Laboratoire Kastler Brossel
Physique quantique et applications

FPGA pour la metrologie

Pierre Cladé

SORBONNE UNIVERSITÉ

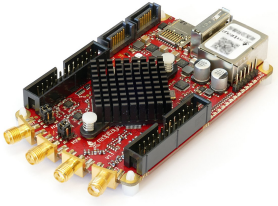
cnrs

ENS
ÉCOLE NORMALE SUPÉRIEURE

COLLÈGE DE FRANCE
1530

Introduction

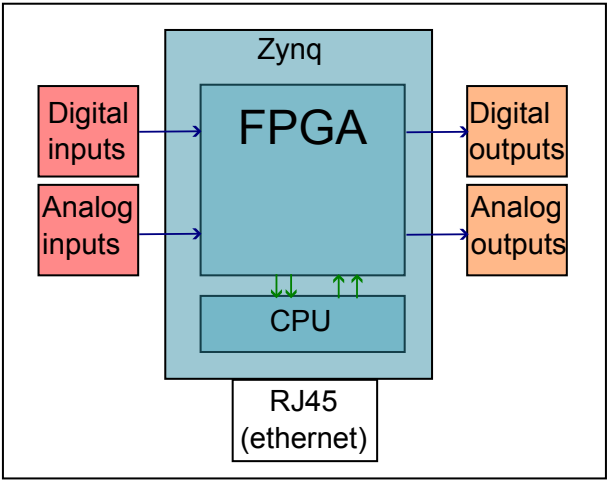
- Utilisation de FPGA depuis plus de 10 ans dans l'équipe d'interférométrie atomique du LKB
  - Asservissement de laser (traitement digital du signal)
  - Pilotage de manip (séquence temporelle)
- Développement d'une librairie Python pour la création d'applications (librairie odeon)
- Hardware : Redpitaya (125 MEch/s) et Koheron Alpha250 (250 MEch/s, bruit plus faible)



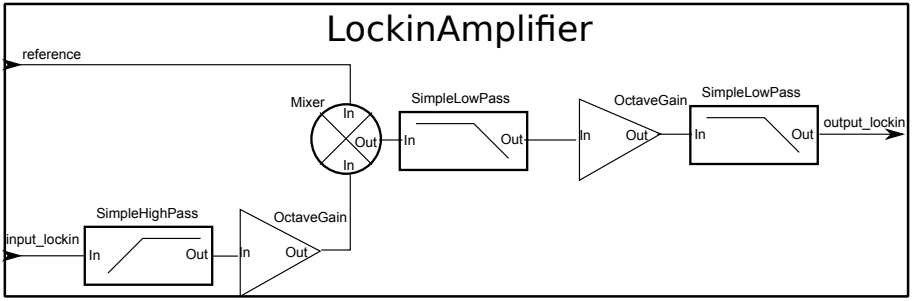
Qu'est-ce qu'un FPGA

- FPGA
  Un réseau de portes programmables (field-programmable gate array - FPGA) est un circuit intégré conçu pour être configuré par l'utilisateur. En utilisant un langage de programmation approprié, l'utilisateur pourra choisir les fonctions matérielles que le FPGA exécutera.
- Différence entre un CPU et un FPGA. Exemple de la réalisation d'un plat de cuisine
  - CPU : un cuisinier qui possède beaucoup de talents. Recette = programme à suivre. Exécution séquentielle
  - FPGA : une usine. Chaque ouvrier réalise une tache spécifique (et rien d'autre). Exécution en parallèle. Pipeline.
- Avantages FPGA vs CPU
  - CPU : Horloge plus rapide (GHz), plus précis (64 bits), instructions complexe (par exemple float)
  - FPGA : parallélisation de tous les calculs, indépendance des instructions (rajouter quelque chose de ralentit pas le reste), flux de donnée plus important.
- Inconvénient des FPGA : programmation difficile (reste de bas niveau). Langage particulier (Hardware Description Language comme le Verilog ou VHDL).

Redpitaya board



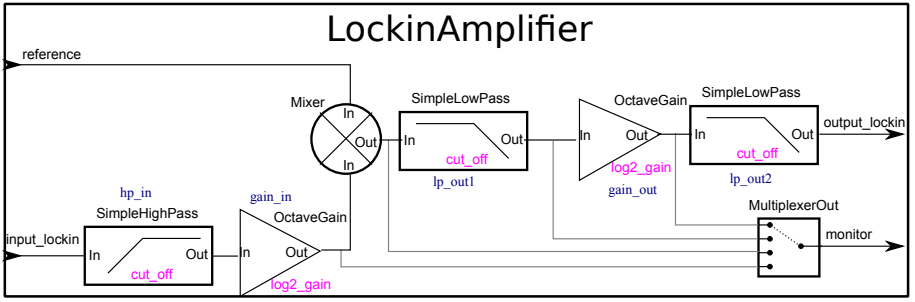
Exemple (simple) : détection synchrone



- A chaque cycle une boite calcule une nouvelle valeur pour la sortie
- Exemple : filtre passe bas (coupure à $f_{\mathrm{clk}}/2^N$)

```
output.next = output + (input - output)>>N
```

Exemple (simple) : détection synchrone



- A chaque cycle une boite calcule une nouvelle valeur pour la sortie
- Exemple : filtre passe bas (coupure à $f_{\mathrm{clk}}/2^N$)

```
output.next = output + (input - output)>>N
```

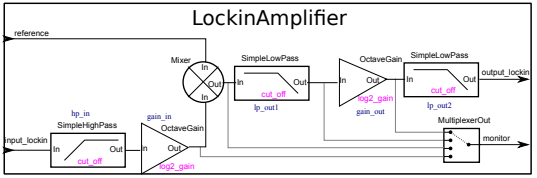
Librairie odeon

- Création des blocs élémentaire (myhdl)
- Gestion automatique des registres (paramètres)
- Assemblage récursif des blocs (et des registres)
- Création des interfaces utilisateur (API python, API SCPI, interface graphique)

Exemple (simple) : détection synchrone

```
class LockinAmplifier(Module):
    # Deinition of the implemented modules
    hp_in = SimpleHighPass
    mixer = Mixer
    lp_out1 = SimpleLowPass
    lp_out2 = SimpleLowPass
    gain_in = OctaveGain
    gain_out = OctaveGain
    mux = LockinMultiplexerOut
```

Blocks




```
    def hdl_logic(self, clk, input_lockin, reference, output_lockin, monitor,
                  **kwd):
        hp_in_o = FixedSignal(0, iwl=0, fwl=17) # Definitions of the signals
        gain_in_o = FixedSignal(0, iwl=0, fwl=17)
        lp_out1_o = FixedSignal(0, iwl=0, fwl=17)
        gain_out_o = FixedSignal(0, iwl=0, fwl=17)
        out_mixer = FixedSignal(0, iwl=0, fwl=17)
```

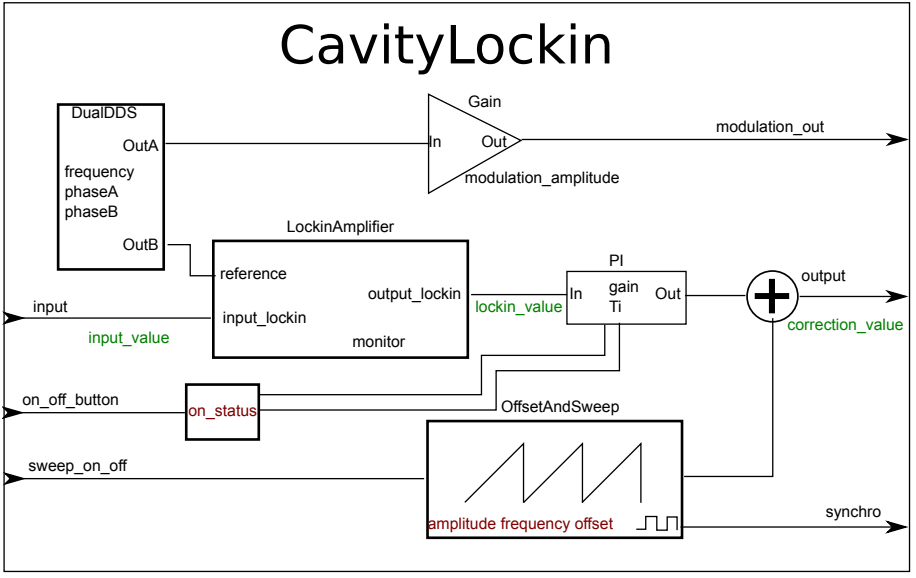
Wires

```
        # Connections between modules using the signals
        hp_in = self.hp_in.hdl(clk, input_lockin, hp_in_o)
        gain_in = self.gain_in.hdl(clk, hp_in_o, gain_in_o)
        mixer = self.mixer.hdl(clk, gain_in_o, reference, out_mixer)
        lp_out1 = self.lp_out1.hdl(clk, out_mixer, lp_out1_o)
        gain_out = self.gain_out.hdl(clk, lp_out1_o, gain_out_o)
        lp_out2 = self.lp_out2.hdl(clk, gain_out_o, output_lockin)
        mux = self.mux(clk, signals=locals(), monitor=monitor)

        return instances()
```

Interconnections

Détection synchrone ⇒ Lock box

Développée pour le projet GBAR (contrôle à distance d'un Ti:Sa)

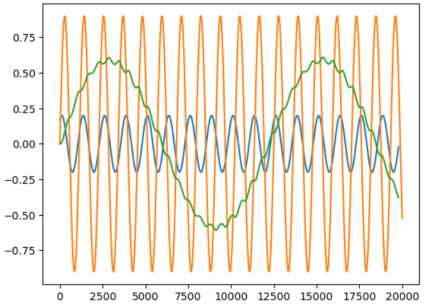


Librairie odeon

- Principe de la librairie
  - Création des blocs élémentaires (myhdl)
  - Gestion automatique des registres (paramètres)
  - Assemblage récursif des blocs (et des registres)
  - Création des interfaces utilisateur (python, SCPI, graphique)
- Utilisation
  - Création d'une application (assemblage des blocs)
  - Simulation de l'application (jupyter notebook)
  - Compilation (quelques minutes). Makefile.
  - Chargement du fichier binaires de plusieurs façons
    - Au démarrage de la carte
    - Interface graphique
    - Interface python

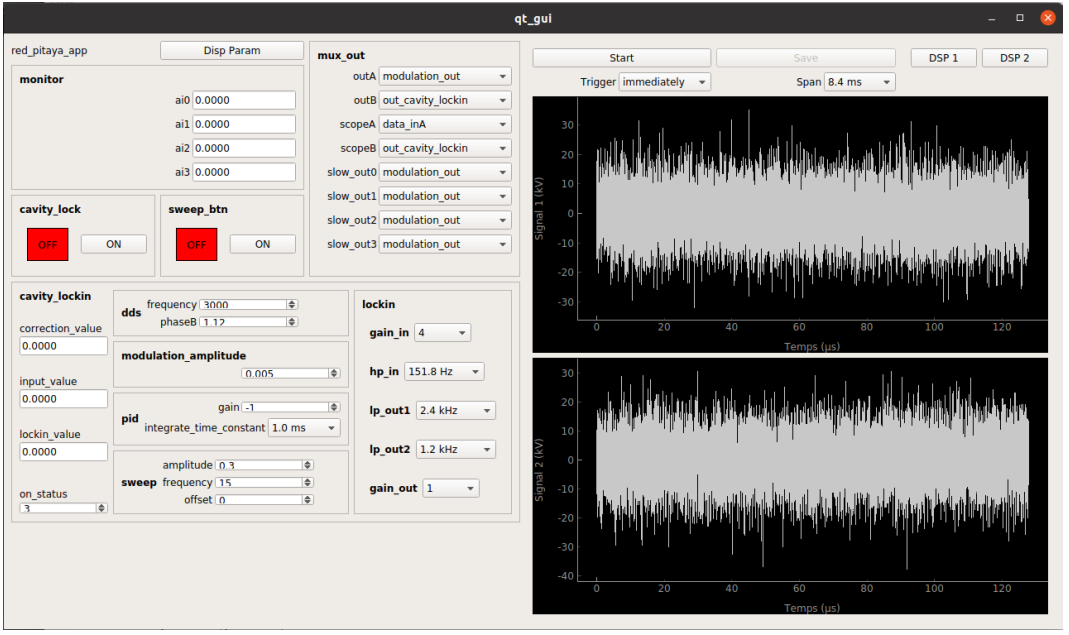
Simulation

```
app = SimpleLockbox()

app.lockin_pid.dds.frequency = 100000
app.lockin_pid.modulation_amplitude.gain = 0.2
app.lockin_pid.lockin.hp_in.cut_off = 10000
app.lockin_pid.lockin.gain_in.log2_gain = 2
app.lockin_pid.lockin.lp_out1.cut_off = 30000
app.lockin_pid.lockin.lp_out2.cut_off = 30000
app.lockin_pid.lockin.gain_out.log2_gain = 0
app.mux_out.outA = 'data_inA'


N = 20000
t = np.arange(N)*8E-9
data_inA = .9*np.sin(2*np.pi*t*110000)
app.data_inA = data_inA

app.simulate(max_clk_cycles=N, start_after_end_of_command=True)
```

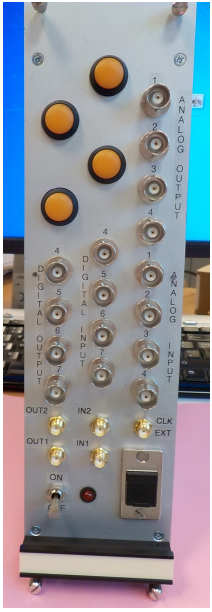


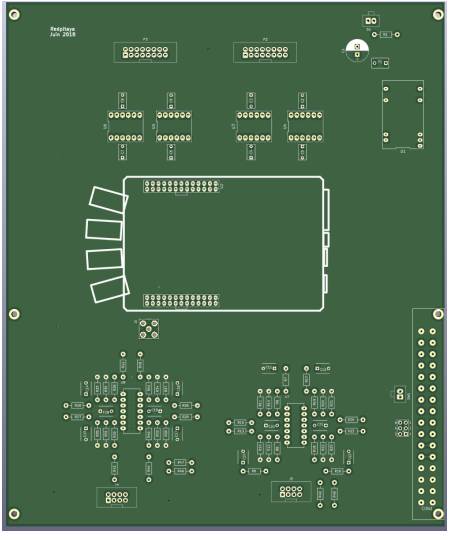
- Le code pour utiliser l'application est le même que celui pour la simulation

Interface graphique



Hardware


ANALOG OUTPUT
1
2
3
4
DIGITAL OUTPUT
4
5
6
7
DIGITAL INPUT
4
5
6
7
ANALOG INPUT
1
2
3
4
OUT2
IN2
CLK EXT
OUT1
IN1
ON



Librairie odeon

- Odeon
  - Coeur de la librairie : 20 000 lignes
  - Test unitaires
  - Continuous integration (machine dédiée pour la compilation), gitlab du CNRS.
  - Fonctionne sur la platine Koheron Alpha 250 (200 MS/s, bruit plus faible que le redpitaya)
- Modules
  - Filtres, PID
  - DDS, synthétiseur RF, générateur arbitraire (interpolation de spline)
  - Détecteur de phase (conversion cartésien vers polaire), détection synchrone, frequency to voltage converter
  - Compteur de photon / horodateur.
  - Oscilloscope (2 fois 2^{14} points sur le redpitaya, 2^{16} pour Alpha250)
- Todo
  - Site web + finir la documentation
  - Mise à disposition du code (choix d'un licence open source)

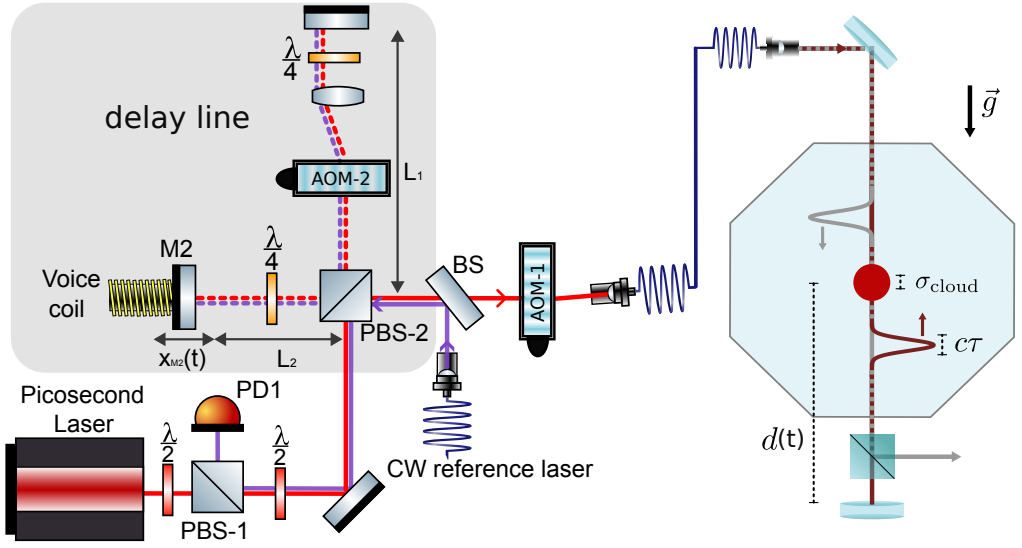
Exemples

- Générateur arbitraire de fréquence / driver AOM
  - Sortie digitale à 111.11 MHz issue de la PLL du FPGA (fixe)
  - Sortie analogique du redpitaya à 31 MHz
  - Mixer et filtre matériel : 80MHz

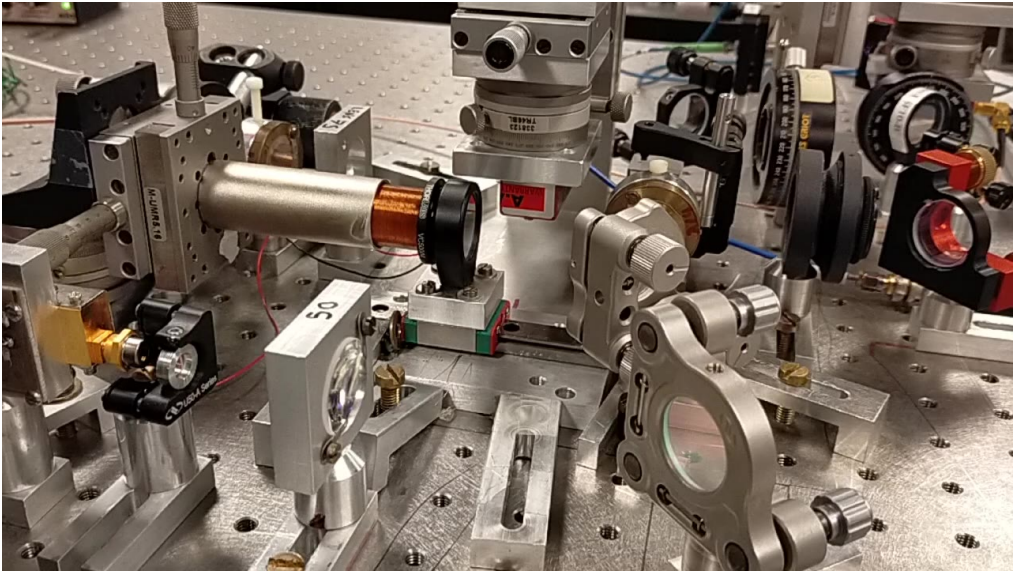
  Sortie contrôlable en fréquence et amplitude. Possibilité de rajouter un asservissement ou des modulations supplémentaires. Sécurité hardware (protection d'un ampli EDFA)
- Asservissement d'un laser sur un peigne de fréquence
  - Battement à 15 MHz
  - Asservissement en fréquence ou en phase (avec dépliement de la phase sur plusieurs tours).
  - Monitoring de l'asservissement

- DDS avec rampe de fréquence. Simulation exacte de la phase et mesure directe dans le FPGA. Rampe de fréquence à pas non entier.

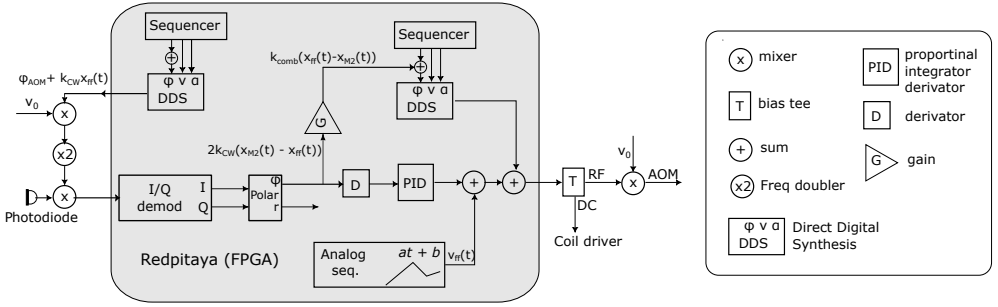
Asservissement d'une ligne à retard mobile

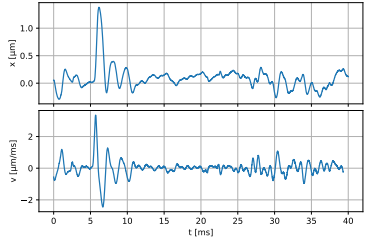


Asservissement d'une ligne à retard mobile



Asservissement d'une ligne à retard mobile





C. Debavelaere et al., Atom interferometer using spatially localized beam splitters. Phys. Rev. A 110, 013310 (2024).

Conclusion

- Odeon : une boite à outil Python pour le traitement du signal avec des FPGAs.
- Projet stable depuis plusieurs années. Tests unitaires. Intégration continue
- Déploiement rapide de nouvelles applications. Indispensable pour certains projets.
- Redpitaya et Koheron Alpha250

Futur ?

- Tutoriel/video
- Enseignement/formation sur les FPGAs
- Interface web dans l'instrument, boîtier 100% autonome

pierre.clade@lkb.upmc.fr