

GRAPHICS PROCESSING UNIT BASED HIGH SPEED DEMODULATION

By

CATHERINE MCINTOSH

A Thesis Submitted to The Honors College

In Partial Fulfillment of the Bachelor's degree
With Honors in

Computer Engineering

THE UNIVERSITY OF ARIZONA

M A Y 2 0 1 9

Approved by:

Gary Redford
UA ENGR 498 Coordinator

Abstract

Demodulation of high speed, wide bandwidth waveforms has typically been performed by application-specific integrated circuits (ASICs) or large central processing unit (CPU) clusters. However, development times for these technologies can be long and expensive, especially in the case of ASICs. While general purpose processors are flexible and easier to program, they often lack the throughput required to handle high speed waveform demodulation. Graphics processing unit (GPUs) open new avenues for flexible demodulators that can be developed quickly, easily modified, and easily maintained. This project seeks to harness the power of GPU parallel processing to increase our effective throughput while maintaining a low implementation loss. To do this, our design utilizes an industry-leading GPU development environment, CUDA, and the highest throughput GPU on the market, RTX 2080, to optimize the repetitive processes during demodulation. With our system, we can exploit these processes through parallelization while maintaining a modest implementation loss. The results of this design produce a large speed-up of 132 Mb/s, impressive compared to the 2 Mb/s of our CPU control system, with a bit error rate within 1 dB of Shannon's theoretical rate for an 8-phase shift-keying detector.



Graphics Processing Unit Based High Speed Demodulation

Final Report

Team #18051

Members:

Kray Althaus, Catherine McIntosh, Josue Ortiz,
Kevin Siruno, Sebastian Thiem

Sponsored by:

General Dynamics

Mentor:

Claude Merrill

7 May 2019



Table of Contents

Table of Contents	1
1.0 Scope	3
2.0 System Block Diagram	3
3.0 Technical Data Package	4
INDENTURED DOCUMENT LIST	4
100,000 High Speed Demodulation System Top Assembly Drawing	6
102,000 Software System Designs	7
102,100 Demodulation SDD	7
102,200 User Interface SDD	11
102,300 Benchmarking SDD	13
102,400 Matched Filter SDD	20
102,500 Slicer and Decoder SDD	22
103,000 Systems Requirement Document	26
4.0 Acceptance Test Procedure	32
101,000 Acceptance Test Procedure	33
5.0 Models / Analyses	39
6.0 Acceptance Test Results	40
7.0 Final Budget	43
8.0 Lessons Learned	43
Technical Lessons	43
Demodulation	43
GPU Development	43
Project Organization Lessons	43
Professional Team Lessons	43
Conclusion	44
Appendix A - Models and Analyses Documents	46
104,000 System Models	46
104,100 Cost Model	46
104,200 Bandwidth Model	48

104,300 Modulation Model	49
104,400 BER vs. Signal Noise Ratio Model	50
104,500 Gaussian Noise Model	51
104,600 Throughput Model	52

1.0 Scope

This project's goal is develop an optimized demodulation system using a graphics processing unit (GPU) platform. This will involve surveying currently available GPUs on the market as suitable candidates and obtaining at least one unit to develop the demodulation system on. The purpose of developing this system is to determine the viability of GPUs as a platform for performing high-speed demodulation and whether they can perform as well as current central processing unit (CPU) and application-specific integrated circuit (ASIC) based systems. Due to its high number of processing cores and high throughput, the development time and maintenance of a GPU based system would be shorter and less expensive than CPU and ASIC based systems. To test the viability of these types of demodulation receivers in this market, our General GPU-Based High Speed Demodulation system is designed to minimize overhead and exploit parallel processing for optimal performance.

2.0 System Block Diagram

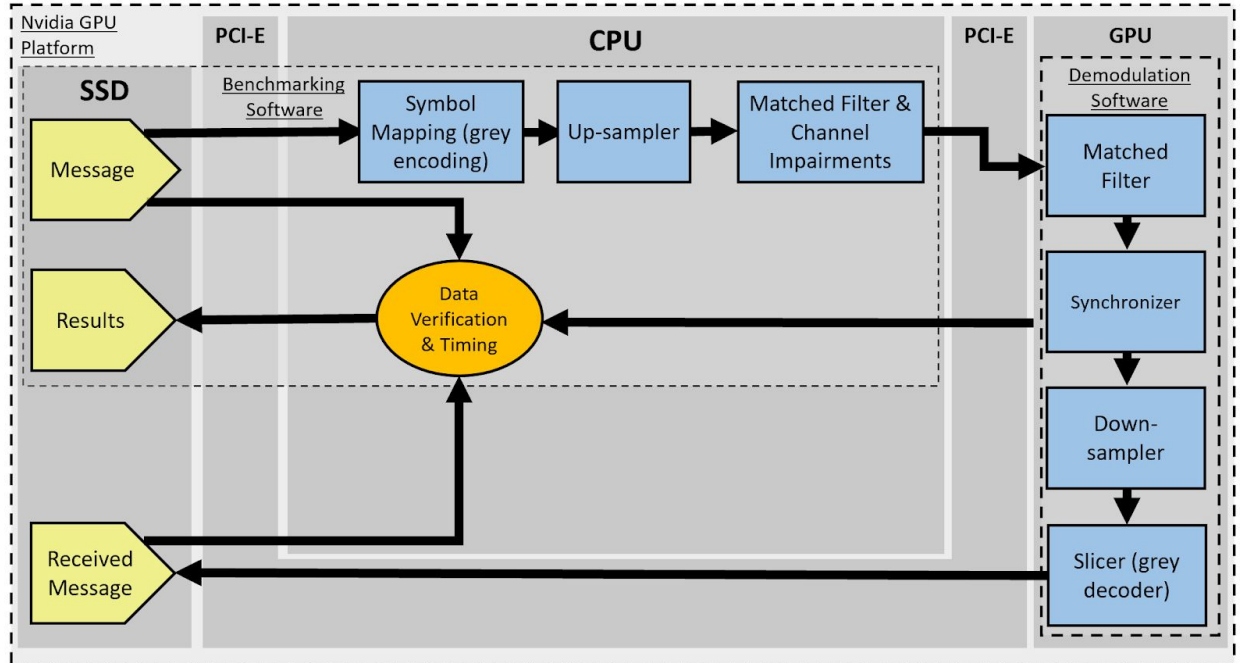


Figure 1: System Block Diagram

3.0 Technical Data Package

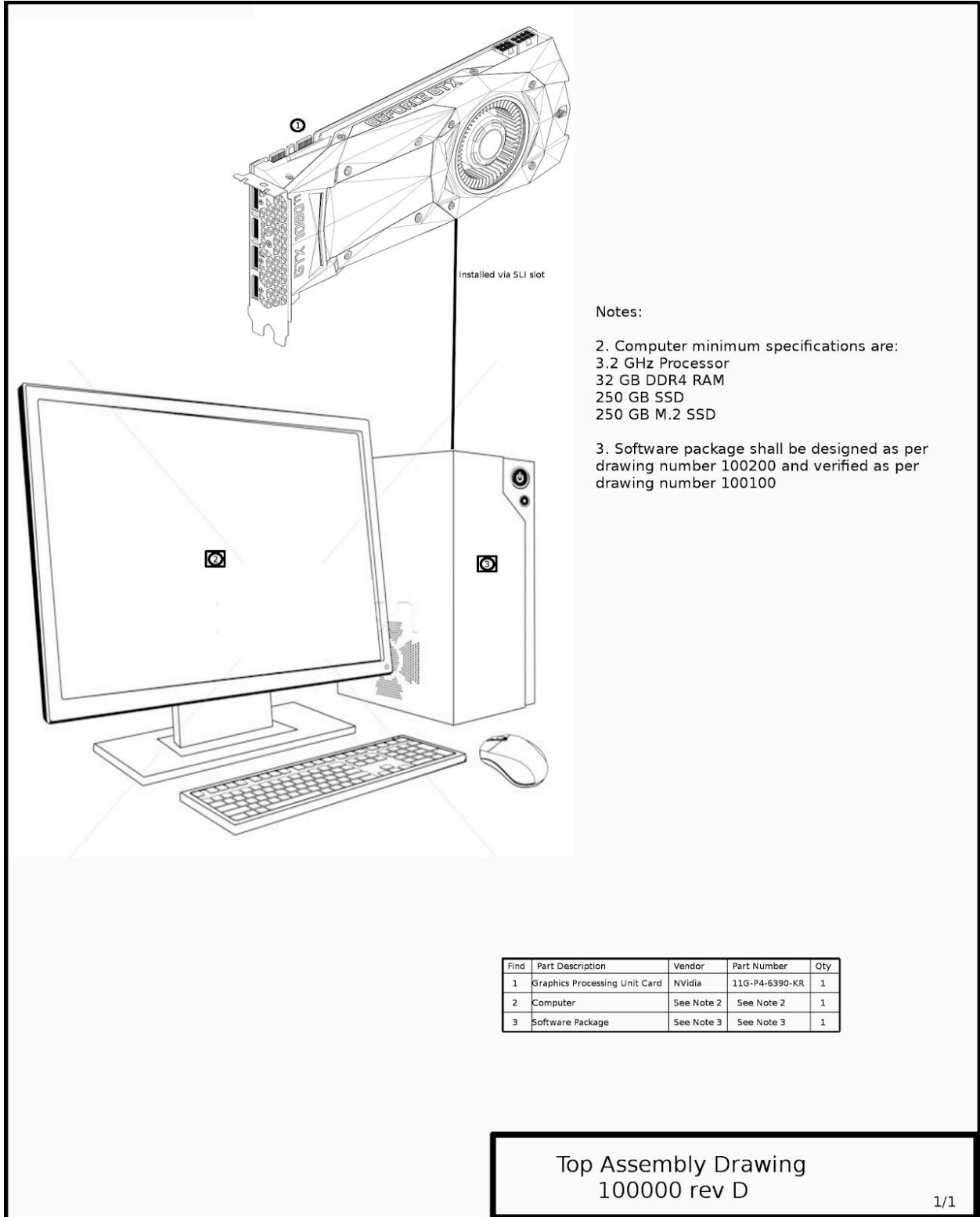
INDENTURED DOCUMENT LIST

The Graphics Processing Unit Based High Speed Demodulation system's technical data packet is comprised of our system top assembly drawing, our acceptance test procedures, test datasheets, several details software design documents, and our current system requirement document. This section includes the Top Assembly Drawing Document, all Software System Designs and the System Requirements document. Document #101,100 Acceptance Test Procedures can be found in the following section 4.0 Acceptance Test Procedure. The documents pertaining to the system models (#104,400 - 104,600) are mentioned in section 5.0 Models / Analyses and can be found in Appendix A.

Table 1. Indentured Document List		
Part Number	Document	Release Date
100,000	High Speed Demodulation System Top Assembly Drawing	11/27/2018
101,000	Acceptance Test Procedures	1/15/2019
102,000	Software System Designs	1/15/2019
102,100	Demodulation SDD	-
102,200	User Interface SDD	-
102,300	Benchmarking SDD	-
102,400	Matched Filter SDD	-
102,500	Decoder and Slicer SDD	-

103,000	System Requirement Document	11/26/2018
104,000	System Models	11/26/2018
104,100	Cost Model	-
104,200	Bandwidth Model	-
104,300	Modulation Model	-
104,400	Bit Error Rate vs Signal-to-Noise Model	-
104,500	Gaussian Noise Model	-
104,600	Throughput Model	-

100,000 High Speed Demodulation System Top Assembly Drawing



Notes:

2. Computer minimum specifications are:
 3.2 GHz Processor
 32 GB DDR4 RAM
 250 GB SSD
 250 GB M.2 SSD

3. Software package shall be designed as per drawing number 100200 and verified as per drawing number 100100

Find	Part Description	Vendor	Part Number	Qty
1	Graphics Processing Unit Card	Nvidia	11G-P4-6390-KR	1
2	Computer	See Note 2	See Note 2	1
3	Software Package	See Note 3	See Note 3	1

Top Assembly Drawing
 100000 rev D

1/1

102,000 Software System Designs

102,100 Demodulation SDD

1.0 Scope

This document provides the detail of the design of the Computer Software Configuration Item (CSCI) Demodulation Script Subsystem for the High Speed GPU Demodulation System. It defines the software modules required to take a modulated signal and coherently demodulate it including removing channel impairment, working with predefined packet sizes, and decoding both redundant encoding and gray coding.

2.0 Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, User Interface Software Design Document #102,200, January 15, 2019, Rev A

Graphics Processing Unit Based High Speed Demodulation, Matched Filter Software Design Document #102,400, January 15, 2019, Rev A

Graphics Processing Unit Based High Speed Demodulation, Slicer & Decoder Software Design Document #102,500, January 15, 2019, Rev A

4.2.1 CUDA enabled NVidia GPU: The software shall run on an NVidia GPU equipped computer.

4.3.2 Arrival Time: The system shall determine the time of the signal's arrival and first symbol.

4.3.3 Unknown Phase: The system shall determine the phase of the signal's first symbol.

4.3.4 Frequency Offset: The system shall operate in the complex baseband without a frequency offset.

4.3.5 Encoding: Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.

4.3.6 Demodulation Coherence: The signal's phase offset shall be adjusted to reference 0°.

4.3.7 Data Type: Data shall be processed as a floating point type.

4.4.1 Modulation Form: The system shall demodulate an 8 phase-shift keying (PSK) signal.

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

4.4.3 Packet Size: Demodulation packet sizes shall be no less than 2^{14} symbols.

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.

3.0 Demodulation Script Subsystem CSCI Design Decisions

The demodulation script's inputs are the modulated message file location, the final demodulated message destination, the roll-off factor, and the oversample rate. All of the inputs except the final demodulated message destination will be passed along to the first RRC filter so that noise, which has been added in compliance with Requirements 4.4.2 and 4.4.4 (SRD Document #103,000), can be removed. The message will be filtered according to the constraints given and the design found in Document #102,400. The final demodulated message destination will be passed into the slicer and decoder module, along with the other outputs of the RRC filter, such as the filtered message, to go through the final steps of the demodulation process. These steps include removing the redundant encoding and converting the gray coded values added in compliance with Requirement 4.3.5 to binary by checking the values from a given set of "identical" values to determine which will be the best fit to the scheme shown in Section 4.4.1 in this document. The detailed description of the slicer and decoder module can be found in Document #102,500. The final output of the demodulation system will be the coherently demodulated message to the designated output file.

The flow diagram for this system can be seen in Section 4.2 and was largely designed around the inputs to the system and the steps necessary to complete all the requirements are highlighted in Section 6 in yellow. Mainly, the majority of the steps needed to be linearized to some extent in order to ensure the message will come out coherently decoded with proper phase, packet size, and noise levels after having been encoded and modulated. All the software defined in this module and its referenced underlying documents will be implemented using CUDA based C++ in order to optimize performance requirements such as Requirement 4.4.5 (SRD Document #103,000).

4.0 Demodulation Script Subsystem CSCI Architectural Design

4.1 Demodulation Script Components

4.1.1 Symbol Mapper

A module is needed to convert 3-bits of binary data to complex floating point data. The conversion follows the gray coding scheme specified in Figure 1 below.

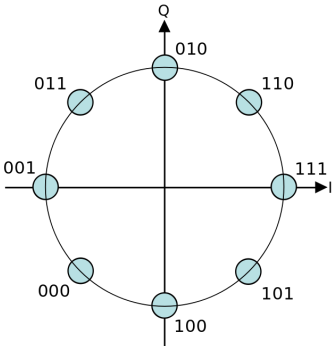


Figure 1. Gray Coding Scheme

4.1.2 Root Raised Cosine Filter

This filter is applied on both ends of the transmission: in the transmitter after the packet is assembled, and in the receiver after the packet is received.

4.1.3 Synchronizer

This module scans the received IQ data for a start condition to signify the beginning of a packet.

4.1.4 Slicer/Decoder

The slicer takes downsampled packets, and outputs the binary equivalent data, based on the gray coding scheme in Figure 1.

4.2 System Flow Diagram

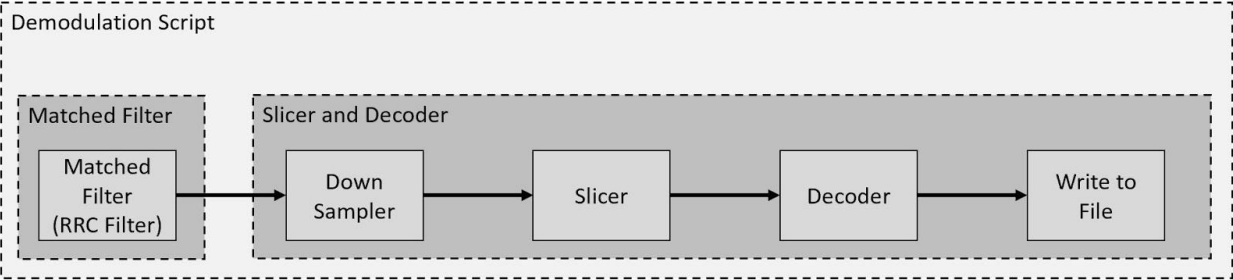


Figure 2. System Flow Diagram

5.0 Demodulation Script Subsystem CSCI Detailed Design

5.1 Root Raised Cosine Filter

The module requires convolution, and as such a convolution module has been drafted as a sub-function to this filter.

5.2 Synchronizer

The file parser reads in a buffer of IQ data from local storage, and calculates the difference in phase between adjacent symbols. An I-Q stream was randomly generated from the C++ Standard Library Mersenne Twister number generator in order to ensure the autocorrelation of the preamble was one at zero and relatively low at other sample points. The phase differences of each I-Q pair in the preamble is compared to the input of the demodulation system. When the phase differentials are correct for a tolerance of 1/8 of the original 64 expected I-Q pairs, the preamble is deemed to be found and a boolean flag is raised. This marks the beginning of downsampling..

5.3 Slicer/Decoder

C++ is limited to byte-wise operations so symbols are decoded at a minimum of 8 symbols at a time; generating 3 bytes symbols. The slicer/decoder will also include the symbol mapping features described in Section 4.1.1

6.0 Requirement Traceability

Traceability Matrix							
System Requirement	VM	Sub-Assembly					
		Demodulation Scripts	V M	Benchmarking	V M	NVidia GPU Platform	V M
4.1.1 Video Interface	I			BMK 1.0 Direct Flow	I	NVG 1.0 Direct Flow	I
4.2.1 CUDA enabled NVidia GPU	I	SDM 1.0 Direct Flow	I	BMK 2.0 Direct Flow	I	NVG 2.0 Direct Flow	I
4.2.2 Simulation Environment	I			BMK 3.0 Direct Flow	I		
4.3.1 Cost	I					NVG 3.0 Allocated \$3000	I
4.3.2 Arrival Time	D	SDM 2.0 Direct Flow	D				
4.3.3 Unknown Phase	D	SDM 3.0 Direct Flow	D				
4.3.4 Frequency Offset	I	SDM 4.0 Direct Flow	I				
4.3.5 Encoding	I	SDM 5.0 Direct Flow	I				
4.3.6 Demodulation Coherence	I	SDM 6.0 Direct Flow	I				

4.3.7 Data Type	I	SDM 7.0 Direct Flow	I	BMK 4.0 Direct Flow	I		
4.4.1 Modulation Form	I	SDM 8.0 Direct Flow	I				
4.4.2 Implementation Loss	T A	SDM 9.0 Direct Flow		BMK 5.0 Direct Flow	T A		
4.4.3 Packet Size	I	SDM 10.0 Direct Flow	I				
4.4.4 Additive Noise Type	I	SDM 11.0 Direct Flow	I				
4.4.5 Throughput	T A			BMK 6.0 Direct Flow	T A		

102,200 User Interface SDD

1.0 Scope

This document provides the detail of the design of the Computer Software Configuration Item (CSCI) User Interface Subsystem (UIS) of the interface for the High Speed GPU Demodulation System. It defines the software modules required for the user to interact with the entire system. This includes the necessary files to be ran by the user, the files which will be output by the system and how to interpret the data in the files which will be output by the system.

2.0 Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document #103,000, January 15, 2019, Rev F

4.1.1 Video Interface: The demodulation software shall be accessed and executed through a terminal interface on a Linux system.

3.0 User Interface Subsystem CSCI Design Decisions

The system begins by executing the compiled binaries with a specified input file, output directory, and optional benchmarking flag, and benchmarking directory. That interface will process a locally stored message file as an input, and output the demodulated message, with optional benchmarking file(s).

Users will be able to select the address of a local 8-psk modulated message file, as well as the address to output the demodulated message and optional benchmarking results.

The output message or benchmarking files can then be viewed by the user, by opening the respective file in their operating system; SRD requirement 4.1.1 Video Interface. File formats shall be supported by default OS software.

Benchmarking data shall include, but is not limited to: total throughput in Gigabytes per second (GB/s), throughput and timing analysis for sub-modules, and error rate in bit errors per million bits.

4.0 User Interface Subsystem CSCI Architectural Design

Below are the CSCs, or modules/views that are used in this CSCI, and their relationships. The logic and routines given for each CSC are to detail the design principles. The designer is free to use of libraries to accomplish the same purposes.

Table 1 CSC Properties Included in the User Interface Subsystem CSCI				
CSC	Inputs (source)	Outputs (destination)	Purpose	Library
Demodulation UI	8-PSK Modulated message (file)	Message (console window)	Allow users to demodulate modulated data.	N/a
Benchmarking UI	8-PSK Modulated message (file)	Message (console window), Benchmarking Results File (ber.csv, all received message files generated during channel sweep)	Allow users to demodulated and benchmark the demodulation process.	N/a

5.0 User Interface Subsystem CSCI Detailed Design

The unit design decisions for each of the CSCs within the CSCI such as mockups, data handling and logic are given below.

5.1 Demodulation UI Component

The Demodulation UI handles how the user selects what file to demodulate, and where to output the message when the data is demodulated. The Demodulation UI is the main view of the system and the first view a user sees upon entering the system. The UI is

built on C++, and provides the functions necessary for a user to demodulate 8-PSK modulated data. The functions are defined in Table 2.

Table 2. Demodulation UI CSC Functions			
Functions	Input (Source)	Output (Destination)	Description
Specify Input Files/Directories	User Text	Input/output Addresses	Getting the address of the modulated data and the directory for output, from the user.
Run Demodulation	8-PSK Modulated Data (file)	Message (console window)	Executes the demodulation program given the user selected parameters.

5.2 Benchmarking UI Component

The Benchmarking UI allows users to view benchmarks on the demodulation software. Users can chose to generate benchmarking data, and select where to save that data to. The Benchmarking UI is embedded in the Demodulation UI, and is the key to presenting the performance of the system. The functions are defined in Table 3.

Table 3. Benchmarking UI CSC Functions			
Functions	Input (Source)	Output (Destination)	Description
Specify Output Directory	User Text	Address of Output Directory	Getting the directory for output, from the user.
Generate Benchmarks	Output Address	Benchmarking Results	Executes the benchmarking software.

6.0 Requirement Traceability

UIS Requirements Flow Down			
Subsystem Requirement	V M	Subassembly	
		Demodulation	Benchmarking
4.1.1 Direct Flow		The user interface shall allow the user to define the directories of the input signal and output signal.	The user interface shall allow the user to define the directories of the benchmarking results output file.

102,300 Benchmarking SDD

1. Scope

The system shall benchmark the performance of the high speed GPU demodulation modules described in document 102,100.

This software is necessary to functionally verify the performance requirements requested by the customer, General Dynamics. The benchmarking is done in a C++ program using the demodulation code and CUDA event libraries to measure throughput and bit error rate (BER) of each module.

This document provides the details of the design of the Computer Software Configuration Item (CSCI), Benchmarking, for the High Speed GPU Demodulation System.

2. Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document #103,000, January 15, 2019, Rev F

4.3.2 Arrival Time: The system shall determine the time of the signal's arrival and first symbol.

4.3.3 Unknown Phase: The system shall determine the phase of the signal's first symbol.

4.3.4 Frequency Offset: The system shall operate in the complex baseband without a frequency offset.

4.3.5 Encoding: Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.

4.3.6 Demodulation Coherence: The signal's phase offset shall be adjusted to reference 0°.

4.3.7 Data Type: Data shall be processed as a floating point type.

4.4.1 Modulation Form: The system shall demodulate an 8 phase-shift keying (PSK) signal.

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

4.4.3 Packet Size: Demodulation packet sizes shall be no less than 2^{14} symbols.

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.

4.4.5 Throughput: The system shall be optimized to have a throughput of 1 Gb/s or above. The system should reach a throughput of 4.46 Gb/s.

3. Benchmarking Subsystem CSCI Design Decisions

The benchmarking software will input: the address for the test message and the directory for the demodulated messages and the benchmarking file containing the bit error rate for each noise level tested in the run.

The test message will be read and modulated into 8-PSK modulated data, and upsampled to go in a predefined packet size of at least 2^{14} 3-bit complex symbols. A predefined preamble is prepended to the packet and then a Root Raised Cosine (RRC) filter is applied to the entire packet.

To simulate a realistic channel, AWGN and a phase offset of 15+/- degrees was applied onto the packet of data before the synchronizer seeks the preamble. The packet before this noise and offset is applied on the chance that the preamble is not found, so the packet can be “resent” to the synchronizer with a newly applied noise vector and phase. This is repeated for each packet in the message file with 32 symbols of zero-padding around each packet. This process is repeated 20 times, each time applying a lower value of Eb/N0 (Energy per bit over noise power spectral density), with an increased level of AWGN applied to the transmitted message.

The software will then begin reading in each of the 20 modulated data files 16,384 symbols at a time, applying the AWGN filter and RRC filter. A script will then start calculating the difference between the phases in each adjacent symbol. The synchronizer, as described in section 5.2 of the Demodulation SDD #102,100, then finds the predefined preamble sequence. The location in the file directly following said start signal is then read into a packet of 16,384 symbols. That packet is down sampled, sliced, decoded using gray convention, and written to a local file in the demodulated messages directory. During each step of the demodulation process, an event timer will be used to determine the latency on the modules. These timed steps shall include but are not limited to: filtering out AWGN, filtering out RRC, decoding, and writing to local storage. This timing data will be written to a local file in the benchmarking directory, as well as an overall throughput, measured as Megabytes per second.

Once all 20 files of modulated data are demodulated and written to a file. A final script is run to calculate the bit error rate for each of the 20 received messages, using the original modulated message as a reference. The bit errors for each file will also be posted to the benchmarking results file.

The code will return to the caller, and the user will then be free to view the results in the benchmarking results directory, and/or view the messages in the received messages directory.

4. Benchmarking Subsystem CSCI Architectural Design

4.1 Benchmarking Components

4.1.1 File Parser

Modules are needed to interface with local storage.

4.1.2 Symbol Mapper

A module is needed to convert 3-bits of binary data to complex floating point data. The conversion follows the gray coding scheme specified in Figure 1 below.

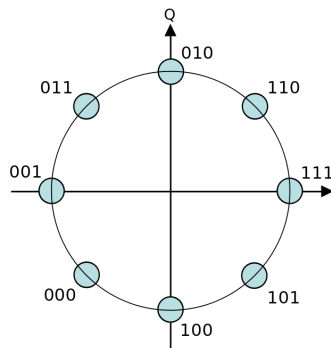


Figure 1. Gray Coding Scheme

4.1.3 Upsampler

The upsampler will zero pad the symbols to fill a packet of size 16,384 symbols.

4.1.4 Root Raised Cosine Filter

This filter is applied on both ends of the transmission: in the transmitter after the packet is assembled, and in the receiver after the packet is received.

4.1.5 Additive White Gaussian Noise Generator

To simulate transmission over a channel, white Gaussian noise is applied to the packets.

4.1.6 Phase Offsetter

A module is needed to apply a shift to the phase of of each symbol in a packet. This module will be called to simulate phase offset in transmission, and to adjust said offset when the receiver determines it.

4.1.7 Synchronizer

This module scans the received IQ data for a start condition to signify the beginning of a packet.

4.1.8 Downsampler

The downsampler removes the zero padding from a packet of symbols.

4.1.9 Slicer/Decoder

The slicer takes downsampled packets, and outputs the binary equivalent data, based on the gray coding scheme in Figure 1.

4.1.10 Event Timer

The time will be used to calculate the throughput of the demodulation. The timing information is exported to the results file, along with the overall throughput in Gigabytes per second.

4.1.11 Bit Error Rate Calculator

This BER calculator is a script that calculates the bit-wise differences between the original test message and the demodulated message.

4.2 System Flow Diagram

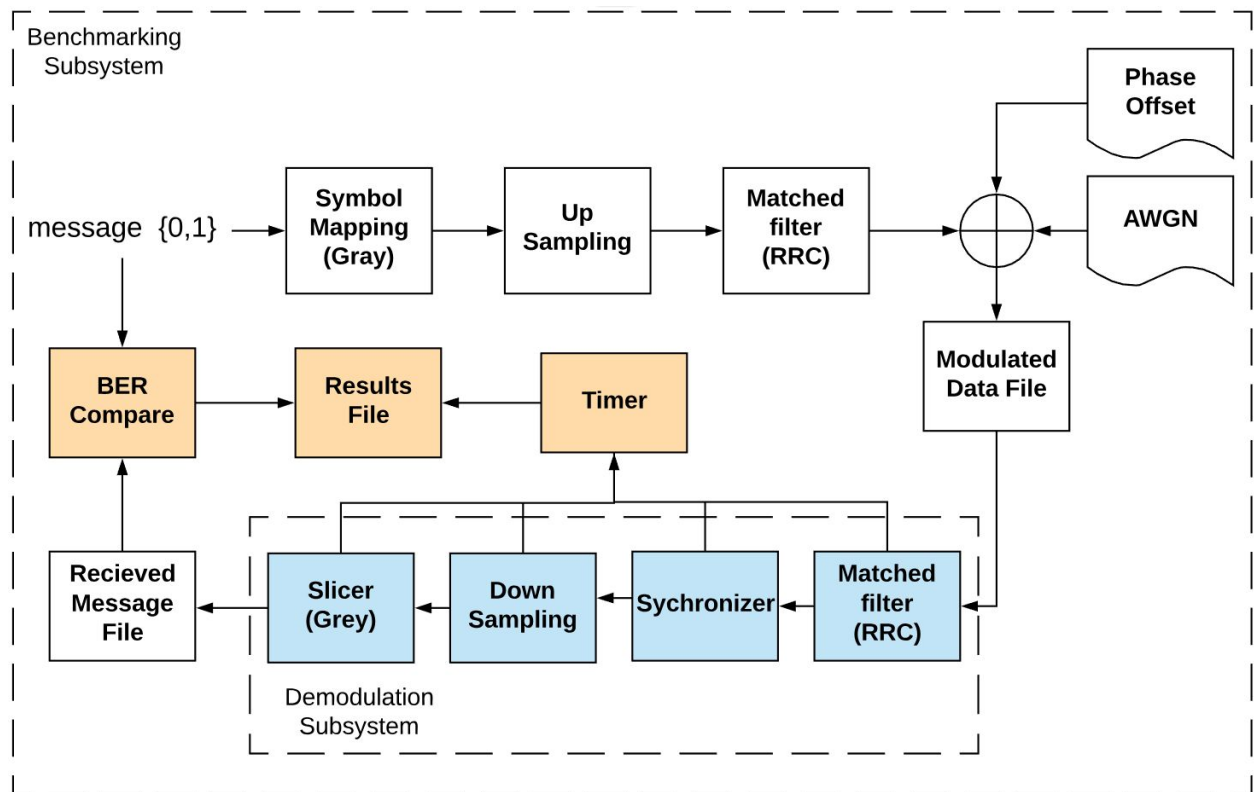


Figure 2. System Flow Diagram

5. Benchmarking Subsystem CSCI Detailed Design

5.1 File Parser

Data will be written to and read from files at a byte resolution, so binary I/O is a must.

5.2 Symbol Mapper

C++ is limited to byte-wise operations so symbols are mapped at a minimum of 3 bytes at a time; generating 8 3-bit 8-PSK symbols, based on the gray coding scheme in Figure 1; SRD requirement 4.3.5 Encoding and 4.4.1 Modulation form

8-PSK. These symbols shall be complex floating point pairs; SRD requirement 4.3.7 Data Type.

5.3 Upsampler

The upsampler shall zero pad to fill a packet of 2^{14} symbols; as requested in SRD 4.4.3 Packet Size.

5.4 Root Raised Cosine Filter

The module requires convolution, and as such a convolution module has been drafted as a sub-function to this filter.

5.5 Additive White Gaussian Noise Generator

The noise is generated using the C normal distribution library, given a mean and standard deviation for the noise, and the Mersenne Twister random number generator (RNG), also found in the C RNG library. The noise shall be uniform in frequency with a Gaussian distribution in time; SRD requirement 4.4.4 Additive Noise Type.

5.6 Phase Offseter

Since the data is complex floating point, adding 15 degrees phase offset can be done by rotating the complex vector 15 degrees, or multiplying the vector by a transformation matrix. The phase shall be adjusted for all symbols by the discovered offset to reference zero degrees; in accordance to SRD section 4.3.6 Demodulation Coherence.

5.7 Synchronizer

The file parser reads in a buffer of IQ data from local storage, and calculates the difference in phase between adjacent symbols. If the difference is found to be 135 degrees for 32 adjacent symbols, then a start condition is flagged; as requested in SRD 4.3.2. Given the known starting condition, the synchronizer will also determine the phase offset; SRD requirement 4.3.3 Unknown Phase.

5.8 Downsampler

The downsampler looks for the first symbol, in a 16384 symbol packet, to hit one of the expected 8 values, then grabs the rest of the symbols based on the upsample rate.

5.9 Slicer/Decoder

C++ is limited to byte-wise operations so symbols are decoded at a minimum of 8 symbols at a time; generating 3 bytes symbols. The decoding shall match the gray coding in Figure 1; SRD requirement 4.3.5.

5.10 Event Timer

Each module involved in demodulating will be timed with CUDA event timers. These timers determine how long a process took to execute. This timing data shall be used verify whether the system performs at 1GB/s throughput on the demodulation process; as explained in the SRD section 4.4.5 Throughput.

5.11 Bit Error Rate Calculator

The BER calculator compares byte by byte looking for each bit difference. The bit errors for each file are summed up respectively and posted to the results file. A minimum of 100 bit errors are needed from each run, and the BER must stay under 1dB above the theoretical BER; as stated in the SRD document sections 4.4.2 Implementation Loss.

6. Requirement Traceability

Traceability Matrix							
System Requirement	VM	Sub-Assembly					
		Demodulation Scripts	V M	Benchmarking	V M	NVidia GPU Platform	V M
4.1.1 Video Interface	I			BMK 1.0 Direct Flow	I	NVG 1.0 Direct Flow	I
4.2.1 CUDA enabled NVidia GPU	I	SDM 1.0 Direct Flow	I	BMK 2.0 Direct Flow	I	NVG 2.0 Direct Flow	I
4.2.2 Simulation Environment	I			BMK 3.0 Direct Flow	I		
4.3.1 Cost	I					NVG 3.0 Allocated \$3000	I
4.3.2 Arrival Time	D	SDM 2.0 Direct Flow	D				
4.3.3 Unknown Phase	D	SDM 3.0 Direct Flow	D				
4.3.4 Frequency Offset	I	SDM 4.0 Direct Flow	I				
4.3.5 Encoding	I	SDM 5.0 Direct Flow	I				
4.3.6 Demodulation Coherence	I	SDM 6.0 Direct Flow	I				
4.3.7 Data Type	I	SDM 7.0 Direct Flow	I	BMK 4.0 Direct Flow	I		
4.4.1 Modulation Form	I	SDM 8.0 Direct Flow	I				

4.4.2 Implementation Loss	T A	SDM 9.0 Direct Flow		BMK 5.0 Direct Flow	T A		
4.4.3 Packet Size	I	SDM 10.0 Direct Flow	I				
4.4.4 Additive Noise Type	I	SDM 11.0 Direct Flow	I				
4.4.5 Throughput	T A			BMK 6.0 Direct Flow	T A		

102,400 Matched Filter SDD

1.0 Scope

This document provides the detail of the design of the Computer Software Configuration Item (CSCI) Matched Filter for the High Speed Demodulation System. It defines the software libraries and expected inputs required for the user to interact with the matched filter subsystem. Additionally, the purpose and workflow of the subsystem is defined in order for adjustments to be made to the subsystem in order to improve bit error rate.

2.0 Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document, January 15, 2019, Rev F

4.3.7 Data Type: Data shall be processed as a floating point type.

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

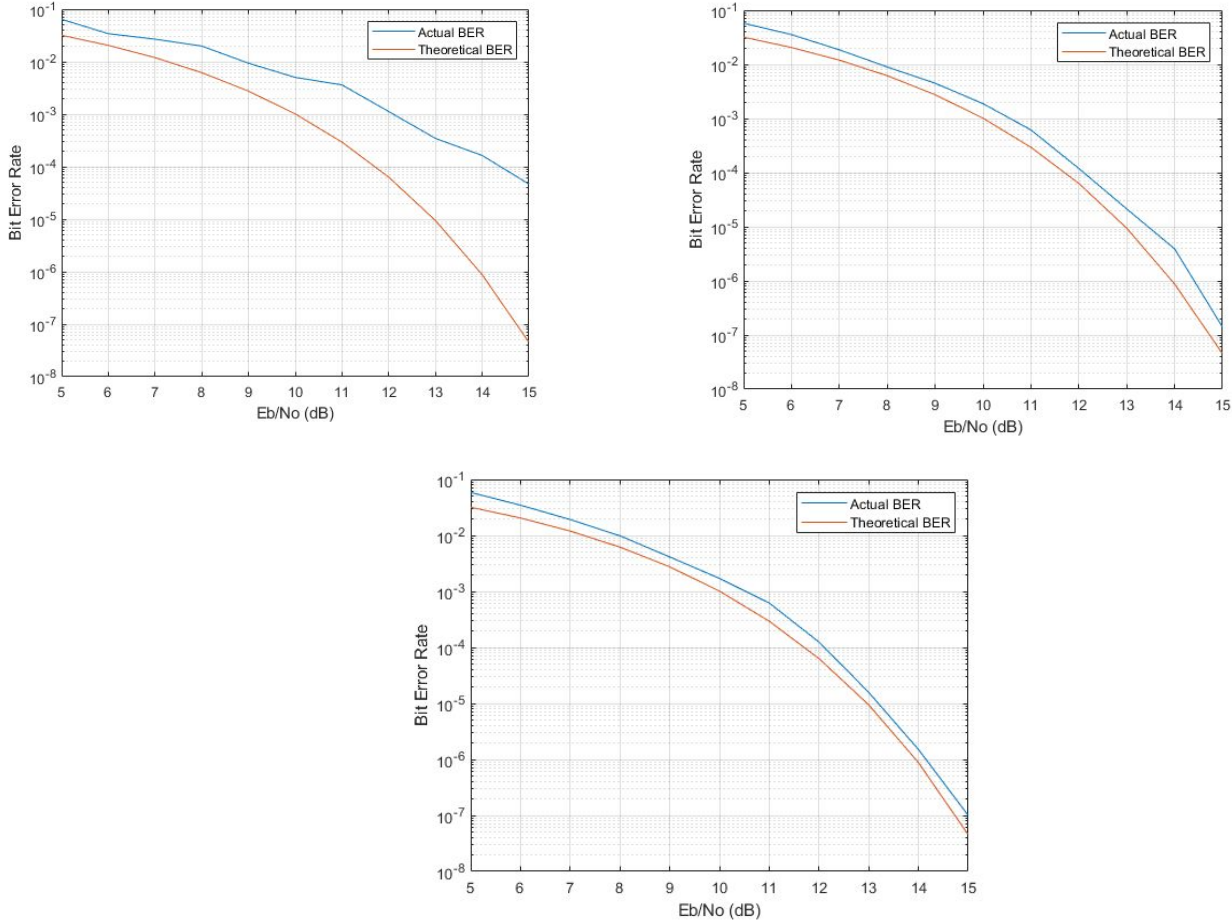
3.0 Matched Filter Subsystem CSCI Design Decisions

The system begins with a square root raised cosine (SRRC) function which takes as inputs floating point values for the oversample factor and roll off value as well as a pointer to an integer value for the size of the generated filter array. The function returns a floating point pointer to a generated array containing the filter. Then there is a convolution function which takes in two floating point array pointers, two integers denoting the size of those floating point arrays and an integer pointer for the size of the generated convolved array.

In order to optimize the speed of the convolution kernel written in CUDA, the coefficients of the matched filter was saved in constant device memory for the GPU to access. The coefficients were generated with MATLAB R2018b and the *rcosdesign* function defined by Simulink. There

were three parameters considered for the matched filter design: the excess bandwidth percentage known as the roll-off factor, the symbol span of the filter, and the filter samples per symbol.

To determine the best roll off factor for the SRRC filter, we simulated the BER curve for multiple values and observed that any roll off factor above 0.2 allowed the detector to achieve our desired performance according to SRD section 4.4.2. This is shown in the following figures:



From top left to bottom: Simulated BER curve with a roll off factor of 0.1, 0.2 and 0.3 respectively.

However, it is known that the roll off factor of a SRRC filter increases the bandwidth of the filter as it approaches 1. Therefore, to increase bandwidth efficiency, we chose not to design our filter with a high roll off factor. There are communications restrictions upon the bandwidth of the filter, therefore, a roll off factor of 0.3 was chosen in order to limit our bandwidth to be within bandwidth constraints. The symbol span was chosen to be 8 symbols in order to cancel as much symbols during convolution and pulse shaping. The samples per symbol must be 8 samples per symbol due to the upsampling rate to be the same frequency.

4.0 Matched Filter Subsystem CSCI Architectural Design

Below are the CSCs, or modules/views that are used in this CSCI, and their relationships. The logic and routines given for each CSC are to detail the design principles. The designer is free to make use of libraries to accomplish the same purposes.

Table 1 CSC Properties Included in the Matched Filter Subsystem CSCI				
CSC	Inputs (source)	Outputs (destination)	Purpose	Library
Conv	floating point value array array size x2 integer pointer	convolved array array size	Allow users to convolve any packet-sized array of floating point values with a SRRC filter.	cmath stddef stdlib string

5.0 Matched Filter Subsystem CSCI Detailed Design

The unit design decisions for each of the CSCs within the CSCI such as mockups, data handling and logic are given below.

5.1 SRRC Component

The SRRC handles making a square root raised cosine filter which can be applied as a matched filter to a given message at the transmitter and receiver in order to minimize intersymbol interference. This filter is created based on the oversample rate and roll off values inputted using the impulse response of a SRRC waveform as shown in Equation 1 and Equation 2 with the range of $-4T$ to $4T$. The oversample rate is used to determine distribution of sample points within the range, the increments will be based on the inverse of the oversample rate. This means that the size of the resulting filter array will always be the absolute value of the range multiplied by the oversample rate plus one to account for zero.

$$p(t) = \frac{\sin(\pi \cdot \frac{t}{T} \cdot (1-\alpha)) + 4 \cdot \alpha \cdot \frac{t}{T} \cdot \cos(\pi \cdot \frac{t}{T} \cdot (1+\alpha))}{\pi \cdot \frac{t}{T} \cdot (1-(4 \cdot \alpha \cdot \frac{t}{T})^2)}$$

Equation 1. Impulse Response of SRRC

$$p(t) = (1 - \alpha) + \frac{4 \cdot \alpha}{\pi}, \text{ when } t = 0$$

$$p(t) = \frac{\alpha}{\sqrt{2}} \left(\left(1 + \frac{2}{\pi}\right) \cdot \sin\left(\frac{\pi}{4 \cdot \alpha}\right) + \left(1 - \frac{2}{\pi}\right) \cdot \cos\left(\frac{\pi}{4 \cdot \alpha}\right) \right), \text{ when } \frac{t}{T} = \pm \frac{1}{4 \cdot \alpha}$$

Equation 2. Filter values at $t=0$ and $\frac{t}{T} = \pm \frac{1}{4 \cdot \alpha}$

5.2 Convolution Component

The convolution functions handles the mathematical convolution of any two arrays of floating point values. This is necessary in order to apply the matched filter to a given signal once it is in a floating point array form. The mathematical equation which defines the process is given below:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot g(t - \tau) d\tau$$

When applied however the bounds are not negative infinity to infinity but rather the start and end of the signal being convolved with the filter being applied. This means the resulting array will always have the size of the signal array plus the size of the filter array minus 1 to account for overlap.

6.0 Requirement Traceability

Table 7 Matched Filter Requirements Flow Down			
Subsystem Requirement	V M	Subassembly	
		Demodulation	Benchmarking
SDM 7.0 Direct Flow	I	X	X
SDM 9.0 Direct Flow	T,A	X	X

102,500 Slicer and Decoder SDD

1. Scope

This document provides the detail of the design of the Computer Software Configuration Item (CSCI) Slicer and Decoder Module of the demodulation software for the High Speed GPU Demodulation System. It defines the software modules required to take a given signal and separate it into an individual packet so that it can be converted into its base binary representation to be recompiled into its highest level (i.e. audio, video, etc.). This includes the necessary files to be ran by the user, the files which will be output by the system and how to interpret the data in the files which will be output by the system.

2. Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, User Interface Software Design Document #102,200, January 15, 2019, Rev A

Graphics Processing Unit Based High Speed Demodulation, System Requirement Document #103,000, January 15, 2019, Rev F

4.3.5 Encoding: Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.

4.4.3 Packet Size: Demodulation packet sizes shall be no less than 2^{14} symbols.

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.

3. Slicer and Decoder CSCI Design Decisions

3.1 Inputs and Outputs

This module will accept only two inputs: a 2-dimensional array consisting of in-phase and quadrature (IQ) data, and a file name to be printed to. The data stream will be filtered so that the majority of the noise has been filtered and will be converted into the final output string that contains the data after gray coding has been used to convert from IQ to binary. The only output of this system will be directly to the file defined in the UI (Document #102,200) and will consist of the binary data which has been decoded from the initial IQ data stream.

3.2 CSCI Behavior According to Inputs

Once the data has been received, the program will begin the “slicing” steps by taking the total packet of defined size and breaking it up into the actual data by separating out the redundant encoding. To do this, another array will be made with a total length of 1/8th the size of the packet since the initial size will be redundantly encoded 8 times to prevent errors. The downsampler will begin to take the 8th symbol of the signal starting from the last preamble symbol and create this smaller sized array. This contains the downsampled signal which will be decoded.

The decoding section is where the data will be converted from its initial IQ data stream to the associated binary values based on the gray coding constellation below.

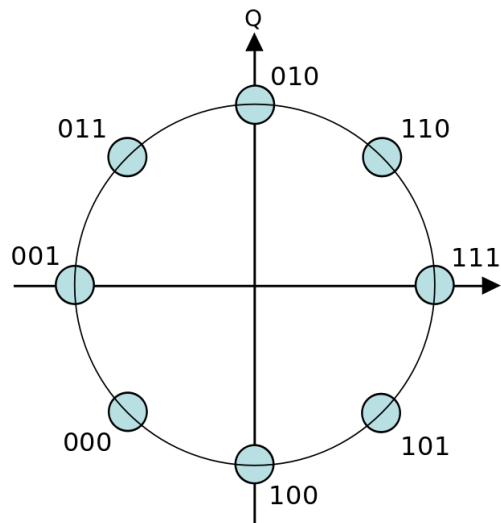


Figure 1. Gray Code

In this depiction, the vertical axis represents the quadrature (Q) value and the horizontal axis represents the in-phase (I) value. Since there is an allowable amount of noise, there will be thresholds for what values will correspond to each

point on the constellation. Initially, the threshold will be set to allow equal distance on either side of the center point (i.e. (1,0) will be assigned for values with angle $x \geq \frac{\pi}{8} > -\frac{\pi}{8}$), however this may change if future data suggests there is a more optimal threshold. Finally, once the data has been converted from it IQ form to the binary representation it will be written to the output file.

3.3 Requirement Based Decisions

The main requirements that influenced the decisions made can be seen in the attached SRD: 4.3.5, 4.4.3, 4.4.4 (Document #103,000). These requirements state that there must be an encoding scheme, a defined packet size, and additive white gaussian noise (AWGN) present in the system. With these in mind, the signal is redundantly encoded and gray coded, a packet size is defined for the signal, and a threshold for noise was added.

4. Slicer and Decoder CSCI Architectural Design Decisions

Below are the CSCs, or modules/views that are used in this CSCI, and their relationships. The details of each module are included in the design decisions section of this document.

Table 1: CSC Properties Included				
CSC	Inputs (source)	Outputs (destination)	Purpose	Library
IQ Stream Slicer	8-PSK Modulated IQ Stream (float*)	8-PSK Modulated/ Separated IQ Stream (decoder)	Removes redundantly encoded values to pass along the “best fit” values along to the decoder.	-complex<float> (C++ Library)
Sliced IQ Stream Decoder	8-PSK Modulated IQ stream (float*) Output File Name (UI)	Decoded Binary Data (file)	Converts the IQ data into the base binary form and writes it to the given file.	-complex<float> (C++ Library)

5. Slicer and Decoder CSCI Detailed Design

The slicer and decoder module will be implemented using C++ and the associated libraries listed in Table 1. The detail behind the design have been detailed in the design decision section of this document.

6. Requirement Traceability

Traceability Matrix							
System Requirement	VM	Sub-Assembly					
		Demodulation Scripts	V M	Benchmarking	V M	NVidia GPU Platform	V M
4.1.1 Video Interface	I			BMK 1.0 Direct Flow	I	NVG 1.0 Direct Flow	I
4.2.1 CUDA enabled NVidia GPU	I	SDM 1.0 Direct Flow	I	BMK 2.0 Direct Flow	I	NVG 2.0 Direct Flow	I
4.2.2 Simulation Environment	I			BMK 3.0 Direct Flow	I		
4.3.1 Cost	I					NVG 3.0 Allocated \$3000	I
4.3.2 Arrival Time	D	SDM 2.0 Direct Flow	D				
4.3.3 Unknown Phase	D	SDM 3.0 Direct Flow	D				
4.3.4 Frequency Offset	I	SDM 4.0 Direct Flow	I				
4.3.5 Encoding	I	SDM 5.0 Direct Flow	I				
4.3.6 Demodulation Coherence	I	SDM 6.0 Direct Flow	I				
4.3.7 Data Type	I	SDM 7.0 Direct Flow	I	BMK 4.0 Direct Flow	I		
4.4.1 Modulation Form	I	SDM 8.0 Direct Flow	I				
4.4.2 Implementation Loss	T A	SDM 9.0 Direct Flow		BMK 5.0 Direct Flow	T A		
4.4.3 Packet Size	I	SDM 10.0 Direct Flow	I				
4.4.4 Additive Noise Type	I	SDM 11.0 Direct Flow	I				
4.4.5 Throughput	T A			BMK 6.0 Direct Flow	T A		

103,000 Systems Requirement Document

1.0 Introduction and Scope

This project's goal is to develop an optimized demodulation system using a graphics processing unit (GPU) platform. This will involve surveying currently available GPUs on the market as suitable candidates and obtaining at least one GPU to perform the demodulation. The purpose of developing this system is to determine the viability of GPUs as a platform for performing high-speed demodulation and whether or not they can perform as well as current central processing unit (CPU) and application-specific integrated circuit (ASIC) based systems.

The message received is compared to the message expected bit-by-bit and this metric is referred to as the bit error rate (BER). Performance tests under additive white Gaussian noise (AWGN) shall be performed by graphing the BER with respect to the signal to noise ratio (SNR), and comparing that plot to the ideal theoretical models. The difference in SNR is known as implementation loss.

Using this collected data and metrics the overall system throughput will be optimized by altering the implemented algorithm. As well as allow the system to operate under the highest SNR possible while maintaining an acceptable BER.

2.0 Applicable and Other Referenced Documents

There are currently no applicable engineering documents referenced

3.0 Definitions, Acronyms and Abbreviations

- API - Application Programming Interface
- ASIC - Application-Specific Integrated Circuit
- BER - Bit Error Rate
- SNR - Signal to Noise Ratio
- AWGN - Additive White Gaussian Noise
- CPU - Central Processing Unit
- GB - Gigabyte
- GPU - Graphics Processing Unit
- SSD - Solid State Drive
- PCI-E - Peripheral Component Interconnect Express
- RAM - Random Access Memory
- TBD - To Be Determined
- PSK- Phase-Shift Keying

Constellation Diagram - This is a representation of a signal modulated by a digital modulation scheme such as 8-phase-shifting keying. The diagram plots the signal on an xy plane in which the angle of the scatter point from 0° represents the phase shift of the carrier wave and the distance of the point from the origin is the magnitude of the signal.

Complex Baseband - System bandwidth will operate around DC frequencies, with energy in both positive and negative frequencies.

4.0 Requirements

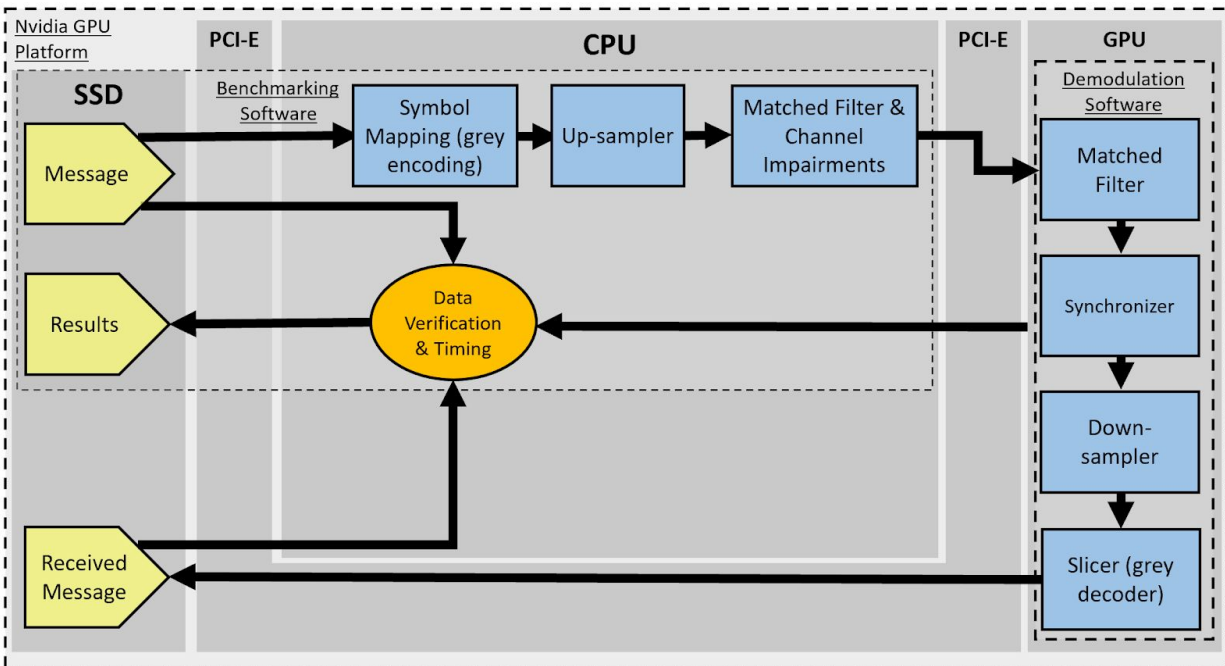


Figure 4.1 System Block Diagram

4.1 Interface Requirements

4.1.1 Video Interface: The demodulation software shall be accessed and executed through a terminal interface on a Linux system.

4.2 Environmental Requirements

4.2.1 CUDA enabled NVidia GPU: The software shall run on an NVidia GPU equipped computer.

4.2.2 Simulation Environment: Simulations shall be performed using complex baseband to represent the signal being demodulated.

4.3 Customer Constraints

4.3.1 Cost: The project shall not exceed the \$4,000 budget.

4.3.2 Arrival Time: The system shall determine the time of the signal's arrival and first symbol.

4.3.3 Unknown Phase: The system shall determine the phase of the signal's first symbol.

4.3.4 Frequency Offset: The system shall operate in the complex baseband without a frequency offset.

4.3.5 Encoding: Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.

4.3.6 Demodulation Coherence: The signal's phase offset shall be adjusted to reference 0° .

4.3.7 Data Type: Data shall be processed as a floating point type.

4.4 Performance Requirements

4.4.1 Modulation Form: The system shall demodulate an 8 phase-shift keying (PSK) signal.

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

4.4.3 Packet Size: Demodulation packet sizes shall be no less than 2^{14} symbols.

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which

performance is below the acceptable implementation loss.

4.4.5 Throughput: The system shall be optimized to have a throughput of 1 Gb/s or above. The system should reach a throughput of 4.46 Gb/s.

5.0 Verification Requirements

5.1 Interface Requirements

5.1.1 Video Interface: Verified by inspecting the results through the display monitor.

5.2 Environmental Requirements

5.2.1 CUDA enabled NVidia GPU: Inspect the running system for for an NVidia GPU.

5.2.2 Simulation Environment: Simulation environment will verified by test and analysis to ensure the system operates with no added frequency offset.

5.3 Customer Constraints

5.3.1 Cost: The system cost shall be verified through inspection.

5.3.2 Arrival Time: The system shall determine the start of the signal's arrival, verified by demonstration.

5.3.3 Unknown Phase: The system shall demonstrate that the phase has been found.

5.3.4 Frequency offset: Baseband shall be verified by testing the system and analyzing the bandwidth, ensuring there is no offset.

5.3.5 Encoding: The system shall be tested and analyzed to verify that gray encoding is assumed.

5.3.6 Demodulation Coherence: Symbols shall be verified by testing and analysis to be coherently received after transmission by confirming the phase at the transmitter is the same phase the signal has in the receiver.

5.3.7 Data Type: Code shall be inspected to verify floating point arithmetic.

5.4 Performance Requirements

- 5.4.1 Modulation Form: The modulation form shall be verified by testing the symbol mapping module and analyzing the constellation diagram, of a unimpaired signal, for 8 distinct phases.
- 5.4.2 Implementation Loss: Implemented bit error rate curve shall be graphed from 1×10^{-3} to 1×10^{-6} BER and compared to the ideal curve. A minimum of 100 errors is required for each point on the plot.
- 5.4.3 Packet Size: Packet sizes of 2^{14} symbols shall be verified by inspecting the code.
- 5.4.4 Additive Noise Type: Noise signal shall be tested and analyzed to have uniform power distribution in the frequency domain and normal distribution of noise in the time domain.
- 5.4.5 Throughput: Throughput shall be measured as memory read and written in bytes over time taken to read and write. The throughput shall be larger than 1 Gb/s, which shall be verified by analysis.

6.0 System Requirement Verification Matrix

	Verification Method			
	T	A	D	I
4.1 Interface Requirements	-	-	-	-
4.1.1 Video Interface				I
4.2 Environmental Requirements	-	-	-	-
4.2.1 CUDA enabled NVidia GPU Platform				I
4.2.2 Simulation Environment	T	A		
4.3 Customer Constraints	-	-	-	-
4.3.1 Cost				I
4.3.2 Arrival Time			D	
4.3.3 Unknown Phase			D	
4.3.4 Frequency Offset	T	A		
4.3.5 Encoding	T	A		
4.3.6 Demodulation Coherence	T	A		
4.3.7 Data Type				I
4.4 Performance Requirements	-	-	-	-
4.4.1 Modulation Form	T	A		
4.4.2 Implementation Loss	T	A		
4.4.3 Packet Size				I
4.4.4 Additive Noise Type	T	A		
4.4.5 Throughput	T	A		

7.0 Notes

7.1 Test: Compliance with requirements is validated by evaluating or executing an item under controlled conditions, configurations, and inputs to observe the response as specified by the requirements. Results are quantified and analyzed.

7.2 Analysis: Compliance with requirements is determined by models or by interpreting results using established technical or mathematical models or simulations, analysis, or other scientific principles and procedures to provide evidence that that item meets its stated requirements. Analysis allows someone to make predictive statements about the typical performance of a product or system.

7.3 Demonstration: Compliance with requirements is validated by observing the item in operation.

7.4 Inspection: Compliance with requirements is determined by formal examination using one or more of the five senses, simple physical manipulation and mechanical/electrical gauging and measurement to verify that the item conforms to its specified requirements.

4.0 Acceptance Test Procedure

The systems requirements will be verified by three acceptance tests:

1. Implementation Loss and Throughput Acceptance Test
2. Simulation Environment Acceptance Test
3. AWGN, Decoder, And Coherence Acceptance Test

Table 2: Requirements and Acceptance Tests	
System Requirement	Acceptance Test Procedures
SRD 4.2.2 - Simulation Environment	ATP 2 - Simulation Environment Acceptance Test
SRD 4.3.4 - Frequency Offset	ATP 2 - Simulation Environment Acceptance Test
SRD 4.3.5 - Encoding	ATP 3 - AWGN, Decoder, And Coherence Acceptance Test

SRD 4.3.6 - Demodulation Coherence	ATP 3 - AWGN, Decoder, And Coherence Acceptance Test
SRD 4.4.1 - Modulation Form	ATP 3 - AWGN, Decoder, And Coherence Acceptance Test
SRD 4.4.2 - Implementation Loss	ATP 1 - Implementation Loss and Throughput Acceptance Test
SRD 4.4.4 - Additive Noise Type	ATP 3 - AWGN, Decoder, And Coherence Acceptance Test
SRD 4.4.5 - Throughput	ATP 1 - Implementation Loss and Throughput Acceptance Test

101,000 Acceptance Test Procedure

1.1 Implementation Loss and Throughput Acceptance Test

1.2 Introduction

This procedure outlines the test that must be conducted on the demodulation system to calculate the loss of performance and the throughput of the system during demodulation. The bit error rate (BER) of the demodulated signal when compared to the actual signal is a measurable characteristic of the system. This test involves adding a range of noise, measured in the signal-to-noise ratios (*sig.* SNR), into the system before the signal is demodulated. The demodulated signal is then compared to the actual signal and for each SNR tested, the BER is plotted.

1.3 Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document, November 26, 2018, Rev E

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

4.4.5 Throughput: The system shall be optimized to have a throughput of 1 Gb/s or above.
The system should reach a throughput of 4.46 Gb/s.

1.4 Required Test Equipment

The Graphics Processing Unit Based High Speed Demodulation system's benchmarking subsystem will perform the test. Therefore, no additional test equipment is required outside the system deliverables.

1.5 Table of Tests

Test	Requirement
BER vs. Signal-To-Noise Ratio	No greater than 1 dB difference between theoretical and actual between BER of 1×10^{-6} through 1×10^{-3}
Timing Analysis and Bit Tracking	Must process at least 1 Gb of data within an average of 1 second to achieve required throughput. Optimally should achieve 4.46 Gb/s

1.6 Step Procedure

1. Configure system according to Assembly Drawing #100,000.
2. Ensure that the Demodulation system software package has been properly installed by running the system executable via command line.
3. Upon running the executable and viewing the user interface, select the option to perform the demodulation of the signal and include the benchmarking results by providing an output directory for the results file.
4. Select the data file that contains the binaries of the modulated signal.
5. After the demodulation process has finished executing, the results will be available to the user via data file.

2.1 Simulation Environment Acceptance Test

2.2 Introduction

This procedure outlines the test that must be conducted on the demodulation system to determine that the digital receiver has a zero frequency offset. All receivers have an

inherent frequency offset, but the demodulation subsystem must perform frequency offset calibration to compensate for this offset. This is a test involves running the full system and analyzing the demodulated signal.

2.3 Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document, November 26, 2018, Rev E

4.2.2 Simulation Environment: Simulations shall be performed using complex baseband to represent the signal being demodulated.

4.3.4 Frequency Offset: The system shall operate in the complex baseband without a frequency offset.

2.4 Required Test Equipment

The Graphics Processing Unit Based High Speed Demodulation system's demodulation subsystem will perform the test. MATLAB R2018b and the MATLAB Communications Toolbox will perform the signal frequency spectrum analysis. Specifically, the signal will be printed to a Comma-Separated Values (CSV) file after the root-raised cosine filter on the transmitter side of the system. We utilize an Fast-Fourier Transform based algorithm implemented by Simulinks, referenced as the Coarse Frequency Compensator, to calculate the estimated frequency offset.

2.5 Table of Tests

Test	Requirement
Signal Frequency Spectrum Analysis	Frequency offset of 0 kHz with a 10% tolerance

2.6 Step Procedure

1. Configure system per Assembly Drawing #100,000.
2. Ensure that the Demodulation system software packet has been properly installed by opening the system executable.
3. Upon opening the executable and viewing the user interface, select the option to perform the demodulation of the signal.
4. Select the data file that contains the raw bits of the signal.

5. After the demodulation process has finished executing, the results will be available to the user via data file.
6. Process the results file as an array input in MATLAB R2018b and use the Coarse Frequency Compensator to calculate the estimated offset.

3.1 AWGN, Decoder, and Coherence Acceptance Test

3.2 Introduction

This procedure outlines the test that must be conducted on the Decoder and Slicer subsystem to determine that the demodulation technique is 8-PSK, the phase of the complex baseband stream is adjusted to 0° , and the 8-PSK phases are correctly corresponding to their grey-coded value. The decoding of the subsystem is the bulk of the demodulation workload and therefore must have additional unit testing in place.

3.3 Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document, November 26, 2018, Rev E

4.3.5 Encoding: Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.

4.4.1 Modulation Form: The system shall demodulate an 8 phase-shift keying (PSK) signal.

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.

3.4 Required Test Equipment

The Graphics Processing Unit Based High Speed Demodulation system's decoder and slicer subsystem will perform the test. The unit testing module corresponding to this subsystem is included in the source code in addition to other unit tests. Therefore, no additional test equipment is required outside the system deliverables.

3.5 Table of Tests

Test	Requirement
Grey-Coding Test	Given a 8-PSK state, the decoder must determine the correct grey-code pair
AWGN Generator Test	The designed noise generator must be additive, white, and Gaussian in distribution

3.6 Step Procedure

1. Configure system per Assembly Drawing #100,000.
2. Ensure that the Demodulation system software packet has been properly installed by opening the system executable.
3. With access to the source code, compile and run the Tests software package with any C/C++ compiler, preferably Visual Studio 2017. The software package will run all unit tests and display results in the output module.

5.0 Models / Analyses

The System Model documents can be found in Appendix A.

6.0 Acceptance Test Results

Implementation Loss and Throughput Acceptance Test Datasheet

Referenced ATP Paragraph Number: 1.1

Name of Test: SRD 4.4.2 Implementation Loss & SRD 4.4.5 Throughput

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

4.4.5 Throughput: The system shall be optimized to have a throughput of 1 Gb/s or above. The system should reach a throughput of 4.46 Gb/s.

Unit Under Test (UUT):

Name: Benchmarking Subsystem

Part Number: 102,100

Serial Number: N/A

Results (Pass / Fail): **Pass - Implementation Loss, Fail - Throughput**

Date of Test: 5/3/2019

Recording of Test

Measurement:

- Largest BER curve difference (between BER of 1×10^{-3} and 1×10^{-6}) = 0.6 dB
- Total bits of data processed = 23,167,056 bits
- Total demodulation time = 175407 us

Requirement (SRD, with

Tolerances):

- BER difference < 1 dB + 0% tolerance
- Total Gb / Total time \geq 1 Gb/s

Test

Equipment

Error: N/A

Adjusted Test Limit:

N/A

Computations, (Include Analyses Results, if any): Effective throughput: 132.076 Mb/s

Signatures:

Tester

Customer

Taul Aragaki

Simulation Environment Acceptance Test Datasheet

Referenced ATP Paragraph Number: 2.1

Name of Test: SRD 4.2.2 Simulation Environment and SRD 4.3.4 Frequency Offset

4.2.2 Simulation Environment: Simulations shall be performed using complex baseband to represent the signal being demodulated.

4.3.4 Frequency Offset: The system shall operate in the complex baseband without a frequency offset.

Unit Under Test (UUT):

Name: Demodulation Subsystem - Benchmarking, Matched Filter, Noise Filter, Decoder and Slicer Modules

Part Number: 102,200; 102,300; 102,400; 102,500

Serial Number: N/A

Results (Pass / Fail): **Pass**

Date of Test: 4/18/2019

Recording of Test Measurement:

Frequency of maximum power: 0 kHz

Requirement (SRD, with Tolerances):
= 0 kHz with no tolerance

Test Equipment

Error: N/A

Adjusted Test Limit:

N/A

Computations, (Include Analyses Results, if any): N/A

Signatures:

Tester

Customer

Taul Aragaki

Decoder and Coherence Acceptance Test Datasheet

Referenced ATP Paragraph Number: 3.1

Name of Test: SRD 4.3.5 Encoding, SRD 4.3.6 Demodulation Coherence, SRD 4.4.1 Modulation Form, and SRD 4.4.4 Additive Noise Type

4.3.5 Encoding: Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.

4.4.1 Modulation Form: The system shall demodulate an 8 phase-shift keying (PSK) signal.

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.

Unit Under Test (UUT):

Name: Decoder and Slicer Module and the Additive White Gaussian Noise module

Part Number: 102,500 (AWGN module not included in Graphics Processing Unit Based High Speed Demodulation system)

Serial Number: N/A

Results (Pass / Fail): **Pass**

Date of Test: 4/22/2019

Recording of Test Measurement:
 Unit Test Passes: 2 AWGN, 2 Decoder
 = 4 tests
 Unit Test Fails: 4 tests

Requirement (SRD, with Tolerances):
 Must pass all unit testing

Test Equipment Error: N/A

Adjusted Test Limit: N/A

Computations, (Include Analyses Results, if any): N/A

Signatures:

Tester _____

Customer Taul Aragaki

7.0 Final Budget

In total, the entire GPU system (including the backup) cost almost \$3,400. The breakdown of the cost of a single GPU platform can be found in the bill of materials under Document #104,100 in Appendix A.

Item	Cost
Nvidia GPU Hardware	\$3,399.95
Polos	\$222.50
Poster Design	\$90.00
Total:	\$3,712.45

8.0 Lessons Learned

Technical Lessons

Demodulation

The task of implementing a software-defined receiver, along with a full benchmarking platform, allowed our team to apply the skills included in the University of Arizona curriculum to build a marketable product for General Dynamics. The implementation involved many advanced digital signal processing and digital communications concepts such as applying a filter through convolution, designing a sync sequence and phase differential synchronizer, and building a maximum likelihood detector for an 8-PSK signal.

Other lessons learned involved testing procedures and how to properly debug a complex subsystem. For example, debugging the matched filter caused an unexpected schedule delay. In order to test whether the filter response was correct, our team researched the impulse response that we were to expect. Instead of relying solely on MATLAB to confirm our expectations, we familiarized ourselves with the parameters and behaviors of the filter and the convolution application. This is similar to when we were finding synchronization issues. Our original preamble was poor in design, but was chosen due to the fast, simple calculations that signaled the preamble was found. The symbols correlated with each other too closely, however, and false

preambles would be found, disrupting the decoding process. Instead, we generated a sequence with a better autocorrelation characteristic than repetitive symbols and immediately found better performance.

GPU Development

Graphics processing unit based development is still a blossoming research field with plenty of fruitful results and optimization resources. However, the optimization portion of our software, as evident by our failed requirement of SRD section 4.4.5, was the most challenging task in our design project. This is mostly due to the fact that CUDA, while well documented with NVIDIA coding samples, is not as versatile when a sample does not fit your particular needs. Most of our kernels were not taken from any samples and we learned how to unroll for loops into independent threads in order to parallelize our code initially. As expected, the speed up from these basic kernels was indeed notable, but not enough for this project to achieve our ultimate throughput goal. Even with a greater packet size, attempts to stream host memory to the device memory during execution, and implementing other more advanced CUDA techniques, we were not able to achieve our desired performance and throughput at once.

Project Organization Lessons

With over 20,000 lines of code, multiple iterations of each subsystem, and over 100 Git commits to a large repository, it was essential for the team to learn quickly how to manage complex software. This was done by laying out ground rules regarding both in-person and online development behavior. These included: booking meeting rooms on campus at least a day in advanced, enforcing comments on each function or large portion of code for better readability, and required peer reviews for small code changes before the changes are implemented into the master source code. This allowed us to learn online tools that encourage accountability and aid in managing issues in an active software product.

Conclusion

In conclusion, our design performs high speed demodulation on the RTX 2080 graphics card with impressive implementation loss and an effective throughput that is still well over our CPU speeds. Demodulation of about 26 million bits over a clean channel on our Intel i7-8700K core processor finished with an effective throughput of 2.38 Mb/s. After the implementation of our proposed design, our GPU based system resulted in an effective throughput of 132.076 Mb/s. Compared to the larger costs of ASIC and FPGA solutions and the bottlenecked throughput of traditional CPU systems, this design exceeded initial expectations on the viability of GPUs in digital communications. With this design, General Dynamics will be able to continue their research in GPU optimization techniques and implement general high speed demodulation for any other modulation scheme.

System Requirement Verification				
Requirement	Method	Limit/Reference	Measured/Ref Value	Pass/Fail
1. Interface Requirements	-	-	-	-
a. <u>Video Interface</u> : The demodulation software shall be accessed and executed through a terminal interface on a Linux system.	I	Interface configured per SDD # 102,200 User Interface	n/a	Pass
2. Environmental Requirements	-	-	-	-
a. <u>CUDA enabled NVidia GPU Platform</u> : The software shall run on an NVidia GPU equipped computer.	I	System assembled per Doc # 100,000 Top Assembly Drawing	n/a	Pass
b. <u>Simulation Environment</u> : Simulations shall be performed using complex baseband to represent the signal being demodulated.	T, A	0Hz Frequency offset SDD # 102,100 Demodulation Script ATP #2.1 Simulation Environment Test		Pass
3. Customer Constraints	-	-	-	-
a. <u>Cost</u> : The project shall not exceed the \$4,000 budget.	I	< \$4,000	\$3339.42	Pass
b. <u>Arrival Time</u> : The system shall determine the time of the signal's arrival and first symbol.	D	Code built per Doc # 102,100 Demodulation Script SDD		Pass
c. <u>Unknown Phase</u> : The system shall determine the phase of the signal's first symbol.	D	Code built per Doc # 102,100 Demodulation Script SDD		Pass
d. <u>Frequent Offset</u> : The system shall operate in the complex	T, A	0Hz Frequency offset		Pass

baseband without a frequency offset.		Doc # 102,100 Demodulation Script SDD ATP #2.1 Simulation Environment Test		
e. <u>Encoding</u> : Symbols shall be formatted such that binary representations of successive symbols shall differ by one binary digit.	T, A	Doc # 102,500 Decoder and Slicer SDD ATP #3.1 AWGN, Decoder, and Coherence Test		Pass
f. <u>Demodulation Coherence</u> : The signal's phase offset shall be adjusted to reference 0°.	T, A	Phases must be adjusted by phase offset as per Slicer and Decoder SDD # 102,500		Pass
g. <u>Data Type</u> : Data shall be processed as a floating point type.	I	Symbols must be floating point as per all Software Design Documents Doc #102,100 - #102,500	n/a	Pass
4. Performance Requirements	-	-	-	
a. <u>Modulation Form</u> : The system shall demodulate an 8 phase-shift keying (PSK) signal.	T, A	Symbols must be 8-PSK encoded as per Slicer and Decoder SDD # 102,500		Pass
b. <u>Implementation Loss</u> : The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .	T, A	Less than 1DB error on the BER curve between 1×10^{-3} to 1×10^{-6} as per Benchmarking SDD # 102,300		Pass
c. <u>Packet Size</u> : Demodulation	I	2^{14} symbol packets	n/a	Pass

packet sizes shall be no less than 2^{14} symbols.		as per all Software Design Documents Doc #102,100 - #102,500		
d. <u>Additive Noise Type</u> : The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.	T, A	Noise must be white and gaussian as per Benchmarking SDD # 102,300		Pass
e. <u>Throughput</u> : The system shall be optimized to have a throughput of 1 Gb/s or above. The system should reach a throughput of 4.46 Gb/s.	T, A	Throughput of at least 1Gb/s is required as per Demodulation SDD #102,100	132.076 Mb/s	Fail

Appendix A - Models and Analyses Documents

104,000 System Models

104,100 Cost Model

The cost model is simply the expenses, purpose of expenses, and remaining balance of the \$4,000 SRD (Document #103,000)

Category	Product	Cost	Need Date	Delivery Date
GPU	RTX 2080	\$749.99	12/11	12/4
CPU	Intel i7- 8700K	\$339.99	12/11	12/4
CPU Cooler	Cooler Master - MasterLiquid ML240L	\$59.99	12/11	12/4
Thermal Compound	Arctic Silver - 5 High-Density Thermal Paste	\$6.65	12/11	12/4
RAM	Corsair Vengeance LPX 32GB DDR4	\$284.99	12/11	12/4
SSD	Samsung Evo 860 250GB	\$55.99	12/11	12/4
2nd SSD	Samsung 860 EVO 250GB M.2	\$67.99	12/11	12/4
Motherboard	MSI Z370-A Pro	\$99.99	12/11	12/4
Case	Corsair Carbide Series 275R	\$69.99	12/11	12/4
Case Fans	Noctua NF-P14s (x2), Kingwin CFR-012LB	\$14.95 (x2) + \$7.93	12/11	12/4
Monitor	Acer G226HQLBdb 21.5"	\$76.49	12/11	12/4
Keyboard	Verbatim 99201 Wired Standard Keyboard	\$4.29	12/11	12/4
Mouse	Logitech M100 Wired Optical Mouse	\$4.44	12/11	12/4
Power Supply	Cooler Master 750 W 80+ Gold Certified	\$123.99	12/11	12/4
Total		\$ 1,982.61		

Table 1. System budget for one testbench GPU platform

Sub-System	Expected Cost	Cost to Date	Outstanding
NVidia GPU Platform	\$3,363.70	\$3,363.70	\$0
Demodulation Software	\$0	\$0	\$0
Benchmarking Software	\$0	\$0	\$0
Misc Purchases	\$500	\$0	\$500
Total	\$3,863.70	\$3,363.70	\$500

Table 2. System budget expenses and outstanding balance

104,200 Bandwidth Model

Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document #103,000, January 15, 2019, Rev F

4.2.2 Simulation Environment: Simulations shall be performed using complex baseband to represent the signal being demodulated.

4.3.4 Frequency Offset: The system shall operate in the complex baseband without a frequency offset.

Purpose

The system must operate in the complex baseband with no frequency offset. To test that the system meets this requirement, the signal, once passed through the receiver, must be centered around zero.

Description

The model used to measure the frequency baseband is the Fourier Transform.

Model(s)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \quad k = 0, \dots, N-1$$
$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi kn/N} \quad n = 0, \dots, N-1$$

Figure 1. The Discrete-Time Fourier Transform

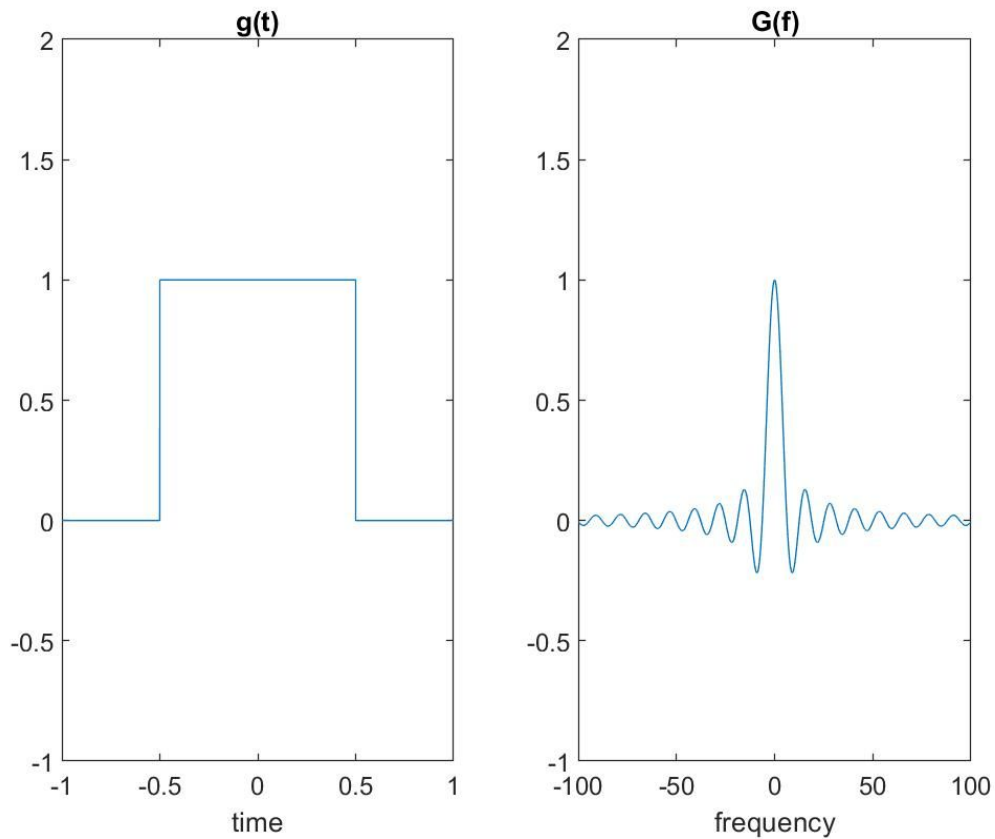


Figure 2. Complex Spectrum Graphs

104,300 Modulation Model

Referenced Documents

4.4.1 Modulation Form: The system shall demodulate an 8 phase-shift keying (PSK) signal.

Purpose

The form of modulation defines the processes that the receiver must perform in order to demodulate and translate the signal.

Description

The modulation model that adheres to the Requirement 4.4.1 is the 8-Phase Shift Keyed modulation constellation. This constellation represents the complex numbers, which are unit vectors with eight uniformly spaced phases, the encoded data is received by our system.

Model(s)

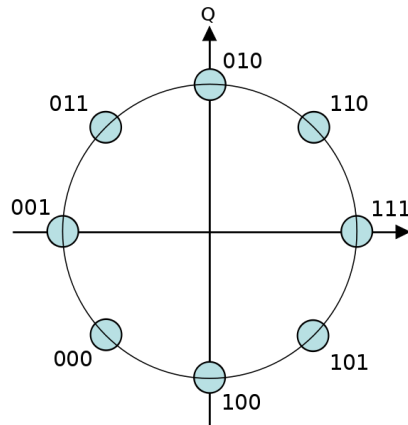


Figure 1. 8-PSK constellation and the equivalent grey code symbols for each position

104,400 BER vs. Signal Noise Ratio Model

Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements
Document #103,000, January 15, 2019, Rev F

4.4.2 Implementation Loss: The difference between the calculated theoretical maximum performance curve and the actual performance curve of bit error rate (BER) vs signal to noise ratio shall be no greater than 1 dB at an un-encoded BER of 1×10^{-6} .

Purpose

Accuracy is required for reliable transmission. As noise is added bits will be miss-read, and this software shall transmit under 1dB above the theoretical minimum error; SRD requirement 4.4.2.

Description

The curve is measured by counting the ratio of incorrect to correctly transmitted bits for increased channel noise levels, where channel noise is measured as a signal to noise ratio. (See Figure 1).

Model(s)

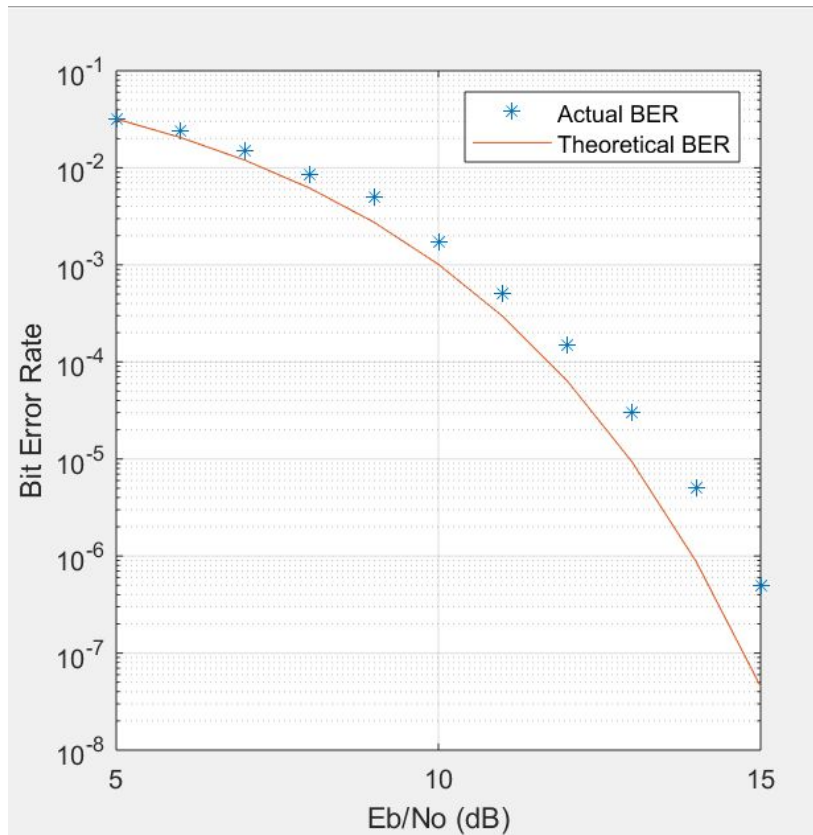


Figure 1. BER v. SNR Curve

104,500 Gaussian Noise Model

Referenced Documents

Graphics Processing Unit Based High Speed Demodulation, System Requirements Document #103,000, January 15, 2019, Rev F

4.4.4 Additive Noise Type: The system shall use additive white gaussian noise (AWGN) to provide channel impairment from zero to the point at which performance is below the acceptable implementation loss.

Purpose

The Additive White Gaussian Noise that is required by the System Requirement Document (Document #103,000) Requirement 4.4.4. is essential to testing the implementation loss of the demodulation system. The noise must adhere to this requirement in order to simulate a realistic channel impairment that a software defined radio receiver will acquire a signal from.

Description

The models define the constraints of our noise in the three main areas: 1) the noise is additive, 2) the noise is uniform in power in the frequency domain (termed “white” noise), and 3) the noise has a Gaussian distribution.

Model(s)

The signal Fourier Transform defines the frequencies represented in the signal (Figure 1), while the Power Spectral Density (PSD) defines the power density in the frequency domain. In order for the noise to be uniform in power, the PSD must be one unit.

$$\hat{x}(\omega) = \frac{1}{\sqrt{T}} \int_0^T x(t) e^{-i\omega t} dt.$$

Figure 1. Signal Fourier Transform Model

$$S_{xx}(\omega) = \lim_{T \rightarrow \infty} \mathbf{E} \left[|\hat{x}(\omega)|^2 \right]$$

Figure 2. Power Spectral Density

The Chi-Square goodness-of-fit test verifies whether the noise distribution is Gaussian via the equation in Figure 3.

$$\chi^2 = \sum \frac{(\text{observed count} - \text{expected count})^2}{\text{expected count}}$$

Figure 3. Chi-squared goodness-of-fit

104,600 Throughput Model

Referenced Documents

[1] NVidia Turing GPU Architecture, WP-09183-001_v01, 2018. pg 59, Table 4.

[2] Graphics Processing Unit Based High Speed Demodulation, System Requirements Document #103,000, January 15, 2019, Rev F

4.4.5 Throughput: The system shall be optimized to have a throughput of 1 Gb/s or above. The system should reach a throughput of 4.46 Gb/s.

Description

In this project throughput is defined as the effective bandwidth, which is bytes written and read over time elapsed: see Figure 2.

Purpose

This software has been developed to utilize the high data throughput of graphics processing unit cards. The bandwidth must be modeled and tested to verify the performance required for this software. The card has a theoretical maximum bandwidth of 448 GB/s[1], and this software shall reach at least 1 GB/s; SRD 4.4.5 Throughput.

Model(s)

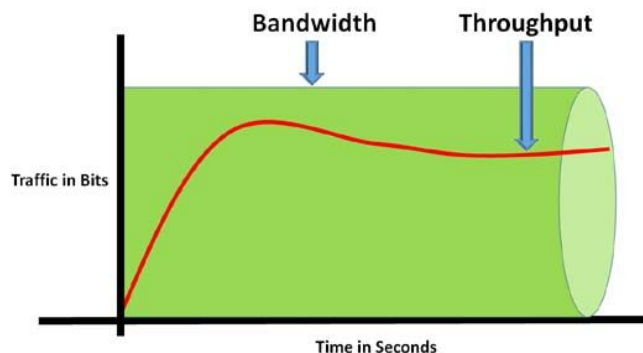


Figure 1. Throughput figure

$$BW_{Theoretical} = \frac{(mem_clkrate)(mem_width_inbytes)}{10^9} \frac{GB}{s}$$

$$BW_{Effective} = \frac{(bytes_read + bytes_written)}{(time_elapsed) * 10^9} \frac{GB}{s}$$

Figure 2. Bandwidth Equations

Roles and Responsibilities Matrix

Name	Role	Responsibilities	Deliverables
*Cathy McIntosh	Software Engineer	<ul style="list-style-type: none"> • Design and implement benchmarking subsystem for implementation loss • Design and implement unit tests for all modules of the demodulation process • Design the Acceptance Test Procedures • Simulate and implement the square-root raised cosine (SRRC) filter in both MATLAB and CUDA/C++ • Design and implement the noise generation in order to simulate AWGN channel • Design and implement synchronizer with unique sync sequence • Initialize the repository and maintain peer review checks and project issues in Git • Organize the testing project in parallel with source code 	ATP Document #101,000 Demodulation Unit Tests AWGN Unit Tests Matched Filter Unit Tests Benchmarking subsystem AWGN Generator Module Matched Filter Module Synchronization Module Preamble Design
Kray Althaus	Software Engineer	<ul style="list-style-type: none"> • Design and implement the CPU based decoder module • Design and implement the CPU based slicer and down-sampling module • Review code additions to the main repository • Order and organize all physical equipment necessary for the project through the Senior Design Purchasing department • Design physical system with CPU trade studies 	Decoder Module Slicer Module Down-sampler Module
Josue Ortiz	Software Engineer	<ul style="list-style-type: none"> • Construct the project plan • Design preliminary matched filter • Document project status and source code organization 	Project Plan Final Acceptance Report
Sebastian Thiem	Software Engineer	<ul style="list-style-type: none"> • Optimize most CPU based modules to CUDA GPU kernels • Design the integration of the benchmarking and demodulation subsystems • Design and implement effective throughput benchmark tests and profiling 	Decoding CUDA kernel Slicer CUDA kernel Matched Filter kernel Downsampler kernel
Kevin Siruno	Team Lead	<ul style="list-style-type: none"> • Document all source code, project plans, schedule delays, and meeting minutes • Organize in-person meetings • Hold all software engineers accountable • Assign peer reviews for the source code 	PDR, CDR, VSR, and FAR Decks Final Acceptance Report