Redpitaya ADC characterization, January 18, 2021

1 Connecting to the Redpitaya

The ssh server `dropbear` is currently not functional on the Redpitaya, so that `ssh` and `scp` **cannot** be used ! Please connect to the Redpitaya with `minicom` and transfer files through NFS as the home directory on the host computer can be exported to the Redpitaya.

2 Data acquisition

Two bitstreams are provided, `adcChanADmaDirect_single.bit.bin` and `adcChanADmaDirect_dual.bit.bin` for single channel measurement or dual channel synchronous measurements. The devicetree overlay requires that the bitstream is called `adcChanADmaDirect_wrapper.bit.bin` so rename (`cp`) the wanted bitstream accordingly.

Launch once `adcChanADmaDirect_us.sh` to initialize the Redpitaya, namely by loading the kernel module `data_dma_direct_core` copying the bitstream to the right location in the GNU/Linux tree structure, and loading the bitstream in the FPGA. All these steps are automated thanks to the `adcChanADmaDirect.dtbo` devicetree overlay.

Once the Redpitaya has been initialized, the application `adcChanADmaDirect_us` will acquire data through the DMA, 1 Msamples for a single channel or 500 ksamples/channel interleaved for two channels. The output filename is `dump.bin`, to be copied to the host PC through NFS.

3 Data processing

The data are stored in binary format, signed 16-bit integers. With GNU/Octave: for a single channel acquisition

```
f=fopen('dump.bin');d=fread(f,'int16');
f=linspace(-125/2,125/2,length(d));
plot(f,abs(fftshift(fft(d-mean(d)))))
```

and for a dual channel acquisition

```
f=fopen('dump.bin');d=fread(f,'int16');
d1=d(1:2:end);
d2=d(2:2:end);
f=linspace(-125/2,125/2,length(d1));
plot(f,abs(fftshift(fft(d1-mean(d1)))))
```

4 Time v.s frequency

Energy must be conserved, whether data are displayed in the time domain or the frequency domain (Parseval theorem analyzed from its energy conservation perspective by Rayleigh). Indeed, as found in Lord Rayleigh, *On the character of the complete radiation at a given temperature*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **27**(169), 460–469 (1889),

The following is an adaptation of Stokes's investigation * of a problem in diffraction.

By Fourier's theorem (9) we have

$$\pi \cdot \phi(x) = \int_0^\infty f_1(u) \cos ux\, du + \int_0^\infty f_2(u) \sin ux\, du, \quad . \quad . \quad (16)$$

where

$$f_1(u) = \int_{-\infty}^{+\infty} \cos uv\, \phi(v)\, dv, \quad . \quad . \quad . \quad . \quad (17)$$

$$f_2(u) = \int_{-\infty}^{+\infty} \sin uv\, \phi(v)\, dv. \quad . \quad . \quad . \quad . \quad (18)$$

In order to shorten the expressions, we will suppose that, as in (11),

$$f_2(u) = 0.$$

We have

$$\pi^2 \cdot \{\phi(x)\}^2 = \int_0^\infty \int_0^\infty f_1(u) f_1(u') \cos ux \cos u'x\, du\, du'.$$

Thus

$$\int_{-\infty}^{+\infty} \{\phi(x)\}^2 dx = \frac{1}{\pi} \int_0^\infty \{f_1(u)\}^2 du. \quad . \quad . \quad . \quad (20)$$

If $f_2(u)$ be finite, we have, in lieu of (20),

$$\int_{-\infty}^{+\infty} \{\phi(x)\}^2 dx = \frac{1}{\pi} \int_0^\infty \left[\{f_1(u)\}^2 + \{f_2(u)\}^2 \right] du. \quad . \quad (21)$$

...

The energy of the signal in the time domain in $\sum x_k^2$ computed in GNU/Octave as `sum(x.^2)`

The challenge in the frequency domain lies in properly normalizing the spectra. Many presentations on the web explain how to correctly normalize power spectra [1], so we will use the readily signal processing tool `pwelch` [2] here

[1] e.g. S. Hageman, *Real spectrum analysis with Octave and MATLAB*, EDN (2015) at `https://www.edn.com/real-spectrum-analysis-with-octave-and-matlab/`

[2] O.M. Solomon, *PSD Computations Using Welch's Method*, Sandia Report (1991) at `https://www.osti.gov/servlets/purl/5688766/`

```
pkg load signal

N=32768;
fs=1;
x=rand(N,1);x=x-mean(x);
window=rectwin(N);
% default is for pwelch to REMOVE the mean value
[pxx,f]=pwelch(x,window,0,N,fs,[],'no-strip');  % W/Hz
subplot(121);plot(f,10*log10(pxx));        ylabel('dBW/Hz')
subplot(122);plot(f,10*log10(pxx*fs/N));  ylabel('dBW/bin')
sum(pxx)
```

demonstrates that indeed `sum(pxx)` is equal to `sum(x.^2)`. Notice that `pwelch` is also available for SciPy as `scipy.signal.pwelch`

For plotting histograms, `hist()` can be used. The excellent `peakfit.m` at `https://terpconnect.umd.edu/~toh/spectrum/InteractivePeakFitter.htm` is used for Gaussian peak fitting. After recording the bin index and number of samples per bin, peak fitting is achieved with

```
pkg load signal
f=fopen('dump.bin');d=fread(f,'int16');
d1=d(1:2:end);d1=d1-mean(d1);
d2=d(2:2:end);d2=d2-mean(d2);
f=linspace(-125/2,125/2,length(d1));
[hh,xx]=hist(d1,floor(max(d1)-min(d1)));
signal=[xx' hh'];
[FitRes,GOF,bl,coeff,res,xi,yi,BootRes]=peakfit(signal,0,0,1,1,0,0,[0 max(xx)],0,0,0);
```

where `FitRes` holds the mean value and width of the peak in its second and fourth attributes respectively, and the result of the fit is in `(xi,yi)`.

Fig. 1 displays an example of such a processing on experimental data with the ADC loaded by a 50 Ω resistor. The case of a sine wave is shown in Fig. 2

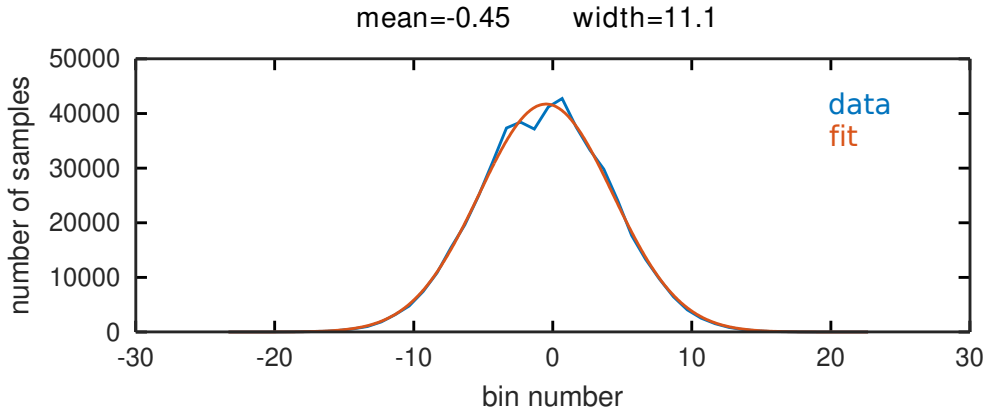

Figure 1: Fit of the bit value distribution of the ADC loaded on a 50 Ω resistor (noise).

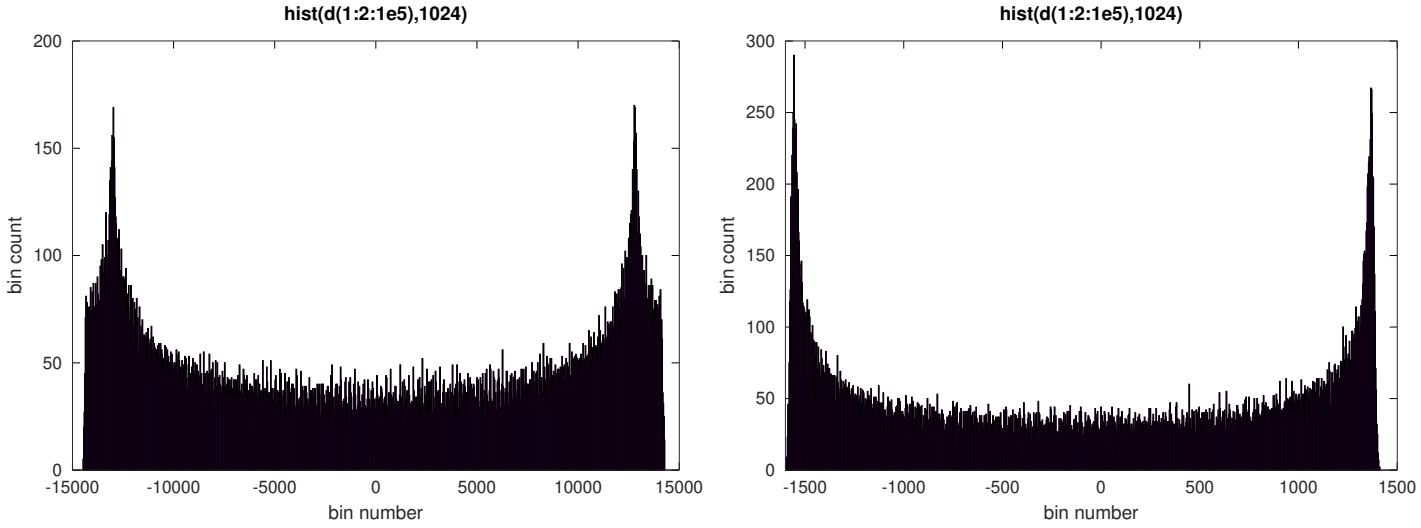

Figure 2: Histogram of an undersampled 250 MHz (left) and 850 MHz (right) sine wave sampled at 122.88 S/s with the 16-bit Redpitaya.