High-performance Real-time Motion Control for Precision Systems

by

Kristofer S. Smeds

BASc. The University Of British Columbia, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE STUDIES
(Mechanical Engineering)

The University Of British Columbia
(Vancouver)

April 2011

© Kristofer S. Smeds, 2011
Abstract

Digital motion controllers are used almost exclusively in automated motion control systems today. Their key performance parameters are controller execution speed, timing consistency, and data accuracy. Most commercially available controllers can achieve sampling rates up to 20kHz with several microseconds of timing variation between control cycles. A few state-of-the art control platforms can reach sampling rates of around 100kHz with several hundred nanoseconds of timing variation. There exist a growing number of emerging high-speed high-precision applications, such as diamond turning and scanning probe microscopy, that can benefit from digital controllers capable of faster sampling rates, more consistent timing, and higher data accuracy.

This thesis presents two areas of research intended to increase the capabilities of digital motion controllers to meet the needs of these high-speed high-precision applications.

First, it presents a new high-performance real-time multiprocessor control platform capable of 1MHz control sampling rates with less than 6ns RMS control cycle timing variation and 16-bit data acquisition accuracy. This platform also includes software libraries to integrate it with Simulink for rapid controller development and LabVIEW for easy graphical user interface development. This thesis covers the design of the control platform and experimentally demonstrates it as a motion controller for a fast-tool servo machine tool.

Second, this thesis investigates the effect of control cycle timing variations (sampling jitter and control jitter) on control performance, with an emphasis on precision positioning degradation. A new approximate discrete model is developed to capture the effects of jitter, enabling an intuitive understanding of it’s effects on the control system. Based on this model, analyses are carried out to determine the relationship between jitter and positioning error for two scenarios: regulation error from jitter’s interaction with measurement noise; and tracking error from jitter’s interaction with a deterministic reference command. Further, several practical methods to mitigate the positioning degradation due to jitter are discussed, including a new jitter compensator that can be easily added to an existing controller. Through simulations and experiments performed on a fast-tool servo machine tool, the model and analyses are validated and the positioning degradation arising from jitter is clearly demonstrated.
Table of Contents

Abstract ................................................................. ii

Table of Contents ...................................................... iii

List of Tables .......................................................... v

List of Figures ......................................................... vi

Acronyms ................................................................. x

Acknowledgments ........................................................ xi

1 Introduction ........................................................... 1
  1.1 Real-time Digital Control Background .......................... 2
  1.2 Related Work ...................................................... 6
    1.2.1 State-of-the-art Real-time Control Platforms .......... 7
    1.2.2 Sampling and Control Jitter ............................ 10
  1.3 Thesis Outline .................................................. 11

2 Tsunami Control Platform Design .................................. 14
  2.1 Design Objectives ............................................... 15
  2.2 Hardware Design ................................................ 17
    2.2.1 Motherboard Design and Implementation ............ 23
    2.2.2 Analog-DAQ Daughterboard Design and Implementation 32
    2.2.3 System Integration .................................... 39
  2.3 Software Design ................................................ 42
    2.3.1 Simulink Controller Development Integration ........ 42
    2.3.2 LabVIEW GUI Development Integration .............. 44

3 Tsunami Control Platform Testing and Results .................... 49
  3.1 Performance Evaluation ....................................... 49
    3.1.1 Analog Inputs ........................................... 49
    3.1.2 Analog Outputs .......................................... 52
    3.1.3 Input-Output Latency .................................... 54
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.4 Sampling Rate</td>
<td>55</td>
</tr>
<tr>
<td>3.1.5 Sampling and Control Jitter</td>
<td>56</td>
</tr>
<tr>
<td>3.2 Case Study: Fast-tool Servo</td>
<td>57</td>
</tr>
<tr>
<td>3.2.1 Controller Implementation</td>
<td>61</td>
</tr>
<tr>
<td>3.2.2 Control System Performance Results</td>
<td>64</td>
</tr>
<tr>
<td>4 Jitter Modeling and Analysis</td>
<td>69</td>
</tr>
<tr>
<td>4.1 Modeling</td>
<td>69</td>
</tr>
<tr>
<td>4.1.1 Zero-order-hold with Control Jitter</td>
<td>72</td>
</tr>
<tr>
<td>4.1.2 Sampler with Sampling Jitter</td>
<td>75</td>
</tr>
<tr>
<td>4.1.3 Overall Discrete Jitter Model</td>
<td>76</td>
</tr>
<tr>
<td>4.2 Analysis of Jitter’s Effect on Positioning Error</td>
<td>78</td>
</tr>
<tr>
<td>4.2.1 Regulation Error Analysis</td>
<td>79</td>
</tr>
<tr>
<td>4.2.2 Tracking Error Analysis</td>
<td>82</td>
</tr>
<tr>
<td>4.3 Solutions to Mitigate Positioning Error from Jitter</td>
<td>84</td>
</tr>
<tr>
<td>5 Jitter Simulation and Experimental Results</td>
<td>86</td>
</tr>
<tr>
<td>5.1 Simulation Results</td>
<td>89</td>
</tr>
<tr>
<td>5.1.1 Regulation Error Simulation Results</td>
<td>90</td>
</tr>
<tr>
<td>5.1.2 Tracking Error Simulation Results</td>
<td>91</td>
</tr>
<tr>
<td>5.2 Experimental Results</td>
<td>93</td>
</tr>
<tr>
<td>5.2.1 Regulation Error Experimental Results</td>
<td>94</td>
</tr>
<tr>
<td>5.2.2 Tracking Error Experimental Results</td>
<td>101</td>
</tr>
<tr>
<td>6 Conclusions</td>
<td>106</td>
</tr>
<tr>
<td>6.1 Future Work</td>
<td>107</td>
</tr>
<tr>
<td>Bibliography</td>
<td>109</td>
</tr>
<tr>
<td>Appendix A Jitter Measurement</td>
<td>114</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1 Daughterboard expansion interface electrical specifications 27
Table 3.1 Fast-tool servo mechanical and controller specifications 61
# List of Figures

| Figure 1.1 | An ideal digital control block diagram. | 2 |
| Figure 1.2 | Typical control cycle timing of a real-time controller. | 3 |
| Figure 1.3 | A digital control block diagram including non-ideal implementation details. | 3 |
| Figure 1.4 | Estimation of digital feedback control system phase-loss at the loop transmission crossover frequency for various sampling rates. | 5 |
| Figure 2.1 | Picture of the completed Tsunami control platform hardware. | 14 |
| Figure 2.2 | Tsunami control platform hardware architecture diagram. | 18 |
| Figure 2.3 | Comparison between the typical control cycle timing and the triple-body control cycle timing. | 20 |
| Figure 2.4 | Diagram showing the separation of the timing critical control path and the non-timing critical host communication path. | 20 |
| Figure 2.5 | Control cycle timing breakdown for Tsunami. | 21 |
| Figure 2.6 | Control cycle timing breakdown for Tsunami with ADC pipelining. | 22 |
| Figure 2.7 | Motherboard functional block diagram. | 24 |
| Figure 2.8 | Star bus topology and trace length specifications for the cluster bus. | 25 |
| Figure 2.9 | Internal FPGA system architecture for Tsunami. | 26 |
| Figure 2.10 | Daughterboard expansion interface mechanical specifications. | 28 |
| Figure 2.11 | Motherboard PCB layer stackup. | 29 |
| Figure 2.12 | Motherboard power distribution system. | 30 |
| Figure 2.13 | Top view of the fully assembled motherboard PCB. | 31 |
| Figure 2.14 | Analog-DAQ daughterboard functional block diagram. | 32 |
| Figure 2.15 | Simplified circuit of the 4MSPS ADC analog front end. | 33 |
| Figure 2.16 | Simplified circuit of the 50MSPS DAC analog front end. | 34 |
| Figure 2.17 | Simplified circuit of the 500kSPS ADC analog front end. | 34 |
| Figure 2.18 | Simplified circuit of the 500kSPS DAC analog front end. | 35 |
| Figure 2.19 | Analog-DAQ daughterboard layer stack-up. | 35 |
| Figure 2.20 | Analog-DAQ PCB layout and routing. | 36 |
| Figure 2.21 | Analog-DAQ power distribution system. | 37 |
| Figure 2.22 | Completed Analog-DAQ daughterboard PCB with key components labeled. | 38 |
| Figure 2.23 | Analog-DAQ I/O PCB board. | 39 |
Figure 2.24 Top view of the Tsunami real-time control hardware with the case open. 40
Figure 2.25 Top view of the Tsunami real-time control hardware with the case closed. 40
Figure 2.26 Rear view of the Tsunami real-time control hardware. 41
Figure 2.27 Tsunami Simulink library for controller development. 41
Figure 2.28 Tsunami Simulink library encoder block configuration. 43
Figure 2.29 Example of a Simulink controller model using the Tsunami library. 44
Figure 2.30 Tsunami LabVIEW library for GUI development. 45
Figure 2.31 Connecting controller signals to the Tsunami Display Signal LabVIEW library block. 46
Figure 2.32 Example of a simple LabVIEW GUI to log a signal from an implemented Tsunami controller. 47
Figure 2.33 Example servo controller GUI created using the Tsunami LabVIEW GUI library. 48

Figure 3.1 Noise floors for the 4MSPS, 16-bit analog inputs. 51
Figure 3.2 Noise floors for the 500kSPS, 16-bit analog inputs. 52
Figure 3.3 Step responses for one of the 50MSPS, 16-bit analog outputs. 53
Figure 3.4 Step responses for one of the 500kSPS, 16-bit analog outputs. 54
Figure 3.5 Experimental setup used to measure input-output latency. 55
Figure 3.6 Measured input-output latency for the Tsunami control platform. 55
Figure 3.7 Measured turnaround time for the execution of the example control algorithm. 56
Figure 3.8 Measured control jitter for the Tsunami control platform. 57
Figure 3.9 Fast-tool servo experimental setup. 58
Figure 3.10 Control system block diagram. 58
Figure 3.11 Measured plant frequency response for the fast-tool servo. 59
Figure 3.12 Loop-shaping controller frequency response for the fast-tool servo. 60
Figure 3.13 Negative loop transmission frequency response for the fast-tool servo control system. 60
Figure 3.14 Closed-loop frequency response for the fast-tool servo control system. 61
Figure 3.15 Simulink implementation of the FTS controller. 62
Figure 3.16 LabVIEW graphical user interface for the FTS control system. 63
Figure 3.17 Block diagram for the LabVIEW graphical user interface for the FTS control system. 64
Figure 3.18 Control cycle turnaround times for the FTS controller implemented on Tsunami. 65
Figure 3.19 Control latency for the FTS controller implemented on Tsunami. 65
Figure 3.20 500nm step responses for the FTS control system. 66
Figure 3.21 Regulation error for the FTS control system. 66
Figure 3.22 6kHz, 8μm peak-peak reference command. 67
Figure 3.23 Tracking error for the FTS system for a 6kHz, 8μm peak-peak reference command with four AFC compensators. 67

Figure 4.1 A Digital control feedback system with non-ideal sampler and non-ideal ZOH. 70
Figure 4.2 The sequential timing process in a typical digital control cycle. 71
Figure 4.3 Equivalent model of a digital control feedback system with non-ideal sampler and ZOH ........................................ 72
Figure 4.4 Modeling of non-ideal ZOH with control jitter ........................................ 74
Figure 4.5 Modeling of non-ideal sampler with sampling jitter ................................... 76
Figure 4.6 Digital control system model with jitter disturbance inputs ........................................ 77
Figure 4.7 Overall discrete jitter disturbance model including the effects of sampling jitter and control jitter ........................................ 78
Figure 4.8 Frequency response of the discrete derivative term $1 - e^{-j\Omega}$ ......................... 82
Figure 4.9 Frequency response of the jitter compensator for mitigating control jitter disturbance on regulation error ........................................ 85

Figure 5.1 Measured and analytical plant frequency response for the fast-tool servo ............... 87
Figure 5.2 Loop-shaping base controller frequency response for the fast-tool servo jitter investigation ........................................ 88
Figure 5.3 Closed-loop frequency response for the fast-tool servo control jitter investigation ........................................ 88
Figure 5.4 Simulink model used to simulate the effect of jitter on system performance ........... 89
Figure 5.5 Simulated and analytical RMS regulation error comparison for various amounts of sampling jitter ........................................ 90
Figure 5.6 Simulated and analytical RMS regulation error comparison for various amounts of control jitter, with and without the jitter compensator ........................................ 91
Figure 5.7 Simulated and analytical RMS tracking error comparison for various amounts of sampling jitter ........................................ 92
Figure 5.8 Simulated and analytical RMS tracking error comparison for various amounts of control jitter ........................................ 92
Figure 5.9 8% RMS normalized jitter data used for the experiments ..................................... 93
Figure 5.10 System block diagram of the experimental setup ........................................... 94
Figure 5.11 Regulation error experimental results for no jitter ........................................... 95
Figure 5.12 Regulation error experimental results for 8% sampling jitter ............................. 97
Figure 5.13 Measured and analytical RMS regulation error comparison for various amounts of sampling jitter ........................................ 98
Figure 5.14 Regulation error experimental results for 8% control jitter ................................ 99
Figure 5.15 Regulation error experimental results for 8% control jitter with the jitter compensator $C_g(z)$ implemented ........................................ 100
Figure 5.16 Measured and analytical RMS regulation error comparison for various amounts of control jitter, with and without the jitter compensator $C_g(z)$ ........................................ 101
Figure 5.17 Tracking error experimental results for no jitter with and without AFC ............... 102
Figure 5.18 Tracking error experimental results for 8% sampling jitter ................................ 103
Figure 5.19 Measured and analytical RMS tracking error comparison for various amounts of sampling jitter ........................................ 104
Figure 5.20 Tracking error experimental results for 8% sampling jitter ................................ 105
Figure 5.21  Measured and analytical RMS tracking error comparison for various amounts of
sampling jitter .............................................................. 105

Figure A.1  Control jitter measurements for various controller implementations ....... 116
Acronyms

**PCB** printed circuit board

**DSP** digital signal processor

**TS-201** TigerSHARC-201

**FPGA** Field Programmable Gate Array

**SGMII** Serial Gigabit Media Independent Interface

**ADC** analog-to-digital converter

**DAC** digital-to-analog converter

**MIMO** multi-input multi-output

**SAR** successive approximation register

**LSB** least significant bit

**FTS** fast-tool servo

**GUI** graphical user interface

**AFC** adaptive feedforward cancellation

**GFLOPS** billion floating point operations per second

**AFE** analog front-end

**PWM** pulse width modulation

**MSPS** mega-samples per second

**KSPS** kilo-samples per second

**PSD** power spectrum density

**I/O** input-output

**ZOH** zero-order hold
Acknowledgments

I would like to begin by thanking my supervisor Dr. Xiaodong Lu for the opportunity to pursue my Master degree under his guidance. I am extremely grateful for all his support, assistance, and knowledge that has formed me into the researcher I am today. His engineering talent, quick thinking, and research drive never ceases to amaze me and it has been an absolute pleasure learning from him. The opportunities he has given me to attend conferences and present my work have been great experiences that I will not forget. Finally, I would like to thank him for his numerous contributions to the real-time computer project in this Thesis, as I could not have finished it alone.

I would also like to thank Dr. Yusuf Altintas whose support, wisdom, advice, and excellent teaching inspired me to pursue my Master’s degree in Mechatronics.

My colleagues in the Precision Mechatronics Lab have all been a pleasure to work alongside and have made my Master’s experience great, both inside and outside of the lab. I am going to miss our weekly sushi lunches. I want to thank Richard Graetz, Arash Jamalian, and Niankun Rao, who created the nanoRAD board for the real-time computer. Also, I want to especially thank Irfan Usman who worked tirelessly with me for weeks on the jitter research in preparation for a conference presentation. Irfan’s willingness to help and insightful feedback has really brought out the best in my research.

Lastly I would like to thank my parents, whose support and encouragement has made it possible for me to get to where I am today. Mom, thank you for taking me back in this last year and being way too accommodating, without you I would have burned out.
Chapter 1

Introduction

A real-time motion controller is a fundamental component in all manufacturing and automation systems. It is responsible for acquiring data from system sensors, processing the data using the implemented control algorithm, and outputting control signals to the system power amplifiers that drive the actuators. For digital controllers, which are used almost exclusively in motion control today, this process is repeated at a fixed sampling frequency. Additionally, the real-time controller must provide a user interface for the engineer to monitor signals, tune parameters, and log data.

The key performance parameters for a real-time controller are control execution speed, timing consistency, and data accuracy. Most commercially available digital controllers can achieve sampling rates of up to 20kHz, with a few state-of-the art controllers reaching sampling rates of 100kHz-200kHz. There exist a growing number of emerging applications that can benefit from higher performance real-time controllers, such as diamond turning, scanning probe microscopy, and laser beam steering.

A key component in a diamond turning process is a fast-tool servo, which enables the manufacturing of free-form surfaces at nanometer resolution. The fast-tool servo presented in [1], has demonstrated a closed-loop bandwidth of 23kHz and a positioning accuracy of 1.6nm RMS. To fully realize this system’s capabilities, a real-time controller with precision data acquisition and a sampling frequency of over 500kHz is needed. Another example is video rate scanning probe microscopy. A major limitation of scanning probe microscopes compared to other microscopy technologies is scanning speed, as it currently takes several minutes to scan a small area of 100x100 micrometers. The speed of the feedback control system and the speed and accuracy of the data acquisition are important challenges that must be overcome to achieve faster scanning speeds [2], enabling high productivity.

The focus of this thesis is on increasing the capabilities of real-time controllers to meet the needs of emerging high-speed high-precision control applications. First, it presents a new real-time multiprocessor control platform that is capable of achieving up to a 1MHz control sampling rate with less than 6ns RMS control cycle timing variation and 16-bit data acquisition accuracy. Second, it investigates the effect of control cycle timing variations on control performance, with an emphasis on degradation to motion control regulation and tracking performance.

The remainder of this chapter provides some background on real-time control, reviews existing state-of-the-art real-time controllers, and reviews prior art on control timing variations. It also includes
1.1 Real-time Digital Control Background

In [3], Gambier introduces important areas and considerations for real-time control system design. It is a cross-discipline task involving electrical hardware engineering, software engineering, and control engineering. Traditionally, software engineers have focused on real-time operating system development and task scheduling, electrical engineers on the development of computing hardware including custom processing boards and input-output interfaces, and control engineers on digital controller design and theory. However, to fully realize the performance potential of real-time control systems the software, hardware, and control design must all be tightly integrated and designed together. More recently, there has been a movement in this direction through Control-Scheduling Co-design [4], but there still remains much potential to improve real-time control performance through tight integration of these disciplines. An important first step for this is understanding how the various implementation aspects affect control performance.

Generally in digital control textbooks [5] [6], the implementation details of the controller are ignored during design, resulting in the ideal control system block diagram shown in Figure 1.1. It consists of a discrete feedback controller, $C(z)$, a continuous plant, $P(s)$, and a reference command input, $r[k]$. The sampling of the plant output and the updating of the plant input, via zero-order hold ($ZOH$), are assumed to happen simultaneously at evenly spaced constant intervals of sampling period $T_0$.

![Figure 1.1: An ideal digital control block diagram.](image)

In reality, the implementation of a digital controller on a real-time computer introduces several non-idealities that limit both timing performance [7] and accuracy of the control system. From an accuracy standpoint, the A/D conversion, controller computation and D/A conversion each introduce additional errors into the system. From a timing standpoint, consider the typical control cycle execution shown in Figure 1.2.

The control cycle starts with the expiration of a periodic timer, which is followed by an initial interrupt time $T_{INT}$ that passes before the computer begins executing the actual control task. The control task, which includes sampling the plant output $y(t)$, computing the controller $C(z)$, and outputting the result $u(t)$, then takes $T_{CTRL}$ time to complete its execution. Lastly, there exists an overhead time $T_{OVR}$ required for the real-time computer to remain responsive to commands from the user interface.
This typical control cycle has several effects on control timing performance. First, it limits the achievable sampling frequency because the next cycle cannot be started until the current cycle is completed. Second, it introduces a delay from the instant $y(t)$ is sampled to the instant $u(t)$ is updated due to analog-digital conversion time, controller computation time, and digital-analog conversion time. This delay is referred to as control latency $\tau_d$. Lastly, the sampling instants and control output instants are not evenly spaced due to factors such as interrupt handling, resource sharing, and task scheduling. These feedback sampling instant and control output instant temporal deviations from the ideal timing are referred to as sampling jitter $\tau_s[k]$ and control jitter $\tau_c[k]$, respectively.

These non-idealities are incorporated into the digital control block diagram shown in Figure 1.3. The timing issues are included as non-ideal sampler and ZOH elements. The accuracy elements are included by introducing additional error sources on the A/D conversion, controller computation, and D/A conversion.

The effect of each of these non-idealities on control performance are expanded on below.
Control Latency and Sampling Frequency

As defined by Shannon’s sampling theorem, the fundamental sampling limit to be able to reconstruct a continuous time signal is twice the highest frequency of interest in the system. In closed-loop control systems the highest frequency of interest is closely related to the system’s closed loop bandwidth; however, delay induced phase loss necessitates that the sampling frequency be significantly higher than twice the bandwidth. In [8], Astrom recommends selecting the sampling rate to be 10 to 30 times the closed-loop bandwidth.

The controller delay arises from two items: the control latency, and the zero-order-hold update rate. When a controller is implemented digitally it takes a finite amount of time to complete a control cycle, resulting in a delay from the instant the plant output, \(y(t)\), is sampled to the instant the control action, \(u(t)\), is updated. As mentioned above, this is referred to as the control latency, \(\tau_d\). The effect of this latency can be incorporated into the control system by introducing a pure delay element \(D(s)\) on the control output, which can be expressed as

\[
D(s) = e^{-\tau_ds}
\]  

(1.1)

The zero-order-hold element introduced by the digital-analog conversion process contributes an additional delay that is dependent on the sampling frequency. The zero-order-hold can be expressed as

\[
ZOH = \left(1 - e^{-T_0s}\right)s
\]

(1.2)

where \(T_0\) is the sampling period. In the frequency region of \(\omega > \omega_s/4\), which is primarily where digital feedback control is concerned, the zero-order-hold can be approximated as

\[
ZOH \approx T_s e^{-\frac{T_s}{2}}
\]

(1.3)

with less than 10% gain error and no phase error [1]. With this approximation the sampling frequency dependent delay is clearly evident.

Using discrete-time signal theory from [9] and considering both the control latency and zero-order-hold effects, the z-domain transfer function representation from \(u[k]\) to \(y[k]\) is

\[
P_z(z) = \frac{Y(z)}{U(z)} = Z\left\{e^{-\tau_ds}(1 - e^{-T_s})P(s)\right\}
\]

(1.4)

where \(Z\{\cdot\}\) is the Z-transform operator. Using the zero-order-hold approximation from Equation 1.3, and assuming proper anti-aliasing design and sampling frequency selection, the discrete plant frequency response can be simplified to

\[
P_z(e^{j\omega T_s}) \approx e^{-j\omega(\tau_d + T_s/2)}P(j\omega), \text{ for } \omega < \frac{\omega_s}{4}
\]

(1.5)

Thus, the total delay resulting from using digital control is approximately \(\tau_d + T_s/2\). This has two effects on control performance. First, it causes a frequency dependent phase loss that limits the
achievable bandwidth of the closed-loop system. Second, $\tau_d$ limits the sampling rate because the current control cycle must complete before the next control cycle can begin.

A simple estimation of phase loss at the loop transmission cross-over frequency versus sampling frequency, shown in Figure 1.4, can be attained by assuming that the entire sampling period is used to complete the control cycle, that is $T_s \approx \tau_d$. This is a conservative estimate because the control latency will typically be less than the sampling period due to interrupt latency, task overhead, and control timing variations. For generalization, the sample frequency, $\omega_s$, in Figure 1.4 has been normalized by the system loop transmission crossover frequency, $\omega_c$.

![Figure 1.4: Estimation of digital control control system phase-loss at the loop transmission crossover frequency for various sampling rates. $\omega_s$ is the digital control sampling rate and $\omega_c$ is the loop transmission crossover frequency](image)

Figure 1.4 clearly shows the effect of control latency and sampling frequency on phase margin and thus the limitations on achievable system bandwidth imposed by digital control. It also shows the benefits of using a control platform capable of low latency and fast sampling. For example, if a control system requires a loop transmission crossover frequency of 10kHz, then a control platform capable of sampling at 50kHz will impose a phase loss of 108 degrees from implementation alone. If using a control platform capable of sampling at 500kHz, the resulting phase loss contribution from implementation decreases to only 11 degrees. Another study performed by White on hard-disk drives showed a similar relationship between sampling frequency, control latency, and system bandwidth [10]. He found that the achievable closed-loop bandwidth ranged from 330Hz to 5100Hz for sampling frequencies ranging from 4kHz to 64kHz.

**Control Cycle Timing Variations**

Control cycle timing variations arise from implementation factors such as resource sharing, interrupt handling, and task scheduling. As introduced above, the variations of the sampling instant and control output instant are referred to as sampling jitter $\tau_s[k]$ and control jitter $\tau_c[k]$, respectively. Since a real-time controller is a hard real-time system, it is the worst case task execution time that determines minimum control period. Therefore, the obvious effect of the control cycle timing variations is to further limit the
maximum sampling frequency based on the worst case sampling and control jitter values.

Control cycle timing variations can also degrade motion control performance and if large enough can cause instability. The relationship between jitter and motion control performance is a non-trivial problem and has resulted in a great deal of research in this area. It is also the focus of Chapter 4 and Chapter 5 of this thesis, which investigate the effect of jitter on positioning error. Since control and sampling jitter are a major topic of this thesis, existing work in this field is reviewed more extensively in Section 1.2.2.

Data Accuracy

There are three real-time controller implementation aspects that contribute additional error to a control system: (1) A/D conversion error, $e_{AD}$; (2) Controller computation error, $e_{CMP}$; (3) D/A conversion error, $e_{DA}$. Each of these error sources will limit the attainable precision of the control system and thus should be minimized.

The A/D conversion error arises from quantization due to the finite converter resolution and additional noise introduced by the converter electronics. To minimize this error the ADC signal-to-noise ratio should be greater than the signal-to-noise ratio of the sensor, and the full scale signal range of each should be matched. This requires appropriate selection of the ADC resolution as well as high quality analog design and implementation of the converter electronics.

The controller computation error arises from rounding errors due to the numerical data format used by the processor. These rounding errors can also effect the controller coefficients, causing the implemented pole and zero locations to differ from the designed locations which can result in unexpected controller characteristics [8]. To minimize this error the controller computations should use a numerical data format with sufficiently high resolution. The most suitable and flexible today are 32-bit or 64-bit floating point formats. To further avoid this issue controller coefficients should be scaled to reduce the spread between large and small coefficients [11].

Similar to the A/D conversion error, the D/A conversion error also arises from quantization due to the finite converter resolution and additional noise introduced by converter electronics. To minimize this error DAC resolution should be selected similar to the ADC resolution and care should be taken during electronic design and implementation to minimize the additional noise.

A more in-depth treatment of these error sources can be found in [8].

1.2 Related Work

This section considers two areas of related work relevant to this thesis. First, Section 1.2.1 reviews existing state-of-the-art real-time control platforms to establish the current capabilities of digital controller implementations. Second, Section 1.2.2 reviews prior research regarding the effects of control cycle timing variations (sampling jitter and control jitter) on control performance.
1.2.1 State-of-the-art Real-time Control Platforms

A real-time control platform is a computer capable of hard real-time execution used to implement a variety of digital controller algorithms. In the context used here a real-time control platform is not tied to a specific control application or controller topology.

From a hardware perspective a control platform includes input devices for acquiring data from system sensors, a processing unit for executing the control algorithm, and output devices for sending data to system actuators. From a software perspective it includes an embedded operating system for real-time controller execution, a communication engine for handling user interactions, hardware drivers for interfacing with the input-output devices, and development tools to support controller development/implementation and graphical user interface (GUI) development. Therefore, important metrics for evaluating control platforms extend beyond just performance and include:

- Control performance (sampling rate, control latency, data accuracy).
- Input-output performance, variety and flexibility.
- Controller algorithm flexibility.
- Controller development/implementation support and overall usability.

The control platforms reviewed next represent the state-of-the-art in terms of control performance. The discussion here is by no means an exhaustive review of available control platforms, as it is only intended to highlight the current capabilities of digital control.

Commercial Control Platforms

Three industry leading control platforms are xPC Target from Mathworks [12], RT-LAB from Opal-RT [13], and the DS1103 from dSPACE [14].

All of these feature a dual-body (host-target) processor architecture utilizing a dedicated real-time target computer for implementing the controller and a standard PC workstation as a host computer for running the GUI. The real-time target processor in these systems must still perform at least two tasks: (1) a periodic real-time controller task; (2) a communication task for interacting with the host computer. Consequently, the periodic real-time control task is performed as an interrupt service routine and some additional overhead must be accounted for each control cycle so the target computer remains responsive to host requests. This implementation aligns with the typical control cycle timing from Figure 1.2.

xPC Target utilizes a standard PC as a target computer, which then runs a custom kernel to facilitate real-time execution of the controller. As xPC Target systems are built around standard PC hardware they are able to support a wide range of input-output (I/O) cards from various vendors that connect via standard PC bus interfaces such as PCI. While this provides a very high degree of flexibility, it also imposes limits on the control cycle timing as read and write times alone on these standard buses can often introduce several microseconds of latency. This, combined with interrupt latency and communication overhead time are main limiting factors of the achievable sampling frequency for xPC Target systems.
According to the xPC Target documentation, control models can achieve sampling rates up to 50kHz. A special polling operation mode can also be used, eliminating communication with the host machine while enabling sampling rates up to 100kHz to be reached; however, eliminating host communication is not an acceptable option for most control applications.

RT-LAB also utilizes a standard PC as a target computer, which then runs the QNX real-time operating system to facilitate real-time execution of the controller. Similar to xPC Target, RT-LAB supports a wide range of PC based standard plug-in I/O cards and thus suffers from the same limitations. According to the RT-LAB documentation controllers can reach sampling rates up to 30-50kHz under standard operation and up to 100kHz when host communication is eliminated.

The DS1103 from dSPACE utilizes a fully custom target hardware solution built around a 1GHz PowerPC processor, which then runs a custom real-time kernel. It also includes a variety of input-output interfaces making it suitable for use in many different control systems. This highly integrated hardware and software solution enables the DS1103 to achieve a higher sampling frequency than PC based systems. Based on the experience of this author, the DS1103 is able to execute controllers at sampling frequencies up to 125kHz.

All of these control platforms also include excellent support for controller development through integration with Simulink [15] and Real-time Workshop [16]. Integration with these software tools facilitates model-based controller development, automatic C-code generation, and automatic hardware implementation. Further, dSPACE also offers its own proprietary GUI development environment, ControlDesk, for laying out front panels of control, indicators, and graphs. RT-LAB offers similar GUI development functionality via integration with LabVIEW [17].

FPGA Motion Controllers

Traditionally, Field Programmable Gate Array (FPGA)s have mainly served as glue logic for interfacing I/O devices with processors. Recent and rapid advancements in FPGA technology have spurred research in standalone FPGA based motion controllers [18]. Unlike processors, which rely on a fixed pipelined hardware architecture to perform computations, FPGAs provide a highly configurable hardware architecture that is inherently parallel. This enables a control algorithm to be implemented in a highly parallel manner using dedicated hardware resources, resulting in very fast controller computations. Further, because FPGAs use dedicated hardware resources for each task the processor interrupt latency and GUI overhead time from the traditional control cycle in Figure 1.2 are eliminated. Combined, this allows very fast sampling frequencies to be achieved with FPGA based motion controllers.

In [19], Mutlu et al. investigated the implementation of several common motion control topologies on a FPGA and evaluated their timing performance. Using a state-of-the-art Virtex-5 FPGA they were able to achieve a 625kHz sampling frequency for a PID plus feedforward controller with encoder
feedback and PWM output. In [20], Osornio-Rios et al. demonstrated a sampling frequency in excess of 1MHz for a simple PID controller implemented on a Spartan-3 FPGA with encoder feedback and PWM output. In [21], Cho et al. presented a complete multi-axis control solution with a sampling frequency exceeding 100kHz on a Virtex-2 FPGA that includes encoder feedback, a PID controller, position interpolation calculation, velocity profile generation, inverse kinematic calculation of the plant, DAC output, and other low level functions.

While FPGA based motion controllers achieve excellent timing performance, they offer very little flexibility. All of the above controllers are restricted to the specific control topology discussed in each paper and offer very limited I/O resources. The added latency associated with ADC and DAC conversion are also not accounted for in these FPGA controllers because they do not include any analog interfaces. Implementing a new controller topology requires complete redesign of the implementation right down to the hardware level, which with existing development tools is a very timing consuming and challenging process. Further, FPGA resources are quickly consumed when floating-point computations are required, thus it may not even be physically possible to implement a more complex control algorithm that relies heavily on floating point computations.

These limitations mostly restrict FPGAs to simple controller implementations and final embedded solutions, making them unsuitable for supporting rapid controller prototyping and development. However, FPGA based motion controllers are still a relatively new technology and offer a lot of future potential once some of these limitations have been overcome and more flexible controller development tools are established. Recently, steps have already been taken in this direction by Celanovic et al. who introduced a FPGA based power electronic hardware-in-loop simulator capable of 1MHz sampling frequency and 1µs latency [22]; however, it does not include any analog I/O interfaces.

Thunderstorm

In 2005, Lu introduced the Thunderstorm real-time controller [1]. It utilized a novel triple-body (host-servant-target) processor architecture built around TigerSHARC-101 digital signal processor (DSP) s and was able to greatly improve controller timing performance relative to the top performing dual-body systems, demonstrating a 1MHz sampling frequency with 1.6µs control latency for a relatively complex control algorithm. Thus the Thunderstorm real-time controller is able to offer timing performance competitive with the fastest FPGA based real-time controllers along with the algorithm flexibility and development ease of a processor based real-time controller.

The triple-body architecture divides the typical control cycle from Figure 1.2 among three processor tiers: a host, servant, and target. The host computer or processor is responsible for running the GUI used to interact with the control model and facilitates signal monitoring, parameter tuning, and data logging for the user. The servant processor is tightly coupled with the target processor, through either shared global memory or dual-port memory, and handles all requests sent from the host computer. Since the servant processor is outside of the time critical control loop, user requests from the host can be handled without adding overhead to the control cycle. Further, the target processor is then left only with the task of executing the controller and thus can be run in polling mode. As a result, the interrupt latency $T_{INT}$
and overhead time $T_{OVR}$ is eliminated from the control cycle executing on the target processor, allowing the sampling frequency to be significantly increased.

The Thunderstorm real-time controller was originally designed specifically for controlling an ultra fast-tool servo also created by Lu in [1], thus in its current form it lacks the input-output interface flexibility and development tools to be easily adapted to other applications. However, the superior control timing performance demonstrated by Thunderstorm has provided strong motivation to develop a complete control platform solution based on the triple-body processing architecture that can act as an enabling technology for a wide range of high-precision high-speed control applications. The development of this new real-time control platform is a major topic of this thesis and is presented in Chapter 2 and Chapter 3.

1.2.2 Sampling and Control Jitter

Usually, sampling jitter and control jitter are assumed small enough to have negligible effects on the closed-loop system performance. Jitter issues have mostly received attention in networked control systems and distributed control systems, in which the sensor node, control calculation node, and actuator node are connected via a network. Networked control can be a cost-effective solution to systems with a large number of sensors and actuators, such as process automation, but such systems can experience large variable delays. Stability and robustness can be a major concern in these systems due to the large magnitudes of random delay and jitter, thus stability criteria for network controlled system with jitter has been investigated extensively [23] [24] [25] [26].

In motion control systems, although networks are widely used for user interface communication and transferring motion trajectory information, the system feedback loop consisting of sensor data acquisition, control calculation, and actuator update is highly localized. Therefore, instability caused by sampling jitter and control jitter is rarely an issue in motion control systems.

Measurements have shown that jitter typically ranges from hundreds of nanoseconds to tens of microseconds for commercially available real-time controllers. For example, a modern digital motion controller running real-time Linux has shown several microseconds of jitter [27] and a National Instruments CompactRIO has shown 40µs of jitter for a 1kHz control loop [28]. Measurements performed in Appendix A of this thesis show the jitter on an xPC Target controller to be 810ns RMS, and the jitter on a dSPACE DS1103 controller to be 160ns RMS. What is of interest is how this small amount of jitter affects the performance of a motion control system.

To facilitate time-variant analysis and simulation of control systems, Cervin, Lincoln et al. have created True Time and Jitterbug [29] [30], which are MATLAB based tools that can be used to evaluate a systems sensitivity to delay and jitter [31]. Antunes et al. presented a True Time simulation of a system with only control jitter and their results showed an increase in positioning error[32]. Zhang et al. have also presented simulation results showing jitter can increase positioning error [33]. Further, from simulations by Marti et. al an increase in step response overshoot can be observed [34].

There have been few experimental results reported in literature to actually demonstrate the effect of jitter on motion system performance. One of the few examples is a motor speed experiment conducted
by Kobayashi et al., which compared a case with fixed sampling period to a case with varying sampling period [35]. Their results showed a small difference in speed error for these two cases as other error sources appear to dominate the system.

Approximate modeling has been conducted by Boje to better understand effect of jitter on a digital control system [36]. He presented an approximate disturbance model in the w-domain for sampling jitter and control jitter by using a Tustin approximation to represent discrete-time signals. Based on this approximate model he then performed simulations to show jitter caused a disturbance to the system.

In order to reduce jitter-induced problems, work has been carried out in both real-time computing and control. Real-time computing has focused on task scheduling methods to directly reduce jitter magnitude [37] [38] [39] [40]. Control has focused on compensator design techniques, such as $H_\infty$ and LQG methods to improve system rejection to jitter disturbance [41] [42].

Another class of jitter compensators are implementation-aware controllers, which take advantage of runtime timing data (timestamps of actual sampling events) to dynamically compensate for jitter [43] [44] [45]. The main limitation of timestamp based controllers is that they introduce additional complexity and overhead into the control task, making them impractical for systems requiring fast sampling rates. Further, timestamps are unavailable in many controller implementations, limiting the applicability of this type of solution. Consequently, literature for these proposed methods only report on simulation results and not experimental results.

The lack of analytical predictions and experimental demonstrations in current literature regarding jitter’s effect on motion control system positioning error has motivated the research in Chapter 4 and Chapter 5 of this thesis. The objectives of the work presented here are to: (1) establish an approximate discrete model for systems with sampling jitter and control jitter; (2) provide a formula to analytically predict jitter’s effect on motion control system positioning error, without requiring simulation; (3) propose a simple add-on jitter compensator to mitigate jitter’s effect on regulation error, without requiring the existing motion controller to be changed; (4) experimentally demonstrate the effect of sampling jitter and control jitter on positioning error for both regulation and tracking scenarios.

1.3 Thesis Outline

This thesis includes two research contributions to the field of high-speed high-precision real-time control. First, it presents a new high-performance real-time multiprocessor control platform that is capable of achieving 1MHz control sampling rates with less than 6ns RMS control cycle timing variation and 16-bit data acquisition accuracy. Second, it investigates the effect of control cycle timing variations (sampling jitter and control jitter) on control performance, with an emphasis on degradation of motion control regulation and tracking performance. The aim of this investigation is to: (1) establish a better understanding of the effects of sampling jitter and control jitter on system performance, specifically positioning error in closed-loop motion control systems; (2) propose practical solutions to significantly attenuate additional positioning error due to jitter; (3) experimentally demonstrate the relationship between timing jitter and positioning error.

Chapter 2 covers the design and realization of the new multiprocessor control platform, which
extends the performance, functionality, and usability of the Thunderstorm real-time controller introduced by Lu [1]. It begins by establishing the design objectives for the control platform and then presents the electronic hardware design and supporting software design. The hardware design covers the overall platform architecture and also provides details on the motherboard development, daughterboard development, and overall system integration and realization. The software design covers the Simulink and LabVIEW libraries created to integrate the control platform with industry standard development environments, facilitating rapid model-based controller development and fast and professional graphical user interface development.

Chapter 3 presents the experimental results for the new multiprocessor control platform. First, several tests are performed to benchmark the performance and characterize the control hardware. Second, a case-study is carried out on a fast-tool to demonstrate the capabilities of the control platform on a real high-speed precision machine tool. This case-study includes the process of implementing a controller on the control platform as well as the achieved performance of the closed-loop fast-tool servo control system.

Chapter 4 begins the investigation of the effects of sampling jitter and control jitter on system performance, covering the modeling and analysis. First, a new discrete jitter model is developed that captures the effects of sampling jitter and control jitter on system performance. This model can be analyzed in the frequency domain using classical discrete-time control theory, enabling an intuitive understanding of jitter’s effects on the control system. Based on this model, analyses are carried out to determine the relationship between jitter and positioning error. Two scenarios are considered: (1) regulation error from jitter’s interaction with measurement noise; (2) tracking error from jitter’s interaction with a deterministic reference command. Using measured frequency responses or analytical transfer functions, the results of these analyses can be easily applied to any motion control system to estimate its sensitivity to sampling jitter and control jitter. Further, with insights obtained from these analyses, several practical methods to mitigate the positioning degradation due to jitter are discussed, including a new jitter compensator that can be easily added to an existing controller.

Chapter 5 presents simulation and experimental results to demonstrate the effect of sampling jitter and control jitter on positioning error for a real machine tool and also to validate the jitter model and analyses from Chapter 3. The simulations and experiments cover both scenarios mentioned above and clearly show that jitter can significantly degrade positioning performance, particularly for high-speed high-precision motion control systems.

Chapter 6 concludes this thesis with an overview of the presented results. It also discusses future work relating to the control platform and jitter investigation.

The main contributions of this thesis can be summarized as follows:

- The design and implementation of a new high performance real-time multiprocessor control platform. This includes electronic hardware development of a multiprocessor motherboard responsible for all core system functionality and an analog front-end daughterboard for interfacing the control platform to a wide range of sensors and actuators. Further, it includes the software development of a Simulink library to facilitate rapid controller development and a LabVIEW
library to facilitate rapid graphical user interface development.

- A new discrete model for sampling jitter and control jitter that can be analyzed using classical digital control theory, enabling an intuitive understanding of jitter’s effect on closed-loop system performance.

- A new and fast analytical method, based on the created model, to predict the additional positioning error introduced into a control system by sampling jitter and control jitter.

- A new jitter compensator to greatly mitigate the positioning error contribution from jitter for the case of motion regulation.

- Extensive experimental results that clearly demonstrate the effects of sampling jitter and control jitter on positioning error and also validate the model and analyses presented in this thesis.
Chapter 2

Tsunami Control Platform Design

This chapter presents the design of the Tsunami real-time control platform pictured in Figure 2.1. It is based on the Thunderstorm real-time controller introduced by Lu [1] and reviewed in the introduction, which demonstrated a 1MHz control sampling frequency with 16-bit data acquisition. Tsunami extends the performance, functionality, and usability of Thunderstorm, thus creating a new rapid control prototyping system designed to meet the needs of the most demanding high-speed high-precision control applications.

Figure 2.1: Picture of the completed Tsunami control platform hardware.

The design of Tsunami requires an extensive amount of both electronic hardware design and software design. Section 2.1 begins by establishing the design objectives for the system. Section 2.2 then discusses the hardware design. It overviews the overall architecture design, highlights how each of the design objectives were met, and then covers the key hardware components in further detail. Section 2.3 discusses the software design, covering the integration with Simulink for controller development and LabVIEW for GUI development.
2.1 Design Objectives

The overall goal of the Tsunami real-time control platform is to create a rapid control prototyping system that greatly increases the performance capabilities of digital control. It is intended as an enabling technology to support the controller design of high-speed high-precision systems and to facilitate research in multirate multiprocessor control theory; therefore, it must be flexible enough to meet the requirements of a broad range of control systems. Further, to be considered a rapid prototyping control system it must facilitate fast and easy controller implementation. Specifically this leads to the following design objectives:

- Achieve the fastest and most consistent control timing
- Include a wide variety of high performance input and output interfaces
- Be highly configurable to any control application
- Support multirate multiprocessor controller implementations
- Facilitate rapid controller development
- Facilitate fast and professional graphical user interface (GUI) development

Each of these objectives are expanded on below. Tsunami realizes all of these objectives through a combination of both electronic hardware design and software design.

Fast and Consistent Control Timing

Control performance for the real-time implementation of a digital controller is closely related to the control cycle latency, sampling frequency, and control cycle jitter. As discussed in the introduction, control cycle latency and sampling frequency both contribute phase loss to the control system and thus limit achievable closed-loop bandwidth. The control cycle jitter further limits the sampling frequency and can also degrade control performance. Control performance degradation arising from control cycle jitter is investigated extensively in Chapter 4.

Tsunami is intended to be an enabling technology for new and demanding high-speed high-precision control applications, therefore it must be capable of better timing performance than currently available platforms. Specifically the objective is to utilize the triple-body real-time controller architecture to achieve a sampling frequency of 1MHz with a control latency of only 1 microsecond for most common controller topologies. Further, the control cycle jitter is to be minimized to less than 1% RMS of the control cycle. For 1MHz sampling this amounts to less than 10ns RMS deviation of the control cycle timing.

Wide Variety of High Performance Input-Output Interfaces

The Tsunami control platform is intended to address a broad range of control applications, therefore it must be able to interface with all common types of sensors and actuators. This requires that the
platform support a wide variety of input-output interfaces, including analog inputs, analog outputs, digital inputs, digital outputs, encoder inputs, and pulse width modulation (PWM) outputs. Further, all of these interfaces must meet the performance requirements for high-speed high-precision control applications. Primarily, this requires that the analog inputs and outputs are able to operate at the desired sampling rate and are able to reach a precision below the control system noise floor. The objective of the analog inputs and outputs on Tsunami is to be able to operate above 1 mega-samples per second (MSPS) (matching the 1MHz sampling frequency objective) while maintaining 16-bit of precision.

**Highly Configurable to Any Control Application**

While the wide variety of high performance I/O available on Tsunami will be able to meet the requirements of most control applications, there will always be more specialized applications with highly custom requirements. Often these are performance driven systems for which standard components are unable to meet the design specifications. Since Tsunami is intended as an enabling technology for these specialized high performance applications, it must be easily configurable to meet any custom design requirements. Specifically, this requires that the control platform is able to accommodate custom input-output interfaces as well as custom pre and post processing of all input and output signals.

**Multirate Multiprocessor Controller Support**

Multirate multiprocessor control has the potential to increase control performance, as has been previously demonstrate by several researchers. In [46], Guo showed the benefits of multiprocessor based control for power converter applications. In [47], Tomizuka investigated multirate multiprocessor control for motion control applications. In [1], Lu improved the sampling rate of a fast-tool servo controller by harnessing the additional computation power offered by multiple processors and decomposing the controller into multiple sampling rates. This made it is possible to execute the most critical control task at a higher sampling rate than otherwise possible.

Multirate multiprocessor control strategies remain an open area of research with lots of future potential. This is particularly true with the recent mainstream availability of multicore processors. One objective of Tsunami is to facilitate research in multirate multiprocessor control, thus it should include at least 3 target processors in order to support many multiprocessor controller decomposition schemes. Further, the processors must be tightly-coupled to facilitate exact interprocessor timing synchronization and low latency data exchange.

**Rapid Controller Development Support**

The important aspects of rapid controller development (often referred to as rapid control prototyping) are model-based controller design, automatic code generation, and automatic hardware implementation. This allows the control engineer to focus entirely on the controller design and theory without having to spend time dealing with the implementation details. Further, this approach greatly accelerates the controller design cycle because effort no longer has to be invested into embedded coding, code
debugging, and hardware integration.

The objective of Tsunami is to fully support all the key aspects of rapid controller development. This means that a control model should be able to become an operational real-time controller on the Tsunami control hardware with only a few mouse clicks.

Fast and Professional GUI Development Support

A GUI is an essential part of any real-time control system as it provides a means for the control engineer or machine operator to interact with and monitor the operation of the system. Without a GUI there is no easy way to evaluate the system performance or to input commands into the system.

Since Tsunami is intended to be an application independent control platform it must include a means to develop custom GUls. Further, this task should be able to be completed in a few hours by a control engineer alone; therefore, the GUI development must offer a simple drag-and-drop method for laying out professional GUls and an intuitive and fast method for linking the various interface elements with the controller’s parameters and signals. To accomplish this task in a matter of hours it is essential that no knowledge of the underlying communication protocol an no text based programming be required by the control engineer to produce a fully functional GUI.

2.2 Hardware Design

The primary goal of the hardware design is to create a high-performance triple-body real-time controller with the flexibility to address a wide range of high-speed high-precision applications. This requires that the hardware meet the following design objectives: (1) Achieve fast and consistent control timing; (2) Include a wide variety of high performance I/O interfaces; (3) Be highly configurable to any control application; (4) Support multirate multiprocessor controller implementations. The designed Tsunami control platform architecture shown in Figure 2.2 is intended to meet all of these objectives.

The platform is divided into an application independent motherboard and two application specific daughterboards. The motherboard, detailed in Section 2.2.1, is the digital processing core of the platform, responsible for processing I/O signals, executing the control algorithm and communicating with a host computer. The daughterboards are application specific front ends for motherboard, responsible for providing all the I/O interfaces for connecting to sensors and actuators. The daughterboards shown are the Analog-DAQ, detailed in Section 2.2.2 and the nanoRAD, designed by Richard Graetz and detailed in [48]. Combined these two daughterboards provide a wide variety of high performance I/O interfaces able to satisfy the requirements of most control systems. If I/O requirements cannot be met by these daughterboards, they can be replaced with custom daughterboards.

The motherboard is built around a multiprocessing core consisting of 4 TigerSHARC-201 (TS-201) DSPs [49]. These are the most computationally powerful DSPs available from Analog Devices, each capable of 3.6 billion floating point operations per second (GFLOPS). Combined with the dedicated controller memory, the processing core is able to implement highly complex control algorithms with large data sets and look-up tables.

The triple-body processor architecture is realized by dedicating one of the DSPs as the servant
processor for host communication and dedicating the other three as target processors for control algorithm execution. Each of the target processors also have several dedicated timing control signals connected to the FPGA to facilitate the tight interprocessor synchronization required by multirate multiprocessor controllers.

The multiprocessing core is interconnected via a high-bandwidth low-latency multiprocessing bus for interprocessor communication. This bus also connects the processors to the FPGA I/O hub for interfacing with external peripherals. The FPGA serves two main functions: (1) it runs a host communication server responsible for buffering, encoding, and decoding the gigabit Ethernet packets used for host communication; (2) it implements the hardware drivers for the daughterboards connected to the expansion ports, interfacing them to the processing core. Due to the parallel nature of FPGAs and

Figure 2.2: Tsunami control platform hardware architecture diagram.
the dedicated host communication memory, these two tasks run in parallel and do not share resources; therefore, the operation of the host communication server does not influence the control cycle timing.

The daughterboards are connected to the FPGA via fully customizable expansion interfaces. Each expansion slot contains 84 fully definable signals, allowing the FPGA to interface directly to devices on the daughterboards without relying on an intermediary bus protocol. This ensures that read and write latencies to the input and output devices are kept to an absolute minimum. Further, this puts very few restrictions on the types of I/O devices that can be interfaced to the FPGA, allowing a very high degree of flexibility for creating custom daughterboards.

This description of the platform architecture from Figure 2.2 already highlights how most of the hardware related design objectives are met. Still, each objective is addressed specifically below.

**Fast and Consistent Control Timing**

The Tsunami hardware design uses four key means to achieve the targeted 1MHz sampling frequency with 1µs control latency and very low control cycle jitter: (1) Utilizing the Host-Servant-Target (triple-body) processing architecture; (2) Separating the timing critical control data path and the non-timing critical host communication data path; (3) Minimizing all factors that contribute to control latency; (4) Pipelining the ADC conversion.

Recall from the introduction, a typical control cycle for a real-time controller includes interrupt latency \( T_{INT} \), control task execution time \( T_{CTRL} \), and GUI/host communication overhead time \( T_{OVR} \). The total time required to perform all of these tasks limits the sampling frequency of the controller. By using the triple-body architecture \( T_{INT} \) is eliminated and \( T_{OVR} \) is re-allocated to the servant processor, thus the sampling frequency can be increased and is only limited by the control task execution time. This is shown in Figure 2.3, which compares the control cycle of the target processor of a typical real-time controller to that of a triple-body real-time controller.

With Tsunami, the triple-body processing architecture is implemented by dedicating one of the DSPs as the servant processor, leaving the other three as target processors. This realization provides a high degree of separation between the timing critical control data path and non-timing critical host communication data path as shown by Figure 2.4, which also contributes to ensuring very consistent control cycle timing. The timing critical control path flows from an input device on a daughterboard to a target processor via the FPGA I/O handling logic and multiprocessing bus. It then follows the same path in reverse to an output device on a daughterboard. The non-timing critical communication path flows from the Gigabit Ethernet to the servant processor via the FPGA host communication server and multiprocessing bus. It then follows the same path in reverse to transmit data to the host computer.

There exists a brief overlap of the two paths on the multiprocessing bus; however, the design ensures this has minimal impact on the timing critical control path by implementing the gigabit Ethernet host communication server within the FPGA and providing it with its own dedicated communication memory. This isolates all large data transfers and packet handling to the non-timing critical region. As a result, the servant processor only needs to perform small data accesses, keeping each multiprocessing bus occupation brief and thus ensuring a target processor will only be delayed a few bus cycles when
Figure 2.3: Comparison between the typical control cycle timing and the triple-body control cycle timing.

Figure 2.4: Diagram showing the separation of the timing critical control path and the non-timing critical host communication path.
attempting to read or write to daughterboard I/O.

Another key means used for achieving the targeted timing performance was to minimize the overall control cycle task time $T_{CTRL}$. Figure 2.5 shows a breakdown of the various elements that make up $T_{CTRL}$ for Tsunami. They are the ADC conversion time $T_{ADC}$, the ADC to processor reading time $T_{RD}$, the controller computation time $T_{CMP}$, the processor to DAC writing time $T_{WR}$, and the DAC conversion time, $T_{DAC}$.

![Diagram showing control cycle timing breakdown for Tsunami.](image)

**Figure 2.5:** Control cycle timing breakdown for Tsunami.

The ADC conversion time $T_{ADC}$ refers to the time it takes the ADC to convert the analog sensor signal to digital data that can be interpreted by the processor. Minimizing this element comes down to proper ADC selection. Of the available ADC architectures, SAR ADCs are most suitable for low latency applications as they contain no pipeline delays [50]. The Texas Instruments ADS8422 [51] 4MSPS ADC was selected because it is the fastest SAR ADC available that also satisfies the 16-bit data acquisition accuracy objective, specifying a conversion time of only 235ns. This is still a significant portion of the targeted 1µs overall control cycle period, but is the best achievable using available technology. This limitation can be partially overcome by initiating the ADC conversion from a hardware timer and pipelining it with the rest of the control cycle, as shown by Figure 2.6. As a result the sampling frequency can be further increased; however, the overall control cycle time $T_{CTRL}$ remains unchanged.

The ADC to processor read time $T_{RD}$ refers to the time it takes the DSP to read the conversion result from the ADC. Similarly, the processor to DAC write time $T_{WR}$ refers to the time it takes the processor to write the controller computation result to the DAC. Minimizing these elements requires implementing a low latency interconnection scheme between the input-output devices and the processor. On Tsunami this is accomplished by a combination of the low latency multiprocessing bus and direct connection between the input-output devices and the FPGA. Further, the flexibility of the FPGA allows an ADC read or DAC write to be treated as a simple memory mapped peripheral access on the multiprocessing bus. As a result an ADC read or DAC write operation can be completed in only 40ns each.

The controller computation time $T_{CMP}$ refers to the time it takes the DSP to complete the execution of the control algorithm. Since the control algorithm can vary greatly for different control systems, the only step to minimize this element that can be taken during hardware design is to use a very powerful DSP processor. The TS-201 DSP used by Tsunami is the most computationally powerful floating-point
DSP available from Analog Devices, capable of 3.6 GFLOPS. If all three target processors are utilized this amounts to a total of 10.8 GFLOPS.

Lastly, the DAC conversion time $T_{DAC}$ refers to the time it takes the digital-to-analog converter to convert the digital data to an analog signal for driving the system actuators. Minimizing this element comes down to proper DAC selection. The Linear Technology LTC1668 [52] 50MSPS DAC used here has a specified conversion time of only 20ns and also satisfies the 16-bit data acquisition accuracy design objective.

Based on the estimates here and experience gained working with the Thunderstorm real-time controller, Tsunami should achieve the targeted 1MHz sampling and 1$\mu$s control latency with adequate overhead time available to absorb any timing variations that arise.

The steps taken to achieve fast sampling rates and low control latency also ensure that the control cycle experiences very little timing variation. First, the triple body architecture eliminates the most prominent sources of timing variations, interrupt handling and task switching. Second, the controller computation time will be very consistent because the target processors are dedicated to only executing the control task and can be run in polling mode. Third, the separation of the time-critical control path and non-time critical communication path ensures highly consistent read and write times to input and output devices. Lastly, initiating the ADC conversion process with a hardware timer essentially eliminates all variation of the control cycle starting instant.

**Wide Variety of High Performance I/O Interfaces**

The two daughterboards designed for this project are intended to make Tsunami immediately applicable for most control applications. Combined they include all of the input-output interfaces specified in the design objectives. The Analog-DAQ daughterboard includes 20 channels of 16-bit ADCs and 36 channels of 16-bit DACs, allowing Tsunami to interface with most analog sensors and actuators. The 4MSPS ADCs and 50MSPS DACs are ideal for high-speed high-precision applications targeting up to 1MHz sampling rates. The 500 kilo-samples per second (KSPS) ADCs and DACs also make
Tsunami ideal for larger MIMO control applications. The nanoRAD daughterboard includes 6 channels of quadrature encoder inputs, 48 channels of PWM outputs, and 20 channels of configurable digital inputs or output, allowing Tsunami to interface to most types of digital sensors and actuators. Further, it contains 2 channels of cameralink inputs, extending the applicability of Tsunami to visual servo applications.

**Highly Configurable to Any Control Application**

The design objectives to make Tsunami highly configurable were the ability to change both the input-output hardware and the input-output firmware. This objective is realized by dividing the control hardware into an application independent motherboard and two interchangeable application specific daughterboards controlled by an FPGA, enabling additional daughterboards to be created when custom input-output interfaces are required. Further, each expansion interface consists of 84 fully configurable signals, placing very few restrictions on the types of devices that can be used on a daughterboard. The FPGA I/O hub then enables a custom hardware driver to be created for a custom daughterboard, facilitating fully customizable pre and post processing of input and output signals. Thus Tsunami’s functionality can be easily extended to even the most specialized control applications.

**Multirate Multiprocessor Controller Support**

To support multirate multiprocessor control, Tsunami contains 3 target DSPs to facilitate a variety of controller decomposition schemes. Successful multirate multiprocessor controller implementation also requires tight interprocessor timing synchronization and low latency interprocessor data exchange. The low latency data exchange is realized via the multi-processing bus, which enables a 32-bit floating-point number to be written from one processor to another in only 20ns. The tight timing synchronization is achieved using custom timing logic implemented in the FPGA, which makes use of dedicated timing control signals between each processor and the FPGA.

**2.2.1 Motherboard Design and Implementation**

The motherboard is an entirely digital board that is the most critical and complex component of the Tsunami real-time computer. It is the central core or ‘brain’ of the system, responsible for all of the data processing, algorithm execution, and host communication. It consists of all the application independent components required by the Tsunami control platform. A function block diagram of the motherboard is shown in [Figure 2.7](#).

The triple-body controller computation core is realized using 4 TigerSHARC-201 DSPs [49]. These DSPs are interconnected via a low latency 64-bit cluster bus operating at 125MHz. Also on this bus is 256MB of SDRAM memory dedicated for controller computation related tasks and a Virtex-5 FPGA [53] for interfacing the processing core with application specific daughterboards via the two expansion ports. Additionally, the FPGA, combined with the 64MB DDR and gigabit Ethernet, form a high-speed host communication interface for parameter tuning, signal monitoring, and data logging. Both the FPGA and DSPs have dedicated JTAG interfaces and static Flash storage for loading and storing application...
code, respectively. The RS-232 port, also connected to the FPGA, serves as a simple interface for configuring and debugging the motherboard.

The remainder of this section covers the most important design and implementation aspects of the motherboard, including the TS-201 processing core, the Virtex-5 FPGA I/O hub, the daughterboard expansion interface, and the printed circuit board (PCB).

**TigerSHARC-201 Processing Core**

The motherboard was built around the TS-201 DSP because it has many critical features that make it ideal for realizing the control platforms design goals. These include:

- **Very high floating point computation power**: The TS-201 is the most powerful DSP available from Analog Devices. Based on a superscalar harvard processor architecture and operating at 500MHz, it is able to perform up to 3.6 billion floating point operations per second. This facilitates extremely fast control algorithm computation.

- **Large internal memory**: The TS-201 includes 24Mbit of on-chip DRAM. This alone is enough to fully contain most control algorithms (application and data), thus much faster memory access times are possible than if external memory is relied on. Further, this also reduces the timing penalties and variations arising from processor cache misses. Overall, these shorter and more consistent memory accesses help to further shorten the control algorithm execution time.

- **Low-latency, high-bandwidth external bus**: The TS-201 includes a 64-bit proprietary external bus operating at 125MHz, referred to as the cluster bus. It has a total of bandwidth of 1GBps and can complete a read or write operation to an external peripheral in as little as 20ns, offering definitive latency advantages over competing gigabit serial point-to-point interconnects. This is an essential
feature for being able to achieve low-latency data acquisition from external devices such as ADCs and DACs.

- **Seamless multiprocessor support**: The cluster bus also includes control signals to seamlessly support up to 8 TS-201 processors. Further, all of the TS-201 processors on the bus share the same global memory space, allowing any DSP to access any others internal memory without interrupting the execution of the DSP being accessed. This is an essential feature for realizing the triple-body processing architecture and multiprocessor controller implementation.

- **External GPIO, timer, and interrupt pins**: These dedicated external signals enable tight timing synchronization to be achieved between the DSPs and FPGA without loading the cluster bus. This further helps facilitate the implementation of multirate multiprocessor controllers.

Operating at 500MHz and packaged in a 576-pin BGA, the TS-201 adds significant complexity to the PCB design. Some of the challenges it introduces are BGA signal fanout, extensive decoupling capacitor layout, input clock generation, and high-speed signal routing. Of these, the most challenging task is the signal routing of the cluster bus. With over 120 synchronous signals operating at 125MHz that interconnect 7 devices, it is pushing the capabilities of parallel bus interfaces. Maintaining signal integrity and meeting timing constraints thus both become major concerns, requiring that all the cluster bus signals be treated as transmission lines. Here, all signals are routed as 50Ω controlled impedance traces with matched trace lengths.

Based on the simulations and recommendations provided in [54], a star bus topology with RC termination is selected to realize the cluster bus. The star layout and trace length constraints created for this design are shown in Figure 2.8.

![Figure 2.8: Star bus topology and trace length specifications for the cluster bus.](image-url)
Virtex-5 FPGA I/O Hub

The FPGA is the central hub of the motherboard, responsible for connecting the processing core to all the required external peripherals. A Xilinx Virtex-5FX70 was selected for this design due to its extensive logic and I/O capacity, integrated PowerPC processor, hard Ethernet MACs and gigabit serial connectivity. With 11,200 slices, 71,680 logic cells and 360 configurable I/O, it can handle all of the IP cores required to operate the DDR memory, gigabit Ethernet and cluster bus, while still having plenty of excess capacity for daughterboard specific hardware drivers. The gigabit serial transceivers allow Ethernet connectivity to be realized with Serial Gigabit Media Independent Interface (SGMII), saving valuable pins for the I/O interfaces. The integrated PowerPC processor allows a dedicated TCP/IP server to be run on the FPGA for communicating with a host computer. This keeps background tasks on the servant DSP to a minimum and large TCP/IP packets off of the cluster bus, both factors that can degrade control timing performance.

As evident from the motherboard block diagram from Figure 2.7 almost every interconnect in the system runs through the FPGA, including the cluster bus, DDR SDRAM, gigabit Ethernet, daughterboard expansion interfaces, and RS-232. Since an FPGA is entirely configurable, this creates a very flexible system that can be highly specialized for a given application. A block diagram of the internal FPGA system design created for Tsunami is shown in Figure 2.9.

![Figure 2.9: Internal FPGA system architecture for Tsunami](image)

Each of the daughterboard drivers interface directly to the cluster bus allowing them to be mapped directly to the peripheral memory space of the TS-201 processing core. The cluster-bus-to-PLB bridge facilitates data transfer from the servant DSP to the host communication server running on the PowerPC. The host communication server then operates completely on the PLB bus, offloading all Ethernet related...
tasks from the cluster bus. The FPGA system also includes custom IP cores for controlling the operation of the DSP processing core. These include a reset controller and a timing controller that both make use of dedicated signals from the processing core.

Pin assignment of the 360 configurable I/O on the Virtex-5 FPGA was done with the goal of maximizing the number of pins available to each daughterboard interface. After allocating pins for all the required peripherals connected to the FPGA this left 84 I/O pins for each daughterboard interface. The specifics of these interfaces are expanded on next.

**Daughterboard Expansion Interface**

The motherboard has been designed to be application independent, thus it must be connected with a daughterboard to interface with application specific input and output devices such as ADCs, DACs, and encoders. Two functionally equivalent daughterboard expansion interfaces have been designed to connect any two daughterboards to the motherboard at a given time. Each interface provides 5V digital power to the daughterboard and 84 customizable pins that connect directly to the FPGA.

The custom pins have been divided into three groups to provide the flexibility to support both 2.5V and 3.3V voltage standards. The voltage level of each selectable group can be set using hardware jumpers located on the motherboard. Further, each group of pins are trace length matched and routed with both 50Ω single-ended controlled impedance and 100Ω differential controlled impedance; however, proper termination of the controlled impedance signals must be done on the daughterboard. The electrical specifications created for each of these interfaces is shown in Table 2.1.

**Table 2.1: Daughterboard expansion interface electrical specifications**

<table>
<thead>
<tr>
<th>Expansion Interface</th>
<th>Pin Group</th>
<th>Trace Length (mil)</th>
<th>Impedance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface A</td>
<td>24-pin 3.3V</td>
<td>7550 ± 50</td>
<td>50Ω SE</td>
</tr>
<tr>
<td></td>
<td>20-pin 2.5V or 3.3V</td>
<td>7800 ± 50 (±5 intra-pair)</td>
<td>50Ω SE and 100Ω Diff.</td>
</tr>
<tr>
<td></td>
<td>40-pin 2.5V or 3.3V</td>
<td>6750 ± 50 (±5 intra-pair)</td>
<td>50Ω SE and 100Ω Diff.</td>
</tr>
<tr>
<td>Interface B</td>
<td>24-pin 3.3V</td>
<td>3400 ± 50</td>
<td>50Ω SE</td>
</tr>
<tr>
<td></td>
<td>20-pin 2.5V or 3.3V</td>
<td>3700 ± 50 (±5 intra-pair)</td>
<td>50Ω SE and 100Ω Diff.</td>
</tr>
<tr>
<td></td>
<td>40-pin 2.5V or 3.3V</td>
<td>3350 ± 50 (±5 intra-pair)</td>
<td>50Ω SE and 100Ω Diff.</td>
</tr>
</tbody>
</table>

Mechanical specifications also had to be established to ensure the daughterboards are compatible with the motherboard and overall Tsunami enclosure. A drawing showing the created mechanical specifications for a Tsunami daughterboard are shown in Figure 2.10. The dimensions of a daughterboard are limited to 7.5in by 5in to ensure that it fits within the enclosure and does not interfere with the second daughterboard. The connector is a SAMTEC MOLC-145-M1-S-Q, which mates with the SAMTEC FOLC-145-M2-S-Q on the motherboard. The connector’s location is restricted to avoid component interferences with the motherboard.
Figure 2.10: Daughterboard expansion interface mechanical specifications

Printed Circuit Board

The motherboard PCB can be classified as a high-speed digital board as many of the components and interconnects operate in the hundreds of megahertz. For example, the cluster bus runs at 125MHz and the DDR SDRAM runs at 250MHz. These high-speeds are essential to achieving the targeted control timing performance; however, they introduce a significant amount of complexity into the routing and layout of the motherboard. Many high-speed digital design techniques are utilized to ensure the PCB is able to meet both the signal integrity and timing requirements of the motherboard components and interconnects. In [55], Johnson provides an excellent review of the issues and techniques associated with high-speed digital design. The techniques used here include controlling trace impedances, terminating signals, matching trace lengths, minimizing capacitive coupling, decoupling ICs, and shielding sensitive signals from EMI and crosstalk.

The designed PCB is constructed from FR-4 and has a total of 16 layers, ten dedicated to signal routing and 6 dedicated to power distribution. Ten signal layers were required largely due to the complexity of the cluster bus routing, which consists of 120 signals connecting the 4 DSPs, FPGA, and SDRAM. Six power layers were then required to accommodate the 10 different voltage levels present on the board and to maintain solid ground and power planes for high-speed signal routing. Figure 2.11 shows the stack-up of these 16 layers.

The top and bottom layers are used primarily for signal fanout. All high-speed signals, where signal
integrity is a concern, are routed on internal layers to isolate them from external EMI. Each internal
signal layer is adjacent to at least one continuous power or ground plane to ensure a minimum impedance
return path and allow for controlled impedance routing. The trace widths and core/pre-preg thicknesses
of this stackup were selected based on results obtained from Polar Instruments Si8000m field solver [56]
for 50Ω single-ended and 100Ω differential controlled impedance routing. The labels 'H' and 'V' in the
stackup refer to the direction of signal routing on that layer, either horizontal or vertical, respectively.
This orthogonal routing technique minimizes crosstalk between signals on adjacent layers, helping to
ensure signal integrity is maintained.

Another important aspect of the PCB design is the powering of all the components, which for this
board requires 10 unique voltage levels. Based on a worst case power analysis of all devices on the
motherboard, the power distribution system shown in Figure 2.12 was created. All of these levels are
distributed to components using the 6 dedicated power and ground layers of the PCB stackup. Most of
the supplies are realized using switching regulators because the digital circuits are generally insensitive
to switching noise and much higher efficiencies can be obtained relative to linear regulators, preventing
heat generation from becoming a concern. A few supplies are realized using linear regulators because
some devices also have secondary power inputs that demand clean analog power.

The completed 8in x 6in motherboard PCB with all of the key components labeled is shown in
Figure 2.13. The schematics and PCB layout were performed using Altium Designer [57]. The
board was manufactured by Sierra Circuits Inc. [58] and all fine pitch components were assembly
by Screaming Circuits Inc. [59]. The remainder of the assembly and testing was completed by this
author.
Figure 2.12: Motherboard power distribution system
Figure 2.13: Top view of the fully assembled motherboard PCB with key functional components labeled.
2.2.2 Analog-DAQ Daughterboard Design and Implementation

The Analog-DAQ daughterboard is designed to be a high performance analog front end for the Tsunami real-time computer. It includes a total of 20 analog inputs and 36 analog outputs, configured as shown by Figure 2.14. All of the inputs and outputs are 16-bit resolution with a single-ended ±10V input/output range. The four 4MSPS analog-to-digital converter (ADC)s and four 50MSPS digital-to-analog converter (DAC)s offer a high-speed, low latency interface for systems targeting sampling rates approaching 1 MHz. These channels are implemented using a 64-bit digital interface, allowing for simultaneous sampling and reading of all input channels as well as simultaneous writing and outputting of all output channels. The other 16 analog inputs and 32 analog outputs enable larger multi-input multi-output (MIMO) systems to also be supported by Tsunami. These are realized using 2 multi-channel ADCs, and 2 multi-channel DACs with serial read and write interfaces, thus simultaneous sampling of only two channels at once is possible with these.

![Figure 2.14: Analog-DAQ daughterboard functional block diagram](image)

The remainder of this section covers the most important design and implementation aspects of the Analog-DAQ daughterboard. These include the design of and selection of each analog interface as well as the design of the printed circuit board.
Component Selection

Selection of the 4 channels of 4MSPS ADCs and 4 channels of 50MSPS DACs was driven by performance. The goal was to achieve the minimum possible conversion times with at least 1MHz sampling, while also maintaining 16-bits of data precision.

The ADC selected for this task was the Texas Instruments ADS8422 [51], which is a 16-bit successive approximation register (SAR) ADC capable of 4MSPS sampling and 235ns conversion time. It is the fastest SAR ADC with a parallel digital interface currently available. The SAR architecture offers the lowest latency conversion times of all ADC architectures [50]. A parallel digital interface is used for two reasons: (1) it offers lower latency than an equivalent serial interface because the data does not have to be serialized and deserialized; (2) it keeps signal rates lower, meaning that high-speed signal integrity concerns can be avoided.

The required input range for the ADS8422 is a 0V-4.096V differential signal with 2.048V common mode. To attain the desired single-ended ±10V input range, a two stage analog front-end (AFE) is designed. A simplified circuit of this AFE is shown in Figure 2.15. The first unity gain op-amp stage creates a high impedance input to prevent the signal source from being adversely loaded by the input circuit. The second inverting op-amp stage scales the signal from ±10V to 0V-4.096V and converts it from single-ended to differential with a 2.048V common-mode. Additionally, the circuit contains a first-order RC anti-aliasing filter on the ADC inputs and 40V breakdown schottky diodes for over-voltage protection. The operation of the circuit was verified with a SPICE simulation using NI Multisim.

![Figure 2.15: Simplified circuit of the 4MSPS ADC analog front end.](image_url)

The DAC selected for the 4 channels of high performance analog outputs was the Linear Technology LTC1668 [52], which is a 16-bit DAC capable of 50MSPS sampling and 20ns output settling time (conversion time). Obtaining high conversion rates and low latency is much simpler for DACs than it is with ADCs, thus this specific DAC was selected more for its parallel digital interface and simple operation. The analog output from this DAC is differential current with maximum 1V peak-peak swing. A simplified schematic of the single stage AFE used to convert this to a ±10V single-ended low impedance output is shown in Figure 2.16.

The other 16 channels of ADCs and 32 channels of DACs are intended to make Tsunami suitable for large MIMO control systems. To incorporate this high channel count some trade-offs with performance were necessary. First the sampling rates had to be reduced to 500kSPS, resulting in longer conversion
Figure 2.16: Simplified circuit of the 50MSPS DAC analog front end.

times. Second, multi-channel ADCs and multi-channel op-amps had to be used to keep the component count down, resulting less isolation between channels and a higher noise floor. Unless the application requires a high channel count it is recommended that the 4MSPS ADCs and 50MSPS DACs be used as they offer superior performance.

The 16 channels of 500kSPS 16-bit analog inputs are realized using two Analog Devices AD7699s [60], which are 8 channel multiplexed SAR ADCs with a 1.6μs conversion time. These ADCs have a serial SPI digital interface, which although results in slower read times, was essential to keep the pin requirements within the 84 available on the motherboard connector. The AFE created for each channel is shown in Figure 2.17. Similar to the 4MSPS ADC AFE, it is a two stage front end, with the first stage providing high input impedance and the second stage scaling the ±10V single-ended input range to the 0V-4.098V required by the AD7699.

Figure 2.17: Simplified circuit of the 500kSPS ADC analog front end.

The 32 channels of 500kSPS 16-bit DACs are realized using two Analog Devices AD5360s [61], which are 16 channel voltage output DACs with a serial input interface. This DAC is capable of outputting ±10V directly, greatly simplifying the AFE design. Figure 2.18 shows a simplified schematic of the AFE, which consists of a unity gain op-amp to provide low output impedance and increased output drive strength.

All of the above AFEs utilize precision resistors, op-amps, and voltage references to help ensure 16-bit precision is maintained. Another critical factor for 16-bit precision is the PCB layout and power distribution system, which are discussed next.
PCB Design

The Analog-DAQ is a mixed signal board, as it contains both analog, digital, and mixed signal components. Unlike digital components, analog components are highly susceptible to noise, particularly when working with 16-bit resolution devices. Accordingly, for this board it was critical to utilize analog and mixed signal design techniques, which are discussed extensively in [50] and [62]. The steps taken to achieve low analog noise are sectioning the board into analog and digital regions, providing separate analog and digital ground planes, using linear power supplies for all analog circuitry, and using precision analog components whenever possible.

The designed PCB is constructed from FR-4 and has a total of 6 layers. The top and bottom layers are used for signal routing and component fanout, two internal layers are dedicated as a ground and power planes, and the last two layers are split between digital interface routing and analog power distribution. Figure 2.19 shows the stack-up of these 16 layers.

The PCB layout of the board is designed to separate the noisy digital components and signals from the sensitive analog components and signals. As shown by Figure 2.20, all the analog components are grouped around the edges of the board and the center of the board is dedicated for digital lines running to the motherboard connector. Also evident in this figure is the separation of the analog and digital ground planes, which share a single common connection point near the top of the board.

The power distribution system of the Analog-DAQ PCB must provide a total of 5 unique power levels and 4 precision voltage references. Figure 2.21 shows the power distribution system created for this board. There are four analog voltage levels, $\pm12V$ and $\pm5V$, for supplying the voltage rails of the converters and their analog front ends, and there is a single 3.3V digital level for driving the digital interfaces of the converters. The analog components and the mixed signal converters are all very susceptible to power supply noise; therefore, all of the analog rails are generated using linear regulators.
Figure 2.20: Analog-DAQ PCB layout and routing. Digital signals are kept to the center of the board, separated from the analog signals which are kept to the perimeter of the board.
and also utilize component level passive filtering.

![Diagram of Analog-DAQ power distribution system]

**Figure 2.21**: Analog-DAQ power distribution system

The completed 7in x 5in Analog-DAQ PCB with all of the key components labeled is shown in Figure 2.22. Similar to the motherboard, the schematic drawings and PCB layout was performed using Altium Designer [57], the PCB manufacturing was done by Sierra Circuits Inc. [58], and all fine pitch component assembly was done by Screaming Circuits Inc. [59]. The remainder of the assembly and testing was completed by this author.
Due to the limited size of a daughterboard PCB, imposed by the motherboard compatibility requirements, it is only possible to use high density headers for accessing the input and output interfaces. Further, the overall control platform is intended to be placed in a robust enclosure that does not provide access to the daughterboards. Accordingly, each daughterboard PCB is also accompanied by an I/O panel board. This PCB contains a convenient layout of connectors for accessing the inputs and outputs and is connected to the daughterboard using flexible ribbon cable. Thus it can be rigidly connected to the top of the Tsunami enclosure, providing convenient strain relieved I/O access.

For the Analog-DAQ, the I/O panel board consists of a 1in by 1in grid of BNC connectors. BNC
connectors were selected because they are easy to work with in a rapid prototyping environment, provide good strain relief, and have a well defined characteristic impedance. The completed Analog-DAQ I/O board is pictured in Figure 2.23. It is a simple two layer board with dimensions of 9.5in x 7.5in.

![Analog-DAQ I/O PCB board](image)

**Figure 2.23:** Analog-DAQ I/O PCB board.

### 2.2.3 System Integration

To integrate the created electronics into a standalone real-time controller a custom enclosure was created. Based on a standard 3.5in x 13in x 16in rack-mount computer case, this enclosure holds the motherboard, daughterboards, I/O boards, power supply, and cooling fans. A top view of the enclosure with the cover removed is shown in Figure 2.24 with the Analog-DAQ and the nanoRAD daughterboards.

Custom waterjet cutouts make all of the input-output interfaces easily accessible via the enclosure.
top panel and rear panel, as shown by Figure 2.25 and Figure 2.26 respectively. The top panel provides strain relieved access to all of the daughterboard application specific I/O via the I/O panel board created to accompany each daughterboard. The rear panel provides access to all of the application independent connections, including the gigabit Ethernet host communication port, RS-232 debug port, and JTAG configuration ports. Additionally, the rear panel includes a 120-240V AC power input, allowing the platform to be powered from any standard wall outlet.

The final result is a robust standalone real-time controller that can be quickly setup and connected to a wide range of sensors and actuators.
Figure 2.25: Top view of the Tsunami real-time control hardware with the case closed.

Figure 2.26: Rear view of the Tsunami real-time control hardware.
2.3 Software Design

The primary goal of the software design is to make it easy to take advantage of the control hardware, extending Tsunami from high performance real-time control hardware to a complete rapid control prototyping solution. It has been a team effort between Dr. Xiaodong Lu and this author, with the intent of meeting two important objectives: (1) Rapid controller development; (2) Fast and professional graphical user interface (GUI) development. Both of these objectives are met through integration with industry standard software packages, thus using Tsunami is an instantly familiar user experience.

As previously discussed, the important aspects of rapid controller development are model based controller design, automatic code generation, and automatic hardware implementation. Mathworks offers two widely accepted industry software packages that facilitate this, Simulink [15] and Real-time Workshop [16]. Simulink is a model based programming environment that allows controllers to be implemented directly as block diagrams and Real-time Workshop is a Simulink add-on that automatically generates embedded C-code from Simulink models. All of the embedded software created for Tsunami has been integrated with Simulink and Real-time Workshop, allowing Simulink models to be directly implemented on the real-time hardware. This is discussed further in Section 2.3.1.

Once the model is implemented on the real-time hardware a GUI is required to perform tasks such as signal monitoring, parameter tuning, and data logging. Achieving fast and professional GUI development requires tools to quickly layout a high-quality interface (buttons, numerical inputs, indicators, and plots) and a simple method to link these various interface elements with parameters and signals from the controller, without resorting to programming. With Tsunami, this is accomplished through integration with NI LabVIEW [17]. LabVIEW is a powerful graphical programming environment that is widely accepted in industry for data acquisition and visualization. Its greatest strengths are its ability to easily create interfaces for visualizing data and quickly interface with hardware for data acquisition, which it does by separating development into front panel layout and block diagram connectivity. Further, it includes a vast array of signal processing tools, allowing for dynamic post-processing of acquired signals prior to displaying them on the front panel. Tsunami’s integration with LabVIEW is discussed further in Section 2.3.2.

2.3.1 Simulink Controller Development Integration

To incorporate Tsunami specific features into Simulink controller models a Tsunami Simulink library, pictured in Figure 2.27 has been created. This library includes blocks for all of the input-output interfaces available on the Tsunami hardware as well as some more advanced tools to facilitate faster controller design. While the library is still growing, at this point these advanced tools include a digital signal analyzer block for performing system identification and an AFC controller block for quickly implementing the adaptive feed-forward cancellation control theory discussed in [1].

Features available for a given block can be easily configured by double-clicking on it. As an example, the quadrature encoder configuration window is shown in Figure 2.28. This allows the encoder channel to be selected, the reset mode and source to be configured, and additional features including timestamping and error detection to be enabled.
Figure 2.27: Tsunami Simulink library for controller development.

Figure 2.28: Tsunami Simulink library encoder block configuration.
To use this library the blocks are simply added to a controller model as shown in Figure 2.29. Here, the Tsunami ADC block has been used for acquiring the feedback signal from the sensor and the Tsunami DAC block has been used for outputting the computed control action to the actuator. At this point more complex models can be separated into subsystems that can be explicitly assigned to one of the 3 target processors available on the real-time hardware. Further, the sampling period for each processor can be set individually (limited to integer multiples of the fastest sampling rate), enabling multirate multiprocessor controller execution to be fully configured from within Simulink.

![Figure 2.29: Example of a Simulink controller model using the Tsunami library.](image)

The model build options for Real-time Workshop are set in the models simulation options, where a custom target computer has been created for Tsunami. Selecting this and building the model ensures that all the automatically generated C-code is compatible with the hardware. Further, this build process automatically generates executables for each processor on Tsunami, allowing the Simulink model to be directly implemented on the hardware without having to perform any programming. This automatic process involves compiling and linking the automatically generated model C-code with system level Tsunami C-code that includes all the application independent functions, such as hardware initialization, timing control, inter-processor synchronization and data exchange, and servant host communication.

The final result of Tsunami’s integration with Simulink enables controller models to be directly implemented on the control hardware, realizing the rapid controller development design objective of this project.

### 2.3.2 LabVIEW GUI Development Integration

LabVIEW enables fast and easy layout of high-quality graphical user interfaces, including buttons, numerical inputs and indicators, and graphs. The created Tsunami LabVIEW GUI Library, shown in Figure 2.30, links these GUI elements with controller parameters and signals on the real-time hardware and facilitates the gigabit Ethernet based communication. The library currently includes LabVIEW VI
blocks to connect/disconnect from the real-time hardware, monitor signals, tune parameters, and log signals. The library is still being expanded and will include more advanced blocks in the future, such as a trajectory generation block and a swept sine system identification block. All of the low level Ethernet based communication and packet handling is taken care of by the library, accordingly the user only needs to be concerned with which controller signals and parameters connect with which LabVIEW indicators and controls.

![Figure 2.30: Tsunami LabVIEW library for GUI development.](image)

During the automatic build process of the Tsunami specific Simulink model a custom version of the LabVIEW GUI library is created specific to the current project. As a result all the parameter and signal names from the Simulink model are immediately available when connecting to the library blocks in LabVIEW. For example, when a LabVIEW constant is connected to the Display Signal VI it automatically lists all the signals from the Simulink model, as shown by Figure 2.31.

To use this library the VI's are simply added to the LabVIEW block diagram that accompanies the GUI and are connected to the various interface elements. A simple example layout and block diagram for logging a signal from an implemented controller is shown in Figure 2.32. The GUI application begins by establishing a connection with Tsunami and then enters a loop which repeatedly logs the signal selected on the front panel while the log button is active. The disconnect button can then be clicked to disconnect from Tsunami and end the application.

Investing a little more time, the above simple program can be extended to a fully functional servo controller GUI as shown by Figure 2.33. This particular GUI allows a user to change the reference command, tune the AFC controller gains and view the current position and error. Further, it allows for any signal from the controller model to be logged and saved, displaying both its waveform and FFT plot.

The final result of Tsunami’s integration with LabVIEW enables customized high quality GUIs to be created without any programming or knowledge of the underlying communication engine, realizing
Figure 2.31: Connecting controller signals to the Tsunami Display Signal LabVIEW library block.

the fast and professional GUI development design objective of this project.
Figure 2.32: Example of a simple LabVIEW GUI to log a signal from an implemented Tsunami controller
Figure 2.33: Example servo controller GUI created using the Tsunami LabVIEW GUI library.
Chapter 3

Tsunami Control Platform Testing and Results

This chapter discusses the testing and results of the Tsunami control platform. It is divided into two sections, performance evaluation and a case study of a fast-tool servo (FTS). The performance evaluation presents a range of tests performed to characterize the platform’s I/O and timing performance. The case study covers the development, implementation, and results of a motion controller for a fast-tool servo to demonstrate the capabilities of Tsunami on a real control system.

3.1 Performance Evaluation

A variety of tests have been performed to evaluate the control platform, primarily focused on areas relating to control performance. Section 3.1.1 and Section 3.1.2 present the tests carried out to evaluate the performance of the analog inputs and outputs, respectively. Section 3.1.3 to Section 3.1.5 present the tests carried out to characterize the timing performance, including input-output latency, achievable sampling rate, and control jitter.

3.1.1 Analog Inputs

From a control standpoint the primary concerns for the analog inputs are the conversion latency and acquisition accuracy. The conversion latency is an internal property of the ADC and is unaffected by implementation, accordingly this is simply equal to the value provided in the ADC datasheet. For the 4MSPS ADCs and the 500kSPS ADCs used here, the conversion latencies are 235ns and 1.6µs, respectively.

The acquisition accuracy is highly dependent on implementation and can be evaluated by measuring the noise floor of each analog input. This is performed by grounding each input with a 50Ω terminator and evaluating the RMS, minimum, and maximum value of the ADC conversion result. The results are presented in units of least significant bit (LSB), which translates to 305µV/LSB since all of the ADC’s are 16-bit resolution with ±10V input range.

Figure 3.1 shows results obtained for the four 4MSPS ADCs. The worst case result is a 0.50 LSB
RMS with a ±1 LSB maximum deviation. The slight differences in the results can be attributed to component variations and PCB layout. All four of these inputs produce meaningful data on all 16-bits, with a worst case effective resolution of 15.5 bits. This is an excellent result, as it is common to see ADC implementations with several LSB RMS of noise, demonstrating that this control platform is very well suited for high-speed precision applications.

Figure 3.2 shows measurements for the two 8 channel 500kSPS ADCs. Only one channel is shown for each ADC because the results are very similar between internal ADC channels. The worst case result is 0.78 LSB RMS with a ±2 LSB maximum deviation. As expected, this is slightly worse than the results for the 4MSPS ADCs because the analog front end components are lower quality in order to support such a large channel count. However, 0.78 LSB RMS deviation is still an very good result and can be attributed to the clean analog power, mixed signal PCB layout, and precision voltage references used in this design.
Figure 3.1: Noise floors for the 4MSPS, 16-bit analog inputs
3.1.2 Analog Outputs

The analog outputs are evaluated based on their small signal and large signal step responses. The noise floors can not easily be measured due to insufficient resolution on the oscilloscopes available to this author; however, this is not a large concern because a control system is generally less sensitive to noise on the analog outputs than on the analog inputs. This is because it is the analog input noise that limits the sensor feedback accuracy for which the control system relies on to evaluate the error and calculate the control action.

A 1V step input is used to evaluate the small signal response of the DACs and a 20V step input, the full scale DAC output, is used to evaluate the large signal response. Since the results for all the 50MSPS outputs are very similar and the results for all the 500kSPS outputs are also very similar, only one of each is presented here.

Step responses for one of the 50MSPS outputs are shown in Figure 3.3. The 1V response has a rise time of 16ns, an overshoot of 12% and a settling time of about 50ns. This settling time is longer than the 20ns expected from the datasheet; however, it is still almost negligible relative to the overall targeted
control latency of $1\mu s$ and will therefore not adversely affect control performance. An increased level of noise is evident prior to the step response due to switching on the DAC digital interface. The full scale 20V step response has a rise time of 48ns, an overshoot of 5% and a settling time of about 90ns.

![Step response graphs](image)

**Figure 3.3: Step responses for one of the 50MSPS, 16-bit analog outputs**

Step responses for one of the 500kSPS outputs are shown in [Figure 3.4](#). The 1V response has a 90% rise time of $1.6\mu s$, no overshoot and a settling time of about $3.5\mu s$. The 20V response has a $24\mu s$ rise time, limited by the slew rate of the DAC. As expected these outputs have a much slower dynamic response than the 50MSPS outputs, thus it is recommended that the 50MSPS be used unless the control system requires more than 4 analog outputs.
3.1.3 Input-Output Latency

The input-output latency is the timing overhead associated with acquiring and outputting an analog signal with the Tsunami real-time controller. Here, it is defined as the time it takes to acquire a signal from one of the 4MSPS ADCs, convert the signal to a floating point number in preparation for computation, convert the floating point number back to a 16-bit unsigned integer in preparation for output, and output the signal to one of the 50MSPS DACs. Since this is a measure of overhead, no controller specific computations are performed. The input-output latency is measured by inputting a square wave from a signal generator to one of the 4MSPS ADCs and outputting it to an oscilloscope from one of the 50MSPS DACs, as shown by Figure 3.5. The signal output from the DAC is then...
compared to the original signal from the signal generator on the oscilloscope.

**Figure 3.5:** Experimental setup used to measure input-output latency.

The result of this measurement is shown in Figure 3.6. The input-output latency is the minimum time from the rising edge of the signal generator waveform to the rising edge of the DAC output waveform and is equal to 620ns. The variation in the output timing arises from the fact that the signal generator and ADC sampling are not synchronized and is equal to the control cycle sampling period, which for this experiment was set at 1µs. This result is larger than predicted during the hardware design because the measurement performed here also includes the floating point conversion times of the processor. Still, when utilizing ADC pipelining, there is around 615ns available for controller computation.

**Figure 3.6:** Measured input-output latency for the Tsunami control platform

### 3.1.4 Sampling Rate

The achievable sampling rate of the control platform is evaluated using a representative control algorithm. The algorithm consists of 3 analog-to-digital conversions, several addition, subtraction, and multiplication computations, 3 5th order controller transfer function computations, 3 sine and cosine computations, 3 square-root computations and 3 digital-to-analog conversions. All computations are performed in 32-bit single precision floating point.

The sampling rate is limited by the worst case control turnaround time, which is a measure of the time from the beginning of a control cycle to the end of a control cycle, including any overhead associated with background tasks such as data logging. Recall from Section 2.2 that Tsunami utilizes
ADC pipelining, meaning that ADC conversion time does not contribute to the control turnaround time. The measured turnaround time for slowest target processor is shown in Figure 3.7, indicating a worst case turnaround time of 986ns. Therefore a 1µs sampling period is possible without overrunning the system. It should be noted that this result was obtained when logging a single variable during controller execution and logging additional variables simultaneously would result in the turnaround time increasing.

![Figure 3.7: Measured turnaround time for the execution of the example control algorithm.](image)

### 3.1.5 Sampling and Control Jitter

Sampling jitter refers to the variation of the sampling instant of the ADC and control jitter refers to the variation of the control output instant of the DAC. These arise mostly from the control hardware implementation and are not very sensitive to the actual control algorithm being executed. The methods used to measure these are discussed in Appendix A.

Tsunami uses a hardware timer from the FPGA to initiate the ADC conversion rather than initiating the conversion from software, as is typical in real-time controllers. This timer is essentially jitter free and has a resolution of 8ns, therefore sampling jitter for Tsunami is approximately zero.

Figure 3.8 shows the measured control jitter for Tsunami, which has a RMS of 5.2ns and a maximum of 60ns. This excellent result contributes significantly to Tsunami’s ability to achieve hard real-time execution at up to 1MHz sampling rates and ensures jitter has a negligible effect on control performance, which is elaborated on in Chapter 4. The triple-body architecture and polling mode controller execution are the main reasons that such consistent timing is possible.
3.2 Case Study: Fast-tool Servo

The purpose of this case study is to demonstrate the process undergone to implement a controller on the Tsunami real-time controller and to showcase the achievable digital control performance on a real machine tool. The controller implementation process is covered in Section 3.2.1 and the control results in Section 3.2.2.

Fast-tool servo’s are machine tools used for manufacturing complex surfaces at nanometer resolution, which finds widespread application in areas such as the manufacturing of molds for light enhancing films and microlens arrays. The fast-tool servo used for this case study is an improved version of the one designed by Xiaodong Lu and presented in [63]. It is a SISO electro-magnetically driven linear actuator that uses a capacitive probe for position feedback. This FTS has a $50$ µm axial stroke and has previously demonstrated 1.6nm regulation error with a 10kHz loop transmission cross-over frequency using the Thunderstorm real-time controller, the predecessor to Tsunami. The combined requirements of high-precision and high-bandwidth make this FTS an ideal system for demonstrating the Tsunami control platform.

The experimental setup, pictured in Figure 3.9, consists of the FTS actuator with capacitive probe sensor, a linear power amplifier, the Tsunami real-time controller, and a host computer. The power amplifier is a custom 1kW linear amplifier that was also design by Lu in [1]. The capacitance probe is a MicroSense ADE5501 which has a bandwidth of 100kHz and a measurement range of 100µm. The host computer has a gigabit Ethernet connection for communication with Tsunami and is running Mathworks Simulink for controller development and NI LabVIEW for GUI development.

A simplified block diagram of the control system is shown in Figure 3.10, where $r(k)$ is the reference command, $y(t)$ is the actuator position, $e(t)$ is the feedback error, and $u(t)$ is the controller output.
The control structure used is a combination of a loop-shaping feedback compensator and an adaptive feedforward cancellation (AFC) feedforward compensator.

The measured small signal frequency response for the FTS from amplifier current command $u(t)$ to actuator position $y(t)$ is shown in Figure 3.11. Up to 2kHz the response is flat as a result of the bearing stiffness. There is then a resonance at 2kHz due to interaction between the bearing stiffness and moving mass assembly and from 2kHz to 35kHz the response is dominated by the moving mass assembly, exhibiting free mass characteristics with a -40dB/decade slope. Around 40kHz there is a strong non-collocated resonance that limits the achievable closed-loop bandwidth. Beyond 45kHz the response becomes dominated by the systems electrical dynamics.

A loop-shaping feedback compensator, $C_{LS}(s)$ is designed in the continuous domain to close the position loop. It consists of an integrator with a break frequency of 2kHz for zero steady-state error, a double lead-lag element centered at 10kHz for adding phase margin, and a low-pass filter at 100kHz for attenuating high-frequency noise. Analytically this controller can be expressed as

$$C_{LS}(s) = 646e4 \times \frac{(s + 12566)}{s} \times \frac{(s + 2.094e4)(s + 3.142e4)}{(s + 1.885e5)(s + 1.257e5)(s + 3.142e5)}$$

(3.1)
Figure 3.11: Measured plant frequency response for the fast-tool servo from current command (A) to actuator position (µm).

Figure 3.12 shows the frequency response of $C_{LS}(s)$. The continuous controller is converted to a discrete controller using the matched pole-zero method [6] with a sampling rate of 1MHz, which was selected to minimize the phase loss arising from digital control and to demonstrate the capabilities of the control hardware.

The frequency responses for the negative of the loop transmission and the closed loop are shown in Figure 3.13 and Figure 3.14, respectively. The resulting control system has a loop transmission crossover frequency of 10kHz with a phase margin of 35°. Further it achieves a -3dB closed loop bandwidth of 25kHz.

In addition to the loop shaping controller, an AFC controller is also incorporated into the system. In turning applications, the FTS tool path experiences a quasi-periodic pattern at harmonics of the spindle rotation frequency, thus including an AFC controller at harmonic frequencies of the spindle rotation can greatly improve tracking performance. Theory regarding the AFC design can be found in [1]. The specifications and parameters for the FTS actuator and control system are summarized in Table 3.1.

Next the implementation of this controller on Tsunami is discussed.
Figure 3.12: Loop-shaping controller frequency response for the fast-tool servo.

Figure 3.13: Negative loop transmission frequency response for the fast-tool servo control system.
3.2.1 Controller Implementation

The controller implementation tools created for the Tsunami real-time controller were discussed as part of the software design in Section 2.3. They include a Simulink library for controller development and a LabVIEW library for GUI development.

The first step to implementing the FTS controller on Tsunami is creating a Simulink model. The loop shaping controller described by Equation 3.1 and the AFC controller are placed into Simulink as shown in Figure 3.15. A switch block is added to activate the controller and appropriate gains are inserted to scale the input and output units to ADC and DAC voltages. The Tsunami Simulink driver blocks, shown in yellow, are used to add input-output functionality to the Simulink model. This system uses one 4MSPS analog input for capacitive probe feedback and two 50MSPS analog outputs to differentially drive the power amplifier. Additionally, the AFC is implemented using the Tsunami Simulink driver AFC block.

The model is explicitly divided onto two DSPs by splitting the model into two subsystems. The
loop shaping controller is implemented on one DSP and the AFC on another, as evident from the dotted lines in Figure 3.15. Dividing the controller in such a fashion is a necessity to reduce the computation time sufficiently to operate at 1MHz without overruns. After configuring the Real-Time Workshop simulation parameters, double-clicking the Tsunami build block from the created Tsunami Simulink Library initiates build processes and automatically generates all the embedded C-code. Further, this build process also updates the LabVIEW driver library with all the signal and parameters in the model.

Once the Simulink model has been built a LabVIEW VI must be created to graphically interact with the control system during operation. The front panel layout created for the FTS, using standard LabVIEW controls and indicators, is shown in Figure 3.16. It includes buttons to enable/disable the controller and AFC, numerical controls to change the reference command and tune the AFC gains, a table listing all the signals in the model and their values, and plots for displaying the waveforms and FFTs for up to two signals.

The block diagram for the LabVIEW VI, shown in Figure 3.17, uses the created LabVIEW library to link the controls and indicators to signals and parameters on the Tsunami target computer. The application begins by establishing a connection with the Tsunami target and then enters a loop with an event structure that reacts to user interactions on the front panel. This event based structure is more responsive and uses significantly less resources than a simple polling application; however, as evident from the block diagram it is more complicated to implement.

Once the embedded C-code has been generated and the LabVIEW GUI is complete, the controller is ready to be loaded and executed on the Tsunami real-time controller. All FPGA related VHDL is independent of the Simulink generated code and is automatically loaded from the Flash PROM upon powering on the Tsunami computer. Currently, the generated C-code must be manually compiled and loaded through JTAG using the VisualDSP++ environment; however, in the future this will also be automatically handled by the Simulink build process. Once the controller is loaded and running, the last
Figure 3.16: LabVIEW graphical user interface for the FTS control system.

Step is to run the LabVIEW GUI to monitor signals, tune parameters, and log data. Results obtained from implementing this controller are discussed next.
3.2.2 Control System Performance Results

The purpose of this section is to evaluate the control performance for the Tsunami implemented fast-tool servo control system.

First, the turnaround time for the implemented controller is shown in Figure 3.18. Recall, the control turnaround time is the time required to execute a control cycle, which for this 1MHz controller implementation must always be below 1µs to avoid overrunning the system. For DSP0, which executes the loop-shaping controller, the maximum turnaround time is 940ns and for DSP1, which executes the AFC controller, the maximum turnaround time is 915ns.

For control timing performance, a more important factor than turnaround time is the control latency, which is the time that elapses between the instant the ADC is sampled to the instant the DAC output is updated. The measured control latency is only 1µs for the implemented FTS controller, as shown by Figure 3.19. The variation in the output timing evident in Figure 3.19 arises from the fact that the signal generator and ADC sampling are not synchronized and is equal to the sampling period, as previously discussed in Section 3.1.3.

This result not only demonstrates the exceptional computation power and low latencies achievable with the Tsunami real-time controller, but also the benefits of utilizing multiple processors to implement a complex controller. Had the entire controller been implemented on a single DSP the turnaround time would have been near 2µs, the sum of the two turnaround times, and would have limited the sampling rate to around 500kHz. It should be noted that dividing the controller among multiple processors cannot be done arbitrarily and justification for the scheme used here has been previously provided in [1].
Figure 3.18: Control cycle turnaround times for the FTS controller implemented on Tsunami.

Figure 3.19: Control latency for the FTS controller implemented on Tsunami.
Unfortunately, multiprocessor controller decomposition schemes are beyond the scope of this Thesis and are left as part of the future work.

The dynamic performance of the FTS is evaluated by measuring the systems small signal step response. Figure 3.20 shows the response to a 500nm step input. The 10% to 90% rise time is 10\(\mu s\) with an overshoot of 26%.

![Figure 3.20: 500nm step responses for the FTS control system.](image)

Lastly, the position regulation and tracking performance of the FTS is evaluated. The regulation error, shown in Figure 3.21, has an RMS of only 1.35nm, giving the FTS a dynamic operating range of 38460:1. This excellent result can be attributed to the very low noise floor of the 4MSPS analog inputs, the linear power amplifier, shielding and choking of the capacitive probe cable, and low electrical coupling from proper signal grounding.

![Figure 3.21: Regulation error for the FTS control system.](image)
The reference command used to evaluate the tracking performance is a 6kHz, 8µm peak-peak sinusoid, shown in Figure 3.22. This command demands a ±650g acceleration from the FTS, near the limits of the system. Without the AFC compensators implemented the system is barely able to track this command, producing a tracking error of over 2.8um RMS which is dominated by the 6kHz component. Implementing the 4 AFC compensators completely eliminates this 6kHz component and its next three harmonics (12kHz, 18kHz, 24kHz), reducing RMS error to only 1.8nm RMS, as shown by Figure 3.23.

![Figure 3.22: 6kHz, 8µm peak-peak reference command.](image1)

![Figure 3.23: Tracking error for the FTS system for a 6kHz, 8µm peak-peak reference command with four AFC compensators.](image2)

Using Tsunami to control the FTS enables the full performance potential of the control system to be realized. First, the 1MHz sampling frequency and 1µs control latency made it possible to reach a closed-loop bandwidth of 25kHz. Second, the 4MSPS 16-bit precision analog inputs made it possible
to reach a dynamic operating range of 38460:1 with a regulation error of only 1.35nm RMS. Lastly, the high computation power and multiprocessing capabilities made it possible to implement multiple AFC compensators along with the loop-shaping controller, achieving a tracking error of only 1.8nm RMS for a 6kHz, 8µm peak-peak sinusoidal reference command.
Chapter 4

Jitter Modeling and Analysis

This chapter presents an investigation into the effects of sampling jitter and control jitter on system performance. First, Section 4.1 develops a new discrete jitter model that captures the effects of sampling jitter and control jitter on system performance. Based on this model, analyses are then carried out in Section 4.2 to determine the relationship between jitter and positioning error. Two scenarios are considered: (1) regulation error from jitter’s interaction with measurement noise; (2) tracking error from jitter’s interaction with a deterministic reference command. Lastly, Section 4.3 proposes practical solutions to mitigate the effect of jitter in each scenario.

4.1 Modeling

The block diagram of a realistic digitally controlled single-input single-output system is shown in Figure 4.1. The plant input signal $u_p(t)$ is related to the plant output signal $y_p(t)$ by

$$\frac{Y_p(s)}{U_p(s)} = P(s)$$

(4.1)

where $P(s)$ is the plant transfer function in the s-domain, and $U_p(s)$ and $Y_p(s)$ are the Laplace-transforms of $u_p(t)$ and $y_p(t)$, respectively. The plant output signal is subsequently sampled by a non-ideal sampler to produce a discrete sequence

$$y_p[k] = y_p(kT_0 + \tau_s[k])$$

(4.2)

where $T_0$ is the mean value of the digital controller sampling period, $k$ is the integer index of sampling events, and $\tau_s[k]$ is the sampling timing deviation from an ideal sampler. In addition to the discrete plant output signal $y_p[k]$, the discrete feedback signal $y[k]$ also includes the noise component $n[k]$, which contains analog-to-digital (ADC) conversion noise, quantization noise, and sampled sensor measurement noise. The control error signal $e[k]$ is then generated by subtracting $y[k]$ from the reference
command $r[k]$ in the digital controller. The discrete control output $u[k]$ is then calculated as

$$\frac{U(z)}{E(z)} = C(z)$$

(4.3)

where $C(z)$ is the controller transfer function in the Z-domain, and $E(z)$ and $U(z)$ are the Z-transforms of the discrete signals $e[k]$ and $u[k]$, respectively. The discrete control signal $u[k]$ is finally converted by a non-ideal ZOH to the plant input signal

$$u_p(t) = u[k], \text{ for } kT_0 + \tau_d + \tau_c[k] < t \leq (k + 1)T_0 + \tau_d + \tau_c[k + 1]$$

(4.4)

where $\tau_d$ represents the mean latency from the sampler sampling instant to the ZOH update instant, and $\tau_c[k]$ is the update timing deviation from an ideal uniformly-spaced ZOH. Recall, these timing deviations $\tau_s[k]$ and $\tau_c[k]$ are referred as sampling jitter and control jitter, respectively.

**Figure 4.1:** A Digital control feedback system with non-ideal sampler and non-ideal ZOH

Although ideal samplers and ideal ZOHs are used almost exclusively in sampled-data control textbooks [6], they do not exist in reality as there is always sampling jitter, sampling-to-ZOH latency, and control jitter resulting from implementation. The measurement of jitter for several real-time computers used for implementing digital controllers is described in Appendix [A]. For commercially available control hardware jitter is found to typically range from hundreds of nanoseconds to tens of microseconds.

In order to illustrate the sources of these non-ideal control timing effects, Figure 4.2 shows a typical timing process for a digital controller. Each control cycle is initiated by the expiring event of a control cycle timer in the digital control hardware, which is then followed by the interrupt latency $T_{INT}[k]$ and task switching delay $T_{SW}[k]$ before the sampling of $y_p(t)$ occurs. The control process then needs to wait during the ADC conversion time $T_{ADC}[k]$. After reading the ADC result in time $T_{RD}[k]$, the control
output is computed in time $T_{CMP}[k]$ using the implemented control algorithm. This computation result is then written to the digital-to-analog converter (DAC) in time $T_{WR}[k]$. Finally, at the end of the DAC conversion time $T_{DAC}[k]$ the analog signal $u_p(t)$ is updated, which corresponds to the ZOH update event.

![Sequential timing process in a typical digital control cycle.](image)

**Figure 4.2:** The sequential timing process in a typical digital control cycle.

In such a process, sampling jitter is determined by the timing variation between the timer event and the sampling event,

$$\tau_s[k] = \tilde{T}_{INT}[k] + \tilde{T}_{TSW}[k]$$

(4.5)

Here, the symbol $\tilde{T}[k]$ represents the AC component of $T[k]$ (i.e. $T[k]$ subtracted by its mean value $\bar{T}$). The control cycle timer is usually a hardware device working at several hundred megahertz and can be considered a jitter free event (i.e. the events are perfectly spaced with a constant sampling time). Control jitter $\tau_c[k]$ is then determined by the accumulated temporal variation from the timer event to the ZOH update,

$$\tau_c[k] = \tilde{T}_{INT}[k] + \tilde{T}_{TSW}[k] + \tilde{T}_{ADC}[k] + \tilde{T}_{RD}[k] + \tilde{T}_{CMP}[k] + \tilde{T}_{WR}[k] + \tilde{T}_{DAC}[k]$$

(4.6)

Based on this analysis, both sampling jitter $\tau_s[k]$ and control jitter $\tau_c[k]$ are zero-mean variables. In addition, the variance of sampling jitter is smaller than that of control jitter. Lastly, the sampling-to-ZOH latency can be expressed as the mean delay from the sampler sampling to the ZOH update,

$$\tau_d = \bar{T}_{ADC} + \bar{T}_{RD} + \bar{T}_{CMP} + \bar{T}_{WR} + \bar{T}_{DAC}$$

(4.7)

This latency can be separated from the non-ideal ZOH in Figure 4.1 resulting in a pure delay element and a zero-latency ZOH with jitter as shown in Figure 4.3. The expression for the continuous control output $u(t)$ and plant input $u_p(t)$ are then

$$u(t) = u[k] \text{ for } kT_0 + \tau_c[k] < t \leq (k + 1)T_0 + \tau_c[k + 1]$$

(4.8)

$$u_p(t) = u(t - \tau_d)$$

(4.9)
The digital control system model in Figure 4.3 is time-variant and thus cannot be analyzed using classical sampled-data control theory. In order to investigate digital control systems with sampling jitter and control jitter using classical sampled-data control theory, discrete disturbance models to approximate the non-ideal sampler and ZOH are developed below.

**Figure 4.3:** Equivalent model of a digital control feedback system with non-ideal sampler and ZOH

### 4.1.1 Zero-order-hold with Control Jitter

In Figure 4.4(a), the control output signal $u(t)$ is compared with the signal

$$u^*(t) = u[k] \text{ for } kT_0 < t \leq (k + 1)T_0$$

which is the result of control sequence $u[k]$ passing through an ideal ZOH (i.e. the update times are perfectly evenly spaced). The difference between $u(t)$ and $u^*(t)$ due to control jitter is represented by a disturbance signal

$$g(t) = u(t) - u^*(t)$$

As shown in Figure 4.4(b), the $g(t)$ waveform is composed of a pulse train, which is zero everywhere except in the regions when the non-ideal ZOH with jitter leads or lags the ideal ZOH. While jitter is a discrete phenomenon, this disturbance is a continuous time signal with inter-sample dynamics. Assuming the sampling rate is much greater than the highest plant dynamics, which is typical for control systems, the disturbance $g(t)$ can be approximated as a piecewise-constant signal

$$g^*(t) = (u[k - 1] - u[k]) \tau_c[k]/T_0 \text{ for } kT_0 < t \leq (k + 1)T_0$$

This selection for $g^*(t)$ conserves the signal momentum (amplitude integration over time) within
each sampling period. Further, $g^*(t)$ can now be represented as the output of a discrete signal $g[k]$ passing through an ideal ZOH, as shown in Figure 4.4(c), where

$$g(k) = (u[k - 1] - u[k]) \gamma[k] = u[k] (z^{-1} - 1) \gamma[k]$$ \hspace{1cm} (4.13)

and

$$\gamma[k] = \frac{\tau_c[k]}{T_0}$$ \hspace{1cm} (4.14)

$\gamma[k]$ is referred to as the normalized control jitter and $u[k] (z^{-1} - 1)$ is simply the discrete derivative of the ideal control output. Further, the two ideal ZOHs from Figure 4.4(c) can be combined together as shown in Figure 4.4(d). Consequently, the ZOH with control jitter in Figure 4.3 can be replaced by this disturbance model and an ideal ZOH.
(a) Comparison between the output of a ZOH with control jitter and an ideal ZOH.

(b) Comparison of waveforms between the output of a ZOH with control jitter and an ideal ZOH

(c) Approximation of jitter disturbance signal $g(t)$

(d) Approximate model of ZOH with control jitter

**Figure 4.4**: Modeling of non-ideal ZOH with control jitter
4.1.2 Sampler with Sampling Jitter

In Figure 4.5(a), the non-ideal sampled plant output signal \( y_p[k] \) is compared with the signal

\[
y_p^*[k] = y_p(kT)\tag{4.15}
\]

which is the result of \( y_p(t) \) going through an ideal sampler (i.e. the sampling times are evenly spaced). The difference between \( y_p[k] \) and \( y_p^*[k] \) due to sampling jitter is represented by a disturbance signal

\[
h_p[k] = y_p[k] - y_p^*[k]\tag{4.16}
\]

Figure 4.5(b) shows the sampled discrete sequences of \( y_p[k] \) and \( y_p^*[k] \). As the sampling rate is usually much greater than the plant’s highest frequency of interest, the difference \( h_p[k] \) can be approximated by a linear interpolated prediction \( h[k] \), expressed as

\[
h_p[k] \approx h[k] = \tau_s[k] \left( \frac{y_p^*[k] - y_p^*[k-1]}{T_0} \right) = \lambda[k] (y_p^*[k] - y_p^*[k-1]) \tag{4.17}
\]

where

\[
\lambda[k] = \frac{\tau_s[k]}{T_0}\tag{4.18}
\]

\( \lambda[k] \) is defined as the normalized sampling jitter. Using this approximation, the discrete sequence \( y_p[k] \) sampled by a non-ideal sampler can be modeled by the block diagram in Figure 4.5, which incorporates the sampling jitter as a disturbance and uses an ideal sampler. This sampling model is very similar to the control jitter model from Figure 4.4(d), with the only difference being where the disturbance enters the control system.
4.1.3 Overall Discrete Jitter Model

Replacing the non-ideal sampler and the non-ideal ZOH in Figure 4.3 with the models developed in Section 4.1.2 and Section 4.1.1, results in the overall digital control system model shown in Figure 4.6. The effects of jitter are incorporated as two disturbances $h[k]$ and $g[k]$ injected into the system at the ideal sampler and ideal ZOH, respectively. Further, applying ZOH equivalence, the dynamic process from $u_0[k]$ through the plant to $y_p^*[k]$ can be represented as

$$P(z) = \mathcal{Z}\left\{ \left(1 - z^{-1}\right) \mathcal{Z}^{-1} \left\{ \frac{P(s)e^{-(\tau_d/s)}}{s} \right\} \right\}$$

where $\mathcal{Z}\{\cdot\}$ is the z-transform of the continuous system impulse response sampled with period $T_0$.  

Figure 4.5: Modeling of non-ideal sampler with sampling jitter
and \( U(z) \) and \( Y_p^p(z) \) are the z-transforms of \( u_z[k] \) and \( y_p^p[k] \), respectively. Assuming a proper anti-aliasing filter is implemented, the discrete domain frequency response of \( P(s) \) can be calculated as

\[
P_z(e^{j\omega T_0}) = \frac{P(j\omega) e^{-j\omega \tau_d} (1 - e^{-j\omega T_0})}{j\omega T_0} = P(j\omega) e^{-j\omega(\tau_d + T_0/2)} \text{sinc} \left( \omega T_0 / 2 \right)
\] (4.20)

As a result, the digital control system model with non-ideal sampler and ZOH from Figure 4.1 has been converted to the discrete model with ideal sampler and ZOH in Figure 4.7. This discrete-time model can now be analyzed using classical digital control theory, enabling an intuitive understanding of jitter’s effects on control performance.

Figure 4.6: Digital control system model with jitter disturbance inputs

At this point some insights can be obtained regarding jitter’s effects on control performance. First, the magnitude of the jitter disturbances are proportional to the ratio of absolute jitter over the sampling period, and thus high-speed systems that require faster sampling rates will be more susceptible to jitter. Second, the jitter disturbances are a result of derivative interactions with other system signals, such as the reference command and measurement noise, thus the higher frequency content of these other inputs will contribute most to the jitter disturbances. Lastly, the time domain multiplication that occurs as part of each jitter disturbance can also be viewed in the frequency domain as modulation, thus high frequency interactions between jitter and the other system inputs can result in low frequency disturbances. These insights will become more evident following the analyses in the next section.
4.2 Analysis of Jitter’s Effect on Positioning Error

As shown from modeling, the normalized sampling and control jitter $\lambda[k]$ and $\gamma[k]$ disturb the digital control system by modulating the discrete derivatives of the feedback signal and control output signal, respectively. Using the discrete model from Figure 4.7, this section analyzes the effects of the jitter disturbances for two scenarios: (1) regulation error resulting from jitter’s interaction with measurement noise $n[k]$; (2) tracking error resulting from jitter’s interaction with a reference command $r[k]$. The positioning error (regulation or tracking) $\varepsilon[k]$ is defined as the desired plant output (reference command) minus the actual plant output,

$$\varepsilon[k] = r[k] - y_p^*[k] \quad (4.21)$$

It should be noted that this definition of positioning error $\varepsilon[k]$ is different from the control error $e[k]$ (the desired plant output minus the sampled sensor feedback), due to the presence of measurement noise and sampling jitter. Positioning error $\varepsilon[k]$ has been selected for evaluation because it represents the control system performance better than the feedback error $e[k]$.

As jitter and measurement noise are both primarily random signals, their Fourier transforms do not exist; therefore, it is not possible to analyze the jitter disturbance model from Figure 4.7 using direct frequency domain techniques. However, from a statistical viewpoint, the autocorrelation of a random signal is an aperiodic sequence with a defined Fourier transform. Accordingly, a stochastic analysis can be performed to draw meaningful conclusions regarding the effect of jitter on positioning error. Here, normalized jitters $\lambda[k]$ and $\gamma[k]$, and measurement noise $n[k]$ are assumed to be stationary white noise with variances $\sigma_\lambda^2$, $\sigma_\gamma^2$, and $\sigma_n^2$, respectively. Their auto-correlation functions are then

$$\phi_{nn}[k] = \sigma_n^2 \delta[k] \quad (4.22)$$
\[ \phi_{\lambda}[k] = \sigma_\lambda^2 \delta[k] \quad (4.23) \]
\[ \phi_{\gamma}[k] = \sigma_\gamma^2 \delta[k] \quad (4.24) \]

where \( \delta[k] \) is the Dirac delta function, and \( \phi_{xx}[k] \) is the autocorrelation of signal \( x[k] \), as used for stochastic signal analysis in [9].

### 4.2.1 Regulation Error Analysis

For position regulation, the reference command can be assumed to be zero without loss of generality: \( r[k] = 0 \). Consequently, the positioning error in the regulation case (regulation error) reduces to

\[ \varepsilon[k] = -y^*_p[k] \quad (4.25) \]

By decomposing the regulation error into components of measurement noise \( n[k] \), control jitter disturbance \( g[k] \), and sampling jitter disturbance \( h[k] \), it can be expressed as

\[ \varepsilon[k] = \varepsilon_n[k] + \varepsilon_g[k] + \varepsilon_h[k] \quad (4.26) \]

where

\[ \varepsilon_n[k] = n[k] * \mathcal{Z}^{-1} \left( \frac{P_z(z)C(z)}{1 + P_z(z)C(z)} \right) \quad (4.27) \]
\[ \varepsilon_g[k] = g[k] * \mathcal{Z}^{-1} \left( -\frac{P_z(z)}{1 + P_z(z)C(z)} \right) \quad (4.28) \]
\[ \varepsilon_h[k] = h[k] * \mathcal{Z}^{-1} \left( \frac{P_z(z)C(z)}{1 + P_z(z)C(z)} \right) \quad (4.29) \]

Here, \( * \) is the convolution operation, and \( \mathcal{Z}^{-1} (\cdot) \) is the inverse z-transform operation. Considering that \( \gamma[k] \) and \( \lambda[k] \) are typically only a few percent, second-order and higher interactions are negligible. This simplifies the expressions for the jitter disturbances to

\[ g[k] = \gamma[k] \left[ n[k] * \mathcal{Z}^{-1} \left( \frac{C(z)(1 - z^{-1})}{1 + P_z(z)C(z)} \right) \right] \quad (4.30) \]
\[ h[k] = \lambda[k] \left[ n[k] * \mathcal{Z}^{-1} \left( \frac{P_z(z)C(z)(z^{-1} - 1)}{1 + P_z(z)C(z)} \right) \right] \quad (4.31) \]

Therefore, \( g[k] \) and \( h[k] \) are white noises with variances equal to

\[ \sigma^2_g = E \left( g^2[k] \right) = \sigma^2 \sigma^2_n \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{C(e^{j\Omega})(1 - e^{-j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} \right|^2 d\Omega \quad (4.32) \]
\[
\sigma_h^2 = E\left(h^2[k]\right) = \sigma_n^2 \sigma_f^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_c(e^{j\Omega})C(e^{j\Omega})(e^{-j\Omega} - 1)}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} d\Omega \quad (4.33)
\]

where \( E(\cdot) \) is the expected value operation. The power spectrum density (PSD) function, which is the Fourier transform of a signal's auto-correlation, can be computed for each regulation error component from Equation 4.26. Accordingly, the PSD of \( \varepsilon_n[k] \), \( \varepsilon_g[k] \), and \( \varepsilon_h[k] \) are respectively

\[
\Phi_{ee}^n(e^{j\omega T_0}) = \sigma_n^2 \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_c(e^{j\Omega})C(e^{j\Omega})(1 - e^{-j\Omega})}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} d\Omega \right) \left( \frac{P_c(e^{j\Omega})C(e^{j\Omega})}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} \right)^2
\]

\[
\Phi_{ee}^g(e^{j\omega T_0}) = \sigma_f^2 \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_c(e^{j\Omega})C(e^{j\Omega})(e^{-j\Omega} - 1)}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} d\Omega \right) \left( \frac{P_c(e^{j\Omega})C(e^{j\Omega})}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} \right)^2
\]

\[
\Phi_{ee}^h(e^{j\omega T_0}) = \sigma_h^2 \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_c(e^{j\Omega})C(e^{j\Omega})(e^{-j\Omega} - 1)}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} d\Omega \right) \left( \frac{P_c(e^{j\Omega})C(e^{j\Omega})}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} \right)^2
\]

For a properly designed control system, \( \frac{P_c(e^{j\Omega})C(e^{j\Omega})(e^{-j\Omega} - 1)}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} \) is much less than 2 for all frequencies. As a result,

\[
\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_c(e^{j\Omega})C(e^{j\Omega})(e^{-j\Omega} - 1)}{1 + P_c(e^{j\Omega})C(e^{j\Omega})} d\Omega < 4 \quad (4.37)
\]

Considering the normalized sampling jitter standard deviation \( \sigma_h \) in most digital control systems is less than 0.1, \( \varepsilon_n[k] \) and \( \varepsilon_g[k] \)'s PSDs can be compared as,

\[
\frac{\Phi_{ee}^h(e^{j\omega T_0})}{\Phi_{ee}^g(e^{j\omega T_0})} < 4\sigma_h^2 < 0.04 \quad (4.38)
\]

Therefore, sampling jitter has negligible effect on regulation error, and the total regulation error reduces to \( \varepsilon[k] \approx \varepsilon_n[k] + \varepsilon_g[k] \). As control jitter \( \gamma[k] \) and measurement noise \( n[k] \) are generally uncorrelated, the PSD of the regulation error can be expressed as

\[
\Phi_{ee}(e^{j\omega T_0}) = \Phi_{ee}^h(e^{j\omega T_0}) + \Phi_{ee}^g(e^{j\omega T_0}) \quad (4.39)
\]

From \( \Phi_{ee}(e^{j\omega T_0}) \)'s expression in Equation 4.35, it can be seen that control jitter operates primarily on high frequency controller gain to produce a low frequency disturbance, which is counteracted by the controller’s disturbance rejection response. Consequently, the presence of control jitter will contribute additional regulation error to the digital control system. The root-mean-square (RMS) regulation error
can be calculated as

\[
\sigma_L^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_L(e^{j\Omega})d\Omega = \\
\sigma_n^2 \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_z(e^{j\Omega})C(e^{j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} d\Omega \right)^2 + \\
\sigma_n^2 \sigma_f^2 \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{C(e^{j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} \right)^2 \left( \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{P_z(e^{j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} d\Omega \right)^2
\] (4.40)

In this result, the first term is the regulation error contribution from measurement noise and the second term is the contribution from control jitter. The overall regulation error magnitude is dependent on the measurement noise, normalized control jitter, controller gain, and controller disturbance rejection.

**Extended Discussion**

To obtain an intuitive understanding of jitter’s effect on regulation error it is beneficial to breakdown and consider the individual components of Equation 4.40. Overall, it can be seen that the error is a function of the plant, measurement noise, normalized jitter, and controller. Given that the plant and measurement noise are usually a system property, a control engineer can still influence the error contribution from jitter through controller design and implementation timing.

First, consider the term \( \frac{C(e^{j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} \). For \( \omega \ll \omega_c \) this follows the inverse of the plant response and for \( \omega \gg \omega_c \) it follows the controller response, where \( \omega_c \) is the loop transmission crossover frequency. The response of the discrete derivative term \( 1 - e^{-j\Omega} \), shown in Figure 4.8, has most of its signal energy concentrated near the Nyquist frequency \( \omega_N \). In control it is typical for the sampling frequency \( \omega_0 = 2\pi/T_0 \) to be at least 10 to 20 times the crossover frequency, so combining the above terms into \( \frac{C(e^{j\Omega})(1 - e^{-j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} \) essentially amplifies the controller magnitude near the Nyquist frequency and attenuates all low frequency content. This effect will be even greater following the squaring of this combined term, thus it can be concluded that the magnitude of the jitter disturbance due to interaction with white measurement noise is very sensitive to the high frequency controller gain.

Further, the term \( \frac{P_z(e^{j\Omega})}{1 + P_z(e^{j\Omega})C(e^{j\Omega})} \), commonly referred to as the controller disturbance rejection transfer function, causes the effect of jitter to appear as a low frequency disturbance where the controller disturbance rejection is minimum.

To summarize, jitter interacts with high frequency controller gain to produce a low frequency disturbance whose magnitude is dependent on the measurement noise, normalized jitter, high-frequency controller gain, and controller disturbance rejection. The expected RMS contribution to regulation error from this disturbance is given by Equation 4.40.
4.2.2 Tracking Error Analysis

In the tracking case the measurement noise is assumed to be zero \( n[k] = 0 \) and the reference command \( r[k] \) is a deterministic signal. By decomposing the positioning error (tracking error) into components from reference signal \( r[k] \), control jitter disturbance \( g[k] \), and sampling jitter disturbance \( h[k] \), it can be expressed as

\[
\varepsilon[k] = \varepsilon_r[k] + \varepsilon_g[k] + \varepsilon_h[k]
\]

(4.41)

where

\[
\varepsilon_r[k] = r[k] \ast 2^{-1} \left( \frac{1}{1 + P_z(z)C(z)} \right)
\]

(4.42)

\[
\varepsilon_g[k] = g[k] \ast 2^{-1} \left( \frac{-P_z(z)}{1 + P_z(z)C(z)} \right)
\]

(4.43)

\[
\varepsilon_h[k] = h[k] \ast 2^{-1} \left( \frac{P_z(z)C(z)}{1 + P_z(z)C(z)} \right)
\]

(4.44)

\( \varepsilon_r[k] \) is the tracking error when there is no jitter, \( \varepsilon_g[k] \) is the tracking error contributed by the control jitter disturbance, and \( \varepsilon_h[k] \) is the tracking error contributed by the sampling jitter disturbance. In motion control applications, repetitive command signals are widely used and can be viewed as the sum of \( M \) distinctive single-tone signals,

\[
r[k] = \sum_{m=1}^{M} R_m \sin \left( \omega_m kT_0 + \phi_m \right)
\]

(4.45)

where \( R_m \), \( \omega_m \), and \( \phi_m \) are the \( m \)-th signal component’s amplitude, frequency in rad/sec, and phase in rad, respectively. In accordance with the discrete model in Figure 4.7, the control jitter disturbance signal can be represented as

\[
g[k] = \gamma[k] u_\Delta[k]
\]

(4.46)
where \( u_\Delta[k] = u[k-1] - u[k] \). By ignoring second-order and higher interactions this can be expressed as

\[
\begin{align*}
  u_\Delta[k] &= r[k] * 2^{k-1} \left( \frac{C(z)(z^{-1} - 1)}{1 + P(z)C(z)} \right) \\
  \text{(4.47)}
\end{align*}
\]

The auto-correlation function of \( g[k] \) is then

\[
\phi_{gg}[k, m] = E(g[k]g[m]) = u_\Delta^2[k] \sigma_\gamma^2 \delta[m-k] \\
\text{(4.48)}
\]

Therefore, \( g[k] \) is a non-stationary white noise signal and its resulting regulation error contribution \( e_g[k] \) is a non-stationary stochastic signal. Although \( e_g[k] \)’s variance is time-varying, its mean value can be used to evaluate the effect of the control jitter disturbance on positioning error. This is calculated as

\[
\begin{align*}
  \overline{E(e_g^2[k])} &= \overline{E(g^2[k])} \left( \frac{1}{2\pi} - \right. \\
  &= \sigma_\gamma^2 u_\Delta^2[k] \left( \frac{1}{2\pi} \right) \left. \right) \\
  &= \sigma_\gamma^2 \left( \sum_{m=1}^{\infty} \frac{R_m^2}{2} \right) \left( \frac{C(e^{-j\omega T_0})(e^{-j\omega T_0} - 1)}{1 + P(e^{j\omega T_0})C(e^{j\omega T_0})} \right)^2 \left( \frac{1}{2\pi} \right) \left( \right. \\
  &= \sigma_\gamma^2 \left( \sum_{m=1}^{\infty} \frac{R_m^2}{2} \right) \left( \frac{C(e^{-j\omega T_0})(e^{-j\omega T_0} - 1)}{1 + P(e^{j\omega T_0})C(e^{j\omega T_0})} \right)^2 \left( \frac{1}{2\pi} \right) \left. \right) \left( \right)
\end{align*}
\]

where \( \overline{\cdot} \) represents a temporal averaging operation.

Similarly, the sampling jitter disturbance \( h[k] \) can be expressed as

\[
\begin{align*}
  h[k] &= \lambda[k]y_\Delta[k] \\
  \text{(4.50)}
\end{align*}
\]

where \( y_\Delta[k] = y_\Delta^* - y_\Delta^* \) is the discrete feedback signal difference. Ignoring second-order and higher terms this can be expressed as

\[
\begin{align*}
  y_\Delta[k] &= r[k] * 2^{k-1} \left( \frac{P(z)C(z)(1-z^{-1})}{1 + P(z)C(z)} \right) \\
  \text{(4.51)}
\end{align*}
\]

The auto-correlation function of \( h[k] \) is then

\[
\begin{align*}
  \phi_{hh}[k, m] &= E(h[k]h[m]) = y_\Delta^2[k] \sigma_\gamma^2 \delta[m-k] \\
  \text{(4.52)}
\end{align*}
\]

Again, like \( g[k] \), \( h[k] \) is a non-stationary white noise and its resulting regulation error contribution \( e_h[k] \) is a non-stationary stochastic signal. Thus the expected value of its variance is used to evaluate the
effect of the sampling jitter on positioning error, which can be calculated as

$$
E \left( e^2_h[k] \right) = \sigma^2_h \gamma(D) \left[ \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{P(z(e^{j\omega T_0}))C(e^{j\omega})}{1 + P(z(e^{j\omega T_0}))C(e^{j\omega})} \right|^2 d\Omega \right]
$$

$$
= \sigma^2_h \left( \sum_{m=1}^{M} \frac{R^2_m}{2} \left| \frac{P_z(e^{j\omega_m T_0})C(e^{j\omega_m T_0}) \left[ 1 - e^{-j\omega_m T_0} \right]}{1 + P_z(e^{j\omega_m T_0})C(e^{j\omega_m T_0})} \right|^2 \right)
$$

(4.53)

Lastly, as \( e_h[k] \) is a deterministic signal, the expected value of its variance is simply its mean-square value,

$$
E \left( e^2_h[k] \right) = \sum_{m=1}^{M} \frac{R^2_m}{2} \left| \frac{1}{1 + P_z(e^{j\omega_m T_0})C(e^{j\omega_m T_0})} \right|^2
$$

(4.54)

Generally, \( e_h[k] \) can be completely eliminated by designing infinite controller gain at frequency \( \omega_m \), therefore the remaining tracking error is a result of jitter’s interaction with the reference command. The overall tracking error magnitude is then dependent on the reference command noise, normalized control jitter, normalized sampling jitter, controller gain, and controller disturbance rejection.

### 4.3 Solutions to Mitigate Positioning Error from Jitter

There are several methods that can be used to mitigate sampling jitter’s and control jitter’s effect on positioning error. Since the control jitter disturbance enters the closed-loop as a disturbance at the plant input, one method is to increase the controller disturbance rejection capability. This attenuates the term \( \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{P_z(e^{j\omega})C(e^{j\omega})}{1 + P_z(e^{j\omega})C(e^{j\omega})} \right|^2 d\Omega \) from Equation 4.40, and Equation 4.49; however, stability constraints will impose limits on the attainable disturbance rejection of the controller.

A second method is to reduce the jitter magnitude directly by improving operating system task scheduling or switching to better controller hardware with less jitter. Several task scheduling methods for reducing jitter were discussed in the introduction [37] [38] [39] [40] and the magnitude of jitter for several high-performance controllers was compared in Appendix A.

A third method, useful for regulation only, is to attenuate the controller gain near the system Nyquist frequency \( \omega_N = \pi / T_0 \). This greatly attenuates the term \( \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{C(e^{j\omega}) \left( 1 - e^{-j\omega} \right)}{1 + P_z(e^{j\omega})C(e^{j\omega})} \right|^2 d\Omega \) from Equation 4.40 as the magnitude is primarily determined by high frequency signal content due to the high-pass filtering effect of \( 1 - e^{-j\Omega} \). This can be done by cascading a jitter compensator \( C_g(z) \), which consists of a zero at the Nyquist frequency, with the existing controller. The expression for this jitter compensator is

$$
C_g(z) = \frac{1 + z^{-1}}{2}
$$

(4.55)

Figure 4.9 shows the frequency response of \( C_g(z) \), which has little effect on controller gain and phase for frequencies less than one-tenth of the Nyquist frequency, but greatly attenuates controller gain near Nyquist frequency. As a result, this jitter compensator can be directly cascaded with an existing controller, largely mitigating the effect of jitter on regulation without requiring redesign of the existing
controller. Note that $C_g(z)$ is generally not helpful in reducing jitter’s effect on tracking error because the frequencies of the reference command are mostly far less than the system Nyquist frequency.

**Figure 4.9:** Frequency response of the jitter compensator for mitigating control jitter disturbance on regulation error. $\omega_N$ is the system’s Nyquist frequency.
Chapter 5

Jitter Simulation and Experimental Results

This chapter presents simulation and experimental results to validate the jitter model and analyses from Chapter 4 and to demonstrate the effects of jitter on positioning error. Section 5.1 presents the simulation results and Section 5.2 presents the experimental results. For both cases two scenarios are considered: (1) regulation error resulting from jitter’s interaction with measurement noise; (2) tracking error resulting from jitter’s interaction with a deterministic reference command. Further, the effect of sampling jitter and control jitter are tested separately.

The system used for both the simulation and experimental investigation is the same fast-tool servo (FTS) used for the case study from Section 3.2, which is an improved version of the FTS presented in [63]. Recall, this FTS is a high-speed electro-magnetically actuated precision machine tool with capacitive probe position feedback and can achieve 50 µm stroke, 1.4 nm positioning error, and 750 g acceleration in continuous operation. The measured frequency response of the FTS is shown in Figure 5.1 along with an analytically fitted frequency response used for simulation.

The Tsunami real-time computer, introduced in Chapter 2, is used to control the FTS since it can achieve very high sampling rates with essentially zero baseline jitter (5.2ns RMS), which enables various amounts of additional jitter to be added to facilitate this investigation. Although execution of the control algorithm takes less than 1 µs on the custom control hardware, a sampling period of $T_0 = 4\mu s$ is used throughout all the experiments to accommodate the additional jitter. Note that the jitter percentage referred to throughout this chapter is relative to the sampling period, as it refers to the normalized jitter from Equation 4.14 and Equation 4.18. For example, 160 ns RMS jitter for a 4 µs sampling period is 4% jitter.

A loop-shaping base controller $C_B(s)$ is designed in the continuous domain to control the FTS, for which the transfer function is

$$C_B(s) = 183e5 \times \frac{(s + 7540)}{s} \times \frac{(s + 1.005e4)(s + 1.676e4)}{(s + 2.513e5)(s + 1.508e5)(s + 6.283e5)} \times \frac{(s^2 + 1885s + 3.553e10)}{(s^2 + 3.77e5s + 3.553e10)}$$

(5.1)
This continuous controller is converted to a discrete controller using the matched pole-zero method [6], yielding

$$C_B(z) = 6.5702 \times \frac{(z - 0.9698)}{(z - 1)} \times \frac{(z - 0.9606)(z - 0.9352)}{(z - 0.3659)(z - 0.5471)(z - 0.081)} \times \frac{(z^2 - 1.3449z - 0.9917)}{(z^2 - 0.8726z - 0.1904)} \quad (5.2)$$

The controller has three components: (1) an integrator active from 0 to 1.2 kHz; (2) a double-lead compensator to add phase from 1 kHz to 20 kHz; (3) a notch filter at 33 kHz to attenuate the plant resonance at this frequency. Its frequency response is shown in Figure 5.2. The resulting closed-loop frequency response is shown in Figure 5.3 and has a -3 dB bandwidth of 15 kHz.

Further specifics regarding the simulation and experimental setups are discussed in their respective sections.
Figure 5.2: Loop-shaping base controller frequency response for the fast-tool servo jitter investigation.

Figure 5.3: Closed-loop frequency response for the fast-tool servo control jitter investigation.
5.1 Simulation Results

A Simulink based simulation is carried out to validate the jitter model and analyses. There are two main reasons for beginning with simulations rather than proceeding directly to experimental validation. First, it ensures there are no unmodeled external disturbances or nonlinearities in the system that will skew the results. Second, the plant output $y_p[k]$ from Figure 4.3 can be directly observed. This is not possible in an experimental setup because measurement noise from the sensor is always introduced prior to observation.

Figure 5.4 shows the created Simulink model used to carry out the simulations. Sampling jitter and control jitter are directly injected by adding variable delays to the continuous portion of the model, which is then executed 100 times faster than the discrete subsystem to provide suitable jitter resolution. The added jitter is a repeating, normally distributed, random sequence generated using the MATLAB random function, with RMS ranging from 0% to 10% of the sampling period. Note that this simulation directly implements jitter as a timing variation and thus makes no assumptions about its effect on system performance.

![Simulink model used to simulate the effect of jitter on system performance.](sjcj_hybrid.png)

**Figure 5.4:** Simulink model used to simulate the effect of jitter on system performance.

The results presented next consider the two scenarios that were analyzed in Chapter 4: (1) regulation error from jitter’s interaction with measurement noise; (2) tracking error from jitter’s interaction with
a sinusoidal reference command. For both scenarios the positioning error is evaluated as $\varepsilon[k] = r[k] - y_p[k]$, as per Figure 4.7. Note that the simulation results presented are condensed as they are only intended to validate the model and analyses. Since the experimental results consider the same scenarios, more extensive treatment of is left for Section 5.2 to avoid repetition.

5.1.1 Regulation Error Simulation Results

For the regulation error simulation, the reference command $r[k]$ is set at 0 and the measurement noise source is a white 8 nm RMS signal. First, the control jitter is set to zero and the simulation is performed for sampling jitter ranging from 0% to 10% RMS of the sampling period. Figure 5.5 compares the simulation results with the analytically predicted RMS regulation error from Equation 4.40. As expected the sampling jitter has a negligible effect on regulation error.

![Figure 5.5](image)

Figure 5.5: Simulated and analytical RMS regulation error comparison for various amounts of sampling jitter.

Next, the sampling jitter is set to zero and the simulation is performed for control jitter ranging from 0% to 10% RMS of the sampling period. The analytical predictions from Equation 4.40 are compared with the simulation results in Figure 5.6. As predicted, the control jitter can significantly degrade the regulation performance of the system. For example, the RMS regulation error increased by 2.4 times, from 3.3nm to 8.1nm, for 8% control jitter.

The simulations are then repeated with the proposed jitter compensator implemented, $C(z) = C_B(z)C_g(z)$, to evaluate its effectiveness. These results are also plotted in Figure 5.6 for comparison with the previous results, clearly showing that the proposed jitter compensator largely mitigates the additional regulation error from control jitter. For example, the RMS regulation error for 8% control jitter has been reduced from 8.1nm to 4.8nm.

In all cases the simulation results match extremely well with the analytical results, validating both
Figure 5.6: Simulated and analytical RMS regulation error comparison for various amounts of control jitter, with and without the jitter compensator.

the created jitter model and stochastic regulation error analysis.

5.1.2 Tracking Error Simulation Results

For the tracking error simulation the noise source is set to zero and the reference command is a 6kHz sinusoidal signal with 4µm peak-peak amplitude: \( r[k] = 2 \sin(2\pi T_0 \times 6000k) \mu m \). When using feedback control only to track such a high frequency command, the total error will be dominated by the component from the reference command. A common way to overcome this is to add a feed-forward controller to the system. Here an adaptive feed-forward cancellation (AFC) controller \([64]\) is added to perfectly track the sinusoidal reference command, resulting in to overall controller being expressed as

\[
C(z) = C_B(z) \left(1 + C_{AFC}(z)\right)
\]

(5.3)

With this control structure the tracking error is reduced back to essentially the measurement noise floor (which for this simulation is zero), allowing the error contribution from jitter to be clearly observed.

For the first set of simulations the control jitter is set to zero and the sampling jitter is varied from 0% to 10% RMS of the sampling period. Figure 5.7 compares the simulation results with the analytically predicted RMS regulation error from Equation 4.53. Unlike for the regulation scenario, the sampling jitter now can significantly contribute to the overall tracking error, which agrees with the analytical prediction. For example, 8% sampling jitter adds 7.5nm RMS tracking error to the FTS control system.

Next, the sampling jitter is set to zero and the simulation is performed for control jitter ranging from 0% to 10% RMS of the sampling period. The analytical predictions from Equation 4.53 are compared with the simulation results in Figure 5.8. For the tracking scenario, the effect of control jitter is slightly
Figure 5.7: Simulated and analytical RMS tracking error comparison for various amounts of sampling jitter.

larger than the effect of sampling jitter, contributing an additional 9.1nm RMS of tracking error to the FTS control system.

Figure 5.8: Simulated and analytical RMS tracking error comparison for various amounts of control jitter.

Again, in all cases the simulation results match extremely well with the analytical results, validating both the created jitter model and stochastic tracking error analysis. Next a similar set of experiments are performed on the actual FTS to demonstrate the effects of jitter on a real system. A more detail
discussion of the results is also provided in the experimental section.

5.2 Experimental Results

This section presents the experimental results carried out on the fast-tool servo for the jitter investigation. Since the custom control hardware has nearly zero jitter, the desired amount of jitter is introduced by inserting a variable delay into the real-time controller execution. When conducting experiments for sampling jitter, the variable delay is inserted before the A/D conversion is initiated and the D/A conversion is initiated by a jitter free hardware timer. When conducting experiments for control jitter, the A/D conversion is initiated by a jitter free hardware timer and the variable delay is inserted before the D/A conversion is initiated. The added jitter uses pre-generated arrays of delay values to produce random white jitter with RMS magnitude ranging from 0% to 10% of the sampling period. Figure 5.9 shows the added jitter data and histogram for the case of 8% jitter. For experiments at other jitter magnitudes, the jitter data in Figure 5.9 is scaled accordingly.

![Jitter Data and Histogram](image)

**Figure 5.9:** 8% RMS normalized jitter data used for the experiments

Referring to Figure 4.7, it is the positioning error $\epsilon[k] = r[k] - y_p^*[k]$ and not the control error $e[k] = r[k] - y[k]$ that represents the control system performance. One challenge faced when attempting to experimentally measure jitter’s effect is that $y_p^*[k]$ is not readily available due to the presence of
measurement noise \( n[k] \) and sampling jitter disturbance \( h[k] \). To overcome \( h[k] \), a double sampling scheme is implemented in the custom real-time computer, as shown in Figure 5.10. Each control cycle, there are two ADC sampling events of the plant output \( y_p(t) \): one ADC with sampling jitter is used to acquire \( y[k] \) for the controller calculation; another ADC with zero sampled jitter is used to acquire \( y_m'[k] \) for positioning performance evaluation. However, \( y_m'[k] \) still contains measurement noise \( n^*[k] \) (the combination of ADC noise and sensor noise, 1.4 nm RMS). In the regulation experiment an additional white noise \( n_a[k] \) of 8 nm RMS is added, shown in Figure 5.10, in order to make \( n^*[k] \)'s contribution negligible, and therefore \( y_m'[k] \) can be used to approximate \( y_p^*[k] \). In the tracking experiment the reference signal’s amplitude is set at 4 \( \mu \)m pk-pk, which results in large enough tracking error to dominate the contribution from \( n^*[k] \).

![System block diagram of the experimental setup.](image)

5.2.1 Regulation Error Experimental Results

For the regulation error experiment, the reference command \( r[k] \) is set at 0 and the added measurement noise \( n_a[k] \) is a white stochastic 8 nm RMS signal. As a reference benchmark, Figure 5.11 shows the measured regulation error and its PSD for zero sampling jitter and zero control jitter. The measured 4.0 nm RMS regulation error is smaller than the added noise \( n_a[k] \) (8 nm RMS) because much of the high frequency noise content is filtered by the plant. As shown in Figure 5.11[b), the regulation error PSD is shaped similar to the system’s closed-loop frequency response of Figure 5.3 as predicted by the analytical PSD in Figure 5.11[c). The analytical PSD response in Figure 5.11[c) is calculated using Equation 4.39.
Figure 5.11: Regulation error experimental results for no jitter
Sampling Jitter’s Effect on Regulation Error

In this experiment the control jitter is set at zero and the sampling jitter is varied from 0% to 10% of the sampling period using the jitter data from Figure 5.9. Figure 5.12(a) shows the measured regulation error for 8% sampling jitter. There is no noticeable increase in both the measured error waveform and PSD compared to the 0% jitter reference case from Figure 5.11(b). This is consistent with the analytically predicted PSD in Figure 5.12(c) calculated from Equation 4.34 and Equation 4.36. As discussed in Section 4.2, the sampling jitter contribution to regulation error $\varepsilon_h[k]$ is much less than the measurement noise contribution to regulation error $\varepsilon_n[k]$. For other magnitudes of sampling jitter, the measured and analytical RMS regulation error is plotted in Figure 5.13, again confirming the earlier conclusion that the sampling jitter has negligible effect on regulation error.
Figure 5.12: Regulation error experimental results for 8% sampling jitter
Control Jitter’s Effect on Regulation Error

In this experiment the sampling jitter is set at zero and the control jitter is varied from 0% to 10% of the sampling period using the jitter data from Figure 5.9. Figure 5.14(a) shows the experimental measured regulation error for 8% control jitter. In comparison with the zero control jitter benchmark result in Figure 5.11, the 8% control jitter causes the RMS regulation error to increase by 90%, from 4.0nm to 7.7nm. Comparing their PSDs, the major difference in the frequency domain occurs around 1 kHz, which corresponds to the controller’s minimum disturbance rejection region. This result is consistent with the analytically predicted PSD in Figure 5.14(c), which is calculated from Equation 4.34 and Equation 4.35. The analytical PSD shows that the control jitter contribution to regulation error $\varepsilon_g[k]$ is dominant over the measurement noise contribution to regulation error $\varepsilon_n[k]$ in the frequency range from 100Hz to 10kHz, thus causing the total regulation error to increase. This result clearly indicates that control jitter’s interaction with measurement noise produces a low frequency disturbance that degrades position regulation performance.

When the proposed jitter compensator $C_g(z)$ is added to the base controller, $C(z) = C_B(z)C_g(z)$, the RMS regulation error greatly decreases from 7.7nm to 4.7nm, despite the 8% control jitter. This result is shown in Figure 5.15(a). A comparison between Figure 5.14 and Figure 5.15 shows that the proposed jitter compensator successfully suppresses the control jitter disturbance at low frequencies, and thus most of the additional error due to control jitter is eliminated.

The control jitter regulation error results are extended in Figure 5.16, which compares the analytical and measured RMS regulation error for control jitter ranging from 0% to 10% RMS, with and without the jitter compensator. The analytical prediction matches the experimental results very well and the jitter compensator significantly attenuates the effect of control jitter on regulation error.
Figure 5.14: Regulation error experimental results for 8% control jitter
Figure 5.15: Regulation error experimental results for 8% control jitter with the jitter compensator $C_g(z)$ implemented.
5.2.2 Tracking Error Experimental Results

For the tracking error experiment there is no added measurement noise, \( n_a[k] = 0 \), and the reference command is a 6kHz sinusoidal signal with a \( 4\mu \text{m} \) peak-valley amplitude:

\[
r[k] = 2 \sin(2\pi T_0 \times 6000k) \mu \text{m}.
\]

Generally, tracking at such a high frequency will result in the dominant tracking error contribution coming from the reference command component \( \varepsilon_r[k] \), as expressed in Equation 4.54. In order to show the jitter tracking error contribution, \( \varepsilon_r[k] \) should be eliminated by increasing the controller gain at 6 kHz to infinity. This can be done by adding an adaptive feed-forward cancellation (AFC) controller \([64]\), in the format of

\[
C(z) = C_B(z)(1 + C_{AFC}(z)) \tag{5.4}
\]

Here, \( C_{AFC}(z) \) contains four compensated frequencies at 6 kHz, 12 kHz, 18 kHz, and 24 kHz, each with a gain of 200. The AFC compensation at higher order harmonics of the reference command is used to attenuate the non-linearity of the FTS actuator. Figure 5.17 shows the tracking experimental results for no jitter, with and without \( C_{AFC}(z) \). After implementing the AFC controller, the tracking error was reduced by a factor of nearly 1000, from \( 1.4\mu \text{m} \) RMS to \( 1.6\text{nm} \) RMS, which is close to the measurement noise floor. From the tracking PSD comparison in Figure 5.17(c), the AFC effectively removes all error components at the reference signal frequency as well as at its higher-order harmonics. In all the following tracking experiments the AFC controller is implemented and the \( 1.6\text{nm} \) RMS tracking error with no jitter is used as a benchmark.
Figure 5.17: Tracking error experimental results for no jitter with and without AFC
Sampling Jitter’s Effect on Tracking Error

In this experiment the control jitter is set at zero and the sampling jitter is varied from 0% to 10% of the sampling period using the jitter data from Figure 5.9. Figure 5.18 shows the experimental measured tracking error and its PSD for 8% RMS sampling jitter. In comparison with the zero sampling jitter benchmark results in Figure 5.17, the tracking error RMS value increased by 6 times, from 1.6nm to 10nm. From the tracking error PSD in Figure 5.18(b) it can be seen that the major increase in tracking error due to sampling jitter occurs in the low frequency region.

![Figure 5.18: Tracking error experimental results for 8% sampling jitter](image)

The sampling jitter tracking error results are extended in Figure 5.19, which compares the analytical RMS tracking error, calculated from Equation 4.53, to the experimentally measured RMS tracking error for sampling jitter ranging from 0% to 10%. In all cases, the analytical results predict the trend of the experimental results with a small amount of mismatch. This is believed to be related to the FTS plant non-linearity which was not modeled in Figure 4.7 or included in the analysis. These
experimental results also indicate that the sampling jitter disturbance $h[k]$ can become the dominant source of positioning error, particularly for high-speed precision motion control systems.

![Graph showing measured vs analytical RMS tracking error comparison for various amounts of sampling jitter](image)

**Figure 5.19:** Measured and analytical RMS tracking error comparison for various amounts of sampling jitter

### Control Jitter’s Effect on Tracking Error

In this experiment the sampling jitter is set at zero and the control jitter is varied from 0% to 10% of the sampling period using the jitter data from Figure 5.9. Figure 5.20 shows the experimental measured tracking error and its PSD with 8% RMS control jitter. In comparison with the zero control jitter benchmark results in Figure 5.17, the tracking error RMS increased by 6 times, from 1.6nm to 9.9nm. From the tracking error PSD in Figure 5.20(b) it can be seen that the major increase in tracking error due to control jitter occurs in the low frequency region, particularly in the frequency range 100Hz to 10kHz where the controller’s disturbance rejection is lowest.

Figure 5.21 extends the tracking error results for control jitter, comparing the analytically predicted RMS tracking error, calculated from Equation 4.49, to the experimentally measured RMS tracking error for control jitter ranging from 0% to 10%. In all cases the results match very well, validating the presented models and analytical results for control jitter. In addition, the experimental results also demonstrate that control jitter can significantly degrade positioning performance, particularly for high-speed precision motion control systems.
Figure 5.20: Tracking error experimental results for 8% sampling jitter

Figure 5.21: Measured and analytical RMS tracking error comparison for various amounts of sampling jitter
Chapter 6

Conclusions

This thesis presented two research contributions to the field of high-speed high-precision real-time control.

First, it introduced the Tsunami real-time computer, a new multiprocessor control platform that greatly increases the performance capabilities of real-time controllers to meet the needs of the most demanding high-speed high-precision control applications. Utilizing a triple-body processing architecture and built around 4 high performance digital signal processors, the platform demonstrated 1MHz control sampling rates with less than 6ns RMS control cycle timing variation and 16-bit data acquisition accuracy. For the configuration designed and implemented in this thesis the platform includes the following input-output interfaces:

- 4x 4MSPS, 16-bit, ±10V analog inputs
- 16x 500kSPS, 16-bit, ±10V analog inputs
- 4x 50MSPS, 16-bit, ±10V analog outputs
- 32x 500kSPS, 16-bit, ±10V analog outputs
- 6x timestamped quadrature encoder inputs with 0.6ns timestamp resolution and 32-bit position counter
- 2x cameralink inputs
- 48x PWM outputs capable of a 200kHz modulation frequency
- 20x general purpose 5V TTL digital I/O

While these interfaces make Tsunami immediately applicable to most applications, the hardware architecture also enables these to be easily substituted for other custom interfaces. Further, all the created software for the control platform has been tightly integrated with industry standard development tools to facilitate both rapid controller development and professional graphical user interface development, resulting in a complete rapid prototyping control solution. This includes Mathworks Simulink and
Real-time Workshop integration for model-based controller development, automatic code generation, and automatic hardware implementation, as well as National Instruments LabVIEW integration for graphical user interface development.

The capabilities of the Tsunami control platform were demonstrated via a case study of a fast-tool servo precision machine tool. Utilizing Tsunami the full potential of the fast-tool servo control system was realized, achieving a closed-loop -3dB bandwidth of 25kHz when sampled at 1MHz. Further, thanks to Tsunami’s high-performance data acquisition a regulation error of only 1.35nm RMS was achieved over the 50µm axial-stroke of the fast-tool servo.

Second, this thesis investigated the effect of control cycle timing variations (sampling jitter and control jitter) on control performance. A new discrete disturbance model for sampling jitter and control jitter was developed, enabling a intuitive understanding of the effect of sampling jitter and control jitter on closed-loop system performance. Based on this model two scenarios were analyzed to establish the relationship between jitter and positioning error: (1) regulation error resulting from jitter’s interaction with measurement noise; (2) tracking error resulting from jitter’s interaction with a reference command. The result of each of these analyses was an equation that can be used to predict the additional positioning error introduced into a control system due to jitter. Further, with insights from these analyses, a new jitter compensator was proposed to greatly mitigate the positioning error contribution from jitter for the case of motion regulation.

Simulation and experiments were then carried out on a fast-tool servo to demonstrate the effects of sampling jitter and control jitter on positioning error. For the regulation scenario the results showed that sampling jitter has a negligible effect on regulation error and that control jitter can significantly degrade regulation performance, with 8% control jitter causing the RMS regulation error to increase by 90%. When the proposed jitter compensator was implemented the RMS regulation error increase for 8% control jitter was reduced from 90% to only 18%. For the tracking scenario the results showed that both sampling jitter and control jitter can significantly degrade tracking performance, with 8% sampling or control jitter causing the RMS tracking error to increase by 600%. In all cases the results and analytical predictions matched well, validating both the presented model and analyses.

6.1 Future Work

As an enabling technology, the Tsunami control platform opens up several possibilities for future studies. One such area is multirate multiprocessor controller decomposition strategies which aim to increase system performance by dividing the controller across multiple processors and executing each component at different rates. Research by Guo [46], Tomizuka [47], and Lu [1] has already shown the benefits of multirate multiprocessor control for some specific scenarios; however, it still remains a largely open area of research with lots of future potential, particularly with the recent mainstream availability of multicore processors. Tsunami is also ideally suited to facilitate research in new high-speed high-precision control applications due to its excellent performance and flexibility. Currently, it is already being used to support 4 active projects in the UBC Precision Mechatronics Lab and will hopefully be expanded to more in the future.
While Tsunami has demonstrated excellent real-time performance, further work is still required to refine and extend the functionality and usability of the supporting software libraries. For the Simulink controller development library this includes supporting a wider range of Simulink functions and enabling the controller model to be downloaded and executed directly from within MATLAB. For the LabVIEW GUI development library this includes adding more high level functions such as a digital signal analyzer, a template interface, and a G-code interpreter, as well as simplifying the use of the existing functions for signal monitoring, parameter tuning, and signal logging. Further case studies on other control systems are also recommended so Tsunami can continue to be refined and its performance benefits can continue to be demonstrated.

The control and sampling jitter investigation resulted in a new model to capture the effects of jitter on control performance and solid experimental results for the two scenarios analyzed. While these studies validated the model, they did not fully explore its significance, thus several potential threads for future studies still exist. First, the existing studies on positioning error can be extended to other jitter distributions (only white jitter was analyzed), and other reference commands (only a sinusoidal reference command was analyzed). Second, jitter’s interaction with other system inputs and characteristics can be explored; for example, during the experimental testing on the fast-tool servo it was observed that jitter amplified the errors caused by non-linearities in the system. Lastly, jitters effect can be evaluated for system performance metrics other than positioning error; for example, simulations by Marti et al. [34] observed an increase in overshoot and settling time for the step response of a system with jitter.
Bibliography


Appendix A

Jitter Measurement

Sampling jitter and control jitter can vary greatly for different controller implementations. It is mainly affected by factors such as task scheduling, input-output device synchronization, interrupt handling, cache misses, and resource sharing. Note that the absolute control jitter is not affected by the sampling rate selected (Only the normalized control jitter from Equation 4.18 is affected by the sampling rate). The simplest way to measure sampling jitter and control jitter is with an input timestamp and output timestamp, respectively; however, many controller implementations do not provide access to this data. Further, even if timestamps are available, their definition and exact implementation can vary from controller to controller, providing misleading results. To guarantee a fair comparison between different controller implementation an external timestamping setup is used here. This external setup only enables output timestamps to be captured, therefore only control jitter measurements are made. Generally, the sampling jitter will be slightly less than the control jitter as previously shown by Equation 4.5 and Equation 4.6.

The control jitter measurement setup introduces an additional function at the end of a conventional PID control cycle that switches a secondary DAC output between 0V and 2.5V. This output is then sent through a comparator and into an FPGA, which captures the signal edges with 5ns timestamp resolution and stores the control output update timestamps. To extract the control jitter from these timestamps $T_S[k]$, a least square linear fitting is performed to minimize the following sum

$$\sum_k (T_S[k] - Ak + B)^2$$  \hspace{1cm} (A.1)

where $A$ and $B$ are the least squares fitting coefficients and $k$ is the current control cycle. This ideal timestamp is then subtracted from the actual timestamp to obtain the control jitter, expressed as

$$\tau_c[k] = T_S[k] - Ak - B$$  \hspace{1cm} (A.2)

Control jitter measurements are made for the following three controller implementations: xPC Target on a P3 800MHz processor with a NI-6036E DAQ card; dSPACE DS1103; and Tsunami from Chapter 2. As shown by Figure A.1, the xPC Target implementation has 810ns RMS of control jitter,
the dSPACE 1103 implementation has 155 ns RMS of control jitter, and the custom UBC platform has 5.2 ns RMS of control jitter. While there are some deterministic components to the control jitter, most of the signal energy comes from the white random component, indicating that the assumptions made for the analyses of Section 4.2 should provide accurate error predictions. These results clearly demonstrate how greatly control jitter can vary depending on the control hardware. Note that even the xPC Target implementation is considered a relatively high-end control platform and other more general purpose platforms exhibit several microseconds of jitter [27] [28].
(a) xPC Target on a P3 800MHz processor with a NI-6036E DAQ card.

(b) dSPACE DS1103

(c) Tsunami real-time control platform

Figure A.1: Control jitter measurements for various controller implementations