Florida State University Libraries

Electronic Theses, Treatises and Dissertations

The Graduate School

2012

Real-Time Hardware Design for Improving Laser Detection and Ranging Accuracy

Jarrod Brown

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ENGINEERING

REAL-TIME HARDWARE DESIGN

FOR IMPROVING LASER DETECTION AND RANGING ACCURACY

By

JARROD BROWN

A Thesis submitted to the
Department of Electrical & Computer Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Summer Semester, 2012

Jarrold Brown defended this thesis on June 25, 2012.

The members of the supervisory committee were:

Linda DeBrunner
Professor Directing Thesis

Clay Hughes
Committee Member

Geoffrey Brooks
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the thesis has been approved in accordance with university requirements.

In loving memory of J.W. and Lucille Hawkins.

And to my teacher, mentor, and friend, Tommy Weaver.

ACKNOWLEDGEMENTS

First and foremost I thank God for giving me the ability and opportunity to perform this study. I would like to express my appreciation to my advisory committee. To my committee chair, Dr. Linda DeBrunner, and committee member, Dr. Clay Hughes, thank you both for your interest in my work, guidance, feedback, ideas, and direction. And thank you to my committee member and advisor Dr. Geoffrey Brooks for recruiting me and advising me through the graduate program. Special thanks to Steven Silvester for conducting the majority of my experiments and for your contribution of expert advice.

I must also thank my family and friends for their encouragement, interest, prayer, and confidence. I have had your support all of my life, and I am truly thankful. The most special thank you goes to my beautiful wife, Kelly. Thank you for your continuous love, support, prayer, and patience throughout this experience. I could not have done it without you.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	ix
ABSTRACT.....	xi
1 INTRODUCTION	1
1.1 Problem	2
1.2 Premise.....	3
1.3 Approach.....	3
1.4 Scope.....	4
1.5 Summary	4
2 BACKGROUND	6
2.1 Laser Detection and Ranging Applications	6
2.2 Laser Detection	7
2.2.1 Leading-Edge Detector	10
2.2.2 Peak Detector	11
2.2.3 Matched Filter Detector	12
2.3 Range Calculation	13
2.4 Digital Pulse Processing Techniques	13
2.4.1 Interpolation	14
2.4.2 Matched Filter	15
2.5 Real-Time Calculations	17
2.6 Previous Work	18
3 IMPROVING LADAR PULSE PROCESSING TECHNIQUES	22
3.1 Real-Time Matched Filter.....	22
3.1.1 Matched Filter Reference Waveform Models.....	22
3.1.2 Matched Filter Implementation.....	24
3.1.3 Hardware Design	26
3.2 Real-Time Matched Filtering with Interpolation	27
3.2.1 Matched Filter with Interpolation Implementation	28
3.2.2 Hardware Design	30
3.2.3 Summary	31
4 METHODOLOGY	33
4.1 Range Accuracy Experiment Design	33
4.1.1 Matched Filter Reference Waveform Test.....	35
4.1.2 Interpolation Method and Factor Test.....	37
4.1.3 Sampling Rate Test	37
4.2 Implementation Experiment Design	37

4.2.1	FPGA Implementation Test	38
4.2.2	Time Latency and Range Accuracy Test	38
4.3	Real-Time LADAR System Experiment Design	39
5	RESULTS	44
5.1	Range Accuracy Experiment Results	44
5.1.1	Matched Filter Reference Waveform Test Results	44
5.1.2	Interpolation Method and Factor Test Results	46
5.1.3	Sampling Rate Test Results	48
5.2	Implementation Experiment Results	49
5.2.1	FPGA Implementation Test Results	50
5.2.2	Time Latency and Range Accuracy Test Results	50
5.3	Real-Time LADAR System Experiment Results	52
5.3.1	500 ns Range Accuracy Test Results	52
5.3.2	1000 ns Range Accuracy Test Results	55
5.3.3	Summary	55
6	CONCLUSION	58
6.1	Conclusions from Results	58
6.2	Recommendations for Future Research	59
APPENDIX A		61
APPENDIX B		67
APPENDIX C		83
APPENDIX D		85
REFERENCES		157
BIOGRAPHICAL SKETCH		162

LIST OF TABLES

Table 2.1: Comparison of performance showing time to process a 512^2 pixel image using 6 different architectures, assuming a 50 MHz clock [41]	18
Table 3.1: Typical number of samples required (N) to build a reference waveform of 6 ns, 10 ns, and 16 ns pulses with interpolation where L is the interpolation factor	24
Table 3.2: Comparison of interpolation methods used with matched filtering showing the number of operations required to implement where L is the interpolation factor and N is the number of samples of the high resolution reference waveform.....	30
Table 4.1: Example pulse processing algorithm outputs with reference range of 38.0000 m.	35
Table 4.2: LADAR system parameters and settings for real-time experiment.....	40
Table 4.3: Example test parameters and results of one laser transmission	41
Table 4.4: Reference waveforms models used in LADAR experiment.....	42
Table 4.5: Test results printout of laser scan using 1 GHz sampling rate, interpolation factor of 32, 6 ns Gaussian reference waveform with stan. dev. of $2e^{-9}$	43
Table 5.1: Matched filter reference waveform test results showing comparison of edge detector, peak detector, and matched filter range estimation techniques with multiple reference waveforms	45
Table 5.2: Interpolation method test results showing comparison of edge detector, peak detector, matched filter, and matched filter with interpolation range estimation techniques with multiple interpolation methods	47
Table 5.3: Sampling rate test results showing comparison of edge detector, peak detector, and matched filter range estimation techniques with multiple starting sampling rates	49
Table 5.4: Compilation results of matched filter VHDL implementation showing logic used to build design and percentage of Stratix IV EP4SGX 230 device logic used	50
Table 5.5: Simulation of VHDL implementation showing clock cycle latency and time-of-flight (TOF) accuracy with 500 MHz sampling rate compared to MATLAB results	51
Table 5.6: Simulation of VHDL implementation showing clock cycle latency and time-of-flight (TOF) accuracy with 1 GHz sampling rate compared to MATLAB results.....	51
Table 5.7: LADAR system 500 ns test results comparing peak detector, sinc interpolation peak detector, and matched filter with interpolation factor 32 range estimation techniques with reference range of 74.8001 m	53

Table 5.8: LADAR system 1000 ns test results comparing peak detector, sinc interpolation peak detector, and matched filter with interpolation factor 32 range estimation techniques with reference range of 149.8370 m	56
--	----

LIST OF FIGURES

Figure 1.1: Air Force Research Lab LADAR technology tests showing high-resolution three-dimensional imagery [16]	1
Figure 1.2: LADAR scan data of a HMMWV at 60 m interpreted as (a) an intensity image and (b) a range image [17]	2
Figure 2.1: Conceptual diagram of LADAR system operation [4]	8
Figure 2.2: Flash single-pulse FPA and MEMS single-point scanning LADAR systems transmission operation [33]	9
Figure 2.3: Flash single-pulse FPA and MEMS single-point scanning LADAR systems receiving operation [33]	9
Figure 2.4: Example of LADAR laser intensity pulses plotted as a function of time	10
Figure 2.5: Example of leading-edge detector showing actual laser pulse, actual peak location, and captured pulse location	11
Figure 2.6: Example of peak detector showing actual laser pulse, actual peak location, and captured pulse location	12
Figure 2.7: Interpolation Methods on LADAR pulse with interpolation factor of 4.	15
Figure 2.8: Example of cross-correlation showing input data, reference waveform, and output .	16
Figure 3.1: Comparison of reference waveform models	23
Figure 3.2: Block diagram of an 8-tap digital FIR filter implementing matched filtering	26
Figure 3.3: Reduced block diagram of an 8-tap FIR filter implementing matched filtering	27
Figure 3.4: Combination of interpolation and matched filtering using parallel 16-tap FIR filters	28
Figure 3.5: Matched filter with zero padding interpolation example with $L=4$ showing 4 output data points (y) as a function of the input data (x) and the reference waveform (r)	29
Figure 3.6: Example of matched filter with zero padding interpolation implementation with interpolation factor 4 where each horizontal box represents a 4-tap FIR filter with coefficients r_n	29
Figure 3.7: FPGA hardware realization showing the amount of MAC operations required	31
Figure 4.1: Example LADAR data sequence showing start and return pulses sampled at 1 GHz	34
Figure 4.2: Reference waveforms showing effects of scalar multiple	36
Figure 4.3: Component layout of real-time experiment	40
Figure 5.1: Standard deviation of amplitude factors 0.1 to 10	45

Figure 5.2: Standard deviation of interpolation factors 1 through 50.....	47
Figure 5.3: Standard deviation of interpolation factors 1 through 200.....	48
Figure 5.4: Percent of range accuracy improvement of standard deviation and root mean square error at each sampling rate.....	53
Figure 5.5: LADAR system 500 ns test results comparing standard deviation.....	54
Figure 5.6: LADAR system 500 ns test results comparing root mean square error	54
Figure 5.7: LADAR system 1000 ns test results comparing standard deviation.....	56
Figure 5.8: LADAR system 1000 ns test results comparing root mean square error	57

ABSTRACT

Digital signal processing (DSP) algorithms for estimating target range and backscatter intensity from sampled laser detection and ranging (LADAR) systems are limited by the sampling rate of data collected and computation time requirements. An interpolating matched filter DSP algorithm is presented to improve range accuracy while maintaining a relatively low sampling rate. The algorithm interpolates sampled data and applies a matched filter with a high resolution reference waveform to recover super-sample positions of the transmitted and backscattered pulses. A custom computer architecture utilizing parallel processing is designed and synthesized on a field programmable gate array (FPGA) to optimize the DSP algorithm to operate in real-time. Research and simulation results comparing the effectiveness of different sampling rates, reference waveform models, and interpolation factors used to determine target range from LADAR data are presented. The FPGA hardware design was realized and tested with a LADAR system. A matched filter with zero padding interpolation design using a Gaussian shape reference waveform and an interpolation factor of 32 showed an 87% improvement in range accuracy over the peak detector design currently used in real-time LADAR systems.

CHAPTER ONE

INTRODUCTION

Laser detection and ranging (LADAR) systems are used to collect information on a scene such as object detection, distance, position, and backscattering characteristics. Distance, or range, to an object is found by measuring the time-of-flight for a laser pulse to travel through air at the speed of light, reflect off the object, and return to an optical receiver at the transmission source. Measuring accurate time-of-flight of a signal traveling at the speed of light is challenging, and exact measurements cannot be consistently obtained [1][2][3]. Section 2.2 describes several methods commonly used on LADAR systems to estimate range. Some methods process data as it is collected, also known as real-time [4][5]. Other methods capture data then implement post-capture digital signal processing (DSP) algorithms to estimate range [1][6][7].

In addition to range estimation, some LADAR systems also have the ability to determine backscattering characteristics of a target by capturing the intensity of the reflected laser pulse [4][8]. Military applications are interested in obtaining intensity and corresponding range values for target recognition and navigation of munitions [9][10][11]. Linear mode direct detection LADAR systems scan a field-of-view, saving the azimuth, elevation, range, and intensity value of each scan [8][12]. The data can be used to create a 3-D point cloud of the scanned area [5][13][14], as seen in Figure 1.1. Figure 1.2 shows collected LADAR data interpreted as an intensity image and as a range image. Image processing algorithms can be applied to the LADAR images to recognize, acquire, and track target location for weapon guidance [9][15].

Figure 1.1: Air Force Research Lab LADAR technology tests showing high-resolution three-dimensional imagery [16]

Figure 1.2: LADAR scan data of a HMMWV at 60 m interpreted as (a) an intensity image and (b) a range image [17]

Munitions with target-seeking navigation systems require on-board real-time LADAR systems since post-processing techniques cannot be performed due to mission timelines [10][18]. Therefore, both accurate distance measurements and intensity values must be found in real-time. The Air Force Research Lab (AFRL) is currently developing a digital LADAR system utilizing an analog-to-digital converter (ADC) to capture the waveform and to provide range and intensity values in real-time [19]. This digital system is capable of accurately recording backscatter intensity at the expense of range accuracy.

1.1 Problem

A major problem of existing range estimation methods used on real-time LADAR systems is inadequate range accuracy [2][6]. Accurate range calculation is essential for collecting truthful scene information used for terrain-mapping and weapon navigation [14][18]. Range accuracy of existing digital LADAR systems using ADCs is limited by the sampling rate of the digitizer [4]. For example, a LADAR system using an ADC with a 1 GHz sampling rate captures a new sample every 1 ns; therefore the system has time-of-flight accuracy of 1 ns and range accuracy of approximately 15 cm. To increase range accuracy, either faster ADCs or computation time-intensive DSP algorithms must be used [1][6][7]. Existing post-processing algorithms used to approximate very precise range measurements from sampled LADAR data, as described in Section 2.6, require several thousand mathematical operations to be completed for each range estimate. Due to the computation time required to complete the DSP algorithms on

standard computer architecture, the design cannot be implemented in real-time [20][21][22]. However, military applications require both accurate real-time range measurement and backscatter intensity readings.

1.2 Premise

This research proposes a hardware design solution to improve range accuracy of a LADAR system while maintaining real-time availability. The proposed algorithm combines existing pulse processing techniques used in detection and ranging systems to find a more accurate solution of time-of-flight. The algorithm interpolates data captured by a relatively low sampling rate to a higher sampling rate and applies a matched filter using high resolution reference waveform models to recover super-sample positions of the start and return pulses; thereby eliminating the need for a faster ADC. The proposed pulse detection algorithm is designed for implementation on specialized computer architecture to maximize performance. A hardware design utilizing parallel processing and pipelining optimizations is presented to execute the DSP algorithm in real-time. Implementing the algorithm in very high speed integrated chip (VHSIC) hardware descriptive language (VHDL) and instantiating it on a field programmable gate array (FPGA) allows accurate range calculation with short processing time.

1.3 Approach

The proposed solution to improve laser ranging accuracy while maintaining real-time capability is verified by a series of experiments. The first experiment uses simulation to show that the proposed DSP algorithm implementing a combination of pulse processing techniques enhances range accuracy of a LADAR system. This simulation is performed by selecting different combinations of matched filter reference waveform models and interpolation methods, implementing the algorithms in software using MATLAB [23] scripts, and simulating each method in MATLAB using actual LADAR data collected by a system at different sampling rates. The range calculated using the DSP algorithms is compared to the range calculated using methods already implemented with LADAR systems in real-time [4] and a reference range. For the MATLAB simulation results, the relationship between sampling rate and range accuracy is shown for each method.

The second experiment uses simulation tools to demonstrate that the DSP algorithm can be implemented on an FPGA and calculations can be completed in real-time. This is done by

converting the algorithm to VHDL, instantiating on an FPGA, and performing timing analysis using Quartus II [24]. Then timing simulations are completed using ModelSim [25] to plot the inputs and outputs of the system against time. Time required to complete range calculation and system throughput are provided from this experiment.

The final experiment demonstrates that the hardware design can be used as a real-time solution to improve range accuracy on an actual LADAR system. The VHDL design is instantiated as a component on an Altera Stratix IV FPGA based LADAR system. The experiment includes testing range accuracy of the design at various distances and comparing the results to other range estimation methods. By performing several hundred scans sequentially, real-time implementation is also verified on the actual laser ranging system.

1.4 Scope

The scope of this research is improving time-of-flight measurements of each laser transmission using DSP algorithms and the hardware design required to implement the algorithms. Alternative approaches for improving range estimation accuracy such as laser transmission patterns, optical receiver designs, and range prediction algorithms are outside the scope of this research. The purpose of this study is to improve range estimation of existing laser ranging systems without designing completely new systems. This research is partially in support of missile guidance technology research and evaluation. LADAR data used in simulations was collected and provided by the Air Force Research Lab; consequently capturing LADAR data is outside the scope of this research. The real-time interpolating matched filter design will assume the sampled LADAR data is available on the input port at a known sampling rate.

Research and testing is performed using 1.064 and 1.56 micrometer lasers. Since LADAR return pulse waveforms vary from system to system depending on the wavelength of the laser used [26], the experiment results presented are dependent on the particular LADAR system components. However, the hardware design proposed includes the capability to easily change the reference waveform to match the expected return pulse shape of another LADAR system.

1.5 Summary

This chapter explains the range accuracy problem of existing real-time LADAR systems and proposes a solution to increase range accuracy while maintaining real-time operation. The proposed solution includes designing a range estimation algorithm intended for parallel

processing optimizations and a hardware design to implement the algorithm in real-time. The rest of the thesis is organized by chapter, beginning with Chapter 2 which provides appropriate background information relating to LADAR system operation including laser ranging applications, laser pulse detection methods, and past work pertaining to the improvement of range accuracy. Chapter 2 also discusses common DSP techniques and the time required to utilize those techniques. The proposed DSP range estimation algorithm and hardware design to improve range accuracy in real-time is described in Chapter 3. Chapter 4 includes the methodology for executing experiments to verify the effectiveness and performance of the proposed design, and Chapter 5 presents results from the experiments and comparisons to other range estimation methods. Final conclusions and suggestions for future research are presented in Chapter 6.

CHAPTER TWO

BACKGROUND

In this chapter, Section 2.1 first explains capabilities and modern applications of laser detection and ranging systems. Section 2.2 then provides a description of the operation of laser detection, typical system components, and pulse detection methods used in LADAR systems. Next Section 2.3 presents the general range equation used to calculate range from recovered laser pulse location. Section 2.4 discusses digital pulse processing techniques which have been used to approximate a more accurate range calculation. The importance of real-time calculation and the computer architecture available to implement real-time processing is then explained in Section 2.5. Finally, Section 2.6 describes previous work relating to improving laser detection and ranging.

2.1 Laser Detection and Ranging Applications

Active detection and ranging systems are capable of approximating distance, position, velocity, and backscatter intensity by transmitting an electromagnetic signal and receiving signals with similar characteristics [26]. Light Detection and Ranging (LIDAR) systems transmit electromagnetic radiation with short wavelengths in the optical radiation or light portion of the electromagnetic spectrum [8][27]. LADAR is a subcategory of LIDAR where lasers are used as the transmission source for short wavelength signals [10][28]. Many applications of LADAR are currently used in a variety of fields.

One of the most popular applications of laser ranging is terrain mapping. Airborne LADAR systems can scan and collect three-dimensional images to generate high-resolution maps of wide-area terrain [8][14]. In agriculture, LADAR systems mounted to planes are used to create accurate topography maps of corn and cotton fields [29]. By reflecting laser pulses off the landscape, characteristics such as elevation, slope, and sun exposure are mapped. Precision farmers use these maps to direct valuable resources to specific areas in order to increase yield rates. Similar applications are used in forestry to determine tree heights and forest density [8][30]. By capturing multiple backscattered pulses from one transmitted pulse additional range data such as crown height can be extracted. LADAR terrain mapping is also used in space exploration applications such as the Lunar Orbiter Laser Altimeter (LOLA) instrument [31] used

on NASA's Lunar Reconnaissance Orbiter mission. LOLA measures the time-of-flight (range), pulse spreading (surface roughness), and transmit/return energy (surface reflectance). Data returned from LOLA has been used to produce a high-resolution global topographic model of the moon that enables precise targeting, safe landing, and surface mobility to perform future exploratory missions.

Military applications use terrain mapping for reconnaissance and surveillance [10][14]. However, the primary research of military LADAR application is focused on target recognition, identification, and tracking [3][10][11][13][15][17][18]. LADAR offers similar benefits as RADAR military applications but are capable of higher precision in distance measurement and much higher angular resolution [27]. One aspect of LADAR that cannot be accomplished using RADAR is the ability to detect and distinguish multiple return pulses from one start pulse [13][28]. By repeatedly illuminating an area covered by foliage or netting and collecting the return data, an image of what is hidden beneath the obstruction becomes visible [13][17]. However, many of the LADAR target acquisition applications are restricted by accuracy and processing speeds [1] [3][6][18].

Another common application of laser detection and ranging is the LADAR speed gun used by law enforcement. Unlike RADAR speed guns which rely on the Doppler Effect, LADAR speed guns consecutively measure distance to a vehicle over a span of time, and then calculate velocity from the change in distance over time [32]. Laser ranging for speed enforcement offers the advantages of a very narrow beam spread and less scattering than RADAR, making speed detection more accurate and reduces early warning.

2.2 Laser Detection

A LADAR system consists of a laser transmitter, an optical receiver, a pulse detector, and a data processing unit [4]. Figure 2.1 illustrates the general operation of a typical LADAR system. The system operates by transmitting a short laser pulse in the optical electromagnetic spectrum range towards a target and measuring backscattered light in the same spectrum range with an optical lens [10][28].

LADAR systems can be categorized into two types of operation, single-point scanning systems and single-pulse flash systems [12][33]. The transmission operation of both flash and scanning systems is shown in Figure 2.2, and the receiving operation is shown in Figure 2.3.

Scanning systems use a micro-electro-mechanical system (MEMS) combined with a mirror to reflect a narrow beam laser pulse to a single point in the field-of-view and measure backscattered light with a photodiode [4][34]. The MEMS then slightly tilts the mirror to image the next point, and the process is repeated to scan the field-of-view. Flash LADAR systems transmit a wider spread laser beam and use a two dimensional array of photodiodes called a focal-plane array (FPA) to measure backscattered light at several points simultaneously [13][27]. Due to the wider beam spread, Flash systems can scan a field-of-view much faster than single-point systems, but require a considerably more powerful laser to illuminate the larger area [5][33].

Figure 2.1: Conceptual diagram of LADAR system operation [4]

The photodiode receiver measuring reflected light continuously converts all light into an electronic waveform [6][9][17]. Figure 2.4 shows an example of a LADAR waveform. By partially reflecting the transmitted pulse near the transmission source, the backscattered pulse captured immediately after transmission can be used as a representative start pulse. This technique is used to capture the transmitted pulse on the same channel as the backscattered pulse [4]. The timeframe in which the transmitted pulse is expected to arrive is referred to as the transmission window. Likewise, the timeframe that light reflected off a target is expected to arrive is referred to as the backscatter window.

Figure 2.2: Flash single-pulse FPA and MEMS single-point scanning LADAR systems transmission operation [33]

Figure 2.3: Flash single-pulse FPA and MEMS single-point scanning LADAR systems receiving operation [33]

Figure 2.4: Example of LADAR laser intensity pulses plotted as a function of time

Time elapsed for the pulse to travel round trip is used to determine range to the object and the peak pulse amplitude is used to determine backscatter intensity [13][28][35]. To measure time-of-flight, approximate position of start and return pulses must be identified within the LADAR waveform. To measure backscatter intensity, the maximum amplitude of both pulses must be identified. There are three types of detectors used in LADAR systems to recover start or return laser pulses from waveform data: leading-edge detector, peak detector, and matched filter detector [4]. All three detector types described in the next sections can be implemented in analog using electronic circuits or digitally by first sampling the LADAR waveform with an ADC.

2.2.1 Leading-Edge Detector

A simple leading-edge detector recovers a LADAR pulse by finding the time at which the LADAR waveform crosses a threshold value [4]. Edge detection pulse recovery using a threshold value is shown in Figure 2.5. When the threshold is exceeded, an enable signal is sent to a time-to-digital converter to capture the time of the leading-edge. Alternatively, some systems use a leading-edge detector to enable a system clock to count the clock cycles between the start and return pulses [32][34]. If the waveform is sampled to give discrete data, the number of samples between the pulses determines time-of-flight. Leading-edge detectors are capable of accurately measuring time-of-flight in real-time and the logic required to implement the detector is simple [36]; however, even advanced variations of this detection method do not have the capability to accurately capture pulse intensity. Therefore the leading-edge detector is not the

preferred pulse detection method for military applications since the backscattered laser pulse intensity is not captured [17].

Figure 2.5: Example of leading-edge detector showing actual laser pulse, actual peak location, and captured pulse location

2.2.2 Peak Detector

A peak detector recovers a LADAR pulse by finding the maximum amplitude in the LADAR waveform [4][34]. The transmitted and backscattered pulses are separately recovered by finding the maximum amplitudes within the respective windows. Time-of-flight is then measured in real-time from transmitted peak to backscattered peak. Similar to the leading-edge detector, time-of-flight can be measured using a time-to-digital converter, the counter on a system clock, or by the number of samples between pulses. The amplitude value of the backscattered pulse peak is captured as the backscatter intensity value. For digital peak detectors, an ADC may not accurately capture the actual pulse peak, as shown in Figure 2.6. Therefore the exact position and the correct backscatter intensity value may exist between two discrete samples, resulting in inaccurate time-of-flight and intensity measurements. Advanced peak detector systems utilize interpolation methods, as discussed in Section 2.4.1, to recover super-sample values from discrete data [1]. However, the interpolation algorithms are not optimized for real-time implementation. In order to achieve accurate range calculation with the peak estimator, extra time must be taken to complete post-processing scripts.

Figure 2.6: Example of peak detector showing actual laser pulse, actual peak location, and captured pulse location

2.2.3 Matched Filter Detector

A matched filter detector records the LADAR waveform and performs cross-correlation with a reference waveform model to recover a LADAR pulse; the maximum correlation value represents the peak of the recovered pulse. In order to successfully perform cross-correlation, the entire pulse shape must first be captured. If a portion of the pulse is not captured, inaccurate range calculations will occur [1]. Typically an ADC is used to sample and record the LADAR waveform [4]. Because of the fast sampling rates and long scan times normally used on LADAR systems, large amounts of memory are required to record all the samples of one LADAR waveform scan. Currently, in order to reduce the amount of samples stored in memory, matched filter detectors require additional circuitry to capture only the portion of the LADAR waveform containing the return pulse [4]. The samples from the ADC enter a shift register long enough to hold the entire length of the sampled return pulse. A threshold circuit detects the arrival of the return pulse and transmits an enable signal. Then the samples in the shift register are stored for post-capture matched filter pulse processing.

The matched filter digital pulse processing technique, discussed further in Section 2.4.2, allows for more accurate and consistent time-of-flight measurements than the peak detector. But, similar to the peak detector, data occurring between discrete samples is not collected. Therefore accuracy of matched filtering is also limited by the sampling rate of the ADC. Another drawback of the matched filter detector is the time latency required to capture the LADAR waveform and

perform post-capture cross-correlation to detect a pulse. Currently, no LADAR system matched filter detector is capable of producing results in real-time [4]. Section 3.2.1 proposes a real-time implementation of the matched filter detector utilizing interpolation to increase pulse position accuracy while using less circuitry. More accurate time-of-flight measurements between the transmitted pulse and the backscattered pulse enable an accurate calculation of range.

2.3 Range Calculation

Using a pulse detection technique, as described in Section 2.2, the time elapsed between the start pulse and the return pulse can be measured. Since the lasers used in LADAR systems emit electromagnetic radiation that propagates at the speed of light, the approximate travel speed of the transmitted laser pulse is known. With time-of-flight and speed of light known, one-way range to a target is calculated using

$$r = \frac{t_{diff} \times c_0}{2} \quad [2.1]$$

where r is range, t_{diff} is time elapsed between pulses, and c_0 is the approximated speed of light through air [4]. Equation 2.1 is a straightforward calculation of range. More advanced range equations are formed by including the velocity of observer, the shape and width of the laser pulse, and characteristics of laser propagation, reflection, and reception [4][28]. However, this research focuses on determining accurate and consistent time-of-flight measurements. Therefore Equation 2.1 is used in simulation results to compare range as a direct function of time-of-flight.

The only contribution of error in Equation 2.1 comes from inaccurately detecting the precise position of a pulse. For an accurate measurement of time-of-flight, exact position of the pulses must be recovered from the data collected. Digital processing can be applied to discrete LADAR data to improve the accuracy of pulse recovery.

2.4 Digital Pulse Processing Techniques

DSP can be used to recover laser pulses within a sequence of discrete time samples. Digital signal pulse processing techniques are capable of approximating a pulse location more accurately than processing raw data [1]. Two DSP techniques that are widely accepted for post-processing digital LADAR data are interpolation and matched filtering [1][4][9][35].

2.4.1 Interpolation

Digital LADAR systems may determine inaccurate time-of-flight and intensity readings if the actual pulse shape or peak is not captured by the ADC. For example, a digital peak detector recovers the return pulse from a LADAR waveform by uniformly sampling to produce a discrete sample set and searching for the maximum sample in the return window [4]. The actual peak of the return pulse could lie somewhere between two samples. Therefore, the sampling rate limits the timing resolution and backscatter intensity. Interpolation is used to up-sample data by estimating points between samples.

Interpolation is a technique of adding new data points within a discrete set of known data points. When used in digital processing of a sampled waveform, interpolation creates new data points between collected samples, referred to as super-sample values [36]. Since ADC collects data at equally spaced intervals, interpolation effectively up-samples the signal to a higher sampling rate. The sampling rate is multiplied by an integer referred to as the interpolation factor, typically denoted by L . A data vector interpolated by a factor of L is expanded, creating $L-1$ empty elements between each original sample. Then these elements are filled according to the type of interpolation method used.

LADAR pulse processing places two constraints on which method to use: accuracy and computation time [5]. Some methods require little or no computation time such as nearest neighbor interpolation where the new data points inserted are assigned the same value as the nearest original samples [37]. Another interpolation method used to up-sample data very quickly is to simply add zeros between data samples. Linear interpolation uses a linear function of two known data points to fill equally spaced empty elements [37][38]. However, when used with the peak estimator these low computation interpolation methods do not increase range or intensity accuracy, because the newly added data points are always less than or equal to existing samples and the maximum samples remain the same.

Computationally intense interpolation methods use higher order polynomials and more surrounding points to estimate more accurate data points between samples [38]. For example, the Cubic Spline interpolation method uses a total of four samples to produce smoother, continuous, and more accurate results [39]. Sinc interpolation convolves a shifting sinc function with a sampling function in order to restore the original continuous function. Figure 2.7 compares zero padding, nearest neighbor, linear, and Cubic Spline interpolation methods used on LADAR data.

To reduce the computation time required to evaluate linear or polynomial functions, look-up tables (LUTs) can be used to approximate super-sample values.

Figure 2.7: Interpolation Methods on LADAR pulse with interpolation factor of 4.

2.4.2 Matched Filter

Matched filtering is an effective pulse processing technique used to reduce noise and recover a specific shape from a data sequence [36]. The filter compares an input data signal with a reference signal to determine the similarity between the pair. The LADAR system matched filter detector described in Section 2.2.3 utilizes matched filtering to recover a pulse from captured LADAR data. Matched filtering is implemented by calculating the cross-correlation of an input data waveform with a shifted reference model waveform to produce an output sequence containing a correlation value corresponding to each reference waveform shift [36]. Figure 2.8

shows the smooth output produced by cross-correlating the noisy input data with the reference waveform.

Figure 2.8: Example of cross-correlation showing input data, reference waveform, and output

The cross-correlation sequence is defined by

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]r[k-n], \quad n = 0, \pm 1, \pm 2, \dots \quad [2.2]$$

where $x[k]$ is the input sequence, $r[k]$ is the reference waveform, and $y[n]$ is the cross-correlation output sequence [40]. The parameter n indicates the time-shift between the input sequence and the reference waveform. For pulse processing LADAR data, the reference waveform is used to describe the expected return pulse.

Matched filtering recovers the location in the LADAR data that best matches the reference waveform by finding the index of highest correlation. The ability of a matched filter to accurately determine the location of the actual return pulse is limited by the detail and shape of the reference waveform [7]. Reference waveform models have been developed to describe the return pulse of a LADAR system as inverse Parabola, Gaussian, and Poisson functions [1]. However, LADAR system parameters such as the laser transmitter, object reflectivity, range, and optical receiver affect the shape of the return pulse [4][28]. As a result, selecting an appropriate

reference waveform model that is typical for all systems is very difficult. Section 3.1.1 discusses and compares the effectiveness of different waveform models.

Other limitations of matched filtering include sample resolution and intensive computation requirements. Since filtering is only as accurate as the nearest index [1][40], a matched filter is limited by the sampling rate of the LADAR waveform sequence. Section 3.2 introduces interpolating the sampled LADAR data to a faster sampling rate before matched filtering to improve sample resolution. The summation in Equation 2.2 must implement numerous multiply-accumulates (MAC) for every input sample shifted into the filter. For matched filtering applications utilizing fast sampling rates, extended calculation is required and should be compensated with adequate processing time [7][40].

2.5 Real-Time Calculations

Real-time processing systems are hardware and/or software systems subject to operation time constraints from event to system response [5][41]. Military LADAR applications such as target identification and tracking require real-time processing for instantaneous decision making [13][18]. To avoid delayed results, data processing of each LADAR scan must be completed within one pulse repetition period. The typical repetition rate of a LADAR system is 10 KHz. With a 1 GHz sampling rate; capturing an entire LADAR sequence requires 100,000 samples. Any DSP applied to the 100,000 samples to improve range accuracy must be completed within one repetition, or 100 μ s. Since the LADAR system is continuously scanning, range throughput of the system must be unity. Consequently, the system must process approximately one billion samples per second. Furthermore, post-capture processing systems using DSP to recover a more accurate pulse location, such as the matched filter detector explained in Section 2.2.3, require several hundred MAC operations per sample.

Section 3.1.2 introduces parallel processing and pipelining techniques to implement matched filtering and interpolation algorithms on LADAR data in real-time. Standard computer architectures are not optimized to support the large number of mathematical operations present in DSP and maintain real-time output availability [42][43]. Digital signal processor chips capable of implementing large calculations very quickly and repeatedly are available. The architecture is optimized for DSP by creating multiple data paths and pipelining each data path to enhance program flow [42]. The fastest DSP core found during this research was the 100 GMAC/s [44],

manufactured by Tensilica. However, the fastest digital signal processor chip found utilizes 8 cores at 40 GMAC/s totaling in 320 GMAC/s [45], manufactured by Texas Instruments.

Custom computer architecture can also be designed to utilize parallel processing. A common device used for developing custom computer architecture is an FPGA [46]. An FPGA is an integrated circuit produced by reprogrammable hardware capable of connecting logic gates together to form multipliers, registers, adders, and more. Unlike a digital signal processor that must use shared resources like memory buses or possibly the processor core [47] an FPGA dedicates logic for receiving the data. Therefore, high rates of I/O are maintained. Also, the parallelism of an FPGA is customized by the designer and limited only by the amount of resources, allowing for architecture that can be optimized for application. Conversely, digital signal processors are not customizable and the amount of parallel cores is set by the manufacturer. Comparisons between programmable DSP chips and FPGAs for real-time convolution are presented in Table 2.1 [41].

Table 2.1: Comparison of performance showing time to process a 512^2 pixel image using 6 different architectures, assuming a 50 MHz clock [41]

Hardware Type	Architecture	No. of Processors	Cycles per pixel	Time to Process (ms)
FPGA	Single	1	4	20.1
	Parallel	4	1	5.24
	Pipelined	4	1	5.24
	Parallel Pipelined	16	0.25	1.31
DSP	Single	1	9.5	200
	Parallel	4	38	50

2.6 Previous Work

To our knowledge, a real-time algorithm for improving LADAR range accuracy using a combination of interpolation and matched filtering techniques has never been developed prior to this research. However, much work has been done on similar topics. This section presents similar

ideas, theories, and simulations while outlining differences between research already explored and new research presented in this thesis.

The Air Force Institute of Technology (AFIT) houses an abundance of knowledge in LADAR technology. One master's thesis paper proposes an algorithm using a gated negative parabolic curve as a model to recover the position of the return pulse [7]. In this approach, a gated parabola is used to approximate the shape of the transmitted pulse of laser light. The algorithm then repetitively scales and slides the gated parabola until the sum-squared difference between the real data and the algorithm's proposed model is minimized. The algorithm iteratively repeats the sum-squared difference algorithm while continuously changing the reference waveform to enhance results. The algorithm developed was shown, through simulation, to be very effective in recovering the return pulse to the nearest sample. However, several thousand iterations are required to find the most accurate pulse location. Additionally, the sum-squared difference calculation requires more operations than matched filtering. Since each iterate requires over a million operations, the algorithm's intense calculations are not optimized for real-time systems. Also, interpolation methods were not discussed and the accuracy of the sum-squared difference algorithm is still limited by the sampling rate.

Another AFIT master's thesis research compares several range estimation algorithms, including a peak estimator and a matched filter [1]. Interpolation methods such as linear, Spline, and Pchip are explored with the peak estimator, but are never combined with the matched filter. The matched filter algorithm is applied in both time and frequency domains on a standard computer and Parabolic, Gaussian, and Hybrid Waveform models are considered. A series of pulse-width estimation techniques are also introduced to estimate the width of the pulse used with matched filtering, along with the normalized form of matched filtering. Data captured by a matched filter detector, as discussed in Section 2.2.3, was used in the thesis paper. The post-processing matched filter algorithm performed well when the detector captured the LADAR backscattered pulse waveform in the center of the range gate. However, the research deemed the matched filter as vulnerable to a bias error because the waveform capture frequently missed a portion of the pulse due to being left or right aligned in the range gate which results in inaccurate range calculations. Although interpolation methods are used with the peak estimator, the research does not discuss interpolating data and cross-correlating with a high resolution reference waveform to improve range accuracy. All simulations shown in the paper were completed using

MATLAB, and no real-time solution is presented. The research concludes that matched filtering increases range accuracy but suffers from increased processing time. Section 3.2 proposes a real-time matched filter design that continuously filters the input signal and no range gating is used. Therefore the entire LADAR waveform is included and matched filtering will not suffer from bias error.

Faisal, et al., proposes digitizing data collected by a nuclear particle detection system and using an FPGA to pre-process data in real time [48]. The operation of a digitizing nuclear particle detection system is similar to a digital LADAR system. It consists of photomultiplier tubes, digitizer unit, and storage and signal processing capabilities. When a particle strikes the detector, a voltage pulse is generated by the photomultiplier tube. The pre-processing technique proposed is normalized cross-correlation to recover pulse data for accurate time of arrival and peak amplitude estimations. Reference waveforms used in the cross-correlation algorithm are a square pulse and averaged return pulse. The algorithm is simulated in MATLAB, and the technique was designed for an FPGA. Therefore the design is very likely capable of real-time implementation, but no VHDL code or simulation results are presented. The work describes reference waveform matching ability of normalized cross-correlation; however the accuracy of the measurements are still limited by the sampling rate of the system since interpolation methods are not used.

Matched filter and interpolation algorithms have also been applied to ultrasound devices used to measure blood velocity [49]. A comparison of interpolation methods used to achieve super-sample positions of echo pulses for measuring time-delay in blood vessels have been presented. One method combines a zero-padding interpolation method with a matched filter. The matched filter provides super-sample linear filtering which maximizes the signal peak to output noise. The matched filter also functions as an anti-aliasing filter which eliminates the image spectra produced in zero padding. The Gaussian function is the only reference waveform model discussed. Simulation results provided show a high performance in blood velocity estimation using the matched filter interpolator. The results also show the matched filter requires far more computation operations than other less accurate methods presented [49]. The estimated number of operations is presented but a real-time implementation of the calculation is not addressed.

In summary, range estimation algorithms for LADAR data have been developed to increase range accuracy but the algorithms require extended processing time to implement after

data is collected [1][7]. Range gating is used to reduce the amount of data to process by capturing only a portion of the LADAR waveform; however, bias error is introduced if the entire pulse shape is not captured [1]. Accuracy of matched filtering is limited by the sampling rate of the data but interpolation methods can be combined with matched filtering to achieve super-sample accuracy as shown in the medical field [49]. Military applications require LADAR systems to achieve accurate range estimations while operating in real-time [13][18]. This research proposes a hardware design for matched filtering and interpolating LADAR data in real-time to increase range accuracy. The hardware design is massively parallel and implements the MAC operations required for matched filtering with short output latency. Since the throughput of the design is unity, LADAR data is continuously processed with no range gating required.

CHAPTER THREE

IMPROVING LADAR PULSE PROCESSING TECHNIQUES

This chapter describes proposed matched filtering techniques and hardware designs for estimating accurate range values of LADAR systems in real-time. Section 3.1 first describes the different reference waveform models used in matched filtering for LADAR data then describes a hardware design capable of implementing matched filtering in real-time. Section 3.2 improves the design of a real-time matched filter by adding interpolation to increase range accuracy and presents a hardware design capable of implementing matched filtering plus interpolation in real-time.

3.1 Real-Time Matched Filter

As previously discussed, range estimation is limited by the sampling rate if the digital LADAR system utilizes an ADC to capture intensity values. To obtain both consistent and accurate range measurements, DSP must be applied to the sampled data sequence. Applying matched filtering to LADAR data as described in Section 2.4.2 and shown in Section 5.1.1 can increase range accuracy by 33%. The increase in performance is achieved by accurately describing the backscattered pulse with a reference waveform model. This section presents a matched filter implementation on a specialized computer architecture which can be utilized in LADAR systems to meet time constraints. The appropriate selection of a reference waveform model to optimize range accuracy is also explained.

3.1.1 Matched Filter Reference Waveform Models

In a LADAR system, the backscattered pulse is a slightly altered version of the transmitted pulse, largely due to reflection and scattering properties of the target. Therefore the exact shape of each backscattered pulse being recovered is not always known. However, the width and shape of the reference waveform used in matched filtering can be modeled from the general shape of typical backscattered pulses. If the reference waveform does not exactly match the backscattered pulse but has a similar width and shape, the matched filter will only be inaccurate by a few samples.

The width and shape of both the transmitted pulse and backscattered pulse may also differ for each LADAR system and is dependent on the laser transmitter and optical receiver used. Pulse width estimation techniques [1] can be used to build typical reference waveform models for specific setups. The techniques work by capturing backscattered pulses of the current setup and repeatedly calculating mean square error with multiple waveforms while altering pulse width and shape parameters. Typical reference waveform model shapes include: Square, Gaussian, Inverse Parabola, Poisson, Weibull, and average return, as seen in Figure 3.1.

Figure 3.1: Comparison of reference waveform models

The number of samples required to build a reference waveform depends on the expected length of the backscattered pulse and the sampling interval of the LADAR system capturing the backscattered pulse. As the sampling rate of the ADC increases, the number of samples required to accurately describe the shape of the reference pulse increases. Therefore data points must be

inserted in the reference waveform to maintain an identical sampling interval as the ADC. The number of samples required to build a reference waveform can be calculated by

$$N = \left(\left\lfloor \frac{\text{Length of pulse}}{\text{Sampling interval}} \right\rfloor + 1 \right) \times L \quad [3.1]$$

where N is the number of samples in the reference waveform and L is the interpolation factor. Table 3.1 shows the number of samples required for typical reference pulse lengths and ADC sampling rates. Implementation of matched filtering requires multiplication and summation operations of every data sample. As discussed further in Section 3.1.3, higher sampling rates require more mathematical operations and thus more processing to complete matched filtering.

Table 3.1: Typical number of samples required (N) to build a reference waveform of 6 ns, 10 ns, and 16 ns pulses with interpolation where L is the interpolation factor

ADC sampling rate	Sampling Interval	6 ns pulse N	10 ns pulse N	16 ns pulse N
250 MHz	4 ns	$2 \times L$	$3 \times L$	$5 \times L$
500 MHz	2 ns	$4 \times L$	$6 \times L$	$9 \times L$
1 GHz	1 ns	$7 \times L$	$11 \times L$	$17 \times L$

3.1.2 Matched Filter Implementation

The matched filter algorithm can be implemented in real-time by utilizing a computer architecture capable of processing the numerous multiply and addition operations described in Equation 2.2 in parallel. By rewriting Equation 2.2 as

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]r[-(n-k)] = x[n] * r[-n] \quad [3.2]$$

it is shown that cross-correlation is given by the convolution sum of $x[n]$ with $r[-n]$, the time-reversed version of the reference waveform $r[n]$ [36]. Since the LADAR reference waveform is constant and has finite length, the summation in Equation 3.2 can be reduced. Substituting $k = n - i$ and letting the reference waveform be defined for $0 \leq i \leq N$ and zero otherwise, cross-correlation is defined by

$$y[n] = \sum_{i=-N}^0 x[n-i]r[-i] = x[n] * r[-n] \quad [3.3]$$

where the LADAR data waveform $x[n]$ is reversed and shifted by n samples. The reference waveform $r[n]$ is also reversed. Each sample of the reference waveform is multiplied by the corresponding sample of the LADAR data, and the products are accumulated.

Convolution can be implemented using a discrete-time finite impulse response (FIR) filter. The output of a FIR filter is defined by

$$y[n] = \sum_{i=0}^N h_i x[n-i] = x[n] * h[n] \quad [3.4]$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, h_i is the filter coefficient, and N is the filter order [36]. By observing the similarities between cross-correlation in Equation 3.3 and convolution in Equation 3.4, it was concluded that the cross-correlation required for matched filtering can be implemented using a FIR filter by reversing the sample order of the reference waveform. This can be accomplished by setting the FIR filter order to the length of the reference waveform and setting the filter coefficients to the desired reference waveform with samples in reverse order.

A FIR filter can be implemented in real-time with digital components by using D flip-flops to register previous input samples, applying parallel multipliers to each sample and the corresponding filter coefficient, and using an adder tree to sum the products. Figure 3.2 shows the block diagram of an 8-tap FIR filter. The eight delayed inputs, or taps, from the D flip-flops are routed to multiplication blocks. Eight predetermined coefficients, shown as h_n , are also routed as inputs to the multiplication blocks. Every clock cycle data flows to the next registered stage of the multiply-accumulate process. Output latency is determined by the number of clock cycles required to multiply one sample and proceed through the adder tree. However, throughput is unity because every clock cycle the next output sample is ready. Therefore the output latency of a FIR filter may be several clock cycles (determined by the number of coefficients), but the system can be designed so that throughput is unity and no bottlenecking occurs.

Matched filtering can be implemented using the same structure and components as the real-time FIR filter by letting the filter coefficients equal the sample-reversed reference waveform. For example, assume the reference waveform for a matched filter application is defined as eight samples denoted as r_n , where r_0 is the first sample of the reference waveform

and r_7 is the last. By letting the FIR filter coefficient h_0 equal r_7 and so on as shown in Figure 3.2, the matched filter is implemented using the FIR filter design. The block diagram of the matched filter can be reduced to a single block as seen in Figure 3.3, where the coefficients are listed inside the block.

3.1.3 Hardware Design

The real-time FIR filter can be realized in hardware using a series of D flip-flops, multipliers, and adders. A FIR filter is fundamentally a sequence of MAC operations resulting in one output sample. Many computer architectures such as digital signal processors or custom FPGA architecture as described in Section 4.2 have the ability to efficiently complete numerous MAC operations at fast processing speeds. The number of MAC operations depends on the filter order (the number of taps). If an 8-tap filter is to be realized, then 8 MAC operations are required. For matched filtering, the number of taps in the FIR filter is equal to the number of samples in the reference waveform. As explained in Section 3.1.1 and shown in Table 3.1, the size of the reference waveform is determined by the time duration of the transmitted pulse and the ADC time resolution.

Figure 3.2: Block diagram of an 8-tap digital FIR filter implementing matched filtering

Figure 3.3: Reduced block diagram of an 8-tap FIR filter implementing matched filtering

The hardware selected to implement matched filtering must have the capability to interface with a high rate ADC and support the numerous MAC operations at high speeds. As shown in Table 3.1, the longest reference waveform for a typical LADAR system with an ADC operating at 1 GHz sampling rate is 16 samples. Since the FIR filter used for matched filtering operates with no bottlenecking, the matched filter produces outputs at the same rate as the ADC. Therefore architecture capable of completing 16 MAC operations per sample at 1 GSPS, or 16 GMAC/s, is required for this design. However, the proposed design in Section 3.2.2 requires 512 GMAC/s.

In addition to completing matched filtering, the hardware design on the LADAR system requires additional logic to detect the peaks of the transmitted and backscattered laser pulses from the matched filter output and then calculate time-of-flight. Essentially, a peak detector is connected to the output of the matched filter. When the laser is triggered a counter begins counting clock cycles elapsed. Every clock cycle the output of the matched filter is compared to the current maximum intensity value. When a peak is detected, the current index and intensity are stored. The process is continued to find the peak intensity and corresponding index in the transmission window and the peak intensity and corresponding index in the backscatter window. The difference between the peaks is used to calculate range. Based on the numerous MAC/s and custom decision logic required to implement pulse detection, a custom architecture on an FPGA is the preferred choice to host this hardware design.

3.2 Real-Time Matched Filtering with Interpolation

Matched filtering LADAR data increases range accuracy significantly; however, the output resolution of the matched filter is limited by the sampling rate of the ADC. As presented in Section 5.1, when interpolation is added to the range estimation algorithm; the range accuracy of the matched filter LADAR system increases by 33%. By adding new interpolated data points between originally captured data samples, higher resolution input data and reference waveforms

can be used in cross-correlation. Therefore, the output of the matched filter contains a higher resolution sequence than the original captured LADAR data, resulting in super-sample range accuracy. As shown in Table 3.1, interpolation directly affects the length of the reference waveform by a factor of L . Consequently, interpolation also increases the number of MAC operations by a factor of L . Therefore the increase in performance from interpolation requires more data processing.

3.2.1 Matched Filter with Interpolation Implementation

Interpolation can be implemented with matched filtering in real-time by inserting samples in the captured input data. With an interpolation factor of L , there will be L output samples for every one input sample. To avoid any bottlenecking, L FIR filters can be implemented in parallel, where each filter is responsible for one output sample. Figure 3.4 shows an example of combining interpolation with matched filtering using a 16-tap filter. Interpolation increases the number of MAC operations required by a factor of L . Therefore a total of $16 \times L$ MAC operations are required. However, many computer architectures have the ability to realize the FIR filters in parallel with minimum impact on output delay. An example is presented in Section 3.2.2.

Figure 3.4: Combination of interpolation and matched filtering using parallel 16-tap FIR filters

As the interpolation factor increases, more LUT, multiply, and addition operations are required. The maximum interpolation factor is determined by the amount of resources available on the architecture. Zero-padding interpolation can achieve higher interpolation factors while using fewer resources. Figure 3.5 shows an example of equations for a 16-tap matched filter with

zero padding interpolation and interpolation factor L of 4. Since zeros are added as new data points, many of the multipliers required to implement matched filtering are eliminated. When the matched filter is realized, as shown in Figure 3.6, only $\frac{1}{4}$ of the multipliers are required for each FIR filter. Therefore the matched filter with zero padding interpolation can be realized using a total of only 16 MAC operations. The only growth in required resources comes from a larger reference waveform resulting from an increase in sample frequency due to interpolation.

$$\begin{aligned}
 y_0 &= x_0 r_{15} + \underline{0r_{14} + 0r_{13} + 0r_{12}} + x_1 r_{11} + \underline{0r_{10} + 0r_9 + 0r_8} + x_2 r_7 + \underline{0r_6 + 0r_5 + 0r_4} + x_3 r_3 + \underline{0r_2 + 0r_1 + 0r_0} \\
 y_1 &= \underline{0r_{15}} + x_0 r_{14} + \underline{0r_{13} + 0r_{12} + 0r_{11}} + x_1 r_{10} + \underline{0r_9 + 0r_8 + 0r_7} + x_2 r_6 + \underline{0r_5 + 0r_4 + 0r_3} + x_3 r_2 + \underline{0r_1 + 0r_0} \\
 y_2 &= \underline{0r_{15} + 0r_{14}} + x_0 r_{13} + \underline{0r_{12} + 0r_{11} + 0r_{10}} + x_1 r_9 + \underline{0r_8 + 0r_7 + 0r_6} + x_2 r_5 + \underline{0r_4 + 0r_3 + 0r_2} + x_3 r_1 + \underline{0r_0} \\
 y_3 &= \underline{0r_{15} + 0r_{14} + 0r_{13}} + x_0 r_{12} + \underline{0r_{11} + 0r_{10} + 0r_9} + x_1 r_8 + \underline{0r_7 + 0r_6 + 0r_5} + x_2 r_4 + \underline{0r_3 + 0r_2 + 0r_1} + x_3 r_0
 \end{aligned}$$

Figure 3.5: Matched filter with zero padding interpolation example with $L=4$ showing 4 output data points (y) as a function of the input data (x) and the reference waveform (r)

Figure 3.6: Example of matched filter with zero padding interpolation implementation with interpolation factor 4 where each horizontal box represents a 4-tap FIR filter with coefficients r_n

Table 3.2 summarizes the LUT and MAC operations required for each interpolation method. Interpolation produces L paths of data which increase the number of MAC operations by a factor of L . Interpolation also increases the number of samples in the reference waveform and hence MAC operations by another factor of L . However, since zero padding interpolation adds zeros which do not require MAC, zero padding interpolation is only increased by one factor of L . Test results presented in Section 5.1.2 show a comparison of each method's performance when used with a matched filter. From the test results, zero padding interpolation offers a great combination of increased accuracy with only a small increase in processing. The amount of resources required to implement the zero padding interpolation method versus other interpolation methods is explored in Section 4.1.2.

Table 3.2: Comparison of interpolation methods used with matched filtering showing the number of operations required to implement where L is the interpolation factor and N is the number of samples of the high resolution reference waveform

Interpolation Method	LUT Ops	MAC Ops
None	0	N/L
Zero Padding	0	N
Nearest	0	$N \times L$
Linear	L	$N \times L$
Cubic	L	$N \times L$

3.2.2 Hardware Design

The matched filter with zero padding interpolation pulse detection method can be realized in hardware using multiple parallel FIR filters and peak detector logic. As shown in Table 3.1, the longest reference waveform for a typical LADAR system with an ADC operating at 1 GHz sampling rate and interpolation factor of 32 is approximately 512 samples. Therefore at 1 GHz, the architecture must support up to 512 GMAC/s. The bit resolution of the ADC currently used in the AFRL LADAR system is 12-bits. In order to maintain accuracy of matched filter calculations, the bit widths of the inputs and coefficients are set to 12-bits. To support the combinatorial logic used in the 12-bit multipliers and avoid error due to clock slack, the clock rate of the FPGA is lowered to $\frac{1}{4}$ of the input sampling rate, or 250 MHz. Since the clock rate of the FPGA is less than the sampling rate of the ADC, four inputs must be taken every clock cycle.

This produces four independent data paths, as shown in Figure 3.7. As explained in Section 3.2.1, each data path requires L parallel FIR filters to perform interpolation. Therefore with an interpolation factor of 32, 128 FIR filters must be realized in parallel. The only increase in output latency introduced by interpolation is a few extra clock cycles required to detect the peak output intensity of the 128 outputs. The peak detector utilizes a counter and a comparison tree to index each pulse peak and to keep output delay time to a minimum. Figure 3.7 shows the hardware layout of the proposed design. The output of the peak detector is stored in local registers which can be read by external entities.

Figure 3.7: FPGA hardware realization showing the amount of MAC operations required

3.2.3 Summary

In summary, accuracy of a real-time LADAR system can be improved by interpolating and matched filtering captured LADAR data. In order to maintain real-time availability, a

hardware design can be optimized to complete numerous MAC operations in parallel with short output latency. The customized hardware design can be realized on an FPGA, described using VHDL. Matched filtering reduces noise and allows specific pulse shapes to be detected. And combining zero padding interpolation with matched filtering improves the time resolution of the sampled data while maintaining relatively low MAC operations.

CHAPTER FOUR

METHODOLOGY

This chapter explains the design and procedure of experiments executed to verify the effectiveness of pulse processing techniques described in Sections 2.2, 3.1, and 3.2. Accuracy of the range estimation algorithms is verified with post-processing experiments on captured LADAR data as described in Section 4.1. Timing requirements and available FPGA resources of the hardware design are verified using timing focused simulation experiments; this is described in Section 4.2. Section 4.3 describes a real-time LADAR system experiment performed to test integration and accuracy of the algorithms with an actual LADAR system. The methods used to analyze results of each experiment are also included in this chapter.

4.1 Range Accuracy Experiment Design

A range accuracy experiment was performed to verify and compare the accuracy of LADAR range estimation techniques. Range accuracy can be determined by implementing the calculations required for each algorithm on captured LADAR data. The digitized LADAR data was captured by an oscilloscope at 5 GHz and provided by the AFRL. The purpose of this experiment is to analyze the range calculation for each transmitted laser pulse; therefore post-capture calculations are sufficient for this experiment and time constraints are not addressed here. Using MATLAB [23], each algorithm was implemented on multiple sets of captured LADAR data. The MATLAB algorithm is presented in Appendix A.

The data collected and provided by the AFRL consists of multiple scans at three different approximate ranges: 38 m, 50 m, and 100 m. At each range, multiple targets were substituted for each scan. The targets include six different types of wire rotated at assorted angles as well as four different calibration panels. The various targets represent different backscatter effects, resulting in a collection of diverse backscatter pulse shapes. The LADAR data was captured using an oscilloscope with a 5 GHz sampling rate. The oscilloscope saved seventeen scan repetitions on each target in *csv* format. The LADAR scan data file was imported into MATLAB and split into seventeen separate LADAR data files for post-capture processing. The entire data sample consists of over 6,000 data files, where each file contains a transmitted pulse and a backscattered pulse, as shown in Figure 4.1. Out of all the data collected, 1,938 files were captured at the 38 m

range. These files were down-sampled from a 5 GHz sampling rate to a 1 GHz sampling rate. By taking different start indices, down-sampling the data produces 7,752 data files with a 1 GHz sampling rate.

Figure 4.1: Example LADAR data sequence showing start and return pulses sampled at 1 GHz

The actual range to each target is not precisely measured, but a reference range can be calculated using a range estimation technique. The sampling rate of the oscilloscope gives a 0.2 ns sampling period resulting in about 50 samples for a 10 ns laser pulse. Since the oscilloscope captured the transmitted and backscattered LADAR pulses at a high sampling rate, the actual shape of each pulse is retained. A cubic spline interpolation with the MATLAB function, *spline*, can be used to accurately predict a finer resolution curve without distorting the shape. This is done for each scan to determine the reference range to each target.

With a known reference range, the pulse processing techniques explained in Chapter 2 and introduced in Chapter 3 can be implemented and compared to determine the range accuracy of each method. Four different range estimation techniques were used in this experiment: edge

detector, peak detector, matched filter, and matched filter with interpolation. Four separate MATLAB algorithms were created to read LADAR data files and implement the respective range estimation technique. A master algorithm was also created capable of calling each range estimation algorithm and repeating the test for multiple LADAR data files. The input to each algorithm is a sequence of sampled LADAR data with a known sampling rate. An example of LADAR input data is shown in Figure 4.1. The output of each algorithm is an estimation of range from the LADAR data input. An example of test results for this experiment is shown in Table 4.1. In order to compare the performance of each algorithm, the master MATLAB algorithm was used to execute the series of tests described next.

Table 4.1: Example pulse processing algorithm outputs with reference range of 38.0000 m.

Method	Sample Difference	Sampling Rate	Time-of-Flight	Range
Edge Detector	1285	5 GHz	257 ns	38.4215 m
Peak Detector	1275	5 GHz	255 ns	38.1225 m
Matched Filter	1270	5 GHz	254 ns	37.973 m
Matched Filter with Interp. Factor 32	40658	5 GHz	254.1125 ns	37.9898 m

4.1.1 Matched Filter Reference Waveform Test

A matched filter reference waveform test was implemented to evaluate the performance of the two matched filter algorithms and observe the effects of different reference waveform shapes. Six reference waveform models were included in this test, as shown in Figure 3.1. Pulse width estimation algorithms [1] were used to generate the correct width reference waveform model to match the particular LADAR system used to capture this test data. In addition to the reference waveform models, a custom reference waveform was created by averaging the multiple backscattered pulses captured in the LADAR data. The 7,752 LADAR data files were analyzed using both the matched filter and matched filter with interpolation algorithms. Each algorithm was repeated using the different reference waveforms described. Range accuracy is determined by calculating standard deviation and root mean square error (RMSE) of each algorithm's range estimation. Range accuracy is tabulated in Section 5.1.1.

Noise tolerance of each range estimation algorithm is evaluated to determine the effectiveness of each algorithm when captured LADAR data contains noise. If the backscattered pulse is hidden in noise, the pulse processing techniques may calculate inaccurate range estimation results. As explained in Section 2.4.2, matched filtering techniques can recover pulses from noisy data. The edge detector, peak detector, and matched filter algorithms were applied data sets that have very low backscatter pulse intensities hidden in noise. The percent of accurate calculations is determined for each method and also presented.

Improvement in range accuracy as a function of the reference waveform's amplitude was also evaluated for each algorithm by varying the scalar multiple of the reference waveform from 0.1 to 10. Figure 4.2 illustrates how varying the scalar multiple changes the uniform size of the waveform but does not alter the shape or curvature. As shown in Section 5.1.1, this test verifies that the amplitude of each reference waveform has no effect on the range estimation. Instead, only the shape and sample resolution of the reference waveform are significant.

Figure 4.2: Reference waveforms showing effects of scalar multiple

4.1.2 Interpolation Method and Factor Test

An interpolation method test was implemented to compare the performance of the range estimation algorithms using different interpolation methods as described in Section 3.2. The 7,752 LADAR data files were analyzed in MATLAB using all four range estimation algorithms. The interpolation and matched filter with interpolation algorithms were repeated on the same LADAR data using four different interpolation methods: zero padding, nearest, linear, and Cubic interpolation. Range accuracy determined by standard deviation and root mean square error of range for each method is tabulated in Section 5.1.2. Improvement in accuracy as a function of the interpolation factor was also evaluated for each interpolation method by varying the interpolation factor from 1 to 64 and plotting the standard deviation of range.

4.1.3 Sampling Rate Test

A sampling rate test was used to evaluate the performance of the range estimation algorithms when LADAR data is sampled at lower rates. The importance of this test is to identify cost, design simplicity, and range accuracy tradeoffs of the ADC used. The maximum sampling rate of the ADC on the LADAR system used in the real-time experiment is 1 GHz. Since the LADAR data used in this experiment was captured at 5 GHz, the data was down-sampled to lower sampling rates ranging from about 250 MHz to 5 GHz. All four range estimation algorithms were used in this test with all 7,752 LADAR data files. Test results are tabulated and plotted in Section 5.1.3.

4.2 Implementation Experiment Design

A simulation experiment was performed to verify that the hardware design proposed in Section 3.2 can indeed be realized on computer architecture and satisfy time constraints. The matched filter with interpolation implementation was realized in VHDL, included in Appendix B, and designed for operation on a Stratix IV FPGA. Altera's FIR Compiler 11.1 MegaWizard [50] was used to create an FIR filter with reloadable coefficients. The MegaWizard creates VHDL components based on configurable parameters such as sampling rate, clock rate, interpolation factor, default coefficients, and bit width. The coefficients were set to the sample-reversed reference waveform desired in each test. An external enable pulse used to trigger the laser was used to reset a clock cycle counter in VHDL. Comparison logic was added to the design to find the maximum matched filter output intensity in the transmission and backscattered

windows. The start pulse intensity, start pulse index, return pulse intensity, and return pulse index are registered and held for one repetition rate. The design was simulated with ModelSim [25] and the input and output signals were plotted against time estimated by the simulation tools. In addition to the timing constraints, range accuracy of the VHDL algorithm was tested and compared to the results of the MATLAB range accuracy experiment using the same input data files.

4.2.1 FPGA Implementation Test

A hardware test was conducted to verify that the matched filter with interpolation range estimation hardware design can be implemented in VHDL on a custom FPGA architecture. The design was altered to produce four designs, each containing a different interpolation factor. The interpolation factors used in this test were ranged from 1 (which is no interpolation) up to 32. As explained in Section 3.2, higher interpolation rates require more circuitry and time to implement. Quartus II software [24] was used to synthesize the VHDL and the details from the compilation report are presented in Section 5.2.1. This test verified the Altera Stratix IV FPGA contains enough resources to host the hardware design.

4.2.2 Time Latency and Range Accuracy Test

A test was performed to determine the time latency of the matched filter with interpolation range estimation hardware design on a custom FPGA architecture and to verify range accuracy. The *vho* and *sdo* files produced by the Quartus II synthesizer were used by ModelSim to simulate each design. The ModelSim simulation utilized 10 randomly selected LADAR data files that were analyzed in the MATLAB sample rate test to verify that the FPGA calculations match the MATLAB calculations. The simulation takes about 30 minutes per design to complete and there are 28 designs simulated for each file (7 interpolation factors at 2 different sampling rates using both 10-bit and 12-bit precision). Since the ModelSim simulation experiment is very time consuming, only 10 files were selected. Using simulation plots, the time required to perform matched filtering and identify the sample difference between transmitted and backscattered pulses is measured for each design. The most complex design has only 22 clock cycles output latency in capturing the peak samples. Simulation results are presented in Section 5.2.2.

The ModelSim simulation results were further analyzed and the calculated sample difference between transmitted and backscattered pulses was recorded. Results are compared to the MATLAB range estimation results for the same LADAR data files and are also presented in Section 5.2.2. Bit precision of the ADC used in the actual LADAR system is fixed at 12-bits; however the bit precision of the reference waveform coefficients were selected as a tradeoff with device logic utilization and range accuracy. Simulations were performed with both 10-bit and 12-bit coefficient precision.

4.3 Real-Time LADAR System Experiment Design

A real-time experiment was performed on actual LADAR system hardware to verify the matched filter with interpolation range estimation realization properly performs and satisfies range accuracy and time requirements. An AFRL custom LADAR system realized using an Altera Stratix IV FPGA was used in this experiment. The matched filter VHDL design verified in the simulation experiment was loaded as a component on the Stratix IV FPGA. The LADAR system uses an external proprietary optical receiver and ADC to capture laser data. The ADC has the ability to operate at 247.5 MHz, 495 MHz, or 990 MHz. The clock signal used by the FPGA operates at $\frac{1}{4}$ the sampling rate. Since the clock rate is slower than the sampling rate, more than one sample must be processed every clock cycle. Therefore each clock cycle 4 parallel outputs are captured from the FPGA external pins. The hardware design of the FPGA is shown in Figure 3.7. The component layout of the experiment is shown in Figure 4.3. Pictures of the real-world experiment are presented in Appendix C.

In order to create a repeatable experiment, an optical laser pulse generator was used as the laser source. The laser generator was set to create two pulses, representing a transmission pulse and a backscattered pulse. The time between the two pulses was set to 500 ns for one series of tests and then changed to 1000 ns for another series of tests. This was done to ensure the results are consistent and do not float over a period of time. Fiber optic cable was used to route the output of the laser generator directly to the optical receiver. The voltage output of the optical receiver was first connected to an oscilloscope to record the data at 40 GHz and measure the reference time between pulses. Next, the output of the optical receiver was connected to the ADC to execute the experiment. The parameters of the LADAR system used in this experiment are shown in Table 4.2.

Figure 4.3: Component layout of real-time experiment

Table 4.2: LADAR system parameters and settings for real-time experiment

LADAR System Parameters	
Laser Source	Berkeley Nucleonics Corp. 6040 Universal Pulse Generator
Laser Pulse Width	10 ns
Laser Wavelength	1064 nm
Time-of-Flight	499/999 ns
Repetition Rate	10 kHz
Image Scan	100 stationary transmissions
ADC Sampling Rate	247.5/495/990 MHz
FPGA Clock Rate	61.875/123.75/247.5 MHz

In addition to the VHDL matched filter with interpolation design, the FPGA used in this experiment also contains a system on a programmable chip (SOPC). The SOPC has the ability to execute embedded C code. The embedded code was altered to continuously read memory registers of the matched filter design containing start pulse index, start pulse intensity, return pulse index, and return pulse intensity. The register values are calculated and latched for every laser transmission. For this experiment, the laser transmission is controlled by an external signal generator operating at 10 kHz. The signal generator is connected to both the FPGA and optical laser generator, as shown in Figure 4.3. The embedded code stores the data captured from the matched filter registers in local arrays. After each scan is complete, the arrays of data are offloaded to a computer and stored in *mat* format. An example of test parameters and results for this experiment is shown in Table 4.3. Test results such as range and time-of-flight are calculated from data collected. The test is repeated to capture at least 100 range values. The value of 100 was chosen arbitrarily to characterize a complete sample set. It is believed that standard deviation and RMSE calculated from 100 data samples will accurately depict the test results.

Table 4.3: Example test parameters and results of one laser transmission

Example Test Results of Laser Transmission		
Test Parameters	Sampling Rate	990 MHz
	Interpolation Factor	32
	Reference Waveform	16 ns Gaussian with $2e^{-9}$ standard deviation
	Time-of-Flight	999 ns
Data Collected	Start Intensity	651565
	Start Index	23358
	Return Intensity	2362886
	Return Index	55006
Test Results	Sample Difference	31648
	Time Difference	998.9899 ns
	Range	149.3490 m

Four matched filter with interpolation architectural designs were separately tested in this experiment, each using different interpolation factors: 1, 4, 16, and 32 and each using different ADC sampling rates: 247.5 MHz, 495 MHz, and then 990 MHz. At each sampling rate, the

resolution of the matched filter reference waveforms was changed to appropriately match the expected pulse resolution after interpolation. Table 4.4 shows the 15 reference waveforms that were used in each test. The Gaussian waveform model performed the best, as presented in Section 5.3. Results from 100 laser transmissions were collected for each reference waveform and are included in Appendix D. Mean, standard deviation, and root mean square of range were calculated from each scan. An example of test results from one scan is shown in Table 4.5. The test was conducted with approximately 500 ns between the two pulses, and repeated with approximately 1000 ns between pulses.

Table 4.4: Reference waveforms models used in LADAR experiment

Model	Length	Description
Average Return	10 ns	Average of multiple captured pulses
Gaussian	6 ns	Stan. Dev. = $2e^{-9}$
		Stan. Dev. = $10e^{-9}$
	10 ns	Stan. Dev. = $2e^{-9}$
		Stan. Dev. = $10e^{-9}$
	16 ns	Stan. Dev. = $2e^{-9}$
		Stan. Dev. = $10e^{-9}$
Poisson	16 ns	Variance = 23
		Variance = 42
Inverse Parabola	6 ns	Upside-down parabola with amplitude = 1
	10 ns	
	16 ns	
Square	6 ns	Square pulse with amplitude = 1
	16 ns	
Impulse	1 sample	Output = Input

The effectiveness of the matched filter with interpolation design can be compared to peak detector and sinc interpolation range estimation methods. The current range estimation method used on this AFRL LADAR system is a peak detector. A prototype of an alternative design currently in research by the AFRL utilizing sinc interpolation to enhance range accuracy is also implemented in VHDL for the Stratix IV FPGA. The sinc interpolation design constructs a higher resolution signal by evaluating the sinc function. This range estimation design was also applied to the 500 ns and 1000 ns tests. Section 5.3 presents the range accuracy results of sinc

interpolation method compared to the original peak detector method results and the matched filter with interpolation factor of 32 results. Time requirements of all the designs were satisfied by verifying all 100 laser transmissions were captured on consecutive reads.

Table 4.5: Test results printout of laser scan using 1 GHz sampling rate, interpolation factor of 32, 6 ns Gaussian reference waveform with stan. dev. of $2e^{-9}$

Example Test Results of Laser Scan
16tap_1000MHz_L32_0512tap_06ns_gaus2.txt start ind: 23153.793 +- 40.141 samples start amp: 601609.828 +- 3.6% counts return ind: 54827.207 +- 39.352 samples return dly : 999.792 +- 0.180 ns return net: 149.869 +- 0.027 m return amp: 1982311.138 +- 2.0% counts rms_dly : 0.278 ns rms_rng : 0.042 m system offset 0.262 ns, 0.039 m, 0.026% percent valid: 100.0

CHAPTER FIVE

RESULTS

The results of the experiments explained in Chapter 4 are presented in this chapter. All range calculations use Equation 2.1 and the same approximated speed of light constant 2.998×10^8 m/s. Section 5.1 presents results from range accuracy experiments and compares the performance of the different range estimation methods used in the experiments. Section 5.2 includes results from experiments testing the ability to implement matched filtering on hardware. Section 5.3 includes experiment results from testing a real-world LADAR system utilizing the hardware design proposed in Section 3.2.2.

5.1 Range Accuracy Experiment Results

Results from the range accuracy experiment are presented in this section. MATLAB was used to conduct the range accuracy experiment to verify and compare the accuracy of LADAR range estimation techniques. The tests described in Section 4.1 were performed and the results are presented in the following sections.

5.1.1 Matched Filter Reference Waveform Test Results

Results from the matched filter reference waveform test show that using matched filtering instead of peak detection can increase range accuracy. A comparison of range accuracy using different reference waveform models is presented in Table 5.1. By comparing the standard deviation and RMSE of each pulse detection method, it is clear the matched filter algorithm had the lowest error in range. The percent of accurate calculations measures noise tolerance and is determined by the number of LADAR data files that are within one meter of the reference range. The matched filter algorithm outperformed both the edge detector and peak detector in all categories. This is because the matched filter smooths the input data and eliminates noise surrounding the peak of the pulse. Also, matched filtering uses the entire 10 ns shape of the pulse to find the peak. Furthermore, by comparing the different reference waveforms used with matched filtering, it is shown the inverse parabola reference waveform performed best overall in this test. This is because the backscatter pulse shape of the majority of the LADAR input data files closely resembles a parabola waveform.

Table 5.1: Matched filter reference waveform test results showing comparison of edge detector, peak detector, and matched filter range estimation techniques with multiple reference waveforms

Reference Waveform	Std Dev Range (m)	RMSE Range (m)	Accurate Calc (%)
Edge Detector			
N/A	0.9354	2.5602	68
Peak Detector			
N/A	0.1135	0.1171	96
Matched Filter			
Square	0.1368	0.1416	100
Gaussian	0.0777	0.0778	100
Inverse Parabola	0.0770	0.0770	100
Poisson	0.0774	0.0777	100
Weibull	0.0860	0.0860	100
Average Return	0.0800	0.0800	100

Figure 5.1: Standard deviation of amplitude factors 0.1 to 10

All the reference waveforms used in the first part of this test were normalized to unity amplitude before matched filtering. However, the second part of this test scaled the amplitude of each reference waveform by a scalar factor and recalculated range using the scaled reference waveform. The results of the test, as shown in Figure 5.1, show that the amplitude of the reference waveform has no effect on matched filtering. The standard deviation of the range graph is flat, representing no change. As long as the amplitude of the reference waveform is uniformly changed, the amplitude of the matched filter output is relative to surrounding output samples. Therefore only the curvature and sample resolution of the reference waveform are significant.

5.1.2 Interpolation Method and Factor Test Results

Results from the interpolation method test show that including interpolation with matched filtering increases range accuracy. A range accuracy comparison of matched filtering using different interpolation methods is presented in Table 5.2. The number of LUT operations and the number of multiplier operations required to implement interpolation is determined by the interpolation factor and the length of the reference waveform. This test used an interpolation factor of 32 and a high resolution Gaussian reference waveform consisting of 320 samples for only a 10 ns pulse. The matched filter with interpolation algorithm has the lowest standard deviation and RMSE of range. Therefore it outperformed all three other methods. The Cubic interpolation method performed best in range accuracy, but requires a high number of multiplier operations to implement the interpolation. The zero padding interpolation method performed very well in range accuracy and requires a low amount of multiplier operations and no LUT operations, as expected from Section 3.2. Therefore as a cost/benefit tradeoff, zero padding is considered the most appropriate interpolation method for this design.

The matched filter with interpolation methods were repeated on the same LADAR data and the interpolation factor was varied from 1 to 50. The standard deviation of range for each interpolation factor is plotted in Figure 5.2. Raising the interpolation factor increases sample resolution allowing peak detection between original input samples. Therefore standard deviation of range decreases and range accuracy increases. A simulation of matched filtering with zero padding interpolation was conducted on 100 LADAR files at interpolation factors 1 to 200. As seen in the simulation results in Figure 5.3, standard deviation decreases as the interpolation factor increases. Therefore an appropriate interpolation factor must be selected as a tradeoff of range accuracy and number of multipliers required.

Table 5.2: Interpolation method test results showing comparison of edge detector, peak detector, matched filter, and matched filter with interpolation range estimation techniques with multiple interpolation methods

Interpolation Method	Std Dev Range (m)	RMSE Range (m)	LUT Ops	Multiplier Ops
Edge Detector				
N/A	0.9354	2.5602	0	0
Peak Detector				
N/A	0.1135	0.1171	0	0
Matched Filter (Gaussian)				
None	0.0777	0.0778	0	10
Matched Filter (Gaussian) with Interpolation Factor 32				
Zero Padding	0.0534	0.0538	0	320
Nearest	0.0526	0.0532	0	10240
Linear	0.0522	0.0529	32	10240
Cubic	0.0528	0.0526	32	10240

Figure 5.2: Standard deviation of interpolation factors 1 through 50

Figure 5.3: Standard deviation of interpolation factors 1 through 200

5.1.3 Sampling Rate Test Results

Results from the sampling rate test show that increasing the sampling rate of the LADAR system increases range accuracy. A range accuracy comparison of range estimation algorithms on data sampled at different sampling rates is presented in

Table 5.3. Data captured at 5 GHz was down-sampled to 1 GHz, 500 MHz, and 250 MHz to observe the effect on range accuracy at lower sampling rates. The number of multiplier operations required to implement matched filtering is also presented. Range accuracy of each estimation method increases as the sampling rate increases. Matched filtering with interpolation outperformed the other methods at each sampling rate tested. The results are as expected because at all sampling rates interpolation increases the sample resolution enabling super-sample accuracy. However, more multiplier operations are required for the matched filtering methods at higher sampling rates. The results also show that the standard deviation and RMSE of the 500

MHz matched filter with interpolation method is only slightly less than the 5 GHz matched filter with interpolation method.

Table 5.3: Sampling rate test results showing comparison of edge detector, peak detector, and matched filter range estimation techniques with multiple starting sampling rates

Sampling Rate	Std Dev Range (m)	RMSE Range (m)	Multiplier Ops
Edge Detector			
250 MHz	1.2041	1.5445	0
500 MHz	1.2495	2.0549	0
1 GHz	0.9354	2.5602	0
5 GHz	0.4882	2.7395	0
Peak Detector			
250 MHz	0.2618	0.2626	0
500 MHz	0.1887	0.1898	0
1 GHz	0.1135	0.1171	0
5 GHz	0.1019	0.1062	0
Matched Filter (Gaussian)			
250 MHz	0.2422	0.2422	3
500 MHz	0.1526	0.1525	5
1 GHz	0.0777	0.0778	10
5 GHz	0.0521	0.0521	50
Matched Filter (Gaussian) with Interpolation Factor 32 (Zero Padding)			
250 MHz	0.0748	0.0748	96
500 MHz	0.0560	0.0560	160
1 GHz	0.0534	0.0538	320
5 GHz	0.0504	0.0504	1600

5.2 Implementation Experiment Results

The results from the implementation experiment are presented in this section. ModelSim was used to experimentally verify the matched filter with interpolation range estimation technique can be implemented in real-time. The tests described in Section 4.2 were performed and the results are presented in the following sections.

5.2.1 FPGA Implementation Test Results

The real-time matched filter with interpolation range estimation technique was successfully realized using custom computer architecture on an Altera Stratix IV FPGA. Quartus II was used to compile and synthesize the custom design. Compilation results for interpolation factors 1, 4, 16, and 32 are presented in Table 5.4. Results show as the interpolation factor increases, more logic is required to build the design. As explained in Section 3.2, increasing the interpolation factor requires more multiplication, addition, and comparison operations. Therefore the results are as expected; more resources are required at higher interpolation factors.

Table 5.4: Compilation results of matched filter VHDL implementation showing logic used to build design and percentage of Stratix IV EP4SGX 230 device logic used

Interpolation Factor	Combinational ALUTs (%)	Memory ALUTs (%)	Logic Registers (%)	DSP blocks (%)
Matched Filter (12-bit Coefs) with Interpolation (Zero Padding)				
1	20	34	56	20
4	21	35	64	40
16	26	37	66	100
32	35	37	72	100

5.2.2 Time Latency and Range Accuracy Test Results

After the matched filter with interpolation design was synthesized, ModelSim was used to simulate the design in order to observe time latency and range accuracy of the VHDL design. Ten of the input LADAR data files used in the MATLAB simulation test were randomly selected, transformed to VHDL signal format, and used as the input to the matched filter. The peak in the start window and the peak in the backscatter window were identified by the hardware design simulation and the difference between the peaks were recorded. The input and output signals were plotted against time and output latency in clock cycles was also recorded. 28 different architectural designs were simulated. 14 designs were implemented using 10-bit precision on the reference waveform and the other 14 designs had 12-bit precision. The 14 designs consists of 7 different interpolation factors designed for a 500 MHz sampling rate and the same 7 interpolation factors designed for a 1 GHz. MATLAB results are used to calculate RMSE. Results are presented in Table 5.5 and Table 5.6.

Table 5.5: Simulation of VHDL implementation showing clock cycle latency and time-of-flight (TOF) accuracy with 500 MHz sampling rate compared to MATLAB results

500 MHz Sampling Rate and 125 MHz Clock Rate				
Interpolation Factor	Avg. MATLAB TOF (samples)	Avg. ModelSim TOF (samples)	RMSE TOF (samples)	Clock Cycles Required
Matched Filter (10 Bit Ref Waveform) with Interpolation (Zero Padding)				
1	128.5	128.5	0.0	9
2	255.5	255.5	0.0	11
4	509.4	509.4	0.0	13
8	1019.4	1019.4	0.0	15
16	2041.7	2039.9	0.8	17
32	4077.9	4079.2	2.2	19
64	8154.3	8161.3	8.4	21
Matched Filter (12 Bit Ref Waveform) with Interpolation (Zero Padding)				
1	128.5	128.5	0.0	9
2	255.5	255.5	0.0	11
4	510.6	509.4	0.0	13
8	1019.4	1019.4	0.0	15
16	2041.7	2041.7	0.0	17
32	4077.9	4077.9	0.0	19
64	8154.3	Cannot be sim.	Cannot be sim.	Cannot be sim.

Table 5.6: Simulation of VHDL implementation showing clock cycle latency and time-of-flight (TOF) accuracy with 1 GHz sampling rate compared to MATLAB results

1 GHz Sampling Rate and 250 MHz Clock Rate				
Interpolation Factor	Avg. MATLAB TOF (samples)	Avg. ModelSim TOF (samples)	RMSE TOF (samples)	Clock Cycles Required
Matched Filter (10 Bit Ref Waveform) with Interpolation (Zero Padding)				
1	255.8	255.8	0.0	10
2	510.6	510.6	0.0	12
4	1019.5	1019.5	0.0	14
8	2041.3	2039.0	1.9	16
16	4076.6	4074.1	2.1	18
32	8153.6	Cannot be simulated	Cannot be simulated	Cannot be simulated
64	16305.1	Cannot be simulated	Cannot be simulated	Cannot be simulated
Matched Filter (12 Bit Ref Waveform) with Interpolation (Zero Padding)				
1	255.8	255.8	0.0	10
2	510.6	510.6	0.0	12
4	1019.5	1019.5	0.0	14
8	2041.3	2041.3	0.0	16
16	4076.6	4076.6	0.0	18
32	8153.6	Cannot be simulated	Cannot be simulated	Cannot be simulated
64	16305.1	Cannot be simulated	Cannot be simulated	Cannot be simulated

At low interpolation factors the VHDL calculations match MATLAB calculations exactly. At higher interpolation factors, bit precision and truncation creates a slight error in the VHDL calculations. Increasing the bit precision of the FIR filter coefficients from 10-bit to 12-bit reduces error at high interpolation factors. The extremely high interpolation factor designs require very complex logic to implement matched filtering with interpolation and ModelSim ran out of memory and crashed when trying to simulate. The ModelSim crash only occurs with an effective sampling rate after interpolation of 32 GHz or higher. The simulation was repeated on multiple computers with the same result.

5.3 Real-Time LADAR System Experiment Results

The full LADAR system experiment described in Section 4.3 was performed on the 500 ns delay and the 1000 ns delay pulses generated by the optical laser generator. The sampling rate of the ADC was adjusted to 3 different sampling rates. The actual sampling rates were measured as 247.5 MHz, 494 MHz, and 990 MHz. The time-of-flight variable used to calculate all range values are determined by counting the number of samples between each pulse and dividing by the actual sampling rate.

5.3.1 500 ns Range Accuracy Test Results

MATLAB post-processing algorithms were used to determine a 499 ns reference delay from 40 GHz oscilloscope data, and reference range was calculated to be 78.8001 m. The real-time LADAR system experiment was executed and range was calculated 100 times for each method. Mean, standard deviation, root mean square error, and range offset were calculated and are presented in Table 5.7. The standard deviation of range for each method is plotted in Figure 5.5, and root mean square error of range is plotted in Figure 5.6. Root mean square error and average range offset were calculated using the reference range determined from the 40 GHz oscilloscope data.

As the sampling rate increases, standard deviation and root mean square error decreases. The matched filter with interpolation factor of 32 outperformed both the peak detector and the peak detector using sinc interpolation at all 3 sampling rates. Test results from each individual test are included in Appendix D. The 16 ns Gaussian reference waveform model with standard deviation of 2 ns produced the best results overall. Range accuracy improvement of replacing the

original peak detector method with the matched filter with interpolation method is calculated for each sampling rate. The percent of range accuracy improvement is plotted in Figure 5.4.

Table 5.7: LADAR system 500 ns test results comparing peak detector, sinc interpolation peak detector, and matched filter with interpolation factor 32 range estimation techniques with reference range of 74.8001 m

Sampling Rate	Mean Range (m)	Std Dev Range (m)	RMSE Range (m)	Avg Offset (m)
Peak Detector				
247.5 MHz	74.9380	0.2683	0.2908	0.1379
495 MHz	74.8016	0.0327	0.0418	0.0015
990 MHz	74.8093	0.0390	0.0399	0.0092
Sinc Interpolation Peak Detector				
247.5 MHz	74.7751	0.1349	0.1450	-0.0250
495 MHz	74.8064	0.0225	0.0313	0.0063
990 MHz	74.8116	0.0189	0.0252	0.0115
Matched Filter with Interpolation Factor 32				
247.5 MHz	74.7752	0.0389	0.0461	-0.0249
495 MHz	74.8014	0.0174	0.0174	0.0013
990 MHz	74.8016	0.0134	0.0134	0.0015

Figure 5.4: Percent of range accuracy improvement of standard deviation and root mean square error at each sampling rate

Figure 5.5: LADAR system 500 ns test results comparing standard deviation

Figure 5.6: LADAR system 500 ns test results comparing root mean square error

5.3.2 1000 ns Range Accuracy Test Results

MATLAB post-processing algorithms were used to determine a 999.58 ns reference delay from 40 GHz oscilloscope data, and reference range was calculated to be 149.837 m. The real-time LADAR system experiment was executed and range was calculated 100 times for each method. Mean, standard deviation, root mean square error, and range offset were calculated and are presented in Table 5.8. The standard deviation of range for each method is plotted in Figure 5.7, and root mean square error of range is plotted in Figure 5.8.

Matching the 500 ns test, the matched filter again outperformed both the peak detector and the peak detector using sinc interpolation at all 3 sampling rates. The same 16 ns Gaussian reference waveform model produced the best results overall. Unlike the 500 ns test, at the 247.5 MHz sampling rate the sinc interpolation has very high standard deviation and root mean square error and the mean range is significantly lower than the other methods. Therefore it is believed that the sinc interpolation peak detector method is collecting a few very low range estimates due to noise which is skewing the data. The percent of range accuracy improvement of matched filtering over the peak detector method is plotted in Figure 5.4. The matched filter method performed exceptionally well at 495 MHz due to the reference waveform accurately depicting the shape of the return pulse. The results of the 1000 ns test concur with the 500 ns test results.

5.3.3 Summary

In summary, the experiments conducted show that the proposed hardware design improves range accuracy of the existing real-time LADAR system. As predicted, matched filtering eliminates erroneous range estimations caused by noise. And combining interpolation with matched filtering improves time resolution and therefore improves range accuracy. The interpolation method test results showed that zero padding interpolation offers a great performance tradeoff of increased range accuracy with a relatively low number of MAC operations. The FPGA hardware design was successfully implemented on an Altera Stratix IV EP4SGX 230 device. The final experiment utilized the actual hardware design with a LADAR system and range accuracy was increased at three different sampling rates and two different ranges.

Table 5.8: LADAR system 1000 ns test results comparing peak detector, sinc interpolation peak detector, and matched filter with interpolation factor 32 range estimation techniques with reference range of 149.8370 m

Sampling Rate	Mean Range (m)	Std Dev Range (m)	RMSE Range (m)	Avg Offset (m)
Peak Detector				
247.5 MHz	149.7666	0.2728	0.2803	-0.0705
495 MHz	149.8589	0.1023	0.1047	0.0219
990 MHz	149.8880	0.0442	0.0684	0.0510
Sinc Interpolation Peak Detector				
247.5 MHz	149.6302	0.5648	0.6015	-0.2069
495 MHz	149.8813	0.0205	0.0489	0.0442
990 MHz	149.8885	0.0165	0.0547	0.0514
Matched Filter with Interpolation Factor 32				
247.5 MHz	149.8284	0.0347	0.0357	-0.0087
495 MHz	149.8395	0.0134	0.0136	0.0025
990 MHz	149.8118	0.0146	0.0291	-0.0252

Figure 5.7: LADAR system 1000 ns test results comparing standard deviation

Figure 5.8: LADAR system 1000 ns test results comparing root mean square error

CHAPTER SIX

CONCLUSION

This chapter concludes the thesis research by highlighting key results and making recommendations for further research. Section 6.1 outlines major conclusions drawn from research and experiment results. Section 6.2 discusses options for future research related to this thesis research.

6.1 Conclusions from Results

Several conclusions can be drawn based on the experiments conducted in the scope of this research. As demonstrated in other LADAR research, matched filtering LADAR data improves range accuracy and reduces variance of multiple scans [1]. Matched filtering also has the ability to recover a pulse hidden in noise. Therefore range estimation in noisy LADAR data is reliable with matched filtering. Accuracy of the matched filter is limited by the contrasts in the shape of the reference waveform and the shape of the actual backscattered pulse. If the reference waveform does not accurately depict the backscattered pulse then the time-of-flight measurement may be off a few samples. As discussed in Section 6.2, further research could characterize waveform shapes for different occurrences in order to anticipate a particular backscatter pulse shape to use as the reference waveform.

Range accuracy of the matched filtering algorithm is also limited by the sampling rate of the ADC used to collect LADAR data. The matched filter with interpolation method improves range accuracy by inserting additional data points to essentially up-sample the data set to a higher sampling rate. Based on the test results in Section 5.3, range accuracy of the matched filter with interpolation method is better than all other methods simulated, including the peak detector, peak detector with sinc interpolation, and the matched filter method.

The hardware design for the matched filter with interpolation successfully executes in real-time with less than 20 clock cycles output latency. Throughput of the design is unity, so the system does not bottleneck and can be used to continuously scan. The maximum interpolation factor achieved on the system is 32. The interpolation factor is limited by the amount of space available on the FPGA. Alternate FPGA devices with more dedicated logic should be able to achieve higher interpolation factors. From the simulation results in Section 5.2.2, as the

interpolation factor increases, output latency and range accuracy increase. Therefore the choice in interpolation factor would rely mainly on the applications requirements for speed, range accuracy, and FPGA hardware. The highest performing design tested operated with a 1 GHz ADC, interpolation factor of 32, and a reference waveform consisting of 512 samples. Therefore the FPGA is performing 16 MAC per sample at 32 GSPS, or 512 GMAC/s. This design reduced the standard deviation of range by 67% when compared to the peak detector method. The similar matched filter with interpolation hardware design operating at 250 MHz reduced standard deviation of range by 87% when compared to the peak detector method.

The FPGA design is implemented with a combination of Altera's FIR Compiler 11.1 MegaWizard and custom VHDL logic. The FIR MegaWizard is utilized to perform zero padding interpolation and matched filtering. The custom VHDL logic routes signals from the ADC to the FIR component and performs peak detection on filtered signals to find the maximum intensities and corresponding indices of the transmitted and backscattered pulses. The FIR Compiler 11.1 MegaWizard successfully created a 512 tap FIR filter with interpolation factor 32 operating on a 250 MHz clock with output latency of less than 20 clock cycles.

6.2 Recommendations for Future Research

There are many options available to expand on the research performed in the scope of this thesis. To improve the reliability of the matched filter with interpolation method, research could be done to characterize reference waveforms based on the different lasers used in LADAR systems. By researching propagation and reflection effects on laser pulses, the shape of the backscattered pulse of a specific target could be anticipated. A deliverable of the research could be a database of different reference waveform pulse shapes for each laser model, optical receiver model, target, and scenario. It may be possible to use matched filtering to accurately recover a particular target, or reject a particular pulse such as backscattered light from dust storms. Another possible method for obtaining an accurate reference waveform could involve adjusting the algorithm to capture the shape of the previous laser pulse and use that shape as the reference waveform for the next scan.

Yet another option could be expanding the capability of the algorithm to detect not just the transmitted pulse and the backscattered pulse, but multiple backscattered pulses. The beam spread of a laser pulse may partially reflect off a closer object then reflect again off another

object. Detecting several backscattered pulses can be used in decision logic to determine multiple targets in a scene or determine the edge of an object. It may be possible to accomplish this using a double pulse shape as the reference waveform model in matched filtering.

Further research in modern hardware may result in an alternative architecture to implement matched filtering with interpolation. Even though the MegaWizard design satisfies design requirements, a custom design optimized for speed may be able to outperform the MegaWizard FIR filter. Also, the FPGA hardware design in this research utilized multiple parallel data paths to quickly complete calculations. However, modern digital signal processors are becoming more capable of completing the high number of MAC operations required for real-time matched filtering with interpolation on LADAR systems with fast ADCs.

Even though the matched filter with interpolation pulse processing method outperformed the other methods tested, alternative DSP algorithms not considered in this research may be able to achieve better range accuracy. A wide variety of DSP techniques are available to find peaks in discrete sample sets. Additional research and testing could involve applying more elaborate DSP to LADAR in order to further increase range accuracy. However, care must be taken to maintain real-time calculations.

APPENDIX A

MATLAB RANGE ESTIMATION ALGORITHMS

A.1 Time-domain cross-correlation

```
% cross-correlate
function cc = cross_corr( data, template )
    % Calculate Cross Correlation
    lt = length(template);
    len = length(data)-lt;
    cc = zeros(1,len);
    for n=0:1:len
        c=zeros(1,lt);
        for m=1:1:lt
            c(m)=template(m)*data(n+m);
        end
        cc(n+1)=sum(c);
    end
end
```

A.2 Edge-detection algorithm

```
function range = edge_detect(data,samp_period,gate,threshold)

    % Find Range
    sta_n_max = round( gate(2) / samp_period );
    ret_n_min = round( gate(3) * 2 / 2.99e8 / samp_period );
    I1=0; I2=0;

    for i = 1:sta_n_max
        if data(i) >= threshold
            I1=i;
            break;
        end
    end
    for i = I1+1+ret_n_min:length(data)
        if data(i) >= threshold
            I2=i;
            break;
        end
    end

    I_diff = I2-I1;
    range = I_diff*samp_period*2.99e8/2;

    time = 1e9*[0:samp_period:samp_period*(length(data)-1)]';
    hold on; plot(data); plot(threshold*ones(1,length(data)),'-r');
hold off;
end
```

A.3 DSP algorithm

```
function range =
DSP2(data,samp_period,gate,interp_method,filter_method,varargin)
    plot_stuff = false;
    % Used on one scan at a time
    % DSP2(data, samp_period, [sta_min sta_max ret_min ret_max],...
    %     'interp_method', 'filter_method', up_rate, pulse_width, sigma)

    % Parse
        if nargin > 5
            up_rate = varargin{1};
        else
            up_rate = 1;
        end
        if nargin > 6
            pulse_width = varargin{2};
        end
        if nargin > 7
            filter_var = varargin{3};
        end
        if nargin > 8
            cor = varargin{4};
        else
            cor = 'cor';
        end
    end

    % Interpolate Data
        switch interp_method,
            case 'none',
                up_rate = 1;
            case 'linear',
                x = 1:up_rate:length(data)*up_rate-(up_rate-1);
                xi = 1:1:length(data)*up_rate-(up_rate-1);
                data = interp1(x,data,xi);
            case 'nearest',
                x = 1:up_rate:length(data)*up_rate-(up_rate-1);
                xi = 1:1:length(data)*up_rate-(up_rate-1);
                data = interp1(x,data,xi,'nearest');
            case 'spline'
                x = 1:up_rate:length(data)*up_rate-(up_rate-1);
                xi = 1:1:length(data)*up_rate-(up_rate-1);
                data = interp1(x,data,xi,'spline');
            case 'zeros',
                data = upsample(data,up_rate);
            case 'DFT'
                data = interpft(data,length(data)*up_rate-(up_rate-1));
        end
        % Calculate new samp_period
        samp_period = samp_period / up_rate;

    % Matched Filter
        % get template
        switch filter_method,
            case 'none',
                plot_stuff = false;
                cc = data;
```

```

x = 0:samp_period:5e-9; % used for plot
template = ones(1,length(x)); % only used for plot
cor = 'none';
case 'gaus',
    plot_stuff = false;
    x = 0:samp_period:pulse_width; % sigma = 10e-9;
    template = gaussmf(x,[filter_var pulse_width/2]);
    template = (template-template(1));
    template = template./max(template);
case 'hybrid',
    % filter_var = [sigma1 pulse_width1 sigma2 pulse_width2]
    %               [3e-9 14e-9 8e-9 145e-9]
    sigma1 = filter_var(1);
    pulse_width1 = filter_var(2);
    sigma2 = filter_var(3);
    pulse_width2 = filter_var(4);
    x1 = 0:samp_period:pulse_width1/2;
    template1 = gaussmf(x1,[sigma1 pulse_width1/2]);
    template1 = (template1-template1(1));
    template1 = template1./max(template1);
    x2 = 0:samp_period:pulse_width2/2;
    template2 = gaussmf(x2,[sigma2 pulse_width2/2]);
    template2 = (template2-template2(1));
    template2 = template2./max(template2);
    template2 = template2(end:-1:1);
    template2 = template2(2:end);
    template = [template1 template2];
    template = template(1:pulse_width/samp_period);
    %plot(template)
case 'para'
    x = -pulse_width/2:samp_period:pulse_width/2-samp_period;
    template = (1-((4*x.^2)/(pulse_width^2)));
    %plot(template)
    plot_stuff=false;
    amp=0.02;
case 'poisson'
    up_rate = 1/samp_period / 1/1e9;
    N = round(pulse_width/1e-9);
    x = 0:1:filter_var*2; % pw needs to be around 30ns
    template = poisspdf(x,filter_var); % var needs to be around 23
    template = template./max(template);
    if filter_var-N/2 > 0
        lb = filter_var-N/2;
    else
        lb = 1;
    end
    if filter_var+N/2 < length(x)
        ub = filter_var+N/2;
    else
        ub = length(x);
    end
    template = template(lb:ub);
    x = 1:up_rate:length(template)*up_rate-(up_rate-1);
    xi = 1:1:length(template)*up_rate-(up_rate-1);
    template = interp1(x,template,xi,'spline');
    plot(template)
case 'weibull'

```

```

        x = 0:2/(pulse_width/samp_period):2;
        template = wblpdf(x,1,filter_var);
        %template = template(241:365);
    case 'square'
        x = 0:samp_period:pulse_width;
        template = ones(1,length(x));
    case 'return'
        plot_stuff = false;
        template_file = filter_var;
        load(template_file)
        template = rtemplate;
        %plot(template);
    end
    % Cross-correlate
    if strcmp(cor,'norm')
%       cc = norm_cross_corr(data,template);
        bits = 11;
        data = normalize(data, bits);
        %template = normalize(template, bits);
        cc = cross_corr(data,template);
    elseif strcmp(cor,'cor')
        cc = cross_corr(data,template);
    end

% Find Range
    sta_n_max = round( gate(2) / samp_period );
    ret_n_min = round( gate(3) * 2 / 2.99e8 / samp_period );
    [amp1,I1] = max(cc(1:sta_n_max));
    [amp2,I2] = max(cc(I1+1+ret_n_min:end));
    I_diff = I2+ret_n_min;
    time_diff_ns = I_diff*samp_period*1e9;
    range = I_diff*samp_period*2.99e8/2;
    if strcmp(interp_method,'zeros')
        plot_stuff=true;
    end
    time = 1e9*[0:samp_period:samp_period*(length(cc)-1)]';

end

```

A.4 Main MATLAB test script

```

% test_scan
% this file loops through data files and calls DSP2 to find the range
% then calls plot_test_results to create graphs of results
% run this file in the directory containing LADAR data files
clear all; clc; close all;

% customize this:
str      = 'test_scan_new_interp_test3';
max_file = 1000;
ivar     = 1:1:max_file;
xvar     = [1 2 4 8 32];
sigma    = 10e-9;    % Gaussian sigma
pulse_wid = 10e-9;    % Start pulse length (in ns)
gate     = [0 200e-9 20 120];    % start/range gate [ns ns m m]

```

```

                % [0 .5e-9 20 120];
                % [0 100e-9 0 120];
up_factor      = 32;          % Interpolation upsample factor
samp_rate      = 1e9;        % desired sample rate to use
temp_name      = 'ret_32GHz_10ns.mat'; %['20ns_para_10ns_1GHz_'
num2str(up_factor)];

% do not change anything below here
disp(['Save file as: ' str])
files = ls(fullfile(cd,'*.mat'));
imax = size(files,1);
tic
for i=1:length(ivar)

    file = strtrim(files(ivar(i),1:end));
    disp(['File number ' num2str(ivar(i))])
    info = load(file);
%     %interpolate input data
%     info.rate = 160e9;
%     up_rate = info.rate/40e9;
%     data = info.data;
%     x = 1:up_rate:length(data)*up_rate-(up_rate-1);
%     xi = 1:1:length(data)*up_rate-(up_rate-1);
%     info.data = interp1(x,data,xi,'spline');

    down_factor = info.rate/samp_rate;
    kvar = 1:down_factor-1; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if isempty(kvar)
        kvar =1;
    end
    %kvar=1;

    for j= 1:length(xvar)      % Loop through xvar
        disp(['xvar = ' num2str(xvar(j))])
        up_factor = xvar(j); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        for k = 1:length(kvar) % Loop through all scans, find range
            samp_offset = kvar(k);
            data = info.data(1+samp_offset:end);
            data = downsample(data,round(down_factor));
            samp_period = 1/samp_rate;

            %plot(data)
            time = 1e9*[0:samp_period:samp_period*(length(data)-1)];
%             figure; plot(time,data,'k','linewidth',1.2)
%             title(['LADAR Data Downsampled to ' num2str(samp_rate) ' Hz']);
%             xlabel('time (ns)'); ylabel('Amplitude');

            n=1;

            % sampling rate test, repeat at 250/500/1/5, no xvar
%             % edge detect
%             t{i,n}(j,k)=edge_detect(data,samp_period,gate,0.01); n=n+1;
%             % peak detect
%             t{i,n}(j,k)=DSP2(data,samp_period,gate,...
%                 'none','none'); n=n+1;
%             % MF

```

```

%           t{i,n}(j,k)=DSP2(data,samp_period,gate,...
%           'none','gaus',1,10e-9,10e-9); n=n+1;
%           % MF w/ Interp Gaus
%           t{i,n}(j,k)=DSP2(data,samp_period,gate,...
%           'zeros','gaus',32,10e-9,10e-9); n=n+1;

% interp method test, 1GHz data, xvar is up_factor 1-50
% MF
t{i,n}(j,k)=DSP2(data,samp_period,gate,...
    'none','gaus',1,10e-9,5e-9); n=n+1;
% MF w/ Interp Gaus
t{i,n}(j,k)=DSP2(data,samp_period,gate,...
    'zeros','gaus',up_factor,10e-9,5e-9); n=n+1;
% MF w/ Interp Gaus
t{i,n}(j,k)=DSP2(data,samp_period,gate,...
    'nearest','gaus',up_factor,10e-9,5e-9); n=n+1;
% MF w/ Interp Gaus
t{i,n}(j,k)=DSP2(data,samp_period,gate,...
    'linear','gaus',up_factor,10e-9,5e-9); n=n+1;
% MF w/ Interp Gaus
t{i,n}(j,k)=DSP2(data,samp_period,gate,...
    'spline','gaus',up_factor,10e-9,5e-9); n=n+1;

%           % amplitude test
%           % MF w/ Interp Square
%           t{i,n}(j,k)=DSP2(data,samp_period,gate,...
%           'zeros','square',up_factor,10e-9); n=n+1;
%           % MF w/ Interp Gaus
%           t{i,n}(j,k)=DSP2(data,samp_period,gate,...
%           'zeros','gaus',up_factor,10e-9,5e-9); n=n+1;

        end
    end
    disp(['Estimated time left = ', num2str((max_file-i)*toc/i/60/60),
'hrs'])
end
oldFolder = cd(['C:\Documents and Settings\jb\My
Documents\SPRING11\Thesis'...
    '\MATLAB\data_results']);
save(str)
cd(oldFolder);

plot_test_results;

```

APPENDIX B

VHDL RANGE ESTIMATION ALGORITHM

B.1 DSP top entity

```
-----
-- Title           :      Top entity of DSP thesis project
-- Description      :      Implement matched filtering, interpolation,
--                      and peak detection using components. Also
uses
--                      registers to connect to NIOS processor.
-----

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use work.dsp_pkg.all;

entity DSP is
    port (
        clk_250MHz           : in  std_logic    := '0';
        master_reset_n        : in  std_logic    := '0';
        ext_trig              : in  std_logic    := '0';
        input0                : in  std_logic_vector(INWIDTH-1
downto 0) := (others => '0');
        input1                : in  std_logic_vector(INWIDTH-1
downto 0) := (others => '0');
        input2                : in  std_logic_vector(INWIDTH-1
downto 0) := (others => '0');
        input3                : in  std_logic_vector(INWIDTH-1
downto 0) := (others => '0');
        address               : in  std_logic_vector (ADDWIDTH-1
downto 0);
        read_ena              : in  std_logic;
        read_data             : out std_logic_vector (REGWIDTH-1 downto
0);
        write_ena             : in  std_logic;
        write_data            : in  std_logic_vector (REGWIDTH-1 downto
0);
        coeff_in_areset       : in  std_logic    := '0'
    );
end entity DSP;
-----
-----
-----

architecture rtl of DSP is
    -- Peak detector signals
    signal sta_index_min      : std_logic_vector(OUTWIDTH-1 downto 0);
    signal sta_index_max      : std_logic_vector(OUTWIDTH-1 downto 0);
    signal ret_index_min      : std_logic_vector(OUTWIDTH-1 downto 0);
    signal ret_index_max      : std_logic_vector(OUTWIDTH-1 downto 0);
    signal sta_index          : std_logic_vector(OUTWIDTH-1 downto 0);
```

```

signal sta_value          : std_logic_vector(OUTWIDTH-1 downto 0);
signal ret_index          : std_logic_vector(OUTWIDTH-1 downto 0);
signal ret_value          : std_logic_vector(OUTWIDTH-1 downto 0);

-- Matched filter signals
signal ast_sink_data      : std_logic_vector(CHANS_IN*INWIDTH-1
downto 0);
signal ast_sink_ready     : std_logic;
signal ast_source_data    : std_logic_vector(CHANS_OUT*OUTWIDTH-1 downto
0);
signal ast_source_valid   : std_logic;
signal ast_source_error   : std_logic_vector(1 downto 0);

-- Register signals
signal reset_n            : std_logic;
signal done               : std_logic;
signal clk_125MHz         : std_logic;
signal clk                : std_logic;
signal clk_sel            : std_logic;
signal pixel_num          : std_logic_vector(OUTWIDTH-1 downto 0);
signal sta_window_min     : std_logic_vector(OUTWIDTH-1 downto 0);
signal sta_window_max     : std_logic_vector(OUTWIDTH-1 downto 0);
signal ret_window_min     : std_logic_vector(OUTWIDTH-1 downto 0);
signal ret_window_max     : std_logic_vector(OUTWIDTH-1 downto 0);
signal sta_index_lat      : std_logic_vector(OUTWIDTH-1 downto 0);
signal sta_value_lat      : std_logic_vector(OUTWIDTH-1 downto 0);
signal ret_index_lat      : std_logic_vector(OUTWIDTH-1 downto 0);
signal ret_value_lat      : std_logic_vector(OUTWIDTH-1 downto 0);
signal pixel_num_lat      : std_logic_vector(OUTWIDTH-1 downto 0);
signal current_index_out  : std_logic_vector(OUTWIDTH-1 downto 0);

-- Coef reloading signals
signal coef               : coef_slv_array(0 to MAXCOEFS-1);
signal reload_coef        : std_logic;
signal reg_coef           : std_logic;
signal num_coeffs         : std_logic_vector(REGWIDTH-1 downto 0);
signal coeff_reload_start : std_logic;
signal coeff_in_address   : std_logic_vector(COEFADDWIDTH-1 downto 0);
signal coeff_in_we        : std_logic_vector(0 downto 0);
signal coeff_in_data      : std_logic_vector(COEFDATAWIDTH-1 downto 0);

-- Laser edge detection signals
signal prev_prev_ext_trig : std_logic;
signal prev_ext_trig      : std_logic;
signal prev_prev_int_trig : std_logic;
signal prev_int_trig      : std_logic;
signal trig_ext_int       : std_logic;
signal int_trig           : std_logic;
signal laser_clk_ena      : std_logic;
signal reg0, reg1, done_re : std_logic;
signal debug1, debug2, debug3 : std_logic_vector(REGWIDTH-1 downto 0);

component registers is
  port (
    clk          : in  std_logic := '0';
    reset_n      : in  std_logic := '0';

```



```

        address          : in  std_logic_vector (ADDWIDTH-1 downto
0)      := (others => '0');
        read_ena         : in  std_logic      := '0';
        read_data        : out std_logic_vector (REGWIDTH-1 downto 0);
        write_ena        : in  std_logic      := '0';
        write_data       : in  std_logic_vector (REGWIDTH-1 downto 0)
:= (others => '0');

        sta_window_min   : out std_logic_vector(OUTWIDTH-1
downto 0);
        sta_window_max   : out std_logic_vector(OUTWIDTH-1
downto 0);
        ret_window_min   : out std_logic_vector(OUTWIDTH-1
downto 0);
        ret_window_max   : out std_logic_vector(OUTWIDTH-1
downto 0);
        sta_index        : in  std_logic_vector(OUTWIDTH-1
downto 0) := (others => '0');
        sta_value        : in  std_logic_vector(OUTWIDTH-1
downto 0) := (others => '0');
        ret_index        : in  std_logic_vector(OUTWIDTH-1
downto 0) := (others => '0');
        ret_value        : in  std_logic_vector(OUTWIDTH-1
downto 0) := (others => '0');
        pixel_num        : in  std_logic_vector(OUTWIDTH-1
downto 0) := (others => '0');
        clk_sel          : out std_logic;
        trig_ext_int     : out std_logic;
        debug1           : in
std_logic_vector(REGWIDTH-1 downto 0) := (others => '0');
        debug2           : in
std_logic_vector(REGWIDTH-1 downto 0) := (others => '0');
        debug3           : in
std_logic_vector(REGWIDTH-1 downto 0) := (others => '0');
        num_coeffs       : out std_logic_vector(REGWIDTH-1
downto 0);
        reload_coef      : out std_logic;
        int_trig         : out std_logic;
        coef             : out coef_slv_array(0 to
MAXCOEFS-1)
    );
end component registers;

component peak_detector is
    port (
        clk             : in  std_logic := '0';
        reset_n         : in  std_logic := '0';
        data             : in  std_logic_vector(CHANS_OUT*OUTWIDTH-1
downto 0);
        sta_window_min   : in  std_logic_vector(OUTWIDTH-1 downto
0);
        sta_window_max   : in  std_logic_vector(OUTWIDTH-1 downto
0);
        ret_window_min   : in  std_logic_vector(OUTWIDTH-1 downto
0);
        ret_window_max   : in  std_logic_vector(OUTWIDTH-1 downto
0);

```

```

        sta_index      : out std_logic_vector(OUTWIDTH-1 downto 0);
        sta_value      : out std_logic_vector(OUTWIDTH-1 downto 0);
        ret_index      : out std_logic_vector(OUTWIDTH-1 downto 0);
        ret_value      : out std_logic_vector(OUTWIDTH-1 downto 0);
        done           : out std_logic;
        current_index_out : out std_logic_vector(OUTWIDTH-1
downto 0)
    );
end component peak_detector;

component MF is
    port (
        clk           : in  std_logic              := '0';
        reset_n       : in  std_logic              := '0';
        coeff_in_clk   : in  std_logic              := '0';
        coeff_in_aret  : in  std_logic              := '0';
        coeff_in_address : in  std_logic_vector(COEFADDWIDTH-1 downto 0)
:= (others => '0');
        coeff_in_we    : in  std_logic_vector(0 downto 0)      := (others
=> '0');
        coeff_in_data  : in  std_logic_vector(COEFDATAWIDTH-1 downto 0)
:= (others => '0');
        ast_sink_data   : in  std_logic_vector(CHANS_IN*INWIDTH-1 downto
0)      := (others => '0');
        ast_sink_ready  : out std_logic;
        ast_sink_valid  : in  std_logic              := '0';
        ast_sink_error  : in  std_logic_vector(1 downto 0)     := (others
=> '0');
        ast_source_data : out std_logic_vector(CHANS_OUT*OUTWIDTH-1
downto 0);
        ast_source_ready : in  std_logic              := '0';
        ast_source_valid : out std_logic;
        ast_source_error : out std_logic_vector(1 downto 0)
    );
end component MF;

-----
-----
-----
begin
-----
-----
---Create reset_n signal-----
-----
        create_reset : process(master_reset_n, clk) is
        begin
            if master_reset_n = '0' then
                reset_n <= '0';
            elsif rising_edge(clk) then
                if laser_clk_ena = '1' then
                    reset_n <= '0';
                else
                    reset_n <= '1';
                end if;
            end if;
        end process;
end process;

```

```

-----
-----
---Coef reloading-----
-----

    coef_reload : process(master_reset_n, clk) is
    begin
        if master_reset_n = '0' then
            reg_coef      <= '0';
            coeff_reload_start <= '0';
        elsif rising_edge(clk) then
            reg_coef      <= reload_coef;
            if reg_coef = '0' and reload_coef = '1' then
                coeff_reload_start <= '1';
            else
                coeff_reload_start <= '0';
            end if;
        end if;
    end process;

    coeff_write_control : process (master_reset_n, clk)
        variable write_cnt      : integer := 0;
        variable coeff_addr     : integer := MAXCOEFS;
    begin
        coeff_addr := to_integer(signed(num_coeffs));
        if master_reset_n = '0' then
            coeff_in_we(0) <= '0';
            coeff_in_address <= std_logic_vector(to_signed(-
1, COEFADDWIDTH));
            coeff_in_data <=
std_logic_vector(to_signed(-1, COEFDATAWIDTH));
            write_cnt := 0;
        elsif rising_edge(clk) then

            if coeff_reload_start = '0' and coeff_in_we(0) = '0' then
                coeff_in_we(0) <= '0';
                coeff_in_address <= std_logic_vector(to_signed(-
1, COEFADDWIDTH));
                coeff_in_data <=
std_logic_vector(to_signed(-1, COEFDATAWIDTH));
                write_cnt := 0;
            elsif ( coeff_reload_start = '1' or coeff_in_we(0) = '1' )
and ( write_cnt < coeff_addr ) then
                coeff_in_we(0) <= '1';
                coeff_in_address <=
std_logic_vector(unsigned(coeff_in_address) + 1);
                coeff_in_data <= coef(write_cnt);
                write_cnt := write_cnt + 1;
            elsif coeff_in_we(0) = '1' and (write_cnt = coeff_addr )
then
                coeff_in_we(0) <= '0';
                coeff_in_address <= std_logic_vector(to_signed(-
1, COEFADDWIDTH));
                coeff_in_data <=
std_logic_vector(to_signed(-1, COEFDATAWIDTH));
            else
                coeff_in_we(0) <= '0';

```

```

        coeff_in_address <= std_logic_vector(to_signed(-
1,COEFADDWIDTH));
        coeff_in_data <=
std_logic_vector(to_signed(-1,COEFDATAWIDTH));
        write_cnt := 0;
    end if;
end if;
end process coeff_write_control;
-----
-----
---Create 125MHz clock and select clock-----
-----
    create_slow_clock : process(master_reset_n, clk) is
        variable cnt: integer;
        variable clock: std_logic;
begin
    if master_reset_n = '0' then
        cnt := 0;
        clock := '0';
    elsif rising_edge(clk) then
        cnt := cnt+1;
        if cnt = 2 then
            cnt := 0;
            clock := not clock;
        end if;
    end if;
end if;
    clk_125MHz <= clock;
end process;
    clk <= clk_125MHz when clk_sel = '1' else clk_250MHz;
    --clk <= clk_250MHz;
-----
-----
---Create latch trigger enable signal-----
-----
    done_edge : process (master_reset_n, clk)
begin
    if master_reset_n = '0' then
        reg0 <= '0';
        reg1 <= '0';
    elsif rising_edge(clk) then
        reg0 <= done;
        reg1 <= reg0;
        if reg1 = '0' and reg0 = '1' and done = '1' then
            done_re <= '1';
        else
            done_re <= '0';
        end if;
    end if;
end process;
-----
-----
---Load index and values to registers on latch trigger-----
-----
    load_latches : process(master_reset_n, clk) is
begin
    if master_reset_n = '0' then

```

```

        sta_index_lat <= (others => '0');
        sta_value_lat <= (others => '0');
        ret_index_lat <= (others => '0');
        ret_value_lat <= (others => '0');
        pixel_num_lat <= (others => '0');
    elsif rising_edge(clk) then
        if done_re = '1' then
            sta_index_lat <= sta_index;
            sta_value_lat <= sta_value;
            ret_index_lat <= ret_index;
            ret_value_lat <= ret_value;
            pixel_num_lat <= pixel_num;
        end if;
    end if;
end process;

-----
-----
---Create laser trigger enable signal-----
-----
laser_trigger : process (clk, master_reset_n)
begin
    if master_reset_n = '0' then
        prev_prev_ext_trig    <= '0';
        prev_ext_trig         <= '0';
        prev_prev_int_trig    <= '0';
        prev_int_trig         <= '0';
        laser_clk_ena         <= '0';
    elsif rising_edge(clk) then
        prev_ext_trig         <= ext_trig;
        prev_prev_ext_trig    <= prev_ext_trig;
        prev_int_trig         <= int_trig;
        prev_prev_int_trig    <= prev_int_trig;

        if trig_ext_int = '0' then
            if prev_prev_ext_trig = '0' and prev_ext_trig = '1'
and ext_trig = '1' then
                laser_clk_ena    <= '1';
            else
                laser_clk_ena    <= '0';
            end if;
        else
            if prev_prev_int_trig = '0' and prev_int_trig = '1'
and int_trig = '1' then
                laser_clk_ena    <= '1';
            else
                laser_clk_ena    <= '0';
            end if;
        end if;
    end if;
end process;

-----
-----
---Count number of pixels-----
-----

```

```

load_reg : process(master_reset_n, clk) is
    variable cnt: integer;
begin
    if master_reset_n = '0' then
        cnt := 0;
    elsif rising_edge(clk) then
        if laser_clk_ena = '1' then
            cnt := cnt+1;
        end if;
    end if;
    pixel_num <= std_logic_vector(to_signed(cnt,OUTWIDTH));
end process;

-----
-----
---Read/Write Registers-----
-----

read_write : component registers
    port map (
        clk            => clk,
        reset_n        => master_reset_n,
        address        => address,
        read_ena        => read_ena,
        read_data      => read_data,
        write_ena       => write_ena,
        write_data     => write_data,

        sta_window_min => sta_window_min,
        sta_window_max => sta_window_max,
        ret_window_min => ret_window_min,
        ret_window_max => ret_window_max,
        sta_index      => sta_index_lat,
        sta_value       => sta_value_lat,
        ret_index       => ret_index_lat,
        ret_value       => ret_value_lat,
        pixel_num       => pixel_num_lat,
        clk_sel         => clk_sel,
        trig_ext_int    => trig_ext_int,
--      debug1         => debug1,
--      debug2         => debug2,
--      debug3         => debug3,
        num_coeffs      => num_coeffs,
        reload_coef     => reload_coef,
        int_trig        => int_trig,
        coef            => coef
    );
--      debug1(INWIDTH-1 downto 0) <= input0;
--      debug1(REGWIDTH-1) <= done;
--      debug1(REGWIDTH-2 downto INWIDTH) <= (others => '0');
--      debug2(OUTWIDTH-1 downto 0) <= ast_source_data(OUTWIDTH-1 downto 0);
--      debug2(REGWIDTH-1 downto OUTWIDTH) <= (others => '0');
--      debug3(OUTWIDTH-1 downto 0) <= current_index_out;
--      debug3(REGWIDTH-1 downto OUTWIDTH) <= (others => '0');

-----
-----

```

```

---Matched Filter-----
-----
    matched_filter : component MF
        port map (
            clk            => clk,
            reset_n        => '1', --reset_n
            coeff_in_clk    => clk,
            coeff_in_areset => coeff_in_areset,
            coeff_in_address => coeff_in_address,
            coeff_in_we      => coeff_in_we,
            coeff_in_data    => coeff_in_data,
            ast_sink_data    => ast_sink_data,
            ast_sink_ready   => ast_sink_ready,
            ast_sink_valid   => '1',
            ast_sink_error   => "00",
            ast_source_data  => ast_source_data,
            ast_source_ready => '1',
            ast_source_valid => ast_source_valid,
            ast_source_error => ast_source_error
        );

    ast_sink_data(0*INWIDTH+INWIDTH-1 downto INWIDTH*0) <= input0;
    ast_sink_data(1*INWIDTH+INWIDTH-1 downto INWIDTH*1) <= input1;
    ast_sink_data(2*INWIDTH+INWIDTH-1 downto INWIDTH*2) <= input2;
    ast_sink_data(3*INWIDTH+INWIDTH-1 downto INWIDTH*3) <= input3;

-----
-----
---Peak Detector-----
-----
    find_peaks : component peak_detector
        port map (
            clk            => clk,
            reset_n        => reset_n,
            data           => ast_source_data,
            sta_window_min => sta_window_min,
            sta_window_max => sta_window_max,
            ret_window_min => ret_window_min,
            ret_window_max => ret_window_max,
            sta_index      => sta_index,
            sta_value      => sta_value,
            ret_index      => ret_index,
            ret_value      => ret_value,
            done           => done,
            current_index_out => current_index_out
        );

-----
-----
end architecture rtl;

```

B.2 Peak detector entity

```

-----
-- Title      :      Peak detector
-- Description :      Use stages or tiers of compares to find max of
--              MF outputs for each clock cycle. Also compare
--              current max with previous max and (if within
--              start/return windows). Create done signal at end
--              of return window to latch max value and
--              corresponding index.
-----

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use work.dsp_pkg.all;

entity peak_detector is
    -- if CHAN_OUT changes, Tiers must be added
    port (
        clk          : in  std_logic  := '0';
        reset_n       : in  std_logic  := '0';
        data          : in  std_logic_vector(CHANS_OUT*OUTWIDTH-1 downto
0);

        sta_window_min : in  std_logic_vector(OUTWIDTH-1 downto 0);
        sta_window_max : in  std_logic_vector(OUTWIDTH-1 downto 0);
        ret_window_min : in  std_logic_vector(OUTWIDTH-1 downto 0);
        ret_window_max : in  std_logic_vector(OUTWIDTH-1 downto 0);
        sta_index       : out std_logic_vector(OUTWIDTH-1 downto 0);
        sta_value       : out std_logic_vector(OUTWIDTH-1 downto 0);
        ret_index       : out std_logic_vector(OUTWIDTH-1 downto 0);
        ret_value       : out std_logic_vector(OUTWIDTH-1 downto 0);
        done            : out std_logic;
        current_index_out : out std_logic_vector(OUTWIDTH-1 downto
0);

    );
end entity peak_detector;

architecture rtl of peak_detector is
    signal current_index       : integer;
    signal sta_done            : std_logic;
    signal ret_done            : std_logic;
    signal smax                : std_logic_vector(OUTWIDTH-1 downto 0);
    signal smaxi               : std_logic_vector(OUTWIDTH-1 downto 0);
    signal rmax                : std_logic_vector(OUTWIDTH-1 downto 0);
    signal rmaxi               : std_logic_vector(OUTWIDTH-1 downto 0);

    -- Create 8 stage (tier) pipeline of two-input compares
    type slv_array is array (natural range <>) of
std_logic_vector(OUTWIDTH-1 downto 0);
    -- --Tier 9--512 MF Outputs
    -- signal tier9          : slv_array(0 to 511);
    -- signal tier9i         : slv_array(0 to 511);
    -- --Tier 8--256 MF Outputs
    -- signal tier8          : slv_array(0 to 255);
    -- signal tier8i         : slv_array(0 to 255);
    -- --Tier 7--128 MF Outputs

```



```

-- signal tier7      : slv_array(0 to 127);
-- signal tier7i     : slv_array(0 to 127);
--Tier 6--64 MF Outputs
signal tier6        : slv_array(0 to 63);
signal tier6i       : slv_array(0 to 63);
--Tier 5--32 MF Outputs
signal tier5        : slv_array(0 to 31);
signal tier5i       : slv_array(0 to 31);
--Tier 4--16 MF Outputs
signal tier4        : slv_array(0 to 15);
signal tier4i       : slv_array(0 to 15);
--Tier 3--8 MF Outputs
signal tier3        : slv_array(0 to 7);
signal tier3i       : slv_array(0 to 7);
--Tier 2--4 MF Outputs
signal tier2        : slv_array(0 to 3);
signal tier2i       : slv_array(0 to 3);
--Tier 1--2 MF Outputs
signal tier1        : slv_array(0 to 1);
signal tier1i       : slv_array(0 to 1);
--Tier 0
signal tier0        : std_logic_vector(OUTWIDTH-1 downto 0);
signal tier0i       : std_logic_vector(OUTWIDTH-1 downto 0);

component cmp_max IS
    port (
        clk          : in  std_logic  := '0';
        reset_n      : in  std_logic  := '0';
        valueA        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        indexA        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        valueB        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        indexB        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        value_out     : out std_logic_vector(OUTWIDTH-1 downto 0);
        index_out     : out std_logic_vector(OUTWIDTH-1 downto 0)
    );
end component cmp_max;

component final_cmp IS
    port (
        clk          : in  std_logic  := '0';
        reset_n      : in  std_logic  := '0';
        window_min    : in  std_logic_vector(OUTWIDTH-1 downto 0);
        window_max    : in  std_logic_vector(OUTWIDTH-1 downto 0);
        current_index : in integer;
        valueA        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        indexA        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        valueB        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        indexB        : in    std_logic_vector(OUTWIDTH-1 downto
0);
        value_out     : out std_logic_vector(OUTWIDTH-1 downto 0);

```

```

        index_out    : out std_logic_vector(OUTWIDTH-1 downto 0);
        done_out      : out std_logic
    );
end component final_cmp;

-----
begin
-- Setup signals
    sta_index <= smaxi;
    sta_value <= smax;
    ret_index <= rmaxi;
    ret_value <= rmax;
    done <= sta_done and ret_done;
    current_index_out <=
std_logic_vector(to_signed(current_index,OUTWIDTH));

-----
-----

-- Create index for each MF output
    counter : process(reset_n, clk) is
        variable index: integer;
    begin
        if reset_n = '0' then
            index := 0;
        elsif rising_edge(clk) then
            index := index+CHANS_OUT;
        end if;
        current_index <= index;
    end process;

-----
-----

-- Map MF output/index to compare tree inputs
    map_inputs: for i in tier2'range generate
        tier2(i) <= data(i*OUTWIDTH+OUTWIDTH-1 downto OUTWIDTH*i);
    end generate;
    create_input_index: for i in tier2i'range generate
        tier2i(i) <=
std_logic_vector(to_signed(current_index+i,OUTWIDTH));
    end generate;

-----
-
---Instantiate Compares-----
-

-----Tier 5-----
-- tier5_cmp: for i in tier5'range generate
--     cmp5: component find_max
--     port map (
--         clk           => clk,
--         reset_n       => reset_n,
--         valueA        => tier6(i*2),
--         indexA        => tier6i(i*2),

```

```

--          valueB          => tier6(i*2+1),
--          indexB          => tier6i(i*2+1),
--          value_out       => tier5(i),
--          index_out       => tier5i(i)
--      );
--  end generate tier5_cmp;

-----Tier 4-----
--  tier4_cmp: for i in tier4'range generate
--      cmp4: component cmp_max
--          port map (
--              clk          => clk,
--              reset_n      => reset_n,
--              valueA        => tier5(i*2),
--              indexA        => tier5i(i*2),
--              valueB        => tier5(i*2+1),
--              indexB        => tier5i(i*2+1),
--              value_out     => tier4(i),
--              index_out     => tier4i(i)
--          );
--  end generate tier4_cmp;

-----Tier 3-----
--  tier3_cmp: for i in tier3'range generate
--      cmp3: component cmp_max
--          port map (
--              clk          => clk,
--              reset_n      => reset_n,
--              valueA        => tier4(i*2),
--              indexA        => tier4i(i*2),
--              valueB        => tier4(i*2+1),
--              indexB        => tier4i(i*2+1),
--              value_out     => tier3(i),
--              index_out     => tier3i(i)
--          );
--  end generate tier3_cmp;

-----Tier 2-----
--  tier2_cmp: for i in tier2'range generate
--      cmp2: component cmp_max
--          port map (
--              clk          => clk,
--              reset_n      => reset_n,
--              valueA        => tier3(i*2),
--              indexA        => tier3i(i*2),
--              valueB        => tier3(i*2+1),
--              indexB        => tier3i(i*2+1),
--              value_out     => tier2(i),
--              index_out     => tier2i(i)
--          );
--  end generate tier2_cmp;

-----Tier 1-----
--  tier1_cmp: for i in tier1'range generate
--      cmp1: component cmp_max
--          port map (
--              clk          => clk,

```

```

        reset_n      => reset_n,
        valueA        => tier2(i*2),
        indexA        => tier2i(i*2),
        valueB        => tier2(i*2+1),
        indexB        => tier2i(i*2+1),
        value_out     => tier1(i),
        index_out     => tier1i(i)
    );
end generate tier1_cmp;

--Tier 0-----
tier0_cmp : component cmp_max
    port map (
        clk            => clk,
        reset_n        => reset_n,
        valueA          => tier1(0),
        indexA          => tier1i(0),
        valueB          => tier1(1),
        indexB          => tier1i(1),
        value_out       => tier0,
        index_out       => tier0i
    );

--Final for Start Pulse-----

start_cmp : component final_cmp
    port map (
        clk            => clk,
        reset_n        => reset_n,
        window_min     => sta_window_min,
        window_max     => sta_window_max,
        current_index   => current_index,
        valueA          => tier0,
        indexA          => tier0i,
        valueB          => smax,
        indexB          => smaxi,
        value_out       => smax,
        index_out       => smaxi,
        done_out        => sta_done
    );

--Final for Return Pulse-----

return_cmp : component final_cmp
    port map (
        clk            => clk,
        reset_n        => reset_n,
        window_min     => ret_window_min,
        window_max     => ret_window_max,
        current_index   => current_index,
        valueA          => tier0,
        indexA          => tier0i,
        valueB          => rmax,
        indexB          => rmaxi,
        value_out       => rmax,
        index_out       => rmaxi,
        done_out        => ret_done
    );

```

```
-----
end architecture rtl;
```

B.3 Compare output entity

```
-----
-- Title           : Find maximum of two inputs
-- Description      : Compares two input values and find max. Outputs
--                  : max input value and corresponding index on
--                  : rising edge of clock.
-----

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use work.dsp_pkg.all;

entity cmp_max is
    port (
        clk          : in  std_logic := '0';
        reset_n       : in  std_logic := '0';
        valueA         : in    std_logic_vector(OUTWIDTH-1 downto 0);
        indexA         : in    std_logic_vector(OUTWIDTH-1 downto 0);
        valueB         : in    std_logic_vector(OUTWIDTH-1 downto 0);
        indexB         : in    std_logic_vector(OUTWIDTH-1 downto 0);
        value_out      : out std_logic_vector(OUTWIDTH-1 downto 0);
        index_out      : out std_logic_vector(OUTWIDTH-1 downto 0)
    );
end entity cmp_max;

architecture rtl of cmp_max is
    signal AgB       : std_logic;
    signal valueA_signed : signed(OUTWIDTH-1 downto 0);
    signal valueB_signed : signed(OUTWIDTH-1 downto 0);
-----
begin
-- Convert inputs to signed inputs
    valueA_signed <= signed(valueA);
    valueB_signed <= signed(valueB);

-- Compare inputs
    AgB <= '1' WHEN valueA_signed > valueB_signed else '0';

-- Load max value to output on rising edge of clock (if within window)
    select_output : process(clk, reset_n) is
        variable max_value : std_logic_vector(OUTWIDTH-1 downto 0);
        variable index     : std_logic_vector(OUTWIDTH-1 downto 0);
    begin
        if reset_n = '0' then
            max_value := (others => '0');
            index     := (others => '0');
        elsif rising_edge(clk) then
```

```

        if AgB = '1' then
            max_value := valueA;
            index      := indexA;
        else
            max_value := valueB;
            index      := indexB;
        end if;

    end if;
    value_out <= max_value;
    index_out <= index;
end process;

end architecture rtl;

```

APPENDIX C

LADAR TEST SETUP PICTURES

Figure C.1: Altera Stratix IV FPGA used on current LADAR system prototype and used to test proposed hardware designs. Each hardware design was instantiated as a component on the existing system.

Figure C.2: Stratix IV FPGA and ADC used in LADAR system test to capture LADAR voltage waveform

Figure C.3: Berkeley Nucleonics Corp. 6040 Universal Pulse Generator used in LADAR system test to generate laser pulses

Figure C.4: Proprietary optical receiver used in LADAR system test to convert laser pulse to voltage

APPENDIX D

LADAR SYSTEM TEST RESULTS

D.1 Interpolation factor = 1, 247.5 MHz sampling rate, 500 μ s delay

16tap_250MHz_L01_0016tap_10ns_return.txt

start ind: 218.707 +- 1.160 samples

start amp: 203851.022 +- 8.3% counts

return ind: 342.232 +- 1.123 samples

return dly : 499.090 +- 2.018 ns

return net: 74.814 +- 0.302 m

return amp: 793684.022 +- 5.6% counts

rms_dly : 2.020 ns

rms_rng : 0.303 m

percent valid: 90.9547738693

4tap_250MHz_L01_0004tap_06ns_gaus10.txt

start ind: 218.087 +- 1.208 samples

start amp: 172115.353 +- 11.5% counts

return ind: 341.587 +- 1.090 samples

return dly : 498.990 +- 2.020 ns

return net: 74.799 +- 0.303 m

return amp: 628574.435 +- 10.6% counts

rms_dly : 2.029 ns

rms_rng : 0.304 m

percent valid: 92.9292929293

4tap_250MHz_L01_0004tap_06ns_gaus2.txt

start ind: 217.950 +- 1.296 samples

start amp: 173985.508 +- 11.1% counts

return ind: 341.464 +- 1.202 samples

return dly : 499.046 +- 2.019 ns

return net: 74.807 +- 0.303 m

return amp: 632432.441 +- 9.8% counts

rms_dly : 2.024 ns

rms_rng : 0.303 m

percent valid: 89.9497487437

4tap_250MHz_L01_0004tap_06ns_para0.txt

start ind: 217.893 +- 1.247 samples

start amp: 174259.612 +- 11.0% counts

return ind: 341.438 +- 1.136 samples

return dly : 499.171 +- 2.012 ns

return net: 74.826 +- 0.302 m

return amp: 627433.360 +- 10.0% counts

rms_dly : 2.012 ns

rms_rng : 0.302 m

percent valid: 89.4472361809

4tap_250MHz_L01_0004tap_10ns_gaus10.txt
start ind: 218.821 +- 1.096 samples
start amp: 216109.957 +- 6.6% counts
return ind: 342.310 +- 1.164 samples
return dly : 498.946 +- 2.020 ns
return net: 74.792 +- 0.303 m
return amp: 819225.636 +- 4.3% counts
rms_dly : 2.033 ns
rms_rng : 0.305 m
percent valid: 92.4623115578

4tap_250MHz_L01_0004tap_10ns_gaus2.txt
start ind: 218.733 +- 1.216 samples
start amp: 192249.182 +- 9.3% counts
return ind: 342.219 +- 1.193 samples
return dly : 498.936 +- 2.019 ns
return net: 74.790 +- 0.303 m
return amp: 708725.727 +- 7.0% counts
rms_dly : 2.034 ns
rms_rng : 0.305 m
percent valid: 93.9698492462

4tap_250MHz_L01_0004tap_10ns_para0.txt
start ind: 217.984 +- 1.155 samples
start amp: 172807.846 +- 12.2% counts
return ind: 341.549 +- 1.151 samples
return dly : 499.256 +- 2.003 ns
return net: 74.839 +- 0.300 m
return amp: 629860.846 +- 10.3% counts
rms_dly : 2.004 ns
rms_rng : 0.300 m
percent valid: 91.4572864322

4tap_250MHz_L01_0004tap_16ns_gaus10.txt
start ind: 219.754 +- 1.194 samples
start amp: 242734.877 +- 5.1% counts
return ind: 343.201 +- 1.121 samples
return dly : 498.775 +- 2.009 ns
return net: 74.766 +- 0.301 m
return amp: 956289.972 +- 3.8% counts
rms_dly : 2.049 ns
rms_rng : 0.307 m
percent valid: 89.9497487437

4tap_250MHz_L01_0004tap_16ns_gaus2.txt
start ind: 219.921 +- 1.286 samples
start amp: 184548.390 +- 10.3% counts
return ind: 343.407 +- 1.096 samples
return dly : 498.933 +- 2.019 ns
return net: 74.790 +- 0.303 m

return amp: 697956.927 +- 7.8% counts
rms_dly : 2.034 ns
rms_rng : 0.305 m
percent valid: 88.9447236181

4tap_250MHz_L01_0004tap_16ns_para0.txt
start ind: 218.783 +- 1.253 samples
start amp: 245535.478 +- 5.4% counts
return ind: 342.228 +- 1.173 samples
return dly : 498.765 +- 2.008 ns
return net: 74.765 +- 0.301 m
return amp: 974453.583 +- 3.6% counts
rms_dly : 2.050 ns
rms_rng : 0.307 m
percent valid: 90.4522613065
**Error, invalid pixel
**Error, invalid pixel

4tap_250MHz_L01_0004tap_16ns_poisson23.txt
start ind: 219.913 +- 1.202 samples
start amp: 240090.842 +- 5.0% counts
return ind: 343.443 +- 1.134 samples
return dly : 499.111 +- 2.017 ns
return net: 74.817 +- 0.302 m
return amp: 940298.317 +- 3.9% counts
rms_dly : 2.018 ns
rms_rng : 0.302 m
percent valid: 92.8934010152

4tap_250MHz_L01_0004tap_16ns_poisson42.txt
start ind: 218.973 +- 1.208 samples
start amp: 257483.545 +- 4.3% counts
return ind: 342.412 +- 1.239 samples
return dly : 498.741 +- 2.005 ns
return net: 74.761 +- 0.301 m
return amp: 1013405.733 +- 3.1% counts
rms_dly : 2.052 ns
rms_rng : 0.308 m
percent valid: 93.9698492462

4tap_250MHz_L01_0004tap_16ns_square0.txt
start ind: 219.078 +- 1.235 samples
start amp: 274172.911 +- 4.1% counts
return ind: 342.620 +- 1.153 samples
return dly : 499.159 +- 2.013 ns
return net: 74.824 +- 0.302 m
return amp: 1103111.972 +- 2.4% counts
rms_dly : 2.013 ns
rms_rng : 0.302 m
percent valid: 89.9497487437

```

4tap_250MHz_L01_0004tap_6ns_square0.txt
start ind: 218.508 +- 1.140 samples
start amp: 253906.232 +- 4.2% counts
return ind: 341.903 +- 1.287 samples
return dly : 498.564 +- 1.975 ns
return net: 74.735 +- 0.296 m
return amp: 985028.459 +- 4.0% counts
rms_dly : 2.069 ns
rms_rng : 0.310 m
percent valid: 92.9648241206

```

```

impulse-250Mhz-L01.txt
start ind: 218.017 +- 1.149 samples
start amp: 169637.884 +- 12.0% counts
return ind: 341.503 +- 1.096 samples
return dly : 498.934 +- 2.019 ns
return net: 74.790 +- 0.303 m
return amp: 640969.039 +- 9.0% counts
rms_dly : 2.034 ns
rms_rng : 0.305 m
percent valid: 90.9547738693

```

D.2 Interpolation factor = 1, 247.5 MHz sampling rate, 1000 μ s delay

```

16tap_250MHz_L01_0016tap_10ns_return.txt
start ind: 212.402 +- 1.120 samples
start amp: 264927.894 +- 7.4% counts
return ind: 459.739 +- 1.187 samples
return dly : 999.340 +- 1.909 ns
return net: 149.801 +- 0.286 m
return amp: 1026421.131 +- 7.0% counts
rms_dly : 1.924 ns
rms_rng : 0.288 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

```

```

4tap_250MHz_L01_0004tap_06ns_gaus10.txt
start ind: 211.513 +- 1.173 samples
start amp: 219586.201 +- 11.7% counts
return ind: 458.970 +- 1.287 samples
return dly : 999.827 +- 2.013 ns
return net: 149.874 +- 0.302 m
return amp: 838861.709 +- 10.9% counts
rms_dly : 2.028 ns
rms_rng : 0.304 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

```

```

4tap_250MHz_L01_0004tap_06ns_gaus2.txt
start ind: 211.367 +- 1.130 samples
start amp: 219362.799 +- 11.8% counts

```

return ind: 458.724 +- 1.198 samples
return dly : 999.421 +- 1.936 ns
return net: 149.813 +- 0.290 m
return amp: 839606.382 +- 11.7% counts
rms_dly : 1.942 ns
rms_rng : 0.291 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_06ns_para0.txt
start ind: 211.392 +- 1.142 samples
start amp: 217747.628 +- 12.4% counts
return ind: 458.769 +- 1.287 samples
return dly : 999.503 +- 1.958 ns
return net: 149.825 +- 0.294 m
return amp: 845645.940 +- 11.1% counts
rms_dly : 1.960 ns
rms_rng : 0.294 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_10ns_gaus10.txt
start ind: 212.030 +- 1.194 samples
start amp: 272569.995 +- 6.5% counts
return ind: 459.392 +- 1.120 samples
return dly : 999.442 +- 1.942 ns
return net: 149.816 +- 0.291 m
return amp: 1078358.608 +- 4.8% counts
rms_dly : 1.946 ns
rms_rng : 0.292 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_10ns_gaus2.txt
start ind: 212.397 +- 1.133 samples
start amp: 244011.683 +- 8.5% counts
return ind: 459.749 +- 1.142 samples
return dly : 999.401 +- 1.929 ns
return net: 149.810 +- 0.289 m
return amp: 939492.638 +- 8.0% counts
rms_dly : 1.938 ns
rms_rng : 0.290 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_10ns_para0.txt
start ind: 211.467 +- 1.159 samples
start amp: 219170.211 +- 12.0% counts
return ind: 458.869 +- 1.257 samples
return dly : 999.604 +- 1.981 ns
return net: 149.841 +- 0.297 m
return amp: 832786.201 +- 11.6% counts

rms_dly : 1.981 ns
rms_rng : 0.297 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_16ns_gaus10.txt
start ind: 213.266 +- 1.086 samples
start amp: 309277.312 +- 4.9% counts
return ind: 460.658 +- 1.179 samples
return dly : 999.563 +- 1.972 ns
return net: 149.835 +- 0.296 m
return amp: 1236446.673 +- 4.6% counts
rms_dly : 1.973 ns
rms_rng : 0.296 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_16ns_gaus2.txt
start ind: 213.422 +- 1.131 samples
start amp: 236436.045 +- 10.2% counts
return ind: 460.809 +- 1.192 samples
return dly : 999.543 +- 1.968 ns
return net: 149.832 +- 0.295 m
return amp: 906523.749 +- 10.4% counts
rms_dly : 1.968 ns
rms_rng : 0.295 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_16ns_para0.txt
start ind: 212.462 +- 1.159 samples
start amp: 314780.005 +- 4.5% counts
return ind: 459.799 +- 1.207 samples
return dly : 999.340 +- 1.909 ns
return net: 149.801 +- 0.286 m
return amp: 1256279.693 +- 4.3% counts
rms_dly : 1.924 ns
rms_rng : 0.288 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_16ns_poisson23.txt
start ind: 213.437 +- 1.132 samples
start amp: 303427.603 +- 4.5% counts
return ind: 460.714 +- 1.229 samples
return dly : 999.096 +- 1.807 ns
return net: 149.765 +- 0.271 m
return amp: 1223899.332 +- 4.8% counts
rms_dly : 1.870 ns
rms_rng : 0.280 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_16ns_poisson42.txt

start ind: 212.387 +- 1.193 samples
start amp: 326491.382 +- 4.2% counts
return ind: 459.739 +- 1.187 samples
return dly : 999.401 +- 1.929 ns
return net: 149.810 +- 0.289 m
return amp: 1306868.849 +- 4.0% counts
rms_dly : 1.938 ns
rms_rng : 0.290 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_16ns_square0.txt

start ind: 212.412 +- 1.139 samples
start amp: 347687.995 +- 3.4% counts
return ind: 459.824 +- 1.145 samples
return dly : 999.645 +- 1.989 ns
return net: 149.847 +- 0.298 m
return amp: 1413682.784 +- 3.2% counts
rms_dly : 1.990 ns
rms_rng : 0.298 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L01_0004tap_6ns_square0.txt

start ind: 211.990 +- 1.186 samples
start amp: 322559.121 +- 4.8% counts
return ind: 459.286 +- 1.118 samples
return dly : 999.178 +- 1.845 ns
return net: 149.777 +- 0.277 m
return amp: 1274693.352 +- 4.1% counts
rms_dly : 1.889 ns
rms_rng : 0.283 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

impulse-250Mhz-L01.txt

start ind: 211.427 +- 1.153 samples
start amp: 218638.668 +- 11.8% counts
return ind: 458.754 +- 1.254 samples
return dly : 999.300 +- 1.895 ns
return net: 149.795 +- 0.284 m
return amp: 846382.910 +- 11.3% counts
rms_dly : 1.916 ns
rms_rng : 0.287 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

D.3 Interpolation factor = 1, 495 MHz sampling rate, 499 μ s delay

16tap_500MHz_L01_0016tap_10ns_return.txt
start ind: 396.774 +- 1.192 samples
start amp: 421279.387 +- 3.9% counts
return ind: 643.717 +- 1.164 samples
return dly : 498.876 +- 0.467 ns
return net: 74.781 +- 0.070 m
return amp: 1613610.519 +- 2.4% counts
rms_dly : 0.557 ns
rms_rng : 0.084 m
percent valid: 53.2663316583

8tap_500MHz_L01_0008tap_06ns_gaus10.txt
start ind: 395.528 +- 1.198 samples
start amp: 341485.213 +- 4.2% counts
return ind: 642.509 +- 1.159 samples
return dly : 498.952 +- 0.387 ns
return net: 74.793 +- 0.058 m
return amp: 1263121.028 +- 2.9% counts
rms_dly : 0.449 ns
rms_rng : 0.067 m
percent valid: 54.2713567839

8tap_500MHz_L01_0008tap_06ns_gaus2.txt
start ind: 395.589 +- 1.110 samples
start amp: 341739.607 +- 4.0% counts
return ind: 642.598 +- 1.093 samples
return dly : 499.009 +- 0.338 ns
return net: 74.801 +- 0.051 m
return amp: 1268153.953 +- 3.0% counts
rms_dly : 0.379 ns
rms_rng : 0.057 m
percent valid: 53.7688442211

8tap_500MHz_L01_0008tap_06ns_para0.txt
start ind: 394.383 +- 1.099 samples
start amp: 342159.869 +- 4.5% counts
return ind: 641.393 +- 1.109 samples
return dly : 499.009 +- 0.338 ns
return net: 74.801 +- 0.051 m
return amp: 1263545.402 +- 3.0% counts
rms_dly : 0.379 ns
rms_rng : 0.057 m
percent valid: 53.7688442211
**Error, invalid pixel
**Error, invalid pixel

8tap_500MHz_L01_0008tap_10ns_gaus10.txt
start ind: 396.410 +- 1.093 samples
start amp: 447723.457 +- 3.6% counts

return ind: 643.410 +- 1.084 samples
return dly : 498.990 +- 0.279 ns
return net: 74.799 +- 0.042 m
return amp: 1713987.467 +- 2.4% counts
rms_dly : 0.337 ns
rms_rng : 0.051 m
percent valid: 53.2994923858
**Error, invalid pixel

8tap_500MHz_L01_0008tap_10ns_gaus2.txt
start ind: 396.491 +- 1.092 samples
start amp: 397603.236 +- 4.3% counts
return ind: 643.462 +- 1.083 samples
return dly : 498.933 +- 0.435 ns
return net: 74.790 +- 0.065 m
return amp: 1496624.792 +- 3.0% counts
rms_dly : 0.500 ns
rms_rng : 0.075 m
percent valid: 53.5353535354

8tap_500MHz_L01_0008tap_10ns_para0.txt
start ind: 395.346 +- 1.224 samples
start amp: 449106.523 +- 4.1% counts
return ind: 642.308 +- 1.234 samples
return dly : 498.914 +- 0.383 ns
return net: 74.787 +- 0.057 m
return amp: 1727437.907 +- 2.4% counts
rms_dly : 0.466 ns
rms_rng : 0.070 m
percent valid: 53.7688442211
**Error, invalid pixel

8tap_500MHz_L01_0008tap_16ns_gaus10.txt
start ind: 398.102 +- 1.312 samples
start amp: 492514.241 +- 3.6% counts
return ind: 645.000 +- 1.255 samples
return dly : 498.784 +- 0.611 ns
return net: 74.768 +- 0.092 m
return amp: 1942161.472 +- 2.2% counts
rms_dly : 0.728 ns
rms_rng : 0.109 m
percent valid: 54.5454545455

8tap_500MHz_L01_0008tap_16ns_gaus2.txt
start ind: 397.863 +- 1.172 samples
start amp: 357241.363 +- 4.0% counts
return ind: 644.863 +- 1.189 samples
return dly : 498.990 +- 0.283 ns
return net: 74.799 +- 0.042 m
return amp: 1346336.951 +- 2.4% counts
rms_dly : 0.341 ns
rms_rng : 0.051 m

percent valid: 51.256281407

8tap_500MHz_L01_0008tap_16ns_para0.txt
start ind: 396.760 +- 1.189 samples
start amp: 497591.279 +- 4.0% counts
return ind: 643.673 +- 1.188 samples
return dly : 498.815 +- 0.568 ns
return net: 74.772 +- 0.085 m
return amp: 1967607.942 +- 2.1% counts
rms_dly : 0.675 ns
rms_rng : 0.101 m
percent valid: 52.2613065327

8tap_500MHz_L01_0008tap_16ns_poisson23.txt
start ind: 397.095 +- 1.183 samples
start amp: 485645.333 +- 3.5% counts
return ind: 644.010 +- 1.191 samples
return dly : 498.817 +- 0.566 ns
return net: 74.773 +- 0.085 m
return amp: 1941785.790 +- 2.2% counts
rms_dly : 0.672 ns
rms_rng : 0.101 m
percent valid: 52.7638190955

8tap_500MHz_L01_0008tap_16ns_poisson42.txt
start ind: 396.615 +- 1.095 samples
start amp: 514040.865 +- 3.8% counts
return ind: 643.548 +- 1.126 samples
return dly : 498.854 +- 0.506 ns
return net: 74.778 +- 0.076 m
return amp: 2058264.279 +- 2.2% counts
rms_dly : 0.602 ns
rms_rng : 0.090 m
percent valid: 52.2613065327

8tap_500MHz_L01_0008tap_16ns_square0.txt
start ind: 396.346 +- 1.175 samples
start amp: 537370.548 +- 3.8% counts
return ind: 643.413 +- 1.106 samples
return dly : 499.126 +- 0.898 ns
return net: 74.819 +- 0.135 m
return amp: 2200454.442 +- 1.8% counts
rms_dly : 0.899 ns
rms_rng : 0.135 m
percent valid: 52.2613065327

8tap_500MHz_L01_0008tap_6ns_square0.txt
start ind: 395.462 +- 1.126 samples
start amp: 507619.538 +- 4.0% counts
return ind: 642.365 +- 1.110 samples
return dly : 498.796 +- 0.596 ns

return net: 74.769 +- 0.089 m
return amp: 1961434.192 +- 2.7% counts
rms_dly : 0.709 ns
rms_rng : 0.106 m
percent valid: 52.2613065327

impulse-500MHz-L01.txt
start ind: 393.952 +- 1.163 samples
start amp: 188092.221 +- 5.0% counts
return ind: 641.029 +- 1.156 samples
return dly : 499.145 +- 0.538 ns
return net: 74.822 +- 0.081 m
return amp: 685196.952 +- 3.4% counts
rms_dly : 0.539 ns
rms_rng : 0.081 m
percent valid: 52.2613065327

D.4 Interpolation factor = 1, 495 MHz sampling rate, 999 μ s delay

16tap_500MHz_L01_0016tap_10ns_return.txt
start ind: 383.709 +- 1.137 samples
start amp: 535398.990 +- 3.3% counts
return ind: 878.427 +- 1.091 samples
return dly : 999.432 +- 0.908 ns
return net: 149.815 +- 0.136 m
return amp: 2109147.819 +- 2.9% counts
rms_dly : 0.921 ns
rms_rng : 0.138 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_06ns_gaus10.txt
start ind: 382.497 +- 1.120 samples
start amp: 440795.276 +- 4.3% counts
return ind: 877.276 +- 1.194 samples
return dly : 999.553 +- 0.838 ns
return net: 149.833 +- 0.126 m
return amp: 1682630.558 +- 3.8% counts
rms_dly : 0.839 ns
rms_rng : 0.126 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_06ns_gaus2.txt
start ind: 382.422 +- 1.104 samples
start amp: 437883.347 +- 3.9% counts
return ind: 877.271 +- 1.150 samples
return dly : 999.695 +- 0.723 ns
return net: 149.854 +- 0.108 m
return amp: 1682366.070 +- 3.7% counts
rms_dly : 0.732 ns
rms_rng : 0.110 m

system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_06ns_para0.txt
start ind: 381.518 +- 1.093 samples
start amp: 436663.623 +- 4.2% counts
return ind: 876.296 +- 1.151 samples
return dly : 999.553 +- 0.838 ns
return net: 149.833 +- 0.126 m
return amp: 1674919.337 +- 3.4% counts
rms_dly : 0.839 ns
rms_rng : 0.126 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_10ns_gaus10.txt
start ind: 383.392 +- 1.097 samples
start amp: 573430.915 +- 3.5% counts
return ind: 878.090 +- 1.200 samples
return dly : 999.391 +- 0.927 ns
return net: 149.809 +- 0.139 m
return amp: 2249740.910 +- 3.1% counts
rms_dly : 0.946 ns
rms_rng : 0.142 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_10ns_gaus2.txt
start ind: 383.437 +- 1.127 samples
start amp: 507173.161 +- 3.7% counts
return ind: 878.151 +- 1.181 samples
return dly : 999.421 +- 0.913 ns
return net: 149.813 +- 0.137 m
return amp: 1962832.211 +- 3.6% counts
rms_dly : 0.927 ns
rms_rng : 0.139 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_10ns_para0.txt
start ind: 382.583 +- 1.085 samples
start amp: 577755.899 +- 3.8% counts
return ind: 877.256 +- 1.160 samples
return dly : 999.340 +- 0.947 ns
return net: 149.801 +- 0.142 m
return amp: 2253020.010 +- 3.4% counts
rms_dly : 0.977 ns
rms_rng : 0.147 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_16ns_gaus10.txt
start ind: 384.925 +- 1.199 samples
start amp: 624617.789 +- 3.0% counts
return ind: 879.618 +- 1.136 samples
return dly : 999.381 +- 0.931 ns
return net: 149.807 +- 0.140 m
return amp: 2517705.633 +- 3.2% counts
rms_dly : 0.952 ns
rms_rng : 0.143 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_16ns_gaus2.txt
start ind: 384.955 +- 1.131 samples
start amp: 454249.327 +- 3.4% counts
return ind: 879.693 +- 1.170 samples
return dly : 999.472 +- 0.888 ns
return net: 149.821 +- 0.133 m
return amp: 1771623.271 +- 3.1% counts
rms_dly : 0.894 ns
rms_rng : 0.134 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_16ns_para0.txt
start ind: 383.834 +- 1.271 samples
start amp: 634384.613 +- 3.2% counts
return ind: 878.563 +- 1.197 samples
return dly : 999.452 +- 0.898 ns
return net: 149.818 +- 0.135 m
return amp: 2554189.749 +- 2.9% counts
rms_dly : 0.907 ns
rms_rng : 0.136 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_16ns_poisson23.txt
start ind: 384.126 +- 1.248 samples
start amp: 621015.236 +- 3.3% counts
return ind: 878.874 +- 1.244 samples
return dly : 999.492 +- 0.876 ns
return net: 149.824 +- 0.131 m
return amp: 2507142.668 +- 2.8% counts
rms_dly : 0.881 ns
rms_rng : 0.132 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_16ns_poisson42.txt
start ind: 383.668 +- 1.186 samples
start amp: 651676.709 +- 3.0% counts
return ind: 878.382 +- 1.222 samples

return dly : 999.421 +- 0.913 ns
return net: 149.813 +- 0.137 m
return amp: 2642264.940 +- 3.0% counts
rms_dly : 0.927 ns
rms_rng : 0.139 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_16ns_square0.txt
start ind: 383.372 +- 1.257 samples
start amp: 690335.322 +- 3.4% counts
return ind: 878.176 +- 1.246 samples
return dly : 999.604 +- 0.802 ns
return net: 149.841 +- 0.120 m
return amp: 2827232.040 +- 2.8% counts
rms_dly : 0.802 ns
rms_rng : 0.120 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L01_0008tap_6ns_square0.txt
start ind: 382.553 +- 1.110 samples
start amp: 645993.874 +- 3.6% counts
return ind: 877.216 +- 1.239 samples
return dly : 999.320 +- 0.955 ns
return net: 149.798 +- 0.143 m
return amp: 2542594.769 +- 2.7% counts
rms_dly : 0.990 ns
rms_rng : 0.148 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

impulse-500MHz-L01.txt
start ind: 381.010 +- 1.211 samples
start amp: 240103.236 +- 4.1% counts
return ind: 875.879 +- 1.234 samples
return dly : 999.736 +- 0.681 ns
return net: 149.860 +- 0.102 m
return amp: 915075.176 +- 4.2% counts
rms_dly : 0.699 ns
rms_rng : 0.105 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

D.5 Interpolation factor = 1, 990 MHz sampling rate, 499 μ s delay

16tap_1000MHz_L01_0016tap_06ns_gaus10.txt
start ind: 749.370 +- 1.262 samples
start amp: 642741.990 +- 3.7% counts
return ind: 1243.350 +- 1.236 samples
return dly : 498.970 +- 0.349 ns
return net: 74.796 +- 0.052 m

return amp: 2392635.830 +- 2.5% counts
rms_dly : 0.408 ns
rms_rng : 0.061 m
percent valid: 50.2512562814
**Error, invalid pixel

16tap_1000MHz_L01_0016tap_06ns_gaus2.txt
start ind: 748.990 +- 1.196 samples
start amp: 584846.900 +- 3.7% counts
return ind: 1242.980 +- 1.200 samples
return dly : 498.980 +- 0.101 ns
return net: 74.797 +- 0.015 m
return amp: 2177143.400 +- 2.3% counts
rms_dly : 0.224 ns
rms_rng : 0.034 m
percent valid: 50.5050505051

16tap_1000MHz_L01_0016tap_06ns_para0.txt
start ind: 747.870 +- 1.293 samples
start amp: 644526.640 +- 3.5% counts
return ind: 1241.840 +- 1.286 samples
return dly : 498.960 +- 0.172 ns
return net: 74.794 +- 0.026 m
return amp: 2397516.830 +- 2.3% counts
rms_dly : 0.280 ns
rms_rng : 0.042 m
percent valid: 50.2512562814

16tap_1000MHz_L01_0016tap_10ns_gaus10.txt
start ind: 750.778 +- 1.097 samples
start amp: 866580.485 +- 3.7% counts
return ind: 1244.657 +- 1.093 samples
return dly : 498.867 +- 0.330 ns
return net: 74.780 +- 0.049 m
return amp: 3330380.323 +- 2.0% counts
rms_dly : 0.454 ns
rms_rng : 0.068 m
percent valid: 49.7487437186
**Error, invalid pixel

16tap_1000MHz_L01_0016tap_10ns_gaus2.txt
start ind: 751.270 +- 1.182 samples
start amp: 704252.910 +- 3.6% counts
return ind: 1245.180 +- 1.211 samples
return dly : 498.899 +- 0.322 ns
return net: 74.785 +- 0.048 m
return amp: 2660471.570 +- 2.2% counts
rms_dly : 0.428 ns
rms_rng : 0.064 m
percent valid: 50.5050505051

16tap_1000MHz_L01_0016tap_10ns_para0.txt
start ind: 749.860 +- 1.200 samples
start amp: 876393.120 +- 4.1% counts
return ind: 1243.730 +- 1.156 samples
return dly : 498.859 +- 0.340 ns
return net: 74.779 +- 0.051 m
return amp: 3361640.700 +- 2.3% counts
rms_dly : 0.468 ns
rms_rng : 0.070 m
percent valid: 50.2512562814

16tap_1000MHz_L01_0016tap_10ns_return.txt
start ind: 752.520 +- 1.081 samples
start amp: 823027.760 +- 3.9% counts
return ind: 1246.410 +- 1.078 samples
return dly : 498.879 +- 0.316 ns
return net: 74.782 +- 0.047 m
return amp: 3171069.340 +- 2.5% counts
rms_dly : 0.437 ns
rms_rng : 0.065 m
percent valid: 50.2512562814
**Error, invalid pixel

16tap_1000MHz_L01_0016tap_16ns_gaus10.txt
start ind: 753.929 +- 1.225 samples
start amp: 988151.263 +- 3.4% counts
return ind: 1247.758 +- 1.190 samples
return dly : 498.816 +- 0.407 ns
return net: 74.773 +- 0.061 m
return amp: 3902442.970 +- 2.1% counts
rms_dly : 0.546 ns
rms_rng : 0.082 m
percent valid: 50.0

16tap_1000MHz_L01_0016tap_16ns_gaus2.txt
start ind: 753.820 +- 1.152 samples
start amp: 721321.780 +- 3.7% counts
return ind: 1247.760 +- 1.069 samples
return dly : 498.929 +- 0.345 ns
return net: 74.790 +- 0.052 m
return amp: 2721148.270 +- 2.1% counts
rms_dly : 0.426 ns
rms_rng : 0.064 m
percent valid: 50.2512562814

16tap_1000MHz_L01_0016tap_16ns_para0.txt
start ind: 752.760 +- 1.218 samples
start amp: 991009.710 +- 3.2% counts
return ind: 1246.510 +- 1.204 samples
return dly : 498.737 +- 0.437 ns
return net: 74.761 +- 0.066 m
return amp: 3955764.790 +- 2.4% counts

rms_dly : 0.622 ns
rms_rng : 0.093 m
percent valid: 50.2512562814
**Error, invalid pixel

16tap_1000MHz_L01_0016tap_16ns_poisson23.txt
start ind: 753.010 +- 1.267 samples
start amp: 984022.929 +- 3.9% counts
return ind: 1246.818 +- 1.274 samples
return dly : 498.796 +- 0.398 ns
return net: 74.770 +- 0.060 m
return amp: 3882607.889 +- 2.3% counts
rms_dly : 0.553 ns
rms_rng : 0.083 m
percent valid: 50.0

16tap_1000MHz_L01_0016tap_16ns_poisson42.txt
start ind: 752.090 +- 1.266 samples
start amp: 1020679.370 +- 3.7% counts
return ind: 1245.900 +- 1.204 samples
return dly : 498.798 +- 0.396 ns
return net: 74.770 +- 0.059 m
return amp: 4095597.850 +- 2.2% counts
rms_dly : 0.550 ns
rms_rng : 0.083 m
percent valid: 50.2512562814

16tap_1000MHz_L01_0016tap_16ns_square0.txt
start ind: 751.460 +- 1.228 samples
start amp: 1082338.880 +- 3.7% counts
return ind: 1245.530 +- 1.127 samples
return dly : 499.061 +- 0.610 ns
return net: 74.809 +- 0.091 m
return amp: 4387083.190 +- 2.2% counts
rms_dly : 0.622 ns
rms_rng : 0.093 m
percent valid: 50.2512562814
**Error, invalid pixel

16tap_1000MHz_L01_0016tap_6ns_square0.txt
start ind: 749.040 +- 1.205 samples
start amp: 967168.152 +- 3.6% counts
return ind: 1242.798 +- 1.180 samples
return dly : 498.745 +- 0.433 ns
return net: 74.762 +- 0.065 m
return amp: 3695975.253 +- 2.5% counts
rms_dly : 0.614 ns
rms_rng : 0.092 m
percent valid: 50.0

impulse- 1000MHz-L01.txt

start ind: 745.909 +- 1.173 samples
start amp: 192858.626 +- 3.5% counts
return ind: 1240.010 +- 1.176 samples
return dly : 499.092 +- 0.337 ns
return net: 74.814 +- 0.050 m
return amp: 697287.889 +- 2.7% counts
rms_dly : 0.348 ns
rms_rng : 0.052 m
percent valid: 49.7487437186

D.6 Interpolation factor = 1, 990 MHz sampling rate, 999 μ s delay

16tap_1000MHz_L01_0016tap_06ns_gaus10.txt

start ind: 722.935 +- 1.248 samples
start amp: 824997.261 +- 3.8% counts
return ind: 1712.427 +- 1.145 samples
return dly : 999.487 +- 0.505 ns
return net: 149.823 +- 0.076 m
return amp: 3163178.714 +- 2.9% counts
rms_dly : 0.513 ns
rms_rng : 0.077 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_06ns_gaus2.txt

start ind: 722.975 +- 1.266 samples
start amp: 757238.688 +- 3.4% counts
return ind: 1712.573 +- 1.209 samples
return dly : 999.594 +- 0.495 ns
return net: 149.839 +- 0.074 m
return amp: 2889108.950 +- 3.2% counts
rms_dly : 0.495 ns
rms_rng : 0.074 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_06ns_para0.txt

start ind: 722.040 +- 1.207 samples
start amp: 826844.065 +- 3.1% counts
return ind: 1711.583 +- 1.157 samples
return dly : 999.538 +- 0.503 ns
return net: 149.831 +- 0.075 m
return amp: 3183225.879 +- 3.3% counts
rms_dly : 0.505 ns
rms_rng : 0.076 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_10ns_gaus10.txt

start ind: 724.899 +- 1.199 samples
start amp: 1111041.789 +- 3.3% counts
return ind: 1714.337 +- 1.126 samples

return dly : 999.432 +- 0.501 ns
return net: 149.815 +- 0.075 m
return amp: 4349384.638 +- 2.7% counts
rms_dly : 0.523 ns
rms_rng : 0.078 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_10ns_gaus2.txt
start ind: 724.899 +- 1.211 samples
start amp: 901733.608 +- 3.3% counts
return ind: 1714.472 +- 1.168 samples
return dly : 999.569 +- 0.500 ns
return net: 149.835 +- 0.075 m
return amp: 3493077.884 +- 3.1% counts
rms_dly : 0.500 ns
rms_rng : 0.075 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_10ns_para0.txt
start ind: 723.950 +- 1.302 samples
start amp: 1122898.915 +- 3.3% counts
return ind: 1713.392 +- 1.151 samples
return dly : 999.437 +- 0.502 ns
return net: 149.816 +- 0.075 m
return amp: 4384326.925 +- 2.8% counts
rms_dly : 0.522 ns
rms_rng : 0.078 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_10ns_return.txt
start ind: 726.568 +- 1.127 samples
start amp: 1056125.683 +- 3.3% counts
return ind: 1716.080 +- 1.183 samples
return dly : 999.508 +- 0.505 ns
return net: 149.826 +- 0.076 m
return amp: 4140550.121 +- 2.6% counts
rms_dly : 0.510 ns
rms_rng : 0.076 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_16ns_gaus10.txt
start ind: 727.839 +- 1.196 samples
start amp: 1258858.553 +- 3.0% counts
return ind: 1717.231 +- 1.218 samples
return dly : 999.386 +- 0.493 ns
return net: 149.808 +- 0.074 m
return amp: 5012151.839 +- 3.0% counts
rms_dly : 0.530 ns

rms_rng : 0.079 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_16ns_gaus2.txt
start ind: 727.879 +- 1.278 samples
start amp: 919044.457 +- 3.2% counts
return ind: 1717.402 +- 1.194 samples
return dly : 999.518 +- 0.505 ns
return net: 149.828 +- 0.076 m
return amp: 3584961.261 +- 3.1% counts
rms_dly : 0.508 ns
rms_rng : 0.076 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_16ns_para0.txt
start ind: 726.824 +- 1.118 samples
start amp: 1274274.156 +- 3.1% counts
return ind: 1716.271 +- 1.142 samples
return dly : 999.442 +- 0.502 ns
return net: 149.816 +- 0.075 m
return amp: 5101371.337 +- 2.8% counts
rms_dly : 0.521 ns
rms_rng : 0.078 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_16ns_poisson23.txt
start ind: 727.196 +- 1.222 samples
start amp: 1249613.754 +- 3.2% counts
return ind: 1716.563 +- 1.158 samples
return dly : 999.360 +- 0.487 ns
return net: 149.804 +- 0.073 m
return amp: 5043151.523 +- 2.8% counts
rms_dly : 0.534 ns
rms_rng : 0.080 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_16ns_poisson42.txt
start ind: 726.000 +- 1.125 samples
start amp: 1301061.528 +- 3.0% counts
return ind: 1715.432 +- 1.109 samples
return dly : 999.426 +- 0.500 ns
return net: 149.814 +- 0.075 m
return amp: 5270236.432 +- 2.5% counts
rms_dly : 0.523 ns
rms_rng : 0.078 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_16ns_square0.txt

start ind: 725.286 +- 1.265 samples
start amp: 1380524.276 +- 3.3% counts
return ind: 1714.945 +- 1.216 samples
return dly : 999.655 +- 0.500 ns
return net: 149.848 +- 0.075 m
return amp: 5645343.116 +- 2.5% counts
rms_dly : 0.506 ns
rms_rng : 0.076 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L01_0016tap_6ns_square0.txt

start ind: 723.095 +- 1.262 samples
start amp: 1233913.497 +- 3.2% counts
return ind: 1712.533 +- 1.124 samples
return dly : 999.432 +- 0.501 ns
return net: 149.815 +- 0.075 m
return amp: 4798467.181 +- 2.7% counts
rms_dly : 0.523 ns
rms_rng : 0.078 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

impulse- 1000MHz-L01.txt

start ind: 720.171 +- 1.203 samples
start amp: 244388.960 +- 3.6% counts
return ind: 1709.824 +- 1.167 samples
return dly : 999.650 +- 0.481 ns
return net: 149.847 +- 0.072 m
return amp: 940106.472 +- 3.3% counts
rms_dly : 0.486 ns
rms_rng : 0.073 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

D.7 Interpolation factor = 4, sampling rate = 247.5 MHz, delay = 499 μ s

16tap_250MHz_L04_0064tap_10ns_return.txt

start ind: 878.863 +- 4.829 samples
start amp: 209769.907 +- 5.3% counts
return ind: 1372.665 +- 4.715 samples
return dly : 498.790 +- 0.876 ns
return net: 74.769 +- 0.131 m
return amp: 801367.016 +- 4.2% counts
rms_dly : 0.959 ns
rms_rng : 0.144 m
percent valid: 91.4572864322

4tap_250MHz_L04_0016tap_06ns_gaus10.txt

start ind: 875.160 +- 4.555 samples
start amp: 176354.968 +- 10.3% counts

```

return ind: 1369.267 +- 4.089 samples
return dly : 499.098 +- 1.707 ns
return net: 74.815 +- 0.256 m
return amp: 629995.070 +- 9.0% counts
rms_dly : 1.709 ns
rms_rng : 0.256 m
percent valid: 93.9698492462
*****

4tap_250MHz_L04_0016tap_06ns_gaus2.txt
start ind: 875.000 +- 4.908 samples
start amp: 171363.575 +- 12.0% counts
return ind: 1368.978 +- 4.453 samples
return dly : 498.968 +- 2.020 ns
return net: 74.795 +- 0.303 m
return amp: 636980.731 +- 9.5% counts
rms_dly : 2.031 ns
rms_rng : 0.304 m
percent valid: 93.4673366834
*****

4tap_250MHz_L04_0016tap_06ns_para0.txt
start ind: 874.800 +- 5.145 samples
start amp: 171738.232 +- 10.5% counts
return ind: 1368.811 +- 4.688 samples
return dly : 499.001 +- 1.738 ns
return net: 74.800 +- 0.261 m
return amp: 645442.157 +- 8.3% counts
rms_dly : 1.748 ns
rms_rng : 0.262 m
percent valid: 92.9648241206
*****

4tap_250MHz_L04_0016tap_10ns_gaus10.txt
start ind: 877.017 +- 4.966 samples
start amp: 218898.144 +- 3.9% counts
return ind: 1370.900 +- 5.018 samples
return dly : 498.872 +- 0.514 ns
return net: 74.781 +- 0.077 m
return amp: 836597.406 +- 2.5% counts
rms_dly : 0.599 ns
rms_rng : 0.090 m
percent valid: 90.4522613065
*****

4tap_250MHz_L04_0016tap_10ns_gaus2.txt
start ind: 877.516 +- 4.905 samples
start amp: 181797.527 +- 9.7% counts
return ind: 1371.394 +- 4.522 samples
return dly : 498.866 +- 1.532 ns
return net: 74.780 +- 0.230 m
return amp: 685994.399 +- 7.6% counts
rms_dly : 1.563 ns
rms_rng : 0.234 m
percent valid: 94.472361809
*****

4tap_250MHz_L04_0016tap_10ns_para0.txt
start ind: 876.417 +- 4.591 samples
start amp: 221174.754 +- 3.5% counts
return ind: 1370.353 +- 4.656 samples

```

return dly : 498.925 +- 0.474 ns
return net: 74.789 +- 0.071 m
return amp: 846969.353 +- 2.5% counts
rms_dly : 0.538 ns
rms_rng : 0.081 m
percent valid: 93.9698492462**Error, invalid pixel

4tap_250MHz_L04_0016tap_16ns_gaus10.txt

start ind: 880.044 +- 4.564 samples
start amp: 250039.377 +- 3.8% counts
return ind: 1373.858 +- 4.481 samples
return dly : 498.802 +- 0.433 ns
return net: 74.770 +- 0.065 m
return amp: 975484.306 +- 2.6% counts
rms_dly : 0.575 ns
rms_rng : 0.086 m
percent valid: 92.4242424242

4tap_250MHz_L04_0016tap_16ns_gaus2.txt

start ind: 879.253 +- 4.734 samples
start amp: 184549.230 +- 9.8% counts
return ind: 1373.388 +- 4.292 samples
return dly : 499.126 +- 1.519 ns
return net: 74.819 +- 0.228 m
return amp: 708763.719 +- 6.9% counts
rms_dly : 1.520 ns
rms_rng : 0.228 m
percent valid: 89.4472361809

4tap_250MHz_L04_0016tap_16ns_para0.txt

start ind: 878.642 +- 4.360 samples
start amp: 251483.229 +- 3.6% counts
return ind: 1372.397 +- 4.382 samples
return dly : 498.742 +- 0.496 ns
return net: 74.761 +- 0.074 m
return amp: 990868.542 +- 2.4% counts
rms_dly : 0.662 ns
rms_rng : 0.099 m
percent valid: 89.9497487437

4tap_250MHz_L04_0016tap_16ns_poisson23.txt

start ind: 878.796 +- 4.663 samples
start amp: 248353.133 +- 4.0% counts
return ind: 1372.729 +- 4.587 samples
return dly : 498.923 +- 0.606 ns
return net: 74.789 +- 0.091 m
return amp: 976291.398 +- 2.5% counts
rms_dly : 0.659 ns
rms_rng : 0.099 m
percent valid: 90.9547738693

4tap_250MHz_L04_0016tap_16ns_poisson42.txt

start ind: 878.678 +- 4.305 samples
start amp: 259628.027 +- 3.8% counts
return ind: 1372.426 +- 4.244 samples

```

return dly : 498.736 +- 0.627 ns
return net: 74.761 +- 0.094 m
return amp: 1027587.749 +- 2.5% counts
rms_dly : 0.768 ns
rms_rng : 0.115 m
percent valid: 91.959798995
*****

4tap_250MHz_L04_0016tap_16ns_square0.txt
start ind: 878.429 +- 4.653 samples
start amp: 273559.077 +- 4.2% counts
return ind: 1372.626 +- 4.316 samples
return dly : 499.190 +- 2.010 ns
return net: 74.829 +- 0.301 m
return amp: 1100924.231 +- 2.5% counts
rms_dly : 2.010 ns
rms_rng : 0.301 m
percent valid: 91.4572864322
*****

4tap_250MHz_L04_0016tap_6ns_square0.txt
start ind: 875.978 +- 4.432 samples
start amp: 256056.422 +- 4.6% counts
return ind: 1369.933 +- 4.798 samples
return dly : 498.945 +- 2.020 ns
return net: 74.792 +- 0.303 m
return amp: 980609.000 +- 3.8% counts
rms_dly : 2.033 ns
rms_rng : 0.305 m
percent valid: 90.4522613065
*****

impulse-250Mhz-L04.txt
start ind: 872.364 +- 4.907 samples
start amp: 172299.909 +- 11.5% counts
return ind: 1366.353 +- 4.563 samples
return dly : 498.979 +- 2.020 ns
return net: 74.797 +- 0.303 m
return amp: 632934.984 +- 9.9% counts
rms_dly : 2.030 ns
rms_rng : 0.304 m
percent valid: 93.9698492462

```

D.8 Interpolation factor = 4, sampling rate = 247.5 MHz, delay = 999 μ s

```

*****

16tap_250MHz_L04_0064tap_10ns_return.txt
start ind: 852.161 +- 4.615 samples
start amp: 267583.357 +- 5.3% counts
return ind: 1841.749 +- 4.740 samples
return dly : 999.584 +- 0.811 ns
return net: 149.838 +- 0.122 m
return amp: 1040340.457 +- 5.5% counts
rms_dly : 0.811 ns
rms_rng : 0.122 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

*****

```


4tap_250MHz_L04_0016tap_06ns_gaus10.txt
start ind: 848.859 +- 4.358 samples
start amp: 220866.899 +- 11.4% counts
return ind: 1838.186 +- 4.715 samples
return dly : 999.320 +- 1.545 ns
return net: 149.798 +- 0.232 m
return amp: 843265.085 +- 10.7% counts
rms_dly : 1.567 ns
rms_rng : 0.235 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L04_0016tap_06ns_gaus2.txt
start ind: 849.533 +- 4.661 samples
start amp: 218643.804 +- 12.7% counts
return ind: 1839.000 +- 4.711 samples
return dly : 999.462 +- 1.947 ns
return net: 149.819 +- 0.292 m
return amp: 842135.704 +- 10.9% counts
rms_dly : 1.951 ns
rms_rng : 0.292 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L04_0016tap_06ns_para0.txt
start ind: 848.271 +- 4.535 samples
start amp: 220062.739 +- 10.7% counts
return ind: 1837.628 +- 4.812 samples
return dly : 999.350 +- 1.512 ns
return net: 149.803 +- 0.227 m
return amp: 845795.884 +- 10.5% counts
rms_dly : 1.529 ns
rms_rng : 0.229 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

4tap_250MHz_L04_0016tap_10ns_gaus10.txt
start ind: 851.296 +- 4.429 samples
start amp: 280115.302 +- 3.2% counts
return ind: 1840.683 +- 4.415 samples
return dly : 999.381 +- 0.492 ns
return net: 149.807 +- 0.074 m
return amp: 1094533.794 +- 3.2% counts
rms_dly : 0.531 ns
rms_rng : 0.080 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_10ns_gaus2.txt
start ind: 851.583 +- 4.508 samples
start amp: 232588.251 +- 9.0% counts
return ind: 1841.025 +- 4.670 samples

return dly : 999.437 +- 1.409 ns
return net: 149.816 +- 0.211 m
return amp: 900329.372 +- 9.2% counts
rms_dly : 1.416 ns
rms_rng : 0.212 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_10ns_para0.txt
start ind: 850.126 +- 4.647 samples
start amp: 281612.467 +- 3.2% counts
return ind: 1839.568 +- 4.554 samples
return dly : 999.437 +- 0.502 ns
return net: 149.816 +- 0.075 m
return amp: 1101245.492 +- 3.2% counts
rms_dly : 0.522 ns
rms_rng : 0.078 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_16ns_gaus10.txt
start ind: 854.111 +- 4.697 samples
start amp: 315192.975 +- 3.6% counts
return ind: 1843.508 +- 4.724 samples
return dly : 999.391 +- 0.504 ns
return net: 149.809 +- 0.076 m
return amp: 1271301.211 +- 3.5% counts
rms_dly : 0.539 ns
rms_rng : 0.081 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_16ns_gaus2.txt
start ind: 854.256 +- 4.431 samples
start amp: 238649.271 +- 8.5% counts
return ind: 1843.678 +- 4.792 samples
return dly : 999.416 +- 1.440 ns
return net: 149.812 +- 0.216 m
return amp: 918229.799 +- 8.4% counts
rms_dly : 1.449 ns
rms_rng : 0.217 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_16ns_para0.txt
start ind: 852.156 +- 4.500 samples
start amp: 320356.558 +- 3.5% counts
return ind: 1841.467 +- 4.469 samples
return dly : 999.305 +- 0.468 ns
return net: 149.796 +- 0.070 m
return amp: 1287742.915 +- 2.8% counts
rms_dly : 0.543 ns

rms_rng : 0.081 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_16ns_poisson23.txt
start ind: 853.608 +- 4.744 samples
start amp: 314246.402 +- 3.6% counts
return ind: 1842.990 +- 4.739 samples
return dly : 999.376 +- 0.595 ns
return net: 149.806 +- 0.089 m
return amp: 1258511.935 +- 3.1% counts
rms_dly : 0.629 ns
rms_rng : 0.094 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_16ns_poisson42.txt
start ind: 852.568 +- 4.723 samples
start amp: 330063.698 +- 3.3% counts
return ind: 1842.010 +- 4.845 samples
return dly : 999.437 +- 0.781 ns
return net: 149.816 +- 0.117 m
return amp: 1325232.497 +- 3.1% counts
rms_dly : 0.794 ns
rms_rng : 0.119 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L04_0016tap_16ns_square0.txt
start ind: 852.487 +- 4.248 samples
start amp: 348902.583 +- 3.6% counts
return ind: 1841.854 +- 4.750 samples
return dly : 999.360 +- 1.916 ns
return net: 149.804 +- 0.287 m
return amp: 1417038.950 +- 3.0% counts
rms_dly : 1.929 ns
rms_rng : 0.289 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L04_0016tap_6ns_square0.txt
start ind: 850.040 +- 4.960 samples
start amp: 324562.035 +- 4.5% counts
return ind: 1839.729 +- 4.769 samples
return dly : 999.685 +- 1.996 ns
return net: 149.853 +- 0.299 m
return amp: 1271480.985 +- 3.8% counts
rms_dly : 1.998 ns
rms_rng : 0.300 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

impulse-250Mhz-L04.txt
start ind: 846.231 +- 4.582 samples
start amp: 220443.859 +- 11.7% counts
return ind: 1835.839 +- 4.941 samples
return dly : 999.604 +- 1.981 ns
return net: 149.841 +- 0.297 m
return amp: 834468.136 +- 11.2% counts
rms_dly : 1.981 ns
rms_rng : 0.297 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

D.9 Interpolation factor = 4, sampling rate = 495 MHz, delay = 499 μ s

16tap_500MHz_L04_0064tap_10ns_return.txt
start ind: 1538.698 +- 4.682 samples
start amp: 513315.714 +- 3.0% counts
return ind: 2526.427 +- 4.698 samples
return dly : 498.853 +- 0.236 ns
return net: 74.778 +- 0.035 m
return amp: 2364727.633 +- 2.3% counts
rms_dly : 0.403 ns
rms_rng : 0.060 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_06ns_gaus10.txt
start ind: 1529.739 +- 4.766 samples
start amp: 415200.965 +- 3.5% counts
return ind: 2517.593 +- 4.738 samples
return dly : 498.916 +- 0.270 ns
return net: 74.788 +- 0.040 m
return amp: 1866630.251 +- 2.1% counts
rms_dly : 0.377 ns
rms_rng : 0.057 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_06ns_gaus2.txt
start ind: 1530.106 +- 4.682 samples
start amp: 375283.221 +- 3.3% counts
return ind: 2517.990 +- 4.680 samples
return dly : 498.932 +- 0.259 ns
return net: 74.790 +- 0.039 m
return amp: 1679356.342 +- 1.9% counts
rms_dly : 0.359 ns
rms_rng : 0.054 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_06ns_para0.txt
start ind: 1528.864 +- 4.767 samples
start amp: 416827.251 +- 3.5% counts
return ind: 2516.759 +- 4.771 samples

return dly : 498.937 +- 0.293 ns
return net: 74.791 +- 0.044 m
return amp: 1875186.849 +- 2.1% counts
rms_dly : 0.381 ns
rms_rng : 0.057 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_10ns_gaus10.txt
start ind: 1534.302 +- 4.619 samples
start amp: 547344.920 +- 3.2% counts
return ind: 2521.935 +- 4.624 samples
return dly : 498.805 +- 0.264 ns
return net: 74.771 +- 0.040 m
return amp: 2504504.191 +- 2.3% counts
rms_dly : 0.459 ns
rms_rng : 0.069 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_10ns_gaus2.txt
start ind: 1534.151 +- 4.652 samples
start amp: 444171.121 +- 3.5% counts
return ind: 2522.015 +- 4.642 samples
return dly : 498.921 +- 0.213 ns
return net: 74.788 +- 0.032 m
return amp: 2003136.869 +- 2.4% counts
rms_dly : 0.335 ns
rms_rng : 0.050 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_10ns_para0.txt
start ind: 1533.186 +- 4.655 samples
start amp: 549073.065 +- 3.4% counts
return ind: 2520.869 +- 4.695 samples
return dly : 498.830 +- 0.246 ns
return net: 74.775 +- 0.037 m
return amp: 2525286.739 +- 2.4% counts
rms_dly : 0.428 ns
rms_rng : 0.064 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_gaus10.txt
start ind: 1539.955 +- 4.778 samples
start amp: 617539.266 +- 3.0% counts
return ind: 2527.457 +- 4.763 samples
return dly : 498.739 +- 0.253 ns
return net: 74.761 +- 0.038 m
return amp: 2891713.724 +- 2.3% counts
rms_dly : 0.508 ns
rms_rng : 0.076 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_gaus2.txt
start ind: 1540.477 +- 4.590 samples
start amp: 452552.990 +- 3.2% counts
return ind: 2528.256 +- 4.572 samples
return dly : 498.878 +- 0.221 ns
return net: 74.782 +- 0.033 m
return amp: 2054484.005 +- 2.3% counts
rms_dly : 0.374 ns
rms_rng : 0.056 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_para0.txt
start ind: 1538.432 +- 4.483 samples
start amp: 622953.045 +- 3.0% counts
return ind: 2526.000 +- 4.445 samples
return dly : 498.772 +- 0.250 ns
return net: 74.766 +- 0.038 m
return amp: 2936327.930 +- 2.3% counts
rms_dly : 0.479 ns
rms_rng : 0.072 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_poisson23.txt
start ind: 1538.090 +- 4.693 samples
start amp: 608768.553 +- 3.5% counts
return ind: 2525.693 +- 4.652 samples
return dly : 498.789 +- 0.281 ns
return net: 74.769 +- 0.042 m
return amp: 2876392.859 +- 2.2% counts
rms_dly : 0.481 ns
rms_rng : 0.072 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_poisson42.txt
start ind: 1537.508 +- 4.891 samples
start amp: 635656.472 +- 3.1% counts
return ind: 2525.060 +- 4.845 samples
return dly : 498.764 +- 0.346 ns
return net: 74.765 +- 0.052 m
return amp: 3017164.960 +- 2.4% counts
rms_dly : 0.541 ns
rms_rng : 0.081 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_square0.txt
start ind: 1536.005 +- 4.923 samples
start amp: 672627.503 +- 2.8% counts
return ind: 2523.985 +- 4.837 samples
return dly : 498.980 +- 0.555 ns
return net: 74.797 +- 0.083 m
return amp: 3236812.663 +- 2.4% counts
rms_dly : 0.590 ns

rms_rng : 0.088 m
percent valid: 100.0

8tap_500MHz_L04_0032tap_6ns_square0.txt
start ind: 1529.930 +- 4.481 samples
start amp: 631955.497 +- 3.4% counts
return ind: 2517.528 +- 4.456 samples
return dly : 498.787 +- 0.607 ns
return net: 74.768 +- 0.091 m
return amp: 2921641.216 +- 2.7% counts
rms_dly : 0.724 ns
rms_rng : 0.108 m
percent valid: 100.0

impulse-500MHz-L04.txt
start ind: 1524.241 +- 4.905 samples
start amp: 235529.915 +- 4.3% counts
return ind: 2512.342 +- 4.908 samples
return dly : 499.041 +- 0.316 ns
return net: 74.806 +- 0.047 m
return amp: 1035532.513 +- 2.0% counts
rms_dly : 0.346 ns
rms_rng : 0.052 m
percent valid: 100.0

D.10 Interpolation factor = 4, sampling rate = 495 MHz, delay = 999 μ s

16tap_500MHz_L04_0064tap_10ns_return.txt
start ind: 1538.040 +- 4.652 samples
start amp: 525643.764 +- 2.9% counts
return ind: 3516.925 +- 4.627 samples
return dly : 999.437 +- 0.321 ns
return net: 149.816 +- 0.048 m
return amp: 2060875.698 +- 2.6% counts
rms_dly : 0.351 ns
rms_rng : 0.053 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_06ns_gaus10.txt
start ind: 1530.332 +- 4.894 samples
start amp: 424760.322 +- 3.5% counts
return ind: 3509.327 +- 4.969 samples
return dly : 999.492 +- 0.457 ns
return net: 149.824 +- 0.069 m
return amp: 1638720.879 +- 3.4% counts
rms_dly : 0.465 ns
rms_rng : 0.070 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_06ns_gaus2.txt
start ind: 1529.794 +- 4.546 samples
start amp: 383790.513 +- 3.0% counts
return ind: 3508.749 +- 4.594 samples
return dly : 999.472 +- 0.393 ns
return net: 149.821 +- 0.059 m
return amp: 1474730.075 +- 3.1% counts
rms_dly : 0.408 ns
rms_rng : 0.061 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_06ns_para0.txt
start ind: 1529.819 +- 4.751 samples
start amp: 425658.503 +- 3.3% counts
return ind: 3508.693 +- 4.906 samples
return dly : 999.432 +- 0.488 ns
return net: 149.815 +- 0.073 m
return amp: 1637321.854 +- 3.0% counts
rms_dly : 0.510 ns
rms_rng : 0.076 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_10ns_gaus10.txt
start ind: 1534.101 +- 4.660 samples
start amp: 557057.869 +- 3.3% counts
return ind: 3512.824 +- 4.635 samples
return dly : 999.355 +- 0.237 ns
return net: 149.803 +- 0.036 m
return amp: 2193325.804 +- 2.7% counts
rms_dly : 0.327 ns
rms_rng : 0.049 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_10ns_gaus2.txt
start ind: 1533.915 +- 4.793 samples
start amp: 451193.000 +- 3.0% counts
return ind: 3512.905 +- 4.803 samples
return dly : 999.490 +- 0.202 ns
return net: 149.824 +- 0.030 m
return amp: 1761147.322 +- 2.9% counts
rms_dly : 0.222 ns
rms_rng : 0.033 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_10ns_para0.txt
start ind: 1533.382 +- 4.449 samples
start amp: 564482.362 +- 3.3% counts
return ind: 3512.181 +- 4.418 samples

return dly : 999.393 +- 0.232 ns
return net: 149.809 +- 0.035 m
return amp: 2216879.231 +- 2.8% counts
rms_dly : 0.298 ns
rms_rng : 0.045 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_gaus10.txt
start ind: 1539.990 +- 5.079 samples
start amp: 628712.256 +- 2.9% counts
return ind: 3518.668 +- 4.983 samples
return dly : 999.333 +- 0.257 ns
return net: 149.800 +- 0.038 m
return amp: 2532774.844 +- 2.9% counts
rms_dly : 0.357 ns
rms_rng : 0.053 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_gaus2.txt
start ind: 1540.472 +- 4.320 samples
start amp: 462679.015 +- 3.3% counts
return ind: 3519.372 +- 4.333 samples
return dly : 999.444 +- 0.175 ns
return net: 149.817 +- 0.026 m
return amp: 1792742.372 +- 2.9% counts
rms_dly : 0.222 ns
rms_rng : 0.033 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_para0.txt
start ind: 1538.899 +- 4.551 samples
start amp: 637522.497 +- 3.2% counts
return ind: 3517.618 +- 4.505 samples
return dly : 999.353 +- 0.227 ns
return net: 149.803 +- 0.034 m
return amp: 2560419.141 +- 2.7% counts
rms_dly : 0.321 ns
rms_rng : 0.048 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_poisson23.txt
start ind: 1538.352 +- 4.609 samples
start amp: 621763.955 +- 3.0% counts
return ind: 3517.055 +- 4.545 samples
return dly : 999.345 +- 0.391 ns
return net: 149.802 +- 0.059 m
return amp: 2514441.663 +- 2.5% counts
rms_dly : 0.456 ns

rms_rng : 0.068 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_poisson42.txt
start ind: 1538.156 +- 4.827 samples
start amp: 650278.809 +- 3.1% counts
return ind: 3516.925 +- 4.659 samples
return dly : 999.378 +- 0.518 ns
return net: 149.807 +- 0.078 m
return amp: 2632225.940 +- 2.7% counts
rms_dly : 0.556 ns
rms_rng : 0.083 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L04_0032tap_16ns_square0.txt
start ind: 1536.528 +- 4.633 samples
start amp: 688797.186 +- 3.2% counts
return ind: 3515.804 +- 4.778 samples
return dly : 999.635 +- 0.778 ns
return net: 149.845 +- 0.117 m
return amp: 2833399.990 +- 2.8% counts
rms_dly : 0.780 ns
rms_rng : 0.117 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L04_0032tap_6ns_square0.txt
start ind: 1530.533 +- 4.609 samples
start amp: 642085.623 +- 3.3% counts
return ind: 3509.327 +- 4.712 samples
return dly : 999.391 +- 0.927 ns
return net: 149.809 +- 0.139 m
return amp: 2554427.372 +- 2.9% counts
rms_dly : 0.946 ns
rms_rng : 0.142 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

impulse-500MHz-L04.txt
start ind: 1524.704 +- 4.802 samples
start amp: 240236.764 +- 4.7% counts
return ind: 3503.799 +- 4.940 samples
return dly : 999.543 +- 0.845 ns
return net: 149.832 +- 0.127 m
return amp: 918498.106 +- 4.8% counts
rms_dly : 0.846 ns
rms_rng : 0.127 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

D.11 Interpolation factor = 4, sampling rate = 990 MHz, delay = 499 μ s

16tap_1000MHz_L04_0064tap_06ns_gaus10.txt
start ind: 2996.859 +- 4.508 samples
start amp: 659004.485 +- 3.8% counts
return ind: 4972.677 +- 4.563 samples
return dly : 498.944 +- 0.170 ns
return net: 74.792 +- 0.025 m
return amp: 2442693.545 +- 2.2% counts
rms_dly : 0.291 ns
rms_rng : 0.044 m
percent valid: 49.7487437186

16tap_1000MHz_L04_0064tap_06ns_gaus2.txt
start ind: 2996.323 +- 4.647 samples
start amp: 595689.556 +- 4.0% counts
return ind: 4972.121 +- 4.591 samples
return dly : 498.939 +- 0.168 ns
return net: 74.791 +- 0.025 m
return amp: 2216650.646 +- 2.3% counts
rms_dly : 0.294 ns
rms_rng : 0.044 m
percent valid: 49.7487437186

16tap_1000MHz_L04_0064tap_06ns_para0.txt
start ind: 2994.990 +- 4.598 samples
start amp: 662788.242 +- 3.7% counts
return ind: 4970.798 +- 4.612 samples
return dly : 498.941 +- 0.185 ns
return net: 74.791 +- 0.028 m
return amp: 2459346.162 +- 2.1% counts
rms_dly : 0.302 ns
rms_rng : 0.045 m
percent valid: 49.7487437186

16tap_1000MHz_L04_0064tap_10ns_gaus10.txt
start ind: 3004.750 +- 4.531 samples
start amp: 879532.410 +- 3.4% counts
return ind: 4980.080 +- 4.417 samples
return dly : 498.821 +- 0.156 ns
return net: 74.773 +- 0.023 m
return amp: 3354320.090 +- 2.1% counts
rms_dly : 0.392 ns
rms_rng : 0.059 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_10ns_gaus2.txt
start ind: 3003.790 +- 4.160 samples
start amp: 710422.890 +- 3.5% counts
return ind: 4979.490 +- 4.208 samples
return dly : 498.914 +- 0.116 ns

return net: 74.787 +- 0.017 m
return amp: 2674396.020 +- 2.5% counts
rms_dly : 0.290 ns
rms_rng : 0.043 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_10ns_para0.txt
start ind: 3003.350 +- 4.698 samples
start amp: 879452.200 +- 3.7% counts
return ind: 4978.730 +- 4.773 samples
return dly : 498.833 +- 0.142 ns
return net: 74.775 +- 0.021 m
return amp: 3389176.630 +- 2.2% counts
rms_dly : 0.375 ns
rms_rng : 0.056 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_10ns_return.txt
start ind: 3013.210 +- 4.885 samples
start amp: 821866.240 +- 3.6% counts
return ind: 4988.850 +- 4.887 samples
return dly : 498.899 +- 0.141 ns
return net: 74.785 +- 0.021 m
return amp: 3159530.630 +- 2.1% counts
rms_dly : 0.314 ns
rms_rng : 0.047 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_16ns_gaus10.txt
start ind: 3015.970 +- 4.833 samples
start amp: 987255.717 +- 3.3% counts
return ind: 4991.081 +- 4.851 samples
return dly : 498.765 +- 0.134 ns
return net: 74.765 +- 0.020 m
return amp: 3907151.020 +- 2.1% counts
rms_dly : 0.436 ns
rms_rng : 0.065 m
percent valid: 49.7487437186
**Error, invalid pixel

16tap_1000MHz_L04_0064tap_16ns_gaus2.txt
start ind: 3017.192 +- 4.199 samples
start amp: 722237.121 +- 4.0% counts
return ind: 4993.020 +- 4.283 samples
return dly : 498.947 +- 0.125 ns
return net: 74.792 +- 0.019 m
return amp: 2734253.131 +- 2.0% counts
rms_dly : 0.265 ns
rms_rng : 0.040 m
percent valid: 50.0

16tap_1000MHz_L04_0064tap_16ns_para0.txt
start ind: 3014.400 +- 4.503 samples
start amp: 992779.780 +- 3.7% counts
return ind: 4989.390 +- 4.474 samples
return dly : 498.735 +- 0.261 ns
return net: 74.760 +- 0.039 m
return amp: 3964934.150 +- 2.0% counts
rms_dly : 0.516 ns
rms_rng : 0.077 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_16ns_poisson23.txt
start ind: 3012.879 +- 5.119 samples
start amp: 968447.616 +- 3.7% counts
return ind: 4987.970 +- 5.026 samples
return dly : 498.760 +- 0.213 ns
return net: 74.764 +- 0.032 m
return amp: 3880349.667 +- 2.1% counts
rms_dly : 0.470 ns
rms_rng : 0.071 m
percent valid: 49.7487437186

16tap_1000MHz_L04_0064tap_16ns_poisson42.txt
start ind: 3012.310 +- 4.634 samples
start amp: 1020484.440 +- 3.5% counts
return ind: 4987.490 +- 4.685 samples
return dly : 498.783 +- 0.326 ns
return net: 74.768 +- 0.049 m
return amp: 4075570.860 +- 2.3% counts
rms_dly : 0.514 ns
rms_rng : 0.077 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_16ns_square0.txt
start ind: 3008.640 +- 5.094 samples
start amp: 1076099.570 +- 3.6% counts
return ind: 4985.320 +- 4.739 samples
return dly : 499.162 +- 0.475 ns
return net: 74.824 +- 0.071 m
return amp: 4372320.400 +- 2.3% counts
rms_dly : 0.475 ns
rms_rng : 0.071 m
percent valid: 50.2512562814

16tap_1000MHz_L04_0064tap_6ns_square0.txt
start ind: 2996.162 +- 4.954 samples
start amp: 964442.818 +- 3.8% counts
return ind: 4971.475 +- 5.010 samples
return dly : 498.816 +- 0.381 ns
return net: 74.773 +- 0.057 m
return amp: 3677816.687 +- 2.3% counts
rms_dly : 0.527 ns

rms_rng : 0.079 m
percent valid: 49.7487437186

impulse-1000MHz-L04.txt
start ind: 2983.434 +- 4.513 samples
start amp: 192905.081 +- 3.6% counts
return ind: 4959.717 +- 4.450 samples
return dly : 499.061 +- 0.259 ns
return net: 74.809 +- 0.039 m
return amp: 699042.838 +- 2.3% counts
rms_dly : 0.285 ns
rms_rng : 0.043 m
percent valid: 49.7487437186

D.12 Interpolation factor = 4, sampling rate = 990 MHz, delay = 999 μ s

16tap_1000MHz_L04_0064tap_06ns_gaus10.txt
start ind: 2892.482 +- 4.615 samples
start amp: 839851.442 +- 3.3% counts
return ind: 6850.342 +- 4.799 samples
return dly : 999.459 +- 0.310 ns
return net: 149.819 +- 0.046 m
return amp: 3222014.035 +- 2.9% counts
rms_dly : 0.333 ns
rms_rng : 0.050 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_06ns_gaus2.txt
start ind: 2892.558 +- 4.736 samples
start amp: 761198.563 +- 3.2% counts
return ind: 6850.583 +- 4.854 samples
return dly : 999.501 +- 0.220 ns
return net: 149.825 +- 0.033 m
return amp: 2922745.859 +- 3.0% counts
rms_dly : 0.234 ns
rms_rng : 0.035 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_06ns_para0.txt
start ind: 2891.543 +- 4.313 samples
start amp: 842550.447 +- 3.6% counts
return ind: 6849.392 +- 4.499 samples
return dly : 999.457 +- 0.296 ns
return net: 149.819 +- 0.044 m
return amp: 3247992.085 +- 2.8% counts
rms_dly : 0.321 ns
rms_rng : 0.048 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_10ns_gaus10.txt
start ind: 2900.648 +- 4.254 samples
start amp: 1116157.583 +- 3.3% counts
return ind: 6858.156 +- 4.296 samples
return dly : 999.371 +- 0.160 ns
return net: 149.806 +- 0.024 m
return amp: 4382280.638 +- 2.9% counts
rms_dly : 0.263 ns
rms_rng : 0.039 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_10ns_gaus2.txt
start ind: 2900.251 +- 4.535 samples
start amp: 904293.347 +- 3.3% counts
return ind: 6857.975 +- 4.575 samples
return dly : 999.425 +- 0.129 ns
return net: 149.814 +- 0.019 m
return amp: 3504897.176 +- 2.9% counts
rms_dly : 0.201 ns
rms_rng : 0.030 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_10ns_para0.txt
start ind: 2899.794 +- 4.532 samples
start amp: 1117489.121 +- 3.2% counts
return ind: 6857.216 +- 4.639 samples
return dly : 999.349 +- 0.146 ns
return net: 149.802 +- 0.022 m
return amp: 4410832.764 +- 2.7% counts
rms_dly : 0.273 ns
rms_rng : 0.041 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_10ns_return.txt
start ind: 2909.970 +- 4.509 samples
start amp: 1047411.834 +- 3.3% counts
return ind: 6867.528 +- 4.431 samples
return dly : 999.383 +- 0.190 ns
return net: 149.808 +- 0.029 m
return amp: 4121587.095 +- 2.8% counts
rms_dly : 0.274 ns
rms_rng : 0.041 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_16ns_gaus10.txt
start ind: 2911.090 +- 4.400 samples
start amp: 1253603.457 +- 2.8% counts

return ind: 6868.296 +- 4.383 samples
return dly : 999.294 +- 0.152 ns
return net: 149.794 +- 0.023 m
return amp: 5051640.442 +- 2.7% counts
rms_dly : 0.324 ns
rms_rng : 0.049 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_16ns_gaus2.txt
start ind: 2912.241 +- 4.659 samples
start amp: 926659.050 +- 2.9% counts
return ind: 6869.955 +- 4.660 samples
return dly : 999.423 +- 0.135 ns
return net: 149.813 +- 0.020 m
return amp: 3593452.618 +- 3.2% counts
rms_dly : 0.207 ns
rms_rng : 0.031 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_16ns_para0.txt
start ind: 2910.588 +- 4.916 samples
start amp: 1270131.005 +- 3.2% counts
return ind: 6867.804 +- 4.862 samples
return dly : 999.297 +- 0.247 ns
return net: 149.795 +- 0.037 m
return amp: 5115226.116 +- 2.8% counts
rms_dly : 0.375 ns
rms_rng : 0.056 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_16ns_poisson23.txt
start ind: 2908.688 +- 4.406 samples
start amp: 1245819.221 +- 3.1% counts
return ind: 6866.111 +- 4.469 samples
return dly : 999.349 +- 0.269 ns
return net: 149.802 +- 0.040 m
return amp: 5012482.899 +- 2.6% counts
rms_dly : 0.355 ns
rms_rng : 0.053 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_16ns_poisson42.txt
start ind: 2907.779 +- 4.931 samples
start amp: 1294921.759 +- 3.2% counts
return ind: 6865.307 +- 4.543 samples
return dly : 999.376 +- 0.423 ns
return net: 149.806 +- 0.063 m
return amp: 5230017.764 +- 2.7% counts

rms_dly : 0.470 ns
rms_rng : 0.070 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_16ns_square0.txt
start ind: 2903.905 +- 5.095 samples
start amp: 1374839.080 +- 3.1% counts
return ind: 6862.055 +- 4.641 samples
return dly : 999.533 +- 0.504 ns
return net: 149.830 +- 0.075 m
return amp: 5653498.573 +- 3.0% counts
rms_dly : 0.506 ns
rms_rng : 0.076 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L04_0064tap_6ns_square0.txt
start ind: 2893.065 +- 4.983 samples
start amp: 1230526.518 +- 2.8% counts
return ind: 6850.372 +- 4.660 samples
return dly : 999.320 +- 0.474 ns
return net: 149.798 +- 0.071 m
return amp: 4790339.970 +- 2.9% counts
rms_dly : 0.540 ns
rms_rng : 0.081 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

impulse-1000MHz-L04.txt
start ind: 2880.322 +- 5.078 samples
start amp: 244432.613 +- 3.3% counts
return ind: 6838.653 +- 4.915 samples
return dly : 999.579 +- 0.498 ns
return net: 149.837 +- 0.075 m
return amp: 941036.030 +- 3.1% counts
rms_dly : 0.498 ns
rms_rng : 0.075 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

D.13 Interpolation factor = 16, sampling rate = 247.5 MHz, delay = 499 μ s

16tap_250MHz_L16_0256tap_10ns_return.txt
start ind: 3517.550 +- 18.641 samples
start amp: 209489.861 +- 5.7% counts
return ind: 5493.628 +- 18.494 samples
return dly : 499.010 +- 0.715 ns
return net: 74.802 +- 0.107 m
return amp: 799181.478 +- 4.2% counts
rms_dly : 0.735 ns
rms_rng : 0.110 m

```

percent valid: 90.4522613065
*****

4tap_250MHz_L16_0064tap_06ns_gaus10.txt
start ind: 3499.782 +- 20.080 samples
start amp: 172758.271 +- 11.5% counts
return ind: 5476.362 +- 18.995 samples
return dly : 499.136 +- 1.591 ns
return net: 74.821 +- 0.238 m
return amp: 638912.255 +- 9.1% counts
rms_dly : 1.591 ns
rms_rng : 0.239 m
percent valid: 94.472361809
*****

4tap_250MHz_L16_0064tap_06ns_gaus2.txt
start ind: 3500.703 +- 20.115 samples
start amp: 172451.269 +- 11.9% counts
return ind: 5477.143 +- 18.699 samples
return dly : 499.101 +- 2.017 ns
return net: 74.815 +- 0.302 m
return amp: 637110.308 +- 9.7% counts
rms_dly : 2.019 ns
rms_rng : 0.303 m
percent valid: 91.4572864322
*****

4tap_250MHz_L16_0064tap_06ns_para0.txt
start ind: 3498.492 +- 20.319 samples
start amp: 171536.877 +- 10.5% counts
return ind: 5475.749 +- 19.042 samples
return dly : 499.307 +- 1.654 ns
return net: 74.846 +- 0.248 m
return amp: 647847.575 +- 7.5% counts
rms_dly : 1.659 ns
rms_rng : 0.249 m
percent valid: 89.9497487437
*****

4tap_250MHz_L16_0064tap_10ns_gaus10.txt
start ind: 3510.924 +- 18.057 samples
start amp: 220988.136 +- 3.9% counts
return ind: 5486.842 +- 18.242 samples
return dly : 498.969 +- 0.326 ns
return net: 74.795 +- 0.049 m
return amp: 842444.799 +- 2.5% counts
rms_dly : 0.388 ns
rms_rng : 0.058 m
percent valid: 92.4623115578
*****

4tap_250MHz_L16_0064tap_10ns_gaus2.txt
start ind: 3509.531 +- 19.540 samples
start amp: 182057.944 +- 10.3% counts
return ind: 5486.107 +- 18.580 samples
return dly : 499.135 +- 1.268 ns
return net: 74.820 +- 0.190 m
return amp: 693653.537 +- 7.4% counts
rms_dly : 1.269 ns
rms_rng : 0.190 m
percent valid: 88.9447236181

```

```

*****
4tap_250MHz_L16_0064tap_10ns_para0.txt
start ind: 3507.295 +- 18.855 samples
start amp: 223355.313 +- 3.9% counts
return ind: 5483.358 +- 18.919 samples
return dly : 499.006 +- 0.362 ns
return net: 74.801 +- 0.054 m
return amp: 847306.358 +- 2.2% counts
rms_dly : 0.402 ns
rms_rng : 0.060 m
percent valid: 88.4422110553
*****
4tap_250MHz_L16_0064tap_16ns_gaus10.txt
start ind: 3520.176 +- 18.632 samples
start amp: 249973.016 +- 3.9% counts
return ind: 5495.765 +- 18.482 samples
return dly : 498.886 +- 0.262 ns
return net: 74.783 +- 0.039 m
return amp: 983232.219 +- 2.5% counts
rms_dly : 0.394 ns
rms_rng : 0.059 m
percent valid: 93.9698492462
*****
4tap_250MHz_L16_0064tap_16ns_gaus2.txt
start ind: 3522.838 +- 19.118 samples
start amp: 189464.229 +- 8.8% counts
return ind: 5498.687 +- 18.594 samples
return dly : 498.952 +- 1.280 ns
return net: 74.793 +- 0.192 m
return amp: 695688.883 +- 7.6% counts
rms_dly : 1.300 ns
rms_rng : 0.195 m
percent valid: 89.9497487437
*****
4tap_250MHz_L16_0064tap_16ns_para0.txt
start ind: 3522.285 +- 18.571 samples
start amp: 253241.344 +- 3.5% counts
return ind: 5497.731 +- 18.332 samples
return dly : 498.850 +- 0.309 ns
return net: 74.778 +- 0.046 m
return amp: 996138.048 +- 2.6% counts
rms_dly : 0.452 ns
rms_rng : 0.068 m
percent valid: 93.4673366834
*****
4tap_250MHz_L16_0064tap_16ns_poisson23.txt
start ind: 3515.732 +- 18.673 samples
start amp: 249521.142 +- 3.9% counts
return ind: 5491.377 +- 18.501 samples
return dly : 498.900 +- 0.431 ns
return net: 74.785 +- 0.065 m
return amp: 974632.546 +- 2.6% counts
rms_dly : 0.514 ns
rms_rng : 0.077 m
percent valid: 91.959798995
*****

```

4tap_250MHz_L16_0064tap_16ns_poisson42.txt

start ind: 3516.783 +- 18.236 samples
start amp: 259122.850 +- 4.4% counts
return ind: 5492.667 +- 17.931 samples
return dly : 498.960 +- 0.540 ns
return net: 74.794 +- 0.081 m
return amp: 1028552.544 +- 2.5% counts
rms_dly : 0.583 ns
rms_rng : 0.087 m
percent valid: 90.4522613065

4tap_250MHz_L16_0064tap_16ns_square0.txt

start ind: 3517.184 +- 21.259 samples
start amp: 275683.119 +- 4.0% counts
return ind: 5494.178 +- 18.739 samples
return dly : 499.241 +- 2.005 ns
return net: 74.836 +- 0.300 m
return amp: 1104036.216 +- 2.6% counts
rms_dly : 2.005 ns
rms_rng : 0.301 m
percent valid: 92.9648241206

4tap_250MHz_L16_0064tap_6ns_square0.txt

start ind: 3505.905 +- 18.715 samples
start amp: 257387.185 +- 4.7% counts
return ind: 5481.862 +- 19.936 samples
return dly : 498.979 +- 2.020 ns
return net: 74.797 +- 0.303 m
return amp: 983431.667 +- 3.8% counts
rms_dly : 2.030 ns
rms_rng : 0.304 m
percent valid: 94.9748743719

impulse-250Mhz-L16.txt

start ind: 3486.108 +- 19.469 samples
start amp: 173943.301 +- 11.0% counts
return ind: 5462.710 +- 18.449 samples
return dly : 499.142 +- 2.014 ns
return net: 74.821 +- 0.302 m
return amp: 633005.371 +- 9.7% counts
rms_dly : 2.015 ns
rms_rng : 0.302 m
percent valid: 93.4673366834

D.14 Interpolation factor = 16, sampling rate = 247.5 MHz, delay = 999 μ s

16tap_250MHz_L16_0256tap_10ns_return.txt

start ind: 3411.975 +- 18.468 samples
start amp: 266870.266 +- 5.2% counts
return ind: 7370.191 +- 18.656 samples
return dly : 999.550 +- 0.758 ns
return net: 149.832 +- 0.114 m
return amp: 1040410.437 +- 5.6% counts
rms_dly : 0.759 ns
rms_rng : 0.114 m

system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

4tap_250MHz_L16_0064tap_06ns_gaus10.txt
start ind: 3397.910 +- 17.450 samples
start amp: 222873.010 +- 11.7% counts
return ind: 7356.779 +- 19.194 samples
return dly : 999.714 +- 1.565 ns
return net: 149.857 +- 0.235 m
return amp: 833670.990 +- 11.1% counts
rms_dly : 1.571 ns
rms_rng : 0.235 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

4tap_250MHz_L16_0064tap_06ns_gaus2.txt
start ind: 3397.005 +- 18.252 samples
start amp: 222264.457 +- 11.6% counts
return ind: 7355.035 +- 19.022 samples
return dly : 999.503 +- 1.958 ns
return net: 149.825 +- 0.294 m
return amp: 836067.899 +- 11.4% counts
rms_dly : 1.960 ns
rms_rng : 0.294 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L16_0064tap_06ns_para0.txt
start ind: 3394.704 +- 16.762 samples
start amp: 219954.985 +- 10.9% counts
return ind: 7353.457 +- 17.637 samples
return dly : 999.685 +- 1.607 ns
return net: 149.853 +- 0.241 m
return amp: 847762.935 +- 10.6% counts
rms_dly : 1.610 ns
rms_rng : 0.241 m
system offset 0.167 ns, 0.025 m, 0.017%
percent valid: 100.0

4tap_250MHz_L16_0064tap_10ns_gaus10.txt
start ind: 3404.734 +- 19.097 samples
start amp: 281669.161 +- 3.4% counts
return ind: 7363.005 +- 19.062 samples
return dly : 999.563 +- 0.241 ns
return net: 149.835 +- 0.036 m
return amp: 1101662.985 +- 3.3% counts
rms_dly : 0.242 ns
rms_rng : 0.036 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

4tap_250MHz_L16_0064tap_10ns_gaus2.txt
start ind: 3401.357 +- 17.007 samples
start amp: 235543.070 +- 8.7% counts
return ind: 7359.523 +- 18.085 samples
return dly : 999.537 +- 1.252 ns
return net: 149.831 +- 0.188 m
return amp: 903198.980 +- 8.9% counts
rms_dly : 1.252 ns
rms_rng : 0.188 m
system offset -0.590 ns, -0.088 m, -0.059%
percent valid: 100.0

4tap_250MHz_L16_0064tap_10ns_para0.txt
start ind: 3404.769 +- 17.997 samples
start amp: 285471.397 +- 3.3% counts
return ind: 7362.884 +- 17.884 samples
return dly : 999.524 +- 0.256 ns
return net: 149.829 +- 0.038 m
return amp: 1110859.739 +- 3.0% counts
rms_dly : 0.262 ns
rms_rng : 0.039 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

4tap_250MHz_L16_0064tap_16ns_gaus10.txt
start ind: 3415.759 +- 18.239 samples
start amp: 317535.945 +- 3.4% counts
return ind: 7373.568 +- 18.313 samples
return dly : 999.447 +- 0.246 ns
return net: 149.817 +- 0.037 m
return amp: 1270224.402 +- 3.2% counts
rms_dly : 0.280 ns
rms_rng : 0.042 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

4tap_250MHz_L16_0064tap_16ns_gaus2.txt
start ind: 3416.563 +- 18.986 samples
start amp: 239614.121 +- 8.7% counts
return ind: 7375.020 +- 20.401 samples
return dly : 999.610 +- 1.269 ns
return net: 149.842 +- 0.190 m
return amp: 930670.744 +- 8.1% counts
rms_dly : 1.269 ns
rms_rng : 0.190 m
system offset -0.843 ns, -0.126 m, -0.084%
percent valid: 100.0

4tap_250MHz_L16_0064tap_16ns_para0.txt
start ind: 3416.075 +- 19.607 samples
start amp: 321009.523 +- 3.2% counts
return ind: 7373.834 +- 19.863 samples

return dly : 999.434 +- 0.325 ns
return net: 149.815 +- 0.049 m
return amp: 1293347.467 +- 3.6% counts
rms_dly : 0.357 ns
rms_rng : 0.053 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

4tap_250MHz_L16_0064tap_16ns_poisson23.txt
start ind: 3413.116 +- 18.249 samples
start amp: 316680.171 +- 3.4% counts
return ind: 7370.744 +- 18.482 samples
return dly : 999.401 +- 0.525 ns
return net: 149.810 +- 0.079 m
return amp: 1256991.437 +- 3.4% counts
rms_dly : 0.555 ns
rms_rng : 0.083 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

4tap_250MHz_L16_0064tap_16ns_poisson42.txt
start ind: 3414.673 +- 18.890 samples
start amp: 329576.678 +- 3.4% counts
return ind: 7372.774 +- 19.425 samples
return dly : 999.520 +- 0.614 ns
return net: 149.828 +- 0.092 m
return amp: 1327624.603 +- 3.4% counts
rms_dly : 0.617 ns
rms_rng : 0.093 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

4tap_250MHz_L16_0064tap_16ns_square0.txt
start ind: 3414.317 +- 17.327 samples
start amp: 349868.090 +- 3.6% counts
return ind: 7372.829 +- 18.406 samples
return dly : 999.624 +- 1.985 ns
return net: 149.844 +- 0.298 m
return amp: 1418464.101 +- 3.2% counts
rms_dly : 1.985 ns
rms_rng : 0.298 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

4tap_250MHz_L16_0064tap_6ns_square0.txt
start ind: 3402.251 +- 19.450 samples
start amp: 325139.799 +- 4.3% counts
return ind: 7360.281 +- 19.238 samples
return dly : 999.503 +- 1.958 ns
return net: 149.825 +- 0.294 m
return amp: 1280522.347 +- 3.7% counts
rms_dly : 1.960 ns

rms_rng : 0.294 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

impulse-250Mhz-L16.txt
start ind: 3384.040 +- 17.562 samples
start amp: 221571.141 +- 11.3% counts
return ind: 7343.035 +- 19.424 samples
return dly : 999.746 +- 2.004 ns
return net: 149.862 +- 0.300 m
return amp: 840957.065 +- 11.4% counts
rms_dly : 2.011 ns
rms_rng : 0.301 m
system offset -1.600 ns, -0.240 m, -0.160%
percent valid: 100.0

D.15 Interpolation factor = 16, sampling rate = 495 MHz, delay = 499 μ s

16tap_500MHz_L16_0256tap_10ns_return.txt
start ind: 6365.408 +- 17.756 samples
start amp: 414595.893 +- 3.5% counts
return ind: 10317.165 +- 17.720 samples
return dly : 498.959 +- 0.131 ns
return net: 74.794 +- 0.020 m
return amp: 1582959.291 +- 2.3% counts
rms_dly : 0.257 ns
rms_rng : 0.038 m
percent valid: 51.7587939698

8tap_500MHz_L16_0128tap_06ns_gaus10.txt
start ind: 6330.552 +- 19.078 samples
start amp: 335684.095 +- 3.7% counts
return ind: 10282.933 +- 18.979 samples
return dly : 499.038 +- 0.237 ns
return net: 74.806 +- 0.035 m
return amp: 1245826.152 +- 2.8% counts
rms_dly : 0.276 ns
rms_rng : 0.041 m
percent valid: 52.7638190955

8tap_500MHz_L16_0128tap_06ns_gaus2.txt
start ind: 6328.198 +- 18.715 samples
start amp: 302757.151 +- 3.6% counts
return ind: 10280.764 +- 18.747 samples
return dly : 499.061 +- 0.177 ns
return net: 74.809 +- 0.027 m
return amp: 1117004.585 +- 2.1% counts
rms_dly : 0.213 ns
rms_rng : 0.032 m
percent valid: 53.2663316583

8tap_500MHz_L16_0128tap_06ns_para0.txt
start ind: 6331.542 +- 19.721 samples
start amp: 335536.477 +- 3.6% counts


```

return ind: 10284.047 +- 19.725 samples
return dly : 499.054 +- 0.277 ns
return net: 74.808 +- 0.042 m
return amp: 1250686.561 +- 2.5% counts
rms_dly : 0.305 ns
rms_rng : 0.046 m
percent valid: 53.7688442211
*****

8tap_500MHz_L16_0128tap_10ns_gaus10.txt
start ind: 6347.346 +- 18.613 samples
start amp: 442303.712 +- 3.7% counts
return ind: 10299.173 +- 18.803 samples
return dly : 498.968 +- 0.104 ns
return net: 74.795 +- 0.016 m
return amp: 1677105.519 +- 2.3% counts
rms_dly : 0.236 ns
rms_rng : 0.035 m
percent valid: 52.2613065327
*****

8tap_500MHz_L16_0128tap_10ns_gaus2.txt
start ind: 6347.379 +- 18.929 samples
start amp: 356811.233 +- 3.9% counts
return ind: 10299.825 +- 18.982 samples
return dly : 499.046 +- 0.122 ns
return net: 74.807 +- 0.018 m
return amp: 1338304.816 +- 2.4% counts
rms_dly : 0.181 ns
rms_rng : 0.027 m
percent valid: 51.7587939698
*****

8tap_500MHz_L16_0128tap_10ns_para0.txt
start ind: 6348.472 +- 18.486 samples
start amp: 441755.085 +- 3.6% counts
return ind: 10299.858 +- 18.497 samples
return dly : 498.912 +- 0.160 ns
return net: 74.787 +- 0.024 m
return amp: 1694784.689 +- 2.1% counts
rms_dly : 0.312 ns
rms_rng : 0.047 m
percent valid: 53.2663316583**Error, invalid pixel

*****

8tap_500MHz_L16_0128tap_16ns_gaus10.txt
start ind: 6371.972 +- 17.996 samples
start amp: 498413.358 +- 3.5% counts
return ind: 10323.226 +- 17.855 samples
return dly : 498.896 +- 0.143 ns
return net: 74.784 +- 0.021 m
return amp: 1956699.208 +- 2.0% counts
rms_dly : 0.318 ns
rms_rng : 0.048 m
percent valid: 53.5353535354
*****

8tap_500MHz_L16_0128tap_16ns_gaus2.txt
start ind: 6369.613 +- 17.594 samples
start amp: 366152.679 +- 3.5% counts

```

return ind: 10322.019 +- 17.640 samples
return dly : 499.041 +- 0.108 ns
return net: 74.806 +- 0.016 m
return amp: 1369573.849 +- 2.0% counts
rms_dly : 0.176 ns
rms_rng : 0.026 m
percent valid: 53.2663316583

8tap_500MHz_L16_0128tap_16ns_para0.txt
start ind: 6365.792 +- 17.281 samples
start amp: 501949.113 +- 3.7% counts
return ind: 10317.094 +- 17.345 samples
return dly : 498.902 +- 0.225 ns
return net: 74.785 +- 0.034 m
return amp: 1992532.368 +- 2.3% counts
rms_dly : 0.358 ns
rms_rng : 0.054 m
percent valid: 53.2663316583

D.16 Interpolation factor = 16, sampling rate = 495 MHz, delay = 999 μ s

16tap_500MHz_L16_0128tap_10ns_return.txt
start ind: 6156.482 +- 18.817 samples
start amp: 530037.327 +- 3.0% counts
return ind: 14072.925 +- 18.976 samples
return dly : 999.551 +- 0.279 ns
return net: 149.833 +- 0.042 m
return amp: 2066376.271 +- 2.9% counts
rms_dly : 0.281 ns
rms_rng : 0.042 m
system offset 0.041 ns, 0.006 m, 0.004%
percent valid: 100.0

16tap_500MHz_L16_0256tap_10ns_return.txt
start ind: 6157.347 +- 18.441 samples
start amp: 525140.925 +- 3.1% counts
return ind: 14073.452 +- 18.348 samples
return dly : 999.508 +- 0.266 ns
return net: 149.826 +- 0.040 m
return amp: 2066384.377 +- 2.8% counts
rms_dly : 0.276 ns
rms_rng : 0.041 m
system offset 0.041 ns, 0.006 m, 0.004%
percent valid: 100.0

8tap_500MHz_L16_0128tap_06ns_gaus10.txt
start ind: 6121.538 +- 18.263 samples
start amp: 427693.889 +- 3.4% counts
return ind: 14038.432 +- 18.150 samples
return dly : 999.608 +- 0.450 ns
return net: 149.841 +- 0.068 m
return amp: 1645098.518 +- 3.2% counts
rms_dly : 0.451 ns

rms_rng : 0.068 m
system offset 0.167 ns, 0.025 m, 0.017%
percent valid: 100.0

8tap_500MHz_L16_0128tap_06ns_gaus2.txt
start ind: 6123.799 +- 18.718 samples
start amp: 384677.106 +- 3.5% counts
return ind: 14040.824 +- 18.805 samples
return dly : 999.624 +- 0.329 ns
return net: 149.844 +- 0.049 m
return amp: 1481427.513 +- 3.0% counts
rms_dly : 0.332 ns
rms_rng : 0.050 m
system offset 0.167 ns, 0.025 m, 0.017%
percent valid: 100.0

8tap_500MHz_L16_0128tap_06ns_para0.txt
start ind: 6122.307 +- 18.873 samples
start amp: 426552.553 +- 3.2% counts
return ind: 14039.216 +- 18.787 samples
return dly : 999.610 +- 0.459 ns
return net: 149.842 +- 0.069 m
return amp: 1651777.553 +- 3.0% counts
rms_dly : 0.460 ns
rms_rng : 0.069 m
system offset 0.167 ns, 0.025 m, 0.017%
percent valid: 100.0

8tap_500MHz_L16_0128tap_10ns_gaus10.txt
start ind: 6138.573 +- 19.009 samples
start amp: 560904.447 +- 3.3% counts
return ind: 14054.643 +- 19.160 samples
return dly : 999.504 +- 0.159 ns
return net: 149.826 +- 0.024 m
return amp: 2200963.568 +- 2.7% counts
rms_dly : 0.176 ns
rms_rng : 0.026 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L16_0128tap_10ns_gaus2.txt
start ind: 6136.774 +- 18.282 samples
start amp: 452264.905 +- 3.0% counts
return ind: 14053.307 +- 18.366 samples
return dly : 999.562 +- 0.096 ns
return net: 149.834 +- 0.014 m
return amp: 1752148.111 +- 2.8% counts
rms_dly : 0.098 ns
rms_rng : 0.015 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L16_0128tap_10ns_para0.txt
start ind: 6137.131 +- 18.513 samples
start amp: 565783.543 +- 3.1% counts
return ind: 14053.236 +- 18.684 samples
return dly : 999.508 +- 0.167 ns
return net: 149.826 +- 0.025 m
return amp: 2216432.382 +- 3.0% counts
rms_dly : 0.182 ns
rms_rng : 0.027 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L16_0128tap_16ns_gaus10.txt
start ind: 6160.090 +- 18.892 samples
start amp: 630108.995 +- 3.4% counts
return ind: 14075.739 +- 18.807 samples
return dly : 999.451 +- 0.165 ns
return net: 149.818 +- 0.025 m
return amp: 2529129.015 +- 2.7% counts
rms_dly : 0.210 ns
rms_rng : 0.031 m
system offset -0.211 ns, -0.032 m, -0.021%
percent valid: 100.0

8tap_500MHz_L16_0128tap_16ns_gaus2.txt
start ind: 6161.663 +- 18.147 samples
start amp: 465411.764 +- 3.0% counts
return ind: 14078.462 +- 18.160 samples
return dly : 999.596 +- 0.101 ns
return net: 149.839 +- 0.015 m
return amp: 1808638.678 +- 2.9% counts
rms_dly : 0.102 ns
rms_rng : 0.015 m
system offset 0.041 ns, 0.006 m, 0.004%
percent valid: 100.0

8tap_500MHz_L16_0128tap_16ns_para0.txt
start ind: 6159.101 +- 18.691 samples
start amp: 638541.578 +- 3.5% counts
return ind: 14074.749 +- 18.321 samples
return dly : 999.451 +- 0.218 ns
return net: 149.818 +- 0.033 m
return amp: 2565997.513 +- 3.0% counts
rms_dly : 0.254 ns
rms_rng : 0.038 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L16_0128tap_16ns_poisson23.txt
start ind: 6153.286 +- 18.834 samples
start amp: 630466.337 +- 3.2% counts

return ind: 14069.090 +- 18.498 samples
return dly : 999.470 +- 0.350 ns
return net: 149.821 +- 0.053 m
return amp: 2510220.960 +- 2.7% counts
rms_dly : 0.367 ns
rms_rng : 0.055 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

8tap_500MHz_L16_0128tap_16ns_poisson42.txt

start ind: 6155.191 +- 19.190 samples
start amp: 654413.126 +- 2.8% counts
return ind: 14071.186 +- 18.833 samples
return dly : 999.494 +- 0.446 ns
return net: 149.824 +- 0.067 m
return amp: 2653582.920 +- 3.0% counts
rms_dly : 0.454 ns
rms_rng : 0.068 m
system offset 0.041 ns, 0.006 m, 0.004%
percent valid: 100.0

8tap_500MHz_L16_0128tap_16ns_square0.txt

start ind: 6150.156 +- 18.048 samples
start amp: 695283.548 +- 3.2% counts
return ind: 14067.583 +- 19.271 samples
return dly : 999.675 +- 0.742 ns
return net: 149.851 +- 0.111 m
return amp: 2833561.764 +- 3.0% counts
rms_dly : 0.748 ns
rms_rng : 0.112 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

8tap_500MHz_L16_0128tap_6ns_square0.txt

start ind: 6123.015 +- 18.661 samples
start amp: 649886.719 +- 3.2% counts
return ind: 14038.513 +- 18.912 samples
return dly : 999.432 +- 0.908 ns
return net: 149.815 +- 0.136 m
return amp: 2562839.613 +- 3.0% counts
rms_dly : 0.921 ns
rms_rng : 0.138 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

impulse-500MHz-L16.txt

start ind: 6099.698 +- 19.025 samples
start amp: 240986.573 +- 4.7% counts
return ind: 14016.965 +- 19.687 samples
return dly : 999.655 +- 0.760 ns
return net: 149.848 +- 0.114 m
return amp: 925472.357 +- 4.4% counts

rms_dly : 0.764 ns
rms_rng : 0.115 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

D.17 Interpolation factor = 16, sampling rate = 990 MHz, delay = 499 μ s

16tap_1000MHz_L16_0256tap_06ns_gaus10.txt

start ind: 11987.580 +- 19.109 samples
start amp: 665559.270 +- 3.6% counts
return ind: 19892.490 +- 19.240 samples
return dly : 499.047 +- 0.167 ns
return net: 74.807 +- 0.025 m
return amp: 2468492.590 +- 2.3% counts
rms_dly : 0.214 ns
rms_rng : 0.032 m
percent valid: 50.5050505051

16tap_1000MHz_L16_0256tap_06ns_gaus2.txt

start ind: 11989.889 +- 17.797 samples
start amp: 600118.535 +- 3.3% counts
return ind: 19895.071 +- 17.846 samples
return dly : 499.065 +- 0.144 ns
return net: 74.810 +- 0.022 m
return amp: 2221743.131 +- 2.4% counts
rms_dly : 0.185 ns
rms_rng : 0.028 m
percent valid: 49.7487437186

16tap_1000MHz_L16_0256tap_06ns_para0.txt

start ind: 11989.535 +- 19.602 samples
start amp: 667198.687 +- 3.7% counts
return ind: 19894.424 +- 19.504 samples
return dly : 499.046 +- 0.172 ns
return net: 74.807 +- 0.026 m
return amp: 2471726.212 +- 2.2% counts
rms_dly : 0.218 ns
rms_rng : 0.033 m
percent valid: 49.7487437186

16tap_1000MHz_L16_0256tap_10ns_gaus10.txt

start ind: 12018.222 +- 18.850 samples
start amp: 881011.343 +- 3.6% counts
return ind: 19921.768 +- 19.146 samples
return dly : 498.961 +- 0.121 ns
return net: 74.794 +- 0.018 m
return amp: 3361471.717 +- 2.3% counts
rms_dly : 0.250 ns
rms_rng : 0.038 m
percent valid: 49.7487437186

16tap_1000MHz_L16_0256tap_10ns_gaus2.txt

start ind: 12022.210 +- 15.884 samples
start amp: 714792.440 +- 3.7% counts
return ind: 19926.150 +- 15.646 samples

```

return dly : 498.986 +- 0.095 ns
return net: 74.798 +- 0.014 m
return amp: 2677628.050 +- 2.0% counts
rms_dly : 0.216 ns
rms_rng : 0.032 m
percent valid: 50.2512562814
*****

16tap_1000MHz_L16_0256tap_10ns_para0.txt
start ind: 12018.590 +- 18.956 samples
start amp: 884121.300 +- 3.1% counts
return ind: 19921.460 +- 19.043 samples
return dly : 498.919 +- 0.117 ns
return net: 74.788 +- 0.017 m
return amp: 3405706.270 +- 2.3% counts
rms_dly : 0.286 ns
rms_rng : 0.043 m
percent valid: 50.2512562814
*****

16tap_1000MHz_L16_0256tap_10ns_return.txt
start ind: 12056.879 +- 18.480 samples
start amp: 827590.182 +- 3.8% counts
return ind: 19960.869 +- 18.663 samples
return dly : 498.989 +- 0.112 ns
return net: 74.798 +- 0.017 m
return amp: 3176846.192 +- 2.2% counts
rms_dly : 0.221 ns
rms_rng : 0.033 m
percent valid: 49.7487437186
*****

16tap_1000MHz_L16_0256tap_16ns_gaus10.txt
start ind: 12062.687 +- 17.461 samples
start amp: 994339.747 +- 3.4% counts
return ind: 19964.101 +- 17.365 samples
return dly : 498.827 +- 0.127 ns
return net: 74.774 +- 0.019 m
return amp: 3913372.162 +- 2.1% counts
rms_dly : 0.375 ns
rms_rng : 0.056 m
percent valid: 49.7487437186
*****

16tap_1000MHz_L16_0256tap_16ns_gaus2.txt
start ind: 12068.333 +- 19.631 samples
start amp: 729659.909 +- 3.5% counts
return ind: 19972.343 +- 19.705 samples
return dly : 498.991 +- 0.104 ns
return net: 74.799 +- 0.016 m
return amp: 2746518.980 +- 2.3% counts
rms_dly : 0.216 ns
rms_rng : 0.032 m
percent valid: 49.7487437186**Error, invalid pixel
*****

16tap_1000MHz_L16_0256tap_16ns_para0.txt
start ind: 12066.727 +- 17.093 samples
start amp: 1006850.889 +- 3.0% counts
return ind: 19968.495 +- 17.231 samples

```

```

return dly : 498.849 +- 0.169 ns
return net: 74.777 +- 0.025 m
return amp: 3976737.717 +- 2.2% counts
rms_dly : 0.372 ns
rms_rng : 0.056 m
percent valid: 50.0
*****

16tap_1000MHz_L16_0256tap_16ns_poisson23.txt
start ind: 12055.040 +- 19.024 samples
start amp: 979579.690 +- 3.5% counts
return ind: 19957.210 +- 18.921 samples
return dly : 498.874 +- 0.160 ns
return net: 74.781 +- 0.024 m
return amp: 3912431.610 +- 2.1% counts
rms_dly : 0.345 ns
rms_rng : 0.052 m
percent valid: 50.2512562814
*****

16tap_1000MHz_L16_0256tap_16ns_poisson42.txt
start ind: 12055.606 +- 18.865 samples
start amp: 1028000.242 +- 3.4% counts
return ind: 19958.717 +- 19.047 samples
return dly : 498.934 +- 0.230 ns
return net: 74.790 +- 0.035 m
return amp: 4100899.778 +- 2.2% counts
rms_dly : 0.337 ns
rms_rng : 0.051 m
percent valid: 49.7487437186
*****

16tap_1000MHz_L16_0256tap_16ns_square0.txt
start ind: 12037.141 +- 22.014 samples
start amp: 1086375.677 +- 3.9% counts
return ind: 19945.182 +- 21.267 samples
return dly : 499.245 +- 0.504 ns
return net: 74.837 +- 0.076 m
return amp: 4420588.737 +- 2.0% counts
rms_dly : 0.509 ns
rms_rng : 0.076 m
percent valid: 49.7487437186
*****

16tap_1000MHz_L16_0256tap_6ns_square0.txt
start ind: 11989.600 +- 19.116 samples
start amp: 971252.590 +- 4.0% counts
return ind: 19892.960 +- 18.890 samples
return dly : 498.949 +- 0.244 ns
return net: 74.793 +- 0.037 m
return amp: 3683425.970 +- 2.1% counts
rms_dly : 0.336 ns
rms_rng : 0.050 m
percent valid: 50.2512562814
*****

impulse-1000MHz-L16.txt
start ind: 11936.960 +- 19.832 samples
start amp: 194042.030 +- 3.7% counts
return ind: 19844.480 +- 18.394 samples
return dly : 499.212 +- 0.418 ns

```


return net: 74.832 +- 0.063 m
return amp: 699666.310 +- 2.6% counts
rms_dly : 0.420 ns
rms_rng : 0.063 m
percent valid: 50.2512562814

D.18 Interpolation factor = 16, sampling rate = 990 MHz, delay = 999 μ s

16tap_1000MHz_L16_0256tap_06ns_gaus10.txt
start ind: 11572.874 +- 19.159 samples
start amp: 842961.513 +- 3.4% counts
return ind: 27405.985 +- 18.799 samples
return dly : 999.565 +- 0.292 ns
return net: 149.835 +- 0.044 m
return amp: 3235490.276 +- 3.1% counts
rms_dly : 0.292 ns
rms_rng : 0.044 m
system offset 0.167 ns, 0.025 m, 0.017%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_06ns_gaus2.txt
start ind: 11571.166 +- 18.020 samples
start amp: 763414.693 +- 3.4% counts
return ind: 27404.749 +- 17.916 samples
return dly : 999.595 +- 0.202 ns
return net: 149.839 +- 0.030 m
return amp: 2949903.090 +- 2.9% counts
rms_dly : 0.202 ns
rms_rng : 0.030 m
system offset 0.104 ns, 0.016 m, 0.010%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_06ns_para0.txt
start ind: 11571.015 +- 18.540 samples
start amp: 847479.744 +- 3.3% counts
return ind: 27404.965 +- 18.413 samples
return dly : 999.618 +- 0.278 ns
return net: 149.843 +- 0.042 m
return amp: 3252256.784 +- 3.2% counts
rms_dly : 0.280 ns
rms_rng : 0.042 m
system offset 0.167 ns, 0.025 m, 0.017%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_10ns_gaus10.txt
start ind: 11604.925 +- 19.730 samples
start amp: 1121235.487 +- 2.9% counts
return ind: 27436.161 +- 19.474 samples
return dly : 999.447 +- 0.131 ns
return net: 149.817 +- 0.020 m
return amp: 4389745.628 +- 2.6% counts
rms_dly : 0.187 ns

rms_rng : 0.028 m
system offset -0.148 ns, -0.022 m, -0.015%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_10ns_gaus2.txt
start ind: 11601.377 +- 19.495 samples
start amp: 908853.563 +- 3.3% counts
return ind: 27433.814 +- 19.604 samples
return dly : 999.523 +- 0.101 ns
return net: 149.828 +- 0.015 m
return amp: 3520752.759 +- 2.8% counts
rms_dly : 0.116 ns
rms_rng : 0.017 m
system offset -0.022 ns, -0.003 m, -0.002%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_10ns_para0.txt
start ind: 11603.568 +- 18.415 samples
start amp: 1129549.578 +- 3.2% counts
return ind: 27435.050 +- 18.519 samples
return dly : 999.462 +- 0.138 ns
return net: 149.819 +- 0.021 m
return amp: 4427130.789 +- 2.9% counts
rms_dly : 0.181 ns
rms_rng : 0.027 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_10ns_return.txt
start ind: 11641.211 +- 17.876 samples
start amp: 1050068.508 +- 3.3% counts
return ind: 27473.437 +- 17.281 samples
return dly : 999.509 +- 0.205 ns
return net: 149.826 +- 0.031 m
return amp: 4137914.151 +- 3.1% counts
rms_dly : 0.217 ns
rms_rng : 0.033 m
system offset 0.041 ns, 0.006 m, 0.004%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_16ns_gaus10.txt
start ind: 11648.302 +- 17.980 samples
start amp: 1260224.804 +- 3.2% counts
return ind: 27478.513 +- 17.777 samples
return dly : 999.382 +- 0.107 ns
return net: 149.807 +- 0.016 m
return amp: 5074357.658 +- 2.8% counts
rms_dly : 0.225 ns
rms_rng : 0.034 m
system offset -0.148 ns, -0.022 m, -0.015%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_16ns_gaus2.txt

start ind: 11649.824 +- 18.968 samples
start amp: 929677.528 +- 3.2% counts
return ind: 27482.623 +- 19.112 samples
return dly : 999.545 +- 0.091 ns
return net: 149.832 +- 0.014 m
return amp: 3619370.809 +- 2.9% counts
rms_dly : 0.097 ns
rms_rng : 0.015 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_16ns_para0.txt

start ind: 11645.377 +- 18.968 samples
start amp: 1274568.352 +- 3.2% counts
return ind: 27475.623 +- 18.765 samples
return dly : 999.384 +- 0.157 ns
return net: 149.808 +- 0.024 m
return amp: 5123235.568 +- 2.7% counts
rms_dly : 0.251 ns
rms_rng : 0.038 m
system offset -0.085 ns, -0.013 m, -0.009%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_16ns_poisson23.txt

start ind: 11638.136 +- 18.620 samples
start amp: 1253148.467 +- 3.6% counts
return ind: 27468.739 +- 18.872 samples
return dly : 999.407 +- 0.284 ns
return net: 149.811 +- 0.043 m
return amp: 5021197.442 +- 2.6% counts
rms_dly : 0.333 ns
rms_rng : 0.050 m
system offset -0.211 ns, -0.032 m, -0.021%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_16ns_poisson42.txt

start ind: 11633.040 +- 19.212 samples
start amp: 1295963.271 +- 3.0% counts
return ind: 27463.839 +- 19.323 samples
return dly : 999.419 +- 0.422 ns
return net: 149.813 +- 0.063 m
return amp: 5288628.372 +- 2.8% counts
rms_dly : 0.451 ns
rms_rng : 0.068 m
system offset -0.464 ns, -0.070 m, -0.046%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_16ns_square0.txt

start ind: 11618.457 +- 19.570 samples
start amp: 1384558.352 +- 2.9% counts

return ind: 27452.347 +- 19.299 samples
return dly : 999.614 +- 0.511 ns
return net: 149.842 +- 0.077 m
return amp: 5639909.568 +- 2.8% counts
rms_dly : 0.512 ns
rms_rng : 0.077 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

16tap_1000MHz_L16_0256tap_6ns_square0.txt

start ind: 11570.894 +- 20.018 samples
start amp: 1232722.020 +- 3.1% counts
return ind: 27402.935 +- 18.947 samples
return dly : 999.497 +- 0.505 ns
return net: 149.825 +- 0.076 m
return amp: 4810797.945 +- 2.9% counts
rms_dly : 0.512 ns
rms_rng : 0.077 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

impulse-1000MHz-L16.txt

start ind: 11523.296 +- 20.180 samples
start amp: 245693.422 +- 3.3% counts
return ind: 27360.080 +- 20.447 samples
return dly : 999.797 +- 0.405 ns
return net: 149.870 +- 0.061 m
return amp: 944666.955 +- 3.0% counts
rms_dly : 0.459 ns
rms_rng : 0.069 m
system offset 0.420 ns, 0.063 m, 0.042%
percent valid: 100.0

D.19 Interpolation factor = 32, sampling rate = 247.5 MHz, delay = 499 μ s

16tap_250MHz_L32_0512tap_10ns_return.txt

start ind: 6827.138 +- 34.066 samples
start amp: 208693.034 +- 4.2% counts
return ind: 10778.690 +- 34.107 samples
return dly : 498.933 +- 0.788 ns
return net: 74.790 +- 0.118 m
return amp: 793728.103 +- 3.4% counts
rms_dly : 0.828 ns
rms_rng : 0.124 m
system offset -0.198 ns, -0.030 m, -0.040%
percent valid: 100.0

D.20 Interpolation factor = 32, sampling rate = 247.5 MHz, delay = 999 μ s

16tap_250MHz_L32_0512tap_10ns_return.txt

start ind: 6832.414 +- 37.160 samples
start amp: 213190.897 +- 6.3% counts

return ind: 14748.966 +- 37.842 samples
return dly : 999.565 +- 0.758 ns
return net: 149.835 +- 0.114 m
return amp: 719553.862 +- 4.6% counts
rms_dly : 0.758 ns
rms_rng : 0.114 m
system offset -0.338 ns, -0.051 m, -0.034%
percent valid: 100.0

D.21 Interpolation factor = 32, sampling rate = 495 MHz, delay = 499 μ s

16tap_500MHz_L32_0512tap_10ns_return.txt
start ind: 13319.286 +- 38.306 ns
start amp: 550118.025 +- 3.1% counts
return ind: 21229.296 +- 38.150 ns
return dly : 499.369 +- 0.229 ns
return net: 74.855 +- 0.034 m
return amp: 2446406.995 +- 2.4% counts
rms_dly : 0.292 ns
rms_rng : 0.044 m

8tap_500MHz_L32_0256tap_06ns_gaus10.txt
start ind: 13255.779 +- 36.927 ns
start amp: 440979.131 +- 3.4% counts
return ind: 21166.543 +- 37.563 ns
return dly : 499.417 +- 0.445 ns
return net: 74.863 +- 0.067 m
return amp: 1924269.246 +- 1.5% counts
rms_dly : 0.500 ns
rms_rng : 0.075 m

8tap_500MHz_L32_0256tap_06ns_gaus2.txt
start ind: 13254.106 +- 38.518 ns
start amp: 398922.864 +- 3.5% counts
return ind: 21165.271 +- 38.855 ns
return dly : 499.442 +- 0.345 ns
return net: 74.866 +- 0.052 m
return amp: 1720386.859 +- 1.7% counts
rms_dly : 0.428 ns
rms_rng : 0.064 m

8tap_500MHz_L32_0256tap_06ns_para0.txt
start ind: 13248.191 +- 34.246 ns
start amp: 445621.588 +- 3.4% counts
return ind: 21159.171 +- 34.836 ns
return dly : 499.431 +- 0.472 ns
return net: 74.865 +- 0.071 m
return amp: 1932943.241 +- 1.9% counts
rms_dly : 0.531 ns
rms_rng : 0.080 m

8tap_500MHz_L32_0256tap_10ns_gaus10.txt
start ind: 13279.010 +- 36.309 ns
start amp: 579980.945 +- 3.2% counts
return ind: 21188.693 +- 36.503 ns
return dly : 499.349 +- 0.170 ns
return net: 74.852 +- 0.026 m
return amp: 2593728.719 +- 2.3% counts
rms_dly : 0.234 ns
rms_rng : 0.035 m

8tap_500MHz_L32_0256tap_10ns_gaus2.txt
start ind: 13286.668 +- 37.811 ns
start amp: 472227.141 +- 3.0% counts
return ind: 21197.482 +- 37.621 ns
return dly : 499.420 +- 0.112 ns
return net: 74.863 +- 0.017 m
return amp: 2061554.980 +- 2.2% counts
rms_dly : 0.258 ns
rms_rng : 0.039 m

8tap_500MHz_L32_0256tap_10ns_para0.txt
start ind: 13285.327 +- 39.158 ns
start amp: 586648.417 +- 3.1% counts
return ind: 21194.894 +- 39.135 ns
return dly : 499.341 +- 0.170 ns
return net: 74.851 +- 0.026 m
return amp: 2606468.839 +- 2.1% counts
rms_dly : 0.229 ns
rms_rng : 0.034 m

8tap_500MHz_L32_0256tap_16ns_gaus10.txt
start ind: 13325.854 +- 37.876 ns
start amp: 659232.608 +- 3.1% counts
return ind: 21233.955 +- 37.568 ns
return dly : 499.249 +- 0.160 ns
return net: 74.837 +- 0.024 m
return amp: 2999514.050 +- 2.2% counts
rms_dly : 0.171 ns
rms_rng : 0.026 m

8tap_500MHz_L32_0256tap_16ns_gaus2.txt
start ind: 13331.613 +- 37.376 ns
start amp: 482217.799 +- 2.9% counts
return ind: 21242.467 +- 37.476 ns
return dly : 499.423 +- 0.101 ns
return net: 74.863 +- 0.015 m
return amp: 2112411.417 +- 2.4% counts
rms_dly : 0.256 ns
rms_rng : 0.038 m

8tap_500MHz_L32_0256tap_16ns_para0.txt

start ind: 13325.226 +- 38.954 ns
start amp: 662320.201 +- 3.3% counts
return ind: 21233.362 +- 39.229 ns
return dly : 499.251 +- 0.182 ns
return net: 74.838 +- 0.027 m
return amp: 3030418.769 +- 2.3% counts
rms_dly : 0.192 ns
rms_rng : 0.029 m

8tap_500MHz_L32_0256tap_16ns_poisson23.txt

start ind: 13316.558 +- 36.991 ns
start amp: 650250.005 +- 3.2% counts
return ind: 21224.588 +- 36.655 ns
return dly : 499.244 +- 0.233 ns
return net: 74.837 +- 0.035 m
return amp: 2977278.709 +- 2.2% counts
rms_dly : 0.240 ns
rms_rng : 0.036 m

8tap_500MHz_L32_0256tap_16ns_poisson42.txt

start ind: 13314.055 +- 36.734 ns
start amp: 677607.754 +- 3.0% counts
return ind: 21223.357 +- 36.008 ns
return dly : 499.325 +- 0.378 ns
return net: 74.849 +- 0.057 m
return amp: 3130820.518 +- 2.2% counts
rms_dly : 0.402 ns
rms_rng : 0.060 m

8tap_500MHz_L32_0256tap_16ns_square0.txt

start ind: 13305.854 +- 40.759 ns
start amp: 716681.352 +- 3.3% counts
return ind: 21217.573 +- 39.467 ns
return dly : 499.477 +- 0.864 ns
return net: 74.872 +- 0.130 m
return amp: 3350729.714 +- 2.3% counts
rms_dly : 0.911 ns
rms_rng : 0.137 m

8tap_500MHz_L32_0256tap_6ns_square0.txt

start ind: 13249.608 +- 39.907 ns
start amp: 672963.889 +- 3.3% counts
return ind: 21157.146 +- 38.420 ns
return dly : 499.213 +- 0.633 ns
return net: 74.832 +- 0.095 m
return amp: 3010601.437 +- 2.9% counts
rms_dly : 0.634 ns
rms_rng : 0.095 m

D.22 Interpolation factor = 32, sampling rate = 495 MHz, delay = 999 μ s

8tap_500MHz_L32_0256tap_06ns_gaus10.txt
start ind: 12246.345 +- 37.024 samples
start amp: 335504.931 +- 2.9% counts
return ind: 28080.931 +- 39.405 samples
return dly : 999.658 +- 0.471 ns
return net: 149.849 +- 0.071 m
return amp: 1124627.483 +- 2.1% counts
rms_dly : 0.477 ns
rms_rng : 0.072 m
system offset 0.294 ns, 0.044 m, 0.029%
percent valid: 100.0

8tap_500MHz_L32_0256tap_06ns_gaus10.txt
start ind: 13255.779 +- 36.927 ns
start amp: 440979.131 +- 3.4% counts
return ind: 21166.543 +- 37.563 ns
return dly : 499.417 +- 0.445 ns
return net: 74.863 +- 0.067 m
return amp: 1924269.246 +- 1.5% counts
rms_dly : 0.500 ns
rms_rng : 0.075 m

8tap_500MHz_L32_0256tap_06ns_gaus2.txt
start ind: 13254.106 +- 38.518 ns
start amp: 398922.864 +- 3.5% counts
return ind: 21165.271 +- 38.855 ns
return dly : 499.442 +- 0.345 ns
return net: 74.866 +- 0.052 m
return amp: 1720386.859 +- 1.7% counts
rms_dly : 0.428 ns
rms_rng : 0.064 m

8tap_500MHz_L32_0256tap_06ns_para0.txt
start ind: 13248.191 +- 34.246 ns
start amp: 445621.588 +- 3.4% counts
return ind: 21159.171 +- 34.836 ns
return dly : 499.431 +- 0.472 ns
return net: 74.865 +- 0.071 m
return amp: 1932943.241 +- 1.9% counts
rms_dly : 0.531 ns
rms_rng : 0.080 m

8tap_500MHz_L32_0256tap_10ns_gaus10.txt
start ind: 13279.010 +- 36.309 ns
start amp: 579980.945 +- 3.2% counts
return ind: 21188.693 +- 36.503 ns
return dly : 499.349 +- 0.170 ns
return net: 74.852 +- 0.026 m
return amp: 2593728.719 +- 2.3% counts

rms_dly : 0.234 ns
rms_rng : 0.035 m

8tap_500MHz_L32_0256tap_10ns_gaus2.txt
start ind: 13286.668 +- 37.811 ns
start amp: 472227.141 +- 3.0% counts
return ind: 21197.482 +- 37.621 ns
return dly : 499.420 +- 0.112 ns
return net: 74.863 +- 0.017 m
return amp: 2061554.980 +- 2.2% counts
rms_dly : 0.258 ns
rms_rng : 0.039 m

8tap_500MHz_L32_0256tap_10ns_para0.txt
start ind: 13285.327 +- 39.158 ns
start amp: 586648.417 +- 3.1% counts
return ind: 21194.894 +- 39.135 ns
return dly : 499.341 +- 0.170 ns
return net: 74.851 +- 0.026 m
return amp: 2606468.839 +- 2.1% counts
rms_dly : 0.229 ns
rms_rng : 0.034 m

8tap_500MHz_L32_0256tap_16ns_gaus10.txt
start ind: 13325.854 +- 37.876 ns
start amp: 659232.608 +- 3.1% counts
return ind: 21233.955 +- 37.568 ns
return dly : 499.249 +- 0.160 ns
return net: 74.837 +- 0.024 m
return amp: 2999514.050 +- 2.2% counts
rms_dly : 0.171 ns
rms_rng : 0.026 m

8tap_500MHz_L32_0256tap_16ns_gaus2.txt
start ind: 13331.613 +- 37.376 ns
start amp: 482217.799 +- 2.9% counts
return ind: 21242.467 +- 37.476 ns
return dly : 499.423 +- 0.101 ns
return net: 74.863 +- 0.015 m
return amp: 2112411.417 +- 2.4% counts
rms_dly : 0.256 ns
rms_rng : 0.038 m

8tap_500MHz_L32_0256tap_16ns_para0.txt
start ind: 13325.226 +- 38.954 ns
start amp: 662320.201 +- 3.3% counts
return ind: 21233.362 +- 39.229 ns
return dly : 499.251 +- 0.182 ns
return net: 74.838 +- 0.027 m
return amp: 3030418.769 +- 2.3% counts
rms_dly : 0.192 ns

rms_rng : 0.029 m

8tap_500MHz_L32_0256tap_16ns_poisson23.txt

start ind: 13316.558 +- 36.991 ns

start amp: 650250.005 +- 3.2% counts

return ind: 21224.588 +- 36.655 ns

return dly : 499.244 +- 0.233 ns

return net: 74.837 +- 0.035 m

return amp: 2977278.709 +- 2.2% counts

rms_dly : 0.240 ns

rms_rng : 0.036 m

8tap_500MHz_L32_0256tap_16ns_poisson42.txt

start ind: 13314.055 +- 36.734 ns

start amp: 677607.754 +- 3.0% counts

return ind: 21223.357 +- 36.008 ns

return dly : 499.325 +- 0.378 ns

return net: 74.849 +- 0.057 m

return amp: 3130820.518 +- 2.2% counts

rms_dly : 0.402 ns

rms_rng : 0.060 m

8tap_500MHz_L32_0256tap_16ns_square0.txt

start ind: 13305.854 +- 40.759 ns

start amp: 716681.352 +- 3.3% counts

return ind: 21217.573 +- 39.467 ns

return dly : 499.477 +- 0.864 ns

return net: 74.872 +- 0.130 m

return amp: 3350729.714 +- 2.3% counts

rms_dly : 0.911 ns

rms_rng : 0.137 m

8tap_500MHz_L32_0256tap_6ns_square0.txt

start ind: 13249.608 +- 39.907 ns

start amp: 672963.889 +- 3.3% counts

return ind: 21157.146 +- 38.420 ns

return dly : 499.213 +- 0.633 ns

return net: 74.832 +- 0.095 m

return amp: 3010601.437 +- 2.9% counts

rms_dly : 0.634 ns

rms_rng : 0.095 m

D.23 Interpolation factor = 32, sampling rate = 990 MHz, delay = 499 μ s

16tap_1000MHz_L32_0512tap_06ns_gaus2.txt

start ind: 23157.586 +- 35.054 samples

start amp: 607821.207 +- 3.5% counts

return ind: 38969.000 +- 35.322 samples

return dly : 499.098 +- 0.157 ns

return net: 74.815 +- 0.023 m

return amp: 2194951.310 +- 2.2% counts
rms_dly : 0.181 ns
rms_rng : 0.027 m
system offset -0.135 ns, -0.020 m, -0.027%
percent valid: 100.0

8tap_500MHz_L32_0256tap_06ns_gaus10.txt
start ind: 13255.779 +- 36.927 ns
start amp: 440979.131 +- 3.4% counts
return ind: 21166.543 +- 37.563 ns
return dly : 499.417 +- 0.445 ns
return net: 74.863 +- 0.067 m
return amp: 1924269.246 +- 1.5% counts
rms_dly : 0.500 ns
rms_rng : 0.075 m

8tap_500MHz_L32_0256tap_06ns_gaus2.txt
start ind: 13254.106 +- 38.518 ns
start amp: 398922.864 +- 3.5% counts
return ind: 21165.271 +- 38.855 ns
return dly : 499.442 +- 0.345 ns
return net: 74.866 +- 0.052 m
return amp: 1720386.859 +- 1.7% counts
rms_dly : 0.428 ns
rms_rng : 0.064 m

8tap_500MHz_L32_0256tap_06ns_para0.txt
start ind: 13248.191 +- 34.246 ns
start amp: 445621.588 +- 3.4% counts
return ind: 21159.171 +- 34.836 ns
return dly : 499.431 +- 0.472 ns
return net: 74.865 +- 0.071 m
return amp: 1932943.241 +- 1.9% counts
rms_dly : 0.531 ns
rms_rng : 0.080 m

8tap_500MHz_L32_0256tap_10ns_gaus10.txt
start ind: 13279.010 +- 36.309 ns
start amp: 579980.945 +- 3.2% counts
return ind: 21188.693 +- 36.503 ns
return dly : 499.349 +- 0.170 ns
return net: 74.852 +- 0.026 m
return amp: 2593728.719 +- 2.3% counts
rms_dly : 0.234 ns
rms_rng : 0.035 m

8tap_500MHz_L32_0256tap_10ns_gaus2.txt
start ind: 13286.668 +- 37.811 ns
start amp: 472227.141 +- 3.0% counts
return ind: 21197.482 +- 37.621 ns
return dly : 499.420 +- 0.112 ns

return net: 74.863 +- 0.017 m
return amp: 2061554.980 +- 2.2% counts
rms_dly : 0.258 ns
rms_rng : 0.039 m

8tap_500MHz_L32_0256tap_10ns_para0.txt
start ind: 13285.327 +- 39.158 ns
start amp: 586648.417 +- 3.1% counts
return ind: 21194.894 +- 39.135 ns
return dly : 499.341 +- 0.170 ns
return net: 74.851 +- 0.026 m
return amp: 2606468.839 +- 2.1% counts
rms_dly : 0.229 ns
rms_rng : 0.034 m

8tap_500MHz_L32_0256tap_16ns_gaus10.txt
start ind: 13325.854 +- 37.876 ns
start amp: 659232.608 +- 3.1% counts
return ind: 21233.955 +- 37.568 ns
return dly : 499.249 +- 0.160 ns
return net: 74.837 +- 0.024 m
return amp: 2999514.050 +- 2.2% counts
rms_dly : 0.171 ns
rms_rng : 0.026 m

8tap_500MHz_L32_0256tap_16ns_gaus2.txt
start ind: 13331.613 +- 37.376 ns
start amp: 482217.799 +- 2.9% counts
return ind: 21242.467 +- 37.476 ns
return dly : 499.423 +- 0.101 ns
return net: 74.863 +- 0.015 m
return amp: 2112411.417 +- 2.4% counts
rms_dly : 0.256 ns
rms_rng : 0.038 m

8tap_500MHz_L32_0256tap_16ns_para0.txt
start ind: 13325.226 +- 38.954 ns
start amp: 662320.201 +- 3.3% counts
return ind: 21233.362 +- 39.229 ns
return dly : 499.251 +- 0.182 ns
return net: 74.838 +- 0.027 m
return amp: 3030418.769 +- 2.3% counts
rms_dly : 0.192 ns
rms_rng : 0.029 m

8tap_500MHz_L32_0256tap_16ns_poisson23.txt
start ind: 13316.558 +- 36.991 ns
start amp: 650250.005 +- 3.2% counts
return ind: 21224.588 +- 36.655 ns
return dly : 499.244 +- 0.233 ns
return net: 74.837 +- 0.035 m

return amp: 2977278.709 +- 2.2% counts
rms_dly : 0.240 ns
rms_rng : 0.036 m

8tap_500MHz_L32_0256tap_16ns_poisson42.txt
start ind: 13314.055 +- 36.734 ns
start amp: 677607.754 +- 3.0% counts
return ind: 21223.357 +- 36.008 ns
return dly : 499.325 +- 0.378 ns
return net: 74.849 +- 0.057 m
return amp: 3130820.518 +- 2.2% counts
rms_dly : 0.402 ns
rms_rng : 0.060 m

8tap_500MHz_L32_0256tap_16ns_square0.txt
start ind: 13305.854 +- 40.759 ns
start amp: 716681.352 +- 3.3% counts
return ind: 21217.573 +- 39.467 ns
return dly : 499.477 +- 0.864 ns
return net: 74.872 +- 0.130 m
return amp: 3350729.714 +- 2.3% counts
rms_dly : 0.911 ns
rms_rng : 0.137 m

8tap_500MHz_L32_0256tap_6ns_square0.txt
start ind: 13249.608 +- 39.907 ns
start amp: 672963.889 +- 3.3% counts
return ind: 21157.146 +- 38.420 ns
return dly : 499.213 +- 0.633 ns
return net: 74.832 +- 0.095 m
return amp: 3010601.437 +- 2.9% counts
rms_dly : 0.634 ns
rms_rng : 0.095 m

D.24 Interpolation factor = 32, sampling rate = 990 MHz, delay = 999 μ s

16tap_1000MHz_L32_0512tap_06ns_gaus2.txt
start ind: 23153.793 +- 40.141 samples
start amp: 601609.828 +- 3.6% counts
return ind: 54827.207 +- 39.352 samples
return dly : 999.792 +- 0.180 ns
return net: 149.869 +- 0.027 m
return amp: 1982311.138 +- 2.0% counts
rms_dly : 0.278 ns
rms_rng : 0.042 m
system offset 0.262 ns, 0.039 m, 0.026%
percent valid: 100.0

8tap_500MHz_L32_0256tap_06ns_gaus10.txt
start ind: 13255.779 +- 36.927 ns

start amp: 440979.131 +- 3.4% counts
return ind: 21166.543 +- 37.563 ns
return dly : 499.417 +- 0.445 ns
return net: 74.863 +- 0.067 m
return amp: 1924269.246 +- 1.5% counts
rms_dly : 0.500 ns
rms_rng : 0.075 m

8tap_500MHz_L32_0256tap_06ns_gaus2.txt
start ind: 13254.106 +- 38.518 ns
start amp: 398922.864 +- 3.5% counts
return ind: 21165.271 +- 38.855 ns
return dly : 499.442 +- 0.345 ns
return net: 74.866 +- 0.052 m
return amp: 1720386.859 +- 1.7% counts
rms_dly : 0.428 ns
rms_rng : 0.064 m

8tap_500MHz_L32_0256tap_06ns_para0.txt
start ind: 13248.191 +- 34.246 ns
start amp: 445621.588 +- 3.4% counts
return ind: 21159.171 +- 34.836 ns
return dly : 499.431 +- 0.472 ns
return net: 74.865 +- 0.071 m
return amp: 1932943.241 +- 1.9% counts
rms_dly : 0.531 ns
rms_rng : 0.080 m

8tap_500MHz_L32_0256tap_10ns_gaus10.txt
start ind: 13279.010 +- 36.309 ns
start amp: 579980.945 +- 3.2% counts
return ind: 21188.693 +- 36.503 ns
return dly : 499.349 +- 0.170 ns
return net: 74.852 +- 0.026 m
return amp: 2593728.719 +- 2.3% counts
rms_dly : 0.234 ns
rms_rng : 0.035 m

8tap_500MHz_L32_0256tap_10ns_gaus2.txt
start ind: 13286.668 +- 37.811 ns
start amp: 472227.141 +- 3.0% counts
return ind: 21197.482 +- 37.621 ns
return dly : 499.420 +- 0.112 ns
return net: 74.863 +- 0.017 m
return amp: 2061554.980 +- 2.2% counts
rms_dly : 0.258 ns
rms_rng : 0.039 m

8tap_500MHz_L32_0256tap_10ns_para0.txt
start ind: 13285.327 +- 39.158 ns
start amp: 586648.417 +- 3.1% counts

return ind: 21194.894 +- 39.135 ns
return dly : 499.341 +- 0.170 ns
return net: 74.851 +- 0.026 m
return amp: 2606468.839 +- 2.1% counts
rms_dly : 0.229 ns
rms_rng : 0.034 m

8tap_500MHz_L32_0256tap_16ns_gaus10.txt
start ind: 13325.854 +- 37.876 ns
start amp: 659232.608 +- 3.1% counts
return ind: 21233.955 +- 37.568 ns
return dly : 499.249 +- 0.160 ns
return net: 74.837 +- 0.024 m
return amp: 2999514.050 +- 2.2% counts
rms_dly : 0.171 ns
rms_rng : 0.026 m

8tap_500MHz_L32_0256tap_16ns_gaus2.txt
start ind: 13331.613 +- 37.376 ns
start amp: 482217.799 +- 2.9% counts
return ind: 21242.467 +- 37.476 ns
return dly : 499.423 +- 0.101 ns
return net: 74.863 +- 0.015 m
return amp: 2112411.417 +- 2.4% counts
rms_dly : 0.256 ns
rms_rng : 0.038 m

8tap_500MHz_L32_0256tap_16ns_para0.txt
start ind: 13325.226 +- 38.954 ns
start amp: 662320.201 +- 3.3% counts
return ind: 21233.362 +- 39.229 ns
return dly : 499.251 +- 0.182 ns
return net: 74.838 +- 0.027 m
return amp: 3030418.769 +- 2.3% counts
rms_dly : 0.192 ns
rms_rng : 0.029 m

8tap_500MHz_L32_0256tap_16ns_poisson23.txt
start ind: 13316.558 +- 36.991 ns
start amp: 650250.005 +- 3.2% counts
return ind: 21224.588 +- 36.655 ns
return dly : 499.244 +- 0.233 ns
return net: 74.837 +- 0.035 m
return amp: 2977278.709 +- 2.2% counts
rms_dly : 0.240 ns
rms_rng : 0.036 m

8tap_500MHz_L32_0256tap_16ns_poisson42.txt
start ind: 13314.055 +- 36.734 ns
start amp: 677607.754 +- 3.0% counts
return ind: 21223.357 +- 36.008 ns

return dly : 499.325 +- 0.378 ns
return net: 74.849 +- 0.057 m
return amp: 3130820.518 +- 2.2% counts
rms_dly : 0.402 ns
rms_rng : 0.060 m

8tap_500MHz_L32_0256tap_16ns_square0.txt
start ind: 13305.854 +- 40.759 ns
start amp: 716681.352 +- 3.3% counts
return ind: 21217.573 +- 39.467 ns
return dly : 499.477 +- 0.864 ns
return net: 74.872 +- 0.130 m
return amp: 3350729.714 +- 2.3% counts
rms_dly : 0.911 ns
rms_rng : 0.137 m

8tap_500MHz_L32_0256tap_6ns_square0.txt
start ind: 13249.608 +- 39.907 ns
start amp: 672963.889 +- 3.3% counts
return ind: 21157.146 +- 38.420 ns
return dly : 499.213 +- 0.633 ns
return net: 74.832 +- 0.095 m
return amp: 3010601.437 +- 2.9% counts
rms_dly : 0.634 ns
rms_rng : 0.095 m

REFERENCES

- [1] Steven P. Jordan, "Range Estimation Algorithm Comparison in 3-D Flash LADAR Data," M.S. thesis, Dept. Elect. And Comp. Eng., Air Force Institute of Tech., Wright-Patterson AFB, OH, 2009.
- [2] Xiaodong Zhi, "A progressive quality control to improve the accuracy of LIDAR data processing," *ISPRS Journal*, vol. 37, no. B1, pp. 421-426, 2008.
- [3] Nora Csanyi, "Improvement of LIDAR data accuracy using LIDAR-specific ground targets," *Photogrammetric Eng. & Remote Sensing*, vol. 73, no. 4, pp. 385-396, Apr., 2007.
- [4] Richard D. Richmond and Stephen C. Cain. "Direct-Detection LADAR Systems". *Tutorial Text in Optical Engineering v. TT85*. Society of Photo-Optical Instrumentation Engineers Press, 2010.
- [5] Peter Cho, Hyrum Anderson, Robert Hatch, and Prem Ramaswami, "Real-time 3-D LADAR imaging," in *HPCMP Users Group Conf.*, 2006 © IEEE. doi: 0-7695-2797-3/06
- [6] Jason R. McMahon, Stephen Cain, and Richard Martin, "Improving 3-D LADAR range estimation via spatial filtering," *IEEEAC*, Paper #1172, Oct, 2008.
- [7] Charles R. Burris, "An Estimation Theory Approach to Detection and Ranging of Obscured Targets in 3-D LADAR Data," M.S. thesis, Dept. Elect. and Comp. Eng., Air Force Institute of Tech., Wright-Patterson AFB, OH, 2006.
- [8] Keil Schmid, Kirk Waters, Lindy Dingerson, Brian Hadley, Rebecca Mataosky, Jamie Carter, and Jennifer Dare, "LIDAR 101: an introduction to LIDAR technology, data, and applications," NOAA Coastal Services Center, Charleston, SC, 2008.
- [9] Sun Zhi-hui, Deng Jia-hao, and Yan Xiao-wei, "Target detection and recognition techniques of line imaging LADAR sensor," in *Int. Symp. On Photoelectronic Detection and Imaging*, 2009 © SPIE. doi: 10.1117/12.836547
- [10] Ove Steinvall, Lena Klasen, Christina Gronwall, Ulf Soderman, Simon Ahlberg, Asa Persson, Magnus Elmqvist, Hakan Larsson, Dietmar Letalick, Pierre Anderson, Tomas Carlsson, and Markus Henriksson, "3 D laser sensing at FOI – overview and a system perspective," in *Proc. SPIE: Laser Radar Technology and Applications IX*, vol. 5412. doi: 10.1117/12.553808
- [11] Lui Zheng-jun, Li Qi, and Wang Qi, "Coherent LADAR range image for target recognition using back-propagation neural network," in *2010 Academic Symp. On Optoelectronics Tech.*, 2010 © IEEE. doi: 10.1109/RCSLPT.2010.5615269

- [12] Zhiyu Xiang and Eryong Wu, "Design and Calibration of a fast 3D scanning LADAR," in *Proc. 2006 IEEE Int. Conf. on Mechatronics and Automation*, 2006 © IEEE. doi: 10.1109/ICMA.2006.257498
- [13] Roger Stettner and Howard Bailey, "Eye-safe laser radar 3D imaging", *Proc. from Society of Photo-Optical Instrumentation Engineers*, 2004 © SPIE. doi: 10.1117/12.440125
- [14] R.M. Marino and W.R. Davis, Jr., "Jigsaw: a foliage-penetrating imaging laser radar system," *Lincoln Laboratory Journal*, vol. 15, no. 1, pp. 23–36, 2005.
- [15] Qinfen Zheng, Sandor Der, and Rama Chellappa, "Model-based target recognition in pulsed LADAR imagery," *IEEE Trans. Image Process.*, vol. 10, no. 4, pp. 565-572, Apr, 2001, doi: 10.1109/83.913591
- [16] Air Force Research Lab. (2007, Aug. 29). *Wright-Patterson Air Force Base* [Online]. Available: <http://www.wpafb.af.mil/shared/media/gg/gallery/hires/AFG-070829-007.jpg>
- [17] B.C. Redman, A.J. Griffis, and E.B. Schibley, "Streak tube imaging LIDAR (STIL) for 3-D imaging of terrestrial targets," Arete Assoc. Tuscan Operations, Tuscan, AZ, 2000.
- [18] Chen Xiao-qing, Ma Jun-guo, Zhao Hong-zhong, and Fu Qiang, "Survey of Automatic target recognition technology for LADAR," in *Int. Symp. On Photoelectronic Detection and Imaging*, 2009 © SPIE. doi: 10.1117/12.836395
- [19] DSPlogic (2011, June 10). *AFRL Awards DSPlogic Follow-on LADAR Tech. Contract* [Online]. Available http://www.dsplogic.com/news/20110610_afrl
- [20] Ahmad Darabiha, Jonathan Rose, and W. James MacLean, "Video-rate stereo depth measurement on programmable hardware," in *Proc. 2003 IEEE Computer Society Conf. CVPR*, 2003 © IEEE. doi: 1063-6919/03
- [21] Brahim Betkaoui, David Thomas, and Wayne Luk, "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing," in *2010 Int. Conf. Field-Programmable Tech.*, 2010 © IEEE. doi: 10.1109/FPT.2010.5681761
- [22] Shuai Che, Jeremy W. Sheaffer, Kevin Skadron, and John Lach, "Accelerating compute-intensive applications with GPUs and FPGAs," in *2008 Symp. Application Specific Processors*, 2008 © IEEE. doi: 10.1109/SASP.2008.4570793
- [23] MATLAB 2009b. Natick, Massachusetts: The MathWorks Inc., 2009.
- [24] Quartus II Version 9.1. San Jose, California: Altera Corporation, 2010.

- [25] Modelsim-Altera Web Edition v6.6d. Wilsonville, Oregon: Mentor Graphics Corporation, 2011.
- [26] Martial Hebert, "Active and passive range sensing for robotics," in *Proc. 2000 IEEE Inter. Conf. Robotics & Automation*, 2000 © IEEE. doi: 10.1109/ROBOT.2000.844046
- [27] B. Gallardo-Hernando, J.M. Munoz-Ferreras, F. Perez-Martinez, and J.M. Lazaro-Gasco, "Performance evaluation for imaging laser radars with focal plane array," in *2007 IET Radar Systems Inter. Conf.*, Edinburgh, United Kingdom, Oct, 2007, pp. 1-5.
- [28] Robin R. Burton, John Schott, and Scott Brown, "Elastic LADAR modeling for synthetic image applications," in *Proc. Imaging Spectrometry VIII*, 2002 © SPIE.
- [29] J.M. McKinion, J.L. Willers, and J.N. Jenkins, "Comparing high density LIDAR and medium resolution GPS generated elevation data for predicting yield stability." *Computers and Electronics in Agriculture*, vol. 74, pp. 244-249, Aug., 2010.
- [30] Tristan Allouis, Sylvie Durrieu, Juan Cuesta, Patrick Chazette, Pierre H. Flamant, and Pierre Coutron, "Assessment of tree and crown heights of a maritime pine forest at plot level using a fullwaveform ultraviolet LIDAR prototype," in *2010 IEEE Inter. Geoscience Remote Sensing Symposium*, 2010 © IEEE. doi: 10.1109/IGARSS.2010.5651026
- [31] David E. Smith, Maria Zuber, Gregory Neumann, Frank Lemoine, Erwan Mazarico, Mark Torrence, Jan McGarry, Ddavid Rowlands, James Head III, Thomas Duxbury, Oded Aharonson, Paul Lucey, Mark Robinson, Olivier Barnouin, John Cavanaugh, Xiaoli Sun, Peter Liiva, Dan-dan Mao, James Smith, and Arlin Bartels, "Initial observations from the Lunar Orbiter Laser Altimeter (LOLA)," *Geophysical Research Letter*, vol. 37, 2010. doi: 10.1029/2010GL043751
- [32] P. David Fisher and John W. Pyhtila, "Timing Quantization Error in LIDAR speed-measurement devices," *IEEE Trans. Veh. Technol.*, vol. 49, no. 1, pp. 276-280, Jan., 2000.
- [33] John Wallace. (2006, July 1). "Real Time comes to three-dimensional LADAR," *Laser Focus World* [Online]. Available: <http://www.laserfocusworld.com/articles/print/volume-42/issue-7/optoelectronics-world/real-time-comes-to-three-dimensional-ladar.html>
- [34] James P. Siepmann and Adam Rybaltowski, "Integrable ultra-compact, high resolution, real-time MEMS LADAR for the individual soldier," in *2005 IEEE Military Communications Conf.*, 2005 © IEEE. doi: 10.1109/MILCOM.2005.1606131

- [35] A.M. Wallace R.C.W. Sung, G.S. Buller, R.D. Harkins, R.E. Warburton, and R.A. Lamb, "Detecting and characterizing returns in a pulsed LADAR system," in *IEE Proc. Vis. Image Signal Process.*, vol. 153, no. 2, pp. 160-172, Apr., 2006. doi: 10.1049/ip-vis:20045023
- [36] Sanjit K. Mitra, *Digital Signal Processing: A Computer Based Approach 3rd ed.*, McGraw-Hill, 2006.
- [37] Daniel Flores-Tapia, Gabriel Thomas, and Stephen Pistorius, "A comparison of interpolation methods for breast microwave radar imaging," in *2009 IEEE Engineering Medicine Biology Society Conf.*, 2009 © IEEE. doi: 10.1109/IEMBS.2009.5333341
- [38] Harry J. Decker and James J. Zakrajsek, "Comparison of interpolation methods as applied to time synchronous averaging," presented at the *Society for Machine Failure Prevention Technology Conference*, Virginia Beach, VA, 1999.
- [39] William G. Hawkins, "FFT interpolation for arbitrary factors: a comparison to cubic spline interpolation and linear interpolation," in *1994 IEEE Nuclear Science Symp. and Medical Imaging Conf.*, 1994 © IEEE. doi: 10.1109/NSSMIC.1994.474614
- [40] J.P. Lewis, "Fast Template Matching," presented at the *Proceedings of Vision Interface 1995*, Quebec City, Canada, pp. 120-123, 1995.
- [41] D.C.M. Bilsby, R.L. Walker, and R.W.M. Smith, "Comparison of a programmable DSP and a FPGA for real-time multiscale convolution," *IEEE Colloq. High Performance Architectures for Real-Time Image Processing*, 1998 © IEEE. doi: 10.1049/ic:19980044
- [42] John L. Hennessy and David A. Patterson, "Embedded Systems," in *Computer Architecture: A Quantitative Approach*, 4th ed., Morgan Kaufmann, 2007, app. D, pp. 2-11.
- [43] Ben Cope, "Implementation of 2D convolution on FPGA, GPU, and CPU," Imperial College Report
- [44] Chris Rowen, "The world's fastest DSP core: breaking the 100 GMAC/s barrier," Tensilica Inc., Santa Clara, CA, 2011.
- [45] *Texas Instruments Data Manual*, rev. c, TI, Dallas, TX, 2012, pp. 13.
- [46] Volnei A. Pedroni, *Circuit Design and Simulation with VHDL*, 2nd ed. Cambridge, MA: MIT Press, 2010.
- [47] Edwin J. Tan and Wendi B. Heinzelman, "DSP architectures: past, present, and future," *Computer Architecture News*, vol. 31, no. 3, Jun., 2003, pp. 6-19.

- [48] Muhammad Faisal, Randolph Schiffer, Marek Flaska, Sara Pozzi, David Wentzloff, “A correlation-based pulse detection technique for gamma-ray/neuron detectors.” *Nucl. Instr. And Meth. A*, 2010. doi: 10.1016/j.nima.2010.10.072
- [49] Xiaoming Lai and Hans Torp, “Interpolation methods for time-delay estimation using cross-correlation method for blood velocity measurement.” *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 46, no. 2, pp. 277-290, Mar, 1999. doi: 10.1109/58.753016
- [50] Altera. “FIR Compiler MegaCore Function User Guide.” Altera Complete Design Suite version 11.1. November 2011.

BIOGRAPHICAL SKETCH

Jarrold Paul Brown received his Bachelors of Science degree in Electrical Engineering (BSEE) with Cum Laude from University of Florida in 2008 and is currently a graduate student at Florida State University (2012). He has made consecutive Dean's list appearances throughout his entire undergraduate and graduate school experience. At University of Florida he received the Engineering "E" award based on exceptionally high GPA.

His engineering career began with an internship for NASA at Kennedy Space Center (2008) which earned a nomination for the Cooperative Education and Internship Association Outstanding Internship Award. He continued his career in the aeronautical industry with the PALACE Acquire Program (2009) for the United States Air Force at Eglin Air Force Base. The three-year training and development plan presented a combination of computer engineering work experience and funded graduate school. He is currently a civil service computer engineer at Eglin Air Force Base. His main research interests include laser detection and ranging, algorithm and architecture co-optimization, and sensor-seeker technology.