# LWA DRX C Language Simulation Report - v0.1

Johnathan York (ARL:UT)

February 1, 2008

## 1 Introduction

This document contains a short report on the Long Wavelength Array (LWA) Digital Receiver (DRX) C simulator and support files. For context, please see the DRX design document contained in [3], and the LWA Station Architecture document [1]. The goal of this simulator is to mimic in software the operation of the DRX hardware. This simulator has a variety of intended uses, including:

- early validation of the digital signal processing algorithms used in the DRX,
- separation of algorithm development from register transfer language (RTL) implementation issues,
- promoting early interoperability testing, specifically with the DAC and other downstream components, and
- promoting more extensive testing by running at 1,000X slower than realtime, rather than 1,000,000X typical for RTL simulations of this size.

In effect, the C simulation complements high-level design documentation by providing a "bit-accurate, functional interface control document" between the externally visible performance requirements and the RTL development of the DRX. That is, potential end-users of the DRX are able to experimentally validate that the DRX design will meet their needs using the C simulation, while RTL designers can validate their implementations at the bit-level against the output of the C simulation.

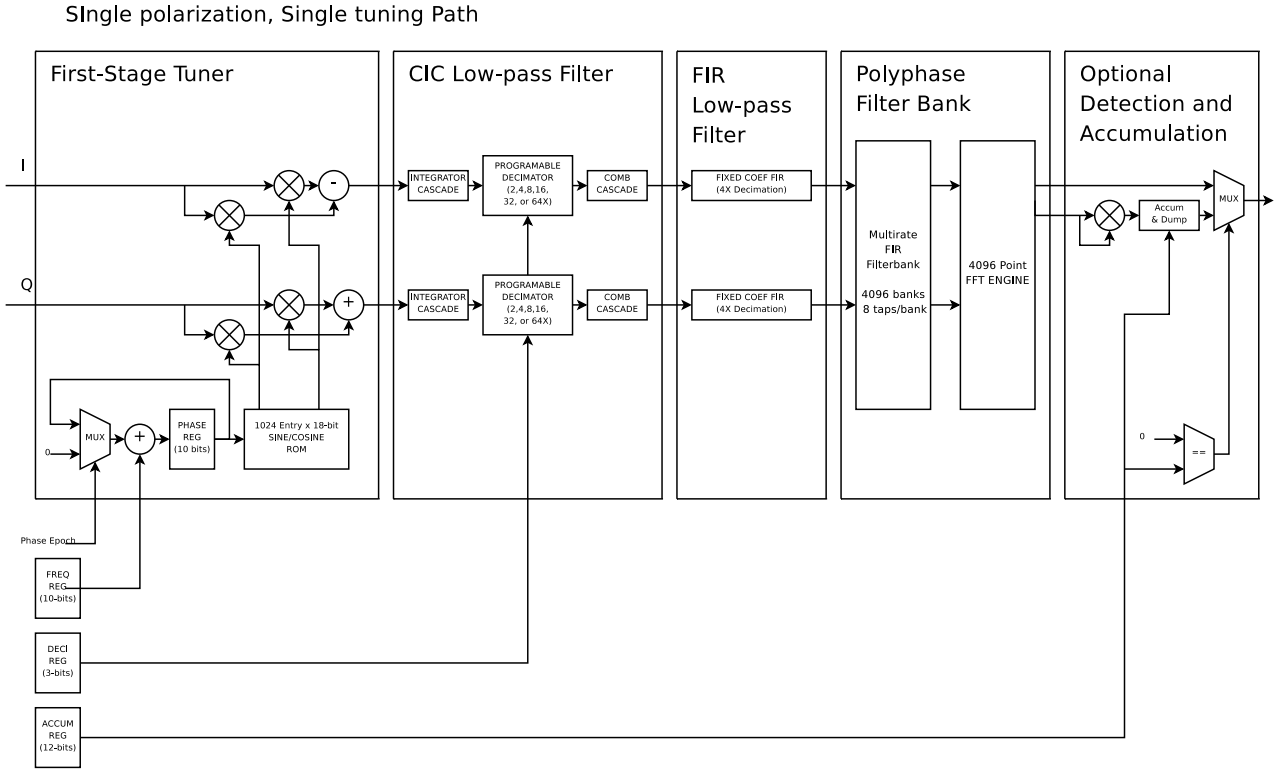


Figure 1: Block diagram of the DRX DSP components

## 2 Design

Figure 1 is a block diagram of the digital signal processing (DSP) components of the DRX. The design of the simulator closely mimics the intended hardware layout, in that separate C source files contain the logic for each module. In all there are 5 modules implementing Digital Signal Processing algorithms. These include

- *tuner.hpp* - Implements a fixed-point lookup table (LUT) based, complex-valued heterodyning mixer
- *cic.hpp* - Implements a Cascaded Integrating Comb (CIC) [4] decimating filter
- *fir.hpp* - Implements a fixed-point Finite Impulse Response (FIR) filter
- *pfb_fir.hpp* - Implements a fixed-point Finite Impulse Response (FIR) filterbank suitable for polyphase filterbank (PFB) implementation
- *fft.hpp* - Implements a fixed-point Fast Fourier Transform (FFT)

Although this document is titled as a "C language" simulation, a "C++ language" simulation is perhaps more accurate. The simulator components, as currently written, make heavy use of C++ templates to facilitate experimentation with varying bit-depths, fixed/floating-point arithmetic, and complex/real valued signals. The file *sample.hpp* provides a templated complex sample type used throughout the simulator. The file *inttraits.hpp* contains trait classes used to appropriately size accumulators, and result values based upon the input data types specified via template parameters. The heavy use of templates allows the compiler to perform substantial compile-time optimizations while preserving the data-type and data-flow agnostic characteristics of the simulator components themselves.

## 3 Interface

While the simulator components are modularized C++ classes, the user-visible interface is provided through 5 executable files, each one testing a specific subset of the DRX components:

- *ddc_testbench* - Tests the tuner, CIC, and fir components. Accepts 16-bit I&Q time-domain samples, outputs 16-bit I&Q time-domain samples.

- *pfbfir_testbench* - Tests Polyphase FIR. Accepts 16-bit I&Q time-domain samples, outputs filtered 16-bit I&Q time-domain (but polyphase) samples.
- *fft_testbench* - Tests the FFT module. Accepts 4096 32-bit I&Q time-domain samples, outputs 4096 32-bit I&Q frequency-domain samples.
- *pfb_testbench* - Test the Polyhase FIR and FFT modules. Accepts 16-bit I&Q time-domain samples, outputs 4096 interleaved 32-bit I&Q frequency channels.
- *drx_testbench* - Tests the entire DRX module. Accepts 16-bit I&Q time-domain samples, outputs 4096 interleaved 32-bit I&Q frequency channels.

The general philosophy is that each of these executables reads binary data from stdin, and writes output binary data to stdout. As noted above, time-domain data is input as signed 16-bit I&Q samples. That is, each sample consists of 4 bytes: 2 for the in-phase component and 2 for the quadrature component. The output of the DRX module are blocks of 4096 samples, each sample consisting of 8 total bytes (4 bytes for the 32-bit signed in-phase component and 4 bytes for the 32-bit signed quadrature component). Each of 4096 samples within a block contains the data for one frequency channel for a single time step.

The drx_testbench module accepts command line parameters to set the various DRX runtime configurable parameters outlined in [3]. In addition to the C++ simulator, sample Python code leveraging the numpy and matplotlib modules are provided to exercise each executable and produce graphical output.

# 4 Validation

While the C simulation is merely an intermediate product of the DRX development effort, it is nevertheless a useful exercise to examine how the C simulation matches up with the LWA technical requirements specified in [2]. Table 1 captures the LWA technical requirements relevant to the DRX C simulation. Due to the deterministic nature of digital signal processing techniques, most of the LWA technical requirements relevant to the DRX can (and must) be correct "by design". That is, proper design techniques ensure that the DRX meets the requirements. For the DRX the relevant design analysis can be found in [3].

| Number | Requirement | Value | Validation By |
|---|---|---|---|
| TR-1A/B | Frequency Range | 20-80 MHz | By Design |
| TR-2 | Instantaneous BW per beam | 8 MHz | By Design |
| TR-3 | Min channel width | 100 Hz | By Design |
| TR-5A | Min temporal resolution | 10 ms | By Design |
| TR-24A/B | RFI mitigation | Qualitative | By Design & Test |

Table 1: Technical requirements relevant to the DRX C simulation

Because many of the technical requirements must be correct "by design", experimental validation of the C language DRX simulation is limited to verification that 1) the C simulation implements the design specified, 2) effects due to fixed-point implementation are tolerable, 3) the simulation handles known DSP corner cases. The tests documented in the remainder of this section are included to give an indication that the simulation is functioning correctly based on these criteria, but the tests here are by no means exhaustive, nor do they capture the full extent of the tests performed the simulation. These tests were conducted in three phases: 1) validation of the "digital downconversion" (DDC) component (including the "first-stage tuner", "CIC low-pass filter", and "FIR low-pass filter" components shown in Figure 1), 2) validation of the remaining "channelizing" components (the "polyphase filter bank" components of Figure 1 ), and 3) integration tests of the entire simulation.

### 4.1 Digital Down-Conversion Tests

The first tests conducted on the DRX simulation validated the gross operation of the first stage tuner, consisting of the mixing, CIC, and FIR filter.

Figure 2 shows the time domain output of the first stage tuner for a single tone for varying CIC decimation factors. Note that in the $deci = 6$ case, in which the tuned bandwidth is $2^{(6-0)} = 64$ times narrower than the deci=0 mode, the tone has moved partially outside of the increasingly narrow passband.

Figure 3 shows the time domain output of the first stage tuner for a single, constant-frequency input tone for various center tuning frequency parameter. This data is taken in the $deci = 0$ mode, which provides approximately 8 MHz of bandwidth. Note the attenuation that occurs beyond the 4 MHz cutoff frequency.

Figure 4 shows the FFTed output of the first stage tuner for a white noise input applied to the DRX. This data was taken in the $deci = 0$ mode,

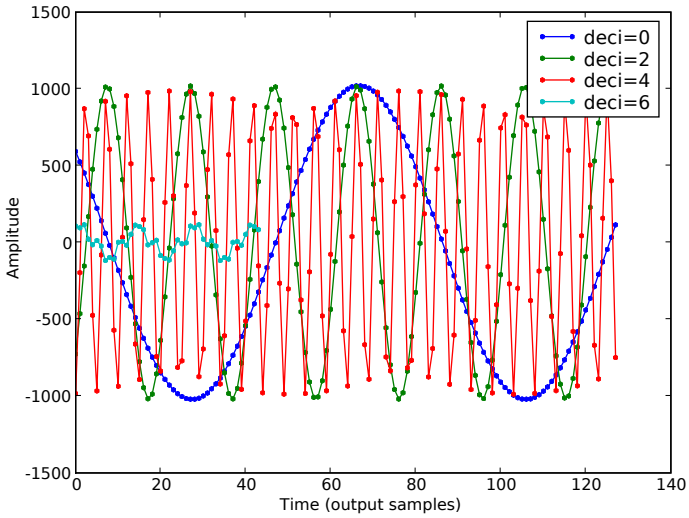


Figure 2: Time-domain output of tuner for single tone with varying CIC decimation factors

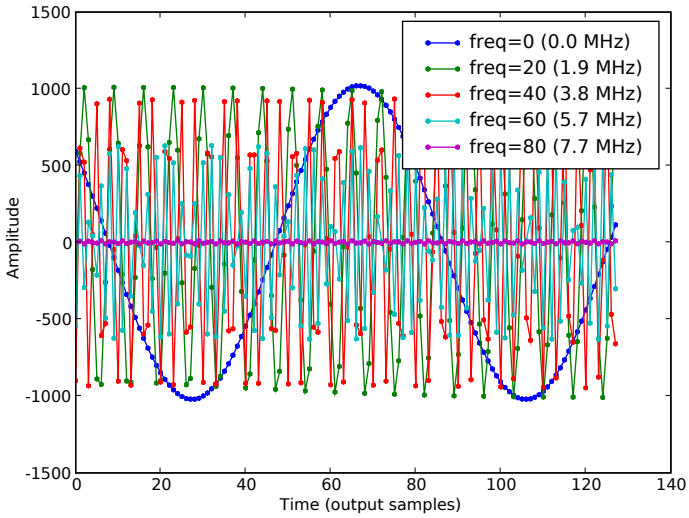


Figure 3: Time-domain output of tuner for single tone with varying frequency tunings

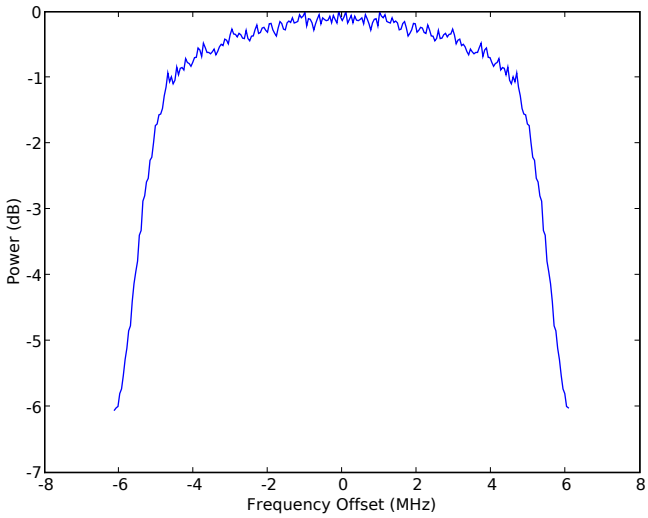


Figure 4: Incoherent average of 4096 FFTs of the first stage tuner output with white noise applied to the DRX input

which provides approximately 8 MHz of usable passband. Note that the passband is within 4 MHz of the center frequency, and that the noise power shown does not directly correspond to the out of band rejection values for the filter.

### 4.2 Channelizer Tests

The fine channelizer component of the DRX is composed of a polyphase FIR filterbank followed by an fixed-point fast Fourier transform (FFT) module.

Figure 5 shows the amplitudes for 8 of 4096 phases of the polyphase FIR component when the input is excited by a 4096-sample length $rect()$ signal. The $rect()$ signal length is equal to the number of phases of the polyphase FIR, and so each phase of the filterbank sees a unit impulse. The 8 phases shown are evenly spread across the entire filterbank, such that evidence of the interpolating effect of the polyphase response is evident across the channels.

Figure 6 shows the output of the fixed-point FFT module when excited by a single value of 16 in input bin 4. The values from a reference floating-point FFT algorithm (numpy) are represented by the green traces, while the values from the fixed-point module under test are shown in blue. The

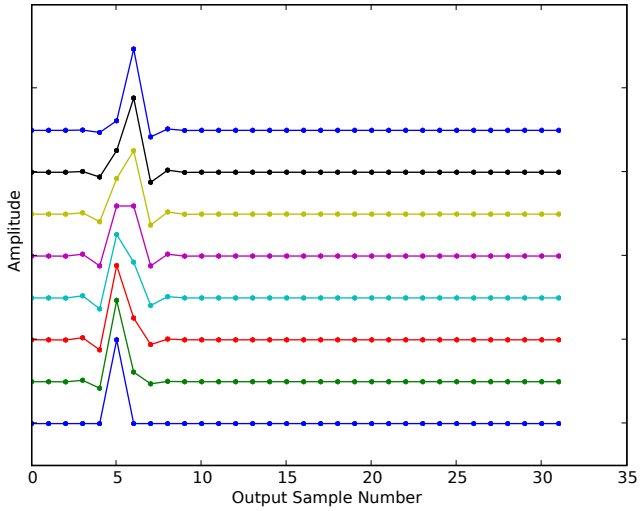


Figure 5: Output of eight (out of 4096) phases of the polyphase FIR when excited by a $rect(x)$ input

maximum deviation is 0.5 LSB units, which indicates the algorithm worked correctly in this test case.

### 4.3 Fully Integrated DRX Tests

After validating each component of the DRX simulation individually, tests of the integrated DRX simulation were performed. For these tests input data from was produced using floating-point arithmetic, then converted to 12-bit fixed-point, so as to mimic an ideal 12-bit analog to digital converter operating at 196 MSPS. Unless otherwise noted, graphs depict a single FFT, captured randomly from the DRX simulator output stream. That is, no post-process averaging has been applied to the output. Furthermore, no dither has been applied to any of the sinusoidal input signals.

Figure 7 shows the amplitude output of the DRX when excited by a -0.5 dBFS single tone aligned with the FFT frequency bins. Figure 8 shows the same results, but zoomed tightly around the FFT bins containing the tone. One FFT bin away the power level is below -75 dBc, and beyond two bins the power levels remains below -97dBc. These results are consistent with the expected results of -75dBc and -100 dBc from [3]. The internal math of the DRX is predominantly at 16 bits, which puts a theoretical bound of

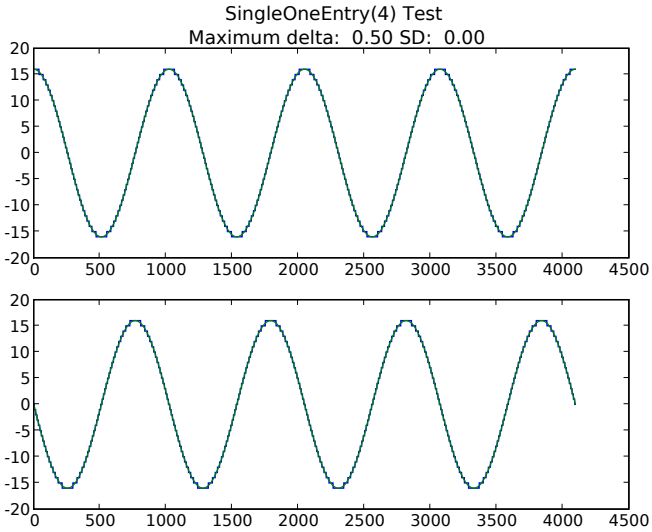


Figure 6: Real (top) and imaginary (bottom) output of the fixed-point FFT module when excited by a single value of 16 in input bin 4. The values from a reference floating-point FFT are shown in green, while data from the unit under test is shown in blue.

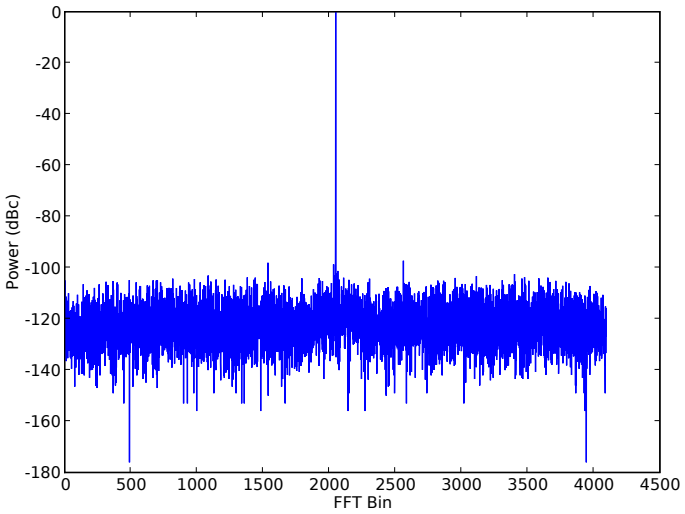


Figure 7: Single-tone, FFT Bin-Aligned Test

$1.76+6.02\times16 \approx 98$ dBc. Thus the measured -97 dBc spurious level appears to be limited by quantization effects rather than filter roll-off.

To ensure that the behavior is stable for input frequencies not related to the FFT bin width, the experiment above was repeated at offsets that are multiples of $0.2\sqrt{2}$ times the FFT bin size. The factor $0.2\sqrt{2}$ is arbitrary, but is a convenient irrational number suitable for generating test tone frequencies not harmonically related to internal DRX frequencies. Figure 9 shows the results of this test. Note the low-level, and relatively well-behaved noise away from the carrier, and the smooth transition between the two adjacent frequency bins.

For completeness, a two tone test was conducted in which two tones $100 + 0.1\sqrt{2}$ FFT bins apart at -6.5 dBFS were provided as inputs into the DRX simulator. Figures 10 and 11 show the results, which notably are absent of detectable nonlinearity-generated products.

To demonstrate out-of-band rejection, the two tone test was repeated but modified to place the two tones just out of the tuned band. The highest in-band spurious product was measured at -94.5 dBc, which is consistent with expectations for the test setup of two -6.5dBFS tones and 16-bit internal arithmetic.

A further test was conducted in which Gaussian white noise was provided

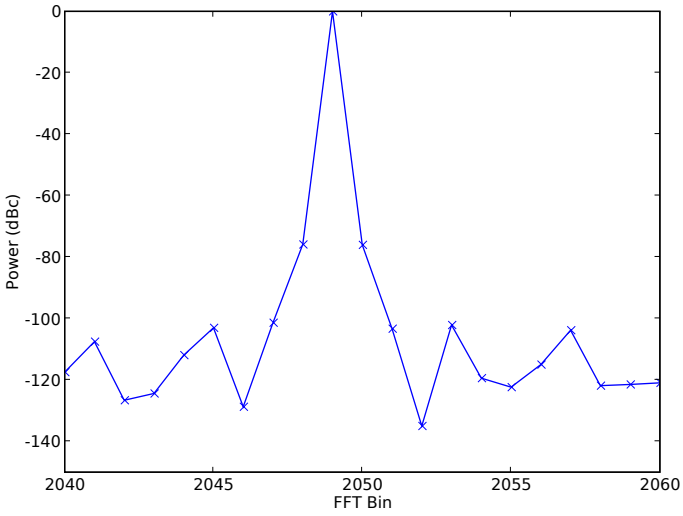


Figure 8: Single-tone, FFT Bin-Aligned Test (zoomed)

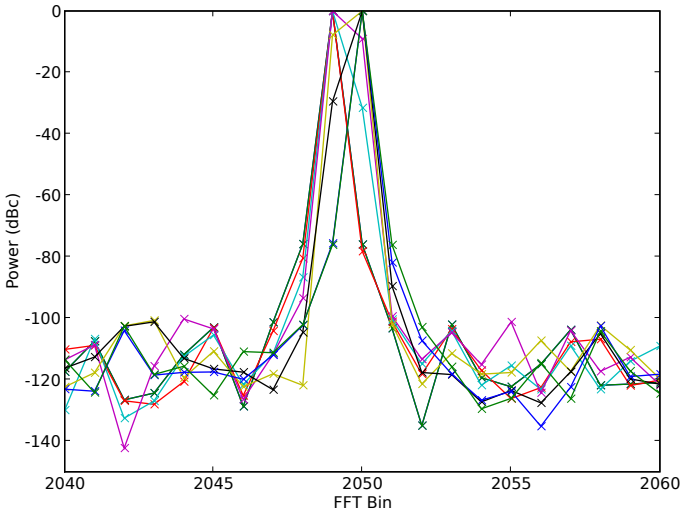


Figure 9: Single-tone tests at steps of $0.2\sqrt{2}$ times the width of an FFT bin

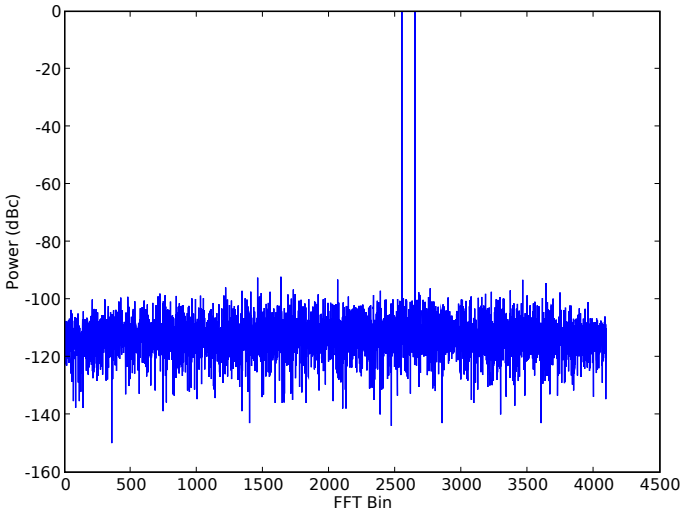


Figure 10: Two-tone Test Results

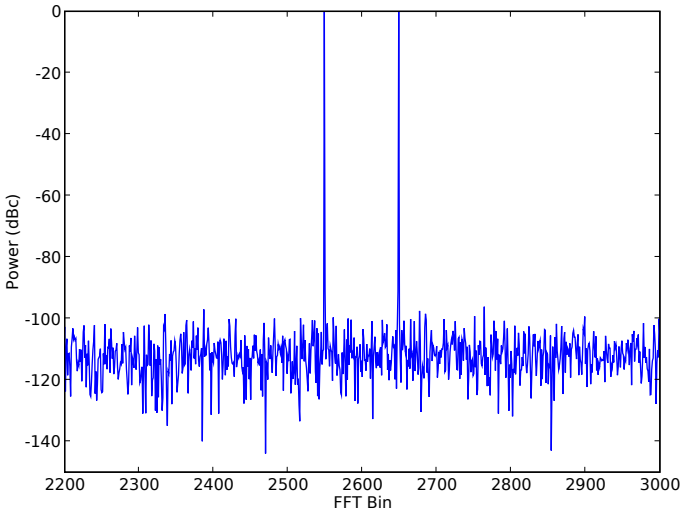


Figure 11: Two-tone Test Results (zoomed)

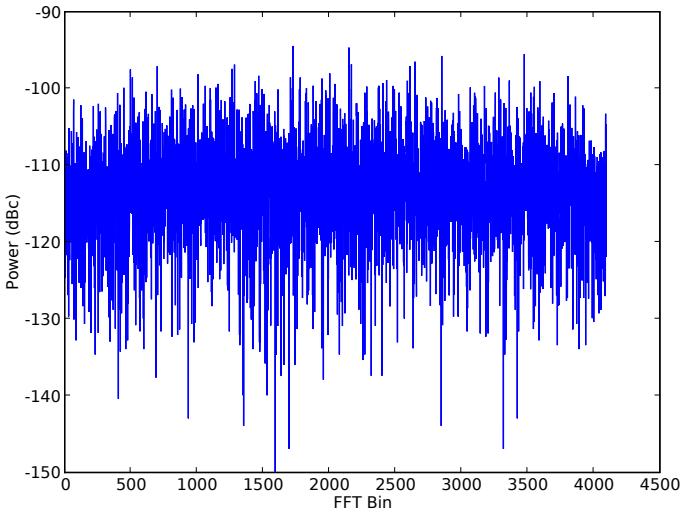


Figure 12: Out-of-band two-tone Test Results

as input to the DRX simulator, the results of which are shown in Figure 13.

# 5 Document History

- Revision 0.1 - This is the initial version of this document

# 6 Conclusion

The DRX C-language simulation provides a "bit-accurate, functional interface control document" between the externally visible performance requirements and the RTL development of the DRX. Results from the output of the simulator have been presented to verify that the simulator functions correctly in a variety of relevant test-cases.

The source code for the DRX simulator can be made available to interested parties upon request. The code is revision controlled with the GIT version control system, and patches, comments and suggestions are welcomed. The code repository also contains Python scripts to generate all the graphs in this document, and the author particularly welcomes additional test cases.

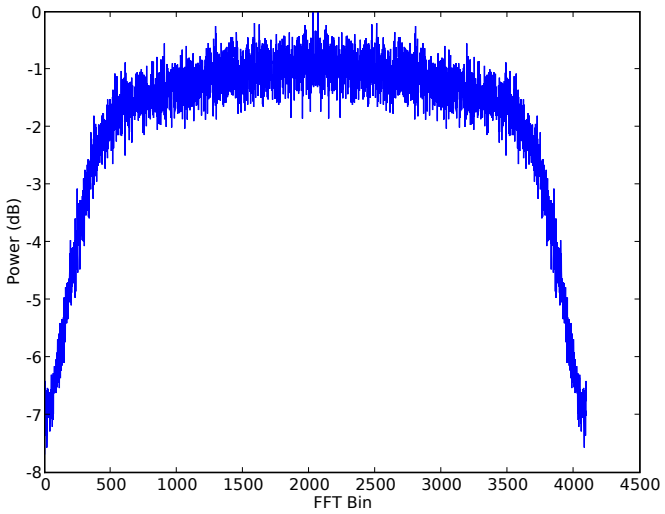


Figure 13: White noise test - average of 256 output periods

## References

[1] Ellingson, S. *LWA Station Architecture Ver 1.0,* LWA Memo 119, November 19, 2007.

[2] Janes, C. *The Long Wavelength Array System Technical Requirements Ver. "Draft #9" ,* LWA Memo 118, November 19, 2007.

[3] York, J. *LWA DRX High-level Design - v0.2,* LWA Memo 114, December, 2007.

[4] Donadio, M. *CIC Filter Introduction,* July 18, 2000.