VLSI Design for Convolutive Blind Source Separation

Jia-Ching Wang, *Senior Member, IEEE*, Chien-Yao Wang, Tzu-Chiang Tai, Min Shih, Shao-Chieh Huang, Ying-Chuan Chen, Yan-Yu Lin, and Li-Xun Lian

***Abstract*—This brief presents an efficient very-large-scale integration architecture design for convolutive blind source separation (CBSS). The CBSS separation network derived from the information maximization (Infomax) approach is adopted. The proposed CBSS chip design consists mainly of Infomax filtering modules and scaling factor computation modules. In an Infomax filtering module, input samples are filtered by an Infomax filter with the weights updated by Infomax-driven stochastic learning rules. As for the scaling factor computation module, all operations including logistic sigmoid are integrated and implemented by the circuit design based on a piecewise-linear approximation scheme. The proposed prototype chip is implemented via a semi-custom design using 90-nm CMOS technology on a die size of approximately 0.54×0.54 mm^2.**

***Index Terms*—Blind source separation (BSS), convolutive BSS (CBSS), convolutive mixing, information maximization (Infomax), very-large-scale integration (VLSI).**

I. Introduction

SEPARATION of mixed sources has received extensive attention in recent years. Blind source separation (BSS) attempts to separate sources from mixed signals when most of the information for sources and mixing process is unknown. Such restrictions make BSS a challenging task for researchers. BSS has become a very important research topic in a lot of fields. Notable examples include audio signal processing, biomedical signal processing, communication systems, and image processing [1]–[3]. Without a filtering effect, instantaneous mixing is considered a simple version of the mixing process of the source signals. However, for audio sources passing through an environmental filtering before arriving at the microphones, a convolutive mixing process occurs, and convolutive BSS (CBSS) [4] is used to recover the original audio sources.

Independent component analysis (ICA) is the conventional means of solving the BSS or CBSS problem [6]. However, this method is often highly computationally intensive and introduces time-consuming processes for software implementation. More than a faster solution than software implementation, hardware solution achieves optimal parallelism [15]. Providing hardware solutions for ICA-based BSS has drawn considerable attention recently. Cohen and Andreou [7] explored the feasibility of combining above-and-subthreshold CMOS circuit techniques for implementing an analog BSS chip that integrates an analog I/O interface, weight coefficients, and adaptation blocks. This chip incorporates the use of the Herault–Jutten ICA algorithm [17]. Cho and Lee [8] implemented a fully analog CMOS chip based on information maximization (Infomax) ICA, as developed by Bell and Sejnowski [5], [19]. The chip incorporated a modular architecture to extend its use as a multichip.

Apart from these analog BSS chips, various field-programmable gate array (FPGA) implementations with digital architectures have been developed. Li and Lin [9] realized the Infomax BSS algorithm based on system-level FPGA design, by using Quartus II, DSP builder, and Simulink. Du and Qi [10] presented an FPGA implementation for the parallel ICA (pICA) algorithm, which focuses on reducing dimensionality in hyperspectral image analysis. The pICA algorithm consists of three temporally independent functional modules that are synthesized individually with some reconfigurable components developed for reuse. Based on Infomax BSS, Ounas *et al.* [11] introduced a low-cost digital architecture implemented on FPGA. This design used merely one neuron to support sequential operations of the neurons in neural network. In 2008, Shyu *et al.* [12] designed a pipelined architecture for FPGA implementation based on FastICA for separating mixtures of biomedical signals, including electroencephalogram (EEG), magnetoencephalography (MEG), and electrocardiogram (ECG). In this design, floating-point arithmetic units were used to increase the precision of the numbers and ensure the FastICA performance.

Although FPGA has a short development time and inexpensive verification of algorithms in hardware, its hardware architecture design is not optimized in comparison with application specific integrated circuit (ASIC) fabricated in chips. Acharyya *et al.* [13] designed an ASIC chip with 0.13-μm standard cell CMOS technology for 2-D Kurtotic FastICA. This design is characterized by reduced and optimized arithmetic units through means of removing dividers in eigenvector computation and whitening. For portable EEG signal processing applications, Chen *et al.* [14] developed a low-power very-large-scale integration (VLSI) chip fabricated using the UMC 90-nm CMOS process. This chip can perform four-channel ICA to separate EEG and mixed EEG-like super-Gaussian signals in real time. However, the aforementioned ASIC chips focus on instantaneous mixing BSS. In this brief, we present a digital ASIC chip for CBSS, in which the source signals are convolutively mixed. The convolutive mixtures are separated using the CBSS separation network [16] extended from Infomax theory.

The CBSS problem was solved in the time domain mainly because in the frequency domain, the permutation and scaling

Manuscript received February 1, 2015; revised July 5, 2015; accepted August 9, 2015. Date of publication August 17, 2015; date of current version January 28, 2016. This brief was recommended by Associate Editor C. Shing-chow.

J.-C. Wang, C.-Y. Wang, M. Shih, S.-C. Huang, Y.-C. Chen, Y.-Y. Lin, and L.-X. Lian are with the Department of Computer Science and Information Engineering, National Central University, Jhongli 35320, Taiwan (e-mail: jcw@csie.ncu.edu.tw).

T.-C. Tai is with the Department of Computer Science and Information Engineering, Providence University, Taichung City, 43301, Taiwan.

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

Digital Object Identifier 10.1109/TCSII.2015.2468928

ambiguity among the frequency bins must be resolved [20]. Tackling the permutation and scaling ambiguity requires a number of irregular operations that complicate the VLSI design of a CBSS chip. The approach used herein does not have this shortcoming. Furthermore, this approach allows us to propose a modular VLSI architecture. The proposed CBSS ASIC chip is characterized by its modular design, high speed, and low power. To the best of our knowledge, the proposed ASIC chip is the first that can execute the CBSS algorithm.

II. BSS Using Infomax

A. CBSS

Assume there are N source signals recorded by M sensors. This brief focuses on convolutive mixing. The related model is mathematically expressed by

$$x_m(t) = \sum_{n=1}^{N} \sum_{k=0}^{L-1} h_{mn}(k) s_n(t-k) \tag{1}$$

where x_m, $m = 1, 2, \ldots, M$, is a mixed signal corresponding to sensor m; s_n, $n = 1, 2, \ldots, N$, is the nth source signal; h_{mn} is the unknown impulse response from source n to sensor m; t is the discrete time index; and L is the number of taps in convolution.

CBSS, a demixing or separation process, finds separated signals that approximate the original sources. The separation process can be expressed as

$$u_n(t) = \sum_{m=1}^{M} \sum_{l=0}^{L-1} w_{nm}^k x_m(t-k) \tag{2}$$

where u_n is the nth separated signal; w_{nm} is the L-tap separation filter for sensor m to separated signal n; and w_{nm}^k denotes the kth tap weight of w_{nm}.

The separation process (2) can be expressed in matrix form as

$$\boldsymbol{u}(t) = \sum_{k=0}^{L-1} \mathbf{W}^k \boldsymbol{x}(t-k) \tag{3}$$

where $\mathbf{W}^k$ is an $N \times M$ matrix with w_{nm}^k as its components; $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_M(t))^T$; and $\boldsymbol{u}(t) = (u_1(t), u_2(t), \ldots, u_N(t))^T$.

According to the separation model described by (2) or (3), seeking the separation filters is of priority concern in CBSS. To tackle this difficult problem, this brief adopts the Infomax approach [5], [19].

B. Infomax Approach for Convolutive Mixing BSS

The BSS problem assumes that statistical independence among source signals exists. Let s_n denote the nth source signal. The joint probability density function of all the sources can be written as

$$p(\boldsymbol{s}) = p(s_1, s_2, \ldots, s_N) = \prod_{n=1}^{N} p(s_n) \tag{4}$$

where $p(s_n)$ is the probability density function of s_n.

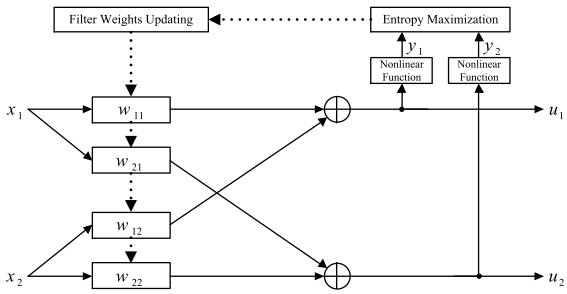

Fig. 1. Infomax-based CBSS separation network for the two-source and two-sensor case. This network contains four causal FIR filters w_{ij}^k, and u_i is the separated signal.

Accordingly, the statistically independent sources do not carry any mutual information I, which is defined in

$$I(s_1, s_2, \ldots, s_N) = \sum_{i=1}^{N} H(s_i) - H(\boldsymbol{s}) \tag{5}$$

where H denotes the differential entropy. While BSS attempts to generate separated signals close to source signals, the separation process focuses on generating output signals with zero mutual information.

To minimize the mutual information, Bell and Sejnowski developed the Infomax approach [5] to learn the separating process. This approach maximizes the joint entropy of the outputs by a stochastic gradient ascent algorithm. As plain maximization of the joint entropy of the outputs may diverge to infinity [18], the Infomax approach maximizes the joint entropy of $\mathbf{y} = g(\mathbf{u})$, where $g(.)$ refers to a nonlinear and monotonically transfer function. In convolutive mixing, the separation process is driven by $\mathbf{W}^k$, which is a matrix comprising filter components. Consider two sources and two sensors, in which Fig. 1 depicts the Infomax-based CBSS separation network [16], the separated signals $u_i(t)$ are obtained by the following network computation:

$$\begin{aligned} u_1(t) &= u_{11}(t) + u_{12}(t) \\ &= \sum_{k=0}^{L_{11}} w_{11}^k(t) x_1(t-k) + \sum_{k=0}^{L_{12}} w_{12}^k(t) x_2(t-k) \end{aligned} \tag{6}$$

$$\begin{aligned} u_2(t) &= u_{21}(t) + u_{22}(t) \\ &= \sum_{k=0}^{L_{22}} w_{22}^k(t) x_2(t-k) + \sum_{k=0}^{L_{21}} w_{21}^k(t) x_1(t-k) \end{aligned} \tag{7}$$

where w_{ij}^k are causal finite-impulse response (FIR) filters, and $u_{ij}(t)$ is the partial result generated from w_{ij}^k and $x_j(t)$.

With logistic sigmoid as the nonlinear transfer function, the stochastic learning rules derived from the Infomax approach for nonzero delay weights are

$$\Delta w_{ij}^k(t) \propto (1 - 2y_i(t))\, x_j(t-k), \qquad k \neq 0. \tag{8}$$

As for the zero delay weights, their stochastic learning rules are given as

$$\Delta w_{ij}^0(t) \propto (1 - 2y_i(t))\, x_j(t) + d_{ij}(t) \tag{9}$$

where $d_{ij}(t) = \text{cofactor}(w_{ij})\, (\det \mathbf{W}^0)^{-1}$; cofactor$(w_{ij})$ is the cofactor of w_{ij}; and det is the determinant.

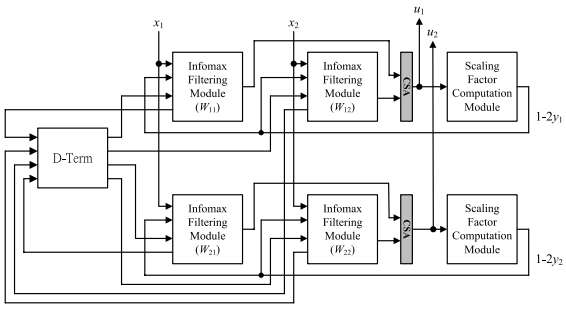

Fig. 2. Block diagram of the proposed CBSS chip that contains four Infomax filtering modules, two scaling factor computation modules, and a D-term unit. Two CSAs are used to sum up the Infomax filtering outputs.

III. Proposed VLSI Blind Source Separator

Fig. 2 shows the block diagram of the proposed CBSS chip. The CBSS chip consists mainly of two functional cores: Infomax filtering module and scaling factor computation module. Additionally, the Infomax filtering outputs are summed up using two small carry-save adders (CSAs). The current prototype chip is used for two sources and two sensors by adopting four Infomax filtering modules and two scaling factor computation modules.

A. VLSI Architecture for Infomax Filtering Module

Fig. 1 depicts the CBSS separation network, which contains four causal FIR filters. These filters are adaptive because their tap coefficients are altered by stochastic learning rules derived from the Infomax approach and are thus referred to herein as the Infomax adaptive filter or the Infomax filter. Equations (8) and (9) describe the stochastic learning rules to adjust the Infomax filter weights. Assume that the filter length of the Infomax filter is L, in which the stochastic learning rules can be written in matrix as

$$\begin{bmatrix} w_{ij}^0(t+1) \\ w_{ij}^1(t+1) \\ \vdots \\ w_{ij}^{L-1}(t+1) \end{bmatrix} = \begin{bmatrix} w_{ij}^0(t) \\ w_{ij}^1(t) \\ \vdots \\ w_{ij}^{L-1}(t) \end{bmatrix} + \mu s(t) \begin{bmatrix} x_j(t) \\ x_j(t-1) \\ \vdots \\ x_j\,(t-(L-1)) \end{bmatrix} + \begin{bmatrix} d_{ij}(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10}$$

where $\boldsymbol{x}_j(t) = [x_j(t), x_j(t-1), \ldots, x_j(t-L+1)]^T$ and $\boldsymbol{w}_{ij}(t) = [w_{ij}^0(t), w_{ij}^1(t), \ldots, w_{ij}^{L-1}(t)]^T$ represent the input vector and filter weight vector, respectively. Moreover, μ represents the step size of filter weight adaptation; and $s(t) = 1-2y_i(t)$ refers to a scaling factor with $y_i(t) = (1+e^{-u_i(t)})^{-1}$. The filter weight of each tap is updated using the scaling factor.

In (10), the stochastic learning rules for zero delay weight is nearly the same as those for nonzero delay weights, except for an extra added term $d_{ij}^0(t)$. Therefore, our designs for all of the weighting updating are in the same manner. Inspired by the architecture of the delayed least mean square adaptive filter [21], [22], the proposed Infomax filtering module is exemplified with six taps in Fig. 3.

In the Infomax filtering module, an input sample passes through lower and upper register chains. The input samples passing through the lower and upper register chains are multiplied with filter weights and scaling factors, respectively.

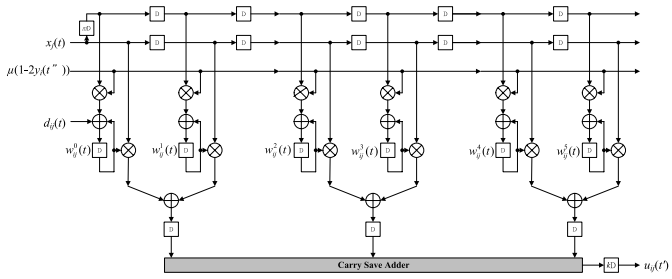

Fig. 3. Example of the proposed Infomax filtering module. The multiplication results of all of the taps are accumulated by a two-stage summation.

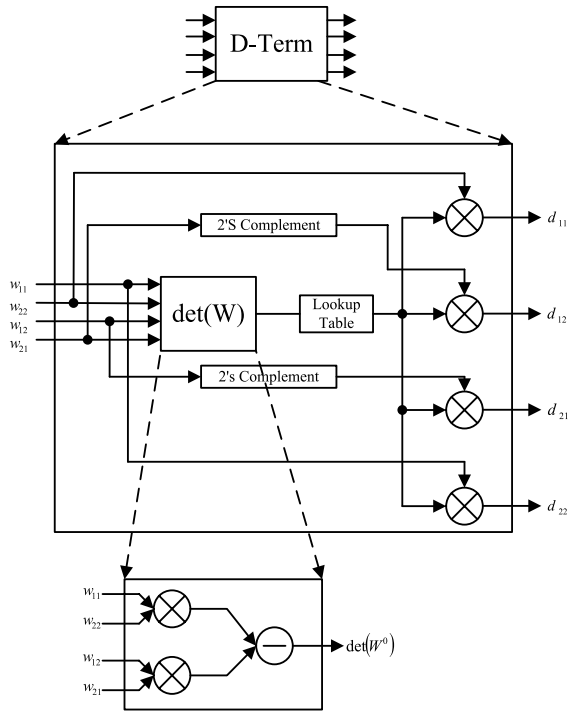

Fig. 4. Architecture of the D-term unit, which comprises a determinant circuit to obtain $\det \mathbf{W}^0$ and a lookup table to generate the inverse of $\det \mathbf{W}^0$.

The multiplication results of all of the taps are accumulated by a two-stage summation. The first stage adopts carry look-ahead adders to generate the intermediate addition results for multiplication of every two successive taps. The second stage sums the above intermediate addition results by using a carry-save addition scheme. A CSA can accept more than two data inputs. As this CSA may accept many intermediate addition results, reducing the critical path as low as $1T_a + 1T_m$ can be achieved by partitioning this CSA with pipeline registers. Here, T_a and T_m denote the critical paths of the carry look-ahead adder and multiplier, respectively. In Fig. 3, k pipeline registers are assumed to partition the CSA.

As for the $d_{ij}^0(t)$, this study designs a D-term unit to execute $d_{ij}(t) = \text{cofactor}(w_{ij})(\det \mathbf{W}^0)^{-1}$. The architecture of the D-term unit is depicted in Fig. 4. The D-term unit comprises a determinant circuit to obtain $\det \mathbf{W}^0$ and a lookup table to generate the inverse of $\det \mathbf{W}^0$. Since $\mathbf{W}$ is a 2×2 matrix, the cofactors(w_{ij}) are w_{22}, $-w_{21}$, $-w_{12}$, and w_{11}, which are multiplied by $(\det \mathbf{W}^0)^{-1}$ in parallel using four multipliers.

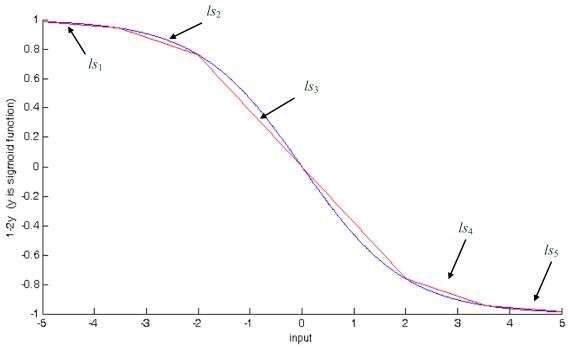

Fig. 5. Five line-segment approximation to the scaling factor computation, where ls_i denotes the ith line segment.

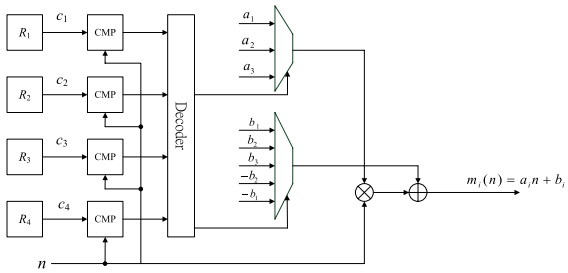

Fig. 6. Proposed scaling factor computation module. Four comparators and a decoder are used to determine the correct line segment.

B. VLSI Architecture for Scaling Factor Computation Module

The scaling factor used in filter weight updating, as described in (8) and (9), is obtained by calculating $s(t) = 1 - 2y(t)$, where $y(t) = (1 + e^{-u(t)})^{-1}$. For a straightforward computation flow, once $y(t)$ is available, $-2y(t)$ can be generated first using 2's complement and a left shift to $y(t)$.

The scaling factor $s(t)$ is then obtained by summing up $-2y(t)$ and one. The above procedure is simple. The emphasis of architecture design in the scaling factor computation module should thus lie in the logistic sigmoid computation.

In this brief, the logistic sigmoid computation is achieved based on a linear piecewise scheme [23], [24]. The scaling factor commutation is approximated directly rather than performing logistic sigmoid computation first and then calculating $1 - 2y(t)$. The target function to be approximated by linear piecewise scheme is

$$s(t) = 1 - 2 \Big/ \left(1 + e^{-u_i(t)}\right). \tag{11}$$

In our numerical analysis, five line segments are sufficient to approximate (11) with a negligible error. Let ls_i, $i = 1, 2, \ldots, 5$ denote the ith line segment, and c_i represent the connected point between two consecutive line segments, ls_i and ls_{i+1}. Fig. 5 plots the approximation results of the five line segments. The average error between the approximate value and the floating value obtained using (11) is around 2.88%.

To implement the line-segment approximation, the circuit design for scaling factor computation is to calculate single variable linear equations. Assume that the equation of ls_i is $m_i(n) = a_i n + b_i$, $i = 1, 2, \ldots, 5$, where $n = u_i(t)$. Fig. 6 describes the proposed circuit for the scaling factor computation module. The linear equation evaluation with input $u_i(t)$ and equation parameters a_i and b_i are implemented using a multiplier and an adder. A line segment is selected by two

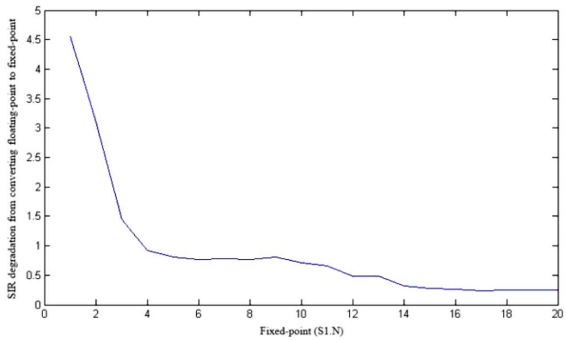

Fig. 7. SIR degradation from converting the floating-point algorithm into the fixed-point algorithm.

multiplexers to choose corresponding a_i and b_i. As the slopes of ls_1 and ls_5 are the same, these two line segments share the equation parameters a_1. In the same manner, line segments ls_2 and ls_4 share the equation parameters a_2. Furthermore, according to the symmetry in Fig. 5, the bias used for line segment ls_5, e.g., $-b_1$, is the negative of the bias b_1 used for line segment ls_1. In addition, line segments ls_4 and ls_2 use biases $-b_2$ and b_2, respectively.

The positions of connected points c_i, $i = 1, 2, \ldots, 4$, are stored using four registers R_i. Moreover, four comparators (CMP) are adopted to compare these positions with $u_i(t)$ in order to determine which line segment should be used for the approximation given in (11). The comparison results of the four comparators are sent to a decoder, allowing for selection of a correct line segment. Furthermore, the $m_i(n)$ value obtained from line-segment approximation is the scaling factor used for filter weight updating.

IV. Chip Implementation

We built the CBSS system using MATLAB software. The required finite word length accuracy is analyzed first by software simulation, allowing for implementation of the floating-point program by a fixed-point structure. The degradation in source-to-interference ratio (SIR) is adopted to measure the performance loss from converting the floating-point algorithm to the fixed-point algorithm. The fixed-point data are represented in the format S1.N, with one sign bit and N bits after the decimal point. Fig. 7 gives the results. The vertical axis represents the SIR degradation from converting the floating-point algorithm to the fixed-point algorithm. The horizontal axis represents N, the number of bits after the decimal point. This figure reveals that the SIR degradation tends to be rather small when N exceeds 14. Therefore, S1.14 is used as the fixed-point data format.

The chip design is implemented by using the TSMC 90-nm CMOS technology and the cell-based design flow. Hardware simulation of the proposed chip architecture was conducted by Verilog HDL, where a prototype chip was designed using Cadence's front-end and back-end tools.

Table I summarizes the chip specifications. The ASIC chip contains 68 pads. The total gate count is around 199 K with core size and die size of roughly 0.54×0.54 mm^2 and 0.99×0.99 mm^2, respectively. With a power supply of 1.8 V, the design can achieve 100 MHz in the worst case; in addition, the power dissipation is roughly 54.86 mW at this speed.

TABLE I
CHIP SPECIFICATION OF CBSS VLSI IMPLEMENTATION

CMOS Technology	TSMC 90 nm Standard Cell Library
Gate Count Number	199 K
Max. Clock Rate	100 MHz
Power Consumption	54.86 mW@1.8V, 100 MHz
Core Size	0.54x0.54 mm^2
Die Size	0.99x0.99 mm^2
Core Voltage	1.8V
I/O Voltage	3.3V
Pad Number	68

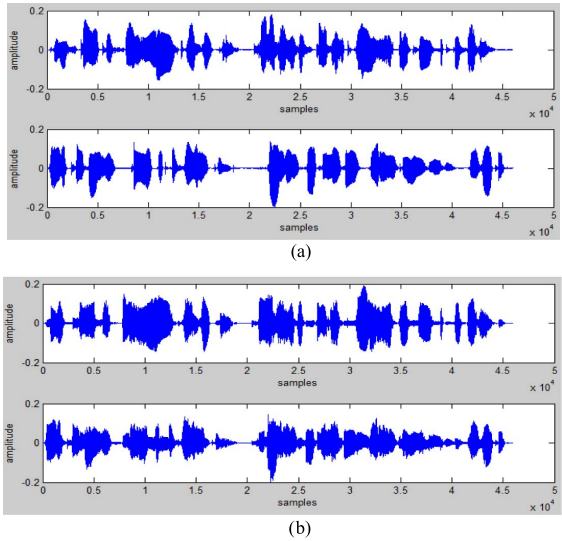

Fig. 8. CBSS separation results. (a) Source speech signals. (b) Separated speech signals.

Low power dissipation makes this chip appropriate for portable applications.

To evaluate the CBSS performance, mixed speech signals were measured in a room in which was placed a two-element microphone array. Twenty pairs of spoken sentences were played. The average length of these spoken sentences was around 3 s, and the sampling rate was 8 kHz. Fig. 8 gives an example of the CBSS separation results. Fig. 8(a) plots the two original source signals, which will be received and mixed by the two sensors. The proposed chip performed CBSS separation on the mixed signals, and Fig. 8(b) shows the two separated outputs. It is apparent that the proposed chip can yield satisfactory separated signals.

V. CONCLUSION

In this brief, an efficient VLSI architecture design for CBSS has been presented. The architecture mainly comprising Infomax filtering modules and scaling factor computation modules performs CBSS separation network derived from the Infomax approach. With TSMC 90-nm CMOS technology, the die size of the proposed ASIC chip is roughly 0.54×0.54 mm^2. For the 1.8-V power supply, the maximum clock rate is 100 MHz. The power dissipation is roughly 54.86 mW under the 100-MHz clock rate. The proposed CBSS ASIC chip can be used in preprocessing and integrated with other audio processing chips and peripheral components to form a whole audio processing system.

REFERENCES

[1] 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.
[2] M. Li, Y. Liu, G. Feng, Z. Zhou, and D. Hu, "OI and fMRI signal separation using both temporal and spatial autocorrelations," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 8, pp. 1917–1926, Aug. 2010.
[3] A. Tonazzini, I. Gerace, and F. Martinelli, "Multichannel blind separation and deconvolution of images for document analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 912–925, Apr. 2010.
[4] H. L. N. Thi and C. Jutte, "Blind source separation for convolutive mixtures," *Signal Process.*, vol. 45, no. 2, pp. 209–229, Aug. 1995.
[5] A. J. Bell and T. J. Sejnowski, "Blind separation and blind deconvolution: An information-theoretic approach," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, May 1995, vol. 5, pp. 3415–3418.
[6] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Netw.*, vol. 13, no. 4/5, pp. 411–430, May/Jun. 2000.
[7] M. H. Cohen and A. G. Andreou, "Analog CMOS integration and experimentation with an autoadaptive independent component analyzer," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 2, pp. 65–77, Feb. 1995.
[8] 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 Separation*, Dec. 2001, pp. 70–73.
[9] Z. Li and Q. Lin, "FPGA implementation of Infomax BSS algorithm with fixed-point number representation," in *Proc. Int. Conf. Neural Netw. Brain*, 2005, vol. 2, pp. 889–892.
[10] H. Du and H. Qi, "An FPGA implementation of parallel ICA for dimensionality reduction in hyperspectral images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Sep. 2004, pp. 3257–3260.
[11] M. Ounas, R. Touhami, and M. C. E. Yagoub, "Low cost architecture of digital circuit for FPGA implementation based ICA training algorithm of blind signal separation," in *Proc. Int. Symp. Signals, Syst. Electron.*, 2007, pp. 135–138.
[12] 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.
[13] A. Acharyya, K. Maharatna, J. Sun, B. M. Al-Hashimi, and S. R. Gunn, "Hardware efficient fixed-point VLSI architecture for 2D Kurtotic FastICA," in *Proc. Eur. Conf. Circuit Theory Des.*, Aug. 23–27, 2009, pp. 165–168.
[14] C. K. Chen *et al.*, "A low power independent component analysis processor in 90 nm CMOS technology for portable EEG signal processing systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 15–18, 2011, pp. 801–804.
[15] 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.
[16] K. Torkkola, "Blind separation of convolved sources based on information maximization," in *Proc. IEEE Signal Process. Soc. Workshop*, Sep. 1996, pp. 423–432.
[17] C. Jutten and J. Herault, "Blind separation of sources—Part I: An adaptive algorithm based on neuromimetic architecture," *Signal Process.*, vol. 24, no. 1, pp. 1–10, Jul. 1991.
[18] J. F. Cardoso, "Infomax and maximum likelihood for blind source separation," *IEEE Signal Process. Lett.*, vol. 4, no. 4, pp. 112–114, Apr. 1997.
[19] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Comput.*, vol. 7, pp. 1129–1159, Nov. 1995.
[20] M. S. Pedersen, J. Larsen, U. Kjems, and L. C. Parra, "A survey of convolutive blind source separation methods," in *Handbook of Speech Processing*. New York, NY, USA: Springer-Verlag, 2007.
[21] L. D. Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 4, pp. 359–366, Apr. 2001.
[22] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 9, pp. 1397–1405, Sep. 1989.
[23] C. Alippi and G. Storti-Gajani, "Simple approximation of sigmoidal functions: Realistic design of digital networks capable of learning," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1991, pp. 1505–1508.
[24] D. J. Myers and R. A. Hutchinson, "Efficient implementation of piecewise linear activation function for digital VLSI neural networks," *Electron. Lett.*, vol. 25, no. 24, pp. 1662–1663, Nov. 1989.