TangerineSDR

TangerineSDR

FPGA Processing of Chirp Signals

White Paper

Version Number: 0.1

Version Date: July 9, 2020

Document Number:

VERSION HISTORY

Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
0.1	T. McDermott	July 9, 2020			Original Issue

1. Introduction

One of the key measurements desirable from TangerineSDR is processing of oblique ionosound data.

Typically there are several chirp transmitters with well-known parameters:

- Start frequency
- Start time
- Chirp rate (kHz / second)
- Stop frequency

Traditionally all-software processing of these signals is accomplished by providing a high-bandwidth capture of the entire received RF spectrum within the frequency range of the chirp. Then that signal is post-processed to create the oblique ionogram.

However this requires a tremendous amount of bandwidth between the receiver and the signal processing device, and thus is practically limited to a few locations with the necessary interconnection hardware.

The TangerineSDR has two receivers each operating at a sample rate of 122.88 Ms/s. Assuming simple packing of 14-bit data into two bytes this equates to a bandwidth of:

122.88 * 16 (bits/sample) * 2 (receivers) = 3.982 gigabits / sec.

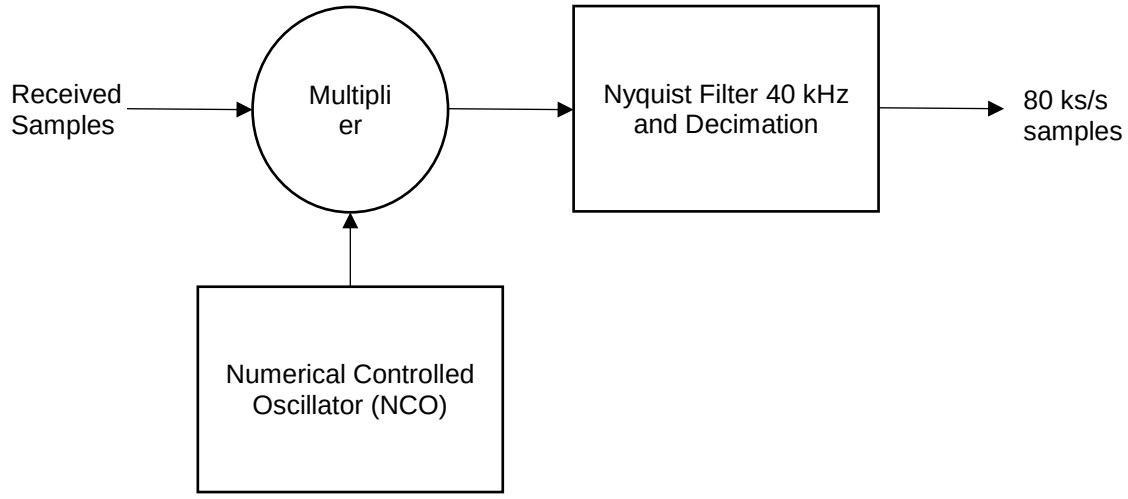
Due to overhead (Ethernet packetization, etc.) the actual rate would be a bit higher. The bandwidth could be halved to cover just HF, resulting in about 2 gigabits/sec of data. The current Data Engine has a 1 GbE interface, and so passing along the data would be problematic in most cases.

However the actual usable information is contained in a fairly narrow spectrum tracking near the chirp itself. Typically the spectrum of interest is about 40 kHz after de-chirping the received signal. Thus processing of the received signal within the TangerineSDR Data Engine (DE) FPGA could substantially reduce the amount of information that needs to be sent upstream to be data-logged for researchers to further process. The reduction of the data needed could also allow home / personal users of the equipment able to send data to researchers with available home Internet uplink speeds.

The rest of this paper examines the nature of the signal processing needed, and discusses some algorithms that could be used within the FPGA on the data engine to make this practical, and to reduce the captured information very substantially.

2. De-Chirping

Because the exact parameters of the transmitter chirp signals are known, the DE can create the same chirp signal locally, referenced to GPS time. The DE then would mix the receive spectrum against a chirped Local Oscillator (LO) rather than the traditional fixed LO used by a standard receiver. The functionality of mixing, filtering, and decimation is exactly the same between the two cases. The de-chirping in the FPGA is almost the same at the standard receive functionality, except that a chirped LO is used instead of a fixed-frequency LO.

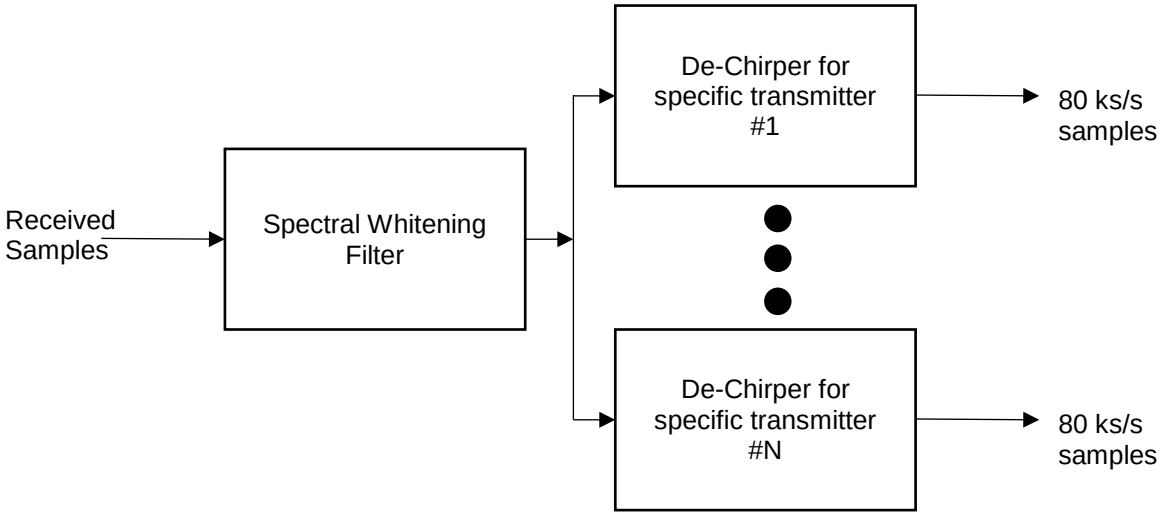


The FPGA processing required for the de-chirping thus will not be analyzed further.

3. Spectral Whitening

One of the difficulties of receiving HF chirped signals over a wide bandwidth is that there are strong fixed-frequency broadcast stations located with the frequency range of interest. These signals have the effect of creating artifacts that mask the desired received signals. One solution to this problem is the remove strong fixed frequency carriers before de-chirping the receive signal. An approach is to use a spectral whitening filter covering the wideband RF spectrum. One such filter can feed multiple different de-chirping blocks in parallel. The desired receive chirp signal does not exist within a single spectral bin long enough to contribute a significant amount of energy to that bin's moving average magnitude.

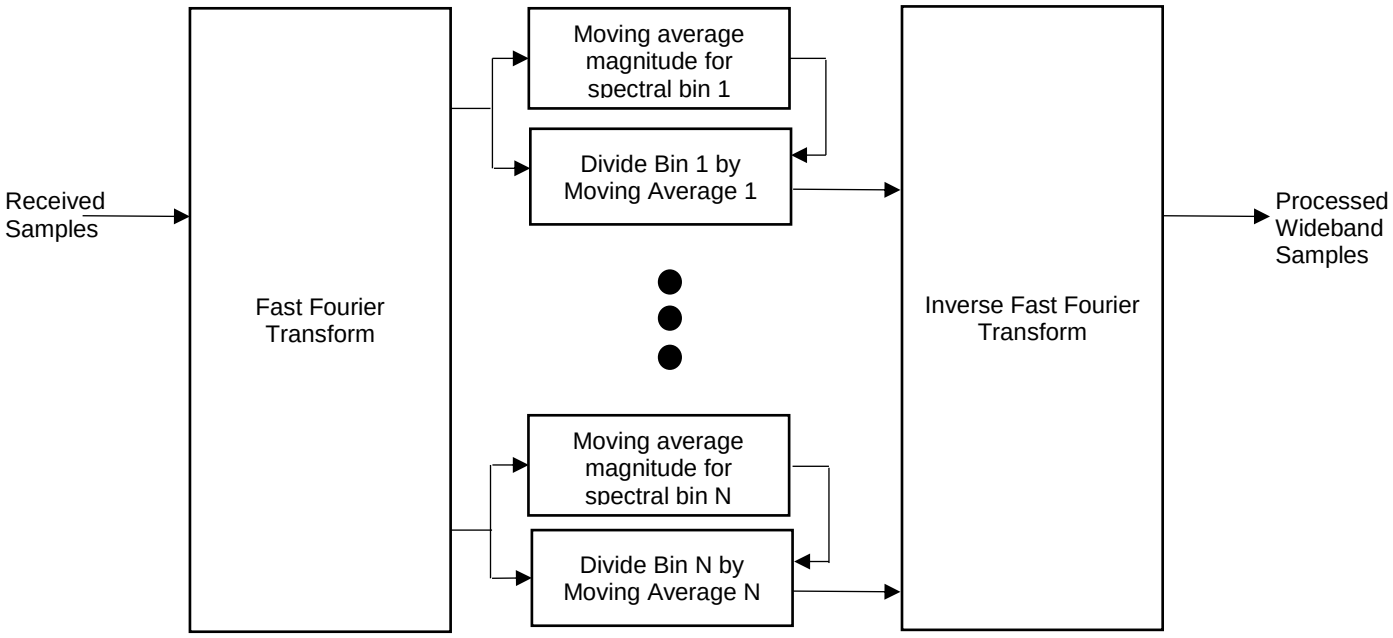
The whitening filter can be challenging to implement in an FPGA. The rest of this section will cover some possible approaches to spectral whitening in an FPGA that are optimized for processing efficiency and minimization of gate count.



Spectral whitening refers to a process where the output of the process contains signals of the same amplitude at the various frequencies in the spectra (similar to white noise). An approach is:

1. Convert the spectra to the frequency domain using an FFT
2. Create a moving average of each spectral bin with about 10 Hz averaging
3. Divide each signal in the input spectrum by its moving average.
4. Transform the resultant signal back to the time domain with an Inverse FFT.

There are several issues with trying to implement this directly in an FPGA.



The resolution of the FFT needs to be sufficient to isolate individual anticipated spectral carrier signals, such as AM broadcast signals. The input signal is sampled at 122.88 Ms/s At this point it is still a real signal and the bandwidth is 122.88 / 2 or 61.44 MHz. Thus an FFT of size 2048 would produce bins of 61.44e6 / 2048 = 30 kHz.

4. Processing for an FPGA Implementation

One key issue with implementing the Spectral Whitening filter directly within an FPGA is caused by the large number of numerical divisions required. Numerical division in an FPGA is an expensive process in terms of the large number of gates needed and the slow processing speed.

When using fixed point 2's complement binary signals, if the division can be constrained to an integer power of 2 (i.e. 2, 4, 8, 16, 32, etc.) then division can be implemented by a simple bit-shift operation, for example shifting right by one bit results in division by two. This is a very economical operation in an FPGA – simply wiring the signals from one stage to the next with an offset in the wiring.

A moving average type of magnitude filter can be implemented using an integrator. This type of integrator builds an average of the magnitude signal by subtracting a fraction of the existing integrated value from the integrator then adding a fraction of the input signal to the integrator. If the factor is constrained to be an integer power of two factor, then the filter could be implemented with bit shifts, subtraction, and addition and no divisions.

It should be noted that the samples coming out of the FFT are reduced in rate by a factor of N – For example, if the FFT is a 2048 point FFT then it produces output samples at 1/2048 of the input signal rate. The IFFT would receive parallel samples at the reduced rate, but then step up the output rate by a factor of 2048 matching the sample rate of the original FFT input signal.

4.1. Average Signal Magnitude

Since the filter is processing real signals, a real FFT can be used, and the magnitude of the signal is simply the absolute value of the signal sample from the FFT bin.

The moving average filter time domain response is given by the integration factor. Calling this factor F, for example 1024, then the operation required for the filter is

> Integrator_next = Integrator_current – (Integrator_current / 1024) + (Magnitude of Input signal bin / 1024)

The value of the integration factor F needs to be constrained to an integer power of 2. The sample rate of 122.88 Ms/s is a time step of about 8 nsec per sample.

The time response of the 1024 integrator would be roughly 8 nsec * 2048 * 1024 or about 17 milliseconds.

4.2. Signal Division

The whitening filter works by normalizing the average output of each spectral bin to unity. This is accomplished by dividing the bin value by the average of the bin value. However the actual magnitude of the output spectra is not important in this application. The purpose of the whitening is to eliminate large constant frequency signals from contaminating the output. The actual output amplitude of each bin is not so important.

Thus, instead it is possible to subtract the average of the output bin from the bin signal, rather than dividing the bin signal. Instead of the bin output averaging unity, it would average zero instead. This should not be detrimental to the actual application.

Thus the division step can instead be replaced by a subtraction step.

4.3. Net Result

The net result of these steps is that no general division is needed anywhere in the FPGA. Instead only bit shifted division, addition, and subtraction is necessary.

In this example, 2048 copies of the moving average filter and bin subtraction would be needed. However the average rate of operation of these copies would be 8 nsec * 2048 or 16.6 microseconds per sample.

Appendix

Magnitude for Complex Signals

The magnitude of a complex signal is equal to sqrt (I^2 + Q^2). These operations are too expensive to be performed directly for all of the spectral bins. Instead the magnitude of each bin can be approximated with a reasonable amount of accuracy[1] by

$$\text{Magnitude} = \max (| I | , |Q|) + 0.25 * \min (| I | , |Q|)$$

The multiplication is accomplished by a bit shift.

[1] https://dspguru.com/dsp/tricks/magnitude-estimator/