AN FPGA-BASED DIGITAL CONTROLLER FOR VIBRATIONAL MEMS GYROSCOPES

DAE HUI LEE

Bachelor of Electronics Engineering

Seoul National University of Technology, Republic of Korea

February, 2001

submitted in partial fulfillment of requirements for the degree

MASTER OF ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

December, 2006

This thesis has been approved
for the Department of Electrical and Computer Engineering
and the College of Graduate Studies by

_____

Dissertation Committee Chairperson, Dr. Zhiqiang Gao

_____

Department/Date

_____

Dr. Lili Dong

_____

Department/Date

_____

Dr. Ana Stankovic

_____

Department/Date

ACKNOWLEDGEMENTS

There are many people who gave me guidance and opportunities to help on this project and thesis, and I would like to express them my appreciation.

First of all, many thanks to my advisor, Dr. Zhiqiang Gao, for giving me the great opportunity to be a member of the Advanced Engineering Research Laboratory (AERL), and supporting me continually with his patience. I also thank Dr. Lili Dong for introducing this thesis topic and for her advice and guidance.

Second, I would like to thank Dr. Ana Stankovic for being on my committee and giving me evaluating my thesis.

Third, I would like to thank Qing Zheng and Zhan Ping. Qing Zheng helped me on the simulation part, and Zhan Ping allowed me to use the FPGA board, Reconfigurable Communication and Commands Module (RCCM) designed by Zhan Ping, in my research.

Fourth, I would like to thank Ivan Jurcic, Madhura Shaligram and David Avanesov who work in the AERL for helping and advising.

Finally, I would like to thank my wife, Etusko Takehisa, and my parents for their encouragement and unvarying support.

AN FPGA-BASED DIGITAL CONTROLER FOR VIBRATIONAL MEMS GYROSCOPES

DAE HUI LEE

ABSTRACT

A digital controller, implementing discrete Linear Active Disturbance Rejection Control, using the Field Programmable Gate Array (FPGA) and Very high speed integrated circuit Hardware Description Language (VHDL), was successfully designed and prototyped. It is tested in the high performance position control system for the drive axis of a vibrating Micro-Electro-Mechanical System (MEMS) gyroscope, designed to measure the rotation rate. Based on the novel Current Discrete Extended State Observer, the states, as well as the unknown disturbance, of the MEMS gyroscope are estimated in real time, providing the ground for active disturbance rejection. A single precision floating-point based on IEEE standard 754 was used in numerical computation to avoid inaccuracy in integer arithmetic. The controller was first simulated in MATLAB/ Simulink, followed by implementation on a FPGA based controller board with a sampling rate of 1 *MHz*. A graphical user interface was also developed using JAVA. The experimental results are quite encouraging: the position signal in the drive axis of the MEMS gyroscope follows a 10.1 *KHz* sinusoidal reference with the steady state error of

10 % or less and a settling time of about 20 *ms*. This is achieved without a detailed mathematical model of the plant.

TABLE OF CONTENTS

Page

NOMENCLATURE

ADC:	Analog to Digital Converter
DAC:	Digital to Analog Converter
FIFO:	First In First Out
ESO:	Extended System Observer
FPGA:	Field Programmable Gate Array
LADRC:	Linear Active Disturbance Rejection Control
MEMS:	Micro-Electro-Mechanical Systems
VHDL:	Very high speed integrated circuit Hardware Description Language

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Background

Gyroscopes are sensors that measure angular rotation rate. They are used widely in high-precision applications such as navigation, homing and stabilization [1]. A mechanical gyroscope is shown in Figure 1. Usage of gyroscopes was limited to applications such as aviation sensors for space exploration and for military applications because they were large and expensive [2]. Recent micro-machining technology, however, enables gyroscopes and other inertial measurement devices to be manufactured inexpensively in a very small package on micro scale [1]. Figure 2 shows an integrated MEMS gyroscope from Analog Devices [3]. Because of small size and low cost, the use of MEMS gyroscope is broadened into many different fields and applications such as Global Positioning System (GPS) assisted inertial navigation, virtual reality systems, interactive pointing devices, image stabilization and rollover detection with simple implementations [1].

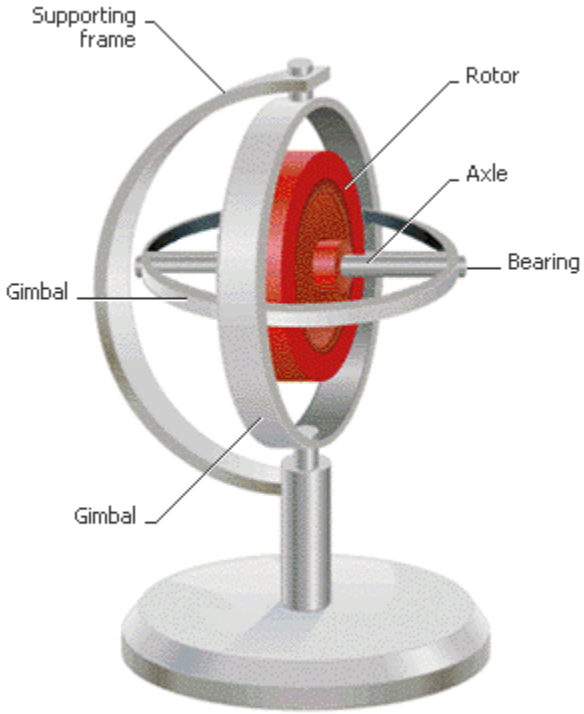


Figure 1 Mechanical Gyroscope

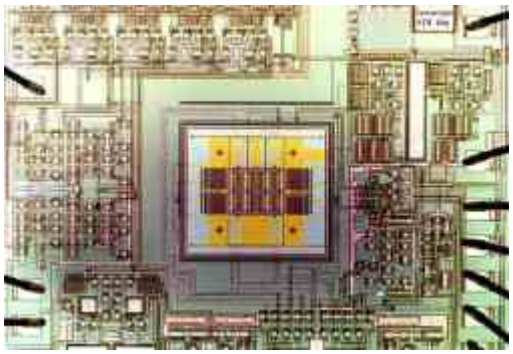


Figure 2 Integrated MEMS Gyroscope (Analog Devices Photo)

On the other hand, because of the imperfections of the MEMS gyroscope such as Zero Rate Output (ZRO), quadrature error and temperature effect, a feedback control system may be used to obtain accurate results. Feedback control is a well known solution in attaining consistent performance in a system consists of unreliable parts. The hope is

that such controller will be able to handle the imperfections and unknown variations in the MEMS gyroscopes.

1.2 Existing Technology and Motivation for New Research

The imperfections in MEMS gyroscopes and their impact on performance, stability and robustness of the sensor are well documented in [4]. This underlines a strong need for a well designed control system to compensate for the imperfections. On the other hand, MEMS gyroscope is a relatively recent technology. Even though there is much research on the MEMS gyroscope, yet there is no acceptable high precision control method for commercial applications of the MEMS gyroscope.

There has been much research done to improve the performance of the MEMS gyroscope. It can be categorized in two different strategies to improve accuracy: 1) enhancing the structural design and fabrication; and 2) implementing a high performance feedback control system. In [4], the research shows that multiple drive-mode oscillators increase bandwidth of the resonance frequency in the drive axis of the MEMS gyroscope, and this implementation improves robustness to structural and thermal parameter fluctuations. In [5], a three degree-of-freedom gyroscope system with two degree-of-freedom sense-mode oscillator is proposed to improve robustness against structural parameter variations. These structural improvements will yield improved performance advantages at additional cost. But such improvement is quite limited due to the inherent uncertainties in the MEMS gyroscopes.

Another way to overcome imperfections in MEMS gyroscope and to achieve good performance is to use control systems. There are two different control system designs: open-loop control and closed-loop control. A comparative study of these two methods is presented in [6]. In the open loop mode, the MEMS gyroscope is driven be a simple input signal. An improvement is made by pre-shaping the input signal using the the information of the gyroscope dynamics. The ultimate step, as shown in [6], to improve the precision and speed is the introduction of feedback control. It is shown that while the open-loop control is simple to implement, it does not handle unknown variations adequately. On the other hand, faster response and higher voltages requirement dictate higher precision in control system, leaving closed-loop as the only choice. Also shown is that the closed-loop controlled gyroscope is significantly less sensitive to changes in the system parameters.

With closed-loop control as the method of choice, we now examince different designed techniques based on force-balancing feedback control [7]-[9] and adaptive control [1], [10]-[14]. The force-balancing control strategy for MEMS gyroscope, as introduced in [7, 8], was based on an earlier technique developed for MEMS accelerometer [9]. The basic idea is to drive the position, velocity and acceleration in the sense axis to zero so that the quadrature error and the Coriolis acceleration can be easily obtained through demodulation. The challenge is that the system dynamics varies during operation and that high resolution is difficult to obtain in angulate estimation. To accommodate such uncertain and time-varying nature of the MEMS gyroscope dynamics, the prevailing closed-loop solutions are dominated by adaptive control. A PI-like adaptive controller for the drive axis of the MEMS gyroscope and a force-to-rebalance

control for the sense axis are proposed [1]. It is compared favorably to the more complex Lyapunov based adaptive control method in [10], for both the drive axis and sense axis. The PI-like adaptive controller is also implemented and tested in an analog circut to control the drive axis of the MEMS gyrosocpe. An adaptive control system was proposed in [11] that adds an adaptive outer loop to the conventional force-balancing scheme. The parameter adaptation algorithm helps to estimate the angular rate and to compensate for the quadrature error. Anotehr adaptive controller is proposed in [12] for tuning the natural frequency of the drive axis and replacing the conventional phase-locked loop. It results in the reduction of sensitivity to imprecise fabrication as well as the effects of nonlinearities and uncertainties. An analog circuit implementation for the drive axis of the MEMS gyroscope is built and tested successfully. Finally, a hybrid discrete/continuous time version of the observer-based adaptive control is introduced in [13]. Here, the parameter adaptation, feedback control and feedfoward control are implemented digitally, but velocity observer remains in continuous time domain. Compare to the original analog adaptive controller in [14], this discrete implementation retains the advantages of wide bandwidth, absence of ZRO, self-calibration and relative robustness to parameter variations [13].

Adaptive control is a control system capable of accomodating the parametric changes in plant dynamics. Unfortunately, this does not completely resolve the difficulties in MEMS gyroscopes due to its fabrication imperfections and other uncertainties. Most existing adaptive control solutions are based on linear models of the plant with the assumptions that the dynamic variations are small and slow enough for the adaptation algorithm to catch up with. In reality, however, the MEMS gyroscope

dynamics is quite nonlinear, complex and uncertain, to the extent far beyond the limited range of adaptive control. Addressing the large amount of uncertainty and the lack of accurate model in MEMS gyroscopes is the key technical issue, which is the focus of this thesis. To this end we propose the Active Disturbance Rejection Control (ADRC), as a possible solution.

1.3 Active Disturbance Rejection Control

The ADRC methodology was introduced by Han [15, 16]. The basic idea is that the uncertainties such as the external force and the internal dynamics in a plant can be treated as disturbance, which can be estimated, in real time, based on the input-output data of the plant. The estimation, in turn, is used to compensate for disturbance and to reduce the plant to the cascade integrators form. Thus, a complex, unknown, plant is reduced to a simple, invariable, form, which is easy to control. Han's original nonlinear ADRC was simplified and parameterized in [17], where the controller parameters are all made an explicit function of one tuning variable: the closed-loop bandwidth. This greatly simplified the ADRC and made its tuning just as easy as that of PID, if not more so.

ADRC seems to be a perfect fit for MEMS gyroscope problems because it does not require an accurate mathematical model to begin with and it is specifically designed to handle uncertainties. It is for this reason that ADRC has been successfully applied to many different applications such as DC-DC power controller [18], power managemint and distribution system [19], health and fault monitoring [20], industrial motion control

[21], and so on. But most of these applications are relatively slow, compare to MEMS gyroscope applications and the controller is implemented in DSP or microcontrollers chips with sufficient speed. In MEMS applications, much high speed is required which poses both software and hardware challenges. Software wise, such speed requirement dictates that the controller be implemented in a most efficient way, both in terms of computation time and phase lag. Much effort in this thesis is devoted to this issue. Hardware wise, the sampling rate for MEMS devices is in the megahertz range, thus excluding most DSP and microcontrollers options. FPGA seems to be the only remaining candidate, as discussed below.

1.4 Field Programmable Gate Array

A Field Programmable Gate Array (FPGA), which is a semiconductor device, can embody programmable logic components and programmable interconnects. In designing the logic components, the functionality of basic logic gates or more complex combinational functions can be duplicated and programmed [22]. Because of FPGA's flexbility, there are many applications such as Digital Signal Processing (DSP), software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer viosion, speech recognition, cryptography, bioinformatics, computer hardware emulation and a growing range of other areas. The behavior of the FPGA can be defined by Hardware Description Language (HDL) and schematic design.

To implement advanced control algorithm, an FPGA board designed by Zhan Ping [19] was chosen. The circuitry is very well compressed in a small and lightweight board, and has many useful and flexible functions such as Field Programmable Gate Array (FPGA), Field Programmable Analog Array (FPAA), Ethernet, Control Area Network (CAN) and serial communication port RS-232. Figure 3 shows the FPGA board. The initial purpose of this board was to control DC-DC power converters in spaceships, a very extreme environment. This board has Altera Stratix EP1S40F780C6 for FPGA, AnaDigm An221E04 for FPAA, LAN91C111I-NE, AM29LV065DU for flash memory and AD7266 (Dual 12 bit, 3-channel ADC).

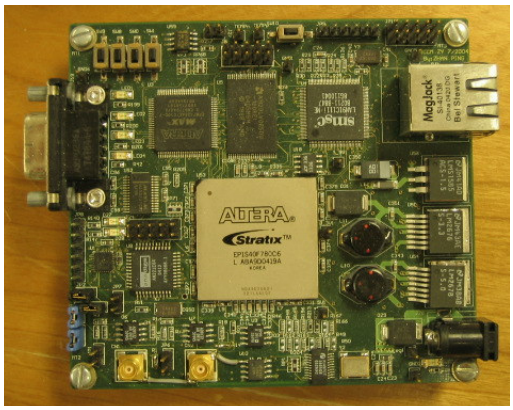


Figure 3 FPGA RCCM board

The Altera Stratix EP1S40F780C6 has 41,250 Logic Elements (LEs), 3,423,744 total RAM bits, 14 DSP blocks, 112 embedded multipliers, 12 PLLs and 615 user I/O pins [23]. For the FPGA clock input, 50 *MHz* is implemented in the FPGA board.

1.5 Thesis Organization

This thesis is organized as following: the background information on the MEMS vibrating gyroscope, such as the operating principles, the Coriolis effect and a model of the MEMS gyroscope, is presented in Chapter 2. Description and implementation of ADRC are introduced in Chapter 3, together with simulation result. Hardware implementation in the FPGA circuit is described in Chapter 4. In Chapter 5, the hardware test results are shown. Finally, the conclusions and future work are presented in Chapter 6.

CHAPTER II

VIBRATING MEMS GYROSCOPE

To understand the control subject better, details of the vibrating MEMS gyroscope is explained in this chapter. First, operation of the MEMS gyroscope is illustrated, and the Coriolis effect is explained. Second, a model of the vibrating MEMS gyroscope is built in mathematical terms. Third, system dynamics, fabrication imperfections and environmental variations which limit the performances of MEMS gyroscope are discussed. Finally, design object is discussed.

2.1 MEMS Gyroscope Operating Principle

The vibrating MEMS gyroscope is operated by energy transfer from the drive axis to the sense axis. The energy transfer is caused by Coriolis effect. This can be explained as a mass attached to a rigid frame by spring shown in Figure 4. Here, three axes, drive axis (x), sense axis (y) and rotation axis (z) are used to explain vibrating MEMS gyroscope. Each of the axes is perpendicular to the other two axes. The mass is vibrated along the drive axis (x) by resonance. If the gyroscope is rotating along the rotation axis

(z), a Coriolios force will be generated along the sense axis(y). Then, the sense axis (y) vibration indicates the rotation rate.

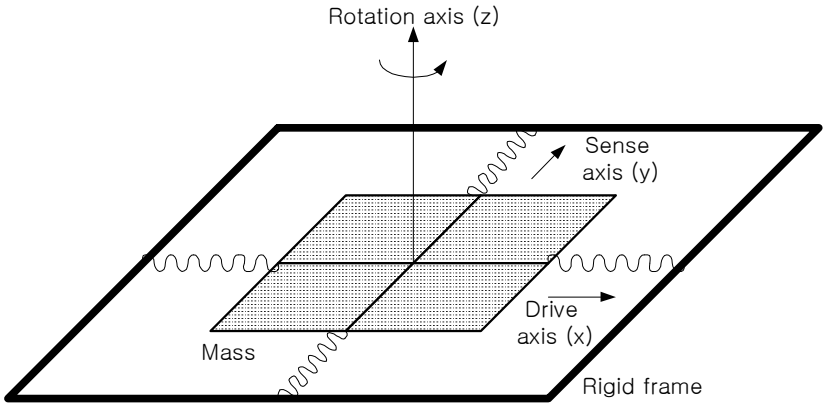


Figure 4 Vibrating Gyroscope [1]

To understand the vibrating gyroscope operating principle better, it is necessary to understand the Coriolis effect, named after the French scientist and engineer G. G. de Coriolis (1792-1843). Figure 5 illustrates the Coriolis effect for better understanding.

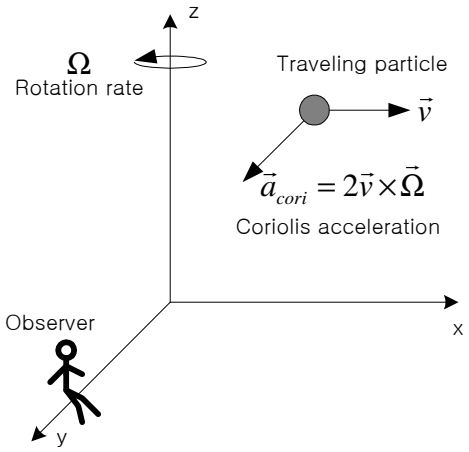


Figure 5 The Coriolis Effect

Here, a particle travels in space with a velocity vector $\vec{v}$. Then, the particle can be observed in y-axis of the x-y-z coordinate system. If the coordinate system is rotating around z-axis with an angular rate Ω, the observation in y-axis is that the traveling particle changes its trajectory toward the y-axis with an acceleration equal to $2\vec{v}\times\vec{\Omega}$. The acceleration is called Coriolis acceleration. The Coriolis acceleration is the energy transfer from the x-axis to the y-axis.

2.2 Model of Gyroscope Drive Axis

As Figure 6 shows, the Z-axis MEMS gyroscope was built as a prototype by MEMS research group of The University of Alabama. To build the model of the vibrating MEMS gyroscope, the governing equation of a Z-axis MEMS gyroscope can be expressed as equations (2.1) and (2.2) [1].

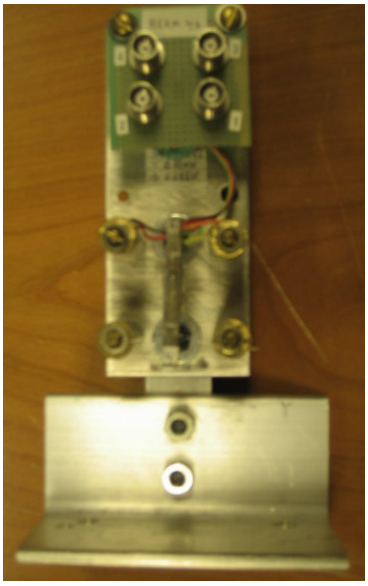


Figure 6 Picture of Vibrating MEMS Gyroscope

$$\ddot{x} + \frac{\omega_n}{Q_x}\dot{x} + \omega_n^2 x + \frac{k_{xy}}{m} y + \frac{d_{xy}}{m}\dot{y} - 2\Omega\dot{y} = \frac{1}{m} u_{drive}(t) \tag{2.1}$$

$$\ddot{y} + \frac{\omega_y}{Q_y}\dot{y} + \omega_y^2 y + \frac{k_{xy}}{m} x + \frac{d_{xy}}{m}\dot{x} - 2\Omega\dot{x} = \frac{1}{m} u_{sense}(t) \tag{2.2}$$

where

$x(t)$: drive axis displacement

$y(t)$: sense axis displacement

Ω: rotation rate about z-axis

u_{drive}: control input to be designed for drive axis

u_{sense}: control input to be designed for sense axis

m: mass of vibrating element

$2\Omega\dot{x}$ and $2\Omega\dot{y}$: Coriolis accelerations which can be used to measure the rotation rate Ω

Q_x and Q_y: quality factors $\left(Q_x = \frac{1}{2\zeta}, Q_y = \frac{1}{2\zeta_y} \right)$

k_{xy}: fabrication imperfections caused mainly by the asymmetric spring

d_{xy}: fabrication imperfections caused mainly by the damping coupling terms

For most MEMS gyroscopes, the mass is in the range of 10^{-6} ~ 10^{-10} *Kg*, the resonant frequency is 2 ~ 30 *KHz*, and the quality factor range is 10 ~ 10^4. k_{xy} and d_{xy} are undesirable coupling in stiffness and damping terms between x-axis and y-axis [1].

We disregard the damping coupling term. Then, the equations (2.1) and (2.2) become equations (2.3) and (2.4) respectively.

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x + \omega_{xy} y - 2\Omega\dot{y} = \frac{K}{m} u_{dd}(t) \tag{2.3}$$

$$\ddot{y} + 2\zeta_y\omega_y\dot{y} + \omega_y^2 y + \omega_{xy} x + 2\Omega\dot{x} = \frac{K}{m} u_{ss}(t) \tag{2.4}$$

In this thesis, only the drive axis of the MEMS gyroscope is considered to be controlled. For this reason, sense axis variations can be assumed zero in the equation (2.3). Then, the equation (2.5) can replace the equation (2.3) for drive axis of gyroscope.

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = \frac{K}{m} u_{dd}(t) \tag{2.5}$$

The constants in equation (2.5) for the vibrational beam gyroscope shown in Figure 6 are $\zeta = 0.0005$, $\omega_n = 63881.1$ *rad/s* and $\frac{K}{m} = \frac{11.1}{24484769.6}$. Then, the equation (2.5) becomes the equation (2.6).

$$\ddot{x} = -63.8811\dot{x} - 4.0807947 \times 10^9 x + 2.7178 \times 10^8 u_{dd}(t) \tag{2.6}$$

Now, the transfer function of the MEMS gyroscope can be expressed as equation (2.7).

$$H(s) = \frac{X(s)}{U_{dd}(s)} = \frac{2.7178 \times 10^8}{s^2 + 63.8811s + 4.0807949 \times 10^9} \tag{2.7}$$

2.3 Disturbances of MEMS Gyroscope

Due to fabrication imperfections and environmental variations of the MEMS gyroscope, undesirable disturbances are caused. The common fabrication steps of the MEMS gyroscope are: deposition, etching, patterning and electroplating of materials [24]. Each step of fabrication could cause imperfections in the MEMS gyroscope such as asymmetrical structures, actuation mechanism misalignment and deviations of mass center from the geometric center. The fabrication imperfections bring undesirable results and degraded performance of a MEMS gyroscope by systematic perturbations in the form of mechanical and electrostatic forces.

Second, quadrature error is caused by coupling terms between the drive axis and the sense axis and misalignment of the sensor and actuator. It is called quadrature error because the quadrature error and the Coriolis deflection have a 90^o phase. The quadrature

error is caused by a large vibration of the sense axis in Zero Ratio Output (ZRO) caused by electrical coupling between the sense and drive electrodes in geometrical imperfections.

Finally, temperature can cause undesirable results in the MEMS gyroscope. As temperature changes, elasticity modulus changing, thermal expansion and thermal stress can decouple the system frequency.

Even though the model of the MEMS gyroscope was built in section 2.2, unknown variables and disturbances still exist. The control system for the vibrating MEMS gyroscope must be able to handle problems such as parameter variations, quadrature errors, natural frequency mismatches between drive and sense axes, and thermal noise.

2.4 Design Objective

The design objective of this thesis is to design a digital control system that drives the drive axis of the MEMS gyroscope to a resonance frequency accurately, and maintains the output amplitude of the drive axis to a set value. To enhance the Coriolis output and to conserve phase synchronization, the drive axis of the MEMS gyroscope must be driven to the resonance frequency which is the sharp resonant peak [12]. This digital control system implements the ADRC using FPGA to control the drive axis of MEMS gyroscope.

According to the hardware results from the analog adaptive controller [12], this controller took about 3 seconds to reach the steady state of the drive axis output, and a 15 % error was introduced. In the outcome of this research, the FPGA based digital implementation of ADRC is to be compared to the analog adaptive control method in [12].

My goal in this project is to design an FPGA digital control system that controls the drive axis of the MEMS gyroscope with faster response and less error than the controls have previously achieved. Then hardware results will be compared with the simulation results and the analog adaptive control introduced in [12].

CHAPTER III

ADRC ANALYSIS AND IMPLEMENTATION

As earlier mentioned, because of unknown variations and disturbance, discrete Linear Active Disturbance Rejection Control (LADRC) is chosen to control the drive axis of the MEMS gyroscope. In this chapter, analysis and implementation of Active Disturbance Rejection Control (ADRC), introduced by Han [15, 16], will be explained. ADRC has two parts: ESO and PD controller. Then, ADRC will be made discrete by Zero-Order-Holder (ZOH). To implement discrete LADRC, Predictive Discrete Extended System Observer (PDESO) and Current Discrete Extended System Observer (CDESO) are introduced to estimate the accurate MEMS gyroscope input and output data in real time [25]. Simulation results of the control implementation, discrete LADRC, will also be illustrated using MATLAB Simulink.

3.1 Active Disturbance Rejection Control

Considering the governing equation of a Z-axis MEMS gyroscope, the equation (2.1) can be rewritten as equation (3.1).

$$\ddot{x} = f(t, x, \dot{x}, w) + bu \tag{3.1}$$

where x is output, u is input, and w is disturbance. Assuming $b \approx b_o$, Then, equation (3.1) can be equation (3.2).

$$\ddot{x} = f + b_o u \tag{3.2}$$

where f is short for $f(t, x, \dot{x}, w)$. If f in the equation (3.2) can be estimated, this can be a simple double integral as equation (3.3) as long as $\hat{f}$, the estimate of f, can be fast enough to estimate f accurately.

$$\ddot{x} = u_o \tag{3.3}$$

where

$$u = \frac{(-\hat{f} + u_o)}{b_o} \tag{3.4}$$

Let $\xi_1 = x$, $\xi_2 = \dot{x}$, $\xi_3 = f$ and $\xi = [\xi_1 \ \ \xi_2 \ \ \xi_3]^{\mathrm{T}}$. Assuming f is differentiable, the state space form of equation (3.1) is equation (3.5).

$$\begin{cases} \dot{\xi} = A\xi + Bu + Eh \\ x = C\xi \end{cases} \tag{3.5}$$

where

$$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix},\quad B=\begin{bmatrix}0\\ b_o\\ 0\end{bmatrix},\quad C=\begin{bmatrix}1 & 0 & 0\end{bmatrix},\quad E=\begin{bmatrix}0\\ 0\\ 1\end{bmatrix} \tag{3.6}$$

In equation (3.5), $h = \dot{f}$ as unknown disturbance. Then, ESO for equation (3.5) can be designed as equation (3.7).

$$\begin{cases}\dot{\hat{\xi}} = A\hat{\xi} + Bu + L(x-\hat{x})\\ \hat{x} = C\hat{\xi}\end{cases} \tag{3.7}$$

With the observer gain, equation (3.8), the state of ESO, $\hat{\xi}$, can estimate the state of the plant, ξ, and f.

$$L=\begin{bmatrix}l_1 & l_2 & l_3\end{bmatrix}^T \tag{3.8}$$

The observer gain can be chosen as

$$l_1 = 3\omega_o,\quad l_2 = 3\omega_o^2,\quad l_3 = \omega_o^3 \tag{3.9}$$

from the characteristic polynomial of the observer as equation (3.10).

$$\lambda_o(s) = s^3 + l_1 s^2 + l_2 s + l_3 = (s+\omega_o)^3 \tag{3.10}$$

This linear ESO makes tuning very easy [17] because there is only one tuning parameter, observer bandwidth ω_o.

3.2 Discrete Extended State Observer

Now, the system (3.5) can be expressed in discrete form by ZOH.

$$\begin{cases} \xi(k+1) = \Phi\xi(k) + \Gamma u(k) \\ x(k) = H\xi(k) \end{cases} \tag{3.11}$$

where

$$\Phi = e^{AT_s},\ \Gamma = \int_0^{T_s} e^{A\eta} d\eta B$$

T_s is the sampling time. An observer can estimate the states. For the stability analysis of the ESO, Predictive Discrete Extended State Observer and Current Discrete Extended State Observer are considered according to [26].

3.2.1 A Predictive Discrete Extended State Observer (PDESO)

A PDESO can be written as equation (3.12) according to [26].

$$\bar{\xi}(k+1) = \Phi\bar{\xi}(k) + \Gamma u(k) + L_p[x(k) - H\bar{\xi}(k)] \tag{3.12}$$

Let current estimated error be $\tilde{\xi}(k) = \bar{\xi}(k) - \xi(k)$. Then, the predicted next state error can be equation (3.13).

$$\tilde{\xi}(k+1) = (\Phi - L_p H)\tilde{\xi}(k) \tag{3.13}$$

For this project,

$$\Phi = \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix},\ \Gamma = \begin{bmatrix} \frac{T_s^2 b_o}{2} \\ T_s b_o \\ 0 \end{bmatrix},\ H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},\ T_s = \frac{1}{1000000} \tag{3.14}$$

3.2.2 A Current Discrete Extended State Observer (CDESO)

A CDESO, which is implemented for this project, can be written as equation (3.15) according to [26].

$$\begin{cases} \hat{\bar{\xi}}(k) = \bar{\xi}(k) + L_c[x(k) - H\bar{\xi}(k)] \\ \bar{\xi}(k) = \Phi\hat{\bar{\xi}}(k-1) + \Gamma u(k-1) \end{cases} \tag{3.15}$$

Equation (3.15) yields equations (3.16) and (3.17).

$$\bar{\xi}(k+1) = \Phi\bar{\xi}(k) + \Gamma u(k) + \Phi L_c[x(k) - H\bar{\xi}(k)] \tag{3.16}$$

$$\tilde{\bar{\xi}}(k+1) = (\Phi - \Phi L_c H)\tilde{\bar{\xi}}(k) \tag{3.17}$$

The desired roots are supposed to be $z = \beta_1, \beta_2, \ldots, \beta_n$, where $\beta_i = e^{-\omega_o T_s}$, and ω_o is the observer bandwidth. Then the characteristic polynomial is equation (3.18).

$$\alpha_c(z) = (z - \beta_1)(z - \beta_2)\ldots(z - \beta_n) \tag{3.18}$$

According to Ackerman's Formula [26]

$$L_c = \alpha_c(\Phi)\begin{bmatrix} H\Phi \\ H\Phi^2 \\ . \\ . \\ . \\ H\Phi^n \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1 \end{bmatrix} \tag{3.19}$$

From equation (3.19), L_p can be as equation (3.20).

$$L_p = \Phi L_c \tag{3.20}$$

In this case, L_c can be as equation (3.21).

$$L_c = \begin{pmatrix} 1-\beta^3 \\ \dfrac{1.5(\beta-1)^2(\beta+1)}{T_s} \\ -\dfrac{(\beta-1)^3}{T_s^2} \end{pmatrix} \tag{3.21}$$

Finally, Figure 7 shows the CDESO block diagram.

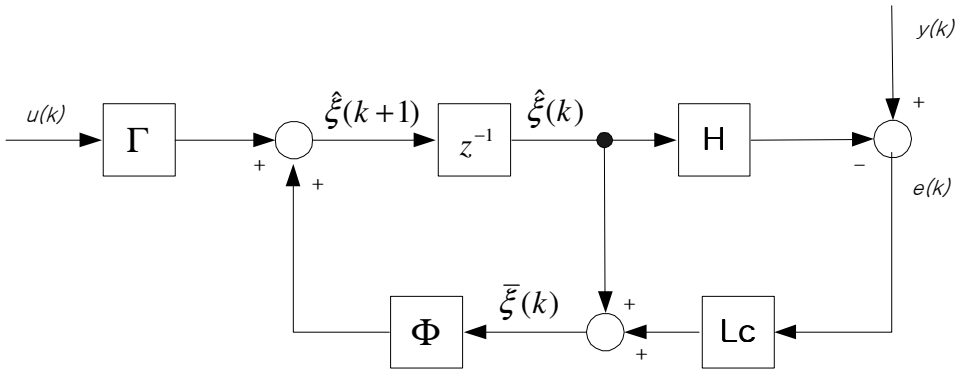


Figure 7 Current Discrete ESO Block Diagram

3.3 PD Control Algorithm

The estimated disturbance, *f(k),* can be canceled out from the feedback of the CDESO estimates, $\hat{\xi}_3(k)$. Considering equation (3.4), discrete PD control law can be as equation (3.22).

$$u_o = k_p(r(k) - \hat{\xi}_1(k)) - k_d\hat{\xi}_2(k) \tag{3.22}$$

where

$k_d = 2\omega_c$, $k_p = \omega_c^2$

ω_c: the trajectory bandwidth

r: the desired trajectory of the drive axis

In this control system, the controller continually traces the reference which is a sinusoidal signal. For this reason, tracking error could cause a delay. To reduce the tracking error, a feedforward PD control is implemented as equation (3.23).

$$u_o(k) = k_p(r(k) - \hat{\xi}_1(k)) + k_d(\dot{r}(k) - \hat{\xi}_2(k)) + \ddot{r} \tag{3.23}$$

Then, equation (3.4) becomes equation (3.24).

$$u(k) = \frac{u_o - \hat{\xi}_3}{b_o} \tag{3.24}$$

3.4 Simulation Results

A control system of discrete LADRC is designed and simulated with a model of a piezoelectrically-driven vibrational beam gyroscope [27], and built on the MATLAB/ Simulink block diagram in Figure 8.

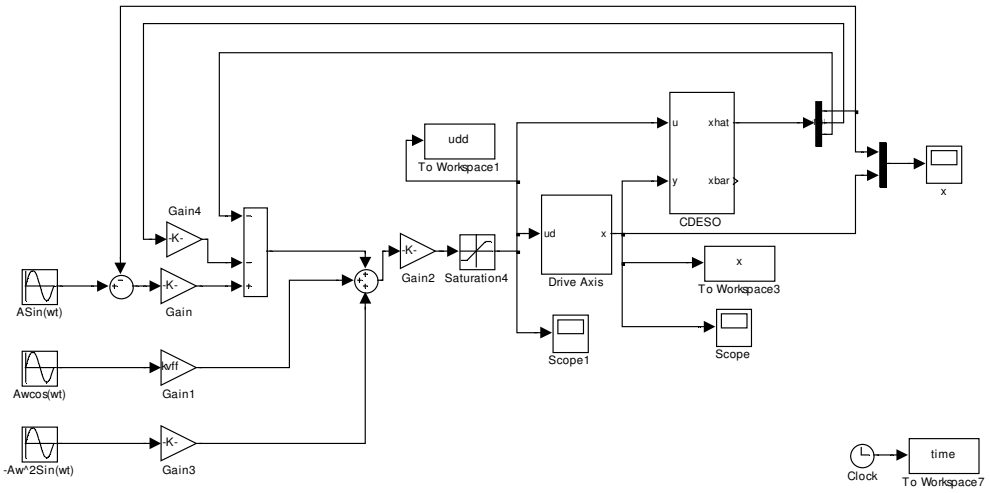


Figure 8 Discrete LADRC MATLAB/Simulink Block Diagram

The parameters for the simulation are ω_n = 63881.1 *rad/sec*, ς = *0.0005,* ω_{xy} = 6000 rad^2/sec^2, $\frac{K}{m} = \frac{11.1}{4.084\times10^{-8}}$. The reference signal for the drive axis is $r_x = A\cos(\omega t)$, where ω = 63428 *rad/sec* and A = 200 *mV*. We use A = 215 in "simulation units" to represent this. The output of the control signal is limited to ±100. The design parameter $b_o = \frac{K}{m} = 2.7178\times10^8$; the controller bandwidth is $\omega_c = 5 \times 10^5$ *rad/sec*; the observer bandwidth is $\omega_o = 2.5 \times 10^6$ *rad/sec*; and the sampling period is $T_s = 1 \times 10^{-8}$ *s*. The output of a MEMS gyroscope drive axis is as Figure 9.

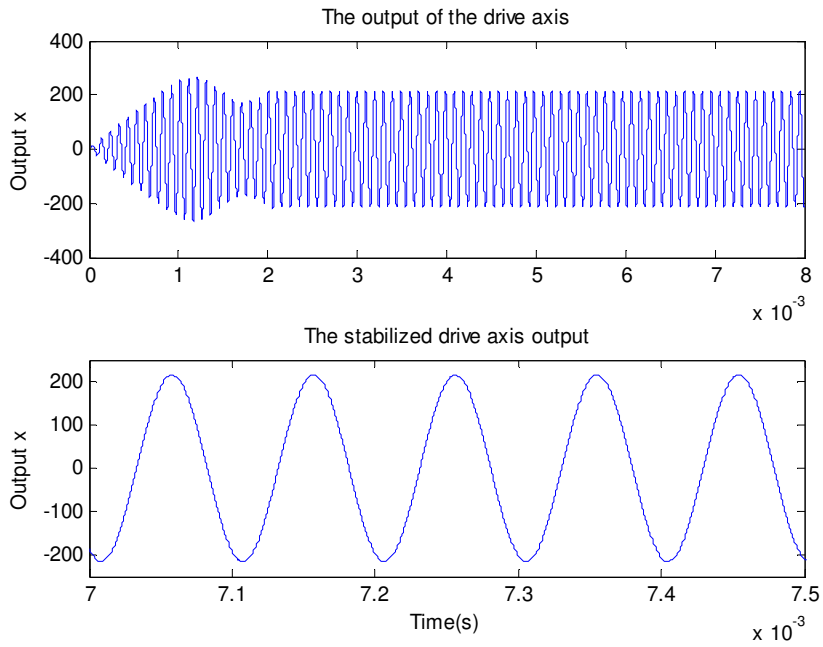


Figure 9 The output of the drive axis with DLADRC ($T_s = 1 \times 10^{-8}$)

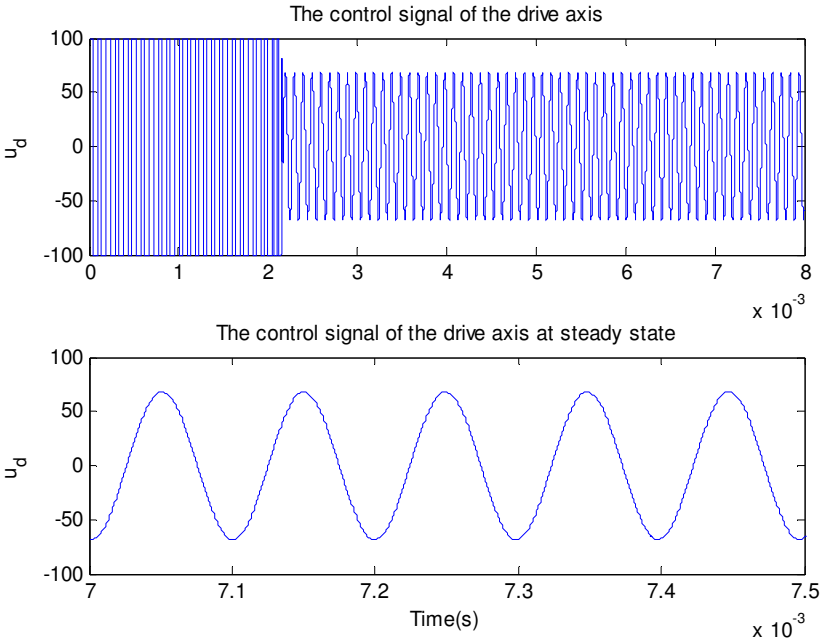


Figure 10 The control signal of the drive axis with DLADRC ($T_s = 1 \times 10^{-8}$)

As the result shows, the frequency of the drive axis is driven to the resonant frequency ω after 2.2 *ms*. Figure 10 shows the control signal of the drive axis with discrete ADRC. The stabilized peak error is less than 0.05 % of the desired amplitude, and the error between the reference signal and the drive axis output is 0.2% as Figure 11.

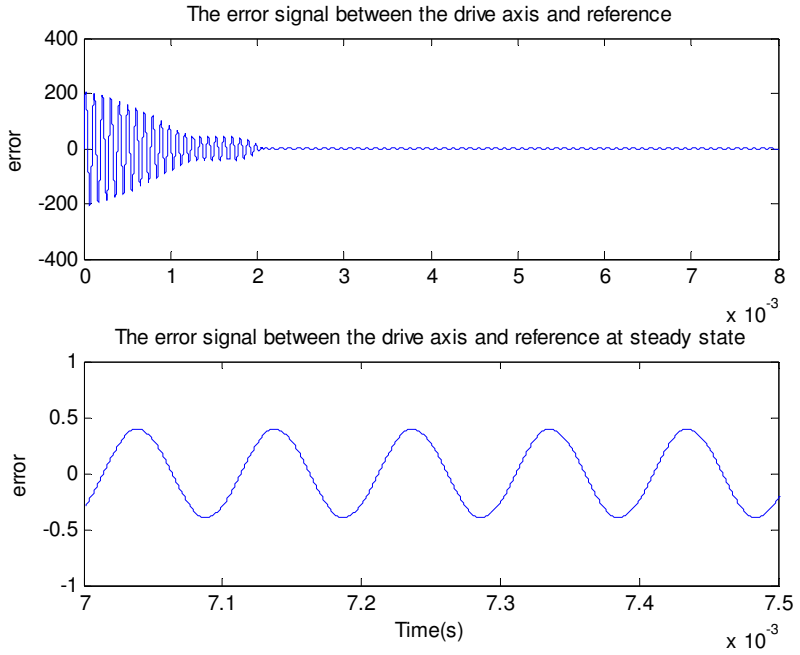


Figure 11 The Error (the Drive Axis and the Reference ($T_s = 1 \times 10^{-8}$))

The sampling time, however, is too fast to implement into the FPGA board, since the FPGA board clock speed is 50 *MHz*, and the digital ADRC requires several clock cycles to process. For this reason, the design parameters are adjusted as the controller bandwidth is $\omega_c = 2.5 \times 10^6$ *rad/sec*, the observer bandwidth is $\omega_o = 2.5 \times 10^6$ *rad/sec* and the sampling period is $T_s = 1 \times 10^{-6}$ *s*. The output of a MEMS gyroscope drive axis is as Figure 12.

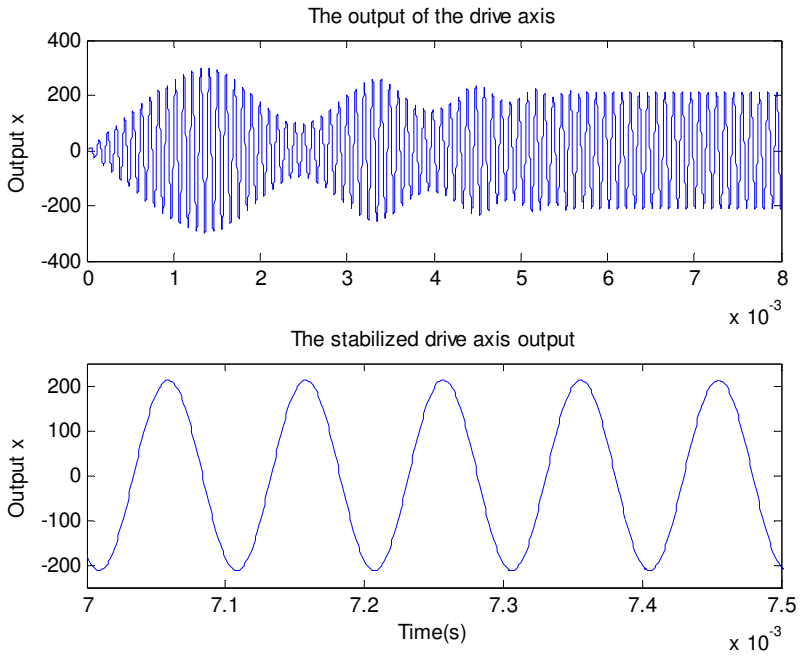


Figure 12 The output of the drive axis with DLADRC ($T_s = 1 \times 10^{-6}$)

As the result shows in Figure 12, the frequency of the drive axis is driven to the resonant frequency ω after 5.5 *ms*. Figure 13 shows the control signal of the drive axis with discrete ADRC when the sampling time is 1×10^{-6} *s*. The stabilized peak error is 1.4 % of the desired amplitude, and the error between the reference signal and the drive axis output is 3.72 % as Figure 14. The results from two different sampling times, 1×10^{-6} *s* and 1×10^{-8} *s,* show that the sampling time is important for obtaining optimum results.

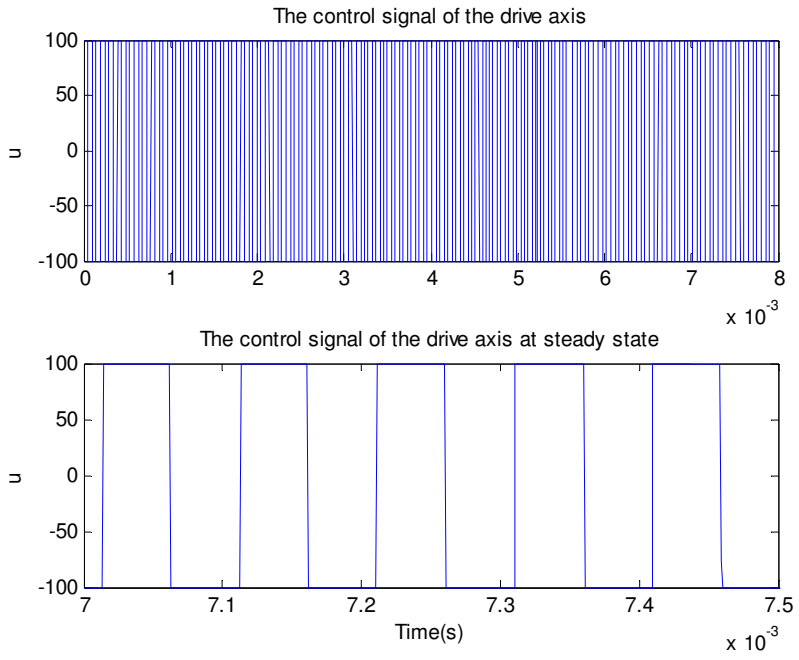


Figure 13 The control signal of the drive axis with DLADRC ($T_s = 1 \times 10^{-6}$)

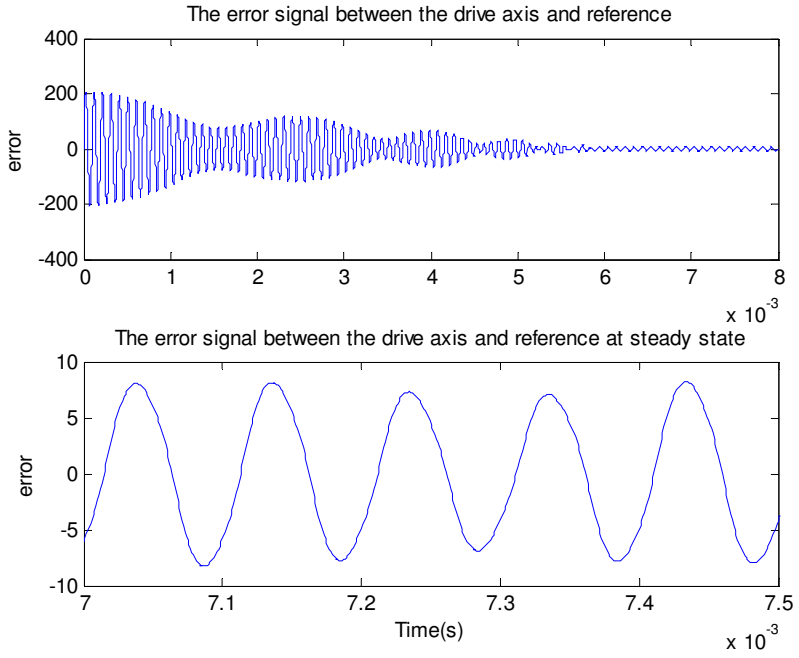


Figure 14 The Error (the Drive Axis and the Reference ($T_s = 1 \times 10^{-6}$))

CHAPTER IV

HARDWARE IMPLEMENTATION

Due to the flexibility of FPGA, the discrete control system can be implemented into FPGA. In the Figure 15, several blocks are employed for the FPGA digital logic. First of all, Phase-Locked Loop (PLL) generates four different clocks, 15 *MHz* for ADC, 25 *MHz* for DAC, 1 *MHz* for reference generator and 300 *KHz* for FIFO buffer. Second, reference generator generates discrete sinusoidal signal for the resonance frequency of MEMS gyroscope. Third, ADC and DAC controllers control the ADC and DAC chips directly, and deliver the signal data to the controller. Fourth, a processor block initializes the system, changes control parameters, and interfaces between peripherals and computer. Finally, a digital control circuit is employed for the main control algorithm to control the MEMS gyroscope.

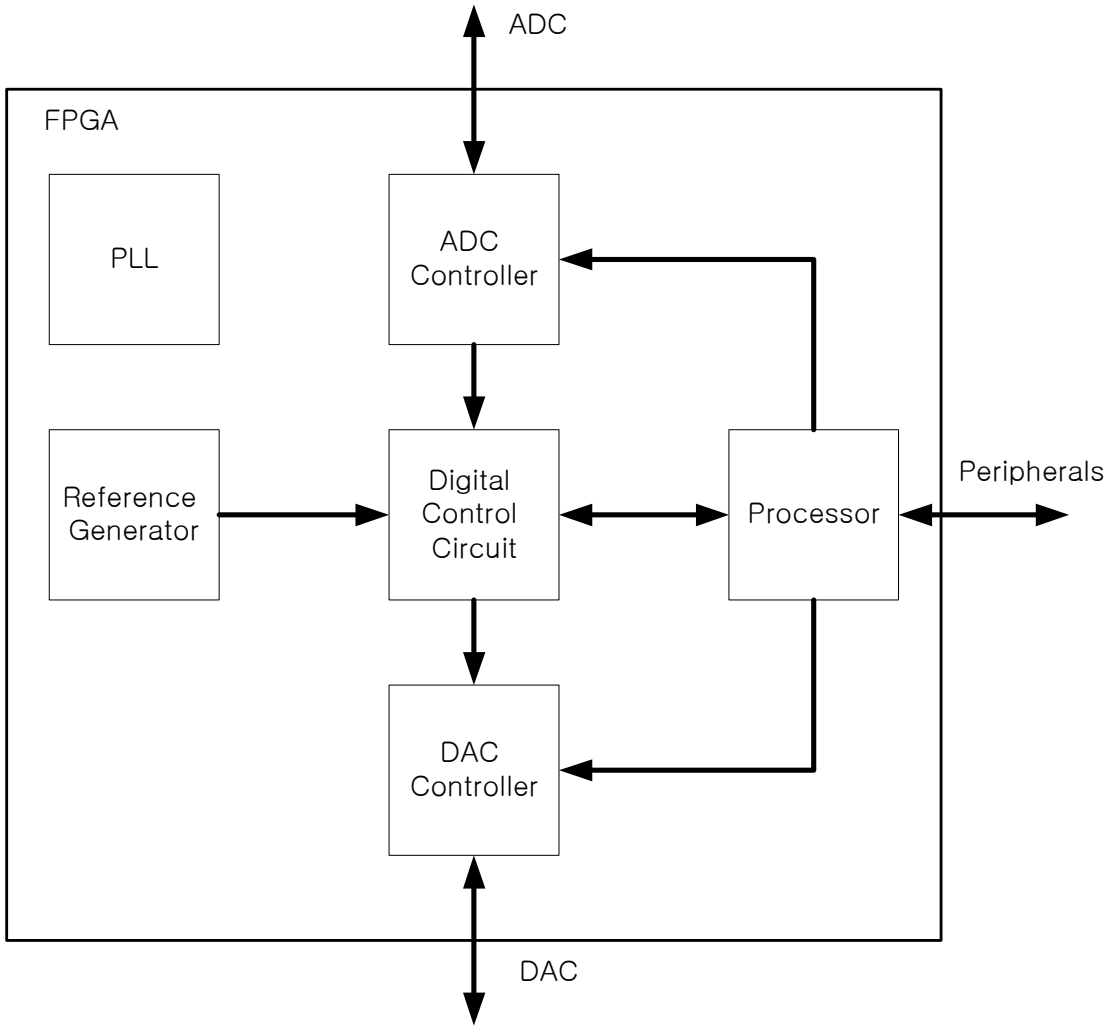


Figure 15 Digital Logic Structure in FPGA

Figure 16 shows the block diagram of the FPGA hardware and circuits for the control system. For this project, LADRC by VHDL, Nios core processor, discrete sinusoid reference signal and First Input First Output buffer (FIFO) are designed into FPGA. 12 bit DAC is implemented to convert a digital signal from FPGA to analog control signal to control the gyroscope. Then, an op-amp circuit is built to amplify the gyroscope output signal which is the feedback signal of the controller.

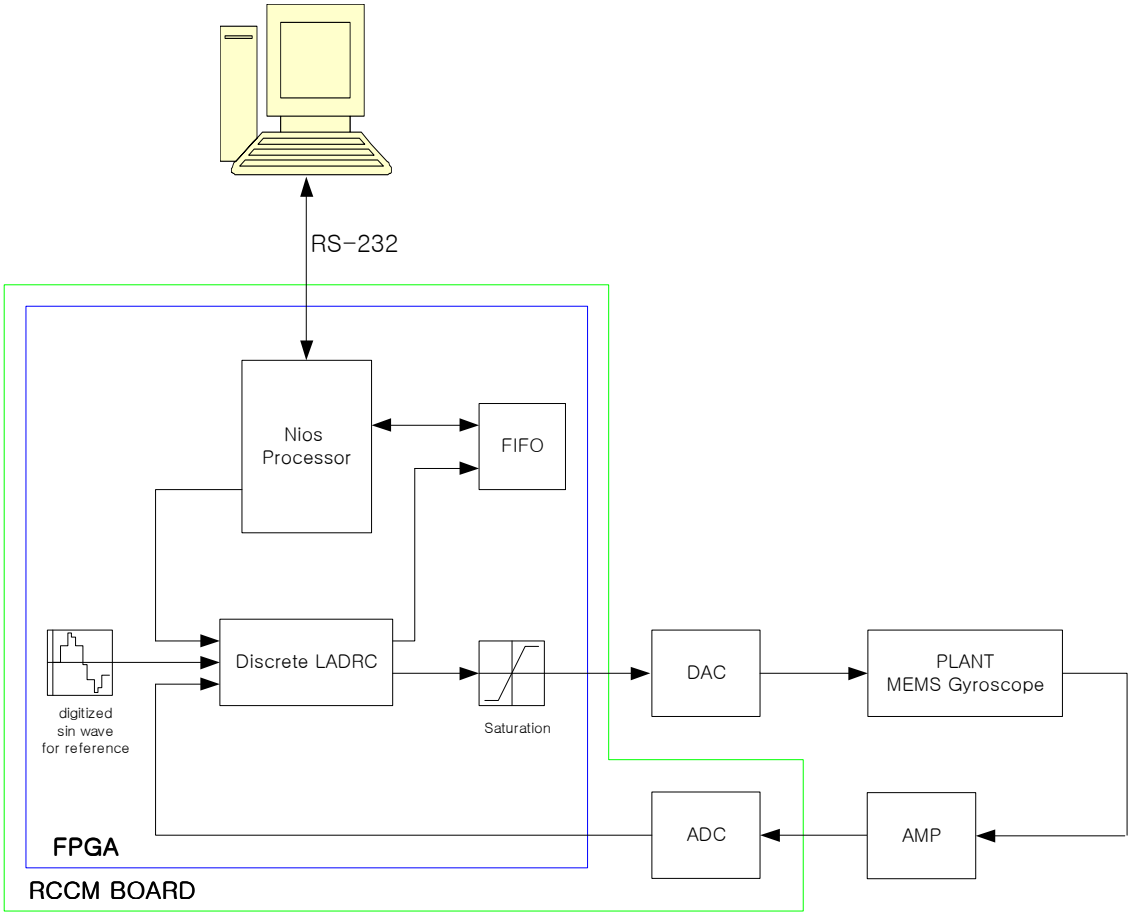


Figure 16 Discrete LADRC block diagram into FPGA

However, calculations in the control system are complex and using very large range of numbers to digital circuits. For this reason, better numerical expression and computation are required for this control system. Single precision floating-point from IEEE standard 754 has the capability to express a very large range of numbers. For this project, discrete LADRC designed by FPGA is based on calculations by a single precision floating-point.

For system design, Quartus II, version 3.2, for FPGA design by VHDL and schematics, SOPC builder for design Nios embedded processor, and GNUPro compiler for building both software and libraries are used. In the hardware design based on FPGA,

the controller is designed for the drive axis of the MEMS gyroscope which has two axes, drive axis and sense axis, for initial work.

4.1 Field Programmable Gate Array Implementation

4.1.1 Floating Point Number in FPGA

Considering discrete LADRC in FPGA, there are many calculations with very large positive and negative numbers and very small positive and negative numbers; decimal numbers. However, integer numbers expressed 32-bit binary have very limited range of numbers; from 2147483647 to -2147483648. With the 32-bit binary integer number expression, it is impossible to operate a discrete LADRC. In other to implement a discrete LADRC, a better method to express numbers using 32-bit must be implemented. It must have the capability of expressing a large range of numbers; not only large numbers but also decimal numbers without increasing any number of bits for data bus.

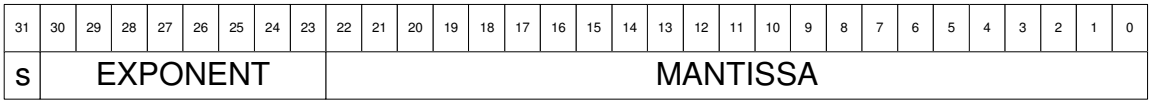


Figure 17 IEEE Standard 754 Floating-Point Bit-Map

Single precision floating-point from IEEE standard 754 could be the best solution for this problem [28]. This method can express from ± 1.17549435 × 10^{-38} to ±

3.4028235×10^{38} which covers most numbers that are required for the ESO calculations, and the PD controller calculations.

Figure 17 shows bit-map for IEEE standard 754 floating-point. S is sign bit; '1' for negative and '0' for positive number. The following 8 bits (23 – 30) are for exponent where base is two. For the exponent, a bias, 127, is added to the actual exponent in order to express decimal number. Finally, 23 bits (22 – 0) for mantissa represent fraction bits. Then, the real value that is presented by IEEE standard 754 floating-point is as the equation (4.1)

$$N = (-1)^{S} \times (1.m) \times 2^{E-127} \tag{4.1}$$

where N is real value, S is sign, m is mantissa, and E is exponent. The TABLE I: shows the special values that are expressed by floating-point.

TABLE I: FLOATING-POINT SPECIAL VALUES

Special Value	Representation	Value
Positive largest number	0 11111110 11111111111111111111111	$+ 3.4028235 \times 10^{38}$
Negative largest number	1 11111110 11111111111111111111111	$- 3.4028235 \times 10^{38}$
Positive smallest number	0 00000001 00000000000000000000000	$+ 1.17549435 \times 10^{-38}$
Negative smallest number	1 00000001 00000000000000000000000	$- 1.17549435 \times 10^{-38}$
+Zero	0 00000000 00000000000000000000000	+ 0.0
-Zero	1 00000000 00000000000000000000000	- 0.0
+Infinite	0 11111111 00000000000000000000000	--
-Infinite	1 11111111 00000000000000000000000	--
NaN	0 11111111 11111111111111111111111	--

Even though, this floating-point system has a very large range of number expression, it has a disadvantage. Because this method uses the same number of bus bits

as integer expression, it is less accurate. For example, if the number is 2.3, the best way to express this number is 2.299999952316284 where the binary code is 0 10000000 00100110011001100110011 in the single precision floating point expression. However, this small error could be ignored in the calculations of ESO.

4.1.2 Floating-Point Arithmetic

As the equations in Chapter 3 to implement CDESO and PD controller, they could be expressed by additions, subtractions, multiplications and divisions. Due to digital implementation, implementation of division causes the speed of system process to slow down. However, if the inverse of division is used for divisor, division could be replaced by multiplication.

To implement FPGA floating-point arithmetic for addition, subtraction and multiplication, the calculation process in Figure 18 is implemented. As Figure 18 shows, the first step is getting two inputs: A and B. Then, two inputs are compared to adjust the exponent and shift the mantissa and complement exponent of the larger number. The next step is calculation (addition, subtraction or multiplication). Right after calculation, the mantissa of the result is shifted, and the exponent of the result is complemented. Then, finally, we have the results.

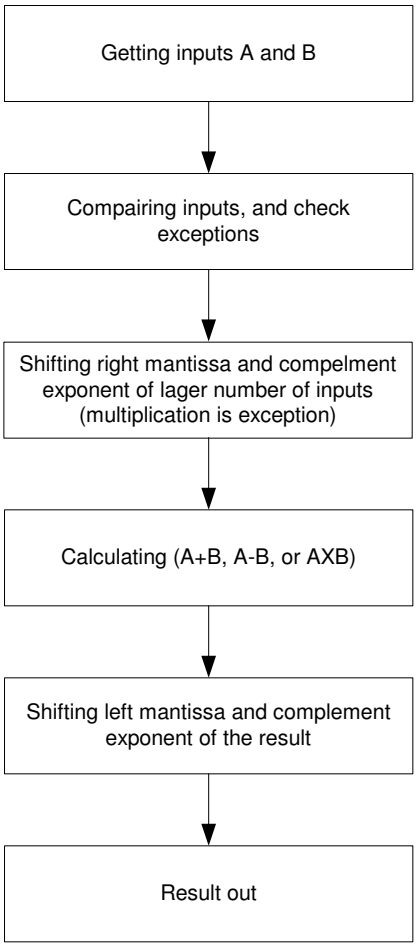


Figure 18 Calculation unit process

4.1.3 Discrete LADRC with FPGA

In Chapter 3, discrete LADRC implementation using MATLAB simulation for the MEMS gyroscope was discussed. Here, the real digital circuit implementation using FPGA for the current discrete LADRC will be explained.

First of all, the discrete ESO that is implemented in the MATLAB simulation must be changed to implemental forms into FPGA because matrixes can not be calculated in a digital circuit. For this reason, the equation (3.15) can be expanded as following:

$$\bar{\xi}_1(k) = \hat{\xi}_1(k-1) + T_s\hat{\xi}_2(k-1) + \frac{T_s^2}{2}\hat{\xi}_3(k-1) + \frac{T_s}{2}b_o u(k-1) \tag{4.2}$$

$$\bar{\xi}_2(k) = \hat{\xi}_2(k-1) + T_s\hat{\xi}_3(k-1) + T_s b_o u(k-1) \tag{4.3}$$

$$\bar{\xi}_3(k) = \hat{\xi}_3(k-1) \tag{4.4}$$

$$\hat{\xi}_1(k) = \bar{\xi}_1(k) + (1-\beta^3)(x(k) - \bar{\xi}_1(k)) \tag{4.5}$$

$$\hat{\xi}_2(k) = \bar{\xi}_2(k) + \frac{1.5(\beta-1)^2(\beta+1)}{T_s}(x(k) - \bar{\xi}_1(k)) \tag{4.6}$$

$$\hat{\xi}_3(k) = \bar{\xi}_3(k) + \frac{(\beta-1)^3}{T_s^2}(x(k) - \bar{\xi}_1(k)) \tag{4.7}$$

Then, for the discrete PD controller, equations (3.23) and (3.24) can be implemented. With the equations (4.2) ~ (4.7), the discrete LADRC block diagram for FPGA can be expressed as Figure 19.

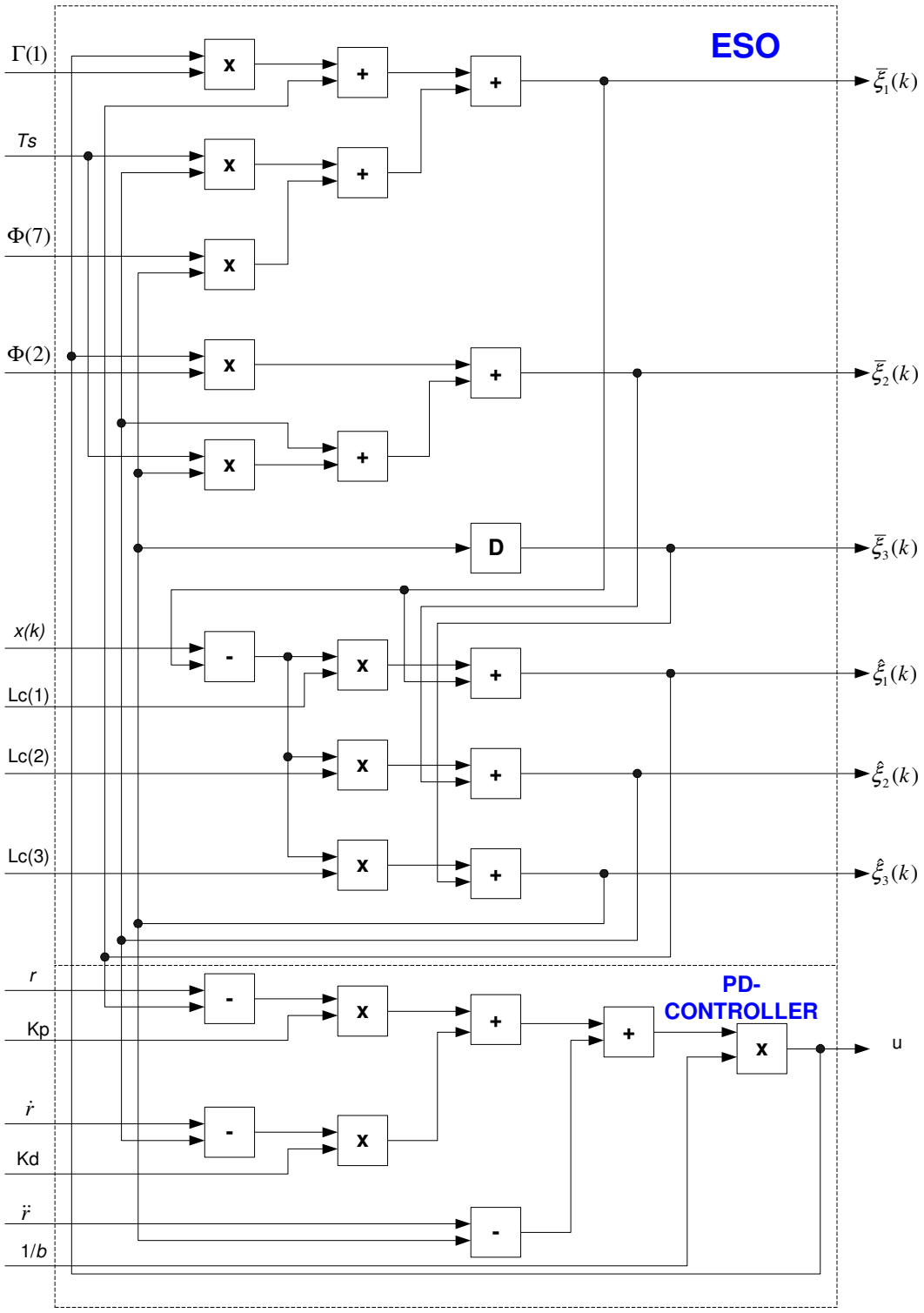


Figure 19 Discrete LADRC Block Diagram for FPGA

4.1.4 Nios Embedded Processor

Nios embedded processor is a soft core CPU, and Altera's System-On-a-Programmable-Chip (SOPC) builder system design tool can create Nios embedded processor design [29]. SOPC builder is a system design tool for design modules consist of processors, memory interfaces, buses and peripherals. At the same time that a system is generated, libraries and header files are generated by GNUPro compiler which is used to build both software and libraries for Nios embedded processor.

Initiating and resetting the system, getting commands from a computer, changing controller and ESO parameters, and controlling FIFO, DAC and ADC could be processed by the Nios-embedded processor in this system. In this system, Nios-embedded processor includes 2 Kbytes ROM for GERMS monitor, which is a simple monitor program that controls the boot process, and provides a way to read from and write to the on-board memories, 32 Kbytes RAM, one timer, 8-bit Universal Asynchronous Receiver/ Transmitter (UART) that translates data between parallel and serial interfaces, and other peripherals. Figure 20 shows the Nios-embedded processor for this system.

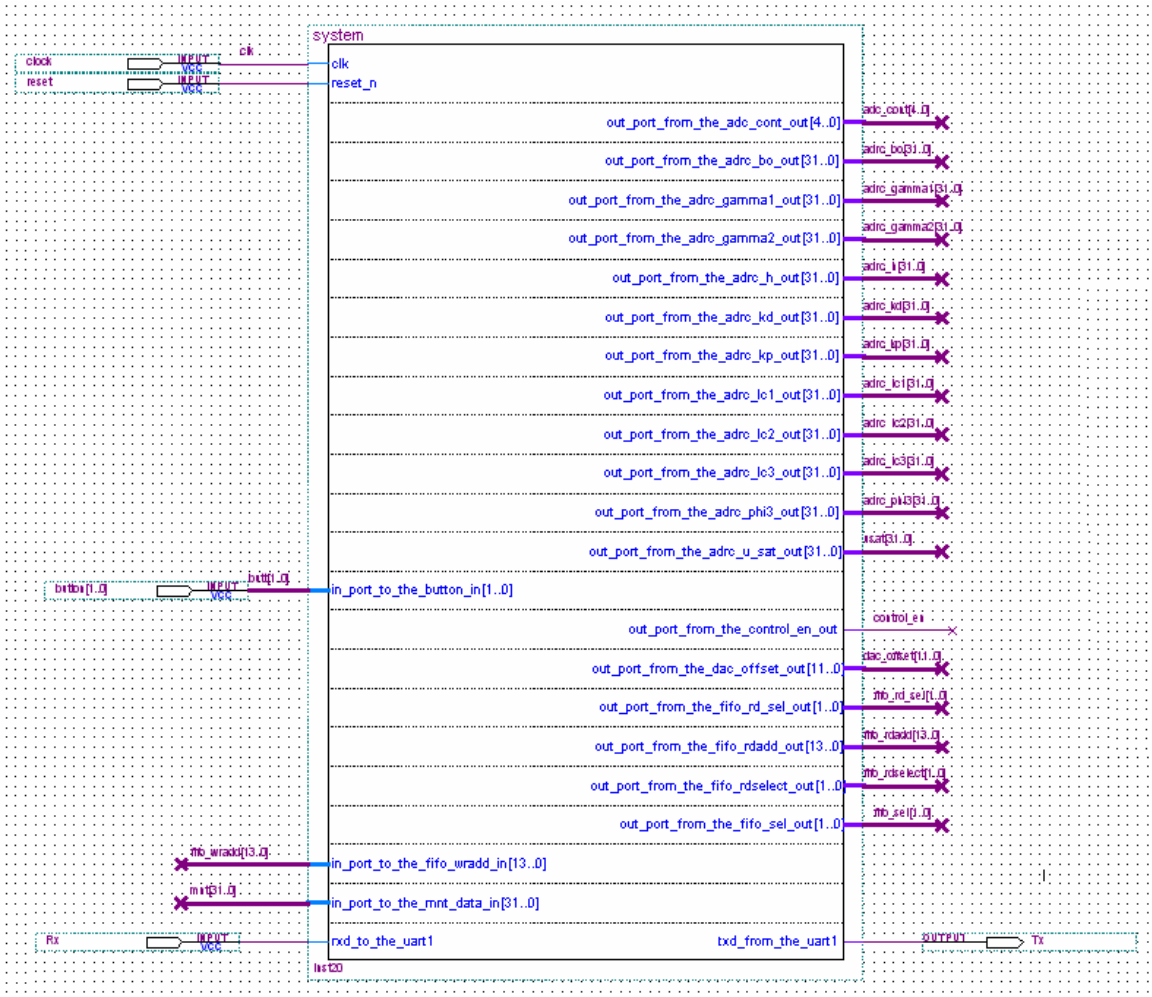


Figure 20 Nios Embedded Processor for the System

For the Nios processor in the system, c-source file must be built into the processor. Flow chart in Figure 21 explains the program. First of all, initialize all parameters and the system, and wait for the command from a computer. The command can be ‘control start’, ‘control stop’, ‘ADRC parameters change’, ‘monitoring options change’ or ‘terminate’. When the command is ‘control start’, the processor enables the system and controller, and sends data from input and output of gyroscope and ADRC to the computer.

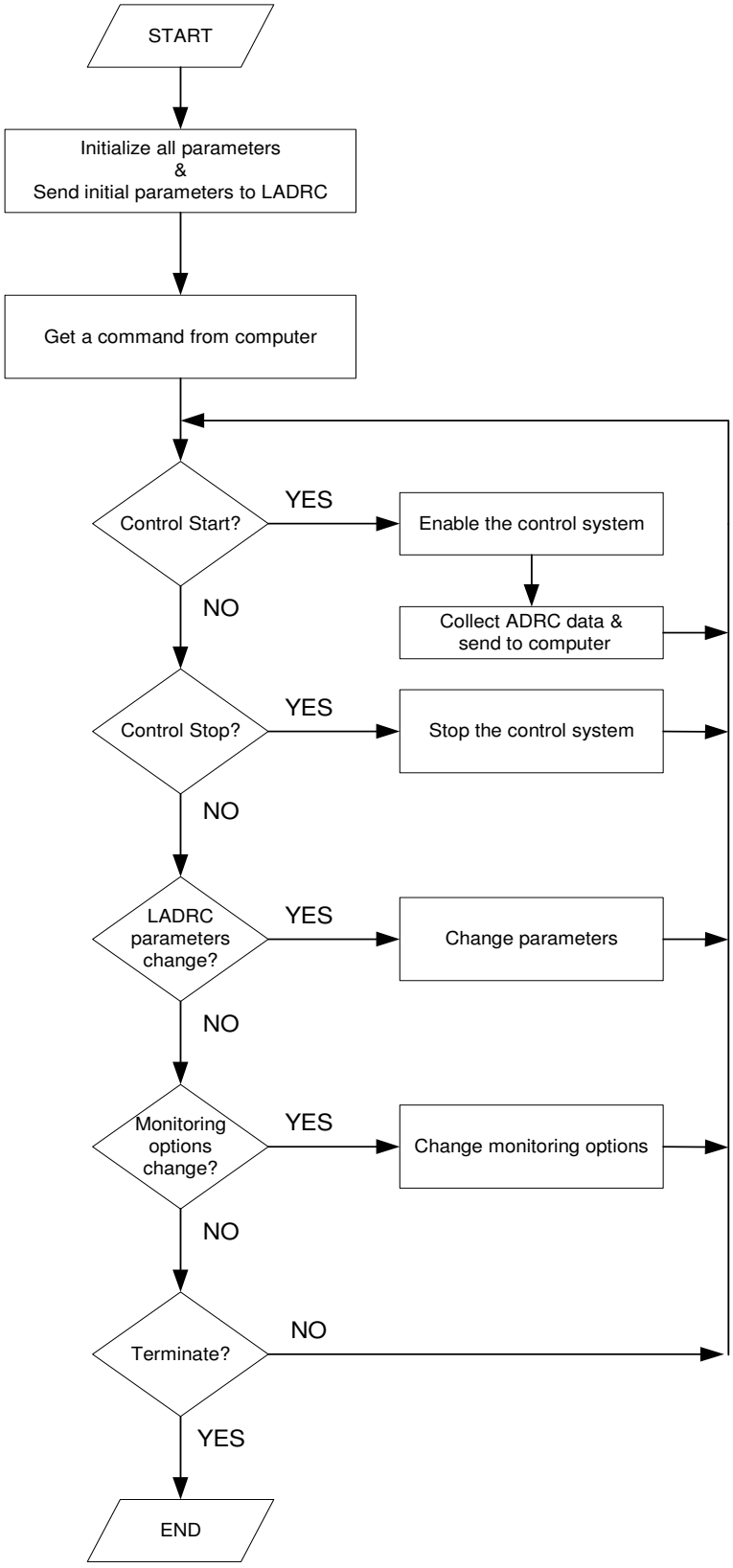


Figure 21 Nios processor flow chart for the system

4.1.5 Reference in FPGA

To control the drive axis, a reference signal, 10.1 *KHz* sinusoidal signal, is required. For this project, the reference could be generated by FPGA ROM (Read Only Memory). Doing this task, the sinusoidal signal must be digitized and saved into ROM.

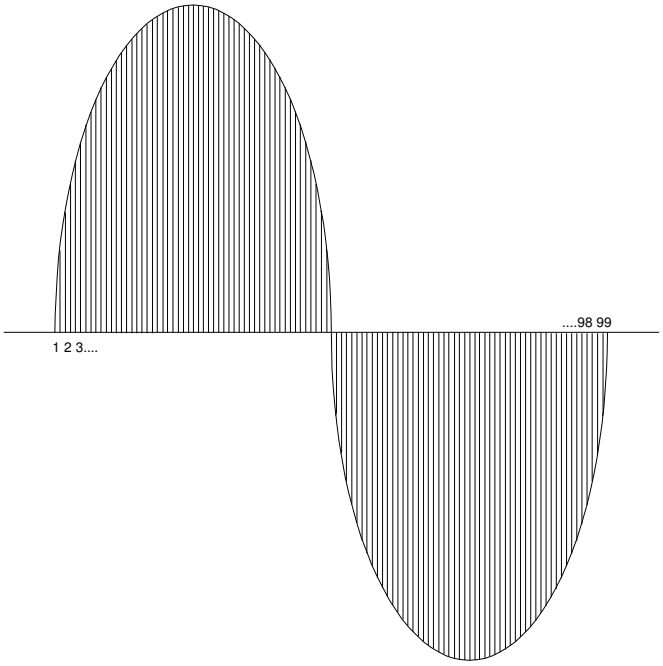


Figure 22 Sampling a sinusoid signal

As Figure 22 shows, one period of sinusoid signal is chopped into 99 samples. Here, 1 *MHz* clock frequency is used to generate this signal because of hardware limitation and clock speed. Using 1 *MHz* clock frequency for the sampling generates 10.1 *KHz* sinusoid signal.

However, as equation (3.23) expressed in Chapter 3, three references, r, $\dot{r}$ and $\ddot{r}$, are needed for the feed forward controller. For this reason, every single sample datum of reference is calculated for $\dot{r}$ and $\ddot{r}$, and saved into the ROM.

4.1.6 First In First Out Buffer Implementation

To control MEMS gyroscope, a very fast controller is required. For this project, 1 *MHz* sampling frequency is used for all system such as the discrete LADRC process speed, DAC and ADC speed, and reference generating speed. However, to observe the control system signals such as a control signal which is gyroscope input and a feedback signal which is gyroscope out, this system collects the signal data, and sends them to a computer through RS-232. Doing this observation, the communication speed, 115200 bps, through RS-232 is much slower than the sampling speed in the system. Because of this reason, a FIFO (First In First Out) buffer is required to save the data, and send it to a computer.

For this, three 16 K dual-port Random Access Memory (RAM) functions are implemented. This RAM function which is provided by Altera has 2 address bus inputs, one to access to write and the other to access to read, and these addresses can be accessed individually. The sampling speed for the FIFO is also slowed to 300 KHz which is enough speed to observe the graphs of the data. Mainly, a feedback signal and a control signal are chosen to collect, and one of the signals, reference, $\hat{\xi}_1$, $\hat{\xi}_2$ and $\hat{\xi}_3$, can be chosen to collect.

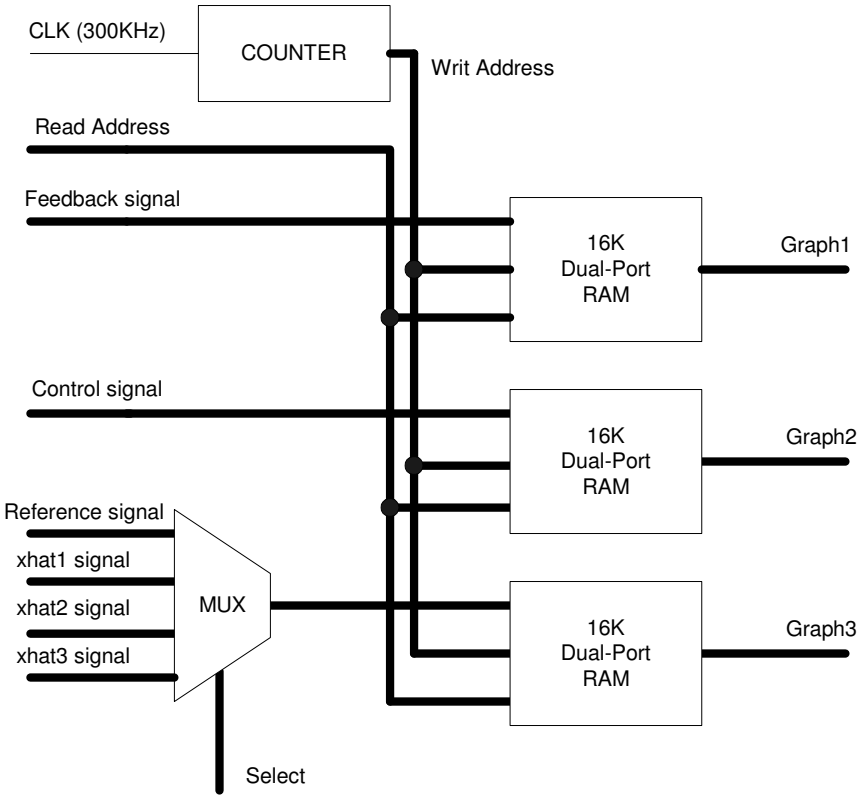


Figure 23 FIFO Block Diagram

4.2 Other Circuits

A digital controller is implemented for this project. However, to control the MEMS gyroscope, an analog signal is required. For this reason, it is necessary to use a DAC (Digital to Analog Converter) for the control signal and an ADC (Analog to Digital Converter) for feedback signal.

4.2.1 Digital to Analog Converter Circuit

The control output from the LADRC in FPGA is a digital signal, and this control signal must be converted to analog to control the gyroscope Beam. For this reason a DAC (Digital to Analog Converter) is implemented for this project. However, RCCM board output pins from FPGA are very limited (10 pins of JP10 and 6 pins of JP9), so AD5444 is chosen for this project. AD5444 is a 12-bit serial input DAC, and 50 *MHz* serial interface. Referring to the manual of AD5444, the following circuit, Figure 24, is implemented.

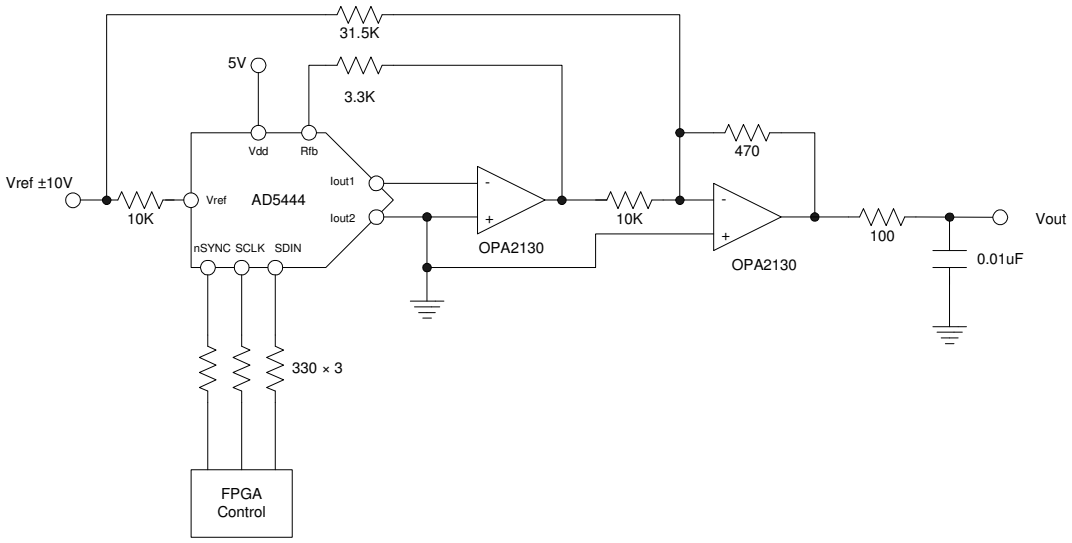


Figure 24 DAC Bipolar Operation Circuit

The clock speed for the DAC is 1 *MHz*. Some resister parameters are changed to adjust output voltage. Especially, the maximum output must be under the maximum input of the gyroscope. Here, the maximum out voltage is ± 100 $mV_{p\text{-}p}$. To reduce the noise, the analog ground must be separated from the digital ground, and some by-pass capacitors are added into the circuit.

4.2.2 Analog to Digital Converter Circuit

Since the implementation of this system is digital, ADC (Analog to Digital Converter) is required to acquire and analyze the feedback signal (MEMS gyroscope output). In the FPGA board, two amplifiers (AD8033 and AD8132) and one ADC (AD7266; Dual 12 bit, 3-channel) are implemented as Figure 25.

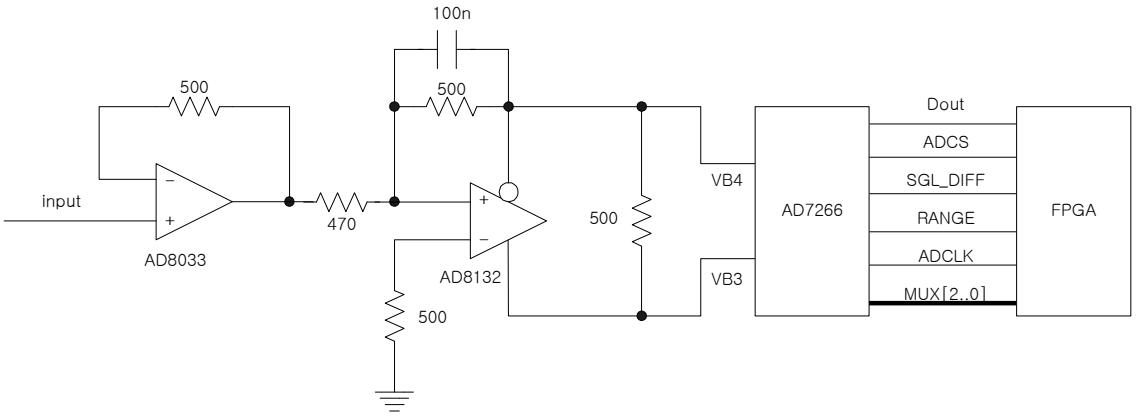


Figure 25 ADC circuits and blocks

Since the output of the MEMS gyroscope is low voltage (about ± 400 $mV_{p\text{-}p}$) comparing to ADC resolution (2 mV per a bit), it is necessary to amplify this signal. For this reason, one non-inverse amp circuit is build as Figure 26.

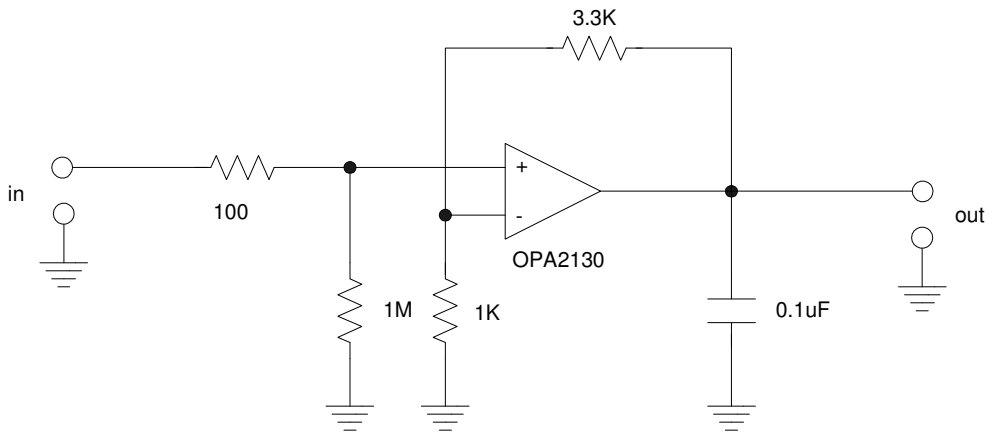


Figure 26 Non-inverse op-amp circuit

In Figure 26, the output can be calculated as $1+\frac{3.3K}{1K}=4.3$ so, the output is v_{out} = $4.3v_{in}$. Now, the resolution of the gyroscope output is increased 4.3 times.

4.3 JAVA User Interface

Java is a high-level programming language that is developed by Sun Microsystems. Like c++, Java is an object-oriented language, but it is simplified to eliminate language features causing common programming errors. A bytecode, .class file, which Java source code files, .java files, are compiled into, can be executed by a Java interpreter. Because Java Virtual Machines (VMs), which are Java interpreters and runtime environments, exist for most Operating Systems (OS) such as UNIX, the Macintosh OS, and Windows, the Java code that is compiled can run on most computers. Most of all, Java is open source language and everybody can download the compiler freely from the webpage of Sun Microsystems.

For this project, Java Development Kit (JDK), version 1.4, is used to design a simple program for communication between the FPGA board and a computer. There are three main operations in the Java user interface. First of all, the FPGA controller can be operated by the Java user interface; running the controller or stopping the controller. Secondly, the Java user interface collects the LADRC data from the FPGA controller, and draws three graphs on the grid monitor to observe the controller signals and the input and output of MEMS gyroscope as Figure 27 (a). Finally, parameters for LADRC can be changed using the Java user interface as Figure 27 (b). Due to changing the parameters, if all the calculations of matrixes, Φ, Γ, and L_c, defined in Chapter 3, are processed in the FPGA board, a big and complicated system is needed to calculate them. For this reason, the java user interface calculates the matrixes, changes the numbers to floating-point binary numbers, and sends them to the FPGA board.

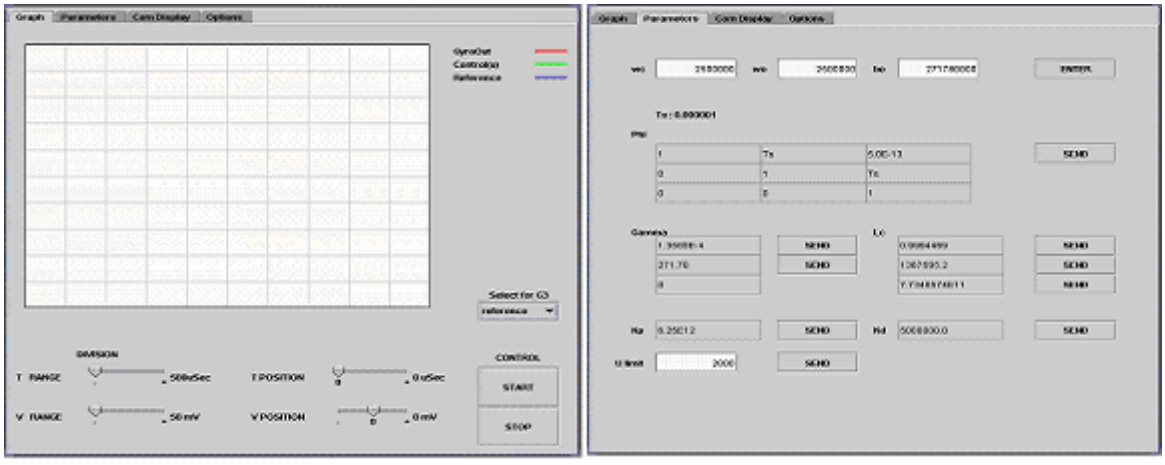


Figure 27 JAVA User Interface Program

CHAPTER V

HARDWARE RESULTS

As earlier mentioned, hardware implementation of discrete LADRC handles unknown disturbances, frequency mismatch and parameter variations well as the following results show. For the parameters for hardware implementation, ω_c is 2500000, ω_o is 2500000 and b_o is 2.7178×10^8. The following graphs in from Figure 28 to Figure 31 are the observations from the JAVA user interface. In Figure 28, the vibrating gyroscope output is shown when the time division is 5 *ms*, and the voltage division is 500 *mV*. The vibrating gyroscope output is constant and stable after about 20 *ms* with amplitude about 1800 mV_{pp} amplified by the circuit in Figure 26. Since the drive axis output is amplified 4.3 times according to the circuit in Figure 26, the actual output of the drive axis of the MEMS gyroscope is 418.6 mV_{pp}.

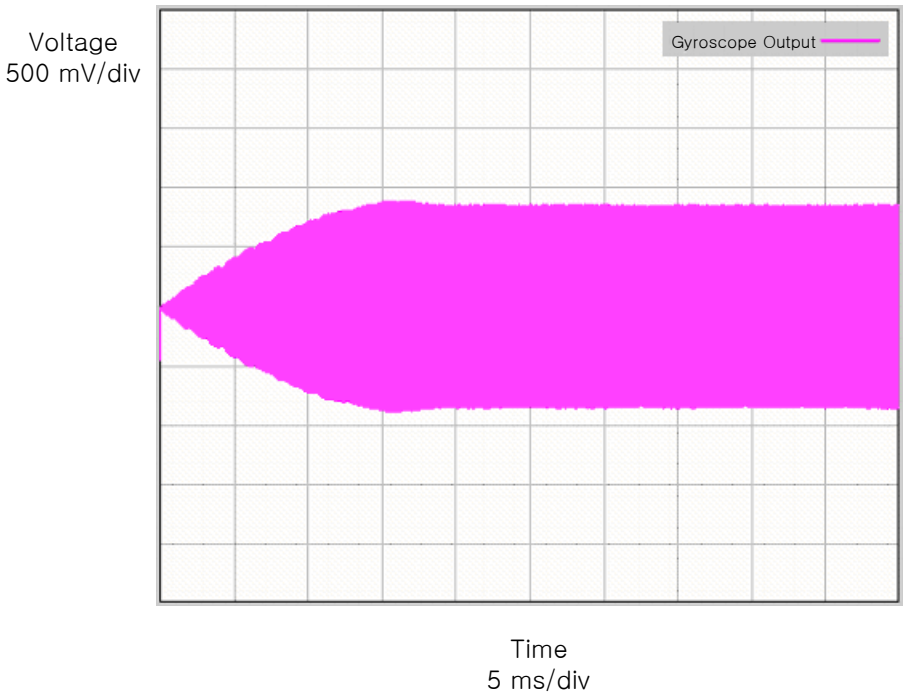


Figure 28 Steady State Output of the Drive Axis of the MEMS Gyroscope

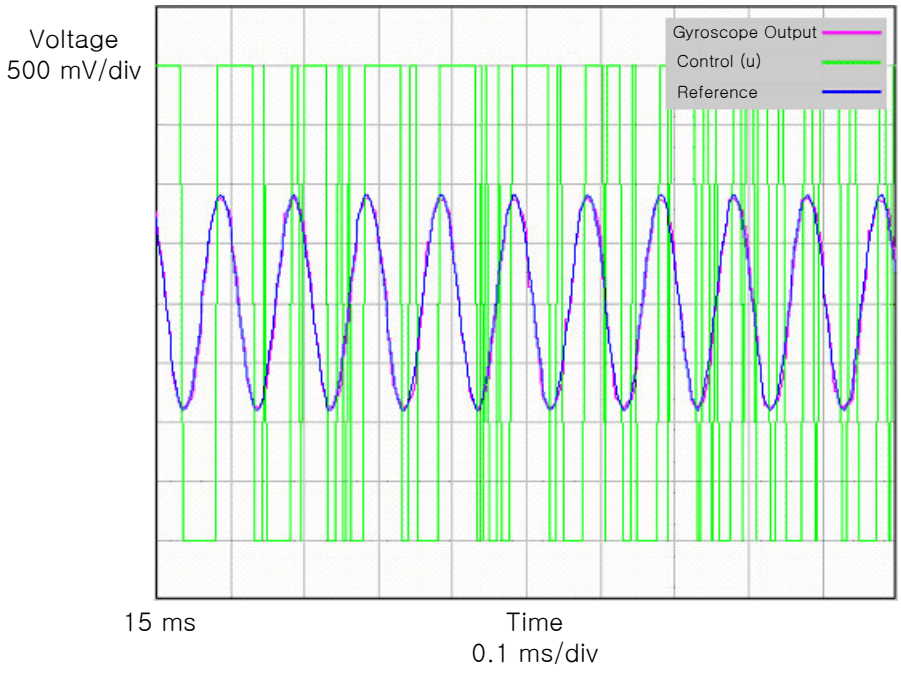


Figure 29 Gyroscope Control Signals

In Figure 29, the vibrating MEMS gyroscope output, red graph, matches to the reference signal, blue graph, very well without time delay at 0.1 *ms* for time division, 500 *mV* for voltage division, and 15 *ms* for time position.

The error is illustrated in Figure 30 at 5 *ms* for time division and 500 *mV* for voltage division. As it shows, the error is reduced until the MEMS gyroscope out gets stable. According the error graph, the error produced is approximately 10 % of the signal.

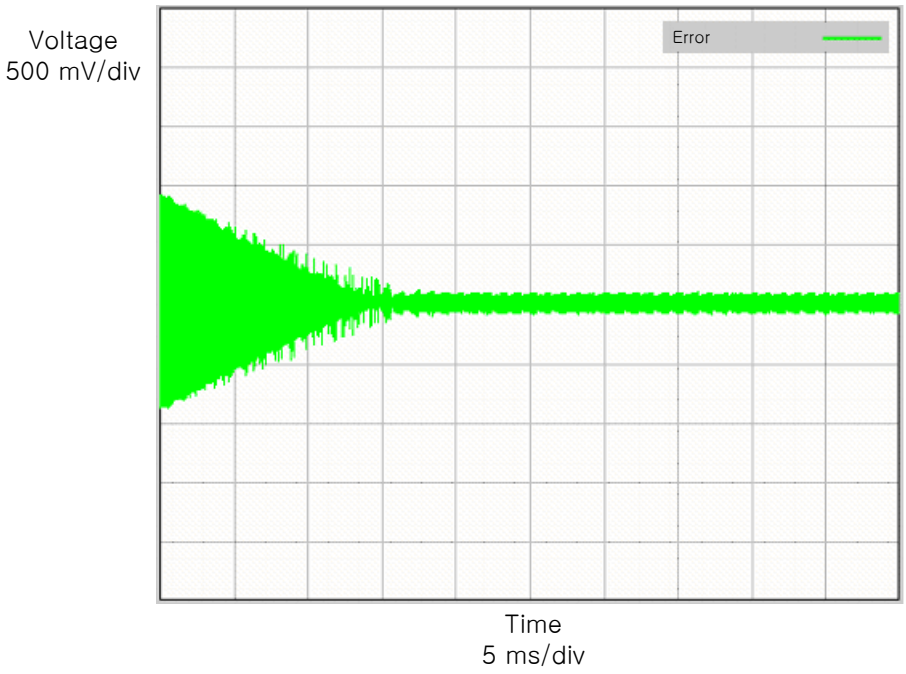


Figure 30 Tracking Error Signal between Reference signal and Gyroscope output

Current Discrete Extended System Observer must estimate the MEMS gyroscope output accurately without time delay. Figure 31 shows the vibrating MEMS gyroscope output and $\hat{\xi}_1$ from the CDESO at the same time. The hardware implementation of

FPGA-based discrete LADRC estimates the MEMS gyroscope output very well as in Figure 31.

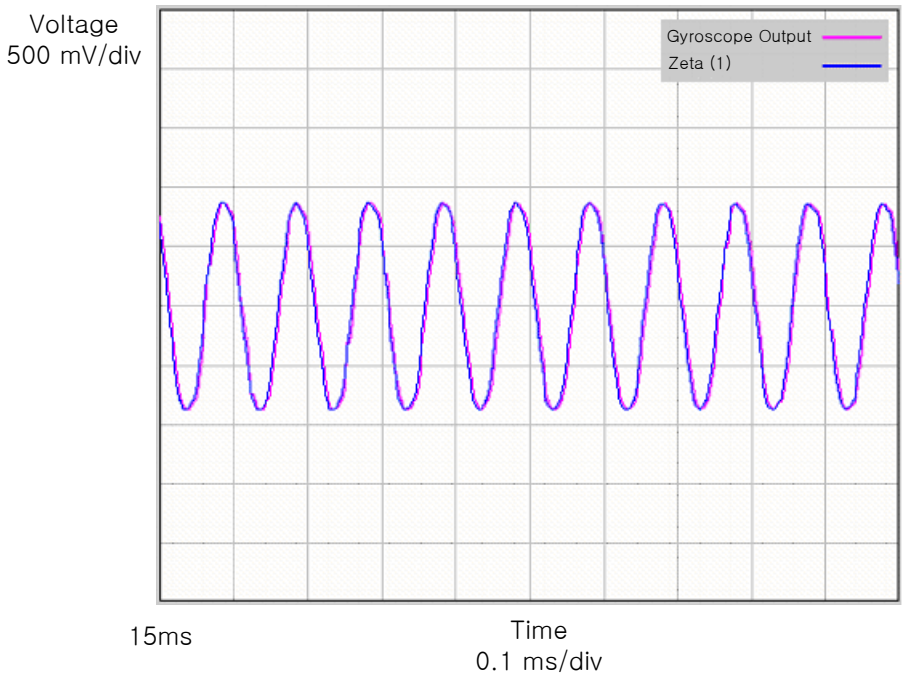


Figure 31 ESO Zeta (1) and the Output of the Drive Axis

The results from the FPGA-based discrete LADRC can be compared with those from the simulation implementation introduced in Chapter 3, and those from the analog adaptive control in [12]. As the TABLE II: shows, even though the FPGA-based discrete LADRC has a slow response and more error compared to simulation, its response is much faster and with less error than those from an analog adaptive control.

TABLE II: COMPARISON OF RESULTS

	Analog adaptive control	Simulation (ADRC)	FPGA-based discrete LADRC
Time	3 *s*	5.5 *ms*	20 *ms*
Error	15 %	3.72 %	10 %

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

A novel controller based on ADRC is implemented to control the drive axis of the vibrating MEMS gyroscope. The complex, nonlinear, and uncertain dynamics commonly seen in MEMS devices is reformulated as a disturbance rejection problem, where the active disturbance rejection concept is readily applied. The results, obtained from both the simulation implementation, using MATLAB/Simulink, and from the hardware implementation based on FPGA, are significantly better than those obtained from the analog adaptive controller [12]. In particular, significant improvement in speed and a slight reduction in error are achieved, meeting the design requirements with minimum requirement on the information of plant dynamics.

The most significant contribution of the thesis research is to meet the speed and accuracy requirements that come with the MEMS applications. The high controller bandwidth is attained by implementing the controller on an FPGA board and by digitizing the ADRC controller so that it can be programmed into the FPGA circuitry using VHDL. To improve the numerical accuracy, the controller is implemented using a

floating-point arithmetic based on the IEEE standard 754. To reduce the phase lag, the disturbance observer is implemented in the CDESO form with single parameter tuning.

There are a number of issues that perhaps can be addressed in future research:

1) The sampling rate can be further increased with a better FPGA implementation or using a faster FPGA chip. This will certainly lead to better performance in terms of smaller settling time and steady state error;

2) The interface circuit, implemented on a breadboard, connecting the FPGA board to the gyroscope, should be implemented in a Printed Circuit Board (PCB) to reduce the noise and further improve the accuracy;

3) Implementation of the FPGA-based ADRC can be extended to control both the drive and sense axes, where the rotation rate is estimated digitally.

REFERENCES

1. Lili Dong, "Adaptive Control System for a Vibrational MEMS Gyroscope with Time-varying Rotation Rates," PhD dissertation, 2005.

2. A. Lawrence, "Modern Inertial Technology," *Springer-Verlag*, New York, 1998.

3. http://images.encarta.msn.com/xrefmedia/aencmed/targets/illus/ilt

4. Cenk Acar and Andrei M. Shekel, "An Approach for Increasing Drive-Mode Bandwidth of MEMS Vibratory Gyroscopes," *Journal of Microelectromechanical Systems*, Vol. 14, No. 3, pp. 520-528, June 2005.

5. Cenk Acar and Andrei M. Shekel, "Inherently Robust Micromachined Gyroscopes With 2-DOF Sense-Mode Oscillator," *Journal of Microelectromechanical Systems*, Vol. 15, No. 2, pp. 380-387, April 2005.

6. B Borovic, A Q Liu, D Popa, H Cai and F L Lewis, "Open-loop versus closed-loop control of MEMS devices: choises and issues," *Journal of micromechanics and micromengineering*, Printed in the UK, pp. 101917-101924, Aug. 2005.

7. S. Chang, M. Chia, P. Castillo-Borelley, W. Higdon, Q. Jiang, J. Johnson, L. Obedier, M. Putty, Q. Shi, D. Sparks, and S. Zarabadi, "An electroformed CMOS integrated angular rate sensor," Sensors Actuators, Vol. A66, pp. 138-143, 1998.

8. X. Jiang, J. Seeger, M. Kraft, and B. E. Boser, "A monolithic surface micromachined Z- axis gyroscope with digital output," in *IEEE 2000 Symposium on VLSI Circuits Digest of Technical papers*, pp. 16-19, June 2000.

9. C. Lu, M. Lamkin and B. E. Boser, "A monolithic surface micromachined accelerometer with digital output," *IEEE Journal of Solid-State Circuits*, Vol. 30, No. 12, pp. 1367-1373, Dec. 1995.

10. R. P. Leland, "Lyapunov based adaptive control of a MEMS gyroscope," in *IEEE Proceedings of the American Control Conference*, Anchorage, AK, pp. 3765-3770, May 8-10, 2002.

11. Sungsu Park and Roberto Horowitz, "Adaptive control for the Conventional Mode of Operation of MEMS Gyroscopes," *Journal of Microelectromechanical Systems*, Vol. 12, No. 1, pp. 101-108, February 2003.

12. R. P. Leland, "Adaptive Mode Tuning for Vibrational Gyroscopes," in *IEEE transactions on control systems technology*, Vol. 11, No. 2, pp. 242-247, March 2003.

13. Sungsu Park and Roberto Horowitz, "Discrete Time Adaptive Control for a MEMS Gyroscope," *International Journal of Adaptive Control and Signal Processing int. J. adapt. Control Signal Process,* 2005.

14. Park S and Horowitz R, "New Adaptive Mode of Operation for MEMS Gyroscopes," in *the ASME Journal of Dynamic Systems, Measurement and Controls*, Vol. 126, pp. 800-810, September 2003.

15. J. Han, "Nonlinear design methods for control systems," Proc. of the 14th IFAC World Congress, Beijing, 1999.

16. Z. Gao, Y. Huang, and J. Han, "An alternative paradigm for control system design," *In Proceedings of the IEEE conference on Decision and Control*, pp. 4578-4585, 2001.

17. Z. Gao, "Scaling and parameterization based controller tuning," *Proceedings of the American Control Conference*, pp. 4989-4996, 2003.

18. Madhura Shaligram, "Disturbance Rejection and Embedded Networking for a Class of Commercial DC-DC Power Converters," master's thesis, Cleveland State University, Cleveland, OH, May 2006.

19. Zhan Ping, "A Hardware-Reconfigurable control Platform for Space Power Management and Distribution Systems," dissertation, Cleveland State University, Cleveland, OH, May 2005.

20. Aaron Radke, "Control System Health and Fault Monitoring Through Data Driven Lumped Disturbance Estimation," Cleveland State University, Cleveland, OH, May 2006.

21. Z. Gao, S. Hu, and F. Jiang, "A novel motion control design approach based on active disturbance rejection," *in Proceedings of the IEEE Conference on Decision and Control*, pp. 4877-4882, 2001.

22. http://en.wikipedia.org/wiki/FPGA

23. Altera Corporation, Stratix Device Handbook, Vol. 1, July 2005.

24. S. D. Senturia, Microsystem Design. Springer Publisher, Dec. 2004.

25. Z. Gao, "Active disturbance rejection control: a paradigm shift in feedback control system design," *In Proceedings of the American Control Conference*, pp. 2399-2405, 2006.

26. G.F. Franklin, J.D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd Ed., Addison-Wesley, 1997.

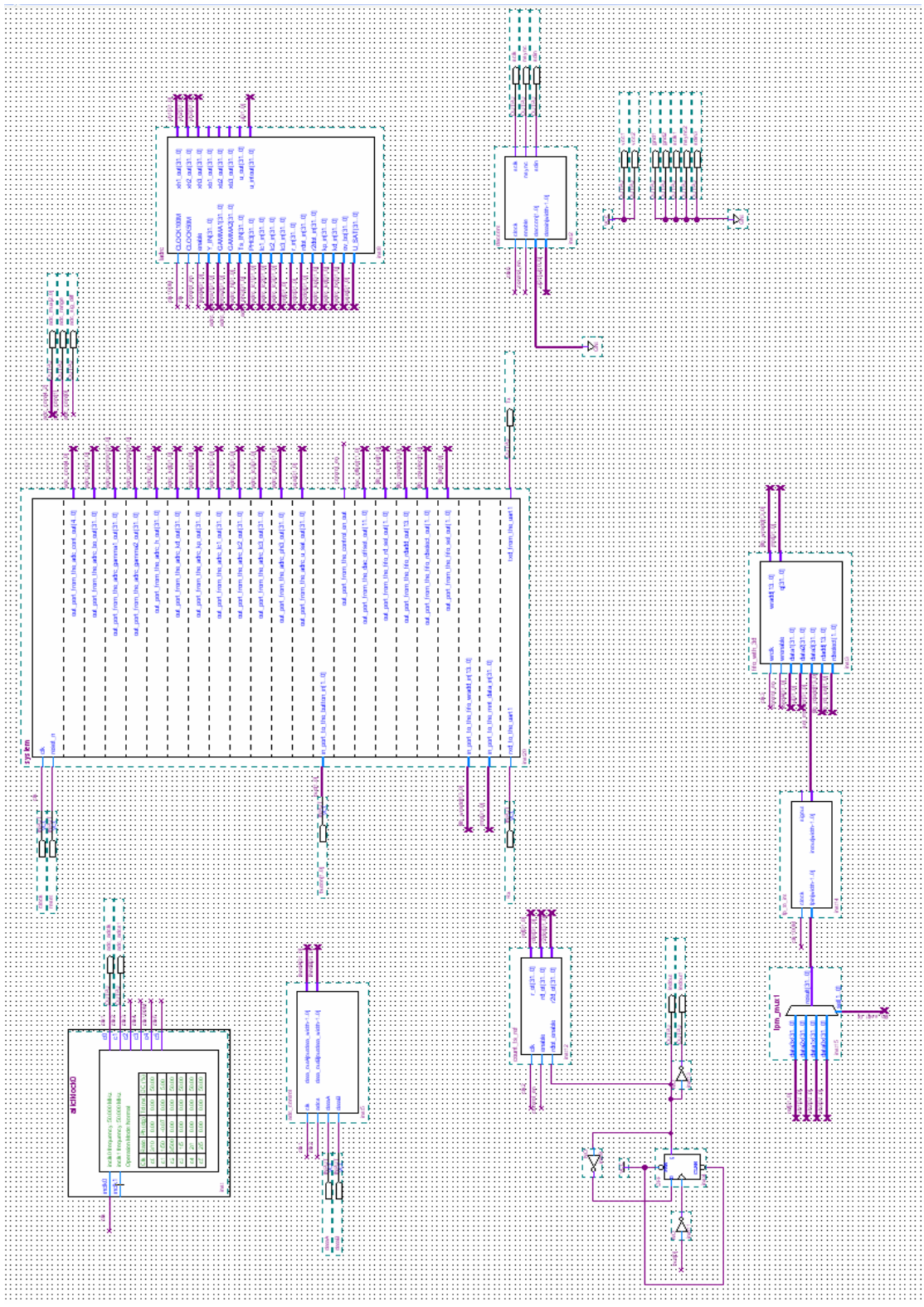
27. R. P. Leland, "Adaptive tuning for vibrational gyroscopes," in *Proceedings of 40th IEEE Conference on Decision and Control*, Orlando, FL, pp. 3447-3452, Dec. 2001.

28. Steve Hollasch, "IEEE Standard 754 Floating Point Numbers", http://steve.hollasch.net/cgindex/coding/ieeefloat.html, Feb. 24, 2005

29. Altera Corporation, "Nios Embedded Processor Software Development Reference Manual," Vol. 3.2, March 2003.

APPENDICES

A. Digital Schematic for FPGA Control System



B. Discrete ESO VHDL Code

```
IF (clock'event AND clock = '1') THEN
IF (enable = '1') THEN
IF (s_cnt = 0) THEN
        complt <= '0';
        s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 1) THEN
        s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 2) THEN
        s_sin_y <= y_in(width-1);
        s_exp_y <= y_in(width-2 DOWNTO width-exp-1);
        s_mant_y<= '1' & y_in(mant-1 DOWNTO 0);
        s_sin_u <= u_in(width-1);
        s_exp_u <= u_in(width-2 DOWNTO width-exp-1);
        s_mant_u<= '1' & u_in(mant-1 DOWNTO 0);
        s_sin_lc1 <= lc1_in(width-1);
        s_exp_lc1 <= lc1_in(width-2 DOWNTO width-exp-1);
        s_mant_lc1<= '1' & lc1_in(mant-1 DOWNTO 0);
        s_sin_lc2 <= lc2_in(width-1);
        s_exp_lc2 <= lc2_in(width-2 DOWNTO width-exp-1);
        s_mant_lc2<= '1' & lc2_in(mant-1 DOWNTO 0);
        s_sin_lc3 <= lc3_in(width-1);
        s_exp_lc3 <= lc3_in(width-2 DOWNTO width-exp-1);
        s_mant_lc3<= '1' & lc3_in(mant-1 DOWNTO 0);
        s_sin_gamma1 <= gamma1_in(width-1);
        s_exp_gamma1 <= gamma1_in(width-2 DOWNTO width-exp-1);
        s_mant_gamma1<= '1' & gamma1_in(mant-1 DOWNTO 0);
        s_sin_gamma2 <= gamma2_in(width-1);
        s_exp_gamma2 <= gamma2_in(width-2 DOWNTO width-exp-1);
        s_mant_gamma2<= '1' & gamma2_in(mant-1 DOWNTO 0);
        s_sin_phi <= phi_in(width-1);
        s_exp_phi <= phi_in(width-2 DOWNTO width-exp-1);
        s_mant_phi<= '1' & phi_in(mant-1 DOWNTO 0);
        s_sin_ts <= ts_in(width-1);
        s_exp_ts <= ts_in(width-2 DOWNTO width-exp-1);
        s_mant_ts<= '1' & ts_in(mant-1 DOWNTO 0);
        s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 3) THEN                          -- COMPARASONS for sub
IF ((s_exp_rt15 = 0) AND (s_exp_y = 0)) THEN
        s_sin_rt0 <= '0';
        s_exp_rt0 <= (OTHERS => '0');
        s_mant_rt0<= (OTHERS => '0');
        s_skip1   <= '1';
ELSIF (s_exp_rt15 = 255) THEN
        s_sin_rt0 <= NOT(s_sin_rt15);
s_exp_rt0 <= s_exp_rt15;
        s_mant_rt0<= s_mant_rt15;
        s_skip1   <= '1';
ELSIF (s_exp_y = 255) THEN
        s_sin_rt0 <= s_sin_y;
        s_exp_rt0 <= s_exp_y;
        s_mant_rt0<= s_mant_y & '0';
```

```
            s_skip1   <= '1';
ELSIF (s_exp_rt15 = 0) THEN
            s_sin_rt0 <= s_sin_y;
            s_exp_rt0 <= s_exp_y;
            s_mant_rt0<= s_mant_y & '0';
            s_skip1   <= '1';
ELSIF (s_exp_y = 0) THEN
            s_sin_rt0 <= NOT(s_sin_rt15);
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= s_mant_rt15;
            s_skip1   <= '1';
ELSIF (s_exp_rt15 > s_exp_y) THEN
            s_rsft_din <= s_mant_y;
            s_rshift   <= s_exp_rt15 - s_exp_y;
            s_skip1   <= '0';
ELSIF (s_exp_rt15 <= s_exp_y) THEN
            s_rsft_din <= s_mant_rt15(mant+1 DOWNTO 1);
            s_rshift   <= s_exp_y - s_exp_rt15;
            s_skip1    <= '0';
END IF;
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 4) THEN
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 5) THEN
IF (s_skip1 = '0') THEN
IF (s_exp_rt15 > s_exp_y) THEN
            s_exp_y <= s_exp_rt15;
            s_mant_y<= s_rsft_dot;
ELSE
            s_exp_rt15 <= s_exp_y;
            s_mant_rt15<= s_rsft_dot & '0';
END IF;
END IF;
s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 6) THEN                -- CALCULATIONS OF SUB
IF (s_skip1 = '0') THEN
IF ((s_sin_rt15 = '1') AND (s_sin_y = '0')) THEN
            s_sin_rt0 <= '0';
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) + ('0' & s_mant_y);
            s_skip1 <= '0';
ELSIF  ((s_sin_rt15 = '0') AND (s_sin_y = '0')) THEN
IF (s_mant_rt15(mant+1 DOWNTO 1) > s_mant_y) THEN
            s_sin_rt0 <= '1';
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) - ('0' & s_mant_y);
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) < s_mant_y) THEN
            s_sin_rt0 <= '0';
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= ('0' & s_mant_y) - ('0' & s_mant_rt15(mant+1 DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) = s_mant_y) THEN
            s_sin_rt0 <= '0';
            s_exp_rt0 <= (OTHERS => '0');
            s_mant_rt0<= (OTHERS => '0');
```

```
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt15 = '1') AND (s_sin_y = '1')) THEN
IF (s_mant_rt15(mant+1 DOWNTO 1) > s_mant_y) THEN
            s_sin_rt0 <= '0';
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) - ('0' & s_mant_y);
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) < s_mant_y) THEN
            s_sin_rt0 <= '1';
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= ('0' & s_mant_y) - ('0' & s_mant_rt15(mant+1 DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) = s_mant_y) THEN
            s_sin_rt0 <= '0';
            s_exp_rt0 <= (OTHERS => '0');
            s_mant_rt0<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt15 = '0') AND (s_sin_y = '1')) THEN
            s_sin_rt0 <= '1';
            s_exp_rt0 <= s_exp_rt15;
            s_mant_rt0<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) + ('0' & s_mant_y);
            s_skip1 <= '0';
END IF;
END IF;
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 7) THEN
IF (s_skip1 = '0') THEN
            s_lshift_min(mant*2+2 DOWNTO mant+1) <= s_mant_rt0;
            s_lshift_min(mant DOWNTO 0)          <= (OTHERS => '0');
            s_lshift_ein                         <= s_exp_rt0;
END IF;
            s_lshift_min1(mant*2+2 DOWNTO mant+1) <= s_mant_rt15;
            s_lshift_min1(mant+1 DOWNTO 0)       <= (OTHERS => '0');
            s_lshift_ein1                        <= s_exp_rt15;
            s_cnt                                <= s_cnt + 1;
ELSIF (s_cnt = 8) THEN
            s_cnt <= s_cnt + 1;                                  -- WAITING FOR SHIFTING BITS
ELSIF (s_cnt = 9) THEN
IF (s_skip1 = '0') THEN
            s_exp_rt0 <= s_lshift_eot + 1;
            s_mant_rt0<= s_lshift_mot(mant*2+2 DOWNTO mant+1);
END IF;
            s_exp_rt15 <= s_lshift_eot1;
            s_mant_rt15<= s_lshift_mot1(mant*2+2 DOWNTO mant+1);
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 10) THEN
IF (s_exp_rt0 = 0) THEN
            s_sin_rt1 <= '0';
            s_exp_rt1 <= (OTHERS => '0');
            s_mant_rt1<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_lc1 = 0) THEN
            s_sin_rt1 <= '0';
            s_exp_rt1 <= (OTHERS => '0');
```

```
            s_mant_rt1<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_rt0 = 255) THEN
            s_sin_rt1 <= '0';
            s_exp_rt1 <= (OTHERS => '1');
            s_mant_rt1<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_lc1 = 255) THEN
            s_sin_rt1 <= '0';
            s_exp_rt1 <= (OTHERS => '1');
            s_mant_rt1<= (OTHERS => '0');
            s_skip1   <= '1';
ELSE
            s_skip1   <= '0';
END IF;
IF (s_exp_rt0 = 0) THEN
            s_sin_rt2 <= '0';
            s_exp_rt2 <= (OTHERS => '0');
            s_mant_rt2<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_lc2 = 0) THEN
            s_sin_rt2 <= '0';
            s_exp_rt2 <= (OTHERS => '0');
            s_mant_rt2<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_rt0 = 255) THEN
            s_sin_rt2 <= '0';
            s_exp_rt2 <= (OTHERS => '1');
            s_mant_rt2<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_lc2 = 255) THEN
            s_sin_rt2 <= '0';
            s_exp_rt2 <= (OTHERS => '1');
            s_mant_rt2<= (OTHERS => '0');
            s_skip2   <= '1';
ELSE
            s_skip2   <= '0';
END IF;
IF (s_exp_rt0 = 0) THEN
            s_sin_rt3 <= '0';
            s_exp_rt3 <= (OTHERS => '0');
            s_mant_rt3<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_lc3 = 0) THEN
            s_sin_rt3 <= '0';
            s_exp_rt3 <= (OTHERS => '0');
            s_mant_rt3<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_rt0 = 255) THEN
            s_sin_rt3 <= '0';
            s_exp_rt3 <= (OTHERS => '1');
            s_mant_rt3<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_lc3 = 255) THEN
            s_sin_rt3 <= '0';
            s_exp_rt3 <= (OTHERS => '1');
```

```
            s_mant_rt3<= (OTHERS => '0');
            s_skip3   <= '1';
ELSE
            s_skip3   <= '0';
END IF;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 11) THEN
IF (s_skip1 = '0') THEN
            s_sin_rt1 <= s_sin_rt0 XOR s_sin_lc1;
            s_exp_rt1 <= ('0' & s_exp_rt0) + ('0' & s_exp_lc1) - base;
            s_mant_rt1<= s_mant_rt0 * s_mant_lc1;
END IF;
IF (s_skip2 = '0') THEN
            s_sin_rt2 <= s_sin_rt0 XOR s_sin_lc2;
            s_exp_rt2 <= ('0' & s_exp_rt0) + ('0' & s_exp_lc2) - base;
            s_mant_rt2<= s_mant_rt0 * s_mant_lc2;
END IF;
IF (s_skip3 = '0') THEN
            s_sin_rt3 <= s_sin_rt0 XOR s_sin_lc3;
            s_exp_rt3 <= ('0' & s_exp_rt0) + ('0' & s_exp_lc3) - base;
            s_mant_rt3<= s_mant_rt0 * s_mant_lc3;
END IF;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 12) THEN
IF (s_skip1 = '0') THEN
IF (s_exp_rt1 < 233) THEN
            s_lshift_min <= s_mant_rt1;
            s_lshift_ein <= s_exp_rt1(exp-1 DOWNTO 0);
            s_skip1      <= '0';
ELSE
            s_mant_rt1 <= (OTHERS => '0');
            s_exp_rt1  <= (OTHERS => '1');
            s_skip1    <= '1';
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF (s_exp_rt2 < 233) THEN
            s_lshift_min1 <= s_mant_rt2;
            s_lshift_ein1 <= s_exp_rt2(exp-1 DOWNTO 0);
            s_skip2       <= '0';
ELSE
            s_mant_rt2 <= (OTHERS => '0');
            s_exp_rt2  <= (OTHERS => '1');
            s_skip2    <= '1';
END IF;
END IF;
IF (s_skip3 = '0') THEN
IF (s_exp_rt3 < 233) THEN
            s_lshift_min2 <= s_mant_rt3;
            s_lshift_ein2 <= s_exp_rt3(exp-1 DOWNTO 0);
            s_skip3       <= '0';
ELSE
            s_mant_rt3 <= (OTHERS => '0');
            s_exp_rt3  <= (OTHERS => '1');
            s_skip3    <= '1';
END IF;
```

```
END IF;
            s_cnt       <= s_cnt + 1;
ELSIF (s_cnt = 13) THEN
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 14) THEN
IF (s_skip1 = '0') THEN
            s_exp_rt1 <= '0' & (s_lshift_eot + 1);
            s_mant_rt1<= s_lshift_mot(mant*2+2 DOWNTO 0);
END IF;
IF (s_skip2 = '0') THEN
            s_exp_rt2 <= '0' & (s_lshift_eot1 + 1);
            s_mant_rt2<= s_lshift_mot1(mant*2+2 DOWNTO 0);
END IF;
IF (s_skip3 = '0') THEN
            s_exp_rt3 <= '0' & (s_lshift_eot2 + 1);
            s_mant_rt3<= s_lshift_mot2(mant*2+2 DOWNTO 0);
END IF;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 15) THEN                                    -- COMPARASON
IF ((s_exp_rt15 = 0) AND (s_exp_rt1 = 0)) THEN
            s_sin_rt4 <= '0';
            s_exp_rt4 <= (OTHERS => '0');
            s_mant_rt4<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_rt15 = 255) THEN
            s_sin_rt4 <= s_sin_rt15;
            s_exp_rt4 <= s_exp_rt15;
            s_mant_rt4<= s_mant_rt15(mant+1 DOWNTO 0);
            s_skip1   <= '1';
ELSIF (s_exp_rt1 >= 255) THEN
            s_sin_rt4 <= s_sin_rt1;
            s_exp_rt4 <= (OTHERS => '1');
            s_mant_rt4<= s_mant_rt1(mant*2+2 DOWNTO mant+1);
            s_skip1   <= '1';
ELSIF (s_exp_rt15 = 0) THEN
            s_sin_rt4 <= s_sin_rt1;
            s_exp_rt4 <= s_exp_rt1(exp-1 DOWNTO 0);
            s_mant_rt4<= s_mant_rt1(mant*2+2 DOWNTO mant+1);
            s_skip1   <= '1';
ELSIF (s_exp_rt1 = 0) THEN
            s_sin_rt4 <= s_sin_rt15;
            s_exp_rt4 <= s_exp_rt15;
            s_mant_rt4<= s_mant_rt15(mant+1 DOWNTO 1) & '0';
            s_skip1   <= '1';
ELSIF (s_exp_rt15 > s_exp_rt1) THEN
            s_rsft_din <= s_mant_rt1(mant*2+2 DOWNTO mant+2);
            s_rshift   <= s_exp_rt15 - (s_exp_rt1(exp-1 DOWNTO 0));
            s_skip1   <= '0';
ELSIF (s_exp_rt15 <= s_exp_rt1) THEN
            s_rsft_din <= s_mant_rt15(mant+1 DOWNTO 1);
            s_rshift   <= (s_exp_rt1(exp-1 DOWNTO 0)) - s_exp_rt15;
            s_skip1   <= '0';
END IF;
IF ((s_exp_rt16 = 0) AND (s_exp_rt2 = 0)) THEN
            s_sin_rt5 <= '0';
            s_exp_rt5 <= (OTHERS => '0');
```

```
            s_mant_rt5<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_rt16 = 255) THEN
            s_sin_rt5 <= s_sin_rt16;
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= s_mant_rt16;
            s_skip2   <= '1';
ELSIF (s_exp_rt2 >= 255) THEN
            s_sin_rt5 <= s_sin_rt2;
            s_exp_rt5 <= (OTHERS => '1');
            s_mant_rt5<= s_mant_rt2(mant*2+2 DOWNTO mant+1);
            s_skip2   <= '1';
ELSIF (s_exp_rt16 = 0) THEN
            s_sin_rt5 <= s_sin_rt2;
            s_exp_rt5 <= s_exp_rt2(exp-1 DOWNTO 0);
            s_mant_rt5<= s_mant_rt2(mant*2+2 DOWNTO mant+1);
            s_skip2   <= '1';
ELSIF (s_exp_rt2 = 0) THEN
            s_sin_rt5 <= s_sin_rt16;
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= s_mant_rt16;
            s_skip2   <= '1';
ELSIF (s_exp_rt16 > s_exp_rt2) THEN
            s_rsft_din1 <= s_mant_rt2(mant*2+2 DOWNTO mant+2);
            s_rshift1   <= s_exp_rt16 - (s_exp_rt2(exp-1 DOWNTO 0));
            s_skip2   <= '0';
ELSIF (s_exp_rt16 <= s_exp_rt2) THEN
            s_rsft_din1 <= s_mant_rt16(mant+1 DOWNTO 1);
            s_rshift1   <= (s_exp_rt2(exp-1 DOWNTO 0)) - s_exp_rt16;
            s_skip2   <= '0';
END IF;
IF ((s_exp_rt17 = 0) AND (s_exp_rt3 = 0)) THEN
            s_sin_rt6 <= '0';
            s_exp_rt6 <= (OTHERS => '0');
            s_mant_rt6<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_rt17 = 255) THEN
            s_sin_rt6 <= s_sin_rt17;
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= s_mant_rt17;
            s_skip3   <= '1';
ELSIF (s_exp_rt3 >= 255) THEN
            s_sin_rt6 <= s_sin_rt3;
            s_exp_rt6 <= (OTHERS => '1');
            s_mant_rt6<= s_mant_rt3(mant*2+2 DOWNTO mant+1);
            s_skip3   <= '1';
ELSIF (s_exp_rt17 = 0) THEN
            s_sin_rt6 <= s_sin_rt3;
            s_exp_rt6 <= s_exp_rt3(exp-1 DOWNTO 0);
            s_mant_rt6<= s_mant_rt3(mant*2+2 DOWNTO mant+1);
            s_skip3   <= '1';
ELSIF (s_exp_rt3 = 0) THEN
            s_sin_rt6 <= s_sin_rt17;
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= s_mant_rt17;
            s_skip3   <= '1';
```

```
ELSIF (s_exp_rt17 > s_exp_rt3) THEN
            s_rsft_din2 <= s_mant_rt3(mant*2+2 DOWNTO mant+2);
            s_rshift2   <= s_exp_rt17 - (s_exp_rt3(exp-1 DOWNTO 0));
            s_skip3   <= '0';
ELSIF (s_exp_rt17 <= s_exp_rt3) THEN
            s_rsft_din2 <= s_mant_rt17(mant+1 DOWNTO 1);
            s_rshift2   <= (s_exp_rt3(exp-1 DOWNTO 0)) - s_exp_rt17;
            s_skip3   <= '0';
END IF;
            s_cnt      <= s_cnt + 1;
ELSIF (s_cnt = 16) THEN
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 17) THEN
IF (s_skip1 = '0') THEN
IF (s_exp_rt15 > s_exp_rt1) THEN
s_exp_rt1 <= '0' & s_exp_rt15;
s_mant_rt1(mant*2+2 DOWNTO mant+2)<= s_rsft_dot;
ELSE
            s_exp_rt15 <= s_exp_rt1(exp-1 DOWNTO 0);
            s_mant_rt15(mant+1 DOWNTO 1)<= s_rsft_dot;
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF (s_exp_rt16 > s_exp_rt2) THEN
            s_exp_rt2 <= '0' & s_exp_rt16;
            s_mant_rt2(mant*2+2 DOWNTO mant+2)<= s_rsft_dot1;
ELSE
            s_exp_rt16 <= s_exp_rt2(exp-1 DOWNTO 0);
            s_mant_rt16(mant+1 DOWNTO 1)<= s_rsft_dot1;
END IF;
END IF;
IF (s_skip3 = '0') THEN
IF (s_exp_rt17 > s_exp_rt3) THEN
s_exp_rt3 <= '0' & s_exp_rt17;
            s_mant_rt3(mant*2+2 DOWNTO mant+2)<= s_rsft_dot2;
ELSE
            s_exp_rt17 <= s_exp_rt3(exp-1 DOWNTO 0);
            s_mant_rt17(mant+1 DOWNTO 1)<= s_rsft_dot2;
END IF;
END IF;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 18) THEN                         -- CALCULATION OF ADD
IF (s_skip1 = '0') THEN
IF ((s_sin_rt15 = '0') AND (s_sin_rt1 = '0')) THEN
            s_sin_rt4 <= '0';
            s_exp_rt4 <= s_exp_rt15;
s_mant_rt4<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) + ('0' & s_mant_rt1(mant*2+2 DOWNTO
mant+2));
s_skip1 <= '0';
ELSIF  ((s_sin_rt15 = '1') AND (s_sin_rt1 = '0')) THEN
IF (s_mant_rt15(mant+1 DOWNTO 1) > (s_mant_rt1(mant*2+2 DOWNTO mant+2))) THEN
s_sin_rt4 <= '1';
s_exp_rt4 <= s_exp_rt15;
s_mant_rt4<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) - ('0' & s_mant_rt1(mant*2+2 DOWNTO
mant+2));
s_skip1 <= '0';
```

```
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) < (s_mant_rt1(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt4 <= '0';
            s_exp_rt4 <= s_exp_rt15;
            s_mant_rt4<= ('0' & s_mant_rt1(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt15(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) = (s_mant_rt1(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt4 <= '0';
            s_exp_rt4 <= (OTHERS => '0');
            s_mant_rt4<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt15 = '0') AND (s_sin_rt1 = '1')) THEN
IF (s_mant_rt15(mant+1 DOWNTO 1) > (s_mant_rt1(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt4 <= '0';
            s_exp_rt4 <= s_exp_rt15;
            s_mant_rt4<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) - ('0' & s_mant_rt1(mant*2+2
DOWNTO mant+2));
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) < (s_mant_rt1(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt4 <= '1';
            s_exp_rt4 <= s_exp_rt15;
            s_mant_rt4<= ('0' & s_mant_rt1(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt15(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt15(mant+1 DOWNTO 1) = (s_mant_rt1(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt4 <= '0';
            s_exp_rt4 <= (OTHERS => '0');
            s_mant_rt4<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt15 = '1') AND (s_sin_rt1 = '1')) THEN
            s_sin_rt4 <= '1';
            s_exp_rt4 <= s_exp_rt15;
            s_mant_rt4<= ('0' & s_mant_rt15(mant+1 DOWNTO 1)) + ('0' & s_mant_rt1(mant*2+2
DOWNTO mant+2));
            s_skip1 <= '0';
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF ((s_sin_rt16 = '0') AND (s_sin_rt2 = '0')) THEN
            s_sin_rt5 <= '0';
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= ('0' & s_mant_rt16(mant+1 DOWNTO 1)) + ('0' & s_mant_rt2(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
ELSIF  ((s_sin_rt16 = '1') AND (s_sin_rt2 = '0')) THEN
IF (s_mant_rt16(mant+1 DOWNTO 1) > (s_mant_rt2(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt5 <= '1';
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= ('0' & s_mant_rt16(mant+1 DOWNTO 1)) - ('0' & s_mant_rt2(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
ELSIF (s_mant_rt16(mant+1 DOWNTO 1) < (s_mant_rt2(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt5 <= '0';
            s_exp_rt5 <= s_exp_rt16;
```

```
            s_mant_rt5<= ('0' & s_mant_rt2(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt16(mant+1
DOWNTO 1));
            s_skip2 <= '0';
ELSIF (s_mant_rt16(mant+1 DOWNTO 1) = (s_mant_rt2(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt5 <= '0';
            s_exp_rt5 <= (OTHERS => '0');
            s_mant_rt5<= (OTHERS => '0');
            s_skip2 <= '1';
END IF;
ELSIF  ((s_sin_rt16 = '0') AND (s_sin_rt2 = '1')) THEN
IF (s_mant_rt16(mant+1 DOWNTO 1) > (s_mant_rt2(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt5 <= '0';
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= ('0' & s_mant_rt16(mant+1 DOWNTO 1)) - ('0' & s_mant_rt2(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
ELSIF (s_mant_rt16(mant+1 DOWNTO 1) < (s_mant_rt2(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt5 <= '1';
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= ('0' & s_mant_rt2(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt16(mant+1
DOWNTO 1));
            s_skip2 <= '0';
ELSIF (s_mant_rt16(mant+1 DOWNTO 1) = (s_mant_rt2(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt5 <= '0';
            s_exp_rt5 <= (OTHERS => '0');
            s_mant_rt5<= (OTHERS => '0');
            s_skip2 <= '1';
END IF;
ELSIF  ((s_sin_rt16 = '1') AND (s_sin_rt2 = '1')) THEN
            s_sin_rt5 <= '1';
            s_exp_rt5 <= s_exp_rt16;
            s_mant_rt5<= ('0' & s_mant_rt16(mant+1 DOWNTO 1)) + ('0' & s_mant_rt2(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
END IF;
END IF;
IF (s_skip3 = '0') THEN
IF ((s_sin_rt17 = '0') AND (s_sin_rt3 = '0')) THEN
            s_sin_rt6 <= '0';
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= ('0' & s_mant_rt17(mant+1 DOWNTO 1)) + ('0' & s_mant_rt3(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
ELSIF  ((s_sin_rt17 = '1') AND (s_sin_rt3 = '0')) THEN
IF (s_mant_rt17(mant+1 DOWNTO 1) > (s_mant_rt3(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt6 <= '1';
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= ('0' & s_mant_rt17(mant+1 DOWNTO 1)) - ('0' & s_mant_rt3(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
ELSIF (s_mant_rt17(mant+1 DOWNTO 1) < (s_mant_rt3(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt6 <= '0';
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= ('0' & s_mant_rt3(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt17(mant+1
DOWNTO 1));
            s_skip3 <= '0';
```

```
ELSIF (s_mant_rt17(mant+1 DOWNTO 1) = (s_mant_rt3(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt6 <= '0';
            s_exp_rt6 <= (OTHERS => '0');
            s_mant_rt6<= (OTHERS => '0');
            s_skip3 <= '1';
END IF;
ELSIF  ((s_sin_rt17 = '0') AND (s_sin_rt3 = '1')) THEN
IF (s_mant_rt17(mant+1 DOWNTO 1) > (s_mant_rt3(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt6 <= '0';
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= ('0' & s_mant_rt17(mant+1 DOWNTO 1)) - ('0' & s_mant_rt3(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
ELSIF (s_mant_rt17(mant+1 DOWNTO 1) < (s_mant_rt3(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt6 <= '1';
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= ('0' & s_mant_rt3(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt17(mant+1
DOWNTO 1));
            s_skip3 <= '0';
ELSIF (s_mant_rt17(mant+1 DOWNTO 1) = (s_mant_rt3(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt6 <= '0';
            s_exp_rt6 <= (OTHERS => '0');
            s_mant_rt6<= (OTHERS => '0');
            s_skip3 <= '1';
END IF;
ELSIF  ((s_sin_rt17 = '1') AND (s_sin_rt3 = '1')) THEN
            s_sin_rt6 <= '1';
            s_exp_rt6 <= s_exp_rt17;
            s_mant_rt6<= ('0' & s_mant_rt17(mant+1 DOWNTO 1)) + ('0' & s_mant_rt3(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
END IF;
END IF;
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 19) THEN
IF (s_skip1 = '0') THEN
            s_lshift_min(mant*2+2 DOWNTO mant+1) <= s_mant_rt4;
            s_lshift_min(mant DOWNTO 0)          <= (OTHERS => '0');
            s_lshift_ein                         <= s_exp_rt4;
END IF;
IF (s_skip2 = '0') THEN
            s_lshift_min1(mant*2+2 DOWNTO mant+1) <= s_mant_rt5;
            s_lshift_min1(mant DOWNTO 0)          <= (OTHERS => '0');
            s_lshift_ein1                         <= s_exp_rt5;
END IF;
IF (s_skip3 = '0') THEN
            s_lshift_min2(mant*2+2 DOWNTO mant+1) <= s_mant_rt6;
            s_lshift_min2(mant DOWNTO 0)          <= (OTHERS => '0');
            s_lshift_ein2                         <= s_exp_rt6;
END IF;
            s_cnt                               <= s_cnt + 1;
ELSIF (s_cnt = 20) THEN
            s_cnt <= s_cnt + 1;                                  -- WAITING FOR SHIFTING BITS
ELSIF (s_cnt = 21) THEN
IF (s_skip1 = '0') THEN
            s_exp_rt4 <= s_lshift_eot + 1;
```

```
            s_mant_rt4<= s_lshift_mot(mant*2+2 DOWNTO mant+1);
END IF;
IF (s_skip2 = '0') THEN
            s_exp_rt5 <= s_lshift_eot1 + 1;
            s_mant_rt5<= s_lshift_mot1(mant*2+2 DOWNTO mant+1);
END IF;
IF (s_skip3 = '0') THEN
            s_exp_rt6 <= s_lshift_eot2 + 1;
            s_mant_rt6<= s_lshift_mot2(mant*2+2 DOWNTO mant+1);
END IF;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 22) THEN
            xh1_ot <= s_sin_rt4 & s_exp_rt4(exp-1 DOWNTO 0) & s_mant_rt4(mant DOWNTO 1);
            xh2_ot <= s_sin_rt5 & s_exp_rt5(exp-1 DOWNTO 0) & s_mant_rt5(mant DOWNTO 1);
            xh3_ot <= s_sin_rt6 & s_exp_rt6(exp-1 DOWNTO 0) & s_mant_rt6(mant DOWNTO 1);
IF (s_exp_u = 0) THEN
            s_sin_rt7 <= '0';
            s_exp_rt7 <= (OTHERS => '0');
            s_mant_rt7<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_gamma1 = 0) THEN
            s_sin_rt7 <= '0';
            s_exp_rt7 <= (OTHERS => '0');
            s_mant_rt7<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_u = 255) THEN
            s_sin_rt7 <= '0';
            s_exp_rt7 <= (OTHERS => '1');
            s_mant_rt7<= (OTHERS => '0');
            s_skip1   <= '1';
ELSIF (s_exp_gamma1 = 255) THEN
            s_sin_rt7 <= '0';
            s_exp_rt7 <= (OTHERS => '1');
            s_mant_rt7<= (OTHERS => '0');
            s_skip1   <= '1';
ELSE
            s_skip1   <= '0';
END IF;
IF (s_exp_rt5 = 0) THEN
            s_sin_rt8 <= '0';
            s_exp_rt8 <= (OTHERS => '0');
            s_mant_rt8<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_ts = 0) THEN
            s_sin_rt8 <= '0';
            s_exp_rt8 <= (OTHERS => '0');
            s_mant_rt8<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_rt5 = 255) THEN
            s_sin_rt8 <= '0';
            s_exp_rt8 <= (OTHERS => '1');
            s_mant_rt8<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_ts = 255) THEN
            s_sin_rt8 <= '0';
            s_exp_rt8 <= (OTHERS => '1');
```

```
            s_mant_rt8<= (OTHERS => '0');
            s_skip2   <= '1';
ELSE
            s_skip2   <= '0';
END IF;
IF (s_exp_rt6 = 0) THEN
            s_sin_rt9 <= '0';
            s_exp_rt9 <= (OTHERS => '0');
            s_mant_rt9<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_phi = 0) THEN
            s_sin_rt9 <= '0';
            s_exp_rt9 <= (OTHERS => '0');
            s_mant_rt9<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_rt6 = 255) THEN
            s_sin_rt9 <= '0';
            s_exp_rt9 <= (OTHERS => '1');
            s_mant_rt9<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_phi = 255) THEN
            s_sin_rt9 <= '0';
            s_exp_rt9 <= (OTHERS => '1');
            s_mant_rt9<= (OTHERS => '0');
            s_skip3   <= '1';
ELSE
            s_skip3   <= '0';
END IF;
IF (s_exp_u = 0) THEN
            s_sin_rt10 <= '0';
            s_exp_rt10 <= (OTHERS => '0');
            s_mant_rt10<= (OTHERS => '0');
            s_skip4   <= '1';
ELSIF (s_exp_gamma2 = 0) THEN
            s_sin_rt10 <= '0';
            s_exp_rt10 <= (OTHERS => '0');
            s_mant_rt10<= (OTHERS => '0');
            s_skip4   <= '1';
ELSIF (s_exp_u = 255) THEN
            s_sin_rt10 <= '0';
            s_exp_rt10 <= (OTHERS => '1');
            s_mant_rt10<= (OTHERS => '0');
            s_skip4   <= '1';
ELSIF (s_exp_gamma2 = 255) THEN
            s_sin_rt10 <= '0';
            s_exp_rt10 <= (OTHERS => '1');
            s_mant_rt10<= (OTHERS => '0');
            s_skip4   <= '1';
ELSE
            s_skip4   <= '0';
END IF;
IF (s_exp_rt6 = 0) THEN
            s_sin_rt11 <= '0';
            s_exp_rt11 <= (OTHERS => '0');
            s_mant_rt11<= (OTHERS => '0');
            s_skip5   <= '1';
```

```
ELSIF (s_exp_ts = 0) THEN
        s_sin_rt11 <= '0';
        s_exp_rt11 <= (OTHERS => '0');
        s_mant_rt11<= (OTHERS => '0');
        s_skip5   <= '1';
ELSIF (s_exp_rt6 = 255) THEN
        s_sin_rt11 <= '0';
        s_exp_rt11 <= (OTHERS => '1');
        s_mant_rt11<= (OTHERS => '0');
        s_skip5   <= '1';
ELSIF (s_exp_ts = 255) THEN
        s_sin_rt11 <= '0';
        s_exp_rt11 <= (OTHERS => '1');
        s_mant_rt11<= (OTHERS => '0');
        s_skip5   <= '1';
ELSE
        s_skip5   <= '0';
END IF;
        s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 23) THEN
IF (s_skip1 = '0') THEN
        s_sin_rt7 <= s_sin_u XOR s_sin_gamma1;
        s_exp_rt7 <= ('0' & s_exp_u) + ('0' & s_exp_gamma1) - base;
        s_mant_rt7<= (s_mant_u & '0') * s_mant_gamma1;
END IF;
IF (s_skip2 = '0') THEN
        s_sin_rt8 <= s_sin_rt5 XOR s_sin_ts;
        s_exp_rt8 <= ('0' & s_exp_rt5) + ('0' & s_exp_ts) - base;
        s_mant_rt8<= s_mant_rt5 * s_mant_ts;
END IF;
IF (s_skip3 = '0') THEN
        s_sin_rt9 <= s_sin_rt6 XOR s_sin_phi;
        s_exp_rt9 <= ('0' & s_exp_rt6) + ('0' & s_exp_phi) - base;
        s_mant_rt9<= s_mant_rt6 * s_mant_phi;
END IF;
IF (s_skip4 = '0') THEN
        s_sin_rt10 <= s_sin_u XOR s_sin_gamma2;
        s_exp_rt10 <= ('0' & s_exp_u) + ('0' & s_exp_gamma2) - base;
        s_mant_rt10<= (s_mant_u & '0') * s_mant_gamma2;
END IF;
IF (s_skip5 = '0') THEN
        s_sin_rt11 <= s_sin_rt6 XOR s_sin_ts;
        s_exp_rt11 <= ('0' & s_exp_rt6) + ('0' & s_exp_ts) - base;
        s_mant_rt11<= s_mant_rt6 * s_mant_ts;
END IF;
        s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 24) THEN
IF (s_skip1 = '0') THEN
IF (s_exp_rt7 < 233) THEN
        s_lshift_min <= s_mant_rt7;
        s_lshift_ein <= s_exp_rt7(exp-1 DOWNTO 0);
        s_skip1      <= '0';
ELSE
        s_mant_rt7 <= (OTHERS => '0');
        s_exp_rt7  <= (OTHERS => '1');
        s_skip1    <= '1';
```

```
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF (s_exp_rt8 < 233) THEN
            s_lshift_min1 <= s_mant_rt8;
            s_lshift_ein1 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_skip2       <= '0';
ELSE
            s_mant_rt8 <= (OTHERS => '0');
            s_exp_rt8  <= (OTHERS => '1');
            s_skip2    <= '1';
END IF;
END IF;
IF (s_skip3 = '0') THEN
IF (s_exp_rt9 < 233) THEN
            s_lshift_min2 <= s_mant_rt9;
            s_lshift_ein2 <= s_exp_rt9(exp-1 DOWNTO 0);
            s_skip3       <= '0';
ELSE
            s_mant_rt9 <= (OTHERS => '0');
            s_exp_rt9  <= (OTHERS => '1');
            s_skip3    <= '1';
END IF;
END IF;
            s_cnt       <= s_cnt + 1;
ELSIF (s_cnt = 25) THEN
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 26) THEN
IF (s_skip1 = '0') THEN
            s_exp_rt7 <= '0' & (s_lshift_eot + 1);
            s_mant_rt7<= s_lshift_mot(mant*2+2 DOWNTO 0);
END IF;
IF (s_skip2 = '0') THEN
            s_exp_rt8 <= '0' & (s_lshift_eot1 + 1);
            s_mant_rt8<= s_lshift_mot1(mant*2+2 DOWNTO 0);
END IF;
IF (s_skip3 = '0') THEN
            s_exp_rt9 <= '0' & (s_lshift_eot2 + 1);
            s_mant_rt9<= s_lshift_mot2(mant*2+2 DOWNTO 0);
END IF;
IF (s_skip4 = '0') THEN
IF (s_exp_rt7 < 233) THEN
            s_lshift_min <= s_mant_rt10;
            s_lshift_ein <= s_exp_rt10(exp-1 DOWNTO 0);
            s_skip4      <= '0';
ELSE
            s_mant_rt10 <= (OTHERS => '0');
            s_exp_rt10  <= (OTHERS => '1');
            s_skip4    <= '1';
END IF;
END IF;
IF (s_skip5 = '0') THEN
IF (s_exp_rt11 < 233) THEN
            s_lshift_min1 <= s_mant_rt11;
            s_lshift_ein1 <= s_exp_rt11(exp-1 DOWNTO 0);
            s_skip5       <= '0';
```

```
ELSE
        s_mant_rt11 <= (OTHERS => '0');
        s_exp_rt11  <= (OTHERS => '1');
        s_skip5    <= '1';
END IF;
END IF;
        s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 27) THEN
        s_cnt  <= s_cnt + 1;
ELSIF (s_cnt = 28) THEN
IF (s_skip4 = '0') THEN
        s_exp_rt10 <= '0' & (s_lshift_eot + 1);
        s_mant_rt10<= s_lshift_mot(mant*2+2 DOWNTO 0);
END IF;
IF (s_skip5 = '0') THEN
        s_exp_rt11 <= '0' & (s_lshift_eot1 + 1);
        s_mant_rt11<= s_lshift_mot1(mant*2+2 DOWNTO 0);
END IF;
        s_cnt  <= s_cnt + 1;
ELSIF (s_cnt = 29) THEN                                 -- COMPARASON
IF ((s_exp_rt4 = 0) AND (s_exp_rt7 = 0)) THEN
        s_sin_rt12 <= '0';
        s_exp_rt12 <= (OTHERS => '0');
        s_mant_rt12<= (OTHERS => '0');
        s_skip1   <= '1';
ELSIF (s_exp_rt4 = 255) THEN
        s_sin_rt12 <= s_sin_rt4;
        s_exp_rt12 <= s_exp_rt4;
        s_mant_rt12<= s_mant_rt4;
        s_skip1   <= '1';
ELSIF (s_exp_rt7 >= 255) THEN
        s_sin_rt12 <= s_sin_rt7;
        s_exp_rt12 <= (OTHERS => '1');
        s_mant_rt12<= s_mant_rt7(mant*2+2 DOWNTO mant+1);
        s_skip1   <= '1';
ELSIF (s_exp_rt4 = 0) THEN
        s_sin_rt12 <= s_sin_rt7;
        s_exp_rt12 <= s_exp_rt7(exp-1 DOWNTO 0);
        s_mant_rt12<= s_mant_rt7(mant*2+2 DOWNTO mant+1);
        s_skip1   <= '1';
ELSIF (s_exp_rt7 = 0) THEN
        s_sin_rt12 <= s_sin_rt4;
        s_exp_rt12 <= s_exp_rt4;
        s_mant_rt12<= s_mant_rt4;
        s_skip1   <= '1';
ELSIF (s_exp_rt4 > s_exp_rt7) THEN
        s_rsft_din <= s_mant_rt7(mant*2+2 DOWNTO mant+2);
        s_rshift   <= s_exp_rt4 - (s_exp_rt7(exp-1 DOWNTO 0));
        s_skip1   <= '0';
ELSIF (s_exp_rt4 <= s_exp_rt7) THEN
        s_rsft_din <= s_mant_rt4(mant+1 DOWNTO 1);
        s_rshift   <= (s_exp_rt7(exp-1 DOWNTO 0)) - s_exp_rt4;
        s_skip1   <= '0';
END IF;
IF ((s_exp_rt8 = 0) AND (s_exp_rt9 = 0)) THEN
        s_sin_rt13 <= '0';
```

```
            s_exp_rt13 <= (OTHERS => '0');
            s_mant_rt13<= (OTHERS => '0');
            s_skip2   <= '1';
ELSIF (s_exp_rt8 >= 255) THEN
            s_sin_rt13 <= s_sin_rt8;
            s_exp_rt13 <= (OTHERS => '1');
            s_mant_rt13<= s_mant_rt8(mant*2+2 DOWNTO mant+1);
            s_skip2   <= '1';
ELSIF (s_exp_rt9 >= 255) THEN
            s_sin_rt13 <= s_sin_rt9;
            s_exp_rt13 <= (OTHERS => '1');
            s_mant_rt13<= s_mant_rt9(mant*2+2 DOWNTO mant+1);
            s_skip2   <= '1';
ELSIF (s_exp_rt8 = 0) THEN
            s_sin_rt13 <= s_sin_rt9;
            s_exp_rt13 <= s_exp_rt9(exp-1 DOWNTO 0);
            s_mant_rt13<= s_mant_rt9(mant*2+2 DOWNTO mant+1);
            s_skip2   <= '1';
ELSIF (s_exp_rt9 = 0) THEN
            s_sin_rt13 <= s_sin_rt8;
            s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_mant_rt13<= s_mant_rt8(mant*2+2 DOWNTO mant+1);
            s_skip2   <= '1';
ELSIF (s_exp_rt8 > s_exp_rt9) THEN
            s_rsft_din1 <= s_mant_rt9(mant*2+2 DOWNTO mant+2);
            s_rshift1   <= (s_exp_rt8(exp-1 DOWNTO 0)) - (s_exp_rt9(exp-1 DOWNTO 0));
            s_skip2   <= '0';
ELSIF (s_exp_rt8 <= s_exp_rt9) THEN
            s_rsft_din1 <= s_mant_rt8(mant*2+2 DOWNTO mant+2);
            s_rshift1   <= (s_exp_rt9(exp-1 DOWNTO 0)) - (s_exp_rt8(exp-1 DOWNTO 0));
            s_skip2   <= '0';
END IF;
IF ((s_exp_rt5 = 0) AND (s_exp_rt11 = 0)) THEN
            s_sin_rt14 <= '0';
            s_exp_rt14 <= (OTHERS => '0');
            s_mant_rt14<= (OTHERS => '0');
            s_skip3   <= '1';
ELSIF (s_exp_rt5 = 255) THEN
            s_sin_rt14 <= s_sin_rt5;
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= s_mant_rt5;
            s_skip3   <= '1';
ELSIF (s_exp_rt11 >= 255) THEN
            s_sin_rt14 <= s_sin_rt11;
            s_exp_rt14 <= (OTHERS => '1');
            s_mant_rt14<= s_mant_rt11(mant*2+2 DOWNTO mant+1);
            s_skip3   <= '1';
ELSIF (s_exp_rt5 = 0) THEN
            s_sin_rt14 <= s_sin_rt11;
            s_exp_rt14 <= s_exp_rt11(exp-1 DOWNTO 0);
            s_mant_rt14<= s_mant_rt11(mant*2+2 DOWNTO mant+1);
            s_skip3   <= '1';
ELSIF (s_exp_rt11 = 0) THEN
            s_sin_rt14 <= s_sin_rt5;
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= s_mant_rt5;
```

```
            s_skip3   <= '1';
ELSIF (s_exp_rt5 > s_exp_rt11) THEN
            s_rsft_din2 <= s_mant_rt11(mant*2+2 DOWNTO mant+2);
            s_rshift2   <= s_exp_rt5 - (s_exp_rt11(exp-1 DOWNTO 0));
            s_skip3   <= '0';
ELSIF (s_exp_rt5 <= s_exp_rt11) THEN
            s_rsft_din2 <= s_mant_rt5(mant+1 DOWNTO 1);
            s_rshift2   <= (s_exp_rt11(exp-1 DOWNTO 0)) - s_exp_rt5;
            s_skip3   <= '0';
END IF;
            s_cnt      <= s_cnt + 1;
ELSIF (s_cnt = 30) THEN
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 31) THEN
IF (s_skip1 = '0') THEN
IF (s_exp_rt4 > s_exp_rt7) THEN
            s_exp_rt7 <= '0' & s_exp_rt4;
            s_mant_rt7(mant*2+2 DOWNTO mant+2)<= s_rsft_dot;
ELSE
            s_exp_rt4 <= s_exp_rt7(exp-1 DOWNTO 0);
            s_mant_rt4<= s_rsft_dot & '0';
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF (s_exp_rt8 > s_exp_rt9) THEN
            s_exp_rt9 <= s_exp_rt8;
            s_mant_rt9(mant*2+2 DOWNTO mant+2)<= s_rsft_dot1;
ELSE
            s_exp_rt8 <= s_exp_rt9;
            s_mant_rt8(mant*2+2 DOWNTO mant+2)<= s_rsft_dot1;
END IF;
END IF;
IF (s_skip3 = '0') THEN
IF (s_exp_rt5 > s_exp_rt11) THEN
            s_exp_rt11 <= '0' & s_exp_rt5;
            s_mant_rt11(mant*2+2 DOWNTO mant+2)<= s_rsft_dot2;
ELSE
            s_exp_rt5 <= s_exp_rt11(exp-1 DOWNTO 0);
            s_mant_rt5<= s_rsft_dot2 & '0';
END IF;
END IF;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 32) THEN                         -- CALCULATION OF ADD
IF (s_skip1 = '0') THEN
IF ((s_sin_rt4 = '0') AND (s_sin_rt7 = '0')) THEN
            s_sin_rt12 <= '0';
            s_exp_rt12 <= s_exp_rt4;
            s_mant_rt12<= ('0' & s_mant_rt4(mant+1 DOWNTO 1)) + ('0' & s_mant_rt7(mant*2+2
DOWNTO mant+2));
            s_skip1 <= '0';
ELSIF  ((s_sin_rt4 = '1') AND (s_sin_rt7 = '0')) THEN
IF (s_mant_rt4(mant+1 DOWNTO 1) > (s_mant_rt7(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt12 <= '1';
            s_exp_rt12 <= s_exp_rt4;
            s_mant_rt12<= ('0' & s_mant_rt4(mant+1 DOWNTO 1)) - ('0' & s_mant_rt7(mant*2+2
DOWNTO mant+2));
```

```
            s_skip1 <= '0';
ELSIF (s_mant_rt4(mant+1 DOWNTO 1) < (s_mant_rt7(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt12 <= '0';
            s_exp_rt12 <= s_exp_rt4;
            s_mant_rt12<= ('0' & s_mant_rt7(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt4(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt4(mant+1 DOWNTO 1) = (s_mant_rt7(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt12 <= '0';
            s_exp_rt12 <= (OTHERS => '0');
            s_mant_rt12<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt4 = '0') AND (s_sin_rt7 = '1')) THEN
IF (s_mant_rt4(mant+1 DOWNTO 1) > (s_mant_rt7(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt12 <= '0';
            s_exp_rt12 <= s_exp_rt4;
            s_mant_rt12<= ('0' & s_mant_rt4(mant+1 DOWNTO 1)) - ('0' & s_mant_rt7(mant*2+2
DOWNTO mant+2));
            s_skip1 <= '0';
            ELSIF (s_mant_rt4(mant+1 DOWNTO 1) < (s_mant_rt7(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt12 <= '1';
            s_exp_rt12 <= s_exp_rt4;
            s_mant_rt12<= ('0' & s_mant_rt7(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt4(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt4(mant+1 DOWNTO 1) = (s_mant_rt7(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt12 <= '0';
            s_exp_rt12 <= (OTHERS => '0');
            s_mant_rt12<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt4 = '1') AND (s_sin_rt7 = '1')) THEN
            s_sin_rt12 <= '1';
            s_exp_rt12 <= s_exp_rt4;
            s_mant_rt12<= ('0' & s_mant_rt4(mant+1 DOWNTO 1)) + ('0' & s_mant_rt7(mant*2+2
DOWNTO mant+2));
            s_skip1 <= '0';
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF ((s_sin_rt8 = '0') AND (s_sin_rt9 = '0')) THEN
            s_sin_rt13 <= '0';
            s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_mant_rt13<= ('0' & s_mant_rt8(mant*2+2 DOWNTO mant+2)) + ('0' & s_mant_rt9(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
ELSIF  ((s_sin_rt8 = '1') AND (s_sin_rt9 = '0')) THEN
IF ((s_mant_rt8(mant*2+2 DOWNTO mant+2)) > (s_mant_rt9(mant*2+2 DOWNTO mant+2))) THEN
s_sin_rt13 <= '1';
s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
s_mant_rt13<= ('0' & s_mant_rt8(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt9(mant*2+2
DOWNTO mant+2));
s_skip2 <= '0';
ELSIF ((s_mant_rt8(mant*2+2 DOWNTO mant+2)) < (s_mant_rt9(mant*2+2 DOWNTO mant+2)))
THEN
```

```
            s_sin_rt13 <= '0';
            s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_mant_rt13<= ('0' & s_mant_rt9(mant*2+2 DOWNTO mant+2)) - ('0' & (s_mant_rt8(mant*2+2
DOWNTO mant+2)));
            s_skip2 <= '0';
ELSIF ((s_mant_rt8(mant*2+2 DOWNTO mant+2)) = (s_mant_rt9(mant*2+2 DOWNTO mant+2)))
THEN
            s_sin_rt13 <= '0';
            s_exp_rt13 <= (OTHERS => '0');
            s_mant_rt13<= (OTHERS => '0');
            s_skip2 <= '1';
END IF;
ELSIF  ((s_sin_rt8 = '0') AND (s_sin_rt9 = '1')) THEN
IF ((s_mant_rt8(mant*2+2 DOWNTO mant+2)) > (s_mant_rt9(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt13 <= '0';
            s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_mant_rt13<= ('0' & s_mant_rt8(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt9(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
ELSIF ((s_mant_rt8(mant*2+2 DOWNTO mant+2)) < (s_mant_rt9(mant*2+2 DOWNTO mant+2)))
THEN
            s_sin_rt13 <= '1';
            s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_mant_rt13<= ('0' & s_mant_rt9(mant*2+2 DOWNTO mant+2)) - ('0' & (s_mant_rt8(mant*2+2
DOWNTO mant+2)));
            s_skip2 <= '0';
ELSIF ((s_mant_rt8(mant*2+2 DOWNTO mant+2)) = (s_mant_rt9(mant*2+2 DOWNTO mant+2)))
THEN
            s_sin_rt13 <= '0';
            s_exp_rt13 <= (OTHERS => '0');
            s_mant_rt13<= (OTHERS => '0');
            s_skip2 <= '1';
END IF;
ELSIF  ((s_sin_rt8 = '1') AND (s_sin_rt9 = '1')) THEN
            s_sin_rt13 <= '1';
            s_exp_rt13 <= s_exp_rt8(exp-1 DOWNTO 0);
            s_mant_rt13<= ('0' & s_mant_rt8(mant*2+2 DOWNTO mant+2)) + ('0' & s_mant_rt9(mant*2+2
DOWNTO mant+2));
            s_skip2 <= '0';
END IF;
END IF;
IF (s_skip3 = '0') THEN
IF ((s_sin_rt5 = '0') AND (s_sin_rt11 = '0')) THEN
            s_sin_rt14 <= '0';
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= ('0' & s_mant_rt5(mant+1 DOWNTO 1)) + ('0' & s_mant_rt11(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
ELSIF  ((s_sin_rt5 = '1') AND (s_sin_rt11 = '0')) THEN
IF (s_mant_rt5(mant+1 DOWNTO 1) > (s_mant_rt11(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt14 <= '1';
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= ('0' & s_mant_rt5(mant+1 DOWNTO 1)) - ('0' & s_mant_rt11(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
ELSIF (s_mant_rt5(mant+1 DOWNTO 1) < (s_mant_rt11(mant*2+2 DOWNTO mant+2))) THEN
```

```
            s_sin_rt14 <= '0';
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= ('0' & s_mant_rt11(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt5(mant+1
DOWNTO 1));
            s_skip3 <= '0';
ELSIF (s_mant_rt5(mant+1 DOWNTO 1) = (s_mant_rt11(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt14 <= '0';
            s_exp_rt14 <= (OTHERS => '0');
            s_mant_rt14<= (OTHERS => '0');
            s_skip3 <= '1';
END IF;
ELSIF  ((s_sin_rt5 = '0') AND (s_sin_rt11 = '1')) THEN
IF (s_mant_rt5(mant+1 DOWNTO 1) > (s_mant_rt11(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt14 <= '0';
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= ('0' & s_mant_rt5(mant+1 DOWNTO 1)) - ('0' & s_mant_rt11(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
ELSIF (s_mant_rt5(mant+1 DOWNTO 1) < (s_mant_rt11(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt14 <= '1';
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= ('0' & s_mant_rt11(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt5(mant+1
DOWNTO 1));
            s_skip3 <= '0';
ELSIF (s_mant_rt5(mant+1 DOWNTO 1) = (s_mant_rt11(mant*2+2 DOWNTO mant+2))) THEN
            s_sin_rt14 <= '0';
            s_exp_rt14 <= (OTHERS => '0');
            s_mant_rt14<= (OTHERS => '0');
            s_skip3 <= '1';
END IF;
ELSIF  ((s_sin_rt5 = '1') AND (s_sin_rt11 = '1')) THEN
            s_sin_rt14 <= '1';
            s_exp_rt14 <= s_exp_rt5;
            s_mant_rt14<= ('0' & s_mant_rt5(mant+1 DOWNTO 1)) + ('0' & s_mant_rt11(mant*2+2
DOWNTO mant+2));
            s_skip3 <= '0';
END IF;
END IF;
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 33) THEN
IF (s_skip1 = '0') THEN
            s_lshift_min(mant*2+2 DOWNTO mant+1) <= s_mant_rt12;
            s_lshift_min(mant DOWNTO 0)         <= (OTHERS => '0');
            s_lshift_ein                        <= s_exp_rt12;
END IF;
IF (s_skip2 = '0') THEN
            s_lshift_min1(mant*2+2 DOWNTO mant+1) <= s_mant_rt13;
            s_lshift_min1(mant DOWNTO 0)         <= (OTHERS => '0');
            s_lshift_ein1                        <= s_exp_rt13;
END IF;
IF (s_skip3 = '0') THEN
            s_lshift_min2(mant*2+2 DOWNTO mant+1) <= s_mant_rt14;
            s_lshift_min2(mant DOWNTO 0)         <= (OTHERS => '0');
            s_lshift_ein2                        <= s_exp_rt14;
END IF;
            s_cnt                               <= s_cnt + 1;
```

```
ELSIF (s_cnt = 34) THEN
        s_cnt <= s_cnt + 1;                             -- WAITING FOR SHIFTING BITS
ELSIF (s_cnt = 35) THEN
IF (s_skip1 = '0') THEN
        s_exp_rt12 <= s_lshift_eot + 1;
        s_mant_rt12<= s_lshift_mot(mant*2+2 DOWNTO mant+1);
END IF;
IF (s_skip2 = '0') THEN
        s_exp_rt13 <= s_lshift_eot1 + 1;
        s_mant_rt13<= s_lshift_mot1(mant*2+2 DOWNTO mant+1);
END IF;
IF (s_skip3 = '0') THEN
        s_exp_rt14 <= s_lshift_eot2 + 1;
        s_mant_rt14<= s_lshift_mot2(mant*2+2 DOWNTO mant+1);
END IF;
        s_cnt    <= s_cnt + 1;
ELSIF (s_cnt = 36) THEN                                 -- COMPARASON
IF ((s_exp_rt12 = 0) AND (s_exp_rt13 = 0)) THEN
        s_sin_rt15 <= '0';
        s_exp_rt15 <= (OTHERS => '0');
        s_mant_rt15<= (OTHERS => '0');
        s_skip1   <= '1';
ELSIF (s_exp_rt12 = 255) THEN
        s_sin_rt15 <= s_sin_rt12;
        s_exp_rt15 <= s_exp_rt12;
        s_mant_rt15<= s_mant_rt12;
        s_skip1   <= '1';
ELSIF (s_exp_rt13 >= 255) THEN
        s_sin_rt15 <= s_sin_rt13;
        s_exp_rt15 <= (OTHERS => '1');
        s_mant_rt15<= s_mant_rt13;
        s_skip1   <= '1';
ELSIF (s_exp_rt12 = 0) THEN
        s_sin_rt15 <= s_sin_rt13;
        s_exp_rt15 <= s_exp_rt13;
        s_mant_rt15<= s_mant_rt13;
        s_skip1   <= '1';
ELSIF (s_exp_rt13 = 0) THEN
        s_sin_rt15 <= s_sin_rt12;
        s_exp_rt15 <= s_exp_rt12;
        s_mant_rt15<= s_mant_rt12;
        s_skip1   <= '1';
ELSIF (s_exp_rt12 > s_exp_rt13) THEN
        s_rsft_din <= s_mant_rt13(mant+1 DOWNTO 1);
        s_rshift   <= s_exp_rt12 - s_exp_rt13;
        s_skip1   <= '0';
ELSIF (s_exp_rt12 <= s_exp_rt13) THEN
        s_rsft_din <= s_mant_rt12(mant+1 DOWNTO 1);
        s_rshift   <= s_exp_rt13 - s_exp_rt12;
        s_skip1   <= '0';
END IF;
IF ((s_exp_rt10 = 0) AND (s_exp_rt14 = 0)) THEN
        s_sin_rt16 <= '0';
        s_exp_rt16 <= (OTHERS => '0');
        s_mant_rt16<= (OTHERS => '0');
        s_skip2   <= '1';
```

```
ELSIF (s_exp_rt10 >= 255) THEN
        s_sin_rt16 <= s_sin_rt10;
        s_exp_rt16 <= (OTHERS => '1');
        s_mant_rt16<= s_mant_rt10(mant*2+2 DOWNTO mant+1);
        s_skip2   <= '1';
ELSIF (s_exp_rt14 >= 255) THEN
        s_sin_rt16 <= s_sin_rt14;
        s_exp_rt16 <= (OTHERS => '1');
        s_mant_rt16<= s_mant_rt14;
        s_skip2   <= '1';
ELSIF (s_exp_rt10 = 0) THEN
        s_sin_rt16 <= s_sin_rt14;
        s_exp_rt16 <= s_exp_rt14;
        s_mant_rt16<= s_mant_rt14;
        s_skip2   <= '1';
ELSIF (s_exp_rt14 = 0) THEN
        s_sin_rt16 <= s_sin_rt10;
        s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
        s_mant_rt16<= s_mant_rt10(mant*2+2 DOWNTO mant+1);
        s_skip2   <= '1';
ELSIF (s_exp_rt10 > s_exp_rt14) THEN
        s_rsft_din1 <= s_mant_rt14(mant+1 DOWNTO 1);
        s_rshift1   <= (s_exp_rt10(exp-1 DOWNTO 0) - s_exp_rt14);
        s_skip2   <= '0';
ELSIF (s_exp_rt10 <= s_exp_rt14) THEN
        s_rsft_din1 <= s_mant_rt10(mant*2+2 DOWNTO mant+2);
        s_rshift1   <= (s_exp_rt14 - s_exp_rt10(exp-1 DOWNTO 0));
        s_skip2   <= '0';
END IF;
        s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 37) THEN
        s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 38) THEN
IF (s_skip1 = '0') THEN
IF (s_exp_rt12 > s_exp_rt13) THEN
        s_exp_rt13 <= s_exp_rt12;
        s_mant_rt13<= s_rsft_dot & '0';
ELSE
        s_exp_rt12 <= s_exp_rt13(exp-1 DOWNTO 0);
        s_mant_rt12<= s_rsft_dot & '0';
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF (s_exp_rt10 > s_exp_rt14) THEN
        s_exp_rt14 <= s_exp_rt10(exp-1 DOWNTO 0);
        s_mant_rt14<= s_rsft_dot1 & '0';
ELSE
        s_exp_rt10 <= '0' & s_exp_rt14;
        s_mant_rt10(mant*2+2 DOWNTO mant+1)<= s_rsft_dot1 & '0';
END IF;
END IF;
        s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 39) THEN                    -- CALCULATION OF ADD
IF (s_skip1 = '0') THEN
IF ((s_sin_rt12 = '0') AND (s_sin_rt13 = '0')) THEN
        s_sin_rt15 <= '0';
```

```
            s_exp_rt15 <= s_exp_rt12;
            s_mant_rt15<= ('0' & s_mant_rt12(mant+1 DOWNTO 1)) + ('0' & s_mant_rt13(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF  ((s_sin_rt12 = '1') AND (s_sin_rt13 = '0')) THEN
IF (s_mant_rt12 > s_mant_rt13) THEN
            s_sin_rt15 <= '1';
            s_exp_rt15 <= s_exp_rt12;
            s_mant_rt15<= ('0' & s_mant_rt12(mant+1 DOWNTO 1)) - ('0' & s_mant_rt13(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt12 < s_mant_rt13) THEN
            s_sin_rt15 <= '0';
            s_exp_rt15 <= s_exp_rt12;
            s_mant_rt15<= ('0' & s_mant_rt13(mant+1 DOWNTO 1)) - ('0' & s_mant_rt12(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt12 = s_mant_rt13) THEN
            s_sin_rt15 <= '0';
            s_exp_rt15 <= (OTHERS => '0');
            s_mant_rt15<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt12 = '0') AND (s_sin_rt13 = '1')) THEN
IF (s_mant_rt12 > s_mant_rt13) THEN
            s_sin_rt15 <= '0';
            s_exp_rt15 <= s_exp_rt12;
            s_mant_rt15<= ('0' & s_mant_rt12(mant+1 DOWNTO 1)) - ('0' & s_mant_rt13(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt12 < s_mant_rt13) THEN
            s_sin_rt15 <= '1';
            s_exp_rt15 <= s_exp_rt12;
            s_mant_rt15<= ('0' & s_mant_rt13(mant+1 DOWNTO 1)) - ('0' & s_mant_rt12(mant+1
DOWNTO 1));
            s_skip1 <= '0';
ELSIF (s_mant_rt12 = s_mant_rt13) THEN
            s_sin_rt15 <= '0';
            s_exp_rt15 <= (OTHERS => '0');
            s_mant_rt15<= (OTHERS => '0');
            s_skip1 <= '1';
END IF;
ELSIF  ((s_sin_rt12 = '1') AND (s_sin_rt13 = '1')) THEN
            s_sin_rt15 <= '1';
            s_exp_rt15 <= s_exp_rt12;
            s_mant_rt15<= ('0' & s_mant_rt12(mant+1 DOWNTO 1)) + ('0' & s_mant_rt13(mant+1
DOWNTO 1));
            s_skip1 <= '0';
END IF;
END IF;
IF (s_skip2 = '0') THEN
IF ((s_sin_rt10 = '0') AND (s_sin_rt14 = '0')) THEN
            s_sin_rt16 <= '0';
            s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
            s_mant_rt16<= ('0' & s_mant_rt10(mant*2+2 DOWNTO mant+2)) + ('0' & s_mant_rt14(mant+1
DOWNTO 1));
```

```
            s_skip2 <= '0';
ELSIF  ((s_sin_rt10 = '1') AND (s_sin_rt14 = '0')) THEN
IF ((s_mant_rt10(mant*2+2 DOWNTO mant+2)) > (s_mant_rt14(mant+1 DOWNTO 1))) THEN
            s_sin_rt16 <= '1';
            s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
            s_mant_rt16<= ('0' & s_mant_rt10(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt14(mant+1
DOWNTO 1));
            s_skip2 <= '0';
ELSIF ((s_mant_rt10(mant*2+2 DOWNTO mant+2)) < (s_mant_rt14(mant+1 DOWNTO 1))) THEN
            s_sin_rt16 <= '0';
            s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
            s_mant_rt16<= ('0' & s_mant_rt14(mant+1 DOWNTO 1)) - ('0' & (s_mant_rt10(mant*2+2
DOWNTO mant+2)));
            s_skip2 <= '0';
ELSIF ((s_mant_rt10(mant*2+2 DOWNTO mant+2)) = (s_mant_rt14(mant+1 DOWNTO 1))) THEN
            s_sin_rt16 <= '0';
            s_exp_rt16 <= (OTHERS => '0');
            s_mant_rt16<= (OTHERS => '0');
            s_skip2 <= '1';
END IF;
ELSIF  ((s_sin_rt10 = '0') AND (s_sin_rt14 = '1')) THEN
IF ((s_mant_rt10(mant*2+2 DOWNTO mant+2)) > (s_mant_rt14(mant+1 DOWNTO 1))) THEN
            s_sin_rt16 <= '0';
            s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
            s_mant_rt16<= ('0' & s_mant_rt10(mant*2+2 DOWNTO mant+2)) - ('0' & s_mant_rt14(mant+0
DOWNTO 1));
            s_skip2 <= '0';
ELSIF ((s_mant_rt10(mant*2+2 DOWNTO mant+2)) < (s_mant_rt14(mant+1 DOWNTO 1))) THEN
            s_sin_rt16 <= '1';
            s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
            s_mant_rt16<= ('0' & s_mant_rt14(mant+1 DOWNTO 1)) - ('0' & (s_mant_rt10(mant*2+2
DOWNTO mant+2)));
            s_skip2 <= '0';
            ELSIF ((s_mant_rt10(mant*2+2 DOWNTO mant+2)) = (s_mant_rt14(mant+1 DOWNTO 1)))
THEN
            s_sin_rt16 <= '0';
            s_exp_rt16 <= (OTHERS => '0');
            s_mant_rt16<= (OTHERS => '0');
            s_skip2 <= '1';
END IF;
ELSIF  ((s_sin_rt10 = '1') AND (s_sin_rt10 = '1')) THEN
s_sin_rt16 <= '1';
s_exp_rt16 <= s_exp_rt10(exp-1 DOWNTO 0);
            s_mant_rt16<= ('0' & s_mant_rt10(mant*2+2 DOWNTO mant+2)) + ('0' & s_mant_rt14(mant+1
DOWNTO 1));
            s_skip2 <= '0';
END IF;
END IF;
            s_cnt <= s_cnt + 1;
ELSIF (s_cnt = 40) THEN
IF (s_skip1 = '0') THEN
            s_lshift_min(mant*2+2 DOWNTO mant+1) <= s_mant_rt15;
            s_lshift_min(mant DOWNTO 0)          <= (OTHERS => '0');
            s_lshift_ein                         <= s_exp_rt15;
END IF;
IF (s_skip2 = '0') THEN
```

```
            s_lshift_min1(mant*2+2 DOWNTO mant+1) <= s_mant_rt16;
            s_lshift_min1(mant DOWNTO 0)          <= (OTHERS => '0');
            s_lshift_ein1                         <= s_exp_rt16;
END IF;
            s_cnt                             <= s_cnt + 1;
ELSIF (s_cnt = 41) THEN
            s_cnt <= s_cnt + 1;                                   -- WAITING FOR SHIFTING BITS
ELSIF (s_cnt = 42) THEN
IF (s_skip1 = '0') THEN
            s_exp_rt15 <= s_lshift_eot + 1;
            s_mant_rt15<= s_lshift_mot(mant*2+2 DOWNTO mant+1);
END IF;
IF (s_skip2 = '0') THEN
            s_exp_rt16 <= s_lshift_eot1 + 1;
            s_mant_rt16<= s_lshift_mot1(mant*2+2 DOWNTO mant+1);
END IF;
            s_sin_rt17 <= s_sin_rt6;
            s_exp_rt17 <= s_exp_rt6;
            s_mant_rt17 <= s_mant_rt6;
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 43) THEN
            complt <= '1';
            xb1_ot <= s_sin_rt15 & s_exp_rt15 & s_mant_rt15(mant DOWNTO 1);
            xb2_ot <= s_sin_rt16 & s_exp_rt16 & s_mant_rt16(mant DOWNTO 1);
            xb3_ot <= s_sin_rt17 & s_exp_rt17 & s_mant_rt17(mant DOWNTO 1);
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 44) THEN
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 45) THEN
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 46) THEN
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 47) THEN
            s_cnt     <= s_cnt + 1;
ELSIF (s_cnt = 48) THEN
            s_cnt     <= s_cnt + 1;
ELSE
            s_cnt <= "000000";
END IF;
ELSE                -- RESETS WHEN IT IS DISABLED
            xb1_ot <= (OTHERS => '0');
            xb2_ot <= (OTHERS => '0');
            xb3_ot <= (OTHERS => '0');
            xh1_ot <= (OTHERS => '0');
            xh2_ot <= (OTHERS => '0');
            xh3_ot <= (OTHERS => '0');
            s_sin_rt4  <= '0';                          --xh1
            s_exp_rt4  <= (OTHERS => '0');
            s_mant_rt4 <= (OTHERS => '0');
            s_sin_rt5  <= '0';                          --xh2
            s_exp_rt5  <= (OTHERS => '0');
            s_mant_rt5 <= (OTHERS => '0');
            s_sin_rt6  <= '0';                          --xh3
            s_exp_rt6  <= (OTHERS => '0');
            s_mant_rt6 <= (OTHERS => '0');
            s_sin_rt15  <= '0';                         --xb1
```

```
            s_exp_rt15  <= (OTHERS => '0');
            s_mant_rt15 <= (OTHERS => '0');
            s_sin_rt16  <= '0';                    --xb2
            s_exp_rt16  <= (OTHERS => '0');
            s_mant_rt16 <= (OTHERS => '0');
            s_sin_rt17  <= '0';                    --xb3
            s_exp_rt17  <= (OTHERS => '0');
            s_mant_rt17 <= (OTHERS => '0');
            s_cnt <= "000000";
END IF;
END IF;
END PROCESS;
END bh;
```

C. JAVA User Interface Code

```
public class Cact extends JPanel implements SerialPortEventListener {
        int bufLength = 16000;
        int graphA[] = new int[bufLength];
        int graphB[] = new int[bufLength];
        int graphC[] = new int[bufLength];
        int realAcount = 0;
        int realBcount = 0;
        int realCcount = 0;
        GraphPanel graphPanel;
        Color backgroundColor = Color.white;
        Color graph1Color = Color.red;
        Color graph2Color = Color.green;
        Color graph3Color = Color.blue;
        float rangeX=1, positionX=0;
        float rangeY=1, positionY=0;
        boolean graph1On = true;
        boolean graph2On = false;
        boolean graph3On = false;
        JSlider timeScaleSlider;
        JSlider voltScaleSlider;
        JSlider timePositionSlider;
        JSlider voltPositionSlider;
        JButton startButton;
        JButton stopButton;
        JComboBox graphOptionComboBox;
        String[] graphOptionList = {"xhat3", "xhat2", "xhat1", "reference", "error"};
        JLabel graph1Label = new JLabel("GyroOut");
        JLabel graph2Label = new JLabel("Control(u)");
        JLabel graph3Label = new JLabel("Reference");
        JLabel graph1LabelBar;
        JLabel graph2LabelBar;
        JLabel graph3LabelBar;
        JLabel timeScaleULabel;
        JLabel voltScaleULabel;
        JLabel timePositionULabel;
        JLabel voltPositionULabel;
        int timeRate = 1;
        int voltRate = 1;
        int positionTimeRate = 0;
        int positionVoltRate = 0;
        JButton[] parameterChangeButton = new JButton[10];
        JTextField[] parameterText = new JTextField[21];
        JRadioButton[] dispOnOffButtons;
        JTextArea textArea;
        String commTextDisp = "";
        boolean commTextDispOn = false;
        JComboBox serialPortList;
        JComboBox[] graphColorList;
        JComboBox backgroundColorList;
        String[] serialPortStrings = {"COM1", "COM2"};
```

```
        String[] colorStrings = {"Red", "Green", "Blue", "Yellow", "Megenta", "Cyan", "Black",
"White"};
        Color[] colorArray = {Color.red, Color.green, Color.blue, Color.yellow, Color.magenta,
Color.cyan, Color.black, Color.white};
        JTextField[] graphLabelText = new JTextField[3];
        int gACount = 0;
        int gBCount = 0;
        int gCCount = 0;
        CommPortIdentifier portId;
        InputStream inputStream;
        OutputStream outputStream;
        SerialPort serialPort;
        String portname;
        int para_kp;
        int para_kd;
        int para_phi3;
        int para_gamma1;
        int para_gamma2;
        int para_lc1;
        int para_lc2;
        int para_lc3;

public Cact() {
        super(new GridLayout(1, 1));
        JTabbedPane tabbedPane = new JTabbedPane();
        JComponent graphPanel = makeGraphPanel();
        graphPanel.setPreferredSize(new Dimension(700, 650));
        tabbedPane.addTab("Graph", null, graphPanel, "Monitoring Signals");
        tabbedPane.setMnemonicAt(0, KeyEvent.VK_1);
        JComponent parameterOptionPanel = makeParameterOptionPanel();
        tabbedPane.addTab("Parameters", null, parameterOptionPanel, "Paremeters Changeing");
        tabbedPane.setMnemonicAt(1, KeyEvent.VK_2);
        JComponent comDisplayPanel = makeComportPanel();
        tabbedPane.addTab("Com Display", null, comDisplayPanel, "Display Com Port Data");
        tabbedPane.setMnemonicAt(2, KeyEvent.VK_3);
        JComponent optionPanel = makeOptionPanel();
        tabbedPane.addTab("Options", null, optionPanel, "Changing Options");
        tabbedPane.setMnemonicAt(3, KeyEvent.VK_4);
        add(tabbedPane);
}
public class GraphPanel extends JPanel {
public GraphPanel() {
          super();
        }
public void paint(Graphics g) {
          int graphdX, graphdY;
          int graphX1, graphX2, graphY1, graphY2;
          int timelimite;
          Dimension d = getSize();
          timelimite = 0;
          graphX1 = 0;
          graphY1 = 0;
          graphX2 = d.width-1;
          graphY2 = d.height-1;
          graphdX = d.width/10;
          graphdY = d.height/10;
```

```
            g.setColor(backgroundColor);
            g.fillRect(graphX1, graphY1, graphX2, graphY2);
            g.setColor(Color.black);
            g.drawRect(graphX1, graphY1, graphX2, graphY2);
            g.setColor(Color.lightGray);
            for(int i = 1; i < 10; i++) {
            g.drawLine(graphX1 + (graphdX*i), graphY1 + 1, graphX1 + (graphdX * i), graphY1 +
graphY2 - 1);
            g.drawLine(graphX1 + 1, graphY1 + (graphdY*i), graphX1 + graphX2 - 1, graphY1 +
(graphdY*i));
            }
            positionX = positionTimeRate * (-d.width/10);
            positionY = (-1 * positionVoltRate) * (d.height/10) + d.height/2;
            g.setColor(graph1Color);
            if(graph1On) {
              for (int i = 0; i < realAcount-3; i++) {
                    g.drawLine((int)((20*i*graphdX)/rangeX + positionX), (int)((-
1)*graphA[i]*graphdY/rangeY + positionY),
                          (int)((20*(i+1)*graphdX)/rangeX + positionX) , (int)((-
1)*graphA[i+1]*graphdY/rangeY + positionY));
              }
            }
            g.setColor(graph2Color);
            if(graph2On) {
              for (int i = 0; i < realBcount-3; i++) {
                    g.drawLine((int)((20*i*graphdX)/rangeX + positionX), (int)((-
1)*(graphB[i])*graphdY/rangeY + positionY) ,
                          (int)((20*(i+1)*graphdX)/rangeX + positionX), (int)((-
1)*(graphB[i+1])*graphdY/rangeY + positionY));
              }
            }
            g.setColor(graph3Color);
            if(graph3On) {
              for (int i = 0; i < realCcount-3; i++) {
                    g.drawLine((int)((20*i*graphdX)/rangeX + positionX), (int)((-
1)*graphC[i]*graphdY/rangeY + positionY),
                          (int)((20*(i+1)*graphdX)/rangeX + positionX), (int)((-
1)*graphC[i+1]*graphdY/rangeY + positionY));
              }
            }
          }
  }
public JComponent makeGraphPanel() {
          JPanel returnPanel = new JPanel();
          returnPanel.setLayout(null);
          Insets insets = returnPanel.getInsets();
          graphPanel = new GraphPanel();
          graphPanel.setBounds(insets.left + 20, insets.top + 30, 500, 400);
          graph1Label.setHorizontalAlignment(JTextField.LEFT);
          graph1Label.setBounds(insets.left + 550, insets.top + 30, 80, 20);
          graph2Label.setHorizontalAlignment(JTextField.LEFT);
          graph2Label.setBounds(insets.left + 550, insets.top + 50, 80, 20);
          graph3Label.setHorizontalAlignment(JTextField.LEFT);
          graph3Label.setBounds(insets.left + 550, insets.top + 70, 80, 20);
          graph1LabelBar = new JLabel();
          graph1LabelBar.setOpaque(true);
```

```
graph1LabelBar.setBackground(graph1Color);
graph1LabelBar.setBounds(insets.left + 650, insets.top + 39, 40, 2);
graph2LabelBar = new JLabel();
graph2LabelBar.setOpaque(true);
graph2LabelBar.setBackground(graph2Color);
graph2LabelBar.setBounds(insets.left + 650, insets.top + 59, 40, 2);
graph3LabelBar = new JLabel();
graph3LabelBar.setOpaque(true);
graph3LabelBar.setBackground(graph3Color);
graph3LabelBar.setBounds(insets.left + 650, insets.top + 79, 40, 2);
JLabel divisionLabel = new JLabel("DIVISION", JLabel.CENTER);
JLabel timeScaleLabel = new JLabel("T   RANGE");
JLabel voltScaleLabel = new JLabel("V   RANGE");
JLabel timePositionLabel = new JLabel("T POSITION");
JLabel voltPositionLabel = new JLabel("V POSITION");
timeScaleSlider = new JSlider(JSlider.HORIZONTAL, 1, 500, timeRate);
voltScaleSlider = new JSlider(JSlider.HORIZONTAL, 1, 100, voltRate);
timePositionSlider = new JSlider(JSlider.HORIZONTAL, 0, 1000, positionTimeRate);
voltPositionSlider = new JSlider(JSlider.HORIZONTAL, -20, 20, positionVoltRate);
timeScaleULabel = new JLabel();
timeScaleULabel.setText("166 uSec");
voltScaleULabel = new JLabel();
voltScaleULabel.setText("50 mV");
timePositionULabel = new JLabel();
timePositionULabel.setText("0 uSec");
voltPositionULabel = new JLabel();
voltPositionULabel.setText("0 mV");
Hashtable timeScaleSliderTable = new Hashtable();
Hashtable voltScaleSliderTable = new Hashtable();
Hashtable timePositionSliderTable = new Hashtable();
Hashtable voltPositionSliderTable = new Hashtable();
divisionLabel.setBounds(insets.left + 10, insets.top + 490, 200, 20);
timeScaleLabel.setBounds(insets.left + 10, insets.top + 520, 100, 30);
voltScaleLabel.setBounds(insets.left + 10, insets.top + 580, 100, 30);
timePositionLabel.setBounds(insets.left + 300, insets.top + 520, 100, 30);
voltPositionLabel.setBounds(insets.left + 300, insets.top + 580, 100, 30);
timeScaleSliderTable.put(new Integer(1), new JLabel("-"));
timeScaleSliderTable.put(new Integer(500), new JLabel("+"));
voltScaleSliderTable.put(new Integer(1), new JLabel("-"));
voltScaleSliderTable.put(new Integer(100), new JLabel("+"));
timePositionSliderTable.put(new Integer(0), new JLabel("0"));
timePositionSliderTable.put(new Integer(1000), new JLabel("+"));
voltPositionSliderTable.put(new Integer(-20), new JLabel("-"));
voltPositionSliderTable.put(new Integer(0), new JLabel("0"));
voltPositionSliderTable.put(new Integer(20), new JLabel("+"));
timeScaleSlider.setBounds(insets.left + 100, insets.top + 520, 100, 30);
timeScaleSlider.addChangeListener(new sliderListener());
timeScaleSlider.setPaintLabels(true);
timeScaleSlider.setLabelTable(timeScaleSliderTable);
timeScaleSlider.setMaximumSize(new Dimension(100, 28));
voltScaleSlider.setBounds(insets.left + 100, insets.top + 580, 100, 30);
voltScaleSlider.addChangeListener(new sliderListener());
voltScaleSlider.setPaintLabels(true);
voltScaleSlider.setLabelTable(voltScaleSliderTable);
voltScaleSlider.setMaximumSize(new Dimension(100, 28));
timePositionSlider.setBounds(insets.left + 400, insets.top + 520, 100, 30);
```

```
        timePositionSlider.addChangeListener(new sliderListener());
        timePositionSlider.setPaintLabels(true);
        timePositionSlider.setLabelTable(timePositionSliderTable);
        timePositionSlider.setMaximumSize(new Dimension(100, 28));
        voltPositionSlider.setBounds(insets.left + 400, insets.top + 580, 100, 30);
        voltPositionSlider.addChangeListener(new sliderListener());
        voltPositionSlider.setPaintLabels(true);
        voltPositionSlider.setLabelTable(voltPositionSliderTable);
        voltPositionSlider.setMaximumSize(new Dimension(100, 28));
        timeScaleULabel.setBounds(insets.left + 200, insets.top + 520, 80, 30);
        voltScaleULabel.setBounds(insets.left + 200, insets.top + 580, 80, 30);
        timePositionULabel.setBounds(insets.left + 500, insets.top + 520, 80, 30);
        voltPositionULabel.setBounds(insets.left + 500, insets.top + 580, 80, 30);
        JLabel graphOptionLabel = new JLabel("Select for G3", JLabel.CENTER);
        graphOptionComboBox = new JComboBox(graphOptionList);
        JLabel controlLabel = new JLabel("CONTROL", JLabel.CENTER);
        startButton = new JButton("START");
        stopButton = new JButton("STOP");
        graphOptionLabel.setBounds(insets.left + 580, insets.top + 400, 100, 20);
        graphOptionComboBox.setBounds(insets.left + 580, insets.top + 420, 100, 30);
        graphOptionComboBox.addActionListener(new cbActionHandler());
        controlLabel.setBounds(insets.left + 580, insets.top + 490, 100, 30);
        startButton.setBounds(insets.left + 580, insets.top + 520, 100, 60);
        startButton.addActionListener(new buttonHandler());
        stopButton.setBounds(insets.left + 580, insets.top + 580, 100, 60);
        stopButton.addActionListener(new buttonHandler());
        returnPanel.add(graphPanel);
        returnPanel.add(graph1Label);
        returnPanel.add(graph2Label);
        returnPanel.add(graph3Label);
        returnPanel.add(graph1LabelBar);
        returnPanel.add(graph2LabelBar);
        returnPanel.add(graph3LabelBar);
        returnPanel.add(timeScaleSlider);
        returnPanel.add(voltScaleSlider);
        returnPanel.add(timePositionSlider);
        returnPanel.add(voltPositionSlider);
        returnPanel.add(divisionLabel);
        returnPanel.add(timeScaleLabel);
        returnPanel.add(voltScaleLabel);
        returnPanel.add(timePositionLabel);
        returnPanel.add(voltPositionLabel);
        returnPanel.add(graphOptionLabel);
        returnPanel.add(graphOptionComboBox);
        returnPanel.add(controlLabel);
        returnPanel.add(startButton);
        returnPanel.add(stopButton);
        returnPanel.add(timeScaleULabel);
        returnPanel.add(voltScaleULabel);
        returnPanel.add(timePositionULabel);
        returnPanel.add(voltPositionULabel);
        return returnPanel;
 }
public JComponent makeParameterOptionPanel() {
        JPanel returnPanel = new JPanel();
        returnPanel.setLayout(null);
```

```
Insets insets = returnPanel.getInsets();
JLabel[] pLabel = new JLabel[10];
pLabel[0] = new JLabel("wc");
pLabel[1] = new JLabel("wo");
pLabel[2] = new JLabel("bo");
pLabel[3] = new JLabel("Ts : 0.000001");
pLabel[4] = new JLabel("Phi");
pLabel[5] = new JLabel("Gamma");
pLabel[6] = new JLabel("Lc");
pLabel[7] = new JLabel("Kp");
pLabel[8] = new JLabel("Kd");
pLabel[9] = new JLabel("U limit");
for (int i = 0; i < 21; i++) {
	parameterText[i] = new JTextField(10);
}
parameterChangeButton[0] = new JButton("ENTER");
for (int i = 1; i < 10; i++) {
	parameterChangeButton[i] = new JButton("SEND");
}
pLabel[0].setBounds(insets.left + 50, insets.top + 50, 30, 30);		//WC
pLabel[1].setBounds(insets.left + 200, insets.top + 50, 30, 30);//WO
pLabel[2].setBounds(insets.left + 350, insets.top + 50, 30, 30);//BO
pLabel[3].setBounds(insets.left + 80, insets.top + 120, 200, 30);		//TS
pLabel[4].setBounds(insets.left + 50, insets.top + 150, 100, 30);		//PHI
pLabel[5].setBounds(insets.left + 50, insets.top + 300, 100, 30);		//GAMMA
pLabel[6].setBounds(insets.left + 350, insets.top + 300, 100, 30);		//LC
pLabel[7].setBounds(insets.left + 50, insets.top + 450, 30, 30);//Kp
pLabel[8].setBounds(insets.left + 350, insets.top + 450, 30, 30);		//Kd
pLabel[9].setBounds(insets.left + 30, insets.top + 500, 50, 30);//U LIMIT
parameterText[0].setBounds(insets.left + 80, insets.top + 50, 100, 30);
parameterText[0].setText("2500000");
parameterText[1].setBounds(insets.left + 230, insets.top + 50, 100, 30);
parameterText[1].setText("2500000");
parameterText[2].setBounds(insets.left + 380, insets.top + 50, 100, 30);
parameterText[2].setText("271780000");
parameterText[3].setBounds(insets.left + 80, insets.top + 180, 130, 30);//phi
parameterText[3].setText("1");
parameterText[4].setBounds(insets.left + 80, insets.top + 210, 130, 30);
parameterText[4].setText("0");
parameterText[5].setBounds(insets.left + 80, insets.top + 240, 130, 30);
parameterText[5].setText("0");
parameterText[6].setBounds(insets.left + 210, insets.top + 180, 130, 30);
parameterText[6].setText("Ts");
parameterText[7].setBounds(insets.left + 210, insets.top + 210, 130, 30);
parameterText[7].setText("1");
parameterText[8].setBounds(insets.left + 210, insets.top + 240, 130, 30);
parameterText[8].setText("0");
parameterText[9].setBounds(insets.left + 340, insets.top + 180, 130, 30);
parameterText[9].setText("5.0E-13");
parameterText[10].setBounds(insets.left + 340, insets.top + 210, 130, 30);
parameterText[10].setText("Ts");
parameterText[11].setBounds(insets.left + 340, insets.top + 240, 130, 30);
parameterText[11].setText("1");
parameterText[12].setBounds(insets.left + 80, insets.top + 320, 130, 30);		//gamma
parameterText[12].setText("1.3589E-4");
parameterText[13].setBounds(insets.left + 80, insets.top + 350, 130, 30);
```

```
            parameterText[13].setText("271.78");
            parameterText[14].setBounds(insets.left + 80, insets.top + 380, 130, 30);
            parameterText[14].setText("0");
            parameterText[15].setBounds(insets.left + 380, insets.top + 320, 130, 30);          //lc
            parameterText[15].setText("0.9994469");
            parameterText[16].setBounds(insets.left + 380, insets.top + 350, 130, 30);
            parameterText[16].setText("1367595.2");
            parameterText[17].setBounds(insets.left + 380, insets.top + 380, 130, 30);
            parameterText[17].setText("7.7340574E11");
            parameterText[18].setBounds(insets.left + 80, insets.top + 450, 130, 30);           //kp
            parameterText[18].setText("6.25E12");
            parameterText[19].setBounds(insets.left + 380, insets.top + 450, 130, 30);          //kd
            parameterText[19].setText("5000000.0");
            parameterText[20].setBounds(insets.left + 80, insets.top + 500, 100, 30);           //ul
            parameterText[20].setText("2000");
            parameterText[0].setHorizontalAlignment(JTextField.RIGHT);
            parameterText[1].setHorizontalAlignment(JTextField.RIGHT);
            parameterText[2].setHorizontalAlignment(JTextField.RIGHT);
            parameterText[20].setHorizontalAlignment(JTextField.RIGHT);
            for (int i = 3; i < 20; i++) {
                        parameterText[i].setEditable(false);
            }
            parameterChangeButton[0].setBounds(insets.left + 550, insets.top + 50, 100, 30);
            parameterChangeButton[0].addActionListener(new buttonHandler());   //enter
            parameterChangeButton[1].setBounds(insets.left + 550, insets.top + 180, 100, 30);
            parameterChangeButton[1].addActionListener(new buttonHandler());   //phi
            parameterChangeButton[2].setBounds(insets.left + 230, insets.top + 320, 100, 30);
            parameterChangeButton[2].addActionListener(new buttonHandler());   //gamma1
            parameterChangeButton[3].setBounds(insets.left + 230, insets.top + 350, 100, 30);
            parameterChangeButton[3].addActionListener(new buttonHandler());   //gamma2
            parameterChangeButton[4].setBounds(insets.left + 550, insets.top + 320, 100, 30);
            parameterChangeButton[4].addActionListener(new buttonHandler());   //lc1
            parameterChangeButton[5].setBounds(insets.left + 550, insets.top + 350, 100, 30);
            parameterChangeButton[5].addActionListener(new buttonHandler());   //lc2
            parameterChangeButton[6].setBounds(insets.left + 550, insets.top + 380, 100, 30);
            parameterChangeButton[6].addActionListener(new buttonHandler());   //lc3
            parameterChangeButton[7].setBounds(insets.left + 230, insets.top + 450, 100, 30);
            parameterChangeButton[7].addActionListener(new buttonHandler());   //kp
            parameterChangeButton[8].setBounds(insets.left + 550, insets.top + 450, 100, 30);
            parameterChangeButton[8].addActionListener(new buttonHandler());   //kd
            parameterChangeButton[9].setBounds(insets.left + 230, insets.top + 500, 100, 30);
            parameterChangeButton[9].addActionListener(new buttonHandler());   //ul
            for (int i = 0; i < 10; i++) {
                        returnPanel.add(pLabel[i]);
                        returnPanel.add(parameterChangeButton[i]);
            }
            for (int i = 0; i < 21; i++) {
                        returnPanel.add(parameterText[i]);
            }
            return returnPanel;
      }
public JComponent makeComportPanel() {
            JPanel returnPanel = new JPanel();
            returnPanel.setLayout(null);
            Insets insets = returnPanel.getInsets();
            JLabel topLabel = new JLabel("MESSAGE DISPLAY");
```

```
            ButtonGroup dispOnOffGroup = new ButtonGroup();
            dispOnOffButtons = new JRadioButton[2];
            dispOnOffButtons[0] = new JRadioButton(" ON ");
            dispOnOffButtons[0].addActionListener(new buttonHandler());
            dispOnOffButtons[1] = new JRadioButton(" OFF");
            dispOnOffButtons[1].addActionListener(new buttonHandler());
            dispOnOffGroup.add(dispOnOffButtons[0]);
            dispOnOffGroup.add(dispOnOffButtons[1]);
            dispOnOffButtons[1].setSelected(true);
            topLabel.setBounds(insets.left + 50, insets.top + 20, 200, 30);
            dispOnOffButtons[0].setBounds(insets.left + 250, insets.top + 20, 200, 30);
            dispOnOffButtons[1].setBounds(insets.left + 450, insets.top + 20, 200, 30);
            textArea = new JTextArea(commTextDisp);
            textArea.setEditable(false);
            textArea.setFont(new Font("Serif", Font.ITALIC, 16));
            textArea.setLineWrap(true);
            textArea.setWrapStyleWord(true);
            JScrollPane scroller =new JScrollPane(textArea);
            scroller.setBounds(insets.left + 10, insets.top + 70, 680, 570);
            returnPanel.add(topLabel);
            returnPanel.add(dispOnOffButtons[0]);
            returnPanel.add(dispOnOffButtons[1]);
            returnPanel.add(scroller);
            return returnPanel;
    }
public JComponent makeOptionPanel() {
            JPanel returnPanel = new JPanel();
            returnPanel.setLayout(null);
            Insets insets = returnPanel.getInsets();
            JLabel[] leftLabel = new JLabel[4];
            JLabel[] g1Label = new JLabel[3];
            JLabel[] g2Label = new JLabel[3];
            leftLabel[0] = new JLabel("SERIAL PORT: ");
            leftLabel[1] = new JLabel("GRAPH COLOR: ");
            leftLabel[2] = new JLabel("GRAPH LABEL: ");
            leftLabel[3] = new JLabel("BACKGROUND COLOR: ");
            for (int i = 0; i < 4; i++) {
                        leftLabel[i].setBounds(insets.left + 50, insets.top + 50 + (i*100), 200, 30);
            }
            for (int i = 0; i < 3; i++) {
                        g1Label[i] = new JLabel("GRAPH " + (i+1));
                        g2Label[i] = new JLabel("GRAPH " + (i+1));
                        g1Label[i].setBounds(insets.left + 250, insets.top + 150 + (i*30), 100, 30);
                        g2Label[i].setBounds(insets.left + 250, insets.top + 250 + (i*30), 100, 30);
            }
            serialPortList = new JComboBox(serialPortStrings);
            graphColorList = new JComboBox[3];
            backgroundColorList = new JComboBox(colorStrings);
            serialPortList.setSelectedIndex(0);
            serialPortList.addActionListener(new cbActionHandler());
            serialPortList.setBounds(insets.left + 350, insets.top + 50, 200, 30);
            for (int i = 0; i < 3; i++) {
                        graphColorList[i] = new JComboBox(colorStrings);
                        graphColorList[i].setSelectedIndex(i);
                        graphColorList[i].addActionListener(new cbActionHandler());
                        graphColorList[i].setBounds(insets.left + 350, insets.top + 150 + (i*30), 200, 30);
```

```
                graphLabelText[i] = new JTextField(5);
                graphLabelText[i].setBounds(insets.left + 350, insets.top + 250 + (i*30), 200, 30);
        }
        graphLabelText[0].setText("GyroOut");
        graphLabelText[1].setText("Control(u)");
        graphLabelText[2].setText("Reference");
        backgroundColorList.setSelectedIndex(7);
        backgroundColorList.addActionListener(new cbActionHandler());
        backgroundColorList.setBounds(insets.left + 350, insets.top + 350, 200, 30);
        for (int i = 0; i < 4; i++) {
                returnPanel.add(leftLabel[i]);
        }
        for (int i = 0; i < 3; i++) {
                returnPanel.add(g1Label[i]);
                returnPanel.add(g2Label[i]);
                returnPanel.add(graphColorList[i]);
                returnPanel.add(graphLabelText[i]);
        }
        returnPanel.add(serialPortList);
        returnPanel.add(backgroundColorList);
        return returnPanel;
  }
public class buttonHandler implements ActionListener {
public void actionPerformed(ActionEvent event) {
        if(event.getSource() == startButton) {
                commTextDisp = "";
                textArea.replaceRange(null, 0, textArea.getDocument().getLength());
                portOpen();
                gACount = 0;
                gBCount = 0;
                gCCount = 0;
        for (int i = 0; i < bufLength; i++) {
                graphA[i] = 0;
                graphB[i] = 0;
                graphC[i] = 0;
                }
        sendMessage("CT");
        System.out.println("start");
                realAcount = 0;
                realBcount = 0;
                realCcount = 0;
        }else if(event.getSource() == stopButton) {
                portOpen();
                sendMessage("CO");                      // send command close
                System.out.println("stop");
                repaint();
                portClose();
        }else if(event.getSource() == parameterChangeButton[0]) {
                portOpen();
                int wc, wo, bo;
                int bo_send;
                double kp, kd;
                double ov_bo;
                double phi3, gamma1, gamma2;
                double lc1, lc2, lc3, Ts, beta;
                String kpStr, kdStr, boStr;
```

```
            String phi3Str, gamma1Str, gamma2Str;
            String lc1Str, lc2Str, lc3Str;
            Ts = 0.000001;
            wc =  java.lang.Integer.parseInt(parameterText[0].getText());
            wo =  java.lang.Integer.parseInt(parameterText[1].getText());
            bo =  java.lang.Integer.parseInt(parameterText[2].getText());
            ov_bo = 1 / (double)bo;
            bo_send = convertNumber((float)ov_bo);
            beta = Math.exp((double)(-wc*Ts));
            phi3 = Ts * Ts / 2;
            gamma1 = phi3 * bo;
            gamma2 = Ts * bo;
            lc1 = 1 - Math.pow(beta, (double)3);
            lc2 = 1.5 * (beta-1)*(beta-1)*(beta+1) / Ts;
            lc3 =  -1 * Math.pow((beta-1), (double)3) / (Ts*Ts);
            kp = ((double)wc) * ((double) wc);
            kd = (double)(2 * wc);
            boStr = "" + bo_send;
            phi3Str = "" + (float)phi3;
            gamma1Str = "" + (float)gamma1;
            gamma2Str = "" + (float)gamma2;
            lc1Str = "" + (float)lc1;
            lc2Str = "" + (float)lc2;
            lc3Str = "" + (float)lc3;
            kpStr = "" + (float)kp;
            kdStr = "" + (float)kd;
            para_kp = convertNumber((double)kp);
            para_kd = convertNumber((double)kd);
            para_phi3 = convertNumber((double)phi3);
            para_gamma1 = convertNumber((double)gamma1);
            para_gamma2 = convertNumber((double)gamma2);
            para_lc1 = convertNumber((double)lc1);
            para_lc2 = convertNumber((double)lc2);
            para_lc3 = convertNumber((double)lc3);
            parameterText[9].setText(phi3Str);
            parameterText[12].setText(gamma1Str);
            parameterText[13].setText(gamma2Str);
            parameterText[15].setText(lc1Str);
            parameterText[16].setText(lc2Str);
            parameterText[17].setText(lc3Str);
            parameterText[18].setText(kpStr);
            parameterText[19].setText(kdStr);
            sendMessage("CD" + boStr + ".");
            System.out.println(boStr);
    }else if(event.getSource() == parameterChangeButton[1]) {
            portOpen();
            String toSend;
            toSend = "" + para_phi3;
            sendMessage("CE" + toSend + ".");
            System.out.println(toSend);
    }else if(event.getSource() == parameterChangeButton[2]) {
            portOpen();
            String toSend;
            toSend = "" + para_gamma1;
            sendMessage("CF" + toSend + ".");
            System.out.println(toSend);
```

```
		}else if(event.getSource() == parameterChangeButton[3]) {
			portOpen();
			String toSend;
			toSend = "" + para_gamma2;
			sendMessage("CG" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == parameterChangeButton[4]) {
			portOpen();
			String toSend;
			toSend = "" + para_lc1;
			sendMessage("CH" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == parameterChangeButton[5]) {
			portOpen();
			String toSend;
			toSend = "" + para_lc2;
			sendMessage("CI" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == parameterChangeButton[6]) {
			portOpen();
			String toSend;
			toSend = "" + para_lc3;
			sendMessage("CJ" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == parameterChangeButton[7]) {
			portOpen();
			String toSend;
			toSend = "" + para_kp;
			sendMessage("CK" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == parameterChangeButton[8]) {
			portOpen();
			String toSend;
			toSend = "" + para_kd;
			sendMessage("CL" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == parameterChangeButton[9]) {
			portOpen();
			int ul, ul_send;
			String toSend;
			ul =  java.lang.Integer.parseInt(parameterText[20].getText());
			ul_send = convertNumber((float)ul);
			toSend = "" + ul_send;
			sendMessage("CM" + toSend + ".");
			System.out.println(toSend);
		}else if(event.getSource() == dispOnOffButtons[0]) {
			portOpen();
			commTextDispOn = true;
			System.out.println("Serial Port On");
		}else if(event.getSource() == dispOnOffButtons[1]) {
			portOpen();
			commTextDispOn = false;
			System.out.println("Serial Port off");
		}
		}
}
```

```
public class cbActionHandler implements ActionListener {
public void actionPerformed(ActionEvent event) {
if(event.getSource() == graphOptionComboBox) {
        String selectedGraphString;
        int selectedGraphNum = 0;
        selectedGraphString = graphOptionList[graphOptionComboBox.getSelectedIndex()];
        if(selectedGraphString == graphOptionList[0]) {selectedGraphNum = 0;}
        else if(selectedGraphString == graphOptionList[1]) {selectedGraphNum = 1;}
        else if(selectedGraphString == graphOptionList[2]) {selectedGraphNum = 2;}
        else if(selectedGraphString == graphOptionList[3]) {selectedGraphNum = 3;}
        else if(selectedGraphString == graphOptionList[4]) {selectedGraphNum = 4;}
        if(selectedGraphNum != 4) {
                portOpen();
                sendMessage("CS" + selectedGraphNum + ".");
                System.out.println("Graph option" + selectedGraphString);
                graph1On = true;
                graph2On = true;
                graph3On = true;
                repaint();
        }else {
                graph1On = false;
                graph2On = true;
                graph3On = false;
        for (int i = 0; i < realAcount-2; i++) {
                graphB[i] = graphC[i] - graphA[i];}
                repaint();
        }
        }else if(event.getSource() == serialPortList) {
        System.out.println("Serial Port" + serialPortStrings[serialPortList.getSelectedIndex()]);
        }else if(event.getSource() == graphColorList[0]){
        graph1Color = colorArray[graphColorList[0].getSelectedIndex()];
        graph1LabelBar.setBackground(graph1Color);
        graph1Label.setText(graphLabelText[0].getText());
        if (graph1Color == Color.white) graph1On = false;
        else graph1On = true;
        System.out.println("Graph 1: " + colorStrings[graphColorList[0].getSelectedIndex()]);
        }else if(event.getSource() == graphColorList[1]){
        graph2Color = colorArray[graphColorList[1].getSelectedIndex()];
        graph2LabelBar.setBackground(graph2Color);
        graph2Label.setText(graphLabelText[1].getText());
        if (graph2Color == Color.white) graph2On = false;
        else graph2On = true;
        System.out.println("Graph 2: " + colorStrings[graphColorList[1].getSelectedIndex()]);
        }else if(event.getSource() == graphColorList[2]){
        graph3Color = colorArray[graphColorList[2].getSelectedIndex()];
        graph3LabelBar.setBackground(graph3Color);
        graph3Label.setText(graphLabelText[2].getText());
        if (graph3Color == Color.white) graph3On = false;
        else graph3On = true;
        System.out.println("Graph 3: " + colorStrings[graphColorList[2].getSelectedIndex()]);
        }else if(event.getSource() == backgroundColorList){
        backgroundColor = colorArray[backgroundColorList.getSelectedIndex()];
        System.out.println("Back Ground Color: " + colorStrings[backgroundColorList.get
SelectedIndex()]);
        }
```

```
            }
}
public class sliderListener implements ChangeListener {
public void stateChanged(ChangeEvent event) {
            JSlider source = (JSlider)event.getSource();
            if(source == timeScaleSlider) {
                        timeRate = source.getValue();
                        rangeX = timeRate * 100;
                        timeScaleULabel.setText(Integer.toString((int)(rangeX/6)) + "uSec");
                        repaint();
            }else if(source == voltScaleSlider) {
                        voltRate = source.getValue();
                        rangeY = voltRate * 100;
                        voltScaleULabel.setText(Integer.toString((int)(rangeY)) + "mV");
                        repaint();
            }else if(source == timePositionSlider) {
                        positionTimeRate = source.getValue();
            timePositionULabel.setText(Integer.toString((int)(positionTimeRate*(rangeX/6))) + "uSec");
                        repaint();
            }else if(source == voltPositionSlider) {
                        positionVoltRate = source.getValue();
                        voltPositionULabel.setText(Integer.toString((int)(positionVoltRate*rangeY)) + "mV");
                        repaint();
            }
}
}
public void portOpen() {
     try {
            portname = serialPortStrings[serialPortList.getSelectedIndex()];
            portId = CommPortIdentifier.getPortIdentifier(portname);
            serialPort = (SerialPort) portId.open("SimpleReadApp", 2000);
            inputStream = serialPort.getInputStream();
            outputStream = serialPort.getOutputStream();
            serialPort.addEventListener(this);
            serialPort.notifyOnDataAvailable(true);
            serialPort.notifyOnCTS(true);
            serialPort.notifyOnDSR(true);
            serialPort.setSerialPortParams(115200, 8, 1, 0);
            serialPort.setFlowControlMode(0);
} catch (Exception e) {
     }
}

 public void portClose() {
     try {
            serialPort.notifyOnDataAvailable(false);
            serialPort.setRTS(!serialPort.isRTS());
            serialPort.setDTR(!serialPort.isDTR());
            serialPort.removeEventListener();
            serialPort.close();
     } catch (Exception error) {
     }
   }
public void sendMessage(String message2send) {
            try {
                        outputStream.write(message2send.getBytes());
```

```
            } catch(Exception e) {
            }
    }
public void serialEvent(SerialPortEvent event) {
            int numSign = 1;
            int dataReceived = 0;
            int intData = 0;
            String inString = "";
            boolean signalData = false;
            boolean signalA = false;
            boolean signalB = false;
            boolean signalC = false;
            boolean commandData = false;
            boolean commStart = true;
            boolean textData = false;
            while((dataReceived != -1) && (event.getEventType() ==
SerialPortEvent.DATA_AVAILABLE)) {
            try {
                        dataReceived = inputStream.read();
                        if(dataReceived == -1) {break;}
                        else if (commStart && (char)dataReceived == 'G') {              // Signal number inable to
get
                        signalData = true;
                        commandData = false;
                        textData = false;
                        commStart = false;
            }else if (commStart && (char)dataReceived == 'C') {             // command charactor inable to get
                        signalData = false;
                        commandData = true;
                        textData = false;
                        commStart = false;
            }else if (commStart && (char)dataReceived == '[') {             // command charactor inable to get
                        signalData = false;
                        commandData = false;
                        textData = true;
                        commStart = false;
            } else if (signalData && (char)dataReceived == 'A') {
                        signalA = true;
                        signalB = false;
                        signalC = false;
            } else if  (signalData && (char)dataReceived == 'B') {
                        signalA = false;
                        signalB = true;
                        signalC = false;
            } else if  (signalData && (char)dataReceived == 'C') {
                        signalA = false;
                        signalB = false;
                        signalC = true;
            } else if (signalData && (dataReceived == 45)) {
                        numSign = -1;
            } else if (signalData && (dataReceived >= 48) && (dataReceived <= 57)) {
                        intData = num2Int(intData, dataReceived-48);
            } else if (signalA && (char)dataReceived == '.') {
                        graphA[gACount] = numSign * intData;
                        numSign = 1;
                        realAcount = gACount;
```

```
                    if (gACount >= bufLength-10) gACount = 0;
                    else gACount++;
            if (commTextDispOn)
                    commTextDisp += "SignalA" + Integer.toString((int)intData) + "\n";
                    intData = 0;
                    commStart = true;
                    signalA = false;
                    signalData = false;
            } else if (signalB && (char)dataReceived == '.') {
                    graphB[gBCount] = numSign * intData;
                    numSign = 1;
                    realBcount = gBCount;
                    if (gBCount >= bufLength-10) gBCount = 0;
                    else gBCount++;
            if(commTextDispOn)
                    commTextDisp += "SignalB" +Integer.toString((int)intData) + "\n";
                    intData = 0;
                    commStart = true;
                    signalB = false;
                    signalData = false;
            } else if (signalC && (char)dataReceived == '.') {
                    graphC[gCCount] = numSign * intData;
                    numSign = 1;
                    realCcount = gCCount;
                    if (gCCount >= bufLength-10) gCCount = 0;
                    else gCCount++;
                    if(commTextDispOn)
                    commTextDisp += "SignalC" +Integer.toString((int)intData) + "\n";
                    intData = 0;
                    commStart = true;
                    signalC = false;
                    signalData = false;
                    repaint();
            } else if (commandData && (char)dataReceived != '.') {
                    inString += (char)dataReceived;
                    commStart = false;
            } else if (commandData && (char)dataReceived == '.') {
            if(commTextDispOn)
                    commTextDisp += inString;
                    inString = "";
                    commandData = false;
                    commStart = true;
            } else if ((textData) && ((char)dataReceived != '.')) {              // if ((!commandData) &&
(!signalData) && (!commStart)) {
            if(commTextDispOn)
                    commTextDisp += (char)dataReceived;
            } else if ((textData) && ((char)dataReceived == '.')) {
                    textData = false;
                    commStart = true;
            }
            if (commTextDispOn){
                    textArea.append(commTextDisp);
                    textArea.setCaretPosition(textArea.getDocument().getLength());
                    commTextDisp = "";
            }
            } catch (Exception error) {
```

```
            }
            }
    }
public int num2Int(int num1, int num2) {
            num1 = num1*10 + num2;
            return num1;
    }
public int convertNumber(double num) {
            int i = 0, exp = 0, mant = 0;
            if (num >= 2) {
            while ((num >= 2) && (i < 127)) {
                        num = num/2;
                        i++;
            }
            } else if ((num < 1) && (num > 0)){
            while ((num < 1) && (i > -127)){
                        num = num*2;
                        i--;
                        }
            } else if (num == 0) {
                        i = -127;
                        num = (double)1.0;
            }
            if (i >= 127) {
                        exp = (int)(255 * Math.pow(2,23));
                        mant = 0;
            } else if (i <= -127) {
                        exp = 0;
                        mant = 0;
            } else {
                        exp = (int)((i + 127) * Math.pow(2,23));
                        mant = (int)((num - 1) * Math.pow(2,23));
            }
            return exp + mant;
}
private static void createAndShowGUI() {
            JFrame.setDefaultLookAndFeelDecorated(true);
            JFrame frame = new JFrame("Cact");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JComponent newContentPane = new Cact();
            newContentPane.setOpaque(true);
            frame.getContentPane().add(new Cact(), BorderLayout.CENTER);
            frame.pack();
            frame.setVisible(true);
    }
public static void main(String[] args) {
            javax.swing.SwingUtilities.invokeLater(new Runnable() {
                        public void run() { createAndShowGUI(); }
            });
    }
}
```