zed-audio-dsp

Guillaume W. Bres

July 16, 2020

Contents

1 System
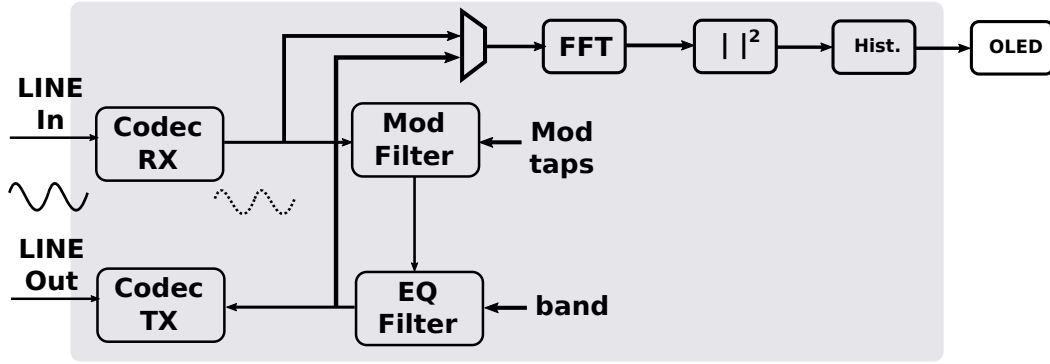
Bloc design

All processing is real time and synchronous to the input stream. The initial data rate is 1.152 Gb/s (much more than needed), we reduce it to a 24 kHz (a little more than needed), which means our internal rate is now 576 kB/s. Data rate reduction is done using a CIC decimation filter.

The modulation filter as well as the EQ filter are controlled from the User Interface.

The FFT processes either the direct RX input or the output signal. This allows to visualize the effect of the modulation & the EQ filters. Switching is done in real time on user specs.

The resulting power spectrum is converted to histogram & displayed on the onboard OLED display. The display is 128x32 pixel wide, which in our case gives a spectral resolution of 187.5 Hz on the display, which is more than decent.
The internal data stream obviously needs a final interpolation, this is done by a CIC interpolation filter, feeding a 1.152 Gb/s stream to the TX side. The CIC interpolation filter is strictly symmetrical to the CIC decimation filter.

1.1 CIC filters

CIC filter being defined by R, M, N parameters where

- R: decimation/interpolation factor

- M: time delay, can either be 1 or 2 cycle

- N: number of stages

To reduce the initial rate from 1.152 Gb/s we fix $R = 128$.

Both the interpolation & decimation filters are comprised of a N stages of integration filters & comb filters.

- an integrator stage is defined as $H(z) = \frac{1}{z^{-1}}$.

- a comb stage is defined as $H(z) = z^{-M}$.

- total CIC filter response is therefore $H(z) = \left| \frac{z^{-M}}{z^{-1}} \right|^{N}$

Increasing N (number of stages) increases the filter performance, therefore the maximum alias rejection encountered at frequency ...

We fix $N = 8$ in our case, not to consume too much ressources. This will give a rejection of -dB for the worst alias ever encountered.

A direct CIC filter response, in this example R=4,M=1, and we compare N=4,N=8,N=12,N=16,N=32, has a $f(x) = \frac{sin(x)}{x}$ shape. Which is far from being convenient because it dramatically disturb the initial flat distribution. We use a compensation filter to compensate for this effect, the compensation filter response is the exact inverse of the CIC filter response $G(z) = \frac{1}{H(z)}$ in the output frequency domain.

CIC filter compensation is required and will be implemented in both the decimation & the interpolation filter, using Xilinx's optimized FIR filter IP core.

The `$git/dsp/cic.py` Python script can plot the frequency response of a given CIC filter

```
$git/dsp/cic.py R=4 M=1 N=12
$git/dsp/cic.py R=8 M=2 N=4
```

1.2 CIC compensation filter

The CIC compensation filter is designed using the python script `$git/dsp/cic-compensator.py`.

```
$git/dsp/cic-compensator.py R=4 M=1 N=12
$git/dsp/cic-compensator.py R=8 M=2 N=4
```

1.3 FFT Histogram

The Histogram IP core converts the power spectrum from the magnitude IP core to a histogram to be displayed on the OLED.

It is a simple mapping of the resulting power spectrum to a 2D array (X,Y) describing the power spectrum along a frequency axis.

2 Simulation

Install `ghdl` to easily simulate modules that come with a testbench:

```
git clone https://github.com/ghdl/ghdl
cd ghdl
./configure
make
sudo make install
```

Once this is done, you can safely simulate a module that comes with a testbench by using *make* in its `sim` folder. For example:

```
cd $git/ip/adau1761/sim
make
```