IMPLEMENTATION OF A SINGLE CHANNEL AUTOMATIC IDENTIFICATION SYSTEM (AIS) ON A FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Electrical Engineering

By

Pranav Ramesh Patel

Dayton, Ohio

August, 2013

IMPLEMENTATION OF A SINGLE CHANNEL AUTOMATIC IDENTIFICATION SYSTEM (AIS) ON A FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Name: Patel, Pranav Ramesh

APPROVED BY:

Monish R. Chatterjee, Ph.D.
Advisory Committee Chair
Professor
Electrical and Computer Engineering
University of Dayton

Eric J. Balster, Ph.D.
Committee Member
Assistant Professor
Electrical and Computer Engineering
University of Dayton

Robert J. McTasney, Ph.D.
Committee Member
Assistant Professor
Electrical and Computer Engineering
Air Force Institute of Technology

John G. Weber, Ph.D.
Associate Dean
School of Engineering
University of Dayton

Tony E. Saliba, Ph.D.
Dean, School of Engineering
& Wilke Distinguished Professor
University of Dayton

ABSTRACT

IMPLEMENTATION OF A SINGLE CHANNEL AUTOMATIC IDENTIFICATION SYSTEM (AIS) ON A FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Name: Patel, Pranav Ramesh
University of Dayton

Advisor: Dr. Monish R. Chatterjee

Automatic Identification System (AIS) plays an important role to track maritime vessels. In a world of Somali pirates hijacking maritime vessels and U.S. Navy destroyers slamming into Supertankers, AIS is an inexpensive low power wireless communication sensor solution used to identify and exchange information (GPS, heading, speed, etc.) with other maritime vessels. This research project is an implementation of a single channel Automatic Identification System (AIS) on a Field Programmable Gate Array (FPGA) chip used on common military signal intelligence systems. Unlike conventional AIS receivers, the design uses commonly existing components that are found in military signal intelligence systems. The AIS receiver algorithms developed in this project can successfully detect the AIS signal, demodulate the Gaussian Minimum Shift Keying (GMSK) modulation, decode, and encode the messages in National Marine

Electronics Association (NMEA 0813) standard format to be compatible with other maritime equipment.

To my beloved grandfather,

Somabhai Hargovindbhai Patel,

without him this journey would not happen.

ACKNOWLEDGEMENTS

I must offer my profoundest gratitude to Mr. Daniel Wetzel. Mr. Wetzel introduced me to Digital Signal Processing and helped grow my skills as a hardware engineer. Mr. Wetzel helped me understand how to apply DSP techniques in digital communications. Without him, a great deal of effort for this project would not have been possible.

I would also like to thank the faculty, staff and students at Air Force Institute of Technology for their support throughout graduate school. Finally, I would like to thank Dr. Monish Chatterjee, my research advisor, for providing me time and resources to help with my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND NOTATIONS

ADC	Analog to Digital Converter
AIS	Automatic Identification System
AM	Amplitude Modulation
ASCII	American Standard Code for Information Interchange
ASK	Amplitude Shift Keying
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
COG	Course Over Ground
CORDIC	COordinate Rotation DIgital Computer
CPM	Continuous Phase Modulation
CRC	Cyclic Redundancy Check
dB	Decibel
DDC	Digital Down Converter
DSP	Digital Signal Processing
E_b/N_0	Energy per Bit to Noise Power Spectral Density Ratio
FCS	Frame Check Sequence
FIR	Finite Impulse Response
FM	Frequency Modulation

FPGA	Field Programmable Gate Array
FSK	Frequency Shift Keying
GHz	Gigahertz
GMSK	Gaussian Minimum Shift Keying
GPS	Global Positioning System
GSPS	GigaSamples Per Second
HDLC	High-Level Data Link Control
Hz	Hertz
I/Q	In Phase/Quadrature
IALA	International Association of Marine Aids to Navigation Lighthouse Authorities
IC	Integrated Circuit
ID	Identification
IEC	International Electrotechnical Commission
IF	Intermediate Frequency
IMO	International Maritime Organization
ITU	International Telecommunication Union
kHz	Kilohertz
kSPS	KiloSamples Per Second
LO	Local Oscillator
LRC	Raised Cosine Filter
LREC	Rectangular Impulse Filter

LSB	Least Significant Bit
MHz	Megahertz
MSB	Most Significant Bit
MSK	Minimum Shift Keying
MSPS	MegaSamples Per Second
NMEA	National Marine Electronics Association
NRZI	Non-Return-Zero-Inverted
P_B	Probability of Bit Error
PM	Phase Modulation
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RAIM	Receiver Autonomous Integrity Monitoring
RF	Radio Frequency
ROT	Rate of Turn
SIGINT	Signal Intelligence
SNR	Signal to Noise Ratio
SOG	Speed Over Ground
SOTDMA	Self-Organizing Time Division Multiple Access
STDMA	Self-Organizing Time Division Multiple Access
SPS	Samples Per Second
SWaP	Size, Weight, and Power
VCO	Voltage Controlled Oscillator

VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuits
SWaP	Size Weight and Power

CHAPTER 1

I. INTRODUCTION

Throughout the course of history, maritime vessels have challenges of tracking each other. The Automatic Identification System (AIS) was developed to solve these challenges. AIS is an inexpensive low power wireless communication sensor solution used to identify and exchange information, e.g. GPS, heading, and speed, with other maritime vessels. In commercial applications, the sensor is mainly used for collision avoidance. In military applications, it is not only used for collision avoidance, but for friend or foe identification. Current AIS receivers require a set of dedicated components to process AIS messages. Size, Weight, and Power (SWaP) reduction is key for any new sensor development for the military. Current Signal Intelligent (SIGINT) systems are comprised of an Analog to Digital Converter (ADC) to quantize the Radio Frequency (RF) signal for further processing by a demodulation/decoding processing chain configured on a Field Programmable Gate Array (FPGA). These components can be reused to develop an AIS receiver thus reducing SWaP. This thesis covers the implementation of an AIS receiver utilizing components commonly found in SIGINT systems.

There are multiple architectures that can be used to develop an AIS receiver. Figure 1 shows a typical architecture of an AIS receiver and it is a similar designed

described in "FM Discriminator for AIS Satellite Detection" conference paper [1]. Most of the tuning is accomplished in analog. There is a dedicated integrated circuit (IC) to handle the demodulation, and a microcontroller to do the decoding. Analog tuning is a preferred method for reduction of power. The disadvantage is the tuning is not programmable for different RF frequencies, and additional hardware is necessary to tune to process signals at various frequencies thus increasing SWaP. For this reason, this architecture is not suitable for SIGINT systems.

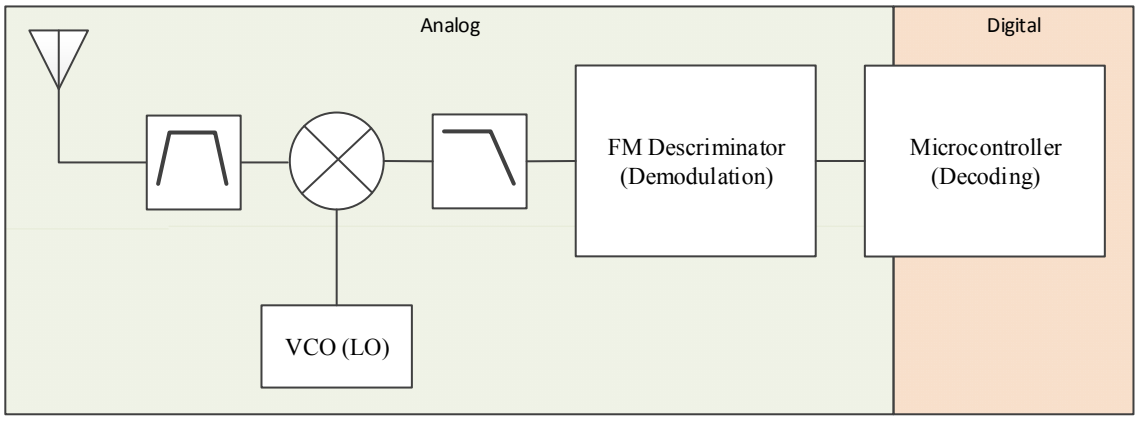


Figure 1: Typical Architecture of AIS Receiver

Figure 2 uses a DSP microcontroller to demodulate the signal. This architecture is similar to the architecture that is discussed on "Software Efficient Implementation of GMSK Modem for an Automatic Identification System Transceiver" conference paper [2]. The benefit of having a DSP microcontroller gives the user the ability to handle various digital demodulation but has the same fixed tuning as above. Also, an ADC is needed to quantize the analog signal for the DSP microcontroller to process the signal. After the demodulation occurs, the demodulated signal is transferred to a microcontroller or a workstation for decoding. This architecture adds additional overhead of a microcontroller or a workstation. Also, similar to the problems with the architecture in

Figure 1, additional hardware is needed to receive signals at different frequencies, thus increasing SWaP. For this reason, this architecture is not suitable for SIGINT systems.

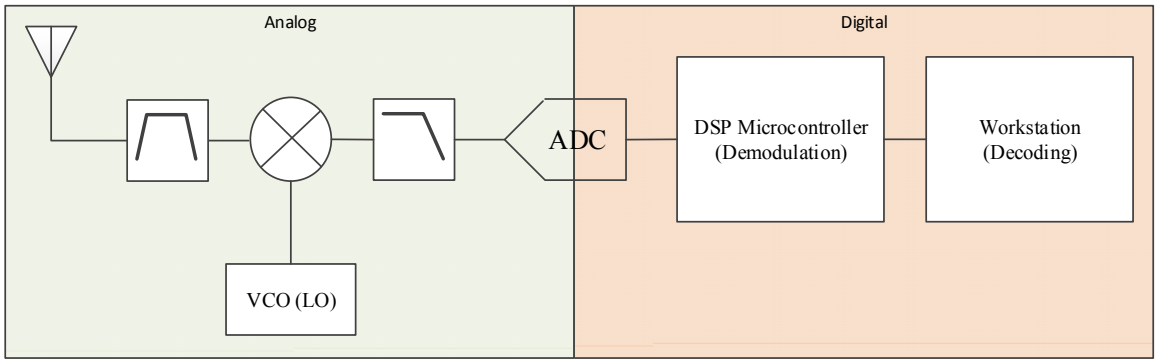


Figure 2: DSP Microcontroller Architecture

Figure 3 shows an FPGA/Microcontroller implementation of the AIS receiver [3]. The FPGA provides the ability to target many low speed and high speed radio applications. This architecture is commonly found on SIGINT systems. The demodulation effort is done in the FPGA and the decoding effort is done in a workstation or a microcontroller. The disadvantage is the added overhead of having a workstation or microcontroller to perform the decoding effort. The FPGA has the necessary bandwidth to digitally tune to the RF signal, thus only needing an antialiasing filter in the RF analog chain. Digitally tuning reduces insertion loss from analog components and have the ability to simultaneously tune to different frequencies in comparison to analog tuning [4].

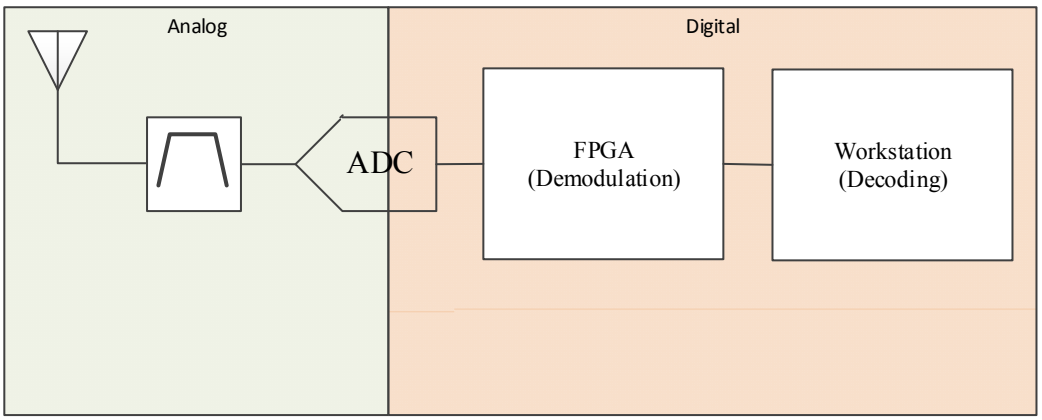


Figure 3: FPGA/Microcontroller Architecture [3]

Figure 4 shows the architecture, which is discussed in this thesis. The architecture on the figure below is mostly similar as the architecture in Figure 3, but the decoding algorithm is designed on the FPGA thus eliminating the need for a workstation or a microcontroller. The goal of this thesis is to reuse existing components on SIGINT systems and adding AIS receiver capability to the system thus being the most efficient design for reducing SWaP.

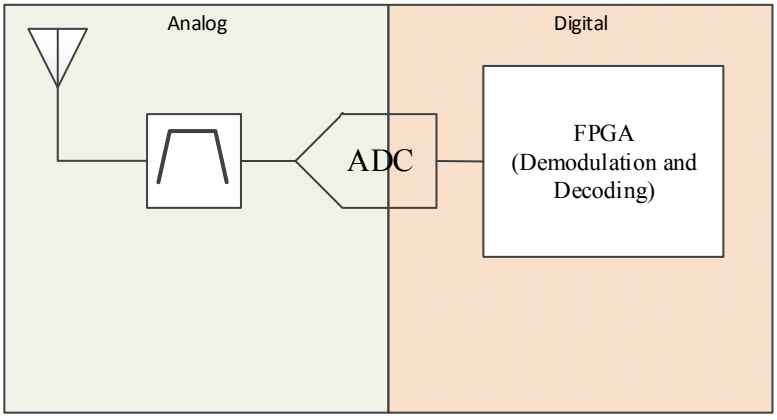


Figure 4: Minimized AIS Architecture Using Only FPGA

This thesis will discuss the system and hardware design of an AIS receiver using the architecture shown on Figure 4. Chapter 2 will present various digital modulation

schemes and details of the AIS system. Chapter 3 will discuss the algorithm development in MATLAB® and in VHDL. The theoretical and applied results will be discussed in Chapter 4. Finally, the conclusions are drawn in Chapter 5.

CHAPTER 2

II. BACKGROUND OF RECEIVER SYSTEMS

1. Digital Modulation

Digital communication uses various digital modulations in industry today. Digital communication has the following advantages over analog communications

- Most of the processing can be done on a single integrated chip
- Data can be secured by using encryption
- Much greater noise immunity using digital filters
- Error detection and correction by using data encoding methods
- Common packet format can be used on similar devices
- Less susceptible to jamming by spreading signal energy into different frequency bands
- Data throughput is higher with complex modulation schemes

The four major classes of digital modulation schemes are Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), and Continuous Phase Modulation (CPM). ASK, FSK, and PSK digital modulation schemes are shown on Figure 5.

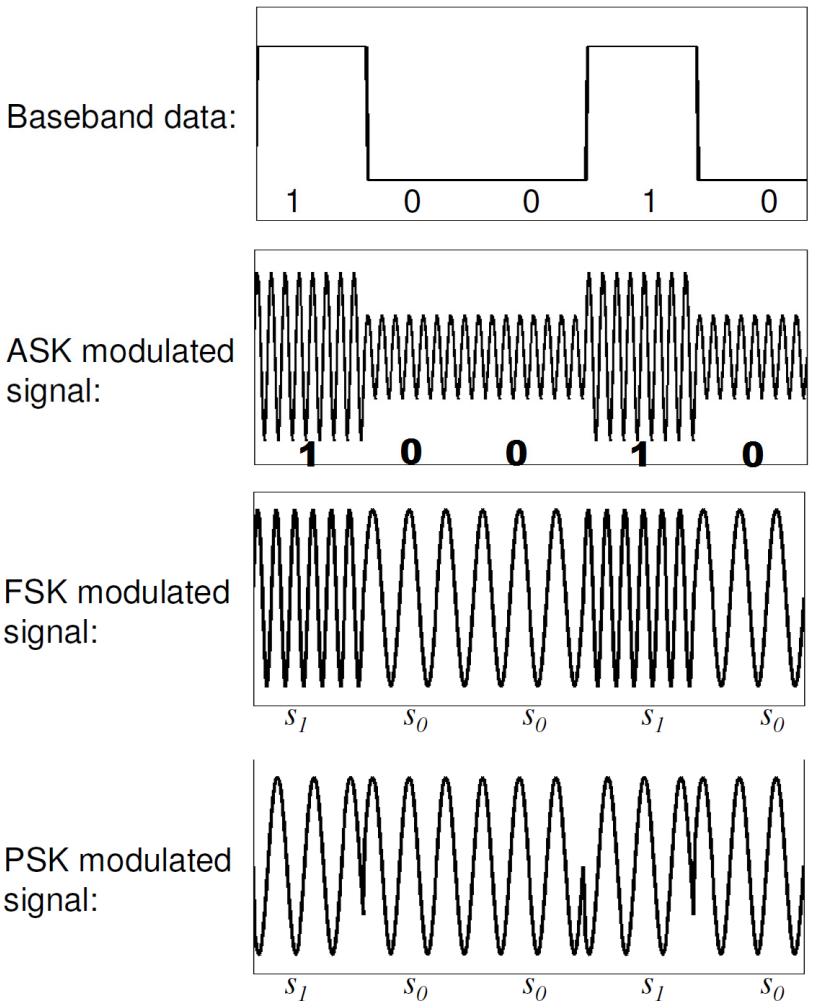


Figure 5: Typical Digital Modulation Schemes [5]

Amplitude Shift Keying (ASK) modulation is a type of digital modulation that varies the amplitude of the signal to the value of the bit at a specific carrier frequency. For every bit transition, the signal may exhibit some form of discontinuity caused by abrupt change in amplitude. ASK of a signal is analytically defined by [6]

$$s_i(t) = \sqrt{\frac{2E_i}{T}}\cos(2\pi f t + \theta), 0 \le t \le T, i = 1, \ldots, M \quad (1)$$

The probability of bit error (P_B) of the ASK modulated signal is analytically determined by [7]

$$P_B(E) = Q\left(\sqrt{\frac{E_b}{N_0}}\right) \quad (2)$$

Frequency Shift Keying (FSK) modulation is a type of digital modulation that varies the frequency of the signal for each bit around the carrier frequency. FSK has the same discontinuity problem as ASK modulated signals. FSK of a signal is analytically defined by [6]

$$s_i(t) = \sqrt{\frac{2E}{T}}\cos(2\pi f_i t + \theta), 0 \le t \le T, i = 1, \ldots, M \quad (3)$$

The probability of bit error (P_B) of the FSK modulated signal is analytically determined by [7]

$$P_B(E) = Q\left(\sqrt{\frac{E_b(1 - \frac{\sin(2\pi h)}{2\pi h}}{N_0}}\right), h = modulation\ index \quad (4)$$

Phase Modulation Keying (PSK) modulation is a type of digital modulation that varies the phase of the carrier signal for every bit change. PSK has the same discontinuity problem as ASK and FSK modulated signals. PSK of a signal is analytically defined by [6]

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi f t + \frac{2\pi i}{M}\right), 0 \le t \le T, i = 1, \dots, M \quad (5)$$

The probability of bit error (P_B) of the PSK modulated signal is analytically determined by [7]

$$P_B(E) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \quad (6)$$

Continuous Phase Modulation (CPM) is a scheme developed to suppress out-of-band interference by eliminating any discontinuity during phase transitions. CPM is used to generate various modulation schemes by using pulse-shaping filters. The most commonly filters are the following

Table 1: Common Pulse-Shaping Filters for CPM [7]

LREC	$g(t) = \begin{cases} \frac{1}{2LT} & (0 \le t \le LT) \\ 0 & (otherwise) \end{cases}$
LRC	$g(t) = \begin{cases} \frac{1}{2LT}\left(1 - cos\frac{2\pi t}{LT}\right) & (0 \le t \le LT) \\ 0 & (otherwise) \end{cases}$
GMSK	$g(t) = Q\left(\frac{2\pi B\left(t - \frac{T}{2}\right)}{\sqrt{\ln 2}}\right) - Q\left(\frac{2\pi B\left(t + \frac{T}{2}\right)}{\sqrt{\ln 2}}\right)$ $Q(t) = \int_t^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dt$
$L = Length\ of\ Pulse, T = Signaling\ Interval,$ $B = -3\ dB\ BW\ of\ Gaussian\ Pulse$	

The Automatic Identification System uses Gaussian Minimum Shift Keying (GMSK) modulation for its signal and is analytically defined by [2]

$$s(t) = \sqrt{\frac{2E_b}{T_b}}\cos(\omega_c t + \Phi(t)) \tag{7}$$

The probability of bit error (P_B) of the GMSK modulated signal is analytically determined by the following equation with $2\alpha = 0.68$ at BT = 0.25. BT is the 3-dB bandwidth-symbol time product where B is the one-sided bandwidth in hertz and T is in seconds. Smaller BT products produce larger pulse widths. Table 2 shows at 15 dB there is only 1 bit error for 1.76 million bits [8].

$$P_B(E) = Q\left(\sqrt{\frac{2\alpha E_b}{N_0}}\right) \tag{8}$$

Table 2: Theoretical Probability of Bit Error of GMSK Signal

Probability of Bit Error of GMSK Signal	
E_b/N_0 (dB)	Theoretical BER
0	0.204793362060363
1	0.177420509263482
2	0.149603257733327
3	0.122048299350502
4	0.0956171549523315
5	0.0712687518259178
6	0.0499503260355157
7	0.0324390797629668
8	0.0191628344299724
9	0.0100601183303870
10	0.00455789362625416
11	0.00171749339373707
12	0.000513769577265486
13	0.000115050452750955
14	1.79110573094723e-05
15	1.76592827469043e-06
16	9.80444483231651e-08

2. Background of Automatic Identification System

The Automatic Identification System (AIS) is an autonomous, automatic, continuous broadcasting system used on maritime vessels. AIS provides an inexpensive method to identify and exchange information, e.g. GPS, heading, and speed, with other maritime vessels for collision avoidance. The AIS standard was developed in 1998 with collaboration from various international communities, e.g. International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA), International Maritime Organization (IMO), International Telecommunication Union (ITU), and International Electrotechnical Commission (IEC). AIS is standardized by the ITU and adopted by the IMO. The IALA maintains and publishes technical guidelines for AIS manufacturers.

Prior to the standardization of AIS, there was no one system that provided dynamic exchange of identification and location information among maritime vessels.

Table 3: Automatic Identification System Signal Specifications [9]

Symbol	Parameter name	Low setting	High setting
PH.RFR	Regional frequencies (range of frequencies within RR Appendix 18)(1) (MHz)	156.025	162.025
PH.CHS	Channel spacing (encoded according to RR Appendix 18 with footnotes)(1) (kHz)	25	25
PH.AIS1	AIS 1 (default channel 1) (2087)(1) (see § 2.3.3) (MHz)	161.975	161.975
PH.AIS2	AIS 2 (default channel 2) (2088)(1) (see § 2.3.3) (MHz)	162.025	162.025
PH.BR	Bit rate (bit/s)	9 600	9 600
PH.TS	Training sequence (bits)	24	24
PH.TXBT	Transmit BT product	~0.4	~0.4
PH.RXBT	Receive BT product	~0.5	~0.5
PH.MI	Modulation index	~0.5	~0.5
PH.TXP	Transmit output power (W)	1	12.5

(1) See Recommendation ITU-R M.1084, Annex 4.

According to the ITU AIS Spec 1371-4 (Table 3), AIS is ran on either of the default frequencies 161.975 MHz or 162.025 MHz [9]. To eliminate any conflict with allocated frequencies in other countries, AIS can also be configured to run on frequencies between 156.025 MHz to 162.025 MHz (VHF band) with 12.5 kHz or 25 kHz channel spacing [9]. There are two standard power levels of transmission, 2 watts and 12.5 watts, but the power level can vary between 1.0 watt and 12.5 watts [9]. The lower power level is used to prevent receiver saturation, i.e. when the vessels are docked at a port. The higher power level is generally used while the vessels are sea-bound.

AIS uses a variety of timing methods to ensure the digital message is transmitted and received properly. AIS uses Self Organizing Time Division Multiple Access (SOTDMA) to properly transmit the digital messages without worrying about collisions

from other transmitters. AIS utilizes 2250 slots per 60 seconds per channel, and each slot has the capability of holding 256 bits (Figure 6). Every transmitted message has a slot number in the Communication State field (Figure 7). By extracting the slot number from the message, the transponder is able to synchronize with the SOTDMA scheme. Following the SOTDMA scheme prevents any collisions or corruption from multiple transmitters.

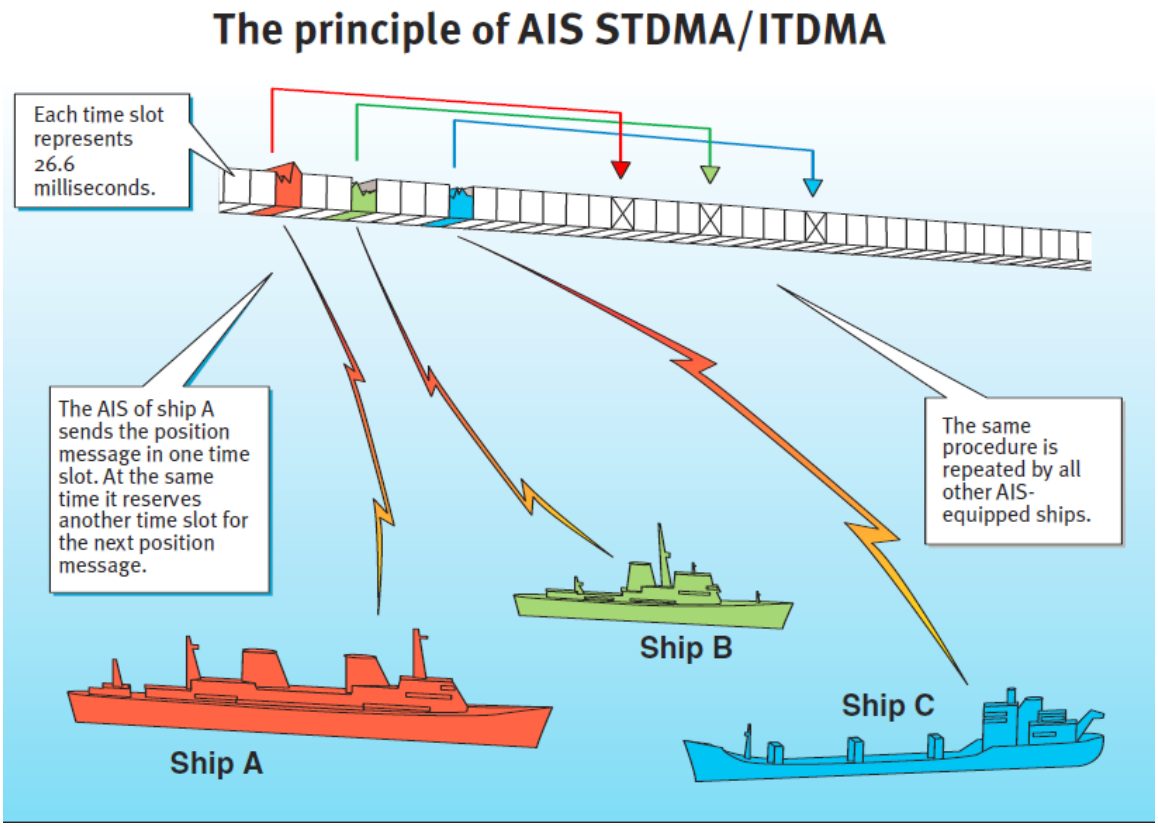


Figure 6: Self-Organized Time Division Multiple Access [10]

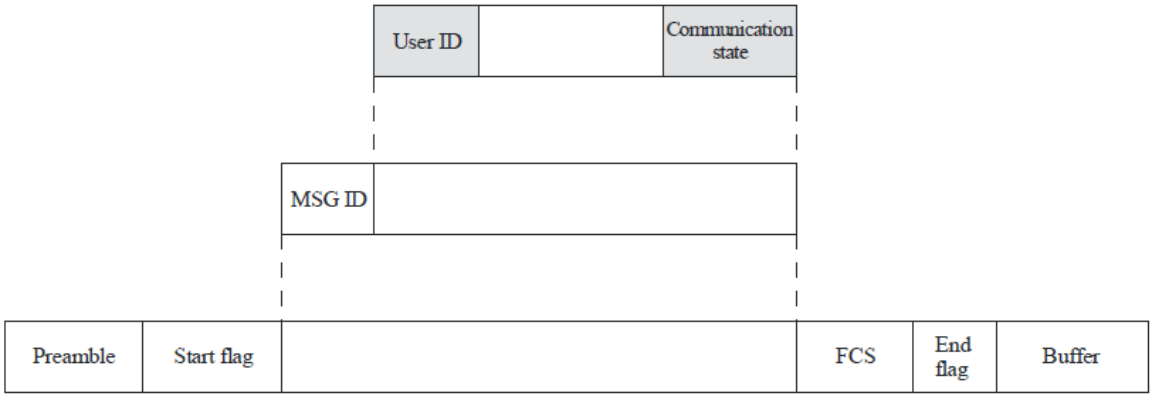


Figure 7: High-Level Data Link Control (HDLC) Packet Structure [9]

In order to accurately keep track of the maritime vessels' position and heading, the transmission of its position must be related to the vessel's velocity (Table 4). Each AIS transponder monitors the velocity and determines the interval when the messages are being sent out. As the velocity of the transponder increases, the frequency of messages increases and vice-versa when the velocity decreases. This method of transmission minimizes error in determining the projected heading of the vessel.

Table 4: Automatic Identification System Reporting Interval [9]

Ship's dynamic conditions	Nominal reporting interval
Ship at anchor or moored and not moving faster than 3 knots	3 min(1)
Ship at anchor or moored and moving faster than 3 knots	10 s(1)
Ship 0-14 knots	10 s(1)
Ship 0-14 knots and changing course	3 1/3 s(1)
Ship 14-23 knots	6 s(1)
Ship 14-23 knots and changing course	2 s
Ship >23 knots	2 s
Ship >23 knots and changing course	2 s

Digital modulation is a key technique used in AIS to transmit and receive digital messages. AIS uses Gaussian Minimum Shift Keying digital modulation to encode bits with a data rate of 9600 bps, and a time bandwidth product of approximately 0.5 [9]. On

every bit change, there is a phase change of $\pm\frac{\pi}{2}$ (Figure 8). The modulation index of the AIS signal is 0.5, thus the frequency changes ±2400 Hz for every bit change. Bits are determined by monitoring the frequency offset of the carrier frequency.

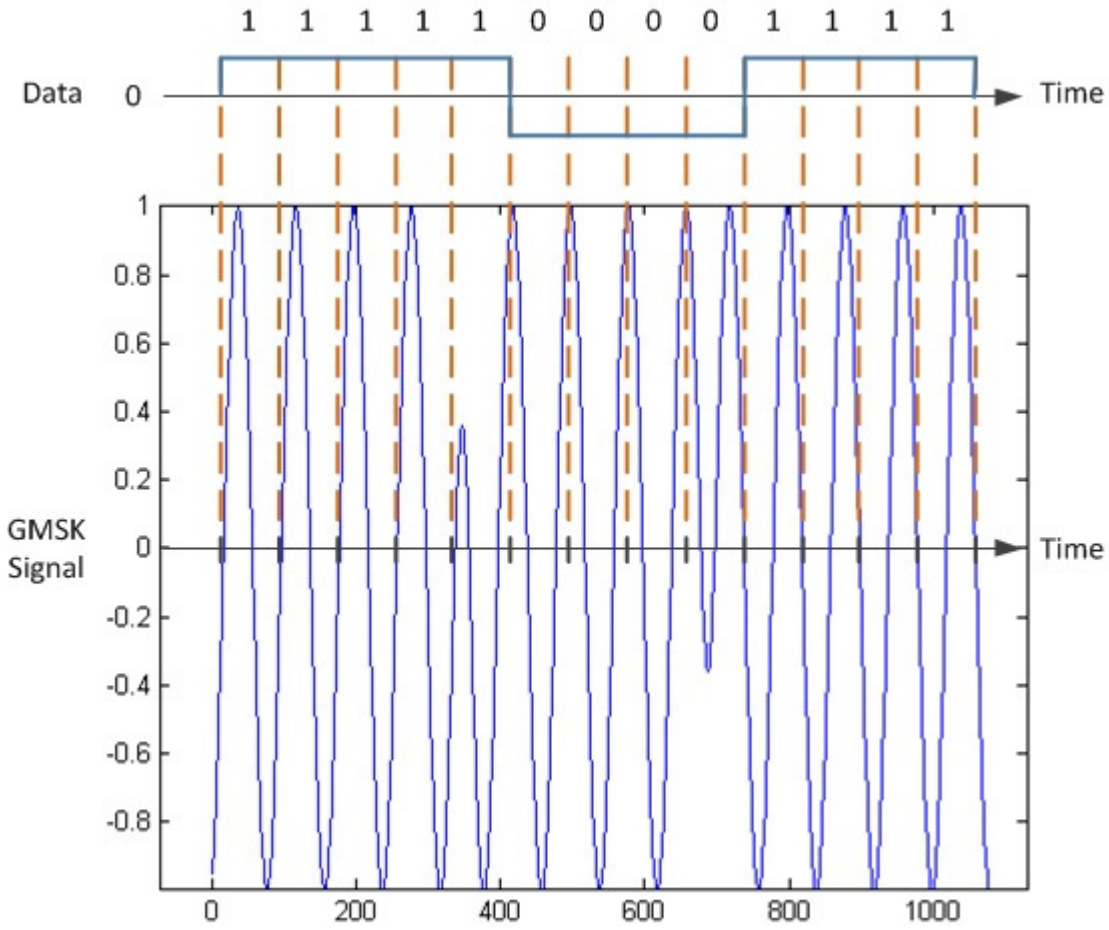


Figure 8: Gaussian Minimum Shift Keying Example

After the bits are extracted, the logic level of each bit is still unknown. The specification requires the data to use Non-Return-Zero-Inverted (NRZI) encoding to eliminate any high side or low side injection mirroring problems. NRZI encoded data is represented by the state of the current bit and the previous bit. If the current bit and

previous bit differ in state, the actual bit value is low; otherwise, the actual bit value is high.

After the logical bits are decoded, the bits are in compliance with the High-Level Data Link Control (HDLC) protocol (Figure 9).

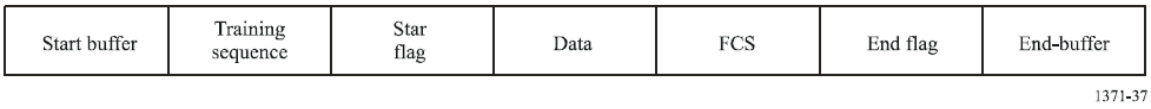


Figure 9: Automatic Identification System Packet Structure [11]

Table 5 shows the bit sizes for each field of the HDLC packet. Knowing the data rate for the AIS signal is 9600 b/s, the timing for each field can be calculated. The start buffer takes approximately 0.833 ms to complete before the start of the training sequence. The Training Sequence is comprised of 24 bits of alternating zeros and ones [9]. The training sequence last approximately 2.5 ms. Afterwards, the Start Flag of "01111110" starts and it lasts approximately 0.833 ms to complete [9]. The data field consists of 168 bits on a default transmission packet, and it takes approximately 17.5 ms to complete [9]. The Frame Check Sequence (FCS) field consists of a 16-bit polynomial cyclic redundancy check (CRC) of the bits inside the Data field, and it takes approximately 1.6 ms to complete [9]. The End Flag has the same bit sequence and timing of the Start Flag [11]. The End Buffer is approximately 24 bits and takes 2.5 ms complete [9]. A correlator is used to determine when the Training Sequence, Start Flag, and End Flag operations are completed thus locating the Data and FCS fields.

Table 5: Automatic Identification System Field Bit Sizes [9]

Ramp up	8 bits
Training sequence	24 bits
Start flag	8 bits
Data	168 bits
CRC	16 bits
End flag	8 bits
Buffer	24 bits
Total	256 bits

Once the Data and FCS fields are located via correlation, both fields must be "bit unstuffed". The HDLC protocol specification requires the fields between the Start and End Flag to be "bit-stuffed" to prevent any confusion of the Start and End Flag fields [11]. For the encoding portion, for every five consecutive logic ones, there is a logic zero inserted into the data field. For the decoding portion, for every five consecutive logic ones, the padded logic zero immediately afterwards must be discarded [11]. After the data is "bit unstuffed", the CRC is calculated and compared to the received CRC in the FCS field [11]. If the calculated CRC and received CRC do not match, the data is ignored; otherwise, the demodulation is a success and is parsed further.

At this point, the data is valid and is parsed into 1 of 27 different AIS messages, or is NMEA 0183 encoded using AIVDM/AIVDO two-layer protocol for a 3rd party external application to decode [9]. The NMEA 0183 message format relies on the data to go through a 6-bit to 8-bit American Standard Code for Information Interchange (ASCII) conversion [9]. Once it has undergone the ASCII conversion, it must be formatted to the

AIVDM/AIVDO protocol standard [12]. Afterwards, the data is sent out through a serial interface.

CHAPTER 3

III. IMPLEMENTATION OF THE AIS RECEIVER

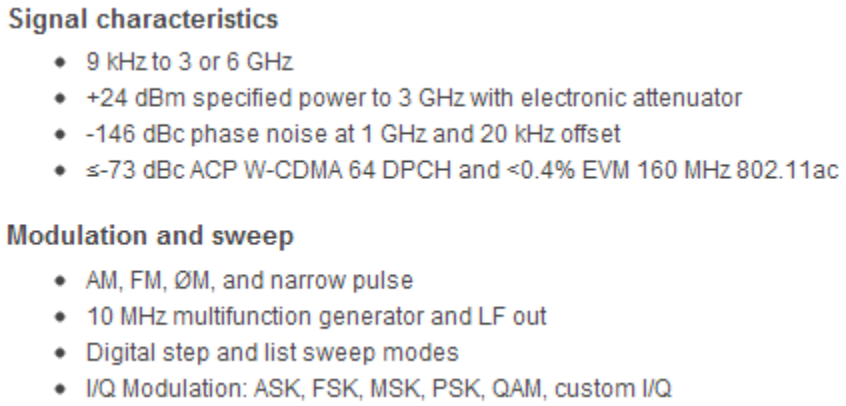
The implementation of the AIS receiver consist of examining the top level design and the details of each component of the complete system. Since digitally down converting signals to baseband is a trivial process, the MATLAB® and VHDL algorithms analyze the signal at baseband. Both MATLAB® and VHDL will use a common stimulus file generated by MATLAB® to exercise each system.

1. System Design

In order to develop a receiver, a complete system design must be developed. The system design for a receiver must include a signal source, radio frequency (RF) front-end, and a digital board. The signal source generates the signal of interest. The RF front-end comprise analog components to condition the signal for digital sampling. The digital board has the necessary components to digitally sample the signal of interest, conduct digital signal processing, and reformat the information to be sent out through a standard communication interface. All of these components must be configured correctly for the receiver to process the signal correctly.

A signal source is used to produce the signal with or without various modulation schemes. The AIS signal is a GMSK modulated signal and is transmitted at 161.975 MHz [9], and therefore a signal source capable of generating a modulated signal at

161.975MHz is needed. With these requirements in mind, the Agilent N5182B is selected to be the signal source for this project. The Agilent N5182B has the capability to generate an unmodulated and modulated signal from 9 kHz to 6 GHz. The N5182B can also be used as an Arbitary Waveform Generator for customizing waveforms. It can also generate a continuous sine wave at and around the AIS frequencies and generate the AIS signal. Its' specifications are listed in Figure 10.



Signal characteristics

- 9 kHz to 3 or 6 GHz
- +24 dBm specified power to 3 GHz with electronic attenuator
- -146 dBc phase noise at 1 GHz and 20 kHz offset
- ≤-73 dBc ACP W-CDMA 64 DPCH and <0.4% EVM 160 MHz 802.11ac

Modulation and sweep

- AM, FM, ØM, and narrow pulse
- 10 MHz multifunction generator and LF out
- Digital step and list sweep modes
- I/Q Modulation: ASK, FSK, MSK, PSK, QAM, custom I/Q

Figure 10: Agilent N5182B Vector Signal Generator Specification [13]

The components for the RF front-end are now considered. For this application, only an anti-aliasing analog filter is needed to prevent the signal of interest from mixing with signals from other Nyquist zones. An anti-aliasing analog filter is a band-pass filter around the signal of interest. It is important that the pass-band bandwidth of the band-pass filter covers the modulated signal's frequency range and does not cross into other Nyquist zones of the Analog to Digital Converter (ADC). Anatech Electronics's AE164B6933 band-pass filter is selected for the task. The 3 dB bandwidth of the pass-band region is approximately 10 MHz from the center frequency of 164 MHz, and the 40 dB bandwidth of the pass-band is approximately 30 MHz as shown in Figure 11. The

filter has a narrow pass-band region and the signal is within the pass-band region and performs within specifications of the AIS signal.

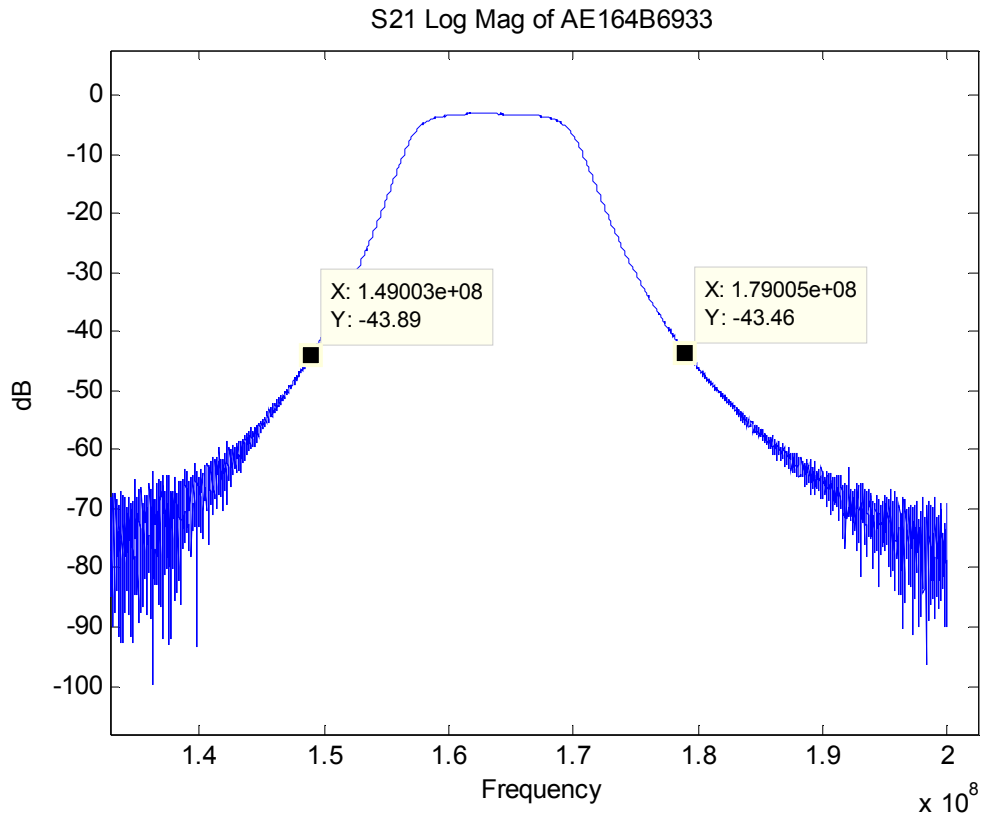


Figure 11: AE164B6933 Frequency Response Plot

For the digital board component, Altera's Stratix II Professional DSP Development Board is chosen. The digital board has a Stratix II FPGA (EP2S180F1020C3) with sufficient in-chip resources, shown in Figure 12, to perform complex Digital Signal Processing (DSP) algorithms [14]. The digital board also has a 12-bit ADC (AD9433) with a maximum sample rate of 125 MSPS and an analog bandwidth of 750 MHz [14]. Additional specifications of the AD9433 are listed in Figure 13. The sampling rate of the ADC is determined by the 40 dB bandwidth of the anti-aliasing filter. The 40 dB lower cut off frequency for the filter is at 149 MHz, and

the upper cut off frequency is at 179 MHz as shown in Figure 11. Both lower and upper cut off frequencies are within a Nyquist zone. It is determined that sampling at 120 MSPS provides three advantages.

1) Adjusting the sampling rate from 120 MSPS to 48 kSPS does not require any upsampling and only requires downsampling
2) As shown in Figure 29, the lower and upper cut off frequency of the anti-aliasing filter fits perfectly into the 3rd Nyquist zone
3) As shown in Figure 29, the signal is in the 3rd Nyquist zone, spectrum inversion is not an issue.

Feature	EP2S15	EP2S30	EP2S60	EP2S90	EP2S130	EP2S180
ALMs	6,240	13,552	24,176	36,384	53,016	71,760
Adaptive look-up tables (ALUTs) *(1)*	12,480	27,104	48,352	72,768	106,032	143,520
Equivalent LEs *(2)*	15,600	33,880	60,440	90,960	132,540	179,400
M512 RAM blocks	104	202	329	488	699	930
M4K RAM blocks	78	144	255	408	609	768
M-RAM blocks	0	1	2	4	6	9
Total RAM bits	419,328	1,369,728	2,544,192	4,520,488	6,747,840	9,383,040
DSP blocks	12	16	36	48	63	96
18-bit × 18-bit multipliers *(3)*	48	64	144	192	252	384
Enhanced PLLs	2	2	4	4	4	4
Fast PLLs	4	4	8	8	8	8
Maximum user I/O pins	366	500	718	902	1,126	1,170

Figure 12: Altera Stratix II FPGA Resource Specification [15]

IF sampling up to 350 MHz
SNR: 67.5 dB, f_{IN} up to Nyquist at 105 MSPS
SFDR: 83 dBc, f_{IN} = 70 MHz at 105 MSPS
SFDR: 72 dBc, f_{IN} = 150 MHz at 105 MSPS
2 V p-p analog input range
On-chip clock duty cycle stabilization
On-chip reference and track-and-hold
SFDR optimization circuit
Excellent linearity
DNL: ±0.25 LSB (typical)
INL: ±0.5 LSB (typical)
750 MHz full power analog bandwidth
Power dissipation: 1.35 W (typical) at 125 MSPS
Twos complement or offset binary data format
5.0 V analog supply operation
2.5 V to 3.3 V TTL/CMOS outputs

Figure 13: Analog Devices AD9433 Analog to Digital Converter Specification [16]

2. AIS Message Structure

A complete AIS system is able to process 27 different messages [9]. For this effort, Message 1, 2, and 3 are considered because the messages follow the same message structure and most used messages. In order to parse other types of messages, parsing components must be designed into the system. Message 1, 2, and 3 contain Latitude and Longitude information for the vessel, and these fields vary to exercise the design (Table 6).

Table 6: Automatic Identification System Message 1, 2, and 3 Structure [9]

Parameter	Number of bits	Description
Message ID	6	Identifier for this Message 1, 2 or 3
Repeat indicator	2	Used by the repeater to indicate how many times a message has been repeated. See § 4.6.1, Annex 2; 0-3; 0 = default; 3 = do not repeat any more
User ID	30	Unique identifier such as MMSI number

Navigational status	4	0 = under way using engine, 1 = at anchor, 2 = not under command, 3 = restricted manoeuvrability, 4 = constrained by her draught, 5 = moored, 6 = aground, 7 = engaged in fishing, 8 = under way sailing, 9 = reserved for future amendment of navigational status for ships carrying DG, HS, or MP, or IMO hazard or pollutant category C, high speed craft (HSC), 10 = reserved for future amendment of navigational status for ships carrying dangerous goods (DG), harmful substances (HS) or marine pollutants (MP), or IMO hazard or pollutant category A, wing in grand (WIG); 11-13 = reserved for future use, 14 = AIS-SART (active), 15 = not defined = default (also used by AIS-SART under test)
Rate of turn ROT_{AIS}	8	0 to +126 = turning right at up to 708° per min or higher 0 to –126 = turning left at up to 708° per min or higher Values between 0 and 708° per min coded by $ROT_{AIS} = 4.733\ SQRT(ROT_{sensor})$ degrees per min where ROT_{sensor} is the Rate of Turn as input by an external Rate of Turn Indicator (TI). ROT_{AIS} is rounded to the nearest integer value. +127 = turning right at more than 5º per 30 s (No TI available) –127 = turning left at more than 5º per 30 s (No TI available) –128 (80 hex) indicates no turn information available (default). ROT data should not be derived from COG information.
SOG	10	Speed over ground in 1/10 knot steps (0-102.2 knots) 1 023 = not available, 1 022 = 102.2 knots or higher
Position accuracy	1	The position accuracy (PA) flag should be determined in accordance with Table 47 1 = high (≤ 10 m) 0 = low (>10 m) 0 = default
Longitude	28	Longitude in 1/10 000 min (±180º, East = positive (as per 2's complement), West = negative (as per 2's complement). 181= ($6791AC0_h$) = not available = default)

Latitude	27	Latitude in 1/10 000 min (±90°, North = positive (as per 2's complement), South = negative (as per 2's complement). 91° (3412140h) = not available = default)
COG	12	Course over ground in 1/10 = (0-3599). 3600 (E10h) = not available = default. 3 601-4 095 should not be used
True heading	9	Degrees (0-359) (511 indicates not available = default)
Time stamp	6	UTC second when the report was generated by the electronic position system (EPFS) (0-59, or 60 if time stamp is not available, which should also be the default value, or 61 if positioning system is in manual input mode, or 62 if electronic position fixing system operates in estimated (dead reckoning) mode, or 63 if the positioning system is inoperative)
special manoeuvre indicator	2	0 = not available = default 1 = not engaged in special manoeuvre 2 = engaged in special manoeuvre (i.e.: regional passing arrangement on Inland Waterway)
Spare	3	Not used. Should be set to zero. Reserved for future use.
RAIM-flag	1	Receiver autonomous integrity monitoring (RAIM) flag of electronic position fixing device; 0 = RAIM not in use = default; 1 = RAIM in use. See Table 47
Communication State	19	See Table 46
Number of bits	168	

3. MATLAB® Algorithm

MATLAB® is used to model the digital receiver design. The MATLAB® model does not include the digital down conversion algorithm to convert the RF signal to an IF

signal. However, the quantization effects of the ADC is included in the MATLAB® model. Once the proof of concept is achieved in MATLAB® simulations, the MATLAB® data is Fixed Point to closely represent the data running on hardware and then hardware algorithm is developed in VHDL.

3.1. Generating a GMSK Baseband Signal

MATLAB® provides a Communications System Toolbox to generate a GMSK baseband signal from discrete values as show in the MATLAB® code in Figure 14. After the HDLC packet is formed from each data point, each HDLC packet is sent through the GMSK modulator, converted into I/Q data at baseband, oversampled by 5, and Fixed Point to a 1.17 number (18-bit total word size with 1 sign bit and 17 fractional bits). The settings for the GMSK modulator is determined by the AIS specification of the transmitter (Table 3).

```
hMod = comm.GMSKModulator('BitInput', true, 'InitialPhaseOffset', 0, ...
                          'BandwidthTimeProduct', 0.4, 'PulseLength', ...
                          4, 'SymbolPrehistory', 1, 'SamplesPerSymbol', ...
                          5, 'OutputDataType', 'double');
modSignal = step(hMod, data);
modSignal = fi(modSignal, 1, 18);
```

Figure 14: MATLAB® Code to Generate Fixed-Point GMSK Signal

The oversampled Fixed Point I/Q data is the stimulus for the MATLAB® algorithm and VHDL functional simulation.

3.2. Demodulation of GMSK Baseband Signal

As discussed in Chapter 2, the AIS signal is GMSK modulated and in order to detect a bit change, the instantaneous frequency must be examined after the carrier frequency is digitally down converted to baseband. Once the signal is digitally down

converted to baseband, the differential phase must be calculated from the signal by implementing the following equation,

$$e^{j(2*\pi*f*nT+\theta(n))} * e^{-j(2*\pi*f*nT+\theta(n-N))} = e^{j(\theta(n)-\theta(n-N))} \quad (9)$$

Even though the differential phase was calculated, the amplitude component still exists in the signal. In MATLAB®, the "angle" function is used to extract the phase component from the signal leaving the instantaneous frequency,

$$\omega = \frac{d\theta}{dt} \quad (10)$$

Figure 15 shows MATLAB® code to calculate the differential phase, extract the phase component, and translate the value to Fixed Point to match the hardware values

```
omega_diff = one_ais(1:end - 5).* conj(one_ais(6:end));
inst freq = fi(angle(omega diff), 1, 20);
```

Figure 15: MATLAB® Code to Calculate Instantaneous Frequency

Once the instantaneous frequency is extracted from the signal, conversion to logic is performed (Figure 16). The amplitude of the signal represents the frequency offset of the signal. If the amplitude of the signal is above the zero threshold, the logic value is either a logical 1 or 0, defined by the user (Figure 17). For example, if the signal is 1.2, it is a logical value of 1, and if the signal is -0.4, it is a logical value of 0. The logical values are swapped because the values are Non-Return-to-Zero-Inverted (NRZI) encoded; NRZI encoding is discussed in the next section.

```
for j=1:length(data)
    if data(j,1) >= 0
        bin_one_sig(j,1) = 1;
    else
        bin_one_sig(j,1) = 0;
    end;
end
```

Figure 16: Converting to Logical Form

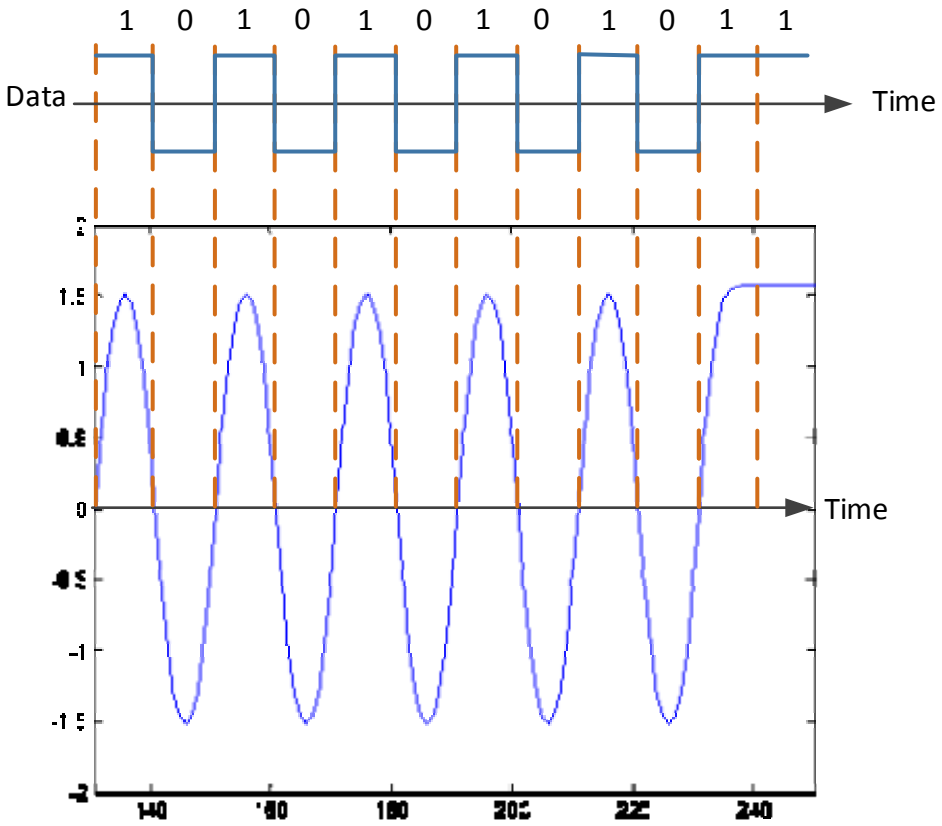


Figure 17: Instantaneous Frequency Over Time

3.3. Decoding AIS Messages

Once the signal is converted to logical 1's and 0's, synchronization must be done on the training sequence for proper align of the HDLC packet. According to the AIS

specification, the training sequence post NRZI decoding is 24 alternative 1's and 0's [9]. At this point, the data is still oversampled by five times the data rate of the AIS signal and still has the NRZI encoding in place. A correlator or a matched filter is used to compare the incoming data with the known values to determine the exact point to synchronize the HDLC packet (Figure 18). Since the data is still NRZI encoded, the correlator or matched filter compares against the known value and its inverse. The maximum resulting value of both comparisons must be greater than 60% of the length of the known value. The 60% threshold value is varied for optimal detection rate. The result is used to indicate the start of the data, but the training sequence is kept for further validation after the data is processed further.

```
sfd_logic_abs1 = abs(xcorr2(double(bin_one_sig), double(logic_sfd1)));
sfd_logic_abs2 = abs(xcorr2(double(bin_one_sig), double(logic_sfd2)));
if max(sfd_logic_abs1) > max(sfd_logic_abs2)
  sfd_logic_corr = find(length(logic_sfd1) * .60 < sfd_logic_abs1);
else
  sfd_logic_corr = find(length(logic_sfd1) * .60 < sfd_logic_abs2);
end
```

Figure 18: MATLAB® Code for Correlation to Locate Training Sequence and Start Data Flag

After the start of training sequence is found, the oversampled data must be downsampled by 5 to match the data rate of the AIS data (Figure 19). In order to sample in the middle of the eye, shown in Figure 20, only the third sample is kept.

```
bin_dn = downsample(bin_trim(3:end), 5)
```

Figure 19: MATLAB® Code to Downsampling to Data Rate

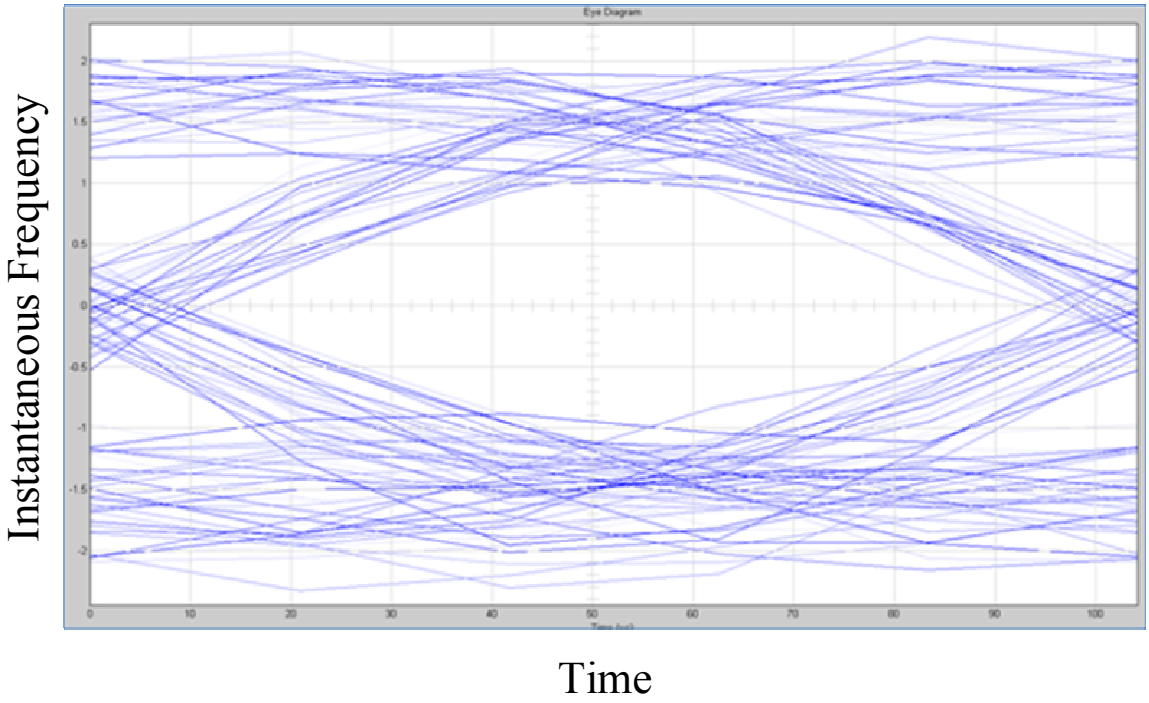


Figure 20: Eye Diagram of Instantaneous Frequency

Once the data is downsampled to the data rate of the AIS data, NRZI decoding must be conducted on the data. Figure 21 shows the incoming data being exclusive-nor with the delayed data. Figure 22 shows the block diagram of the NRZI decoding. The resulting data is the HDLC packet.

```
nrzi_decode = not(xor(bin_dn(1:end - 1), bin_dn(2:end)));
```

Figure 21: MATLAB® Code for Non-Return-to-Zero-Inverted Decoding

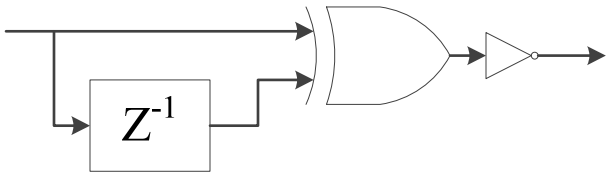


Figure 22: Block Diagram for Non-Return-to-Zero-Inverted Decoding

Once the HDLC packet is formatted, Start and End Flags must be located, and the Data and FCS fields must be "bit-unstuffed". The HDLC packet is ran through a correlator or a matched filter with the known Start and End Flag field values to determine the location of the Data and FCS fields (Figure 23).

```
sfd_logic_abs = abs(xcorr2(double(nrzi_decode), double(logic_sfd)));
efd_logic_abs = abs(xcorr2(double(nrzi_decode), double(logic_efd)));
sfd_logic_corr = find(max(sfd_logic_abs)-.5 < sfd_logic_abs);
efd_logic_corr = find(max(efd_logic_abs)-.5 < efd_logic_abs);
nrzi_decode_corr = nrzi_decode(sfd_logic_corr(1) + 1:...
    efd_logic_corr(efd_logic_corr > sfd_logic_corr(sig_num) + (168)) - ...
    length(logic_efd));
```

Figure 23: MATLAB® Code for Correlation to Determine Location of Training Sequence, Start and End Data Flags

Once the Data and FCS fields are found, both fields must be "bit-unstuffed". A correlator or a matched filter is used to determine if there are five consecutive logical 1's and a 0 afterwards. If the correlator or matched filter determined an area with "bit-stuffing", the MATLAB® code in Figure 24 strips off the padded logic 0. Once the Data and FCS is "bit-unstuffed", the FCS of the data is calculated and verified.

```
xcorr_bits = xcorr2(double(nrzi_decode_corr), bit_stuff);
loc_unstuff = find(xcorr_bits == (length(bit_stuff) - 1));
for i= 1:(length(loc_unstuff) + 1)
    if i == 1
        bits_unstuff = nrzi_decode_corr(1:loc_unstuff(1) - 1);
    elseif i == (length(loc_unstuff) + 1)
        bits_unstuff = vertcat(bits_unstuff, nrzi_decode_corr(...
            loc_unstuff(i - 1)+1:length(nrzi_decode_corr)));
    else
        bits_unstuff = vertcat(bits_unstuff, nrzi_decode_corr(...
            loc_unstuff(i - 1)+1:loc_unstuff(i)-1));
    end
end
```

Figure 24: MATLAB® Code for Correlation to Determine Bit-Stuffing to Bit-Unstuff Data

The FCS is used to determine if the incoming data is valid. The incoming FCS is compared to the calculated FCS. If both FCS values are the identical, the data is valid, otherwise the data is invalid. The following equation is the 16-bit ITU-T CRC polynomial equation used for the FCS [9]. Figure 25 shows the MATLAB® implementation of the 16-bit ITU-T CRC polynomial equation. After the last value is processed, the result of the calculated FCS is inverted and compared to the incoming FCS.

$$G(x) = x^{16} + x^{12} + x^5 + 1 \quad (11)$$

```
r=ones(1,16);

for c3=1:length(msg)

    s1=bitxor(msg(c3),r(1));
    s2=bitxor(s1,r(12));
    s3=bitxor(s1,r(5));

    r=[r(2:16) s1];

    r(11)=s2;
    r(4)=s3;
end

msg_FCS=r;
```

Figure 25: MATLAB® Code for CRC Polynominal

The values from the data is extracted by using Table 6, and the data is now translated into NMEA 0813 standards. For this application, "!AIVDM,1,1,,1," remains static while the Data field is converted into ASCII characters by using the 6-bit to 8-bit conversion. First, each byte must be bit swapped, so the Most Significant Bit (MSB) of the byte is now the Least Significant Bit (LSB). Second, the bits are grouped into 6-bit words. Third, if the 6-bit words are over 39, 56 is added to the 6-bit word, otherwise 48 is added to the 6-bit word (Figure 26). The result is used with an ASCII table to determine the ASCII value. According to the NMEA 0813 specification, the checksum is calculated by exclusive OR of all the characters except for "$" or "!" and "*" [12]. The 8 LSB of the result is used for the checksum. The 4 MSB is the first character; the 4 LSB is the second character of the message. The format of the string is shown on Figure 27. For example, the resulting string is "!AIVDM,1,1,,A,10lviS003pIvc7nFg?aP0?wp0000,0*66"

```
byte_flip_ext=vertcat(byte_flip, zeros(8-mod(length(byte_flip),8),1));
for j=1:6:length(byte_flip)
  debug{counter, 1} = reshape(num2str(byte_flip(j:j+5)),1,6);
  debug{counter, 2} = bin2dec(reshape(num2str(byte_flip(j:j+5)),1,6));
  if debug{counter, 2} > 39
    debug{counter, 3} = char(debug{counter, 2} + 56);
  else
    debug{counter, 3} = char(debug{counter, 2} + 48);
  end
  counter = counter + 1;
end
```

Figure 26: MATLAB® Code to Convert 6-Bit ASCII to NMEA 0813 Format

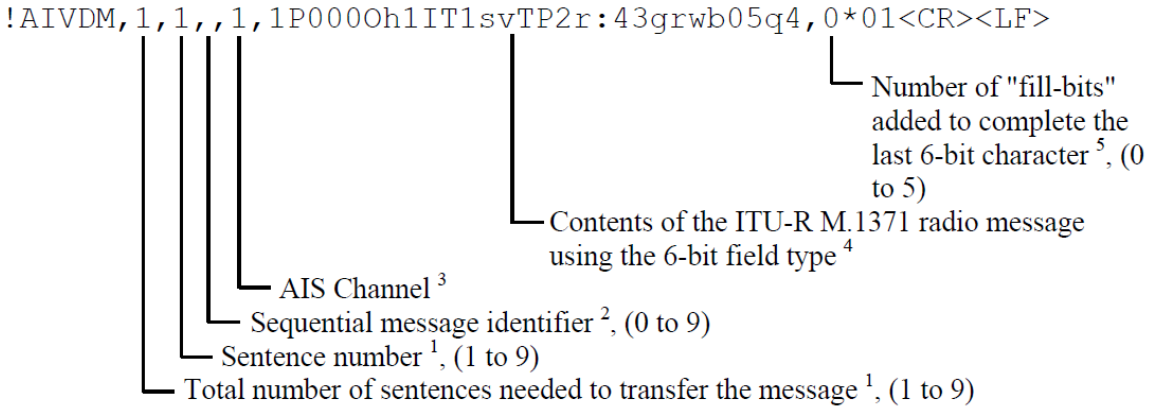


Figure 27: NMEA 0813 Automatic Identification System Message Format [12]

4. Hardware Design for FPGA

After the MATLAB® model is verified as correct, the hardware design is can be developed based upon the verified model. Since the hardware is sampling at RF, a digital down converter (DDC) algorithm is designed into the algorithm. The DDC takes the RF signal and mix the signal down to baseband into an I/Q channel. Afterwards, the DDC downsamples the signal to the appropriate data rate. The resulting signal goes through the demodulation and decoding algorithm described in the previous section. The VHDL code for the DDC, demodulation, and decoding of the AIS message is presented in Appendix A.

4.1. Digital Down Converter (DDC)

The digital board receives the signal at RF. The signal is processed through a band-pass filter, and then mixed with a 41.975 MHz (Figure 29) signal from the Local Oscillator (LO), and goes the low-pass filter. The band-pass filter isolates the signal at the frequency of interest. The mixer shifts the signal down to baseband and converts the real signal into a complex signal. The low-pass filter eliminates any unwanted frequencies caused by the shift in the frequency of interest down to baseband. The result is I/Q Data of the signal at baseband.

According to the AIS specification, the data rate of the AIS signal is 9.6 kSPS [9] and the digital board has sampled the signal at 120 MSPS, thus resulting in an oversampling by 12500. For demodulation of the signal, only an oversampling by the factor of 5 is required, and the signal is downsampled by 2500. The clock rate of the digital board is 120 MHz, and designing a narrow filter to downsample by 2500 would use much of the FPGA's resources and would cause problems when making internal routing of components. A multi-stage downsample is needed to properly downsample by 2500 without running into any FPGA resource or routing problems (Figure 28). The low-pass filter must have an appropriate effective bandwidth before downsampling (Equation 12); otherwise, signals from the second Nyquist zone is aliased over to the first Nyquist zone.

$$Effective\ Bandwidth \leq \frac{\pi}{2 * downsampling\ rate} \quad (12)$$

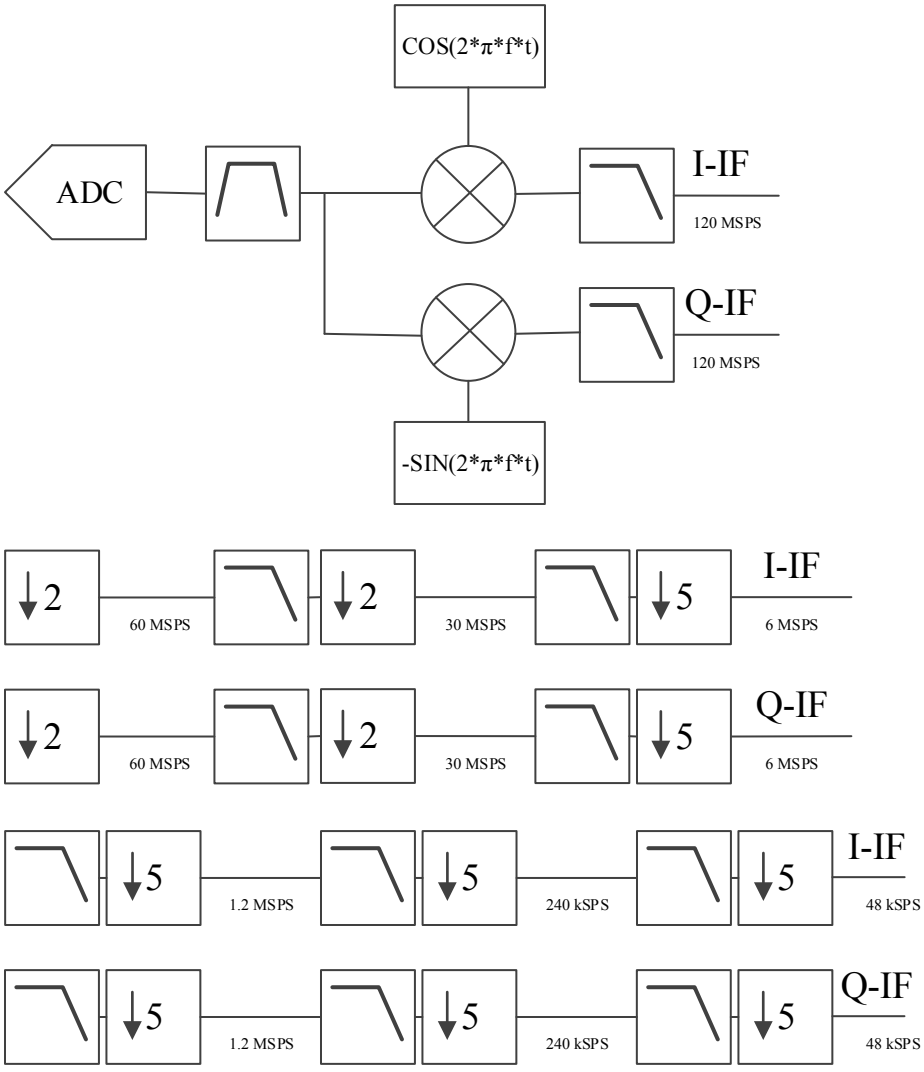


Figure 28: Digital Down Converter (DDC) w/ Downsampler Structure for Automatic Identification System [17]

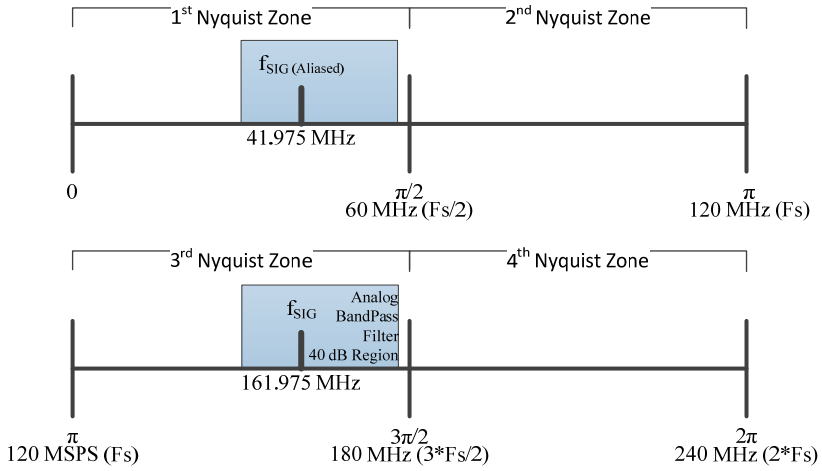


Figure 29: Frequency Spectrum of the System

The band-pass filter is used to eliminate any unwanted signals except for the signal of interest. MATLAB® is used to generate the coefficients for the Kaiser Window band-pass filter. The Kaiser Window filter is used for its flat response and low attenuation in the pass-band region. The frequency pass region is approximately 10% from the carrier frequency, and the frequency stop region is approximately 20% from the carrier frequency. Since the ADC has 12-bits of resolution, the attenuation at the frequency stop region is greater than or equal to 72 dB. With the selected parameters, MATLAB® generated a 128-order filter with 129 coefficients (Figure 30). The 129 coefficients are Fixed Point to an 18-bit word. Limiting the word size reduces the performance of the filter but it is neglectable due to having sufficient bits for this application. In the FPGA, filters are commonly developed by using the Finite Impulse Response (FIR) architecture (Equation 13 and Figure 31).

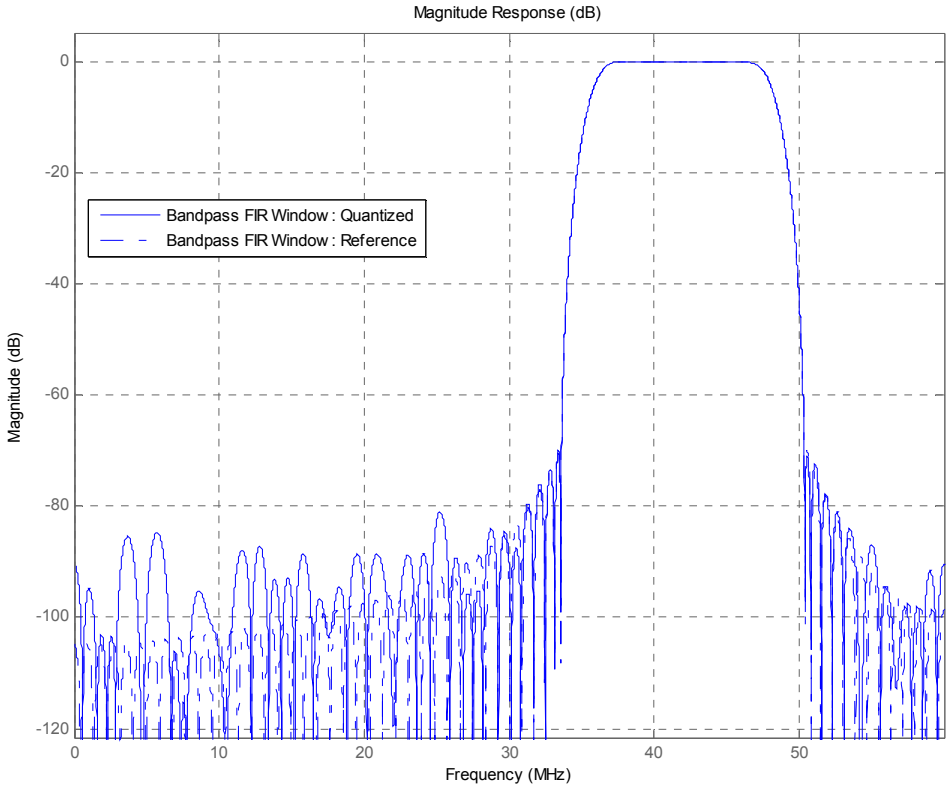


Figure 30: Band-Pass Filter Frequency Response Plot

$$y(k) = \sum_{n=0}^{N-1} a(n)x(k-n)\,, k = 0, 1, 2, 3, \ldots \quad (13)$$

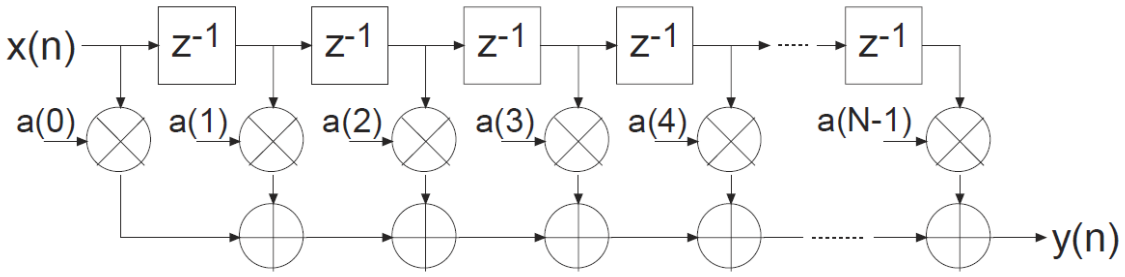


Figure 31: Finite Impulse Response (FIR) Filter Structure [18]

A low-pass filter is used for every downsample stage and right before the data is downsampled. The frequency response plot of the low-pass filter for the downsample by

2 is shown on Figure 26. The frequency response plot of the low-pass filter for the downsample by 5 is shown on Figure 33. Both filters' parameters were normalized when creating the coefficients. The low-pass filter for the downsample by 2 has a pass-band bandwidth of .3 and 72 dB of attenuation in the stopband. The low-pass filter for the downsample by 5 has a pass-band bandwidth of .1 and 72 dB of attenuation in the stopband.

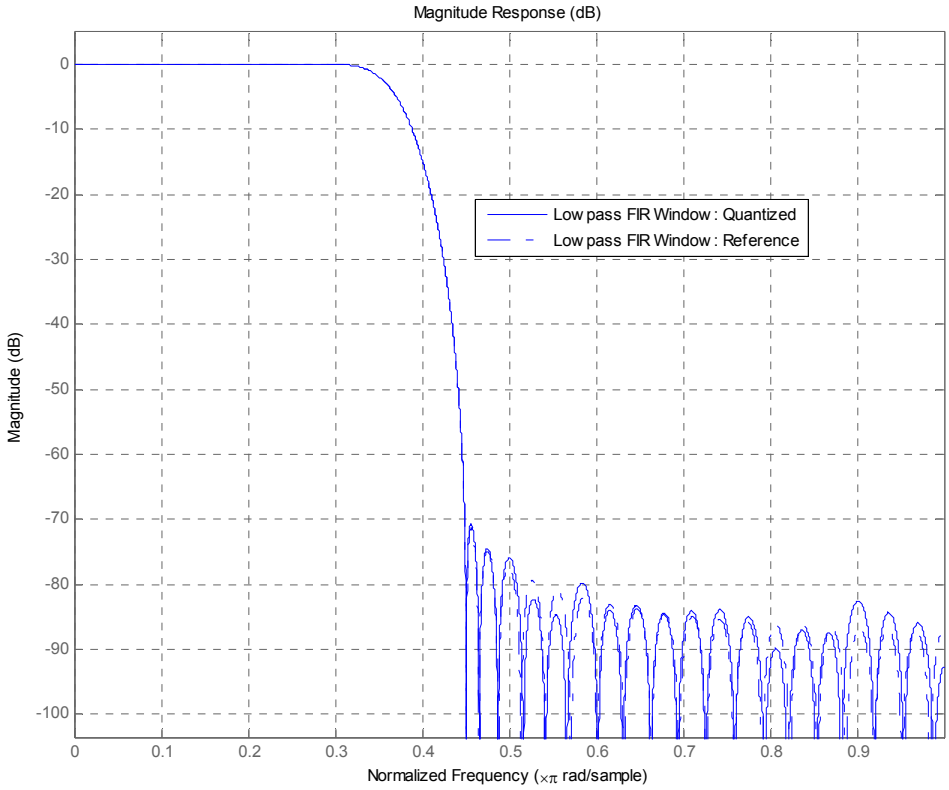


Figure 32: Frequency Response Plot for Downsample by 2 Low-Pass Filter

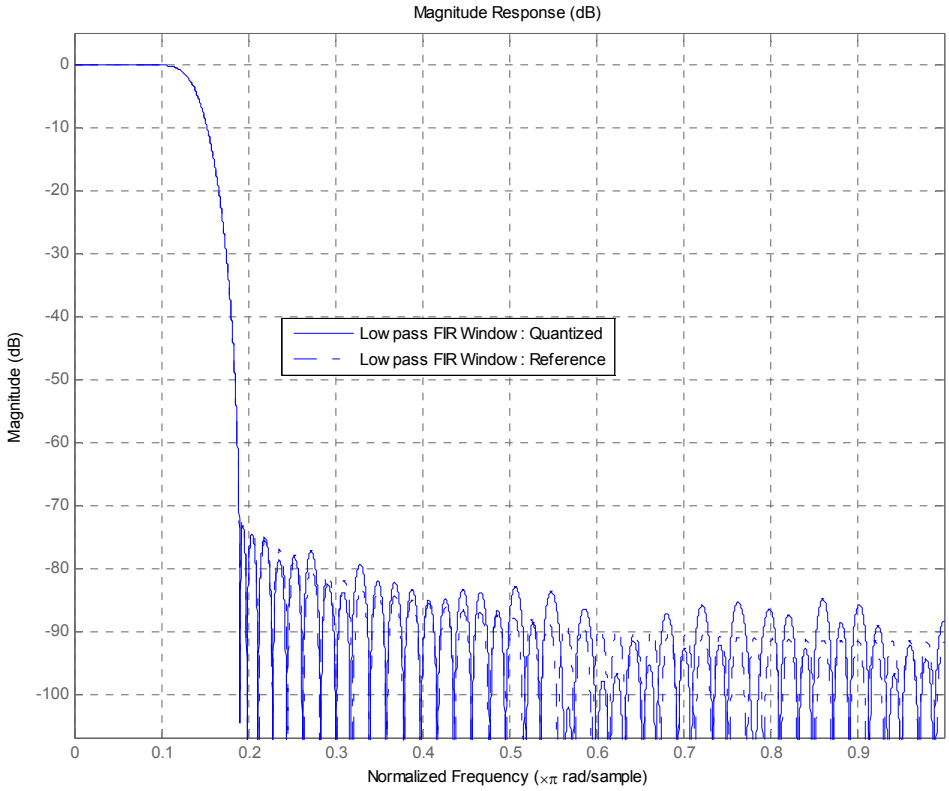


Figure 33: Frequency Response Plot for Downsample by 5 Low-Pass Filter

4.2. Demodulation of GMSK Baseband Signal

Once the RF signal is down converted to baseband and downsampled to the appropriate data rate, the signal is ready to be demodulated. Similar to the MATLAB® design, the differential phase must be extracted by taking the conjugate of the delayed incoming signal and performing a complex multiply with the incoming signal (Figure 34). Afterwards, a COordinate Rotation DIgital Computer (CORDIC) is used to extract the differential phase or the instantaneous frequency from the resulting signal. In hardware, designing a complex multiply is trivial process. The CORDIC design was obtained from opencores.org and is used to extract the phase component from the signal.

The phase component extracted from the signal is the instantaneous frequency of the signal.

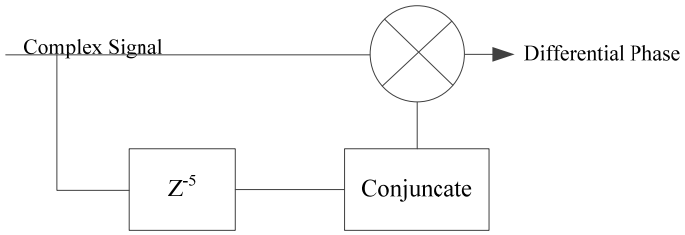


Figure 34: Delay Line Correlator

At this point, the instantaneous frequency is extracted and is ready to be converted to logical data. The signal is still oversampled by five and must be downsampled down to the data rate of the AIS signal. Once the signal is downsampled to the appropriate data rate, it is ready to be converted to 1's or 0's. Similar to the MATLAB® code, when the amplitude of the signal is above zero, the logic value is either a logical 1 or 0. If the amplitude falls below zero, it is the inverse of the value when it is above zero. At this point the quantized signal is now converted to logical bits.

4.3. Decoding AIS Messages

Once the AIS signal is converted to logical form, it is further processed to extract the message. Unlike the MATLAB® model, the hardware model has already downsampled the signal down to AIS's data rate. Before the AIS messages is extracted from the data, the data has to be conditioned. First, the data must be NRZI decoded. Second, the Training, Start, and End Flags must be located. Third, the Data and FCS fields must be "bit-unstuffed". Fourth, the FCS is calculated from the Data and compared with the given FCS. Fifth, if the FCS passes the comparison test, the data is extracted by

using Table 4, and the data is converted to NMEA 0813 standards to be compatible with 3rd party products.

The data is decoded using a process similar to the MATLAB® algorithm. The NRZI decoding is accomplished by exclusive OR the current value with the previous value. After the data is decoded, the data follows the HDLC packet specifications. Each field in the HDLC packet must be identified, before "bit-unstuff" is conducted to the data and FCS fields. In order to identify each field, the Training Sequence, Start and End Flags must be identified. The field sizes are known and a shift register, Figure 35, and a comparator is used to mark when the fields have arrived.

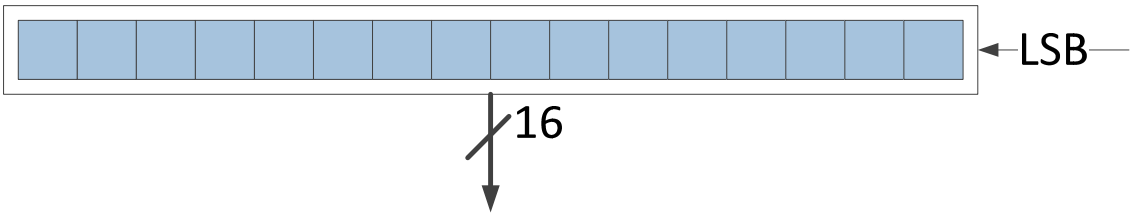


Figure 35: Serial to Parallel Shift Register

A state machine is used to mark each field. After the fields are identified, the Data and FCS fields must be "bit-unstuffed". The design detects five consecutive logical 1's and a 0 afterwards on the incoming data. If the above is detected, the design eliminates the padded logical 0 value. Afterwards, the FCS is calculated with the incoming data and compared with the incoming FCS. Afterwards, the data for the message is extracted and converted to the NMEA 0813 structure (Figure 27).

CHAPTER 4

IV. RESULTS

1. MATLAB® and VHDL Results

AIS is essentially a vessel tracking system. In order to exercising the MATLAB® and VHDL algorithms requires the AIS signal to be generated with various latitude and longitude points and passed through each algorithm. Figure 36 shows how the testing process for the receiver. Appendix B.1 and Appendix B.2 have the NMEA 0813 results from the MATLAB® and VHDL models. Appendix C.1 is the stimulant of latitude and longitude coordinates used to exercise the MATLAB® and VHDL models. Appendix C.2 is the resulting latitude and longitude coordinates from the MATLAB® and VHDL models. Since the results from the MATLAB® and VHDL models are identical, only one list was documented.

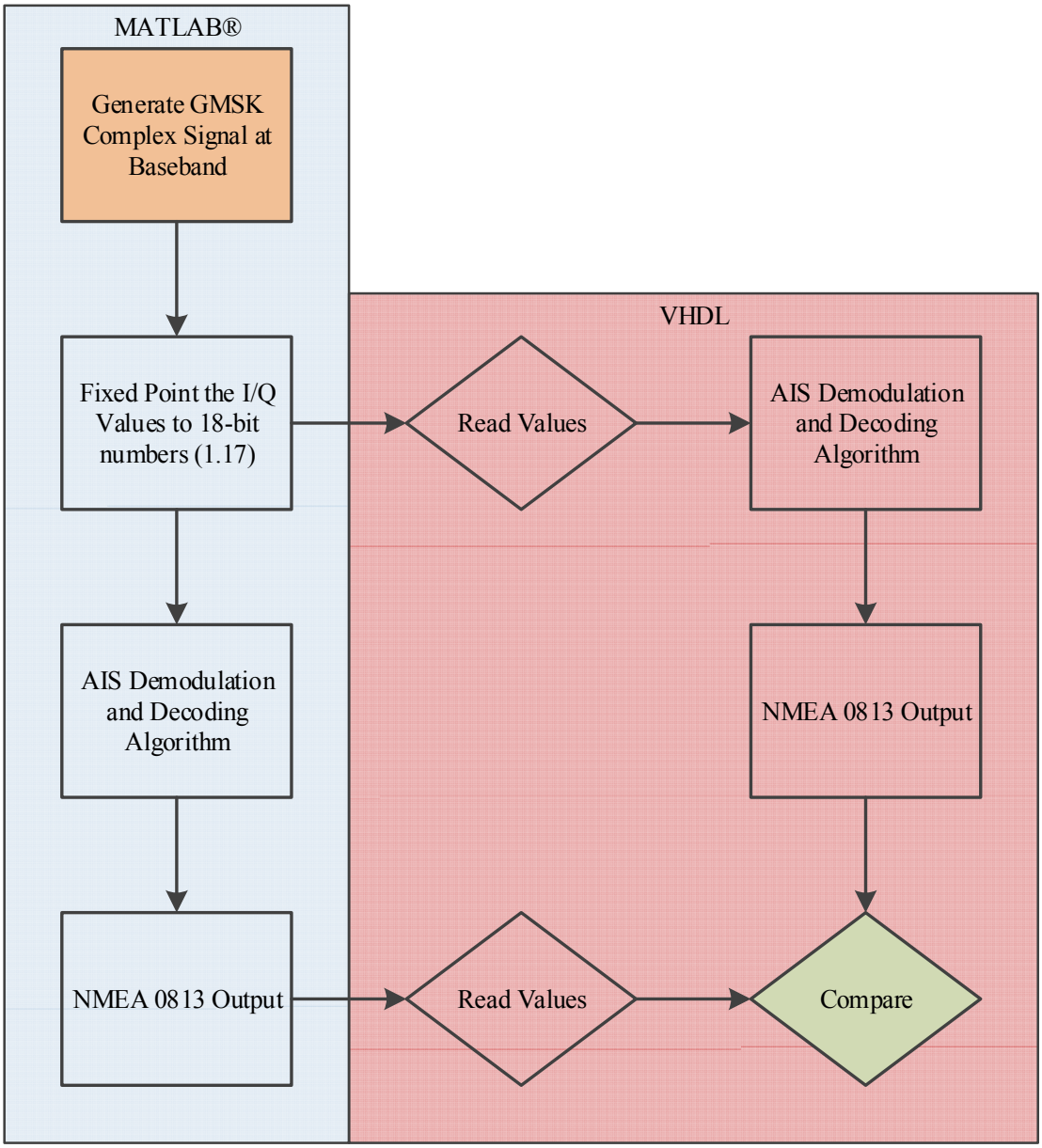


Figure 36: Testing Process for MATLAB® and VHDL Algorithms

The testing process will test both the MATLAB® and VHDL algorithms and compare the results from the stimulus of the system. 131 points around University of Dayton (UD) are selected and the latitude and longitude extracted. 131 AIS packets are created from the extracted latitude and longitude, and the signal is generated (Figure 37). The generation of the AIS signal is described in Chapter 3, Section 3.1. The AIS signal is over 16 dB SNR and is quantized to mimic real world conditions. Afterwards, the signal is fed into the MATLAB® and VHDL algorithms. The coordinates from the MATLAB® results is compared to the coordinates from the source (Figure 37). The same comparison is done with the coordinates from the VHDL results. As expected, the resulting coordinates (Figure 38) match up with the original coordinates (Figure 37) that were used to create the signal.

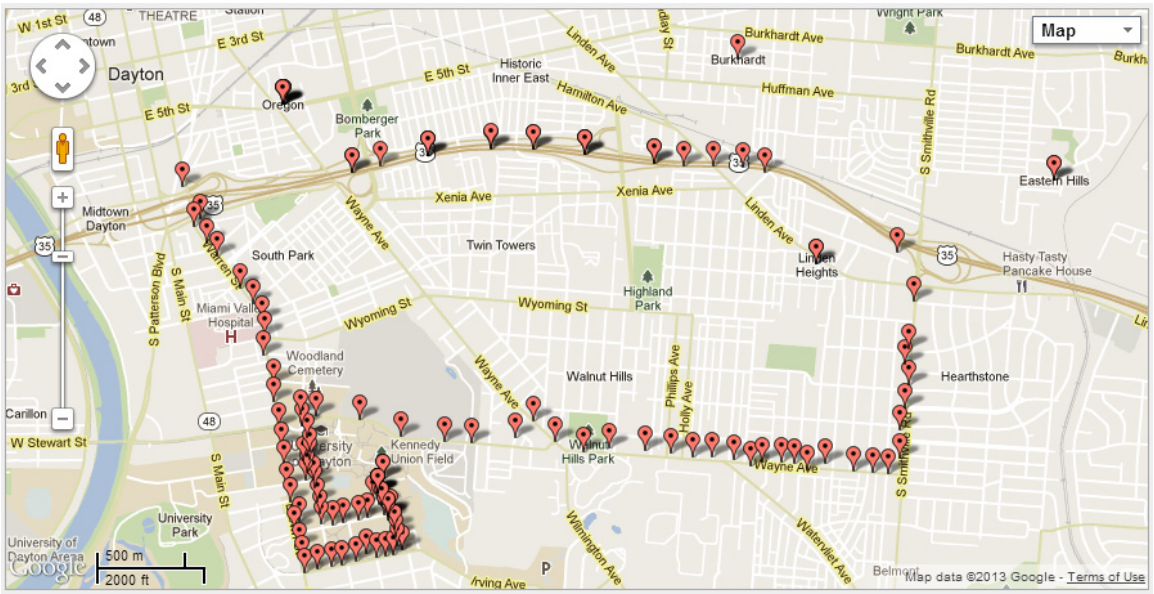


Figure 37: Stimulus Coordinates

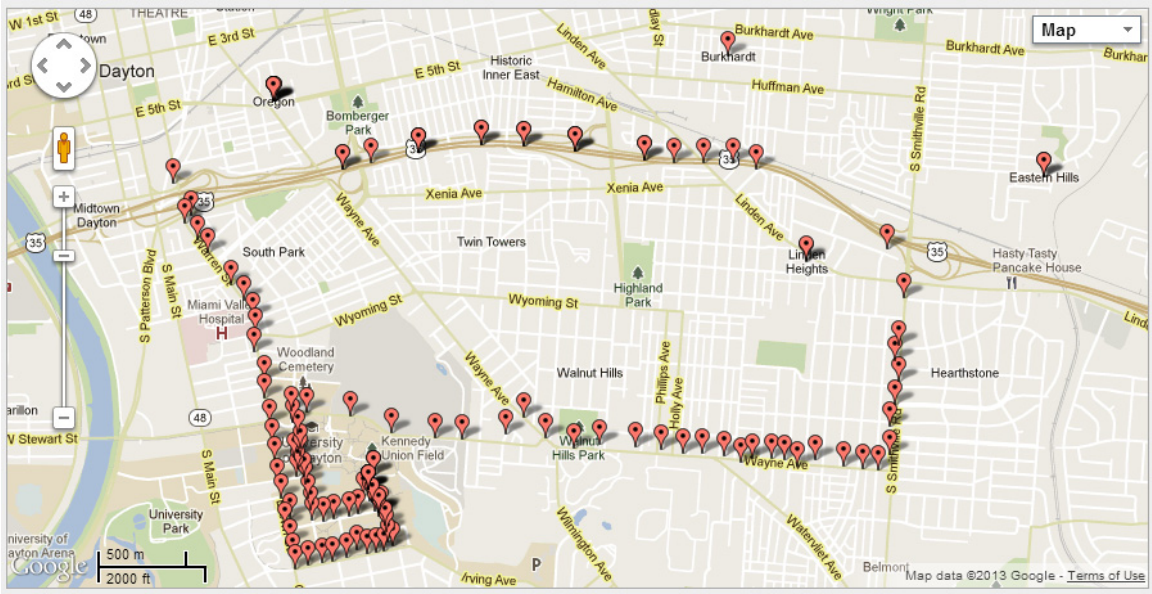


Figure 38: MATLAB® and VHDL Resulting Coordinates

2. Bit Error Rate Results

The E_b/N_0 is calculated by varying the Additive White Gaussian Noise (AWGN) over a certain number of points. The architecture is tested against 1003537 random sample points from E_b/N_0 values between 0 dB and 16 dB. As shown on Table 7, the Bit Error Rate (BER) for the receiver drops to zero at 16 dB E_b/N_0 for 1003537 bits. In contrast, the theoretical calculations (Table 2) shows there is 0 bit errors for every 1003537 bits at theoretical E_b/N_0 of 16 dB. Whenever there was a bit error in the AIS message, the calculated FCS would not match the received FCS and the system would flag the message as invalid and would not process the message any further.

Table 7: Practical Probability of Bit Error of GMSK Signal

Probability of Bit Error (1003537 Bits Tested)		
E_b/N_0 (dB)	VHDL BER	MATLAB® BER
0	0.406058	0.404548
1	0.359582	0.357041
2	0.307392	0.304441
3	0.252864	0.249366
4	0.200052	0.196880
5	0.150587	0.147444
6	0.105459	0.103097
7	0.067408	0.065797
8	0.038468	0.037629
9	0.018845	0.018301
10	0.007643	0.007438
11	0.002555	0.002609
12	0.000660	0.000680
13	0.000102	0.000130
14	0.000014	0.000022
15	0.000000	0.000002
16	0.000000	0.000000

CHAPTER 5

V. CONCLUSION

The goal of this project is to consolidate most of the RF analog processing and data decoding into a single FPGA to reduce SWaP requirements. The tuning and downsampling processes were moved from the analog circuit into a digital circuit implemented on a FPGA. In addition, the decoding process was also implemented on the FPGA thus consolidating the entire digital processing into a single integrated circuit chip.

Prior to implementation of the model in VHDL, the complete design is verified by generating signals from MATLAB® Digital Communication toolbox. The packets were formed in MATLAB® with 131 different latitude and longitude coordinates and were modulated into an analog representation of 131 different GMSK modulated signals. Once the signals have been created, the signals were used as a stimulus to test the MATLAB® and VHDL models. The stimulus signals served as inputs to the MATLAB® and VHDL models . The goal is achieved when both the MATLAB® and VHDL models decoded these signals and those decoded results matched the messages that generated the stimulus signals. Once white noise was generated and added to the input signal, a BER test was conducted on both MATLAB® and VHDL models and the BER table was compared to the theoretical BER table. The BER test revealed the MATLAB® and VHDL models were equivalent to the theoretical model.

If more time and resources were available, there is much to do for future expansion of this project. AIS is a two channel system and adding support to handle the second channel will improve its capability to receive additional messages. In addition of adding capability of processing the second channel, the current receiver design handles messages commonly sent from other maritime vessels and adding support for the remaining 24 messages will add capability to receive messages from non-maritime vessels. Another thing to explore is to integrate and test the DDC. The DDC will complete the entire system by being able to accept raw RF signals instead of just IF signals. Finally, developing a BER chart for the entire system will test the AIS receiver's sensitivity.

REFERENCES

[1] M. Marchese, M. Ruggieri and I. Bisio, "FM Discriminator for AIS Satellite Detection," in *Personal Satellite Services*, Springer Berlin Heidelberg, 2010, pp. 19-34.

[2] M. G. Souissi, K. Grati, A. Ghazel and A. Kouki, "Software Efficient Implementation of GMSK Modem for an Automatic Identification System Transceiver," in *Canadian Conference on Electrical and Computer Engineering*, Niagara Falls, 2008.

[3] L. Gao and J. Liu, "Design of Dual-Channel AIS Digital Receiver," in *International Conference on Instrumentation & Measurement, Computer, Communication and Control*, Heilongjiang, 2012.

[4] G. Liersch and C. Dick, "Reconfigurable Gate Array Architectures for Real Time Digital Signal Processing," *Signals, Systems and Computers,* vol. 2, pp. 1383-1387, 31 October 1994.

[5] ELECTUNATED, "Android FSK," [Online]. Available: http://electunated.wordpress.com/category/android-fsk/. [Accessed 15 May 2013].

[6] B. Sklar, Digital Communications Fundamentals and Applications, 2nd ed., R. Kernan, Ed., Upper Saddle River, New Jersey: Bernard Goodwin, 2003.

[7] M. K. Simon and M. S. Alouini, Digital Communication over Fading Channels, 3rd ed., J. G. Proakis, Ed., New York, New York: John Wiley & Sons, Inc., 2000.

[8] K. Murota and K. Hirade, "GMSK Modulation for Digital Mobile Radio Telephony," *IEEE Transactions on Communications,* Vols. COM-29, no. 7, pp. 1044-1050, July 1981.

[9] International Telecommunications Union, Recommendation ITU-R M.1371-4: Technical Characteristics for an Automatic Identification System Using Time-Division Multiple Access in the VHF Maritime Mobile Band, Geneva: International Telecommunications Union, 2010.

[10] Swedish Maritime Administration, "Automatic Identification System," Swedish Maritime Administration, Feburary 2004. [Online]. Available: http://www.sjofartsverket.se/upload/1486/a171_2.pdf. [Accessed 15 May 2013].

[11] International Organization for Standardization, ISO/IEC 13239: Information Technology - Telecommunications and Information Exchange Between Systems - High-level Data Link Control (HDLC) Procedures, 3rd ed., Geneva: International Organization for Standardization, 2002.

[12] National Marine Electronics Association, NMEA 0183: Standard For Interfacing Marine Electronic Devices, 3.01 ed., Severna Park, Maryland: National Marine Electronics Association, 2002.

[13] Agilent Technologies, "N5182B MXG X-Series RF Vector Signal Generator," [Online]. Available: http://www.home.agilent.com/en/pd-2115999-pn-N5182B/mxg-x-series-rf-vector-signal-generator. [Accessed 15 May 2013].

[14] Altera, "Stratix II EP2S180: Reference Manual," Altera, August 2005. [Online]. Available: http://www.altera.com/literature/manual/mnl_stx2_pro_dsp_dev_kit_ep2s180.pdf. [Accessed 15 May 2013].

[15] Altera, "Stratix II Device Handbook," [Online]. Available: http://www.altera.com/literature/hb/stx2/stx2_sii51001.pdf. [Accessed 15 May 2013].

[16] Analog Devices, "12-Bit, 105 MSPS/125 MSPS, AD9433," Analog Devices, [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/AD9433.pdf. [Accessed 15 May 2013].

[17] F. J. Harris, Multirate Signal Processing for Communication Systems, 1st ed., Upper Saddle River, New Jersey: Prentice Hall, 2004.

[18] Xilinx, "Distributed Arithmetic FIR Filter v9.0," 28 April 2005. [Online]. Available: http://www.xilinx.com/ipcenter/catalog/logicore/docs/da_fir.pdf. [Accessed 15 May 2013].

[19] D. Devi and A. Sharma, "BER Performance of GMSK Using MATLAB," *International Journal of Advanced Research in Computer Engineering & Technology,* vol. 2, no. 4, pp. 1389-1392, April 2013.

[20] P. J. Ashenden, The Designer's Guide to VHDL (Systems on Silicon), 3rd ed., Burlington, Massachusetts: Morgan Kaufmann, 2010.

[21] Texas Instruments, "Digital Frequency Modulation - GMSK," [Online]. Available: www.ti.com/ww/cn/uprogram/share/ppt/c5000/21modulation_v110.ppt. [Accessed

15 May 2013].

[22] Marinetraffic.com, "DIY VHF AIS Receiver," Marinetraffic.com, 2012. [Online]. Available: http://www.marinetraffic.com/ais/downloads/DIY_VHF_Radio_setup_guide.pdf. [Accessed 15 May 2013].

[23] W. Changrui, K. Chao, X. Shigen and C. Huizhi, "Design and FPGA Implementation of Flexible and Efficiency Digital Down Converter," in *Signal Processing (ICSP)*, Beijing, 2010.

[24] Y. Wang, T. Zhang and J. Zhang, "A High Efficiency DDC Algorithm for Narrow Band Signal," in *E-Product E-Service and E-Entertainment*, Henan, 2010.

APPENDIX A

Appendix A.1: Delay and Conjugate Block

```
if (i_data_rdy = '1') then
  f_data_i      <= i_data_i;
  f_data_q      <= i_data_q;
  ff_data_i     <= f_data_i;
  ff_data_q     <= f_data_q;
  fff_data_i    <= ff_data_i;
  fff_data_q    <= ff_data_q;
  f4_data_i     <= fff_data_i;
  f4_data_q     <= fff_data_q;
  f5_data_i     <= f4_data_i;
  f5_data_q_n   <= not(f4_data_q) + 1;
  f6_data_i     <= f5_data_i;
  f6_data_q_n   <= f5_data_q_n;
  f_data_rdy    <= '1';
else
  f_data_rdy    <= '0';
end if;
```

Appendix A.2: Complex Multiply

```
complex_multiply: process(i_clk)
begin
  if rising_edge(i_clk) then
    f_x1          <= f_data_i * f6_data_i;
    f_x2          <= f_data_q * f6_data_q_n;
    f_y1          <= f_data_q * f6_data_i;
    f_y2          <= f_data_i * f6_data_q_n;
    f_mix_data_i  <= sxt(f_x1, 37) - f_x2;
    f_mix_data_q  <= sxt(f_y1, 37) + f_y2;
  end if;
end process;
```

Appendix A.3: NRZI Decoding

```
if (f_logic_conv_rdy = '1') then
  f_nrzi_decode     <= not(f_logic_conv xor ff_logic_conv);
  f_nrzi_decode_rdy <= '1';
else
  f_nrzi_decode_rdy <= '0';
end if;
```

Appendix A.4: Downsample by 5

```
oversampled: process(i_clk)
begin
  if rising_edge(i_clk) then
    if (i_inst_freq_rdy = '1') then
      if (f_oversampled_counter /= k_oversampled - 1) then
        f_oversampled_counter <= f_oversampled_counter + 1;
      else
        f_oversampled_counter <= "000";
      end if;
    end if;
  end if;
end process;

dn_sampled: process(i_clk)
begin
  if rising_edge(i_clk) then
    if (f_oversampled_counter = 2 and i_inst_freq_rdy = '1') then
      f_dn_sampled     <= i_inst_freq;     .
      f_dn_sampled_rdy <= '1';
    else
      f_dn_sampled_rdy <= '0';
    end if;
  end if;
end process;
```

Appendix A.5: Logic Conversion

```
logic_conv: process(i_clk)
begin
  if rising_edge(i_clk) then
    if (f_dn_sampled_rdy = '1' and f_dn_sampled > 0) then
      f_logic_conv     <= '1';
      ff_logic_conv    <= f_logic_conv;
      f_logic_conv_rdy <= '1';
    elsif (f_dn_sampled_rdy = '1') then
      f_logic_conv     <= '0';
      ff_logic_conv    <= f_logic_conv;
      f_logic_conv_rdy <= '1';
    else
      f_logic_conv_rdy <= '0';
    end if;
  end if;
end process;
```

Appendix A.6: Serial to Parallel Shift Register

```
if (f_nrzi_decode_rdy = '1') then
  f_shft_reg_sfd <= f_shft_reg_sfd(f_shft_reg_sfd'length - 2 downto 0) & f_nrzi_decode;
  f_shft_reg_efd <= f_shft_reg_efd(f_shft_reg_efd'length - 2 downto 0) & f_nrzi_decode;
end if;
```

Appendix A.7: State Machine to Identify Training Sequence, Start and End Flags

```
detect_fd: process(i_clk)
begin
  if rising_edge(i_clk) then
    case f_detect_fd is
      when s_det_sfd =>
        if (f_nrzi_decode_rdy = '1' and f_shft_reg_sfd = k_sfd) then
          f_detect_sfd <= '1';
          f_detect_fd <= s_det_efd;
        else
          f_detect_efd <= '0';
          f_detect_sfd <= '0';
          f_detect_fd  <= s_det_sfd;
        end if;
      when s_det_efd =>
        if (f_nrzi_decode_rdy = '1' and f_shft_reg_efd = k_efd) then
          f_detect_efd <= '1';
          f_detect_fd  <= s_det_sfd;
        else
          f_detect_efd <= '0';
          f_detect_sfd <= '0';
          f_detect_fd  <= s_det_efd;
        end if;
      when others =>
    end case;
  end if;
end process;
```

Appendix A.8: Bit-Unstuffing Detection and Filtering

```
bit_unstuff: process(i_clk)
begin
  if rising_edge(i_clk) then
    case f_bit_unstuff is
      when s_wait_sfd =>
        if (f_detect_sfd = '1') then
          if (ff_nrzi_decode = '1') then
            f_bs_counter  <= "001";
            f_bs          <= ff_nrzi_decode;
            f_bs_rdy      <= '1';
            f_bit_unstuff <= s_scan_data;
          else
            f_bs_counter <= "000";
            f_bs         <= ff_nrzi_decode;
            f_bs_rdy     <= '1';
            f_bit_unstuff <= s_scan_data;
          end if;
        end if;
      when s_scan_data =>
        if (f_nrzi_decode_rdy = '1' and f_shft_reg_efd = k_efd) then
          f_bs_counter  <= "000";
          f_bs_rdy      <= '0';
          f_bit_unstuff <= s_wait_sfd;
        else
          if (f_bs_counter = 5 and f_nrzi_decode = '0' and f_nrzi_decode_rdy = '1') then
            f_bs_counter <= "000";
          elsif (f_nrzi_decode = '1' and f_nrzi_decode_rdy = '1') then
            f_bs         <= f_nrzi_decode;
            f_bs_rdy     <= '1';
            f_bs_counter <= f_bs_counter + 1;
          elsif (f_nrzi_decode = '0' and f_nrzi_decode_rdy = '1') then
            f_bs         <= f_nrzi_decode;
            f_bs_rdy     <= '1';
            f_bs_counter <= "000";
          else
            f_bs_rdy <= '0';
          end if;
        end if;
      when others =>
    end case;
  end if;
end process;
```

Appendix A.9: CRC Polynomial

```
crc_i    <= crc_const when soc = '1' else
            crc_r;

crc_c(0) <= data XOR crc_i(15);
crc_c(1) <= crc_i(0);
crc_c(2) <= crc_i(1);
crc_c(3) <= crc_i(2);
crc_c(4) <= crc_i(3);
crc_c(5) <= data XOR crc_i(4) XOR crc_i(15);
crc_c(6) <= crc_i(5);
crc_c(7) <= crc_i(6);
crc_c(8) <= crc_i(7);
crc_c(9) <= crc_i(8);
crc_c(10) <= crc_i(9);
crc_c(11) <= crc_i(10);
crc_c(12) <= data XOR crc_i(11) XOR crc_i(15);
crc_c(13) <= crc_i(12);
crc_c(14) <= crc_i(13);
crc_c(15) <= crc_i(14);
```

Appendix A.10: Isolate Incoming FCS and Compare with Calculated FCS

```
crc_compare: process(i_clk)
begin
  if rising_edge(i_clk) then
    ff_detect_sfd <= f_detect_sfd;
    if (f_bs_rdy = '1' and f_data_counter > 167 and f_data_counter < 184) then
      f_rx_crc <= f_rx_crc(f_rx_crc'length - 2 downto 0) & f_bs;
    elsif (f_bs_rdy = '1' and f_data_counter = 184) then
      f_rx_crc <= not(f_rx_crc);
      f_rx_crc_rdy <= '1';
    else
      f_rx_crc_rdy <= '0';
    end if;

    if (f_nrzi_decode_rdy = '1' and f_data_counter = 167) then
      f_crc_eoc <= '1';
    else
      f_crc_eoc <= '0';
    end if;

    if (w_cal_crc_rdy = '1') then
      f_cal_crc     <= w_cal_crc;
      f_cal_crc_rdy <= '1';
    else
      f_cal_crc_rdy <= '0';
    end if;

    if (f_rx_crc_rdy = '1' and f_cal_crc = f_rx_crc) then
      f_valid_crc <= '1';
    else
      f_valid_crc <= '0';
    end if;

    if (f_valid_crc = '1') then
      ff_ascii_buffer <= f_ascii_buffer;
      f_rdy           <= '1';
    else
      f_rdy <= '0';
    end if;

  end if;
end process;
```

Appendix A.11: Convert 6-bit to 8-bit ASCII

```
six2eight_ascii_conv: process(i_clk)
begin
  if rising_edge(i_clk) then
    for k in 27 downto 0 loop
      if unsigned(w_ais_data_bs(k*6 + 5 downto k*6)) > 39 then
        f_ascii_buffer(41 - k) <= unsigned(w_ais_data_bs(k*6 + 5 downto k*6)) + unsigned(conv_std_logic_vector(56, 8));
      else
        f_ascii_buffer(41 - k) <= unsigned(w_ais_data_bs(k*6 + 5 downto k*6)) + unsigned(conv_std_logic_vector(48, 8));
      end if;
    end loop;
      if (f_cal_crc_rdy = '1') then
        f_checksum <= (others => '0');
        f_checksum_counter <= conv_std_logic_vector(1, 7);
      elsif (f_checksum_counter /= 44) then
        f_checksum <= ext(f_ascii_buffer(conv_integer(f_checksum_counter)), 16) xor (f_checksum);
        f_checksum_counter <= f_checksum_counter + 1;
      end if;

      if (f_rx_crc_rdy = '1') then
        if (unsigned(f_checksum(7 downto 4)) < 10) then
          f_ascii_buffer(45) <= ext(f_checksum(7 downto 4), 8) + 48;
        else
          f_ascii_buffer(45) <= ext(f_checksum(7 downto 4), 8) + 55;
        end if;

        if (unsigned(f_checksum(3 downto 0)) < 10) then
          f_ascii_buffer(46) <= ext(f_checksum(3 downto 0), 8) + 48;
        else
          f_ascii_buffer(46) <= ext(f_checksum(3 downto 0), 8) + 55;
        end if;
      end if;
  end if;
end process;
```

APPENDIX B

Appendix B.1: Results from MATLAB® Model

!AIVDM,1,1,,A,10lviS0080Ivc4lFg?Ih0?wp0000,0*3C
!AIVDM,1,1,,A,10lviS003PIvc9rFg?SP0?wp0000,0*66
!AIVDM,1,1,,A,10lviS0099Ivc>`Fg?Q00?wp0000,0*72
!AIVDM,1,1,,A,10lviS001uIvcCvFg?O00?wp0000,0*43
!AIVDM,1,1,,A,10lviS00:gIvcF@Fg?1@0?wp0000,0*67
!AIVDM,1,1,,A,10lviS009VIvcF>Fg>`P0?wp0000,0*6B
!AIVDM,1,1,,A,10lviS000rIvcEpFg=uP0?wp0000,0*1D
!AIVDM,1,1,,A,10lviS000rIvcENFg=I@0?wp0000,0*0F
!AIVDM,1,1,,A,10lviS002MIvcM>Fg=;@0?wp0000,0*38
!AIVDM,1,1,,A,10lviS000EIvcONFg<<P0?wp0000,0*56
!AIVDM,1,1,,A,10lviS006vIvcPjFg;I@0?wp0000,0*3A
!AIVDM,1,1,,A,10lviS00=DIvcSHFg:Ch0?wp0000,0*01
!AIVDM,1,1,,A,10lviS009pIvcU0Fg9IP0?wp0000,0*7E
!AIVDM,1,1,,A,10lviS008EIvcWNFg8?00?wp0000,0*21
!AIVDM,1,1,,A,10lviS00=lIvcaRFg7H00?wp0000,0*5F
!AIVDM,1,1,,A,10lviS001TIvcQnFg7A@0?wp0000,0*1E
!AIVDM,1,1,,A,10lviS00>QIvcGRFg72P0?wp0000,0*5D
!AIVDM,1,1,,A,10lviS001gIvc=JFg6n@0?wp0000,0*4B
!AIVDM,1,1,,A,10lviS008AIvc0nFg6U@0?wp0000,0*76
!AIVDM,1,1,,A,10lviS002CIvbkrFg6D00?wp0000,0*59
!AIVDM,1,1,,A,10lviS008uIvbQ@Fg5t@0?wp0000,0*2E
!AIVDM,1,1,,A,10lviS0005IvbE4Fg5fh0?wp0000,0*3C
!AIVDM,1,1,,A,10lviS00<AIvb3@Fg5GP0?wp0000,0*5F
!AIVDM,1,1,,A,10lviS00=UIvaj>Fg52h0?wp0000,0*23
!AIVDM,1,1,,A,10lviS00>bIvag>Fg6MP0?wp0000,0*5E
!AIVDM,1,1,,A,10lviS00?jIvachFg8800?wp0000,0*1E
!AIVDM,1,1,,A,10lviS0085Iva`8Fg9n00?wp0000,0*42
!AIVDM,1,1,,A,10lviS004FIvaU6Fg;F00?wp0000,0*2C
!AIVDM,1,1,,A,10lviS001`IvaPhFg=N00?wp0000,0*5A
!AIVDM,1,1,,A,10lviS0088IvaLvFg?Bh0?wp0000,0*5F
!AIVDM,1,1,,A,10lviS009HIvaHRFgB2h0?wp0000,0*03
!AIVDM,1,1,,A,10lviS00<=IvaEPFgD<@0?wp0000,0*5C
!AIVDM,1,1,,A,10lviS001EIvaAdFgFN00?wp0000,0*19
!AIVDM,1,1,,A,10lviS00:UIva<@FgIVh0?wp0000,0*14

!AIVDM,1,1,,A,10lviS008AIva9lFgKkh0?wp0000,0*14
!AIVDM,1,1,,A,10lviS002gIva4<FgNt@0?wp0000,0*57
!AIVDM,1,1,,A,10lviS00?1Iva0HFgQM00?wp0000,0*2A
!AIVDM,1,1,,A,10lviS009MIv`thFgSF@0?wp0000,0*4C
!AIVDM,1,1,,A,10lviS0073Iv`jfFgUIP0?wp0000,0*35
!AIVDM,1,1,,A,10lviS00?4Iv`S0FgW?00?wp0000,0*41
!AIVDM,1,1,,A,10lviS00:OIv`5@FgbtP0?wp0000,0*37
!AIVDM,1,1,,A,10lviS007?IvWpvFgd`@0?wp0000,0*0C
!AIVDM,1,1,,A,10lviS00=LIvWWnFgfr@0?wp0000,0*5A
!AIVDM,1,1,,A,10lviS008QIvWS@Fgi>P0?wp0000,0*3B
!AIVDM,1,1,,A,10lviS008oIvW`HFgif@0?wp0000,0*76
!AIVDM,1,1,,A,10lviS00:pIvWhfFgiw@0?wp0000,0*5C
!AIVDM,1,1,,A,10lviS005pIv`4nFgjF@0?wp0000,0*02
!AIVDM,1,1,,A,10lviS003mIv`R8FgjSP0?wp0000,0*2C
!AIVDM,1,1,,A,10lviS009AIvaFfFgk7h0?wp0000,0*1C
!AIVDM,1,1,,A,10lviS00=oIvan0Fgkg00?wp0000,0*40
!AIVDM,1,1,,A,10lviS006QIvb>@Fgl?P0?wp0000,0*69
!AIVDM,1,1,,A,10lviS001lIvbSvFglkP0?wp0000,0*5C
!AIVDM,1,1,,A,10lviS0077IvblHFgm<P0?wp0000,0*56
!AIVDM,1,1,,A,10lviS004lIvc=dFgmgh0?wp0000,0*11
!AIVDM,1,1,,A,10lviS006KIvcW2Fgn=@0?wp0000,0*79
!AIVDM,1,1,,A,10lviS00=EIvd3tFgnv@0?wp0000,0*12
!AIVDM,1,1,,A,10lviS006MIvdMpFgoK00?wp0000,0*27
!AIVDM,1,1,,A,10lviS006@Ive36Fgomh0?wp0000,0*6D
!AIVDM,1,1,,A,10lviS005iIve`NFgowh0?wp0000,0*76
!AIVDM,1,1,,A,10lviS002@Ivf1vFgovP0?wp0000,0*0B
!AIVDM,1,1,,A,10lviS004;IvfWnFgok00?wp0000,0*75
!AIVDM,1,1,,A,10lviS001IIvg0fFgoQh0?wp0000,0*0E
!AIVDM,1,1,,A,10lviS006pIvgEhFgo700?wp0000,0*75
!AIVDM,1,1,,A,10lviS0048IvggbFgnnh0?wp0000,0*17
!AIVDM,1,1,,A,10lviS004iIvhEDFgnHP0?wp0000,0*53
!AIVDM,1,1,,A,10lviS006kIvhnrFgn1h0?wp0000,0*0F
!AIVDM,1,1,,A,10lviS001rIvi>6Fgmr@0?wp0000,0*6C
!AIVDM,1,1,,A,10lviS007sIvigPFgn0@0?wp0000,0*15
!AIVDM,1,1,,A,10lviS00;CIvjE4Fgn0@0?wp0000,0*6C
!AIVDM,1,1,,A,10lviS003rIvjrlFgmQP0?wp0000,0*48
!AIVDM,1,1,,A,10lviS00<TIvkFtFglg@0?wp0000,0*6B
!AIVDM,1,1,,A,10lviS001<IvkfjFgkTh0?wp0000,0*2C
!AIVDM,1,1,,A,10lviS006CIvl?>FgiV00?wp0000,0*06
!AIVDM,1,1,,A,10lviS0004IvlfPFggL00?wp0000,0*54
!AIVDM,1,1,,A,10lviS003RIvm:rFgeM00?wp0000,0*4D
!AIVDM,1,1,,A,10lviS0002IvmPRFgd?h0?wp0000,0*4F
!AIVDM,1,1,,A,10lviS0032Ivmf0FgU`P0?wp0000,0*4E
!AIVDM,1,1,,A,10lviS002BIvmadFgP<@0?wp0000,0*25
!AIVDM,1,1,,A,10lviS004CIvm`HFgN3@0?wp0000,0*1E

!AIVDM,1,1,,A,10lviS002kIvmUnFgKSh0?wp0000,0*6E
!AIVDM,1,1,,A,10lviS002>IvmT:FgHwP0?wp0000,0*71
!AIVDM,1,1,,A,10lviS009UIvmQRFgF=00?wp0000,0*58
!AIVDM,1,1,,A,10lviS00>JIvmO0FgB>00?wp0000,0*3B
!AIVDM,1,1,,A,10lviS00?1Ivm=DFg@s00?wp0000,0*08
!AIVDM,1,1,,A,10lviS003SIvlqlFgA4P0?wp0000,0*25
!AIVDM,1,1,,A,10lviS007fIvlRTFgA@h0?wp0000,0*43
!AIVDM,1,1,,A,10lviS0061IvkwHFgASh0?wp0000,0*38
!AIVDM,1,1,,A,10lviS008HIvkd4FgAa00?wp0000,0*4A
!AIVDM,1,1,,A,10lviS004?IvkI@FgAn00?wp0000,0*67
!AIVDM,1,1,,A,10lviS0016Ivk5fFgAvP0?wp0000,0*49
!AIVDM,1,1,,A,10lviS006wIvjfrFgB6@0?wp0000,0*1A
!AIVDM,1,1,,A,10lviS002jIvjQ8FgB8h0?wp0000,0*58
!AIVDM,1,1,,A,10lviS000KIvj:BFgBK00?wp0000,0*41
!AIVDM,1,1,,A,10lviS00?AIviePFgBUh0?wp0000,0*4C
!AIVDM,1,1,,A,10lviS006qIviG0FgBfP0?wp0000,0*3C
!AIVDM,1,1,,A,10lviS00?HIvhu@FgBq00?wp0000,0*38
!AIVDM,1,1,,A,10lviS00<<IvhLvFgC5h0?wp0000,0*5D
!AIVDM,1,1,,A,10lviS0008Ivgi0FgCIP0?wp0000,0*7D
!AIVDM,1,1,,A,10lviS00:pIvg@RFgCVh0?wp0000,0*53
!AIVDM,1,1,,A,10lviS00;CIvfe0FgDgP0?wp0000,0*29
!AIVDM,1,1,,A,10lviS00:DIvf@tFgGC@0?wp0000,0*79
!AIVDM,1,1,,A,10lviS008nIverVFgEBh0?wp0000,0*69
!AIVDM,1,1,,A,10lviS003PIve4PFgD`P0?wp0000,0*07
!AIVDM,1,1,,A,10lviS00<GIvdQdFgDoP0?wp0000,0*40
!AIVDM,1,1,,A,10lviS003bIvccHFgEGP0?wp0000,0*5A
!AIVDM,1,1,,A,10lviS005tIvbobFgGL00?wp0000,0*04
!AIVDM,1,1,,A,10lviS00>@Ivb1:FgGo@0?wp0000,0*6E
!AIVDM,1,1,,A,10lviS00=eIvafbFgGuh0?wp0000,0*76
!AIVDM,1,1,,A,10lviS006LIvag<FgF`P0?wp0000,0*27
!AIVDM,1,1,,A,10lviS0056IvanlFgDrh0?wp0000,0*2F
!AIVDM,1,1,,A,10lviS009gIvaqBFgCQ00?wp0000,0*3F
!AIVDM,1,1,,A,10lviS00>TIvapTFgBTh0?wp0000,0*40
!AIVDM,1,1,,A,10lviS00>SIvahVFgBL00?wp0000,0*1D
!AIVDM,1,1,,A,10lviS009NIvajhFg@vP0?wp0000,0*63
!AIVDM,1,1,,A,10lviS005EIvalTFg@7P0?wp0000,0*1F
!AIVDM,1,1,,A,10lviS00=aIvauTFg@:P0?wp0000,0*27
!AIVDM,1,1,,A,10lviS0075IvawHFg?<h0?wp0000,0*26
!AIVDM,1,1,,A,10lviS00>NIvb2RFg=Qh0?wp0000,0*67
!AIVDM,1,1,,A,10lviS000RIvb5VFg<7h0?wp0000,0*11
!AIVDM,1,1,,A,10lviS008QIvb8DFg:s@0?wp0000,0*6F
!AIVDM,1,1,,A,10lviS00;MIvbFTFg:gh0?wp0000,0*22
!AIVDM,1,1,,A,10lviS002pIvbRTFg:u@0?wp0000,0*38
!AIVDM,1,1,,A,10lviS005IIvbmbFg;F@0?wp0000,0*3D
!AIVDM,1,1,,A,10lviS0031Ivc66Fg;`@0?wp0000,0*6B

!AIVDM,1,1,,A,10lviS005:IvcFLFg;vh0?wp0000,0*52
!AIVDM,1,1,,A,10lviS006NIvcO6Fg<800?wp0000,0*47
!AIVDM,1,1,,A,10lviS008jIvcMNFg=1P0?wp0000,0*7F
!AIVDM,1,1,,A,10lviS000jIvcBdFg=Rh0?wp0000,0*09
!AIVDM,1,1,,A,10lviS008nIvc9HFg>9h0?wp0000,0*3A
!AIVDM,1,1,,A,10lviS004JIvc64Fg>t00?wp0000,0*74
!AIVDM,1,1,,A,10lviS003pIvc7nFg?aP0?wp0000,0*66

Appendix B.2: Results from VHDL Model

!AIVDM,1,1,,A,10lviS0080Ivc4lFg?Ih0?wp0000,0*3C
!AIVDM,1,1,,A,10lviS003PIvc9rFg?SP0?wp0000,0*66
!AIVDM,1,1,,A,10lviS0099Ivc>`Fg?Q00?wp0000,0*72
!AIVDM,1,1,,A,10lviS001uIvcCvFg?O00?wp0000,0*43
!AIVDM,1,1,,A,10lviS00:gIvcF@Fg?1@0?wp0000,0*67
!AIVDM,1,1,,A,10lviS009VIvcF>Fg>`P0?wp0000,0*6B
!AIVDM,1,1,,A,10lviS000rIvcEpFg=uP0?wp0000,0*1D
!AIVDM,1,1,,A,10lviS000rIvcENFg=I@0?wp0000,0*0F
!AIVDM,1,1,,A,10lviS002MIvcM>Fg=;@0?wp0000,0*38
!AIVDM,1,1,,A,10lviS000EIvcONFg<<P0?wp0000,0*56
!AIVDM,1,1,,A,10lviS006vIvcPjFg;I@0?wp0000,0*3A
!AIVDM,1,1,,A,10lviS00=DIvcSHFg:Ch0?wp0000,0*01
!AIVDM,1,1,,A,10lviS009pIvcU0Fg9IP0?wp0000,0*7E
!AIVDM,1,1,,A,10lviS008EIvcWNFg8?00?wp0000,0*21
!AIVDM,1,1,,A,10lviS00=lIvcaRFg7H00?wp0000,0*5F
!AIVDM,1,1,,A,10lviS001TIvcQnFg7A@0?wp0000,0*1E
!AIVDM,1,1,,A,10lviS00>QIvcGRFg72P0?wp0000,0*5D
!AIVDM,1,1,,A,10lviS001gIvc=JFg6n@0?wp0000,0*4B
!AIVDM,1,1,,A,10lviS008AIvc0nFg6U@0?wp0000,0*76
!AIVDM,1,1,,A,10lviS002CIvbkrFg6D00?wp0000,0*59
!AIVDM,1,1,,A,10lviS008uIvbQ@Fg5t@0?wp0000,0*2E
!AIVDM,1,1,,A,10lviS0005IvbE4Fg5fh0?wp0000,0*3C
!AIVDM,1,1,,A,10lviS00<AIvb3@Fg5GP0?wp0000,0*5F
!AIVDM,1,1,,A,10lviS00=UIvaj>Fg52h0?wp0000,0*23
!AIVDM,1,1,,A,10lviS00>bIvag>Fg6MP0?wp0000,0*5E
!AIVDM,1,1,,A,10lviS00?jIvachFg8800?wp0000,0*1E
!AIVDM,1,1,,A,10lviS0085Iva`8Fg9n00?wp0000,0*42
!AIVDM,1,1,,A,10lviS004FIvaU6Fg;F00?wp0000,0*2C
!AIVDM,1,1,,A,10lviS001`IvaPhFg=N00?wp0000,0*5A
!AIVDM,1,1,,A,10lviS0088IvaLvFg?Bh0?wp0000,0*5F
!AIVDM,1,1,,A,10lviS009HIvaHRFgB2h0?wp0000,0*03
!AIVDM,1,1,,A,10lviS00<=IvaEPFgD<@0?wp0000,0*5C
!AIVDM,1,1,,A,10lviS001EIvaAdFgFN00?wp0000,0*19
!AIVDM,1,1,,A,10lviS00:UIva<@FgIVh0?wp0000,0*14
!AIVDM,1,1,,A,10lviS008AIva9lFgKkh0?wp0000,0*14
!AIVDM,1,1,,A,10lviS002gIva4<FgNt@0?wp0000,0*57
!AIVDM,1,1,,A,10lviS00?1Iva0HFgQM00?wp0000,0*2A
!AIVDM,1,1,,A,10lviS009MIv`thFgSF@0?wp0000,0*4C

!AIVDM,1,1,,A,10lviS0073Iv`jfFgUIP0?wp0000,0*35
!AIVDM,1,1,,A,10lviS00?4Iv`S0FgW?00?wp0000,0*41
!AIVDM,1,1,,A,10lviS00:OIv`5@FgbtP0?wp0000,0*37
!AIVDM,1,1,,A,10lviS007?IvWpvFgd`@0?wp0000,0*0C
!AIVDM,1,1,,A,10lviS00=LIvWWnFgfr@0?wp0000,0*5A
!AIVDM,1,1,,A,10lviS008QIvWS@Fgi>P0?wp0000,0*3B
!AIVDM,1,1,,A,10lviS008oIvW`HFgif@0?wp0000,0*76
!AIVDM,1,1,,A,10lviS00:pIvWhfFgiw@0?wp0000,0*5C
!AIVDM,1,1,,A,10lviS005pIv`4nFgjF@0?wp0000,0*02
!AIVDM,1,1,,A,10lviS003mIv`R8FgjSP0?wp0000,0*2C
!AIVDM,1,1,,A,10lviS009AIvaFfFgk7h0?wp0000,0*1C
!AIVDM,1,1,,A,10lviS00=oIvan0Fgkg00?wp0000,0*40
!AIVDM,1,1,,A,10lviS006QIvb>@Fgl?P0?wp0000,0*69
!AIVDM,1,1,,A,10lviS001lIvbSvFglkP0?wp0000,0*5C
!AIVDM,1,1,,A,10lviS0077IvblHFgm<P0?wp0000,0*56
!AIVDM,1,1,,A,10lviS004lIvc=dFgmgh0?wp0000,0*11
!AIVDM,1,1,,A,10lviS006KIvcW2Fgn=@0?wp0000,0*79
!AIVDM,1,1,,A,10lviS00=EIvd3tFgnv@0?wp0000,0*12
!AIVDM,1,1,,A,10lviS006MIvdMpFgoK00?wp0000,0*27
!AIVDM,1,1,,A,10lviS006@Ive36Fgomh0?wp0000,0*6D
!AIVDM,1,1,,A,10lviS005iIve`NFgowh0?wp0000,0*76
!AIVDM,1,1,,A,10lviS002@Ivf1vFgovP0?wp0000,0*0B
!AIVDM,1,1,,A,10lviS004;IvfWnFgok00?wp0000,0*75
!AIVDM,1,1,,A,10lviS001IIvg0fFgoQh0?wp0000,0*0E
!AIVDM,1,1,,A,10lviS006pIvgEhFgo700?wp0000,0*75
!AIVDM,1,1,,A,10lviS0048IvggbFgnnh0?wp0000,0*17
!AIVDM,1,1,,A,10lviS004iIvhEDFgnHP0?wp0000,0*53
!AIVDM,1,1,,A,10lviS006kIvhnrFgn1h0?wp0000,0*0F
!AIVDM,1,1,,A,10lviS001rIvi>6Fgmr@0?wp0000,0*6C
!AIVDM,1,1,,A,10lviS007sIvigPFgn0@0?wp0000,0*15
!AIVDM,1,1,,A,10lviS00;CIvjE4Fgn0@0?wp0000,0*6C
!AIVDM,1,1,,A,10lviS003rIvjrlFgmQP0?wp0000,0*48
!AIVDM,1,1,,A,10lviS00<TIvkFtFglg@0?wp0000,0*6B
!AIVDM,1,1,,A,10lviS001<IvkfjFgkTh0?wp0000,0*2C
!AIVDM,1,1,,A,10lviS006CIvl?>FgiV00?wp0000,0*06
!AIVDM,1,1,,A,10lviS0004IvlfPFggL00?wp0000,0*54
!AIVDM,1,1,,A,10lviS003RIvm:rFgeM00?wp0000,0*4D
!AIVDM,1,1,,A,10lviS0002IvmPRFgd?h0?wp0000,0*4F
!AIVDM,1,1,,A,10lviS0032Ivmf0FgU`P0?wp0000,0*4E
!AIVDM,1,1,,A,10lviS002BIvmadFgP<@0?wp0000,0*25
!AIVDM,1,1,,A,10lviS004CIvm`HFgN3@0?wp0000,0*1E
!AIVDM,1,1,,A,10lviS002kIvmUnFgKSh0?wp0000,0*6E
!AIVDM,1,1,,A,10lviS002>IvmT:FgHwP0?wp0000,0*71
!AIVDM,1,1,,A,10lviS009UIvmQRFgF=00?wp0000,0*58
!AIVDM,1,1,,A,10lviS00>JIvmO0FgB>00?wp0000,0*3B

!AIVDM,1,1,,A,10lviS00?1Ivm=DFg@s00?wp0000,0*08
!AIVDM,1,1,,A,10lviS003SIvlqlFgA4P0?wp0000,0*25
!AIVDM,1,1,,A,10lviS007fIvlRTFgA@h0?wp0000,0*43
!AIVDM,1,1,,A,10lviS0061IvkwHFgASh0?wp0000,0*38
!AIVDM,1,1,,A,10lviS008HIvkd4FgAa00?wp0000,0*4A
!AIVDM,1,1,,A,10lviS004?IvkI@FgAn00?wp0000,0*67
!AIVDM,1,1,,A,10lviS0016Ivk5fFgAvP0?wp0000,0*49
!AIVDM,1,1,,A,10lviS006wIvjfrFgB6@0?wp0000,0*1A
!AIVDM,1,1,,A,10lviS002jIvjQ8FgB8h0?wp0000,0*58
!AIVDM,1,1,,A,10lviS000KIvj:BFgBK00?wp0000,0*41
!AIVDM,1,1,,A,10lviS00?AIviePFgBUh0?wp0000,0*4C
!AIVDM,1,1,,A,10lviS006qIviG0FgBfP0?wp0000,0*3C
!AIVDM,1,1,,A,10lviS00?HIvhu@FgBq00?wp0000,0*38
!AIVDM,1,1,,A,10lviS00<<IvhLvFgC5h0?wp0000,0*5D
!AIVDM,1,1,,A,10lviS0008Ivgi0FgCIP0?wp0000,0*7D
!AIVDM,1,1,,A,10lviS00:pIvg@RFgCVh0?wp0000,0*53
!AIVDM,1,1,,A,10lviS00;CIvfe0FgDgP0?wp0000,0*29
!AIVDM,1,1,,A,10lviS00:DIvf@tFgGC@0?wp0000,0*79
!AIVDM,1,1,,A,10lviS008nIverVFgEBh0?wp0000,0*69
!AIVDM,1,1,,A,10lviS003PIve4PFgD`P0?wp0000,0*07
!AIVDM,1,1,,A,10lviS00<GIvdQdFgDoP0?wp0000,0*40
!AIVDM,1,1,,A,10lviS003bIvccHFgEGP0?wp0000,0*5A
!AIVDM,1,1,,A,10lviS005tIvbobFgGL00?wp0000,0*04
!AIVDM,1,1,,A,10lviS00>@Ivb1:FgGo@0?wp0000,0*6E
!AIVDM,1,1,,A,10lviS00=eIvafbFgGuh0?wp0000,0*76
!AIVDM,1,1,,A,10lviS006LIvag<FgF`P0?wp0000,0*27
!AIVDM,1,1,,A,10lviS0056IvanlFgDrh0?wp0000,0*2F
!AIVDM,1,1,,A,10lviS009gIvaqBFgCQ00?wp0000,0*3F
!AIVDM,1,1,,A,10lviS00>TIvapTFgBTh0?wp0000,0*40
!AIVDM,1,1,,A,10lviS00>SIvahVFgBL00?wp0000,0*1D
!AIVDM,1,1,,A,10lviS009NIvajhFg@vP0?wp0000,0*63
!AIVDM,1,1,,A,10lviS005EIvalTFg@7P0?wp0000,0*1F
!AIVDM,1,1,,A,10lviS00=aIvauTFg@:P0?wp0000,0*27
!AIVDM,1,1,,A,10lviS0075IvawHFg?<h0?wp0000,0*26
!AIVDM,1,1,,A,10lviS00>NIvb2RFg=Qh0?wp0000,0*67
!AIVDM,1,1,,A,10lviS000RIvb5VFg<7h0?wp0000,0*11
!AIVDM,1,1,,A,10lviS008QIvb8DFg:s@0?wp0000,0*6F
!AIVDM,1,1,,A,10lviS00;MIvbFTFg:gh0?wp0000,0*22
!AIVDM,1,1,,A,10lviS002pIvbRTFg:u@0?wp0000,0*38
!AIVDM,1,1,,A,10lviS005IIvbmbFg;F@0?wp0000,0*3D
!AIVDM,1,1,,A,10lviS0031Ivc66Fg;`@0?wp0000,0*6B
!AIVDM,1,1,,A,10lviS005:IvcFLFg;vh0?wp0000,0*52
!AIVDM,1,1,,A,10lviS006NIvcO6Fg<800?wp0000,0*47
!AIVDM,1,1,,A,10lviS008jIvcMNFg=1P0?wp0000,0*7F
!AIVDM,1,1,,A,10lviS000jIvcBdFg=Rh0?wp0000,0*09

!AIVDM,1,1,,A,10lviS008nIvc9HFg>9h0?wp0000,0*3A
!AIVDM,1,1,,A,10lviS004JIvc64Fg>t00?wp0000,0*74
!AIVDM,1,1,,A,10lviS003pIvc7nFg?aP0?wp0000,0*66

APPENDIX C

Appendix C.1: Coordinates Used to Develop AIS Packet

39.737772,-84.175957
39.737837,-84.175685
39.737820,-84.175433
39.737807,-84.175148
39.737608,-84.175027
39.737443,-84.175028
39.737157,-84.175047
39.736915,-84.175068
39.736822,-84.174655
39.736403,-84.174535
39.736062,-84.174465
39.735598,-84.174327
39.735210,-84.174240
39.734713,-84.174108
39.734347,-84.173998
39.734302,-84.174408
39.734203,-84.174958
39.734122,-84.175498
39.734008,-84.176168
39.733893,-84.176858
39.733735,-84.177853
39.733645,-84.178503
39.733490,-84.179453
39.733352,-84.180362
39.733957,-84.180522
39.734667,-84.180707
39.735400,-84.180900
39.736040,-84.181062
39.736947,-84.181293
39.737725,-84.181495
39.738898,-84.181732
39.739815,-84.181893
39.740787,-84.182097
39.742125,-84.182387

39.743065,-84.182517
39.744402,-84.182817
39.745473,-84.183020
39.746282,-84.183213
39.747157,-84.183748
39.747940,-84.184587
39.749523,-84.186173
39.750242,-84.186828
39.751215,-84.187742
39.752203,-84.187987
39.752415,-84.187713
39.752528,-84.187268
39.752682,-84.186195
39.752770,-84.184633
39.753012,-84.181828
39.753273,-84.180160
39.753490,-84.178867
39.753730,-84.177708
39.753897,-84.176833
39.754132,-84.175483
39.754328,-84.174132
39.754655,-84.172590
39.754847,-84.171207
39.755025,-84.169222
39.755092,-84.167228
39.755083,-84.165868
39.755007,-84.163848
39.754892,-84.162522
39.754713,-84.161400
39.754605,-84.160018
39.754403,-84.158010
39.754252,-84.156218
39.754202,-84.154982
39.754242,-84.153200
39.754242,-84.151197
39.754037,-84.149183
39.753702,-84.147683
39.753205,-84.146412
39.752360,-84.144682
39.751440,-84.143013
39.750593,-84.141498
39.750078,-84.140345
39.747257,-84.139627
39.744935,-84.139857
39.744022,-84.139927

39.742958,-84.140062
39.741863,-84.140152
39.740673,-84.140292
39.738973,-84.140427
39.738420,-84.141370
39.738483,-84.142410
39.738565,-84.143650
39.738692,-84.145527
39.738727,-84.146557
39.738813,-84.147560
39.738870,-84.148602
39.738922,-84.149818
39.738938,-84.150553
39.739060,-84.151772
39.739132,-84.153307
39.739190,-84.154507
39.739260,-84.155880
39.739345,-84.157602
39.739477,-84.159947
39.739565,-84.161678
39.740050,-84.163573
39.741142,-84.165070
39.740285,-84.166262
39.740003,-84.169147
39.740103,-84.171003
39.740317,-84.173900
39.741200,-84.176658
39.741382,-84.179565
39.741425,-84.180552
39.740857,-84.180523
39.740125,-84.180117
39.739527,-84.179985
39.739125,-84.180023
39.739067,-84.180448
39.738443,-84.180333
39.738077,-84.180237
39.738097,-84.179757
39.737685,-84.179660
39.736972,-84.179492
39.736372,-84.179328
39.735862,-84.179183
39.735785,-84.178423
39.735875,-84.177783
39.736042,-84.176765
39.736162,-84.175888

39.736312,-84.175017
39.736373,-84.174555
39.736757,-84.174642
39.736978,-84.175217
39.737238,-84.175713
39.737573,-84.175890
39.737877,-84.175795

Appendix C.2: Resulting Coordinates from MATLAB® and VHDL Model

39.737772,-84.175957
39.737837,-84.175685
39.737820,-84.175433
39.737807,-84.175148
39.737608,-84.175027
39.737443,-84.175028
39.737157,-84.175047
39.736915,-84.175068
39.736822,-84.174655
39.736403,-84.174535
39.736062,-84.174465
39.735598,-84.174327
39.735210,-84.174240
39.734713,-84.174108
39.734347,-84.173998
39.734302,-84.174408
39.734203,-84.174958
39.734122,-84.175498
39.734008,-84.176168
39.733893,-84.176858
39.733735,-84.177853
39.733645,-84.178503
39.733490,-84.179453
39.733352,-84.180362
39.733957,-84.180522
39.734667,-84.180707
39.735400,-84.180900
39.736040,-84.181062
39.736947,-84.181293
39.737725,-84.181495
39.738898,-84.181732
39.739815,-84.181893
39.740787,-84.182097
39.742125,-84.182387
39.743065,-84.182517
39.744402,-84.182817
39.745473,-84.183020
39.746282,-84.183213

39.747157,-84.183748
39.747940,-84.184587
39.749523,-84.186173
39.750242,-84.186828
39.751215,-84.187742
39.752203,-84.187987
39.752415,-84.187713
39.752528,-84.187268
39.752682,-84.186195
39.752770,-84.184633
39.753012,-84.181828
39.753273,-84.180160
39.753490,-84.178867
39.753730,-84.177708
39.753897,-84.176833
39.754132,-84.175483
39.754328,-84.174132
39.754655,-84.172590
39.754847,-84.171207
39.755025,-84.169222
39.755092,-84.167228
39.755083,-84.165868
39.755007,-84.163848
39.754892,-84.162522
39.754713,-84.161400
39.754605,-84.160018
39.754403,-84.158010
39.754252,-84.156218
39.754202,-84.154982
39.754242,-84.153200
39.754242,-84.151197
39.754037,-84.149183
39.753702,-84.147683
39.753205,-84.146412
39.752360,-84.144682
39.751440,-84.143013
39.750593,-84.141498
39.750078,-84.140345
39.747257,-84.139627
39.744935,-84.139857
39.744022,-84.139927
39.742958,-84.140062
39.741863,-84.140152
39.740673,-84.140292
39.738973,-84.140427

39.738420,-84.141370
39.738483,-84.142410
39.738565,-84.143650
39.738692,-84.145527
39.738727,-84.146557
39.738813,-84.147560
39.738870,-84.148602
39.738922,-84.149818
39.738938,-84.150553
39.739060,-84.151772
39.739132,-84.153307
39.739190,-84.154507
39.739260,-84.155880
39.739345,-84.157602
39.739477,-84.159947
39.739565,-84.161678
39.740050,-84.163573
39.741142,-84.165070
39.740285,-84.166262
39.740003,-84.169147
39.740103,-84.171003
39.740317,-84.173900
39.741200,-84.176658
39.741382,-84.179565
39.741425,-84.180552
39.740857,-84.180523
39.740125,-84.180117
39.739527,-84.179985
39.739125,-84.180023
39.739067,-84.180448
39.738443,-84.180333
39.738077,-84.180237
39.738097,-84.179757
39.737685,-84.179660
39.736972,-84.179492
39.736372,-84.179328
39.735862,-84.179183
39.735785,-84.178423
39.735875,-84.177783
39.736042,-84.176765
39.736162,-84.175888
39.736312,-84.175017
39.736373,-84.174555
39.736757,-84.174642
39.736978,-84.175217

39.737238,-84.175713
39.737573,-84.175890
39.737877,-84.175795