PAUL SCHERRER INSTITUT

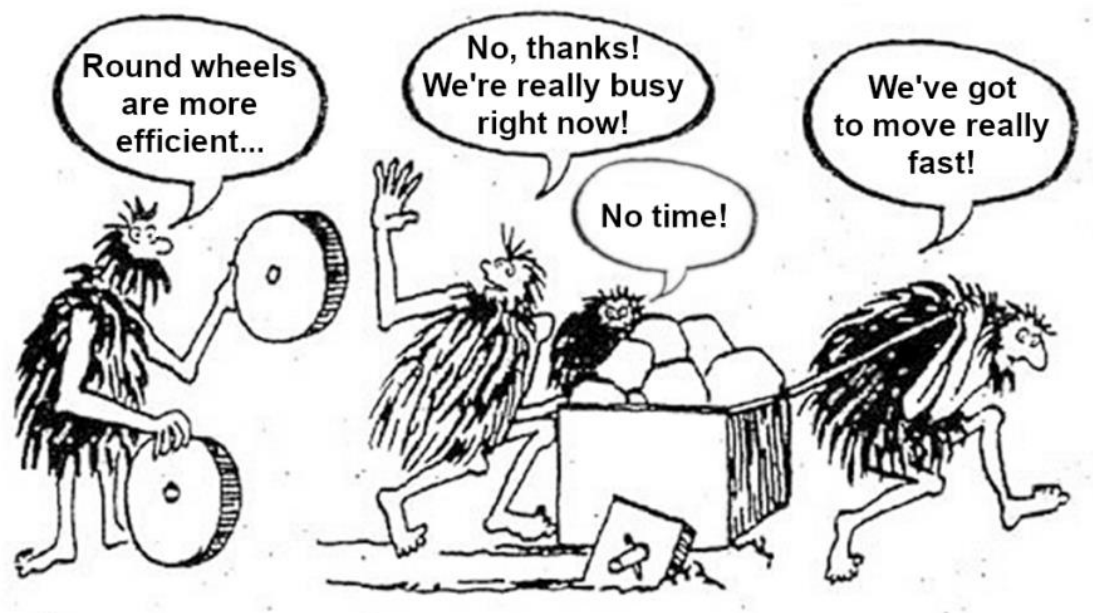
Benoît Stef:: FPGA Engineer :: Paul Scherrer Institute

Signal Processing Development Methodologies for FPGA Platforms using Reusable and Generic PSI Library components

12/10/2022 - LLRF workshop 2022 – Brugg-Windisch

Agenda

1. Motivation to create open-source library
2. What can you find in the library and where ?
   I. PSI Common
   II. PSI FIX
   III. Workflow
3. How do we handle our project repo
4. LLRF FPGA Firmware architecture overview
5. Reality of open-sourcing



Motivation to go open source

- Create a **community** for standard VHDL blocks that do not depends on a unique technology

- Matrix Reorganization – competences based. **Quality + Synergy**
- Reusing by thinking **Generics** and get users to verify, exchange and contribute to the library
- Transfer our knowledge to whoever needs it to spend time **on what matters**: **Concept and architecture**
- Create stronger **link with industry** and allow the community to **benefit** from our **expertise**

Courtesy of O. Bruendler former colleague and now working as FPGA/SoC System Expert at Enclustra GmbH

What can you find in the library and where?

https://github.com/paulscherrerinstitute/psi_fpga_all
https://github.com/paulscherrerinstitute/psi_common
https://github.com/paulscherrerinstitute/psi_fix

- Python
- TCL
- VHDL
- Vivadolp
- scripts

submodule

- en_cl_fix @ 11c11ae
- psi_common @ 7074a5d
- psi_fix @ 8a9000b
- psi_multi_stream_daq @ 28b79e9
- psi_tb @ cc9477b

- vivadoIP_axi_mm_reader @ 09af4f1
- vivadoIP_axis_data_gen @ 41a1d11
- vivadoIP_clock_measure @ af531eb
- vivadoIP_data_rec @ da03917
- vivadoIP_fpga_base @ b4317c0
- vivadoIP_i2c_devreg @ 437514f
- vivadoIP_mem_test @ feb2274
- vivadoIP_power_sink @ 0198c69
- vivadoIP_psi_ms_daq @ 6cf9ae3
- vivadoIP_spi_simple @ b750a12
- vivadoIP_sync_edge_det @ bb0635e

Content

- psi_common_arb_priority.vhd
- psi_common_arb_round_robin.vhd
- psi_common_array_pkg.vhd
- psi_common_async_fifo.vhd
- psi_common_axi_master_full.vhd
- psi_common_axi_master_simple.vhd
- psi_common_axi_multi_pl_stage.vhd
- psi_common_axi_pkg.vhd

- psi_fix_bin_div.vhd
- psi_fix_cic_dec_cfg_1ch.vhd
- psi_fix_cic_dec_cfg_nch_par_tdm.vhd
- psi_fix_cic_dec_cfg_nch_tdm_tdm.vhd
- psi_fix_cic_dec_fix_1ch.vhd
- psi_fix_cic_dec_fix_nch_par_tdm.vhd
- psi_fix_cic_dec_fix_nch_tdm_tdm.vhd
- psi_fix_cic_int_fix_1ch.vhd
- psi_fix_comparator.vhd
- psi_fix_complex_abs.vhd

Comprehensive documentation & index

PSI Common

About **50 components** and more to come

Memory components

Component	Source	Description
Simple dual port RAM	psi_common_sdp_ram.vhd	link
Simple dual port RAM with byte enable	psi_common_sp_ram_be.vhd	link
True Dual port RAM	psi_common_tdp_ram.vhd	link
True dual port RAM with byte enable	psi_common_tdp_ram_be.vhd	link

FIFO components

Component	Source	Description
Asynchronous FIFO	psi_common_async_fifo.vhd	link
Synchronous FIFO	psi_common_sync_fifo.vhd	link

Clock domain crossing (CDC) components

Component
Pulse clock crossing (asynchronous pulse/vld transfer)
Simple clock crossing (asynchronous data value transfer)
Status clock crossing (asynchronous slow changing value transfer)
Synchronous CDC with AXI-S handshaking from **Lower** clock to **Higher** multiple integer clock frequency
Synchronous CDC with AXI-S handshaking from **Higher** clock to **lower** multiple integer clock frequency
Bit CDC

Other components that can be used as cdc

- psi_common_tdp_ram
- psi_common_async_fifo

Time Division Multiplexing (TDM) data Handling components

Component	Sou
TDM data to parallel	psi_common_td
Parallel to TDM data	psi_common_pa
TDM data to Parallel with configurable valid output channel number	psi_common_td
TDM data multiplexer	psi_common_td
Parallel to TDM with configurable valid output output channel	psi_common_pa
TDM data to parallel with last support and completion	psi_common_td

Arbiters components

Component	Source	Description
Priority	psi_common_arb_priority.vhd	link
Round robin	psi_common_arb_round_robin.vhd	link

Interfaces components

Package	Source	Description
SPI master	psi_common_spi_master.vhd	link
SPI master configurable width	psi_common_spi_master_cfg.vhd	link
I2C master	psi_common_i2c_master.vhd	link
AXI master Simple	psi_common_axi_master_simple.vhd	link
AXI master Full	psi_common_axi_master_full.vhd	link
AXI slave IP (32 bits)	psi_common_axi_slave_ipif.vhd	link
AXI slave IP (64 bits)	psi_common_axi_slave_ipif64.vhd	N.A
AXI multi pipeline stage	psi_common_axi_multi_pl_stage.vhd	N.A
AXI slave Lite IP	psi_common_axilite_slave_ipif.vhd	link

miscellaneous components

Component
Delay settable via generics
Pipeline stage
Multi pipeline stage
Sizable Ping pong buffer // & tdm (interface to stream continuously data into DPRAM)
Delay settable via register
Generic Watchdog
Don't optimize (Xilinx) allows evaluating synthesis
Generic Debouncer
Analog Trigger Generator
Digital Trigger Generator
Dynamic Shifter
Pulse/Ramp generator
Pulse generator ctrl static
Parallel to serial
Serial to parallel
Find Min Max
Min Max Sum
PRBS

Packages

Package	Source	Description
Math	psi_common_math_pkg.vhd	link
array	psi_common_array_pkg.vhd	link
logic	psi_common_logic_pkg.vhd	link

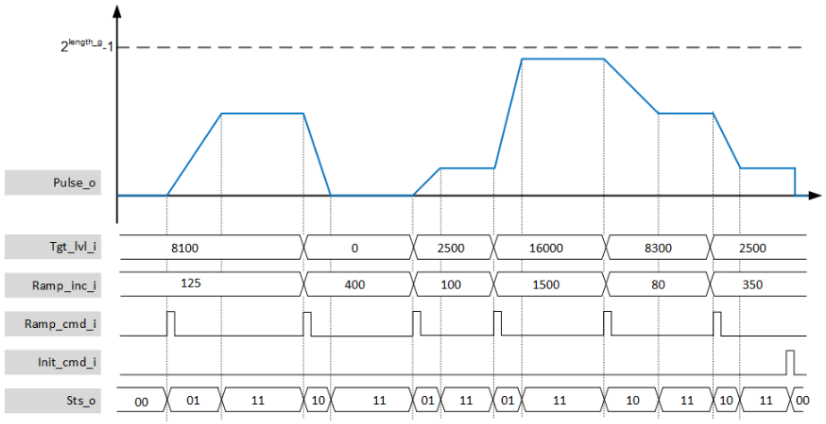
Comprehensive documentation

PSI Common

- VHDL source: psi_common_ramp_gene.vhd
- Testbench: psi_common_ramp_gene_tb.vhd

Description

This component implements a ramp generator where the user can set the target level to reach and the step number prior to reach this level. It can from a certain value either continue ramping up either ramping down as the figure below shows. The direction is selected with the next value to reach, if current value is higher than previous one then it ramps up and opposite. Orignally the block handled only unsigned value and now can asapt to signed via generics.



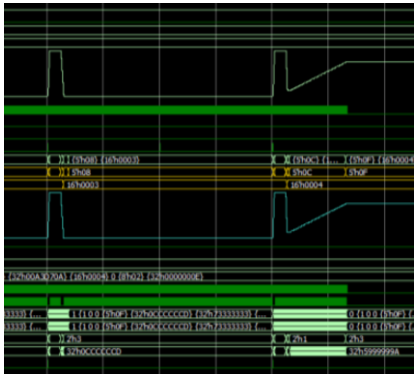
Generics

Generics	Description
rst_pol_g	reset polarity ('1' or '0')
width_g	Width of the data in bits
is_sign_g	sign = True / unsign = False
init_val_g	init value integer

Interfaces

Signal	Direction	Width	Description

```
      end if;
    end if;
  end process;

  --*** automatic check process 2 ***
  process(clk_i)
  begin
    if falling_edge(clk_i) then
      sOutPuls_dff <= sOutPuls;
      if RampCmd = '1' and sOutSts = "11" then
        print("[info]: new ramp command sent at" & to_string(now, ns));
      else
        if sOutStr = '1' then
          if sOutSts = "01" then
            assert (signed(sOutPuls_dff) < signed(sOutPuls)) report "###ERROR### info: ramp is not increasing" severity error;
          elsif sOutSts = "11" then
            if RampCmd = '0' then
              assert TgtLevel2 = sOutPuls report "###ERROR### info: error arrival data, expected " &
                                    to_string(to_integer(signed(TgtLevel2)))&
                                    ", got " & to_string(to_integer(signed(sOutPuls))) severity error;
            end if;
          elsif sOutSts = "10" then
            assert (signed(sOutPuls_dff) > signed(sOutPuls)) report "###ERROR### info: ramp is not decreasing" severity error;
          end if;
        end if;
      end if;
    end if;
  end process;

  --*** clock process ***
  proc_ck : process
    variable count : integer := 0;
  begin
    while tb_run loop
      count := 0;
      while count /= 10 loop
```



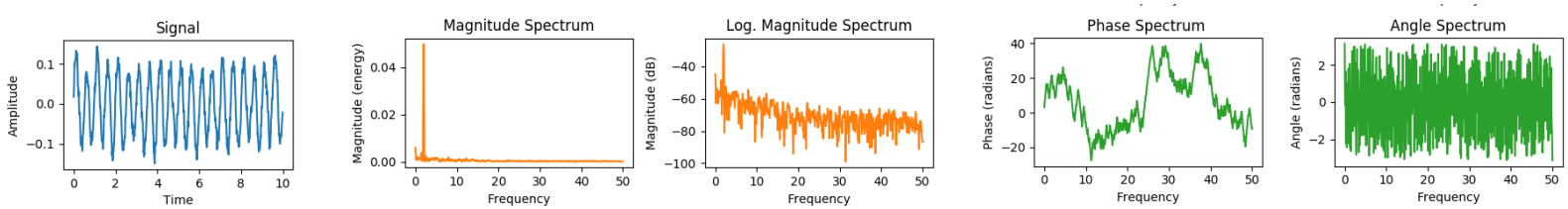
PSI FIX Library

A remedy for fixed-point design related headaches

- A VHDL package (based on en_cl_fix from Enclustra GmbH)
  - Handles format conversions including rounding and saturation
  - Functions for all common arithemetic operations (Add, Mult, Resize, etc.)

- A VHDL Library
  - Contains commonly used entities (FIR,CIC, Approx. Function, etc.)
  - Strongly parametrizable (also number formats)

- A Python library
  - Contains Python bit-true models of all functions in the VHDL package
  - Contains bit-true models of all entities that exist in VHDL
  - Makes the conversion between VHDL and bittrue Python model easy
  - Based on the well known NumPy package for fast processing of arrays



Comprehensive documentation

PSI FIX – 34 components

PAUL SCHERRER INSTITUT

3.4.4 Architecture

For details on the filter mathematics, refer to 3.2.4. This section only describes how the multi channel filter is implemented.

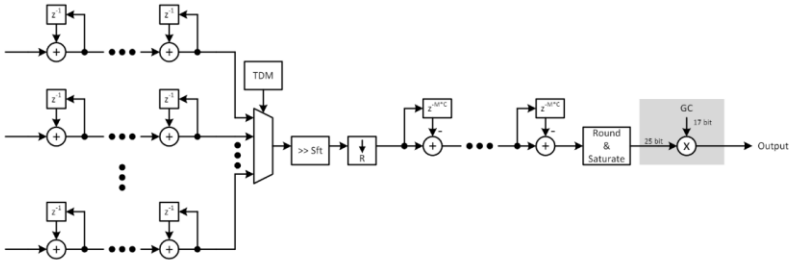


Figure 9: psi_fix_cic_dec_fix_nch_par_tdm Architecture

3.9.4 Architecture

The figure below shows a base structure that is used in all filter configurations. The structure consists of three binary shifts (acting as a multiplication by constant coefficient) followed by two adders.

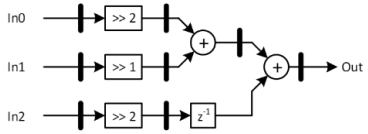
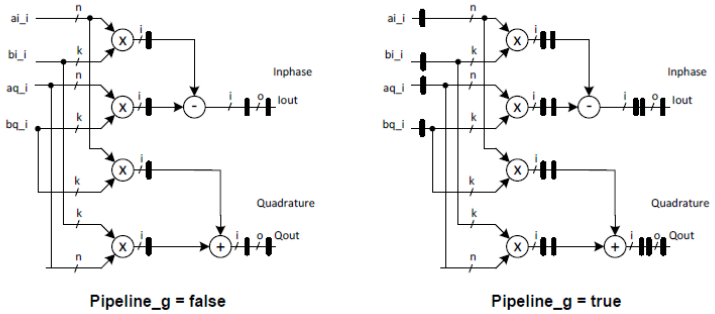
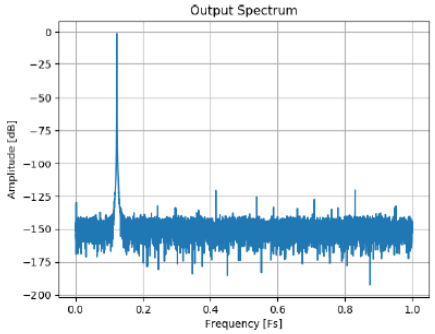


Figure 12: psi_fix_fir_3tap_hbw_dec base structure

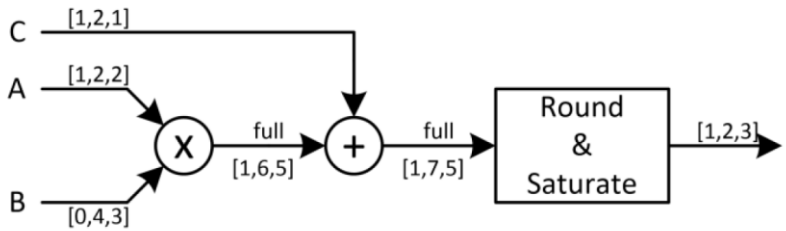
3.19.4 Architecture





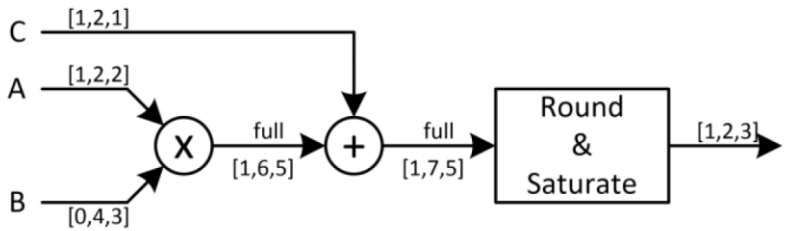
IQ modulator & demodulator, DDS, CORDIC, CIC deimation and interpolation, FIR, IIR, Linear approximation, SQRT, Complex arithmetic, moving average, binary divider, etc.

VHDL example of traditional multiplier adder



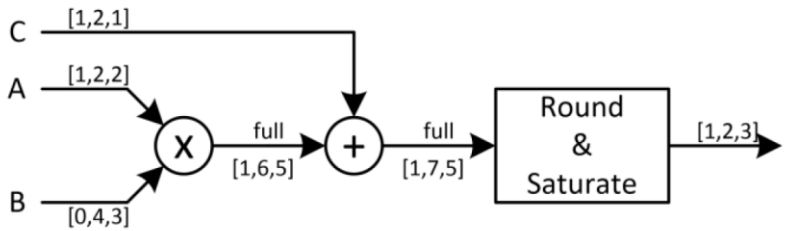

```
signal a : std_logic_vector(4 downto 0);          --Format : (1,2,2)
signal b : std_logic_vector(6 downto 0);          --Format : (0,4,3)
signal c : std_logic_vector(3 downto 0);          --Format : (1,2,1)

signal m        : std_logic_vector(12 downto 0);  --Format : (1,6,5) --> unusual
signal sum      : std_logic_vector(12 downto 0);  --Format : (1,7,5) --> unusual
signal rnd_add  : std_logic_vector(12 downto 0);  --Format : (1,7,5) --> unusual
signal rnd      : std_logic_vector(10 downto 0);  --Format : (1,7,3)
signal sat      : std_logic_vector( 5 downto 0);  --Format : (1,2,3)

m       <= std_logic_vector(signed(a)*signed('0'& b));
sum     <= std_logic_vector(signed(m) + signed(c&"0000"));      -- unusual
rnd_add <= std_logic_vector(signed(sum) + 4);                   -- why 4?
rnd     <= rnd_add(12 downto 2);                                -- new range?

if signed(rnd) > 31 then                                        -- why 31?
  sat <= "0111111";                                             -- upper value
else
  sat <= rnd(5 downto 0);                                       --new range
end if;
```

Same design using the library




```
constant a_fmt : PsiFixFmt_t := (1,2,2);
constant b_fmt : PsiFixFmt_t := (0,4,3);
constant c_fmt : PsiFixFmt_t := (1,2,1);
constant s_fmt : PsiFixFmt_t := (1,2,3);

--Automatic scaling of internal formats
constant m_fmt : PsiFixFmt_t := (max(a_fmt.S, b_fmt.S), a_fmt.I+b_fmt.I, a_fmt.F+b_fmt.F);

--Automatic scaling of internal formats
signal a   : std_logic_vector(PsiFixSize(a_fmt)-1 downto 0);   --Format : (1,2,2)
signal b   : std_logic_vector(PsiFixSize(b_fmt)-1 downto 0);   --Format : (0,4,3)
signal c   : std_logic_vector(PsiFixSize(c_fmt)-1 downto 0);   --Format : (1,2,1)
signal m   : std_logic_vector(PsiFixSize(m_fmt)-1 downto 0);   --Format : (1,6,5)
signal sat: std_logic_vector(PsiFixSize(s_fmt)-1 downto 0);   --Format : (1,2,3)

-- short wand readable functional code
m       <= PsiFixMult(a, a_fmt, b, b_fmt, m_fmt);
sat     <= PsiFixAdd(m, m_fmt, c, c_fmt, s_fmt, PsiFixRound, PsiFixSat);
```

Python model of the same design




```python
from psi_fix_pkg import *
import numpy as np

a_fmt = PsiFixFmt(1,2,2)
b_fmt = PsiFixFmt(0,4,3)
c_fmt = PsiFixFmt(1,2,1)
s_fmt = PsiFixFmt(1,2,3)

#Automatic scaling of internal formats
m_fmt = PsiFixFmt(max(a_ftm.S, b_fmt.S), a_fmt.I+b_fmt, a_fmt.F+b_fmt.F)

def Model(a : np.ndarray, b : np.ndarray, c: np.ndarray) -> np.array:
    m   <= PsiFixMult(a, a_fmt, b, b_fmt, m_fmt);
    sat <= PsiFixAdd(m, m_fmt, c, c_fmt, s_fmt, PsiFixRound, PsiFixSat);
    return s
```

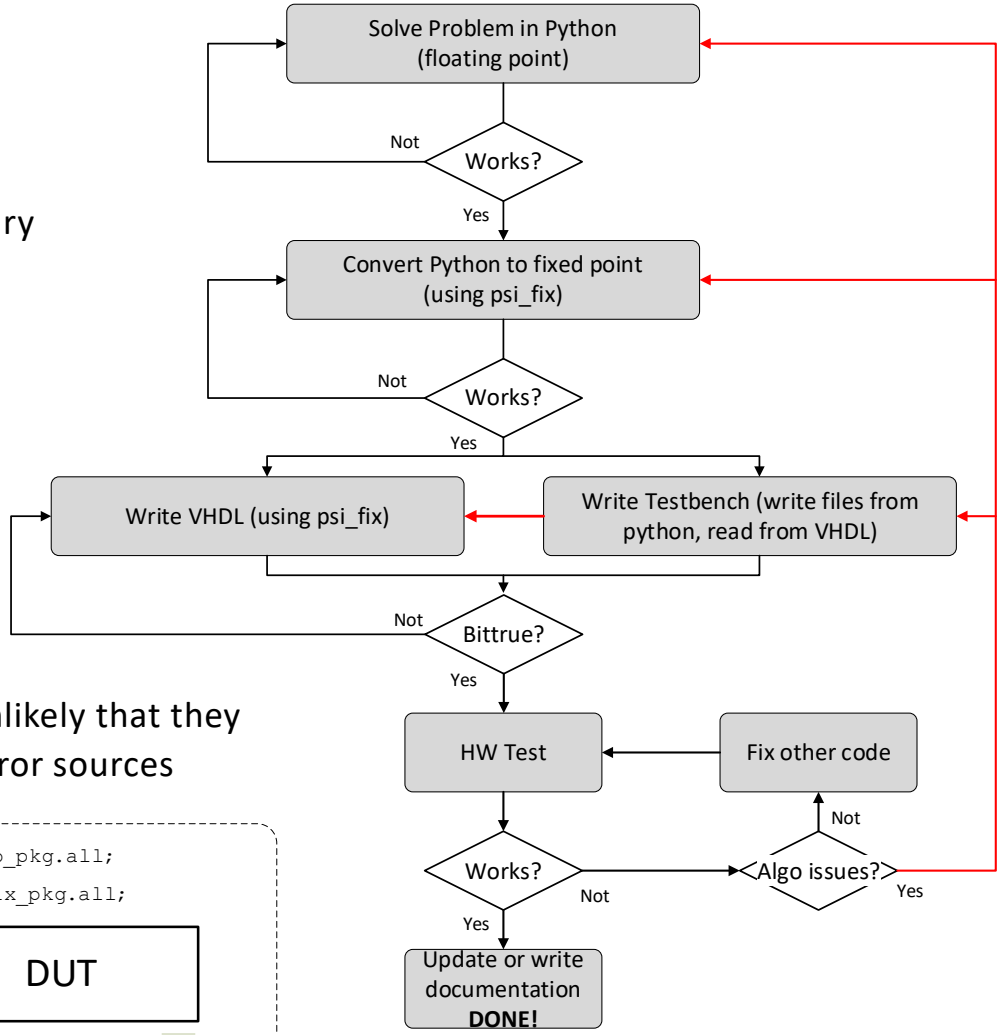
Workflow

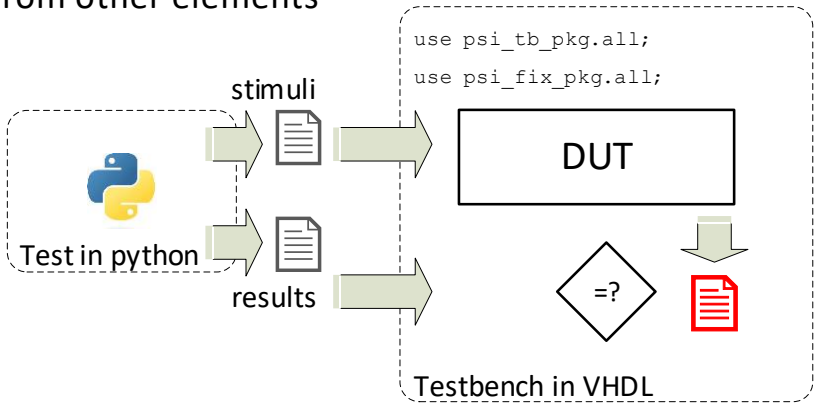
- Concept in python using floating point and SciPy

- Convert to fixed point using the library to check if saturation/rouding do not affect behaviour

- Testbench, testbench, testbench Python script generates stimuli and write to text files that are compared VHDL = PYTHON?

- Timing closure if bugs occurs very unlikely that they are produced by work in progress but error sources come from other elements

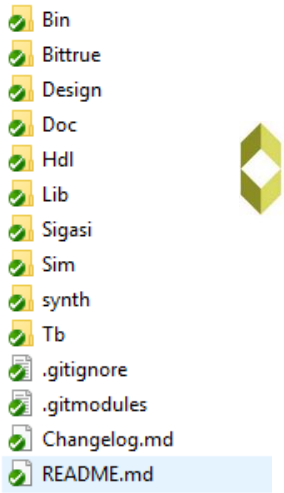




How do we handle our FW project repo?

ISE Project GIT folder structure

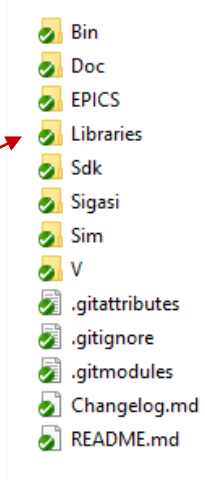
- **Bin**: Unique Binary file (Release)
- **BitTrue**: Python model and stimuli generation
- **Design**: Python simulation for analysis
- **Doc**: MD files
- **HDL**: project specfic design files
- **Lib**: Submodule PSI_FIX, PSI_COMMON, others
- **Sigasi**: VHDL Project
- **Sim**: modelsim & tcl script
- **Synth**: Xilinx project
- **Tb**: Unitary Test (Generic) & System Test



Vivado Project GIT folder structure

- **EPICS**: UI & templates
- **Sdk**: Xilinx Vivado project.tcl or ISE build script
- **V**: IP repository, hdl, contraints & project tcl

No **nested submodule** here **we rely on relative path**
This to avoid mess with repository version



Internal GitLab Libraries structure & scripts

GitLab

Libraries
Group ID: 682

Libraries not specific to any domain

Subgroups and projects | Shared projects | Archived projects

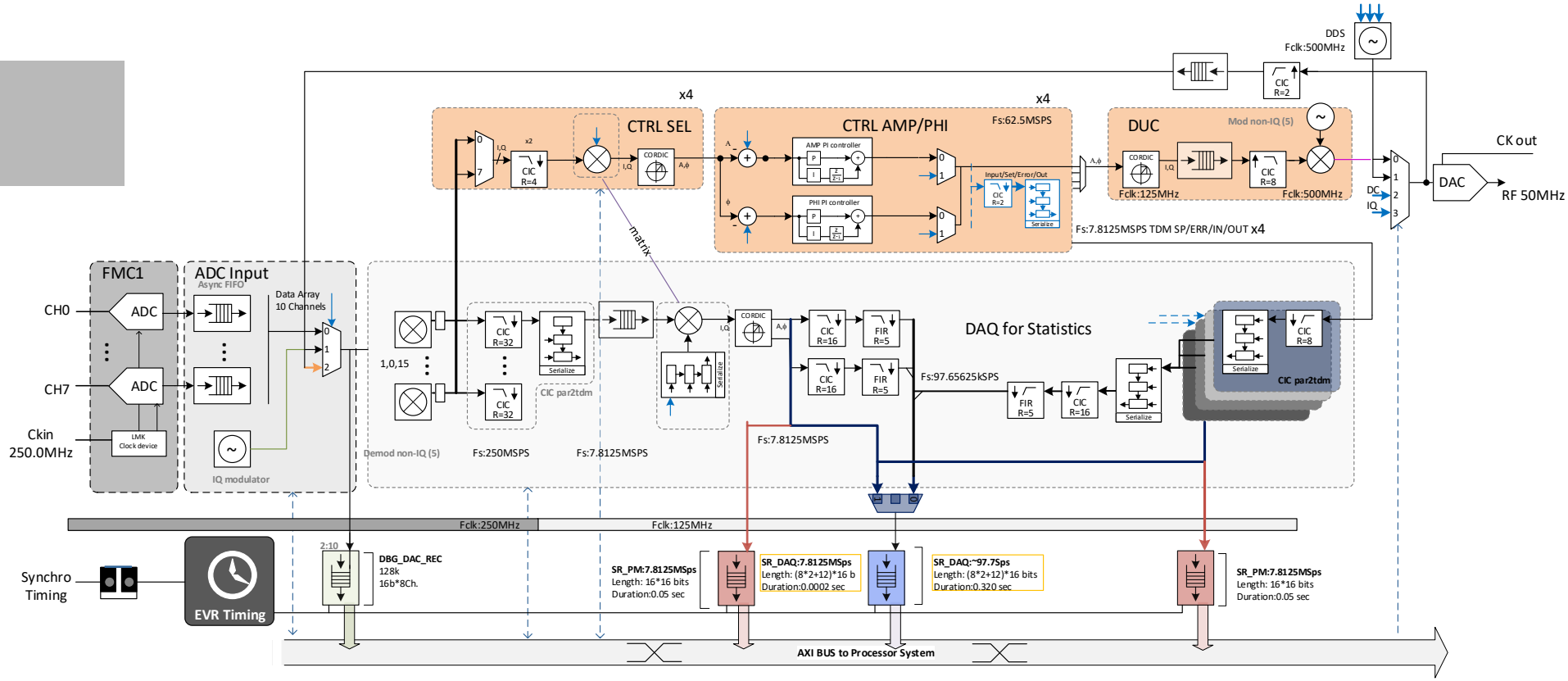
- Sw
- CtrlSys — Control System Related Libraries
- EbSw — Embedded software
- BoardSupport (Owner) — Any files or repositories that help with the usage of specific hardware
- Firmware (Owner) — FPGA Firmware libraries that are not domain specific
  - Edklp — Collection of pcores for Xilinx Embedded Development Kit (EDK). No general VHDL (that belongs to "VH... — 0 21 0
  - Python — 0 8 0
  - Vivadolp — Only put reusable Vivado IPI IP-Cores here. No general VHDL (that belongs to "VHDL") and not project-... — 0 31 0
  - VHDL — VHDL Libraries. Note that library elements must have self-checking testbenches and documentation. Ot... — 0 14 0
  - TCL — FPGA Firmware related TCL libraries — 0 2 0

```
create_tb_run "modified_cic_dec_fix_nch_par_tdm_tb"
tb_run_add_pre_script "python3" "preScript.py" "../Lib/VHDL/psi_fix/testbench/psi_fix_cic_dec_fix_nch_pa
set dataDir [file normalize "../Lib/VHDL/psi_fix/testbench/psi_fix_cic_dec_fix_nch_par_tdm_tb/Data"]
tb_run_add_arguments    "-gOrder_g=3 -gRatio_g=10 -gDiffDelay_g=1 -gAutoGainCorr_g=True -gInFile_g=input
                                                "-gOrder_g=4 -gRatio_g=9 -gDiffDelay_g=2 -gAutoGainCorr_
                                                "-gOrder_g=4 -gRatio_g=6 -gDiffDelay_g=2 -gAutoGainCorr_

add_tb_run

create_tb_run "llrf_proc_adc_in_ddc_tb"
tb_run_add_pre_script "python3" "llrf_proc_adc_in_ddc_sim.py" "../Bittrue/Sim"
set dataDir [file normalize "../Bittrue/Stimuli/llrf_proc_adc_in_ddc"]
tb_run_add_arguments "-gDataDir_g=$dataDir"
add_tb_run

create_tb_run "llrf_proc_fdbk_ctrl_pi_tb"
tb_run_add_pre_script "python3" "llrf_proc_fdbk_ctrl_pi_sim.py" "../Bittrue/Sim"
set dataDir [file normalize "../Bittrue/Stimuli/llrf_proc_fdbk_ctrl_pi"]
tb_run_add_arguments "-gFileFolder_g=$dataDir"
add_tb_run
```

Non-regression test for LLRF project on-going
Before each merge to master branch
the maintainer is responsible to run such a script
Where generics can be modified on the fly

Mirror to github

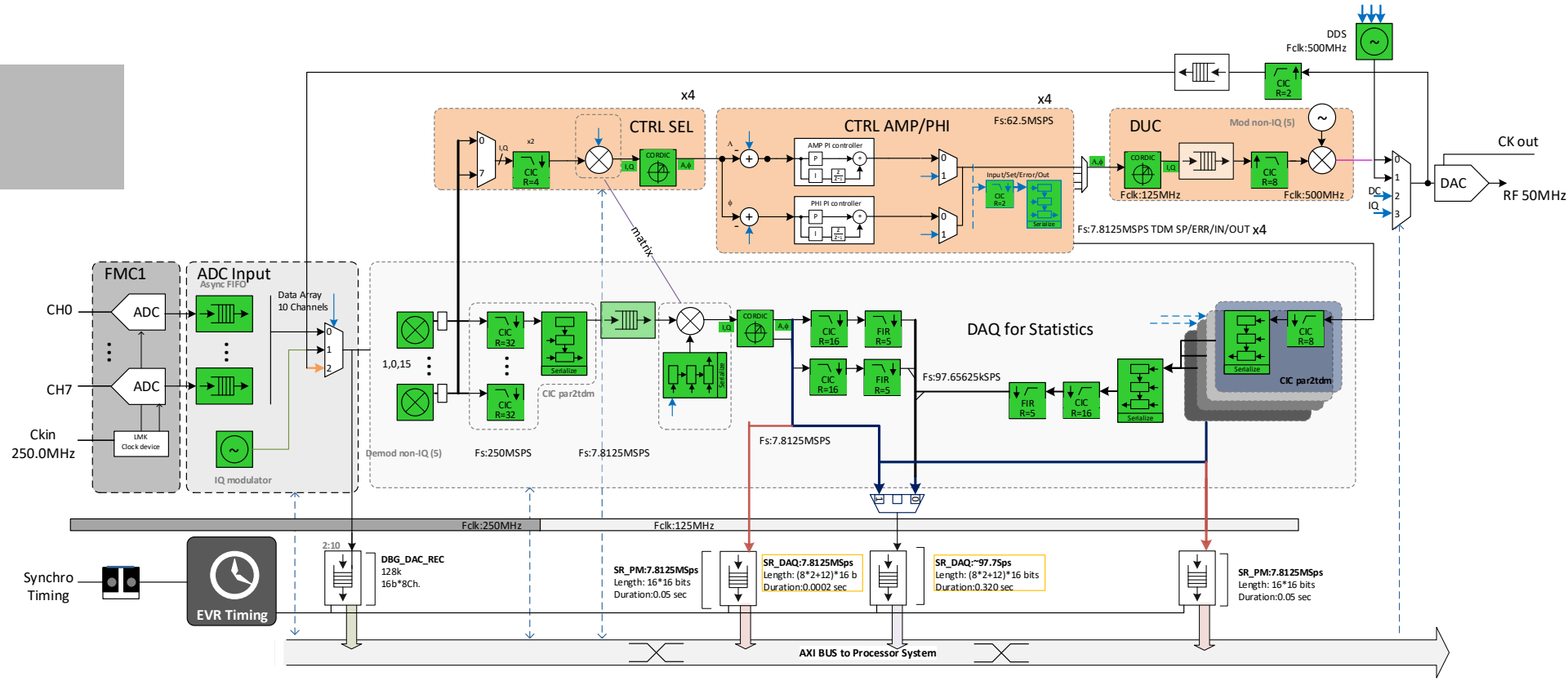
LLRF FPGA Firmware architecture overview



Demodulator, Down-Converter, filters, CORDIC, DDS, Matrix rotation, PI Controllers, Data acquisition buffer, fifos, DDS, modulator etc…

Does it sound familiar?

LLRF FPGA Firmware architecture overview



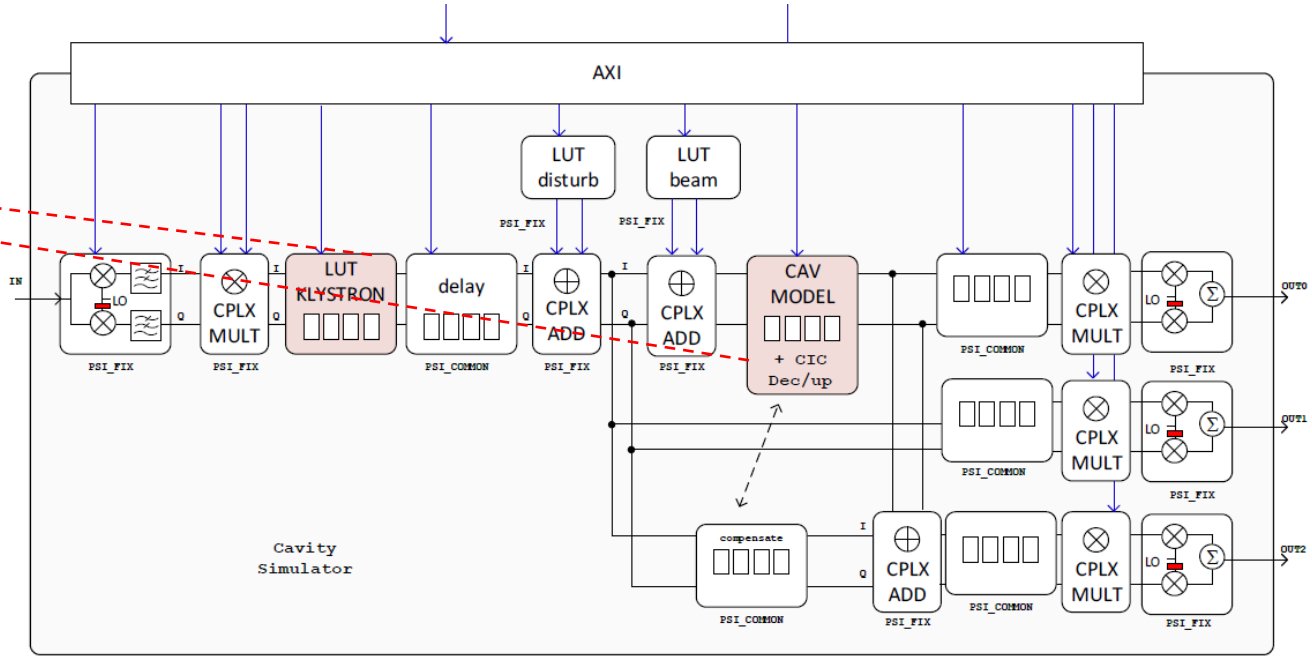
LLRF Architecture design using basic blocks belonging to PSI library
Architecture depends on you and when new component need to be developped think generic and contribute

Summer student experience

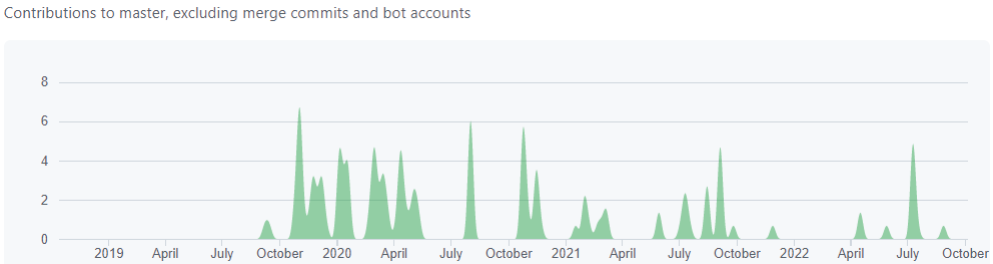
- Cavity simulator designed in VHDL by a summer student, 1st year of his Master. (*R.Basso*)
- Python model using psi_fix and jump to functionnal verification very quickly with few experience of FPGA design
- Project is on-going to go on target.
- Add two new components in the library State Space model solver & generic PRBS

Done by student

All other blocks from psi library

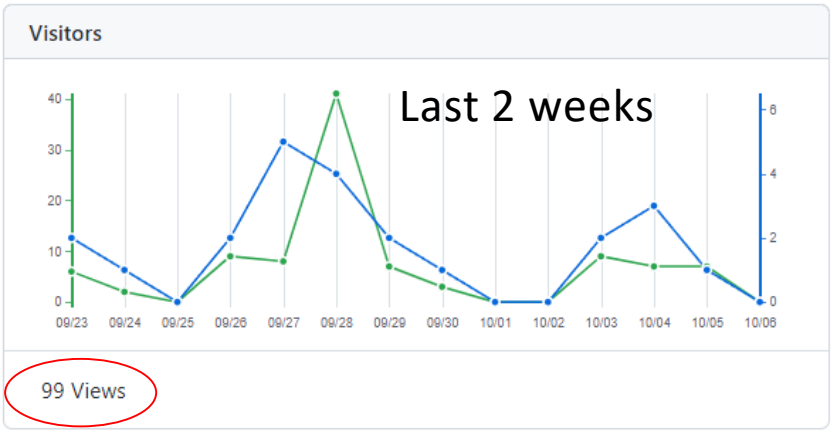


Reality of open-sourcing



GitHub Statistics - **PSI Common**

12 Contributors – 14 forks



Known users & contributors

- **Increase exchange**
- **Increase quality**
- **Increase productivity**

- **More work to maintain**
- **Not enough contribution from outside**
- **How to engage?**
- **Detect pertinence of user feedback**
- **Be aware that few from outside may ask you to do their job and this for free**

Summary

- Think «ReUse»
- Be curious and dare to use open source code, not everything is perfect not everything is trash let's try to make it work better together
- Participate whenever something is almost as you wish but not yet there
- Reactivity and comprehesive documentation are key for user to adhere
- Spare time and avoid same mistakes that others did in the past to focus on what really matters: Your architecture!

Contact: benoit.stef@psi.ch
0041 56 310 33 46

Git repository
https://github.com/paulscherrerinstitute/psi_common - *Resp. B.Stef*
https://github.com/paulscherrerinstitute/psi_fix - *Resp. R.Rybaniec*

My thanks go to

- GFA/AEK DSP Group
- Radek
- W. Koprek
- J. Purtschert
- Oliver Bruendler
- Enclustra GmbH
- Future contributors