OscImpDigital CPU-FPGA co-design framework in the context of satellite communication

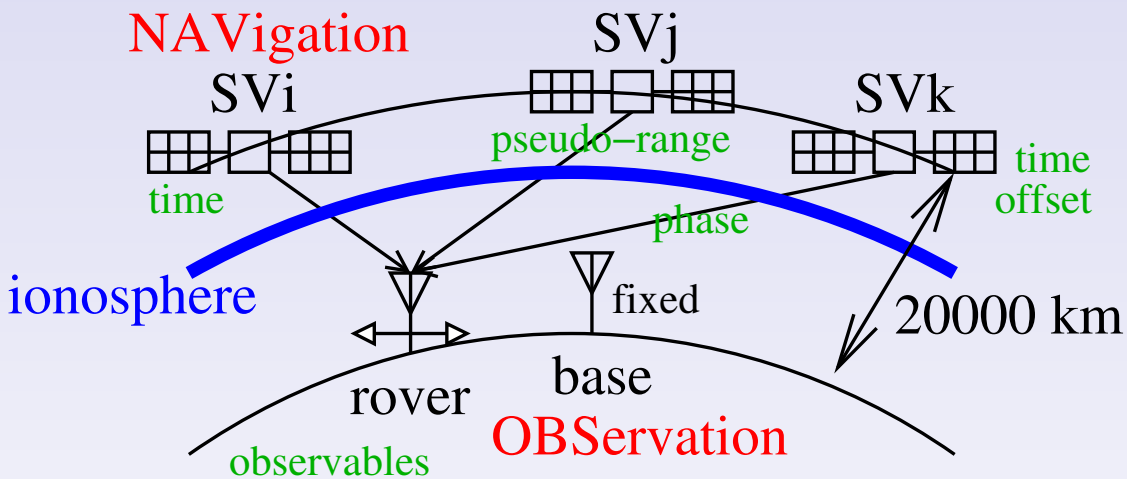
Goavec-Merou, Jean-Michel Friedt
FEMTO-ST Time & Frequency department, Besançon, France
Contact: {gwenhael.goavec,jmfriedt}@femto-st.fr

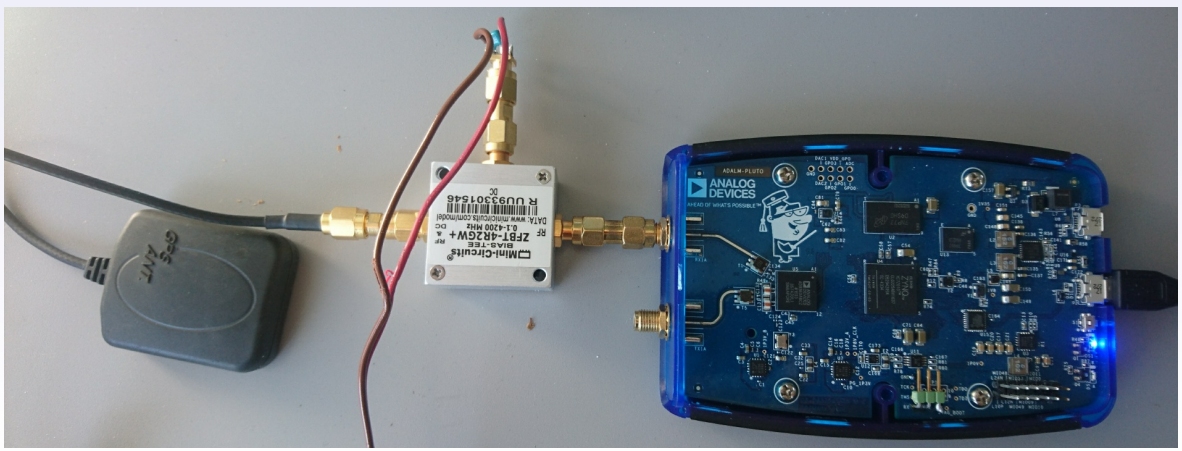
Slides at
https://github.com/oscimp/oscimpDigital/tree/master/doc/tutorials/plutosdr/2-PRN_on_PL

November 28, 2019

Why SDR-based GNSS decoding ?

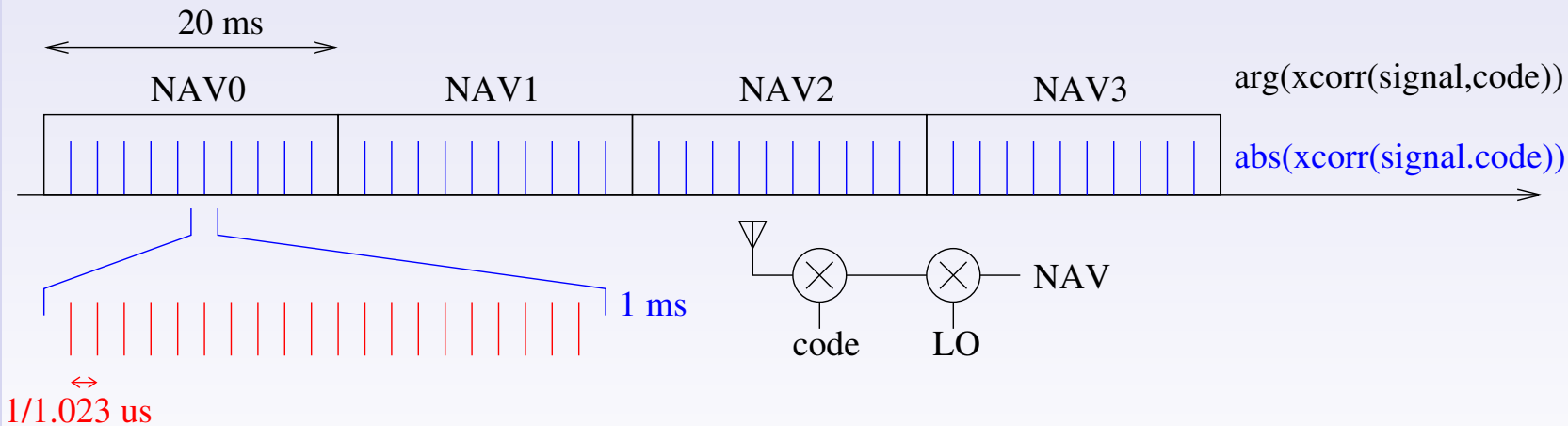
1. Flexibility of adding new features without updating hardware
2. Beyond timing & positioning: access to the raw I/Q stream
   - basic physics (reflectometry)
   - security (phased array for spoofing detection)
   - 1575.42 MHz within range of the PlutoSDR (AD9363 + Zynq SoC)





Basics on GPS encoding

1. CDMA (Code Division Multiple Access): all satellites transmit on the same frequency and their messages are encoded with individual orthogonal codes (Gold Codes)
2. Satellite identification: $xcorr(signal, code)$
3. Code orthogonality: $xcorr(code_i, code_j) = \delta_{i,j}$
4. Doppler shift: need to compensate for remote clock frequency wrt ground clock & local clock offset wrt remote atomic clocks



Basics on GPS encoding

1. CDMA (Code Division Multiple Access): all satellites transmit on the same frequency and their messages are encoded with individual orthogonal codes (Gold Codes)
2. Satellite identification: $xcorr(signal, code)$
3. Code orthogonality: $xcorr(code_i, code_j) = \delta_{i,j}$
4. Doppler shift: need to compensate for remote clock frequency wrt ground clock & local clock offset wrt remote atomic clocks

Intensive use of correlations [1]

$$xcorr(x, y)(\tau) = \int x(t)y(t+\tau)dt$$

or through the convolution theorem:

$$FFT(xcorr(x, y)(\tau)) = FFT(x) \cdot FFT(y^*)$$

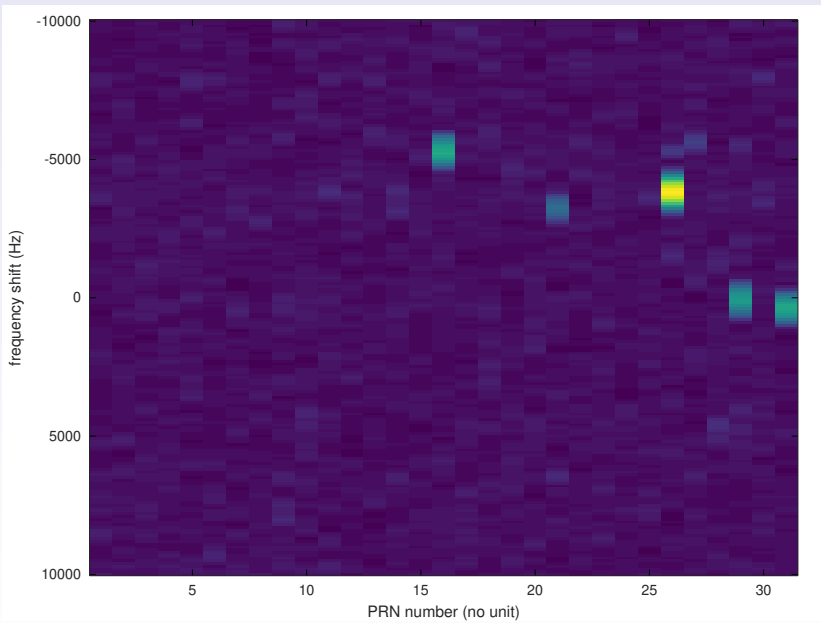
[1] Time-domain implementation on FPGA allows for pipelined computation as samples are collected

Basics on GPS encoding

GPS acquisition in 10 lines of Matlab program [2] (two nested loops – satellite number and frequency)

```
1  pkg load signal
2  x=read_complex_binary(filename,1024*128); fs=1.023; % sampling rate in MHz
3  x=x-mean(x);
4  freq0=[-10.5e3:500:10.5e3];                         % Doppler range
5  time=[0:1/fs/1e6:length(x)/fs/1e6]';time=time(1:end-1);
6  for m=[1:31]                                        % loop on all satellites
7     a=cacode(m,fs/1.023); a=a-mean(a);
8     l=1;
9     for freq=freq0                                   % loop on all frequency offsets
10      mysine=exp(j*2*pi*(-freq)*time);
11      xx=x.*mysine;                                  % frequency shift the signal
12      [u(l,m),v(l,m)]=max(abs(xcorr(a,xx,'none')));  % check for cross correlation max.
13      l=l+1;
14    end
15 end
```

- Orbital mechanics: *Doppler* $\in [-5000, 5000]$ Hz
- Map xcorr max as a function of space vehicle number and frequency shift
- When a satellite is visible, sharp xcorr peak when frequency offset is compensated for



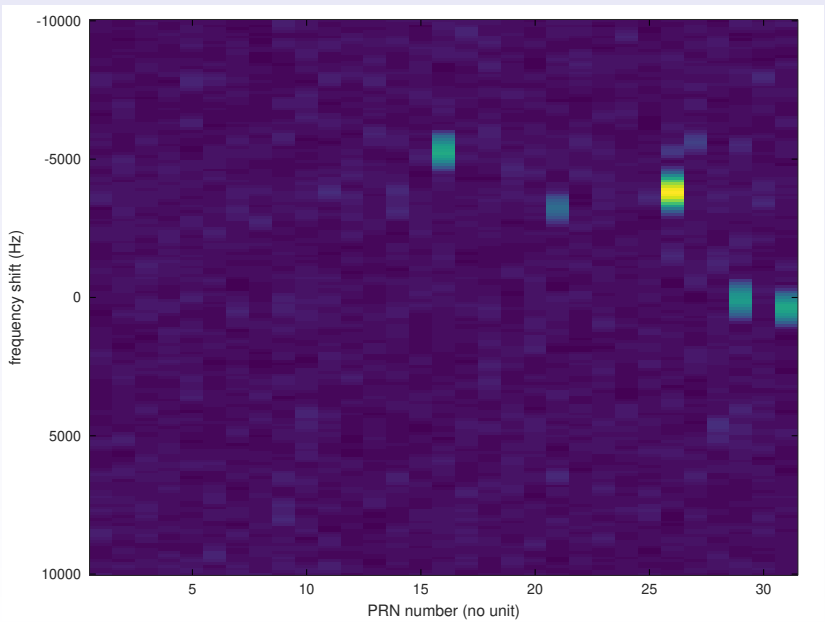
[2] using the C/A code generator https://www.mathworks.com/matlabcentral/fileexchange/14670-gps-c-a-code-generator

Basics on GPS encoding

GPS acquisition in 10 lines of Matlab program (single loop on space vehicle number)

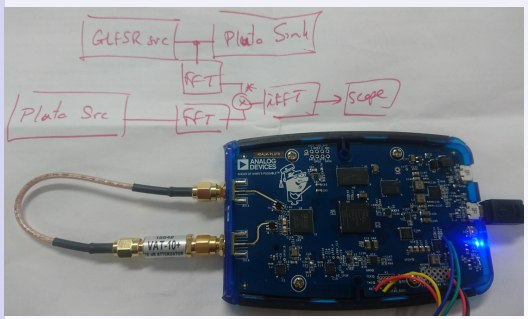
```
1  pkg load signal
2  x=read_complex_binary(filename,1024*128); fs=1.023; % sampling rate in MHz
3  x=x-mean(x);
4  freq0=[-10.5e3:500:10.5e3];                         % Doppler range
5  time=[0:1/fs/1e6:length(x)/fs/1e6]';time=time(1:end-1);
6  % doppler frequency shift matrix whose FFT is computed
7  doppler=exp(j*2*pi*freq0'*time');                   % 43x131072 matrix
8  data=ones(43,1)*x';
9  all=doppler.*data;                                  % Doppler-shifted data
10 allf=fft(all');
11 for m=[1:31]                                        % loop on all satellites
12   a=cacode(m,fs/1.023);                             % CA code of satellite m
13   a=[a zeros(1,length(all)-length(a))];             % zero padding
14   a=a-mean(a);
15   pattern=ones(43,1)*a;                             % 43x131072 matrix
16   af=fft(pattern');
17   correlation=ifft(af.*conj(allf))';
18 end
```

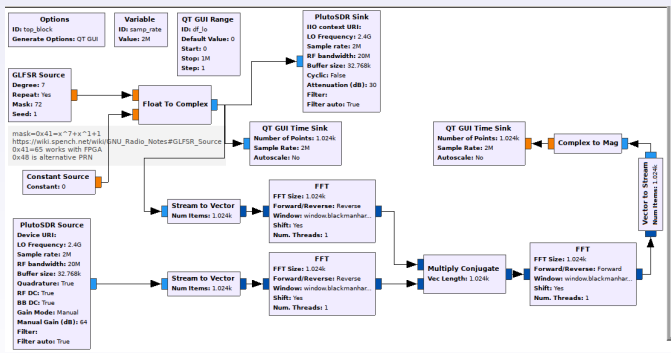
- Replace loops (inefficient) with matrix multiplication
- Parallelizing the frequency operations halves the computation time

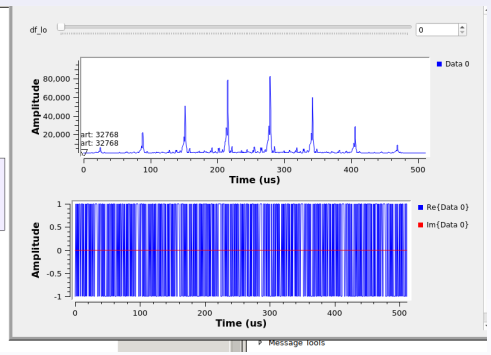


Using the embedded FGPA

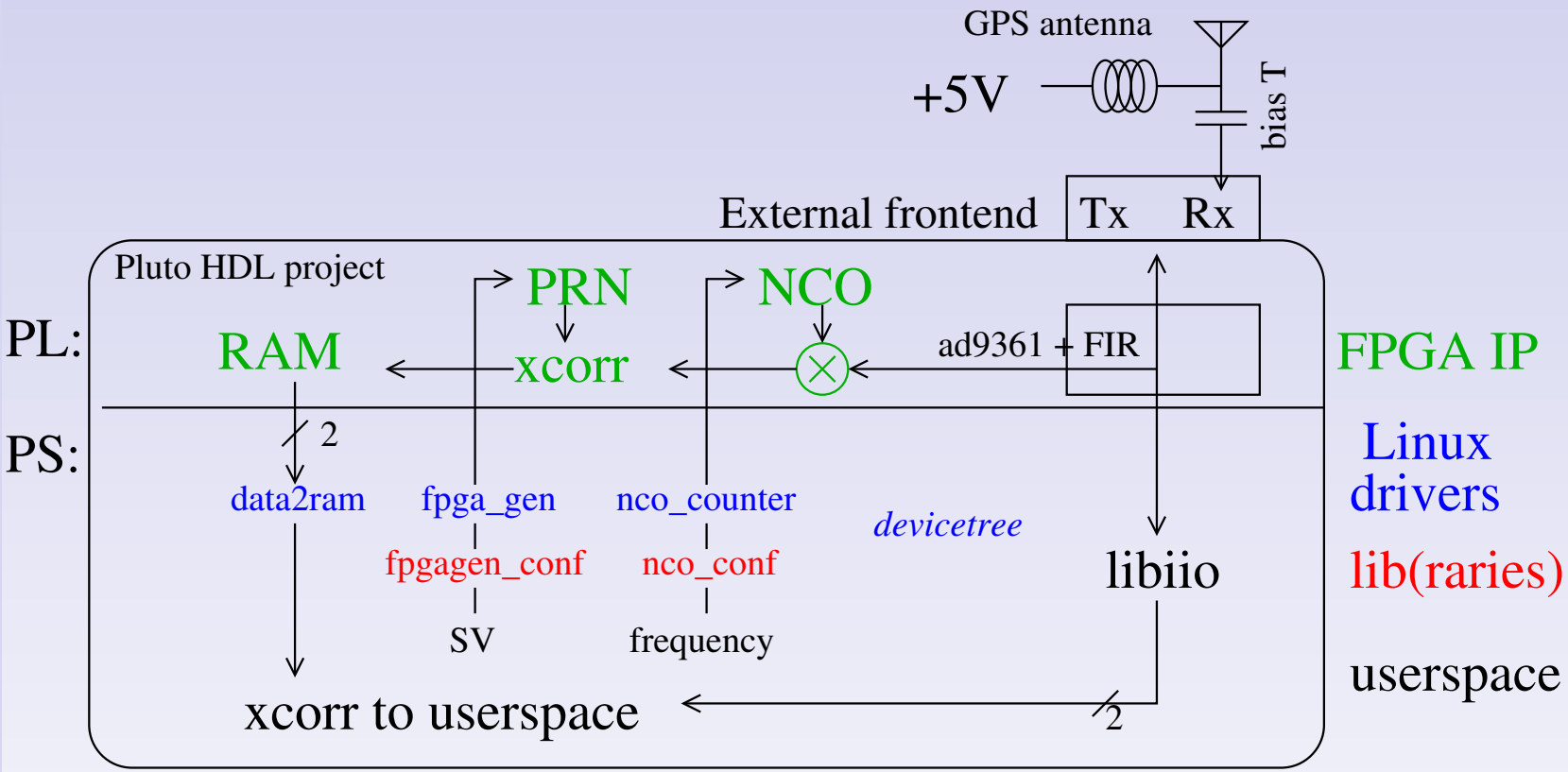
- GNU/Octave implementation: 1 to 2 second/satellite
  $\Rightarrow \simeq$1 min for acquisition depending on frequency steps
- The PlutoSDR Zynq is only used for data collection and transfer to the PC (bandwidth limited by USB)
- Preprocessing on the Zynq FPGA removes the communication bandwidth bottleneck
- Making best use of the available resources on the embedded FPGA (PL)
- Possible additional pre-processing on the embedded CPU (PS) running GNU Radio before sending over USB







Principle



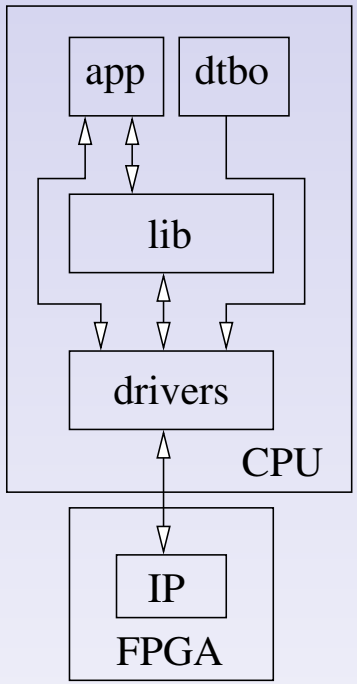
- PL: collect data from AD9363, frequency transposition (NCO), Gold Code generation & correlation
- PS: loop frequency, loop space vehicle number, fetch correlation, control AD9363 (libiio)

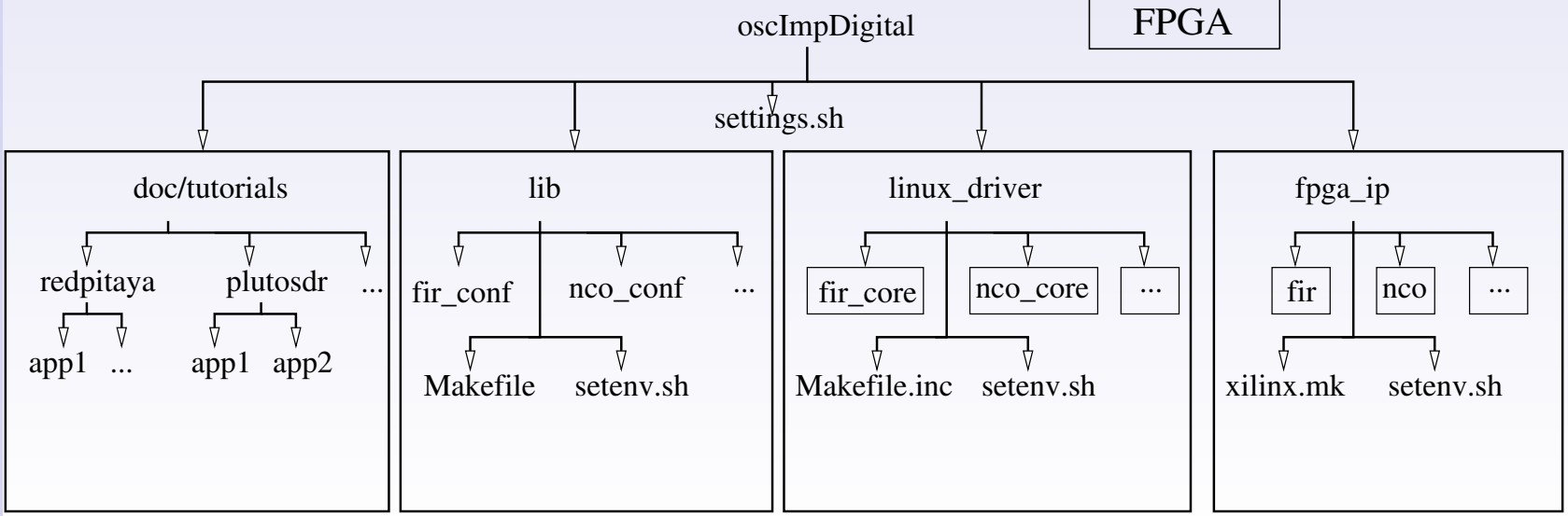
Complex interaction between FPGA processing blocks and processor userspace through Linux drivers (modules)

Oscimp EcoSystem

Purpose: provide a coherent environment to create design (FPGA), and application:

- blocks (IP) with algorithm level of implementation (FPGA);
- GNU/Linux hierarchy compliance (driver/library/application);
- tools to generate some files and scripts/Makefile to factorize most common part.





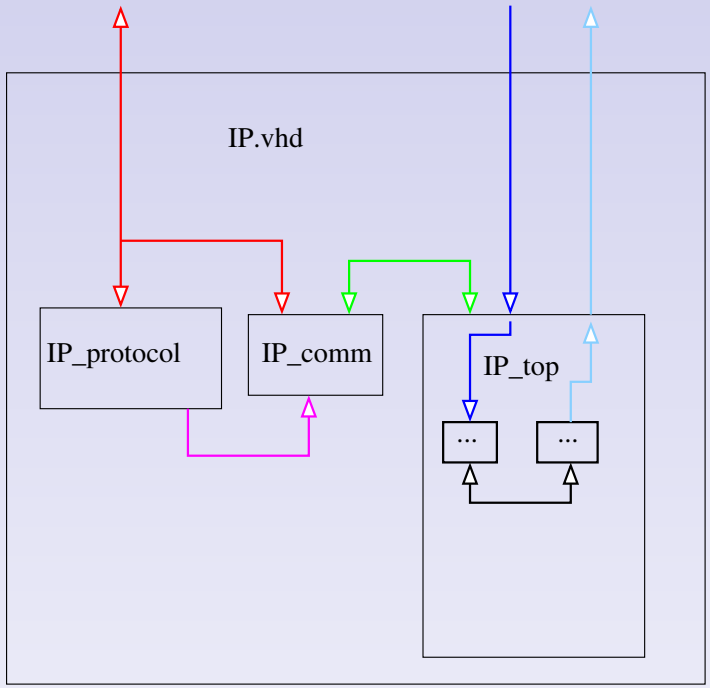
FPGA

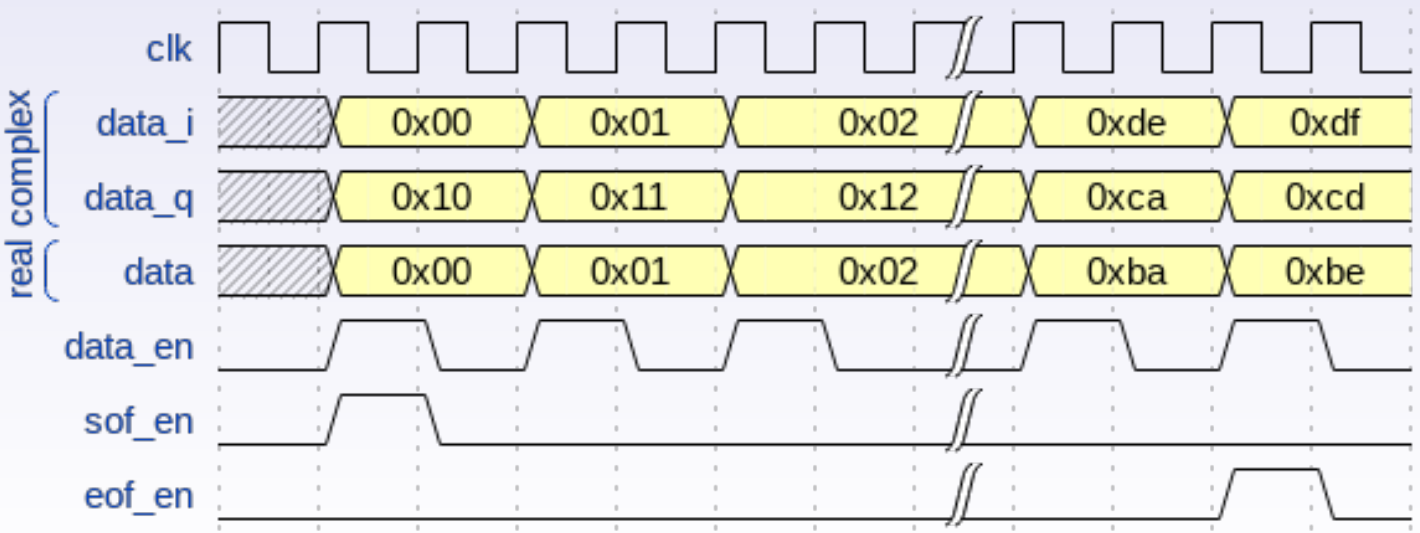
Algorithms or utilities functions.
Developer aspect:

- normalize interfaces between blocks
- isolation between implementation and communication

End user aspect:

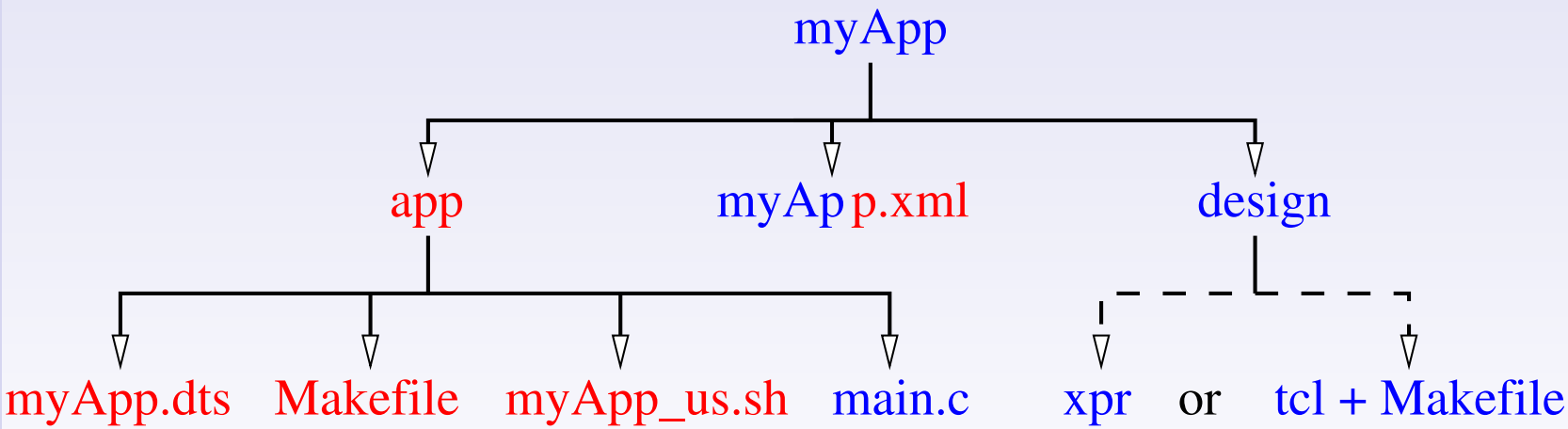
- 0, 1 or more interface to connect;
- AXI interface automatically connected.





Project structure

- TCL script or GUI generated FPGA design
- devicetree (`.dts`) provides which driver must be used and base addresses;
- `Makefile` to cross-compile application and generate the `dtbo` from dts
- `applicationName_us.sh`: a shell script used to flash FPGA, load devicetree and drivers;
- `main.c`: user application



user defined

automatically created by module_generator (based on XML file)

CPU: module_generator

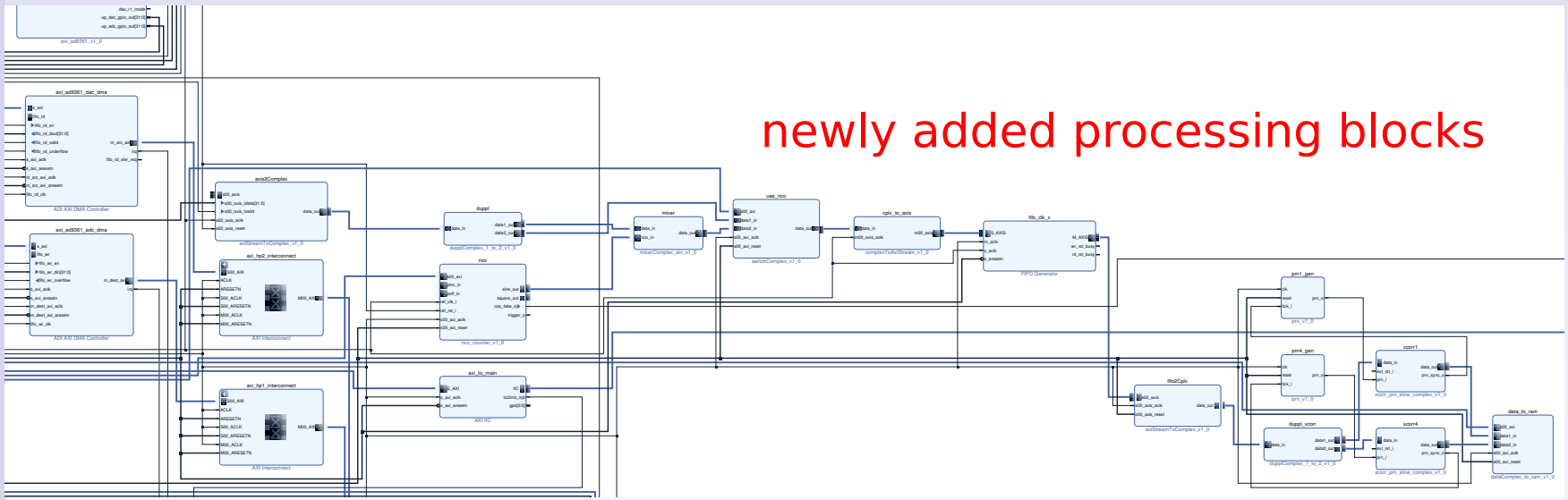
- Used to generate some files in app directory.
- use an XML file for design's informations.

```
module_generator -dts myApp.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<project name="tutorial5" version="1.0">
  <options>
    <option target="makefile" name="USE_STATIC_LIB">1</option>
    <option target="makefile" name="LDFLAGS">-liio</option>
  </options>
  <ips>
    <ip name ="dataComplex_to_ram" >
      <instance name="data1600" id = "0"
        base_addr="0x43c00000" addr_size="0xffff" />
    </ip>
    <ip name ="nco_counter">
      <instance name="nco" id = "0"
        base_addr="0x43c10000" addr_size="0xffff" />
    </ip>
  </ips>
</project>
```

Application to GPS decoding

- TCL scripts define the processing functions, their settings and how they are connected to each other
- Zynq on the PlutoSDR ⇒ Xilinx Vivado (despite platform independence of OscimpDigital)



Dual PRN generator and cross-correlation with the received datastream frequency transposed using the NCO.

22 s on Zynq PL (limited by the FPGA area limiting the number of parallel correlations) v.s **108 s on 2.6 GHz PC** (GNU/Octave)

Conclusion

- OscimpDigital as a **flexible framework** for assembling signal processing blocks on the FPGA in charge of collecting radiofrequency data
- Platform **independence** (useful investment for Intel/Altera SoC as well)
- **Consistent** IP–Linux kernel module–library–userspace application
- Application to GPS decoding (dual channel, acquisition step) as SatCom demonstration
- **Perspective**: port to ADRV9361 (Zynq 7035 ≫ 7010)

Users **not familiar with VHDL will benefit** from this framework since functional processing blocks are provided

EU GNU Radio Days

Resources:

`https://github.com/trabucayre/redpitaya` (Buildroot BR2_external)
`https://github.com/oscimp/PlutoSDR` (Buildroot BR2_external)

`https://github.com/oscimp/oscimpDigital` (IP, driver, lib, tools & doc)

Clone repository and submodules:

```
git clone --recursive https://github.com/oscimp/oscimpDigital.git
```