IMPROVED THRESHOLDING TECHNIQUE FOR THE MONOBIT RECEIVER

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering

By

JONATHAN GORDON BUCK
B.S., University of Dayton, 2003

2007
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Jonathan Gordon Buck ENTITLED Improved Thresholding Technique for the Monobit Receiver BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering.

____________________
John Emmert, Ph.D.
Thesis Director

____________________
Fred Garber, Ph.D.
Department Chair

Committee on Final Examination

____________________
John Emmert, Ph.D.

____________________
Henry Chen, Ph.D.

____________________
Raymond Siferd, Ph.D.

____________________
James Tsui, Ph.D.

____________________
Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies
ABSTRACT


Wideband digital receivers typically use a very high sampling rate to capture a signal for near real-time processing. Very robust digital techniques, like the Monobit FFT technique, have been developed to provide near real-time processing of captured time-domain signals. Most of these techniques trade accuracy for processing speed by approximating computationally complex mathematical operations.

The approximation sometimes makes it difficult to detect multiple signals if the difference between their amplitudes is more than a few dB. This thesis presents a dynamic thresholding technique for multi-tone signal detection. The technique is based on setting an allowable ratio between the magnitude of the largest signal detected and any secondary signals. The technique is appropriate for any near real-time wideband receiver like the Monobit receiver. In the thesis we explain the progression to the "Ratio Thresholding Technique," and we present the test data and analysis.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. PREVIOUS AND RELATED WORK</td>
<td>7</td>
</tr>
<tr>
<td>III. MONOBIT-II RECEIVER DEVELOPMENT PROGRESSION</td>
<td>19</td>
</tr>
<tr>
<td>3.1 Original Frequency Detection, Tracking, and Thresholding</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Update #1: Improved Signal Tracking</td>
<td>23</td>
</tr>
<tr>
<td>3.3 Update #2: Improved Frequency Selection</td>
<td>25</td>
</tr>
<tr>
<td>3.4 Update #3: Hysteresis Threshold Implementation</td>
<td>29</td>
</tr>
<tr>
<td>3.5 Update #4: Second Improvement to Signal Tracking</td>
<td>32</td>
</tr>
<tr>
<td>3.6 Update #5: Continuous Wave (CW) Signal Reporting</td>
<td>33</td>
</tr>
<tr>
<td>3.7 Update #6 (final): Ratio Threshold Implementation</td>
<td>35</td>
</tr>
<tr>
<td>3.8 Final Configuration Notes</td>
<td>40</td>
</tr>
<tr>
<td>IV. DATA COLLECTION SETUP AND RESULTS</td>
<td>41</td>
</tr>
<tr>
<td>4.1 Test Setup Information</td>
<td>41</td>
</tr>
<tr>
<td>4.2 Actual Data Collection Results</td>
<td>45</td>
</tr>
<tr>
<td>4.3 The Ratio Threshold Implementation</td>
<td>72</td>
</tr>
<tr>
<td>V. CONCLUSION AND FUTURE WORK</td>
<td>81</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1: Overlapping pulses in time at different frequencies</td>
<td>24</td>
</tr>
<tr>
<td>4.1: ChipScope Data, FFT Magnitude Squared for Single Signal Input</td>
<td>47</td>
</tr>
<tr>
<td>4.2: ChipScope Data, FFT Magnitude Squared for Simultaneous Signal Input</td>
<td>48</td>
</tr>
<tr>
<td>4.3: MATLAB Simulation, FFT Magnitude Squared for Single Signal Input</td>
<td>49</td>
</tr>
<tr>
<td>4.4: MATLAB Simulation, FFT Magnitude Squared for Simultaneous Signal Input</td>
<td>50</td>
</tr>
<tr>
<td>4.5: MATLAB Simulation of Changes in Magnitude Squared Peaks, Fs = 2.5GHz</td>
<td>52</td>
</tr>
<tr>
<td>4.6: MATLAB Simulation of Changes in Magnitude Squared Peaks, 1.53-1.65GHz</td>
<td>53</td>
</tr>
<tr>
<td>4.7: MATLAB Simulation of Changes in Magnitude Squared Peaks, Fs = 2.56GHz</td>
<td>54</td>
</tr>
<tr>
<td>4.8: MATLAB Simulation of Changes in Magnitude Squared Peaks, 1.5-1.7GHz</td>
<td>55</td>
</tr>
<tr>
<td>4.9: Fall 2004 Test, 2211MHz constant signal, 1375-2375MHz sweep, 3D view</td>
<td>56</td>
</tr>
<tr>
<td>4.10: Fall 2004 Test, 2211MHz constant signal, 1375-2375MHz sweep, 4 data plots</td>
<td>58</td>
</tr>
<tr>
<td>4.11: Fall 2004 Test, 1474MHz constant signal, 1375-2375MHz sweep, 4 data plots</td>
<td>60</td>
</tr>
<tr>
<td>4.12: 2004 Improved Frequency Binning, 1474MHz constant signal, 1375-2375MHz sweep, 3D view</td>
<td>61</td>
</tr>
<tr>
<td>4.13: 2004 Improved Frequency Binning, 1474MHz constant signal, 1375-2375MHz sweep, 4 data plots</td>
<td>62</td>
</tr>
<tr>
<td>4.14: 2004 Improved Frequency Binning, 1375-2375MHz sweep, 4 data plots</td>
<td>63</td>
</tr>
<tr>
<td>4.15: 2004 Hysteresis, 1474MHz constant signal, 1375-2375MHz sweep, 3D view</td>
<td>64</td>
</tr>
<tr>
<td>4.16: 2004 Hysteresis, 1474MHz Constant Signal, 1375-2375MHz sweep, 4 data plots</td>
<td>65</td>
</tr>
<tr>
<td>4.17: 2004 Hysteresis, 1375-2375MHz sweep, 4 data plots</td>
<td>66</td>
</tr>
</tbody>
</table>
4.18: 2004 Hysteresis, 1375-2375MHz Sweep, 3D view, two signal threshold ........ 67

4.19: 2005 New Threshold, Constant Signal 1474MHz, 1375-2375MHz Swept Input,
3D view ................................................................................................. 68

4.20: 2005 New Threshold, Constant Signal 1474MHz, 1375-2375MHz Swept Input,
4 plots ................................................................................................ 69

4.21: 2005 New Threshold, 1375-2375MHz Swept Input, 3D view .................... 70

4.22: 2005 New Threshold, 1375-2375MHz Swept Input, 4 plots ..................... 70

4.23: Feb 2007, 1375-2375 Sweep, CW signal input at 1858MHz, 3D view .......... 72

4.24: Spring 2007, Ratio Threshold = 0.6, 1375-2375MHz Swept Input, 4 plots ........ 73

4.25: Spring 2007, Ratio Threshold = 0.5, 1375-2375MHz Swept Input, 4 plots ....... 74

4.26: Spring 2007, Ratio Threshold = 0.4, 1375-2375MHz Swept Input, 4 plots ...... 75

4.27: Spring 2007, Ratio Threshold = 0.3, 1375-2375MHz Swept Input, 4 plots ...... 76

4.28: Spring 2007, Ratio Threshold = 0.4, Constant Signal 2000MHz,
1375-2375MHz Swept Input, 3D view ......................................................... 77
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1: 256 Point FFT statistics, with Truncation</td>
<td>14</td>
</tr>
<tr>
<td>2.2: 256 Point FFT statistics, without Truncation</td>
<td>15</td>
</tr>
<tr>
<td>3.1: Sample PDW outputs prior to hysteresis</td>
<td>30</td>
</tr>
<tr>
<td>3.2: Desired PDW outputs</td>
<td>30</td>
</tr>
<tr>
<td>4.1: Monobit Receiver Output</td>
<td>42</td>
</tr>
<tr>
<td>4.2: Initial Test and Evaluation Results</td>
<td>78</td>
</tr>
<tr>
<td>4.3: Recent Test and Evaluation Results</td>
<td>79</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

The motivation for this work was the delivery of the Monobit-II Electronic Warfare (EW) receiver. It required testing, and the reprogrammable nature of this receiver meant that any modifications would require additional tests. Overall, the most rewarding and vital aspect of the work on the Monobit-II receiver was that we were able to work with the inventor and patent holder, Dr. James Tsui. Dr. Tsui was able to shape the test plans, suggest receiver upgrades, and help document the receiver capabilities. Throughout the test and evaluation of the system, the goal of a robust receiver was never lost thanks to the input from Dr. Tsui.

The reason that the Monobit-II receiver came about in the first place is due to many contributing factors. The requirement for wideband queuing receivers, the requirement for lower cost, size, weight and power (C-SWaP) receivers, and the attractiveness of a reprogrammable receiver all contributed to the design of the Monobit-II receiver. The history of electronic intelligence (ELINT) receivers is chronicled in the book, *Digital Techniques for Wideband Receivers* by Dr. Tsui [11]. This history helps to show that the requirements of a wideband receiver, or a device capable of monitoring over 500MHz of bandwidth, is necessary for the analysis and understanding of the radio frequency (RF) environment that is present at any given location. An ELINT receiver is used when the RF signals are unknown, and a user of this system would gain an understanding of their surroundings. The large bandwidth requirement conflicts with the capabilities of an EW system capable of measuring the details of a specific signal source,
or narrowband signal. Therefore, a secondary receiver, or queuing receiver, is often used to search over a wide bandwidth, and provide information to a narrowband system regarding the approximate frequency spectrum to focus upon. We will discuss the common methodology and shortcomings of current queuing receivers, specifically the inability of instantaneous frequency measurement (IFM) receivers to detect simultaneous signals. The EW processing that occurs after a signal is detected can actually identify the specific system that RF signals have originated from, but the Monobit-II receiver and the work performed for this thesis is concerned with the initial detection and reporting of up to two simultaneous RF signals incident upon the receiver.

A secondary factor, the reduction of C-SWaP in EW receivers, motivated the creation of a simplistic hardware design. The capabilities of digital receivers have increased hand in hand with the increased performance of analog to digital converter (ADC) technology and digital integrated circuit technology. The digital signal processing capabilities available in the form of field programmable gate array (FPGA) devices and dedicated digital signal processing (DPS) chips has increased at nearly the same rate as commercial central processing unit (CPU) application specific integrated circuit (ASIC) chips. Although cutting edge CPU chips can operate at very high speeds and computer software can be designed to operate with these chips at high speeds, the data rates of modern ADC technology can create a digital input stream more suited for a custom ASIC chip. The major tradeoff when designing a custom ASIC chip versus utilizing an FPGA chip is that any changes to the processing algorithm can be reprogrammed in a matter of seconds on an FPGA while an ASIC would require a redesign, followed by a new
fabrication run, which is then followed by new packaging and integration with the ADC device.

The Monobit-II receiver design team selected an FPGA as the processing unit for the ability to reprogram the chip along with other benefits of FPGA technology. The design flow for FPGA devices typically begins with the creation of a hardware description language (HDL) version of the algorithm and processing desired for implementation. This HDL code is the synthesized into the logic gates and devices available on the target FPGA chip, typically look-up tables (LUTs), flip-flops (FF) and multiplexers (MUX). The on chip devices that HDL code utilized must then be automatically laid out on the available FPGA resources, and arranged in such a fashion that the operating speed of the design is optimized. This place and route step is repeated until an acceptable programming file for the FPGA is created from the initial HDL code. Luckily, the manufacturers of FPGA devices provide design tools that accept HDL code and provide a programming file as an output. This automation helps to keep the redesign and reprogramming time for FPGA technology at a minimum.

The Monobit-II receiver was named primarily for the “Monobit” algorithm that is implemented as the core processing block on its FPGA. The reason it is the second Monobit is due to the fact that after the invention of the Monobit algorithm, the first receiver design was implemented as an ASIC chip. As discussed earlier, any reprogramming of an ASIC would require a new chip, so the motivation for FPGA technology was clear for laboratory purposes.

The Monobit algorithm that is Dr. Tsui’s patented invention, and can be thought of as a Fast Fourier Transform (FFT) approximation. The FFT is a mathematical
operation that takes time domain sampled data and converts the data set into the frequency domain. Fourier transform theory utilizes complex multiplications and summations, which were optimized with the Cooley-Turkey algorithm into a pipelined butterfly set of multiplications and additions. The further optimization of the Monobit algorithm replaces the complex multiplications in the FFT with additions and simple binary shifts to obtain an approximation of the FFT. The fact that the Monobit algorithm does not perform the exact multiplications necessary for an exact FFT basically means that the frequency spectrum will have a smaller signal to noise ratio (SNR). However, the ability to implement the Monobit FFT at much higher speeds and a much smaller hardware footprint than an exact FFT makes it an attractive option.

As mentioned earlier, an ELINT system is designed such that the frequency content of the bandwidth of interest can be obtained by the user. The implementation of the Monobit FFT onto an FPGA unit is the first step for processing digitized data from an ADC. The following steps, and the focus of this thesis, are the determination of the actual signals present in the Monobit-II receiver from the Monobit FFT output. We know that the frequency spectrum from the Monobit FFT is imperfect, and we must also figure out the theoretical limitations of the Monobit-II system based upon the signal conditioning hardware present before the Monobit FFT is performed. The signal detection algorithms do not need to search for signals that the system devices are not capable of passing through.

The actual hardware that conditions RF signals on the Monobit-II system is comprised of RF bandpass filters, limiting amplifiers, and the ADC chip. The limiting amplifier is a device that automatically adjusts its gain based upon the input signal.
strength. It can be thought of as a device that will take as input any signal down to a given power range and amplify this signal to the maximum output voltage swing the amplifier is set to. Below this specified power the limiting amplifier will roll-off in its ability to amplify signals, until it passes through only a voltage swing corresponding to the ambient noise of the environment. The ADC chip for the Monobit-II receiver is a very capable device that was designed by Hughes Research Labs, which is now HRL, Inc. The ADC is an indium phosphide (InP) device designed to work up to a sampling rate of 8GHz. The output of this device is three bits, and custom demultiplexer circuits were also designed by HRL to work with this device. The Monobit-II receiver design decided to use the ADC at a speed of 2.5GHz, and use the two most significant bits (MSBs), effectively making the ADC a two bit device. Due to the fact that the ADC can operate at up to 8GHz, the device is capable of directly sampling in Nyquist zones other than baseband. Therefore, the RF bandpass filters determine the operating region for the system, and a 1GHz filter centered at 1.875GHz was selected. There are two copies of this filter, one prior to the limiting amplifier, and one after the limiting amplifier prior to the ADC.

As previously mentioned, the ability to detect two simultaneous signals is an important benefit of using the Monobit-II receiver, but it is also vital to know the limits of this two signal detection capability. Knowing the specifics of the circuitry prior to the FPGA, the theoretical characteristics of input signals can be determined. The devices with the most impact are the limiting amplifier and the 2-bit ADC. A limiting amplifier effectively equalized the amplitude of a large input power range, while the number of ADC bits is directly proportional to the two signal dynamic range. The two signal
dynamic range is the maximum power difference between two input signals that will still be reported correctly as two signals by the receiver.

The increased noise generated by the Monobit FFT approximation also causes what we refer to as spurious signal peaks to be generated in the frequency domain when a single RF signal is present. The magnitude of these spurious signals depends upon the input frequency, but simply compensating for these erroneous peaks based upon the input signal can lead to problems when a second signal detection is also desired. A two signal input will cause not only two peaks in the frequency spectrum, but also peaks and the inter-modulation points of these frequencies.

Overall, the problem of correctly identifying a signal incident upon the receiver is a complex task, but this problem is greatly increased when searching for a possible second signal while maintaining a low probability of reporting a false signal. The primary metrics used for receiver performance are the probability of detection ($P_d$) and the false alarm rate ($FAR$). The probability of detection is the probability of correctly identifying an input signal every time it passes into the receiver. The false alarm rate is the probability that the receiver will generate an incorrect signal identification over a given time period.
II. PREVIOUS AND RELATED WORK

The design of an instantaneous frequency measurement (IFM) receiver is discussed by McCormick et al [1]. It was shown that an analog IFM receiver consisted of an analog correlator designed to give an analog estimate of the covariance $\epsilon^{j\omega \tau}$, where $\omega$ is the angular frequency of the input and $\frac{1}{\tau}$ represents the single correlator unambiguous bandwidth, which is proportional to the delay line length. It was also shown that it requires multiple correlator units with different delay line lengths to achieve an accurate estimate of the frequency. The solution to the ambiguity problem that arises from multiple correlator units was addressed in this paper. The bandwidth, $B_\mu$, of the IFM receiver is related to the delay line lengths of the correlators and the noise protection setting $q$.

$$B_\mu = \frac{\prod_{n=1}^{N} \frac{1}{\tau_n}}{(4q + 1)^{N-1}}$$

Noise protection was required by the IFM receiver because of inexact frequency estimation, and the equation was derived from number theory, including the Chinese Remainder Theorem. The motivation for solving the ambiguity in the IFM receiver was vital not only for single signal detection reliability, but it also laid the groundwork for the study of multiple signal inputs to an IFM receiver.
Hero et. al. [2] outline a method to detect and classify simultaneous signals when a maximum acceptable value for the probability of false alarms ($P_{FA}$) is given. The important realization presented in this paper is that not only is it unknown if multiple signals are present at any given time, it is also unknown if any signals are present at all. The presented “max-min” strategy takes into account the probability of detection ($P_d$), the classification problem, and the simultaneous signal detection and classification problem. The actual mathematical model presented in this paper was not implemented as part of our study, but the conflicting requirements for a static threshold are demonstrated. The resulting $P_d$ must be maximized while the $P_{FA}$ is minimized, and we will show that this can be achieved when a different type of ratio threshold technique is utilized.

Chang et. al. [3] describes the idea of dynamically adjusting the detection thresholds of the system. His theory that a feedback mechanism between the detection and tracking portions of a system is necessary to improve the performance of both steps was also a key realization from our work. The tradeoff between $P_d$ and $P_{FA}$ was also discussed, specifically the waterfall effect of errors from one stage producing errors in the following stages. There are other historical algorithms presented, including nearest neighbor (NN), probabilistic data association (PDA), PDA and joint probabilistic data association (PDA/JPDA), maximum likelihood (ML) and multiple hypothesis tracking (MHT). Although Chang is concerned with target tracking in cluttered environments, we can extrapolate upon his ideas to the “frequency domain, signal tracking in noise space.” The ML and NN techniques are compared, and the dependence of both techniques on the signal to noise ratio (SNR) and detection thresholds is shown. From this paper we can
see that the existing techniques for detection and tracking may be implemented in
different fashions, but the overall performance is not incredibly different under similar
operating conditions.

In [4], Weiss provides a good overview of the assembly, capabilities and
drawbacks of IFM receivers, along with a proposed method to identify simultaneous
signals in an IFM receiver. The noted capabilities that an IFM receiver provides are a
very wide instantaneous bandwidth, very accurate frequency measurement, high dynamic
range, a real time frequency estimate, and good sensitivity. Weiss describes an output
temperature from the IFM to be $e^{-j\omega n}$, which is the filtered complex multiplication of the
delayed signal and the undelayed signal. The actual phase shift from a delay line $n$ is the
product of the input frequency, $\omega$, and the delay, $\tau_n$.

$$\theta_n = \omega_n \tau_n$$

The problem arising when more than one input frequency is present in an IFM
receiver’s delay lines is apparent, because the multiplications to derive the output voltage
assume that only one frequency component is creating a phase shift, the resulting voltage
level would still be a combination of multiple frequencies. Weiss’s compares the least
squares algorithm, a 2x2 matrix suboptimal search, and a polynomial rooting technique to
resolve the outputs of multiple delay lines containing more than one signal. The
polynomial rooting technique was simulated with a four delay line IFM receiver, and the
two input signal performance was presented. The standard deviation and bias of the
frequency estimate from the actual input signals was less than 5 MHz total for the worst
case scenarios when the two signals were input at the same power level. The two signal
dynamic range, or the ability to detect a second signal at a lower input power, was also
presented. The results show almost 20dB of two signal dynamic range with errors less than 10MHz for the weaker signal. This theoretical approach did show some promise for two signal detection with an IFM receiver.

The initial patent of the Monobit receiver [7] was accepted in 1998, and gave an outline of the Monobit receiver architecture. The idea was to use a 1-bit ADC connected to circuitry capable of performing the Fourier Transform through simplified complex mathematical processes. The name Monobit was derived from the 1-bit ADC output processed with a “1-bit” FFT approximation, reducing the DFT kernel to the values of (1, j, -1, -j), or (0, π/2, π, -π/2) on the unit circle.

The discrete Fourier Transform (DFT).

\[ X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} \]

The DFT kernel

\[ e^{-j2\pi nk/N} \]

The resulting approximation would no longer require complex multiplications, and a hardware implementation would only require additions and shifting operations. The benefit of a Fourier domain receiver was said to be that it would still give an accurate frequency estimation of the incident signals on an EW receiver. This was desirable because other EW systems utilized IFM receivers that were proven to have difficulty discriminating multiple input signals.

Despain [12] focused on further optimizing the FFT algorithm known as the Cooley-Tukey method for easier hardware implementation. He demonstrates that multiplications can be replaced by shifting operations with variations in gain, where the
gain would then be compensated for in later stages. The basis for his algorithm is the CORDIC technique for vector rotations.

His definition of the DFT is:

\[ A_r = \sum_{k=0}^{N-1} B_k W_N^{rk} \quad r = 0, 1, 2, \ldots, N - 1 \]

The input signal is \( B_k \), and the size of the DFT is \( N \). Where the kernel function, or twiddle factor is defined as:

\[ W_N^{rk} = e^{(-2j\pi k)/N} \]

One conclusion that Despain arrives at is that “An actual computation of the Fourier transform is inherently inexact using any method.” [12] This true because finite amounts of time or frequency sampled data is all that can be processed. His observation also reinforces the acceptability of approximation techniques.

The original Monobit receiver designed for the Air Force Research Labs (AFRL) as an application specific integrated circuit (ASIC) was presented by Pok et. al [5]. The receiver consisted of a nonlinear RF front end, an ADC, and the ASIC chip. The ASIC was designed to accept the ADC input as 2 bit, 2.5GHz samples, perform the Monobit FFT approximation, followed by hardware logic to decide the number of peaks sent to the output. The FFT circuitry performs a 256 point Monobit FFT, providing 128 data sets of 7 bit real and 7 bit imaginary outputs. The logic following the FFT found the four strongest signals based on thresholding of real and imaginary values. The reasoning for the thresholding selection is described to be a technique that compensates for the non-linear front end by utilizing two different thresholds. A high threshold is set and if a signal crosses this threshold, the secondary threshold crossings are ignored. A lower
threshold is used when the high threshold is not crossed, and the two outputs above this threshold are passed to the next stage. The outputs of the initial sorting stage are real values, imaginary values, addresses, and flag bits. The real parts squared plus the imaginary parts squared of the four strongest signals were then computed in hardware circuitry. There is a final sorting subsystem compares all four magnitude-squared values and outputs the strongest two address and flag bits, if there are at least two signals present. The final sort will output less than two addresses, one or zero, if there is one or zero inputs to the subsystem, respectively. The ASIC design is shown to result in a 34 input and 16 output chip, sized 15mm by 15 mm, which could operate at 156.25MHz with 812,931 transistors. The final conclusions suggest that further study of the thresholding techniques is required, and that a 99.89% probability of detection with 1.37% false alarm rate would be expected from the design.

Grajal et. al. [6], at ETSIT, Universidad Politecnica Madrid in Madrid, Spain were able to implement the Monobit FFT approximation and study the theoretical and experimental results when utilizing a one bit digitizer. The work by Grajal and other provides insight into the selection of a set threshold based upon a desired probability of false alarms. This theoretical analysis accounted for the actual calculated values of different Monobit FFT channels or frequency bins, regardless of input noise power. It should be noted that a linear detector was used to determine if a signal was present from the real and imaginary outputs of each channel.

\[ P(k') = \begin{cases} \left( \frac{N}{N/2} \right) \frac{1}{2^N} & \text{if } k' = 0 \\ \left( \frac{N}{N-k'} \right) \frac{2}{2^N} & \text{if } 0 < k' \leq N \end{cases} \]
\[ P_{fa}(T_h) = \sum_{k \geq T_h}^N P(k') \]

This threshold calculation is useful when a value for a minimal noise threshold is needed of a given FFT size and given probability of false alarm, or false alarm rate.

A quick overview of different EW receiver technologies is provided by Tsui and Stephens [8], highlighting the benefits of digital signal processing over analog receiver technologies. Receivers using channelization or Bragg Cell techniques are shown to benefit from digital processing techniques. Another receiver type, compressive receivers, are implemented with a dispersive delay line and crystal video detectors, so the benefits of digital processing can only be obtained in the later stages of processing. The idea of implementing a channelized receiver with digital filter banks is discussed, but the issues of the filter complexity and large hardware circuitry requirements limited the capabilities of even digital channelized receivers. The Monobit receiver is presented as an efficient implementation of channelization through the use of an FFT approximation. The ability to detect two simultaneous signals, albeit with a low instantaneous dynamic range, is discussed. The details of the receiver are referenced seen in Pok, et al. [5]. A result presented is that a 2-bit ADC input provides much better results than a 1-bit ADC, but there is no significant improvement expected by moving to a 3-bit ADC.

The patented Monobit FFT approximation for EW systems was later expanded upon in a 2002 patent [9], suggesting that multiple “Monobit Receiver” channels could be implemented in parallel, becoming a similar to channelized receiver commonly implemented with parallel bandpass filters. This patent combines the benefit of a Monobit receiver capable of measuring two simultaneous signals in its processing
bandwidth with the benefit of a channelized receiver that is capable of monitoring the environment in multiple bandwidths simultaneously.

A very thorough study of the footprint of various FFT sizes and techniques was presented by Sanchez et al [10]. The methodologies for the implementation of an FFT as either a feedback or feed-forward design is presented. The benefit of a feedback design is that the same hardware resources will be used for different stages of the FFT, saving space in an FPGA design, but limiting the input data rate. The benefit of a feed-forward design is that a pipelined architecture is implemented, where different resources are used at each stage, greatly increasing the operating speed and through-put capabilities at the expense of the hardware footprint. The selection of the radix size for the FFT is also treated as a variable, and this radix is basically the degree of FFT decomposition from a Cooley-Tukey FFT algorithm. An interesting comparison that is also presented is the hardware footprint and processing capabilities of the Monobit FFT approximation versus a full FFT implementation. The final variable considered is the truncation of the FFT data from one stage to the next, and also at the output, basically limiting the FPGA resources from growing at each FFT stage. The results do follow the tradeoffs expected from the different methodologies, and sample data is copied in table 2.1 below.

Table 2.1: 256 Point FFT statistics, with Truncation

<table>
<thead>
<tr>
<th>FFT Description</th>
<th>Radix</th>
<th>Stages</th>
<th>Area (slices)</th>
<th>Latency (usec)</th>
<th>Speed (MHz)</th>
<th>Speed (MS/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit FFT, Feedback</td>
<td>4</td>
<td>4</td>
<td>1903</td>
<td>1.128</td>
<td>274</td>
<td>274</td>
</tr>
<tr>
<td>8-bit FFT, Feed-Forward</td>
<td>2</td>
<td>8</td>
<td>4533</td>
<td>2.031</td>
<td>227</td>
<td>454</td>
</tr>
<tr>
<td>8-bit FFT, Feed-Forward</td>
<td>4</td>
<td>4</td>
<td>4586</td>
<td>0.810</td>
<td>236</td>
<td>943</td>
</tr>
<tr>
<td>8-bit Monobit, Feedback</td>
<td>4</td>
<td>4</td>
<td>1430</td>
<td>0.868</td>
<td>319</td>
<td>319</td>
</tr>
<tr>
<td>2-bit Monobit, Feedback</td>
<td>4</td>
<td>4</td>
<td>474</td>
<td>0.868</td>
<td>319</td>
<td>319</td>
</tr>
<tr>
<td>2-bit Monobit, Feed-Forward</td>
<td>4</td>
<td>4</td>
<td>1019</td>
<td>0.733</td>
<td>243</td>
<td>972</td>
</tr>
</tbody>
</table>
Table 2.2: 256 Point FFT statistics, without Truncation

<table>
<thead>
<tr>
<th>FFT Description</th>
<th>Radix</th>
<th>Stages</th>
<th>Area (slices)</th>
<th>Latency (usec)</th>
<th>Speed (MHz)</th>
<th>Speed (MS/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit FFT, Feedback</td>
<td>4</td>
<td>4</td>
<td>2914</td>
<td>1.254</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>8-bit FFT, Feed-Forward</td>
<td>2</td>
<td>8</td>
<td>6401</td>
<td>2.154</td>
<td>227</td>
<td>454</td>
</tr>
<tr>
<td>8-bit FFT, Feed-Forward</td>
<td>4</td>
<td>4</td>
<td>6703</td>
<td>0.861</td>
<td>236</td>
<td>943</td>
</tr>
<tr>
<td>8-bit Monobit, Feedback</td>
<td>4</td>
<td>4</td>
<td>1980</td>
<td>0.902</td>
<td>306</td>
<td>306</td>
</tr>
<tr>
<td>2-bit Monobit, Feedback</td>
<td>4</td>
<td>4</td>
<td>1048</td>
<td>0.865</td>
<td>319</td>
<td>319</td>
</tr>
<tr>
<td>2-bit Monobit, Feed-Forward</td>
<td>4</td>
<td>4</td>
<td>2527</td>
<td>0.733</td>
<td>243</td>
<td>972</td>
</tr>
</tbody>
</table>

The results included statistics without truncation of the data, and this data is shown in table 2.2. Without truncation, the area increased drastically while there were slight increases in latency and slight decreases in speed. The main tradeoff that was presented when truncating data is that the noise level increases, leading to the increase of false alarms. Overall, this study provided a direct comparison of a traditional FFT implementation in an FPGA with the Monobit FFT implementation, showing that the area required for a Monobit FFT is less than that of the full resolution FFT.

Tsui [11] gives a very detailed description of the Monobit receiver background and potential applications. The idea came from global positioning system (GPS) receivers that used only one or two bits, and this reduced hardware approach was adapted to wideband receivers. The original design relied on computer simulations to suggest a threshold setting that would then be implemented in an ASIC chip. It was part of the original vision that the Monobit receiver design could be integrated with an ADC design on the same chip. The original Monobit receiver system included a limiting amplifier surrounded by bandpass filters that would limit the signal to a 1GHz bandwidth, 1.375GHz to 2.375GHz. A 2-bit 2.5GHz ADC was used, and a 4-point Monobit FFT
would process the demultiplexed data. The chip used a threshold after the Monobit FFT, prior to any magnitude calculations, which would find up to four large bin locations. The magnitude of the four real and imaginary values was then calculated, and another threshold was used to determine the actual signals. There is a good discussion of the tradeoffs between false alarms and sensitivity, then the selection of a threshold to limit spurious signals but maintain simultaneous signal capabilities. The stated goal for the output of the frequency selection portion, or frequency encoder chip, was a zero when no signals were present, the frequency of one signal if it is present, or the frequencies of two signals when there are two signals.

The basic operation of the threshold selection logic was that a primary threshold was determined to not allow noise to be considered a signal. Next, the first threshold somewhere above the primary threshold was set based on single signal inputs. This first threshold must eliminate the single signal spurs and create only a single signal when only a single signal was present. The reason given for the difficulty in finding a two signal threshold, or second threshold, was that the limiting amplifier and 2-bit ADC create a non-linear system that is further complicated by the interference of two input signals. The second threshold would actually be between the primary and first threshold, and the threshold logic behaved as follows. If the first threshold is crossed, it is assumed only one signal is present and this frequency is saved. If the first threshold is not crossed, the second threshold is used and the frequencies above this threshold are saved. The real and imaginary values that made it through the initial threshold logic would then be input into equation 1 to determine the magnitude of the signal.

\[ |X(k)| = \sqrt{X_r(k)^2 + X_i(k)^2} \]
Here, $X_r$ is the real part of the FFT output and $X_i$ is the imaginary part. After this magnitude was calculated, the result was compared to a final threshold, and only signals above this threshold were reported.

Luckily, the receiver was thoroughly tested, and the results are reproduced in table 2.3 below:

<table>
<thead>
<tr>
<th>Amplitude of 2nd Signal vs. 1st Signal (dB)</th>
<th>Found 1st Signal (%)</th>
<th>Found 2nd Signal (%)</th>
<th>Found Both Signals (%)</th>
<th>Found Neither (%)</th>
<th>Found Erroneous Signal (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>65.1</td>
<td>59.3</td>
<td>24.4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>78.9</td>
<td>45.0</td>
<td>23.9</td>
<td>0</td>
<td>0.45</td>
</tr>
<tr>
<td>-2</td>
<td>89.2</td>
<td>29.9</td>
<td>20.9</td>
<td>0</td>
<td>0.38</td>
</tr>
<tr>
<td>-3</td>
<td>93.9</td>
<td>18.0</td>
<td>12.0</td>
<td>0.13</td>
<td>0.38</td>
</tr>
<tr>
<td>-4</td>
<td>97.9</td>
<td>9.5</td>
<td>7.6</td>
<td>0.13</td>
<td>0.25</td>
</tr>
<tr>
<td>-5</td>
<td>99.8</td>
<td>3.3</td>
<td>3.0</td>
<td>0</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Buck et. al. [13] provide a brief overview of the Monobit-II receiver from both a hardware and algorithmic standpoint. The overall form-factor of the system is provided, along with details of the analog components and the system voltage requirement. The main accomplishment detailed by Buck et. al. is the same ratio thresholding technique presented in this work. However, at the time of publication extensive hardware tests of the ratio threshold FPGA configuration had not been performed. It will be shown later that the suggested ratio setting of 0.6 in [13] was actually higher than needed to achieve the necessary false alarm rates.

The algorithmic description details how frequency bins are zeroed and completely ignored in the hardware design. This is due to the desire to keep false DC level peaks from being reported, and also the knowledge that the analog filters will remove virtually all signals from these bins.
The operational complexity of the binary search algorithm after the magnitude squared calculation is given here as:

$$O(\log_2 N)$$

Where \(N\) is the number of FFT points. This is an improved search algorithm over the linear search presented by Pok et. al [1]. Another interesting point of discussion is the sharing of the Monobit-II receiver prototype with Northrop Grumman Corp., and their plan to perform flight tests with this system.
III. MONOBIT-II RECEIVER DEVELOPMENT PROGRESSION

Our work on the Monobit-II receiver was focused primarily on thresholding techniques for simultaneous signal detection. The techniques for multiple signal detection that were included with the initial delivery of the Monobit-II receiver were designed by Dr. Emmert, (WSU and AFRL). Throughout the testing and evaluation of the Monobit-II receiver, alternative thresholding techniques were proposed by our fellow engineers, notably Dr. Tsui. The baseline performance of the receiver was documented, and then through MATLAB simulations and further HDL code development, we were able to improve the performance of the Monobit-II receiver.

3.1 Original Frequency Detection, Tracking, and Thresholding

The Monobit-II receiver hardware consists of two sets of dip switches, each with 16 individual switches, and referred to as “threshold high” and “threshold low.” These two dip switch sets were intended to let the users of the receiver manually set threshold levels, and dynamically change the levels to optimize the performance of the receiver. Thresholding is utilized in the detection algorithm by comparing the magnitude squared of the Monobit FFT approximation to the set threshold values.

The low threshold value was implemented in HDL such that all the magnitude squared values would be reduced by this value. Practically, all the magnitude squared values had the low threshold value subtracted from them. The reason for a reduction of
all the magnitudes by this threshold was to effectively remove the noise across the
spectrum.

After the reduction by the low threshold, and prior to utilizing the high threshold
value, the magnitude squared values are input to a binary search tree algorithm. This
search finds the largest magnitude and the corresponding frequency bin. The largest
frequency bin is referred to as the peak bin. The peak bin and magnitude is saved, while
the magnitude squared spectrum is modified such that the frequency bins surrounding the
peak bin are set to zero. The two lower and two higher bins are zeroed, the reason being
that the modified spectrum is passed through the binary search a second time. The output
of the second binary search is next highest magnitude squared value and frequency bin.
The second search results in what is referred to as the secondary peak bin.

The high threshold value was then compared the peak bin and the secondary peak
bin, and if the values were larger than the high threshold, these frequency bins were
considered to contain signals of interest.

After determining of the signals of interest, additional HDL code required that for
a new peak bin or secondary peak bin to be considered a valid signal, the peak must be
present for two consecutive FFT cycles.

If a valid signal was already detected, the peak and secondary peak bin locations
were compared to this valid bin location. If a current peak value was within two
frequency bins of a valid signal, the pulse width associated with that signal would be
incremented by one, and it would continue to be a valid signal. However, if a signal that
was considered to be valid on the previous cycle did not match a peak of the current
cycle, then that valid signal bin number, pulse width, and time of arrival would be output as a new pulse descriptor word (PDW) from the receiver.

The original logic used to track two simultaneous signals relied on the relationship between the frequency bin values. If two simultaneous signals were present, the signal with a frequency bin closer to zero was considered the low frequency, and the signal with a larger value for the frequency bin was considered the high frequency. This logic assumed the output from the search algorithms that calculated the peak bin and the secondary peak bin would consistently output any simultaneous signals, and as long as the peak values remained above the high threshold, the tracking based upon the frequency bin would not be corrupted.

After performing initial testing and evaluation of the receiver performance for both single signal and simultaneous signal inputs, a higher than desired false alarm rate, along with a lower than desired probability of detection contributed to the implementation of more robust frequency selection and tracking algorithms. Below is a Pseudo-Code sample of this original logic.
IF low_bin_mag > high_threshold
   IF f1_data_valid = 1
       f1_frequency_bin = low_frequency_bin
       f1_pw = f1_pw + 1
       f1_toa = f1_toa
   ELSEIF low_frequency_bin = f1_frequency_bin
       f1_frequency_bin = low_frequency_bin
       f1_pw = 2
       f1_toa = f1_toa
   ELSE
       f1_frequency_bin = low_frequency_bin
       f1_pw = 1
       f1_toa = current_toa
   END
ELSEIF high_bin_mag > high_threshold
   IF f2_data_valid = 1
       f2_frequency_bin = high_frequency_bin
       f2_pw = f2_pw + 1
       f2_toa = f2_toa
   ELSEIF high_frequency_bin = f2_frequency_bin
       f2_frequency_bin = high_frequency_bin
       f2_pw = 2
       f2_toa = f2_toa
   ELSE
       f2_frequency_bin = high_frequency_bin
       f2_pw = 1
       f2_toa = current_toa
   END
ELSEIF low_bin_mag > low_threshold OR high_bin_mag > low_threshold
   IF low_bin_mag > low_threshold
      IF f1_data_valid = 1
         f1_frequency_bin = low_frequency_bin
         f1_pw = f1_pw + 1
         f1_toa = f1_toa
      ELSEIF low_frequency_bin = f1_frequency_bin
         f1_frequency_bin = low_frequency_bin
         f1_pw = 2
         f1_toa = f1_toa
      ELSE
         f1_frequency_bin = low_frequency_bin
         f1_pw = 1
         f1_toa = current_toa
      END
   END
   IF high_bin_mag > low_threshold
      IF f2_data_valid = 1
         f2_frequency_bin = high_frequency_bin
         f2_pw = f2_pw + 1
         f2_toa = f2_toa
      ELSEIF high_frequency_bin = f2_frequency_bin
         f2_frequency_bin = high_frequency_bin
         f2_pw = 2
         f2_toa = f2_toa
      END
   END
END
3.2 Update #1: Improved Signal Tracking

After theorizing that the probability of detection and the false alarm rate could be improved by creating a new FPGA configuration, the original VHDL code was analyzed, and the area known as stage four was selected for modification. The analysis of the original signal tracking algorithm in stage four brought to light that possible scenarios could exist in which simultaneous or overlapping pulse events could cause erroneous results. A relatively simple example of an overlapping pulse event is demonstrated in the figure below.
The figure shows that if the original two signals are detected, and then the subsequent tracking logic assigns a 2.0GHz signal the high frequency demarcation and the 1.4GHz signal the low frequency demarcation, the third pulse at 2.2GHz would cause an erroneous event, because the signal being maintained as the higher frequency at 2.0GHz would cause the 2.2GHz signal to either occur undetected, or cause the 2.0GHz signal to loose its track, and output a PDW with incorrect information.

This simple example, along with other possible simultaneous signal events, led to the creation of a more complex signal tracking algorithm. The new algorithm would still accept the two peak values from the search operations, along with the previous cycle’s signal information. The method for pulse tracking now relied upon comparing the actual frequency bins from one cycle to the next. A signal would therefore be considered a valid signal as long as the magnitude in a bin plus or minus two bins away was above the high threshold.
3.3 Update #2: Improved Frequency Selection

The original method from frequency selection would update the frequency bin of a valid signal each time the signal crossed the high threshold. The logic behind continuously updating the frequency bin was because the receiver would output a PDW at the start of a pulse, or once it was considered valid, and also output a PDW at the end of a pulse, or when the magnitude squared fell below the high threshold. By outputting two PDW’s per pulse, the receiver would give an initial frequency bin estimate and an ending bin estimate, theoretically to identify any linear frequency modulation (LFM) pulses. However, after studying the frequency bin estimates at the beginning and end of normal, or continuous wave (CW) pulses, the estimates were not consistent from one pulse to the next. It was then realized that the number of time domain samples that are actually captured at the beginning and end of a pulsed signal input is a random variable. Therefore, the tradeoff was made to sacrifice LFM signal detection for improved frequency selection.

The implementation of an improved frequency selection was grounded in the observation all the time domain input points for the second FFT cycle would contain the pulsed signal. This FFT would then provide a much more accurate estimate of the frequency bin than the first FFT or last cycle that crossed the high threshold. The original assumption was that a 200ns pulse was the minimum detectable pulse width for the receiver, derived from the requirement in the detection algorithm that the FFT bin energy must be above the high threshold for two consecutive 100ns cycles. Studying the
amount of signal present in the time domain for one cycle, an evenly distributed random variable would have a probability density function shown below.

\[ F(x) = \frac{1}{n} \]

Here, \( n = 256 \) possible samples, or 100ns.

The threshold is set such that on average, if a signal is present for only 128 samples, or 50ns, it should be above the threshold. Using this logic, 50% of the time a signal will have 128 to 256 samples in the first and last FFT cycles that are considered valid.

If the length of a signal is also treated as a random variable from 512 samples to the maximum detectable pulse size \( (p_{\text{max}}) \), defined here as approximately 51.1us. Then the worst case scenario would be when an FFT cycle contains the first 127 samples of the signal, and this FFT cycle magnitude squared does not cross threshold.

To derive the maximum pulse samples \( p_{\text{max}} \).

\[
 s = \frac{256 \text{ samples}}{100 \text{ ns}}, \quad t = 51.1 \text{ us} \quad \text{(maximum size)}:
\]

\[
p_{\text{max}} = s \times t = \left(\frac{256}{100\text{ns}}\right) \times (51,100\text{ns}) = 130,816 \text{ samples total}
\]

Therefore, the following FFT cycle will contain 256 samples, and this magnitude squared value will cross threshold, and be considered the first FFT cycle. If the worst case scenario pulse width is considered, out of 512 samples, only 129 samples will be present in the second FFT, or in reality, the third FFT cycle containing actual signal data. Although this is the worst case, any signal that meets the criteria of cross the threshold with at least 128 samples would still be detected. The worst case is a signal where cycle 1 has 127 samples and is not detected, then cycle 2 has 256 samples and is the valid cycle. Then cycle 3 has 256 samples and is the second valid cycle. Therefore, any
signal with a pulse width of at least 639 (127+256+256) samples, or 249.6ns, would have 256 samples in the second valid FFT cycle, and this frequency bin estimate would be more accurate than a computation in which 128-255 samples are used, as would be the case in 49.6% of the time.

The probability that the first or last FFT cycle contains between 128 and 255 samples:

\[ F(x) = \frac{127}{256} = 0.496 \]

The actual probability of a given pulse duration is not of interest at this time, nor is it readily available information. However, the range of 512 samples to 130,816 samples leaves many possible pulse sizes greater than the 639 samples necessary for the newly developed optimal signal detection.

Below is a Pseudo-Code sample of the improved frequency selection and tracking, but with the plus two bins or minus two bins logic not explicitly shown.

```
------------ Improved Frequency Selection Thresholding ------------
IF primary_peak > high_threshold
  IF f1_data_valid = 1
    IF primary_peak_bin = f1_frequency_bin
      f1_data_valid = 1
      f1_pw = f1_pw + 1
      f1_toa = f1_toa
      f1_frequency_bin = f1_frequency_bin
    ELSE
      output = f1_PDW
      f1_data_valid = 0
      f1_pw = 1
      f1_toa = current_toa
      f1_frequency_bin = primary_peak_bin
  ELSE
    ELSEIF f2_data_valid = 1
      IF primary_peak_bin = f2_frequency_bin
        f2_data_valid = 1
        f2_pw = f2_pw + 1
        f2_toa = f2_toa
        f2_frequency_bin = f2_frequency_bin
      ELSE
        output = f2_PDW
```

27
f2_data_valid = 0
f2_pw = 1
f2_toa = current_toa
f2_frequency_bin = primary_peak_bin

END

ELSE

IF primary_peak_bin = f1_frequency_bin
f1_data_valid = 1
f1_pw = f1_pw + 1
f1_toa = f1_toa
f1_frequency_bin = primary_peak_bin

ELSEIF primary_peak_bin = f2_frequency_bin
f2_data_valid = 1
f2_pw = f2_pw + 1
f2_toa = f2_toa
f2_frequency_bin = primary_peak_bin

ELSE

f1_data_valid = 0
f1_pw = 1
f1_toa = current_toa
f1_frequency_bin = primary_peak_bin

END

END

ELSEIF primary_peak > low_threshold

IF f1_data_valid = 1

IF primary_peak_bin = f1_frequency_bin
f1_data_valid = 1
f1_pw = f1_pw + 1
f1_toa = f1_toa
f1_frequency_bin = f1_frequency_bin

ELSE

output = f1_PDW
f1_data_valid = 0
f1_pw = 1
f1_toa = current_toa
f1_frequency_bin = primary_peak_bin

END

ELSEIF f2_data_valid = 1

IF secondary_peak_bin = f2_frequency_bin
f2_data_valid = 1
f2_pw = f2_pw + 1
f2_toa = f2_toa
f2_frequency_bin = f2_frequency_bin

ELSE

output = f2_PDW
f2_data_valid = 0
f2_pw = 1
f2_toa = current_toa
f2_frequency_bin = secondary_peak_bin

END

ELSE

IF primary_peak_bin = f1_frequency_bin
f1_data_valid = 1

28
3.4 Update #3: Hysteresis Threshold Implementation

Another problem that can arise when thresholding on the magnitude squared values of a Monobit FFT output is that the actual magnitude squared values for a single signal can vary from one FFT cycle to the next. The theoretical reason behind this variation is the calculation through the FFT butterfly stages is sensitive to the phase of the input signal and noise for every FFT cycle. This variation in the magnitude squared value was shown to cause problems when the actual Monobit-II receiver was tested. The PDW output would contain incorrect pulse width information, and often one PDW would follow another PDW with identical frequency bins and consecutive time of arrivals. A sample of PDW outputs is shown in table 3.1 below.
Table 3.1: Sample PDW outputs prior to hysteresis

<table>
<thead>
<tr>
<th>Frequency Bin</th>
<th>Pulse Width</th>
<th>TOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5</td>
<td>1508201</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>1508208</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1508301</td>
</tr>
</tbody>
</table>

The input signal for this example would be a 1.52GHz pulse with a 1us pulse width and a 10us PRI. The expected values for consecutive PDW outputs are shown in table 3.2. The 1.52GHz signal corresponds to frequency bin 100, and a 1us pulse corresponds to a pulse width count of 10.

Table 3.2: Desired PDW outputs

<table>
<thead>
<tr>
<th>Frequency Bin</th>
<th>Pulse Width</th>
<th>TOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10</td>
<td>1508201</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1508301</td>
</tr>
</tbody>
</table>

The difference between the sample PDW outputs and the desired PDW outputs is the additional reporting of a signal almost immediately following the initial signal. This is the PDW with a pulse width of 5 followed by the PDW with a pulse width of 3. It is observed that the TOA for the first pulse, plus the reported pulse width, gives a relative time count value of 1508206. The TOA for the next pulse is seen to be the following clock cycle, 1508208, and has a pulse end that can be calculated to be 1308211. We know that the input pulse should have a PDW with a pulse width count around 10, and by observing the two consecutive pulses with shorter PW values; we know that a “chopping” of the PDW occurred. There appears to be a single clock cycle in which the pulse tracking logic did not find a magnitude squared value that crossed the high threshold, and this would have caused the PW with a 5 count to be sent to the output of the receiver. However, the next cycle, of 158208, found the magnitude squared value to again cross
the threshold and begin another PDW track until the end of the actual pulse. The idea to implement hysteresis was motivated by these observations of incorrect pulse width reports that are caused by “chopping” of the pulse.

Hysteresis was implemented in the receiver using the following logic. First, the magnitude squared values must cross the high threshold for two consecutive cycles within plus or minus tow bins of the initial crossing bin. Second, the value of approximately one half the high threshold is computed and used as the hysteresis threshold. Thirdly, the hysteresis threshold is used only for frequency bins plus or minus two bins of the frequency bin matching the second FFT cycle, which we now know is being used to more accurately track the signal. Therefore, after the initial detection of a signal, the hysteresis threshold is used for comparisons until the end of a pulse is found. The implementation of the hysteresis threshold was successful in greatly reducing the occurrence of “chopped” pulses, and provided a much more consistent pulse width measurement for the duration of any input signals.

A sample of this hysteresis thresholding is shown in Pseudo-Code below. The blocks marked with “etc.” are very similar to the previous section’s logic.

-------------- Hysteresis Thresholding Pseudo-Code --------------------

hysteresis_threshold = high_threshold / 2

IF f1_data_valid = 1
    IF primary_peak > hysteresis_threshold
        IF primary_peak_bin = f1_frequency_bin
            f1_data_valid = 1
            f1_pw = f1_pw + 1
            f1_toa = f1_toa
            f1_frequency_bin = f1_frequency_bin
        ELSE
            output = f1_PDW
            f1_data_valid = 0
            f1_pw = 1
            f1_toa = current_toa
            f1_frequency_bin = primary_peak_bin
        END
    ELSE
        f1_data_valid = 0
        f1_pw = 1
        f1_toa = current_toa
        f1_frequency_bin = primary_peak_bin
    END
ELSE
  IF primary_peak_bin = f1_frequency_bin
    f1_data_valid = 1
    f1_pw = f1_pw + 1
    f1_toa = f1_toa
    f1_frequency_bin = primary_peak_bin
  ELSE
    f1_data_valid = 0
    f1_pw = 1
    f1_toa = current_toa
    f1_frequency_bin = primary_peak_bin
  END
END
ELSEIF primary_peak > high_threshold
  etc.
END

IF f2_data_valid = 1
  IF secondary_peak > hysteresis_low_threshold
    IF secondary_peak_bin = f2_frequency_bin
      f2_data_valid = 1
      f2_pw = f2_pw + 1
      f2_toa = f2_toa
      f2_frequency_bin = f2_frequency_bin
    ELSE
      output = f2_PDW
      f2_data_valid = 0
      f2_pw = 1
      f2_toa = current_toa
      f2_frequency_bin = secondary_peak_bin
    END
  ELSE
    IF secondary_peak_bin = f2_frequency_bin
      f2_data_valid = 1
      f2_pw = f2_pw + 1
      f2_toa = f2_toa
      f2_frequency_bin = secondary_peak_bin
    ELSE
      f2_data_valid = 0
      f2_pw = 1
      f2_toa = current_toa
      f2_frequency_bin = secondary_peak_bin
    END
  END
ELSEIF secondary_peak > low_threshold
  etc.
END

----------- End Hysteresis Thresholding Pseudo-Code -----------
3.5 Update #4: Second Improvement to Signal Tracking

The frequency selection and tracking logic was previously improved upon to account for the cases of two simultaneous signals overlapping in any manner in the time domain. The parameters of interest to the tracking algorithm, if they exist, include the current maximum frequency bin and magnitude, the second maximum frequency bin and magnitude, the previous frequency bin, pulse width, time of arrival and whether or not it was valid, the second previous frequency bin, pulse width, time of arrival and whether or not it was valid, along with the high threshold value and low threshold value.

Previously, if a magnitude squared value crossed the high threshold, it was assumed only one signal was present. If the magnitude squared of the new peak signal did not cross the high threshold but crossed the lower threshold, a second signal was searched for.

The major change proposed was to use the lower threshold as a noise threshold, rather than a second signal threshold. Therefore, the implementation of the tracking logic would only change minimally, but the reasoning for selection of the lower threshold was now based upon the receiver environment. Previous attempts to tweak the high and the low thresholds were based on observed results from one test to the next.

3.6 Update #5: Continuous Wave (CW) Signal Reporting

The improvements to signal tracking and frequency selection resulted in a more accurate pulse descriptor word (PDW) being generated at the end of every pulse incident upon the system. However, the simple decision to output a PDW only at the end of a
pulse had a very clear shortcoming, the possibility of continuous wave (CW) signals input to the receiver. The theoretical and observed behavior of the Monobit-II receiver to a CW input signal was a problem, simply because the receiver did not output anything while the CW signal was present. As soon as the CW source was turned off, the receiver would output a PDW with an arbitrary pulse width, but a correct TOA and frequency bin. The pulse width would likely be random because the 16 bits for a pulse width count was equivalent to the following maximum time constraint.

\[ T_{\text{max}} = (2^{\text{pw bits}}) \times (100\text{ns}) = 6.5536 \text{ ms} \]

This means that any pulse greater than 6.5536ms would cause the pulse width counter to roll over. It is possible that a calculation based on the time of arrival minus the current Real Time Clock output could find an actual pulse width. However, the decision was made to create a maximum pulse width that once this pulse width value was reached, a PDW would be output from the system. Therefore, CW signals would be output every time the pulse width count reached this maximum, and also abnormally long pulses would also create more than one PDW report that must be recognized in later process to belong to the same input.

The decision was made to report a pulse once its pulse width reached approximately 50us. The implementation of this logic was to monitor the pulse width value, and once the counter reached 511, or “0000 0001 1111 1111,” the PDW would be output and the frequency detection and tracking logic must once again find the signal to cross the detection threshold for two consecutive cycles. This counter threshold corresponds to an actual maximum pulse width that must be less than or equal to 51.1us before a PDW is generated. The additional logic was reliable for outputting a PDW when
a CW signal was present, and it only slightly increased the pulse tracking algorithm complexity by requiring an additional check of the pulse width value during each pass.

3.7 Update #6 (final): Ratio Threshold Implementation

The final update to the Monobit-II receiver VHDL in the stage four of processing was the implementation of a Ratio Threshold algorithm. The basic theory behind the Ratio Threshold algorithm was that it would allow for improved probability of detection for both individual and simultaneous pulses while at the same time decreasing the probability of a false detection, otherwise referred to as the false alarm rate.

A ratio of the secondary peak signal to the primary peak signal was studied to determine if the spurious signal ratio from a single signal input could be differentiated from a true two signal input ratio. MATLAB simulations were able to show that a ratio between 0.6 and 0.4 would eliminate virtually all of the false detections for single signal input cases. Actual FPGA configuration files were then created based upon the ratio threshold theory. The algorithm would take the largest magnitude squared value every FFT cycle and reduce it by the ratio threshold amount, resulting in a variable threshold for each cycle. Approximate values were used when testing the 0.6, 0.5, and 0.4 ratio settings. A ratio of 0.5 is a reduction by one half and this was closely approximated by taking a one bit shift of the peak value. If the magnitude has 14 bits represented by $Bits(13:0)$, the variable ratio threshold $R_{0.5}$ would be as follows.

$$R_{0.5} = Bits(13:1)$$

The faction of the total value is equivalent to
Here \( n \) is bit shift and \( k_x \) are the possible values for \( n \).

\[ R_{tot} = 2^{-1} = 0.5 \]

The other ratio values were obtained by different combinations of bit shifting operations summed together. A ratio close to 0.6 was obtained from a 1 bit shift, plus a 4 bit shift, plus a 5 bit shift.

\[ R_{0.6} = \text{Bits}(13:1) + \text{Bits}(13:4) + \text{Bits}(13:5) \]

The actual fraction of the total for this case would be:

\[ R_{tot} = 2^{-1} + 2^{-4} + 2^{-5} = 0.59375 \]

The ratio 0.4 was obtained from a 2 bit shift, a 3 bit shift and a 5 bit shift of the peak magnitude.

\[ R_{0.4} = \text{Bits}(13:2) + \text{Bits}(13:3) + \text{Bits}(13:5) \]

The actual fraction of the total magnitude would be:

\[ R_{tot} = 2^{-2} + 2^{-3} + 2^{-5} = 0.40625 \]

The fraction of the total magnitude calculations were selected based on the fewest number of sub-shifting operations that would be required within each processing cycle. There was uncertainty surrounding the appropriate ratio, due to the number of variables in the MATLAB simulations contributing to the selection of a single ratio. The appropriate range was suggested to be between 0.6 and 0.4, so actual hardware tests were performed with the FPGA reconfigured for each of the three ratios detailed above. Due to the fact that 0.4 was the lower limit of the suggested range, the plan was to not go any lower. However, the 0.4 configuration performed very well, and a configuration was developed with an approximately 0.3 ratio threshold. The ambitious lowering of the
threshold to 0.3 did result in a higher false alarm rate, proving that 0.4 was a good lower limit, and an ideal operating range that would not sacrifice the receiver sensitivity.

Pseudo-Code for the final implementation of a ratio threshold with CW detection is shown below.

```
--------------- Ratio Thresholding with CW Detection ------------------

CW_PW_THRESHOLD = "111111111"

IF primary_peak > noise_threshold

    IF f1_data_valid = 1
        IF primary_peak_bin = f1_frequency_bin
            IF f1_pw > CW_PW_THRESHOLD
                output = f1_PD
                f1_data_valid = 0
                f1_pw = 1
                f1_toa = current_toa
                f1_frequency_bin = primary_peak_bin
            ELSE
                f1_data_valid = 1
                f1_pw = f1_pw + 1
                f1_toa = f1_toa
                f1_frequency_bin = f1_frequency_bin
            END
        ELSE
            f1_data_valid = 0
            f1_pw = 1
            f1_toa = current_toa
            f1_frequency_bin = primary_peak_bin
        END
    ELSE
        output = f1_PD
        f1_data_valid = 0
        f1_pw = 1
        f1_toa = current_toa
        f1_frequency_bin = primary_peak_bin
    END
ELSE
    output = f1_PD
    f1_data_valid = 0
    f1_pw = 1
    f1_toa = current_toa
    f1_frequency_bin = primary_peak_bin
END
ELSE
    IF primary_peak_bin = f1_frequency_bin
        f1_data_valid = 1
        f1_pw = f1_pw + 1
        f1_toa = f1_toa
        f1_frequency_bin = primary_peak_bin
    ELSE
        f1_data_valid = 0
        f1_pw = 1
        f1_toa = current_toa
        f1_frequency_bin = primary_peak_bin
    END
END
END

ratio_threshold = primary_peak(13:1)
```
IF secondary_peak > ratio_threshold
    IF f2_data_valid = 1
        IF secondary_peak_bin = f2_frequency_bin

            IF f2_pw > CW_PW_THRESHOLD
                output = f2_PDW
                f2_data_valid = 0
                f2_pw = 1
                f2_toa = current_toa
                f2_frequency_bin = secondary_peak_bin
            ELSE
                f2_data_valid = 1
                f2_pw = f2_pw + 1
                f2_toa = f2_toa
                f2_frequency_bin = f2_frequency_bin
            END
        ELSE
            f2_data_valid = 0
            f2_pw = 1
            f2_toa = current_toa
            f2_frequency_bin = secondary_peak_bin
        END
    ELSE
        output = f2_PDW
        f2_data_valid = 0
        f2_pw = 1
        f2_toa = current_toa
        f2_frequency_bin = secondary_peak_bin
    END
ELSE
    IF secondary_peak_bin = f2_frequency_bin
        f2_data_valid = 1
        f2_pw = f2_pw + 1
        f2_toa = f2_toa
        f2_frequency_bin = secondary_peak_bin
    ELSE
        f2_data_valid = 0
        f2_pw = 1
        f2_toa = current_toa
        f2_frequency_bin = secondary_peak_bin
    END
END

------- End Ratio Thresholding with CW detection Pseudo-Code --------

The pseudo-code demonstrated for the different cases is extremely simplified. To give a sense of the multiple levels that would actually go into a stage four logic design, the below Pseudo-Code is provided. The below code does not include the variable assignments, only the multiple levels of “if” and “else” statements.
------------ Complex structure -----------------------

IF primary_peak > noise_threshold
    IF f1_data_valid = 1 AND f2_data_valid = 1
        IF primary_peak_bin = f1_frequency_bin - 2 OR
            primary_peak_bin = f1_frequency_bin - 1 OR
            primary_peak_bin = f1_frequency_bin OR
            primary_peak_bin = f1_frequency_bin + 1 OR
            primary_peak_bin = f1_frequency_bin + 2

        IF f1_pw > CW_PW_THRESHOLD
            IF secondary_peak > ratio_threshold
                IF secondary_peak_bin = f2_frequency_bin
                    ELSE
                        END
                    ELSE
                        END
                    END
                ELSE
                    IF secondary_peak > ratio_threshold
                        ELSE
                            END
                        ELSEIF secondary_peak > ratio_threshold
                            IF secondary_peak = f1_frequency_bin - 2 OR
                                secondary_peak = f1_frequency_bin - 1 OR
                                secondary_peak = f1_frequency_bin OR
                                secondary_peak = f1_frequency_bin + 1 OR
                                secondary_peak = f1_frequency_bin + 2

                            IF f1_pw > CW_PW_THRESHOLD
                                IF primary_peak_bin = f2_frequency_bin
                                    ELSE
                                        END
                                    ELSE
                                        IF primary_peak_bin = f2_frequency_bin
                                            ELSE
                                                END
                                            END
                                        ELSEIF secondary_peak = f2_frequency_bin - 2 OR
                                            secondary_peak = f2_frequency_bin - 1 OR
                                            secondary_peak = f2_frequency_bin OR
                                            secondary_peak = f2_frequency_bin + 1 OR
                                            secondary_peak = f2_frequency_bin + 2

                                        ELSE
                                            ELSE
                                                END
                                            END
                                        END
                                    ELSE
                                        IF secondary_peak > ratio_threshold
                                            ELSE
                                                END
                                            ELSEIF f1_data_valid = 1
                                                etc.
                                            ELSEIF f2_data_valid = 1
                                                etc.
                                            ELSE
                                                END
                                            END
                                        END
                                    END
                                ELSE
                                    IF secondary_peak > ratio_threshold
                                        ELSE
                                            END
                                        END
                                    END
                                END
                            END
                        END
                    END
                END
            END
        END
    END
END

-------------End Complex Structure Pseudo-Code ------------------
3.8 Final Configuration Notes

The final configuration of the stage 4 logic was also incorporated into a hardware design utilizing a novel data collection platform that took the place of the HP logic analyzer. This new data collection platform eliminated the 140 parallel logic lines by adding an output formatting section to the FPGA configuration. The new output format was to use an 8-bit width output word that basically serialized the PDW output. This 8-bit word was then converted to a USB packet and sent directly to the controlling PC. The details of this data collection system are beyond the scope of the current work, but the new setup did allow for direct collection and control of the Monobit-II receiver. This improved test fixture control was instrumental in quickly collecting multiple data sets for the many different configurations.
IV. DATA COLLECTION SETUP AND RESULTS

4.1 Test Setup Information

The initial testing of the Monobit-II receiver was performed with a pulsed signal input and a logic analyzer collecting the pulse descriptor word (PDW) output. The system was clocked at 2.5GHz, and data was collected with and without the band-pass filters surrounding the limiting amplifier. A pulse generator was used to modulate the signal generator output, and a personal computer (PC) would receive the logic analyzer data over a general purpose interface bus (GPIB) connector.

Initial test cases included a single pulsed signal input stepping across the second Nyquist zone of the receiver and two simultaneous pulsed signal inputs, one with a constant frequency and the second a stepped frequency. The logic analyzer was synchronized with the signal generators, such that every time the signal generator would change the input frequency, the logic analyzer would wait approximately one second for the signal generator to settle prior to looking for a trigger signal. The reason for waiting a finite amount of time before looking for a logic analyzer trigger was that the actual output of the signal generator when switching between frequencies was unpredictable, and any PDWs generated by the receiver would be irrelevant.

The logic analyzer would fill a single memory buffer of approximately $2^{14}$ or 16384 bytes. The output of the receiver was known to be two 56 bit PDWs, (PDW1 and PDW2) and a 32 bit Real Time Clock (RTC). This data format is shown in the figure below, and totaled 144 bits.
The Monobit-II receiver was designed to output the 9.765625MHz clock on
whose rising edge would coincide with an updated set of output data. This clock speed,
$CLK_{out}$ is equivalent to the main system clock, $CLK_{sys}$, divided by the Monobit FFT size,
$FFT_{size}$. Therefore, the receiver’s output data rate is directly proportional to the ADC
sampling clock.

$$CLK_{out} = \frac{CLK_{sys}}{FFT_{size}}$$

The system clock was later increased to 2.56GHz and the FFT size remained at
256, resulting in an output clock rate of 10MHz.

The initial testing consisted of logic analyzer data files being sent of the GPIB bus
to the PC, and after the successful receipt of a logic analyzer file, the data collection
program would change the signal generator output and the cycle would repeat. It was
previously mentioned that the data file size of the logic analyzer was 16384 bytes, and
each output, or line, from the receiver was 144 bits wide. Therefore, the maximum
number of receiver outputs, or lines, would be the data size, converted to bits, divided by
the output size.

$$Lines_{max} = \frac{TotalBytes \times 8}{DataOut_{size}}$$
The calculation results in a maximum number of lines to be 910.22. The possibility of multiple data sets being collected at each input was proposed, but the downside of transferring one data set, then waiting for a trigger for the next data set means that all the receiver outputs that occurred between the end of the transferred data set and the listening for a new trigger would all be lost. Therefore, the two data sets would not be continuous, and would be of little relevance to each other.

The decision was made to analyze the first 100 PDW outputs related to the actual input signal. This post processing was performed in MATLAB, and operated as follows. First, the data set would be read in, and the first frequency bin found would be compared to the expected frequency bin based on the signal generator setting. If this frequency bin matched the expected value, the time of arrival would be recorded. If the frequency bin did not match, frequency bin values would be examined until a match was found, and the corresponding TOA would be recorded. When the first TOA was found, the TOA for a pulse input 100 pulses later could be predicted, based on the pulse repetition interval (PRI) of the input signal. This expected ending TOA would have an error buffer of $err$ number of cycles.

$$TOA_{end} = \left( TOA_{first} \times NumPulses \times PRI_{us} \times 10 \right) + err$$

If the initial TOA was 2399022, the PRI was 10us, the number of desired pulses was 100, and the error buffer was 10, the TOA of the final pulse would be expected to be 2409032.

The MATLAB code would then read in PDW lines and check each TOA, and as long as the TOA values were less than $TOA_{end}$ the pulses would be saved.
After the collection of all the pulses between the first valid detection and the pulse output corresponding to the final TOA of interest were found, a statistical analysis of the receiver could be performed for the current data set. The probability of detection could be calculated, based on the number of correct signals detected. The false alarm rate, based on the number of false alarms could be found, or the total number of incorrect PDW outputs. The existence of multiple reports for a single pulse input, or chopping, could also be found.

A calculation performed prior to the probability of detection calculation is the determination of the number of PDW outputs that correspond to the same input pulse, and are actually a chopped PDW. The logic for finding this occurrence relies on knowing the actual pulse width, and compares two consecutive TOA values around the same frequency bin location. If two consecutive PDW outputs have the same frequency bin, plus or minus a buffer $f_{buf}$, then the TOA difference is calculated. If this TOA difference is less than the known input pulse width, $PW_{in}$, the two PDW reports are assumed to be from the same input pulse and are actually a receiver error indicating a chopped pulse.

\[
\text{TOA}_{\text{diff}} = \text{TOA}_{\text{last}} - \text{TOA}_{\text{current}} \\

P_{\text{chop}} = \sum \text{TOA}_{\text{diff}} < PW_{\text{in}}
\]

The actual number of pulse reports, $P_r$, is then the total number of different pulse detections, not the total number of pulse reports around a particular frequency bin of interest.

\[
P_{\text{rp}} = P_{\text{tot}} - P_{\text{chop}}
\]

An example input would be a signal with a pulse width of 1us, corresponding to a TOA count of 10. In this case, if there were 99 reports around the frequency bin of
interest, but in one case two consecutive PDW reports were a pulse width of 3, TOA of 2399222, and the second was a pulse width of 6, TOA of 2399226. Therefore, $TOA_{\text{diff}}$ would be 4, causing the $P_{\text{chop}}$ to be 1, and the total reported pulses to actually be 98 instead of 99.

This value can then be used for probability of detection calculations.

$$P_d = \left(1 - \frac{P_{\text{in}} - P_{\text{rp}}}{P_{\text{in}}} \right) \times 100\%$$

The probability of detection, $P_d$, is the number of input pulses, $P_{\text{in}}$, minus the number of pulses reported, $P_{\text{rp}}$, divided by the number of input pulses. An example is if 98 pulses were reported and 100 pulses were input, $P_d$ is 98% for that particular signal generator input.

$$FA\% = \left(\frac{P_{\text{tot}} - \left( P_{\text{rp}} + P_{\text{chop}} \right)}{P_{\text{tot}}} \times 100\% \right) = \frac{P_f}{P_{\text{tot}}} \times 100\%$$

This calculation of the false alarm rate (FAR), is actually an indication of the percentage of false reports from the receiver as a percentage of the total outputs, $P_{\text{tot}}$. The total number of pulses not corresponding to the input signal, $P_f$, is obtained by subtracting the total correct pulses from the total pulse count. If 102 PDWs were found between the first and final TOA indicators, and 99 of these PDWs were from the input signal, the false alarm percentage would be 2.94%.

### 4.2 Actual Data Collection Results

The first complete data sets collected from the Monobit-II receiver were taken in the fall of 2004. The configuration file used was the original VHDL delivered with the hardware. Up to this point, most of the laboratory testing was done by a visual inspection
of the logic analyzer output. If the frequency bin, pulse width, and time difference between TOA matched the signal generator settings, the system was assumed to be operating correctly. The dip switches for the threshold settings were also adjusted while watching the logic analyzer output. A reliable low threshold, or noise threshold, was settled upon when the system would not output any false alarms when the signal generators were off. The higher threshold, or detection threshold, was constantly tweaked, mainly because the false detections of a second signal were directly proportional to this threshold. However, when two simultaneous signal were input to the system, this threshold to eliminate false second signal detections turned out to be much too high for any detections with two simultaneous signals.

The theoretical reason for this high threshold needing to vary based on the number of input signals could be traced directly to the limiting amplifier. The fact that the limiting amplifier would drive its input signal voltage to the full swing output voltage means that when two signals of equal power are combined at the input of a limiting amplifier, the output voltage is still the full swing, but the two signals are sharing this energy. Data collected from on the FPGA with the Xilinx logic analysis cores from ChipScope Pro were able to show the frequency bin values calculated by the Monobit-II receiver on chip. Figure 4.1 shows the ChipScope Pro data for all 128 frequency bins displayed in MATLAB. The input for this test was a single signal pulse at 1894MHz.
The same ChipScope Pro data collection program was used, and the input was changed to simultaneous pulses at 1894MHz and 1474MHz. The MATLAB plot of this data is shown below in figure 4.2.
The drastic change in magnitude squared values can be observed by comparing figures one and two. The peak value in figure 1 is over 2400, while the peak values in figure 4.2 are near 1100 and 600.

The MATLAB simulations of this phenomenon also agreed upon the fact that simultaneous signal could possibly reduce by half their magnitude squared values from that of a single signal input case. Figure 4.3 shows a MATLAB simulation of the FFT output magnitude squared. The simulated input signal was a 1950MHz signal, and this magnitude squared value corresponds to the ChipScope Pro data peak near 2500.
A MATLAB simulation of two simultaneous pulse signal inputs was also performed, and the result is shown in figure 4.4.
The observable difference between the MATLAB magnitude squared data and the real collected data is that there are more peaks visible in the simulation. This is due to the fact that on the actual receiver, there was a thresholding step performed immediately after the magnitude squared calculation, followed by a zeroing of the neighboring bins. This thresholding and zeroing was not performed on the simulated data prior to the generation of the plot, but the major point of study was the peak value fluctuation.

Originally, the system clock was set at 2.5GHz, creating a frequency bin size of 9.765625MHz. The performance of the receiver for two signal cases was found to vary depending upon the location of the input signals relative to the 9.766MHz bin. A signal
in the middle of a frequency bin would have a higher magnitude squared value, and therefore cross a higher threshold, than a signal input between two frequency bins. This difference in magnitude squared values is common for all frequency transformations, due to signals between two bins having their energy split between these bins. This variation of performance with respect to frequency bin location was observed most drastically in the receiver for two signal cases, due to the combination of energy sharing within the limiting amplifier and energy splitting between two frequency bins.

MATLAB simulations of the FFT magnitude squared outputs were used to study the fluctuation of the peak values and secondary peak values as they changed with respect to each FFT cycle performed on the same input, along with how the peaks changed due to changes in frequency. Figure 4.5 shows a stepped frequency input from 1375MHz to 2375MHz with 200kHz steps. At each frequency input approximately ten FFT calculations were performed, with a random initial phase generated for the input signal. After each FFT calculation, the peak value and the second highest peak were saved. Then after the ten FFT calculations at each frequency, the highest and lowest first peak values were saved, along with the highest and lowest secondary peak values.
The colors of the points in figure 4.5 correspond to the variations of the primary peak values and secondary peak values for each frequency step. The dark blue “x” is the highest primary peak value, and the light blue triangle is the lowest primary peak. The green square is the highest secondary peak value, and the black star is the lowest secondary peak. Figure 4.6 shows a closer view of the same data as figure 4.5, with the area from approximately 1530MHz to 1650MHz visible. Figures 4.5 and 4.6 were from a simulation with the sampling frequency set to 2.5GHz. Later figures will show this same data with the sampling rate changed to 2.56GHz.
It is important to see that in both figures 4.5 and 4.6 that in some cases the green squares denoting the peak spurious signal level are higher than the light blue triangles denoting the minimum peak signal level of different frequencies. This shows an inherent problem of a static threshold that must sacrifice the probability of detection or increase the false signal detections.

Later simulations adjusted to the change of the receiver hardware to an input clock of 2.56GHz, and figure 4.7 shows the same data as figure 4.5 when this change was implemented.
Figure 4.7: MATLAB Simulation of Changes in Magnitude Squared Peaks, Fs = 2.56GHz

Figure 4.8 is a close up view of the data from figure 4.7, but over a wider selection of frequencies than figure 4.6. Here the range of data visible is from 1500-1700MHz
An interesting point of emphasis to study in figure 4.6 and figure 4.8 is the presence of an unusually high spurious peak in both plots. In figure 4.6, when the sampling frequency is 2.5GHz, the peak was at 1.5625GHz, and in figure 4.8, the sampling frequency is 2.56GHz and the peak is at 1.6GHz. The point that this peak manifests itself is 5/8 the sampling frequency for both cases. A similar unusually high spurious peak was found at 7/8 the sampling frequency for both cases. A theory proposed by the author is that the major harmonic spurs are all folded into the same frequency bin at these fractions of the sampling frequency.

The problem of needing different thresholds depending upon the number of input signals was originally pushed outward as needing to be solved at a later date. The initial intent was to collect data with the optimal threshold settings, knowing the number of

Figure 4.8: MATLAB Simulation of Changes in Magnitude Squared Peaks, 1.5-1.7GHz
input signals. Data from an early test is shown below. Here one signal generator was kept at 2211MHz, while a second signal generator swept across the receiver’s bandwidth, from 1375MHz to 2375MHz. The output power of the signal generators was typically kept between -50dBm and -3dBm, simply to remain in the optimal operating region of the limiting amplifier. The test results displayed below are with a -3dBm signal level from both generators.

Figure 4.9: Fall 2004 Test, 2211MHz constant signal, 1375-2375MHz sweep, 3D view

Figure 4.9 shows a three dimensional mesh plot from MATLAB. The x-axis shows the input frequency range, and the y-axis shows the calculated frequency from the output PDW. The pulse repetition interval (PRI) was set at 5us, and it was desired that 100 pulses be input at each frequency step. Although the pulse generator was not set to
output 100 pulses exactly, post processing of the collected data could provide a virtual
cutoff for data collection. The previously defined $TOA_{end}$ calculation would tell us the
approximate TOA of a pulse $n$ number of pulses later meant that by checking the TOA of
each pulse in a data file, the desired number of pulses could be studied with the post-
processing analysis programs.

Figure 4.9 shows the number of PDW outputs between the first and 100th PDW of
the input signal, including the number of swept signal PDWs and false PDWs. This total
number of outputs per bin is the value on the z-axis. A perfect output would show a
diagonal line with a constant magnitude of 100 for the swept frequency, and a strait line
with a constant magnitude of 100 at the constant frequency bin. However, it can be seen
in figure 4.9 that the diagonal and strait lines are present, but they are not at a constant
value of 100. Also, figure 4.9 shows a number of false signal detection PDW outputs
when the swept signal and constant signal are the same. It should be noted that although
the unevenness of the constant and variable lines seems to imply missed detections, it is
also possible that some error in frequency estimation occurred, as was discussed in
Chapter 3 regarding the original frequency selection logic. The post-processing program
allowed for some error in frequency estimation, corresponding to the estimation error
within the FPGA code of +2 bins or -2 bins from the expected value. This variation in
the PDW frequency bin means that the number of frequency bin reports could be spread
out in figure 4.9, but the system was still credited with a high probability of detection
even if it does not appear so in this figure.

A second visualization of this data, along with more detailed analysis, is provided
in figure 4.10. The upper left plot shows the probability of detection ($P_d$) of the constant
signal (dark blue) and the number of pulses that were chopped (cyan), also thought of as multiple PDWs of the same input signal. The lower left plot shows the same information but for the swept signal (red and magenta, respectively). The upper right plot shows the average pulse width calculation (us) for both the constant and swept signals. The lower right plot shows the number of PDWs that were a false detection. All the plots are with respect to the output file at each stepped frequency.

Figure 4.10: Fall 2004 Test, 2211MHz constant signal, 1375-2375MHz sweep, 4 data plots
The important metric, probability of detection ($P_d$) is also calculated for the whole sweep as an average of all the $P_d$ values. It is obvious that there is a large variation in PoD as the second signal is swept across the receiver’s bandwidth. It can be observed that the swept frequency shows a probability of zero when it overlaps the constant frequency. This drop in $P_d$ is due to the zeroing of bins +2 and -2 on either side of the peak detection. The fact is that no secondary signal is output, so only one signal could be considered a successful detection.

The average pulse width for all the PDW outputs at each frequency step can be seen to correspond to the amount of chopping at that same input frequency. The lower probability of detection also affects the pulse width average as it would be likely that a pulse would only be found during part of its on-time.

The total number of false detections is also a very important metric and an area of concern if any false detections are reported. The fact that in some cases, almost the same number of false detections were created as valid detections, was a cause for major concern. The majority of these occurrences corresponds to an overlap of the two signal inputs and the assumption was that either a product of the mixing of the two signals, or a spurious signal that would be cutoff with a higher single signal threshold was being seen. A second example of an original data collection is shown below in figure 4.11.
Figure 4.11: Fall 2004 Test, 1474MHz constant signal, 1375-2375MHz sweep, 4 data plots

The format of the data displayed in figure 4.11 is identical to figure 4.10, but for this set of data the constant signal was at 1474MHz. The constant signal probability of detection was slightly lower, at 83.89%, while the swept signal probability of detection was higher, up to 93.67%. The false alarms also decreased to an average of 2.66, and it is visible that less chopping occurred for both constant and variable frequency pulses.

One of the first improvements, as discussed in chapter 3, was to use the second FFT cycle as the frequency bin to report and monitor. The three dimensional PDW output view, similar to figure 4.9, is shown below in figure 4.12 after the implementation of the new frequency bin monitoring logic. The obvious improvement is that the strait line
corresponding to the constant signal and the diagonal line corresponding to the swept signal have both leveled off drastically compared to figure 4.9.

Figure 4.12: 2004 Improved Frequency Binning, 1474MHz constant signal, 1375-2375MHz sweep, 3D view

However, the presence of some chopping and false detections are also visible in figure 4.12, and a four plot representation of this data is shown below in figure 4.13. A number of chopped PDW outputs appear for the swept signal as it reached the higher end of the receiver bandwidth. It was discovered later that the signal generator output power decreased slightly as the output frequency increased. Across the receiver bandwidth, from 1.375GHz to 2.375GHz, the signal generator output power actually decreased almost 1dB, although the output power reading on the signal generator remained the
same. Later testing utilized a different signal generator with a smaller amount of loss as frequency output increased. Figure 4.13 also shows a higher probability of detection for both constant signal input and variable signal input, due to the slight adjustment of the detection threshold. The lowering of the detection threshold because a two signal input was known to not be practical, and the consequence was a large number of false alarms at the two signal overlap point, showing the downfall of this lower threshold.

Figure 4.13: 2004 Improved Frequency Binning, 1474MHz constant signal, 1375-2375MHz sweep, 4 data plots

A single signal input case is shown below in figure 4.14 after the implementation of improved frequency tracking. Chopping is visible, along with false alarms at
1.562GHz and 2.187GHz, which are approximately the signal inputs at 5/8 the sampling frequency and 7/8 the sampling frequency, seen here in MATLAB simulations.

The next improvement to the receiver detection algorithm was the implementation of hysteresis after the detection of a signal. The motivation and background was discussed in chapter 3, but basically the chopping of the PDW output was undesirable, and hysteresis helped solve this problem.

Figure 4.15 shows the output in a three dimensional view after hysteresis was implemented. The threshold was reduced to the optimal two signal detection point.
Figure 4.15: 2004 Hysteresis, 1474MHz constant signal, 1375-2375MHz sweep, 3D view

The four plot summary of the data for figure 4.15 is shown in figure 4.16.
The main improvement seen in figure 4.16 over figure 4.13 is the virtual elimination of chopped pulses for both the constant frequency and the variable frequency. The probability of detection remained steady, and there was a slight reduction in the false detections. However, the fact remains that the threshold was tweaked knowing that two signals would be present, and when the signals overlapped the false detections skyrocketed.

Figure 4.17 shows a single signal input after the threshold is raised expecting only a single signal. It is important to note that when two signals were input to the receiver and the threshold was at the optimal single signal setting such as for figure 4.17, the two
signal inputs resulted in no outputs from the receiver. This probability of detection equivalent to zero is obviously very bad, so a push to find a static threshold somewhere between the optimal two signal setting and the optimal single signal setting was researched.

Figure 4.17: 2004 Hysteresis, 1375-2375MHz sweep, 4 data plots

It was noted earlier that the setting for optimal two signal detection would result in a large number of false alarms if only a single signal was present. Figure 4.18 shows this case after the improved frequency detection and hysteresis were implemented.
Figure 4.18: 2004 Hysteresis, 1375-2375MHz Sweep, 3D view, two signal threshold

It is seen in figure 4.18 that setting a lower threshold and expecting two input signals is dangerous when only a single signal arrives. Here there is a false detection for every true detection, and this can be thought of as a 50% false detection rate, or 100 false alarms for every 100 detections.

The actual settings on the dip switches for the optimal high threshold, or detection threshold, were 128 for two signal inputs and 384 for single signal inputs. A high threshold of 240 was chosen as a compromise of these optimal settings. Figure 4.19 shows the three dimensional view of the two signal input performance at the new threshold setting. Figure 4.20 follows this with the same data presented in the four plot format. The data set shows a consistent detection of both input pulses, 99% for the
constant frequency and 90% for the swept frequency. The false alarm rate is very low, even when the signals overlap it does not drastically increase.

Figure 4.19: 2005 New Threshold, Constant Signal 1474MHz, 1375-2375MHz Swept Input, 3D view
The obvious test is how well this new threshold would perform with a single signal after observing the success at two signal detection. Figures 4.21 and 4.22 show the 3D view and the four plot results, respectively.
Figure 4.21: 2005 New Threshold, 1375-2375MHz Swept Input, 3D view

Figure 4.22: 2005 New Threshold, 1375-2375MHz Swept Input, 4 plots
Studying figures 4.21 and 4.22 will show that although the single signal input is detected 100% of the time, there are some obvious problem areas centered around 1.562GHz and 2.187GHz with respect to false alarms. There was an overall reduction in the false alarms compared to the results shown in figure 4.18, and the fact that the threshold was not changed between figures 4.19 and 4.22 allowed for a very reasonable receiver performance metric.

The improved capability to detect continuous wave (CW) signals was then added, and the system was very reliable when a single CW signal was present. The system would generate a PDW every 511 cycles, indicating the correct frequency, TOA, and a pulse width of 51.1us. This pulse width would be an indicator, along with other post-processing techniques, to show that a signal with a pulse width longer than 51.1us was present, possibly indicating a CW signal. A secondary input signal could also be detected, as seen in the 3D view seen in figure 4.23 below.
In figure 4.23, the shorter light blue line at 1858MHz indicates that a signal was detected consistently, but not as many pulses were reported as the swept frequency input. This is the desirable result, knowing that the pulsed signal PDW should occur approximately five times more often, since the pulse repetition interval was 10us.

4.3 The Ratio Threshold Implementation

A configuration file was created with a ratio threshold set to 0.6, based on the MATLAB simulations suggesting this level as good tradeoff between sensitivity and false alarms. The simulation monitored every threshold calculation, and the initial assumption was to set the hardware threshold above the calculated level. The assumption
did not take into account the fact that for a signal to be considered valid, the secondary peak, or ratio threshold crossing, must occur for two consecutive cycles. This ability to absorb a single ratio crossing allowed for the study of ratio settings below the 0.6 point. This evaluation of lower thresholds, with special attention given to any increase in false alarms, was the motivation to create additional configuration files were the ratio threshold was lowered slowly. Figure 4.24 shows the ratio set to 0.6 and the resulting probabilities of detection and number of false detections.

Figure 4.24: Spring 2007, Ratio Threshold = 0.6, 1375-2375MHz Swept Input, 4 plots

The impressive results shown in figure 4.24 can be highlighted by the two lower plots. The lower left plot shows the swept frequency response of 100% probability of
detection. The lower right plot shows the false detection occurrences to be zero for all inputs. The results are promising, but it was known that any lowering of the ratio threshold that could be tolerated would benefit the two signal detection performance of the system.

The next step was the testing of a configuration where the ratio threshold was lowered to 0.5. Figure 4.25 shows the 4 plot result of an input signal sweep with this setup.

![Figure 4.25: Spring 2007, Ratio Threshold = 0.5, 1375-2375MHz Swept Input, 4 plots](image)

The 0.5 ratio configuration displayed the same promising results as the 0.6 ratio setting. A 100% PoD and a 0% FAR were enough to encourage a configuration file with
a ratio of 0.4 to be created. Dr. Tsui suggested that a threshold for the FAR, defined here as the percentage of signals reported that are incorrect, to be below 0.1%. The false PDW was therefore the key metric monitored as the ratio threshold setting was lowered. Figure 4.26 shown below is the 0.4 ratio output.

![Figure 4.26: Spring 2007, Ratio Threshold = 0.4, 1375-2375MHz Swept Input, 4 plots](image)

The resulting calculations from the 0.4 ratio threshold configuration were just as good as 0.6 and 0.5 for both PoD and FAR. A 100% PoD and a 0% FAR calculation were maintained. The possibility to further lower the ratio was explored, and a configuration using a 0.3 ratio was created. The resulting data collection summary is shown in figure 4.27.
Figure 4.27: Spring 2007, Ratio Threshold = 0.3, 1375-2375MHz Swept Input, 4 plots

Figure 4.27 shows that when the ratio threshold is lowered to 0.3, the false alarm rate increases beyond the desirable setting 0.1% to 0.15%. There are also issues regarding the probability of detection, as it has fallen slightly to 99.92%. The fact that a lower limit for the ratio threshold setting was found to be at 0.4 was surprising, due to the initial simulation suggestions. However, as more tests were performed with this configuration, the reliability was maintained, and 0.4 was determined to be the best ratio threshold configuration.

Figure 4.28 is a final example of the performance of the 0.4 ratio, showing the 3D view of the output PDW data when two simultaneous signals are input to the receiver.
The tables below account for a majority of the data collections performed on the Monobit-II receiver. The data collection’s dated name, threshold settings, probabilities of detection and false alarm rates are all provided. It is important to note that although the single signal and two signal cases look promising, often the thresholds were manually changed between these data collections to get the optimal results. The single signal input case where no thresholds were changed is the “041201_freq_dat_t2_sweep” data set. The high FAR is an indication of how the optimal two signal setting would operate in the single signal case. In the “05xxxx” sets, the same threshold was used.
Table 4.2: Initial Test and Evaluation Results

<table>
<thead>
<tr>
<th>Data Set Date and Name</th>
<th>F1 POD</th>
<th>F2 POD</th>
<th>FAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>041104_new_1474</td>
<td>97.46</td>
<td>96.6798</td>
<td>2.7316</td>
</tr>
<tr>
<td>041104_old_1474</td>
<td>83.4136</td>
<td>96.0934</td>
<td>1.4621</td>
</tr>
<tr>
<td>041201_freq_dat_neg70_sweep</td>
<td>15.5944</td>
<td>n/a</td>
<td>0.36411</td>
</tr>
<tr>
<td>041201_freq_dat_t2_sweep</td>
<td>99.9915</td>
<td>n/a</td>
<td>49.9777</td>
</tr>
<tr>
<td>041201_freq_dat_th3_1474</td>
<td>99.2892</td>
<td>97.7642</td>
<td>0.05989</td>
</tr>
<tr>
<td>041201_freq_dat_th3_sweep</td>
<td>99.9995</td>
<td>n/a</td>
<td>5.7846</td>
</tr>
<tr>
<td>041202_freq_dat_01dec_1938</td>
<td>98.7453</td>
<td>99.2058</td>
<td>3.5518</td>
</tr>
<tr>
<td>041202_freq_dat_30nov_hyst5</td>
<td>99.3457</td>
<td>99.4396</td>
<td>2.5829</td>
</tr>
<tr>
<td>041202_freq_dat_neg55_sweep</td>
<td>89.6543</td>
<td>n/a</td>
<td>0</td>
</tr>
<tr>
<td>041202_freq_dat_neg57_sweep</td>
<td>80.2158</td>
<td>n/a</td>
<td>0</td>
</tr>
<tr>
<td>041203_freq_dat_neg55_sweep</td>
<td>89.6543</td>
<td>n/a</td>
<td>0</td>
</tr>
<tr>
<td>041203_freq_dat_sweep</td>
<td>99.9845</td>
<td>n/a</td>
<td>0.36015</td>
</tr>
<tr>
<td>041213_freq_dat_b2_hyst5_1474</td>
<td>99.3402</td>
<td>96.4361</td>
<td>2.2088</td>
</tr>
<tr>
<td>041213_freq_dat_b2_hyst5_1938</td>
<td>98.9795</td>
<td>96.5125</td>
<td>2.5692</td>
</tr>
<tr>
<td>041213_freq_dat_b2_prom_1474</td>
<td>85.6314</td>
<td>91.7393</td>
<td>1.0734</td>
</tr>
<tr>
<td>041215_freq_dat_b2_nhyt5_sweep</td>
<td>100</td>
<td>n/a</td>
<td>1.2583</td>
</tr>
<tr>
<td>041215_freq_dat_b2_wf_t2_hyt5_1474</td>
<td>99.3067</td>
<td>95.8077</td>
<td>2.0657</td>
</tr>
<tr>
<td>050322_freq_dat_sweep</td>
<td>100</td>
<td>n/a</td>
<td>5.1722</td>
</tr>
<tr>
<td>050323_freq_dat_1474</td>
<td>98.5909</td>
<td>90.9036</td>
<td>0.00820</td>
</tr>
<tr>
<td>050329_freq_dat_sweep</td>
<td>100</td>
<td>n/a</td>
<td>2.5463</td>
</tr>
<tr>
<td>050330_freq_dat_1474</td>
<td>98.494</td>
<td>91.7078</td>
<td>0.01337</td>
</tr>
<tr>
<td>050518_freq_dat_brd6_sweep</td>
<td>99.9635</td>
<td>n/a</td>
<td>2.918</td>
</tr>
<tr>
<td>050608_new4_sweep</td>
<td>99.957</td>
<td>n/a</td>
<td>2.0176</td>
</tr>
<tr>
<td>060524_NGC_hyst5_freq_dat_sweep</td>
<td>100</td>
<td>n/a</td>
<td>0.71211</td>
</tr>
</tbody>
</table>
Table 4.3: Recent Test and Evaluation Results

<table>
<thead>
<tr>
<th>Data Set Date and Name</th>
<th>POD1</th>
<th>POD2</th>
<th>FAR</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>20070205_formatted_2005_sweep_-65dBm</td>
<td>86.16</td>
<td>54.13</td>
<td>0.001</td>
<td>0.6</td>
</tr>
<tr>
<td>20070206_f1_sweep_-59dBm_f2_2000MHz_-60</td>
<td>83.94</td>
<td>76.95</td>
<td>0.004</td>
<td>0.6</td>
</tr>
<tr>
<td>20070207_f1_sweep_-58dBm_f2_2000MHz_-60Bm_synced</td>
<td>87.74</td>
<td>52.97</td>
<td>0.002</td>
<td>0.6</td>
</tr>
<tr>
<td>20070207_f1_cwsweep_-60dBm_f2_cw2000_-60</td>
<td>81.12</td>
<td>95.67</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>20070208_f1_sweep_-60dBm_f2_2000_-60_hyst5</td>
<td>83.94</td>
<td>76.95</td>
<td>0.004</td>
<td>0.6</td>
</tr>
<tr>
<td>20070209_f1_sweep_-60dBm_f2_2000_-60_hyst5</td>
<td>61.29</td>
<td>91.74</td>
<td>0.002</td>
<td>0.6</td>
</tr>
<tr>
<td>20070209_f1_sweep_-60dBm_f2_2000_-60_hyst5</td>
<td>99.39</td>
<td>n/a</td>
<td>0.006</td>
<td>0.4</td>
</tr>
<tr>
<td>20070209_f1_sweep_11MHz_-20dBm_hystrat0.5</td>
<td>100</td>
<td>n/a</td>
<td>0.184</td>
<td>0.4</td>
</tr>
<tr>
<td>20070209_f1_sweep_11MHz_-20dBm_hystrat0.5</td>
<td>100</td>
<td>n/a</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>20070209_f1_sweep_11MHz_-20dBm_hystrat0.6</td>
<td>100</td>
<td>n/a</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>20070209_f1_sweep_11MHz_-20dBm_hystrat0.6</td>
<td>99.04</td>
<td>n/a</td>
<td>0.351</td>
<td>0.5</td>
</tr>
<tr>
<td>20070209_f1_sweep_-60dBm_f2_2000_-60Bm_sync_hystrat0.5</td>
<td>88.41</td>
<td>27.26</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>20070212_f1_sweep_-60dBm_f2_2000_-60Bm_sync_hystrat0.4</td>
<td>81.87</td>
<td>99.79</td>
<td>0.005</td>
<td>0.4</td>
</tr>
<tr>
<td>20070212_f1_sweep_-3dBm_f2_2000_-3dBm_sync_hystrat0.4</td>
<td>94.87</td>
<td>99.65</td>
<td>0.009</td>
<td>0.4</td>
</tr>
<tr>
<td>20070212_f1_sweep_-60dBm_f2_2000_-60Bm_sync_hystrat0.4_newthresh</td>
<td>93.6</td>
<td>99.8</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070212_f1_sweep_-3dBm_f2_2000_-4dBm_newthresh_hystrat0.4_sync</td>
<td>99.3</td>
<td>n/a</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070215_f1_sweep_-59dBm_f2_2000_-60Bm_newthresh_hystrat0.4</td>
<td>93.67</td>
<td>97.22</td>
<td>0.001</td>
<td>0.4</td>
</tr>
<tr>
<td>20070215_f1_sweep_-58dBm_f2_2000_-60Bm_newthresh_hystrat0.4</td>
<td>98.19</td>
<td>89.74</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070216_f1_sweep_-3dBm_newthresh_hystrat0.3</td>
<td>99.24</td>
<td>n/a</td>
<td>0.145</td>
<td>0.3</td>
</tr>
<tr>
<td>20070216_f1_sweep_-60dBm_f2_2005_-60Bm_sync_hystrat0.4</td>
<td>97.31</td>
<td>67.77</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070216_f1_sweep_-3dBm_f2_1900_-4dBm_sync_hystrat0.4</td>
<td>97.95</td>
<td>85.73</td>
<td>0.014</td>
<td>0.4</td>
</tr>
<tr>
<td>20070221_f1_sweep_-60dBm_f2_2103_-60Bm_sync_hystrat0.4</td>
<td>94.74</td>
<td>84.93</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070222_f1_sweep_-60dBm_f2_1858cw_-60Bm_hystrat0.4</td>
<td>86.42</td>
<td>100</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070207_f1_sweep_-59dBm_f2_1738_cw_-60Bm_old_hystrat0.4</td>
<td>86.94</td>
<td>100</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>20070227_f1_sweep_-59dBm_f2_1738_cw_-60Bm_new_hystrat0.4</td>
<td>87.43</td>
<td>100</td>
<td>0</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Table 4.3 shows a large amount of data, all collected in the spring of 2007 with a USB interface program written by Mr. Caleb Shreffler of AFRL/SN. The data processing was performed in MATLAB, and although data was collected when the signal generator input was outside of the receivers operating bandwidth, the tabulated data is only for the 1375-2375MHz range. Due to the complex nature of two signal dynamic range, and the variation in performance based upon the actual frequency inputs, the above table offers some insight into the raw performance of the receiver.

Changes in the power of one signal generator relative to another are provided such that a preview of the two signal dynamic range can be seen. Also shown are pulsed signal inputs when a CW signal is present. The actual signal generator power is described in the filename, but the cables, combiners, and connectors all contributed additional loss prior to the RF signal input on the receiver. The RF input can be considered the first bandpass filter, where losses could range from approximately 5dB at the lower end of the band, to almost 6dB at the high end of the band. It is important to calibrate out any losses between the signal generator output level and the true signal power level input to the receiver when determining the true performance of the receiver [11]. Overall the ratio threshold testing was very promising and was considered to show an acceptable receiver performance result.
V. CONCLUSION AND FUTURE WORK

Throughout the testing and evaluation of the Monobit-II receiver, we were constantly asking ourselves, “Can we modify the underlying algorithm to improve the performance of this system?” The answer to this question turned out to be “yes,” but there were physical limits to how far the current hardware could be pushed. The FPGA is reconfigurable, but the digitization of the analog signal to 2-bits was set in stone. The actual sampling rate for digitization was adjustable, but due to the hardwired clock dependence of the FPGA on the demultiplexed clock speed, there were absolute limits here as well.

The possibility for future work is very real, and there are multiple avenues to focus this development. By breaking the Monobit-II system down into multiple sub-systems, each area can be modified, but this will also affect the surrounding steps. The main driver of system performance is the analog to digital converter (ADC). By increasing the number of bits or the sampling rate, a more accurate representation of the analog signals could be created. However, increasing the number of bits input to the FPGA would mean that a larger logic footprint would result. Also, more bits would mean that the limiting amplifier may not be the best choice, and a linear amplifier would make more sense to take advantage of the increased number of digitization levels. The frequency domain detection logic would also be affected by more input bits and a modified analog amplifier, possibly leading to an increase two signal dynamic range and the ability to report the input signal amplitude. The tradeoff between additional
capabilities and performance versus a possible increase in the system’s required operating power for a new ADC would also have to be studied.

A second major area that drives the performance of the receiver is the actual Monobit FFT approximation. Simulations have been performed [10] showing that increasing the number of kernel points would lead to an increased frequency domain approximation. A change from the four point kernel to a twelve point kernel or higher primarily results in the increased logic footprint of the FFT approximation. There are also discussions in [10] to add automatic compensation to the frequency domain amplitudes across the spectrum. This compensation looks promising for increasing the two signal dynamic range, and when the proper conditions are met, allowing for three simultaneous signal detection.

Additional parameters could also be derived from the digitized signal, just as continuous wave signal detection was added. A very real possibility is the presence of modulated signals incident upon the receiver, and the ability to describe this modulation in the PDW output would be valuable. The fidelity of these modulation estimations would likely be affected by the digitization hardware and the Monobit FFT approximation kernel.

A very real improvement to the Monobit receiver system would be the creation of a silicon chip implementation of the FFT approximation and frequency detection logic. The first Monobit receiver [11] used this type of hardware, and the obvious tradeoff was the inability to reprogram the chip’s logic. Another option for hardware change would be a silicon or hybrid material chip containing both the ADC and receiver logic.
Finally, another obvious improvement to the Monobit-II receiver would be the updating of the actual FPGA device. Any improvements to the logic, along with future configurations, could take advantage of the rapidly advancing FPGA market with built-in processors and advanced I/O capabilities. The actual device speed that a new chip could operate at would complement any increases in the digitization rate, and the increase in programmable logic area would complement more complex designs.

Overall, it was extremely beneficial to work with the actual Monobit-II receiver hardware every day. The ability to quickly reprogram this system, along with the availability of on chip diagnosis tools such as ChipScope Pro were priceless whenever the system behaved in an unexpected manor due to a human miscalculation in logic design. The presence of Dr. Tsui throughout the testing, evaluation, and improvement of this system was invaluable to all of us involved in this work. The study of improved thresholding techniques for the Monobit-II receiver resulted in a novel algorithm that was able to increase the probability of detection while reducing the number of false detections from this important device.
REFERENCES


