Low Complexity CORDIC based methodologies for BSS

Adapa Bhagyaraja

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Indian Institute of Technology Hyderabad

Department of Electrical Engineering

June 2016

Declaration

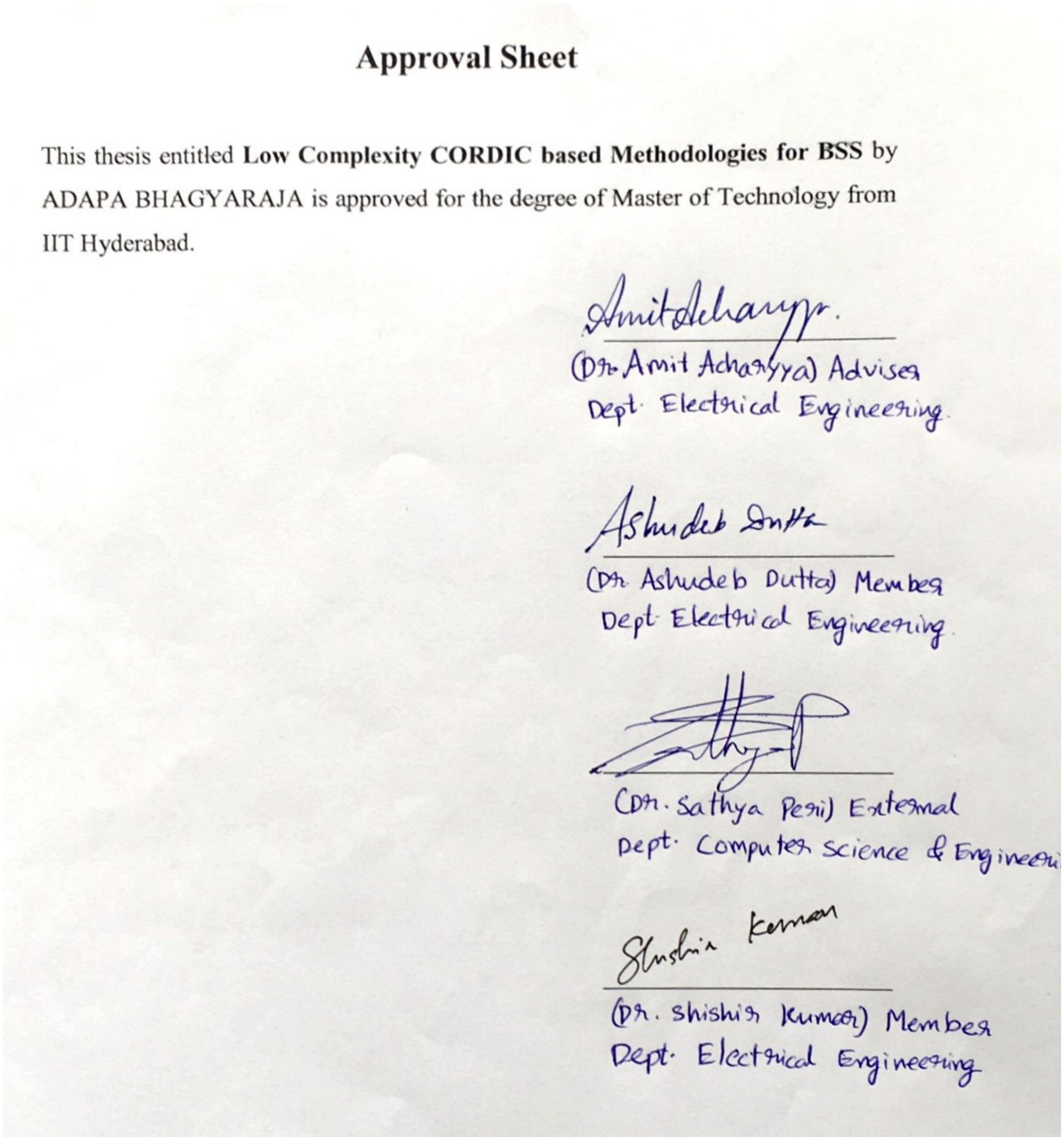
I declare that this written submission represents my ideas in my own words, and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

A. Bhagyaraja

(Signature)

ADAPA BHAGYARAJA

EE13M1019



Approval Sheet

This thesis entitled **Low Complexity CORDIC based Methodologies for BSS** by ADAPA BHAGYARAJA is approved for the degree of Master of Technology from IIT Hyderabad.

Amit Acharyya.

(Dr. Amit Acharyya) Adviser
Dept. Electrical Engineering.

Ashudeb Dutta

(Dr. Ashudeb Dutta) Member
Dept. Electrical Engineering.

[illegible]

(Dr. Sathya Peri) External
Dept. Computer Science & Engineeri

Shishir Kumar

(Dr. Shishir Kumar) Member
Dept. Electrical Engineering

Acknowledgements

I would like to acknowledge my guide Dr. Amit Acharyya for his guidance and support in my M.tech thesis. I would like to acknowledge Dr. Dwaipayan Biswas(University of Southampton) for his valuable inputs. Also I would like to acknowledge my faculties Dr. Asudeb Dutta, Dr. Shiv Govind and Dr. Shiva Rama Krishna who taught us M.Tech courses so that I got the knowledge to accomplice my M.Tech Project. I would also like to acknowledge Swati Bhardwaj and Shashank R for their motivation and discussions. I would like to thank all participants for their consent and kind cooperation in these experiments. This work was supported by the Information and Communication Technologies Theme of the European Union Seventh Framework Programme, under the project name "StrokeBack: Telemedicine system empowering stroke patients to fight back" (grant number 288692) and DEITY India for partly supporting our work, under the "IOT for Smarter Healthcare" under Grant No: 13(7)/2012-CC&BT, Dated 25th Feb, 2013.

Dedication

To

My parents, Friends and Dr. Amit Acharyya

Abstract

Our Project aim is to develop a real time chip to process the sensor signals and separating the source signals, which is used in Health care and Wireless Sensor Networks (WSN) like Autism, ECG and disease detection. Autism is a disease which affects the child mental behavior. So If we analyze the signals form the brain so we can observe the how effectively the disease is cured. So to analyze the Autism we need EEG signals from almost 128 Leads from the scalp of child. In cardiovascular diseases, patient's ECG signal is monitored, which is a combination of ECG, Artefacts and noise. Desired ECG signal is separation from mixed signals is vital for accurate analysis of disease. All these operations has to be done in real-time scenario with patient being monitored all the time. So, there is a need of low-complexity, low-power architecture for this signal separation is needed. Here, we proposed a COordinate Rotation DIgital Computer (CORDIC) based engine to separate mixed signals. The proposed algorithm can merge the two key steps of conventional FastICApreprocessing and update and is therefore capable of reducing the hardware complexity of the conventional FastICA significantly. 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 architecture. Architecture has been further improved such that same hardware can be used for any number of input signals by making it reconfigurable. Further, a hybrid architecture has been proposed by combining FastICA with cross product. Computational time reduced by this hybrid implementation which makes it feasible for real-time scenario.

We proposed a low-complexity architecture for inferring the unique number of arm movements performed in an out-of-laboratory environment using wrist worn accelerometers. We propose that this framework can be used as a clinical tool to assess rehabilitation progress in neurodegenerative pathologies tracking the number of unique movements performed by patients with their impaired arm. For this investigation, we consider four healthy and stroke survivors performing an archetypal activity of daily living making-cup-of tea-coffee having repeated occurrences of three elementary arm-movements(reach-retrieve, lift-cup-to-mouth, rotate-arm). We use an unsupervised learning approach: k-means clustering and estimate the optimal number of clusters to infer the unique number of movements from the kinematic data. This is representative of an ambient assisted living scenario, where the arm usage of stroke survivors in activities of daily living can be determined to track rehabilitation progress. Furthermore, a low-complexity architecture using CORDIC is proposed which can enable data processing on the sensor node itself thereby reducing energy expenditure incurred in data transmission. This will help to elongate the battery life of the sensors especially for scenarios where adequate computing facilities are not available. The design synthesized in STMicroelectronics 130nm technology occupied core area of 0.61mm2 and consumed power of 9.21mW at 1.2V @ 100-MHz frequency, making it applicable for real-time high-speed operations within a node of Wireless Sensor Networks (WSN). Our experimental results show that three arm movements were detected for the healthy subjects whereas for the stroke survivors the number of movements reported were on the higher side for the experimental duration, reflecting the variability inherent within the movement profiles.

Contents

Chapter 1

Introduction

Emerging applications in the areas of Wireless Sensor Networks (WSN), remote health monitoring and pervasive computing require sophisticated signal processing algorithms and corresponding architectural implementation with limited area and low power consumption. Sensors used for such resource constrained applications capture the signals and transmit to some central node on continuous basis. Most of the processing are performed within the central node and it is commonly assumed that this central node has continuous power supply. However since the sensor nodes are mostly run by battery back-up and their radio front end consumes significantly high power, these nodes run out of power soon in such approach. To overcome this problem, researchers have been working on developing decorrelation based approaches and intelligent compression techniques to reduce the redundancy in the data and also to inhibit the traffic load over the communication channel. However, apart from this, it can also be envisaged to have on-sensor resource constrained signal processing algorithms and architectures eliminating the need of continuous data transmission. Since the signals captured by the sensors in the above mentioned emerging application environment are corrupted with noise and other signals, it is important to explore those signal processing algorithms which can help denoising and signal separation. This thesis investigates such algorithms and their corresponding architectures considering the fundamental constraints of resources in terms of power, area and hardware complexity.

Human activity recognition in natural settings has gained prominence in recent years used in remote health monitoring systems to assess patient mobility. The advent of mobile and ubiquitous computing facilities using low-cost inertial sensors, radio-frequency identification (RFID) and fusion of vision-based and inertial sensor based approaches [22] have helped continuous subject monitoring. The last two approaches have the bottleneck of primarily being restricted to indoor activities within a defined region and require an un-hindered surveillance of the vision system [23]. Furthermore, the use of high complexity image processing algorithms can result in slower analysis which can be particularly challenging if real-time information analysis is required [24]. With the shift in focus of the research community towards monitoring human activities performed in daily life which is a more natural indicator of the subjects' involvement as compared to monitoring only during a prescribed exercise/training phase, the use of body-worn inertial sensors have gained prominence over other aproaches.

Advancements in wireless sensor networks (WSN) and information communication technology have played a key role in the miniaturization of low-cost, body-worn sensors enabling cost efficient patient monitoring and rehabilitation within the home environment [25]. This is particularly relevant when considering the increasing healthcare costs of an ageing population, especially those related to the treatment of chronic arthritis, cardiovascular diseases or neurodegenerative diseases, and the need to reduce the amount of time the patient spends at the clinic.

Recognition of activities using body-worn sensor data in an unconstrained daily living scenario is particularly challenging owing to the considerable amount of variability inherent in movement patterns of each subject without any manual intervention or a-priori knowledge [26]. The huge amount of kinematic data collected over long-duration which is commonly unlabelled or without any information on the corresponding activity, poses a challenge with respect to their validation post any classification that is carried out on them. Moreover, the fundamental requirement for such pervasive sensors used in real-time multimodal data acquisition and analysis, is to prolong the battery life of the sensors by compensating the significant energy required at the radio front-end of the sensors for continuous data transmission [27]. These two challenges - firstly on the algorithmic level and secondly at the sensor level can be solved by selecting an optimal activity detection algorithm which can be carried out at the sensor node itself yielding energy efficient solutions as compared to conventional remote monitoring approaches based on continuous transmission of vital sign data. Hence, from the long-term system operation perspective, when implementing a wireless body area network that is comprised of heterogeneous sensors capturing physiological data, it is imperative to have a low-complexity algorithm-to-architecture implementation.

In principle there are three steps for activity recognition using inertial sensors: (1) data capture by appropriate sensor; (2) segmentation of the captured data to identify the beginning and end of an activity; and (3) recognition of the activity using appropriate classification techniques [23]. Continuous monitoring of activities in an unconstrained scenario involves data segmentation and activity recognition which are in practice interrelated but are individually two separate research problems owing to the possible qualitative non-uniqueness of an activity pattern exhibited by an individual subject and due to inter-person variability. In the research reported here we concentrate only on the activity recognition part as a proof-of-concept methodology.

In this thesis, we look into the domain of activity monitoring in daily life involving a subject population with body-worn inertial sensors for long durations. As a particular case study we consider the field of stroke rehabilitation, wherein a patient wears a wrist-based tri-axial accelerometers, collecting data aimed at monitoring arm movements performed by the patient. Impairment of the arm is a common post-stroke phenomenon and as part of the prescribed rehabilitation protocol, clinicians are particularly interested on whether the patients are using their impaired arm in daily life following their initial rehabilitation within the clinic. Moreover, recognizing the number of distinct activities performed with the impaired arm would also help to gauge the qualitative involvement of the patient in their daily life. Hence, we perform a systematic exploration to recognize the distinct arm movements performed by the upper limb which are generally associated with activities of daily living, using data collected from a wrist-worn, wireless tri-axial accelerometers. The exploration was carried on the data collected from four healthy subjects and four stroke survivors wearing a wrist-worn inertial sensor module as they performed multiple trials of an archetypal activity of daily living *'making-a-cup-of-tea'* involving multiple occurrences of three fundamental movements - (1) reach out and retrieve object, (2) lift cup to mouth and (3) performing pouring/(un)locking action, all of which involve rotations of the forearm about various axes.

Chapter 2

Reconfigurable FastICA

In Chapter 1 it has been mentioned that in the emerging fields such as person-centric remote health monitoring, Wireless sensor networks, separation of signals from a set of mixed sensory data is an important requirement [5], [6]. It has also been pointed out that BSS can have potential application in such fields where without any or much knowledge of the mixed signals or mixing channels, retrieval of source signals is the fundamental goal [3]. It has already been outlined Chapter 1 that ICA is the most commonly used statistical technique for solving BSS problem [3] and among the existing ICA algorithms, FastICA is popular because of its higher convergence speed and accuracy [1], [2]. Being efficient from the algorithmic point of view, FastICA can be deployed in the fore-mentioned applications.

However these emerging applications are limited by the fundamental constraints of resource such as power and area. In most of the cases the devices used for such applications are battery powered and are meant to be tiny and unobtrusive which necessitate the development of reduced complexity low power signal processing techniques. Although algorithmically effective, the computationally intensive nature of FastICA makes its direct mapping into architecture 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 at the same time from its architectural perspective. FastICA algorithm consists of two basic steps - Preprocessing and the main Iterative step [2]. Preprocessing step is composed of two further steps - centering and whitening [2]. Centering requires add and accumulation unit whereas whitening needs computation of eigen values and the corresponding eigen vectors of the covariance matrix formed using the mixed sensory data set [2]. Coordinate Rotation Digital Computer (CORDIC) is the widely used technique for implementing Eigen Value Decomposition (EVD) in hardware [4] . The main FastICA Iterative steps consists of costly arithmetic operations involving division, square root evaluation and multiplications. Direct implementation of these operations increases the hardware complexity and contribute to high power consumption.

In biomedical, Wireless Sensor Network (WSN) applications FastICA is used to separate mixed signals. Different applications needed different architecture as number of inputs are differ based on applications, viz ECG (n=3),EEG(n=128). There was an attempt in [**?**], [**?**] to use the same processing architecture for ECG and EEG. Multiple ICA engines are required in those cases. So, there is a need for a reconfigurable architecture which can dynamically work for different number of signals. So we proposed an algorithm, based on nD-FICA [**?**], for Reconfigurable FastICA which can dynamically works for different number of signals for different cases.

In this chapter the aim is to make an attempt to merge the whitening stage with the FastICA Iterative step so that the same hardware unit can be reused for both of these stages. And make it Reconfigurable so that same architecture separate any number of signals (any n). Since, CORDIC is popularly known technique for implementing EVD, the research goal in this chapter is to explore how, if at all possible, the concept of co-ordinate rotation can be applied in this context for implementing the FastICA Iterative step in a generalized sense.In this chapter, firstly the link between the CORDIC and FastICA is identified considering the nD case where n 2 and then Co-ordinate Rotation based low complexity nD FastICA algorithm and architecture are proposed. Subsequently the hardware complexity analysis is carried out and necessary functionality validation of the algorithm is provided.

2.1 ND-FastICA

The main objective of FastICA is to find the N-Estimator vectors of length N by processing N signals. The algorithm to find the estimator vectors is proposed by Aapo Hyvrinen in [**?**].

The estimator vector 'w' can be calculated from X_w by using the following equation based on [**?**],

$$w(:,i)^{p+1} = \left(X_w \times \left(\left(X_w^T \times \underline{w(:,i)^p}\right).\hat{}3\right)\right)/L - 3 \times \underline{w(:,i)^p} \tag{2.1}$$

$$\underline{w(:,i)} = w(:,i)/norm\left(w(:,i)\right) \tag{2.2}$$

Where w is Estimator matrix of order $[N \times N]$ and X_w is Whitened Matrix of order $[n \times L]$ i.e. Whitening matrix has N- Whitened signals of each frame length is L. And $i = 1, 2, \ldots, N$.

We can write the 2.1 as follows,

$$\begin{bmatrix} w_{1,i}^{p+1} \\ w_{2,i}^{p+1} \\ \vdots \\ w_{N,i}^{p+1} \end{bmatrix} = \begin{bmatrix} E[z_{1,j}\{z_{1,j}\underline{w}_{1,i}^p + z_{2,j}\underline{w}_{2,i}^p + \ldots + z_{N,j}\underline{w}_{N,i}^p\}^3] \\ E[z_{2,j}\{z_{1,j}\underline{w}_{1,i}^p + z_{2,j}\underline{w}_{2,i}^p + \ldots + z_{N,j}\underline{w}_{N,i}^p\}^3] \\ \vdots \\ E[z_{N,j}\{z_{1,j}\underline{w}_{1,i}^p + z_{2,j}\underline{w}_{2,i}^p + \ldots + z_{N,j}\underline{w}_{N,i}^p\}^3] \end{bmatrix} - 3 \times \begin{bmatrix} \underline{w}_{1,i}^p \\ \underline{w}_{2,i}^p \\ \vdots \\ \underline{w}_{N,i}^p \end{bmatrix} \tag{2.3}$$

Where $j = 1, 2, \ldots, L$.

Now we can write 2.10 as follows,

$$\begin{bmatrix} w_{1,i}^{p+1} \\ w_{2,i}^{p+1} \\ \vdots \\ w_{N,i}^{p+1} \end{bmatrix} = \begin{bmatrix} E[z_{1,j}\{G_{ND}\}^3] \\ E[z_{2,j}\{G_{ND}\}^3] \\ \vdots \\ E[z_{N,j}\{G_{ND}\}^3] \end{bmatrix} - 3 \times \begin{bmatrix} \underline{w}_{1,i}^{p} \\ \underline{w}_{2,i}^{p} \\ \vdots \\ \underline{w}_{N,i}^{p} \end{bmatrix} \tag{2.4}$$

Where G_{ND} is column vector of length L. So

$$G_{ND}(j) = z_{1,j}\underline{w}_{1,i}^{p} + z_{2,j}\underline{w}_{2,i}^{p} + \ldots + z_{N,j}\underline{w}_{N,i}^{p} \tag{2.5}$$

For $j = 1, 2, \ldots, L$. And from the 2.2 we can write it as,

$$\underline{w_{i,k}} = \frac{w_{i,k}}{\sqrt{w_{1,k}^2 + w_{2,k}^2 + \ldots + w_{N,k}^2}} \tag{2.6}$$

For $k = 1, 2, \ldots, N$. So our challenge is to design an architecture which is reconfigurable so that for different values of N and L it can solve the equations 2.5 and 2.6. But in [?] Amit Acharyya et. al had proposed a static architecture based on CORDIC which is fixed for a chip, So we can not use it for reconfigurable applications. Hence we are proposing ND-FastICA based on CORDIC which is reconfigurable so that we can use it in our ND-SCICA problem. So from [?] we can write the equations 2.5 using CORDIC as follows,

$$G_{ND}(j) = Rot_x^{N-1}(z_{N,j}, Rot_x^{N-2}(z_{N-1,j}, \ldots, Rot_x^{1}(z_{2,j}, z_{1,j}, \theta_1,) \ldots, \theta_{N-2}), \theta_{N-1}) \tag{2.7}$$

Here $j = 1, 2, \ldots, L$ where

$$\begin{aligned}
\theta_1 &= Vec_\theta^1(w_{2,i}, w_{1,i}) \\
\theta_r &= Vec_\theta^r(w_{r+1,i}, Vec_x^{r-1}(w_{r,i}, Vec_x^{r-2}(w_{r-1,i}, \ldots, Vec_x^1(w_{2,i}, w_{1,i})))) \qquad (2.8)
\end{aligned}$$

Here $r = 2, 3, \ldots, N-1$ and For $i = 1, 2, \ldots, N$.

Similarly we can calculate 2.6 using cordic as follows,

$$\begin{aligned}
\underline{w}_{1,k} &= Rot_x^{N-1}(0, Rot_x^{N-2}(0, \ldots, Rot_x^1(0, 1, \theta_{N-1}), \ldots, \theta_2), \theta_1) \\
\underline{w}_{m,k} &= Rot_y^{N-m+1}(0, Rot_x^{N-m}(0, \ldots, Rot_x^1(0, 1, \theta_{N-1}), \ldots, \theta_2), \theta_1) \qquad (2.9)
\end{aligned}$$

Here $m = 2, 3, \ldots, N$. And for θ terms we can get from 2.8.

From 2.7, 2.8 and 2.9 we can observe that to get the i^th estimate vector $w\{:, i\}$ we have to follow these steps:

Step-1. Take N random values for the vector $w(:, i)$.

Step-2. Find the $N-1$ θ terms for the vector taken in step-1 (for first iteration) or from the step-6 (for second iteration onwards). i.e. we have to use VectorMode Cordic for $N-1$ times.

Step-3. Find the Normalized vector $\underline{w(:, i)}$ by using θ terms from step-2. i.e we have to use RotationMode Cordic for $N-1$ times.

Step-4. Find the vector G_{ND} by using whitened matrix X_w of order $[N \times L]$ and the θ terms from step-2. i.e we have to use RotationMode for $N-1 \times L$.

Step-5. We have to use equation 2.11 using G_{ND} vector from step-4 and $\underline{w(:, i)}$ from step-3. Thus we will get maximum Kurtosis stimator vector $w(:, i)$.

Step-6. We have to check the estimator vector $w(:,i)$ in step-6 with the vector used in previous iteration. If both are in same direction, i.e. angle between them is zero, otherwise goto step-2.

2.2 Architecture Mapping with CORDIC

The main processing steps of the FastICA algorithm are Iteration, Normalization and Estimation. In the subsequent sections, how CORDIC can be mapped to compute above steps is discussed, considering the case of n=4.

2.2.1 Iteration

Iteration is first step of processing and can be expressed as

We can write the 2.1 as follows,

$$\begin{bmatrix} w_{1,i}^{p+1} \\ w_{2,i}^{p+1} \\ w_{3,i}^{p+1} \\ w_{4,i}^{p+1} \end{bmatrix} = \begin{bmatrix} E[z_{1,j}\{z_{1,j}\underline{w}_{1,i}^{p} + z_{2,j}\underline{w}_{2,i}^{p} + z_{3,j}\underline{w}_{3,i}^{p} + z_{4,j}\underline{w}_{4,i}^{p}\}^3] \\ E[z_{2,j}\{z_{1,j}\underline{w}_{1,i}^{p} + z_{2,j}\underline{w}_{2,i}^{p} + z_{3,j}\underline{w}_{3,i}^{p} + z_{4,j}\underline{w}_{4,i}^{p}\}^3] \\ E[z_{3,j}\{z_{1,j}\underline{w}_{1,i}^{p} + z_{2,j}\underline{w}_{2,i}^{p} + z_{3,j}\underline{w}_{3,i}^{p} + z_{4,j}\underline{w}_{4,i}^{p}\}^3] \\ E[z_{4,j}\{z_{1,j}\underline{w}_{1,i}^{p} + z_{2,j}\underline{w}_{2,i}^{p} + z_{3,j}\underline{w}_{3,i}^{p} + z_{4,j}\underline{w}_{4,i}^{p}\}^3] \end{bmatrix} - 3 \times \begin{bmatrix} \underline{w}_{1,i}^{p} \\ \underline{w}_{2,i}^{p} \\ \underline{w}_{3,i}^{p} \\ \underline{w}_{4,i}^{p} \end{bmatrix} \tag{2.10}$$

Where $j = 1, 2, \ldots, L$.

Now we can write 2.10 as follows,

$$\begin{bmatrix} w_{1,i}^{p+1} \\ w_{2,i}^{p+1} \\ w_{3,i}^{p+1} \\ w_{4,i}^{p+1} \end{bmatrix} = \begin{bmatrix} E[z_{1,j}\{G_{4D}\}^3] \\ E[z_{2,j}\{G_{4D}\}^3] \\ E[z_{3,j}\{G_{4D}\}^3] \\ E[z_{4,j}\{G_{4D}\}^3] \end{bmatrix} - 3 \times \begin{bmatrix} \underline{w}_{1,i}^{p} \\ \underline{w}_{2,i}^{p} \\ \underline{w}_{3,i}^{p} \\ \underline{w}_{4,i}^{p} \end{bmatrix} \tag{2.11}$$

where

$$
\begin{aligned}
G_{4D} &= Rot_x(z_{4,j}, R^{3D}_{x,j}, Vec_\theta(\underline{w}^p_{1,4}, V^{3D}_x)) = R^{4D}_{x,j}\\
G_{3D} &= Rot_x(z_{3,j}, R^{2D}_{x,j}, Vec_\theta(\underline{w}^p_{1,3}, V^{2D}_x)) = R^{3D}_{x,j}\\
R^{2D}_{x,j} &= Rot_x(z_{2,j}, z_{1,j}, Vec_\theta(\underline{w}^p_{1,2}, \underline{w}^p_{1,1}))\\
V^{4D}_x &= Vec_x(\underline{w}^p_{1,4}, V^{3D}_x)\\
V^{4D}_\theta &= Vec_\theta(\underline{w}^p_{1,4}, V^{3D}_x)\\
V^{3D}_x &= Vec_x(\underline{w}^p_{1,3}, V^{2D}_x)\\
V^{3D}_\theta &= Vec_\theta(\underline{w}^p_{1,3}, V^{2D}_x)\\
V^{2D}_x &= Vec_x(\underline{w}^p_{1,1}, \underline{w}^p_{1,2})\\
V^{2D}_\theta &= Vec_\theta(\underline{w}^p_{1,1}, \underline{w}^p_{1,2}) \qquad (2.12)
\end{aligned}
$$

Architecture of the CORDIC based Iteration step can be represented as shown in figure 2.1

Modifying this architecture by keeping demultiplexers at outputs and proper control signals, this can architecture can be extended to work for any dimension (any n). Modified architecture for Iteration step is as shown in figure 2.2

2.2.2 Normalization

After every Iteration step, Normalization has to be done on every vector. This section describe how Normalization has been implemented using CORDIC. Similar to Iteration stage Normalization stage requires (n-1) CORDIC stages to implement n dimensional Normalization. Figure 2.3 represents the architecture for 4D FastICA Normalization stage

Modifying this architecture by keeping demultiplexers at outputs and proper control signals, this can architecture can be extended to work for any dimension (any n). Modified architecture for Normalization step is as shown in figure 2.4

2.2.3 Estimation

Estimation is done after the convergence of all the vectors and is the final step of the FastICA algorithm. It will multiply final Unmixing matrix with whitened inputs, which results in separated signals. Architectural implantation of Estimation is similar to Iteration stage with the exception that in Estimation step inputs are Normalized vectors.Figure 2.3 represents the architecture for 4D FastICA Estimation stage

Modifying this architecture by keeping demultiplexers at outputs and proper control signals, this can architecture can be extended to work for any dimension (any n). Modified architecture for Normalization step is as shown in figure 2.6

2.2.4 Complete Multiplexed Architecture

All the above steps implemented sequentially. So, we can use single CORDIC block to implement all these architectures by using proper multiplexers and demultiplexers at inputs and outputs respectively.

Figure 2.7 shows the multiplexed architecture of CORDIC based Reconfigurable ND-FastICA architecture.

From above steps, we can conclude that, for a single iteration we have to use total $(N-1) \times (L+1)$ times RotationMode and $(N-1)$ times Vectoring mode CORDIC.

2.3 Results

2.3.1 Methodology Validation

To verify the functionality of the proposed methodology, it is coded in Matlab and verified. Same architecture was coded in VHDL and synthesized in Synopsys Design Compiler(DC) using 130nm standard cell CMOS technology.

The algorithm was functionally verified by evaluation a set of sinusoidal and audio signals. Figure 2.8 shows the mixed, whitened, scaled whitened and separated signals using proposed architectures with sinusoidal inputs.

Same architecture can be used for separating any non-gaussian signals. Figure 2.9 shows the mixed, whitened, scaled whitened and separated signals using proposed architectures with audio inputs.

2.4 Hybrid ICA

Even thought hardware complexity reduced, computational delay increased with CORDIC based FastICA architecture. IN real time scenario, there is a need of separating engine which separates signals with minimal computational delay by maintaining low-complex hardware.

A new methodology has been proposed by combining FastICA with cross product to increase speed with little area overhead. In FastICA, all independent components(IC) are orthogonal to each other and n^{th} stage output is orthogonal to (n-1) previous ICs. It is well perceived fact that Cross product of (n-1) vectors results in new vector, that is perpendicular to previous vectors. So, nth IC is determined by calculating cross product of (n-1) vectors, which can be done in 1 clock cycle. There by, computational delay is decreased.

2.4.1 CORDIC based Hybrid nD-FICA architecture

In HybridICA architecture, (n-1)D ICA is combined with one stage of Cross product for nD signal separation. Cross product stage is used after ICA stage. Cross product output gives the n^{th} dimension unmixing vector. Here, for last vector there is no need to do all the iterations. Computational time, which originally used for computation of all Iteration, Normalization stages till convergence is saved by this hybrid implementation without compromising on functionality. After getting last vector from cross product, it is given to estimation stage. Estimation stage will give final separated signal. Here only one additional stage of Cross product is used. This is the area overhead of the proposed architecture. This additional area because of cross product is little compared to overall architecture. By this, we can achieve desired speed without adding much area overhead. As only one step of cross product is done instead of all Iteration and Normalization steps, that computational power is also reduced.

Figure 2.7 shows the multiplexed architecture of CORDIC based Reconfigurable ND-FastICA architecture.

2.4.2 Results

By this, separation of signals is done with optimum use of resources and with in less computational time. Here, we achieved both low-complexity and low-power and high speed, which is the main constraint of the today's biomedical real-time applications. Figure 2.7 shows the multiplexed architecture of CORDIC based Reconfigurable ND-FastICA architecture.

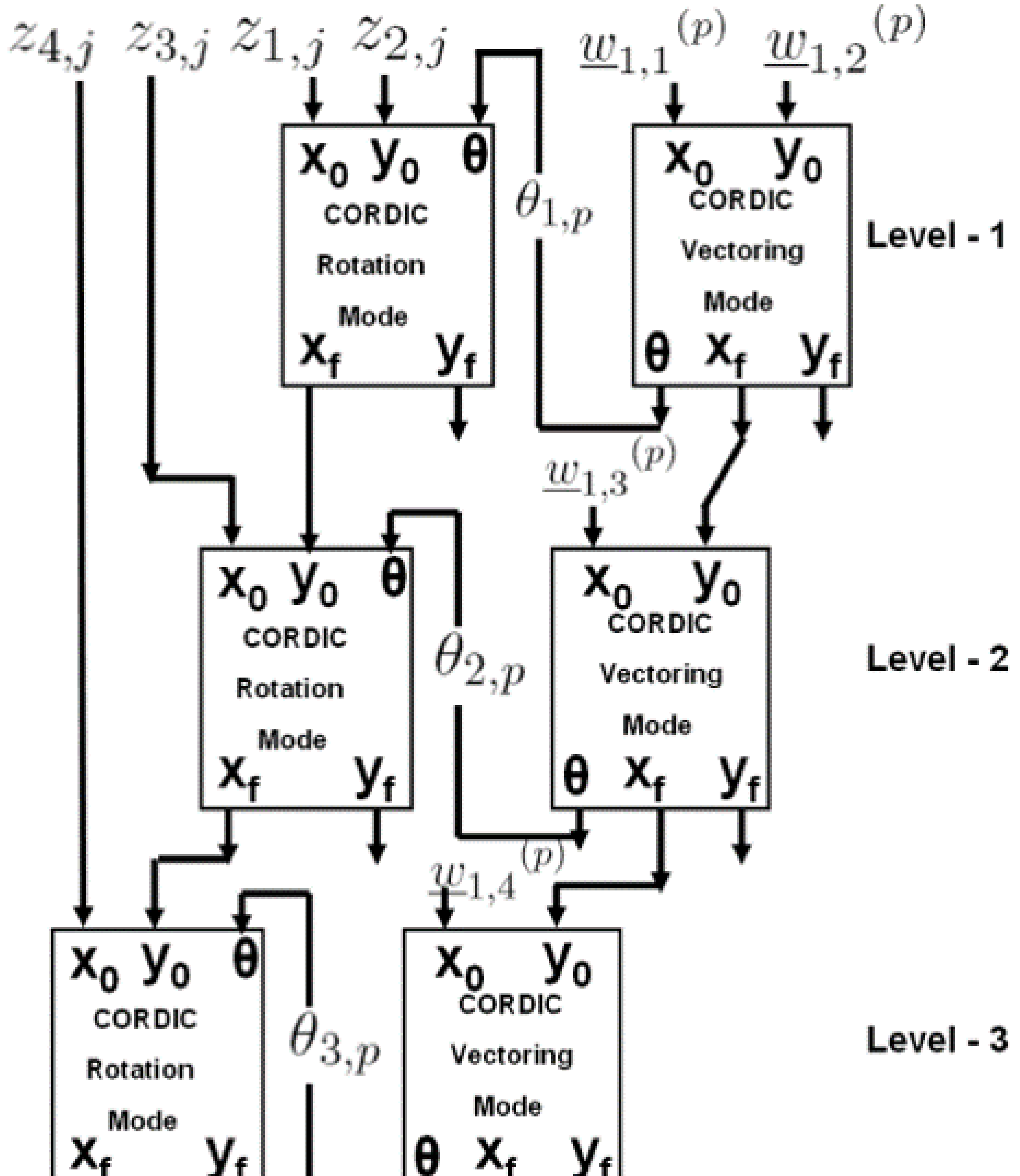
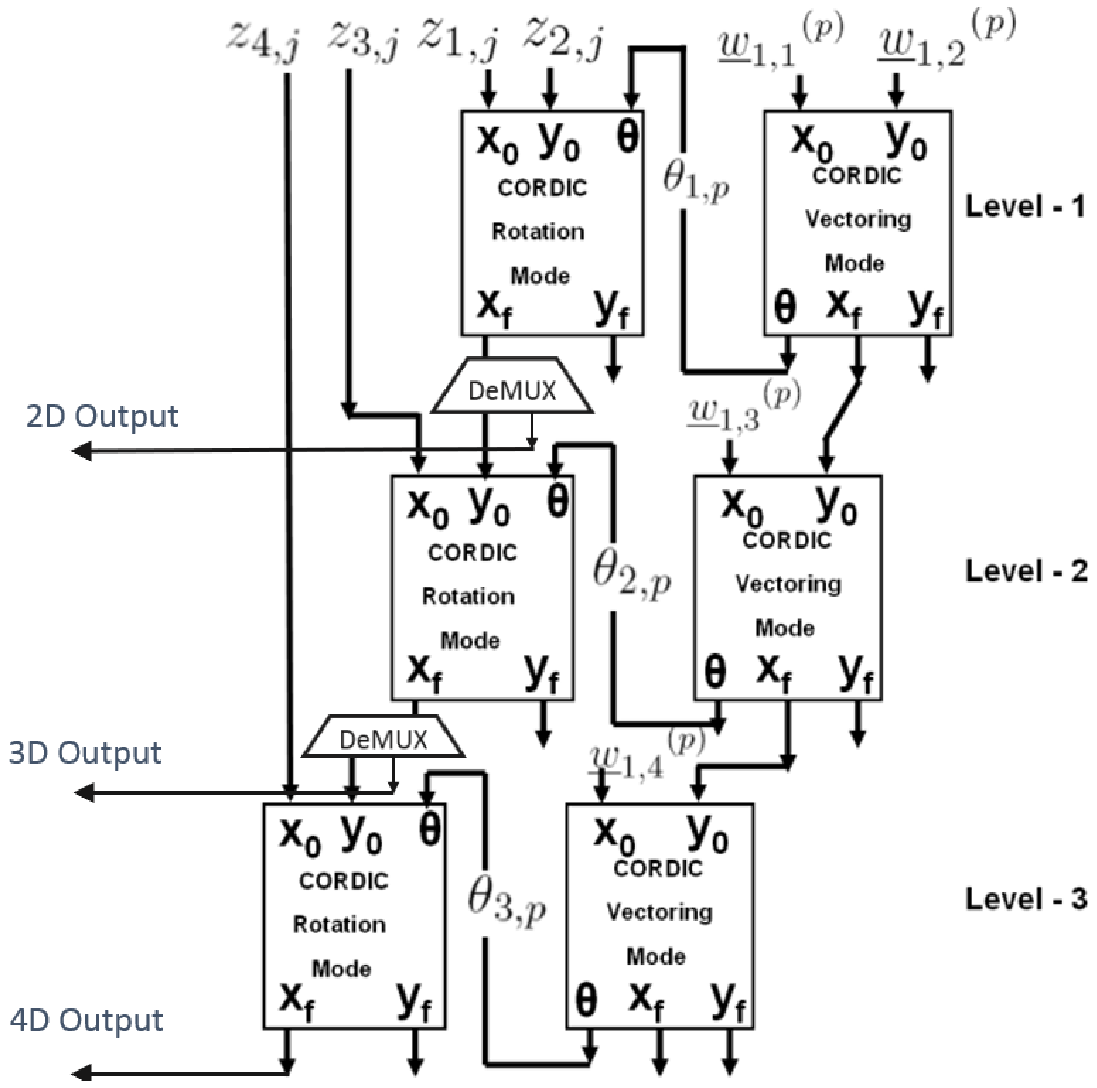


Figure 2.1: CORDIC based Iteration step



$z_{4,j}$
$z_{3,j}$
$z_{1,j}$
$z_{2,j}$
$\underline{w}_{1,1}{}^{(p)}$
$\underline{w}_{1,2}{}^{(p)}$
x_0 y_0 θ CORDIC Rotation Mode x_f y_f
$\theta_{1,p}$
x_0 y_0 CORDIC Vectoring Mode θ x_f y_f
Level - 1
DeMUX
2D Output
$\underline{w}_{1,3}{}^{(p)}$
$\theta_{2,p}$
Level - 2
3D Output
$\underline{w}_{1,4}{}^{(p)}$
$\theta_{3,p}$
Level - 3
4D Output

Figure 2.2: CORDIC based Reconfigurable Iteration step

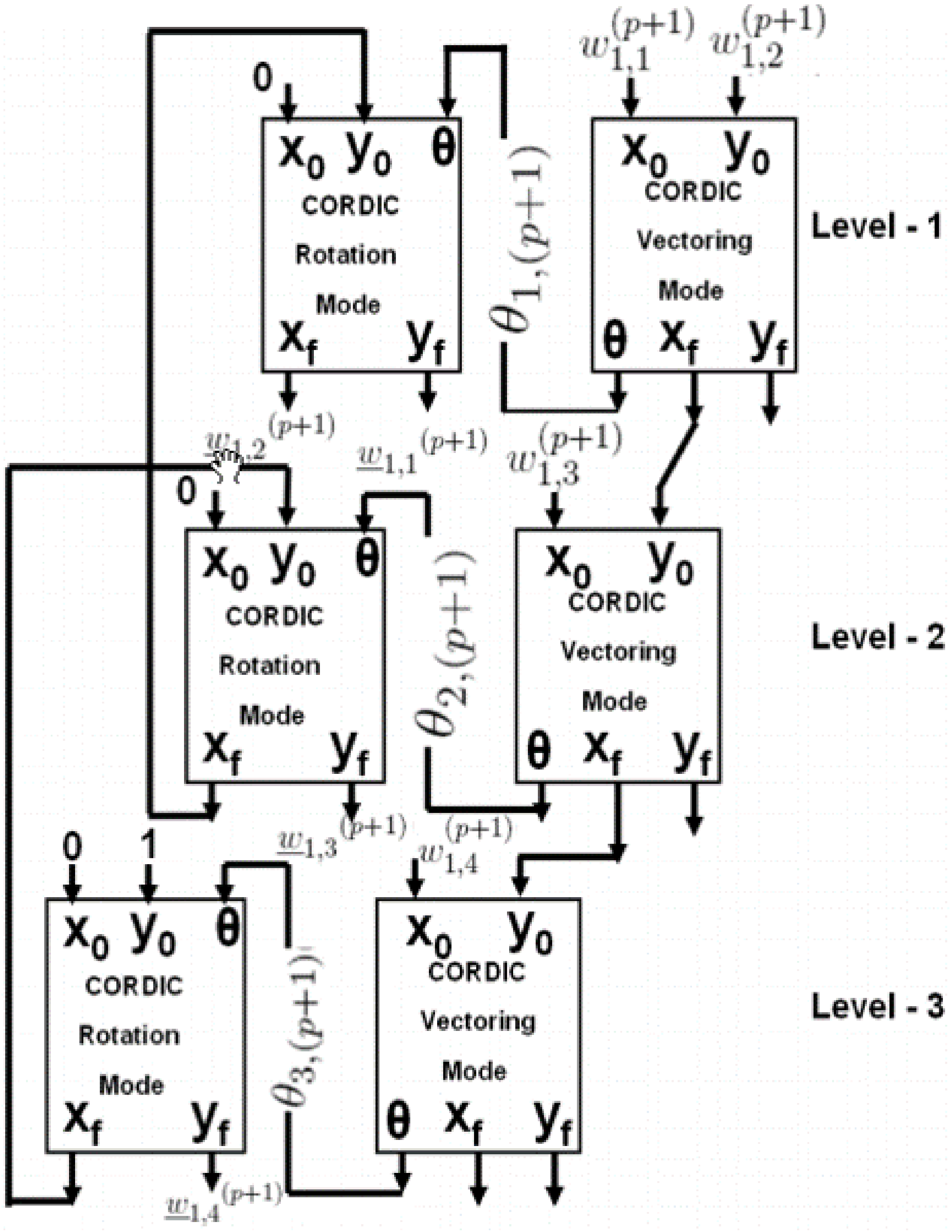


Figure 2.3: CORDIC based Normalization step

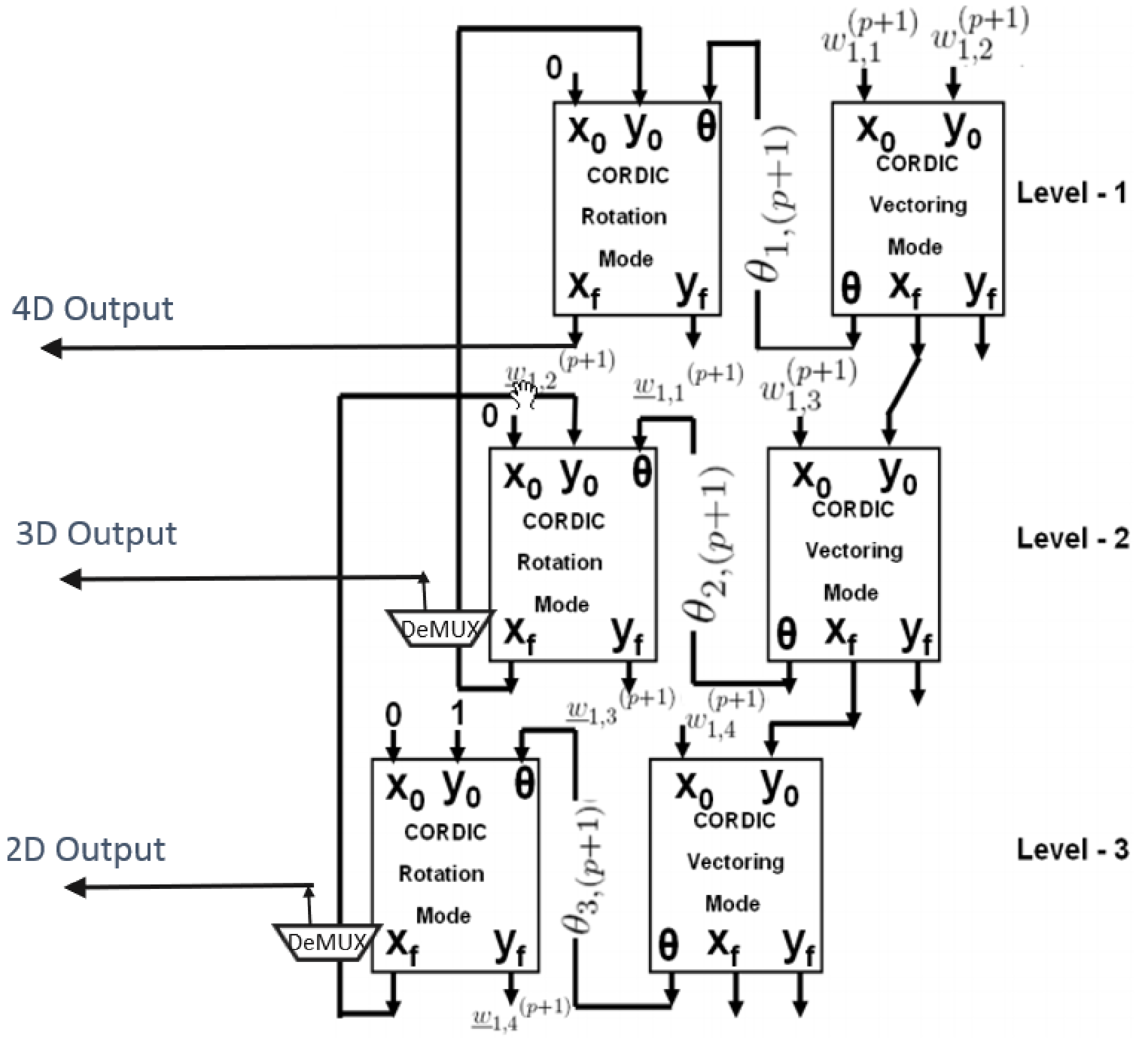


Figure 2.4: CORDIC based Reconfigurable Normalization step

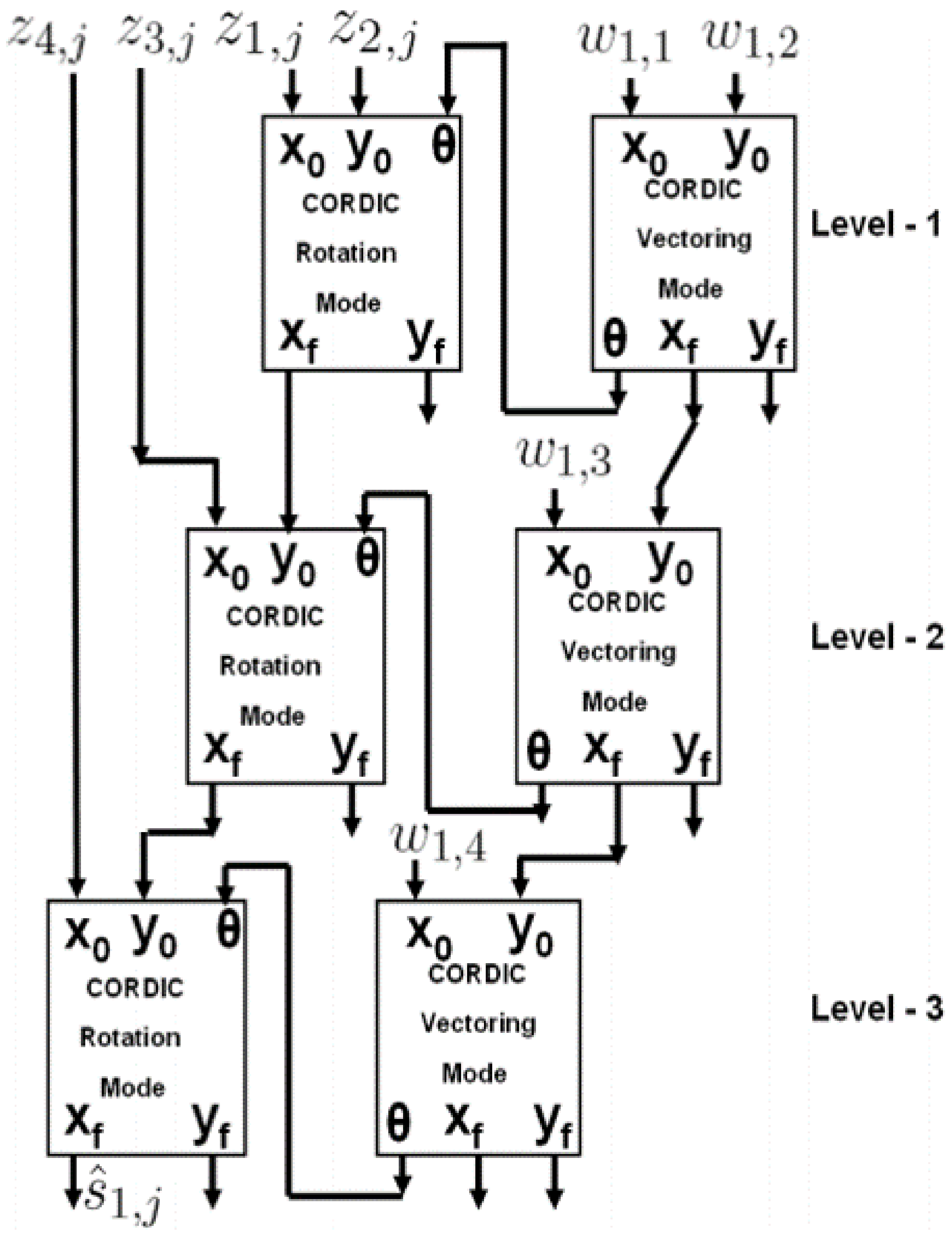


Figure 2.5: CORDIC based Estimation step

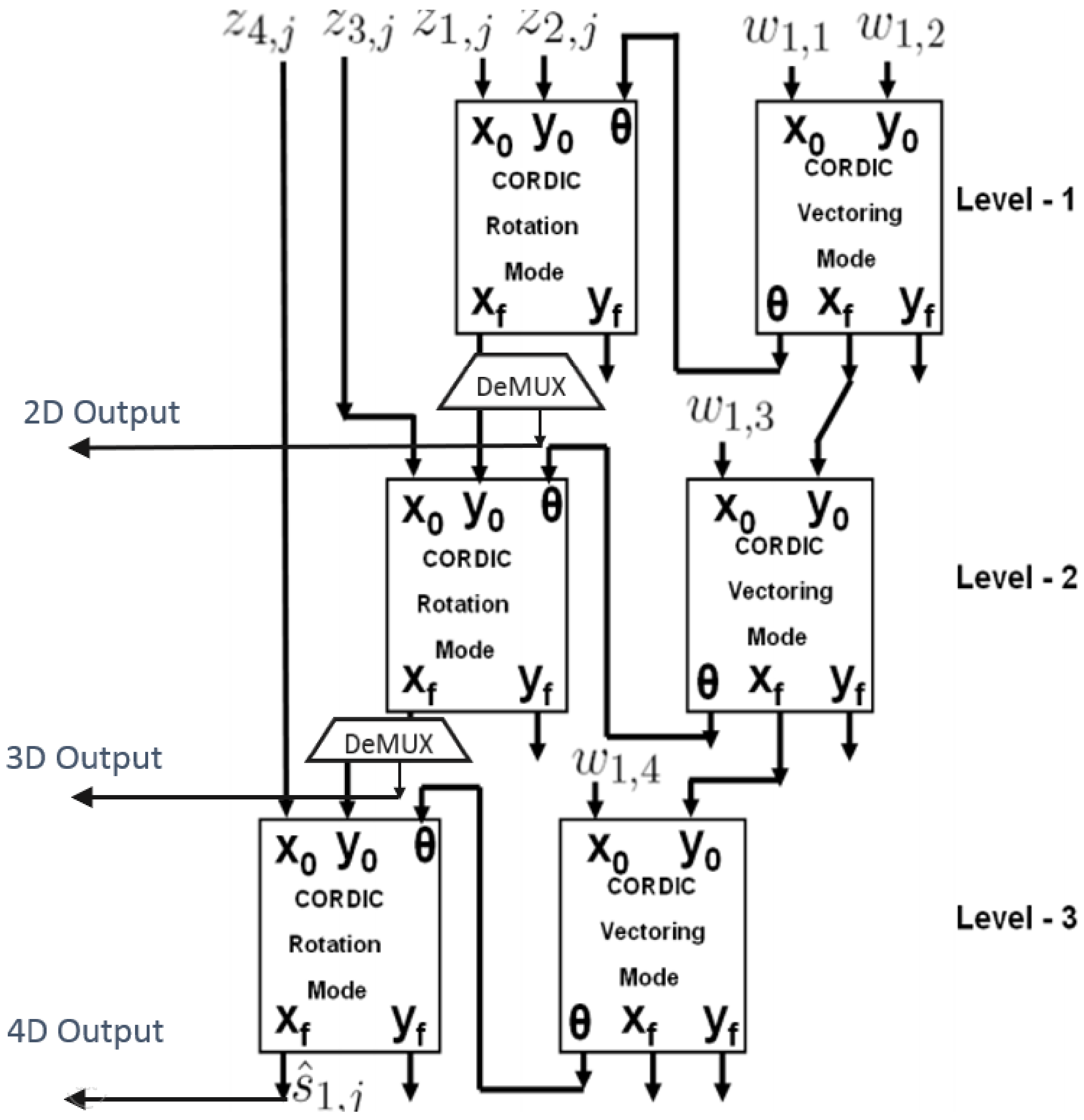


Figure 2.6: CORDIC based Reconfigurable Estimation step

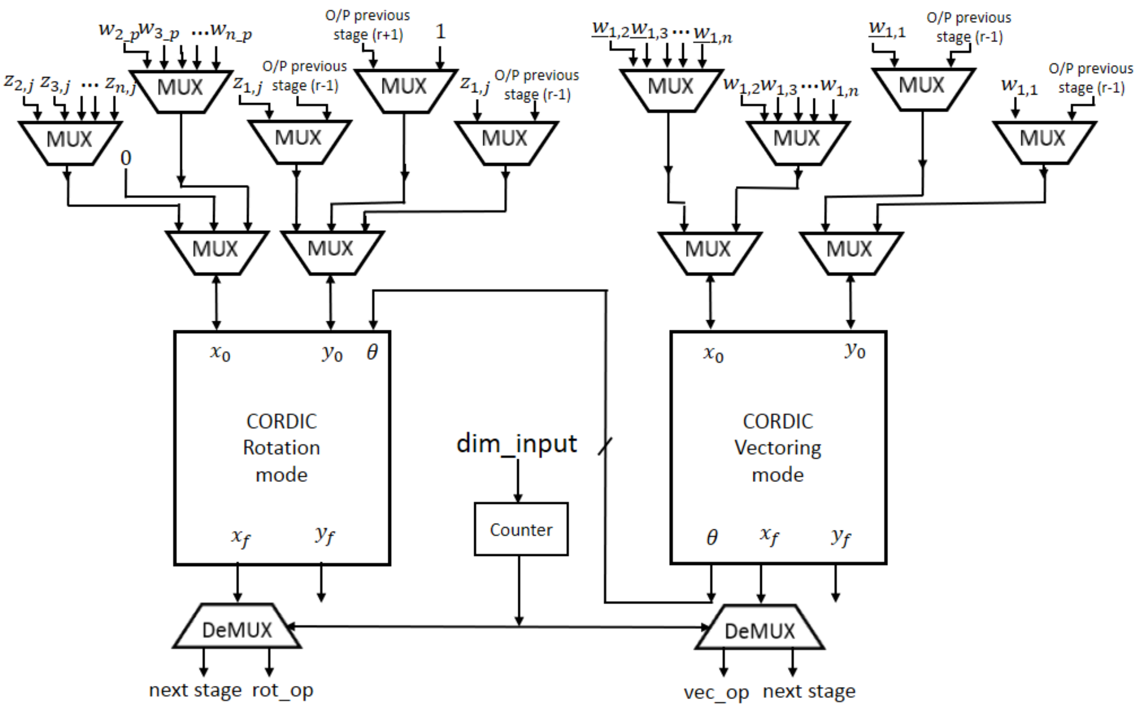


Figure 2.7: CORDIC based Reconfigurable ND-FastICA architecture

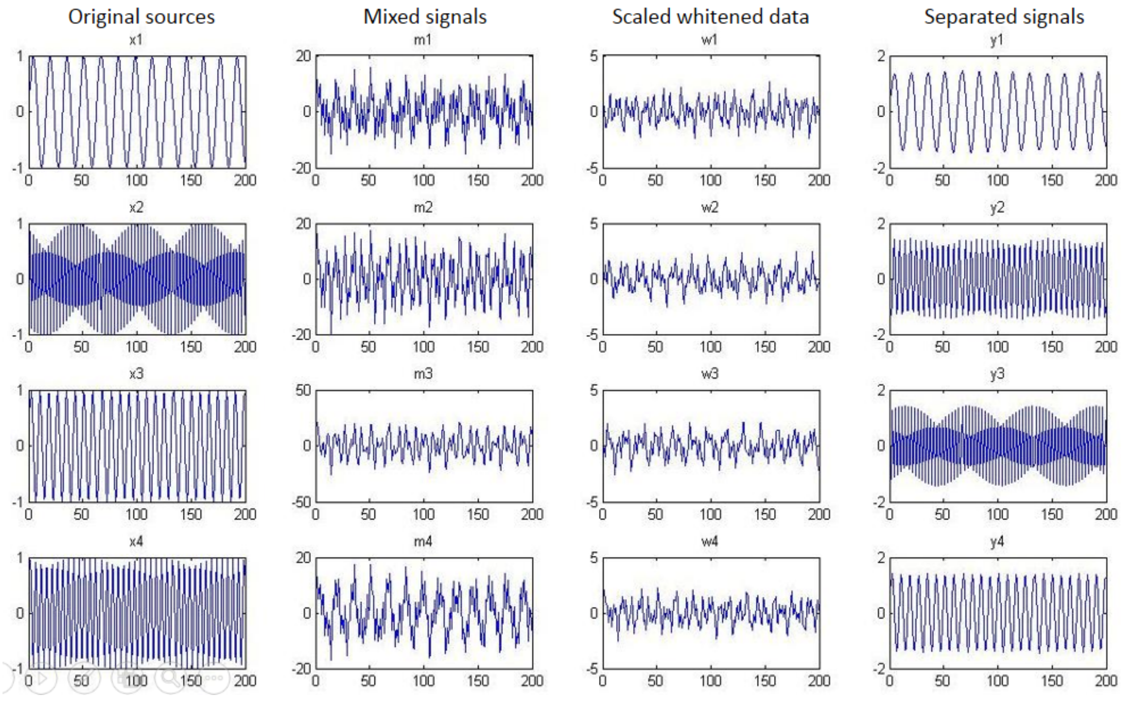


Figure 2.8: Separated outputs with Sinusoidal input

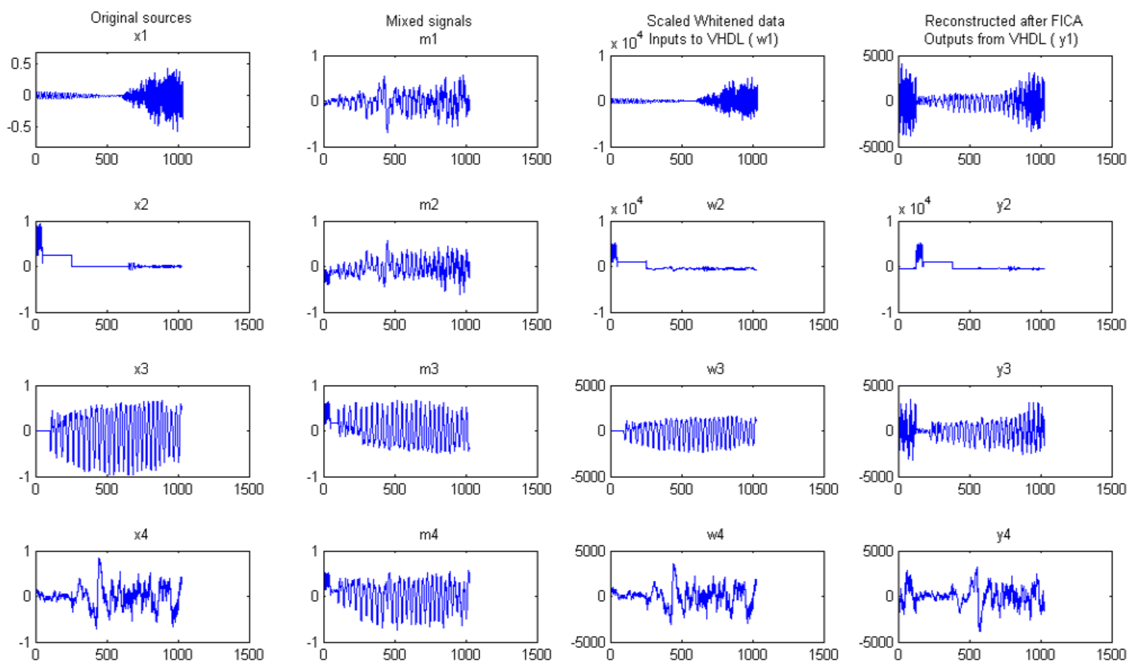


Figure 2.9: Separated outputs with Audio input

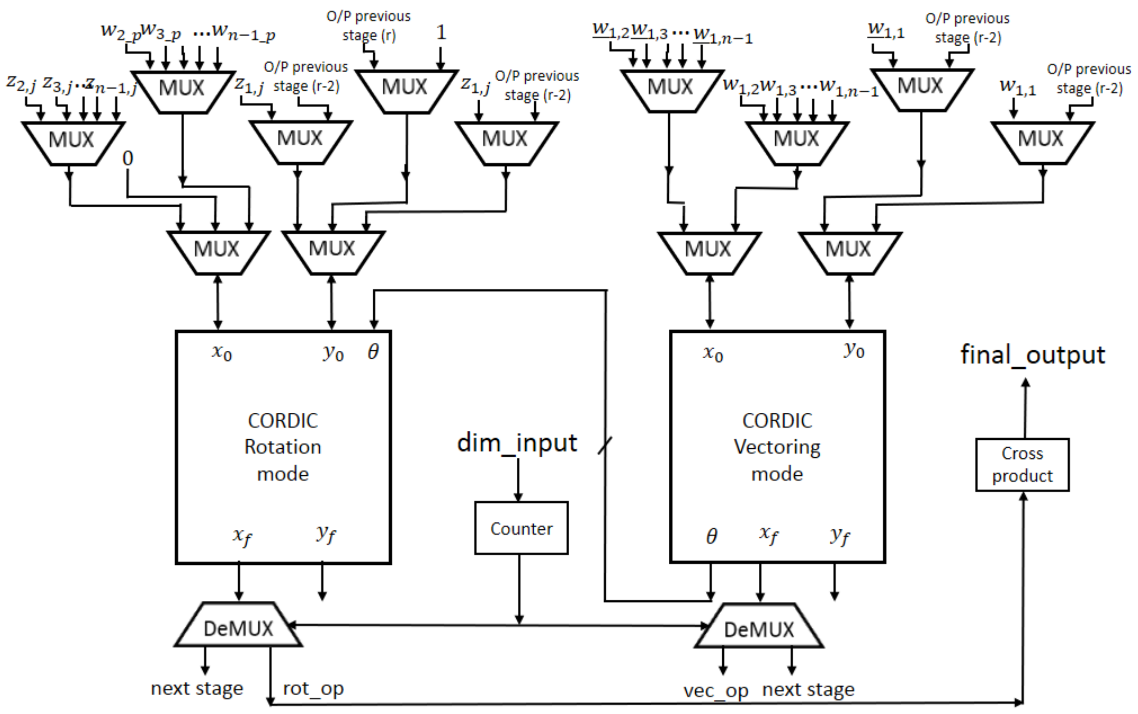


Figure 2.10: CORDIC based Hybrid ND-FastICA architecture

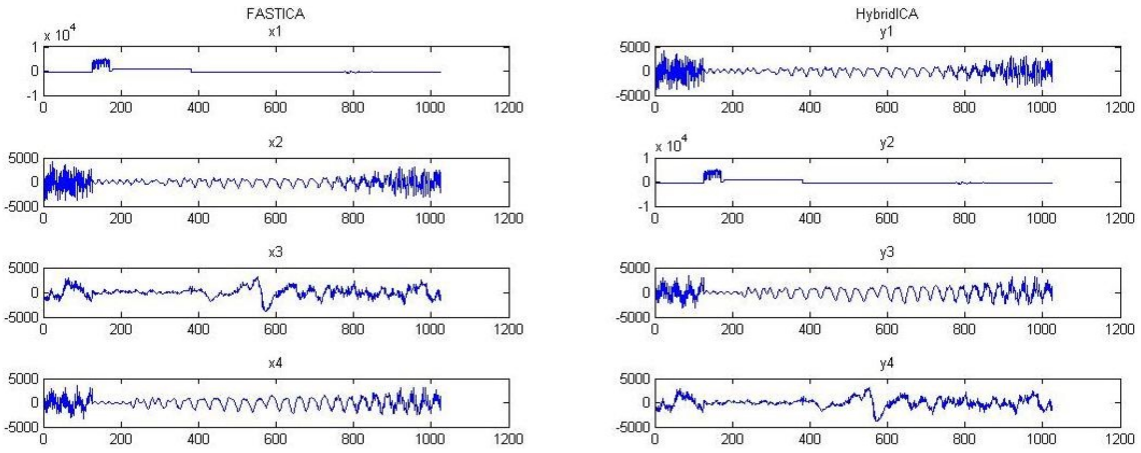


Figure 2.11: Comparative analysis of HybridICA with FastICA

Chapter 3

Co-ordinate Rotation based K-Means Clustering

The fundamental concept of cluster analysis is to form groups of similar objects as a means of distinguishing them from each other [7]. Clustering techniques that can be applied in any disciplines involving multivariate data, have been successfully used in diverse fields such as medicine (EEG, Functional MRI, activity recognition), geography or marketing and can be conveniently deployed with limited resources (memory and CPU) [7],[8].Hence, using K-Means for real- time cluster analysis in resource constrained applications such as wireless sensor networks (WSN) for remote health care monitoring systems where online multimodal data acquisition and analysis is the key (e.g. cardiovascular disease prognosis), requires an effective implementation strategy. The fundamental requirement for such applications is a low-power operation to prolong their battery life owing to their resource constrained nature. Hence, in this study, we investigate the use of a CORDIC-based low-complexity engine to implement K-Means clustering algorithm.

3.1 Preliminaries

3.1.1 K-Means clustering

With a given dataset X = x_i, $i = 1$,...,n to be clustered into a set of k clusters, the K-Means algorithm iterates to minimize the squared error between the empirical mean of a cluster and the individual data points, defined as the cost function,

$$J(\theta, u) = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}(x_i - \theta_j)^2 \tag{3.1}$$

where, θ_j is the cluster center and $u_{ij} = 1$ if x_i lies close to θ_j, or 0 if otherwise. Initially k centroids are defined and the data vectors are assigned to a cluster label depending on how close they are to each centroid. The k centroids are recalculated from the newly defined clusters and the process of reassignment of each data vector to each new centroid is repeated. The algorithm iterates over this loop until the data vectors from the dataset X form clusters and the cost function J is minimized [9].

3.1.2 Coordinate Rotation Digital Computer

CORDIC is an efficient implementation technique for vector rotation and arctangent computation. Since it can be realized using simple shift and add operations, it is very effective in terms of low hardware complexity [40]-[43]. CORDIC in general, can be operated in two modesVectoring and Rotation [43]. In the Vectoring mode, given the vector coordinates, the magnitude of vector, angle between the initial vector and the principal coordinate axis is computed and in the rotation mode, given the angle of rotation and the initial vector, the final vector is computed. In the proposed methodology, 16 stage CORDIC is used in vectoring mode which takes 16 clock cycles to complete one vectoring mode operation. Considering the rotation in the clockwise direction, the basic CORDIC expressions can be expressed as

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{3.2}$$

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 this paper, we use CORDIC in vectoring mode for our implementation.

3.2 Proposed Methodology

Fig.1 shows the architecture of the proposed CORDIC based K-Means clustering engine. As can be seen from Fig.1 all the data inputs are stored in a memory unit for further usage and are transmitted to different blocks via control unit (CU). Memory unit also serves the purpose of storing intermediate values. The Euclidean distance is calculated in distance unit (DU) using low complexity CORDIC vectoring module. The distance from each point to the centroids are sent to a comparator block to identify the cluster to which it belongs. Once the clustering is done, centroid calculation block will be activated to compute the new centroids. If these new centroids differ significantly from the previous iteration values, then clustering will be done one more time. Otherwise, clustered data will be sent to the output. The CU governs the data flow among all the modules. The proposed engine utilizes CORDIC to compute Euclidean distance between two points, which is a metric to compute the clusters and has been explained through an illustrative example.

In this study, our focus is to propose a methodology for utilizing CORDIC to compute Euclidean distance between two points, which is a metric to compute the clusters. In two dimensional signal space, if (x_1, x_2) , (y_1, y_2) are two points, the Eucliean distance between these two points will be

$$dist_{2D} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

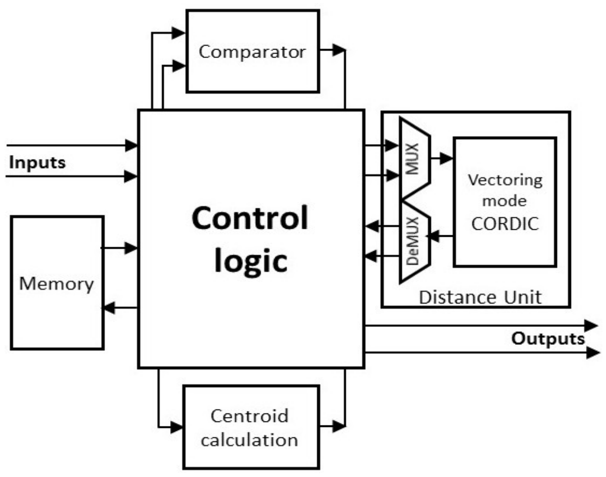


Figure 3.1: Complete architecture of the proposed system.

One square rooter, two square, one adder and two subtraction operations are involved in this computation. If we give a and b as the x- and y-inputs to the vectoring mode CORDIC, the x output will be the magnitude of Vector(a, b), which is $\sqrt{a^2 + b^2}$. So, with $(x_1 - y_1)$ and $(x_2 - y_2)$ as the x- and y- inputs repectively, the vectoring mode will give distance between the two points. Architecture of the 2D distance measurement unit using vectoring mode CORDIC is shown in the Fig. 2(a). We can extend this methodology to n-dimensional signal space to formulate distance between two n-dimensional vectors. Considering the case of 3-D signal space (n=3), distance between these two points $(x_1, x_2, x_3), (y_1, y_2, y_3)$ will be

$$distance_{3D} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}.$$

With inputs as $(x_1 - y_1)$ and $(x_2 - y_2)$ to vectoring mode as in 2-dimensional case, the x-output of vectoring mode CORDIC(vec_x^{l1}) is represented as

$$distance_{2D} = vec_x^{l1}((x_1 - y_1), (x_2 - y_2)).$$

If we pass this x-output to next CORDIC level as one input, with $(x_3 - y_3)$ as second input, x-output will be the desired 3-dimesional distance

$$distance_{3D} = (vec_x^{l2}(vec_x^{l1}((x_1 - y_1), (x_2 - y_2)), (x_3 - y_3)).$$

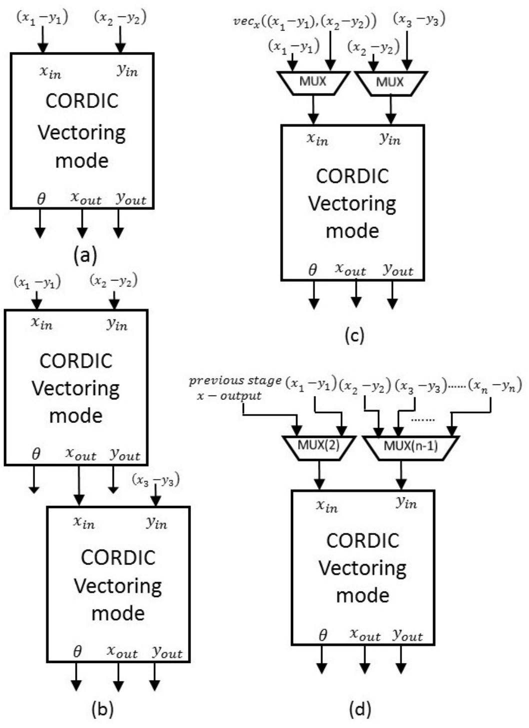


Figure 3.2: CORDIC based distance measurement (a)2-D vectoing;(b)3-D vectoring;(c)3-D multiplexed architecture;(d)n-D multiplexed architecture.

Fig. 2(b) shows the 3D distance measurement unit. Since these two stages are executed sequentially, same CORDIC unit can be reused only at the expense of two multiplexers as shown in Fig.2(c). For two n-dimensional vectors $(x_1, x_2, x_3, \ldots, x_n)$ and $(y_1, y_2, y_3, \ldots, y_n)$, the $distance_{nD}$ between them,

$$distance_{nD} = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

can be calculated using (n-1) CORDIC stages as shown in Fig. 2(d). For calculating distance between two n-dimensional vectors, the inputs to the vectoring mode CORDIC block will be as follows.

For 1st level of CORDIC x-input: $(x_1 - y_1)$

(vec_x^{l1}) y-input: $(x_2 - y_2)$

For pth level of CORDIC x-input: previous level x-output

(vec_x^{lp}) y-input:$(x_p - y_p)$,where $p = 2$ to n

Thus, recursively using the fundamental 2D-CORDIC unit we can generalize it for computing the n-dimensional distance. As the calculations are done sequentially, the same CORDIC unit can be reused for calculations in two levels, only at the expense of one 2-input multiplexer and one (n-1) input muliplexer at inputs of CORDIC unit (as shown in fig. 2(d)). Multiplexers at input and Demultiplexers at output of CORDIC block has been designed accordingly.

3.3 Results, Analysis and Discussion

3.3.1 Methodology Validation

The proposed architecture was coded in VerilogHDL and synthesized using Synopsys Design Compiler (DC) using $0.13\mu m$ standard cell CMOS technology.

To verify its functional correctness, the algorithm was evaluated on a set of 24 datasets each of 60 samples of kinematic data (collected from a wrist-worn tri-axial accelerometer measuring human arm movements). The output was validated against Matlab model of the K-Means algorithm with same initial seed values for both the cases. The results indicate the similarity in predicting the number of clusters and the iterations taken by both the approaches. Moreover, we achieve precision up to 8 decimal places using a 16-stage vectoring mode CORDIC with input magnitudes ranging from 1 to 10^{15}. Fig. 3(a) shows the resulting clustered data from Matlab inbuilt kmeans function and Fig. 3(b) shows the clustered data output using the proposed methodology. The comparison of Fig. 3(a) and Fig 3(b) shows that the proposed methodology produces 100% accuracy, exhibiting a robust system.

3.3.2 Hardware Complexity Analysis

Throughout the hardware complexity analysis, we keep a generalized view of word-length b and followed the same procedure used in [26] and [39]. Since the distance

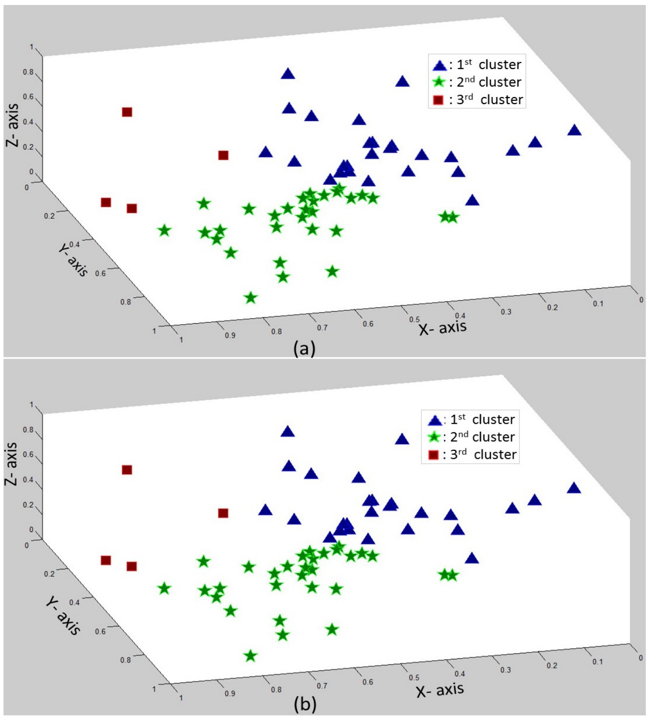


Figure 3.3: 3D plot of the data points clustered using (a) inbuilt kmeans function; (b) CORDIC based kmeans function.

computation in K-Means Clustering using CORDIC is an iterative procedure, we consider only one single iteration because the same hardware can be reused for the next iterations as well as for successive stages of CORDIC in Vectoring Mode for dimensions higher than 2.

Computation of distance between 2 n-dimensional points $(x_1, x_2, x_3, \ldots, x_n)$ and $(y_1, y_2, y_3, \ldots, y_n)$ using the conventional method, i.e. $\sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$ requires n squaring operations,$(n-1)$ addition operations and 1 square-root operation. Since this distance is only used for comparison, the absolute value of the distance is not required and hence, square rooting operation can be omitted without compromising on accuracy. To provide a comparison on a uniform platform, we consider only Ripple Carry Adder (RCA) and Conventional Array Multiplier (CAM) as the means of implementing the arithmetic operations.

One $b-$bit RCA requires b full adders (FA) (in a simplified view) [39], and $b\ X\ b$ CAM requires $b*(b-2)$ FA plus b half adders (HA) and b^2 AND gates [39]. Similarly,

Table 3.1: COMPARISONS OF THE PROPOSED ARCHITECTURE WITH OTHER REPORTED ARCHITECTURE

	TMC 11[16]	ETCAS 11[17]	TCAS 13[18]	Proposed
Technology	TSMC 90nm	TSMC 90nm	TSMC 90nm	TSMC 130nm
Clock Frequency	500MHz	625MHz	400MHz	100MHz
Area	$0.67mm^2$	$0.14mm^2$	-	$0.36mm^2$
Normalized Area*	$0.67mm^2$	$0.14mm^2$	-	$0.1725mm^2$
Power	209mW	12.1mW	6.02mW	9.21mW
Normalized Power*	209mW	12.1mW	6.02mW	4.6mW
Vector Dimension	1-5	1-4	1-16	2-6
Maximum Data	2^{20}	2^{20}	2^{16}	2^{22}

* Normalized to 90nm from 130nm, 180nm, and the normalization factor is $(90/180)^2$, $(90/130)^2$ for 90nm and 130nm respectively.

one b-bit SQRT needs $0.125 * (b+6)b$ FA and XOR gates [72]. In addition, considering one FA cell requires 24 transistors, one HA cell, and one two input XOR gate consist of 12 transistors and a two input AND gate consists of six transistors [39], we can calculate TCA $= 24b$, TCM $= 6b(5b-6)$, where TC* are the transistor counts for RCA and CAM respectively. Following the same procedure used in [26], savings in terms of arithmetic operations for distance computation in different dimensions without using CORDIC are computed. For n-dimensional distance computation, $(n-1)$ RCAs and n CAMs are required. The transistors used here will not be required when the proposed CORDIC based engine is used for K-Means clustering, since we are reusing the CORDIC unit. Therefore, the total Transistor Count (TC) computed here will be the Transistor Saving (TS), given by:

$$TS_{nD} = nTC_M + (n-1)TC_A$$

Expressing TS_{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 [68]. Being the function of b and n, the Figs. 4 and 5 show that the increase in TSPW for the proposed CORDIC based K-Means Clustering methodology is significantly higher than the conventional design for two dimensional to six dimensional $((184.5 * b - 69)TSPW)$ points, for different word lengths $(4 <= b <= 32)$.

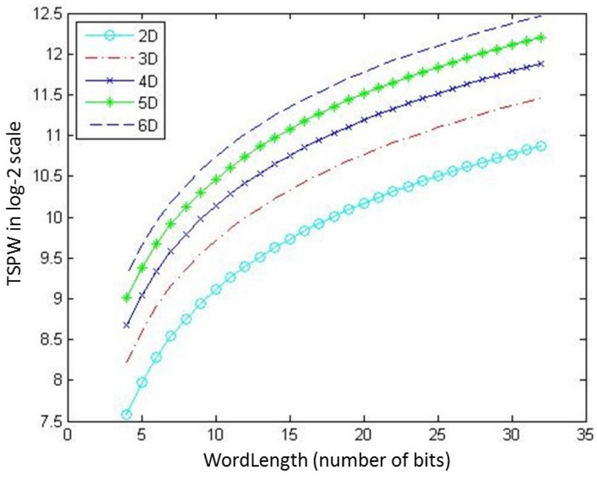


Figure 3.4: Variation of Transistor saving per word-length of the proposed algorithm with word-length and dimension.

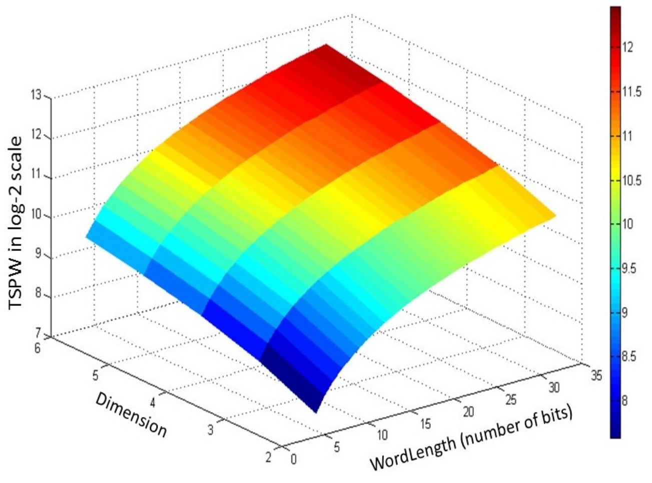


Figure 3.5: Comparative variation of Transistor Saving Per Word-length (TSPW) of the proposed algorithm word-length for 2D to 6D.

3.3.3 Error Analysis

Accuracy is determined by comparing outputs from proposed methodology with Matlab inbuilt 'kmeans' function. In the proposed technique CORDIC is used for calculating the Euclidean distance. The accuracy depends on the input dimension (dim)

and the number of micro-rotations, which is equal to number of stages(n) of CORDIC. A set of 1000 randomly generated signals are taken as input and Mean Absolute Percentage Error(MAPE) was calculated with various stages of CORDIC for different input dimensions. Fig.6 shows the variation of MAPE for the proposed architecture with dimensions ranging from 2D to 6D, with different CORDIC stages n=8, 12, 16 and 20, and with word length b=4, 8, 16 and 32. MAPE is of the order of 10^{-2} with number of CORDIC stages(n) as 8 for a given dimension while with n=16, MAPE decreases to the order of 10^{4}. From Fig.6 it is evident that as number of CORDIC stages increase, MAPE decreases due to the increase in resolution of CORDIC. Furthermore, for a fixed number of stages of CORDIC, MAPE increases with the dimensionality of the input because subsequent dimensions are passed on to CORDIC in a cascaded fashion. In addition, from Fig.6 it is observed that for a given dimension and number of CORDIC stages, MAPE does not change significantly with the word length b. Moreover, this (MAPE) is the error in Euclidean distance which is used to cluster data points points separated by higher order distance. The final output is the co-ordinate of the cluster centroids and not the Euclidean distance itself, thereby ensuring a robust output.

3.3.4 Analysis of computational time

In the proposed architecture 16 stage CORDIC is being used. So, vectoring stage takes 16 clock cycles to compute single 2 dimensional Euclidean distance. However, all CORDIC stages are sequential and at any instant of time only one of these 16 stages works on the particular data point. So, 16 different data points can be processed in 16 stages of CORDIC at any clock cycle, thereby achieving 100% throughput with the latency of 16 clock cycles.

For each cluster (K), distances from every cluster centroid to every data point has to be measured. And to calculate n dimensional distance, $(n-1)$ stages of CORDIC

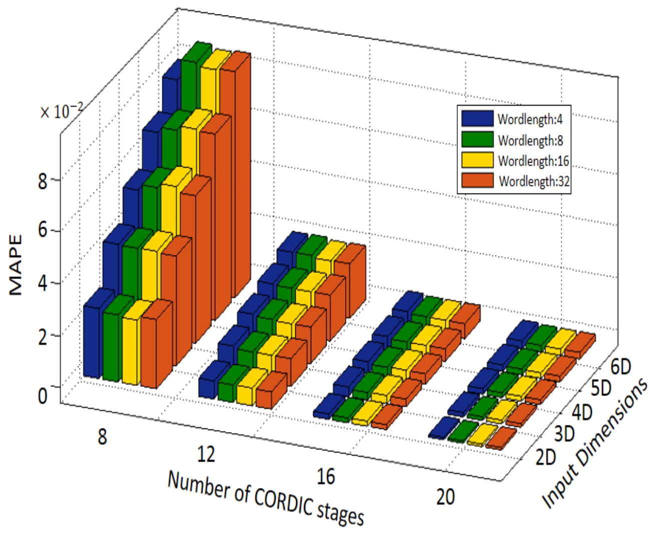


Figure 3.6: Variation of Average Percentage error of the proposed algorithm with number of CORDIC stages, dimension and wordlength .

are needed. So, each iteration takes $k * num * (dim - 1) + 16$ clock cycles, where num is number of data points, dim is dimensionality of data and k is number of clusters. Each iteration is having 16 clock cycles of latency as we are using 16 stage CORDIC. The variation of required number of clock cycles with varying dimensions and number of clusters for a dataset having 64 points is shown in Fig.7. For clustering 64 datapoints into 3 clusters, a 16 stage CORDIC distance measuring unit takes 400 clock cycles. Proposed architecture at $100MHz$ operating frequency will take $4us$ to complete single iteration step. Here, number of iterations depend on initial seeds and distribution of data. Considering worst case scenario of 1000 iterations, total K-Means architecture takes less than $1ms$ to complete. So, we can clearly say that proposed design can achieve real-time standards.

3.3.5 Comparison with other Architectures

The proposed architecture is synthesized by Synopsys Design Compiler (DC) and the place an route was performed using IC Compiler (ICC) using $0.13\mu m$ standard cell CMOS technology. The core area and power consumption of the proposed engine

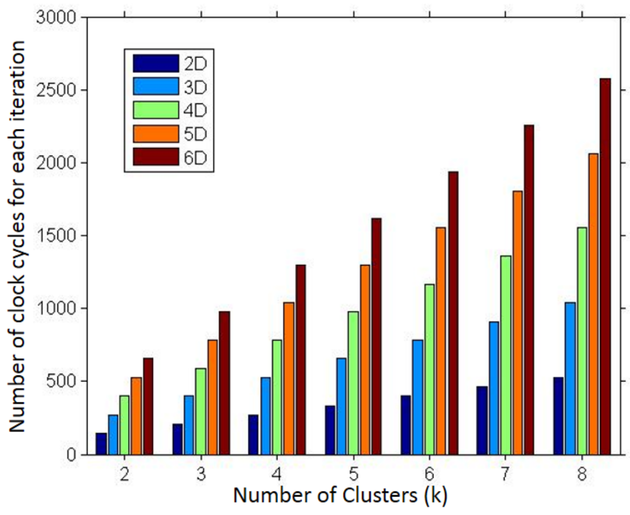


Figure 3.7: Variation of required Number of clock cycles for each iteration with number of clusters and dimension of input.

are $0.61mm^2$ and $9.21mW$ at 100-MHz frequency for VDD = 1.2 V. The engine consumes 62% less power with a comparable area consumption w.r.t. state-of-the art architecture for ASIC implementation in [17](The power reported is from backend simulation using SoC Encounter).

A comparison of the area requirement and power consumption of the proposed engine with state-of-the-art architectures have been highlighted in Table I. Since different architectures use different technologies, it is unfair to compare them on a one-to-one basis. So, area and power values are normalized to same technology node [?]. These results are provided to give an insight about the performance of the proposed-methodology-based architecture. The proposed CORDIC based clustering engine compares favorably in terms of area with respect to the other reported architectures as illustrated in Table I. It is to be noted that due to the unavailability of an appropriate memory module in our standard cell library, the architecture is implemented using registers. We believe that the use of appropriate memory will significantly reduce the area and power consumption.

Chapter 4

Estimation of Unique Activities using Unsupervised Learning

The primary motivation of this work was to detect the number of these distinct arm movements in an *out-of-laboratory* condition (i.e. real world) using minimal number of sensors, as opposed to detecting gross static or dynamic activities and postures like standing, sitting, running, cycling, brushing teeth [28]-[31], etc. The detection and enumeration of the number of different categories of arm movements (e.g. prescribed exercises) during daily activities, could over time provide a measure of arm rehabilitation progress in remote health monitoring applications, especially in neurodegenerative pathologies such as stroke or cerebral palsy. We employ an unsupervised clustering technique using k-means which can infer on the unique category of movements performed with paretic arm in a daily life scenario by analyzing the wrist-worn inertial sensor data. The number of distinct clusters over a longitudinal period could qualify as an impaired arm usage and an improvement of the underlying motor functionality. This is due to the fact that in the initial phase of rehabilitation, the kinematic signals pertaining to the different arm movements performed by the patients will have overlapping or inconsistent patterns reflecting the high degree of variability inherent in their movement. However, as the patient improves each unique category of movement will exhibit a particular characteristic leading to the formation of unique clusters in a multi-dimensional feature space.

One of the most significant facts often overlooked in relevant state-of-the-art literature is the absence of requisite computing facilities available at the disposal of the patients particularly while monitoring in nomadic settings. A nomadic monitoring scenario also nullifies the use of a dedicated training session, one of the significant factors considered for most supervised classification approaches which leads to the collection of a huge set of data under controlled conditions from each subject for a personalized evaluation. The envisaged application scenario comprises of the patient wearing a sensor device throughout a monitoring duration and performing daily activities. The data generated by the body-worn sensors worn by the patients during daily life monitoring can be thus processed within the sensor node itself to infer the unique category of movements performed by a patient over the monitoring duration. This in a true sense characterizes the concept of daily life monitoring where the subject need not be enclosed within their home and remain dependent on the availability of computing facilities at all times. Recognition of unique arm-movements involves two steps feature extraction and clustering to deduce the unique category of arm movements. Hence, in this work we propose a unified architecture for computing relevant time-domain features from the sensor data and formulating clusters on them. CORDIC based architectures exploring its different transcendental functions to compute complex arithmetic operations [32]-[37] have been used widely for various low-power biomedical and computationally intensive signal processing applications. An efficient way for feature extraction using a low-complexity CORDIC-based engine was proposed in [35]. In this work, we propose a low-complex methodology to do unsupervised clustering reusing CORDIC. A further reduction of hardware is achieved by reusing the CORDIC engine for clustering as feature calculation and the clustering steps are done serially.

4.1 Background

Activity recognition of gross movements and postures using inertial sensors involves the primary steps of feature extraction, feature selection and classification [28],[29],[30],[38],[39]. Different machine learning techniques have been used for movement recognition, e.g. Support Vector Machines (SVM) [40]-[41], Decision Trees (DT) [38], [39], [42], k-Nearest Neighbor [42], Naive Bayes (NB) [39], [42], Multi-Layer Perceptron (MLP) [43], Artificial Neural Networks (ANN) [38], Hidden Markov Models (HMM) [44], or a combination of these techniques [24]. Instance-based classification algorithms have also been used successfully, but they suffer from high memory usage and long processing times [45]. A stochastic approximation framework for intensity-independent activity recognition was employed in [46]. A combination of HMMs and Gaussian mixture models on waist-mounted accelerometer data was used for classification in [47].

The accuracy of any movement recognition technique is dependent on the system components and requirements, covering areas such as: type of activities, number of activities, type of sensors, number of sensors, placement of sensors [43], level of data fusion [84], and most importantly the recognition methodology adopted. A few studies have included sensors on different parts of the body and on accessories such as belt, shirt pocket, etc., besides using multi-modal sensor devices (accelerometer, microphone, phototransistor, barometer, etc.) to recognize a wide array of activities. On the other hand, a few studies have targeted using minimal number of sensors (e.g. single tri-axial accelerometer on smart phones) to make the system less invasive for daily life monitoring. Recognition strategies generally employ supervised learning which requires a subject specific training [48] and follow any of the three themes.

Firstly, using only data collected under controlled conditions (e.g. in the laboratory) for training as well as testing [38]. Secondly, using both controlled and un-controlled data (e.g. out-of-laboratory) for both training and testing [29], [48] and finally, using controlled data for training and only un-controlled data for testing [29], [48].

Unsupervised learning techniques used for activity recognition [57] are Neural networks [58], Web mining [59], [60], [61], suffix-trees [62], Jeffrey divergence [63]. Clustering has also been a popular unsupervised technique in the domain of activity recognition. Fuzzy clustering has been used to classify data in [64] for different applications but they use multi- camera system instead of inertial sensors. To the best of our knowledge, very little has been reported in terms of recognizing fine grain activities e.g., upper limb movement in out-of-laboratory settings, which is an important aspect for assessing rehabilitation of impaired limb functionality such as in stroke. Gesture recognition has however been prevalent in the research community, aimed at monitoring specific hand gestures for human-computer interaction or the monitoring of dietary intake for nutrition monitoring applications. Sensors have been placed on the objects being manipulated [49] or embedded in special gloves [50]. Tool-based workshop activities such as sawing, hammering etc. were recognized in [51] using an intensity-analysis on the signals from two microphones depending on the sound associated with a particular hand tool. Gaussian HMMs were successfully used for modeling the gestures which were detected with an overall accuracy of 84%. Gestures involved in human feeding motion have been looked in [52] using a two-fold approach - first detecting object-interaction gestures and second, focusing on dietary intake gestures. HMMs were used to model the spatial temporal variations using five inertial sensors, recognizing sporadically occurring gestures from continuous data stream with 70-80% precision. An attempt to classify 10 arm activities as part of assembly-line workers in a car production environment were made in [53] using meta-classifier that fuses information of classifiers operating on 19 individual sensor nodes distributed over the two arms. Here, HMMs were used to handle temporal variations in gestures, along with the naive Bayes classifier to achieve recognition up to 80% using data from a single inertial sensor node. Accelerometer based gesture recognition using Continuous Time Recurrent Neural Networks (CTRNN) has been performed in [54]. This method has the advantage of operating on the raw data directly rather than using features as used in other classification methods. Eight gestures such as sitting, standing, reading books, opening drawers, etc., performed in a unconstrained environment were classified with 64% accuracy. An SVM based gesture recognition using objects attached with sensors to recognize drinking, phone use and writing activities was introduced in [55], which achieved a performance of 72%, 84% and 80% respectively for each activity.

A recent study in [56] have used k-means clustering for arm movement recognition. However, this study incorporates a supervised scenario where a large training database for each subject is first collected to form the clusters. Hence this does not suffice the application scenario where unsupervised recognition of movements using inertial sensors is the key. In view of this we intend to use the k-means algorithm to infer the distinct types of arm movement using wrist-worn inertial sensors with an aim of developing a low-power framework for resource constrained environment of a 200 WSN node [67] in remote monitoring systems. We demonstrate here a completely personalized approach accommodating di?erent levels of impairment and/or rehabilitation status.

4.2 Experimental Protocol

In this investigation, experiments were performed at the University of Southampton (UoS) with four healthy subjects (age range 24 to 40, male, all right arm dominant) and at the Brandenburg Klinik (BBK) with four stroke impaired patients (age range

45 to 73, both sexes, both left and right arm dominant). Experiments were performed within a kitchen attached to the laboratory at UoS and within a treatment centre at BBK under the supervision of the expert physiotherapist members of the research team, using the same set of equipment. An archetypal activity-list (cf. Table 1) emulating the process of *'making-cup-of-tea-coffee'*, was designed. It is a common activity performed in daily life, having repeated occurrences of elementary types of arm movement (*actions*) [23]. In consultation with the clinicians we looked into three elementary types of arm movements:

- *Action A* – Reach and retrieve an object (extension and flexion of the forearm).
- *Action B* – Lift cup to mouth (rotation of the forearm about the elbow).
- *Action C* – Perform pouring or (un)locking action (rotation of the wrist about long axis of forearm).

In principle, these elementary movements constitute a significant proportion of the complex movements performed with the upper limb in daily life and also resemble three of the tasks in the standard Wolf Motor Function Test (WMFT); an established clinical assessment method for quantifying upper extremity motor ability [68]-[70]. There were more number of trials for *Action A* since it is a generic movement performed more frequently in our daily lives.

The activity list in our experiment protocol comprises 20 individual activities including 10 occurrences of *Action A*, and 5 each of *Action B* and *Action C*. There were no restrictions on the various physical factors of the experiment such as the seating position or standing position with respect to the kitchen surface or the time required to complete the actions. The experiment was unconstrained in this manner to ensure a wider range of variability in the data paving the way for a robust arm movement recognition system which will produce acceptable levels of accuracy in a real world application. Healthy participants were requested to perform four repetitions of the

	Activity	Action
1	Fetch cup from desk	A
2	Place cup on kitchen surface	A
3	Fetch kettle	A
4	Pour out extra water from kettle	C
5	Put kettle onto charging point	A
6	Reach out for the power switch on the wall	A
7	Drink a glass of water while waiting for kettle to boil	B
8	Reach out to switch off the kettle	A
9	Pour hot water from the kettle in to cup	C
10	Fetch milk from the shelf	A
11	Pour milk into cup	C
12	Put the bottle of milk back on shelf	A
13	Fetch cup from kitchen surface	A
14	Have a sip and taste the drink	B
15	Have another sip while walking back to desk	B
16	Unlock drawer	C
17	Retrieve biscuits from drawer	A
18	Eat a biscuit	B
19	Lock drawer	C
20	Have a drink	B

Table 4.1: Use case activity list - 'Making-a-cup-of-tea'

activity-list in Table I, at a comfortable speed in a kitchen, with a 10-minute rest period between repetitions. Similarly, the stroke patients were requested to perform 2 repetitions of the same activity-list. The disparity in the number of trials performed by healthy subjects and stroke patients was due to the fact that the latter tend to tire quickly and were asked to perform the tasks only whilst they felt comfortable to do so.

The activity-list was prepared to facilitate the evaluation of the recognition methodology under semi-naturalistic conditions [48], which helps us to verify the results of the unsupervised methodology used in this approach. In previous studies dense sensing-based activity monitoring has been used to detect such holistic activities as 'making tea or coffee' by the use of ambient and wearable sensors to detect user-object interactions within the paradigm of ambient assisted living or smart homes [23]. However, in this work our focus is on detecting the unique number of arm movements performed during the archetypal activity of *'making-cup-of-tea-coffee'*. The start and stop time of the activities were noted down by the researcher observing them as they performed the designated tasks. The corresponding data collected was segmented using the annotations from the researcher and used for the *testing* phase. Since our focus was primarily on developing a low-complexity framework for recognizing the occurrence of these elementary movements, we did not implement an automated segmentation method.

A Shimmer 9DoF wireless kinematic sensor module comprising mutually orthogonal tri-axial accelerometers, rate gyroscopes and magnetometers, was used as the sensing platform [71]. For our experiments we use only the tri-axial accelerometer (range ± 1.5 g) and exclude the gyroscope and the magnetometer. The gyroscope was excluded in view of using a minimal number of sensors and minimizing the associated data processing. Furthermore, gyroscopes also tend to use more electrical power than accelerometers, limiting the operational lifetime in battery operated WBANs [85]. Magnetometers can be affected by the presence of ferromagnetic materials which are expected to be present in the home environment [72]. The dorsal side of the forearm proximal to the wrist on the dominant arm for healthy subjects or impaired arm for stroke patients was chosen as the sensing position. The dorsal side was in contact with the XY plane of the sensor with the X-axis pointing towards the hand and the Z-axis pointing away from the dorsal aspect. Sensor data was collected at a rate of 50 Hz, deemed sufficient for assessing habitual limb movement, which is on the high side when compared to assessing holistic activity as in [39], [43].

4.3 Methodology

The overview of the methodology followed for each subject's data is illustrated in Figure 1 and the primary steps are described in detail in the following sections.

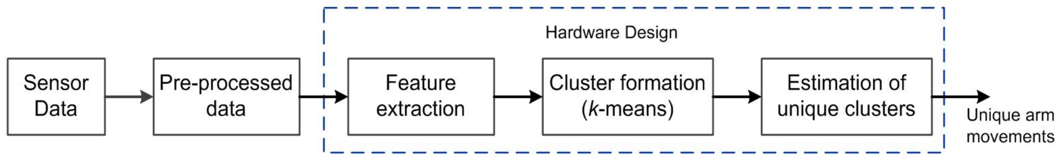


Figure 4.1: Overview of the methodology which includes feature extraction, cluster formation and estimation of correct clusters all performed as part of the hardware design.

4.3.1 Acquisition and Pre-processing

The tri-axial accelerometers located on the wrist transmit data along with a time stamp to a host computer using the Bluetooth wireless transmission protocol. The raw sensor data is band-pass filtered with a 3rd order Butterworth filter having cut-off frequencies of 0.1 Hz and 12 Hz to respectively attenuate the low frequency artefacts and high frequency noise components introduced in the data due to physical effects such as drift [39].

4.3.2 Feature Extraction

Typical feature sets for human activity recognition include statistical functions, time and/or frequency domain features, as well as heuristic features [30]. Each accelerometer data stream (x, y and z) exhibit signal patterns that are distinctive for each of the arm movements, which is characterized by a set of features extracted from the signals [39]. In this investigation, we consider 8 time-domain features which are listed in Table 2.

Although the last two features are usually associated with defining the shape of a

No	Features	Description
1	*standard deviation*	measure of the variability from the mean of the signal
2	*root mean square*	measure of the signal energy normalized by the number of samples
3	*information entropy*	measure of the randomness of a signal [177]
4	*jerk metric*	rms value of the derivative of the data normalized with respect to the maximum value of the integral [5]
5	*absolute difference*	absolute difference between the maximum and the minimum value of a signal
6	*index of dispersion*	ratio of variance to the mean
7	*kurtosis*	measure of the peakedness of a signal assuming a non-Gaussian distribution in the data
8	*skewness*	measure of the symmetry of the data assuming a non-Gaussian distribution in the data [178]

Table 4.2: Use case activity list - 'Making-cup-of-tea-coffee'

probability distribution, they can still be used as classifying features if they routinely return values that distinguish one pattern of data from another. The jerk metric is reflective of the movement fluidity associated with assessing motor functionality in human limb [38]. Hence, we compute 8 one-dimensional features on each individual accelerometer (*acc_x, acc_y, acc_z*) data segment for each movement trial of each subject. The subsequent process of feature selection and cluster formation is performed on the sensor specific feature space (comprising of 30 features), represented as:

$$FS = [f_1\_x \dots f_8\_x, f_1\_y \dots f_8\_y, f_1\_z \dots f_8\_z]$$

computed on each tri-axial data segment (with suffix x, y or z). Each feature was computed on data segments of varying lengths representative of the time taken to complete each movement trial. It illustrates the variations in accelerometer data during a single example of each action for a healthy and a stroke survivor respectively, clearly highlighting the difference in movement profiles characterized by 1) longer time duration for action completion and 2) the less smoothness of the movement in case of the stroke survivors. The features are linearly normalized and used for cluster formation as discussed in section 4.3.

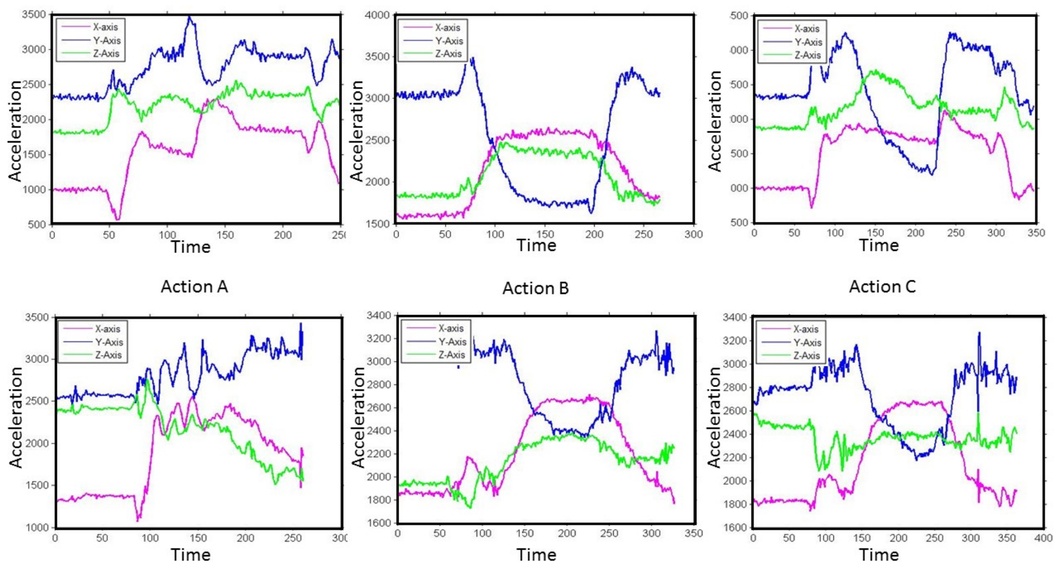


Figure 4.2: Data from a tri-axial accelerometer located on the wrist collected while performing arm actions A, B and C from a healthy subject (upper) and a stroke survivor (lower).

4.3.3 Cluster Formation

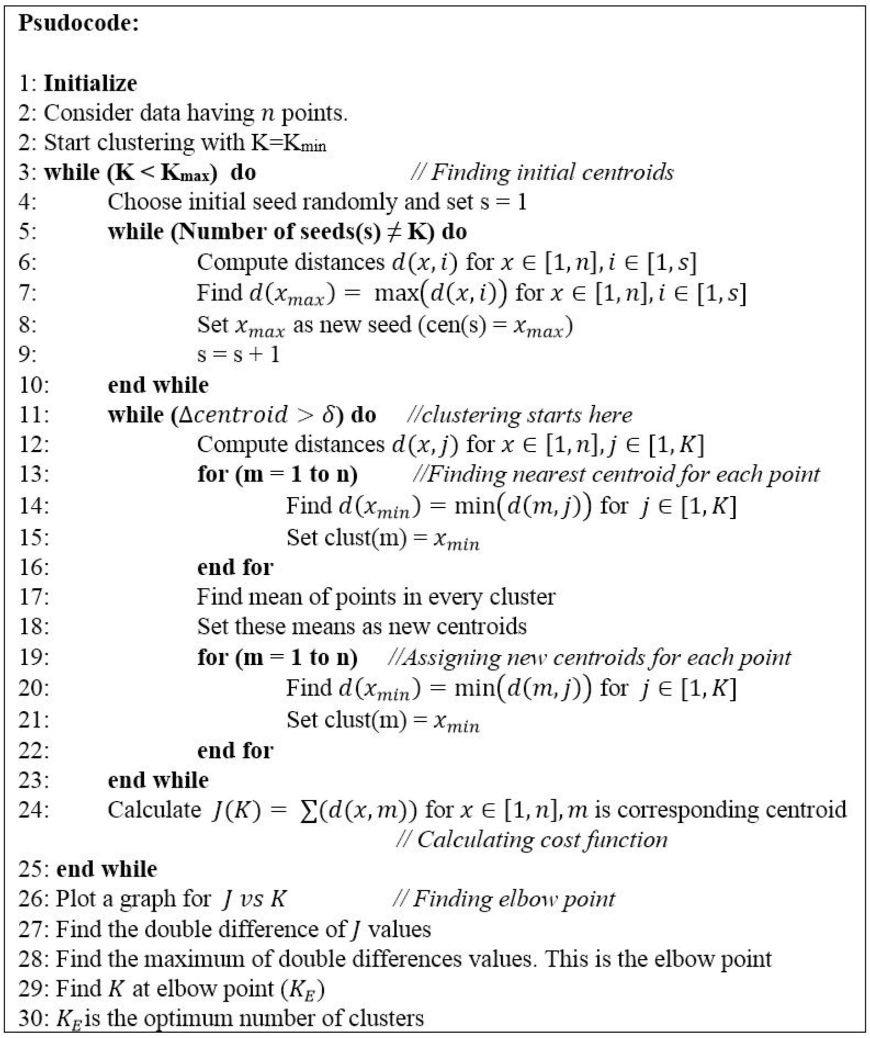
The fundamental concept of cluster analysis is to form groups of similar objects as a means of distinguishing them from each other and can be applied in any discipline involving multivariate data [73]. The k-means algorithm owing to its computational simplicity is a popular clustering technique applied for a wide variety of applications [74]. It is a well-perceived fact in the research community that cluster analysis is primarily used for unsupervised learning where the class labels for the training data are not available. With a given dataset $X = x_i$, $i = 1, \ldots, n$ to be clustered into a set of k clusters, the k-means algorithm iterates to minimize the squared error between the empirical mean of a cluster and the individual data points, defined as the cost function (J),

$$J(\theta, u) = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}(x_i - \theta_j)^2 \tag{4.1}$$

where θ_j is the cluster centre and $u_{ij} = 1$ if x_i lies close to θ_j, or 0 if otherwise [75]. Although k-means is a simple and efficient technique for clustering, it is sensitive to outliers and its efficiency and number of iterations largely depends on seeds. A bad choice of initial centroids can have great impact on performance. In some cases, it may converge to a local minimum. To surmount these issues, k-means is augmented with a simple seeding technique. Initial centroids have been chosen with probability proportional to the overall contribution. After defining initial centroids, the data vectors are assigned to a cluster label depending on how close they are to each centroid. The k centroids are recalculated from the newly defined clusters and the process of reassignment of each data vector to each new centroid is repeated. The algorithm iterates over this loop until the data vectors from the dataset X form clusters and the cost function J is minimized [74]).

4.3.4 Estimation of unique clusters

As mentioned in section I, the estimation of unique clusters on the movement data could be representative of the unique number of hand movements performed in an *out-of-laboratory environment.* The k-means clustering algorithm has been used for unsupervised scenario for cluster estimation in the underlying data. The algorithm requires an initial estimate of the number of clusters, hence the algorithm is operated with a range of initial estimates for k, in which it reasonably lies. In essence, the algorithm iterates over a range of k value. To determine, the optimal cluster value an elbow method is applied where by the rate of change in the cost function (double difference of J) is computed and its maximum value gives the optimal number of



```
Psudocode:

1: Initialize
2: Consider data having $n$ points.
2: Start clustering with K=K_min
3: while (K < K_max)  do                  // Finding initial centroids
4:     Choose initial seed randomly and set s = 1
5:     while (Number of seeds(s) $\neq$ K) do
6:         Compute distances $d(x,i)$ for $x \in [1,n], i \in [1,s]$
7:         Find $d(x_{max}) = \max(d(x,i))$ for $x \in [1,n], i \in [1,s]$
8:         Set $x_{max}$ as new seed (cen(s) = $x_{max}$)
9:         s = s + 1
10:    end while
11:    while ($\Delta centroid > \delta$) do     //clustering starts here
12:        Compute distances $d(x,j)$ for $x \in [1,n], j \in [1,K]$
13:        for (m = 1 to n)     //Finding nearest centroid for each point
14:            Find $d(x_{min}) = \min(d(m,j))$ for $j \in [1,K]$
15:            Set clust(m) = $x_{min}$
16:        end for
17:        Find mean of points in every cluster
18:        Set these means as new centroids
19:        for (m = 1 to n)    //Assigning new centroids for each point
20:            Find $d(x_{min}) = \min(d(m,j))$ for $j \in [1,K]$
21:            Set clust(m) = $x_{min}$
22:        end for
23:    end while
24:    Calculate $J(K) = \sum(d(x,m))$ for $x \in [1,n], m$ is corresponding centroid
                                          // Calculating cost function
25: end while
26: Plot a graph for $J$ vs $K$              // Finding elbow point
27: Find the double difference of $J$ values
28: Find the maximum of double differences values. This is the elbow point
29: Find $K$ at elbow point ($K_E$)
30: $K_E$ is the optimum number of clusters
```

Figure 4.3: pseudocode of the cluster estimation process

clusters (k). This can be observed visually by plotting variation of J with input k range. The location of the bend (elbow) in the plot is the indicator of the appropriate number of clusters [76]. The pseudocode of the cluster estimation process has been presented in Figure 2.

4.3.5 Algorithm to Architecture mapping

Given the requirements for a low-complexity implementation for WSN platforms, in this section, we present an effective mapping of the algorithm to its relevant architecture. As demonstrated in sections 4.2, 4.3 and 4.4, the three significant stages of our methodology are: feature extraction, cluster formation and estimation of unique clusters. These stages involve basic arithmetic operations like addition, subtraction, multiplication, division, square root and logarithm which can be realized using the CORDIC algorithm. CORDIC is a well-researched area and several specialized architectural implementations [77]-[82] of it have been proposed over the years which can be utilized for processing algorithms in low-power WSN nodes. The primary motivation for using the CORDIC algorithm is to explore its different transcendental functions and compute the complex arithmetic operations reusing the same architecture which can be implemented at low-cost with basic shift-add operations of the form $a \pm b.2^{-i}$[77]. Hence, we use CORDIC for the key steps of feature extraction and cluster formation to achieve an optimized low-complexity implementation. Prior to looking into these two architectural aspects, we present a brief overview of the CORDIC fundamentals that have been used for the algorithmic formulation in this exploration.

CORDIC is an iterative algorithm for computing different transcendental functions using 2D vector rotation by employing the following iterative equation:

$$\begin{aligned} x_{j+1} &= x_j - \mu\sigma_j.2^{-j} \\ y_{j+1} &= y_j + \sigma_j.2^{-j}.x_j \\ z_{j+1} &= z_j - \sigma_j.\alpha_j \end{aligned} \tag{4.2}$$

where, $[x_j, y_j]^T, z_j$ and $\sigma_j \epsilon \{1, -1\}$ are the intermediate result vector, the residual angle and the direction of vector rotation at the j-th iteration stage respectively; $\mu \epsilon 1, 0$ being the coordinates of rotation - circular and linear respectively.

Given an input vector $[x_0, y_0]^T$, in different coordinate system, CORDIC operates in two modes viz. rotation $(z_0 \rightarrow 0)$ and vectoring $(y_0 \rightarrow 0)$, for computing a series of

μ	ROTATION MODE ($z_0 \to 0$)	VECTOTING $Y_0 \to 0$
	$x_n = K(x_0 cos\, z_0 - y_0 sin\, z_0)$	$x_n = K\sqrt{x_0^2 + y_0^2}$
1	$y = K(y_0 cos\, z_0 + x_0 sin\, z_0)$	$y_n = 0$
	$z_n = 0$	$z_n = z_0 + tan^{-1}(y_0/x_0)$
	$x_n = x_0$	$x_n = x_0$
0	$y_n = y_0 + x_0 z_0$	$y_n = 0$
	$z_n = 0$	$z_n = z_0 + (y_0/x_0)$

Table 4.3: Generalized CORDIC algorithm in two co-ordinate systems

transcendental functions as shown in Table 3. For a detailed survey of CORDIC please refer to [83]. These forms of the transcendental functions, more specifically, those generated by the vectoring operation of CORDIC in different coordinate systems could be adapted for computing the target features and the clustering. For convenience we use the operators Vec$_c$, and Vec$_l$ to represent vectoring operation of CORDIC in circular and linear coordinate systems respectively. In our formulation, the input dataset is represented by x_{di}, where $i \epsilon 0, 1, 2 \ldots n-1$ and x_i is the output of vectoring CORDIC operation on $x_{d(i-1)}$ data sample. With this convention, the formulation of the features and cluster formation in terms of CORDIC operation is described below.

Feature Extraction

As mentioned in section 4.2, there are eight time-domain features to be computed on the on the accelerometer data. A detailed architecture and implementation of seven of these eight features have already been presented in [35] highlighting the merits of using the CORDIC algorithm for formulating the features compared to other implementations. In this design, we re-use the feature extraction engine proposed in [35] and present the formulation of the feature: jerk metric (not shown in [35]).

The jerk metric (as illustrated in Table 2) characterizes the average rate of change of acceleration in a movement. It is calculated as the *rms* value of the derivative of the acceleration (jerk) normalized by the maximum value of the integral (velocity)

[43] as shown in (2).

$$jerkmetric = \frac{rms\left[\frac{d(d_{si})}{dt}\right]}{max[\int(d_{si})dt]} \tag{4.3}$$

Since the data samples are equally spaced due to the constant sampling frequency, the first derivative is computed as the difference of the consecutive data samples using a subtractor. The integral of the data is computed using trapezoidal integration which involves the addition of the consecutive data samples and a divide by 2 (implemented with a one-bit right shift). From (2), it can be deduced that the *rms* of the first derivative of the data samples ($\dot{d_{si}}$) can be computed using the operator Vec_c, which is shown in (3).

$$rms = \frac{1}{\sqrt{n}}\left[\prod_{i=0}^{n-1} Vec_c[d_i \;\; \dot{d_{si}}]^T\right]_x \tag{4.4}$$

The samples $\dot{d_{si}}$ are fed in the y input of the CORDIC while the x-component of the output is fed back to the x-component of its input. Therefore, at every clock cycle as the new data sample d_{si} arrives, the computed x-component of the CORDIC is given by:

$$d_i = K\sqrt{d_{s0}^2 + d_{s1}^2 + \cdots + d_{s(i-1)}^2} \tag{4.5}$$

One point to be noted here is that after every complete CORDIC operation the x-component of the output is scaled with the scale factor K. If uncompensated, feeding back this result into the x-component of the CORDIC input will result in accumulation of this scale factor corresponding to each d$_{si}$ and therefore (4) will not hold true. To avoid this problem, after every complete CORDIC operation (comprising of N stages) with a set of input data, the scale factor compensation step needs to be invoked before feeding this output to the x-input of the CORDIC for the next iteration. With this scale factor compensation step in place, after n number of operations the final result at the x-output of the CORDIC needs to multiplied with $1/\sqrt{n}$ to obtain the true result of rms. However, since n is a fixed number, the value of $1/\sqrt{n}$ could be pre-computed and finally multiplied with the CORDIC output using a reduced complexity fixed-number multiplier or multiplier-less shift-and-add technique.

The jerk metric is finally computed using the CORDIC operator Vec_l as shown in (5). Referring to Table 3, $max(\int d_{si})$ and $rms[\dot{d_{si}}]$ are set as the x_0 and y_0 inputs to the CORDIC, operating in vectoring mode in the linear coordinate system.

$$jerkmetric = \left[Vec_l[max(\int d_{si}) \;\; rms\left[\dot{d_{si}}\right]]^T \right]_z \tag{4.6}$$

The implementation includes 1 subtractor and CORDIC for computing $rms[\dot{d_{si}}]$ and 1 adder for computing $(\int d_{si})$ by trapezoidal integration. Finally, CORDIC (Vec_l) is used for computing the value of the feature. The computation of the *jerk metric* is dependent on the *rms* of the derivative and maximum of the integral taking $(n+1)$ cycles. Here, we consider n as 256 data samples, representative of a movement for approximately 5 seconds (@50 Hz) hence facilitating multiplier-less shift-and-add operation.

Cluster Formation

The architecture of the proposed CORDIC based k-means clustering engine is shown in Figure 4. The input data are stored in a memory bank and are transmitted to different blocks via control unit (CU). The first step of this methodology, calculation of initial centroids, is done by the seeds calculation unit. The Euclidean distance is calculated in distance unit (DU) using low-complexity CORDIC vectoring module. The distance from each point to the centroids are sent to a comparator block to identify the cluster to which it belongs. Once the clustering is done, centroid calculation block will be activated to compute the new centroids. If these new centroids differ significantly from the previous iteration values, then clustering is repeated else they are sent to the output. The CU governs the data flow among all the modules. The proposed engine utilizes CORDIC to compute Euclidean distance between two points, which is a metric to compute the clusters and has been explained in the following section.

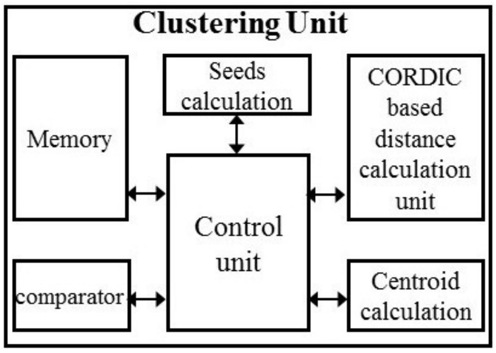


Figure 4.4: Architecture for the Clustering unit.

Distance calculation using CORDIC

In this section, we present the architecture for computing the Euclidean distance using CORDIC. The Euclidean distance is used to compute the squared distance between the vectors x_i and the mean of each cluster h_j. The distance computation is performed on the feature vectors computed from the accelerometer data which are in 3-dimensional space. Two CORDIC stages are required to compute 3-dimensional distance between two points $(x_1, x_2, x_3), (y_1, y_2, y_3)$:

$$d_{xy} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}.$$

With inputs as $(x_1 - y_1)$, $(x_2 - y_2)$ to first level of CORDIC vectoring mode (Vec_l), the x-output is represented as $vec_x^{l1}((x_1 - y_1), (x_2 - y_2))$ which is the computation of $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ [77]. This x-output when fed to the next CORDIC level as one input, with $(x_3 - y_3)$ as the second input, we achieve the desired 3-dimesional distance $(vec_x^{l2}(vec_x^{l1}((x_1 - y_1), (x_2 - y_2)), (x_3 - y_3))$. The 3D distance measurement unit is shown in Figure 5(a), where the two stages are executed sequentially and hence the same CORDIC unit can be reused only at the expense of two multiplexers (Fig 5b).

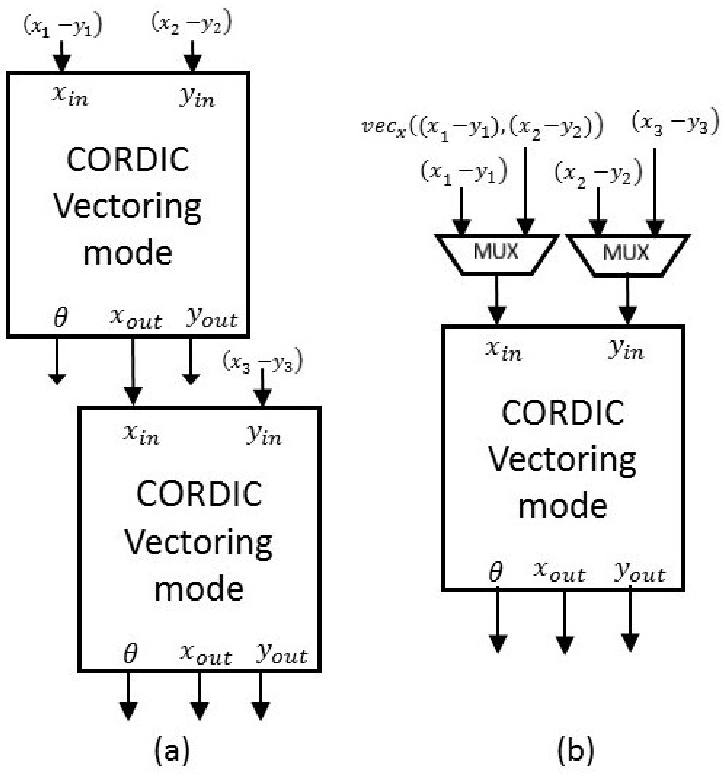


Figure 4.5: CORDIC based distance measurement (a) 3-D vectoring unit. (b) 3-D Multiplexed architecture.

Estimation of unique clusters

In this section the process for estimating the unique number of clusters for a given range of values for $'k'$ has been highlighted. This module is activated after the clustering unit has completed the computation for a particular value of $'k'$. A comparator checks for the value of K to compare with the maximum attainable value K_{max}. If K value is less than K_{max}, it is incremented and Clustering unit runs with new K value. After finding the J values for all the possible K, they are sent to Difference Unit. Difference Unit finds the double difference of all the J values. Peak detector finds K value at which peak obtained. This represents the optimal number of clusters formed, which is representative of the unique number of movements performed by the subjects over a trial period. The architecture is designed to reuse one clustering unit for differing values of K ($< K_{max}$).

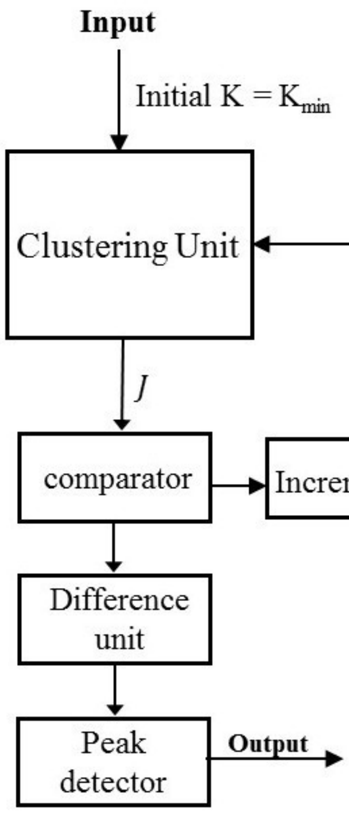


Figure 4.6: Complete architecture for Estimating Unique Clusters.

4.4 Results and Analysis

This section presents the results for simulation of the designed hardware architecture on the data collected during the experiments. A hardware complexity analysis has also been presented here.

4.4.1 Synthesis and Verification

The proposed architecture was coded in Verilog HDL and synthesized using Synopsys Design Compiler (DC) and the place an route was performed using IC Compiler (ICC) using $130nm$ standard cell CMOS technology. The synthesized core area and power consumption of the proposed architecture are $0.61mm^2$ and $9.21mW$ at $100MHz$ frequency for $VDD = 1.2V$ using $130nm$ technology. Since the application area we consider is that of human activity recognition where these time domain features are generally extracted from kinematic data sampled at very low frequencies ($20 \sim 50Hz$) from body-worn inertial sensors, we synthesize our design at $50Hz$.

To verify its functional correctness, the design was evaluated on the dataset collected during the experiments. For the healthy subjects we have a total of 640 datasets (8 features computed on 80 movement trials - 40 of Action A, 20 of B and 20 of C), where each dataset was restricted to have 256 samples using an interpolation/ extrapolation. This was done to represent the number of samples on a dyadic scale, hence any division involving the total number of samples could be implemented using a reduced complexity shift operation. Similarly, for the stroke survivors we have 320 datasets (8 features computed on 40 movement trials - 20 of Action A, 10 of B and 10 of C) of 256 samples each. The HDL outputs were compared against a Matlab implementation to check for matching results. As our experiment of *making-cup-of-tea-coffee* has 3 elementary actions (cf. Table 1), the desired outcome of the proposed methodology is 3. It can be observed from the results in Table 4 that the number of clusters reported for the healthy subjects is 3 whereas for stroke survivors it is found to be on the higher side i.e. 7, showing the inter-subject variability involved in performing the same type of movements.

While we expect there to be three unique number of clusters in relation to the archetypal activity performed, however there are potential reasons which lead to the detection of higher number of clusters for stroke survisors. Firstly, the movements 6, 8, 10 and 12 (cf. Table 2) involved reaching out for the switch and the milk bottle kept sideways and at a different altitude. Furthermore, Action C also involves the component movements - reaching out for the object, performing a rotation of the wrist and retrieving back. For survivors 2, 3 and 4 (who were at their early stage of their rehabilitation as assessed by clinicians), a rotation task was quite difficult to achieve fully. The kinematic characteristics pertaining to these particular variable movements have caused the formation of more number of clusters. An illustrative J *vs* K plot for one healthy subject and a stroke survivor has been shown in Fig 6, highlighting the variation of change of slope of J for different values of K (cf. section 4.4). The maximum change of slope occurs for $K = 3$ (for healthy) and $K = 5$ (for stroke) highlights the estimation of unique clusters.

Subject No.	Number of Clusters	
	Healthy subject	Stroke Survivors
1	3	5
2	3	6
3	3	6
4	3	7

Table 4.4: Summary of distinct movement estimated for healthy subjects and stroke survivors.

4.4.2 Hardware Complexity Analysis

In this section we present a hardware complexity analysis of the proposed cluster implementation in terms of total Transistor Count (TC). Throughout the hardware complexity analysis, we keep a generalized view of word-length b and follow the same procedure used in [81] and [82]. The distance computation in k-means Clustering using CORDIC is an iterative procedure, and therefore the same hardware can be reused for the subsequent iterations as well as for successive stages of CORDIC in Vectoring Mode. Therefore, we consider only one single CORDIC block. Computation of distance between 2 n-dimensional points$(x_1, x_2, x_3), (y_1, y_2, y_3)$ using the

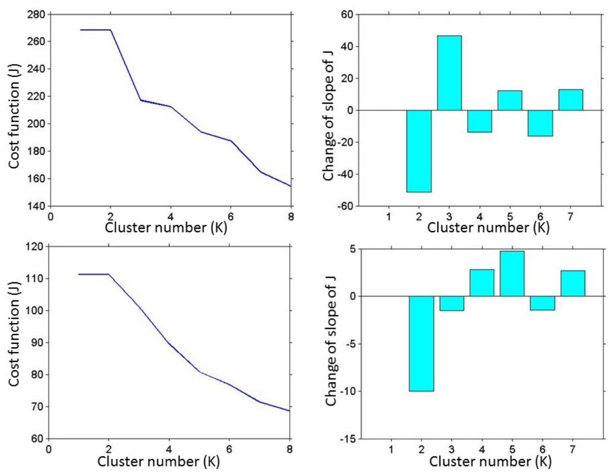


Figure 4.7: J vs K plot highlighting the optimal number of clusters ($K = 3$ and 5) estimated by the double difference of J for each value of K for healthy subject (a,b) and stroke survivor (c,d) respectively.

conventional method, i.e. $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$ requires 3 squaring operations, 2 addition operations and 1 square-root operation. To provide a comparison on a uniform platform we consider only Ripple Carry Adder (RCA), Conventional Array Multiplier (CAM) and non-restoring iterative cellular square rooter (SQRT) as the means of implementing the arithmetic operations. One bbit RCA requires b full adders (FA) (in a simplified view) [33] and b * b CAM requires b(b -2) FA plus b half adders (HA) and b2 AND gates [33]. Similarly, one b-bit SQRT needs 0.125 * (b + 6)b FA and XOR gates [32]. Considering one FA cell requires 24 transistors, one HA cell and one 2-input XOR gate consist of 12 transistors and a 2-input AND gates consists of six transistors [33], we can calculate $TC_A = 24b$, $TC_M = 6b(5b - 6)$, $TC_{SQRT} = 18(b/2 + 1)(b/2 + 3)$, where $TC*$ are the transistor counts for RCA, CAM and SQRT respectively.

Following the same procedure used in [81] and [82], savings in terms of arithmetic operations for distance computation in different dimensions without using CORDIC are computed. For 3-Dimensional distance computation, 1 SQRT, 3 CAMs and 2 RCAs are required. All the transistors required for these operations are considered saved since they will not be required when using the proposed CORDIC based architecture as the same CORDIC block used for pre-processing can be reused for this purpose as well. Therefore, the total Transistor Count (TC) computed here will be the Transistor Saving (TS), given by: $TS_D = 3*TC_M + 2*TC_A + TC_{SQRT}$ Expressing TSD 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 [68]. The figure below shows the variation of TSPW as a function of b for the proposed architecture over the conventional ones, for different word lengths ($4 <= b <= 32$).

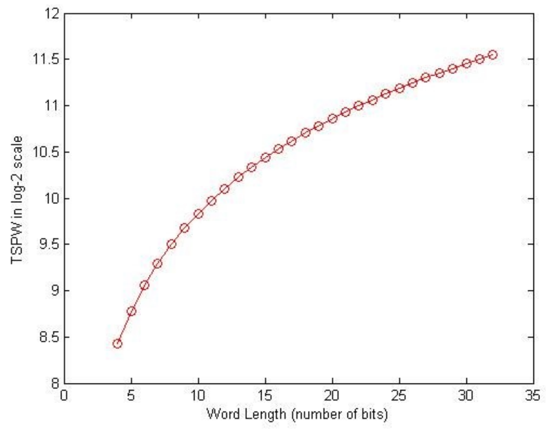


Figure 4.8: Variation of Transistor Savings Per Word-length for different Word length

Chapter 5

Conclusion and Future work

In this paper we introduced coordinate rotation concept into FICA and based on this concept we proposed a generalized recursive reconfigurable nDFICA algorithm and architecture using 2DFICA as the fundamental core. The proposed algorithm has been proved very effective over the conventional one in terms of 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 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 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 FICA and have the potential to open up new application domain of CORDIC. Further, a hybrid architecture has been proposed by combining FICA with cross product for high speed applications.

We have presented the architecture and implementation of a low-power framework for estimating the unique category (or number) of arm movements performed in an *out-of-laboratory* environment. For this investigation, data was collected using a wrist-worn tri-axial accelerometers from four healthy subjects and four stroke survivors as they performed an archetypal activity of *making-a-cup-of-tea-coffee* in the kitchen. Our study focusses on the key area of daily life activity monitoring using only inertial sensors and hence having no information on the type/number of activities performed (i.e. class labels for verification). Given such a scenario, unsupervised approach of k-means clustering performed on the kinematic data and subsequent estimation of the unique number of clusters would help to infer on the unique category of movements being performed by a subject under clinical observation. A longitudinal study enumerating the number of unique movements performed over time could act as a rehabilitation indicator for patients suffering from less fluidic movements in neurodegenerative diseases. Furthermore, we look into the aspect of a low-complexity algorithm-to-architecture mapping, which would enable data processing on the sensor node itself, aiding an optimized low-power implementation. Our results show that the unique number of clusters (or movements) estimated by the algorithm are in agreement to actual number of movements (Actions A, B and C) for healthy subjects. However, for the stroke survivors, the number of optimal clusters reported is on the higher side. This is primarily due to the variations inherent within the movement profiles of the subject population at differing stages of their rehabilitation. Hence, we believe, as the patient improves over a period of time, the unique number of movements reported would stabilise or tend towards being constant. This can be further verified by a clinical intervention. We aim to perform a detailed longitudinal study in the near future observing a set of patients at regular intervals (e.g. weekly) over a period of time (e.g. for 6 months). In this work we have presented a *proof-of-concept* implementation of the k-means clustering algorithm and double differentiation method to estimate the optimal number of clusters and our results demonstrate - 1) difference in the movements of healthy and stroke survivors and 2) low-power implementation ($0.5mW$ of dynamic power@ $50Hz$) which can be used within a sensor node.

In FICA, for orthogonalization step, multipliers has been used extensively. An optimized architecture can be proposed by using existing CORDIC for Gram-Schmidt orthogonalization. FICA computational time depends on initial random vectors for some extent. There is a scope to find a way to choose initial random values, so that FICA will converge with in less iterations. There by processing delay can be further decreased. FICA can separate mixed signals which has at most one Gaussian signal. This algorithm can be improved by using different cost-function (with additional constraints). K-Means can be make reconfigurable so that same core engine works for different K values in unsupervised case.

References

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

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

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

[4] J. Gotze and G. J. Hekstra, An Algorithm and Architecture based on Orthonormal -rotations for Computing the Symmetric EVD, Integration, The VLSI Journal, vol. 20, pp. 21-39, 1995.

[5] D. Estrin, D. Culler, K. Pister and G. Sukhatme, Connecting the Physical Worldwith Pervasive Networks, IEEE Pervasive Computing, vol. 1, no. 1, pp. 59-69,2002.

[6] P. Celka, R. Vetter, P. Renevey, C. Verjus, V. Neuman, J. Luprano, J. Decotignieand C. Piguet, Wearable Biosensing: Signal Processing and Communication,Journal of Telecommunications and Information Technology, vol. 4, pp. 90-104,2005.

[7] Altera, DE2-115 User Manual, 2012. [Online]. Available: ftp://ftp.altera.com/up/pub/Altera Material/12.1/Boards/DE2-115/DE2 115 User Manual.pdf. [Accessed: 0106-2014].

[8] SerialPort Class. [Online]. Available: http://msdn.microsoft.com - /enus/library/system.io.ports.serialport%28v=vs.110%29.aspx.[Accessed: 06-2014].

[9] S. Theodoridis and K. Koutroumbas, Pattern Recognition, 4th ed., Elsevier, pp. 30-31 and pp. 280-288, 2008.

[10] A. Acharyya, K. Maharatna, and B. M. Al-Hashimi, Co-ordinate rotation based low complexity 2-D FastICA algorithm and architecture, in Proc. IEEE Int. Conf. Green Circuits Syst., Shanghai, China, Jun. 2123, 2010, pp. 6064.

[11] J. E. Volder, The CORDIC trigonometric computing technique, IRE Trans. Electron. Comput., vol. EC-8, pp. 330334, 1959.

[12] J. S. Walther, A unified algorithm for elementary functions, in Proc. Spring Joint Comput. Conf., 1971, pp. 379385.

[13] Y. H. Hu, CORDIC based VLSI architecture for digital signal processing, IEEE Signal Process. Mag., pp. 16 35, Jul. 1992.

[14] T.-W. Chen, C.-H. Sun, J.-Y. Bai, H.-R. Chen, and S.-Y. Chien, Architectural analyses of K-means silicon intellectual property for image segmentation, in Proc. IEEE Int. Symp. Circuits Syst., May 2008, pp. 25782581.

[15] T.-W. Chen and S.-Y. Chien, Bandwidth adaptive hardware architecture of K-means clustering for video analysis, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst., vol. 18, no. 6, pp. 957966, Jun. 2010.

[16] T.-W. Chen, Y.-L. Chen, and S.-Y. Chien, Photo retrieval based on spatial layout with hardware acceleration for mobile devices, IEEE Trans. Mobile Comput., vol. 10, no. 11, pp. 16461660, Nov. 2011.

[17] T.-W. Chen, C.-H. Sun, H.-H. Su, S.-Y. Chien, D. Deguchi, I. Ide, and H. Murase, Power-efficient hardware architecture of K-Means clustering with Bayesian-information-criterion processor for multimedia processing applications, IEEE Journal on Emerging and Selected Topics in Circuits and Systems , vol. 1, no. 3, pp. 357368, Sep. 2011.

[18] Chen, T. W., and Ikeda, M. (2013). Design and Implementation of Low-Power Hardware Architecture With Single-Cycle Divider for On-Line Clustering Algorithm. IEEE Transactions on Circuits and Systems I: Regular Papers, 60(8), 2165-2176.

[19] A. Acharyya, K. Maharatna, and B. M. Al-Hashimi, Hardware reduction methodology for 2-dimensional Kurtotic FastICA based on algorithmic analysis and architectural symmetry, in Proc. IEEE Workshop Signal Process. Syst., Oct. 2009, pp. 6974.

[20] J. C. Majithia, Pipeline array for square-root extraction, Electronics Letters, vol. 9, no. 1, pp. 45, 1973.

[21] A. Acharyya, K. Maharatna, B. M. Al-Hashimi, and S. R. Gunn, Memory reduction methodology for distributed arithmetic based DWT/IDWT0 exploiting data symmetry, IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 4, pp. 285289, Apr. 2009.

[22] M.S. Raisinghania, A. Benoit, and J. Ding, Ambient intelligence: Changing forms of human-computer interaction and their social implications, Texas Digital Library, vol. 5, no.4, 2004.

[23] L. Chen, J. Hoey, C.D. Nugent, D.J. Cook, and Z. Yu, Sensor-Based activity recognition, IEEE Trans. Syst. Man And Cybernetics, vol. 42, no. 6, pp. 790-808, Nov. 2012.

[24] C. Zhu and W. Sheng, Motion- and location-based online human daily activity recognition, Pervasive and Mobile Computing, vol. 7, no. 2, pp. 256-269, Apr. 2011.

[25] J.L. Semmlow, Biomedical and Medical Image Processing, 2nd ed., CRC Press, pp. 361-398, 2008.

[26] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea, Machine recognition of human activities, IEEE Trans. Circ. Syst. Video Technol., vol. 18, no. 11, pp. 1473-1488, Nov. 2008.

[27] A. Hadjidj, A. Bouabdallah, and Y. Challal, Rehabilitation supervision using wireless sensor networks, in Proc. IEEE WoWMoM, Luca, Italy, pp.1-3, Jun. 2011.

[28] B. Naja, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C.J. Bula, and P. Robert, Ambulatory system for human motion analysis using a kinematic sensor: Monitoring of daily physical activity in the elderly, IEEE Trans. Biomed. Eng., vol. 50, no.6, pp. 711-723, Jun. 2003.

[29] M. Ermes, J. Parkka, J. Mantyjarvi, and I. Korhonen, Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions, IEEE Trans. Inf. Technol. Biomed., vol. 12, no.1, pp. 20-26, Jan. 2008.

[30] K. Altun, B. Barshan, O. Tuncel, Comparative study on classifying human activities with miniature inertial and magnetic sensors, Pattern Recognition, vol. 43, no. 10, pp. 3605-3620, Oct. 2010.

[31] J.J. Guiry, P.V.D. Ven, J. Nelson, L. Warmerdam and H. Riper, Activity recognition with smartphone support, Medical Engineering & Physics, vol. 36, no. 6, pp. 670-675, Jun. 2014.

[32] J. C. Majithia, Pipeline array for square-root extraction, Electronics Letters, vol. 9, no. 1, pp. 45, 1973.

[33] A. Acharyya, K. Maharatna, and B. M. Al-Hashimi, Hardware reduction methodology for 2-dimensional Kurtotic FastICA based on algorithmic analysis and architectural symmetry, in Proc. IEEE Workshop Signal Process. Syst., Oct. 2009, pp. 6974.

[34] A Acharyya, K Maharatna, Bashir M. Al-Hashimi and Jeff Reeve, Coordinate Rotation Based Low Complexity N-D FastICA Algorithm and Architecture, IEEE Transactions On Signal Processing, Vol. 59, No. 8, August 2011. doi: 10.1109/TSP.2011.2150219

[35] D. Biswas, K. Maharatna A CORDIC-based Low-power Statistical Feature Computation Engine for WSN Applications

[36] Bhardwaj S, Jadhav P, Adapa B, Acharyya A, Naik G R Online and automated reliable system design to remove blink and muscle artefact in EEG

[37] Vemishetty N, Jadhav P, Adapa B, Acharyya A, Pachamuthu R, Naik, G R Affordable low complexity heart/brain monitoring methodology for remote health care

[38] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, Activity classification using realistic data from wearable sensors, IEEE Trans. Inf. Technol. Biomed., vol.10, no.1, pp.119-128, Jan. 2006.

[39] O. Banos, M. Damas, H. Pomares, A. Prieto, and I. Rojas, Daily living activity recognition based on statistical feature quality group selection, Expert Systems with Applications, vol. 39, no. 9, pp. 8013-8021, Jul. 2012.

[40] A. Fleury, M. Vacher, and N. Noury, SVM-based multimodal classication of activities of daily living in health smart homes: Sensors, algorithms, and rst experimental results, IEEE Trans. Inf. Technol. Biomed., vol.14, pp. 274-283, Mar. 2010.

[41] D. Fuentes, L. Gonzalez-Abril, C. Angulo, and J.A. Ortega, Online motion recognition using an accelerometer in a mobile device, Expert Systems with Applications, vol. 39, no. 3, pp. 2461-2465, Feb. 2012.

[42] Maurer, U., Smailagic, A., Siewiorek, D., & Deisher, M. 2006. Activity recognition and monitoring using multiple sensors on different body positions. In IEEE Proceedings on the International Workshop on Wearable and Implantable Sensor Networks, 3(5).

[43] S. Chernbumroong, S. Cang, A. Atkins, and H. Yu, Elderly activities recognition and classification for applications in assisted living, Expert Systems with Applications, vol. 40, no. 5, pp. 1662-1674, Apr. 2013.

[44] H. Junker, O. Amft, P. Lukowicz, and G. Trster, Gesture spotting with body-worn inertial sensors to detect user activities, Pattern Recognition, vol. 41, no. 6, pp. 2010-2024, Jun. 2008.

[45] N. Bicocchi, M. Mamei, F. Zambonelli, Detecting activities from body-worn acclerometers via instance-based algorithms, Pervasive and Mobile Computing, vol. 6, no. 4, pp. 482-495, Aug. 2010.

[46] N. Alshurafa, W. Xu, J. J. Liu et al., Designing a robust activity recognition framework for health and exergaming using wearable sensors, IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 5, pp. 16361646, 2014.

[47] Nguyen A, Moore D and McCowan I 2007 Unsupervised clustering of free-living human activities using ambulatory accelerometry 29th Annual Conf. of the IEEE Engineering in Medicine and Biology Society (Lyon) pp 48958

[48] L. Bao and S. Intille, Activity recognition from user-annotated acceleration data, in Proc. 2nd Int. Conf. Pervasive Comput., 2004, pp. 1-17.

[49] S. Antifakos, F. Michahelles, and B. Schiele, Proactive instructions for furniture assembly, in UbiComp 2002: Ubiquitous Computing, Springer, 2002, pp. 351360.

[50] G. Fang, W. Gao, and D. Zhao, Large vocabulary sign language recognition based on hierarchical decision trees, in Proceedings of the 5th international conference on Multimodal interfaces, 2003, pp. 125131.

[51] P. Lukowicz, J. A. Ward, H. Junker, M. Stger, G. Trster, A. Atrash, and T. Starner, Recognizing workshop activity using body worn microphones and accelerometers, in Pervasive Computing, Springer, 2004, pp. 1832.

[52] H. Junker, O. Amft, P. Lukowicz, and G. Trster, Gesture spotting with body-worn inertial sensors to detect user activities, Pattern Recognition, vol. 41, no. 6, pp. 20102024, 2008.

[53] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Trster, Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection, in Wireless sensor networks, Springer, 2008, pp. 1733.

[54] G. Bailador, D. Roggen, G. Trster, and G. Trivio, Real time gesture recognition using continuous time recurrent neural networks, in Proceedings of the ICST 2nd international conference on Body area networks, 2007, p. 15.

[55] S. Wang, J. Yang, N. Chen, X. Chen, and Q. Zhang, Human activity recognition with user-free accelerometers in the sensor networks, in Neural Networks and Brain, 2005. ICNN&B05. International Conference on, 2005, vol. 2, pp. 12121217.

[56] Biswas, D., Cranny, A., Gupta, N., Maharatna, K., Achner, J., Klemke, J., ... & Ortmann, S. (2015). Recognizing upper limb movements with wrist worn inertial sensors using k-means clustering classification. Human movement science, 40, 59-76.

[57] J. Ye, S. Dobson, and S. McKeever, Situation identification techniques in pervasive computing: A review, Pervasive and Mobile Computing, vol. In Press., 2011.

[58] J.-Y. Yang, J.-S. Wang, Y.-P. Chen, Using acceleration measurements for activity recognition: an effective learning algorithm for constructing neural classifiers, Pattern Recognition Letter 29 (16) (2008) 22132220.

[59] M. Perkowitz, M. Philipose, K. Fishkin, D.J. Patterson, Mining models of human activities from the web, in: WWW04: Proceedings of the 13th International Conference on World Wide Web, ACM, New York, NY, USA, 2004, pp. 573582.

[60] P. Palmes, H.K. Pung, T. Gu, W. Xue, S. Chen, Object relevance weight pattern mining for activity recognition and segmentation, Pervasive Mobile Computing 6 (1) (2010) 4357.

[61] T. Gu, S. Chen, X. Tao, J. Lu, An unsupervised approach to activity recognition and segmentation based on object-use fingerprints, Data and Knowledge Engineering 69 (6) (2010) 533544.

[62] R. Hamid, S. Maddi, A. Johnson, A. Bobick, I. Essa, C. Isbell, A novel sequence representation for unsupervised analysis of human activities, Artificial Intelligence 173 (14) (2009) 12211244.

[63] O. Brdiczka, J. Maisonnasse, P. Reignier, J.L. Crowley, Detecting small group activities from multimodal observations, International Journal of Applied Intelligence 30 (1)(2009) 4757.

[64] P. DUrso, R. Massari, Fuzzy clustering of human activity patterns, Fuzzy Sets and Systems, vol. 215, pp. 29-54, 2013.

[65] Wang, Z.L., Jiang, M., Hu, Y.H. and Li, H.Y. (2012) An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors. IEEE Trans. Inf. Technol. Biomed., 16, 691699.

[66] Robards, M. and Sunehag, P. (2009). Semi-Markov kmeans clustering and activity recognition from body-worn sensors. In International Conference on Data Mining.

[67] K. Maharatna, E. B. Mazomenos, J. Morgan, and S. Bonfiglio, Towards the development of next-generation remote healthcare system: Some practical considerations, in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), Seoul, pp. 14, May. 2012.

[68] S. L. Wolf, H. Newton, D. Maddy, S. Blanton, Q. Zhang, C. J. Winstein, D. M. Morris, and K. Light, The Excite trial: relationship of intensity of constraint

induced movement therapy to improvement in the wolf motor function test, Restorative Neurology and Neuroscience, vol. 25, no. 5, pp. 549562, Mar. 2007.

[69] S. L. Wolf, J. P. McJunkin, M. L. Swanson, and P. S. Weiss, Pilot normative database for the wolf motor function test, Archives of Physical Medicine and Rehabilitation, vol. 87, no. 3, pp. 443445, Mar. 2006.

[70] D. M. Morris, G. Uswatte, J. E. Crago, E. W. Cook III, and E. Taub, The reliability of the Wolf Motor Function Test for assessing upper extremity function after stroke, Archives of Physical Medicine and Rehabilitation, vol. 82, no. 6, pp. 750755, Jun. 2001.

[71] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, Shimmer A wireless sensor platform for noninvasive biomedical research, IEEE Sensors Journal, vol. 10, no. 9, pp.15271534, Sep. 2010.

[72] C. Kendell and E.D. Lemaire, Effect of mobility devices on orientation sensors that contain magnetometers, J. Rehab. Res. Dev., vol. 46, no. 7, pp. 957-962, 2009.

[73] T.Warren Liao, Clustering of time series data-a survey, Pattern Recognition, vol. 38, no. 11, pp. 1857-1874, Nov. 2005.

[74] S. Theodoridis and K. Koutroumbas, Pattern Recognition, 4th ed., Elsevier, pp. 30-31 and pp. 280-288, 2008.

[75] J. Mao, A.K. Jain, A self-organizing network for hyperellipsoidal clustering (HEC), IEEE Trans. Neural Networks, vol. 7, no.1, pp. 16-29, Jan. 1996.

[76] Ketchen, D. J.f & Shook, C. L. 1996. The application of cluster analysis in strategic management research: An analysis and critique. Strategic Management Journal, 17: 441-485

[77] P.K. Meher, J. Valls, T.B. Juang, K. Sridharan, K. Maharatna, 50 Years of CORDIC: lgorithms, Architectures, and Applications, IEEE Trans. Circuits and Systems I, vol. 56, no. 9, pp. 1893-1907, Sept. 2009.

[78] C.Y. Kang and E. Swartzlander, Digit-Pipelined Direct Digital Frequency Synthesis Based on Differential CORDIC, IEEE Trans. Circuits and Systems-I, vol. 53, no. 5, pp. 1035-1044, May. 2006.

[79] E. Grass, B. Sarker, and K. Maharatna, A Dual-Mode Synchronous/Asynchronous CORDIC Processor, in Proc. Eighth Int. IEEE Symposium on Asynchronous Circuits and Systems, pp. 7683, Apr. 2002.

[80] K. Kota and J.R. Cavallaro, Numerical Accuracy and Hardware Tradeoffs for CORDIC Arithmetic for Special-Purpose Processors, IEEE Trans. Computers, vol. 42, no. 7, pp. 769-779, Jul. 1993.

[81] L. Vachhani, K. Sridharan, and P.K. Meher, Efficient CORDIC Algorithms and Architectures for Low Area and High Throughput Implementation, IEEE Trans. Circuits and Systems-II, vol. 56, no. 1, pp. 2385-2396, Jan. 2009.

[82] C.H. Lin and A.Y. Wu, Mixed-Scaling-Rotation CORDIC (MSR-CORDIC) Algorithm and Architecture for High-Performance Vector Rotational DSP Applications, IEEE Trans. Circuits and Systems-I, vol. 52, no. 11, pp. 2385-2396, Nov. 2005.

[83] M.D. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann, pp. 549 607.

[84] D. Biswas, A. Cranny, A. F. Rahim, N. Gupta, K. Maharatna, N. Harris and S. Ortmann, On the data analysis for classification of elementary upper limb movements, Biomedical Engineering Letters, Springer, Vol. 4, pp. 403-413, 2014.

[85] D. Biswas *et. al.*, Recognition of elementary arm movements using orientation of a tri-axial acelerometer located near the wrist, Physiological Measurement, IOP, Vol. 35, no. 9, pp. 1751-1768, 2014.