4-1-2004

Real Time Digital Signal Processing Adaptive Filters for Correlated Noise Reduction in Ring Laser Gyro Inertial Systems

David A. Doheny
University of South Florida

Follow this and additional works at: https://scholarcommons.usf.edu/etd
Part of the American Studies Commons

Scholar Commons Citation
https://scholarcommons.usf.edu/etd/1015

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Real Time Digital Signal Processing

Adaptive Filters for Correlated Noise Reduction in Ring Laser Gyro Inertial Systems

by

David A. Doheny

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ravi Sankar, Ph.D.
Kenneth A. Buckle, Ph.D.
David Snider, Ph.D.

Date of Approval:
April 1, 2004

Keywords: Mean, Recursive, Least, Square, Estimator

© Copyright 2004, David A. Doheny
Dedication

To Terri, Seth and Aaron
My life’s inspiration…

Though the road is long and fraught with peril,
without a start, there is no finish….
Acknowledgments

It is imperative that government, industry and academia find opportunities for close, prolonged associations to insure scientific exchange of technical ideas and solutions. This work was supported in part by grants from Honeywell International Academic Research Program, The Florida Space Grant Consortium and The Florida Legislature Grant Program for the I-4 High Tech Corridor Initiative. With support and guidance of Dr. Sankar, Professor, Electrical Engineering, University of South Florida, the author presents this thesis, Real Time Digital Signal Processing Adaptive Filters for Correlated Noise Reduction in Ring Laser Gyro Inertial Systems.

No work is authored alone. From my first shoe tie to the last stroke of ivory in this writing the encouragement, to ponder, to conjecture, to learn, to formulate and to validate with support from a collage of individuals throughout my life has resulted in this work. There are infinite contributions though the years from family, friends, and colleagues. Thank-you all.

To Dr. Ravi Sankar, a heartfelt thank-you for your patience, scholastic enthusiasm and academic expertise. I am one of many that you have instilled the craving and excitement of science. You will forever be a part of us all! To my countless teachers, professors and mentors you have my sincere appreciation. To my colleagues who listened to my endless ramblings, I extend a hearty thank-you.
Table of Contents

List of Tables iii

List of Figures iv

Abstract viii

Chapter One – Introduction 1
  1.1 Definition 1
  1.2 Applications 1
  1.3 Basic Properties 1
  1.4 Adaptive Noise Cancellation Algorithms 2
  1.5 Description and Organization of Thesis 3

Chapter Two – Adaptive Signal Processing for Ring Laser Gyros 4
  2.1 Background 4
  2.2 Motivation for Real Time Adaptive Filters in RLG Based Systems 5
  2.3 Algorithm Development 5
    2.3.1 LMS Algorithm 6
    2.3.2 Normalizing LMS Algorithm 10
    2.3.3 LMS Full Matrix Algorithm 11
    2.3.4 Recursive Least Squares Algorithm 13
    2.3.5 Joint Process Gradient Estimator Algorithm 17
  2.4 Second System Analysis 20
  2.5 Convergence and Stability 20
  2.6 Computational Complexity 21
  2.7 Covariance Analysis of Algorithm Types 22
  2.8 Adaptive Algorithm Summary 27

Chapter Three – Considerations For Hardware Development 29
  3.1 Development 29
  3.2 A Query of Tool suites 29
    3.2.1 Mentor Graphics Monet 30
    3.2.2 Synopsys COSSAP 31
    3.2.3 Frontier Design Digital Signal Processing Station 32
Chapter Four – Algorithm Implementation
4.1  Direction 35
4.2  Algorithm Development Using DSP Station 35
4.3  Tool Independent Development 38
4.4  VHDL and Functional Simulation 38
4.5  ASIC Selection 39
4.6  Synthesis 40
4.7  Place and Route 41
4.8  Worst Case Timing 41
4.9  Post Route Simulation 41
4.10  Test Bench for Simulation and Validation 42
4.11  VHDL Architecture 44
4.12  Scaling 48
4.13  Results and Analysis 48

Chapter Five – Other Considerations
5.1  Finite Word Length Effects 50
  5.1.1  Overflow 51
  5.1.2  Quantization 51
5.2  Evaluation of Overflow and Quantization of LMS Algorithm 51
5.3  Scaling Optimization 54

Chapter Six – Conclusion and Future Research
6.1  Conclusion 56
6.2  Future Research 57

References 58

Bibliography 59

Appendices 60
  Appendix A: Algorithm Figures 61
  Appendix B: MathCAD Scripts 113
  Appendix C: Sample Listing of MATLAB Program 130
  Appendix D: VHDL Listings 154
  Appendix E: Field Programmable Gate Array Pin List and Timing 173
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Effects of $\mu$ on Convergence</td>
<td>20</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Trade-offs in Adaptive Filter Algorithms</td>
<td>22</td>
</tr>
<tr>
<td>Table 2.3</td>
<td>Statistical Measurements for System #1 and #2 Input Parameters</td>
<td>24</td>
</tr>
<tr>
<td>Table 2.4</td>
<td>Statistical Parameter Acronym Definitions I</td>
<td>25</td>
</tr>
<tr>
<td>Table 2.5</td>
<td>Statistical Measurements for System #1</td>
<td>26</td>
</tr>
<tr>
<td>Table 2.6</td>
<td>Statistical Measurements for System #2</td>
<td>26</td>
</tr>
<tr>
<td>Table 2.7</td>
<td>Algorithm Acronym Definitions</td>
<td>27</td>
</tr>
<tr>
<td>Table 2.8</td>
<td>Statistical Parameter Acronym Definitions II</td>
<td>27</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Worst Case Analyses Summary</td>
<td>41</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>MATLAB Versus VHDL Output</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1 Single Channel Least Mean Square Adaptive Filter 6
Figure 2.2 Full Matrix Least Mean Square Adaptive Filter 12
Figure 2.3 Single Channel Recursive Least Square Adaptive Filter 13
Figure 2.4 Full Matrix Recursive Least Square Adaptive Filter 16
Figure 2.5 Single Channel Joint Process Gradient Estimator 19
Figure 2.6 Full Matrix Joint Process Gradient Estimator 19
Figure 3.1 Monet Algorithm/Architecture Design Environment 31
Figure 3.2 COSSAP Digital Signal Processing (DSP) Design Environment 32
Figure 3.3 Frontier Design Digital Signal Processing Station Platform 33
Figure 4.1 Schematic Representation of Single Channel Real Time LMS Filter 35
Figure 4.2 Operational LMS Input/Output Trace History 36
Figure 4.3 Data Analysis of RLG and LMS Filter Output 37
Figure 4.4 Platform and Test Bench Environment 43
Figure 4.5 Top Level Data Flow/Schematic Diagram of VHDL LMS Algorithm 45
Figure 4.6 LMS State Machine Sequence Diagram 47
Figure 5.1 Finite Word Length Effects 50
Figure A1 PSD of Correlated Gyro Readout Noise 61
Figure A2 Cumulative PSD-Correlated Gyro Readout Noise 61
Figure A3 PSD of Gyro Dither Pick Off 62
Figure A4 Cumulative PSD of Gyro Dither Pick Off 62
Figure A5 LMS Performance Surface 63
Figure A6 Gyro X,Y, Z LMS Learning Curve 63
Figure A7 LMS G1 Gain Values 64
Figure A8 LMS G1 Gains 64
Figure A9 LMS G2 Gain Values 65
Figure A10 LMS G2 Gains 65
Figure A11 LMS G1 Gain Versus G2 Gain Curves 66
Figure A12 Performance Surface Using Actual Gains and MSE X Channel 66
Figure A13 LMS Gains-Vs-MSE 67
Figure A14 LMS G1 Gains Versus Uncorrelated Readout Noise 67
Figure A15 LMS PSD-Stripped Delta Theta Uncorrelated Readout Noise 68
Figure A16 LMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise 68
Figure A17 Full Matrix LMS G1 Gain Values 69
Figure A18 Full Matrix LMS G1 Gains 69
Figure A19 Full Matrix LMS G2 Gain Values 70
Figure A20 Full Matrix LMS G2 Gains 70
| Figure A21 | FMLMS PSD – Stripped Delta Theta Uncorrelated Readout Noise | 71 |
| Figure A22 | FMLMS CPSD – Stripped Delta Theta Uncorrelated Readout Noise | 71 |
| Figure A23 | Normalized LMS G1 Gain Values | 72 |
| Figure A24 | Normalized LMS G1 Gains | 72 |
| Figure A25 | Normalized LMS G2 Gain Values | 73 |
| Figure A26 | Normalized LMS G2 Gains | 73 |
| Figure A27 | NLMS PSD-Stripped Delta Theta Uncorrelated Readout Noise | 74 |
| Figure A28 | NLMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise | 74 |
| Figure A29 | RLS W0 Weight Values | 75 |
| Figure A30 | RLS W0 Weight Values | 75 |
| Figure A31 | RLS W1 Weight Values | 76 |
| Figure A32 | RLS W1 Weight Values | 76 |
| Figure A33 | RLS Main Channel Uncorrelated Delta Theta PSD | 77 |
| Figure A34 | RLS Main Channel Uncorrelated Delta Theta CPSD | 77 |
| Figure A35 | Full Matrix RLS W0 Gain Values | 78 |
| Figure A36 | Full Matrix RLS W0 Gains | 78 |
| Figure A37 | Full Matrix RLS W1 Gain Values | 79 |
| Figure A38 | Full Matrix RLS W1 Gains | 79 |
| Figure A39 | Full Matrix RLS PSD-Stripped Delta Theta Uncorrelated Readout Noise | 80 |
| Figure A40 | Full Matrix RLS CPSD-Stripped Delta Theta Uncorrelated Readout Noise | 80 |
| Figure A41 | JPGE Gains | 81 |
| Figure A42 | JPGE Gamma Values | 81 |
| Figure A43 | JPGE – PSD Uncorrelated Gyro Readout Noise E(2) | 82 |
| Figure A44 | JPGE – CPSD Uncorrelated Gyro Readout Noise E(2) | 82 |
| Figure A45 | JPGE – PSD Uncorrelated Gyro Readout Noise E(3) | 83 |
| Figure A46 | JPGE – CPSD Uncorrelated Gyro Readout Noise E(3) | 83 |
| Figure A47 | FMJPGE Gains | 84 |
| Figure A48 | FMJPGE Gamma Values | 84 |
| Figure A49 | FMJPGE – PSD Uncorrelated Gyro Readout Noise E(2) | 85 |
| Figure A50 | FMJPGE – CPSD Uncorrelated Gyro Readout Noise E(2) | 85 |
| Figure A51 | FMJPGE – PSD Uncorrelated Gyro Readout Noise E(3) | 86 |
| Figure A52 | FMJPGE – CPSD Uncorrelated Gyro Readout Noise E(3) | 86 |
| Figure A53 | LMS – Variance Correlated Readout Noise | 87 |
| Figure A54 | LMS – Variance Dither Pick Off | 87 |
| Figure A55 | LMS – Covariance Correlated Readout Noise to DPO | 88 |
| Figure A56 | LMS – Correlation Coefficient Correlated Readout Noise to DPO | 88 |
| Figure A57 | LMS – Variance Uncorrelated Readout Noise | 89 |
| Figure A58 | LMS – Covariance Uncorrelated Readout Noise to Dither Pick Off | 89 |
| Figure A59 | LMS – Correlation Coefficient Uncorrelated Readout Noise to DPO | 90 |
| Figure A60 | FMLMS – Variance Uncorrelated Readout Noise | 90 |
| Figure A61 | FMLMS – Covariance Uncorrelated Readout Noise to DPO | 91 |
| Figure A62 | FMLMS – Correlation Coefficient Uncorrelated Readout | 91 |
| Figure A63 | NLMS – Variance Uncorrelated Readout Noise | 92 |
| Figure A64 | NLMS – Covariance Uncorrelated Readout Noise to DPO | 92 |
| Figure A65 | NLMS – Correlation Coefficient Uncorrelated Readout Noise to DPO | 93 |
| Figure A66 | Main Channel RLS – Variance Uncorrelated Readout Noise to DPO | 93 |
| Figure A67 | Main Channel RLS – Covariance Uncorrelated Readout to DPO | 94 |
| Figure A68 | Main Channel RLS – Correlation Coefficient Uncorrelated Gyro Readout Noise to Pick Off | 94 |
| Figure A69 | Full Matrix RLS – Variance Uncorrelated Readout Noise | 95 |
| Figure A70 | Full Matrix RLS – Covariance Uncorrelated Readout Noise to DPO | 95 |
| Figure A71 | FMRLS-Correlation Coefficient Uncorrelated Gyro Readout Noise to DPO | 96 |
| Figure A72 | JPGE – Variance Uncorrelated Readout Noise | 96 |
| Figure A73 | JPGE – Covariance Uncorrelated Readout Noise to DPO | 97 |
| Figure A74 | JPGE-Correlation Coefficient Uncorrelated Readout Noise to DPO | 97 |
| Figure A75 | FMJPGE – Variance Uncorrelated Readout Noise | 98 |
| Figure A76 | FMJPGE – Covariance Uncorrelated Readout Noise to DPO | 98 |
| Figure A77 | FMJPGE – Correlation Coefficient Uncorrelated Readout Noise to DPO | 99 |
| Figure A78 | Isolated System LMS PSD – Delta Theta Correlated Readout Noise | 99 |
| Figure A79 | Isolated System LMS CPSD – Delta Theta Correlated Readout Noise | 100 |
| Figure A80 | Isolated System LMS PSD – Delta Dither Pick Off | 100 |
| Figure A81 | Isolated System LMS CPSD – Delta Dither Pick Off | 101 |
| Figure A82 | Isolated System LMS G1 Gain Values | 101 |
| Figure A83 | Isolated System LMS G2 Gains | 102 |
| Figure A84 | Isolated System LMS G2 Gain Values | 102 |
| Figure A85 | Isolated System LMS G2 Gains | 103 |
| Figure A86 | Isolated System LMS PSD – Stripped Delta Theta Uncorrelated Readout Noise | 103 |
| Figure A87 | Isolated System LMS CPSD – Stripped Delta Theta Uncorrelated Readout Noise | 104 |
| Figure A88 | Isolated System – Variance Correlated Readout Noise | 104 |
| Figure A89 | Isolated System – Variance Dither Pick Off | 105 |
| Figure A90 | Isolated System LMS – Covariance Correlated Readout Noise to DPO | 105 |
| Figure A91 | Isolated System JPGE – Correlation Coefficient Correlated Readout Noise to DPO | 106 |
| Figure A92 | Isolated System LMS – Variance Uncorrelated Readout Noise | 106 |
| Figure A93 | Isolated System LMS – Covariance Uncorrelated Readout Noise to DPO | 107 |
| Figure A94 | Isolated System LMS – Correlation Coefficient Uncorrelated Readout Noise | 107 |
| Figure A95 | Isolated System FMLMS G1 Gain Values | 108 |
| Figure A96 | Isolated System FMLMS G1 Gain Values | 108 |
| Figure A97 | Isolated System FMLMS G2 Gain Values | 109 |
| Figure A98 | Isolated System FMLMS G2 Gain Values | 109 |
| Figure A99 | Isolated System FMLMS PSD – Stripped Delta Theta Uncorrelated Readout Noise | 110 |
Readout Noise

Figure A100 Isolated System FMLMS CPSD – Stripped Delta Theta Uncorrelated Readout Noise

Figure A101 Isolated System FMLMS – Variance Uncorrelated Readout Noise

Figure A102 Isolated System FMLMS – Covariance Uncorrelated Readout Noise to Dither Pick Off

Figure A103 Isolated System FMLMS – Correlation Coefficient Uncorrelated Readout Noise To DPO

Figure B1 PSD Raw Xgyro

Figure B2 CPSD Raw Xgyro

Figure B3 PSD Raw Xdpo

Figure B4 CPSD Raw Xdpo

Figure B5 VHDL LMS Output Xgyro

Figure B6 VHDL Adaptive Gain h0

Figure B7 VHDL Adaptive Gain h1

Figure B8 VHDL PSD Filtered Xgyro

Figure B9 VHDL CPSD Filtered Xgyro

Figure B10 MATLAB LMS Output Xgyro

Figure B11 MATLAB Adaptive Gain h0

Figure B12 MATLAB Adaptive Gain h1

Figure B13 MATLAB PSD Filtered Xgyro

Figure B14 MATLAB CPSD Filtered Xgyro

Figure B15 Quadratic Error Surface
Real Time Digital Signal Processing
Adaptive Filters For Correlated Noise Reduction in Ring Laser Gyro
Inertial Systems

David A. Doheny

ABSTRACT

Existing opportunities in advanced interceptor, satellite guidance and aircraft
navigation technologies, requiring higher signal processing speeds and lower noise
environments, are demanding Ring Laser Gyro (RLG) based Inertial Systems to reduce
initialization and operational data latency as well as correlated noise magnitudes.
Existing signal processing algorithms are often less than optimal when considering these
requirements. Advancements in micro-electronic processes have made Application
Specific Integrated Circuits (ASIC) a fundamental building block for system
implementation when considering higher-level signal processing algorithms.

Research of real time adaptive signal processing algorithms embedded in ASICs
for use in RLG based inertial systems will help to understand the trade-off in finite
register length effects to correlated noise magnitude, organizational complexity,
computational efficiency, rate of convergence, and numerical stability. Adaptive filter
structures selected will directly affect meeting inertial system performance requirements
for data latency, residual noise budgets and real time processing throughput. Research in
this area will help to target specific adaptive noise cancellation algorithms for RLG based inertial systems in a variety of military and commercial space applications.

Of particular significance is an attempt to identify an algorithm embedded in an ASIC that will reduce the correlated noise components to the theoretical limit of the RLG sensor itself. This would support a variety of applications for the low noise space environments that the RLG based inertial systems are beginning to find promise for such as advanced military interceptor technology and commercial space satellite navigation, guidance and control systems.
Chapter One

Introduction

1.1 Definition

Adapt. v.t., To make suitable to or fit for a specific use or situation. *(Webster's Revised Unabridged Dictionary, © 1996, 1998 MICRA, Inc.)*

The root word for adaptive is adapt. Early definitions for the word adapt, listed in a variety of dictionaries, almost exclusively target environmental (physiological, psychological, and sociological) adaptation. As man continues his progress in developing mechanical and electrical technologies, the definition becomes more generic, covering not only the biological but also the tools of his development and use.

1.2 Applications

With the advancement of microelectronics, allowing for the development of extremely small application specific integrated circuits operating at increasingly higher speeds, adaptive signal processing algorithms have become more prevalent in today’s technology. Adaptive Noise Cancellation (ANC) algorithms have been used extensively in audio, medical and communications. Examples are echo cancellers (filters) in telephone equipment, system identification in communications networks, heart component suppression in bio-medical electrocardiograph monitors. Adaptive quantization has found promise in areas of speech signal processing allowing for adaptive dynamic range and quantization levels. Adaptive control algorithms are used widely in digital control systems such as Automatic Gain Control (AGC). Adaptive beam formers are used extensively in radar, sonar and seismology. As we continue to understand the innate environmental adaptation that surrounds us, so too will we adapt our understanding to processes that enhance our everyday lives.

1.3 Basic Properties

The “adapt” definition listed above, does not alone dictate the properties of an adaptive process. An adaptive process can be open or closed. An adaptive process can be self-controlling. In some instances, the process can exhibit artificial intelligence by learning. These processes can be recoverable. They incorporate both linear and non-
linear components. They can be time variant and time invariant. In general, adaptive processes allow for a wide variety of attributes to be part of their characteristics.

Fixed design processes assume that the input to output relationships are well defined. The non-adaptive design approach assumes bounded attributes (e.g. gain and phase margins) insuring a transfer function of the input to a known output based on characteristics ideally selected for the application. germane to the adaptive processes is the ability to self adjust. Adaptive processes allow for a continuous adjustment of the transfer function properties based on some error criteria. As such, adaptive processes differ from fixed design processes in that they are inherently non-linear unless held fixed once the error criteria has been met. As well, adaptive processes operate to optimize, through adjustment, their outputs by controlling to some predefined value.

Systems, linear and non-linear, fixed or adaptive, are classified as open loop or closed loop. If the system takes a component of the output and feeds it back into the system, the system is considered closed loop. Adaptive systems can operate in either open or closed loop mode. There is a subtle difference to the definition of the open loop system for an adaptive process. All adaptive systems require knowledge of specific performance parameters to adapt or adjust. An open loop adaptive system may be a system where the output data is collected and analyzed in an offline process of the adaptive process itself. The results of the analysis are then feedback manually to the selected adjustment criteria so that adaptation to performance requirements will occur. In a closed loop adaptive system, the analysis would be a component of the process and the adjustment to the performance parameters would occur automatically in order to optimize it’s output.

1.4 Adaptive Noise Cancellation Algorithms

While adaptive processes can be applied to a plethora of mechanical and electrical applications. The focus of this writing is cancellation of correlated noise. As such adaptive cancellation algorithms have been researched. Such algorithms include the Least Mean Square (LMS), Recursive Least Square (RLS) and Joint Process Gradient Estimator lattice (JPGE) Algorithms. In each of these, the adaptive processes are closed loop with the adjustments made to minimize the mean squared error criteria.

All of these algorithms can be traced to early work by such mathematicians as Gram, Schmidt, Wiener, Woodrow, and Hopf. Grahm and Schmidt’s development of the normal equations, Wiener’s and Hopf’s adaptation of the Grahm-Schmidt orthogonalization to estimation theory and Woodrow-Hopf’s work toward the development of the LMS algorithm all set a foundation for development of the Normalized LMS, RLS and JPGE.
1.5 Description and Organization of Thesis

The thesis main objective was to research, develop and code real time Adaptive Digital Signal Processing, ADSP, Filters for use in Ring Laser Gyro, RLG, based Inertial Systems to reduce correlated noise components. Trade-off in the filters structures, computation complexity, convergence, finite register length effects and effectiveness has been evaluated. Comparison of a Least Mean Square MATLAB model and an Application Specific Integrated Circuit, ASIC, implementation have been evaluated.

Each of the real time adaptive digital signal processing algorithms was developed for a diagonal and full matrix implementation. The diagonal algorithm implementations focus on reduction of correlated noise in each of the RLGs X, Y and Z channel. These are common art formulations of the classical filter structures. The full matrix implementation extends the common art to eliminate cross-coupled noise from channel to channel. There are also two RLG based systems that the algorithms were executed. One of the systems does not incorporate Mechanical Vibration Isolation, MVI, between the Inertial Sensor Assembly on which the RLGs are mounted and the systems chassis. The other system does incorporate MVIs.

Chapter One provides a definition of adaptation, reviews applications and basic properties for adaptive filters and outlines the thesis. Chapter Two reviews the basic operation of the RLG based Inertial Systems, the motivation to use and development diagonal and full matrix ADSP algorithms for correlated noise reduction and trade off in performance parameters for the algorithm types. Chapter Three presents considerations when targeting an ADSP for hardware implementation to include a query of tool suites available in today’s technology. Chapter Four outlines the steps taken for algorithm development in targeting an ASIC. Chapter five looks at other considerations, such as dynamic range, overflow and quantization after having successfully implemented the LMS algorithm in an ASIC. Appendix A illustrates the numerous plots of data referenced throughout the writing. Appendix B lists two MathCAD scripts that reflect evaluation of the LMS filter. Appendix C is a sample MATLAB script of the LMS filter. Appendix D is a sample of the top-level test bench and LMS filter in VHDL that was used to target the ASIC. Appendix E contains excerpts from the tool output of the ASIC pin list and static timing analysis.
Chapter Two

Adaptive Signal Processing for Ring Laser Gyros

2.1 Background

There are a variety of adaptive signal processing algorithms that have been developed and proven to work for a broad range of applications. In the past, technology has also produced a wide variety of inertial sensing instruments. By far, the most commonly used inertial sensor today is the strap down Ring Laser Gyro (RLG). In order to sense rotational motion in three-dimensional space, a typical inertial system will have three RLGs embedded in it, one for sensing each axis. To extract any usable information from the RLG a variety of analog circuitry, digital circuitry and signal processing support is required.

One physical drawback of the RLG is, at low input rates, the inertial sensor experiences a phenomena known as “lock-in”. Lock-in is the inability of the sensor support electronics to disseminate any rotational rate data from the information bearing output signal of the RLG. To alleviate this physical constraint, the RLG, mounted on a rigid block, is electro-mechanically dithered using a sinusoidal drive signal force and a component of Pseudo Random Noise (PRN), thereby insuring a nearly continuous component of rate into the RLG at all times.

The support electronics of the RLG converts incremental angle (rate) information into digital words. These digital words not only have the induced incremental angle due to the measured base motion, but also contain the incremental angle due to the sinusoidal dither drive as well. The PRN component of the induced noise is filtered in the electromechanical dither oscillators’ control loop. The signal processing of the RLG output words requires the “filtering” or “stripping” of the unwanted sinusoid while passing the true rotational rate information.

Filtering of the information can be accomplished but with the penalty of data latency due to iterative algorithm effects. A more elaborate approach of dither removal, referred to as “dither stripping”, lends itself to real time signal processing. In past dither stripping algorithms, the estimate of the incremental angle due to the sinusoidal dither component is determined from a digitized reference sample, referred to as the dither pick off (DPO) of the sinusoidal motion. This digitized information is then subtracted from the incremental RLG data thereby producing true rate motion with residual correlated
noise components. The algorithm model is a batch process that imposes a moderate amount of computational complexity for a real time system.

2.2 Motivation for Real Time Adaptive Filters in RLG Based Systems

To date, a variety of real time adaptive filters such as the LMS, RLS and JPGE have been researched and developed with some success in identifying candidate structure for low noise and low data latency requirements. Specifically, a three channel, two gain (sometimes referred to as weights or coefficients) per channel LMS, RLS and JPGE structures were implemented as a main channel correlated noise canceller. To illustrate trade off in performance, the basic LMS algorithm was then altered to a Normalized LMS adaptive filter. Cross channel coupling of the individual RLG mechanical dithered motion from RLG axis to RLG axis motivated research into a full matrix algorithm structure. The basic LMS, RLS and JPGE structures were expanded from six to eighteen gains for all three channels. These algorithms not only strip the main channel correlated noise from their respective RLGs, but also strip any cross-coupled correlated noise component terms.

Real time data from two candidate RLG based systems was recorded and used for evaluation of the LMS, RLS and JPGE algorithms. The first system showed very little cross coupling from channel to channel of the mechanical dither. Power Spectral Density (PSD) and Cumulative PSDs (CPSD) were generated as a means for an evaluation of the LMS filters’ effectiveness and convergence properties for these algorithms.

Figure A1 in appendix A, shows the superimposed PSD of the correlated information bearing signals, $\Delta \Theta x$, $\Delta \Theta y$ and $\Delta \Theta z$ for the three RLG channels. Inclusive in each channels’ data are the three components of signal information; the base motion of rotation, the sinusoidal reference signal and the three components of signal information. Figure A2, shows the CPSD of the same information. Note that the sinusoidal dither reference signals at approximately 525 Hz, 575 Hz and 600 Hz have the greatest magnitude of energy associated with them. Figure A3 and A4 are the superimposed PSD and CPSD of the sinusoidal reference signals, X, Y and Z, of the three RLG channels. The sinusoidal energy peaks were validated to be true representations of the digitized “control to” magnitudes of the closed loop electromechanical dither amplitudes for each of the three channels.

2.3 Algorithm Development

There are a variety of adaptive filter algorithms discussed throughout the literature. Development of any adaptive filter algorithm should evaluate the real time implications when considering targeting the algorithm for hardware or software. Trade studies into each of the algorithms organizational and computational complexity were performed. Evaluation of the LMS, RLS and JPGE structures has helped to identify
optimal implementations for use in a variety of mechanically dithered Ring Laser Gyro Inertial Systems.

2.3.1 LMS Algorithm

One of the most well known, and often implemented, adaptive filter algorithms is the Least Mean Square algorithms, or LMS. Its popularity is due, in large part, to its simplicity and ease of computation. The algorithm is based on the gradient decent approach to correlated noise cancellation. As a gradient decent algorithm, the intent of the algorithm is to extract or de-correlate a reference signal from an input signal containing correlated reference signal components.

Fundamental to the LMS algorithm operation, is the requirement that the input signal and the correlated noise source be available. A simplified block diagram of the LMS filter can be seen in Figure 2.1. The change of angle input signal, \( \Delta \Theta_{\text{input}} \), includes components consisting of the desired rotation signal, \( \Delta \Theta_{\text{base}} \), the correlated noise dither \( \Delta \Theta_{\text{ref}} \), and the uncorrelated noise signal, \( S_{\text{noise}} \). The operation of the LMS algorithm is to subtract or “strip” the correlated noise reference from the input signal, leaving the desired and uncorrelated noise residual signals.

![Figure 2.1 – Single Channel Least Mean Square Adaptive Filter](image)

\[
\Delta \Theta_{\text{input}} = \Delta \Theta_{\text{base}} + \Delta \Theta_{\text{ref}} + S_{\text{noise}}
\]

\[
\Delta \Theta_{\text{Output}} = \Delta \Theta_{\text{base}} + S_{\text{noise}}
\]

The LMS algorithms’ purpose is to cancel the component of the reference and any other non-orthogonal signal components that may exist between the reference and the input. Assuming the base motion of the input in Figure 2.1 is static or zero, \( \Delta \Theta_{\text{base}} = 0 \), the output of the LMS can be viewed as the uncorrelated error signal, \( S_{\text{noise}} \). The LMS algorithm finds its roots in the Adaptive Linear Combiner, ALC. The ALC algorithm takes the input and the successive samples of the delayed versions of the input, multiplies each by a weight (liberally referred to as a gain or coefficient), linearly combines them and then subtracts the sum from the desired response. The LMS algorithm advances one step further in the process by using its output to adjust the weights adaptively in order to minimize an error criterion.
The LMS algorithm’s error criterion for determining the effectiveness of the process is the measurement of the Mean Squared Error, MSE, of the output. This can be derived from the basic equations governing the algorithm itself. Equations (2.1) through (2.6) reflect the derivation for the MSE. From this, the MSE is derived to be a quadratic form. Continuing the development in equations (2.7) and (2.8), we arrive at the LMS algorithm weight update equations.

From the adaptive linear combiner, the LMS algorithm can be derived. The output of the LMS filter can be calculated as:

$$\Delta \Theta_{\text{output}} = \Delta \Theta_{\text{input}} - \Delta \Theta_{\text{estimate}}$$  \hspace{1cm} (2.1)

The estimate can be calculated for $n^{th}$ order, so the formulation requires the output to be calculated using an expansion of the estimate as:

$$\Delta \Theta_{\text{output}} = \Delta \Theta_{\text{input}} - \Delta \Theta_{\text{estimate}} = \Delta \Theta_{\text{input}} - (W^T \Delta \Theta_{\text{ref}})$$  \hspace{1cm} (2.2)

Where $W^T = [w_1, w_2, w_3, \ldots w_k]^T$ is a $k$ by 1 vector of weights and $\Delta \Theta_{\text{ref}} = [\Delta \Theta_{\text{ref}}(n), \Delta \Theta_{\text{ref}}(n-1), \Delta \Theta_{\text{ref}}(n-2), \ldots]$ is a 1 x $k$ vector of present and past values of the reference.

Squaring both sides of (2.2) and expanding, we get a quadratic equation of the form:

$$(\Delta \Theta_{\text{output}})^2 = (\Delta \Theta_{\text{input}})^2 + (W^T \Delta \Theta_{\text{ref}} \Delta \Theta_{\text{ref}}^T W) - (2 \Delta \Theta_{\text{input}} \Delta \Theta_{\text{ref}}^T W)$$  \hspace{1cm} (2.3)

If we define the auto correlation, $R$, of the reference as:

$$R = E[\Delta \Theta_{\text{ref}} \Delta \Theta_{\text{ref}}^T]$$  \hspace{1cm} (2.4)

And the cross correlation of the input and the reference as:

$$P = E[\Delta \Theta_{\text{input}} \Delta \Theta_{\text{ref}}]$$  \hspace{1cm} (2.5)

We can express the output in terms of the mean square error as:

$$\text{MSE} \: \Delta \hat{\xi} = E[\Delta \Theta_{\text{input}}] + W^T R W - 2P^T W$$  \hspace{1cm} (2.6)

The LMS algorithm operates on the quadratic error performance surface defined in (2.6) above is a gradient decent estimation process. There are two classical processes in which the algorithm chases the minimum of the quadratic performance surface: the Newton and the steepest decent process. The Newton process varies from the steepest gradient decent process in that its weight vector updates are always towards the error minimum and attempts to estimate the minimum in a single step. The steepest gradient decent process weight vector updates are always in the direction of the negative gradient.
of the error surface and is inherently a multi-step process. Throughout this writing, the gradient decent algorithm is discussed.

In order for the steepest gradient decent algorithm to converge to a minimum mean square error, the weight vector must continuously be updated. At the minimum mean square error, the weight vector is considered to be at its optimal value. In absence of any noise, the gradient from estimate to estimate at the minimum error would be ideally zero. The gradient, $\nabla (\xi)$, of the mean square error can be calculated by taking the partial derivative of the MSE and setting the column vector to zero.

$$\nabla (\xi) = [\partial \xi/\partial W_1 \quad \partial \xi/\partial W_2 \quad \partial \xi/\partial W_3 \ldots]^T = 2RW - 2P = 0$$ (2.7)

where $R$ and $P$ are defined by equations (2.4) and (2.5) respectively. This matrix equation illustrates that the time varying weight vector for the Weiner-Hopf solution, which when reduced, is given by:

$$W_{opt} = R^{-1}P$$ (2.8)

Which states that the optimum weight vector, resulting in the lowest possible error, is equal to the cross correlation of the input, $\Delta \Theta_{input}$, and the reference input, $\Delta \Theta_{ref}$, divided by the autocorrelation of the reference input, $\Delta \Theta_{ref}$.

In order to insure continued convergence of the mean square error to an operational minimum, the weight vector must be forever updated and the error calculated. The reduction in error is insured by recalculating the weight vector at subsequent iterations with a negative gradient or $\xi(W + \Delta W) \leq \xi(W)$.

In equation (2.6) above, the performance index was defined as a quadratic. When taking the partial with respect to the weight vector we get:

$$\partial \xi/\partial W = -2 (RW-P)$$ (2.9)

If the weights are allowed to become time dependent, the weight updates can be defined as a function of the negative gradient as:

$$W_{n+1} = W_n + \mu(-\nabla(\xi))$$ (2.10)

Where $\mu$ is a constant that controls the step size, and therefore the rate of the gradient search, resulting in an achievable optimum. Remembering that the gradient estimate, $\nabla(\xi)$ is equal to the squared error estimate, $\nabla(\epsilon^2)$ where $\epsilon = \Delta \Theta_{input} - (W^T \Delta \Theta_{ref})$, we get the estimate of the gradient as:

$$\nabla(\xi) = \nabla(\epsilon^2) = [\partial \epsilon_n^2/\partial W_1 \quad \partial \epsilon_n^2/\partial W_2 \quad \partial \epsilon_n^2/\partial W_3 \ldots \partial \epsilon_n^2/\partial W_k] = 2 \epsilon_n [\partial \epsilon_n/\partial W_1 \quad \partial \epsilon_n/\partial W_2 \quad \partial \epsilon_n/\partial W_3 \ldots \partial \epsilon_n/\partial W_k] = -2 \epsilon_n \Theta_{ref_n}$$ (2.11)
From this, we can write the steepest decent weight update equation as:

$$W_{n+1} = W_n + \mu(-\nabla(\xi)) = W_n + 2\mu\varepsilon_n \Theta_{ref_n}$$

(2.12)

Where, $n$ is an integer index of time, $W_{n+1}$ is the updated weight (gain or coefficient) vector, $W_n$ is the current weight vector, $\mu$ is the convergence factor, $\varepsilon_n$ is the error signal, and $\Theta_{ref_n}$ is the reference input.

Further derivation of the weight update equations reflect that the eigenvalues of the auto correlation matrix, $R$, represent a set of equations that describes the transient behavior of the iterative process from the initial value of the weights to the optimal solution. Using actual recorded data, plotting the MSE against the weight values produces the quadratic performance surface as seen in Figure A5. The convergence of the algorithm follows the performance surface to the MSE minimum where we find the optimal weight values. The algorithm continues to search the minimum with a continuous variation in the weights and MSE value.

The LMS algorithm was developed in MATLAB for the purpose of stripping the sinusoidal electro-mechanical dither reference input frequency components from their respective RLG channels. A matrix implementation of the relative equations listed in (2.1) through (2.12) is generated in MATLAB. The matrix version incorporates all three channels of dither stripping in a concise mathematical format. Another MATLAB script was developed to generate the quadratic performance surface. Reference Figure A5. In the process of generating the quadratic performance surface, the auto correlation matrix was derived and the eigenvalues were obtained. The stability criteria derived in the derivations (and validated in the literature) shows the LMS to be stable for the condition that the geometric ratio, $r$ is:

$$|r| = |1-2\mu\lambda| < 1$$

(2.13)

is valid. Where $\mu$, $\mu$, is the convergence factor and $\lambda$, is the eigenvalue of the autocorrelation matrix $R$. Reformulating this equation for stability when considering the rate of convergence gives

$$1/\lambda_{max} > \mu > 0$$

(2.14)

From the autocorrelation matrix, the maximum eigenvalue was obtained for each of the three channels. For ease of implementation, a single convergence factor, $\mu$, was used in the algorithm.

In Figure A1 we see the PSD and of the raw gyro data prior to any filtering. The magnitudes of the fundamental dither frequencies at 525 Hz, 575 Hz and 600 Hz are
clearly visible. Figure A2 illustrates the CPSD of this PSD plot. Figure A3 and A4 shows the three channels of dither reference PSD and CPSD respectively. Figure A5 show the quadratic performance surface using the channel RLG X data.

Figure A6 shows the learning curves (MSE versus sample) for the three X, Y, and Z RLG channels respectively. As the algorithm converges towards the minimum weights, the filters output reflects a minimum MSE or uncorrelated readout noise. Figure A7 and A9 shows the adaptive weights (gains) for all three channels reflecting their convergence to the optimal values. Figure A8 and A10 show the same gain values separated for ease of viewing. The cross-coupled terms, to be discussed later, are shown to be zero.

Figure A11 shows the convergence of the g1 versus g2 gain values as they approach their optimum values. Figure A12 show a cross cut of the actual performance surface for the RLG X channel, reflecting the interaction of the g1, g2 and mean square error. Figures A13 illustrates the g1 gains versus the mean square error (or uncorrelated readout error) for each of the channels in two-dimensional and three-dimensional respectively.

Figure A15 and A16 respectively shows a plot of the PSD and CPSD of the three RLG channels at the output of the LMS algorithm. These plots reflect the uncorrelated gyro readout noise magnitude (mean square error) under static conditions. Notice the relative magnitude of the data to that of the correlated RLG data plot in Figure A1 and A2. By interpretation of these figures we can see the LMS filter as having “stripped” the correlated readout noise leaving uncorrelated readout noise and residual. Noticeable jumps in the cumulative PSD data at other frequencies reflect folded frequencies from the fundamental and harmonic components in each of the ring laser gyro main channels as well as effects of cross channel coupling.

2.3.2 Normalizing LMS Algorithm

Corrections applied to the gains (often referred to as weights or coefficients) are directly proportional to the digitized dither (reference) input. When the dither variations are large, the basic LMS structure can experience problems with gradient noise amplification. In order to minimize the effects of less than optimal convergence factors and diminish the gradient noise amplification, a Normalized LMS algorithm was developed. This algorithm does not have knowledge of the input correlation matrix. It is not necessary then to estimate a convergence factor. Because of this, the rate of convergence (and the gradient estimate) depends on the norm of the input data. The new weight update equation is shown in equation (2.15). A constant between the value of 0 and 2 is selected as a normalized step size, $\beta$. A small value, $\delta$, is added to the norm in the ratio should the initial input data be very small or zero, thereby insuring convergence.
\[ W_{n+1} = W_n + \mu (-\nabla (\xi)) = W_n + \left[ \beta / (\| \Theta_{ref_n} \|^2 + \delta) \right] \varepsilon_n \Theta_{ref_n} \]  

Again, convergence of the order two gains can be seen clearly in figures A23 through A26 for the Normalized LMS. The full matrix g1 gains can be seen on a single graph in figure A23, with an exploded view illustrated in figure A24. Like wise, the full matrix g2 gains can be seen on a single graph in figure A25, with an exploded view illustrated in figure A26. While the problem with noise amplification is diminished, the possibility of the input dither reference signal being sampled consisting near zero can lead to instability in the algorithm. To insure against instability, a small positive value is added to the normalizing factor. As can be seen in the PSD and CPSD figures A27 and A28 respectively, the effectiveness of correlated noise removal parallels that of the standard LMS adaptive filter. Convergence of the NLMS algorithm is slightly longer than that of the LMS.

2.3.3 LMS Full Matrix Algorithm

Early in the development of the LMS algorithm, a particular RLG systems’ data reflected the contents of cross-coupled mechanical dither energy into each of the three main RLG channels. Further development of the LMS algorithm added cross channel stripping terms to remove these components of noise. This is referred to as the Full Matrix, FM, implementation of the LMS Algorithm. A top-level block diagram of this algorithm can be seen in Figure 2.2. As can be seen in the figure, each channel receives its own reference as well as the other two channels reference signals at the LMS input.
The full matrix LMS adaptive filter has an order two update matrix implementation with nine gain values for magnitudes and nine gains for phase. This gives each channel a set of six gains for a total of 18 gains. The off diagonal elements are responsible for the elimination of the correlated cross channel dither noise. Figure A17 and A18 shows full matrix g1 gains in matrix form. As well, Figure A19 and A20 show the full matrix g2 gains in matrix form. Figure A21 and A22 show the PSD and CPSD respectively of the full matrix performance relative to spectral energy in the nyquist range.

The Normalized LMS algorithm was also updated to a full matrix algorithm. Figure A23 through Figure A26 are plot of the gains. Figure A27 and A28 are the gyro readout noise PDS and CPSD after noise cancellation.
As with the basic LMS, a single convergence factor was used to reduce organizational and operational complexity of the filter. Initially the full matrix implementation was performed on a system with data that did not show mechanical coupling. This was done as a reference system to insure converges of the gains and decorrelation of the data. It was hopeful that the relative magnitude of uncorrelated noise would be reduced.

Once implemented, the full matrix LMS algorithm and the normalized LMS algorithms were easily adapted for a main channel (no cross terms) noise canceller. This was accomplished by multiplying the gain (weight) update equation by a 3 x 3 identity matrix with the off diagonal terms zeroed: leaving the diagonal of the matrix responsible for the gain updates for the main channel noise cancellation.

2.3.4 Recursive Least Squares Algorithm

Previous discussions relating to the LMS algorithm have been based on gradient decent convergence to the minimum mean square error. These algorithms require knowledge of the auto correlation of the input and the inputs cross correlation to the output. The mean square error is calculated in the real time adaptive LMS using an estimate of these statistics from the input data. For many applications where convergence time is not critical this approach is sufficient. However, for those applications that require fast convergence, the trade off for rapid convergence becomes excessive mean square error. An alternative algorithm would then require that the criteria for the error measurement not be a function of the statistics of the data, but on the data itself. One such algorithm is the Recursive Least Squares, RLS. A functional block diagram of the RLS can be seen in figure 2.3.

Figure 2.3 – Single Channel Recursive Least Square Adaptive Filter

\[
\Delta \Theta_{input} = \Delta \Theta_{base} + \Delta \Theta_{ref} + S_{noise} \quad \Rightarrow \quad \Delta \Theta_{output} = \Delta \Theta_{base} + S_{noise}
\]
Derivation of the RLS follows from the minimization of the error at time \( n \) based on weighted least squares. The performance index is then calculated as:

\[
\xi(n) = \varepsilon(n) = \sum_{k=0}^{n} \lambda^{k-j} |e(i)|^2, \quad k = 0 \text{ to } n \tag{2.16}
\]

For this formulation, lambda, \( \lambda \), is a forgetting factor between the value of 0 and 1 and \( e(i) \) is the error for the \( i \)th step. The exponential weight of lambda limits the amount of data in the past used for the error estimation and helps the algorithm to better track nonstationarities in the signal. Like the LMS, \( e(i) \) is defined as

\[ \Delta \Theta_{\text{output}} = \Delta \Theta_{\text{input}} - \Delta \Theta_{\text{estimate}} \text{ or } e_i = d_i - y_i \]

Like the LMS the derivation proceeds to find the weights that minimize \( \varepsilon(n) \) by taking the partial derivatives with respect to the weight vector and then setting it to zero. From these equations optimal weight vector can be formulated as

\[ W_n = r_n / R_n \tag{2.17} \]

Where \( r_n \) and \( R_n \) are the deterministic cross and auto correlation matrices and are given by:

\[ r(n) = \sum \lambda^{n-i} \Delta \Theta_{\text{input}}(i) \Delta \Theta_{\text{ref}}(i) \quad \text{for } i = 0 \text{ to } n \tag{2.18} \]

and

\[ R(n) = \sum \lambda^{n-i} \Delta \Theta_{\text{ref}} \Delta \Theta_{\text{ref}}(i) \quad \text{for } i = 0 \text{ to } n \tag{2.19} \]

Equation (2.16) is referred to as the deterministic normal equations. Instead of trying to solve the normal equations directly for each value at time instant, \( n \), the algorithm allows for calculating the weights, the auto correlation matrix and the cross correlation matrix recursively by expressing the current values at the index \( n \) in terms of the past values, \( (n-1) \). In order to calculate these values recursively and efficiently the matrix inversion lemma (also known as the Woodbury’s Identity) is applied to the auto-correlation matrix. Taking the inverse of the autocorrelation matrix as \( P(n) \), defining a recursive gain vector \( g(n) \) and information vector \( z(n) \), the RLS can be formulated. The equations for this algorithm are listed in (2.20) through (2.24).

\[ z(n) = P(n-1) \Delta \Theta_{\text{ref}}(n) \tag{2.20} \]
\[ g(n) = z(n) / [\lambda + \Delta \Theta_{\text{ref}}(n) \Delta \Theta_{\text{ref}}(n)^T] \tag{2.21} \]
\[ \alpha(n) = \Delta \Theta_{\text{input}} - W_{n-1} \Delta \Theta_{\text{ref}}(n) \tag{2.22} \]
\[ W_n = W_{n-1} + \alpha(n) g(n) \tag{2.23} \]
\[ P(n) = 1 / \lambda [P(n-1) - g(n) z(n)^T] \tag{2.24} \]
Where $W(n)$ is the coefficient (weight) matrix. It should be noted that equation (2.22) describes the filtering operations of the filter. In essence this operation illustrates an excitement of a transversal filter in order to compute the \textit{a priori} estimation error, $\alpha(n)$. Equation (2.23) describes the tap weight vector update by incrementing the previous value by an amount equivalent to the \textit{a priori} estimation error, multiplied by the time varying gain vector, $g(n)$. Equations (2.20) and (2.21) allow us to update the gain vector itself leaving equation (2.24) to calculate the inverse correlation matrix.

As with the LMS algorithm, two formulations of the RLS were developed using MATLAB. First, a matrix formulation for main channel cancellation was developed. Once validated, a Full Matrix RLS version was created. Figure 2.4 illustrates a Full Matrix RLS adaptive filter.
Figure 2.4 – Full Matrix Recursive Least Square Adaptive Filter

\[
\Delta \Theta^{\text{input}} = \Delta \Theta^{\text{base}} + \Delta \Theta^{\text{ref}} + \Delta \Theta^{\text{yref}} + \Delta \Theta^{\text{zref}} + S^{\text{noise}}
\]

\[
\Delta \Theta^{\text{output}} = \Delta \Theta^{\text{base}} + S^{\text{noise}}
\]

\[
\Delta \Theta^{\text{input}} = \Delta \Theta^{\text{base}} + \Delta \Theta^{\text{ref}} + \Delta \Theta^{\text{yref}} + \Delta \Theta^{\text{zref}} + S^{\text{noise}}
\]

\[
\Delta \Theta^{\text{output}} = \Delta \Theta^{\text{ybase}} + S^{\text{noise}}
\]

\[
\Delta \Theta^{\text{input}} = \Delta \Theta^{\text{base}} + \Delta \Theta^{\text{ref}} + \Delta \Theta^{\text{yref}} + \Delta \Theta^{\text{zref}} + S^{\text{noise}}
\]

\[
\Delta \Theta^{\text{output}} = \Delta \Theta^{\text{zbase}} + S^{\text{noise}}
\]

For comparison to the LMS gain convergence, Figure A29 and A31 show all three W0 (g0) and W1 (g1) values on the same plot respectively. For a better illustration, Figure A30 shows the individual W0 gain (coefficient) values with Figure A32 showing the individual W1 gain (coefficient) values. The same data set that was recorded and run through the LMS algorithm was used as the input data for the RLS algorithm. Therefore the PSD and CPSD shown in Figure A1 and A2 are valid for the correlated input data. Figure A33 and A34 show the uncorrelated output from the RLS algorithm.

The Full Matrix RLS resulted in Figures A35 and A37 showing all nine W0 and W1 gain values on the same plots respectively. Figure A36 shows the individual W0 gains values with Figure A38 showing the individual W1 gain values. The same data set that was recorded and run through the LMS algorithm was used as the input data for the RLS algorithm. Therefore the PSD and CPSPD shown in Figure A1 and A2 are valid for the correlated input data. Figure A39 and A40 show the uncorrelated output from the RLS algorithm. The plots reflect the convergence to a minimum error with a significant
reduction in real time, relative to the LMS algorithm, with very little degradation in error cancellation.

2.3.5 Joint Process Gradient Estimator Algorithm

To further the study of adaptive processes, another Main channel and Full matrix adaptive process was developed. While computationally more intensive, the Joint Process Gradient Estimator, JPGE, filter was developed in hopes of finding a more effective noise cancellation algorithm. The JPGE employs a recursive lattice structure. Lattice structures have long been known for their ease of implementation when considering digital signal processing. Because subsequent sections of the lattice are identical, implementing them in processing algorithms and Very Large Scale Integrated, VLSI, circuit devices becomes very straightforward. The basic concept behind a JPGE is that it is both a predictor and an estimator, hence the name joint process. Each subsequent lattice section essentially orthogonalizes the components of the reference inputs by prediction. The outputs of the lattice predictor are then incorporated in an adaptive estimation of those correlated components in the desired input.

While the derivation of the computations are beyond the scope of this paper, the JPGE finds its theory in the basics of the Grahm-Schmidt orthogonalization procedures, lattice prediction theory, and adaptive estimation theory. The gradient lattice filters predominant features are overall computational efficiency, very fast convergence, independence of the eigenvalue spread of the input covariance matrix, and modularity of its structure.

Both a Main channel and Full Matrix algorithm was developed in MATLAB. Comparison of the results of the Main channel to Full Matrix implementations is parallel in their effectiveness when considering noise cancellation. The Full Matrix formulation of the JPGE was not as effective on noise cancellation as the previous algorithms developed. Figure 2.5 illustrates a single channel JPGE diagram. A Main Channel algorithm would incorporate three of these diagrams, one for each channel. Figure 2.6 illustrates a Full Matrix Joint Process Gradient Estimator, FMJPGE, lattice with the cross coupling terms (gains) shown.

The basic Joint Process Gradient Estimator algorithm is listed in equations (2.25) through (2.26) below. Remember, while the below equations reflect a single channel, the algorithm was developed in matrix form for both the three channel Main and Full Matrix implementation.

At time n, the quantities \( \gamma_p(n), d_p(n) \) for \( p = 1,2,...,M \) and \( g_p(n), d_p^{-1}(n) \), for \( p = 0,1,2,...,M \) are available, as well as \( \Delta \Theta_{\text{input}} \) and \( \Delta \Theta_{\text{ref}} \).
Initialize:

\[ e_0^\pm(n) = \Delta \Theta_{\text{ref}}, \quad \Delta \Theta_{\text{est}}(n) = g_0(n)e_0'(n), \quad e_0(n) = \Delta \Theta_n - \Delta \Theta_{\text{est}}(n) \] (2.25)

\[ d_0(n) = \lambda d_0(n-1) + e_0'(n) \] (2.26)

\[ g_0(n+1) = g_0(n) + \left[ \beta / d_0(n) \right] e_0(n) e_0'(n) \] (2.27)

For order = 1, 2, ….

\[ e_0^+ = e_0^+(n) - \gamma_0(n) e_0^+(n-1) \] (2.28)

\[ e_0^- = e_0^-(n) - \gamma_0(n) e_0^+(n-1) \] (2.29)

\[ d_p(n) = \lambda d_p(n-1) + e_p^+(n) + e_p^-(n-1) \] (2.30)

\[ \gamma_p(n+1) = \gamma_p(n) + \left[ \beta / d_p(n) \right] \left[ e_p^+(n) e_p^-(n-1) + e_p^-(n) e_p^+(n-1) \right] \] (2.31)

\[ \Delta \Theta_{\text{est}}(n) = \Delta \Theta_{\text{est}}(n-1) - g_p(n)e_p^+(n) \] (2.32)

\[ e_p(n) = e_p(n-1) - g_p(n)e_p'(n) \] (2.33)

\[ d_p(n) = \lambda d_p(n-1) + e_p^+(n) \] (2.34)

\[ d_p(n) = \lambda d_p(n-1) + e_p^-(n) \] (2.35)

\[ g_p(n+1) = g_p(n) + \left[ \beta / d_p(n) \right] e_p(n) e_p'(n) \] (2.36)

Continue to the next time instant, \( n \rightarrow n + 1 \)
Figure 2.5 – Single Channel Joint Process Gradient Estimator (JPGE)

Figure 2.6 – Full Matrix Joint Process Gradient Estimator (JPGE)
Both main channel and full matrix data was plotted and can be referenced in figures A41 through A46 and A47 through A52 respectively. For a main channel with no cross terms operating, A41 illustrates the convergence of the adaptive weight updates while A42 reflects the predictive coefficients. A43 and A44 reflect PSD and CPSD of the first stage error output values, $E(2)$, while A44 and A45 reflect PSD and CPSD of the second stage errors, $E(3)$. For the Full Matrix JPGE, Figure A47 illustrates the adaptive weights while A48 reflects the prediction coefficients. Full Matrix PSD and CPSD plots for the first and second stage output errors are illustrated in Figures A49, A50, A51 and A52 respectively.

2.4 Second System Analysis

To show the importance of migrating from a main channel to full matrix implementation of the algorithms, a candidate system (#2) with known cross channel dither motion coupling was instrumented and data recorded. Figures A78 through A87 and Figures 92 through 100 reflect the same type of data recorded in previous algorithms. What can be seen is the relative reduction of uncorrelated gyro readout noise from the PSD and CPSD plots between the two algorithm outputs. Statistical data tabulated in this paper reflects a gross improvement in noise reduction. To reduce the amount of data attached, only the LMS algorithm was executed on system #2’s data.

2.5 Convergence and Stability

Trade off in convergence in the algorithms developed to date can be seen in the plots of the gains. For the LMS algorithms the trade off in convergence comes at the cost of a higher standard deviation of the uncorrelated noise. The convergence factor $\mu$, has an effect on the stability of the LMS much like the damping factor in a classic filter. Stability can be measured from one extreme, convergent, to the other extreme, non-convergent, with three incremental granularity measures as listed in Table 2.1 below. In each case the convergence factor is a function of the maximum eigenvalue, which establishes the performance surface slope.

<table>
<thead>
<tr>
<th>Stable(Convergent)</th>
<th>$0 &lt; \mu &lt; 1/\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overdamped</td>
<td>$0 &lt; \mu &lt; 1/2\lambda$</td>
</tr>
<tr>
<td>Critically Damped</td>
<td>$\mu = 1/2\lambda$</td>
</tr>
<tr>
<td>Underdamped</td>
<td>$1/2\lambda &lt; \mu &lt; 1/\lambda$</td>
</tr>
<tr>
<td>Unstable(Non-Convergent)</td>
<td>$\mu \geq 1/\lambda$ and $\mu &lt; 0$</td>
</tr>
</tbody>
</table>

For applications where the uncorrelated noise values are required to approach the theoretical limit, an over damped system requiring the slowest convergence times would be selected. For application that require fast convergence migration to the under damped
system would be selected with knowledge that the correlated noise values would have a larger standard deviation.

While the LMS algorithm is attractive because of its ease of implementation, the RLS algorithm offers both rapid convergence and low noise. Because the least squares process is minimizing a squared error that depends explicitly on the specific values of the input data and the reference input, the coefficients are optimal at each iteration and therefore the uncorrelated data approaches the optimum almost immediately.

It should be recognized that the convergence analysis of the RLS is based in independence theory and is beyond the scope of this writing. The results of the analysis can be summarized for a clear understanding of the RLS convergence properties. The LMS algorithm requires that the mean value convergence occur for \( n \to \infty \), dictated by its gradient search for the optimum weights (minimum error). The RLS algorithm is convergent in the mean value for the case that the number of transversal filter taps is less than or equal to the number of weights. The mean squared error in the weight vector, and ultimately the sensitivity of the RLS algorithm, is inversely proportional to the smallest eigenvalue. This implies that ill-conditioned least squares problems will undoubtedly lead to unstable convergence problems. The weight vector in the LMS tends to be convergent as an exponential due to the gradient estimation of the least mean squared error. The weight vector in the RLS algorithm decays linearly due to the very nature of the multiple linear regression model as it applies to the transversal filter within the RLS structure.

Deductions can be made by an analysis of the learning curves for the LMS and RLS algorithms. First, convergence of the LMS is explicitly dependent on the eigenvalues of the ensemble-averaged correlation matrix. The RLS algorithm is independent of the eigenvalues of the correlation matrix. As the number of iterations becomes large (infinity in the limit), the LMS algorithm produces an excess mean square error (average mean-square error less the minimum mean square error) as a result of the noise in the gradient search process. The RLS algorithm produces (in theory) zero excess mean squared error because the mean squared error approaches a value equal to the variance of the measurement error.

2.6 Computational Complexity

The algorithms developed in this research were coded using MATLAB: predominantly matrix formulation. While MATLAB is a computationally accurate tool, efficiency in coding comes at the cost of experience with the tool. It became intuitive as this research progressed that matrix formulations of the algorithms, while more concise, was less time efficient than extracted single line computations.

Computational efficiency is realized by examining the operations required for each of the algorithms. The LMS algorithm requires on the order of \( p \) (order)
multiplications and additions. The RLS algorithm increases its computational complexity by requiring \(3(p+1)^2+2(p+1)\) multiplications with an equal amount of additions. What is gained with this increase in computational complexity is performance in convergence and insensitivity to the eigenvalue spread of the correlation matrix for stationary data.

It should be noted that the RLS tracking is dependent on the exponential weighting factor. The LMS algorithm employed a high pass filter in the gain update equation to insure the gain updates did not track the rate data at the input to the algorithm. The RLS algorithm can adjust the weighting factor to minimize impact to gain updates; thereby minimizing the effects of rate input. Under extreme conditions the weight update equations in the RLS and JPGE may require high pass filters as well.

The JPGE is by far the worst algorithm of choice when considering computational complexity; however, the lattice structure is simple. Another trade is the modular structure of the JPGE allowing for ease of implementation in VLSI and signal processing applications. The convergence or the adaptation of the weights to their optimum supports the use of the JPGE algorithm in applications when rapid response is required. In summary, the table below lists the basic trade-offs of the three algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Speed</th>
<th>Complexity</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS</td>
<td>Slow</td>
<td>Simple</td>
<td>Stable</td>
</tr>
<tr>
<td>RLS</td>
<td>Fast</td>
<td>Complex</td>
<td>Stable</td>
</tr>
<tr>
<td>Lattice</td>
<td>Fast</td>
<td>Simple</td>
<td>Stable</td>
</tr>
</tbody>
</table>

2.7 Covariance Analysis of Algorithm Types

A good statistical measure of correlated noise cancellation for the different algorithms researched are the variance and covariance of the input data and the output data. A MATLAB script was used to calculate these values. The covariance matrix is defined as follows

\[
COV_{xy} = \begin{bmatrix}
\sigma_x^2 & r\sigma_x\sigma_y \\
r\sigma_x\sigma_y & \sigma_y^2
\end{bmatrix}
\]  \hspace{1cm} (2.37)

Where \(\sigma_x^2\) is the variance of \(x\), \(\sigma_y^2\) is the variance of \(y\), and \(r\sigma_x\sigma_y\) is the covariance of \(x\) and \(y\). From this the correlation coefficient \(r\) can be calculated by

\[
r = \frac{COV_{xy}}{\sqrt{\sigma_x^2 \sigma_y^2}}
\]  \hspace{1cm} (2.38)
For each of the algorithms, the variance and covariance of the input parameters, output parameters and input to output parameters is calculated using the mean value of a 50-point sliding window. Figure A53 through A77 show a variety of statistical data for each of the algorithm types implemented using system #1 data. Specifically Figure A53 and A54 respectively show the variance of the X, Y and Z correlated gyro readout input, $\Delta \Theta_{\text{input}}$, and the dither reference (also referred to as the dither pick off) input, $\Delta \Theta_{\text{ref}}$. Figure A55 plots the covariance of these two inputs reflecting a higher degree of correlation in the two signals. A good measure of how closely the two signals are correlated is illustrated in Figure A56, showing a value of the correlation coefficient to be nearly one. Table 2.3 tabulates the mean values calculated for each of the statistical parameters for both system #1 and system #2.

To show the effectiveness of the noise cancellation, the variance $V_{\text{ung}}$, of the uncorrelated gyro readout noise output, $\Delta \Theta_{\text{output}}$, was calculated and plotted for each of the algorithm types. To show reduction in correlation, the covariance, $C_{V_{\text{ung}}}$, of each of the algorithms output, $\Delta \Theta_{\text{output}}$ to the reference input, $\Delta \Theta_{\text{ref}}$ was calculated. Again, as a true measure of the correlation between the input reference, $\Delta \Theta_{\text{ref}}$, to the uncorrelated output signal, $\Delta \Theta_{\text{output}}$, the correlation coefficient, $C_{C_{V_{\text{ung}}}}$, was calculated and plotted.

Figure A57 is a plot of the three channels of $V_{\text{ung}}$, $V_{\text{unpy}}$, $V_{\text{unpgz}}$ for the main channel LMS. From the left of the plot it can be seen that the variance of the data is large prior to the gains converging to their optimal value. As the gains do converge, the variance of the data is reduced to near zero. Figure A58 is a plot of the variance $C_{V_{\text{ung}}}$, $C_{V_{\text{unpy}}}$, $C_{V_{\text{unpgz}}}$, for the main channel LMS. Again it is evident that a high degree of correlation exists between the reference input and LMS output at the start of the algorithm with a decreasing covariance value as the gains converge. The correlation coefficient bares this out as well. Figure A59 reflects a value of the correlation coefficients, $C_{C_{V_{\text{ung}}}}$, $C_{C_{V_{\text{unpy}}}}$, $C_{C_{V_{\text{unpgz}}}}$ close to value of one at the start of the algorithm slowly converging to a value about zero once the adaptive weights converge. For each of the other algorithms, the same three plots were generated in the respective order, variance, covariances, and correlation coefficients. Figure A60 through A62 are these same plots for the Full Matrix LMS algorithm. Figures A63 through A65 are plots for these parameters for the full matrix Normalized LMS algorithm. The main channel RLS algorithms’ statistical parameters are illustrated in Figures A66 through A68 with Figures A69 through A71 illustrating the Full Matrix RLS statistical parameters. For the JPGE, the variance of the main channels uncorrelated readout noise and its covariance with the dither reference signal are plotted in Figure A72 and A73 respectively, with the correlation coefficient being plotted in Figure A74. In each case, the variances, covariance’s, and correlation coefficients converge to their minimum with the adaptive algorithms’ convergence of the gains. The mean value of the 50-point sliding window of the statistical data was tabulated and can be seen in Table 2.4.
A second system was used as a candidate for showing a high degree of cross coupling the mechanical dither frequencies. The same data was generated for the second system as that for the first system. To reduce the amount of data generated only the main channel and full matrix LMS algorithm was run on this second system. Figure A78 and A79 show the PSD and CPBD of the correlated gyro readout noise inputs. Figure A80 and A81 show the PSD and CPBD of the dither reference signals. Figure A82 through A85 show the adaptive gains for the main channel LMS. The uncorrelated gyro readout noise PSD and CPBD for System #2 is illustrated in Figure A86 and A87 respectively. Figure A88 through A94 show the statistical data as described in the preceding paragraphs. Table 2.3 and 2.5 list the parameters, the 50-point sliding window mean value and associated reference figure numbers.

In comparison to the main channel LMS data, the same data was generated for the Full Matrix LMS for System #2. For system #2 Figure A95 through A98 show the adaptive gains for the Full Matrix LMS. Figure A99 and A100 show the Full Matrix PSD and CPBD of the uncorrelated gyro readout noise. Figure A101 through A103 show the same type of statistical data for the LMS Algorithm performed on System #2 as was generated for System #1. Table 2.3 shows the variance of the correlated RLG inputs and the dither reference inputs. Also listed is the covariance value of the two inputs relative to one another. The correlation coefficient between the correlated gyro readout noise input and the dither reference input for the X, Y, and Z channels.

Table 2.3 – Statistic Measurement for System #1 and #2 Input Parameters

<table>
<thead>
<tr>
<th>Statistic Parameter</th>
<th>Reference Figure System #1</th>
<th>System #1 input Data</th>
<th>Reference Figure System #2</th>
<th>System #2 input Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcorgx</td>
<td>Figure A53</td>
<td>1.267e5</td>
<td>Figure A88</td>
<td>1.940e6</td>
</tr>
<tr>
<td>Vcorgy</td>
<td>Figure A53</td>
<td>1.033e5</td>
<td>Figure A88</td>
<td>9.466e6</td>
</tr>
<tr>
<td>Vcorgz</td>
<td>Figure A53</td>
<td>1.066e5</td>
<td>Figure A88</td>
<td>1.298e6</td>
</tr>
<tr>
<td>Vpx</td>
<td>Figure A54</td>
<td>2.710e6</td>
<td>Figure A89</td>
<td>4.535e7</td>
</tr>
<tr>
<td>Vpy</td>
<td>Figure A54</td>
<td>2.596e6</td>
<td>Figure A89</td>
<td>4.662e7</td>
</tr>
<tr>
<td>Vpz</td>
<td>Figure A54</td>
<td>2.467e6</td>
<td>Figure A89</td>
<td>4.557e7</td>
</tr>
<tr>
<td>CVcorgpx</td>
<td>Figure A55</td>
<td>5.857e5</td>
<td>Figure A90</td>
<td>-7.0143e6</td>
</tr>
<tr>
<td>CVcorgpy</td>
<td>Figure A55</td>
<td>5.1783e5</td>
<td>Figure A90</td>
<td>6.62086e6</td>
</tr>
<tr>
<td>CVcorgpz</td>
<td>Figure A55</td>
<td>5.1146e5</td>
<td>Figure A90</td>
<td>7.584e6</td>
</tr>
<tr>
<td>CCcorgpx</td>
<td>Figure A56</td>
<td>0.9995</td>
<td>Figure A90</td>
<td>-0.9958</td>
</tr>
<tr>
<td>CCcorgpy</td>
<td>Figure A56</td>
<td>0.9996</td>
<td>Figure A90</td>
<td>0.9966</td>
</tr>
<tr>
<td>CCcorgpz</td>
<td>Figure A56</td>
<td>0.9996</td>
<td>Figure A90</td>
<td>0.9970</td>
</tr>
</tbody>
</table>
Table 2.4 – Statistic Parameter Acronym Definitions I

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{corgx}$</td>
<td>Variance of Correlated Gyro Readout Noise X Channel</td>
</tr>
<tr>
<td>$V_{corgy}$</td>
<td>Variance of Correlated Gyro Readout Noise Y Channel</td>
</tr>
<tr>
<td>$V_{corgz}$</td>
<td>Variance of Correlated Gyro Readout Noise Z Channel</td>
</tr>
<tr>
<td>$V_{px}$</td>
<td>Variance Dither Reference X Channel</td>
</tr>
<tr>
<td>$V_{py}$</td>
<td>Variance Dither Reference Y Channel</td>
</tr>
<tr>
<td>$V_{pz}$</td>
<td>Variance Dither Reference Z Channel</td>
</tr>
<tr>
<td>$C V_{corgpx}$</td>
<td>Covariance of the X Channel Correlated Gyro Readout Noise to X Dither Reference</td>
</tr>
<tr>
<td>$C V_{corgpy}$</td>
<td>Covariance of the Y Channel Correlated Gyro Readout Noise to Y Dither Reference</td>
</tr>
<tr>
<td>$C V_{corgpz}$</td>
<td>Covariance of the Z Channel Correlated Gyro Readout Noise to Z Dither Reference</td>
</tr>
<tr>
<td>$C C_{corgpx}$</td>
<td>Correlation Coefficient X Channel Correlated Gyro Readout Noise to X Dither Reference</td>
</tr>
<tr>
<td>$C C_{corgpy}$</td>
<td>Correlation Coefficient Y Channel Correlated Gyro Readout Noise to Y Dither Reference</td>
</tr>
<tr>
<td>$C C_{corgpz}$</td>
<td>Correlation Coefficient Z Channel Correlated Gyro Readout Noise to Z Dither Reference</td>
</tr>
</tbody>
</table>

Comparing table 2.3 with table 2.5 entries, it can be seen that the algorithms are fairly close in their ability to strip the reference inputs. The variances for the uncorrelated gyro readout noise, $V_{ungx}$, $V_{ungy}$ and $V_{ungz}$ of the two systems track very closely to one another. For System #1 it is difficult to see the benefit of the full matrix implementation for any of the algorithms. For instance, the $V_{ungx}$ parameter actually increases for each of the algorithm types when going from a main channel (-D) to a full matrix (-FM) implementation.

For System #2, it is readily apparent by the data in Table 2.5 that the full matrix implementation reduces the cross-coupled frequency components from channel to channel. In comparison, the uncorrelated gyro readout noise magnitude in the CPSD plots of System #2, Figure A87 and A100, reflect a factor of 4x reduction in noise magnitude. The variance values show a maximum variance variation of almost 7x for the X channel with a minimum of 2x for the Z channel. It is also interesting to note the frequency components in the PSD plots for the uncorrelated gyro readout noise of System #2 when comparing the main channel to full matrix implementations. The initial dither frequencies for System #2 are around 625 Hz for the X channel, 575 Hz for the Y channel and 525 Hz for the Z channel (see figure A78 PSD Plot). After the main channel LMS, a noticeable reduction in these three frequencies is evident (see Figure A86 PSD plot). Notice that the components of dither frequencies that are illustrated in Figure A86 happen to be cross-coupled frequencies. This can be seen because we have the Z channels data...
reflecting a 625 Hz dither frequency component of X channels dither frequency, 
validating that there is, in fact, cross coupling of the dither frequencies. Further 
inspection reveals that not only do we have the cross-coupled dither frequencies but we 
also have folded harmonics as a function of sampling. When we reference Figure A99 
PSD plot we see the cross-coupled dither frequency magnitudes greatly reduced, 
validating the full matrix algorithm is in fact stripping the cross-coupled dither 
components.

Table 2.5 – Statistical Measurements for System #1

<table>
<thead>
<tr>
<th>Statistic Parameter</th>
<th>LMS-D</th>
<th>LMS-FM</th>
<th>NLMS-D</th>
<th>NLMS-FM</th>
<th>RLS-D</th>
<th>RLS-FM</th>
<th>JPEG-D</th>
<th>JPEG-FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vungx</td>
<td>.378</td>
<td>.409</td>
<td>.331</td>
<td>.335</td>
<td>.3615</td>
<td>.447</td>
<td>0.365</td>
<td>.413</td>
</tr>
<tr>
<td>Vungy</td>
<td>.550</td>
<td>.589</td>
<td>.546</td>
<td>.549</td>
<td>.5229</td>
<td>.630</td>
<td>0.527</td>
<td>.539</td>
</tr>
<tr>
<td>Vungz</td>
<td>.560</td>
<td>.515</td>
<td>.504</td>
<td>.435</td>
<td>.5553</td>
<td>.600</td>
<td>0.537</td>
<td>.480</td>
</tr>
<tr>
<td>CVungpy</td>
<td>2.490</td>
<td>0.701</td>
<td>26.54</td>
<td>25.045</td>
<td>-28.26</td>
<td>3.8761</td>
<td>27.760</td>
<td>-106.63</td>
</tr>
<tr>
<td>CCungpx</td>
<td>-0.004</td>
<td>-0.002</td>
<td>0.011</td>
<td>0.012</td>
<td>-0.010</td>
<td>-0.295</td>
<td>0.017</td>
<td>-0.184</td>
</tr>
<tr>
<td>CCungpy</td>
<td>0.002</td>
<td>-0.001</td>
<td>0.025</td>
<td>0.026</td>
<td>-0.023</td>
<td>0.2958</td>
<td>0.024</td>
<td>0.089</td>
</tr>
<tr>
<td>CCungpz</td>
<td>0.007</td>
<td>0.008</td>
<td>0.004</td>
<td>0.009</td>
<td>-0.023</td>
<td>0.3078</td>
<td>0.043</td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table 2.6 – Statistical Measurements for System #2

<table>
<thead>
<tr>
<th>Statistic Parameter</th>
<th>LMS-D</th>
<th>Reference Figure</th>
<th>LMS-FM</th>
<th>Reference Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vungx</td>
<td>14.5389</td>
<td>Figure A92</td>
<td>2.8472</td>
<td>Figure A101</td>
</tr>
<tr>
<td>Vungy</td>
<td>11.8725</td>
<td>Figure A92</td>
<td>2.3216</td>
<td>Figure A101</td>
</tr>
<tr>
<td>Vungz</td>
<td>8.6235</td>
<td>Figure A92</td>
<td>3.787</td>
<td>Figure A101</td>
</tr>
<tr>
<td>CVungpx</td>
<td>-2.426e2</td>
<td>Figure A93</td>
<td>-1.596e2</td>
<td>Figure A102</td>
</tr>
<tr>
<td>CVungpy</td>
<td>1.5317e3</td>
<td>Figure A93</td>
<td>1.575e3</td>
<td>Figure A102</td>
</tr>
<tr>
<td>CVungpz</td>
<td>4.4234e2</td>
<td>Figure A93</td>
<td>4.651e2</td>
<td>Figure A102</td>
</tr>
<tr>
<td>CCungpx</td>
<td>-0.0135</td>
<td>Figure A93</td>
<td>-0.0224</td>
<td>Figure A103</td>
</tr>
<tr>
<td>CCungpy</td>
<td>0.0766</td>
<td>Figure A93</td>
<td>.1632</td>
<td>Figure A103</td>
</tr>
<tr>
<td>CCungpz</td>
<td>0.0354</td>
<td>Figure A93</td>
<td>.0577</td>
<td>Figure A103</td>
</tr>
</tbody>
</table>
Table 2.7 – Algorithm Acronym Definitions

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS-D</td>
<td>Least Mean Square Diagonal (Main channel cancellation only)</td>
</tr>
<tr>
<td>LMS-FM</td>
<td>Least Mean Square Full Matrix (Main channel and cross channel cancellation)</td>
</tr>
<tr>
<td>NLMS-D</td>
<td>Normalized Least Mean Square Diagonal</td>
</tr>
<tr>
<td>NLMS-FM</td>
<td>Normalized Least Mean Square Full Matrix</td>
</tr>
<tr>
<td>RLS-D</td>
<td>Recursive Least Square Diagonal</td>
</tr>
<tr>
<td>RLS-FM</td>
<td>Recursive Least Square Full Matrix</td>
</tr>
<tr>
<td>JPGE-D</td>
<td>Joint Process Gradient Lattice Diagonal</td>
</tr>
<tr>
<td>JPGE-FM</td>
<td>Joint Process Gradient Lattice Full Matrix</td>
</tr>
</tbody>
</table>

Table 2.8 – Statistic Parameter Acronym Definitions II

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vungx</td>
<td>Variance of Uncorrelated (Stripped) Gyro Readout noise X channel</td>
</tr>
<tr>
<td>Vungy</td>
<td>Variance of Uncorrelated (Stripped) Gyro Readout noise Y channel</td>
</tr>
<tr>
<td>Vungz</td>
<td>Variance of Uncorrelated (Stripped) Gyro Readout noise Z channel</td>
</tr>
<tr>
<td>Cvungpx</td>
<td>Covariance of Uncorrelated (Stripped) Gyro Readout noise X channel to Dither Reference X</td>
</tr>
<tr>
<td>CVungpy</td>
<td>Covariance of Uncorrelated (Stripped) Gyro Readout noise Y channel to Dither Reference Y</td>
</tr>
<tr>
<td>CVungpy</td>
<td>Covariance of Uncorrelated (Stripped) Gyro Readout noise Z channel to Dither Reference Z</td>
</tr>
<tr>
<td>Ccungpx</td>
<td>Correlation Coefficient of Uncorrelated (Stripped) Gyro Readout noise X channel to Dither Reference X</td>
</tr>
<tr>
<td>Ccungpy</td>
<td>Correlation Coefficient of Uncorrelated (Stripped) Gyro Readout noise Y channel to Dither Reference Y</td>
</tr>
<tr>
<td>Ccungz</td>
<td>Correlation Coefficient of Uncorrelated (Stripped) Gyro Readout noise Z channel to Dither Reference Z</td>
</tr>
</tbody>
</table>

2.8 Adaptive Algorithm Summary

Basic LMS, RLS and JPGE structures have been implemented. Statistical Analysis of the structures reveals that each of the algorithms has advantages and disadvantages. The LMS reflects slow convergence properties but simplicity in design. The RLS algorithm shows fast convergence with complexity in design. The Joint Process Gradient Estimator reveals modularity of design, fast convergence and increase in computational operations. Statistical data reveals trade offs in a main channel to full matrix implementation when considering two system configurations. It is apparent that for some systems the expansion of the algorithm to a full matrix structure does not insure a decrease in uncorrelated gyro readout noise magnitudes. The data illustrates that for those systems where uncorrelated noise magnitudes are influenced by cross-coupled dither components, the full matrix algorithm is warranted. From system to system, the requirements dictating noise levels will ultimately dictate which algorithm should be used.
The derivation of main and full matrix real time adaptive filter structures supports evaluating performance tradeoffs such as convergence rate, effectiveness of correlated noise cancellation, organizational complexity and computational efficiency. Development of real time adaptive filters for use in dithered ring laser gyro based inertial systems will insure an optimal selection of candidate structures for a wide variety of commercial and military applications.
Chapter Three

Considerations For Hardware Development

3.1 Development

Once the algorithms were developed, the focus of the effort targets the implementation of a real time adaptive filter algorithm into an Application Specific Integrated Circuit (ASIC) using Very High Scale Integrated Circuit Hardware Description Language (VHDL). Specifically, a candidate structure was selected, industry tool suites were researched and selected, an algorithm was designed, compiled and validated using VHDL, a test bench environment was written in VHDL for use in digital simulation, validation and analysis, platform scripts were written to insure that an automated, systematic and orderly control of data files was in place and preliminary analysis of the algorithm output data was accomplished.

The goal of the effort was to synthesize the behavioral VHDL into structural VHDL (logic gates) targeting Honeywell Inc. Radiation Insensitive Metal Oxide Semiconductor II (RICMOS II) ASIC technology and analyze the effects of finite register lengths on system performance parameters. The effort fell short of this goal. A need to migrate from workstation platform tool types became necessary. This migration was due to tool and license obsolescence. A lot of effort went into understanding the tools initially selected with research directed toward both a bit serial and bit parallel implementation.

Of the algorithms researched and developed, the Least Mean Square (LMS) algorithm was selected for migration to VHDL. Organizational and computational simplicity allows for the LMS algorithm to be easily migrated with little degradation to performance parameters such as convergence, stability, and correlated noise magnitude reduction. The selection of this algorithm will satisfy the evaluation of these performance parameters when considering finite word length effects, limit cycling, and hardware resource implementation limitations.

3.2 A Query of Tool Suites

The Frontier Design Tool DSP Station’s MISTRAL I and MISTRAL II were originally selected as the tool suite for the research effort. Both tools were researched, used and subsequently abandoned. The MISTRAL I tool suite was used early on in the
research in hopes of generating a bit serial implementation of the LMS algorithm. Little success was obtained in generating/compiling the Design Flow Language into usable VHDL code for ASIC targets.

The bit parallel implementation tool suite, MISTRAL II, was used later in the research with some success in generating structures requiring programmable read only memory and random access memory devices. These structures may be warranted when considering the migration to a full matrix implementation of any of the adaptive structured researched, however the initial intent was to target a single channel for evaluation.

Both of these tool suites have been phased out by Frontier Design and are now obsolete. Frontier Design has migrated to a new tool suite and no longer supports the DSP Work Station tool suite including the MISTRAL I & II compilers. Initial costs in seat license and learning curve effort made the use of the new tool suite prohibitive. As a result, it was decided to approach the VHDL at a less abstract (component) implementation level and write the VHDL behaviorally, making it independent of the higher level (system) of abstraction code compilers and generators.

This writing reflects all of the work done with the exception of the MISTRAL I & II work outputs. The data structures generated for that effort were consistently generated in real time as a function of tool invocation. Tool and License obsolescence prohibit acquiring any usable output. Early work reflecting the Design Flow Language is presented to reflect work accomplished under the research funding.

Three ASIC targeting Computer Aided Design environments were researched. Mentor Graphic Corporations’ Interactive Architectural Behavioral Design Exploration Monet, Synopsys Corporations’ COSSAP and Frontier Designs’ DSP Station tool suites were traded with the selection of Frontier Designs’ DSP Station for the migration process. While each tool suite provides robust and adequate components for each step in the design process cycle, availability at the time of the Frontier Design Corporations tool suite through Honeywell Inc. Computer Aided Design ultimately dictated its use. The Frontier Design DSP Station tool suite was a third party tool operating under Mentor Graphics Corporations’ Falcon Framework platform.

3.2.1 Mentor Graphics Monet

The first tool suite looked at was Mentor Graphics Corporations’ Monet. This is primarily an interactive architectural exploration tool used to evaluate algorithms at a higher level of abstraction behavioral level. The tool allows for evaluation of an algorithm for design trade-offs in algorithm speed, operational complexity, resource utilization, delay constraints and data flow relationships. Monet allows the designer to quickly assess alternatives at the behavioral level and then automatically generate VHDL Register Transfer Logic (RTL) for synthesis. Algorithms developed can be evaluated
with tool outputs such as data flow diagrams, Gantt charts, state diagrams, and data path schematics. Flexibility in the tool allows for VHDL generation anywhere in the process so that simulation, synthesis and analysis can be accomplished using other downstream tools. Figure 3.1 depicts a simplified path when using the Monet tool.

Figure 3.1 – Monet Algorithm/Architecture Design Environment

3.2.2 Synopsys COSSAP

The second tool researched was Synopsys Corporations COSSAP. COSSAP is marketed as a complete system level design environment. It supports a wide range of design levels from systems to logic. It provides a variety of application capabilities for the design, development, test and analysis of digital signal processing algorithms, architectures and implementations. Figure 3.2 shows the COSSAP design environment reflecting a robust front end tool suite for developing DSP algorithms and various levels of abstraction for both hardware and software implementations. The Synopsys computer aided design environment is considered one of the industries premier logic synthesis tool suites when targeting ASIC development.
3.3.3 Frontier Design Digital Signal Processing Station

The third digital signal processing Computer Aided Design tool suite researched was Frontier Designs’ Digital Signal Processing (DSP) Station. Figure 3.3 depicts the DSP Station design environment. This tool suite is specifically designed with DSP algorithm to ASIC implementation migration path in mind. The DSP Station is a third party tool suite operating in conjunction with Mentor Graphics Falcon FrameWork platform. A very robust front-end design environment gives maximum flexibility for algorithm development through schematics, Design Flow Language, DSP C or ANSI-C.
Post processing and simulation environment exists through the DSP Design Lab for the interim algorithm evaluation with analysis tools being provided through Filter Design Lab for finite word length effects and limit cycling evaluation.

Figure 3.3 – Frontier Design Digital Signal Processing Station Platform
Costs for each of the primary DSP tools, and the platform environments that they operate under, were in the tens of thousands of dollars. Hence, selection of the tool to be used was dictated by availability. Earlier in the research, Honeywell Inc. had updated their contracts with Y2K compliant copies of Frontier Design DSP Station and Mentor Graphics Falcon Frame Work tool suites, thereby making them accessible for development.
Chapter Four

Algorithm Implementation

4.1 Direction

Direction to proceed with implementation is always a milestone point in the process. Decisions made at this juncture can have drastic and costly effects in costs and schedule. Cost and availability, support and obsolescence and tool automation and ease of use all effect the migration of the design from algorithm to silicon: as will be seen.

4.2 Algorithm Development Using DSP Station

Using Frontier Designs DSP Station, a three-channel LMS algorithm was developed. The LMS algorithm was implemented in schematic form and then compiled into the Design Flow Language specific to the DSP Station environment. Figure 4 illustrates a single channel LMS schematic. The schematic diagram was replicated for each of the three orthogonal RLG Channels. This replicated the orthogonal or main channel adaptive filter structure implemented in MATLAB. This implementation does not provide for cross channel correlated noise reduction.

Figure 4.1 – Schematic Representation of Single Channel Real Time LMS Filter

As part of the schematic diagram capture process, the LMS component provided with the DSP Station Libraries was updated to initialize internal variables that were

35
causing immediate overflow at the algorithm start. Actual recorded RLG sensor data from a candidate system was used to exercise the inputs to the three channels. Offsets were added as required because of previous processing within the RLG systems. The sampling of the RLG and Dither Reference data within the system is a running integration and requires a delta calculation prior to feed forwarding into the LMS core.

Data taken at the inputs and outputs include: the dither reference inputs, DPOX, DPOY, and DPOZ, the RLG inputs, THETAX, THETAY and THETAZ, the stripped (filtered) RLG information, THETA_STRIPPED_X, THETA_STRIPPED_Y and THETA_STRIPPED_Z, and the first coefficient of a two-coefficient model, Coefficient_h0X, Coefficient_Y, and Coefficient_Z. Reference Figure 4.2.

Subsequently FFTs were taken to show the frequency component magnitudes of the original RLG Data, THETAX_fft, the Dither Reference data, DPOX_fft, and the stripped (filtered) RLG Data, Theta_Stripped_x_fft. The system sampling rate was 1600 Hz. The FFT plots in Figure 6 show an output folded about the nyquist bandwidth 0 to 800 Hz. The tool does not omit the folded data and subsequently the plot includes it. The data above the 800 Hz frequency is a function of the FFT algorithm and not true frequency content.

Figure 4.2 – Operational LMS Input/Output Trace History
The development process was to be continued by evaluating the finite register length limit required to meet the dynamic range and quantization levels needed to operate the algorithm efficiently. The DSP Station provided tools for word length and limit cycling analysis. This work was subsequently abandoned due to tool and license obsolescence. The goal was to evaluate the scaling and word length of data parameters at each node in the algorithm and then adjust to optimize the design for migration to VHDL implementation. This would have insured an efficient target of ASIC resource utilization while maintaining performance optimization.

Parallel to the optimization process described above, the design was compiled using the Mistral I Bit Serial Behavioral VHDL Compiler. This process was laborious and was never able to provide efficient conversion from the Design Flow Language (DFL) output to behavioral VHDL. The VHDL generated was unable to be turnkey synthesized to structural VHDL implementing simulatable logic gates using either Mentor Graphics Leonardo Synthesis tool or Synopsys Design Compiler synthesis tools. A VHDL test bench was created with the appropriate Test Vector Generators (TVGs) to exercise the design at the logic level. Data Analysis was then to be executed to insure operational compliance and performance requirements could be met.
With the continued inability of generating efficient and synthesizable structural VHDL code from the bit serial MISTRAL I tool, the research migrated to the bit parallel MISTRAL II tool. Data structures using programmable read only memory (PROM) and random access memory (RAM) were generated in conjunction with a higher level of abstraction of behavioral VHDL code. These data structures were inherent in the tool suite compilation process with little success in preventing the use of such high levels of abstraction. About this time in the research effort, the tool suite license supporting both MISTRAL I and MISTRAL II became obsolete. The other two tool suites initially researched, MONET and COSSAP, were not well understood and the licensing for user seats was cost prohibitive making the migration of the research effort in that direction impossible.

4.3 Tool Independent Development

With effort expended on understanding and using these high level compilation tools, and the costs associated with migrating to another high level tool suite, it was decided to generate the VHDL using lower levels of behavioral abstraction. This would allow the research effort to depend on tools that have been common to the ASIC user community for years and made the VHDL code generated independent of high levels of abstraction compilers and tool suites.

4.4 VHDL and Functional Simulation

Starting nearly from scratch, the LMS algorithm was generated in behavioral VHDL in support of a single channelized approach. With a single channel generated, it is easy to migrate to a three-channel implementation for use in a strictly orthogonal (no cross channel reduction of coupled noise) configuration. The VHDL test bench previously generated was in large part usable but had to be rewritten, in part, to support the lower level behavioral approach. A suitable directory structure was defined and implemented. Platform script files were written in order to provide systematic and orderly control over tool invocation, data reception, data storage and user access. Functional simulation was performed using Mentor Graphics Modelsim at sub functional levels, with migration toward a top-level hierarchical simulation at the top level.

Data analysis was accomplished by porting the output of the LMS VHDL to a PC platform. Data from both the MATLAB algorithm and VHDL algorithm were subsequently read into MathCAD using script files. These script files were used to restructure the data and generate PSD and CPSD graphs.

A self-contained test bench was developed in VHDL consisting of the LMS filter component, data read and write components and clock generation components. The test-bench supports timing and control of clock generation, data input/output and LMS filter execution times.
4.5 ASIC Selection

Preliminary trade studies were accomplished in deciding the Application Specific Integrated Circuit, ASIC, device to target for the LMS algorithm. An architecture was selected. Synthesizers were researched. Synthesis of the behavioral VHDL to Structural VHDL was accomplished. Layout of logic in candidate ASIC device was accomplished. Post layout simulation and validation were performed. Worst-case timing analysis was performed. Finite register length effects were evaluated.

The term ASIC has become a generic term for Sea of Gates, Custom Array or Field Programmable Gate Array, FPGA, architectures. Selection criteria for which device used depends on system flow-down requirements in the application intended. For commercial and military non-space products, there is a vast array of ASIC devices available. The availability of devices for space based systems narrows significantly due to the relative costs associated with manufacturing these devices to survive the stringent radiation tolerant environment. Regardless of the system environments, parameters such as size, power, speed, reliability, costs and risks to development schedule are considered in selecting the device.

For both commercial and military product developments, FPGA devices have been the designer's choice. Re-programmability of these device help to mitigate risks to final product cost and schedule due to multi-pass design cycles. Advancements in micro-circuit design have helped to make the FPGA competitive to the sea of gate and standard cell arrays when considering size, gate count, power, unit device and design and development costs. In recent years larger gate count FPGA devices have entered into the market allowing designers to target final product configuration with these devices. While million gates commercial FPGA devices are now common in the market, availability of higher gate count radiation tolerant FPGAs is still lacking. However, for smaller designs, radiation tolerant FPGAs are now becoming commonplace. Architectures supporting triple module redundancy with embedded voting schemes and advancements in Complimentary Metal Oxide Semiconductor, CMOS, fabricated on silicon on sapphire substrate have helped to push the FPGAs ability to survive in harsh radiation environments.

The target for the LMS VHDL was selected based on availability of tools supporting the advancement of the algorithm from VHDL to logic gates and the ability to migrate the development design to a final product design with minimum impact to cost and time to market cycle. The device selected is the ACTEL RT54SX72S FPGA. The ACTEL RT54SX-S architecture is based on a high voltage twin-well CMOS process using 0.25 micron geometry design rules. It is a metal-to-metal anti-fuse device with a very low on state resistance of <25 ohms and a capacitance of 1.0 femto-farad, providing for very low signal impedance. The device can operate with internal frequencies of 300 MHz allowing for very fast execution of embedded algorithms. It implements a triple
module redundancy architecture that makes it suitable for all but strategic level (nuclear) spaced based applications. The 256 pin package allows development in a less costly commercial device and migration to a pin for pin compatible space high-reliability device. The operational voltage selection allows for a +2.5 VDC array core and a +5 VDC I/O ring, minimizing the need for level translation when using other standard +5 VDC CMOS circuits in the design.

### 4.6 Synthesis

Today’s supporting VHDL computer aided design tool suites offer a wide range of VHDL to logic gate synthesizers. Mentor Graphic Leonardo, Synopsis Design Compiler, and Synplicity Synplify_Pro are used extensively throughout the industry. Trades on which synthesizer to use include cost, availability, ease of use, architecture library support and synthesis execution time. All of the aforementioned synthesizers were evaluated. Each of the synthesizers is available, will support the selected architecture and is comparably easing to use. The original decision was to use Synopsis Design Compiler. Difficulty with routing internal signals to I/O ports forced a change of the synthesis tools to the Synplify Pro synthesizer. Synplicity’s Synplify_Pro is a stand-alone third party tool that has shown exceptional ease of use and speed of synthesis with minimum need for user intervention requiring manual placement and route.

The Synplify_Pro allows for either command script or graphic user interface. For larger, very high-speed designs the user is often required to generate constraint scripts that control the synthesis output. The LMS algorithm operates at a synchronous 16 MHz clock speed with the data path flows operating at 1600 Hz. With these relatively low execution times, constraint scripts were not required so that the graphic user interface was used.

The LMS single channel design was synthesized with no errors. Additions of VHDL buffer registers were required in order to port the gain h0 and h1 values to the chip I/O for evaluation. In a production design, these values would not be required outputs, thereby minimizing sequential logic and I/O usage. Warnings were generated reflecting an optimization and elimination of superfluous logic. The synthesizer reads the Behavior VHDL as input and generates structural VHDL as an output. The structural VHDL is in a logically flattened format necessary for layout place and route tools. The original VHDL was written at a Register Transfer Logic (RTL) level as opposed to a higher level of abstraction so that the designers could control the design implementation. Higher levels of behavioral VHDL coding style leave the structure of the design to the synthesizer, thereby making it more difficult to debug and evaluate. The synthesized structural VHDL is available in hard copy on request.
4.7 Place and Route

Targeting the Actel FPGA device warranted use of the Actel suite of place and route tools. Synthesis output is read in by the Actel Designer tool and, via a graphical user interface, allows for placement and routing of the synthesized design. Once placed and routed, the design is then evaluated for post layout functionality. The Actel Designer tool supports timing analysis, I/O pin to signal assignments and back annotation of the routed design.

Timing analysis reflects no violations in meeting setup and hold times. I/O port to signal assignments were not constrained and the router was allowed to assign signals to pins. In a production design, some constraints for signal to pin assignments would be undertaken in order to facilitate printed wiring board flow and noise coupling considerations. Bus structures would be placed to minimize ground bounce and high frequency signals would be isolated between unused pins as appropriate. For this effort pin placement is not critical. The synthesized schematic of the place and routed design are available in hard copy on request.

4.8 Worst Case Timing

Maximum route time constraints of 62.5 ns (1/16 MHz) were placed on the place and route tool to meet the register to register, input to register, output to register and input to output over worst case temperature, voltage and total dose radiation and manufacturing process variations. Table 4.1 tabulates the worst-case analysis summary. A snapshot of the worst-case path delays is listed in Appendix E.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Requirement</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Clock Speed</td>
<td>16 MHz</td>
<td>17.76 MHz</td>
</tr>
<tr>
<td>Maximum Input To Register Delay</td>
<td>62.5 ns</td>
<td>35.86 ns</td>
</tr>
<tr>
<td>Maximum Register to Output Delay</td>
<td>62.5 ns</td>
<td>10.82 ns</td>
</tr>
<tr>
<td>Maximum Input to Output</td>
<td>62.5 ns</td>
<td>No paths</td>
</tr>
</tbody>
</table>

4.9 Post Route Simulation

Once timing analysis is performed the place and routed design was then back annotated. The back annotation is output with a standard delay format with actual FPGA device delays embedded, which is used in the same test bench environment that the behavior VHDL was validated in. The test bench validates the post route design for operational compliance and validates the worst-case analysis. The output of the
simulations is then transferred to a PC environment where the data can be read by
MathCAD scripts for evaluation and comparison to prior MATBLAB algorithm and
simulation results.

The VHDL test bench previously generated was in large part usable but had to be
updated, in part, to support the structural back annotated simulation. A suitable directory
structure was defined and implemented. Platform script files were written in order to
provide systematic and orderly control over tool invocation, data reception, data storage
and user access.

Data analysis was accomplished by porting the output of the LMS VHDL to a PC
platform. Data from both the MATLAB algorithm and VHDL algorithm were
subsequently read into MathCAD using script files. These script files were used to
restructure the data and generate PSD and CPSD graphs of the post route design.

4.10 Test Bench for Simulation and Validation

A self-contained test bench was developed in VHDL consisting of the LMS filter
component, data read and write components and clock generation components. The test-
bench supports timing and control of clock generation, data input/output and LMS filter
execution times. Figure 4.4 illustrates the platform and test-bench environment designed.
Figure 4.4 – Platform and Test Bench Environment

LMS Mean Square Test Bench

Least Mean Square

Lms.vhdl

Calculate Delta

calc_delta.vhdl

Adder/ Subtractor

add_sub.vhdl

Fourty-Eight Bit Multiplier

mutl48.vhdl

LMS

State Machine

lms_sm.vhdl

Sixteen Bit Multiplexer

mux_16.vhdl

Sixteen Bit

Multiplier

mutl16.vhdl

Fourty-Eight Bit

Register

reg_48.vhdl

Fourty-Eight Bit

Register

reg_48.vhdl

Sampled Ring Laser
Gyro Data

Θ(n)
File

Sampled Dither Pick
Off Data

α(n)
File

Raw Data Record:
Inertial Measurement
System

Θx, Θy, Θz, αx, αy, αz

Personal Computer
Windows Based

Automated System Control
Scripts

Digital Simulation:
Model Technology
VSIM

16 MHz

1600 Hz

Read Data

read_data.vhdl

Write Data

write_data.vhdl

Stripped Ring Laser
Gyro Data

ΔΘ(n), h0(n), h1(n)

Fraction File

Stripped Ring Laser
Gyro Data

ΔΘ(n), h0(n), h1(n)

Integer File

Analysis: MathCAD
PSD, CPSD
Scripts

Personal Computer
Windows Based

Unix Based
Workstation

1600 Hz

16 MHz

Analysis: MathCAD
PSD, CPSD
Scripts

Personal Computer
Windows Based

43
4.11 VHDL Architecture

The VHDL was written such that the entire design could be targeted for synthesis without the use of PROM, RAM or Intellectual Property (IP) super cells. The design was formulated taking advantage of the Actel FPGA architecture in mind. The Actel FPGA architecture lends itself to highly multiplexed designs. In order to reduce the arithmetic entity instantiations, the design implements a multiplex data path scheme resulting in a single multiply, a single accumulate and two differencing functions. This approach was taken with the knowledge that the sample period is relatively slow in comparison to the high-speed operation of the ACTEL FPGA. Higher sampling periods may preclude this option.
Figure 4.5 – Top Level Data Flow /Schematic Diagram of VHDL LMS Algorithm

\[ X(n) = \sum_{m=0}^{M} h_m(n)y(n-m) \]

\[ e(n) = X(n) - \hat{X}(n) \]

\[ h_0(n+1) = h_0(n) + 2\mu e(n)y(n) \]

\[ h_1(n+1) = h_1(n) + 2\mu e(n)y(n-1) \]
The VHDL consists of the top level, lms.vhd, source code instantiating eight VHDL sub-components: add_sub.vhd, calc_delta.vhd, mult_48.vhd, reg_16.vhd, reg_48.vhd, mux_16.vhd, mux_48.vhd and lms_sm.vhd. The lms.vhd is responsible for mapping the sub-components and sign extension of incoming data. Reference Figure 4.5 for a pseudo block/schematic diagram of the lms.vhd component and data flow structure. The calc_delta.vhd code is responsible for taking the delta of the incoming data. Previous work in adaptive filter algorithms reflects that noise reduction techniques using deltas on incoming data gives better random walk performance in Ring Laser Gyro based systems. The reg_16.vhd and reg_48.vhd are VHDL representations of sixteen and forty-eight bit registers respectively. Temporary register storage of iterative data variable values is imperative for correct algorithmic operation.

The mux_16.vhd and mux_48.vhd are VHDL representations of sixteen bit and 48 bit digital multiplexers respectively. The add_sub.vhd is a 48 bit signed adder/subtractor. The mult_48_slv.vhd is a forty-eight bit standard logic vector multiplier implemented using a common “shift and add” multiplication algorithm. The lms_sm.vhd is the controller for the multiplier, adder/subtractor, multiplexer, register and data path control. The lms_sm.vhd is implemented as a 32 state sequence machine of which twenty-five states are needed. The control signals illustrated in Figure 4.5 are outputs of the LMS state machine. The lms_sm.vhd is the controller for the multiplier, adder/subtractor, multiplexer, register and data path control. Figure 4.6 illustrates lms_sm.vhd state diagram, as it would appear using only the twenty-five states. Top level VHDL code is listed in Appendix D.
Figure 4.6 – LMS State Machine State Sequence Diagram

If Reset Active
next State = S0
Else
RTI Enabled?

State-S0

State-S1
Register \( \Delta \Theta(n) \& \Delta \alpha(n) \)

State-S2
Add \( \Delta \Theta(n) \) to ALU

State-S3
Multiply \( h_0(n) \& \Delta \alpha(n) \)

State-S4
Multiply Complete Yet?

State-S5
Subtract \( h_0(n) \& \Delta \alpha(n) \) From ALU

State-S6
Multiply \( h_1(n) \& \Delta \alpha(n-1) \)

State-S7
Multiply Complete Yet?

State-S8
Subtract \( h_1(n) \& \Delta \alpha(n-1) \) From ALU

State-S9
Register \( \Delta \Theta_s(n) \)

State-S10
Reset ALU

State-S11
Add \( h_0(n) \) to ALU

State-S12
Multiply \( 2\mu \star \Delta \Theta_s(n) \)

State-S13
Multiply Complete Yet?

State-S14
Register \( 2\mu \star \Delta \Theta_s(n) \)

State-S15
Multiply \( 2\mu \star \Delta \Theta_s(n) \& \Delta \alpha(n) \)

State-S16
Multiply Complete Yet?

State-S17
Add \( 2\mu \star \Delta \Theta_s(n) \) to ALU

State-S18
Register \( h_0(n+1) \)

State-S19
Reset ALU

State-S20
Add \( h_1(n) \) to ALU

State-S21
Multiply \( 2\mu \star \Delta \Theta_s(n) \& \Delta \alpha(n-1) \)

State-S22
Multiply Complete Yet?

State-S23
Add \( 2\mu \star \Delta \Theta_s(n) \) to ALU

State-S24
Register \( h_1(n+1) \)

State-S25
Reset ALU
4.12 Scaling

Scaling was selected as <48,28> using a total of forty-eight bits: giving twenty bits of integer and twenty-eight bits of fraction. The use of twenty-eight bits of fraction was required in order to represent the optimal value of the LMS algorithm’s convergence factor, \( \mu \). For this application eigenvalue analysis reflected this value in the neighborhood of 7.68 e-9. Twenty eight bits provides for a convergence factor at approximately half the selected value or \( 1/2^{28} \) (3.73 e-9) although a value closest to the selected value was used. The decision to go with twenty bits of integer was based on providing orders of magnitude scaling above the sixteen bit RLG and reference dither pickoff data being read into the algorithm.

4.13 Results and Analysis

Each VHDL component was individually tested and simulated at the component level to insure accuracy for positive and negative values using a digital simulator and a variety of interim/ altered test-benches. Overflow detection is available in both the multiplier and adder/subtractor but is not currently reported or used. Scaling is such that no overflow conditions were detected in simulations to date. For efficiency of logic, the multiplier output was truncated as opposed to rounded.

Once the routed LMS algorithm VHDL was proven to be operational at the digital simulator level, the stripped output, \( \Delta \theta_s(n) \), and adaptive gains, \( h_0(n) \) and \( h_1(n) \), data were converted from bit vector format to integer format. The non-availability of a conversion utility, converting data from bit vector to a scaled integer format, required splitting the data at the implied decimal point and writing two data files in integer format. This allowed the data to be easily ported into a MathCAD Script. The MathCAD script scaled the fractional portion of the data and then combined the integer and fractional data values into a format that is usable for analysis without losing any accuracy. Appendix A illustrates the MathCAD script. The MathCAD script reads in the raw RLG, dither reference, VHDL output, and MATLAB output data. The raw data, VHDL output and MATLAB output data is plotted. PSDs and CPSDs are calculated and plotted to illustrate the effectiveness of the LMS algorithms.

Initial evaluation of the VHDL algorithm in both pre-route and post route simulations reflects similar results to that seen in the MATLAB algorithm. Pre-route VHDL and post route back annotated FPGA design results were exactly the same. Using similar convergence factors, and consistent window snapshot of the data from the MATLAB algorithm and the VHDL algorithm the results are compared in Table 4.2.
Table 4.2 – MATLAB Versus VHDL Outputs

<table>
<thead>
<tr>
<th></th>
<th>CPSD</th>
<th>h0 gain</th>
<th>h1 gain</th>
<th>(CPSD/2)½</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB</td>
<td>0.35572 arc-sec²</td>
<td>0.21926</td>
<td>0.00718</td>
<td>0.42173 arc-sec</td>
</tr>
<tr>
<td>VHDL</td>
<td>0.34997 arc-sec²</td>
<td>0.21916</td>
<td>0.00705</td>
<td>0.41831 arc-sec</td>
</tr>
<tr>
<td>FPGA</td>
<td>0.34997 arc-sec²</td>
<td>0.21916</td>
<td>0.00705</td>
<td>0.41831 arc-sec</td>
</tr>
<tr>
<td>Percent Deviation</td>
<td>0.81 %</td>
<td>0.04 %</td>
<td>1.81 %</td>
<td>0.81 %</td>
</tr>
</tbody>
</table>

The MATLAB algorithm was used as the more accurate representation of the LMS algorithm performance. The MATLAB uses floating point processing, whereas the VHDL is a fixed point processing algorithm. The original MATLAB algorithm used a 2µ value or 15.2 x 10⁻⁹, whereas the VHDL algorithm used a 2µ value of 14.9011 x 10⁻⁹. Variations in the data between the two can be contributed in large part to the 1.9% difference in the 2µ values. The CPSD of the MATLAB algorithms final magnitude is 0.35572 arc-sec² compared to the VHDL algorithms final magnitude of 0.34997 arc-sec². Reducing the RMS values to the readout noise peak magnitude gives a MATLAB noise magnitude of 0.42173 arc-sec versus the VHDL noise magnitude of 0.4831 arc-sec. This represents less than a 0.81 percent deviation of the VHDL to the MATLAB. Reference the MATHCAD script for deviation calculation. It is also noted that the adaptive gains between the two models closely track each other with convergence of the gains to their optimal approximately 200 real-time (1600 Hz) samples into the filtering process. The higher deviation of the h1 gain is suspected because of the relative magnitudes being processed. Convergence of the adaptive gains can be increased with trade-off in deviation of correlated noise magnitudes.

It should be noted that the 572 Hz incremental rise in the CPSD data reflects a cross coupling of dither energy from an orthogonal gyro within the selected candidate system. The dither reference signal that was fundamental to this algorithms filtering was around 512 Hz. This component of frequency can be seen in the raw data plots in the CPSD plots of the MATHCAD scripts. The CPSD of the filtered data reflects a flattening about the 512 Hz frequency, indicating an effective adaptive filter output.
Chapter Five

Other Considerations

5.1 Finite Word-Length Effects

Representation of digital information is, by definition, finite. All digital systems, at the lowest level, represent data in a binary format. Values represented are therefore limited in dynamic range. Both the maximum and the minimum values of the information are constrained. The unwanted and unfavorable effect of digital representations is often referred to as “finite word-length effects”.

Finite word length effects are the study of an approximation to the true value of information. The approximation contributes non-linear behavior to the value represented. A simple model of the non-linear representation can be formulated as:

\[ \text{Signal}_{\text{nonlinear}}(n) = \text{Signal}_{\text{linear}}(n) + \text{error}(n). \]

Figure 5.1 – Finite Word-Length Effects

Finite Word Length Effects

\[ S_{\text{nonlinear}}(n) = S_{\text{linear}}(n) + \text{Error}(n) \]

Overflow (Large Scale approximation effects)

- Saturation
- Zero saturation wrapping
- Sign magnitude wrapping

Quantization (Small Scale approximation effects)

- Rounding
- Truncation (value truncation)
- Zero Truncation (magnitude truncation)

Uncorrelated (noise errors)

Correlated (small-scale limit cycles overflow)
Figure 5.1 illustrates a top-level view of finite word length effects. In general, finite word length effects can be sectioned into overflow and quantization. Overflow is due to a violation in the large-scale approximation when selecting the most significant bit to represent the information. Quantization is due to a violation in small-scale approximations when selecting the least significant bit to represent the information.

Typical techniques for overcoming overflow are saturation, zero saturation wrapping and sign magnitude wrapping. Typical techniques for overcoming quantization are rounding, truncation, zero truncation.

5.1.1 Overflow

Overflow errors are caused by the result of a mathematical operation result exceeding the maximum allowable dynamic range set by $2^n-1$. Where $n$ is the number of bits chosen to represent the integer part of the value. If there are $n$ bits available to represent the integer part of the value, and the result of the operation requires $n+b$ bits, then the maximum magnitude of the integer error is $2^b - 2^n$. Calculation resulting in overflow often cause cycling between the most positive and most negative values represented by the $n$ bits. This is referred to as large scale overflow oscillations or limit cycling. In most applications, these oscillations can be eliminated by saturation techniques or evaluating the maximum value anticipated and designing the system with ample number of bits to insure against overflow. For addition calculations the number of bits selected to represent the result to guard against overflow would be equal to twice the largest magnitude expected. For multiplication calculations, the number of bits selected to represent the result to guard against overflow would be equal to the square of the largest magnitude expected.

5.1.2 Quantization

Quantization errors are caused by the result of a mathematical operation result being less than the minimum allowable dynamic range set by $1-2^n$. Where $n$ is the number of bits chosen to represent the fractional amount of the value. If there are $n$ bits available to represent the fractional part of the value, and the result requires $n+b$ bits, then the error would be $2^n$. The quantization can be viewed as either correlated or uncorrelated. Correlated quantization effects are classed as noise. Uncorrelated quantization effects sometimes add signals with an undesired frequency content. This can cause small-scale limit cycle oscillations with a small input or even zero input small-scale limit cycles with zero input into the calculation.

5.2 Evaluation of Overflow and Quantization of LMS Algorithm

In the LMS algorithm there are two sources of quantization error to be considered. The first arises from the use of an A/D converter to convert the digital inputs. And the second from the use of finite word-length arithmetic.
The use of an A/D converter for the reference dither pick off signal contributes
the error associated with an assumed uniform step size, δ, and quantizing levels at 0, ± δ,
± 2δ, ± … nδ, where n is the number of bits associated with the A/D conversion output.

The quantizing level can be calculated as (|Fs⁺| + |Fs⁻|)/2^n, where Fs⁺ is the positive
full-scale input voltage range and Fs⁻ is the negative full scale input voltage range.
 Converted values at kT can be represented by kδ - (δ/2) to kδ + (δ/2), where k is a positive
or negative integer and kδ is the quantizer output and T is the inverse of the sampling
frequency rate. This leads to a converted value at the output within the quantizer
uncertainty of δ centered at δ/2. When the dynamic range is not extreme and the number
of bits used are sufficient for a reasonable representation of the data, the contribution
of the quantization noise is shown to be independent of the input signal with zero mean
process. The quantization error, ε, from the analog input to the converted output that is
defined in the range between - (δ/2) ≤ ε ≤ +(δ/2) can be shown to be uniformly
distributed with a variance determined by the step size, δ. The mean square value, which
can be shown to be the variance of the quantization error, is then formulated as:

\[
\overline{\varepsilon^2} = \sigma^2 = \frac{\delta^2}{12} - \frac{1}{\delta} \int_{-\delta/2}^{\delta/2} \varepsilon^2 \, d\varepsilon
\]

(5.1)

The A/D converter used in this application is a 12-bit converter with full scaling
between ±10V giving a quantization, δ, value of 20/2^{12} = 4.8828125x10^{-3} Volts/Bit. The
variance, σ^2, is then calculated to be 1.9868x10^{-6} Volts/Bit. The root mean square or
standard deviation of the error is then:

\[
\varepsilon_{\text{rms}} = \sigma = \frac{\delta}{\sqrt{12}}
\]

(5.2)

which is equal to (20/2^{12})/12^{1/2} = 1.4095x10^{-3} Volts/Bit. The nominal amplitude of the
reference dither pick off signal is ±3Volts. This represents 30 % of the full-scale range of
the A/D converter.

The LMS gyro input data is generated from a scaled output of a digitized analog
signal. The quantization of this information is 1.1125 arc-sec/pulse and is accumulated
over the 1600 Hz sampling period. Like the A/D converter, when the quantization is
granular enough and the signal spectrum is sufficiently broad, the distortion produced by
the quantizing may be modeled as an additive independent source of white noise with
zero mean and variance determined by the mean squared value of the error. For the gyro
then, we calculate a variance of 1.1125^2/12 = 103.1380x10^{-3} and a root mean square or
standard deviation of 321.1511x10^{-3}.
Finite word length errors associated with the LMS algorithm itself can be attributed to the arithmetic calculations of addition or multiplication. There is only one adder and one multiplier. Assuming that sufficient word-length is selected and no overflow will occur during the calculation process there should be no error associated with the addition, however, there will be error associated with the multiplication. Unlike analog quantization, digital quantization due to round off or truncation may result in a non-zero mean error.

There are two issues that require attention in a discrete finite word length implementation of an adaptive filter: numerical stability and numerical accuracy. Numerical stability is required to insure the filter doesn’t diverge and begin overflowing. Numerical accuracy is a function of the number of bits used for any given calculation. Certainly, the larger the number of bits used, the more accurate the calculations and the closer to an analog representation the discrete version of the adaptive filter converges to.

The referenced dither pick off and the gyro input quantization have already been discussed with variance and standard deviations calculated. The discrete LMS filter has quantized tap weights and a quantized output. The tapped weights and the outputs are both calculated using the same multiply and accumulate section of the VHDL. The multiplier is a generic shift and add algorithm that uses only one adder that successively accumulates the partial products into a register. The partial products are shifted to the right leaving the partial product and the multiplicand in their required relative position for further processing. The adder performs both addition and subtraction. This implementation insures that the add never causes an overflow because the addition and subtraction operations alternate and the two numbers being added or subtracted always have opposite signs. This algorithm makes an efficient implementation for behavior VHDL and ultimately a structural layout.

The multiplier is designed as a 48 by 48 bit multiplication, requiring a 96-bit output. The multiplier is designed to work with a \(<48,28>\) input scaling, giving 20 bits of integer and 28 bits of fraction. The output of the multiplier is scaled at \(<96,56>\) giving 40 bits of integer and 56 bits of fraction. This result fed forward to the accumulator truncates both the least significant 28 bits of fraction and the 20 most significant bits of integer leaving 20 bits of integer and 28 bits of fractional representation or a scaling of \(<48,20>\). With 20 bits of integer, 1 sign and 19 magnitude bits, a maximum magnitude value of \(2^{19}\) or 524,288 can be represented. With 28 bits of fraction, a minimum value of \(1/2^{28} = 3.7529098 \times 10^{-9}\) can be represented. Calculations through the multiplier include the \(h0(n) \times \Delta \alpha(n)\), \(h1(n) \times \Delta \alpha(n-1)\), \(2\mu \times \Delta \Theta(s(n))\), \(2\mu \Delta \Theta(s(n)) \times \Delta \alpha(n)\), and \(2\mu \Delta \Theta(s(n)) \times \Delta \alpha(n-1)\).

For each multiplicative case, the scaling is selected such that numerical stability and numerical accuracy is achieved. The delta alphas, \(\Delta \alpha(n)\) and delta thetas, \(\Delta \Theta(n)\), results will always be less than the absolute maximum magnitudes of the alphas, \(\alpha(n)\), which was 3480 and thetas, \(\Theta(n)\) which was 26490. The \(h0\) and \(h1\) absolute magnitude
maximums have been shown to be 0.24076 and 0.07642 respectively. The absolute magnitude maximum of the mean squared error or $\Delta \Theta s(n)$ (referred to as $e(n)$ or $\text{mean}(\epsilon^2(n))$ have been shown to be 778. The convergence factor, $\mu$, for the VHDL algorithm was chosen to be 100 times less or $754.15146570201 \times 10^{-9}$ than the minimum to insure minimum variance of the stripped gyro data. The $2\mu$ value of the VHDL was selected as $7.450580596924E-9$. This value was approximately twice the minimum capable representation given the 48,28 scaling.

5.3 Scaling Optimization

The maximum magnitude of the delta alphas, $\Delta \alpha(n)$ and $\Delta \Theta(n)$ presented to the multiplier are representative of the rate input of the system in a stable test configuration. When considering the requirements on scaling, the system dynamic range must be considered.

The mechanical peak dither rate contribution to the maximum rate induced at the RLG has been shown to be 225 degrees/second. A typical system dynamic rate + noise range is 400 degrees/second. This gives a total input range at the RLG of 625 degrees/second. Given a scale factor of 1.125 arc-seconds per pulse out of the digitizing electronics and a 1600 Hz sampling period, a maximum value for delta theta, $\Delta \Theta(n)$, would be approximately $1.406.25.5$ pulses/sample period. The theta, $\Theta(n)$, pulse accumulation is implemented in a 16-bit roll over counter, therefore the holding register inputs are required to be 16 bits as well. But the $\Delta \Theta(n)$ values out of the differencing logic in the LMS filter could have been 12 bits.

The dither reference input maximum is based on a 6 volt peak to peak sine wave. In this system there is a -2048 count bias on each data sample. Given the 4.8828125 volts/bit scale factor of the A/D, a maximum value of alpha, $\alpha(n)$, less the bias count, is 1,229. Given a minimum dither frequency of 500 Hz sampled at 1600 Hz, the maximum value of delta alpha, $\Delta \alpha(n)$, (less the bias) is 870.

Given the absolute maximum values for each of the parameters discussed in the previous sections, the absolute maximum magnitude values associated with each multiply given the dynamic requirements of the system are:

MAX[$h0(n) \times \Delta \alpha(n)$] = 0.24076 x 870 = 240.76
MAX[$h1(n) \times \Delta \alpha(n-1)$] = 0.07624 x 870 = 66.3288
MAX[$2\mu \times \Delta \Theta s(n)$] = 7.450580596924E-9 X 1462.5 = 10.8964741E-6
MAX[$2\mu \times \Delta \Theta s(n) \times \Delta \alpha(n)$] = 7.450580596924E-9 x 1462.5 x 870 = 4.73996624E-3
MAX[$2\mu \times \Delta \Theta s(n) \times \Delta \alpha(n-1)$] = 7.450580596924E-9 x 1462.5 x 870 = 4.73996624E-3
By calculating the $2\mu \Delta \Theta_s(n)$ before multiplying by $\Delta \alpha(n)$, we limit the large output associated with a 1462.5 x 870 value reflecting a maximum integer value out of the multiplier to be 910,237.5. This would require a 20-bit integer value.

Given the scaling of $<48,20>$, a maximum allowable magnitude of the integer would be $2^{19} = 524,288$ and the fractional component would be $1/2^{28} = 3.725290298 \times 10^{-9}$. The maximum inputs to the multiplier would be required to be 12 bits for the gyro and 12 bits for the dither reference. The largest multiplier magnitude output would require 8 bits. We have therefore shown that there is sufficient dynamic range and quantization in the scaling selection to insure numerical stability and accuracy. The system was never exercised over the maximum dynamic range input so the LMS filter was never exercised to its fullest capability reflecting no overflow conditions.

It is obvious to the investigators that the maximum gain values of $h_0$ and $h_1$ are not sufficiently known for the maximum dynamic range of the system nor is it known what the optimum value of the convergence factor, $\mu$, is under these conditions. For a thorough analysis of the multiplier scaling, the a priori data would be required under a maximum dynamic range. But it is sufficient to say that the current scaling would be sufficed.

The variance and standard deviation of the multiplier output is then calculated to be $(2^{-28})^2/12 = 6.0047 \times 10^{-12}$ and $1.07539 \times 10^{-9}$. These values reflect a very small contribution to the overall LMS algorithm.

The analysis from the Cumulative Power Spectral Density reflects a final noise value difference of less than 1% when comparing the MATLAB model to the routed version of the VHDL MODEL. Based on the calculations under maximum dynamic range of the system, the overall scaling of $<48,28>$ would be adequate. It is also true that the 28 bits of fractional processing could be reduced at the cost of a greater mean square error value at the LMS filter output. Remembering that the convergence factor was selected to be 100 times less than the calculated maximum in order to achieve a smaller mean square error. So a fractional representation of 22 bits could have been used giving an overall scaling of $<42,22>$ or 20 bits of integer and 22 bits of fraction. While the analysis reflects this, a re-write of the VHDL and a test configuration under maximum dynamic range of the system would to prove this fact would be cost and schedule prohibitive.
Chapter Six

Conclusion and Future Research

6.1 Conclusion

The research and design effort met its intended goals. Research and development of diagonal and full matrix real time adaptive digital signal processing algorithms were accomplished. To date, a variety of real time adaptive LMS, RLS and JPGE structures have been researched and developed with some success in identifying candidate structure for low noise and low data latency requirements. Trade off in architecture and organizational complexity was evaluated. An understanding into efficiency and reduction in correlated noise was gained. Error criteria versus convergence of the algorithms were explored. Contribution of this parameter to system level performance has been identified.

Tool suite trade studies have made the researcher more sensitive to selection criteria when considering tool suite maturity and longevity, especially when considering the associated costs and schedule impacts that can occur when having to migrate from one tool suite to another. The VHDL coding and subsequent synthesis, place and route brought a wealth of knowledge to the author when considering the platform system configuration and control, the fundamental design processes needed to be established, and the algorithmic research and development design, validation and verification pre and post layout. Cross tool and platform boundaries were established with success in porting data between a variety of data types and third party tools needed for analysis. Overall, the research was a success.

Synthesis of the VHDL algorithm using the Synplify_Pro synthesis tool was accomplished. Continued analysis with regard to the effects of finite register lengths to convergence, limit cycling, and reduction of correlated noise magnitudes was evaluated. Evaluation of overflow detection and the need for correction was looked at. It is the expressed hopes of the investigators that a VHDL architecture in support of a full matrix LMS implementation for reduction of both fundamental and cross channel noise canceling can be achieved. Possibilities now exist for other adaptive filter algorithms to be researched. The Recursive Least Square algorithm is one such adaptive filter algorithm that exists in block processing configuration using a microprocessor. It may be possible to generate an efficient RLS algorithm in VHDL that will eliminate the need for
a microprocessor-based system, thereby reducing both non-recurring and recurring unit product costs.

6.2 Future Research

For every question asked, numerous questions are formulated. For every data point acquired and analyzed, additional data is needed to further understand the nature of the subject. Lucky for mankind, there are endless questions and endless answers. This subject is no different.

There is opportunity to reduce the correlated noise components even further. The power spectral density plots reveal harmonic components of both main channel and cross channel coupling. The opportunity exists to develop higher order filter structures that will reduce the quadratic (second harmonic) and cubic (third harmonic) components for both the diagonal and full matrix implementations. Algorithms could be researched and developed extending the two gain models to nth gain models. Evaluation of frequency components due to beat frequency generation of the fundamental and their harmonics against each other can be researched. Efficient three channel diagonal and full matrix implementations of each of the filter structures can be developed in VHDL for targeting ASICs. Of particular interest would be research of more concise structures using the full breadth of Systems On Chip, SOC, ASICs. Structures including memory and Multiply and Accumulate, MAC, super cells could be evaluated. There will undoubtedly be continued interest and research that comes from these studies: validating that there is no end to the questions and answers in the area of real time adaptive digital signal processing in Ring Laser Gyro based Inertial Systems.
References


Bibliography


Appendices
Appendix A: Algorithm Figures

Figure A1 - PSD of Correlated Gyro Readout Noise

Figure A2 - Cumulative PSD-Correlated Gyro Readout Noise
Appendix A: (Continued)

Figure A3-PSD of Gyro Dither Pick Off

Noise Reference: A/D Count Value/Hertz

Figure A4-Cumulative PSD of Gyro Dither Pick Off

Noise Reference: A/D Count Value/Hertz
Figure A5 - LMS Performance Surface

Figure A6 - Gyro X,Y,Z LMS Learning Curve
Appendix A: (Continued)

Figure A7-LMS G1 Gain Values

Gain Magnitude

1.6K Hz Samples

G1xx

G1xy

G1xz

G1yx

G1yy

G1yz

G1zx

G1zy

G1zz

Figure A8-LMS G1 Gains

Gain Magnitude

1.6K Hz Samples
Appendix A: (Continued)

Figure A9-LMS G2 Gain Values

Figure A10-LMS G2 Gains
Appendix A: (Continued)

![Figure A11-LMS G1 Gain Versus G2 Gain Curves](image1)

![Figure A12 - Performance Surface Using Actuals Gains and MSE X Channel](image2)
Appendix A: (Continued)

Figure A13-LMS Gains VS MSE

Figure A14-LMS G1 Gains Versus Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A15-LMS PSD-Stripped Delta Theta Uncorrelated Readout Noise

Figure A16-LMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise
Figure A17- Full Matrix LMS G1 Gain Values

Figure A18- Full Matrix LMS G1 Gains
Figure A19-Full Matrix LMS G2 Gain Values

Figure A20-Full Matrix LMS G2 Gains
Appendix A: (Continued)

Figure A21-FMLMS PSD-Stripped Delta Theta Uncorrelated Readout Noise

Figure A22-FMLMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A23-Normalized LMS G1 Gain Values

Figure A24-Normalized LMS G1 Gains
Appendix A: (Continued)

Figure A25-Normalized LMS G2 Gain Values

Figure A26-Normalized LMS G2 Gains
Appendix A: (Continued)

Figure A27-NLMS PSD-Stripped Delta Theta Uncorrelated Readout Noise

Figure A28-NLMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A29 - RLS W0 Weight Values

Figure A30 - RLS W0 Weight Values
Appendix A: (Continued)

Figure A33 - RLS Main Channel Uncorrelated Delta Theta PSD

Figure A34 - RLS Main Channel Uncorrelated Delta Theta CPSD
Appendix A: (Continued)

Figure A35—Full Matrix RLS-W0 Gain Values

Figure A36—Full Matrix RLS W0 Gains
Figure A37-Full Matrix RLS W1 Gain Values

Figure A38-Full Matrix RLS W1 Gains
Appendix A: (Continued)

Figure A39 - Full Matrix RLS PSD-Stripped Delta Theta Uncorrelated Readout Noise

Figure A40 - Full Matrix RLS-CPSD Stripped Delta Theta Uncorrelated Readout Noise
Figure A41 - JPGE Gains

Figure A42 - JPGE Gamma Values
Appendix A: (Continued)

Figure A43 - JPGE-PSD Uncorrelated Gyro Readout Noise $E(2)$

Figure A44 - JPGE-CPSD Uncorrelated Gyro Readout Noise $E(2)$
Appendix A: (Continued)

Figure A45 - JPGE-PSD Uncorrelated Gyro Readout Noise $E(3)$

Figure A46 - JPGE-CPSD Uncorrelated Gyro Readout Noise $E(3)$
Appendix A: (Continued)

Figure A47 - FMJPGE Gains

Figure A48 - FMJPGE Gamma Values

84
Appendix A: (Continued)

Figure A49-FMJPGE-PSD Uncorrelated Gyro Readout Noise $E(2)$

Figure A50-FMJPGE-CPSD Uncorrelated Gyro Readout Noise $E(2)$
Appendix A: (Continued)

Figure A51 - FMJPGE-PSD Uncorrelated Gyro Readout Noise E(3)

Figure A52 - FMJPGE-CPSD Uncorrelated Gyro Readout Noise E(3)
Appendix A: (Continued)

Figure A53 - LMS - Variance Correlated Readout Noise

Figure A54 - LMS - Variance Dither Pick Off
Appendix A: (Continued)

Figure A57 - LMS Variance Uncorrelated Readout Noise

Figure A58 - LMS Covariance Uncorrelated Readout Noise to DPO
Appendix A: (Continued)

Figure A61-FMLMS-Covariance Uncorrelated Readout Noise to DPO

Figure A62-FMLMS-Correlation Coefficient Uncorrelated Readout Noise to DPO
Appendix A: (Continued)

Figure A63 - NLMS-Variance Uncorrelated Readout Noise

Figure A64 - NLMS - Covariance Uncorrelated Readout Noise to Dither Pick Off
Appendix A: (Continued)

Figure A65 - NLMS - Correlation Coefficient Uncorrelated Readout Noise to DPO

Figure A66 - Main Channel RLS-Variance Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A67 - Main Channel RLS-Covariance Uncorrelated Readout Noise to DPO

Figure A68 - Main Channel RLS-Correlation Coefficient Uncorrelated Gyro Readout Noise to DPO
Appendix A: (Continued)

Figure A69 - Full Matrix RLS-Variance Uncorrelated Readout Noise

Figure A70 - Full Matrix RLS-Covariance Uncorrelated Readout Noise to DPO
Appendix A: (Continued)

Figure A71-FMRLS-Correlation Coefficient Uncorrelated Gyro Readout Noise to DPO

Figure A72 - JPGE-Variance Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A73 - JPGE-Covariance Uncorrelated Readout Noise to DPO

Figure A74 - JPGE-Correlation Coefficient Uncorrelated Readout Noise to DPO
Appendix A: (Continued)

Figure A75 - FMJPGE-Variance Uncorrelated Readout Noise

Figure A76 - FMJPGE-Covariance Uncorrelated Readout Noise to DPO
Figure A77 - FMJPGA-Correlation Coefficient Uncorrelated Readout Noise to DPO

Figure A78 - Isolated System LMS PSD-Delta Theta-Correlated Readout Noise

Correlated Gyro Readout Noise: $\text{ARC} \cdot \text{SEC}^2/\text{Hertz}$
Figure A79 - Isolated System LMS CPSD-Delta Theta-Correlated Readout Noise

Figure A80 - Isolated System LMS PSD-Delta Dither Pick Off
Appendix A: (Continued)

Figure A81 - Isolated System LMS CPSD-Delta Dither Pick Off

Figure A82 - Isolated System LMS G1 Gain Values
Appendix A: (Continued)

Figure A83: Isolated System-LMS G1 Gains

Figure A84: Isolated System-LMS G2 Gain Values
Appendix A: (Continued)

Figure A85 - Isolated System LMS G2 Gains

Gain Magnitude

Figure A86 - Isolated System LMS PSD-Stripped Delta Theta Uncorrelated Readout Noise

Uncorrelated Gyro Readout Noise/Hz: ARC-SEC²/Hz
Appendix A: (Continued)

Figure A87 - Isolated System LMS CPD-Stripped Delta Theta Uncorrelated Readout Noise

Figure A88 - Isolated System LMS-Variance Correlated Readout Noise
Appendix A: (Continued)

Figure A89 - Isolated System LMS-Variance Dither Pick Off

Figure A90 - Isolated System LMS-Covariance Correlated Readout Noise to DPO
Appendix A: (Continued)

Figure A91 - Isolated System-Correlation Coefficient Correlated Readout Noise to DPO

Figure A92 - Isolated System LMS-Variance Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A93 - Isolated System LMS-Covariance Uncorrelated Readout Noise to DPO

Figure A94 - Isolated System LMS-Correlation Coefficient Uncorrelated Readout Noise to DPO
Figure A95-Isolated System FMLMS G1 Gain Values

Figure A96-Isolated System FMLMS G1 Gain Values
Figure A97 Isolated System-FMLMS G2 Gain Values

Figure A98-Isolated System FMLMS G2 Gain Values

Appendix A: (Continued)
Appendix A: (Continued)

Figure A99 - Isolated System FMLMS PSD-Stripped Delta Theta Uncorrelated Readout Noise

Figure A100 - Isolated System FMLMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise
Appendix A: (Continued)

Figure A101 - Isolated System FMLMS-Variance Uncorrelated Readout Noise

Figure A102 - Isolated System FMLMS-Covariance Uncorrelated Readout Noise to DPO
Figure A103 - Isolated System FMLMS-Correlation Coefficient Uncorrelated Readout Noise to DPO

Correlation Coefficient Magnitude vs. 50 Point Sliding Window Sample

- GyroX (Red)
- GyroY (Blue)
- GyroZ (Green)
Appendix B: MathCAD Scripts


Read In raw RLG, Q, data and plot PSD and CPSD. Remember the RLG contains base motion + dither reference + uncorrelated noise.

data := READPRN("C:\Documents and Settings\dadoheny\Desktop\MSEE_DATA\plt_hrss1.dat")

X := data
Θraw := data

N := rows(data)  N = 1.00000 × 10^4  i := 1..(N - 2000)  µrad := 10^{-6}\text{rad}  asec := \frac{\text{deg}}{3600}
P := 800  k := 1..P  Θssraw_k := Θraw_{k+400}

F_s := 1600\text{Hz}  ΔT := \frac{1}{F_s}  M := \text{floor}\left(\frac{P}{2}\right) + 1  M = 401.00000

j := 2..M  ΔF := \frac{1}{P\cdotΔT}  F_{j} := (j - 1)\cdotΔF  ΔF = 2.00000\text{Hz}

\text{FFT}_{Θssraw} := \text{cfft}(Θssraw)

S_{Θssraw} := 2\left(\frac{\text{FFT}_{Θssraw}}{F_s}\right)^2

CumS_{Θssraw_{j-1}} := 0  CumS_{Θssraw_{j}} := CumS_{Θssraw_{j-1}} + S_{Θssraw_{j}}\cdotΔF

Figure B1 - PSD Raw Xgyro

Figure B2 - CPSD Raw Xgyro
Appendix B: (Continued)

Read In raw Dither Reference data and plot CPSD and PSD.

\[ \text{xdp}_\text{raw} := \text{READPRN} ("C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\xdpo.dat") \]

\[ \alpha_{\text{raw}} := \text{xdp}_{\text{raw}} \]
\[ N := 3200 \quad P := 800 \quad \mu\text{rad} := 10^{-6}\text{rad} \quad \text{asec} := \frac{\text{deg}}{3600} \]
\[ i := 1..P \quad \alpha_{\text{ssraw}} := \alpha_{\text{raw}} + 400 \]
\[ F_s := 1600\text{Hz} \quad \Delta T := \frac{1}{F_s} \quad M := \text{floor}\left(\frac{P}{2}\right) + 1 \quad M = 401.00000 \]
\[ j := 2..M \quad \Delta F := \frac{1}{P \cdot \Delta T} \quad F_j := (j - 1) \cdot \Delta F \quad \Delta F = 2.0000\text{Hz} \]
\[ \text{FFT}_{\alpha_{\text{ssraw}}} := \text{cfft} (\alpha_{\text{ssraw}}) \quad S_{\alpha_{\text{ssraw}}} := \frac{2 \cdot (|\text{FFT}_{\alpha_{\text{ssraw}}}|)^2}{F_s} \]

\[ \text{CumS}_{\alpha_{\text{ssraw}}} := 0 \quad \text{CumS}_{\alpha_{\text{ssraw}}} := \text{CumS}_{\alpha_{\text{ssraw}}} + S_{\alpha_{\text{ssraw}}} \cdot \Delta F \]

Figure B3 - PSD Raw Xdpo

Figure B4 - CPSD Raw Xdpo
Appendix B: (Continued)

Read in Data processed by VHDL algorithm.

\[
\begin{align*}
    i &:= 1..1600 \\
    h0_{\text{int}} &:= \text{READPRN}('C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\h0_{\text{int.dat}}') \\
    h0_{\text{frac}} &:= \text{READPRN}('C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\h0_{\text{frac.dat}}') \\
    h0_i &:= h0_{\text{int}} + \frac{h0_{\text{frac}}}{2^{28}} \\
    h1_{\text{int}} &:= \text{READPRN}('C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\h1_{\text{int.dat}}') \\
    h1_{\text{frac}} &:= \text{READPRN}('C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\h1_{\text{frac.dat}}') \\
    h1_i &:= h1_{\text{int}} + \frac{h1_{\text{frac}}}{2^{28}} \\
    \Delta\Theta_{\text{int}} &:= \text{READPRN}('C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\en_{\text{int.dat}}') \\
    \Delta\Theta_{\text{frac}} &:= \text{READPRN}('C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\en_{\text{frac.dat}}') \\
    \Delta\Theta_i &:= \Delta\Theta_{\text{int}} + \frac{\Delta\Theta_{\text{frac}}}{2^{28}}
\end{align*}
\]
Appendix B: (Continued)

Plot filtered output and adaptive gains from VHDL algorithm, h0, h1.

Figure B5 - VHDL LMS Output Xgyro

Figure B6 - VHDL Adaptive Gain h0

Figure B7 - VHDL Adaptive Gain h1
Appendix B: (Continued)

Calculate and plot PSD and CPSD on VHDL algorithm data.

\[
N := 3200 \quad P := 800 \quad \mu \text{rad} := 10^{-6} \cdot \text{rad} \quad \text{asec} := \frac{\text{deg}}{3600}
\]

\[
i := 1..P \quad \Delta \Theta_{ss,i} := \Delta \Theta_{i+400}
\]

\[
F_s := 1600 \cdot \text{Hz} \quad \Delta T := \frac{1}{F_s} \quad M := \text{floor}\left(\frac{P}{2}\right) + 1 \quad M = 401.00000
\]

\[
j := 2..M \quad \Delta F := \frac{1}{P \cdot \Delta T} \quad F_j := (j - 1) \cdot \Delta F \quad \Delta F = 2.0000 \cdot \text{Hz}
\]

\[
\text{FFT}_{\Delta \Theta_{ss}} := \text{cfft}(\Delta \Theta_{ss}) \quad S_{\Delta \Theta_{ss}} := 2 \cdot \frac{(|\text{FFT}_{\Delta \Theta_{ss}}|)^2}{F_s}
\]

\[
\text{CumS}_{\Delta \Theta_{ss,1}} := 0 \cdot \text{asec}^2 \quad \text{CumS}_{\Delta \Theta_{ss,j}} := \text{CumS}_{\Delta \Theta_{ss,j-1}} + S_{\Delta \Theta_{ss,j}} \cdot \Delta F
\]

Filtered Frequency is at 512 Hz, 572 Hz Component is cross coupled dither energy from an orthogonally mounted RLG.
Appendix B: (Continued)

Read in LMS Data generated in MATLAB LMS Algorithm.

\[ \Delta \Theta_{\text{matlab}} := \text{READPRN}("C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\strpgymatlab.dat") \]

\[ h_{0, \text{matlab}} := \text{READPRN}("C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\g1matlab.dat") \]

\[ h_{1, \text{matlab}} := \text{READPRN}("C:\Documents and Settings\dadoheny\Desktop\MSEE\DATA\g2matlab.dat") \]

Plot filtered output, \( \Delta \Theta \), and adaptive gains from VHDL algorithm, \( h_0 \), \( h_1 \).
Appendix B: (Continued)

Calculate and plot PSD and CPSD on MATLAB algorithm data.

\[ N := 3200 \quad P := 800 \quad \mu\text{rad} := 10^{-6}\text{rad} \quad \text{asec} := \frac{\text{deg}}{3600} \]

\[ i := 1..P \quad \Delta \Theta_{\text{ssmatlab}} := \Delta \Theta_{\text{matlab}}_{i+400} \]

\[ F_s := 1600\text{Hz} \quad \Delta T := \frac{1}{F_s} \quad M := \text{floor}\left(\frac{P}{2}\right) + 1 \quad M = 401.00000 \]

\[ j := 2..M \quad \Delta F := \frac{1}{P \cdot \Delta T} \quad F_j := (j - 1) \cdot \Delta F \quad \Delta F = 2.0000\text{Hz} \]

\[ \text{FFT}_{\Delta \Theta_{\text{ssmatlab}}} := \text{cfft}\left(\Delta \Theta_{\text{ssmatlab}}\right) \quad S_{\Delta \Theta_{\text{ssmatlab}}} := 2 \cdot \left(\frac{\text{FFT}_{\Delta \Theta_{\text{ssmatlab}}}}{F_s}\right)^2 \]

\[ \text{CumS}_{\Delta \Theta_{\text{ssmatlab}}} := 0 \quad \text{CumS}_{\Delta \Theta_{\text{ssmatlab}}} := \text{CumS}_{\Delta \Theta_{\text{ssmatlab}}} + S_{\Delta \Theta_{\text{ssmatlab}}} \cdot \Delta F \]

Figure B13 - MATLAB PSD Filtered Xgyro

Figure B14 - MATLAB CPSD Filtered Xgyro
Appendix B: (Continued)

Determine deviation in Noise Magnitude and Adaptive gains between MATLAB and VHDL algorithms.

Calculate deviation Noise Magnitude:

\[
\text{CPSD}_{\text{matlab}} := 0.35572 \quad \text{CPSD}_{\text{vhdl}} := 0.34997
\]

\[
\text{NoiseMag}_{\text{matlab}} := \sqrt{\frac{\text{CPSD}_{\text{matlab}}}{2}} \quad \text{NoiseMag}_{\text{vhdl}} := \sqrt{\frac{\text{CPSD}_{\text{vhdl}}}{2}}
\]

\[
\text{NoiseMag}_{\text{matlab}} = 0.42173
\quad \text{NoiseMag}_{\text{vhdl}} = 0.41831
\]

\[
\%\text{dev}_{\text{noisemag}} := \left| \frac{\text{NoiseMag}_{\text{matlab}} - \text{NoiseMag}_{\text{vhdl}}}{\text{NoiseMag}_{\text{matlab}}} \right| \cdot 100
\]

\[
\%\text{dev}_{\text{noisemag}} = 0.81151
\]

Calculate deviation of adaptive h0 gain:

\[
\text{h0}_{\text{matlab}} := 0.21926 \quad \text{h0}_{\text{vhdl}} := 0.21916
\]

\[
\%\text{dev}_{\text{h0} \text{gain}} := \left| \frac{\text{h0}_{\text{matlab}} - \text{h0}_{\text{vhdl}}}{\text{h0}_{\text{matlab}}} \right| \cdot 100
\]

\[
\%\text{dev}_{\text{h0} \text{gain}} = 0.04561
\]

Calculate deviation of adaptive h1 gain:

\[
\text{h1}_{\text{matlab}} := 0.00718 \quad \text{h1}_{\text{vhdl}} := 0.00705
\]

\[
\%\text{dev}_{\text{h1} \text{gain}} := \left| \frac{\text{h1}_{\text{matlab}} - \text{h1}_{\text{vhdl}}}{\text{h1}_{\text{matlab}}} \right| \cdot 100
\]

\[
\%\text{dev}_{\text{h1} \text{gain}} = 1.81058
\]
Analysis of the stripped gyro data.
Mean and standard deviation of first set of 400 data points.
Reference plot VHDL - Real Time LMS Output - Xgyro.
\[ n := 400 \quad j := 0..n \quad \Delta \Theta_{anal,j} := \Delta \Theta_j \]
\[ \nu \Delta \Theta := \text{mean}(\Delta \Theta_{anal}) \quad \nu \Delta \Theta = -65.21803 \]
\[ \text{stdev} \Delta \Theta := \text{stdev}(\Delta \Theta_{anal}) \quad \text{stdev} \Delta \Theta = 1.29438 \times 10^3 \]
\[ \text{stdev} \Delta \Theta_{unbiased} := \text{stdev}(\Delta \Theta_{anal}) \cdot \frac{n}{n-1} \quad \text{stdev} \Delta \Theta_{unbiased} = 1.29600 \times 10^3 \]
Mean and standard deviation of second set of 400 data points.
\[ n := 400 \quad j := 0..n \quad \Delta \Theta_{anal,j} := \Delta \Theta_{j+400} \]
\[ \nu \Delta \Theta := \text{mean}(\Delta \Theta_{anal}) \quad \nu \Delta \Theta = -0.01101 \]
\[ \text{stdev} \Delta \Theta := \text{stdev}(\Delta \Theta_{anal}) \quad \text{stdev} \Delta \Theta = 0.57594 \]
\[ \text{stdev} \Delta \Theta_{unbiased} := \text{stdev}(\Delta \Theta_{anal}) \cdot \frac{n}{n-1} \quad \text{stdev} \Delta \Theta_{unbiased} = 0.57666 \]
Mean and standard deviation of third set of 400 data points.
\[ n := 400 \quad j := 0..n \quad \Delta \Theta_{anal,j} := \Delta \Theta_{j+800} \]
\[ \nu \Delta \Theta := \text{mean}(\Delta \Theta_{anal}) \quad \nu \Delta \Theta = -0.01046 \]
\[ \text{stdev} \Delta \Theta := \text{stdev}(\Delta \Theta_{anal}) \quad \text{stdev} \Delta \Theta = 0.60578 \]
\[ \text{stdev} \Delta \Theta_{unbiased} := \text{stdev}(\Delta \Theta_{anal}) \cdot \frac{n}{n-1} \quad \text{stdev} \Delta \Theta_{unbiased} = 0.57666 \]
Mean and standard deviation the gain weight values data points post convergence.
\[ n := 400 \quad j := 0..n \quad g_{0,j} := h_{0,j+800} \quad g_{1,j} := h_{1,j+800} \]
\[ \nu g_0 := \text{mean}(g_0) \quad \nu g_0 = 0.21918 \]
\[ \text{stdev} g_0 := \text{stdev}(g_0) \quad \text{stdev} g_0 = 3.60951 \times 10^{-5} \]
\[ \text{stdev} g_{0,unbiased} := \text{stdev}(g_0) \cdot \frac{n}{n-1} \quad \text{stdev} g_{0,unbiased} = 3.61403 \times 10^{-5} \]
\[ \nu g_1 := \text{mean}(g_0) \quad \nu g_1 = 0.21918 \]
\[ \text{stdev} g_1 := \text{stdev}(g_1) \quad \text{stdev} g_1 = 4.45001 \times 10^{-5} \]
\[ \text{stdev} g_{1,unbiased} := \text{stdev}(g_1) \cdot \frac{n}{n-1} \quad \text{stdev} g_{1,unbiased} = 4.45558 \times 10^{-5} \]
Appendix B: (Continued)

Correlation Coefficient of the first 400 data points.

\[
\begin{align*}
\text{n} &:= 400 & \text{j} &:= 0..n \\
\Theta_{\text{anal}}_j &:= \Theta_j & \Delta\Theta_{\text{anal}}_j &:= \Delta\Theta_j \\
\text{corr}\Theta\text{and}\Delta\Theta &:= \text{corr}\left(\Theta_{\text{anal}}, \Delta\Theta_{\text{anal}}\right) & \text{corr}\Theta\text{and}\Delta\Theta &= 0.06271 \\
\text{corr}\Theta\text{and}\Delta\Theta\text{unbiased} &:= \text{corr}\left(\Theta_{\text{anal}}, \Delta\Theta_{\text{anal}}\right) : \left(\frac{n}{n-1}\right) & \text{corr}\Theta\text{and}\Delta\Theta\text{unbiased} &= 0.06279
\end{align*}
\]

Correlation Coefficient of the second set of 400 data points.

\[
\begin{align*}
\text{n} &:= 400 & \text{j} &:= 0..n \\
\Theta_{\text{anal}}_j &:= \Theta_{j+400} & \Delta\Theta_{\text{anal}}_j &:= \Delta\Theta_{j+400} \\
\text{corr}\Theta\text{and}\Delta\Theta &:= \text{corr}\left(\Theta_{\text{anal}}, \Delta\Theta_{\text{anal}}\right) & \text{corr}\Theta\text{and}\Delta\Theta &= -8.71603 \times 10^{-4} \\
\text{corr}\Theta\text{and}\Delta\Theta\text{unbiased} &:= \text{corr}\left(\Theta_{\text{anal}}, \Delta\Theta_{\text{anal}}\right) : \left(\frac{n}{n-1}\right) & \text{corr}\Theta\text{and}\Delta\Theta\text{unbiased} &= -8.72695 \times 10^{-4}
\end{align*}
\]

Correlation Coefficient of the third set of 400 data points.

\[
\begin{align*}
\text{n} &:= 400 & \text{j} &:= 0..n \\
\Theta_{\text{anal}}_j &:= \Theta_{j+800} & \Delta\Theta_{\text{anal}}_j &:= \Delta\Theta_{j+800} \\
\text{corr}\Theta\text{and}\Delta\Theta &:= \text{corr}\left(\Theta_{\text{anal}}, \Delta\Theta_{\text{anal}}\right) & \text{corr}\Theta\text{and}\Delta\Theta &= 4.42111 \times 10^{-3} \\
\text{corr}\Theta\text{and}\Delta\Theta\text{unbiased} &:= \text{corr}\left(\Theta_{\text{anal}}, \Delta\Theta_{\text{anal}}\right) : \left(\frac{n}{n-1}\right) & \text{corr}\Theta\text{and}\Delta\Theta\text{unbiased} &= 4.42664 \times 10^{-3}
\end{align*}
\]
Appendix B: (Continued)

Covariance of the first 400 data points.

\[ n := 400 \quad j := 0..n \]

\[ \Theta_{\text{anal}} j := \Theta j \quad \Delta \Theta_{\text{anal}} j := \Delta \Theta j \]

\[ \text{covar} \Theta \text{and} \Delta \Theta := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \]

\[ \text{covar} \Theta \text{and} \Delta \Theta := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \sqrt{\frac{n}{n - 1}} \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \sqrt{\frac{n}{n - 1}} \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.55970 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.55900 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta = 1.69398 \times 10^4 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta = 1.69610 \times 10^4 \]

Covariance of the second set of 400 data points.

\[ n := 400 \quad j := 0..n \]

\[ \Theta_{\text{anal}} j := \Theta j + 400 \quad \Delta \Theta_{\text{anal}} j := \Delta \Theta j + 400 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \]

\[ \text{covar} \Theta \text{and} \Delta \Theta := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \sqrt{\frac{n}{n - 1}} \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \sqrt{\frac{n}{n - 1}} \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.10438 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.10424 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta = -0.10424 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = -0.10438 \]

Covariance of the third set of 400 data points.

\[ n := 400 \quad j := 0..n \]

\[ \Theta_{\text{anal}} j := \Theta j + 800 \quad \Delta \Theta_{\text{anal}} j := \Delta \Theta j + 800 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \]

\[ \text{covar} \Theta \text{and} \Delta \Theta := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \sqrt{\frac{n}{n - 1}} \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} := \text{cvar}(\Theta_{\text{anal}}, \Delta \Theta_{\text{anal}}) \sqrt{\frac{n}{n - 1}} \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.10438 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.55900 \]

\[ \text{covar} \Theta \text{and} \Delta \Theta \text{unbiased} = 0.55970 \]
Mean and standard deviation of first set of 400 data points.

\[ n := 400 \quad j := 0..n \quad \Delta \Theta_{\text{anal}} \] := \Delta \Theta_j \]

\[ \nu \Delta \Theta := \text{mean}(\Delta \Theta_{\text{anal}}) \quad \nu \Delta \Theta = -65.21803 \]

\[ \text{stdev} \Delta \Theta := \text{stdev}(\Delta \Theta_{\text{anal}}) \quad \text{stdev} \Delta \Theta = 1.29438 \times 10^3 \]

\[ \text{stdev} \Delta \Theta_{\text{unbiased}} := \text{stdev}(\Delta \Theta_{\text{anal}}) \cdot \sqrt{\frac{n}{n-1}} \quad \text{stdev} \Delta \Theta_{\text{unbiased}} = 1.29600 \times 10^3 \]

Mean and standard deviation of second set of 400 data points.

\[ n := 400 \quad j := 0..n \quad \Delta \Theta_{\text{anal}} \] := \Delta \Theta_{j+400} \]

\[ \nu \Delta \Theta := \text{mean}(\Delta \Theta_{\text{anal}}) \quad \nu \Delta \Theta = 0.01101 \]

\[ \text{stdev} \Delta \Theta := \text{stdev}(\Delta \Theta_{\text{anal}}) \quad \text{stdev} \Delta \Theta = 0.57594 \]

\[ \text{stdev} \Delta \Theta_{\text{unbiased}} := \text{stdev}(\Delta \Theta_{\text{anal}}) \cdot \sqrt{\frac{n}{n-1}} \quad \text{stdev} \Delta \Theta_{\text{unbiased}} = 0.57666 \]

Mean and standard deviation of third set of 400 data points.

\[ n := 400 \quad j := 0..n \quad \Delta \Theta_{\text{anal}} \] := \Delta \Theta_{j+800} \]

\[ \nu \Delta \Theta := \text{mean}(\Delta \Theta_{\text{anal}}) \quad \nu \Delta \Theta = -0.01046 \]

\[ \text{stdev} \Delta \Theta := \text{stdev}(\Delta \Theta_{\text{anal}}) \quad \text{stdev} \Delta \Theta = 0.60578 \]

\[ \text{stdev} \Delta \Theta_{\text{unbiased}} := \text{stdev}(\Delta \Theta_{\text{anal}}) \cdot \sqrt{\frac{n}{n-1}} \quad \text{stdev} \Delta \Theta_{\text{unbiased}} = 0.57666 \]

Mean and standard deviation the gain weight values data points post convergence.

\[ n := 400 \quad j := 0..n \quad g_0^j := h_0^j+800 \quad g_1^j := h_1^j+800 \]

\[ \nu g_0 := \text{mean}(g_0) \quad \nu g_0 = 0.21918 \]

\[ \text{stdev} g_0 := \text{stdev}(g_0) \quad \text{stdev} g_0 = 3.60951 \times 10^{-5} \]

\[ \text{stdev} g_0_{\text{unbiased}} := \text{stdev}(g_0) \cdot \sqrt{\frac{n}{n-1}} \quad \text{stdev} g_0_{\text{unbiased}} = 3.61403 \times 10^{-5} \]

\[ \nu g_1 := \text{mean}(g_1) \quad \nu g_1 = 0.21918 \]

\[ \text{stdev} g_1 := \text{stdev}(g_1) \quad \text{stdev} g_1 = 4.45001 \times 10^{-5} \]

\[ \text{stdev} g_1_{\text{unbiased}} := \text{stdev}(g_1) \cdot \sqrt{\frac{n}{n-1}} \quad \text{stdev} g_1_{\text{unbiased}} = 4.45558 \times 10^{-5} \]
Appendix B: (Continued)

MathCAD 2003 Script: lms_eig_calc.mcd

The following derivation reflects calculations associated with a Least Mean Square algorithm. Data captured from an actual Inertial Measurement System is read in and calculation are made.

The script is fairly self documenting and is meant to reflect calculations in pursuit of the minimum mean square error output and maximum convergence factor required in targeting the algorithm for a VHDL implementation.

Read Data from Data File: raw Ring Laser Gyro and Dither Pickoff Data

data := READPRN("C:\Documents and Settings\dadoheny\Desktop\MSEE_DATA\plt_hrss1.dat")

Define index:

\[ N := \text{rows(data)} \quad N = 10000 \quad i := 1..N - 1 \]

Extract X-axis sample number, gyro angle and dither pickoff data:

\[ X := \text{data}^\langle 1 \rangle \quad GX := \text{data}^\langle 5 \rangle \quad DPOX := \text{data}^\langle 8 \rangle \]

Determine maximum values of gyro, delta gyro, dither pickoff and delta dither pick off:

\[ \max \left| \overrightarrow{GX} \right| = 26290 \quad \Delta GX_i := GX_i - GX_{i-1} \quad \max \left| \Delta GX \right| = 526 \]

\[ \max \left| \overrightarrow{DPOX} \right| = 3480 \quad \Delta DPOX_i := DPOX_i - DPOX_{i-1} \quad \max \left| \Delta DPOX \right| = 2429 \]

Determine slope and intercept for detrend of gyro data:

\[ b := \text{intercept}(X, GX) \quad b = -25946.755814 \]

\[ m := \text{slope}(X, GX) \quad m = -0.003847 \]

Find the mean value of the X-axis gyro data:

\[ \text{msGX} := \left[ \sum_i \left| GX_i - \left( m \cdot X_i + b \right) \right|^2 \right] \cdot \frac{1}{N - 1} \quad \text{msGX} = 43384.242661 \]
Appendix B: (Continued)

Find mean_squared value of X-axis pickoff data:

\[ \text{msDPOX} := \left[ \sum_i \left( \text{DPOX}_i + 2048 \right)^2 \right] \cdot \frac{1}{N-1} \]

\[ \text{msDPOX} = 927882.20152 \]

Find the average value of X-axis pickoff sample \( k \) versus sample \( \left( k-1 \right) \) i.e. cross-correlation:

\[ \text{ccDPOX} := \left[ \sum_i \left( \text{DPOX}_i + 2048 \right) \cdot \left( \text{DPOX}_{i-1} + 2048 \right) \right] \cdot \frac{1}{N-1} \]

\[ \text{ccDPOX} = -398111.409641 \]

Find the average value of X-axis detrended gyro data versus current pickoff sample:

\[ \text{ccGPOX} := \left[ \sum_i \left( \text{GX}_i - \left( \text{m} \cdot \text{X}_i + \text{b} \right) \right) \cdot \left( \text{DPOX}_i + 2048 \right) \right] \cdot \frac{1}{N-1} \]

\[ \text{ccGPOX} = 200549.444027 \]

Find average value of X-axis detrended gyro data versus previous pickoff sample:

\[ \text{ccGPOXP} := \left[ \sum_i \left( \text{GX}_i - \left( \text{m} \cdot \text{X}_i + \text{b} \right) \right) \cdot \left( \text{DPOX}_{i-1} + 2048 \right) \right] \cdot \frac{1}{N-1} \]

\[ \text{ccGPOXP} = -80735.006997 \]

From the above calculations, the input correlation matrix, \( R \), and the Cross Correlation matrix, \( P \), can be constructed:

\[ R := \begin{pmatrix} \text{msDPOX} & \text{ccDPOX} \\ \text{ccDPOX} & \text{msDPOX} \end{pmatrix} \]

\[ P := \begin{pmatrix} \text{ccGPOX} \\ \text{ccGPOXP} \end{pmatrix} \]

\[ R = \begin{pmatrix} 927882.20152 & -398111.409641 \\ -398111.409641 & 927882.20152 \end{pmatrix} \]

\[ P = \begin{pmatrix} 200549.444027 \\ -80735.006997 \end{pmatrix} \]

The expected value of the "desired" signal squared, \( E[d^2] \), is already represented above by \( \text{msGX} \).

The equation for the mean-squared error (MSE), as a function of the gains is:

\[ \text{MSE} = E[d^2] + \text{G}^T \text{RG} - 2 \text{P}^T \text{G} \]

Expanding this and inserting the above calculations gives:

\[ f(g_0,g_1) := R_{0,0} \cdot g_0^2 + R_{0,1} \cdot g_1^2 + 2 \cdot R_{0,1} \cdot g_0 \cdot g_1 - 2 \cdot P_0 \cdot g_0 - 2 \cdot P_1 \cdot g_1 + \text{msGX} \]
From this data, a 3-dimensional plot can be constructed.
Define the number of divisions in the g0 and g1 directions.

\[ \text{Ng0} := 10 \quad \text{Ng1} := 10 \]

Define the upper and lower limits for g0 and g1:

\[ \text{Ug0} := 10 \quad \text{Ug1} := 10 \]
\[ \text{Lg0} := 0.0001 \quad \text{Lg1} := 0.0001 \]

Generate an array of mean-squared error values as a function of g0 and g1:

\[ i := 0..\text{Ng0} \quad j := 0..\text{Ng1} \]

\[ \text{Dg0} := \frac{1}{\text{Ng0}} \cdot (\text{Ug0} - \text{Lg0}) \quad \text{Dg1} := \frac{1}{\text{Ng1}} \cdot (\text{Ug1} - \text{Lg1}) \]

\[ S_{\text{Ng1-j},i} := f(\text{Lg0} + i \cdot \text{Dg0}, \text{Lg1} + j \cdot \text{Dg1}) \]
Appendix B: (Continued)

The minimum mean-square error is found by taking partial derivatives of Equation (1) first with respect to \( g_0 \) and then with respect to \( g_1 \) and setting each equation equal to zero. Solving the system of equations for \( g_0 \) and \( g_1 \) gives the pair of gains that yield the minimum of the performance surface:

\[
g_{0\text{ min}} := \frac{\left(R_{0,0} \cdot R_{0,1} \cdot P_1\right) - \left(R_{0,0}\right)^2 \cdot P_0}{R_{0,0} \cdot \left(R_{0,1} - R_{0,0} \right) \cdot \left(R_{0,1} + R_{0,0}\right)} \quad \text{g}_{0\text{ min}} = 0.2191469
\]

\[
g_{1\text{ min}} := \frac{R_{0,1} \cdot P_0 - R_{0,0} \cdot P_1}{\left(R_{0,1}\right)^2 - \left(R_{0,0}\right)^2} \quad \text{g}_{1\text{ min}} = 0.0070159
\]

Inserting the minimum gain values into Equation (2) gives the minimum mean_squared error:

\[
\text{MSE}_{\text{min}} := f\left(g_{0\text{ min}}, g_{1\text{ min}}\right) \quad \text{MSE}_{\text{min}} = 0.872778
\]

The two gains calculated above \( g_{0\text{ min}} \) and \( g_{1\text{ min}} \) agree with the are consistent with the independent calculations using the MATLAB algorithms. The MATLAB algorithms calculate a \( g_{0\text{ min}} \) value of 0.21914928 and a \( g_{1\text{ min}} \) value of 0.0070046 and a mean-squared error minimum of 0.7508333.

Alternatively, the minimum mean-square error may be found by taking the gradient of Equation (1) and setting the results equal to zero. The result, in matrix form is:

\[
2RG-2P=0
\]

Substituting the optimal value of the gains and solving gives:

\[
G_{\text{Opt}}=R^{-1}P
\]

Substituting this int Equation (1):

\[
\text{MSE}_{\text{min}}=E[d^2]+(G_{\text{Opt}})^T R (G_{\text{Opt}})-2P^T G_{\text{Opt}}
\]

\[
\text{MSE}_{\text{min}}=E[d^2]+(R^{-1}P)^T R (R^{-1}P)-2P^T (R^{-1}P)
\]

Taking advantage of correlation matrix symmetry:

\[
\text{MSE}_{\text{min}}=E[d^2] - P^T R^{-1}P = E[d^2] - P^T (G_{\text{Opt}})
\]

\[
G_{\text{Opt}} := R^{-1} \cdot P \quad G_{\text{Opt}} = \begin{pmatrix} 0.219147 \\ 0.007016 \end{pmatrix}
\]

\[
\text{MSE}_{\text{min}} := \text{msGX} - P^T \cdot G_{\text{Opt}} \quad \text{MSE}_{\text{min}} = 0.872778
\]
Appendix B: (Continued)

From this we can calculate the eigenvalues of the correlation matrix and subsequently the optimal convergence factor.

\[
\lambda := \text{eigvals} (R) \\
\lambda = \begin{pmatrix}
1325993.611161 \\
529770.791879
\end{pmatrix}
\]

Or by taking the maximum value of the square root of the eigenvalues or \(R^T R\) we get the max eigen_value:

\[
\lambda_{\text{max}} := \sqrt{\text{max(eigvals}} (\begin{pmatrix} R^T \\ R \end{pmatrix})} \\
\lambda_{\text{max}} = 1325993.611161
\]

It can be shown that the LMS algorithm converges when \(0 < m < 1/\lambda_{\text{max}}\):

\[
\mu_{\text{max}} := \frac{1}{\lambda_{\text{max}}} \\
\mu_{\text{max}} = 754.15146570201 \times 10^{-9}
\]

In order to represent this maximum value to insure convergence, the following calculation reflects the number of bits required to represent this in a finite register implementation based on \(2^n=1/\mu_{\text{max}}\)

\[
\text{Bits}_{\mu_{\text{max}}} := \log \left( \frac{1}{\mu_{\text{max}}} \right) \log(2) \\
\text{Bits}_{\mu_{\text{max}}} = 20.338642
\]

Twenty-one bits are required to meet the requirement of the \(\mu_{\text{max}}\) value. However their is a trade off between convergence and the variance of the mean square error. In the VHDL algorithm a suitable selection was made to insure convergence and minimization of the mean square error. Because the algorithm actually uses a \(2\mu\) value, a number approximately equal to the \(\mu_{\text{max}}\) divided by a factor of 100 was used and then divided by two giving:

\[
\mu_{\text{vhdl}} := \frac{\mu_{\text{max}}}{100 \cdot 2} \\
\mu_{\text{vhdl}} = 3.770757232851 \times 10^{-9}
\]

\[
\text{Bits}_{\mu_{\text{max}} \text{selected}} := \log \left( \frac{1}{\mu_{\text{max}}} \right) \log(2) \\
\text{Bits}_{\mu_{\text{max}} \text{selected}} = 27.982499
\]

This reflects a value using a minimum of 28 bits or \(1/2^{28}\), automatically forcing a scaling sufficient to meet the quantization of \(\mu\) and requiring a 28 bit fractional bit representation.

\[
\mu_{\text{finite}} := \frac{1}{2^{28}} \\
\mu_{\text{finite}} = 3.725290298462 \times 10^{-9}
\]

\[
\mu_{\text{finite \_x2}} := 2 \cdot \mu_{\text{finite}} \\
\mu_{\text{finite \_x2}} = 7.450580596924 \times 10^{-9}
\]
Appendix C: Sample Listing of MATLAB Program

The following is a sample listing of the MATLAB program for the LMS program. Similar programs were generated for the Main Channel and Full Matrix LMS, Normalized LMS, RLS and Joint Process Gradient Lattice.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name:   LMS717d.m
% Purpose:   Real Time Adaptive Least Mean Square (LMS) Algorithm
% Author:   David A. Doheny
% Description:  Three channel LMS algorithm. Supports Diagonal, Main channel and Cross channel LMS algorithm on input signal and a reference signal. Identity matrix multiply of gain update equations allow for zeroing out the cross terms.
% Last Revision Date: January 15, 2004.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clear all previous variables
% Close all previous plots
% Specify display accuracy

clear all;
close all;
format long;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read Data and assign id number
% Format is Defined as follows: 11 columns of data
% Record #, Line #, Striped X, Striped Y, Striped Z, Gyro X, Gyro Y, Gyro Z, Pickoff x, Pickoff y, Pickoff z

fid=fopen('c:\matlab\MSEE\hrss1.dat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% set colors for sreen print
%set(0,'DefaulttextColor','black');
%set(0,'DefaultaxesXColor','black');
%set(0,'DefaultaxesYColor','black');
%set(0,'DefaultaxesZColor','black');
%set(0,'DefaultFigureColor','white');
%set(0,'DefaultaxesColor','white');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Appendix C: (Continued)

% Number of records to read
% Sampling frequency of data
% Number of records to process
% Transpose input
% Clear a and save PC memory
% Extract count index

n2read=8192;
fs=1600;
n2proc=4096;
a=fscanf(fid,'%d',[11,n2read]);
b=a';
clear a;
count=b(1:n2proc,2);

mu=7.6e-9;
muc=7.6e-23;
ident=[1,0,0;0,1,0;0,0,1];

gy=[b(:,6),b(:,7),b(:,8)];
gyx=gy(:,1);
gyy=gy(:,2);
gyz=gy(:,3);

131
from pick off data matrix extract z vector

\[
\begin{align*}
\text{po} &= [b(:,9) + 2048, b(:,10) - 2048, b(:,11) - 2048]; \\
\text{pox} &= \text{po}(:,1); \\
\text{poy} &= \text{po}(:,2); \\
\text{poz} &= \text{po}(:,3);
\end{align*}
\]

Form delta theta and delta pick off values, \([X(n)-X(n-1)]\), preferred data to use, eliminates random walk; dtheta and theta are \([n2read \times 3]\) matrix. Once deltas are formed, extract x, y, and z vectors.

\[
\begin{align*}
\text{dgy}(n,:) &= \text{gy}(n,:) - \text{gy}(n-1,:); \\
\text{dpo}(n,:) &= \text{po}(n,:) - \text{po}(n-1,:);
\end{align*}
\]

For the LMS with Cubic terms, cube each pickoff value. Then reform into matrix

\[
\begin{align*}
\text{dpoxcu} &= \text{dpo}(,1) \times \text{dpo}(,1) \times \text{dpo}(,1); \\
\text{dpoycu} &= \text{dpo}(,2) \times \text{dpo}(,2) \times \text{dpo}(,2); \\
\text{dpoczcu} &= \text{dpo}(,3) \times \text{dpo}(,3) \times \text{dpo}(,3); \\
\text{dpocub} &= [\text{dpoxcu}, \text{dpoycu}, \text{dpoczcu}];
\end{align*}
\]

LMS ALGORITHM

Initialize variables, vectors or matrix to zero
Appendix C: (Continued)

% Current weights (gains or coefficients)
    G1CUR=zeros(3)*1;
    G2CUR=zeros(3)*2;
    CUCUR=zeros(3)*3;

% Previous weights matrix
    G1NP1=zeros(3,3);
    G2NP1=zeros(3,3);
    CUNP1=zeros(3,3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Begin LMS Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for n=4:(n2read);

% Delta theta estimate calculations
% Alias the estimate, x_hapn
% Two weights per channel
% Linear implementation only
    dthtd_est=(G1CUR*dpo(n,:)')+((G2CUR*dpo(n-1,:))');
    dthtdest(n,:)=dthtd_est';                   %Use for no cubic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use for Linear plus a cubic term
%
    dthtd_est=((G1CUR*dpo(n,:)')-(CUCUR*dpocub(n,:)'))+(G2CUR*dpo(n-1,:')');
    dthtdest(n,:)=dthtd_est';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use for Cubic
% Delta theta - delta theta estimate
% Alias the error equation, e(n)
%
    dths_gy=dgy(n,:)-dthtdest(n,:);
    dhtsgy(n,:)=dths_gy;

% High pass filter on weight update to insure stationary process in weight update
% Under rotation, the gains tend to track rates. This insures that they don't
    dhts_hp=-.25*dhtsgy(n,:)'+0.5*dhtsgy(n-1,:)'+-0.25*dhtsgy(n-2,:');

133
Appendix C: (Continued)

dthtshp(n,:)=dths_hp’;

% Calculation first adaptive weight or [gain], w(0) [or g(0) or h(0)]
% Save current weight update
% Convert 3 x3 matrix to 1 X 9 for storage and ease of index when plotting

G1NP1=(G1CUR+2.*mu.*dthtshp(n,:)*dpo(n-1,:)).*ident;

% Use when not filtering the gain update equation through hipass
% G1NP1=G1CUR+2.*mu.*dthtshp(n,:)*dpo(n,:);

G1CUR=G1NP1;
g1(n,:)=reshape(G1CUR,1,9);

% Calculation of second adaptive weight or [gain], w(0) [or g(0) or h(0)]
% Save current weight update
% Convert 3 x3 matrix to 1 X 9 for storage and ease of index when plotting

G2NP1=(G2CUR+2.*mu.*dthtshp(n,:)*dpo(n-1,:)).*ident;

% Use when not filtering the gain update equation through hipass
% G2NP1=G2CUR+2.*mu.*dthtshp(n,:)*dpo(n-1,:);

G2CUR=G2NP1;
g2(n,:)=reshape(G2CUR,1,9);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Uncomment next three lines when using linear and cubic model implementation
% Cubic gain update equation. only one weight for cubic terms.
% CUNP1=CUCUR+2.*muc.*dthtshp(n,:)*dpocub(n-1,:);
% CUCUR=CUNP1;
% cub(n,:)=reshape(CUCUR,1,9);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End LMS Loop

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract Stripped delta theta x gyro data
% Extract Stripped delta theta x gyro data
% Extract Stripped delta theta x gyro data

134
Appendix C: (Continued)

dthtsgyx=dthtsgy(:,1);
dthtsgyy=dthtsgy(:,2);
dthtsgyz=dthtsgy(:,3);

%%% If needed save data to MATLAB format data files.
% save c:\matlab\dthtsgy.mat dthtsgy
% save c:\matlab\g1.mat g1
% save c:\matlab\g2.mat g2
%
% Use for generating portable graphs in Post script or windows meta file formats
%print -dpsc2 <parameter name>
%print -dmeta <parameter name>
%
%%% Plot Dither Gyro Information
%
%%% Calculate running sums for use in PSD and CPSD
%%% Detrend delta theta gyro data
%%% Generate PSD delta theta gyro data
%%% Generate CPSD delta theta gyro data

sum(1,:) = dgy(n2proc,:);
for I=2:n2proc,
    sum(I,1) = sum(I-1,1) + dgy(I+n2proc,1);
    sum(I,2) = sum(I-1,2) + dgy(I+n2proc,2);
    sum(I,3) = sum(I-1,3) + dgy(I+n2proc,3);
end;
for I = 1:3,
    p=polyfit(count,sum(:,I),1);                 %remove trend in gyro data
    dsum(:,I)=sum(:,I)-polyval(p,count);
    Y(:,I) = fft(dsum(:,I),n2proc);
    Pyy(:,I) = Y(:,I).*conj(Y(:,I))/n2proc/n2proc;
    CPyy(:,I) = 2*Pyy(:,I);
    for j = 2:n2proc/2,
        CPyy(j,I) = CPyy(j-1,I) + 2*Pyy(j,I);
    end;

135
end;

% Prepare for plots
% Save plot data

f = fs/n2proc*(0:n2proc/2-1);
freq = f';
clear f;
figure;
plot(freq, Pyy(1:n2proc/2,1),'r',freq, Pyy(1:n2proc/2,2),'b',freq, ...
Pyy(1:n2proc/2,3),'g');
title('Figure 6 - PSD of Correlated Gyro Readout Noise')
xlabel('Frequency (HZ)');
ylabel('Correlated Gyro Readout Noise/Hertz; ARC*SEC^2/Hz')
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green')
grid;
print -dmeta Fig6;

figure;
plot(freq, CPyy(1:n2proc/2,1),'r',freq, CPyy(1:n2proc/2,2),'b',freq, ...
CPyy(1:n2proc/2,3),'g');
title('Figure 7 - Cumulative PSD-Correlated Gyro Readout Noise')
xlabel('Frequency (HZ)');
ylabel('Correlated Gyro Readout Noise: ARC*SEC^2')
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green')
grid;
print -dmeta Fig7;

% If needed save cplsd data in ascii format
% cpsdlms = [freq CPyy(1:n2proc/2,1) CPyy(1:n2proc/2,2) CPyy(1:n2proc/2,3)];
% save c:\matlab\cpsdlms.dat cpsdlms -ascii;
% Plot Dither Pick Off Information
%
% Calculate running sums for use in PSD and CPSD
% Detrend Gyro dither pick off data
% Generate PSD Gyro dither pick off data
% Generate CPSD Gyro dither pick off data

sum(1,:) = dpo(n2proc,:);
for I=2:n2proc,
    sum(I,1) = sum(I-1,1) + dpo(I+n2proc,1);
    sum(I,2) = sum(I-1,2) + dpo(I+n2proc,2);
    sum(I,3) = sum(I-1,3) + dpo(I+n2proc,3);
end;

% Detrend data
% Generate PSD Pick Off Data
% Generate CPSD Pick Off Data

for I = 1:3,
    p=polyfit(count,sum(:,I),1);
    dsum(:,I)=sum(:,I)-polyval(p,count);
    Y(:,I) = fft(dsum(:,I),n2proc);
    Pyy(:,I) = Y(:,I).*conj(Y(:,I))/n2proc/n2proc;
    CPyy(:,I) = 2*Pyy(:,I);
    for j = 2:n2proc/2,
        CPyy(j,I) = CPyy(j-1,I) + 2*Pyy(j,I);
    end;
end;

% Prepare for plots
% Save plot data

f = fs/n2proc*(0:n2proc/2-1);
freq = f';
clear f;
figure;
plot(freq, Pyy(1:n2proc/2,1), 'r', freq, Pyy(1:n2proc/2,2), 'b', freq, ...  
    Pyy(1:n2proc/2,3), 'g');
title('Figure 8-PSD of Gyro Dither Pick Off')
xlabel('FREQUENCY (HZ)');
ylabel('Noise Reference: A/D Count Value/Hz')
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green')
grid;
print -dmeta Fig8;

% Prepare for plots
% Save plot data

figure;
plot(freq, CPyy(1:n2proc/2,1), 'r', freq, CPyy(1:n2proc/2,2), 'b',freq, ...
Appendix C: (Continued)

CPyy(1:n2proc/2,3), 'g');
title('Figure 9-Cumulative PSD of Gyro Dither Pick Off')
xlabel('FREQUENCY (HZ)');
ylabel('Noise Reference: A/D Count Value')
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green')
grid;
print -dmeta Fig9;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot learning curves
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
subplot(3,1,1), plot(count(1:500), dthtsgyx(1:500), 'red');
title('Figure 11a-LMS Learning Curve')
legend('GyroX-Red')
grid;
subplot(3,1,2), plot(count(1:500), dthtsgyy(1:500), 'green');
title('Figure 11b-LMS Learning Curve')
ylabel('Uncorrelated Readout Noise: ARC-SEC^2')
legend('GyroY-Green')
grid;
subplot(3,1,3), plot(count(1:500), dthtsgyz(1:500), 'blue');
title('Figure 11c-LMS Learning Curve')
xlabel('Sample');
legend('GyroZ-Blue')
grid;
print -dmeta Fig11;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Plot G1 Gains (Weights or coefficients) all together.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
plot (g1(1:2000,1),'red'), title('Figure 12-LMS G1 Gain Values'), ...
xlabel('1.6K Hz Samples'), ylabel('Gain Magnitude')
Grid;
hold on
%plot (g1(1:2000,2),'cyan')
Appendix C: (Continued)

```matlab
% plot (g1(1:2000,3),'black')
% plot (g1(1:2000,4),'yellow')
plot (g1(1:2000,5),'blue')
% plot (g1(1:2000,6),'magenta')
% plot (g1(1:2000,7),'black')
% plot (g1(1:2000,8),'blue')
plot (g1(1:2000,9),'green')
hold off
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green')

print -dmeta Fig12

figure;
subplot(3,3,1), plot (g1(1:2000,1),'red'), title('Figure 13a-LMS G1xx'), ... 
ylabel('Gain Magnitude')
Grid;
subplot(3,3,2), plot (g1(1:2000,2),'blue'), title('Figure 13b-LMS G1xy')
Grid;
subplot(3,3,3), plot (g1(1:2000,3),'black'), title('Figure 13c-LMS G1xz')
Grid;
subplot(3,3,4), plot (g1(1:2000,4),'cyan'), title('Figure 13d-LMS G1yx'), ... 
ylabel('Gain Magnitude')
Grid;
subplot(3,3,5), plot (g1(1:2000,5),'blue'), title('Figure 13e-LMS G1yy')
Grid;
subplot(3,3,6), plot (g1(1:2000,6),'magenta'), title('Figure 13f-LMS G1yz')
Grid;
subplot(3,3,7), plot (g1(1:2000,7),'green'), title('Figure 13g-LMS G1zx'), ... 
xlabel('1.6K Hz Samples'), ylabel('Gain Magnitude')
Grid;
subplot(3,3,8), plot (g1(1:2000,8),'blue'), title('Figure 13h-LMS G1zy'), ... 
xlabel('1.6K Hz Samples')
Grid;
subplot(3,3,9), plot (g1(1:2000,9),'green'), title('Figure 13i-LMS G1zz'), ... 
xlabel('1.6K Hz Samples')
Grid;
print -dmeta Fig13;
```
Appendix C: (Continued)

```
figure;
plot (g2(1:2000,1),'red'), title('Figure 14-LMS G2 Gain Values'), ...
xlabel('1.6K Hz Samples'), ylabel('Gain Magnitude')
grid;
hold on
%plot (g2(1:2000,2),'black')
%plot (g2(1:2000,3),'yellow')
%plot (g2(1:2000,4),'cyan')
plot (g2(1:2000,5),'blue')
%plot (g2(1:2000,6),'magenta')
%plot (g2(1:2000,7),'cyan')
%plot (g2(1:2000,8),'black')
plot (g2(1:2000,9),'green')
hold off
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green')
print -dmeta Fig14;
```

```
figure;
subplot(3,3,1), plot (g2(1:2000,1),'red'), title('Figure 15a-LMS G2xx'), ...
ylabel('Gain Magnitude')
Grid;
subplot(3,3,2), plot (g2(1:2000,2),'yellow'), title('Figure 15b-LMS G2xy')
Grid;
subplot(3,3,3), plot (g2(1:2000,3),'black'), title('Figure 15c-LMS G2xz')
Grid;
subplot(3,3,4), plot (g2(1:2000,4),'cyan'), title('Figure 15d-LMS G2yx'), ...
ylabel('Gain Magnitude')
Grid;
subplot(3,3,5), plot (g2(1:2000,5),'blue'), title('Figure 15e-LMS G2yy')
Grid;
```
subplot(3,3,6), plot (g2(1:2000,6),'magenta'), title('Figure 15f-LMS G2yz')
Grid;
subplot(3,3,7), plot (g2(1:2000,7),'black'), title('Figure 15g-LMS G2zx'), ...
xlabel('1.6K Hz Samples'), ylabel('Gain Magnitude')
Grid;
subplot(3,3,8), plot (g2(1:2000,8),'blue'), title('Figure 15h-LMS G2zy'), ...
xlabel('1.6K Hz Samples')
Grid;
subplot(3,3,9), plot (g2(1:2000,9),'green'), title('Figure 15i-LMS G2zz'), ...
xlabel('1.6K Hz Samples')
Grid;
print -dmeta Fig15;

%%%%%% Plot Cubic Gains (Weights or coefficients) individually.
%%%%%%

%figure;
% subplot(3,3,1), plot (cub(:,1),'green'), title('Cubxx'), ylabel('Gain Magnitude')
% Grid;
% subplot(3,3,2), plot (cub(:,2),'blue'), title('Cubxy')
% Grid;
% subplot(3,3,3), plot (cub(:,3),'black'), title('Cubxz')
% Grid;
% subplot(3,3,4), plot (cub(:,4),'cyan'), title('Cubyx'), ylabel('Gain Magnitude')
% Grid;
% subplot(3,3,5), plot (cub(:,5),'red'), title('Cubyy'),
% Grid;
% subplot(3,3,6), plot (cub(:,6),'magenta'), title('Cubyz')
% Grid;
% subplot(3,3,7), plot (cub(:,7),'green'), title('Cubzx'), ...
xlabel('1.6K Hz Samples'), ylabel('Gain Magnitude')
% Grid;
% subplot(3,3,8), plot (cub(:,8),'blue'), title('Cubzy'), xlabel('1.6K Hz Samples')
% Grid;
% subplot(3,3,9), plot (cub(:,9),'black'), title('Cubzz'), xlabel('1.6K Hz Samples')
% Grid;
% print -dmeta LCUBSUBS;
Appendix C: (Continued)

%%%%%% Plot Cubic Gains (Weights or coefficients) all together.
%%%%%%

figure;
plot (cub(:,1),'green'), title('Cubic Gain Values'), ...
xlabel('1.6K Hz Samples'), ylabel('Gain Magnitude')
hold on
plot (cub(:,2),'blue')
plot (cub(:,3),'black')
plot (cub(:,4),'cyan')
plot (cub(:,5),'red')
plot (cub(:,6),'magenta')
plot (cub(:,7),'green')
plot (cub(:,8),'blue')
plot (cub(:,9),'black')
hold off
Grid;
print -dmeta LCBUALL;

figure;
subplot(3,1,1), plot(g1(1:8192,1), g2(1:8192,1),'red');
title('Figure 16a,b,c-LMS G1 Gain Versus G2 Gain Curves')
xlabel('G1x Gain Magnitude');
ylabel('G2x Gain Magnitude')
legend('GyroX-Red')
grid;
subplot(3,1,2), plot(g1(1:8192,5), g2(1:8192,5), 'green');
xlabel('G1y Gain Magnitude');
ylabel('G2y Gain Magnitude')
legend('GyroY-Green')
grid;
subplot(3,1,3), plot(g1(1:8192,9), g2(1:8192,9), 'blue');
xlabel('G1z Gain Magnitude');
ylabel('G2z Gain Magnitude')
Appendix C: (Continued)

legend('GyroZ-Blue')
grid;
print -dmeta Fig16;

... 

figure;
subplot(3,1,1), plot(g1(1:8192,1), dhtsgyx(1:8192), 'red');
hold on;
plot(g2(1:8192,1), dhtsgyx(1:8192), 'blue');
hold off;
title('Figure 19a,b,c-LMS G1 Gains Versus Uncorrelated Readout Noise')
xlabel('G1x & G2x Gain Magnitude');
legend('Gyro G1X-Red', 'Gyro G2X-Blue');
grid;
subplot(3,1,2), plot(g1(1:8192,5), dhtsgyy(1:8192), 'red');
hold on;
plot(g2(1:8192,5), dhtsgyy(1:8192), 'blue');
hold off;
xlabel('G1y and G2y Gain Magnitude');
ylabel('Uncorrelated Gyro Readout Noise/Hertz: ARC-SEC^2/Hz')
legend('Gyro G1Y-Red', 'Gyro G2Y-Blue');
grid;
subplot(3,1,3), plot(g1(1:8192,9), dhtsgyz(1:8192), 'red');
hold on;
plot(g2(1:8192,9), dhtsgyz(1:8192), 'blue');
hold off;
xlabel('G1z and G2z Gain Magnitude');
legend('Gyro G1Z-Red', 'Gyro G2Z-Blue');
grid;
print -dmeta Fig19;

...
Appendix C: (Continued)

% Extract X,Y and Z Uncorrelated data from matrix into vectors
% Calculate running sums for use in PSD and CPSD
% Detrend Uncorrelated Gyro data
% Generate PSD Uncorrelated Gyro data
% Generate CPSD Uncorrelated Gyro data

dthtsgyx = dthtsgy(:,1);           %extract stripped delta theta x data
dthtsgyy = dthtsgy(:,2);           %extract stripped delta theta y data
dthtsgyz = dthtsgy(:,3);           %extract stripped delta theta z data

sum(1,:) = dthtsgy(n2proc,:);
for I=2:n2proc,
    sum(I,1) = sum(I-1,1) + dthtsgy(I+n2proc,1);
    sum(I,2) = sum(I-1,2) + dthtsgy(I+n2proc,2);
    sum(I,3) = sum(I-1,3) + dthtsgy(I+n2proc,3);
end;

for I = 1:3,
    p=polyfit(count,sum(:,I),1);           %remove trend in gyro data
    dsum(:,I)=sum(:,I)-polyval(p,count);
    Y(:,I) = fft(dsum(:,I),n2proc);
    Pyy(:,I) = Y(:,I).*conj(Y(:,I))/n2proc/n2proc;
    CPyy(:,I) = 2*Pyy(:,1);
    for j = 2:n2proc/2,
        CPyy(j,I) = CPyy(j-1,I) + 2*Pyy(j,I);
    end;
end;

% Prepare for plots
% Save plot data

f = fs/n2proc*(0:n2proc/2-1);
freq = f;
clear f;
figure;
plot(freq, Pyy(1:n2proc/2,1),'r',freq, Pyy(1:n2proc/2,2),'b',freq, ...    Pyy(1:n2proc/2,3),'g');
title('Figure 20-LMS PSD-Stripped Delta Theta Uncorrelated Readout Noise')
xlabel('FREQUENCY (HZ)');
ylabel('Uncorrelated Gyro Readout Noise/Hertz: ARC-SEC^2/Hz');
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig20;

% Prepare for plots
% Save plot data

figure;
plot(freq, CPyy(1:n2proc/2,1),'r', freq, CPyy(1:n2proc/2,2),'b', freq, ...
     CPyy(1:n2proc/2,3),'g');
title('Figure 21-LMS CPSD-Stripped Delta Theta Uncorrelated Readout Noise');
xlabel('FREQUENCY (HZ)');
ylabel('Uncorrelated Gyro Readout Noise: ARC-SEC^2');
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig21;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate covariance of 50 point sliding window
% Entries are
%   COV[X,Y]=(1,1)=variance X                       (1,2)=correlation coefficient*varX*varY
%   (2,1)=correlation                       (2,2)=variance of Y
% coefficient*varX*varY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m=50;
for n=1:7951;
% Covariance Matrix of Correlated Gyro data and Pick Off
  covxd=cov(dgyx(n:m),dpox(n:m));
  covyd=cov(dgyy(n:m),dpoy(n:m));
  covzd=cov(dgyz(n:m),dpoz(n:m));
% Covariance Matrix of Uncorrelated Gyro data and Pick Off
  covsxd=cov(dthtsgyx(n:m),dpox(n:m));
  covsyd=cov(dthtsgyy(n:m),dpoy(n:m));
  covszd=cov(dthtsgyz(n:m),dpoz(n:m));
% Reformat data from each 2 by 2 matrix to n by 4 Matrix for
% ease of plot index
  recordcovxd(n,:)=reshape(covxd,1,4);
  recordcovyd(n,:)=reshape(covyd,1,4);
  recordcovzd(n,:)=reshape(covzd,1,4);
% Reformat data from each 2 by 2 matrix to n by 4 Matrix for ease of plot

\[
\text{recordcovsd}(n,:) = \text{reshape}([\text{covsxd}, 1, 4]);
\]
\[
\text{recordcovsd}(n,:) = \text{reshape}([\text{covsyd}, 1, 4]);
\]
\[
\text{recordcovsd}(n,:) = \text{reshape}([\text{covszd}, 1, 4]);
\]

% Calculate Correlation Coefficient of Correlated data

\[
\text{CorCox}(n) = \frac{\text{covx}(1,2)}{\sqrt{\text{covx}(1,1)\cdot\text{covx}(2,2)}};
\]
\[
\text{CorCoy}(n) = \frac{\text{covy}(1,2)}{\sqrt{\text{covy}(1,1)\cdot\text{covy}(2,2)}};
\]
\[
\text{CorCoz}(n) = \frac{\text{covz}(1,2)}{\sqrt{\text{covz}(1,1)\cdot\text{covz}(2,2)}};
\]
% Transpose

\[
\text{CorCoefx} = \text{CorCox}';
\]
\[
\text{CorCoefy} = \text{CorCoy}';
\]
\[
\text{CorCoefz} = \text{CorCoz}';
\]

% Calculate Correlation Coefficient of Correlated data

\[
\text{CorCosx}(n) = \frac{\text{covsx}(1,2)}{\sqrt{\text{covsx}(1,1)\cdot\text{covsx}(2,2)}};
\]
\[
\text{CorCosy}(n) = \frac{\text{covsy}(1,2)}{\sqrt{\text{covsy}(1,1)\cdot\text{covsy}(2,2)}};
\]
\[
\text{CorCosz}(n) = \frac{\text{covsz}(1,2)}{\sqrt{\text{covsz}(1,1)\cdot\text{covsz}(2,2)}};
\]
% Transpose

\[
\text{CorCoefsx} = \text{CorCosx}';
\]
\[
\text{CorCoefsy} = \text{CorCosy}';
\]
\[
\text{CorCoefsz} = \text{CorCosz}';
\]

% increase index

\[
m = m + 1;
\]

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Variance of Correlated Gyro Readout Noise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure;
plot(recordcovxd(1:1000,1), 'red');
hold on;
plot(recordcovyd(1:1000,1), 'blue');
plot(recordcovzd(1:1000,1), 'green');
hold off;
xlabel('50 Point Sliding Window Sample');
ylabel('Variance Correlated Readout Noise');
Title('Figure 58 - LMS - Variance Correlated Readout Noise')
Appendix C: (Continued)

legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig58;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Variance of Dither Pick Off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
plot(recordcovxd(1:1000,4),'red');
hold on;
pplot(recordcovyd(1:1000,4), 'blue');
pplot(recordcovzd(1:1000,4), 'green');
hold off;
xlabel('50 Point Sliding Window Sample');
ylabel('Variance Magnitude');
Title('Figure 59 - LMS - Variance Dither Pick Off');
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig59;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Co-variance of Correlated Gyro Readout noise to Dither Pick Off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
plot(recordcovxd(1:1000,3), 'red');
hold on;
pplot(recordcovyd(1:1000,3), 'blue');
pplot(recordcovzd(1:1000,3), 'green');
hold off;
xlabel('50 Point Sliding Window Sample');
ylabel('Co-Variance Magnitude');
Title('Figure 60 - LMS - CoVariance Correlated Readout Noise to Dither Pick Off')
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig60;
Appendix C: (Continued)

```matlab
figure; plot(CorCoefx(1:7000)', 'red'); hold on; plot(CorCoefy(1:7000)', 'blue'); plot(CorCoefz(1:7000)', 'green'); hold off; xlabel('50 Point Sliding Window Sample'); ylabel('Correlation Coefficient Magnitude'); Title... ('Figure 61-LMS-Correlation Coefficient Correlated Readout Noise to Pick Off Gyro'); legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green'); grid; print -dmeta Fig61;

figure; plot(recordcovsxd(1:1000,1), 'red'); hold on; plot(recordcovsyd(1:1000,1), 'blue'); plot(recordcovszd(1:1000,1), 'green'); hold off; xlabel('50 Point Sliding Window Sample'); ylabel('Variance Correlated Readout Noise'); Title('Figure 62 - LMS - Variance Un-Correlated Readout Noise') legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green') grid; legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green') print -dmeta Fig62;

figure; plot(recordcovsxd(1:1000,1), 'red'); hold on; plot(recordcovsyd(1:1000,1), 'blue'); plot(recordcovszd(1:1000,1), 'green'); hold off; xlabel('50 Point Sliding Window Sample'); ylabel('Variance Correlated Readout Noise'); Title('Figure 62 - LMS - Variance Un-Correlated Readout Noise') legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green') grid; legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green') print -dmeta Fig62;
```

% % Plot Variance of Dither Pick Off %
Appendix C: (Continued)

```matlab
figure;
plot(recordcovsxd(1:1000,4),'red');
hold on;
plot(recordcovsyd(1:1000,4),'blue');
plot(recordcovszd(1:1000,4),'green');
hold off;
xlabel('50 Point Sliding Window Sample');
ylabel('Variance Magnitude');
Title('Figure TBD Variance Dither Pick Off');
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;

% Plot Co-variance of Un-Correlated Gyro Readout noise to Dither Pick OFF
figure;
plot(recordcovsxd(1:1000,3),'red');
Hold on;
plot(recordcovsyd(1:1000,3),'blue');
plot(recordcovszd(1:1000,3),'green');
hold off;
xlabel('50 Point Sliding Window Sample');
ylabel('Co-Variance Magnitude');
Title...
('Figure 63 - LMS - Co-Variance Un-Correlated Readout Noise to Dither Pick Off');
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig63;

figure;
plot(CorCoefsx(1:7000),'red');
Hold on;
```
plot(CorCoefsy(1:7000)','blue');
plot(CorCoefsz(1:7000)','green');
hold off;
xlabel('50 Point Sliding Window Sample');
ylabel('Correlation Coefficient Magnitude');
Title...
('Figure 64-LMS-Correlation Coefficient Un-Correlated Readout Noise to Pick Off Gyro');
legend('GyroX-Red', 'GyroY-Blue', 'GyroZ-green');
grid;
print -dmeta Fig64;
% take a 1000 point sample mean of the:
%1.) Variance of Correlated Readout noise
Vcorgx=mean(recordcovxd(1000:2000,1))
Vcorgy=mean(recordcovyd(1000:2000,1))
Vcorgz=mean(recordcovzd(1000:2000,1))
%2.) Variance Correlated Pickoff
Vpx=mean(recordcovxd(1000:2000,4))
Vpy=mean(recordcovyd(1000:2000,4))
Vpz=mean(recordcovzd(1000:2000,4))
%3.) Covariance of Correlated gyro to pick off
CVcorgpx=mean(recordcovxd(1000:2000,2))
CVcorgpy=mean(recordcovyd(1000:2000,2))
CVcorgpz=mean(recordcovzd(1000:2000,2))
%4.) Correlation Coefficient Before Stipping
CCgpx=mean(CorCoefx(1000:2000))
CCgpy=mean(CorCoefy(1000:2000))
CCgpz=mean(CorCoefz(1000:2000))
%5.) Variance uncorrelated Gyro Readout noise after stripping
Vungx=mean(recordcovsxd(1000:2000,1))
Vungy=mean(recordcovsyd(1000:2000,1))
Vungz=mean(recordcovszd(1000:2000,1))
%2.) Variance Uncorrelated Pickoff
Vunpx=mean(recordcovxd(1000:2000,4))
Vunpy=mean(recordcovyd(1000:2000,4))
Vunpz=mean(recordcovzd(1000:2000,4))
%6.) Covariance Uncorrelated Gyro Readout noise to picokoff
CVungpx=mean(recordcovsxd(1000:2000,2))
CVungpy=mean(recordcovsyd(1000:2000,2))
CVungpz=mean(recordcovszd(1000:2000,2))
% 7.) Correlation Coefficient after stripping
CCungpx=mean(CorCoefsx(1000:2000))
CCungpy=mean(CorCoefsy(1000:2000))
CCungpz=mean(CorCoefsz(1000:2000))

% Size of matrix, vectors and variables
% Name     Size     Bytes     Class
% CCgpx    1x1       8         double array
% Ccgpy    1x1       8         double array
% CCgpz    1x1       8         double array
% CCungpz  1x1       8         double array
% CCungpx  1x1       8         double array
% CCungpy  1x1       8         double array
% CPyy     4096x3  98304    double array
% CUCUR    3x3       72       double array
% CUNP1    3x3       72       double array
% CVcorgpx 1x1       8         double array
% CVcorgpy 1x1       8         double array
% CVcorgpz 1x1       8         double array
% CVungpz  1x1       8         double array
% CVungpx  1x1       8         double array
% CVungpy  1x1       8         double array
% CorCoefsx 7951x1  63608     double array
% CorCoefsy 7951x1  63608     double array
% CorCoefsz 7951x1  63608     double array
% CorCoefx  7951x1  63608     double array
% CorCoefy  7951x1  63608     double array
% CorCoefz  7951x1  63608     double array
% CorCosx  1x7951  63608     double array
% CorCosy  1x7951  63608     double array
% CorCosz  1x7951  63608     double array
% CorCox   1x7951  63608     double array
% CorCoy   1x7951  63608     double array
% CorCoz   1x7951  63608     double array
% G1CUR    3x3       72       double array
% G1NP1    3x3       72       double array
% G2CUR    3x3       72       double array
% G2NP1    3x3       72       double array
% I        1x1       8         double array
% Pyy      4096x3  98304    double array
% Vcorgx   1x1       8         double array
% Vcorgy   1x1       8         double array
Appendix C: (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Size</th>
<th>Count</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcorgz</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vpx</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vpy</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vpz</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vungx</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vungy</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vungz</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vunpx</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vunpy</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Vunpz</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Y</td>
<td>4096x3</td>
<td>196608</td>
<td>double array</td>
</tr>
<tr>
<td>b</td>
<td>8192x11</td>
<td>720896</td>
<td>double array</td>
</tr>
<tr>
<td>count</td>
<td>4096x1</td>
<td>32768</td>
<td>double array</td>
</tr>
<tr>
<td>covsxld</td>
<td>2x2</td>
<td>32</td>
<td>double array</td>
</tr>
<tr>
<td>covsxd</td>
<td>2x2</td>
<td>32</td>
<td>double array</td>
</tr>
<tr>
<td>covsxd</td>
<td>2x2</td>
<td>32</td>
<td>double array</td>
</tr>
<tr>
<td>covxd</td>
<td>2x2</td>
<td>32</td>
<td>double array</td>
</tr>
<tr>
<td>covyd</td>
<td>2x2</td>
<td>32</td>
<td>double array</td>
</tr>
<tr>
<td>covzd</td>
<td>2x2</td>
<td>32</td>
<td>double array</td>
</tr>
<tr>
<td>dgy</td>
<td>8192x3</td>
<td>196608</td>
<td>double array</td>
</tr>
<tr>
<td>dgyx</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dgyy</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dgyz</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dpo</td>
<td>8192x3</td>
<td>196608</td>
<td>double array</td>
</tr>
<tr>
<td>dpoxy</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dpoxcu</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dpy</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dpoycu</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dpoz</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dpozcu</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dsum</td>
<td>4096x3</td>
<td>98304</td>
<td>double array</td>
</tr>
<tr>
<td>dthtd_est</td>
<td>3x1</td>
<td>24</td>
<td>double array</td>
</tr>
<tr>
<td>dthtdest</td>
<td>8192x3</td>
<td>196608</td>
<td>double array</td>
</tr>
<tr>
<td>dthts_gy</td>
<td>1x3</td>
<td>24</td>
<td>double array</td>
</tr>
<tr>
<td>dthts_hp</td>
<td>3x1</td>
<td>24</td>
<td>double array</td>
</tr>
<tr>
<td>dthtsgy</td>
<td>8192x3</td>
<td>196608</td>
<td>double array</td>
</tr>
<tr>
<td>dthtsgyx</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dthtsgyy</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dthtsgyz</td>
<td>8192x1</td>
<td>65536</td>
<td>double array</td>
</tr>
<tr>
<td>dthshp</td>
<td>8192x3</td>
<td>196608</td>
<td>double array</td>
</tr>
<tr>
<td>fid</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Shape</td>
<td>Size</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>freq</td>
<td></td>
<td>2048x1</td>
<td>16384</td>
</tr>
<tr>
<td>fs</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>g1</td>
<td></td>
<td>8192x9</td>
<td>589824</td>
</tr>
<tr>
<td>g2</td>
<td></td>
<td>8192x9</td>
<td>589824</td>
</tr>
<tr>
<td>gy</td>
<td></td>
<td>8192x3</td>
<td>196608</td>
</tr>
<tr>
<td>gyx</td>
<td></td>
<td>8192x1</td>
<td>65536</td>
</tr>
<tr>
<td>gyy</td>
<td></td>
<td>8192x1</td>
<td>65536</td>
</tr>
<tr>
<td>gyz</td>
<td></td>
<td>8192x1</td>
<td>65536</td>
</tr>
<tr>
<td>ident</td>
<td></td>
<td>3x3</td>
<td>72</td>
</tr>
<tr>
<td>j</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>m</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>mu</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>muc</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>n2proc</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>n2read</td>
<td></td>
<td>1x1</td>
<td>8</td>
</tr>
<tr>
<td>p</td>
<td></td>
<td>1x2</td>
<td>16</td>
</tr>
<tr>
<td>po</td>
<td></td>
<td>8192x3</td>
<td>196608</td>
</tr>
<tr>
<td>pox</td>
<td></td>
<td>8192x1</td>
<td>65536</td>
</tr>
<tr>
<td>poy</td>
<td></td>
<td>8192x1</td>
<td>65536</td>
</tr>
<tr>
<td>poz</td>
<td></td>
<td>8192x1</td>
<td>65536</td>
</tr>
<tr>
<td>recordcovsx</td>
<td></td>
<td>7951x4</td>
<td>254432</td>
</tr>
<tr>
<td>recordcovsy</td>
<td></td>
<td>7951x4</td>
<td>254432</td>
</tr>
<tr>
<td>recordcovsz</td>
<td></td>
<td>7951x4</td>
<td>254432</td>
</tr>
<tr>
<td>recordcovxd</td>
<td></td>
<td>7951x4</td>
<td>254432</td>
</tr>
<tr>
<td>recordcovyd</td>
<td></td>
<td>7951x4</td>
<td>254432</td>
</tr>
<tr>
<td>recordcovzd</td>
<td></td>
<td>7951x4</td>
<td>254432</td>
</tr>
<tr>
<td>sum</td>
<td></td>
<td>4096x3</td>
<td>98304</td>
</tr>
</tbody>
</table>

%Grand total is 935584 elements using 7582976 byte
Appendix D: VHDL Listings

lms_tb_48.vhdl and Least Mean Square, lms.vhdl code are listed in this appendix.

I.) lms_tb_48.vhdl (listed)
   clk_gen.vhdl(not listed)
   div2_q0.vhdl(not listed)
   div2_tc.vhdl(not listed)
   div5_tc.vhdl(not listed)
   div8_q0tc.vhdl(not listed)
   clk_ng.vhdl(not listed)
   lms_read_data.vhdl(not listed)
   lms_write_data.vhdl(not listed)

II.) lms.vhdl (listed)
   lms_sm.vhdl(not listed)
   reg_16.vhdl(not listed)
   reg_48.vhdl(not listed)
   mux_16.vhdl(not listed)
   mux_48.vhdl(not listed)
   calc_delta.vhdl(not listed)
   add_sub.vhdl(not listed)
   mult_48_slv.vhdl(not listed)
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.all;
LIBRARY work;

ENTITY tb_48_ent IS
  --Empty Entity
END tb_48_ent;

ARCHITECTURE tb_48_arch OF tb_48_ent IS

-- List CONSTANTS used in the EPIC TESTBENCH:

CONSTANT lms_clk_period : time := 31 ns; --32 MHz oscillator
PORT (  
c32mhz : IN std_logic;  
c16mhz : IN std_logic;  
reset : IN std_logic;  
  
c16mhz_out : OUT std_logic;  
p1600hz : OUT std_logic);  
END COMPONENT;  
  
--synopsys translate_off  
-- for all : clkgen_ent USE entity lms.clkgen_ent(clkgen_arch);  
-- for all : clkgen_ent USE entity work.clkgen_ent(clkgen_arch);  
-- synopsys translate_on  
COMPONENT clock_ng_ent  
PORT (  
control : IN std_logic;  
clock_period : IN time;  
  
clk_signal : OUT std_logic := '1'  
);  
END COMPONENT;  
  
-- synopsys translate_off  
-- for all: clock_ng_ent USE entity lms.clock_ng_ent(clock_ng_arch);  
-- for all: clock_ng_ent USE entity work.clock_ng_ent(clock_ng_arch);  
-- synopsys translate_on  
COMPONENT lms_read_data_ent  
generic (  
  infile1 : string := "/project/lms_dsp/MSEE_DSP/source/input_data/xdpo.dat";  
  infile2 : string := "/project/lms_dsp/MSEE_DSP/source/input_data/xgyro.dat" );  
PORT (  
c16mhz : IN std_logic;  
p1600hz : IN std_logic;  
  
dpo : OUT std_logic_vector(15 downto 0);  
gyro : OUT std_logic_vector(15 downto 0);  
);
COMPONENT lms_write_data_ent
  generic (
    outfile1 : string := "./project/lms_dsp/MSEE_DSP/source/output_data/en_int.dat";
    outfile2 : string := "./project/lms_dsp/MSEE_DSP/source/output_data/en_frac.dat";
    outfile3 : string := "./project/lms_dsp/MSEE_DSP/source/output_data/h0_int.dat";
    outfile4 : string := "./project/lms_dsp/MSEE_DSP/source/output_data/h0_frac.dat";
    outfile5 : string := "./project/lms_dsp/MSEE_DSP/source/output_data/h1_int.dat";
    outfile6 : string := "./project/lms_dsp/MSEE_DSP/source/output_data/h1_frac.dat"
  );
  PORT(
    c16mhz : IN std_logic;
    p1600hz : IN std_logic;
    en : IN std_logic_vector;
    h0 : IN std_logic_vector;
    h1 : IN std_logic_vector
  );
END COMPONENT;

-- synopsys translate_off
-- for all: lms_write_data_ent USE entity work.lms_write_data_ent(lms_write_data_arch);
-- synopsys translate_on

COMPONENT lms_ent
  PORT (
    reset : IN std_logic;
    clk : IN std_logic;
    clk_en : IN std_logic;
    gyro_data : IN std_logic_vector(15 DOWN TO 0);
    dpo_data : IN std_logic_vector(15 DOWN TO 0);
  );
END COMPONENT;
Appendix D: (Continued)

h0 : OUT std_logic_vector(47 downto 0);

h1 : OUT std_logic_vector(47 downto 0);

en : OUT std_logic_vector(47 downto 0)
);

END COMPONENT;

-- synopsys translate_off
-- for all: lms_ent   USE entity lms.lms_ent(lms_arch);
-- for all: lms_ent   USE entity work.lms_ent(lms_arch);
-- synopsys translate_on

---------------------------------------------------------------
-- List Signals Used in test bench
---------------------------------------------------------------
SIGNAL start       : std_logic := '1';
SIGNAL reset       : std_logic := '1';
SIGNAL c32mhz     : std_logic;
SIGNAL c16mhz     : std_logic;
SIGNAL p1600hz    : std_logic;

SIGNAL dpo_data    : std_logic_vector(15 downto 0):=(others => '0');
SIGNAL gyro_data   : std_logic_vector(15 downto 0):=(others => '0');
SIGNAL h0          : std_logic_vector(47 downto 0):= X"000000000003";
SIGNAL h1          : std_logic_vector(47 downto 0):= X"000000000003";
SIGNAL             : std_logic_vector(47 downto 0):=(others => '0');

BEGIN -- lms_tb_arch

CLK_32M: clock_ng_ent
PORT MAP (  
Control => start,
  clock_period => lms_clk_period,
  clk_signal => c32mhz
);

CLKGEN_1: clkgen_ent
PORT MAP (  
c32mhz => c32mhz,
c16mhz => c16mhz,
);
reset => reset,
c16mhz_out => c16mhz,
p1600hz => p1600hz);

LMS_READ_DATA1:lms_read_data_ent
PORT MAP (c16mhz => c16mhz,
p1600hz => p1600hz,
dpo => dpo_data,
gyro => gyro_data);

LMS_WRITE_DATA1:lms_write_data_ent
PORT MAP (c16mhz => c16mhz,
p1600hz => p1600hz,
h0 => h0,
h1 => h1,
en => en);

LMS_FILTER: lms_ent
PORT MAP(reset => reset,
clk => c16mhz,
clk_en => p1600hz,
gyro_data => gyro_data,
dpo_data => dpo_data,
h0 => h0,
h1 => h1,
en => en);

---------------------------------------------------------------------
--Miscellaneous Signal Definition
---------------------------------------------------------------------
reset <= '1',
    '0' after 100 ns;

END tb_48_arch;
LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE ieee.numeric_std.all;
LIBRARY lms;
  USE lms.add_sub_ent;
  USE lms.calc_delta_ent;
  USE lms.mult_48_slv_ent;
  USE lms.reg_16_ent;
  USE lms.reg_48_ent;
  USE lms.mux_16_ent;
  USE lms.mux_48_ent;
  USE lms.lms_sm_ent;

ENTITY lms_ent IS
  PORT(
    Reset                   :IN std_logic;
  )
END lms_ent;

Appendix D: (Continued)

```
clk  :IN std_logic;
clk_en  :IN std_logic;
dpo_data  :IN std_logic_vector(15 downto 0);
gyro_data :IN std_logic_vector(15 downto 0);

h0  :OUT std_logic_vector(47 downto 0) :=X"000000000003";
h1  :OUT std_logic_vector(47 downto 0) :=X"000000000003";
en   :OUT std_logic_vector(47 downto 0) :=X"000000000003";
);
END lms_ent;
```

---------------------------------------------
-- Start Architecture of lms
---------------------------------------------

ARCHITECTURE lms_arch OF lms_ent IS

---------------------------------------------
-- Define Components Used in Least Mean Square Algorithm
---------------------------------------------

COMPONENT calc_delta_ent
  -- generic ();
PORT(
    reset  :IN std_logic;
    clk  :IN std_logic;
    clk_en  :IN std_logic;
    data  :IN std_logic_vector(15 downto 0);

    delta  :OUT std_logic_vector(15 downto 0)
  );
END COMPONENT;

--synopsys translate_off
--for all : calc_delta_ent USE entity lms.calc_delta_ent(calc_delta_arch);
--for all : calc_delta_ent USE entity work.calc_delta_ent(calc_delta_arch);
-- synopsys translate_on

COMPONENT add_sub_ent
PORT ( 
  reset  :IN std_logic;
  clr_sum  :IN std_logic;
  clk  :IN std_logic;
);
COMPONENT mult_48_slv_ent
PORT(
    reset  : IN std_logic;
    clk    : IN std_logic;
    load   : IN std_logic;
    data1  : IN std_logic_vector(47 downto 0);
    data2  : IN std_logic_vector(47 downto 0);
    done   : OUT std_logic;
    prod_out : OUT std_logic_vector(47 downto 0)
);
END COMPONENT;

-- synopsys translate_off
--for all: mult_48_slv_ent USE entity lms.mult_48_slv_ent(mult_48_slv_arch);
--for all: mult_48_slv_ent USE entity work.mult_48_slv_ent(mult_48_slv_arch);
-- synopsys translate_on

COMPONENT lms_sm_ent
PORT(
    clk     : IN std_logic;
    reset   : IN std_logic;
    clk_en  : IN std_logic;
    mult_done : IN std_logic;
    alu_en   : OUT std_logic;
    alu_rst  : OUT std_logic;
    alu_add_sub : OUT std_logic;
    mult_en  : OUT std_logic;
    dgy_reg_en : OUT std_logic;
    ddpo_reg_en : OUT std_logic;
);
Appendix D: (Continued)

```vhdl
en_reg_en : OUT std_logic;
h0_reg_en : OUT std_logic;
h1_reg_en : OUT std_logic;
m2en_reg_en : OUT std_logic;
ddp0_ddponm1_mux : OUT std_logic;
ddp0_mu2_mux : OUT std_logic;
h0h1_mux : OUT std_logic;
gh0h1_mux : OUT std_logic;
en_mu2en_mux : OUT std_logic;
h0h1_enmu2en_mux : OUT std_logic;
sum_in_mux : OUT std_logic);
```

END COMPONENT;

```vhdl
--synopsys translate_off
--for all: lms_sm_ent USE entity lms.lms_sm_ent(lms_sm_arch);
--for all: lms_sm_ent USE entity work.lms_sm_ent(lms_sm_arch);
-- synopsys translate_on

COMPONENT reg_16_ent
PORT (
        clk : IN std_logic;
        clk_en : IN std_logic;
        reset : IN std_logic;
        data : IN std_logic_vector(15 downto 0);
        reg_q : OUT std_logic_vector(15 downto 0);
    );

END COMPONENT;

-- synopsys translate_off
--for all: reg_16_ent USE entity lms.reg_16_ent(reg_16_arch);
--for all: reg_16_ent USE entity work.reg_16_ent(reg_16_arch);
-- synopsys translate_on

COMPONENT reg_48_ent
PORT ( 
    clk : IN std_logic;
    clk_en : IN std_logic;
    reset : IN std_logic;
    data : IN std_logic_vector(47 downto 0);
) ;
```
Appendix D: (Continued)

    reg_q : OUT std_logic_vector(47 downto 0)
);
END COMPONENT;

-- synopsys translate_off
--for all: reg_48_ent USE entity lms.reg_48_ent(reg_48_arch);
--for all: reg_48_ent USE entity work.reg_48_ent(reg_48_arch);
-- synopsys translate_on

COMPONENT mux_16_ent
PORT (  
    sel : IN std_logic;
    ina : IN std_logic_vector(15 downto 0);
    inb : IN std_logic_vector(15 downto 0);

    mux_out : OUT std_logic_vector(15 downto 0)
);
END COMPONENT;

--synopsys translate_off
--for all: mux_16_ent USE entity lms.mux_16_ent(mux_16_arch);
--for all: mux_16_ent USE entity work.mux_16_ent(mux_16_arch);
--synopsys translate_on

COMPONENT mux_48_ent
PORT (  
    sel : IN std_logic;
    ina : IN std_logic_vector(47 downto 0);
    inb : IN std_logic_vector(47 downto 0);

    mux_out : OUT std_logic_vector(47 downto 0)
);
END COMPONENT;

--synopsys translate_off
--for all: mux_48_ent USE entity lms.mux_48_ent(mux_48_arch);
--for all: mux_48_ent USE entity work.mux_48_ent(mux_48_arch);
--synopsys translate_on

-------------------------------------------------------------------------
-- Define Constant, Variables Signals used in Least Mean Square Algorithm

164
Appendix D: (Continued)

-------------------------------------------------------------------------------
SIGNAL vcc                  :std_logic;
SIGNAL gnd                  :std_logic;
SIGNAL delta_gyro  :std_logic_vector(15 downto 0);
SIGNAL delta_dpo  :std_logic_vector(15 downto 0);
SIGNAL alu_rst  :std_logic;
SIGNAL alu_en  :std_logic;
SIGNAL alu_add_sub :std_logic;
SIGNAL alu_in_data  :std_logic_vector(47 downto 0);
SIGNAL alu_cout  :std_logic;
SIGNAL alu_ovrfl  :std_logic;
SIGNAL alu_out_data  :std_logic_vector(47 downto 0);
SIGNAL mult_en   :std_logic;
SIGNAL mult_done  :std_logic;
SIGNAL mult_data  :std_logic_vector(47 downto 0);
SIGNAL dgy_reg_en  :std_logic;
SIGNAL ddpo_reg_en :std_logic;
SIGNAL en_reg_en  :std_logic;
SIGNAL h0_reg_en  :std_logic;
SIGNAL h1_reg_en   :std_logic;
SIGNAL mu2en_reg_en :std_logic;
SIGNAL ddpo_ddponm1_mux:std_logic;
SIGNAL ddpo_mu2_mux :std_logic;
SIGNAL h0_h1_mux  :std_logic;
SIGNAL gy_h0h1_mux :std_logic;
SIGNAL en_mu2en_mux :std_logic;
SIGNAL h0h1_enmu2en_mux:std_logic;
SIGNAL sum_in_mux :std_logic;
SIGNAL dgy_data  :std_logic_vector(15 downto 0) :="0000";
SIGNAL ddpo_data  :std_logic_vector(15 downto 0) :="0000";
SIGNAL ddponm1_data :std_logic_vector(15 downto 0) :="0000";
SIGNAL ddpo_ddponm1_data:std_logic_vector(15 downto 0) :="0000";
SIGNAL gy_h0h1_data :std_logic_vector(47 downto 0) :="0000";
SIGNAL h0_data  :std_logic_vector(47 downto 0) :="000000000003";
Appendix D: (Continued)

SIGNAL h1_data : std_logic_vector(47 downto 0) := X"000000000003";
SIGNAL h0_h1_data : std_logic_vector(47 downto 0) := X"000000000003";
SIGNAL en_data : std_logic_vector(47 downto 0) := X"000000000000";
SIGNAL mu2_data : std_logic_vector(47 downto 0) := "000000000004";
SIGNAL ddpo_mu2_data : std_logic_vector(47 downto 0) := "000000000000";
SIGNAL mu2en_data : std_logic_vector(47 downto 0) := "000000000000";
SIGNAL h0h1_data : std_logic_vector(47 downto 0) := "000000000000";
SIGNAL h0h1_enmu2en_data : std_logic_vector(47 downto 0) := "000000000000";
SIGNAL en_mu2en_data : std_logic_vector(47 downto 0) := "000000000000";
SIGNAL ddpo_ddponm1_data_scld : std_logic_vector(47 downto 0) := "000000000000";
SIGNAL dgy_data_scld : std_logic_vector(47 downto 0) := "000000000000";

BEGIN

-------------------------------------------------------------------------------
-- Map calc_delta signals for gyro channel
-------------------------------------------------------------------------------

DELTA_GYRO_CALC1: calc_delta_ent
PORT MAP (
  reset => reset,
  clk => clk,
  clk_en => clk_en,
  data => gyro_data,
  delta => delta_gyro
);

-------------------------------------------------------------------------------
-- Map add_offset signals for dither pick off
-------------------------------------------------------------------------------

DELTA_DPO_CALC1: calc_delta_ent
PORT MAP (
  reset => reset,
  clk => clk,
  clk_en => clk_en,
  data => dpo_data,
  delta => delta_dpo
);

ADD_SUB: add_sub_ent
PORT MAP (
  reset => reset,
Appendix D: (Continued)

clr_sum => alu_rst,
clk   => clk,
clk_en   => alu_en,
add_sub  => alu_add_sub,
data_in  => alu_in_data,

data_out  => alu_out_data
);

MULTIPLY_48:mult_48_slv_ent
PORT MAP(
  reset   => reset,
  clk  => clk,
  load  => mult_en,
data1  => ddpo_mu2_data,
data2  => h0h1_enmu2en_data,
done   => mult_done,
prod_out => mult_data
);

LMS_STATE_MACHINE:lms_sm_ent
PORT MAP(
  reset   => reset,
  clk   => clk,
  clk_en => clk_en,
mult_done => mult_done,
alu_en   => alu_en,
alu_rst  => alu_rst,
alu_add_sub => alu_add_sub,
mult_en  => mult_en,
dgy_reg_en  => dgy_reg_en,
ddpo_reg_en  => ddpo_reg_en,
en_reg_en  => en_reg_en,
h0_reg_en  => h0_reg_en,
h1_reg_en  => h1_reg_en,
mu2en_reg_en => mu2en_reg_en,

ddpo_ddponm1_mux  => ddpo_ddponm1_mux,

167
Appendix D: (Continued)

ddpo_mu2_mux => ddpo_mu2_mux,
h0_h1_mux => h0_h1_mux,
gy_h0h1_mux => gy_h0h1_mux,
en_mu2en_mux => en_mu2en_mux,
h0h1_enmu2en_mux => h0h1_enmu2en_mux,
sum_in_mux => sum_in_mux
);

REG_DELTA_GYRO:reg_16_ent
PORT MAP(
  reset => reset,
  clk => clk,
  clk_en => dgy_reg_en,
  data => delta_gyro,
  reg_q => dgy_data
);

REG_DELTA_DPO:reg_16_ent
PORT MAP(
  reset => reset,
  clk => clk,
  clk_en => ddpo_reg_en,
  data => delta_dpo,
  reg_q => ddpo_data
);

REG_DELTA_DPONM1:reg_16_ent
PORT MAP(
  reset => reset,
  clk => clk,
  clk_en => clk_en,
  data => ddpo_data,
  reg_q => ddponm1_data
);

REG_H0:reg_48_ent
PORT MAP(
  Reset => reset,
  clk => clk,
  clk_en => h0_reg_en,
Appendix D: (Continued)

data  => alu_out_data,
reg_q => h0_data
);

REG_H1:reg_48_ent
PORT MAP(
  reset  => reset,
  clk    => clk,
  clk_en => h1_reg_en,
  data   => alu_out_data,

  reg_q  => h1_data
);

REG_EN:reg_48_ent
PORT MAP(
  reset  => reset,
  clk    => clk,
  clk_en => en_reg_en,
  data   => alu_out_data,

  reg_q  => en_data
);

REG_MU2EN:reg_48_ent
PORT MAP(
  reset  => reset,
  clk    => clk,
  clk_en => mu2en_reg_en,
  data   => alu_in_data,

  reg_q  => mu2en_data
);

--------------------------------------------------------------------------
-- Updated to include registered outputs. Allows the synthesizer to route
-- signal buses independently
--------------------------------------------------------------------------

REG_H0_OUT:reg_48_ent
PORT MAP(
  reset  => reset,
  clk    => clk,
Appendix D: (Continued)

CLK EN => VCC,
data => h0 DATA,

REG H1 OUT: REG 48 ENT
PORT MAP(
    reset => reset,
    clk => clk,
    clk_en => VCC,
    data => h1 DATA,

    REG Q => h1
);

REG EN OUT: REG 48 ENT
PORT MAP(
    reset => reset,
    clk => clk,
    clk_en => VCC,
    data => en DATA,

    REG Q => en
);

MUX DDPO DDPONM1: MUX 16 ENT
PORT MAP(
    Sel => ddpo DDPONM1 MUX,
    Ina => ddpo DATA,
    Inb => ddponm1 DATA,

    mux OUT => ddpo DDPONM1 DATA
);

MUX DDPO MU2: MUX 48 ENT
PORT MAP(
    Sel => ddpo mu2 MUX,
    Ina => ddpo DDPONM1 DATA SCLD,
    Inb => mu2 DATA,
Appendix D: (Continued)

mux_out => ddpo_mu2_data
);

MUX_H0_H1: mux_48_ent
PORT MAP(
    sel   => h0_h1_mux,
    ina   => h0_data,
    inb   => h1_data,
    mux_out => h0_h1_data
);

MUX_GY_H0H1: mux_48_ent
PORT MAP(
    sel   => gy_h0h1_mux,
    ina   => dgy_data_scld,
    inb   => h0_h1_data,
    mux_out => gy_h0h1_data
);

MUX_EN_MU2EN: mux_48_ent
PORT MAP(
    sel   => en_mu2en_mux,
    ina   => en_data,
    inb   => mu2en_data,
    mux_out => en_mu2en_data
);

MUX_H0H1_ENMU2EN: mux_48_ent
PORT MAP(
    sel   => h0h1_enmu2en_mux,
    ina   => h0_h1_data,
    inb   => en_mu2en_data,
    mux_out => h0h1_enmu2en_data
);

MUX_SUM_IN: mux_48_ent
PORT MAP(
    sel   => sum_in_mux,
Appendix D: (Continued)

```vhdl
ina => gy_h0h1_data,
inb => mult_data,
mux_out => alu_in_data
);

--------------------------------------------------------------------------
-- Miscellaneous VHDL
--------------------------------------------------------------------------
vcc <= '1';
gnd <= '0';

SIGN_EXTEND_GYRO: PROCESS(dgy_data)
BEGIN
  IF (dgy_data(15) = '0') THEN
    dgy_data_scld(27 downto 0) <= X"0000000";
    dgy_data_scld(43 downto 28) <= dgy_data;
    dgy_data_scld(47 downto 44) <= "0000";
  ELSIF (dgy_data(15) = '1') THEN
    dgy_data_scld(27 downto 0) <= X"0000000";
    dgy_data_scld(43 downto 28) <= dgy_data;
    dgy_data_scld(47 downto 44) <= "1111";
  ELSE
    dgy_data_scld(47 downto 0) <= X"000000000000";
  END IF;
END PROCESS;

SIGN_EXTEND_DPO: PROCESS(ddpo_ddponm1_data)
BEGIN
  IF (ddpo_ddponm1_data(15) = '0') THEN
    ddpo_ddponm1_data_scld(27 downto 0) <= X"0000000";
    ddpo_ddponm1_data_scld(43 downto 28) <= ddpo_ddponm1_data;
    ddpo_ddponm1_data_scld(47 downto 44) <= "0000";
  ELSIF (ddpo_ddponm1_data(15) = '1') THEN
    ddpo_ddponm1_data_scld(27 downto 0) <= X"0000000";
    ddpo_ddponm1_data_scld(43 downto 28) <= ddpo_ddponm1_data;
    ddpo_ddponm1_data_scld(47 downto 44) <= "1111";
  ELSE
    ddpo_ddponm1_data_scld(47 downto 0) <= X"000000000000";
  END IF;
END PROCESS;

END lms_arch;
```
Appendix E: Field Programmable Gate Array Pin List and Timing

The following are output tool listings for the FPGA target including:
- Pin List By Number
- Timing analysis - sample

***********************************************************
Pin Report - Date: Tue May 20 12:08:56 2003 Pin checksum: f3db059b_c28e675c
  Design Name: lms_042103  Family: 54SX4
  Die: RT54SX72S  Package: 256 CQFP
***********************************************************

<table>
<thead>
<tr>
<th>Number</th>
<th>Port Name</th>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
<td>GND</td>
<td>RESERVED</td>
</tr>
<tr>
<td>2</td>
<td>---</td>
<td>TDI/IO</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>3</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>6</td>
<td>dpo_data(0)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>dpo_data(8)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>dpo_data(2)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>dpo_data(1)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>dpo_data(3)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>gyro_data(14)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>gyro_data(9)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>h0(38)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>en(47)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>h1(36)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>en(23)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>en(20)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>h0(23)</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>251</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>252</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>253</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>254</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>255</td>
<td>---</td>
<td>---</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>256</td>
<td>---</td>
<td>TCK/IO</td>
<td>UNASSIGNED</td>
</tr>
</tbody>
</table>
Appendix E: (Continued)

**Timer Version 01.01.01**

Actel Corporation - Actel Designer Software Release R1-2003
Service Pack 1,
Copyright (c) 1989-2003
Date: Tue May 20 07:36:24 2003

Design: lms_ent
Family: 54SX4
Die: RT54SX72S
Package: 256 CQFP
Radiation Exposure: 100 KRad
Temperature: MIL
Voltage: MIL
Speed Grade: STD
Design State: Post-Layout
Timing: Worst Case
Path Tracing: Longest Paths
Break at Clk/G pins: True
Break at Preset/Clr pins: True
Break at Data pins of Latchs: True
Section Clock Frequency

<table>
<thead>
<tr>
<th>Actual</th>
<th>Required</th>
<th>ClockName</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.67MHz</td>
<td>16.00MHz</td>
<td>CLK</td>
</tr>
</tbody>
</table>

End Section

Section $Inputs() to $Outputs()

--- No Paths found ---

End Section

Section $Inputs() to $Registers(CLK):$DataPins()

<table>
<thead>
<tr>
<th>Delay(ns)</th>
<th>Slack(ns)</th>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.86</td>
<td>26.64</td>
<td>reset</td>
<td>DELTA_DPO_CALC1/delta_15:D</td>
</tr>
<tr>
<td>35.08</td>
<td>27.42</td>
<td>reset</td>
<td>DELTA_DPO_CALC1/delta_14:D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MULTIPLY_48/regq_26:D</td>
</tr>
</tbody>
</table>

End Section

Section $Inputs() to $Registers(CLK):$ClockPins()

<table>
<thead>
<tr>
<th>Delay(ns)</th>
<th>Slack(ns)</th>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.89</td>
<td>56.61</td>
<td>CLK</td>
<td>REG_H1_OUT/reg_q_14:CLK</td>
</tr>
<tr>
<td>5.63</td>
<td>56.87</td>
<td>CLK</td>
<td>REG_H1_OUT/reg_q_15:CLK</td>
</tr>
</tbody>
</table>

End Section
## Appendix E: (Continued)

### Section $Inputs()$ to $Registers(CLK):AsyncPins()$

<table>
<thead>
<tr>
<th>Delay (ns)</th>
<th>Slack (ns)</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.54</td>
<td>N/A</td>
<td>reset</td>
<td>REG_H0_OUT/reg_q_33:CLR</td>
</tr>
<tr>
<td>23.54</td>
<td>N/A</td>
<td>reset</td>
<td>REG_H0_OUT/reg_q_32:CLR</td>
</tr>
</tbody>
</table>

### Section $Registers(CLK):InputPins()$ to $Outputs()$

<table>
<thead>
<tr>
<th>Delay (ns)</th>
<th>Slack (ns)</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.82</td>
<td>51.68</td>
<td>REG_EN_OUT/reg_q_8:CLK</td>
<td>en&lt;8&gt;</td>
</tr>
<tr>
<td>10.17</td>
<td>N/A</td>
<td>REG_EN_OUT/reg_q_8:CLR</td>
<td>en&lt;8&gt;</td>
</tr>
</tbody>
</table>

### Section $Registers(CLK):ClockPins()$ to $Registers(CLK):InputPins(TmacEX_ASYNCPINS)$

<table>
<thead>
<tr>
<th>Delay (ns)</th>
<th>Slack (ns)</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.91</td>
<td>6.18</td>
<td>LMS_STATE_MACHINE/serio_state_h/serio_state_0Z1Z_0:CLK</td>
<td>ADD_SUB/accum_val_42:D</td>
</tr>
<tr>
<td>55.78</td>
<td>5.92</td>
<td>LMS_STATE_MACHINE/serio_state_h/serio_state_0Z1Z_0:CLK</td>
<td>ADD_SUB/accum_val_43:D</td>
</tr>
</tbody>
</table>

### Section $Registers(CLK):ClockPins()$ to $Registers(CLK):AsyncPins()$

--- No Paths found ---

### Section $Registers(CLK):AsyncPins()$ to $Registers(CLK):InputPins()$

<table>
<thead>
<tr>
<th>Delay (ns)</th>
<th>Slack (ns)</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.92</td>
<td>N/A</td>
<td>REG_H0/reg_q_7:CLR</td>
<td>ADD_SUB/accum_val_42:D</td>
</tr>
<tr>
<td>31.80</td>
<td>N/A</td>
<td>REG_H1/reg_q_14:CLR</td>
<td>ADD_SUB/accum_val_42:D</td>
</tr>
</tbody>
</table>