Co-ordinate Rotation based Low Complexity N-D FastICA Algorithm and Architecture

Amit Acharyya, Koushik Maharatna, *Member, IEEE,* Bashir M. Al-Hashimi, *Fellow, IEEE,* and Jeff Reeve, *Senior Member, IEEE,*

Abstract

This paper proposes a low complexity n-dimensional (nD) FastICA algorithm and architecture by introducing the concept of co-ordinate rotation where $n \geq 2$. The proposed algorithm can merge the two key steps of Conventional FastICA - Preprocessing and Update and is therefore capable of reducing the hardware complexity of the conventional FastICA significantly as demonstrated in this paper. Hardware implementation can further be simplified due to the recursive nature of the proposed algorithm where the same $2D$ hardware module can be used as the fundamental core to implement nD architecture. Together with the algorithm formulation, its functionality is also validated and hardware complexity is analyzed and compared with the conventional nD FastICA.

Index Terms

Independent Component Analysis, FastICA, Blind Source Separation, CORDIC, Low Complexity Algorithm and Architecture.

EDICS Category: HDW-AAOP - Algorithm and architecture co-optimization

The Authors are with the Pervasive Systems Centre (PSC) research group in School of Electronics and Computer Science (ECS), University of Southampton, SO17 1BJ, UK e-mail: (aa07r, km3, bmah, jsr)@ecs.soton.ac.uk

Preliminary version of this research (see Section III) has appeared in [13].

I. Introduction

Signal separation is an important requirement in the emerging fields such as wireless sensor networks and mobile healthcare technologies [1]-[2]. Devices used for such emerging applications are mostly battery-powered and need to be tiny and unobtrusive imposing constraints on the design in terms of power and area. It necessitates the development of low-complexity light-weight signal processing techniques. Although Independent Component Analysis (ICA) can have potential applications in these fields [3] and FastICA (FICA) is popular among existing ICA algorithms because of its higher convergence speed and accuracy [4]-[5], the computationally intensive nature makes the direct algorithm to architecture mapping of FICA unsuitable for such resource constrained applications. Therefore an algorithm-architecture holistic optimization approach is necessary for maintaining its algorithmic efficiency and making it 'low-complexity' from the architectural perspective at the same time which is the main focus of our research.

Conventional FICA algorithm consists of two steps - Preprocessing and Update [5]. Preprocessing step mainly involves computationally intensive Eigen Value Decomposition (EVD) which is implemented using Coordinate Rotation Digital Computer (CORDIC) in hardware [6]-[12]. The aim of this paper is to investigate whether the same CORDIC can be used to propose FICA Update step so that costly arithmetic operations involving division, square root evaluation and multiplications can be removed from the algorithm. Recently we introduced the concept of co-ordinate rotation in the basic $2D$ FICA and subsequently proposed its corresponding CORDIC based algorithm and architecture in [13]. We have also shown that such concept is capable of reducing the hardware complexity significantly for $2D$ FICA and pointed out that unlike other reported $2D$ architectures [14]-[15], this concept can possibly be generalized for nD.

In this paper, we generalize this co-ordinate rotation concept of $2D$ into higher-dimensional nD space ($n \geq 3$) by formulating CORDIC based recursive nD FICA algorithm using $2D$ as the fundamental core, present its corresponding architecture, explore the possibility of further architectural optimization and analyze the hardware complexity of the proposed CORDIC based nD FICA in detail.

The rest of this paper is organized as follows: Section II does the necessary theoretical groundwork on FICA and CORDIC, Section III discusses in brief our preliminary research on CORDIC based $2D$ FICA as reported in [13], Section IV proposes the CORDIC based nD FICA algorithm, Section V validates the proposed algorithm, Section VI presents the corresponding architecture and explores further optimization possibility, Section VII analyzes the hardware complexity of this proposed algorithm over the conventional one and Section VIII concludes the discussion.

II. Theoretical Background

A. Conventional FICA Algorithm

Denoting independent sources by S, mixed signal (X) can be defined as [5]: $X = AS$, where $X = \{x_i\}$, $S = \{s_i\}$, $i \in (1, n)$; A is a full-rank $n \times n$ mixing matrix where n is the number of independent sources; $s_i = \{s_{i,j}\}$, $x_i = \{x_{i,j}\}$ where $j \in (1, m)$ and m is equal to the frame-length. FICA, as mentioned in the last section, comprises of two steps - Preprocessing and Update. Since our focus here is on FICA Update step, discussion of Preprocessing step is omitted for brevity.

FICA Update step computes the outputs ($\widehat{S}$) from the whitened mixed signals (Z) (as obtained after preprocessing) by estimating the unmixing matrix B of dimension $n \times n$ which can be defined as [5]:

$$\widehat{S} = B^T Z \tag{1}$$

The k^{th} column of B represents the estimated vector $\mathbf{w_k}$ associated with k^{th} Independent Components (IC) to be computed where $k \in (1, n)$. FICA stage frames an adaptive iterative equation by introducing a contrast function (g) to maximize non-gaussianity (which is a measure of independence) for estimating ICs as follows [5]:

$$\mathbf{w_k} \leftarrow \mathcal{E}\{Z\dot{g}(\mathbf{w_k}^T Z)\} - \mathcal{E}\{\ddot{g}(\mathbf{w_k}^T Z)\}\mathbf{w_k} \tag{2}$$

where $\dot{g}$ and $\ddot{g}$ are the first and second derivative of g respectively. Considering *Kurtosis*-based contrast function, (2) can be modified as [5]:

$$\mathbf{w_k} \leftarrow \mathcal{E}\{Z(\mathbf{w_{k-1}}^T Z)^3\} - 3\mathbf{w_{k-1}} \tag{3}$$

If $k = 1$, the next step to follow is the normalization of the estimated vector [5] as:

$$\underline{\mathbf{w}}_\mathbf{k} \leftarrow \mathbf{w_k}/\|\mathbf{w_k}\| \tag{4}$$

where "_" indicates *normalized value* in rest of the paper. When $k > 1$, to prevent different $\mathbf{w_k}$ from converging to the same maxima the estimated vectors need to be orthogonalized before this normalization step. If the estimated normalized vector $\mathbf{w_k}$ has not converged, the whole Update step needs to be repeated [5]. This FICA Update step continues until $k = n$ is reached. Then using B in (1), output ICs $\widehat{S}$ are obtained.

B. Coordinate Rotation Digital Computer

CORDIC is an efficient implementation technique for vector rotation and arctangent computation and since it can be realized using simple shift and add operations, it is very effective in terms of low hardware complexity [16] - [18]. CORDIC, in general, can be operated in two modes - rotation and vectoring [18].

In rotation mode, given the angle of rotation and the initial vector, final vector is computed and in the vectoring mode, the angle between the initial vector and the principal coordinate axis is computed. Considering the rotation in clockwise sense, the basic CORDIC expressions can be expressed as [16] - [18]:

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{5}$$

where x_0, y_0 and x_f, y_f are the initial and final components of the vector and the angle of rotation is θ. In Rotation mode, angle θ is approximated by a sequence of micro-angles and after a finite number of iterations x_f and y_f are generated. In vectoring mode, where angle θ is unknown, y_f is forced to zero after finite number of iterations.

In this paper, now onwards, instead of using the complete expression shown in (5), following concise notations are used for brevity: $x_f = \mathcal{R}ot_x(x_0, y_0, \theta)$ and $y_f = \mathcal{R}ot_y(x_0, y_0, \theta)$, and $\theta = \mathcal{V}ec_\theta(x_0, y_0)$ where $\mathcal{R}ot_{x/y}(\cdot)$ denotes the x/y component of the rotation mode CORDIC outputs and $\mathcal{V}ec_{\theta/x}$ denotes the θ/x output of the Vectoring mode CORDIC.

III. CORDIC BASED $2D$ FICA

In this section we briefly discuss our initial research on CORDIC based $2D$ FICA algorithm as reported in [13].

A. 2D FICA Iteration Stage

For $2D$ FICA (i.e. $n = 2$), expanded form of (3) can be written as:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)}\}^3] \\ \mathcal{E}[z_{2,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \end{bmatrix} \tag{6}$$

where p denotes the number of iteration stage, $z_{i,j}$ represents the i^{th} whitened data containing j number of samples where $i = \{1, 2\}$ and $j \in (1, m)$ where m denotes the frame-length, ${w_{1,q}}^{(p+1)}$ is the 1^{st} column of the unmixing matrix after p^{th} iteration where $q = \{1, 2\}$ and ${\underline{w}_{1,q}}^{(p)}$ indicates the *normalized value* of ${w_{1,q}}^{(p)}$ used in p^{th} iteration as given by (4). Since ${\underline{w}_{1,1}}^{(p)}$ and ${\underline{w}_{1,2}}^{(p)}$ are the components of a unit norm vector ${\underline{\mathbf{w}}_1}^{(p)}$, using Cartesian to Polar transformation, ${\underline{\mathbf{w}}_1}^{(p)}$ can be written as:

$${\underline{\mathbf{w}}_1}^{(p)} = [{\underline{w}_{1,1}}^{(p)} \ {\underline{w}_{1,2}}^{(p)}]^T = [\cos\theta_p \ \sin\theta_p]^T \tag{7}$$

where polar angle $\theta_p = \tan^{-1}(\underline{w}_{1,2}{}^{(p)}/\underline{w}_{1,1}{}^{(p)})$ at p^{th} iteration stage. Using (7) and expressing (6) in terms of $\mathcal{R}ot$ and $\mathcal{V}ec$ notations, we get:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \theta_p)\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \theta_p)\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \end{bmatrix} \tag{8}$$

and

$$\theta_p = \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}) \tag{9}$$

Using (9) and (8), conventional $2D$ FICA Iteration stage architecture can be designed using rotation and vectoring mode CORDIC as shown in Fig. 1(a).

B. 2D FICA Normalization Stage

Denoting the normalized components of the vector obtained after p^{th} iteration using (6) by $\underline{w}_{1,1}{}^{(p+1)}$ and $\underline{w}_{1,2}{}^{(p+1)}$ the following equation holds:

$$\underline{w}_{1,i}{}^{(p+1)} = \frac{w_{1,i}{}^{(p+1)}}{\sqrt{|\mathbf{w}_1{}^{(p+1)}|^2}} \tag{10}$$

where $i = \{1, 2\}$. Using Cartesian to Polar Co-ordinate transformation following (7), (10) can be represented as:

$$\underline{\mathbf{w}}_1{}^{(p+1)} = [\underline{w}_{1,1}{}^{(p+1)} \ \underline{w}_{1,2}{}^{(p+1)}]^T = [\cos\theta_{(p+1)} \ \ \sin\theta_{(p+1)}]^T \tag{11}$$

If input vector in (5) is $[x_0 \ y_0] = [0 \ 1]$, output vector becomes $[x_f = \sin\theta \ y_f = \cos\theta]$. Therefore (11) can be written as:

$$\begin{bmatrix} \underline{w}_{1,1}{}^{(p+1)} \\ \underline{w}_{1,2}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,1}{}^{(p+1)}, w_{1,2}{}^{(p+1)})) \\ \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,1}{}^{(p+1)}, w_{1,2}{}^{(p+1)})) \end{bmatrix} \tag{12}$$

Fig. 1(b) shows the corresponding architecture of $2D$ FICA normalization stage.

C. 2D FICA Component Estimation Stage

Denoting the converged normalized vector by $\underline{\mathbf{w}}_1{}^c = [\underline{w}_{1,1}{}^c \ \underline{w}_{1,2}{}^c]^T$, the estimated component by $\hat{s}_1 = \{\hat{s}_{1,j}\}$ where $j \in (1, m)$ and m being the frame-length and using the same set of arguments used to derive (8), (1) can be written as:

$$\begin{aligned} \hat{s}_{1,j} &= z_{1,j}\underline{w}_{1,1}{}^c + z_{2,j}\underline{w}_{1,2}{}^c \\ &= \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(\underline{w}^c_{1,1}, \underline{w}^c_{1,2})) \end{aligned} \tag{13}$$

Fig. 1(c) presents the corresponding architecture of the $2D$ FICA component estimation stage. It is important to note that the vectoring mode CORDIC needs not to be used in this stage because the

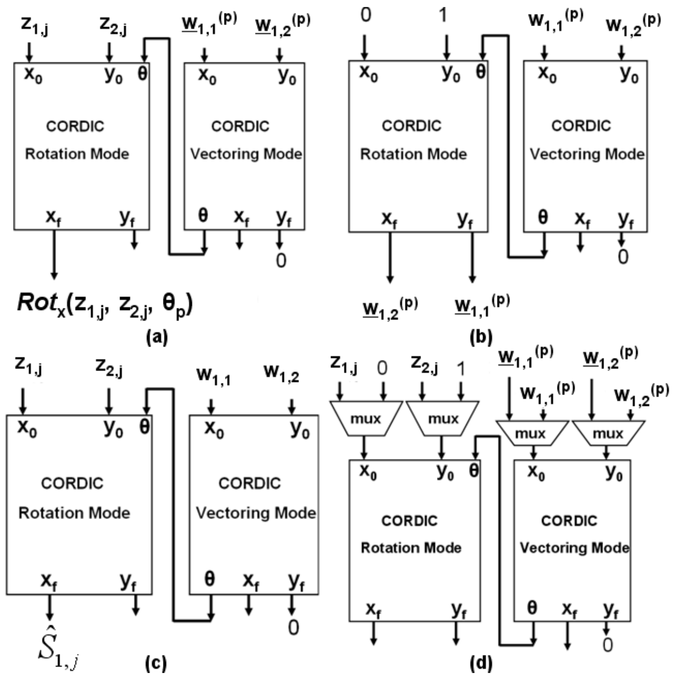

Fig. 1. CORDIC based $2D$ FICA. (a) Iteration, (b) Normalization, (c) Estimation stage and (d) Multiplexed architecture.

accumulated angle information is already computed during the previous normalization step and can be reused as the θ input of the Rotation mode CORDIC. Due to this reason relinquishing "c" from (13), $\hat{s}_{1,j}$ can be rewritten as:

$$\hat{s}_{1,j} = \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) \tag{14}$$

where $w_{1,1}$ and $w_{1,2}$ are the components of the unnormalized vector obtained after FICA iteration prior to convergence checking.

D. Multiplexed Architecture: CORDIC Reuse for $2D$ FICA

Since iteration, normalization and estimation - all stages are executed sequentially, same CORDIC unit can be reused for implementing these stages only at the expense of multiplexers at the inputs of the rotation and vectoring mode CORDIC. This multiplexed architecture, capable of executing the entire $2D$ FICA algorithm is shown in 1(d).

IV. PROPOSED CORDIC BASED nD FICA

Following the same procedure used in the last section to formulate the stages of $2D$ FICA, we proceed here to propose CORDIC based nD FICA algorithm in the following subsections.

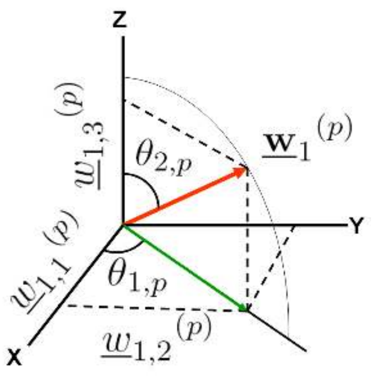
Z
Y
X
$\underline{\mathbf{w}}_1{}^{(p)}$
$\theta_{2,p}$
$\theta_{1,p}$
$\underline{w}_{1,3}{}^{(p)}$
$\underline{w}_{1,1}{}^{(p)}$
$\underline{w}_{1,2}{}^{(p)}$

Fig. 2. 3D representation of $\underline{\mathbf{w}}_1{}^{(p)}$ in Spherical Co-ordinate system.

A. nD FICA Iteration Stage

Considering first the case of 3D FICA (i.e. $n = 3$) and defining a new term G_{3D} by:

$$G_{3D} = z_{1,j}\underline{w}_{1,1}{}^{(p)} + z_{2,j}\underline{w}_{1,2}{}^{(p)} + z_{3,j}\underline{w}_{1,3}{}^{(p)} \tag{15}$$

(3) can be expanded as follows:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \\ w_{1,3}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{G_{3D}\}^3] \\ \mathcal{E}[z_{2,j}\{G_{3D}\}^3] \\ \mathcal{E}[z_{3,j}\{G_{3D}\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \\ \underline{w}_{1,3}{}^{(p)} \end{bmatrix} \tag{16}$$

where the symbols have usual meanings as stated in Section II-A. Once again, like (7), applying Cartesian to Polar transformation, (16) can be written as:

$$\underline{\mathbf{w}}_1{}^{(p)} = \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \\ \underline{w}_{1,3}{}^{(p)} \end{bmatrix} = \begin{bmatrix} \sin\theta_{2,p}\cos\theta_{1,p} \\ \sin\theta_{2,p}\sin\theta_{1,p} \\ \cos\theta_{2,p} \end{bmatrix} \tag{17}$$

where, the spherical angles $\theta_{1,p}$ and $\theta_{2,p}$ at p^{th} iteration stage, as shown in Fig. 2, are defined as $\tan^{-1}(\underline{w}_{1,2}{}^{(p)}/\underline{w}_{1,1}{}^{(p)})$ and $\tan^{-1}(\underline{w}_{1,3}{}^{(p)}/\sqrt{(\underline{w}_{1,1}{}^{(p)})^2 + (\underline{w}_{1,2}{}^{(p)})^2})$ respectively.

Now using (17), (15) can be re-written using $\mathcal{R}ot$ and $\mathcal{V}ec$ as:

$$G_{3D} = \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \theta_{1,p}), \theta_{2,p}) \tag{18}$$

and

$$\begin{aligned} \theta_{1,p} &= \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}) \\ \theta_{2,p} &= \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})) \end{aligned} \tag{19}$$

where $l*$ denotes the *levels* of *Rotation/Vectoring* mode CORDIC required as also shown in Fig. 3. Using (19) and (18), the conventional $3D$ FICA Iteration stage can be mapped into the rotation and vectoring mode CORDIC as shown in Fig. 3(a).

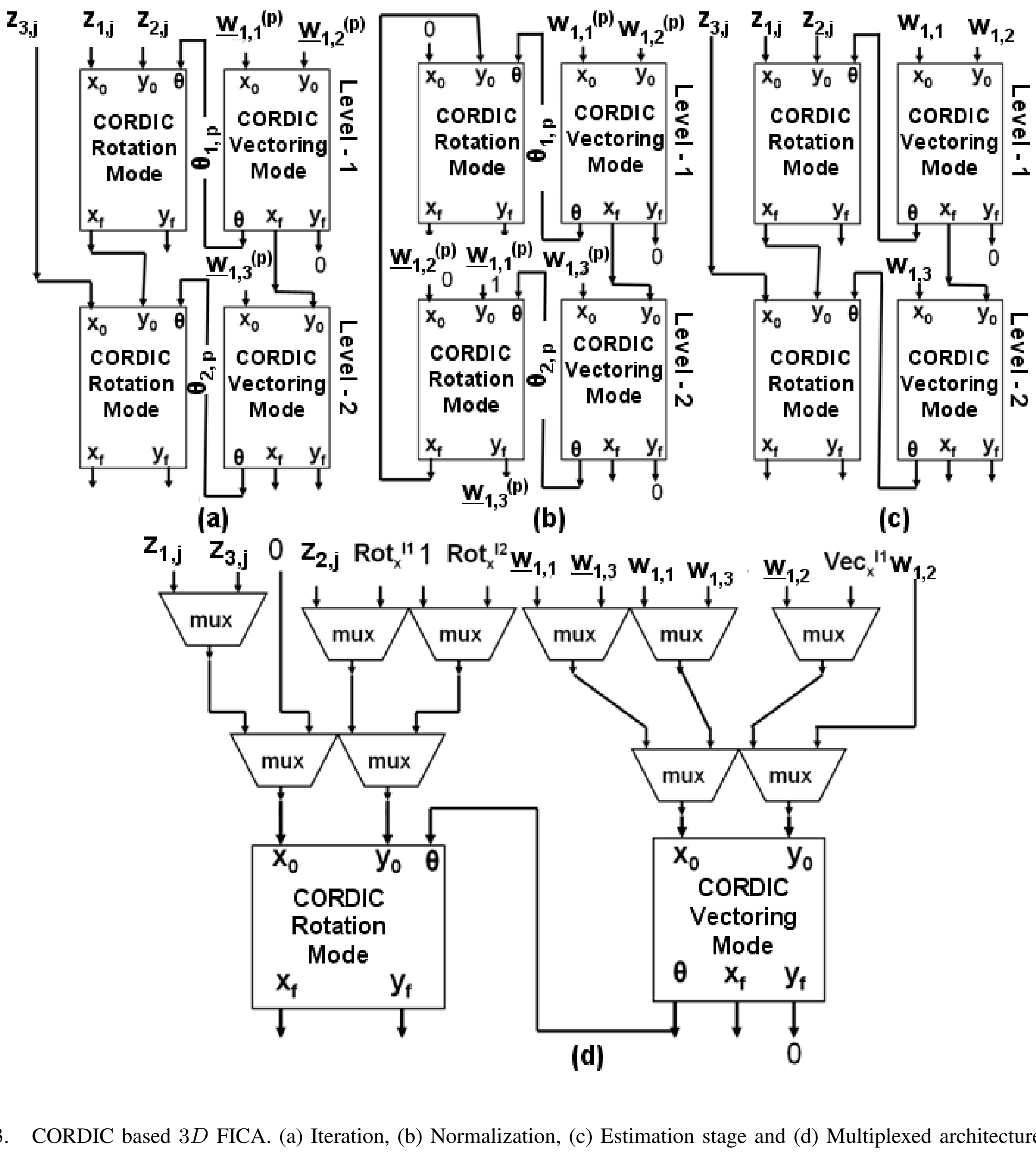

Fig. 3. CORDIC based $3D$ FICA. (a) Iteration, (b) Normalization, (c) Estimation stage and (d) Multiplexed architecture.

Similarly for 4D FICA (i.e. $n = 4$) iteration, defining G_{4D} by:

$$G_{4D} = z_{1,j}\underline{w}_{1,1}^{(p)} + z_{2,j}\underline{w}_{1,2}^{(p)} + z_{3,j}\underline{w}_{1,3}^{(p)} + z_{4,j}\underline{w}_{1,4}^{(p)} \quad (20)$$

(3) can be expanded as:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \\ w_{1,3}{}^{(p+1)} \\ w_{1,4}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{G_{4D}\}^3] \\ \mathcal{E}[z_{2,j}\{G_{4D}\}^3] \\ \mathcal{E}[z_{3,j}\{G_{4D}\}^3] \\ \mathcal{E}[z_{4,j}\{G_{4D}\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \\ \underline{w}_{1,3}{}^{(p)} \\ \underline{w}_{1,4}{}^{(p)} \end{bmatrix} \tag{21}$$

Following the same approach used to get (7) and (17), $\underline{\mathbf{w}}_1{}^{(p)}$ can be represented as:

$$\begin{aligned} \underline{\mathbf{w}}_1{}^{(p)} &= [\underline{w}_{1,1}{}^{(p)} \ \underline{w}_{1,2}{}^{(p)} \ \underline{w}_{1,3}{}^{(p)} \ \underline{w}_{1,4}{}^{(p)}]^T \\ &= \begin{bmatrix} \sin\theta_{3,p}\sin\theta_{2,p}\cos\theta_{1,p} \\ \sin\theta_{3,p}\sin\theta_{2,p}\sin\theta_{1,p} \\ \sin\theta_{3,p}\cos\theta_{2,p} \\ \cos\theta_{3,p} \end{bmatrix} \end{aligned} \tag{22}$$

where, angles $\theta_{1,p}$, $\theta_{2,p}$ and $\theta_{3,p}$ at p^{th} iteration stage, following the same convention used after (17), can be defined as: $\tan^{-1}(\underline{w}_{1,2}{}^{(p)}/\underline{w}_{1,1}{}^{(p)})$, $\tan^{-1}(\underline{w}_{1,3}{}^{(p)}/\sqrt{(\underline{w}_{1,1}{}^{(p)})^2 + (\underline{w}_{1,2}{}^{(p)})^2})$ and

$$\tan^{-1}(\underline{w}_{1,4}{}^{(p)}/\sqrt{(\underline{w}_{1,1}{}^{(p)})^2 + (\underline{w}_{1,2}{}^{(p)})^2 + (\underline{w}_{1,3}{}^{(p)})^2})$$

respectively. Using (22), (20) can be expressed as:

$$G_{4D} = \mathcal{R}ot_x^{l3}(z_{4,j}, \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \theta_{1,p}), \theta_{2,p}), \theta_{3,p}) \tag{23}$$

and $\theta_{1,p}$, $\theta_{2,p}$ and $\theta_{3,p}$ can be derived as follows:

$$\begin{aligned} \theta_{1,p} &= \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}) \\ \theta_{2,p} &= \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})) \\ \theta_{3,p} &= \mathcal{V}ec_\theta^{l3}(\underline{w}_{1,4}^{(p)}, \mathcal{V}ec_x^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}))) \end{aligned} \tag{24}$$

Combination of (24) and (23) represents the mapping of the conventional $4D$ FICA Iteration stage in terms of the rotation and vectoring mode CORDIC as shown in Fig. 4(a). *Observation* - 1*:* From (18), (19) and Fig. 3(a) for $3D$ and (23), (24) and Fig. 4(a) for $4D$ *Iteration* stage, it can be observed that all *levels* of Rotation and Vectoring mode are connected in Cascaded Feed Forward fashion.

Now by comparing the $2D$, $3D$ and $4D$ FICA Iteration stages, it can be noticed that there exists some kind of similarity which may be extended to the n^{th}-dimension. To understand this similarity, let us

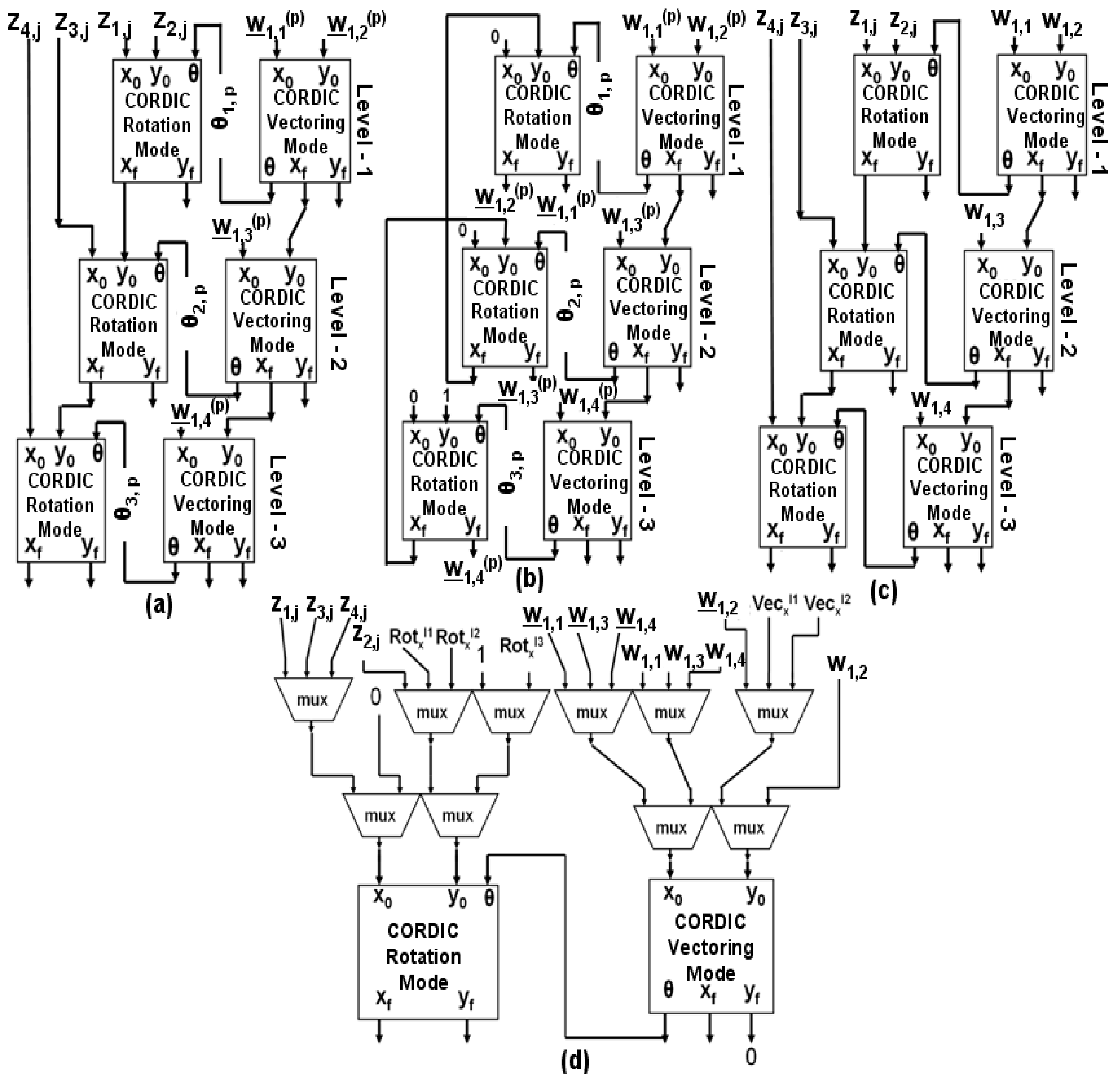

Fig. 4. CORDIC based $4D$ FICA. (a) Iteration, (b) Normalization, (c) Estimation stage and (d) Multiplexed architecture.

define the following terms for $2D$ FICA:

$$\begin{aligned} \mathcal{R}_{x,j}^{2D} &= \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)})) \\ \mathcal{V}_\theta^{2D} &= \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}) \\ \mathcal{V}_x^{2D} &= \mathcal{V}ec_x(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}) \end{aligned} \tag{25}$$

where $\mathcal{R}_{x,j}^{2D}$ may be read as "x-output corresponding to the j^{th} input of the Rotation mode 2D CORDIC" and so on. Re-writing (18) using (25) we get:

$$G_{3D} = \mathcal{R}ot_x(z_{3,j}, \mathcal{R}^{2D}_{x,j}, \mathcal{V}ec_\theta(\underline{w}^{(p)}_{1,3}, \mathcal{V}^{2D}_x)) = \mathcal{R}^{3D}_{x,j} \tag{26}$$

Subsequently following the similar notations used in (25) for 2D, following notations are introduced for 3D:

$$\begin{aligned} \mathcal{V}^{3D}_x &= \mathcal{V}ec_x(\underline{w}^{(p)}_{1,3}, \mathcal{V}^{2D}_x) \\ \mathcal{V}^{3D}_\theta &= \mathcal{V}ec_\theta(\underline{w}^{(p)}_{1,3}, \mathcal{V}^{2D}_x) \end{aligned} \tag{27}$$

Similarly for $4D$ FICA Iteration stage, (23) and (24), can be rewritten as:

$$\begin{aligned} G_{4D} &= \mathcal{R}ot_x(z_{4,j}, \mathcal{R}^{3D}_{x,j}, \mathcal{V}ec_\theta(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_x)) = \mathcal{R}^{4D}_{x,j} \\ \mathcal{V}^{4D}_x &= \mathcal{V}ec_x(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_x) \\ \mathcal{V}^{4D}_\theta &= \mathcal{V}ec_\theta(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_x) \end{aligned} \tag{28}$$

Comparing (25), (26) and (28), it can be seen that (26) can be realised by recursive use of (25) and (28) can be realised by recursively using (26) and hence by (25). Therefore, since (25) involves only one CORDIC *Rotation* and one *Vectoring* in 2D plane, any higher dimensional ($n > 2$) FICA Iteration stage can be realized using the 2D CORDIC Rotation and Vectoring. Thus it is possible to propose a generalized theorem for realizing nD FICA Iteration stage by recursive use of $(n-1)$D FICA Iteration where $n > 2$ using 2D *Rotation* and *Vectoring* mode CORDIC.

Theorem 1. **CORDIC based Recursive Formulation of the Iteration Stage of nD FICA Algorithm:**

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \\ w_{1,3}{}^{(p+1)} \\ . \\ . \\ . \\ w_{1,n}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \\ . \\ . \\ . \\ \mathcal{E}[z_{n,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \\ \underline{w}_{1,3}{}^{(p)} \\ . \\ . \\ . \\ \underline{w}_{1,n}{}^{(p)} \end{bmatrix} \tag{29}$$

where, considering $\mathcal{R}^{2D}_{x,j} = \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}))$ as the basic Rotation mode and $\mathcal{V}^{2D}_\theta = \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)})$; $\mathcal{V}^{2D}_x = \mathcal{V}ec_x(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)})$ as the basic Vectoring mode (as obtained from (25)) of CORDIC operation, for $n \geq 3$, Iteration stage of the nD FICA algorithm can be expressed in recursive way as follows:

$$\begin{aligned}
\mathcal{R}_{x,j}^{nD} &= \mathcal{R}ot_x(z_{n,j}, \mathcal{R}_{x,j}^{(n-1)D}, \mathcal{V}ec_\theta(\underline{w}_{1,n}^{(p)}, \mathcal{V}_x^{(n-1)D})) \\
\mathcal{V}_x^{nD} &= \mathcal{V}ec_x(\underline{w}_{1,n}^{(p)}, \mathcal{V}_x^{(n-1)D}) \\
\mathcal{V}_\theta^{nD} &= \mathcal{V}ec_\theta(\underline{w}_{1,n}^{(p)}, \mathcal{V}_x^{(n-1)D})
\end{aligned} \tag{30}$$

Proof: This theorem can easily be proved by following the fore-mentioned logical arguments establishing the relationship between (25), (26), (27) and (28) for 2D, 3D and 4D cases and extending this concept to the higher dimensions ($n > 2$) using method of induction. ■

B. nD FICA Normalization Stage

Considering $3D$ FICA normalization stage first and denoting the normalized components of the vector for $3D$ FICA obtained after p^{th} iteration using (16) by $\underline{w}_{1,1}{}^{(p+1)}$, $\underline{w}_{1,2}{}^{(p+1)}$ and $\underline{w}_{1,3}{}^{(p+1)}$, Cartesian to Polar Co-ordinate transformation yields:

$$\begin{aligned}
\underline{\mathbf{w}}_1{}^{(p+1)} &= \begin{bmatrix} \underline{w}_{1,1}{}^{(p+1)} \\ \underline{w}_{1,2}{}^{(p+1)} \\ \underline{w}_{1,3}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \sin\theta_{2,(p+1)}\cos\theta_{1,(p+1)} \\ \sin\theta_{2,(p+1)}\sin\theta_{1,(p+1)} \\ \cos\theta_{2,(p+1)} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{R}ot_y^{l1}(0, \mathcal{R}ot_x^{l2}(0,1,\theta_{2,(p+1)}), \theta_{1,(p+1)}) \\ \mathcal{R}ot_x^{l1}(0, \mathcal{R}ot_x^{l2}(0,1,\theta_{2,(p+1)}), \theta_{1,(p+1)}) \\ \mathcal{R}ot_y^{l2}(0,1,\theta_{2,(p+1)}) \end{bmatrix}
\end{aligned} \tag{31}$$

and $\theta_{1,(p+1)}$ and $\theta_{2,(p+1)}$ can be derived as:

$$\begin{aligned}
\theta_{1,(p+1)} &= \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}) \\
\theta_{2,(p+1)} &= \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))
\end{aligned} \tag{32}$$

Combination of (32) and (31) maps the conventional $3D$ FICA normalization stage into Rotation and Vectoring mode CORDIC as shown in Fig. 3(b).

Following the same approach as above, normalized components of the computed vector for $4D$ FICA can be represented as (33) as shown on page 13. $\theta_{1,(p+1)}$, $\theta_{2,(p+1)}$ and $\theta_{3,(p+1)}$ can be derived as:

$$\begin{aligned}
\theta_{1,(p+1)} &= \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}) \\
\theta_{2,(p+1)} &= \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})) \\
\theta_{3,(p+1)} &= \mathcal{V}ec_\theta^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_x^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, \\
&\qquad w_{1,2}^{(p+1)})))
\end{aligned} \tag{34}$$

$$\begin{aligned}\underline{\mathbf{w}}_1{}^{(p+1)} = [\underline{w}_{1,1}{}^{(p+1)}\ \underline{w}_{1,2}{}^{(p+1)}\ \underline{w}_{1,3}{}^{(p+1)}\ \underline{w}_{1,4}{}^{(p+1)}]^T &= \begin{bmatrix}\sin\theta_{3,(p+1)}\sin\theta_{2,(p+1)}\cos\theta_{1,(p+1)}\\ \sin\theta_{3,(p+1)}\sin\theta_{2,(p+1)}\sin\theta_{1,(p+1)}\\ \sin\theta_{3,(p+1)}\cos\theta_{2,(p+1)}\\ \cos\theta_{3,(p+1)}\end{bmatrix}\\ &= \begin{bmatrix}\mathcal{R}ot_y^{l1}(0,\mathcal{R}ot_x^{l2}(0,\mathcal{R}ot_x^{l3}(0,1,\theta_{3,(p+1)}),\theta_{2,(p+1)}),\theta_{1,(p+1)})\\ \mathcal{R}ot_x^{l1}(0,\mathcal{R}ot_x^{l2}(0,\mathcal{R}ot_x^{l3}(0,1,\theta_{3,(p+1)}),\theta_{2,(p+1)}),\theta_{1,(p+1)})\\ \mathcal{R}ot_y^{l2}(0,\mathcal{R}ot_x^{l3}(0,1,\theta_{3,(p+1)}),\theta_{2,(p+1)})\\ \mathcal{R}ot_y^{l3}(0,1,\theta_{3,(p+1)})\end{bmatrix}\end{aligned} \tag{33}$$

Fig. 4(b) presents its corresponding architecture.

Observation - 2: From (31), (32) and Fig. 3(b) for $3D$ and (33), (34) and Fig. 4(b) for $4D$ *normalization* stage, it can be observed that the *levels* of Vectoring mode are connected in feed forward fashion, where as the *levels* of Rotation mode are connected in feed backward fashion.

To derive the generic formulation for nD FICA Normalization stage, following the same approach used in the last section, here also we define the θ and x-output of the Vectoring mode of the CORDIC for $2D$ FICA as follows:

$$\begin{aligned}\mathcal{V}_\theta^{2D} &= \mathcal{V}ec_\theta(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}) = \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})\\ \mathcal{V}_x^{2D} &= \mathcal{V}ec_x(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}) = \mathcal{V}ec_x(w_{1,1}, w_{1,2})\end{aligned} \tag{35}$$

It is to be noted here that the superfix "p" is removed for the sake of simplicity. Following the same notation convention, the θ and x-output of the Vectoring mode CORDIC for 3D FICA Normalization stage can be written as:

$$\begin{aligned}\mathcal{V}_\theta^{3D} &= \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\\ &= \mathcal{V}ec_\theta(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2}))\\ &= \mathcal{V}ec_\theta(w_{1,3}, \mathcal{V}_x^{2D})\\ \mathcal{V}_x^{3D} &= \mathcal{V}ec_x(w_{1,3}, \mathcal{V}_x^{2D})\end{aligned} \tag{36}$$

Similarly for 4D case following set of equations can be derived using (36):

$$\begin{aligned}\mathcal{V}_\theta^{4D} &= \mathcal{V}ec_\theta^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_x^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})))\\ &= \mathcal{V}ec_\theta(w_{1,4}, \mathcal{V}ec_x(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2})))\\ &= \mathcal{V}ec_\theta(w_{1,4}, \mathcal{V}_x^{3D})\\ \mathcal{V}_x^{4D} &= \mathcal{V}ec_x(w_{1,4}, \mathcal{V}_x^{3D})\end{aligned} \tag{37}$$

Lemma 1*:* The θ and x-output of the Vectoring mode CORDIC for nD can be represented in terms of $(n-1)D$ as follows:

$$\begin{aligned}\mathcal{V}_\theta^{nD} &= \mathcal{V}ec_\theta^{l_{(n-1)}}(w_{1,n}, \mathcal{V}_x^{(n-1)D}) = \mathcal{V}ec_\theta(w_{1,n}, \mathcal{V}_x^{(n-1)D})\\ \mathcal{V}_x^{nD} &= \mathcal{V}ec_x^{l_{(n-1)}}(w_{1,n}, \mathcal{V}_x^{(n-1)D}) = \mathcal{V}ec_x(w_{1,n}, \mathcal{V}_x^{(n-1)D})\end{aligned} \tag{38}$$

Proof: As has been discussed above, comparing (35), (36) and (37), it can be found that (37) ($4D$) can be expressed in terms of (36) ($3D$) and this can be also be expressed in terms of (35) ($2D$). Proceeding the same way and using method of induction this *lemma* can be proved. ■

Using these notations and removing $(p+1)$-term for the sake of simplicity, (12) can be re-framed as follows:

$$\begin{aligned}\underline{w}_{1,1} &= \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) = \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{2D})\\ &= \mathcal{R}_y^{2D}\\ \underline{w}_{1,2} &= \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) = \mathcal{R}_x^{2D}\end{aligned} \tag{39}$$

Similarly for $3D$ FICA Normalization stage, (31) can be expressed as:

$$\begin{aligned}\underline{w}_{1,3} &= \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2})))\\ &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{3D}) = \mathcal{R}_y^{3D}\\ \underline{w}_{1,1} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) = \mathcal{R}_y^{2D}\\ \underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) = \mathcal{R}_x^{2D}\end{aligned} \tag{40}$$

Proceeding the same way, for 4D FICA Normalization stage, (33) can be re-framed as:

$$\begin{aligned}\underline{w}_{1,4} &= \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,4}, \mathcal{V}ec_x(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2}))))\\ &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{4D}) = \mathcal{R}_y^{4D}\\ \underline{w}_{1,3} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{4D}, \mathcal{V}_\theta^{3D}) = \mathcal{R}_y^{3D}\\ \underline{w}_{1,1} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) = \mathcal{R}_y^{2D}\\ \underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) = \mathcal{R}_x^{2D}\end{aligned} \tag{41}$$

Comparing (39), (40) and (41), it can be observed that the y-inputs of the *Rotation* mode for $\underline{w}_{1,1}$, $\underline{w}_{1,2}$ and $\underline{w}_{1,3}$ are different for $2D$, $3D$ and $4D$ cases. This is also clearly shown in Fig. 1(b), 3(b) and 4(b).

Based on the above discussion the following generalized recursive algorithm for CORDIC based nD FICA normalization stage can be proposed:

Theorem 2*:* **CORDIC based Recursive Formulation of the Normalization Stage of nD FICA Algorithm:**

(a) When $n = 2$; $\underline{w}_{1,1}$ and $\underline{w}_{1,2}$ can be represented as:

$$\begin{aligned} \underline{w}_{1,1} &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{2D}) \\ \underline{w}_{1,2} &= \mathcal{R}ot_x(0, 1, \mathcal{V}_\theta^{2D}) \end{aligned} \tag{42}$$

(b) When $n = 3$; $\underline{w}_{1,1}$, $\underline{w}_{1,2}$ and $\underline{w}_{1,3}$ can be represented as:

$$\begin{aligned} \underline{w}_{1,1} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ \underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ \underline{w}_{1,3} &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{3D}) \end{aligned} \tag{43}$$

(c) When $n \geq 4$;

(i) for $i = (n-1), (n-2), ..., 3$; $\underline{w}_{1,i}$ can be represented as:

$$\begin{aligned} \underline{w}_{1,i} = \mathcal{R}_y^{iD} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{(i+1)D}, \mathcal{V}_\theta^{iD}) \\ &\text{and} \\ \underline{w}_{1,n} = \mathcal{R}_y^{nD} &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{nD}) \end{aligned} \tag{44}$$

(ii) $\underline{w}_{1,1}$ and $\underline{w}_{1,2}$ can be represented as:

$$\begin{aligned} \underline{w}_{1,1} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) = \mathcal{R}_y^{2D} \\ \underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) = \mathcal{R}_x^{2D} \end{aligned} \tag{45}$$

Proof. (a) Proof follows straightway from the same procedure used to derive (39) using *Lemma-1*.

(b) The procedure discussed above to derive (40) can be used along with *Lemma-1* to prove (43).

(c) Considering $n = 4$, it was shown in (41) that the 4^{th} component is the y-output of the *Rotation* mode when 0 and 1 are fed to its x and y inputs respectively. The x-output of this Rotation mode is fed back as the y-input of this mode to obtain the 3^{rd} component as shown in (41). This same analytical treatment can be extended in higher dimensions where $n > 4$ and part *(i)* can be proved.

It can be observed from (41) that the 1^{st} and the 2^{nd} components of $4D$ are always the y and x outputs of the same *Rotation* mode whose x-input is 0 and y-input is the x-output of the rotation mode

CORDIC used for the 3rd component computation. Proceeding the same way it can be shown that it holds for higher dimension $n > 4$ as well. This proves part *(ii)*. ■

C. nD FICA Component Estimation Stage

Considering again $3D$ case first and denoting the converged normalized vector by $\underline{\mathbf{w}}_1{}^c = [\underline{w}_{1,1}{}^c \ \underline{w}_{1,2}{}^c \ \underline{w}_{1,3}{}^c]^T$ and the estimated component by $\hat{s}_1 = \{\hat{s}_{1,j}\}$, like (18) and (19), for $3D$ FICA, (1) can explicitly be written as:

$$\begin{aligned}\hat{s}_{1,j} &= z_{1,j}\underline{w}_{1,1}{}^c + z_{2,j}\underline{w}_{1,2}{}^c + z_{3,j}\underline{w}_{1,3}{}^c \\ &= \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c)), \mathcal{V}ec_\theta^{l2} \\ &\quad (\underline{w}_{1,3}^c, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c)))\end{aligned} \tag{46}$$

Fig. 3(c) presents its corresponding architecture. As discussed in Section III-C, here also Vectoring needs not to be used in *estimation* mode because the angle is already computed during the previous normalization step and can be reused as the θ input of the Rotation mode CORDIC. Therefore relinquishing "c", (46) can be modified to:

$$\begin{aligned}\hat{s}_{1,j} &= \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(w_{1,1}, w_{1,2})), \mathcal{V}ec_\theta^{l2} \\ &\quad (w_{1,3}, \mathcal{V}ec_x^{l1}(w_{1,1}, w_{1,2})))\end{aligned} \tag{47}$$

Similarly for 4D FICA Component Estimation stage, estimated component $\hat{s}_1$ can be expressed as:

$$\begin{aligned}\hat{s}_{1,j} &= z_{1,j}\underline{w}_{1,1}{}^{(p)} + z_{2,j}\underline{w}_{1,2}{}^{(p)} + z_{3,j}\underline{w}_{1,3}{}^{(p)} + z_{4,j}\underline{w}_{1,4}{}^{(p)} \\ &= \mathcal{R}ot_x^{l3}(z_{4,j}, \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(w_{1,1}, \\ &\quad w_{1,2})), \mathcal{V}ec_\theta^{l2}(w_{1,3}, \mathcal{V}ec_x^{l1}(w_{1,1}, w_{1,2}))), \mathcal{V}ec_\theta^{l3} \\ &\quad (w_{1,4}, \mathcal{V}ec_x^{l2}(w_{1,3}, \mathcal{V}ec_x^{l1}(w_{1,1}, w_{1,2}))))\end{aligned} \tag{48}$$

Fig. 4(c) presents its corresponding architecture.

Observation - 3*:* From (47) and Fig. 3(c) for $3D$ and (48) and Fig. 4(c) for $4D$ *estimation* stage, it can be observed that all *levels* of the Rotation and Vectoring mode are connected in feed forward fashion.

Based on the above discussion, the following recursive formula can be framed for nD FICA Component Estimation Stage.

Theorem 3. **CORDIC based Recursive Formulation of the Estimation Stage of nD FICA Algorithm:**

$$\hat{s}_{1,j} = \mathcal{R}_{x,j}^{nD} \tag{49}$$

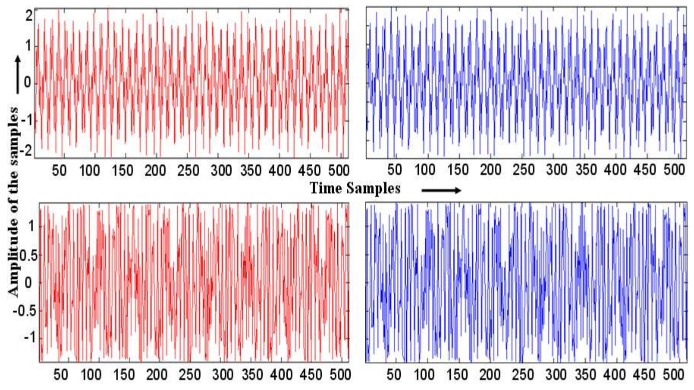

Fig. 5. Left side - estimated waveforms from conventional $2D$ FICA, right side - estimated waveforms from proposed CORDIC based $2D$ FICA.

where, considering $\mathcal{R}_{x,j}^{2D} = \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2}))$ as the basic Rotation mode and $\mathcal{V}_\theta^{2D} = \mathcal{V}ec_\theta(w_{1,1}, w_{1,2}); \mathcal{V}_x^{2D} = \mathcal{V}ec_x(w_{1,1}, w_{1,2})$ as the basic Vectoring mode of CORDIC operation, for $n \geq 3$, Estimation stage of the nD FICA algorithm can be expressed in recursive way as follows:

$$\begin{aligned}\mathcal{R}_{x,j}^{nD} &= \mathcal{R}ot_x(z_{n,j}, \mathcal{R}_{x,j}^{(n-1)D}, \mathcal{V}ec_\theta(w_{1,n}, \mathcal{V}_x^{(n-1)D}))\\ \mathcal{V}_x^{nD} &= \mathcal{V}ec_x(w_{1,n}, \mathcal{V}_x^{(n-1)D})\\ \mathcal{V}_\theta^{nD} &= \mathcal{V}ec_\theta(w_{1,n}, \mathcal{V}_x^{(n-1)D})\end{aligned} \quad (50)$$

Proof: The notations adopted to prove *Theorem-1* can also be used here to prove the above theorem. Comparing (14) ($2D$) with (47) ($3D$) Estimation stage, it can be observed that 3D Estimation needs one more *Rotation* over the 2D Estimation. In the similar way, comparing (47) with (48) for 4D Estimation stage, it can be seen that the 4D stage needs one more *Rotation* mode over the 3D stage. Continuing this way for higher dimensions where $n > 4$, it can also be shown that nD Estimation stage needs one more *Rotation* mode over the $(n-1)$D Estimation stage. Thus the estimated component for the nD FICA stage can be represented in recursive fashion as one extra rotation over the previous $(n-1)$D FICA estimation stage considering 2D stage as the fundamental unit. This proves the above theorem. ■

V. Algorithm Validation

To validate the proposed algorithm, we generated C models for both the conventional [5] as well as the proposed co-ordinate rotation based $2D$ and $3D$ FICA, compiled using GNU C Compiler in the Linux Platform and compared the performance of the estimated outputs.

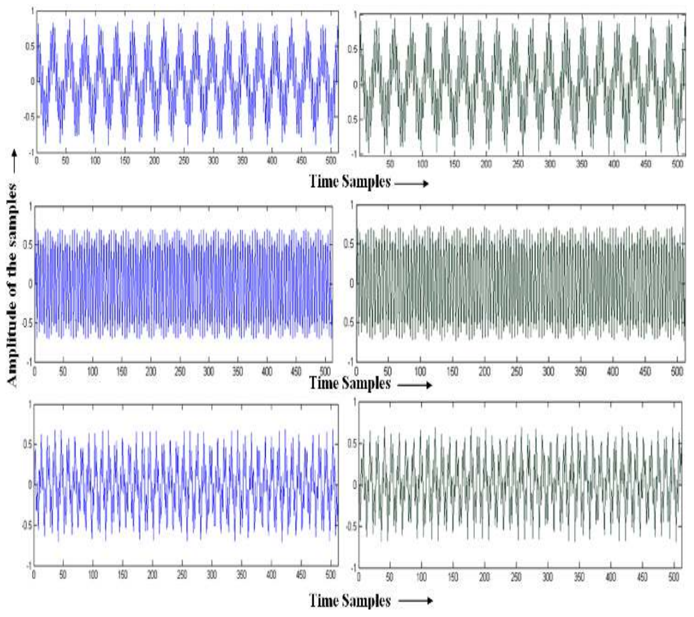

Fig. 6. Left side - estimated waveforms from conventional $3D$ FICA, right side - estimated waveforms from proposed CORDIC based $3D$ FICA.

Left hand side of Fig. 5 and 6 plot the outputs from the conventional FICA and right hand side shows the outputs from the proposed CORDIC based $2D$ and $3D$ FICA respectively. It is evident from these figures that the proposed co-ordinate rotation based $2D$ and $3D$ FICA algorithms maintain the same functionality as that of the conventional FICA algorithm without sacrificing its algorithmic efficiency. Since there is no gold standard present to measure the performance of ICA algorithms, like many other research papers, visual inspection is considered here as a metric for performance comparison [19]. However, Mean Squared Error (MSE) between the sources and the estimated outputs from the proposed algorithm are also computed. MSEs for Fig. 5 (starting from the top) are 3.0863^{-4} and 4.63×10^{-4} and for Fig. 6 (starting from the top), MSEs are 4.5722×10^{-4}, 5.7386×10^{-4} and 5.1314×10^{-4} respectively.

VI. Proposed CORDIC Based nD FICA Architecture

As mentioned in section III-D, one *Rotation* and *Vectoring* mode CORDIC can be reused for implementing all stages of the proposed $2D$ FICA only at the expense of multiplexers at the inputs of these two modes. Unlike $2D$, each stage of nD FICA ($n > 2$) consists of $(n-1)$ levels. Since none of these levels is concurrent, it is possible to fold all the levels together into a single level of CORDIC but at the expense of additional multiplexers. For $3D$ and $4D$ cases, such multiplexed architectures are shown in

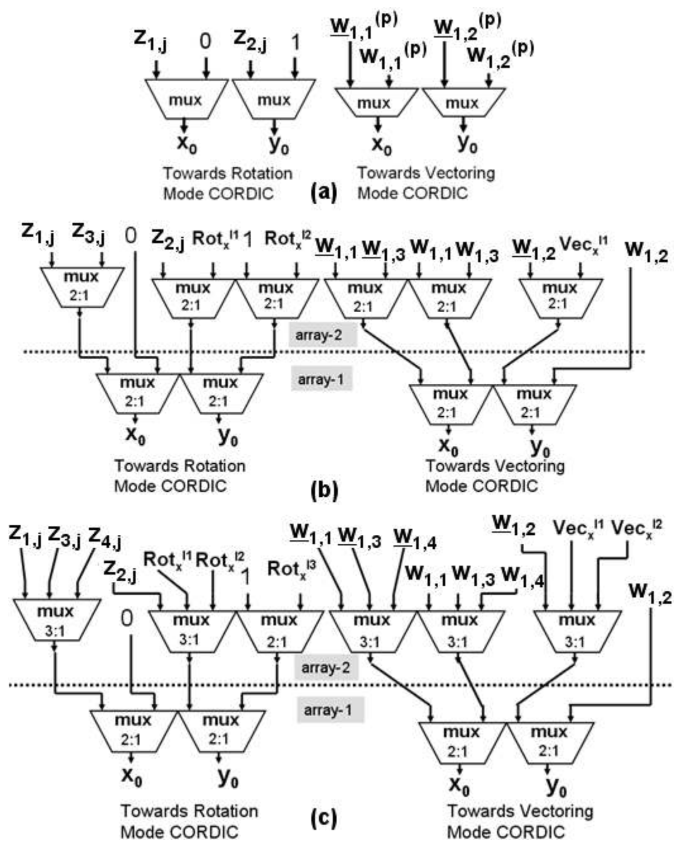

Fig. 7. Multiplexer array used at the front CORDIC for the proposed architectures of (a) $2D$, (b) $3D$ and (c) $4D$ FICA.

Fig. 3(d) (by folding Fig. 3(a), (b) and (c)) and Fig. 4(d) (by folding Fig. 4(a), (b) and (c)) respectively.

A. Multiplexed Architecture: CORDIC Reuse for nD FICA

To generalize this for nD multiplexed architecture, only the multiplexers used in Fig. 1(d), 3(d) and 4(d) are reproduced in Fig. 7(a), (b) and (c) respectively. Comparing these figures, it can be noticed that the multiplexer array used for $2D$ architecture (Fig. 7(a)) are also used in $3D$ and $4D$ (shown as "array-1" in Fig. 7(b) and 7(c)). Moreover, due to the inclusion of the *level* information, $3D$ and $4D$ architectures need another array of multiplexers, shown as "array-2" in Fig. 7(b) and (c). However the notable difference between the "array-2" in Fig. 7(b) and (c) is the *type* of multiplexer used. For $3D$ case (Fig. 7(b)), "array-2" comprises of *three* $(2:1)$ multiplexers at both Rotation and Vectoring side. Similarly, for $4D$ (Fig. 7(c)), "array-2" comprises of *three* $(3:1)$ multiplexers at the Vectoring side and *two* $(3:1)$ and *one* $(2:1)$ multiplexer at the Rotation side.Exploiting this similarity, the following can

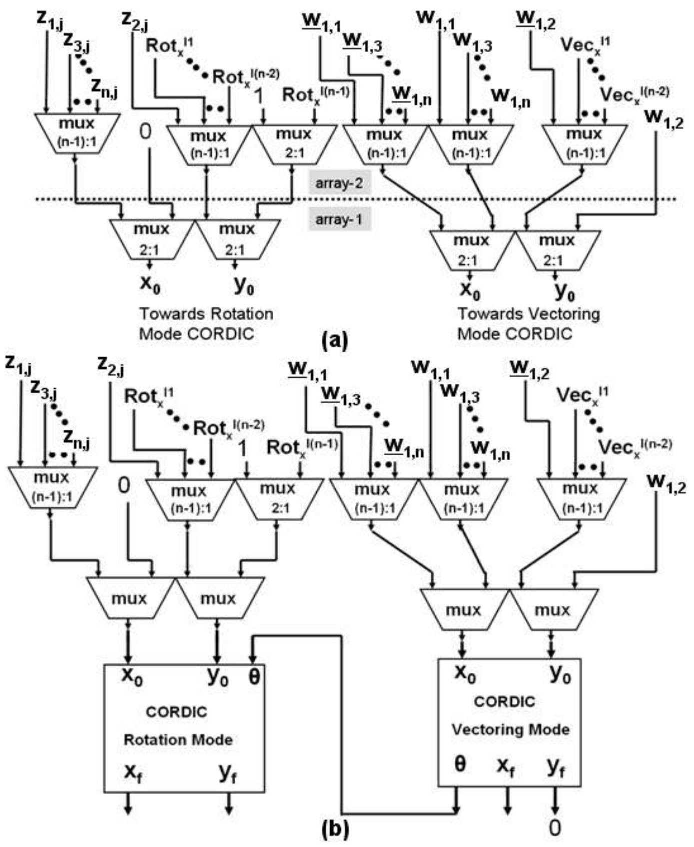

Fig. 8. Generalized (a) multiplexer arrays and (b) architecture of the proposed CORDIC based nD FICA.

be inferred about the multiplexer arrays of the proposed nD architecture:

Multiplexer Array-1: This same as that used for $2D$ case shown in Fig. 7(a).

Multiplexer Array-2: It comprises of *three* $((n-1):1)$ multiplexers at the Vectoring side and *two* $((n-1):1)$ multiplexers and *one* $(2:1)$ multiplexer at the Rotation Side.

These multiplexer arrays for nD case are shown in Fig. 8(a). Fig. 8(b) presents the architecture for the proposed CORDIC based nD FICA algorithm.

B. Architectural Optimization of CORDIC Based nD FICA

All architectures discussed so far has one thing in common - the explicit *angle* information obtained at the primary output of Vectoring mode CORDIC is connected to the θ input of the Rotation mode CORDIC. However architectural optimization is possible by removing the necessity of explicit angle computation in Vectoring mode CORDIC by using "Doubly Pipeline" method [30].

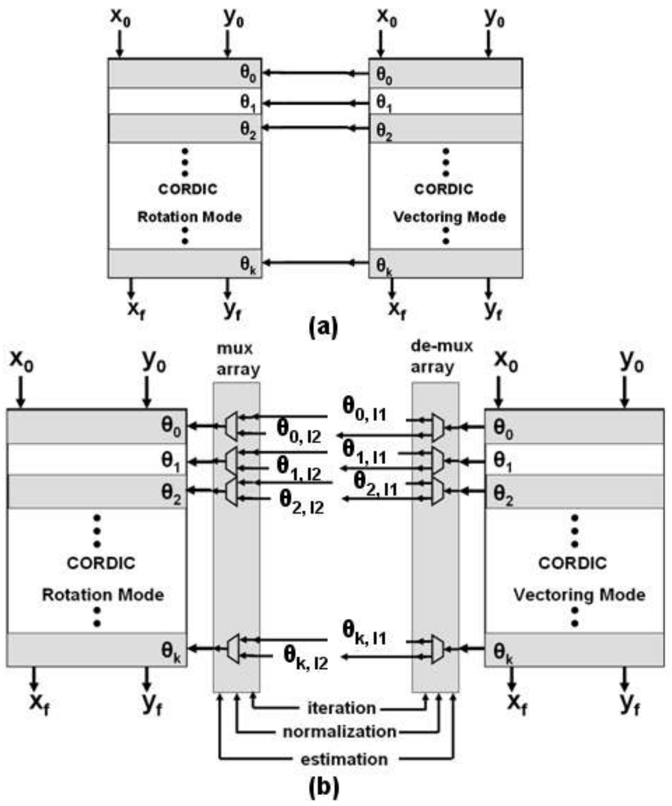

Fig. 9. Doubly Pipelining of the proposed CORDIC based (a) 2-D and (b) $3D$ FICA architectures.

Both Rotation and Vectoring mode CORDIC comprise of finite number of iteration steps and each step modifies x and y values of the initial vector as well as the rotation direction known as "micro angle". Denoting it by θ_i where $i = 0, 1, 2, 3, .., k$ and total number of iteration stages $= (k + 1)$, depending upon the rotation direction (clockwise or anticlockwise), θ_i is considered to be $+1$ or -1. After $(k+1)$-number of such iterations, each θ_i is weighed by 2^{-i} and these results are summed together to produce the accumulated angle θ. When this θ value is fed to the Rotation mode, it is divided into the sequence of same θ_i as obtained in the Vectoring thereby making computation of θ redundant. Thus θ_i generated from the Vectoring mode can be used straightway in the Rotation mode.

Fig. 9(a) shows this *doubly pipeline* method for the proposed CORDIC based $2D$ FICA architecture. However, since the concept *level* comes in for $n \geq 3$, the micro-angle connection from the Vectoring to the Rotation mode CORDIC may not be as straight forward as shown in Fig. 9(a).

Considering $3D$ architecture (Fig. 3) first, since each of its three stages has two *levels* with different set of θ_i, it is essential to use *one* $(1 : 2)$ de-multiplexer and *one* $(2 : 1)$ multiplexer at the θ_i-outputs and

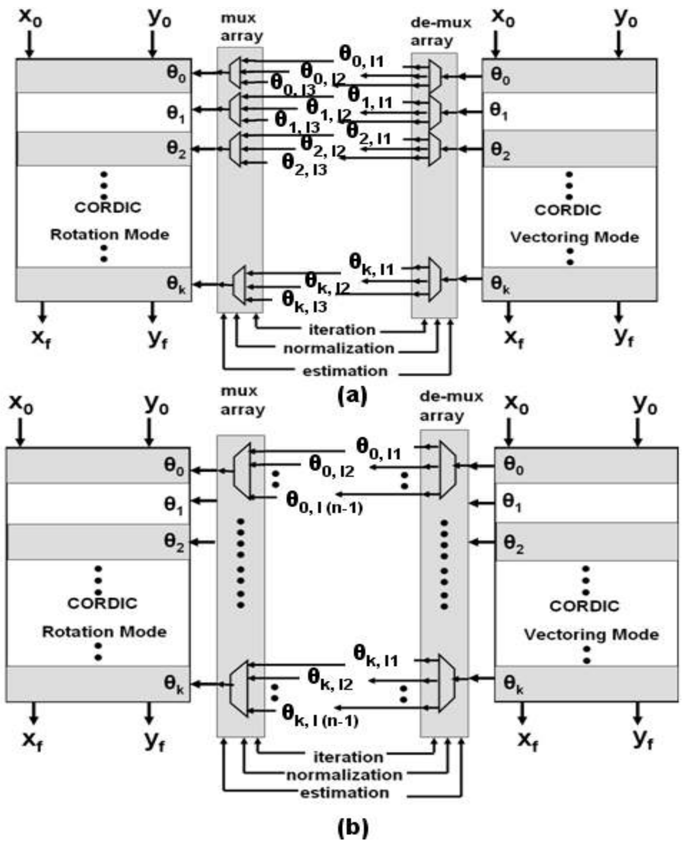

Fig. 10. Doubly Pipelining of the proposed CORDIC based (a) $4D$ and (b) nD FICA architecture.

inputs of the Vectoring and Rotation mode as shown in Fig. 9(b). Similarly, Fig. 10(a) presents CORDIC based $4D$ FICA architecture employing doubly pipeline.

Proceeding the same way for the proposed nD FICA, as shown in Fig. 10(b), *one* $((n-1):1)$ de-multiplexer and *one* $((n-1):1)$ multiplexer are necessary at the θ_i-outputs and inputs of the Vectoring and Rotation mode respectively. Using Fig. 10(b) in Fig. 8(b), proposed nD FICA architecture can further be modified as shown in Fig. 11.

Interested readers may look into [20]-[23] to get an insight of the implementation of CORDIC in silicon/FPGA and also into [24]-[28] to get an idea of quantization error and numerical accuracy of CORDIC.

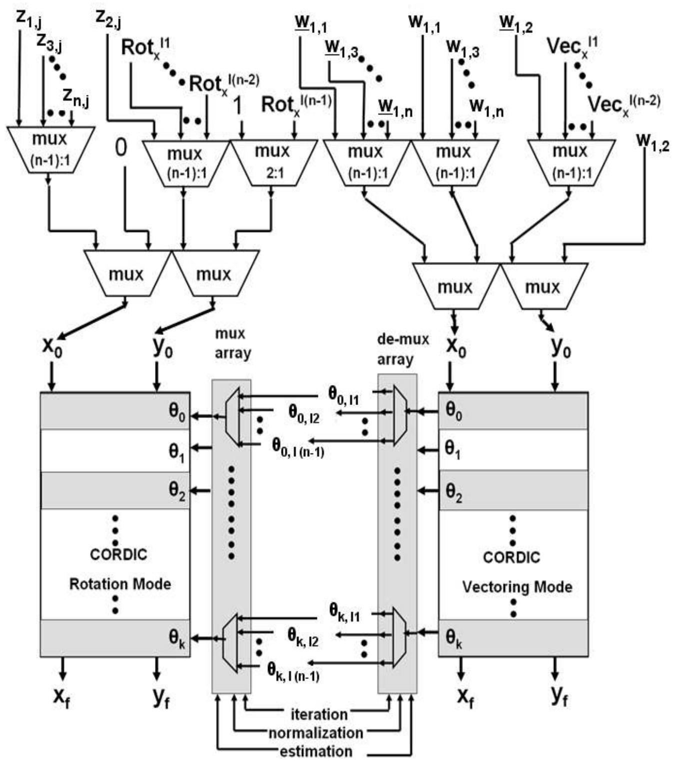

Fig. 11. CORDIC based optimized generic nD FICA architecture using Doubly pipeline.

VII. Hardware Complexity Analysis

A. Important Assumptions

Throughout the hardware complexity analysis we keep a generalized view of frame-length m and word-length b and follow the same procedure used in [13] and [15]. Since FICA is an iterative procedure, we consider only one single iteration because the same hardware resources can be reused for the next iterations.

To provide a comparison on a uniform platform we consider only Ripple Carry Adder (RCA), Conventional Array Multiplier (CAM), Non-restoring Iterative Cellular Square Rooter (SQRT) and Non-restoring Array Divider (NAD) as the means of implementing the arithmetic operations. Considering a b-bit RCA requires b Full Adders (FA) (in a simplified view) [15], $b \times b$ CAM requires $b(b-2)$ FA plus b Half Adders (HA) and b^2 AND gates [15]. Similarly one $b \times b$ NAD consists of $0.5 \times b(3b-1)$ FA and $0.5 \times b(3b-1)$ XOR gates [15] and one b-bit SQRT needs $0.125 \times (b+6)b$ FA and XOR gates [31]. In addition, considering one FA cell requires 24 transistors, one HA cell and one two input XOR gates

TABLE I
SAVING IN TERMS OF ARITHMETIC COMPUTATIONS (MUL- MULTIPLICATION, ADD- ADDITION, SQRT- SQUARE ROOT AND DIV- DIVISION) FOR THE CORDIC BASED PROPOSED $2D$, $3D$ AND $4D$ FICA.

Dim	Iteration		Normalization				Estimation	
	Mul	Add	Mul	Add	SqRt	Div	Mul	Add
$2D$	$2m$	m	2	1	1	2	$2m$	m
$3D$	$3m$	$2m$	3	2	1	3	$3m$	$2m$
$4D$	$4m$	$3m$	4	3	1	4	$4m$	$3m$

consists of 12 transistors and a two input AND gates consists of 6 transistors [15], we can calculate $TC_A = 24b$, $TC_M = 6b(5b-6)$, $TC_D = 18b(3b-1)$ and $TC_{SQ} = 18(b/2+1)(b/2+3)$, where TC_* are the transistor counts for RCA, CAM, NAD and SQRT respectively.

One basic single bit 2 : 1 Transmission Gate Multiplexer comprises of 4 transistors and thus b-bit 2 : 1 multiplexer array has got $TC^{2:1}_{mux} = 4b$ [33]. Any larger multiplexer can be realised using cascaded 2 : 1 multiplexer [33] and thus total transistor count of $n : 1$ multiplexer ($TC^{n:1}_{mux}$) can be expressed in terms of $TC^{2:1}_{mux}$ as: $TC^{n:1}_{mux} = (n-1) * TC^{2:1}_{mux}$. Furthermore, we assume the complexity of a de-multiplexer is same as that of a multiplexer.

TABLE II
TRANSISTOR SAVING AND MULTIPLEXER PENALTY OF THE CORDIC BASED PROPOSED $2D$, $3D$ AND $4D$ FICA. (DPL - DOUBLY PIPELINING)

Dim	Transistor Saving(TS)	Penalty	Mux-Array 1	Mux-Array 2	Mux-DPL
$2D$	$TS_{2D} = (4m+2)TC_M + (2m+1)TC_A + TC_{SQ} + 2TC_D$	P_{2D}	$4TC^{2:1}_{mux}$	-	-
$3D$	$TS_{3D} = (6m+3)TC_M + 2(2m+1)TC_A + TC_{SQ} + 3TC_D$	P_{3D}	$4TC^{2:1}_{mux}$	$5TC^{2:1}_{mux} + TC^{2:1}_{mux}$	$2(k+1)TC^{2:1}_{mux}$
$4D$	$TS_{4D} = (8m+4)TC_M + 3(2m+1)TC_A + TC_{SQ} + 4TC_D$	P_{4D}	$4TC^{2:1}_{mux}$	$5TC^{3:1}_{mux} + TC^{2:1}_{mux}$	$2(k+1)TC^{3:1}_{mux}$

B. Hardware Saving for Proposed nD Architecture

Following the same procedure used in [13], savings in terms of arithmetic operations for $2D$, $3D$ and $4D$ cases for each of the three stages are shown in Table I. Adopting the same approach used in [15] and [13], total saving in terms of arithmetic operations is expressed in terms of Transistor Count (TC) and all these values of TC_* are translated in Transistor Saving (TS) as shown in Table II.

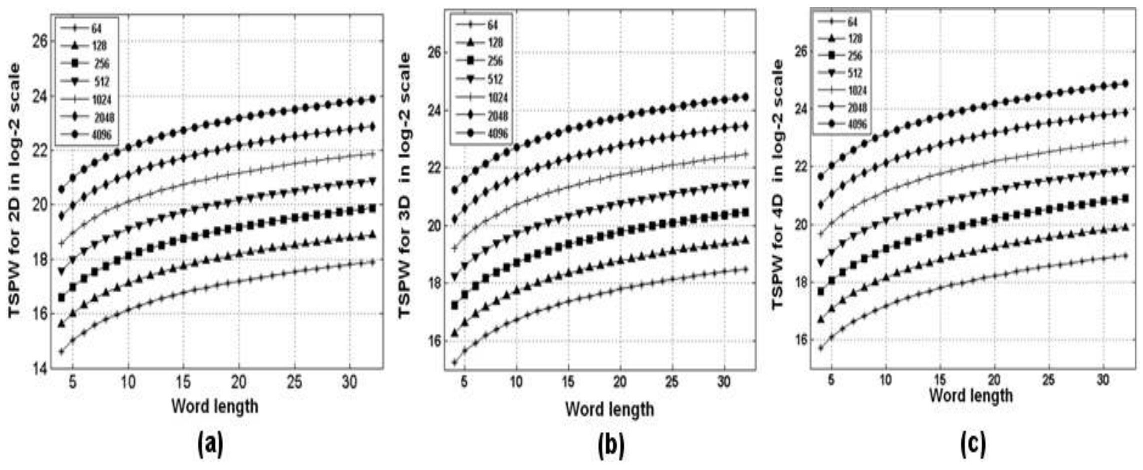

Fig. 12. Variation of Transistor Saving Per Word-length of the proposed algorithm for (a) $2D$, (b) $3D$ and (c) $4D$ with Word-length and frame-length.

Proceeding the same way for the proposed nD FICA overall saving per iteration can be computed and can be translated in terms of TC as follow:

$$TS_{nD} = n(2m+1)TC_M + (n-1)(2m+1)TC_A + TC_{SQ} + nTC_D \tag{51}$$

C. Multiplexer Penalty for nD Architecture

CORDIC reuse introduces multiplexer penalty which are shown in Table II for $2D$, $3D$ and $4D$ cases. Following the trend shown in Table II, overall penalty for the proposed CORDIC based nD FICA architecture (P_{nD}) can be given as:

$$\begin{aligned} P_{nD} &= 4TC_{mux}^{2:1} + 5TC_{mux}^{(n-1):1} + TC_{mux}^{2:1} + 2(k+1)TC_{mux}^{(n-1):1} \\ &= 5TC_{mux}^{2:1} + 5(n-2)TC_{mux}^{2:1} + 2(k+1)(n-2)TC_{mux}^{2:1} \\ &= (5(n-1) + 2(k+1)(n-2))TC_{mux}^{2:1} \end{aligned} \tag{52}$$

D. Effective Hardware Saving for nD Architecture

Effective hardware saving for the proposed nD FICA architecture (ES_{nD}), if there is any, can be given by: $ES_{nD} = TS_{nD} - P_{nD}$. Expressing ES_{nD} in terms of total number of transistors saved and normalizing with respect to b, a metric - Transistor Saving Per Word-length ($TSPW$) can be computed following the approach presented in [32].

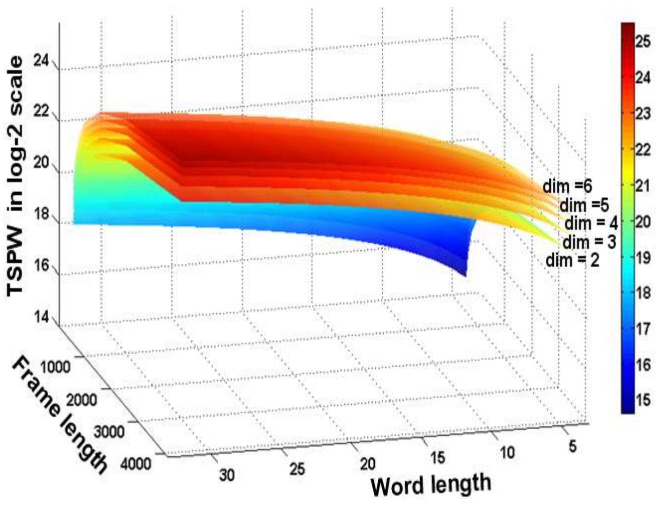

Fig. 13. Comparative variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm with Frame-length and Word-length for $2D$ to $6D$ (denoted by dim = 2 to dim = 6).

Being the function of m and b, Fig. 12(a), (b) and (c) show the variation of $TSPW$ for the Proposed $2D$, $3D$ and $4D$ architectures over the conventional ones and Fig. 13 shows the comparative $TSPW$ variation analysis for the proposed nD FICA architecture for $n = 2$ to 6, with respect to different frame-length ($64 \leq m \leq 4096$) and word-length ($4 \leq b \leq 32$). It can be observed from these figures that $TSPW$ for the proposed CORDIC based FICA is significantly higher than that of the conventional FICA.

However it is well known that reuse of the same hardware unit may have adverse effect on the timing performance of the design which is discussed next.

E. Effect of CORDIC Reuse on Computational Delay

Denoting the delay of b-bit two-operand RCA, b-by-b CAM, b-by-b NAD and b bit SQRT by $2b\triangle$ $(= \tau_a)$, $8b\triangle$ $(= \tau_m)$, $(3b+2)n\triangle$ $(= \tau_d)$ [29] and $(b+1)(2b+3)\triangle$ $(= \tau_{sqrt})$ [31] respectively where $\triangle$ represents the delay of a two-input NAND gate, following the same procedure used in [15], computational delay can be calculated for $2D$, $3D$ and $4D$ cases as shown in Table III. Fig. 1, 3, 4 and 11 and Table I are used to derive the delay expressions of Table III.

Exploiting the similarity of these expressions, computational delay for conventional as well as the proposed nD FICA can be computed for Iteration ($\mathbf{\Gamma}_{it\_conv}$ and $\mathbf{\Gamma}_{it\_prop}$), Normalization ($\mathbf{\Gamma}_{nrm\_conv}$ and $\mathbf{\Gamma}_{nrm\_prop}$) and Component Estimation ($\mathbf{\Gamma}_{est\_conv}$ and $\mathbf{\Gamma}_{est\_prop}$) stages as: $\mathbf{\Gamma}_{it\_conv} = n\tau_m + (n-1)\tau_a$,

TABLE III

COMPUTATIONAL DELAY IN TERMS OF ARITHMETIC OPERATIONS FOR THE PROPOSED AND CONVENTIONAL $2D$, $3D$ AND $4D$ FICA.

Dim	Iteration ($\mathbf{\Gamma}_{it}$)		Normalization ($\mathbf{\Gamma}_{nrm}$)		Estimation ($\mathbf{\Gamma}_{est}$)	
	Conventional	Proposed	Conventional	Proposed	Conventional	Proposed
$2D$	$2\tau_m + \tau_a$	$[(k+1)+1]\tau_a$	$2\tau_m + \tau_a + \tau_{sqrt} + 2\tau_d$	$(k+2)\tau_a$	$2\tau_m + \tau_a$	$(k+1)\tau_a$
$3D$	$3\tau_m + 2\tau_a$	$[2(k+1)+1]\tau_a$	$3\tau_m + 2\tau_a + \tau_{sqrt} + 3\tau_d$	$[(k+2)+2(k+1)]\tau_a$	$3\tau_m + 2\tau_a$	$2(k+1)\tau_a$
$4D$	$4\tau_m + 3\tau_a$	$[3(k+1)+1]\tau_a$	$4\tau_m + 3\tau_a + \tau_{sqrt} + 4\tau_d$	$[(k+2)+4(k+1)]\tau_a$	$4\tau_m + 3\tau_a$	$3(k+1)\tau_a$

$\mathbf{\Gamma}_{it\_prop} = [(n-1)(k+1)+1]\tau_a$, $\mathbf{\Gamma}_{nrm\_conv} = n\tau_m + (n-1)\tau_a + \tau_{sqrt} + n\tau_d$, $\mathbf{\Gamma}_{nrm\_prop} = [(k+2) + 2(n-2)(k+1)]\tau_a$, $\mathbf{\Gamma}_{est\_conv} = n\tau_m + (n-1)\tau_a$, $\mathbf{\Gamma}_{est\_prop} = (n-1)(k+1)\tau_a$.

Approximating $(k+1)$ by b and normalizing with respect to $\triangle$, the normalized delay is computed for Iteration, Normalization and Component Estimation stages and is compared between the proposed and the conventional FICA for $2D$ - $5D$ cases as shown in Fig. 14(a), (b) and (c) respectively. It can be observed from Fig. 14(a) and (c) that for Iteration and Estimation stages, $\mathbf{\Gamma}_{it\_prop}$ and $\mathbf{\Gamma}_{est\_prop}$ are higher than $\mathbf{\Gamma}_{it\_conv}$ and $\mathbf{\Gamma}_{est\_conv}$ respectively. On the other hand, Fig. 14(b) shows that for the Normalization stage, $\mathbf{\Gamma}_{nrm\_prop}$ is less than $\mathbf{\Gamma}_{nrm\_conv}$.

Fig. 15 shows the overall computational delay for $2D$ - $5D$ cases for word-length range of 4 to 32, obtained by adding delay of each stage together. Fig. 15 shows that with the increase of word-length the rate of increase of the computational delay for the proposed algorithm ($\mathbf{\Gamma}_{prop}$) grows faster than that of the conventional one ($\mathbf{\Gamma}_{conv}$). However, for $2D$, $\mathbf{\Gamma}_{prop}$ is significantly less than $\mathbf{\Gamma}_{conv}$, for $3D$ cases these two are same when word-length reaches 30. With the increase of $n \geq 4$, $\mathbf{\Gamma}_{prop}$ starts exceeding $\mathbf{\Gamma}_{conv}$ for word-length < 16. However, such negative effects will have minimal effect on the envisaged resource constrained applications such as mobile healthcare and wireless sensor networks (see Section I) where low power light-weight design is of prime concern, not the computational delay because most of the vital signs monitored in such environment, are sampled at very low frequency (≤ 1 KHz) [34] and thus a system with clock speed higher than this may ensure achieving an expected throughput.

VIII. CONCLUSION

In this paper we introduced co-ordinate rotation concept into FICA and based on this concept we proposed a generalized recursive nD FICA algorithm and architecture using $2D$ as the fundamental core. The proposed algorithm has been proved very effective over the conventional one in terms of

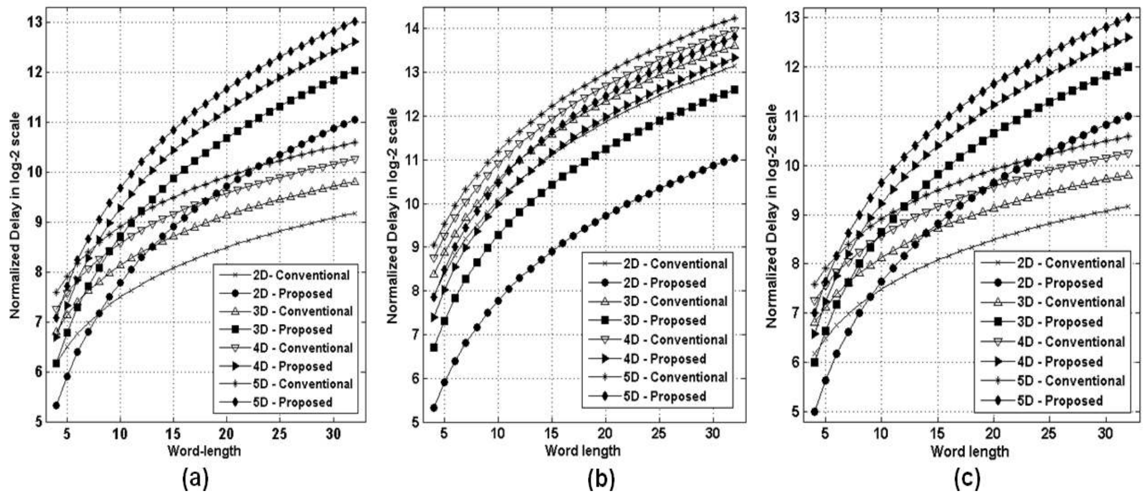

Fig. 14. Comparison of the normalized computational delay of (a) Iteration, (b) Normalization and (c) Component Estimation stages between the proposed and conventional algorithm for different dimensions with respect to word-length.

hardware complexity achieved by reusing the CORDIC unit in both preprocessing as well as in the FICA Update step. It has also been shown that further hardware simplification is obtained because of the recursive nature of the proposed algorithm allowing implementation of the nD architecture using 2D. Moreover, to obtain better architectural performance, we have identified and eliminated some redundant computations and presented further optimized architectures for the proposed CORDIC based nD FICA algorithm. To the best of our knowledge the proposed algorithms and architectures are the first of its kind in the field of low-complexity design of nD FICA and have the potential to open up new application domain of CORDIC. Part of our future research includes the investigation of the relationship among the word-length, FICA dimension and numerical accuracy of the proposed algorithm.

REFERENCES

[1] 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.

[2] 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.

[3] S. Choi, A. Cichocki, H. M. Park and S. Y. Lee, "Blind Source Separation and Independent Component Analysis", *Neural Information Processing- Letters and Reviews*, vol. 6, no. 1, January 2005.

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

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

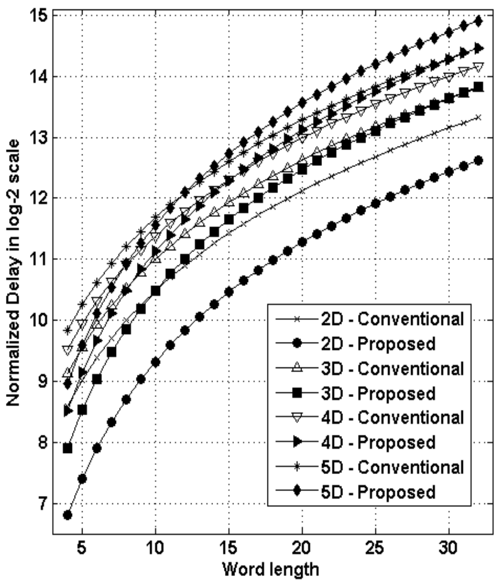

Fig. 15. Comparison of the overall normalized computational delay between the proposed and conventional algorithm combining all three stages together.

[6] R. P. Brent, F. T. Luk and C. V. Loan, “Computation of the Singular Value Decomposition Using Mesh Connected Processors”, *Journal of VLSI and Computer Systems*, vol. 1, no. 3, pp. 242-270, 1985.

[7] J. Götze and G. J. Hekstra, “An Algorithm and Architecture based on Orthonormal μ-rotations for Computing the Symmetric EVD”, *Integration, The VLSI Journal*, vol. 20, pp. 21-39, 1995.

[8] J. R. Cavallaro and F. T. Luk, “CORDIC Arithmetic for an SVD Processor”, *Journal of Parallel and Distributed Computing*, vol. 5, pp. 271-290, 1988.

[9] J. Götze, S. Paul and M. Sauer, “An Efficient Jacobi-like Algorithm for Parallel Eigenvalue Computation”, *IEEE Trans. Computers*, vol. 42, no. 9, September, 1993.

[10] S. F. Hsiao and J. M. Delosme, “Parallel Singular Value Decomposition of Complex Matrices Using Multidimensional CORDIC Algorithms”, *IEEE Trans. Signal Processing*, vol. 44, no. 3, March, 1996.

[11] I. Bravo, P. Jiménez, M. Mazo, J. L. Lázaro and A. Gardel, “Implementation in FPGAs of Jacobi method to Solve the Eigenvalue and Eigenvector Problem”, *International Conf. Field Prog. Logic and Applications*, 2006.

[12] I. Bravo et. al., “Novel HW Architecture Based on FPGAs Oriented to Solve the Eigen Problem”, *IEEE Trans. VLSI Systems*, vol. 16, no. 12, December, 2008.

[13] A. Acharyya, K. Maharatna and B. M. Al-Hashimi, “Co-ordinate Rotation Based Low Complexity 2D FastICA Algorithm and Architecture”, *IEEE International Conference on Green Circuits and Systems*, pp. 60-64, Shanghai, China, June 21-23, 2010.

[14] 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.

[15] A. Acharyya, K. Maharatna and B. M. Al-Hashimi, “Hardware Reduction Methodology for 2-Dimensional Kurtotic FastICA Based on Algorithmic Analysis and Architectural Symmetry”, *IEEE Workshop on Signal Processing Systems*, pp. 69-74, October, 2009.

[16] J. E. Volder, “The CORDIC Trigonometric Computing Technique”, *IRE Trans. Electron. Comput.*, EC-8, pp. 330-334, 1959.

[17] J. S. Walther, “A Unified Algorithm for Elementary Functions”, *Spring Joint Computer Conference*, pp. 379-385, 1971.

[18] Y. H. Hu, “CORDIC Based VLSI Architecture for Digital Signal Processing”, *IEEE Signal Processing Mag.*, pp. 16-35, July, 1992.

[19] A. Mansour, M. Kawamoto and N. Ohnishi, “A Survey of The Performance Indexes of ICA Algorithms”, *Proc. IASTED Int. Conf. Modeling, Identification and Control*, pp. 660-666, Austria, February, 2002.

[20] R. Andraka, “A Survey of CORDIC algorithms for FPGA based computers”, *Proc. of the ACM/SIGDA* 6^{th} *Int. Symp. on Field Programmable Gate Arrays*, pp. 191-200, USA, 1998.

[21] T. Vladimirova and H. Tiggeler, “FPGA Implementation of Sine and Cosine Generators Using CORDIC Algorithm”, *Military and Aerospace Applications of Programmable Devices and technologies Conference*, USA, September, 1999.

[22] O. Mencer, L. S$\acute{e}$m$\acute{e}$ria, M. Morf and J. M. Delosme, “Application of Reconfigurable CORDIC Architectures”, *Journal of VLSI Signal Processing Systems*, vol. 24, pp. 211-221, 2000.

[23] S. Kadam, M. Soderstrand and L. Johnson, “CORDIC Implementation of Digital Heterodyne Filter in VLSI”, 35^{th} *Asilomar Conference on Signals, Systems and Computers*, pp. 529-532, November, 2001.

[24] Y. H. Hu, “The Quantization Effects of the CORDIC Algorithm”, *IEEE Trans. Signal Processing*, pp. 834-844, vol. 40, no. 4, April, 1992.

[25] K. Kota and J. Cavallaro, “Numerical Accuracy and Hardware Tradeoffs for CORDIC Arithmetic for Special-Purpose Processors”, *IEEE Trans. Computers*, pp. 769-779, vol. 42, no. 7, July, 1993.

[26] X. Hu and S. C. Bass, “A Neglected Error Source in the CORDIC Algorithm”, *Int. Symposium on Circuits and Systems*, pp. 766-769, 1993.

[27] E. Antelo, J. D. Bruguera, T. Lang and E. L. Zapata, “Error Analysis and Reduction for Angle Calculation Using the CORDIC Algorithm”, *IEEE Trans. Computers*, pp. 1264-1271, vol. 46, no. 11, November, 1997.

[28] S. Y. Park and N. I. Cho, “Fixed-Point Error Analysis of CORDIC Processor Based on the Variance Propagation Formula”, *IEEE Trans. Circuits and Systems-I: Regular Papers*, pp. 573-584, vol. 51, no. 3, March, 2004.

[29] K. Hwang, “Computer Arithmetic: Principles, Architecture and Design”, *Wiley Publishing*, chapter 11, 1979.

[30] T. Y. Sung, Y. H. Hu and H. J. Yu, “Doubly Pipelined CORDIC Array for Digital Signal Processing Algorithms”, *ICASSP*, pp. 1169-1172, 1986.

[31] J. C. Majithia, “Pipeline Array for Square-Root Extraction”, *Electronics Letters*, vol. 9, no. 1, pp. 4-5, 1973.

[32] A. Acharyya, K. Maharatna, B. M. Al-Hashimi and S. R. Gunn, “Memory Reduction Methodology for Distributed Arithmetic Based DWT/IDWT Exploiting Data Symmetry”, *IEEE Trans. Circuits and Systems-II: Express Briefs*, vol. 56, no. 4, pp. 285-289, April 2009.

[33] N. H. E. Weste and D. Harris,“CMOS VLSI Design: A Circuits and Systems Perspective”, *Pearson-Addison Wesley*, chap-1, Third International Edition, 2005.

[34] U. Varshney, “Pervasive Healthcare and Wireless Health Monitoring”, *Mobile Networks and Applications, Springer*, pp. 113-127, vol. 12, no. 2, 2007.