Refimeve+
Réseau fibré métrologique à vocation européenne

femto-st SCIENCES & TECHNOLOGIES

AG – 20 octobre 2018, Besançon ensmm

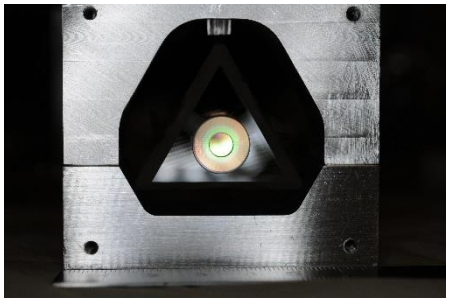
Cavités ultra-stables et raccordement au réseau REFIMEVE+

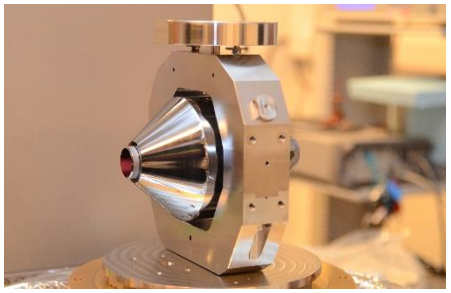
J. Millo, B. Marechal, S. Denis, F. S. Tetsing, A. Didier, A. Kumar, B. Dubois, C, Fluhr, G. Goavec, C. Cardenas, P-Y. Bourgeois, V. Giordano, M. Delehaye, C. Lacroûte, E. Rubiola, Y. Kersalé

cnes RÉGION BOURGOGNE FRANCHE COMTE UBFC UFC UNIVERSITÉ DE FRANCHE-COMTÉ cnrs dépasser les frontières PROJET FINANCÉ PAR L'ANR ANR PROJECT FUNDED BY THE ANR FIRST TF Oscillator IMP Oscillator Instability Measurement Platform

Outline

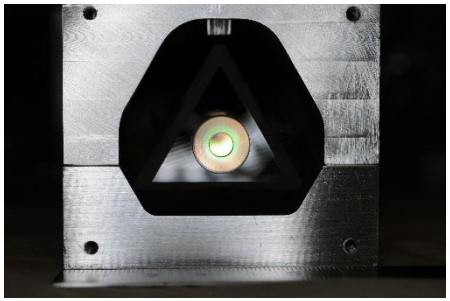
- Compact cavity stabilized laser
- Silicon cavity stabilized laser
- Digital electronic
- Outlook: Refimeve+



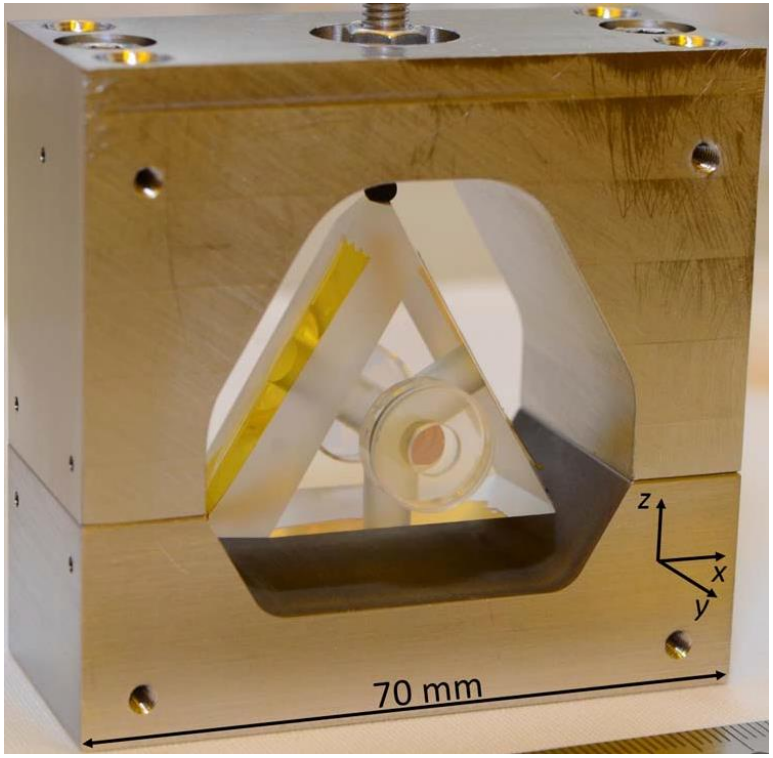


Outline

- Compact cavity stabilized laser
- Silicon cavity stabilized laser
- Digital electronic
- Outlook: Refimeve+



Compact ultra-stable cavity

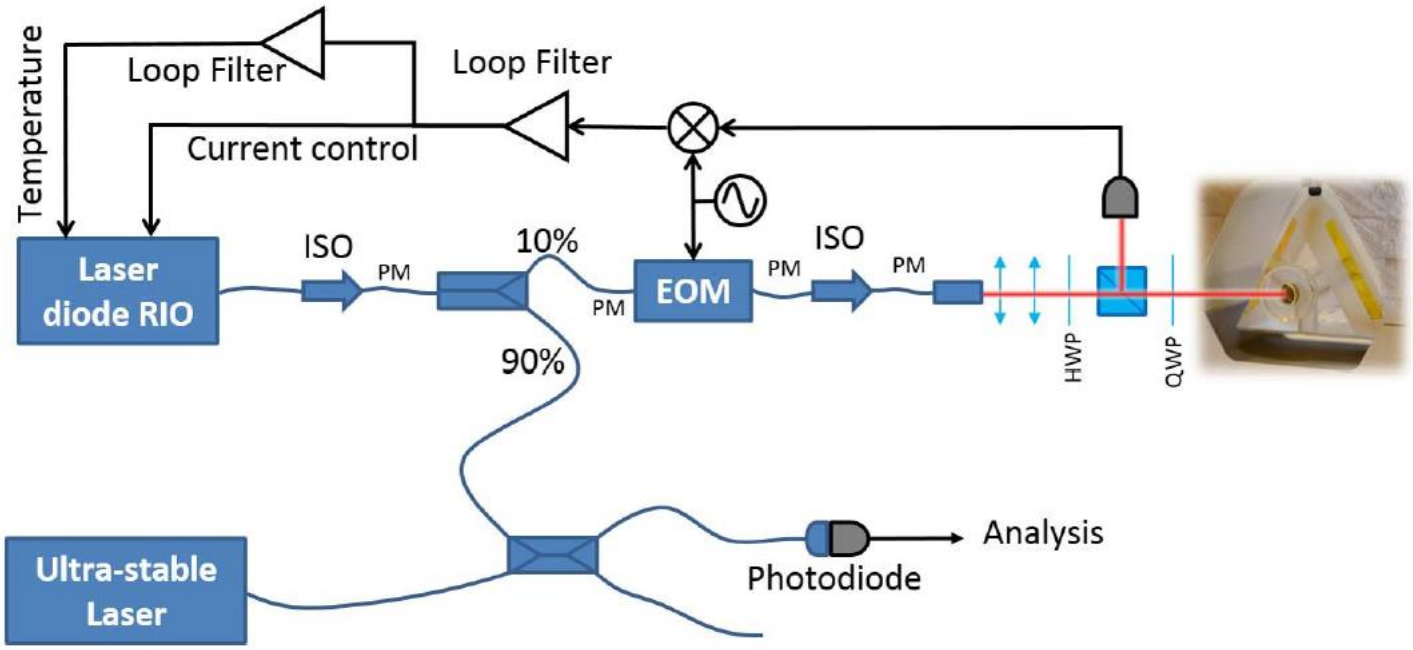


- 25 mm long cavity
- ULE spacer and SiO2 mirrors
- Low loss crystalline coating
- ULE rings
- Compact vacuum chamber
- 30 L (excluding electronic)

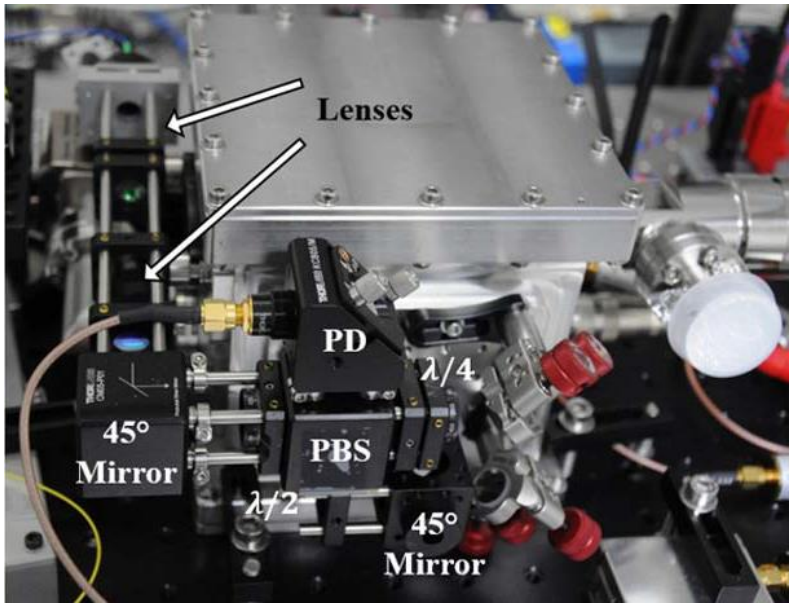
Expected thermal noise limit :

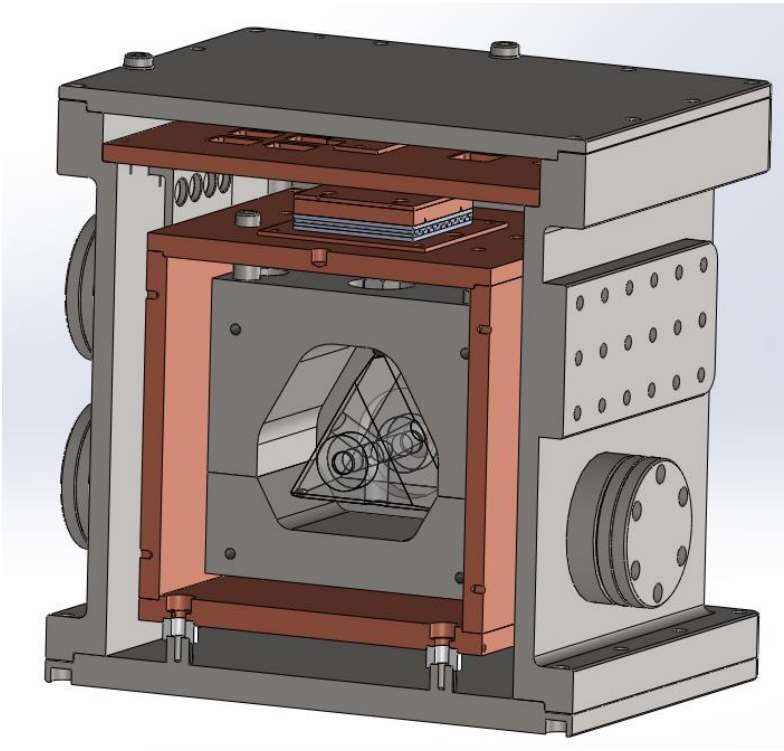
$$\sigma_y(\tau) \approx 2 \times 10^{-15}$$

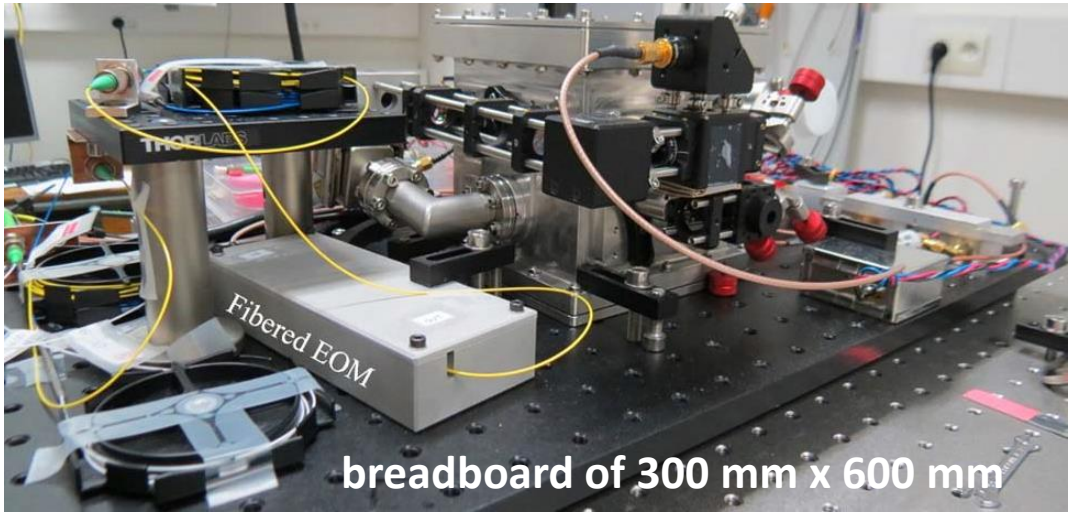
Setup


Temperature
Loop Filter
Loop Filter
Current control
Laser diode RIO
ISO
PM
10%
90%
PM
EOM
PM
ISO
PM
HWP
QWP
Ultra-stable Laser
Photodiode
Analysis

Setup

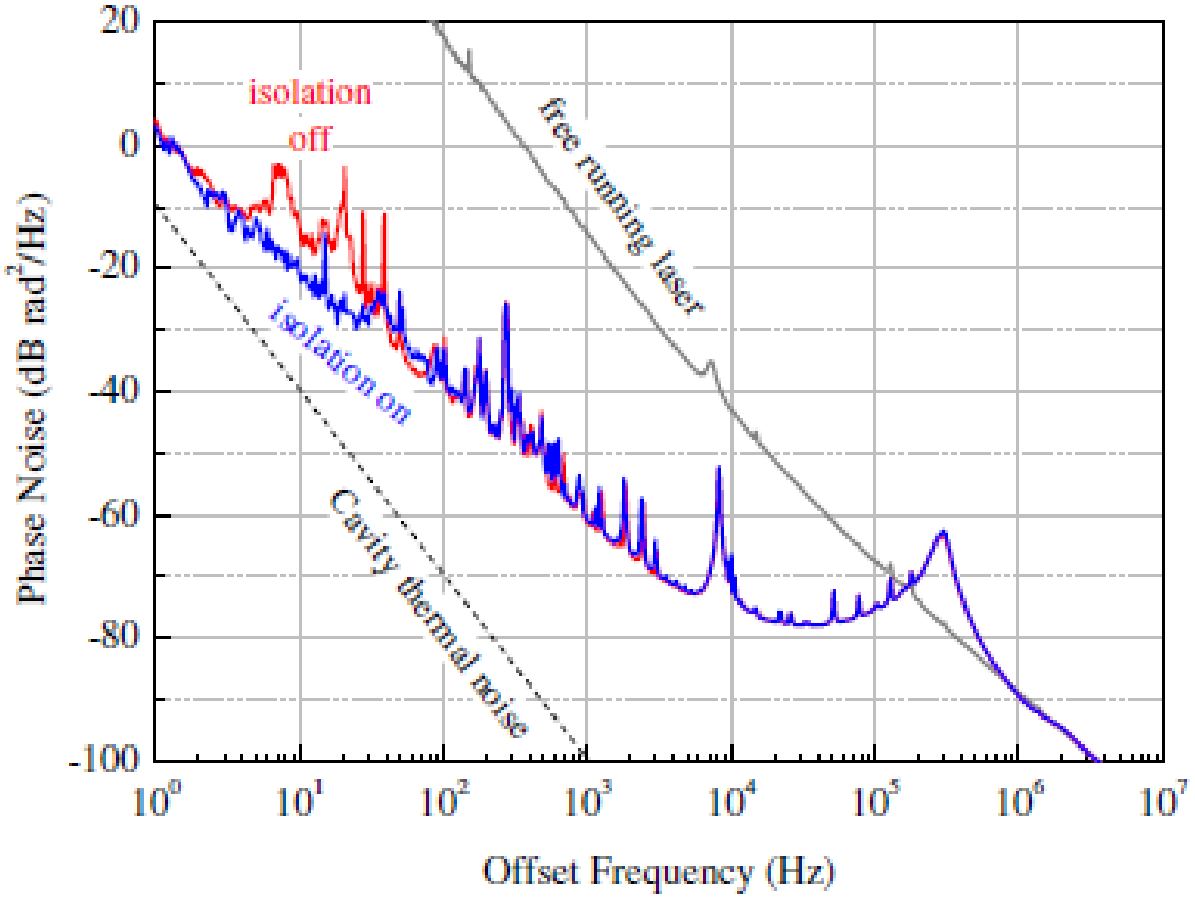

Lenses
PD
λ/4
45°
Mirror
PBS
λ/2
45°
Mirror




Fibered EOM
breadboard of 300 mm x 600 mm

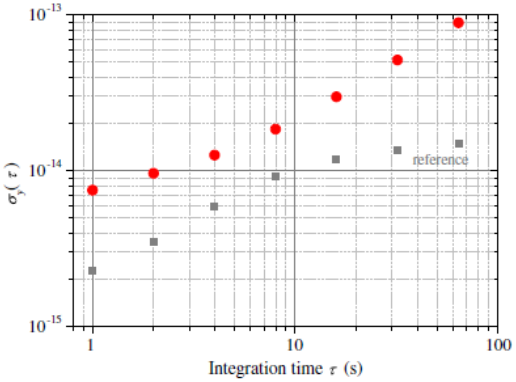
Characterization

Phase noise



Fractional frequency stability measured :

$$\sigma_y(1\,\mathrm{s}) \approx 7.5\times10^{-15}$$

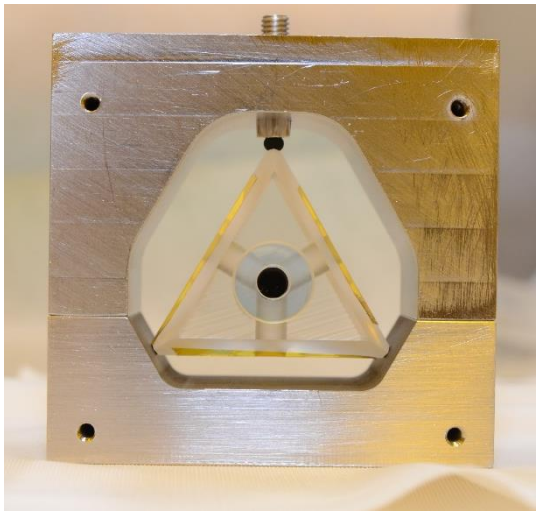


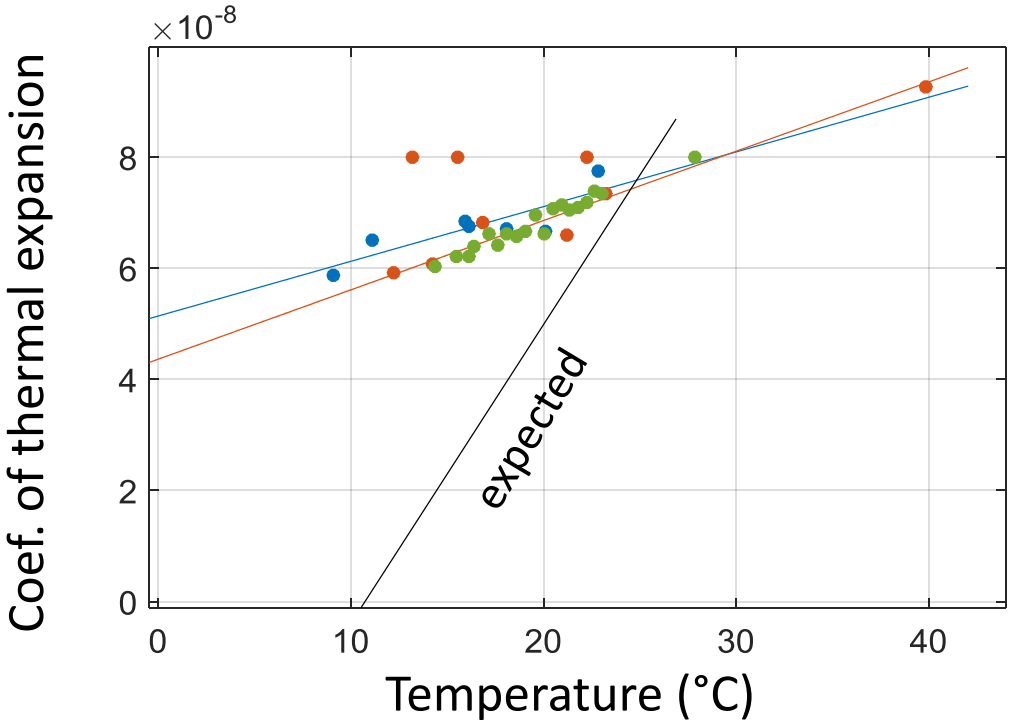
Some issues with …

Vibrations sensitivity:

Direction	Measured at 2 Hz	Measured at 6 Hz
x	$1,5 \times 10^{-11}/\mathrm{m{\cdot}s^{-2}}$	$1,2 \times 10^{-11}/\mathrm{m{\cdot}s^{-2}}$
y	$8,7 \times 10^{-12}/\mathrm{m{\cdot}s^{-2}}$	$7,7 \times 10^{-12}/\mathrm{m{\cdot}s^{-2}}$
z	$1,1 \times 10^{-10}/\mathrm{m{\cdot}s^{-2}}$	$1,9 \times 10^{-10}/\mathrm{m{\cdot}s^{-2}}$

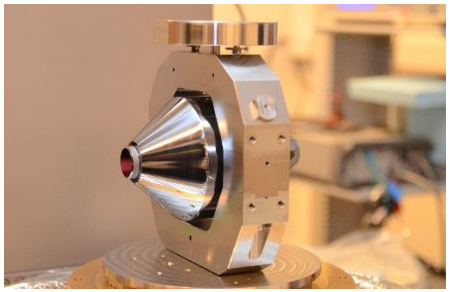
CTE of the full cavity:





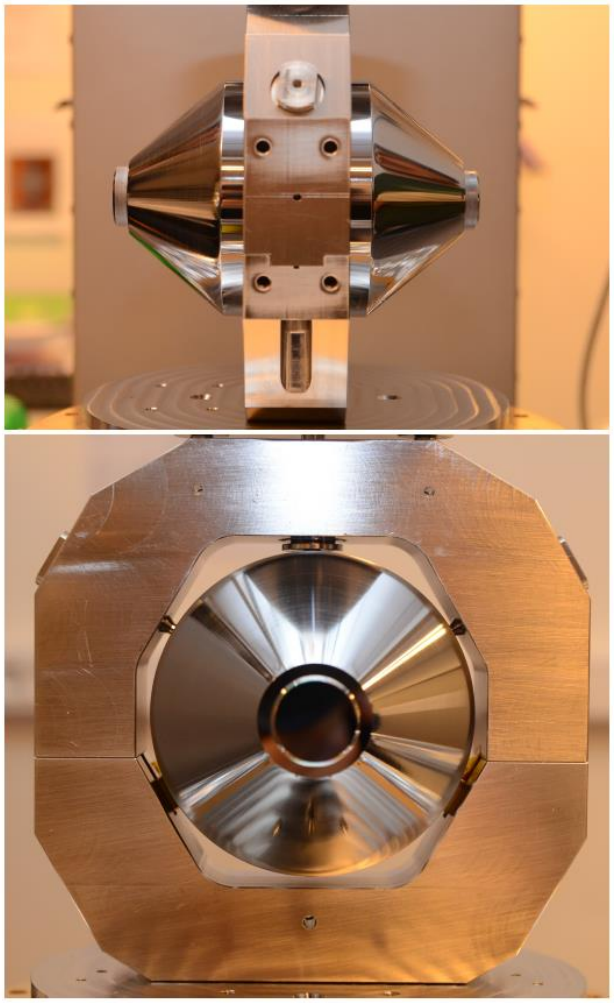
Outline

- Compact cavity stabilized laser
- Silicon cavity stabilized laser
- Digital electronic
- Outlook: Refimeve+



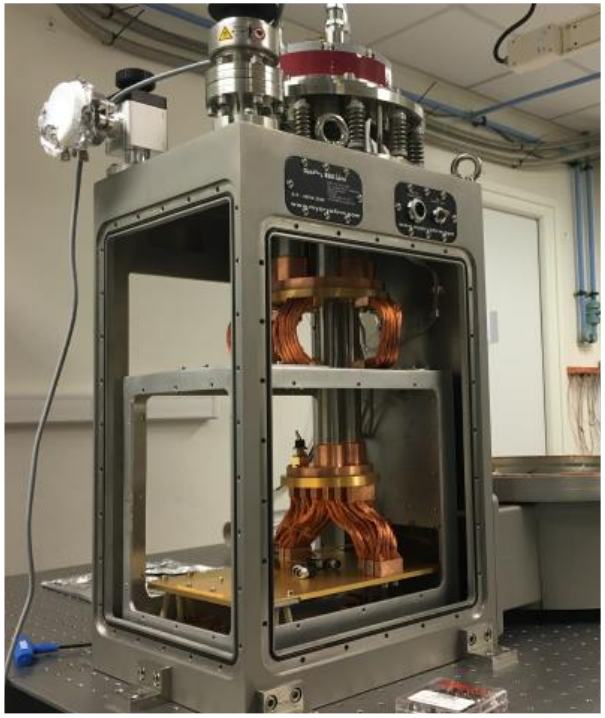
Silicon cavity stabilized laser

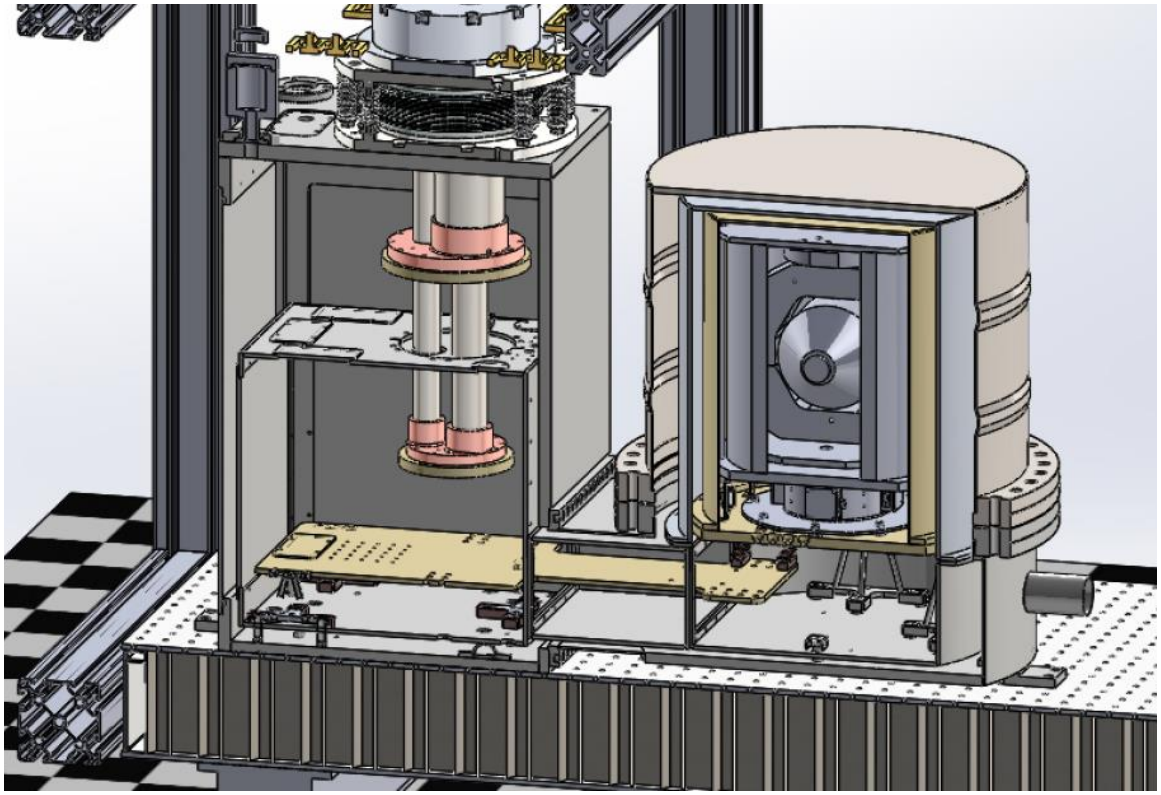
- Mono-crystalline silicon
- Zero CTE at **17 K** and 124 K
- Dielectric coatings
- 140 mm long
- Thermal noise limit :
  $\sigma_y(\tau) = 3 \times 10^{-17}$
- Finesse : 78000



Cryocooler setup

Pulsed tube based: PT410,
1W@4K, 3.4K min loaded

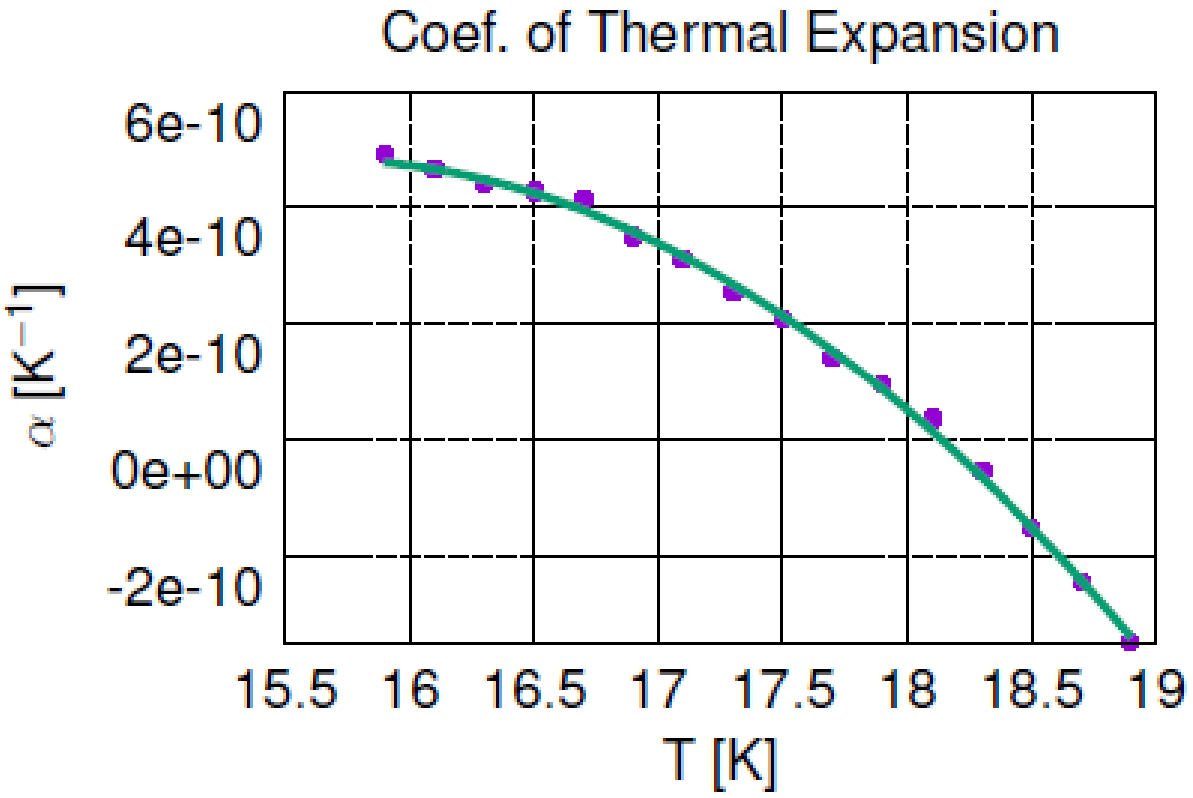




Temperature sensitivity

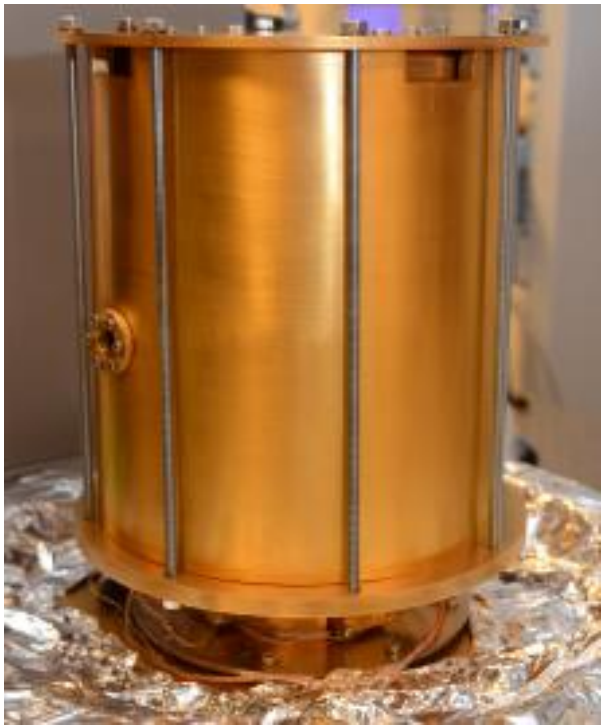
0-CTE measured at 18.1 K

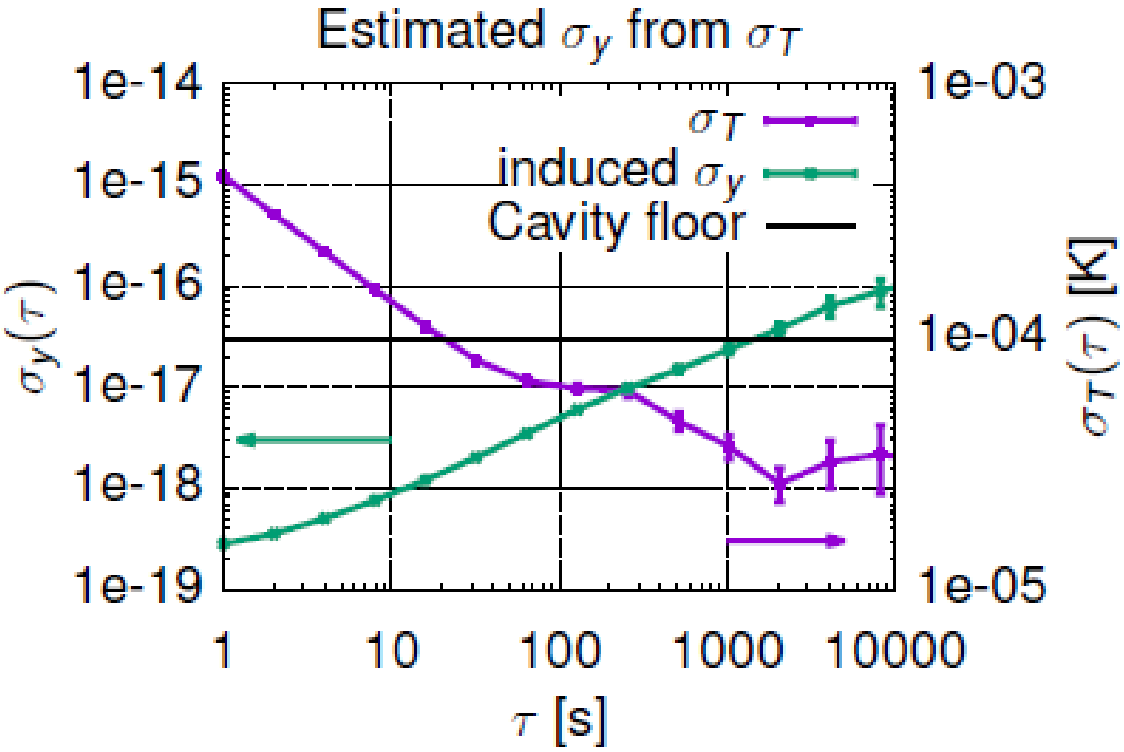
Residual sensitivity : $-3.9\ 10^{-10}\ K^{-2}$



Impact on Frequency Stability

Time constant : 6600 s

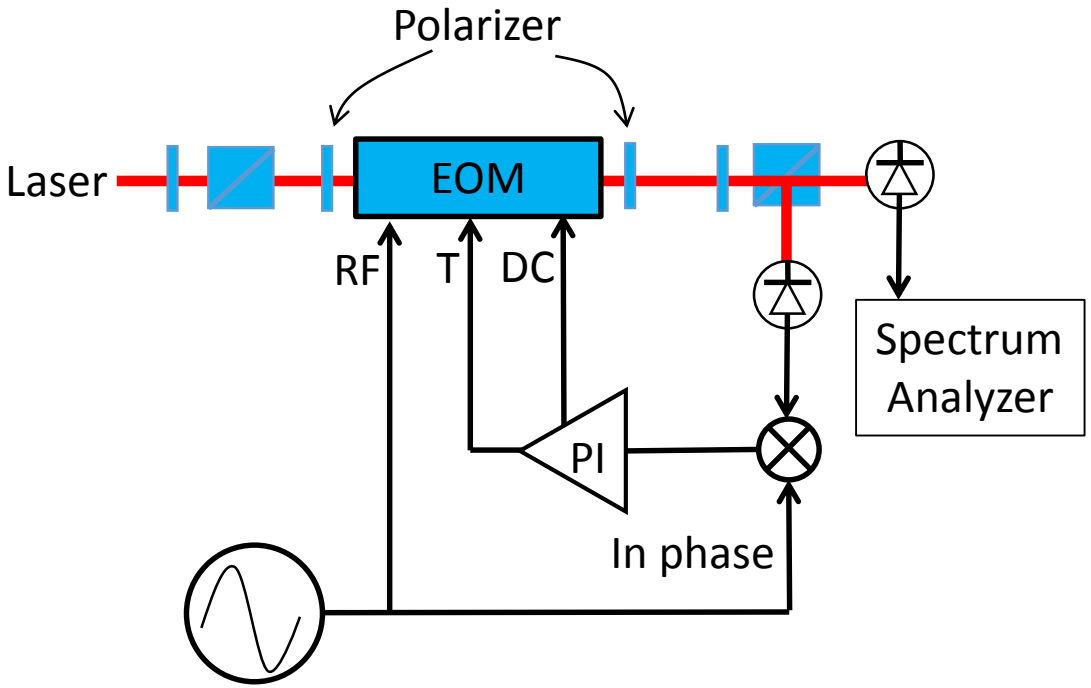


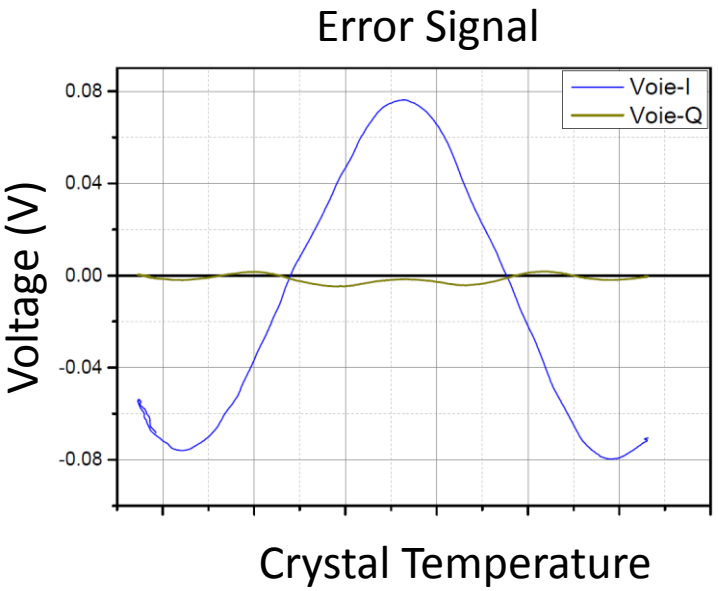


Residual amplitude modulation

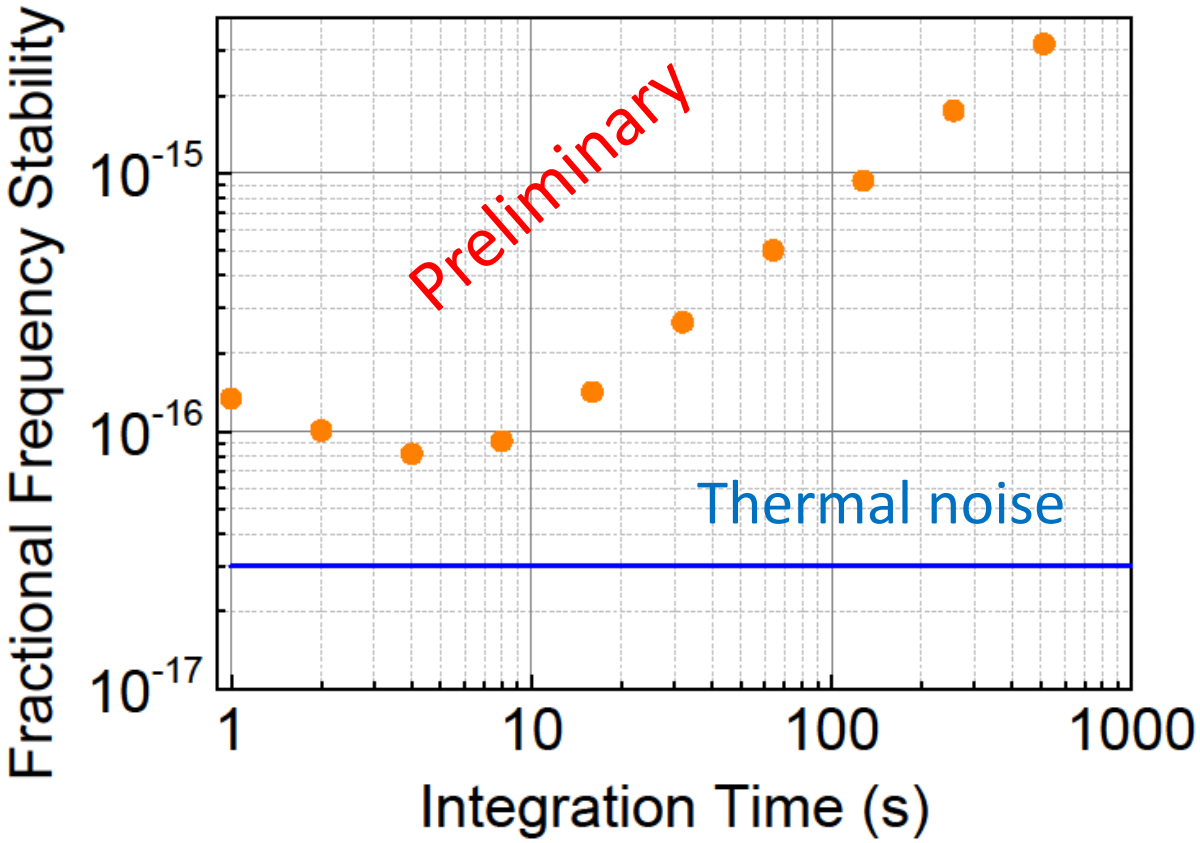
$$E(t) = E_0\left(1 + m(t)\sin(\Omega t)\right)\exp\left(j\omega t + j\beta(t)\sin(\Omega t)\right)$$

$$y_{RAM}(t) = m(t)\frac{\Delta\nu}{4\nu_l}\sqrt{\frac{P_c}{P_s}}\left(\frac{P_s}{P_c} - \mathrm{Re}(R_0)\left(1 + 2\frac{P_s}{P_c}\right)\right)$$





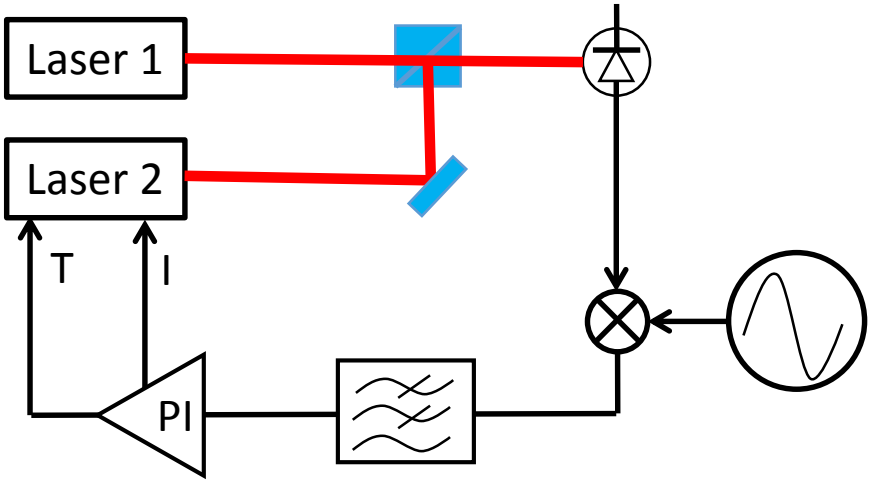
Residual amplitude modulation

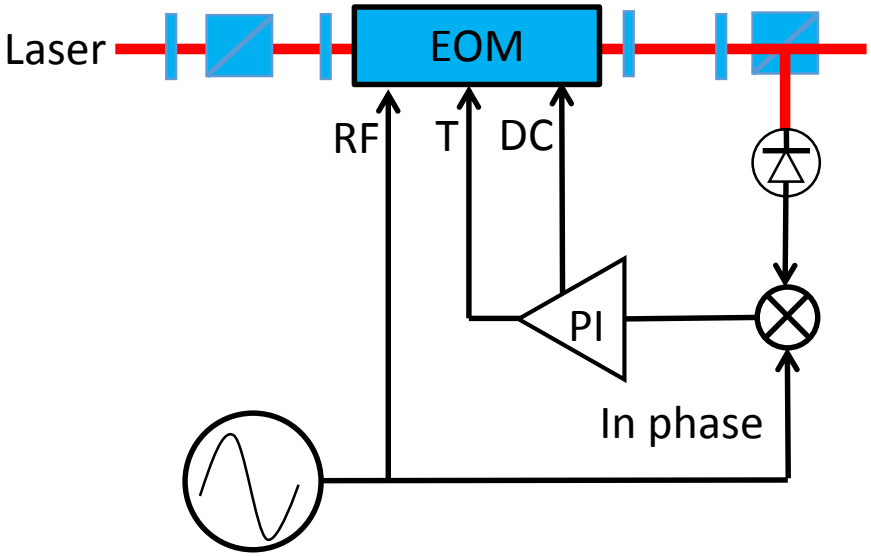


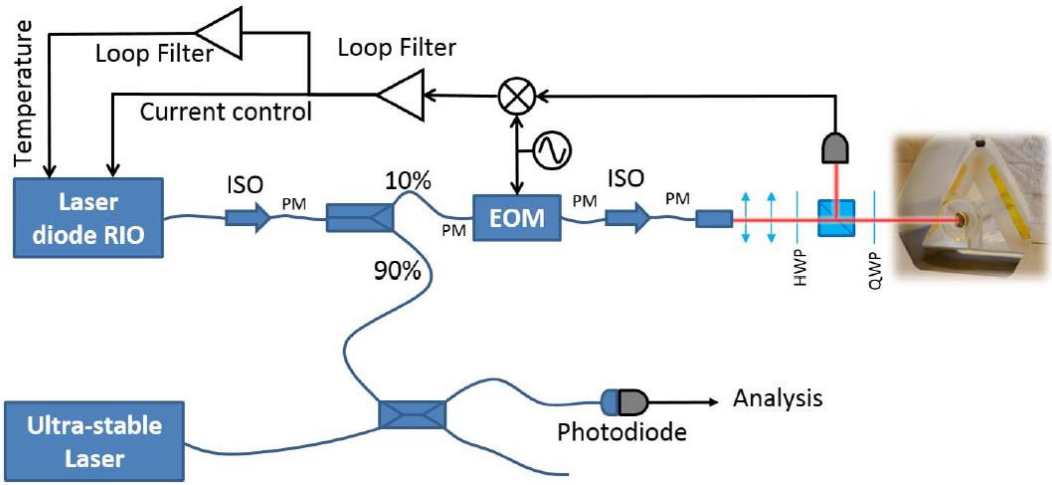
Outline

- Compact cavity stabilized laser
- Silicon cavity stabilized laser
- Digital electronic
- Outlook: Refimeve+

Digital Electronic







Demodulation I Q
Loop Filter
Synthesizer / DDS

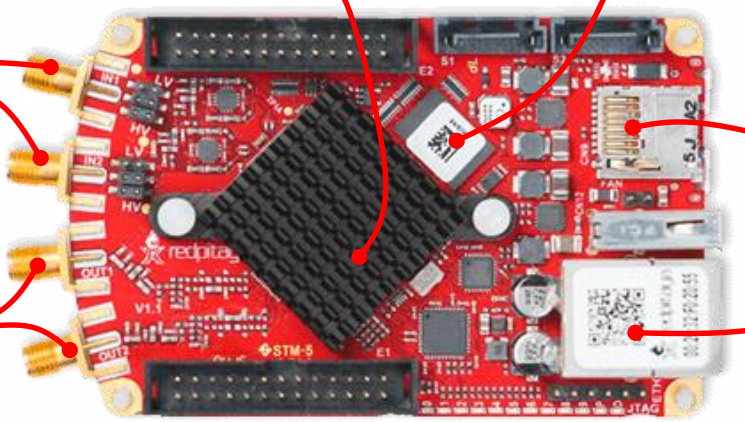
RedPitaya

SoC : FPGA and Microprocessor

RAM (memory)

Outputs:
125 MS/s, 14 bits (+/- 1 V)

Inputs:
125 MS/s, 14 bits (+/- 1 V or +/- 10 V)



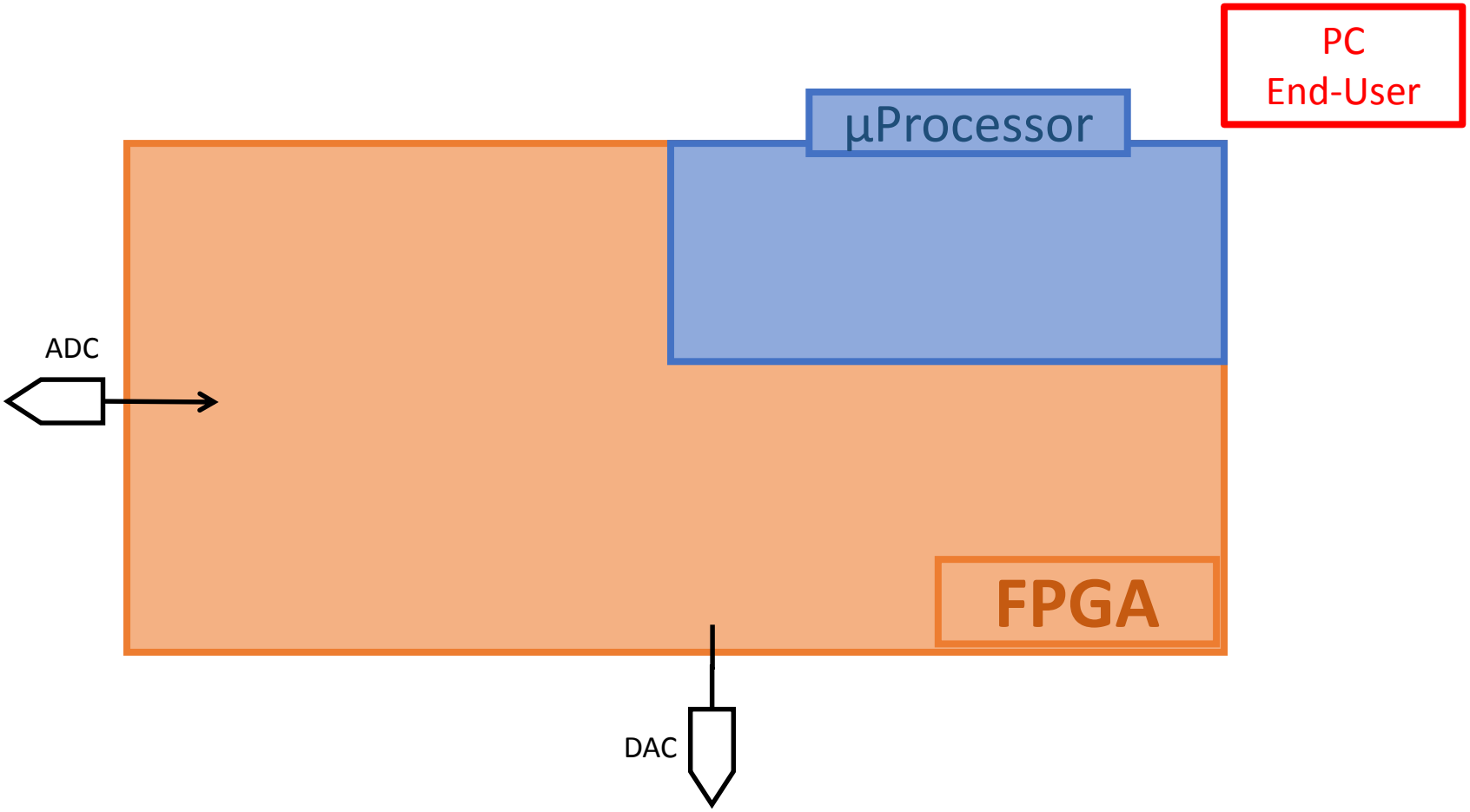
Micro SD Slot

Ethernet

FPGA (Field-Programmable Gate Array):

- logic functions
- Multipliers (48 bits)
- Memory RAM, DMA (CPU-FPGA data transfert)
- LookUp tables (LUT)

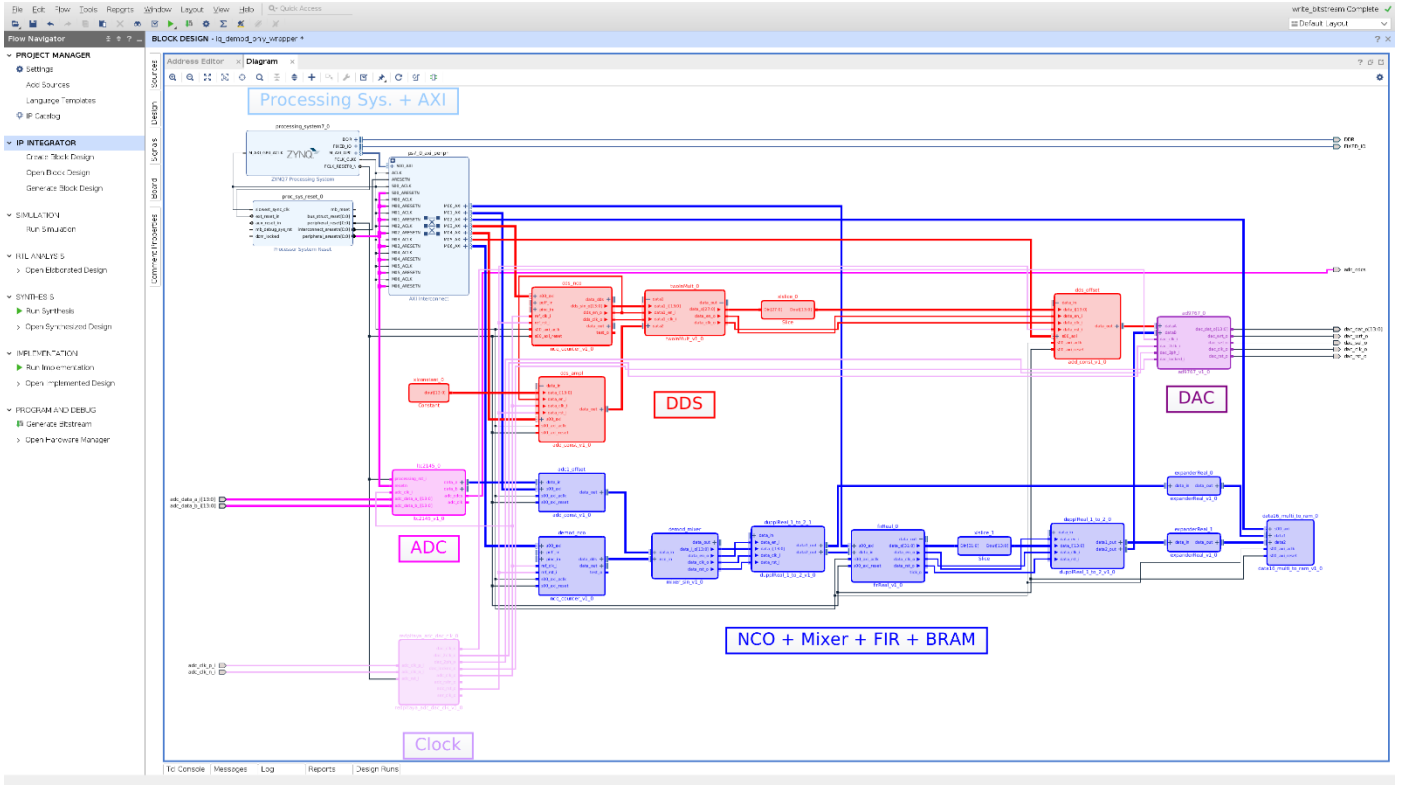
Step 0: …



Step 1: Design of FPGA functions

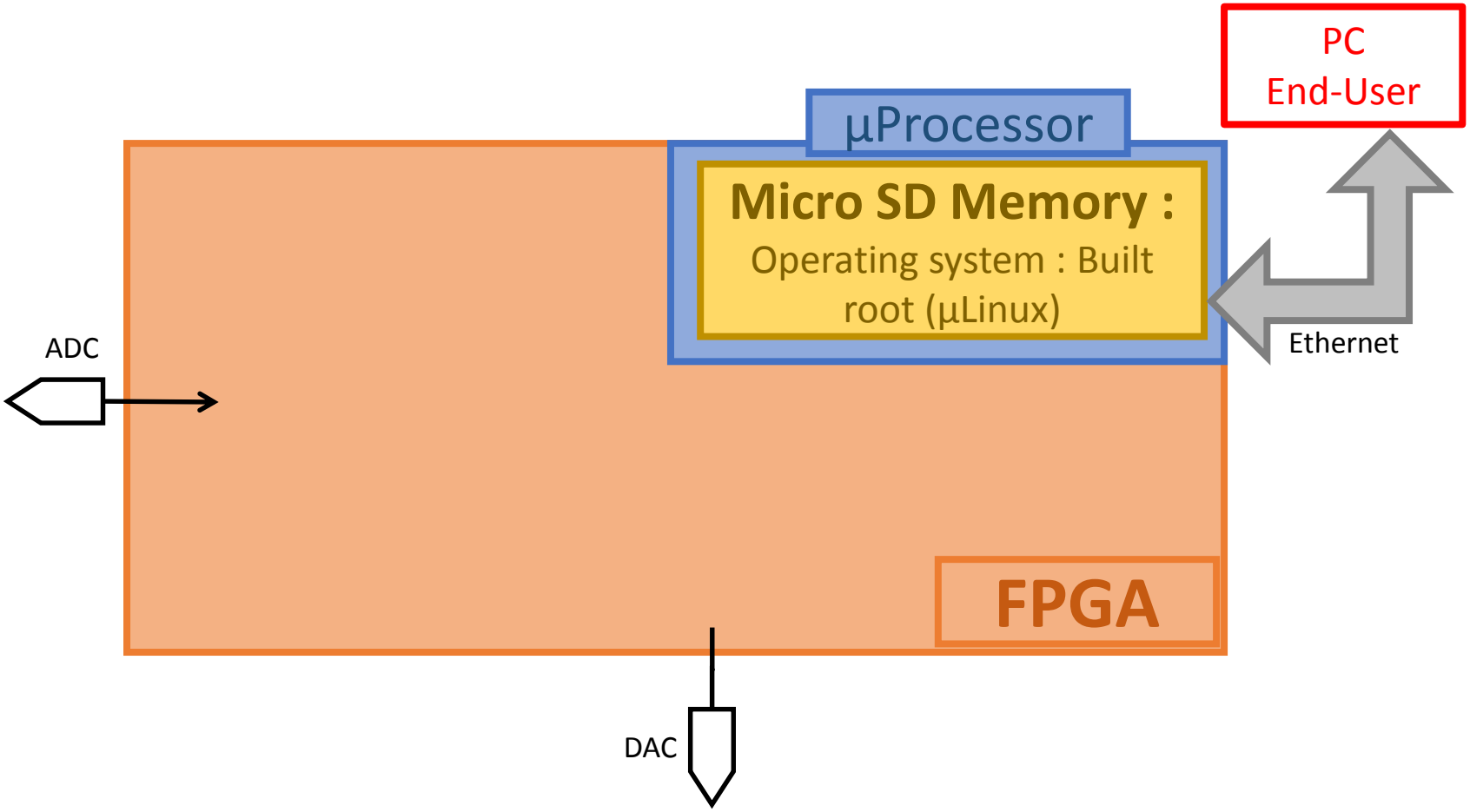
Use of Xilinx Software (Vivado)

Based on Intellectual Properties (IP) Core elements



Generation of bitstream file used for Programming the FPGA

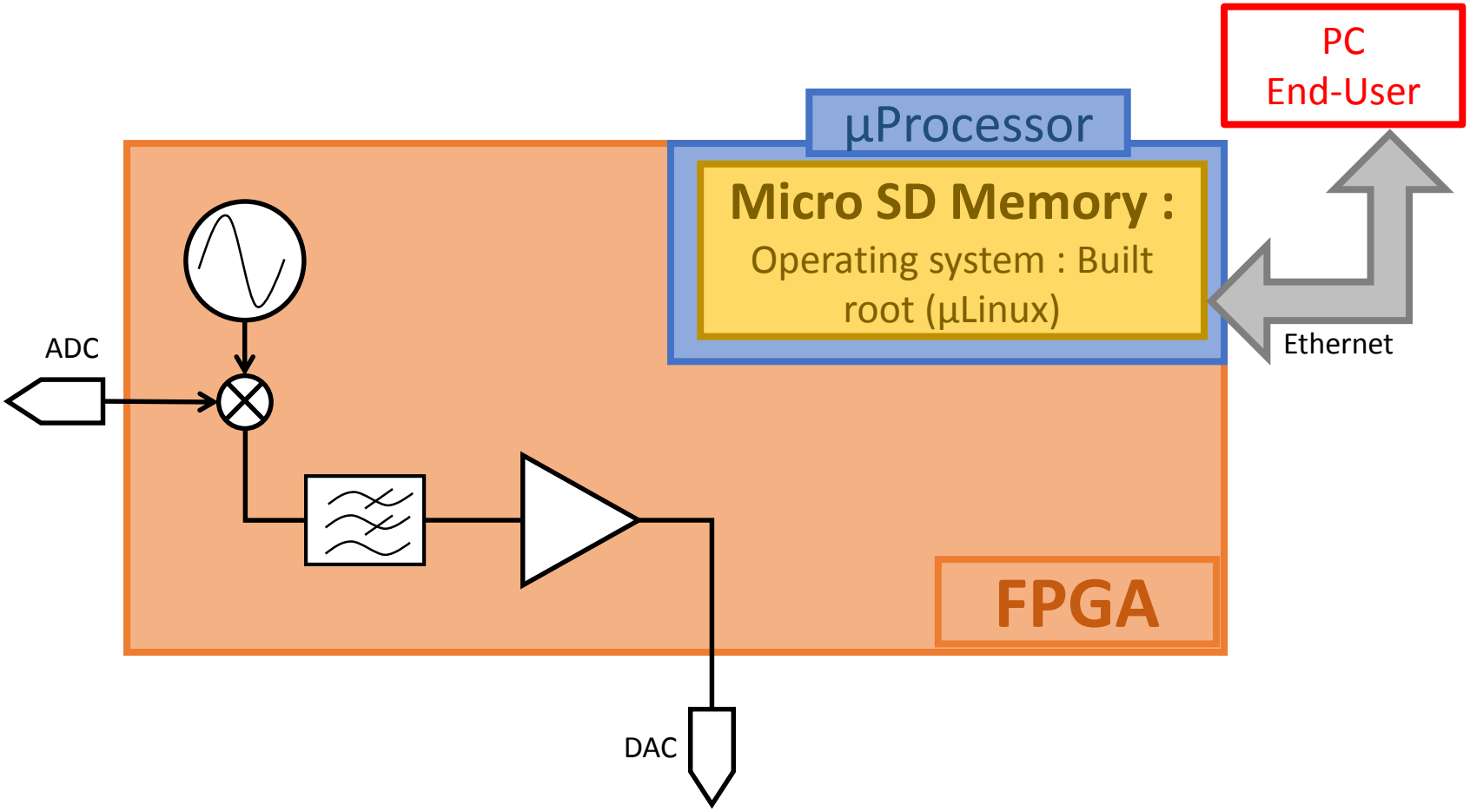
Step 2: Run an embedded OS



Step 3: Programming

No communication between functions and the End-user

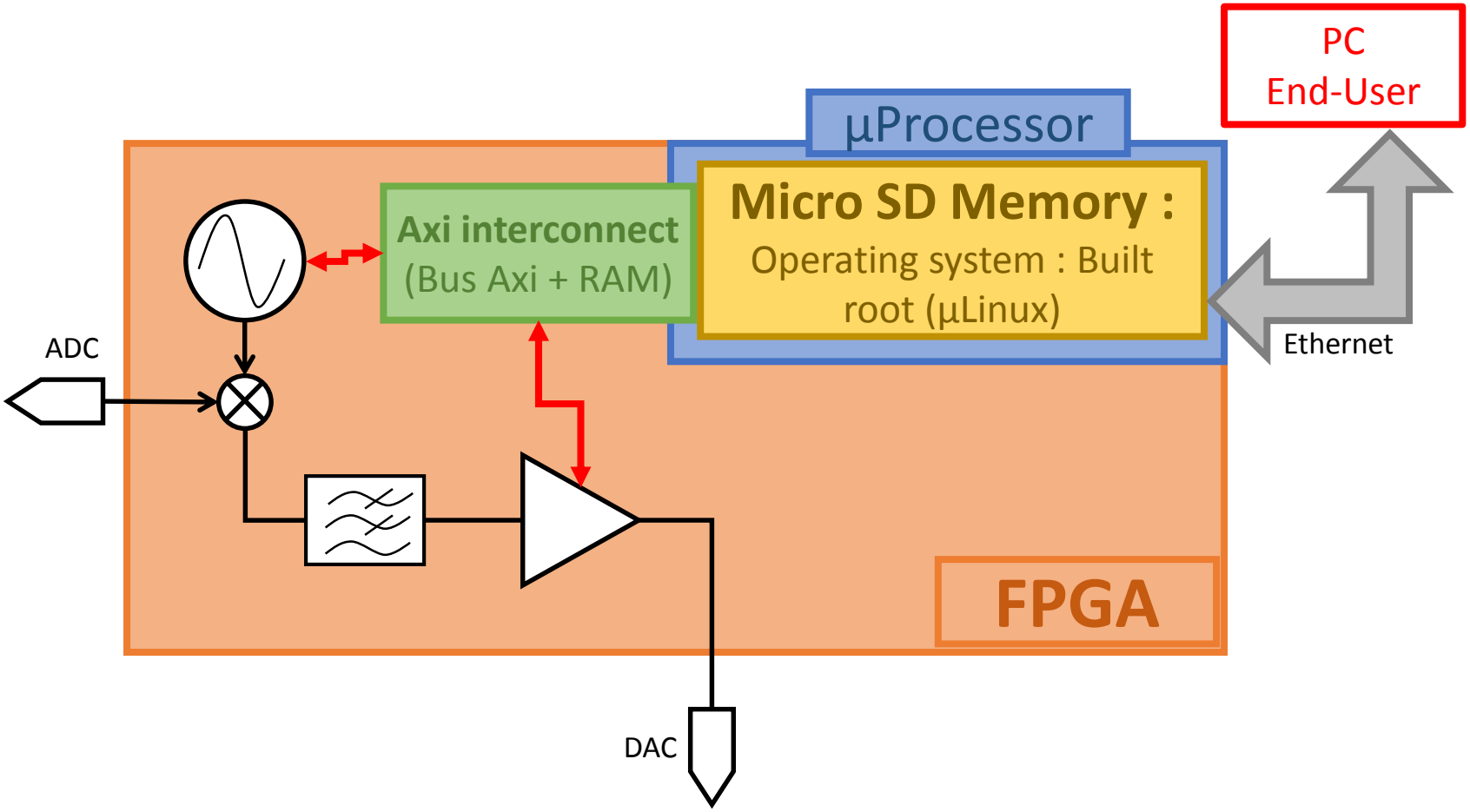
➔ Need to link µP and IP parameters



Step 4 - 1: Axi interconect

New IP exploiting data transfert bus and memory

➔ Define memory address



Step 4 - 2: Drivers

Development of Drivers:

- Define communication protocol/rules between l'IP and µlinux
- Need one for each IP
- C language
- Portability (independent of the FPGA type)

Generator (software):

- link driver and IPs address
- instantiate and mount IPs in µlinux
- CPP language
- main function and 8 files

```
int main(int argc, char **argv)
{
    bool legacy = false, use_dts = true;
    string xmlFile;
    string configName = string(getenv("HOME")) + "/.modulegenrc";

    ConfigHandler cfhandl(configName);

    if (argc < 2 || argc > 4) {
        printUsage(string(argv[0]));
        return EXIT_FAILURE;
    }

    if (argc == 2) {
        xmlFile = string(argv[1]);
    } else {
        xmlFile = string(argv[2]);
        if (!strcmp(argv[1], "-nodts")) {
            use_dts = false;
            if (!strcmp(argv[2], "-legacy")) {
                legacy = true;
                xmlFile = string(argv[3]);
            }
        } else if (!strcmp(argv[1], "-dts")) {
            use_dts = true;
        } else {
            printUsage(string(argv[0]));
            return EXIT_FAILURE;
        }
    }

    string fpga_driver_dir;
    try {
        fpga_driver_dir = string(getenv(DRIVER_DIR.c_str()));
    } catch (exception exec) {
        printError("Erreur: env var " + DRIVER_DIR + " not defined");
        return EXIT_FAILURE;
    }
    //cout << fpga_driver_dir+"/driver.xml" << endl;

    XmlWrapper xmlWrapper(xmlFile);
    //XmlWrapper *libWrapper = new XmlWrapper(fpga_driver_dir+"/driver.xml");
    XmlWrapper libWrapper(fpga_driver_dir+"/driver.xml");

    DTSGenerator dtsGen(xmlWrapper, libWrapper, false);//true);
    DriverGenerator drvGen(&xmlWrapper, &libWrapper, false);

    AppGenerator appGen(&xmlWrapper, &libWrapper, legacy);

    string rootName(xmlWrapper.getRoot()->Attribute("name"));
    string appDir(string(rootName).append("/app"));

    string moduleDir(string(rootName).append("/modules"));

    /* Makefile generation */
    if (use_dts == false) {
        if (-1 == mkpath(moduleDir.c_str(), S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH)
```

```
struct nco_counter_dev {
    char *name;         /* name of the instance */
    void *membase;      /* base address for instance  */
    struct resource *mem_res;
    struct device *dev;
    struct miscdevice misc;
    struct list_head list;
};

static LIST_HEAD(nco_counter_data_list);

static long nco_counter_ioctl(struct file *filp, unsigned int cmd,
              unsigned long arg)
{
    int retval = 0;
    struct nco_counter_dev *nco_counter;
    u64 val;
    u64 ioc;
    u64 t1, t2;
    printk("hello\n");

    /* guard against device removal before, or while,
     * we issue this ioctl.
     */
    nco_counter = filp->private_data;
    if (nco_counter == NULL)
        return -ENODATA;

    if (_IOC_NR(cmd) < 0)
        return -EINVAL;

    val = _IOC_NR(cmd);
    if (_IOC_DIR(cmd) & _IOC_READ) {
        ioc = 0;
        switch (val) {
        case REG_PINC:
            t1 = readfpga(nco_counter->membase + (REG_PINC_L << 2));
            t2 = readfpga(nco_counter->membase + (REG_PINC_H << 2));
            ioc = (t2 << 32) | (0xffffffff&t1);
            break;
        case REG_MAX_ACCUM:
            t1 = readfpga(nco_counter->membase + (REG_MAX_ACCUM_L << 2));
            t2 = readfpga(nco_counter->membase + (REG_MAX_ACCUM_H << 2));
            ioc = (t2 << 32) | (0xffffffff&t1);
            break;
        default:
            ioc = readfpga(nco_counter->membase + (val << 2));
        }
        printk("read %llu\n", ioc);
        if (put_user(ioc, (u64 __user*) arg))
            return -EACCES;
    }else {
        if (get_user(ioc, (u64 __user*) arg)) {
            printk("merdouille\n");
            return -EACCES;
        }

        printk("write %llu\n", ioc);
        switch (val) {
        case REG_PINC:
            writefpga(ioc & 0xffffffff, nco_counter->membase + (REG_PINC_L << 2));
            writefpga(ioc >> 32, nco_counter->membase + (REG_PINC_H << 2));
            break;
        case REG_MAX_ACCUM:
            writefpga(ioc & 0xffffffff, nco_counter->membase + (REG_MAX_ACCUM_L << 2));
            writefpga(ioc >> 32, nco_counter->membase + (REG_MAX_ACCUM_H << 2));
            break;
        default:
            writefpga(ioc, nco_counter->membase + (val << 2));
        }
    }
    printk("fin\n");
    return retval;
}

int nco_counter_open(struct inode *inode, struct file *filp)
{
    struct nco_counter_dev *pos, *data = NULL;
    /* Allocate and fill any data structure to be put in filp->private_data */
    list_for_each_entry(pos, &nco_counter_data_list, list) {
        if (pos->misc.minor == iminor(inode)) {
```

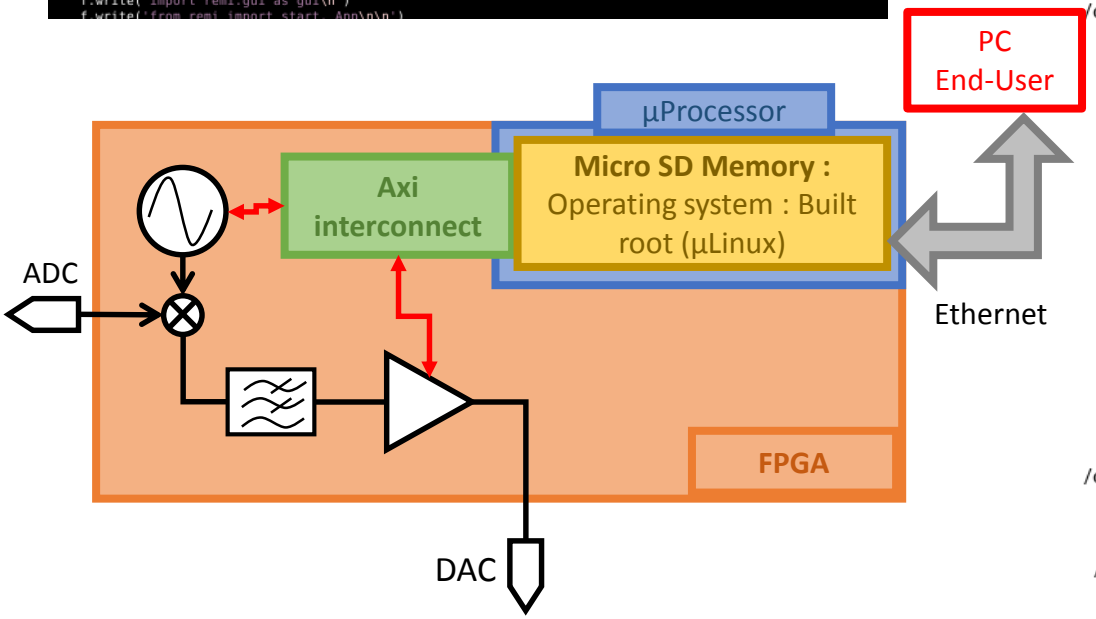
Step 4 - 3: Library

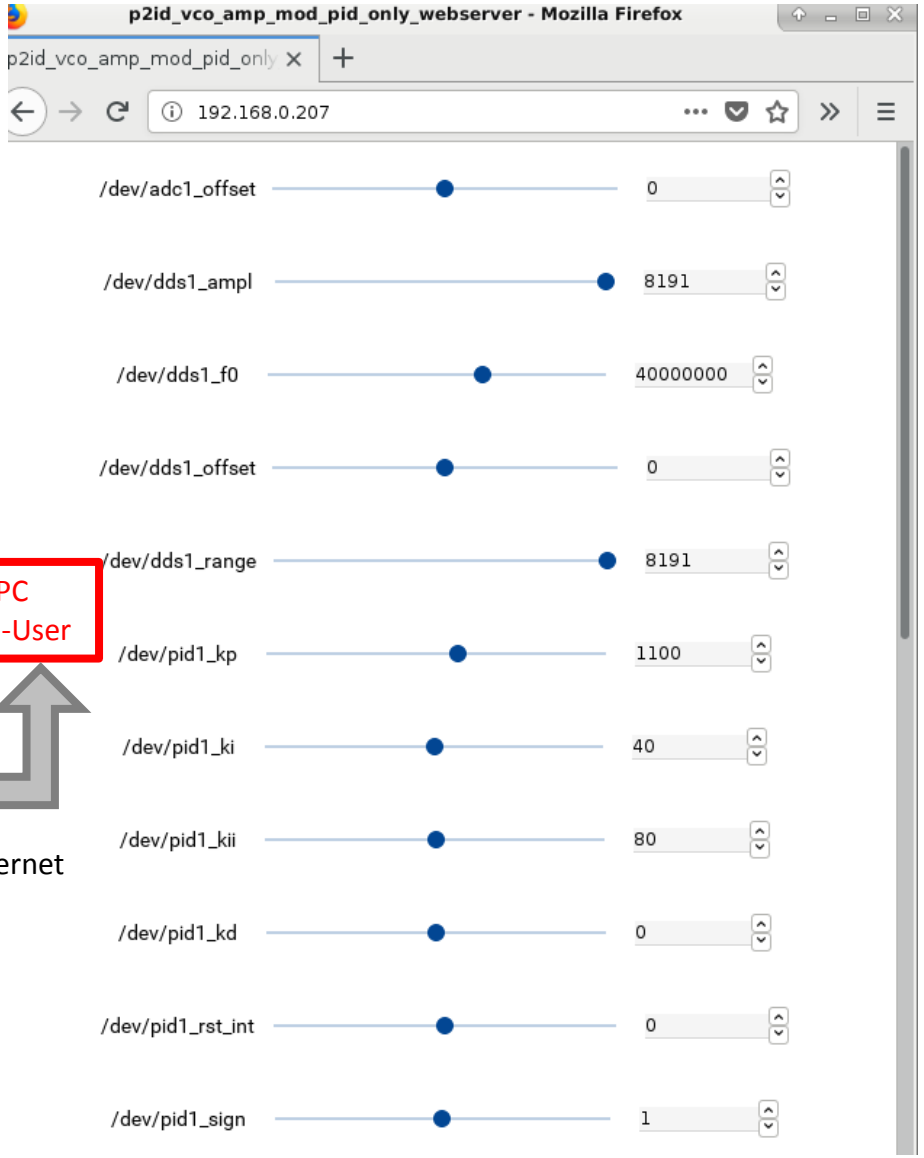
- User friendly purpose
- High level function (“independent” of hardware)
- C or python language
- Portability (independent of the FPGA type)

```c
/*
 * SPI testing utility (using spidev driver)
 *
 * Copyright (c) 2007  MontaVista Software, Inc.
 * Copyright (c) 2007  Anton Vorontsov <avorontsov@ru.mvista.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 *
 * Cross-compile with cross-gcc -I/path/to/cross-kernel/include
 */

#include <math.h>
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
/* memory management */
#include <sys/mman.h>
#include <nco_core/nco_config.h>
#include <nco_counter_core/nco_counter_config.h>

int nco_send_conf(const char *filename, const int period)
{
    int nco = open(filename, O_RDWR);
    if (nco < 0) {
        printf("erreur d'ouverture de %s\n", filename);
        return nco;
    }
    printf("configuration nco\n");
    ioctl(nco, NCO_SET_PERIOD, &period);
    close(nco);
    return 0;

}

int nco_counter_send_conf(const char *filename,
        const int freqHz_ref, const double freqHz_out,
        const int accum_size,
        const int offset,
        const char pinc_sw,
        const char poff_sw)
{
    long double step = (long double)freqHz_ref/powl(2,accum_size);
    long double incrf = ((long double)freqHz_out)/step;
    //unsigned int incr = (unsigned int)incrf;
    uint64_t incr = (uint64_t)round(incrf);
    printf("step %Lf incr %lld freqHz_out %lf incrf %Lf\n", step, incr, freqHz_out,
        incrf);
    long double real_freq = freqHz_ref/(powl(2,accum_size)/(long double)incr);
    printf("%Lf\n", real_freq);

    unsigned int ctrl_reg = 0;
    if (pinc_sw == 1) {
        ctrl_reg = ctrl_reg | CTRL_PINC_SW;
    }
    if (poff_sw == 1) {
        ctrl_reg = ctrl_reg | CTRL_POFF_SW;
    }

    int nco = open(filename, O_RDWR);
    if (nco < 0) {
        printf("erreur d'ouverture de %s\n", filename);
        return nco;
    }
    ioctl(nco, NCO_COUNTER_SET(REG_CTRL), &ctrl_reg);
    ioctl(nco, NCO_COUNTER_SET(REG_PINC), &incr);
    ioctl(nco, NCO_COUNTER_SET(REG_POFF), &offset);
    close(nco);
    return 0;
}
```

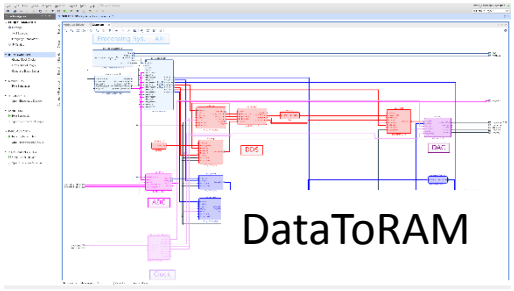
Step 4 - 4: WebServer

```
#!/usr/bin/env python

from xml.dom import minidom
import sys
import os

board_driver_array = []
xmldoc = minidom.parse(sys.argv[1])
name = sys.argv[1].split('.')[0]

driver_list = xmldoc.getElementsByTagName('driver')

for driver in driver_list:
    board_driver_list = driver.getElementsByTagName('board_driver')
    for board_driver in board_driver_list:
        board_driver_array.append([str(driver.attributes['name'].value), str(board_driver.attributes['name']

try:
    os.chdir('%s/app'%name)
except:
    os.chdir(name)
    os.mkdir('app')
    os.chdir('app')

try:
    os.remove('%s_webserver.py'%name)
except:
    pass

with open('%s_webserver.py'%name, 'a') as f:
    f.write('#!/usr/bin/env python\n\n')
    f.write('import liboscimp_fpga\n')
    f.write('import ctypes\n')
    f.write('import remi.gui as gui\n')
    f.write('from remi import start, App\n\n')
```





Step 4 - 5: Monitoring

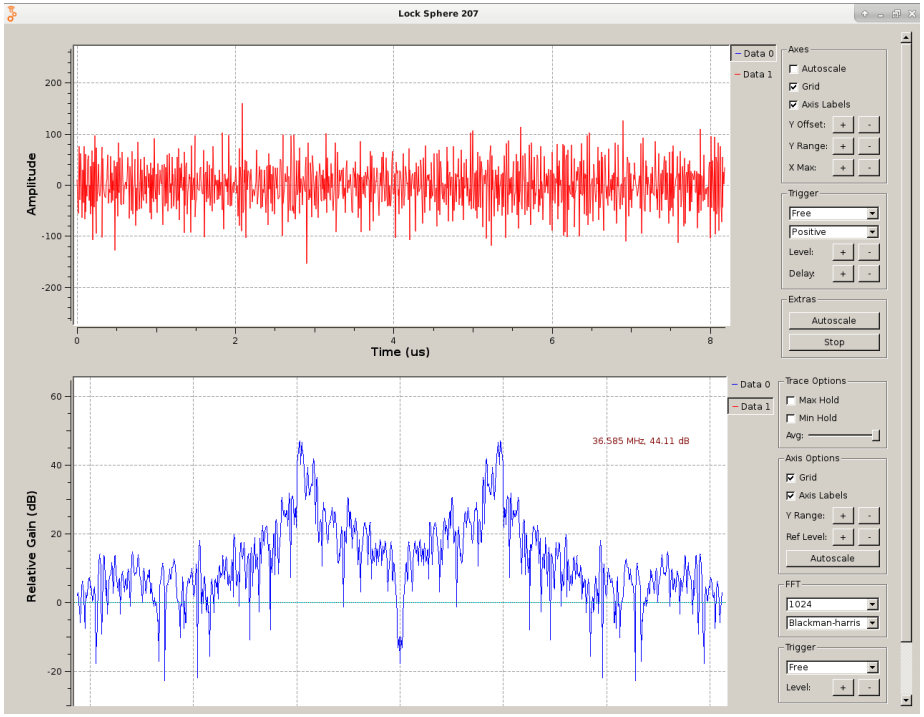


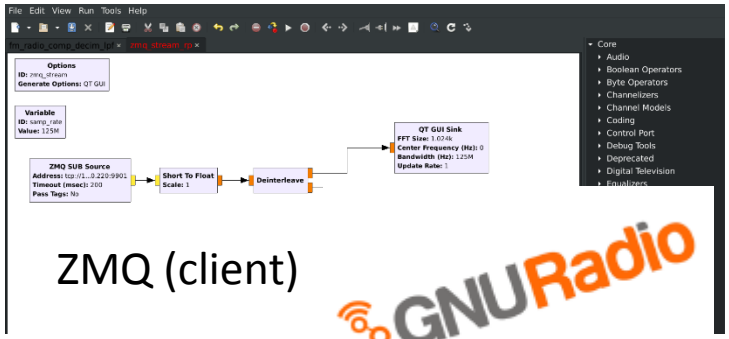
DataToRAM

µLinux

ZMQ software
Data exchange (server)
30 ko/s

PC / End-user



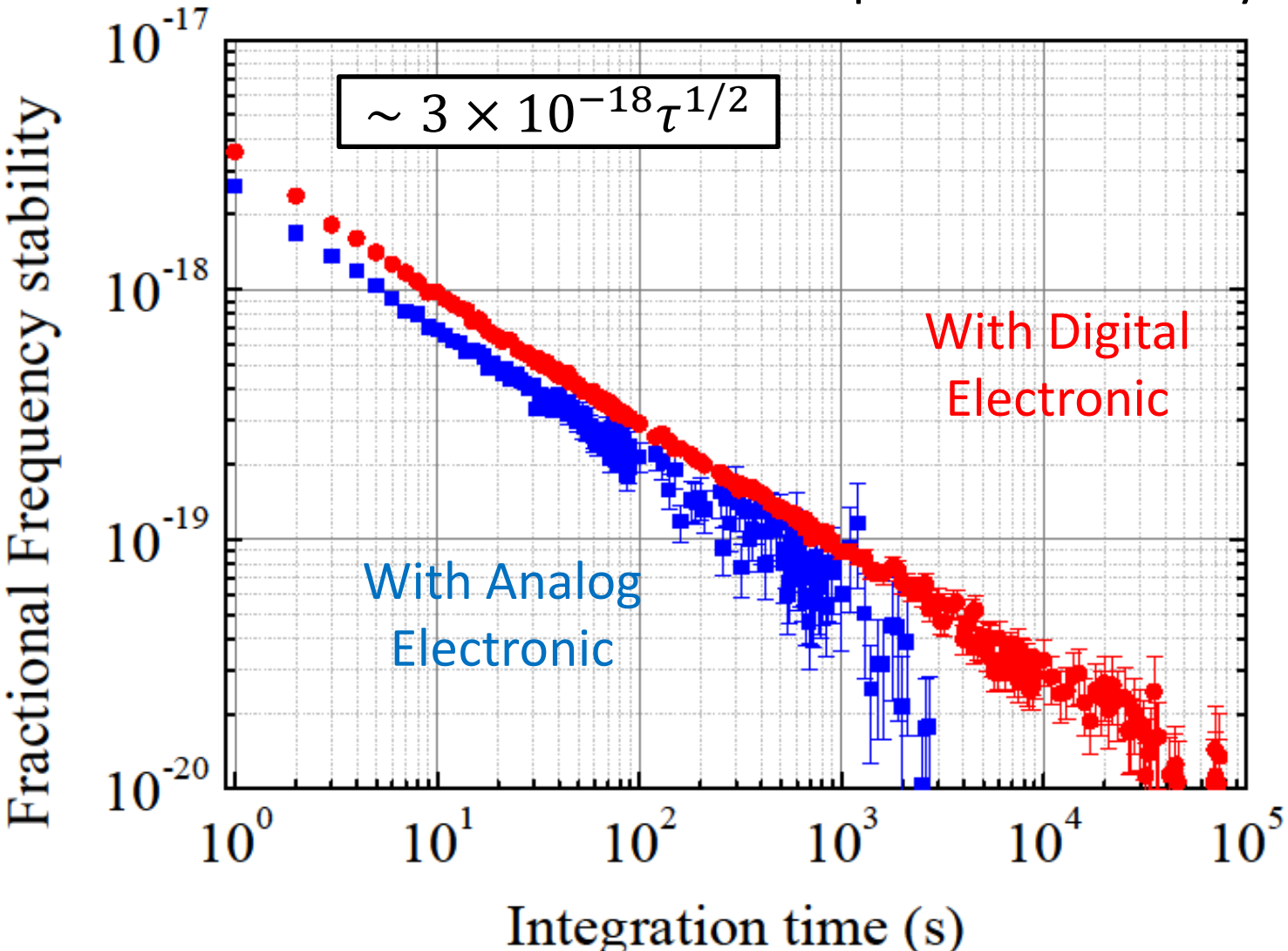


ZMQ (client)

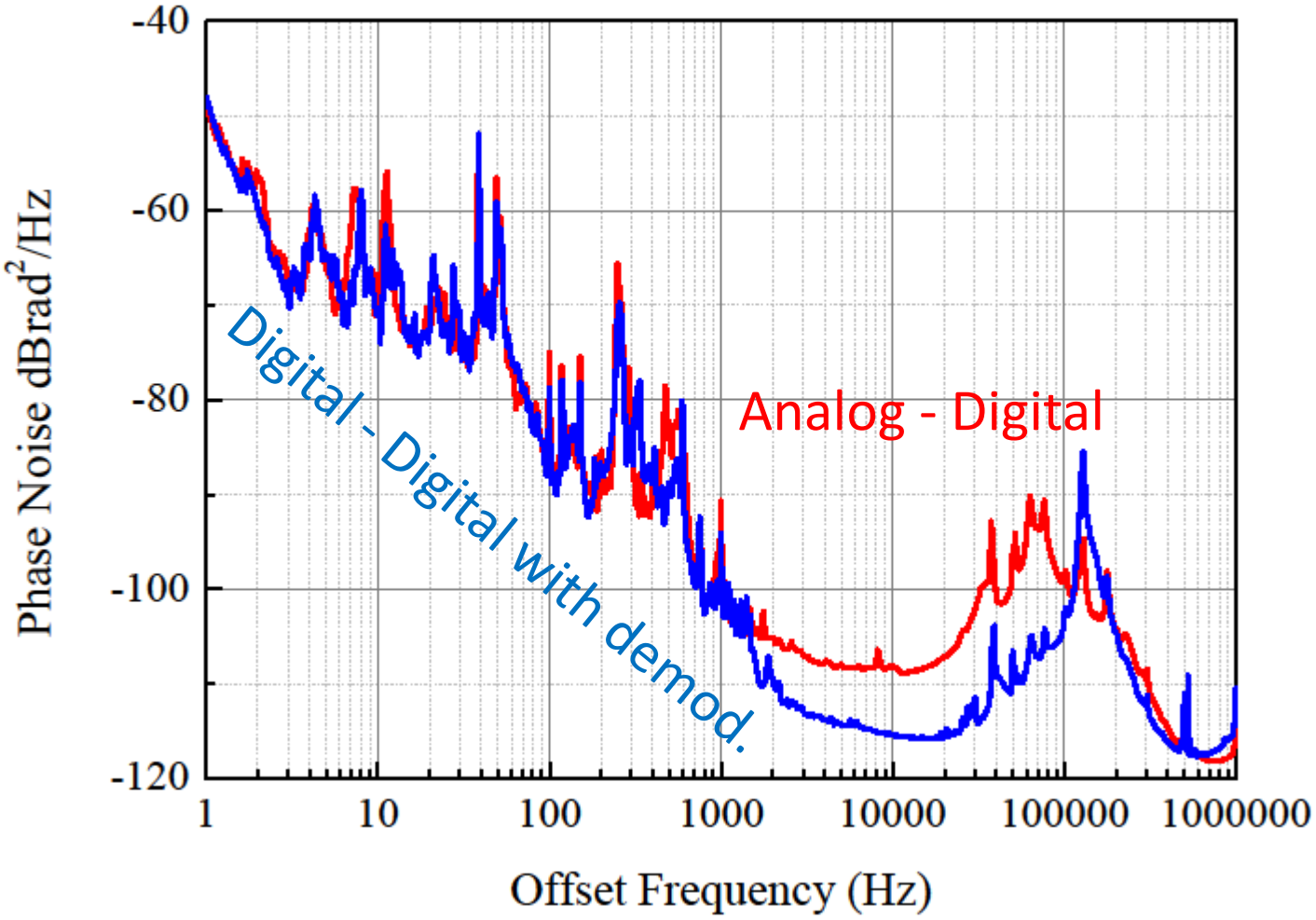
GNURadio

Test: Laser Phase Lock

Loop Filter Noise only



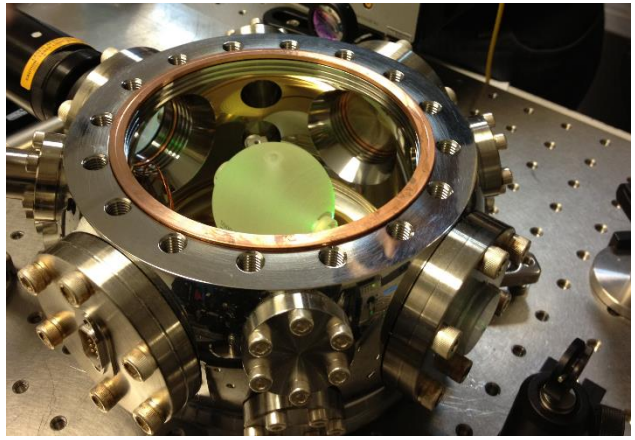
Test: Link Phase Lock



Outlook: Refimeve+

Digital Electronic implemented in :

- Diode laser phase lock to center channel IUT44 (1542.14 nm)
- Laser Phase Lock to H-Maser signal
- Noise cancelled Link
- Pound Drever Hall
- Plan a comparison in Strasbourg:
  SYRTE - PTB - FEMTO-ST

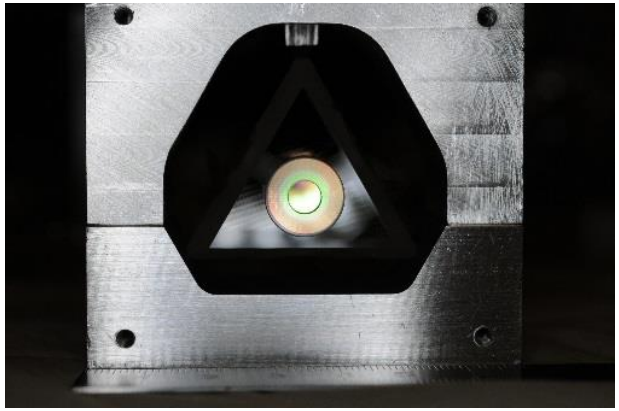


➔ Help to self diagnostic and autocontrol: reliability

Outlook: Cavities

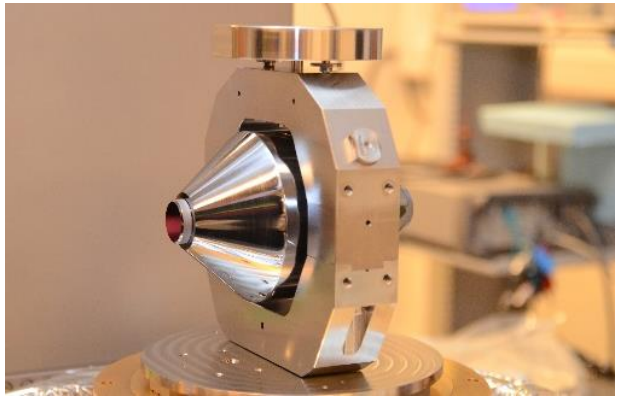
Compact cavity:

- 10^{-15} at 1 s range in 30 L
- Find answer about CTE and vib.
- Integrate digital elec (PDH …)



Silicon cavity:

- Thermal design validated
- RAM control in progress
- Vibration sensitivity optimization
- …



Outlook:

Yb+ compact optical clock:

- Optical synthesis from reference laser (1.542µm) to clock transition (435nm)
- R+ use for absolute frequency measurement