

# INLINE COHERENT IMAGING BASED CONTROL OF LASER MACHINING PROCESSES USING A FIELD-PROGRAMMABLE GATE ARRAY

by

ETHAN JENKINS

A thesis submitted to the Graduate Program in  
Physics, Engineering Physics and Astronomy  
in conformity with the requirements for the  
Degree of Master of Applied Science

Queen's University  
Kingston, Ontario, Canada  
September 2015

Copyright © Ethan Jenkins, 2015

## Abstract

Lasers are becoming an increasingly popular tool for use in industrial manufacturing settings. Their high power density, contact-free nature, and ease of automation give them an edge over traditional methods in processes such as welding, cutting, and drilling. However, laser beams are a difficult tool to control since they are made of light. Therefore, they cannot be monitored in the same way as a mechanical tool with constant physical dimensions. A recent invention, inline coherent imaging (ICI), uses the interference properties of light to monitor the depth of the processing beam at high speeds. However, the computational demands of ICI data processing are intensive, and it is difficult for standard computers to keep up with the high volumes of data that are output by an ICI system. As a result, the real-time control capabilities of ICI are limited by the rate at which the computer can process the data. In this work, I present a system that uses a field-programmable gate array (FPGA) to process ICI data with a high level of determinism and low latency compared to computer-based ICI. The algorithms that were developed to take advantage of the FPGA's architecture are presented, followed by a detailed description of how they were implemented programmatically. I also describe the hardware that was used to perform experiments with closed-loop ICI feedback control. Finally, I present the results of tests done with two applications, autofocus and welding depth control. I show that FPGA-based ICI achieves superior performance to computer-based control, pushing the latency of an ICI calculation down from  $3600 \pm 900 \mu\text{s}$  to  $54 \pm 5 \mu\text{s}$ . Areas of future work include fine-tuned welding control, implementation of ASIC and/or advanced interpolation or FFT algorithms, and investigation of other laser processes such as additive manufacturing and surgical bone ablation.

# Acknowledgements

This work, which was conducted under the guidance and supervision of Prof. James M. Fraser, would not have been possible without his support. When interacting with James in an academic environment, it is obvious that he is passionate about teaching and mentoring his students. During our weekly meetings when we discussed progress, worked out issues, and planned for future work, he facilitated the smooth running of this project. I would also like to acknowledge my colleagues in James' research group: Yang Ji, Christopher Galbraith, Jordan Kanko, Cara Yin, Matthew Windeler, Allison Sibley, Alex Grindal, Stephen Nestor, Sacha Ruzzante, Mitchell Anderson, and Meghan Beattie. Working with this group has been an enriching experience thanks to the the friendly, professional, and efficient atmosphere in our office and lab.

This project also would have been impossible without the help of Paul Webster, Cole Van Vlack, and Matthew Marsh from Laser Depth Dynamics. As specialists in the field, they were invaluable intellectual resources who did not hesitate to answer any of my inquiries. They have also been generous in allowing my colleagues and me to use their industry-grade ICI equipment.

I have also received a considerable amount of support from my mother, Lynn Jenkins, who is an expert in the English language and has helped me develop my professional writing skills throughout my academic career.

Finally, this project could not have been undertaken without financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada Foundation for Innovation (CFI), the Government of Ontario, and the Queen's University Department of Physics, Engineering Physics, and Astronomy.

# Table of Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iii</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Abbreviations</b>	<b>ix</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Background</b>	<b>6</b>
2.1 Inline Coherent Imaging . . . . .	6
2.2 Field-Programmable Gate Arrays . . . . .	12
2.3 Feedback Control . . . . .	18
2.4 Laser Keyhole Welding . . . . .	22
<b>Chapter 3 System Design</b>	<b>27</b>
3.1 Laser Processing Station . . . . .	27
3.2 ICI Signal Processing . . . . .	31
3.2.1 DC Subtraction . . . . .	31
3.2.2 Resampling . . . . .	32
3.2.3 Depth Recovery via FFT . . . . .	37

3.3	LabVIEW FPGA Processing Algorithms . . . . .	39
3.3.1	Initialization and Calibration Arrays . . . . .	42
3.3.2	Camera Reading Loop . . . . .	45
3.3.3	Pixel Unbundling Loop . . . . .	46
3.3.4	Pixel Skipping and Repeating Loops . . . . .	46
3.3.5	Interpolation Loop . . . . .	49
3.3.6	Pixel Rebundling Loop . . . . .	49
3.3.7	FFT and Depth Tracking Loop . . . . .	51
3.3.8	Feedback and DAC Loops . . . . .	52
3.4	Feedback Signal Hardware . . . . .	53
3.4.1	DAC Circuit . . . . .	54
3.4.2	Amplifier Circuit . . . . .	55
<b>Chapter 4</b>	<b>Application and Results</b>	<b>59</b>
4.1	Latency Assessment . . . . .	59
4.2	Autofocus . . . . .	62
4.3	Welding Depth Control . . . . .	69
<b>Chapter 5</b>	<b>Conclusion and Future Work</b>	<b>76</b>
5.1	Resampling and FT Methods . . . . .	76
5.2	ASIC Design . . . . .	78
5.3	Other Laser Manufacturing Processes . . . . .	79
<b>Appendix A</b>	<b>MATLAB Code</b>	<b>92</b>

# List of Tables

2.1	Truth table for an AND logic gate. . . . .	13
3.1	List of resource usage specifications, timing limitations, and compilation time for the most recent iteration of the compiled FPGA ICI code (as of July 16, 2015). . . .	40

# List of Figures

1.1	Comparison of HDL to LabVIEW FPGA code. . . . .	5
2.1	Michelson Interferometer diagram. . . . .	7
2.2	Programmable Logic Array (PLA) diagram. . . . .	14
2.3	Programmable Array Logic (PAL) diagram. . . . .	15
2.4	Complex Programmable Logic Device (CPLD) diagram. . . . .	16
2.5	Field-Programmable Gate Array (FPGA) diagram. . . . .	17
2.6	Basic structure of a general feedback control system. . . . .	19
2.7	Schematic showing the structure of a PID controller. . . . .	21
2.8	Step response of a feedback system. . . . .	23
2.9	Keyhole welding diagram. . . . .	24
2.10	Four common laser welding geometries. . . . .	25
3.1	Photograph of laser beam delivery head. . . . .	28
3.2	Diagram showing essential components of the ICI apparatus. . . . .	29
3.3	ICI signal DC spectrum. . . . .	32
3.4	Sample ICI spectrum. . . . .	32
3.5	Sample spectrum with DC subtracted. . . . .	33
3.6	Visualization of skipped and repeated pixels for linear interpolation. . . . .	34
3.7	Sample index shift array. . . . .	36
3.8	Sample interpolation coefficient array. . . . .	36
3.9	Sample spectrum after being resampled. . . . .	37
3.10	Comparison of resampled and raw spectral peak widths. . . . .	38

3.11	General code structure flowchart. . . . .	43
3.12	Index shift calculation code. . . . .	44
3.13	Interpolation array calculation code. . . . .	44
3.14	NoC array calculation code. . . . .	44
3.15	Camera pixel reading code. . . . .	45
3.16	Example of NoC array. . . . .	47
3.17	Linear interpolation code. . . . .	50
3.18	FFT preparation code. . . . .	51
3.19	DAC box wiring diagram. . . . .	55
3.20	Op amp box wiring diagram. . . . .	57
3.21	Op amp box step response. . . . .	58
4.1	Direct latency measurement of PC ICI processing. . . . .	60
4.2	Direct latency measurement of FPGA ICI processing. . . . .	61
4.3	Autofocus feedback loop block diagram. . . . .	64
4.4	Settling time and overshoot while tuning $K_i$ gain. . . . .	65
4.5	Settling time and overshoot while tuning $K_{i2}$ . . . . .	66
4.6	Outline of inclined plate autofocus scan. . . . .	67
4.7	Autofocus RMS following errors while for various $K_{i2}$ values. . . . .	68
4.8	Autofocus RMS following errors while for various deadband values. . . . .	68
4.9	Autofocus settling times with extra latency introduced. . . . .	69
4.10	Welding depth control to various target depths. . . . .	72
4.11	Depth measurements during welding control over a step interface. . . . .	73
4.12	Laser power during welding control over a step interface. . . . .	74
A.1	MATLAB implementation of the index shift array generation code. . . . .	92
A.2	MATLAB implementation of the interpolation array generation code. . . . .	92
A.3	MATLAB implementation of the NoC array generation code. . . . .	93

A.4 MATLAB implementation of the interpolation calculation. . . . . 94

# List of Abbreviations

ASIC	Application-Specific Integrated Circuit
ATX	Advanced Technology Exchange
BNC	Bayonet Neill-Concelman (Connector)
CL	Camera Link
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CW	Continuous-Wave
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
FD-OCT	Fourier-Domain Optical Coherence Tomography
FFT	Fast Fourier Transform
FIFO	First-In First-Out (Buffer)
FPGA	Field-Programmable Gate Array
FWHM	Full Width at Half-Maximum
GPU	Graphical Processing Unit
HDL	Hardware Description Language
ICI	Inline Coherent Imaging
I/O	Input/Output

IR	Infrared
NoC	Number of Occurrence
NUFFT	Non-Uniform Fast Fourier Transform
OCT	Optical Coherence Tomography
Op Amp	Operational Amplifier
PLA	Programmable Logic Array
PLD	Programmable Logic Device
PC	Personal Computer
PID	Proportional-Integral-Differential
SCTL	Single-Cycle Timed Loop
SD-OCT	Spectral-Domain Optical Coherence Tomography
SLD	Superluminescent Diode
TD-OCT	Time-Domain Optical Coherence Tomography
VI	Virtual Instrument

# Chapter 1 Introduction

When the first laser was invented in 1960, it was not evident that it would have any useful applications; however, over the last 50 years, lasers have become standard components of technology. Lasers see widespread use today, ranging from common consumer electronics to the most cutting edge science. Lasers are also used in industry as a tool for manufacturing processes such as welding[1], cutting[2], and drilling[3]. In the field of medicine, lasers, by way of bone ablation[4] for example, have the potential to replace surgical tools. Furthermore, by controlling industrial CO<sub>2</sub> and fiber lasers with computers and robotics, an unmatched level of precision and consistency can be achieved. In addition, lasers do not suffer from the type of tool wear that is present with traditional contact-based tools.

With the rapidly rising popularity of lasers as machining and surgical tools, there is increasing demand for quality assurance and feedback control, the type of which is dependent on the application. For example, in factories where laser beam keyhole welding is performed, it is common practice to calibrate the laser parameters at regular intervals by performing a test weld and destroying the part in order to obtain a cross-section of the welded area to verify that the system is functioning. Not only is such a practice wasteful, but it does not guarantee the integrity of every part on the line as the performance of the welding system may be degraded by external factors after the machine has been calibrated. The ability to precisely monitor and control the penetration depth of the laser beam in real-time would allow for manufacturers to be assured that every weld is successful without having to destroy a single part. Other laser machining processes that could benefit from monitoring and control include beam autofocus for cutting, bone breakthrough detection for neurosurgery, and rapid pulse modulation in ultrafast ablation.

Today, there is still a general lack of established quality assurance and closed-loop feedback

control methods in the field of laser manufacturing[5]. Industrial laser beam control is an active field of research with a number of techniques gaining popularity in both academia and industry. While it is relatively simple to control processing parameters such as laser power, scanning speed, and pulse rate to change the performance of the machining process, the difficulty in creating a closed-loop feedback control system arises from the lack of established real-time sensing methods. The tracking of the location of the physical tip of a mechanical tool is an uncomplicated task as the length does not change. However, it is a less trivial task to track the position of a laser beam, since it travels into the material through a very narrow path (on the order of the beam's spot size, which is typically hundreds of microns), and the beam length changes rapidly. A number of solutions have been proposed in research for sensing methods that can achieve control of industrial laser machining. One such method involves examining the "full penetration hole" using a camera to measure whether or not the laser beam is fully penetrating through the bottom material[5]. Other efforts focus on frequency analysis of the light that is emitted from the plasma where the laser beam is interacting with the material surface[6, 7]. Infrared cameras are also used to perform thermal analysis of the molten pool during laser processes[8].

While these feedback sensing methods have been shown to be effective, they all measure properties that are correlated to the laser penetration depth. Therefore, there is a need for a method that can perform a direct, inline measurement of the beam depth itself. Inline Coherent Imaging (ICI) is a recently developed solution that addresses this need. ICI is based on a medical imaging technique called Optical Coherence Tomography (OCT). OCT was invented in 1991 as a non-invasive imaging modality and is especially applicable in retinal imaging[9]. A common form of OCT, Fourier-Domain OCT (FD-OCT), uses a broadband light source in a Michelson Interferometer setup to measure the optical path length of a focused beam. FD-OCT is the scheme from which ICI is adapted, wherein the imaging beam propagates coaxially with the machining beam. By measuring the optical path length of a light beam that follows the same path as the machining beam, we effectively recover a direct and inline measurement of the distance the machining beam travels, and by extension, the depth that it is penetrating into the material. ICI provides measurements of

the beam penetration depth at rates up to 312 kHz (limited by the maximum acquisition rate of the spectrometer camera). It has been tested and applied for a number of industrial and medical purposes and is commercially available on the market.

ICI data is typically processed on a standard personal computer (PC) running the Windows operating system. Modern PCs contain powerful Central Processing Units (CPUs) and Graphical Processing Units (GPUs) that are able to efficiently process large amounts of ICI data simultaneously. However, for real-time purposes wherein it is desirable to process individual ICI frames as quickly as possible, the performance of PCs can be unreliable. Furthermore, a relatively large amount of latency (the difference in time between the input and output signals of the system) is introduced. Application-specific integrated circuits (ASICs) are capable of processing data rapidly; however, they are expensive to design and cannot be modified once manufactured. A solution that offers the performance advantage of ASICs while preserving the software reconfigurability of PCs is a Field-Programmable Gate Array (FPGA). The FPGA was invented by Xilinx Inc. in 1985 as a successor to existing programmable logic devices that were based on hardwired logic gates. FPGAs consist of up to millions of logic gates implemented using look-up tables (LUTs) and typically consist of dedicated digital signal processing (DSP) slices designed to efficiently perform specific complex tasks such as multiplication. They also contain block read only memories (BRAMs) for temporary storage and transfer of data through buffers.

FPGAs are typically programmed using a hardware description language (HDL); however National Instruments Corp., in partnership with Xilinx Inc., has recently developed FPGA units that are programmed using their proprietary graphical programming language LabVIEW. This allows FPGA algorithms to be developed with a high level of abstraction, making the technology accessible to researchers and engineers that do not possess any expertise in HDL programming. An example of a comparison of HDL code to LabVIEW FPGA code is shown in figure 1.1.

In this thesis, I will present an implementation of ICI processing using a LabVIEW FPGA module. In chapter 2, I will present an overview of the background information upon which my design is based. I will start with a description of the basic theory of OCT and ICI. I will then

discuss programmable logic devices and the evolution of the FPGA. Next, I will summarize the basic concepts of feedback control, with a focus on the type of PID control that was used for the experiments described in chapter 4. Finally, I will briefly discuss laser keyhole welding and the various geometries that are used in industry for research and production purposes.

In chapter 3, I will explain the design of the FPGA-based ICI system that I developed. I will start with a brief description of the laser processing station used to perform experiments in our lab. Next is a general description of the signal processing steps involved in extracting the depth information from an ICI signal, followed by a detailed explanation of the programmatic implementation of the signal processing steps with examples of important portions of the LabVIEW FPGA code. I will then end this chapter with a description of a few of the circuits that I built in order to interface the FPGA with existing lab equipment so that feedback signals could efficiently be transferred between them.

Chapter 4 will provide an overview of the experiments I performed to test my FPGA ICI system. I will start with a comparison of direct latency measurements between PC and FPGA-based signal processing, and show that the FPGA achieves order-of-magnitude lower latency and improved determinism over the PC. I will then present an autofocus system, wherein the beam delivery head for the processing laser is kept at a fixed height above the target in the beam path using FPGA-based feedback control, such that the processing beam can be kept in focus. Finally, I describe real-time keyhole welding depth control, in which the FPGA ICI system is used to modulate the power of the processing laser so that a constant keyhole depth is maintained.

In the final chapter, I will touch on several topics for possible future exploration, namely alternate algorithms for the interpolation and/or FFT calculations, the possible future use of application-specific integrated circuit hardware in place of an FPGA, and additional laser processing applications that could benefit from an FPGA-based ICI control system.

# Simple Counter Functionality

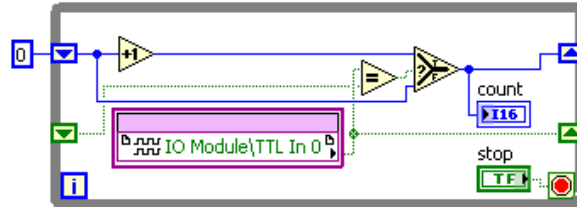
VHDL
LabVIEW

```

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

entity COUNTS is
port(
    DIN : in  std_logic_vector(7 downto 0);
    CLK : in  std_logic;
    LOAD : in  std_logic;
    DOUT : out std_logic_vector(7 downto 0)
);
end COUNTS;

architecture behavior of COUNTS is
begin
    -- notice the process statement and the variable COUNT
    clk_proc:process(CLK)
    variable COUNT:unsigned(7 downto 0) := "00000000";
    begin
        if (CLK'EVENT AND CLK = '1') then
            if LOAD = '1' then
                COUNT := DIN;
            else COUNT := COUNT + 1;
            end if;
        end if;
        DOUT <= COUNT after 50 ns;
    end process clk_proc;
end architecture behavior of COUNTS;
    
```

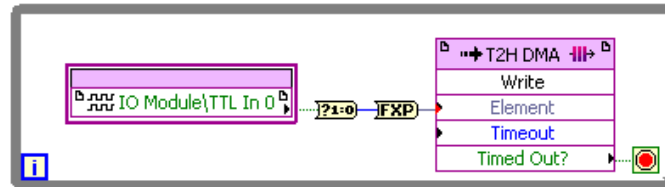


# I/O with Direct Memory Access Transfer

VHDL
LabVIEW



66 pages, ~4000 lines



**Figure 1.1:** Comparison of HDL code complexity to that of the equivalent LabVIEW FPGA implementation. Reproduced from [10] using code from [11].

# Chapter 2 Background

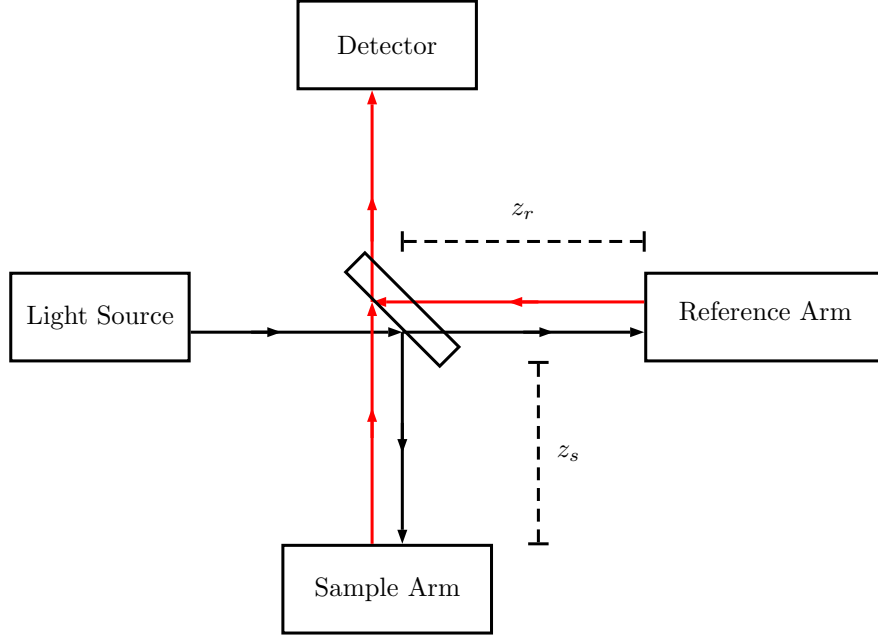
## 2.1 Inline Coherent Imaging

ICI is an industrial interferometric imaging technique that was invented at Queen's University[12]. While ICI is novel technology that solves new and important problems in the manufacturing industry, it borrows heavily from a medical imaging modality called Optical Coherence Tomography (OCT). OCT was invented in 1991 as a non-invasive method of imaging biological systems using light as an alternative to existing dangerous and low-resolution methods such as x-ray CT and MRI[13]. The first implementation of OCT used an 830 nm superluminescent diode (SLD) as a light source coupled to a Michelson interferometer[9]. The interferometer consisted of a 50/50 beam splitter that split the beam into a sample arm and a reference arm. The light was reflected back by both arms into the splitter, which coupled the two beams back together and transmitted the interfered light back to a photodetector. This basic scheme is shown in figure 2.1.

In this section, I will follow the derivations for OCT theory as presented by Brezinski[14] and will arrive at the equation that describes the interference pattern that is read by the spectrometer in ICI and processed using the algorithms in chapter 3. I will start by expressing the electric field of the light emitted by the SLD as a monochromatic plane wave:

$$E_{\text{SLD}} = E_0 e^{-ikz} \quad (2.1)$$

where  $k = 2\pi/\lambda$  is the wavenumber related to the frequency of the light source and  $E_0 = A e^{i\omega t}$  contains the amplitude and time-oscillation of the electric field. The 50/50 beam splitter transmits half of this beam's power and reflects the other half, such that the amplitude of the electric fields entering the reference and sample arms have amplitudes that have been reduced by a factor of



**Figure 2.1:** Basic Michelson interferometer scheme used for ICI and OCT with transmitted light shown in black and reflected light shown in red.

$1/\sqrt{2}$ . Additionally, the reflected beam (which we represent as the sample arm in figure 2.1) undergoes a  $180^\circ$  phase shift due to the reflection. We can therefore express the electric fields entering the reference and sample arms as:

$$E_r = \frac{1}{\sqrt{2}} E_{\text{SLD}} \quad (2.2)$$

$$E_s = \frac{-1}{\sqrt{2}} E_{\text{SLD}} \quad (2.3)$$

where the negative sign in the sample arm is representative of the  $180^\circ$ . The two beams propagate through free space of lengths  $z_R$  and  $z_S$  and are reflected back into the beam splitter. We can express the reflectivity in the two arms as  $r_r$  and  $r_s$  and then write the expressions for the electric fields of

the reflected beams when they arrive at the beam splitter as:

$$E'_r = \frac{1}{\sqrt{2}} r_r E_{\text{SLD}} e^{-i2kz_r} \quad (2.4)$$

$$E'_s = \frac{-1}{\sqrt{2}} r_s E_{\text{SLD}} e^{-i2kz_s} \quad (2.5)$$

where the factor of two in each exponent term arises from the fact that each beam travels a round trip through the arm. The beam splitter reflects the reference arm beam and transmits the sample arm beam into the detector arm. We can write the electric field of the beam that travels to the detector as follows:

$$E_D = \frac{1}{\sqrt{2}} E'_r + \frac{1}{\sqrt{2}} E'_s \quad (2.6)$$

$$= E_R e^{-i2kz_r} - E_S e^{-i2kz_s} \quad (2.7)$$

where I have written  $E_R = \frac{1}{2} r_r E_{\text{SLD}}$  and  $E_S = \frac{1}{2} r_s E_{\text{SLD}}$  for simplicity. Since the detector measures intensity and not electric field, we can express the signal that is measured by the detector as:

$$I_D \propto |E_D|^2 = E_D E_D^* \quad (2.8)$$

$$= E_R E_R^* + E_S E_S^* - E_R E_S^* e^{-i2k\Delta z} - E_S E_R^* e^{i2k\Delta z} \quad (2.9)$$

$$\propto I_R + I_S - \sqrt{I_R I_S} \left( e^{i2k\Delta z} + e^{-i2k\Delta z} \right) \quad (2.10)$$

where  $\Delta z = z_r - z_s$  is the difference between the reference and sample arm lengths. In the above equation, I have ignored the phase shift that might arise from the complex nature of the reflectivity terms, since it is only the frequency of the interference fringes that are of interest in recovering OCT data. Simplifying the above equation using Euler's identity, we can write:

$$I_D \propto I_R + I_S - 2\sqrt{I_R I_S} \cos(2k\Delta z) \quad (2.11)$$

The first two terms of this equation are DC intensities and are not of interest in recovering OCT images (in fact, they are typically subtracted from the measured signal before any data processing occurs). The third term contains a cosine term that represents the interference of the light from the two arms. This interference term is dependent on three quantities: the wavenumber of the light source ( $k$ ) and the lengths of the reference and sample arms ( $z_r$  and  $z_s$ ).  $k$  and  $z_r$  are known quantities that are controlled in the experimental setup and  $z_s$ , the location of the reflecting interface in the sample arm, is the unknown quantity that we are trying to recover. In the original time-domain OCT scheme, the reference arm length is varied in order to “sweep” across various values of  $\Delta z$  to obtain an interference spectrum in the  $z$ -domain.

While time-domain OCT is able to produce high-quality images with a large field of view, it suffers from the drawbacks associated with the requirement of having a moving reference arm. As such, the image acquisition rate is limited by the mechanical motion of the reflector. Additionally, the presence of a rapidly moving reflector in the system design introduces an extra level of complexity and potential for malfunction. To improve upon these features of the OCT design, Fourier-domain OCT (FD-OCT) was developed in 1998[15]. FD-OCT uses essentially the same setup, but with the reference arm held fixed. This eliminates the need for any moving parts, and research also suggests that there are improvements in sensitivity for high-speed imaging applications in FD-OCT when compared to TD-OCT[16, 17]. In FD-OCT,  $\Delta z$  is fixed and the photodiode is replaced with a spectrometer, such that the interference pattern is resolved in the  $k$ -domain. If we consider the light to consist of an infinitely broad spectrum, we can express the intensity distribution collected in the spectrometer in  $k$ -space as:

$$I_D(k) \propto I_R + I_S - 2\sqrt{I_R I_S} \cos(2k\Delta z) \quad (2.12)$$

Calculating the analytical Fourier transform of equation (2.12) gives us the following expression:

$$\tilde{I}_D(z) \propto \sqrt{2\pi} \delta(z) (I_R + I_S) - \sqrt{2\pi I_R I_S} [\delta(z - 2\Delta z) + \delta(z + 2\Delta z)] \quad (2.13)$$

where  $\delta(z)$  is the Dirac delta function. We can see that, for the idealized case in which we only treat a single interface, and the light source has an infinitely broad spectrum, we obtain delta functions at  $\pm\Delta z$ . Calculating the Fourier transform of equation (2.12) therefore allows us to obtain a direct measurement of the difference between the reference and sample arms. Since the reference arm is fixed in FD-OCT, this is effectively a measurement of the location of the interface in the sample arm.

In a real OCT system, the light source does not emit light with an infinitely broad spectrum. The emitted light has a finite bandwidth associated with it, which is approximately 100 nm centered at 840 nm for the SLD used in our ICI system. We can treat this mathematically as an envelope function  $G(k)$ :

$$I_D(k) \propto G(k) \left[ I_R + I_S - 2\sqrt{I_R I_S} \cos(2k\Delta z) \right] \quad (2.14)$$

and when we calculate the Fourier transform we obtain:

$$\tilde{I}_D(z) \propto \tilde{G}(z) * \left( \sqrt{2\pi} \delta(z) (I_R + I_S) - \sqrt{2\pi I_R I_S} [\delta(z - 2\Delta z) + \delta(z + 2\Delta z)] \right) \quad (2.15)$$

where  $*$  denotes convolution. In practice,  $G(k)$  is typically approximated using a Gaussian function, and the peaks caused by the interface therefore take the same shape (since they are delta functions convolved with a Gaussian).

Another approximation that is implicit in this derivation is that there is only a single reflecting interface in the sample arm. This is not the case in most applications of OCT, and it is therefore common practice to write the interference term in equation (2.14) as a sum representing the cross-correlations between each of the sample arm reflections with the reference arm, as well as autocorrelation terms between the various sample arm reflections. While it is important to track each of these interfaces (and their relative amplitudes) for OCT imaging, in ICI we are typically interested in just the single, brightest interface (which could be the surface of the part or the bottom of a weld keyhole). Therefore, for the ICI feedback control presented in this thesis, the single-reflector approximation that is inherent in equation (2.14) is reasonable.

Note also that there is a peak at zero with an amplitude that scales with the intensities  $I_R$  and  $I_S$ . These two intensities are independent of the reference and sample arm lengths, and are therefore referred to as “DC components.” It is common practice in OCT and ICI to subtract these components from the interference spectrum whenever possible so that the amplitude of the DC peak in the Fourier transform is minimized.

Axial resolution of an OCT system can be defined theoretically by the following expression:

$$\delta z = \frac{2 \ln(2) \lambda_0^2}{\pi \Delta \lambda} \quad (2.16)$$

where  $\Delta \lambda$  is the wavelength bandwidth of the light source and  $\lambda_0$  is the center wavelength. As mentioned above, the LDD ICI system uses an SLD with a center wavelength of 840 nm and a bandwidth of  $\sim 100$  nm, so the theoretical axial resolution calculated using equation (2.16) is  $3.11 \mu\text{m}$ .

The form of FD-OCT described above is only one form of spectral OCT. Another common method known as “swept-source” OCT exists, wherein the light source has a very narrow linewidth that is rapidly swept over a range of interest. A single fast photodetector is synchronized with the sweeping of the light source so that the spectrum is encoded in the time-domain but is resolved in the  $k$ -domain[18]. While swept-source has been shown to produce higher-quality imagery than traditional SD-OCT[19, 20], it requires more complex light sources[21], resulting in significantly higher capital costs.

As a final note, it is important to mention that the  $z$  terms in the above equations represent optical path lengths rather than actual spacial propagation distances. The optical path length is given by  $z = nx$  where  $n$  is the index of refraction of the material through which the beam is propagating. In theory, this could cause distortions in the signal due to unmatched indices in the reference and sample arms, so it is common practice to introduce additional glass in the reference arm to achieve index matching with the focusing optics in the beam delivery head in the sample arm. While additional index variations caused by effects such as the vapor and plasma in the

machining area could also cause signal distortions, these effects are not large enough to have a significant impact on the accuracy or resolution of the ICI system within the context of the applications discussed in this thesis[22].

## **2.2 Field-Programmable Gate Arrays**

At the most fundamental level, all digital electronic devices can be broken down into complex networks of logic gates. While they are the driving factor behind all of our modern day devices, the concept of using switching devices to implement binary logic has been around for many decades. The use of electronic switches can be traced back to the 1835 invention of the electromechanical relay[23], the use of which was popularized by Samuel Morse's famous invention which used the relays to transmit morse-code signals over long distances. Electromagnetic relay computers followed, reaching 5 tons and 16 m long with over 750,000 relays at their peak[24]. The invention of the vacuum tube allowed for faster and more compact devices to be built, with the technology being pioneered by military scientists for cryptographic applications in the 1940s[25]. The 1950s marked the origin of the transistor computers that we are familiar with today[26].

Digital logic is a term that describes networks of logic gates that output a number of binary output signals based on a set of binary inputs. Logic is typically made up of the six basic logic gates: NOT, AND, OR, NAND, NOR, and XOR. Each of these gates accepts two inputs (with the exception of the NOT gate, which takes one) and outputs a single output that is dependent on the values of the inputs. Logic gates can be easily described using truth tables, which are tables reflecting the output value that corresponds to every possible combination of inputs. An example of the truth table for an AND gate, which outputs a 1 only when both of the input values are 1, is shown in table 2.1. While the logic implementations of these individual gates are relatively simple, they are capable of performing highly complex computations when millions of gates are interconnected.

Although networks of logic gates have impressive computational abilities, it is a tedious and

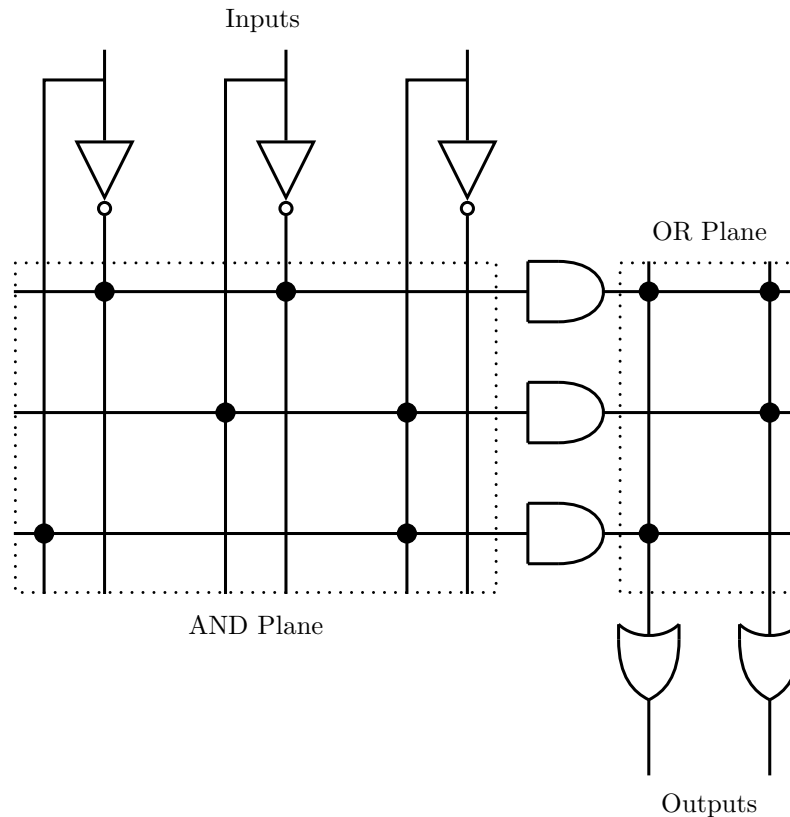
Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

**Table 2.1:** Truth table for an AND logic gate.

sometimes impossible task to design and construct circuits that connect millions of transistors on a single chip. While modern day circuit manufacturing techniques such as printed circuit boards (PCBs) offer highly automated methods of constructing these boards, they still suffer from the downside of being non-reconfigurable. Once a hardwired digital circuit has been built, it is impossible to change it if a bug is found in the design, the application needs change, or an upgrade to the system is required. In these cases, a completely new circuit must be designed and created and the old chip is thrown away. Programmable Logic Devices (PLDs) are a class of devices that became popular in the 1970s to address these issues. The first of these devices is the Programmable Logic Array (PLA)[27]. The PLA consists of two planes, one containing AND gates and the other containing OR gates.

A diagram depicting the general structure of the PLA is shown in figure 2.2. The input signals are inverted and both the inverted and original values are fed into the AND plane. In the AND plane, connections are made according to a pre-designed program, and the connected signals are wired to the AND gates. The outputs of the AND gates serve as the inputs to the OR gate plane, where another set of programmed connections determine how the signals are passed through the OR gates. PLAs were programmed using electronic programming hardware typically provided by the device manufacturer.

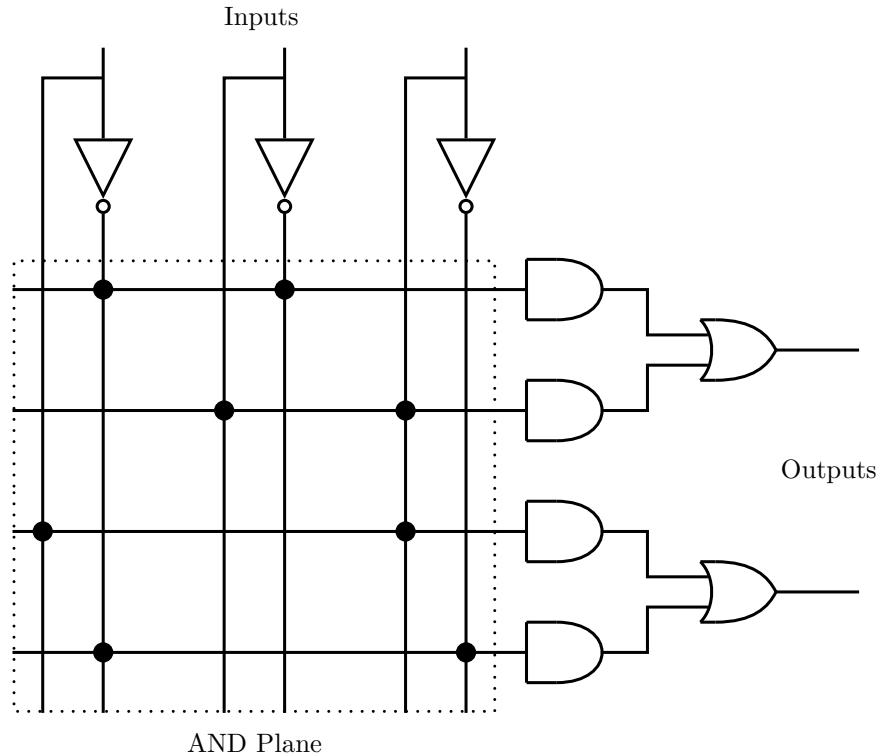
A modified version of the PLA called the Programmable Array Logic (PAL) was invented in 1976[29]. The PAL is very similar in design to the PLA; however, the OR gate plane is replaced by a series of OR gates that are hardwired to the outputs of the AND gate plane. Since only one plane has to be programmed, PALs offer improved price and performance and reduced programming



**Figure 2.2:** Diagram of a Programmable Logic Array. Nodes marked with dots represent connections that have been programmed. Modified from [28].

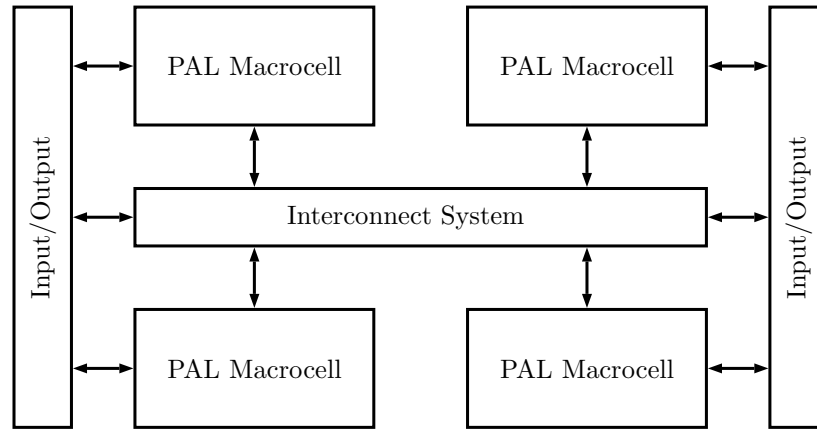
complexity, but at the cost of reduced flexibility in the design. A diagram depicting the PAL design is shown in figure 2.3.

As circuit complexity increased, the capability of PLAs and PALs to scale in order to meet the demands of larger circuit designs started to diminish. While PALs were able to meet the demands of designs with up to about 32 inputs and outputs, they were not able to accommodate larger projects. To meet these demands, the Complex Programmable Logic Device (CPLD) was introduced in the 1980s. The CPLD effectively consists of multiple PAL-like blocks known as “macrocells” that surround a reprogrammable interconnect region in the center[28]. In addition to the advantage in size over PLA and PAL logic devices, CPLDs also offer a minor degree of reprogrammability. While the macrocells are made of hardwired PAL and PLD components, the interconnects are fully reprogrammable. A diagram outlining this structure is shown in figure 2.4.



**Figure 2.3:** Diagram of a Programmable Array Logic block. Nodes marked with dots represent connections that have been programmed. Modified from [28].

Field Programmable Gate Arrays (FPGAs) were invented by Ross Freeman of Xilinx, Inc. in 1985[30]. They were developed alongside the CPLD and share similarities in structure. Like CPLDs, FPGAs consist of blocks of logic that can be programmed using interconnection switches. The difference comes from the type of logic that is used within the blocks. Unlike CPLDs, which use PAL and PLA macrocells that consist of AND, OR, and XOR gates, FPGAs are typically implemented using lookup tables (LUTs). FPGA logic blocks are organized in a 2D array and connected using programmable interconnect switches, such that they are completely reprogrammable. This allows for FPGAs to be reprogrammed by the end user in the field rather than by the chip manufacturer, making them more versatile than CPLDs. The additional complexity and flexibility of FPGAs make them a higher-cost and performance alternative to CPLDs[31]. Because of their implementation scheme, FPGAs are able to scale to much larger sizes than CPLDs. While CPLDs can be built with an “equivalent gate” number of around 10,000, FPGAs can offer millions of



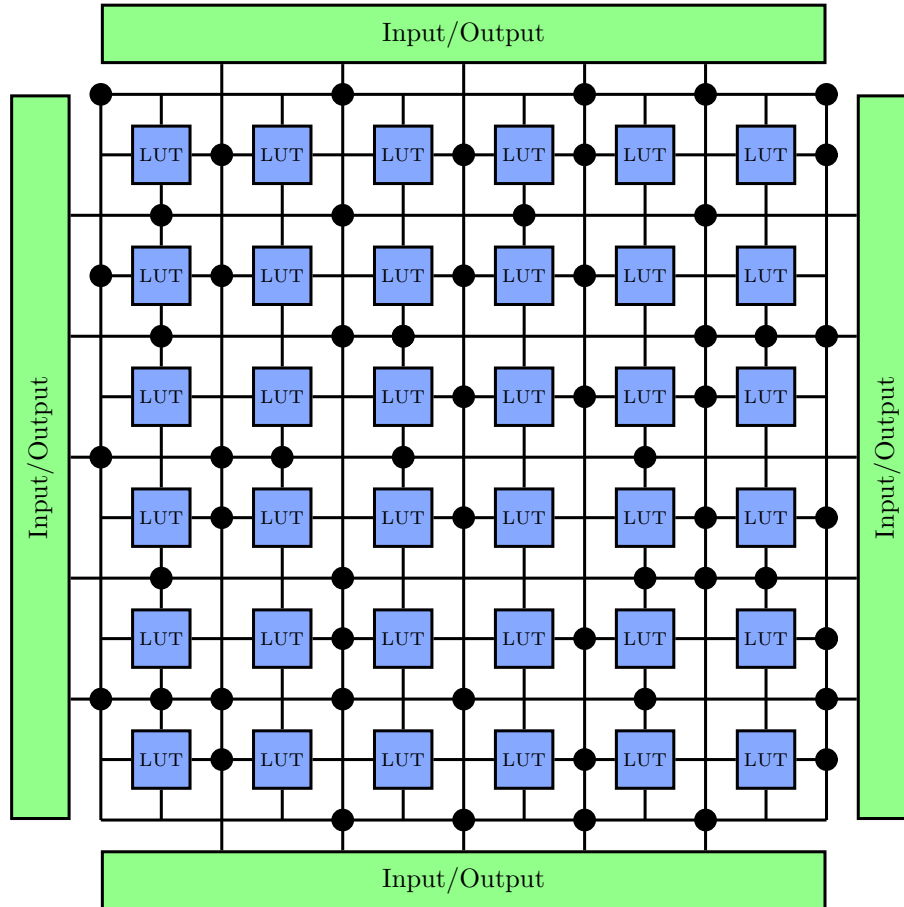
**Figure 2.4:** Diagram showing the srchitecture of a Complex Programmable Logic Device. Modified from [28].

equivalent gates[28]. The FPGA architecture is shown in figure 2.5.

Because both the logic blocks and the interconnection system are reprogrammable, and the number of gates scales to the millions, FPGAs are a popular tool for developing and prototyping digital signal processing systems. Additionally, the ability to add additional processing hardware such as block RAM and DSP slices makes them even more appealing for general use. The development of FPGA technology over the last few decades has also pushed prices down, making them more affordable for a wider market.

In industrial settings, FPGAs can be used to replace computers that control tasks on production lines[32, 33]. The advantage of FPGAs over PCs is seen in applications where signal processing is required at high throughput rates with low latency (where latency is defined as the time difference between the input and output of the signal). PCs can be unreliable due to system interrupts that are prioritized above the user's code because they are essential for the performance of the operating system. The dedicated processing hardware scheme used by FPGAs offers a much higher level of determinism. Research has also shown that FPGAs can achieve high throughputs in processing high-definition OCT images for live-viewing applications[34, 35] and are more capable than GPUs of scaling to meet the high-volume demands of OCT in the future[36].

Traditionally, FPGAs are programmed using a hardware description language (HDL). The most



**Figure 2.5:** Diagram of the FPGA's architecture. The blue boxes represent lookup tables, the wires and dots represent the reprogrammable interconnect switches, and the green boxes represent the I/O blocks. Modified from [28].

common HDLs are low-level languages such as Verilog and VHDL, which have existed since the 1980s[37]. While the use of HDLs makes FPGAs more available for mainstream use, the abstraction level of these languages introduces a steep learning curve such that a certain level of expertise in the field of FPGA programming is required to make effective use of them. Efforts have been made to develop FPGA programming capabilities for higher-level programming languages[38, 39]; however, these are typically not well-developed and result in implementations that make inefficient use of FPGA resources. More recently, Xilinx Inc. partnered with National Instruments Corp. to develop FPGA systems that are programmed using NI's LabVIEW development environment. LabVIEW is a graphical programming language that allows the user to develop their code in an

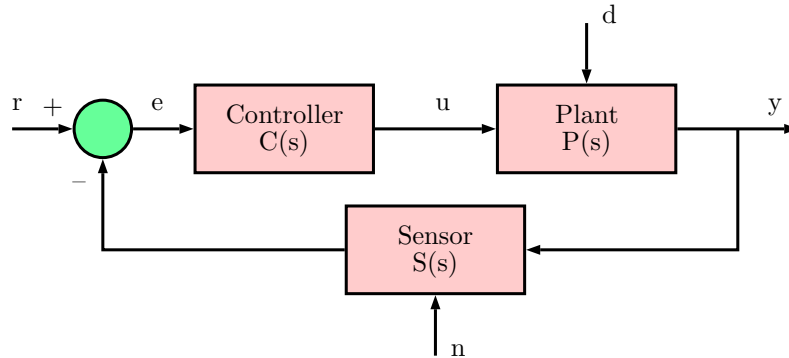
intuitive manner without requiring any knowledge of HDLs. It is a popular language that many engineers and scientists are already familiar with. As such, LabVIEW makes FPGA development more accessible to non-specialists, and it is a LabVIEW FPGA system that was used to develop the FPGA ICI system demonstrated in this thesis.

## 2.3 Feedback Control

Feedback control is, in a general sense, any system in which a sensor causes a response to achieve a desired effect (typically maintaining a set value with a high level of stability) within one or more of its properties. The mathematics of modern-day control theory can be traced back to 1868 when an attempt was made to develop a formalism based on linearization of differential equations of motion to stabilize a centrifugal governor system for steam engines[40]. Since then, control theory has been developed extensively and is used in many fields. Examples include relatively simple systems such as the thermostat that controls the temperature in a home as well as complex systems such as quadcopters[41, 42], missile guidance[43], and attitude control in satellites[44].

A basic feedback control loop is generally broken down into three components: the plant, the controller, and the sensor[45]. The plant is the parameter of the system that is to be controlled, which in the applications discussed in chapter 4, is either the height of the beam delivery head or the power of the processing laser. The sensor is the device that monitors part of the system for deviations from the expected behavior. In ICI feedback control, the sensor is made up of an interferometer, a spectrometer, and a data processing device such as an FPGA. Finally, the controller is the device that reads values from the sensor, makes decisions about adjustments that need to be made to the system, and instructs the plant to change accordingly. In chapter 4, the controller is integrated in the motion control suite for autofocus and is incorporated into the FPGA design for welding depth control. A diagram showing the general structure of a feedback system is shown in figure 2.6. A setpoint value  $r$  is entered into the controller so that it knows what target value it is trying to achieve. A disturbance  $d$  enters the plant causing the state of the plant (which

is measured by the sensor) to deviate from this setpoint. The controller assesses the error signal  $e$ , which is calculated by taking the difference between the setpoint and the plant output  $y$ , and makes a decision about how to control the plant.



**Figure 2.6:** Basic structure of a general feedback control system. The setpoint (or target value) of the plant is denoted by  $r$ , the output of the plant (and hence the value measured by the sensor) is  $y$ , the error signal is denoted  $e$ , the controller output is denoted  $u$ , the noise in the sensor is denoted by  $n$ , and the disturbance to the plant is denoted  $d$ . Figure created by author using concepts from [46].

Each of the three components in figure 2.6 has an effect on the signal going into it that manifests itself in the output signal. The controller has an effect on the signal if the error  $e(t)$  has a non-zero value, which indicates that the system is not in the desired state. The plant is affected by an external disturbance  $d(t)$  to the system and the controller value  $u(t)$ . Finally, the sensor is exposed to noise  $n(t)$  that is present in the environment and in the sensor's hardware. Mathematically, we can express these effects using their transfer functions. Transfer functions of linear systems are defined by the ratio of the Laplace transforms of the output and input signals of the system. This model for linear systems was used as an approximation for the systems used to perform the experiments in the following sections. Noting that the error signal  $e(t)$  is given by the difference between the setpoint  $r(t)$  and the sensor value, we can express the outputs of the three components

in 2.6 in Laplace space in terms of their transfer functions:

$$U(s) = C(s)E(s) \quad (2.17)$$

$$Y(s) = P(s)(U(s) + D(s)) \quad (2.18)$$

$$E(s) = R(s) - S(s)(Y(s) + N(s)) \quad (2.19)$$

By combining and rearranging the above equations, we can write the system output  $Y(s)$  in terms of the setpoint  $R(s)$  (the contributions from the external disturbance and noise are small in many cases, so I have excluded them for simplicity):

$$Y(s) = \frac{P(s)C(s)}{1 + P(s)C(s)S(s)}R(s) \quad (2.20)$$

$$= H(s)R(s) \quad (2.21)$$

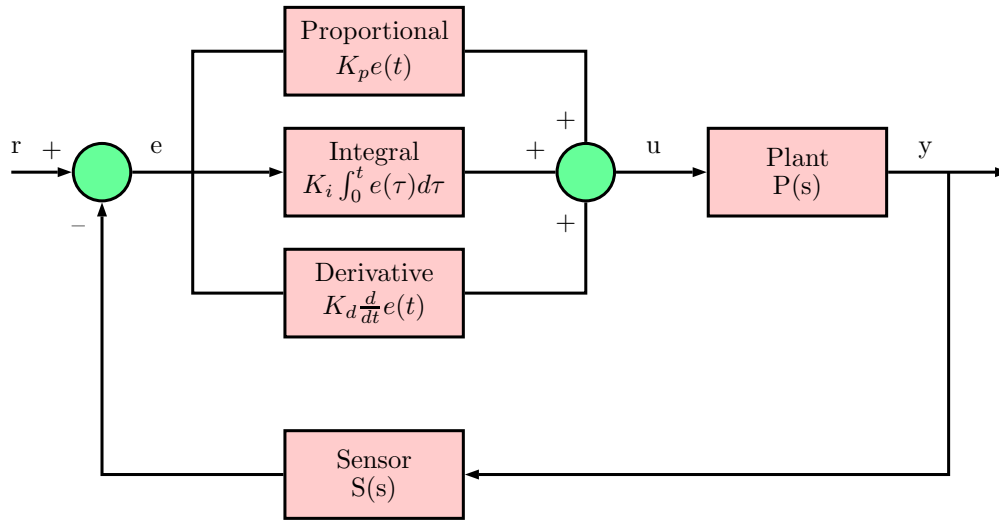
where we have defined the overall transfer function of the system,  $H(s)$  as:

$$H(s) = \frac{P(s)C(s)}{1 + P(s)C(s)S(s)} \quad (2.22)$$

$H(s)$  is the closed-loop transfer function of the system. The stability of a system can be assessed by studying  $H(s)$ . We can see that, when  $P(s)C(s)S(s) = -1$ , the denominator of the transfer function becomes zero. Any value of  $s$  for which this occurs is called a pole[47]. Since the  $s$  parameter of the Laplace transform is related to the frequency of the input signal, the poles of the transfer function represent frequencies at which the system diverges, indicating regions of instability.

The most common type of feedback controller in modern engineering is the PID controller. A PID controller was used in the welding depth control experiments described in chapter 4, and the autofocus experiments used a similar scheme (with the derivative component replaced by a secondary integral component). PID stands for Proportional-Integral-Differential, which is a description of the three types of feedback that are present in the system. The names of the three

feedback mechanisms effectively describe their action. The proportional controller is a simple feedback that is based only on the present sensor value, the integral controller is based on an integral over all previous sensor measurements, and the derivative gain is based on the current rate of change of the sensor value. Each of these controllers has a gain associated with it, denoted  $K_p$ ,  $K_i$ , and  $K_d$  respectively. A schematic showing the basic feedback loop described above with a PID controller is shown in figure 2.7.



**Figure 2.7:** Structure of a PID controller. The controller block in figure 2.6 is replaced by three separate feedback blocks for the proportional, integral, and derivative gains. Figure created by author using concepts from [46].

With PID control, the output of the controller is given by the sum of the three components:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \quad (2.23)$$

Calculating the Laplace transforms of each of the three terms allows us to express the transfer function for the controller as:

$$C(s) = K_p + \frac{K_i}{s} + sK_d \quad (2.24)$$

If we plug this expression for the controller transfer function into the expression for the overall transfer function in equation (2.22), we can see that the PID gains have an effect on the stability

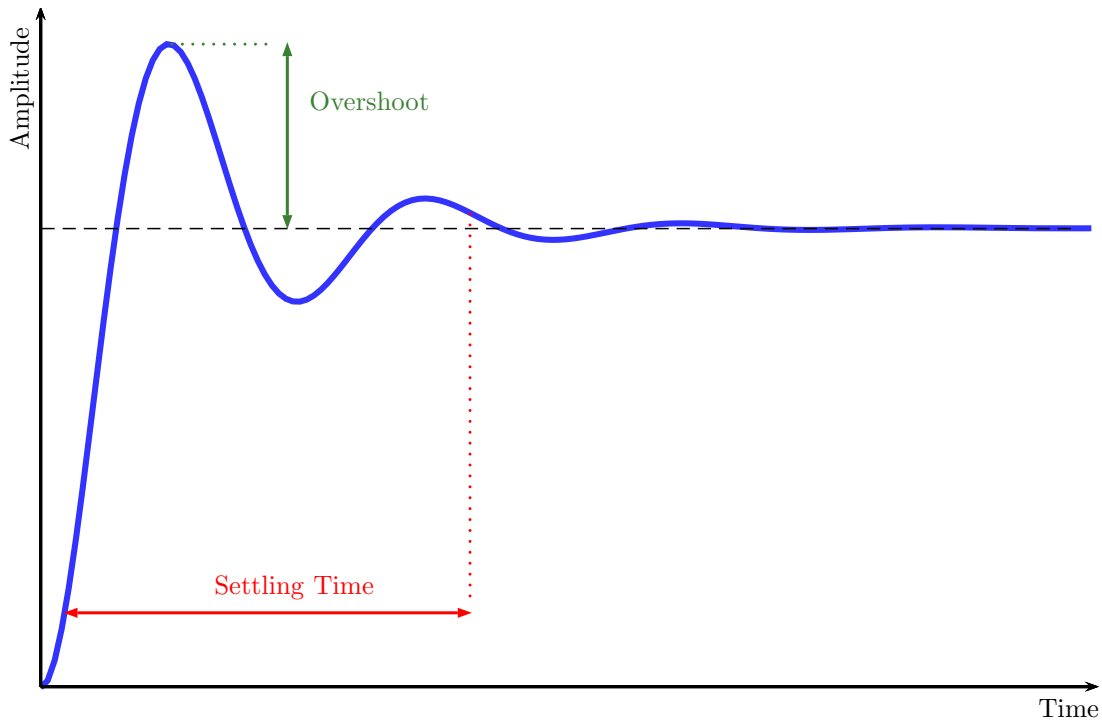
of the system. In general, increasing  $K_p$  and  $K_i$  degrades stability, whereas increasing  $K_d$  improves stability[48].

The performance of feedback loops is typically evaluated by analyzing the step response of the system[46]. The step response is, in general terms, the response of a system to an instantaneous change in the input, represented mathematically by a Dirac delta function. In practice, the step response of a control system can be measured by imposing an abrupt disturbance in the system or by changing the setpoint value while the system is in steady-state. In addition to the stability of the system, there are two parameters that are commonly used to characterize feedback loops that can be measured from the step response: settling time and overshoot. For a second-order feedback control system such as those used in the experiments described in chapter 4, the settling time of the step response is defined as the time it takes for the system to go from 10% of the final value to 90% of the final value, and the percent overshoot is equal to the maximum value divided by the difference between the initial and final values. Figure 2.8 shows a visual depiction of settling time and overshoot.

In designing feedback systems, it is important to choose optimal values of the PID gains  $K_p$ ,  $K_i$ , and  $K_d$ . This is achieved by minimizing overshoot and settling time for the step response and maximizing the steady-state stability of the system. Tuning can be achieved using theory-based methods such as the popular Ziegler-Nichols method[49, 50, 51] or the Tyreus-Luyben method[52], but it is also common (and often simpler) to tune the gains manually by varying one gain with the others held fixed until a desired level of performance is achieved. In chapter 4, the gains were tuned manually for the ICI control applications that were investigated.

## 2.4 Laser Keyhole Welding

In chapter 4, the application of FPGA-based ICI to control laser welding penetration will be discussed. Since laser welding is a dynamic process that the reader may be unfamiliar with, I will briefly discuss the basic principles in this section.

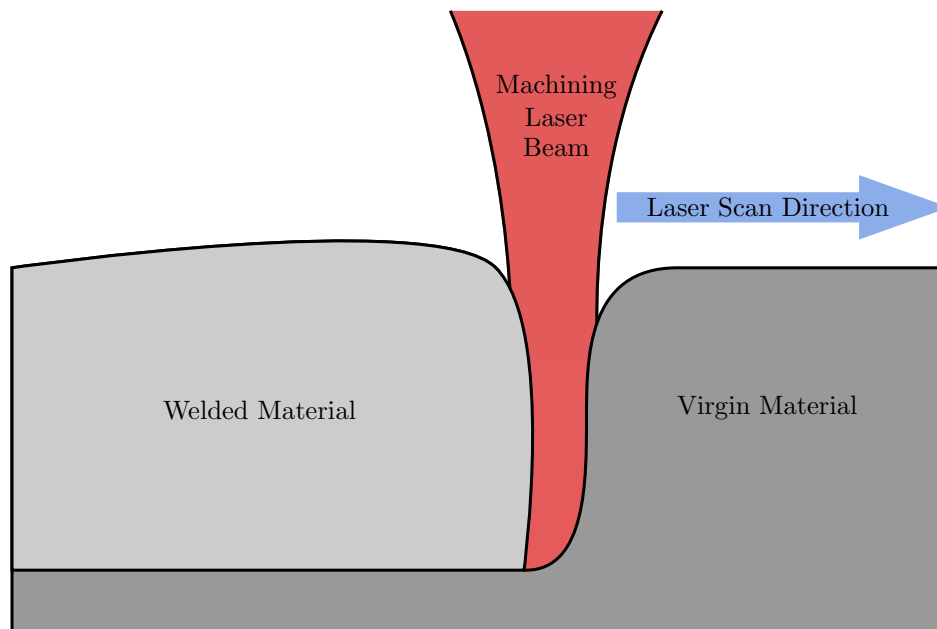


**Figure 2.8:** Step response of a feedback system. The system starts from zero and attempts to reach the setpoint (shown as the dashed black line) as quickly as possible. Overshoot and settling time are highlighted in green and red.

Laser welding has been used in industry as a replacement for traditional contact-based methods for decades, with the first laser welding machines appearing just several years after the invention of the laser itself[53]. When compared to other energy sources, industrial lasers output high energy density, allowing them to penetrate deep into a material. Additionally, the contact-free nature of lasers renders them immune to problems related to tool wear that are present in alternative welding methods such as friction-stir welding[54]. While electron beam welding, a similar process to laser beam welding, offers the same advantages in tool wear and energy density, lasers have lower capital costs and are therefore more appropriate for industrial deployments[55]. Other advantages over traditional welding methods include high processing speed, no x-rays generated, no filler material required, small heat-affected zone, low contamination, and ease of automation[56].

The most common form of laser welding is keyhole welding. This method is named after the characteristic “keyhole” capillary that develops in the material during the welding process. A basic

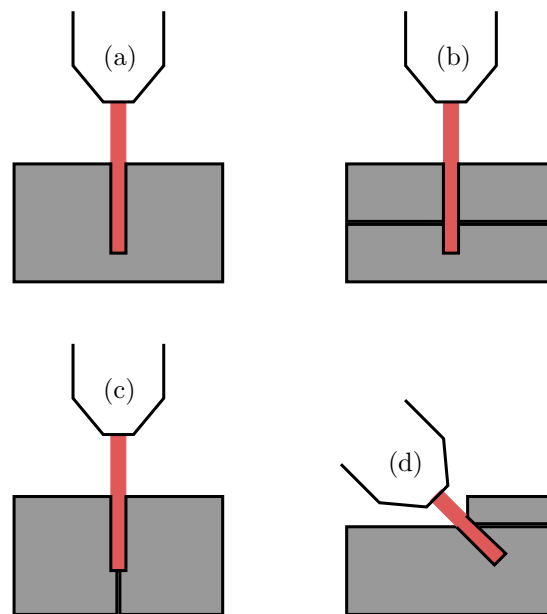
schematic of the geometry of a keyhole weld is shown in figure 2.9. When the laser turns on, the energy is absorbed by the material, and a melt pool forms around the region where the beam is incident. The laser continues to deposit energy until material begins to evaporate and a hole forms in the melt pool due to the large pressure differential. The laser continues to evaporate material in the hole until a long, thin channel called a keyhole forms. The geometry of the keyhole is typically characterized by a depth that is much larger than the diameter of the hole. The existence of the vapor-filled keyhole increases the efficiency at which energy is deposited in the material[57], and it is this high-efficiency absorption that keeps the keyhole open for the duration of the time that the beam is incident on the material. Once the keyhole is formed, the laser beam is scanned across the material to cover the desired weld area, as shown in figure 2.9.



**Figure 2.9:** Diagram showing the basic layout of a keyhole weld.

As the beam scans across the material, new material is melted at the front of the keyhole as the back of the keyhole is filled in by surface tension. The beam is scanned in such a way that melting occurs at an interface between two parts so that the re-filled keyhole joins the two materials together. There are four types of weld geometries that are common for laser keyhole welding. These four schemes are depicted in figure 2.10. In (a), the keyhole is only formed in one

material, so parts are not actually joined together. This is known as a “bead-on-plate” weld and is the common weld type used in research settings to test apparatus and determine useful parameter spaces to achieve specific keyhole depths[58, 59]. In the welding tests described in chapter 4, bead-on-plate welds were performed in stainless steel. In (b), (c), and (d), which represent lap, butt, and fillet welds respectively, two materials are joined together[60]. In the case of the fillet weld, the laser beam is angled so that it is incident on the corner of the contact between the two materials. In industrial settings, the type of weld geometry chosen largely depends on the nature of the parts being joined.



**Figure 2.10:** Diagrams showing the four common weld geometries: (a) represents a bead-on-plate weld, (b) represents a lap weld, (c) represents a butt weld, and (d) represents a fillet weld.

Keyhole welding is typically done in industrial settings with either CO<sub>2</sub> lasers with a wavelength of 10.6  $\mu\text{m}$ [61, 62] or fiber lasers with various gain mediums, typically ytterbium-doped fiber emitting at around 1.07  $\mu\text{m}$ [63, 64], although 2  $\mu\text{m}$  thallium fiber or 1.5  $\mu\text{m}$  erbium-doped fiber are also sometimes used[65]. Laser powers from 1 kW to 10 kW are common, although specialized applications use laser powers ranging from 200 mW for micro-welding of polymers[66] to 100 kW for deep penetration welding in stainless steel[67]. The welding experiments performed in

this thesis were performed on the “Macro” station described in the following chapter, which uses a 1.07  $\mu\text{m}$  ytterbium-doped fiber laser with a maximum output of 1.1 kW.

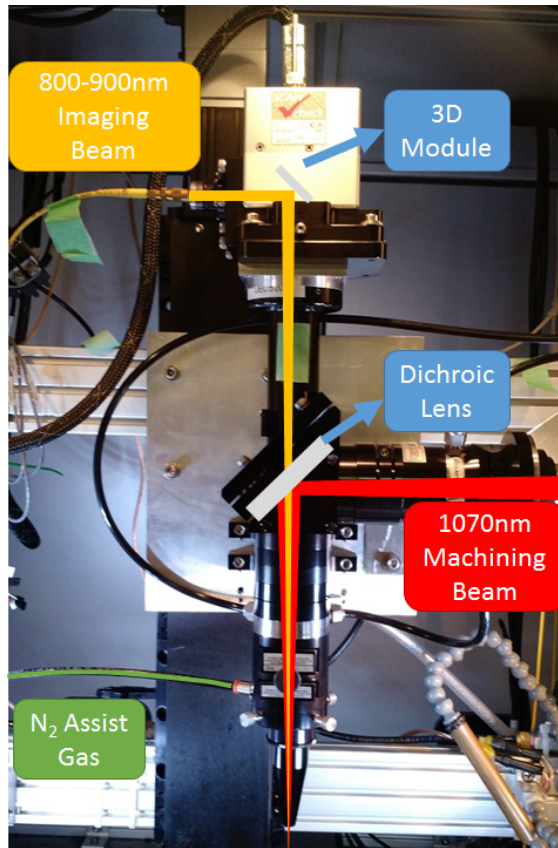
# Chapter 3 System Design

In order to examine the potential impact that the use of a programmable logic device such as an FPGA could have on ICI data processing, I developed an FPGA-based controller that was integrated into the existing laser processing station and ICI spectrometers in our lab. This required the ICI processing code to be completely re-designed, as the structure of FPGA code is very different from PC code. In this chapter, I will discuss the existing laser processing station and how I adapted it for FPGA processing. I will then explain the general structure of the signal processing algorithms used by the FPGA, followed by a detailed description of the important sections of LabVIEW code that are used to implement those algorithms. Finally, I will briefly discuss the DAC and Op Amp circuits that I built to make the output signals from the FPGA compatible with the hardware used in two feedback applications, autofocus of the beam delivery head and depth control of laser keyhole welding.

## 3.1 Laser Processing Station

In order to assess the performance of FPGA ICI, it was necessary to perform experiments using a realistic industrial laser processing station. The “Macro” station in our lab was used to perform these experiments (the naming convention arises from the existence of a similar “Micro” station that consists of laser sources that are better suited to micro-machining processes).

The processing light source in the Macro station is a 1 kW CW Ytterbium fiber laser that emits at  $1.07\ \mu\text{m}$ , manufactured by IPG Photonics (model number YLS-1000). While 1 kW is a relatively low power by today’s industrial standards (CW sources up to 100 kW are currently being used for laser welding in research settings[68]), it is able to produce laser keyhole welds with similar behavior to those that might be seen in industrial deployments. The machining beam

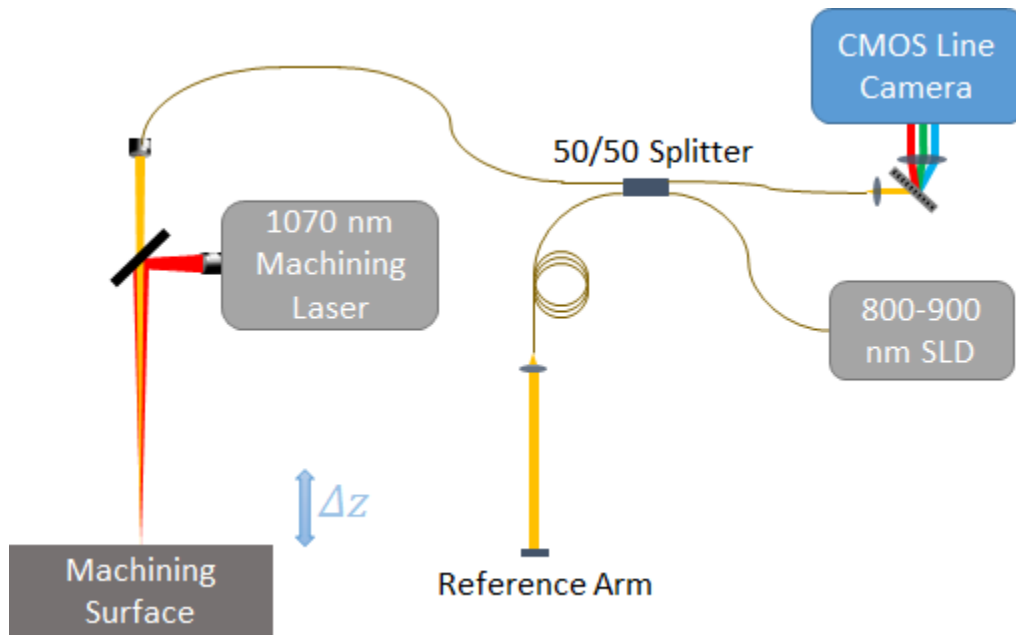


**Figure 3.1:** Picture of the laser beam delivery head with the relevant components highlighted and labeled. For scale, the metal mounting plate behind the laser head has a height of 210 mm.

travels through fiber into the Macro enclosure where it is coupled into an AccuFiber welding head manufactured by Laser Mechanisms Inc. The AccuFiber head, like most modern industrial laser beam delivery heads, includes a viewing port for a camera to be attached with a dichroic filter that reflects the  $1.07\ \mu\text{m}$  beam to the machining surface but transmits visible (and near-IR) light through the camera viewing port. Since the ICI imaging light is in the range of 800 – 900 nm, this allows us to introduce the ICI beam through the camera port and use the existing dichroic to couple/uncouple the imaging and machining beams. In the Macro enclosure, a 3D Module Head Interface from Laser Depth Dynamics is attached to the AccuFiber head’s camera viewing port so that the ICI beam can be scanned for alignment and 3D viewing purposes. Nitrogen assist gas is fed through the tip of the delivery head at a pressure of 40 psi in order to prevent any material ejecta

or other harmful materials from entering the head and damaging the focusing optics. A picture of the setup with the above components highlighted is shown in figure 3.1.

The imaging light fiber is connected to a commercial LD-600 ICI unit from Laser Depth Dynamics that sits outside of the Macro enclosure. A schematic of the experimental apparatus used to acquire ICI images is shown in figure 3.2.



**Figure 3.2:** Diagram showing essential components of the ICI apparatus. The components in this diagram constitute a mostly-fiber implementation of the Michelson interferometer in figure 2.1

The light source is a Superlum BLM2-D-I superluminescent diode (SLD) with a bandwidth of  $\sim 100$  nm centered at 840 nm. The diode is coupled to a single-mode optical fiber which is split with a 50/50 beam splitter, out of which the reference and sample arms are fed. The sample arm is coupled to the machining head described above, and the reference arm is coupled to free space and reflected off of several mirrors that produce an optical path length that is similar to that in the machining head path with the 3D module attached. Several of the mirrors are attached to an electronic motion stage to adjust the reference arm path length so that it can match a variety of sample arm lengths depending on the distance to the part and the focusing optics in the head. The

final output of the 50/50 splitter is connected to the spectrometer, which uses a Basler spl4096-140km line scan camera to acquire data with an integration time of  $3.6 \mu\text{s}$  (the minimum value supported for 512 pixels with a 4-tap 12-bit readoff at 40 MHz). The camera uses the Camera Link (CL) protocol to communicate with, and stream data to, external devices. In order to achieve the maximum data throughput possible, the “4-tap 12-bit” mode is used to stream data to the FPGA. In this mode, four 12-bit pixels are streamed to the FPGA in each 40 MHz clock cycle using the CL medium configuration. Since only eight bits can be transferred per port per clock cycle, six ports are used to transfer the four pixels simultaneously, with two of the ports containing four bits of information from two separate pixels[69]. A frame grabber card connected to the data processing hardware (PC or FPGA) is used to interface with the camera and acquire frames at rates up to 312 kHz.

The measurement depth range is given by the field-of-view of the ICI system. This is calculated by taking the product of the number of points in the FFT of the interference spectrum and the difference in depth between each point. On the FPGA ICI system I developed, there were 256 points in the spectrum with a spacing of approximately  $23.8 \mu\text{m}$  per point, giving a total depth range of approximately  $6100 \mu\text{m}$ . Since a brightest-pixel tracking algorithm was used, the spacing of  $23.8 \mu\text{m}$  is also an effective estimate of the real resolution of the system.

Motion of the sample relative to the beam head is realized using an Aerotech A3200 motion control suite. The part is moved in the X-Y plane using two horizontal motion stages, and the head is moved in the Z plane using a vertical stage (PRO115 stages are used for the Z and Y axes, and a PRO165 stage is used for the X axis). Motion commands are executed and synchronized with laser firing and imaging tasks using Aerotech’s Motion Composer suite. Motion programs are scripted in the industry-standard G-code language.

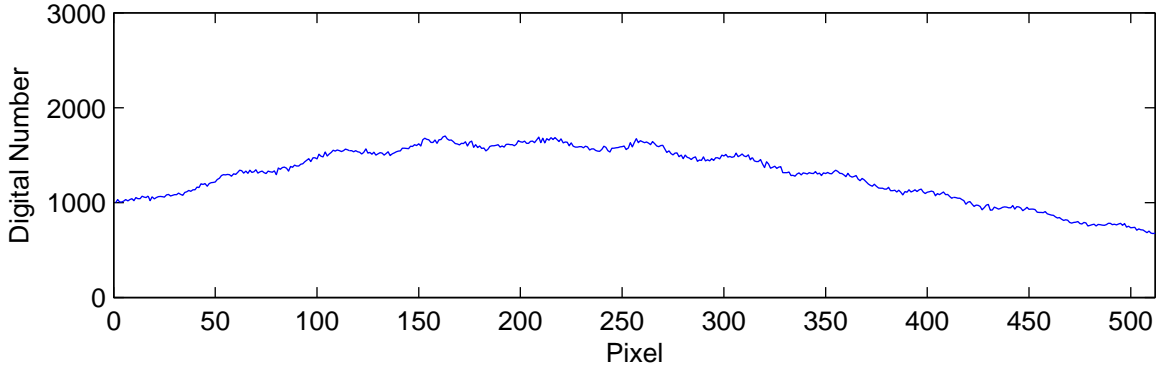
## 3.2 ICI Signal Processing

In order to develop algorithms that take advantage of the FPGA architecture to efficiently process high volumes of ICI data, it is necessary to understand in detail the steps involved in processing each frame of data. In this section, I will give an overview of the three main steps (DC subtraction, resampling, and FFT) required to extract a depth measurement from a single spectrum of ICI data.

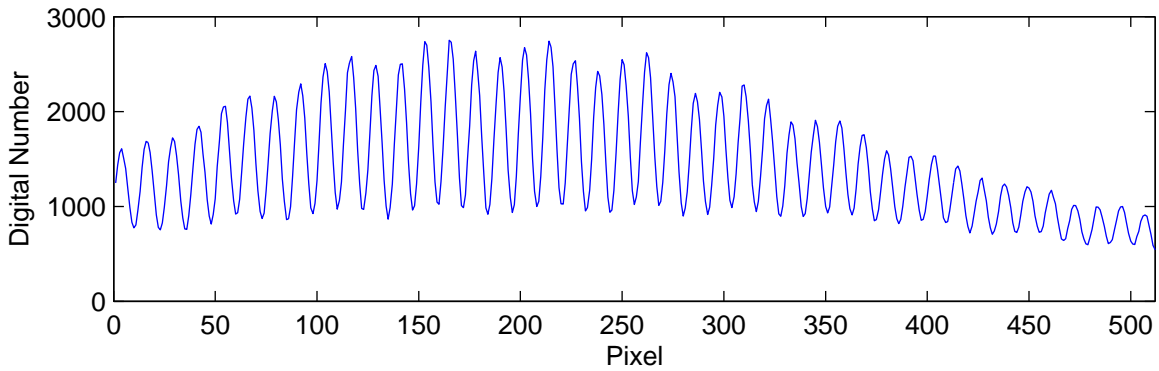
### 3.2.1 DC Subtraction

From equation (2.12), we can see that the interference pattern that is incident on the spectrometer consists a superposition of multiple DC components in addition to the oscillatory term that represents the interference. There is also a contribution to the DC spectrum from the sample arm reflection; however, since the properties of the interface are not constant, it is impossible to consistently subtract off. The shape of the reference arm DC offset depends on the frequency bandwidth of the SLD that is used for illumination. An example of a DC spectrum is shown in figure 3.3 and in figure 3.4 with an interference pattern superimposed on top of it. Note that the units on the y-axis are a 12-bit “Digital Number,” since that is the format in which the data is provided by the camera in the spectrometer. Also note that the spectrum size is 512 pixels. This number was chosen because the FFT is a ‘divide and conquer’ algorithm that takes advantage of the repeated divisibility of a signal by two, and therefore only works on signals with a number of points that is a power of two. The spectrometer apparatus only produces a significant signal on about 570 of the camera’s pixels, so 512 was chosen since it is the closest power of two.

After the DC component is subtracted, we are left with a signal with a significantly reduced DC offset (figure 3.5). The signal still has a slight offset due to the DC component that is reflected from the sample arm; however, since the intensity of the sample arm signal is much less than that of the reference arm signal, the DC offset it causes is relatively small and can be ignored.



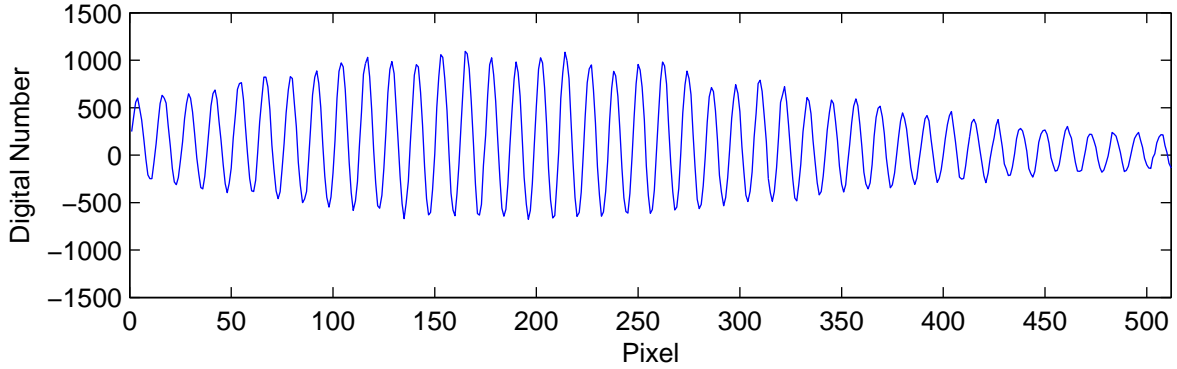
**Figure 3.3:** DC spectrum from the SLD in the macro processing station spectrometer obtained by blocking the sample arm and capturing a camera image with just the reference arm signal passing through.



**Figure 3.4:** Sample ICI interference pattern from a single backscattering surface in the sample arm including the offset caused by the DC components in equation (2.12).

### 3.2.2 Resampling

The spectrum that is read from the spectrometer camera contains an x-axis that is equally spaced in position, defined by the spacing between the pixel elements of the camera’s CMOS sensor array. While each sensor position maps to a wavelength (and a corresponding wavenumber  $k$ ) of light, the spacing in the  $k$  domain is not uniform. We must therefore resample the spectrum so that it is equally spaced in  $k$  before we can perform an FFT on it. As space on the FPGA is limited, as much processing as possible is done on the host computer prior to the execution of the FPGA code. The program starts by generating a fourth-order polynomial using a set of coefficients that maps pixel



**Figure 3.5:** Interference spectrum from the macro processing station spectrometer after the DC spectrum from the reference arm has been subtracted.

number to  $k$ . These “calibration coefficients” were generated by comparing the locations of the emission peaks of argon on the spectrometer to tables of argon peaks from the National Institute of Standards and Technology (NIST)[70].

The resampling is done by linearly interpolating between pixels in the interference spectrum. More complicated algorithms that implement nonlinear interpolation methods have been presented for similar applications[71]; however, these sophisticated algorithms present a tradeoff wherein signal clarity is improved at the cost of increased data processing latency, resource usage, and implementation complexity. Additionally, research suggests that peak widths of A-line measurements obtained using cubic spline and nearest neighbor interpolation algorithms are comparable to those obtained using linear interpolation, and therefore they do not provide sufficient improvement to justify the additional computational strain[72]. A basic equation for a linear interpolation of the function  $f(x)$  at an arbitrary point  $x$  that lies between the two points  $x_1$  and  $x_2$  is given in equation (3.1).

$$f(x) = f(x_1) + (x - x_1) \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad (3.1)$$

where, in the case of ICI,  $x$  corresponds to the pixel number of the detector, and  $f(x)$  corresponds to the signal intensity measured by that pixel.

This is a rather simple formula; however, it is not optimized for the FPGA architecture. Imple-

mentation of fixed-point or floating-point division on FPGAs introduces additional complexity and latency and should be avoided if possible. Equation (3.1) contains six operations, one of which is a division. However, for our application, the  $x$  values are independent of the interference spectrum and are constant throughout the execution of the program. We can therefore reduce latency and save FPGA resources by rearranging the equation so that some of the computation is done *a priori* on the host PC, with only the resulting arrays being sent to the FPGA. While I developed this algorithm independently, a similar algorithm for OCT on an FPGA has been developed and presented by Ustun et al[34]; this provides external verification that this algorithm is indeed optimized for the FPGA. First, we generalize equation (3.1) for the  $n^{\text{th}}$  interpolated point. This point does not necessarily lie in between the  $n^{\text{th}}$  and  $(n + 1)^{\text{th}}$  points of the original spectrum, as illustrated in figure 3.6. The top image shows one possible case, where there can be two points in the raw spectrum between consecutive interpolated points, requiring one of them to be skipped. The bottom image shows a second case, when there may be two interpolated points between consecutive raw spectrum values. This requires the two raw spectrum values to be repeated for consecutive interpolations.



**Figure 3.6:** Visualization showing the unevenly sampled raw signal  $x$ -values as red dots and the evenly spaced resampled  $x$ -values as blue lines. The case where two raw spectrum values fall between consecutive interpolated values is shown on the top, and the case where two interpolated values fall between consecutive raw spectrum values is shown on the bottom.

To account for this, an “index shift” is assigned for each value of  $n$  and denoted  $m(n)$ :

$$f(x) = f(x_{n+m(n)}) + (x - x_{n+m(n)}) \frac{f(x_{n+m(n)+1}) - f(x_{n+m(n)})}{x_{n+m(n)+1} - x_{n+m(n)}} \quad (3.2)$$

The index shifts are calculated by taking the difference between the  $k$  value of the  $n^{\text{th}}$  pixel from the raw interference spectrum (denoted  $k_{\text{raw}}(n)$ ) and the  $k$  value of the  $n^{\text{th}}$  point in the interpolated spectrum (denoted  $k_{\text{int}}(n)$ ) as follows:

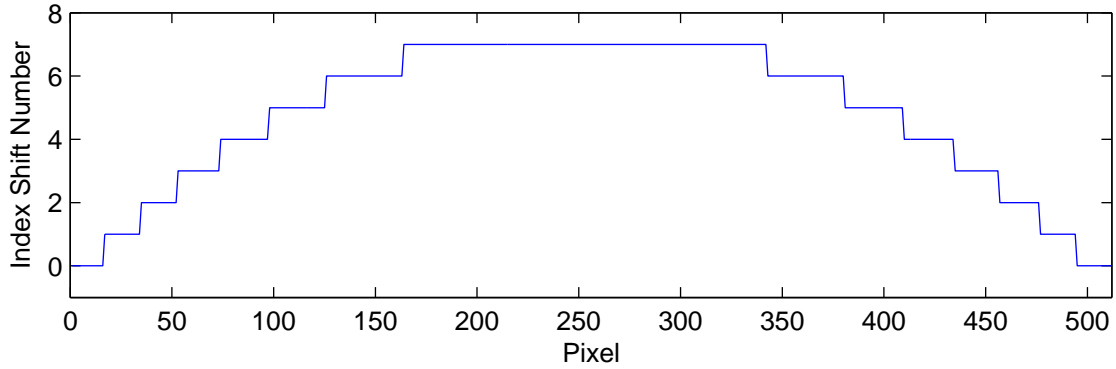
$$m(n) = \text{floor} \left[ \frac{k_{\text{int}}(n) - k_{\text{raw}}(n)}{\Delta k_{\text{int}}} \right] \quad (3.3)$$

where  $\Delta k_{\text{int}}$  is the spacing between the  $k$  values in the interpolated spectrum (which are equally spaced), and the floor operation rounds down to the nearest integer. Since the index shifts depend only on the experimental apparatus used in the spectrometer and not on the measured ICI spectrum (and only change whenever a physical change has been made to the spectrometer setup), they are calculated *a priori* and accessed as needed when ICI processing is occurring. An example of an index shift array is plotted in figure 3.7. When the index array value is the same as the previous value (ie. the plot is flat), each consecutive pair of pixels has one interpolated point that lies between them. An increase of one in the index shift value identifies a pair of pixels between which there is no interpolated point (this is the “skip” condition in figure 3.6). Finally, a decrease of one in the index shift value identifies a pair of pixels that have two interpolated points between them (the “repeat” condition in figure 3.6).

To further simplify the interpolation equation such that it is optimized for performance on the FPGA, we can define an interpolation constant and denote it  $S(n)$ :

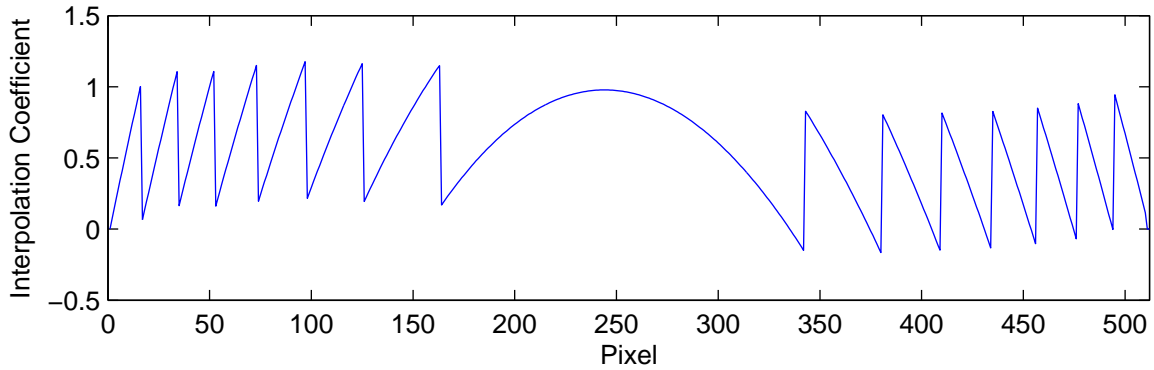
$$S(n) = \frac{x_n - x_{n+m(n)}}{x_{n+m(n)+1} - x_{n+m(n)}} \quad (3.4)$$

The interpolation constant is only dependent on x-axis values (which are  $k$  numbers) and can, as with the index shift array, be calculated *a priori* then accessed as needed when ICI processing



**Figure 3.7:** Index shift array generated for the macro spectrometer.

is occurring. An example is plotted in figure 3.8.



**Figure 3.8:** Interpolation coefficient array for the macro spectrometer.

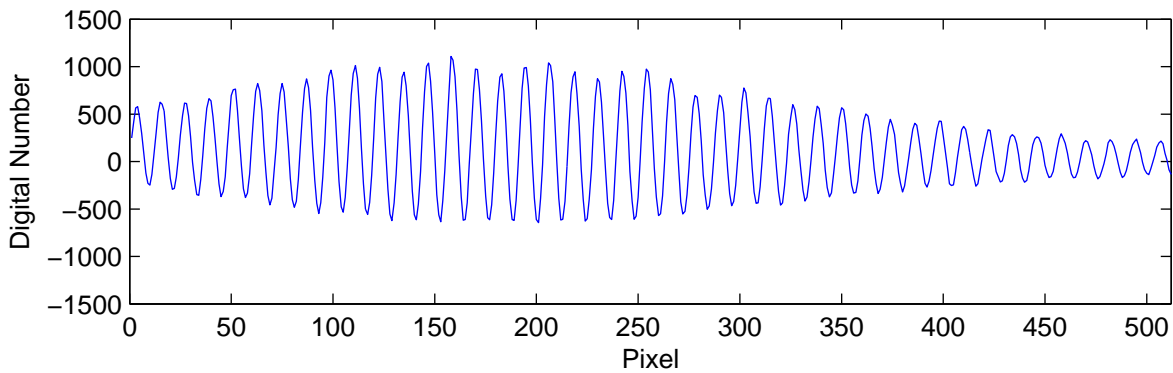
Now that we have defined these two arrays, we can write the interpolation equation as follows:

$$f(x_n) = f(x_{n+m(n)}) + S(n) [f(x_{n+m(n)+1}) - f(x_{n+m(n)})] \quad (3.5)$$

Comparing to equation (3.2), we can see that we have reduced a calculation with six operations (three subtractions, one addition, one multiplication, and one division) to one with only three (one subtraction, one addition, and one multiplication), and that we have eliminated the problematic division operation. This algorithm is much more suited to the abilities of the FPGA as it consumes fewer resources and reduces the complexity of implementation.

Figure 3.9 shows the spectrum in figure 3.5 after it has been resampled as described in this

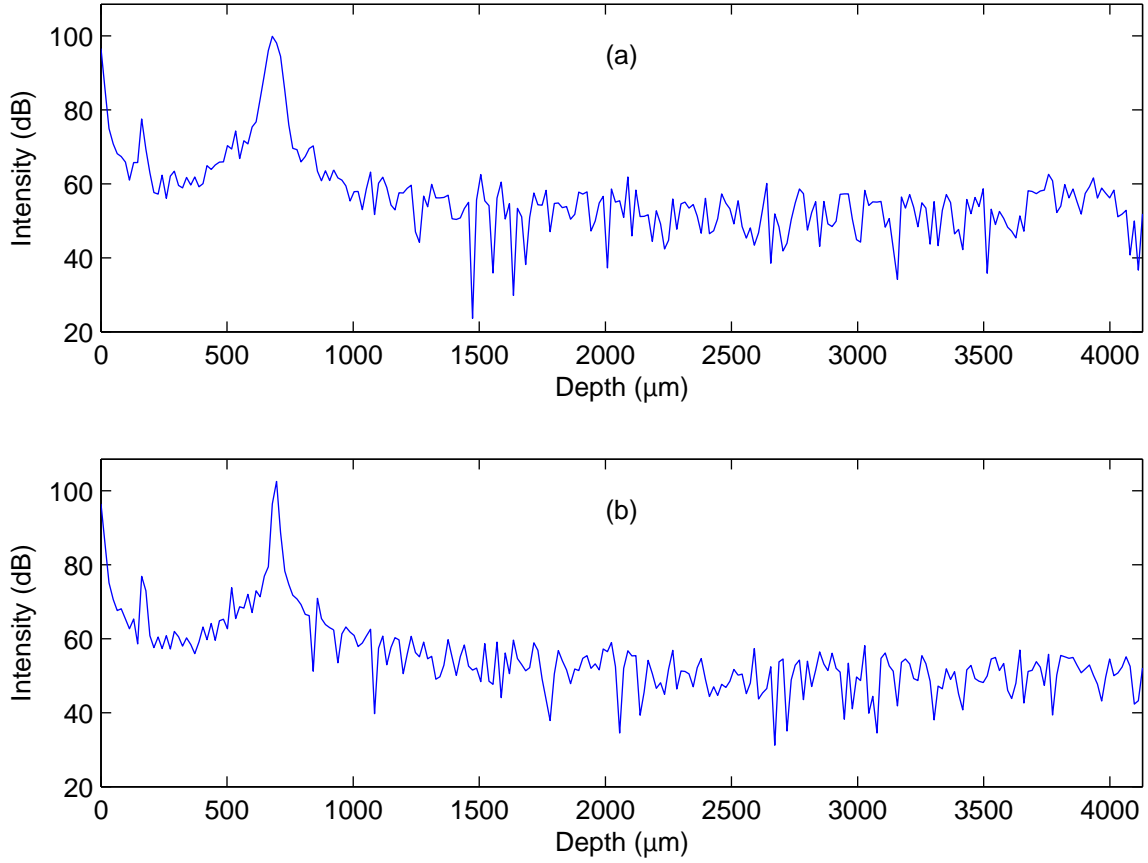
section. Upon first inspection, it is not immediately evident that there is any improvement over the original spectrum in figure 3.5. However, the increase in resolution becomes obvious after calculating the Fourier transform, as illustrated in figure 3.10. The FWHM of the spectrum without interpolation in figure 3.10(a) is  $58.8 \mu\text{m}$ , whereas the spectrum with the interpolation in figure 3.10(b) has a FWHM of  $25.5 \mu\text{m}$ . Also note that this spectrum was acquired with an interface that is relatively close to the zero-delay point (the point where the reference and sample arms in the interferometer are matched). The effect of resampling is more dramatic for interfaces that are closer to the maximum depth window since they produce fringes that are more closely spaced.



**Figure 3.9:** Interpolated spectrum calculated using the data in figure 3.5.

### 3.2.3 Depth Recovery via FFT

After the spectrum has been resampled, a Fourier transform is applied to extract the useful depth information from the interference pattern. This is achieved using the Fast Fourier Transform (FFT) algorithm, which is an efficient implementation of the Discrete Fourier Transform that takes advantage of the divisibility of signals with a number of samples that is a power of two (in the case of ICI, I used 512-point samples). After the FFT is calculated, a scaling factor is applied to the x-axis to give it units of microns. The scaling factor ensures that the 0<sup>th</sup> FFT point corresponds to the “zero-delay” point of  $0 \mu\text{m}$ , and the spacing between FFT points is given by:



**Figure 3.10:** Comparison of the FFT of the raw (uninterpolated) spectrum (a) and the FFT of the interpolated spectrum (b). It is clear that the interpolated spectrum produces a much sharper peak, thereby increasing the resolution of the system.

$$\text{FFT Spacing} = \frac{\pi}{\Delta k_{\text{int}}} \quad (3.6)$$

where  $\Delta k_{\text{int}}$  is the  $k$ -spacing used in equation 3.3. Note that the numerator only has a factor of  $\pi$  (rather than  $2\pi$ ) due to the fact that the imaging beam travels a round trip through the reference and sample arm paths, and therefore we must divide by two to obtain the distance to the interface in the sample arm. Figure 3.10(b) shows the transformed spectrum with the scaled x-axis in  $\mu\text{m}$ . As there is an obvious dominant frequency component in figure 3.10, it is a simple task of finding the FFT point with the greatest amplitude (or the “brightest pixel”) to obtain the ICI depth measurement.

More sophisticated methods such as Gaussian fits may be used, however the brightest pixel method introduces less latency and complexity, and provides a satisfactory level of accuracy for control of the laser machining processes discussed in this thesis. Additionally, there is a peak at the zero-delay point due to the sample arm DC contribution that could falsely be detected as the interface location by a brightest pixel calculation. This peak arises from the presence of the DC component of the spectrum. While DC subtraction is performed to minimize the size of this peak, it is not entirely eliminated primarily due to the DC contribution from the sample arm (as discussed in section 3.2.1). It is therefore also necessary to exclude the first several points of the spectrum when parsing for the brightest pixel.

### **3.3 LabVIEW FPGA Processing Algorithms**

The previous section describes, in a general sense, the algorithms used to process an ICI signal that were developed with consideration given to the limitations of the FPGA architecture and how to best take advantage of the capabilities of an FPGA. However, there are a variety of subtle aspects of the code that is used to implement signal processing algorithms that are important to consider. In this section, I will provide a more complete description of the code I wrote for the FPGA so that one may reference this section if they are attempting to modify my code in the future.

FPGA code is typically developed using a hardware description language (HDL), with the two most common languages for FPGAs being VHSIC Hardware Description Language (VHDL) and Verilog[73]. HDLs are text-based programming languages that follow a syntax similar to the C programming language. While they are effective and a number of tools exist to simplify the development of HDL code, there is still a steep learning curve that must be conquered before one can effectively implement complicated real-time signal processing tasks such as ICI processing. To simplify the process of developing FPGA code, National Instruments has created a platform that allows code for Xilinx FPGA devices to be written in their graphical programming software LabVIEW. LabVIEW allows for an intuitive representation of parallel tasks by placing them sep-

arately in single-cycle time loops (SCTL) in LabVIEW’s flow-chart style visual layout. SCTLs differ from loops on regular computers because they execute at a constant rate that is defined by the hardware clock on the FPGA (40 MHz for the system that I used to develop ICI code). While this provides a deterministic environment in which to place code, it also limits the size of the code that can be placed inside the loop since any signal path within any loop cannot take longer than a single clock cycle to execute.

The system used to develop the FPGA ICI code is an NI PXIe-7965R, which includes a Xilinx Virtex-5 SX95T FPGA with 58,880 Flip-Flips, 640 Multiplication DSP Slices, and 8784 kbits of embedded block RAM[74]. The large number of DSP slices allows for multiplication tasks to be implemented efficiently without using flip-flops, freeing them up for other logic. The PXIe system includes a PC running the Windows XP operating system with the LabVIEW development environment installed, as well as the FPGA module that allows code to be developed and compiled for FPGA devices. Attached to the system is a NI 1483 Camera Link (CL) adapter module, which includes connections for 26-pin CL cables as well as a 15-pin D-sub connection that contains four general-purpose digital I/O lines. This CL adapter module is used to connect the FPGA to the ICI spectrometers in our lab for data acquisition, and the I/O lines are used to generate feedback control and triggering signals. A list of resource usage, maximum clock rates, and compilation time specifications for my ICI code is shown in table 3.1.

Specification	Value
Total Slice Usage	12488 (84.8%)
Registers	32742 (55.6%)
LUTs	37624 (63.9%)
Block RAM	167 (68.4%)
DSP48s	134 (20.9%)
40 MHz Clock Maximum Rate	53.60 MHz
160 MHz Clock Maximum Rate	160.69 MHz
Compilation Time	2:30:29

**Table 3.1:** List of resource usage specifications, timing limitations, and compilation time for the most recent iteration of the compiled FPGA ICI code (as of July 16, 2015).

LabVIEW FPGA code is developed in the Windows environment on the host PC, then compiled to FPGA “bitfiles” that are uploaded to the FPGA target. LabVIEW programs are referred to as virtual instruments, or VIs, which can be executed independently or reference one another similarly to functions or methods in object-oriented programming languages. Code can be developed to run independently on the FPGA, but can also be designed to interface with code that is running simultaneously on the host using direct memory access (DMA) first-in first-out (FIFO) buffers. In order to access information stored on the host PC’s storage media, it is necessary for a host-side VI to first load data from the hard drive then stream it to the FPGA using a DMA FIFO. Similarly, for data that is processed by an FPGA VI to be permanently stored for future viewing, it must be streamed via DMA FIFO to a host-side VI, which can then save the data to the hard drive.

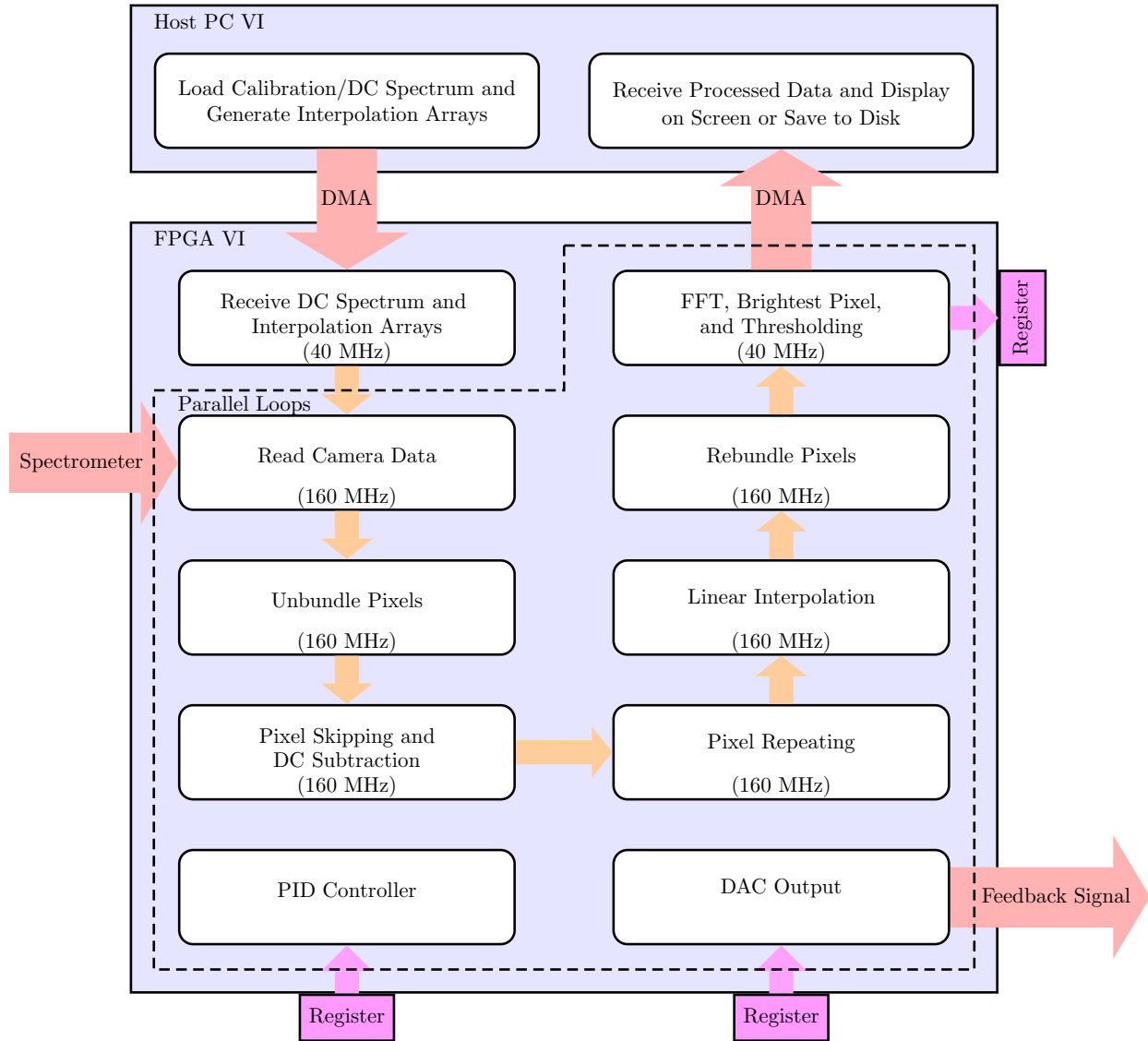
In this section, I will describe in some detail the code that I developed for the FPGA to process ICI data and generate feedback control signals. The general structure is outlined in figure 3.11. Program execution starts by running the host computer VI, which loads the calibration and configuration files that are saved on the disk, initializes the FPGA VI, then sends the DC spectrum and interpolation arrays to the FPGA using a DMA FIFO. The FPGA receives the information and stores them in block RAM since they are only calculated once, then begins to read frames from the camera. Because the camera operates in a 4-tap mode, wherein four pixels are streamed simultaneously per 40 MHz clock cycle (set by the FPGA’s clock), the pixels must be “unbundled” so that they can be resampled one at a time in a 160 MHz clock domain. The pixels are then “rebundled” and processed using the FFT VI in the 40 MHz domain, and the resulting spectrum can be streamed to the host computer via DMA FIFO. The brightest pixel detection is performed in the same SCTL as the FFT function, and the results are passed to registers that are read by the feedback processing loop, which runs a feedback algorithm and outputs control signal instructions to a register. Finally, the control signal register is read by the DAC loop, which updates the system’s output voltage whenever a new value is read. I will include important pieces of LabVIEW code throughout this section so that readers who are familiar with the LabVIEW language may refer to them. Algorithms were developed in MATLAB prior to the LabVIEW implementations in order to

ensure the accuracy of the calculations by comparing the results produced by the FPGA algorithms to those from MATLAB's built-in functions. I have included samples of the equivalent MATLAB code in Appendix A when applicable so that readers more familiar with that language may instead refer to them.

### 3.3.1 Initialization and Calibration Arrays

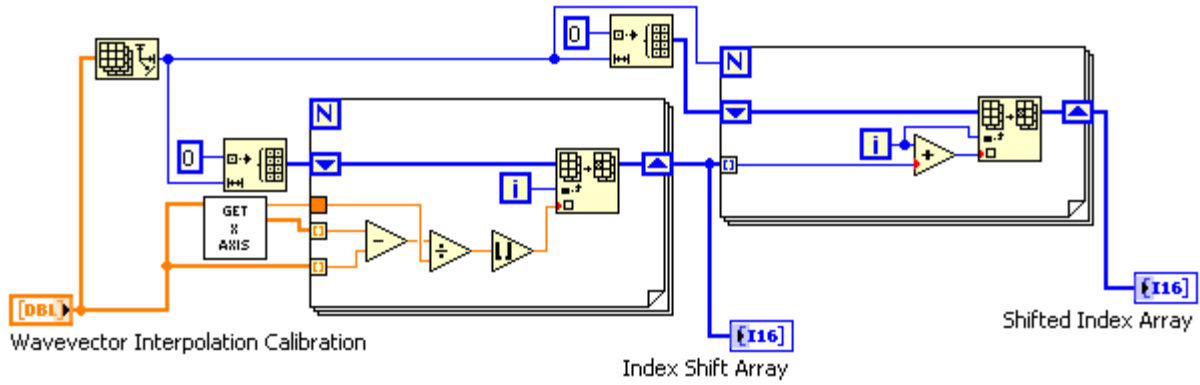
Because setup tasks are performed only once when the program is run, and resources on the FPGA are limited, it is preferable to perform as many setup tasks as possible on the host PC and then stream configuration data to the FPGA. Therefore, the loading of the DC Spectrum and calculation of the interpolation and index shift arrays are done prior to the execution of the FPGA code and then streamed via DMA FIFO. The code used to generate the index shift array is shown in figure 3.12, and the equivalent MATLAB code is shown in figure A.1. The interpolation array is generated using the code in figure 3.13 (MATLAB code is shown in figure A.4). Both the index shift and interpolation arrays are generated using the algorithm described in section 3.2.2. A "Depth per FFT Point" value is also calculated so that the ICI spectrum can be properly interpreted in units of  $\mu\text{m}$ .

In addition to the index and interpolation arrays, the "Number of Occurrence" (NoC) arrays are also generated at this stage. Whenever a change occurs in the index shift array, it corresponds to a pixel that is either skipped or repeated for consecutive linear interpolations in the resampling of the spectrum. The NoC arrays track these changes and assign a value of 0, 1, or 2 to each pixel depending on how many times it is used (there is necessarily an equal number of 0s and 2s so that they cancel out, and we end up with the same number of pixels as we started with). Two NoC arrays are generated (one to hold the  $x_1$  values in equation (3.2) and another to hold the  $x_2$  values). The LabVIEW code that is used to generate the NoC arrays is shown in figure 3.14 and the MATLAB equivalent in figure A.3. The use of these arrays will be further explained in section 3.3.4 where I discuss the FPGA implementation of the resampling in more detail. To simplify the

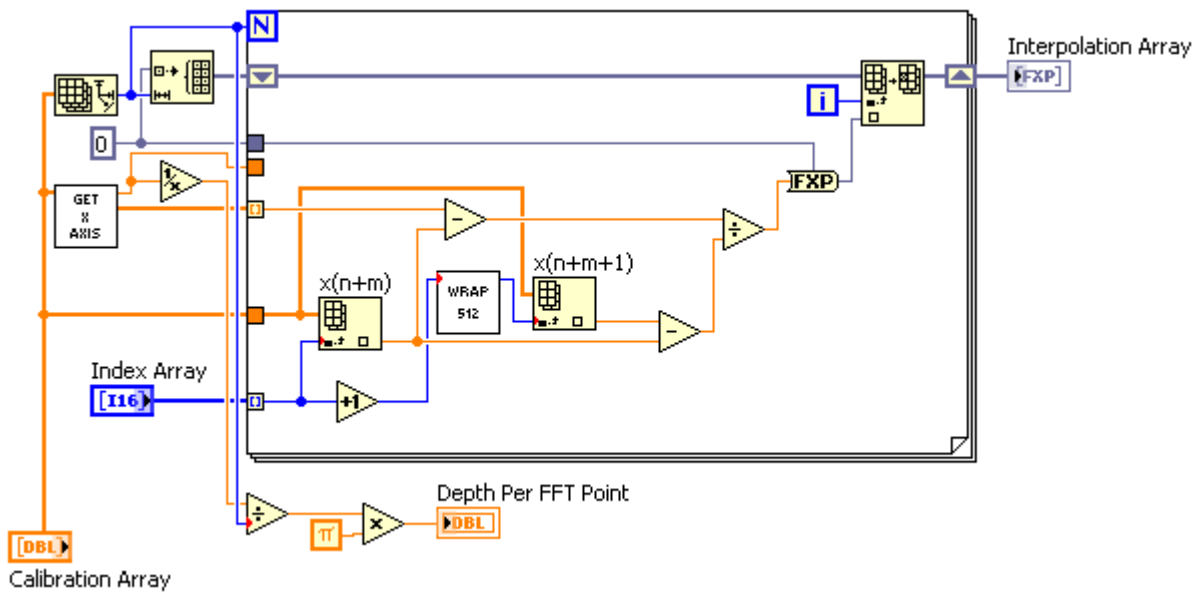


**Figure 3.11:** Flowchart summarizing all of the essential processing steps and loops that comprise the FPGA ICI processing code. For tasks running in single-cycle timed loops, the clock rate is shown. Yellow arrows indicate data flow between FPGA tasks through target-scoped FIFOs; purple arrows indicate data being written to/read from internal registers, and red arrows indicate signal transmission between physical devices. The steps enclosed in the dashed line all occur in parallel with chunks of data being passed between them.

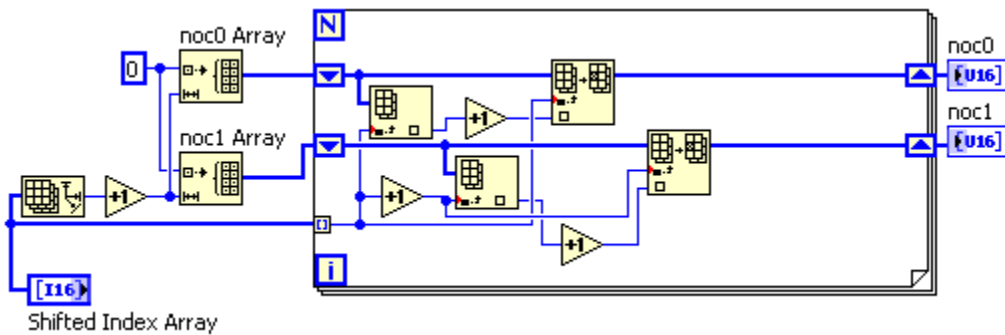
FPGA implementation, each NoC array is converted into two separate boolean arrays (one that tracks only the skipped pixels and another that tracks only the repeated pixels) so that there are a total of four NoC arrays sent to the FPGA.



**Figure 3.12:** LabVIEW code used to calculate the index shift array.



**Figure 3.13:** LabVIEW code used to calculate the interpolation array.

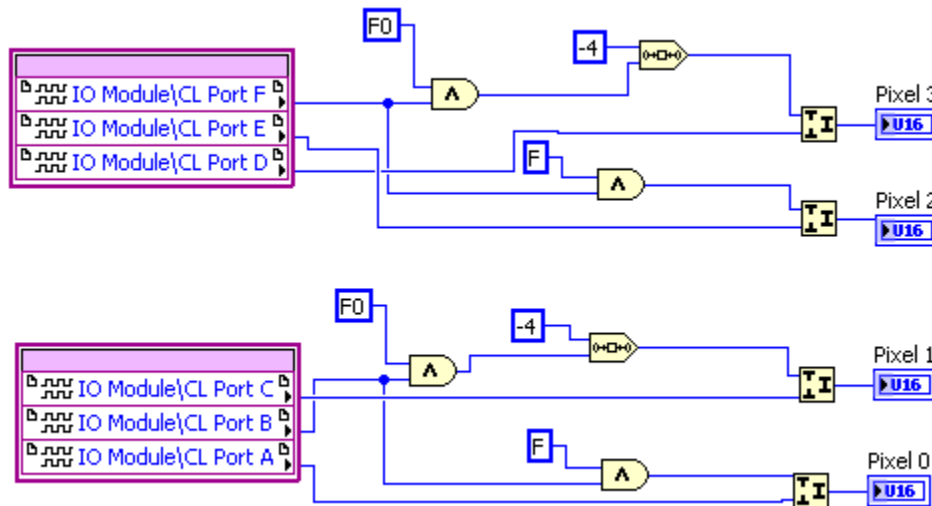


**Figure 3.14:** LabVIEW code used to calculate the NoC arrays.

After it has finished generating all of the necessary configuration arrays, the host PC initializes the FPGA VI and streams the interpolation array, NoC arrays, and DC spectrum via DMA FIFO. The index shift array is not sent since all of its useful information is contained in the NoC arrays. After the FPGA VI has received the configuration arrays, it proceeds to execute the ICI processing loops.

### 3.3.2 Camera Reading Loop

The first of the parallel loops that run on the FPGA is the camera reading loop. Pixels are read using the “4-tap 12-bit” configuration as described in section 3.1. The LabVIEW FPGA code that handles the CL input and interprets the pixels is shown in figure 3.15. Integer data types contain numbers of bits that are powers of two, so each pixel is padded with four zeros so that they are represented using 16 bits. After the pixels are generated, the four 16-bit integers are merged into a single 64-bit integer and written to a FIFO so that they can be read by the next loop in the processing stage.



**Figure 3.15:** LabVIEW code used to read pixels using the 4-tap 12-bit CL configuration.

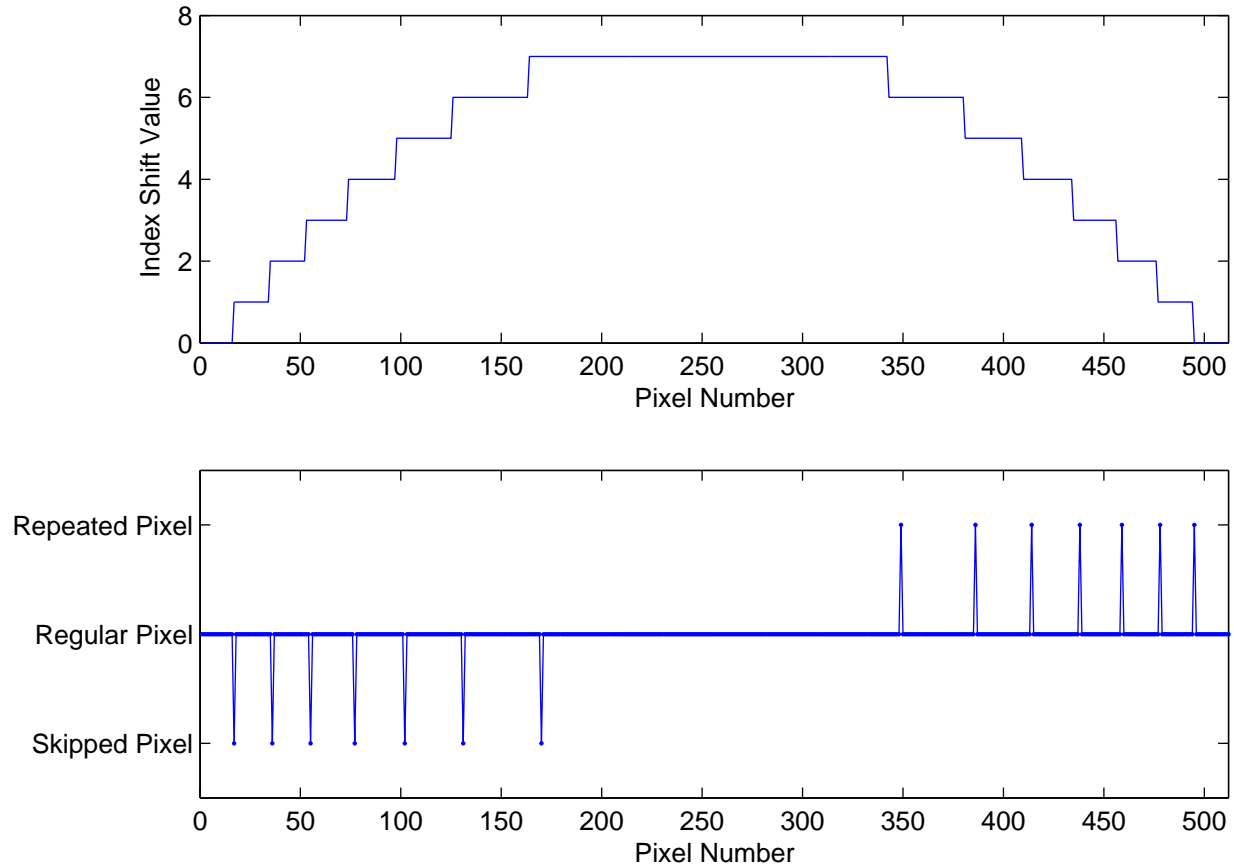
### 3.3.3 Pixel Unbundling Loop

The pixels that are streamed from the spectrometer to the FPGA are transferred in “bundles” of four at a time in the 40 MHz clock domain. This presents a complication due to the fact that there is no straightforward method of resampling between consecutive pixels (some of which are skipped or repeated) when they are flowing in such a parallel scheme. The solution is to “unbundle” the pixels and perform the resampling in a 160 MHz clock domain. This way we are able to switch from a parallel scheme to a serial configuration without seeing a reduction in overall data throughput. The unbundling SCTL runs in the 160 MHz clock domain, reading a 64-bit number and splitting it into the four separate 16-bit numbers that each contain the value of one pixel. Each pixel is written sequentially to the output FIFO before another 64-bit integer is read from the input FIFO. In this way, data is read at 40 MHz and written at 160 MHz so that proper data synchronization is achieved. Algorithmically, the unbundling loop is implemented using a state machine architecture with five states; the first four write respective pixels to the output FIFO with the fourth also reading a new set of pixels from the input FIFO, and the fifth handles the timeout case when there is no data to be read from the input FIFO.

### 3.3.4 Pixel Skipping and Repeating Loops

As described in section 3.2.2, the input spectrum must be resampled using linear interpolation so that it is linear in  $k$ -space. The resampling algorithm uses index shifts that are generated by assigning an offset value for each pixel in the resampled spectrum. We then simply use the  $(n + m)^{\text{th}}$  and  $(n + m + 1)^{\text{th}}$  points from the raw spectrum to interpolate the  $n^{\text{th}}$  point in the resampled spectrum, where  $m$  is the  $n^{\text{th}}$  element in the index shift array. However, on the FPGA architecture, processing of incoming frames starts immediately upon receipt of the first set of four pixels (before the entire frame has been read), and it is therefore not possible to access arbitrary points in the spectrum on demand. The index shifting is therefore implemented by designating a “skip” or “repeat” value to each pixel that automatically takes the index shifting into account (as discussed

in section 3.2.2). These skip and repeat values are contained in a “Number of Occurrence” array (NoC). Figure 3.16 shows the NoC array for the example index shift array used to describe the resampling process.



**Figure 3.16:** Example index shift array repeated from figure 3.7 (top) with corresponding NoC array showing the skipped and repeated pixels (bottom) aligned with the respective changes in the index shift value.

To further simplify the processing on the FPGA, the NoC array is split into two separate boolean arrays, one that contains only the skipped indices and one that contains only the repeated indices. The pixel skipping array is a boolean array consisting of values of false for “regular pixels” and “repeated pixels” and values of true for “skipped pixels.” This logical assignment was arbitrarily chosen, however there would be no chance of seeing any performance improvements if it were switched due to the fact that the FPGA operates at a set clock rate which is limited by other

more complex parts of the code. Pixel skipping is then a simple matter of excluding pixels from being written to the next processing step when they have a corresponding value of true in the pixel skipping array. The pixel skipping array runs in the 160 MHz domain and processes incoming pixels one at a time. Since two consecutive pixels are required for an interpolation, there are two buffers that are written to by this loop. The first is generated based on the pixel skipping array as shown in figure 3.16, and the second shifts the pixel skipping array by one pixel. This loop also handles the DC subtraction step by loading the DC spectrum value for each pixel from memory and subtracting it from the raw spectrum value before writing it to the buffer.

The repeating loop performs a similar action as the skipping loop, but repeats pixels by writing them to the buffer twice instead of skipping them. The repeated pixel array is a boolean array with values of true for “repeated pixels” in figure 3.16 and values of false for “regular pixels.” Note that the “skipped pixels” are not represented in this array since they have been skipped by the previous loop, and the length of the repeated pixel array is therefore less than the number of pixels in the raw spectrum (it is equal to the number of skipped pixels subtracted from number of pixels in the raw spectrum, which is  $512 - 7 = 505$  in the case of the example spectrum in figure 3.16). The pixel repeating loop runs in the 160 MHz domain using a state-machine architecture. State 1 reads the pixels from the previous loop and the corresponding value from the repeated pixel array and writes the pixel to the output buffer. If the repeated pixel value is true, then it enters state 2 which writes the same pixel to the output buffer a second time without reading a new pixel from the input buffer. Since the number of skipped pixels is necessarily equal to the number of repeated pixels, the number of clock cycles for which this loop does not read data from the input buffer will also equal the number of clock cycles for which the previous loop does not write data to same buffer. Therefore, by structuring the data flow this way, we have ensured that the buffers will not overflow during the pixel skipping/repeating steps (which would cause the final output spectrum to be corrupted).

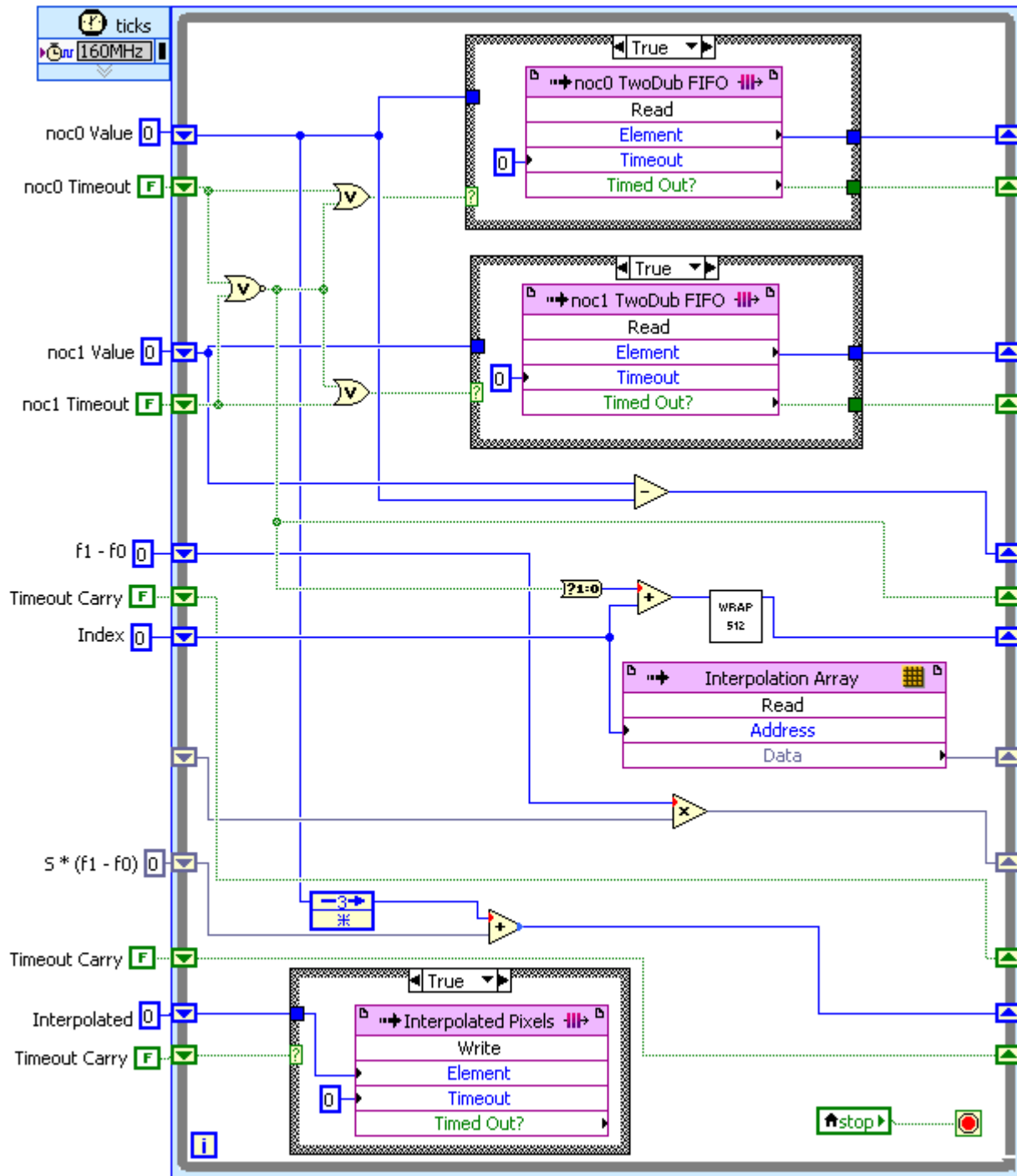
### 3.3.5 Interpolation Loop

This loop also runs in the 160 MHz domain and performs the linear interpolation to resample the spectrum so that it is evenly sampled in  $k$ -space. By performing the pixel skipping and repeating in the previous two loops, we have greatly simplified this algorithm by eliminating the need for any consideration of index shifts as described in section 3.2.2. This loop simply waits until both of the buffers are populated with data, then reads one value from each. It also reads the appropriate value from the interpolation array, which is stored in memory. The linear interpolation is then calculated using equation (3.5) where the subtraction is between the two values that were read from the input buffer. The LabVIEW FPGA implementation is shown in figure 3.17, and the MATLAB equivalent implementation is shown for clarity in figure A.4.

The majority of the complexity in the LabVIEW implementation of this calculation arises from the need for multiple pipelining stages. Because the SCTL performs certain operations that are costly in execution time (namely the multiplication calculation), it is impossible for equation (3.5) to be calculated in a single 160 MHz clock cycle. Therefore, information is pipelined using shift registers before and after the multiplication is performed. Timeout information is propagated through the pipeline stages so that only valid interpolated numbers are written to the output buffer.

### 3.3.6 Pixel Rebundling Loop

At this point in the data flow, the spectrum has been resampled via linear interpolation and written to a FIFO buffer in the 160 MHz clock domain. The next step is to calculate the FFT of the spectrum, however the LabVIEW FPGA Express VI that is used to calculate the FFT must run at a lower clock rate in order to function properly. As of the 2014 update to the LabVIEW FPGA Module, the FFT Express VI supports a “Single Channel, Multiple Samples” operating mode which accepts data in chunks rather than one point at a time. This allows for the VI to run in the original 40 MHz clock domain with four input (and output) pixels per clock cycle. Therefore, we must reverse the “unbundling” process from section 3.3.3 and “rebundle” the pixels so that data flow

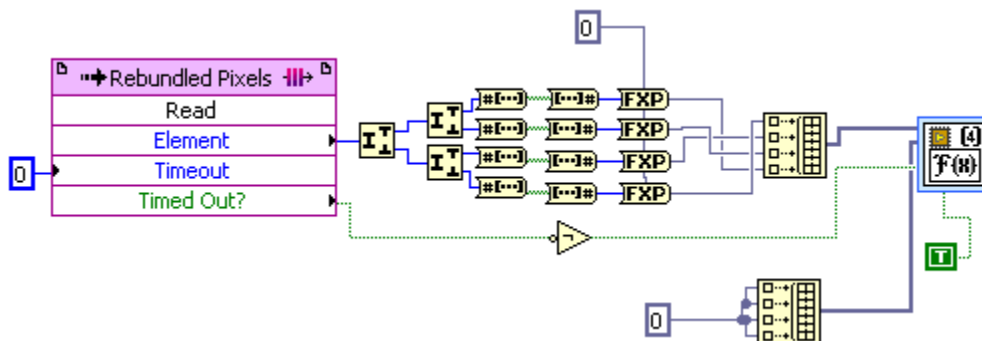


**Figure 3.17:** LabVIEW code used to perform the linear interpolation calculation for the resampling of the spectrum.

returns to the parallel “4-tap” configuration that we originally received from the Basler camera. The pixel rebundling loop effectively performs the reverse of the unbundling loop, using a state machine architecture to collect one pixel per clock cycle in the 160 MHz domain and write four pixels simultaneously to the output buffer as a 64-bit integer on every fourth clock cycle.

### 3.3.7 FFT and Depth Tracking Loop

As mentioned above, the FFT is done in the 40 MHz clock domain using the “Single Channel, Multiple Samples” mode of the FFT Express VI to process four consecutive FFT points in parallel. The input buffer streams unsigned 64-bit integers, from which the four pixels are extracted and typecast into signed 16-bit integers before being converted to a fixed-point representation that is compatible with the FFT VI. The imaginary portion of the input spectrum is set to zero. The LabVIEW code that handles this conversion is shown in figure 3.18.



**Figure 3.18:** LabVIEW code used to read the resampled spectrum four points at a time and convert to the correct format before being input to the FFT VI.

The FFT VI also outputs the transform of the spectrum (called an “A-line”) four pixels at a time. The sum of squares of the imaginary and real components is calculated to obtain an absolute value. Since the brightest pixel of the output A-line is used to calculate the depth measurement, the brightest of the four pixels is fed into a VI that tracks the brightest pixel over the entire A-line. Once a full 256-point A-line has been fed through the brightest pixel VI, it writes the index value of the brightest pixel to a register that can be read from other loops on the FPGA. Because many

ICI measurements come back “dark,” wherein there is no observable interface and the resulting A-line is effectively composed of noise, a threshold is applied to ensure that the brightest pixel is “bright enough.” This threshold is configurable by the user on the program’s front panel, since the optimal threshold value will vary depending on the application. A “binning” algorithm is also applied, which groups a number of consecutive frames defined by the “bin size” together such that the brightest pixel is extracted from a group of ICI measurements rather than every single A-line. This is also a precaution against tracking measurements that arise from dark lines.

The FFT and depth tracking loop also provides the option to stream data back to the host VI for cases when the user desires the ability to view or save the original A-lines. A DMA FIFO buffer is used for this streaming, and the actual data that is streamed can be configured by the user. On the host VI, the data is read as quickly as possible in a dedicated loop because of the high rate at which the FPGA VI populates the buffer with data. Data is then either displayed on the screen or stored on the computer’s hard drive (or both) in a separate loop using LabVIEW’s Queue functionality, which passes information between loops on the host PC in effectively the same manner as the target-scoped FIFO buffers on the FPGA.

### **3.3.8 Feedback and DAC Loops**

The output of the previous loop was a single value written to an internal register representing the index of the brightest pixel in the FFT of the ICI spectrum. The way in which this value is used depends on the application and the hardware that the FPGA will be interacting with. In the case of autofocus experiments with Aerotech’s A3200 hardware, a feedback algorithm is integrated in the stage controller, and the FPGA therefore must only output a voltage that scales with the measured depth. The digital-to-analog (DAC) hardware described in section 3.4.1 is used to generate this voltage since the I/O hardware connected to the FPGA only provides digital communication capabilities. The DAC circuit uses an MCP4725 to regulate the output voltage using the I<sup>2</sup>C serial protocol. Since the time taken to perform a full I<sup>2</sup>C write sequence is comparable to the time for

a full ICI calculation to be performed, the DAC voltage is only updated when the output value changes in order to minimize additional latency.

For other applications, there is no on-board feedback processing built in to the system hardware, and the FPGA must output a control voltage for a particular system parameter. An example is laser power modulation for keyhole welding control. In such systems, the feedback algorithm must be programmed on the FPGA. A separate feedback loop exists to handle this feedback processing using LabVIEW's built-in PID controller VI. The loop reads the the depth tracking register that is written to by the brightest pixel tracking SCTL and outputs an adjustment to the current laser power, which is written to another register for the DAC loop to handle. The PID gains, setpoint, and output constraints are all configurable on the front panel so that they can be tuned to fit any particular process.

### **3.4 Feedback Signal Hardware**

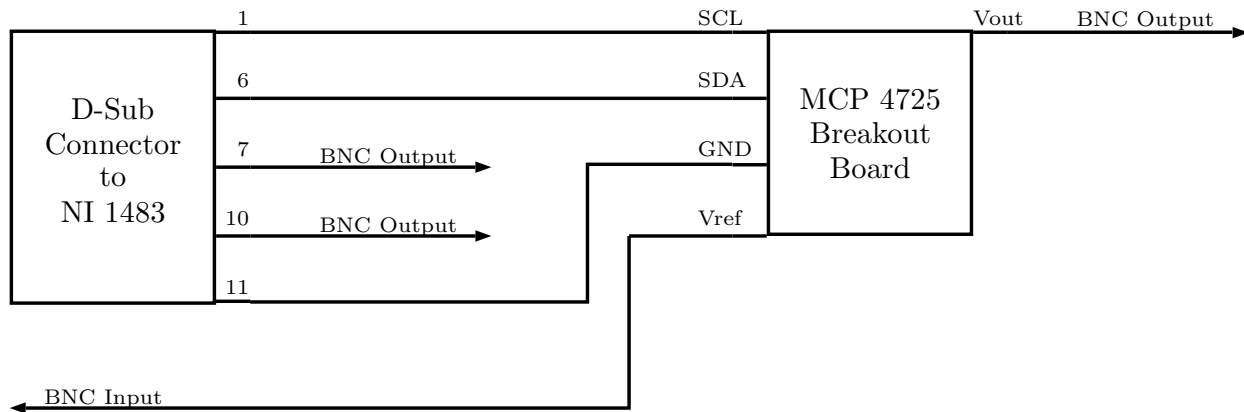
In order to complete a feedback loop with the ICI depth calculated using the FPGA system, an appropriate method of generating an output signal must be developed. The NI 1483 frame grabber card includes four digital TTL I/O lines in the general-purpose 15-pin D-Sub connector and no analog I/O lines. In order to have the ability to output an analog voltage to act as a control signal for the autofocus and welding control applications described in chapter 4, a digital-to-analog converter (DAC) must be used. Additionally, because the desired voltage output range varies for different applications, it is necessary for the system to be capable of amplifying and shifting the output voltage. This is done using an operational amplifier (Op Amp). In this section, I will briefly describe the two circuits that I designed to generate output feedback voltages without introducing a parasitic amount of latency that nullifies the advantages offered by the use of an FPGA, and I will include schematics to show their basic functionality.

### 3.4.1 DAC Circuit

DACs work by reading digital information using a serial communication protocol and updating an output voltage every time an instruction to do so is received. Therefore, the use of a DAC introduces an additional amount of latency in the system that is primarily dictated by the number of bits that constitute a write instruction and the clock rate of the serial communication. The maximum serial clock rate achievable is given by the settling time of the digital outputs on the FPGA, which is listed in the manual for the NI 1483[75] and confirmed experimentally to be approximately 500 ns. We are therefore limited to a serial clock cycle of 1  $\mu$ s (for a full rise and fall of the clock signal) or longer.

The I<sup>2</sup>C serial protocol, developed by Philips Semiconductors, offers a “fast mode” that allows for data transfer at rates up to 400 kHz, or a 2.5  $\mu$ s clock cycle[76]. Additionally, it only uses two wires (one for the clock signal and one for the data transfer), which allows two of the TTL I/O lines on the FPGA frame grabber card to remain unoccupied and available for triggering. Furthermore, the popularity of the I<sup>2</sup>C protocol makes it a common choice for many devices. For this application, I chose to use the 12-bit MCP4725 DAC with a reference voltage of 5 V. A complete serial write to the MCP4725 is made up of 22 bits, 12 for the voltage value and 10 for various addressing and acknowledgment bits required by the I<sup>2</sup>C protocol[77].

In order to make the DAC connection convenient and compatible with existing lab equipment, I built the MCP4725 breakout board from SparkFun Electronics into an electronics box that breaks out the D-Sub connector from the FPGA framegrabber card to BNC connectors, two of which pass through the DAC. This allows easy access to both digital and analog communication with the FPGA and convenient connectivity with existing lab equipment. The wiring diagram for the FPGA I/O breakout box is shown in figure 3.19.



**Figure 3.19:** Wiring diagram for the FPGA I/O breakout box, including the MCP4725 DAC.

### 3.4.2 Amplifier Circuit

For autofocus, it is sufficient to simply provide a 0 to 5 V signal, which corresponds to the measured ICI depth, to the Aerotech controller, which has its own built-in feedback processor. However, the same does not hold true for welding depth control. The IPG kW laser used in the Macro system does not have any control logic built-in, but instead accepts a 0 to 10 V signal that controls the laser's output power. Because the DAC used to generate analog output voltages with the FPGA only operates with reference voltages up to 5.5 V, additional hardware must be used to scale the voltage output so that it is within the appropriate range. Additionally, it is not desirable to operate the laser at voltages under 2 V, as the output is unreliable and does not scale linearly with voltage in that range. Therefore, it is also desirable to shift the input voltage so that a 0-5 V output swing from the DAC is converted to a 2-10 V input swing to the laser.

Amplifying and shifting a voltage signal is most easily and effectively done using an operational amplifier (Op Amp). Op amps are analog devices that amplify a differential input signal to produce a single-ended output signal. They can be used in either an inverting or a non-inverting configuration, and can also be provided a reference voltage to shift the output by a constant amount. There are many Op Amps available that would be acceptable for this application, so I chose the LM741 due to its performance as well as its availability and popularity. The most important con-

sideration for the performance of the Op Amp in our FPGA feedback system is the timing. The datasheet for the LM741[78] lists a transient response of  $0.3 \mu\text{s}$  and a slew rate of  $0.5 \text{ V}/\mu\text{s}$ . For a full swing of the target  $8 \text{ V}$  range, this corresponds to an approximate delay of  $16 \mu\text{s}$ . Since an  $8 \text{ V}$  swing represents an instantaneous change in the measured depth over the entire field-of-view of the ICI system, this number represents an absolute maximum delay, and changes in output voltage that are less than  $1 \text{ V}$  are more likely. Therefore, the delays that should be introduced by this Op Amp are in the low single-digit  $\mu\text{s}$  range, which are a negligible contribution to the total latency when compared to the DAC and the FFT processing steps.

The Op Amp circuit that I used to achieve the desired output voltage is shown in figure 3.20. The LM741 is powered using a supply voltage of  $\pm 12 \text{ V}$  and, in this design, a reference voltage of  $5 \text{ V}$ . These voltages were chosen because they are available from a standard PC power supply with an Advanced Technology Extended (ATX) breakout board, both of which are already used in our lab and therefore easily available. Using the standard Op Amp equations for a non-inverting and level-shifting configuration[79], with the resistor values shown in figure 3.20, the amplifier gain is given by:

$$A = \frac{R_4}{R_1} \times \frac{R_1 + R_2}{R_3 + R_4} \quad (3.7)$$

$$= \frac{39\text{k}}{10\text{k}} \times \frac{10\text{k} + 10\text{k}}{10\text{k} + 39\text{k}} \quad (3.8)$$

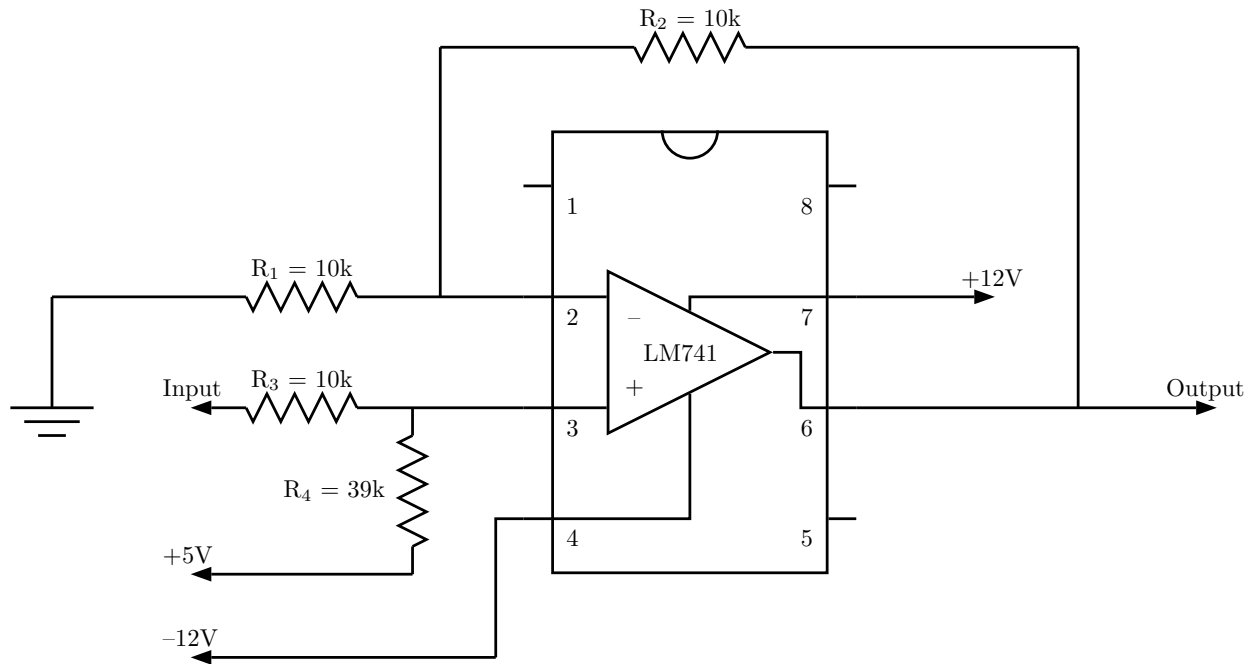
$$= 1.59 \quad (3.9)$$

and the shift is given by the reference voltage gain:

$$V_{\text{offset}} = 5\text{V} \times \left( \frac{R_2 + R_1}{R_1} \times \frac{R_3}{R_3 + R_4} \right) \quad (3.10)$$

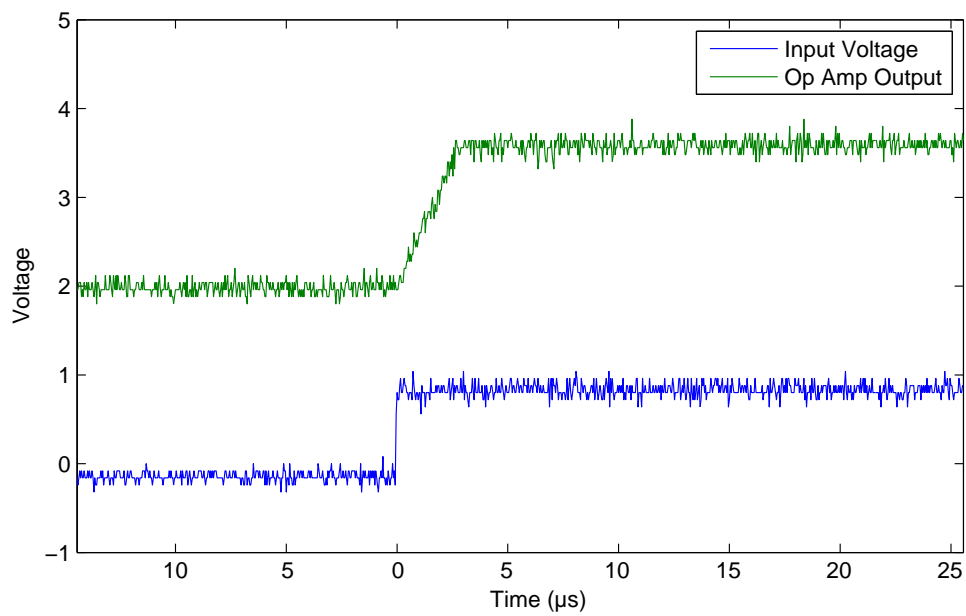
$$= 5\text{V} \times \left( \frac{10\text{k} + 10\text{k}}{10\text{k}} \times \frac{10\text{k}}{10\text{k} + 39\text{k}} \right) \quad (3.11)$$

$$= 2.04 \quad (3.12)$$



**Figure 3.20:** Schematic of the operational amplifier circuit used to convert a 0-5 V output from the DAC to a 2-10 V input for controlling the laser power on the Macro system.

The performance of the amplifier was tested experimentally with a 1 V step input. The response is shown in figure 3.21, which confirms that the amplifier circuit achieves the desired gain of  $\sim 1.6$ , desired voltage shift of  $\sim 2$  V, and a low latency contribution. The slight shift that places the input signal in figure 3.21 below 0 V is due so a small offset in the function generator used to generate the input signal.



**Figure 3.21:** Step response of the amplifier circuit. The desired gain of  $\sim 1.6$  V, shift of  $\sim 2$  V, and delay of less than  $5 \mu\text{s}$  are visible in the response.

# Chapter 4 Application and Results

To evaluate the performance of the FPGA system described in chapter 3, tests were performed to assess its performance by taking direct measurements of its latency. The system was also tested in realistic industrial laser processing scenarios. In this chapter, I will start by presenting a direct comparison of the latency of the FPGA system to that of a PC running the same data processing. This measurement shows that the FPGA is capable of reducing the latency from  $3600\ \mu\text{s}$  to  $54\ \mu\text{s}$  when compared to the PC. I will then present FPGA ICI applied in autofocus, wherein the height of the beam delivery head is automatically adjusted using FPGA ICI control to keep the machining beam in focus. Finally, I will discuss some preliminary demonstrations of welding depth control, which was achieved by modulating the power of the machining laser so that the depth to which it penetrates into the sample is controlled to achieve a set target depth.

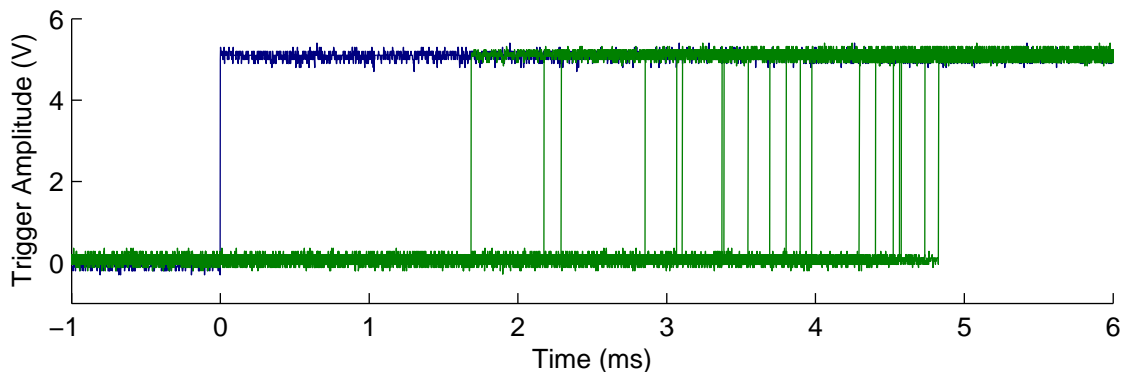
## 4.1 Latency Assessment

In order to obtain a quantitative assessment of the latency improvement obtained by switching from PC to FPGA processing, a direct measurement of the latency was performed for the two systems. To replicate an accurate measurement of latency in a realistic ICI processing scenario, the latency is defined as the time the system takes to detect and respond to a change in the ICI signal.

The experimental setup for measuring the latency in this manner on the PC is as follows. Two interference spectra were collected using the LD-600 setup described in section 3.1 connected to the Macro processing station at two different depths. The two spectra were loaded into memory, and a LabVIEW VI that continuously processes a single ICI spectrum by performing DC subtraction, interpolation, FFT, and brightest pixel calculation was then executed. The decision of which spectrum to load from memory was controlled by a digital voltage input using a NI DAQ-MX

system. An input trigger signal was generated manually that was interpreted by the PC as an instruction to stop loading the first spectrum to be processed and instead load the second. The VI then generated an output voltage using the DAQ-MX system depending on which depth it extracted from the most recent ICI processing cycle. The latency could therefore be measured by triggering an oscilloscope with the input to the PC and measuring the output signal from the PC. The time difference between the input and output signals would then serve as a direct measurement of the time that it took the PC, using a CPU for ICI processing, to detect and respond to an external change.

The results for the PC ICI latency measurement are shown in figure 4.1. The measurement was repeated 20 times by triggering the PC ICI manually at arbitrary moments in time to simulate the quasi-random behavior that is observed in laser processing. An average and standard deviation calculation of the response time for the PC gives an estimate of the PC latency time as  $3.6 \pm 0.9$  ms.

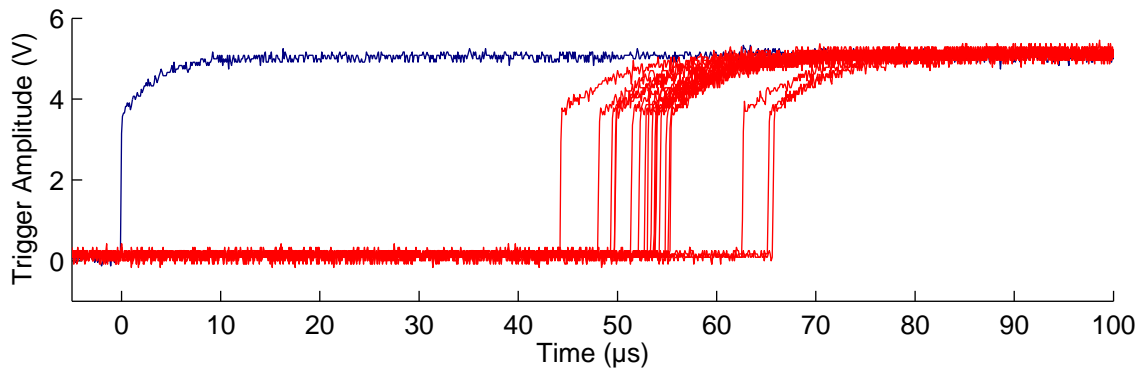


**Figure 4.1:** Direct latency measurement of ICI processing on a PC. The blue line represents the trigger (input) signal and the green lines represent the response (output) signals over 20 different runs. The average latency for the PC is  $3.6 \pm 0.9$  ms.

An equivalent experiment was performed with the FPGA ICI system to obtain a direct comparison to the PC's performance. A modified FPGA VI that has the capability to load a spectrum from its block memory rather than reading from the camera was used. The VI contains a switch on the front panel that allows the user to switch between loading the spectrum from memory and reading a new spectrum from the camera. Whenever a new spectrum is read from the camera, it

replaces the spectrum that is stored in memory so that the most recently read spectrum is used when the program is switched to reading from memory. Since timing on the FPGA is strictly enforced using SCTLs in MHz clock domains, the FPGA was used to generate both the input and output trigger signals; the “input” signal is generated on one of the FPGA’s digital outputs and is modulated whenever the user switches the processing mode on the front panel, and the “output” signal is generated on another digital output and is modulated when the ICI depth tracking crosses a preset threshold value. With this setup, a latency measurement is obtained by placing a physical interface under the imaging beam, switching to memory reading mode, moving the interface such that it crosses the threshold, and then switching back to camera reading mode. The FPGA VI reacts when it detects the new interface, and then outputs a trigger signal when it has detected a change in the tracked depth.

The results for 20 different FPGA latency measurements are shown in figure 4.2. The average and standard deviation are calculated to be  $54 \pm 5 \mu\text{s}$ , which constitutes a significant improvement over the PC latency ( $3600 \mu\text{s}$ ), as well as a dramatic reduction in uncertainty. This indicates that processing individual ICI frames in real-time is much faster than the equivalent processing on a PC CPU, and processing times are more predictable and vary less than they do on the PC.



**Figure 4.2:** Direct latency measurement of ICI processing on the FPGA. The blue line represents the trigger (input) signal and the red lines represent the response (output) signals over 20 different runs. The average latency for the FPGA is  $54 \pm 5 \mu\text{s}$ .

The value of  $\sim 50 \mu\text{s}$  for the FPGA latency agrees with expectations. We can calculate a

“theoretical” latency value by adding the number of clock cycles that we expect the FPGA to take to process a complete spectrum, then multiplying by the clock rate. For this experiment, the FFT VI has a latency of 1349 cycles, and we must also account for the time it takes for a full A-line to be run through the brightest pixel tracking VI. Considering just these two largest contributions, we obtain  $1349 + 512 = 1861$ , which corresponds to  $47 \mu\text{s}$  at 25 ns per cycle. This “best-case” estimate verifies that there isn’t a significant amount of latency in the system from factors other than the actual ICI processing. The  $5 \mu\text{s}$  of uncertainty in the FPGA latency is most likely due to the fact that the trigger signal is randomly generated, and could therefore occur at any point in the middle of the readoff of a spectrum. If the trigger signal is received when a spectrum is being read, then we must add the time that the FPGA takes to finish reading that spectrum to the response time, which manifests as an additional amount of latency that is different for each iteration of the experiment.

## 4.2 Autofocus

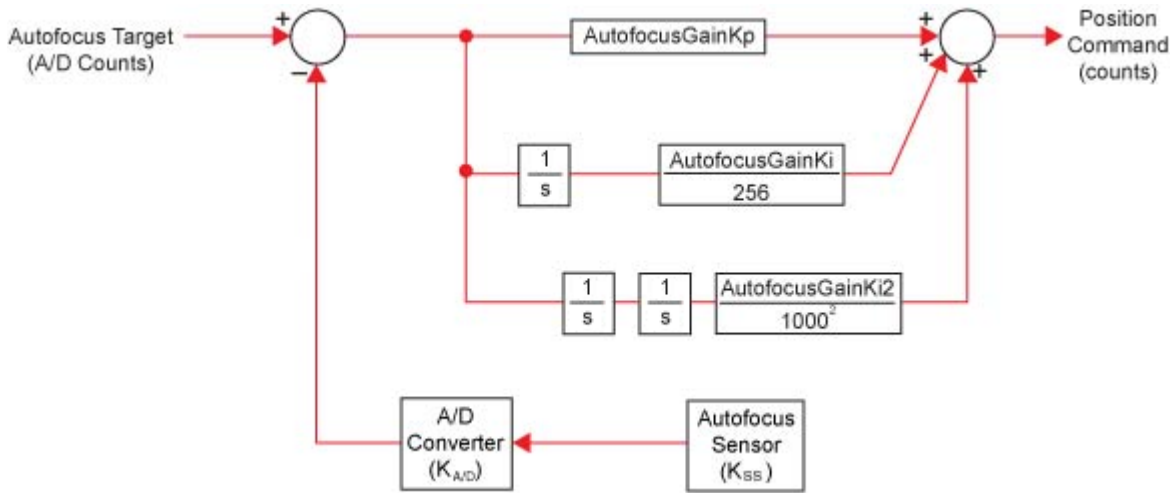
While a direct measurement of ICI processing latency is a useful measurement of the performance of the FPGA ICI system, it is also important to test its applicability in real-life laser machining processes. The effect of the reduced latency and increased determinism offered by the FPGA is most beneficial in applications where it is desirable for a system to respond to changes in the machining process as quickly as possible. One such application is autofocus of the laser machining head. While industrial laser beams carry within them an impressive amount of energy that can be used to heat, melt, and vaporize most materials, they are most efficient at doing so when the target is within the Rayleigh range of the processing beam, which, for the Macro laser processing station described in section 3.1, is approximately 4 mm. Not only is the power density maximized when the energy of the beam is distributed over a minimal area on the target’s surface, but a properly focused beam produces a smaller heat affected zone, resulting in cleaner cuts[80].

While it is possible to program the morphology of the part into the guidance computer so that

the machining head is guided along the surface, this does not account for discrepancies that can arise due to alignment errors (such as a misplaced part on the production line resulting in a tilt or translation relative to the program's expectations), inconsistencies in the surface structure of the part, or other unexpected factors. There is therefore a need for a closed-loop feedback control system that can track changes in the focal distance and adjust the height of the machining head on-the-fly. There is a variety of existing techniques that attempt to achieve laser autofocusing for a range of applications[81, 82, 83]; however, they are not based on direct inline measurements of the laser penetration depth. The ICI imaging beam travels through the same focusing optics as the machining beam, and if the reference arm of the ICI system is adjusted to be equal to the focusing length of the machining laser, then the zero-delay point of the ICI spectrum represents the point where the interface is in focus. ICI therefore provides a very simple but effective method of determining whether or not the machining surface is within the Rayleigh range of the processing laser beam.

To test autofocus, I wired the FPGA to the Aerotech A3200 motion control hardware described in section 3.1 using the DAC interface as described in section 3.4.1. The A3200 system includes a built-in feedback loop that is used to control the position of any of the connected motion stages. The FPGA outputs an analog voltage through the DAC that scales linearly with the measured depth. The voltage is connected to one of the analog inputs on the A3200 system, and that input is specified in the Aerotech suite's autofocus loop configuration file. An autofocus target is also specified, such that the difference between the input voltage and the target voltage represents the feedback error. A polarity parameter specifies whether a positive or negative change in the stage position should be used to compensate for a positive feedback error, and a deadband parameter defines a small input voltage range relative to the target voltage over which no feedback action will occur. Three feedback gains are specified: a proportional gain  $K_p$  and two integral gains  $K_{i1}$  and  $K_{i2}$ , although the proportional gain was left equal to zero for all experiments in accordance with the instructions provided with the Aerotech documentation. Figure 4.3 shows a block diagram representation of the role that these gain parameters play in the autofocus algorithm. Since this

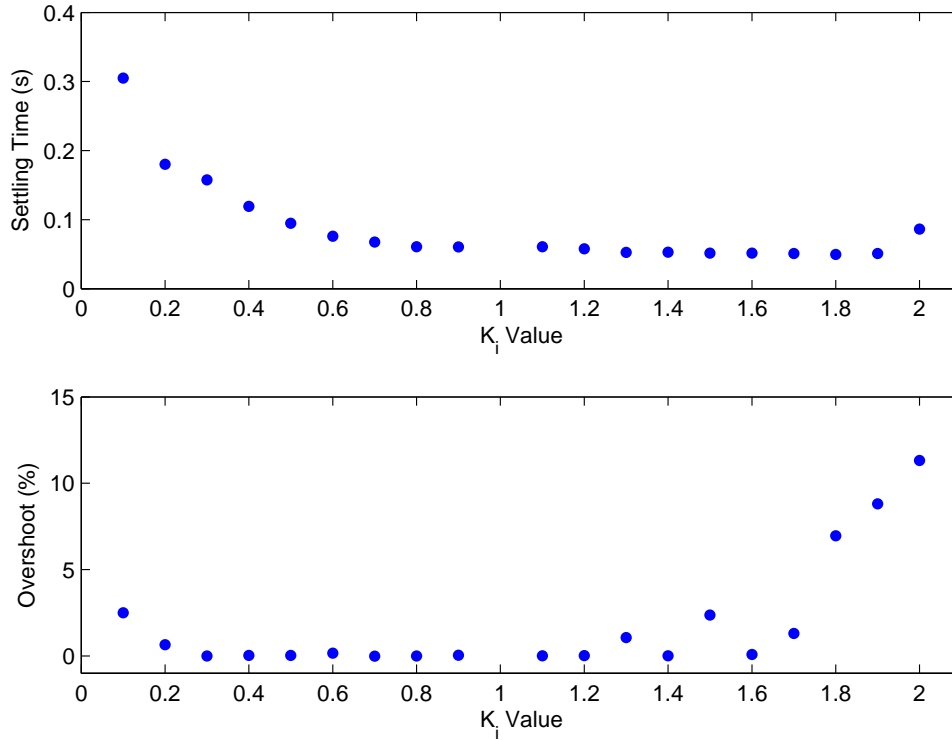
autofocus loop is handled entirely by the Aetorech system, it was treated as a back-box for the purposes of my experiments.



**Figure 4.3:** Block diagram representation of the Aerotech A3200 feedback loop. The “Autofocus Sensor” block represents the depth signal that comes from the FPGA. Adapted from [84].

Autofocus feedback parameter tuning is performed experimentally by varying the gains individually until a desired level of stability is achieved. As per the instructions in the A3200 help documentation[84],  $K_p$  and  $K_{i2}$  were both set to zero, and  $K_i$  was adjusted to find an optimal value. A linear scan across a step was used to test the stability of the feedback loop for various  $K_i$  values since it effectively acts as a step response and provides two metrics to test system stability, settling time and percent overshoot. These quantities are defined in section 2.3. The step was approximately 0.5 mm thick, and the laser was scanned at a speed of 50 mm/s. Figure 4.4 shows the result of the  $K_i$  tuning. A range of values between approximately 0.8 and 1.2 provide both minimal overshoot and settling time. In proceeding with further experiments, a  $K_i$  value of 0.8 was chosen since it lies in this range and, as described in section 2.3, a smaller  $K_i$  increases system stability.

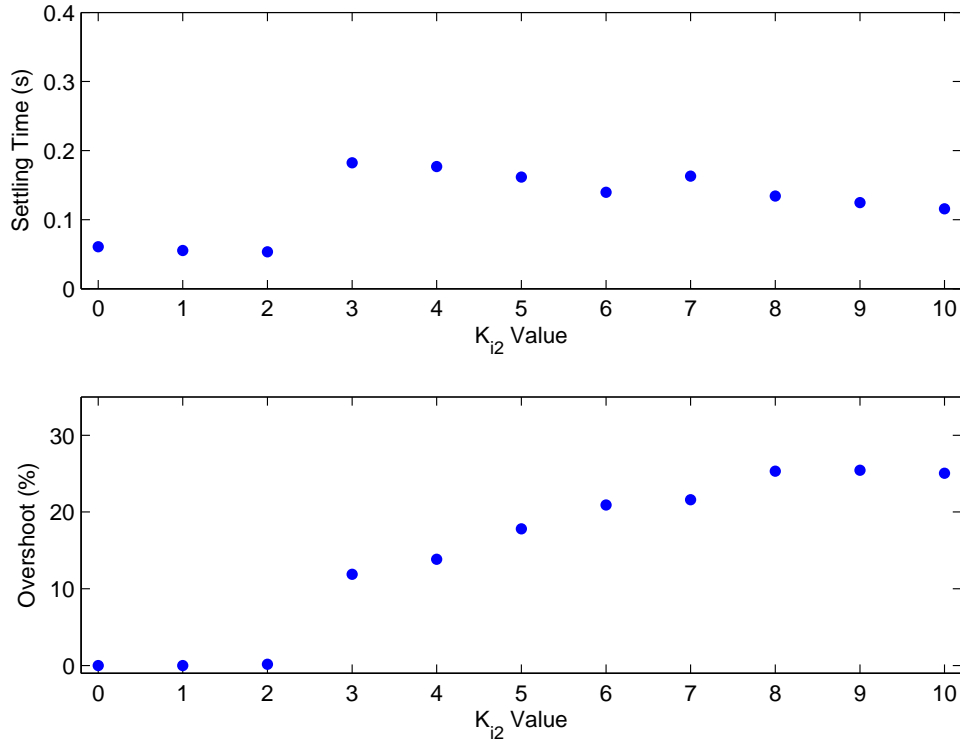
$K_{i2}$  was tuned by fixing  $K_i$  at the chosen value of 0.8 and varying the value of  $K_{i2}$  until the performance of the system was optimized. Tuning using a step response in the same way as the  $K_i$  tuning produced the results in figure 4.5. From these results, it seems that a value of 2 is the optimal value for  $K_{i2}$ , with a settling time of about 50 ms and nearly 0% overshoot. The settling



**Figure 4.4:** Settling time (top) and percent overshoot (bottom) of the autofocus system as the  $K_i$  gain is varied with the  $K_p$  and  $K_{i2}$  gains set to zero.

time is much larger than the latency of the FPGA ICI system because the mechanical motion of the stage serves as the bottleneck.

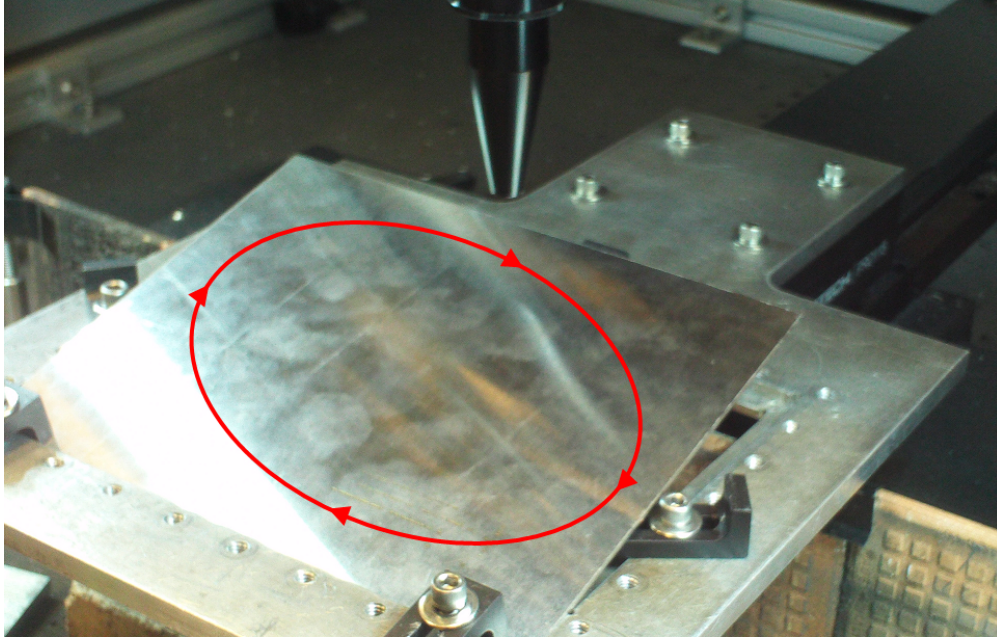
The above result is only representative of the step response of the system. As a more realistic test of the performance of the feedback system, autofocus was performed using a circular motion over a flat, tilted metal plate which causes a smooth sinusoidal motion of the interface in the  $z$ -axis. For this experiment, the laser was scanned at 120 mm/s in a 60 mm radius circle. This scheme is illustrated in figure 4.6. In a real-life laser machining scenario, the most important quality of the autofocus system is its ability to minimize the deviation of the machining interface from the focal position of the beam. Therefore, the RMS following error was used as a quantitative assessment of the autofocus system. The results are displayed in figure 4.7 and show that, for this type of motion, following error decreases when  $K_{i2}$  increases, plateauing around 4. The difference in optimal values between the step function experiment and the inclined plate experiment suggests



**Figure 4.5:** Settling time (top) and percent overshoot (bottom) of the autofocus system as the  $K_{i2}$  gain is varied with  $K_p = 0$  and  $K_i = 0.8$ .

that the optimal values of the autofocus loop gains depend on the particular application and type of performance desired. In the case of laser cutting with the IPG kW laser in our lab, the beam's Rayleigh range is 4 mm, and the achieved RMS error is therefore well within the acceptable range to keep the beam in focus.

The deadband value is also an important parameter in optimizing the system, especially with a low-latency sensor. A deadband value that is too large will not allow the system to respond to small changes in height, and a deadband value that is too low will cause the system to respond to noise in the voltage signal. Since the output voltage of the FPGA is generated from a digital number and is therefore discretized, we can calculate a theoretical deadband value by calculating the step size of the output voltage signal based on the fact that a 256-point FFT signal is mapped to a 0-5 V signal:



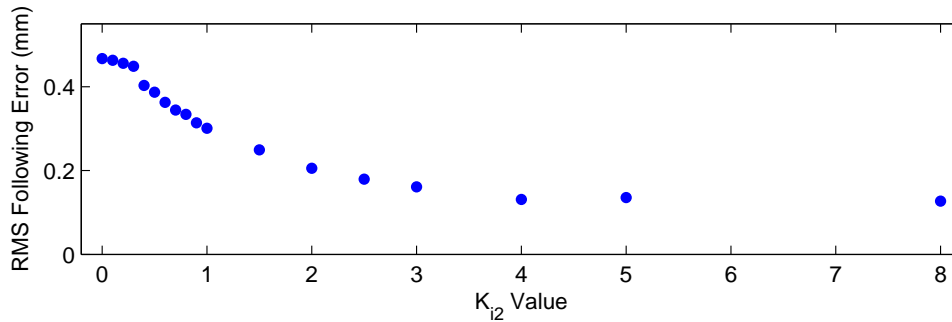
**Figure 4.6:** Outline of the scanning path across the inclined metal plate used to simulate a sinusoidally moving interface for autofocus.

$$\text{Deadband} = \frac{5}{256} = 0.0195V \quad (4.1)$$

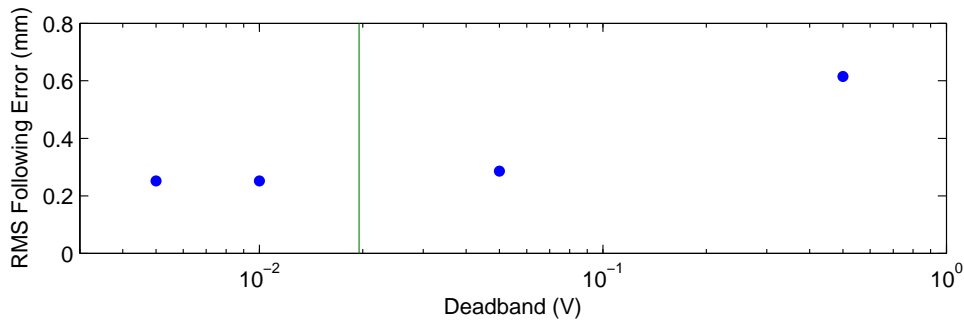
The RMS following error for several deadband values is shown in figure 4.8 along with the theoretical value for comparison. As expected, the improvement seen by decreasing the deadband value starts to plateau around the theoretical value. Ultimately, with settings of  $K_i = 0.8$ ,  $K_{i2} = 8.0$ , and  $\text{Deadband} = 0.005$ , a minimal RMS following error of 0.05 mm was achieved over a range of motion of 120 mm. This is well within the  $\sim 4$  mm Rayleigh range of the IPG 1 kW machining laser and therefore is a reliable candidate for autofocus control in real industrial machining settings.

As a final assessment of the autofocus loop, the latency was artificially increased by varying the bin size of the FPGA ICI processing. Since increasing the bin size forces the FPGA to wait for additional frames to be processed before any interfaces are tracked, we can estimate the actual amount of latency that is artificially added by increasing the bin size. Since the camera acquires frames using an integration time of  $3.6 \mu s$  (and the camera readoff is less than the integration time),

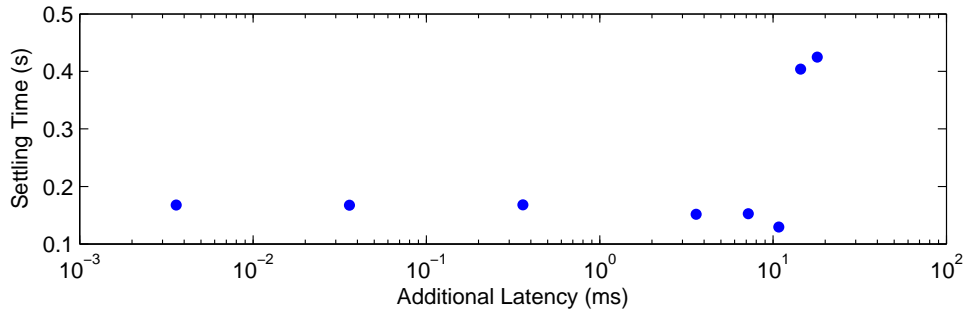
we can estimate that each additional bin adds  $3.6 \mu\text{s}$  of latency. Using this estimation, the settling time was measured for various artificial latency values, and the results are shown in figure 4.9. It is obvious that there is a dramatic degradation in performance once the latency hits the 10 ms point. Since this threshold lies between the measured latency values for FPGA and PC ICI as described in section 4.1, this result highlights a clear advantage to using FPGAs for ICI feedback control rather than traditional methods.



**Figure 4.7:** RMS following errors for autofocus over an inclined plate for various  $K_{i2}$  values with  $K_p = 0$  and  $K_i = 0.8$ .



**Figure 4.8:** RMS following errors for several deadband values. The green line represents the theoretical optimal deadband value of 0.0195 V.



**Figure 4.9:** Measured settling time with artificial latency introduced by increasing the bin size.

### 4.3 Welding Depth Control

The desire in industry to achieve closed-loop control of laser penetration depth during welding is highlighted by the active level of ongoing research in the field. A number of techniques exist that take advantage of various features of the welding process in order to achieve closed-loop control, such as spectral analysis of optical emission from the plasma[7, 85] and observation of the full penetration hole[5]. While these processes have been shown to work, they are all based on features that are indirectly related to the penetration depth of the laser rather than direct, inline measurement of the laser penetration depth itself. This makes them sensitive to optical, thermal, and/or electrical properties of the material, and they therefore require an extensive amount of calibration for each different application. ICI provides a direct depth measurement using an apparatus that is independent of the material being processed. It is important to note that while welding depth control has previously been demonstrated using ICI-based control with settling times of approximately 50 ms being achieved[22], a PC (rather than dedicated signal processing hardware such as an FPGA) was used for ICI data processing.

While low-latency autofocus control is an important demonstration of FPGA ICI’s applicability for deterministic feedback control in industrial settings, the system’s response time to a step input is limited by the slow motion of the mechanical stage relative to the latency of the FPGA. Another parameter of laser processing that can be modulated on timescales in the 100s of  $\mu$ s is the power

of the machining laser. Since  $100\ \mu\text{s}$  is much closer to the measured latency of the FPGA, laser power modulation for controlling the penetration depth in laser keyhole welding is an especially interesting application to investigate.

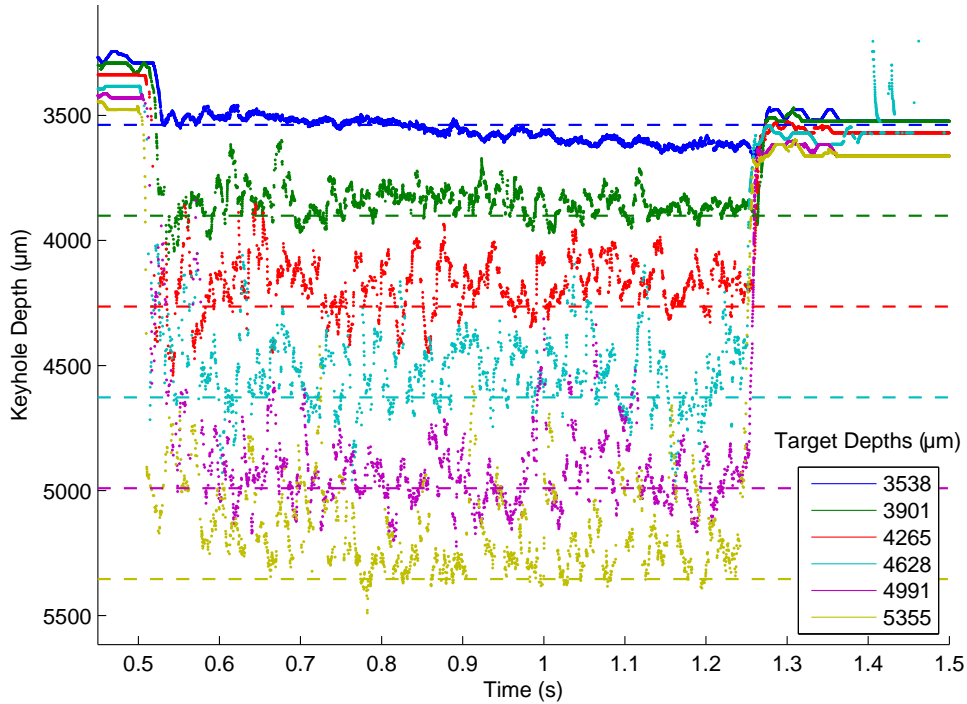
Welding control is done on the Macro processing station described in section 3.1. The IPG kW fiber laser can be set to run in an externally modulated mode wherein the output power is controlled with an analog voltage signal. This signal was generated using the DAC and Op Amp loops described in sections 3.4.2 and 3.4.1. The IPG laser does not have a built-in feedback controller as was the case with the Aerotech stage motion controller. The feedback control is therefore performed on the FPGA using the PID controller VI that is included with the LabVIEW FPGA Module. The PID controller allows for the three gains and the setpoint as well as constraints on the output signal to be configured by the user on the program's front panel. There is also a "reset button" that resets the controller's integral and differential errors. This reset button was wired to a digital signal that reset the controller every time the laser's power was turned on so that previous values of ICI depth from the time the processing laser was off did not affect the controller after the laser had been turned on. The output signal from the FPGA is then used to directly control the power of the processing laser by sending an analog voltage signal between 2 V and 10 V using the Op Amp circuit described in section 3.4.2.

The welding experiments in this section were performed by first moving the top surface of the part to a position within the ICI imaging's field of view. The absolute value of this top surface depth was not strictly controlled, however it is not necessary for it to be consistent between experiments, since the important information in any ICI measurement is derived from the depth relative to the top surface. State-of-the-art commercial quality-control ICI systems account for this fact by measuring the "top-surface reference" repeatedly during the weld. Since all of the experiments described in this thesis are feedback control experiments, the measurements are all compared to a setpoint value, and therefore the top surface location was not measured.

With the system arranged as above, several bead-on-plate welds were performed on a mild stainless steel sample with the system parameters set to achieve a variety of target depths. The scan

speed was set to 20 mm/s to match the previous welding control experiments[22]. As discussed in section 2.4, a bead-on-plate weld is a common method used to study laser keyhole welding in academic and research settings, wherein the weld is scanned linearly in one direction across a single, flat plate. Since only one part is used in bead-on-plate welding, no material joining occurs; however, the keyhole dynamics are similar in industrial lap, butt, and fillet laser welding geometries. A series of six bead-on-plate welds were performed over a range of target depths with the laser power modulated using the FPGA ICI feedback. The results of this are shown in figure 4.10. The keyhole depth measurements were acquired by streaming the ICI measurements to the host PC as the welds were being performed. Since the ICI data inside a weld keyhole is volatile[86], smoothing was applied to the datasets using a 10 ms moving average to obtain a clearer picture of the keyhole depths. We can see that the FPGA ICI system is able to reach and maintain a target depth over a range of depths. Note that there are still fluctuations in the depth signals despite the application of a smoothing function. These fluctuations can be traced to real fluctuations in the depth of the keyhole due to the dynamics of the welding process, as well as fluctuations caused by the control algorithm. Because an unoptimized PID feedback loop was used to perform the experiments, some instability in the system that results in signal oscillations is expected.

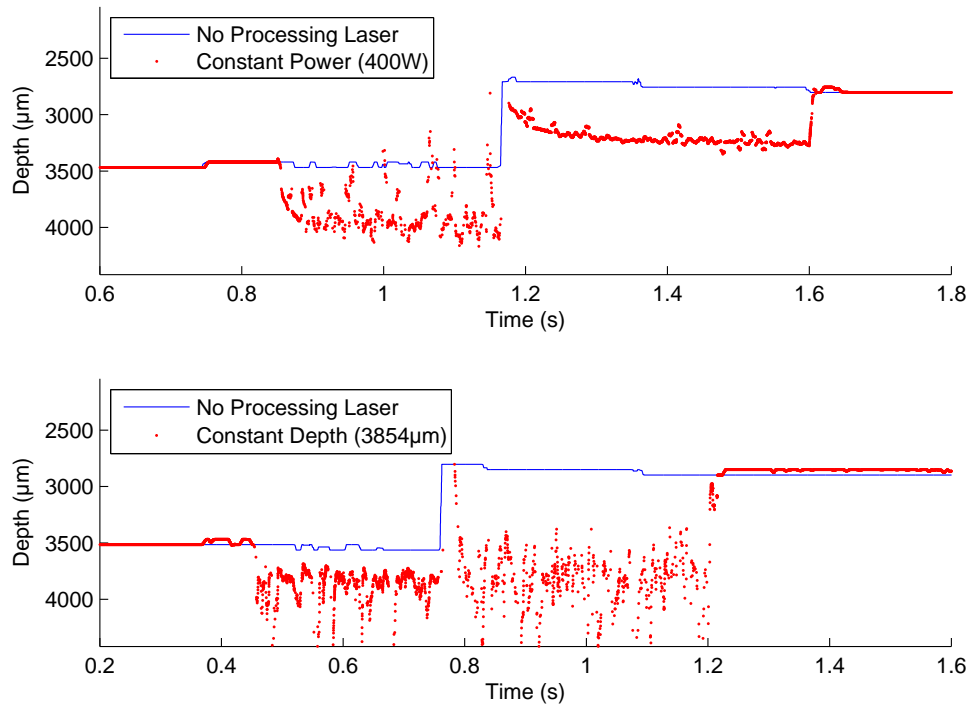
Another preliminary test of welding depth control using FPGA ICI is to examine its capability of maintaining a constant target depth when the surface morphology changes. This was done by welding over a step function with constant laser power, and then welding over a similar step with ICI feedback control set to achieve a target depth. The measured ICI depth profiles can then be compared to assess the quality of the two welds. These results are shown in figure 4.11. The blue curves were acquired by scanning over the step and acquiring ICI measurements with the processing laser off, and the red points were acquired over the same path with the laser on. The blue curves are superimposed on the plots of the red points in figure 4.11 in order to illustrate the magnitude of the step that was traversed; however, they do not represent the absolute position of the step in time because the two data sets were collected in separate runs. The ICI depth signals in figure 4.11 were smoothed using a 5 ms moving average algorithm. A lap weld configuration



**Figure 4.10:** Laser welding depth control in bead-on-plate welds to various target depths. The target depths were set in the welding control VI as brightest pixel points, and the corresponding depths in  $\mu\text{m}$  are shown in the legend. The target depths are outlined by the dashed lines.

was used in which a 0.7 mm piece of stainless steel rests on top of a thick bottom part, also made of stainless steel. For the first half of the scan, the beam only forms a bead-on-plate weld into the bottom part. It then hits the edge of the top part and welds into it as well. The top image shows the results for a constant laser power of 400 W. The keyhole initially forms in the bottom part and reaches a steady depth that is related to this power. Once it reaches the step caused by the edge of the top part, it penetrates into approximately the same amount of material. However, since the thickness of the top part is greater than the keyhole depth, only the top material is penetrated and the result is a failed weld. The bottom figure shows the same welding geometry but with FPGA ICI depth control active and set to a target depth point of 1500 (which corresponds to a real depth of  $3854 \mu\text{m}$ ). Once the processing laser reaches the step, it automatically adjusts the laser power so that both materials are penetrated, resulting in a successful weld. The average depths and standard

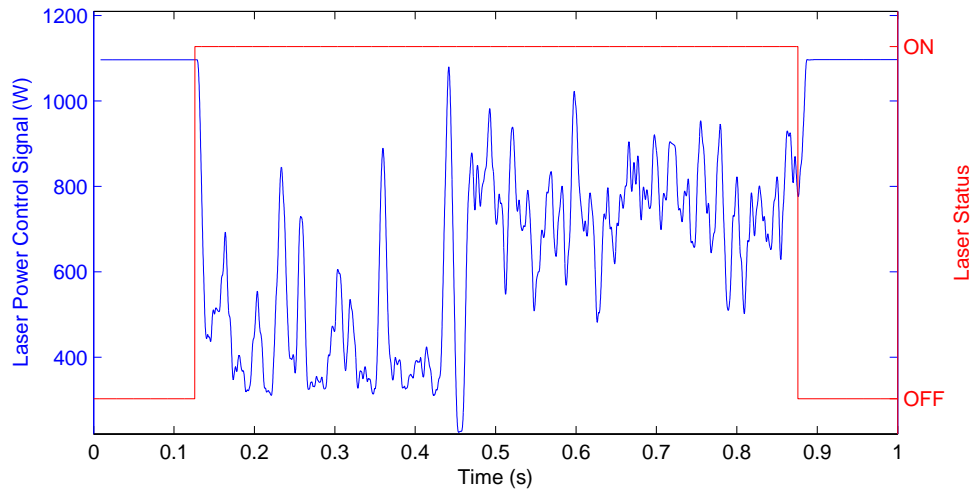
deviations measured before the step and after the system reaches quasi-stead-state in the controlled (bottom) weld depicted in figure 4.11 are  $3840 \pm 250 \mu\text{m}$  and  $3800 \pm 460 \mu\text{m}$ , respectively.



**Figure 4.11:** Lap weld with a step interface with constant laser power of 400 W (top) and constant target depth of 3854  $\mu\text{m}$  using FPGA ICI control (bottom).

The signal that modulates the power of the laser as well as the trigger that turns the laser on/off were also collected for the depth control experiment described above, and they are shown in figure 4.12. Note that the zero-points of the time axes in figures 4.11 and 4.12 do not correlate to each other because the former was collected using the FPGA host PC and the latter was collected using an oscilloscope; however, the step locations can be roughly correlated with each other by noting that the instant the laser is turned on in figure 4.12 is at approximately the same time as when the material penetration begins in figure 4.11. A moving average smoothing algorithm was applied to filter out some of the high-frequency oscillations in the power signal so that the average power would be more visible. The sudden increase in average laser power confirms that the FPGA ICI controller was able to adjust the laser power to compensate for the extra material in its path. A

short period of oscillation can also be seen where the step in laser power occurs in figure 4.12, indicating that there was an overshoot in the power adjustment that took a small amount of time to settle. This is further confirmation that the feedback loop was functioning, since the overshoot and settling time are characteristics of a PID control system as described in section 2.3.



**Figure 4.12:** Laser modulation (red) and power control signal (blue) for the controlled weld in figure 4.11.

The welding depth control setup is much more sensitive to parameter tuning and differences in the experimental setup than the depth autofocus. This is because the imaging beam is being reflected inside a volatile keyhole rather than a flat, static interface. Since finding an optimal parameter space is therefore a task that requires information about the particular parts and equipment that are to be used for welding and time spent tuning parameters for each individual application, the optimization of welding control parameters is a question for further research.

In summary, the FPGA-based ICI system has demonstrated by reducing the processing latency from  $3600 \mu\text{s}$  to  $54 \mu\text{s}$  that it can outperform the PC-based system in latency by orders of magnitude. This is illustrated by a comparison of the direct latency measurements of the two systems, which are  $3.6 \pm 0.9 \text{ ms}$  for the PC and  $54 \pm 5 \mu\text{s}$  for the FPGA. The applicability of FPGA-based ICI feedback technology has been tested in real industrial processes and shown to be functional. In autofocus, an RMS following error that is much less than the Rayleigh range of the processing

beam was achieved for a smooth motion pattern, and the settling time for an abrupt change in surface morphology has been shown to be dominated by the limitations of the mechanical motion stage. In welding depth control, the FPGA ICI feedback system was able to achieve and maintain a variety of target depths inside the material, and respond to a rapid change in the geometry of the surface material.

# Chapter 5 Conclusion and Future Work

In this thesis, I have presented a novel method of processing ICI data on dedicated hardware using an FPGA. I have described in a moderate amount of detail the process involved in designing and implementing an FPGA-powered ICI system for use in industrial settings, and have shown through experiment that it is capable of producing the desired results with a higher level of determinism than can be achieved with traditional PC-based processing methods. In this section, I will briefly describe some lines of investigation that could branch from my work and constitute the subject of future research in this area. Namely, I will describe some more sophisticated resampling and FFT methods that could replace the complex and cumbersome linear interpolation algorithm described in section 3.2.2. I will also discuss an alternate type of dedicated processing hardware known as application-specific integrated circuits and touch on their advantages and disadvantages when compared to FPGAs. Finally, I will mention some other laser manufacturing process parameters that were not the subject of investigation of this thesis but that could benefit from the type of low-latency feedback control offered by FPGA ICI.

## 5.1 Resampling and FT Methods

In section 3.2.2, I described an algorithm that was designed to perform a linear interpolation calculation to resample the spectrum collected by the ICI spectrometer so that it is linearly sampled in  $k$ -space. This is a necessary step in order for the FFT to correctly extract information about the relative optical path lengths of the sample and reference arms in the interferometer. The linear interpolation method was chosen because it provides a significant improvement in resolution (see figure 3.10) without introducing a parasitic amount of latency in the processing chain. While this method provides the optimal performance for the applications investigated in this thesis, there are

alternative methods available that could provide improvements in resolution and computational performance as the technology develops. Because these alternative methods could be important avenues of investigation for future development of this technology, I will briefly mention them in this section.

The most promising alternative is the so-called non-uniform FFT (NUFFT). The NUFFT is a relatively recent FFT algorithm that is able to calculate the Fourier transform of a signal that is unevenly spaced with the same  $O(N\log N)$  efficiency that is achieved with the standard FFT algorithm[87, 88]. From chapter 3, one can see that the interpolation algorithm is the most complex portion of the signal processing code, mostly due to the irregular timing requirements that are in place to ensure that the on-board buffers do not overflow. The use of the NUFFT would therefore greatly reduce the implementation complexity of FPGA ICI processing code. In addition to the simplified implementation, the NUFFT has the potential to offer a reduction in the depth-dependent sensitivity falloff, which increases the effective field of view of the ICI system. Research has shown that the application of the NUFFT in OCT systems can produce a significant improvement in this regard[89, 90]. While no NUFFT implementation exists in the LabVIEW FPGA module that was used for the development of the system presented in this thesis, efficient FPGA implementations exist on other platforms[91], suggesting that this would be a worthwhile topic for future investigation.

If a resampling is to be performed, there are alternative methods to the linear interpolation described in this thesis. Resampling is known to be a major contributor to limitations in imaging speed and quality for OCT[92], and is therefore the subject of active research that aims to find improved methods. A comparison of three common and well-known interpolation techniques (linear, cubic spline, and nearest-neighbor) shows that the three methods produce comparable results, with the nearest-neighbor method producing a peak with a slightly narrower FWHM at the cost of a slightly decreased signal-to-noise ratio[72]. A novel method has been proposed specifically for the resampling of OCT data that reduces the resampling time on a PC-based OCT system by approximately 95%[92]. Yet another OCT-specific solution uses phase information in a swept-source OCT

system to perform the interpolation with high efficiency[93]. Therefore, while the linear interpolation has been shown to be adequate for high-speed control applications of ICI, other applications may benefit from the investigation of more sophisticated interpolation techniques.

As a final point, it is worth mentioning another alternative FFT algorithm that is designed for specialized applications, the Sparse FFT. The Sparse FFT is an algorithm that is used to calculate only the  $k$  most dominant frequency components in a signal with efficiency  $O(k \log N)$ , as opposed to the  $O(N \log N)$  of the standard FFT algorithm[94]. ICI feedback control signals are typically made up of a single bright interface that is 40dB or more above the noise floor (as in figure 3.10), and therefore have the type of sparsity that this algorithm was designed for. However, in its current form, the Sparse FFT only outperforms the standard FFT algorithm for signals with more than  $2^{17}$  samples in them[95]. Since ICI signals are typically composed of  $2^9 - 2^{10}$  points, they cannot directly benefit from the Sparse FFT; however, this may not be the case in the future as improvements are made to the algorithm.

## 5.2 ASIC Design

In designing an ICI feedback control system that could overcome the lack of reliability that is inherent in a PC running on a traditional operating system, an FPGA was chosen because it is the current state-of-the-art programmable logic technology. As described in section 2.2, the FPGA provides a tradeoff when compared to the PC, wherein the performance is improved at the cost of increased complexity and turnaround time. The FPGA was chosen for this project because it is ideal for the type of prototyping that was performed for this thesis, however there is a class of devices that go beyond the FPGA. These devices are known as application-specific integrated circuits (ASICs). Integrated circuits are a broad category of devices consisting of hardwired electronics onboard a single chip made of silicon. An ASIC is, as the name suggests, an integrated circuit that has been designed and manufactured with a specific application in mind. Modern fabrication technology has pushed the production cost of integrated circuits down so that they can be manufactured at low

costs through batch processing[96].

Due to the nature of their design, ASICs provide a speed advantage over FPGAs due to the reconfigurable logic wiring scheme used in FPGAs[97]. ASICs are developed using the same standard HDLs that are used for FPGA development; in fact, it is common to design and test ASIC designs on FPGAs prior to their production so that their code can be tested in an environment with comparable performance[98]. ASICs also offer lower per-unit production costs and smaller form factor than FPGA deployments[99]. Additionally, ASICs have been shown to consume considerably less power than FPGAs, making them more appropriate in applications where power consumption is an issue[100].

ASICs are not typically used for OCT processing because they lack the reconfigurability needed by OCT researchers[36]. However, for an industrial technology such as ICI that is deployed on factory floors, there may be less of a need to tune for different optical parameters. Compared to the type of clinical settings that OCT is used for, ICI is used to perform repetitive tasks on production lines for quality assurance and control purposes, and the need to adjust the on-board logic therefore arises less frequently. Additionally, the use of ASICs would drastically reduce production costs of ICI systems if they were manufactured in high quantities. Therefore, while FPGAs offer an ideal tradeoff between cost, complexity, and performance to meet today's needs, it may be worth investigating ASICs as an alternative when production volume of ICI systems increases to meet the rising demand in industry.

### **5.3 Other Laser Manufacturing Processes**

In chapter 4, I presented the results of applying FPGA ICI control in two applications, autofocus of the machining head and control of the laser penetration depth in keyhole welding. While the results for these applications are promising and suggest that FPGAs can offer advantages over traditional processing methods for ICI, there are other industrial (and medical) processes that could benefit from ICI-driven process control. ICI has already been applied in micro-processing of silicon[101]

as well as ablation of hard tissue[102], and current research is investigating its use in laser additive manufacturing. With a wide field of laser processing methods to investigate, there is a rich future for exploration of the feedback control capabilities of an FPGA-based ICI system like the one I have presented in this thesis.

# Bibliography

- [1] X. Cao, M. Jahazi, J. Immarigeon, and W. Wallace, “A review of laser welding techniques for magnesium alloys,” *Journal of Materials Processing Technology*, vol. 171, no. 2, pp. 188 – 204, 2006.
- [2] J. M. Tessier and W. G. Brown, “Method and apparatus for laser cutting a hollow metal workpiece,” US Patent 5 073 694, 1991.
- [3] X. Wang, A. Michalowski, D. Walter, S. Sommer, M. Kraus, J. Liu, and F. Dausinger, “Laser drilling of stainless steel with nanosecond double-pulse,” *Optics & Laser Technology*, vol. 41, no. 2, pp. 148 – 153, 2009.
- [4] R. C. Nuss, R. L. Fabian, R. Sarkar, and C. A. Puliafito, “Infrared laser bone ablation,” *Lasers in Surgery and Medicine*, vol. 8, no. 4, pp. 381–391, 1988.
- [5] A. Blug, D. Carl, H. Hfler, F. Abt, A. Heider, R. Weber, L. Nicolosi, and R. Tetzlaff, “Closed-loop control of laser power using the full penetration hole image feature in aluminum welding processes,” *Physics Procedia*, vol. 12, Part A, no. 0, pp. 720 – 729, 2011, Lasers in Manufacturing 2011 - Proceedings of the Sixth International WLT Conference on Lasers in Manufacturing.
- [6] L. Mrna, M. Sarbort, S. Rerucha, and P. Jedlicka, “Feedback control of laser welding based on frequency analysis of light emissions and adaptive beam shaping,” *Physics Procedia*, vol. 39, no. 0, pp. 784 – 791, 2012, Laser Assisted Net shape Engineering 7 (LANE 2012).
- [7] A. Konuk, R. Aarts, A. H. int Veld, T. Sibillano, D. Rizzi, and A. Ancona, “Process control of stainless steel laser welding using an optical spectroscopic sensor,” *Physics Procedia*,

- vol. 12, Part A, no. 0, pp. 744 – 751, 2011, Lasers in Manufacturing 2011 - Proceedings of the Sixth International WLT Conference on Lasers in Manufacturing.
- [8] X. Gao, D. You, and S. Katayama, “Infrared image recognition for seam tracking monitoring during fiber laser welding,” *Mechatronics*, vol. 22, no. 4, pp. 370 – 380, 2012.
- [9] D. Huang, E. Swanson, C. Lin, J. Schuman, W. Stinson, W. Chang, M. Hee, T. Flotte, K. Gregory, and C. Puliafito, “Optical coherence tomography,” *Science*, vol. 254, no. 5035, pp. 1178–1181, 1991.
- [10] FPGA Fundamentals. National Instruments Corporation. Accessed September 23, 2015. [Online]. Available: <http://www.ni.com/white-paper/6983/en/>
- [11] Using VHDL to Describe Counters. Accessed September 29, 2015. [Online]. Available: <http://www.cs.uregina.ca/Links/class-info/301/counter/lecture.html>
- [12] P. J. L. Webster, “Inline coherent imaging,” Ph.D. dissertation, Queen’s University, Kingston, Ontario, Canada, November 2012.
- [13] J. P. Dunkers, D. P. Sanders, D. L. Hunston, M. J. Everett, and W. H. Green, “Comparison of optical coherence tomography, x-ray computed tomography, and confocal microscopy results from an impact damaged epoxy/e-glass composite,” *The Journal of Adhesion*, vol. 78, no. 2, pp. 129–154, 2002.
- [14] M. Brezinski, *Optical Coherence Tomography: Principles and Applications*. Academic Press, 2006.
- [15] G. Hausler and M. W. Lindner, “Coherence radar and spectral radar - new tools for dermatological diagnosis,” *Journal of Biomedical Optics*, vol. 3, no. 1, pp. 21–31, 1998.
- [16] R. Leitgeb, C. Hitzenberger, and A. Fercher, “Performance of fourier domain vs. time domain optical coherence tomography,” *Optics Express*, vol. 11, no. 8, pp. 889–894, April 2003.

- [17] M. Choma, M. Sarunic, C. Yang, and J. Izatt, "Sensitivity advantage of swept source and fourier domain optical coherence tomography," *Optics Express*, vol. 11, no. 18, pp. 2183–2189, September 2003.
- [18] S. R. Chinn, E. A. Swanson, and J. G. Fujimoto, "Optical coherence tomography using a frequency-tunable optical source," *Optics Letters*, vol. 22, no. 5, pp. 340–342, March 1997.
- [19] L. S. Lim, G. Cheung, and S. Y. Lee, "Comparison of spectral domain and swept-source optical coherence tomography in pathological myopia," *Eye*, vol. 28, no. 4, pp. 488–491, April 2014.
- [20] S. Yun, G. Tearney, J. de Boer, N. Iftimia, and B. Bouma, "High-speed optical frequency-domain imaging," *Optics Express*, vol. 11, no. 22, pp. 2953–2963, November 2003.
- [21] Z. Yaqoob, J. Wu, and C. Yang, "Spectral domain optical coherence tomography: a better OCT imaging strategy," *Biotechniques*, vol. 39, no. 6 Sppl, pp. S6–13, December 2005.
- [22] P. J. L. Webster, L. G. Wright, Y. Ji, C. M. Galbraith, A. W. Kinross, C. V. Vlack, and J. M. Fraser, "Automatic laser welding and milling with in situ inline coherent imaging," *Optics Letters*, vol. 39, no. 21, pp. 6217–6220, November 2014.
- [23] G. Dalakov. The electromechanical relay of Joseph Henry. (Accessed July 17, 2015). [Online]. Available: <http://history-computer.com/ModernComputer/Basis/relay.html>
- [24] H. H. Aiken and G. M. Hopper, "The automatic sequence controlled calculator - 1," *Electrical Engineering*, no. 8-9, pp. 384–391, August 1946.
- [25] M. Guarnieri, "The age of vacuum tubes: Merging with digital computing," *Industrial Electronics Magazine, IEEE*, vol. 6, no. 3, pp. 52–55, September 2012.
- [26] T. Kilburn, R. Grimdale, and D. Webb, "A transistor digital computer with a magnetic-drum store," *Proceedings of the IEE - Part B: Radio and Electronic Engineering*, vol. 103, no. 3, pp. 390–406, April 1956.

- [27] K. Parnell and N. Mehta, *Programmable Logic Design Quick Start Handbook*. Xilinx, Inc., April 2004.
- [28] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*, 3rd ed. McGraw Hill Higher Education, 2009.
- [29] E. D. Fabricius, *Modern Digital Design and Switching Theory*. CRC Press, June 1992.
- [30] W. Wolf, *FPGA-Based System Design*. Prentice Hall, 2004.
- [31] C. vs. FPGAs Comparing High-Capacity Programmable Logic, Altera, Tech. Rep., 1995.
- [32] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, and M. Naouar, “FPGAs in industrial control applications,” *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 2, pp. 224–243, May 2011.
- [33] Y. Lin, “Using FPGAs to solve challenges in industrial applications,” *EE Times*, November 2011. [Online]. Available: [http://www.eetimes.com/document.asp?doc\\_id=1279239](http://www.eetimes.com/document.asp?doc_id=1279239)
- [34] T. E. Ustun, N. V. Iftimia, R. D. Ferguson, and D. X. Hammer, “Real-time processing for fourier domain optical coherence tomography using a field programmable gate array,” *Review of Scientific Instruments*, vol. 79, no. 11, 2008.
- [35] V. Bandi, J. Goette, M. Jacomet, T. von Niederhusern, A. H. Bachmann, and M. Duell, “FPGA-based real-time swept-source OCT systems for B-scan live-streaming or volumetric imaging,” *Proc. SPIE*, vol. 8571, pp. 85 712Z–85 712Z–6, 2013.
- [36] J. Li, M. Sarunic, and L. Shannon, “Scalable, high performance fourier domain optical coherence tomography: Why FPGAs and not GPGPUs,” in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, May 2011, pp. 49–56.

- [37] D. F. Bacon, R. Rabbah, and S. Shukla, “FPGA programming for the masses,” *ACM Queue*, vol. 11, no. 2, February 2013.
- [38] D. Sanchez-Roman, V. Moreno, S. Lopez-Buedo, G. Sutter, I. Gonzalez, F. J. Gomez-Arribas, and J. Aracil, “FPGA acceleration using high-level languages of a monte-carlo method for pricing complex options,” *Journal of Systems Architecture*, vol. 59, no. 3, pp. 135 – 143, 2013.
- [39] B. L. Hutchings and B. E. Nelson, “Using general-purpose programming languages for FPGA design,” in *Proceedings of the 37th Annual Design Automation Conference*, ser. DAC ’00. New York, NY, USA: ACM, 2000, pp. 561–566.
- [40] J. C. Maxwell, “On governors,” *Proceedings of the Royal Society of London*, vol. 16, pp. 270–283, 1867.
- [41] L. Argentim, W. Rezende, P. Santos, and R. Aguiar, “PID, LQR and LQR-PID on a quadcopter platform,” in *Informatics, Electronics Vision (ICIEV), 2013 International Conference on*, May 2013, pp. 1–6.
- [42] S. Boubdallah, “Design and control of quadcopters with application to autonomous flying,” Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, 2007.
- [43] Naval Personnel Program Support Activity, Training Publications Division, Washington, D.C, *Gunner’s Mate M 3 & 2*. Naval Training Support Command, 1972.
- [44] B. Zhang, K. Liu, and J. Xiang, “A stabilized optimal nonlinear feedback control for satellite attitude tracking,” *Aerospace Science and Technology*, vol. 27, no. 1, pp. 17 – 24, 2013.
- [45] J. Doyle, B. Francis, and A. Tannenbaum, *Feedback Control Theory*. Macmillan Publishing Co., 1990.
- [46] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 6th ed. Pearson Higher Education Inc., 2010.

- [47] J. Bechhoefer, "Feedback for physicists: A tutorial essay on control," *Reviews of Modern Physics*, vol. 77, pp. 783–836, August 2005.
- [48] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, July 2005.
- [49] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *J. Dyn. Sys., Meas., Control*, vol. 115, no. 2B, pp. 220–222, June 1993.
- [50] K. Astrom and T. Hagglund, "Revisiting the ziegler-nichols step response method for PID control," *Journal of Process Control*, vol. 14, no. 6, pp. 635 – 650, 2004.
- [51] C. C. Hang, K. J. Astrom, and W. K. Ho, "Refinements of the Ziegler-Nichols tuning formula," *IEE Proceedings D (Control Theory and Applications)*, vol. 138, no. 2, pp. 111–118, March 1991.
- [52] B. D. Tyreus and W. L. Luyben, "Tuning PI controllers for integrator/dead time processes," *Ind. Eng. Chem. Res.*, vol. 31, no. 11, pp. 2625–2628, November 1992.
- [53] H. M. Muncheryan, "Laser welding machine," US Patent 3 383 491, 1968.
- [54] G. Fernandez and L. Murr, "Characterization of tool wear and weld optimization in the friction-stir welding of cast aluminum 359+20% SiC metal-matrix composite," *Materials Characterization*, vol. 52, no. 1, pp. 65 – 75, 2004.
- [55] J. Lucas, "Laser beam vs. electron beam welding: Which process works best for what?" NASA, Tech. Rep., 2011.
- [56] W. M. Steen and J. Mazumder, *Laser Material Processing*, 4th ed. Springer, 2010.
- [57] H. Ki, J. Mazumder, and P. Mohanty, "Modeling of laser keyhole welding: Part i. mathematical modeling, numerical methodology, role of recoil pressure, multiple reflections, and free

- surface evolution,” *Metallurgical and Materials Transactions A*, vol. 33, no. 6, pp. 1817–1830, 2002.
- [58] X. Ficquet, D. Smith, C. Truman, E. Kingston, and R. Dennis, “Measurement and prediction of residual stress in a bead-on-plate weld benchmark specimen,” *International Journal of Pressure Vessels and Piping*, vol. 86, no. 1, pp. 20 – 30, 2009, the NeT Residual Stress Measurement and Modelling Round Robin on a Single Weld Bead-on-Plate Specimen.
- [59] P. Sathiya, S. Aravindan, P. Ajith, B. Arivazhagan, and A. N. Haq, “Microstructural characteristics on bead on plate welding of AISI 904 L super austenitic stainless steel using gas metal arc welding process,” *International Journal of Engineering, Science and Technology*, vol. 2, no. 6, pp. 189–199, 2010.
- [60] C. Dawes, *Laser Welding: A practical guide*. Abington Publishing, 1992.
- [61] A. Matsunawa, N. Seto, J.-D. Kim, M. Mizutani, and S. Katayama, “Dynamics of keyhole and molten pool in high-power CO<sub>2</sub> laser welding,” *Proc. SPIE*, vol. 3888, pp. 34–45, 2000.
- [62] N. Seto, S. Katayama, and A. Matsunawa, “High-speed simultaneous observation of plasma and keyhole behavior during high power CO<sub>2</sub> laser welding: Effect of shielding gas on porosity formation,” *Journal of Laser Applications*, vol. 12, no. 6, pp. 245–250, 2000.
- [63] Y. Kawahito, N. Matsumoto, Y. Abe, and S. Katayama, “Relationship of laser absorption to keyhole behavior in high power fiber laser welding of stainless steel and aluminum alloy,” *Journal of Materials Processing Technology*, vol. 211, no. 10, pp. 1563 – 1568, 2011.
- [64] A. G. Paleocrassas and J. F. Tu, “Low-speed laser welding of aluminum alloy 7075-T6 using a 300-W, single-mode ytterbium fiber laser,” *Welding Journal*, vol. 86, no. 6, p. 179, June 2007.

- [65] L. Quintino, A. Costa, R. Miranda, D. Yapp, V. Kumar, and C. Kong, “Welding with high power fiber lasers – a preliminary study,” *Materials & Design*, vol. 28, no. 4, pp. 1231 – 1237, 2007.
- [66] T. Ussing, L. Petersen, C. Nielsen, B. Helbo, and L. Hjslet, “Micro laser welding of polymer microstructures using low power laser diodes,” *The International Journal of Advanced Manufacturing Technology*, vol. 33, no. 1-2, pp. 198–205, 2007.
- [67] S. Katayama, M. Mizutani, Y. Kawahito, S. Ito, and D. Sumimori, “Fundamental research of 100 kW fiber laser welding technology,” *Proceedings of the Lasers in Manufacturing Conference 2015*, 2015.
- [68] D. Belforte. (2015, January) What’s new with 100kW fiber lasers? Industrial Laser Solutions. [Online]. Available: <http://www.industrial-lasers.com/articles/print/volume-30/issue-1/departments/update/what-s-new-with-100kw-fiber-lasers.html>
- [69] *Specifications of the Camera Link Interface Standard for Digital Cameras and Frame Grabbers*, Image Labs international, October 2000. [Online]. Available: <http://www.imagelabs.com/wp-content/uploads/2010/10/CameraLink5.pdf>
- [70] A. Kramida, Yu. Ralchenko, J. Reader, and NIST ASD Team, NIST Atomic Spectra Database (ver. 5.2), [Online]. Available: <http://physics.nist.gov/asd> [2015, February 10]. National Institute of Standards and Technology, Gaithersburg, MD., 2014.
- [71] S. Vergnole, D. Lévesque, and G. Lamouche, “Experimental validation of an optimized signal processing method to handle non-linearity in swept-source optical coherence tomography,” *Opt. Express*, vol. 18, no. 10, pp. 10 446–10 461, May 2010.
- [72] C. N. Copeland and A. K. Ellerbee, “Analysis of the effects of different resampling techniques for optical coherence tomography,” pp. 82 270W–82 270W–6, 2012.

- [73] R. Wain, I. Bush, M. Guest, M. Deegan, I. Kozin, and C. Kitchen, *An overview of FPGAs and FPGA programming: Initial experiences at Daresbury*. Council for the Central Laboratory of the Research Councils, 2006.
- [74] (2008-2009) NI FlexRIO FPGA Module Specifications. National Instruments Corporation. [Online]. Available: <http://www.ni.com/pdf/manuals/372525d.pdf>
- [75] *NI 1483R User Guide and Specifications*, National Instruments Corporation, September 2011. [Online]. Available: <http://www.ni.com/pdf/manuals/375502b.pdf>
- [76] *UM10204 I<sup>2</sup>C-bus specification and user manual*, NXP Semiconductors, April 2014. [Online]. Available: [http://www.nxp.com/documents/user\\_manual/UM10204.pdf](http://www.nxp.com/documents/user_manual/UM10204.pdf)
- [77] *MCP4725 Datasheet*, Microchip Technology Inc., 2009, Accessed September 23, 2015. [Online]. Available: <http://ww1.microchip.com/downloads/en/DeviceDoc/22039d.pdf>
- [78] *LM741 Operational Amplifier Datasheet*, Texas Instruments Incorporated, March 2013, Accessed September 23, 2015. [Online]. Available: <http://www.ti.com/lit/ds/snosc25c/snosc25c.pdf>
- [79] (2004) Non-inverting op-amp level shifter. Daycounter, Inc. Accessed May 22, 2015. [Online]. Available: <http://www.daycounter.com/Circuits/OpAmp-Level-Shifter/OpAmp-Level-Shifter.phtml>
- [80] M. J. Madic and M. R. Radovanovic, "Analysis of the heat affected zone in CO<sub>2</sub> laser cutting of stainless steel," *Thermal Science*, vol. 16, no. 2, pp. S363–S373, 2012.
- [81] J. E. Duffin, "Laser drilling with optical feedback," US Patent 6 201 214 B1, March, 2001.
- [82] M. Nantel and D. Grozdanovski, "Autofocus feedback positioning system for laser processing," US Patent 6 621 060 B1, September, 2003.

- [83] M. Forrer, M. Frenz, A. D. Zweig, V. Romano, H. P. Weber, and J. Ochsenbein, “Autofocus system for a surgical CO<sub>2</sub> laser,” *Applied Optics*, vol. 30, no. 12, pp. 1480–1486, Apr 1991.
- [84] *Automation 3200 Help Documentation*, Aerotech, Inc., 2001-2011.
- [85] T. Sibillano, D. Rizzi, F. P. Mezzapesa, P. M. Lugar, A. R. Konuk, R. Aarts, B. H. in’t Veld, , and A. Ancona, “Closed loop control of penetration depth during CO<sub>2</sub> laser lap welding processes,” *Sensors*, vol. 12, no. 8, pp. 11 077–11 090, August 2012.
- [86] C. Galbraith, “Inline coherent imaging of laser keyhole welding,” Master’s thesis, Queen’s University, Kingston, Ontario, Canada, April 2015.
- [87] J.-Y. Lee and L. Greengard, “The type 3 nonuniform FFT and its applications,” *Journal of Computational Physics*, vol. 206, no. 1, pp. 1 – 5, 2005.
- [88] L. Greengard and J.-Y. Lee, “Accelerating the nonuniform fast fourier transform,” *SIAM Review*, vol. 46, no. 3, pp. 443–454, 2004.
- [89] K. Wang, Z. Ding, T. Wu, C. Wang, J. Meng, M. Chen, and L. Xu, “Development of a non-uniform discrete fourier transform based high speed spectral domain optical coherence tomography system,” *Opt. Express*, vol. 17, no. 14, pp. 12 121–12 131, July 2009.
- [90] K. K. H. Chan and S. Tang, “High-speed spectral domain optical coherence tomography using non-uniform fast fourier transform,” *Biomed. Opt. Express*, vol. 1, no. 5, pp. 1309–1319, December 2010.
- [91] S. Kestur, S. Park, K. Irick, and V. Narayanan, “Accelerating the nonuniform fast fourier transform using FPGAs,” in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, May 2010, pp. 19–26.
- [92] Y. Zhang, X. Li, L. Wei, K. Wang, Z. Ding, and G. Shi, “Time-domain interpolation for fourier-domain optical coherence tomography,” *Optics Letters*, vol. 34, no. 12, pp. 1849–1851, June 2009.

- [93] T. Wu, Z. Ding, L. Wang, and M. Chen, "Spectral phase based k-domain interpolation for uniform sampling in swept-source optical coherence tomography," *Optics Express*, vol. 19, no. 19, pp. 18 430–18 439, September 2011.
- [94] D. K. Haitham Hassanieh, Piotr Indyk and E. Price., "Nearly optimal sparse fourier transform," *MIT Symposium on Theory of Computing*, May 2012.
- [95] D. K. Haitham Hassanieh, Piotr Indyk and E. Price, "Simple and practical algorithm for sparse fourier transform," *ACM-SIAM Symposium on Discrete Algorithms*, January 2012.
- [96] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 5th ed. Wiley, 2009.
- [97] K. Gaj and P. Chodowiec, "FPGA and ASIC implementations of AES," in *Cryptographic Engineering*. Springer US, 2009, pp. 235–294.
- [98] D. Markovic, C. Chang, B. Richards, H. So, B. Nikolic, and R. Brodersen, "ASIC design and verification in an FPGA environment," in *Custom Integrated Circuits Conference, 2007. CICC '07. IEEE*, September 2007, pp. 737–740.
- [99] FPGA vs. ASIC. Xilinx Inc. Accessed July 25, 2015. [Online]. Available: <http://www.xilinx.com/fpga/asic.htm>
- [100] A. Amara, F. Amiel, and T. Ea, "FPGA vs. ASIC for low power applications," *Microelectronics Journal*, vol. 37, no. 8, pp. 669 – 677, 2006.
- [101] K. X. Z. Yu, L. G. Wright, P. J. L. Webster, and J. M. Fraser, "Deep nonlinear ablation of silicon with a quasi-continuous wave fiber laser at 1070 nm," *Optics Letters*, vol. 38, no. 11, pp. 1799–1801, June 2013.
- [102] B. Y. Leung, P. J. Webster, J. M. Fraser, and V. X. Yang, "Real-time guidance of thermal and ultrashort pulsed laser ablation in hard tissue using inline coherent imaging," *Lasers in Surgery and Medicine*, vol. 44, no. 3, pp. 249–256, 2012.

# Appendix A MATLAB Code

```
StartingPixel = 1793;
NumOfPixels = 512;

Poly = StartingPixel:1:StartingPixel+NumOfPixels-1;
Poly = (2*pi()*1E+9) ./ (co0 + co1 * Poly + co2 * Poly.^2 + co3 * Poly.^3 +
    co4 * Poly.^4);

spacing = (Poly(end) - Poly(1)) / (NumOfPixels-1);
xAxis = Poly(1):spacing:Poly(end);

IndexShift = floor((xAxis-Poly)/spacing);
```

**Figure A.1:** MATLAB implementation of the index shift array generation code.

```
InterpArray = zeros(1,NumOfPixels);
for n=1:NumOfPixels
    firstIndex = n;

    secondIndex = n+IndexShift(n);
    if(secondIndex < 1)
        secondIndex = 512-secondIndex;
    end

    thirdIndex = n+IndexShift(n)+1;
    if(thirdIndex > 511)
        thirdIndex = thirdIndex-511;
    end

    InterpArray(n) = (xAxis(firstIndex) - Poly(secondIndex)) / (Poly(
        thirdIndex) - Poly(secondIndex));
end
```

**Figure A.2:** MATLAB implementation of the interpolation array generation code.

```

for n = 1:512
    index0(n) = n+IndexShift(n);
    index1(n) = n+IndexShift(n)+1;
end

numOfOccurrences0 = zeros(1,513);
numOfOccurrences1 = zeros(1,513);

for n = 1:512
    numOfOccurrences0(index0(n)) = numOfOccurrences0(index0(n)) + 1;
    numOfOccurrences1(index1(n)) = numOfOccurrences1(index1(n)) + 1;
end

isZero0 = zeros(1,513);
isZero1 = zeros(1,513);

for n = 1:513
    if(numOfOccurrences0(n) == 0)
        isZero0(n) = 1;
    end

    if(numOfOccurrences1(n) == 0)
        isZero1(n) = 1;
    end
end

isTwo0 = zeros(0);
isTwo1 = zeros(0);

for n = 1:513
    if(numOfOccurrences0(n) == 1)
        isTwo0 = [isTwo0 0];
    elseif(numOfOccurrences0(n) == 2)
        isTwo0 = [isTwo0 1];
    end

    if(numOfOccurrences1(n) == 1)
        isTwo1 = [isTwo1 0];
    elseif(numOfOccurrences1(n) == 2)
        isTwo1 = [isTwo1 1];
    end
end
end

```

**Figure A.3:** MATLAB implementation of the NoC array generation code.

```

zeroElim0 = zeros(0);
zeroElim1 = zeros(0);

%Pixel Skipping Loop
for n=1:513

    m = n;
    if(m > 512)
        m=512;
    end

    if(isZero0(n) == 0)
        zeroElim0 = [zeroElim0 SampleNoDC(m)];
    end

    if(isZero1(n) == 0)
        zeroElim1 = [zeroElim1 SampleNoDC(m)];
    end
end

twoDub0 = zeros(0);
twoDub1 = zeros(0);

%Pixel Repeating Loop
for n=1:505
    twoDub0 = [twoDub0 zeroElim0(n)];
    if(isTwo0(n) == 1)
        twoDub0 = [twoDub0 zeroElim0(n)];
    end

    twoDub1 = [twoDub1 zeroElim1(n)];
    if(isTwo1(n) == 1)
        twoDub1 = [twoDub1 zeroElim1(n)];
    end
end

%Interpolation Loop
for n=1:512
    NewInterp(n) = twoDub0(n) + InterpArray(n) * (twoDub1(n) - twoDub0(n));
end

```

**Figure A.4:** MATLAB implementation of the interpolation calculation including the skipping and repeating of pixels as regulated by the index shift array.