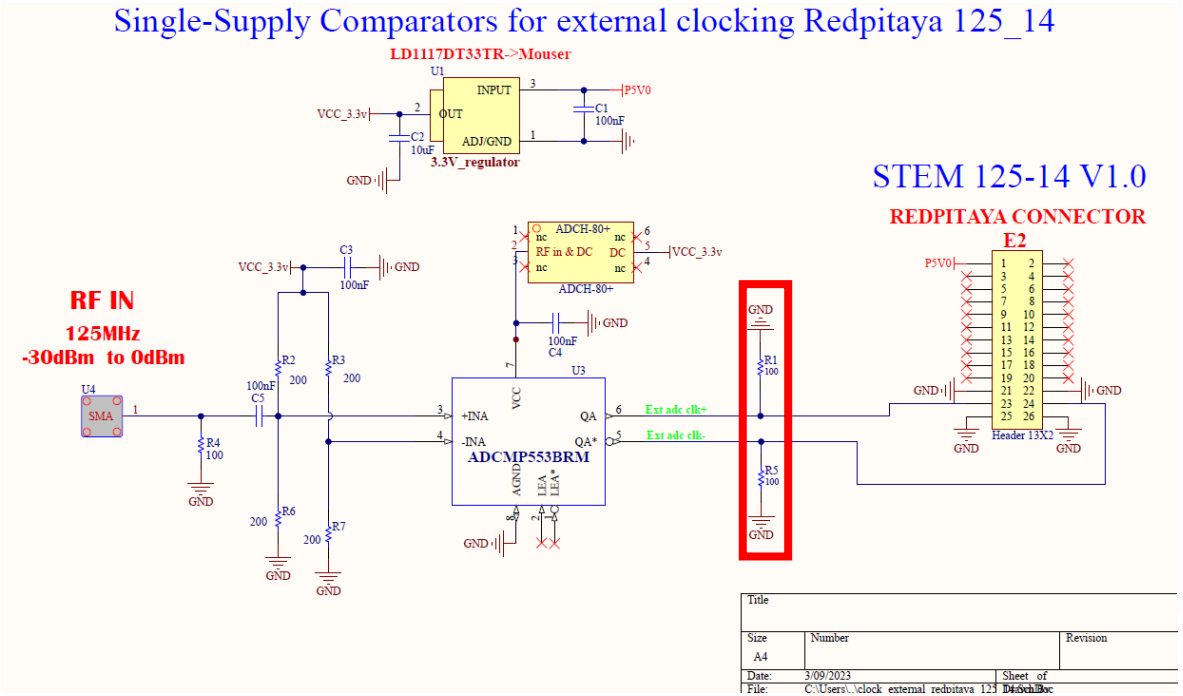


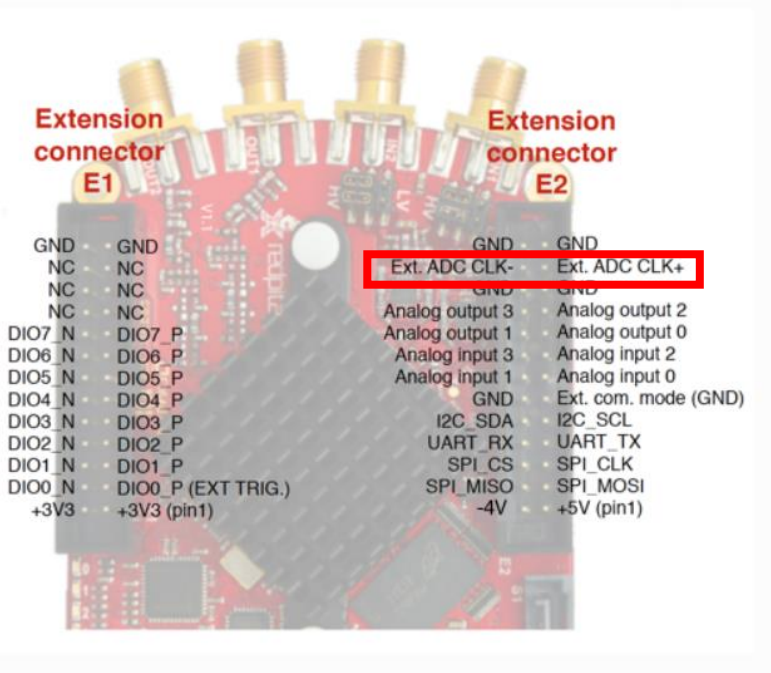


Détection synchrone numérique

Sur carte Red Pitaya 125-14

Les entrées sont différentielles et il faut donc adapter un montage pour référencer correctement l'horloge de la Red Pitaya (ZYNQ)





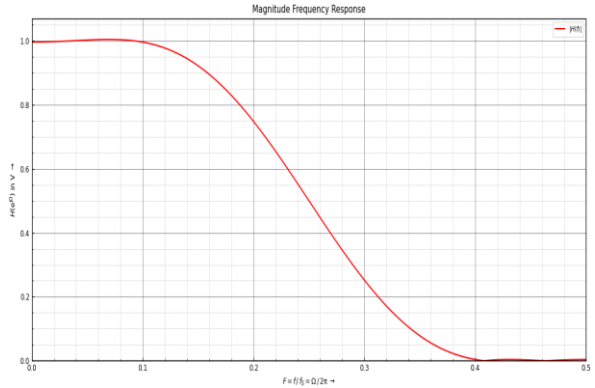
Détection synchrone numérique

Sur carte Red Pitaya 125-14

pyFDA (Python Filter Design and Analysis)

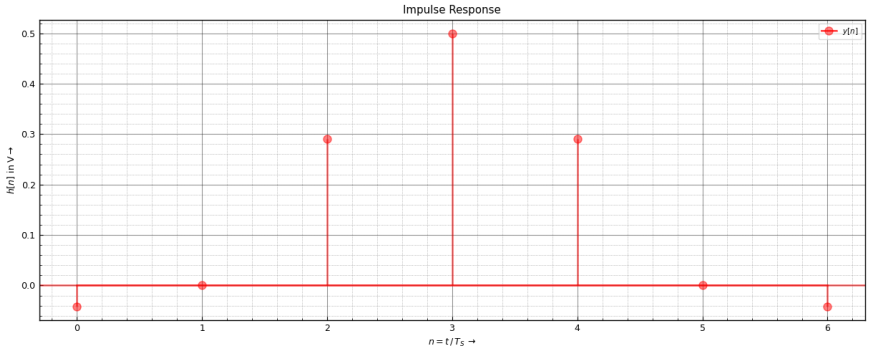
https://github.com/chipmuenk/pyfda

HBF 7 coefficient filter
b0 = b6 = -0.0423476
b1 = b5 = 0
b2 = b4 = 0.2902549
b3 = 0.5



Loop latency :
HBF1 7 cycles at clock
HBF2 7 cycles at clock/2 -> 14 cycles at clock
HBF3 7 cycles at clock/4 -> 28 cycles at clock

Fs = 125MHz
Total = 7 + 14 + 28 cycles = 49 cycles
Latency = 0,84us
Pi/4 = 2piFm x 0,84us
Fm = 1/ (8 x 0,84us) = 149KHz.



Détection synchrone numérique

Liens utiles

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

https://pavel-demin.github.io/red-pitaya-notes/
https://github.com/dspsandbox/FPGA-Notes-for-Scientists
https://github.com/fabzz60/TP_FPGA3_stepper_motor
https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14
https://github.com/fabzz60/Cmod-A7-Horloge-Demo-

Lien pour la présentation de la carte Red Pitaya avec des exemples de mise en œuvre sur VIVADO 2022.2 et VITIS 2022.2
https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

Vivado 2022.2 et Vitis 2022.2

Exporter vers VITIS le projet VIVADO 2022.2 et sauvegarder le projet HDL

Lien pour la prise en main de VIVADO et VITIS 2022.2 sur carte CORA z7 : https://github.com/fabzz60/VIVADO

Une fois le projet VIVADO transféré sur VITIS on peut générer le fichier boot.bin sur la carte micro-SD

Fichier Boot dans:
C:/Xilinx/TP_VIVADO_2022_2/Projets_VITIS_2022/red_pitaya_timing_system/_ide/bootimage/BOOT.bin

37

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.

https://github.com/fabzz60/demo_adc_dac_Redpitaya_125_14

Merci de votre attention !