TUT-07: Software Defined Radio in the context of stable time and frequency signal generation and dissemination

J.-M Friedt

FEMTO-ST Time & Frequency department, Besançon, France

jmfriedt@femto-st.fr

video recording at `https://www.youtube.com/watch?v=EAXi6saVBXE`
slides and references at `http://jmfriedt.free.fr/jmfriedt_tutorialIFCS2025.pdf`
datasets available at `http://jmfriedt.free.fr/jmfriedt_tutorialIFCS2025.tar.gz`

May 18, 2025

Outline

1. What is Software Defined Radio (SDR)? Hardware & software architecture
2. Discrete time signal processing: complex numbers and aliasing (GNU Radio)
3. Digital super-heterodyne architecture to avoid IQ imbalance (Sentinel 1 spaceborne RADAR & NanoVNA)
4. Spectrum spreading for time and frequency transfer
5. GNSS signal acquisition (GPS L1 BPSK, Galileo E1 BOC)
6. VLF and TWSTFT signals

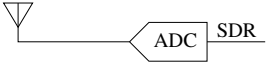
SDR: flexible and stable approach to RF signal processing

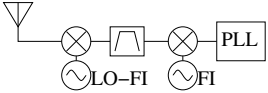
Software Defined Radio (SDR): digital radiofrequency (RF) digital [1] signal processing [2]

- **stable**: an algorithm will not drift over time ($\neq$ passive component, e.g. capacitor) or with environmental conditions
- **flexible**: ability to tune operating conditions without halting operation
- **reconfigurable**: one hardware, many application only requiring reconfiguration of connections

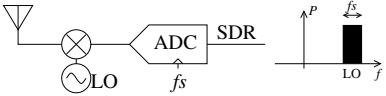
+ data logging + communication over networks ...

Application to **oscillator metrology**, (secure) **time transfer** and **timing** (of active/passive RADAR signals)





In the current architecture, carrier (LO) will not matter (removed by RF frontend), only bandwidth (sampling rate f_s) will matter $\longrightarrow$



[1] D.A. Mindell, *Digital Apollo: Human and Machine in Spaceflight*, MIT Press (2011)

[2] D.A. Mindell, *Between Human and Machine: Feedback, Control, and Computing before Cybernetics*, Johns Hopkins University Press (2003)

Non exaustive literature review of SDR for time & frequency

- T. Pany, *& al.*, *GNSS software-defined radio: history, current developments, and standardization efforts*, J. of the Institute of Navigation **71** (2024)

Red Pitaya/STEMLab (baseband)

- T. Preuschoff *& al.*, *Digital laser frequency and intensity stabilization based on the STEMlab platform (originally Red Pitaya)*, Rev. Sci. Instrum. **91** (8) 083001 (2020)
- A. Tourigny-Plante *& al.*, *An open and flexible digital phase-locked loop for optical metrology*, Rev. Sci. Instrum. **89** (9) 093103 (2018)
- P. Mahnke, *Characterization of a commercial software defined radio as high frequency lock-in amplifier for FM spectroscopy*, Rev. Sci. Instrum. **89** (1) 013113 (2018)
- J.A. Sherman *& al.*, *Oscillator metrology with software defined radio*, Rev. Sci. Instrum. **87** (5) 054711 (2016)
- C. Hasselwander *& al.*, *gr-MRI: A software package for magnetic resonance imaging using software defined radios*, Journal of Magnetic Resonance **270** 47–55 (2016)
- G.A. Stimpson *& al.*, *An open-source high-frequency lock-in amplifier*, Rev. Sci. Instrum., **90** (9) 094701 (2019)
- A. C. Cárdenas *& al.*, *Phase Noise and Frequency Stability of the Red-Pitaya Internal PLL*, IEEE Trans. Ultrasonics, Ferroelectrics, and Frequency Control **66** (2) 412–416 (2019)
- S.J. Yu *& al.*, *The performance and limitations of FPGA-based digital servos for atomic, molecular, and optical physics experiments*, Rev. Sci. Instrum. **89** 025107 (2018)

Ettus Research B210:

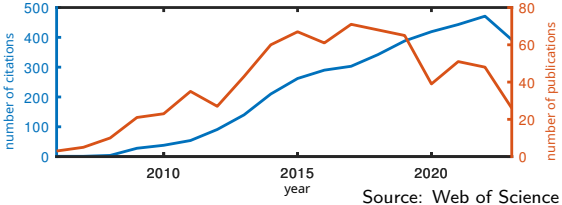
- Paul Meaney *& al.*, *A 4-channel, vector network analyzer microwave imaging prototype based on software defined radio technology*, Rev. Sci. Instrum. **90** 044708 (2019)

Ettus Research E312:

- S. Prager *& al.*, *Wireless subnanosecond RF synchronization for distributed ultrawideband software-defined radar networks*, IEEE Trans. Microwave Theory and Techniques **68**(11) 4787–4804 (2020)

Ettus Research N210:

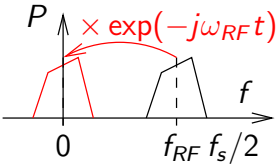
- C. Andrich *& al.*, *High-precision measurement of sine and pulse reference signals using software-defined radio*, IEEE Trans. Instrum. & Meas. **67** (5) 1132–1141 (2018)

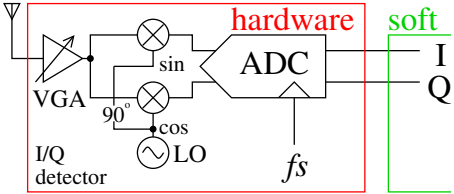


Source: Web of Science

Why SDR handles complex numbers, ...

- real signal Fourier transform is conjugate symmetric (negative frequency and positive frequency magnitude equal)
- the spectrum transposed from RF band $s = A\exp(j\omega t + \varphi)$ to baseband need **not** be symmetric $\Rightarrow$ complex mixing to create I and Q (Identity and Quadrature)
- $I = s \cdot \cos(\omega_{RF} t)$ and $Q = s \cdot \sin(\omega_{RF} t)$ so that $A = |I + jQ|$ and $\varphi = arg(I + jQ)$ if $\omega_{RF} = \omega$
- In other words ... imagine a single frequency transposition $s(t) \cdot \cos(\omega_{RF} t)$: if the modulation is on the amplitude, then $A \cos\varphi = 0$ if $\varphi = \pi/2$, $\forall A$.
- Solution: add a second signal maximized when $\cos\varphi = 0$, i.e. using sin
- since $\sin(x)$ is $\cos(x + \pi/2)$: quadrature of the local oscillator





... and double frequency transposition: digital IQ vs analog IQ [3]

also used in the NanoVNA (`https://github.com/ttrftech/NanoVNA`)

MPC-S1 **Sentinel-1**

Ref	
MPC Nom	DI-MPC-IPFDPM
MPC Ref	MPC-0307
Issue/Revision:	2/2
Date	07/06/2019

Sentinel-1 Level 1 Detailed Algorithm Definition

4.1Raw Data Analysis

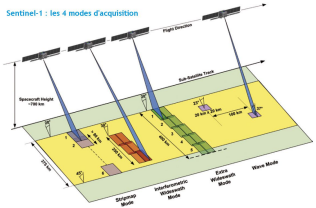
Raw data analysis is required in order to perform corrections of the I and Q channels of the raw signal data. The classical raw data correction (applied for instance in the case of ENVISAT-ASAR and RADARSAT-2) involves (see also Section 9.2):

- I/Q bias removal
- I/Q gain imbalance correction
- I/Q non-orthogonality correction

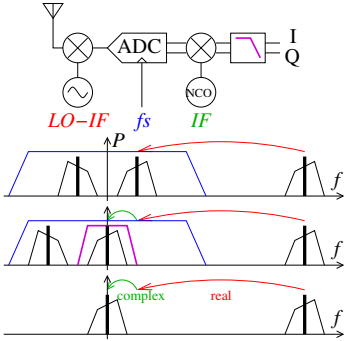
For Sentinel-1 however, the instrument's receive module performs the demodulation in the digital domain, therefore the I/Q gain imbalance and I/Q non-orthogonality corrections are no longer necessary.

The raw data analysis necessary for the raw data correction of ASAR data is defined in [R-6]. Since the IPF also supports the processing of ASAR data, for completeness, the ASAR raw data analysis scheme is reproduced in this section.

Even though for Sentinel-1 the I/Q gain imbalance and the I/Q non-orthogonality corrections are not necessary, they will be made available optionally, using configuration input parameters. Irrespective to the correction flag though, the Raw Data Analysis described in this section will be performed and the results reported for both ASAR and Sentinel-1 data.



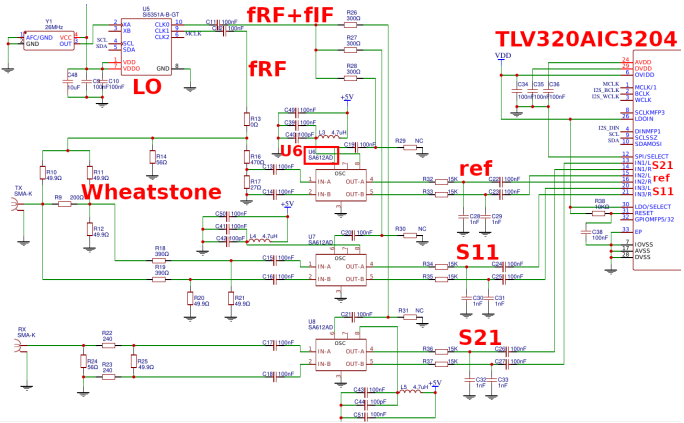
- real case: $I = s(t)\cos(\omega_{RF}t)$ $Q = s(t)\cdot(1+\varepsilon)\sin(\omega_{RF}t+\delta\varphi)$: analog IQ imbalance
- avoid analog IQ imbalance with dual transposition step
- digital domain frequency transposition: `Xlating FIR Filter`



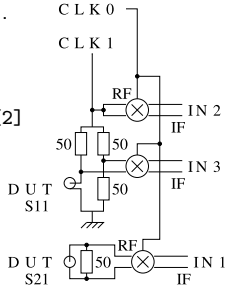
[3] `http://www.esa.int/var/esa/storage/images/esa_multimedia/images/2016/03/sentinel-1_radar_mission/15857809-1-eng-GB/Sentinel-1_radar_mission_pillars.jpg`

NanoVNA frequency transposition architecture

Openhardware[4]/opensource[5] vector network analyzer



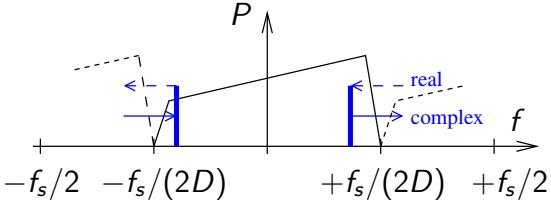
- `main.c`: `static int32_t frequency_offset = 5000;` the intermediate frequency sampled by the audio codec, with (`si5351.c`) `CLK0`: `frequency + offset` and `CLK1`: `frequency` when requesting `si5351_set_frequency_with_offset(uint32_t freq, int offset, uint8_t drive_strength)`.
- Audio signal sampled at 48 kHz ⇒ `sincos_tbl[48]` [2] in `dsp.c` for transposition by 5 kHz in `dsp_process()` of `dsp.c`).

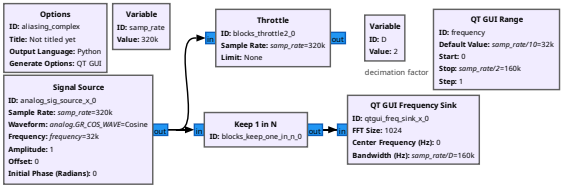


[4] https://raw.githubusercontent.com/hugen79/NanoVNA-H/master/doc/Schematic_nanovna-H_REV3_4_2.pdf
[5] https://github.com/ttrftech/NanoVNA

Discrete time processing ⇒ aliasing

Simple examples to become familiar with **GNU Radio** and discrete time signal processing, filter configuration

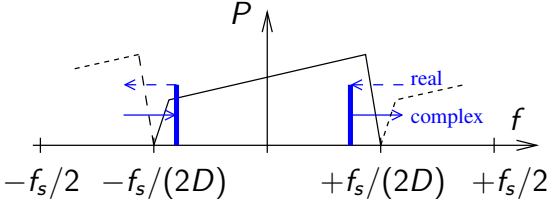


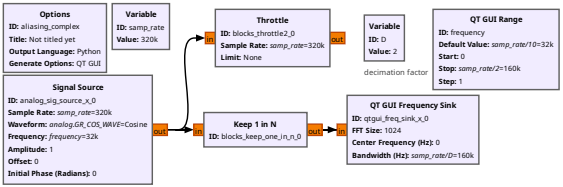


- alway low-pass filter before decimating by factor D
- cutoff-frequency is frequency range after decimation: $f_s/(2D)$
- transition width δf determines processing load: FFT resolution f_s/N or FIR filter number of taps N is determined as $\delta f \leq f_s/N \Rightarrow N \geq f_s/\delta f$. Select `transition width` as f_s/N for a FIR filter with N taps.
- GNU Radio `Low pass filter` includes low-pass filtering and decimation (for reduced processing load) but filtering is performed at input bandwidth and only 1 in D output is computed.

Discrete time processing ⇒ aliasing

Simple examples to become familiar with **GNU Radio** and discrete time signal processing, filter configuration

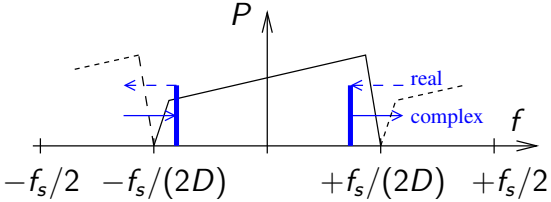


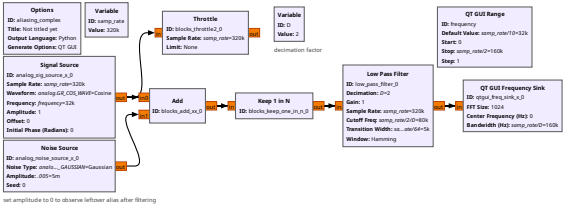


- alway low-pass filter before decimating by factor D
- cutoff-frequency is frequency range after decimation: $f_s/(2D)$
- transition width δf determines processing load: FFT resolution f_s/N or FIR filter number of taps N is determined as $\delta f \leq f_s/N \Rightarrow N \geq f_s/\delta f$. Select `transition width` as f_s/N for a FIR filter with N taps.
- GNU Radio `Low pass filter` includes low-pass filtering and decimation (for reduced processing load) but filtering is performed at input bandwidth and only 1 in D output is computed.

Discrete time processing ⇒ aliasing

Simple examples to become familiar with **GNU Radio** and discrete time signal processing, filter configuration





- alway low-pass filter before decimating by factor D
- cutoff-frequency is frequency range after decimation: $f_s/(2D)$
- transition width δf determines processing load: FFT resolution f_s/N or FIR filter number of taps N is determined as $\delta f \leq f_s/N \Rightarrow N \geq f_s/\delta f$. Select `transition width` as f_s/N for a FIR filter with N taps.
- GNU Radio `Low pass filter` includes low-pass filtering and decimation (for reduced processing load) but filtering is performed at input bandwidth and only 1 in D output is computed.

Fundamental of time transfer: outline

How to demonstrate time transfer with SDR?

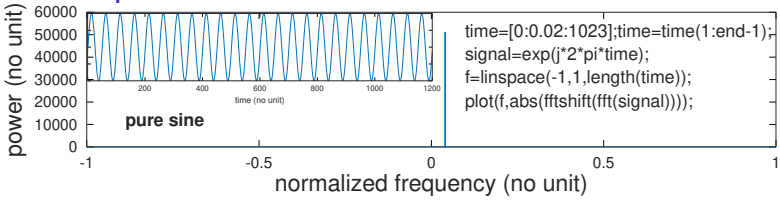
- RADAR range resolution: $\Delta R \geq \frac{c_0}{2B}$ ($c_0 = 300$ m/μs, bandwidth B)
- spectrum spreading: maximize B by all means (pulse, frequency sweep, frequency steps, noise ...)
- receive time delayed copies of the transmitted signal: matched filter = correlation (search for delayed copies of the emitted signal)

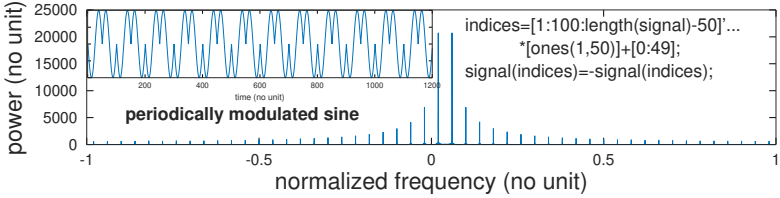
$$xcorr(x, y)(\tau) = \int_{-T/2}^{T/2} x(t)y(t+\tau)dt \Rightarrow \text{ identify } \tau \text{ maximizing } xcorr$$

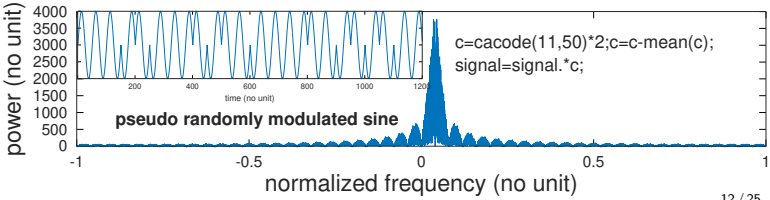
- maximize averaging time T to smooth out noise
- maximize B for the correlation peak width $1/B$ to be as narrow as possible
- Pulse Compression Ratio: $B \times T$

Spectrum spreading numerical experiments

- Carrier frequency and bandwidth are two unrelated quantities which can be tuned independently
- Carrier frequency defined by first frequency transposition stage (RF frontend) whereas bandwidth defined by ADC sampling rate
- Binary Phase shift keying: $\varphi \in [0; \pi]$ for spectrum spreading

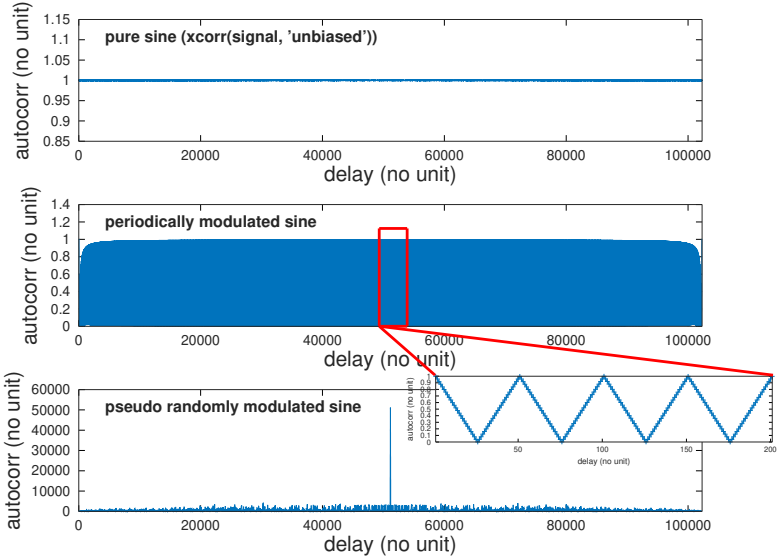






Spectrum spreading numerical experiments

- Carrier frequency and bandwidth are two unrelated quantities which can be tuned independently
- Carrier frequency defined by first frequency transposition stage (RF frontend) whereas bandwidth defined by ADC sampling rate
- Binary Phase shift keying: $\varphi \in [0; \pi]$ for spectrum spreading



Spectrum spreading numerical experiments

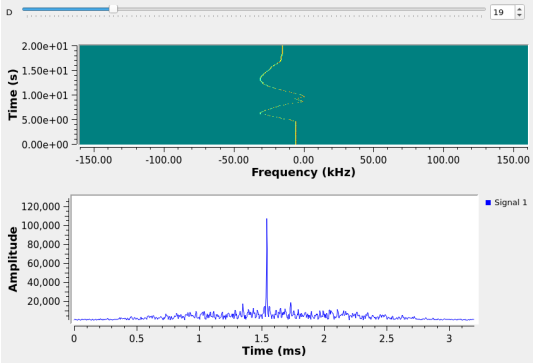
From convolution to correlation:

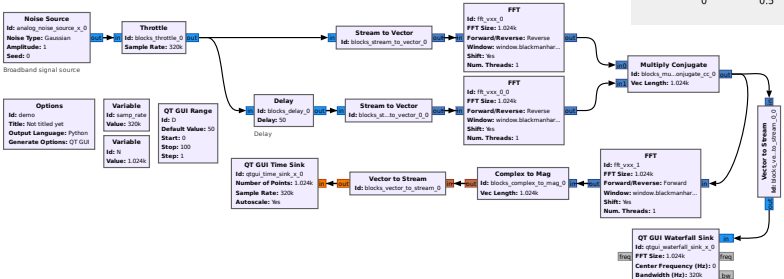
- Convolution: $conv(s,r)(\tau) = \int s(t)r(\tau - t)dt$
- Practical computation of convolution:

$$FT(conv(s,r)) = FT(s) \cdot FT(r)$$

- Correlation: $corr(s,r)(\tau) = \int s(t)r(t + \tau)dt$
- Convolution $\rightarrow$ correlation: time reversal
- since $\exp(j\omega t)^* = \exp(-j\omega t)$, we conclude

$$FT(corr(s,r)) = FT(s) \cdot FT^*(r)$$





Spectrum spreading numerical experiments

From convolution to correlation:

- Convolution: $conv(s,r)(\tau) = \int s(t)r(\tau - t)dt$
- Practical computation of convolution:

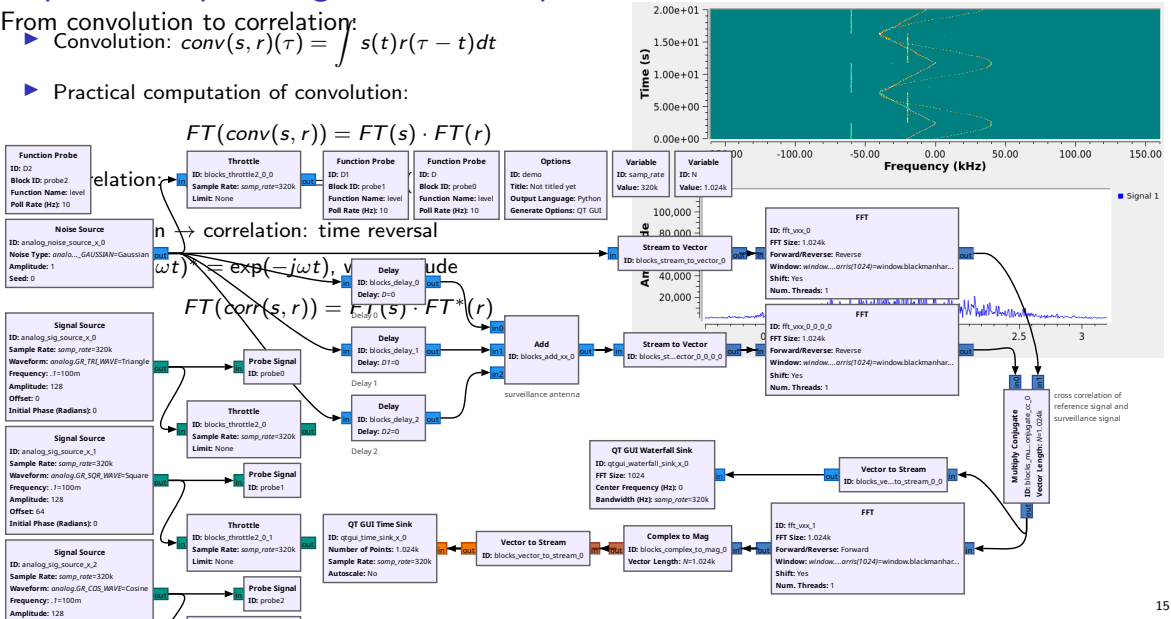
$$FT(conv(s,r)) = FT(s) \cdot FT(r)$$

relation:

n $\rightarrow$ correlation: time reversal

$\omega t)^* = \exp(-j\omega t)$, v ude

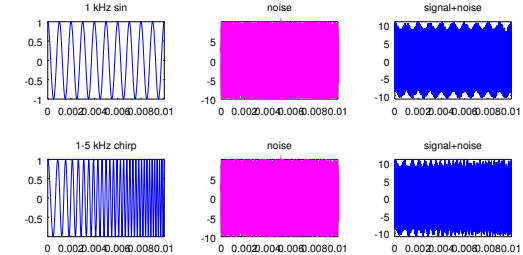
$$FT(corr(s,r)) = FT(s) \cdot FT^*(r)$$

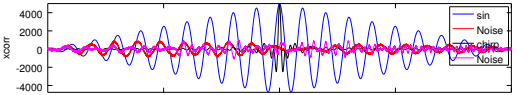


Pulse compression basics

- The longer the code (T), the longer the time during which the integral of xcorr accumulates energy and **smoothes noise**,
- but long pulse induces **loss of time resolution** $\Rightarrow$ cross-correlation is a broad peak
- strong variation of code over time $\Rightarrow$ increased bandwidth B $\Rightarrow$ cross correlation peak width $1/B$
- for digital systems: $B \times T = N$ the number of bits in the pseudo-random code sequence
- SNR improvement by $10\log_{10}(N)$ (e.g. $N = 1023 \Rightarrow 31$ dB for GPS L1)

pulse compression ratio (PCR) $= B \cdot T$





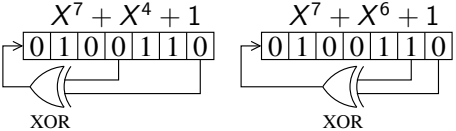

```
time=[0:1e-6:1e-2]; %samp. rate=1 us
x=chirp(time,1e3,time(end),1e3);
noise=20*rand(length(x),1)';
noise=noise-mean(noise);
xx=xcorr(x,x); xb=xcorr(x,noise);
plot(xx,'b-');hold on;plot(xb,'r-');

x=chirp(time,1e3,time(end),5e3);
xx=xcorr(x,x); xb=xcorr(x,noise);
plot(xx,'k-');hold on;plot(xb,'m-');

for N=[10 100 1000 10000]       %  10       100     1000    10000
   x=rand(N,1);x=x-mean(x);
   var(x,1)                     % 0.0553 0.0759 0.0819  0.0844
   max(abs(xcorr(x)))           % 0.5528 7.5949 81.931  844.30
   max(abs(xcorr(x)))/var(x,1)  % 10.000 100.00 1000.0  10000.
end
```

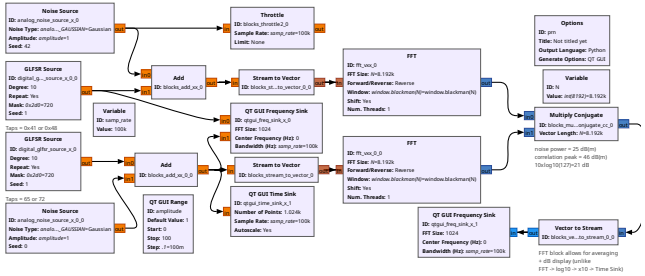
Pseudo-random code sequence orthogonality

- Code Division Multiple Access: each messaege bit is encoded as a sequence of **known** PRN chips
- $xcorr(c_i, c_j) = \delta_{i,j} = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i == j \end{cases}$
- (Galois) Linear Feedback Shift Registers (LFSR) with wisely [a] selected XOR taps



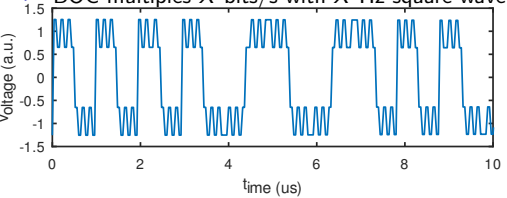
- Rationale approach to reproduce a local copy of the Pseudo-Random Number (PRN) sequence on the receiver

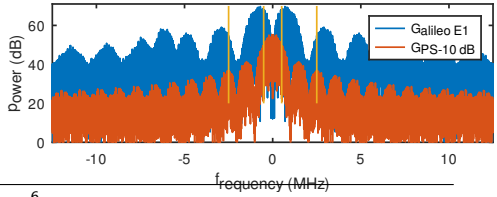
[a] https://users.ece.cmu.edu/~koopman/lfsr/

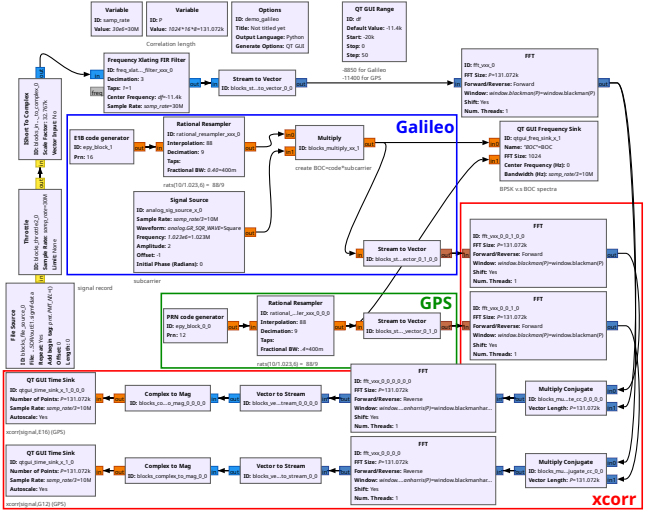


GPS L1 BPSK and Galileo E1 BOC decoding

- Both GPS L1 and Galileo E1 are broadcasting in the same 1575.42 MHz band
- How to cohabit? Move E1 energy away from carrier (Binary Offset Carrier)
- A BPSK at X bits/s exhibits nulls at $\pm X$ Hz
- BOC multiples X bits/s with X Hz square wave[6]







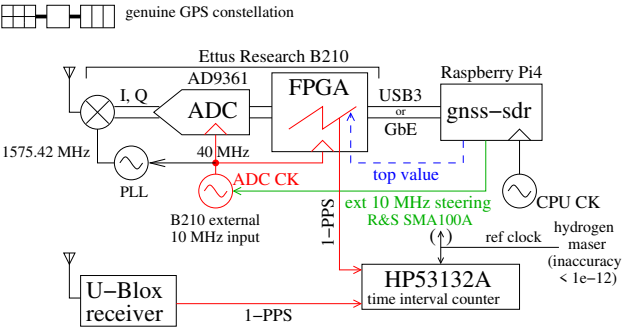
[6] Galileo open service signal-in-space interface control document (OS SIS ICD) at https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_OS_SIS_ICD_v2.1.pdf

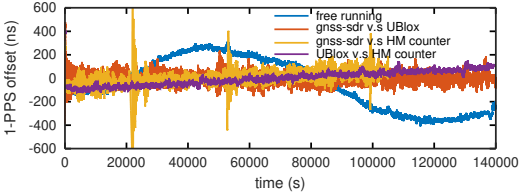
[6] GNSS codes: https://github.com/danipascual/GNSS-matlab/tree/master/prn_codes

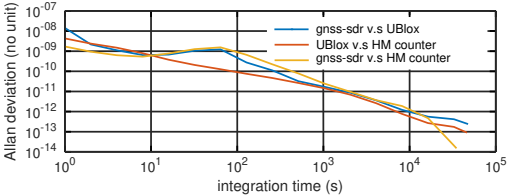
[6] GNSS code generators in Python: https://github.com/pmonta/GNSS-DSP-tools/

1-PPS generation from SDR based GNSS receiver (gnss-sdr)

- distributed timing: only the ADC can **timestamp samples**, all further processing is asynchronous





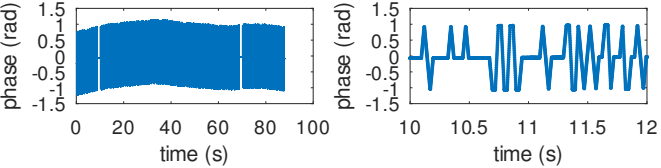


- gnss-sdr [7] provides a PVT solution with the time delay between a local copy of the PRN sequence and the received signals ⇒ control the clock feeding FPGA+ADC accordingly.
- Monitor UDP port for distributed acquisition and processing by receiving all internal states of the PVT solver.

[7] https://gnss-sdr.org/ and our fork at https://github.com/oscimp/gnss-sdr-1PPS

VLF time and frequency transfer

- GNSS is highly sensitive to **spoofing** and jamming (50 W microwave signal broadcast from 20000+ km)
- VHF/UHF ground based emitter communication range limited by Earth curvature
- Very Low Frequency (VLF) benefit from ionospheric waveguide propagation for long range communication
- MSF (60 kHz UK), WWVB (60 kHz USA), JJY (60 kHz Japan), DCF77 (77.5 kHz Germany)[8], eLORAN (100 kHz), ALS162 (162 kHz France)[9], CZAS (225 kHz Poland)[10]
- readily available from WebSDR[11] recordings



ALS162 phase modulation

```
filename='websdr_recording_start_2025...';
fs=audioinfo(filename).SampleRate;
x=audioread(filename);
t=[0:length(x)-1]'/fs;   % discrete time
lo=exp(j*2*pi*t*(750));  % CW is 750 Hz off:
xlo=x.*lo;               % coarse freq offset
b=firls(256,[0 200 400 fs/2]*2/fs,[1 1 0 0]);
xlob=filter(b,1,xlo); xlob=xlob(256:end);
xlob=xlob(1:5:end);   fs=fs/5; % decimate
[a,b]=polyfit([0:length(xlb)-1]/fs,unwrap(→
    ↪angle(xlb)),1);  % fine freq. offset
```

[8] J.-M Friedt & *al.*, *Software defined radio decoding of DCF77: time and frequency dissemination with a sound card*, Radio Science **53**(1) 48–61 (2018)

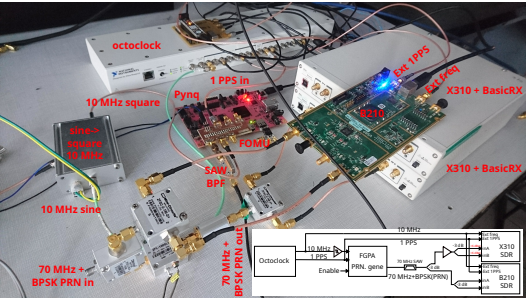
[9] H. Maier, *ALS162 Time Signal SDR Receiver for GNU Radio*, Proc. GNU Radio Conference (GRCon) 2023 at https://pubs.gnuradio.org/index.php/grcon/article/view/134

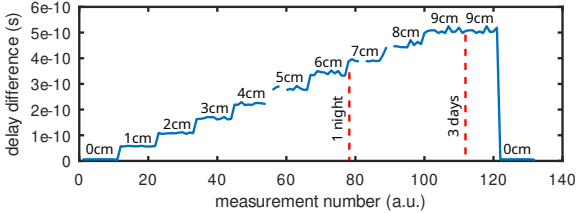
[10] https://e-czas.gum.gov.pl/wp-content/uploads/2024/06/e-CzasPL-Opis-ramki-czasu-e-Czas-Radio.pdf

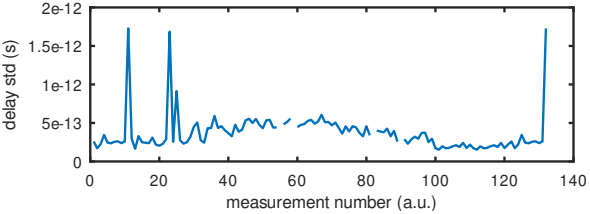
[11] http://websdr.ewi.utwente.nl:8901/

Timing with sub-sampling period resolution

- Correlation peak samples are separated by T_s the sampling period
- Use the neighbours of the peak for polynomial fit
- Analytical solution: $dt = \frac{T_s}{2} \times \frac{y_3 - y_1}{y_3 + y_1 - 2y_2}$ with y the correlation magnitude
- Similar to Early/Prompt/Late DLL tracking loop except for the $-2y_2$
- dt uncertainty given by y SNR, improved by PCR[a]







[a] J.-M Friedt, Proc. GNU Radio Conference (2023)

Drawbacks of digital: floating point number representation

Be aware of digital calculation limitations, especially with floating point number representation

- Discrete time: `t=[0:∞]'/f_s;` at sampling rate f_s
- Numerically controlled oscillator [12]: `lo=exp(j*2*pi*f*t);`
- Frequency transposition: `st=s.*lo;`

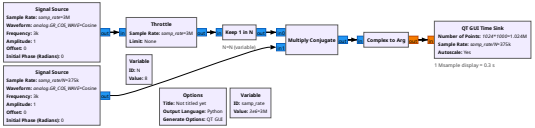
Example: keep trigonometric arguments in the $[-\pi : \pi]$ range where precision is maximized [13]

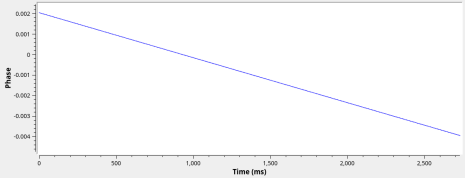
$$cos(2\pi(f/8) \times t) \neq cos(2\pi f \times (t/8))$$

```
// increment current phase angle
void step(int n = 1)
{
  phase += phase_inc * n;
  if (fabs(phase) > GR_M_PI) {
      while (phase > GR_M_PI)
        phase -= 2 * GR_M_PI;

      while (phase < -GR_M_PI)
        phase += 2 * GR_M_PI;
  }
}

void nco<o_type, i_type>::sin(float* output,int noutput_items,double ampl)
{
    for (int i = 0; i < noutput_items; i++) {
        output[i] = (float)(sin() * ampl);
        step();
    }
}
```



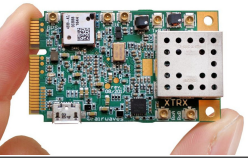


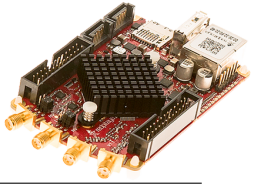
[12] transpose time to make a vector, or make sure to transpose `lo` with `.'` to avoid using the complex conjugate of `lo`

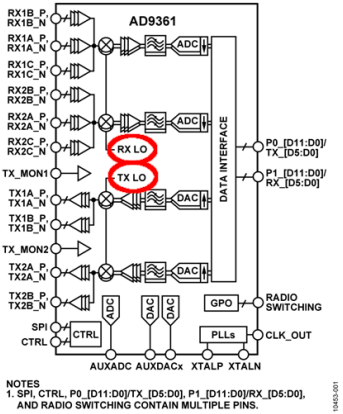
[13] https://github.com/gnuradio/gnuradio/blob/master/gnuradio-runtime/include/gnuradio/nco.h#L50

Hardware

- Baseband: Red Pitaya (14 bits, 16 bits with aliasing [14])
- Open: HackRF, BladeRF, LimeSDR, Ettus Research hardware
- Cost: DVB-T receivers (<10 euros/$), ADi PlutoSDR AD936x **input and output are not coherent !** (different LO)
- VLF: computer sound card
- Size: Fairwaves XTRX (30 × 51 mm), Enjoy Digital M2SDR
- General purspose: radiofrequency grade oscilloscope! (discontinuous stream but matches the ideal SDR definition)[15]
- Tradeoff between bandwidth, resolution and cost







https://www.crowdsupply.com/fairwaves/xtrx
https://www.redpitaya.com/
https://www.andoer.com/p-v1315.html

[15] E. Richter, *Usage of higher order Nyquist Zones with direct sampling Devices*, Software Defined Radio Academy (SDRA) 2020 at https://www.youtube.com/watch?v=PI_ROLXqO_Q

[15] https://github.com/jmfriedt/gr-oscilloscope38

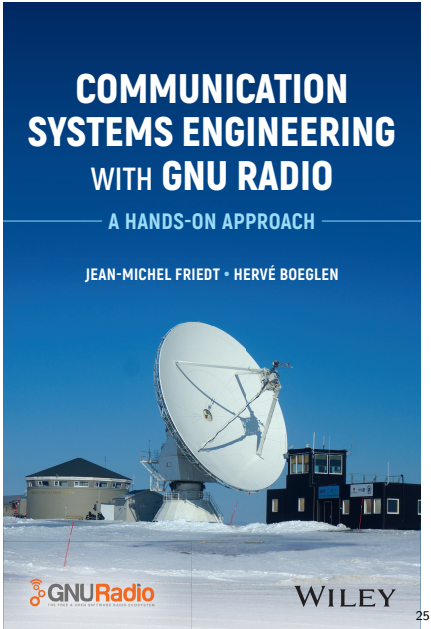
Conclusion

Software Defined Radio for time & frequency generation and dissemination

- FOSS software infrastructure: GNU Radio
- benefits of stability, flexibility and reconfigurability
- invest in hardware once, deploy for most investigations by tuning software
- carrier frequency is cancelled early by RF frontend: only bandwidth (sampling rate) matters for SDR
- challenging software combination (FPGA HDL, GP-CPU C++/Python, user interface & networking)
- issue of time delay in closed loop control systems ($\varphi = 2\pi f \times \tau_N$ phase shift at frequency f with $\tau_N = N \cdot T_s$ time delays at sampling rate T_s introduced by discrete time processing)
- GNU Radio available in Buildroot for embedded systems[16](e.g. Raspberry Pi 4/5: quad-core & USB3 and even PCIe for > 60 MHz bandwidth) →

SDR for research, development and **educational & training purposes** at the intersection between computer science, radiofrequency and digital signal processing (DVB-T dongles <10$)

[16] G. Goavec-Merou & J.-M Friedt, *Never compile on the target ! GNU Radio on embedded systems using Buildroot*, FOSDEM 2021 at `https://fosdem.org/2021/schedule/event/fsr_gnu_radio_on_embedded_using_buildroot/`

Selected bibliography

1. T. Collins & *al.*, *Software-Defined Radio for Engineers*, (2018) at `https://www.analog.com/en/education/education-library/software-defined-radio-for-engineers.html`
2. S.W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing, 2nd Ed* (1999) at `https://www.dspguide.com/pdfbook.htm`
3. R.G. Lyons, *Understanding Digital Signal Processing*, Prentice Hall (2004)
4. T. McDermott, *Wireless Digital Communications : Design and Theory*, Tucson Amateur Packet Radio Corporation – TAPR (1997)
5. J.G. Proakis, D.K. Manolakis, *Digital Signal Processing*, Prentice Hall (2006)
6. A.V. Oppenheim, R.W. Schafer, *Discrete-Time Signal Processing (3rd Edition)*, Prentice-Hall Signal Processing Series (2009), and videos of his lectures at `ocw.mit.edu/resources/res-6-007-signals-and-systems-spring-2011/video-lectures/lecture-1-introduction/`
7. K. Borre, D.M. Akos, N. Bertelsen, *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*, Birkhäuser (2007)
8. K. Borre & *al.*, *GNSS Software Receivers*, Cambridge University Press (2023)
9. E.D. Kaplan, C. Hegarty, *Understanding GPS: Principles and Applications, 2nd Ed.*, Artech House (2005)
10. Principles of Digital Communications course at `ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-450-principles-of-digital-communications-i-fall-2006/video-lectures/`
11. Yearly conferences: GNU Radio Conference (GRCon), European GNU Radio Days and FOSDEM Free Software Radio devroom