Hardware Efficient Fixed-Point VLSI Architecture for 2D Kurtotic FastICA

Amit Acharyya, Koushik Maharatna, Jinhong Sun, Bashir M. Al-Hashimi and Steve R. Gunn
Pervasive Systems Centre, School of Electronics and Computer Science,
University of Southampton, Southampton - SO17 1BJ, United Kingdom
Email: {aa07r, km3, js3v07, bmah, srg}@ecs.soton.ac.uk

***Abstract*— Fixed-point VLSI architecture for 2-Dimensional Kurtotic FastICA with reduced and optimized arithmetic units, is proposed. This reduction is achieved through the removal of the dividers for eigenvector computation and replacing the dividers in the Whitening block of the architecture by multipliers. In addition, the number of multipliers required in the Whitening block is further reduced by exploiting datapath symmetry present in that block. We have addressed also the numerical error issue associated with the finite wordlength representation of fixed-point arithmetic and proposed an efficient approach in dealing with such error. The proposed architecture occupies $3.55\ mm^2$ silicon area and consumes $27.1\ \mu$W power at 1.2 V @ 1 MHz using $0.13\ \mu$m standard cell CMOS technology.**

I. Introduction

Independent Component Analysis (ICA) is one of the most commonly used algorithm in blind source separation [1]. In emerging applications such as Wireless Sensor Network (WSN) [2], this property of ICA can be utilised [3]. Although different algorithms for ICA have been reported, the FastICA (FICA) algorithm has been shown to have advantage in terms of convergence speed [4]. Recently, the first VLSI implementation of FICA based on floating-point arithmetic has been reported in [5]. However it involves costly arithmetic operations including matrix inversion, square root evaluation, multiplication and division. Such implementation occupies large silicon area and consumes significant power and may not be suitable for resource-constrained applications like WSN where the sensors are typically battery powered and are expected to operate for long time. In addition floating-point arithmetic contributes to the silicon area and thus fixed-point arithmetic may be an optimal choice although a compromise with accuracy may be necessary.

In this paper we propose a hardware-efficient fixed-point architecture for 2-D Kurtotic FICA. The efficiency is achieved through (i) removal of division operation for eigenvector computation, (ii) replacement of division operations by multiplications and (iii) reduction of number of multipliers and adders for whitening matrix computation through detailed algorithmic analysis and exploiting the resulting architectural symmetry. Low numerical error is achieved by (i) introducing suitable Scaling Factors (SF) and (ii) internal data-bus width variability wherever necessary. The rest of the paper is organized as follows: brief introduction to FICA is given in Section II, and the proposed architecture is described in Section III. The performance analysis, validation and implementation results are given in Section IV and the conclusions are drawn in Section V.

II. Theoretical Background

In this paper we restrict ourselves to the common case where the number of independent sources (n) is equal to the number of sensors. A mixed signal (X) can be defined as [5]:

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

where $X = \{x_i\}$, $S = \{s_i\}$, $i \in (1, n)$; A is a full-rank $n \times n$ mixing matrix; $s_i = \{s_{i,j}\}$, $x_i = \{x_{i,j}\}$ where $j \in (1, m)$ and m is equal to the frame-length. To apply FICA first X needs a preprocessing step that converts it to a zero-mean signal (Centering process - Fig. 1) and then transforms this zero-mean X to a new vector Z whose components are uncorrelated with variances equal to unity (Whitening process - Fig. 1) [6]. This process is carried out by Eigen Value Decomposition (EVD) of the Covarience Matrix (C_X) of X. Mathematically, the vector Z can be described as [5]:

$$Z = PX = [D^{-1/2}E^T]X \tag{2}$$

where $Z = \{z_i\}$ and P is the whitening matrix, $D = \text{diag}(d_i)$ is a diagonal matrix containing the eigenvalues of C_X and $E = \{e_i\}$ is an orthonormal matrix of eigenvectors. The next step is to estimate the output vector $(\widehat{S})$ from Z by computing an unmixing matrix B of dimension $n \times n$ which can be defined as [5]:

$$\widehat{S} = B^T Z \tag{3}$$

The k^{th} column of B represents the weight vector w_k associated with k^{th} estimated independent component where $k \in (1, n)$. FICA computes w_k by introducing a contrast function within the basic iterative equation [6] and checks for its convergence (see Fig. 1). To prevent convergence of different w_k to the same maxima the generated vectors are orthonormalized after every iteration [6]. Once all the w_ks are derived, $\widehat{S}$ can be computed using (3).

III. Proposed FastICA Architecture

The block diagram of the complete FICA architecture is shown in Fig. 1. It is clear from Fig. 1 that the direct implementation of the complete FICA architecture requires complex arithmetic operations like division, square root and multiplication which are costly in terms of silicon area and

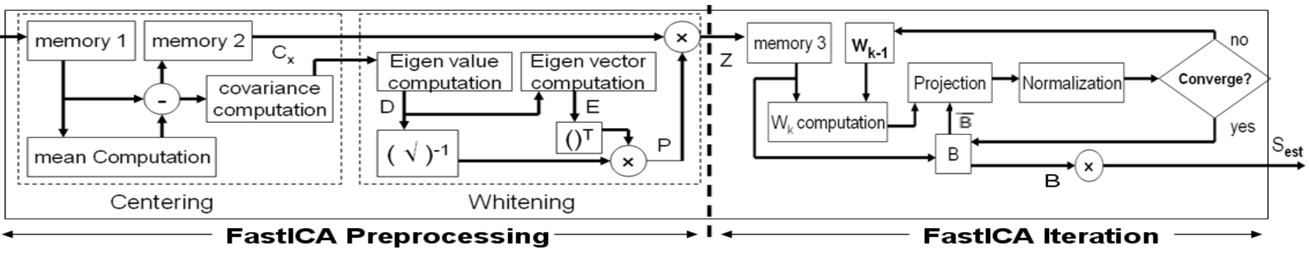

Fig. 1. Complete FastICA Architecture

power dissipation. To design an area and a power efficient FICA architecture special attention has to be paid to (i) optimize different arithmetic units and at the same time (ii) use the architectural symmetry to reduce number of arithmetic operations wherever possible.

The proposed architecture is for 2 dimensional (two source - two sensors) scenario and for demonstration purpose, we designed it considering frame-length 512 and wordlength 16-bit for the incoming data samples. The implementation of the Centering Unit is done by accumulating the incoming data samples and then shifting the accumulated sum to the right by nine bits (division by the frame-length = 512 = 2^9).

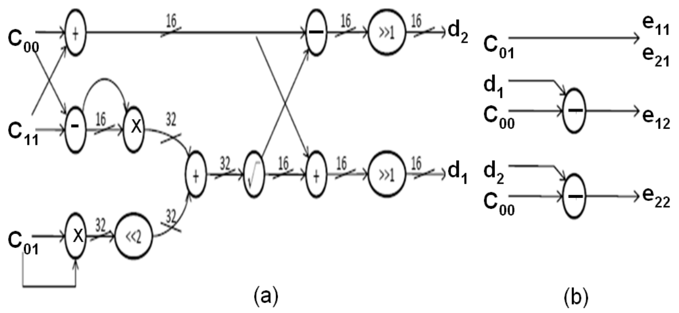

Fig. 2. Proposed divider-less architecture for (a) eigenvalue and (b) eigenvector computation

A. Eigenvalue and Eigenvector Computation (Divider Removal)

Solving the characteristic polynomial of C_X, the corresponding eigenvalues (d_1, d_2) can be given as:

$$d_1, d_2 = ((C_{00} + C_{11}) \pm \sqrt{(C_{00} - C_{11})^2 + 4C_{01}C_{10}})/2 \quad (4)$$

where C_{ij} are the elements of C_X. In this particular case C_{01} = C_{10} [5]. Fig. 2(a) shows the architectural implementation of this unit along with the variation of wordlengths adopted to keep the numerical error low. The term '$4C_{01}C_{10}$' is implemented using one multiplier and shifting the result by two bits left. The non-restoring square root algorithm [7], has been followed here to implement the square root circuit. Finally d_1 and d_2 are obtained by shifting the numerator of (4) right by one bit. Considering the above condition, the eigen matrix E can be given by:

$$E = \begin{bmatrix} e_{11} & e_{21} \\ e_{12} & e_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ (d_1 - C_{00})/C_{01} & (d_2 - C_{00})/C_{01} \end{bmatrix} \quad (5)$$

It is evident from (5) that to compute E one needs to compute only e_{12} and e_{22}. But it needs two division operations. To reduce the hardware complexity we apply the following concept : if an eigenvector e_i corresponding to the eigenvalue d_i satisfies the characteristic equation of C_X, then any scalar multiple of e_i will also satisfy the characteristic equation. From (5) we observe that the denominator C_{01} of e_{12} and e_{22} remains fixed for a frame and thus can be treated as a scalar quantity. Therefore, (5) can be modified as:

$$E = \begin{bmatrix} e_{11} & e_{21} \\ e_{12} & e_{22} \end{bmatrix} = \begin{bmatrix} C_{01} & C_{01} \\ d_1 - C_{00} & d_2 - C_{00} \end{bmatrix} \quad (6)$$

Comparing (5) and (6) it can be noted that the divider circuit can be removed completely from the eigenvector computation. The resulting simplified divider-less architecture is shown in Fig. 2(b).

B. P and Z computation (Replacement of Division by Multiplication)

Using the normalized form of (6), the whitening matrix P can be represented as:

$$P = \begin{bmatrix} e_{11}/\sqrt{d_1(e_{11}^2 + e_{12}^2)} & e_{12}/\sqrt{d_1(e_{11}^2 + e_{12}^2)} \\ e_{21}/\sqrt{d_2(e_{21}^2 + e_{22}^2)} & e_{22}/\sqrt{d_2(e_{21}^2 + e_{22}^2)} \end{bmatrix} \quad (7)$$

Using (7) in (2) the whitened data matrix Z can be defined as:

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} (e_{11}X_1 + e_{12}X_2)/\sqrt{d_1(e_{11}^2 + e_{12}^2)} \\ (e_{21}X_1 + e_{22}X_2)/\sqrt{d_2(e_{21}^2 + e_{22}^2)} \end{bmatrix} \quad (8)$$

The block diagram of the architecture for computing (8) is shown in Fig. 3(a). Since the denominators of $Z1$ and $Z2$ are constants for a frame, its value has been calculated only once at the beginning of each frame and stored in a memory (in the form of inverse). To maintain the accuracy of this term while applying the inversion operation, we represent decimal 1 as 2^{15}. This division operation is performed following the approach presented in [8]. For rest of the data within the frame this value has been used repeatedly as a multiplication factor (shown as the link "inverse divisor" in Fig. 3(a)) and thereby translating divisions into multiplications. The data wordlengths adopted at different parts of this circuit are also shown in Fig. 3(a). Observing the dataflow graph from Fig. 3(a) one can do further optimization for the section shown within the dashed line segment by exploiting the data-flow symmetry. The optimized circuit is shown in Fig. 3(b) where three multipliers have been replaced by three simple multiplexers reducing the hardware cost further.

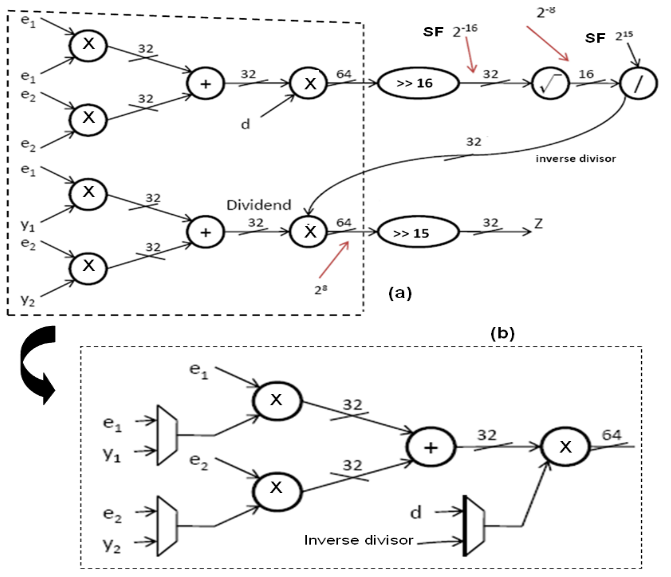

Fig. 3. (a) Proposed architecture of the Whitening block with divider to multiplier translation, (b) optimized architecture of the segment surrounded by dashed line in (a).

C. Fixed-point w_k Computation, Projection and Normalization

Fig. 4(a) shows the block diagram of w_k computation unit within FICA Iteration block (see Fig.1). Computation of w_k based on Kurtosis contrast function [5], involves computing 4^{th} power of the whitened data. Since the output wordlength of the whitening unit is 32 bits, we use 64-bit internal wordlength for the w_k computation unit. The effect of inclusion of the SF ($= 2^{16}$) before square-rooting (Fig. 3(a) and sub-section III-B), after each arithmetic operation inside the FICA block is clearly shown in Fig. 4(a). However from practical wordlength consideration, when the overall data scaling reaches 2^{24} (shown in the sub-block (a-2) of Fig. 4(a)), we downscaled it by 16-bit. The block diagrams of the projection and normalization units are shown in Fig. 4(b) and Fig. 4(c) respectively, where $\bar{B} = [\bar{b}_1, \bar{b}_2]$, is the previously determined column of B. The architecture in Fig. 4(b) is derived from the following projection equation [5], [6]:

$$w_k \leftarrow \begin{bmatrix} w_{1,k} \\ w_{2,k} \end{bmatrix} - \begin{bmatrix} \bar{b}_1(\bar{b}_1 w_{1,k} + \bar{b}_2 w_{2,k}) \\ \bar{b}_2(\bar{b}_1 w_{1,k} + \bar{b}_2 w_{2,k}) \end{bmatrix} \tag{9}$$

The normalization operation of w_k in the normalization unit, raises the same concern of loosing the information contained in the fractional part of the normalized data as discussed in

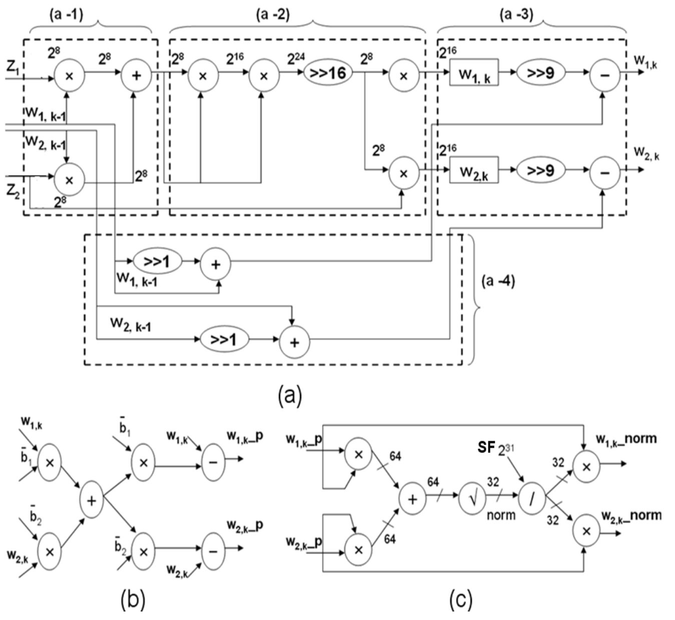

Fig. 4. Fixed-point architecture of (a) w_k computation, (b) Projection and (c) Normalization unit. '$w_{i,k}$_P' and '$w_{i,k}$_norm' represent "projected" and "normalized" $w_{i,k}$ respectively.

subsection III-B. To overcome this problem, as shown in Fig. 4(c), an SF $= 2^{31}$ is introduced in the design.

IV. Results and Discussion

A. Functional Validation and Performance Analysis

To do the functional validation of the proposed architecture, we generated a C-model of the FICA algorithm. The functional output of this model is compared with the corresponding Verilog model of the proposed architecture. As test vectors, we have chosen the data samples as given in [9] equivalent to one frame-length. Due to page limitation only two sets of comparison results are shown in Fig. 5. The estimated outputs from the C model and Verilog model are shown in the left and right side of the Fig 5 respectively. As can be seen from Fig. 5, there is close correlation between the waveforms confirming correct functionality of the proposed fixed-point FICA architecture.

To examine the overall effect of the numerical error accumulation on the estimated output, we have plotted the probability of error vs bit position in Fig. 6. It can be seen from Fig. 6, mostly the error occurs 8^{th} bit-position onwards which is acceptable from practical implementation point of view.

Following the traditional approach of measuring the computational complexity, we have determined total number of arithmetic operations involved in 2D Kurtotic FICA process as shown in Table I. It can be observed from Table I that the proposed architecture needs 15 more additions and 1019 multiplications than the direct-mapping method, but requires 1049 less division operations than the direct-mapping one considering maximum iteration for convergence $M = 5$. However, since the hardware complexity of divider in terms

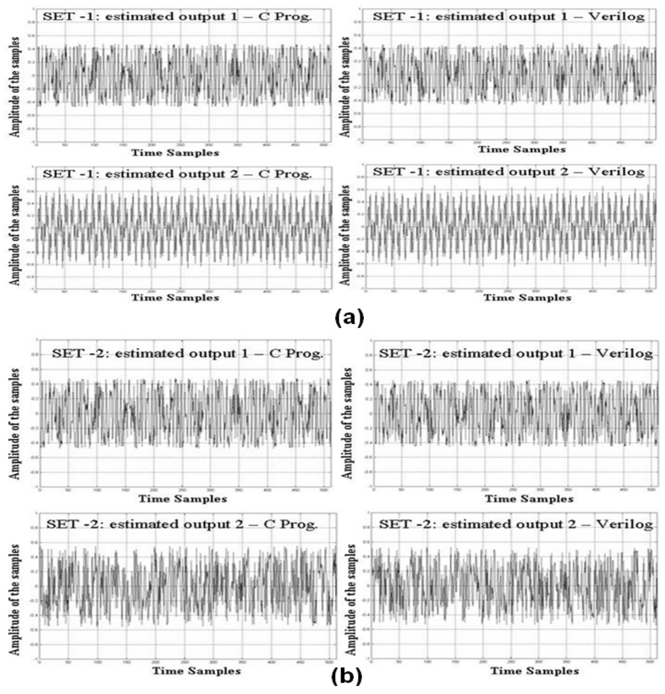

Fig. 5. Left side - Estimated waveforms generated from the C model, right side - estimated waveforms generated from the Verilog model of the proposed FICA architecture. (a) SET 1 results, (b) SET 2 results.

of gate-count and delay is much higher than that of multiplier and adder, the effective gain (as shown in Table I) in hardware complexity of the proposed architecture is expected to be greater than the direct mapping approach.

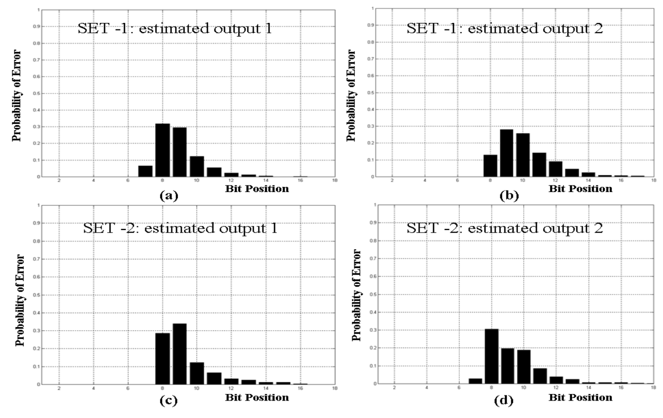

Fig. 6. Probability of error vs. bit position in the proposed architecture. (a) Set-1, estimated source-1, (b) Set-1, estimated source-2, (c) Set-2, estimated source-1, (d) Set-2, estimated source-2.

B. Implementation Results

To give insight into the area and power cost, the proposed low complexity architecture is coded using Verilog and synthesized using Synopsys Design Compiler in 0.13μm standard cell CMOS technology. The synthesized area and power consumption are $3.55\ mm^2$ and $27.1\ \mu$W @ 1 MHz frequency for $V_{DD} = 1.2$ V. The power value is obtained by feeding continuously 16-bit random vectors equal to one frame-length into the synthesized netlist and applying Synopsys Prime Time. Since, to the best our knowledge, there is no such published results available on fixed-point implementation in terms of area requirement and power consumption, we were unable to compare these parameters with any other work.

TABLE I

COMPARISON IN TERMS OF NUMBER OF ARITHMETIC OPERATIONS. "M" IS THE MAXIMUM NUMBER OF ITERATIONS FOR CONVERGENCE OF w_k. POSITIVE GAIN DENOTES THE SAVINGS IN HARDWARE AND NEGATIVE DEONOTES THE OVERHEAD.

Architec ture	Add	Subtract	Multiply	Divide	Sq.Root
Direct [5]	4608 $+3074M$	1028 $+8M$	5640 $+6163M$	1031 $+6M$	$3+2M$
Proposed	4608 $+3077M$	1028 $+8M$	6664 $+6160M$	$2+2M$	$3+2M$
Gain	$-3M$	0	$-(1024$ $-3M)$	$+(1029$ $+4M)$	0

V. CONCLUSION

We proposed in this paper a hardware-efficient fixed-point VLSI architecture of the FICA algorithm. The detail architecural description showed that it is possible to reduce the hardware complexity by optimizing different arithmetic units instead of direct one-to-one mapping of the algorithm into architecture. In the proposed architecture, the reduction in hardware leads to the low power consumption and thereby making it a good candidate for WSN applications. However, the impact of different algorithmic parameters e.g. frame-length, convergence threshold, are under investigation which may lead to further architectural optimization.

REFERENCES

[1] H. Du, H. Qi and X. Wang, "Comparative Study of VLSI Solutions to Independent Component Analysis", *IEEE Trans. Industrial Electronics*, vol. 54, no. 1, February, 2007.

[2] D. Estrin, D. Culler, K. Pister and G. Sukhatme."Connecting the Physical World with Pervasive Networks", *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 59-69, 2002.

[3] B. Lo, F. Deligianni and G. Z. Yang, "Source Recovery for Body Sensor Network", *IEEE International Workshop on Wearable and Implantable Body Sensor Networks*, April, 2006.

[4] E. Oja and Z. Yuan, "The FastICA Algorithm Revisited: Convergence Analysis", *IEEE Trans. Neural Networks*, vol. 17, no. 6, November, 2006.

[5] 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 Networks*, vol. 19, no. 6, pp. 958-970, June, 2008.

[6] A. Hyvärinen, "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis", *IEEE Trans. Neural Networks*, vol. 10, no. 3, May, 1999.

[7] Y. Li and W. Chu, "A New Non-Restoring Square Root Algorithm and Its VLSI Implementation", *IEEE International Conference on Computer Design*, pp. 538-544, 1996.

[8] J. P. Deschamps, G. J. A. Bioul and G. D. Sutter, "Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems", *John Wiley and Sons Inc.*, 2006.

[9] A. Cichocki and R. Unbehauen, "Robust Neural Networks with On-Line Learning for Blind Identification and Blind Separation of Sources", *IEEE Trans. Circuits and Systems-I: Fundamental Theory and Applications*, Vol. 43, no. 11, pp. 894-906, November, 1996.