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

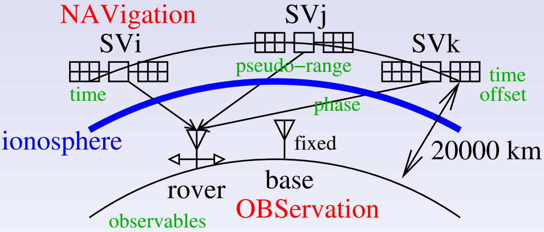
Goavec-Merou, Jean-Michel Friedt
FEMTO-ST Time & Frequency department, Besançon, France
Contact: {gwenhael.goavec,jmfriedt}@femto-st.fr

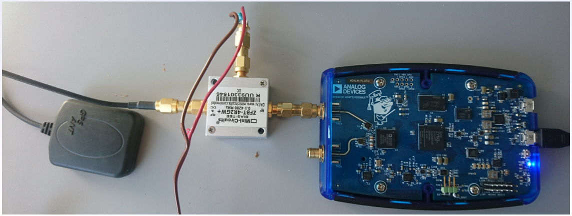
References at
http://jmfriedt.free.fr

November 21, 2019

Why SDR-based GNSS decoding ?

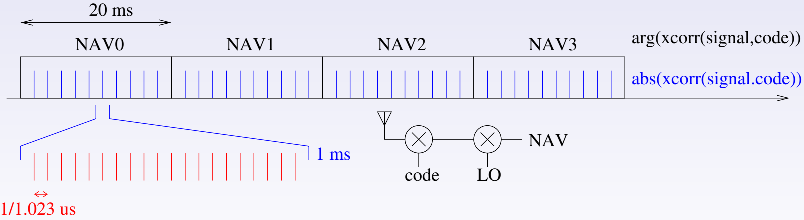
1. Flexibility of adding new features without updating hardware
2. Beyond timing & positioning: access to the raw I/Q stream
   - basic physics (reflectometry)
   - security (phased array for spoofing detection)
   - 1575.42 MHz within range of the PlutoSDR (AD9363 + Zynq SoC)





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:

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

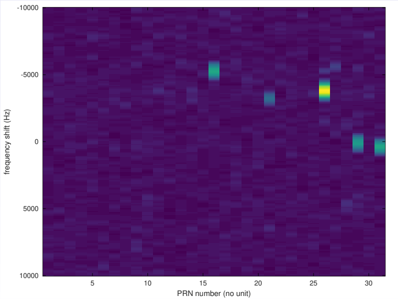
[1] Time-domain implementation on FPGA allows for pipelined computation as samples are collected

Basics on GPS encoding

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

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



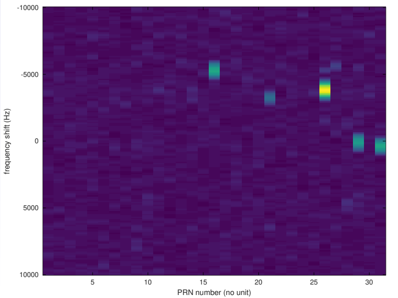
[2] using the C/A code generator https://www.mathworks.com/matlabcentral/fileexchange/14670-gps-c-a-code-generator

Basics on GPS encoding

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

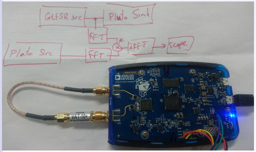
```
pkg load signal
x=read_complex_binary(filename,1024*128); fs=1.023; % sampling rate in MHz
x=x-mean(x);
freq0=[-10.5e3:500:10.5e3];                        % Doppler range
time=[0:1/fs/1e6:length(x)/fs/1e6]';time=time(1:end-1);
% doppler frequency shift matrix whose FFT is computed
doppler=exp(j*2*pi*freq0'*time');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

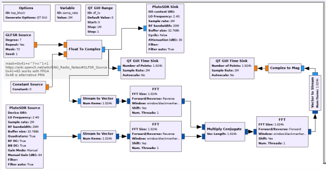
- Replace loops (inefficient) with matrix multiplication
- Parallelizing the frequency operations halves the computation time

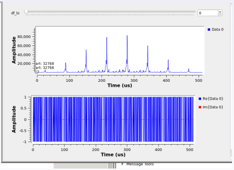


Using the embedded FGPA

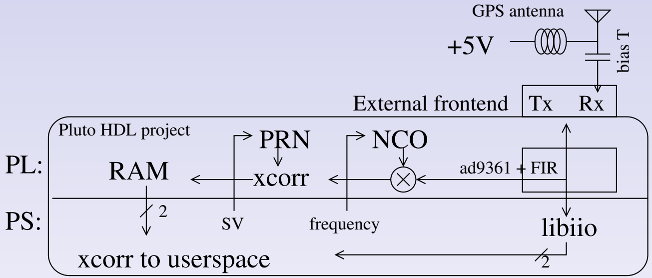
- GNU/Octave implementation: 1 to 2 second/satellite
  $\Rightarrow \simeq 1$ min for acquisition depending on frequency steps
- The PlutoSDR Zynq is only used for data collection and transfer to the PC (bandwidth limited by USB)
- Preprocessing on the Zynq FPGA removes the communication bandwidth bottleneck
- 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)

The OscimpDigital framework

The OscimpDigital framework