國 立 交 通 大 學

電機與控制工程學系

碩 士 論 文

四通道即時 EEG 訊號獨立事件分析之 FPGA 實現

FPGA Implementation of Four-Channel ICA for On-line EEG Signal Separation

研 究 生：黃煒忠

指導教授：林進燈 博士

中華民國 九十七 年 七 月

四通道即時 EEG 訊號獨立事件分析之 FPGA 實現

FPGA Implementation of 4-Channel ICA for On-line EEG Signal Separation

研　究　生：黃煒忠　　　　Student：Wei-Chung Huang

指導教授：林進燈　　　　Advisor : Dr. Chin-Teng Lin

國立交通大學

電機與控制工程學系

碩士論文

A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

July 2008

Hsinchu, Taiwan, Republic of China

中華民國 九十七 年 七月

四通道即時 EEG 訊號獨立事件分析之 FPGA 實現

學生：黃煒忠　　　　　　　　　　　指導教授：林進燈 博士

國立交通大學電機與控制工程研究所

中文摘要

在真實世界的多感應器應用中，如何從混合訊號中分析出獨立訊號的瞎訊號分離是一個常見的問題，例如:音訊和生醫訊號處理。本論文提出一個基於資訊最大化之獨立事件分析方法應用於四通道 EEG 訊號分離。並用定點數實現於 FPGA,再藉由藍芽傳輸分離後的訊號。經由實驗的結果，本論文所提出的硬體方式比軟體運算快 56 倍，且絕對相關係數和離線訊號處理比較至少有 80% 。 最後，實際示範將用 Altera DE2 發展板展示，此設計使用 16605 邏輯單元。

而本論文所提出的四通道即時獨立事件分析系統也加入彈性的介面用於實際 EEG 訊號分離的應用。用資訊最大化演算法的即時生醫訊號分離其取樣頻率設定在 64Hz，並藉由整合性的算術運算架構可讓整體操作速度在 68MHz。

FPGA Implementation of 4-Channel ICA for On-line EEG Signal Separation

Student：Wei-Chung Huang Advisor : Dr. Chin-Teng Lin

Department of Electrical and Control Engineering
National Chiao Tung University

Abstract

Blind source separation of independent sources from their mixtures is a common problem for multi-sensor applications in real world, for example, speech or biomedical signal processing. This thesis presents an independent component analysis (ICA) method with information maximization (Infomax) update applied into 4-channel one-line EEG signal separation. This can be implemented on FPGA with a fixed-point number representation, and then the separated signals are transmitted via Bluetooth. As experimental results, the proposed design is faster 56 times than soft performance, and the correlation coefficients at least 80% with the absolute value are compared with off-line processing results. Finally, live demonstration is shown in the DE2 FPGA board, and the design is consisted of 16,605 logic elements.

The 4-channel On-line ICA accompanied with flexible communication interface for real EEG signal separation has been presented in this thesis. The proposed integrated mathematics architecture can allow high-speed at 68MHz and real-time biomedical signal separation with Infomax ICA at sampling rate 64 Hz.

誌謝

兩年的研究所生涯隨著論文的完成劃上了句號，這兩年間，要感謝許多人的鼓勵和幫忙，使我獲得充實的專業能力並順利完成研究所的學業。
首先要感謝的是我的指導教授-林進燈老師。林老師是國內十分傑出的一位教授，在不同領域內都有相當好的研究成果。感謝老師提供了很理想的研究環境、豐富的資源及正確的引導，使我在研究上非常順利。在老師悉心的指導下，讓我學習到解決問題的能力及做研究應有的態度，使我獲益良多。

在實驗室裡，鍾仁峰博士給予我最直接的教導，不管遇到課業上或研究上的問題，常常去請教鍾仁峰博士，感謝鍾學長不厭其煩地教導，使我增進了對積體電路設計上的專業知識，開拓了我的視野。在學校裡，感謝范倫達教授時常關心我學業上的研究，時常與我討論論文方向及進度。還要特別感謝洪紹航學長對於我研究上的指導，幫解決了我許多的問題。另外也感謝實驗室所有的夥伴，經翔、德瑋、俊傑、靜瑩及智文等學長姐們。還有我的同學們，毓廷、煒忠、建昇、孟修、寓鈞、舒愷、孟哲、俊彥、依伶。以及實驗室的學弟妹，昕展、哲睿、介恩、家欣、有德，感謝大家在研究上及生活上的互相扶持及鼓勵。

最後要感謝家人爸爸、媽媽、妹妹的支持，讓我能專心於學術上的研究，渡過所有難關，謝謝！人生值得感謝的人其實很多，感謝老天、感謝許多親人、朋友和同學，在生命的旅途中，因為有你們，因為我們彼此珍惜、相互扶持，才能有無比的力量。

Contents

List of Figures

List of Tables

Chapter1
Introduction

1-1 Motivation

In recent years, Independent Component Analysis (ICA) has been proved as a powerful algorithm to solve blind source separation (BSS) [1] problems in a variety of signal processing applications such as speech [2], image, or biomedical signal processing. Especially biomedical signals, which are different signal sources from organs such as brain, heart, or muscles, push the ICA algorithm to process more channels than speech or image applications. However, the characteristic of general ICA is limited to only process off-line and enormous data. On clinic, this cannot assist doctors in real-time diagnosis. Thus, more researches focus on on-line and faster ICA from points of view on software or hardware implementation.

The applications of ICA are separation of artifacts in Magnetoencephalography data, finding hidden factors in financial data, reducing noise in natural images, and telecommunications. Another, very different application of ICA is on feature extraction. A fundamental problem in digital signal processing is to find suitable representations for image, audio or other kind of data for tasks like compression and denoising. In On-line ICA application, it can detect the characteristic of biomedical signals immediately by On-line processing and send out the correct response to human. It is helpful to real-time biomedical monitor. Another application, it can used for reduce dimension on high-channels data or extract noise in clamant environment.

Since the large number of matrix operation and complicated non-linear computation are required, it is hard to real-time process in embedded systems. However, the FPGA implementation not only accelerate the speed of the operation circuit by parallel processing, but also show real-time computation and low-power property by fast symmetrical non-linear lookup table.

1-2 Goal and Summary

The Infomax ICA algorithm which is based on the concept of information maximization is designed to solve the problems of blind signal separation (BSS). There are two problems: The algorithm is not suitable for on-line computation, and complicated mathematics operation which make Infomax ICA hard to implement in VLSI. The algorithm has been improved by a new effective hardware and overlap memory scheduling to solve those problems. Finally, the thesis using pipeline flow to increase calculation throughput, and add dynamic branch predict to overlapping memory access time in pipeline.

1-3 Organization of the Thesis

This thesis is organized as follows. The Infomax theory and system level design are introduced in chapter 2 and chapter 3 individually. Chapter 4 describes FPGA implementation of ICA. The experimental results and discussions are presented in Chapter 5, and conclusions are made in the last chapter.

Chapter2
ICA Algorithm

2-1 Basic Concepts of ICA

ICA could be used in different fields, for example: image, audio signal processing, and biomedical data analysis. In this section, the basic concepts of ICA from the viewpoints of signal processing and statistics are introduced.

2-1-1 Problem Description

ICA is created to solve cocktail-party problems in signal processing. There are situations where there are a number of signals produced by some physical sources. These signals could be, for example, electric signals from different brain areas, speech signals from different people speaking in the same room [3], or radio waves from different mobile phones in the same area [4]. The sensors are placed in different positions, so that mixtures are different from one another as a result of space factors.

In practice, the information about the original signals and the mixing system are unknown, and the information of mixed signals from sensors. For this reason, drawing out original signals from those mixtures is professed Blind Source Separation. The BSS problem is illustrated in Fig. 2- 1. ICA is one of the useful methods to precede

BSS problems which separate signals mainly by independence. In the following contents, representations will be called components due to the name of ICA.

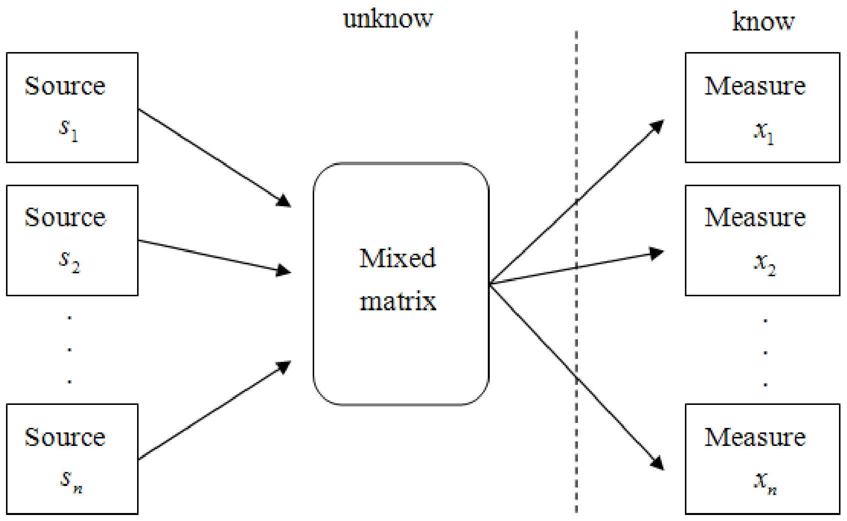


Fig. 2- 1 Illustration of the BSS problem.

ICA looks for components that are both statistically independent and non-Gaussian from mixed data which distinguishes ICA from other BSS methods. Besides, ICA gives good representations of source signals through the linear combination of mixed signals with non-linear decorrelation methods. In practical situations, it is easy to find the components which are really non-Gaussian.

On the other hand, the best de-mixing matrix that makes the components really independent to each other can not be found in general. It should be noted that algorithms exist to make the components as independent as possible.

2-1-2 Formulation

Most of the work on BSS so far addresses the case of mixtures, where a linear mixture model is assumed:

$$x(t) = A \times s(t) \tag{2.1}$$

where s(t) is the vector of sources at instant t, A is the mixing matrix, and the observed vector of mixtures (ignoring noise). ICA now consists of estimating both the matrix A and s when x is the only given signal. It should be noted that the number of independent components $s(t)_i$ equals to the number of observed variables $x(t)_i$ which is a simplifying assumption and is not completely necessary. ICA obtains a $n \times n$ matrix W where

$$\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \cong \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = W \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.2}$$

Above the equation, $s_i(t) \approx y_i(t) = W \times x_i(t)$, y_i is called the representation of sources from the measurements. If is more similar to S, it is a better representation. Fig. 2- 2 shows the formulation of ICA.

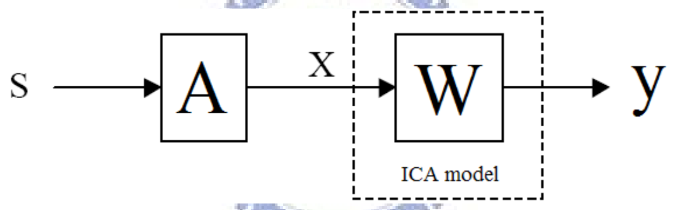


Fig. 2- 2 Illustration of ICA formulation.

2-1-3 Independent Conditions

Sources are assumed to be statistically independent and non-Gaussian in ICA. The condition is a critical technique that makes ICA different from other methods.

According to the central limit theorem (CLT), sum of non-Gaussian random variables are closer to Gaussian than original ones. However, non-Gaussian assumption is also due to a natural disadvantage of ICA. In the statistical point of view, uncorrelated data are independent only when those data are Gaussian. Therefore, if the original independent components are Gaussian, their mixtures must be Gaussian. After mixtures are turned into uncorrelated by pre-processing, they are still Gaussian and already independent. However, those uncorrelated mixtures are always dissimilar to the original independent components. There are two kinds of non-gaussian data: super-Gaussian and sub-Gaussian as illustrations in Fig. 2- 3.

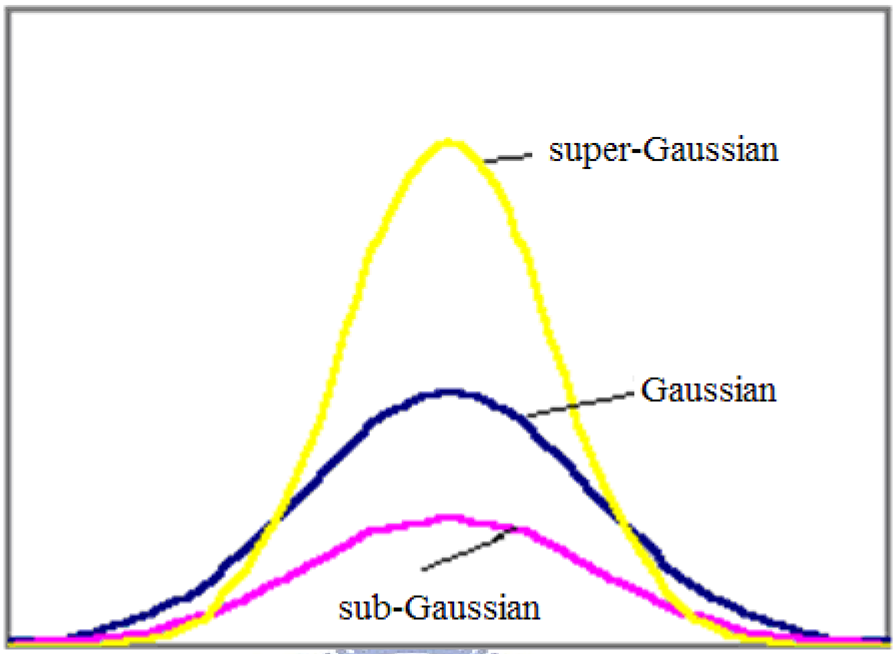


Fig. 2- 3 Illustration of probability density distribution.

2-2 Two kinds of ICA Algorithm

We consider methods for estimating solution to the unknowns in ICA problem. In the simple case, we assume noiseless ICA **x**=A**s** are the elements of the mixing matrix, A, and the sources **s** which we consider as being estimated by a recovered source set **a**.

2-2-1 The Concept of Entropy and Mutual Information

The ICA algorithm assume all of the sources are independent in the module. The M sources together generate an M-dimensional probability density function (p.d.f) p(**s**). Statistical independence between the sources means that the joint source density factorizes as (2.3)

$$\mathrm{p}(\mathbf{s})=\prod_{m=1}^{M} p(s_m(t)) \qquad (2.3)$$

If the p.d.f of the estimated sources also factorizes then the recovered sources are independent and the separation has been successful. Independence between the recovered sources is measured by their mutual information, which is defined in terms of entropies.The entropy of an M-dimensional random variable **x** with p.d.f p(**x**) is

$$H[\mathbf{x}]=H[p(\mathbf{x})]=-\sum p(x)\log p(x). \tag{2.4}$$

The entropy measures the average amount of information that observation of **x** yields. The joint entropy H[**x,y**] of two random variable **x** and **y** is defined as:

$$H[\mathbf{x},\mathbf{y}]=H[p(\mathbf{x},\mathbf{y})]=-\sum p(x,y)\log p(x,y), \tag{2.5}$$

where p(x,y) is the Joint probability density of variable **x** and **y**. We can consider the entropy of x and y as a set. In Fig. 2- 4, joint entropy H[**x,y**] is continuum of H[**x**] and H[**y**], for $H[x,y] \le H[x]+H[y]$, if x and y are independent, $H[x,y]=H[x]+H[y]$.

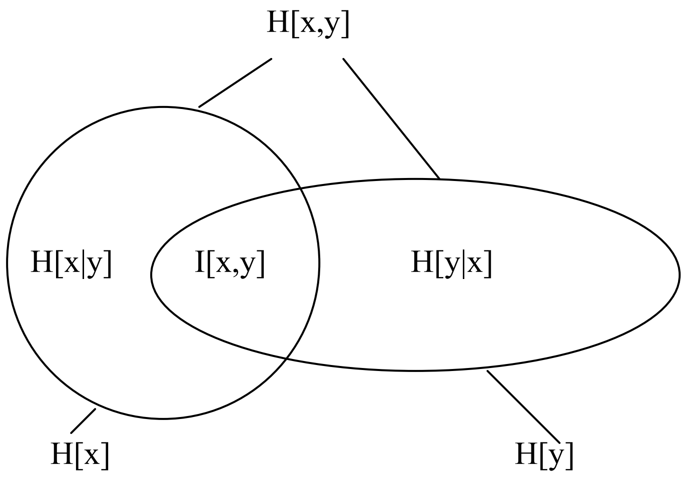


Fig. 2- 4 Entropy relationship by the concept of set.

The conditional entropy of y given x is

$$H[\mathbf{y}|\mathbf{x}]=-\sum p(y\mid x)\log p(y\mid x). \tag{2.6}$$

Conditional entropy H[**y**|**x**] is entropy H[**y**] without H[**x**]. From which it follow that

$$\begin{aligned} H[\mathbf{x},\mathbf{y}] &= H[\mathbf{x}]+ H[\mathbf{y}|\mathbf{x}] \\ &= H[\mathbf{y}]+ H[\mathbf{x}|\mathbf{y}]. \end{aligned} \tag{2.7}$$

The above equation means that the sum of the information encoded by **x** alone and the information encoded by **y** given a knowledge of **x**. The entropy of each variable is related to probability of observations.

The mutual information between two random variables x and y is defined in terms of their entropies as follow (2.8):

$$\begin{aligned} I[x,y] &= H[x]+H[y]-H[x,y] \\ &= H[x]-H[x|y] \\ &= H[y]-H[y|x]. \end{aligned} \tag{2.8}$$

From the equation above, the mutual information contains the sum of the entropy of each variables and the difference of the joint entropy of all variables.

Using concept of mutual information, an m-dimensional random variable y_i, i=1…n, the mutual information of all variable is defined as:

$$I(y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} H(y_i) - H(y). \tag{2.9}$$

Where H is entropy, H(y) is the joint entropy $H(y_1, y_2, \ldots, y_n)$ of variable y_i, the value of mutual information always positive or zero. If and only if the value is zero, each variable is independent. If the target is finding the minimum mutual information between each variable, it is equal to find the direction of non-Gaussian distribution. Next section, we will describe two methods of ICA: Information maximization ICA and FastICA.

2-2-2 Infomax ICA

(1)Information maximization

Bell and Sejnowski [5] proposed to learn the separating matrix W by minimizing the mutual information between components of y(t) = g(u(t)) , where g is a nonlinear function approximating the cumulative density function (cdf) of the sources. Bell &

Sejnowski formulated blind source separation algorithms in terms of information maximization.

Consider the information transmitted by mapping **f**: $\mathbf{x} \rightarrow \mathbf{y}$. K.Torkkola [6] consider the two-stage mapping, which might be implemented by a single layer feed-forward neural network in Fig. 2- 5, as follows:

$$u=W*x, \tag{2.10}$$

$$y=g(u), \tag{2.11}$$

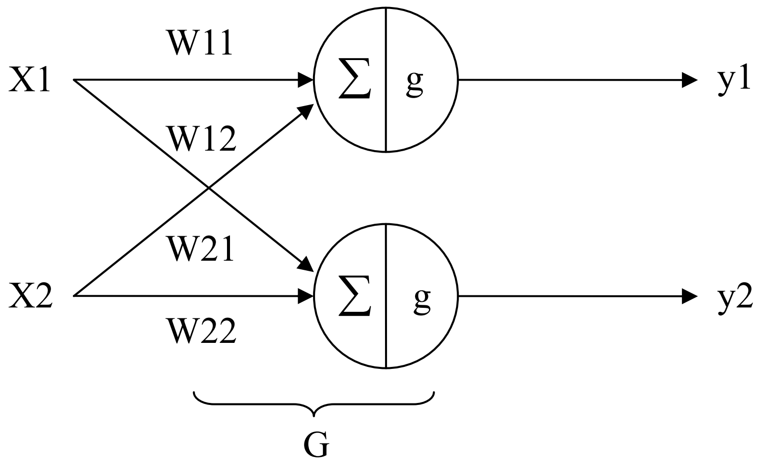


Fig. 2- 5 Blind separation network architectures for two-source mixtures.

where W is a linear transformation and **g** is a bounded nonlinearity applied to each individual output **u**. The information transmitted by the mapping is the mutual information between the input and output:

$$\begin{aligned} \mathbf{I[x,y]} &= \mathbf{H[x]+H[y]-H[x,y]} \\ &= \mathbf{H[y]-H[y|x]}, \end{aligned} \tag{2.12}$$

where **H[y]** is the entropy of the output, while **H[y|x]** is whatever entropy the output has which didn't come from the input. In the case that we have no noise, the mapping between x and y is deterministic and **H[y|x]** has its lowest possible value. This divergence is one of the consequences of the generalization of information theory to continuous variables. In order to reduce complexities, the algorithm consider only the gradient of information theoretic quantities with respect to some parameter, w, in the network.

Since gradients are as well behaved as discrete variable entropies, the reference terms involved in the definition of differential entropies disappear. The above equation can be differentiated as follows, with respect to a parameter, w, involved in the mapping from x to y:

$$\frac{\partial}{\partial w} I(x,y) = \frac{\partial}{\partial w} H(y), \tag{2.13}$$

H[y|x] does not depend on w, and lead $\frac{\partial}{\partial w} H(y \mid x) = 0$. Thus for invertible continuous deterministic mappings, the mutual information between inputs and outputs can be maximized by maximizing the entropy of the outputs alone.

When a single input x pass through a transforming function g(x) to give an output variable y, both I(y| x) and H(y) are maximized when we align high density parts of the probability density function of x with highly sloping parts of the function g(x). This is the idea of "matching a neuron's input-output function to the expected distribution of signals".

From another point of view, thus $I(y_1, y_2, ..., y_n) = \sum_{i=1}^{n} H(y_i) - H(y)$, to minimize mutual information to each outputs y_i existence when $H(y) = H(y_{1,} y_2, ... y_n)$, and $I(y_1, y_2, ..., y_n) = 0$. Output y_i is independent. In order to let mutual information of outputs be zero, need to satisfy below situation:

1. The choice of non-linear function g(.) is crucial.

$$y = g(u) = \frac{1}{1+e^{-u}} \qquad u = Wx. \tag{2.14}$$

2. According to maximum entropy theorem, if bounded variable with

uniform distribution has the maximum entropy. y_i must between 0~1 because of it is the p.d.f of independent component u_i. Therefore in order to maximize the entropy of output, we need output y uniform distribution

The output y have to independent to satisfy both of situation. Although the transformation between y and u is a monotonic transform, information maximization using this concept to achieve the target of ICA.

(2)Gradient information

Consider a network with an input vector x, a weight matrix W, a bias vector w_0 and a nonlinearly transformed output vector y=g(u), u= Wx+ w_0.

Providing **W** is a square matrix and **g** is an invertible function, the multivariate probability density function of y can be written

$$P(y) = \frac{P(x)}{|J|}, \tag{2.15}$$

where |J| is the absolute value of the Jacobian of the transformation. Bell simplifies to the product of the determinant of the weight matrix and the

derivative y_i', of the outputs, y_i, with respect to their net inputs:

$$J = (\det W)\prod_{i=1}^{n} y_i'. \tag{2.16}$$

For example, in the case where the nonlinearity is the logistic sigmoid:

$$y = g(u) = \frac{1}{1+e^{-u}} \quad \text{and} \quad y' = \frac{\partial y}{\partial u} = y(1-y). \tag{2.17}$$

We can perform gradient ascent in the information that the outputs transmit about inputs by noting that the information gradient is the same as the entropy gradient for invertible deterministic mappings. The joint entropy of the outputs is:

$$H(y) = -E[\ln P(y)] = E[\ln |J|] - E[\ln P(x)]. \tag{2.18}$$

Weights can be adjusted to maximize H(y). As before, they only affect the E[ln |J|] term above:

$$\Delta W \alpha \frac{\partial H(y)}{\partial W} = \frac{\partial}{\partial W}\ln|J| = \frac{\partial}{\partial W}\ln|\det W| + \frac{\partial}{\partial W}\ln\prod_{i=1}^{n}|y_i^{'}|. \qquad (2.19)$$

For the full weight matrix, we use the definition of the inverse of a matrix, and the fact that the adjoint matrix, adj W, is the transpose of the matrix of cofactors. This gives:

$$\frac{\partial}{\partial W}\ln|\det W| = [W^T]^{-1}. \qquad (2.20)$$

For the second term, we note that the product splits up into a sum of log-terms, only one of which depends on a particular w.

$$\frac{\partial}{\partial W}\ln\prod_{i=1}^{n}|y_i^{'}| = \frac{\partial}{\partial W}\ln\prod_{i=1}^{n}|\frac{\partial y}{\partial x}| = \prod_{i=1}^{n}(\frac{\partial y}{\partial x})^{-1}\frac{\partial}{\partial W}(\frac{\partial y}{\partial x}) = (1-2y)x^T. \qquad (2.22)$$

The resulting learning rules are familiar in form:

$$\Delta W \propto [W^T]^{-1} + (1-2y)x^T. \qquad (2.23)$$

Except that now x, y, W, and1, are vectors (1 is a vector of ones). But this learning rule is too complex to calculate because of the inverter matrix. Multiplied by W^TW change the rescale of the rule, the new learning rules as follow:

$$\Delta W = (I + (1-2y)u^T)W = (I + \varphi(u)u^T)W. \qquad (2.24)$$

Thus, the simplification much uncomplicated than before, and this learning rules is suitable to separate blind sources. An ICA model consists of two distinct components, the first is the formulation of a valid contrast function and second is the algorithm for estimating the free parameters of the system.

Considering approaches which rely on the gradient of the contrast to ascend or descend to an extreme contrast measure. It is computationally attractive to have access, to the analytic form for the gradient of the contract function with respect to the free parameters. The bias of the contrast function to be that of a generative model approach, Gradient-ascent, or steepest-gradient, methods require this first order information and update W in direction of the gradient. The update rule for W in discrete time t<-t+1 defined in equation as follows:

$$W(t+1) = W(t) + l\Delta W\,. \tag{2.25}$$

where l is the adaptation parameter (learning rate) which is fixed this procedure corresponds to maximum likelihood re-estimation with an exponential weighting over successive samples. Note that this learning rule describes an online learning procedure because data are processed sequentially as they are received. Gradient ascent to the likelihood for a batch of T observation is performed with the modified rule

$$W(t+1) = W(t) + l(I + \frac{1}{T}\sum_{t=1}^{T}\varphi(t)u^{T}(t))\,. \tag{2.26}$$

Since the learning rule (2.26) is obtained from (2.25) by dropping the averaging operation it is sometimes called stochastic gradient ascent. The use of steepest-gradient techniques to ascend the likelihood to near its supremum was formulated by Bell & Sejnowski [1995]. One of the key problems is their poor convergence in region of shallow gradient and in regions where the likelihood landscape is far from isotropic. To overcome some of these issues, Bell & Sejnowski utilized batching, whereby the mean gradient over a set of consecutive samples is utilized rather than the sample by sample estimate.

2-2-3 FAST ICA

After we have defined a measure of non-gaussian, we have to develop a practical method for maximizing it. The basic method used in this kind of problems is the gradient method. However, FastICA is based on a fixed-point iteration scheme for finding a maximum of the non-gaussian of $W^{T}x$. More rigorously, it can be derived as an approximative Newton iteration. The FastICA algorithm using negentropy combines the superior algorithmic properties resulting from the fixed-point iteration with preferable statistical properties due to negentropy.

(1)Fixed-point algorithm

In this subsection, we will introduce Fixed-point algorithm. Fixed-point iteration would be equate W to the gradient measure of non-gaussian. This is because if W equals this gradient, then due to normalization to unit norm. This suggests the following fixed-point iteration:

$$W < -\varepsilon[xg(W^T x)]. \qquad (2.27)$$

The iteration does not have good convergence properties. Therefore, the iteration has to be modified. Multiplied by constant α and add W on both side of this equation as below:

$$(1+\alpha)W = \varepsilon[xg(W^T x)] + \alpha W. \qquad (2.28)$$

Thus, by choosing α wisely, it may possible to obtain an algorithm that convergence very fast.

The suitable coefficient α, and thus the FastICA algorithm, can be found using an approximative Newton method [7]. The Newton method is a powerful method for solving equations. When it is applied to the gradient, it gives optimization method that usually converges in a small number of steps. The problem with the Newton method is that it usually requires a matrix inversion at every step. Therefore, the total computational load may not be smaller than with gradient methods. This approximative Newton method gives a fixed-point algorithm of the form.

To derive the approximative Newton method, first note that the maxima of the approximative of the negentropy of $W^T x$ are obtained at certain optima of $\varepsilon[G(W^T x)]$. According to Kuhn-Tucker conditions, the optima of $\varepsilon[G(W^T x)]$ under the constraint $\varepsilon[(W^T x)^2] = \|W\|^2 = 1$ are obtained at points where

$$\varepsilon[g(W^T x)] + \beta W = 0\,, \qquad (2.29)$$

where β is some constant. Denoting the function on the left-hand side by F, we obtain its Jacobian matrix JF(w) as

$$JF(W) = \varepsilon[xx^T g'(W^T x)] + \beta I\,. \qquad (2.30)$$

To simplify the inversion of this matrix, we decide to approximate the first term. Since the data is sphered, a reasonable approximation seem to be

$$\varepsilon[xx^T g'(W^T x)] \approx \varepsilon[xx^T]\varepsilon[g'(W^T x)] = \varepsilon[g'(W^T x)]I\,. \qquad (2.31)$$

Thus the Jacobian matrix becomes diagonal, and can easily be inverted. Thus, obtain approximative Newton iteration:

$$W = W - \varepsilon[xg(W^T x)] + \beta W \Big/ \varepsilon[g'(W^T x)] + \beta\,. \qquad (2.32)$$

This algorithm can be further simplified by multiplying both sides by $JF(W) = \varepsilon[g'(W^T x)] + \beta$. After straightforward algebraic simplification

$$W < -\varepsilon[xg(W^T x) - \varepsilon[g'(W^T x)]W]\,. \qquad (2.33)$$

This is the basic fixed-point iteration in FastICA.

(2)Estimating several Independent Components

The key point to estimate more than one independent component is based on the following property: the vectors W_i corresponding to different independent components are orthogonal in whitened space. Thus, to estimate several independent components, we need to run any of the above one unit algorithm using several units with weight vectors, and to prevent different vectors from converging to the same maxima we must orthogonalize the vectors after every iteration. We present in the following different methods for achieving decorrelation.

A simple way of orthogonalization is deflationary orthogonalization using the Gram-Schmidt method. This means that we estimation the independent components one by one. When we have estimated p independent components, we run any one unit algorithm for W_{p+1}, and after every iteration step subtract from W_{p+1} the 'projections' $(W_p^T W_j)W_j, j=1,...p$, of the previously estimated p vectors, and then renormalize W_{p+1}. The result FastICA algorithm [8] with deflationary orthogonalization is show in Table 2- 1.

Table 2- 1 FastICA algorithm with deflationary orthogonalization

Step	Description
1.	Center the measured data x to make its mean zero.
2.	Whiten the zero-mean data to give x.
3.	Set counter p=1. Set m equals to the number of sources.
4.	Choose an initial value of unit norm for W_p randomly.
5.	Let $W_p = \varepsilon[xg(W_p{}^T x)] - \varepsilon[g'(W_p{}^T x)]W$, where g is defined nonlinearity function.
6.	$W_p = W_p - \sum_{j=1}^{p-1} (W_p{}^T W_j)]W_j$.
7.	Let $W_p = W_p / \|W_p\|$.
8.	If W_p has not converged, go back to 5.
9.	Set $p \leftarrow p+1$. If p = m, go back to step 4.

In certain application, it may be desirable to use a symmetric decorrelation, in which no vectors are 'privileged' over others. This means that the vectors W_i are not estimated one by one, instead, they are estimated in parallel. One motivation for this is that the deflationary method has the drawback that estimation errors in the first vectors are accumulated in the subsequent ones by the orthogonalization. This is done by first doing the iteration step of one unit algorithm on every vector W_i, and afterwards orthogonalization all the W_i by special symmetric methods.

The symmetric orthogonalization of W can be accomplished by the classical method involving matrix square roots,

$$W = (WW^T)^{-1/2}W \tag{2.34}$$

The inverse square root is obtain the eigenvalue decomposition,

$$(WW^T)^{-\frac{1}{2}} = Ediag(d_1^{-1/2},...d_m^{-1/2})E^T \tag{2.34}$$

Using the former symmetric orthogonalization, we give the correspond version of the FastICA algorithm in Table 2- 2.

Table 2- 2 FastICA algorithm

Step	Description
1.	Center the measured data x to make its mean zero.
2.	Whiten the zero-mean data to give x.
3.	Choose m, the number of independent components to estimate.
4.	Choose initial values for the W_i each of unit norm.
5.	For every i=1⋯m, $W_i = \varepsilon[xg(W_i^T x)] - \varepsilon[g^{'}(W_i^T x)]W$, where g is defined nonlinearity function
6.	Do a symmetric orthogonalization of the matrix by $W \leftarrow (WW^T)^{-1/2}W$
7.	If not converged, go back to step 5.

2-3 Main Structure of ICA Methods

Base on the algorithm of off-line ICA flow, in order to achieve real-time calculation, we divide the input of mixed sources to process real-time signals individually, and overlap the previous mixed sources to calculate new separated signals.

2-3-1 The Choice of ICA Algorithm

We consider two reasons feasibility and complexity of real time to realize on-line ICA from two algorithms above. In order to reach real time, we need to choose a low complicated computation and suitable property for the separation of super Gaussian signals. The reason for why we choose infomax ICA is that the flexible transmission and unnecessary preprocessing, it can reduce complicated calculation when real-time.

Though the infomax ICA constringency is harder than FastICA, we add a decrease parameter with growing data to accelerate the convergence. In next section, we will propose a solution to fix the order of weight in real time processing.

2-3-2 Solution of on-line ICA

Since the original blind source separation algorithm by Jutten and Herault [9], several on-line and batch mode algorithms have been formulated under the umbrella of independent component analysis. While some of the batch ICA algorithms such as JADE [10] and FastICA [11] give relatively fast convergence in estimating W, they are not quite suitable for on-line implementation in a real-time setting. Depend on Gradient information learning rules, the weight need update at each new division. In order to fix the direction of weight are the same in every division, overlapping previous mixed sources will fix the order of the weight. The method of data-stream processing illustrate in Fig. 2- 6.

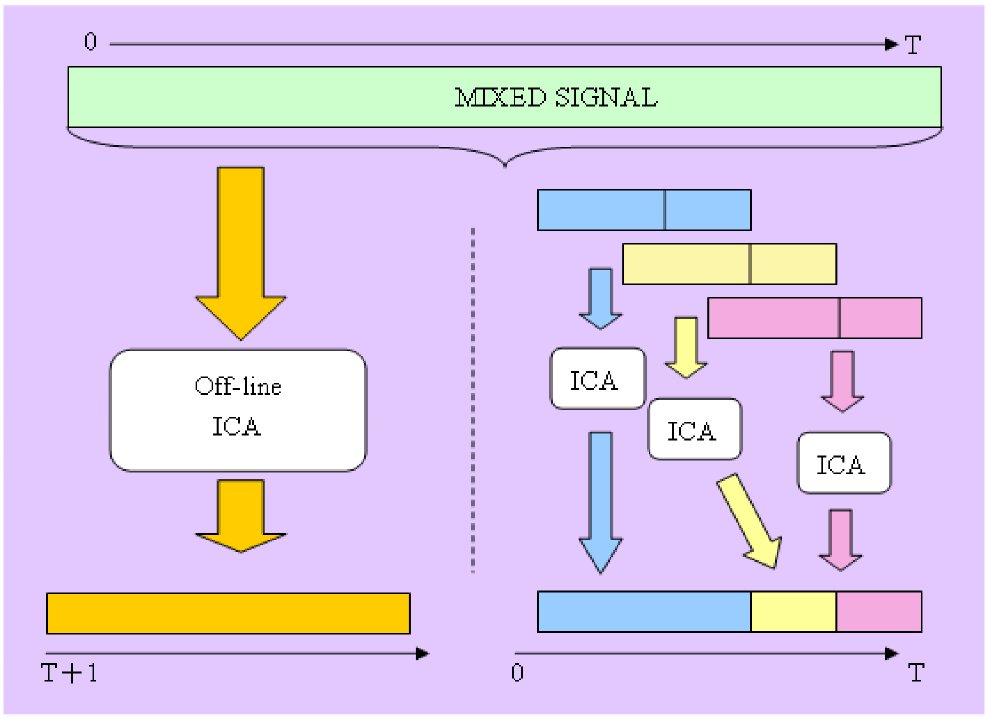
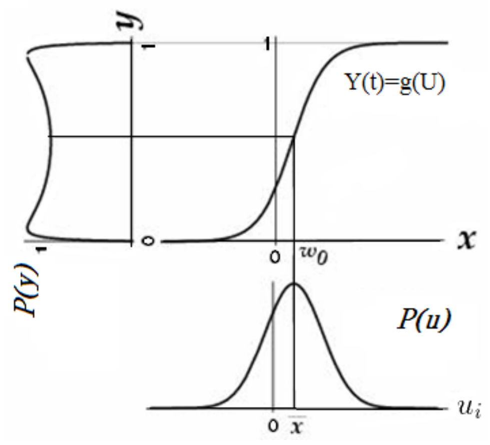


Fig. 2- 6 Illustration of off-line and on-line algorithm.

Fig. 2- 7 show the ICA MODEL, X(t) are measure data. After preprocessing, the model will enter the main calculation unit, including non-linear transformation, and gradient information update. However, in on-line processing of data stream, each division data stream will flow through the ICA model, then, each time the weight be calculated for each division data. Briefly, we take eight seconds division data to processing, and updating two seconds because of the overlap are six seconds. Finally, if the weight is stable, Y(t) represent the independent components which is the product of input X(t) and weight matrix.

ICA MODEL:



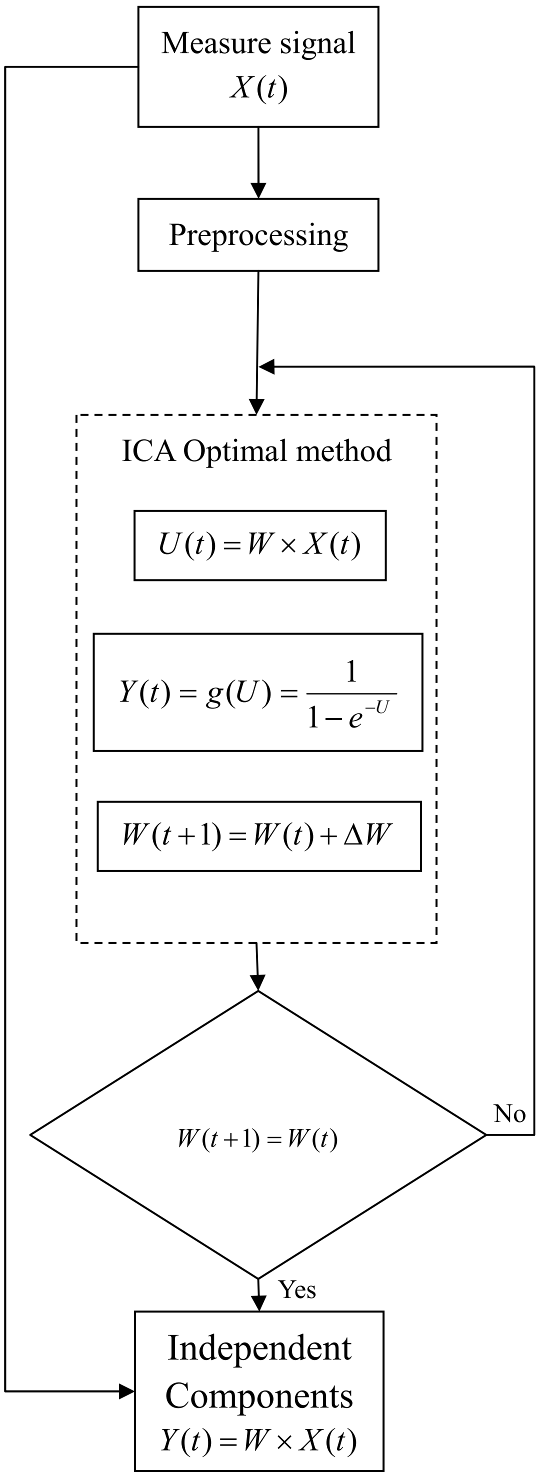


Fig. 2- 7 Diagram flow of the computation of implementation of on-line ICA learning algorithm.

Chapter3
System Architecture Design and Simulation Result

3-1 System Architecture

In this chapter, we will discuss the execution flow of real time algorithm in software. With the system level simulation [12] and data stream flow, we can set specifications of overall system architecture, e.g. the resolution of input, the core speed, and the flexible interface.

3-1-1 Computing Flow

Before setting specifications, we need to analyze the process of data stream in system. First, set the sample rate for 64Hz. In the system of algorithm, we put 512 points data into ICA model with growing data. The 128 points are the set of result because of the old data overlap. Illustrate the system level process with Fig. 3- 1.

However, we discuss the algorithm for on-line execution in MATLAB. In software simulation, we measure the weight update and memory access time by profile command that records information about once recursive time. Table 3- 1 shows the detail of measurement. Total recorded time means the average execution time in ones iteration. And Fig. 3- 2 show execution time with some test data and average execution time at last. The iteration includes two parts: weight update calculation and memory access. We can find that 86 percent of the total time is Weight update calculation.

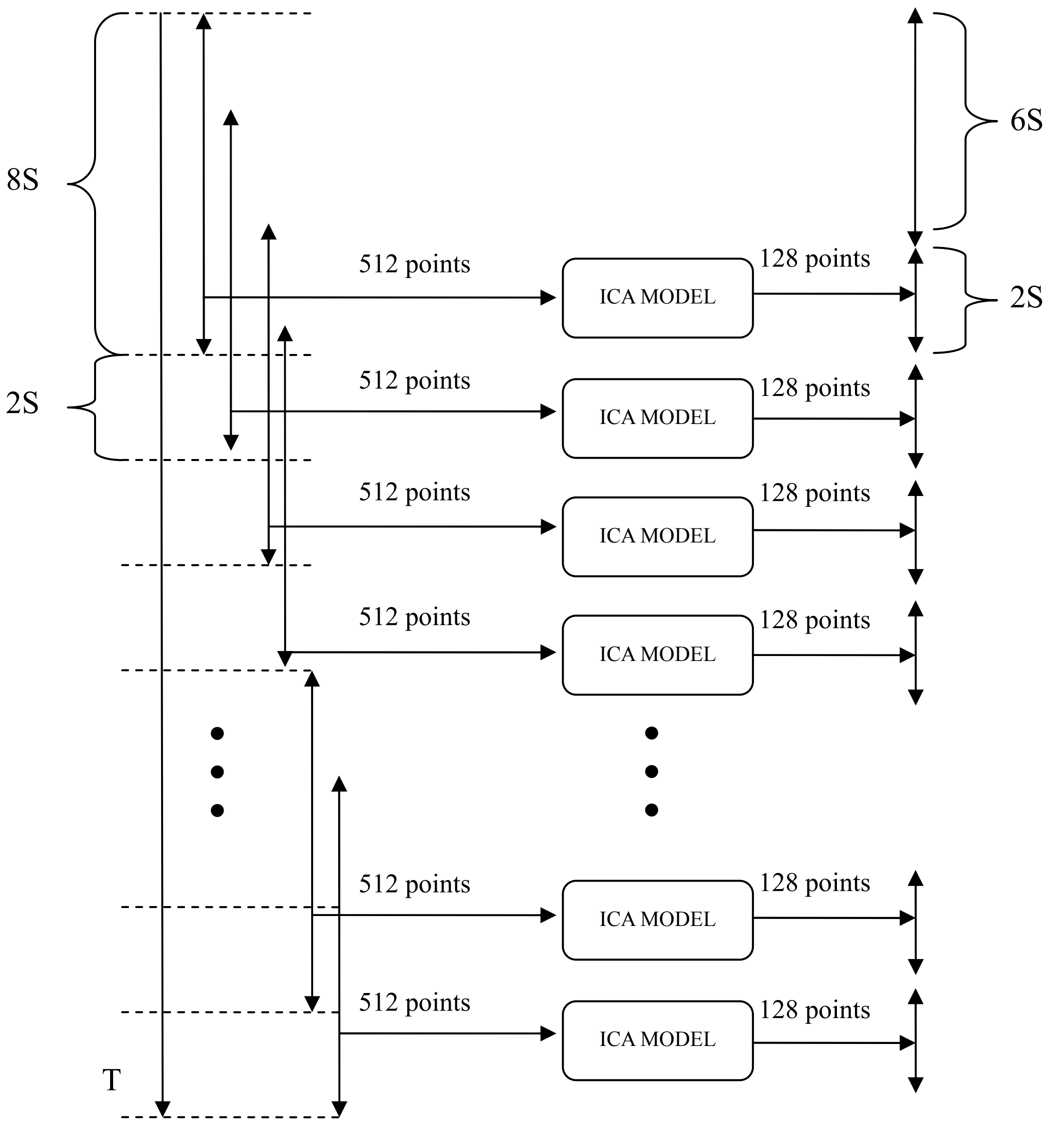


Fig. 3- 1 Illustrate of time process in on-line ICA.

Table 3- 1 Matlab profile

Total recorded time	8.30956 ms
Clock precision	0.00000006 s
Clock Speed	1650 MHz

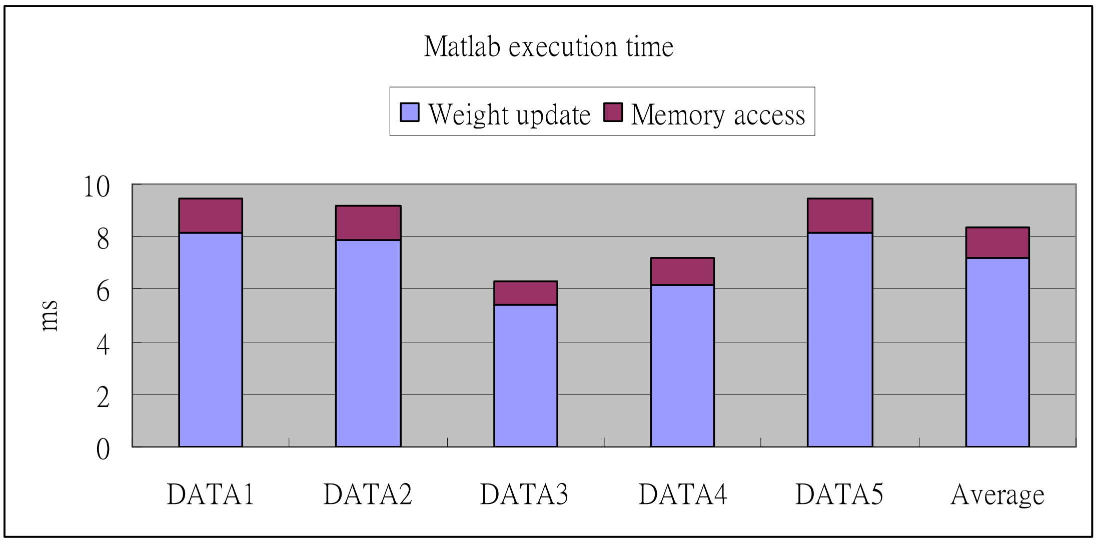


Fig. 3- 2 Matlab execution time.

On the other hand, if the sample rate is 64Hz and the training needs 10 times, we can not finish the training in 16ms. As a result, the software execution time is not fast enough to achieve on-line processing. So we develop a suitable hardware for on-line ICA by FPGA.

3-1-2 Specification of On-line Process

In this system scheme, we need to formulate three parts: the resolution of the input signal, expected core speed, and the core speed. As a result of the quantity of four channels transmission and the restrictions of bus, the resolution of the input signal set as 8 bit. Since the memory bandwidths are 32 bits, it will read four channel data at the same time and it is efficient for memory controller design. While in the data stream, ICA model will process eight seconds data, it means if the sample rate is 64 Hz, the model can process 512 points once, and next iteration, the new two second data combines with old six second data into the ICA model. In order to achieve real-time execution, the ICA model should finish the eight seconds data iteration before the next point into memory. Since it can be derived the required speed of overall system. We set the sample rate is 64 Hz because the frequency of the signal which we analyze less than 32Hz. Most of the cerebral signal observed in the scalp EEG falls in the range of 1-20 Hz. So the required speed of overall system derived as follow:

$$core\_speed = sample\_rate \times train\_loop \times (w\_calculate + converge\_decision) \quad (3.1)$$

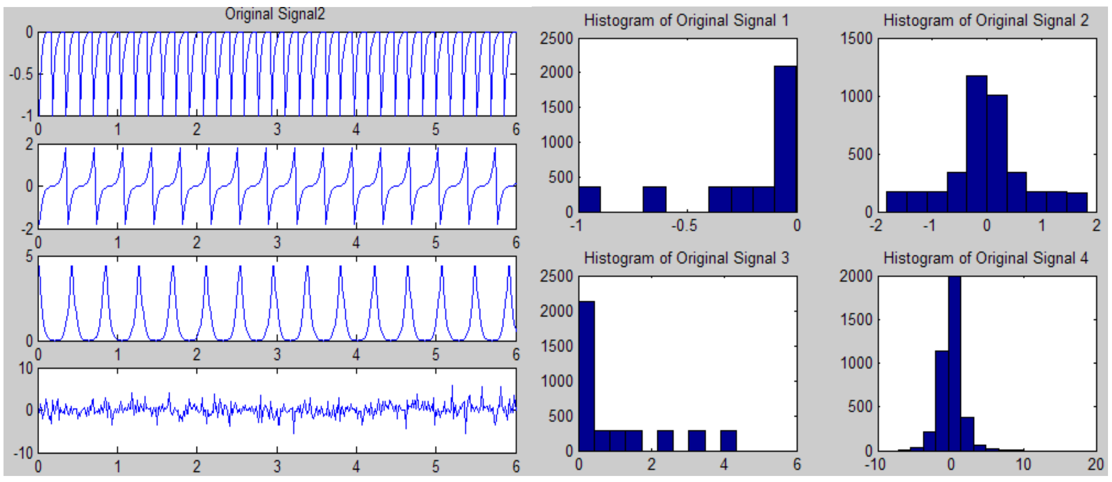
In equation (3.1), $train\_loop$ represent the maximum number of times of weight update. If the number of times of weight update greater than maximum value, the iteration would stop training. The $w\_calculate$ represent the consumption of clock cycles in one iteration. And the $\cov erage\_decision$ represent the number of clock cycles which calculate the difference between new weight and old weight for the decision of weight coverage. In the situation we defined, the core speed should be at least 68MHz for on-line execution.

3-2 Comparison with off-line Sup-Gaussian BSS Methods

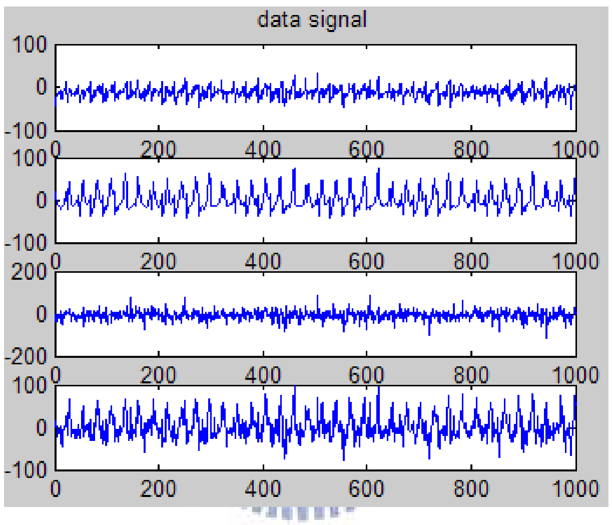
This section introduces the on-line system simulation of Independent Component Analysis. In order to verify the algorithm is suit to on-line process, we will use MATLAB to simulate the accuracy of on-line ICA behavior, and, define the specification and resolution of real circuit architecture. Then implement the arithmetic operation circuit and memory control circuit by the real-time processing simulation.

3-2-1 Simulation 8-bit Super Gaussian Mixed Pattern 1

In verification of MATLB, we create four original signals which p.d.f histogram are super Gaussian with 8 bit resolution and 64Hz sample rate. In real case, we mixed these four signals with a linear mixed matrix, and, thus the mixed four signals are the inputs of real-time ICA model in Fig. 3- 3.



(a)

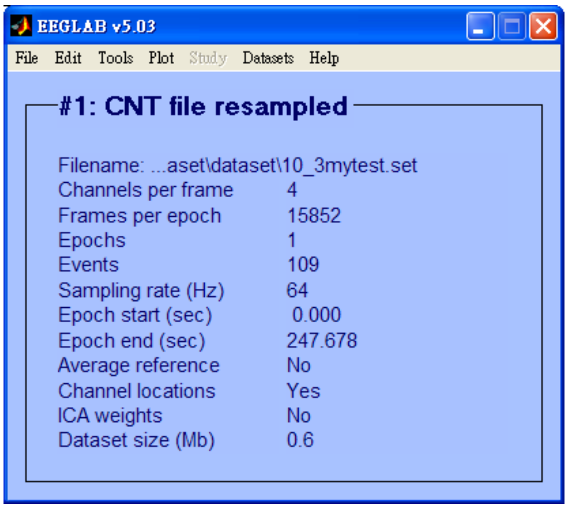


(b)

Fig. 3- 3 (a) original signal and p.d.f of original signal (b) mixed signal.

(1)Time-domain Comparison

In order to verify the algorithm, we compare with the EEGLAB which developed by UCSD in Fig. 3- 4. EEGLAB is an interactive Matlab toolbox for processing continuous and event-related EEG, MEG and other electrophysiological data using independent component analysis (ICA). EEGLAB provides an interactive graphic user interface (GUI) allowing users to flexibly and interactively process their high-density EEG and other dynamic brain data using independent component analysis (ICA) and/or time/frequency analysis (TFA), as well as standard averaging methods. The algorithm of EEGLAB is suit to off-line process, but we just compare time domain and frequency domain result.



#1: CNT file resampled

Filename: ...aset\dataset\10_3mytest.set	
Channels per frame	4
Frames per epoch	15852
Epochs	1
Events	109
Sampling rate (Hz)	64
Epoch start (sec)	0.000
Epoch end (sec)	247.678
Average reference	No
Channel locations	Yes
ICA weights	No
Dataset size (Mb)	0.6

Fig. 3- 4 Off-line EEGLAB toolbox.

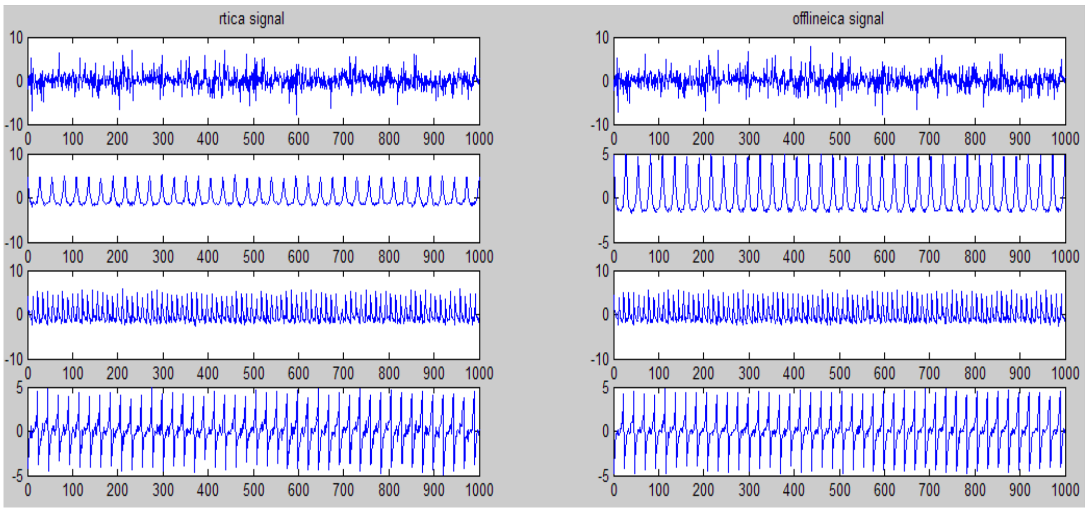


Fig. 3- 5 Time-domain Comparison of on-line and off-line algorithm.

See above Fig. 3- 5, although the order of each result may different, and the amplitude may reverse, but the characteristic of output are the same. The result accords with restriction of the ICA algorithm.

(2)Correlation coefficients Comparison

The result of correlation coefficients between off-line and real-time algorithm in

Fig. 3- 6, it means the relation between each output signal. The negative value means each output of the first channel are reverse, and the value approach one means that these two signals are equivalent.

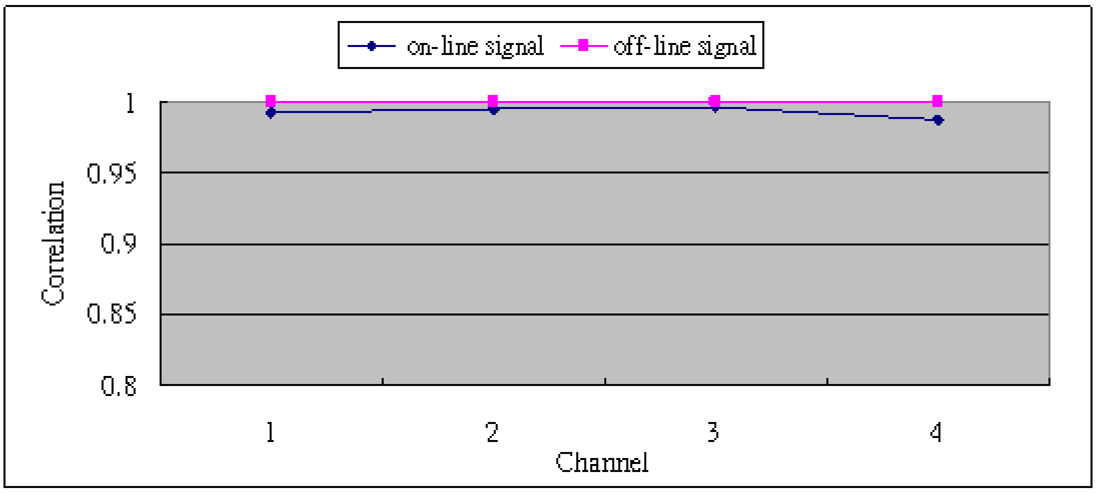


Fig. 3- 6 Correlation coefficients Comparison of on-line and off-line algorithm.

(3)Frequency-domain Comparison

In this part, insure outputs from real-time algorithm are similar to off-line processing, we compare the Fast Fourier transform (FFT) of each output in Fig. 3- 7.

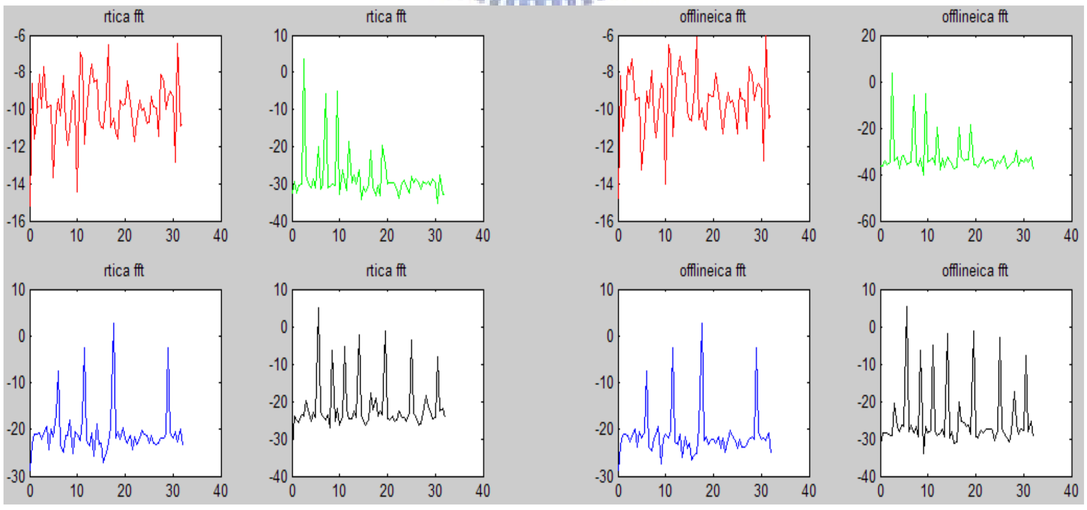


Fig. 3- 7 FFT comparison of on-line and off-line algorithm.

(4)Time-Frequency Comparison

Besides, Fig. 3- 8 shows the time-frequency between off-line and real-time algorithm. Time-frequency: frequency power spectral density estimate on a single channel. The horizontal axis is time domain and vertical axis is frequency domain. The colors represent the power spectral.

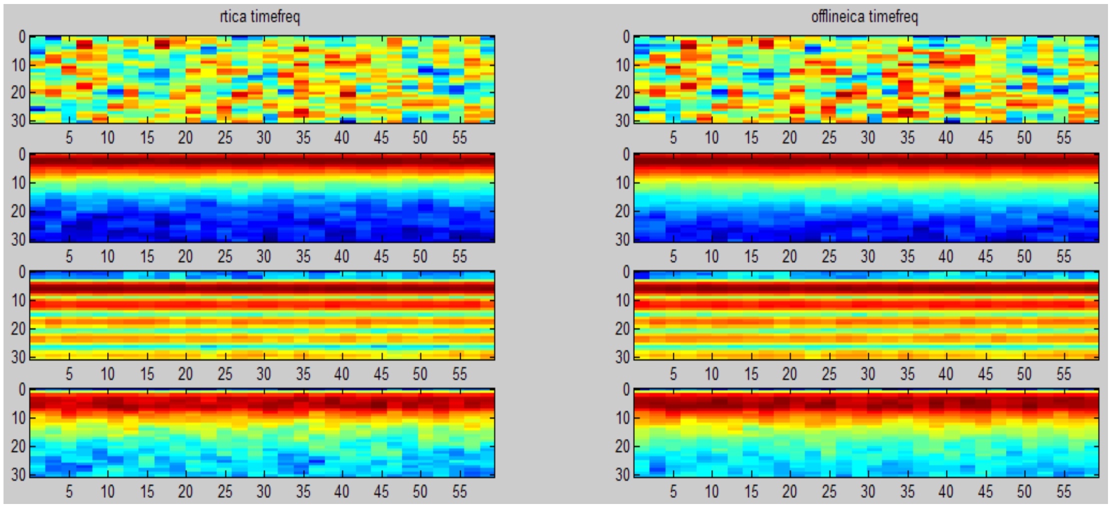
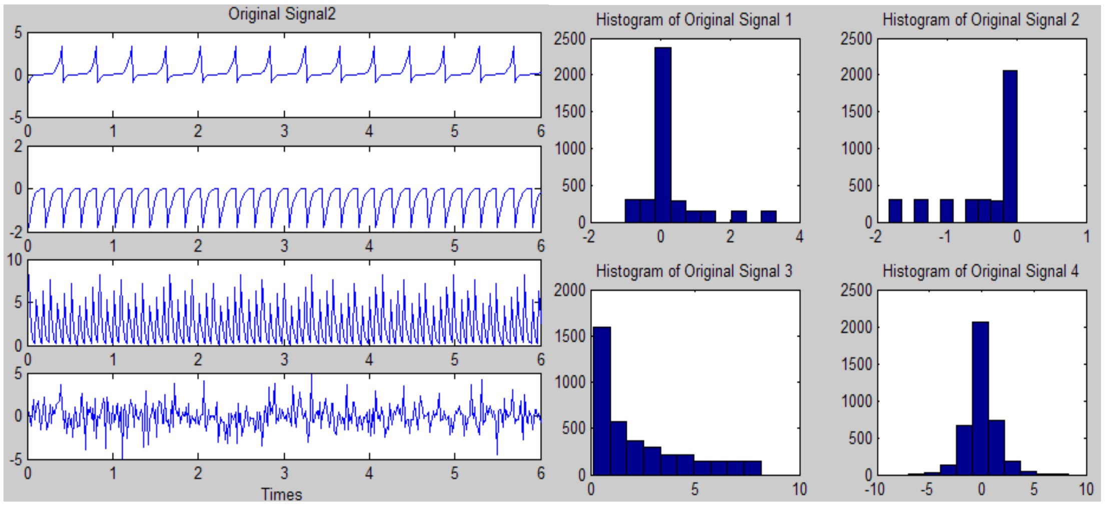


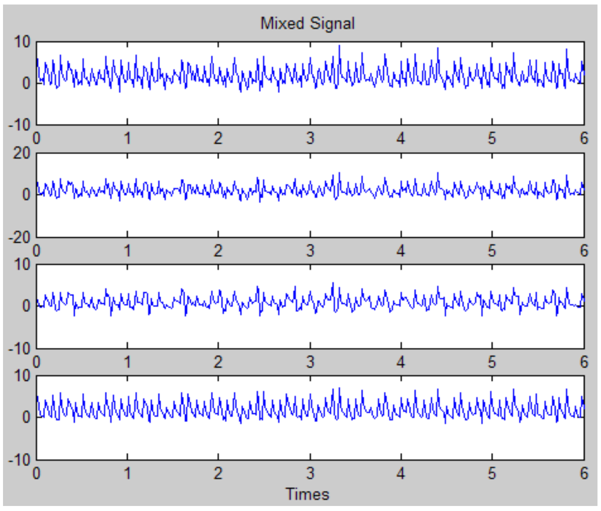
Fig. 3- 8 Time-Frequency Comparison of on-line and off-line algorithm.

3-2-2 Simulation 8-bit Super Gaussian Mixed Pattern 2

The same as section 3-2-1, this section compare with another super Gaussian which has the same frequency like EEG about at 5Hz , 12Hz in Fig. 3- 9. We also compare with off-line tool box contains time-domain, frequency-domain, time-frequency, and correlation coefficients.



(a)



(b)

Fig. 3- 9 (a) original signal and p.d.f of original signal (b) mixed signal.

(1)Time-domain Comparison

As Fig. 3- 10 shows, although the amplitude may reverse, the characteristic of output are still the same. The result accords with restriction of the ICA algorithm.

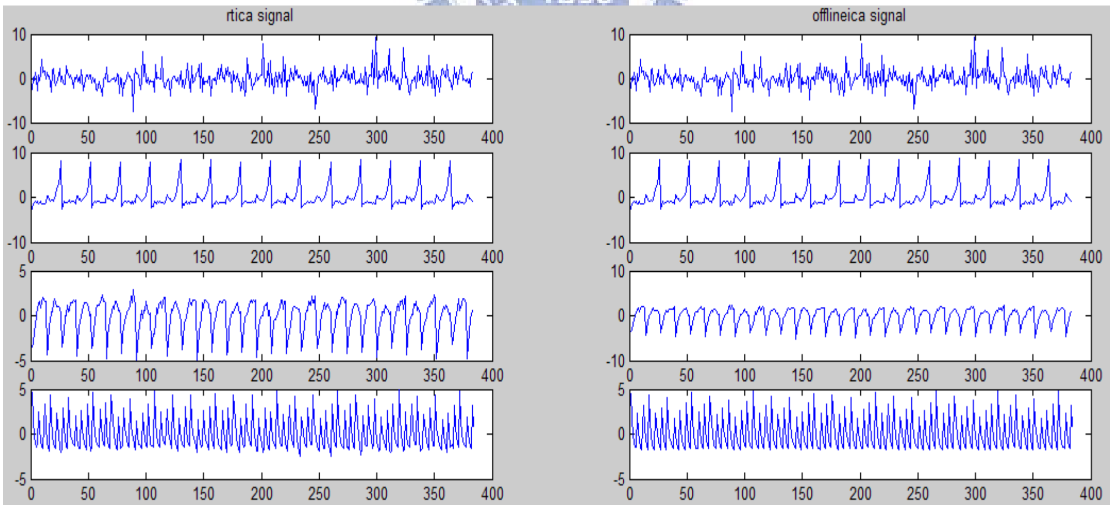


Fig. 3- 10 Time-domain Comparison of on-line and off-line algorithm.

(2)Correlation coefficients Comparison

Though the kurtosis of original signal are large than three, it can be regarded as pure super Gaussian. The result of correlation coefficients between off-line and real-time algorithm are high in Fig. 3- 11. Software simulation has reached the target.

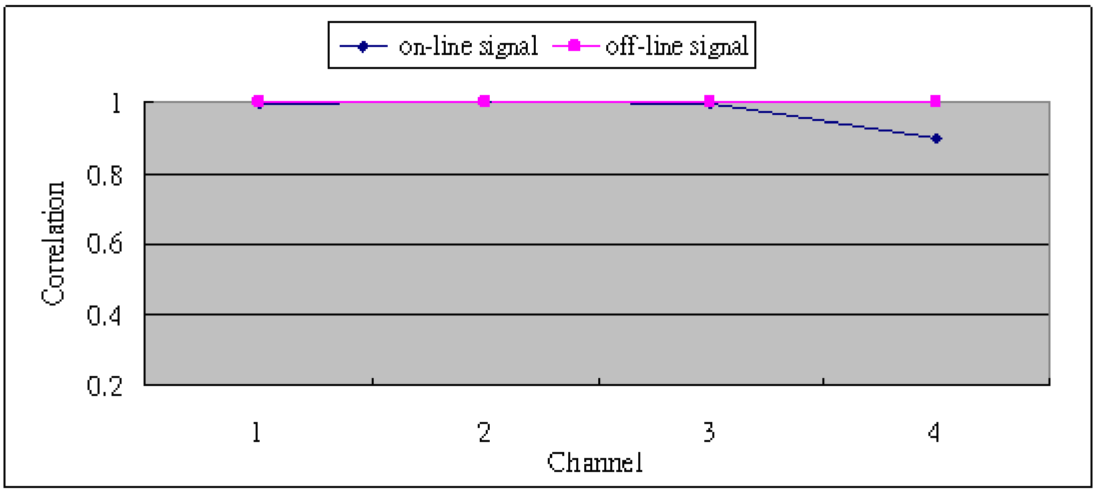


Fig. 3- 11 Correlation coefficients Comparison of on-line and off-line algorithm.

(3)Frequency-domain Comparison

We create four signals that two of them were 5 Hz, 12Hz like EEG frequency as Fig. 3- 12 shown. Alpha is the frequency range from 8 Hz to 12 Hz. It is brought out by closing the eyes and by relaxation. Theta is the frequency range from 4 Hz to 7 Hz. Theta is seen normally in young children. It may be seen in drowsiness or arousal in older children and adults. From the result of third and fourth channel, there have significant power at 5 Hz, 12Hz indeed.

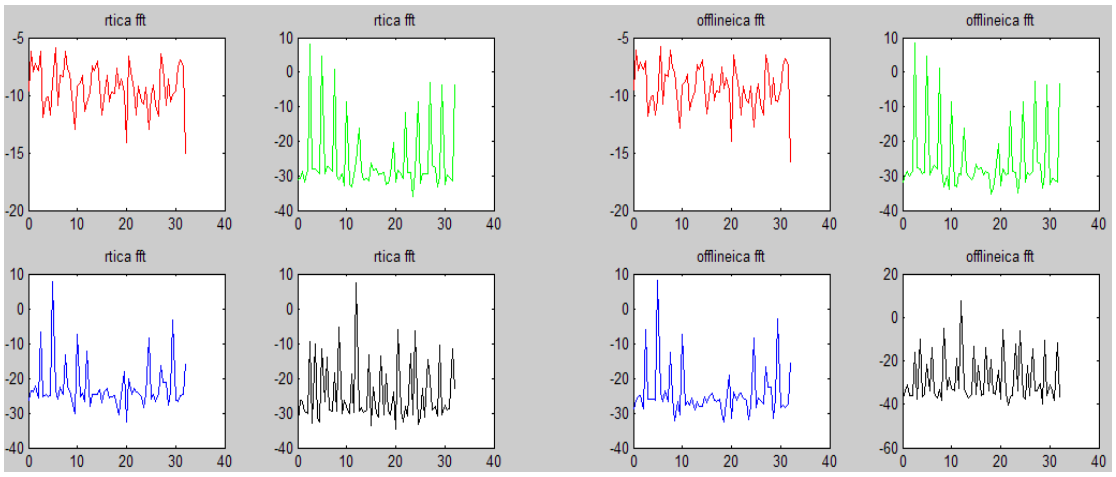


Fig. 3- 12 FFT comparison of on-line and off-line algorithm.

(4)Time-Frequency Comparison

Fig. 3- 13 shows the time-frequency between off-line and real-time algorithm. The same, second and fourth channel in time-frequency there have significant power with red color.

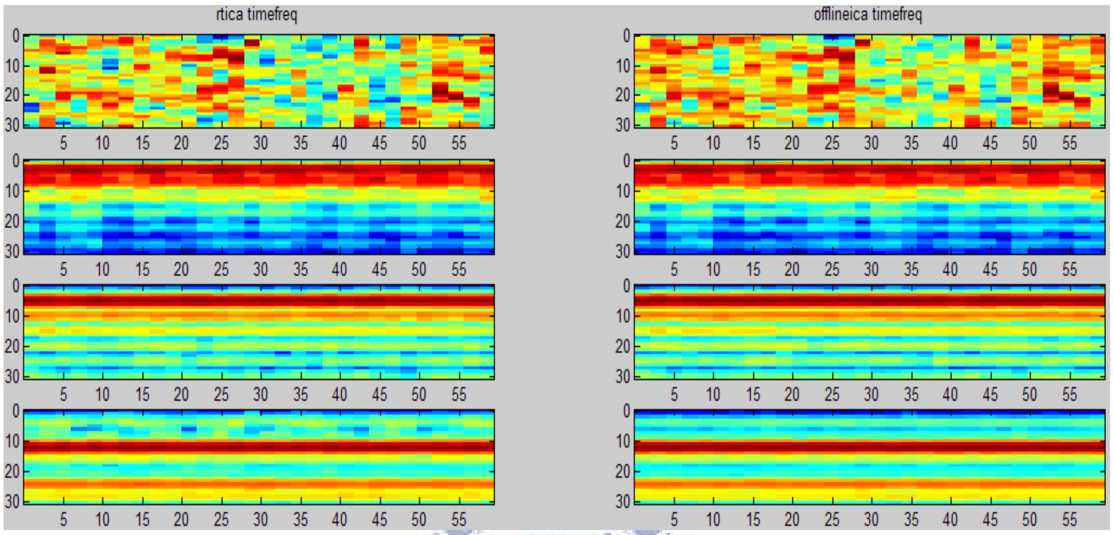


Fig. 3- 13 Time-Frequency Comparison of on-line and off-line algorithm.

3-3 Comparison with off-line EEG BSS Methods

The real EEG signals are given by NCTU BRC, which are recorded in real environment. The row data type with 8 bit resolution and 64Hz sample rate.

3-3-1 Simulation 8-bit EEG Mixed Pattern 1

In real environment, we can create pure EEG signals, and can not mix them as a mixed signal like Fig. 3- 14. We can analyze the measured signal as mixed signals directly.

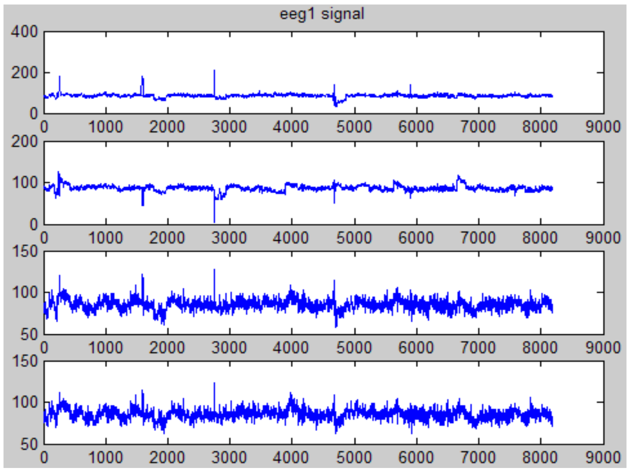


Fig. 3- 14 Mixed EEG signal.

(1)Time-domain Comparison

In real environment, EEG signals may have *EOG*, Alpha, Beta, Theta etc. In order to analyze pure signal, we need to separate each signal by ICA algorithm. Fig. 3- 15 shows the compare of off-line and real-time algorithm.

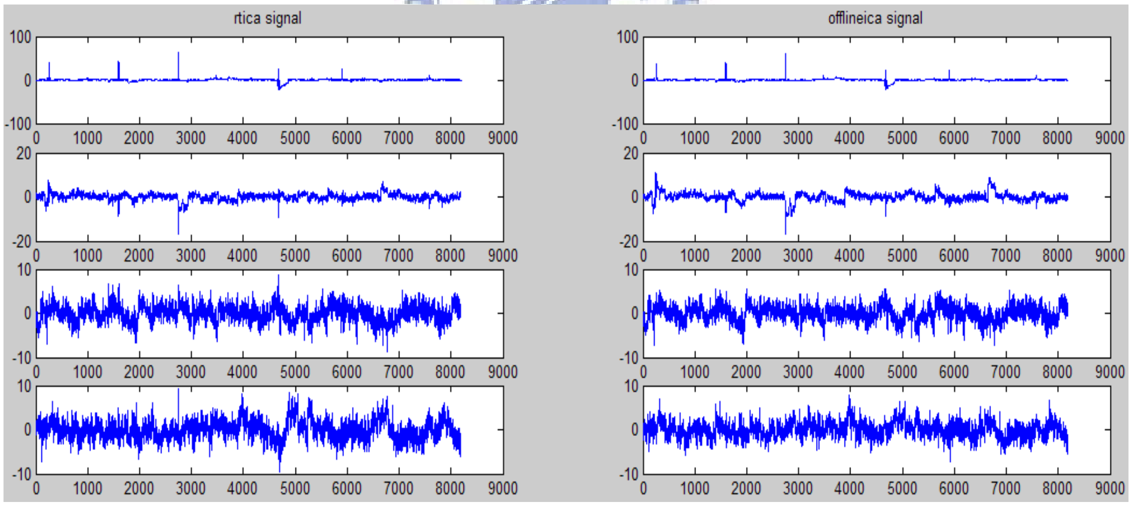


Fig. 3- 15 Time-domain Comparison of on-line and off-line algorithm.

(2)Correlation coefficients Comparison

The Correlation coefficients of EEG signals show below, it shows the highest Correlation coefficient is up to 99% compare with off-line algorithm. But in real environment with unknown noises may cause reduction of Correlation coefficients. According to the result of Correlation coefficient in Fig. 3- 16, the correlation at least 80% compare with off-line is a good method for real-time.

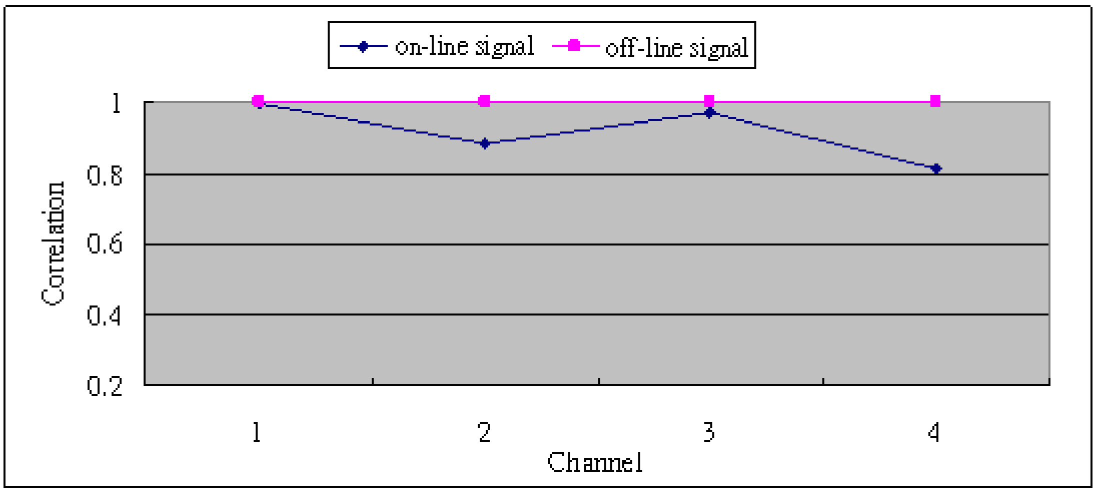


Fig. 3- 16 Correlation coefficients Comparison of on-line and off-line algorithm.

(3)Frequency-domain Comparison

In this part, we compare the Fast Fourier transform (FFT) of each output. Apparently the second and fourth channel of each algorithm with a little different in Fig. 3- 17, and the frequency aliasing is caused by the less information with real-time process.

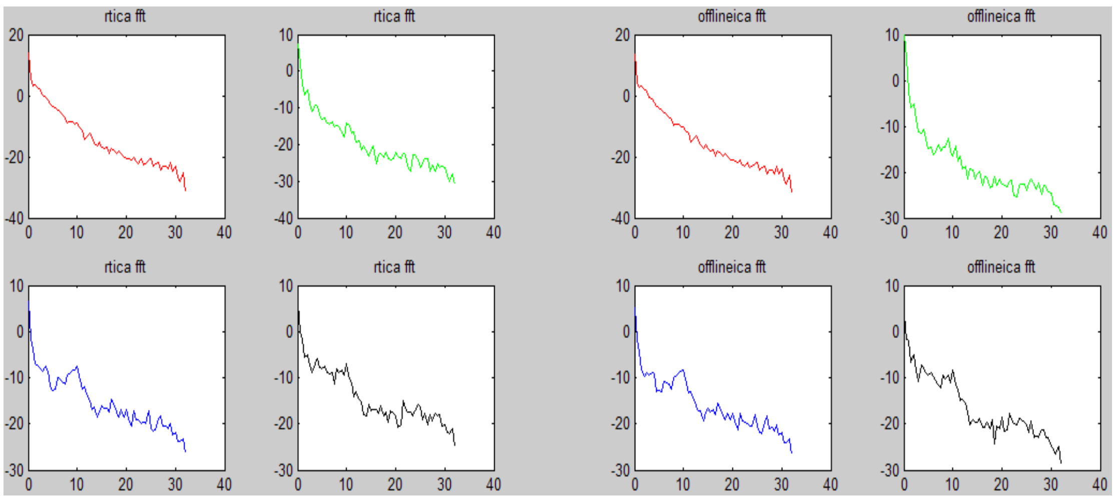


Fig. 3- 17 FFT comparison of on-line and off-line algorithm.

(4)Time-Frequency Comparison

See Fig. 3- 18, the time-frequency show the frequency power spectral with little difference at second and fourth channel. However, we can find from Fig. 3- 18, the third channel with a significant power spectral at 10Hz.

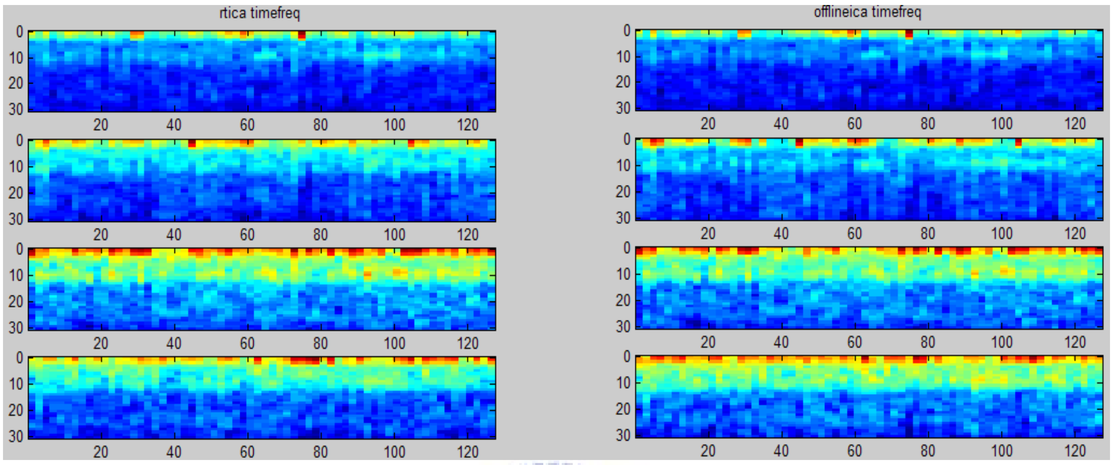


Fig. 3- 18 Time-Frequency Comparison of on-line and off-line algorithm.

3-3-2 Simulation 8-bit EEG Mixed Pattern 2

The second pattern shows in Fig. 3- 19. The characteristic of pattern output are like each other. This is because the sensors are located closely, the signals we measure will be very much like.

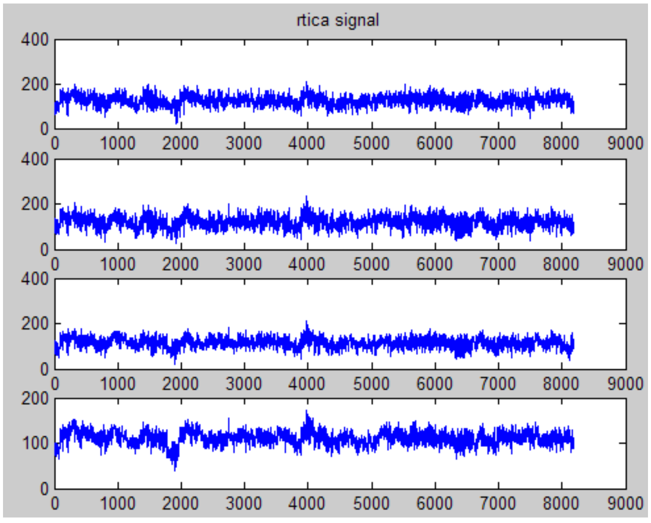


Fig. 3- 19 Mixed EEG signal2.

(1)Time-domain Comparison

The same as section 3-3-1, this section discuss the different between on-line and off-line in real environment. The result of ICA with more noise than before, because the sensors are located closely in Fig. 3- 20.

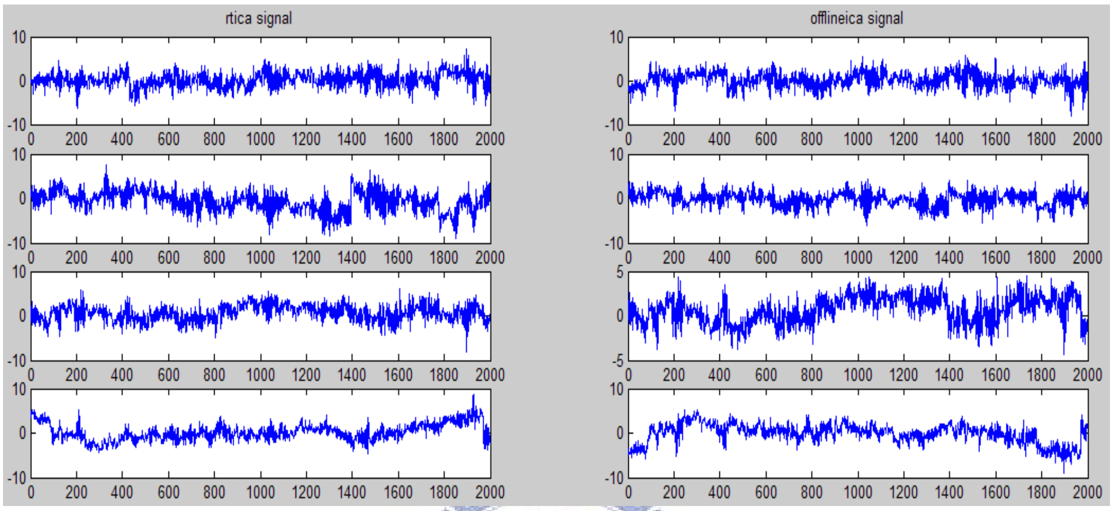


Fig. 3- 20 Time-domain Comparison of on-line and off-line algorithm.

(2)Correlation coefficients Comparison

The correlation coefficients of second pattern are lower than before in Fig. 3- 21, but at least 80% in absolute value is also an acceptable method for real-time.

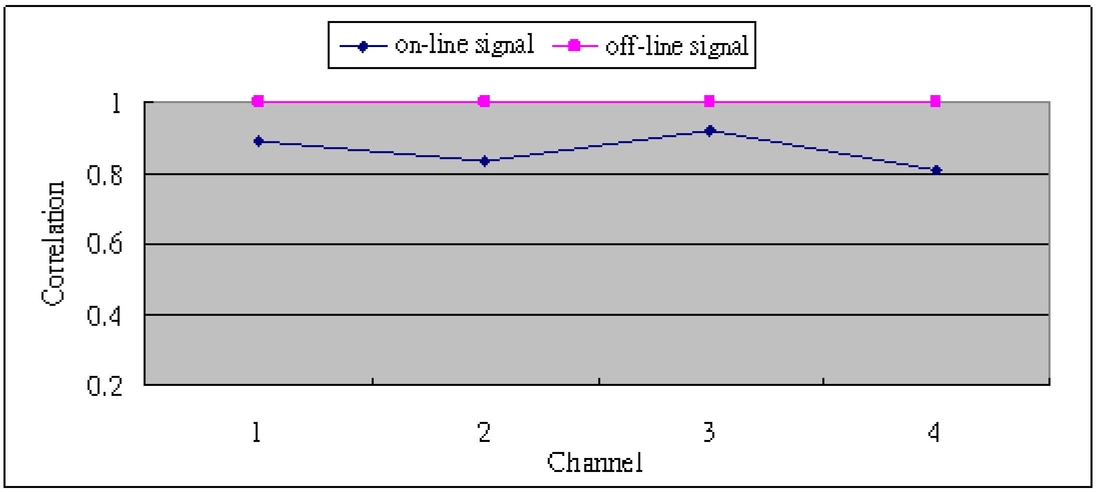


Fig. 3- 21 Correlation coefficients Comparison of on-line and off-line algorithm.

(3)Frequency-domain Comparison

Fig. 3- 22 shows the FFT result comparison. We can see the result of the fourth channel with more error than others. But there is still more than 80 percent of the correlation.

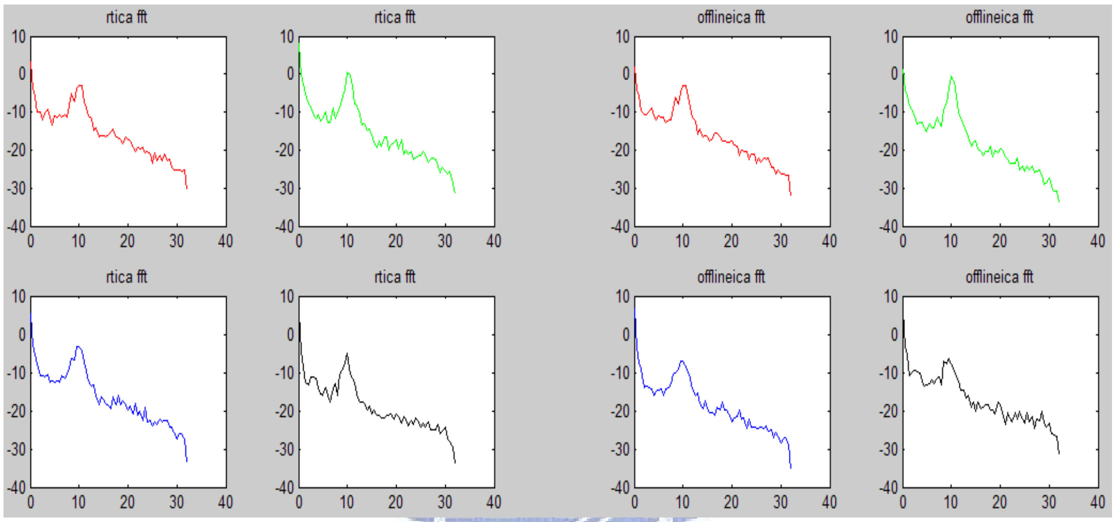


Fig. 3- 22 FFT comparison of on-line and off-line algorithm.

(4)Frequency-domain Comparison

Fig. 3- 23 shows that the first and second channel has more information at 10 Hz than third and fourth channel. Besides, Alpha is the frequency range from 8 Hz to 12 Hz. This is activity in the 8-12 Hz range seen in the posterior regions of the head on both sides, being higher in amplitude on the dominant side. It is brought out by closing the eyes and by relaxation. As a result, it is more helpful to analyze the measured signal than without ICA process.

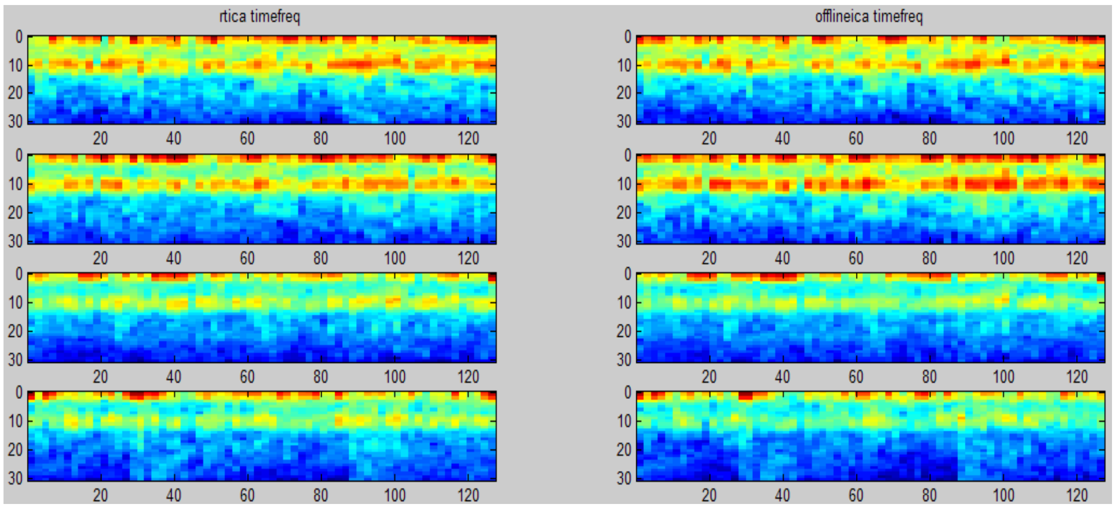


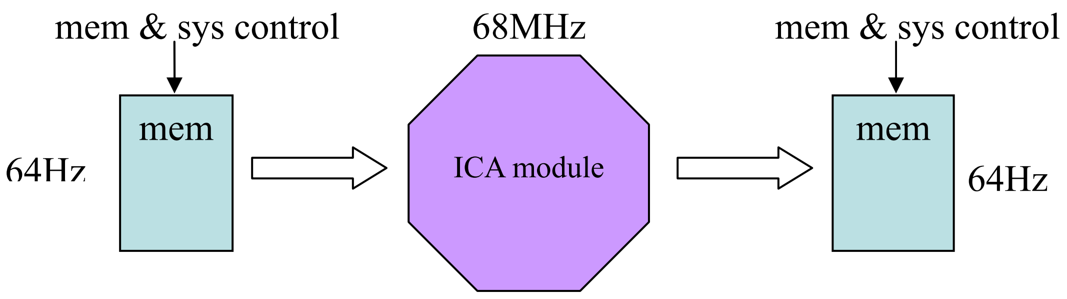
Fig. 3- 23 Time-Frequency comparison of on-line and off-line algorithm.

3-4 Summery of Comparison

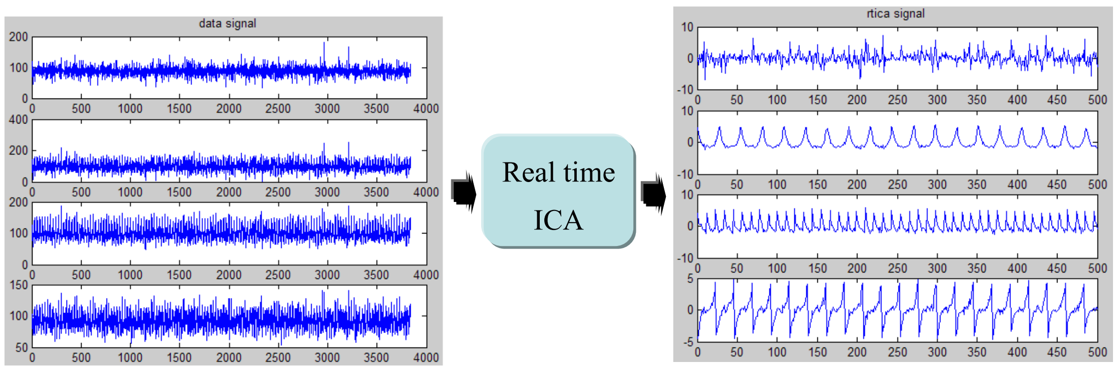
The simulation results from the above we can know that the amplitude of signal may be reverse or different in the ICA algorithm. These problems conform to the constraints referred in Chapter II. However, in real-time and off-line comparison, we can find the analysis result is better if the original signals are pure super Gaussian distribution. In real environment, the EEG measurement will contain many unknown noise or non-super Gaussian signal, and makes the effect drop in EEG analysis. In the comparison between on-line and off-line, because the on-line process collect small amount of information than off-line process, the correlation of on-line system might be different from off-line. But we can accept such a result that correlations are at least more than 80%.

Chapter4
Implementation of the On-line ICA System on FPGA

Top level real-time hardware architecture shown in Fig. 4- 1.



(a)



(b)

Fig. 4- 1 (a) Top level hardware architecture

(b) Illustration of real-time systems.

4-1 Architecture of real-time Systems

In this section we discuss the architecture of the digital circuit of implementation of the algorithm. In overall system, the main architecture required three parts in Fig. 4- 2: Infomax operation circuit, memory control circuit, and interface control circuit. Arithmetic operation circuit consists of matrix operation circuit and non linear transform. Though a large number of computation in ICA arithmetic operation circuit, the memory access will be frequently and complex. By using efficient memory controller to optimize the memory scheduling and reduce the circuit power consumption. The concept of the non linear transform, it is hard to execute for real-time cause of the DSP processor will approximate the value by loop iteration. If we implement by FPGA, it will be developed with very low cost hardware, and reduce unnecessary operation, and fast enough to execute for real-time.

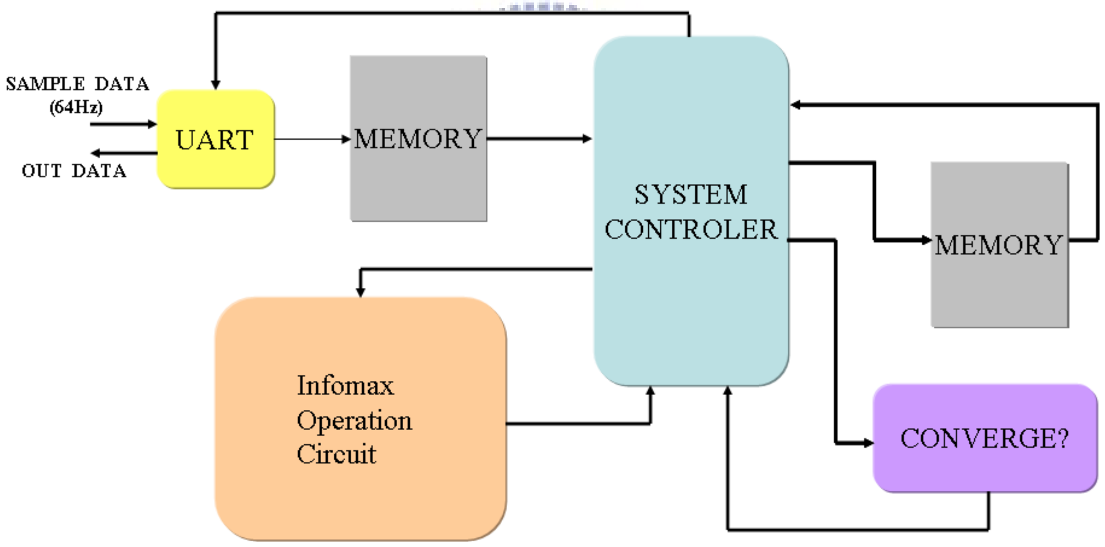


Fig. 4- 2 ICA main system architecture.

4-1-1 Implementation of Recursive Operation Circuit

The stability and high-precision are the properties of the recursive operation circuit. The errors may grow up with the growing iteration. In order to reduce errors in iteration, we develop a precision symmetrical non-linear piecewise look up table. Besides, we simplify the complex weight updating by deep pipeline design. And if the final weight coverage, the effective and fast matrix multiplier would be also designed in the system. We will discuss these three parts as follows:

(1)Precision symmetrical non-linear piecewise look up table

Considerations in design of non-linear look up table. We will focus on accuracy, hardware area, and computing time. On the accuracy, sign-bit fix point and symmetrical look-up table will be used by the property of non-linear symmetry and the simulation results of the algorithm [13]. It can reduce half areas by using symmetrical look-up table. For accuracy, in order to reduce errors, it will be 11 bits resolution for the symmetrical look-up table. However, the error estimates about

$$error = \varepsilon \times i, \tag{4.1}$$

where ε represent the resolution error of look-up table, and i represent the number of the system iteration. Then $error$ is the approximation of maximum error. As a result, the error will increase each iteration.

$$y = g(u) = \frac{1}{1+e^{-u}}. \tag{4.2}$$

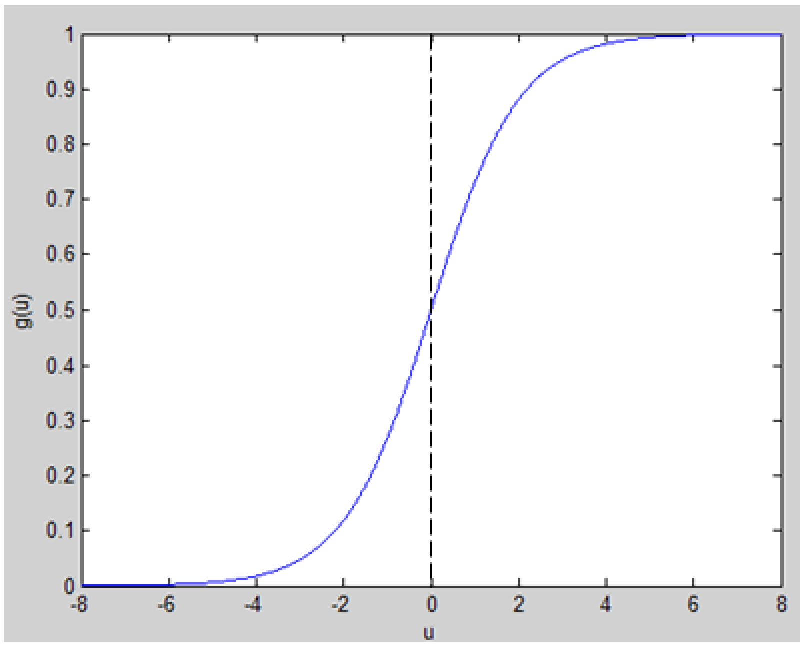


Fig. 4- 3 Non-linear function.

Table 4- 1 Description of the non-linear circuit

Name	Description
In[22:0]	$u = w \times x$
Out[10:0]	$\frac{1}{1+e^{-u}}$

Fig. 4- 3 and Fig. 4- 4 show the non-linear function and circuit block diagram. The operation can use symmetry due to non-linear function is singular function. Symmetry in block diagram is the judgement of whether input is less than zero or not. If input less than zero, it will use the same table to operate. As a result, it can save half area or double precision with the same table. Table 4- 1 shows the description of the non-linear look-up table.

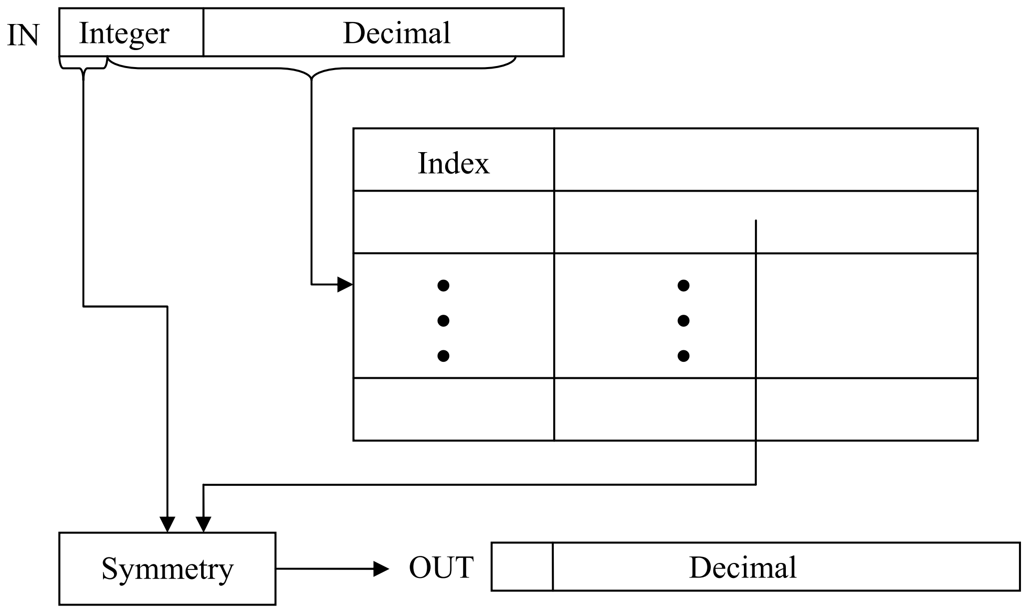


Fig. 4- 4 Precision Symmetrical Non-linear Piecewise Look up Table.

The error tolerance in block diagram is the need to verify. In MATLAB, all equation calculated by floating point. So we compare difference between fix point and floating-point. The root mean square error (RMSE) is 0.00029956

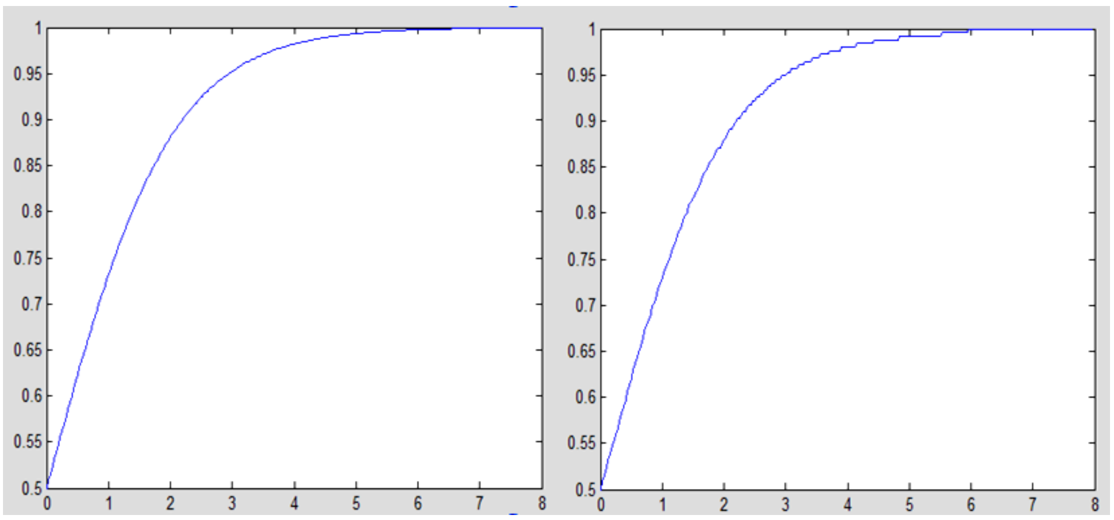


Fig. 4- 5 Comparison of floating-point and fix point function.

Left Fig. 4- 5 shows nonlinear function do precision computing by floating-point, and right show the look up table result. It can be seen slight error, but has been reached the real time implementation by increasing in non-linear processing.

(2)Weight Recursive operation

In this section, we integrate the weight update of infomax in a module. The module is the mathematical core of overall system. The processing speed decides whether to real time. So in this complex computing module, in order to increase the core speed and throughput, we use the method of deep pipeline. Although the integrated computing module will increase some areas, it is suitable for memory controller design. With less memory access times, we can save more power consumption. Besides, we can use more than one integrated computing unit to improve system efficiency. We will focus on the design of integrated computing module. The mathematics expressions as follow:

$$\Delta W = (I + (1-2y)u^T)W = (I + \varphi(u)u^T)W\,, \qquad (4.3)$$

$$W(t+1) = W(t) + l\Delta W\,. \qquad (4.4)$$

Table 4- 2 Description of recursive circuit

Name	Description
Weight[15:0]	Initial weight or previous weight
data[10:0]	Sample data from ADC
bias	For infomax non-linear
lrate	Initial learning rate is $0.0039 \approx 2^{-8}$, $lrate \propto t^{-1}$
New weight[15:0]	Gradient information update
New bias	Gradient information compensation

In hardware, however, fixed-point numeric is more practical. Although several groups have implemented floating-point adders and multipliers using FPGA, very few practical systems exist. The main disadvantages using floating-point in hardware are high resource requirements and high clock frequency. We use 16-bit fixed-point numeric, and the rough precision of the fixed point number representation can reach 0.000030518. Two bits for integer part, and 14 bits for the fractional part. When performing Infomax operations, normalization was performed to avoid overflow and make sure that the data path was always best utilized. The detail description is in Table 4- 2.

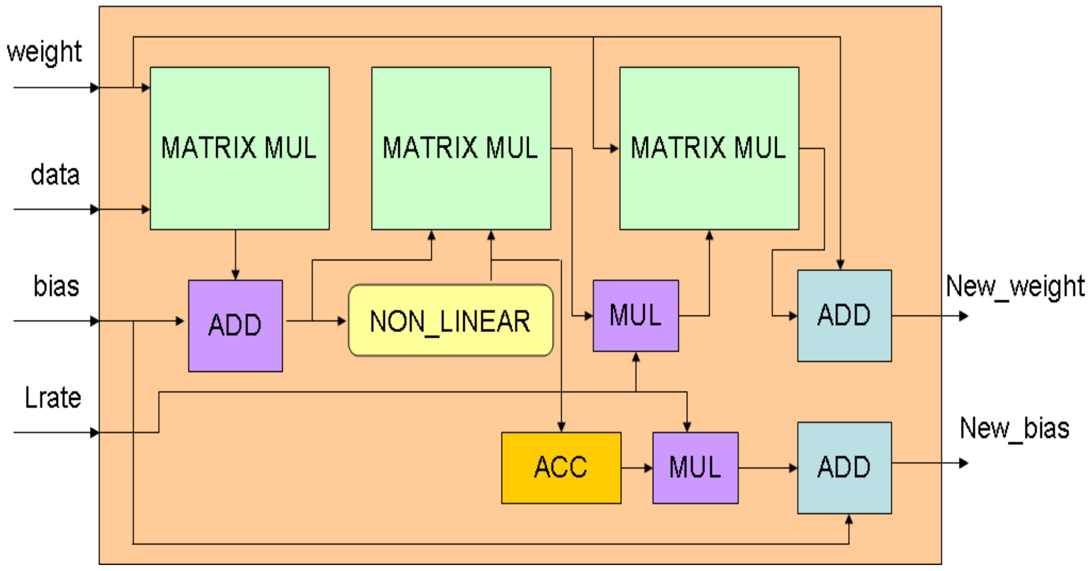


Fig. 4- 6 Integrated computing unit.

Fig. 4- 6 shows the integrated computing unit, it can process 4*512 matrix operation. The unit includes three 4*4 matrix operation, one accumulation, and mathematics operator. All speed must be greater than 68MHz to achieve real time execution. Then we design the calculation module with pipeline. In order to on-line execution, we estimate the consumption of cycle must less than 8300 cycles when core speed is 68MHz and 128 times training. As a result, the unit consumes 8192 cycles to find a new weight with gradient information update. The expressions as follow

$$operation\_cycle = \frac{core\_speed}{sample\_rate * iteration}. \quad (4.5)$$

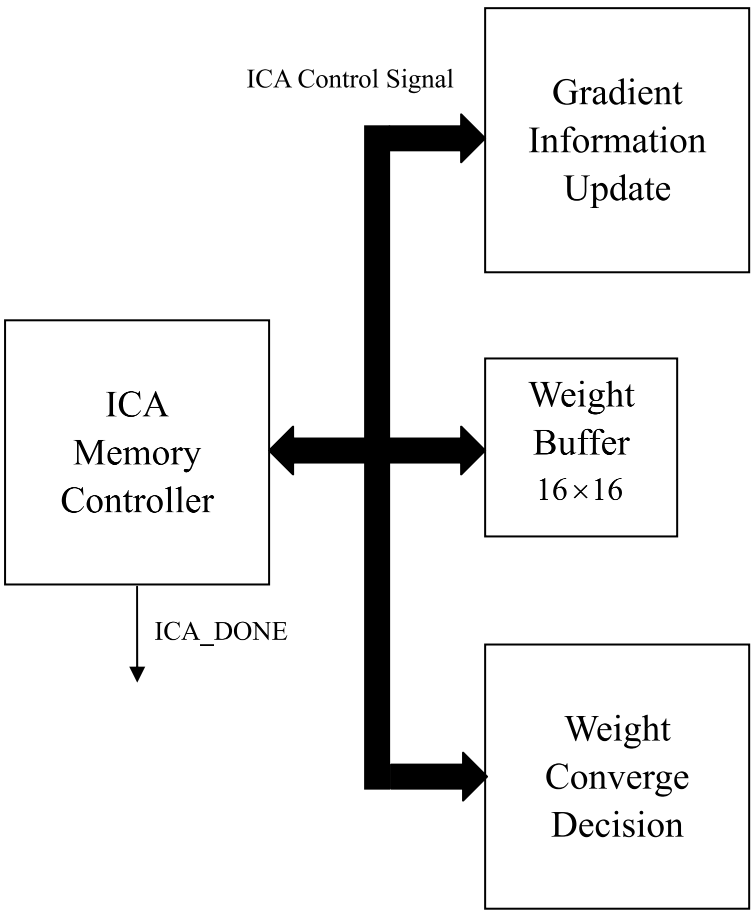


Fig. 4- 7 Main Calculation Model architecture.

Fig. 4- 7 shows the hierarchy architecture of main calculation model which with other components in iteration. In informax iteration, weight coverage decision determine whether the weight coverage. Because it is four-channel design, weight buffer would be 16 entries and 16bits resolution. Through each update, controllers will receive signals from the convergence module to decide whether the completion of the iterative. When weight converges, the memory controller will send a signal ICA_DONE to the result module.

(3)Fast matrix multiplier

When ICA completed the algorithm iteration, result module will receive a signal ICA_DONE from memory controller in Fig. 4- 8. Then read original signal multiplied by weight from IN_MEMORY. And the characteristic of this circuit is using parallel computing to find the four channels results at the same time in one cycle. It includes mean calculation and matrix multiplication, and also using pipeline design to increase throughput in unit time. Then put the result into OUT_MEMORY.

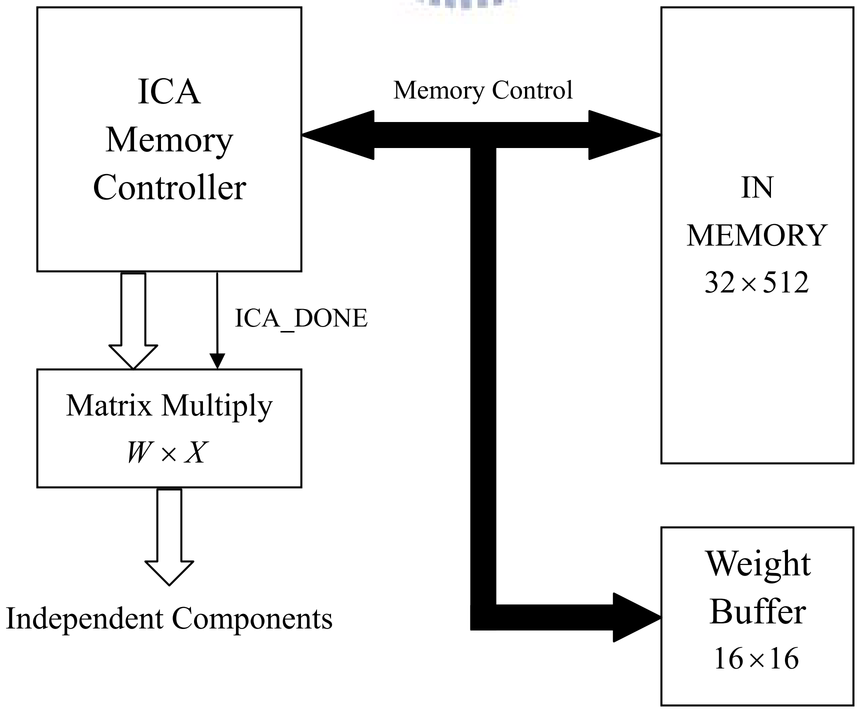

ICA Memory Controller
Memory Control
IN MEMORY 32×512
ICA_DONE
Matrix Multiply $W \times X$
Independent Components
Weight Buffer 16×16

Fig. 4- 8 Final Result architecture.

4-1-2 Implementation of system Controller

We would introduce overall system controller in this section. In Fig. 4- 9, it includes asynchronous memory controller and ICA system controller. In the system controller is mainly control the data from UART and sent the control signal to various modules in computing. Block diagram as follow:

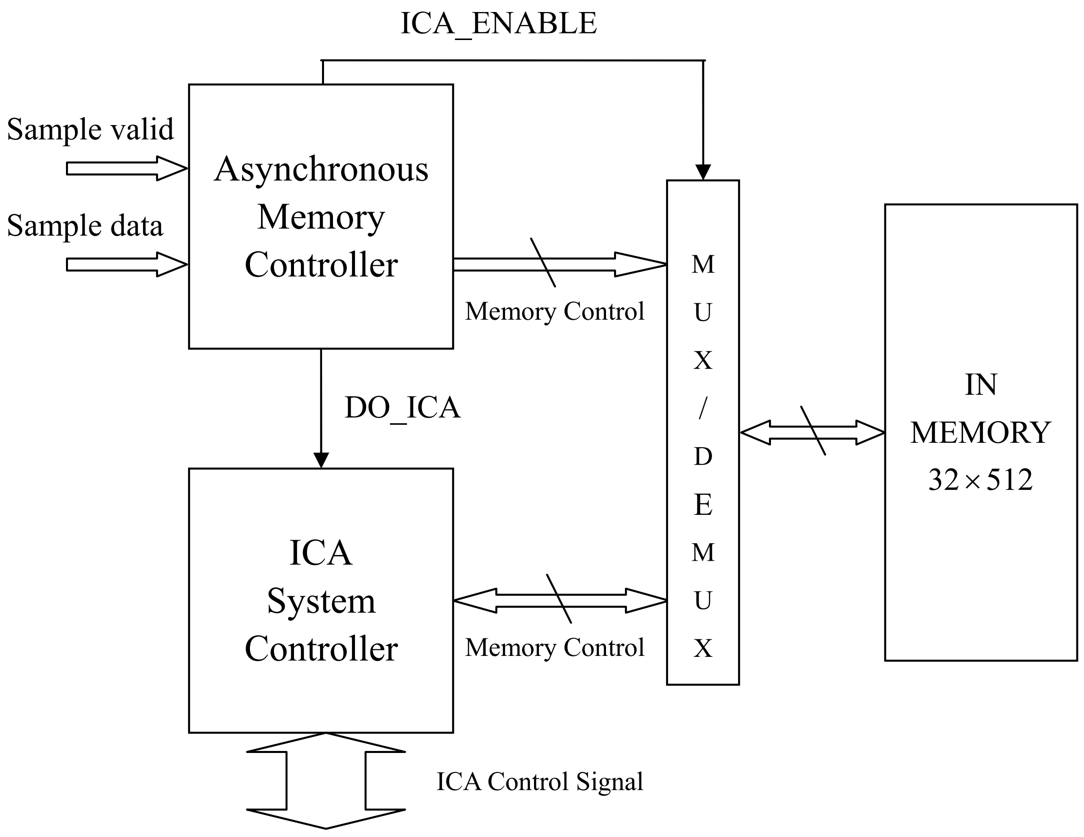


Fig. 4- 9 Main controller architecture.

(1)Asynchronous Memory Controller

In asynchronous memory controller, because of the external frequency is different from internal in Fig. 4- 10, so to use asynchronous conversion to the same speed of system input. External data would be sent into the system memory by a similar way as interrupted. We also placed a data counter that can be judged by the amount of data, and the ICA system controller would send signals to the correct path.

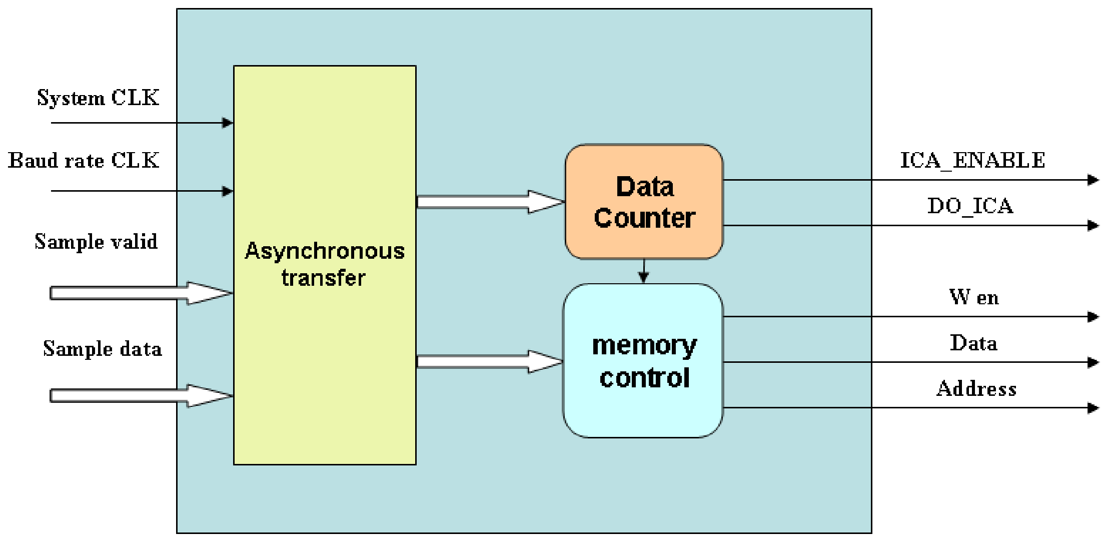


Fig. 4- 10 Asynchronous memory controller circuit.

(2)ICA System Controller

In this section, ICA system controller is the most important and complex core. It is like the microcontroller of the system. ICA system controller control various components which includes operate unit and memory unit. In order to keep from the control signal conflict, we divide the controller into two parts: system control signal and memory control signal. The control signals of system in Fig. 4- 11, old weight, data have sent to operated in weight update module. And the result is given into converge module, new_weight is sent into weight buffer for initial parameter of next iteration. In the memory control, we use an effective memory scheduling. In DSP instruction scheduling, it may waste on memory space and the efficiency of execution because of the data hazard may cause by read after write (RAW). In Fig. 4- 12 System pipeline flow,we use two recursive circuits and pipeline flow to reduce half amount of memory access. It is an effective memory scheduling.

In our design, we make the memory scheduling close together to reduce waiting time for hazard and to achieve on-line. In another hand, we also use enable signal in memory. All memory registers ports are controlled by the enable signal. When system needs access memory, the signal will trigger memory. We add a counter in memory bank, when specific memory block was accessed, the counter would count up. And the counter reached a critical value, controller will found that the memory has no demand for access. Therefore, the memory can into the power save mode. It can save power at memory access dynamically.

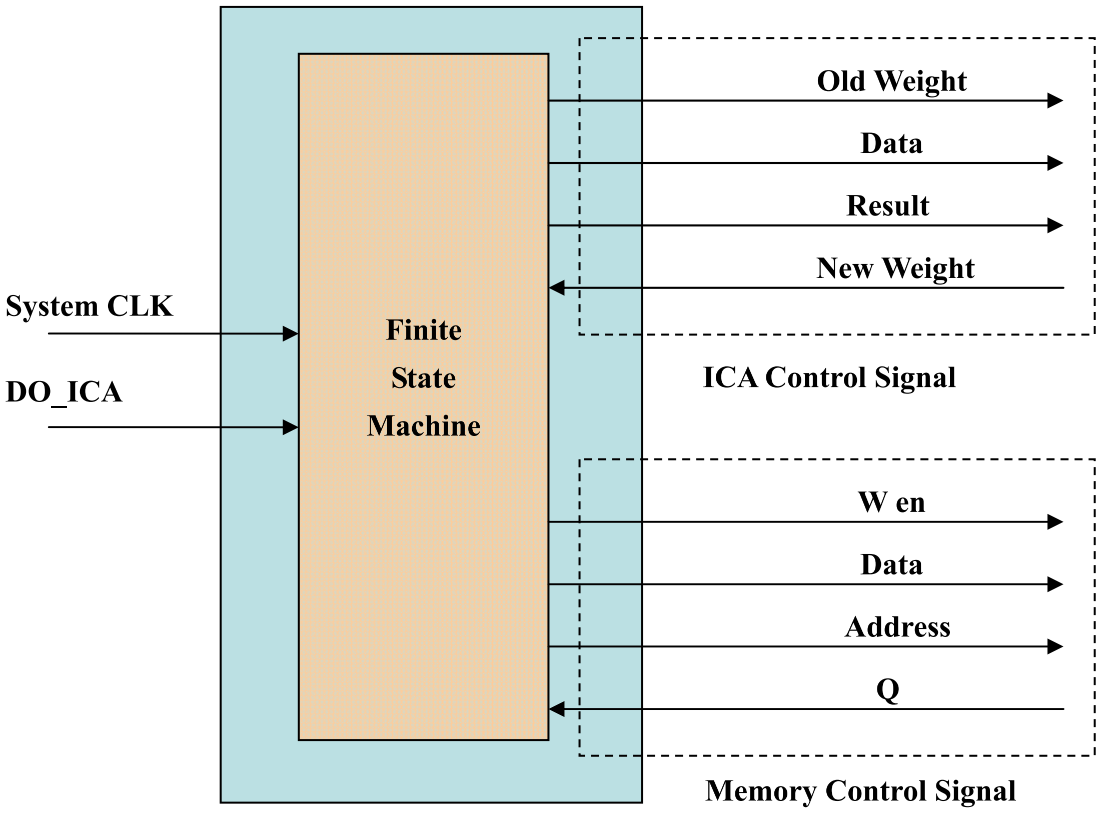


Fig. 4- 11 System controller circuit.

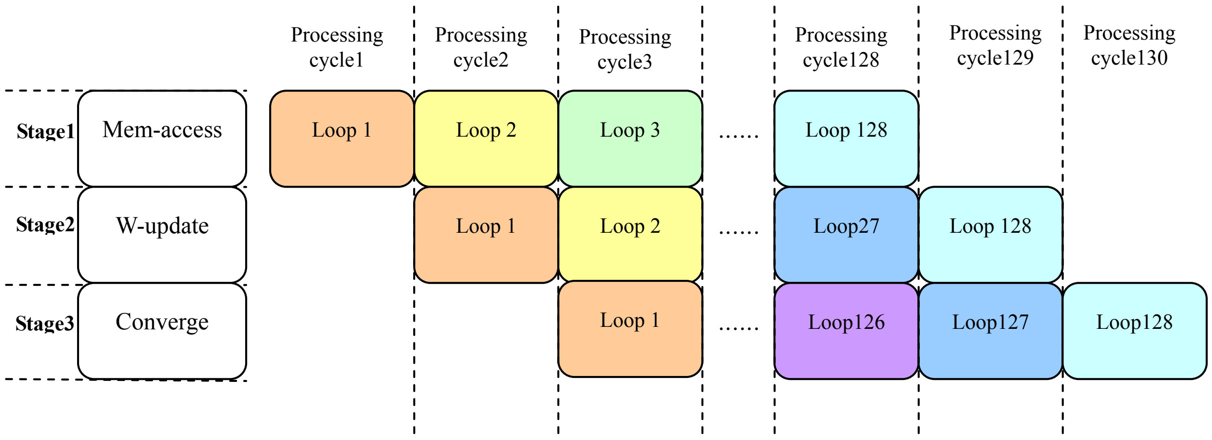


Fig. 4- 12 System pipeline flow.

See Fig. 4- 13, DO_ICA which will drive the entire system controller, the detail of finite state machine show in Table 4- 3. The state IDLE means that the system is receiving data, and can not doing operate. The state Training that is being execution mainly informax computing. Each time the training will consume 8192 cycles, and it will jump to next state Coverage when computing end. The state will jump to Done when the largest to 128 times of neural training. When state is Done, the data can only be written into memory. So the overall core speed depends on sample rate. The whole design detail of the micro-controller shows as follows:

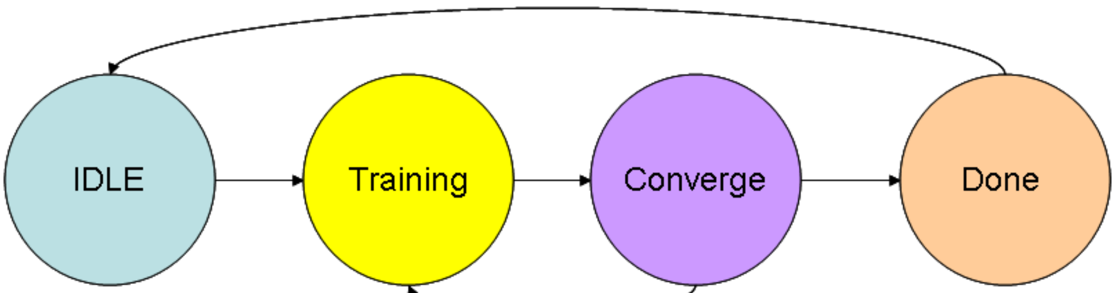


Fig. 4- 13 Illustration of micro-controller.

Table 4- 3 FSM of micro-controller

	State: IDLE	State: TRAINING	State: CONVERGE	State: DONE
Next state	TRAINING	If((&counter)& (&block)) CONVERGE else TRAINING	If(decision_step) If(&step) DONE else TRAINING else CONVERGE	If(DOICA) IDLE else DONE

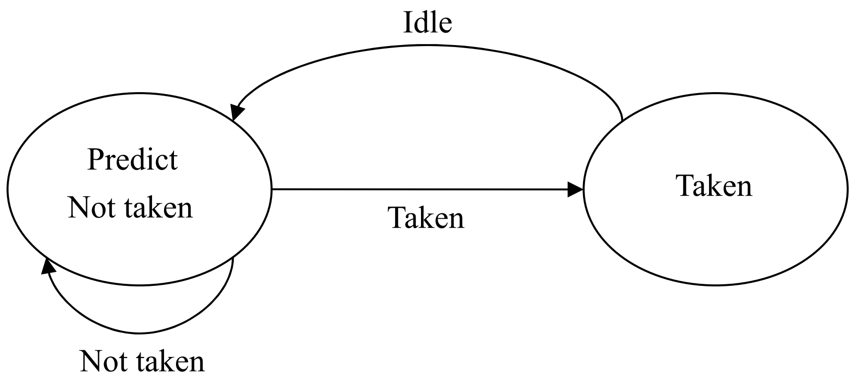


Fig. 4- 14 Dynamic Branch Prediction.

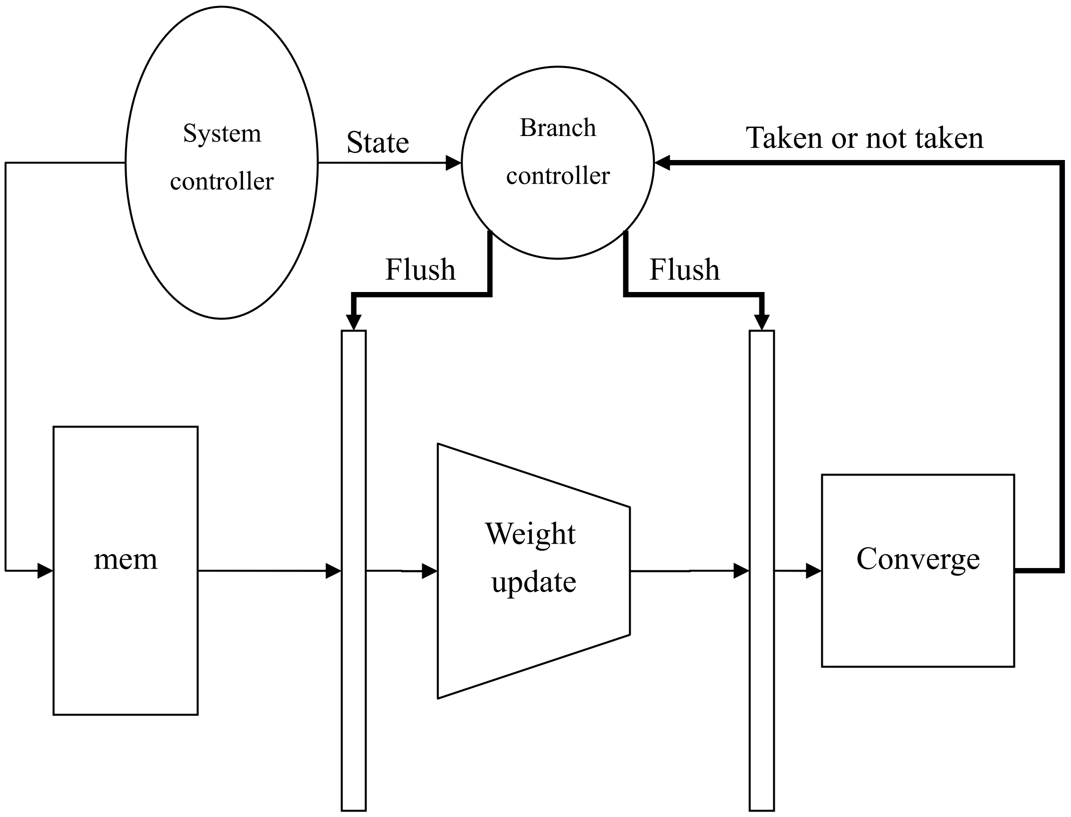


Fig. 4- 15 Branch Controller and Flush Line.

In order to overlapping memory access time by pipeline, this thesis use dynamic branch prediction in Fig. 4- 14. And add flush line to clear forward pipeline register in Fig. 4- 15. According to the characteristic of ICA algorithm, the branch prediction can reduce memory access time effectively. Fig. 4- 15 illustrate that we predict the branch always not taken, the branch controller would send a flush signal to clear forward pipeline register if taken happen and state at idle.

4-1-3 Interface Design

(1)RS232

RS232 is the old standard and is starting to become obsolete. Few if any laptops even have RS232 ports (serial ports) today, with USB becoming the new universal standard for attaching hardware.

(2)Tx and Rx

Tx represents transmit and Rx represents receive. The transmit pin always transmits data, and the receive pin always receives it. Notice Tx is connected to Rx, and Rx is connected to Tx.

(3)Baud Rate

Baud is a measurement of transmission speed in asynchronous communication. The computer, any adaptors, and the UART must all agree on a single speed of information - bits per second (bps).

(4)Asynchronous Serial Transmission

Baud rate defines bits sent per second. But baud only has meaning if the two communicating devices have a synchronized clock. Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

When a word is given to the UART for Asynchronous transmissions in Fig. 4- 16, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word.

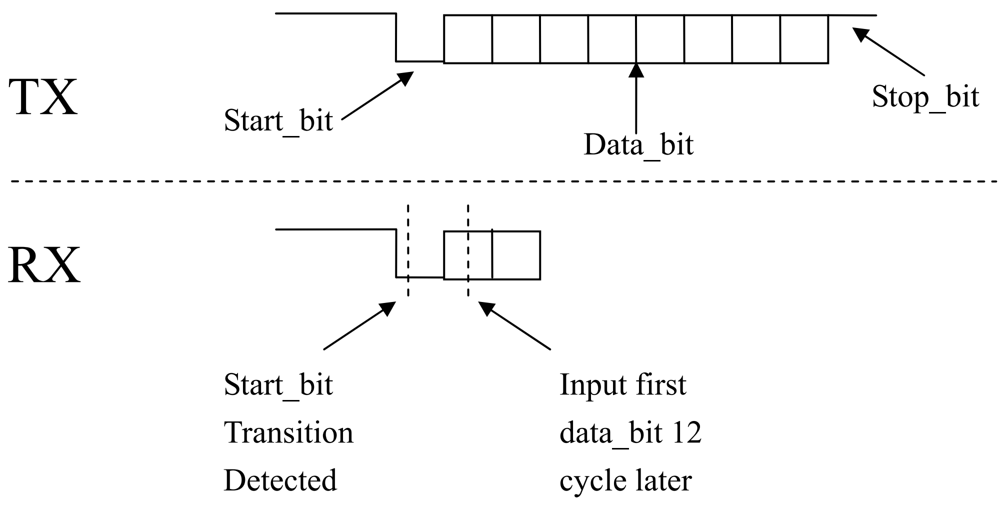


Fig. 4- 16 Illustration of RS232 Protocol.

When data is being transmitted, the sender does not know when the receiver has 'looked' at the value of the bit - the sender only knows when the clock says to begin transmitting the next bit of the word.

When the receiver has received all of the bits in the data word, it may check for the Parity Bits, and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted. Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host. If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent.

(5)HEADER controller

In the header controller, we divide into two parts. One is receiver header controller and another is transmitter header controller. In whole system with four channels, transmit and receive from UART respectively. Fig. 4- 17 shows receiver header controller design. We define the transmitter protocol first. When receiver header controller receives data FF, it means that the FF is the header of the next four channels. Hence we combine the four channel data as a memory bandwidth. It is helpful and efficient to control these data show in GUI. However, the addresses of memory are 512 entries, it means that we process 512 data in each iteration. The signal Sample data is the main module input with 32 bit bandwidth.

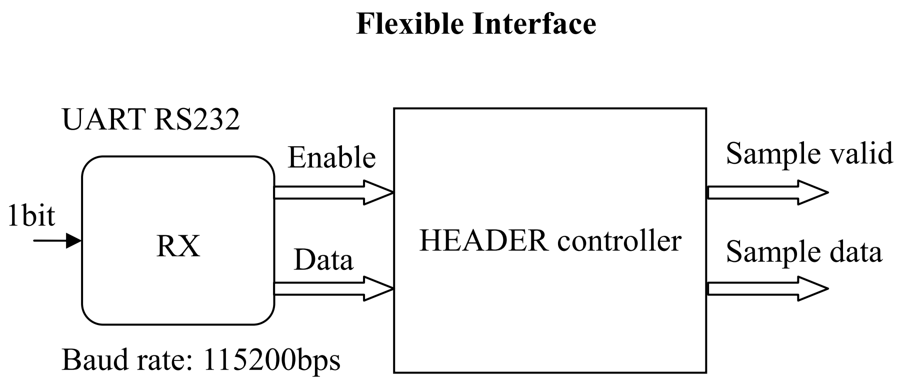


Fig. 4- 17 Receiver header controller architecture.

On the other hand, we also develop a transmitter header controller. It is harder design than receiver. It is because the system frequency higher than interface. In order to transmit system data, we need an asynchronous (first in first out) FIFO buffer to control the result data. The design shows in Fig. 4- 18. the asynchronous FIFO with two different CLK, represent the speed of input and output respectively. The numbers of entry are 128, because we update the result data are two second with 64Hz sample rate. And in transmitter header controller design, we add an encoder to encode the fix point result to 8 bit integer. As a result of the transmitter protocol is the same as the receiver, we also add a header FF in front of the result data. Finally, we connect the header controller with the TX module.

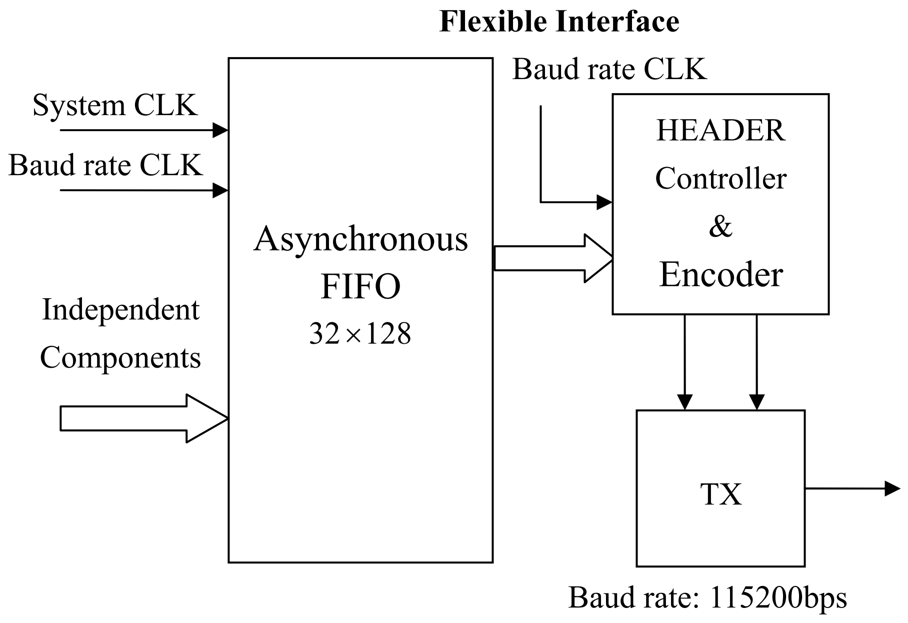


Fig. 4- 18 Transmitter header controller architecture.

In HEADER Controller design, we add header FF and control data in order by using finite state machine. The detail FSM and control signal show in Fig. 4- 19 and Table 4- 4.

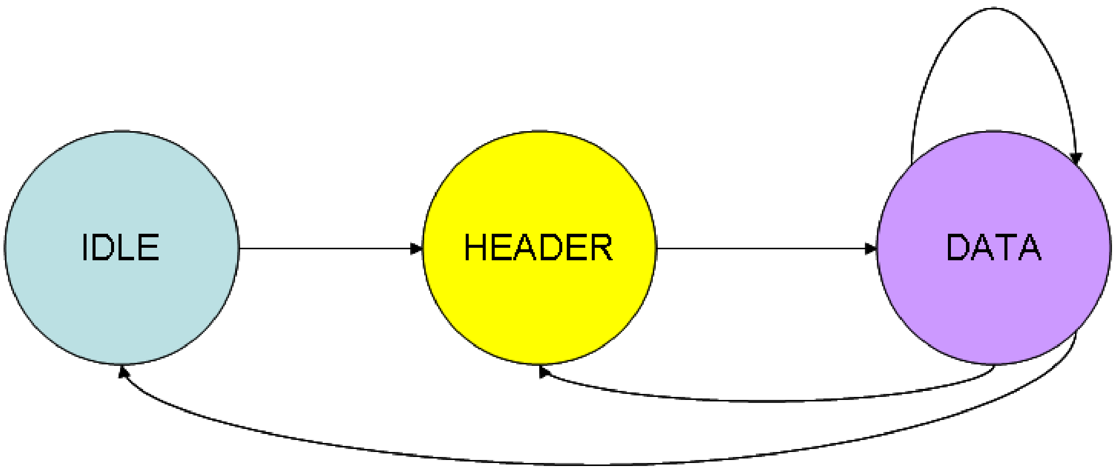


Fig. 4- 19 Illustration of Header Controller.

Table 4- 4 FSM of Header Controller

	IDLE	HEADER	SEND_DATA

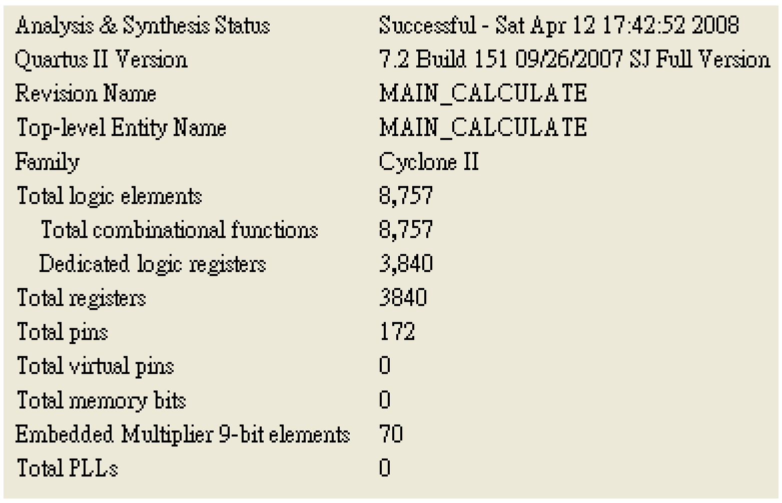
Next state	If(empty)
	IDLE
	else
	HEADER
		SEND_DATA	If(empty && state=stop)
			IDLE
			else if(state=stop && counter=4)
			HEADER
			else
			SEND_DATA

4-2 FPGA Simulation Result in Integrated System

We develop the algorithm by VHDL which gives the implementation not only better performance but also less consumption of gate array in the FPGA. In this section, we will show the FPGA simulation result in each component. It contains compilation report, timing report, and simulation report.

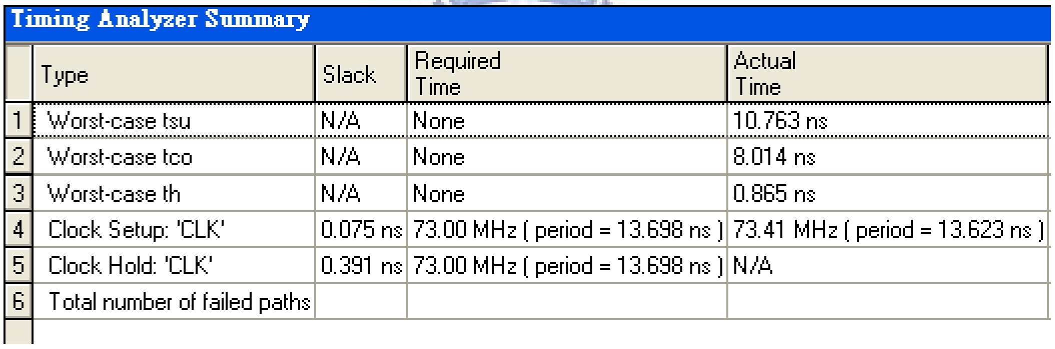
4-2-1 FPGA Simulation in Recursive Operation Circuit

We can see Fig. 4- 20, it show the device type and the detail of logic elements. We synthesized using Altera DE2 targeted for Cyclone II family. The design has introduced in 4-1-1, our system frequency depend on the recursive core speed. Because of we use deep pipeline to enhance the performance, the result will cause increase of the number of register and area. On the other hand, we also use embedded multiplier to improve the system speed.



Analysis & Synthesis Status	Successful - Sat Apr 12 17:42:52 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	MAIN_CALCULATE
Top-level Entity Name	MAIN_CALCULATE
Family	Cyclone II
Total logic elements	8,757
Total combinational functions	8,757
Dedicated logic registers	3,840
Total registers	3840
Total pins	172
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	70
Total PLLs	0

(a)



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	10.763 ns
2	Worst-case tco	N/A	None	8.014 ns
3	Worst-case th	N/A	None	0.865 ns
4	Clock Setup: 'CLK'	0.075 ns	73.00 MHz (period = 13.698 ns)	73.41 MHz (period = 13.623 ns)
5	Clock Hold: 'CLK'	0.391 ns	73.00 MHz (period = 13.698 ns)	N/A
6	Total number of failed paths			

(b)

Fig. 4- 20 Detail of compilation report (a) summary (b) timing.

The hardware simulation we used by ModelSim, the post simulation result shown in Fig. 4- 21. The library provided by Altera. The recursive operation circuit design should calculate new weight in 8192 cycles. If the maximum numbers of training are 128, it may cost 13ms totally and less than sample time 16ms. Fig. 4- 22 provides the speed consumption of recursive circuit compare with software. The weight update performance is 56 times faster than software.

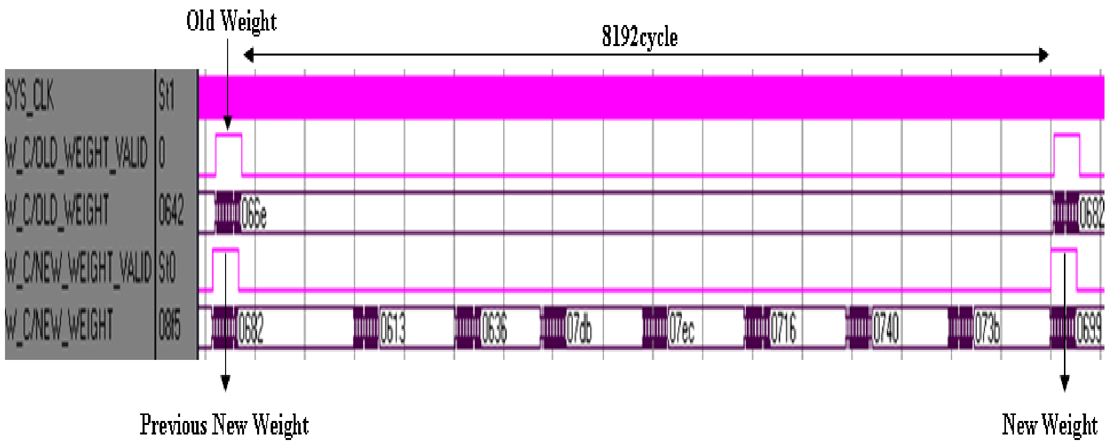


Fig. 4- 21 post simulation of recursive operation circuit.

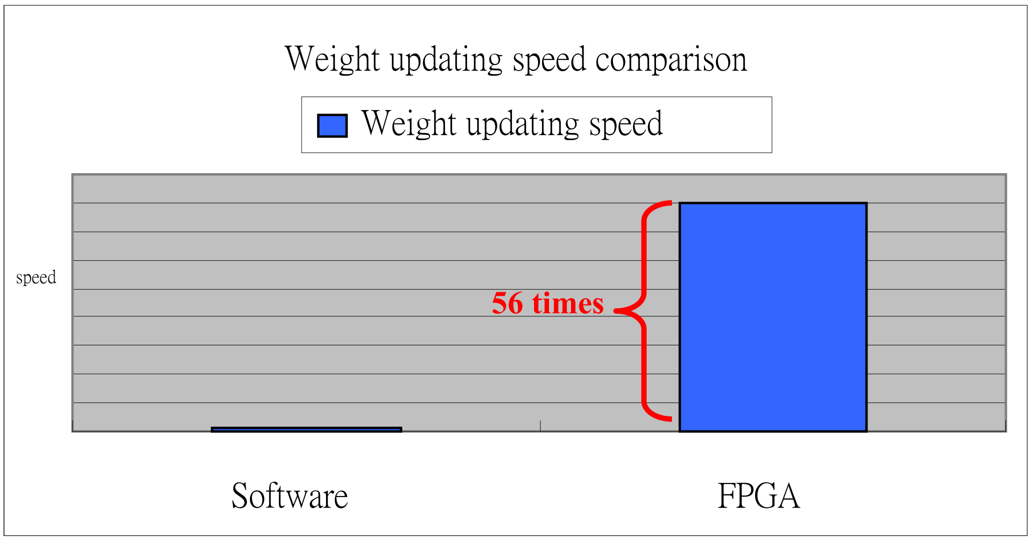
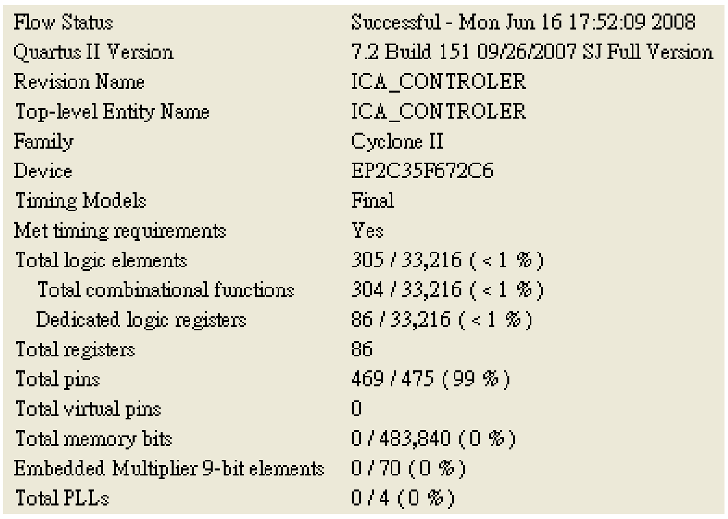


Fig. 4- 22 Time consumption of Weight calculation.

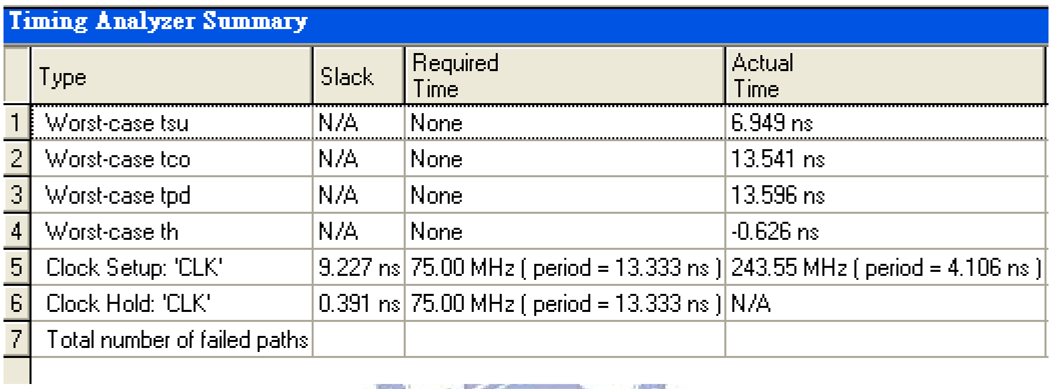
4-2-2 FPGA Simulation in System Controller

The system controller detail report shown in Fig. 4- 23. We can find that the system controller implementation with very low cost logic elements. This is because we simplify the controller into two parts: data controller and memory controller. There are a lot of pins in controller because it connects all components that include main calculate operation, weight buffer, and memory.



Flow Status	Successful - Mon Jun 16 17:52:09 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	ICA_CONTROLER
Top-level Entity Name	ICA_CONTROLER
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	305 / 33,216 (< 1 %)
Total combinational functions	304 / 33,216 (< 1 %)
Dedicated logic registers	86 / 33,216 (< 1 %)
Total registers	86
Total pins	469 / 475 (99 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

(a)



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	6.949 ns
2	Worst-case tco	N/A	None	13.541 ns
3	Worst-case tpd	N/A	None	13.596 ns
4	Worst-case th	N/A	None	-0.626 ns
5	Clock Setup: 'CLK'	9.227 ns	75.00 MHz (period = 13.333 ns)	243.55 MHz (period = 4.106 ns)
6	Clock Hold: 'CLK'	0.391 ns	75.00 MHz (period = 13.333 ns)	N/A
7	Total number of failed paths			

(b)

Fig. 4- 23 Detail of compilation report (a) summary (b) timing.

The behavior simulation shows in Fig. 4- 24 and Fig. 4- 25. The flow path of finite state machine is simple. Fig. 4- 24 illustrates the situation of the signal DOICA triggers the system controller and the new weight do not coverage. Fig. 4- 25 show that if new weight converge, the FSM will jump to next stat DONE.

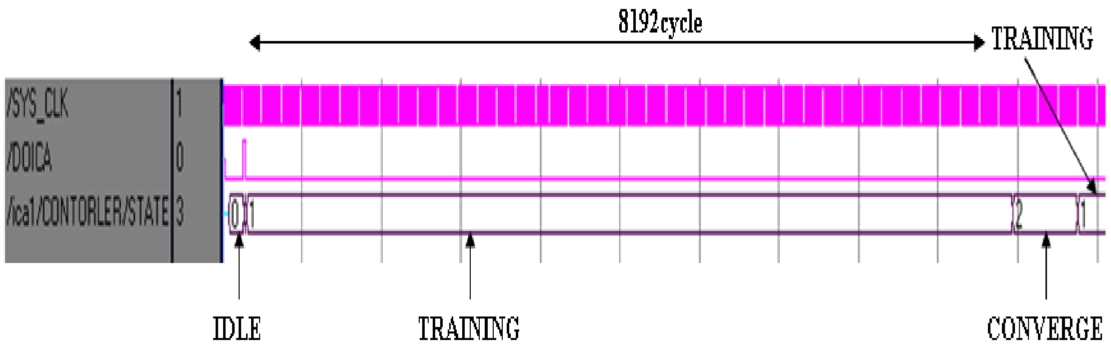


Fig. 4- 24 post simulation of system controller with no converge.

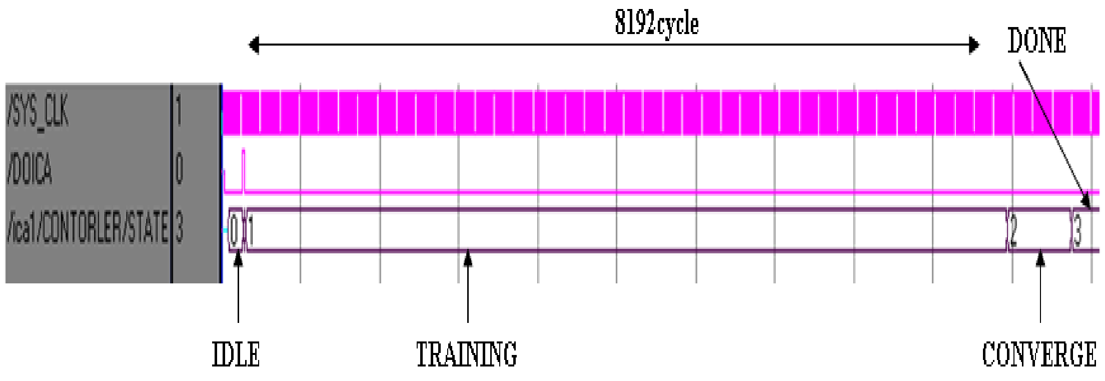
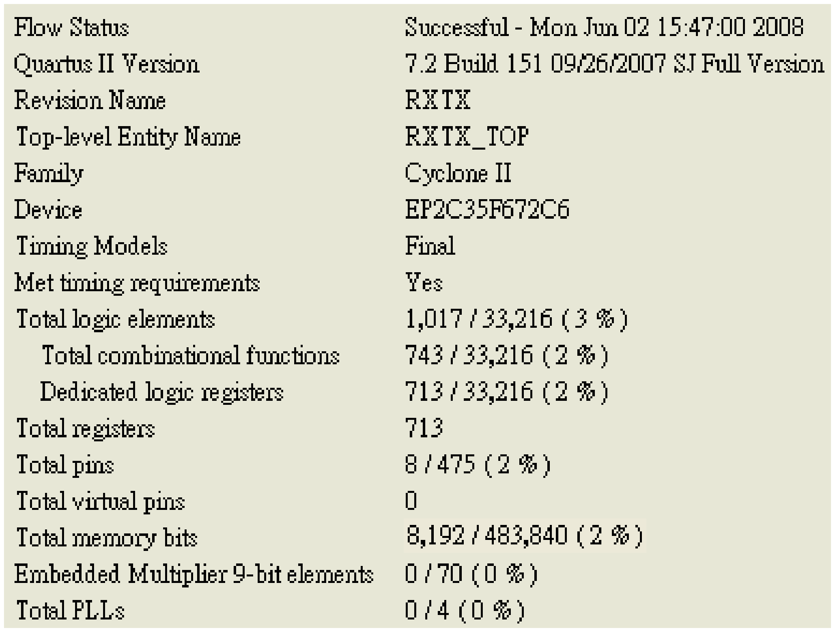


Fig. 4- 25 post simulation of system controller with converge.

4-2-3 FPGA Simulation in Interface Design

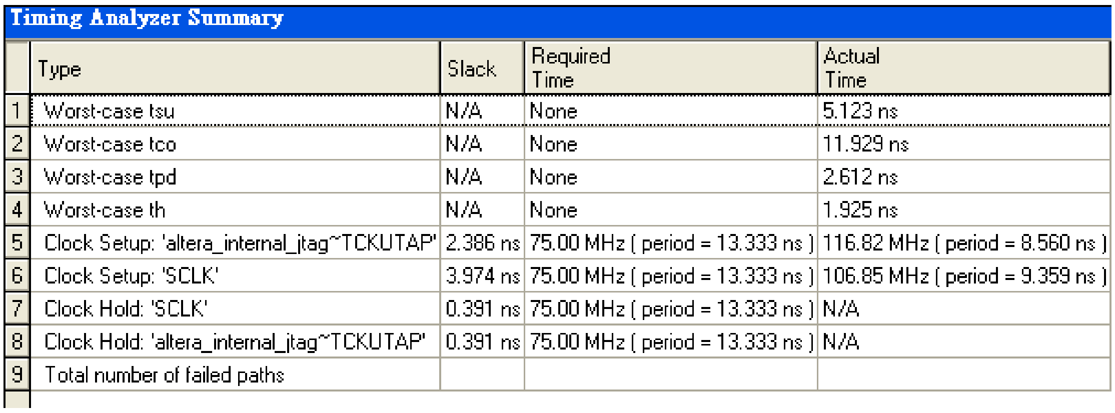
In interface hardware design, we choose UART with baud rate 115200bps. The implementation contains RS232 transmitter, receiver, header controller, encoder and asynchronous FIFO.

The interface behavior simulation show in Fig. 4- 27 and Fig. 4- 28 individually. Input interface receive EEG signal by RX, and the data format will stream in with header. Fig. 4- 27 shows that the SAMPLEDATA are 32bits finally. And Fig. 4- 28 shows transmitter format. Signal PUSH means that push results in asynchronous FIFO, then pop them in order by header controller.



Flow Status	Successful - Mon Jun 02 15:47:00 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	RXTX
Top-level Entity Name	RXTX_TOP
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1,017 / 33,216 (3 %)
Total combinational functions	743 / 33,216 (2 %)
Dedicated logic registers	713 / 33,216 (2 %)
Total registers	713
Total pins	8 / 475 (2 %)
Total virtual pins	0
Total memory bits	8,192 / 483,840 (2 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

(a)



	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	5.123 ns
2	Worst-case tco	N/A	None	11.929 ns
3	Worst-case tpd	N/A	None	2.612 ns
4	Worst-case th	N/A	None	1.925 ns
5	Clock Setup: 'altera_internal_jtag~TCKUTAP'	2.386 ns	75.00 MHz (period = 13.333 ns)	116.82 MHz (period = 8.560 ns)
6	Clock Setup: 'SCLK'	3.974 ns	75.00 MHz (period = 13.333 ns)	106.85 MHz (period = 9.359 ns)
7	Clock Hold: 'SCLK'	0.391 ns	75.00 MHz (period = 13.333 ns)	N/A
8	Clock Hold: 'altera_internal_jtag~TCKUTAP'	0.391 ns	75.00 MHz (period = 13.333 ns)	N/A
9	Total number of failed paths			

(b)

Fig. 4- 26 Detail of compilation report (a) summary (b) timing.

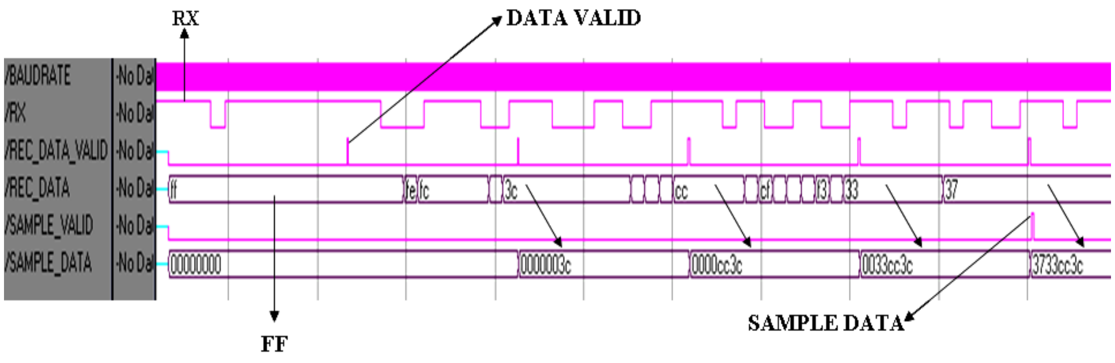


Fig. 4- 27 post simulation of input interface.

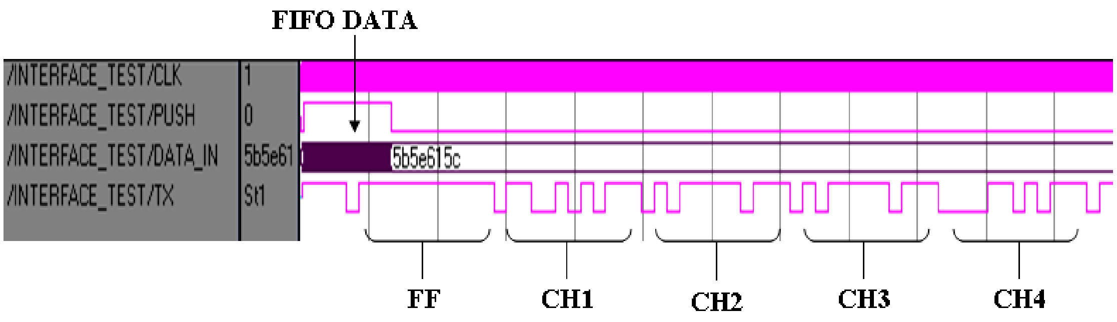
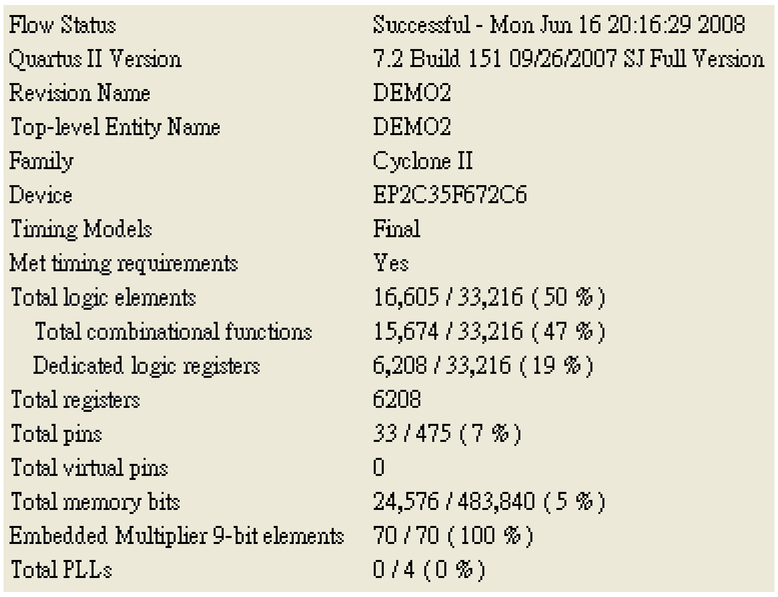


Fig. 4- 28 post simulation of output interface.

4-2-4 FPGA Simulation in Integrated System

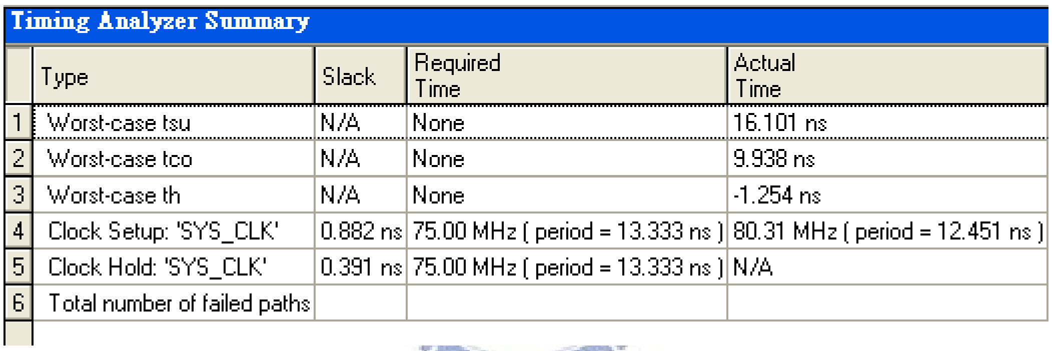
The system total logic elements shown in Fig. 4- 29, it costs about 16600 logic elements. And the memory bits about 24576, it accords with the initial design. The total memory bits show as blow:

$$total\_memorybits = data\_bandwidth \times memory\_entries + data\_bandwidth \times fifo\_entries$$



Flow Status	Successful - Mon Jun 16 20:16:29 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	DEMO2
Top-level Entity Name	DEMO2
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	16,605 / 33,216 (50 %)
Total combinational functions	15,674 / 33,216 (47 %)
Dedicated logic registers	6,208 / 33,216 (19 %)
Total registers	6208
Total pins	33 / 475 (7 %)
Total virtual pins	0
Total memory bits	24,576 / 483,840 (5 %)
Embedded Multiplier 9-bit elements	70 / 70 (100 %)
Total PLLs	0 / 4 (0 %)

(a)



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time
1	Worst-case tsu	N/A	None	16.101 ns
2	Worst-case tco	N/A	None	9.938 ns
3	Worst-case th	N/A	None	-1.254 ns
4	Clock Setup: 'SYS_CLK'	0.882 ns	75.00 MHz (period = 13.333 ns)	80.31 MHz (period = 12.451 ns)
5	Clock Hold: 'SYS_CLK'	0.391 ns	75.00 MHz (period = 13.333 ns)	N/A
6	Total number of failed paths			

(b)

Fig. 4- 29 Detail of compilation report (a) summary (b) timing.

Since add the efficient memory scheduling, the core frequency determines the system performance. Fig. 4- 29 shows the core frequency, and in order to achieve real-time operation in real environment, we need to overdesign the system. The system speed is up to 80 MHz actually.

The behavior simulation shows in Fig. 4- 30. It means that the ICA did 20 times Infomax weight training in this division with 512 point. The total process time is about 2.1ms, $process\_time = cycle\_time \times 8192 \times training$, and the result will write in system memory before next input data. The maximum input-output delay is 13.7ms with 128 training loops. Another way, reduce memory access time by overlap processing can speed up total system perform 69 times than software at 68 MHz. If the core speed up to 80 MHz, the total system perform can 81 times than software.

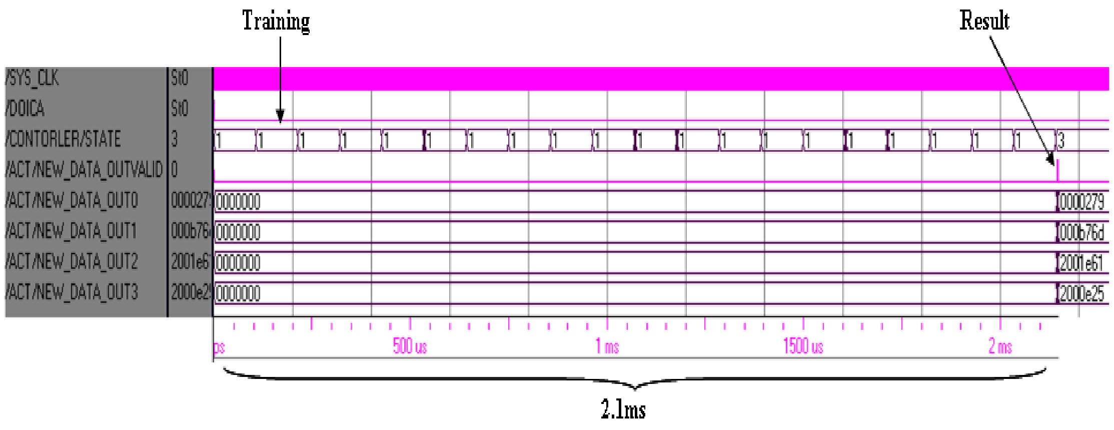


Fig. 4- 30 post simulation of overall system.

4-3 Device for Demonstration

In the part of demonstration, we need to integrate three parts into a prototype system. Fig. 4- 31 illustrates the main components of the prototype system that we developed. We will discuss these three parts in next subsection.

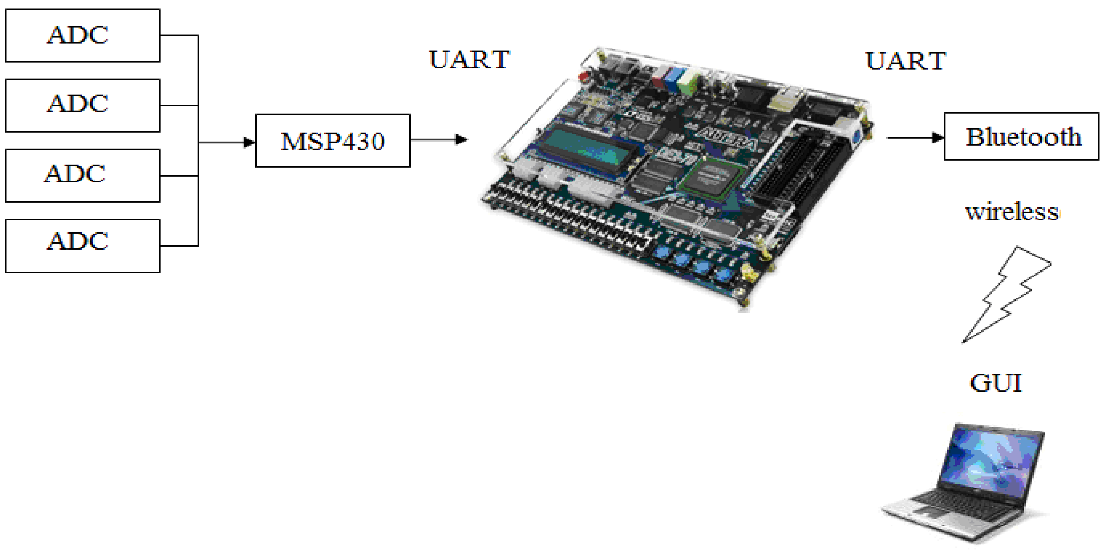


Fig. 4- 31 Illustration of demonstration.

4-3-1 Four channel EEG Brain-computer Interface

As well as invasive experiments, we have been experiments in humans using non-invasive neuroimaging technologies as interfaces. Although they are easy to wear, non-invasive implants produce poor signal resolution because the skull dampens signals, dispersing and blurring the electromagnetic waves created by the neurons. Although the waves can still be detected it is more difficult to determine the area of the brain that created them or the actions of individual neurons. So we need ICA process to split these signals wave.

(1)Electrode

Fig. 4- 32 Medi-Trace 200.

We use Medi-Trace 200 in Fig. 4- 32. All electrodes have an Ag/AgCl sensor of the highest quality and a push button. These electrodes us a solid gel which is an excellent adhesive and conducts perfectly. The electrodes use a new gel which sticks faster to the skin and reduces the skin-impedance even further.

(2)ADC

The AD7466 is 12-bit, high speed, low power, successive approximation analog-to-digital converters (ADCs) in Fig. 4- 33. In order to fit our specification, we truncate to 8-bit resolution. The parts operate from a single 1.6 V to 3.6 V power supply and feature throughput rates up to 200 kSPS with low power dissipation. The parts contain a low noise, wide bandwidth track-and-hold amplifier, which can handle input frequencies in excess of 3 MHz.

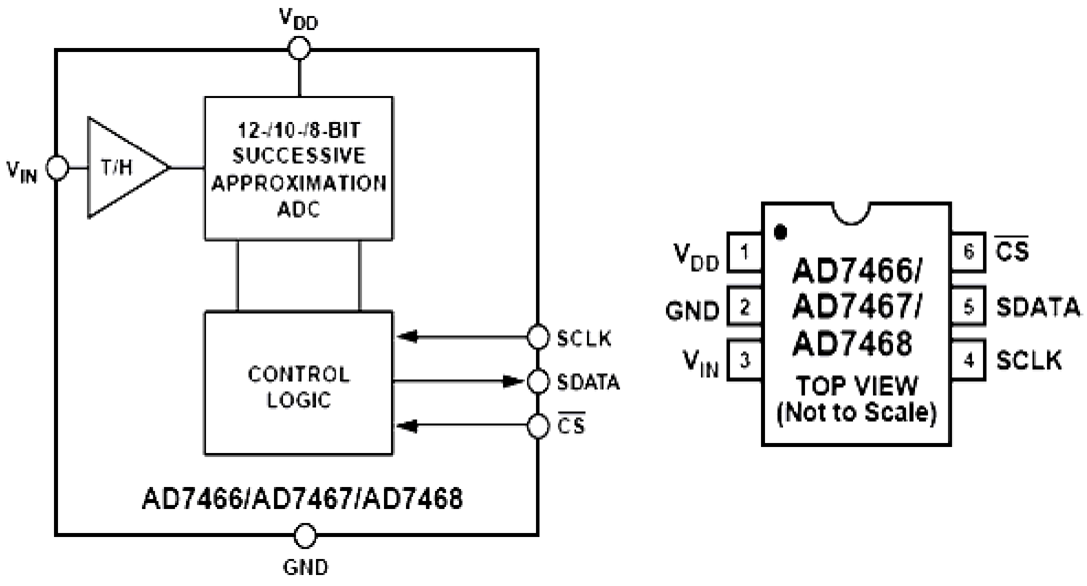


Fig. 4- 33 Pin Configuration of AD7466.

The serial interface on the AD7466 allows the parts to be connected directly to many different micro-processors. This section explains how to interface the AD7466 with some of the more common microcontroller and DSP serial interface protocols.

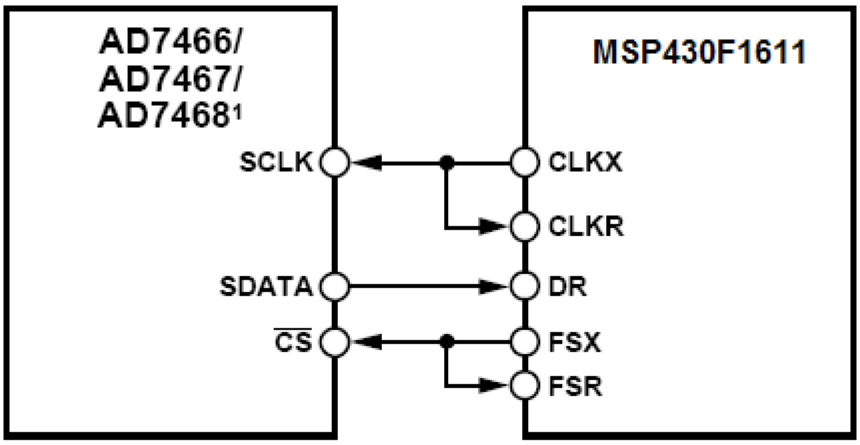


Fig. 4- 34 Interfacing to the MSP430F161.

Fig. 4- 34 shows the connection diagram. For signal processing applications, it is imperative that the frame synchronization signal from the MSP430F1611 provide equidistant sampling.

(3)Microcontroller

The MSP430 is a microcontroller family from Texas Instruments in Fig. 4- 35. Built around a 16-bit CPU, the MSP430 is designed for low cost, low power consumption embedded applications. The architecture is reminiscent of the DEC PDP-11. The MSP430 is particularly well suited for wireless RF or battery powered applications.

The MSP430 CPU has a 16-bit RISC architecture that is highly transparent to the application. All operations, other than program-flow instructions, are performed as register operations in conjunction with seven addressing modes for source operand and four addressing modes for destination operand. The CPU is integrated with 16 registers that provide reduced instruction execution time. The register-to-register operation execution time is one cycle of the CPU clock. Peripherals are connected to the CPU using data, address, and control buses, and can be handled with all instructions.

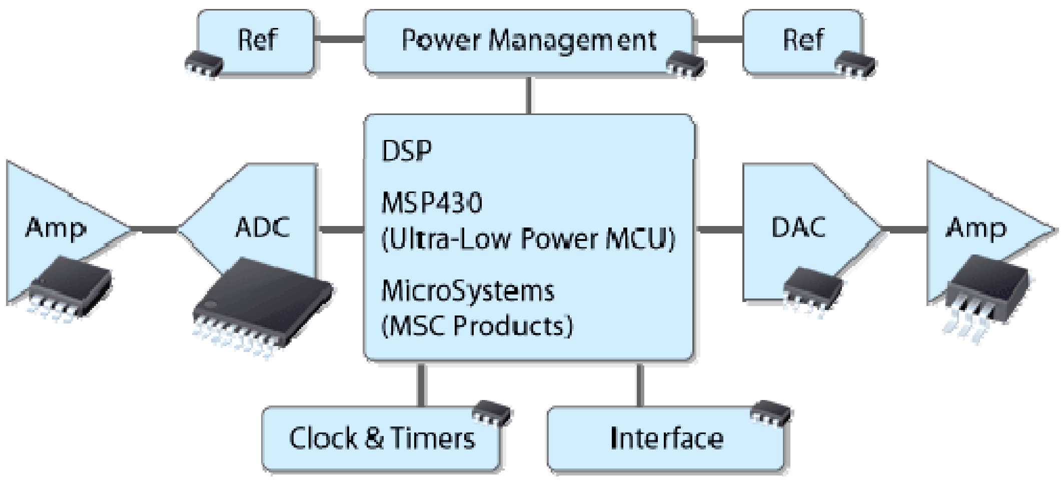


Fig. 4- 35 MSP430 Applications.

The MSP430f161 devices have a second hardware universal synchronous/asynchronous receive transmit (USART1) peripheral module that is used for serial data communication. The USART supports synchronous SPI and asynchronous UART communication protocols, using double-buffered transmit and receive channels.

(4)BCI (brain-computer interface)

The detail of BCI we discuss above. As we know capability and specification in each component, we can integrate them into a BCI. Fig. 4- 36 shows the detail signal connection of BCI. The EEG signal recoded by sensor, then transmit them by AD7466 with Serial Peripheral Interface protocol. If a single slave device is used, the CS pin *may* be fixed to logic low if the slave permits it. With multiple slave devices, an independent CS signal is required from the master for each slave device. Most devices have tri-state outputs that become high impedance when the device is not selected. Devices without tri-state outputs can't share SPI bus segments with other devices; only one such slave may talk to the master, and only its CS may be activated.

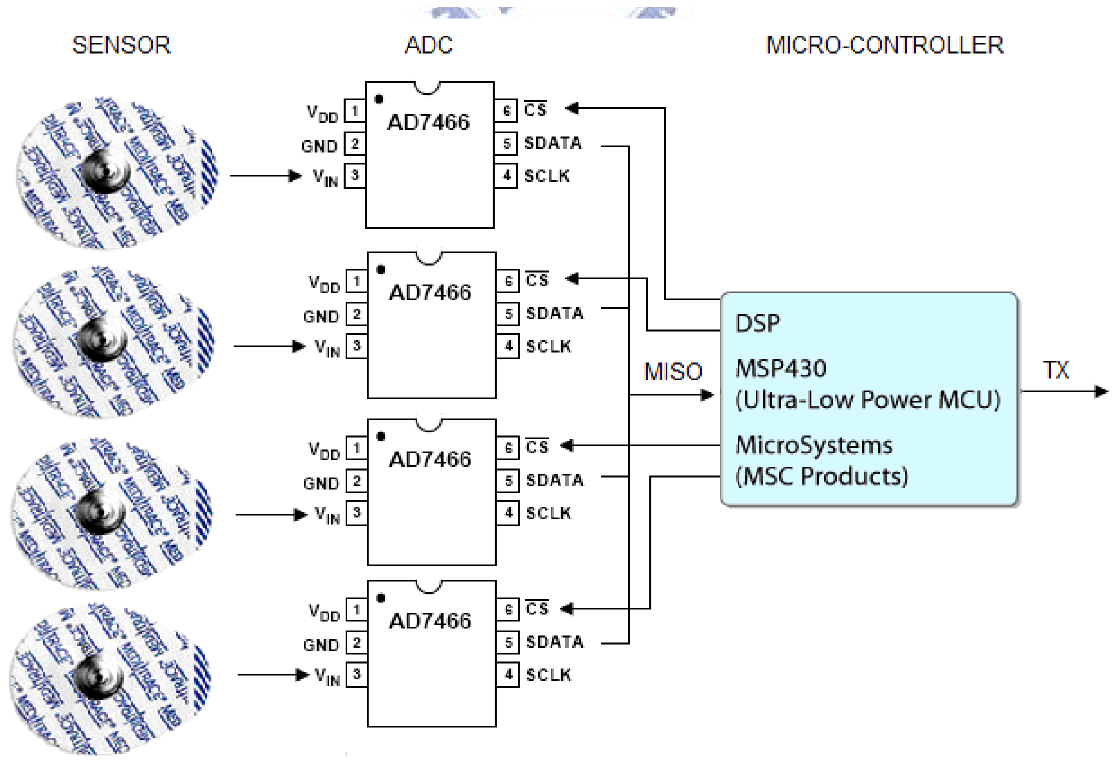


Fig. 4- 36 four channels brain-computer interface.

4-3-2 Wireless Transmission Model

Bluetooth is a standard and communications protocol primarily designed for low power consumption, with a short range in Table 4- 5 based on low-cost transceiver microchips in each device. Bluetooth enables these devices to communicate with each other when they are in range. The devices use a radio communications system, so they do not have to be in line of sight of each other, and can even be in other rooms, as long as the received transmission is powerful enough. Bluetooth device class indicates the type of device and the supported services of which the information is transmitted during the discovery process.

Table 4- 5 low-cost transceiver microchips

Class	**Maximum Permitted Power mW**	**Range (approximate)**
Class1	100 mW	~100 meters
Class2	2.5 mW	~10 meters
Class3	1 mW	~1 meter

Bluetooth exists in many products, such as telephones, printers, modems and headsets. The technology is useful when transferring information between two or more devices that are near each other in low-bandwidth situations. Bluetooth is commonly used to transfer sound data with telephones or byte data with hand-held computers. Bluetooth simplifies the discovery and setup of services between devices. Bluetooth devices advertise all of the services they provide. This makes using services easier because there is no longer a need to set up network addresses or permissions as in many other network.

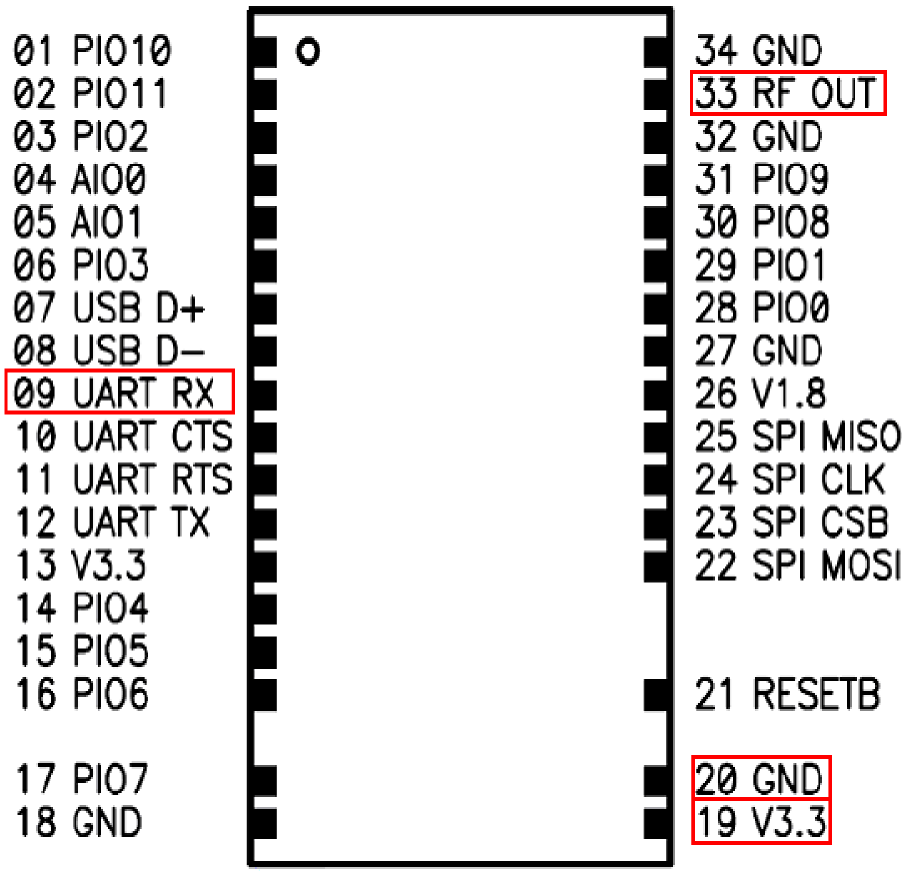


Fig. 4- 37 Pin Configuration of BM0203.

When we have defined the transmission protocol, it can conform to various application of transmission interface easily. We choose wireless transmission by using Bluetooth. BM0203 is an integrated Bluetooth module to ease the design gap and increase time-to-market performance. BM0203 uses CSR BuleCore4-External as the major Bluetooth chip. With simple commands to communicate with BM0203, the host does not need to worry about the details and complexity of Bluetooth profiles. The application allows Bluetooth object-transfer capability to be easily added to systems with no Bluetooth capability.

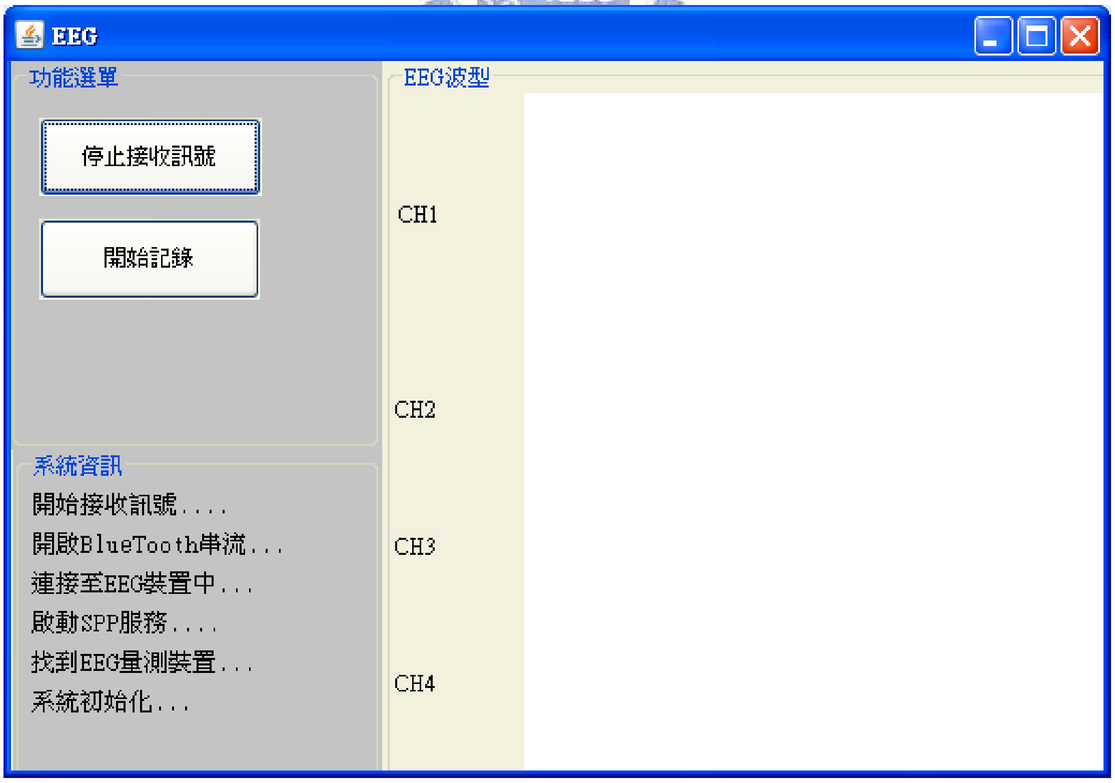
We use the pad to connect FPGA and Bluetooth show in Fig. 4- 37. The detail pad description in Table 4- 6.

Table 4- 6 Detail pad description

Power and Signal	Pad and Number	Pad Type	Description
GND	GND 20	Ground	Ground connections for digital
VCC3.3	V3.3 19	Regulator input	Voltage supplier from 2.8 to 4V
UART_RX	UART_RX 09	CMOS input with weak internal	UART data input
RF	RF OUT 33	RF	RF Output

4-3-3 GUI for Display

The EEG GUI (Graphical User Interface) develop by JAVA, it can receive data from Bluetooth. The receiver format stare with header FF and the channel data can be transmitted one by one in order. The button on the top is detection of Bluetooth device (SPP service) and stare Bluetooth streams. The GUI display in Fig. 4- 38.



開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

Fig. 4- 38 EEG Graphical User Interface.

4-4 Summary

In the fourth chapter, we discuss circuit design of the system. We design overall system up to down from software simulation of system level to real hardware implementation. In the hardware design, we assess the speed of system should have by software first. Then simplified whole system into a few individual and achieve them. However, in the error estimation we found that the fix-point calculation with more aliasing than floating point calculation. But we make a choice to reduce circuit area and accelerate the overall system speed. And the error tolerance is under our control. On the other hand, the transmission control protocol design tally with the front-end circuit (MSP430) interface particularly, and the back-end protocol also tally with wireless transmission method. In order to achieve faster computing, we using the precise symmetric look-up table and parallel computing operation. The specification of the hardware we developed show in Table 4- 7 which with input sample rate 64Hz, 128 times iteration in neural training, each training spends 8192 cycles, and system frequency is 68MHz with UART transmission interface.

Table 4- 7 System specification

Operate Frequency	**68MHz**
Sample Rate	**64Hz**
Gate Counts(million)	**0.315**
Operate Voltage	**3.3v**
Transmission Interface	**UART 115200bps**
Embedded Memory (M4K)	**24576bits**
ADC Resolution	**8-bits**

Chapter5
Experimental Results

In this section, we will show the real-time calculation result in GUI and comparison with other ICA design.

5-1 Result Super Gaussian BSS Methods in GUI

We compare software simulation result in chapter 3, and in this part we will verify the real ICA hardware results in real-time with GUI display. The post-simulation and off-line correlation has shown in Fig. 5- 1, Fig. 5- 2, Fig. 5- 3, and Fig. 5- 4 individually. In Fig. 5- 5(a) and Fig. 5- 6(a) show the GUI display of four channel mixed signals. In GUI display, we set the data bandwidth are 8-bit and the header is FF. However, in Fig. 5- 5(b) and Fig. 5- 6(b) show the ICA result in GUI display, we can find that if the original signals are pure super-Gaussian like this, the system will has a good result. Another way, we discuss EEG signal that has less information without ICA process apparently in Fig. 5- 7(a), and Fig. 5- 8(a). After ICA infomax update, the analysis signals will have more distinct information than original signals without ICA process. The ICA result of EEG signals shows in Fig. 5- 7(b), and Fig. 5- 8(b). In real-time ICA verification, we divided into two parts: super-Gaussian signals and EEG signal in real environment.

(1) Post-Simulation and Off-line Correlation

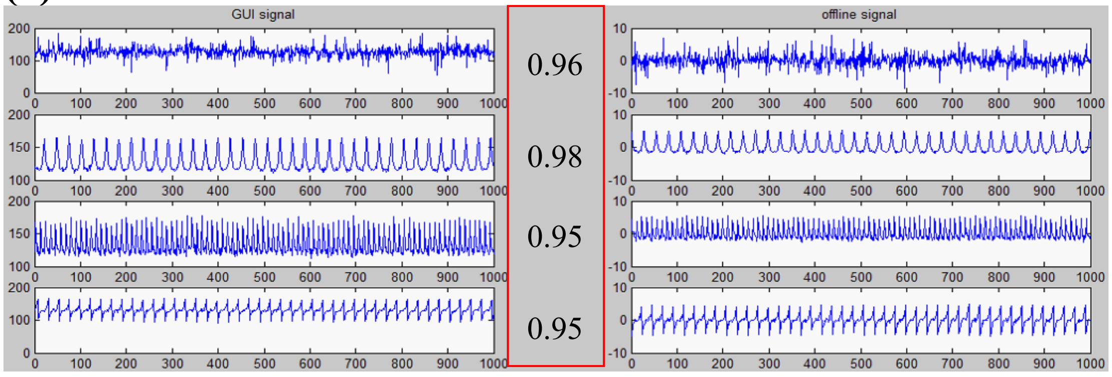


Fig. 5- 1 left is pattern1 post simulation, and right is offline ICA result.

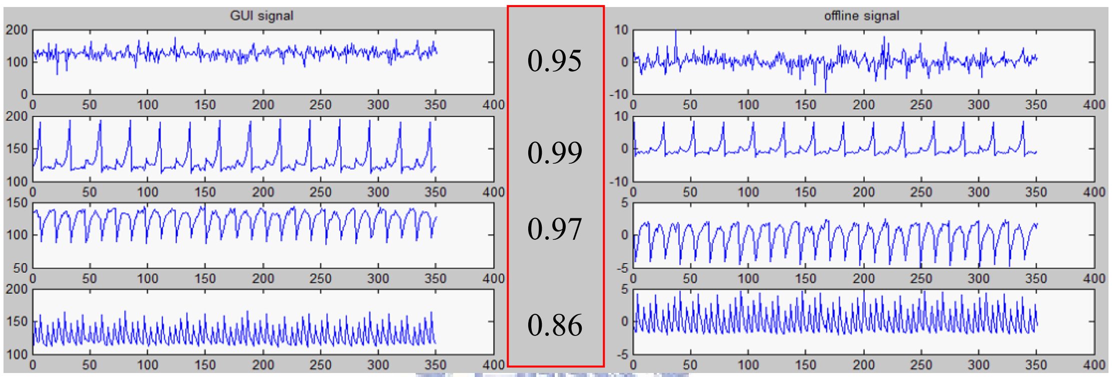


Fig. 5- 2 left is pattern2 post simulation, and right is offline ICA result.

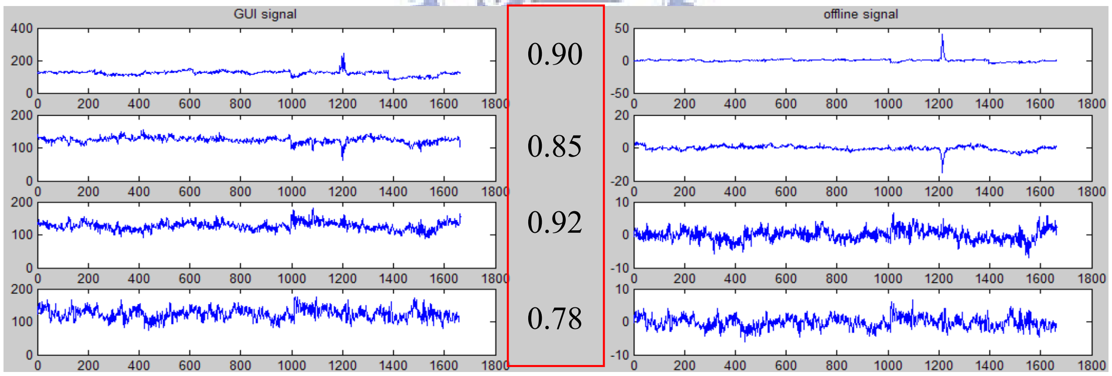


Fig. 5- 3 left is EEG1 post simulation, and right is offline ICA result.

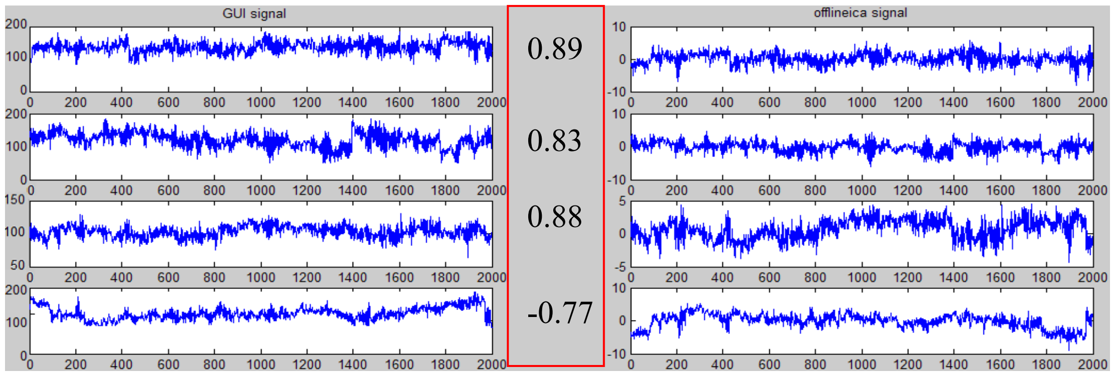
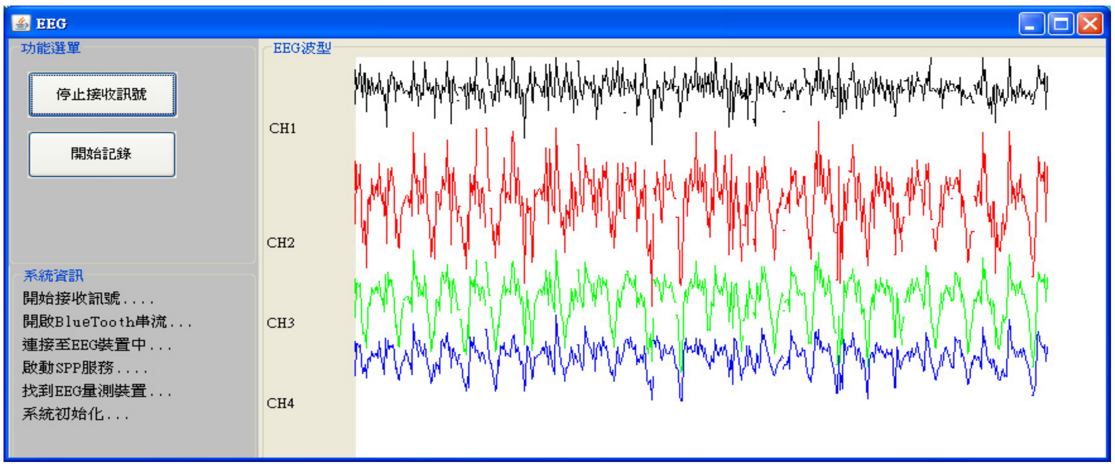


Fig. 5- 4 left is EEG2 post simulation, and right is offline ICA result.

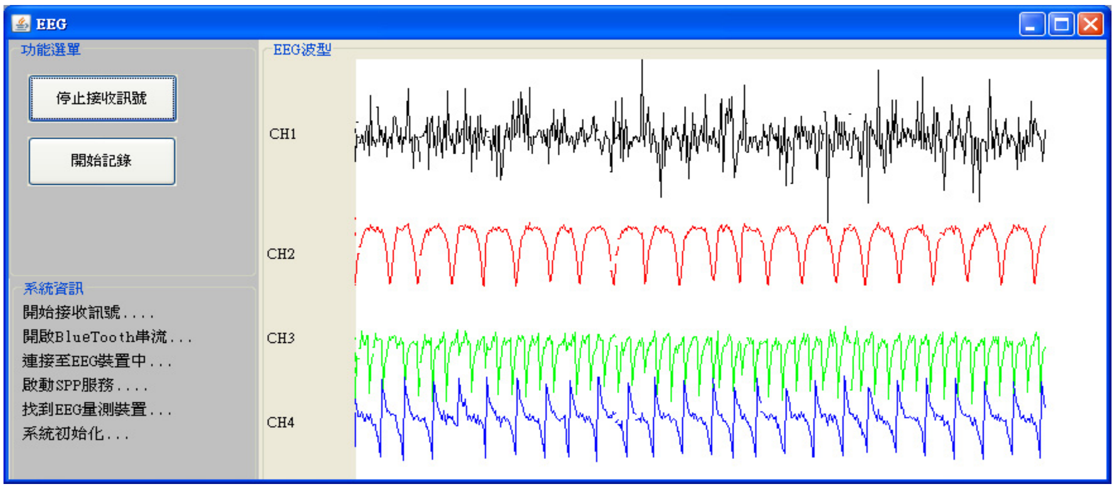
(2)Super Gaussian in GUI

1. Super Gaussian Pattern 1



開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(a)

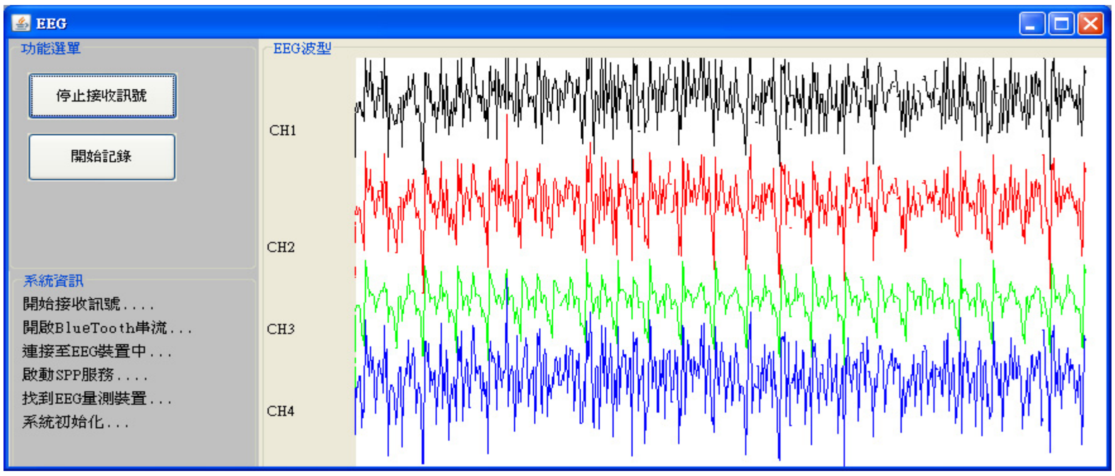


開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(b)

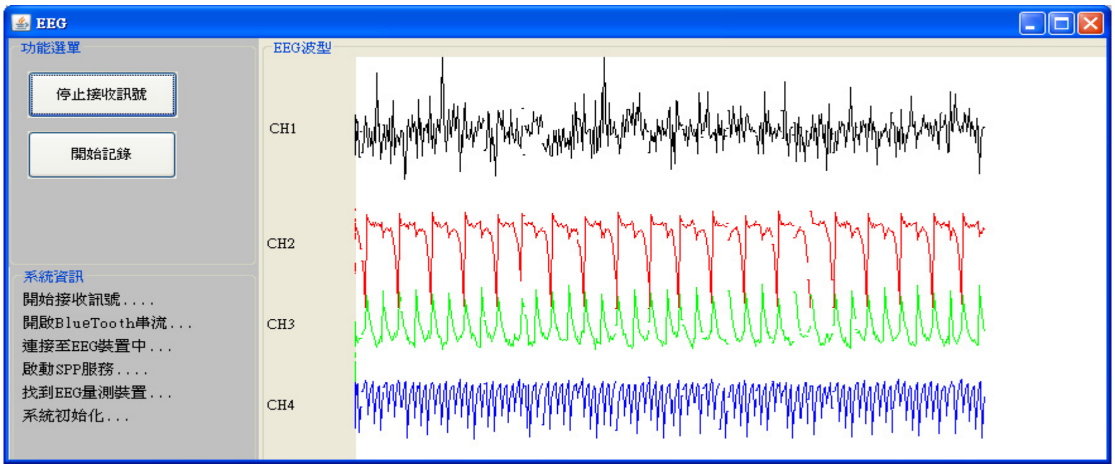
Fig. 5- 5 (a) mixed signal (b) ICA signal.

2. Super Gaussian Pattern 2



開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(a)



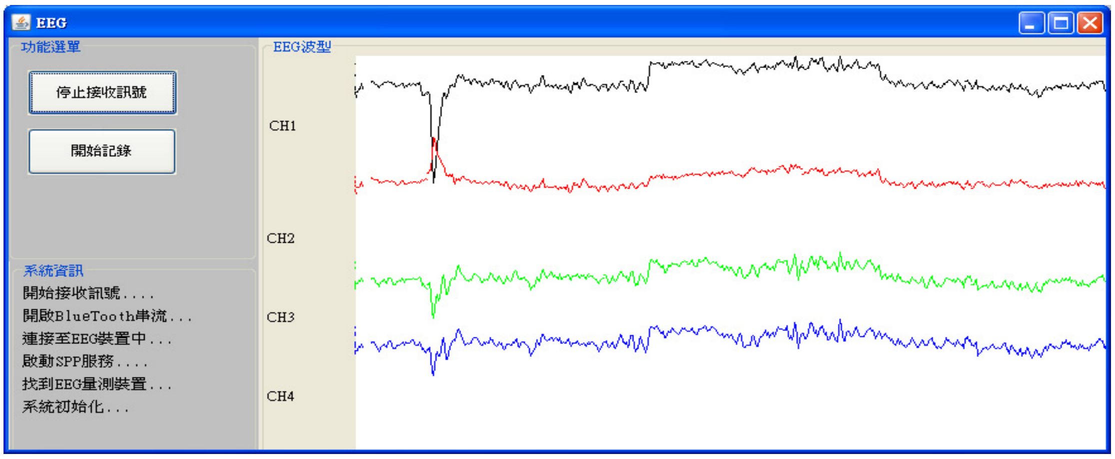
開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(b)

Fig. 5- 6 (a) mixed signal (b) ICA signal.

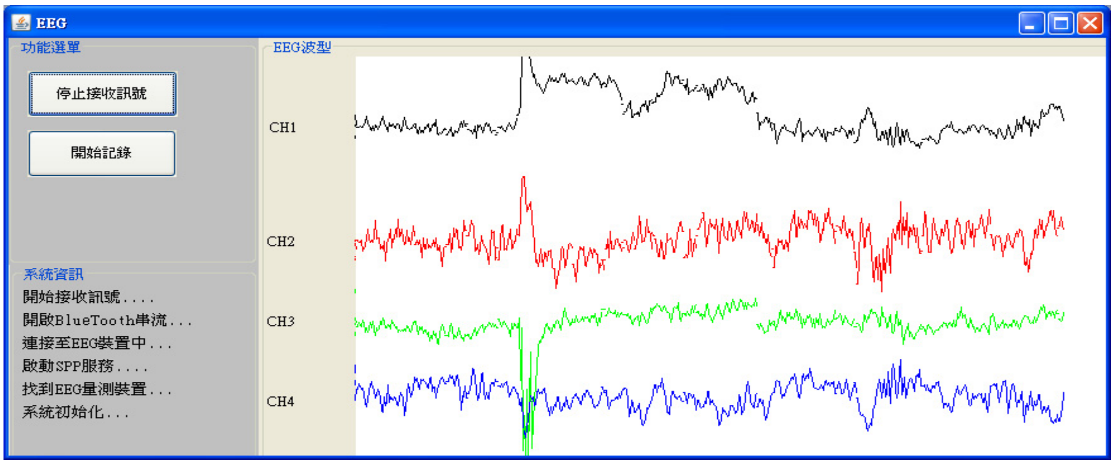
(3)EEG in GUI

1. EEG Pattern 1



開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(a)

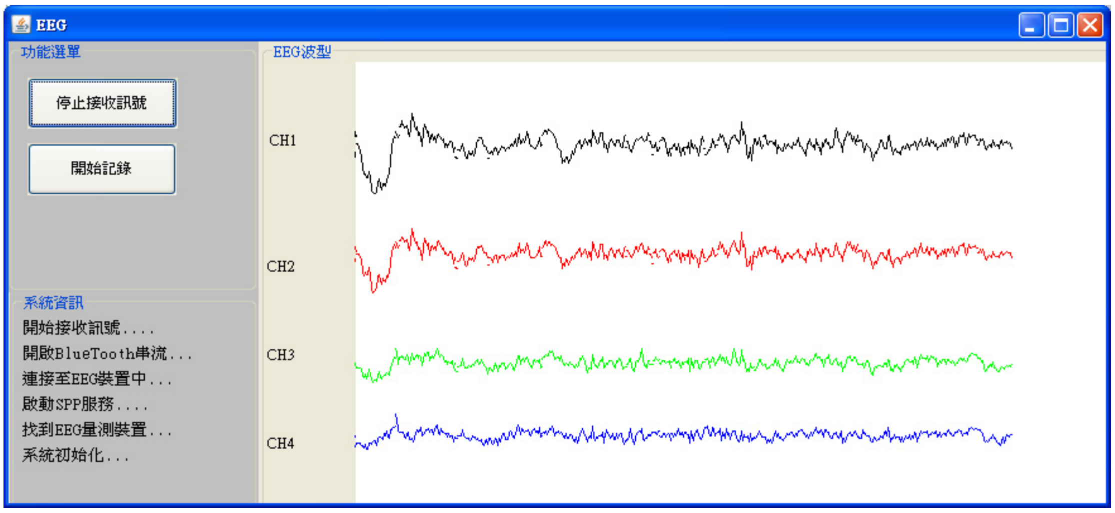


開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(b)

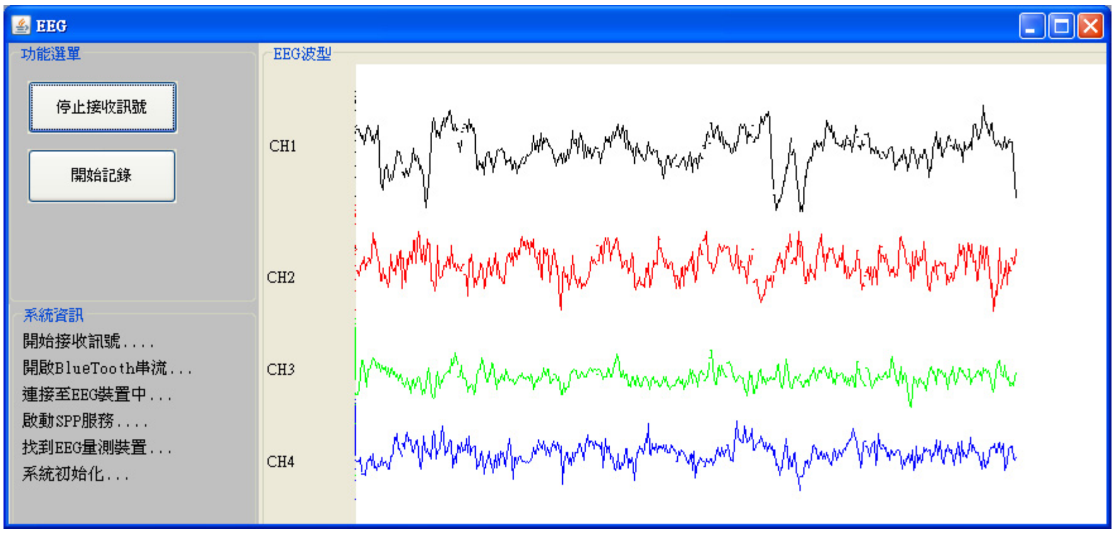
Fig. 5- 7 (a) mixed signal (b) ICA signal.

2. EEG Pattern 2



系統資訊
開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(a)



系統資訊
開始接收訊號....
開啟BlueTooth串流...
連接至EEG裝置中...
啟動SPP服務....
找到EEG量測裝置...
系統初始化...

(b)

Fig. 5- 8 (a) mixed signal (b) ICA signal.

5-2 Comparison with other ICA Design

There have been few studies about the real-time implementation of ICA which has been implemented as an ASIC by FPGA. In this thesis, we will discuss with the differentiation between our proposed and others in Table 5- 1.

In recent years, there have been few studies about the real-time implementation of ICA. In 2002, Scatter and Charayaphan [14] implement an ICA-based BSS algorithm on Xilinx Virtex E that contains 0.6 million logic gates. In 2004, Du and Qi [15] proposed an FPGA implementation of parallel ICA on a pilchard board, and used dual inline memory module (DIMM) random access memory (RAM) slot as an interface to communicate with central processing unit (CPU) and exchange data with memory in a SUN workstation. Charoensak and Sattar [16] proposed an FPGA design for real-time ICA-based BSS in 2005, using software to translate the high-level language, MATLAB Simulink, into hardware description language (HDL) code. And Pipelined FastICA[17] using the hardware floating-point (FP) arithmetic units to increase the numbers precision in 2008.

Table 5- 1 Comparison with other ICA design

Name	Application	Channel	Gate counts	Speed
Low cost 2002[14]	speech	2CH	0.6 million gates	20 MHz
Parallel ICA 2004[15]	hyperspectral image	N/A	0.226 million gates	20.1MHz
A Single-Chip FPGA 2005[16]	speech	2CH	0.1 million gates	71.2MHz
Pipelined FastICA 2008[17]	speech	2CH	N/A	50MHz
This work	EEG	4CH	0.315 million gates	68MHz

Chapter6 Conclusion

In this thesis, we had implemented four channels on-line ICA accompanied with flexible UART interface for real environment signal processing at 68MHz. We proposed integrated mathematics architecture allows high-speed real-time signal processing of Infomax ICA with sample rate up to 64Hz. Furthermore, the effective system controller and memory scheduling provide a high performance processing for real-time execution. And the system memory with enable signal is a low power method for portable application. The prototype demonstration in Fig. 6- 1, we have complete four channels EEG receive interface, Bluetooth wireless transmission and a GUI program for portable device. In our system, it is helpful for real-time biomedical monitor.

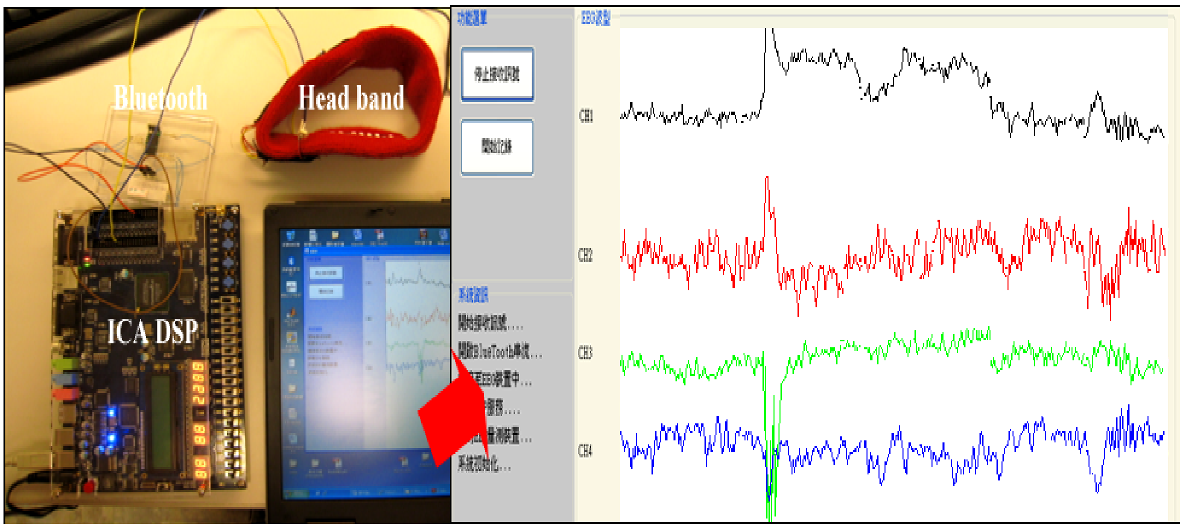


Fig. 6- 1 ICA prototype demonstration.

In the future, the design can be developed for two parts:

1. We can improve the operation precision and system error tolerance by floating point calculating in next generation..
2. We can integrate ICA into a system chip as an intellectual property. By integrating with ADC, micro processor, and ICA itself to achieve SOC design.
3. If the number of channels up to eight, it can be used more than one integrated operate modules for on-line processing, than the memory buffer and controller might be modified a little bit.
4. In practical applications, we can use the characteristics of real-time analysis to replace the thing which offline did and to achieve real-time detection processing. For example, immediately detect sleep signals to send a warning to driver. We use on-line ICA to substantially increase accuracy than without ICA process.

References

[1] T-W Lee, "Independent Component Analysis - Theory and Applications", Kluwer Academic Publishers, 1998.

[2] C. M. Kim and S. Y. Lee, “A digital chip for robust speech recognition in noisy environment,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 1089–1092, 2001.

[3] Saruwatari, H., Kawamura, T., Sawai, K.; Kaminuma, A., Sakata, M, “Blind source separation based on fast-convergence algorithm using ICA and beamforming for real convolutive mixture,” IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, pp. 13-17, 2002

[4] Ristaniemi, T.and Joutsensalo, J. “Advanced ICA-based receivers for DS-CDMA systems,” Personal, Indoor and Mobile Radio Communications, Vol. 1, pp. 276 -281, 2000

[5] A. J. Bell and T. J. Sejnowski, “An information maximization approach to blind separation and blind deconvolution,” *Neurocomputing,* Vol. 7, pp. 1129- 1159, 1995.

[6] K. Torkkola, "Blind separation of convolved sources based on information maximization", *IEEE Workshop Neural Networks for Signal Processing*, Kyoto, Japan, Sept 4-6, 1996.

[7] A. Hyvärinen and E. Oja. “A Fast Fixed-Point Algorithm for Independent Component Analysis,” Neural Computation, Vol. 9, pp. 1483-1492, 1997

[8] A. Hyvärinen. “Fast and Robust Fixed-Point Algorithms for Independent Component Analysis.” IEEE Transactions on Neural Networks 10(3), 626-634, 1999.

[9] C. Jutten and J. Herault, “Blind Separation of Sources 1. An Adaptive Algorithm Based on Neuromimetic Architecture,” *Signal Processing*, Vol. 24, pp 1-10, 1991.

[10] J.F. Cardoso, A. Souloumiac, “Blind Beamforming for Non-Gaussian Signals,” *IEE Proc.F* Vol. 140, pp. 362-370, 1993.

[11] A. Hyvärinen and E. Oja. “A Fast Fixed-Point Algorithm for Independent Component Analysis,” Neural Computation, Vol. 9, pp. 1483-1492, 1997

[12] C. Charoensak, and F. Sattar, "System-level design of low-cost FPGA hardware for real-time ICA-based blind source separation," SOC Con. 2004. Proceedings. IEEEInternat ional, pp. 139 - 140, 2004.

[13] H. Amin., K.M. Curtis, and B.R. Hayes-Gill, “Piecewise Linear Approximation Applied to Nonlinear Function of a Neural Network, ” *IEE Proc. Crcuits Divices syst,* Vol. 144, pp. 313-3171, 1997

[14] F. Sattar and C. Charayaphan, *Low-cost design and implementation of an ICA-based blind source separation algorithm*, 15th Annual IEEE International ASIC/SOC Conference, pp.15-19, 2002.

[15] H. Du and H. Qi, “An FPGA implementation of parallel ICA for dimensionality reduction in hyperspectral images,” in *Proc. IEEE Int. Symp. Geosci. Remote Sens.*, Sep. 2004, vol. 5, pp. 3257–3260.

[16] C. Charoensak and F. Sattar, “A single-chip FPGA design for real-time ICA-based blind source separation algorithm,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Vol. 6, pp. 5822–5825, 2005.

[17] Kuo-Kai Shyu and Ming-Huan Lee, “Implementation of Pipelined FastICA on FPGA for Real-Time Blind Source Separation” in *Proc IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 19, 2008