Energy-Efficient FastICA Implementation for Biomedical Signal Separation

Lan-Da Van, *Member, IEEE*, Di-You Wu, and Chien-Shiun Chen

***Abstract*— This paper presents an energy-efficient fast independent component analysis (FastICA) implementation with an early determination scheme for eight-channel electroencephalogram (EEG) signal separation. The main contributions are as follows: 1) energy-efficient FastICA using the proposed early determination scheme and the corresponding architecture; 2) cost-effective FastICA using the proposed preprocessing unit architecture with one coordinate rotation digital computer-based eigenvalue decomposition processor and the proposed one-unit architecture with the hardware reuse scheme; and 3) low-computation-time FastICA using the four parallel one-units architecture. The resulting power dissipation of the FastICA implementation for eight-channel EEG signal separation is 16.35 mW at 100 MHz at 1.0 V. Compared with the design without early determination, the proposed FastICA architecture implemented in united microelectronics corporation 90 nm 1P9M complementary metal–oxide–semiconductor process with a core area of 1.221 × 1.218 mm^2 can achieve average energy reduction by 47.63%. From the post-layout simulation results, the maximum computation time is 0.29 s.**

***Index Terms*— Blind source separation, electroencephalogram, energy efficiency, fast independent component analysis, hardware implementation.**

I. Introduction

INDEPENDENT component analysis (ICA) has been widely used to solve the problem of the blind source separation (BSS) with the applications to speech, image or biomedical signal processing [1]–[3]. On the other hand, since many scientists need to observe the corresponding pure brain activities, the signals of electroencephalogram (EEG), functional magnetic resonance imaging and magnetoencephalogram (MEG) can be analyzed by the ICA algorithm in the brain research [4]–[6]. In the previous analyses [5], the signal characteristics of EEG and MEG are conformed to independent components processed by the ICA algorithm. The ICA approach enables us to project each independent component onto a multiple-dipole map for EEG signals [3].

In terms of algorithms, there exist many ICA-related approaches [6]–[15]. The information-maximization (INFOMAX) algorithm [7] derived from higher-order statistics possesses high computation complexity. On the other hand, the fast ICA (FastICA) algorithm [8]–[12] based on approximate negentropy and Newton iteration can reduce the computation. A comparative study of ICA algorithms for brain computer interface (BCI) systems [6] shows that the FastICA algorithm has good quality of the extracted component. Recently, a new contrast function [14] for ICA can avoid the permutation ambiguity and has better separation quality than that of the conventional ICAs. The RobustICA algorithm [15] shows better extraction quality than FastICA by iteratively maximizing the Kurtosis contrast function with algebraic optimal step size, and has efficient computational cost required to reach a given source extraction quality. Beyond the scope of this paper, some non-ICA-based BSS methods [16], [17] are used to extract one global signal [16] and to separate dependent sources [17]. However, the above literatures do not focus on ICA hardware implementation.

In terms of implementations, a nice survey using very large-scale integration (VLSI) approaches has been described in [18]. Although some ICAs are implemented by analog [19], [20] or mixed-signal [21] approaches, these approaches either cost much deign time or mostly not exhibit the same behavior as they do in the ideal simulation. Thus, field programmable gate array (FPGA) and application specific integrated circuit (ASIC) design methodology will be a promising approach. Several FPGA implementations of the ICA algorithm have been proposed in the literature. Kim *et al.* [22] proposed the FPGA implementation of the ICA algorithm constructed by the adaptive noise canceling (ANC) module for 2-channel BSS. The power dissipation of the 2-channel ANC module is 98.8 mW at 12.288 MHz and 1.8 V in FPGA. Du *et al.* [23] proposed the parallel ICA (pICA) algorithm and the corresponding FPGA implementation on a pilchard board. The parallel ICA based on FastICA divides the process into several sub-processes to achieve the single program multiple data parallelism. In [24], Du *et al.* proposed the corresponding pICA ASIC deign with four weight vectors. Charoensak *et al.* [25] provided an FPGA design of the ICA-based BSS. This FPGA implementation is translated from the high-level language in MATLAB into the hardware description language. However, the translated code from the high level language usually cannot lead to an optimized register-transfer-level code and the corresponding hardware performance may become worse. A pipelined FastICA design [26] adopts the floating-point arithmetic unit to increase the precision. An INFOMAX ICA design in FPGA for the four-channel EEG signal separation [27] uses the fixed-point arithmetic; however, the resulting BSS quality can be improved.

Manuscript received September 3, 2010; revised April 29, 2011, July 29, 2011, and August 15, 2011; accepted August 17, 2011. Date of publication October 3, 2011; date of current version November 2, 2011. This work was supported in part by the National Science Council (NSC) under Grant NSC-99-2911-I-009-101, Grant NSC-99-2220-E-009-029, and Grant NSC-100-2220-E-009-058.

The authors are with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: ldvan@cs.nctu.edu.tw; dywu@viplab.cs.nctu.edu.tw; cs@viplab.cs.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2166979

Since BCI with ICA computation is demanded to control an external device on-line and to monitor the biomedical signals, the low computation time is desired. On the other hand, due to the energy limitation of a portable device, the low-energy system is expected. In order to support the portable demand and achieve the on-line feedback, a low-energy and low-computation-time ICA implementation is required. However, the existing FPGA-based ICA implementations either consume huge power or have high area cost/computational time. Thus, one energy-efficient cost-effective low-computation-time FastICA chip which is suitable to be embedded in a portable device with satisfactory BSS quality of the mixed signals or the EEG component is desired. Note that, in this paper, the BSS quality of the hardware implementation is evaluated by the absolute correlation coefficient of the mixed signals or the EEG component compared with simulation results in MATLAB. The originality of the target design is described in the following.

1) To our best knowledge, the proposed energy-efficient, low-area and low-computation-time eight-channel FastICA architecture and chip layout implementation with satisfactory BSS quality should be the first one for EEG signal separation.

The originality of the proposed architectures is summarized as follows.

1) The early determination scheme and the corresponding architecture are proposed to save the energy.
2) The low-area preprocessing unit architecture using one CORDIC-based eigenvalue decomposition (EVD) processor is proposed in the preprocessing part.
3) The low-area one-unit architecture using the hardware reused scheme is proposed in the fixed-point iteration part.

In addition, the four parallel one-units architecture is used to lower the computation time. Note that the four parallel one-units architecture is different from that of [23] and [24] due to the proposed low-area one-unit architecture. As a result, the proposed energy-efficient eight-channel FastICA architecture consumes 16.35 mW at 100 MHz with the core size of 1.221×1.218 mm^2 in united microelectronics corporation (UMC) 90 nm CMOS process. The maximum computation time of the FastICA implementation is 0.29 s. Compared with the reference design without using the early determination scheme, the energy reduction by 39.06% and 47.63% of mixed signals and EEG signals, respectively, can be attained.

This paper is organized as follows. A brief review of the FastICA algorithm is described in Section II. In Section III, the low-energy, area-cost-effective and low-computation-time VLSI architecture of the FastICA algorithm is proposed. In Section IV, we show the software simulation results and corresponding post-layout simulation results for the validity of the FastICA implementation. Finally, the conclusion is remarked in the last section.

II. Background of the FastICA Algorithm

A BSS system with n blind source signals is defined as

$$\mathbf{X} = \mathbf{AS} \tag{1}$$

where $\mathbf{A}$ is an n by n mixing matrix. $\mathbf{X}$ and $\mathbf{S}$ are expressed in (2) and (3), respectively

$$\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_n]^T \tag{2}$$

$$\mathbf{S} = [\mathbf{s}_1\ \mathbf{s}_2\ \ldots\ \mathbf{s}_n]^T \tag{3}$$

where $\mathbf{X}$ is a matrix with n observed mixed signal vectors, and $\mathbf{S}$ is a matrix with n blind source signal vectors that are statistically independent and no more than one signal is Gaussian distributed. $\mathbf{x}_m$ and $\mathbf{s}_m$ are expressed in (4) and (5), respectively

$$\mathbf{x}_m = [x_m(1)\ x_m(2)\ \ldots\ x_m(i)]^T,\ \text{for } m = 1, 2, 3, \ldots, n \tag{4}$$

$$\mathbf{s}_m = [s_m(1)\ s_m(2)\ \ldots\ s_m(i)]^T,\ \text{for } m = 1, 2, 3, \ldots, n \tag{5}$$

where $x_m(i)$ and $s_m(i)$ denote a mixed signal and a source signal at a discrete time i, respectively. The goal of the ICA is to recover the source signal $\mathbf{S}$. In order to achieve the purpose, the ICA algorithm estimates the matrix $\mathbf{A}$ by observing the matrix $\mathbf{X}$ and computes the weight matrix $\mathbf{W}^T$ that is equal to the inverse of matrix $\mathbf{A}$. Subsequently, the blind source signal $\mathbf{S}$ can be obtained by the ICA unmixing model as described below

$$\mathbf{S} = \mathbf{W}^T\mathbf{X} \tag{6}$$

where

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & & w_{1n} \\ w_{21} & w_{22} & & w_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{n1} & w_{n2} & & w_{nn} \end{bmatrix}.$$

In order to estimate one of the independent components, a linear combination $\mathbf{w}^T\mathbf{X}$ needs to be considered, where $\mathbf{w}$ is a column vector of the matrix $\mathbf{W}$. In other words, one of the independent components can be obtained by maximizing the non-Gaussianity of $\mathbf{w}^T\mathbf{X}$. The FastICA algorithm proposed by Hyvärinen-Oja in [2] and [8]–[11] can successfully extract a wide class of non-Gaussian source signals. The FastICA algorithm consists of two steps. One is the preprocessing step and the other is the fixed-point algorithm.[1] The corresponding descriptions of the steps are stated as follows.

A. Preprocessing of FastICA

The aim of the preprocessing is to center and whiten the mixed signals. That means the mixed signals with zero mean and unit variance can be obtained through this preprocessing. The centering expression of the preprocessing part can be written as follows:

$$\bar{x}_m(i) = x_m(i) - E\{x_m\},\ \text{for } m = 1, 2, 3, \ldots, n \tag{7}$$

where $\bar{x}_m(i)$ and $E\{x_m\}$ denote the mixed signal with zero mean and expected value of the random variable $x_m(i)$ of $\mathbf{x}_m$, respectively. After subtracting the mean value, $\bar{\mathbf{x}}$ is a

[1] The fixed-point algorithm is named in [2] and [9].

zero-mean vector. Due to the number of n random vectors (i.e., $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, …, $\mathbf{x}_n$), the centering matrix can be expressed as below

$$\bar{\mathbf{X}} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \\ \bar{\mathbf{x}}_2^T \\ \vdots \\ \bar{\mathbf{x}}_n^T \end{bmatrix} = \begin{bmatrix} \bar{x}_1(1)\ \bar{x}_1(2)\ \cdots\ \bar{x}_1(i) \\ \bar{x}_2(1)\ \bar{x}_2(2)\ \cdots\ \bar{x}_2(i) \\ \vdots \quad \vdots \quad \cdots \quad \vdots \\ \bar{x}_n(1)\ \bar{x}_n(2)\ \cdots\ \bar{x}_n(i) \end{bmatrix}. \tag{8}$$

The EVD can be used to decompose the covariance matrix of $\bar{\mathbf{X}}$ and the corresponding operation is expressed as below

$$\mathbf{C_X} = E\left\{\bar{\mathbf{X}}\ \bar{\mathbf{X}}^T\right\} = \mathbf{EDE}^T \tag{9}$$

where $\mathbf{E}$ consisting of eigenvectors denotes the orthogonal matrix of $\mathbf{C_X}$ and $\mathbf{D}$ in (10) represents the diagonal matrix of $\mathbf{C_X}$

$$\mathbf{D} = diag(d_1, d_2, \ldots, d_n) \tag{10}$$

where $d_1, d_2, \ldots, d_n$ denote the eigenvalues of $\mathbf{C_X}$. The whitening process of $\bar{\mathbf{X}}$ is expressed below

$$\mathbf{Z} = \mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T\bar{\mathbf{X}} = \mathbf{P}\bar{\mathbf{X}} \tag{11}$$

where $\mathbf{P}$ equals $\mathbf{D}^{-1/2}\mathbf{E}^T$ and denotes the whitening matrix of $\bar{\mathbf{X}}$. The centered matrix $\bar{\mathbf{X}}$ is linearly transformed to a matrix $\mathbf{Z}$ such that the covariance matrix of $\mathbf{Z}$ equals the identity matrix. That means the matrix $\mathbf{Z}$ is uncorrelated, where the corresponding proof is released in [26].

B. Fixed-Point Algorithm

In Section II-A, the preprocessing part including centering and whitening has been illustrated. Next, the fixed-point algorithm is required for the weight training. Herein, the one-unit operation [2] of the fixed-point algorithm for the FastICA algorithm is discussed. One-unit operation can be regarded to update an artificial neuron with a weight vector $\mathbf{w}$ by a learning rule. In [2], non-Gaussianity is measured by the approximation of negentropy $J(\mathbf{w}^T\mathbf{X})$ as shown below

$$J(y) \propto [E\{G(y)\} - E\{G(v)\}]^2 \tag{12}$$

where G is a non-quadratic function and is defined as follows:

$$G(u) = \frac{1}{a}\log\cosh(au) \tag{13}$$

where a denotes a constant parameter. The FastICA algorithm based on the fixed-point iteration scheme is to find the maximum of the non-Gaussianity of $\mathbf{w}^T\mathbf{X}$ as measured by negentropy. The unit vector $\mathbf{w}$ is substituted into the projection $\mathbf{w}^T\mathbf{X}$ such that the negentropy is maximized. The fixed-point iteration operations [2], [9] of the FastICA algorithm using an approximate negentropy and Newton iteration are addressed as follows.

Step 1: Choose an initial (e.g., random) vector $\mathbf{w}$ with unit norm.

Step 2: Calculate $\mathbf{w}^+ = E\{\mathbf{Z}[g(\mathbf{w}^T\mathbf{Z})]^T\} - E\{g'(\mathbf{w}^T\mathbf{Z})\}\mathbf{w}$.

Step 3: Calculate $\mathbf{w} = \mathbf{w}^+ / \left\|\mathbf{w}^+\right\|$.

Step 4: If not converged, go back to Step 2

where g is the derivative of the non-quadratic function G. When the old and new vectors $\mathbf{w}$ are in the same direction, the learning converges and the absolute dot-product value of two vectors is close to 1. To avoid different vectors converging to the same maxima, the vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$ are needed to orthogonalize before each iteration. The deflation scheme [2], [9] is a simple way to orthogonalize the weight vectors since the deflation scheme estimates each independent component one by one at each iteration step based on Gram–Schmidt orthonormalization. Gram–Schmidt orthonormalization for the $(k+1)$-th component as expressed below

$$\mathbf{w}_{k+1}^+ = \mathbf{w}_{k+1} - \sum_{j=1}^{k}(\mathbf{w}_{k+1}^T\mathbf{w}_j)\mathbf{w}_j \tag{14}$$

$$\mathbf{w}_{k+1} = \frac{\mathbf{w}_{k+1}^+}{\left\|\mathbf{w}_{k+1}^+\right\|} \tag{15}$$

where a new weight vector $\mathbf{w}_{k+1}$ is obtained by subtracting the vector projected from the old weight vector. We perform the fixed-point algorithm for $\mathbf{w}_{k+1}$ such that k independent components or k vectors including $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k\}$ are estimated and $\mathbf{w}_{k+1}$ is orthogonalized with other vectors in (14) at each iteration.

C. Fixed-Point Algorithm with Loop Unrolling

In Section II-B, the independent components are estimated one by one using the deflation approach. Assume the number of channels, n, is a multiple of four, and $\overline{\mathbf{W}} = [\mathbf{w}_1\ \mathbf{w}_2\mathbf{w}_3 \ldots \mathbf{w}_{n-1}\ \mathbf{w}_n]$, where $\mathbf{w}_i$, for $i = 1, 2, 3, \ldots, n$, denotes the column vector. In order to reduce the computation time, the loop for the one-unit operation can be unrolled by four such that the independent components are estimated in parallel [10]. Thus, the fixed-point algorithm with unrolling the loop of the one-unit operation by four for n channels is addressed as follows.

Step 1: Set initial n vectors with unit norm (i.e., $\mathbf{w}_i$ for $i = 1, 2, 3, \ldots, n$), and $j = 0$.

Step 2: Unroll the loop for the one-unit processing by four at the j-th loop

$$\mathbf{w}_{4j+1}^+ = E\{\mathbf{Z}[g(\mathbf{w}_{4j+1}^T\mathbf{Z})]^T\} - E\{g'(\mathbf{w}_{4j+1}^T\mathbf{Z})\}\mathbf{w}_{4j+1}$$

$$\mathbf{w}_{4j+2}^+ = E\{\mathbf{Z}[g(\mathbf{w}_{4j+2}^T\mathbf{Z})]^T\} - E\{g'(\mathbf{w}_{4j+2}^T\mathbf{Z})\}\mathbf{w}_{4j+2}$$

$$\mathbf{w}_{4j+3}^+ = E\{\mathbf{Z}[g(\mathbf{w}_{4j+3}^T\mathbf{Z})]^T\} - E\{g'(\mathbf{w}_{4j+3}^T\mathbf{Z})\}\mathbf{w}_{4j+3}$$

$$\mathbf{w}_{4j+4}^+ = E\{\mathbf{Z}[g(\mathbf{w}_{4j+4}^T\mathbf{Z})]^T\} - E\{g'(\mathbf{w}_{4j+4}^T\mathbf{Z})\}\mathbf{w}_{4j+4}.$$

Step 3: If $j = (n/4) - 1$, go to Step 4. Otherwise, increase j by one and go back to Step 2.

Step 4: $\overline{\mathbf{W}}$ is processed sequentially by Gram–Schmidt orthonormalization.

Step 5: If satisfying the convergence threshold or reaching the maximum iteration, the fixed-point algorithm process is terminated. Otherwise, go back to Step 2 with resetting $j = 0$. When four one-unit processes are computed in parallel, the number of iterations for n weight vectors estimation can be reduced to $n/4$ times due to Step 2 process. In Step 5, the convergence threshold is a predetermined value and is used to

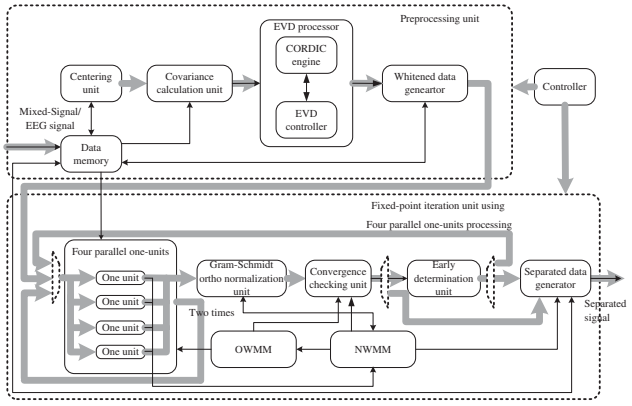

Fig. 1. System diagram of the proposed eight-channel FastICA architecture.

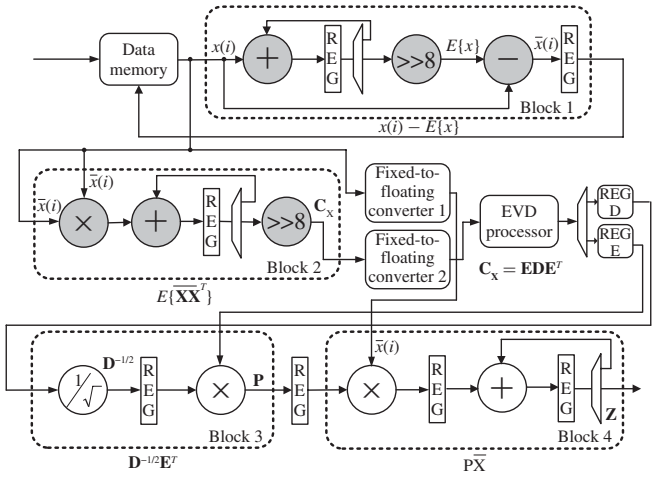

Fig. 2. Block diagram of the proposed preprocessing unit.

compare with the sum of absolute dot-products (SAD). The function of the SAD value is defined as

$$SAD = \sum_{i=1}^{n} abs(\mathbf{w}_i^T \mathbf{w}_i^+) \tag{16}$$

where *abs* denotes an absolute operator and each dot-product value is obtained by calculating the inner product of an old weight vector and a new weight vector. At each iteration, the matrix $\overline{\mathbf{W}}$ is substituted into the projection $\overline{\mathbf{W}}^T\mathbf{X}$ such that the negentropy is maximized. Finally, the maximum of the non-Gaussianity of $\overline{\mathbf{W}}^T\mathbf{X}$ can be estimated when achieving the convergence.

III. Energy-Efficient and Cost-Effective FastICA Architecture

In this section, an energy-efficient cost-effective eight-channel FastICA architecture as shown in Fig. 1 is designed for mixed signal and EEG signal separation, where the gray-line arrows represent the main block control precedence and the black-line arrows denote the data flow. The proposed FastICA architecture consists of two parts: the preprocessing unit and the fixed-point iteration unit using four parallel one-units. According to the algorithm, the preprocessing unit and the fixed-point iteration unit are operated sequentially such that the same data memory can be shared. On the other hand, two matrix memories are required for the fixed-point algorithm. One is to store an old weight matrix and another is to keep a new weight matrix. Without loss of generality, we use eight channels and 256 samples per channel to demonstrate the operations of the proposed energy-efficient FastICA architecture as shown in Fig. 1. First, the input data are stored in the data memory. Second, data are fetched from the data memory to perform centering through the centering unit and the processed data are written back to the data memory. Third, data are fetched from the data memory to calculate covariance through the covariance calculation unit and the treated data are sent to the EVD processor to calculate the eigenvalue and eigenvector. Fourth, the whitened data produced by the whitened data generator are written back to the data memory. Thus, the preprocessing process is completed. Next, data fetched from the data memory and the old weight matrix memory (OWMM) are fed to the four parallel one-units to perform four one-unit operations by two loops/times. Through Gram–Schmidt orthonormalization unit, the resulting data are written back to the new weight matrix memory (NWMM). Through the convergence checking unit, the convergence can be detected. On satisfying the convergence threshold or reaching the maximum iteration, the fixed-point iteration process is terminated. Otherwise, the early determination unit determines whether the difference between an old SAD value and a new SAD value is small enough. If the difference value is small enough, the iteration process can be terminated for saving energy consumption. Otherwise, go back to the four parallel one-units. Finally, the separated signals are obtained and written back to the data memory. Thus, the FastICA training is finished.

A. Implementation of the High-Dimensional and Low-Area Preprocessing Unit

In the preprocessing unit, the proposed architecture is divided into four blocks and one EVD processor as shown in Fig. 2. Since EEG signals captured by sensors are digitized via an analog-to-digital converter, the inputs of the BSS system are fixed-point EEG signals. In order to increase the computational accuracy, the floating-point operation is adopted after the centering and covariance operations. Thus, a fixed-point operation is performed in the gray computation elements of Block 1 and Block 2. The EVD processor is implemented by one CORDIC engine. Block 3 and Block 4 perform floating-point operation. The detailed operations of each block are described in the following.

1) Centering: Centering is to force signals with zero mean. The centering operation in (7) can be recast as follows:

$$\bar{x}(i) = x(i) - E\{x\} = x(i) - \left(\frac{\sum_{j=1}^{256} x(j)}{256}\right)$$
$$= x(i) - \left(\sum_{j=1}^{256} x(j)\right) >> 8 \tag{17}$$

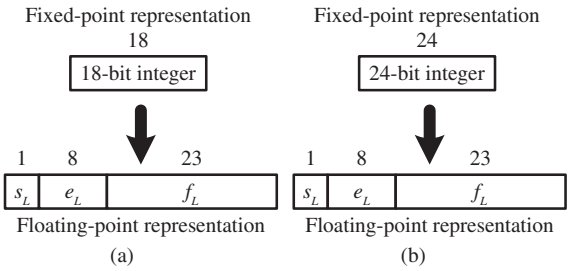

Fig. 3. Illustration of (a) fixed-to-floating converter 1 and (b) fixed-to-floating converter 2.

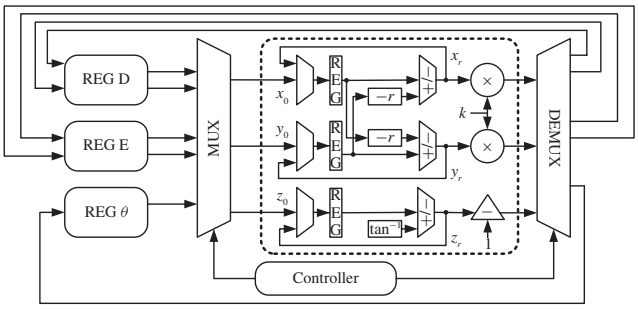

Fig. 4. EVD Processor using one CORDIC engine.

where $i = 1, 2, \ldots, 256$ and $>>8$ denotes a right-shifted eight-bit operator. The term on the right hand side in (17) can be realized in Block 1 of Fig. 2. Note that the input bit width of $x(i)$ and output bit width of $\bar{x}(i)$ are 12 and 18 bits, respectively, in Block 1 of the preprocessing unit. The detailed operations are described in the following. The input signals are stored in the one-port data memory. Data are read sequentially from the data memory, and the elements of $\mathbf{x}$ are accumulated one by one and channel by channel. After accumulating 256 element values of one channel, a right-shifted eight-bit operation is utilized instead of division to obtain the mean of the mixed signals for each channel. The next step is to read the same mixed signals again from the data memory and subtract the mean of each channel. After the results are written back to the data memory, the centering operation is finished.

2) Whitening: The main task of the preprocessing unit is to whiten the mixed signals. The first step is to calculate the covariance matrix of $\bar{\mathbf{X}}$ in Block 2 of Fig. 2, where $\bar{\mathbf{X}}$ and the corresponding covariance matrix are defined in (8) and (18), respectively

$$\mathbf{C_X} = E\{\bar{\mathbf{X}}\ \bar{\mathbf{X}}^T\} = \frac{1}{256}\begin{bmatrix} \bar{\mathbf{x}}_1^T\bar{\mathbf{x}}_1 & \bar{\mathbf{x}}_1^T\bar{\mathbf{x}}_2 & \ldots & \bar{\mathbf{x}}_1^T\bar{\mathbf{x}}_8 \\ \bar{\mathbf{x}}_2^T\bar{\mathbf{x}}_1 & \bar{\mathbf{x}}_2^T\bar{\mathbf{x}}_2 & \ldots & \bar{\mathbf{x}}_2^T\bar{\mathbf{x}}_8 \\ \vdots & \vdots & \ldots & \vdots \\ \bar{\mathbf{x}}_8^T\bar{\mathbf{x}}_1 & \bar{\mathbf{x}}_8^T\bar{\mathbf{x}}_2 & \ldots & \bar{\mathbf{x}}_8^T\bar{\mathbf{x}}_8 \end{bmatrix}. \quad (18)$$

The covariance matrix of mixed signals is a real symmetric matrix due to the results of $\bar{\mathbf{x}}_1^T \cdot \bar{\mathbf{x}}_2 = \bar{\mathbf{x}}_2^T \cdot \bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_1^T \cdot \bar{\mathbf{x}}_3 = \bar{\mathbf{x}}_3^T \cdot \bar{\mathbf{x}}_1$ and so forth. Thus, only 36 elements are needed to be calculated. A multiplier and one accumulator are used to calculate all elements of $\mathbf{C_X}$. Since one-port data memory is used, only one data can be obtained at one time. Thus, two data streams are retrieved from memory after three cycles and then to perform the multiplication. After 256 multiplications and accumulations, the right-shifted eight-bit can achieve the division of 256. In order to satisfy the more accurate requirement of EEG signal processing, the matrix $\mathbf{C_X}$ is transformed from the fixed-point number representation into the floating-point number representation to ensure data to approach the whitened data. In the fixed-to-floating converter, the fixed-point number L_{fixed} adopts an 18- or 24-bit integer format and the floating-point number L_{float} adopts the IEEE-754 single-precision format as shown in Fig. 3, where s_L, e_L, and f_L denote a sign bit, an 8-bit biased exponent and a 23-bit fraction part of a mantissa, respectively. The value of a normalized number L_{float} can be expressed as

$$L_{\text{float}} = (-1)^{s_L} \times (1.f_L) \times 2^{e_L - 127}. \quad (19)$$

The next step is to calculate the eigenvalue and eigenvector of $\mathbf{C_X}$ in (9). A special solution for obtaining eigenvalues and eigenvectors of a 2×2 matrix is used in previous literature [26]. This preprocessing method is not easily applicable to an $n \times n$ matrix for $n > 2$. In order to implement the high-dimensional EVD processor in the preprocessing part, the cyclic Jacobi method is adopted in this paper. The cyclic Jacobi method is known as a simpler algorithm to calculate the eigenvalue and eigenvector [28] and can be implemented in hardware [29]–[31]. The basic principle of the cyclic Jacobi method is described as follows. The cyclic Jacobi method applying a sequence of Jacobi rotations $\mathbf{J}$s to the right side and the left side of the symmetric matrix $\mathbf{B}$ is expressed below

$$\mathbf{B}' = \mathbf{J}_{pq}^T \mathbf{B} \mathbf{J}_{pq} \quad (20)$$

where a symmetric matrix $\mathbf{B}$ and a Jacobi rotation $\mathbf{J}$ are defined in (21) and (22), respectively

$$\mathbf{B} = \begin{bmatrix} & \cdots & b_{1p} & \cdots & b_{1q} & \cdots & \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ b_{p1} & \cdots & b_{pp} & \cdots & b_{pq} & \cdots & b_{pn} \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ b_{q1} & \cdots & b_{qp} & \cdots & b_{qq} & \cdots & b_{qn} \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ & \cdots & b_{np} & \cdots & b_{nq} & \cdots & \end{bmatrix} \quad (21)$$

$$\mathbf{J}(p, q, \theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & \sin\theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ p \\ \\ q \\ \\ \end{matrix} \quad (22)$$

where (p, q) denotes a position index pair whose range is $1 \le p < q \le n$ in the cyclic-by-row manner. This is also

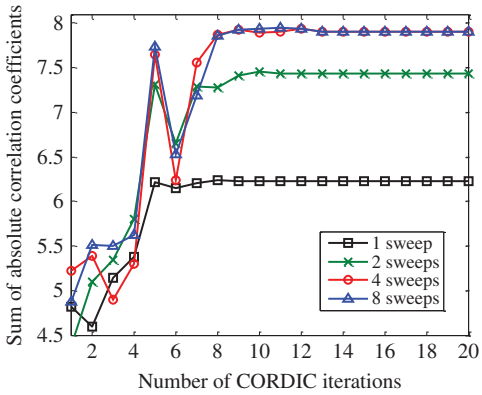

Fig. 5. Accuracy analysis of the CORDIC algorithm.

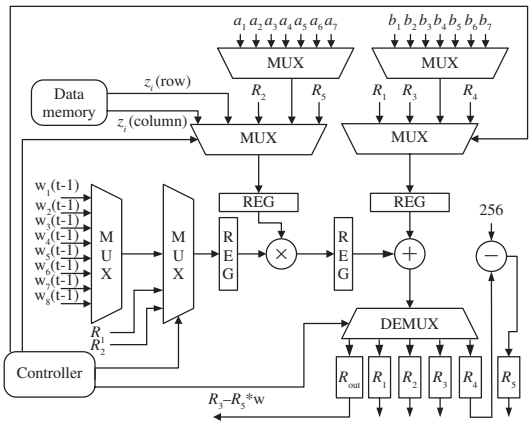

Fig. 6. Block diagram of the proposed hardware reused one-unit architecture.

called as a Jacobi sweep. The optimal rotation angle in the (p, q) plane is derived from the following equation:

$$\begin{bmatrix} b'_{pp} & 0 \\ 0 & b'_{qq} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{23}$$

Since $b_{pq} = b_{qp}$, the optimal rotation angle can be expressed as [30]

$$\theta_{opt} = \frac{1}{2}\tan^{-1}\frac{2b_{pq}}{b_{qq} - b_{pp}}. \tag{24}$$

In general, a symmetric matrix $\mathbf{B}$ can be transformed into a diagonal matrix $\mathbf{D}$ by a series of Jacobi rotations. That means the off-diagonal elements will approximate to zero by a series of Jacobi rotations. For $n = 8$, diagonal matrices $\mathbf{D}$ and $\mathbf{E}$ can be expressed in (25) and (26), respectively

$$\mathbf{D} = \mathbf{E}^T\mathbf{B}\mathbf{E} \tag{25}$$

$$\mathbf{E} = \mathbf{J}_{12}\mathbf{J}_{13}\ldots\mathbf{J}_{18}\mathbf{J}_{23}\ldots\mathbf{J}_{28}\mathbf{J}_{34}\ldots\mathbf{J}_{78} \tag{26}$$

where each Jacobi rotation matrix $\mathbf{J}$ is recorded by the matrix $\mathbf{E}$ in (26). The initial value of the matrix $\mathbf{B}$ is the covariance matrix of $\mathbf{C_X}$, and the initial value of the matrix $\mathbf{E}$ is the identity matrix $\mathbf{I}$.

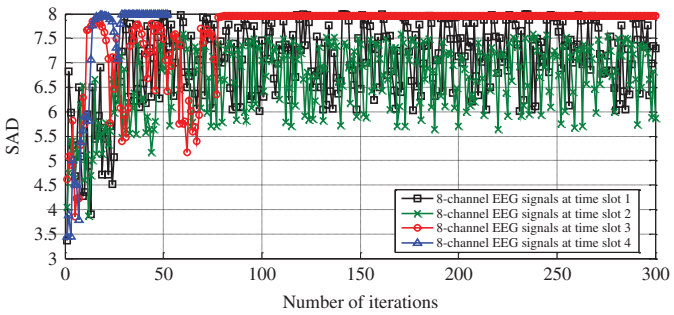

Fig. 7. Simulation analysis of SAD values.

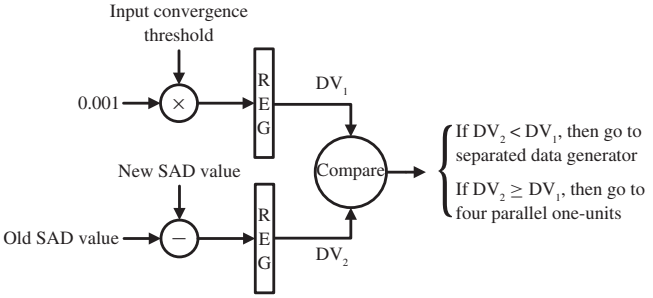

Fig. 8. Block diagram of the proposed early determination unit.

The CORDIC algorithm that performs iterative operations with shift-and-add behavior for vector rotations by arbitrary angles [29], [30] is used to implement the cyclic Jacobi method. The CORDIC algorithm is operated in two modes including the vectoring mode and the rotation mode. In this paper, two modes for vectoring and rotation are implemented in one CORDIC engine to reduce hardware cost. One CORDIC-engine-based EVD processor architecture as shown in Fig. 4 adopts the floating-point operation, where the controller is used to determine which mode can be executed. The CORDIC engine in the vectoring mode can obtain $z_r = z_0 + \tan^{-1}(y_0/x_0)$ for inputs x_0, y_0 and z_0 [31] with r CORDIC iterations. In order to obtain the optimal angle in the (p, q) plane, $b_{qq} - b_{pp}$ feeds to x_0, $2b_{pq}$ which is obtained by adding one for the exponential term feeds to y_0, and an initial value zero feeds to z_0 in Fig. 4. After r CORDIC iterations in the vectoring mode, the CORDIC engine outputs z_r and then subtracts one from the exponential term of z_r to obtain the optimal angle in (24) that stored at the register θ. Thus, (24) is realized by shift-and-add operation and arctan table without the dedicated division and arctan computation hardware. In other words, the vectoring mode is used to generate the optimal angle in (24). Next, the corresponding vectors are rotated in the rotation mode with the optimal angle and are saved at the register D. Register E is operated in a similar behavior. Registers D and E are used to store eigenvalues and eigenvectors, respectively. Since the covariance matrix is a symmetric matrix, some rotation results are the same in the plane, e.g., $\mathbf{J}^T[\,b_{p1}\ b_{q1}\,]^T\mathbf{J}$ equals $\mathbf{J}^T[\,b_{1p}\ b_{1q}\,]^T\mathbf{J}$. Thus, only two vectors $[\,b_{pp}\ b_{qp}\,]^T$ and $[\,b_{pq}\ b_{qq}\,]^T$ are needed to rotate in the second rotations [30]. We also need to take down the accumulative records of all orthonormal transforms to obtain matrix $\mathbf{E}$ by (26). After eight Jacobi sweeps, the eigenvalue

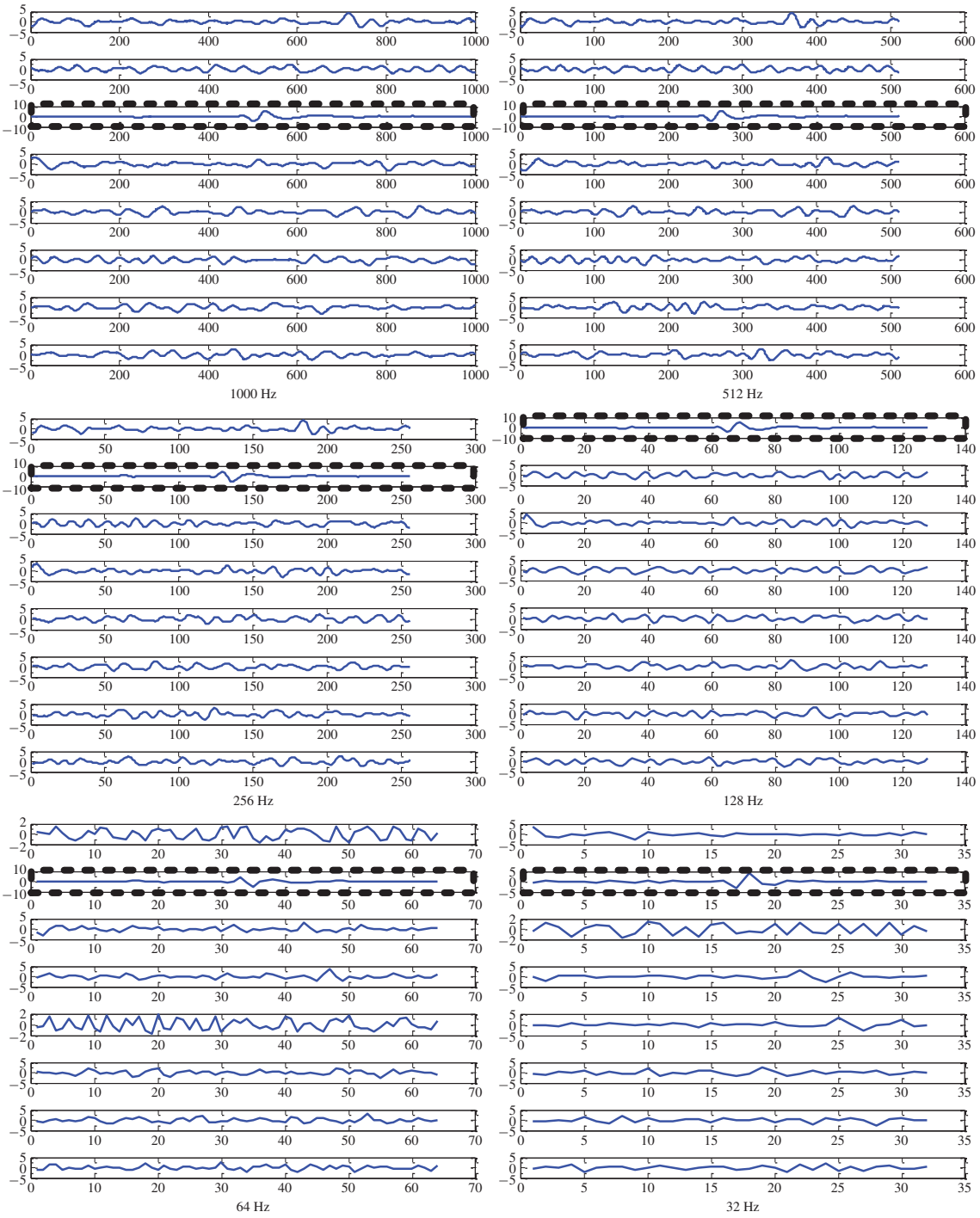

Fig. 9. Separation results of 12-bit 8-channel EEG signals with one second window size at 1000, 512, 256, 128, 64 and 32 Hz sample rates.

and eigenvector are obtained. The matrix $\mathbf{P}$ is obtained by multiplying $\mathbf{D}^{-1/2}$ and $\mathbf{E}^T$ in Block 3 of Fig. 2, where we apply an inverse square root structure [32] to implement $\mathbf{D}^{-1/2}$. In [32, Fig. 3], the inverse square-root structure applies an initial approximation and a modified Newton-Raphson iteration, where the former one uses a lookup table, operand modification and multiplication, and the latter one consists of one square unit, one multiplier-complement unit and one multiplier-add operation unit. Converting $\bar{\mathbf{X}}$ into the floating-point representation and multiplying the matrix $\mathbf{P}$ is to obtain the whitened signal in Block 4 of Fig. 2. Finally, the whitened data in the floating-point number representation are stored at the data memory after preprocessing.

The accuracy of eigenvalue and eigenvector of the CORDIC algorithm can be evaluated by the number of CORDIC iterations and Jacobi sweeps. In Fig. 5, when the numbers of CORDIC iterations and Jacobi sweeps are greater than 10 and 4, respectively, we find that the BSS quality evaluated

TABLE I
HIGH ACCURACY PIECEWISE LINEAR FUNCTION APPROXIMATION FOR HYPERBOLIC TANGENT

x	a	b	RMSE
$\|x\| \geq 7$	0	$1 \times \text{sign(x)}$	9.924×10^{-8}
$3 \leq \|x\| < 7$	0.0006965	$0.9959 \times \text{sign(x)}$	7.079×10^{-4}
$2 \leq \|x\| < 3$	0.02922	$0.9113 \times \text{sign(x)}$	2.113×10^{-3}
$1.5 \leq \|x\| < 2$	0.1162	$0.7358 \times \text{sign(x)}$	2.024×10^{-3}
$1 \leq \|x\| < 1.5$	0.2844	$0.4878 \times \text{sign(x)}$	4.45×10^{-3}
$0.5 \leq \|x\| < 1$	0.598	$0.1788 \times \text{sign(x)}$	6.959×10^{-3}
$0 \leq \|x\| < 0.5$	0.9533	0	5.661×10^{-3}

by the sum of absolute correlation coefficients can be more promising for mixed signals. That means more accuracy of the FastICA algorithm can be achieved. The absolute correlation coefficients can be obtained between the output values of the FastICA algorithm in different CORDIC iterations/Jacobi sweeps and output values of the deflation FastICA algorithm in MATLAB. In order to increase reliability, in our experiment, the numbers of CORDIC iterations and Jacobi sweeps are set to 18 and 8, respectively.

B. Implementation of the Low-Area One-Unit Processing Unit

Considering eight channels and 256 samples per channel, the one-unit processing equation can be recast in the following:

$$\begin{aligned}\mathbf{w}^+ &= E\left\{\mathbf{Z}[g(\mathbf{w}^T\mathbf{Z})]^T\right\} - E\left\{g'(\mathbf{w}^T\mathbf{Z})\right\}\mathbf{w} \\ &= E\left\{\mathbf{Z}[\tanh(\mathbf{w}^T\mathbf{Z})]^T\right\} - \sum_{i=1}^{256}\left\{\frac{1-\tanh^2(\mathbf{w}^T\mathbf{z}_i)}{256}\right\}\mathbf{w} \\ &= E\left\{\mathbf{Z}[\tanh(\mathbf{w}^T\mathbf{Z})]^T\right\} - \left\{\frac{256-\sum_{i=1}^{256}\tanh^2(\mathbf{w}^T\mathbf{z}_i)}{256}\right\}\mathbf{w} \end{aligned} \quad (27)$$

where $\mathbf{z}_i$ is the i-th column vector of the matrix $\mathbf{Z}$. A low-area one-unit architecture using the hardware reusing scheme is shown in Fig. 6, where the reused operations are addressed as follows.

Step 1: Calculate one element of the vector $\mathbf{w}^T\mathbf{Z}$ and save the element at the register R_1, where one element equals $\mathbf{w}^T\mathbf{z}_i$.

Step 2: Calculate one element of the vector $\tanh(\mathbf{w}^T\mathbf{Z})$ and save the element at the register R_2, where one element equals $\tanh(\mathbf{w}^T\mathbf{z}_i)$.

Step 3: Calculate one element of the vector $\mathbf{Z}[\tanh(\mathbf{w}^T\mathbf{Z})]^T$ and save the element at the register R_3

Step 4: Calculate the value of $\sum_{i=1}^{256}\tanh^2(\mathbf{w}^T\mathbf{z}_i)$ and save the value at the register R_4.

Step 5: Calculate the value of $256-\sum_{i=1}^{256}\tanh^2(\mathbf{w}^T\mathbf{z}_i)$ and save the value at the register R_5.

Step 6: Calculate one element of the vector $\mathbf{w}^+$ and save the element at the output register.

Step 7: Repeat Step 1–Step 6 until eight elements of the vector $\mathbf{w}^+$are obtained.

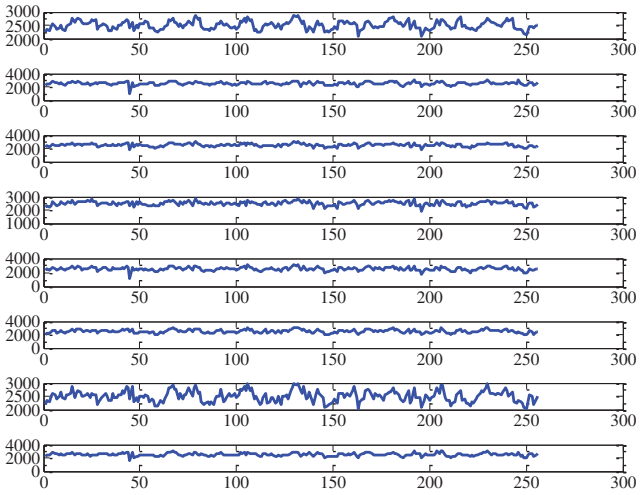

Fig. 10. Eight-channel mixed signals generated by MATLAB.

In the proposed hardware reuse architecture as shown in Fig. 6, the data memory has two access types for read operation. One is read in row scan and another is read in column scan for the matrix $\mathbf{Z}$. In Step 1, one element of the vector $\mathbf{w}^T\mathbf{Z}$ is calculated, where the whitened data $\mathbf{Z}$ are read in column scan from the data memory. Then, the element is saved to the register R_1. In Step 2, the value of the register R_1 is retrieved to calculate the hyperbolic tangent of that, and the result is saved to the register R_2. In [33], only nine-piecewise linear function approximation is adopted such that the accuracy is not enough to our work. In order to achieve high accuracy, the 13-piecewise linear function approximation is adopted to calculate the hyperbolic tangent. The hyperbolic tangent is expressed as follows:

$$\tanh x \approx ax + b \quad (28)$$

where x, a, and b denote an input variable and two parameters, respectively. The range of x and the values of a and b are listed in Table I, where the first six ranges represent 12-piecewise linear function approximations due to the symmetry. At the worst case, the root mean square error (RMSE) can be attained in the order of 10^{-3}. In Step 3, the value of the register R_2 is retrieved and the whitened data $\mathbf{Z}$ are read in row scan from the memory, after 256 iterative accumulations, one element of the vector $\mathbf{Z}[\tanh(\mathbf{w}^T\mathbf{Z})]^T$ is obtained and saved to the register R_3. In order to simplify the computation, we do not really compute $E\{g'(\mathbf{w}^T\mathbf{Z})\} = \{\sum_{i=1}^{256}[1-\tanh^2(\mathbf{w}^T\mathbf{z}_i)]/256\}$. In Step 4, corresponding to the register R_2, the value of $\tanh^2(\mathbf{w}^T\mathbf{z}_i)$ is accumulated by 256 iterations and saved to the register R_4. Thus, $\sum_{i=1}^{256}\tanh^2(\mathbf{w}^T\mathbf{z}_i)$ is obtained. In Step 5, the result of 256 minus the value of the register R_4 is saved to the register R_5. In Step 6, the resulting value of one element of $\mathbf{Z}[\tanh(\mathbf{w}^T\mathbf{Z})]^T$ at the register R_3 minus the element of $[256-\sum_{i=1}^{256}\tanh^2(\mathbf{w}^T\mathbf{z}_i)]\mathbf{w}$ is stored at the output register. Division by 256 is not necessary in Step 6 because the expectation will not affect the normalization value. Repeat Step 1∼Step 6 until eight elements of the vector $\mathbf{w}^+$ are obtained. Thus, the one-unit processing is finished.

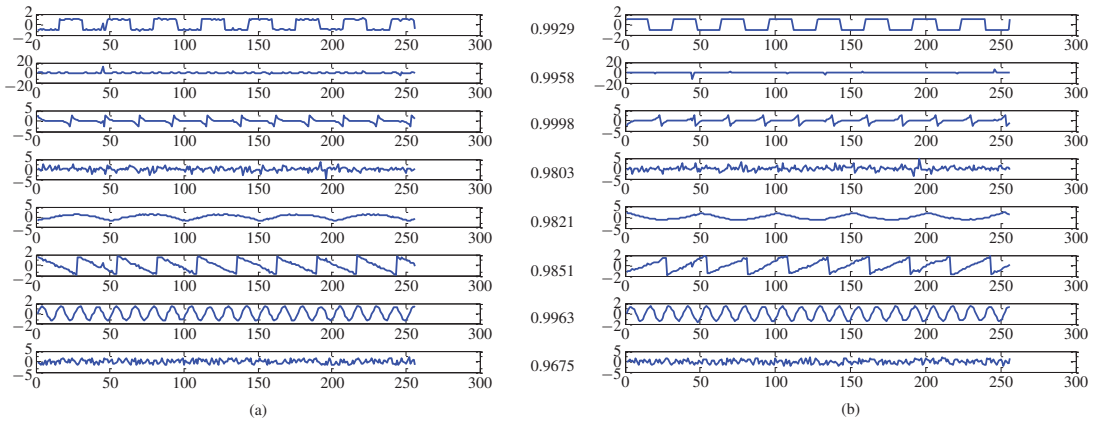

Fig. 11. Comparison results of (a) deflation FastICA and (b) FastICA using four parallel one-units by MATLAB simulation for mixed signals.

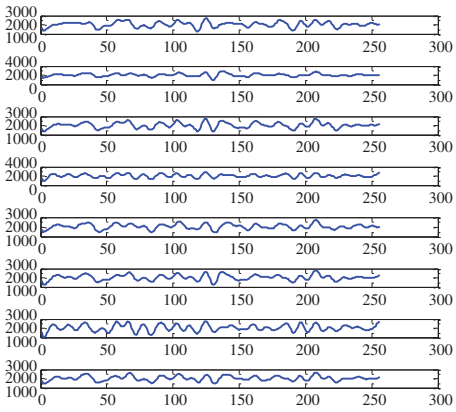

Fig. 12. Eight-channel EEG signals.

TABLE II
CHIP SPECIFICAITON SUMMARY OF THE PROPOSED FASTICA

Process Technology		UMC 90 nm 1P9M CMOS process
Power Supply		1.0 V
Max. Operating Frequency		100 MHz
Memory	Data Memory	2048 × 32 single-port SRAM
	OWMM	64 × 32 single-port SRAM
	NWMM	64 × 32 dual-port SRAM
CoreSize		1.221 × 1.218 mm^2
Gate Count		272 K*
Core Power Consumption		16.35 mW@1.0 V
No. of Channels		8
Max Computation Time		0.29 s

* The number is counted by the size of two-input NAND gate.

C. Implementation of the Early Determination Unit and Analysis of the Data Memory Size

From the simulation analysis as shown in Fig. 7, four time-slots EEG signals denote four eight-channel EEG signals that captured at different times. Among four time slots, before achieving the convergence or maximum iteration number, it can be observed that the SAD values are nearly the same in several adjacent iterations. That means the direction of the new weight vector is close to that of the old vector. Thus, we can set up one constraint to early terminate the iteration to save computation. In our experiment, the maximum iteration number is set to 300. Similarly, for some converged signals such as EEG signals at time slot 4 in Fig. 7, the SAD values have this consumption behavior before attaining the convergence. Thus, the early determination scheme and the corresponding low-cost architecture in Fig. 8 for energy saving are motivated to develop. The early determination unit is enabled when the process does not converge after the convergence checking unit. In the early determination unit, when the difference value 2 (DV_2) between an old SAD value and a new SAD value is less than the difference value 1 (DV_1) that is obtained by 0.001 × (input convergence threshold), the control flow is determined to go to the separated data generator unit to obtain the separated results. Otherwise (i.e., DV_2 is larger than or equal to DV_1), the control flow goes back to the four parallel one-units to train $\overline{\mathbf{W}}$ again. In Fig. 8, since the early determination unit only consists of one multiplier, one subtractor, and one comparator, the unit is cost effective with the slightly hardware overhead. Thus, the proposed early determination unit not only significantly saves energy as shown in Section IV but also shows cost-effective.

The energy-efficient FastICA architecture is designed for eight-channel BSS. Due to low area cost, low energy and satisfactory BSS quality requirement, the data memory size of the input data length will be constrained. Fig. 9 shows the separation results of 12-bit 8-channel EEG signals with one second window size at sample rates of 1000, 512, 256, 128, 64, and 32 Hz using four parallel one-units FastICA. From Fig. 9, the left finger movement component in dotted rectangular box can be observed at each sample rate. In order to increase reliability, in our experiment, the sample rate of EEG signals is set to 256 Hz. Thus, the data memory size with $256 \times 8 \times 32 = 65\,536$ bits can achieve low area cost and satisfactory BSS results in the proposed architecture. Note that the output data in the 32-bit floating-point format is also stored in the data memory. The other detailed environment setting will be addressed in Section IV-A.

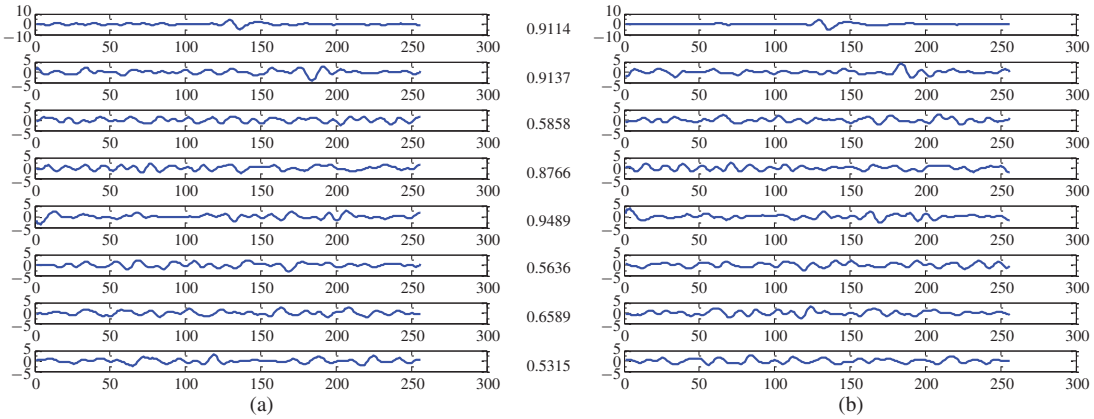

Fig. 13. Comparison results of (a) deflation FastICA and (b) FastICA using four parallel one-units by MATLAB simulation for eight-channel EEG signals.

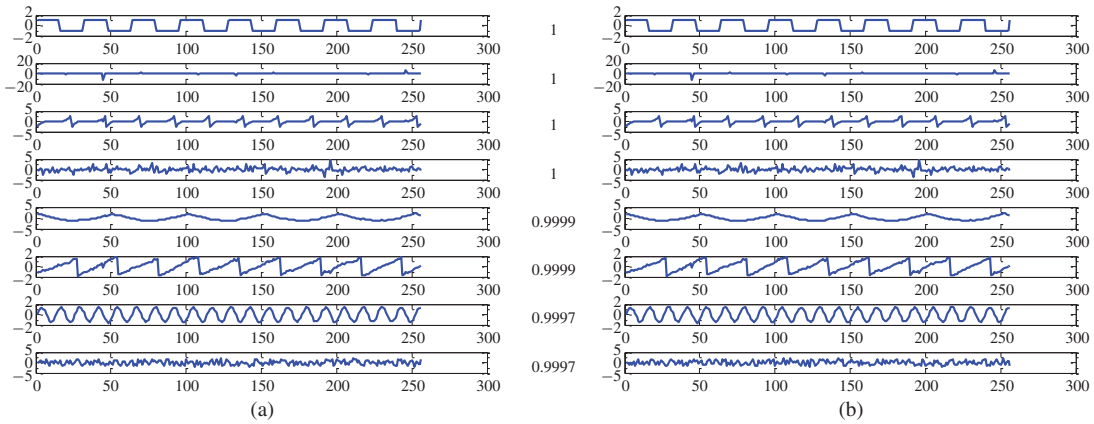

Fig. 14. Comparison results of FastICA using four parallel one-units by (a) MATLAB and (b) post layout simulation without the early determination scheme for mixed signals.

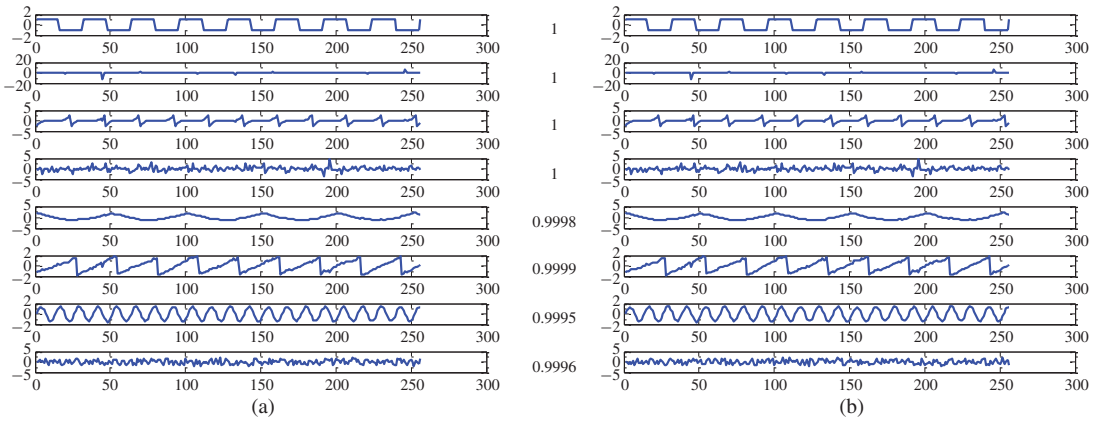

Fig. 15. Comparison results of FastICA using four parallel one-units by (a) MATLAB and (b) post layout simulation with the early determination scheme for mixed signals.

In summary, the EVD processor with floating-point operation, sufficient number of CORDIC iterations and Jacobi sweeps, 13-piecewise linear function approximation, and suitable data memory size can support high accurate FastICA such that the satisfactory BSS quality can be attained. Most importantly, the energy-efficient, cost-effective, low-computation-time FastICA implementation is achieved by the following architectures: 1) the early determination scheme and the corresponding architecture; 2) the proposed preprocessing unit architecture with one CORDIC-based EVD processor and the proposed one-unit architecture with the hardware reuse scheme; and 3) the four parallel one-units architecture.

IV. Implementation and Comparison Results

In this section, the software simulation is illustrated to verify the algorithm and the post-layout simulation is explained to verify the proposed architecture and schemes.

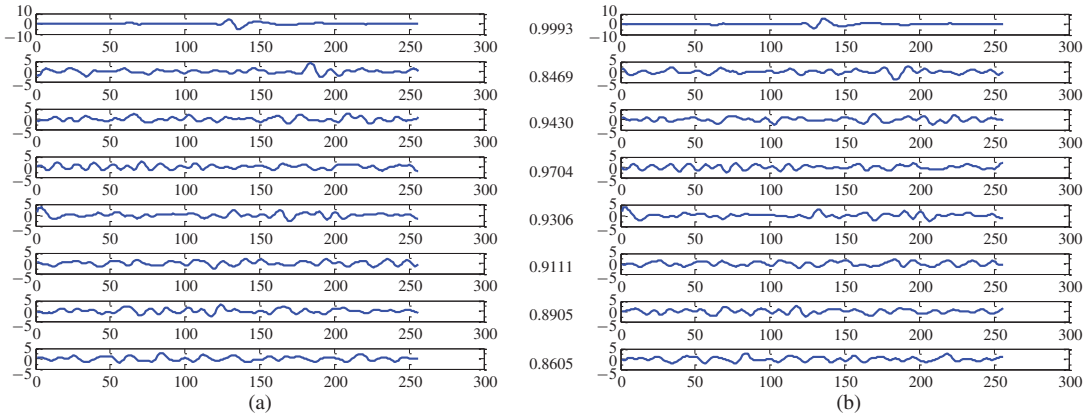

Fig. 16. Comparison results of FastICA using four parallel one-units by (a) MATLAB and (b) post layout simulation without the early determination scheme for eight-channel EEG signals.

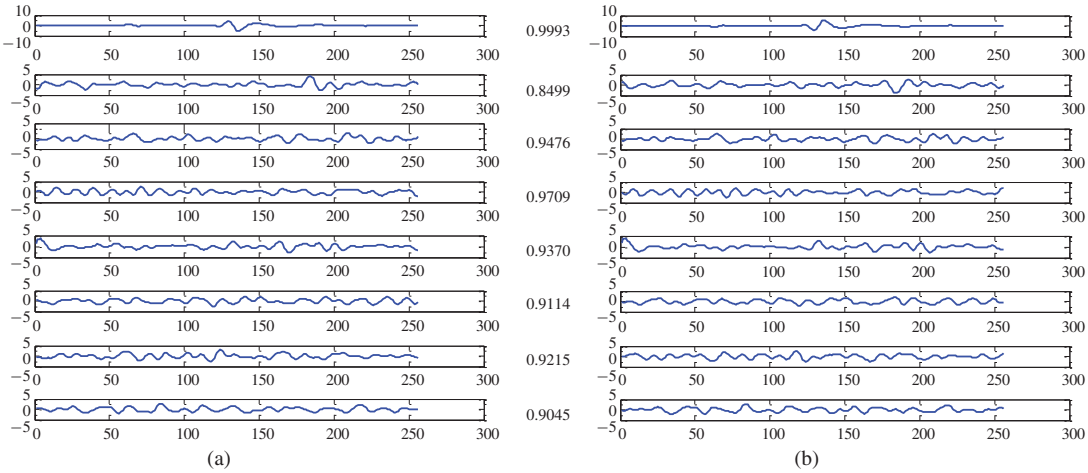

Fig. 17. Comparison results of FastICA using four parallel one-units by (a) MATLAB and (b) post layout simulation with the early determination scheme for eight-channel EEG signals.

The comprehensive evaluation and comparison results are addressed in this section.

A. Software Simulation and Post-Layout Simulation

In order to ensure the BSS quality of the proposed FastICA implementation using four parallel one-units, we compare absolute correlation coefficients with the deflation FastICA [34]. The inputs of eight-channel mixed signal patterns with 12-bit precision are generated by seven non-Gaussian distributions plus one Gaussian distribution in MATLAB as shown in Fig. 10. Fig. 11(a) and (b) shows the output waveforms via deflation FastICA and FastICA using four parallel one-unit operations by MATLAB, respectively. The numbers in the middle of Fig. 11 denote the absolute correlation coefficients of the separated signals. Each absolute correlation coefficient is obtained by calculating the absolute correlation between deflation FastICA simulation values on the left-hand side and FastICA simulation values using four parallel one-units on the right-hand side of Fig. 11. Herein, the real eight-channel EEG signals with the 12-bit precision as shown in Fig. 12 that captured from FZ, FC3, FCZ, FC4, C3, CZ, C4, and CPZ of the international 10–20 electrodes placement system with 256 Hz sample rate and one second window size are applied in this experiment. Since the input EEG signals are captured in a planned manner, a left finger movement component, to our best knowledge, appears in the middle of the signals in Fig. 12 and is tested in this experiment. The MATLAB output simulation waveforms of deflation FastICA and FastICA using four parallel one-units are shown in Fig. 13(a) and (b), respectively. The smallest and average absolute correlations of the left finger movement component are 0.9114 in Fig. 13 and 0.9874 among 50 separation test cases/events, respectively. On the other hand, since we do not know whether there exist other components in other separated channels, we do not comment on other absolute correlation coefficients of other channels in Fig. 13. From Figs. 11 and 13, the values of the absolute correlation coefficient of mixed signals and one EEG component are 0.9675 and 0.9114, respectively. In other word, the FastICA algorithm using four parallel one-units can approach the BSS quality of the deflation FastICA algorithm.

In the proposed FastICA architecture, a 12-bit integer input and 32-bit floating-point output are chosen to ensure the performance accuracy. Fig. 14(a) and (b) shows the output waveforms after the FastICA operation by MATLAB and

TABLE III
COMPARISON RESULTS AMONG VARIOUS ICA IMPLEMENTATIONS

	[22]	[23]	[24]	[26]	[27]	This paper
Application	Speech	Image	Image	Speech	EEG	EEG
Algorithm	ICA	pICA	pICA	FastICA	INFOMAX	FastICA
No. of Channels/Weight Vectors (WVs)	2	20 (WVs)	4 (WVs)	2	4	8
Speed (MHz)	12.288, N/A	21.829*1 21.357*2 35.921*3	20.161, N/A	50	68	100
Power Dissipation (mW)	98.8, 14.5	N/A	N/A	N/A	N/A	16.35
High-Dimensional Preprocessing ($n > 2$)	No	No	No	No	No	Yes
Gates (million)	0.0114, N/A (for ANC)	N/A	0.2295, N/A	N/A	0.315	0.272
Computation Time (s)	≥60, N/A	1129.5	N/A	0.003	N/A	0.29 (Max)
Implementation Approach	FPGA, ASIC	FPGA	FPGA, ASIC	FPGA	FPGA	ASIC

*1: For Sub-matrix. *2: For External Decorrelation. *3: For Comparison.

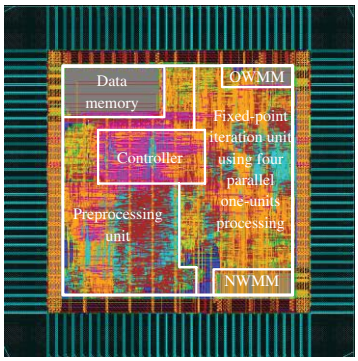

Fig. 18. Proposed eight-channel FastICA layout.

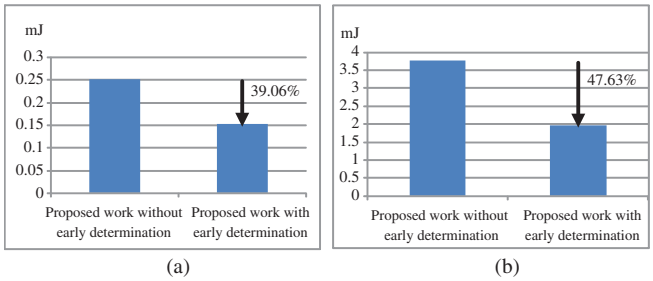

Fig. 19. Energy evaluation results with/without the early determination scheme for mixed signals and EEG signals. (a) Mixed signals. (b) EEG signals.

post-layout simulation, respectively, where the absolute correlation coefficients are also shown in the middle. Considering the early determination scheme, the output waveforms and absolute correlation coefficients are shown in Fig. 15. As we can see, the mixed signals are well separated for each channel and the corresponding absolute correlation coefficients are 1, 1, 1, 1, 0.9998, 0.9999, 0.9995, and 0.9996, respectively. The simulation result indicates the validity of the FastICA implementation using four parallel one-units with the early determination scheme. Considering the real eight-channel EEG signals as shown in Fig. 12, the MATLAB and post-layout simulation waveforms, and the corresponding absolute correlation coefficients are shown in Fig. 16. With the early determination scheme, the output waveforms and absolute correlation coefficients are shown in Fig. 17. In summary, the simulation results as shown in Figs. 14–17 show that the proposed architecture can approach the behavior of the FastICA algorithm. For mixed signals in Fig. 10, the absolute correlation coefficient is 0.9995 at least with the early determination scheme. For eight-channel EEG signals in Fig. 12, the absolute correlation coefficient for the EEG component is 0.9993 with the early determination scheme.

B. Implementation and Comparison Results

Concerning the chip implementation, the cell-based design flow with the standard cell library in UMC 90 nm CMOS process is adopted. A chip layout of the proposed FastICA is shown in Fig. 18. The chip specifications of the proposed architecture are listed in Table II.

In Fig. 19, compared with the design without early determination, the energy reduction can be achieved by 39.06% and 47.63% for the cases of the mixed signals and EEG signals, respectively. Next, the comprehensive comparison results in terms of the application, algorithm, number of channels/weight vectors, speed, power dissipation, gate count, computation time and implementation approach among the existing ICA architectures are listed in Table III. In terms of application and algorithm, the architectures in [22] and [26] are applied to speech processing. The former one [22] applies the ICA algorithm and the latter one [26] uses the FastICA algorithm. Both architectures implemented by FPGA are capable of

processing two-channel speech signals and operating at 12.288 and 50 MHz, respectively. The power dissipation of the design in [22] is 98.8 mW in FPGA approach. The architecture [23] based on pICA has speeds of 21.829, 21.357, and 35.921 MHz for submatrix group, external decorrelation group and comparison group, respectively, in FPGA and can be applied to hyperspectral image analysis. The architecture [27] based on INFOMAX achieves the speed of 68 MHz in FPGA and is capable of processing four-channel EEG signals. Here, in Table III, the high-dimensional preprocessing means an eight-channel preprocessing unit is provided. In [26], 2-D closed-form solution is adopted to calculate eigenvalues in the preprocessing part; however, this closed form solution may not be easily applied to the high-dimensional preprocessing system. The proposed FastICA architecture is applicable to the high-dimensional preprocessing system due to the CORDIC-based EVD processor. Considering the gate count per channel/weight vector, the proposed FastICA implementation has lower gate count compared with the works in [24] and [27]. In terms of the computation time, the proposed FastICA has less computation time than the works in [22] and [23]. The maximum computation time in this paper is obtained when the maximum iteration number is reached. When the chip operates at 100 MHz, the maximum computation time is about 0.29 s which is less than one second window size. Thus, the resulting chip implementation of the proposed architecture has enough speed for real-time processing. From the simulation, implementation and comparison results, to our best knowledge, we should be the first one to present the energy-efficient, low-cost and low-computation-time eight-channel FastICA architecture and implementation with satisfactory BSS quality for EEG signal separation via a digital ASIC approach.

V. Conclusion

In this paper, we have presented the energy-efficient, low-cost and low-computation-time FastICA architecture and chip layout implementation with satisfactory BSS quality for eight-channel EEG separation. The applied schemes include the early determination scheme for energy saving, the CORDIC reuse scheme for the high-dimensional and low-area EVD processor, hardware reuse scheme for the low-area one-unit architecture, four parallel one-units for low computation time, and the high-accuracy piecewise linear function approximation for hyperbolic tangent computation in one-unit operation. With such performance, the proposed design has the potential to be embedded in the portable biomedical systems.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable suggestions that improved this paper. Also, they thank Prof. C.-T. Lin and Prof. Y.-S. Chen at National Chiao Tung University, Hsinchu, Taiwan, for providing suggestions and EEG data, and P.-Y. Huang for his assistance. They also thank the Chip Implementation Center, Hsinchu, Taiwan, for technical support.

References

[1] T. W. Lee, *Independent Component Analysis: Theory and Applications*. Boston, MA: Kluwer, 1998.

[2] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.

[3] A. Cichocki and S. Amari, *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. New York: Wiley, 2002.

[4] R. N. Vigário, "Extraction of ocular artifacts from EEG using independent component analysis," *Electr. Clin. Neurophys.*, vol. 103, no. 3, pp. 395–404, Sep. 1997.

[5] R. Vigário, J. Särelä, V. Jousmäki, M. Hämäläinen, and E. Oja, "Independent component approach to the analysis of EEG and MEG recordings," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 5, pp. 589–593, May 2000.

[6] A. Kachenoura, L. Albera, L. Senhadji, and P. Comon, "ICA: A potential tool for BCI systems," *IEEE Signal Process. Mag.*, vol. 25, no. 1, pp. 57–68, Jan. 2008.

[7] A. J. Bell and T. J. Sejnowski, "An information maximization approach to blind separation and blind deconvolution," *Neural Comput.*, vol. 7, no. 6, pp. 1129–1159, Nov. 1995.

[8] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Comput.*, vol. 9, no. 7, pp. 1483–1492, Oct. 1997.

[9] A. Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 626–634, May 1999.

[10] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Netw.*, vol. 13, nos. 4–5, pp. 411–430, Jun. 2000.

[11] E. Oja and Z. Yuan, "The fastICA algorithm revisited: Convergence-analysis," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1370–1381, Nov. 2006.

[12] E. Ollila, "The deflation-based fastICA estimator: Statistical analysis revisited," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1527–1541, Mar. 2010.

[13] S. Choi, A. Cichocki, and S. Amari, "Flexible independent component analysis," *J. VLSI Signal Process.*, vol. 26, nos. 1–2, pp. 25–38, 2000.

[14] V. Zarzoso, P. Comon, and R. Phlypo, "A contrast function for independent component analysis without permutation ambiguity," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 863–868, May 2010.

[15] V. Zarzoso and P. Comon, "Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 248–261, Feb. 2010.

[16] Y. Washizawa, Y. Yamashita, T. Tanaka, and A. Cichocki, "Blind extraction of global signal from multi-channel noisy observations," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1472–1481, Sep. 2010.

[17] G. Zhou, Z. Yang, S. Xie, and J. M. Yang, "Online blind source separation using incremental nonnegative matrix factorization with volume constraint," *IEEE Trans. Neural Netw.*, vol. 22, no. 4, pp. 550–560, Apr. 2011.

[18] H. Du, H. Qi, and X. Wang, "Comparative study of VLSI solutions to independent component analysis," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 548–558, Feb. 2007.

[19] K. S. Cho and S. Y. Lee, "Implementation of InfoMax ICA algorithm with analog CMOS circuits," in *Proc. Int. Workshop Independent Compon. Anal. Blind Signal Separat.*, Vancouver, BC, Canada, Dec. 2001, pp. 70–73.

[20] M. H. Cohen and A. G. Andreou, "Analog CMOS integration and experimentation with an autoadaptive independent component analyzer," *IEEE Trans. Circuits Syst. II-Anal. Digital Signal Process.*, vol. 42, no. 2, pp. 65–77, Feb. 1995.

[21] A. Celik, M. Stanacevic, and G. Cauwenberghs, "Mixed-signal real-time adaptive blind source separation," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, pp. 760–763.

[22] C. M. Kim, H. M. Park, T. Kim, Y. K. Choi, and S. Y. Lee, "FPGA implementation of ICA algorithm for blind signal separation and adaptive noise canceling," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1038–1046, Sep. 2003.

[23] H. Du and H. Qi, "A reconfigurable FPGA system for parallel independent component analysis," *EURASIP J. Embedded Syst.*, vol. 2006, no. 23025, pp. 1–12, 2006.

[24] H. Du, H. Qi, and G. D. Peterson, "Parallel ICA and its hardware implementation in hyperspectral image analysis," *Proc. SPIE*, vol. 5439, pp. 74–83, Apr. 2004.

[25] 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. May 2005, pp. 5822–5825.
[26] K. K. Shyu, M. H. Lee, Y. T. Wu, and P. L. Lee, "Implementation of pipelined fastICA on FPGA for real-time blind source separation," *IEEE Trans. Neural Netw.*, vol. 19, no. 6, pp. 958–970, Jun. 2008.
[27] W. C. Huang, S. H. Hung, J. F. Chung, M. H. Chang, L. D. Van, and C. T. Lin, "FPGA implementation of 4-channel ICA for on-line EEG signal separation," in *Proc. IEEE BioCAS*, Nov. 2008, pp. 65–68.
[28] G. H. Golub and C. F. Van Loan, *Matrix Computation*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.
[29] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. 8, no. 3, pp. 330–334, Sep. 1959.
[30] M. Kim, K. Ichige, and H. Arai, "Design of Jacobi EVD processor based on CORDIC for DOA estimation with MUSIC algorithm," in *Proc. PIMRC Conf.*, vol. 1. Sep. 2002, pp. 120–124.
[31] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. ACM/SIGDA Conf.*, North Kingston, RI, Feb. 1998, pp. 192–200.
[32] M. J. Schulte and K. E. Wires, "High-speed inverse square roots," in *Proc. 14th Int. Symp. Comput. Arithmetic*, Apr. 1999, pp. 124–131.
[33] S. Papaharalabos, P. Sweeney, B. G. Evans, G. Albertazzi, A. Vanelli-Coralli, and G. E. Corazza, "Performance evaluation of a modified sum-product decoding algorithm for LDPC codes," in *Proc. Int. Symp. Wireless Commun. Syst.*, Sep. 2005, pp. 800–804.
[34] *The FastICA Package for MATLAB* [Online]. Available: http://www.cis.hut.fi/projects/ica/fastica/

Lan-Da Van (S'98–M'02) received the B.S. (honors) and M.S. degrees from the Tatung Institute of Technology, Taipei, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree from National Taiwan University (NTU), Taipei, in 2001, all in electrical engineering.

He was an Associate Researcher with National Chip Implementation Center, Hsinchu, Taiwan, from 2001 to 2006. He joined the Faculty with the Department of Computer Science, National Chiao Tung University, Hsinchu, where he has been an Assistant Professor since February 2006. He has published more than 45 journal and conference papers and holds one U.S. and one Taiwan patent in these areas. His current research interests include very large scale integration algorithms, architectures, chips for digital signal processing systems, design of low-power/high-performance/cost-effective 3-D graphics systems, computer arithmetic, adaptive filters, and transform designs.

Dr. Van was the recipient of the Chunghwa Picture Tube and Motorola Fellowships in 1996 and 1997, respectively. He was an elected Chairman of the IEEE NTU Student Branch in 2000. In 2002, he has received the IEEE Award for Outstanding Leadership and Service to the IEEE NTU Student Branch. From 2009 to 2010, he served as an Officer of the IEEE Taipei Section. He served as a reviewer for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE SIGNAL PROCESSING LETTERS.

Di-You Wu received the B.S. degree in mathematics from National Cheng Kung University, Tainan, Taiwan, in 2006, and the M.S. degree from the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, in 2008. He is currently pursuing the Ph.D. degree with National Chiao Tung University.

His current research interests include very large scale integration digital signal processing and baseband communication systems.

Chien-Shiun Chen received the B.S. degree from Yuan Ze University, Taoyuan, Taiwan, in 2008, and the M.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 2010, both in computer science.

His current research interests include very large scale integration information processing algorithms and architecture for bio-signal processing.