Development of a real time bistatic radar receiver using signals of opportunity

Nicholas Rainville

Purdue University

Follow this and additional works at: http://docs.lib.purdue.edu/open_access_theses

Part of the Aerospace Engineering Commons, Electrical and Electronics Commons, and the Remote Sensing Commons

Recommended Citation
Rainville, Nicholas, "Development of a real time bistatic radar receiver using signals of opportunity" (2014). Open Access Theses. 671.
http://docs.lib.purdue.edu/open_access_theses/671

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Nicholas Rainville

Entitled
DEVELOPMENT OF A REAL TIME BISTATIC RADAR RECEIVER USING SIGNALS OF OPPORTUNITY

For the degree of Master of Science in Engineering

Is approved by the final examining committee:

James L. Garrison

John P. Sullivan

Stephen J. Katzberg

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University’s “Policy on Integrity in Research” and the use of copyrighted material.

James L. Garrison

Approved by Major Professor(s):

Approved by: Wayne Chen 07/23/2014

Head of the Department Graduate Program  Date
DEVELOPMENT OF A REAL TIME BISTATIC RADAR RECEIVER USING SIGNALS OF OPPORTUNITY

A Thesis
Submitted to the Faculty
of
Purdue University
by
Nicholas Rainville

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering

August 2014
Purdue University
West Lafayette, Indiana
This thesis is dedicated to my family
ACKNOWLEDGMENTS

I would like to thank my adviser, Professor James L. Garrison, for his guidance and support throughout my graduate studies. I would also like to thank the members of my committee Stephen J. Katzberg and John P. Sullivan for their input and improvements to this thesis.

Thanks to Rashmi Shah for her support with this work, as well as James Warnecke and Paul Flaherty at NOAA and Mark Beaubien at Yankee Environmental Systems. This research was supported by Exelis Inc through award number 12044169.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>ABBREVIATIONS</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Outline</td>
<td>3</td>
</tr>
<tr>
<td>2.1 History of Bistatic Radar Remote Sensing</td>
<td>5</td>
</tr>
<tr>
<td>2.2 GNSS Bistatic Radar Remote Sensing</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 History</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2 Continuing Work</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Signals of Opportunity Bistatic Radar</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Theory of Operation</td>
<td>9</td>
</tr>
<tr>
<td>2.5 Geometry of a Bistatic Remote Sensing System</td>
<td>11</td>
</tr>
<tr>
<td>2.6 Scattering Model</td>
<td>13</td>
</tr>
<tr>
<td>3 XM Radio</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Coverage Area</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Signal Structure</td>
<td>16</td>
</tr>
<tr>
<td>3.2.1 Auto Correlation properties of the XM Radio signal</td>
<td>17</td>
</tr>
<tr>
<td>4 SoOp RT Algorithm</td>
<td>21</td>
</tr>
<tr>
<td>4.1 Software model</td>
<td>23</td>
</tr>
<tr>
<td>5 SoOp RT Hardware</td>
<td>27</td>
</tr>
<tr>
<td>5.1 Signal Processing Overview</td>
<td>27</td>
</tr>
<tr>
<td>5.2 Hardware</td>
<td>28</td>
</tr>
<tr>
<td>5.2.1 Chassis</td>
<td>31</td>
</tr>
<tr>
<td>5.2.2 Hardware Details</td>
<td>32</td>
</tr>
<tr>
<td>5.2.3 Hardware Modifications</td>
<td>38</td>
</tr>
<tr>
<td>5.2.4 Power Routing</td>
<td>39</td>
</tr>
<tr>
<td>5.2.5 Dataflow Signal Cabling</td>
<td>39</td>
</tr>
<tr>
<td>6 FPGA Development</td>
<td>43</td>
</tr>
<tr>
<td>6.1 FPGA Architecture Overview</td>
<td>43</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.2 Data Capture</td>
<td>46</td>
</tr>
<tr>
<td>6.3 Correlation Processing</td>
<td>49</td>
</tr>
<tr>
<td>6.4 Averaging</td>
<td>53</td>
</tr>
<tr>
<td>6.5 Control</td>
<td>55</td>
</tr>
<tr>
<td>7 SoOp RT External Control</td>
<td>57</td>
</tr>
<tr>
<td>7.1 Data Recording Procedure</td>
<td>58</td>
</tr>
<tr>
<td>7.2 Data removal procedure</td>
<td>59</td>
</tr>
<tr>
<td>7.3 Control Scripts</td>
<td>60</td>
</tr>
<tr>
<td>7.4 ASCII Control Strings</td>
<td>60</td>
</tr>
<tr>
<td>8 Experimental Results</td>
<td>63</td>
</tr>
<tr>
<td>8.1 Purdue RNL Testing</td>
<td>63</td>
</tr>
<tr>
<td>8.2 NOAA WP-3D</td>
<td>68</td>
</tr>
<tr>
<td>8.3 Yankee Environmental Systems SoAp box</td>
<td>73</td>
</tr>
<tr>
<td>9 Continuing Work</td>
<td>77</td>
</tr>
<tr>
<td>10 Conclusion</td>
<td>79</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>96</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>58</td>
</tr>
<tr>
<td>7.2</td>
<td>59</td>
</tr>
<tr>
<td>7.3</td>
<td>60</td>
</tr>
<tr>
<td>7.4</td>
<td>61</td>
</tr>
<tr>
<td>7.5</td>
<td>62</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Cross-correlation of finite length discrete signal</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Single reflection path from Satellite to Receiver</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Multiple reflection paths from Satellite to Receiver due to a rough surface</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>Power from rough surface scattering for XM radio signals</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>GPS Scattering model at 5 m/s and 10 m/s surface wind speed [6]</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>XM Radio Frequency Allocation [16]</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>XM-3 Footprint courtesy of <a href="http://www.satbeams.com">www.satbeams.com</a></td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>QPSK signal model</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Power Spectral Density of QPSK signal</td>
<td>17</td>
</tr>
<tr>
<td>3.5</td>
<td>Autocorrelation function of a random input QPSK signal</td>
<td>18</td>
</tr>
<tr>
<td>3.6</td>
<td>Ambiguity Function of XM Radio Signal. Adapted from <em>Correlation Properties of Digital Satellite Signals and Their Applicability In Bistatic Remote Sensing</em> [15].</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>8192 Sample Spectrum of cross-correlated XM-3 signal</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Correlation of a simulated XM-3 signal which includes two delayed components</td>
<td>25</td>
</tr>
<tr>
<td>5.1</td>
<td>SoOp RT Dataflow Schematic. The two RF signals first pass through separate bias-tees, followed by the MAX2140 boards which perform the I/Q downconversion. The resulting four channels are sampled by the MAX1127 ADC which is then read by the ML402 FPGA board where the signal processing operations are performed.</td>
<td>30</td>
</tr>
<tr>
<td>5.2</td>
<td>Rackmount Chassis with RF Front Ends highlighted in purple, ADC in yellow, FPGA in green, and Logging PC in red.</td>
<td>31</td>
</tr>
<tr>
<td>5.3</td>
<td>Mini Circuits ZFBT-4R2G+ Bias-Tee</td>
<td>32</td>
</tr>
<tr>
<td>5.4</td>
<td>Maxim 2140EVKIT</td>
<td>33</td>
</tr>
<tr>
<td>5.5</td>
<td>Maxim 1127EVKIT</td>
<td>34</td>
</tr>
<tr>
<td>5.6</td>
<td>Xilinx ML402</td>
<td>35</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Raspberry Pi Model B</td>
<td>36</td>
</tr>
<tr>
<td>5.8</td>
<td>Power Supplies</td>
<td>38</td>
</tr>
<tr>
<td>5.9</td>
<td>FPGA Data Input Connector for the Rack Mount System</td>
<td>41</td>
</tr>
<tr>
<td>5.10</td>
<td>FPGA Data Input Connector for the YES System</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>SoOp RT FPGA Flow Chart. The dotted blocks represent inputs and outputs outside the FPGA. The solid blocks represent the logic partitions on the FGPA itself. The data on the FGPA flows from the ADC capture partition, to the correlation processing partition, to the magnitude and averaging partition, and finally to the IO control partition, where it is transferred over serial to the Logging PC.</td>
<td>45</td>
</tr>
<tr>
<td>6.2</td>
<td>Logic diagram of ADC asynchronous interface and input buffers. The input data from the ADC first passes through two registers. The first synchronizes the input data to the ADC clock and the second to the internal FPGA clock. The synchronized data is then sent to the two input buffers.</td>
<td>47</td>
</tr>
<tr>
<td>6.3</td>
<td>Correlation Processing Stage. The data in the correlation processing stage follows the circular correlation algorithm. An initial FFT is followed by a conjugate multiply operation which is followed by an inverse FFT.</td>
<td>50</td>
</tr>
<tr>
<td>8.1</td>
<td>SoOp RT rackmount system undergoing testing at Purdue RNL</td>
<td>64</td>
</tr>
<tr>
<td>8.2</td>
<td>XM Spectrum at Purdue RNL</td>
<td>64</td>
</tr>
<tr>
<td>8.3</td>
<td>XM Autocorrelation as observed on the SoOp RT. The first plot shows the first 100 delay bins corresponding to the first 6.25 µs, the second the complete correlation waveform.</td>
<td>66</td>
</tr>
<tr>
<td>8.4</td>
<td>XM delay line cross-correlation. Peak of Correlation is shifted 9 bins (.56 µs) compared to Autocorrelation result.</td>
<td>67</td>
</tr>
<tr>
<td>8.5</td>
<td>NOAA P-3 Aircraft</td>
<td>68</td>
</tr>
<tr>
<td>8.6</td>
<td>Downward facing RHCP Antcom Antenna</td>
<td>69</td>
</tr>
<tr>
<td>8.7</td>
<td>WP-3D equipment rack containing SoOp RT and original SoOp receivers</td>
<td>70</td>
</tr>
<tr>
<td>8.8</td>
<td>SoOp RT 8-bit correlation waveform from Fairbanks, Ak.</td>
<td>71</td>
</tr>
<tr>
<td>8.9</td>
<td>The SoAp box can be seen installed in the gray box underneath the foil wrapped ethernet adapter. Adapted from <em>HDSS WB-57 Flight Test Summary</em> [21]</td>
<td>73</td>
</tr>
<tr>
<td>8.10</td>
<td>Input Buffer sample from WB-57 flight. No signal present.</td>
<td>75</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to Digital Converter</td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
<td></td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
<td></td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
<td></td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
<td></td>
</tr>
<tr>
<td>LHCP</td>
<td>Left Hand Circularly Polarized</td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
<td></td>
</tr>
<tr>
<td>NOAA</td>
<td>National Oceanic and Atmospheric Administration</td>
<td></td>
</tr>
<tr>
<td>PARIS</td>
<td>Passive Reflectometry and Interferometry System</td>
<td></td>
</tr>
<tr>
<td>PRN</td>
<td>Pseudo Random Number</td>
<td></td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
<td></td>
</tr>
<tr>
<td>RHCP</td>
<td>Right Hand Circularly Polarized</td>
<td></td>
</tr>
<tr>
<td>RNL</td>
<td>Radio Navigation Lab</td>
<td></td>
</tr>
<tr>
<td>RT</td>
<td>Real Time</td>
<td></td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Language</td>
<td></td>
</tr>
<tr>
<td>SDARS</td>
<td>Satellite Digital Audio Radio Service</td>
<td></td>
</tr>
<tr>
<td>SoOp</td>
<td>Signals of Opportunity</td>
<td></td>
</tr>
<tr>
<td>USRP</td>
<td>Universal Software Radio Peripheral</td>
<td></td>
</tr>
<tr>
<td>YES</td>
<td>Yankee Environmental Systems</td>
<td></td>
</tr>
</tbody>
</table>
ABSTRACT


Passive bistatic radar remote sensing offers a novel method of monitoring the Earth’s surface by observing reflected signals of opportunity. The Global Positioning System (GPS) has been used as a source of signals for these observations and the scattering properties of GPS signals from rough surfaces are well understood. Recent work has extended GPS signal reflection observations and scattering models to include communications signals such as XM radio signals. However the communication signal reflectometry experiments to date have relied on collecting raw, high data-rate signals which are then post-processed after the end of the experiment. This thesis describes the development of a communication signal bistatic radar receiver which computes a real time correlation waveform, which can be used to retrieve measurements of the Earth’s surface. The real time bistatic receiver greatly reduces the quantity of data that must be stored to perform the remote sensing measurements, as well as offering immediate feedback. This expands the applications for the receiver to include space and bandwidth limited platforms such as aircraft and satellites. It also makes possible the adjustment of flight plans to the observed conditions. This real time receiver required the development of an FPGA based signal processor, along with the integration of commercial Satellite Digital Audio Radio System (SDARS) components. The resulting device was tested both in a lab environment as well on NOAA WP-3D and NASA WB-57 aircraft.
1. Introduction

Widespread observations of the Earth’s atmosphere and surface are necessary to better understand Earth’s climate and weather. Knowledge such as the wind speed inside developing hurricanes can lead to improved hurricane models and most importantly to improved hurricane forecasts. Similarly, predicting the growth and loss of sea ice and its impact on ocean circulation depends on knowledge of the sea ice conditions during past seasons. Both of these are examples of observations which can be made by a remote sensing technique called bistatic radar. This is a radar configuration which depends on a geographically separated transmitter and receiver pair. It also has the benefit of functionally separating the transmitter and receiver; the bistatic receiver does not require a cooperative transmitter. For a remote sensing radar, this means that existing satellite transmissions can be co-opted as ‘signals of opportunity’. The bistatic radar system then avoids the cost and complexity of a dedicated transmitter [1]. Practically, this both simplifies receiver development and allows installation of the receiver in a wider variety of platforms.

Most work performed to date on passive bistatic remote sensing radar systems has relied on Global Navigation Satellite System (GNSS) signals as the illumination source. This is mostly due to the convenient signal structure of GNSS signals. For navigation purposes GNSS signals include a Pseudorandom Noise (PRN) sequence, this is an approximately random sequence which periodically repeats at known times. This PRN is intended to enable the calculation of the time of flight between the transmission at the GNSS satellite and reception at the navigation receiver. When the position of the satellite constellation is known along with the time of flight calculations, the receiver can generate a position solution. In a remote sensing application, there is a similar requirement on the signal structure. Knowledge of the delay from a reflection point on a target surface to the receiver provides a means of remotely
sensing that target. However there are several downsides to GNSS signals. The first is that the GNSS signals are transmitted with relatively low power; the signal content of the transmission is known due to the PRN sequence so high power transmissions are not required for navigation receivers. The second is due to the orbit of the GNSS satellites, which are in a Mid Earth Orbit (MEO) with an approximately 12 hour period. This causes the geometry of the transmitter to receiver path to change over time, even when the receiver is stationary. Because of these limitations, new efforts are being made to develop passive bistatic radar systems which use digital communication signals as the illumination source.

Unlike GNSS signals, the signal content of a communications signal is unknown and cannot be generated independently at the receiver. This prevents a direct calculation of the time of flight from the transmitter to the receiver. However by comparing a directly received version of the transmitted signal with the reflected version of that signal, the relative delay between the two signals can be found and the requirements of the remote sensing bistatic radar system can be met. However this does impose some requirements on the signal choice, the signal cannot contain periodic components shorter in period than the maximum expected delay, since they would cause ambiguity in the time delay calculation. The signal must also provide a signal structure which supports an adequate delay resolution calculation.

XM satellite radio signals are an example of a non-periodic signal, due to their compressed digital encoding. These signals are continuously broadcast by two satellites over the North American continent for radio listening. Due to this fit with the passive bistatic radar requirements, they have been investigated as an illumination source in a remote sensing system. Following this they were used in a successful experiment to perform ocean wind surface observations [2]. This experiment recorded the raw direct and reflected signals from the XM satellites through a software defined radio (SDR). This data was later post-processed and compared to scattering models to recover the wind speed values.
This thesis covers the development of a bistatic remote sensing radar receiver, the Signals of Opportunity (SoOp) Real Time (RT) receiver. The SoOp RT receives and processes XM radio signals reflected from the Earth’s surface and returns processed results in real time. By providing these results immediately the receiver supports deployment on platforms which require short-term feedback, such as flights into developing hurricanes, or on platforms which have limited data down-link capacity, such as a satellite. Due to the high computational load of the signal processing algorithm, the receiver required the development of an FPGA based software defined radar.

1.1 Outline

This thesis is split into nine chapters which describe the development of the SoOp system as well as the principals behind its operation. Chapter 1 is this introduction and outline. Chapter 2 describes the background and history of bistatic remote sensing, starting with GNSS remote sensing and continuing with remote sensing using other signals of opportunity. Chapter 3 discusses the application of XM Radio as a signal of opportunity. Chapter 4 details the algorithm implemented in the SoOp system, and shows a software model that was developed to simulated it. Chapter 5 covers the hardware content and development of the SoOp radar receiver. Chapter 6 provides a detailed description about the FPGA development required for the SoOp receiver. Chapter 7 explains the functionality and operation of that receiver. Chapter 8 includes the experimental results of the system, both in the lab and in the field. Chapter 9 describes continuing work and proposes future work. Chapter 10 then summarizes this thesis and provides the conclusion. Finally, the Appendix includes simulation code, a description of the FPGA toolset, and documentation for the receiver hardware.

The classic radar system is monostatic, the radar transmitter and receiver are physically collocated and often share the same antenna. This allows the monostatic receiver to observe backscatter reflections from the target of interest. A bistatic or multistatic radar system separates the transmitter and receiver components, often by significant distances. This extends the observable reflections to include scatter in the forward direction as well, though it adds the requirement of independent transmitter and receiver hardware. While a monostatic radar implies a cooperative transmitter, a bistatic radar system can include either a cooperative or uncooperative transmitter. This thesis is concerned with the latter, which is referred to as passive bistatic radar. The uncooperative signals are referred to as either "illuminators of opportunity" or more generally "signals of opportunity." Since this restricts the transmitter design in the bistatic system to the selection of an existing signal, the development effort is then focused on the receiver side of the radar system. The following chapter will present an overview of the general bistatic radar remote sensing technique, which historically has relied on GNSS signals.

2.1 History of Bistatic Radar Remote Sensing

Early deployments of bistatic radar systems were carried out on the Lunar Orbiter and Explorer 35 lunar probe missions. These early spacecraft carried radar transmitters directed towards the lunar surface, with the intention of reflecting a signal back towards receivers on Earth. This effort was successful and signals from the lunar surface were recorded. To interpret the resulting data, Tyler et al. developed a relationship between the surface roughness, scattering area, and the geometry of the transmitter-moon-receiver system [3]. They found that for signal reflections with
a moderate angle of incidence, an increase in surface roughness of the lunar surface resulted in an increase in the observed radar cross-section at Earth. This model then led to confirmation of the discovery that the lunar highlands are less reflective to RF transmissions than the lunar seas.

2.2 GNSS Bistatic Radar Remote Sensing

2.2.1 History

While the lunar bistatic radar experiment included a dedicated transmitter, Earth ocean altimetry was the first proposed use for passive GNSS remote sensing. In 1993 the "passive reflectometry and interferometry system" (PARIS) was evaluated as an alternative to a constellation of dedicated nadir observing radar altimetry satellites [4]. At the time a new altimetry method was necessary due to the desire to observe mesoscale current flows in the ocean. This required an altimeter that could measure over at least a 250km swath of ocean and existing nadir-looking radar altimeters offered only a limited field of view. When paired with a modified navigation receiver, the large number of GPS transmitters combined with the geometry of the GPS satellites compared favorably to a dedicated five satellite radar altimetry constellation. By mounting an antenna on a bridge near Rotterdam in the Netherlands, it was experimentally shown that processing the C/A embedded code in a reflected GPS signal resulted in an altimetry measurement with a 3.3 m RMS error. [4].

Following the altimetry experiments, ocean reflected GNSS signals were recorded by a specialized GPS receiver over a variety of sea surface states [5]. These observations showed a dependency of the structure of the reflected signal on the roughness of the ocean surface. This application of a bi-static radar system relied on cross-correlating the reflected signal from the ocean surface with an internally generated PRN sequence, which yielded the power reflected at given time delays. The result from this cross-correlation was found to match the theory of bi-static scattering of a range coded signal off of a rough surface. Matching the power in the correlation
calculation result to the scattering model then allowed for the retrieval of wind speed at the ocean surface, due to the relationship between the ocean surface roughness and the surface winds [6].

However these initial experiments observed the ocean surface at relatively low wind speeds. By mounting a GPS receiver on a NOAA WP-3D "Hurricane Hunter" aircraft, Katzb erg et al. were able to collect data which covered hurricane speed winds. This data showed that higher wind speeds were underreported due to the non-linearity of the surface mean squared slope compared to the wind speed, which is the physical relationship necessary for GPS wind speed retrieval. This data was then used to calibrate the model to account for the non-linear behavior and to allow wind retrieval up to 35 m/s surface wind speed [7].

The technique of passive GNSS bistatic remote sensing was extended to other surfaces of interest as well. In 2000, a bistatic remote sensing receiver was flown over a variety of land features in both wet and dry conditions. The resulting data was found to vary with the moisture content in the soil at those features [8]. An analysis of this data found a relationship between the mean reflected power in the returned signal and the soil condition. In a different study reflections from GPS signals were also observed over sea ice in the Beaufort Sea [9]. In this experiment the peak power of the reflected signal was found to correlate well with both the presence of sea ice, as well as its age and condition.

These initial experiments each observed GNSS signal reflections at relatively low altitudes. The first altimetry experiments to support the PARIS analysis did so by mounting an antenna to a bridge. Later reflectometry recordings for ocean surface and land surface observations were made from aircraft mounted receivers. The flexibility of GNSS remote sensing was expanded when a much higher altitude space based reflection was found in data recorded during the Spaceborne Imaging Radar-C (SIR-C) experiment aboard the Space Shuttle [10]. Capturing a GPS signal during this experiment was unintended, however analysis of the recorded data showed an
adequate signal to noise ratio (SNR) of the reflected GPS signal which would support remote sensing applications.

2.2.2 Continuing Work

The availability of space based reflected signals from the Earth’s surface led to a proposal for an 8 satellite constellation of GNSS bistatic receivers in Earth orbit, the Cyclone Global Navigation Satellite System (CYGNSS) [11]. The CYGNSS constellation of satellites is intended to provide high temporal resolution ocean surface observations from the inner core of tropical cyclones. These observations will then be applied to improving hurricane intensity forecasts. By basing the system on a bistatic radar listening to signals of opportunity, the cost of each satellite in the constellation is reduced compared to one with an active transmitter. CYGNSS is expected to launch in 2016 [12].

2.3 Signals of Opportunity Bistatic Radar

With the success of GNSS signals as a bistatic illuminator, alternate signal sources have been investigated. While the PRN sequence contained in GNSS signals is convenient for remote sensing applications, the signals are also relatively low power and have a time-varying geometry with respect to the Earth’s surface. Communications signals in contrast are high power and are available from a multitude of satellite and Earth based transmitters. Initially, communications signals were looked at for air target detection rather than remote sensing use [13]. Cherniakov et al. investigated an Iridium like satellite constellation as illumination source for a bistatic radar system to passively observe aircraft. It was confirmed experimentally that the power budget for a reflected signal from that constellation off of a helicopter would high be enough for practical detection use.

Due to the large number of transmission sources, space based communication signals exist for many frequencies and with many different methods of modulation.
Early investigations of the impact of these variations on a bistatic radar system were conducted by Griffiths et al. [14]. Signals with repetitive components such as analog voice and tv broadcasts were found to have range and doppler ambiguities which could impact the bistatic radar system. Digitally encoded signals avoided these periodic ambiguities and became the preferred source for signals of opportunity for bistatic radar [1].

One source of a digitally modulated communications signals is a Satellite Digital Audio Radio Service (SDARS). XM radio is one of these and is broadcast at a frequency with scattering properties useful for ocean surface observation [15]. Assuming that the data encoded in the compressed digital signal is random allowed the analytical derivation of the ambiguity function of the XM signal. The GNSS scattering model was then applied to the signal, but updated with the XM signal properties. The positive results of that effort led to experimental observations of reflected XM radio signals and the recovery of ocean surface roughness conditions [2].

These experimental observations were taken with software radio USRP2 receivers from Ettus research with data logged to a hard-drive through a PC104 based linux system. Sampling the complex RF signals at 8MHz resulted in a 32MBps data rate from the receiver which necessitated that all processing was performed after the flight in Matlab. This receiver continued to fly on the NOAA WP-3D throughout the 2013 hurricane season.

2.4 Theory of Operation

Sensing the change in a signal caused by a reflection from the Earth’s surface requires a mathematical tool to compare the similarity of the original signal with the reflected copy. A discrete cross-correlation function accomplishes this by a summation of the product of the two signals at a time shift \( n \) as is shown in Equations 2.4, where \( N \) is the length of the correlation. The cross-correlation result is maximized when the signals are most similar to each other, the value of \( n \) at that point then imparts
information about the time delay present in the reflected signal. The shape of the
peak in the cross-correlation result as well as any periodic ambiguities present in it
depend on the signal structure of the original signal. These properties can be observed
in the cross-correlation of the original signal with itself, called an auto-correlation.

\[ R_{xy}[m] = \sum_{n=0}^{N} x[n] y[m+n] \]  \hspace{1cm} (2.1)

Figure 2.1.: Cross-correlation of finite length discrete signal

A signal reflecting off of a flat surface is dominated by a specular reflection, so it
can be thought of as traversing a single path from the transmitter to the reflection
point and ending at the receiver. That received signal would be delayed by the travel
time due to the path length, but would otherwise be nearly identical to the originally
transmitted signal. Cross-correlating the original and received signals with each other
would then return the auto-correlation of the original signal, delayed by the time delay
due to the path length.

However, if the surface is rough, the reflected signal will include many components
with varying time delays compared to the specular reflection point. Since the received
signal includes all of those components, the cross-correlation between the original
and received signals will show a distribution at delays around the specular delay [6].
Squaring that cross-correlation result provides a waveform of the correlation power
at each time step, the shape of that waveform can then provide information about
the state of the reflecting surface.

Since this method relies on the cross-correlation of the direct and reflected signals,
the auto-correlation properties of the transmission signal heavily influence the quality
of the entire bistatic radar system. These properties will be shown for XM radio
signals in Chapter 3.
2.5 Geometry of a Bistatic Remote Sensing System

A satellite based transmitter can be located in either Low Earth Orbit (LEO) such as Iridium, MEO such as GPS, or Geostationary orbit (GEO), such as XM radio. For XM radio there are two broadcast satellites in orbit: Rhythm at 85 West Longitude and Blues at 115 West Longitude. Since the satellites are geostationary, those longitudes remain constant and the motion of the overall bistatic radar system is dependent entirely on the receiver. This also limits the doppler shift in the received signal to only the doppler caused by the receiver velocity. In practice, this means that a geostationary transmitter and an aircraft mounted receiver would experience significantly less doppler shift than a MEO GPS transmitter and a stationary receiver. This is significant since it simplifies the cross-correlation calculation, a GPS bistatic receiver requires additional tests at a range of doppler frequencies since the reflected signal will have shifted in frequency compared to the original PRN sequence.

A specular reflection is a reflection where the incident and reflected angles to the reflecting surface are equal. The specular path from the satellite to the receiver in the SoOp system is shown in Figure 2.2. However the total received signal is the summation of all signals reflected from the ocean surface inside the ground pattern of the receiving antenna as shown in Figure 2.3. Since the transmitter is at a much greater distance from the reflecting surface than the receiver, the incident angle of each ray path is assumed to be the same and the paths are assumed to be parallel. In comparison, the reflected angles will vary for each signal component and so are not parallel to each other. This causes the differing path lengths and differing time delays in the received signal. The correlation result is then the sum of all the delayed signals.
Figure 2.2.: Single reflection path from Satellite to Receiver

Figure 2.3.: Multiple reflection paths from Satellite to Receiver due to a rough surface
2.6 Scattering Model

A scattering model for XM radio signals has been developed by Shah et al. based on the model for GNSS reflectometry [2]. The reflected power at the receiver for a given time, frequency, and position is modeled as:

\[
\langle |Y(\tau, f; P_v)|^2 \rangle = \frac{T_i^2}{4\pi} \int \int \frac{\pi|\Omega|^2 D^2(\hat{\rho}) q^4(\hat{\rho})}{R_{tr}^2(\hat{\rho}) R_{rec}^2(\hat{\rho}) q_x^4(\hat{\rho})} P_v(\frac{-q_h}{q_z}) \sigma^2 d^2 \rho
\]  

(2.2)

Figure 2.4.: Power from rough surface scattering for XM radio signals

Where \( T_i \) is the integration time, \( \Omega \) is the Fresnel reflection coefficient, \( D \) is the antenna gain pattern, \( R_{tr} \) is the distance to the transmitter, \( R_{rec} \) is the distance to the receiver, \( q_h \) and \( q_v \) are the components of the scattering vector, \( P_v \) is the slope PDF of the surface, \( \sigma \) is the self ambiguity function of the signal, and \( \rho \) is a 2D coordinate on the ocean surface. While the slope PDF of the surface is modeled as a Gaussian distribution representing the slope of surface facets on a rough surface, it has been empirically shown to have a dependence on ocean surface wind speed [2]. Similar to the GPS model, the result of simulating this relationship produces a waveform dependent on surface winds as well, particularly slope of the trailing edge of the waveform. This can be seen in Figure 2.5 which plots the scattering model for GPS signals. The higher wind speeds extend the power distribution over greater delay bins, resulting in a shallower slope of the trailing edge of the correlation waveform.

Since this model provides a relationship between correlation power and a particular wind speed, when the experimental correlation waveform is fit to the scattering model the surface wind speed can then be retrieved from the observed data. In practice, this estimation has been performed by applying a non-linear least squares fit of the collected data to the scattering model [6]. However this estimation calculation is outside the scope of the SoOp RT system, which generates the correlation waveform only. This is due to both the relatively low processing requirements of the estimation.
calculation as well as the desire to maintain the maximum flexibility of the SoOp receiver. Additional experimental observations will likely cause further scattering model development and a change to the model is more easily accommodated on a general purpose PC than an FPGA.

Figure 2.5.: GPS Scattering model at 5 m/s and 10 m/s surface wind speed [6].
3. XM Radio

The XM Radio service broadcasts digital communication signals across North America. For a subscription fee, the service provides access to music, news, and sports radio programs on 156 digital channels. These signals are transmitted in a 12.5MHz band between 2332.5MHz and 2345.0MHz. Within this band, there are six frequency allocations. Two are dedicated to terrestrial repeaters, and two each are allocated to each satellite as is shown in Figure 3.1. The A1 and B1 channels are broadcast by the Rhythm satellite while A2 and B2 are broadcast by Blues. These channels are combined at the receiver to provide more robust reception than a single channel would provide. All satellite transmissions are left hand circularly polarized while the terrestrial repeaters transmit a vertically polarized signal [16].

Figure 3.1.: XM Radio Frequency Allocation [16]

Most importantly for a bistatic remote sensing application, XM radio transmits a digitally compressed signal. Each channel occupies 1.886MHz of bandwidth with a symbol rate of 1.64 Mega-symbols per second (Msps). Quadrature Phase Shift Keying (QPSK) modulation is used for an effective bit rate of 3.28 Mbps, which after error correction is included provides a broadcast bit rate of more than 2 Mbps. The signals are also broadcast at high power, each satellite transmits 6.4 kW of RF power and when launched were the most powerful commercial communications satellites ever built [16].
3.1 Coverage Area

The two XM satellites both cover the majority of the North American continent, however they have different antenna footprints due to differing positions in equatorial orbit. The XM-3 satellite, Rhythm, has better signal coverage at higher latitudes compared to XM-4 as can be seen in Figure 3.2. While the SoOp RT system can select either satellite as a transmission source, all experiments were conducted with XM-3 due to the wider footprint.

![XM-3 Footprint](www.satbeams.com)

Figure 3.2.: XM-3 Footprint courtesy of [www.satbeams.com](http://www.satbeams.com)

3.2 Signal Structure

The QPSK baseband signals as received at the bistatic radar receiver can be modeled as a sum of sinusoids shifted in phase depending on a random input, as is shown in the noise free model in Equation 3.1.
The data encoded in the signal is represented by \( a_k \), which can take values of 0, 1, 2, and 3 which represent the bit strings 00, 01, 10, and 11. This results in a symbol rate at half the bit rate of the non-encoded datastream, which is 3.28 Mbps for XM signals. The \( p() \) function is a unit pulse and \( T_{symb} \) is the period of a symbol (.6098 \( \mu s \) for XM radio) [15]. Assuming that the data input is random, the power spectral density (PSD) of the QPSK baseband signals is then shown in Equation 3.2 translated by \( \pm f_c \), the carrier frequency [17].

\[
x[n] = \sum_{k=-\infty}^{\infty} e^{i \left( \frac{\pi}{4} j + \frac{\pi}{2} a_k j \right)} p \left( \frac{n - kT_{symb}}{T_{symb}} \right)
\]  (3.1)

Figure 3.3.: QPSK signal model

\[
S(f) = 2T_{symb} \left| \frac{\sin(\pi f T_{symb})}{\pi f T_{symb}} \right|^2
\]  (3.2)

Figure 3.4.: Power Spectral Density of QPSK signal

3.2.1 Auto Correlation properties of the XM Radio signal

The shape of the power spectral density function of the QPSK signal in Equation 3.2 is an important property of the signal, since it impacts the possible resolution of the bistatic radar system. It is related to the the self-ambiguity function of the signal, which is a standard description of the delay and doppler resolution of a radar pulse. The self-ambiguity function is calculated by mapping the matched filter response of a signal when delay and Doppler are varied [15]. This function for the XM signal is shown in Figure 3.6.

Since the SoOp bistatic receiver assumes small changes in doppler and only delays are considered in the scattering model fit, the range resolution is the dimension of
interest and the ambiguity function can be simplified by taking a 1-d cross section about the 0 doppler axis. This is same as the auto-correlation function of the signal and can be found by taking the time-domain transform of the frequency-domain PSD shown in Equation 3.2, which is the triangle function in Equation 3.3 where $T_{sam}$ is the sample period, $T_{sym}$ is the symbol period, and $n$ is the index.

$$R_{xx}[n] = \begin{cases} 
1 - |n\frac{T_{sam}}{T_{sym}}| : & -T_{sym} < nT_{sam} < T_{sym} \\
0 : & \text{otherwise}
\end{cases}$$

(3.3)

Figure 3.5.: Autocorrelation function of a random input QPSK signal

The range resolution of the passive bistatic radar can be measured by the time resolution constant of the ambiguity function at zero Doppler, $t_{res}$ [18].

$$t_{res} = \frac{Max(R_{xx})}{\frac{1}{N}\sum_{m=0}^{N} R_{xx}[n]}$$

(3.4)

Where $N$ is half the coherent window length. The value of this constant shows the dependence of the resolution on the autocorrelation shape. A tall and narrow autocorrelation function such as the one in Equation 3.3 is desirable for this reason and is another benefit of selecting the XM signal as the radar illumination source.
Figure 3.6.: Ambiguity Function of XM Radio Signal. Adapted from *Correlation Properties of Digital Satellite Signals and Their Applicability In Bistatic Remote Sensing* [15].
4. SoOp RT Algorithm

The first step to measuring ocean surface roughness with the SoOp RT system depends on the calculation of the cross-correlation function. The inputs to the cross-correlation are a directly received copy of the transmission signal and one reflected off the target, the ocean surface. After that calculation, the correlation waveform can be fit to a scattering model in order to estimate the surface condition. Since the cross-correlation is the most computationally intensive step of the process, this is the step accelerated by the SoOp RT system. It does so by generating a real time waveform of the cross-correlation between direct and reflected copies of an XM radio broadcast. To keep the device as general purpose as possible, fitting a model to these waveforms is left for post-processing. The model fit is within the capabilities of a general purpose computer, so this method supports real time results while increasing the flexibility of the device.

The cross-correlation algorithm implemented on the SoOp RT is based on a circular correlation method commonly used in GNSS signal processing. The circular cross correlation is performed in three stages. The two direct and reflected input signals, \( x[n] \) and \( y[n] \), are transformed into the frequency domain by a Fourier transform as is shown in Equation 4.1.

\[
X[k] = \sum_{n=0}^{N} x[n] e^{-j2\pi kn/N} \\
Y[k] = \sum_{n=0}^{N} y[n] e^{-j2\pi kn/N}
\] (4.1)

Where \( n \) is the index of the captured data and \( N \) is the window length, in this case 8192. \( k \) is an integer between \( -N/2 \) and \( N/2 \). The direct signal is then multiplied by the complex conjugate of the reflected signal as is shown in Equation 4.2.
Finally the result is transformed back into the time domain by an inverse Fourier Transform shown in Equation 4.3.

\[ R_{xy}[n] = \frac{1}{N} \sum_{k=0}^{K} C[k] e^{\frac{i2\pi kn}{N}} \quad (4.3) \]

At this stage the correlation result is in a complex format, consisting of both phase and magnitude. However only the power is required for the scattering model, so the magnitude at each delay bin is taken which is the square-root of the power.

\[ RM[n] = \sqrt{\text{Re}(R_{xy}[n])^2 + \text{Im}(R_{xy}[n])^2} \quad (4.4) \]

Using real-world signals also requires two additional steps. Noisy input data requires a filtering stage either before the initial FFT or immediately after. To simplify the computation in the SoOp RT receiver, this is performed after the first FFT by multiplying the frequency domain signals against a stored filter. Then to reduce noise in the output waveform, the result is time averaged.

\[ RA[n] = \sum_{l=0}^{L} RM_l[n] \quad (4.5) \]

Where \( l \) is the index of the correlation and \( L \) is the number of correlations taken before the average is calculated, which for the SoOp RT is 4096.

This method assumes a periodic signal, with a period matching the length of the data window being cross-correlated. For GNSS signals with a repeating PRN sequence, this is a valid assumption. When evaluating the correlation result at a
delay, the circular correlation method wraps the input data around circularly. For example if one input \( y[n] \) is equal to \( x[n] \) delayed by \( N/2 \) as in the example shown in Equation 4.6, the circular correlation would be represented by Equation 4.7.

\[
y[n] = x[n - N/2]
\] (4.6)

\[
R_{xy}[N/2] = \sum_{n=0}^{N} x[n]y[n + N/2] = \sum_{n=0}^{N/2} x[n]x[N + n] + \sum_{n=N/2}^{N} x[n]^2
\] (4.7)

For a periodic signal with length \( N \) the value \( x[N+n] \) is equivalent to \( x[n] \), however this is not the case for a digitally compressed communication signal which is assumed to be entirely random and non-periodic. Instead \( x[N+n] \) will not correlate with \( x[n] \) and so the magnitude of the correlation at \( N/2 \) would decrease by 1/2 compared to an infinite length correlation. The correlation result can be scaled to account for this effect, which decreases linearly as the delay increases, though the noise at greater delays will be scaled as well.

### 4.1 Software model

A software model of the algorithm described in the prior section was developed to both simulate the overall bistatic system as well as to validate results produced by completed hardware, the full code for this simulation is available in the appendix. The QPSK signal from the XM-3 satellite was modeled with a random input data over .5\( \mu s \) at .0625 \( \mu s \) discrete time step to match a single sample window of the SoOp RT receiver. A second baseband signal is also created by combining two delayed copies of the the original QPSK signal at 5\( \mu s \) and 8\( \mu s \) to represent the reflected signal. While this is not representative of a true signal reflection off a rough surface, the purpose of this simulation is to support the in lab verification which will be discussed in Section
8. In the lab the "reflected" signal input is tested with a time delayed direct signal, similar to the simulation.

Both direct and reflected signals are up-converted by multiplication with a 2.44GHz carrier signal. This signal is aliased at a discrete time step matching the 16MHz sample rate frequency in order to reduce computation time. Next the signal is multiplied by both a sine and cosine at 2.34GHz, again aliased to 16MHz, to match the I/Q down-conversion performed by front end of the SoOp RT receiver. This creates the desired 6MHz offset of the original QPSK signal. Noise is added to both of these signals which results in a 3dB Signal to Noise Ratio (SNR). The spectrum of the resulting signal is shown in Figure 4.1. This spectrum provides a visualization of the 6MHz offset introduced by the down-conversion.

![Figure 4.1.: 8192 Sample Spectrum of cross-correlated XM-3 signal](image)

An FFT was then applied to both signals, followed by a band pass filter to remove satellites outside the 1.64MHz wide channel centered at 6.155MHz representing the XM-3 satellite. This is followed by a conjugate multiply operation between the direct and reflected signals, then an inverse FFT to complete the circular correlation. The
magnitude of that result is shown in Figure 4.2. As expected it shows two triangle functions matching the autocorrelation function for the QPSK signal delayed in time and centered at 5 µs and 8 µs.

Figure 4.2.: Correlation of a simulated XM-3 signal which includes two delayed components
5. SoOp RT Hardware

There are two versions of the Signals of Opportunity Real Time hardware, a rack-mount system and a system installed in the Yankee Environmental Systems (YES) dropsonde chassis. The rack-mount chassis can be installed in any standard 19 in. equipment rack, such as the racks on the NOAA WP-3D aircraft. The YES system is mounted in a custom chassis designed for installation in a NASA WB-57 aircraft. They both contain the same core components, however each have different power supplies and logging PCs. The internal cabling on each is also slightly different as well.

5.1 Signal Processing Overview

As discussed in the algorithm section, the SoOp RT system compares an XM radio signal received directly from the satellite with one reflected from the ocean surface. This requires the SoOp RT box to processes two S-band RF inputs. The direct signal is received from an upward facing LHCP antenna, while the reflected is received from an otherwise identical downward facing RHCP antenna. These antennas are tuned for the center of the XM band at 2338.75 MHz with a 20MHz bandwidth [19].

The RF signals from the antennas are input to the SDARS boards which downmix the microwave signal to baseband, which is centered at 2337.89 MHz. The SDARS boards also generate a separate inphase and quadrature version of the baseband signal for each input, which results in 4 analog channels that are sent to the ADC. There is a 6MHz wide low pass filter included on the output of the SDARS board, which allows both XM satellites as well as the XM terrestrial band to pass through.

As can be seen in Figure 3.1, since the SDARS board is centered at 2337.89 MHz, the XM satellite signals are in the upper portion of the 6 MHz spectrum output from
the SDARS board. Only one satellite is sampled at a time, however this filtering is only performed later by applying a digital bandpass filter in the FPGA. This method allows the satellite choice to be altered without modifying the hardware, however it requires the full 6MHz to be sampled by the ADC at a minimum of 12MSps.

To meet that 12MSps minimum sample rate, the ADC over-samples the SDARS output at 16MSps, since 16MSps is the lowest sample rate directly supported by the ADC development board. The 12MSps rate would be possible if the board were instead set to sample at 24MSps and dropped every other sample during FPGA capture. This method was tested at 16MSps for an effective sampling rate of 8MSps, however at that time in development the ADC hardware for the WB-57 system had already been shipped and set to a 16MSps rate, so it was not possible to update the rack-mount system to 12MSps without breaking equivalence between the two sets of hardware.

After converting the four inputs to the digital domain, the ADC outputs a 12-bit binary offset format number for each channel along with a capture clock over four 13 pin parallel connectors. Only the upper 8-bits are captured by the FPGA, which effectively divides the original number by 16 and takes the floor of the result. This is adequate, since only 8-bit resolution is necessary for the correlation. The final result sent to the FPGA is an 8-bit wide binary number centered at a value of 127.

5.2 Hardware

The core SoOp RT system is comprised of four development boards and support circuits. From front to back starting at the antenna input, the signals first pass through bias-tees which provide a 5V DC bias to power the antenna amplifiers. Those then connect to the RF front ends, two Maxim 2140 SDARS boards, which remove the 2.3GHz carrier signal and generate baseband Inphase and Quadrature signals. Those feed into the Maxim 1127 ADC board which converts the four analog signals to four 12 bit binary signals, which are deserialized and sent to the FPGA for processing. The
FPGA is on the Xilinx ML402 board, which reads the parallel digital signals from the ADC, processes the signal, and transmits the results over a serial connection to the logging PC. The logging PC then both records the data to an SD card and controls the run-mode of the FPGA board. This dataflow signal path can be seen going from left to right in the schematic in Figure 5.1.
Figure 5.1.: SoOp RT Dataflow Schematic. The two RF signals first pass through separate bias-tees, followed by the MAX2140 boards which perform the I/Q down-conversion. The resulting four channels are sampled by the MAX1127 ADC which is then read by the ML402 FPGA board where the signal processing operations are performed.
5.2.1 Chassis

The rack-mount chassis in Figure 5.2 is a 19 in. wide 1U server rack manufactured by Circuit Specialists. The IO ports include a MS3470W12-3P power connector, two bulkhead SMA connectors, a female-female bulkhead Ethernet port, and two female-female bulkhead USB ports. The USB ports have been found to be unreliable when first connected and often require upward pressure to complete the connection. These ports will require eventual replacement, but were functional for initial testing.

The WB-57 chassis was designed and manufactured by YES in order to fit into spare space in their dropsonde launcher. The power, network, and USB connections are internal to the YES system, only the two rf ports required for the antenna connections are unique IO for the SoOp RT system.
5.2.2 **Hardware Details**

**Bias-Tees**

The Bias-Tees are manufactured by Mini-Circuits, with the part number ZFBT-4R2G+. There are two identical bias-tees in both the rack-mount and YES systems, one each for the direct and reflected input signals. The antennas require a DC bias to power the internal amplifier which is provided by applying 5V DC to the bias-tee through an SMA connector.

![Mini Circuits ZFBT-4R2G+ Bias-Tee](image)

**Figure 5.3.:** Mini Circuits ZFBT-4R2G+ Bias-Tee

**SDARS**

The Maxim 2140EVKIT SDARS boards function as the RF Front end of the SoOp RT system and down-mix the raw direct and reflected XM signals from the antennas to a 6 MHz wide baseband signal. This is wide enough to cover two satellites and a terrestrial band, or one satellite and two terrestrial bands. The boards are manufactured by Maxim as a development platform for the Maxim 2140 SDARS chip and include a complete two-stage receiver with automatic gain control. The center frequency is adjustable over either an I2C bus or parallel port connection, however the SoOp RT box leaves them centered at the default frequency of 2337.89 MHz. The clocks are synchronized between both boards, which is documented in the appendix.
The ADC is a Maxim 1127EVKIT development board. It includes four input channels which can be sampled between 16 and 65Msps, based on a rate set by an external reference clock. The high sampling rate capability was desired to allow for alternate RF front ends to be swapped into the system and to support signals with higher bandwidth than XM.

In the SoOp RT system which receives XM signals, the sampling clock is generated by the FPGA and is set to 16MHz. This meets the minimum 12Msps requirement to fully sample the 6MHz bandwidth from the SDARS boards.

Once sampled, the analog inputs are converted to digital 12-bit unsigned integers, which are output over both 12-bit Low-voltage differential signaling (LVDS) and parallel outputs. For the SoOp RT systems, only the parallel outputs are used. While the LVDS output would require fewer physical connections to the FPGA, at a 16 MHz sampling rate the LVDS channel switches at the relatively high frequency of 96 MHz. In order to reduce noise inside the system and to simplify development, the lower frequency 16 MHz parallel outputs are sent to the FPGA.

During system build, several errors were found on the 1127EVKIT datasheet. Jumpers JU1 through JU5, which enable the input channels, must be set to pins 2-3 instead of pins 1-2 as defined in the datasheet. Additionally, the 1127EVKIT board
only has provisions for hook-connectors to provide power to the ADC. Since these would be unstable in a high vibration environment, power leads were soldered to the decoupling capacitors on each power rail.

Figure 5.5.: Maxim 1127EVKIT
FPGA

The FPGA is a Xilinx Virtex-4 SX35 mounted on a ML402 development board. The SX35 includes 34,560 logic cells, 192 DSP slices, and 3456Kb block ram. The Virtex 4 has since been superseded by the Virtex 5, 6, and 7 families of parts, however the Virtex 4 was chosen due to the availability of the radiation hardened XQR4VSX55, which is compatible with designs developed for the SX35 and would allow for a future deployment in a space environment. The ML402 development board is no longer available from Maxim, however used boards are generally available through internet auctions. Care should be taken when handling this board in particular, since due to the large number of peripherals the ML402 has been found to be fragile.

Those peripherals on the board include a wide variety of IO including VGA, USB, Ethernet, RS-232, TTL GPIO, as well as an LCD display. However only the serial port, GPIO, and clock outputs are used by the SoOp RT system. The USB port is also used in the rack-mount system, however it is only used to provide 5V power to the logging PC and is not used for communications.

Figure 5.6.: Xilinx ML402
Logging PC

The logging PC in the rack-mount system is a Raspberry Pi Model B. The Raspberry Pi is an ARM based PC which runs a Debian based Linux distribution and includes Ethernet, USB, and serial IO. All local storage as well as the operating system is contained on an SD Card. All serial communication with the FPGA is handled by Python 2.7 scripts, which are compatible with the YES logging PC with a few minor edits.

Since only one serial port is included on the Raspberry Pi, a USB to TTL Serial adapter is used to connect to the FPGA, while the on-board serial port is available externally through another USB to TTL Serial adapter. The external port provides a terminal connection to the Raspberry Pi during the boot process. Communication after boot is generally performed through the Ethernet port, however until an IP address is assigned only the serial terminal is available.

The YES system uses the logging PC provided by dropsonde launcher. Since it is also based on Linux and Python 2.7 it is functionally similar to the Raspberry Pi. However since in the YES system the PC is shared with the dropsonde system, a limited number of CPU cycles are available for the SoOp RT control and processing.
**FPGA Programmer**

An FPGA programmer is included in both the rack-mount and YES systems to facilitate re-flashing the FPGA bitstream. On the rack-mount system the USB port to the programmer is available externally through a bulkhead connector. The programmer for both systems is the Digilent XUP USB-JTAG programming cable, which is compatible with software that supports the Xilinx Platform Cable USB.

**Active Serial Cables**

The rack-mount system includes two USB to serial adapters, both are FTDI TTL-232R-3V3-WE cables. These cables include an active serial adapter that converts 3.3V TTL serial to USB. One is used by the Raspberry Pi to communicate with the FPGA, the other is an external output from the Raspberry Pi to the chassis. This bypasses the need for an RS-232 adapter, which is bulkier than the TTL adapter. The YES system uses a direct RS-232 serial connection to the logging PC and does not include any serial to USB adapters.

**Power Supplies**

There are three power supplies in the rack-mount system, a V-Infinity VYB20W-Q24-D5-T, a Vicor V28C5M50B, and an LM317 based breakout board. The V-Infinity board supplies +5V and -5V DC at 2A when provided with a 9-36V input, however it is a linear-mode supply so it is only used for SDARS and ADC boards. The Vicor is a switched-mode 5V DC-DC supply that also requires a 9-36V input and powers the FPGA and logging PC. The LM317 board is connected to the 5V output of the V-Infinity supply and provides 1.5V, 1.8V, and 3.3V for the ADC board.

Since the external power input to the system only connects to the V-Infinity and Vicor boards as well as the status LED, for bench testing the power input can be anywhere from 9-36V. However the LED assumes a 28V input and may not light up
at lower voltages. When installed in an aircraft, 28V DC is provided and the LED functions as a power on indicator.

The YES system power is supplied by the power supply in the YES chassis, no additional components were provided except for an example of the LM317 breakout board which was re-designed by YES.

![Power Supplies](image)

Figure 5.8.: Power Supplies

### 5.2.3 Hardware Modifications

Since the SDARS and ADC boards are development boards, several modifications are necessary for them to work in the SoOp RT system. All modifications are documented step by step in Appendix 11.3.

The SDARS boards require two modifications to work properly. First, the PWM outputs are shorted to themselves. The PWM output controls the automatic gain control on the SDARS board, shorting them leaves it set to the default value. Second, the oscillators on the two boards need to be synchronized. This is done by moving a capacitor on the first board and connecting the REF_TEST output on the second board to REF_IN on the first.

The ADC also requires two modifications. The first is to provide power to the board. There are 5 power connections designed for hook connectors. Since permanent connections are needed, wire leads are soldered to the decoupling caps on the power
buses to provide an alternate power connection. Next, the ADC requires a reset whenever the sampling clock is changed. The reset is controlled by a micro-switch on the ADC board. This is overridden by soldering a BJT to the terminals of the micro-switch, then the BJT is driven by an output of the FPGA board.

5.2.4 Power Routing

The four development boards and logging PC require the availability of +5V, +3.3V, +1.8V, +1.5V, and -5V DC power. The FPGA, logging PC, and Bias-Tees only require connections to 5V and GND. In the rack-mount system, due to space limitations the FPGA is powered through a pair of wires soldered to the back side of the barrel plug power connector. The Raspberry Pi is powered from the USB port on the FPGA board, which itself provides 5V power. This removes the need to solder a power connection to the PC board. The Bias-Tees are powered through an SMA port labeled DC, this is connected to an SMA to two wire adapter.

The SDARS boards each require +5V, -5V, and GND connections, these are connected through three two pin headers on the board. The ADC requires +3.3V, +1.8V, +1.5V, and GND which is supplied through removable a nine-pin connector soldered to the board.

5.2.5 Dataflow Signal Cabling

Internally, the SoOp RT box uses RG-58 cabling for analog RF connections. The parallel connection between the ADC and FPGA is over four 20 wire 1.27mm pitch ribbon cables. On the ADC side the cables are terminated with 20 pin crimp-on IDC headers, one for each channel. On the FPGA side, the four cables are combined and rearranged according to the pattern in the dataflow schematic to end at two connectors. Crimp-on connectors were found to be unreliable when the ribbon cables were re-ordered, so instead the cables terminate at through-hole pin headers which were soldered to the cable through a short length of circuit board.
There are two functional differences with the signal connections between the rack-mount and YES systems. The first is the rack-mount system serial connection is over the FPGA GPIO and is at TTL voltage levels. The YES system uses the DB-9 port on the FPGA board and switches at RS-232 voltage levels. The second is a one pin offset between the data input connection on the FPGA. When the two nine bit parallel signals from the ADC are combined on one 20 position connector, there are two unused positions. On the rack-mount system, the unused positions are at the very end of the connector as is shown in Figure 5.9. On the WB-57 there is an unused position at pin one, then another at pin 10 as is shown in Figure 5.10.
Figure 5.9.: FPGA Data Input Connector for the Rack Mount System

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>m1</td>
<td>NC</td>
<td>NC</td>
</tr>
</tbody>
</table>

Figure 5.10.: FPGA Data Input Connector for the YES System

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
<td>m2</td>
</tr>
</tbody>
</table>
6. FPGA Development

The FPGA in the SoOp RT system is a Xilinx Virtex-4 SX35. This part was chosen due to the large number of DSP slices available on the chip as well as for the availability of space rated version of the part, the Virtex-4QV SX55. The DSP slices are dedicated multiply-accumulate logic blocks, which improve the performance of the two large FFT stages required by the circular correlation algorithm. The design HDL was implemented in Verilog as well as with IP from the Xilinx CORE Generator. All synthesis, mapping, and placement was performed using the Xilinx ISE.

As described in the algorithm section, the FPGA is tasked with computing the correlation waveform given four signals from the ADC, which represent the Inphase and Quadrature components of the direct and reflected input signals. In addition to that coherent correlation, it also provides an incoherent average of the results which serves to both lower the noise floor in the resulting waveform and to reduce the amount of data required to be sent to the logging PC.

6.1 FPGA Architecture Overview

To accomplish the correlation processing and time averaging, the FPGA segments the 16-bit complex input data from the ADC into 8192 wide sample blocks, which form a coherent sample window. These are input to the data processing pipeline, which applies the correlation algorithm to each window. After this data processing operation, the half of the waveform representing a positive time delay is time averaged to generate a 4096 element waveform representing approximately two seconds of sensing.

As is seen in Figure 6.1, the design is separated into four partitions. Note that these are architectural partitions only and do not match the HDL hierarchy. The
first partition is the data capture stage, an asynchronous interface between the ADC input signals and the local FPGA global clock (gclk) domain. The second is the correlation stage, which performs the core signal processing operations. The third is the averaging block, which selects the positive time delay bins and computes the time average. The last is the control block, which includes the serial port state machine as well as the clock generator for the ADC.
Figure 6.1.: SoOp RT FPGA Flow Chart. The dotted blocks represent inputs and outputs outside the FPGA. The solid blocks represent the logic partitions on the FPGA itself. The data on the FPGA flows from the ADC capture partition, to the correlation processing partition, to the magnitude and averaging partition, and finally to the IO control partition, where it is transferred over serial to the Logging PC.

HDL Conventions

Several coding conventions were used during the HDL development. Since the pipeline depth of the design is dependent on the Xilinx IP used for the FFT, complex
multiply, and vector magnitude calculations, all cycle names are relative to the IP signals. The "_mX" suffix denotes a signal that is X cycles earlier than base signal name, while "_pX" is a signal X cycles later than the base name.

The Xilinx design tools can infer latches and flops from behavioral code, however in order to maintain consistent behavior most flops in the design are encapsulated in a layer of hierarchy. These are called dflop.v or dflop_X.v where X is the width of the input and output bus.

Additionally, the data busses in the design are big-endian; the most significant bit is in the smallest bit position. The only exception to this is the input data bus from the ADC input pins. This bus is little-endian and requires an endian conversion before it is captured by input buffer memory.

**Pipeline Latency**

While the design is not optimized to reduce the overall pipeline delay, the ADC sample rate of 16MHz combined with the internal FPGA clock frequency of 100MHz, imposes a maximum constraint on the non-pipelined cycle delay the design can tolerate before data is dropped. A delay of more than 6 cycles per sample would cause the input buffer to overflow. Because of this, any non-pipelined operation on the entire window must take less than 49152 cycles.

### 6.2 Data Capture

The Data Capture partition contains the logic for the interface between the external ADC and the data processing pipeline, it consist of two 8192 wide arrays that buffer the input data as well as the capture logic. The primary challenge for this partition is crossing from the 16MHz clocked data inputs to the 100MHz internal gclk domain.

The ADC outputs data across four 13 pin parallel connections. 12 bits are data in binary offset format and the 13th bit is a clock signal. The 52 pins are connected to
the GPIO pins of the ml402 board and are labeled in the HDL as ch1_data, ch2_data, ch3_data, and ch4_data. The clock for each connection is available as well, however all are tied to the same 16MHz sampling clock which matches the ADC’s sampling frequency. This allows only the first channel’s clock, ch1_pclk, to be used as the capture clock for all four input channels.

The four input data channels from the ADC are first captured in the four 8x2 data_buf registers, which are clocked by ch1_pclk. The register is two bytes wide and is switched by the ch1_sw signal which toggles on the positive edge of ch1_pclk. These are the only two uses of the ch1_pclk signal, in order to simplify internal timing constraints.

Figure 6.2.: Logic diagram of ADC asynchronous interface and input buffers. The input data from the ADC first passes through two registers. The first synchronizes the input data to the ADC clock and the second to the internal FPGA clock. The synchronized data is then sent to the two input buffers.

The output of the data_buf register then reaches the data_buf_gclk register, which is clocked by gclk. At this point the signal is in the gclk domain and is sent to the 8192 deep input buffer memory. The write address for that memory is incremented with each rising edge of ch1_pclk, as is shown in the timing diagram in Figure 6.3.
Figure 6.3.: ADC Input Signal Timing Diagram. The ch*_data bus is first synchronized with ch1_pclk, the ADC clock, to create the ch*_data_buf bus. This is then synchronized with gclk, the FPGA clock, to create ch*_data_buf_gclk.

The hand-off to the correlation processing partition is made through the input_buf_mem0,1 memories. These are two 8192x32 dual port RAM blocks, built using the Xilinx Block Memory tool. The first, input_buf_mem0, contains a buffer of the channel 1 and channel 2 data, which represent the Inphase and Quadrature components one of the direct input signal. The second, input_buf_mem1, can alternately contain channel 3 and 4 (Inphase and Quadrature of the reflected signal) or a copy of channel 1 and 2. There is a multiplexer (mux) preceding the memory that selects between the two. In normal operation, channel 3 and 4 are always selected. However if there is a need to debug the internal correlation logic, it can be useful to compute an auto-correlation of one signal instead of the cross-correlation of both, so the select line on that mux is controlled by a dip-switch on the FPGA board. This can select channel 1 and 2 to be replicated in the second memory as well.

The input buffer memories are each 32 bits wide, though they only contain two 8-bit signals. The additional 16 bits are used to prevent the correlation processing stage from reading data from a sample window that is still in the process of being captured. This is necessary since the internal clock is more than six times faster than the sample clock. This switch is performed inside the blkmem_8192x32_wr macro, which switches between writing to bit locations 0:15 and 16:31 whenever an address value of 8192 is reached.
When the correlation processing stage is ready for a new sample window, it waits for the mem_valid[0:1] signal to go high. This is a one hot two-bit wide signal that marks which side of the memory is not currently being written to. By triggering the data processing on the positive edge of the mem_valid signal, the correlation processing logic always starts working on data as soon as the window is complete.

6.3 Correlation Processing

The correlation processing partition applies the circular correlation algorithm that contains the majority of the data processing on the FPGA.

There are four operations performed as part of the correlation processing stage. The stage first performs an FFT on each input signal. Once the signals are in the frequency domain, there is then an integer multiply operation to apply the bandpass satellite selection filter to each signal. That is followed by a complex conjugate multiply operation to perform the circular correlation. The result is then transformed back into the time domain by a final inverse FFT function. These operations are contained in three macros, transform_filt.v, conj_mult.v, and ifft_wr.v, as is shown in Figure 6.4.
Figure 6.4.: Correlation Processing Stage. The data in the correlation processing stage follows the circular correlation algorithm. An initial FFT is followed by a conjugate multiply operation which is followed by an inverse FFT.
The first macro, transform_filt, contains the logic for the first FFT operation on the input signals as well as the bandpass multiply logic. The FFT calculations are performed by an 8192 deep two input FFT from the Xilinx LogiCore library. This logic core implements a Radix-4 FFT algorithm with an 8192 long transform length. The resulting transform is represented by unscaled and full 22-bit coefficients. The FFT is configured to use a minimum amount of memory, by using a 3-multiplier configuration along with CLB logic for the butterfly arithmetic. This results in a core that requires 36 DSP slices and 774Kb of block ram.

The input data has a three clock cycle offset. This allows the FFT core to index the input data instead of requiring an external counter. The xn_index output is used as the input buffer read address, the output of which is then sent back to the FFT. Since this only takes one cycle the data is then delayed two cycles to match the expected input address.

As discussed in 6.2 The FFT calculation is triggered by a change in state of the memory_valid signal from either bank of the input buffer memory. When that occurs, the FFT begins incrementing the request address location. The 8-bit data from the input buffers then flows into the FFT logic. The data address increments once per cycle and when it reaches the final address, the FFT switches into a busy state until it completes the processing. The FFT then begins incrementing the output address, xk_index, and the 22-bit data matching that address is placed on the dir_fft and ref_fft outputs. This entire process consumes 30889 clock cycles.

Since the data is read out serially from the FFT, this is a convenient place to perform the filtering operation. A 8192 by 44-bit ROM contains the filter. By default, a simple boxcar filter is used, which multiplies all the frequencies in the pass range by one and all other frequencies by zero. Since at this point the input data still contains data from both satellites as well as any terrestrial broadcast data, this is necessary to reduce the data set to include only the desired satellite. For testing the desired satellite was XM-3, which is in the upper part of the 6 MHz frequency band sent from the SDARS boards. Changing the satellite selection, or applying a more
complicated filter can be performed by editing the ROM initialization file found in the Xilinx Block Memory Generator dialog and then re-synthesizing.

The 22-bit output of the filter is then sent to the conjugate multiply macro, conj_mult. This macro is mostly a wrapper around a complex multiplier based on Xilinx IP. The Xilinx IP is used to target the DSP cores on the Virtex-4 FPGA, the 22-bit multiply uses 16 cores. The multiplier is also fully pipelined, which masks the 10 cycle delay it requires to complete. In order to maintain cycle equivalence with the address and valid signals, there are also two sets of ten cycle deep shift registers.

After the multiplication operation, the output data has grown in width to a 45-bit integer. This runs into a limitation of the Xilinx FFT core, which has a maximum input size of 32-bits. Two methods were investigated to reduce the data width to fit that requirement. The first looked for the highest order bit the in data set, set that as the most significant bit and included the 32-bits consecutive to it. However, while it was possible to keep a consistent range for a single sample window, different data in multiple windows would not necessarily be consistent.

The second method attempted, and what was eventually used for testing, was a hard coded range which dropped the same bits for each window. Based on test data, the upper 13 bits from each sample are dropped. These bits represent the most significant bits in the data, however in practice these bit locations were always empty since they were generated by a multiplication of two FFTs, the source data for which is non-periodic. This property implies that none of the coefficients would be very large. In practice, this was found to be true of the test data.

The final step of the correlation processing stage is an inverse FFT in the ifft_wr macro. Similar to the first FFT, this is based on a full width Radix-4 FFT implementation. Since the input data is 32-bits wide, this results in 46-bit wide output data after a delay of 30952 clock cycles.
Bit Width Expansion

The FFT algorithm is based on a Radix-4 decomposition, which breaks the 8192 element wide FFT into seven successive stages. Since 8192 is not a power of four, only the first six stages are Radix-4 and the final uses a Radix-2 decomposition. This algorithm was chosen to minimize the memory footprint of the FFT while meeting the latency target. A Radix-2 algorithm would have saved two block rams, but would have more than doubled the latency to 69898 cycles and exceeded the limitation described in Section 6.0.1.

Each stage of the FFT consists of an addition/subtraction operation followed by a multiplication with the complex phase twiddle factor. Since both the summation and multiplication can result in a carry over bit, to represent the full result the output signal must be larger than the input. The Xilinx library provides a scaling feature to constrain this bit growth, however this requires that the input data be well characterized since the scaling either clips strong frequency components or drops low frequency components. Since only a limited dataset was available for testing, this may have resulted in lost data if the scaling factor was misjudged. The amount of bit growth can be calculated with:

$$output\_width = input\_width + \log_2(FFT\_Length) + 1$$

Which with an 8-bit input width and an 8192 bin FFT results in a 22-bit output width.

6.4 Averaging

The output of the correlation stage represents a coherent circular correlation of 8192·0.0625 μs samples, which results in a total window length of .512 ms. In order to both reduce the data throughput requirements as well as to reduce the noise present in the correlation result, blocks of 4096 windows are averaged together. This provides a 2.1 s long time average of the correlation waveform.
The two 46-bit buses from the ifft_wr represent the real and imaginary components of the correlation result. Since only the magnitude of the correlation is required, this result can be compressed by treating the result as a vector and finding the norm of that vector. This is done in the mag_8 macro. However only the lower 32-bits of each bus is used for the calculation. This is done to save bandwidth when later transmitting the result over the serial connection, since in practice those bits go unused.

The mag_8 macro is a wrapper around a Xilinx CORDIC vector processing core. It takes 43 cycles to compute the magnitude of a 2-D vector and maintains the 32-bit data precision. The macro also includes two shift registers to delay the address and valid signals the same number of cycles.

After the data is compressed into a single 32-bit wide bus, it is sent to the output buffer, which is in the corr_output_buffer macro. This is the macro that performs the average across 4096, or 2^12, sample windows. However, it first selects only the half of the waveform that represents positive time delays. This is done by a mux with the select line tied to a control register than can be set by the serial data connection. This allows the setting to be changed by the logging PC to account for an incorrect connection to the direct and reflected sma ports on the chassis, which if reversed, would flip the waveform in the time axis.

The averaging itself is performed by storing the first window to an array, then summing the data from each new window with the result already in the array and saving it back to the same location. When 4096 windows have been summed, the data has grown by 12 bits. The 44-bit result is divided by 4096 by dropping the 12 least significant bits. When this occurs the avg_done flag is set, which notifies the control partition that data is available.
6.5 Control

The control partition contains both the data transmission logic as well as the logic to respond to serial commands. Additionally, tasks such as setting the ADC serial clock and storing debug data are performed by the control partition.

The ADC sampling rate is set by an external clock input; when this clock changes the ADC development board requires a micro-switch to be toggled in order to reset the clock synchronization of the on-board deserialization chip. This requires two inputs from the FPGA: the sampling clock itself, and a reset override signal for the micro-switch. Both are generated in the sample_rate_clk_gen.v macro. The clock signal is the output of a two stage clock division from the 100MHz global clock. The first stage takes it down to 32MHz, then the output stage divides that by two to create the 16MHz srate_clk signal. The clock is then output over to the D8 pin location, which is tied to the DIFF CLK OUT sma port on the ML402 board. The adc_ovrd reset signal switches high on initial power-up and stays high for one second, this is long enough for the ADC board to power up and acquire the sampling clock. Since the override signal is output over the ML402 GPIO pins, which switch at 3.3V, there is an inline bjt transistor to isolate the microswitch, as described in Appendix ADDREF.

The serial controller is spread over three macros: uart_txrx.v, control_sm.v, and corr_debug_buf.v. The first, uart_txrx, contains the bit level state machine that implements the serial protocol. The next, control_sm, is the state machine that parses the byte level inputs received from uart_tx. The last is a dedicated storage buffer that stores a copy of intermediate signals from the data flow pipeline and is only used for debug.

Uart_txrx is built around an 8-bit, non-parity, single stop bit implementation of an asynchronous serial protocol. The serial clock is set similarly to the sampling clock for the ADC. There is a two stage clock divider. The first divides the clock to 58.8235MHz, the next divides that intermediate signal by $\frac{2}{9}$ for a clock signal of 114.9
kbaud, which within the tolerance for 115.2 kbaud communication. Any incoming data bits are saved to a byte buffer and output as an 8-bit bus rx_data. At that point the rx_byte_rec signal goes high to notify that a new byte has been received. Any outgoing bytes are placed on the tx_data bus and the transmission is started with the tx_en input. When the transmission is finished, the tx_done_gc signal is set high.

Control_sm contains a state machine that sets internal configuration registers based on the serial input data.

Corr_debug_buff is not used in the default run mode, it is only enabled when the debug configuration registers are set. There is an input mux that selects between four 32-bit internal signals. The selected signal is saved to an 8192x32 memory in order to capture an entire window of data. That memory is a buffer that is redirected to the serial output when the appropriate configuration register is set.
7. SoOp RT External Control

The SoOp RT System is controlled by a UART serial interface to the Xilinx SX35 FPGA. This interface is compatible with most UART interfaces at TTL logic levels, however the two deployments to date have included single board computers with the SoOp RT system to manage the serial interface. In the rackmount system, the Raspberry Pi logging PC manages the interface either interactively or through four batched Python scripts. In the YES system, the same scripts are run exclusively in batch mode on the YES logging PC. This mode provides for continuous data collection as well as periodic sampling of the debug registers to support calibration or debugging efforts that might be necessary after the experiment.

While the single board computers control the serial interface with the FPGA, they require additional interfaces to communicate with the outside world. The YES logging PC is managed by YES and control is limited to specifying which scripts to run for the experiment and which data files are to be removed after. The Raspberry Pi can be controlled directly and so two interfaces are exposed, both of which are required in normal operation. The first is through a terminal connection to the onboard TTL serial port. This connection is available immediately after power on and logs the boot status of the Raspberry Pi. Once the boot process is complete, this terminal can be used to access a Linux command prompt. If the SoOp system is only being used for data collection, this prompt is sufficient to launch the logging scripts. However the serial connection does not support a convenient method for data transfer. After data collection ends and the recorded data must be removed from the SoOp RT system, an Ethernet connection must be used instead.

The Raspberry Pi runs a full Linux distribution which includes an SSH server that loads on boot. This allows any computer on the same network to access it. However the Raspberry Pi’s IP address is assigned by DHCP, so the IP address must
first be discovered. This can be done through the serial connection as is described in the data removal procedure below. Alternately, if the user has access to the DHCP routing tables for the local network, the IP address can be found there and the serial connection can be skipped. Once the IP address is known an scp connection can be made to the Rapsberry Pi and the data can be transferred to the desired location.

7.1 Data Recording Procedure

Before powering on the SoOp RT System, all inputs and control connections should first be attached, including the Serial converter USB port on the front panel. That port should be attached to an external PC with serial terminal software installed, such as hyperterm on Windows or minicom on Linux. Only the FPGA Programmer USB port can be left empty.

As discussed in Section 5.2 The USB Serial Port connects through an FTDI active serial cable. Drivers for this cable are included in Windows 7 as well as the Ubuntu Linux distribution. Note that the active serial cable is powered through the USB port, so the SoOp box does not need to be powered on to establish a connection to it. Once that cable is attached, a serial terminal should be opened on the external PC with the following settings.

Table 7.1: Serial Settings

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate</td>
<td>115200</td>
</tr>
<tr>
<td>Bits</td>
<td>8</td>
</tr>
<tr>
<td>Stop Bits</td>
<td>1</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Flow Control</td>
<td>None</td>
</tr>
</tbody>
</table>

Once the terminal is open, the front power switch of the SoOp RT can be flipped to the on position. The terminal should immediately show the boot process of the
Linux kernel on the Rapsberry Pi. The boot process takes approximately two minutes
to complete, at which point the terminal will display a log in prompt. Log in with
the username and password in Table 7.2

Table 7.2: Username and Password

<table>
<thead>
<tr>
<th>Username</th>
<th>pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>mllab</td>
</tr>
</tbody>
</table>

Once log in is complete, the Raspberry Pi provides a full Linux command prompt.
The control scripts are located in the python directory, so first move to that directory
with:

cd /python

From the python directory any of the four control scripts described in subsection
7.3 can be executed.

7.2 Data removal procedure

Follow the Data Recording procedure until presented with the command prompt
over the serial terminal. At that point the IP address of the Raspberry Pi can be
discovered by running:

ifconfig -eth0

Once the IP address is known, a PC on the same network as the Raspberry Pi can
access it over SSH. On Windows the SSH Secure Shell Client provides file transfer
access over the SSH connection. On Linux the scp utility can be used instead.

The logging scripts save the recorded data to .dat files in the same directory they
were executed in. If they were launched from the python directory, all .dat files can
be transferred from that directory to the external PC and then deleted.
7.3 Control Scripts

The four control scripts are built on a common template. They first set the control registers on the FPGA to the desired output setting, send a data read request, then save the resulting data to a binary .dat file. The .dat file name is a combination of the script name and a numerical time. The time format is: week, month, year, hour, minute and second.

The scripts are described in Table 7.3

<table>
<thead>
<tr>
<th>Script Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>xm_dbg_read_input.py</code></td>
<td>Reads Four 8-bit Input Buffers</td>
</tr>
<tr>
<td><code>xm_dbg_full_corr.py</code></td>
<td>Reads rounded 32-bit IFFT stage output</td>
</tr>
<tr>
<td><code>xm_dbg_mag.py</code></td>
<td>Reads rounded 32-bit Magnitude stage output</td>
</tr>
<tr>
<td><code>xm_dbg_cm.py</code></td>
<td>Reads 32-bit Averaged stage output</td>
</tr>
</tbody>
</table>

The four scripts each read and capture a different debug buffer. By collecting data from each the data from the entire datapath in the SoOp RT FPGA can be reconstructed. This can assist debug efforts as well as checking for any overflow conditions that might have occurred due to calibration issues. For this reason, it is recommended to dump the entire set of data periodically, even during normal data collection operation.

7.4 ASCII Control Strings

The SoOp RT system can also be controlled by directly writing the ASCII control strings over the serial connection to the FPGA. The output is then received as raw binary data over the serial connection. This process is performed inside the Python scripts when creating the .dat files.
Each command recognized by the SoOp RT is a 4 character ASCII string followed by a carriage return character. The types of commands are broken down into two types. The first command sets or resets an output enable register. These enables are one-hot, so unless the state of the other registers is already known, it is good practice to disable the remainder of the enable registers before any read request. However once the output is set, the register can stay enabled until the output buffer changes.

Table 7.4: Output Enable ASCII Commands

<table>
<thead>
<tr>
<th>L0ON</th>
<th>Enable Input Buffer Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0OF</td>
<td>Disable Input Buffer Read</td>
</tr>
<tr>
<td>L1ON</td>
<td>Enable FFT output Read</td>
</tr>
<tr>
<td>L1OF</td>
<td>Disable FFT output Read</td>
</tr>
<tr>
<td>L2ON</td>
<td>Enable Magnitude output Read</td>
</tr>
<tr>
<td>L2OF</td>
<td>Disable Magnitude output Read</td>
</tr>
<tr>
<td>L3ON</td>
<td>Enable Averaged output Read</td>
</tr>
<tr>
<td>L3OF</td>
<td>Disable Averaged output Read</td>
</tr>
</tbody>
</table>

The next command requests the output read from the FPGA. After receiving this command the selected output buffer will be dumped in entirety over the serial connection. Once the read is complete, this register must be reset before an additional read operation can be performed.

The serial connection transfers data out of each output buffer in 8-bit words. When reading the four 8-bit input buffers, this matches well with the stored data and that data is transferred as four sets of 8192 8-bit binary numbers. The first represents the channel 1 input buffer, the second the channel 2 input buffer, etc.

The remaining three modes contain sets of 32-bit numbers, which does not align well with the 8-bit buffer structure. To simplify the read operation, these are treated
the same was as the input buffers and that data is sent in four batches of 8192 8-bit numbers. The first set represents the highest order bits, the next set the next highest order bits, etc. After the data transfer is complete, these must be reconstructed back into 32-bit integer form by combining each element with the corresponding elements in the other 8192 wide arrays.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0ON</td>
<td>Depreciated</td>
</tr>
<tr>
<td>R0OF</td>
<td>Depreciated</td>
</tr>
<tr>
<td>R1ON</td>
<td>Start Output Read</td>
</tr>
<tr>
<td>R1OF</td>
<td>Reset Read Register</td>
</tr>
</tbody>
</table>
8. Experimental Results

In parallel with the final development of the SoOp RT system, the receiver underwent testing to further refine the design. The SoOp RT was tested in the Purdue Radio Navigation Lab (RNL), then fielded on two deployments. The first was on a NOAA WP-3D Hurricane Hunter in Alaska and the second on a NASA WB-57 in Texas. Both deployments uncovered operational issues and resulted in design changes to the SoOp RT receiver.

8.1 Purdue RNL Testing

The initial testing of the SoOp RT receiver was performed at the Purdue RNL lab. The bench setup is shown in Figure 8.1. The SoOp system was tested with both the direct and reflected SMA inputs connected to LHCP upward facing XM antennas installed on the roof of Armstrong Hall. The antennas were both Audiovox/Terk XM6 XM Radio Home Antennas. This model has an antenna pattern with 8.5 dBi Gain, as well as 30dB gain due to the included LNA. An inline amplifier was placed on each channel, an 18dB Terk XMAMP18 also designed for XM radio receivers.

Prior to connecting the SoOp RT, both lines were tested for the presence of an XM radio signal. A Rohde and Schwarz FS300 Spectrum Analyzer was used to observe the power in the XM spectrum, as is show in Figure 8.2. The spectrum analyzer is centered at 2335.305MHz, the center of the A2 channel from the XM-4 Blues satellite. The A1 XM-3 signal can be seen at 2333.465MHz and the B1 at 2344.045MHz. Both are between 5-10 dB above the noise floor as seen in the portion of the band dedicated to terrestrial repeaters (which are not active near the Purdue test site.)
Figure 8.1.: SoOp RT rackmount system undergoing testing at Purdue RNL

Figure 8.2.: XM Spectrum at Purdue RNL
The first tests included both antennas in an identical configuration. A cross-correlation of both signals in this case is expected to return the auto-correlation of the XM signal. The debug features of the SoOp RT were used to dump the contents of the input buffers on the FPGA, this provided a second confirmation of an XM signal. The correlation result was then observed, as can be seen in Figure 8.3. The expected autocorrelation is the triangle function shown in 3.5 and outlined in green on the plot. The waveform closely matches that function with some spreading and side bands due to the filtering performed in the FPGA. This result provided initial confirmation that the datapath processing of the signal was functioning correctly.

A delay line was then introduced between the antenna and the reflected input port. The delay line consisted of 600 feet of RG-6 cable with two additional Terk XMAMP18 amplifiers inserted every 300 feet. The expected waveform is the same autocorrelation shape delayed by approximately .9 µs. However, the antennas do not have an identical run length and a .3µs delay difference was observed between the two, which results in an expected delay of roughly .6 µs. The observed result matches this, as is seen in Figure 8.4 which plots the autocorrelation function at a delay of .56 µs in green. This result confirms that correct positive delay side of the waveform was transmitted by the FPGA.
Figure 8.3.: XM Autocorrelation as observed on the SoOp RT. The first plot shows the first 100 delay bins corresponding to the first 6.25 \( \mu s \), the second the complete correlation waveform.
Figure 8.4.: XM delay line cross-correlation. Peak of Correlation is shifted 9 bins (.56 µs) compared to Autocorrelation result.
8.2 NOAA WP-3D

A software radio based XM receiver had been developed to record raw XM signals as an extension of the work described in Section 2.3. This receiver contained two USRP 2 software radios, a Linux based PC, and a hard drive inside a 4U rackmount chassis. It was installed for the 2013 Hurricane season on the NOAA WP-3D Orion Hurricane Hunter aircraft, shown in Figure 8.5. The WP-3D is a research aircraft equipped with remote sensing instrumentation to study hurricanes and storms [20]. The SoOp receiver occupies rack space on the WP-3D and is connected to an LHCP upward facing antenna immediately in front of the vertical stabilizer as well as an RHCP downward facing antenna shown in Figure 8.6. Both antennas are closely matched Antcom units, the upper is a 5G121523RL-AA-XTT-1, while the lower is a custom ordered part with reversed polarization. At 90° the antennas have a 4.9dB gain with a 129° 3dB beam width [19]. The included LNA provides a 33 dB gain. This equipment was tested on a flight over the Gulf of Mexico west of Tampa, Florida in July 2013 and observed reflected XM signals from the ocean surface in calm conditions.

The equipment rack containing the software radio receiver also included an open 1U rack slot. This provided the opportunity to install the SoOp RT alongside the
software radio based system and to make comparative measurements using the same antennas under similar conditions. However during hurricane season the WP-3D flies on short notice, so there was not enough time to install the SoOp RT system prior to a flight. Post-hurricane season there was a planned flight series over the Northern coast of Alaska to measure sea ice development, called Arctic Flux. While these flights were in a geographically less desirable location, they included scheduled flights which would allow the short-term installation of the SoOp RT receiver.

The Arctic Flux flights began at Fairbanks, Alaska and transited north to either the Beaufort Sea or the Chuckchi Sea. The SoOp RT receiver was installed above the software radio receiver as is seen in Figure 8.7. Due to a lack of power connectors, only one receiver could be powered on at a time and the plan was to alternate between the two to collect data from similar terrain on both receivers. The typical flight path over the sea and sea ice consisted of a ladder pattern with the horizontal legs flown alternating at 200’ and 3500’ altitude, with a 6 minute duration. The 3500’ legs were the segments of interest and so the receivers were switched during the 200’ leg.
The first flight was to the Chuckchi sea on October 22nd, which is off the Northwest coast of Alaska and is outside the footprint of the XM-3 Satellite. Due to the location, XM signal reflections were not expected to be observed over the sea. However the receivers were turned on during the transit from Fairbanks to confirm their operation. This uncovered an issue with the software radio SoOp receiver. In the software radio receiver, a PC104 single board computer was responsible for logging data from the USRP radios. The VGA port on that board developed a hard failure due to the stress and vibration imparted by an unsupported cable. This in turn prevented the board from powering on and so no data was collected.

A fix to the PC104 board was attempted prior to the second flight on October 24th, since a replacement board was not available. Unfortunately the VGA error reoccurred and prevented the software radio receiver from recording data. This flight
however was over the Beaufort sea, at the edge of the XM-3 footprint and the SoOp RT receiver collected data over the course of the flight. This uncovered a problem with the original design of the data compression in the SoOp RT system.

The output of the final IFFT stage of the correlation algorithm results in two 46-bit integers. When these are converted to a serial signal with 8-bit magnitude the data is compressed and lower amplitude signals can be lost. In lab testing, the signals were strong enough that this was not an issue and the 8-bit compression did not mask the correlation result. However during the Alaska flights, only noise was observed at the SoOp RT output. Without the reference point of the software radio receiver, the signal strength at the SoOp RT receiver is unknown. It is possible that a reflected signal was hidden by the output compression. An example output waveform is shown in Figure 8.8.

![Figure 8.8: SoOp RT 8-bit correlation waveform from Fairbanks, Ak.](image)

The results of this experiment led to a change in how the data is output from the FPGA to the logging PC in the SoOp system. Instead of compressing the output data to 8-bit, the output is compressed to 32-bits which causes minimal loss of resolution compared to the raw 46-bit output. This did however necessitate a longer averaging
time compared to the original design. The original .52 s time average was increased four fold to 2.1 s.
8.3 Yankee Environmental Systems SoAp box

The Yankee High Definition Sounding System (YHDSS) is an automatic dropsonde deployer developed by Yankee Environmental Systems (YES.) One application of the YHDSS is the NASA WB-57 research aircraft, which flies at altitudes of 60,000 feet and higher with remote sensing payloads [20]. The WB-57 installation of this deployer contained additional space, power, and communication capabilities which supported the installation of a version of the SoOp RT system. As integrated into the YHDSS this is referred to as the SoAp box. In November 2013, the YHDSS along with the SoAp box underwent qualification testing on the NASA WB-57 aircraft offshore from Corpus Christie to Brownsville, Texas [21]. The chassis is shown in Figure 8.9.

Figure 8.9.: The SoAp box can be seen installed in the gray box underneath the foil wrapped ethernet adapter. Adapted from HDSS WB-57 Flight Test Summary [21]
Antcom antennas were again used with the receiver, though a different model than was installed on the WP-3D. The 3M23L-A-XT-2 LHCP XM antenna was installed in the upward facing direction and the 3M23R-A-XT-2 RHCP antenna in the downward facing direction. These models both provided a 4.6dB gain at $90^\circ$ and a 33dB LNA gain. Verification of the YHDSS system was the priority on this flight and so the SoAp box was not integrated until shortly before delivery. Additionally, all integration work was performed by YES employees. This greatly limited the testing and debug effort available prior to the flight.

As a result, while communication to the SoOp RT FPGA was successful and data was retrieved, the data does not appear to contain an XM signal in either the direct or reflected channels. This can be see in Figure 8.10 which shows the data stored in the FPGA input buffer. This plot shows only noise sampled through the ADC and not the expected amplified XM signal from the SDARs front end. Due to a lack of access to the receiver, the root cause of the missing signal has not been discovered. It could have been caused by any error in cabling between the ADC, SDARS boards, Bias-Tee, and the antennas.

This result uncovered the need for a simple signal presence test to be incorporated into the SoOp RT design. Since the XM signals are not decoded, in order to check the presence of an XM signal the power spectrum of the XM radio band must be observed. This is easily accomplished in the lab with a spectrum analyzer, but not when the receiver is installed in an aircraft. Incorporating this feature into the current design would be difficult. The SDAR front ends apply automatic gain control to the input signals which would distort the power spectrum. Controlling this gain would require the addition of two parallel interfaces to the SDARS boards from the FPGA. However instead of this, a follow on version of the SoOp RT will incorporate a more tightly integrated RF front end and ADC and will support a power spectrum measurement, as will be discussed in Continuing Work.
Figure 8.10.: Input Buffer sample from WB-57 flight. No signal present.
9. Continuing Work

Many of the difficulties in developing the SoOp RT system were due to relying on development boards from multiple manufacturers. The Xilinx ML402 boards are no longer in production, must be sourced used, and have a high failure rate. Out of three boards, one failed completely, one developed a fault with the compact flash card controller (which was necessarily bypassed in the SoOp RT design), and only one was fully functional. The SDARs boards were more reliable, however they are not consistently stocked and can require a four month lead time when ordering. Additionally, while the overall SoOp system was designed to function with any signal of opportunity with the right characteristics, the front end board limited the available signals to only XM radio broadcasts.

To address these issues, a custom single board solution was considered as a follow on to the SoOp RT. However before work began in that direction, Ettus research announced the availability of the USRP B210 software radio board. This board includes a configurable two channel 56 MHz wide front end which covers the spectrum from 70MHz to 6GHz [22]. However unlike previous USRP radios, such as were used in the initial SoOp receiver, the B210 includes an FPGA with comparable resources as the Xilinx Virtex-4 in the SoOp RT. This Xilinx Spartan6 part can reuse the same codebase that was developed for the SoOp RT and can support a real time cross-correlation. Additionally, it has access to open source software radio tools such as GNU Radio which includes a power spectrum viewer.

As of June 2014, transfer of the SoOp RT to the B210 board is being performed by a Design, Build, Test course offered at Purdue University by Professor James Garrison. The goal for the class is to complete a working receiver for September 2014 and to be included on the WB-57 for the 2014 Hurricane Season.
In addition to improving the SoOp RT hardware, continuing work could extend the applications for the SoOp RT receiver. Including a controllable gain front end would allow power measurements for uses such as soil moisture monitoring. Also, in the ocean surface sensing application the phase of the processed waveform is discarded. This data may be useful for as yet unidentified purposes and with a higher bandwidth data connection the SoOp RT could easily be extended to output phase data as well.
10. Conclusion

This thesis discussed the development and early testing of a real time bistatic remote sensing radar receiver, the SoOp RT. By using signals of opportunity as the transmission source, the bistatic radar system required development of only the receiver, a low cost and small footprint device. This builds on a history of bistatic remote sensing and applies the techniques developed for GNSS signals of opportunity to communications signals. XM radio signals in particular were investigated and then used as part of the bistatic radar system.

A circular correlation algorithm was implemented on an FPGA at the core of the receiver, which demonstrated real time processing of a 16MSps input signal. This required integration of five development boards consisting of an RF front end, ADC, FPGA, and a single board PC for data logging. A 1U rackmount chassis was built to house the SoOp RT system.

The FPGA implementation was built around an 8192 wide sample window capturing .5 ms of input data. This data was processed through an input FFT stage, a filter multiply operation, a conjugate multiply, and then an inverse FFT to complete the correlation. This complex result was then converted to a magnitude and compressed to 32-bits for data transmission over a 115.2 k baud serial connection.

The SoOp RT system was tested under three conditions. The first was in the RNL at Purdue University and verified the reception of XM signals and the calculation of the cross-correlation of a time delayed input. A flight on the NOAA WP-3D uncovered a need for higher resolution output data and led to design changes for the SoOp RT system. An additional flight on the NASA WB-57 highlighted the need for continued development of the system to simplify installation and operation by third party users.

Work is continuing at Purdue University to further develop the SoOp RT into a single board solution based on the Ettus Research USRP product. This will simplify
construction and increase reliability by reducing the number hardware components. Due to the configurable RF front end on the USRP, it also opens up multiple signal sources as transmission sources and will enable future investigations of alternate signals of opportunity.
APPENDIX
FPGA Development Tool Set

The Xilinx ISE version 14.4 was used to build the bitstream for Virtex-4 FPGA. The bitstream flashing guide included below was developed in support of the YES system, though it is also applicable to the rackmount SoOp RT.
Xilinx ML402 Bitstream Flashing Guide

The FPGA board is flashed through the Xilinx Impact utility. This is included as part of the Xilinx Webpack software, however note that only the “Lab Tools” need to be installed, not the full Xilinx ISE.

The latest version of Webpack is available here: http://www.xilinx.com/support/download.html

The version of webpack used for development and testing was the 14.4 Linux version, which has been preserved here: http://www.carbio.org/rnl_backup/Xilinx_ISE_DS_Lin_14.4_P.49d.3.0.tar.gz

Follow the instruction from Xilinx to install webpack, be sure to install the Programming Cable Drivers as part of the installation. On Linux this is an option, while it is the default behavior on Windows.

Using Impact

Before opening Impact, connect the programming cable to the USB port of the PC as well as the JTAG port on the ML402 development board. Open Impact and select “Configure devices using Boundary-Scan (JTAG).”

This will then configure Impact based on the devices detected by the Programming cable. If there is a problem with the cable or the cable drivers, this may result in an Error.

When the devices are successfully detected, Impact will prompt to assign a configuration file. Skip this step for now and click Cancel. Impact will then show a prompt to set the programming settings. Select
the second device, PROM2 xcf32p and configure the following settings:

Then click OK. The PROM2 chip is the one and only chip that will be programmed. Now impact shows 4 devices, with the xcf32p the second from the left. Right-click the xcf32p devices and click “Select Configuration File.” This will open a file browser where you can select the desired device image (the file will end in .mcs).

Once the configuration file is assigned, Right-Click the xcf32p again and select program. This will program the device, the process takes roughly two minutes to complete. Impact will show the following when finished:
The FPGA has now been programmed. When the FPGA Dev board is power cycled, the new code will now be loaded.
Simulation Python Code

#!/usr/bin/python
import matplotlib.pyplot as plt
import sys
import struct
import numpy.fft as fft
import random
import cmath
import math
import numpy

plt.close("all")

#ADC Settings
f_samp = 16e6
t_samp = 1/f_samp
coh_win_size = 8192
t_i = t_samp*coh_win_size
t_v = [x*t_samp for x in range(coh_win_size)]

default_wave = [math.sin(2*math.pi*(1/(53*t_samp))*t) for t in t_v]

#XM Signal Settings
t_symb = 1/(3.28e6/2)

corr_mag_avg = [0] * len(t_v)
range_avg = 10
for avg_ind in range (0,range_avg) :
    #Create Input Data 00,01,10,11
    input_data = []
data_len = 2*int(math.ceil(coh_win_size*(t_samp/t_symb)))
for i in range(data_len):
    input_data.append(random.randrange(0,4,1))

#Create Input Data CH2
input_data_ch2 = []
for i in range(data_len):
    input_data_ch2.append(random.randrange(0,4,1))

noise_var = 1

#Create Baseband QPSK
qpsk_sig_dir = []
qpsk_sig_dir_clean = []
qpsk_sig_dir_noise = []
for t in t_v:
    data_ind_dir = int(math.floor(t/t_symb))
    v_dir = cmath.exp((math.pi/4)*1j +
                      (math.pi/2)*input_data[data_ind_dir]*1j)
    v_noise = noise_var*random.random()
    qpsk_sig_dir_clean.append(v_dir)
    qpsk_sig_dir_noise.append(v_noise)
    qpsk_sig_dir.append(v_dir + v_noise)

qpsk_sig_dir_clean_mag = [cmath.polar(a)[0]**2 for a in qpsk_sig_dir_clean]
qpsk_sig_dir_noise_mag = [cmath.polar(a)[0]**2 for a in qpsk_sig_dir_noise]

clean_rms = cmath.sqrt(numpy.mean(qpsk_sig_dir_clean_mag))
noise_rms = cmath.sqrt(numpy.mean(qpsk_sig_dir_noise_mag))
qpsk_dir_i = [x.real for x in qpsk_sig_dir]
qpsk_dir_q = [x.imag for x in qpsk_sig_dir]

qpsk_sig_ref = []
#delays_ref = [0]
delays_ref = [5e-6, 8e-6] #, 9e-6, 9.5e-6, 9.8e-6]
for t in t_v :
    v_ref = 0
    for d in delays_ref :
        data_ind_ref = int(math.floor((t+d)/t_symb))
        v_ref = v_ref + cmath.exp((math.pi/4)*1j +
        (math.pi/2)*input_data[data_ind_ref]*1j)
    v_ref = v_ref/len(delays_ref)
    noise = noise_var*random.random()
    qpsk_sig_ref.append(v_ref + v_noise)

qpsk_ref_i = [x.real for x in qpsk_sig_ref]
qpsk_ref_q = [x.imag for x in qpsk_sig_ref]

#Create Baseband QPSK CH2
qpsk_sig_dir_ch2 = []
for t in t_v :
    data_ind_dir = int(math.floor(t/t_symb))
    v_dir_ch2 = cmath.exp((math.pi/4)*1j +
    (math.pi/2)*input_data[data_ind_dir]*1j)
    qpsk_sig_dir_ch2.append(v_dir_ch2)

qpsk_sig_ref_ch2 = []
for t in t_v :
    v_ref_ch2 = 0
    for d in delays_ref :
data_ind_ref = int(math.floor((t+d)/t_symb))

v_ref_ch2 = v_ref_ch2 + cmath.exp((math.pi/4)*1j + 
                  (math.pi/2)*input_data[data_ind_ref]*1j)

v_ref = v_ref/len(delays_ref)
qpsk_sig_ref_ch2.append(v_ref_ch2)

#RF
carrier_sig_dir = []
f_offset = 2344.045e6 # - (1/t_symb)/2 #6.155e6
f_offset_ch2 = 2333.465e6 # - (1/t_symb)/2 #6.155e6
for t in t_v :
    sig_ind = int(math.floor(t/t_samp))
    v_c_ch1 = cmath.exp(2*math.pi*f_offset*t*1j)*qpsk_sig_dir[sig_ind]
    #Complex, single sided
    v_c_ch2 =
              cmath.exp(2*math.pi*f_offset_ch2*t*1j)*qpsk_sig_dir_ch2[sig_ind]
    #Complex, single sided
    v_c = v_c_ch1.real # + (v_c_ch2.real - 1j*v_c_ch2.imag)
    carrier_sig_dir.append(v_c)

carrier_i = [x.real for x in carrier_sig_dir]

carrier_sig_ref = []
for t in t_v :
    v_c = 0
    for d in delays_ref :
        sig_ind = int(math.floor((t)/t_samp))
        v_c = v_c +
              cmath.exp(2*math.pi*f_offset*(t+d)*1j)*qpsk_sig_ref[sig_ind]
        #Complex, single sided
carrier_sig_ref.append(v_c)

# Downconversion
if_sig_dir = []
if_sig_dir_real = []
if_sig_dir_imag = []
if_sig_ref = []
if_sig_ref_real = []
if_sig_ref_imag = []
f_down = 2337.89e6 # 2338.755e6 #

print (f_offset + f_down)/1e3
print (f_offset - f_down)/1e3
print (f_offset_ch2 + f_down)/1e3
print (f_offset_ch2 - f_down)/1e3
for t in t_v:
    if_ind = int(math.floor((t)/t_samp))
    v_if_i = math.cos(2*math.pi*f_down*t) * carrier_sig_dir[if_ind]
    v_if_q = math.sin(2*math.pi*f_down*t) * carrier_sig_dir[if_ind]
    if_sig_dir_real.append(v_if_i)
    if_sig_dir_imag.append(v_if_q)
    if_sig_dir.append(v_if_i + 1j*v_if_q)
    v_if_i = math.cos(2*math.pi*f_down*t) * carrier_sig_ref[if_ind]
    v_if_q = math.sin(2*math.pi*f_down*t) * carrier_sig_ref[if_ind]
    if_sig_ref_real.append(v_if_i)
    if_sig_ref_imag.append(v_if_q)
    if_sig_ref.append(v_if_i + 1j*v_if_q)

sig_spec = fft.ifft(if_sig_dir)

sig_spec_len = len(sig_spec)
sig_spec_real = [a.real for a in sig_spec]
sig_spec_imag = [a.imag for a in sig_spec]
sig_spec_shift_real = sig_spec_real[sig_spec_len/2:sig_spec_len-1] +
        sig_spec_real[0:sig_spec_len/2]
sig_spec_shift_imag = sig_spec_imag[sig_spec_len/2:sig_spec_len-1] +
        sig_spec_imag[0:sig_spec_len/2]
sig_spec_mag = [abs(a+1j*b) for a,b in
        zip(sig_spec_shift_real,sig_spec_shift_imag)]
sig_spec_power = [a**2 for a in sig_spec_mag]

#Correlation
fft_dir = fft.fft(if_sig_dir)
fft_ref = fft.fft(if_sig_ref)
#fft_dir = fft.fft(default_wave)
#fft_ref = fft.fft(default_wave)

###LPF
#fft_len = len(d_fft)
#d_filt = list(d_fft[0:900]) + [0]*(fft_len-900*2) +
        list(d_fft[fft_len-900:fft_len])
#r_filt = list(r_fft[0:900]) + [0]*(fft_len-900*2) +
        list(r_fft[fft_len-900:fft_len])

cm = [a * b.conjugate() for a,b in zip(fft_dir,fft_ref)]
corr_raw = fft.ifft(cm)
corr_len = len(corr_raw)
corr_real = [a.real for a in corr_raw]
corr_imag = [a.imag for a in corr_raw]
corr_shift_real = corr_real[corr_len/corr_len/2:corr_len-1] +
        corr_real[0:corr_len/2]
corr_shift_imag = corr_imag[corr_len/2:corr_len-1] +
corr_imag[0:corr_len/2]
t_v_shift = [t - max(t_v)/2 + t_samp/2 for t in t_v]
corr_mag = [abs(a+b*1j) for a,b in zip(corr_shift_real,corr_shift_imag)]
max_corr = max(corr_mag)
corr_mag_avg = [a+b for a,b in zip(corr_mag,corr_mag_avg)]

psd = fft.fft(corr_raw)
psd_real = [a.real for a in psd]
psd_imag = [a.imag for a in psd]
psd_len = len(psd)
psd_shift_real = psd_real[psd_len/2:psd_len-1] + psd_real[0:psd_len/2]
psd_shift_imag = psd_imag[psd_len/2:psd_len-1] + psd_imag[0:psd_len/2]
psd_mag = [abs(a+b*1j) for a,b in zip(psd_shift_real,psd_shift_imag)]
lag_v = [(x-psd_len/2)/8192.0*f_samp/1e6 for x in range(psd_len-1)]

corr_mag_avg = [a/range_avg for a in corr_mag_avg]

Development Board Hardware Modifications
Description:

The MAX1127EVKIT ADC receives a clock signal from the ML402 FPGA to set its sample rate. This clock signal must be present at power on. If the clock signal changes after power on, the MAX1127EVKIT needs to be reset. This can be done by cycling the power or by pressing the microswitch at SW1.

Since the FPGA and ADC power on at the same time, there is no guarantee that the clock signal will be available in time to configure the ADC, even if power to the overall system is cycled. Since the microswitch cannot be accessed once the system is installed, it needs to be overridden by the FPGA. This is accomplished by connecting a BJT to either side of the microswitch, the base of the BJT is tied to an output pin from the FPGA.

At power on, the FPGA holds the output pin high for ~1 sec, during this time the BJT will be in saturation and the switch will be closed. LED DS12 on the ML402 will be lit and LED D3 on the MAX1127EVKIT will be off. When the output pin goes low, DS12 will turn off and D3 will turn on. D3 signals that the ADC has locked onto the sample clock and that the switch is again open.

Procedure:

A reset cable has been provided which includes the in-line BC547B transistor. One side terminates in a 2 pin connector and needs to be connected to Pin 2 of connector J6 on the ML402 FPGA board:
The other end needs to be soldered to either side of the SW1 microswitch on the FPGA. The black wire connects to the upper pad of the switch and the red to the lower:
Description:

The two MAX2140EVKIT boards require synchronized oscillators in order to prevent oscillator bias from pushing the baseband signals more than the coherent integration time apart. Since each SDARS board has a local 29MHz oscillator, in order to synchronize the two boards the local oscillator on one board needs to be bypassed and replaced by a reference signal from the other board.

The MAX2140EVKIT provides two SMA ports to support this, REF_TEST and REFIN. REF_TEST is the output of the 29MHz oscillator. REFIN is connected to an unpopulated capacitor pad C42. When the capacitor at C62 is removed and replaced at the C42 location, the connection to the local oscillator is broken and REFIN instead drives the clock signal.

Procedure:

On one board only, remove the capacitor at C62:

Solder the capacitor in place at C42:
Connect an sma cable from REF_TEST on the unmodified board to REFIN on the modified board.
LIST OF REFERENCES
LIST OF REFERENCES


