2012

Independent Component Analysis Algorithm FPGA Design to Perform Real-Time Blind Source Separation

Crispin Odom

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ENGINEERING

INDEPENDENT COMPONENT ANALYSIS ALGORITHM FPGA DESIGN TO PERFORM REAL-TIME BLIND SOURCE SEPARATION

By

CRISPIN ODOM

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

Degree Awarded:
Spring Semester, 2012

Crispin Odom defended this thesis on April 2, 2012.

The members of the supervisory committee were:

Uwe Meyer-Baese
Professor Directing Thesis

Simon Foo
Committee Member

Rodney Roberts
Committee Member

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

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dr. Meyer-Baese, and the members of my thesis committee, Dr. Simon Foo and Dr. Rodney Roberts for their encouragement and support throughout this endeavor. This document is dedicated to my mother, Rosaline Odom. Thank you for believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The conditions that arise in the Cocktail Party Problem prevail across many fields creating a need for of Blind Source Separation. The need for BSS has become prevalent in several fields of work. These fields include array processing, communications, medical signal processing, and speech processing, wireless communication, audio, acoustics and biomedical engineering. The concept of the cocktail party problem and BSS led to the development of Independent Component Analysis (ICA) algorithms. ICA proves useful for applications needing real time signal processing. The goal of this thesis was to perform an extensive study on ability and efficiency of Independent Component Analysis algorithms to perform blind source separation on mixed signals in software and implementation in hardware with a Field Programmable Gate Array (FPGA). The Algebraic ICA (A-ICA), Fast ICA, and Equivariant Adaptive Separation via Independence (EASI) ICA were examined and compared. The best algorithm required the least complexity and fewest resources while effectively separating mixed sources. The best algorithm was the EASI algorithm. The EASI ICA was implemented on hardware with Field Programmable Gate Arrays (FPGA) to perform and analyze its performance in real time.

CHAPTER ONE

INTRODUCTION

1.1 Brief Overview

The Cocktail Party Problem is a classic Digital Signal Processing problem focused on trying to separate voices or music mixed simultaneously based only on their mixtures. [1] The problem arises when using a microphone or antenna a signal is received by a sensor signal. This signal is a mixture of elements that are called sources. The signal received is a superimposition of signals emitted by the source which are in its area of reception. [2] In the cocktail party scenario, two people are talking at the same time on two microphones in different locations. The recorded signal would then consist of a mixture of two speech signals. [3] The microphones provide two recorded time signals, which can be denoted by $x_1(t)$ and $x_2(t)$. The amplitudes are represented x_1 and x_2 and the variable t, represents the time index. [4] The recorded signal is a weighted sum of the speech signals emitted by the two speakers. The signals are denoted by $s_1(t)$ and $s_2(t)$ which can be expressed as a linear equation:

$$X_1(t) = a_{11}s_1 + a_{12}s_2$$

$$X_2(t) = a_{21}s_1 + a_{22}s_2 \quad [4]$$

The parameters, $a_{11}, a_{12}, a_{21},\ and\ a_{22}$, depend on the distances of the microphones from the speakers. The two original speech signals $s_1(t)$ and $s_2(t)$ can be estimated using only the recorded signals $x_1(t)$ and $x_2(t)$. The sources and mixtures are unknown. A method for solving the cocktail party problem is to use some information on the statistical properties of the signals $x_i(t)$ to estimate the a_{ij}, and assume $s_1(t)$ and $s_2(t)$ are statistically independent. [4]

The aforementioned solution is Blind Source Separation (BSS). As can be seen in Figure 1.1, BSS is able to estimate the coefficients that characterize this linear combination, and estimate the original signals. [1]

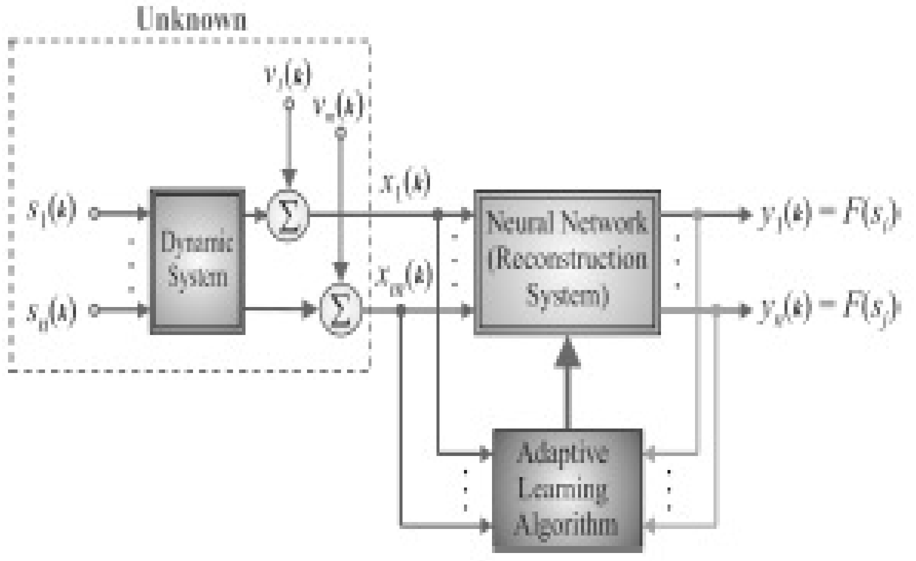

Figure 1.1: A general model of blind source separation. [5]

BSS is performed by analyzing mixtures of independent sources and using their components only to recover the original signals. [6] There is an existence of *n* statistically independent signals, s(t) = [s_1(t), ..., s_n(t)] and observed n-mixes that are linear combinations to create x(t) = [x_1(t), ..., x_n(t)]. The mixtures are generated from the following model: [1]
The simplified version of this equation is:

$$x(t) = \boldsymbol{A}s(t). \text{ [6].}$$

A is a square n x n mixing matrix with mixing components a_{ij}. Where x(t) is the observed vectors of mixture, which usually ignores noise. [6] Sources are separated by using the matrix **W**. **W** is used in the following equation:

$$\mathbf{WA} = \mathbf{PD} \text{ [6]}$$

Where **P** is a permutation matrix and **D** is a Diagonal matrix. The resulting recovered sources are then made available in y(t) in the following calculation:

$$y(t) = \mathbf{W}*x(t) = \mathbf{P}*\mathbf{D}*s(t) \text{ [6].}$$

The recovered sources are permutated and scaled versions of the original signals. The result is ultimately the separation of the two original source signals $s_1(t)$ and $s_2(t)$ from their mixtures $x_1(t)$ and $x_2(t)$. [4] [7] [8] [5]

1.2 Motivation

The conditions that arise in the Cocktail Party Problem prevail across many fields creating a need for of Blind Source Separation. [2] The need for BSS has become prevalent in several fields of work. These fields include array processing, communications, medical signal processing, and speech processing, wireless communication, audio, acoustics and biomedical engineering. [9] All of the mentioned fields use several source signals without exact knowledge of their transmission channel, or extraction, which makes it difficult to analyze signals when necessary. Solving this problem allows many applications such as mobile multiuser telecommunication systems to eliminate redundancy and sparse coding in noise cancellation. [7] Additionally, BSS helped to create voice reinforcement in noisy environments, such as urban ecology where noise pollution caused by high sound levels [1] . The concept of the cocktail party problem and BSS led to the development of Independent Component Analysis (ICA) algorithms. [4] ICA proves useful for applications needing real time signal processing such a speech signal enhancement noise canceling and ECG signal analysis.[10] ICA is being used in various signal processing applications such as audio signal processing, watermarking, and financial signal analysis. [11]

1.3 Problem Scope

The goal of this thesis was to perform an extensive study on ability and efficiency of Independent Component Analysis algorithms to perform blind source separation on mixed signals in software and implementation in hardware with a Field Programmable Gate Array (FPGA). The Algebraic ICA (A-ICA), Fast ICA, and Equivariant Adaptive Separation via Independence (EASI) ICA were examined and compared. The best algorithm required the least complexity and fewest resources while effectively separating mixed sources. The best algorithm was the EASI algorithm. The EASI ICA was implemented on hardware with Field Programmable Gate Arrays (FPGA) to perform and analyze its performance in real time.

1.4 Organization of Thesis

In this thesis the framework of the thesis is presented as follows:

- Chapter 2 provides a literary review of the Independent Component Analysis, three algorithms, and use of ICA with FPGA for real time implementation.

- Chapter 3 provides the analysis and comparison of the each of the Algebraic ICA, Fast ICA and EASI ICA effectiveness and resources required for hardware implementation that ultimately led to the EASI ICA selection.
- Chapter 4 provides a further analysis of the EASI ICA and methods for setting the best parameter for the implementation of the algorithm on hardware.
- Chapter 5 provides the methodology and details for creating the EASI ICA system on hardware. This chapter also provides an analysis of the EASI ICA hardware implementation and compares it to the software simulations.
- Chapter 6 provides a summary and conclusion of the findings from the research.

CHAPTER TWO

INDEPENDENT COMPONENT ANALYSIS

2.1 Background Information on ICA

Principal Component Analysis (PCA) is defined by the eigenvectors of the covariance matrix of the input data. ICA is an extension of PCA that has been developed to utilize blind separation of independent sources from their linear mixtures. [9] Standard PCA is optimal in approximating the input data in the mean-square error sense. However, a problem arises because PCA does not provide the most meaningful representation for describing fundamental characteristics of data. ICA provides important analytic representations of the data than PCA. [12] Independent component analysis, or simply ICA, was introduced in 1986 by Jeanny Herault and Christian Jutte as a neural network based on Hebb learning law capable of performing blind signal separation. [1] ICA is used to recover independent sources from given sensor signals that have been mixed through unknown channels. [13] ICA recovers the source signals by finding a linear transformation that can maximize the mutual independence of the mixture. [10] Specifically, these algorithms try to separate a number of statistically independent signals from the same number of input signals are the linear sum of the first. [14] As can be seen in Figure 2.1, The ICA algorithms can be used to estimate the a_{ij} based on the information of their independence. [4] [5] [14]

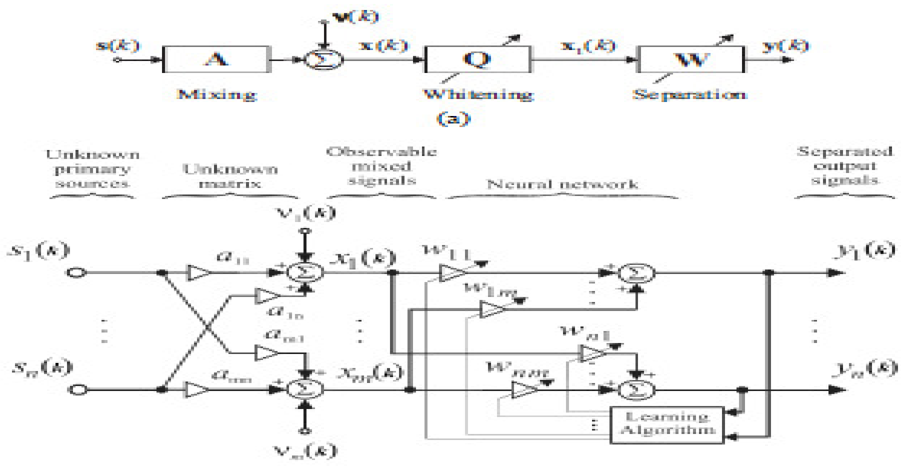

Figure 2.1: Basic diagram of the ideology of ICA.

A basic ICA algorithm begins with assuming n linear mixtures , $x = x_1 \ldots. x_n$. These mixture are made from independent component such that $x_j = a_{j1}s_1 + \cdots \ldots\ldots + a_{jn}s_n$ for all j. The mixtures x_j and the components of the sources $s = s_1 \ldots. s_n$ are assumed to be statistically independent and random. [4] The mixed signals in x_j , are a product of the mixing matrix A, and the independent source signals, s. [11]. The mixing matrix **A**, contains elements a_{ij}. The observed values in x_j are similar to the microphone x(t) signals in the cocktail party problem, but are random instead of a proper time signal. It is assumed that both the mixture variables and the independent components have zero mean. If this is not true, then the observable variables x_i can always be centered by subtracting the sample mean, which makes the model zero-mean. The starting point for ICA is the assumption that the components in s are statistically independent and have non-Gaussian distributions. The unknown mixing matrix, **A** is assumed to be square. After estimating the matrix **A**, it is possible to compute its inverse **W** or **B** to obtain the separated source signals. This can be done in the following equation: **y =Bx** [4]. Consider the global system denoted $\boldsymbol{C_t}$ which is obtained by chaining the mixing matrix $\boldsymbol{A}$ and the separating matrix$\boldsymbol{B}$, that is

$$\boldsymbol{C_t} \overset{\text{def}}{=} \boldsymbol{BA}$$

Ideally, an adaptive source separator should converge to a matrix $\boldsymbol{B}$, such that $\boldsymbol{B_t A = I}$, or equivalently, the global system ***Ct*** should converge to the *n* x *n* identity matrix ***I***. [12]
The key to estimating the ICA model is non-gaussainity and statistical independence. [11]
The method of ICA has specific characteristics that should be considered. ICA allows the separation of the signals whenever these are statistically independent by maximizing non-Gaussianity. Therefore Gaussian sources cannot be separated. In addition, there are two uncertainties in the method of ICA. The first is ICA cannot get the original amplitude of the mixed sources. The second is the outputs can be exchanged. [1] Consider two scalar valued random variable y_1 and y_2. These variables are said to be independent if information on the value of y_1 does not give any information on the value y_2 and vice versa. This should always be the case for $s_1 \ldots. s_n$ but not with the mixture variables $x_1 \ldots. x_n$. Independence can be defined by the probability densities. The joint probability density of y_1 and y_2 is p(y_1 , y_2). [4]
The marginal densities for y_1 and y_2 is denoted by the following equation:

$$p_1(y_1) = \int p(y_1, y_2) d\ y_2$$

$$p_2(y_2) = \int p(y_1, y_2) d\ y_1$$ [4]

Hence y_1 and y_2 are independent if and only if the joint probability density is factorable in the following equation:

$$p(y_1, y_2) = p_1(y_1) * p_2(y_2)$$ [4]

The matrix **A** is not identifiable for Gaussian independent components. If just one of the independent components is Gaussian the ICA model can be estimated. The classical measure of non-gaussianity is kurtosis. The kurtosis of y is defined by

$$Kurtosis = E\{y^4\} - 3\{E\{y^4\}\}^2$$

Its assumed that y is of unit variance the right hand side simplifies to $E\{y^4\} - 3$. This shows that kurtosis is simply a normalized version of the fourth moment $E\{y^4\}$. For a gaussian y, the fourth moment equals $3\{E\{y^4\}\}^2$. Kurtosis is zero for a Gaussian random variable. [4]

Kurtosis is not zero for most nongaussian random variables. Kurtosis can be both positive and negative. Random variables that have a negative kurtosis are called subgaussian and those with positive kurtosis are called supergaussian. Nongaussianity is measured by the absolute value of kurtosis. These are zero for a gaussian variable and greater than zero for most nongaussian random variables. Kurtosis or its absolute value has been widely used as a measure of nongaussity in ICA and related fields. The main reason is its simplicity. Computationally kurtosis can be estimated simply by using the fourth moment of the sample data. [4]

The statistical robustness achieved from the ICA and methods used depend on the choice of the objective function and the algorithm implementation. [11] The performance of a good algorithm is determined based on factors including the following:

- Convergence speed
- Memory requirements
- Numerical stability. [11]

ICA falls into basic methods adaptive or algebraic. The framework of most algebraic method can be expressed as whitening rotating. [15] After centering of the observed signal a linear transformation is performed on them using principal component analysis such as that the transformed variables are uncorrelated. ICA algorithms revolve around the orthogonal matrix rotating problem. The more the independent component could be separated completely the more powerful the ICA algorithm is. Most ICA algorithms focus on separating mixtures of multiple sources. [5]

2.2 Algebraic ICA

The Algebraic ICA (A-ICA) algorithm is based on the algebraic dot product between two n-d data vectors as a distance measure. The dot product between two vectors $\bar{a}\ and\ \bar{b}$ in same basis $\Psi \epsilon R^n$ is defined as $\bar{a} * \bar{b} \triangleq \bar{a}^T * \bar{b} = |\bar{a}||\bar{b}| \cos\theta_{ab}$ where θ_{ab} n-d angle between the vectors $\bar{a}\ and \bar{b}$. If the two vectors are lying along the same n-d direction, the absolute value of the cosine of the n-d angle θ_{ab} is maximum or one. If they are orthogonal to each other, the cosine of the angle θ_{ab}becomes zero. After projection onto a unit hyper-sphere, the notion of minimum distance is equivalent to the notion of maximizing the absolute dot-product between the projected vectors. Using the absolute value of the dot product produces a sign ambiguity that is inherent in any ICA algorithm i.e. $|\bar{a}\,\bar{b}| = |\bar{b}\,\bar{a}|$ where $\bar{a}$ = -$\bar{a}$. Another useful property for the dot-product is its insensitivity to the order of the product, i.e . $|\bar{a}\,\bar{b}| = |\bar{b}\,\bar{a}|$ [16]. [17]
The A-ICA algorithm is fast stable and does not use the whitening rotating framework. The A-ICA algorithm is for two sources and based on non-whitening preprocessing. [12]

Assume there are two source signals. If $s = [s_1\ s_2]^T$ and $x = [x_1\ x_2]^T$ are a random source vector and mixed signal vector. This system is represented by x = **A**s where **A** is a 2x2 mixing matrix in the form

$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ [15]

According the ICA principle, s_1 and s_2 are mutually independent with unit variance and zero mean.

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} * \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$ [18]

The symbols α and β are unknown mixing rates given by

$$\beta = \frac{(\alpha C_2 - C_3)}{(\alpha C_3 - C_1)} \text{ [18]}$$

$(C_2C_{10} - C_{11}C_3)\alpha^4 + (3C_9C_3 - 3C_8C_2 - C_3C_{10} + C_1C_{11})\alpha^3 + (3C_6C_2 + 3C_8C_3 - 3C_9C_1 - 3C_7C_3)\alpha^2 + (C_5C_3 - 3C_7C_1 - 3C_6C_3 - C_2C_4)\alpha + C_3C_4 - C_1C_5 = 0$ [18]

$$C_1 = E[X_1{}^2] - \{E[X_1]\}^2$$

$$C_2 = E[X_2{}^2] - \{E[X_2]\}^2$$

$$C_3 = E[X_1X_2] - E[X_1]E[X_2]$$

$$C_4 = E[X_1{}^4] - E[X_1{}^3]E[X_1]$$

$$C_5 = E[{X_1}^3 X_2] - E[{X_1}^3]E[X_2]$$

$$C_6 = E[{X_1}^3 X_2] - E[{X_1}^2 X_2]E[X_1]$$

$$C_7 = E[{X_1}^2 {X_2}^2] - E[{X_1}^2 X_2]E[X_2]$$

$$C_8 = E[{X_1}^2 {X_2}^2] - E[X_1 {X_2}^2]E[X_1]$$

$$C_9 = E[X_1 {X_2}^3] - E[X_1 {X_2}^2]E[X_2]$$

$$C_{10} = E[X_1 {X_2}^3] - E[X_1]E[{X_2}^3]$$

$$C_{11} = E[{X_2}^4] - E[{X_2}^3]E[X_2]$$

Where E[] denotes expectation or the mean of the random x vectors. [18] [17]

2.3 Fast ICA

Fast ICA is a fixed point iterative algorithm uses a nonlinear function f (y) = tanh (y), which is applied to the separation matrix, **W**. **W** is continuously recalculated at every iteration of the algorithm. The input to the Fast ICA algorithm must be whitened by three steps. The first step is centering the data over the average. The second is to normalize the variance. The third is to make the data orthogonal. Multiple steps are necessary to implement the Fast ICA algorithm. The first step is to center the data to make its mean zero. The next to step is to whiten the data to give **z**. The third step is to choose an initial random vector w of unit norm. The next step is to let **w**=E{**z**g(**wTz**)}- E{g'(**wTz**)}**w**. In the next step let **w=w/**‖**w**‖. The last step of the Fast ICA check for convergence of the algorithm. If there isn't any convergence the algorithm repeats the third and four steps. If there is convergence then the **w** is used in **y = w*x** to obtain the separated signal. The Fast ICA are show in detail below: [1]

1. Center the data to be zero mean

$$MeanValue = Mean\{s\}$$
$$snew = s - MeanValue * Matrix\ of\ 1s$$

2. Recreate the mixed signals

$$x = \boldsymbol{A} * s$$

3. Whiten the data

$$x = \boldsymbol{E}\boldsymbol{D}^{-1/2}\boldsymbol{E}^{\boldsymbol{T}}x$$

4. Choose and initial w

$$\boldsymbol{w} = \boldsymbol{A}^{-1}$$

5. Compute **w**

$$\boldsymbol{w} = E\{\boldsymbol{zg}(\boldsymbol{w^T z})\} - E(g'(\boldsymbol{w^T z})\}\boldsymbol{w}$$

6. Find the norm of **w**

$$\boldsymbol{w} = \frac{\boldsymbol{w}}{\sqrt{\|\boldsymbol{ww^T}\|}}$$

   See if there is convergence. If there isn't repeat steps five and six.
7. Compute **w**

$$\boldsymbol{W} = \frac{3}{2}\boldsymbol{w} - \frac{1}{2}\boldsymbol{ww^T w}$$

8. Perform separation

$$yt = \boldsymbol{W} * x$$

2.4 EASI ICA

The EASI algorithm was initially created by Cardoso. The theory behind its approach revolves around the use of batch estimators that are equivariant. The key property shared by equivariant batch estimators for source separation is that they offer uniform performance [12] The Equivariant Adaptive Separation via Independence (EASI) algorithm is simple in parallel structure. The algorithm is performed using the following equations:

1. Create y_k vectors , **B** initially is an identity matrix

$$yk(t) = \boldsymbol{B}(t) * x(t)$$

2. Create the H matrix using the yk and a nonlinearity function

$$\boldsymbol{H}(t) = I - yk(t) * yk(t)^T + f\big(y(t)\big) * yk(t)^T - y(t) * f(y(t))^T$$

3. Create the **B** matrix with **H** , current **B** and the learning rate μ

$$\boldsymbol{B}(t+1) = \boldsymbol{B}(t) + \mu\boldsymbol{H}(t)\boldsymbol{B}(t)$$

4. Repeat steps 1 through 3 until convergence
5. Generate separated signals

$$y_t(t) = \boldsymbol{B}(t+1) * x(t)$$

The **B** matrix updates itself each cycle of the algorithm. The matrix initially is an identity matrix. The f(y) represents the nonlinearity used that will shape the convergence and separation of the mixed signals. Upon convergence of the algorithm the final updated matrix if multiplied with the mixed signals to produce vectors of separated signals. The nonlinearity *f(y)* = -tanh(y) and a learning rate of $\mu = 2^{-9}$ are typically used with this algorithm. [19] [7] [8]

2.5 ICA Implementation With Field Programmable Gate Arrays (FPGA)

FPGA technology can implement the digital signal processing algorithm and quickly verify the result in hardware. Most FPGAs have on chip hardware multipliers and memory blocks which make them fit in the implementation of ICA which require high volumes of mathematical operations. VHDL used to the design the hardware. Floating point arithmetic with high accuracy is necessary in calculation and large dynamic range of numbers is necessary for such signal processing techniques. Floating point is difficult to implement on an FPGA because of the arithmetic complexity and large number of logic elements needed to implement it. [10]

The complicated arithmetic, the iterative computation, slow convergence rate, and the generally large volumes of raw and processed data make ICA algorithms time-consuming for software implementation. Hardware implementation provides potentially fast and real-time solutions. Software implementation is useful for investigating the capabilities of ICA algorithms. Hardware implementation is essential to benefit from the parallel architecture and to facilitate high-speed processing. The major difference between hardware and software implementations is hardware subroutines are executed by integrated circuits (ICs) instead of a series of microinstructions. Hardware implementation also solves the insufficient memory problem encountered by software for large data sets and high dimensionality. [20]

The FPGA based on the reconfiguration technology are the most economic and efficient solutions to ICA. The complicated arithmetic of ICA is one of the main barricades in ICA hardware implementation. The hierarchy involves dividing an ICA process into sub processing modules until the complexity of the bottom level sub modules becomes manageable. These sub modules are independently developed, integrated put into a design and development environments. FPGAs provide the most economic and efficient solutions to comparatively simple ICA algorithms. FPGAs are standard and general-purpose products fabricated by hardware companies. FPGAs are developed based on reconfigurable technologies that allow end users to modify their designs for multiple times and program the interconnections instead of waiting several weeks for the final fabrication. These savings in the development expense and turnaround time of prototyping directly lead to time-to-market reduction and profit increase.

Typical FPGAs are composed of a two-dimensional array of input/output blocks, interconnects, and configurable logic blocks (CLBs) that can be customized to implement logic functions. The programmable interconnects between these CLBs allow end users to implement the multilevel logic functions [20]

CHAPTER THREE

ICA ALGORITHM COMPARISON AND SELECTION

3.1 Comparison Criteria for ICA Data

The Algebraic ICA (A-ICA), Fast ICA, and EASI ICA were implemented in Matlab to analyze each algorithms ability to separate the mixed signals. The independent sources used were cosine signals of varying frequencies.

Each algorithm's effectiveness was accessed on the following criteria:

- Convergence speed
- Visible separation:
- Separated signal FFT spectrum energy error
- Mixing matrix and separated matrix product error
  (Mixing Matrix * Separation Matrix = Identity Matrix)
- Separated matrix and mixing matrix difference error
  (Separation Matrix – Inverse of Mixing = 0)
- Number of resources required for implementation

3.2 Algebraic ICA Data

The first algorithm analyzed was the A-ICA. The A- ICA was simulated with cosine signals of 2 kHz and 7 kHz frequencies as shown in Figure 3.2.1.

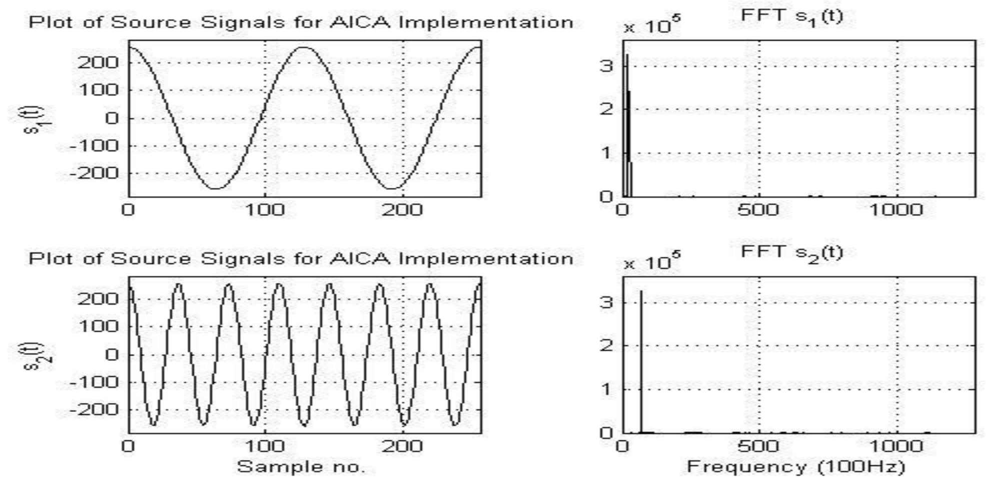

Figure 3.2.1: Independent source signals used for A-ICA simulation.

The following mixing matrix was used to create the mixed signals of the sources:

$$\boldsymbol{A} = \begin{pmatrix} 1 & .25 \\ -.75 & 1 \end{pmatrix}$$

The mixed signals generated from the product of mixing matrix and the source signals can be seen in Figure 3.2.2.

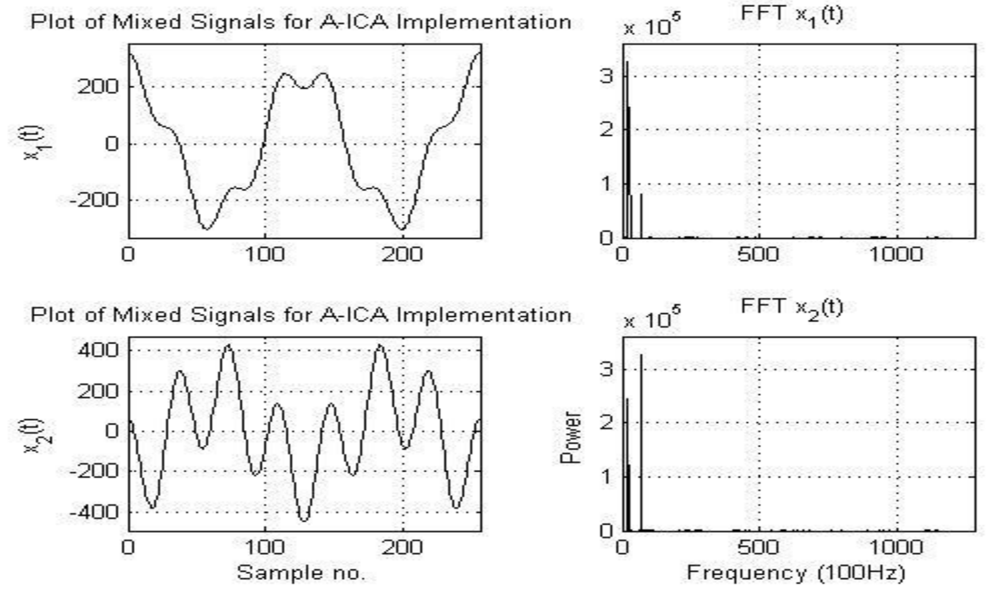

Figure 3.2.2: Mixed signals used for A-ICA simulation.

The A-ICA algorithm execution created the following separated signals as shown in Figure 3.2.3.

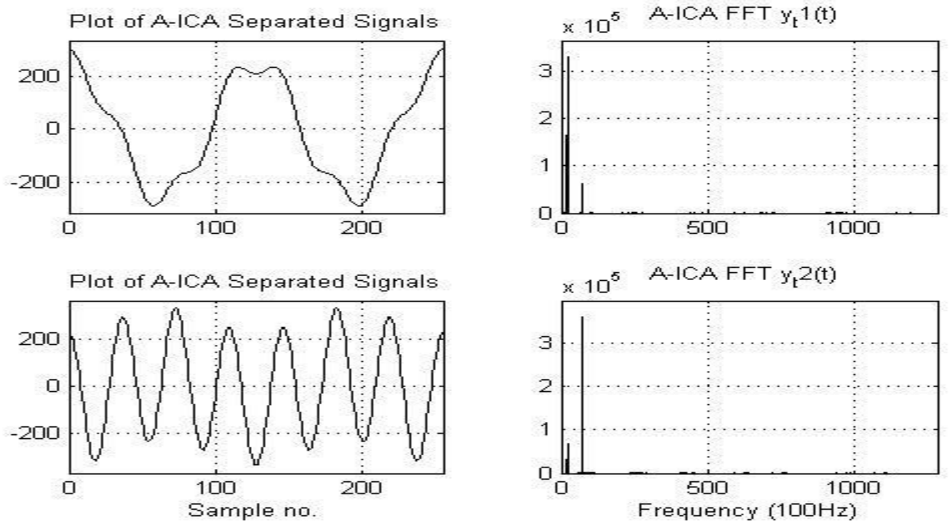

Figure 3.2.3: Separated signals used for A-ICA simulation.

The signals plotted in Figure 3.2.1 are clearly distorted replicas of the original source signals. Figure 3.2.4 clearly outlines differences between the source and separated signals. Despite the fast convergence speed A-ICA visibly does not effectively separate the signals mixed signals.

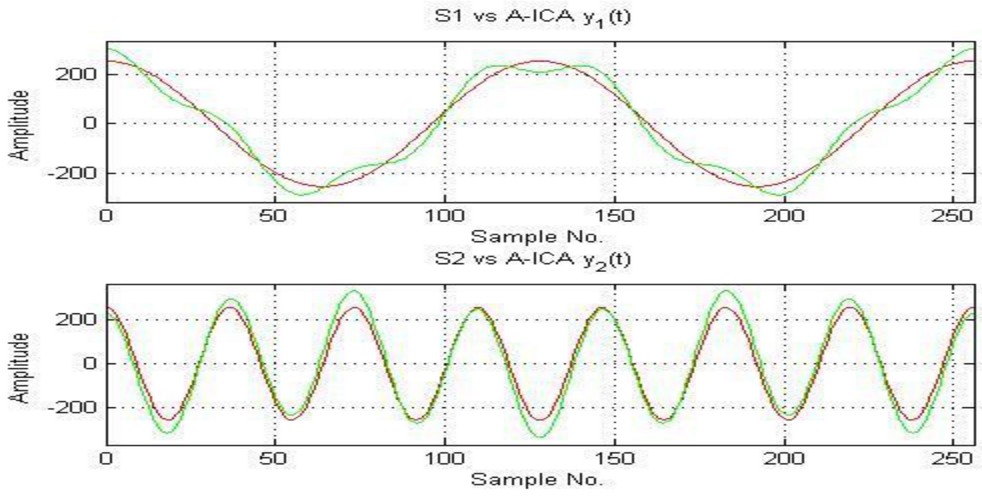

Figure 3.2.4: Source and separated signals comparison of the A-ICA simulation.

As shown in Table 3.2.1 , the maximum FFT spectrum energies were detected to be approximately 2 kHz and 7 kHz for signal one and signal two respectively. The excess energies detected in the FFT spectrum plots for signals one and two created spectrum energy errors of 0.374% and 2.3741% respectively. The product of the mixing matrix and separation matrix had an error of 10.43%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 1.98%.

Table 3.2.1: A-ICA simulation data analysis results.

Table 3.2.1: A-ICA Data Analysis Results	
Performance Aspect	Quantitative Values
Convergence Speed	1 iteration
FFT Measurements	
Signal 1 – Max Freq (%Error)	2 kHz (03.741%)
Signal 2 – Max Freq (%Error)	7 kHz (2.3741%)
Mixing Separation Product Error	10.43%
Mixing Separation Difference Error	1.98%

The A-ICA requires a large number of resources for implementation as see in Table 3.2.2. Although the ICA requires only an iteration to converge, the algorithm requires over well over 100 multiplier and adders of resources for hardware implementation. The finite number of resources is not known because several elements of the algorithm depend on the length of the signal.

Table 3.2.2: The total required resources needed to implement the A-ICA.

Table: 3.2.2 A-ICA Required Resources			
Equation	Operation	Breakdown N = # of elements	Total Resources
${X_1}^2$	1 multipliers	1 mults	1 multipliers
${X_1}^3$	2 multipliers	2 mults	2 multipliers
${X_1}^4$	3 multipliers	3 mults	3 multipliers
${X_2}^2$	1 multipliers	1 mults	1 multipliers
${X_2}^3$	2 multipliers	2 mults	2 multipliers
${X_2}^4$	3 multipliers	3 mults	3 multipliers
$X_1 * X_2$	1 multipliers	1 mults	1 multipliers
${X_1}^2 * {X_2}^2$	3 multipliers	3 mults	3 multipliers
${X_1}^2 * X_2$	2 multipliers	2 mults	2 multipliers
${X_1}^3 * X_2$	3 multipliers	3 mults	3 multipliers
$X_1 * {X_2}^2$	2 multipliers	2 mults	2 multipliers
$X_1 * {X_2}^3$	3 multipliers	3 mults	3 multipliers
E[*] = Expectation	Mean : (N-1) adders N dividers	E[*] = Expectation	Mean : (N-1) adders N dividers

Table: 3.2.2 A-ICA Required Resources continued			
Equation	Operation	Breakdown N = # of elements	Total Resources
$C_1 = E[{X_1}^2] - \{E[X_1]\}^2$ $C_2 = E[{X_2}^2] - \{E[X_2]\}^2$ $C_3 = E[X_1X_2] - E[X_1]E[X_2]$ $C_4 = E[{X_1}^4] - E[{X_1}^3]E[X_1]$ $C_5 = E[{X_1}^3X_2] - E[{X_1}^3]E[X_2]$ $C_6 = E[{X_1}^3X_2] - E[{X_1}^2X_2]E[X_1]$ $C_7 = E[{X_1}^2{X_2}^2] - E[{X_1}^2X_2]E[X_2]$ $C_8 = E[{X_1}^2{X_2}^2] - E[X_1{X_2}^2]E[X_1]$ $C_9 = E[X_1{X_2}^3] - E[X_1{X_2}^2]E[X_2]$ $C_{10} = E[X_1{X_2}^3] - E[X_1]E[{X_2}^3]$ $C_{11} = E[{X_2}^4] - E[{X_2}^3]E[X_2]$	11*1 Expectation 11*1 multipliers 11*1 adder 11*1 Expectation Squared	11*(N-1) adders 11*N dividers 11*1 multiplier 11*1 adder (subtraction) 11*(N-1)^2 adders 11*N^2 dividers	11 Expectation (11N-11) adders ,11N dividers)) 11 multipliers 11 adder 11 Expectation Squared (121N^2+242N+11) adders + 121N^2 dividers)
$\beta = \frac{(\alpha C_2 - C_3)}{(\alpha C_3 - C_1)}$	2 Multipliers 2 adders	2*(1 mults + 1 adder) (2 sub)	2 Multipliers 2 adders
$(C_2C_{10} - C_{11}C_3)\alpha^4 +$ $(3C_9C_3 - 3C_8C_2 \; -C_3C_{10} + C_1C_{11})\alpha^3 +$ $(3C_6C_2 + 3C_8C_3 - 3C_9C_1 - 3C_7C_3)\alpha^2$ $+$ $(C_5C_3 - 3C_7C_1 - 3C_6C_3 - C_2C_4)\alpha$ $+ C_3C_4 - C_1C_5 = 0$	25 Multipliers 14 Adders (8 Subtraction)	(2+7+8+6+2) Mults (1+1+1+1+1+1 +1+2+2+3+1) Adders (1+2+2+3+1) Sub	25 Multipliers 14 Adders (8 Subtractions)
		Total Resources	64 multipliers 121N^2 + 11Ndividers 121N^2+253N+49 adders (8 subtractions)

3.3 Fast ICA Data

The Fast ICA was simulated with cosine signals of 2 kHz, 7 kHz, and 10 kHz frequencies as shown in Figure 3.3.1.

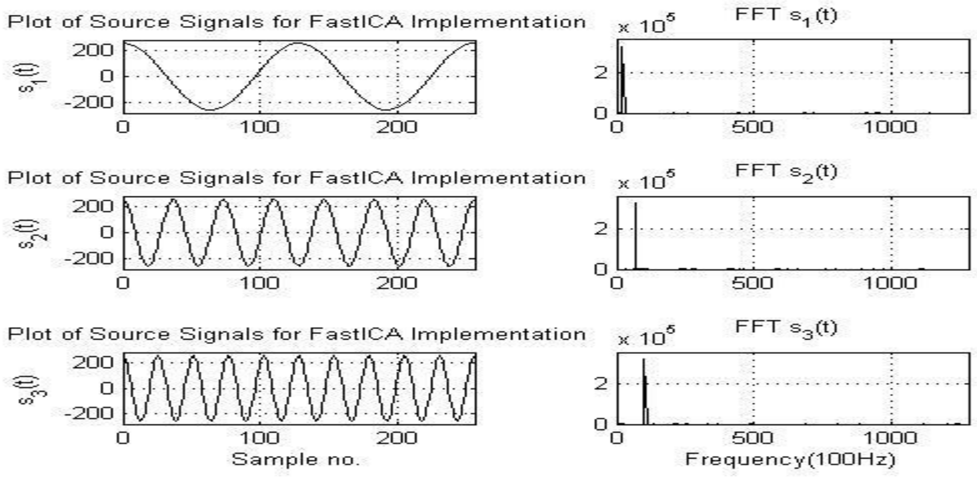

Figure 3.3.1: Independent source signals used for Fast ICA simulation.

The following mixing matrix was used to create the mixed signals of the sources:

$$\boldsymbol{A} = \begin{pmatrix} 0.25 & 1.0 & 0.5 \\ 1.0 & 0.5 & -0.75 \\ 0.5 & -1.0 & 0.75 \end{pmatrix}$$

The mixed signals created from the product of mixing matrix and the source signals can be seen in Figure 3.3.2.

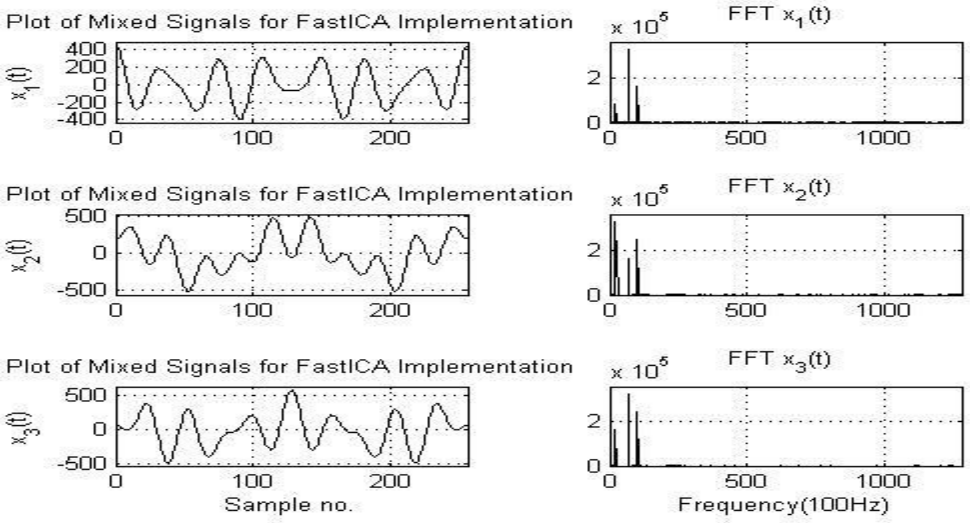

Figure 3.3.2: Mixed signals used for Fast ICA simulation.

The Fast ICA algorithm simulation produced the following separated signals as shown in Figure 3.3.3.

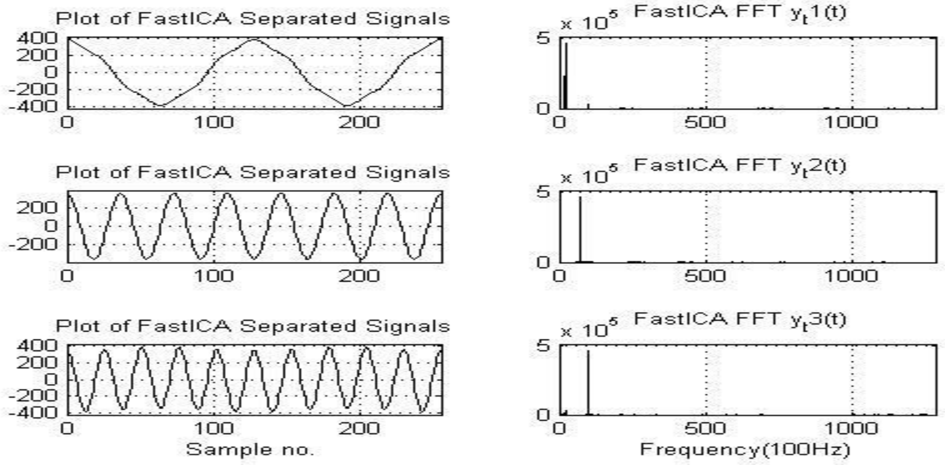

Figure 3.3.3: Separated signals used for Fast ICA simulation.

The signals plotted in Figure 3.3.3 appear to be very similar to the source signals. Figure 3.3.4 highlights the similarities between the source and separated signals.

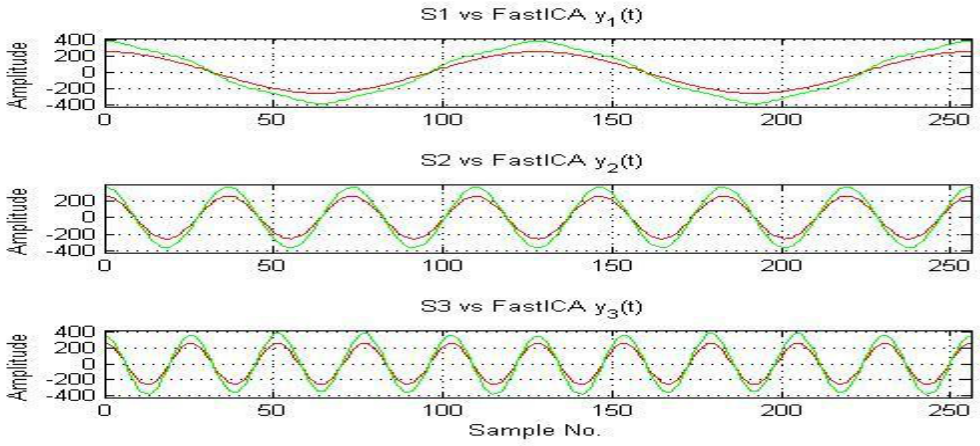

Figure 3.3.4: Source signals and separated signal comparison of the Fast ICA simulation.

As shown Table 3.3.1, the maximum FFT spectrum energies were detected at approximately 2 kHz for signal one, 7 kHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one, two and three created a spectrum error of 0.116%., 2.000% and 2.116% respectively. The product of the mixing matrix and separation matrix had an error of 0.995%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 1.483%.

Table 3.3.1: Fast ICA simulation data analysis results.

Table 3.3.1: Fast ICA Analysis Data	
Performance Aspect	Quantitative Values
Convergence Speed	11 iterations
FFT Measurements	
Signal 1 – Max Freq (%Error)	2 kHz (0.116%)
Signal 2 – Max Freq (%Error)	7 kHz (2.000%)
Signal 3 – Max Freq (%Error)	10 kHz (2.116%)
Mixing Separation Prod Error	0.995%
Mixing Separation Diff Error	1.483%

The Fast ICA requires a large number of resources for implementation as can be seen in Table 3.3.2. The algorithm requires over well over 200 multipliers and adders of resources for hardware implementation. The finite number of resources is not known because several elements of the algorithm depend on the length of the signal.

Table 3.3.2: The total of required resources needed to implement Fast ICA.

Table 3.3.2: Fast ICA Required Resources			
Equation	Operation	Breakdown N = # of elements	Total Resources
Center data to make mean = 0 $MeanValues = Mean\{s\}$ $Snew = S - MeanValues * Matrix\ of\ 1s$	9 multipliers 7 + (N-1) adders N dividers (1 subtraction)	3 mults,2 adders 3 mults,2 adders 3 mults,2 adders (N-1) adders N dividers 1adder(subtraction)	9 multipliers 7 + (N-1) adders N dividers (1 subtraction)

Table 3.3.2: Fast ICA Required Resources continued			
Equation	Operation	Breakdown N = # of elements	Total Resources

$x = \boldsymbol{A} * s$
Multipliers = (n^2)
Adders = (n-1)*n
	9 multipliers
	6 adders
		3 mults,2 adders
		3 mults,2 adders
		3 mults,2 adders
			9 multipliers
			6 adders

Whiten Data to give z $x = \boldsymbol{E}\boldsymbol{D}^{-1/2}\boldsymbol{E}^{\boldsymbol{T}}x$ Choose initial **W** $\boldsymbol{W} = \boldsymbol{A}^{-1}$	27 multipliers 27 adders (18 subtraction)	9*(3 multipliers + 3 adders) (9*2 subtraction)	27 multipliers 27 adders (18 subtraction)
$\boldsymbol{w} = E\{\boldsymbol{z}\boldsymbol{g}(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{z})\} - E(g'(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{z})\}\boldsymbol{w}$	2*(9 multipliers +6 adders) 2 multipliers 2* (N-1) adders 2*N dividers	2*(3 mults,2 adders 3 mults,2 adders 3 mults,2 adders) 2 multipliers 2* (N-1) adders 2*N dividers	N*(20 multipliers 12+2N-2 adders 2N dividers)
$\boldsymbol{w} = \frac{\boldsymbol{w}}{\sqrt{\lVert\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\rVert}}$	2*(9 multipliers 6 adders) (divider) 6 adders(norm)	3 mults,2 adders 3 mults,2 adders 3 mults,2 adders 3 mults,2 adders 3 mults,2 adders 3 mults,2 adders 6 adders	N* (18 multipliers +6 adders (divider) +6 adders(norm))
Repeat n times until convergence : $\boldsymbol{w} = E\{\boldsymbol{z}\boldsymbol{g}(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{z})\} - E(g'(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{z})\}\boldsymbol{w}$ $\boldsymbol{w} = \frac{\boldsymbol{w}}{\sqrt{\lVert\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\rVert}}$			

Table 3.3.2: Fast ICA Required Resources continued			
Equation	Operation	Breakdown N = # of elements	Total Resources
$\boldsymbol{w} = \frac{3}{2}\boldsymbol{w} - \frac{1}{2}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$	29 multipliers (2 multipliers) 19 adders (1subtraction)	27multipliers 2 multipliers 18 adders 1 adder (1subtraction)	29 multipliers 19 adders

$yt = \boldsymbol{w} * x$
Multipliers = (n^2)
Adders = (n-1)*n
	9 multipliers
	6 adders
		3 mults,2 adders
		3 mults,2 adders
		3 mults,2 adders
			9 multipliers
			6 adders

		Total Breakdown For Finite elements	83 multipliers 58 adders (18 Subtraction)
		Complete Total	141 finite Total Resources

3.4 EASI ICA Data

The EASI ICA was simulated with cosine signals of 2 kHz, 7 kHz, and 10 kHz frequencies as shown in Figure 3.4.1

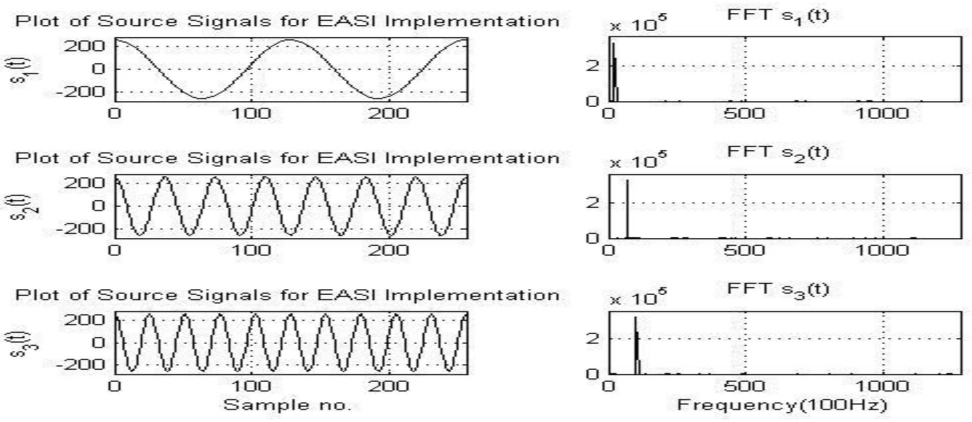

Figure 3.4.1: Independent source signals used for EASI ICA simulation.

The following mixing matrix was used to create the mixed signals of the sources:

$$\boldsymbol{A} = \begin{pmatrix} 0.25 & 1.0 & 0.5 \\ 1.0 & 0.5 & -0.75 \\ 0.5 & -1.0 & 0.75 \end{pmatrix}$$

The mixed signals created from the product of mixing matrix and the source signals can be seen in Figure 3.4.2.

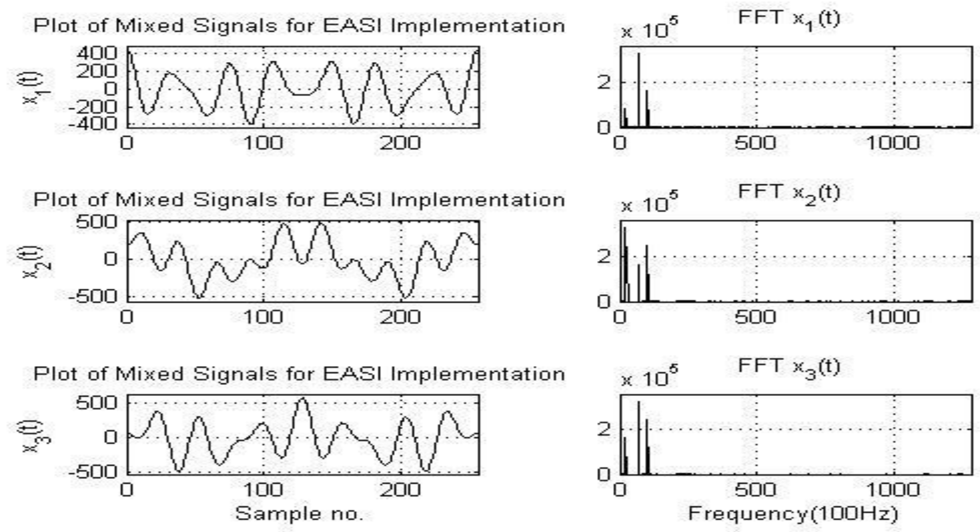

Figure 3.4.2: Mixed signals used for EASI ICA simulation.

The EASI ICA algorithm was simulated to generate the following separated signals as shown in Figure 3.4.3.

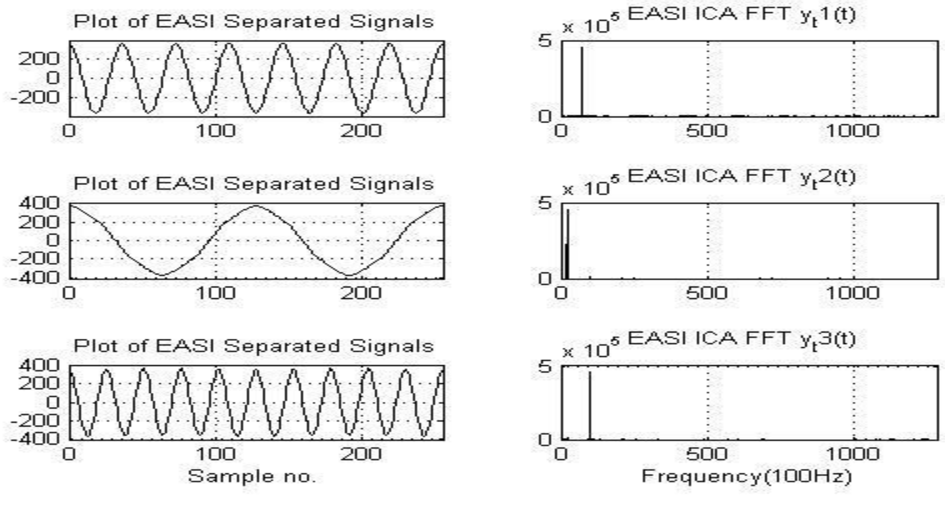

Figure 3.4.3: Separated signals used for EASI ICA simulation.

The signals in Figure 3.4.3 appear to have similar to the appearance of the source signals. Figure 3.4.4 provides a visual comparison of the source and separated signals.

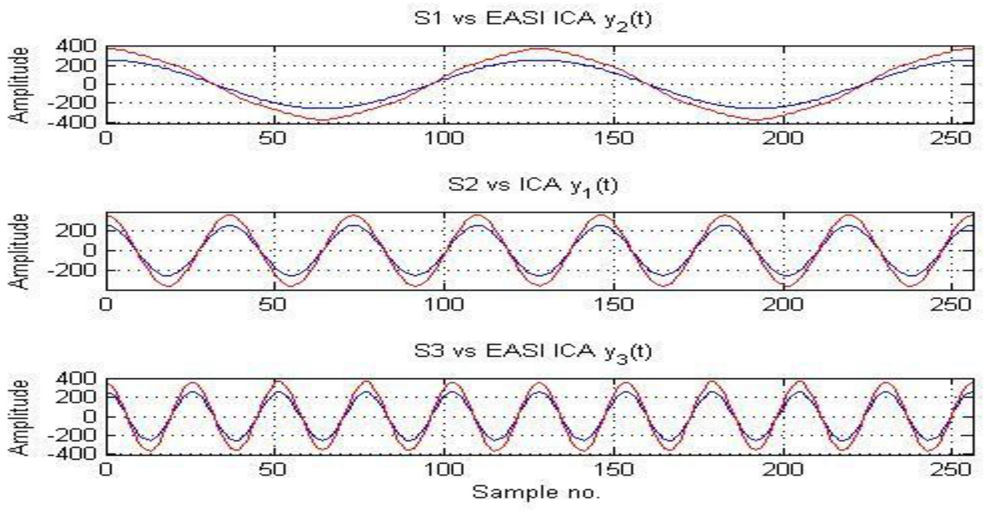

Figure 3.4.4: Source and separated signals comparison of the EASI ICA simulation.

As shown in Table 3.4.1, the maximum FFT spectrum energies were detected at approximately 7 kHz for signal one, 2 kHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created an spectrum error of 0.0049%. Signals two and three experienced spectrum errors of 2.0555%and 2.0624% respectively. The product of the mixing matrix and separation matrix had an error of 1.199%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.0823%.

Table 3.4.1 EASI ICA simulation data analysis results.

Table 3.4.1: EASI ICA Analysis Data	
Performance Aspect	Quantitative Values
Convergence Speed	10 iterations
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (0.0049%)
Signal 2 – Max Freq (%Error)	2 kHz (2.0555%)
Signal 3 – Max Freq (%Error)	10 kHz (2.0624%)
Mixing Separation Product Error	1.199%
Mixing Separation Difference Error	0.0823%

The EASI ICA requires a significant number of resources for implementation as shown in Table 3.4.2. The EASI ICA algorithm needs 63 multipliers, 48 adders, and 18 subtractions resources implement this algorithm.

Table 3.4.2: The total required resources needed to implement the EASI ICA

Table 3.4.2: EASI ICA Required Resources			
Equation	Operation	Breakdown N = # of elements	Total Resources

$Yk = B * X$
Multipliers = (n^2)
Adders = (n-1)*n
	9 multipliers
	6 adders
		3 mults,2 adders
		3 mults,2 adders
		3 mults,2 adders
			9 multipliers
			6 adders

H = I – yk*yk' + yk*f ' - f*yk'
Multipliers =n^3
Adders = n^3
	27 multipliers
	27 adders
	(18 subtraction)
		9*(3 multipliers
		+ 3 adders)
		(9*2 subtraction)
			27 multipliers
			27 adders
			(18 subtraction)

B = B + m*H*B
Multipliers = 2* n^2
Adders = n^2
	18 multipliers
	9 adders
		9*(2 mults
		+ 1 adders)
			18 multipliers
			9 adders

$Yt = B * X$
Multipliers = (n^2)
Adders = (n-1)*n
	9 multipliers
	6 adders
		3 mults,2 adders
		3 mults,2 adders
		3 mults,2 adders
			9 multipliers
			6 adders

		Total Breakdown	63 multipliers 48 adders (18 Subtraction) 111 Total Res.

3.5 Comparisons of ICA Algorithms

3.5.1 ICA Algorithm Comparison Data

The data in Table 3.5.1.1 shows the convergence speed, and visible characteristics of the signals of the algorithms. The A-ICA had the fastest algorithm convergence speed, but was most inaccurate. The Fast ICA and EASI ICA were the most accurate algorithms visibly.

Table 3.5.1.1: Comparison of convergence speed and visible qualifications for each ICA.

Table 3.5.1.1 Comparison of Convergence Speed & Visible Qualifications			
Algorithm	Iterations	Speed	Visibly Accurate
A-ICA	1	Fast	No
Fast ICA	11	Medium	Yes
EASI ICA	10	Medium	Yes

As shown in table 3.5.1.2 the EASI algorithm had the smallest FFT spectrum error. The A-ICA had the worst error of the three algorithms.

Table 3.5.1.2: Comparison of the FFT spectrum error and frequency location for each ICA.

Table 3.5.1.2 Comparison of FFT Spectrum Error		
Algorithm – Signal #	Percent Error	Max Freq Location
A-ICA – Signal 1	0.374%	2 kHz
A-ICA – Signal 2	2.374%	7 kHz
Fast ICA - Signal 1	0.116%	2 kHz
Fast ICA - Signal 2	2.000%	7 kHz
Fast ICA - Signal 3	2.116%	10 kHz
EASI ICA - Signal 1	0.0049%	7 kHz
EASI ICA - Signal 2	2.0555%	2 kHz
EASI ICA - Signal 3	2.0624%	10 kHz

In Table 3.5.1.3 the EASI had the smallest mixing matrix and separation matrix product error. The A-ICA had the worst percentage error of the three algorithms.

Table 3.5.1.3: Comparison of the mixing and separation matrices product error for each ICA.

Table 3.5.1.3 Comparison of the Product of Mixing and Separated Matrices Errors	
Algorithm	Percent Error
A-ICA	10.43%
Fast ICA	1.483
EASI ICA	0.0823%

In table 3.5.1.4 shows all of percent errors for the mixing and separation matrix for all the algorithms. The A-ICA had the worst percentage error of the three algorithms.

Table 3.5.1.4: Comparison of the difference of mixing and separation matrices for each ICA.

Table 3.5.1.4 Comparison of the Difference of the Mixing and Separation Matrices Errors	
Algorithm	Percent Error
A-ICA	1.98%
Fast ICA	0.995%
EASI ICA	1.199%

3.5.2 ICA Algorithm Comparison Discussion

The A- ICA requires on one iteration to converge, and is relatively quick to execute. The A-ICA algorithm does not require pre-whitening, which makes the algorithm easy to implement. However the algorithm poorly separates the signals, and works best for two signals. The algorithm requires a large number of resources for implementation. The Fast ICA has a relatively fast convergence and effectively separates mixed signals. The Fast ICA separated several source signals. The Fast ICA algorithm requires pre-whitening. The need for pre-whitening means the Fast ICA is a complex algorithm to implement and needs a vast supply of resources for implementation.The EASI algorithm has relatively fast convergence and effectively separates mixed signals for multiple sources. The EASI algorithm does not need pre-whitening, and is very simple to implement in hardware. The only drawback with this algorithm is a large number of resources required for hardware implementation.

3.6 Conclusion

The Algebraic-ICA, Fast ICA and EASI ICA were implemented and their performance was analyzed. The performance and the number of resources required for implementation of each algorithm was compared and contrasted. The EASI algorithm was chosen as the algorithm for further analysis and implementation on hardware. The convergence speed, simplicity of the algorithm, and effectiveness made it the best algorithm in the end.

CHAPTER FOUR

EASI ANALYSIS

After selection for hardware implementation, the EASI algorithm performance was further examined. The performance and other elements of the algorithm were analyzed to see what the best conditions are necessary for the most accurate implementation in hardware.

4.1 Mixing Matrix Analysis

The mixing matrix was analyzed to see whether it was the best for the EASI ICA implementation. The best mixing matrix need to meet the following criteria:

- The product of the mixing matrix and the separation matrix error needed to be minimal or close to an identity matrix as possible.
  (Mixing Matrix * Separation Matrix = Identity Matrix)
- The difference of the separation matrix and the inverse of the mixing matrix needed to be minimal or close to 0 as possible.
  (Separation Matrix – Inverse of Mixing = 0)
- The mixing matrix should contain values that require a low cost and are easy to design in hardware.

Four mixing matrices were analyzed to see which would be the best for EASI ICA implementation. The nonlinearity of hyperbolic tangent, and independent sources used in the previous implementation of EASI ICA in chapter three.

4.1.1 Mixing Matrix One Data

Mixing matrix one and its inverse are given by the following values:

$$\boldsymbol{A} = \begin{pmatrix} 0.25 & 1.0 & 0.5 \\ 1.0 & 0.5 & -0.75 \\ 0.5 & -1.0 & 0.75 \end{pmatrix} \boldsymbol{A^{-1}} = \begin{pmatrix} 0.203 & 0.678 & 0.542 \\ 0.610 & 0.0340 & -0.373 \\ 0.678 & -0.407 & 0.475 \end{pmatrix}$$

The values represent the original mixing matrix chosen to test the algorithms. The values were chosen because each value is easy to implement in hardware. The mixed signals created with the mixing matrix are shown in Figure 4.1.1.1.

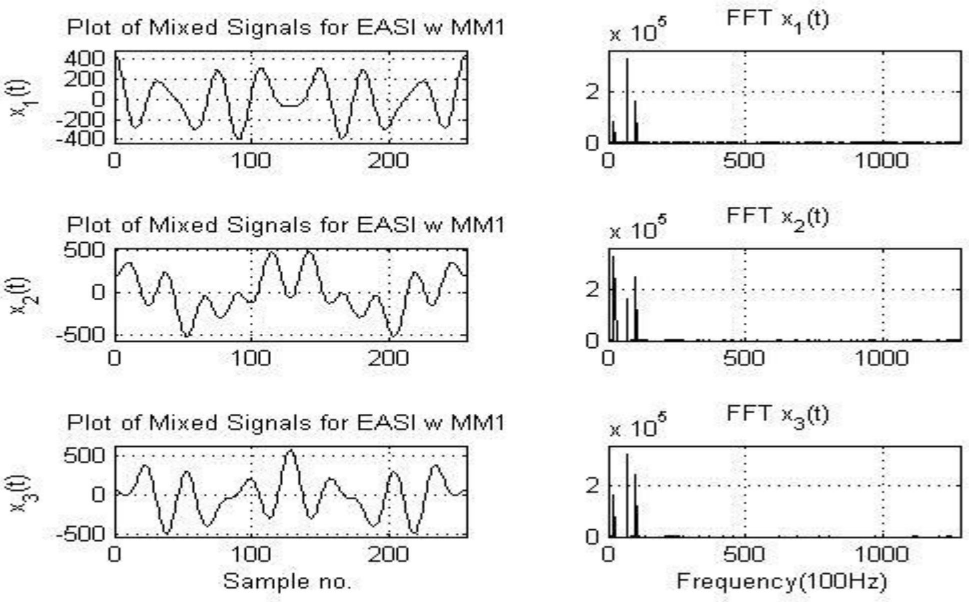

Figure 4.1.1.1 Mixed signal created with mixing matrix one.

The separations of the mixed signals are shown in Figure 4.1.1.2. The visual comparison of the source and separated signals are shown in Figure 4.1.1.3.

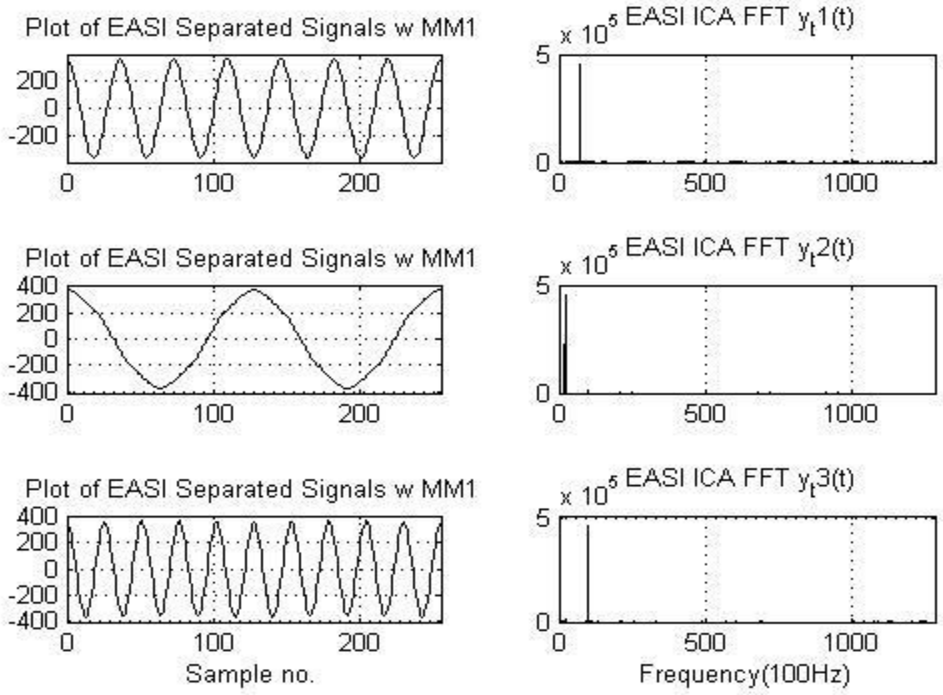

Figure 4.1.1.2: Separated signals created with mixing matrix one.

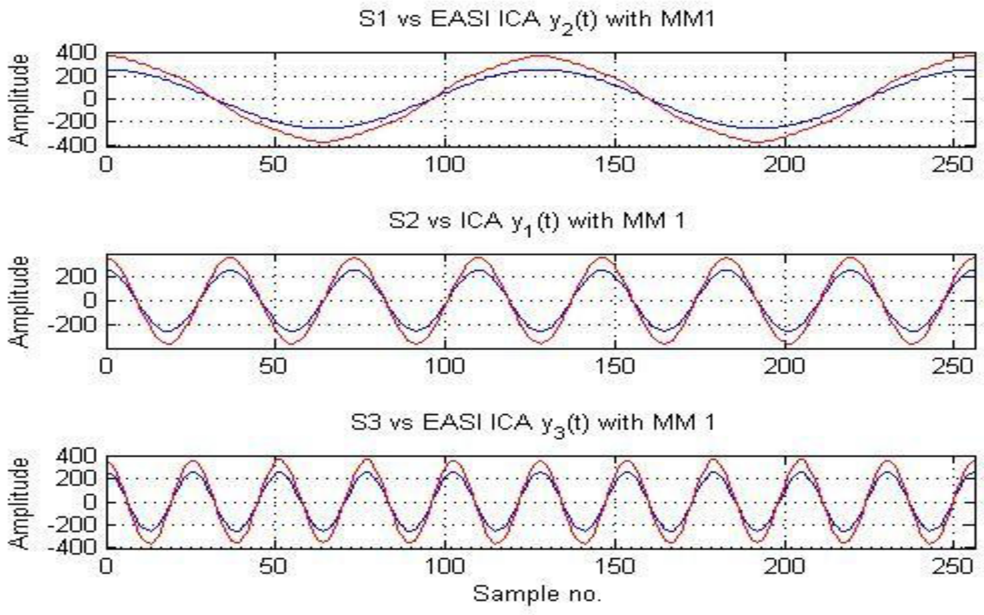

Figure 4.1.1.3: Comparison of source and separated signals created with mixing matrix one.

The separation matrix through the EASI ICA convergence and the resulting errors are shown in Figure 4.1.1.4 and Figure 4.1.1.5 respectively.

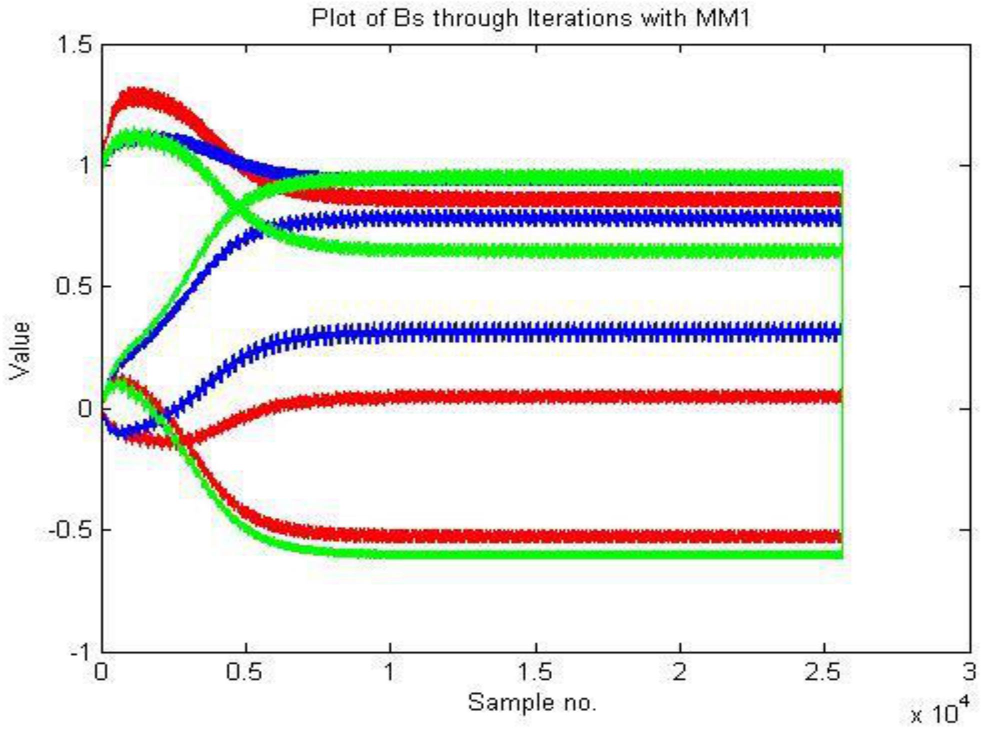

Figure 4.1.1.4: Convergence of separation matrix using mixing matrix one.

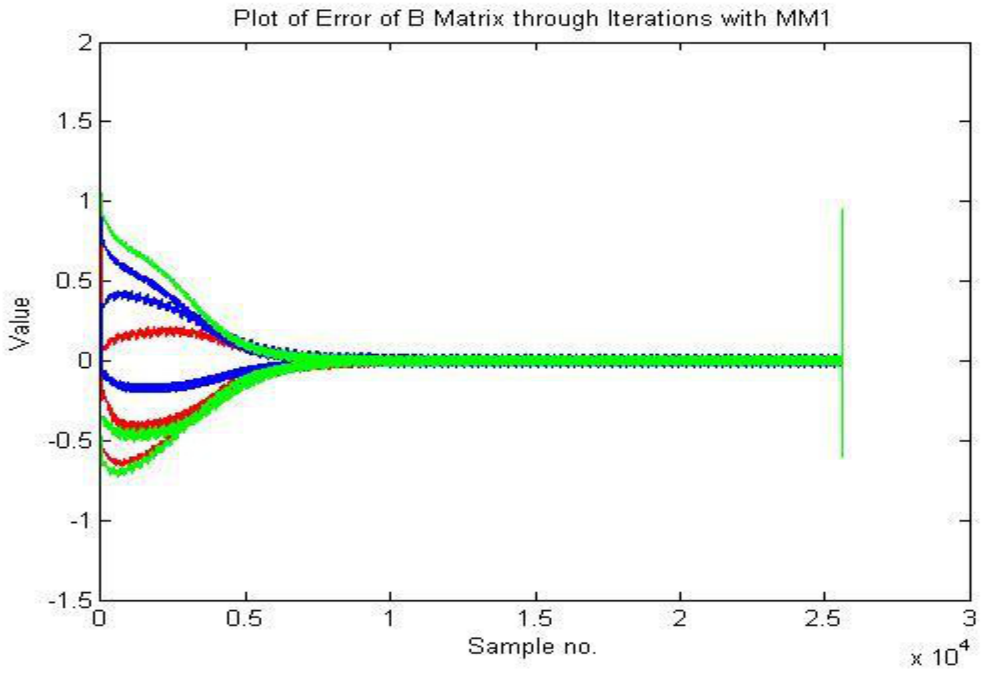

Figure 4.1.1.5: Errors of separation matrix using mixing matrix one.

In Table 4.1.1.1, the maximum FFT spectrum energies were detected at approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum for signal one created a FFT spectrum error of 0.0049% . Signal two and signal three experienced FFT spectrum errors of 2.0555%and 2.0624% respectively. The product of the mixing matrix and separation matrix had an error of 1.199%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.0823%.

Table 4.1.1.1: EASI ICA data for analysis for mixing matrix one

Table 4.1.1.1:EASI ICA Analysis Data for Mixing Matrix One	
Performance Aspect	Quantitative Values
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (0.0049%)
Signal 2 – Max Freq (%Error)	2 KHz (2.0555%)
Signal 3 – Max Freq (%Error)	10 kHz (2.0624%)
Mixing Separation Product Error	1.199%
Mixing Separation Difference Error	0.0823%

4.1.2 Mixing Matrix Two Data

Mixing matrix two is given by the following values:

$$A = \begin{pmatrix} 1 & 0.75 & 0.5 \\ 0.3 & 7/8 & -1 \\ 15/16 & -0.3 & 1 \end{pmatrix} \boldsymbol{A^{-1}} = \begin{pmatrix} -0.711 & 1.114 & 1.469 \\ 1.531 & -0.657 & -1.423 \\ 1.126 & -1.241 & -0.804 \end{pmatrix}$$

The values were chosen to represent a matrix with values easy and difficult to design in hardware. The values were used in an earlier implementation of this algorithm.
The mixed signals created with the mixing matrix are shown in Figure 4.1.2.1 .

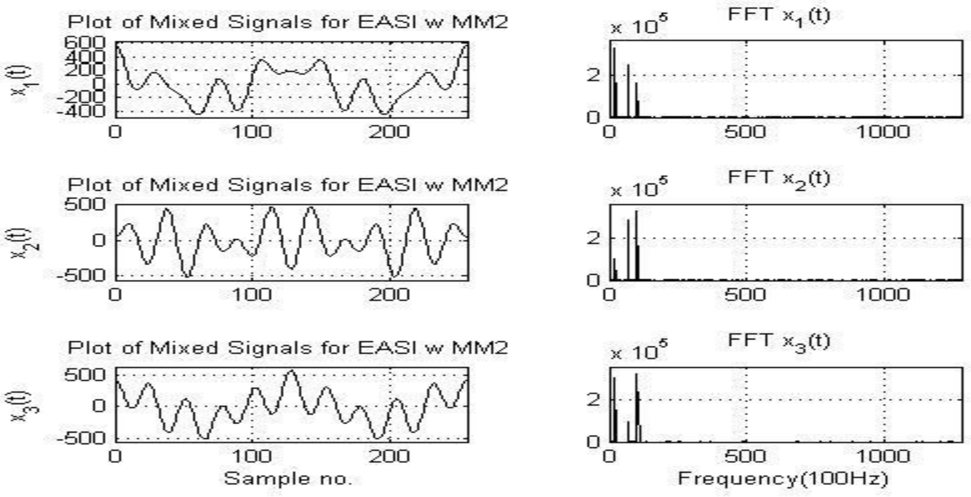

Figure 4.1.2.1: Mixed signals created from mixing matrix two.

The separation of the mixed signals is shown in Figure 4.1.2.2. The visual comparison of the source and separated signals are shown in Figure 4.1.2.3.

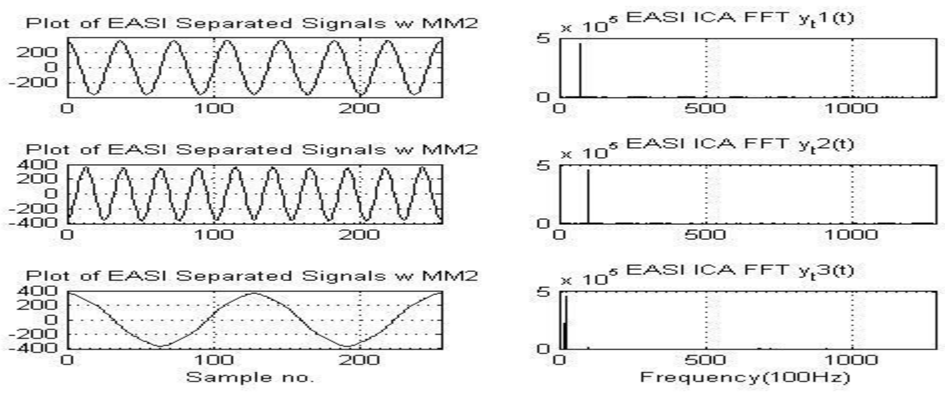

Figure 4.1.2.2: Separated signals created with mixing matrix two.

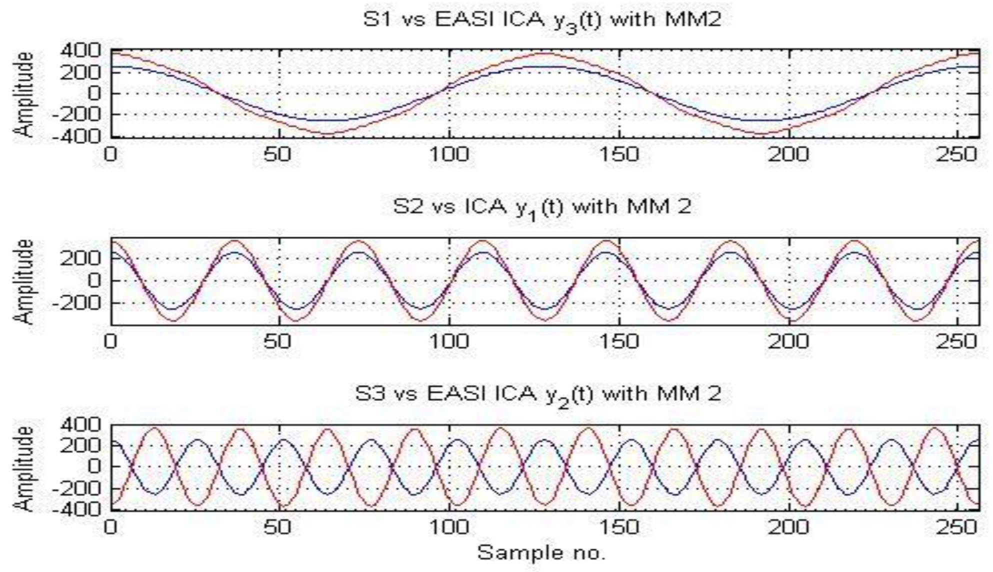

Figure 4.1.2.3: Comparison of source and separation signals created with mixing matrix two.

The separation matrix through the EASI ICA convergence and the resulting errors are shown in Figure 4.1.2.4 and Figure 4.1.2.5 respectively.

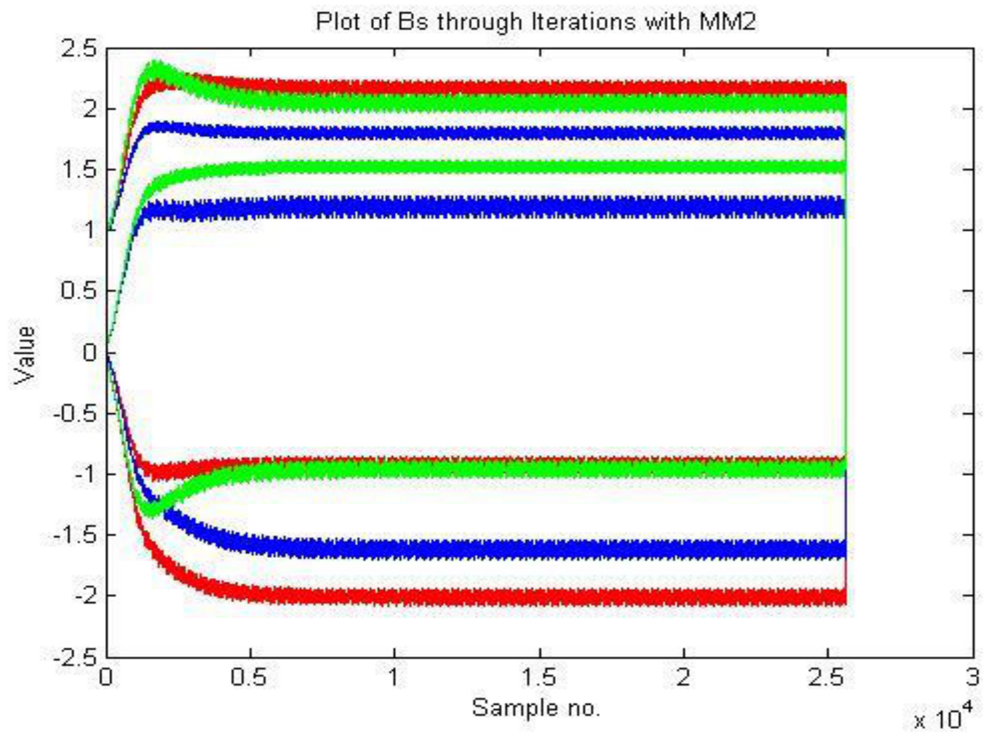

Figure 4.1.2.4: Convergence of separation matrix using mixing matrix two.

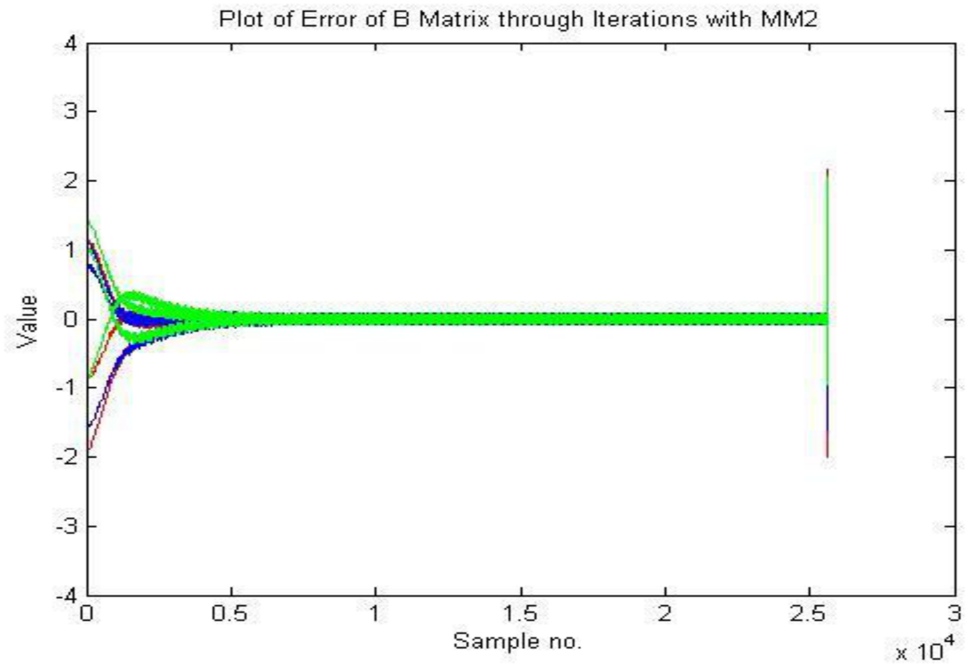

Figure 4.1.2.5: Errors of separation matrix using mixing matrix two.

In Table 4.1.2.1, the maximum FFT energies were detected at approximately 7 kHz for signal one, 10 kHz for signal two and 2 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created an FFT spectrum error of 0.0049%. Signal two and signal three experienced FFT spectrum errors of 2.0554% and 2.0634% respectively. The product of the mixing matrix and separation matrix had an error of 2.889%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 2.359%.

Table 4.1.2.1 EASI ICA analysis data for mixing matrix two.

Table 4.1.2.1: EASI ICA Analysis Data for Mixing Matrix Two	
Performance Aspect	Quantitative Values
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (0.0049%)
Signal 2 – Max Freq (%Error)	10 kHz (2.0554%)
Signal 3 – Max Freq (%Error)	2 kHz (2.0634%)
Mixing Separation Product Error	2.889%
Mixing Separation Difference Error	2.359%

4.1.3 Mixing Matrix Three Data

Mixing matrix three is given by the following values:

$$A = \begin{pmatrix} 0.8 & 0.1 & 0.6 \\ 0.3 & 0.5 & -0.7 \\ 0.4 & -.9 & 0.2 \end{pmatrix} \boldsymbol{A^{-1}} = \begin{pmatrix} -4.907 & -5.185 & -3.426 \\ -3.148 & -3.704 & -3.519 \\ -4.352 & -6.293 & -3.9815 \end{pmatrix}$$

The values are from a mixing matrix used to implement another variation of EASI ICA in. The values are that were randomly generated.

The mixed signals created with the mixing matrix are shown in Figure 4.1.3.1.

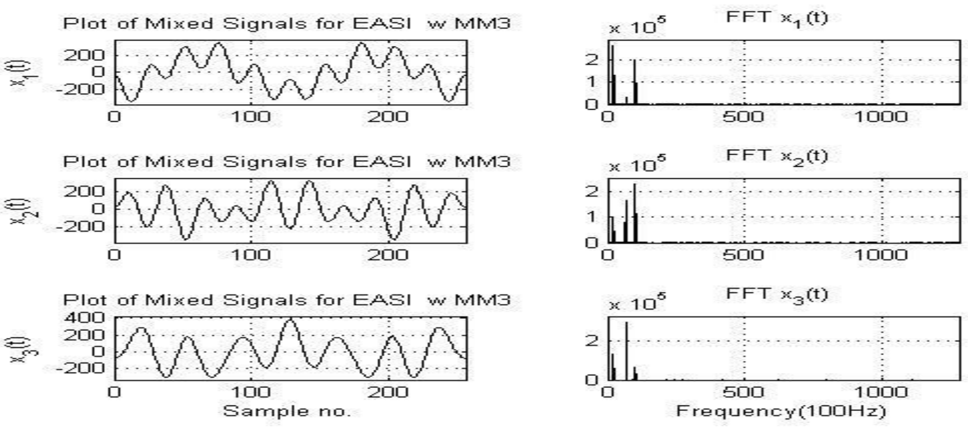

Figure 4.1.3.1: Mixed signals created from mixing matrix three.

The separation of the mixed signals is shown in Figure 4.1.3.2. The visual comparison of the source and separated signals are shown in Figure 4.1.3.3.

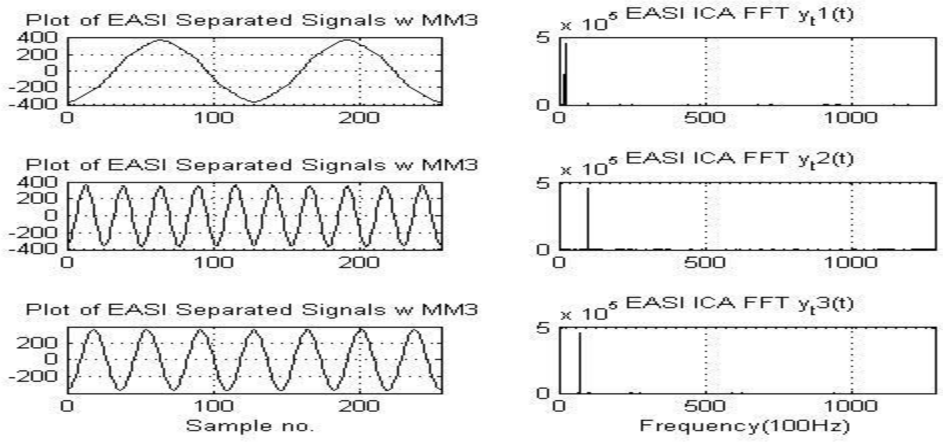

Figure 4.1.3.2: Separated signals created with mixing matrix three.

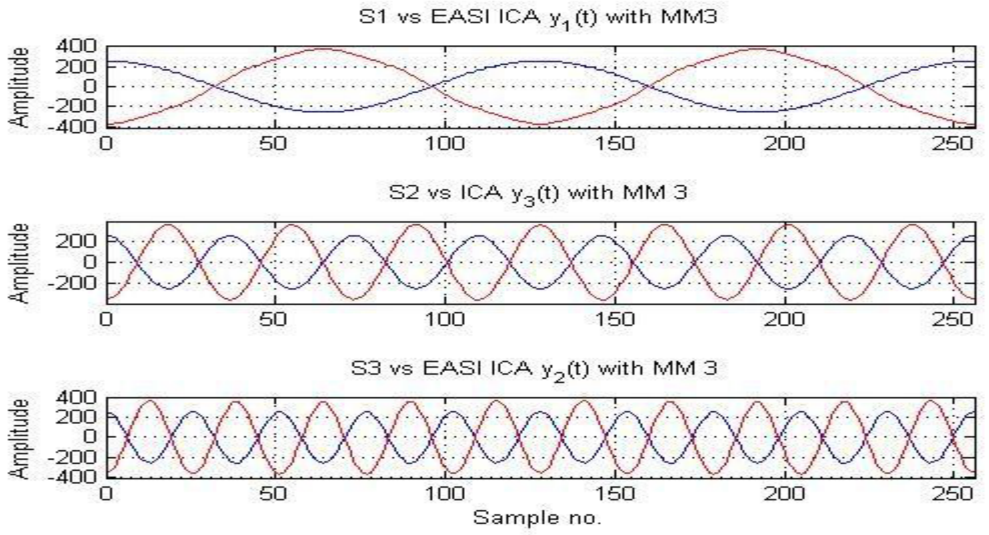

Figure 4.1.3.3: Comparison of source and separation signals created mixing matrix three.

The separation matrix EASI ICA convergence and the resulting errors are shown in Figure 4.1.3.4 and Figure 4.1.3.5 respectively.

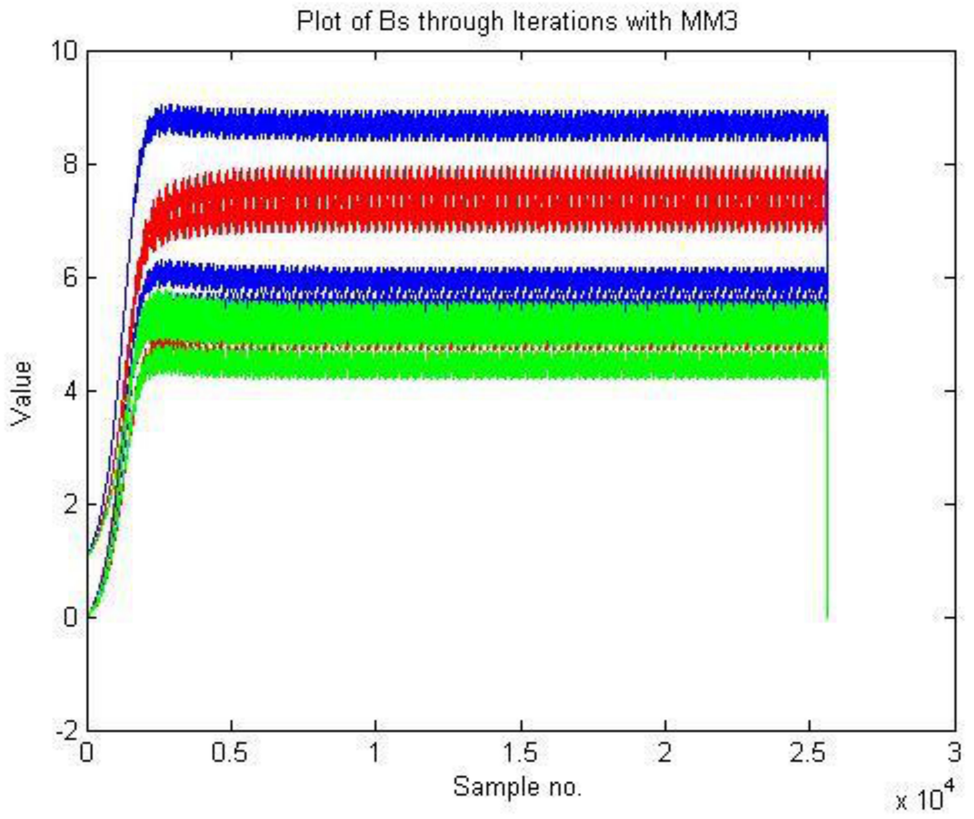

Figure 4.1.3.4: Convergence of separation matrix using mixing matrix three.

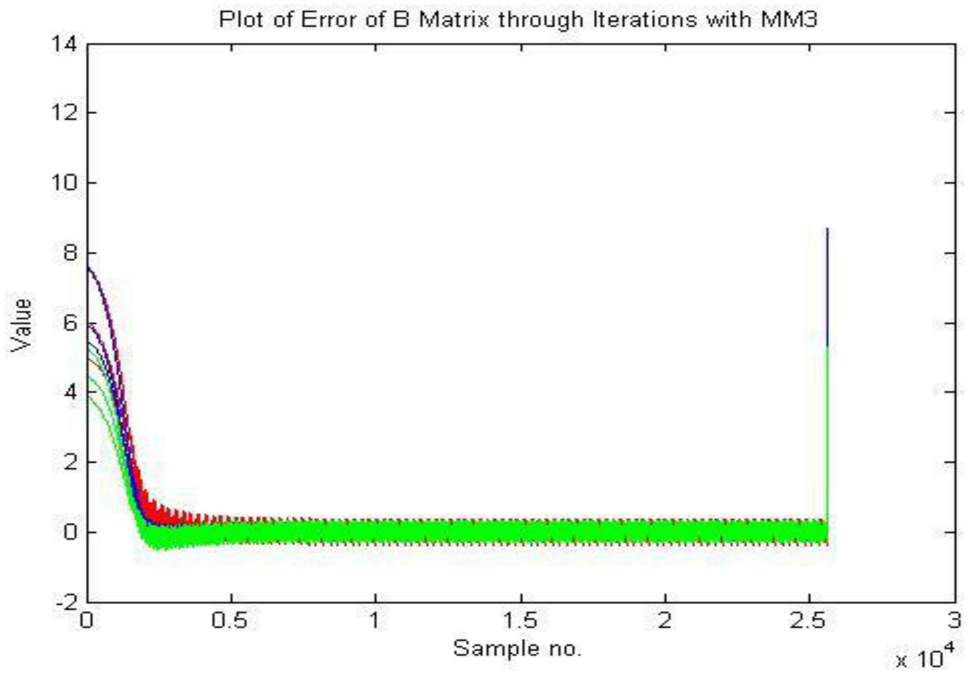

Figure 4.1.3.5: Errors of separation matrix using mixing matrix three.

In Table 4.1.3.1, the maximum energies were detected around approximately 2 KHz for signal one, 10 kHz for signal two and 7 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created a FFT spectrum error of 0.0623% . Signal two and signal three experienced FFT spectrum errors of 2.0550% and 2.005% respectively. The product of the mixing matrix and separation matrix had an error of 3.839%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 26.401%.

Table 4.1.3.1EASI ICA analysis data for mixing matrix three

Table 4.1.3.1: EASI ICA Analysis Data for Mixing Matrix Three	
Performance Aspect	Quantitative Values
FFT Measurements	
Signal 1 – Max Freq (%Error)	2 kHz (0.0623%)
Signal 2 – Max Freq (%Error)	10 kHz (2.055%)
Signal 3 – Max Freq (%Error)	7 kHz (2.005%)
Mixing Separation Product Error	3.839%
Mixing Separation Difference Error	26.401%

4.1.4 Mixing Matrix Four Data

Mixing matrix four is given by the following values:

$$A = \begin{pmatrix} -0.75 & 0.25 & 0.5 \\ 0.5 & 0.75 & -0.25 \\ 0.25 & -0.5 & 0.75 \end{pmatrix} \boldsymbol{A^{-1}} = \begin{pmatrix} 0.203 & 0.678 & 0.542 \\ 0.610 & 0.0340 & -0.373 \\ 0.678 & -0.407 & 0.475 \end{pmatrix}$$

The values are varied values of mixing matrix three. The values were modified to be values that are easier to implement on hardware. The mixed signals created with the mixing matrix are shown in Figure 4.1.4.1.

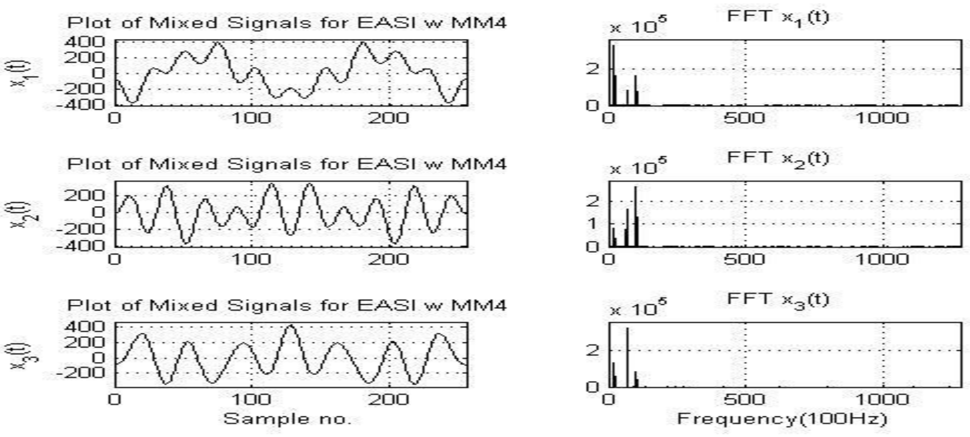

Figure 4.1.4.1: Mixed signals created from mixing matrix four.

The separation of the mixed signals is shown in Figure 4.1.4.2. The visual comparison of the source and separated signals are shown in Figure 4.1.4.3

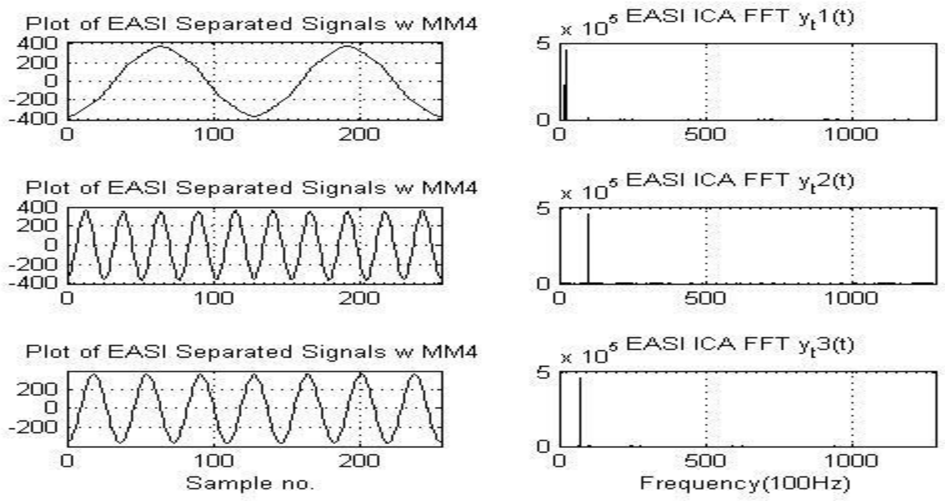

Figure 4.1.4.2: Separated signals created with mixing matrix four.

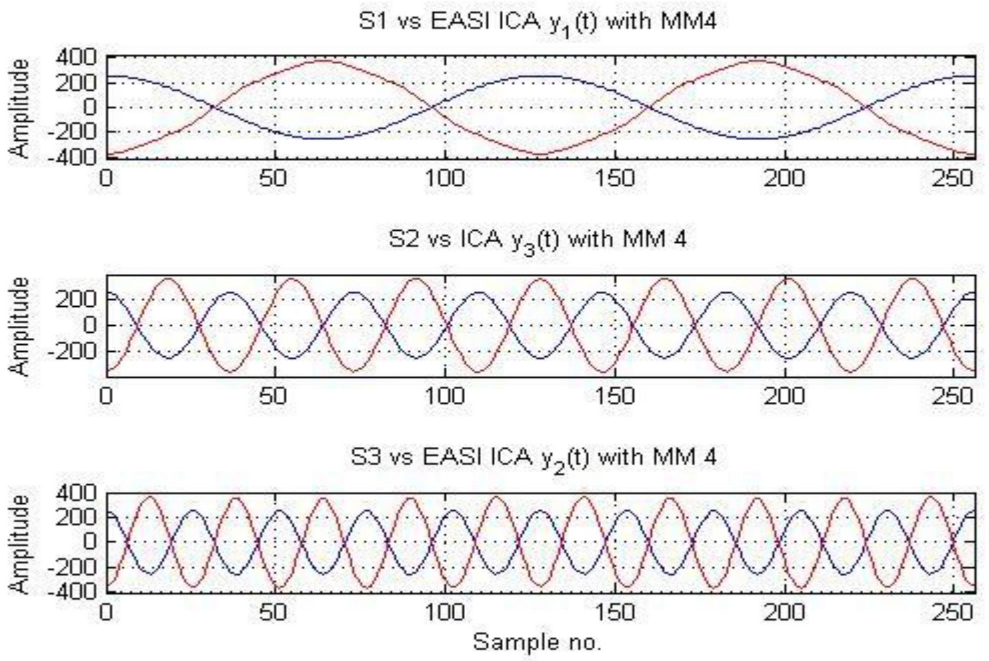

Figure 4.1.4.3: Comparison of source and separation signals created with mixing matrix four.

The separation matrix evolution through the EASI ICA convergence and the resulting errors are shown in Figure 4.1.4.4 and Figure 4.1.4.5 respectively.

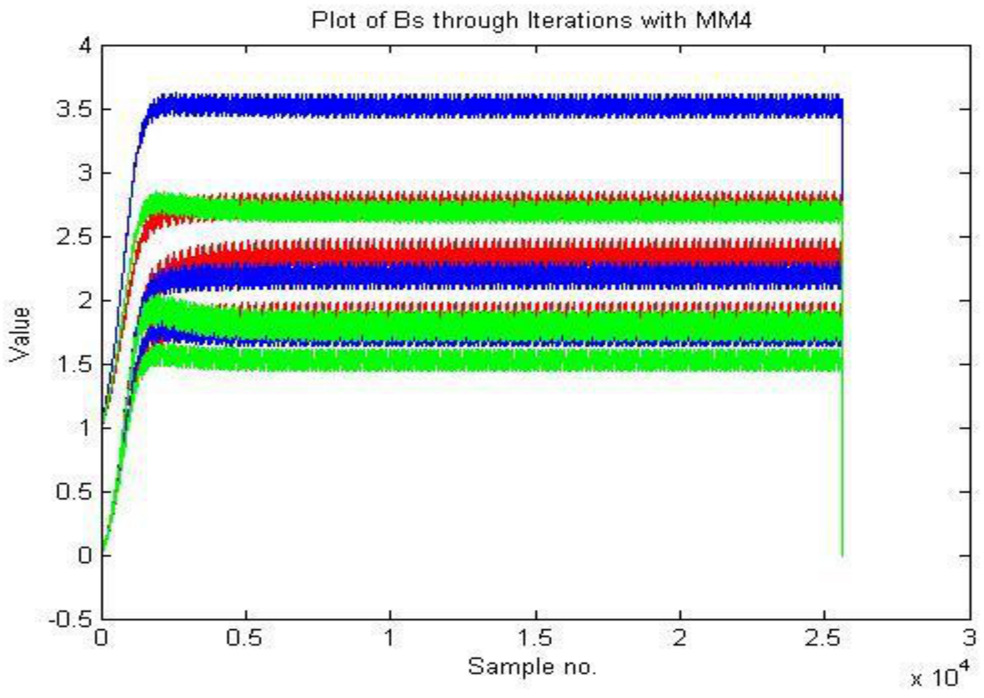

Figure 4.1.4.4: Convergence of separation matrix using mixing matrix four.

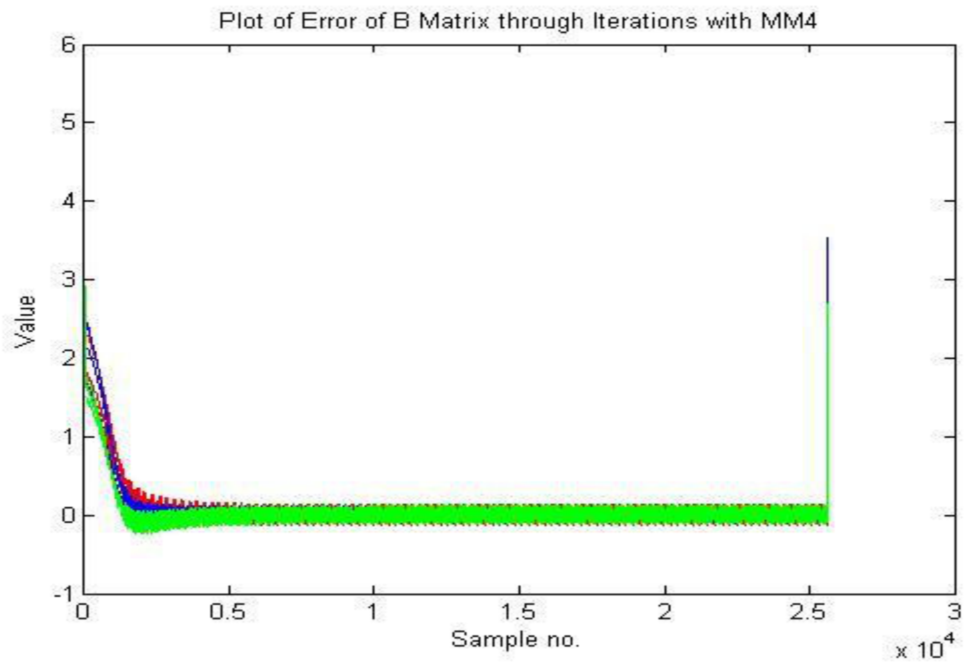

Figure 4.1.4.5: Errors of separation matrix using mixing matrix four.

In Table 4.1.4.1, the maximum energies were detected around approximately 2 KHz for signal one, 10 kHz for signal two and 7 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created an FFT spectrum error of 0.0623% . Signal two and signal three experienced FFT spectrum errors of 2.0557% and 2.0050% respectively. The product of the mixing matrix and separation matrix had an error of 2.792%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 10.244%.

Table 4.1.4.1EASI ICA analysis data for mixing matrix four.

Table 4.1.4.1: EASI ICA Analysis Data for Mixing Matrix Four	
Performance Aspect	Quantitative Values
FFT Measurements	
Signal 1 – Max Freq (%Error)	2 KHz (0.0623%)
Signal 2 – Max Freq (%Error)	10 kHz (2.0557%)
Signal 3 – Max Freq (%Error)	7 kHz (2.005%)
Mixing Separation Product Error	2.792%
Mixing Separation Difference Error	10.244%

4.1.5 Mixing Matrix Data Comparison and Selection

In Table 4.1.5.1 compares data and errors for each mixing matrix. All of the mixing matrices had the similar FFT spectrum errors. Mixing matrix one has the smallest product error while mixing matrix three had the worse. The worst difference error belonged to mixing matrix three while the best belongs to mixing matrix. The best mixing matrix overall was mixing matrix one, and is the easiest to implement in hardware. The matrix remained the matrix for EASI implementation.

Table 4.1.5.1 Comparison of EASI ICA data for analysis for all mixing matrices

Table 4.1.5.1: Comparison of EASI ICA Analysis Data for Each Mixing Matrices				
Performance Aspect	Mixing Matrix Quantitative Values			
FFT Measurements	MM 1	MM 2	MM3	MM4
Signal 1 – Max Freq (%Error)	7 kHz (0.0049%)	7 kHz (0.0049%)	2 KHz (0.0623%)	2 KHz (0.0623%)
Signal 2 – Max Freq (%Error)	2 KHz (2.0555%)	10 kHz (2.0554%)	10 kHz (2.055%)	10 kHz (2.0557%)
Signal 3 – Max Freq (%Error)	10 kHz (2.0624%)	2 KHz (2.0634%)	7 kHz (2.005%)	7 kHz (2.005%)
Mixing Separation Product Error	1.199%	2.889%	3.839%	2.792%
Mixing Separation Difference Error	0.0823%	2.359%	26.401%	10.244%

4.2 Nonlinearity Testing

4.2.1 Nonlinearities

The nonlinearity portion of the EASI algorithm is important because it impacts the ability of the algorithm to separate the mixed signals. The tanh(y) nonlinearity was utilized because it has been noted as the best nonlinearity. The tanh(y) nonlinearity and seven other nonlinearities were tested with the EASI algorithm.

The eight nonlinearities are shown in the following plots:

1. Signum Function as showing Figure 4.2.1.1

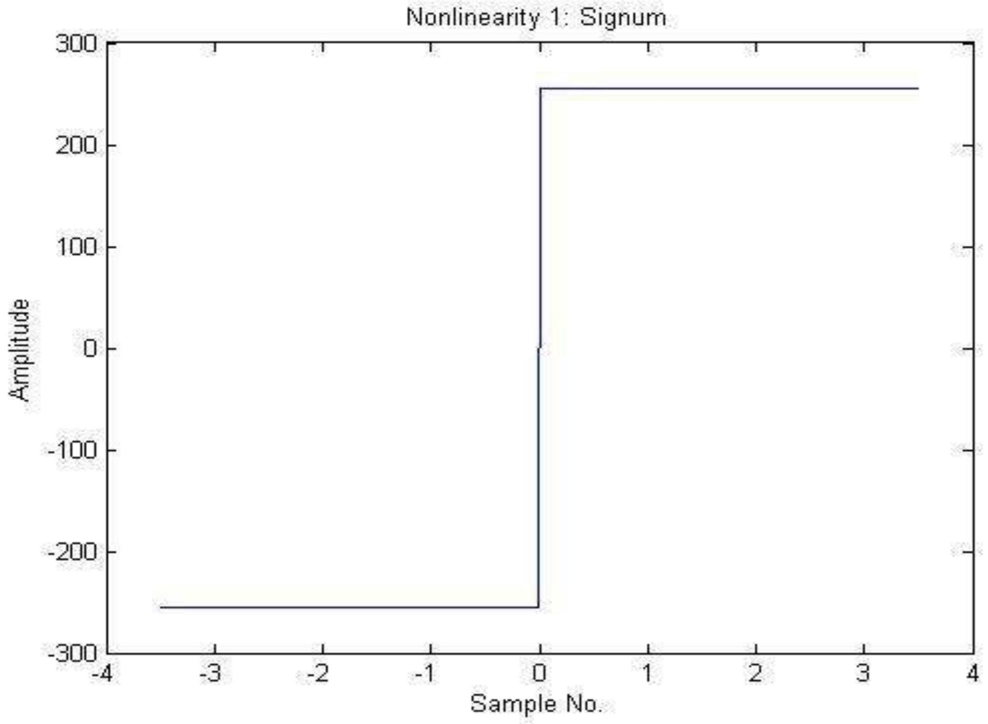

Figure 4.2.1.1: Nonlinearity one: Signum.

2. Function of Y (F(y) = y) as shown in Figure 4.2.1.2

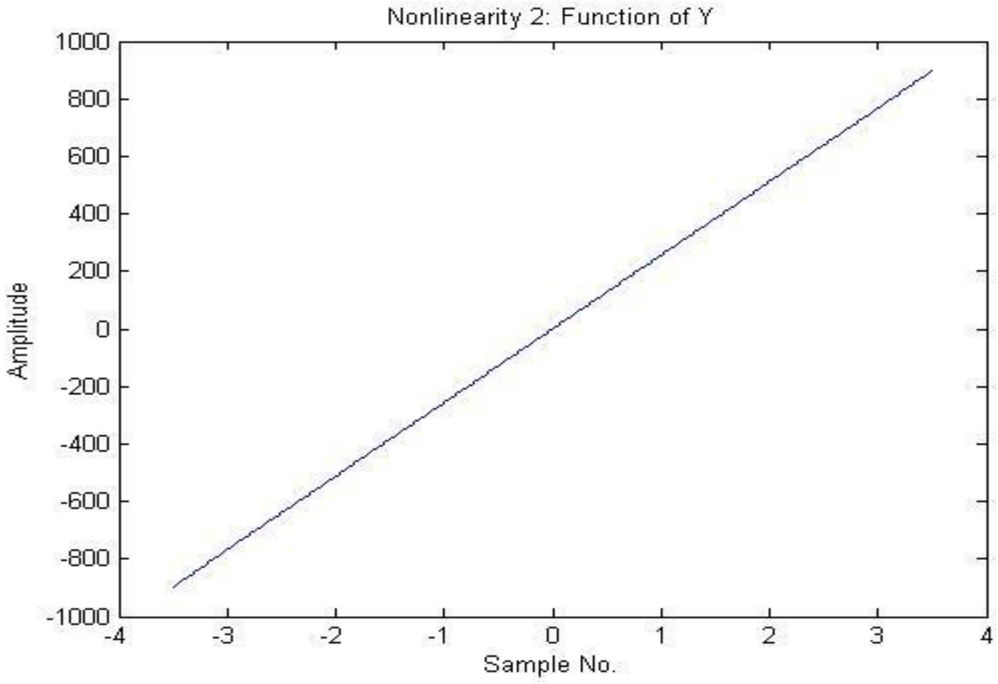

Figure 4.2.1.2: Nonlinearity two: Function of y.

3. Cubic Function ($F(y) = y^3$) as shown in Figure 4.2.1.3

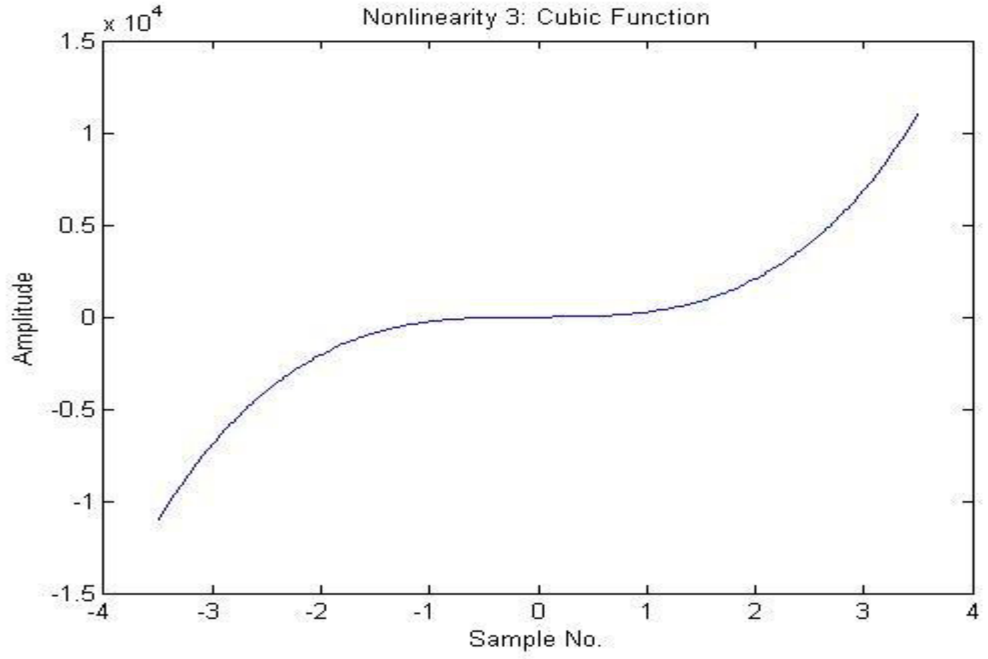

Figure 4.2.1.3: Nonlinearity three: Cubic function

4. Piecewise function as shown in Figure 4.2.1.4

$$f(y) = \begin{cases} -1 & y < -1 \\ y & -1 \le y \le 1 \\ 1 & y > 1 \end{cases}$$

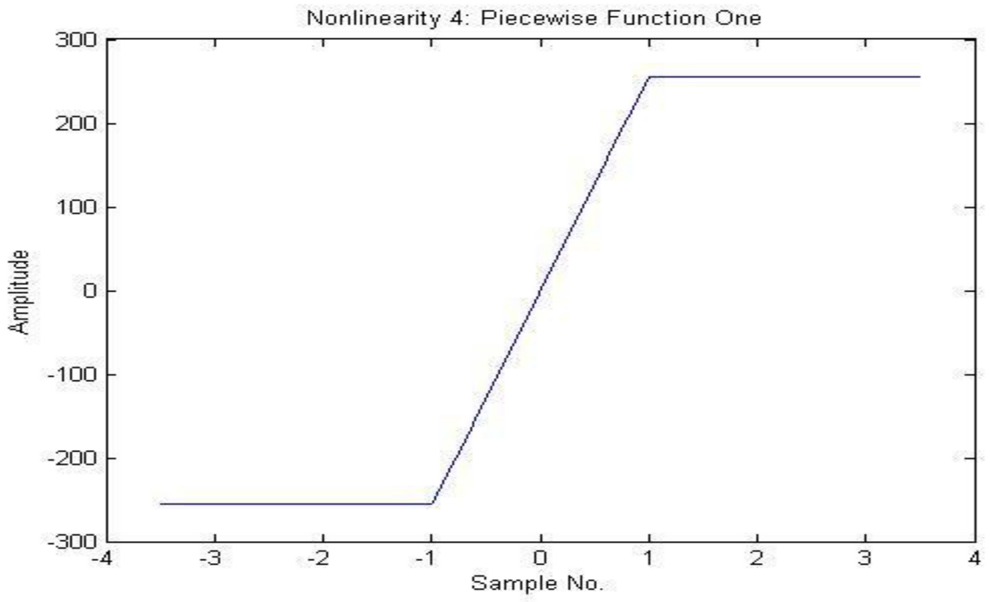

Figure 4.2.1.4: Nonlinearity four: Piecewise function one

5. Inverse Tangent Function as shown in Figure 4.2.1.5

$$f(y) = \tan^{-1}(y) = abs\left(\left(\frac{\frac{\log(1+y)}{\log(1-y))}}{2}\right)\right)$$

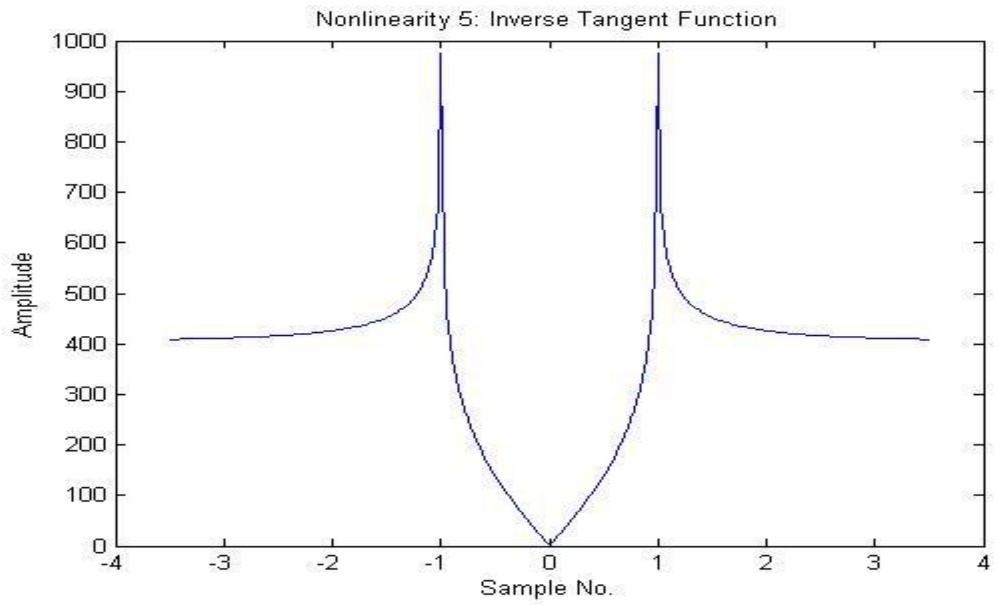

Figure 4.2.1.5: Nonlinearity five: Inverse tangent function.

6. Hyperbolic Tangent Function as shown in Figure 4.2.1.6.

$$f(y) = \tanh(y) = \frac{(e^{2y}-1)}{(e^{2y}+1)}$$

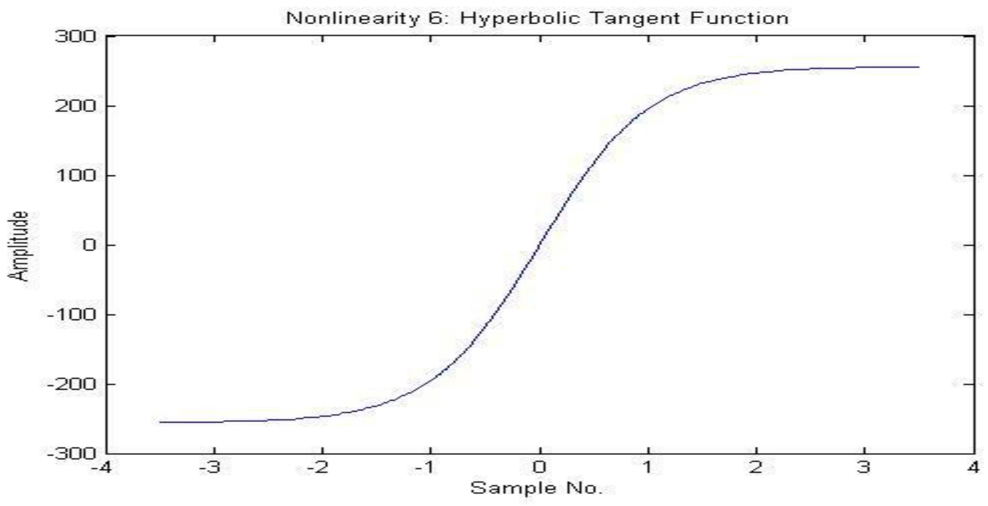

Figure 4.2.1.6: Nonlinearity six: Hyperbolic tangent function.

7. Exponential Function as shown in Figure 4.2.1.7

$$f(y) = \frac{2}{(1 + e^{-y})} - 1$$

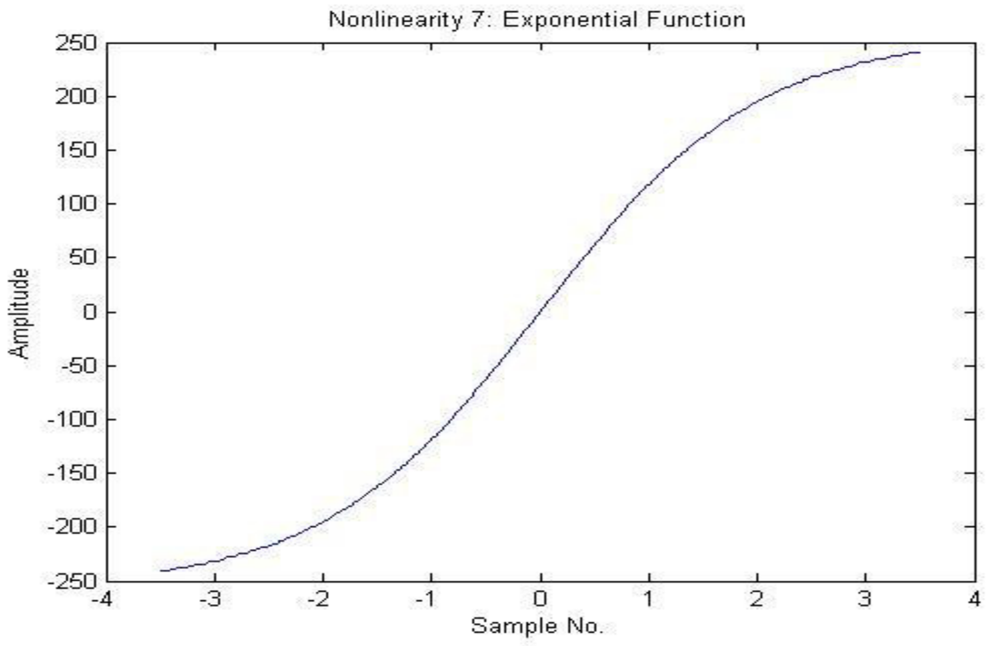

Figure 4.2.1.7: Nonlinearity seven: Exponential function.

8. Difference between Function of Y and Hyperbolic Tangent Function as shown in Figure 4.2.1.8.

$$f(y) = y - \tanh(y)$$

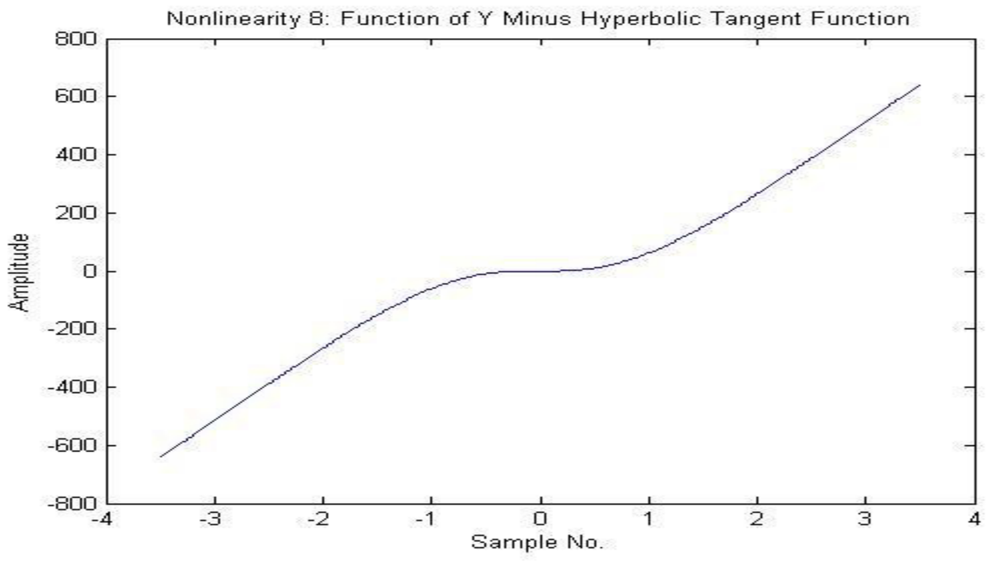

Figure 4.2.1.8: Nonlinearity eight: Difference between Function of y and hyperbolic tangent.

9. Piecewise Function as shown in Figure 4.2.1.9.

$$f(y) = \begin{cases} y+1 & y < -1 \\ 0 & -1 \leq y \leq 1 \\ y-1 & y > 1 \end{cases}$$

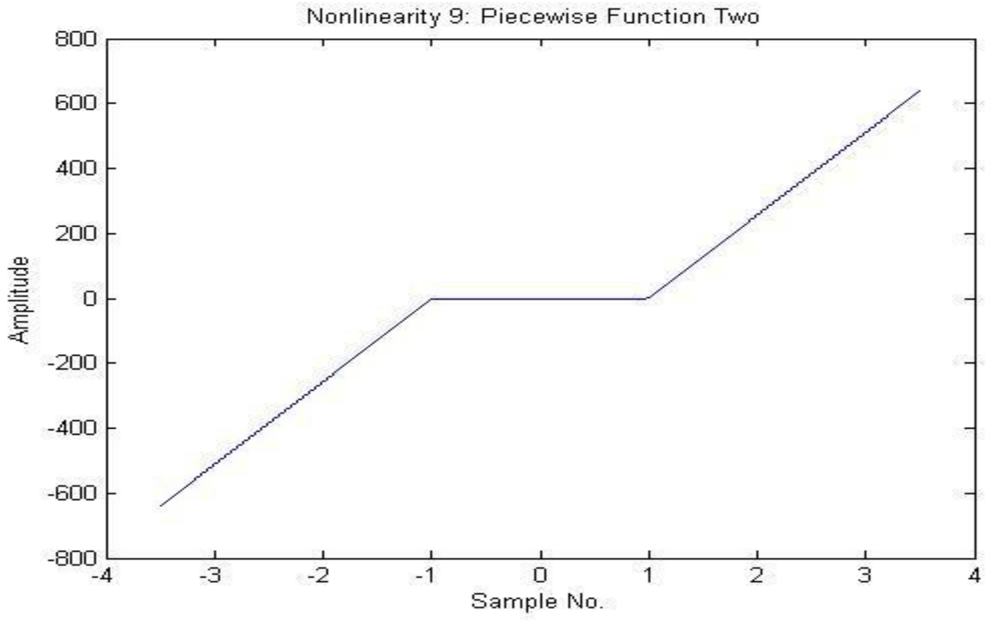

Figure 4.2.1.9: Nonlinearity nine: Piecewise function two.

4.2.2 Nonlinearity Data Analysis

The nine nonlinearities were implemented with the EASI algorithm. Each of the nonlinearities for analyzed for the following:

- Convergence speed
- Visible separation
- Separated signal FFT spectrum error
- Mixing matrix and separated matrix product error
  (Mixing Matrix * Separation Matrix = Identity Matrix)
- Separated matrix and mixing matrix difference error
  (Separation Matrix – Inverse of Mixing = 0)
- Number of resources required for implementation

4.2.2.1 Nonlinearity One: Signum

The EASI ICA algorithm was implemented using nonlinearity one to create the following separated signals seen in Figure 4.2.2.1.1

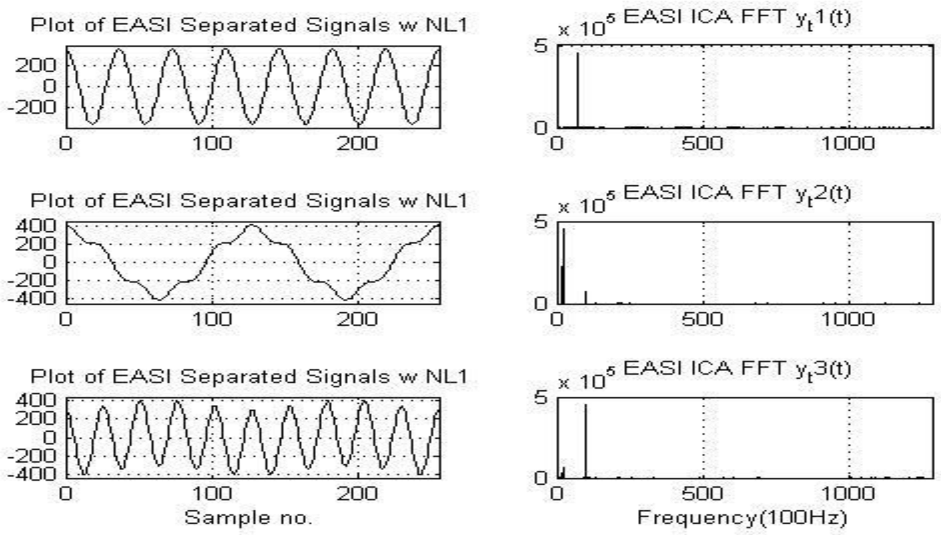

Figure 4.2.2.1.1: Separated signals using nonlinearity one.

A comparison of the source and separated signals are shown in Figure 4.2.2.1.2.

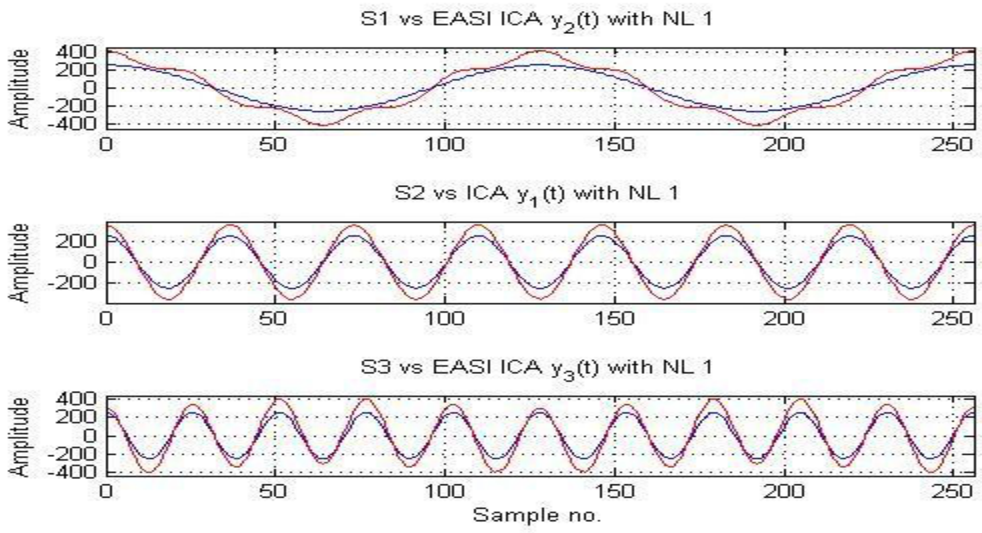

Figure 4.2.2.1.2: Comparison of source and separated signals for nonlinearity one.

In Table 4.2.2.1, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created of 0.0167% . Signal two and signal three experienced FFT spectrum errors of 2.309%and 2.304% respectively. The product of the mixing matrix and separation matrix had an error of 1.219%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.105%.

Table 4.2.2.1 Nonlinearity one data for analysis

Table 4.2.2.1 NL 1 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	Partially Yes
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (0.0167%)
Signal 2 – Max Freq (%Error)	2 kHz (2.309%)
Signal 3 – Max Freq (%Error)	10 kHz (2.303%)
Mixing Separation Product Error	1.218%
Mixing Separation Difference Error	0.105%

4.2.2.2 Nonlinearity Two: Function of Y

The EASI ICA algorithm was implemented using nonlinearity two to create the following separated signals seen in Figure 4.2.2.2.1.

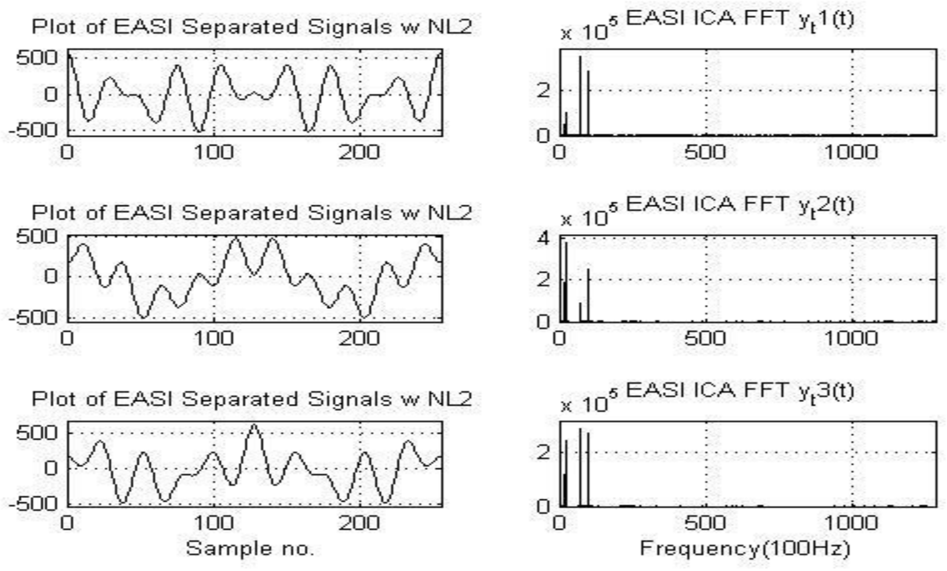

Figure 4.2.2.2.1: Separated signals using nonlinearity two.

A comparison of the source and separated signals are shown in Figure 4.2.2.2.2.

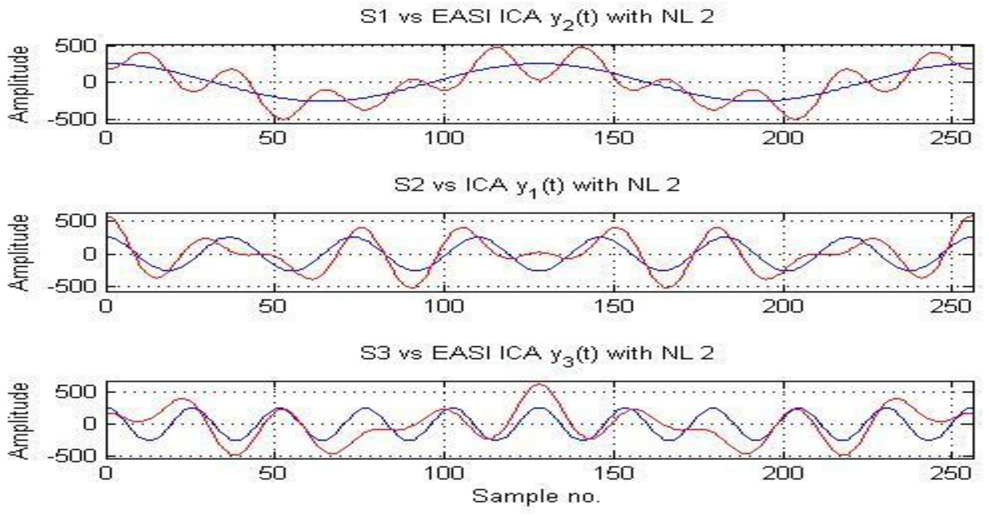

Figure 4.2.2.2.2: Comparison of source and separated signals for nonlinearity two.

In Table 4.2.2.2, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one, two and three created a FFT spectrum error of 22.03%, 37.940% and 55.720% respectively. The product of the mixing matrix and separation matrix had an error of 5.595%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 12.36%.

Table 4.2.2.2 Nonlinearity two data for analysis

Table 4.2.2.2 NL 2 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	No
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (22.030%)
Signal 2 – Max Freq (%Error)	2 kHz (37.940%)
Signal 3 – Max Freq (%Error)	10 kHz (55.720%)
Mixing Separation Product Error	5.595%
Mixing Separation Difference Error	12.36%

4.2.2.3 Nonlinearity Three: Cubic function

The EASI ICA algorithm was implemented using nonlinearity three to create the following separated signals seen in Figure 4.2.2.3.1.

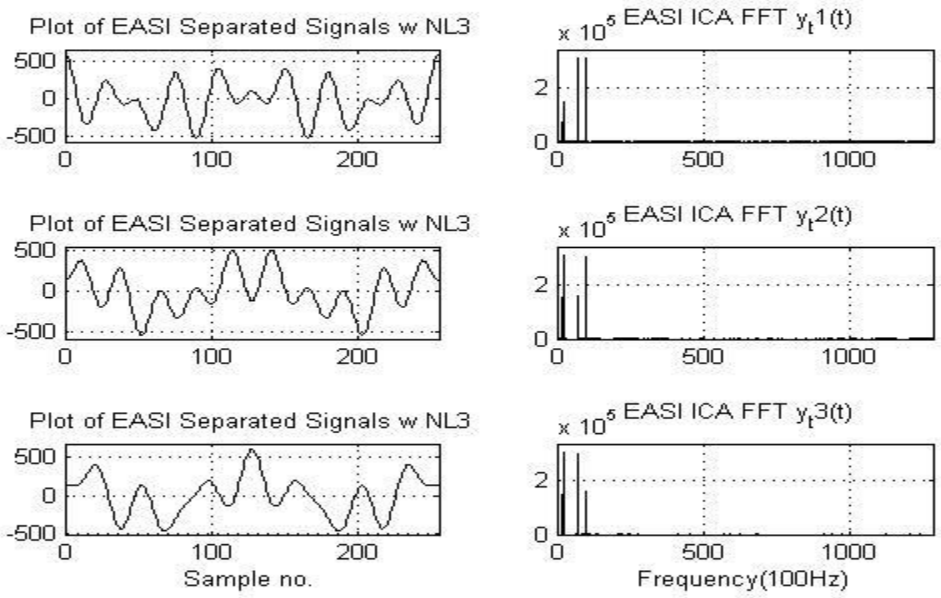

Figure 4.2.2.3.1: Separated signals using nonlinearity three.

A comparison of the source and separated signals are shown in Figure 4.2.2.3.2.

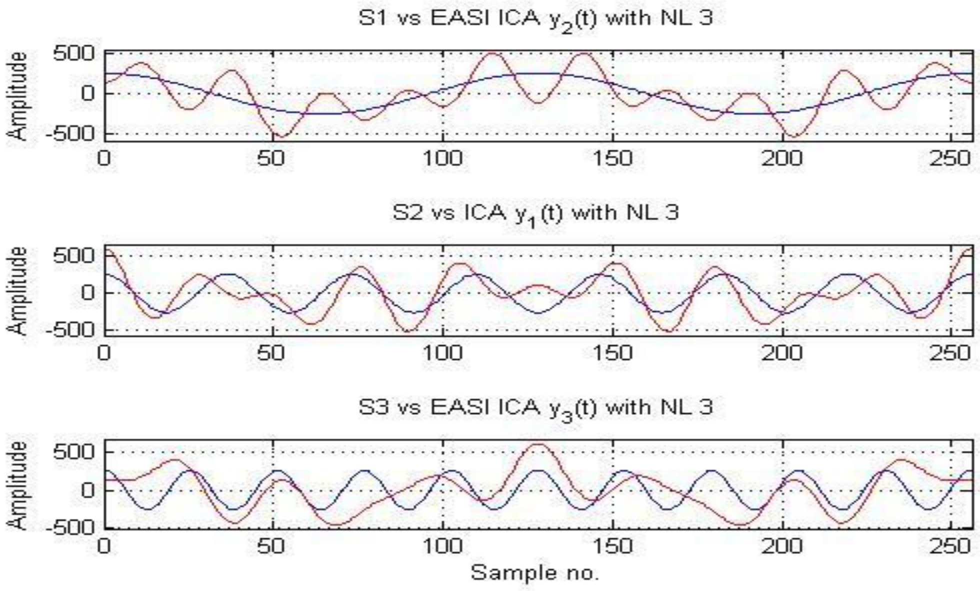

Figure 4.2.2.3.2: Comparison of source and separated signals for nonlinearity three.

In Table 4.2.2.3, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one, two and three created a FFT spectrum error of 29.390%, 49.660% and 29.390% respectively. The product of the mixing matrix and separation matrix had an error of 3.676%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 3.614%

Table 4.2.2.3 Nonlinearity three data for analysis

Table 4.2.2.3 NL 3 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	No
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (29.390%)
Signal 2 – Max Freq (%Error)	2 kHz (49.660%)
Signal 3 – Max Freq (%Error)	10 kHz (29.390%)
Mixing Separation Product Error	3.676%
Mixing Separation Difference Error	3.614%

4.2.2.4 Nonlinearity Four: Piecewise function one

The EASI ICA algorithm was implemented using nonlinearity four to create the following separated signals shown in Figure 4.2.2.4.1.

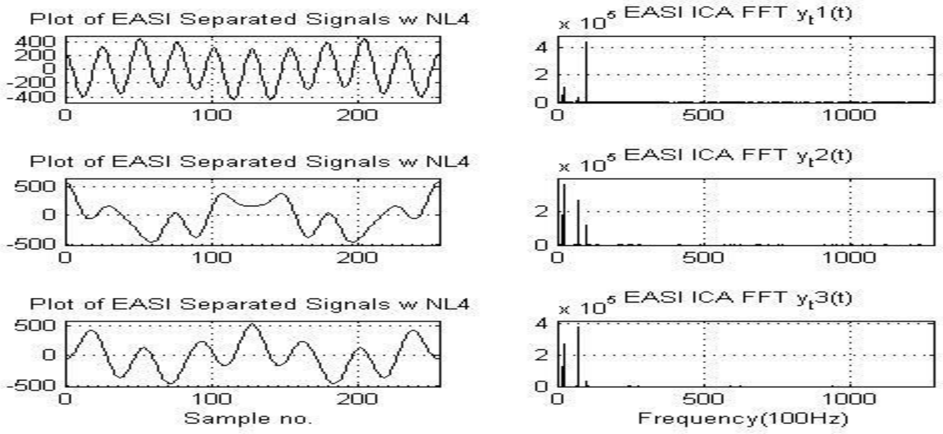

Figure 4.2.2.4.1: Separated signals using nonlinearity four.

A comparison of the source and separated signals are shown in Figure 4.2.2.4.2.

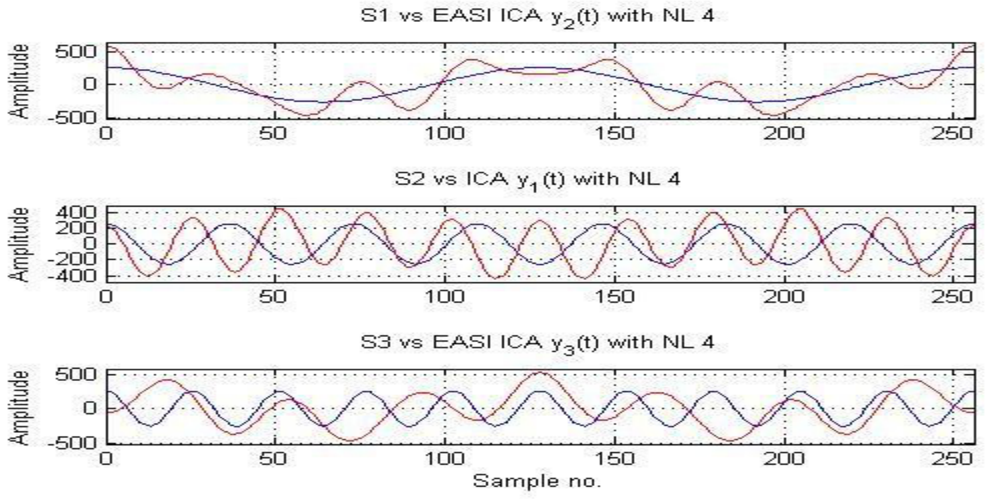

Figure 4.2.2.4.2: Comparison of source and separated signals for nonlinearity four

In Table 4.2.2.4, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created a FFT spectrum error of 6.860%. Signal two and signal three experienced FFT spectrum errors of 41.100% and 35.83% respectively. The product of the mixing matrix and separation matrix had an error of 1.836%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 2.220%.

Table 4.2.2.4 Nonlinearity four data for analysis

Table 4.2.2.4 NL 4 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	No
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (6.860%)
Signal 2 – Max Freq (%Error)	2 kHz (41.10%)
Signal 3 – Max Freq (%Error)	10 kHz (35.83%)
Mixing Separation Product Error	1.836%
Mixing Separation Difference Error	2.220%

4.2.2.5 Nonlinearity Five: Inverse tangent

The EASI ICA algorithm was implemented using nonlinearity one to create the following separated signals seen in Figure 4.2.2.5.1.

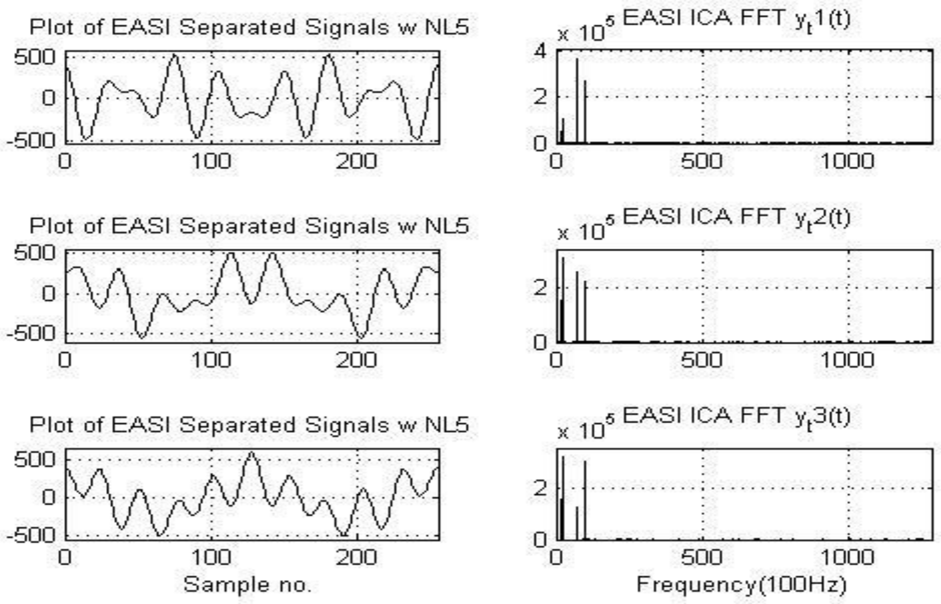

Figure 4.2.2.5.1: Separated signals using nonlinearity five.

A comparison of the source and separated signals are shown in Figure 4.2.2.5.2.

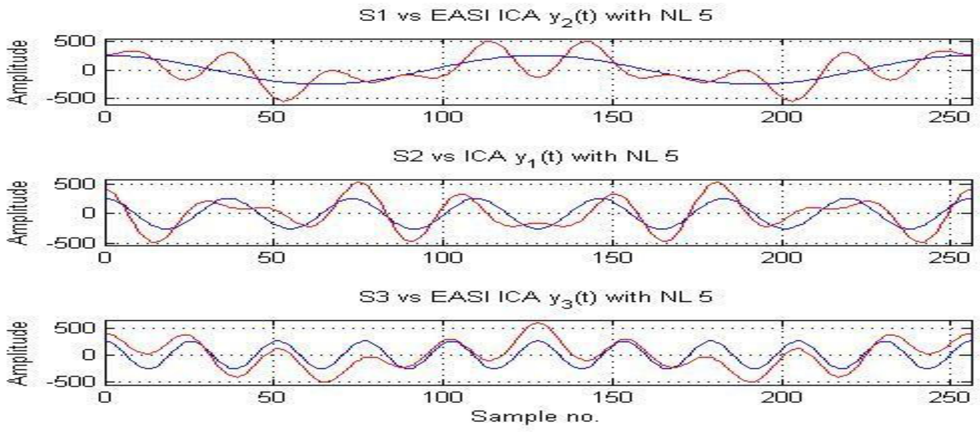

Figure 4.2.2.5.2: Comparison of source and separated signals for nonlinearity five.

In Table 4.2.2.5, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the

FFT spectrum plot for signal one created a FFT spectrum error of 0.0049% . Signal two and signal three experienced FFT spectrum errors of 2.0555%and 2.0624% respectively. The product of the mixing matrix and separation matrix had an error of 1.199%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.0823%.

Table 4.2.2.5 Nonlinearity five data for analysis

Table 4.2.2.5 NL 5 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	No
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (2.043%)
Signal 2 – Max Freq (%Error)	2 kHz (5.096%)
Signal 3 – Max Freq (%Error)	10 kHz (4.656%)
Mixing Separation Product Error	1.2801%
Mixing Separation Difference Error	0.169%

4.2.2.6 Nonlinearity Six: Hyperbolic tangent

The EASI ICA algorithm was implemented using nonlinearity six to create the following separated signals seen in Figure 4.2.2.6.1.

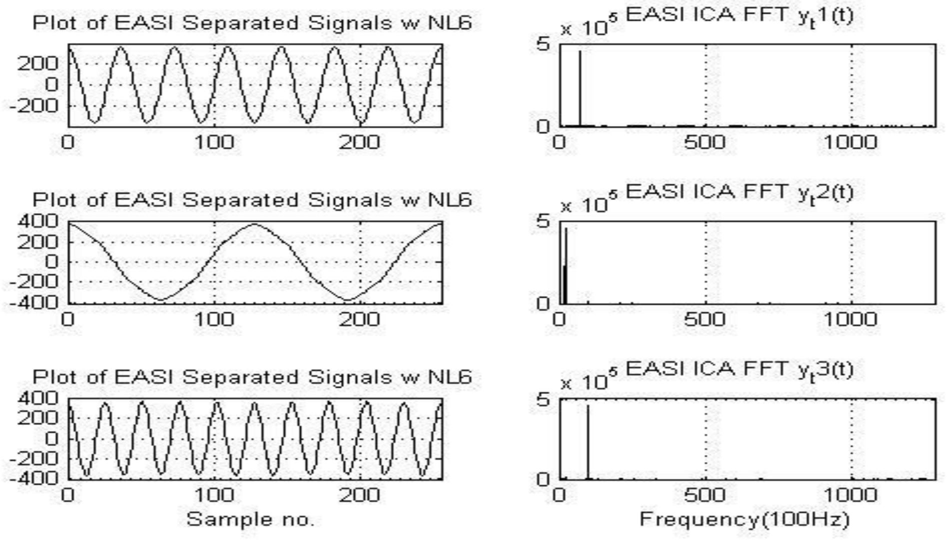

Figure 4.2.2.6.1: Separated signals using nonlinearity six.

The source and separated signals are shown in Figure 4.2.2.6.2.

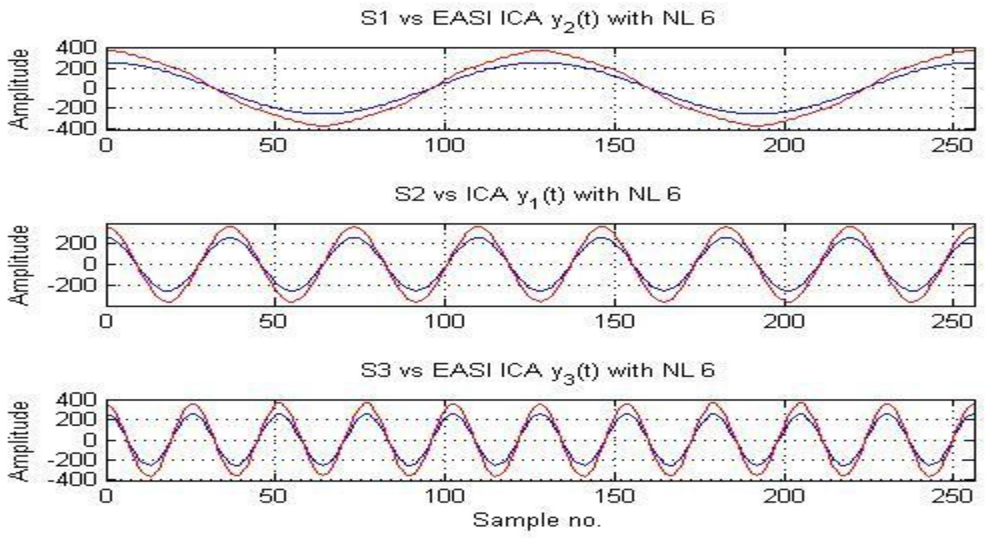

Figure 4.2.2.6.2: Comparison of source and separated signals for nonlinearity six.

In Table 4.2.2.6, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created a FFT spectrum error of 0.0049% . Signal two and signal three experienced FFT spectrum errors of 2.0555%and 2.0624% respectively. The product of the mixing matrix and separation matrix had an error of1.199%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.0823%.

Table 4.2.2.6 Nonlinearity six data for analysis

Table 4.2.2.6 NL 6 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	Yes
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (0.0049%)
Signal 2 – Max Freq (%Error)	2 kHz (2.0555%)
Signal 3 – Max Freq (%Error)	10 kHz (2.0624%)
Mixing Separation Product Error	1.199%
Mixing Separation Difference Error	0.0823%

4.2.2.7 Nonlinearity Seven: Exponential function

The EASI ICA algorithm was implemented using nonlinearity one to create the following separated signals shown in Figure 4.2.2.7.1.

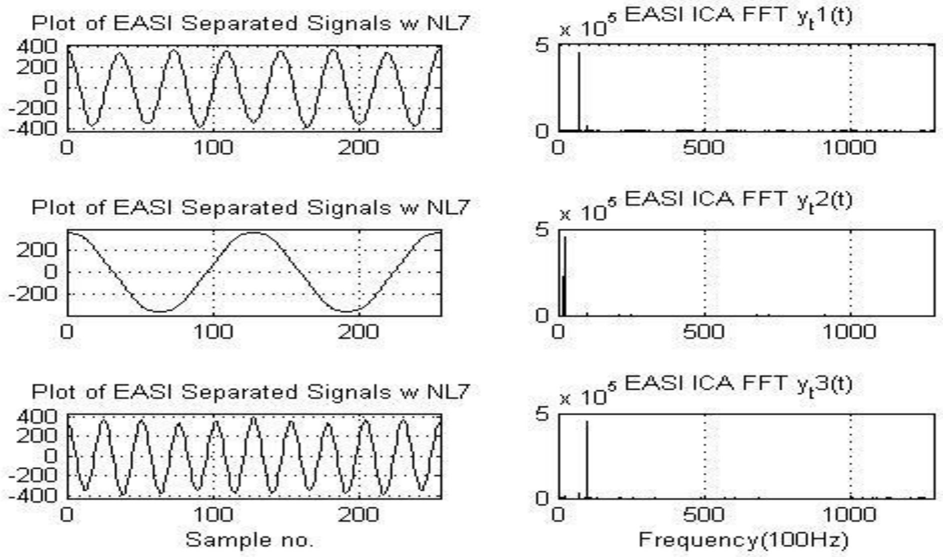

Figure 4.2.2.7.1: Separated signals using nonlinearity one.

A comparison of the source and separated signals are shown in Figure 4.2.2.7.2.

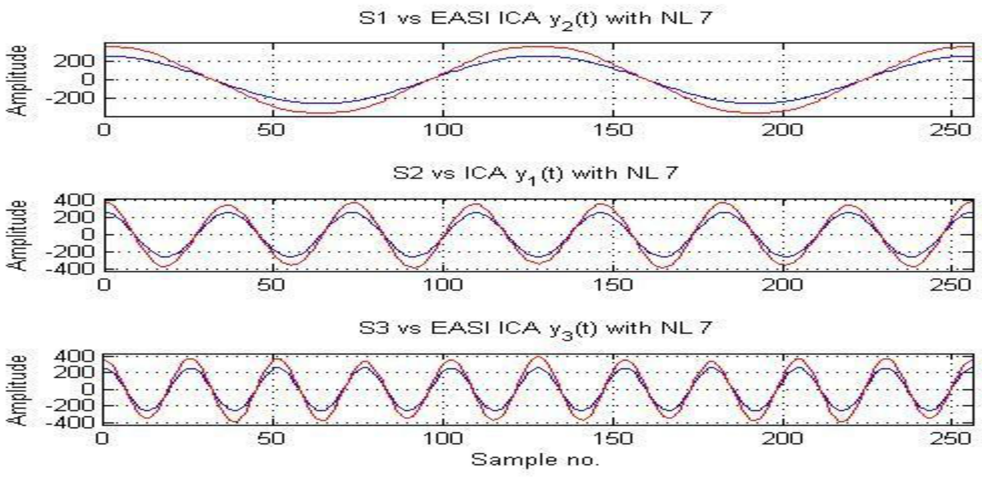

Figure 4.2.2.7.2: Comparison of source and separated signals for nonlinearity seven

In Table 4.2.2.7, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the

FFT spectrum plot for signal one created a FFT spectrum error of 0.129%. Signal two and signal three experienced FFT spectrum errors of 2.0580% and 2.175% respectively. The product of the mixing matrix and separation matrix had an error of 1.108%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.129%.

Table 4.2.2.7 Nonlinearity seven data for analysis

Table 4.2.2.7 NL 7 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	Mostly Yes
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (0.129%)
Signal 2 – Max Freq (%Error)	2 kHz (2.0580%)
Signal 3 – Max Freq (%Error)	10 kHz (2.175%)
Mixing Separation Product Error	1.108%
Mixing Separation Difference Error	0.129%

4.2.2.8 Nonlinearity Eight: Difference between Function of y and hyperbolic tangent

The EASI ICA algorithm was implemented using nonlinearity one to create the following separated signals shown in Figure 4.2.2.8.1.

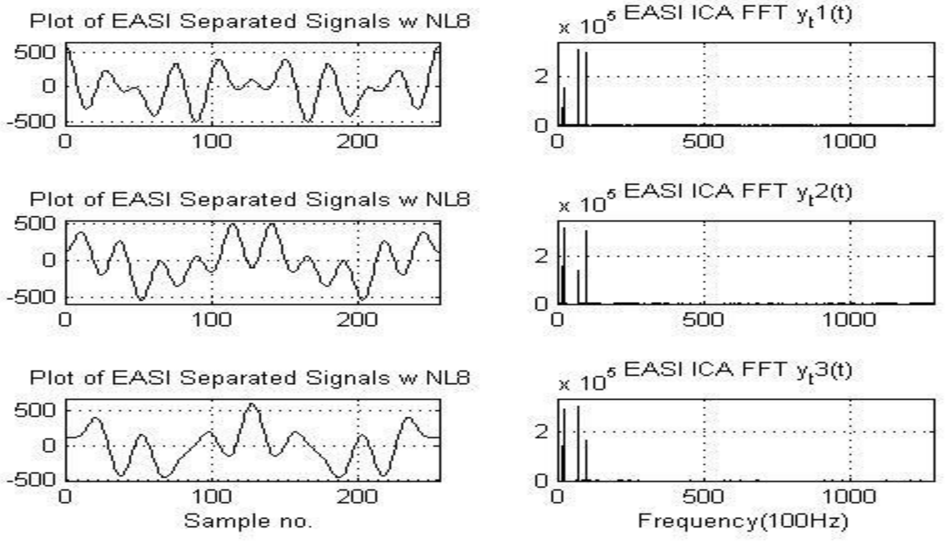

Figure 4.2.2.8.1: Separated signals using nonlinearity eight.

A comparison of the source and separated signals are shown in Figure 4.2.2.8.2.

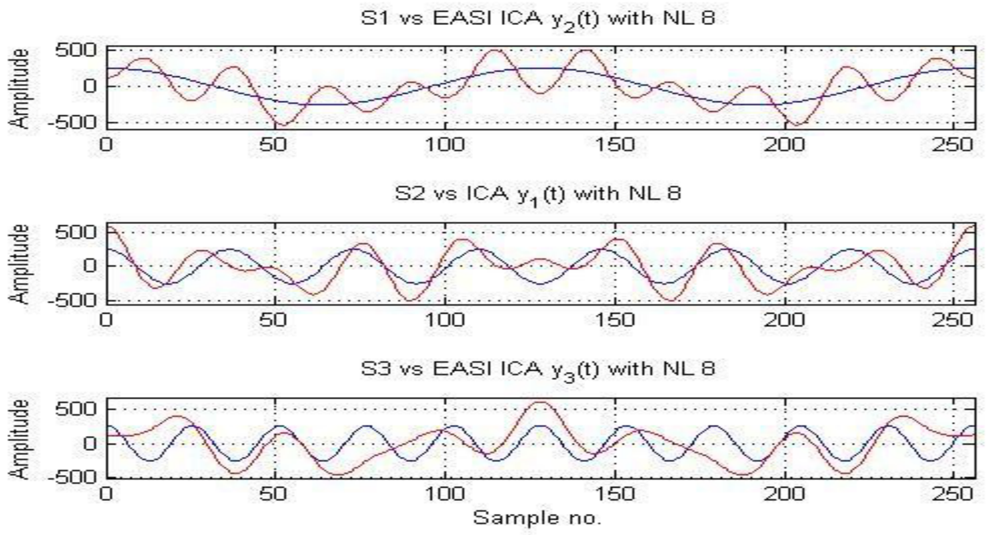

Figure 4.2.2.8.2: Comparison of source and separated signals for nonlinearity eight.

In Table 4.2.2.8, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created a FFT spectrum error of 29.130%. Signal two and signal three experienced FFT spectrum errors of 48.000%and 50.276% respectively. The product of the mixing matrix and separation matrix had an error of 4.800%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.331%

Table 4.2.2.8 Nonlinearity eight data for analysis

Table 4.2.2.8 NL 8 Analysis Data	
Performance Aspect	Quantitative Values
Visibly Correct	No
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (29.130%)
Signal 2 – Max Freq (%Error)	2 kHz (48.000%)
Signal 3 – Max Freq (%Error)	10 kHz (50.276%)
Mixing Separation Product Error	4.800%
Mixing Separation Difference Error	0.331%

4.2.2.9 Nonlinearity Nine: Piecewise function two

The EASI ICA algorithm was implemented using nonlinearity nine to create the following separated signals seen in Figure 4.2.2.9.1.

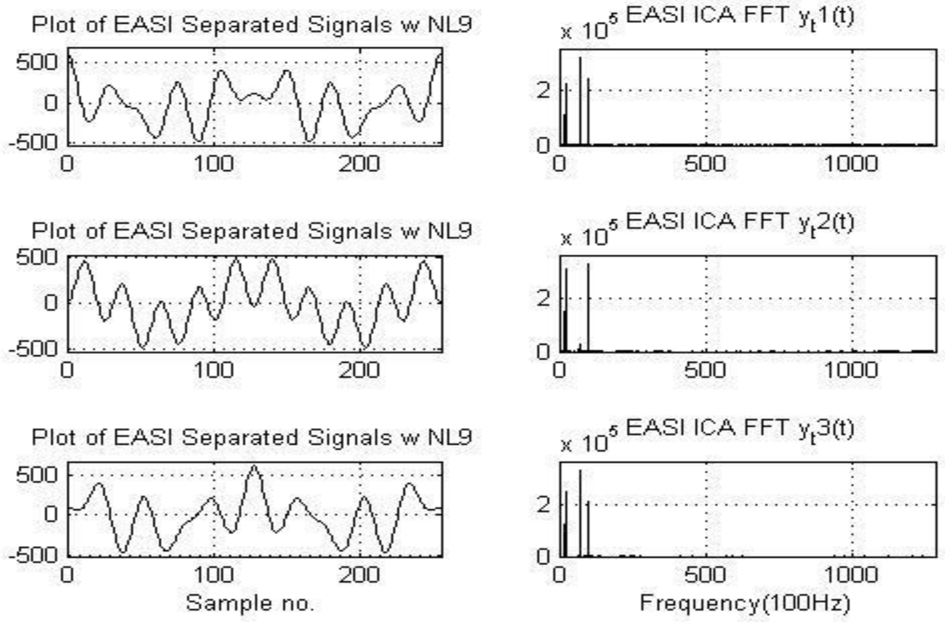

Figure 4.2.2.9.1: Separated signals using nonlinearity nine.

A comparison of the source and separated signals are shown in Figure 4.2.2.9.2.

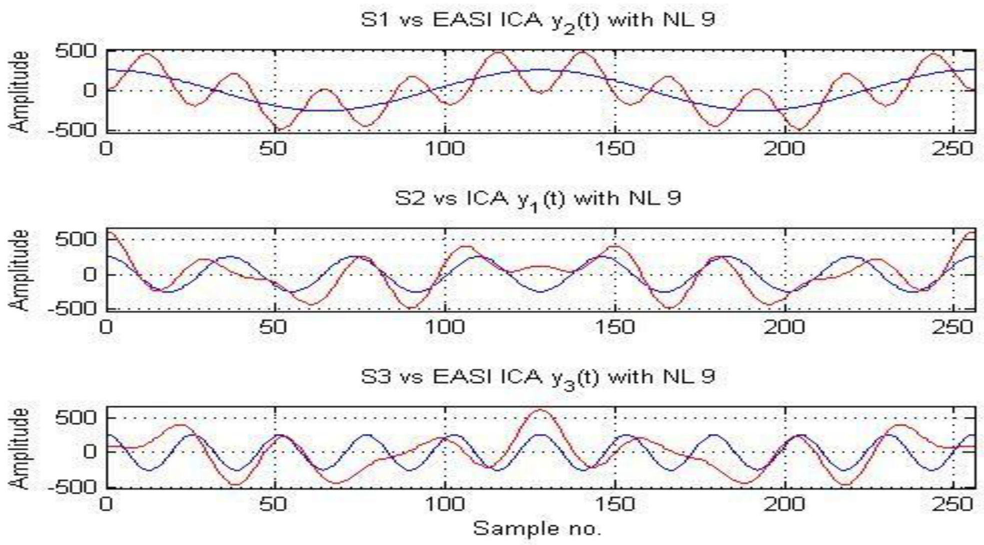

Figure 4.2.2.9.2: Comparison of source and separated signals for nonlinearity nine.

In Table 4.2.2.9, the maximum energies were detected around approximately 7 kHz for signal one, 2 KHz for signal two and 10 kHz for signal three. The excess energies detected in the FFT spectrum plot for signal one created a FFT spectrum error of 29.160%. Signal two and signalthree experienced FFT spectrum errors of 40.570% and 47.570% respectively. The product of the mixing matrix and separation matrix had an error of 0.154%. The difference of the separation matrix and the inversion of the mixing matrix had an error of 0.725%.

Table 4.2.2.9 Nonlinearity nine data for analysis

Table 4.2.2.9 NL 9 Analysis Data	
Performance Aspect	Quantitative Values
Visible Correct	No
FFT Measurements	
Signal 1 – Max Freq (%Error)	7 kHz (29.160%)
Signal 2 – Max Freq (%Error)	2 kHz (40.570%)
Signal 3 – Max Freq (%Error)	10kHz (47.570%)
Mixing Separation Product Error	0.154%
Mixing Separation Difference Error	0.725%

4.2.3 Resource Requirements for Nonlinearities

The nonlinearity resource requirements were determined. The nonlinearities with the same number of resource requirements were grouped together. Table 4.2.3.1 shows the resources required to implement the EASI algorithm with nonlinearities one, two, and four in hardware. The nonlinearities would not require any extra resources for implementation.

Table 4.2.3.1: Resources required for nonlinearities one, two, and four.

Table 4.2.3.1: Resources Required for Nonlinearities One, Two, and Four			
Equation	Operation resource	Breakdown	Total Resources
Yk = B * x	9 multipliers 6 adders	3 mults, 2adds 3 mults, 2adds 3 mults, 2adds	9 multipliers 6 adders

Table 4.2.3.1: Resources Required for Nonlinearities One, Two, and Four continued			
Equation	Operation resource	Breakdown	Total Resources
H = I – yk*yk' + f*yk' - yk*f ', f = f(y)	27 multipliers 27 adders (18 subtraction)	9*(3 multipliers + 3 adders) (9*2 subtraction)	27 multipliers 27 adders (18 subtraction)
B = B + m * **H * B**	18 multipliers 9 adders	9*(2 mults + 1 adders)	18 multipliers 9 adders
Yt = B * x	9 multipliers 6 adders	3 mults,2 adders 3 mults,2 adders 3 mults,2 adders	9 multipliers 6 adders
		Total Resources	111

Table 4.2.3.2 shows the resources required to implement the EASI algorithm with nonlinearity 9 in hardware. The nonlinearities would all require look up tables in the form of mif files for implementation. The nonlinearities require two additional adders for implementation.

Table 4.2.3.2: Resources required for nonlinearity nine.

Table 4.2.3.2: Resources Required for Nonlinearity Nine			
Equation	Operation resource	Breakdown	Total Resources
Yk = B * x	9 multipliers 6 adders	3 mults, 2adds 3 mults, 2adds 3 mults, 2adds	9 multipliers 6 adders
H = I – yk*yk' + f*yk' - yk*f ', f = f(y)	27 multipliers 27 adders (18 subtraction) (2adders(subtraction))	9*(3 multipliers + 3 adders) (9*2 subtraction) (1 adder(subtraction)) (1 adder subtraction)	27 multipliers 29 adders (18 subtraction) (2 function adders(subtraction))
B = B + m * **H * B**	18 multipliers 9 adders	9*(2 mults + 1 adders)	18 multipliers 9 adders

Table 4.2.3.2: Resources Required for Nonlinearity Nine continued			
Equation	Operation resource	Breakdown	Total Resources
Yt = B * x	9 multipliers 6 adders	3 mults, 2adds 3 mults, 2adds 3 mults, 2adds	9 multipliers 6 adders
		Total Resources	113

Table 4.2.3.3 shows the resources required to implement the EASI algorithm with nonlinearities three, five, six ,and eight in hardware. The nonlinearities would all require look up tables in the form of mif files for implementation. The nonlinearities require four additional multipliers and a LUT for implementation.

Table 4.2.3.3: Resources required for nonlinearities three, five, six, and eight.

Table 4.3.4.3: Resources Required for nonlinearities three, five, six and eight			
Equation	Operation resource	Breakdown Resource	Total Resources
Yk = B * x	9 multipliers 6 adders	3 mults, 2adds 3 mults, 2adds 3 mults, 2adds	9 multipliers 6 adders
H =I – yk * yk' + f* yk' – yk* f'	27 multipliers 27 adders (18 subtraction) (4 function multipliers) +LUT	9*(3 multipliers + 3 adders) (9*2 subtraction) (2 function multipliers) (2 function multipliers)	31 multipliers 27 adders (18 subtraction) (4 function multipliers) +LUT
B = B + m * **H * B**	18 multipliers 9 adders	9*(2 mults + 1 adders)	18 multipliers 9 adders
Yt = B * x	9 multipliers 6 adders	3 mults, 2adds 3 mults, 2adds 3 mults, 2adds	9 multipliers 6 adders
		Total Resources	115

4.2.4 Nonlinearity Comparison and Selection

Table 4.2.4 shows the comparison of the all the elements of each of the nonlinearities. As can be seen by the table is the worse nonlinearity is nonlinearity two. The best nonlinearities are nonlinearities six and seven. The best fit for hardware implementation of the EASI ICA variation is nonlinearity six as it has the lowest error percentages across the board.

Table 4.2.4. Comparison of nonlinearity data results

Table 4.2.4 Comparison of Nonlinearity Data Results					
Performance Aspect	1	2	3	4	5
Visibly Correct	Partially Yes	No	No	No	No
FFT Measurements					
Signal 1 – Max Freq (%Error)	7 kHz 0.0167%	7 kHz 22.030%	7 kHz 29.390%	7 kHz 6.860%	7 kHz 2.043%
Signal 2 – Max Freq (%Error)	2 kHz 2.309%	2 kHz 37.940%	2 kHz 49.660%	2 kHz 41.100%	2 kHz 5.096%
Signal 3 – Max Freq (%Error)	10kHz 2.303%	10kHz 55.720%	10kHz 29.390%	10kHz 35.830%	10 kHz 4.656%
Mixing Separation Product Error	1.218%	5.595%	3.676%	1.836%	1.280%
Mixing Separation Difference Error	0.105%	12.36%	3.614%	2.220%	0.169%

Table 4.2.4 Comparison of Nonlinearity Data Results continued				
Performance Aspect	6	7	8	9
Visibly Correct	Yes	Mostly Yes	No	No
FFT Measurements				
Signal 1 – Max Freq (%Error)	7 kHz 0.0049%	7 kHz 0.129%	7 kHz 29.130%	7 kHz 29.160%
Signal 2 – Max Freq (%Error)	2 kHz 2.0555%	2 kHz 2.0580%	2 kHz 48.000%	2 kHz 40.570%
Signal 3 – Max Freq (%Error)	10 kHz 2.0624%	10 kHz 2.175%	10 kHz 50.276%	10kHz 47.570%
Mixing Separation Product Error	1.199%	1.108%	4.800%	0.154%
Mixing Separation Difference Error	0.0823%	0.129%	0.331%	0.725%

4.3 Additional Analysis

4.3.1 Multiple Sources

The possibility of using more sources was analyzed. Table 4.3.1 shows the amount of resources that are required for implementations of the EASI up to five sources.

Table 4.3.1: Resources required for EASI implementation using three to five resources.

Table 4.3.1: EASI –ICA Resources for Three to Five Sources			
Equation	3 Sources	4 Sources	5 Sources
$\boldsymbol{Yk = B * X}$ Multipliers = (n^2) Adders = (n-1)*n	9 multipliers 6 adders	16 multipliers 12 adders	25 multipliers 20 adders

Table 4.3.1: EASI –ICA Resources for Three to Five Sources continued			
Equation	3 Sources	4 Sources	5 Sources

H = **I** -yk*yk' + f*yk' - yk*f ' f = f(y)
 	
Multipliers =n^3
Adders = n^3
	27 multipliers
	27 adders
	(18 subtraction)
		64 multipliers
		64 adders
		(32 subtraction)
			125 multipliers
			125 adders
			(50 subtraction)

B = **B** + m***H*****B**
Multipliers = 2* n^2
Adders = n^2
	18 multipliers
	9 adders
		18 multipliers
		16 adders
			100 multipliers
			25 adders

$\boldsymbol{Yt} = \boldsymbol{B} * \boldsymbol{X}$
Multipliers = (n^2)
Adders = (n-1)*n
	9 multipliers
	6 adders
		16 multipliers
		12 adders
			25 multipliers
			20 adders

Total Resources	63 multipliers 48 adders (18 Subtraction)	114 multipliers 104 adders (32 subtraction)	275 multipliers 190 adders (50 subtraction)

4.3.2 Comparison of the EASI general algorithm the EASI variation

The general version of the EASI algorithm and the variation of the algorithm are very similar in implementation. The difference in each algorithms implementation occurs during the **H** matrix equation implementation. The variation uses two nonlinearities instead of one used by the original as can be seen in Table 4.3.2. The variation would require twice as many resources to efficiently implement it. The original algorithm remains the best approach.

Table 4.3.2: Comparison of the general EASI algorithm and the two function variation.

Table 4.3.2: Comparison of the General EASI and EASI Two Function Variation		
Algorithm Version	H Matrix Difference	Resources Max Possible
EASI General	**H** = **I** – yk*yk' + f*yk' - yk*f', f = f(y)	63 multipliers 48 adders + 4 multiplier + LUT

Table 4.3.2: Comparison of the General EASI and EASI Two Function Variation Continued		
Algorithm Version	H Matrix Difference	Resources Max Possible
EASI Variation	**H** = **I** – yk*yk’ + f1*f2’ – f2*f1’, f = f(y)	63 multipliers 48 adders +8 multiplier + 2LUT

4.3.3 The Use of Independent Audio Sources

The EASI ICA was implemented with three audio files shown in Figure 4.3.3.1 to see if audio files could be effectively separated with the EASI algorithm. The first signal was a clip of a flute maximum FFT spectrum energy at 940 kHz. The second signal was of a clip of a flute with a maximum FFT spectrum energy at 490 kHz. The last signal was a female speaking with maximum FFT spectrum energy at 240 kHz. The clips had a sampling frequency of 8 kHz.

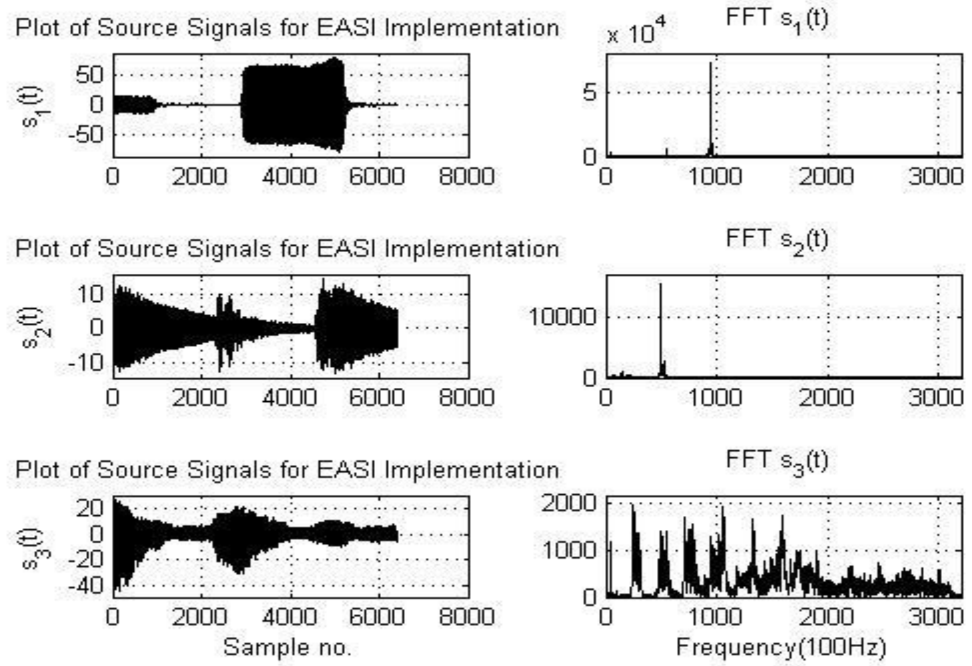

Figure 4.3.3.1: Independent audio signals used for EASI ICA analysis.

. The mixed audio signals were created as shown in Figure 4.3.3.2.

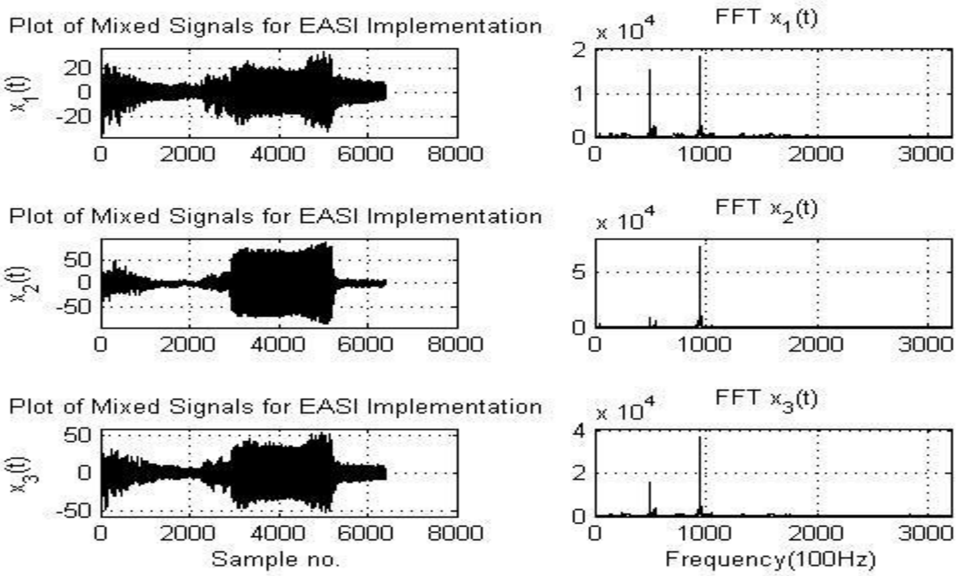

Figure 4.3.3.2: Mixed audio signals created for EASI ICA analysis.

The separations of the mixed signals are shown in Figure 4.3.3.3. The visual comparison of the source and separated signals are shown in Figure 4.3.3.4.

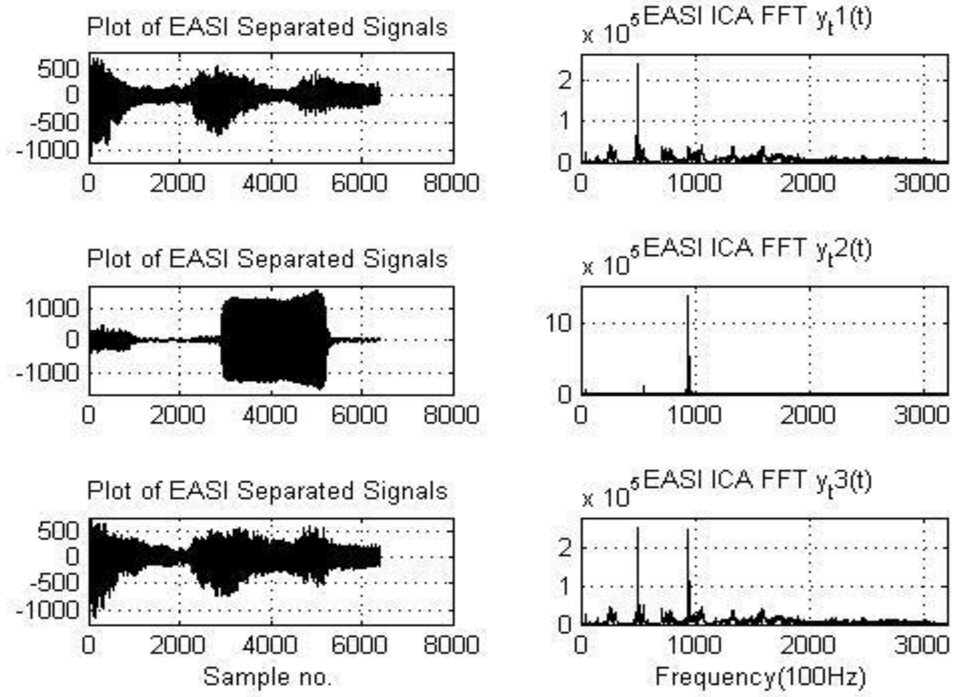

Figure 4.3.3.3: Separated audio signals using the EASI ICA algorithm

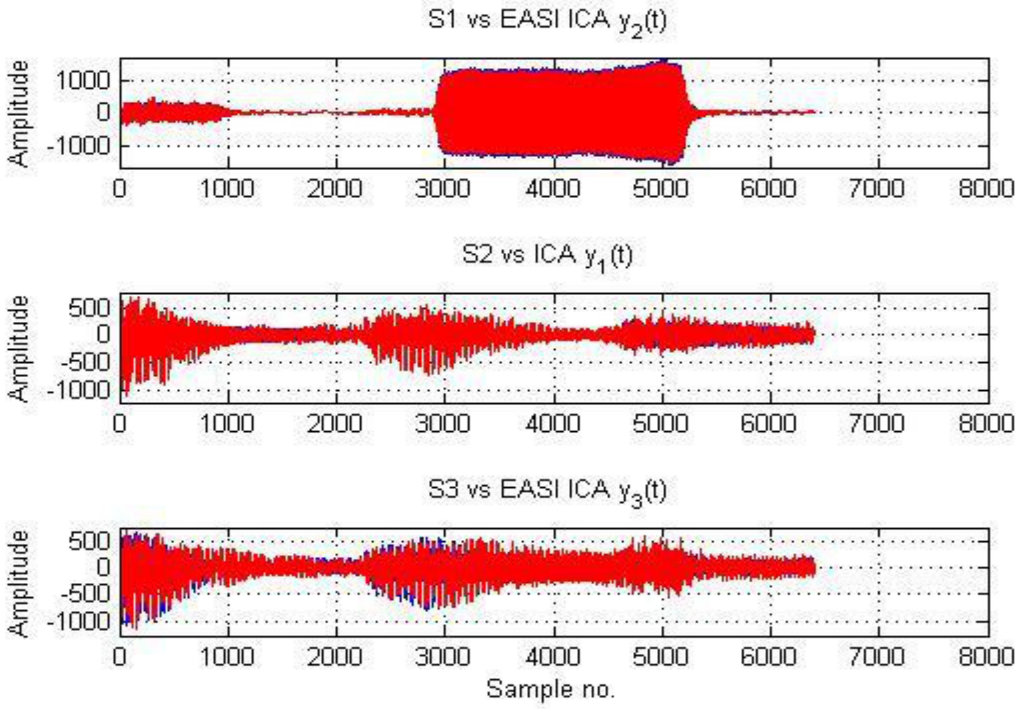

Figure 4.3.3.4: Comparison of source and separated audio signals.

The separation matrix through the EASI ICA convergence and the resulting errors are shown in Figure 4.3.3.5 and Figure 4.3.3.6 respectively.

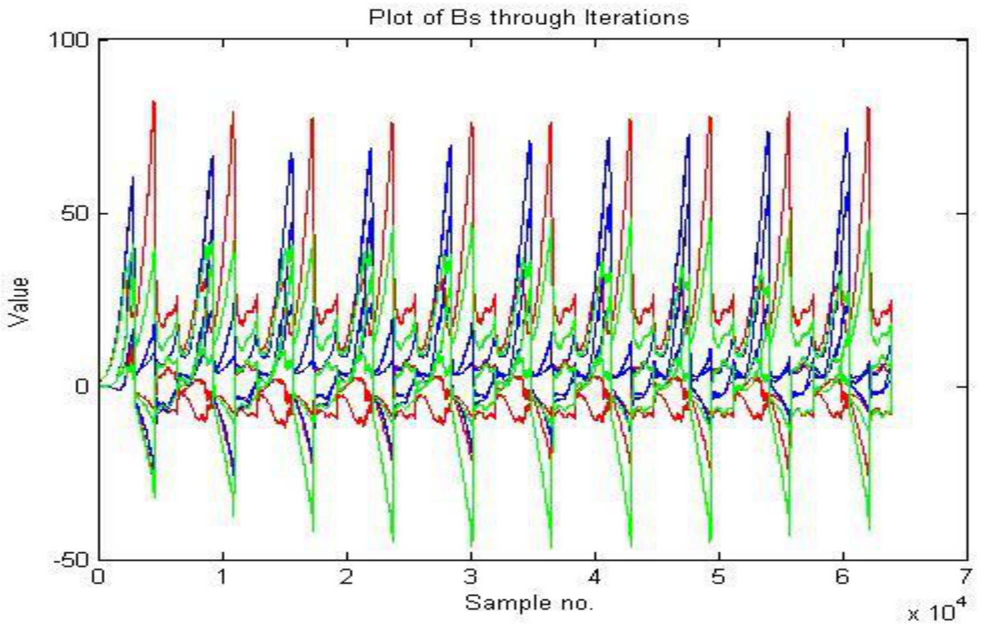

Figure 4.3.3.5: Convergence of separation matrix for the audio signals using EASI ICA.

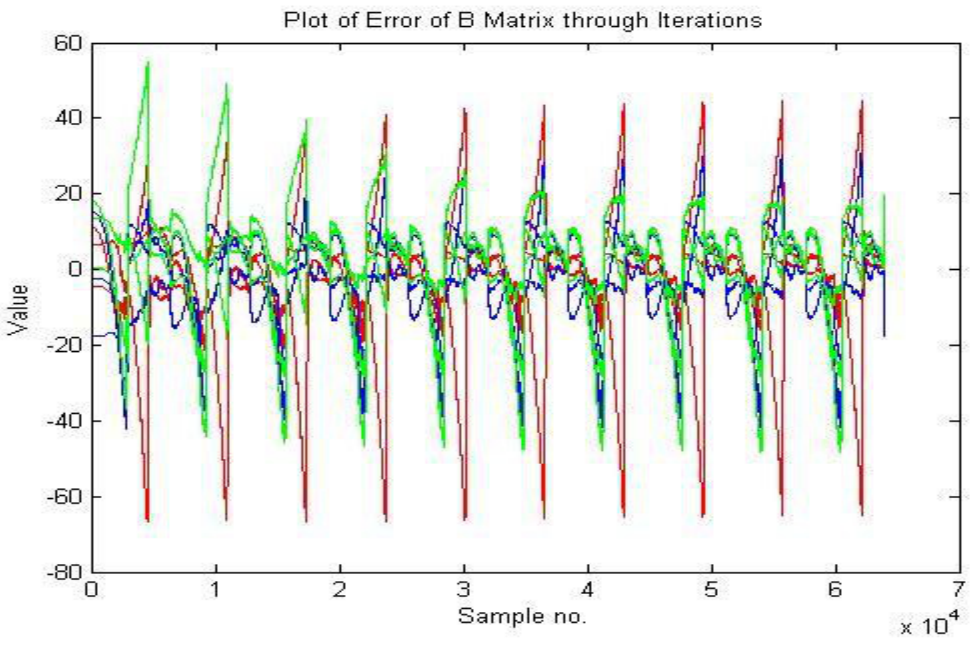

Figure 4.3.3.6: Errors of separation matrix for the audio signals using EASI ICA.

In Table 4.3.3, the maximum FFT spectrum energies were detected at approximately 49kHz for signal one, 940 kHz for signal two and 490 kHz for signal three . The excess energies detected in the FFT spectrum for signal one created a FFT spectrum error of 199.94%. Signal two and signal three experienced FFT spectrum errors of 23.376% and 206.82% respectively. The product of the mixing matrix and separation matrix had an error of 11.769%.The difference of the separation matrix and the inversion of the mixing matrix had an error of 1.412%

Table 4.3.3: EASI ICA data for analysis for audio signals

Table 4.3.3: EASI ICA Analysis Data for Audio Signals	
Performance Aspect	Quantitative Values
FFT Measurements	
Signal 1 - Max Freq (%Error)	490kHz (199.94%)
Signal 2 – Max Freq (%Error)	940kHz (23.376%)
Signal 3 – Max Frey (%Error)	490kHz (206.82%)
Mixing Separation Product Error	11.769%
Mixing Separation Difference Error	1.412%

As can be seen by the audio signals while they are independent source are difficult to analyze with the EASI algorithm. This means use audio sources would difficult to analyze the effectiveness of the EASI ICA in hardware. Simple independent sources are necessary for easy analysis. The three sinusoidal of the 2 kHz, 7 kHz, and 10 kHz frequencies are the best for EASI ICA implementation in hardware.

4.4 Conclusion

The EASI ICA algorithm is the chosen algorithm for implementation in hardware. The mixing matrices, nonlinearities, and the use best use of sources for the algorithm were examined. The current mixing matrix, hyperbolic tangent and independent sinusoidal sources were the best and simplest for implementation. The algorithm will be implemented with three sources because of its feasibility and lowest requirement of resources.

CHAPTER FIVE

EASI IMPLEMENTION WITH AN FPGA

5.1 The EASI Algorithm Hardware Implementation

The EASI algorithm was chosen for implementation in hardware. As can be seen in Figure 5.1.1 the EASI algorithm has multiple stages. The matrix equations with in the algorithm had to be broken down to the individual multipliers and dividers as can be seen in Table 5.1.1.

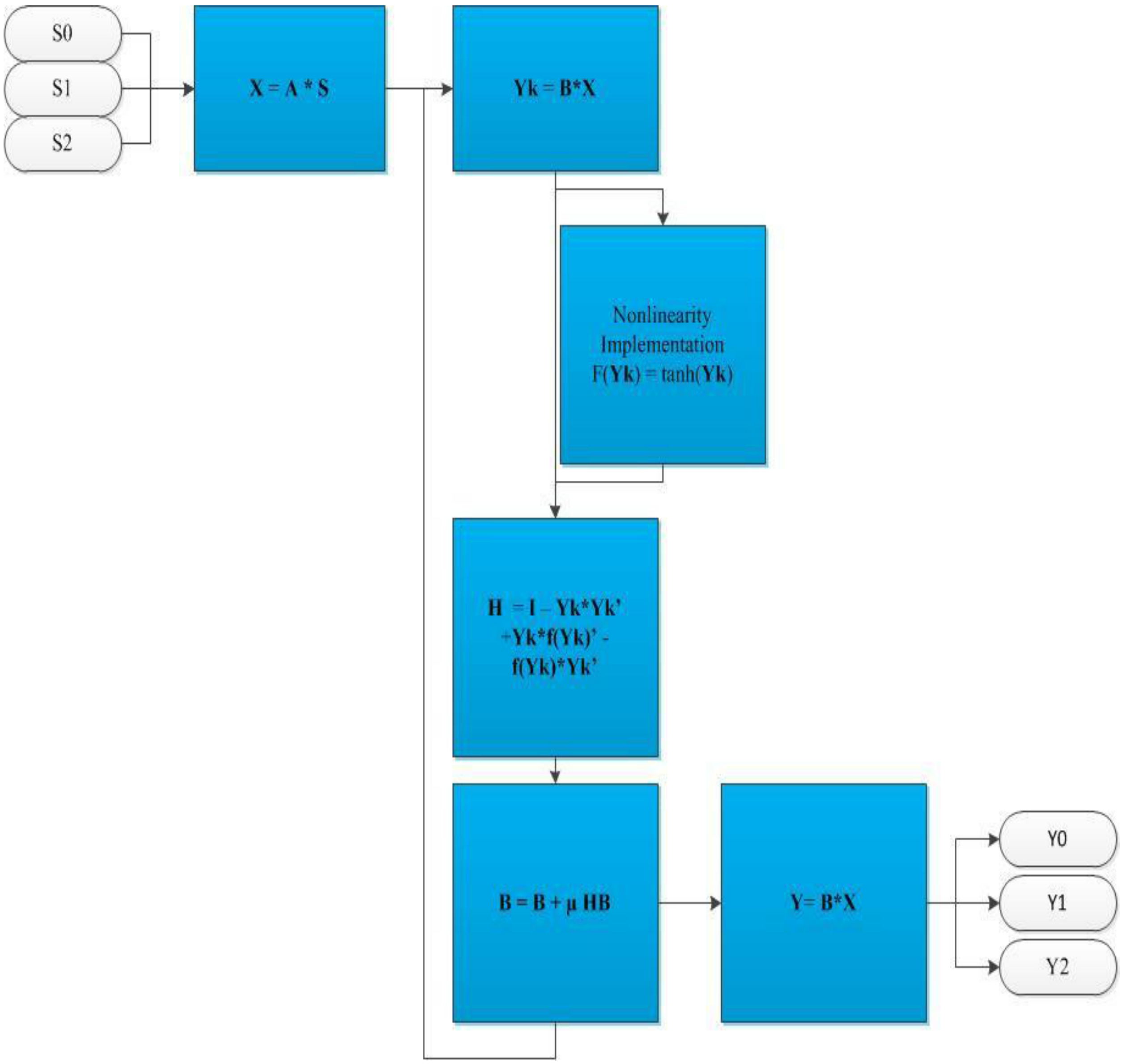
S0
S1
S2
X = A * S
Yk = B*X
Nonlinearity Implementation F(Yk) = tanh(Yk)
H = I – Yk*Yk' +Yk*f(Yk)' - f(Yk)*Yk'
B = B + μ HB
Y= B*X
Y0
Y1
Y2

Figure 5.1.1: The stages of the EASI algorithm.

Table 5.1 The matrix equations needed to implement the algorithm in hardware.

Table 5.1: EASI ICA Matrix Multiplication Requirements
$x_0 = A_{00}s_0 + A_{01}s_1 + A_{02}s_2$ $x_1 = A_{10}s_0 + A_{11}s_1 + A_{12}s_2$ $x_2 = A_{20}s_0 + A_{21}s_1 + A_{22}s_2$
$yk_0 = B_{00}x_0 + B_{01}x_1 + B_{02}x_2$ $yk_1 = B_{10}x_0 + B_{11}x_1 + B_{12}x_2$ $yk_2 = B_{20}x_0 + B_{21}x_1 + B_{22}x_2$
$H_{00} = 1 - \mathrm{yk}_0 * \mathrm{yk}_0 + \mathrm{f}_0 * \mathrm{yk}_0 - \mathrm{yk}_0 * \mathrm{f}_0$ $H_{01} = 0 - \mathrm{yk}_0 * \mathrm{yk}_1 + \mathrm{f}_0 * \mathrm{yk}_1 - \mathrm{yk}_0 * \mathrm{f}_1$ $H_{02} = 0 - \mathrm{yk}_0 * \mathrm{yk}_2 + \mathrm{f}_0 * \mathrm{yk}_2 - \mathrm{yk}_0 * \mathrm{f}_2$ $H_{10} = 0 - \mathrm{yk}_1 * \mathrm{yk}_0 + \mathrm{f}_1 * \mathrm{yk}_0 - \mathrm{yk}_1 * \mathrm{f}_0$ $H_{11} = 1 - \mathrm{yk}_1 * \mathrm{yk}_1 + \mathrm{f}_1 * \mathrm{yk}_1 - \mathrm{yk}_1 * \mathrm{f}_1$ $H_{12} = 0 - \mathrm{yk}_1 * \mathrm{yk}_2 + \mathrm{f}_1 * \mathrm{yk}_2 - \mathrm{yk}_1 * \mathrm{f}_2$ $H_{20} = 0 - \mathrm{yk}_2 * \mathrm{yk}_0 + \mathrm{f}_2 * \mathrm{yk}_0 - \mathrm{yk}_2 * \mathrm{f}_0$ $H_{21} = 0 - \mathrm{yk}_2 * \mathrm{yk}_1 + \mathrm{f}_2 * \mathrm{yk}_1 - \mathrm{yk}_2 * \mathrm{f}_1$ $H_{22} = 1 - \mathrm{yk}_2 * \mathrm{yk}_2 + \mathrm{f}_2 * \mathrm{yk}_2 - \mathrm{yk}_2 * \mathrm{f}_2$
$B_{00} = B_{00} + \mu * (H_{00} * B_{00} + H_{01} * B_{10} + H_{02} * B_{20})$ $B_{01} = B_{01} + \mu * (H_{00} * B_{01} + H_{01} * B_{11} + H_{02} * B_{21})$ $B_{02} = B_{02} + \mu * (H_{00} * B_{02} + H_{01} * B_{12} + H_{02} * B_{22})$ $B_{10} = B_{10} + \mu * (H_{10} * B_{00} + H_{11} * B_{10} + H_{12} * B_{20})$ $B_{11} = B_{11} + \mu * (H_{10} * B_{01} + H_{11} * B_{11} + H_{12} * B_{21})$ $B_{12} = B_{12} + \mu * (H_{10} * B_{02} + H_{11} * B_{12} + H_{12} * B_{22})$ $B_{20} = B_{20} + \mu * (H_{20} * B_{00} + H_{21} * B_{10} + H_{22} * B_{20})$ $B_{21} = B_{21} + \mu * (H_{20} * B_{01} + H_{21} * B_{11} + H_{22} * B_{21})$ $B_{22} = B_{22} + \mu * (H_{20} * B_{02} + H_{21} * B_{12} + H_{22} * B_{22})$
$yt_0 = B_{00}x_0 + B_{01}x_1 + B_{02}x_2$ $yt_1 = B_{10}x_0 + B_{11}x_1 + B_{12}x_2$ $yt_2 = B_{20}x_0 + B_{21}x_1 + B_{22}x_2$

The algorithm was implemented with the following parameters:

- The numerical input values were 16 integer bit values where first 8 bits are integer bits and the last bits are 8 fractional bits.
- The look up tables values were 16 bit integer values.
- Three independent cosine function look up tables for frequencies of 2kHz, 7kHz, and 10kHz
- One look up table will contain values corresponding to the hyperbolic tangent nonlinearity
- The signals were mixed with values of mixing matrix one
- The learning rate will be 2-9
- The matrix multipliers will be scales by a factor of 2^8 to prevent overflow
- The numerical output will be 8 bit integer values.
- The implementation will be done in Quartus 2 for Altera utilizing a Cyclone II FPGA.

The algorithm was implemented in hardware using three processes and a finite state machine as seen in Figure 5.1.2.

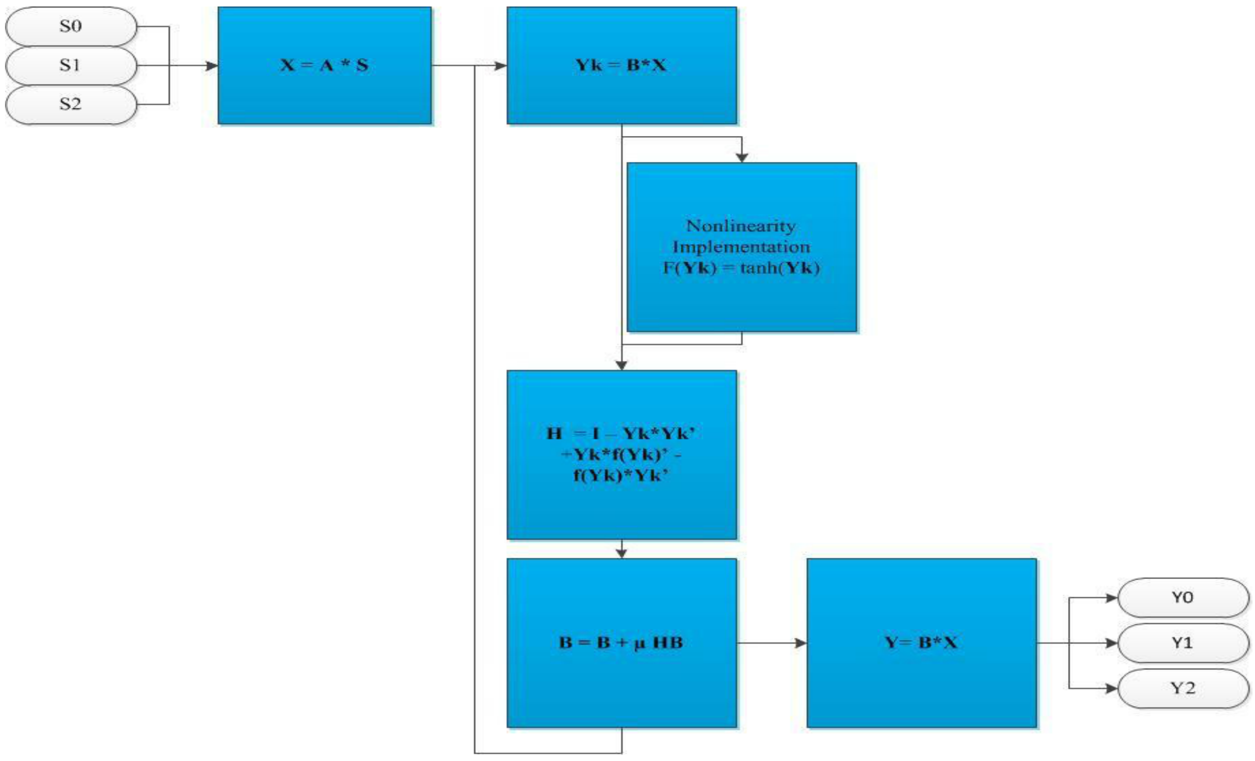
S0
S1
S2
X = A * S
Yk = B*X
Nonlinearity Implementation F(Yk) = tanh(Yk)
H = I – Yk*Yk' +Yk*f(Yk)' - f(Yk)*Yk'
B = B + μ HB
Y= B*X
Y0
Y1
Y2

Figure 5.1.2: The stages of the EASI algorithm in hardware.

5.2 Source and Mixed Signals Test bench

The implementation of the EASI algorithm in hardware began with the development of a source and mixture test bench as shown in diagram in Figure 5.2.1. The test bench was created to ensure the source inputs were correctly loaded into Altera, and the mixing matrix mixed the signals correctly.

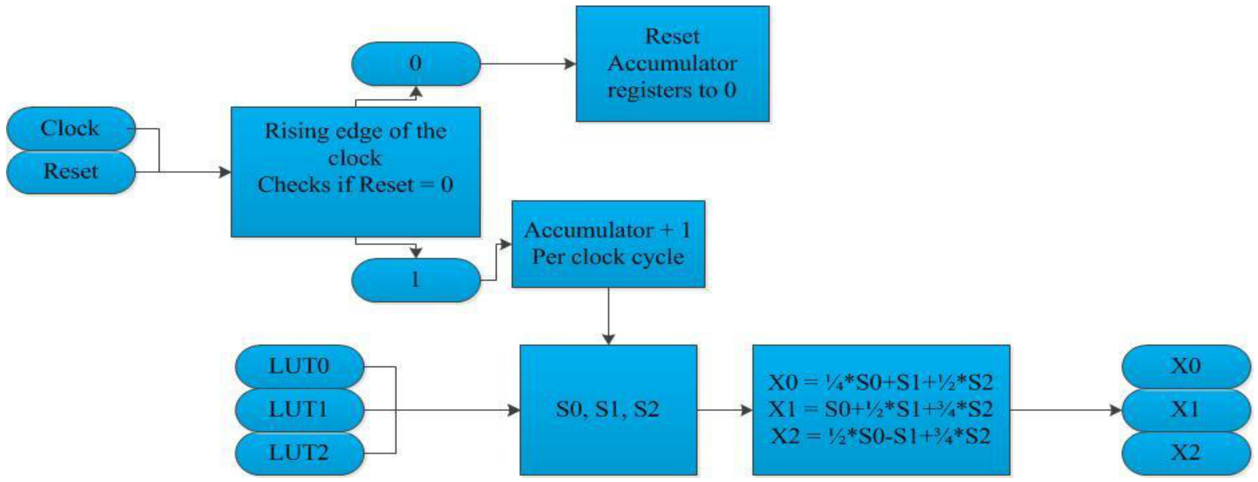
0
Reset Accumulator registers to 0
Clock
Reset
Rising edge of the clock Checks if Reset = 0
Accumulator + 1 Per clock cycle
1
LUT0
LUT1
LUT2
S0, S1, S2
X0 = ¼*S0+S1+½*S2
X1 = S0+½*S1+¾*S2
X2 = ½*S0-S1+¾*S2
X0
X1
X2

Figure 5.2.1: Diagram of the source and mixed signals Test Bench.

The simulation of the test bench was run using components seen in Table 5.2.

Table 5.2 Components of the Source and Mixed Signals Test Bench

Table 5.2: Components of the Source and Mixed Signals Test Bench	
Component	Purpose
Clock 50	50 MHz clock for the simulation
Reset	Used to clear out the registers and reset the accumulator to zero.
Accumulator	Increments the loading inputs from the Look Up Tables
S0, S1 & S2	Each array holds information from the look up table for each source
X0, X1 & X2	Mixed signals arrays
LUT 0, 1, 2, 3	Look up tables for the three independent signals and Nonlinearity signal

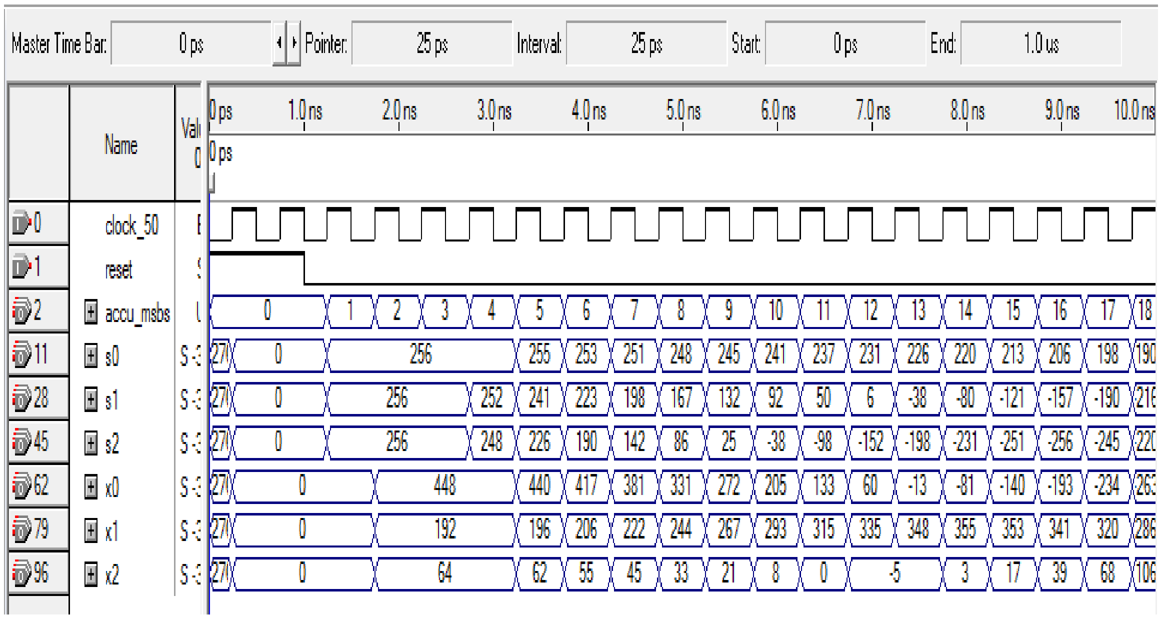
Master Time Bar: 0 ps
Pointer: 25 ps
Interval: 25 ps
Start: 0 ps
End: 1.0 us
Name
0 ps 1.0 ns 2.0 ns 3.0 ns 4.0 ns 5.0 ns 6.0 ns 7.0 ns 8.0 ns 9.0 ns 10.0 ns
0 clock_50
1 reset
2 accu_msbs 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
11 s0 27 0 256 255 253 251 248 245 241 237 231 226 220 213 206 198 190
28 s1 27 0 256 252 241 223 198 167 132 92 50 6 -38 -80 -121 -157 -190 -216
45 s2 27 0 256 248 226 190 142 86 25 -38 -98 -152 -198 -231 -251 -256 -245 -220
62 x0 27 0 448 440 417 381 331 272 205 133 60 -13 -81 -140 -193 -234 -263
79 x1 27 0 192 196 206 222 244 267 293 315 335 348 355 353 341 320 286
96 x2 27 0 64 62 55 45 33 21 8 0 -5 3 17 39 68 106

Figure 5.2.2 Simulation results of the Source and Mixed Signals Test Bench.

5.3 Nonlinear Function Test Bench

A Nonlinearity Implementation Test Bench was developed as shown in diagram in Figure 5.3.1. The test bench was created to ensure the nonlinearity values were loaded and converted correctly. The VHDL code for the test bench is in the appendix.

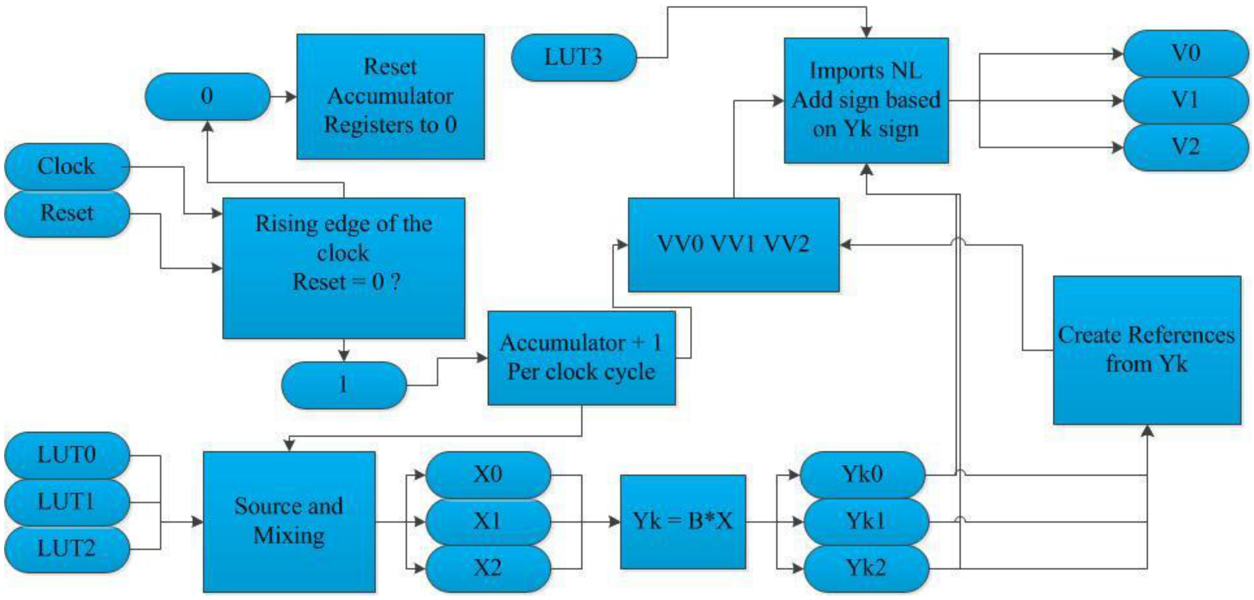
0
Reset Accumulator Registers to 0
LUT3
Imports NL Add sign based on Yk sign
V0
V1
V2
Clock
Reset
Rising edge of the clock Reset = 0 ?
VV0 VV1 VV2
Accumulator + 1 Per clock cycle
1
Create References from Yk
LUT0
LUT1
LUT2
Source and Mixing
X0
X1
X2
Yk = B*X
Yk0
Yk1
Yk2

Figure 5.3.1: Diagram of the nonlinear test bench.

The simulation of the test bench was run using components seen in Table 5.3.

Table 5.3 Components of the Nonlinearity Implementation Test Bench

Table 5.3: Components of the Source and Mixed Signals Test bench	
Component	Purpose
Clock 50	50 MHz clock for the simulation
Reset	Used to clear out the registers and reset the accumulator to zero.
Accumulator	Increments the loading inputs from the Look Up Tables
S0, S1 & S2	Each array holds information from the look up table for each source
X0, X1 & X2	Mixed signals arrays
LUT 0, 1, 2, 3	Look up tables for the three independent signals and Nonlinearity signal
VV 0, VV1, VV2	Index References for the nonlinearity
V0, V1, V2	Nonlinearity Values

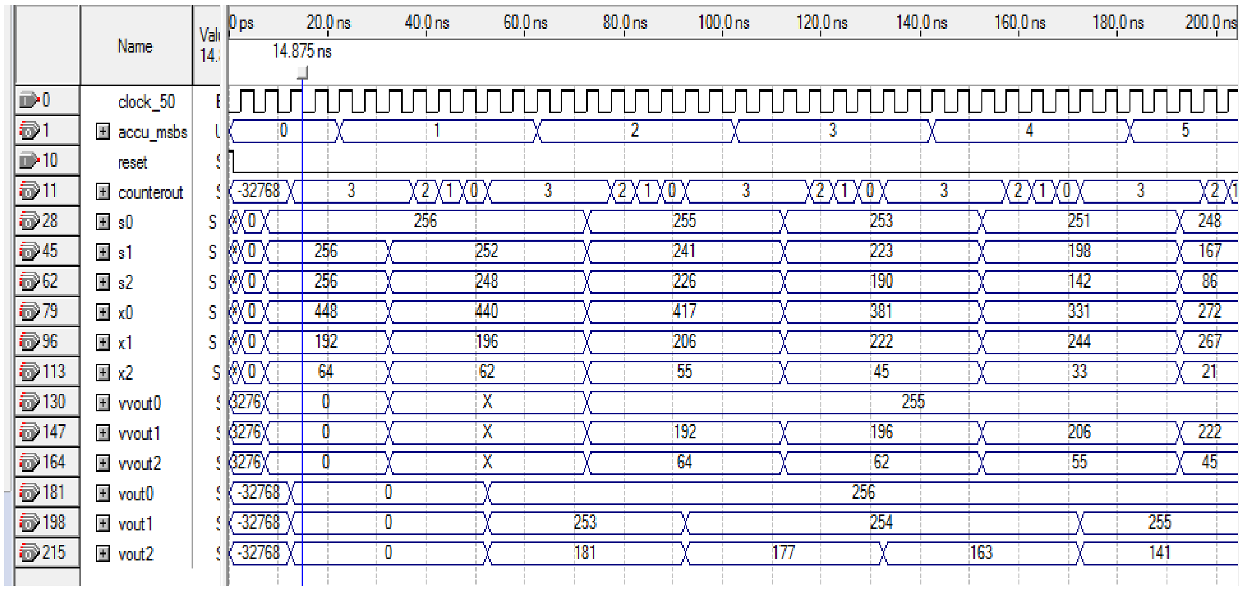

Figure 5.3.2 Simulation results of the Nonlinearity Implementation Test Bench.

5.4 EASI Full Implementation Test Bench

5.4.1 The EASI Full Implementation Overview

The functionality of the Source and Mixed Signals and Nonlinearity Implementation Test Benches were added together. The test benches compilation and the addition of a finite state machine led to the development of the EASI Full Implementation Test bench as shown in the diagram in Figure 5.4.1. The VHDL code for this test bench is located in Appendix

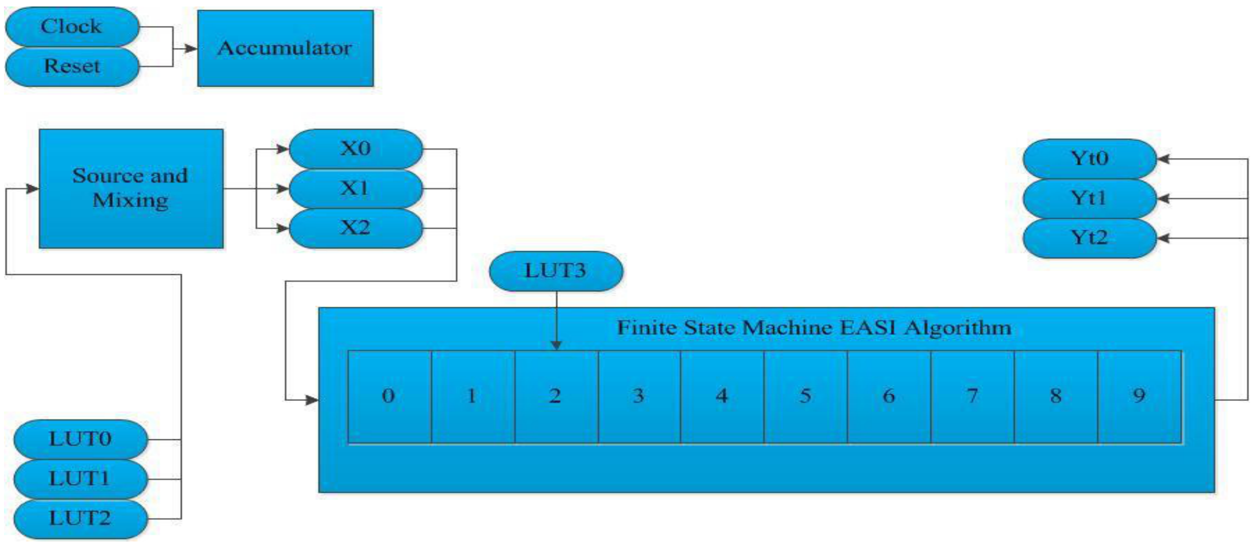

Figure 5.4.1: Diagram of Full Implementation EASI Test Bench.

5.4.2 Finite State Machine Addition

A Finite State Machine was created to move the algorithm through each step as shown in Figure 5.4.2.1.

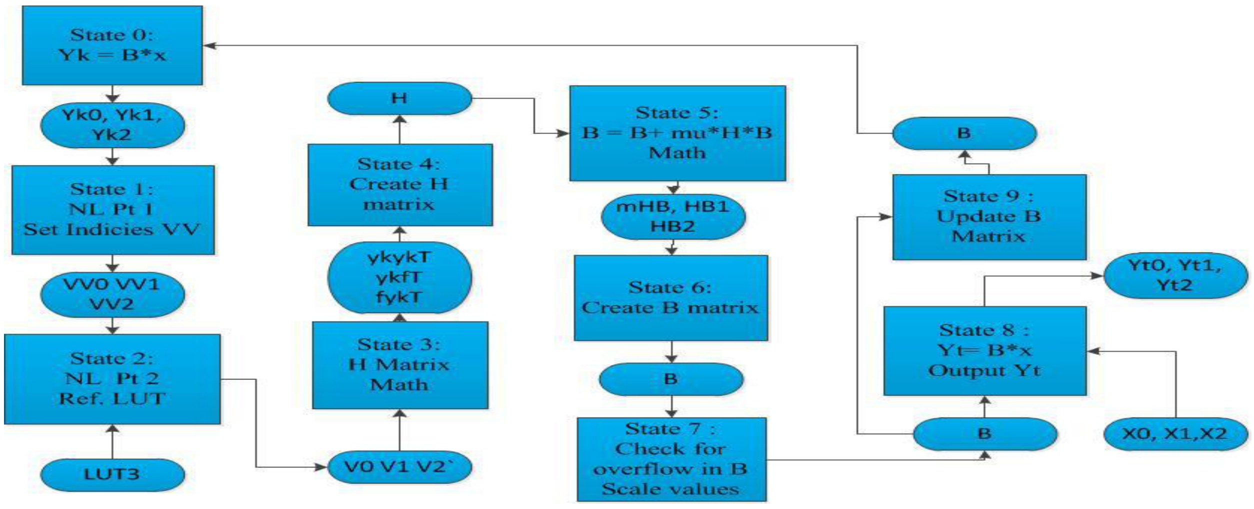
State 0: Yk = B*x
Yk0, Yk1, Yk2
State 1: NL Pt 1 Set Indicies VV
VV0 VV1 VV2
State 2: NL Pt 2 Ref. LUT
LUT3
H
State 4: Create H matrix
ykykT ykfT fykT
State 3: H Matrix Math
V0 V1 V2`
State 5: B = B+ mu*H*B Math
mHB, HB1 HB2
State 6: Create B matrix
B
State 7 : Check for overflow in B Scale values
B
State 9 : Update B Matrix
Yt0, Yt1, Yt2
State 8 : Yt= B*x Output Yt
B
X0, X1,X2

Figure 5.4.2.1: Diagram of the finite state machine used to implement EASI ICA.

Counters were implemented to insure all steps of the algorithm were performed at the appropriate times. The counters were necessary because:

- Some portions of the algorithm such as the nonlinearity implementation required multiple clock cycles to load or calculate values before the next step.
- VHDL is procedural coding environment so all portions of the algorithm happen in real time at the same time. Including matrix updates.

Without the counters and state machine the EASI algorithm would move to the next step of implementation before Matrix and arrays were updated correctly. The resulting values would be calculated incorrectly.

The complete EASI Full Implementation Test Bench is shown in the diagram of Figure 5.4.2.2.

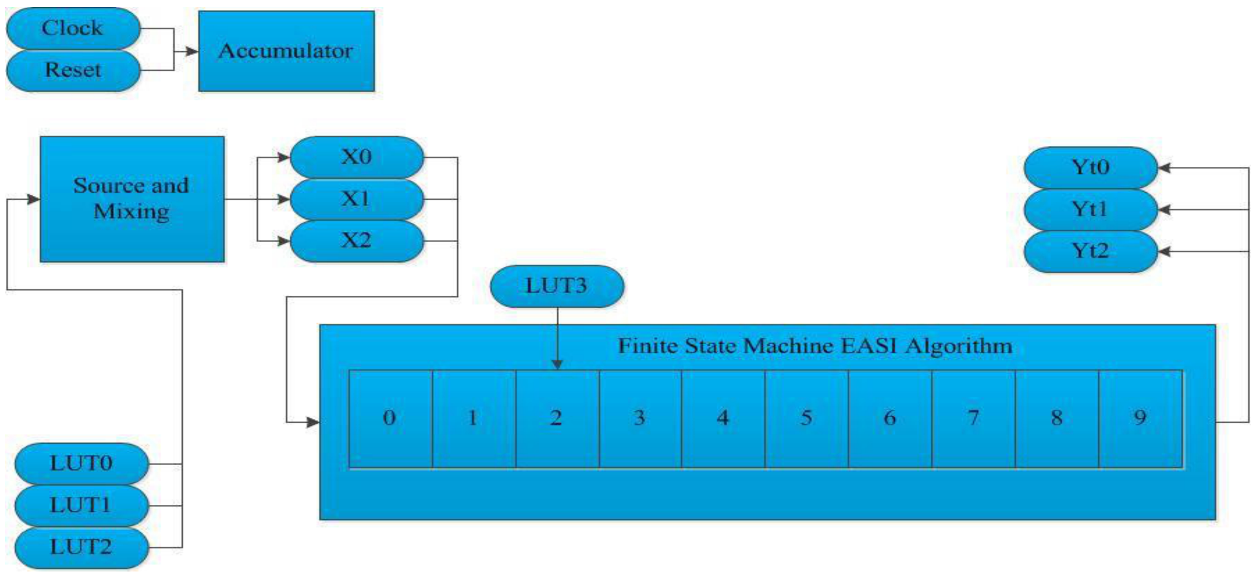

Figure 5.4.2.2: Diagram of complete EASI ICA Test Bench.

The full implementation of the EASI algorithm contained the components seen in Table 5.4.2.

Table 5.4.2 Components of the Full EASI Implementation

Table 5.4.2: Components of the Full EASI Implementation	
Component	Purpose
Clock 50	50 MHz clock for the simulation
Reset	Used to clear out the registers and reset the accumulator to zero.

Table 5.4.2: Components of the Full EASI Implementation Continued	
Component	Purpose
Accumulator	Increments the loading inputs from the Look Up Tables
S0, S1 & S2	Each array holds information from the look up table for each source
X0, X1 & X2	Mixed signals arrays
LUT 0, 1, 2, 3	Look up tables for the three independent signals and Nonlinearity signal
VV 0, VV1, VV2	Index References for Nonlinearity
V0, V1, V2	Nonlinearity Values
Ykykt, fykt, ykft	Matrix values for H matrix arithmetic
H	H matrix
muHB, HB1, HB2	Matrix values for B matrix arithmetic
Btemp,Bmat	Temporary matrix and B matrix
Yt, ytout0, ytout1, ytout2	The output values of the Full EASI implementation

The simulation result for the EASI Full Implementation Test Bench are shown in Figure 5.4.2.3.

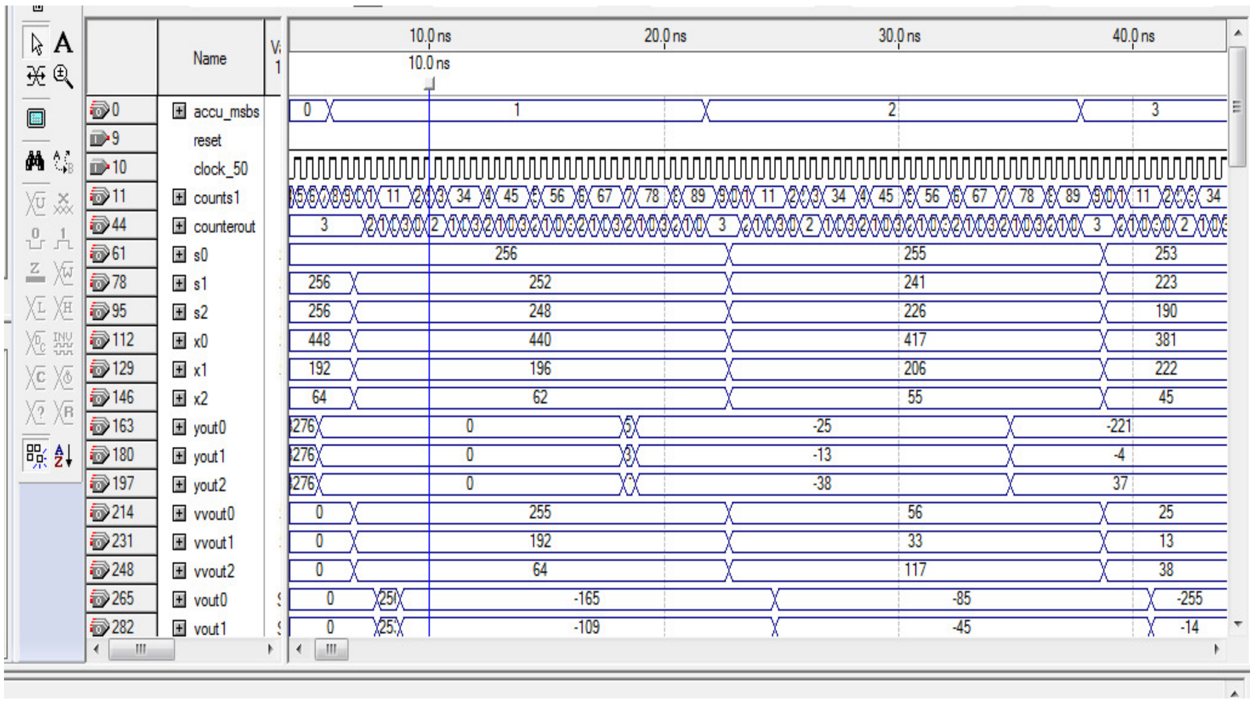

Figure 5.4.2.3: Simulation of the EASI Full Implementation Test Bench.

5.5 Result of the EASI Full Implementation Test Bench

The EASI algorithm was successfully implemented in hardware with an FPGA. The values from the hardware simulation are similar to values seen in the Matlab simulation as shown in Table 5.5.1 and Table 5.5.2.

Table 5.5.1 Comparison of Matlab and VHDL simulation values for the source signals.

Table 5.5.1: Comparison of Matlab and VHDL Data Source Signals									
Matlab Source Signals									
256	255.6	254.7	253.2	251.08	248.3	244.9	241.0	236.5	231.4
256	252.2	241.0	222.7	197.89	167.2	131.6	92.1	49.9	6.2
256	248.3	225.7	189.6	142.22	86.2	25.0	-37.5	-97.9	-152.4
VHDL Source Signals									
256	255	254	253	251	248	245	241	237	231
256	252	241	222	198	167	132	92	50	6
256	248	225	190	142	86	25	-38	-98	-152

Table 5.5.2 Comparison of Matlab and VHDL simulation values for the mixed signals.

Table 5.5.2: Comparison of Matlab and VHDL Data Mixed Signals									
Matlab Mixed Signals									
448	440.32	417.61	380.89	331.77	272.41	205.40	133.61	60.08	-12.11
192	195.56	205.95	222.33	243.35	267.25	291.96	315.27	334.95	348.93
64	61.86	55.67	46.13	34.31	21.63	9.69	0.212	-5.16	-4.94
VHDL Mixed Signals									
448	440	418	381	332	272	205	134	60	-12
192	196	206	222	243	267	292	315	334	349
64	62	56	46	34	22	10	0	-5	-5

There is a variation in the value between the Matlab and Quartus simulations shown in Table 5.5.3 because of the truncation errors and nonlinear value limitations. In Matlab, the actual hyperbolic tangent may be used and the scaling of the function happens through the actual function. In VHDL, a look up table is required. The values are rounded to integer representations, so the values do not exactly correlate completely to values seen in matlab. Truncation errors arise because Matlab uses floating point numbers while integers were used in VHDL

Table 5.5.3 Comparison of Matlab and VHDL simulation values for the separated signals.

Table 5.5.3: Comparison of Matlab and VHDL Data Separated Signals									
Matlab Separated Signals									
361.9	356.6	340.8	314.9	279.8	236.5	186.3	130.6	71.0	9.4
372.2	371.5	369.2	365.4	360.3	354.0	346.6	338.3	329.3	319.7
64	61.8	55.6	46.1	34.3	21.6	9.6	0.2	-5.1	-4.9
VHDL Separated Signals									
256	221	114	124	151	246	66	23	17	12
13	4	3	1	10	4	97	0	-6	-7
238	217	111	117	118	219	30	31	13	15

5.6 Conclusion

The EASI algorithm was successfully implemented in hardware with an FPGA. The values from the hardware simulation are similar to values seen in the Matlab simulation. Truncation errors arose because Matlab uses floating point numbers while integers were used in VHDL. The implementation of the EASI ICA in hardware required 8150 Logic Elements, 120 multipliers, and 24576 Memory bits.

CHAPTER SIX

CONCLUSION

6.1 Summary of the Thesis

The Algebraic ICA, Fast ICA, and Equivariant Adaptive Separation via Independence ICA were examined and compared. The best algorithm required the least complexity and fewest resources while effectively separating mixed sources. The best algorithm chosen was the EASI algorithm. The EASI algorithm was further analyzed by looking and stabling the best fit mixing matrix, nonlinearity, number of resources, and a variation of the algorithm. The algorithm and parameters were set from these results to create the best set up for its implementation on hardware. The EASI ICA was implemented with a Cyclone II FPGA in an Altera Quartus 2 VHDL coding environment. A Source and Mixed Signal Test Bench and a Nonlinearity Implementation Test Bench were developed to ensure the EASI algorithm was correctly implemented in hardware. Simulations were run to ensure the hardware implementation matched the Matlab simulations. The values were similar to the values generated from the Matlab simulation. There were truncation errors present and slight discrepancy of the simulation output values because the hardware implementation utilized integers only, while Matlab values are floating point numbers.

6.2 Future Work

The work done for this thesis can be expanded in a variety of ways. The first would be to add a floating point converter to the EASI ICA implementation. This converter would help to eliminate the number truncation errors. Another extension would be to implementation of the EASI algorithm utilizing the algorithmic variation to compare the effectiveness of both algorithms and see which signals it works best for. An additional extension would be to incorporate the analysis of music and speech signals for real time analysis.

BIBLIOGRAPHY

[1] M. G. Lopez, H. M. Lozano, L. P. Sanchez and L. N. O. Moreno, "Blind Source Separation of Audio Signals Using Independent Component Analysis and Wavelets," Centro de Investigacion en Computaciion-IPN, Escuela Superior de Computo-IPN, Mexico, 2009.

[2] C. Jutten and J. Herault, "Blind Separation of Source, Part I: An Adaptive Algorithm Based on Neuromimetic Architecture," *Signal Processing* , vol. 24, no. 1, pp. 1-10, 1991.

[3] F. Sattar and C. Charayaphan, "Low Cost Design And Implementation of An ICA Based Blind Source Separation Algorithm," School of Electrical and Electronic Engineering, Nanyang Technological University, IEEE, Nanyang, Singapore, 2002.

[4] A. Hyvarinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks,* vol. 13, no. 4-5, pp. 411-430, 2000.

[5] S. Choi, A. Cichocki, H. Park and S.-Y. Lee, "Blind Source Separtion and Independent Component Analysis: A Review," *Neural Information Processing - Letters and Reviews,* vol. 6, no. 1, pp. 1-57, 2005.

[6] K. Torkkola, "Blind Separation of Convolved Sources Based on Information Maximization," Motorola Inc., Phoenix Corporate Research Laboratories, Tempe, AZ, 1996.

[7] L. Yuan and Z. Sun, "A Survey of Using Sign Function to Improve The Performance of EASI Algorithm," in *Proceedings of the 2007 IEEE International Conference on Mechatronic and Automation*, Harbin, China, 2007.

[8] G. X. S. Li, "Adaptive Step-size EASI Algorithm Based on Separating Degree," in *ICWMMN2006 Proceeding*, P.R. China, 2006.

[9] J. Karhunen, E. Oja, L. Wang, R. Vigario and J. Joutsensalo, "A Class of Neural Networks for Independent Component Analysis," *IEEE Transactions on Neural Networks,* vol. 8, no. 3, pp. 486-504, 1997.

[10] K.-K. Shyu and M.-H. Li, "FPGA Implementation of FastICA Based on Floating-Point Arithmetic Design for Real-Time Blind Source Separation," in *2006 International Joint Conference on Neural Networks*, Vancouver, BC, Canada, 2006.

[11] Y. Wei and C. Charoensak, "FPGA Implementation of Non-Iterative ICA for Detecting Motion in Image Sequences," in *Seventh International Conference on Control, Automation, Robotics and Vision*, Singapore, 2002.

[12] J.-F. Cardoso and B. H. Laheld, "Equavariant Adaptive Source Separation," *IEEE Transcations On Signal Processing* , vol. 44, no. 12, pp. 3017-3031, 1996.

[13] H.-M. Park, S.-H. Oh and S.-Y. Lee, "Adaptive Noise Cancelling Based on Independent Component Analysis," *Electronic Letters,* vol. 38 , no. 15, pp. 833-834, 2002.

[14] J. Karhunen, E. Oja, R. Vigario and J. Joutsensalo, "A Class of Neural Networks for Independent Component Analysis," *IEEE Transactions on Neural Networks* , vol. 8, no. 3, pp. 486 - 506, 1997.

[15] Y. Hongwei, Z. Hongmei and B. Wang, "A Novel ICA Algorithm For Two Source," in *International Conference on Signal Processing Proceedings* , Beijing, China, 2004.

[16] K. Waheed and F. Salem, Algebraic Independent Component Analysis: An Approach for Separation of Overcomplete Speech Mixture, East Lansing, MI: Circuits, Systems and Neural Networks Laboratory, IEEE, 2003.

[17] K. Waheed and S. F.M., "Algebraic Overcomplete Independent Component Analysis," in *4th Internationational Symposium on Independent Component Analysis and Blind Signal Separation (ICA 2003)*, Nara, Japan, 2003.

[18] D. Acharya, G. Panda and Y. Lakshmi, "Fixed-point Error Evaluation of Fast ICA and Algebraic ICA Algorithms," in *IEEE International Conference on Industrial Technology, 2006*, Mumbai, India, 2006.

[19] S.-J. Kim, K. Umeno and R. Takahashi, "FPGA Implementation of EASI," *IEICE Electronics Express,* vol. 4, no. 22, pp. 707-711, 2007.

[20] H. Du, H. Qi and X. Wang, "Comparative Study of VLSI Solutions to Independent Component Analysis," *IEEE Transactions on Industrial Electronics,* vol. 54, no. 1, pp. 548-559, 2007.

BIOGRAPHICAL SKETCH

Crispin Odom is from Atlanta, Georgia. She graduated with Honors from the Georgia Institute of Technology with a Bachelor of Science in Electrical Engineering in December 2009. In the Spring of 2012, she will graduate with Master of Science in Electrical Engineering.