UNIVERSITY OF SOUTHAMPTON

Resource Constrained Signal Processing Algorithms and Architectures

by

Amit Acharyya

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

March 2011

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Resource Constrained Signal Processing Algorithms and Architectures

by Amit Acharyya

Emerging applications with limited resources in Wireless sensor networks require sophisticated signal processing algorithms and their efficient hardware implementation. Direct mapping of traditional signal processing algorithms to hardware may not be suitable for such resource constrained applications. It is therefore necessary to investigate low complexity and low power consumption algorithms and their implementation using an algorithm-architecture holistic optimization approach.

Denoising and signal separation are two key signal processing tasks in such applications and often accomplished by Wavelet Transform (WT) and Independent Component Analysis (ICA) algorithms. However these algorithms are computationally intensive in their present form and would consume significant area and power resources when implemented in hardware. This thesis investigates the development of these two important signal processing algorithms and their architectures which facilitates their deployment in emerging resource constrained applications. Firstly a memory reduction methodology based on exploitation of symmetry is proposed leading to a low complexity and low power consumption Discrete WT architecture. Secondly, having identified FastICA as the most efficient algorithm among existing ICA algorithms in terms of convergence speed, the impact of its algorithmic parameters on its corresponding hardware implementation is investigated using an algorithm-architecture holistic optimization approach. Furthermore, a hardware complexity reduction methodology of 2-dimensional FastICA architecture based on eliminating redundant arithmetic operations is proposed. Two new concepts - vector cross product and coordinate rotation are introduced to reduce the implementation complexity of n-dimensional FastICA and their architectures are proposed. It is shown that these concept leads to reduction in complexity and lower power consumption when compared with the traditional implementation. All the proposed methodologies reported in this thesis are substantiated using C and Matlab models as well as VHDL based designs synthesized using 0.13μm technology.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, *Amit Acharyya*, declare that this thesis entitled

"Resource Constrained Signal Processing Algorithms and Architectures"

and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Either none of this work has been published before submission, or parts of this work have been published as listed in Appendix-A on page 198.

Signed:

Date:

Acknowledgements

National Centre for Radio Astrophysics (NCRA), Tata Institute of Fundamental Research (TIFR), Pune, India has introduced to me the wonder-world of research. So I would like to give my heartiest thanks to NCRA and its people especially Professor Yashwant Gupta, Dr. Dipanjan Mitra, Dr. Subhashis Roy, Dr. Chiranjib Konar, Dr. Bhaswati Bhattachryya, Mr. Jayanta Roy and Mr. Samir Dhurde.

I would like to convey my warm regards to Professor Sanjit Kumar Datta, former Head of the Department of Electronics and Communication Engineering Department, National Institute of Technology (NIT), Durgapur, India and Professor A. C. Ganguli, former Director of NIT, Durgapur, India who helped me in several ways to take part in a short-time research at NCRA, TIFR. I would like to thank Dr. Sumit Kundu, Associate Professor at NIT, Durgapur to offer me a good research project during my final year of Engineering course and helped me writing my first research paper.

My special thanks to Professor Yashwant Gupta of NCRA, TIFR, an Electrical Engineer turned into an Astro-physicist and Professor P. Ramachandra Rao, former Vice-Chancellor of Defence Institute of Advanced Technology, Pune, India, a Physicist turned into a world-renowned Metallurgist who motivated me to undertake the multi-disciplinary approach in research.

I would like to convey my warm regards to Dr. V. K. Manchanda, Director of Radio Chemistry Division of Bhabha Atomic Research Centre, India and Mr. Soumitra Bhowmick, former scientist in Defence Research and Development Organisation, India who not only motivated me to come abroad for cutting-edge research exposure but also advised me to come back and serve my own country - India after finishing my study.

I am indebted to my PhD supervisors Professor Bashir M. Al-Hashimi, Dr. Koushik Maharatna and Professor Steve R. Gunn for their continuous technical and financial support and having faith on me. I am grateful to Dr. Koushik Maharatna and his family who made me feel at home even miles away from my home.

This is my pleasure to acknowledge Hasitha Sunjeeva Tudugalle, Wen Tao He, Liang Li, Jinhong Sun, Abdul Saboor Mahar, Sayanta Mondal, Saransh Kothari, Vibhu Gautam, Ramona Sanadi, Taihai Chen, Wenyu Jin, Matthew Lokes and Bogdan Opris who worked with me during their BEng and MSc projects.

I would like to acknowledge my colleagues in my research group, to name a few - Mr. Karthik Baddam, Dr. Rishad A. Safik, Dr. Saqib Khursheed, Dr. Dafeng Zhou, Mr. Sheng Yang, Mr. Mustafa Ali, Mr. Aissa Melouki, Dr. Arash Ahmadi, Dr. Saket Srivastava, Miss Ghaithaa Manla, Mr. Eduardo Mangieri, Mr. Hamed Shahidipour, Mr. Evangelos B. Mazomenos, Mr. Dirk De Jager, Mr. Anton Kulakov, Mr. Jatin Mistry, Mr. Jedrzej J Kufel, Mr. Shida Zhong and Mr. Karim El Shabrawy for their co-operation throughout my PhD research.

I would also like to thank all administrative staffs with a special mention of Ms. Lucyna I. Palmer for their helpful and friendly attitude.

And finally, I would like to offer my warmest thanks to my parents and Swati without whose support and inspiration it would not have been possible for me to do my research.

I believe, it is too small space to acknowledge all the people who directly or indirectly contributed to my research. I would like to apologise for omission of any name. I hope your blessings and wishes will always be with me and make this a new beginning.

To my parents and Swati

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[1].

The rest of this chapter is organized as follows. Section 1.1 highlights the focus of this research study, Section 1.2 outlines the vision where resource constrained signal processing algorithms and architectures can be applied and Section 1.3 discusses the contribution of this thesis.

[1]The contents of Section 1.2 have appeared partly as "Hardware Development for Pervasive Healthcare Systems: Current Status and Future Directions"- by Acharyya *et. al.* in *IEEE Asia Pacific Conference on Circuits and Systems*. Please see Appendix-A for further detail.

1.1 Research Focus

The focus throughout this research-study is on the *resource constrained signal processing* algorithms and architectures in terms of low power, small area overhead and reduced hardware complexity.

In resource constrained applications such as WSN, personalized healthcare, since the data captured at the sensors are mostly corrupted with noise and are mixed with other signals, it requires sophisticated signal processing algorithms for signal denoising and signal separation. Wavelet Transform (WT), Kalman filtering and Short Time Fourier Transform (STFT) are widely used for these purposes [1], [2], [3], [4], [5]. Approaches like Blind Source Separation (BSS), in particular Independent Component Analysis (ICA), have been used for accurately separating out the target signal from the mixture of several signals and in its feature extraction [6], [7]. However, all these traditional algorithms are computationally intensive in terms of multiplications, divisions and square roots evaluation. Thus, in a resource constrained environment, owing to the lack of energy at the sensor nodes, these processes are generally carried out on a more powerful computation platform like a central server or using an on-board microprocessor. Therefore under such scenario, sensors are used only as data collector and the unconditioned data are continuously transmitted to the central server for further processing on continuous basis. This puts burden on the wireless network owing to the plethora of data to be transmitted continuously. In addition, the continuous data transmission puts high energy demand on the analog front-end of the embedded radio structure leading to fast draining of the battery.

In contrast to this classical approach, signal processing can be done at the sensor nodes themselves in order to reduce the burden on the network. Separate optimisation of signal processing algorithms and architecture cannot be applied here since this approach does not guarantee overall energy optimisation. Thus a holistic optimization approach needs to be taken for increasing energy efficiency. Following this route, novel low power, low area, reduced complexity signal processing algorithms and architectures will be reported in this thesis which would be suitable for emerging resource constrained applications.

1.2 Resource Constrained Signal Processing for Person-Centric Continuous Remote Health Monitoring

This section identifies one challenging application domain of the resource constrained signal processing algorithms and architectures to be proposed in the subsequent chapters as person-centric continuous remote health monitoring, does a comprehensive literature survey and presents the conceptual architecture of the envisaged system.

1.2.1 State-of-the Art

In recent years, rapid development in communication technology, micro-electronics industry and sensor networks opened up significant opportunity in continuous person-centric remote health monitoring system development [8], [9]. Collecting and disseminating patient data to caregivers or doctors for better understanding many facets of patients' daily lives and activities and modifying therapies to the individual is the important application of technology in healthcare. Another important context of it is emergency care to accelerate access to medical records at the emergency site or to bring the experts to the scene virtually. By giving medical professionals appropriate, complete information, it is expected to deliver better care that is tuned not only to the situation but also to the patient's specificities. It has also utmost importance in surgical field as surgeons and nurses must monitor and control various vital functions under intensely stressful conditions. So it is one of the primary research goals to develop systems to collect and process an ever increasing range of telemetry data from instruments used in an operating room and to augment human ability to detect patterns of concern that could require immediate action [10]. Recently these applications have started to be deployed - in one end, consumer devices easily network with home computers to gather sensor data and make it accessible to the physicians and in the other end, tele-surgery is becoming a practical reality where remote physicians able to consult on a patient's condition as well as take part in surgical procedure [10].

Such system has tremendous potential to provide in-time care and safety interventions, lowering healthcare cost and allowing early detection of health related problems [11]. Depending upon current research thrust, recent works can be classified in following three categories - system architecture, sensing and communication, and service.

System Architecture

Remote continuous health monitoring applications would require sensing, communication and services. This represents conflicting requirements and therefore numerous architectures have been proposed. Recently a generic reference architecture for such application is discussed in [8] on the basis of current status of it, and in conjunction with ideological and security mechanism. High level view of personalized healthcare is reported in [12]. An adaptable architecture is discussed in [13] where main research challenges have been identified as - pre-processing of the collected physiological signal to extract particular feature and analyzing the extracted data to make decision. A wide area mobile patient monitoring system architecture is discussed in [14]. According to this, the potential benefits of such system architecture are - continuous monitoring for chronically ill patients, better quality care and feedback for patients, increased medical capacity and reduced medical cost for patient care.

Sensing and Communication

In [15], 4C's - Computing, Communication, Cognition and Collaboration are identified

as the basic requirement of technology development for remote healthcare. In recent years considerable amount of research has been performed on Computing and Communication - the first two of 4Cs assuming that the sensor nodes can only capture data and transmit to the central node which has adequate resources for computation. Accordingly emphasis is given towards developing new generation distributed wireless intelligent sensor system [16] which is convenient to use, noninvasive in nature and unobtrusive so that it does not affect the normal activity. One such effort is given in [17] where a real-time remote arrhythmia monitoring system prototype has been developed in NASA. It describes the system architecture and implementation issues for such system from the perspective of wireless communication technology. The management framework for comprehensive wireless patient monitoring is discussed in [18]. The main research focus is in reliable network architecture development. The applications and challenges of remote person-centric health monitoring are discussed in detail in [19] by the same author where important issues in communication layer includes throughput, quality of service, network reliability, network coverage and traffic management are discussed. Recently an interesting research has been carried out in [20] to provide a common taxonomy of such health monitoring systems.

The importance of medical sensors for remote health monitoring of the patient is depicted in [14]. In [21] and [22], taxonomies of Wireless Sensor Networks (WSN) models have been presented depending upon the different communication functions and classification is done based on data delivery models, network dynamics and communication protocol perspective. Survey on wireless sensor network has been done in [23]. Wireless integrated network sensors combining micro-sensor technology, low-power signal processing, computation and low-cost wireless networking in a compact system is presented in [24]. The importance of signal processing in the wireless sensor networks has been described in [25] to reduce the number of bits transmitted over the communication channel. It has been mentioned that different optimum algorithms to solve different research problems related to signal processing are not efficient when implemented because the total power consumption overshadows their benefits. Power consumption has been identified as one of the most important constraint for designing sensor rich wireless networks in [26] and [27]. The advantages of such pre-processing unit inside the sensor nodes as it makes the transmitter/ receiver coding/ decoding strategies optimum and reduces number of bits transmitted over communication channel is identified in [28]. This reduces the burden on communication layer as well as consumes less power. In [27] and [29], it is shown that the sensor nodes have limited energy resources and mainly depend on on-board energy (battery back-up) or harvested energy.

Service

[12] indicates that next generation of remote healthcare services must provide personalized solution across distributed networks, where care professionals or informal care givers can monitor an individual's wellbeing in their own home. This will accomplish

the goal of independent living and cost effectiveness in healthcare delivery. Subject to user agreement, information can be sent to a variety of care providers. Security and authentication have been identified as main concern in service layer [12]. One important issue in this layer is how to provide better healthcare services to an increasing number of people using limited financial and human resources [18]. The current and emerging technology could improve the overall quality of service for patients in both cities and rural areas, reduce the stress and strain on healthcare providers, while enhancing their productivity, retention and quality of life, reduce long term cost of healthcare services [18]. There should be communication link among hospital, ambulance and healthcare providers for getting emergency service. Healthcare professionals have to be trained so that effective utilisation of such system is possible [18]. Along with this longitudinal and clinical studies are needed to realize the benefits of person-centric remote continuous health monitoring [11].

On the basis of the above discussion, the vision of a person-centric remote continuous health monitoring system and applicability of resource constrained signal processing in such a system is outlined in the next sub-section along with a conceptual architecture.

1.2.2 Envisaged Remote Health Monitoring: Conceptual Architecture

Based on various architectures proposed in [8], [12], [13], [14], it can be said that an effective system for person-centric remote health monitoring comprises of three layers - Sensor, Communication and Service, as shown in Fig. 1.1. In the first layer a set of sensors are deployed for collecting the vital signs of the person under monitoring. Generally these collected data are transmitted via communication layer to the central facility. WLAN/ cellular/ GSM/ 3G network could be used for serving this purpose. In service layer appropriate healthcare service can be informed to take action.

This architecture may be effective, but it faces the problem of transmitting significant amount of data over the communication layer and effective management of the data at the central facility. Continuous data transmission signifies the transceiver module within the sensor node is active most of the time. Since for remote health monitoring environment sensors are run by battery back-up, it may not be ideal to keep the transceiver module active for long time because its embedded radio front-end is the most power-hungry part of the sensor. So continuous data transmission may lead the sensor node to run out of power very soon and thus makes this state-of-the art architecture unsuitable for such health monitoring system.

One possible solution is to envisage an intelligent signal processing circuitry embedded within the sensor node in the Sensor Layer which can monitor the patient's vital signs on a continuous basis. However this approach calls for resource constrained signal processing algorithm and architecture development in terms of small area overhead and

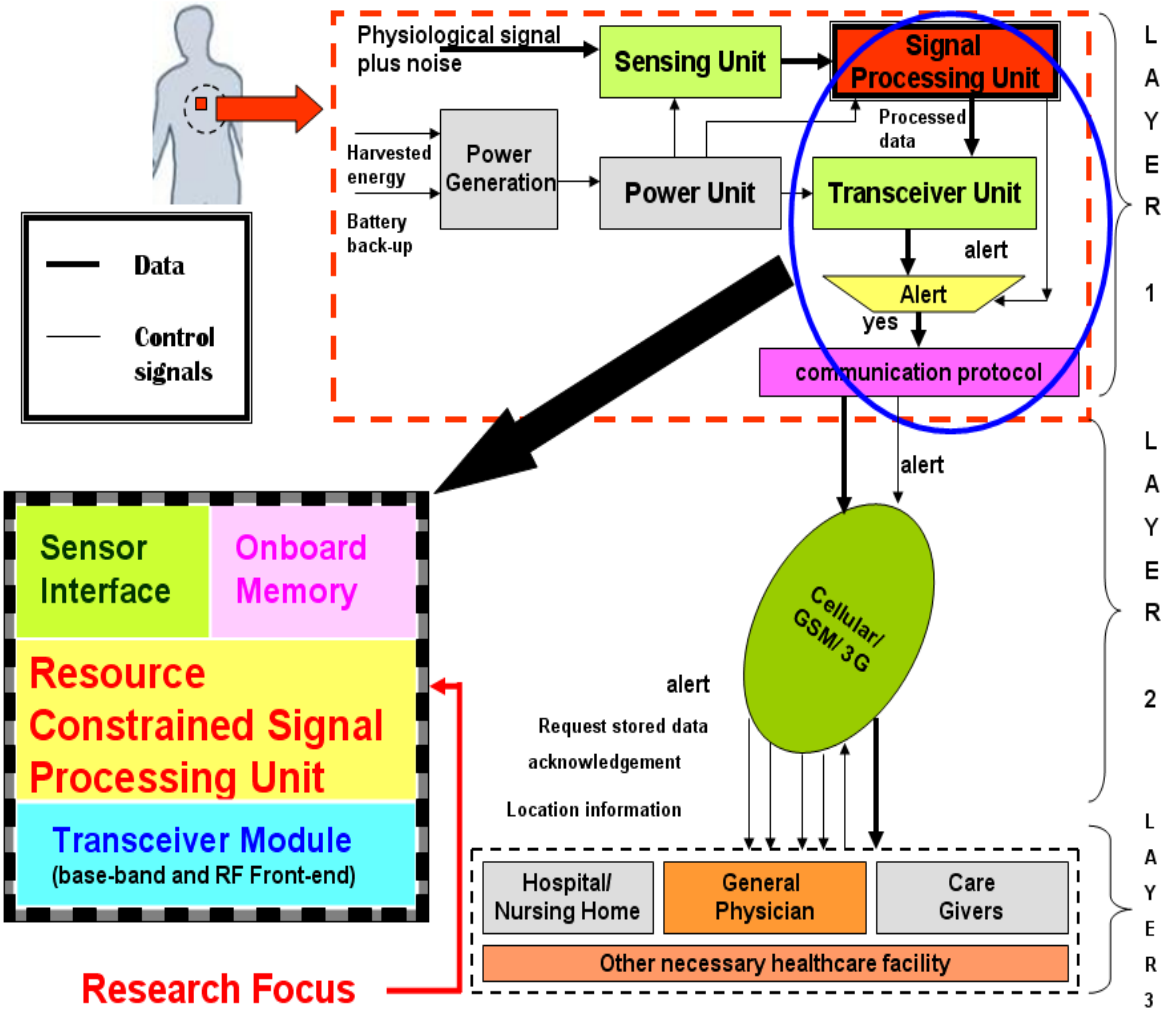


FIGURE 1.1: Conceptual Block diagram of personalized remote health monitoring system, Layer-1 : Sensor Layer, Layer-2: Communication Layer, Layer-3: Service Layer.

low power consumption which would be different from the classical signal processing algorithms design.

A remote health monitoring system, as shown in Fig. 1.1, can be envisaged where the specialized signal processing unit and corresponding intelligent decision making circuitry will be embedded inside the sensor layer within the sensor nodes for continuous monitoring of patient's vital signs. It can communicate with transceiver inside the sensor node to store processed data in home computer and only transmit the signal to the central facility through this transceiver when it finds variation of the vital sign's pattern from the normal pattern. This design unit inside the Sensor Layer within the sensor node will check the trend of the vital sign from the received sensor data. If any abnormality is sensed, then only the central node will be communicated. Otherwise the processed medical data will be stored in the local computer using home wireless infrastructure. This approach will reduce the burden on the communication layer significantly. If required, the central facility can call for the data stored in the home computer and

other relevant information over the communication layer for detail diagnosis of the vital signs. However, processing the physiological signals on a continuous basis in this type of infrastructure is an extremely challenging problem owing to the limited resources as mentioned in [24], [25] and [27].

To materialise such a system, it is necessary to reduce the complexity of the required signal processing algorithms and architectures in such a way that each of the processing circuit occupies low area and consumes as small power as possible. Since the physiological data captured by the sensors are mixed with noise and several artifacts, the on-sensor processing unit has to run some denoising and artifact separation algorithms prior to the intelligent decision making circuitry checks for the variation of the patterns of the person's vital signs. This necessitates the investigation of the traditional signal processing algorithms from their architectural perspective and follow algorithm-architecture holistic optimization approach so that these can be fit into such resource-constrained environment. Thus throughout this research, primary focus will be on designing several low-power reduced complexity signal processing algorithms and architectures which can be useful for emerging resource constrained applications such as the envisaged remote health monitoring system.

1.3 Thesis Synopses and Contribution

Low-power low-complexity signal processing algorithms and architectures design is the key part for energy minimisation in the resource constrained applications mentioned above which cannot be warranted by separate optimisation of algorithms and architectures. Thus throughout this thesis signal processing algorithms and architectures holistic optimization approach will be considered. In terms of signal processing algorithms, the main focus will be on - signal denoising and signal separation.

Typically Fast Fourier Transform is used for signal denoising. However since most of the real-life signals e.g. biomedical signals are non-stationary in nature, this approach is deemed to undermine the achievable quality besides being computationally expensive. A better way is to apply the Discrete Wavelet Transform (DWT) by exploiting its time-frequency localisation property to capture the essence of non-stationary nature of the real-life signals. However DWT is computationally intensive and thus its direct mapping in hardware is not suitable for resource constrained applications. The research undertaken in this thesis has shown that it is possible to reduce the hardware complexity and power consumption of the DWT significantly when the problem of algorithm and architectural optimisation is addressed in a cooperative way. This will be detailed in Chapter 2 where a novel memory reduction methodology will be proposed and it will be shown that the proposed architecture consumes significantly low power and area.

Separation of a target signal from a mixture of several signals can be categorized un-

der the classical BSS problem. Typical BSS algorithms like ICA require complicated mathematical operations in terms of divisions, multiplications, square root evaluations resulting in significant power consumption and silicon area. Therefore although effective for off-line workstation based processing these algorithms pose big problem from energy consumption point of view for resource constrained systems like the envisaged person-centric remote health monitoring system. However to develop an efficient but low-complexity BSS scheme and corresponding maximally power efficient architecture for the same, a rigorous study of the impact of several algorithmic parameters (viz. frame-length to be considered, algorithmic accuracy, iteration of convergence for the adaptive signal processing algorithm etc.) on the corresponding architecture is essential prior to its implementation in silicon to ascertain the optimum performance of it. Although important from the system designer's point of view, in the literatures, no research effort has been put forth in this area. In Chapter 3, identifying FastICA as one of the most popular among existing ICA algorithms, impact of such algorithmic parameters is investigated on its corresponding architecture and subsequently a novel low complexity 2-dimensional ($2D$) FastICA algorithm and architecture are proposed.

Then the concept of FastICA is explored in the n^{th}-dimension and several low complexity novel algorithms and architectures are proposed without sacrificing its algorithmic efficiency. To do that the concept of n-dimensional vector cross product is introduced for the first time, in the literatures, in signal processing domain of ICA in Chapter 5. Due to the lack of known engineering applications of cross product, it has not got much attention from the engineering research community so far. Motivated by this fact a recursive generalized n-dimensional cross product architecture based on the fundamental 3-dimensional cross product is proposed in Chapter 4. By exploiting the inherent symmetry of the cross product computation structure, a further hardware reduction methodology of this proposed generalized architecture is also introduced in Chapter 4. This architecture is used in Chapter 5 to show how it will significantly reduce the hardware complexity of the conventional FastICA.

Since FastICA, as will be discussed later in the subsequent chapters, comprises of two steps - Pre-processing and FastICA Iteration, it has been investigated further how these two steps can be merged together so that same hardware module used for pre-processing unit can be reused to design the FastICA Iteration step. Subsequently the concept of Coordinate Rotation is introduced for the first time in FastICA which will be discussed in Chapter 6 and Coordinate Rotation based novel FastICA algorithms and architectures in n^{th} dimension are proposed. It will be shown how this concept helps reducing the hardware complexity significantly by reusing the same hardware module for both of these two steps.

Finally, Chapter 7 concludes the thesis by pointing out its salient research contributions and identifying some challenging research problems for future research. Proposed memory reduction methodology based DWT architecture is used to design a low complexity

Fetal ECG extraction architecture and is discussed in Appendix-B. Like any other ICA techniques, FastICA also has the problem of permutation ambiguity. This happens in any BSS algorithm because it tries to estimate the source signals without having any knowledge of the channels or without having any idea of how they have been mixed. Appendix-C addresses this permutation ambiguity considering 3-dimensional noise and artifact separation environment from ECG signals and subsequently a robust and automated solution is proposed in this context.

Each research contribution of this thesis as mentioned above has either been published/ accepted or is under consideration or to be communicated in some prestigious research journals and conferences. The published/ accepted research papers are listed in Appendix-A on page 198.

Chapter 2

Memory Reduction Methodology for Discrete Wavelet Transform

Literature survey on current research trend in the previous chapter highlighted the need of designing the low-power, low-complexity signal processing algorithms and architectures for resource constrained environment such as WSN and personalized healthcare. It has been envisaged to embed such signal processing unit inside the sensor nodes present in the wireless sensor networks to analyse the data within the node itself and thereby reduce the burden on the communication layer. Keeping this as the basic goal of the research, some available signal processing schemes have been studied. Wavelet Transform (WT) is a perfect match for such application.[1]

This chapter begins with a review on the evolution of the concept of WT from the Fourier Transform and explains its applicability in one of the most challenging domain of signal processing - healthcare. Then this chapter provides the basics of WT algorithm, surveys the critical issues of available WT architectures and proposes one memory efficient architecture of Discrete WT (DWT). The proposed architecture is multiplier-less and reduces the hardware requirement in terms of memory significantly. Besides, the proposed architecture is generic and independent of any specific wavelet function. Thus it is suitable for any type of WT architecture design.

[1]The contents of Section 2.4 - Section 2.8 have partly appeared as "Memory Reduction Methodology for Distributed Arithmetic Based DWT/ IDWT Exploiting Data Symmetry" by Acharyya *et. al.* in *IEEE Transactions on Circuits and Systems - II: Express Briefs*. The contents of Section 2.9 along with Appendix-B have partly appeared as "VLSI Architecture for Fetal ECG Extraction for Personalized Healthcare Application within Resource Constrained Environment" by Acharyya *et. al.* in *Proceedings of the UK Electronics Forum-2010*. Please see Appendix-A for further detail.

2.1 Wavelet Transform: A Review

WT is popular among research communities. But it is not a new technique, it is indeed two decades ago when J. Morlet and A. Grossman proposed the concept of wavelet analysis to reach automatically the best trade-off between time-frequency resolution and developed the mathematical foundation of wavelets. Later this proposition has been considered as the extension of the ideas given by Haar in 1910 and Gabor in 1946. As any discovery in science, wavelets resulted from numerous contributions, they are based on the concepts that already existed before this proposed idea [30]. But at that stage wavelets were still very much in the realms of pure mathematics and concentrated more on the theory rather than the application [31]. Only a handful of scientific papers used to be published each year and that was also mainly by the mathematics community [4]. In [32] gives the basic introduction of wavelets and dilation equation. But Mallat, Daubechies and Meyer changed this by defining the connection between wavelets and digital signal processing. In [33] multi-resolution signal decomposition using WT technique is proposed. In [34] the relationship between the WT and its applicability in signal processing discussed in detail. The design aspects of wavelets and its theoretical detail have been combined and presented nicely in [35]. Design and development of fast algorithm for both discrete as well as continuous time WT has been depicted in [36]. In [37] discrete time multi-resolution theory of WT is proposed which came out to be pioneering work in the domain of wavelet architecture design in digital signal processing research.

Approximation using superposition of functions has existed since the early 1800's, when Fourier discovered that he could superpose sines and cosines to represent other functions. Let us consider a signal, $x(t) = \cos(2\pi 10t) + \cos(2\pi 25t) + \cos(2\pi 50t) + \cos(2\pi 100t)$. This is an example of a stationary signal because it has frequencies of 10, 25, 50 and 100 Hz. For this stationary signal, Fourier transform provides the necessary frequency information. But, on the contrary, now let us consider another signal having four different frequency components at four different time intervals; 100 Hz sinusoid within 0 to 300 ms interval, 50 Hz sinusoid in 300 to 600 ms, 25 Hz within 600 to 800 ms interval and 10 Hz sinusoid within 800 to 1000 ms interval. This signal is a representative of non-stationary signal. In this case Fourier transform fails to localize frequency with its respective time of occurrence. For many decades, scientists have wanted more appropriate functions than the sines and cosines which comprise the bases of Fourier analysis, to approximate the choppy signals. By their definitions, these functions are non-local i.e. stretched out to infinity. They therefore, do a poor job in approximating sharp spike. Here comes the advantage of wavelet analysis.

The fundamental idea behind wavelets is to analyze the data at different scales and resolutions. If a signal is looked with a large "window", its gross features would come out. Similarly, if the signal is looked with a small "window", the small features can be

noticed. So, the result in wavelet analysis is *to see both the forest and the trees*, so to speak [38]. This makes wavelets interesting and useful.

The wavelet analysis procedure is to adopt a wavelet prototype function, called *analyzing wavelet* or *mother wavelet*. Temporal analysis is performed with a contracted, high frequency version of the prototype wavelet, while frequency analysis is performed with a dilated, low frequency version of the same wavelet. Because the original signal or function can be represented in terms of a wavelet expansion i.e using coefficients in a linear combination of the wavelet functions and data operations can be performed using just the corresponding wavelet coefficients [38].

Considering $s(t)$ be the time-domain description of the incoming signal, $\psi(t)$ be the mother wavelet, a and b be the scale and time-shift variable respectively, WT can be given as [31], [37]:

$$WT(a,b) = a^{-1/2} \int s(t)\psi^*\left(\frac{t-b}{a}\right) \mathrm{d}t \tag{2.1}$$

where "$*$" denotes the "complex conjugate" notation. From (2.1), it can be seen that the WT performs decomposition of signal $s(t)$ into a weighted set of basis functions which are scaled version of the mother wavelet $\psi(t)$ [31]. Decomposition into these basis functions can be seen as time varying spectral analysis in which scale a plays a role of local frequency: with the increase of a, wavelets are stretched and analyze low frequencies and so on [37].

In [39] a clear and comprehensive mathematical background of wavelets is provided. A comparative study of classical Fourier analysis and comparatively new wavelet analysis has been presented in [40]. The necessary theoretical detail of WT has been well depicted in book [41] and the applications of WT in different signal processing tasks has been well explained in [42].

2.2 Healthcare: An Important Application Domain of WT

WT technique differs from the traditional Fourier transform by the way in which it localizes the information in the time-frequency plane; in particular, it is capable of trading one type of resolution for the other, which makes it suitable for analysing non-stationary signals. One privileged application, where these properties have been found to be relevant is healthcare. Due to the wide variety of signals and problems encountered in medicine and biology, the spectrum of applications of WT has been extremely large. It ranges from the analysis of the more traditional physiological signals such as Electrocardiogram (ECG) to the recent imaging modalities including Positron Emission Tomography (PET) and Magnetic Resonance Imaging (MRI)[64], [65].

Most biomedical signals, however, tend not be stationary and they typically have highly

complex time-frequency characteristics. Frequently, they consist of brief, high-frequency components closely spaced in time, accompanied by long-lasting, low-frequency components closely spaced in frequency. Any appropriate analysis method for dealing with them should therefore exhibit good frequency resolution along with fine time resolution - the first to localize the low-frequency components, and the second to resolve high-frequency components.

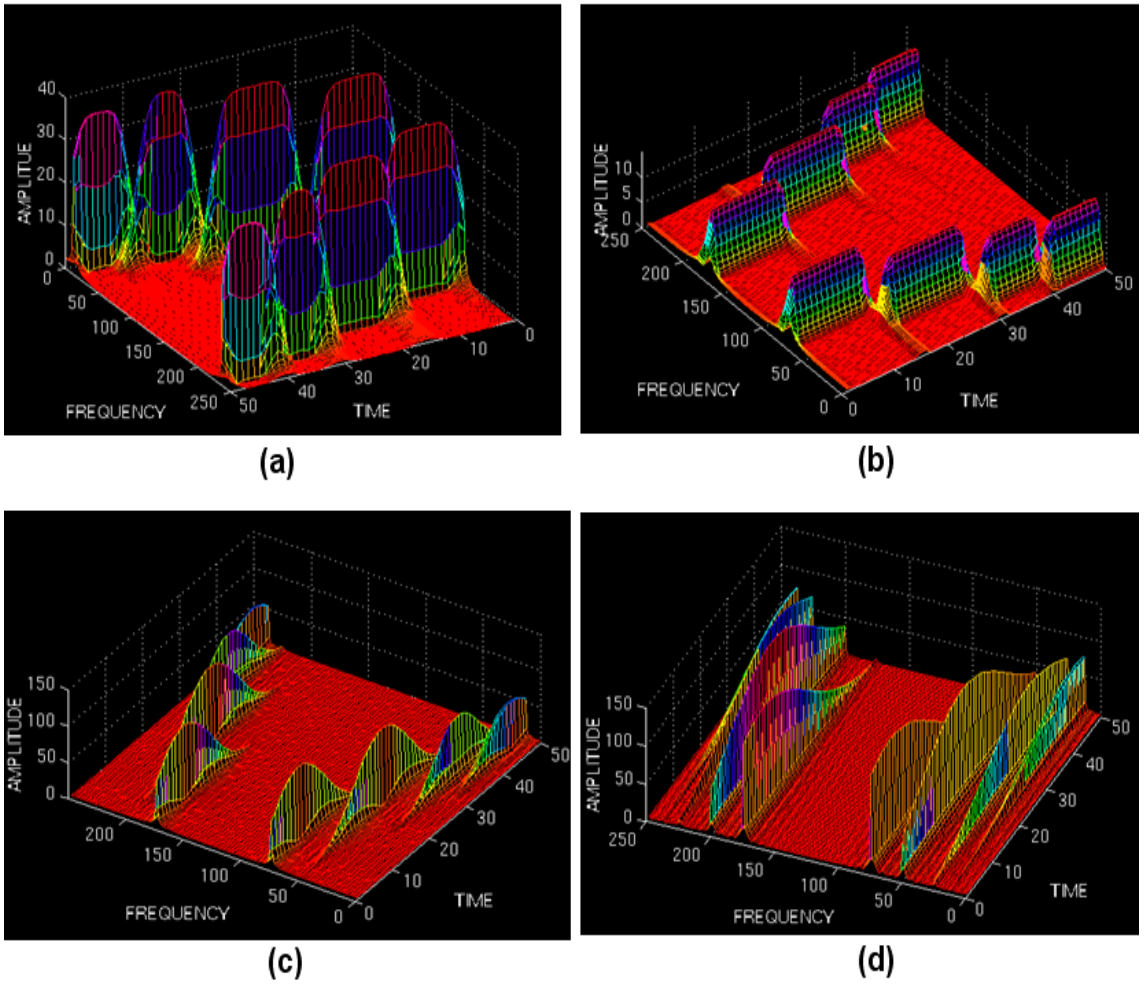


FIGURE 2.1: Short Time Fourier Transform with different window size. (a) and (b) represents same window size but observed from two different angles. Window size decreases from (b) to (d).(Taken from [72])

In practice, it is often possible to treat non-stationary signals as stationary ones by dividing them into blocks of short, pseudo-stationary segments - that is, segments whose statistics remain essentially unchanged for their duration. This approach appears to be valuable in many applications and popularly known as Short Time Fourier Transform (STFT). The STFT of a signal is plotted in Fig. 2.1(a) and (b) from two different angles. From these two plots, it may appear that time-frequency localization problem is solved and now it can easily be seen what are the frequency components are present within a certain time window. However, it also has one potential disadvantage: if the time-domain analysis window is made too short, frequency resolution will suffer. Fig. 2.1(c)

and (d) present such scenario where frequency components belonging to different time windows start overlapping with each other because of the selection of the smaller window size. Lengthening of the window size, on the other hand, may invalidate the assumption of stationarity within the window. So for many practical signals such as physiological signals, STFT may not be a very suitable candidate for such time-frequency analysis.

An alternate way to analyze non-stationary biomedical signals is to expand them onto basis functions created by expanding, contracting, and shifting the mother wavelet, specifically selected for the signal under consideration [66]. This wavelet method acts as a sort of mathematical microscope through which different parts of the signal may be examined by adjusting the focus. As mentioned before, this prototype function is known as "analyzing wavelet" or "mother wavelet" of the signal [66]. WT is performed on the same signal chosen in Fig. 2.1 and is plotted in Fig. 2.2(a) and (b) from two different angles. The time-frequency localization property is now clearly visible from this figure.

Fig. 2.3 illustrates the shortcomings of STFT analysis in detecting signal features of short duration. This figure contains a scalogram[2] and a spectrogram[3] corresponding to the rhythmic ECG signal shown at the top of the figure. The time domain signal is derived from a pig heart which has been shocked several times. The spectrogram is generated from an STFT employing a 3.4s Hanning window typical for the analysis of this type of signal. The smearing and hence loss of local information across the spectrogram over these time scales is evident in the plot.

The main difficulty in dealing with biomedical objects is the extreme variability of the signals and the necessity to operate on a case by case basis. Often, one does not know *a priori* what is the pertinent information and/ or at which scale it is located. For example, it is frequently the deviation of some signal feature from the normal that is most relevant information for diagnosis. As a result, the problems tend to be less well defined than those in engineering and the emphasis is more on designing robust methods that work in most circumstances, rather than procedures that are optimal under very specific assumptions. Another important aspect of biomedical signals is that the information of interest is often a combination of features that are well localized temporally or spatially (e.g. spikes and transients in Electroencephalograph (EEG) signals and micro-calcifications in mammograms) and others that are more diffuse (e.g. small oscillations, bursts and texture). This requires the analysis methods sufficiently versatile in terms of their time-frequency localization [64].

As shown in Fig. 2.4, WT can provide both good time resolution at high frequencies and good frequency resolution at low frequencies. This excellent combination of time-frequency resolution makes wavelets potentially invaluable in numerous applications, many of which fall into the realm of medical research and diagnostics. Among them

[2] distribution of energy i.e. square modulus of the wavelet coefficients of the signal in time-scale plane [34]

[3] distribution of energy i.e. square modulus of the Fourier coefficients in time-frequency plane [34]

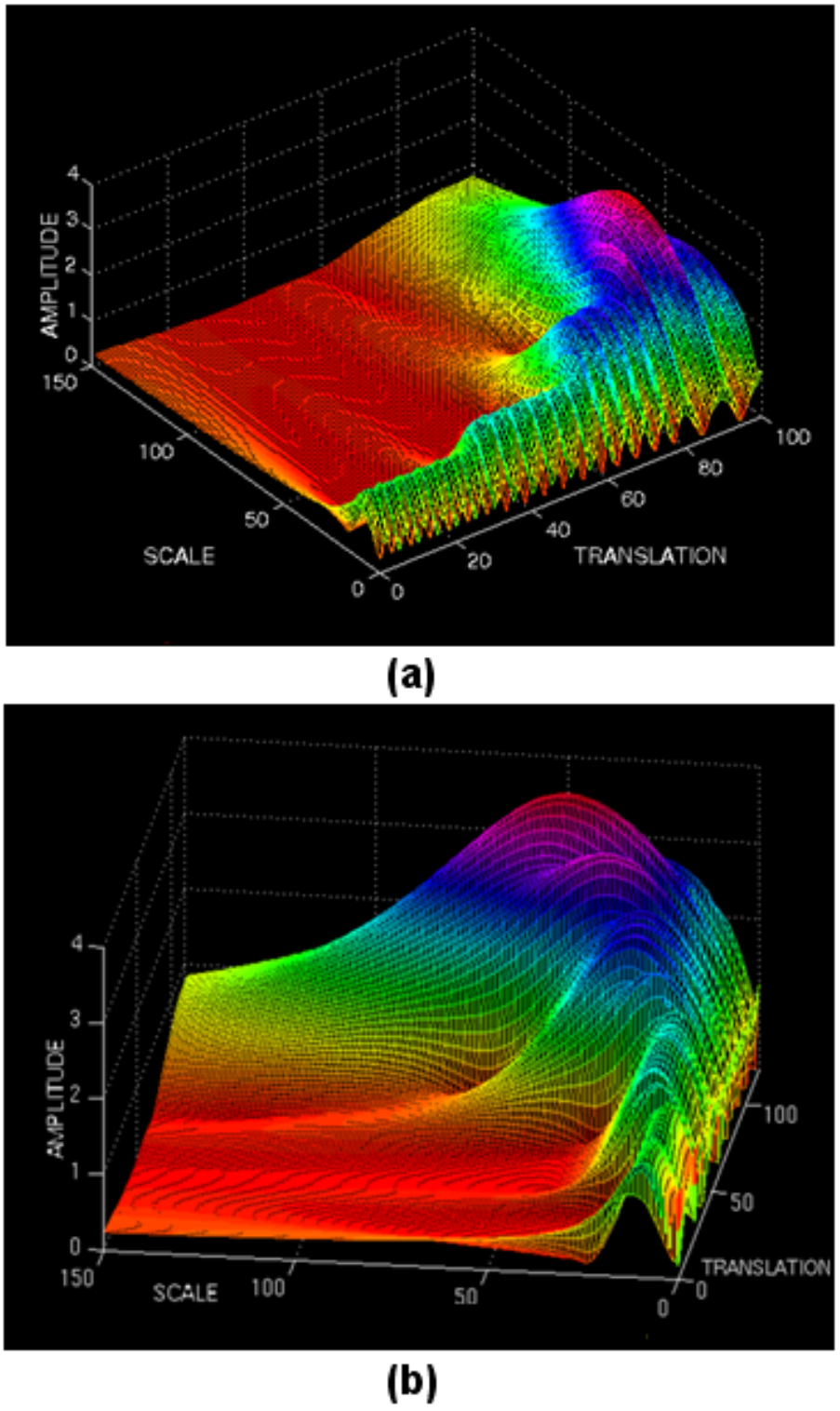


FIGURE 2.2: Wavelet Transform performed on the same signal chosen for Fig. 2.1. Translation and Scale represent b and a respectively in (2.1).(Taken from [72])

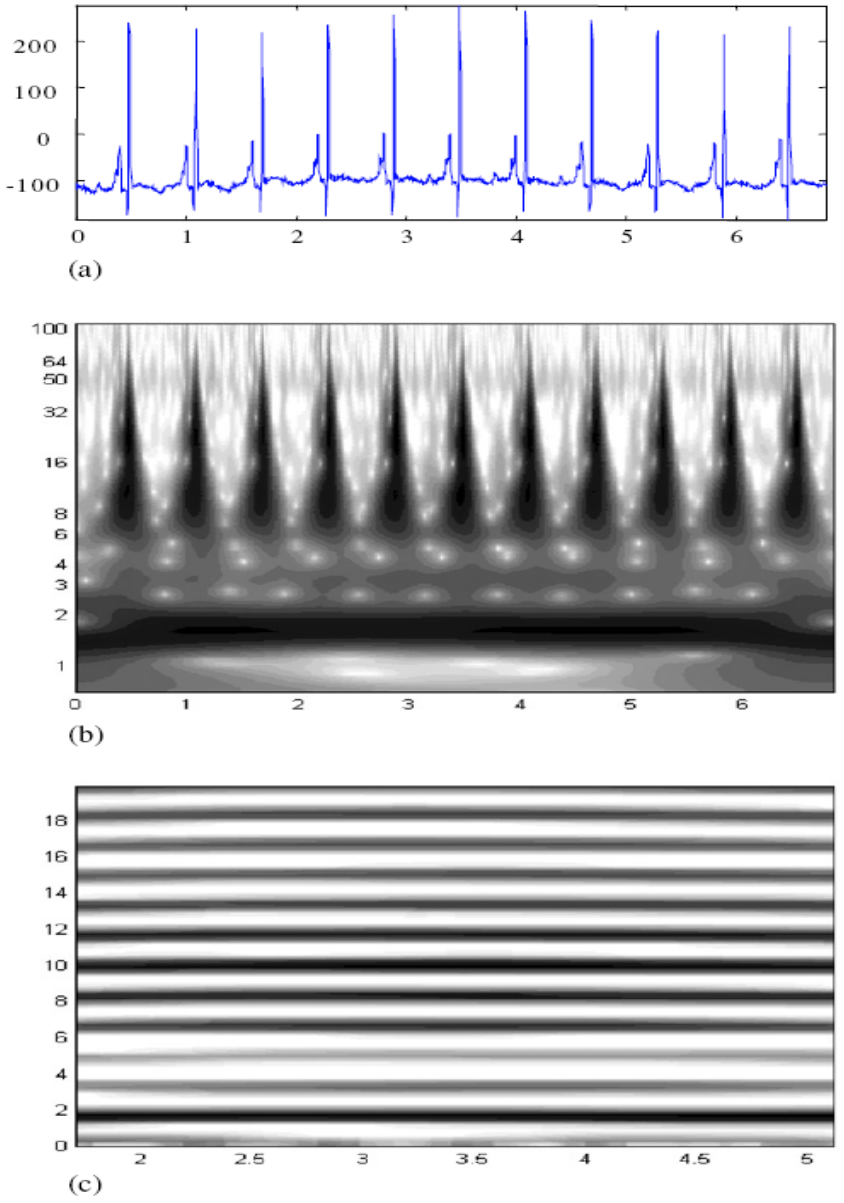


FIGURE 2.3: Wavelet scalogram versus STFT spectrogram for rhythmic signal. (a) original time-domain rhythmic ECG signal (X-axis: time, Y-axis: amplitude). (b) Morlet based scalogram corresponding to (a) (X-axis: time, Y-axis: Scale as defined by a in (2.1)). (c) Spectrogram corresponding to (a) generated using STFT with a 3.4s Hanning window (X-axis: time, Y-axis: frequency). (Taken from [71])

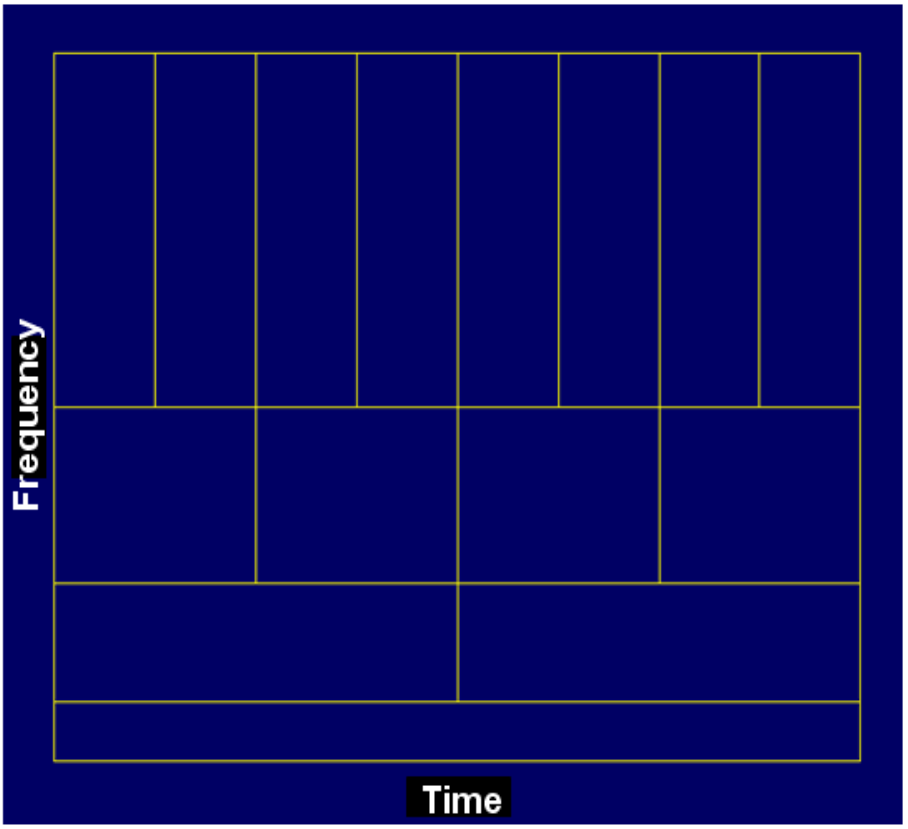


FIGURE 2.4: Time-Frequency Localization Property of Wavelet Transform.(Taken from [72])

may be found the early discovery of the precursors of heart disease, studies of fetal breathing, fetal electrocardiogram detection, heart rate variability, extraction of speech from background noise in digital hearing aids, the detection of breast cancer, medical image compression and many more [66].

2.3 Preliminaries of DWT and IDWT

Typically DWT and Inverse DWT (IDWT) are computed using Analysis and Synthesis filter banks respectively as shown in Fig. 2.5 and Fig. 2.6 for four levels of resolution. In Analysis Bank the original discrete time sequence is decomposed into coarser half resolution approximation level with a half-band low-pass and high-pass filters.

In dyadic space, this analysis filtering scheme at each stage can be depicted as [48], [52]:

$$w_k^j = \sum_n x_n h_{2k-n}, \qquad n, k \in Z \tag{2.2}$$

$$r_k^j = \sum_n x_n g_{2k-n}, \qquad n, k \in Z \tag{2.3}$$

where, x_n is input discrete time sequence to the filter, h_n and g_n are high-pass and low-

pass filter impulse response respectively; and w_k^j and r_k^j are their corresponding outputs (wavelet and residue) for j^{th} stage. The relationship between these two filters is given as [52]:

$$h_{L-1-n} = (-1)^n g_n \tag{2.4}$$

where, L is filter length and $(-1)^n$ is the modulation parameter which transforms low-pass filter into high-pass one.

Similarly, IDWT at each resolution level of synthesis bank can be described as [48], [52]:

$$s_n = \sum_k [w_k^j h_{n-2k} + r_k^j g_{n-2k}], \quad n, k \in Z \tag{2.5}$$

where s_n is the reconstructed data output from the Synthesis Bank.

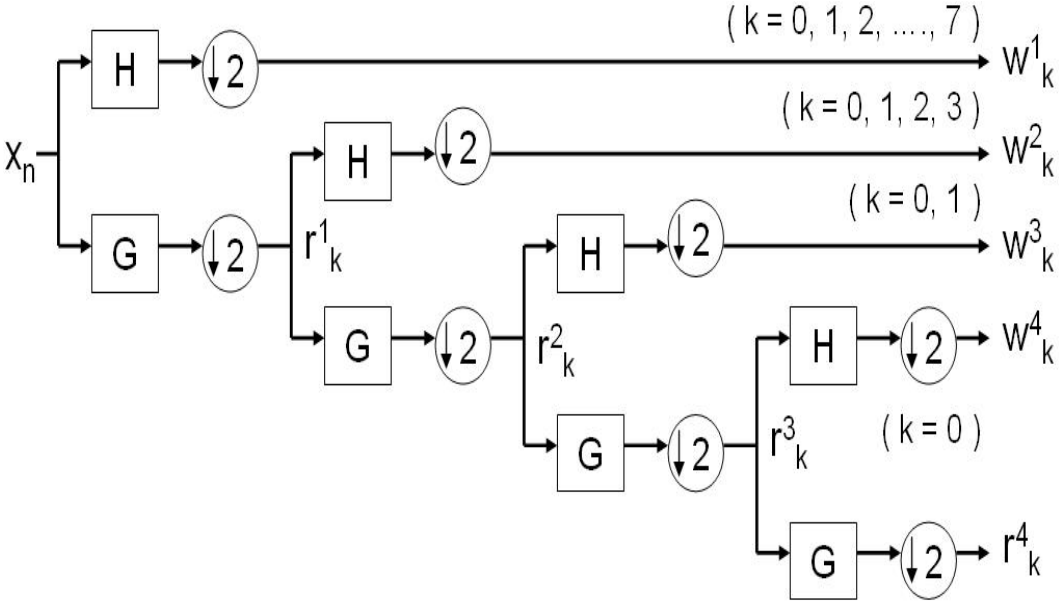


FIGURE 2.5: Four-resolution level Analysis Bank for DWT

2.4 DWT Architectures: A Review

The DWT is of great research interest due to its numerous applications in several fields including signal processing [43], [44]. DWT provides a discrete-time, discrete-scale representation of signals as an alternative to traditional time-frequency representations and helps overcoming some limitations of such approaches [45], [46].

Over last two decades numerous architectures have been proposed in literatures for implementing Forward and Inverse DWT (IDWT). First such architecture was designed by Knowels [47]. Parhi and Nishitani have proposed folded and digit-serial architectures for One Dimensional DWT (1-D DWT) [48]. Grzeszczak, Yeap and Pachanathan proposed the DWT architecture with less number of multipliers than the existing methods [49].

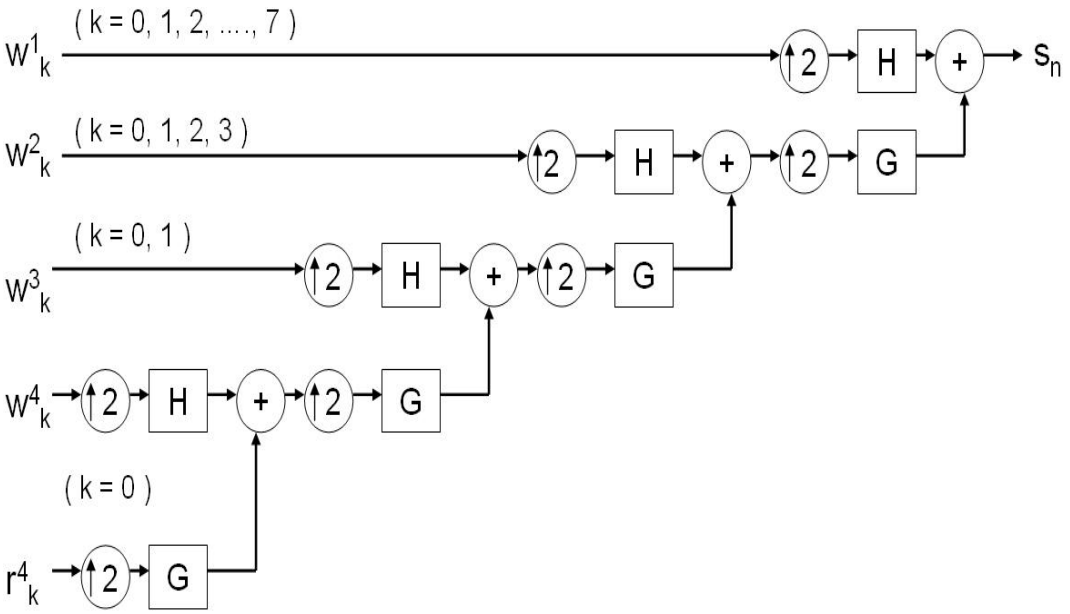


FIGURE 2.6: Four-resolution level Synthesis Bank for IDWT

Fridman and Manolakos came up with the first fastest parallel architecture of 1-D DWT [50]. Then onwards various implementation schemes have been proposed [45], [51]-[60] and improvement of architectures has been made over the period of time. In general these implementations have been influenced by certain characteristics such as computational time and complexity, regularity, smoothness and linear phase characteristics, transform efficiency as well as perfect reconstruction of the decomposed signals [59]. But, most of the reported hardware approaches, though focuses on improving computational speed, does not adequately address severe power and area constraints issues [59]; which often are the two most important metrics in today's high performance signal processing systems. As for example sophisticated applications such as remote health monitoring system require light-weight signal processing with particular emphasis on low power and low area due to scarcity of resources. So, design of optimal Application Specific Integrated Circuit (ASIC) tailored to such applications consisting of DWT/ IDWT demands power and area efficient implementation of DWT algorithms.

The main power consuming operation in DWT/IDWT computation is the filtering which requires significant number of multiplications [56]. Distributed Arithmetic (DA) can be adopted for eliminating the requirement of multiplications [56] which may lead to reduction of power consumption. However in the conventional DA based approach one needs to store all the possible combinations of filter coefficients in a memory which increases exponentially in size with the frame-length [56]. Thus for longer frame-length the advantage of using DA may eventually be lost owing to the significant increase in memory size.

Here we propose a novel methodology for memory reduction in DA based design of DWT/IDWT architecture by exploiting its inherent algorithmic symmetry resulting in

data repeatability. Subsequently a 16-bit fixed-point DWT/IDWT architecture is developed for frame-length 16, which requires less silicon area and power consumption compared to some of the published DWT/IDWT architectures. This architecture will be independent of any specific mother wavelet and thus can be reconfigured for any type of wavelet function.

2.5 Motivational Example: DA Based Approach

Distributed Arithmetic (DA) was proposed about two decades ago and has since used widely in VLSI implementation of Digital Signal Processing (DSP) architectures [56], [57], [61]-[63]. Although DA based methods are advantageous, but in the context of convolution-based filtering such as DWT/ IDWT (as an example, 9/7 - 5/3 wavelet vlsi architecture [58], [60]), its memory requirement has exponential relationship with incoming data-frame length [56]. We will demonstrate that it is possible to restrict this exponential growth in memory requirement through pre-computing carefully various combinations of filter coefficients.

In this section, the relationship between memory requirement of DA based filtering technique and frame-length and its implications in DWT/ IDWT architecture will be shown. DA is a bit serial computational operation which is used to compute the inner product of a pair of vectors without using multipliers. The basic DA equation is given as follows [61]-[63]:

$$x_n = -x_{n,l}.2^l + \sum_{b=0}^{l-1} x_{n,b}.2^b \tag{2.6}$$

where l =(total number of bits/ sample). In dyadic space, a convolution based wavelet filter can be represented as:

$$w_a = \sum_{n=0}^{3} x_n h_{2a-n}, \tag{2.7}$$

where x_n and h_n are input samples and filter coefficients respectively. Considering frame-length = 4 and wordlength = 4, (as an example) and using (2.6) in (2.7) and assuming $a = 1$,

$$\begin{aligned} w_1 = -[x_{03}h_0 + x_{13}h_1 + x_{23}h_2]2^3 + ... \\ ... + [x_{00}h_0 + x_{10}h_1 + x_{20}h_2]2^0 \end{aligned} \tag{2.8}$$

where $x_{ij} = i^{th}$ sample j^{th} bit of the input data. The possible combinations of filter coefficients obtained from (2.8) are shown in the first 6 rows of Table 2.1 which occupies

16 memory locations. However, it can be observed from Table 2.1 that there exists redundant (such as '0') and repetitive filter coefficients (such as 'h_0', 'h_1','h_2','$h_1 + h_2$', '$h_0 + h_1$', '$h_0 + h_2$', '$h_0 + h_1 + h_2$') occupying more than a single memory location. Thus if only the unique combinations of the filter coefficients are stored in the memory the other filter coefficients can be obtained on-the-fly using simple addition operation. In this particular example, the proposed methodology leads to only four memory locations, as shown in the last two rows of Table 2.1, rather than 16 locations in the conventional approach.

TABLE 2.1: Conventional and reduced memory requirement to store combinations of filter coefficients in DA for data-frame length = 4 and 4 bits/ sample case

Adr	Data	Adr	Data	Adr	Data
0000	0	0001	h_2	0010	h_1
0011	$h_2 + h_1$	0100	h_0	0101	$h_0 + h_2$
0110	$h_0 + h_1$	0111	$h_0 + h_1 + h_2$	1000	0
1001	h_2	1010	h_1	1011	$h_2 + h_1$
1100	h_0	1101	$h_0 + h_2$	1110	$h_0 + h_1$
1111	$h_0 + h_1 + h_2$	—	—	—	—
00	h_0	01	h_2	10	h_1
11	$h_1 + h_2$	—	—	—	—

However, reducing memory at the expense of adder arises two particular issues. Firstly, a new addressing scheme needs to be formulated to address the reduced memory system. Secondly the hardware savings obtained due to reduction in memory size, can be negated if the total number of adders used in the design is more than a certain limit. These issues are discussed in Sections 2.7 and 2.8 respectively.

2.6 Proposed Memory Reduction Methodology for DWT/ IDWT

Considering wavelet computation in dyadic space for frame-length (p) = number of filter coefficients; and assuming i^{th} level of resolution consists of j number of filter coefficients, the following relationships hold: $\forall i \in [1, \log_2 p]$, $j \in [1, p/2^i]$, where i and j are integers. Now if the j^{th} coefficient consists of s number of data samples, considering the causality of the system, s can be represented as:

$$s = 1 + (j-1)2^i \tag{2.9}$$

The maximum number of samples (s_{max}) present at the i^{th} level of resolution can be obtained by substituting $j = p/2^i$ in (2.9) which leads to :

$$s_{max} = 1 + p - 2^i \tag{2.10}$$

It is to be noted from (2.10) that $(s_{max} - 1)$ is an even number. In this methodology, $(s_{max} - 1)$ number of samples is divided into k sub-frames. Each of the sub-frames consists of a pair of data samples x_{n-1} and x_n. As shown later in this Section, this approach will lead to reduction in memory. k can be represented as:

$$k = (p/2) - 2^{(i-1)} \tag{2.11}$$

At each time instant the binary value of m^{th} bit of x_{n-1} and x_n are checked and depending on their combination the appropriate linear combination of the filter coefficients is fetched from the memory (as can be obtained by expanding (2.7)). However, one of these combinations is '00' which means no filter coefficient needs to be multiplied with the input data. As an example in (2.8) if '$x_{1,3}x_{0,3}$' = '00' then multiplication of these input bits with h_1 and h_2 will always yield 0. Thus one needs to store the combinations of filter coefficients corresponding to only three bit-combinations of the sample pair (namely '01', '10' and '11'). Arranging $(s_{max} - 1)$ samples from s_{max} samples at the i^{th} resolution level, we are left with only one sample. Since m^{th} bit of this sample can assume either '0' or '1', only one combination of filter coefficients needs to be stored in the memory corresponding to this sample. Thus, the total memory requirement (M_i) in this methodology for i^{th} level of resolution can be given by:

$$M_i = k \times 3 + 1 = [(p/2) - 2^{(i-1)}] \times 3 + 1 \tag{2.12}$$

The memory requirement for the analysis bank can be computed by summing M_i for all i. But, at the last resolution level of the analysis bank, all is left is one residue signal along with the wavelet coefficient. This residue signal consists of one sample for which, as discussed above, two combinations of filter coefficients are possible in which one is '0'. Therefore, the combination of filter coefficients corresponding to this residue signal occupies one memory block only. Thus, according to this proposed methodology, the Total Memory Requirement (TMR) can be represented as:

$$TMR = 1 + \sum_{i=1}^{\log_2 p} M_i \tag{2.13}$$

From (2.13) it is evident that TMR grows nearly linearly with the increase of frame-length.

Since the filter coefficients used in this proposed methodology are generic, these can be reconfigured to generate any type of mother wavelet and thus can be used for any kind of WT architecture design.

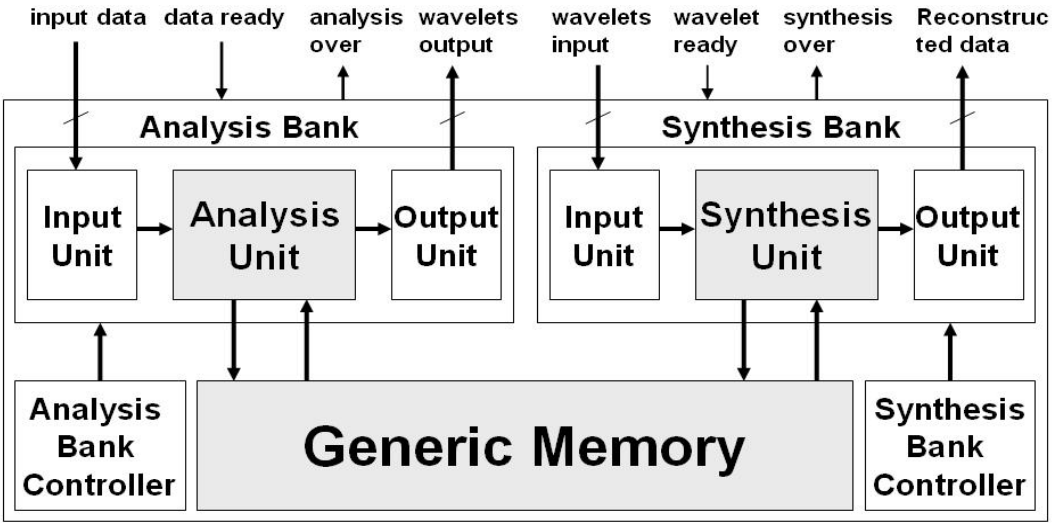


FIGURE 2.7: Proposed VLSI Architecture of DWT and IDWT

2.7 Architectural Overview

The block diagram of the DWT/ IDWT architecture is shown in Fig. 2.7. At the block level the architecture is similar to a standard DA based architecture. However, the main novelty of the architecture lies in the formulation of a new addressing scheme and the corresponding address generation unit design for the reduced memory unit (shown as Generic Memory in Fig. 2.7) discussed in Section 2.5.

In a typical DA based approach the bits of the input samples are checked and addresses are generated to fetch data from the Memory. These data are then used in the Analysis Bank to compute DWT by applying (2.8) in (2.2) and (2.3). Similarly, IDWT is computed in the Synthesis Bank by applying (2.8) in (2.5).

2.7.1 The Memory Unit

As outlined in Section 2.5, in the memory unit of this architecture only the non-repetitive combinations of filter coefficients are stored from the set of data. The strategy for address generation is explained here with an example of frame-length 16, for the 1^{st} resolution level of the analysis bank. Fig. 2.8 shows the incoming data-samples and the associated filter coefficients required for computing each wavelet coefficient in this case. It can be noted from Fig. 2.8 that the filter coefficients are the same for different samples present in the same inclined slices. For example, as shown in the encircled regions in Fig. 2.8, the filter coefficients associated with the samples for computing the 3^{rd} wavelet coefficient are the same for the last five samples of the 4^{th} wavelet coefficient computation. Mathematically, the filter coefficients associated with the q^{th} data sample of r^{th} and $(q+2)^{th}$ data sample of $(r+1)^{th}$ wavelet coefficients are the

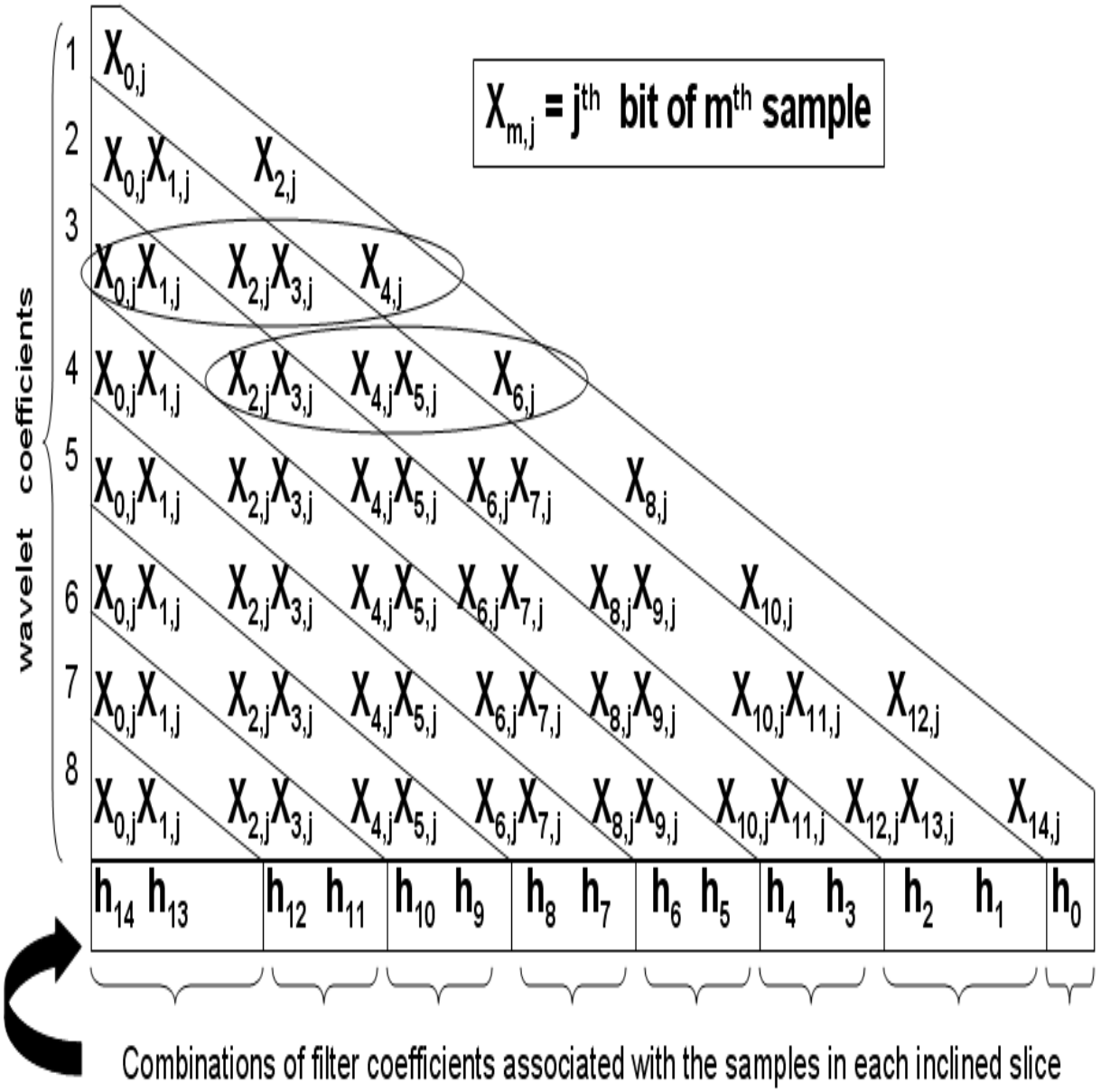


FIGURE 2.8: Example of address generation logic for 1^{st} resolution-level wavelet coefficients with frame-length 16.

same (symmetry property). This symmetry can be represented in generalized form for i^{th} level of resolution as follows:

$$x_m^j = x_{m+2^i}^{j+1} \tag{2.14}$$

where, m = sample number and j = wavelet coefficient. While designing the address generation logic, this symmetry has been exploited at different levels of resolution. Table 2.2 shows how the appropriate filter coefficients are fetched from the memory for different combinations of j^{th} bit of samples x_2 and x_3 for the 3^{rd} wavelet coefficient computation. In this way, we effectively deal with the first issue related to the new addressing scheme raised in section 2.5.

2.7.2 Analysis and Synthesis Bank

The Analysis and Synthesis banks are designed following the equations given in [48] and [52], which represent wavelet analysis and synthesis in dyadic space. DA based filter design technique has been applied to realize the FIR filters of these blocks which replaces multiplication operations by additions only and thereby making the entire architecture multiplier-less. In addition; there are two control units responsible for controlling the analysis and synthesis banks. As shown in Fig. 2.7, two active-low single bit signals- "data ready" and "wavelet ready" enables the Analysis Bank and Synthesis Bank respectively. These signals may be enabled by the user or the previous processing unit in case of a complete system. When the output of these banks are ready, the respective controllers produce two active-low single bit signals - "analysis over" and "synthesis over" which indicates the completion of the process.

The complete architecture is designed using fixed-point 2's complement arithmetic for frame-length 16 with each sample having 16-bit word-length. The complete architecture is modelled in VHDL and is synthesized using 0.13 μm CMOS standard cell library. The address generation logic has been implemented following the relationship described in (2.14) and Fig. 2.8.The implementation results are discussed next.

TABLE 2.2: Example of address generation logic samples x_2 and x_3 of 3^{rd} wavelet coefficient in 1^{st} resolution level

$x_{2,j}$	$x_{3,j}$	Memory
0	0	no fetch
0	1	h_1
1	0	h_2
1	1	$h_1 + h_2$

2.8 Performance Analysis

2.8.1 Hardware Cost

It was mentioned in section 2.5 that in the proposed methodology there is a possibility that the area saving due to reduction of memory size may be outweighed by the requirement of extra adders. This subsection explicitly addresses this issue in terms of total number of transistor saving. To do that the total memory requirement using conventional DA is derived first. For simplicity, here only the memory requirement for the Analysis bank is considered. The memory requirement for the synthesis bank can be derived using the same approach.

Following the same notations adopted in section 2.5, the number of memory block (N_{ij}) required for computing j^{th} wavelet coefficient at the i^{th} resolution level using conventional DA can be given by:

$$N_{ij} = 2^s = 2^{1+(j-1)2^i} \tag{2.15}$$

Since each wavelet coefficient is a convolution sum of the input samples and filter coefficients, computation of j number of wavelet coefficients at the i^{th} level of resolution requires j times application of DA. It means that (2.15) has to be iterated j times where j varies from 1 to $p/2^i$ for each i. Thus the total memory required for i^{th} level of resolution can be expressed as:

$$N_i = \sum_{j=1}^{p/2^i} N_{ij} = 2 \times (2^p - 1)/((4)^{2^{(i-1)}} - 1) \tag{2.16}$$

The memory requirement for the analysis bank can be computed by summing N_i for all i. But, since the last level of resolution consists of one wavelet coefficient and one residue signal which requires two more memory blocks (refer to section 2.5), the total Conventional Memory Requirement (CMR) for the complete analysis bank can be expressed as:

$$CMR = 2 + \sum_{i=1}^{\log_2 p} N_i \tag{2.17}$$

Compared to CMR the total memory requirement in this proposed methodology is given by (2.13). The total hardware cost required in this approach is the summation of TMR and the number of extra adders required for generating the appropriate filter coefficients on-the-fly.

To find out the number of required adders we define a parameter "Adder Penalty" (AP). Unlike TMR, AP is dependent on the number of coefficients present per level of resolution. In the proposed methodology, for computing j^{th} wavelet coefficient at i^{th} resolution level $(s-1)/2$ number of W-bits adders is needed where s is given by (2.9). Denoting the adder requirement for i^{th} stage as P_i and expressing s in terms of j we get:

$$\begin{aligned} P_i &= \sum_{j=1}^{p/2^i} (j-1)2^{(i-1)} \times W \\ &= (p/2^2) \times [(p/2^i) - 1] \times W \end{aligned} \tag{2.18}$$

Thus the total adder requirement for the analysis bank can be given by

$$AP = \sum_{i=1}^{\log_2(p)-1} P_i \tag{2.19}$$

In (2.19), the upper limit of the summation is set to $\log_2(p) - 1$ due to the fact that the last resolution level does not need any addition because of the presence of only one

sample in the corresponding wavelet coefficient. Considering a single-bit memory cell consists of t number of transistors and one single-bit adders consists of Kt number of transistors, Transistor Savings (TS) can be given as:

$$TS = [CMR \times W - (TMR \times W + AP \times K \times W)] \times t \tag{2.20}$$

More specifically, from (2.20), a new metric called Transistor Savings Per Word-length ($TSPW$) can be defined as:

$$TSPW = [CMR - (TMR + AP \times K)] \times t \tag{2.21}$$

To obtain an effective savings in hardware in terms of total transistor count in the proposed methodology $TSPW$ should to be positive.

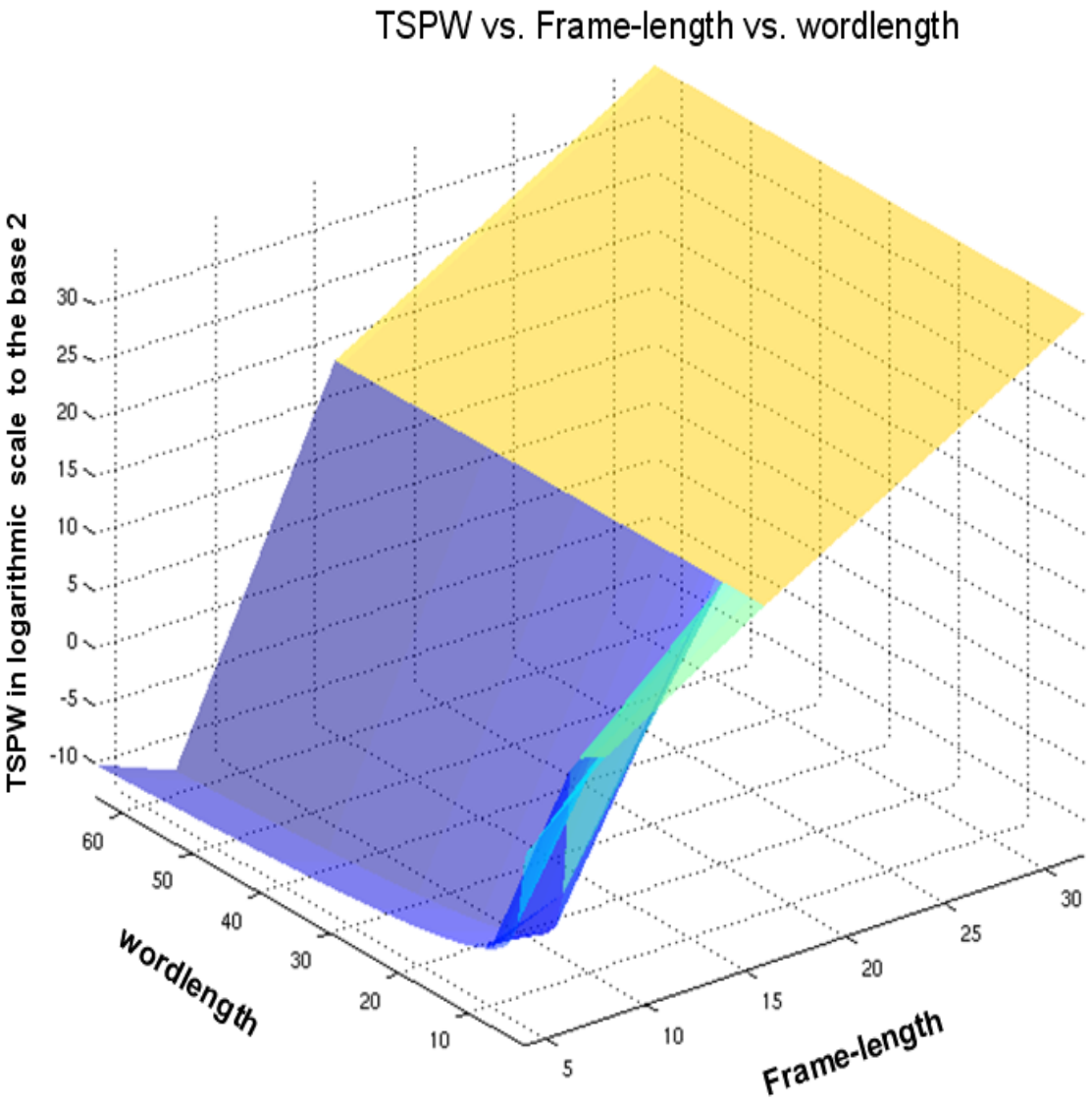


FIGURE 2.9: Variation of TSPW with frame-length and word-length.

Considering a single bit SRAM cell requires 6 transistors and one single bit adder requires

18 transistors [62], the variation of TSPW with four different discrete values of frame-length (= 4, 8, 16 and 32) and word-length is plotted in Fig. 2.9. It can be observed that for smaller frames (< 8), $TSPW$ is negative meaning that for frame-length < 8, the proposed methodology does not achieve hardware savings over conventional DA based method. However for frame-length = 8, TSPW is positive for word-length up to 7, but above that $TSPW$ becomes negative. This means that over word-length 7, the rate of quadratic growth in AP dominates over the rate of exponential growth of CMR. But for longer frame-length (> 8), $TSPW$ increases for a fixed word-length. This means for longer frames, the value of CMR dominates over the summation of AP and TMR resulting in hardware savings with the proposed methodology. It is to be noted that for longer frames also, keeping the frame-length fixed, $TSPW$ starts falling gradually with the increase in word-length. But, this falling rate is much less compared to the rate of increase in CMR for longer frames. Following things are also to be observed from this figure. For the word-length < 8, TSPW improves approximately by 2^8 from the frame-length 8 to 16. When word-length = 8, as mentioned above, TSPW becomes negative for frame-length = 8 for the proposed algorithm. However, when frame-length = 16, TSPW is positive throughout and its improvement over the frame-length 8 for the fixed word-length = 8 is approximately by 2^{22} which increases with the higher word-length by an approximate amount of 2^{30}. Such improvement in TSPW from 2^8 to 2^{22} to 2^{30} can also be observed as the sharp discontinuity in the Fig. 2.9. This overall analysis demonstrates that this proposed methodology is suitable for longer frames.

2.8.2 Functional Validation and Error Analysis

To do the functional validation of the proposed methodology a wavelet analysis model is generated in C language running on a PC. The functional output of the C-model is compared with the VHDL model of the proposed methodology based architecture. Throughout the experiment, frame-length is considered as 16. As test vectors, $2,64,000$ samples of a female speech signal recorded in real life has been used. The comparison result is shown in Fig. 2.10. For brevity we have shown only the wavelet coefficients generated at the first level of resolution for the analysis bank. The result from the C-model are shown on the left side of Fig. 2.10 while the hardware output as simulated from VHDL description are shown on the right side. It is evident that the designed architecture shows functional similarity to the software model.

However, because of fixed-point implementation the actual hardware is prone to have errors due to finite wordlength representation of the input data as well as the truncation error which gets accumulated at every arithmetic operation. To examine the overall effect of these sources of error (Fig. 2.11), the probability of error with respect to bit position derived from the VHDL implementation has been plotted. It is to be noted that error probability reaches to 0.05 at the 9^{th} bit position which is ignorable for all

practical purposes. The maximum probability of error (0.18) occurs at 21^{st} bit-position which physically means that in this case 16-bit wordlength can be considered as a good practical choice.

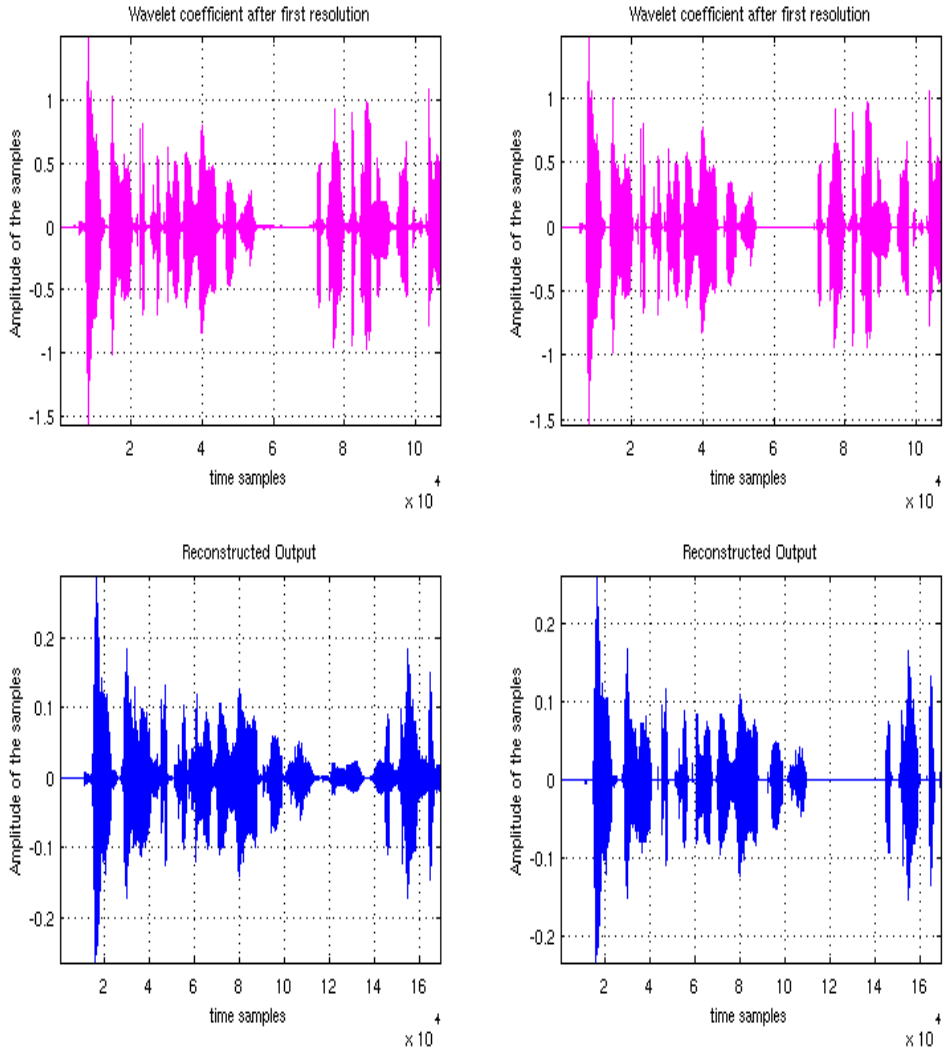


FIGURE 2.10: Comparative study between software and hardware results. Left hand and right-hand columns represent C-model and VHDL-model generated results respectively. The top row represents wavelet coefficients of the first resolution level and bottom row represents the reconstructed output.

2.8.3 Comparison with Other Architectures

The proposed architecture is synthesized by Synopsys Design Compiler (DC) using 0.13 μm standard cell CMOS technology. The synthesized area and power consumption of the proposed methodology based architecture are 6.5 mm^2 and 46.8 μW @ 1 MHz frequency for V_{DD} = 1.2 V. The power value is obtained by feeding continuously 16-bit $2,64,000$ random vectors into the synthesized netlist and applying Synopsys Prime Time.

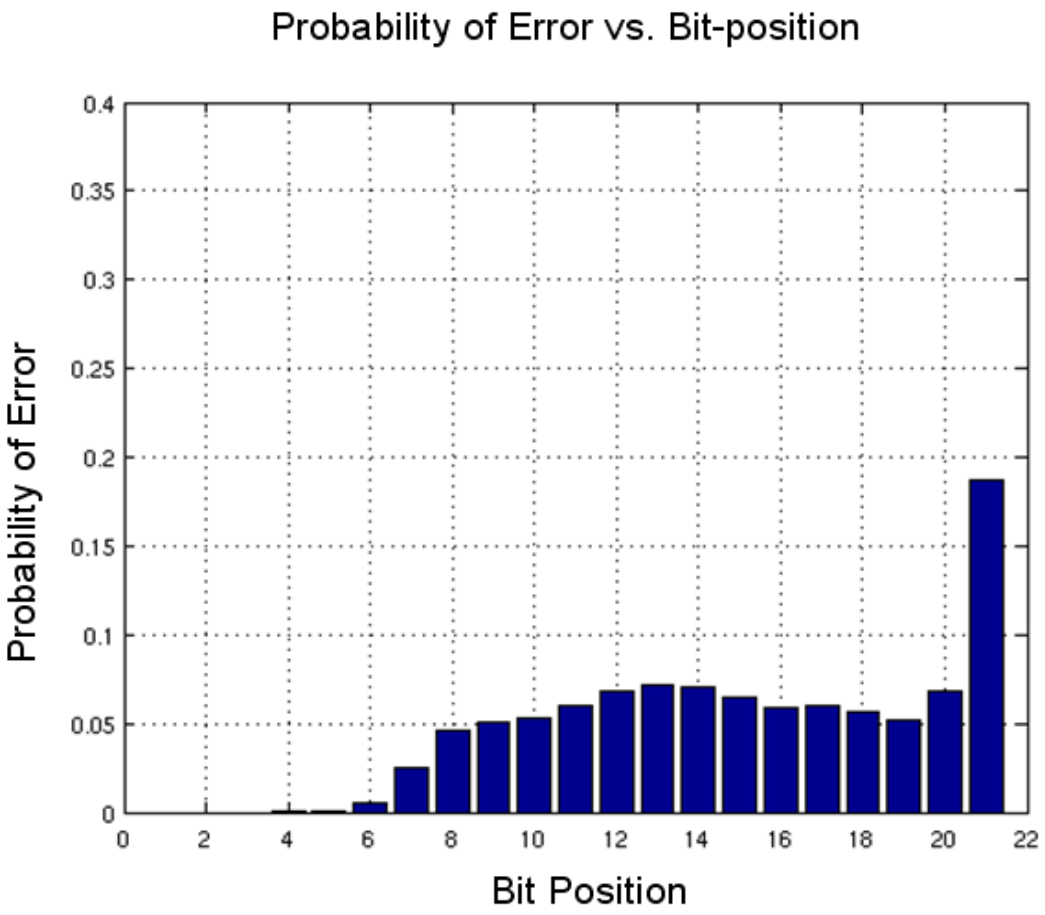


FIGURE 2.11: Probability of Error vs Bit Position in the proposed architecture.

Table 2.3 shows the comparison of area requirement and power consumption of the proposed methodology based DWT/ IDWT architecture with previously proposed architectures. Since different architectures use different technologies it is unfair to compare them on a one-to-one basis. However, these results are provided to give an insight about the performance of the proposed methodology based architecture. Most of the results shown in Table 2.3 are taken from [60]. Table 2.3 shows that the proposed memory reduction methodology based DWT/ IDWT architecture compares very favorably in terms of area and power with respect to the other reported architectures. It is to be noted that due to unavailability of appropriate memory module in our standard cell library, the architecture is implemented using registers. We believe that the use of appropriate memory will reduce the area and power consumption significantly.

2.8.4 More Results on Architectural Implementation

To give an insight into the performance of the proposed architecture, Fig. 2.12 shows that latency of Analysis and Synthesis Bank are 12 and 20 cycles respectively. In Fig. 2.12 $x_{n,k}$ corresponds to n^{th} input data to Analysis Bank for k^{th} Data Frame, $s_{i,j}$ corresponds to i^{th} reconstructed data as output of the Synthesis Bank for j^{th} frame, $w^{m}_{p,q}$ corresponds

TABLE 2.3: Comparisons of the proposed architecture with other reported architectures

Architecture	Word length [bits]	Frequency [MHz]	Technology [μm]	Area [mm^2]	Power [mW]
[53]	8	20	1.2	70	500 - 1000
[54]	8	25	0.8	8.5	855
[55]-I	16	50	0.35	25	35.66
[55]-II	16	50	0.35	25	29.43
[56]	—	—	—	—	33.90
[58]	—	—	0.13	—	12.88
[57]	13	–	—	—	15.15
[60]	—	200	0.13	—	9.74
Proposed	16	1	0.13	6.5	0.0468

to p^{th} wavelet in m^{th} stage corresponding to q^{th} frame as output from the Analysis Bank and input to Synthesis Bank. As can be seen, the first wavelet output of the first stage for the first data-frame ($w^1_{0,1}$) appears after 12 clock cycles from the first input sample of the first data-frame ($x_{0,1}$) as expected following the explanation of the Analysis Bank in Section 2.6. Then onwards wavelets are available at consecutive clock cycles and the residue ($r^4_{0,1}$) after the final stage for the first data-frame is available at 27^{th} clock cycle. This input-output relationship of Analysis Bank is shown in 'white'-coloured blocks in Fig. 2.12. Similarly, it can be seen from 'gray'-coloured blocks in Fig. 2.12, the first reconstructed data output corresponding to first data-frame ($s_{0,1}$) is available after 20 clock cycles from the first input wavelet ($w^1_{0,1}$) to the Synthesis Bank corresponding to first frame. Then onwards each reconstructed data are obtained at consecutive clock cycles and last reconstructed data ($s_{15,1}$) corresponding to first frame is obtained at 35^{th} cycle.

2.9 Case Study: Fetal ECG Extraction Using WT

One important application of signal processing in healthcare is to extract physiological signal from mixed multi-sensory data [67]. Extraction of Fetal Electrocardiogram (FECG) from abdominal composite signal of mother is an important example of this. Typically, the recorded signal is the mixture of weak FECG, a strong Maternal ECG (MECG), and random contaminations due to other non-cardiac sources [68]. Because of this, extraction of FECG from the composite signal is a challenging problem in signal processing. Different available techniques for efficient detection and extraction of FECG from the mixed signal are overviewed in [3]. The drawbacks of the available methods are their simplistic assumptions namely additive model for the composite signal and high correlation between the thoracic and maternal component of abdominal signal profiles [69]. To solve this problem a robust method has been reported in [69] to extract FECG from the composite signal. But this method deals with complex signal processing tasks

Cycle	Analysis Bank		Synthesis Bank		Cycle	Analysis Bank		Synthesis Bank	
	Input	Output	Input	Output		Input	Output	Input	Output
1	$X_{0,1}$		$W^1_{0,1}$		21	$X_{4,2}$	$W^2_{1,1}$	$W^1_{4,2}$	$S_{1,1}$
2	$X_{1,1}$		$W^1_{1,1}$		22	$X_{5,2}$	$W^2_{2,1}$	$W^1_{5,2}$	$S_{2,1}$
3	$X_{2,1}$		$W^1_{2,1}$		23	$X_{6,2}$	$W^2_{3,1}$	$W^1_{6,2}$	$S_{3,1}$
4	$X_{3,1}$		$W^1_{3,1}$		24	$X_{7,2}$	$W^3_{0,1}$	$W^1_{7,2}$	$S_{4,1}$
5	$X_{4,1}$		$W^1_{4,1}$		25	$X_{8,2}$	$W^3_{1,1}$	$W^2_{0,2}$	$S_{5,1}$
6	$X_{5,1}$		$W^1_{5,1}$		26	$X_{9,2}$	$W^4_{0,1}$	$W^2_{1,2}$	$S_{6,1}$
7	$X_{6,1}$		$W^1_{6,1}$		27	$X_{10,2}$	$r^4_{0,1}$	$W^2_{2,2}$	$S_{7,1}$
8	$X_{7,1}$		$W^1_{7,1}$		28	$X_{11,2}$	$W^1_{0,2}$	$W^2_{3,2}$	$S_{8,1}$
9	$X_{8,1}$		$W^2_{0,1}$		29	$X_{12,2}$	$W^1_{1,2}$	$W^3_{0,2}$	$S_{9,1}$
10	$X_{9,1}$		$W^2_{1,1}$		30	$X_{13,2}$	$W^1_{2,2}$	$W^3_{1,2}$	$S_{10,1}$
11	$X_{10,1}$		$W^2_{2,1}$		31	$X_{14,2}$	$W^1_{3,2}$	$W^4_{0,2}$	$S_{11,1}$
12	$X_{11,1}$	$W^1_{0,1}$	$W^2_{3,1}$		32	$X_{15,2}$	$W^1_{4,2}$	$r^4_{0,2}$	$S_{12,1}$
13	$X_{12,1}$	$W^1_{1,1}$	$W^3_{0,1}$		33	$X_{0,3}$	$W^1_{5,2}$	$W^1_{0,3}$	$S_{13,1}$
14	$X_{13,1}$	$W^1_{2,1}$	$W^3_{1,1}$		34	$X_{1,3}$	$W^1_{6,2}$	$W^1_{1,3}$	$S_{14,1}$
15	$X_{14,1}$	$W^1_{3,1}$	$W^4_{0,1}$		35	$X_{2,3}$	$W^1_{7,2}$	$W^1_{2,3}$	$S_{15,1}$
16	$X_{15,1}$	$W^1_{4,1}$	$r^4_{0,1}$		36	$X_{3,3}$	$W^2_{0,2}$	$W^1_{3,3}$	$S_{0,2}$
17	$X_{0,2}$	$W^1_{5,1}$	$W^1_{0,2}$		37	$X_{4,3}$	$W^2_{1,2}$	$W^1_{4,3}$	$S_{1,2}$
18	$X_{1,2}$	$W^1_{6,1}$	$W^1_{1,2}$		38	$X_{5,3}$	$W^2_{2,2}$	$W^1_{5,3}$	$S_{2,2}$
19	$X_{2,2}$	$W^1_{7,1}$	$W^1_{2,2}$		39	$X_{6,3}$	$W^2_{3,2}$	$W^1_{6,3}$	$S_{3,2}$
20	$X_{3,2}$	$W^2_{0,1}$	$W^1_{3,2}$	$S_{0,1}$	40	$X_{7,3}$	$W^3_{0,2}$	$W^1_{7,3}$	$S_{4,2}$

FIGURE 2.12: Latency in Analysis and Synthesis Bank in the designed architecture

including WT, fuzzy membership function computation, reconstruction which increase the complexity of the circuit in terms of extensive memory and power requirement.

Here, a VLSI architecture for above mentioned problem is envisaged based on the proposed memory reduction methodology of DWT/IDWT discussed in Section 2.6 and also presented in [69]. The novelty of this design would be the development of a memory-efficient, multiplierless, parameterized DWT and IDWT architecture and corresponding decision-making circuitry which eliminates complex fuzzy membership function computation [69].

As discussed in Section 2.8, this approach reduces the hardware complexity and power consumption significantly and thus it will be able to reduce the overall complexity of such FECG extraction architecture. Moreover, since there is no intuitive way to know which mother wavelet to choose [70], this proposed generic methodology lets the user freedom to choose any mother wavelet suitable for the application by simply reconfiguring the generic memory.

However detailed theoretical background on FECG extraction and its corresponding architectural description based on the proposed low complexity DWT technique along with a novel simplified methodology for fuzzy membership function computation unit will be discussed in Appendix-B from the conceptual point of view. It will be shown there that the whole FECG extraction system consumes 101.5 μW power and occupies 14.2 mm^2 silicon area using 0.13 μm technology at 1 MHz target frequency at the post-synthesis level.

2.10 Concluding Remarks

In this chapter, a novel memory reduction methodology of DWT/ IDWT architecture was proposed. The DWT/ IDWT architecture based on this methodology has been demonstrated. It has been shown that this proposed architecture resulted in low power and area consumption. This architecture is generic and can be reconfigured for any type of mother wavelet. It has also been shown that the DWT/ IDWT architecture consumes 46.8 μW power and 6.5 mm^2 silicon area using 0.13 μm technology at 1 MHz target frequency.

Chapter 3

Low Complexity 2-Dimensional FastICA

It has been pointed out in Chapter 1 that in emerging applications such as WSN and remote health monitoring signal separation is one challenging task. In a large number of cases, the signal received by a sensor (antenna, microphone etc.) is actually the mixture of several signal sources. For example, the signal received by an antenna is a superposition of signals emitted by all the sources which are in its receptive field. Generally, sources as well as mixtures are unknown and these are estimated without any prior knowledge of the sources - this problem is called "Blind Source Separation" (BSS)[1] [88]. The lack of prior knowledge about the mixture is compensated by a statistically strong but often physically plausible assumption of independence between the source signals. This so-called blindness is not the weakness, but the strength of the BSS model [89], making it a versatile tool for exploiting the spatial diversity provided by an array of sensors. Independent Component Analysis (ICA) is a concept which is employed for solving this BSS problem [101]. ICA has very well been formulated in [90] and [91] and also its various applications are discussed. In this chapter we investigate the impact of ICA algorithmic parameters on its corresponding architectural implementation which will lead to the development of a novel low-complexity 2-dimensional ICA algorithm.

The rest of this chapter is organized as follows. Section 3.1 does a brief literature review on ICA and its applications and identifies FastICA as one of the most popular among the existing ICA algorithms. Section 3.2 emphasizes on the necessity of the algorithm-architecture holistic optimization approach of ICA if it has to be applied in the resource constrained environment. Section 3.3 provides the preliminaries of the conventional FastICA. Section 3.4 identifies the FastICA algorithmic parameters which are believed

[1]The contents of Section 3.8 - Section 3.10 have partly appeared as "Hardware Efficient Fixed-Point VLSI Architecture for $2D$ Kurtotic FastICA" by Acharyya *et. al.* in 19^{th} *IEEE European Conference on Circuit Theory and Design* and "Hardware Reduction Methodology for 2-Dimensional Kurtotic FastICA Based on Algorithmic Analysis and Architectural Symmetry" by Acharyya *et. al.* in *IEEE Workshop on Signal Processing Systems*. Please see Appendix-A for further detail.

to have significant impact on the corresponding architecture design. Section 3.5 discusses the basics of statistical data modeling techniques and Section 3.6 presents the generic signal modeling strategies broadly classifying whole signal domain into stationary and non-stationary signals. Section 3.7 provides the experimental results showing the impact of these algorithmic parameters on architecture design and provides a guideline for the hardware designer for hardware implementation of FastICA. Section 3.8 proposes a novel algebraic methodology for hardware reduction in 2-D FastICA architecture through detailed algorithmic analysis and exploiting the resulting architectural symmetry and Section 3.9 presents the corresponding architectural details. The hardware efficiency is achieved in the proposed methodology through (i) removal of division operation for eigenvector computation, (ii) replacement of division operations by multiplications and (iii) reduction of number of multipliers and adders for whitening matrix computation. In Section 3.10, measures of hardware saving are also provided for generic 2-D FastICA using the proposed methodology and subsequently, as an example, a 16-bit fixed-point 2-D FastICA architecture is developed for frame-length 512 is designed and is shown to consume significantly low power and silicon area.

3.1 ICA, Its Applications and FastICA

ICA has potential applications in diverse fields including healthcare [28], [74]. In [92] the importance of ICA in brain science has been discussed in detail. Thrust has been given there on developing novel and advanced ICA algorithm to analyze multi-sensory biomedical signals, especially electroencephalographic (EEG), magnetoencephalographic (MEG) and electromyographic (EMG). Novel and innovative techniques in BSS are required in Brain Science since brain sources are unidentified, extremely weak, non-stationary, distorted by noise and other interferences. On the basis of ICA and other intelligent algorithms, it is possible in near future to develop and design intelligent electronic devices or machines that similar to human being will be able to have selective attention, enhance speech and recognize voices, odors or human faces [92].

In [93] and [7] the application of ICA in healthcare has been described considering one important task in biomedical signal processing - separating heart sound from lung sound. Acoustical analysis of lung sounds provides important and helpful information in the diagnosis and monitoring of lung diseases. But, for the cardiologists, the main interest is in heart sound and lung sound is considered as noise. So, both of these signals are of interest for different set of medical practitioners. Lung and heart sounds are considered as independent source signals. So, separating the mixed signals recorded on the skin is of utmost importance. In [93], the applicability and feasibility of ICA for such signal processing problem has been studied and in [7], ICA has been applied successfully for separating these two signals.

Application and quality assessment of different available ICA algorithms have been done in [94] on EEG recordings to separate the independent source signals as well as to remove the possible artifacts in those recordings. A comparative study of different ICA algorithms - JADE, BS Infomax, FastICA and EGLD ML - has been performed in terms of absolute activation correlations, variance in the reconstructed signals, absolute correlation between between source signals and corresponding ICA estimated activations, Signal to Interference Ratio (SIR), Cross Talk Error (CTE) and CPU processing time [94]. Depending upon all these figures of merit, the performance of these algorithms has been classified and it has been found that FastICA is the superior of them all [94] and it outperforms most of the existing ICA algorithms in terms of terms of higher convergence speed [75].

According to [95], ICA has recently demonstrated considerable promise in characterizing functional Magnetic Resonance Imaging (fMRI) data. ICA has been successfully used in a number of exciting fMRI applications, including the identification of various signal types such as task and transiently task-related and physiology related signals in the spatial and temporal domain [95]. ICA especially FastICA can be applied to the feature extraction of pulse wave of human being [96]. Pulse wave is the rhythmic change of the inflow of blood to the different parts of the body in the form of waves as measured from the outside of the body. This wave is modified by the physiological conditions such as the heart beat, the circulation of the blood, and changes in the state of the minor artery system, which leads to the distortion of the shape of the waves [96]. FastICA can be successfully applied to remove out these distortions and get back the original pulse wave.

ICA can also be applied in Electrocardiography (ECG) [74]. Routinely recorded ECG are often corrupted by different types of artefacts and many methods have been made to enhance their quality by reducing the noise and artefacts [6]. ICA has shown tremendous potential for artefacts removal and has been successfully used in [6], [156]. ICA can also be applied to separate spinal cord motor signals [97]. ICA has also got potential applications in detecting fetal arrhythmias from fetal Magnetocardiographs (fMCG) obtained by ultra-sonographic imaging [98]. Besides, other applications of ICA algorithm has been well depicted in [99] and part IV of the book [100] with necessary minute detail including feature extraction, brain imaging, telecommunication and multiuser detection.

3.2 Necessity for Algorithm-Architecture Holistic Optimization for ICA

Although different algorithms for ICA have been reported, the FastICA algorithm has been shown to have advantage in terms of convergence speed [75]. Till now, most of the publications concentrated on off-line computation of FastICA. But, the problem

in that case, such methods can not be applied in real time applications. In the last section we have already seen that FastICA can be used broadly in biomedical signal processing especially to identify artifact and interference from the noisy sensory data of EEG, ECG, MEG etc.. So, architectural studies would be ideal for the hardware implementation of this algorithm [73] so that it can be applied in such real time signal processing applications. In [73], it has been identified that VLSI implementation of ICA algorithm requires extremely efficient hardware designs and sufficient IC resources and it has been anticipated that due to the rapid growth in VLSI industries, significant solutions of complex ICA algorithms will be possible. However there have been very few studies about the real-time implementation of ICA in hardware and so there was no publication on VLSI implementation of FastICA algorithm even a year ago. One reason behind it may be FastICA in its present form requires complex mathematical operations including division, square root evaluation and multiplication and thus demands large silicon area and power consumption and thus is not suitable for resource constrained applications like mobile healthcare because the body-worn sensors are typically battery powered, unobtrusive, tiny and are expected to operate for long time.

Recently, the first VLSI implementation of FastICA based on floating-point arithmetic has been reported in [76]. This is direct algorithm into architecture mapping and such implementation contributes significantly to the silicon area and consume significant power. Thus fixed-point arithmetic may be an optimal choice for low complexity hardware design of FastICA although a compromise with accuracy may be necessary. However, given the assiduous attention the FastICA algorithm has received over the last decade, we believe, there are still potential gaps in algorithm and architecture holistic optimization approach. Therefore it is essential to perform a thorough analysis of the impact of several algorithmic parameters on the corresponding architecture and we believe, this can lead to optimum implementation of FastICA. Unfortunately, to the best of our knowledge, no clear procedure exists so far which can be used straightway to perform such analysis. This motivates us to propose a generalized procedure for such analysis on algorithm-architecture holistic optimization approach which can be used as hardware designer's guideline for implementation of FastICA on hardware in an optimum fashion. Based on this procedure, later on in this chapter we will perform our initial research study on designing a low complexity fixed-point arithmetic based VLSI architecture for FastICA algorithm and show that it will consume low power and occupy small silicon area.

3.3 Preliminaries of FastICA

Denoting independent sources by S, mixed signal (X) can be defined as [76]:

$$X = AS \tag{3.1}$$

where, n = number of independent sources, $X = \{x_i\}$, $S = \{s_i\}$, $i \in (1, n)$; A is a full-rank $n \times n$ mixing matrix; $s_i = \{s_{i,j}\}$, $x_i = \{x_{i,j}\}$ where $j \in (1, m)$ and m is equal to the frame-length. FastICA comprises of two steps - Preprocessing and FastICA Iteration.

3.3.1 Preprocessing

Preprocessing can further be divided into to steps - Centering and Whitening. In Centering, mean vector of $x_{i,j}$ is computed and then subtracted from $x_{i,j}$ to make the mixed signal a zero-mean matrix. It can be represented as [101],[99]:

$$x_{i,j} = x_{i,j} - \mathcal{E}[x_{i,j}] \tag{3.2}$$

where $\forall i \in (1, n)$, $j \in (1, m)$.

The next preprocessing step is whitening which linearly transforms vector X into a new vector Z which is white, i.e., its components are uncorrelated and their variances are equal to unity. One way to do this is Eigen Value Decomposition (EVD) which can be defined as [101],[99]:

$$\mathcal{E}[XX^T] = C_X = EDE^T \tag{3.3}$$

where C_X is the covariance matrix of X; $E = \{e_i\}$ is the orthogonal matrix of eigenvectors e_i of C_X and $D = diag(\{d_i\})$ is the diagonal matrix of its eigenvalues $\forall i \in (1, n)$. The whitening process can be described by [101],[99]:

$$Z = PX \tag{3.4}$$

where $Z = \{z_i\}$ and P is the whitening matrix and can be defined as [101],[99]:

$$P = D^{-1/2} \times E^T \tag{3.5}$$

The utility of whitening resides in the reduction of number of parameters to be estimated [101],[99].

3.3.2 FastICA Iteration

The FastICA algorithm produces the estimated output vector ($\widehat{S}$) from the whitened mixed signal vector (Z)by estimating the unmixing matrix B of dimension $n \times n$ which can be defined as [101],[99]:

$$\widehat{S} = B^T Z \tag{3.6}$$

The k^{th} column of B represents the weight vector w_k associated with k^{th} estimated IC where $k \in (1, n)$. FastICA algorithm for estimating n ICs can be given as follows

[101],[99]:

(i) Choose n, the number of ICs to estimate and set counter $k \leftarrow 1$.

(ii) Choose an initial value of unit norm of $\mathbf{w_k}$.

(iii) FastICA introduces the optimized contrast function (g) within the basic iterative equation as follows:

$$\mathbf{w_k} \leftarrow \mathcal{E}\{Z\dot{g}(\mathbf{w_k}^T Z)\} - \mathcal{E}\{\ddot{g}(\mathbf{w_k}^T Z)\}\mathbf{w_k} \tag{3.7}$$

where $\dot{g}$ and $\ddot{g}$ are the first and second derivative of contrast function g respectively. Using Kurtosis-based contrast function in (3.7) we get [101],[99]:

$$\mathbf{w_k} \leftarrow \mathcal{E}\{Z(\mathbf{w_{k-1}}^T Z)^3\} - 3\mathbf{w_{k-1}} \tag{3.8}$$

(iv) To prevent different weight vectors w_k from converging to the same maxima, the next step is to orthogonalize the vectors after every iteration. Considering the *deflationary orthogonalization* and following Gram-Schmidt orthogonalization, this step can be defined as [101],[99]:

$$\mathbf{w_k} \leftarrow \mathbf{w_k} - \sum_{j=1}^{k-1}(\mathbf{w_k}^T \mathbf{w_j})\mathbf{w_j} \tag{3.9}$$

(v) Normalize w_k as follows [101],[99]:

$$\mathbf{w_k} \leftarrow \mathbf{w_k}/\|\mathbf{w_k}\| \tag{3.10}$$

(vi) If $\mathbf{w_k}$ has not converged, go back to step (iii), if converged go to next step.

(vii) Set $k \leftarrow k + 1$. If $k \leq n$, go back to step (ii).

Now using computed B in (3.6) estimated output $\widehat{S}$ is obtained. The detailed procedure and corresponding block diagrams can be found in [76] and [77].

3.4 Identification of FastICA Algorithmic Parameters from Architectural Perspective

In this section three important FastICA algorithmic parameters are identified which are believed to have tremendous impact on corresponding architectural implementation.

These parameters are - *frame-length*, *Convergence Accuracy* and *Iteration of Convergence*.

3.4.1 Frame-length

Keeping the convergence threshold fixed, with the increase in frame-length iteration steps for convergence decreases [102]. From the point of view of algorithm, this essentially means increasing frame-length requires less computation time. But from architectural implementation point of view, advantage obtained due to this improved computation speed may get negated because of more hardware requirement in terms of memory and arithmetic operations. Thus it is essential to reach one point where algorithmic performance and its corresponding architectural implementation is optimum.

3.4.2 Convergence Accuracy

In practice, convergence accuracy can be defined as the degree of closeness between two vectors. Referring back to the last section, let us consider $\mathbf{w_k}$ and $\mathbf{w_{k-1}}$ be the normalized vectors computed after k^{th} and $(k-1)^{th}$ FastICA iteration. $\mathbf{w_k}$ converges if the angle ϕ between $\mathbf{w_k}$ and $\mathbf{w_{k-1}}$ is less than or equal to the pre-defined convergence threshold θ (please see Fig. 3.1). If this condition is not met i.e. if $\theta < \phi$, it is decided that $\mathbf{w_k}$ has not yet converged and further iteration is required.

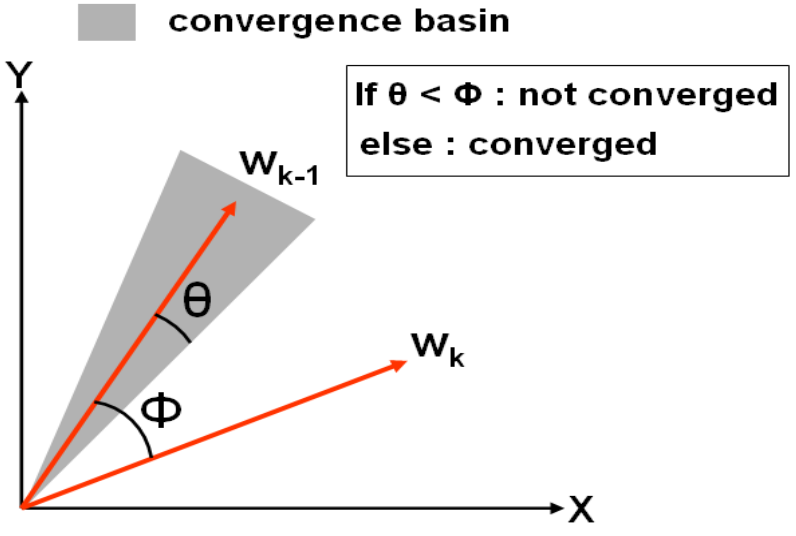


FIGURE 3.1: Concept of Convergence Accuracy in 2-Dimensional (2D) FastICA. θ denotes the convergence basin. Decreasing value of θ indicates more accuracy of convergence.

Convergence threshold θ is a measure of convergence accuracy because smaller is the θ, higher is the convergence accuracy. Keeping the frame-length fixed, if the convergence accuracy is improved by decreasing θ, number of iteration steps for convergence will also

increase. Thus convergence threshold is an essential parameter which has effects on both - algorithmic performance in terms of computational speed as well as on architectural implementation in terms of hardware requirement.

3.4.3 Iteration of Convergence

It has been mentioned in the last section that FastICA needs to perform a number of iterations unless a pre-defined convergence accuracy is achieved. The problem is that in general the number of necessary iterations depends on the nature of data, so it cannot be given in advance. Unlike software implementation, in hardware design on the fly allocation of extra resources for extra iteration stage is not possible. These iterations of convergence make FastICA a computational demanding algorithm, requires a significant amount of hardware resources such as multipliers, adders and memory space. To realize the FastICA for real time signal processing, a finite number of iterations of convergence should be estimated prior to the implementation.

The impact of these afore-mentioned FastICA algorithmic parameters on its corresponding architectural implementation will shortly be investigated. But prior to that, statistical approach of data modeling will be discussed in brief because it is necessary for developing the generic signal models which will be used for the intended investigation.

3.5 Basics of Statistical Data Modeling

Statistical methodology involves the collection, analysis, interpretation, and presentation of data. In this section, basic concepts associated with statistical data analysis are presented in brief.

3.5.1 Random Number Generator (RNG)

Generation of quality random number sequence having attributes of non-repeatability and non-predictability is crucial for statistical simulations [104]. In the early 20^{th} century people started the use of random numbers for different scientific applications. Initially they used mechanical devices to generate random number sequence. As the technology advanced, computers were used to efficiently generate the random number sequences. Some essential properties for good RNGs are as follows:

Long Period: The period length of RNGs should be large, to reliably generate random numbers for simulations.

Uniformity: This is a fundamental requirement for good RNGs, which has a wide period length to produce a uniform random sequence.

Efficiency: The generation method for random numbers should take only few arithmetic operations.

3.5.1.1 Linear Congruential Generator (LCG)

Derrick Henry Lehmer in 1949 introduced the linear Congruential method to generate the random numbers. This algorithm is based on a simple mathematically formula defined as: $X_{n+1} = (aX_n + c)\ mod\ m$, where X_n is the sequence of random values and a, c and m are the non-negative co-efficients for the RNG. The LCG is dependent on the choice of coefficients values. This algorithm is still in use in less sensitive applications [103].

3.5.1.2 Linear Feedback Shift Register (LFSR)

Linear feedback shift registers (LFSRs) are one class of RNGs widely used for low power and high speed applications. A linear feedback shift register is a mechanism to generate a random bit stream [105]. The shifts registers are connected in series and known as cells. The registers are set by some initialization vector. The content of registers is controlled by clock and on every instant of clock; the content of shift registers is moved by one position to right. Feedback mechanism is controlled by XOR function to ensure the generation of random bit sequence. The implementation of LFSR is very easy and straight forward both in software and hardware [105].

3.5.1.3 The Mersenne Twister RNG

In 1998, Makoto Matsumoto and Takuji Nishimura proposed the Mersenne twister algorithm. Within few years time the Mersenne twister has been recognized as a fast and efficient RNG. The algorithm has a long period length of $2^{19937-}1$ and 623-dimensional equi-distribution up to 32 bit accuracy while consuming a working area of only 624 words [106]. It is a modified version of Twisted Generalized Feedback Shift Register RNG.

3.5.2 RNG to be Used for the Generic Signal Modeling

In this research study, the Mersenne Twister algorithm is used for generation of random samples for the simulation in order to ensure that the repetition period is long enough for generation of random signals effectively satisfying the property of independence.

Let N be the number of random numbers required for one signal generation and M be the period length of RNG. So for K number of signals: $K = M/N$, where $M = 4.3 \times 10^{6001}$ from Mersenne Twister algorithm and $N = 3276800$ random numbers are required

for the generation of one signal where each signal consist of 2048 samples. Therefore $K = 1.3 \times 10^{5997}$ number of signals can be generated.

3.5.3 Monte Carlo Approach and Confidence Interval

Monte Carlo Simulation is an important tool in modern scientific research for solving the complex numerical mathematical problems by performing statistical sampling experiments on a computer. It allows modeling of complex systems to determine the performance parameters by simulating the uncertainty. The researchers may need to predict the likelihood of failure of system, which need to work over long period of time [103]. Monte Carlo Simulation uses the random numbers to conduct the experiments. It depends on one essential assumption that RNG gives enough random numbers for the simulation. Simulation is a sampling method; therefore reliability of results is a critical issue. The principles associated with Monte Carlo approach for yielding the reliable results are as follows:

Generation: The generation of random numbers is pivotal for reliable Monte Carlo simulation. The period of RNG should be wide enough to ensure the independence of random numbers. The approximation of performance parameters of system model, which are under investigation, has a greater impact of true random numbers.

Efficiency: Efficiency of the simulation is important for predicting the failure or success of the system. Therefore, realistic efficiency criteria should be decided for the system model to yield the required result in a very short span of time [107].

Accuracy: Accuracy of Monte Carlo simulation depends on the size of samples selected for the estimation of parameters. The accuracy of simulation requires a large set of random samples to approximate the true behaviour of the system.

But due to time and resources constraints, in practice it is impossible to perform Monte Carlo simulation with a large set of random samples. Therefore, to determine the accuracy of Monte Carlo simulation a mathematical function- Confidence interval is used [108]. Some terminologies associated with the confidence interval are defined as follows:

Population: It can be defined as an entire collection of entities from which statistical inference can be drawn.

Samples: It can be defined as a collection of selected units of population.

Confidence Level: It is a probability value related to the confidence interval.

3.6 Proposed Generic Signal Model: Stationary and Non-stationary

In many scientific analyses, researchers need to represent a signal in such a way that the synthetic version of the signal is as close to the original one as possible. In this research the whole signal set is classified into two domains - stationary and non-stationary.

3.6.1 Stationary Signal Modeling

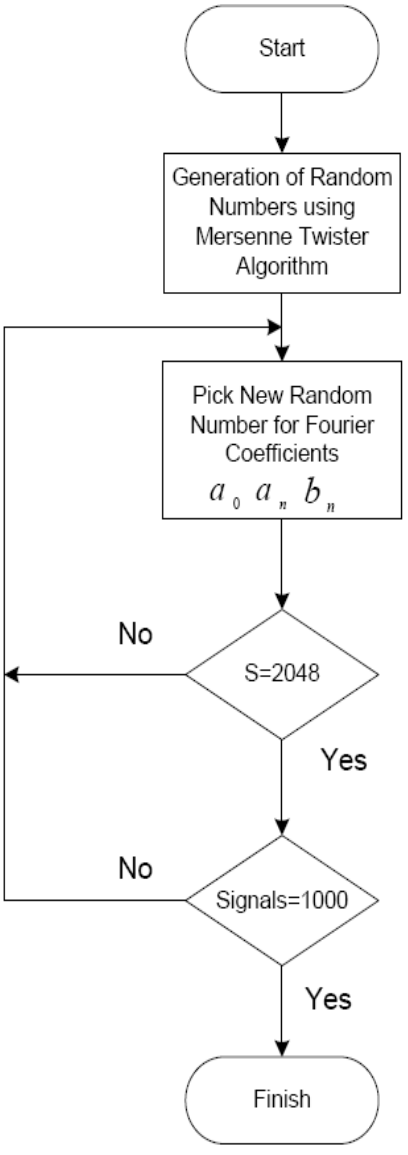


FIGURE 3.2: Stationary signal generation flow diagram developed for the experiment.

Stationary signals are constant in their statistical parameters over time i.e. their joint probability distribution does not change when shifted in time or space. Fourier series is

an established method for representation of stationary signals.

Fourier theory describes that a signal can be expressed as the sum of a series of sines and cosines known as the basis functions for Fourier series. Mathematically Fourier series can be defined as follows [109]:

$$S = \frac{a_0}{2} + \sum_{i=1}^{n}(a_n \cos(nx) + b_n \sin(nx))$$

where,

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,\mathrm{d}x;\ \ a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,\mathrm{d}x;\ \ b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,\mathrm{d}x$$

where a_0, a_n and b_n are known as coefficients of Fourier series and $n = 1, 2, 3...N$.

The stationary signals generation process is shown in Fig. 3.2. Initially random numbers are generated using Mersenne Twister algorithm, which have long period length to ensure the independence of the generated signals. Then these random numbers are used as a numerical values for Fourier coefficients a_0,a_n and b_n. Referring the Fourier series equation shown above, without any loss of generality n is arbitrarily chosen 5 for generating one sample value for our experiments. Considering S be the collection of 2048 data-samples for one stationary signal, this process is continued till 1000 random stationary signals are generated.

3.6.2 Non-Stationary Signal Modeling

However, most of the real life signals encountered in practice in various application areas, such as wireless communication, biomedical science, acoustic signals, radar and sonar signals [110], does not satisfy this stationarity assumption. This puts emphasis on the investigation of time-frequency representation of the non-stationary signals which exhibit time-varying properties where the signal statistics evolve with time. Since the basis functions of the Fourier series are "Sine" and "Cosine" functions which are localized in frequency but not in time, Fourier series may not be the suitable one for representation of such signals because Fourier series can only provide the frequency resolution but no time resolution [109]. Therefore it is important to model the non-stationary signal in a different way.

Several methods have been proposed in literature to analyze the time-varying signals. Wigner distribution (WD), short time Fourier transform (STFT) and Wavelet Transform (WT) are widely known methods to solve the representation problem of time-varying signals. The Wigner distribution method is suitable for those time-varying signals which are concentrated in only one region of the time-frequency plane. Thus, WD method can-

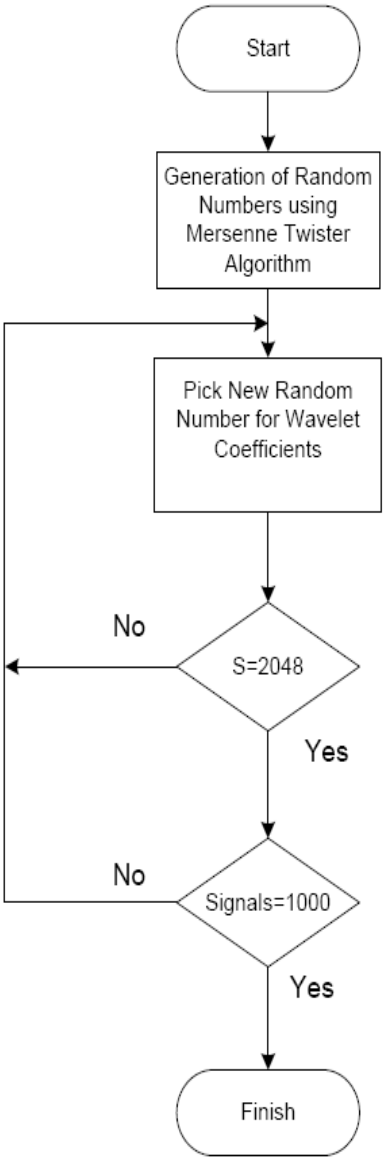


FIGURE 3.3: Non-stationary signal generation flow diagram developed for the experiment.

not distinguish between two closely spaced components and is not appropriate method for analysis of multicomponent non-stationary signals [111].

Among all these afore-mentioned techniques, WT, thanks to its good time-frequency localization property, has been receiving more and more attention for modeling non-stationary processes [112], [113], [114]. Here also WT concept will be used for modeling non-stationary signals in generic fashion. Wavelet theory is already discussed in the last chapter, however the generation of non-stationary signals using wavelets is discussed below.

We have used Symlet and Daubechies wavelets to generate the non-stationary signals. This non-stationary signal generation process is shown in Fig. 3.3. Like stationary sig-

nals generation using Fourier method, here also random numbers are generated using Mersenne Twister algorithm. Then these random number are used as numerical values for Wavelets coefficients. Recalling the basic wavelet analysis and synthesis bank equations from the last chapter, the filter coefficients are generated using Matlab. Considering S be the collection of 2048 data-samples for one non-stationary signal, this process is continued till 1000 random non-stationary signals are generated.

3.7 Impact of Algorithmic Parameters on Architecture Design: Experimental Results and Discussion

This section investigates how the variation of the algorithmic parameters, identified in section 3.4, has profound effect on the corresponding architecture design in terms of hardware complexity and computational delay using our simulation based experiments. Here Transistor Count (TC) is chosen as a metric to represent hardware complexity and delay of a NAND Gate is considered as a metric to represent the computational delay.

It is validated further by providing the post-synthesis results in terms of silicon area and power consumption variation with the variation of these parameters. After that, based on our generic stationary and non-stationary signal modeling, discussed above, we will present the experimental results which we believe can be used as initial guidelines prior to the hardware implementation of FastICA. All the experimental results to be shown here have confidence level $\geq 95\%$.

Although the experimental results to be shown are carried out only on FastICA, this approach can be followed for hardware implementation of any type of ICA algorithms. However, our main aim here is to present an initial guidelines which, if followed before architecture design, could lead to optimum hardware implementation.

3.7.1 Algorithmic Impact on Architecture

Fig. 3.4 shows the variation of transistor counts with different frame-length under practical consideration of four different data-bus width - 4, 8, 16 and 32. We also consider TC (detailed discussion on its justification and derivation is intentionally deferred until section 3.10) is a representative of hardware complexity.

In Fig. 3.4 four separate graphs are plotted for four different data-bus width. Each of the graphs clearly shows that with the increase of frame-length, hardware complexity increases significantly. Another important point to be noted from Fig. 3.4(a)-(d) is that the hardware complexity increases significantly with the increase of data-bus width for a specific frame-length.

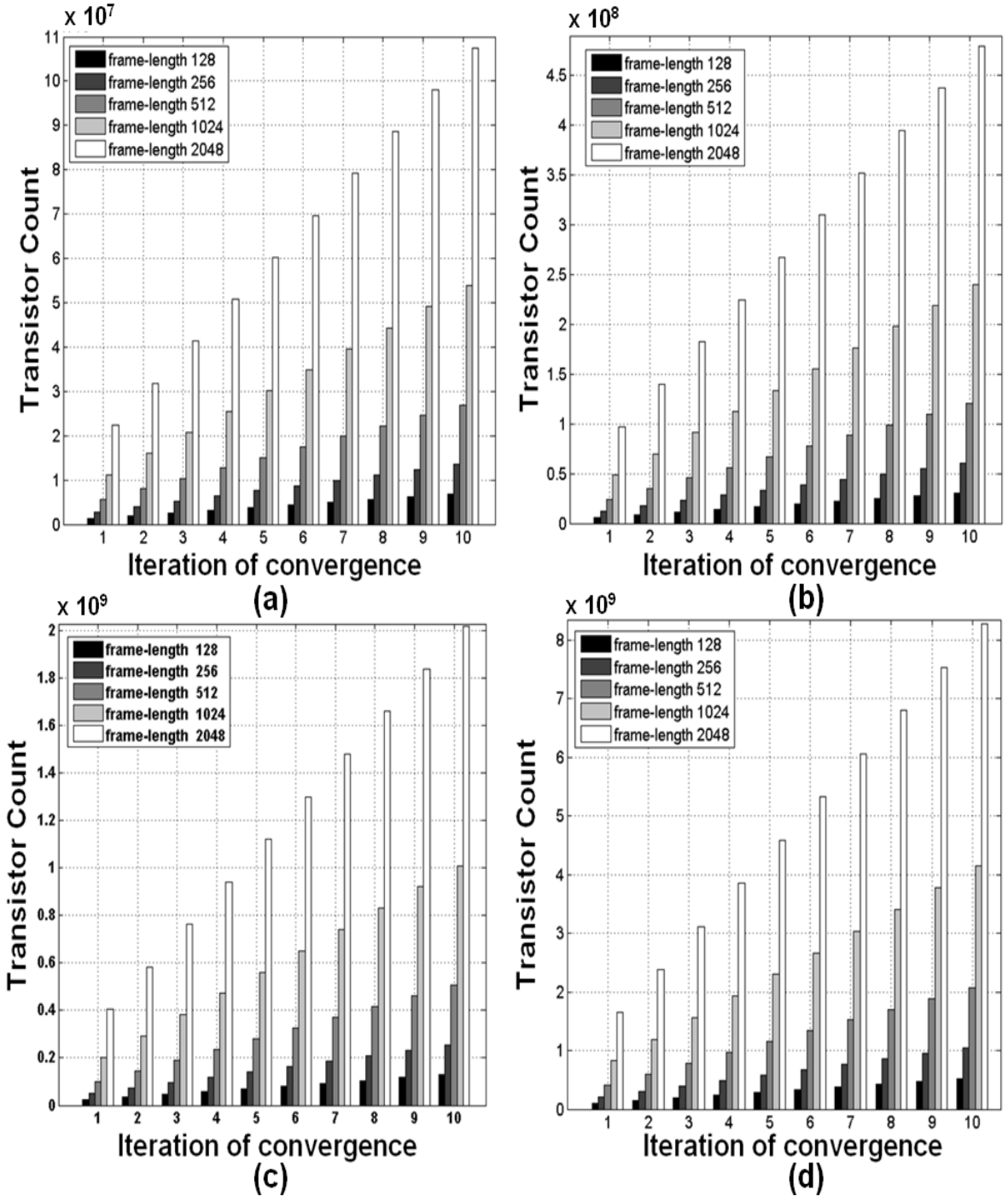


FIGURE 3.4: Transistor counts versus the iteration of convergence for (a) data-bus width = 4, (b) data-bus width = 8, (c) data-bus width = 16 and (d) data-bus width = 32.

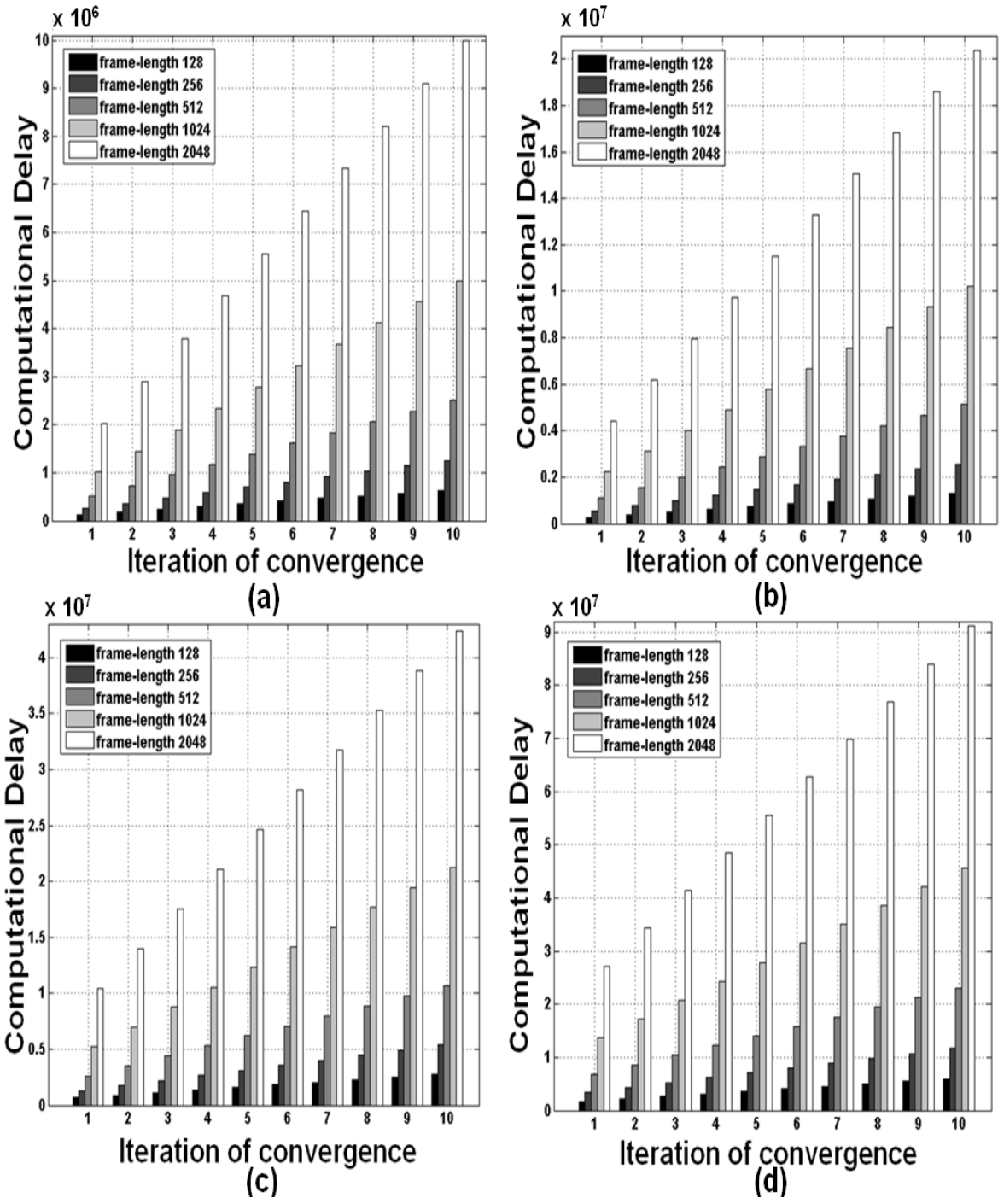


FIGURE 3.5: Computational Delay versus the iteration of convergence for (a) data-bus width = 4, (b) data-bus width = 8, (c) data-bus width = 16 and (d) data-bus width = 32.

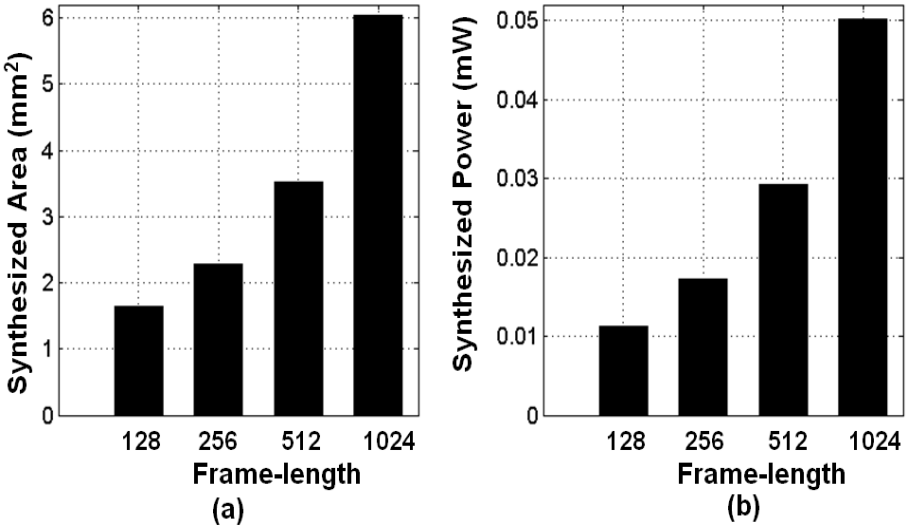


FIGURE 3.6: Synthesized results with different frame-lengths and data-bus width = 16. (a) Core Area in mm^2 and (b) Power in mW.

Similarly the delay of a standard two-input NAND gate (will be discussed in detail in section 3.10) has been considered as the fundamental unit for calculating the computational delay of the algorithm when mapped into the architecture. Like Fig. 3.4, the variation of computational delay with respect to different frame-length under four different data-bus width is shown in Fig. 3.5. It can be observed that this delay increases significantly with the increase of frame-length for a specific data-bus width. Comparing Fig. 3.5(a)-(d), it can again be noted that the delay increase even with the increase of data-bus width for a particular frame-length.

The optimized 2D FastICA architecture (to be proposed in section 3.8) has been designed using VHDL in RTL level and has been simulated and synthesized using available industry standard CAD tools (for detailed discussion please see section 3.10.4). Fig. 3.6 shows the variation of the post-synthesis area and power consumption for four different frame-length considering iteration of convergence as 5 and convergence threshold as $\theta = 1^o$ when the data-bus width is kept constant at 16. This figure clearly shows that with the increase in the frame-length, consumption of hardware resources in terms of area and power increases significantly.

Similarly same set of experiments can be run considering different iteration of convergence and for different values for convergence accuracy and the variation of hardware complexity and computational delay can be derived and post-synthesis results can be obtained.

3.7.2 Experimental Results

To the best of our knowledge there exists not a single reported study on algorithm-architecture holistic optimization approach for any of the existing ICA algorithms. So the main motivation behind this research is to present a generalized procedure which we believe is the first of its kind and can be adopted for any ICA algorithm and may lead to the development of design guidelines for the hardware designers. For that purpose, without any loss of generality, we have considered conventional $2D$ FastICA for our experiments and all results are obtained based on the generic signal modeling as discussed in section 3.6.

Since the whole signal domain has broadly been classified into two classes - stationary and non-stationary as mentioned in section 3.6, under the assumption of $2D$, these two generic classes can be mixed in the following three ways -

category 1: Two stationary signals (considered as the independent sources) are mixed together randomly to produce the mixed signals,
category 2: Two non-stationary signals (both are considered as the independent sources) are mixed together randomly to generate mixed signals and
category 3: One non-stationary and another stationary (both are considered as independent sources) are mixed together randomly to generate mixed signals.

The left-most column of Fig. 3.7 and 3.8 represents the variation of iteration of convergence for afore-mentioned *category-1* with difference accuracy of convergence range for different frame-length. Similarly the middle and the right-most column of Fig. 3.7 and 3.8 represent the variation of the iteration of convergence with different accuracy of convergence and frame-length for afore-mentioned *category-2* and *category-3* respectively. It is to be noted that five different *convergence accuracy* values have been chosen viz. 1^0, 0.8^0, 0.6^0, 0.4^0 and 0.2^0 and frame-lengths are considered as 128, 256, 512, 1024 and 2048 for experimental purpose. However same set of experiments can be run with different set of values and different results can be obtained, however the underlying principle as well as the trend of the results will remain the same.

As per the definitions of different categories mentioned above, the independent sources are mixed with random weights generated using Mersenne Twister RNG. The left, middle and right-most column of Fig. 3.7 and 3.8 show the variation of iteration of convergence with different degree of convergence accuracy for different frame-length for *category-1*, *category-2* and *category-3*. It can be observed that if convergence accuracy is increased i.e. if the convergence threshold θ (as shown in Fig. 3.1) is made smaller from 1^0 towards 0.2^0, FastICA needs more iterations for convergence for all three categories. The "black coloured boxes" and "grey coloured boxes" in the figures represent the mean and standard deviation of the iteration of convergence values obtained after running Monte Carlo simulation with $\geq 95\%$ confidence. However it can be observed from these

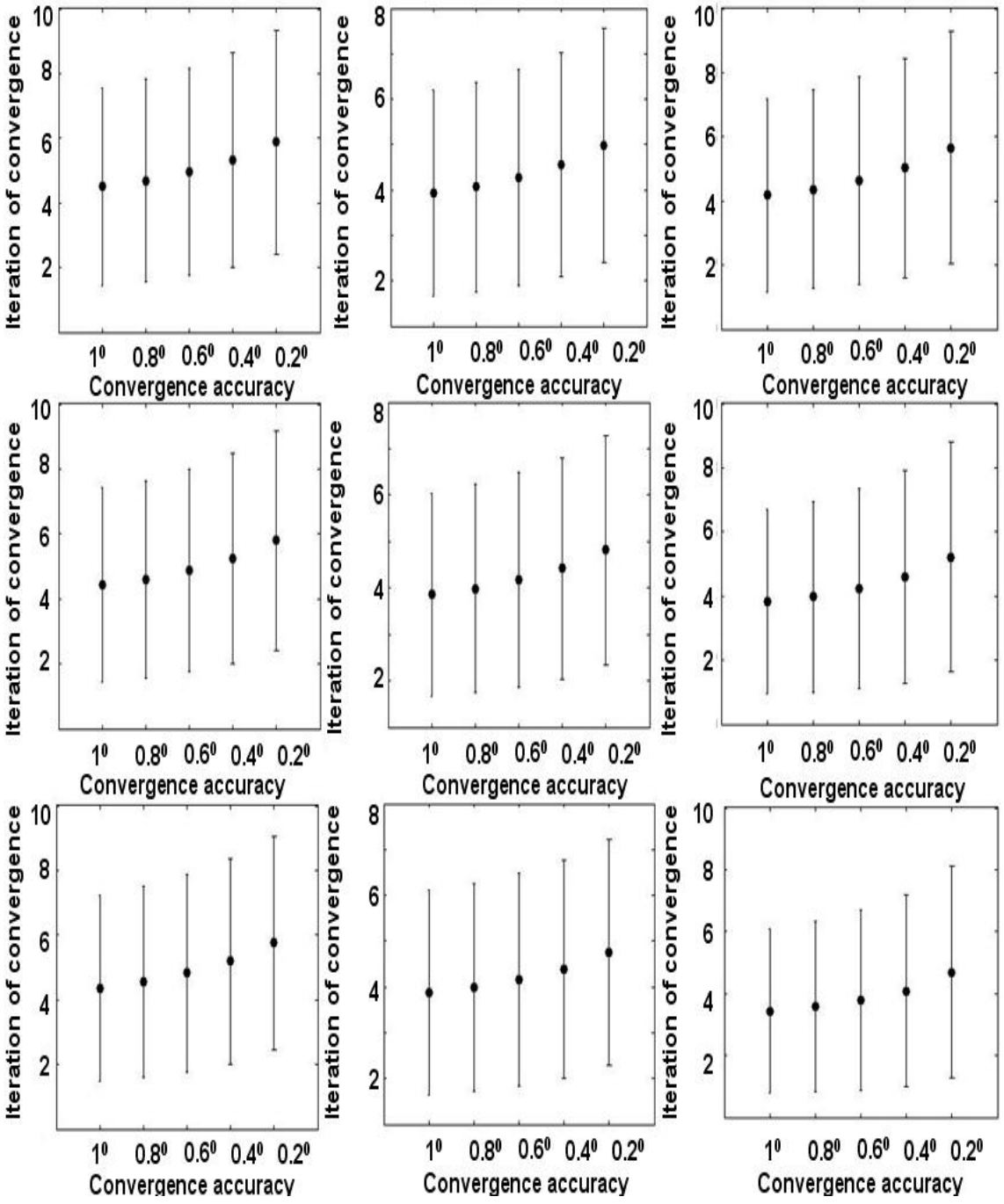


FIGURE 3.7: Iteration of Convergence with Mean and Standard Deviation under the above mentioned categories of the generic signal model with different levels of convergence accuracy. (a) Category - 1 and frame-length = 128, (b) Category - 2 and frame-length = 128, (c) Category - 3 and frame-length = 128, (d) Category - 1 and frame-length = 256, (e) Category - 2 and frame-length = 256, (f) Category - 3 and frame-length = 256, (g) Category - 1 and frame-length = 512, (h) Category - 2 and frame-length = 512 and (i) Category - 3 and frame-length = 512.

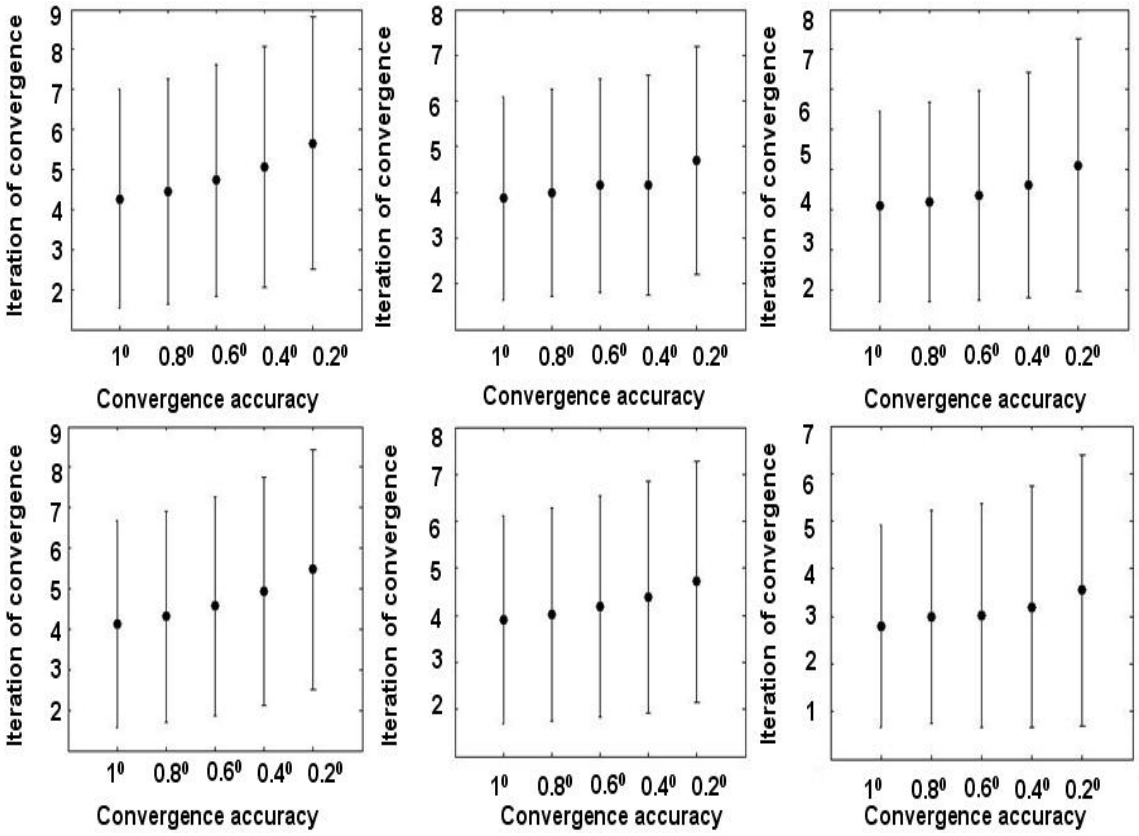


FIGURE 3.8: Iteration of Convergence with Mean and Standard Deviation under the above mentioned categories of the generic signal model with different levels of convergence accuracy. (a) Category - 1 and frame-length = 1024, (b) Category - 2 and frame-length = 1024, (c) Category - 3 and frame-length = 1024, (d) Category - 1 and frame-length = 2048, (e) Category - 2 and frame-length = 2048, (f) Category - 3 and frame-length = 2048.

figures that for a specific frame-length, iteration of convergence necessary to meet a particular convergence accuracy may be different for these three different categories.

3.8 Proposed Algebraic Methodology for Hardware Complexity Reduction of $2D$ FastICA

From operational point of view, as already mentioned in Section 3.3, FastICA consists of two main steps : Preprocessing and FastICA Iteration. The hardware reduction methodology to be outlined in this Section deals mainly with the FastICA Preprocessing unit which involves EVD. For 2D FastICA, considering C_{ij} (where $i = 0, 1$ and $j = 0, 1$) as the elements of C_X (as shown in (3.3)), its characteristic equation can be given as:

$$(C_{00} - \lambda) \times (C_{11} - \lambda) - C_{01}C_{10} = 0 \tag{3.11}$$

where $\lambda = d_1, d_2$ are the eigen values. Since C_X is a symmetric matrix, $C_{01} = C_{10}$. Using this in (3.11) and simplifying we get:

$$d_1, d_2 = \Big((C_{00} + C_{11}) \pm \sqrt{(C_{00} - C_{11})^2 + 4{C_{01}}^2}\Big)/2 \tag{3.12}$$

Proposition 3.1. Denoting $m_i = (d_i - C_{00})/C_{01}$, the eigenvector (e_i) corresponding to d_i will be given by :

$$e_i = [1 \quad m_i]^T, \text{where} \quad i = 1, 2 \tag{3.13}$$

Proof. The proof follows directly from the first proposition of [78]. ■

This technique of eigenvector estimation saves the computational resource for all 2×2 real matrices [78]. In terms of d_i, (3.13) can be written as:

$$e_i = [1 \quad (d_i - C_{00})/C_{01}]^T \tag{3.14}$$

Proposition 3.2. If $[d_i, e_i]$ is the *eigenpair* of C_X, $[d_i, \beta e_i]$ will also become *eigenpair* of C_X, where β is the real scalar quantity $\neq 0$.

Proof. Proof follows from the section 5.7.2 of [79]. ■

From (3.14) we observe that the denominator C_{01} of e_i remains fixed for a frame and thus can be treated as a scalar quantity. Therefore, using *Proposition* 3.2, (3.14) can be modified as:

$$e_i = [C_{01} \quad (d_i - C_{00})]^T \tag{3.15}$$

Using (3.15), E can be written as:

$$E = \begin{bmatrix} e_{11} & e_{21} \\ e_{12} & e_{22} \end{bmatrix} = \begin{bmatrix} C_{01} & C_{01} \\ d_1 - C_{00} & d_2 - C_{00} \end{bmatrix} \tag{3.16}$$

Comparing (3.14) and (3.16) it can be noted that the required division operation can be removed completely from the eigenvector computation. Using the normalized form of (3.16), the whitening matrix P can be represented as:

$$P = \begin{bmatrix} e_{11}/\sqrt{d_1(e_{11}^2 + e_{12}^2)} & e_{12}/\sqrt{d_1(e_{11}^2 + e_{12}^2)} \\ e_{21}/\sqrt{d_2(e_{21}^2 + e_{22}^2)} & e_{22}/\sqrt{d_2(e_{21}^2 + e_{22}^2)} \end{bmatrix} \tag{3.17}$$

Using (3.17) in (3.4) the whitened data matrix Z can be defined as:

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} (e_{11}X_1 + e_{12}X_2)/\sqrt{d_1(e_{11}^2 + e_{12}^2)} \\ (e_{21}X_1 + e_{22}X_2)/\sqrt{d_2(e_{21}^2 + e_{22}^2)} \end{bmatrix} \tag{3.18}$$

Since the denominators of Z_1 and Z_2 are constants for a particular frame, its inverse value can be computed only once at the beginning of each frame. This value can be used

repeatedly as a multiplication factor for the rest of that frame and thereby translating divisions into multiplications and resulting into hardware and operating speed efficiency since the hardware cost of a divider is much higher than that of a multiplier and at the same time incurs more delay than a multiplier.

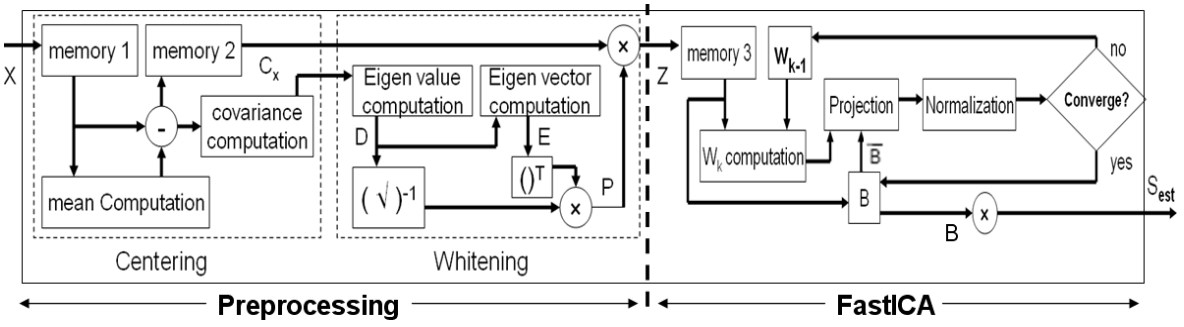


FIGURE 3.9: Overview of 2D FastICA Architecture

3.9 Architectural Overview of Proposed Methodology - $2D$ FastICA

The block diagram of the complete FastICA architecture is shown in Fig. 3.9. It consists of two main blocks - Preprocessing and FastICA. The preprocessing has two units - Centering and Whitening and their operation is to make the incoming data a zero mean variable and decorrelate these zero mean signals. The FastICA block estimates the final output vector.

It is clear from Fig. 3.9 that the direct implementation of the complete architecture requires complex arithmetic operations like division, square root evaluation, multiplication and normalization which makes it area and power inefficient. Thus we pay special attention to optimize different arithmetic units and at the same time use the architectural symmetry to reduce number of arithmetic operations wherever possible.

To show the effectiveness of the proposed hardware reduction methodology in 2D Kurtotic FastICA architecture here we present a fixed-point architectural design of it considering incoming data frame-length = 512 and wordlength = 16 bits as an example. The generic performance analysis is provided in next Section. Owing to the number of arithmetic operations and fixed-point nature of the implementation it is expected that the numerical error may be significantly high. To combat this effect we have introduced suitable Scaling Factor (SF) and variable databus width wherever necessary (as shown in the subsequent figures). The implementation of the Centering Unit is done by accumulating the incoming data samples and then shifting the accumulated sum to the right by nine bits (division by the frame-length = $512 = 2^9$). Fig. 3.10(a) shows the architecture of the eigenvalue computation unit following (3.12). The term $4{C_{01}}^2$ is implemented

using a squaring multiplier and shifting the result to left by two bits. The non-restoring square root algorithm [80] has been followed here to implement the square root circuit. Finally d_1 and d_2 are obtained by shifting the numerator of (3.12) right by one bit. The simplified divider-less architecture for eigenvector computation unit following (3.16) is shown in Fig. 3.10(b).

The block diagram of the architecture for computing whitened data matrix Z following (3.18) is shown in Fig. 3.11(a). To maintain the accuracy while applying the inversion operation (See the paragraph next to (3.18)), decimal 1 has been represented by 2^{15} and this inversion has been done following the approach presented in [81]. The constant multiplication factor (as mentioned above) is shown as the "inverse divisor" in Fig. 3.11(a). The variable wordlengths adopted at different parts of this circuit are also shown in Fig. 3.11(a).

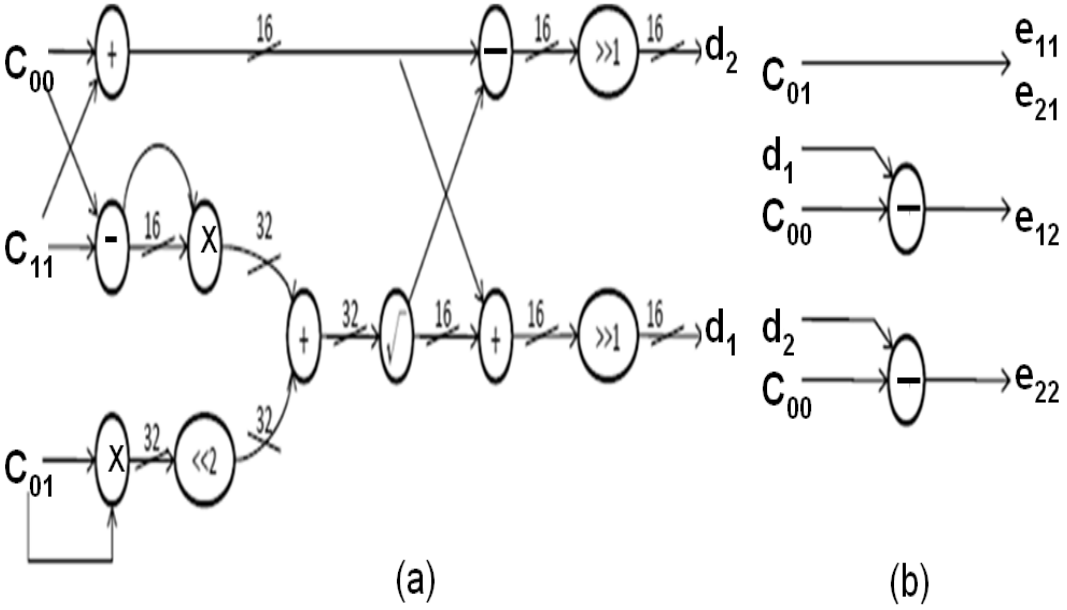


FIGURE 3.10: Proposed divider-less architecture for (a) eigenvalue and (b) eigenvector computation.

Exploiting the datapath symmetry the section shown in the dashed line segment can be further optimised as shown in Fig. 3.11(b) where three multipliers have been replaced by three simple multiplexers reducing the hardware cost further. Fig. 3.12(a) shows the block diagram of w_k computation unit within FastICA Iteration block. Computation of w_k based on Kurtosis contrast function involves computing 4^{th} power of the whitened data. Since the output wordlength of the whitening unit is 32 bits, we use 64-bit internal wordlength for the w_k computation unit. The effect of inclusion of the SF ($= 2^{16}$) before square-rooting (Fig. 3.11(a)), after each arithmetic operation inside the FastICA block is clearly shown in Fig. 3.12(a). However from practical word-length consideration, when the overall data scaling reaches 2^{24} (shown in the sub-block (a-2) of Fig. 3.12(a)), we downscaled it by 16-bit. The block diagrams of the projection and normalization

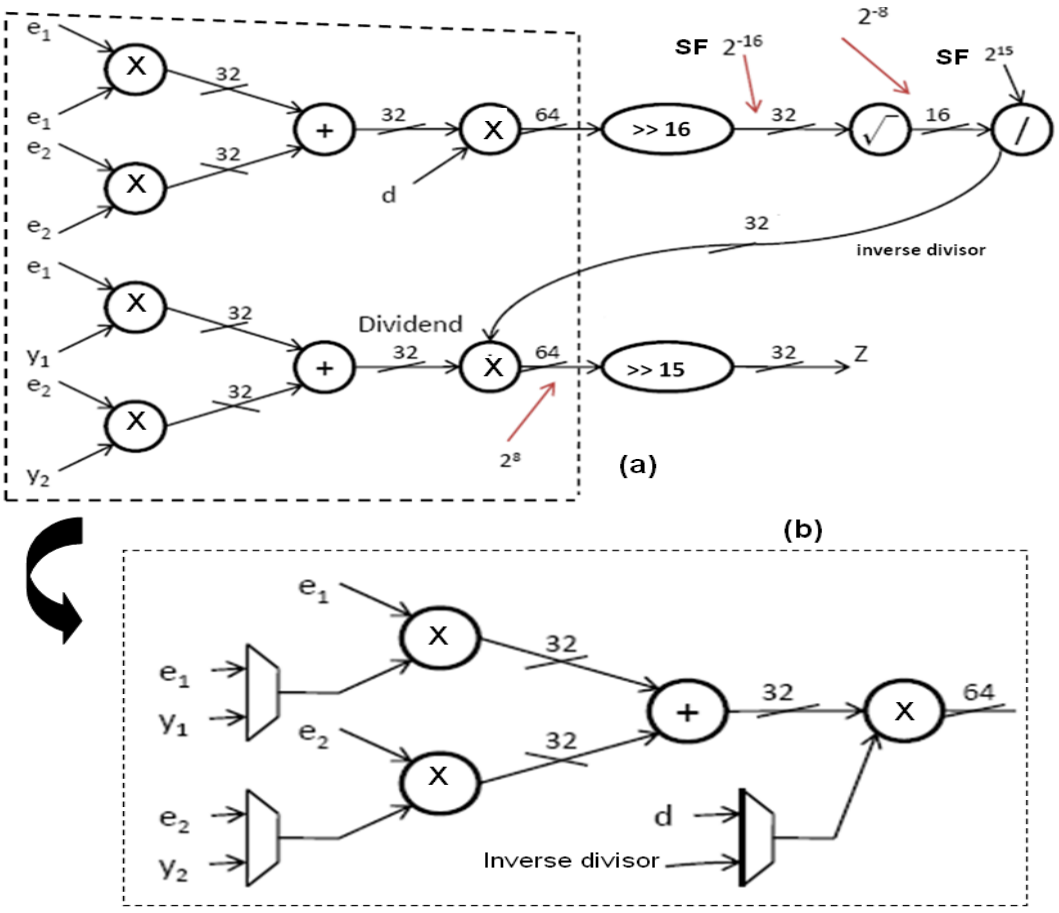


FIGURE 3.11: (a) Proposed architecture of the Whitening block replacing dividers by multipliers, (b) optimized architecture of the segment surrounded by dashed line in (a) exploiting datapath symmetry.

units are shown in Fig. 3.12(b) and Fig. 3.12(c) respectively following the projection equation given in [77]. The normalization operation of w_k in the normalization unit raises the same concern of loosing the information contained in the fractional part of the normalized data as discussed earlier in this section. To overcome this problem, as shown in Fig.3.12(c), an SF $= 2^{31}$ is introduced in the design.

3.10 Performance Analysis of the Proposed Methodology - $2D$ FastICA

In this Section we analyze the performance gain attainable by 2D Kurtotic FastICA using the hardware reduction methodology described in the previous Sections. In particular we concentrate on hardware saving and operational speed-up. Throughout the analysis we keep a generalized view of frame-length K, wordlength n and the maximum number of Iteration of Convergence (IOC) as M. However, since implementation of different arithmetic unit can be done in different ways, we present our results based

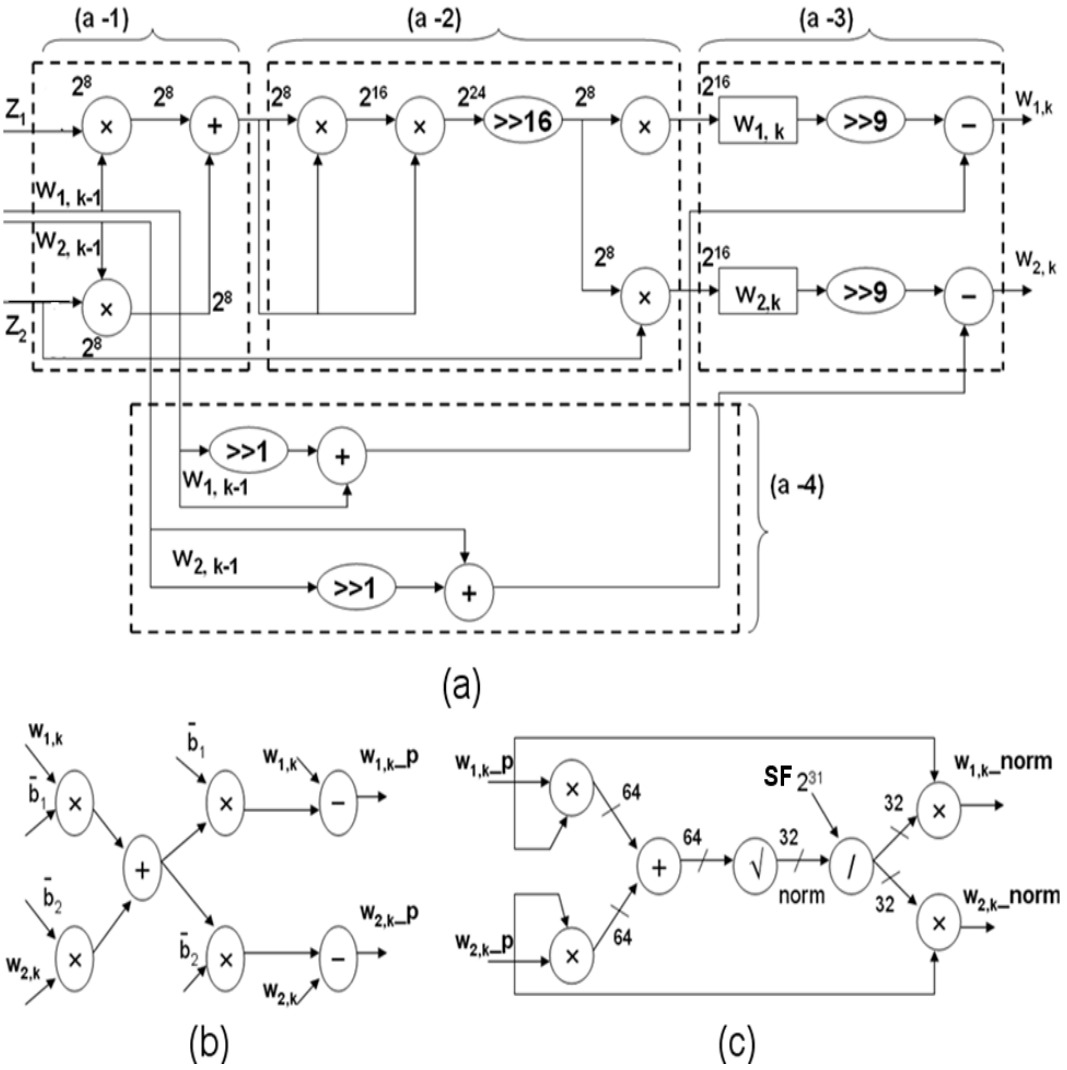


FIGURE 3.12: Fixed-point architecture of (a) w_k computation, (b) Projection and (c) Normalization unit. '$w_{i,k}$_P' and '$w_{i,k}$_norm' represent "projected" and "normalized" $w_{i,k}$ respectively.

on the number of arithmetic operation considering a flat unfolded architecture without resource sharing or parallel computing capability. Also to provide a comparison on a

TABLE 3.1: Comparison in terms of number of arithmetic operations.

Arch	Addition	Multiplication	Division
Direct[76]	$9K + 6KM$ $+2M$	$11K + 12KM$ $+19M + 8$	$7 + 2K$ $+6M$
Proposed	$9K + 6KM$ $+5M$	$13K + 11KM$ $+14M + 5$	$2 + 2M$
Gain	$-3M$	$-(2K - KM$ $-5M - 3)$	$+5 + 2K$ $+4M$

uniform platform we consider only Ripple Carry Adder (RCA), Conventional Array Multiplier (CAM) and Nonrestoring Array Divider (NAD) as the means of implementing

the arithmetic operations. Additionally it has been considered that the complexity of a multiplier used for squaring is half of the complexity of a full multiplier multiplying two different terms and the delay of a squaring multiplier is 80% to that of a full multiplier [82]. Also to the best our knowledge only one hardware implementation of 2D Kurtotic FastICA exists in literature (see [76]). Thus we compare the performance gain of the proposed method with this one only considering its possible fixed-point implementation.

3.10.1 Hardware Reduction

Table 3.1 shows the total number of arithmetic operations involved in the direct-mapping based architecture [76] and the proposed one. Since both of these schemes involve same number of subtraction and square rooting operations, they have been omitted from the Table 3.1. It can also be observed from Table 3.1 that the proposed methodology reduces $(5+2K+4M)$ number of division operations at the cost of $3M$ number of more addition and $(2K - KM - 5M - 3M)$ number of more multiplications. Considering a n-bit RCA requires n Full Adders (FA) (in a simplified view) [83], $n \times n$ CAM requires $n(n-2)$ FA plus n Half Adders (HA) and n^2 AND gates [84]. Similarly one $n \times n$ NAD consists of $0.5 \times n(3n-1)$ FA and $0.5 \times n(3n-1)$ XOR gates [85]. In addition, considering one FA cell requires 24 transistors, one HA cell and one two input XOR gates consists of 12 transistors and a two input AND gates consists of 6 transistors [62], we can calculate $TC_A = 24n$, $TC_M = 6n(5n-6)$ and $TC_D = 18n(3n-1)$, where TC_* are the transistor counts for RCA, CAM and NAD respectively. Putting the values of TC_A, TC_M and TC_D in Table 3.1, we get Transistor Savings due to reduction of Divider $TS_D = (5+2K+4M) \times TC_D$, Adder Penalty $(AP) = 3M \times TC_A$ and Multiplier Penalty $(MP) = (2K - KM - 5M - 3) \times TC_M$. Hence, the effective Transistor Saving (TS) can be given by:

$$TS = TS_D - (AP + MP) \tag{3.19}$$

Replacing the value of TS_D, AP and MP in (3.19) as derived above, we get:

$$\begin{aligned} TS = [&(360n + 48Kn + 366Mn + 30KMn \\ &+ 36K) - (198 + 324M + 36KM)]n \end{aligned} \tag{3.20}$$

Using a metric Transistor Saving Per Wordlength $(TSPW)$ [86], from (3.20), we can find:

$$\begin{aligned} TSPW = [&(360n + 48Kn + 366Mn + 30KMn \\ &+ 36K) - (198 + 324M + 36KM)] \end{aligned} \tag{3.21}$$

It can be observed from (3.21), $TSPW$ is a function of the wordlength (n), frame-length (K) and IOC (M). Fig. 3.13(a) shows the variation of $TSPW$ with n and K keeping $M = 5$ (as considered in the example architecture) where the range of K is considered

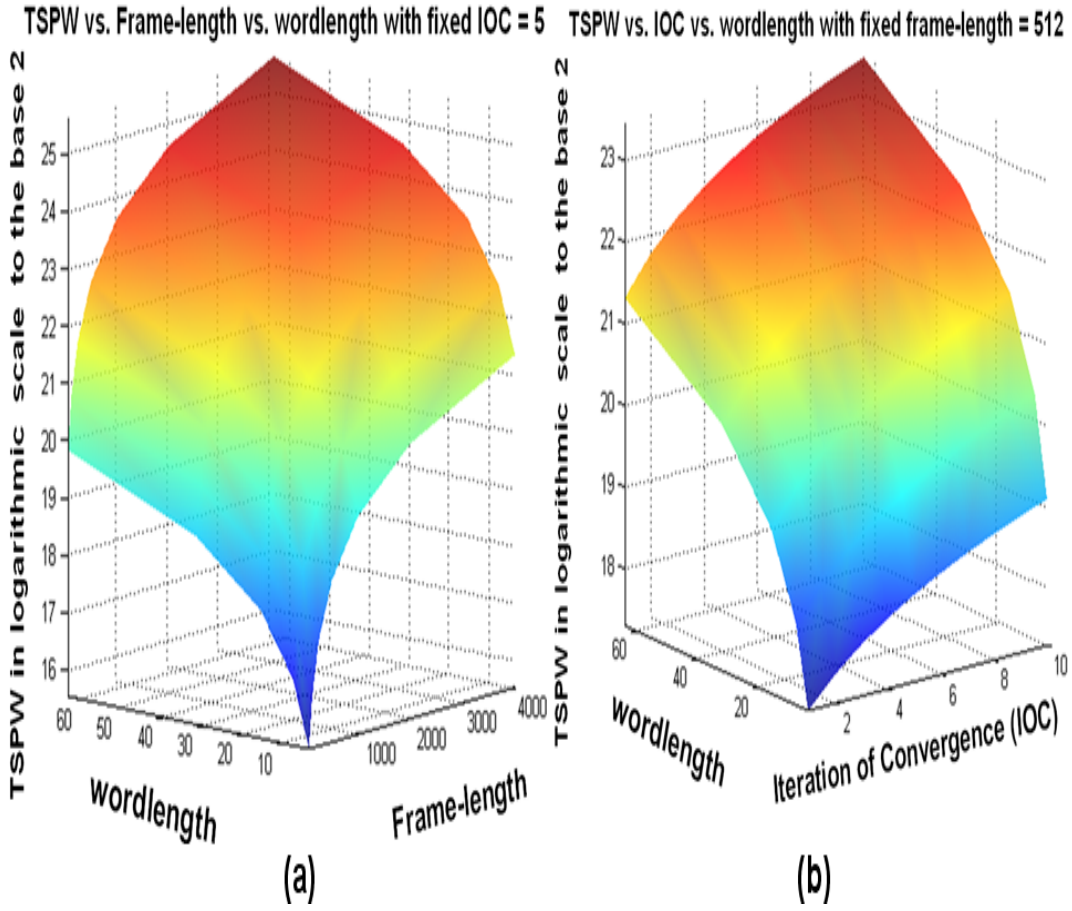


FIGURE 3.13: (a) Variation of $TSPW$ with Frame-length and wordlength for fixed IOC = 5 , (b) Variation of $TSPW$ with IOC and wordlength for fixed framelength = 512.

from 64 to 4096. It can be seen from Fig. 3.13(a) that as K varies the $TSPW$ increases exponentially (in dyadic space) resulting in significant hardware saving. Another point to be noted is that as K increases the AP and MP will become significant causing the gradual decrement of the rate of exponential growth of $TSPW$. In Fig. 3.13(b) $TSPW$ is plotted for fixed $K = 512$ with range of M from 1 to 10. As shown in Fig. 3.13(b) the variation of $TSPW$ with M increases linearly. However, since in this case the AP and MP will also increase linearly, the slope of rate of growth of $TSPW$ decreases slowly.

3.10.2 Delay Analysis

To compute the delay of the proposed methodology based architecture we follow [83] which uses $\triangle$ as the delay of a 2-input NAND gate. According to this convention the delay for n-bit RCA, CAM and NAD respectively can be given as $DL_A = 2n\triangle$, $DL_M = 8n\triangle$ [83] and $DL_D = (3n+2)n\triangle$ [85]. The total number of multipliers shown in Table 3.1 includes $(13K+10KM+12M+2)$ full multipliers plus $(2KM+4M+6)$ squaring

multipliers. As mentioned earlier in this Section, the delay of a squaring multiplier is 80% of a full multiplier delay, therefore the penalty due to squaring multiplier can be translated in terms of full multiplier. Using these values in Table 3.1, the effective Delay

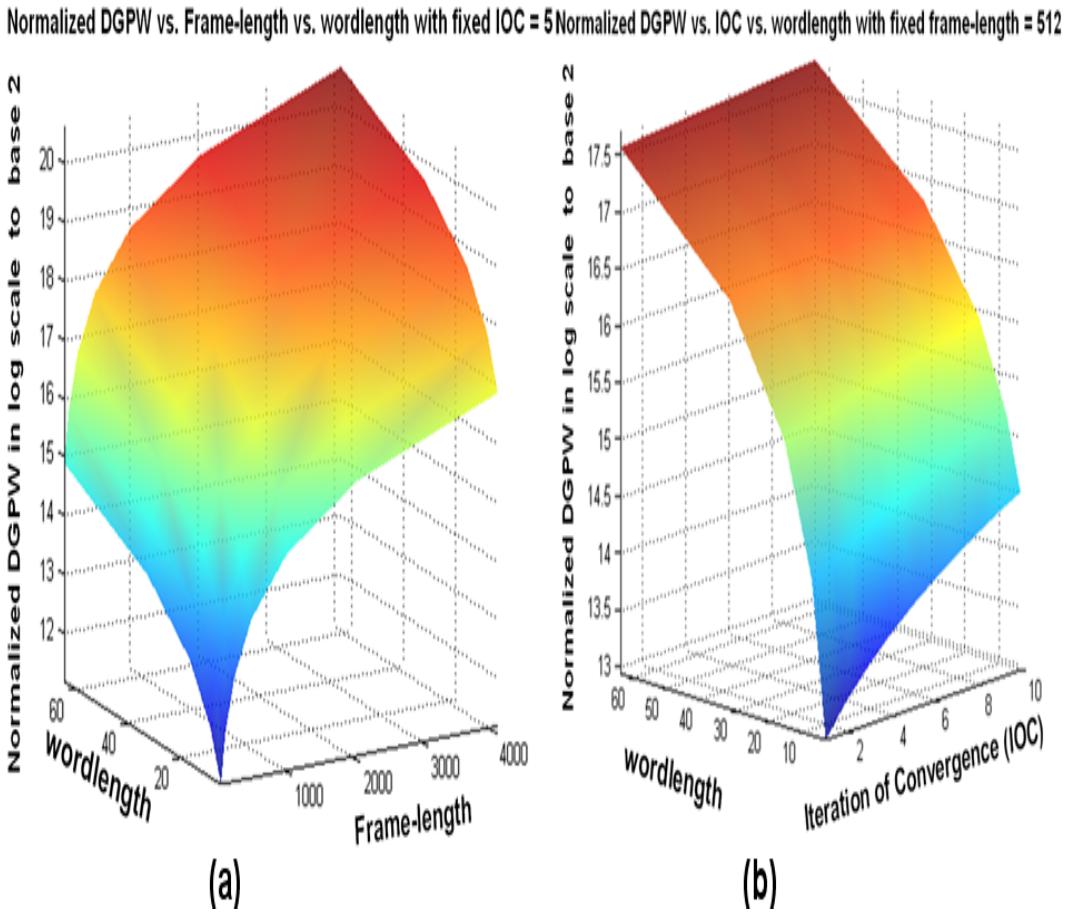


FIGURE 3.14: (a) Variation of Normalized $DGPW$ with Frame-length and wordlength for fixed IOC = 5 , (b) Variation of normalized $DGPW$ with IOC and wordlength for fixed framelength = 512.

Gain (DG) can be given by:

$$\begin{aligned} DG = (5 + 2K + 4M) \times DL_D - 3M \times DL_A \\ - 0.2(10K - 2KM - 19M - 6) \times DL_M \end{aligned} \tag{3.22}$$

Using the values of DL_A, DL_M and DL_D in (3.22) as mentioned above, DG can further be written as:

$$\begin{aligned} DG = [(15n + 6nK + 12Mn + 19.6 + 32.4M \\ + 3.2KM) - (12K)]n\triangle \end{aligned} \tag{3.23}$$

More specifically, from (3.23), we define a new metric called Delay Gain Per Wordlength ($DGPW$) as:

$$\begin{aligned} DGPW = [(15n + 6nK + 12Mn + 19.6 \\ + 32.4M + 3.2KM) - (12K)]\triangle \end{aligned} \tag{3.24}$$

Fig. 3.14(a) shows the variation of normalized $DGPW$ with n and K for $M = 5$. Like Fig 3.13(a), Fig 3.14(a) also shows the exponential increase in normalized $DGPW$ at decremental rate with the increase of K. Fig. 3.14(b) shows the variation of normalized DGPW with n and M for $K = 512$ showing the smooth linear growth of normalized $DGPW$.

3.10.3 Functional Validation and Error Analysis

To do the functional validation of the proposed architecture, we generated a C-model of the FastICA algorithm which is compared with the corresponding Verilog model of the proposed methodology based architecture. As test vectors, we have chosen the data samples as given in [87] equivalent to one frame-length. The estimated outputs from the C model and Verilog model are shown in the left and right side of the Fig. 3.15 respectively. As can be seen from Fig. 3.15, there is close correlation between the waveforms confirming correct functionality of the proposed fixed-point FastICA architecture. However, because of the fixed-point implementation, the actual hardware is prone to numerical errors due to the finite wordlength and quantization effects, which gets accumulated at every arithmetic operation. To examine the overall error effect on the estimated output, we have plotted the probability of error vs bit position in Fig. 3.16 following the approach discussed in [86]. This is derived by comparing the VHDL implementation with the C-model based implementation of the outputs (as shown in Fig. 3.15) and by converting the absolute value of the error (E) to binary using $E_2 = |\ln(E)/\ln(2)|$, where E_2 is the bit position (from the MSB) at which the error occurs (in magnitude). It can be seen from Fig. 3.16 that mostly the error occurs 8^{th} bit-position onwards which is acceptable from practical implementation point of view. However in the cases where more accuracy is required the value of the SF and wordlength needs to be varied as per requirement. The Mean Squared Error (MSE) is also computed by comparing the original source signals (as shown in Fig. 3.15) with the VHDL generated output and it has been found that MSEs for the Set-1 and Set-2 of Fig. 3.15(a) are 4.63×10^{-4} and 5.3159×10^{-4} respectively and for the Set-1 and Set-2 of Fig. 3.15(b) are 4.8681×10^{-4} and 1.2×10^{-3} respectively.

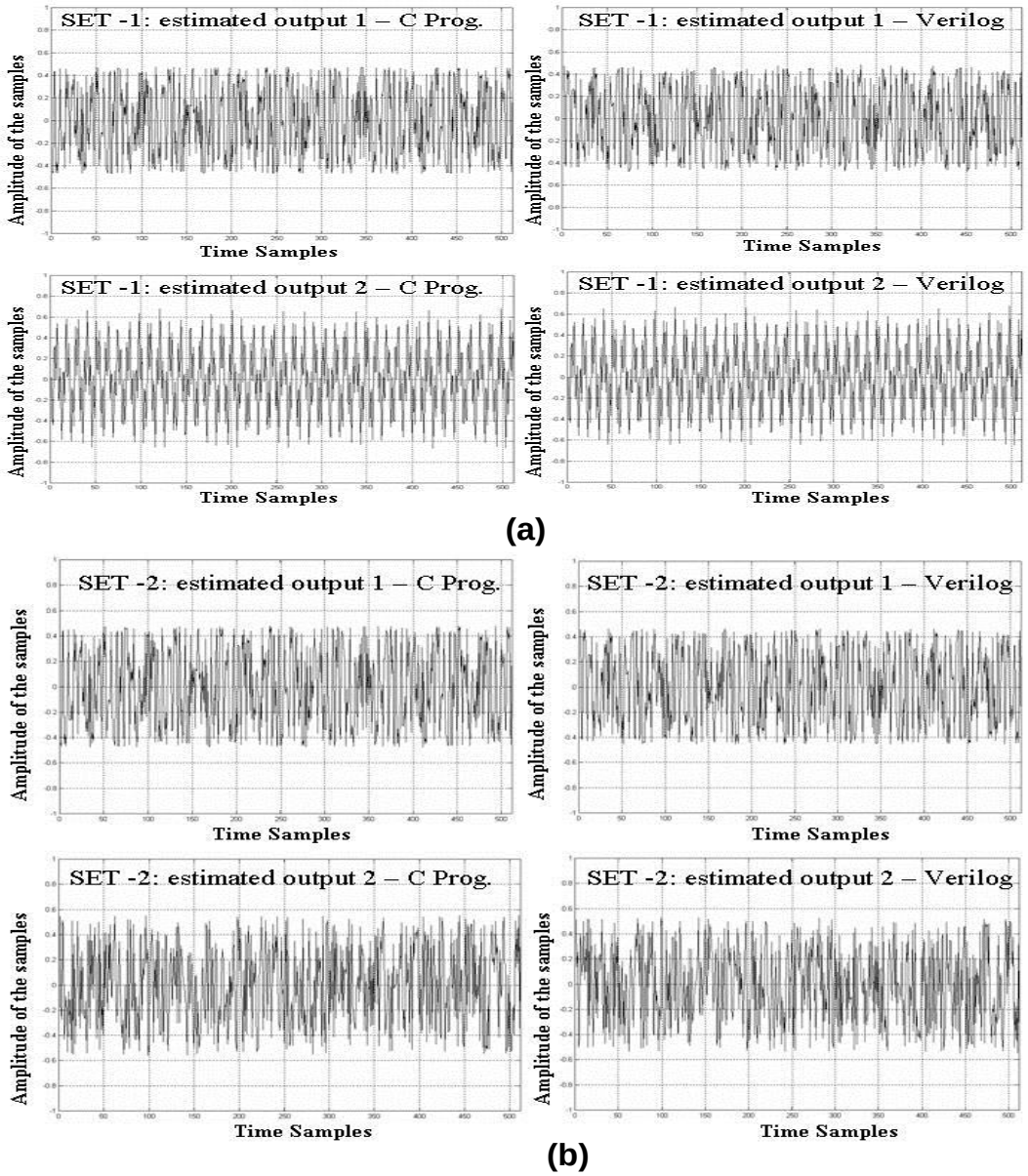


FIGURE 3.15: Left side - Estimated waveforms generated from the C model, right side - estimated waveforms generated from the Verilog model of the proposed FastICA architecture. (a) SET 1 results, the source signals were $0.5\sin[500t + 5\cos(60t)]$ and $0.7\sin(450t)\sin(40t)$, (b) SET 2 results, the source signals were $0.5\sin[500t+5\cos(60t)]$ and $n(t)$- uniformly distributed random noise within the range $[-1, 1]$.

3.10.4 Implementation Results and Comparisons

To give an insight into the area and power cost of the implemented architecture, it is coded using Verilog and synthesized using Synopsys Design Compiler in 0.13 μm standard cell CMOS technology. The synthesized area and power consumption are 3.55 mm^2 and 27.1 μW @ 1 MHz frequency for V_{DD} = 1.2 V. The power value is

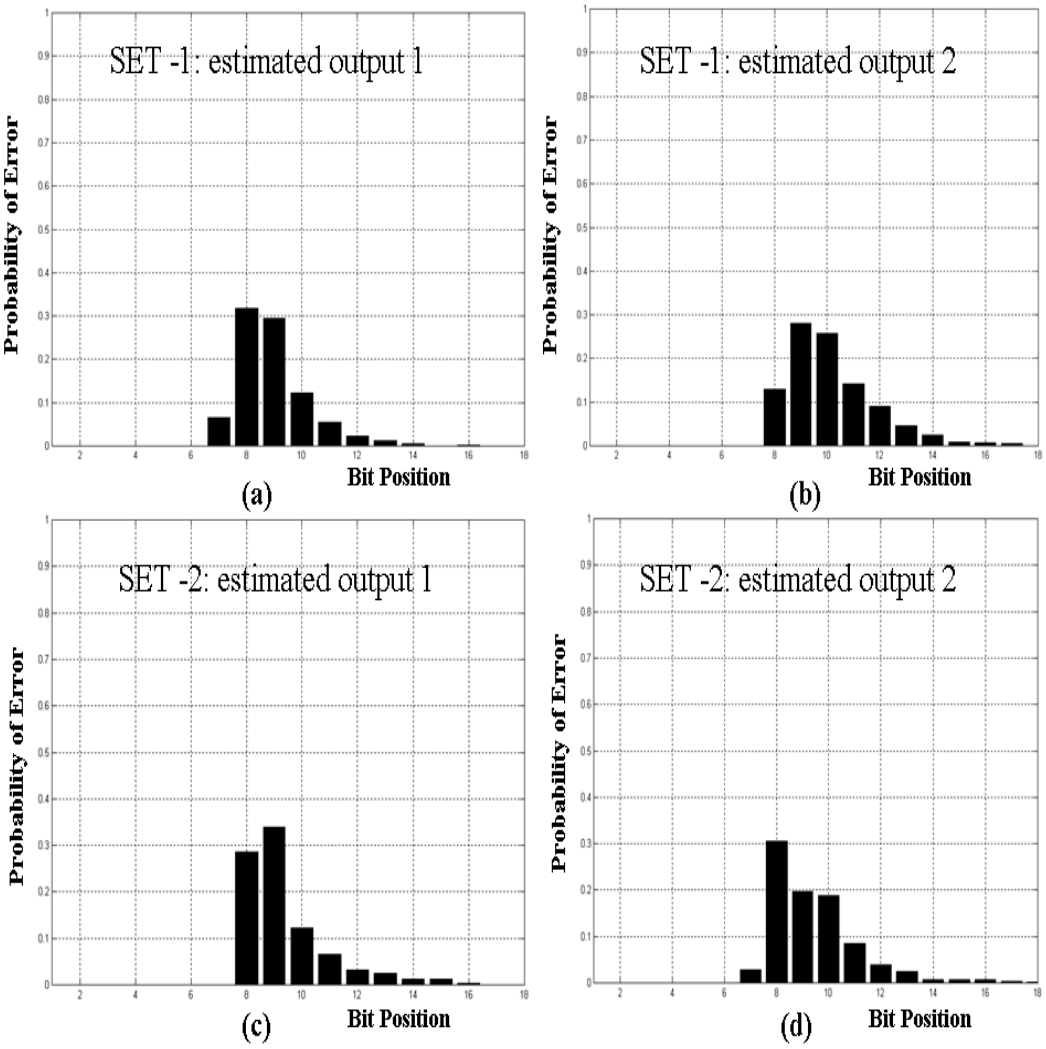


FIGURE 3.16: Probability of error vs. bit position in the proposed architecture. (a) Set-1, estimated source-1, (b) Set-1, estimated source-2, (c) Set-2, estimated source-1, (d) Set-2, estimated source-2.

obtained by feeding continuously 16-bit random vectors equal to one frame-length into the synthesized netlist and applying Synopsys Prime Time. Since, to the best our knowledge, there is no such published result on fixed-point implementation of FastICA we are unable to compare these parameters with any other work. However the results show that using the adopted methodology a 2D Kurtotic-FastICA will require very small silicon area and consume very low power.

3.11 Concluding Remarks

In this chapter firstly the impact of several FastICA algorithmic parameters on the corresponding architectural implementation has been investigated and a generalized experimental procedure was presented which can be used as a designer's guideline prior to hardware implementation. Following this procedure, given the knowledge of the available resource and accuracy wanted, a hardware designer can determine the required frame-length and iteration of convergence for optimum hardware design. Since this proposed procedure is generic, it can be extended to any ICA algorithm for its architectural implementation.

Secondly, based on such algorithm-architecture holistic optimization approach, a hardware reduction methodology for 2-D Kurtotic FastICA was proposed. This proposed algebraic method resulted in hardware saving by eliminating division operations and exploiting datapath symmetry as shown in Table 3.1. As shown in Section 3.10, this methodology increased the overall speed of operation and also exhibited low power consumption and thereby making it a good candidate for resource constrained applications.

Chapter 4

n-Dimensional Cross Product Computation

In Section 3.2 of the last chapter the necessity of algorithm-architecture holistic optimization approach for FastICA has been introduced in the context of resource constrained applications. It has been shown how such approach has led to the development of a low complexity basic 2D FastICA architecture. Here FastICA is explored further in nD space. For that purpose, in this chapter concept of a novel recursive computation of nD cross product algorithm and architecture based on the fundamental 3D cross product computation core is proposed. This method will lead to the development of low complexity nD FastICA which will be discussed in detail in the next chapter.

Due to its numerous applications in signal processing, VLSI architectures of vector dot product (or simply dot product, inner product or scalar product) has been exhaustively studied over the decades [115], [116]. Unfortunately, no such effort has been made for VLSI implementation of Vector cross product [1]. However the concept of $3D$ vector cross-product and its extension in higher dimensions have been well researched in mathematics since nineteen forties and the corresponding topological as well as algebraic proofs have already established the fact that generalization of r-fold vector cross product in nD space exists precisely in the following cases [125] - [129]:

(1) $r = 1$ and n is even.

(2) $r = n - 1$ and n is arbitrary.

(3) $r = 2$ and $n = 3$ (i.e. common cross product) and 7.

(4) $r = 3$ and $n = 8$.

[1] The contents of this chapter have partly appeared as "Algorithm and Architecture for N-D Cross product Computation" by Acharyya *et. al.* in IEEE Trans. Signal Processing. Please see Appendix-A for further detail.

But despite the presence of the concept of nD cross product in mathematics for long time, due to lack of known engineering applications in general purpose computer arithmetic and signal processing, it appears no effort has been made so far for its hardware realization. However, the emerging field of signal processing like BSS using ICA [117], [118] may get substantial benefit from the concept of vector cross product.

Since ICA involves intensive numerical computations [99], from the architectural point of view, it is very unlikely that such method will be able to satisfy the strict budget of power and area while maintaining excellent precision simultaneously within resource constrained applications such as next generation mobile healthcare system unless the number of associated arithmetic operations is reduced significantly. This is where the concept of vector cross product can be used while staying within the general framework of ICA owing to the fact that cross product of a set of orthogonal vectors results in a vector orthogonal to each of them - the exact requirement of ICA. To the best of our knowledge, cross product based low complexity ICA has not been studied so far and a detailed discussion on the benefit of using cross product in ICA will be discussed in detail in the next chapter.

In this chapter, considering first the case-2 ($r = n - 1$ and n is arbitrary) mentioned above we will propose a novel algorithm which uses $3D$ cross product as the basic operation and recursively computes nD cross product in terms of kD cross product, where $k \in \{3, 4, ..., (n-1)\}$, exploiting their inherent similarity of mathematical structures. Subsequently we propose a possible architecture for nD cross product. Secondly, we will show that further hardware saving is possible at the architecture level if instead of $3D$; the $4D$ cross product is used as the basic operational block exploiting its inherent symmetry (the symmetry-based approach).

Section 4.1 gives the preliminaries of the cross product and discusses certain terminologies from the basic linear algebra which will be used throughout the rest of this chapter. Section 4.2 describes the recursive nD cross product algorithm using $3D$ cross product as the basic operation (designated as the generalized algorithm from this point onward), Section 4.3 discusses the architecture of the generalized nD cross product and illustrates the symmetry based hardware reduction methodology. Mathematical models for the hardware complexity, delay and bit precision error for both of the generalized and symmetry-based approaches are described and compared in Section 4.4. Section 4.5 provides the post-synthesis results of these architectures in terms of power dissipation, area consumption and gate complexities and conclusions are drawn in Section 4.6.

4.1 Preliminaries of Cross Product

Let us consider a sub-space $\mathbf{H} \in \mathbb{R}^3$ spanned by two mutually orthogonal vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ where:

$$\mathbf{v_1} = \mathbf{e_1}v_{1,1} + \mathbf{e_2}v_{1,2} + \mathbf{e_3}v_{1,3} \tag{4.1}$$

$$\mathbf{v_2} = \mathbf{e_1}v_{2,1} + \mathbf{e_2}v_{2,2} + \mathbf{e_3}v_{3,2} \tag{4.2}$$

$\mathbf{e_1}$, $\mathbf{e_2}$ and $\mathbf{e_3}$ are the unit vectors in $\mathbb{R}^3$ and $[v_{1,1}\ v_{1,2}\ v_{1,3}]$ and $[v_{2,1}\ v_{2,2}\ v_{2,3}]$ are the components of $\mathbf{v_1}$ and $\mathbf{v_2}$. The cross-product of $\mathbf{v_1}$ and $\mathbf{v_2}$ is defined as:

$$\begin{aligned}\mathbf{v_3} &= [v_{3,1}\ v_{3,2}\ v_{3,3}] \\ &= \begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \\ v_{1,1} & v_{1,2} & v_{1,3} \\ v_{2,1} & v_{2,2} & v_{2,3} \end{vmatrix}\end{aligned} \tag{4.3}$$

where $|\cdot|$ represents **determinant** operation. By definition of cross product the vector $\mathbf{v_3}$ is orthogonal to $\mathbf{H}$ with the possibility of sign inversion. Proceeding the same way, nD cross product can be represented as [130] - [134]:

$$\mathbf{v_n} = \begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} & \dots & \mathbf{e_n} \\ v_{1,1} & v_{1,2} & v_{1,3} & \dots & v_{1,n} \\ v_{2,1} & v_{2,2} & v_{2,3} & \dots & v_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ v_{n-1,1} & v_{n-1,2} & v_{n-1,3} & \dots & v_{n-1,n} \end{vmatrix} \tag{4.4}$$

Now, using the **expansion by cofactors** in the first row of $\mathbf{v_n}$, (4.4) can be expressed as [120]:

$$\begin{aligned}\mathbf{v_n} &= \mathbf{e_1}V_{11} + \mathbf{e_2}V_{12} + \mathbf{e_3}V_{13} + \cdots + \mathbf{e_n}V_{1n} \\ &= \sum_{k=1}^{n} e_k V_{1k}\end{aligned} \tag{4.5}$$

where, V_{1k} is the $(1k)^{th}$ **cofactor** of $n \times n$ matrix inside the determinant of $\mathbf{v_n}$ and can be defined as [121]:

$$V_{1k} = (-1)^{1+k}|M_{1k}| \tag{4.6}$$

where M_{1k}, the $(n-1)\times(n-1)$ matrix is the $(1k)^{th}$ **Minor** which is obtained by deleting the first row and k^{th} column of the fore-mentioned $n \times n$ matrix whose determinant is

$\mathbf{v_n}$. Now, substituting value of V_{1k} from (4.6) in (4.5), we get:

$$\mathbf{v_n} = \sum_{k=1}^{n} \mathbf{e_k}(-1)^{1+k}|M_{1k}| \tag{4.7}$$

Based on these preliminary discussions, we approach towards developing the novel generalized algorithm for nD cross-product computation in the next section.

4.2 Proposed Generalized Algorithm for nD Cross Product

The fundamental principle of formulating a generalized algorithm for nD cross product computation hinges on the fact that there exists a structural similarity between nD and $(n-1)D$ cross product when expanded in their cofactor form. This similarity can be explained using the following example of computing $4D$ cross product. Putting $n = 4$ in (4.4) we get:

$$\begin{aligned} \mathbf{v_4} &= [v_{4,1}\ v_{4,2}\ v_{4,3}\ v_{4,4}] \\ &= \begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} & \mathbf{e_4} \\ v_{1,1} & v_{1,2} & v_{1,3} & v_{1,4} \\ v_{2,1} & v_{2,2} & v_{2,3} & v_{2,4} \\ v_{3,1} & v_{3,2} & v_{3,3} & v_{3,4} \end{vmatrix} \end{aligned} \tag{4.8}$$

Expanding (4.8) following cofactor expansion method using (4.7), we get:

$$\begin{aligned} \mathbf{v_4} \quad = \quad & \mathbf{e_1}\begin{vmatrix} v_{1,2} & v_{1,3} & v_{1,4} \\ v_{2,2} & v_{2,3} & v_{2,4} \\ v_{3,2} & v_{3,3} & v_{3,4} \end{vmatrix} - \mathbf{e_2}\begin{vmatrix} v_{1,1} & v_{1,3} & v_{1,4} \\ v_{2,1} & v_{2,3} & v_{2,4} \\ v_{3,1} & v_{3,3} & v_{3,4} \end{vmatrix} \\ & + \mathrm{e}_3\begin{vmatrix} v_{1,1} & v_{1,2} & v_{1,4} \\ v_{2,1} & v_{2,2} & v_{2,4} \\ v_{3,1} & v_{3,2} & v_{3,4} \end{vmatrix} - \mathbf{e_4}\begin{vmatrix} v_{1,1} & v_{1,2} & v_{1,3} \\ v_{2,1} & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{vmatrix} \end{aligned} \tag{4.9}$$

Comparing (4.9) with (4.3) it can be easily seen that each of the four components of $\mathbf{v_4}$ (e.g. $v_{4,1}$, $v_{4,2}$, $v_{4,3}$, $v_{4,4}$ as shown in (4.8)) is *structurally* similar to one complete $\mathbf{v_3}$ (as in (4.3)). Therefore, using $3D$ cross product operation *four times* one can compute a $4D$ cross product. However, the following facts can be observed while comparing (4.9) and (4.3):

Observation 1. (i) the unit vectors of $\mathbf{v_3}$ in $3D$ cross product in (4.3) should be mapped

accordingly into the components of the first vector $\mathbf{v_1}$ for $4D$ cross-product computation in (4.8); for example $\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{e_3}$ from (4.3) have to be mapped into $v_{1,2}$, $v_{1,3}$, $v_{1,4}$ respectively in (4.9) for $v_{4,1}$ computation and so on.

(ii) the rest of the components of $\mathbf{v_1}$ and $\mathbf{v_2}$ in (4.3) should be properly mapped into the components of $\mathbf{v_2}$, $\mathbf{v_3}$ in each cofactor for $4D$ cross-product in (4.8); for example, $v_{1,1}$, $v_{1,2}$ and $v_{1,3}$ in (4.3) have to be mapped into $v_{2,2}$, $v_{2,3}$, $v_{2,4}$ respectively in (4.9) for $v_{4,1}$ computation and so on.

(iii) the sign of each component in $\mathbf{v_4}$ in (4.8) should properly be assigned while using four $3D$ cross product computation units to realize one $4D$ cross-product.

The same similarity holds between $5D$ and $4D$ crossproduct as well.

Lemma 1. An nD cross-product operation can be realized by a sequence of 3-D cross product operation.

Proof. By putting $n > 3$ in (4.4) and expanding the resultant matrix using (7) for any arbitrary value of n it can be observed that, $\mathbf{v_n}$ (the cross-product in $\mathbb{R}^n$) can be realized using the n times $\mathbf{v_{n-1}}$ ($\in \mathbb{R}^{n-1}$) cross-product operation. Similarly, $\mathbf{v_{n-1}}$ ($\in \mathbb{R}^{n-1}$) cross product can be realized by using $\mathbf{v_{n-2}}$ ($\in \mathbb{R}^{n-2}$) cross-product operation $(n-1)$ times and so on until $n = 3$ where this structural symmetry seized to exist. Therefore it can be concluded intuitively that any nD cross product where $n > 3$ can be realised by repetitive use of $3D$ cross product operation. ■

Thus the $3D$ cross product operation can be considered as the core operation for computing multi-dimensional cross product. Unless otherwise mentioned, from now on we will use the terminology $3D$ core to denote the structure of $3D$ cross product matrix as shown in (4.3). However, when representing nD cross product by n number of $(n-1)D$ cross products three mapping schemes need to be employed (generalization of *Observation* 1): (i) **coefficient mapping** - where the unit vectors are termed as coefficients, (ii) **minor mapping** and (iii) **sign mapping**. Fig. 4.1 shows the overall flow of the generalized cross product computation algorithm to be proposed in terms of these mapping schemes.

4.2.1 Coefficient Mapping

Lemma 2. nD cross-product will require $d = (n-3)$ decomposition levels to get decomposed into structure of $3D$ cores where, decomposition level (d) denotes the number of levels of cofactor based expansion necessary to decompose an nD cross-product matrix into several $3D$ core structures (as shown in (4.3)).

Proof. It can be proved by the method of induction. To compute $5D$ cross product (using $n = 5$ in (4.4)), first level of decomposition ($d = 1$) yields five $4D$ structures (as in (4.8). Subsequently, in the second level of decomposition ($d = 2$), each of the

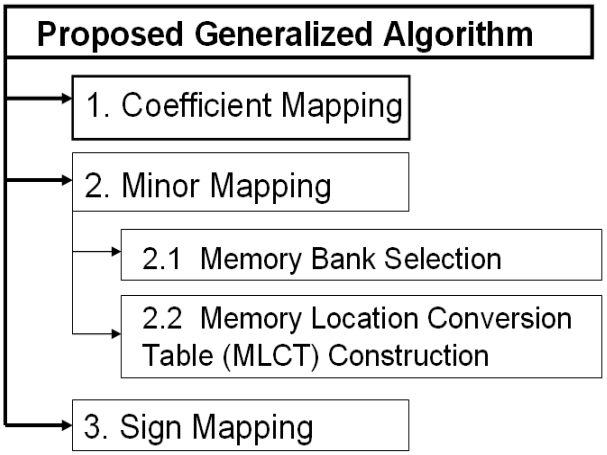


FIGURE 4.1: Flow of the proposed generalized algorithm for nD cross product computation comprising of three mapping schemes.

$4D$ structures can be decomposed into four $3D$ cores (as in (4.3)). Therefore, $4D$ and $5D$ cross-product computation need $(4-3)=1$ and $(5-3)=2$ decomposition levels respectively to get decomposed into structure of $3D$ cores. Proceeding the same way, the nD cross product requires $d = (n-3)$. ■

Theorem 1. **Coefficient Mapping -** Considering i and j be the number of unit vectors in $\mathbb{R}^n$ and $\mathbb{R}^{n-1}$ respectively (i.e. $1 \leq i \leq n$, $1 \leq j \leq (n-1)$), the coefficients (i.e. the unit vectors in $(n-1)D$ cross-product (${}^{(n-1)D}\mathbf{e_j}$)) can structurally be mapped into the elements of first row of each Minor of every component of nD cross-product using the following equation:

$$ {}^{(n-1)D}\mathbf{e_j} \mapsto \begin{cases} {}^{(nD)}v_{d,j+(d-1)} & \text{if} \quad j < i \\ {}^{(nD)}v_{d,j+d} & \text{otherwise} \end{cases} \tag{4.10} $$

where, number of decomposition level (d) is delimited by:

$$ 1 \leq d \leq (n-3) \tag{4.11} $$

where, $n \geq 4$.

Proof. It can be proved using the method of induction. Considering $n = 5$ in (4.4), and expanding using (4.7), at $d = 1$ it can be seen that each of its minors will show the similar structure as in (4.8). If $\mathbf{e_1}$, $\mathbf{e_2}$, $\mathbf{e_3}$ and $\mathbf{e_4}$ are mapped using (4.10), it yields the correct value of elements corresponding to the first row of each Minor of $5D$ cross-product. Similarly, $\mathbf{e_1}$, $\mathbf{e_2}$ and $\mathbf{e_3}$ in (4.3) for $3D$ unit can correctly be mapped into the elements of the first row of each minor when (4.8) is expanded using (4.7) for $4D$ cross-product computation. Proceeding this way, it can be inferred that the stated theorem will hold for nD cross-product computation case as well. The range of number of decomposition levels (d) can be proved using *Lemma 2.* ■

4.2.2 Minor Mapping

To explain the Minor mapping scheme we use the cases of $3D$ and $4D$ cross products as examples. Let us expand (4.3) using (4.7):

$$\mathbf{v_3} = \mathbf{e_1}\begin{vmatrix} v_{1,2} & v_{1,3} \\ v_{2,2} & v_{2,3} \end{vmatrix} - \mathbf{e_2}\begin{vmatrix} v_{1,1} & v_{1,3} \\ v_{2,1} & v_{2,3} \end{vmatrix} + \mathbf{e_3}\begin{vmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{vmatrix} \tag{4.12}$$

For simplicity of the discussion, without any loss of generality, here we do not consider the unit vectors and their corresponding signs (this sign-mapping will be discussed in the next subsection). Now, let us expand (4.9) using (4.7) and express it in vector-component form as below:

$$v_{4,1} = v_{1,2}\begin{vmatrix} v_{2,3} & v_{2,4} \\ v_{3,3} & v_{3,4} \end{vmatrix} - v_{1,3}\begin{vmatrix} v_{2,2} & v_{2,4} \\ v_{3,2} & v_{3,4} \end{vmatrix} + v_{1,4}\begin{vmatrix} v_{2,2} & v_{2,3} \\ v_{3,2} & v_{3,3} \end{vmatrix} \tag{4.13}$$

$$v_{4,2} = v_{1,1}\begin{vmatrix} v_{2,3} & v_{2,4} \\ v_{3,3} & v_{3,4} \end{vmatrix} - v_{1,3}\begin{vmatrix} v_{2,1} & v_{2,4} \\ v_{3,1} & v_{3,4} \end{vmatrix} + v_{1,4}\begin{vmatrix} v_{2,1} & v_{2,3} \\ v_{3,1} & v_{3,3} \end{vmatrix} \tag{4.14}$$

$$v_{4,3} = v_{1,1}\begin{vmatrix} v_{2,2} & v_{2,4} \\ v_{3,2} & v_{3,4} \end{vmatrix} - v_{1,2}\begin{vmatrix} v_{2,1} & v_{2,4} \\ v_{3,1} & v_{3,4} \end{vmatrix} + v_{1,4}\begin{vmatrix} v_{2,1} & v_{2,2} \\ v_{3,1} & v_{3,2} \end{vmatrix} \tag{4.15}$$

$$v_{4,4} = v_{1,1}\begin{vmatrix} v_{2,2} & v_{2,3} \\ v_{3,2} & v_{3,3} \end{vmatrix} - v_{1,2}\begin{vmatrix} v_{2,1} & v_{2,3} \\ v_{3,1} & v_{3,3} \end{vmatrix} + v_{1,3}\begin{vmatrix} v_{2,1} & v_{2,2} \\ v_{3,1} & v_{3,2} \end{vmatrix} \tag{4.16}$$

From (4.12), (4.13) - (4.16), as an example, considering the first minor M_{11} in (4.12) it can be observed that the first row consists of $v_{1,2}$ and $v_{1,3}$ which are different components of the first vector $\mathbf{v_1}$ whereas the first column consists of $v_{1,2}$ and $v_{2,2}$ which are the second component of the first and second vector respectively. This observation can be generalized as:

(i) Each row of a Minor (i.e vector components) belongs to the same vector (for example $[v_{i,j}\ v_{i,k}]$) and (ii) Each column of every minor corresponds to the same component of different vectors (for example $[v_{i,j}\ v_{l,j}]^T$); where from the architectural perspective, i, l may correspond to **memory bank** and j, k may correspond to **memory location**.

Since, to the best of our knowledge the research undertaken in this chapter is the first of its kind, we aim to propose here a generalized view and to explain the concept of mapping in simple fashion, without any loss of generality, in terms of architecture, one may consider that each of the vectors and their components are stored in different **memory locations** within a **memory bank**. Therefore the problem of **Minor Mapping** gets reduced to the selection problem of **memory bank** and **memory location** only.

However, memory design is an implementation issue and can be achieved in different ways for different area-throughput trade-offs. Therefore depending upon different applications and resource availability, the proposed scheme can be considered as the basic guideline and suitable implementation strategies can be adopted for its further optimization. Under such consideration, it can be seen that the memory location varies only along the row of the minors for a particular memory bank and they remain unchanged within a column where the memory bank changes, hence if the memory location selection procedure is formulated only for a single row, that can easily be repeated for different memory banks.

Theorem 2. **Memory Bank Selection -** If ${}^{(nD)}\phi_m$ denotes m^{th} memory bank for nD cross-product, then the following relationship holds:

$$ {}^{(nD)}\phi_m = {}^{(n-1)D}\phi_m + 1 \tag{4.17}$$

Proof. It can be proved using method of induction putting $n = 3, 4, ...$ in (4.4) and expanding using (4.7). For example, the memory banks in the minors of $4D$ case (see (4.13) - (4.16)) can rightly be selected in terms of $3D$ core (see (4.12)) using the above theorem. ■

To develop the generalized formula for memory location selection, let us start with comparing the minors of (4.13)-(4.16) with that of (4.12), we find following two cases are only possible:

Case (i). Memory location index of the vectors at the same minor position in (4.12) needs to be incremented by *one* when compared with each of (4.13) - (4.16). As an example, consider the first element of the first row of the second minor of (4.12) and (4.13). Comparing these two, it can be observed that the memory location-1 has to be changed into memory location-2 in the respective memory banks selected using *Theorem* 2. From this observation, it can be inferred that addition of 1 with the memory location index of $3D$ cross product generates the memory location index of the $4D$ cross product for this particular example.

Case (ii). Memory location index of the vectors at the same minor position in (4.12) can straightway be mapped into the memory locations of the vectors in (4.13) - (4.16) without any change. As an example, consider the first element of the first row of the first minor in (4.12) and (4.16). Comparing these two, it can easily be observed that memory location index does not need any change for this particular example.

Let us denote *Case (i)* and *Case (ii)* by 0 and 1 respectively. Now considering $3D$ core as shown in (4.12) it can be observed that there are three minors each of which has 2 rows and each row consists of 2 elements. Let us denote the position of the minor (as shown in (4.12)) by r where $1 \leq r \leq 3$ and the memory location of an element in any minor

by β_i where $1 \leq i \leq 2$ in this case. One $4D$ cross-product consists of 4 components as shown in (4.13) - (4.16) and let c be the positions of the components, where $1 \leq c \leq 4$.

r → / c ↓	r = 1	r = 2	r = 3
	β_1 β_2	β_1 β_2	β_1 β_2
c = 1	0 0	0 0	0 0
c = 2	0 0	1 0	1 0
c = 3	1 0	1 0	1 1
c = 4	1 1	1 1	1 1

FIGURE 4.2: 3D to 4D Memory Location Conversion Table.

Lemma 3. Using the notations introduced above, the mapping of the memory locations from $3D$ to $4D$ can be given by the Memory Location Conversion Table (**MLCT**) shown in Fig. 4.2.

Proof. By definition of the notation introduced above, $c = 1, 2, 3, 4$ indicates $v_{4,1}$, $v_{4,2}$, $v_{4,3}$ and $v_{4,4}$ (as in (4.16)) respectively as shown in (4.13) - (4.16). $\forall c, 1 \leq r \leq 3$ and $\forall r$, comparing β_i where $1 \leq i \leq 2$ in (4.12) sequentially with (4.13) - (4.16) the *Lemma* 3 can be proved. ■

In order to generalize the concept of $3D$ to $4D$ **MLCT** to $(n-1)D$ to nD **MLCT** without expanding nD cross-product in matrix form, first we introduce the following:

$$\begin{cases} \text{component position} \;=\; c : 1 \leq c \leq n \\ \text{minor position} \;=\; r : 1 \leq r \leq (n-1) \\ \text{element location within minor} \;=\; \beta_i : 1 \leq i \leq (n-2) \end{cases} \tag{4.18}$$

This set of equations (4.18) will be used to derive nD cross-product memory location from $(n-1)D$ unit.

Theorem 3. $(n-1)D$ **to** nD **MLCT construction -** Considering ${}^{n}_{(n-1)}\beta_i$ denotes the β_i involved in the $(n-1)D$ to nD **MLCT**, $(n-2)D$ to $(n-1)D$ **MLCT** can be used to construct $(n-1)D$ to nD **MLCT** using the following formula for $n > 4$.

(i) when $1 \leq c \leq (n-1)$ and $\forall c$, $1 \leq r \leq (n-2)$:

$${}^{n}_{(n-1)}\beta_i = \begin{cases} {}^{(n-1)}_{(n-2)}\beta_i & \text{for} \quad 1 \leq i \leq (n-3) \\ 0 & \text{for} \quad i = (n-2) \end{cases} \tag{4.19}$$

(ii) when $1 \leq c \leq (n-2)$ and $\forall c$, $r = (n-1)$:

$$ {}^{n}_{(n-1)}\beta_i|_{r=(n-1)} = {}^{n}_{(n-1)}\beta_i|_{r=(n-2)} \text{ for } \quad 1 \leq i \leq (n-2) \tag{4.20}$$

(iii) when $c = (n-1)$ and $r = (n-1)$:

$$ {}^{n}_{(n-1)}\beta_i = 1 \text{ for } \quad 1 \leq i \leq (n-2) \tag{4.21}$$

(iv) when $c = n$, then $\forall c$, $1 \leq r \leq (n-1)$:

$$ {}^{n}_{(n-1)}\beta_i = 1 \text{ for } \quad 1 \leq i \leq (n-2) \tag{4.22}$$

Proof. It can be proved by the method of induction for different values of n. The procedure used to prove *Lemma* 3, can be used to derive the $4D$ to $5D$ **MLCT**. Then exploiting the symmetry between $3D$ to $4D$ **MLCT** and $4D$ to $5D$ **MLCT**, the above theorem can be proved. ■

r → / c ↓	r = 1	r =2	r = 3	r = 4
	$\beta_1\ \beta_2\ \beta_3$	$\beta_1\ \beta_2\ \beta_3$	$\beta_1\ \beta_2\ \beta_3$	$\beta_1\ \beta_2\ \beta_3$
c = 1	0 0 0	0 0 0	0 0 0 →	0 0 0
c = 2	0 0 0	1 0 0	1 0 0 →	1 0 0
c = 3	1 0 0	1 0 0	1 1 0 →	1 1 0
c = 4	1 1 0	1 1 0	1 1 0	1 1 1
c = 5	1 1 1	1 1 1	1 1 1	1 1 1

FIGURE 4.3: Example of construction of $4D$ to $5D$ **MLCT** from $3D$ to $4D$ **MLCT** as shown in Fig. 4.2.

4.2.2.1 Example : Construction of $4D$ to $5D$ MLCT using $3D$ to $4D$ MLCT:

In this example first we show how to construct $4D$ to $5D$ MLCT using *Theorem* 3 and then to make it more illustrative, we show its physical significance by choosing any arbitrary row from the constructed MLCT.

Consider (4.19) - (4.22) and see Fig. 4.2. Since we plan to compute $5D$ cross-product, so choose $n = 5$. Using this in *Theorem* 3*(i)* we get, $1 \leq c \leq 4$ and $\forall c$, $1 \leq r \leq 3$ and

using this range of values of c and r in (4.19), we determine β for the **MLCT** under construction. For these specified ranges of c and r, β_1 and β_2 (since $1 \leq i \leq 2$ in (4.19)) values are same with those in Fig. 4.2 and are shown in the three vertical shaded regions in Fig. 4.3. For the same range of c and r, β_3 (when $i = 3$ in (4.19)) will simply be zero (see zero under the column of β_3 beside the vertically shaded regions in Fig. 4.3). Now consider *Theorem* 3*(ii)* where $\forall c$, $1 \leq c \leq 3$, using (4.20), $\beta_1\beta_2\beta_3$ already computed under $r = 3$ will be repeated for $r = 4$. This repetition is indicated by the arrows in Fig. 4.3). Now consider *Theorem* 3*(iii)* and *(iv)* and following the same procedure using (4.21) and (4.22), β values can be obtained which all are 1. This is also shown by the horizontal shaded region starting from the bottom-most row in Fig. 4.3 and bounded by the bold dark-line. Thus, this example demonstrates the procedure to construct $4D$ to $5D$ **MLCT** (Fig. 4.3) starting from the basic $3D$ to $4D$ **MLCT** (Fig. 4.2). This newly built $MLCT$ can again be used to construct $5D$ to $6D$ **MLCT** using exactly the same procedure and following this sequential approach any higher dimensional **MLCT** can be constructed.

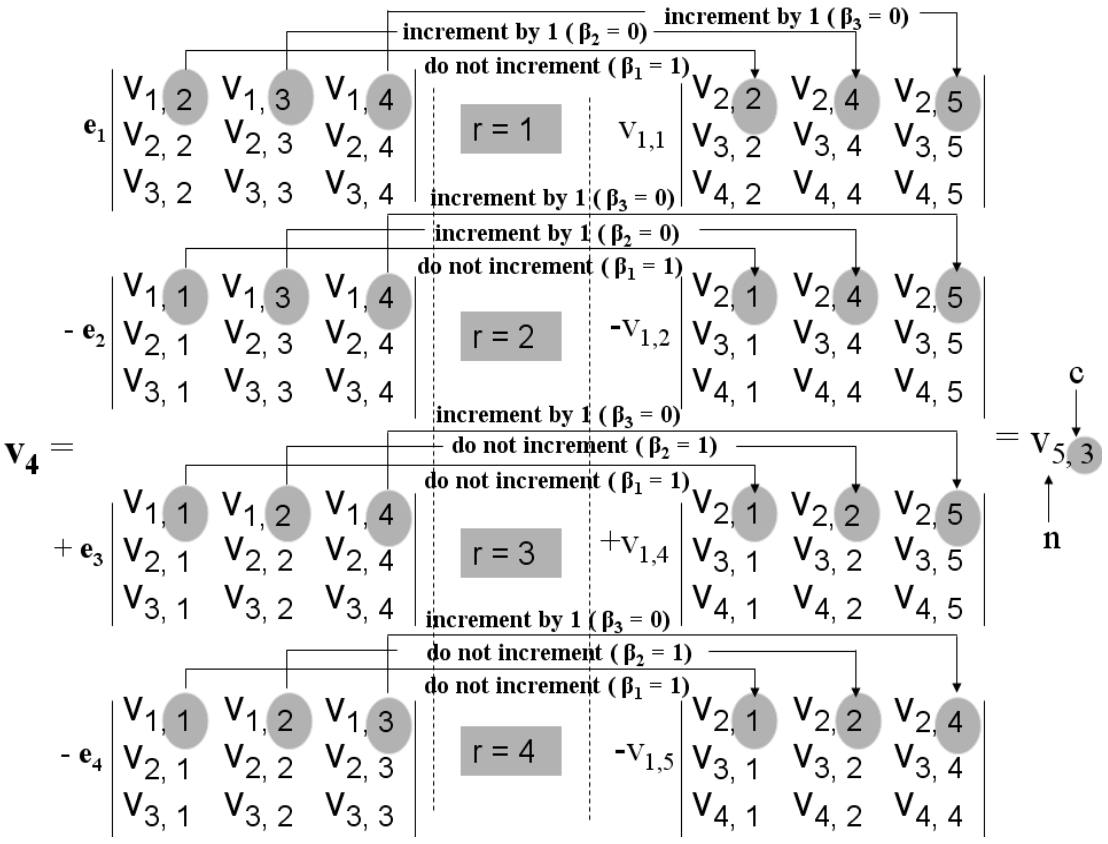


FIGURE 4.4: Structural similarity between $\mathbf{v_4}$ and the third component of $\mathbf{v_5}$. Left side: expanded form of $\mathbf{v_4}$, right side: expanded form of $v_{5,3}$.

Now we investigate the underlined principle of this constructed $4D$ to $5D$ MLCT. For that purpose, randomly choose the row corresponding to $c = 3$ in the newly constructed MLCT as shown in Fig. 4.3. Using the definition of the notations introduced in (4.18), in this MLCT, $c = 3$ signifies the 3^{rd} component of the $5D$ cross product denoted by

$v_{5,3}$. Putting $n = 5$ in (4.4) and following cofactor expansion method using (4.7), $v_{5,3}$ can be expanded as:

$$v_{5,3} = v_{1,1}\begin{vmatrix} v_{2,2} & v_{2,4} & v_{2,5} \\ v_{3,2} & v_{3,4} & v_{3,5} \\ v_{4,2} & v_{4,4} & v_{4,5} \end{vmatrix} - v_{1,2}\begin{vmatrix} v_{2,1} & v_{2,4} & v_{2,5} \\ v_{3,1} & v_{3,4} & v_{3,5} \\ v_{4,1} & v_{4,4} & v_{4,5} \end{vmatrix}$$

$$+ \mathrm{v}_{1,4}\begin{vmatrix} v_{2,1} & v_{2,2} & v_{2,5} \\ v_{3,1} & v_{3,2} & v_{3,5} \\ v_{4,1} & v_{4,2} & v_{4,5} \end{vmatrix} - v_{1,5}\begin{vmatrix} v_{2,1} & v_{2,2} & v_{2,4} \\ v_{3,1} & v_{3,2} & v_{3,4} \\ v_{4,1} & v_{4,2} & v_{4,4} \end{vmatrix} \quad (4.23)$$

The presence of the unit vector is intentionally omitted from (4.23) for brevity. It can easily be seen that there exists a structural similarity between (4.9) and (4.23). To visualize such similarity clearly, expressions of $\mathbf{v_4}$ and $v_{5,3}$ as appeared in (4.9) and (4.23) are shown in the left and right side of Fig. 4.4 respectively where all minors appear vertically. Since the memory location, as mentioned before *Theorem* 2, remains unchanged along a column within a minor, therefore the structural similarity is only shown between the first rows of each minor (denoted by r as defined in (4.18)) of $\mathbf{v_4}$ and $v_{5,3}$ in Fig. 4.4. Three interconnecting arrows across the minors represent β_1, β_2 and β_3 respectively starting from the bottom-most arrow. It can easily be verified from Fig. 4.4 that corresponding to $r = 1, 2, 3$ and 4, $\beta_1\beta_2\beta_3$ are "100", "100", "110" and "110" respectively which are the same as the row corresponding to $c = 3$ in the newly constructed MLCT in Fig. 4.3. In similar way, other components of $\mathbf{v_5}$ can be compared with the structure of $\mathbf{v_4}$ and all rows of the $4D$ to $5D$ MLCT can be verified.

4.2.3 Sign Mapping

The sign of the each component of the nD cross-product can be computed using the following theorem.

Theorem 4. **Proper Sign Selection -** Each component of the nD cross-product vector:
(i) will be positive when c is even.
(ii) needs to be complemented when c is odd.

Proof. It can be proved using the method of induction. Consider $3D$ cross-product computation as shown in (4.12), it can be seen that the component corresponding to even values of c keeps the same sign, but for odd c, the sign is complemented. The same nature holds for $4D$ cross-product computation as shown in (4.9). Then, further extending the same approach for several values of n in (4.4), the above theorem can be proved. ■

4.2.4 Illustration of the Proposed Algorithm with an Example

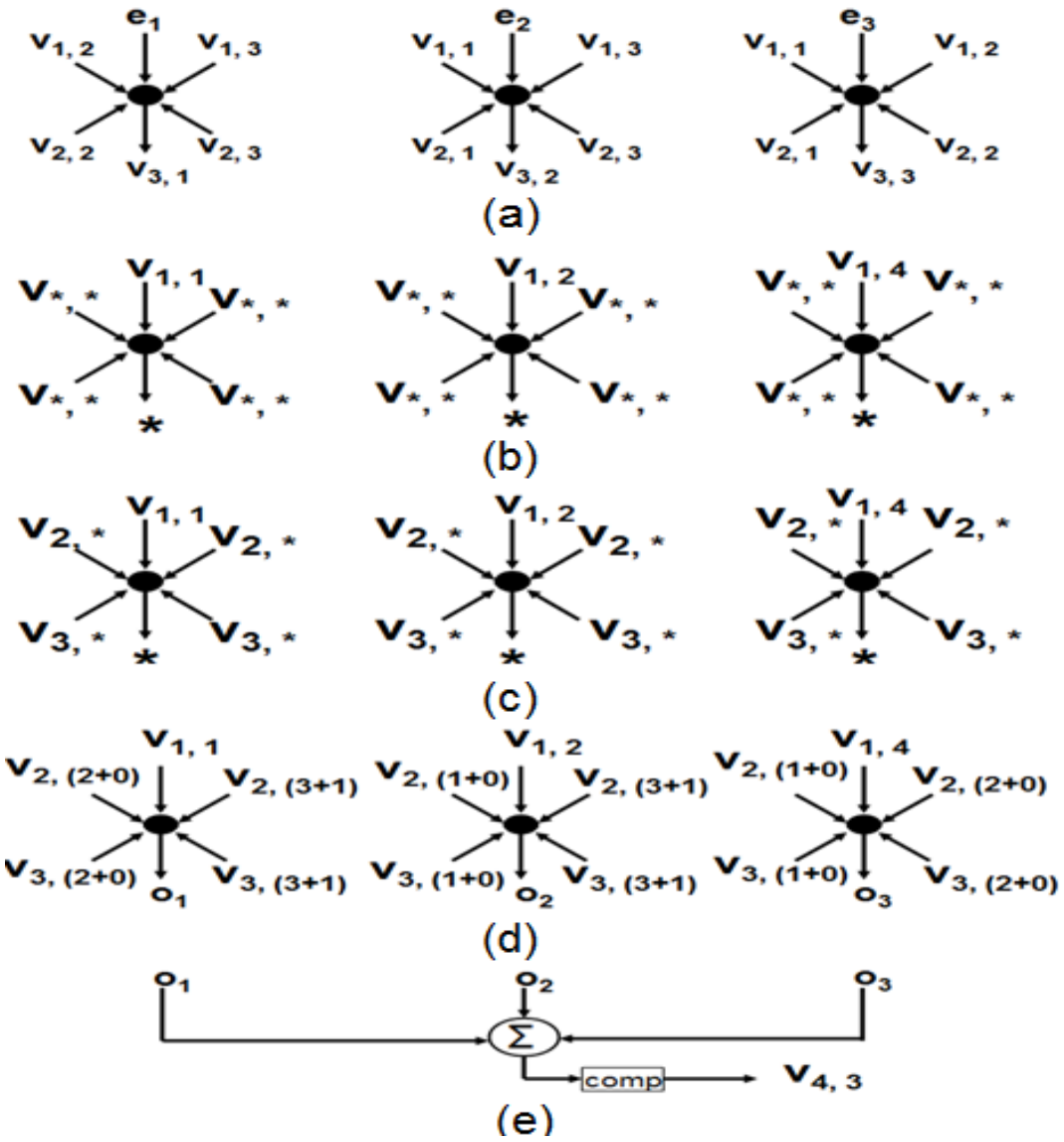


FIGURE 4.5: Illustration of the proposed algorithm for $v_{4,3}$ computation. The dark node indicates total computation involving the operands pointed by the arrows, "*" means value to be computed and "comp" means complement operation. (a) Data flow graph representation of $3D$ core, (b) step 1 - Coefficient Mapping, (c) step 2 - Memory Bank Selection, (d) step 3 - use of $3D$ to $4D$ **MLCT**, o_1, o_2 and o_3 indicates the outputs of this step and (e) step 4 - Sign Mapping.

Let us explain the principle of the proposed algorithm through the example of computing $v_{4,3}$ using $3D$ core as the fundamental operation. The data flow graph associated with this computation is depicted in Fig. 4.5 where the solid dark nodes represent respective arithmetic operations (including the signs) involved on the input data (as described in (4.12)).

step 1. **Coefficient Mapping:** The first operation of this step is to compute the range of d using (4.11). In this particular example, since $n = 4$, according to (4.11), $d = 1$.

Using this value in (4.10) one may see that $v_{1,1}$, $v_{1,2}$ and $v_{1,4}$ replace the units vectors $\mathbf{e_1}$, $\mathbf{e_2}$ and $\mathbf{e_3}$ respectively in the $3D$ core. This mapping is shown in Fig. 4.5(b).

step 2. **Memory Bank Selection:** The selection of memory banks involved for $v_{4,3}$ computation can be done using (4.17) as illustrated in Fig. 4.5(c). Note that here $v_{1,*}$ and $v_{2,*}$ of Fig. 4.5(a) are replaced with $v_{2,*}$ and $v_{3,*}$ respectively where "$*$" indicates the index of memory location to be computed.

step 3. **Use of** $3D$ **to** $4D$ **MLCT:** Since we are concerned with the third component of $\mathbf{v_4}$, refer to the row corresponds to $c = 3$ in Fig. 4.2. The formation of the memory locations based on Fig. 4.2 is shown in Fig. 4.5(d).

step 4. **Sign Mapping:** Since, in the example under consideration, $c = 3$ (as obtained in *step* 3) which is odd, therefore output needs to be complemented (according to *Theorem* 4) which is shown in Fig. 4.5(e).

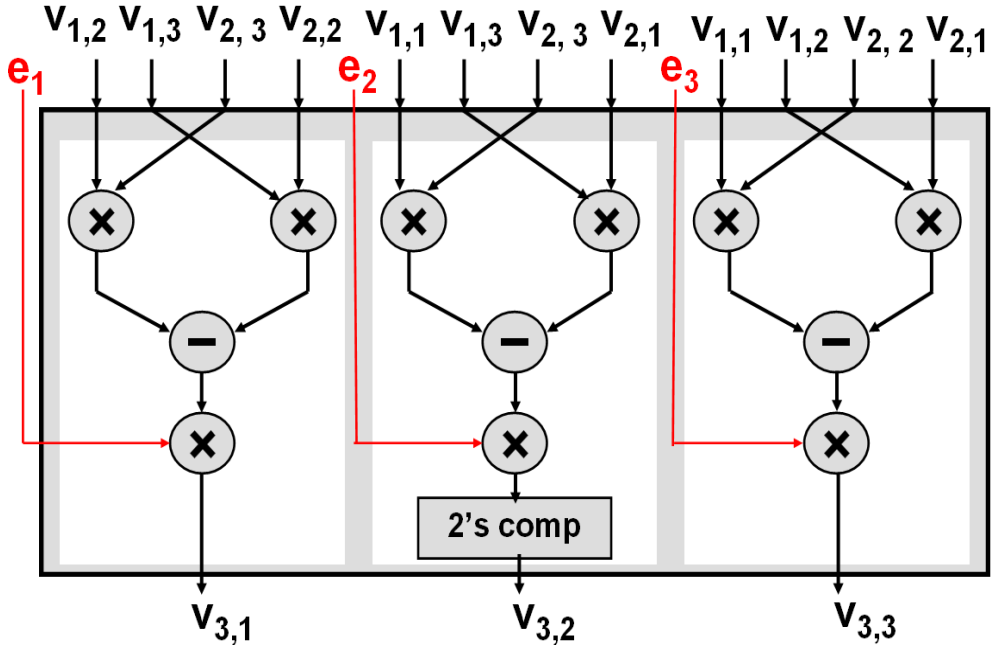


FIGURE 4.6: Architecture of $3D$ **Core** for the generalized sequential multi-dimensional cross-product computation scheme.

4.3 Architecture of the Proposed Algorithm

This section provides the basic architecture corresponding to the generalized algorithm outlined in section 3.7 and also proposes further hardware reduction methodology exploiting the internal symmetry of the nD vector cross-product matrices (now onwards will be referred as **Symmetry Based Approach**). Before we start the architectural description, it is important to note that the architectures only show the arithmetic op-

erations involved and the position of these arithmetic modules within the architectures does not provide any timing information.

4.3.1 Generalized Architecture - based on Proposed Algorithm

The architecture of $3D$ **core** is shown in Fig. 4.6 following (4.12). Since the second term of (4.12) is associated with a negative sign, therefore in Fig. 4.6 the second-term output is complemented to produce $v_{3,2}$. It is to be noted that being magnitude unity, physical presence of $\mathbf{e_i}$, $\forall i$ $(1 \leq i \leq 3)$ as inputs to the architecture for $3D$ cross product computation is not required. Fig. 4.7 shows this $3D$ cross-product computation unit.

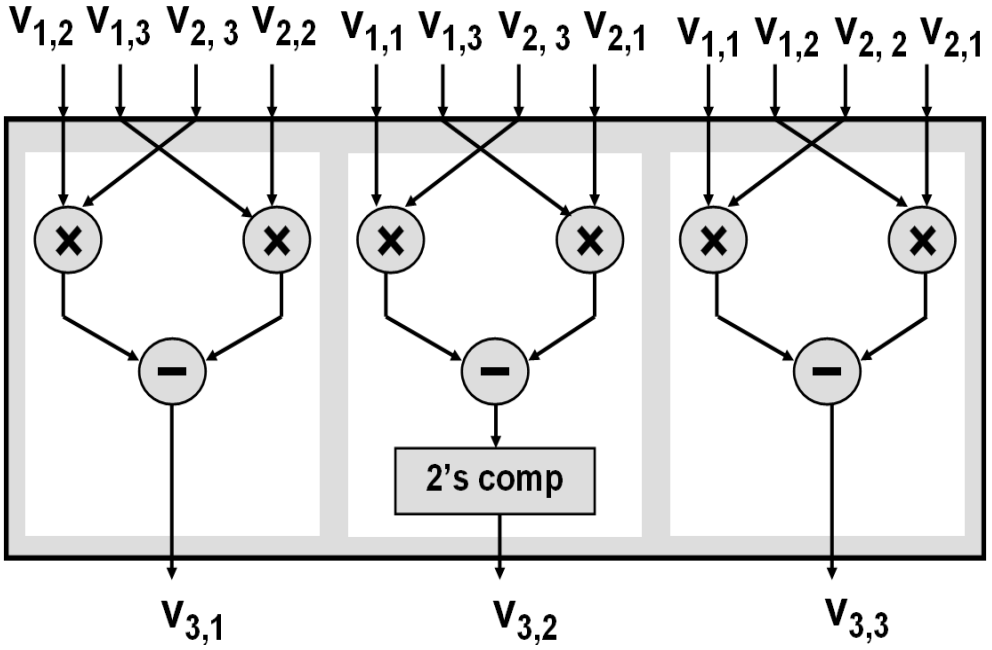


FIGURE 4.7: Architecture of $3D$ cross product computation unit.

However, as pointed out in *Theorem* 1, computation of nD cross product using $(n-1)D$ cross product computation unit (for $n > 3$) needs the mapping of unit vectors into suitable vector components and hence the presence of these unit vectors are shown in Fig. 4.6. It is important to recall from *Lemma* 1 that nD cross-product can be computed using $(n-1)D$ cross-product computation unit n times $(n > 3)$ by repetitive use of $3D$ **core** as shown in Fig. 4.6. Detailed hardware complexity and delay analysis of this generalized architecture are given in Section 3.9.

4.3.2 Hardware Reduction - Symmetry Based Approach

As mentioned in Section 3.7, the proposed generalized algorithm shows the way to realize nD cross product using $3D$ core where $n > 3$. Here we show that to compute $4D$

cross product, 4 times use of $3D$ core (as mentioned in *Lemma* 1) may not be essential and from the architectural point of view number of arithmetic operations involved in $4D$ cross product computation can be reduced by exploiting the internal symmetry of $4D$ cross product matrix. Based on this symmetry based approach, now we develop a generalized methodology and discuss its mathematical formulation. For ease of discussion, considering $4D$ cross-product computation and instead of using explicit expression, (4.13)-(4.16) can be reproduced as follow denoting the determinants of the associated minors by "**det**":

$$v_{4,1} = v_{1,2}\mathbf{det}_{1,1} - v_{1,3}\mathbf{det}_{1,2} + v_{1,4}\mathbf{det}_{1,3} \tag{4.24}$$

$$v_{4,2} = v_{1,1}\mathbf{det}_{2,1} - v_{1,3}\mathbf{det}_{2,2} + v_{1,4}\mathbf{det}_{2,3} \tag{4.25}$$

$$v_{4,3} = v_{1,1}\mathbf{det}_{3,1} - v_{1,2}\mathbf{det}_{3,2} + v_{1,4}\mathbf{det}_{3,3} \tag{4.26}$$

$$v_{4,4} = v_{1,1}\mathbf{det}_{4,1} - v_{1,2}\mathbf{det}_{4,2} + v_{1,3}\mathbf{det}_{4,3} \tag{4.27}$$

Now, comparing (4.13)-(4.16) with (4.24)-(4.27) respectively, it can be found that:

$$\begin{cases} \mathbf{det}_{1,1} = \mathbf{det}_{2,1}\ ; \mathbf{det}_{1,2} = \mathbf{det}_{3,1}\ ; \mathbf{det}_{1,3} = \mathbf{det}_{4,1} \\ \mathbf{det}_{2,2} = \mathbf{det}_{3,2}\ ; \mathbf{det}_{2,3} = \mathbf{det}_{4,2} \\ \mathbf{det}_{3,3} = \mathbf{det}_{4,3} \end{cases} \tag{4.28}$$

If the sequence of the equations are considered from (4.24)-(4.27), we call it **top-down order** and if the sequence is considered other-way round (i.e (4.27)-(4.24)), then we call it **bottom-up order**. Although this scheme does not change the orientation of the architecture, it has an effect on the generalized formulation of the symmetry based Approach as described below.

Theorem 5. **Symmetry -** *(i)* For **top-down order**, $c = \{1, 2, 3, 4\}$, $r = \{1, 2, 3\}$ and $r \geq c$, the symmetry can be given by:

$$\mathbf{det}_{c,r} = \mathbf{det}_{r+1,c} \tag{4.29}$$

(ii) For **bottom-up order**, $c = \{4, 3, 2, 1\}$, $r = \{1, 2, 3\}$ and $r < c$, the symmetry can be given by:

$$\mathbf{det}_{c,r} = \mathbf{det}_{r,c-1} \tag{4.30}$$

Proof. This theorem can be proved straightway from (4.28) and comparing (4.13)-(4.16) with (4.24)-(4.27). ■

Fig. 4.8 shows the architecture corresponding to $4D$ cross-product computation based on the symmetry based approach as given by (4.29) and (4.30). Unlike, Fig. 4.6, the operands involved for $4D$ computation are not shown intentionally to give a simplified

view of the architecture, but these operands can easily be correlated to this architecture following (4.13) - (4.16).

It should be noted that the **Coefficient Mapping** (as shown in (4.10), (4.11)), **Minor Mapping** (as shown in (4.17) and (4.19)-(4.22)) and **Sign Mapping** (as mentioned in *Theorem* 4) procedures will still be valid for the Symmetry based Approach.

From (4.29) and (4.30), it can be seen that instead of computing total 12 determinants as shown in (4.24)-(4.27), it is sufficient to compute 6 determinants (also refer to Fig. 4.8). *Theorem 5* can also be used as the memory access strategy for reusing 6 pre-computed determinants to get the results of 12 determinants.

However it can be argued that symmetry based approach appears to be less complex in terms of computational complexity but at the expense of extra storage penalty and this may negate the advantage obtained out of this symmetry with the increase of dimension n. But from the synthesized results of the symmetry based approach for different dimensions, as will be shown in the next section, we will see that even after considering the storage penalty, it gives significantly higher computational saving over the proposed generalized approach.

Since the proposed generalized cross-product computation algorithm is sequential, therefore the symmetry based $4D$ cross-product computation structure can also be used repetitively to realize nD cross-product and thereby results in less hardware complexity and faster computation. These will be discussed in detail in next section.

4.4 Architectural Performance Analysis

In this section we analyze the performance of the architectures described before in terms of hardware complexity, overall delay and the worst case bit precision error under quantization effect and provide initial post-synthesis implementation results of the example architectures based on the proposed generalized algorithm (section 3.7) and the symmetry based approach (section 3.8).

To compute overall hardware complexity and delay, two metrics - transistor counts and delay of a basic two input NAND gate are introduced. Since accurate estimation of these two parameters are difficult because of the basic differences in the technology employed, assumptions made for the derivations and their implementation dependencies, our attempt here is to provide the comparative studies which, we believe, can be considered as the first-order approximation to the actual values of the hardware complexity and delay. To do this, first the generic hardware complexity and delay expression are

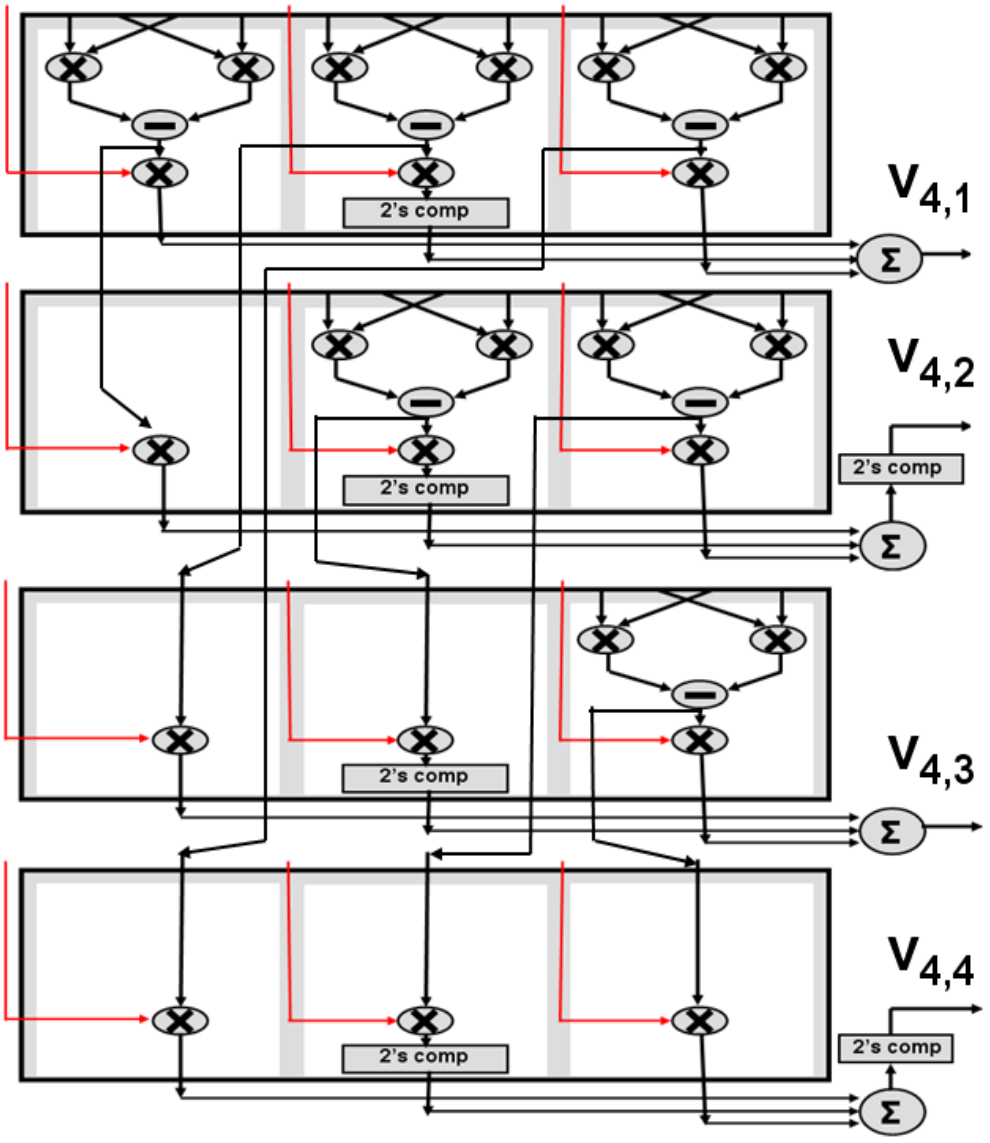


FIGURE 4.8: Architecture of $4D$ cross-product computation unit based on **symmetrical approach**.

derived irrespective of any specific implementation. Then to compare the results under the uniform platform, we consider only Ripple Carry Adder (RPA) and Subtractor (RPS), Conventional Array Multiplier (CAM) and basic combinatorial control-enabled 2's complementer structure.

4.4.1 Hardware Complexity

Since different arithmetic units can be implemented in different ways, we consider the flat unfolded architecture without any resource-sharing capability following [122]. Under these assumptions, the hardware complexity of the architectures in terms of the arithmetic modules can equivalently be expressed in terms of total arithmetic operations involved [122].

Lemma 4. Denoting number of multiplier, subtractor and two's complement by M, S, C, total hardware involved to realize the $3D$ **Cross-product** ($^{3D}HW_{org}$) can be expressed as:

$$\begin{aligned} {}^{3D}HW_{org} &= 3 \times (2 \times M + S) + \lfloor 3/2 \rfloor \times C \\ &= 3 \times {}^{3D}HW_{org,u} + \lfloor 3/2 \rfloor \times C \end{aligned} \tag{4.31}$$

where, $^{3D}HW_{org,u}$ = the hardware involved to compute each determinant of the minor (see (4.12)) is expressed as:

$$^{3D}HW_{org,u} = 2 \times M + S \tag{4.32}$$

Proof. It can be proved straightway from Fig. 4.6 and not considering the physical presence of the unit vectors as outlined in Section 4.3. ■

4.4.1.1 Proposed Generalized Approach

Here, we provide a recursive formulation for the hardware complexity of the architecture based on the proposed generalized algorithm discussed in Section 4.3.

Lemma 5. Denoting hardware complexity of the nD cross-product computation unit based on the proposed generalized algorithm by $^{nD}HW_{str}$ and n-operand adder by ^{n}A, the following formula holds:

$$^{4D}HW_{str} = 4 \times {}^{3D}HW_{core} + 4 \times {}^{3}A + \lfloor 4/2 \rfloor \times C \tag{4.33}$$

where, $^{3D}HW_{core}$ is the hardware complexity of the $3D$ **Core** as shown in Fig. 4.6 and can be given by:

$$^{3D}HW_{core} = 3 \times (3 \times M + S) + \lfloor 3/2 \rfloor \times C \tag{4.34}$$

Proof. $^{3D}HW_{core}$ can be derived straightway from Fig. 4.6. From each of (4.13) - (4.16), it can be observed that structurally each $3D$ component has to be added to result in a $4D$ component and thereby necessitating the use of 3-operand adders. It is relevant from *Remark* 2 and *Lemma* 1 that 4 such adders are necessary to realize complete $4D$ cross-product. Moreover, from *Theorem* 4, it can be said that two 2's complementer will be needed to obtain $v_{4,2}$ and $v_{4,4}$ with proper sign. Taking all into consideration, *Lemma* 5 can be proved. ■

Theorem 6. Following the notations introduced in *Lemma* 4 and 5, the hardware complexity of the nD structural cross-product computation unit can be expressed in terms of the hardware complexity involved in $(n-1)D$ cross-product in recursive way as follows:

$$^{nD}HW_{str} = n \times {}^{(n-1)D}HW_{str} + n \times {}^{(n-1)}A + \lfloor n/2 \rfloor \times C \tag{4.35}$$

considering, ${}^{3D}HW_{core}$ as the initial unit.

Proof. It can be proved using the method of induction. Following the same procedure used to obtain (4.33) in *Lemma* 5, expanding (4.4) for different values of $n \geq 4$, the above theorem can be proved. ■

4.4.1.2 Symmetry Based Approach

Here, we derive the recursive formula to obtain the hardware complexity of nD cross-product in terms of $(n-1)D$ cross-product computation based on the symmetry based approach as detailed in Section 4.3.

Lemma 6. Denoting hardware complexity of the nD cross-product computation unit based on symmetry by ${}^{nD}HW_{sym}$, the following formula holds:

$$\begin{aligned} {}^{4D}HW_{sym} = 6\times {}^{3D}HW_{org,u} + 4\times 3M + 4\times {}^{3}A \\ + (4\times \lfloor 3/2\rfloor + \lfloor 4/2\rfloor)\times C \end{aligned} \tag{4.36}$$

where, ${}^{3D}HW_{org,u}$ is defined in (4.32).

Proof. Consider the symmetry mentioned in (4.29) and (4.30) and denote the hardware complexity for $v_{4,i}$ computation (see Fig. 4.8) by ${}^{4D}HW_{sym,i}$ where, $\forall i,\ 1 \leq i \leq 4$. From Fig. 4.8, the following can be derived:

$${}^{4D}HW_{sym,i} = (4-i)\times {}^{3D}HW_{org,u} + 3\times M + \lfloor 3/2\rfloor \times C + {}^{3}A \tag{4.37}$$

Taking sum of (4.37) $\forall i,\ 1 \leq i \leq 4$, (4.36) can be proved. ■

Theorem 7. The hardware complexity of nD cross-product computation unit based on symmetry can be expressed in terms of the hardware complexity involved in $(n-1)D$ cross-product in recursive way as follows:

$${}^{nD}HW_{sym} = {}^{(n-1)D}HW_{sym} + n\times {}^{(n-1)}A + \lfloor n/2\rfloor \times C \tag{4.38}$$

considering ${}^{4D}HW_{sym}$ as the initial unit.

Proof. It can be proved using the method of induction. Procedure adopted to prove (4.36) using (4.37) can be followed here and can be expanded for different values of $n \geq 4$ in (4.4) to reach (4.38). ■

4.4.1.3 Comparison of Hardware Complexities

To do the comparative study of the hardware complexities (see (4.35) and (4.38)) of the two architectures for consideration, we follow the approach described in [86] and make

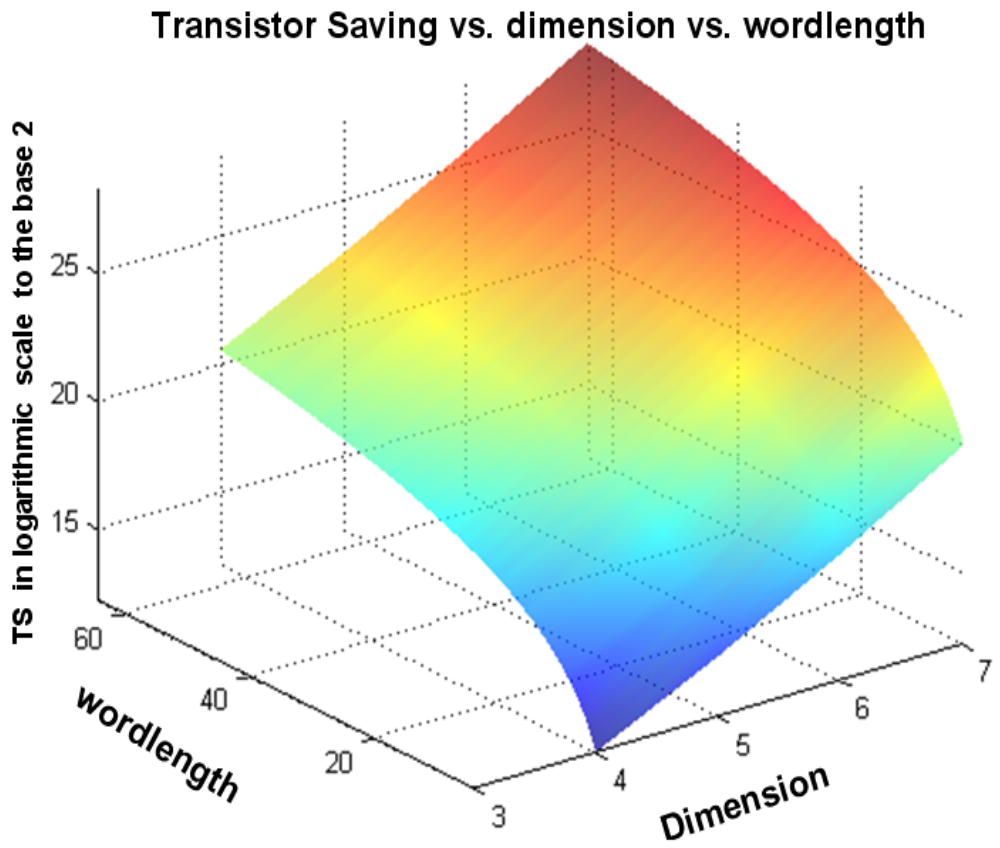


FIGURE 4.9: Variation of Transistor Savings (TS) with dimension and word-length

the following assumptions:

AS 1.1. *(i)* One b-bit RCA and RPS needs b Full-Adders (FA) [83], *(ii)* one b by b CAM needs $b(b-2)$ FA, b Half-Adders (HA) and b^2 AND gates [84], *(iii)* one b-bit 2's complementer needs $(b-1)$ OR gates, b AND gates and b XOR gates [83].

AS 1.2. Transistor Count (TC) of *(i)* 1-bit FA is 24, *(ii)* 1-bit HA is 12, *(iii)* one 2-input XOR gate is 12, *(iv)* one 2-input AND gate is 6 and *(v)* one 2-input OR gate is 6 [62].

Although, several optimized multi-operand adder structures are available, for ease of comparison we will only consider two-operand Adders.

Lemma 7. m-operand Adder (^mA) can be dissolved into $(m-1)$ cascade levels each consisting of one two-operand adder (^2A) structure and this method can be expressed as:

$$^mA = (m-1) \times (^2A) \tag{4.39}$$

Proof. Consider three-operand adder and it can easily be shown that $(3-1) = 2$ level of cascade of two-operand adders is equivalent to three-operand adder. Proceeding the same way for several values of m, (4.39) can be proved. ■

Considering *AS* 1.1 and *AS* 1.2 and using (4.39) in (4.35) and (4.38), the hardware complexities can be expressed in terms of TC for both of the architectures. Then,

taking the difference between these results, Transistor Saving (TS) for the symmetry based approach can be obtained over the structural approach and Fig. 4.9 shows the variation of TS in logarithmic scale (in base 2) with the change of dimension and word-length. The range of word-length and dimension are considered between 4 to 64 and 3 to 7 respectively. Realization of $3D$ cross-product computation is same under both the architectures under consideration and therefore Fig. 4.9 shows that for $3D$, $TS = 0$. But from $4D$ onwards, as can be observed from Fig. 4.9, for a fixed dimension, TS will increase with word-length and for a fixed word-length TS is significant at higher dimension.

4.4.2 Delay Analysis

Delay can be defined as the time between the first input to do the first computation and the last output from the final computation [51]. In this context of delay analysis, we consider deriving the worst-case delay of both of the architectures based on the proposed generalized algorithm and symmetry based approach. Moreover, we assume no parallel computing capability of the architectures under consideration and therefore we consider only one arithmetic operation is permitted at one cycle [122]. No interconnect delay is considered. Under these assumptions the worst-case delay of the architectures will be the additive delay of the individual arithmetic module involved for overall computation [122].

Lemma 8. Denoting the delay associated with a multiplier, subtractor, 2's complementer by τ_m, τ_s and τ_c, the worst-case delay of the **original** $3D$ **cross-product unit** ($^{3D}\Gamma_{org}$) can be given as:

$$\begin{aligned} {}^{3D}\Gamma_{org} &= 3 \times (2 \times \tau_m + \tau_s) + \lfloor 3/2 \rfloor \times \tau_c \\ &= 3 \times {}^{3D}\Gamma_{org,u} + \lfloor 3/2 \rfloor \times \tau_c \end{aligned} \tag{4.40}$$

where, $^{3D}\Gamma_{org,u}$ = the delay to compute each determinant of the minor (see (4.12)) is expressed as:

$${}^{3D}\Gamma_{org,u} = 2 \times \tau_m + \tau_s \tag{4.41}$$

Proof. It can be proved straightway from Fig. 4.6 and using (4.31) and (4.32). ■

4.4.2.1 Proposed Generalized Approach

Here, we provide a recursive formulation of the delay of the architecture based on the proposed generalized algorithm as discussed in Section 4.3.

Lemma 9. Denoting delay of the nD cross-product computation unit based on the proposed generalized algorithm by $^{nD}\Gamma_{str}$ and delay of n-operand adder by $^{n}\tau_a$, the

following formula holds:

$$^{4D}\Gamma_{str} = 4 \times {}^{3D}\Gamma_{core} + 4 \times {}^{3}\tau_a + \lfloor 4/2 \rfloor \times \tau_c \tag{4.42}$$

where, $^{3D}\Gamma_{core}$ is the delay of the $3D$ **Core** as shown in Fig. 4.6 and can be given by:

$$^{3D}\Gamma_{core} = 3 \times (3 \times \tau_m + \tau_s) + \lfloor 3/2 \rfloor \times \tau_c \tag{4.43}$$

Proof. $^{3D}\Gamma_{core}$ can be derived straightway from Fig. 4.6 and using (4.33) and (4.34). ■

Theorem 8. The delay of the proposed generalized algorithm based nD cross-product computation unit ($^{nD}\Gamma_{str}$) can be expressed in terms of the delay involved in $(n-1)D$ cross-product ($^{(n-1)D}\Gamma_{str}$) in recursive way as follows:

$$^{nD}\Gamma_{str} = n \times {}^{(n-1)D}\Gamma_{str} + n \times {}^{(n-1)}\tau_a + \lfloor n/2 \rfloor \times \tau_c \tag{4.44}$$

considering, $^{3D}\Gamma_{core}$ as the initial unit.

Proof. Proof follows using the same procedure adopted to prove (4.35). ■

4.4.2.2 Symmetry Based Approach

The recursive formulation for expressing the delay of nD cross-product computation unit using the delay of $(n-1)D$ cross-product computation unit based on the symmetry is derived here.

Lemma 10. Denoting the delay of the nD cross-product computation unit based on symmetry by $^{nD}\Gamma_{sym}$, the delay of the $4D$ cross-product computation based on the same ($^{4D}\Gamma_{sym}$) can be expressed as:

$$\begin{aligned} ^{4D}\Gamma_{sym} = 6 \times {}^{3D}\Gamma_{org,u} + 4 \times 3\tau_m + 4 \times {}^{3}\tau_a \\ + (4 \times \lfloor 3/2 \rfloor + \lfloor 4/2 \rfloor) \times \tau_c \end{aligned} \tag{4.45}$$

where, $^{3D}\Gamma_{org,u}$ is defined in (4.41).

Proof. Following the same procedure to prove (4.36), the delay for $v_{4,i}$ computation ($^{4D}\Gamma_{sym,i}; \forall i,\ 1 \leq i \leq 4$) can be given by:

$$^{4D}\Gamma_{sym,i} = (4-i) \times {}^{3D}\Gamma_{org,u} + 3 \times \tau_m + \lfloor 3/2 \rfloor \times \tau_c + {}^{3}\tau_a \tag{4.46}$$

Proof follows by taking the sum of (4.46) $\forall i,\ 1 \leq i \leq 4$. ■

Theorem 9. The delay of nD cross-product computation unit based on symmetry can be expressed in terms of the delay associated in $(n-1)D$ cross-product in recursive way

as follows:

$$^{nD}\Gamma_{sym} = {}^{(n-1)D}\Gamma_{sym} + n \times {}^{(n-1)}\tau_a + \lfloor n/2 \rfloor \times \tau_c \tag{4.47}$$

considering $^{4D}\Gamma_{sym}$ as the initial unit.

Proof. Proof follows using the method adopted to prove (4.38). ■

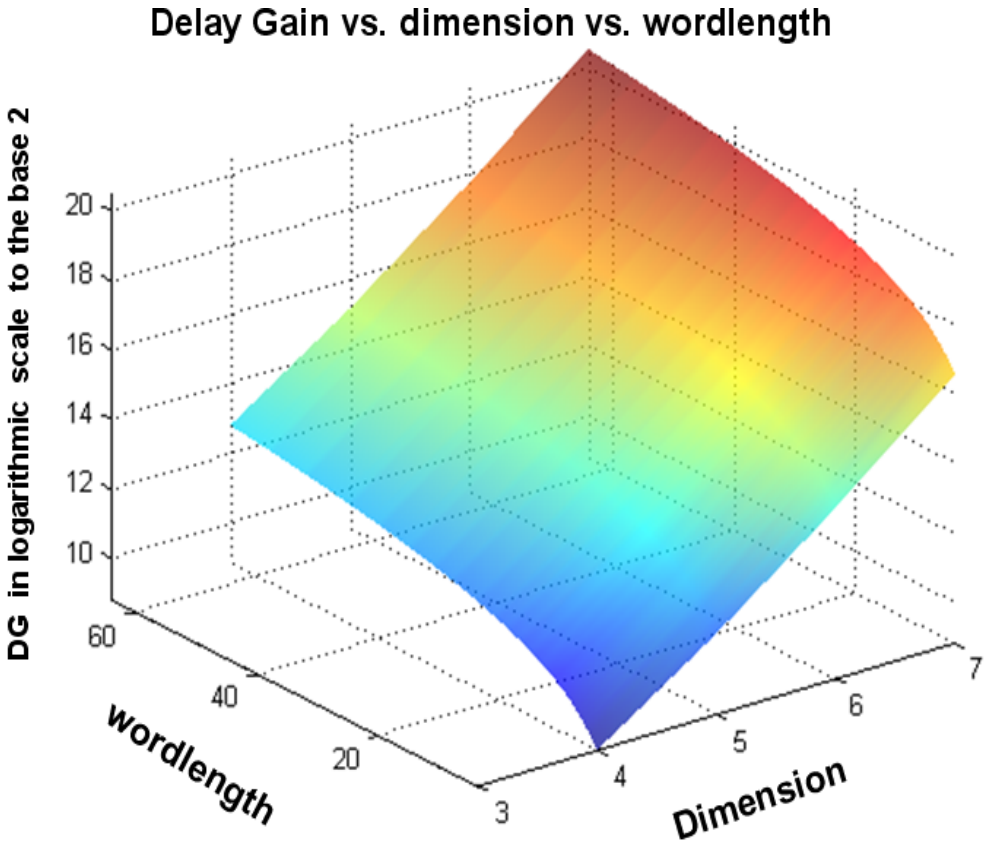


FIGURE 4.10: Variation of Delay Gain (DG) with dimension and word-length

4.4.2.3 Comparison of Delays

To do the comparison of two architectures on the basis of worst-case delay derived in (4.44) and (4.47) following assumptions are considered:
AS 2.1. Denoting the delay of a two-input NAND gate by $\triangle$, the delay of *(i)* b-bit two-operand RCA and RCS is $2b\triangle$ (τ_a and τ_s), *(ii)* b-by-b CAM is $8b\triangle$ ($= \tau_m$) and *(iii)* b bit combinatorial 2's complementer is $(2b+3)\triangle$ ($= \tau_c$) [83].

Since multi-operand adder is present in (4.44) and (4.47), the corresponding delay can be expressed in terms of the delay of two-operand adder following the concept presented in (4.39). Using *AS* 2.1 in (4.44) and (4.47), $^{nD}\Gamma_{str}$ and $^{nD}\Gamma_{sym}$ can be expressed in terms of $\triangle$. Then, taking their difference and normalizing the result with respect to $\triangle$, the normalized Delay Gain (DG) for the symmetrical approach based architecture

can be obtained over of the structural approach based architecture. Fig. 4.10 plots the variation of DG in logarithmic scale to the base 2 with different word-length (considered in the range from 4 to 64) and dimensions (considered within the range from 3 to 7). Since $3D$ cross product computation under both of the architectures is same, Fig. 4.10 shows that no gain in delay is achieved for this case. But $4D$ cross product computation onwards DG will increase with the word-length for any fixed dimension and significant gain in delay is achieved in higher dimension for fixed word-length.

4.4.3 Precision Error Analysis

Since numerical error analysis of any algorithm is essential for efficient design of its corresponding architecture, this section studies the precision error propagation throughout the architecture, provides a generalized recursive formulation of worst-case error for nD cross-product computation and presents the criterion for optimum word-length selection to achieve the final accuracy of one unit in the last place (ULP) under the normalized floating point representation of any data. Following assumptions are made on the sources of error:

AS 3.1 (i) rounding error introduced by the present computation, *(ii)* rounding error propagated from previous computations and *(iii)* initial input data are also assumed to be rounded because it is most likely that the cross-product computation unit is fed by the data generated by some other processing units.
It is worth mentioning that we consider the combinatorial 2's complementer circuit which does not contribute any error and so irrespective of its presence, the worst-case error bound derived below will be valid. For derivation, we follow the error computation approach used in [123]. Therefore we assume that any elementary arithmetic operation can only take place between two arguments. Now, let us consider any point on the data-flow path through the architecture where error occurs if any one or more of the assumptions mentioned in AS 3.1 is valid and define this point as node. Let us denote the computational process involved in any node by ξ and also denote the relative error associated with each ξ by ϵ_ξ. For example, consider the block computing $v_{3,1}$ from Fig. 4.6. As per the definition of node and from the assumptions AS 3.1, the computational processes involved can be written as: $v_{1,2} = \xi_1$, $v_{2,3} = \xi_2$, $v_{1,3} = \xi_3$, $v_{2,2} = \xi_4$, $\xi_1 \times \xi_2 = \xi 5$, $\xi_3 \times \xi_4 = \xi_6$, $\xi_5 \times \xi_6 = \xi_7$ and the respective relative errors associated with these processes can be denoted as ϵ_{ξ_i} where $1 \leq i \leq 7$. It is to be noted that AS 3.1*(iii)* will contribute to ϵ_{ξ_i} for $1 \leq i \leq 4$ and AS 3.1*(ii)* will contribute to ϵ_{ξ_i} for $5 \leq i \leq 7$. Let us denote the rounding error introduced by the present computation (as per AS 3.1*(i)*) for the processes ξ_i by $\epsilon_{\xi_i}^{(r)}$ where $5 \leq i \leq 7$. Now tracing through these several computational processes, error relations can be found as follow [124]:

$$\epsilon_{\xi_5} = \epsilon_{\xi_1} + \epsilon_{\xi_2} + \epsilon_{\xi_5}^{(r)} \tag{4.48}$$

$$\epsilon_{\xi_6} = \epsilon_{\xi_3} + \epsilon_{\xi_4} + \epsilon_{\xi_6}^{(r)} \tag{4.49}$$

$$\epsilon_{\xi_7} = \frac{\xi_5}{\xi_5 - \xi_6}\epsilon_{\xi_5} - \frac{\xi_6}{\xi_5 - \xi_6}\epsilon_{\xi_6} + \epsilon_{\xi_7}^{(r)} \tag{4.50}$$

Considering no error damping or error cancellation and denoting $|\epsilon_{\xi_{5,6}}| = \max\{|\epsilon_{\xi_5}|, |\epsilon_{\xi_6}|\}$, the upper bound of ϵ_{ξ_7} can be written as using (4.50):

$$|\epsilon_{\xi_7}| \leq |\epsilon_{\xi_{5,6}}| + |\epsilon_{\xi_7}^{(r)}| \tag{4.51}$$

Considering the computation with t-significant binary places, then the upper bound for machine precision ϵ_m can be expressed as: $|\epsilon_m| \leq 2^t$ [123]. Now, using this, the upper-bound of $\epsilon_{\xi_{5,6}}$ can be given by:

$$|\epsilon_{\xi_{5,6}}| \leq 3|\epsilon_m| \tag{4.52}$$

Lemma 11. For $3D$ **cross-product unit**, the worst-case error bound ($\epsilon_{org}^{(3D)}$) associated with each vector component can be given as:

$$|\epsilon_{org}^{(3D)}| \leq 4|\epsilon_m| \tag{4.53}$$

Proof. ϵ_{ξ_7} in (4.51) corresponds to $\epsilon_{org}^{(3D)}$. Proof follows using (4.52) in (4.51). ■

Lemma 12. For the $3D$ **core**, the worst-case error bound ($\epsilon_{core}^{(3D)}$) associated with each vector component can be given as:

$$|\epsilon_{core}^{(3D)}| \leq 6|\epsilon_m| \tag{4.54}$$

Proof. Considering $v_{3,1}$ computation from Fig. 4.6 it can be observed that two computational processes viz. ξ_8 and ξ_9 are included which correspond to $\mathbf{e_1}$ and $(\xi_7 \times \xi_8)$ respectively. ξ_9 will introduce $\epsilon_{\xi_9}^{(r)}$. Therefore, ϵ_{ξ_9} can be given as:

$$\epsilon_{\xi_9} = \epsilon_{\xi_7} + \epsilon_{\xi_8} + \epsilon_{\xi_9}^{(r)} \tag{4.55}$$

ϵ_{ξ_9} corresponds to $\epsilon_{core}^{(3D)}$. Proof follows using (4.53) in (4.55) and using individual upper-bound of ϵ_{ξ_7} and $\epsilon_{\xi_9}^{(r)}$. ■

Point to be noted that, although the worst-case relative error bound to be formulated for nD cross-product is based on the proposed generalized algorithm, the same derivations are valid for the symmetry based approach as well.

Lemma 13. Denoting the relative error associated with the nD cross-product computa-

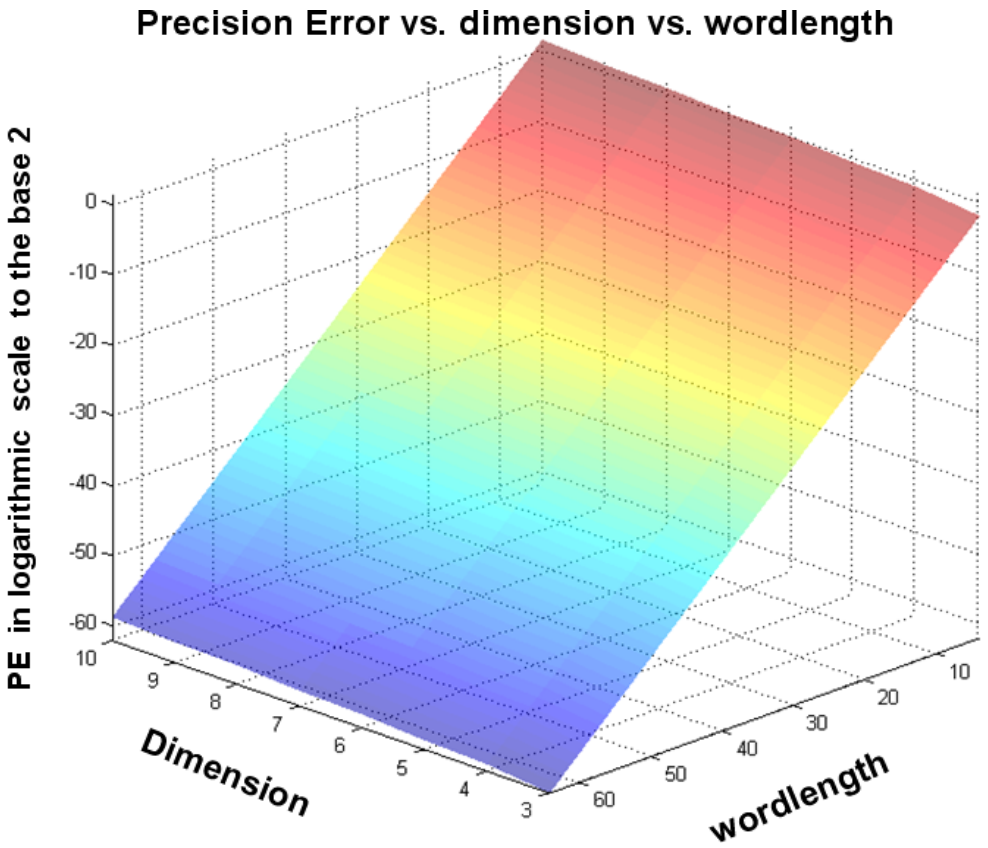


FIGURE 4.11: Variation of Precision Error (PE) with dimension and word-length

tion unit by $\epsilon_{(nD)}$, the following formula holds:

$$|\epsilon_{(4D)}| \leq |\epsilon_{core}^{(3D)}| + 2|\epsilon_m| \tag{4.56}$$

Proof. Comparing (4.13)-(4.16) with (4.12), it can easily be seen that addition of three components of a $3D$ **core** results in one component of $4D$ cross-product. According to (4.39), two cascade stages of two-operand adders equivalently represent one three-operand adder. This introduces two more computational processes ξ_{10} and ξ_{11} with the associated rounding error $\epsilon_{\xi_{10}}^{(r)}$ and $\epsilon_{\xi_{11}}^{(r)}$ respectively. Following the same procedure used before, $\epsilon_{\xi_{10}}$ and $\epsilon_{\xi_{11}}$ can be given as follows:

$$|\epsilon_{\xi_{10}}| \leq |\epsilon_{\xi_9}| + |\epsilon_{\xi_{10}}^{(r)}| \tag{4.57}$$

$$|\epsilon_{\xi_{11}}| \leq |\epsilon_{\xi_{10}}| + |\epsilon_{\xi_{11}}^{(r)}| \tag{4.58}$$

Proof follows using (4.54) in (4.57) and replacing $\epsilon_{\xi_{10}}$ in (4.58). ■

Theorem 10. For $n \geq 4$, the recursive relation between $\epsilon_{(nD)}$ and $\epsilon_{(n-1)D}$ can be given as follows:

$$|\epsilon_{(nD)}| \leq |\epsilon_{(n-1)D}| + (n-2)|\epsilon_m| \tag{4.59}$$

Proof. Using the procedure adopted to prove (4.56) can be extended for $n > 4$ in the similar fashion and (4.59) can be proved. ■

Corollary 1. For $n \geq 4$, $\epsilon_{(nD)}$ can be represented in terms of $\epsilon_{core}^{(3D)}$ as follows:

$$|\epsilon_{(nD)}| \leq |\epsilon_{core}^{(3D)}| + \frac{n(n-3)}{2}|\epsilon_m| \tag{4.60}$$

Proof. Proof follows by putting $n = 4, 5, 6, \ldots$ in (4.59). ■

Fig. 4.11 plots the variation of relative precision error (PE) in logarithmic scale to the base 2 as obtained from (4.60) with different dimensions (range considered is within 3 to 10) and word-lengths (range considered is within 4 to 64). For $3D$ case, (4.53) is used in Fig. 4.11. As expected, it can be seen from Fig. 4.11 that for a fixed dimension, PE will decrease with the increase of word-length and for fixed word-length, PE increases with the increase of dimension.

Using (4.54) in (4.60) we get:

$$\begin{aligned} |\epsilon_{(nD)}| &\leq \{6 + \frac{n(n-3)}{2}\}|\epsilon_m| \\ &\leq K|\epsilon_m| \end{aligned} \tag{4.61}$$

where $K = 6 + \frac{n(n-3)}{2}$.

Theorem 11. To obtain a precision of t-bits at the final output of nD cross-product, $t + \lceil log_2(K) \rceil$ bits are required at the input.

Proof. Consider the databus width is b-bit and denote $(b - t)$ by g where t-bit precision is expected at the output. Therefore, using (4.61), the expected condition can be framed as $2^{-t} \geq K|\epsilon_m|$ where $|\epsilon_m| \leq 2^b$ and $b = (t + g)$. From here, the bound of g can be obtained as:

$$g \geq \log_2(K) \tag{4.62}$$

The proof follows from (4.62). ■

4.5 Experimental Results

The purpose of this section is to provide an insight to the implementation results in terms of the power dissipation and area consumption of the architectures based on the proposed generalized algorithm and the Symmetry based Approach introduced in Section 4.3. The architectures are coded in VHDL for $3D$, $4D$ and $5D$ cases with different word-lengths, synthesized using Synopsys Design Compiler using 0.13 μm technology library @1 MHz frequency with the $V_{dd} = 1.2V$ and power dissipation results are obtained using

Synopsys Prime Time. The throughput of the designed system is one complete output per clock cycle at the steady state. However it is important to note that all experimental results to be presented here are simply for proof of the concept discussed in previous sections.

TABLE 4.1: Post-synthesis Power, area consumption and gate complexity results for $3D$ cross-product computation unit for different word-lengths (W)

W (bits)	Power (μW)	Area (mm^2)	NAND Gate complexity
4	0.5294	0.0046	760
8	1.3606	0.0123	2032
16	4.9989	0.0387	6395
32	22.4658	0.1365	22554

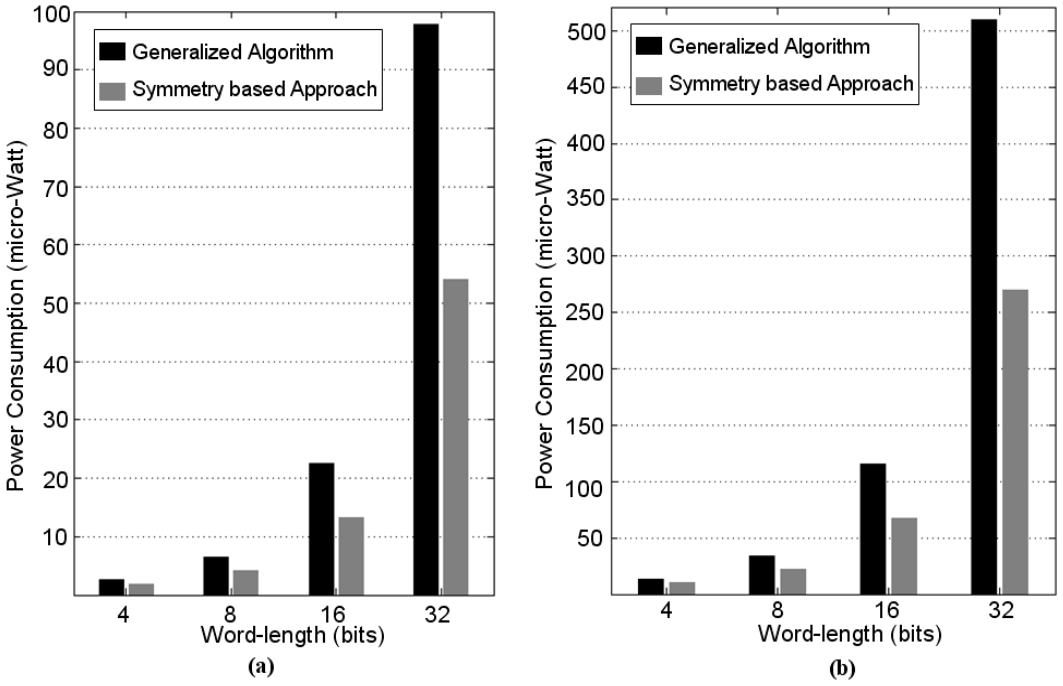


FIGURE 4.12: Comparative Post synthesis Power consumption results for different Word-lengths using proposed Generalized Algorithm and Symmetry-based Approach for (a) $4D$ cross product, (b) $5D$ cross product.

Table 4.1 and 4.2 show the power dissipation, area consumption and gate complexities of the $3D$ cross-product computation unit, and $4D$ and $5D$ cross product computation units. Fig. 4.12, 4.13 and 4.14 also present the power, area consumption and NAND gate complexity comparison for $4D$ and $5D$ cross product computation units for the proposed generalized as well as the symmetry based approach respectively. As expected, it can be seen from each of the tables that the power dissipation and area consumption increase with the word-length for a fixed dimension. Similarly, it can also be noted from these tables that for a fixed word-length, the power dissipation and area consumption increase with the dimension. Another point to be noted from Table 4.2 that both the power dissipation and area consumption for any word-length by the Symmetry based Approach (shown as Approach-B in Table 4.2) is less than that in the architecture based

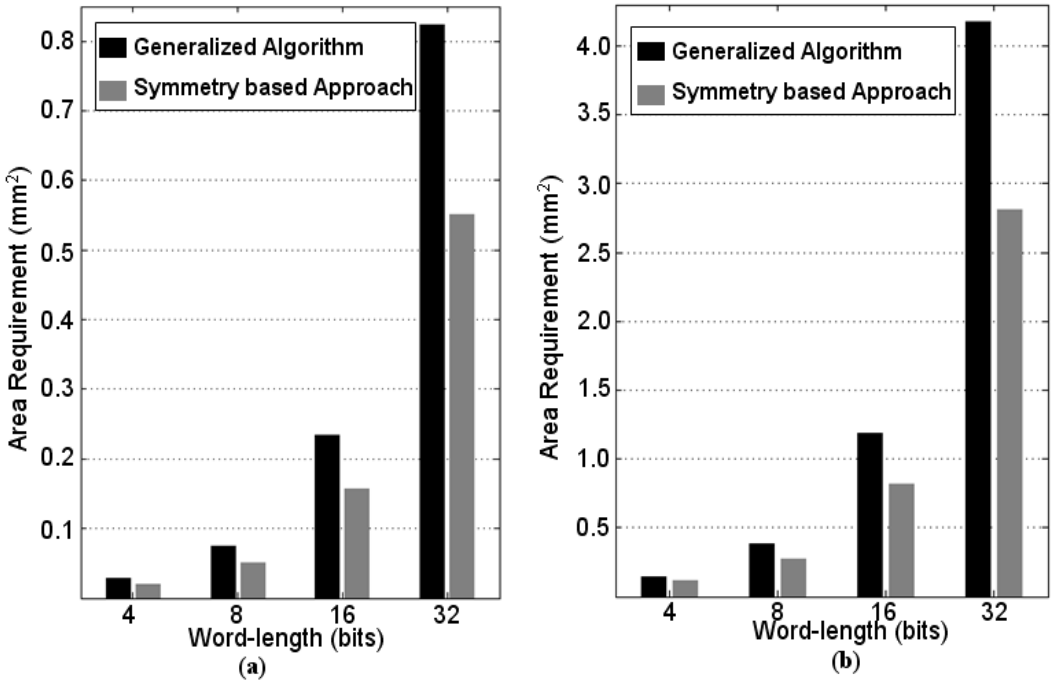


FIGURE 4.13: Comparative Post synthesis Area consumption results for different Word-lengths using proposed Generalized Algorithm and Symmetry-based Approach for (a) $4D$ cross product, (b) $5D$ cross product.

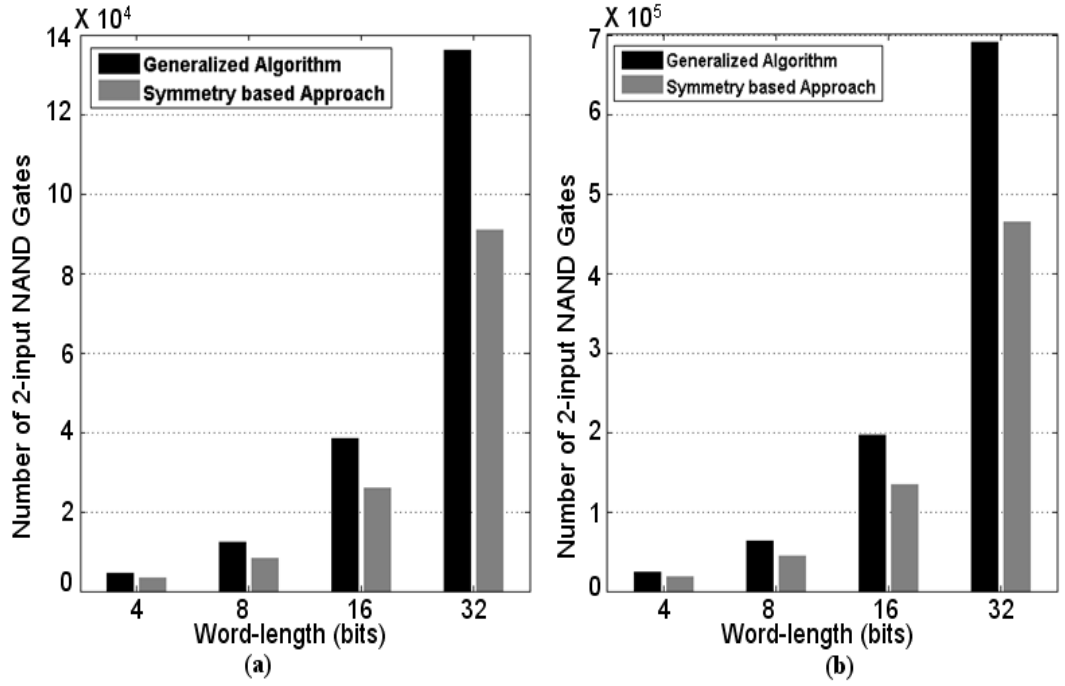


FIGURE 4.14: Comparative Post synthesis NAND gate complexity results for different Word-lengths using proposed Generalized Algorithm and Symmetry-based Approach for (a) $4D$ cross product, (b) $5D$ cross product.

on the proposed generalized Algorithm (shown as Approach-A in Table 4.2). Therefore, these results shown in Table 4.2 also validate the numerical results on hardware saving of the Approach-B over Approach-A as derived in the last section and also provide a proper justification of Fig. 4.9.

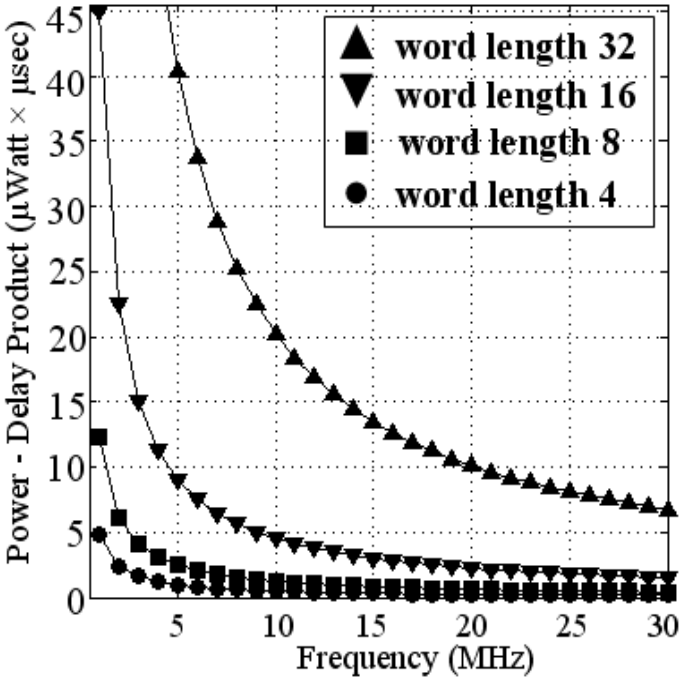


FIGURE 4.15: Variation of Power - Delay Product with frequency and word-length for $3D$ cross product architecture based on the generalized approach.

All experimental results presented above are simply for proof of the concept discussed in the previous sections. However we believe, since power consumption is directly related to frequency, it would be a good idea to use Power-Delay Product (PDP) as the metric and show its variation considering a reasonable range of frequency of operation for the generalized as well as symmetry based approaches. Keeping that in mind, we plot PDP with frequency ranging from 1 to 30 MHz for $3D$, $4D$ and $5D$ cases in Fig. 4.15, 4.16 and 4.17 respectively. It is important to note that the PDP values as appeared in these figures are obtained considering the worst case delays derived in (4.44) and (4.47) for generalized and symmetry based approach respectively. To show the effectiveness of the symmetry based approach over the generalized approach in terms of PDP, Fig. 4.16 and Fig. 4.17 show the variation of PDP for both of these two approaches. Since lower PDP signifies faster and efficient design, comparing Fig. 4.16(a) with Fig. 4.16(b) for $4D$ case and Fig. 4.17(a) with Fig. 4.17(b) for $5D$ case, it can easily be observed that for a fixed word-length and at a particular frequency the PDP value for the symmetry based approach is much lower than the corresponding generalized approach. Thus the efficiency of the symmetry based approach over the generalized one can once again be verified.

To the best of our knowledge, due to the unavailability of any cross product computation

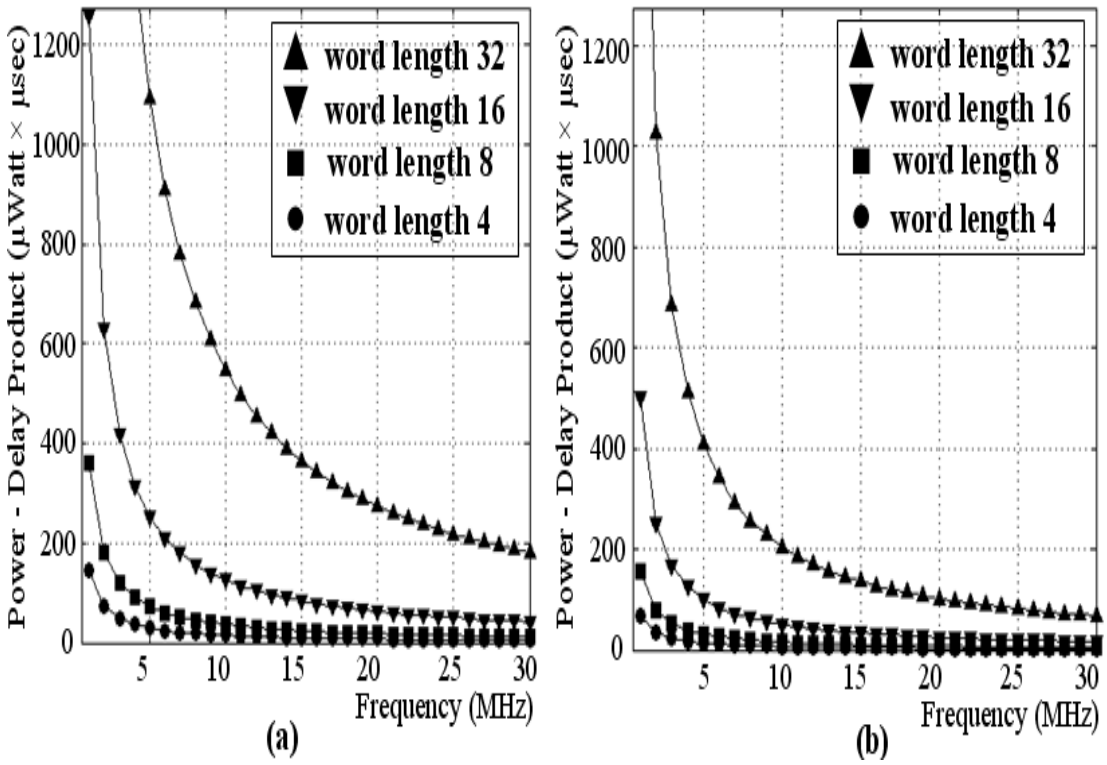


FIGURE 4.16: Variation of Power - Delay Product with frequency and word-length for $4D$ cross product. (a) Generalized Approach, (b) Symmetry based Approach.

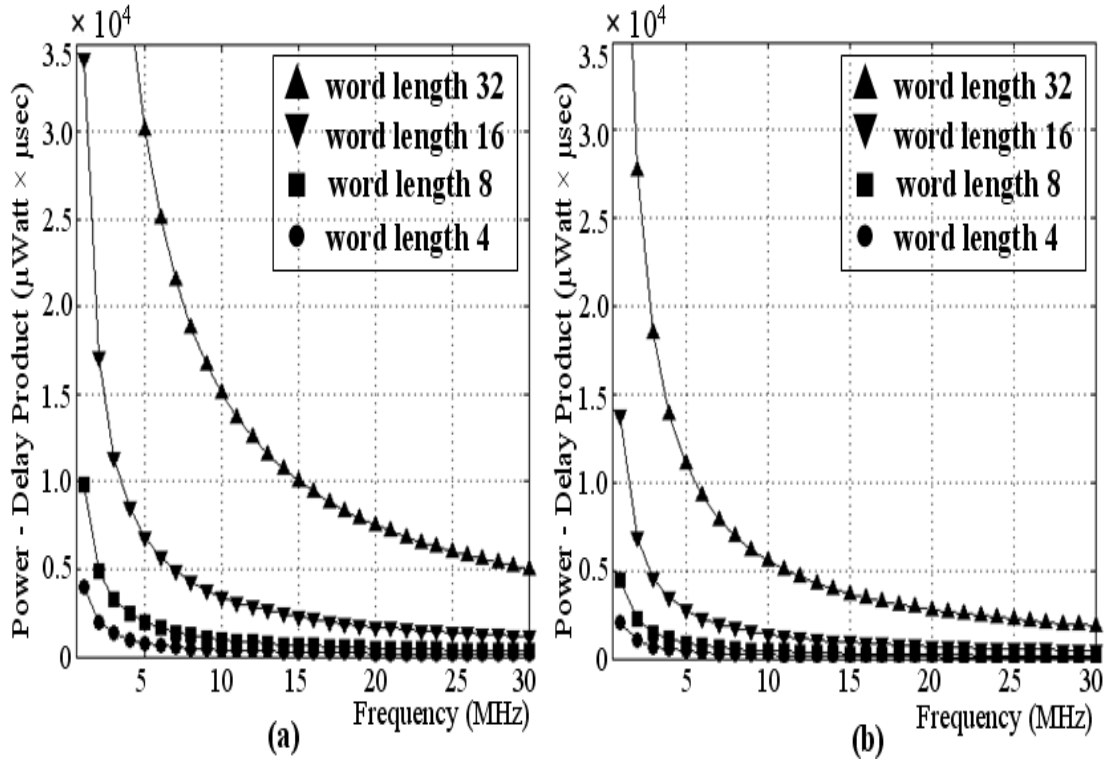


FIGURE 4.17: Variation of Power - Delay Product with frequency and word-length for $5D$ cross product. (a) Generalized Approach, (b) Symmetry based Approach.

architecture, we are unable to compare the results of our proposed architectures with any other published results.

TABLE 4.2: Post-synthesis Power and Area results of $4D$ and $5D$ cross-product computation architectures based on the Structural Approach (App-A) and Symmetry based Approach (App-B) with variable word-lengths (W bits)

D	W	Power (μW)		Area (mm^2)		NAND Gate Complexity	
		App - A	App - B	App - A	App -B	App - A	App -B
4	4	2.5665	1.7777	0.0278	0.0196	4594	3239
	8	6.4506	4.1313	0.0738	0.0503	12194	8311
	16	22.4782	13.1685	0.2332	0.1570	38533	25942
	32	97.7398	54.0180	0.8229	0.5505	135971	90962
5	4	13.4830	10.0228	0.1475	0.1122	24372	18539
	8	33.4430	22.0020	0.3819	0.2705	63103	44696
	16	115.6822	67.3969	1.1893	0.8143	196514	134551
	32	509.8800	270.1250	4.1785	2.8120	690433	464640

4.6 Concluding Remarks

In this chapter a generalized algorithm comprising of several mapping schemes for computing nD vector cross-product using $3D$ cross-product computation unit as the fundamental block is proposed. The working principle of the proposed algorithm has been demonstrated with an illustration of data-flow graph and an architecture has been discussed based on this algorithm. A new symmetry based approach was introduced for hardware reduction and has been proved very effective over the proposed generalized architecture in terms of low hardware complexity and improved faster operation. Worst case precision error has been analyzed and a optimum data-bus width selection criterion to achieve a pre-specified output precision has also been formulated. To give an insight into the VLSI implementation of the proposed schemes, post-synthesis power and area results are provided with different word-lengths for 3, 4 and 5 dimensional cases under a specific technology. Moreover, it has been pointed out that the proposed algorithm can enable low-complexity hardware design for computation intensive ICA algorithms which will be discussed in the next chapter.

Chapter 5

FastICA Based on nD Cross Product

In the last chapter the concept of vector cross product has been introduced in the context of ICA and its corresponding n-Dimensional (nD) cross product computation architectures have been proposed. In this chapter that concept is utilized to propose a novel low complexity nD FastICA.

High algorithmic efficiency [117], [118] and better convergence speed [99], [75] has made FastICA an excellent candidate for application in BSS problems where separation of a signal of interest from the mixture of multiple signals is of paramount importance. However it is computationally intensive algorithm and therefore its hardware implementation is deemed to be constrained by large silicon area and power consumption [1].

As for an example, it can be envisaged that the sensors are placed over the patients' body within remote healthcare monitoring environment for capturing the vital signs. These physiological signals are always mixed with artifacts and noise. If a on-board processing unit is envisaged within the sensors which helps separating these artifacts and noise from the wanted physiological signal, then FastICA will find its potential application here because of its algorithmic efficiency. However, sensors used in such remote health monitoring environment, need to be tiny, unobtrusive and need to be run in battery backup. So technically, although FastICA is very effective, because of its computation intensive nature, under such resource constrained environment, may not suitable for its hardware implementation in its present form. Therefore, any modification in this algorithm that results in reduction of arithmetic computations may thus be beneficial for its implementation in hardware and from real-time operation perspective.

FastICA, needs to solve an adaptive iterative equation involving several computationally

[1]The contents of Section 5.3.4 and Section 5.3.5 appeared partly in "Algorithm and Architecture for N-D Cross product Computation" by Acharyya *et. al.* in IEEE Trans. Signal Processing. See Appendix-A for further detail.

intensive arithmetic operations including square root evaluation, multiplications and divisions. Being iterative in nature, these operations have to be performed repeatedly until each estimated vector of the unmixing matrix (as detailed in Chapter 3) falls within a pre-defined convergence basin. Since in n-dimensional (n-D) FastICA the unmixing matrix consists of n vectors, conventional n-D FastICA needs n such iteration stages for computation of these n vectors. Unfortunately, to the best of our knowledge, there exists no algorithm which can reduce this significant arithmetic computations.

In this context the concept of the generalized n-D vector cross product, as introduced in the last chapter, can be brought in. Despite the presence of such concept in mathematics for a long time [125] - [131], due to lack of known engineering applications in general purpose computer arithmetic and signal processing, this has not received much attention. In this chapter the concept of generalized n-D vector cross product is utilized for reformulating n-D FastICA algorithm and show that such approach eliminates the necessity of the n^{th} iteration stage completely from the conventional n-D FastICA without sacrificing the algorithmic efficiency and numerical accuracy. Consequently number of arithmetic operations can be reduced significantly which is beneficial for hardware implementation from silicon area and power consumption perspectives.

The rest of this chapter is organized as follows - Section 5.1 proposes a novel low complexity predictive $2D$ FastICA algorithm as the motivation behind this research, Section 5.2 describes the proposed cross-product based low complexity nD FastICA algorithm, Section 5.3 analyzes the performance of the proposed algorithms in terms of the hardware complexity and delay, Section 5.4 validates the proposed algorithm and finally Section 5.5 concludes the discussion.

5.1 Motivation: Proposed Predictive $2D$ FastICA Algorithm

In this section, firstly conventional $2D$ FastICA algorithm is explored further from the perspective of its internal geometry. Then, it will be shown that the conventional scheme, in its present form, has some redundancies which, if eliminated, may give rise to a potential low complexity solution from its hardware implementation point of view. Subsequently, a novel low complexity $2D$ FastICA algorithm is proposed which will be called as - "Predictive $2D$ FastICA Algorithm".

Considering the conventional 2-D FastICA (i.e. $n = 2$) the unmixing matrix B can be written as:

$$\begin{aligned} B &= [\underline{\mathbf{w}}_{\mathbf{1}}\ \underline{\mathbf{w}}_{\mathbf{2}}] \\ &= \begin{bmatrix} \underline{w}_{1,1} & \underline{w}_{2,1} \\ \underline{w}_{1,2} & \underline{w}_{2,2} \end{bmatrix} \end{aligned} \tag{5.1}$$

where, the i^{th} unit-norm vector of B to be estimated is denoted by $\underline{\mathbf{w_i}} \in \mathbb{R}^2$ and $\forall i = 1, 2$ and $\underline{w}_{i,j}$ denotes the j^{th} component of $\underline{\mathbf{w_i}}$ where $j = 1, 2;\ \forall i$. Consider $\underline{\mathbf{w_1}}$ is already determined following the conventional FastICA using (3.7) and (3.10).

Now, considering any arbitrary vector $\mathbf{u} \in \mathbb{R}^2$ and denoting $\mathbf{u} = [u_1\ u_2]^T$ and $\underline{\mathbf{w_1}} = [\underline{w}_{1,1}\ \underline{w}_{1,2}]^T$ and the projection of $\mathbf{u}$ on $\underline{\mathbf{w_1}}$ by $\mathbf{Proj}_{\underline{\mathbf{w_1}}}{}^{\mathbf{u}}$, the following equation holds:

$$\begin{aligned} \mathbf{Proj}_{\underline{\mathbf{w_1}}}{}^{\mathbf{u}} &= \langle \mathbf{u}, \underline{\mathbf{w_1}} \rangle \underline{\mathbf{w_1}} \\ &= \begin{bmatrix} u_1(\underline{w}_{1,1})^2 + u_2 \underline{w}_{1,1} \underline{w}_{1,2} \\ u_1 \underline{w}_{1,1} \underline{w}_{1,2} + u_2(\underline{w}_{1,2})^2 \end{bmatrix} \end{aligned} \tag{5.2}$$

where $\langle \cdot\,, \cdot \rangle$ represents the vector scalar product or dot product operation. Using Gram-Schmidt orthogonalization (as shown in (3.9)) in (5.2) and considering the usual Euclidean norm for normalization, the following equation holds:

$$\begin{aligned} \mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp} &= \mathbf{u} - \mathbf{Proj}_{\underline{\mathbf{w_1}}}{}^{\mathbf{u}} \\ &= \begin{bmatrix} u_1 - (u_1(\underline{w}_{1,1})^2 + u_2 \underline{w}_{1,1} \underline{w}_{1,2}) \\ u_2 - (u_1 \underline{w}_{1,1} \underline{w}_{1,2} + u_2(\underline{w}_{1,2})^2) \end{bmatrix} \\ &= \begin{bmatrix} u_1 - \frac{u_1 {w_{1,1}}^2 + u_2 w_{1,1} w_{1,2}}{{w_{1,1}}^2 + {w_{1,2}}^2} \\ u_2 - \frac{u_1 w_{1,1} w_{1,2} + u_2 {w_{1,2}}^2}{{w_{1,1}}^2 + {w_{1,2}}^2} \end{bmatrix} \\ &= \begin{bmatrix} u_1(\underline{w}_{1,2})^2 - u_2 \underline{w}_{1,1} \underline{w}_{1,2} \\ u_2(\underline{w}_{1,1})^2 - u_1 \underline{w}_{1,1} \underline{w}_{1,2} \end{bmatrix} \end{aligned} \tag{5.3}$$

where the orthogonal component of $\mathbf{u}$ on $\underline{\mathbf{w_1}}$ is denoted by $\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp}$. Transforming (5.3) from Cartesian to Polar co-ordinate system, $\underline{\mathbf{w_1}}$ can be defined as:

$$\underline{\mathbf{w_1}} = [\underline{w}_{1,1}\ \underline{w}_{1,2}]^T = [\cos\theta\ \ \sin\theta]^T \tag{5.4}$$

where, polar angle $\theta = \tan^{-1}(\underline{w}_{1,2}/\underline{w}_{1,1})$. Since $\underline{\mathbf{w_1}}$ is a vector with unit norm, the magnitude of the radial vector is one and thus does not appear in (5.4). Using (5.4) in (5.3), $\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp}$ can be given as:

$$\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp} = \begin{bmatrix} u_1 \sin^2\theta - u_2 \sin\theta\cos\theta \\ u_2 \cos^2\theta - u_1 \sin\theta\cos\theta \end{bmatrix} \tag{5.5}$$

The norm of $\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp}$ (i.e. $|\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp}|$) can be determined from (5.5) as:

$$|\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp}| = \pm(u_1 \sin\theta - u_2 \cos\theta) \tag{5.6}$$

It is to be noted from (5.6) that $|\mathbf{u}_{\underline{\mathbf{w_1}}}{}^{\perp}|$ can take any one out of two probable values. Normalizing (5.5) by (5.6) and using (5.4), the new normalized vector $\underline{\mathbf{u}}_{\underline{\mathbf{w_1}}}{}^{\perp}$ can be given

as:

$$\underline{\mathbf{u}}_{\underline{\mathbf{w}}_1}{}^{\perp} = \begin{bmatrix} \pm\sin\theta \\ \mp\cos\theta \end{bmatrix} = \begin{bmatrix} \pm\underline{w}_{1,2} \\ \mp\underline{w}_{1,1} \end{bmatrix} \tag{5.7}$$

It is to be noted from (5.7) that there is only one possible orthonormal vector (with an associated phase indeterminacy) with respect to a pre-determined unit-norm vector ($\underline{\mathbf{w}}_1$) in 2-D vector space and this orthonormal vector ($\underline{\mathbf{u}}_{\underline{\mathbf{w}}_1}{}^{\perp}$) can be derived straightway in terms of $\underline{\mathbf{w}}_1$ *without the necessity of any further computation.*

From Section 3.3, it can be seen that since all calculations involved in FastICA Iteration are carried out in the whitened space, the necessity of (3.10) and (3.9) after (3.7) is therefore to ensure all the estimated columns of unmixing matrix B are mutually orthonormal. Hence for the 2-D FastICA case under consideration, without any loss of generality it can also be said that within B, $\underline{\mathbf{w}}_1$ and $\underline{\mathbf{w}}_2$ are mutually orthonormal. It has already been proved in (5.7) that in a 2-D vector space if $\underline{\mathbf{w}}_1$ is already known, then there can only be one vector possible which is orthonormal to $\underline{\mathbf{w}}_1$. Combining all these arguments together, a *predictive algorithm* can be framed for 2-D FastICA as follows:

Theorem 1. **Predictive Algorithm -** If the first column (i.e. the first vector $\underline{\mathbf{w}}_1$) of the unmixing matrix B (as shown in (5.1)) of 2-D FastICA Iteration stage is already determined, then the second column (i.e. the second vector $\underline{\mathbf{w}}_2$) can directly be evaluated from $\underline{\mathbf{w}}_1$ without any further computation as follows:

$$\underline{\mathbf{w}}_2 = \begin{bmatrix} \underline{w}_{2,1} \\ \underline{w}_{2,2} \end{bmatrix} = \begin{bmatrix} \pm\underline{w}_{1,2} \\ \mp\underline{w}_{1,1} \end{bmatrix} \tag{5.8}$$

Proof. Proof follows after replacing $\underline{\mathbf{u}}_{\underline{\mathbf{w}}_1}{}^{\perp}$ by $\underline{\mathbf{w}}_2$ in (5.7) and using fore-mentioned arguments. ■

Fig. 5.1 shows the overall flow of the modified FastICA Iteration stage of 2-D FastICA algorithm based on the proposed predictive approach in pseudo-code format.

Theorem 1 leads to the following inferences:

Conjecture 1.1. *Low Computational Complexity* (in terms of arithmetic operations)- One conventional FastICA Iterative stage, as outlined in Section 3.3, needs solving an adaptive equation (see (3.7)), orthogonalization (see (3.9)), normalization (see (3.10)) for estimating $\underline{\mathbf{w}}_2$ and these three steps are repeated until convergence is achieved. Comparing these with Fig. 5.1, it can be seen that the proposed algorithm does not need any of these three steps for $\underline{\mathbf{w}}_2$ computation and thereby removes significant amount of complex arithmetic operations such as division, square rooting and multiplications. From architectural point of view, such reduction in overall arithmetic computations makes the proposed algorithm highly efficient for resource constrained applications which will be analyzed in Section 5.3.

(i) **Initialization:** Choose an initial value of unit norm of w_1 randomly.

(ii) **Adaptive Learning Equation:**

$$\mathbf{w_1} \leftarrow \mathcal{E}\{Z\dot{g}(\underline{\mathbf{w}_1}^T Z)\} - \mathcal{E}\{\ddot{g}(\underline{\mathbf{w}_1}^T Z)\}\underline{\mathbf{w}_1}$$

where $\dot{g}$ and $\ddot{g}$ are the first and second derivative of contrast function g respectively.

(iii) **Normalisation:**

$$\underline{\mathbf{w}_1} \leftarrow \mathbf{w_1}/\|\mathbf{w_1}\|$$

(iv) **Convergence Check:** If $\underline{\mathbf{w}_1}$ has not converged, go back to step (ii), if converged go to next step.

(v) **Proposed Predictive Algorithm:** $\underline{\mathbf{w}_2}$ is computed following *Theorem* 1 as:

$$\underline{\mathbf{w}_2} = \begin{bmatrix} \underline{w}_{2,1} \\ \underline{w}_{2,2} \end{bmatrix} = \begin{bmatrix} \pm\underline{w}_{1,2} \\ \mp\underline{w}_{1,1} \end{bmatrix}$$

FIGURE 5.1: Pseudocode for the proposed Predictive Algorithm based 2-D FastICA.

Conjecture 1.2. *Low Numerical Error* - Following *Conjecture* 1.1, it is imperative that reduction in total number of arithmetic operation leads to less numerical error in fixed-point arithmetic based architecture.

Conjecture 1.3. *Symmetric Approach* - It can be seen from (5.8) that $\underline{\mathbf{w}_2}$ can be computed from $\underline{\mathbf{w}_1}$ by simple complement and hardware wiring operations without the necessity of any additional arithmetic operation. Therefore $\underline{\mathbf{w}_1}$ and $\underline{\mathbf{w}_2}$ can be obtained in parallel and thus the proposed algorithm transforms the deflationary scheme of estimating ICs into symmetric one.

Conjecture 1.4. *Phase or Sign Indeterminacy* - Due to the absence of any *apriori* information of the channel and the source, phase or sign indeterminacy is the typical problem in any ICA and the proposed algorithm based 2-D FastICA is also not an exception. As can be seen from (5.6) - (5.8), $\underline{\mathbf{w}_2}$ can be estimated up to a sign change probability. However, such sign inversion does not change the morphology of the signal and preserves the total energy, hence it does not worsen the effectiveness of the algorithm.

Fig. 5.2 shows the geometrical interpretation of the aforementioned predictive algorithm. From (5.8) it can be observed that the absolute value of the abscissa and ordinate of the predicted $\underline{\mathbf{w}_2}$ are same as the ordinate and the abscissa of the pre-determined $\underline{\mathbf{w}_1}$ respectively. Therefore exploiting the circular symmetry of the 2-D coordinate system

it can be said that if $\underline{\mathbf{w_1}}$ makes an angle θ with the X-axis in 2-D co-ordinate plane (see Fig. 5.2(a)), then given the condition that the two vectors will be *mutually orthonormal*, the second vector $\underline{\mathbf{w_2}}$ will also make an angle θ with the Y-axis in the same plane as shown in Fig. 5.2(b). Since both of these two vectors are of unit norm, they can be considered as the radii of the unit circle as shown in Fig. 5.2.

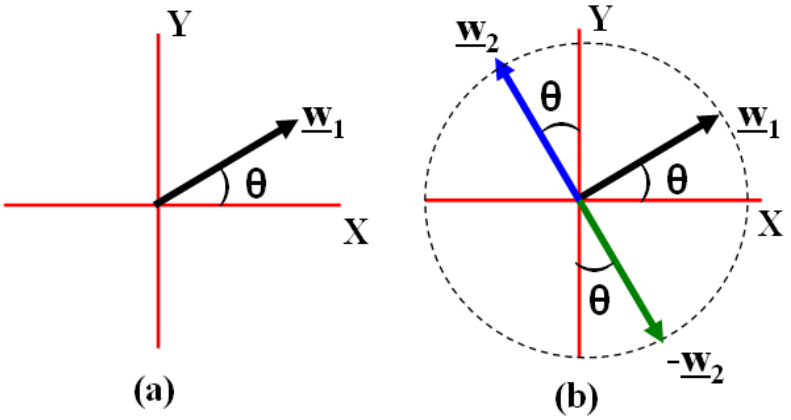


FIGURE 5.2: Geometrical interpretation for the predictive 2-D FastICA. (a) Predetermined $\underline{\mathbf{w_1}}$ is making an angle θ with the X-axis in 2-D plane. (b) To maintain the condition of *orthonormality*, second vector $\underline{\mathbf{w_2}}$ will also make an angle θ with the Y-axis in the same plane.

The associated sign or phase indeterminacy which was mentioned after (5.7), can also be seen in Fig. 5.2(b). The predicted vector can make angle θ with positive Y axis or negative Y axis and thus $+\underline{\mathbf{w_2}}$ or $-\underline{\mathbf{w_2}}$ - both are possible. Therefore it is associated with sign or phase indeterminacy problem and detailed discussion on this issue is deferred until the next section.

This aforementioned predictive algorithm along with the geometric interpretation shown in Fig. 5.2 motivate us to explore how the same concept can be extended to higher dimension where $n > 2$. In this context, the concept of vector cross product will be utilized and subsequently introduce a generalized n-D FastICA algorithm based on this concept in the next section.

5.2 Proposed Vector Cross Product Based nD FastICA Algorithm

To present the idea of extending the concept of predictive 2-D FastICA described in the last section in the higher dimension (where $n > 2$) in a simple fashion initially the geometrical view is provided and then later in this section the mathematical model will be formulated.

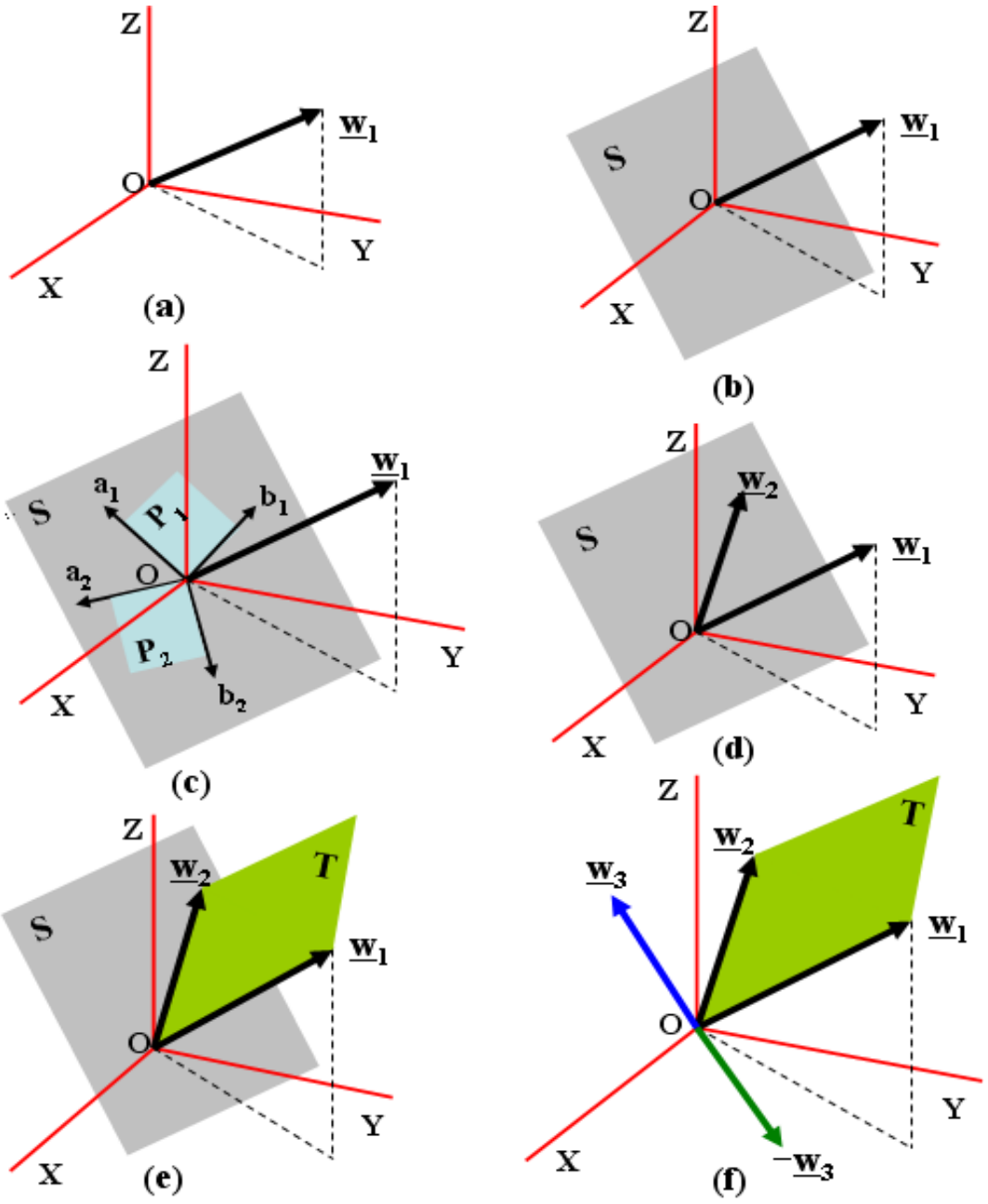


FIGURE 5.3: Geometrical interpretation of the step-by-step development of the $3D$ FastICA based on the concept of vector cross product. (a) $\underline{\mathbf{w}}_1$ is pre-determined in 3-D space. (b) $\underline{\mathbf{w}}_1$ can be assumed to be orthonormal to a surface S. (c) Several combinations of a pair of mutually orthonormal vectors in S which are orthonormal to $\underline{\mathbf{w}}_1$ as well. (d) Consider $\underline{\mathbf{w}}_2$ is also known. (e) A 2-D plane T can be considered to be spanned by $\underline{\mathbf{w}}_1$ and $\underline{\mathbf{w}}_2$. (f) $\underline{\mathbf{w}}_3$ (or $-\underline{\mathbf{w}}_3$) can be obtained by taking cross-product between $\underline{\mathbf{w}}_1$ and $\underline{\mathbf{w}}_2$ which is orthonormal to surface T.

Considering first the $3D$ FastICA (i.e. $n = 3$), the unmixing matrix B can be given as:

$$\begin{aligned} B &= [\underline{\mathbf{w_1}}\ \underline{\mathbf{w_2}}\ \underline{\mathbf{w_3}}] \\ &= \begin{bmatrix} \underline{w}_{1,1} & \underline{w}_{2,1} & \underline{w}_{3,1} \\ \underline{w}_{1,2} & \underline{w}_{2,2} & \underline{w}_{3,2} \\ \underline{w}_{1,3} & \underline{w}_{2,3} & \underline{w}_{3,3} \end{bmatrix} \end{aligned} \tag{5.9}$$

where, the i^{th} unit-norm vector of B to be estimated is denoted by $\underline{\mathbf{w_i}} \in \mathbb{R}^3$ and $\forall i = 1, 2, 3$ and $\underline{w}_{i,j}$ denotes the j^{th} component of $\underline{\mathbf{w_i}}$ where $j = 1, 2, 3$; $\forall i$. Consider $\underline{\mathbf{w_1}}$ is already determined using (3.7) and (3.10). Consider Fig. 5.3 depicts one such 3-D coordinate space denoted by "$O-X-Y-Z$" where the orientation of the pre-determined vector $\underline{\mathbf{w_1}}$ is shown arbitrarily in Fig. 5.3(a).

Since within a 3-D space any vector can be considered as the normal to a surface, without any loss of generality $\underline{\mathbf{w_1}}$ can also be considered as the normal to an arbitrary surface as denoted by S in Fig. 5.3(b).

From the discussion of the Section 5.1, it is important to recall that in 3-D FastICA, B comprises of three mutually orthonormal vectors. Since one vector i.e $\underline{\mathbf{w_1}}$ is already known and it is orthonormal to the surface S, therefore it is obvious that other two vectors i.e. $\underline{\mathbf{w_2}}$ and $\underline{\mathbf{w_3}}$ also lie in S.

Since any combination of a pair of mutually orthonormal vectors in S is also orthonormal to $\underline{\mathbf{w_1}}$, therefore if a unique combination of a pair of such mutually orthonormal vectors can be found in plane S, then the problem under consideration will be solved. But this raises following two issues.

Since from (5.8) it can be seen that knowledge of one vector within B in 2-D FastICA is sufficient to predict the orientation of the other vector, *(i)* in higher dimensional space (i.e. where $n > 2$), is the knowledge of one vector $\underline{\mathbf{w_1}}$ sufficient to predict $\underline{\mathbf{w_2}}, \underline{\mathbf{w_3}}, \cdots, \underline{\mathbf{w_n}}$? or *(ii)* is it necessary to know $\underline{\mathbf{w_1}}, \underline{\mathbf{w_2}}, \cdots, \underline{\mathbf{w_{n-1}}}$ to predict $\underline{\mathbf{w_n}}$?

To address the first issue, consider Fig. 5.3(c). It shows two different combinations of a pair of mutually orthonormal vectors on S - $\mathbf{a_1}, \mathbf{b_1}$ and $\mathbf{a_2}, \mathbf{b_2}$ which define space P_1 and P_2 respectively where both P_1 and $P_2 \subseteq S$. It can also be seen from Fig. 5.3(c) that both vectors within any of these two combinations are mutually orthonormal to $\underline{\mathbf{w_1}}$.

In more generalized way, it can be said that there exists several such combinations of a pair of mutually orthonormal vectors $\mathbf{a_k}$ and $\mathbf{b_k}$ in S where k is very large. Since it is associated with such large degree of freedom, it is not possible to find a unique combination of a pair of mutually orthonormal vectors in S only based on the knowledge of one vector $\underline{\mathbf{w_1}}$. Thus the first issue is resolved and at this point it is clear that knowledge of one vector is not enough to predict the rest of the vectors in B.

Now consider that the second vector $\underline{\mathbf{w_2}}$ is also determined using (3.7), (3.10) and (3.9)

repetitively and from the discussion above it is evident that $\underline{\mathbf{w_2}}$ lies in S as shown in Fig. 5.3(d). A surface T can be considered to be spanned by $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$ as shown in Fig. 5.3(e). Therefore S and T are the two orthogonal planes where $\underline{\mathbf{w_2}}$ lies at their intersection. It was already discussed above that $\underline{\mathbf{w_3}}$ lies in S, but its orientation is not known so far. However it is known that $\underline{\mathbf{w_3}}$ is orthonormal to both $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$. Therefore it has to be orthonormal to the surface T which is defined by $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$ as shown in Fig. 5.3(f).

The concept of 3-D vector cross product can be introduced at this point since this a mathematical tool used to derive the vector which is the normal to a surface spanned by two known vectors [120], [121]. Since based on our assumptions $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$ of B are already known and it has already been shown that $\underline{\mathbf{w_3}}$ is orthonormal to the surface T defined by $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$, therefore $\underline{\mathbf{w_3}}$ can be computed simply by considering the cross product operation between $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$ using (4.3), as follows:

$$\begin{aligned} \underline{\mathbf{w_3}} &= [\underline{w}_{3,1}\ \underline{w}_{3,2}\ \underline{w}_{3,3}] \\ &= \begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \\ \underline{w}_{1,1} & \underline{w}_{1,2} & \underline{w}_{1,3} \\ \underline{w}_{2,1} & \underline{w}_{2,2} & \underline{w}_{2,3} \end{vmatrix} \end{aligned} \tag{5.10}$$

Using the cofactor expansion method following (4.5) and (4.7), (5.10) can be expressed as:

$$\begin{aligned} \underline{\mathbf{w_3}} &= \sum_{k=1}^{3} e_k W_{3,1k} \\ &= \sum_{k=1}^{3} \mathbf{e_k} (-1)^{1+k} |M_{1k,\underline{\mathbf{w_3}}}| \end{aligned} \tag{5.11}$$

where $W_{3,1k}$ and $M_{1k,\underline{\mathbf{w_3}}}$ are the $(1k)^{th}$ **cofactor** of 3×3 matrix shown in (5.10) and the corresponding **Minor** respectively. Thus the second issue mentioned above is resolved and it means that in the case of 3-D FastICA, two vectors are needed to predict the third vector based on the concept of vector cross product.

Before proceeding further, at this important juncture, it is necessary to recall the validity and generalization of the concept of cross product in n-D vector space from the last chapter where it was mentioned that generalization of r-fold vector cross product in nD space exists precisely in the following cases:

(1) $r = 1$ and n is even.

(2) $r = n - 1$ and n is arbitrary.

(3) $r = 2$ and $n = 3$ (i.e. common cross product) and 7.

(4) $r = 3$ and $n = 8$.

Since *Theorem* 1 stated clearly that for 2-D FastICA the second vector of the unmixing matrix can be predicted straightway from the first one and the detailed explanation of Fig. 5.3 established the fact that for 3-D FastICA, the third vector of the unmixing matrix is simply the cross product of the first two vectors, following the same analogy and using the case-2 ($r = n - 1$ and n is arbitrary) among the four afore-mentioned cases, the same concept can be extended further for n-D FastICA (where $n > 3$) based on the concept of the generalized n-D vector cross product as follows.

Theorem 2. **Vector Cross-Product based n-D FastICA -** If $(n-1)$ columns (i.e. $(n-1)$ vectors $\underline{\mathbf{w}}_{\mathbf{1}}, \underline{\mathbf{w}}_{\mathbf{2}} \ \cdots \ \underline{\mathbf{w}}_{(\mathbf{n-1})}$) of the unmixing matrix B for n-D FastICA Iteration stage are already determined, then the n^{th} column (i.e. the n^{th} vector $\underline{\mathbf{w}}_{\mathbf{n}}$) is simply the vector cross product of the already determined $(n-1)$ vectors as follows:

$$\begin{aligned}\underline{\mathbf{w}}_{\mathbf{n}} &= \begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} & \cdots & \mathbf{e_n} \\ \underline{w}_{1,1} & \underline{w}_{1,2} & \underline{w}_{1,3} & \cdots & \underline{w}_{1,n} \\ \underline{w}_{2,1} & \underline{w}_{2,2} & \underline{w}_{2,3} & \cdots & \underline{w}_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \underline{w}_{n-1,1} & \underline{w}_{n-1,2} & \underline{w}_{n-1,3} & \cdots & \underline{w}_{n-1,n} \end{vmatrix} \\ &= \sum_{k=1}^{n} e_k W_{n,1k} = \sum_{k=1}^{n} \mathbf{e_k} (-1)^{1+k} |M_{1k,\underline{\mathbf{w}}_{\mathbf{n}}}|\end{aligned} \tag{5.12}$$

where $W_{n,1k}$ and $M_{1k,\underline{\mathbf{w}}_{\mathbf{n}}}$ are the $(1k)^{th}$ **cofactor** of $n \times n$ matrix shown in (5.12) and the corresponding **Minor** respectively.

Proof. The concept follows straightway from the above discussion. The generalized expression for the n-D vector cross product follows from (4.4), (4.5) and (4.7). ■

Although at this point it can be argued that conventional orthogonalization techniques such as Gram-Schmidt or Householder methods can be used instead of the cross-product, it is important to understand that those methods also need the computation of the complete *iteration update* stage within the nD FastICA Iteration which involves significant amount of computations (as will be shown in the next Section), whereas cross-product based approach is able to remove that stage as well. Moreover for each data-frame computation of cross-product is required only once which removes the necessity of iterative computations of update, orthogonalization and normalization stages from the conventional FastICA.

Fig. 5.4 shows the overall flow of the proposed vector cross product based n-D FastICA algorithm in pseudocode format. It is to be noted from this figure that the n-D FastICA iteration stage is now needed to run only for $(n-1)$ times. Each such iteration includes computation of the adaptive learning equation several times iteratively before it converges to a pre-defined convergence range and each time the estimated intermediate vector has to be normalized and orthogonalized. The proposed algorithm removes completely the necessity of the n^{th} FastICA iteration stage from the n-D FastICA Iteration

(i) **Initialization:** (a) Choose $m = (n-1)$ where n is the number of vectors (i.e. number of independent components) within the unmixing matrix B and set a counter $k \leftarrow 1$; (b) Choose an initial value of unit norm of $\mathbf{w_k}$ randomly.

(ii) **Adaptive Learning Equation:**

$$\mathbf{w_k} \leftarrow \mathcal{E}\{Z\dot{g}(\mathbf{w_k}^T Z)\} - \mathcal{E}\{\ddot{g}(\mathbf{w_k}^T Z)\}\underline{\mathbf{w}}_\mathbf{k}$$

where $\dot{g}$ and $\ddot{g}$ are the first and second derivative of contrast function g respectively.

(iii) **Orthogonalization:**

$$\mathbf{w_k} \leftarrow \mathbf{w_k} - \sum_{j=1}^{k-1}(\mathbf{w_k}^T \mathbf{w_j})\mathbf{w_j}$$

(iv) **Normalization:**

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

(v) **Convergence Check:** If $\underline{\mathbf{w}}_\mathbf{k}$ has not converged, go back to step (ii), if converged go to next step.

(vi) Set $k \leftarrow k+1$. If $k \leq m$, go back to step (i)(b).

(vii) **Proposed Vector Cross Product based Algorithm:** $\underline{\mathbf{w}}_\mathbf{n}$ is computed following *Theorem* 2 as:

$$\underline{\mathbf{w}}_\mathbf{n} = \begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} & \cdots & \mathbf{e_n} \\ \underline{w}_{1,1} & \underline{w}_{1,2} & \underline{w}_{1,3} & \cdots & \underline{w}_{1,n} \\ \underline{w}_{2,1} & \underline{w}_{2,2} & \underline{w}_{2,3} & \cdots & \underline{w}_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \underline{w}_{n-1,1} & \underline{w}_{n-1,2} & \underline{w}_{n-1,3} & \cdots & \underline{w}_{n-1,n} \end{vmatrix}$$
$$= \sum_{k=1}^{n} e_k W_{n,1k} = \sum_{k=1}^{n} \mathbf{e_k}(-1)^{1+k}|M_{1k,\underline{\mathbf{w}}_\mathbf{n}}|$$

$\mathrm{e}_i \in \mathbb{R}^n, \forall i = 1,2,3,..,n$, are the unit vectors in n-D space; $W_{n,1k}$ and $M_{1k,\underline{\mathbf{w}}_\mathbf{n}}$ are the $(1k)^{th}$ **cofactor** of $n \times n$ matrix shown above and the corresponding **Minor** respectively.

FIGURE 5.4: Pseudocode for the Proposed Vector Cross Product based n-D low complexity FastICA Algorithm.

by introducing the concept of the generalized n-D vector cross-product.

Theorem 2 leads to the following inferences:

Conjecture 2.1. *Low Computational Complexity* - It is to be noted from (5.12) that the vector cross product based approach does not require computation of the FastICA adaptive iterative equation as shown in (3.7) several times until convergence is achieved and it does not need to go through the steps of normalizations and orthogonalizations for each such iteration as shown in (3.10) and (3.9) respectively. Thus the proposed algorithm removes one complete FastICA Iteration stage. However the proposed algorithm needs to compute n-D vector cross product. The detailed performance analysis will be carried out in Section 5.3 where it will be shown that the computation of n-D cross product operation needs significantly less arithmetic operations in comparison with the conventional n-D FastICA Iteration stage. Therefore cross-product based n-D FastICA is capable of reducing the computational complexity and improving operational speed-up significantly.

Conjecture 2.2. *Low Numerical Error* - Following *Conjecture* 2.1, it is imperative that reduction in total number of arithmetic operation leads to less numerical error in fixed-point arithmetic based architecture. However explicit numerical error analysis will not be carried out here.

Conjecture 2.3. *Phase or Sign Indeterminacy* - Since the vector cross product is an *anti-commutative* operation, it is associated with the inherent sign or phase indeterminacy. Considering cross product based 3-D FastICA once again, it can be seen from Fig. 5.3(f) that depending upon the chosen order of vectors $\underline{\mathbf{w_1}}$ and $\underline{\mathbf{w_2}}$, the cross product can be $\underline{\mathbf{w_3}}$ or $-\underline{\mathbf{w_3}}$ where *minus* sign indicates the estimated vector is 180^0 out-of-phase with $\underline{\mathbf{w_3}}$. In the same way, proposed n-D cross product based FastICA algorithm also suffers from such sign or phase indeterminacy problem. But due to the absence of any *apriori* information of the channel and the source, phase or sign indeterminacy is the typical problem in any ICA and therefore the proposed algorithm is also not an exception. However, such sign inversion does not change the morphology of the signal and preserves the total energy, hence it does not worsen the effectiveness of the algorithm.

5.3 Performance Analysis

This Section analyzes the architectural performance of the predictive $2D$ FastICA and generalized vector cross product based n-D FastICA algorithm proposed in Section 5.1 and 5.2 respectively in terms of hardware complexity (see subsection 5.3.2 and 5.3.4) and delay (see subsection 5.3.3 and 5.3.5). Although accurate estimation of these two parameters are difficult because of the basic differences of the adopted technology, assumptions used for the derivations and their implementation dependencies; nevertheless

our attempt here is to perform the approximate analysis and provide a comparative study with the available architectures.

5.3.1 Important Considerations

Throughout the analysis *Kurtosis* is considered as the contrast function and keep a generalized view of frame-length K, word-length b and the number of Iteration of Convergence (IOC) as P. To compute overall hardware complexity and delay, two metrics - transistor count and delay of a basic two input NAND gate are used respectively. However, since implementation of different arithmetic units can be done in different ways, first the generic hardware complexity and delay expression are derived here based on the number of arithmetic operations considering a flat unfolded architecture without resource sharing (used for hardware complexity computation in the following subsections) or parallel computing capability (used for delay analysis in the following subsections) following the approach presented in [122] irrespective of any specific implementation.

Then to compare the performances under a uniform platform, Ripple Carry Adder (RPA) and Subtractor (RPS), Conventional Array Multiplier (MUL), Non-restoring Array Divider (NAD), Square Rooter (SQRT) and basic combinatorial control-enabled 2's complementer structures are considered. Additionally it has been considered that the hardware complexity and delay of a squaring unit is 50% and 80% respectively of a conventional two-input multiplier [82].

It is important to note the following assumptions which will be used frequently to design the mathematical models for the hardware complexity and delay in the subsequent subsections.

Assumption 1.1. *(i)* One b-bit RCA and RPS needs b Full-Adders (FA) [83], *(ii)* one b by b MUL needs $b(b-2)$ FA, b Half-Adders (HA) and b^2 AND gates [84], *(iii)* one b-bit 2's complementer needs $(b-1)$ OR gates, b AND gates and b XOR gates [83], (iv) one b by b NAD consists of $0.5 \times b(3b-1)$ FA and $0.5 \times b(3b-1)$ XOR gates [85] and (v) one b-bit SQRT needs $0.125 \times (b+6)b$ FA and XOR gates [138].

Assumption 1.2. Transistor Count (TC) of *(i)* 1-bit FA is 24, *(ii)* 1-bit HA is 12, *(iii)* one 2-input XOR gate is 12, *(iv)* one 2-input AND gate is 6 and *(v)* one 2-input OR gate is 6 [62].

Assumption 1.3. Denoting the delay of a two-input NAND gate by $\triangle$, the delay of *(i)* b-bit two-operand RCA and RCS is $2b\triangle$ (τ_a and τ_s), *(ii)* b-by-b MUL is $8b\triangle$ ($= \tau_m$), *(iii)* b bit combinatorial 2's complementer is $(2b+3)\triangle$ ($= \tau_c$) [83], *(iv)* b-by-b NAD is $(3b+2)n\triangle$ ($= \tau_d$) [85] and *(v)* b bit SQRT is $(b+1)(2b+3)\triangle$ ($= \tau_{sqrt}$) [138].

Assumption 1.4. Since different arithmetic units can be implemented in different ways, it is considered the flat unfolded architecture without any resource-sharing capability

following [122]. Under these assumptions, the hardware complexity of the architectures in terms of the arithmetic modules can equivalently be expressed in terms of total arithmetic operations involved [122].

Assumption 1.5. Delay can be defined as the time between the first input to do the first computation and the last output from the final computation [51]. In this context of delay analysis, the worst-case delay of the n-D cross product architecture and one FastICA Iteration stage is derived. Moreover, it is assumed that there is no parallel computing capability of the architectures under consideration and therefore it is considered that only one arithmetic operation is permitted at one cycle [122]. No interconnect delay is considered. Under these assumptions the worst-case delay becomes the additive delay of the individual arithmetic module involved for overall computation [122].

TABLE 5.1: Comparison of the architectures in terms of number of arithmetic operations where "K" and "M" denotes the frame-length and iteration of convergence respectively. PP = Preprocessing, iteration = FastICA Iteration, Div = Division.

FastICA Architecture		Number of Arithmetic Operations involved				
PP	Iteration	Add	Subtract	Multiply	Div	Sq.Root
Shyu [76]	Shyu [76]	$9K+6KP$ $+2P$	$2K+8P$ $+4$	$11K+12KP$ $+19P+8$	$7+2K$ $+6P$	$3+2P$
Acharyya [137]	Acharyya [137]	$9K+6KP$ $+5P$	$2K+8P$ $+4$	$13K+11KP$ $+14P+5$	$2+2P$	$3+2P$
Shyu [76]	Proposed	$9K+3KP$	$2K+3P$ $+4$	$11K+6KP$ $+6P+8$	$7+2K$ $+3P$	$3+P$
Acharyya[137]	Proposed	$9K+3KP$ $+2P$	$2K+3P$ $+4$	$13K+6KP$ $+5P+5$	$2+P$	$3+P$

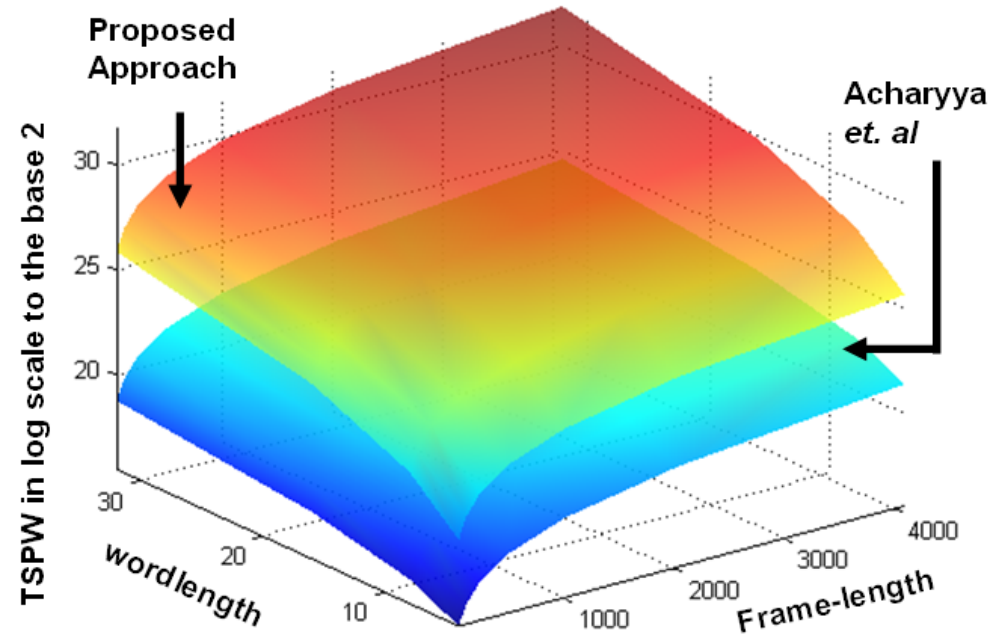
5.3.2 Hardware Complexity: Proposed Predictive $2D$ FastICA

Table 5.1 shows the total number of arithmetic operations involved in several architectures including the proposed predictive one. Recalling that the proposed predictive algorithm can be applied for the FastICA Iteration stage, last two rows of Table 5.1 include the FastICA Iteration architecture based on this proposed predictive algorithm along with the existing architectures for Preprocessing stage.

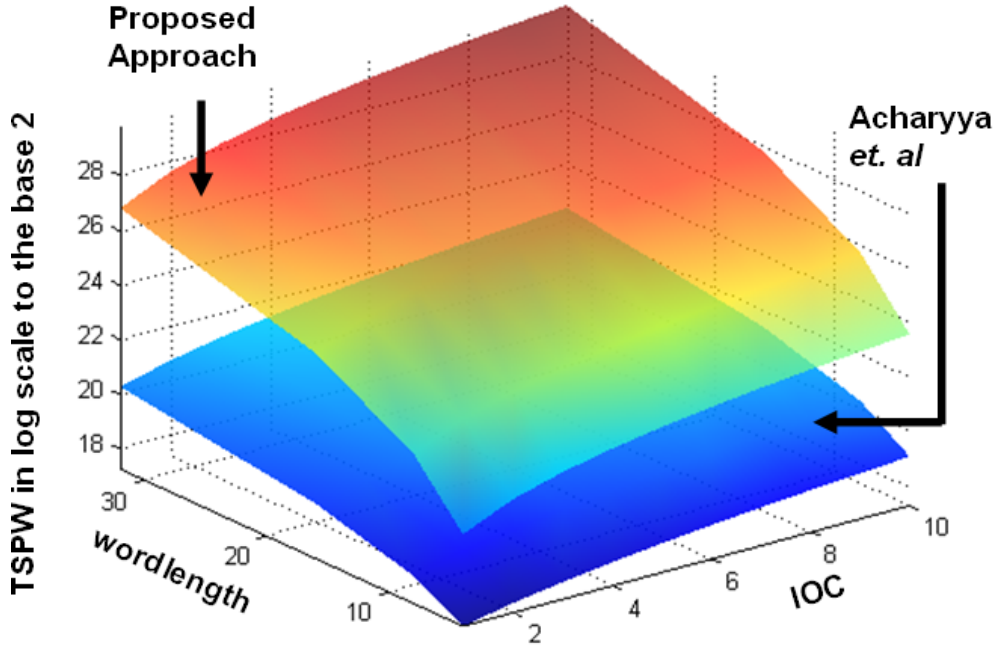
Since it was already shown in one of our research work [137] (and can also be formulated from Table 5.1) that the performance of [137] is better than [76] in terms of hardware complexity and delay, therefore for simplicity of the discussion and clarity of the simulation results, here all comparisons are done with [137]. Unless otherwise stated, now onwards the complete FastICA architecture presented in the last row of Table 5.1 (i.e. Preprocessing based on [137] + FastICA Iteration based on the proposed predictive algorithm) will be denoted as **Proposed Approach**.

It can be observed from Table 5.1 that **Proposed Approach** reduces $(3K+3)P$ number of additions, $5P$ number of subtractions, P number of divisions and square rooting,

and $(5K + 9)P$ multiplications (in which $(2K + 2)P$ are squaring operations) over the architecture presented in [137].



(a) TSPW vs Framelength vs wordlength with fixed IOC = 5



(b) TSPW vs IOC vs wordlength with fixed Framelength = 512

FIGURE 5.5: Comparative TSPW analysis between [137] and **Proposed Approach** (i.e. [137] for Preprocessing + proposed algorithm based architecture for FastICA Iteration). (a) Variation of TSPW with Frame-length and word-length for fixed IOC = 5, (b) Variation of TSPW with IOC and wordlength for fixed frame-length = 512.

Denoting the transistor counts for RCA, RPS, MUL, NAD and SQRT by TC_A, TC_S, TC_M, TC_D and TC_{SQ} respectively and using these TC_* in the fore-mentioned gain in

terms of arithmetic operations, we get $TS_A = P(3K+3) \times TC_A$, $TS_S = 5P \times TC_S$, $TS_M = P(5K+9) \times TC_M$, $TS_D = P \times TC_D$ and $TS_{SQ} = P \times TC_{SQ}$, where TS_* denotes Transistor Saving over the architecture proposed in [137] due to reduction in number of adders, subtractors, multipliers, dividers and square rooters. Summing over these TS_* and putting the corresponding values of TC_*, total Transistor Saving (TS) can be computed. TS can be normalized with respect to n and a metric - Transistor Saving Per Word-length (TSPW) can be formulated using the approach presented in [86]. Being the function of K, P and n, Fig. 5.5 shows the variation of TSPW for the **Proposed Approach** over [137] with respect to these parameters and compares with the TSPW as presented in [137].

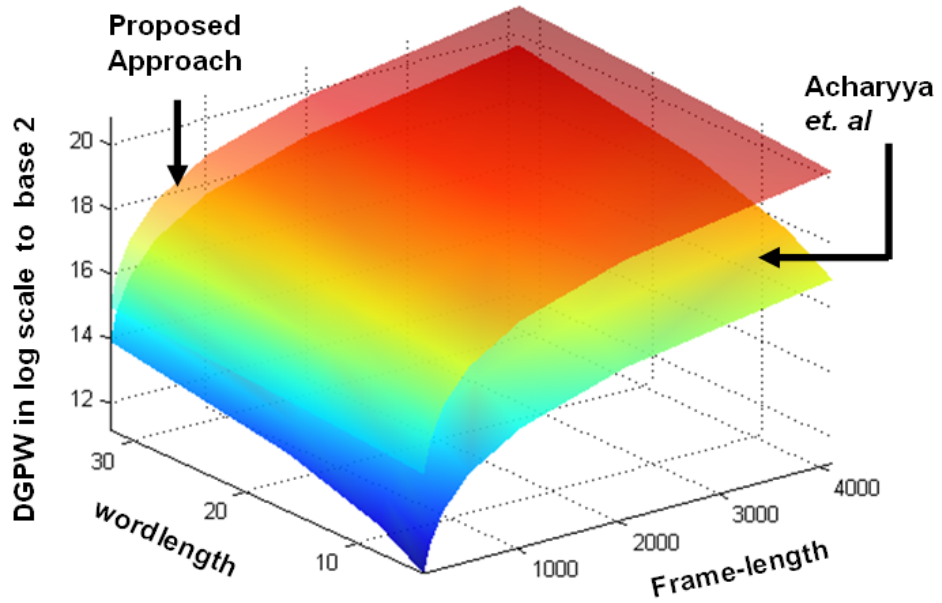
Fig. 5.5(a) plots the variation of TSPW of the **Proposed Approach** with respect to the frame-length K and word-length n keeping the IOC $P = 5$ where $64 \leq K \leq 4096$ and $4 \leq n \leq 32$. It can also be observed from Fig. 5.5(a) that TSPW for the **Proposed Approach** is significantly higher than that presented in [137] for higher K and n. Fig. 5.5(b) shows the variation of TSPW of the **Proposed Approach** with respect to P (where $1 \leq P \leq 10$) and n (where $4 \leq n \leq 32$) keeping $K = 512$ (as considered for the example architecture whose implementation results are presented in Section 5.4). It can be seen from Fig. 5.5(b) that with the increase of P and n TSPW of the **Proposed Approach** increases exponentially and leads to substantial amount of transistor saving over the approach presented in [137].

5.3.3 Delay Analysis: Proposed Predictive $2D$ FastICA

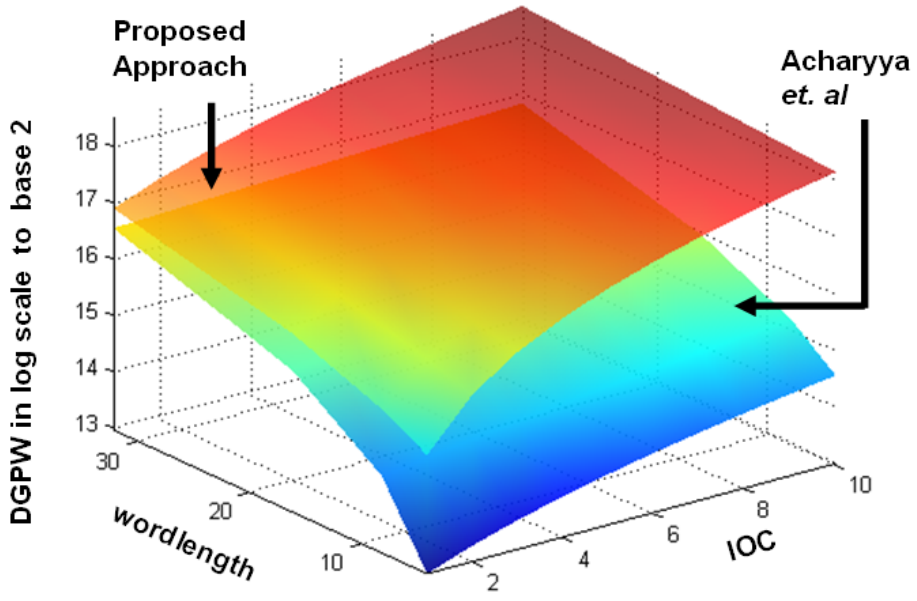
Proceeding the same way adopted to derive the hardware complexity, first the gain of the **Proposed Approach** over [137] in terms of arithmetic computations can be found out from Table 5.1.

It is to be recalled from last subsection that Table 5.1 shows the number of multiplication saved is $(5K+9)P$ in the **Proposed Approach** which includes $(2K+2)P$ number of squaring units as well. Therefore, total saving in terms of number of full multiplication becomes $(4K+8)P$. As mentioned in section 5.3.1, the delay of a squaring unit is almost 80% of a full multiplier, therefore the delay of a squaring unit can be translated into a full multiplier delay. Effective Delay Gain (DG) over the architecture described in [137] can be formulated following the approach presented in [137] combining τ_* along with the total savings in arithmetic computations. DG can then be normalized with respect to $n\triangle$ to produce a metric called Delay Gain Per Word-length (DGPW) following the same approach presented in [137]. Being a function of frame-length K, iteration of convergence P and wordlength n, Fig. 5.6(a) shows the variation of DGPW with K (where $64 \leq K \leq 4096$) and n ($4 \leq n \leq 32$) with $P = 5$ and Fig. 5.6(b) plots the variation of DGPW with P (where $1 \leq P \leq 10$) and n ($4 \leq N \leq 32$) keeping $K = 512$. DGPW as provided in [137] is reproduced here in Fig. 5.6(a) and (b) and compared with

the **Proposed Approach**. It can be observed from Fig. 5.6(a) that with the increase of K and n, DGPW for the proposed approach is higher than that presented in [137]. DGPW for the proposed approach, as shown in Fig. 5.6(b) is significantly greater than that shown in [137] for higher P.



(a) Normalized DGPW vs Framelength vs wordlength with fixed IOC = 5



(b) Normalized DGPW vs IOC vs wordlength with fixed Framelength = 512

FIGURE 5.6: Comparative DGPW analysis between [137] and **Proposed Approach**. (a) Variation of DGPW with Frame-length and wordlength for fixed IOC = 5, (b) Variation of DGPW with IOC and wordlength for fixed framelength = 512.

5.3.4 Hardware Complexity: Proposed Cross product based nD FastICA

Since it was mentioned in *Conjecture* 2.1 in Section 5.2 that one complete n-D FastICA Iteration stage is replaced by an n-D cross product computation operation, therefore to prove the low complexity of the proposed algorithm, the hardware complexity and delay of one n-D FastICA iteration stage as well as the n-D cross product computation operation are determined individually.

5.3.4.1 Complexity of n-D Cross product

Detailed discussion on the hardware complexity issues of nD Cross Product computations has been carried out in the last chapter. Recalling these derivations from the last chapter and using (4.39) in (4.35), ${}^{nD}HW_{str}$ can be expressed as:

$$ {}^{nD}HW_{str} = n \times {}^{(n-1)D}HW_{str} + n(n-2) \times {}^{2}A + \lfloor n/2 \rfloor \times C \tag{5.13} $$

Here the proposed "generalized approach" for cross product computation (as discussed in the last chapter) is considered for the purpose of comparison of hardware complexity. However, it is imperative to say, if the proposed "symmetry based approach" is used, further hardware complexity reduction is achievable.

5.3.4.2 Complexity of one n-D FastICA Iteration

As mentioned at the beginning of this subsection, considering *Kurtosis* (denoting a zero-mean random variable by γ, *Kurtosis* can be defined as: $Kurt(\gamma) = \mathcal{E}\{\gamma^4\} - 3(\mathcal{E}\{\gamma^2\})^2)$ as the *contrast function* (g) in (3.7), the adaptive iterative equation becomes:

$$ \mathbf{w_k} \leftarrow \mathcal{E}\{Z(\mathbf{w_k}^T Z)^3\} - 3\mathbf{w_k} \tag{5.14} $$

Using the notations introduced before and denoting the number of divider, squaring unit and square rooter by D, S_{qrng} and S_{qrt}, the hardware complexity of the adaptive equation of n-D FastICA (${}^{n}HW_{adp}$) as shown in (5.14) can be given as:

$$ \begin{aligned} {}^{n}HW_{adp} = {} & [(n+1)KP + nKP] \times M + [(n-1)KP + \\ & n(K-1)P + nP] \times {}^{2}A + nP \times S + KP \times S_{qrng} \end{aligned} \tag{5.15} $$

Following the same procedure, the hardware complexity for normalization (${}^{n}HW_{nrm}$) in n-D FastICA as shown in (3.10) can be derived as:

$$ {}^{n}HW_{nrm} = (n-1)P \times {}^{2}A + nP \times D + nP \times S_{qrng} + P \times S_{qrt} \tag{5.16} $$

and, the hardware complexity for orthogonalization (${}^{n}HW_{oth}$) within n-D FastICA as shown in (3.9) can be given as:

$$\begin{aligned} {}^{n}HW_{oth} = [n(n-1)P + n^2P] \times M + (n-1)^2P \times {}^{2}A \\ + n(n-1)P \times S \end{aligned} \tag{5.17}$$

Combining (5.15), (5.16) and (5.17), hardware complexity of one n-D FastICA Iteration stage (${}^{n}HW_{fica}$) can be written as:

$$\begin{aligned} {}^{n}HW_{fica} &= {}^{n}HW_{adp} + {}^{n}HW_{nrm} + {}^{n}HW_{oth} \\ &= P[K(2n+1) + n(2n-1)] \times M + n^2P \times S + \\ &P[K(2n-1) + n(n-1)] \times {}^{2}A + nP \times D \\ &+ P(K+n) \times S_{qrng} + P \times S_{qrt} \end{aligned} \tag{5.18}$$

5.3.4.3 Proof of Low Complexity of the Proposed Algorithm

To prove that the proposed vector cross product based FastICA algorithm reduces the hardware complexity significantly over the conventional FastICA, it is sufficient if the hardware complexity involved in computing one n-D FastICA Iteration can be shown to be significantly higher than that involved in an n-D cross product computation. Therefore, it is essential to compare the hardware complexities ${}^{nD}HW_{str}$ and ${}^{n}HW_{fica}$ as shown in (5.13) and (5.18) respectively.

To do this, the approach described in [86] is followed and the following assumptions are made:

Considering *Assumption* 1.1 and *Assumption* 1.2, hardware complexities as shown in (5.13) and (5.18), can be expressed in terms of TC. Then, taking the difference between these results, Transistor Saving (TS) can be computed. It can easily be shown thereafter that TS is significantly higher for the proposed generalized vector cross product based FastICA algorithm than the conventional one.

To give an insight into the low complexity, Fig. 5.7(a) shows the variation of TS with the change of dimension (n, denoted as *dim* in the figure), word-length (b) and frame-length (K) for a fixed Iteration of Convergence (IOC) $M = 5$ and Fig. 5.7(b) plots the variation of TS with the change of dimension, word-length and IOC for a fixed frame-length $K = 512$. The range of word-length and dimension are considered between 4 to 32 and 3 to 6 respectively. From both of these figures it can be seen that TS increases significantly with the increase of the dimension and word-length. From both the figures, it can also be seen that TS is very prominent among different dimensions from the word-length $b = 8$ onwards.

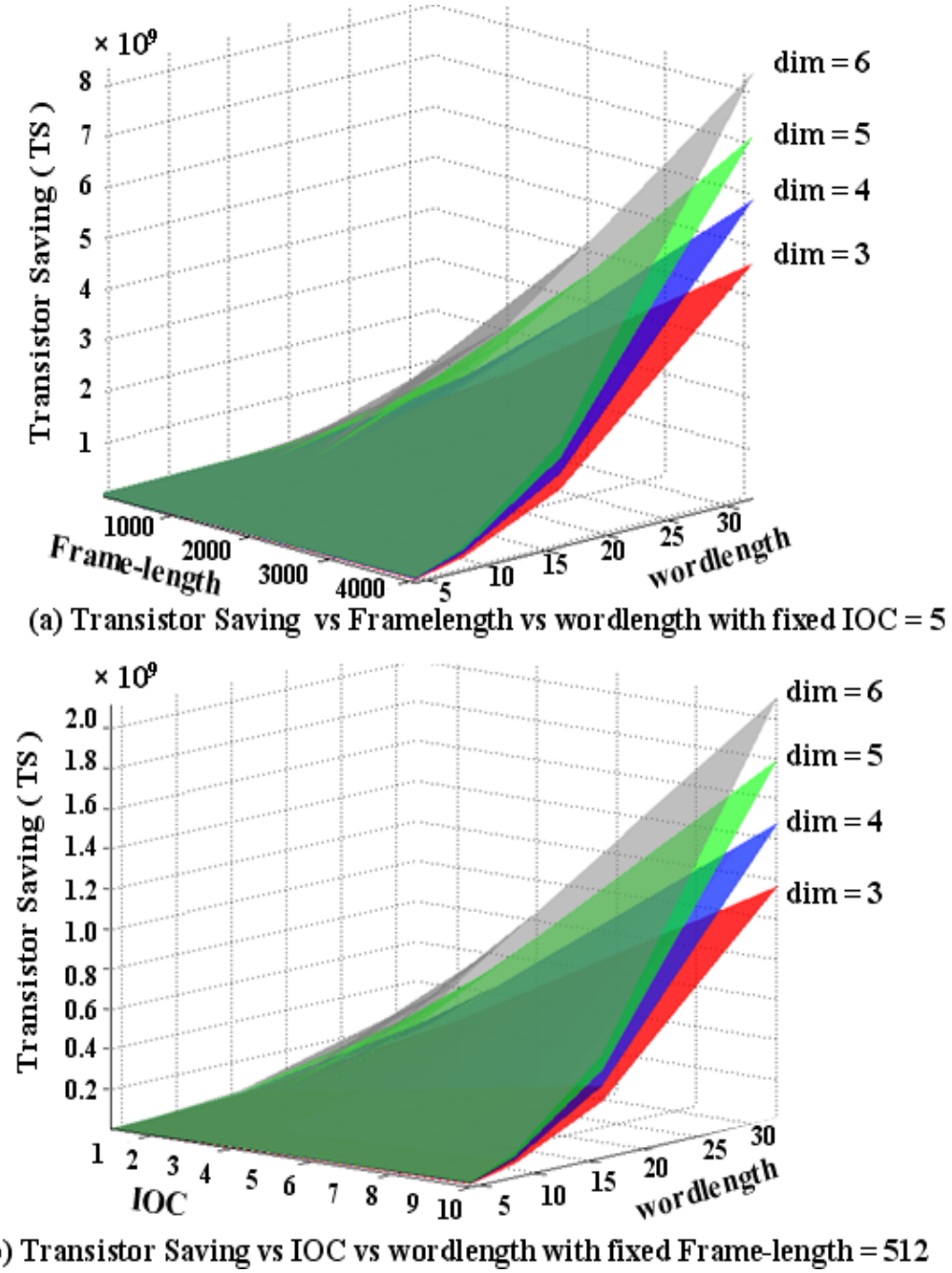


FIGURE 5.7: Comparative Transistor Savings (TS) analysis between the proposed cross-product based FastICA and the conventional FastICA. (a) Variation of TS with Frame-length, Word-length and Dimension (dim = $3, 4, 5, 6$) for fixed IOC= 5, (b) variation of TS with IOC, Word-length and Dimension (dim = $3, 4, 5, 6$) for fixed Frame-length = 512.

5.3.5 Delay Analysis: Proposed Cross product based nD FastICA

This subsection analyzes the performance of the proposed cross-product based FastICA algorithm in terms of delay. The same strategy used to design the mathematical models for the hardware complexity of this proposed algorithm in subsection 5.3.4 is adopted here as well.

5.3.5.1 Delay of n-D Cross Product

The delay of the nD cross-product computation has been discussed in detail in the last chapter. Here for the comparison purpose the proposed generalized nD cross product computation algorithm is considered. However, if symmetry based approach is used, the performance in terms of the operational delay will improve.

5.3.5.2 Delay of one n-D FastICA Iteration

Considering *Kurtosis* as the contrast function again, following the same approach used in the last subsection to obtain hardware complexity as shown in (5.18), delay of an n-D FastICA Iteration stage (${}^{nD}\Gamma_{fica}$) can be written as:

$$\begin{aligned} {}^{nD}\Gamma_{fica} = P[K(2n+1)+n(2n-1)] \times \tau_m + n^2P \times \tau_s + \\ P[K(2n-1)+n(n-1)] \times {}^{2}\tau_a + nP \times \tau_d \\ + P(K+n) \times \tau_{sqrn} + P \times \tau_{sqrt} \end{aligned} \tag{5.19}$$

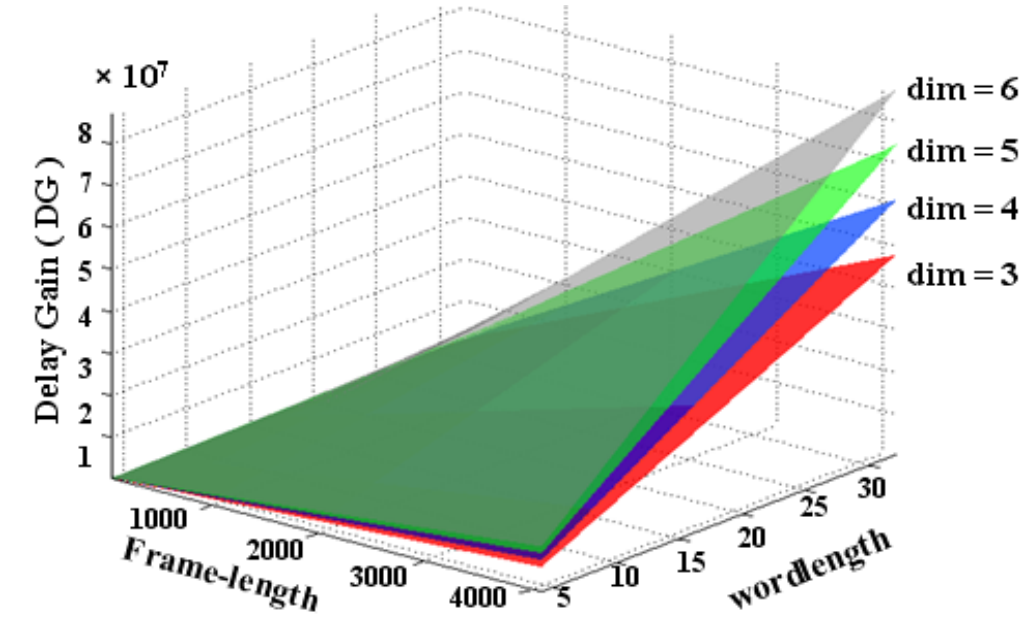
where along with the notations introduced before, τ_d, τ_{sqrn} and τ_{sqrt} are the delay associated with a divider, squaring unit and a square rooter.

5.3.5.3 Proof of Operational Speed-up

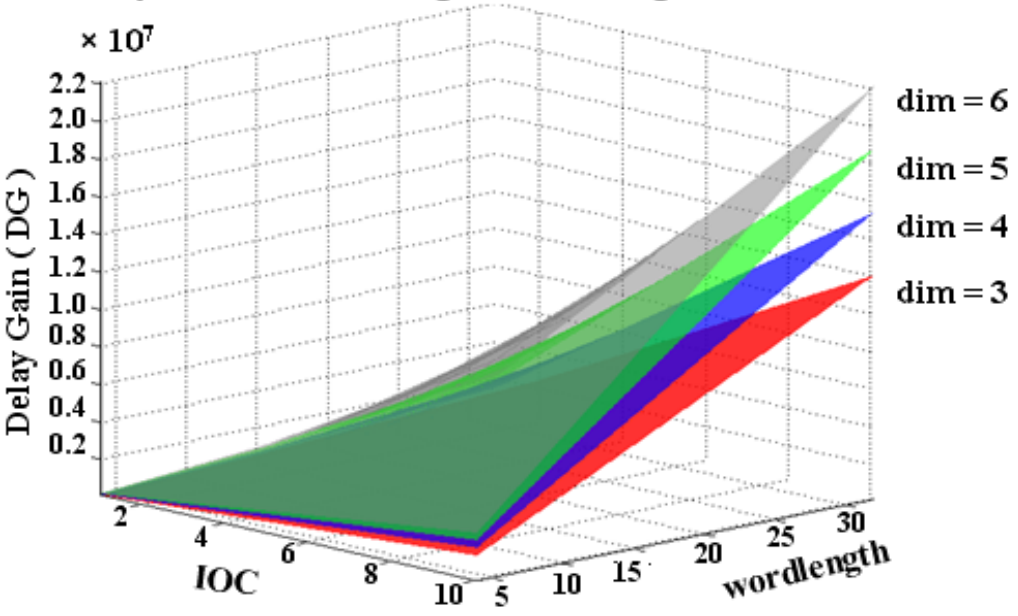
To show the operational speed up due to the proposed vector cross product based FastICA over the conventional FastICA, a comparison of the delay, as derived in (4.44) and (5.19), is carried out here using the following assumptions:

Using *Assumption* 2.1 in (4.44) and (5.19), ${}^{nD}\Gamma_{str}$ and ${}^{nD}\Gamma_{fica}$ can be expressed in terms of $\triangle$. Thereafter taking their difference and normalizing the result with respect to $\triangle$, the normalized Delay Gain (DG) for the proposed algorithm over the conventional one can easily be derived.

To give an insight into the comparative results on delay, Fig. 5.8(a) plots the variation of DG with different dimensions (n, denoted by *dim* in the figure), word-length (b, considered in the range from 4 to 32) and frame-length (K, considered within the range



(a) Delay Gain vs Framelength vs wordlength with fixed IOC = 5



(b) Delay Gain vs IOC vs wordlength with fixed Frame-length = 512

FIGURE 5.8: Comparative Delay Gain (DG) analysis between the proposed cross-product based FastICA and conventional FastICA. (a) Variation of DG with Frame-length, Word-length and Dimension (dim = $3, 4, 5, 6$) for fixed IOC = 5, (b) variation of DG with IOC, Word-length and Dimension (dim = $3, 4, 5, 6$) for fixed Frame-length = 512.

from 128 to 4096) for a fixed Iteration of Convergence (IOC) $M = 5$ and Fig 5.8(b) shows the variation of DG with respect to different dimensions, word-length and IOC (M, considered within a range from 1 to 10) for a fixed frame-length $K = 512$. It can easily be seen from these two figures that significant operational speed up is achieved in the proposed algorithm over the conventional one with the increase in word-length as well as dimension.

5.4 Algorithm Validation

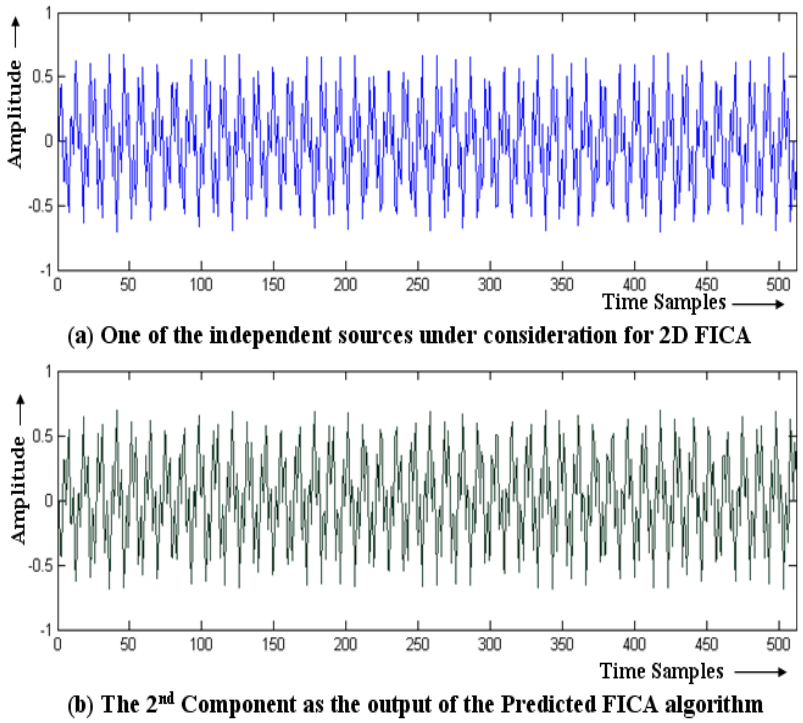


FIGURE 5.9: (a) $0.7\sin(450t)\sin(40t)$, considered as one of the two sources for 2-D FastICA, (b) 2^{nd} component as the output of the Predicted FastICA algorithm given as motivational example in Section 5.1.

Before proceeding further, it is important to note that the dimensionality (n) of FastICA is dependent on the corresponding applications. So theoretically, n may be very high. But to be realistic, in our envisaged model for remote health monitoring environment, it is not possible to place large number of sensors over a patient's body because ethically technology should not be a burden to him/her and should not prevent him/her to lead a normal life. Therefore, although a generalized cross product based nD FastICA algorithm is proposed, but from the point of view of its implementation, validation of this algorithm will be restricted here only upto 4^{th} ($n = 4$) dimension.

This section validates the proposed algorithm for $2D$, $3D$ and $4D$ as representative cases. For this purpose software model for $2D$, $3D$ and $4D$ cases for both conventional as well as the proposed algorithm has been generated in C and simulation results are

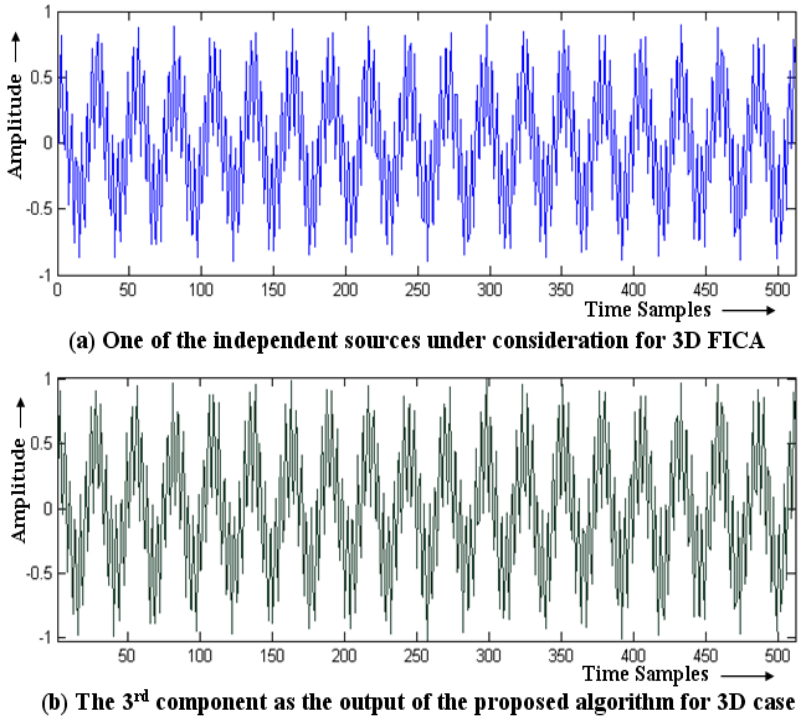


FIGURE 5.10: (a) One of the sources for 3-D FastICA, (b) 3^{rd} component as one of the estimated outputs of the 3-D FastICA based on the proposed algorithm.

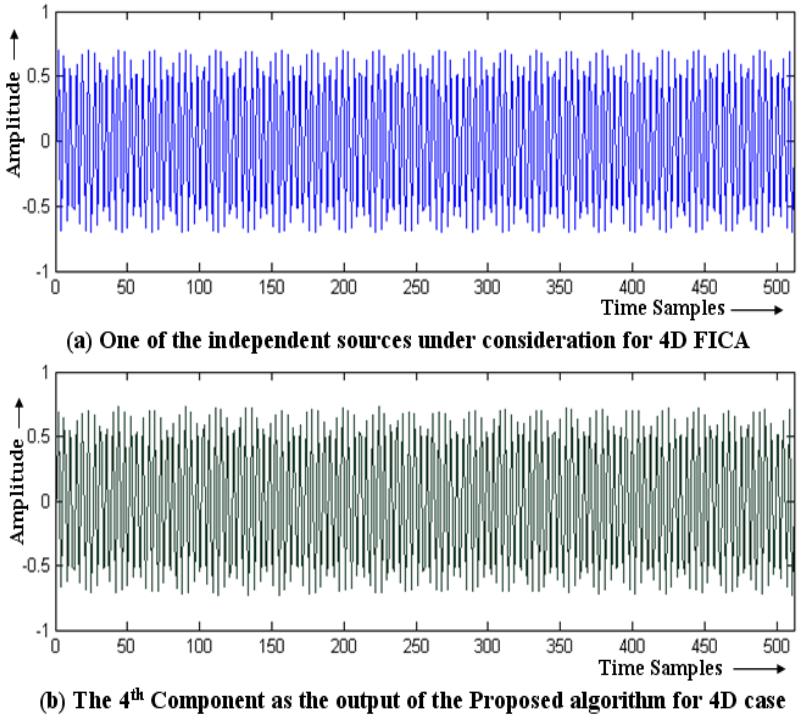


FIGURE 5.11: (a) One of the independent sources for 4-D FastICA, (b) 4^{th} component as one of the estimated outputs of the 4-D FastICA based on the proposed algorithm.

shown in Fig. 5.10 and 5.11. Without loss of generality, the same method of proof can be extended to any arbitrary dimensions.

Although the proposed cross product based algorithm, as described in Section 5.2, considers 3 or higher dimensional FastICA, its motivation came from the predictive algorithm discussed in Section 5.1 and so first the simulation results are provided to validate 2-D predictive FastICA in Fig. 5.9.

Since in the *Predictive algorithm*, the first vector of the unmixing matrix is determined using the conventional 2-D FastICA and the second vector is predicted from this first one, therefore Fig. 5.9(b) only shows the second estimated component which is the output of the predicted algorithm. It can clearly be seen from this figure that the output exactly matches (upto a sign inversion) with one of the two sources considered (as shown in Fig. 5.9(a)) in this case and thus Fig. 5.9 validates the *Predicted Algorithm*.

It was discussed in detail in Section 5.2 that for n-D FastICA, the proposed algorithm derives the n^{th} vector of the unmixing matrix from the $(n-1)$ vectors obtained using the conventional FastICA algorithm. Therefore for 3-D and 4-D FastICA cases it is essential to check whether the respective 3^{rd} and 4^{th} component match with one of the independent sources. Thus only the 3^{rd} and 4^{th} estimated components as the outputs of the 3-D and 4-D cross product based FastICA algorithm are provided in Fig. 5.10(b) and Fig. 5.11(b) respectively.

From Fig. 5.10(b), it can easily be observed that the 3^{rd} component obtained as the output of the proposed algorithm for the 3-D case, exactly matches with one of the original sources considered for the 3-D FastICA as shown in Fig. 5.10(a). In the same way it can be observed from Fig. 5.11(b) that the 4^{th} component as the output of the proposed cross product based 4-D FastICA algorithm exactly matches with one of the sources considered for 4-D FastICA case as shown in Fig. 5.11(a).

To give an insight into the corresponding hardware implementation result, the $2D$ architecture based on the Proposed Predictive Approach is designed using Verilog HDL, synthesized using Synopsys Design Compiler using $0.13-\mu$m standard cell CMOS technology and power analysis is done using Synopsys Prime Time. The synthesized area and power consumption are obtained as 3.34 mm^2 and 20.6 μW at 1-MHz frequency for $V_{DD} = 1.2$V.

Due to unavailability of the implementation results in terms of silicon area and power consumption in [76], we are unable to compare the fore-mentioned results with [76]. The reduced computational complexity of the proposed algorithm leads to 6% area reduction and 24% power consumption reduction when the aforementioned results are compared with the latest reported architecture implementation results in [137].

5.5 Concluding Remarks

In this chapter, a low complexity $2D$ predictive FastICA algorithm has been proposed and then it has been explored further to understand how this predictive algorithm can be extended in higher dimensions by investigating the concept of vector cross product. It has been shown that the generalized cross product concept can be utilised for the reduction of computational complexity of the conventional FastICA and subsequently a low complexity nD FastICA algorithm has been proposed based on the generalized nD vector cross product introduced in the last chapter. It has also been shown in Section 5.3 that the introduction of this concept is capable of reducing the computational complexity significantly compared to the conventional FastICA. It has also been pointed out in the first section of this chapter that the targeted application domain of the proposed cross product based FastICA algorithm would be in the remote health monitoring environment. In such environment, from the practical point of view, patients are fitted with wearable sensors which are supposed to be very less in number so that these do not interfere with the daily life activities of the patients. Therefore two to five sensors are generally used over the patient's body in such environment. In this case the proposed algorithm would outperform the conventional FastICA in terms of less hardware complexity as shown in Fig. 5.5 and Fig. 5.7.

Chapter 6

FastICA Based on Co-ordinate Rotation

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 [28] - [74]. 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 [117]. It has already been outlined Chapter 1, 3 and 5 that ICA is the most commonly used statistical technique for solving BSS problem [117] and among the existing ICA algorithms, FastICA is popular because of its higher convergence speed and accuracy [75], [77]. Being efficient from the algorithmic point of view, FastICA can be deployed in the fore-mentioned applications.[1]

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.

Recalling from Chapter 3 and 5, FastICA algorithm consists of two basic steps - Preprocessing and the main Iterative step [77]. Preprocessing step is composed of two further steps - centering and whitening [77]. Centering requires add and accumulation unit

[1]The contents of Section 6.2, 6.3, 6.12 and Section 6.13 have partly appeared as "Co-ordinate Rotation Based Low Complexity 2D FastICA Algorithm and Architecture" by Acharyya *et. al.* in *IEEE International Conference on Green Circuits and Systems*. Please see Appendix-A for further detail.

whereas whitening needs computation of eigen values and the corresponding eigen vectors of the covariance matrix formed using the mixed sensory data set [77]. Coordinate Rotation Digital Computer (CORDIC) is the widely used technique for implementing Eigen Value Decomposition (EVD) in hardware [139] - [145]. 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. The first hardware implementation of $2D$ (i.e. the basic two sources - two sensors scenario) FastICA was reported in [76] which is a floating-point-based direct mapping of the algorithm and no consideration was given to the silicon area and power consumption reduction. Recently we have reported a hardware efficient $2D$ FastICA architecture in [122] which uses fixed-point arithmetic and optimized the Preprocessing unit by removing some computationally intensive arithmetic operations. This proposed methodology and its corresponding architectures have already been discussed in detail in Section 3.8 of Chapter - 3. However such architectural modifications are obtained using algebraic method which can not be extended in higher dimensions where $n > 2$. Therefore the FastICA architectures discussed in [76] and [122] can not be generalized for nD cases where $n \geq 2$.

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. 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 $n \geq 2$ 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.

The rest of this chapter is organized as follows. Section 6.1 does the necessary theoretical groundwork in brief, Section 6.2, 6.4, 6.6 deal with the relationship between CORDIC and 2D, 3D and 4D FastICA algorithm and propose the corresponding Co-ordinate Rotation based FastICA algorithm for those dimensions, Section 6.3, 6.5 and 6.7 present the corresponding architectures, Section 6.8 proposes the generalized nD FastICA algorithm based on CORDIC following the same treatment used for derivation of 2D, 3D and 4D algorithm, Section 6.9 provides the CORDIC based nD FastICA architecture, Section 6.10 brings the concept of doubly pipelining within the proposed CORDIC based nD FastICA algorithm for architectural optimization, Section 6.11 discusses about some important implementation issues of the proposed algorithm, Section 6.12 analyzes the hardware complexity of the proposed architecture, Section 6.13 validates the proposed algorithm and finally Section 6.14 concludes the chapter.

6.1 Preliminaries

6.1.1 Conventional FastICA Algorithm

Comprehensive theoretical background of the conventional FastICA algorithm has already been discussed in Section 3.3.

6.1.2 Coordinate Rotation Digital Computer

CORDIC is an efficient implementation technique for vector rotation and arctangent computation and since it can be realized using simple shift and add operations, it is effective in terms of low hardware complexity [146] - [148]. CORDIC, in general, can be operated in two modes - rotation and vectoring [148]. In rotation mode, given the angle of rotation and the initial vector, final vector is computed and in the vectoring mode, the angle between the initial vector and the principal coordinate axis is computed. Considering the rotation in clockwise sense, the basic CORDIC expressions can be expressed as [146] - [148]:

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{6.1}$$

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

6.2 Proposed CORDIC based 2D FastICA Algorithm

6.2.1 2D FastICA Iteration Stage

In the case under consideration for $2D$ FastICA (i.e. $n = 2$), expanded form of (3.8) can be written as:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)}\}^3] \\ \mathcal{E}[z_{2,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \end{bmatrix} \tag{6.2}$$

where p denotes the number of iteration stage, $z_{i,j}$ represents the i^{th} whitened data containing j number of samples where $i = \{1, 2\}$ and $j \in (1, m)$ where m denotes the frame-length, ${w_{1,q}}^{(p+1)}$ is the 1^{st} column of the unmixing matrix after p^{th} iteration where $q = \{1, 2\}$ and ${\underline{w}_{1,q}}^{(p)}$ indicates the *normalized value* of ${w_{1,q}}^{(p)}$ used in p^{th} iteration as given by (3.10). Without any loss of generality, ${\underline{w}_{1,1}}^{(p)}$ and ${\underline{w}_{1,2}}^{(p)}$ can be considered as

the components of a unit norm vector $\underline{\mathbf{w}}_1{}^{(p)}$. Now transforming from Cartesian to Polar co-ordinate system, $\underline{\mathbf{w}}_1{}^{(p)}$ can be re-written as:

$$\underline{\mathbf{w}}_1{}^{(p)} = [\underline{w}_{1,1}{}^{(p)} \ \underline{w}_{1,2}{}^{(p)}]^T = [\cos\theta_p \ \ \sin\theta_p]^T \tag{6.3}$$

where polar angle $\theta_p = \tan^{-1}(\underline{w}_{1,2}{}^{(p)}/\underline{w}_{1,1}{}^{(p)})$ at p^{th} iteration stage. Since the vector $\underline{\mathbf{w}}_1{}^{(p)}$ is of unit-norm, the magnitude of the radial vector is one and thus does not appear in (6.3). From (6.1) it can be written as:

$$x_f = x_0 \cos\theta + y_0 \sin\theta \tag{6.4}$$

Using (6.3) in (6.2) and considering the bracketed portion of (6.2) on which cubing has to be performed, a similarity can be found with (6.4) where $x_0 = z_{1,j}$, $y_0 = z_{2,j}$ and $\theta = \theta_p$ at the p^{th} FastICA iteration stage.

Instead of using the complete expression shown in (6.1), one concise notation can be introduced as follows for brevity: $x_f = \mathcal{R}ot_x(x_0, y_0, \theta)$ and $y_f = \mathcal{R}ot_y(x_0, y_0, \theta)$ where $\mathcal{R}ot_{x/y}(\cdot)$ denotes the x/y component of the rotation mode CORDIC outputs. Using this notation, (6.2) can be modified as:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \theta_p)\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \theta_p)\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \end{bmatrix} \tag{6.5}$$

From (6.5) it can be seen that rotation mode CORDIC can be used in the main FastICA Iterative stage. However explicit information of θ_p is not available and has to be derived from $\underline{w}_{1,1}{}^{(p)}$ and $\underline{w}_{1,2}{}^{(p)}$ instead. From the line following (6.3) it has already been seen that θ_p is the *arctangent* of two components of the vector $\underline{\mathbf{w}}_1{}^{(p)}$ which is exactly the same as the CORDIC operating in Vectoring mode as described in Section 6.1. Again, introducing another concise notation for the sake of simplicity, θ_p can be denoted as $\mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)})$ where the x and y-inputs of the vectoring mode CORDIC is denoted by $\underline{w}_{1,1}{}^{(p)}$ and $\underline{w}_{1,2}{}^{(p)}$ respectively. Using this notation, (6.5) can be modified as:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}))\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}))\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \end{bmatrix} \tag{6.6}$$

Thus (6.6) represents the mapping of the conventional $2D$ FastICA Iterative stage in terms of the rotation and vectoring mode CORDIC.

6.2.2 2D FastICA Normalization Stage

Since the basic FastICA Iteration is not a norm-preserving operation, normalization of the newly obtained vector is necessary as pointed out in (3.10). This step can also be accomplished using the same CORDIC in the following way.

Denoting the normalized components of the vector obtained after p^{th} iteration using (6.2) by $\underline{w}_{1,1}{}^{(p+1)}$ and $\underline{w}_{1,2}{}^{(p+1)}$ the following equation holds:

$$\underline{w}_{1,i}{}^{(p+1)} = \frac{w_{1,i}{}^{(p+1)}}{\sqrt{|\mathbf{w}_1{}^{(p+1)}|^2}} \tag{6.7}$$

where $i = \{1, 2\}$. Using Cartesian to Polar Co-ordinate transformation following (6.3), these two components can be represented as:

$$\underline{\mathbf{w}}_1{}^{(p+1)} = [\underline{w}_{1,1}{}^{(p+1)} \ \underline{w}_{1,2}{}^{(p+1)}]^T = [\cos\theta_{(p+1)} \ \ \sin\theta_{(p+1)}]^T \tag{6.8}$$

Now, considering $x_0 = 0$ and $y_0 = 1$ in (6.1), it can be found that $x_f = \sin\theta$ and $y_f = \cos\theta$. Thus a similarity can be established between this (y_f, x_f) and (6.8) and following the same notations used in (6.5) and (6.6), (6.8) can be written as:

$$\begin{bmatrix} \underline{w}_{1,1}{}^{(p+1)} \\ \underline{w}_{1,2}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,1}{}^{(p+1)}, w_{1,2}{}^{(p+1)})) \\ \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,1}{}^{(p+1)}, w_{1,2}{}^{(p+1)})) \end{bmatrix} \tag{6.9}$$

It is to be observed from (6.9) that $\underline{w}_{1,1}{}^{(p+1)}$ and $\underline{w}_{1,2}{}^{(p+1)}$ are the y and x output of the Rotation mode CORDIC respectively.

6.2.3 2D FastICA Component Estimation Stage

After finite number of FastICA Iterations once the normalized vector converges, using (3.6), one independent component is estimated. Here it will be shown that unlike the conventional FastICA algorithm, complete array of multiplication operations can be removed by reusing CORDIC.

Denoting the converged normalized vector by $\underline{\mathbf{w}}_1{}^c = [\underline{w}_{1,1}{}^c \ \underline{w}_{1,2}{}^c]^T$ where superfix "c" stands for *convergence*, denoting the estimated component by $\hat{s}_1 = \{\hat{s}_{1,j}\}$ where $j \in (1, m)$ and m is the frame-length and using the same set of arguments used to derive (6.5) and (6.6), (3.6) can explicitly be written as:

$$\begin{aligned} \hat{s}_{1,j} &= z_{1,j}\underline{w}_{1,1}{}^c + z_{2,j}\underline{w}_{1,2}{}^c \\ &= \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(\underline{w}^c_{1,1}, \underline{w}^c_{1,2})) \end{aligned} \tag{6.10}$$

Relinquishing "c" from (6.10), $\hat{s}_{1,j}$ can be rewritten as:

$$\hat{s}_{1,j} = \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) \tag{6.11}$$

where $w_{1,1}$ and $w_{1,2}$ are the components of the unnormalized vector obtained after FastICA iteration prior to convergence checking. However the reason behind such relinquishment of "c" from (6.11) is deferred until the next section.

6.3 Proposed Architecture for CORDIC based 2D FastICA

Fig. 6.1, 6.2 and 6.3 present the architecture of the co-ordinate rotation based $2D$ FastICA algorithm proposed in the last Section. Since it has already been mentioned in the first section that CORDIC is widely used for EVD which is the fundamental operation in the preprocessing step of FastICA and since the proposed algorithm in the last section has already shown how CORDIC can also be used for implementing the main FastICA step, the controller design of the overall architecture becomes an important task to ensure proper data enters the CORDIC unit at the right time. As outlined in the last section, the controller operation can also be divided into three modes of operations - (i) *iteration*, (ii) *normalization* and (iii) *estimation*. These three modes can be used to generate *select* signals for the multiplexers and demultiplexer at the inputs and outputs of the CORDIC shown in Fig. 6.4.

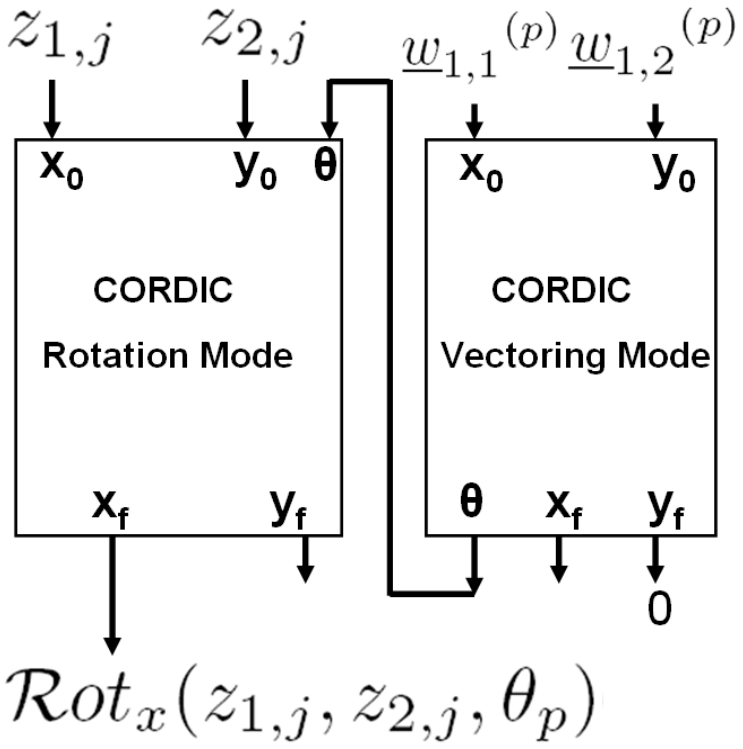


FIGURE 6.1: Iteration stage of the proposed CORDIC based 2-D FastICA.

6.3.1 $2D$ Iteration Mode: Unfolded Architecture

When *iteration* mode is active, the x and y inputs of the Vectoring mode CORDIC are $\underline{w}_{1,1}{}^{(p)}$ and $\underline{w}_{1,2}{}^{(p)}$ respectively as also shown in (6.6) where p denotes the iteration number (see Fig. 6.1). The accumulated angle θ_p is obtained at the output of the

Vectoring mode CORDIC which can be connected to the θ-input of the Rotation mode CORDIC as shown in Fig. 6.4. The x and y inputs of the Rotation mode CORDIC under this active *iteration* mode become $z_{1,j}$ and $z_{2,j}$ respectively as also shown in (6.6). These data can be accessed serially from the respective memories. The x-output of the Rotation mode CORDIC becomes $\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \theta_p)$. Corresponding y-output is not used and should be ignored (see Fig. 6.1).

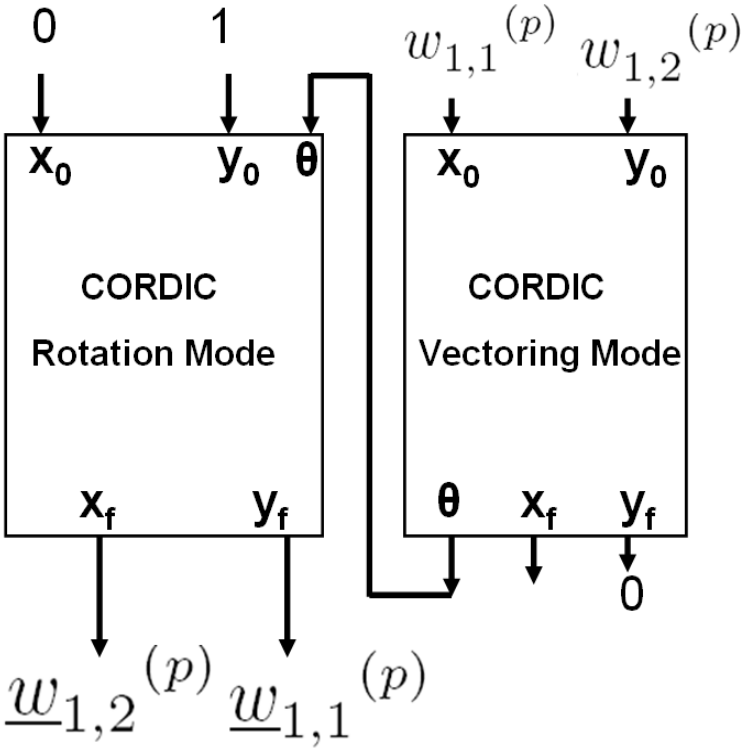


FIGURE 6.2: Normalization stage of the proposed CORDIC based 2-D FastICA.

6.3.2 $2D$ Normalization Mode: Unfolded Architecture

When *normalization* mode is active, the unnormalized vector components ${w_{1,1}}^{(p)}$ and ${w_{1,2}}^{(p)}$ are to be considered as the x and y inputs of the Vectoring mode CORDIC respectively as shown in Fig. 6.2. The accumulated angle θ_p is computed and fed to the input of the Rotation mode CORDIC where 0 and 1 are considered as the inputs to x and y inputs respectively as shown in (6.9) (see Fig. 6.2). As mentioned in the line following (6.9) care has to be taken while connecting the outputs of Rotation mode CORDIC with the next computational block in the *normalization* mode. The x and y outputs of the Rotation mode CORDIC become ${\underline{w}_{1,2}}^{(p)}$ and ${\underline{w}_{1,1}}^{(p)}$ respectively (see Fig. 6.2).

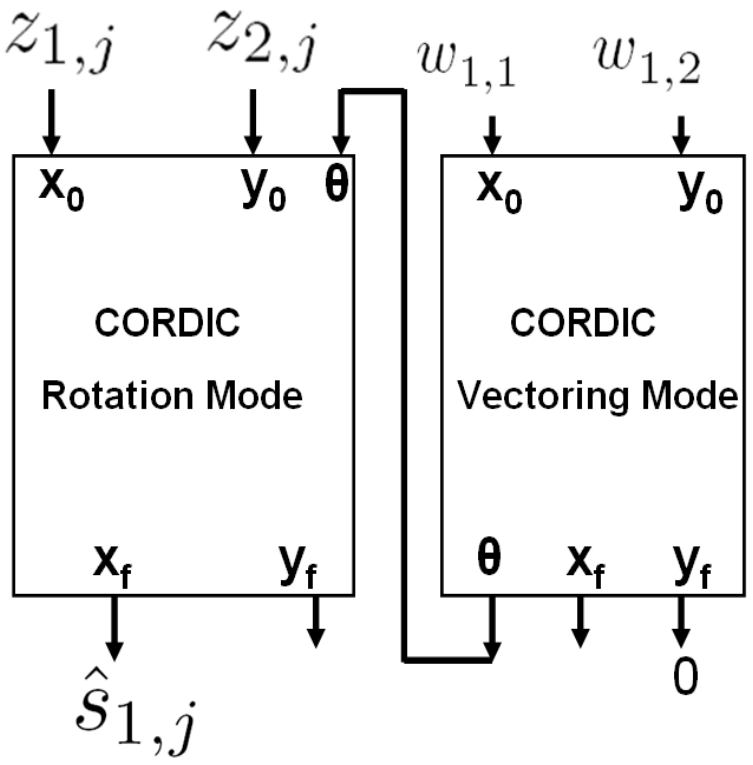


FIGURE 6.3: Estimation stage of the proposed CORDIC based 2-D FastICA.

6.3.3 $2D$ Estimation Mode: Unfolded Architecture

When *estimation* mode is active, Vectoring mode CORDIC needs not to be used because the accumulated angle information is already computed during the previous normalization step and can be reused as the θ input of the Rotation mode CORDIC. This is the reason why "c" is relinquished from (6.10) to derive (6.11). Similar to *iteration* mode, in this *estimation* mode also $z_{1,j}$ and $z_{2,j}$ are to be considered as the x and y inputs of the Rotation mode CORDIC and here also the y-output has to be ignored (see Fig. 6.3).

6.3.4 Multiplexed Architecture: CORDIC Reuse for $2D$

Since iteration, normalization and estimation - all stages are executed sequentially, same CORDIC unit can be reused for implementing these stages only at the expense of multiplexers at the inputs of the rotation and vectoring mode CORDIC. Combining these three stages as shown in Fig. 6.1, 6.2 and 6.3, it is possible to model the multiplexers based single CORDIC unit for $2D$ FastICA implementation as shown in Fig. 6.4. It can be noted from Fig. 6.4 that one $2:1$ multiplexer is used at each of the x and y inputs of the Rotation and Vectoring mode CORDIC. Use of the demultiplexer as shown in Fig. 6.4 may not be mandatory and can be replaced by direct hardware wiring.

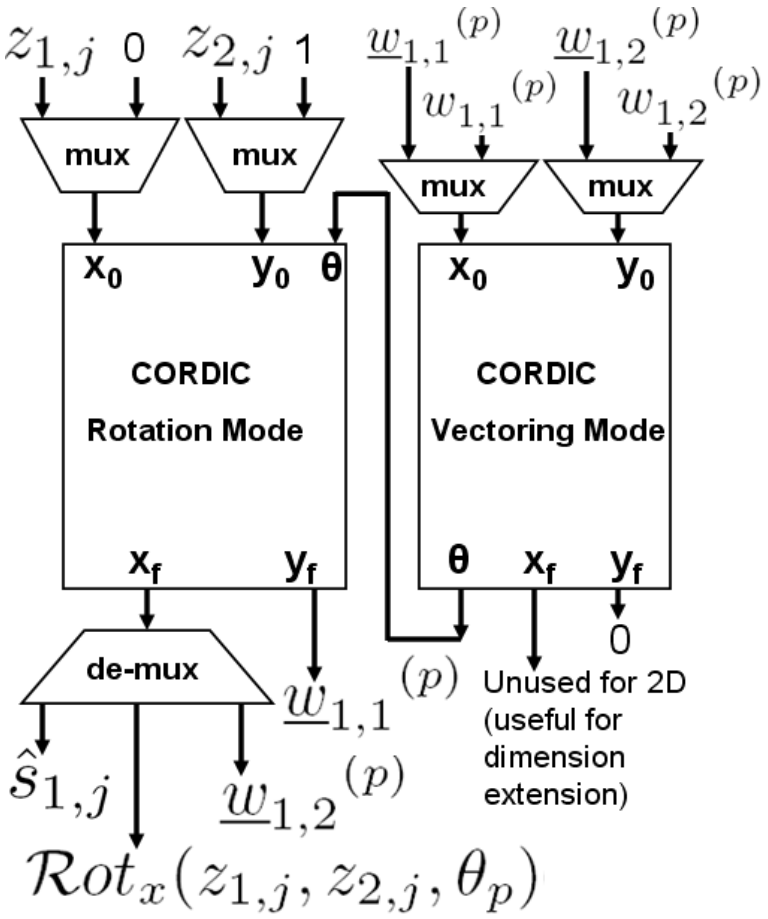


FIGURE 6.4: Architecture of the proposed Co-ordinate Rotation based 2-D FastICA.

6.3.5 Architectural Optimization Possibility

At this point, one question may arise that the proposed algorithm may increase the latency of computation because the accumulated angle has to be fed to the input of the Rotation mode CORDIC and it is not available until the Vectoring mode CORDIC finishes its operation. However, it is important to mention that the explicit information of the accumulated angle θ_p is not necessary and thus instead of using θ_p, each micro-rotation sequence can be fed to each unit of Rotation mode CORDIC by using *Doubly Pipelining* approach [149]. But further discussion on this approach is deferred until Section 6.10.

6.3.6 Scope of Dimension Extension

In the first section it has been mentioned that the main aim is to generalize FastICA algorithm in nD vector space using co-ordinate rotation based approach and in this section and in the previous one initial research on $2D$ FICA along that line is presented. Since the proposed algorithm in the last section is based on the topological rather than

the algebraical pattern exploitation of the basic FastICA algorithm, it is indeed possible to extend the $2D$ concept in nD vector space and in that process increase of the controller complexity will be inevitable. It is to be noted from Fig. 6.4 that the y-output of the Vectoring mode CORDIC is 0 (approximately) and x-output provides the magnitude of the radial vector in $2D$ $x-y$ plane. This x-output, in case of $3D$, can be used as the x-input of the Vectoring mode CORDIC along with the third component of the vector of the unmixing matrix as the y-input. In this way, this concept of $2D$ co-ordinate rotation based FastICA can be extended from $x-y$ plane to $x-y-z$ plane and higher dimensions. Thus nD extension of $2D$ co-ordinate rotation concept based FastICA will be formulated in the sections to follow and in Section 6.8 CORDIC based generalized nD FastICA algorithm will be proposed and in Section 6.9 its corresponding architectures will be presented.

6.4 Proposed CORDIC based 3D FastICA Algorithm

6.4.1 3D FastICA Iteration Stage

The case under consideration here in this section is 3D FastICA (i.e. $n = 3$) and using (3.8) its expanded form can be written as:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)}\}^3] \\ \mathcal{E}[z_{2,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)}\}^3] \\ \mathcal{E}[z_{3,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \end{bmatrix} \tag{6.12}$$

where p denotes the number of iteration stage, $z_{i,j}$ represents the i^{th} whitened data containing j number of samples where $i = \{1, 2, 3\}$ and $j \in (1, m)$ where m denotes the frame-length, ${w_{1,q}}^{(p+1)}$ is the 1^{st} column of the unmixing matrix after p^{th} iteration where $q = \{1, 2, 3\}$ and ${\underline{w}_{1,q}}^{(p)}$ indicates the *normalized value* of ${w_{1,q}}^{(p)}$ used in p^{th} iteration as given by (3.10). Without any loss of generality, ${\underline{w}_{1,1}}^{(p)}$, ${\underline{w}_{1,2}}^{(p)}$ and ${\underline{w}_{1,3}}^{(p)}$ can be considered as the components of a unit norm vector ${\underline{\mathbf{w}}_1}^{(p)}$. Now transforming from Cartesian to Polar co-ordinate system, ${\underline{\mathbf{w}}_1}^{(p)}$ can be re-written as:

$$\begin{aligned} {\underline{\mathbf{w}}_1}^{(p)} &= [{\underline{w}_{1,1}}^{(p)} \ {\underline{w}_{1,2}}^{(p)} \ {\underline{w}_{1,3}}^{(p)}]^T \\ &= \begin{bmatrix} \sin\theta_{2,p}\cos\theta_{1,p} \\ \sin\theta_{2,p}\sin\theta_{1,p} \\ \cos\theta_{2,p} \end{bmatrix} \end{aligned} \tag{6.13}$$

where, the spherical angles $\theta_{1,p}$ and $\theta_{2,p}$ at p^{th} iteration stage, as shown in Fig. 6.5, are defined as $\tan^{-1}({\underline{w}_{1,2}}^{(p)}/{\underline{w}_{1,1}}^{(p)})$ and $\tan^{-1}({\underline{w}_{1,3}}^{(p)}/\sqrt{({\underline{w}_{1,1}}^{(p)})^2 + ({\underline{w}_{1,2}}^{(p)})^2})$ respec-

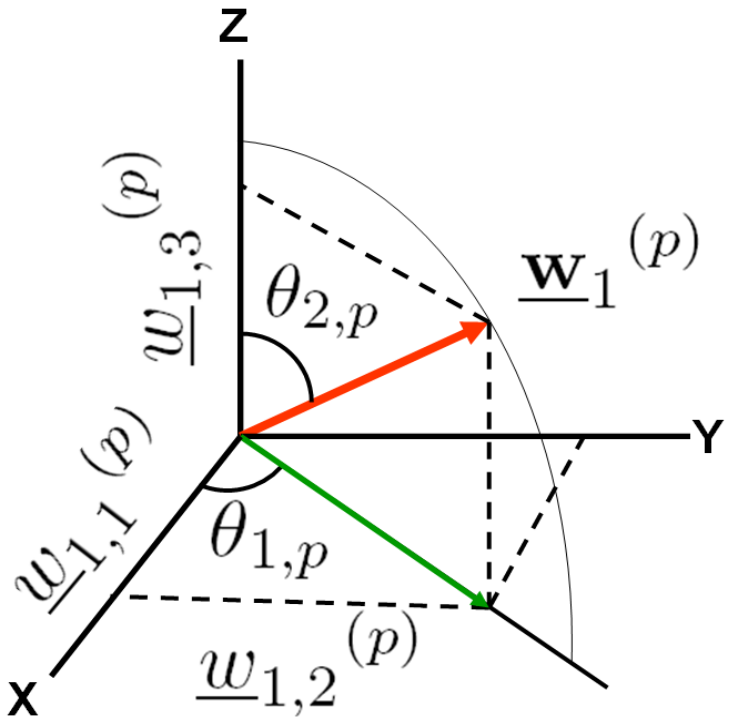


FIGURE 6.5: 3D representation of $\underline{\mathbf{w}}_1{}^{(p)}$ in Spherical Co-ordinate system.

tively. Since the vector $\underline{\mathbf{w}}_1{}^{(p)}$ is unit norm vector, the magnitude of this vector is one and thus does not appear in (6.13). Using (6.13), (6.12) can be modified to:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \\ w_{1,3}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{z_{1,j}\sin\theta_{2,p}\cos\theta_{1,p} + z_{2,j}\sin\theta_{2,p}\sin\theta_{1,p} + z_{3,j}\cos\theta_{2,p}\}^3] \\ \mathcal{E}[z_{2,j}\{z_{1,j}\sin\theta_{2,p}\cos\theta_{1,p} + z_{2,j}\sin\theta_{2,p}\sin\theta_{1,p} + z_{3,j}\cos\theta_{2,p}\}^3] \\ \mathcal{E}[z_{3,j}\{z_{1,j}\sin\theta_{2,p}\cos\theta_{1,p} + z_{2,j}\sin\theta_{2,p}\sin\theta_{1,p} + z_{3,j}\cos\theta_{2,p}\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \\ \underline{w}_{1,3}{}^{(p)} \end{bmatrix} \tag{6.14}$$

Taking $\sin\theta_{2,p}$ common, (6.14) can be rearranged as:

$$\begin{bmatrix} w_{1,1}{}^{(p+1)} \\ w_{1,2}{}^{(p+1)} \\ w_{1,3}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{z_{3,j}\cos\theta_{2,p} + (z_{1,j}\cos\theta_{1,p} + z_{2,j}\sin\theta_{1,p})\sin\theta_{2,p}\}^3] \\ \mathcal{E}[z_{2,j}\{z_{3,j}\cos\theta_{2,p} + (z_{1,j}\cos\theta_{1,p} + z_{2,j}\sin\theta_{1,p})\sin\theta_{2,p}\}^3] \\ \mathcal{E}[z_{3,j}\{z_{3,j}\cos\theta_{2,p} + (z_{1,j}\cos\theta_{1,p} + z_{2,j}\sin\theta_{1,p})\sin\theta_{2,p}\}^3] \end{bmatrix} - 3 \begin{bmatrix} \underline{w}_{1,1}{}^{(p)} \\ \underline{w}_{1,2}{}^{(p)} \\ \underline{w}_{1,3}{}^{(p)} \end{bmatrix} \tag{6.15}$$

Recalling the notations of "$\mathcal{R}ot$" and "$\mathcal{V}ec$" introduced in the section 6.2, (6.15) can be rewritten as:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \theta_{1,p}), \theta_{2,p})\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \theta_{1,p}), \theta_{2,p})\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \theta_{1,p}), \theta_{2,p})\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \end{bmatrix} \tag{6.16}$$

It is to be noted from (6.16) that one new notation "*level*" (denoted as "l") is introduced with the $\mathcal{R}ot$ notation. It simply indicates number of *Rotation* required. It can be observed from (6.16) that the innermost $\mathcal{R}ot$ is identified by "*level-1*" (denoted by $l1$) and similarly the next $\mathcal{R}ot$ mode is identified by "$l2$". It is also to be noted from (6.16) that "$\mathcal{R}ot_x^{l1}$" (i.e. the x-output of *level-1* Rotation) is the y-input (i.e. y_0 in (6.1)) of *level-2* Rotation "$\mathcal{R}ot^{l2}$".

From (6.16) it can be seen that rotation mode CORDIC can be used in the main FastICA Iterative stage. However like CORDIC based 2D FastICA, here also explicit angle information of $\theta_{1,p}$ and $\theta_{2,p}$ are not available and have to be derived from ${\underline{w}_{1,1}}^{(p)}$, ${\underline{w}_{1,2}}^{(p)}$ and ${\underline{w}_{1,3}}^{(p)}$ instead.

From the line following (6.13) it has already been seen that $\theta_{1,p}$ is the *arctangent* of the ratio of the first two components of the vector ${\underline{\mathbf{w}}_1}^{(p)}$ which is exactly the same as the CORDIC operating in Vectoring mode. Thus without any loss of generality, using the notation "$\mathcal{V}ec$", $\theta_{1,p}$ can be expressed as follows:

$$\theta_{1,p} = \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}) \tag{6.17}$$

Similarly in the line following (6.13), it has been shown that $\theta_{2,p}$ can be represented as the *arctangent* of the ratio of the third component of ${\underline{\mathbf{w}}_1}^{(p)}$ and the *norm* of the vector defined by the first two components of ${\underline{\mathbf{w}}_1}^{(p)}$ on the $X-Y$ co-ordinate plane (see Fig. 6.5). Therefore using "$\mathcal{V}ec$" notation, $\theta_{2,p}$ can be expressed as:

$$\theta_{2,p} = \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})) \tag{6.18}$$

Using (6.17) and (6.18), the first part of right hand side of (6.16) can be modified as:

$$\begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})), \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})), \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})), \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))\}^3] \end{bmatrix} \tag{6.19}$$

Thus (6.19) represents the mapping of the conventional $3D$ FastICA Iterative stage in terms of the rotation and vectoring mode CORDIC.

6.4.2 3D FastICA Normalization Stage

Since the basic FastICA Iteration, as mentioned in the section 6.2 is not a norm-preserving operation, normalization of the newly obtained vector is necessary as pointed out in (3.10). This step can also be accomplished using the same CORDIC in the following way.

Denoting the normalized components of the vector obtained after p^{th} iteration using (6.12) by $\underline{w}_{1,1}{}^{(p+1)}$, $\underline{w}_{1,2}{}^{(p+1)}$ and $\underline{w}_{1,3}{}^{(p+1)}$ the following equation holds:

$$\underline{w}_{1,i}{}^{(p+1)} = \frac{w_{1,i}{}^{(p+1)}}{\sqrt{|\mathbf{w}_1{}^{(p+1)}|^2}} \tag{6.20}$$

where $i = \{1, 2, 3\}$. Using Cartesian to Polar Co-ordinate transformation following (6.13), these three components can be represented as:

$$\begin{aligned} \underline{\mathbf{w}}_1{}^{(p+1)} &= [\underline{w}_{1,1}{}^{(p+1)} \ \underline{w}_{1,2}{}^{(p+1)} \ \underline{w}_{1,3}{}^{(p+1)}]^T \\ &= \begin{bmatrix} \sin\theta_{2,(p+1)} \cos\theta_{1,(p+1)} \\ \sin\theta_{2,(p+1)} \sin\theta_{1,(p+1)} \\ \cos\theta_{2,(p+1)} \end{bmatrix} \end{aligned} \tag{6.21}$$

Now, considering $x_0 = 0$, $y_0 = \sin\theta_{2,(p+1)}$ and $\theta = \theta_{1,(p+1)}$ in (6.1), it can be found that $x_f = \sin\theta_{2,(p+1)} \sin\theta_{1,(p+1)}$ and $y_f = \sin\theta_{2,(p+1)} \cos\theta_{1,(p+1)}$. Again, considering $x_0 = 0$, $y_0 = 1$ and $\theta = \theta_{2,(p+1)}$ in (6.1), it can be found that $y_f = \cos\theta_{2,(p+1)}$. Thus a similarity can be established between this (y_f, x_f) and (6.21) and following the "$\mathcal{R}ot$" notation, (6.21) can be written as:

$$\begin{bmatrix} \underline{w}_{1,1}{}^{(p+1)} \\ \underline{w}_{1,2}{}^{(p+1)} \\ \underline{w}_{1,3}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{R}ot_y^{l1}(0, \sin\theta_{2,(p+1)}, \theta_{1,(p+1)}) \\ \mathcal{R}ot_x^{l1}(0, \sin\theta_{2,(p+1)}, \theta_{1,(p+1)}) \\ \mathcal{R}ot_y^{l2}(0, 1, \theta_{2,(p+1)}) \end{bmatrix} \tag{6.22}$$

Now, considering $x_0 = 0$, $y_0 = 1$ and $\theta = \theta_{2,(p+1)}$ in (6.1), it can be found that $x_f = \sin\theta_{2,(p+1)}$. Using this concept and also using the notation used in (6.17), (6.22) can be modified as:

$$\begin{bmatrix} \underline{w}_{1,1}{}^{(p+1)} \\ \underline{w}_{1,2}{}^{(p+1)} \\ \underline{w}_{1,3}{}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{R}ot_y^{l1}(0, \mathcal{R}ot_x^{l2}(0, 1, \theta_{2,(p+1)}), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})) \\ \mathcal{R}ot_x^{l1}(0, \mathcal{R}ot_x^{l2}(0, 1, \theta_{2,(p+1)}), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})) \\ \mathcal{R}ot_y^{l2}(0, 1, \theta_{2,(p+1)}) \end{bmatrix} \tag{6.23}$$

It is to be noted from (6.23) that the explicit information of $\theta_{1,(p+1)}$ is unavailable and

that is why it was necessary to represent it using (6.17) in (6.22). Following the same way, representing $\theta_{2,(p+1)}$ using (6.18), $\underline{w}_{1,1}{}^{(p+1)}$, $\underline{w}_{1,2}{}^{(p+1)}$ and $\underline{w}_{1,3}{}^{(p+1)}$ from (6.23) can be written as follows:

$$\begin{aligned}\underline{w}_{1,1}{}^{(p+1)} = \mathcal{R}ot_y^{l1}(0, \mathcal{R}ot_x^{l2}(0, 1, \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))),\\ \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\end{aligned} \tag{6.24}$$

$$\begin{aligned}\underline{w}_{1,2}{}^{(p+1)} = \mathcal{R}ot_x^{l1}(0, \mathcal{R}ot_x^{l2}(0, 1, \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))),\\ \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\end{aligned} \tag{6.25}$$

$$\underline{w}_{1,3}{}^{(p+1)} = \mathcal{R}ot_y^{l2}(0, 1, \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))) \tag{6.26}$$

Thus (6.24) - (6.26) are the CORDIC based normalized versions of the output vector of (6.12) of 3D FastICA.

6.4.3 3D FastICA Component Estimation Stage

After finite number of 3D FastICA Iterations once the normalized vector converges, using (3.6), one independent component is estimated. Here it will be shown that unlike the conventional FastICA algorithm, complete array of multiplication operations can be removed by reusing CORDIC.

Denoting the converged normalized vector by $\underline{\mathbf{w}}_1{}^c = [\underline{w}_{1,1}{}^c \ \underline{w}_{1,2}{}^c \ \underline{w}_{1,3}{}^c]^T$ where superfix "c" stands for *convergence*, denoting the estimated component by $\hat{s}_1 = \{\hat{s}_{1,j}\}$ where $j \in (1, m)$ and m is the frame-length and using the same set of arguments used to derive (6.16) and (6.19), for 3D FastICA (3.6) can explicitly be written as:

$$\begin{aligned}\hat{s}_{1,j} &= z_{1,j}\underline{w}_{1,1}{}^c + z_{2,j}\underline{w}_{1,2}{}^c + z_{3,j}\underline{w}_{1,3}{}^c\\ &= \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c)), \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^c, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c)))\end{aligned} \tag{6.27}$$

Relinquishing "c" from (6.27), $\hat{s}_{1,j}$ can be rewritten as:

$$\hat{s}_{1,j} = \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(w_{1,1}, w_{1,2})), \mathcal{V}ec_\theta^{l2}(w_{1,3}, \mathcal{V}ec_x^{l1}(w_{1,1}, w_{1,2}))) \tag{6.28}$$

where $w_{1,1}$, $w_{1,2}$ and $w_{1,3}$ are the components of the unnormalized vector obtained after 3D FastICA iteration prior to convergence checking. However the reason behind such relinquishment of "c" from (6.28) is deferred until the next section.

6.5 Proposed Architecture for CORDIC based 3D FastICA

Like the 2D architecture as described in Section 6.3, 3D FastICA architecture also has three fundamental stages - *iteration*, *normalization* and *estimation*. As introduced in the last section, each of these stages can further be divided into two "levels" for 3D case. Recalling the definition of "level" from the paragraph following (6.16), it can be said that from the architectural point of view, it is a concept which essentially means that 3D CORDIC FastICA needs two rotation and two vectoring modes for each of the afore-mentioned stages.

6.5.1 3*D* Iteration Mode: Unfolded Architecture

When *level-1* of *iteration* mode is active, the x and y inputs of the Vectoring mode CORDIC are $\underline{w}_{1,1}{}^{(p)}$ and $\underline{w}_{1,2}{}^{(p)}$ respectively as shown in (6.17) where p denotes the iteration number (see Fig. 6.6). The accumulated angle $\theta_{1,p}$ is obtained at the output of the Level-1 Vectoring mode CORDIC which can be connected to the θ-input of the Level-1 Rotation mode CORDIC as shown in Fig. 6.6. The x and y inputs of the Rotation mode CORDIC under this active *level-1 iteration* mode become $z_{1,j}$ and $z_{2,j}$ respectively as also shown in (6.19). These data can be accessed serially from the respective memories. Level-1 Rotation mode y-output is not used and should be ignored (see Fig. 6.6).

When *level-2* of *iteration* mode is active, the x and y inputs of the Vectoring mode CORDIC are $\underline{w}_{1,3}{}^{(p)}$ and the x-output of the Level-1 Vectoring mode CORDIC respectively as shown in (6.18) where p denotes the iteration number (see Fig. 6.6). The accumulated angle $\theta_{2,p}$ is obtained at the output of the Level-2 Vectoring mode CORDIC which can be connected to the θ-input of the Level-1 Rotation mode CORDIC as shown in Fig. 6.6. The x and y inputs of the Rotation mode CORDIC under this active *level-2 iteration* mode become $z_{3,j}$ and the x-output of the Level-1 Rotation mode CORDIC respectively as also shown in (6.19) (see Fig. 6.6). Level-2 Rotation mode y-output is not used and should be ignored.

It can be observed from Fig. 6.6 that Level-1 and Level-2 of Rotation and vectoring modes are connected in Cascaded Feed Forward fashion.

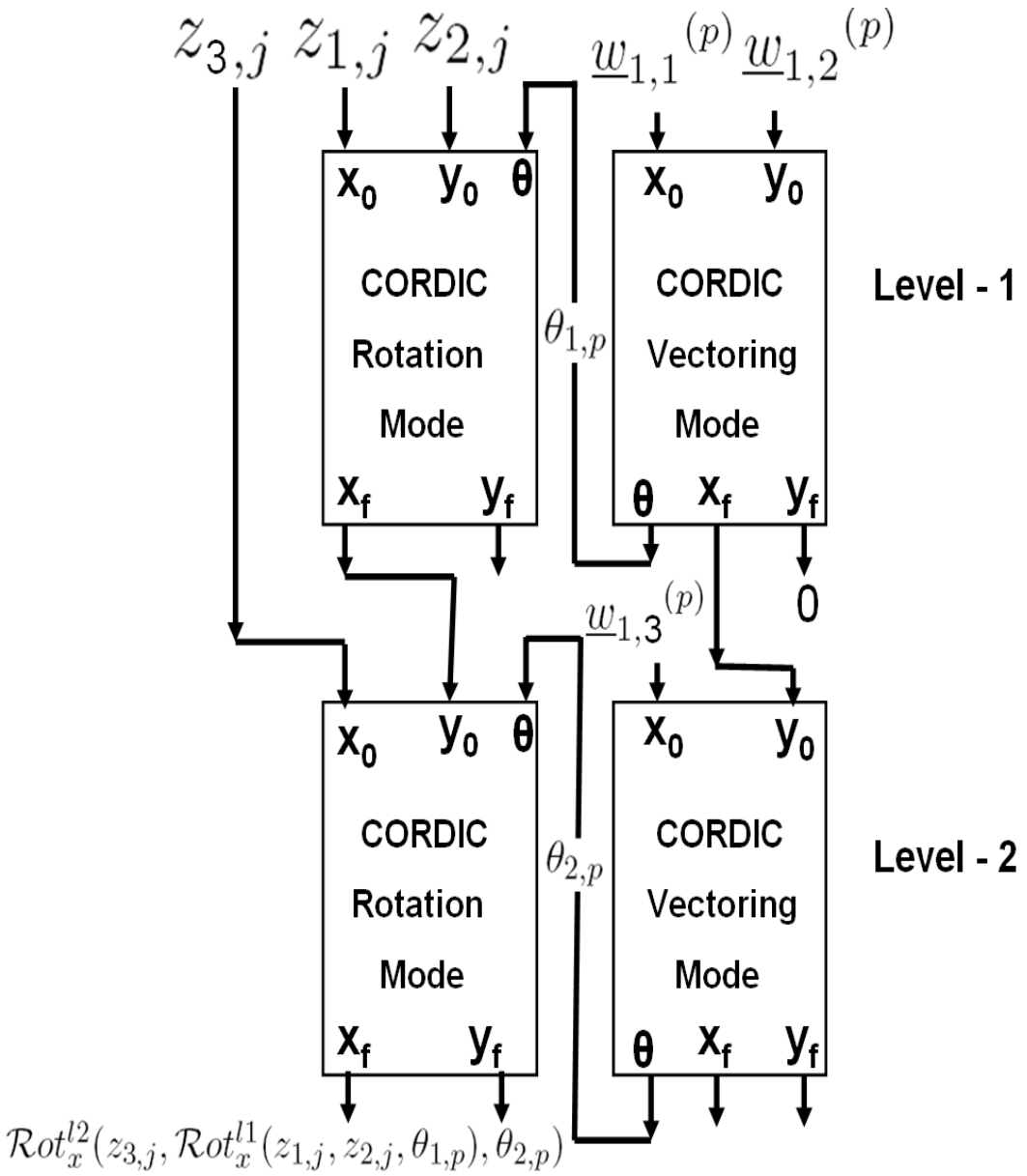


FIGURE 6.6: Iteration stage of the proposed Co-ordinate Rotation based 3-D FastICA.

6.5.2 $3D$ Normalization Mode: Unfolded Architecture

When *level-1 normalization* mode is active, the unnormalized vector components ${w_{1,1}}^{(p+1)}$ and ${w_{1,2}}^{(p+1)}$ are to be considered as the x and y inputs of the Level-1 Vectoring mode CORDIC respectively as shown in (6.23) and in Fig. 6.7. When *level-2 normalization* mode is active, the x-output of the Level-1 Vectoring mode CORDIC will be fed to the y-input of the Level-2 Vectoring mode and ${w_{1,3}}^{(p+1)}$ will be considered as its x-input (please see (6.23) and Fig. 6.7).

Thus for the *normalization* stage, like the iteration stage, the Vectoring modes are connected in feed forward fashion for both the levels as shown in Fig. 6.7.

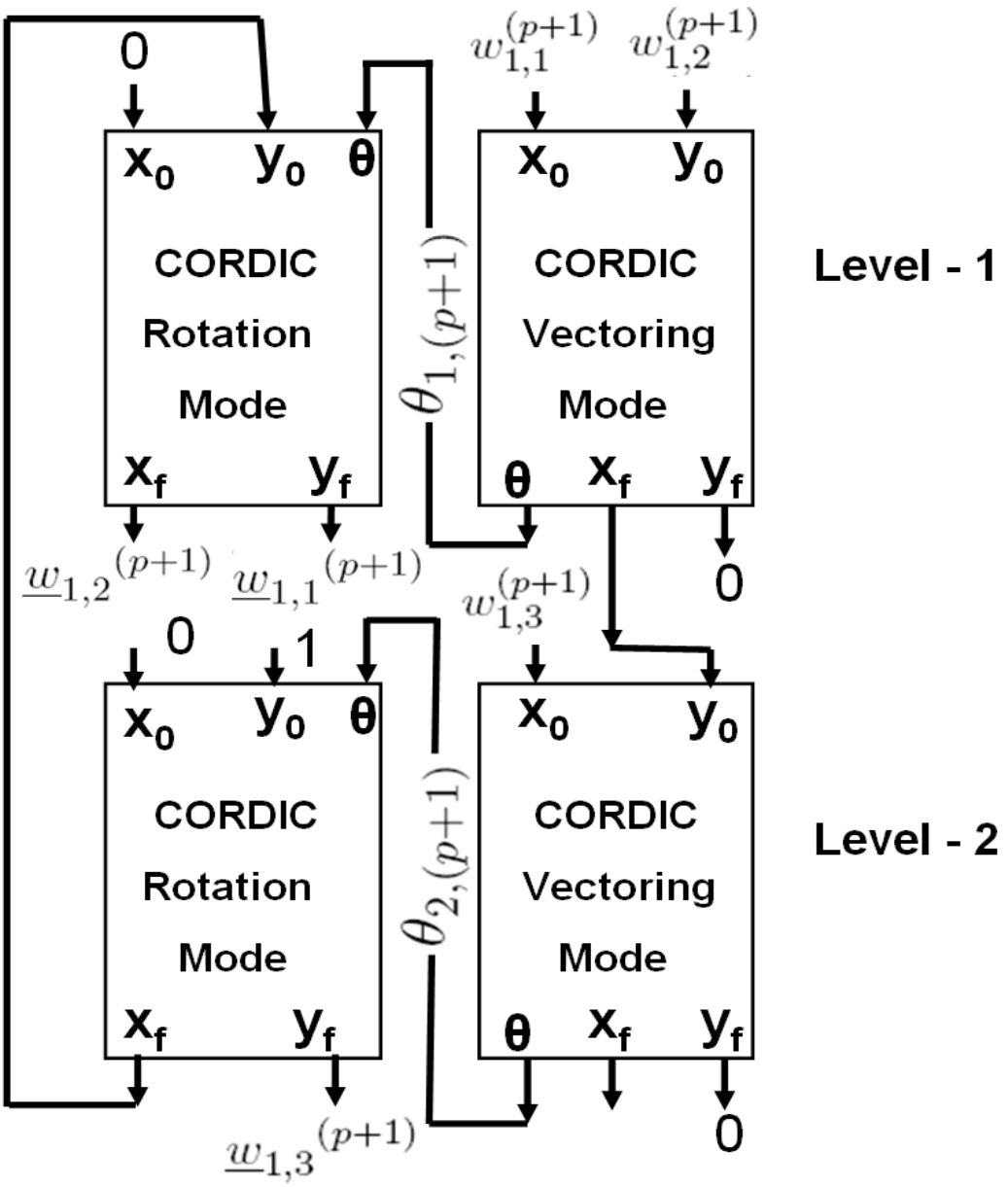


FIGURE 6.7: Normalization stage of the proposed Co-ordinate Rotation based 3-D FastICA.

For the Rotation mode, when *level-2 normalization* mode is active, 0 and 1 are considered as the inputs of the x and y inputs of the Rotation mode CORDIC respectively as shown in Fig. 6.7. The accumulated angle out of the Level-2 Vectoring mode $\theta_{2,(p+1)}$ is considered as the θ-input of this Level-2 Rotation mode CORDIC. The y-output of this Level-2 Rotation mode CORDIC is $\underline{w}_{1,3}{}^{(p+1)}$ as shown in (6.26). The corresponding x-output of this Rotation is fed back as the y-input of the Level-1 Rotation mode CORDIC and its x-input is considered as 0. Its θ-input is $\theta_{1,p}$ which has already been computed during Level-1 Vectoring mode computation as described above. The y and x-outputs of the Level-1 Rotation mode CORDIC is $\underline{w}_{1,1}{}^{(p+1)}$ and $\underline{w}_{1,2}{}^{(p+1)}$ respectively as shown in Fig. 6.7 and in (6.24)-(6.25).

Thus for the *normalization* stage, unlike the iteration stage, the Rotation modes are connected in feed backward fashion for both the levels as shown in Fig. 6.7.

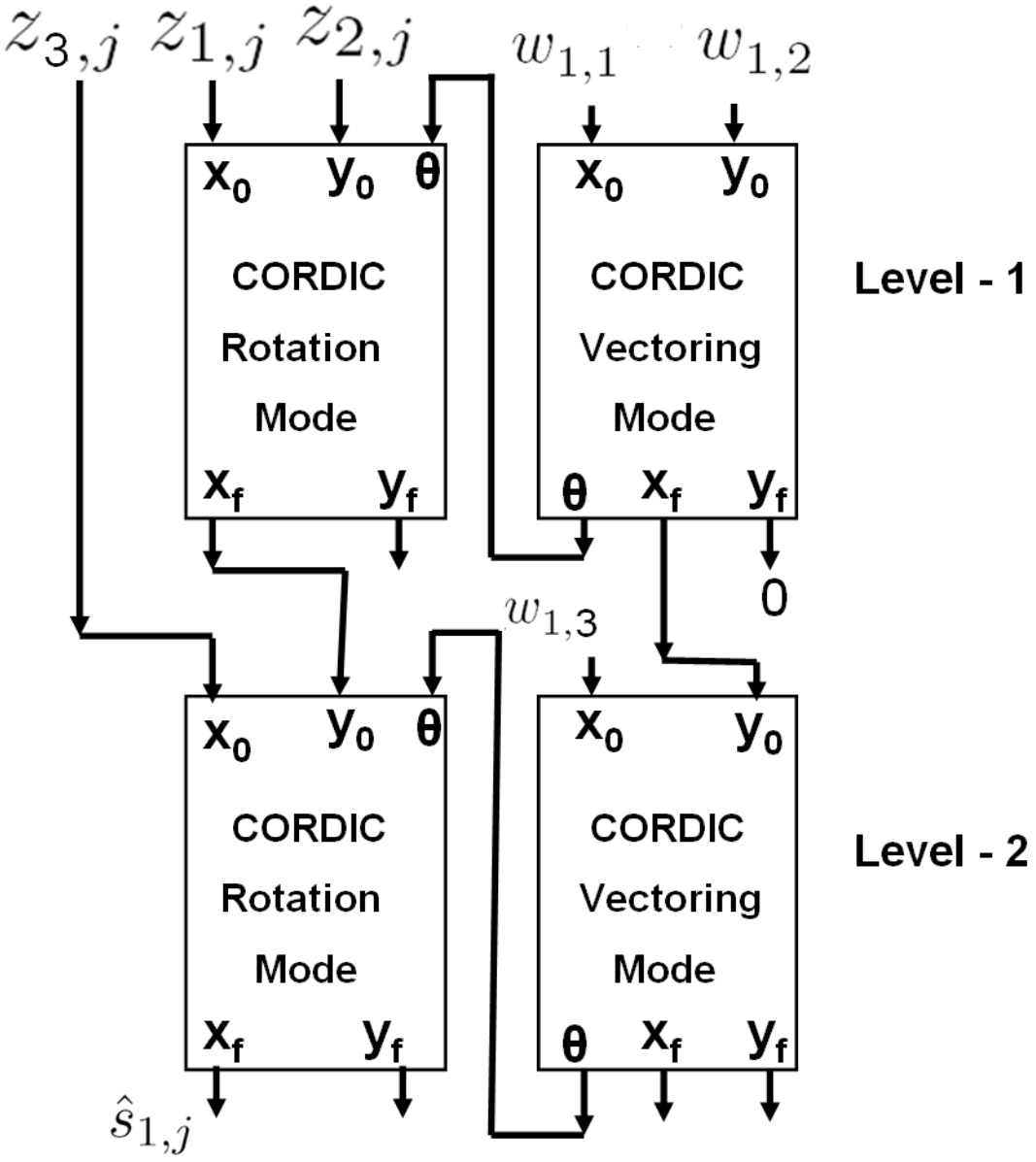


FIGURE 6.8: Estimation stage of the proposed Co-ordinate Rotation based 3-D FastICA.

6.5.3 3*D* Estimation Mode: Unfolded Architecture

When *estimation* mode is active for both of the levels, Vectoring mode CORDIC needs not to be used because the accumulated angle information is already computed during the previous normalization step and can be reused as the θ input of the Rotation mode CORDIC. This is the reason why "c" is relinquished from (6.27) to derive (6.28). Comparing Fig. 6.7 and Fig. 6.8, it can be found that the Vectoring mode inputs are same

and superfix "p" is removed from Fig. 6.8 intentionally for the sake of clarity.

When *level-1 estimation* mode is active, similar to *iteration* mode, in this *estimation* mode also $z_{1,j}$ and $z_{2,j}$ are to be considered as the x and y inputs of the Rotation mode CORDIC and here also the y-output has to be ignored. The θ-input of this Rotation mode CORDIC is the angle computed during the Level-1 Vectoring mode computation. The corresponding x-output of this Level-1 Rotation mode is fed to the y-input of the Level-2 Rotation mode CORDIC when *level-2 estimation* mode is active. The x-output of this Level-2 Rotation mode CORDIC is the estimated output as shown in (6.28) and in Fig. 6.8.

Thus for the *estimation* stage, like the iteration stage, the Rotation modes are connected in feed forward fashion for both the levels as shown in Fig. 6.8.

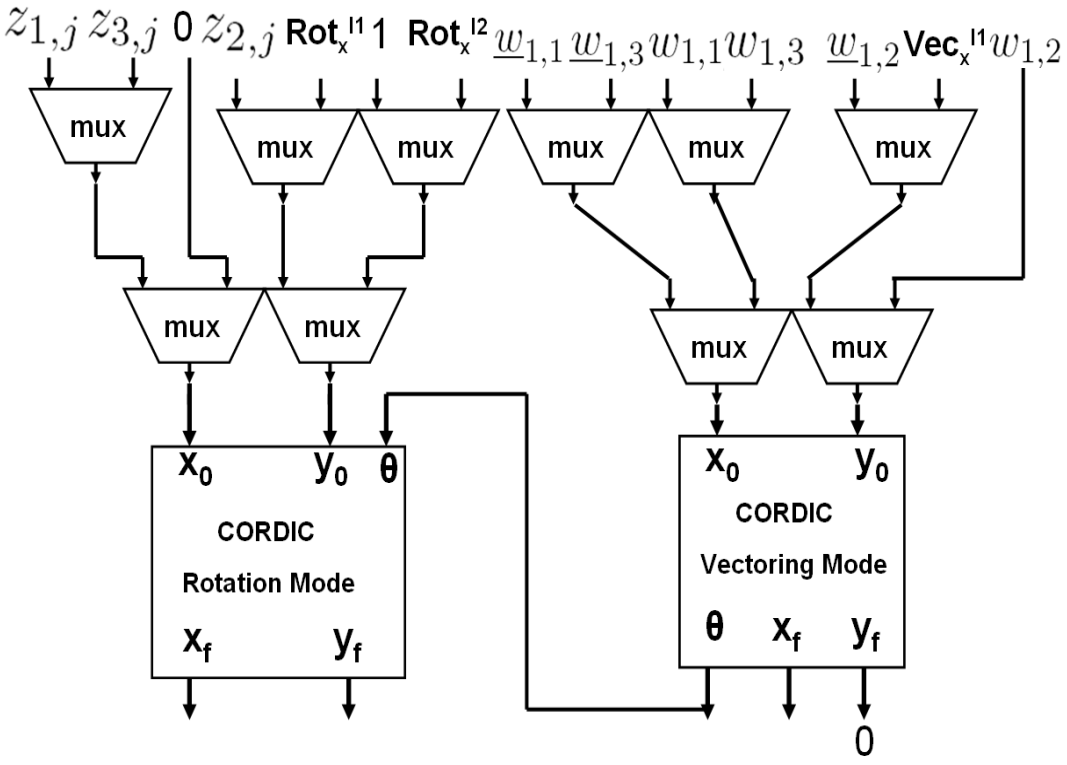


FIGURE 6.9: Architecture of the proposed Co-ordinate Rotation based 3-D FastICA.

6.5.4 Multiplexed Architecture: CORDIC Reuse for $3D$

As mentioned in section 6.3, since iteration, normalization and estimation stages need not to be executed concurrently, same CORDIC unit can be reused for implementing these stages only at the expense of multiplexers at the inputs of the rotation and vectoring mode CORDIC. However unlike $2D$ case, each of these stages within $3D$ FastICA, as introduced in the last section, consists of two levels. But again level-1 and level-2 of each of the three stages are not executed at the same time. Thus for each of these

three stages two levels of CORDIC, as shown in Fig. 6.6, 6.7 and 6.8, can be folded to a single level of CORDIC structure at the expense of multiplexers. This structure is similar to the $2D$ architecture discussed in section 6.3 and thus the same procedure can be adopted here as well to propose the multiplexed architecture for the $3D$ case. Combining the modified multiplexed version of the iteration, normalization and estimation stages, it is possible to model the multiplexers based single CORDIC unit for $3D$ FastICA implementation as shown in Fig. 6.9. It can be noted from Fig. 6.9 that one $2:1$ multiplexer is used at each of the x and y inputs of the Rotation and Vectoring mode CORDIC which is similar to the $2D$ architecture shown in Fig. 6.4 and the top level multiplexer array is used because of the internal level folding within each of the three stages as discussed before. In Fig. 6.9, Rot_x^{l*} and Vec_x^{l*} denote the output of the level $l*$ irrespective of any stage.

6.6 Proposed CORDIC based 4D FastICA Algorithm

6.6.1 4D FastICA Iteration Stage

The case under consideration here in this section is 4D FastICA (i.e. $n = 4$) and using (3.8) its expanded form can be written as:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \\ {w_{1,4}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)} + z_{4,j}{\underline{w}_{1,4}}^{(p)}\}^3] \\ \mathcal{E}[z_{2,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)} + z_{4,j}{\underline{w}_{1,4}}^{(p)}\}^3] \\ \mathcal{E}[z_{3,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)} + z_{4,j}{\underline{w}_{1,4}}^{(p)}\}^3] \\ \mathcal{E}[z_{4,j}\{z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)} + z_{4,j}{\underline{w}_{1,4}}^{(p)}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \\ {\underline{w}_{1,4}}^{(p)} \end{bmatrix} \tag{6.29}$$

where p denotes the number of iteration stage, $z_{i,j}$ represents the i^{th} whitened data containing j number of samples where $i = \{1, 2, 3, 4\}$ and $j \in (1, m)$ where m denotes the frame-length, ${w_{1,q}}^{(p+1)}$ is the 1^{st} column of the unmixing matrix after p^{th} iteration where $q = \{1, 2, 3, 4\}$ and ${\underline{w}_{1,q}}^{(p)}$ indicates the *normalized value* of ${w_{1,q}}^{(p)}$ used in p^{th} iteration as given by (3.10). Without any loss of generality, ${\underline{w}_{1,1}}^{(p)}$, ${\underline{w}_{1,2}}^{(p)}$, ${\underline{w}_{1,3}}^{(p)}$ and ${\underline{w}_{1,4}}^{(p)}$ can be considered as the components of a unit norm vector ${\underline{\mathbf{w}}_1}^{(p)}$.

For the sake of clarity let us introduce a term "G" which is defined as follows:

$$G = z_{1,j}{\underline{w}_{1,1}}^{(p)} + z_{2,j}{\underline{w}_{1,2}}^{(p)} + z_{3,j}{\underline{w}_{1,3}}^{(p)} + z_{4,j}{\underline{w}_{1,4}}^{(p)} \tag{6.30}$$

Using (6.30), (6.29) can be modified as:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \\ {w_{1,4}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{G\}^3] \\ \mathcal{E}[z_{2,j}\{G\}^3] \\ \mathcal{E}[z_{3,j}\{G\}^3] \\ \mathcal{E}[z_{4,j}\{G\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \\ {\underline{w}_{1,4}}^{(p)} \end{bmatrix} \tag{6.31}$$

Now following the procedure used in (6.3) and (6.13), transforming from Cartesian to Polar co-ordinate system, ${\underline{\mathbf{w}}_1}^{(p)}$ can be re-written as:

$$\begin{aligned} {\underline{\mathbf{w}}_1}^{(p)} &= [{\underline{w}_{1,1}}^{(p)} \ {\underline{w}_{1,2}}^{(p)} \ {\underline{w}_{1,3}}^{(p)} \ {\underline{w}_{1,4}}^{(p)}]^T \\ &= \begin{bmatrix} \sin\theta_{3,p}\sin\theta_{2,p}\cos\theta_{1,p} \\ \sin\theta_{3,p}\sin\theta_{2,p}\sin\theta_{1,p} \\ \sin\theta_{3,p}\cos\theta_{2,p} \\ \cos\theta_{3,p} \end{bmatrix} \end{aligned} \tag{6.32}$$

where, the spherical angles $\theta_{1,p}$, $\theta_{2,p}$ and $\theta_{3,p}$ at p^{th} iteration stage, following the same convention used after (6.13), can be defined as

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

and $\tan^{-1}({\underline{w}_{1,4}}^{(p)}/\sqrt{({\underline{w}_{1,1}}^{(p)})^2 + ({\underline{w}_{1,2}}^{(p)})^2 + ({\underline{w}_{1,3}}^{(p)})^2})$ respectively. Since the vector ${\underline{\mathbf{w}}_1}^{(p)}$ is unit norm vector, the magnitude of this vector is one and thus does not appear in (6.32). Using (6.32), (6.30) can be modified to:

$$\begin{aligned} G &= z_{1,j}\sin\theta_{3,p}\sin\theta_{2,p}\cos\theta_{1,p} + z_{2,j}\sin\theta_{3,p}\sin\theta_{2,p}\sin\theta_{1,p} + z_{3,j}\sin\theta_{3,p}\cos\theta_{2,p} + z_{4,j}\cos\theta_{3,p} \\ &= z_{4,j}\cos\theta_{3,p} + (z_{3,j}\cos\theta_{2,p} + (z_{1,j}\cos\theta_{1,p} + z_{2,j}\sin\theta_{1,p})\sin\theta_{2,p})\sin\theta_{3,p}) \end{aligned} \tag{6.33}$$

Recalling the notations of "$\mathcal{R}ot$" and "$\mathcal{V}ec$" introduced in the section 6.2, (6.33) can be rewritten as:

$$G = \mathcal{R}ot_x^{l3}(z_{4,j}, \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \theta_{1,p}), \theta_{2,p}), \theta_{3,p}) \tag{6.34}$$

It is to be noted from (6.34) that the notation "*level*" introduced in (6.16), is also used here. Only notable difference is (6.34) uses one more *level* than the 3D FastICA Iteration stage as shown in (6.16). This essentially means 4D FastICA Iteration stage requires one more Rotation over 3D case.

From (6.35) it can be seen that rotation mode CORDIC can be used in the main 4D FastICA Iterative stage. However like CORDIC based 2D and 3D FastICA, here also explicit angle information of $\theta_{1,p}$, $\theta_{2,p}$ and $\theta_{3,p}$ are not available and have to be derived from $\underline{w}_{1,1}{}^{(p)}$, $\underline{w}_{1,2}{}^{(p)}$, $\underline{w}_{1,3}{}^{(p)}$ and $\underline{w}_{1,4}{}^{(p)}$ instead.

From the line following (6.32) it has already been seen that $\theta_{1,p}$ is the *arctangent* of the ratio of the first two components of the vector $\underline{\mathbf{w}}_1{}^{(p)}$ which is exactly the same as the CORDIC operating in Vectoring mode. Thus without any loss of generality, using the notation "$\mathcal{V}ec$", $\theta_{1,p}$ can be expressed as follows:

$$\theta_{1,p} = \mathcal{V}ec_{\theta}^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}) \tag{6.35}$$

Similarly in the line following (6.32), it has been shown that $\theta_{2,p}$ can be represented as the *arctangent* of the ratio of the third component of $\underline{\mathbf{w}}_1{}^{(p)}$ and the *norm* of the vector defined by the first two components of $\underline{\mathbf{w}}_1{}^{(p)}$ on the $X-Y$ co-ordinate plane. Therefore using "$\mathcal{V}ec$" notation, $\theta_{2,p}$ can be expressed as:

$$\theta_{2,p} = \mathcal{V}ec_{\theta}^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_{x}^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})) \tag{6.36}$$

Similarly it has been shown in the line following (6.32) that $\theta_{3,p}$ can be represented as the *arctangent* of the ratio of the fourth component of $\underline{\mathbf{w}}_1{}^{(p)}$ and the *norm* of the vector defined by the first three components of $\underline{\mathbf{w}}_1{}^{(p)}$ on the $X-Y-Z$ co-ordinate space. Therefore using "$\mathcal{V}ec$" notation, $\theta_{3,p}$ can be expressed as:

$$\theta_{3,p} = \mathcal{V}ec_{\theta}^{l3}(\underline{w}_{1,4}^{(p)}, \mathcal{V}ec_{x}^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_{x}^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}))) \tag{6.37}$$

Using the expressions (6.35) - (6.37) in (6.34) we get:

$$\begin{aligned} G = \mathcal{R}ot_{x}^{l3}(z_{4,j}, \mathcal{R}ot_{x}^{l2}(z_{3,j}, \mathcal{R}ot_{x}^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_{\theta}^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})), \mathcal{V}ec_{\theta}^{l2}(\underline{w}_{1,3}^{(p)}, \\ \mathcal{V}ec_{x}^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)}))), \mathcal{V}ec_{\theta}^{l3}(\underline{w}_{1,4}^{(p)}, \mathcal{V}ec_{x}^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_{x}^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))) \end{aligned} \tag{6.38}$$

Thus (6.38) represents the mapping of the conventional $4D$ FastICA Iterative stage in terms of the rotation and vectoring mode CORDIC.

6.6.2 4D FastICA Normalization Stage

Since the basic FastICA Iteration, as mentioned for 2D and 3D cases earlier, is not a norm-preserving operation, normalization of the newly obtained vector after the iteration

stage is necessary as pointed out in (3.10). This step can also be accomplished using the same CORDIC in the following way.

Denoting the normalized components of the vector obtained after p^{th} iteration using (6.29) by ${\underline{w}_{1,1}}^{(p+1)}$, ${\underline{w}_{1,2}}^{(p+1)}$, ${\underline{w}_{1,3}}^{(p+1)}$ and ${\underline{w}_{1,4}}^{(p+1)}$ the following equation holds:

$$ {\underline{w}_{1,i}}^{(p+1)} = \frac{{w_{1,i}}^{(p+1)}}{\sqrt{|{\mathbf{w}_1}^{(p+1)}|^2}} \tag{6.39} $$

where $i = \{1, 2, 3, 4\}$. Using Cartesian to Polar Co-ordinate transformation following (6.32), these four components can be represented as:

$$ {\underline{\mathbf{w}}_1}^{(p+1)} = \begin{bmatrix} {\underline{w}_{1,1}}^{(p+1)} \\ {\underline{w}_{1,2}}^{(p+1)} \\ {\underline{w}_{1,3}}^{(p+1)} \\ {\underline{w}_{1,4}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)} \cos\theta_{1,(p+1)} \\ \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)} \sin\theta_{1,(p+1)} \\ \sin\theta_{3,(p+1)} \cos\theta_{2,(p+1)} \\ \cos\theta_{3,(p+1)} \end{bmatrix} \tag{6.40} $$

Now, considering $x_0 = 0$, $y_0 = \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)}$ and $\theta = \theta_{1,(p+1)}$ in (6.1), it can be found that $x_f = \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)} \sin\theta_{1,(p+1)}$ and $y_f = \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)} \cos\theta_{1,(p+1)}$. Then considering $x_0 = 0$, $y_0 = \sin\theta_{3,(p+1)}$ and $\theta = \theta_{2,(p+1)}$ in (6.1), it can be found that $y_f = \sin\theta_{3,(p+1)} \cos\theta_{2,(p+1)}$. Again, considering $x_0 = 0$, $y_0 = 1$ and $\theta = \theta_{3,(p+1)}$ in (6.1), it can be found that $y_f = \cos\theta_{3,(p+1)}$. Thus a similarity can be established between this (y_f, x_f) and (6.40) and following the "$\mathcal{R}ot$" notation, (6.40) can be written as:

$$ \begin{bmatrix} {\underline{w}_{1,1}}^{(p+1)} \\ {\underline{w}_{1,2}}^{(p+1)} \\ {\underline{w}_{1,3}}^{(p+1)} \\ {\underline{w}_{1,4}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{R}ot_y^{l1}(0, \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)}, \theta_{1,(p+1)}) \\ \mathcal{R}ot_x^{l1}(0, \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)}, \theta_{1,(p+1)}) \\ \mathcal{R}ot_y^{l2}(0, \sin\theta_{3,(p+1)}, \theta_{2,(p+1)}) \\ \mathcal{R}ot_y^{l3}(0, 1, \theta_{3,(p+1)}) \end{bmatrix} \tag{6.41} $$

Now considering $x_0 = 0$, $y_0 = \sin\theta_{3,(p+1)}$ and $\theta = \theta_{2,(p+1)}$ in (6.1), it can be found that $x_f = \sin\theta_{3,(p+1)} \sin\theta_{2,(p+1)}$. Using this concept and also using the notation used in (6.35), (6.41) can be modified as:

$$ \begin{bmatrix} {\underline{w}_{1,1}}^{(p+1)} \\ {\underline{w}_{1,2}}^{(p+1)} \\ {\underline{w}_{1,3}}^{(p+1)} \\ {\underline{w}_{1,4}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{R}ot_y^{l1}(0, \mathcal{R}ot_x^{l2}(0, \sin\theta_{3,(p+1)}, \theta_{2,(p+1)}), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})) \\ \mathcal{R}ot_x^{l1}(0, \mathcal{R}ot_x^{l2}(0, \sin\theta_{3,(p+1)}, \theta_{2,(p+1)}), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})) \\ \mathcal{R}ot_y^{l2}(0, \sin\theta_{3,(p+1)}, \theta_{2,(p+1)}) \\ \mathcal{R}ot_y^{l3}(0, 1, \theta_{3,(p+1)}) \end{bmatrix} \tag{6.42} $$

Now, considering $x_0 = 0$, $y_0 = 1$ and $\theta = \theta_{3,(p+1)}$ in (6.1), it can be found that $x_f = \sin\theta_{3,(p+1)}$. Using this concept and also using the notation used in (6.36), ${\underline{w}_{1,1}}^{(p+1)}$,

$\underline{w}_{1,2}{}^{(p+1)}$, $\underline{w}_{1,3}{}^{(p+1)}$ and $\underline{w}_{1,4}{}^{(p+1)}$ in (6.42) can be expressed as:

$$\begin{aligned}\underline{w}_{1,1}{}^{(p+1)} = \mathcal{R}ot_y^{l1}(0, \mathcal{R}ot_x^{l2}(0, \mathcal{R}ot_x^{l3}(0, 1, \theta_{3,(p+1)}), \mathcal{V}ec_\theta^{l2} \\ (w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\end{aligned} \tag{6.43}$$

$$\begin{aligned}\underline{w}_{1,2}{}^{(p+1)} = \mathcal{R}ot_x^{l1}(0, \mathcal{R}ot_x^{l2}(0, \mathcal{R}ot_x^{l3}(0, 1, \theta_{3,(p+1)}), \mathcal{V}ec_\theta^{l2} \\ (w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\end{aligned} \tag{6.44}$$

$$\underline{w}_{1,3}{}^{(p+1)} = \mathcal{R}ot_y^{l2}(0, \mathcal{R}ot_x^{l3}(0, 1, \theta_{3,(p+1)}), \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))) \tag{6.45}$$

$$\underline{w}_{1,4}{}^{(p+1)} = \mathcal{R}ot_y^{l3}(0, 1, \theta_{3,(p+1)}) \tag{6.46}$$

Now using (6.37), (6.43) - (6.46) can be modified as follows:

$$\begin{aligned}\underline{w}_{1,1}{}^{(p+1)} = \mathcal{R}ot_y^{l1}(0, \mathcal{R}ot_x^{l2}(0, \mathcal{R}ot_x^{l3}(0, 1, \mathcal{V}ec_\theta^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_x^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})))), \\ \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\end{aligned} \tag{6.47}$$

$$\begin{aligned}\underline{w}_{1,2}{}^{(p+1)} = \mathcal{R}ot_x^{l1}(0, \mathcal{R}ot_x^{l2}(0, \mathcal{R}ot_x^{l3}(0, 1, \mathcal{V}ec_\theta^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_x^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})))), \\ \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))), \mathcal{V}ec_\theta^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))\end{aligned} \tag{6.48}$$

$$\begin{aligned}\underline{w}_{1,3}{}^{(p+1)} = \mathcal{R}ot_y^{l2}(0, \mathcal{R}ot_x^{l3}(0, 1, \mathcal{V}ec_\theta^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_x^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1} \\ (w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})))), \mathcal{V}ec_\theta^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})))\end{aligned} \tag{6.49}$$

$$\underline{w}_{1,4}{}^{(p+1)} = \mathcal{R}ot_y^{l3}(0, 1, \mathcal{V}ec_\theta^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_x^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_x^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})))) \tag{6.50}$$

It is to be noted from (6.47) - (6.50) that the explicit information of $\theta_{1,(p+1)}$, $\theta_{2,(p+1)}$ and $\theta_{3,(p+1)}$ are unavailable and that is why it was necessary to use the expressions (6.35) -

(6.37) in above equations.

Thus (6.47) - (6.50) are the CORDIC based normalized versions of the output vector of (6.29) of 4D FastICA.

6.6.3 4D FastICA Component Estimation Stage

After finite number of 4D FastICA Iterations once the normalized vector converges, using (3.6), one independent component is estimated. Here it will be shown that unlike the conventional FastICA algorithm, complete array of multiplication operations can be removed by reusing CORDIC.

For 4D FastICA Component Estimation stage, denoting the estimated component by $\hat{s}_1 = \{\hat{s}_{1,j}\}$ where $j \in (1, m)$ and m is the frame-length, (3.6) can explicitly be written as:

$$\hat{s}_{1,j} = z_{1,j}\underline{w}_{1,1}{}^{(p)} + z_{2,j}\underline{w}_{1,2}{}^{(p)} + z_{3,j}\underline{w}_{1,3}{}^{(p)} + z_{4,j}\underline{w}_{1,4}{}^{(p)} \quad (6.51)$$

Denoting the converged normalized vector by $\underline{\mathbf{w}}_1{}^c = [\underline{w}_{1,1}{}^c \ \underline{w}_{1,2}{}^c \ \underline{w}_{1,3}{}^c \ \underline{w}_{1,4}{}^c]^T$ where superfix "c" stands for *convergence*, and using the same set of arguments used to derive (6.34) and (6.38), for 4D FastICA (6.51) can be rewritten as:

$$\begin{aligned}\hat{s}_{1,j} = \mathcal{R}ot_x^{l3}(z_{4,j}, \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c)), \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^c, \\ \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c))), \mathcal{V}ec_\theta^{l3}(\underline{w}_{1,4}^c, \mathcal{V}ec_x^{l2}(\underline{w}_{1,3}^c, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^c, \underline{w}_{1,2}^c))))\end{aligned} \quad (6.52)$$

Relinquishing "c" from (6.52), $\hat{s}_{1,j}$ can be modified as:

$$\begin{aligned}\hat{s}_{1,j} = \mathcal{R}ot_x^{l3}(z_{4,j}, \mathcal{R}ot_x^{l2}(z_{3,j}, \mathcal{R}ot_x^{l1}(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta^{l1}(w_{1,1}, w_{1,2})), \mathcal{V}ec_\theta^{l2}(w_{1,3}, \\ \mathcal{V}ec_x^{l1}(w_{1,1}, w_{1,2}))), \mathcal{V}ec_\theta^{l3}(w_{1,4}, \mathcal{V}ec_x^{l2}(w_{1,3}, \mathcal{V}ec_x^{l1}(w_{1,1}, w_{1,2}))))\end{aligned} \quad (6.53)$$

where $w_{1,1}$, $w_{1,2}$, $w_{1,3}$ and $w_{1,4}$ are the components of the unnormalized vector obtained after 4D FastICA iteration prior to convergence checking. However the reason behind such relinquishment of "c" from (6.53) is deferred until the next section.

6.7 Proposed Architecture for CORDIC based 4D FastICA

Like the 2D and 3D architectures as described in Section 6.3 and 6.5, 4D FastICA architecture also has three fundamental stages - *iteration*, *normalization* and *estimation*. As mentioned in Section 6.5, each of these stages can further be divided into number of "levels" for 4D case. Recalling from the paragraph following (6.34), it can be said that 4D CORDIC FastICA needs three rotation and three vectoring levels for each of the afore-mentioned stages. These modes will be discussed below in detail.

6.7.1 4*D* Iteration Mode: Unfolded Architecture

When *level-1* of *iteration* mode is active, the x and y inputs of the Vectoring mode CORDIC are ${\underline{w}_{1,1}}^{(p)}$ and ${\underline{w}_{1,2}}^{(p)}$ respectively as shown in (6.35) where p denotes the iteration number. The accumulated angle $\theta_{1,p}$ is obtained at the output of the Level-1 Vectoring mode CORDIC which can be connected to the θ-input of the Level-1 Rotation mode CORDIC as shown in Fig. 6.10. The x and y inputs of the Rotation mode CORDIC under this active *level-1 iteration* mode become $z_{1,j}$ and $z_{2,j}$ respectively as also shown in (6.34). These data can be accessed serially from the respective memories. Level-1 Rotation mode y-output is not used and should be ignored.

When *level-2* of *iteration* mode is active, the x and y inputs of the Vectoring mode CORDIC are ${\underline{w}_{1,3}}^{(p)}$ and the x-output of the Level-1 Vectoring mode CORDIC respectively as shown in (6.36) where p denotes the iteration number. The accumulated angle $\theta_{2,p}$ is obtained at the output of the Level-2 Vectoring mode CORDIC which can be connected to the θ-input of the Level-2 Rotation mode CORDIC as shown in Fig. 6.10. The x and y inputs of the Rotation mode CORDIC under this active *level-2 iteration* mode become $z_{3,j}$ and the x-output of the Level-1 Rotation mode CORDIC respectively as also shown in (6.34). Level-2 Rotation mode y-output is not used and should be ignored.

When *level-3* of *iteration* mode is active, the x and y inputs of the Vectoring mode CORDIC are ${\underline{w}_{1,4}}^{(p)}$ and the x-output of the Level-2 Vectoring mode CORDIC respectively as shown in (6.37) where p denotes the iteration number. The accumulated angle $\theta_{3,p}$ is obtained at the output of the Level-3 Vectoring mode CORDIC which can be connected to the θ-input of the Level-3 Rotation mode CORDIC as shown in Fig. 6.10. The x and y inputs of the Rotation mode CORDIC under this active *level-3 iteration* mode become $z_{4,j}$ and the x-output of the Level-2 Rotation mode CORDIC respectively as also shown in (6.34). Level-3 Rotation mode y-output is not used and should be ignored.

It can be observed from Fig. 6.10 that Level-1, Level-2 and Level-3 of Rotation and

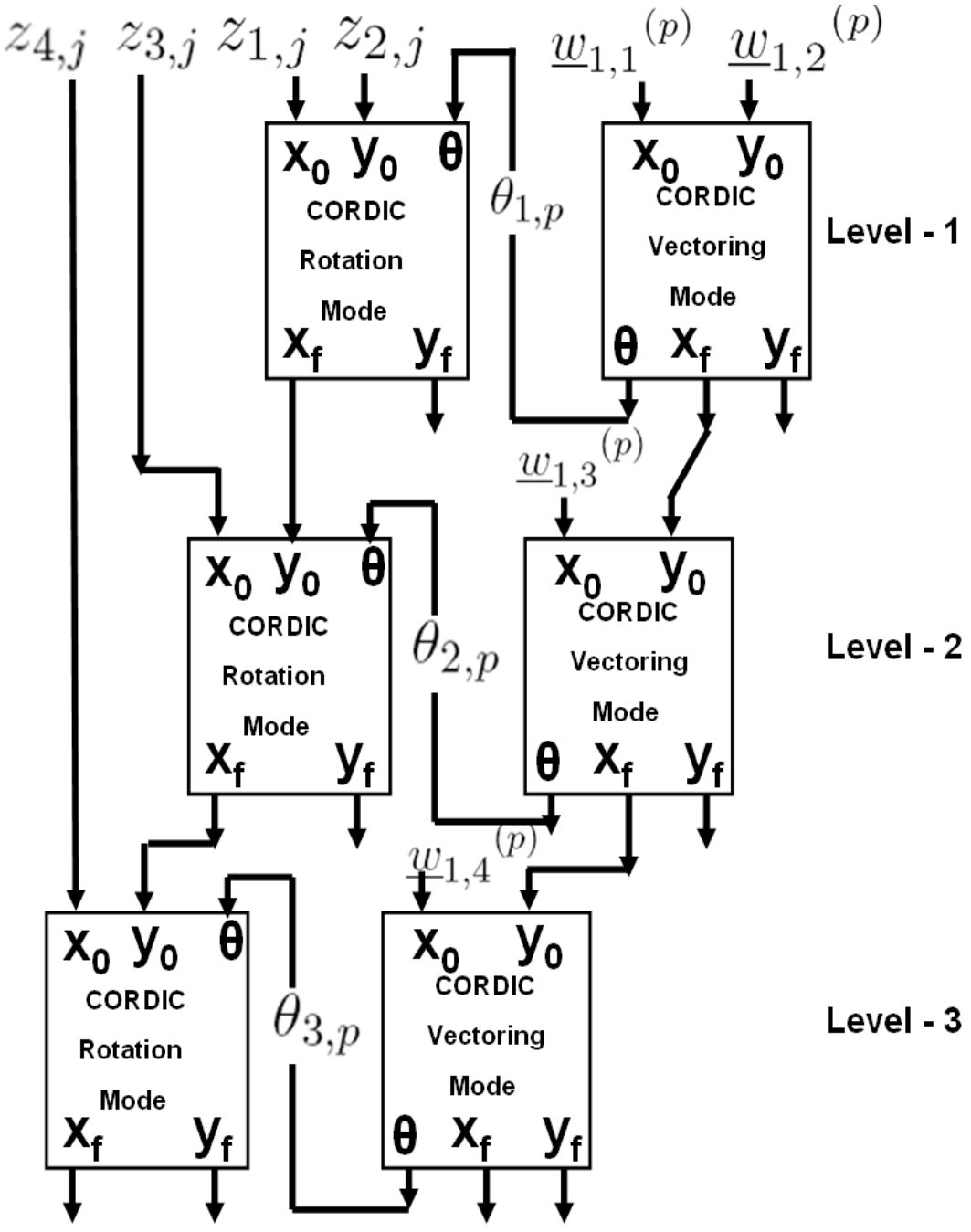


FIGURE 6.10: Iteration stage of the proposed Co-ordinate Rotation based 4-D FastICA.

Vectoring modes are connected in Cascaded Feed Forward fashion.

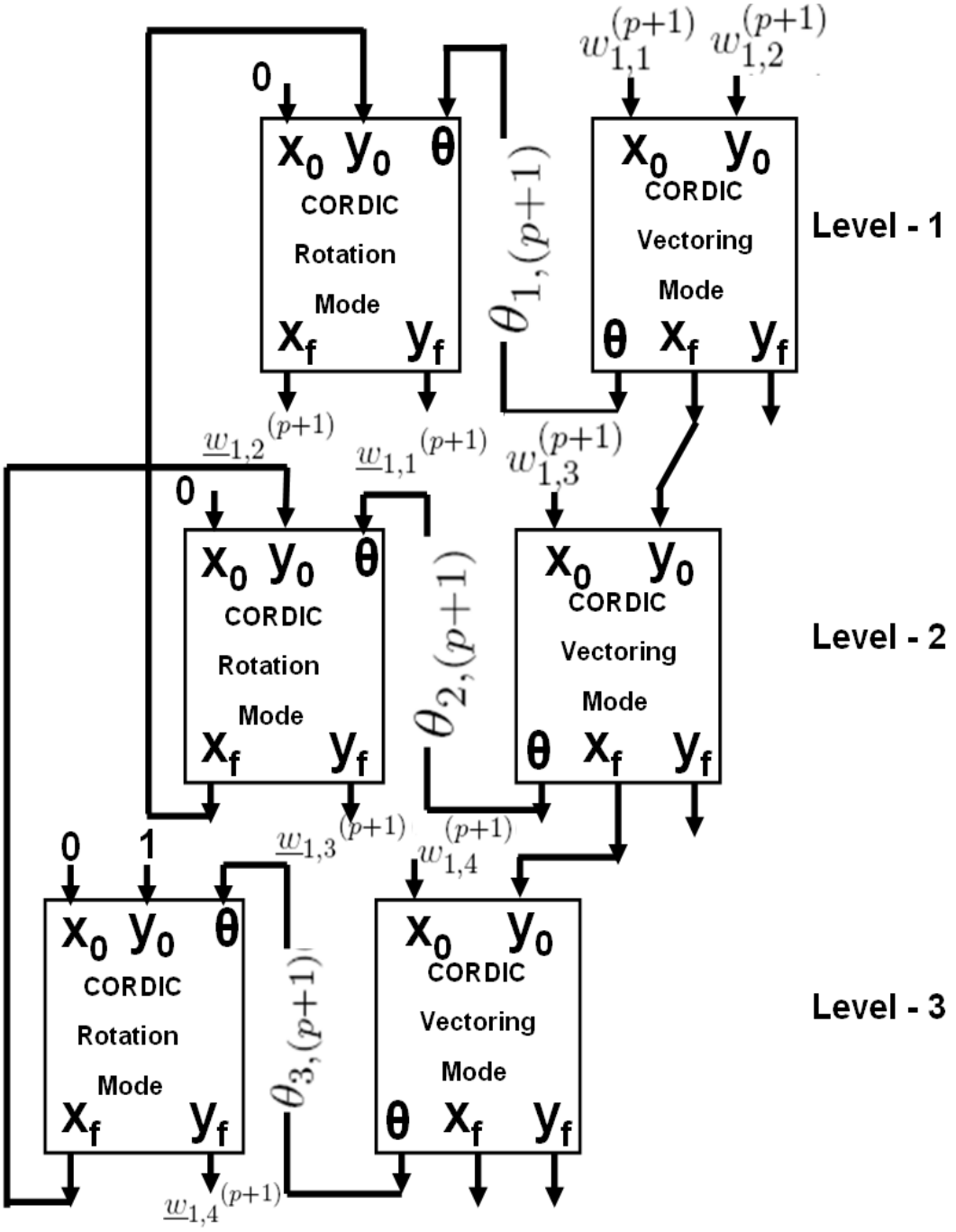


FIGURE 6.11: Normalization stage of the proposed Co-ordinate Rotation based 4-D FastICA.

6.7.2 *4D* Normalization Mode: Unfolded Architecture

When *level-1 normalization* mode is active, the unnormalized vector components ${w_{1,1}}^{(p+1)}$ and ${w_{1,2}}^{(p+1)}$ are to be considered as the x and y inputs of the Level-1 Vectoring mode

CORDIC respectively as shown in (6.42)-(6.42) and in Fig. 6.11. When *level-2 normalization* mode is active, the x-output of the Level-1 Vectoring mode CORDIC will be fed to the y-input of the Level-2 Vectoring mode and ${w_{1,3}}^{(p+1)}$ will be considered as its x-input (please see (6.42)). Similarly when *level-3 normalization* mode is active, the x-output of the Level-2 Vectoring mode CORDIC will be fed to the y-input of the Level-3 Vectoring mode and ${w_{1,4}}^{(p+1)}$ will be considered as its x-input (please see (6.42)).

Thus for the *normalization* stage, like the iteration stage, the Vectoring modes are connected in feed forward fashion for all the three levels as shown in Fig. 6.11.

For the Rotation mode, when *level-3 normalization* mode is active, 0 and 1 are considered as the inputs of the x and y inputs of the Rotation mode CORDIC respectively as shown in Fig. 6.11. The accumulated angle out of the Level-3 Vectoring mode $\theta_{3,(p+1)}$ is considered as the θ-input of this Level-3 Rotation mode CORDIC. The y-output of this Level-3 Rotation mode CORDIC is ${\underline{w}_{1,4}}^{(p+1)}$ as shown in (6.50). The corresponding x-output of this Rotation is fed back as the y-input of the Level-2 Rotation mode CORDIC and its x-input is considered as 0. Its θ-input is $\theta_{2,p}$ which has already been computed during Level-2 Vectoring mode computation as described above. The y-output of the Level-2 Rotation mode CORDIC is ${\underline{w}_{1,3}}^{(p+1)}$ as shown in Fig. 6.11 and in (6.49). The corresponding x-output of this Rotation is fed back as the y-input of the Level-1 Rotation mode CORDIC and its x-input is considered as 0. Its θ-input is $\theta_{1,p}$ which has already been computed during Level-1 Vectoring mode computation as described before. The y and x-outputs of the Level-1 Rotation mode CORDIC is ${\underline{w}_{1,1}}^{(p+1)}$ and ${\underline{w}_{1,2}}^{(p+1)}$ respectively as shown in Fig. 6.11 and in (6.47)-(6.48).

Thus for the *normalization* stage, unlike the iteration stage, the Rotation modes are connected in feed backward fashion for all three levels as shown in Fig. 6.11.

6.7.3 $4D$ Estimation Mode: Unfolded Architecture

When *estimation* mode is active for any of the three levels, Vectoring mode CORDIC needs not to be used because the accumulated angle information is already computed during the previous normalization step and can be reused as the θ input of the Rotation mode CORDIC. This is the reason why "c" is relinquished from (6.52) to derive (6.53). Comparing Fig. 6.11 and Fig. 6.12, it can be found that the Vectoring mode inputs are same and superfix "p" is removed from Fig. 6.12 intentionally for the sake of brevity.

When *level-1 estimation* mode is active, similar to *iteration* mode, in this *estimation* mode also $z_{1,j}$ and $z_{2,j}$ are to be considered as the x and y inputs of the Rotation mode CORDIC and here also the y-output has to be ignored. The θ-input of this Rotation mode CORDIC is the angle computed during the Level-1 Vectoring mode computation. The corresponding x-output of this Level-1 Rotation mode is fed to the y-input of the Level-2 Rotation mode CORDIC when *level-2 estimation* mode is active. The x-output

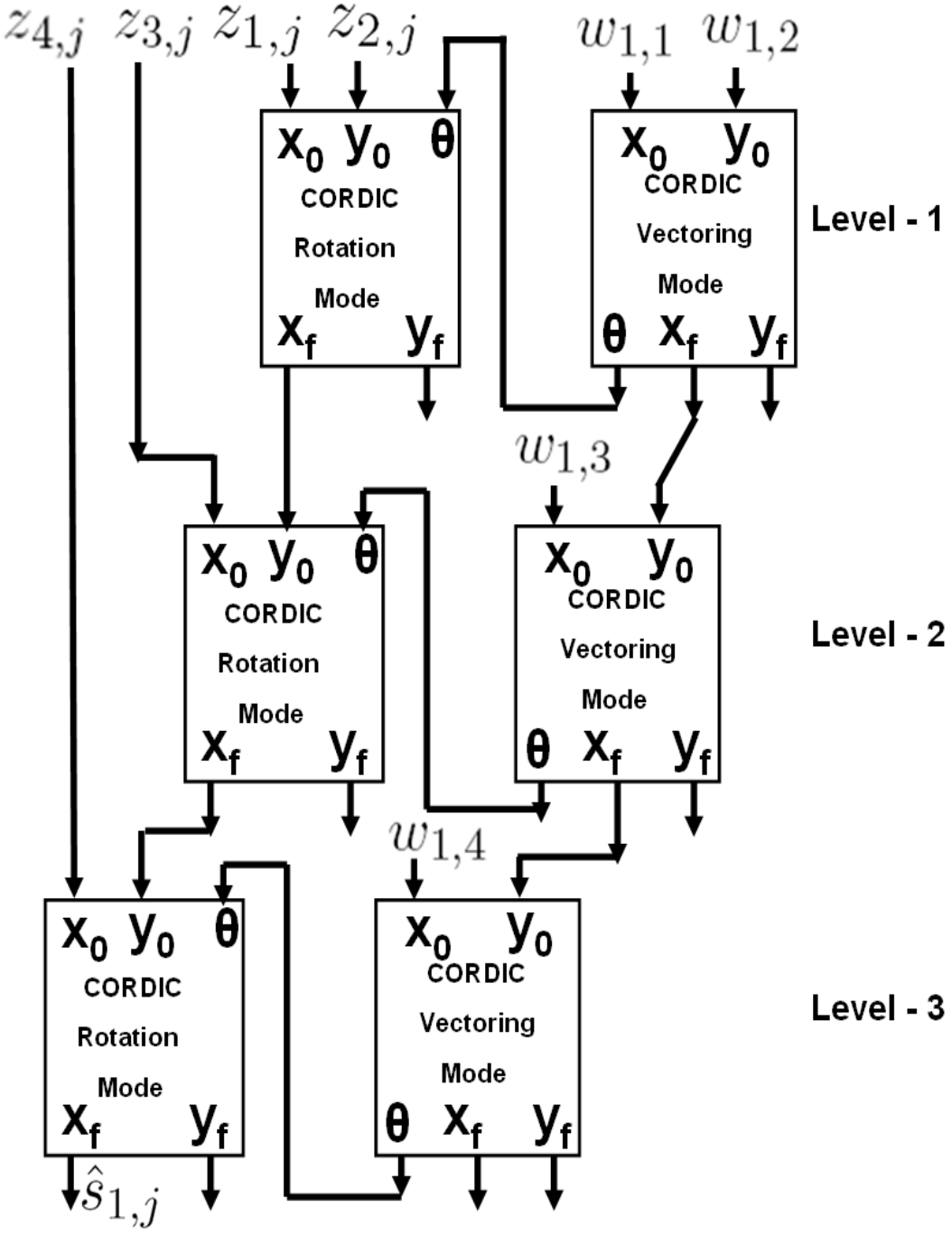


FIGURE 6.12: Estimation stage of the proposed Co-ordinate Rotation based 4-D FastICA.

of this Level-2 Rotation mode CORDIC is the estimated output as shown in (6.53) and in Fig. 6.12.

Thus for the *estimation* stage, like the iteration stage, the Rotation modes are connected in feed forward fashion for both the levels as shown in Fig. 6.12.

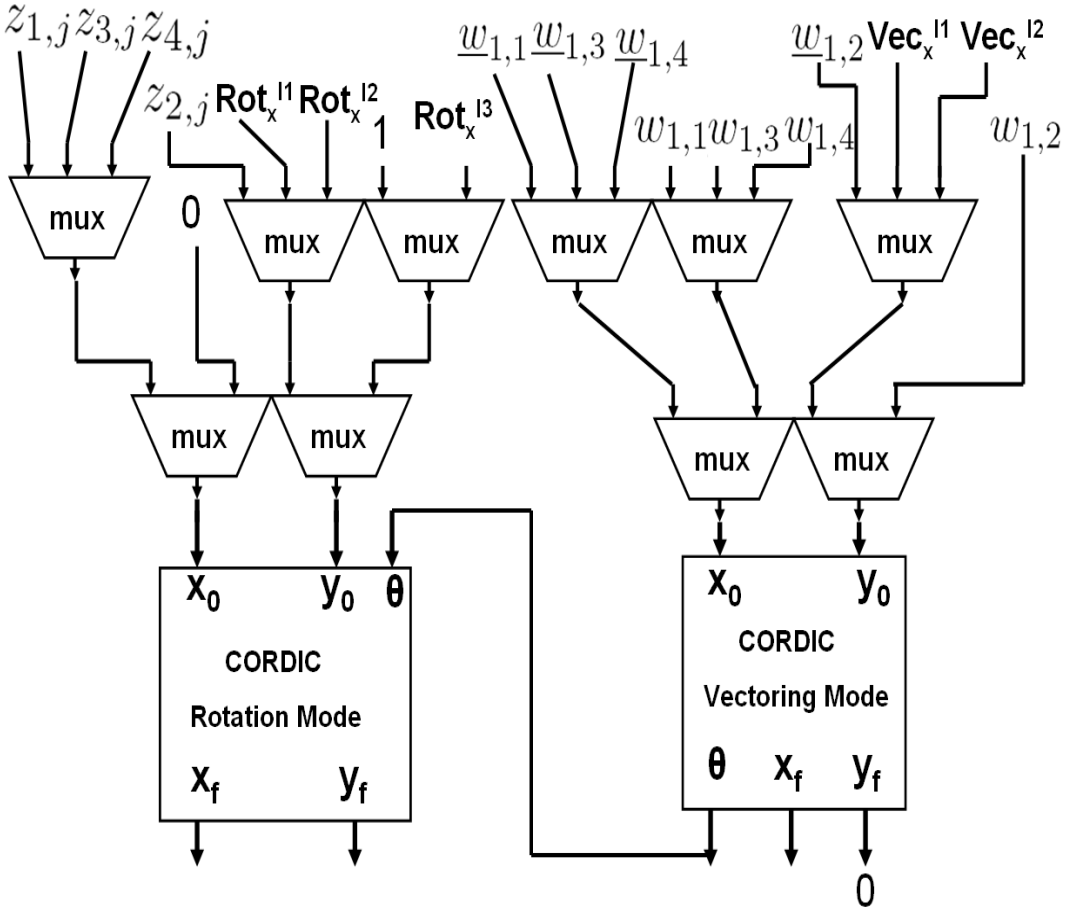


FIGURE 6.13: Architecture of the proposed Co-ordinate Rotation based 4-D FastICA.

6.7.4 Multiplexed Architecture: CORDIC Reuse for $4D$

As mentioned in section 6.3, since iteration, normalization and estimation stages need not to be executed concurrently, same CORDIC unit can be reused for implementing these stages only at the expense of multiplexers at the inputs of the rotation and vectoring mode CORDIC. However unlike $2D$ case, each of these stages within $4D$ FastICA, as mentioned in the last section, consists of three levels. But again level-1, level-2 and level-3 of each of the three stages are not executed in parallel. Thus for each of these three stages three levels of CORDIC, as shown in Fig. 6.10, 6.11 and 6.12, can be folded to a single level of CORDIC structure at the expense of multiplexers. Like $3D$ case as mentioned in section 6.5, this structure is also similar to the $2D$ architecture discussed in section 6.3 and thus the same procedure can be adopted here as well to propose the

multiplexed architecture for the $4D$ case. Combining the modified multiplexed version of the iteration, normalization and estimation stages, it is possible to model the multiplexers based single CORDIC unit for $4D$ FastICA implementation as shown in Fig. 6.13. It can be noted from Fig. 6.13 that one $2:1$ multiplexer is used at each of the x and y inputs of the Rotation and Vectoring mode CORDIC which is similar to the $2D$ architecture shown in Fig. 6.4 and the top level multiplexer array is used because of the internal level folding within each of the three stages as discussed before. In Fig. 6.13, Rot_x^{l*} and Vec_x^{l*} denote the output of the level $l*$ irrespective of any stage information.

6.8 Proposed CORDIC based Generalized nD FastICA Algorithm

6.8.1 nD FastICA Iteration Stage

To derive generalized algorithm for the CORDIC based nD FastICA Iteration stage, let us first recall the CORDIC based formulation of 2D FastICA Iteration stage from (6.6) in section 6.2:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta({\underline{w}_{1,1}}^{(p)}, {\underline{w}_{1,2}}^{(p)}))\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta({\underline{w}_{1,1}}^{(p)}, {\underline{w}_{1,2}}^{(p)}))\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \end{bmatrix} \tag{6.54}$$

Let us now introduce a new term - $\mathcal{R}_{x,j}^{2D}$ and define it as follows:

$$\mathcal{R}_{x,j}^{2D} = \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_\theta({\underline{w}_{1,1}}^{(p)}, {\underline{w}_{1,2}}^{(p)})) \tag{6.55}$$

It may be read as "x-output corresponding to the j^{th} input of the Rotation mode 2D CORDIC". Using (6.55) in (6.54) we get:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}_{x,j}^{2D}\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}_{x,j}^{2D}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \end{bmatrix} \tag{6.56}$$

Similarly let us introduce the following terms for the Vectoring mode for 2D CORDIC:

$$\mathcal{V}_\theta^{2D} = \mathcal{V}ec_\theta({\underline{w}_{1,1}}^{(p)}, {\underline{w}_{1,2}}^{(p)}) \tag{6.57}$$

$$\mathcal{V}_x^{2D} = \mathcal{V}ec_x({\underline{w}_{1,1}}^{(p)}, {\underline{w}_{1,2}}^{(p)}) \tag{6.58}$$

Now it is interesting to note that (6.19) can be rewritten using (6.55) as follows:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta^{l2}(\underline{w}_{1,3}^{(p)}, \mathcal{V}ec_x^{l1}(\underline{w}_{1,1}^{(p)}, \underline{w}_{1,2}^{(p)})))\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \end{bmatrix} \tag{6.59}$$

Now it can be observed from (6.59) that it can further be modified using (6.58) as follows:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta(\underline{w}_{1,3}^{(p)}, \mathcal{V}_x^{2D}))\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta(\underline{w}_{1,3}^{(p)}, \mathcal{V}_x^{2D}))\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta(\underline{w}_{1,3}^{(p)}, \mathcal{V}_x^{2D}))\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \end{bmatrix} \tag{6.60}$$

Now at this juncture it is important to note that the way (6.54) was written as (6.56) for 2D case, (6.60) can also be written as follows for 3D case without any loss of any generality:

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}_{x,j}^{3D}\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}_{x,j}^{3D}\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}_{x,j}^{3D}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \end{bmatrix} \tag{6.61}$$

where, following the similarity as (6.55), a new term - $\mathcal{R}_{x,j}^{3D}$ is introduced which can be defined as:

$$\mathcal{R}_{x,j}^{3D} = \mathcal{R}ot_x(z_{3,j}, \mathcal{R}_{x,j}^{2D}, \mathcal{V}ec_\theta(\underline{w}_{1,3}^{(p)}, \mathcal{V}_x^{2D})) \tag{6.62}$$

Subsequently following the similar notations used in (6.57) and (6.58) for 2D case, for 3D case following notations hold:

$$\mathcal{V}_x^{3D} = \mathcal{V}ec_x(\underline{w}_{1,3}^{(p)}, \mathcal{V}_x^{2D}) \tag{6.63}$$

$$\mathcal{V}_\theta^{3D} = \mathcal{V}ec_\theta(\underline{w}_{1,3}^{(p)}, \mathcal{V}_x^{2D}) \tag{6.64}$$

Similarly (6.38) can be rewritten as follows using (6.62) and (6.63):

$$\begin{aligned} G &= \mathcal{R}ot_x(z_{4,j}, \mathcal{R}^{3D}_{x,j}, \mathcal{V}ec^{l3}_{\theta}(\underline{w}^{(p)}_{1,4}, \mathcal{V}ec^{l2}_{x}(\underline{w}^{(p)}_{1,3}, \mathcal{V}ec^{l1}_{x}(\underline{w}^{(p)}_{1,1}, \underline{w}^{(p)}_{1,2})))) \\ &= \mathcal{R}ot_x(z_{4,j}, \mathcal{R}^{3D}_{x,j}, \mathcal{V}ec_{\theta}(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_{x})) \end{aligned} \tag{6.65}$$

Now noting the way (6.62) - (6.64) were framed for 3D case, here also for 4D case following equations hold:

$$\mathcal{R}^{4D}_{x,j} = \mathcal{R}ot_x(z_{4,j}, \mathcal{R}^{3D}_{x,j}, \mathcal{V}ec_{\theta}(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_{x})) \tag{6.66}$$

$$\mathcal{V}^{4D}_{x} = \mathcal{V}ec_x(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_{x}) \tag{6.67}$$

$$\mathcal{V}^{4D}_{\theta} = \mathcal{V}ec_{\theta}(\underline{w}^{(p)}_{1,4}, \mathcal{V}^{3D}_{x}) \tag{6.68}$$

Now at this juncture, it is important to note the similarity of involved computations among different dimensions of FastICA Iteration stage. For example, considering (6.55) and (6.62) for 2D and 3D cases respectively, it is worth noting that (6.62) can be realised using (6.55) recursively. Similarly, comparing (6.62) and (6.66) for 3D and 4D cases respectively it can be observed that (6.66) can be realized using (6.62) recursively. Since (6.62) involves recursive usage of (6.55), (6.66) thus can be realized using (6.55) recursively. Moreover, (6.55) involves only one CORDIC *Rotation* and one *vectoring* in 2D plane. So, this essentially mean that if the same procedure is followed, any higher dimensional FastICA Iteration stage can be realized using the 2D CORDIC Rotation and vectoring. Thus it is possible to propose a generalized theorem which may provide the theoretical basis of nD FastICA Iteration stage realization with the recursive use of its immediate lower dimension and ultimately boils down to the 2D CORDIC *Rotation* and *Vectoring* mode.

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

$$\begin{bmatrix} {w_{1,1}}^{(p+1)} \\ {w_{1,2}}^{(p+1)} \\ {w_{1,3}}^{(p+1)} \\ . \\ . \\ . \\ {w_{1,n}}^{(p+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{E}[z_{1,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \\ \mathcal{E}[z_{2,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \\ \mathcal{E}[z_{3,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \\ . \\ . \\ . \\ \mathcal{E}[z_{n,j}\{\mathcal{R}^{nD}_{x,j}\}^3] \end{bmatrix} - 3 \begin{bmatrix} {\underline{w}_{1,1}}^{(p)} \\ {\underline{w}_{1,2}}^{(p)} \\ {\underline{w}_{1,3}}^{(p)} \\ . \\ . \\ . \\ {\underline{w}_{1,n}}^{(p)} \end{bmatrix} \tag{6.69}$$

where, considering $\mathcal{R}^{2D}_{x,j} = \mathcal{R}ot_x(z_{1,j}, z_{2,j}, \mathcal{V}ec_{\theta}({\underline{w}_{1,1}}^{(p)}, {\underline{w}_{1,2}}^{(p)}))$ as the basic Rotation

mode (as obtained from (6.55)) and $\mathcal{V}_{\theta}^{2D} = \mathcal{V}ec_{\theta}(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)}); \mathcal{V}_{x}^{2D} = \mathcal{V}ec_{x}(\underline{w}_{1,1}{}^{(p)}, \underline{w}_{1,2}{}^{(p)})$ as the basic Vectoring mode (as obtained from (6.57) and (6.58)) of CORDIC operation, for $n \geq 3$, Iteration stage of the nD FastICA algorithm can be expressed in recursive way as follows:

(i)

$$\mathcal{R}_{x,j}^{nD} = \mathcal{R}ot_{x}(z_{n,j}, \mathcal{R}_{x,j}^{(n-1)D}, \mathcal{V}ec_{\theta}(\underline{w}_{1,n}^{(p)}, \mathcal{V}_{x}^{(n-1)D})) \tag{6.70}$$

(ii)

$$\mathcal{V}_{x}^{nD} = \mathcal{V}ec_{x}(\underline{w}_{1,n}^{(p)}, \mathcal{V}_{x}^{(n-1)D}) \tag{6.71}$$

(iii)

$$\mathcal{V}_{\theta}^{nD} = \mathcal{V}ec_{\theta}(\underline{w}_{1,n}^{(p)}, \mathcal{V}_{x}^{(n-1)D}) \tag{6.72}$$

Proof: This theorem can easily be proved by method of induction and following the forementioned logical arguments establishing the relationship between (6.55), (6.56) and (6.66) for 2D, 3D and 4D cases and extrapolating this concept to the higher dimensions. Similarly (6.57) and (6.58) for 2D case can be compared with (6.63) and (6.64) for 3D case and latter can again be compared with (6.67) and (6.68) for 4D case and a similarity can be found out. Extending this idea to higher dimension using method of induction the above theorem can be proved. ■

6.8.2 nD FastICA Normalization Stage

Recalling (6.9) of normalization mode for 2D FastICA and using the notations introduced in (6.57), the angle information out of the vectoring mode in 2D mode CORDIC can be written as:

$$\begin{aligned} \mathcal{V}_{\theta}^{2D} &= \mathcal{V}ec_{\theta}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}) \\ &= \mathcal{V}ec_{\theta}(w_{1,1}, w_{1,2}) \end{aligned} \tag{6.73}$$

It is to be noted here that the superfix "p" is removed for the sake of simplicity. Following the same way, the "x"-output of the Vectoring mode of the CORDIC of the 2D case can be written as:

$$\begin{aligned} \mathcal{V}_{x}^{2D} &= \mathcal{V}ec_{x}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}) \\ &= \mathcal{V}ec_{x}(w_{1,1}, w_{1,2}) \end{aligned} \tag{6.74}$$

Now considering the 3D FastICA case, using the expressions given in (6.24) - (6.26) and using the notations in (6.64), the θ output of the Vectoring mode CORDIC for 3D

FastICA Normalization stage can be written as:

$$\begin{aligned}\mathcal{V}_{\theta}^{3D} &= \mathcal{V}ec_{\theta}^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_{x}^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)})) \\ &= \mathcal{V}ec_{\theta}(w_{1,3}, \mathcal{V}ec_{x}(w_{1,1}, w_{1,2})) \\ &= \mathcal{V}ec_{\theta}(w_{1,3}, \mathcal{V}_{x}^{2D})\end{aligned} \tag{6.75}$$

Similarly, using (6.24) - (6.26) and (6.63) the x-output of the Vectoring mode CORDIC for 3D FastICA normalization stage can be written as:

$$\mathcal{V}_{x}^{3D} = \mathcal{V}ec_{x}(w_{1,3}, \mathcal{V}_{x}^{2D}) \tag{6.76}$$

It is important to note that for 3D case (6.75) and (6.76) are expressed in terms of 2D case as given in (6.73) and (6.74) in recursive way. Similarly for 4D case, using (6.47) - (6.50) and using (6.67), the following equation holds:

$$\begin{aligned}\mathcal{V}_{\theta}^{4D} &= \mathcal{V}ec_{\theta}^{l3}(w_{1,4}^{(p+1)}, \mathcal{V}ec_{x}^{l2}(w_{1,3}^{(p+1)}, \mathcal{V}ec_{x}^{l1}(w_{1,1}^{(p+1)}, w_{1,2}^{(p+1)}))) \\ &= \mathcal{V}ec_{\theta}(w_{1,4}, \mathcal{V}ec_{x}(w_{1,3}, \mathcal{V}ec_{x}(w_{1,1}, w_{1,2}))) \\ &= \mathcal{V}ec_{\theta}(w_{1,4}, \mathcal{V}_{x}^{3D})\end{aligned} \tag{6.77}$$

Similarly using (6.47) - (6.50) and using (6.68), the x-output of the Vectoring mode CORDIC in 4D normalization case can be written as follows:

$$\mathcal{V}_{x}^{4D} = \mathcal{V}ec_{x}(w_{1,4}, \mathcal{V}_{x}^{3D}) \tag{6.78}$$

It is important to note that (6.77) and (6.78) are expressed in terms of (6.76) in recursive way.

Lemma 1*:*

$$\mathcal{V}_{\theta}^{nD} = \mathcal{V}ec_{\theta}^{l_{(n-1)}}(w_{1,n}, \mathcal{V}_{x}^{(n-1)D}) = \mathcal{V}ec_{\theta}(w_{1,n}, \mathcal{V}_{x}^{(n-1)D}) \tag{6.79}$$

$$\mathcal{V}_{x}^{nD} = \mathcal{V}ec_{x}^{l_{(n-1)}}(w_{1,n}, \mathcal{V}_{x}^{(n-1)D}) = \mathcal{V}ec_{x}(w_{1,n}, \mathcal{V}_{x}^{(n-1)D}) \tag{6.80}$$

Proof: It has been shown before that for 4D case (6.77) and (6.78) can be expressed in terms of (6.76) which belongs to 3D. Similarly it has also been shown that for 3D case (6.75) and (6.76) can be expressed in terms of 2D case as shown in (6.74). Thus following the same way using the method of induction, it can be proved that the vectoring mode

output of nD case can be expressed in a recursive way in terms of the $(n-1)$D which proves the lemma. ■

For the normalization stage of 2D FastICA i.e. when $n = 2$, removing $(p+1)$-term for the sake of simplicity, expression of the normalized components as given in (6.9) can be re-framed as follows:

$$\begin{aligned} \underline{w}_{1,1} &= \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) \\ &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_y^{2D} \end{aligned} \tag{6.81}$$

$$\underline{w}_{1,2} = \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) = \mathcal{R}_x^{2D} \tag{6.82}$$

Now, recalling the normalization stage of 3D FastICA (i.e. when $n = 3$), removing the $(p+1)$-terms and the level information for the sake of simplicity and using (6.73), (6.26) can be re-framed as follows:

$$\begin{aligned} \underline{w}_{1,3} &= \mathcal{R}ot_y(0, 1, \mathcal{V}ec_\theta(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2}))) \\ &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{3D}) \\ &= \mathcal{R}_y^{3D} \end{aligned} \tag{6.83}$$

Following the same notations and using (6.73), (6.24) and (6.25) can be re-framed as follows:

$$\begin{aligned} \underline{w}_{1,1} &= \mathcal{R}ot_y(0, \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2}))), \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) \\ &= \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_y^{2D} \end{aligned} \tag{6.84}$$

$$\begin{aligned} \underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2}))), \mathcal{V}ec_\theta(w_{1,1}, w_{1,2})) \\ &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_x^{2D} \end{aligned} \tag{6.85}$$

The following point to be noted here - although the notations used to express $\underline{w}_{1,1}$ and $\underline{w}_{1,2}$ for both 2D and 3D cases are same (as shown in (6.81), (6.82), (6.84) and (6.85) respectively), the y-inputs of the Rotation mode CORDIC are different. For 2D case, 1

is fed as the y-input whereas for 3D case, the x-input of the next level Rotation mode CORDIC is fed back as the present Rotation mode y-input. This is clear from Fig. 6.2 and Fig. 6.7 as well.

Similarly for 4D FastICA Normalization stage, removing the $(p+1)$ terms and the "level" information from the expression for the sake of simplicity, (6.50) can be re-framed as:

$$\begin{aligned}\underline{w}_{1,4} &= \mathcal{R}ot_y(0,1,\mathcal{V}ec_\theta(w_{1,4},\mathcal{V}ec_x(w_{1,3},\mathcal{V}ec_x(w_{1,1},w_{1,2})))) \\ &= \mathcal{R}ot_y(0,1,\mathcal{V}_\theta^{4D}) \\ &= \mathcal{R}_y^{4D}\end{aligned} \tag{6.86}$$

Now, using (6.77) in the expression of $\underline{w}_{1,3}$ as obtained from (6.49) for 4D FastICA can be written as:

$$\begin{aligned}\underline{w}_{1,3} &= \mathcal{R}ot_y(0,\mathcal{R}ot_x(0,1,\mathcal{V}ec_\theta(w_{1,4},\mathcal{V}ec_x(w_{1,3},\mathcal{V}ec_x(w_{1,1},w_{1,2})))),\mathcal{V}_\theta^{3D}) \\ &= \mathcal{R}ot_y(0,\mathcal{R}_x^{4D},\mathcal{V}_\theta^{3D}) \\ &= \mathcal{R}_y^{3D}\end{aligned} \tag{6.87}$$

where,

$$\mathcal{R}_x^{4D} = \mathcal{R}ot_x(0,1,\mathcal{V}ec_\theta(w_{1,4},\mathcal{V}ec_x(w_{1,3},\mathcal{V}ec_x(w_{1,1},w_{1,2}))))$$

following the same notation convention used in (6.86).

Following the same concept and notations described above, (6.47) and (6.48) for 4D normalization case can be written as:

$$\begin{aligned}\underline{w}_{1,1} &= \mathcal{R}ot_y(0,\mathcal{R}ot_x(0,\mathcal{R}ot_x(0,1,\mathcal{V}ec_\theta(w_{1,4},\mathcal{V}ec_x(w_{1,3},\mathcal{V}ec_x(w_{1,1},w_{1,2})))),\mathcal{V}ec_\theta(w_{1,3}, \\ &\quad \mathcal{V}ec_x(w_{1,1},w_{1,2}))),\mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}ot_y(0,\mathcal{R}_x^{3D},\mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_y^{2D}\end{aligned} \tag{6.88}$$

$$\begin{aligned}\underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}ot_x(0, \mathcal{R}ot_x(0, 1, \mathcal{V}ec_\theta(w_{1,4}, \mathcal{V}ec_x(w_{1,3}, \mathcal{V}ec_x(w_{1,1}, w_{1,2})))), \mathcal{V}ec_\theta(w_{1,3}, \\ &\quad \mathcal{V}ec_x(w_{1,1}, w_{1,2}))), \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_x^{2D}\end{aligned} \tag{6.89}$$

From (6.88) and (6.89) it is to be noted that the x-output of the Rotation mode for the 3D case is fed backward as the y-input of the Rotation mode again.

Based on the above discussion on the CORDIC based 2D, 3D and 4D FastICA normalization cases and their relationship, the following generalized recursive algorithm for CORDIC based nD FastICA normalization stage can be proposed:

Theorem 2*:* **CORDIC based Recursive Formulation of the Normalization Stage of nD FastICA Algorithm:** (a). When $n = 2$, $\underline{w}_{1,1}$ and $\underline{w}_{1,2}$ can be represented as:

$$\underline{w}_{1,1} = \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{2D}) \tag{6.90}$$

$$\underline{w}_{1,2} = \mathcal{R}ot_x(0, 1, \mathcal{V}_\theta^{2D}) \tag{6.91}$$

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

$$\underline{w}_{1,1} = \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \tag{6.92}$$

$$\underline{w}_{1,2} = \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \tag{6.93}$$

$$\underline{w}_{1,3} = \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{3D}) \tag{6.94}$$

(c). When $n \geq 4$,

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

$$\begin{aligned}\underline{w}_{1,i} &= \mathcal{R}_y^{iD} \\ &= \mathcal{R}ot_y(0, \mathcal{R}_x^{(i+1)D}, \mathcal{V}_\theta^{iD})\end{aligned} \tag{6.95}$$

and

$$\begin{aligned} \underline{w}_{1,n} &= \mathcal{R}_y^{nD} \\ &= \mathcal{R}ot_y(0, 1, \mathcal{V}_\theta^{nD}) \end{aligned} \tag{6.96}$$

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

$$\begin{aligned} \underline{w}_{1,1} &= \mathcal{R}ot_y(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_y^{2D} \end{aligned} \tag{6.97}$$

$$\begin{aligned} \underline{w}_{1,2} &= \mathcal{R}ot_x(0, \mathcal{R}_x^{3D}, \mathcal{V}_\theta^{2D}) \\ &= \mathcal{R}_x^{2D} \end{aligned} \tag{6.98}$$

Proof. (a) Proof follows straightway from the same procedure used to derive (6.81) and (6.82) using *Lemma-1*.

(b) The procedure discussed above to derive (6.84), (6.85) and (6.83) can be used along with *Lemma-1* to prove (6.92), (6.93) and (6.94) respectively.

(c) For 4D case i.e. when $n = 4$, it was already shown in (6.86) that the 4^{th} (i.e. n^{th}) component can be obtained as the y-output of the Rotation mode CORDIC when 0 and 1 are fed respectively to the x and y inputs of this mode. The x-output of this Rotation mode is fed back as the y-input of this mode to obtain the 3^{rd} (i.e. $(n-1)^{th}$) component as shown in (6.87). Following the same analytical treatment adopted to derive (6.86) and (6.87) for 4D case, can be extended in higher dimensions where $n > 4$. For example for 5D case, the 5^{th} component would be the y-output of the Rotation mode CORDIC and the x-output would be fed back to this mode as the y-input to compute the 4^{th} component. This will appear again as the y-output of the Rotation mode and in similar way the 3^{rd} component can be computed by feeding back the x-output of the rotation mode. Thus part *(i)* of the above theorem can be proved.

Again referring back to the 4D normalization case, it can be observed from (6.88) and (6.89) that the 1^{st} and the 2^{nd} components are always the y and x outputs of the same rotation mode whose x-input is 0 and y-input is the x-output of the rotation mode CORDIC used for the 3^{rd} component computation. Following the same analytical treatment adopted to derive (6.88) and (6.89), first and second components for any higher dimensional (i.e. $n > 4$) FastICA normalized vector can be derived. This proves part *(ii)* of the above theorem. ■

6.8.3 nD FastICA Component Estimation Stage

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

$$\hat{s}_{1,j} = \mathcal{R}_{x,j}^{nD} \tag{6.99}$$

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

(i)

$$\mathcal{R}_{x,j}^{nD} = \mathcal{R}ot_x(z_{n,j}, \mathcal{R}_{x,j}^{(n-1)D}, \mathcal{V}ec_\theta(w_{1,n}, \mathcal{V}_x^{(n-1)D})) \tag{6.100}$$

(ii)

$$\mathcal{V}_x^{nD} = \mathcal{V}ec_x(w_{1,n}, \mathcal{V}_x^{(n-1)D}) \tag{6.101}$$

(iii)

$$\mathcal{V}_\theta^{nD} = \mathcal{V}ec_\theta(w_{1,n}, \mathcal{V}_x^{(n-1)D}) \tag{6.102}$$

Proof: The notations adopted to prove *Theorem-1* will also be used here to prove the above theorem. Considering (6.11) for 2D Estimation stage and comparing with (6.28) for 3D Estimation stage, it can be observed that 3D Estimation needs one more *Rotation* over the 2D Estimation. In the similar way, comparing (6.28) with (6.53) for 4D Estimation stage, it can be seen that the 4D stage needs one more *Rotation* mode over the 3D stage. Continuing this way for higher dimensions where $n > 4$, it can also be shown that nD Estimation stage needs one more *Rotation* mode over the $(n-1)$D Estimation stage. Thus the estimated component for the nD FastICA stage can be represented in recursive fashion as one extra rotation over the previous $(n-1)$D FastICA estimation stage considering 2D stage as the fundamental unit. This proves the above theorem. ■

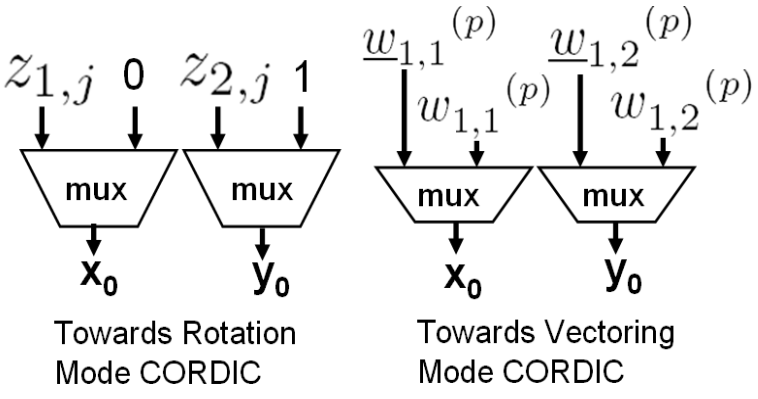


FIGURE 6.14: Multiplexer array used in front of the 2-D FastICA.

6.9 Generalized Architecture for the Proposed CORDIC based nD FastICA

This section discusses the architectural details of the proposed CORDIC based generalized nD FastICA algorithm described in the last section. From the discussion of CORDIC based $2D$ FastICA architecture in Section 6.3, it has been seen that one Rotation mode CORDIC and one Vectoring mode CORDIC was necessary for each of the three stages - iteration, normalization and estimation. No concept of *level* was introduced at that point until we described CORDIC based $3D$ FastICA algorithm in Section 6.4 where it has been shown that each of these three stages need *two levels*. Corresponding architecture of this $3D$ algorithm described in Section 6.5 has depicted this concept clearly. Following the same fashion, it has been described the respective architecture of the proposed CORDIC based $4D$ FastICA algorithm in Section 6.7 where it has been shown that each of the three stages - iteration, normalization and estimation, needs *three levels* of CORDIC (including Rotation and Vectoring mode). Proceeding in the same way and as discussed in the last Section, without any loss of generality it can be said that each of the three FastICA stages needs $(n-1)$ *levels* for its architectural implementation.

At this point it can be argued that with the increase of the dimension n, number of levels increases and this may be disadvantageous from the point of view of architectural performance. But, thanks to the inherent property of the FastICA deflationary method, none of the three stages - iteration, normalization and Estimation,needs to be executed concurrently. This leaves us with an option of reuse of one rotation and one vectoring mode CORDIC time and again to accomplish the desired operation only at the expense of simple multiplexer arrays. The following paragraphs deal with the formulation of such multiplexer arrays used for the proposed CORDIC based nD FastICA architecture.

6.9.1 Multiplexer arrays for CORDIC Reuse

Recalling the multiplexed architecture of the proposed $2D$ algorithm shown in Fig. 6.4 in Section 6.3, it can be seen that each data input to the Rotation and Vectoring mode CORDIC is associated with *one* $(2:1)$ multiplexer at their very beginning. This is shown in Fig. 6.14. Since the concept of *level* was introduced from $3D$ onwards, their multiplexed architectures also got two arrays of multiplexers as shown in multiplexed architectures of $3D$ and $4D$ in Fig. 6.9 and 6.13 respectively. These multiplexer arrays are shown in Fig. 6.15 and 6.16 respectively for $3D$ and $4D$ cases.

Comparing Fig. 6.14, 6.15 and 6.16, it can be observed that the multiplexer arrays used for $2D$ architecture are also used in $3D$ and $4D$ (as shown within "array-1" in Fig. 6.15 and 6.16). Moreover, due to the inclusion of the *level* information, $3D$ and

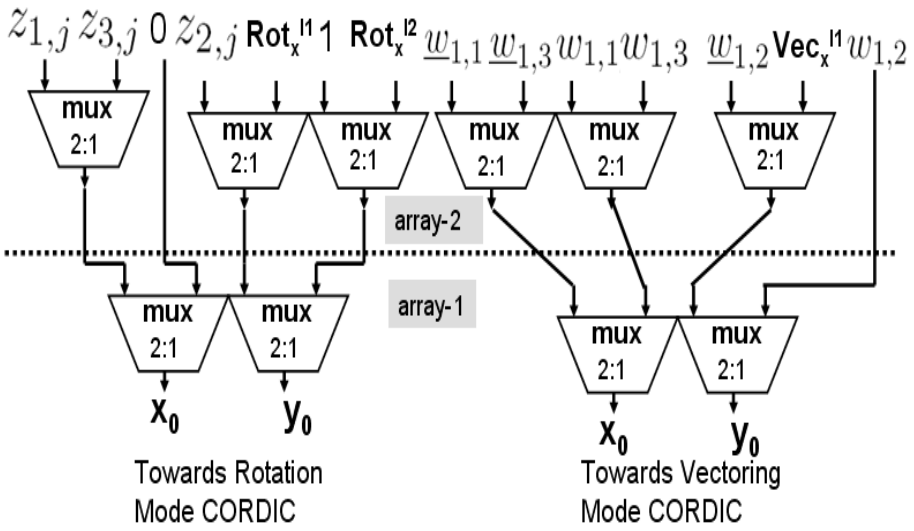


FIGURE 6.15: Multiplexer arrays used in front of the 3-D FastICA.

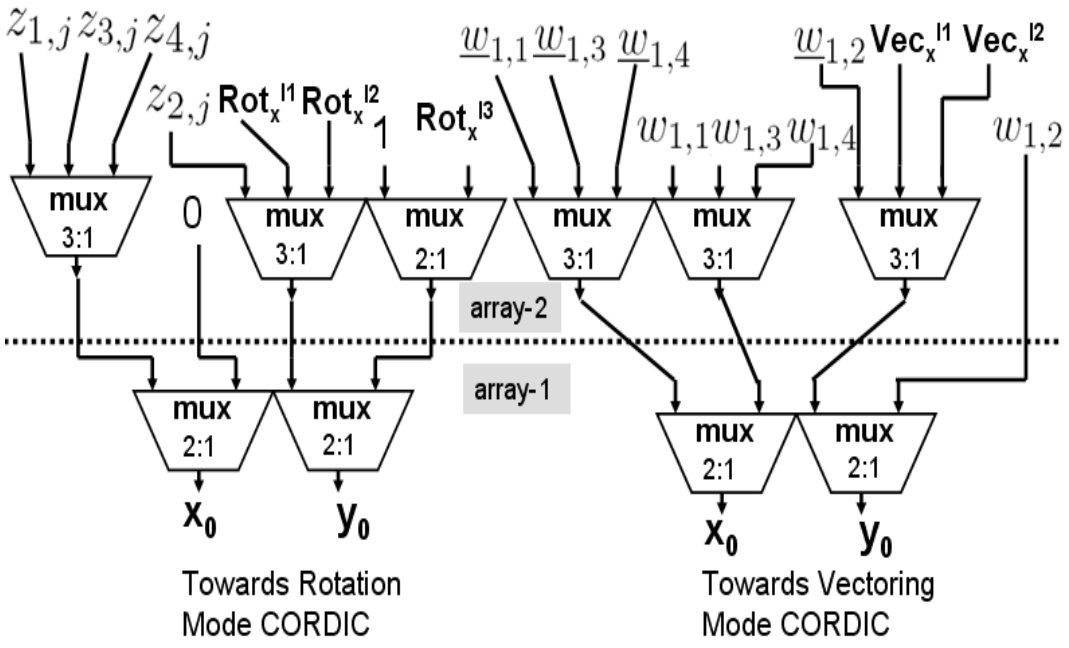


FIGURE 6.16: Multiplexer arrays used in front of the 4-D FastICA.

$4D$ architectures need another array of multiplexers, shown as "array-2" in Fig. 6.15 and 6.16. However the notable difference between the "array-2" in Fig. 6.15 and 6.16 is the *type* of multiplexer used. For $3D$ case, as shown in Fig. 6.15, "array-2" comprises of *three* $(2:1)$ multiplexers at the Rotation side and *three* $(2:1)$ multiplexers at the vectoring side. Similarly, from Fig. 6.16, it can be observed that "array-2" comprises of *three* $(3:1)$ multiplexers at the Vectoring side and *two* $(3:1)$ and *one* $(2:1)$ multiplexer at the rotation side. Following this trend, if multiplexed architecture is to be designed for $5D$ case, then without any loss of generality, it can be inferred that "array-1" will have same structure as shown in Fig. 6.14 and "array-2" will comprise of *three* $(4:1)$ multiplexers at its Vectoring side and *two* $(4:1)$ and *one* $(2:1)$ multiplexer at its rotation side. Therefore using the method of induction and exploiting this above mentioned similarity, the following can be inferred about the multiplexer arrays of the proposed CORDIC based generalized nD FastICA architecture:

Multiplexer Array-1: It comprises of *two* $(2:1)$ multiplexers at the Rotation Side and *two* $(2:1)$ multiplexers at the Vectoring side. This multiplexer array is the same as that used for $2D$ case shown in Fig. 6.14.

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

This multiplexer arrays for nD case are shown in Fig. 6.17.

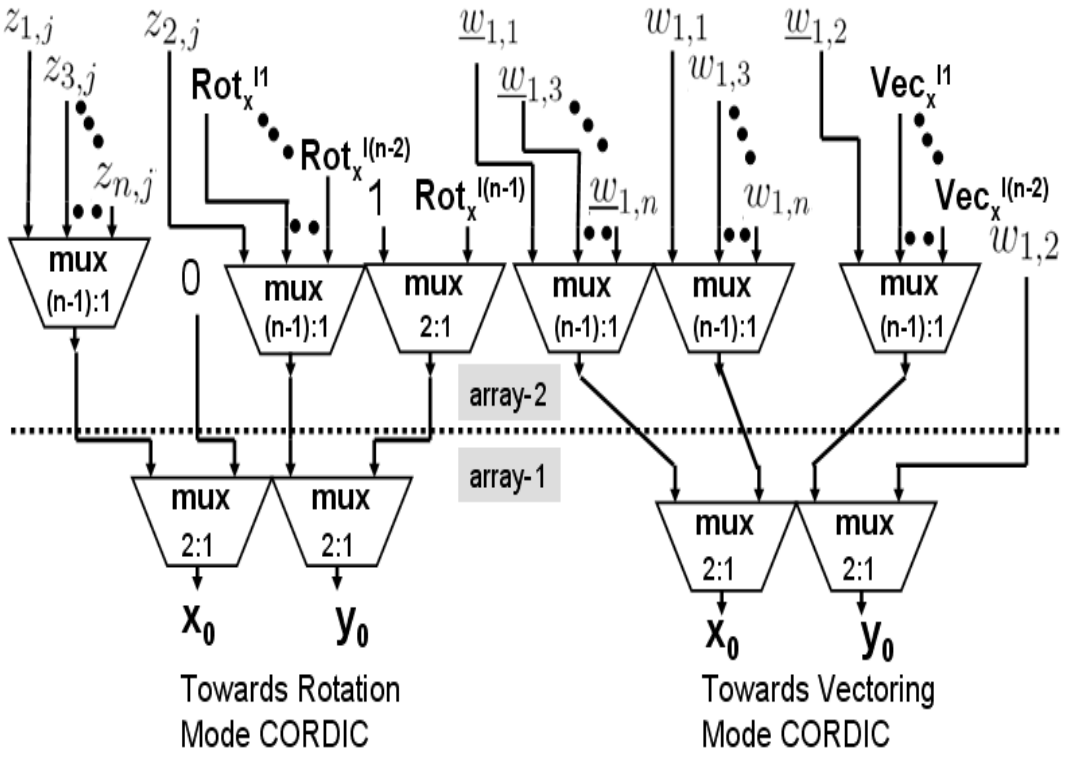


FIGURE 6.17: Multiplexer arrays used in front of the n-D FastICA.

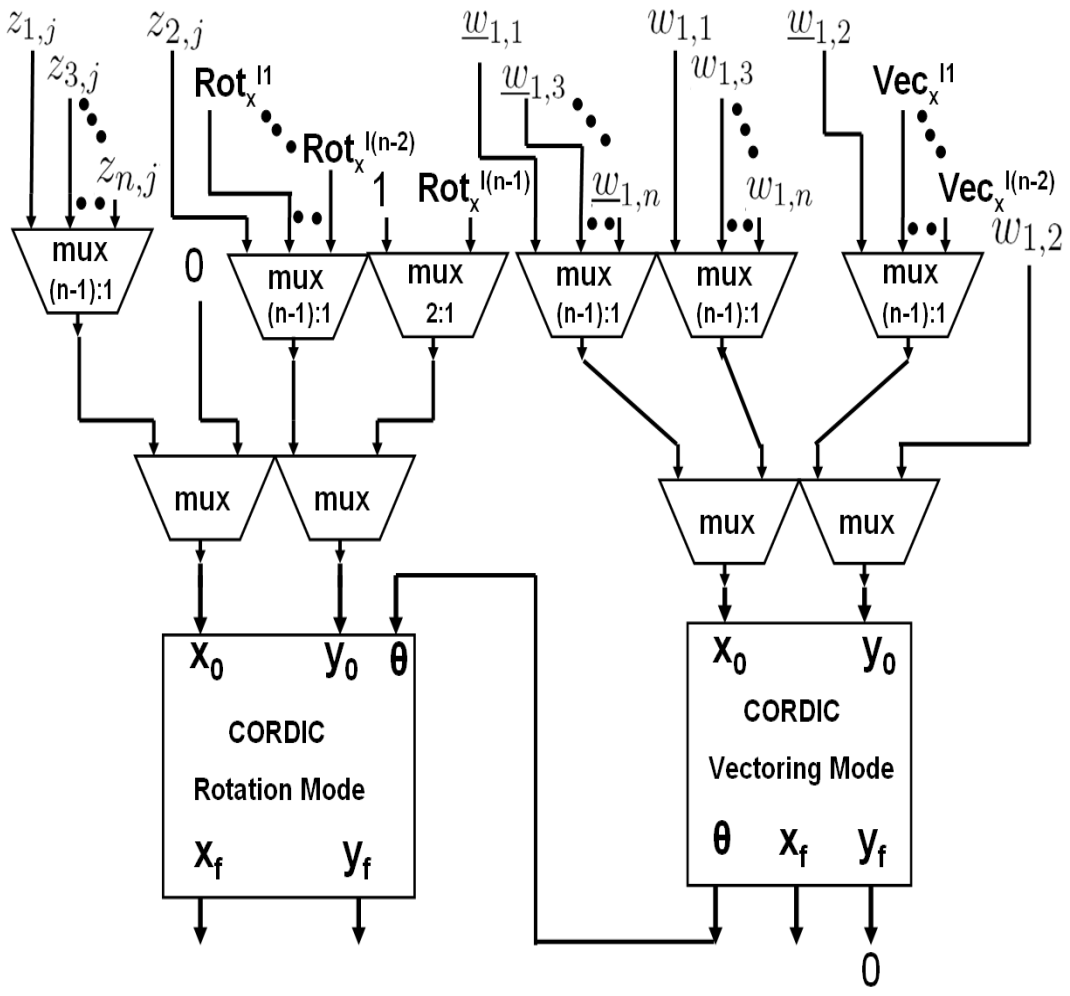


FIGURE 6.18: Architecture of the proposed Co-ordinate Rotation based n-D FastICA.

6.9.2 CORDIC based Multiplexed nD Architecture

Fig. 6.18 presents the generalized architecture for the proposed CORDIC based nD FastICA algorithm and shows the multiplexer arrays used for CORDIC reuse as discussed in the last subsection. In case of $2D$ architecture, comparing Fig. 6.18 with Fig. 6.4, it can be observed that multiplexer "array-2" will not exist in this generalized architecture. However, for higher dimensions where $n \geq 3$, putting $n = 3, 4, ..$ in Fig. 6.18, any dimensional FastICA architecture can be realised. For example, considering $n = 3$ in Fig. 6.18 and comparing with the $3D$ architecture shown in Fig. 6.9 in section 6.5, or considering $n = 4$ in Fig. 6.18 and comparing with the $4D$ architecture shown in Fig. 6.13 in Section 6.7, the above statement can be verified.

Thus, as claimed at the beginning of this chapter that the algorithm and architecture to be proposed here can be extended in any dimension, is verified. This provides the designer with the flexibility of using one Rotation mode CORDIC and one Vectoring mode CORDIC for both FastICA Preprocessing as well as the main Iteration stage in

generic way.

6.10 Doubly Pipelining for the Proposed CORDIC based nD FastICA Architecture

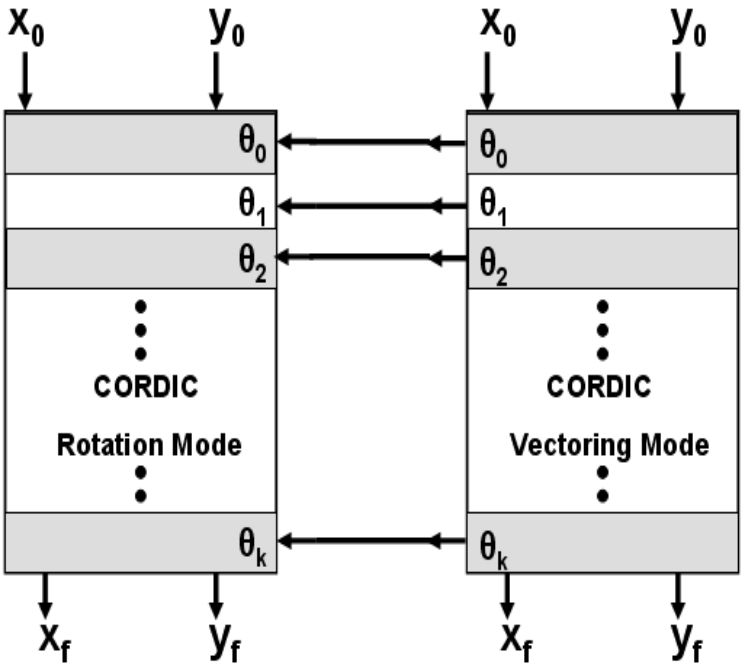


FIGURE 6.19: Doubly Pipelining of the proposed CORDIC based 2-D FastICA architecture.

All architectures discussed so far has one thing in common - the explicit *angle* information obtained at the primary output of Vectoring mode CORDIC is connected to the θ input of the Rotation mode CORDIC. While describing CORDIC based $2D$ FastICA in Section 6.3, it has been mentioned that the possibility of further architectural optimization by removing the necessity of explicit angle computation in vectoring mode CORDIC. The method used to accomplish this is known as "Doubly Pipelining" [149]. Although this term has been mentioned in Section 6.3, this method will be discussed in detail here in the following paragraphs.

It has already been discussed in Section 6.1 that in the Vectoring mode CORDIC, y-output is made zero after finite number of iterations. The rotation of the initial input vector in each such iteration stage is known as "micro-rotation". Each micro-rotation outputs the modified x and y value of the initial vector as well as the rotation direction known as "micro angle". Let us denote it by θ_i where $i = 0, 1, 2, 3, .., k$ and $(k+1)$ is the total number of iterations. Depending upon the rotation direction i.e. whether clockwise or anticlockwise, θ_i considered to be $+1$ or -1. After $(k+1)$-number of iterations, each of these micro-angles are are weighed by 2^{-i} and the weighted results

are summed to produce the accumulated angle θ. When this θ value is fed to the Rotation mode CORDIC, again this is divided into sequence of micro-angles after finite number of iterations. These micro-angles are the same as those ones found in the Vectoring mode. Due to this fact and since Vectoring and Rotation mode CORDIC are used together, explicit θ information becomes redundant and *micro angles* generated from the vectoring mode can be used straightway in the Rotation mode instead. Fig. 6.19 shows this operation for the proposed CORDIC based $2D$ FastICA architecture. Comparing 6.4 described in Section 6.3 with Fig. 6.19, it can be observed that here micro-angles (θ_i, $i = 0, 1, ..., k$) are connected to the respective micro rotation stages of the Rotation mode CORDIC which not only eliminates the necessity of θ computation but also improves the overall operational speed.

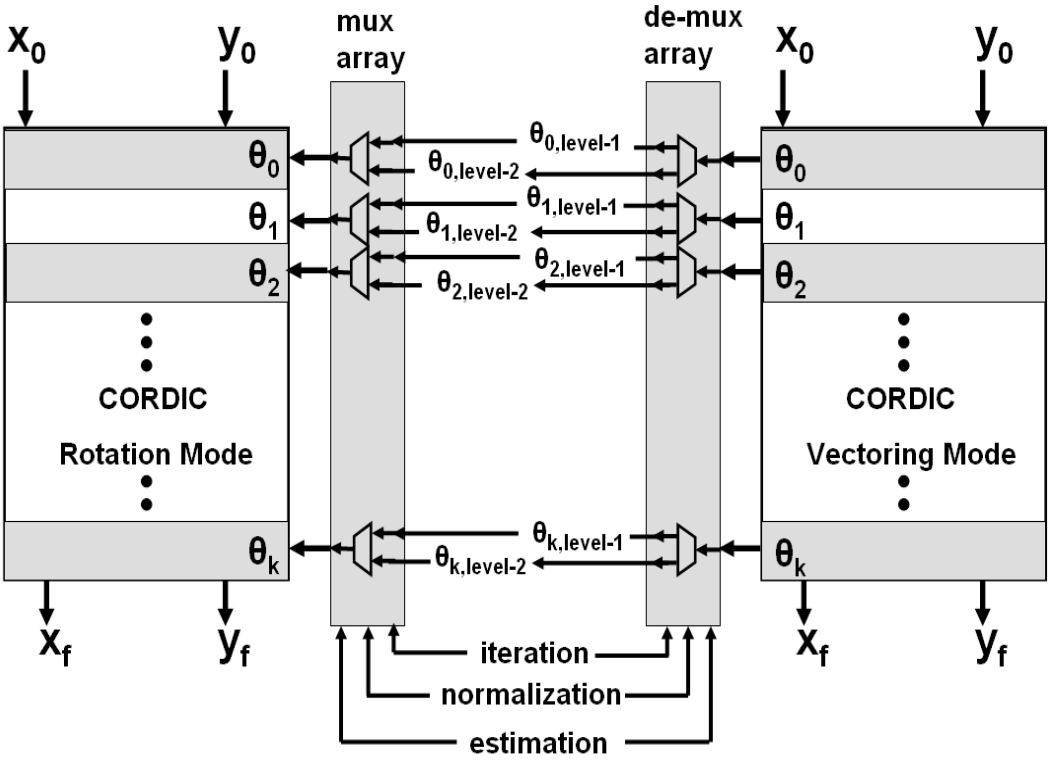


FIGURE 6.20: Doubly Pipelining of the proposed CORDIC based 3-D FastICA architecture.

Since the aim here is to investigate how the same idea of $2D$ can be extended into nD, it is important to explore further this doubly pipelining issue in higher dimension. From Fig. 6.19 it has been seen that each micro-angle can be connected straight to the micro-angle input to the respective Rotation mode CORDIC. However, for $n \geq 3$, since the concept *level* comes in, the micro-angle connection from the Vectoring mode to the Rotation mode CORDIC may not be as straight forward as shown in Fig. 6.19. Considering first $3D$ architecture as discussed in Section 6.5, each of the three stages - iteration, normalization and estimation, has two *levels* and each level has different set of micro-angles generated from the Vectoring mode CORDIC. Since the same Vectoring

and Rotation mode CORDIC is reused time and again, it is essential to use *one* $(1:2)$ de-multiplexer at the output of each micro-angle in the Vectoring mode CORDIC and *one* $(2:1)$ multiplexer at the input of each micro angle in the rotation mode CORDIC. This scenario is depicted in Fig. 6.20.

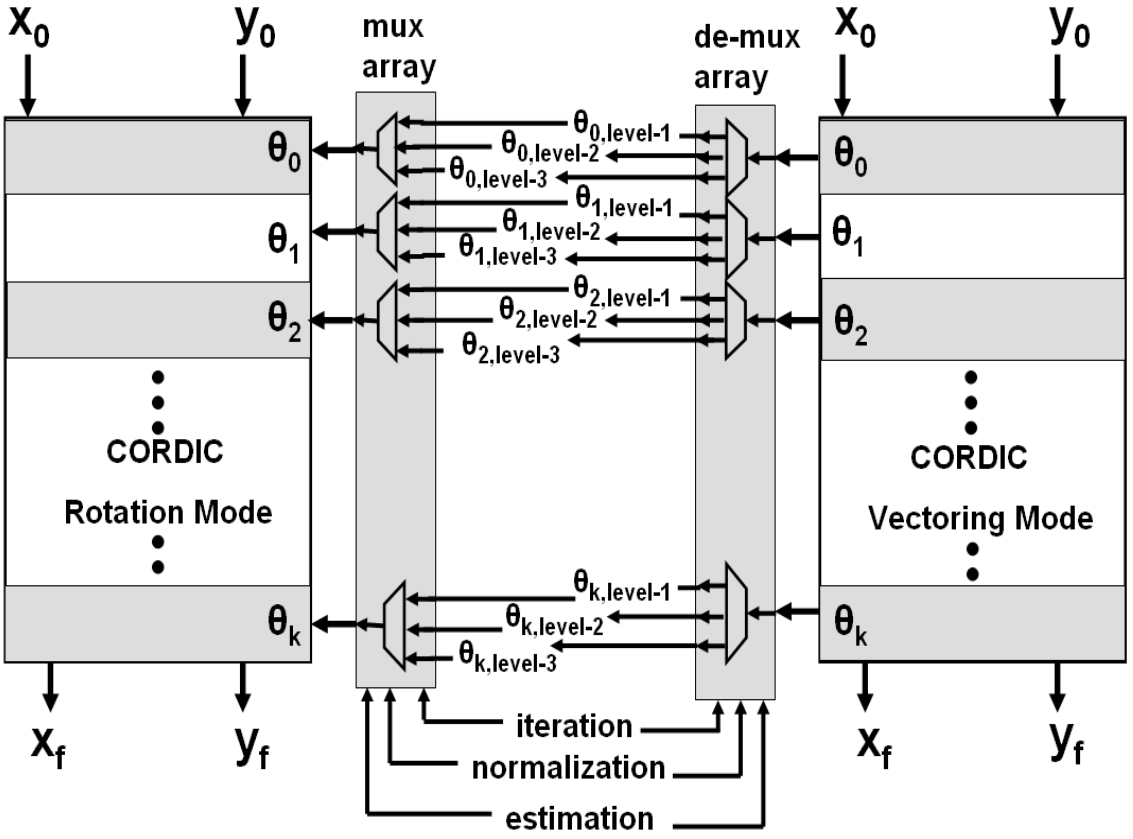


FIGURE 6.21: Doubly Pipelining of the proposed CORDIC based 4-D FastICA architecture.

The same procedure can be adopted to describe the use of doubly pipelining method in case of $4D$ architecture. Since the proposed CORDIC based $4D$ FastICA consists of *three levels* (described in Section 6.6), each micro-angle output from the Vectoring mode CORDIC goes to *one* $(1:3)$ de-multiplexer and each micro-angle input of the Rotation mode CORDIC has *one* $(3:1)$ multiplexer at its beginning. Fig. 6.21 depicts this CORDIC based $4D$ FastICA architecture employing doubly pipelining method.

The above mentioned procedure can be followed to propose the CORDIC based generalized nD FastICA architecture employing doubly pipelining method. In the last section it was mentioned that the proposed CORDIC based nD FastICA algorithm comprises of $(n-1)$ levels. Therefore, if the same set of rotation and Vectoring mode CORDIC is to be used time and again, using the above arguments, each micro-angle from the Vectoring mode CORDIC should be connected to *one* $((n-1):1)$ de-multiplexer and each micro-angle input to the Rotation mode CORDIC should have *one* $((n-1):1)$

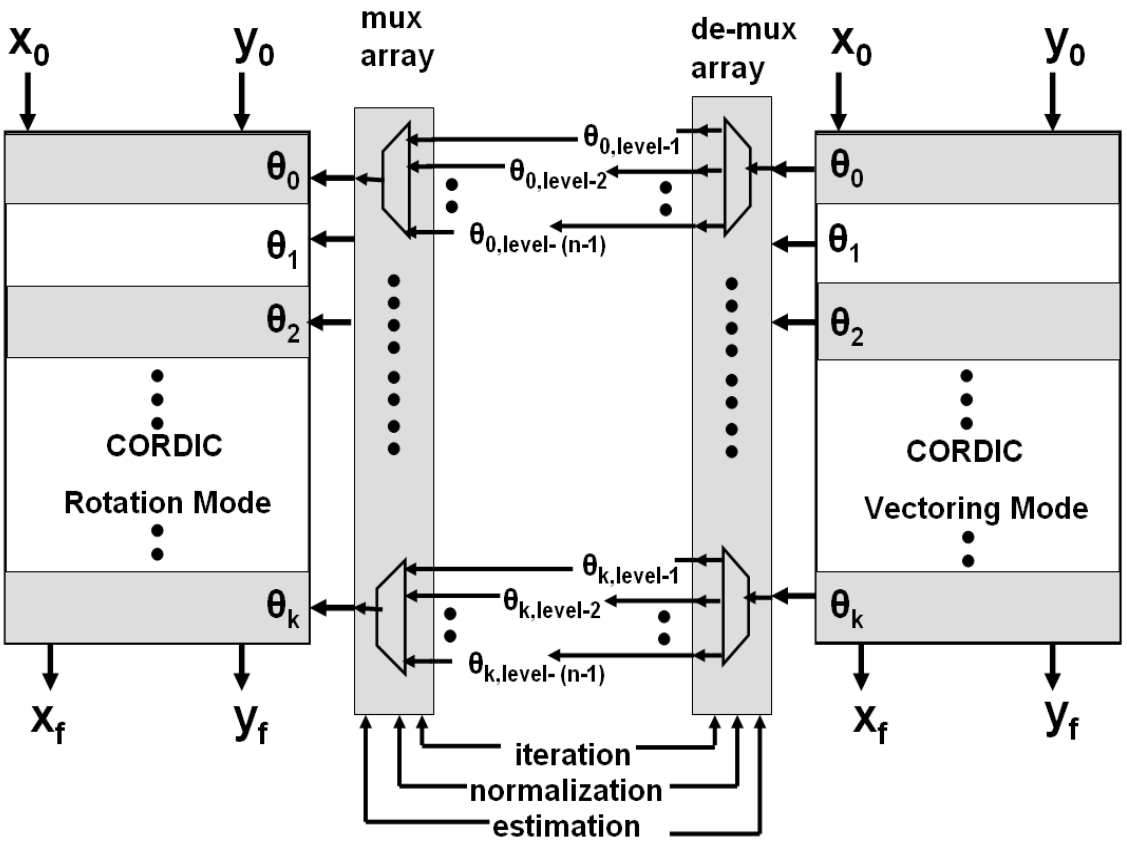


FIGURE 6.22: Doubly Pipelining of the proposed CORDIC based n-D FastICA architecture.

multiplexer at its very beginning. Fig. 6.22 shows this architecture for nD case where three different stages - iteration, normalization and estimation can be used to for the multiplexer's and de-multiplexer's select signals generation.

Now replacing the conventional Rotation and Vectoring mode CORDIC architecture from Fig. 6.18 by the doubly pipelining architecture shown in Fig. 6.22, CORDIC based generalized nD FastICA architecture proposed in Section 6.9 can further be modified as shown in Fig. 6.23. The multiplexer arrays discussed in the last section will remain same at the beginning of the data inputs of the Rotation as well as the Vectoring mode CORDIC. Only the doubly pipelining for nD CORDIC shown in Fig. 6.22 is used in Fig. 6.23 to enhance the architectural performance in terms of better operational speed.

6.11 Some Necessary Considerations for CORDIC based FastICA Implementation

From the discussion of the proposed architectures so far in this chapter it is well understood at this point that the Vectoring mode CORDIC is used first to compute the angle between the initial vector and the principal axis and then the rotation mode CORDIC

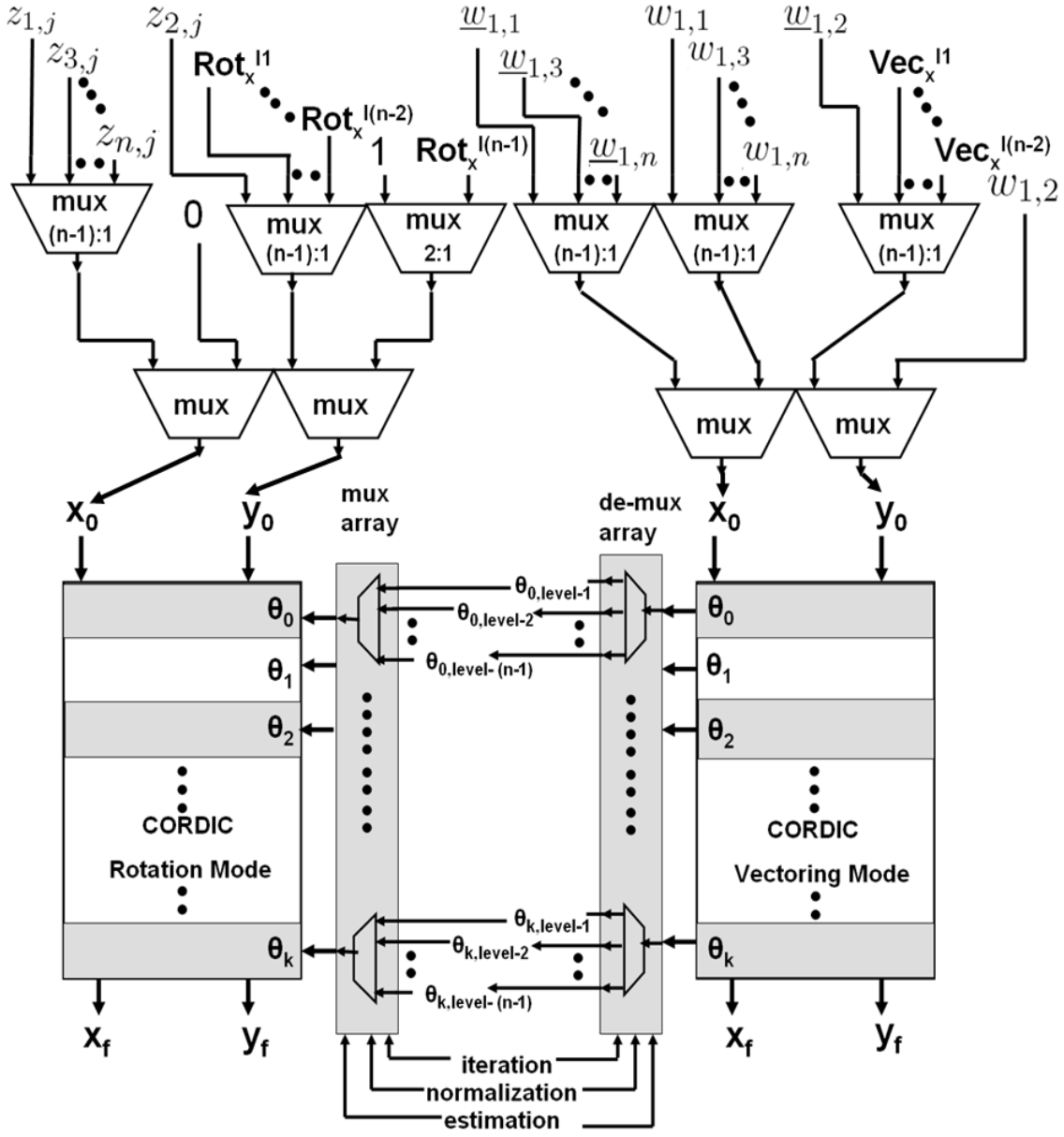


FIGURE 6.23: Overall architecture of the proposed CORDIC based n-D FastICA algorithm.

uses this angle information to rotate the random incoming vectors. In the last section it has been observed that explicit angle information from the vectoring mode CORDIC is not necessary for this purpose which allows us to speed up the operation quite significantly. The method discussed in the last section is commonly known as doubly pipelining. Denoting the *clockwise micro rotation* of the input vector to the *Vectoring* mode CORDIC by "0" and *anti-clockwise micro rotation* by "1" and using the doubly pipelining approach, the rotation mode CORDIC starts its computation based on this 0 and 1 angular information fed from the respective blocks of the vectoring mode and thereby improving the latency of the system significantly. However, from the system designer point of view, before starting the implementation of the proposed CORDIC based

nD FastICA algorithm based on the proposed architecture, it is necessary to consider some of the important issues as will be discussed shortly here in this section.

From (6.1) it has been seen that the input vector to the CORDIC vectoring mode is denoted by $[x_0\ y_0]$ and then it was to be rotated by the acute angle θ so that it overlaps with the positive principal axis x. The formula shown in (6.1) is the basic one and only valid for the first quadrant in the $2D$ co-ordinate plane in its present form. But to be realistic, the input vector may lie in either of the four quadrants and is completely random. Therefore it is important to find out a scheme which maps any vector lying in any quadrant into the first quadrant and thereby necessitating the modification of the basic CORDIC equation given in (6.1).

Considering the data-bus width of the digital hardware used for the implementation of the proposed architecture is b-bits, the quadrant where the input vector is lying can easily be found in the following way. As for example, if any vector is lying in the second quadrant, the x_0 value will be negative and y_0 will be positive. From the point of view of digital system designer, the negative and positive signs of any data are hold within its "sign" bit of its overall representation. This essentially means that if the "sign" bits of the input vector is checked, its respective quadrant can easily be traced. So just by checking the sign bits of the x and y components of the incoming vector, the initial quadrant can be found out. Table 6.1 provides such quadrant determination approach explicitly. It is to be noted from the Table 6.1 that a two-bit variable "quad" is used to hold the quadrant information.

TABLE 6.1: Quadrant information generation based on the sign-bit (b^{th} bit of the b-bit data-bus) of the Input Vector to the Vectoring mode CORDIC

Quadrant	$x_0(b)$	$y_0(b)$	quad
1^{st}	0	0	00
2^{nd}	1	0	10
3^{rd}	1	1	11
4^{th}	0	1	01

Thus once the quadrant information is computed and stored in "quad", only the absolute value of the input vector to the vectoring mode CORDIC is considered for further computation within this vectoring mode. This means the orientation of the vector has been changed from any quadrant to the first quadrant and thus the angle between the positive x-axis and this vector is less than $\pi/2$. This will definitely bring in some modifications of the basic CORDIC rotation formulation which will be derived soon.

But before proceeding further to the formulation of the quadrant mapping scheme it is important to do the following. Since (6.1) is the basic CORDIC rotation/vectoring formula in the clockwise sense, let us see what happens if the rotation of the incoming vector is in the anti-clockwise direction. Hence using $\theta = -\beta$ in (6.1), the following equation can be derived:

$$\begin{aligned}\begin{bmatrix} x_f \\ y_f \end{bmatrix} &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \\ &= \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\end{aligned} \tag{6.103}$$

Comparing (6.1) and (6.103), it can be seen that only the sign corresponding to the *sine* term changes and rest remains the same.

Now considering the input vector to the vectoring mode CORDIC is lying within the *second* quadrant and it has to be rotated by an obtuse angle α to ensure it overlaps with the positive x-axis, exploiting the circular symmetry of the $2D$ co-ordinate plane, obtuse angle α can be represented in terms of the acute angle θ as follows: $\alpha = (\pi - \theta)$. This signifies that the incoming vector to the rotation mode CORDIC $[x_{0,rot}\ y_{0,rot}]$ are to be rotated by angle $\alpha = (\pi - \theta)$. Using this expression in (6.1), the following equation can be derived:

$$\begin{aligned}\begin{bmatrix} x_f \\ y_f \end{bmatrix} &= \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix} \\ &= \begin{bmatrix} \cos(\pi-\theta) & \sin(\pi-\theta) \\ -\sin(\pi-\theta) & \cos(\pi-\theta) \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} -x_{0,rot} \\ -y_{0,rot} \end{bmatrix}\end{aligned} \tag{6.104}$$

It can be noted from (6.104) that if the input vector to the Vectoring mode CORDIC lies within the second quadrant, then the incoming vector to the rotation mode CORDIC has to be complemented. Moreover it is important to note that although we started deriving the above expression from the basic clockwise rotation (as given in (6.1)), we ended up having rotation in anti-clockwise sense as shown in (6.103). This essentially means if the incoming vector to the vectoring mode CORDIC lies within the second quadrant which is already obtained using Table 6.1 and represented by "$quad = 10$", then to ensure all rotations are taking place within the first quadrant by an acute angle θ, the micro rotation "0" signifies *anti-clockwise direction* and "1" signifies *clockwise direction*. This is shown in Table 6.2.

Similarly, if the input vector to the Vectoring mode CORDIC lies within the third quadrant, which is denoted by "$quad = 11$" in Table 6.1, then the rotation angle α can be represented in terms of an acute angle θ as : $\alpha = (\pi + \theta)$. Using this expression in (6.1), the following expression can be derived:

$$\begin{aligned}\begin{bmatrix} x_f \\ y_f \end{bmatrix} &= \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix} \\ &= \begin{bmatrix} \cos(\pi+\theta) & \sin(\pi+\theta) \\ -\sin(\pi+\theta) & \cos(\pi+\theta) \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} -x_{0,rot} \\ -y_{0,rot} \end{bmatrix}\end{aligned} \tag{6.105}$$

From (6.105), it can be noted that although the incoming vectors to the Rotation mode CORDIC need to be complemented, still "0" signifies *clockwise rotation* and "1" signifies *anti-clockwise rotation*. It is also shown in Table 6.2.

Similarly, if the input vector to the Vectoring mode CORDIC lies within the fourth quadrant, which is denoted by "$quad = 01$" in Table 6.1, then the rotation angle α can be represented in terms of an acute angle θ as : $\alpha = (2\pi - \theta)$. Using this expression in (6.1), the following expression can be derived:

$$\begin{aligned}\begin{bmatrix} x_f \\ y_f \end{bmatrix} &= \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix} \\ &= \begin{bmatrix} \cos(2\pi-\theta) & \sin(2\pi-\theta) \\ -\sin(2\pi-\theta) & \cos(2\pi-\theta) \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{0,rot} \\ y_{0,rot} \end{bmatrix}\end{aligned} \tag{6.106}$$

From (6.106), it can be noted that the incoming vectors to the Rotation mode CORDIC need not to be complemented in this case, however "0" signifies *anti-clockwise rotation* and "1" signifies *clockwise rotation*. It is also shown in Table 6.2.

TABLE 6.2: Relationship between Quadrant, Micro Rotations, Rotation Directions and Rotation mode CORDIC Inputs

Quadrant	Micro Rot	Rot Directions	Rot Inputs ($x_{0,rot}$, $y_{0,rot}$)
1^{st}	$\theta_i = 0$	clockwise	$x_{0,rot}$, $y_{0,rot}$
($quad = 00$)	$\theta_i = 1$	anti-clockwise	$x_{0,rot}$, $y_{0,rot}$
2^{nd}	$\theta_i = 0$	anti-clockwise	$-x_{0,rot}$, $-y_{0,rot}$
($quad = 10$)	$\theta_i = 1$	clockwise	$-x_{0,rot}$, $-y_{0,rot}$
3^{rd}	$\theta_i = 0$	clockwise	$-x_{0,rot}$, $-y_{0,rot}$
($quad = 11$)	$\theta_i = 1$	anti-clockwise	$-x_{0,rot}$, $-y_{0,rot}$
4^{th}	$\theta_i = 0$	anti-clockwise	$x_{0,rot}$, $y_{0,rot}$
($quad = 01$)	$\theta_i = 1$	clockwise	$x_{0,rot}$, $y_{0,rot}$

Apart from the afore-mentioned quadrant adjustment procedure, another important

implementation-specific issue is the numerical accuracy of CORDIC. It is a well researched topic and was first discussed in [147]. However the thorough analysis of the round-off error of CORDIC was done in [150] and followed by [151]. Both of these papers considered finite-precision arithmetic of the machine and also the finite number of CORDIC iterations and provided a detailed discussion of CORDIC's numerical accuracy in terms of the worst-case error bound. In the very next year, [152] brought in another practical assumption within the numerical accuracy models discussed in [150] and [151] which states the accumulated round-off error may also change the direction of the micro-rotation (denoted by θ_i in Table 6.2) and this may be considered as another source of round-off error of CORDIC. [153] considered the possibility of the input data of CORDIC may have been generated from other processes and thus may already be corrupted. Thus [153] assumed the round-off error within the input data of CORDIC itself and discussed in detail the numerical accuracy of CORDIC. However, from the implementation perspective, the approach proposed in [151] can be followed which states that if n-bit precision is wanted, a CORDIC unit with $(n + \log_2(n) + 2)$-bits and n iterations is sufficient. This concept can be utilized during the implementation of the proposed CORDIC based nD FastICA algorithm as well.

6.12 Hardware Complexity Analysis

In this section analyzes the performance of the proposed algorithm in terms of the hardware complexity. Since it has already been mentioned in the first section that the first reported FastICA implementation [76] was direct one-to-one mapping of the algorithm into architecture and recently reported architecture in [122] has shown better performance in terms of less hardware complexity over [76], [122] will be considered as the basis of comparison in the present context. However, it is important to note that although a hardware reduction methodology was proposed in [122], main thrust has been given on the preprocessing unit and no consideration was given on the main FastICA unit. Since from Section 6.2 to Section 6.10 it has been shown that how the same CORDIC which can be used for the preprocessing unit, can also be reused for FastICA Iteration, Normalization and output estimation phases for different dimensions, it is bound to reduce the area overhead by removing significant amount of computational blocks from the design as reported by the existing architectures. Detailed analysis is as follows.

6.12.1 Important Assumptions

Throughout the analysis we keep a generalized view of frame-length m and word-length b. Since FastICA is an iterative procedure, we consider only one single iteration because the same hardware resources can be reused for the next iterations.

Now considering the same assumption used in [122]- no resource sharing or parallel computing capability in one iteration, each arithmetic operation can be translated in terms of one arithmetic unit.

To provide a comparison on a uniform platform we consider only Ripple Carry Adder (RCA), Conventional Array Multiplier (CAM), Non-restoring Iterative Cellular Square Rooter (SQRT) and Non-restoring Array Divider (NAD) as the means of implementing the arithmetic operations. Considering a b-bit RCA requires b Full Adders (FA) (in a simplified view) [122], $b \times b$ CAM requires $b(b-2)$ FA plus b Half Adders (HA) and b^2 AND gates [122]. Similarly one $b \times b$ NAD consists of $0.5 \times b(3b-1)$ FA and $0.5 \times b(3b-1)$ XOR gates [122] and one b-bit SQRT needs $0.125 \times (b+6)b$ FA and XOR gates [138]. In addition, considering one FA cell requires 24 transistors, one HA cell and one two input XOR gates consists of 12 transistors and a two input AND gates consists of 6 transistors [122], we can calculate $TC_A = 24b$, $TC_M = 6b(5b-6)$, $TC_D = 18b(3b-1)$ and $TC_{SQ} = 18(b/2+1)(b/2+3)$, where TC_* are the transistor counts for RCA, CAM, NAD and SQRT respectively. One basic single bit $2:1$ Transmission Gate Multiplexer comprises of 4 transistors and thus b-bit $2:1$ multiplexer array has got $TC^{2:1}_{mux} = 4b$ [154]. Any larger multiplexer can be realised using cascaded $2:1$ multiplexer [154] and thus total transistor count of $n:1$ multiplexer ($TC^{n:1}_{mux}$) can be expressed in terms of $TC^{2:1}_{mux}$ as: $TC^{n:1}_{mux} = (n-1) * TC^{2:1}_{mux}$. All these results will be used in the following subsections for derivation of overall hardware complexity reduction achieved by the proposed CORDIC based nD FastICA architecture.

6.12.2 Performance Analysis for $2D$

Considering the $2D$ FastICA case first, in Iteration phase, comparing (6.2) with (6.5), it can be found that 2 multiplications and 1 addition operation is saved per sample per frame in each iteration. Therefore, for the whole frame m, total saving is - $2m$ multiplications and m additions. In Normalization phase, comparing (6.7) and (6.9), it can be observed that 2 multiplications, 1 addition, 1 square-rooting and 2 divisions operations are saved per iteration. In the output estimation phase, similar to the iteration step, $2m$ number of multiplications and m number of addition operations are saved. Considering all these operations in account, it can be found that overall saving per iteration is - $(4m+2)$ multiplications, $(2m+1)$ additions, 1 square-rooting and 2 divisions.

Using all these values of TC_* in the arithmetic operations savings as derived in the last paragraph, overall Transistor Saving (TS) can be computed as:

$$TS_{2D} = (4m+2)TC_M + (2m+1)TC_A + TC_{SQ} + 2TC_D \tag{6.107}$$

6.12.3 Multiplexer Penalty for $2D$ Architecture

The hardware saving obtained for CORDIC based $2D$ FastICA at (6.107), is mitigated somewhat by the insertion of the multiplexer arrays as shown in Fig. 6.14. Therefore, using Fig. (6.14) and subsection 6.12.1, overall penalty for CORDIC based $2D$ architecture (P_{2D}) can be given as:

$$P_{2D} = 4 * TC_{mux}^{2:1} \tag{6.108}$$

Since it has already been shown in Fig. 6.19 and discussed in 6.10, doubly pipelining approach does not include any multiplexer overhead within the architecture for storing micro-angle information, (6.108) does not contain this. However, as will be seen in the following subsections, any higher-dimensional CORDIC based FastICA will have this penalty as well.

6.12.4 Effective Hardware Saving for $2D$

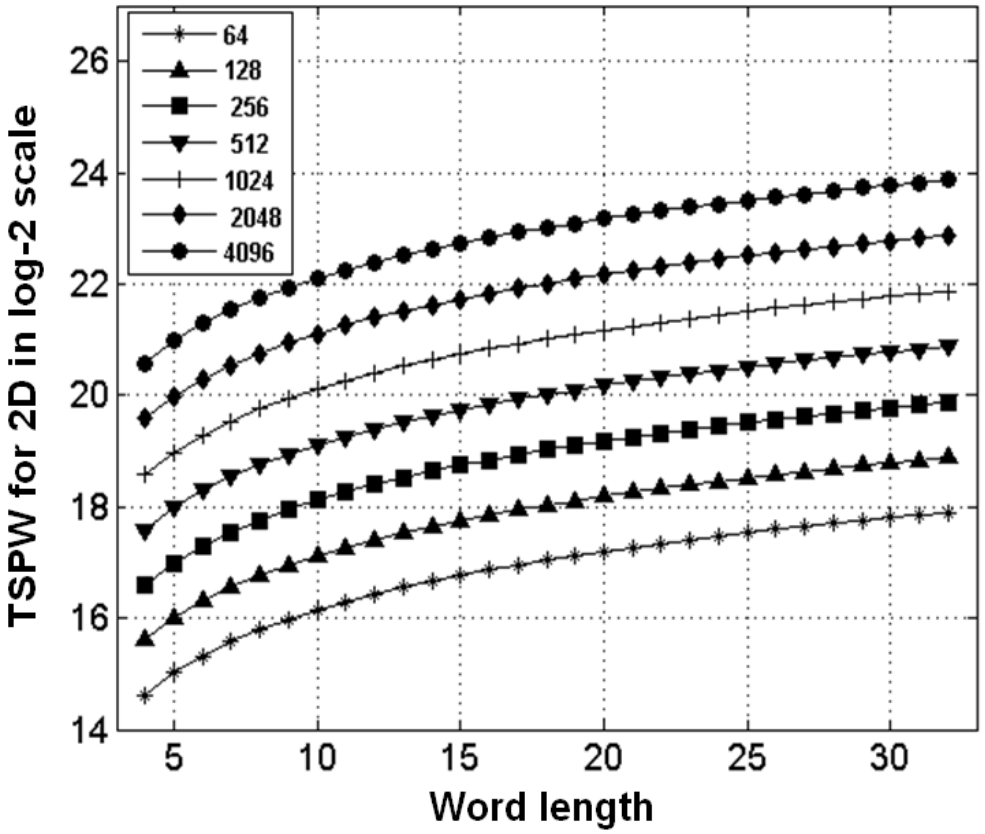


FIGURE 6.24: Variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm for $2D$ with Word-length and frame-length.

Effective hardware saving for the proposed CORDIC based $2D$ architecture (ES_{2D}), if

there is any, can be given by:

$$ES_{2D} = TS_{2D} - P_{2D} \tag{6.109}$$

where TS_{2D} and P_{2D} are obtained from (6.107) and (6.108) respectively. Using (6.12.1) in (6.109), ES_{2D} can be represented in terms of total number of transistors saved. Then effective TS obtained above for $2D$ case can be normalized with respect to b and a metric - Transistor Saving Per Word-length ($TSPW$) can be formulated using the approach presented in [86].

Being the function of m and b, Fig. 6.24 shows the variation of $TSPW$ for the Proposed $2D$ Architecture over [76] and [122] with respect to these parameters considering $64 \leq m \leq 4096$ and $4 \leq b \leq 32$. It can be observed from Fig. 6.24 that $TSPW$ for the Proposed Algorithm is significantly higher than that presented in [122] for higher m and b.

6.12.5 Performance Analysis for $3D$

Now considering the $3D$ FastICA case, in Iteration phase, comparing (6.12) with (6.19), it can be found that 3 multiplications and 2 addition operation is saved per sample per frame in each iteration. Therefore, for the whole frame m, total saving is - $3m$ multiplications and $2m$ additions. In Normalization phase, comparing (6.20) and (6.24)-(6.26), it can be observed that 3 multiplications, 2 addition, 1 square-rooting and 3 divisions operations are saved per iteration. In the output estimation phase, similar to the iteration step, $3m$ number of multiplications and $2m$ number of addition operations are saved. Considering all these operations in account, it can be found that overall saving per iteration is - $(6m + 3)$ multiplications, $(4m + 2)$ additions, 1 square-rooting and 3 divisions.

Following the aforementioned approach to derive (6.107), TS for $3D$ case can be given as follows:

$$TS_{3D} = (6m + 3)TC_M + 2(2m + 1)TC_A + TC_{SQ} + 3TC_D \tag{6.110}$$

6.12.6 Multiplexer Penalty for $3D$ Architecture

The hardware saving obtained for CORDIC based $3D$ FastICA at (6.110), is reduced somewhat by the insertion of the multiplexer arrays as shown in Fig. 6.15 and (6.20). Therefore, using Fig. (6.15) and 6.20 and subsection 6.12.1, overall penalty for CORDIC

based $3D$ architecture (P_{3D}) can be given as:

$$\begin{aligned} P_{3D} &= 4 * TC_{mux}^{2:1} + 6 * TC_{mux}^{2:1} + 2 * (k+1) * TC_{mux}^{2:1} \\ &= (10 + 2 * (k+1)) * TC_{mux}^{2:1} \end{aligned} \tag{6.111}$$

where, $(k+1)$ denotes total number of CORDIC micro-rotations as shown in Fig. 6.20. In (6.111), we have assumed TC of a multiplexer is same as that of a demultiplexer for brevity.

Comparing (6.111) and (6.108), it can be observed that P_{3D} has the extra overhead of $6 * TC_{mux}^{2:1}$ due to *Multiplexer Array-2* (as discussed in Section 6.9) and overhead of $2 * (k+1) * TC_{mux}^{2:1}$ because of the doubly pipelining approach (as described in Section 6.10) over the penalty P_{2D} for the proposed $2D$ architecture.

6.12.7 Effective Hardware Saving for $3D$

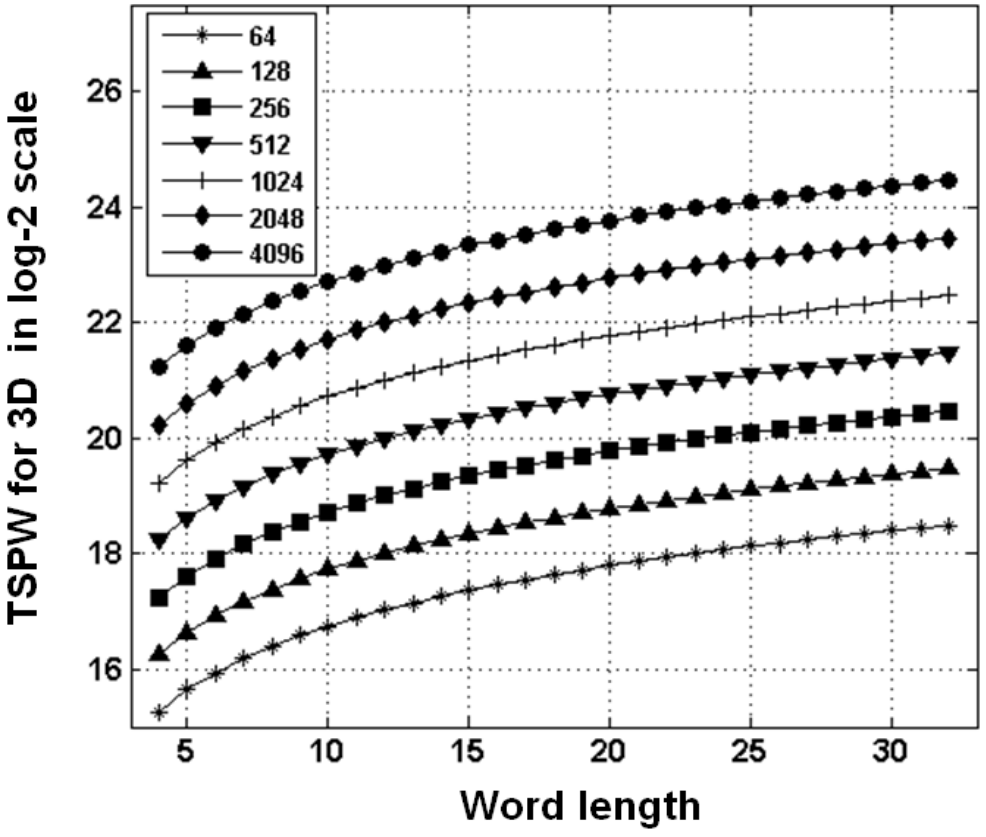


FIGURE 6.25: Variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm for $3D$ with Word-length and frame-length.

Following the same procedure adopted for the proposed $2D$ architecture above, effective hardware saving for the proposed CORDIC based $3D$ architecture (ES_{3D}), if there is

any, can be given by:

$$ES_{3D} = TS_{3D} - P_{3D} \tag{6.112}$$

where TS_{3D} and P_{3D} are obtained from (6.110) and (6.111) respectively. Using (6.12.1) in (6.112), ES_{3D} can be represented in terms of total number of transistors saved. Then effective TS obtained above for $3D$ case can be normalized with respect to b and a metric - Transistor Saving Per Word-length ($TSPW$) can be formulated using the approach presented in [86].

Being the function of m and b, Fig. 6.25 shows the variation of $TSPW$ for the Proposed $3D$ Architecture over the conventional $3D$ FastICA architecture (direct mapping of $3D$ FastICA algorithm into architecture) with respect to these parameters considering $64 \leq m \leq 4096$ and $4 \leq b \leq 32$. It can be observed from Fig. 6.25 that $TSPW$ for the Proposed Algorithm is significantly higher than the corresponding conventional architecture for higher m and b.

6.12.8 Performance Analysis for $4D$

Now considering the $4D$ FastICA case, in Iteration phase, comparing (6.30) with (6.38), it can be found that 4 multiplications and 3 addition operation is saved per sample per frame in each iteration. Therefore, for the whole frame m, total saving is - $4m$ multiplications and $3m$ additions. In Normalization phase, comparing (6.39) and (6.47)-(6.50), it can be observed that 4 multiplications, 3 addition, 1 square-rooting and 4 divisions operations are saved per iteration. In the output estimation phase, similar to the iteration step, $4m$ number of multiplications and $3m$ number of addition operations are saved. Considering all these operations in account, it can be found that overall saving per iteration is - $(8m + 4)$ multiplications, $(6m + 3)$ additions, 1 square-rooting and 4 divisions.

Following the approach mentioned before to derive (6.107) and (6.110), TS for $4D$ case can be given as follows:

$$TS_{4D} = (8m + 4)TC_M + 3(2m + 1)TC_A + TC_{SQ} + 4TC_D \tag{6.113}$$

6.12.9 Multiplexer Penalty for $4D$ Architecture

The hardware saving obtained for CORDIC based $4D$ FastICA at (6.113), is diminished somewhat by the insertion of the multiplexer arrays as shown in Fig. 6.16 and (6.21). Therefore, using Fig. (6.16) and 6.21 and subsection 6.12.1, overall penalty for CORDIC

based $4D$ architecture (P_{4D}) can be given as:

$$\begin{aligned} P_{4D} &= 4 * TC_{mux}^{2:1} + 5 * TC_{mux}^{3:1} + TC_{mux}^{2:1} + 2 * (k+1) * TC_{mux}^{3:1} \\ &= 5 * TC_{mux}^{2:1} + 5 * 2 * TC_{mux}^{2:1} + 2 * (k+1) * 2 * TC_{mux}^{2:1} \\ &= (15 + 4 * (k+1)) * TC_{mux}^{2:1} \end{aligned} \tag{6.114}$$

where, $(k+1)$ denotes total number of CORDIC micro-rotations as shown in Fig. 6.21. In (6.114), we have assumed TC of a multiplexer is same as that of a demultiplexer for brevity.

Comparing (6.114) and (6.108), it can be observed that P_{4D} has the extra overhead of $11 * TC_{mux}^{2:1}$ due to *Multiplexer Array-2* (as discussed in Section 6.9) and overhead of $4 * (k+1) * TC_{mux}^{2:1}$ because of the doubly pipelining approach (as described in Section 6.10) over the penalty P_{2D} for the proposed $2D$ architecture.

6.12.10 Effective Hardware Saving for $4D$

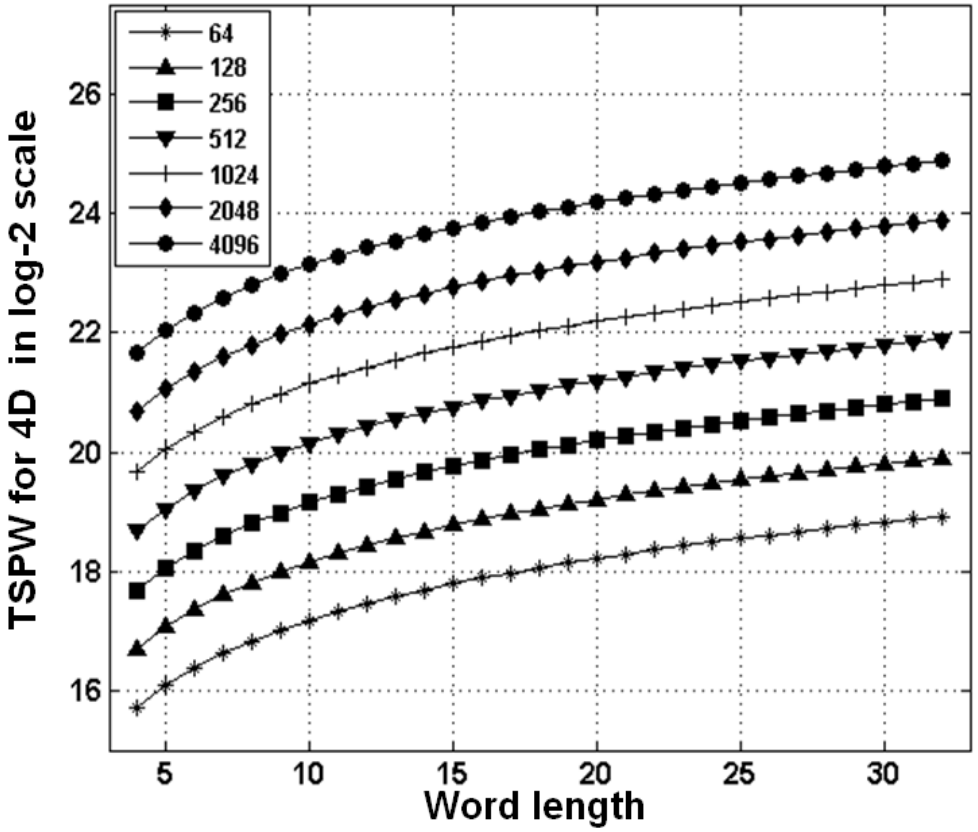


FIGURE 6.26: Variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm for $4D$ with Word-length and frame-length.

Following the same procedure adopted for the proposed $2D$ and $3D$ architectures above, effective hardware saving for the proposed CORDIC based $4D$ architecture (ES_{4D}), if

there is any, can be given by:

$$ES_{4D} = TS_{4D} - P_{4D} \tag{6.115}$$

where TS_{4D} and P_{4D} are obtained from (6.113) and (6.114) respectively. Using (6.12.1) in (6.115), ES_{4D} can be represented in terms of total number of transistors saved. Then effective TS obtained above for $4D$ case can be normalized with respect to b and a metric - Transistor Saving Per Word-length ($TSPW$) can be formulated using the approach presented in [86].

Being the function of m and b, Fig. 6.26 shows the variation of $TSPW$ for the Proposed $4D$ Architecture over the conventional $4D$ FastICA architecture (direct mapping of $4D$ FastICA algorithm into architecture) with respect to these parameters considering $64 \leq m \leq 4096$ and $4 \leq b \leq 32$. It can be observed from Fig. 6.26 that $TSPW$ for the Proposed Algorithm is significantly higher than the corresponding conventional architecture for higher m and b.

6.12.11 Performance Analysis for nD

Following the same procedure adopted to analyze the performance of the proposed $2D$, $3D$ and $4D$ FastICA architectures in the subsections above, here we will do the performance analysis of the proposed CORDIC based nD FastICA architecture in generalized fashion.

In the Iteration phase of the proposed nD FastICA architecture, as obtained from *Theorem-1* in Section 6.8, it can be found that n multiplications and $(n-1)$ addition operations are saved per sample per frame in each iteration. Therefore, for the whole frame m, total saving is - nm multiplications and $(n-1)m$ additions.

In nD Normalization phase, from *Theorem-2* in Section 6.8, it can be observed that n multiplications, $(n-1)$ addition, 1 square-rooting and n divisions operations are saved per iteration. In the output estimation phase, similar to the iteration step, nm number of multiplications and $(n-1)m$ number of addition operations are saved. Considering all these operations into account, it can be found that overall saving per iteration is - $(2m+1)n$ multiplications, $(2m+1)(n-1)$ additions, 1 square-rooting and n divisions.

Following the approach mentioned before to derive (6.107), (6.110) and (6.113), TS for nD case can be given as follows:

$$TS_{nD} = n(2m+1)TC_M + (n-1)(2m+1)TC_A + TC_{SQ} + nTC_D \tag{6.116}$$

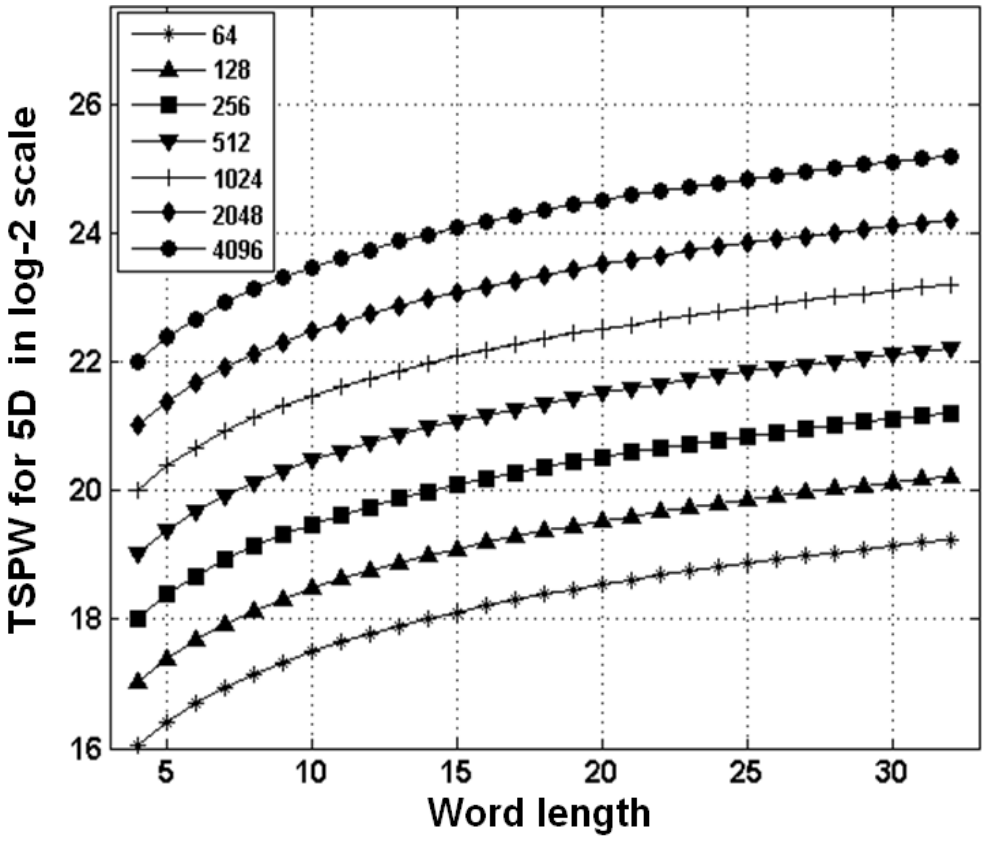


FIGURE 6.27: Variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm for $5D$ with Word-length and frame-length.

6.12.12 Multiplexer Penalty for nD Architecture

The hardware saving obtained for CORDIC based nD FastICA at (6.116), is reduced somewhat by the insertion of the multiplexer arrays as shown in Fig. 6.17 and (6.22). Therefore, using Fig. (6.17) and 6.22 and subsection 6.12.1, overall penalty for CORDIC based nD architecture (P_{nD}) can be given as:

$$\begin{aligned} P_{nD} &= 4 * TC_{mux}^{2:1} + 5 * TC_{mux}^{(n-1):1} + TC_{mux}^{2:1} + 2 * (k+1) * TC_{mux}^{(n-1):1} \\ &= 5 * TC_{mux}^{2:1} + 5 * (n-2) * TC_{mux}^{2:1} + 2 * (k+1) * (n-2) * TC_{mux}^{2:1} \\ &= (5 * (n-1) + 2 * (k+1) * (n-2)) * TC_{mux}^{2:1} \end{aligned} \tag{6.117}$$

where, $(k+1)$ denotes total number of CORDIC micro-rotations as shown in Fig. 6.22. In (6.117), we have assumed TC of a multiplexer is same as that of a demultiplexer for brevity.

Comparing (6.117) and (6.108), it can be observed that P_{nD} has the extra overhead of $5(n-2) * TC_{mux}^{2:1}$ due to *Multiplexer Array-2* (as discussed in Section 6.9) and overhead of $2 * (k+1)(n-2) * TC_{mux}^{2:1}$ because of the doubly pipelining approach (as described in Section 6.10) over the penalty P_{2D} for the proposed $2D$ architecture.

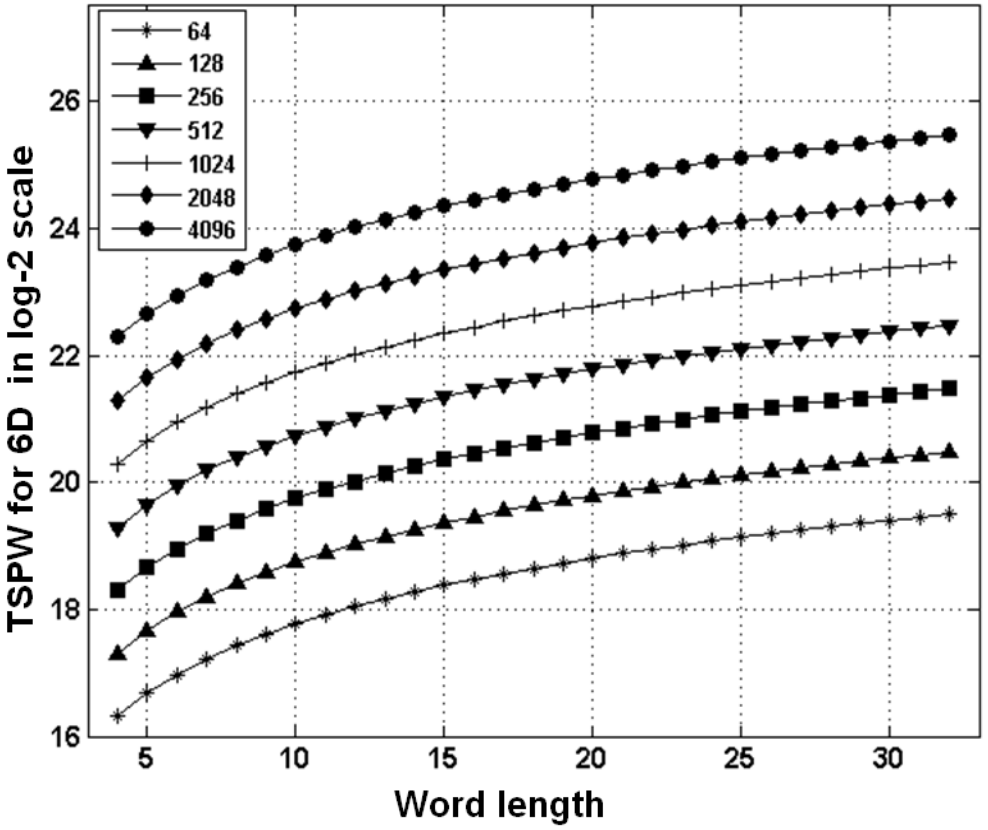


FIGURE 6.28: Variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm for $6D$ with Word-length and frame-length.

6.12.13 Effective Hardware Saving for nD

Following the same procedure adopted for the proposed $2D$, $3D$ and $4D$ architectures above, effective hardware saving for the proposed CORDIC based nD architecture (ES_{nD}), if there is any, can be given by:

$$ES_{nD} = TS_{nD} - P_{nD} \tag{6.118}$$

where TS_{nD} and P_{nD} are obtained from (6.116) and (6.117) respectively. Using (6.12.1) in (6.118), ES_{nD} can be represented in terms of total number of transistors saved. Then effective TS obtained above for nD case can be normalized with respect to b and a metric - Transistor Saving Per Word-length ($TSPW$) can be formulated using the approach presented in [86].

Considering, $n = 5$ and $n = 6$ in (6.118), $TSPW$ for the proposed $5D$ and $6D$ cases can be obtained. Being the function of m and b, Fig. 6.27 and 6.28 show the variation of $TSPW$ for the Proposed $5D$ and $6D$ Architectures over the conventional FastICA architectures (direct mapping of $5D$ and $6D$ FastICA algorithms into architectures) with respect to these parameters considering $64 \leq m \leq 4096$ and $4 \leq b \leq 32$. It can be

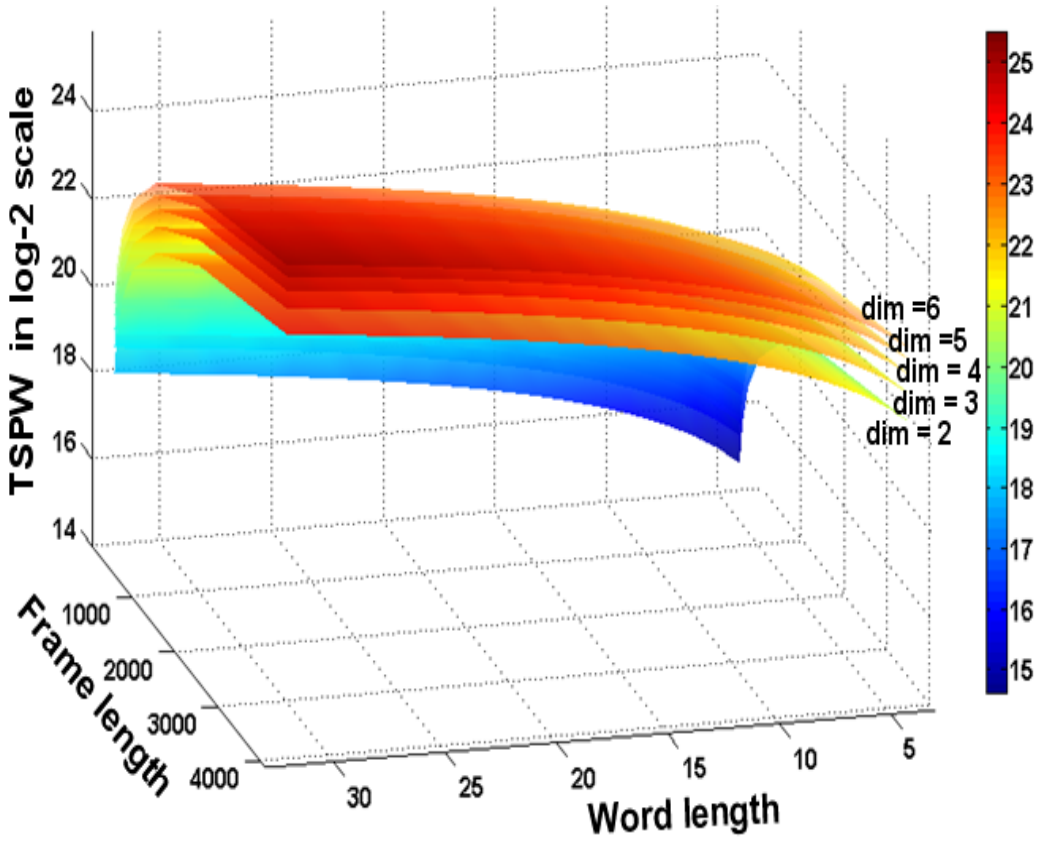


FIGURE 6.29: Comparative variation of Transistor Saving Per Word-length ($TSPW$) of the proposed algorithm with Frame-length and Word-length for $2D$ to $6D$ (denoted by dim = 2 to dim = 6).

observed from these figures that, like Fig. 6.24 - 6.26, here also $TSPW$ for the Proposed Algorithm is significantly higher than the corresponding conventional architectures for higher m and b.

Fig. 6.29 shows the comparative $TSPW$ variation analysis for the proposed $2D$ to $6D$ FastICA cases with respect to different frame-length ($64 \leq m \leq 4096$) and word-length ($4 \leq b \leq 32$). It can be observed from Fig. 6.29 that $TSPW$ for the Proposed Algorithm increases even with the increase in dimension n as well with the increase of frame-length m and word-length b.

6.13 Algorithm Validation

To validate the proposed algorithm, C models have been generated for both the conventional [77] as well as the proposed co-ordinate rotation based $2D$ and $3D$ FastICA and compared the performance of the estimated outputs.

Left hand side of Fig. 6.30 and 6.31 plot the outputs from the conventional FastICA and right hand side shows the outputs from the proposed CORDIC based $2D$ and $3D$

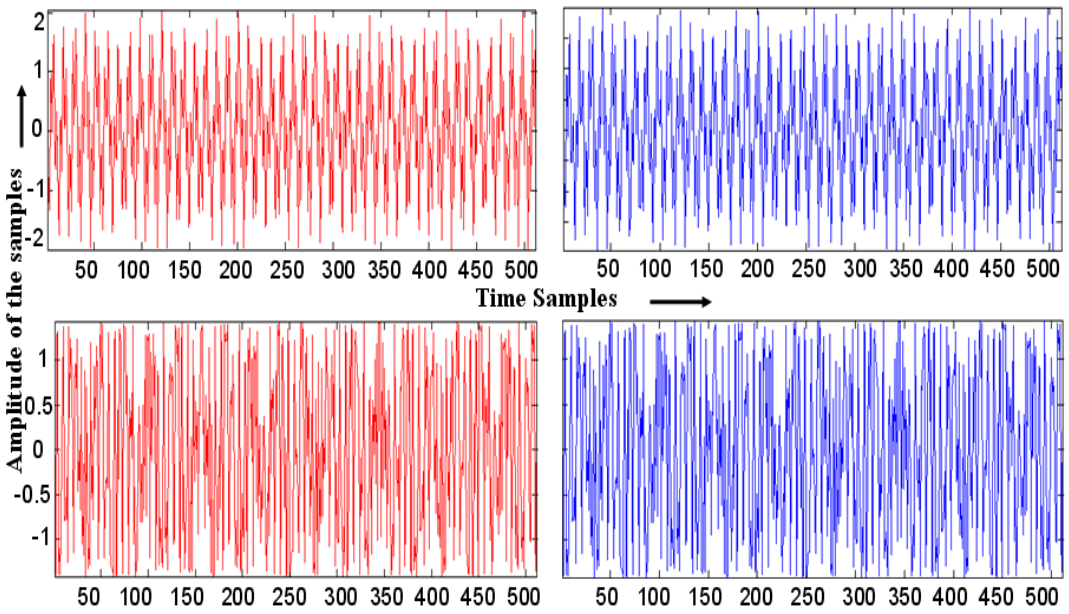


FIGURE 6.30: Left side - estimated waveforms from conventional $2D$ FastICA, right side - estimated waveforms from proposed co-ordinate rotation based $2D$ FastICA. Source signals were - $0.7\sin(450t)\sin(40t)$ and $0.5\sin[500t + 5\cos(60t)]$ as obtained from [122].

FastICA respectively. It is evident from these figures that the proposed co-ordinate rotation based $2D$ and $3D$ FastICA algorithms maintain the same functionality as that of the conventional FastICA algorithm without sacrificing its algorithmic efficiency.

6.14 Concluding Remarks

In this chapter co-ordinate rotation concept based nD FastICA algorithm has been introduced and its corresponding architecture has been presented. It has also been shown that this algorithm is capable of reducing hardware complexity significantly by reusing the CORDIC unit in both preprocessing as well as in the FastICA Iteration stage and this has been established with a comparative study with the available architectures. It has also been proved here that unlike the already reported approaches in [76] and [122], the proposed $2D$ FastICA algorithm is not based on the algebraic approach and thus it has been shown how this concept can be extended in higher dimensions ($n > 2$) by exploiting the geometry of the algorithm. Performance of the proposed architectures has been analyzed in terms of hardware complexity and it has been proved that the proposed architectures provide significantly higher hardware saving if compared with the conventional direct-mapping based FastICA architectures. Moreover, to obtain better architectural performance, some redundant computations have been identified and eliminated and further optimized architectures are introduced for the proposed CORDIC based nD FastICA algorithm.

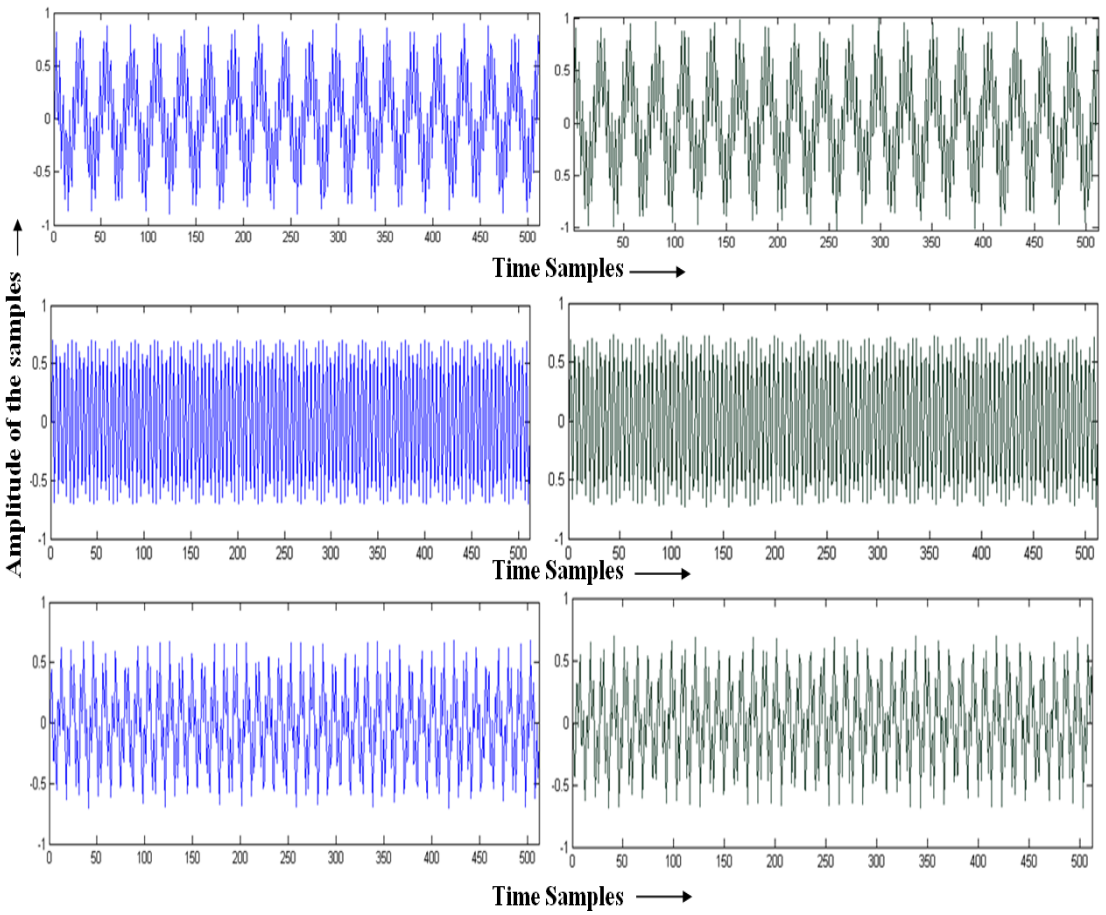


FIGURE 6.31: Left side - estimated waveforms from conventional $3D$ FastICA, right side - estimated waveforms from proposed co-ordinate rotation based $3D$ FastICA. Source signals were - (i) top: $0.9\sin(800t)\sin(60t)$, (ii) middle: $0.7\sin(90t)$, (iii) bottom: $0.7\sin(450t)\sin(40t)$

To the best of out knowledge the proposed CORDIC based algorithms and architectures are the first of its kind in the field of low-complexity design of FastICA and have the potential to open up a new application domain of CORDIC.

Chapter 7

Conclusions and Future Research

Sophisticated signal processing techniques will find potential applications in the emerging areas of WSN, remote health monitoring and pervasive computing. However traditional signal processing, in its present form, is not suitable for such applications which are constrained by the scarcity of resources in terms of low area and power consumption. It necessitates investigation into the traditional signal processing techniques from the perspective of algorithm-architecture holistic optimization so that it would become suitable for such resource constrained applications. This has been the focus of this research study as detailed in Chapter 1.

In principle, the sensors used in above-mentioned applications capture signals which are often corrupted by noise and mixed with several other signals. Section 1.1 in Chapter 1 identified the efficient signal processing algorithms used for denoising and signal separation are WT and ICA. Despite being algorithmically efficient, these algorithms are computationally intensive and their direct architectural mapping is not suitable for fore-mentioned resource constrained applications. In this thesis, therefore, the research aim has been to investigate WT and ICA and propose their low complexity algorithms and architectures for their efficient hardware implementation.

In the next section research contribution of the thesis is summarized and Section 7.2 highlights how this research study has created new avenues for further research in some challenging fields. Overall thesis contribution and future research challenges together with research aim are also depicted in Fig. 7.1.

7.1 Thesis Contributions

In Chapter 2, a novel memory reduction methodology for Distributed Arithmetic based multiplierless DWT/IDWT has been proposed. By a comparative study with the existing architectures (see Table 2.3 on page 31), it has been proved that the proposed

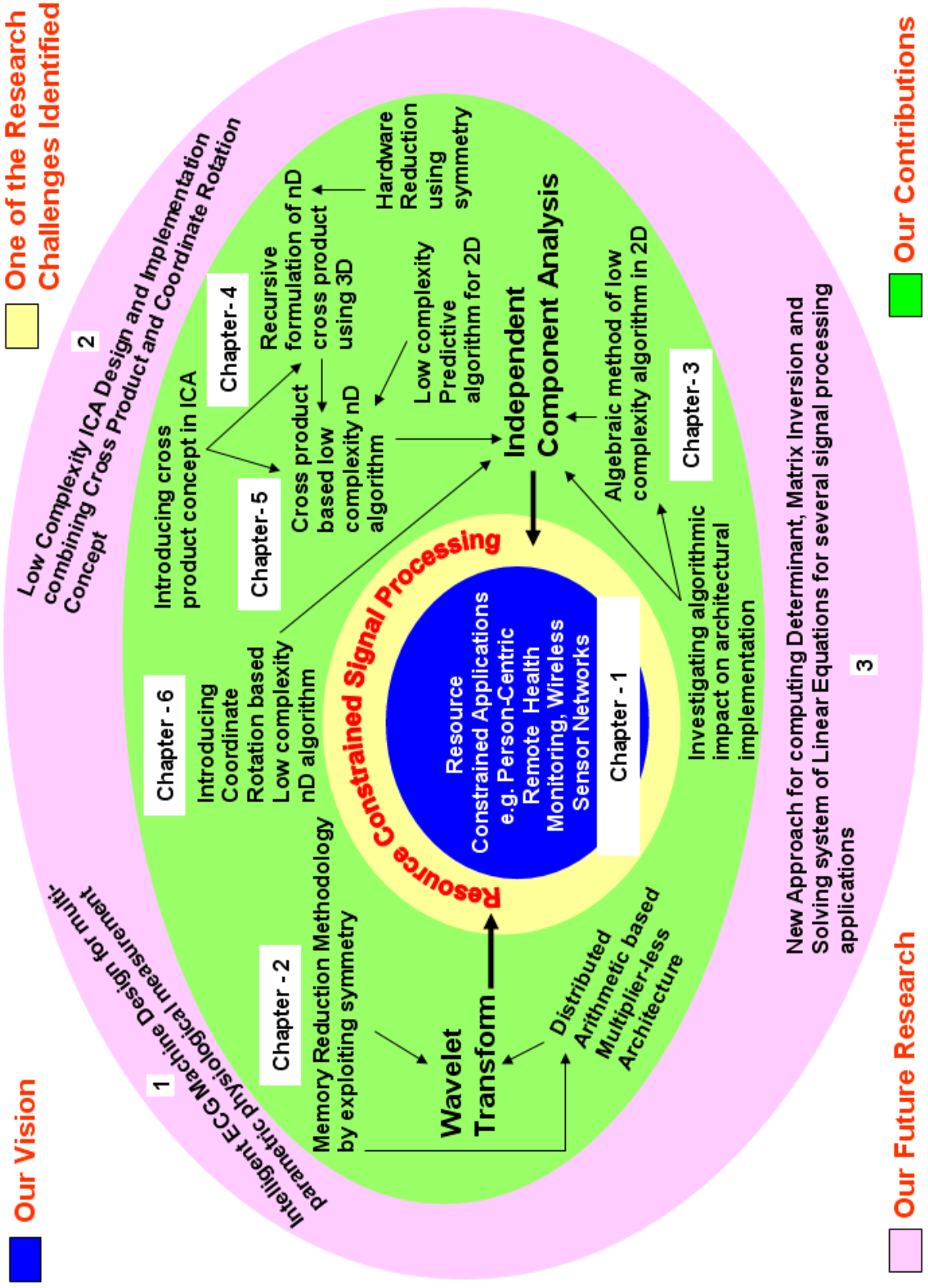


FIGURE 7.1: Thesis overview: Our vision and research challenges identified in Chapter-1, Our research contributions as described in Chapter 2 to Chapter 6 and the Future Research identified in Chapter 7.

methodology based architecture consumes significantly less area and power. It has also been pointed out that the proposed methodology is generic and independent of any particular wavelet function.

In Chapter-3 FastICA has been identified as the most popular among the existing ICA algorithms in terms of convergence speed and accuracy and it has been pointed out that despite its algorithmic efficiency, its direct mapping into hardware is not suitable for resource constrained environment. So the research aim was to propose a novel FastICA techniques which would be architecturally efficient in terms of low power consumption and low hardware complexity without sacrificing its algorithmic efficiency. But prior to that it has been realised that for efficient implementation of any ICA algorithms, it is important to investigate the impact of its algorithmic parameters on the corresponding architectures. It has been noted in Chapter 3 that such an important study is missing in the available literatures. Investigating such impact on the respective architectural implementation, a hardware design procedure has been presented which can be used as a guideline by the hardware designer of FastICA. Subsequently a novel algebraic method of hardware reduction for the basic 2-dimensional FastICA and proved that it consumes significantly low area and power (see Table 3.1 on page 58).

In the chapter followed after that generic n-dimensional FastICA has been explored. Concept of the cross product has been introduced for the first time in the context of ICA in Chapter 4. Since, to the best of our knowledge, this is the first attempt to bring such concept in this domain, it was not a surprise that no generic architectural model was present on the hardware implementation of Cross product. Motivated by this fact a generalized algorithm and architecture for n-dimensional cross product computation in hardware using the fundamental 3-dimensional vector cross-product computation unit recursively has been proposed in Chapter-4. Subsequently at the latter half of that chapter a concept of further hardware reduction strategy by exploiting symmetry among the mathematical structures of the cross product in different dimensions has been introduced (see Table 4.2 on page 98).

Motivated by the concept presented in Chapter-4, a novel cross-product based low complexity FastICA algorithm has been proposed in Chapter 5. In the same chapter a new concept of hardware reduction for 2-dimensional FastICA by exploiting the algorithmic redundancies has also been introduced (see Fig. 5.5, 5.6 on page 113 and 115 respectively). The hardware complexity and computational delay analysis proved that the proposed cross product based FastICA has significantly less hardware complexity and higher computational speed (see Fig. 5.7 and 5.8 on page 118 and 120 respectively).

In Chapter 6, the concept of Coordinate rotation has been introduced in the domain of FastICA implementation and a novel n-dimensional generalized CORDIC based FastICA algorithm and architecture has been proposed. It has been shown there that such approach is capable of reusing the same hardware units by merging the Pre-processing

stage and the FastICA Iteration stage together. By hardware complexity analysis it has also been proved that the proposed approach has significantly less complexity when compared with the conventional FastICA (see Fig. 6.29 on page 188).

Fig. 7.1 clearly shows the research contributions of this thesis and also portrays the link among them. Each contribution in this thesis has been substantiated with several experiments. It is hoped that the outcome of this research will contribute towards the growth of the signal processing circuits and systems research community and the future research challenges, as will be discussed next, will motivate researchers to investigate further in these fields.

7.2 Future Research

The following subsections overview the future research challenges identified by the research study undertaken in this thesis.

7.2.1 Intelligent ECG Machine Design for Multi-parametric Physiological Measurement

Fig. 7.2 clearly indicates the massive growth of the Cardio Vascular Diseases (CVD) among the whole population of several industrialized countries [165] - [181]. This makes the early detection and prevention of such diseases extremely important. Since electrocardiogram, vastly known as ECG, shows the electrical characteristics of heart of a person, may be considered as a prime candidate for designing an electronic system for such early detection and prevention of CVD. Significant amount of research has already been done on this and many of them have been reported in literatures also. However, if proper healthcare has to be provided to the people in their home, then the practical challenge becomes to minimize technical intervention on their daily lives yet to provide them the better healthcare. So, although theoretically different sensors can be designed and developed for acquiring various physiological signals, all of these can not be deployed on patient's body because these may restrict their free movement and eventually will become a burden on them. This kind of system thus will not become a sustainable system. Motivated by this fact and inspired by our initial research finding, only two or three ECG sensors can be envisaged which will not only collect and process the ECG data, but will also help us processing and diagnosing some other important physiological parameters e.g. blood pressure, blood serum potassium level etc.. This will require further sophisticated low power and tiny sensors design with embedded resource constrained signal processing capabilities.

Recently, as our initial research study on this domain, we have applied the wavelet transform, as discussed in Chapter 2 and some very interesting preliminary results are

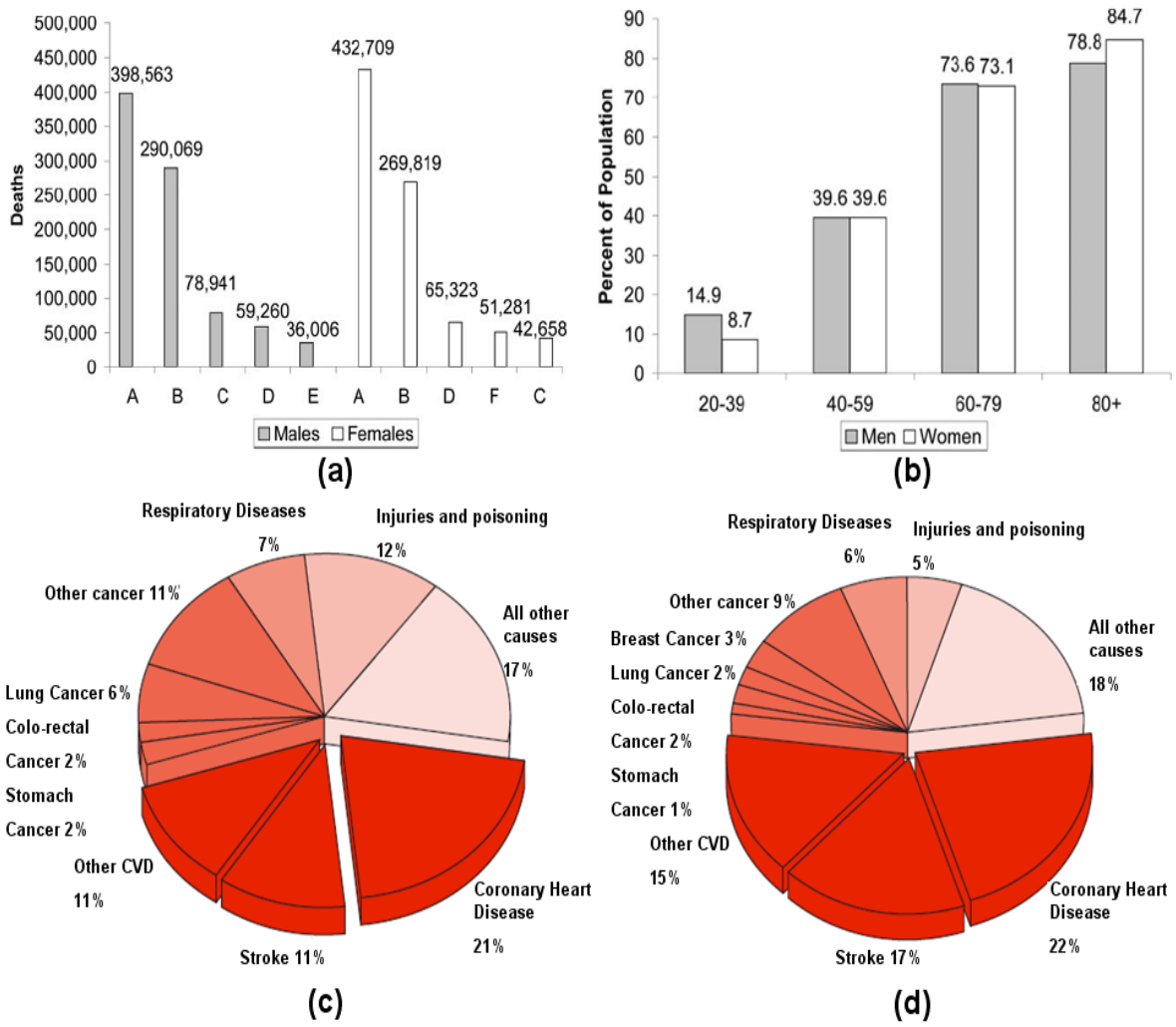


FIGURE 7.2: (a) CVD and other major causes of death for all males and females (United States, 2006). A: CVD plus congenital CVD; B: Cancer; C: Accidents; D: Chronic Lower Respiratory Disease (CLRD); E: Diabetes and F: Alzheimer's Disease [165], (b) Prevalence of CVD in adults $\geq$ 20 years of age "by age and sex" [165], (c) Deaths by cause, men, year 2008 in Europe [166], (d) Deaths by cause, women, year 2008 in Europe [166].

obtained as shown in Fig. 7.3. These results show that different CVD may show some different characteristics in frequency domain if proper time-frequency localization is done and thus provide some clinically valuable information from diagnostic point of view. Since these initial experiments have shown some interesting results, one of the future research plans is to continue along this line and design an intelligent ECG machine capable of measuring several physiological parameters. The initial results of this study can be found in [182].

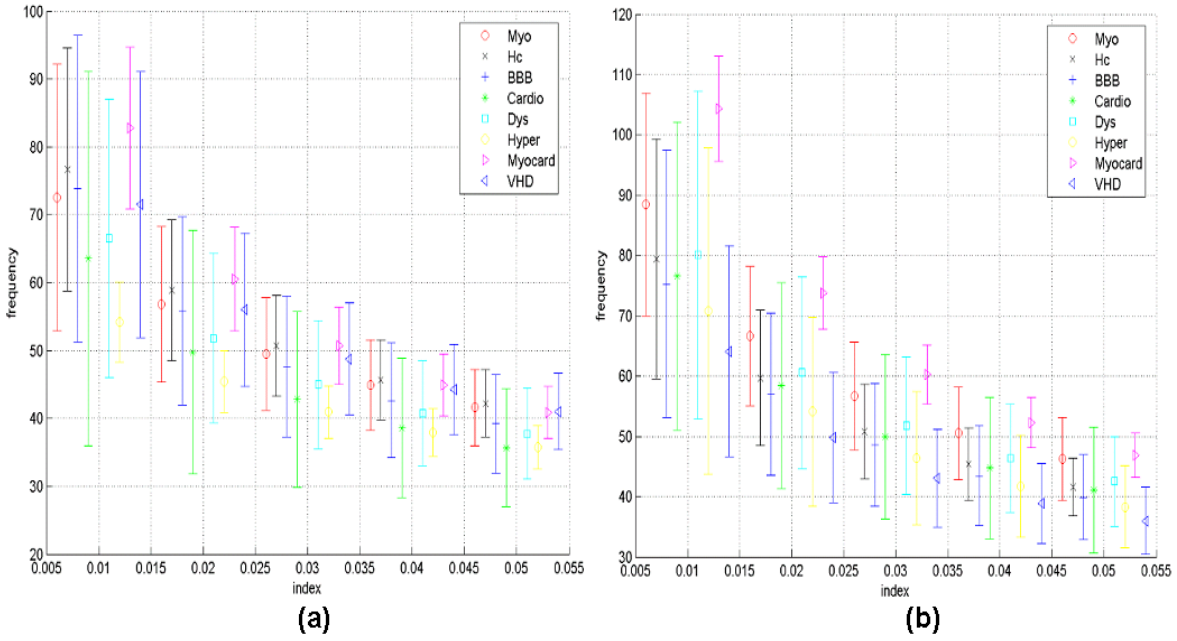


FIGURE 7.3: Mean frequency variation for 8 different cardio vascular diseases for different patients at different power spectral density distribution point for (a) Lead - i and (b) Lead - ii of standard ECG measurement method. Error bars indicate the standard deviation across the mean frequency. Definition of abbreviations used: Myo = Myocardial Infarction, HC = Healthy Controls, BBB = Bundle Branch Block, Cardio = Cardiomyopathy/ Heart Failure, Dys = Dysrhythmia, Hyper = Myocardial Hypertrophy, Myocard = Myocarditis, VHD = Valvular Heart Disease.

7.2.2 Low Complexity ICA Design Combining Cross Product and Coordinate Rotation Concept

Implementation of low complexity ICA on silicon using the combination of cross-product concept (described in Chapter 5) and Coordinate Rotation concept (discussed in Chapter 6) is identified as one of the future research plans. It was discussed in Chapter 6 that the coordinate rotation digital computer concept can be adopted to design low complexity ICA architectures by merging two main steps of FastICA. It was shown in Chapter 5 that the cross-product concept helps minimizing the hardware requirement of the conventional nD FastICA significantly by removing the need of the n^{th} iteration stage completely. So if these two concepts can be combined together, further hardware reduction can be achieved. In that case, CORDIC concept can be applied to design $(n-1)$ iteration stages of the conventional nD FastICA whereas the n^{th} stage will be manipulated using the concept of cross product.

7.2.3 New Approach for Computing Determinant, Matrix Inversion and Solving System of Linear Equations

Motivated by the concept presented in Chapter 4 about the generalized algorithm and architectures for n-dimensional cross product implementation, we started investigating different application domains of this approach. Our initial research, as shown in [183], shows that similar concept can be adopted to design computationally efficient n-dimensional Determinant computation algorithms and architectures using the 2-dimensional computation unit as the basic kernel. The concept of symmetry based approach, as presented in Chapter 4, can also be applied here for further hardware reduction. Since matrix inversion problems and solving systems of linear equations need computation of determinant, we believe, such approach may be very beneficial in terms of low hardware complexity and high computational speed. Matrix inversion problem and linear equations are encountered very often in different signal processing application such as Principal Component Analysis (PCA) and Eigen Value Decomposition (EVD). So it is believed, this work has potential to propose hardware efficient PCA and EVD techniques.

Appendix A

Publications

Journal Article

1. Amit Acharyya, Koushik Maharatna, Bashir M. Al-Hashimi and Steve R. Gunn, "Memory Reduction Methodology for Distributed Arithmetic Based DWT/ IDWT Exploiting Data Symmetry", *IEEE Transactions on Circuits and Systems - II: Express Briefs*, pp. 285-289, vol. 56, no. 4, 2009.

2. Amit Acharyya, Koushik Maharatna and Bashir M. Al-Hashimi, "Algorithm and Architecture for N-D Vector Cross-Product Computation", *IEEE Transactions on Signal Processing*, pp. 812-826, vol. 59, no. 2, 2011.

3. Amit Acharyya, Koushik Maharatna and Bashir M. Al-Hashimi, "Co-ordinate Rotation based Low Complexity N-D FastICA Algorithm and Architecture", *IEEE Transactions on Signal Processing* (accepted with minor revision)

Conference Contribution

1. Amit Acharyya, Koushik Maharatna and Bashir M. Al-Hashimi, "Hardware Development for Pervasive Healthcare Systems: Current Status and Future Directions", *IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1304-1307, 30 November-3 December, 2008, Macao, China.

2. Amit Acharyya, Koushik Maharatna, Jinhong Sun, Bashir M. Al-Hashimi and Steve R. Gunn, "Hardware Efficient Fixed-Point VLSI Architecture for 2D Kurtotic FastICA", 19^{th} *IEEE European Conference on Circuit Theory and Design*, pp. 165-168, 23-27 August, 2009, Antalya, Turkey.

3. Amit Acharyya, Koushik Maharatna and Bashir M. Al-Hashimi, "Hardware Reduction Methodology for 2-Dimensional Kurtotic FastICA", *IEEE Workshop on Signal Processing Systems*, pp. 69-74, 7-9 October, 2009, Tampere, Finland.

4. Amit Acharyya, Koushik Maharatna and Bashir M. Al-Hashimi, "Co-ordinate Rotation Based Low Complexity 2D FastICA Algorithm and Architecture", *IEEE International Conference on Green Circuits and Systems*, pp. 60-64, 21-23 June, 2010, Sanghai, China.

5. Amit Acharyya, Hasitha Tudugalle, Koushik Maharatna, Bashir M. Al-Hashimi and Steve R. Gunn, "VLSI Architecture for Fetal ECG Extraction for Personalized Healthcare Application within Resource Constrained Environment", *In Proceedings of the UK Electronics Forum*, 30 June - 1 July, 2010, Newcastle-upon-Tyne, United Kingdom.

6. Amit Acharyya, Sayanta Mondal, Koushik Maharatna and Bashir M. Al-Hashimi, "Automated and Robust Channel Identification Algorithm and Architecture to Solve Permutation Problem of ICA for Artifacts Removal from ECG in Remote Health Monitoring Environment", *In Proceedings of the UK Electronics Forum*, 30 June - 1 July, 2010, Newcastle-upon-Tyne, United Kingdom.

7. Amit Acharyya, Koushik Maharatna, Bashir M. Al-Hashimi and Sayanta Mondal , "Robust Channel Identification Scheme: Solving Permutation Indeterminacy of ICA for Artifacts Removal from ECG", 32^{nd} *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1142-1145, 31 August-4 September, 2010, Buenos Aires, Argentina.

8. Amit Acharyya, Koushik Maharatna, Bashir M. Al-Hashimi and Hasitha Tudugalle, "Simplified Logic Design Methodology for Fuzzy Membership Function based Robust Detection of Maternal Modulus Maxima Location : A Low Complexity Fetal ECG Extraction Architecture for Mobile Health Monitoring Systems", *The IEEE International Symposium on Circuits and Systems*, 15-18 May, 2011, Rio de Janeiro, Brazil. (in press)

Appendix B

A Low Complexity Fetal ECG Architecture for Personalized Mobile Healthcare

It has already been mentioned in Section 2.9 in Chapter-2 that the extraction of FECG from the composite signal is a challenging problem in signal processing. One robust method was reported in [69] to extract FECG. But this method deals with complex signal processing tasks including wavelet transform, fuzzification, reconstruction (shown in Fig. B.1) which increase the complexity of the circuit in terms of extensive memory and power requirement[1]. Hence, despite being efficient, we believe direct mapping of this method into architecture is not suitable for the envisaged person-centric mobile home-care medical device which operates on battery-backup enforcing a resource constrained environment in terms of low area and power consumption.

In this appendix, first we propose a simplified logic design methodology for the complex Fuzzy membership function and subsequently present an architecture for FECG extraction from the abdominal composite. The simplified architecture eliminates the necessity of complex fuzzification by exploiting the inherent "time-position information" within the wavelet coefficients while keeping the robustness and reliability of the original fuzzy membership function intact. Since this simplified logic design methodology to be proposed can be implemented only using simple comparison logic, it is capable of reducing the hardware complexity significantly. On the other hand, the novelty of the FECG extraction architecture is its memory-efficient, multiplierless design based on our recently proposed reduced complexity Discrete Wavelet Transform (DWT) and Inverse DWT

[1]The contents of this Appendix have partly appeared as "VLSI Architecture for Fetal ECG Extraction for Personalized Healthcare Application within Resource Constrained Environment" in *Proceedings of the UK Electronics Forum - 2010* and as "Simplified Logic Design Methodology for Fuzzy Membership Function based Robust Detection of Maternal Modulus Maxima Location : A Low Complexity Fetal ECG Extraction Architecture for Mobile Health Monitoring Systems" in *IEEE International Symposium on Circuits and Systems* by Acharyya *et. al.*. Please see Appendix-A for further detail.

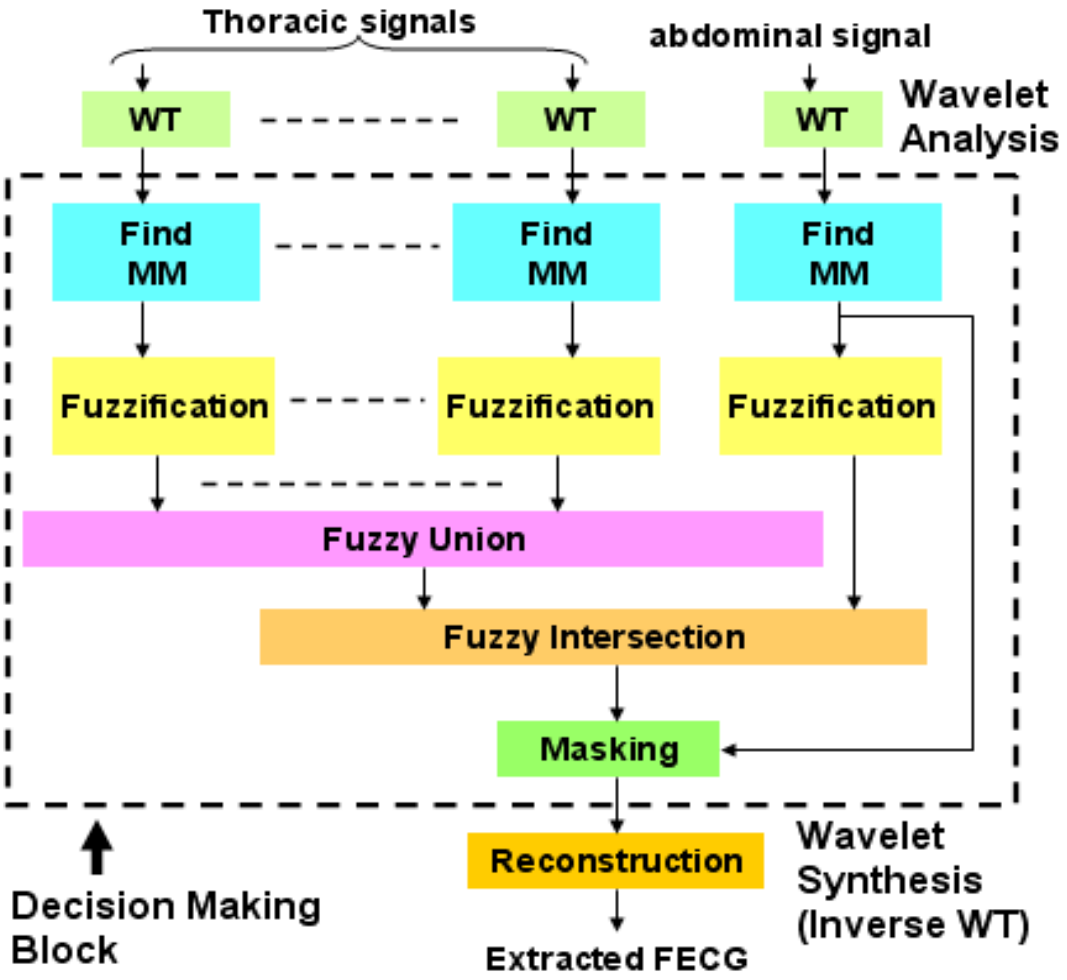


FIGURE B.1: Overview of FECG extraction method [69]. WT and MM indicate Wavelet Transform and Modulus Maxima respectively.

(IDWT) method reported in [86] and detailed in Chapter-2. Moreover, since there is no

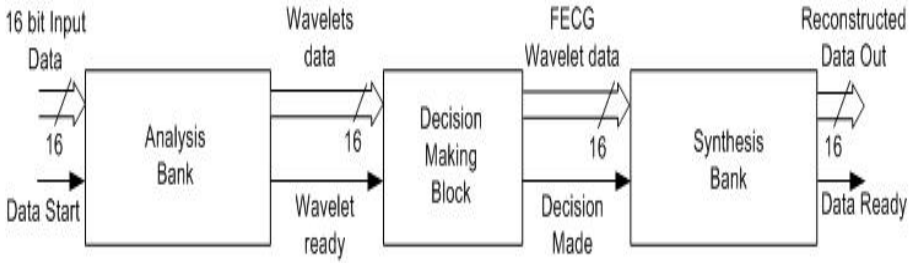


FIGURE B.2: Block diagram of FECG extraction method.

intuitive way to know which mother wavelet to choose [70], the proposed design lets the user freedom to choose any orthonormal wavelet suitable for the application by simply reconfiguring the generic memory unit.

The rest of this appendix is organized as follows. Section B.1 provides the necessary background for FECG extraction including the concept of fuzzy membership function, Section B.2 proposes the simplified logic design methodology of the fuzzy membership

function, Section B.3 presents the complete VLSI architecture for FECG extraction system and Section B.4 analyses the performance of the design and concludes this appendix.

B.1 Preliminaries

B.1.1 FECG Extraction Strategy

The concept of FECG extraction from abdominal composite signals has been adopted from [69] and shown in Fig. B.1. This method is based on the locations of the singular points but not on the amplitude at those points. It employs signal characterization, singular point detection and reconstruction of FECG based on wavelet transform [2]. Fig.B.2 presents the fundamental computational units required for FECG extraction method discussed in [69]. The method uses maternal thoracic signals and abdominal composite as the inputs and decomposes these into wavelet coefficients. Then the corresponding modulus maxima are found. The properties of wavelet transform have been utilised here because in an ECG signal, the modulus maxima of wavelet coefficients provides important information about the principal characteristic points and waveform. Next step is the comparison of the modulus maxima locations of thoracic and abdominal signal at all scales. Then the common ones are decided in favour of MECG. Once MECG is detected, the remaining set of modulus maxima is used to reconstruct the FECG.

B.1.2 Fuzzy Membership Function

The moment when wavelet decomposition is over and the modulus-maxima are about to be determined, a problem is often encountered which is to detect the maternal modulus maxima accurately. To address this issue, a triangular fuzzy membership function based robust and reliable technique has been devised in [69] and shown in Fig. B.1. This function helps to average out the error due to misalignment of the position of the modulus maxima. Integration of all available information from different thoracic channels is done by computing the union of the fuzzy sets associated with the modulus maxima locations of these signals (Fig. B.1). For the abdominal channel and each maternal thoracic channel different two-dimensional functions are defined. The maternal modulus maxima fuzzified locations are expressed by the unions of these functions computed from each maternal signal. Once such union operation is over, a mask function is designed to mask out the maternal modulus maxima in the abdominal composite (Fig. B.1). The mask function is determined by applying intersection operation to the maternal and abdominal modulus maxima fuzzified locations. Similarly a fetal mask is defined which is derived from the previous mask function. Comparing these two functions modulus maxima belong to FECG is identified and those belong to maternal signal are removed

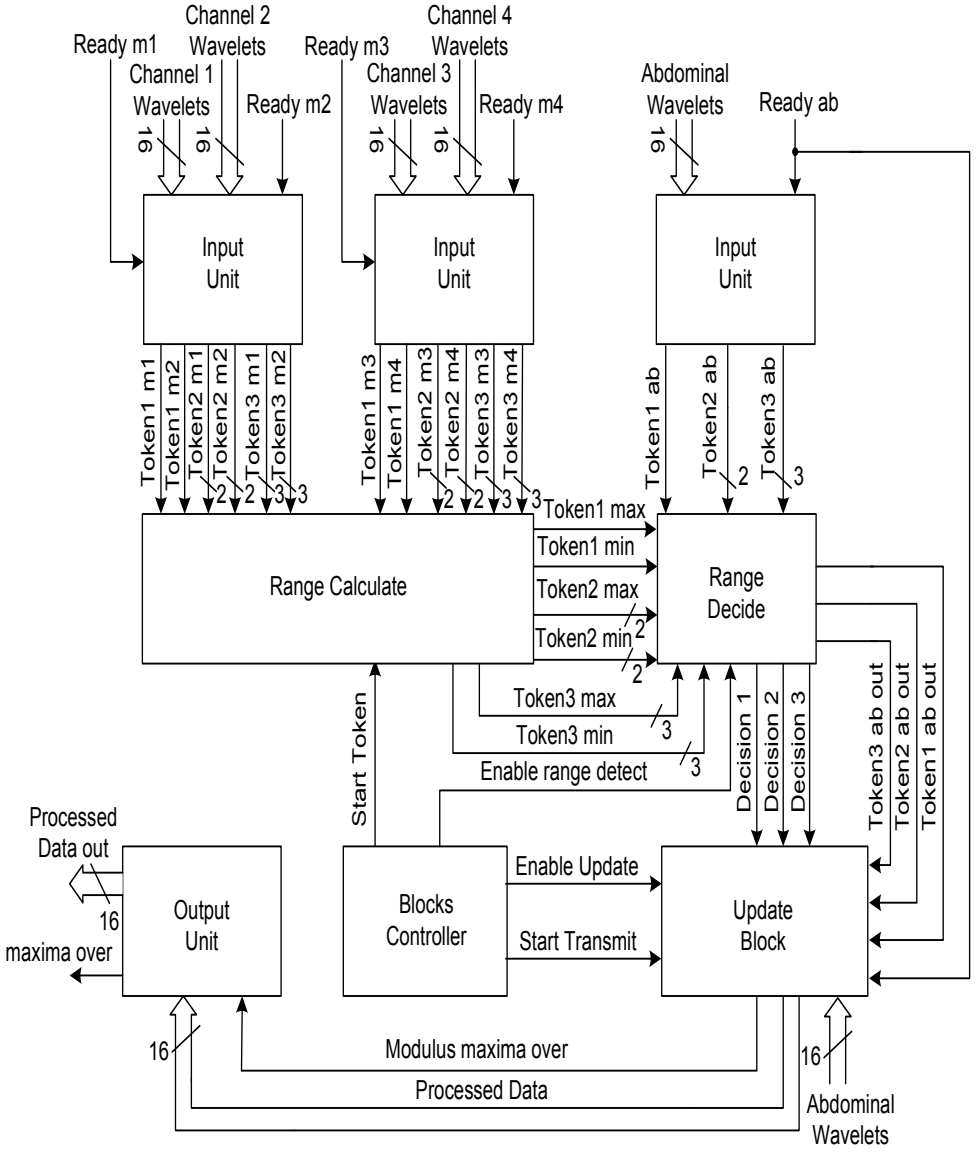


FIGURE B.3: Decision Making Block for Modulus-maxima computation.

from the composite. Next is the update the wavelet coefficient values and reconstructs the FECG signal.

B.2 Proposed Logic Design for Fuzzy Membership Function : Simplified Architecture

Considering four maternal thoracic channels and one abdominal channel, this section deals with the wavelet coefficients obtained after wavelet decomposition within each analysis bank corresponding to every channel and proposes a simplified logic equivalent

TABLE B.1: Token attributes for different wavelet resolution levels

Wavelet Resolution Level-1 : Token 1	
Wavelet Coefficient	*Token1* Value
$w_{1,0}$	000
$w_{1,1}$	001
$w_{1,2}$	010
$w_{1,3}$	011
$w_{1,4}$	100
$w_{1,5}$	101
$w_{1,6}$	110
$w_{1,7}$	111
Wavelet Resolution Level-2 : Token 2	
Wavelet Coefficient	*Token2* Value
$w_{2,0}$	00
$w_{2,1}$	01
$w_{2,2}$	10
$w_{2,3}$	11
Wavelet Resolution Level-3 : Token 3	
Wavelet Coefficient	*Token3* Value
$w_{3,0}$	0
$w_{3,1}$	1

design for the fuzzy membership function based robust and reliable detection of maternal modulus maxima locations within the abdominal composite discussed in Section B.1. Such simplification is achieved by assigning simple "*Token*" attribute to each wavelet within a specific resolution level. This particular unit is denoted by *Decision Making Block* in Fig.B.2. The overall architecture of this block is shown in Fig. B.3. It comprises of four main design blocks - (i) Input Unit, (ii) Range Calculate Block, (iii) Range Decide Block and (iv) Update Block. The simplified logic design of the fuzzification is the cumulative effect of the computations involved in each of these fore-mentioned blocks which are discussed in the following subsections.

B.2.1 Input Unit: Token Assignment

This unit receives wavelet coefficients from the analysis banks corresponding to each channel and computes *modulus maxima* per wavelet resolution level per channel by employing simple comparison logic. Since 16-point DWT is considered for designing the architecture, it consists of four such resolution levels and each level produces half the number of wavelets compared to the previous level and the final level produces one wavelet along with one residue signal as output.

Simple *Token* attribute is used to code the position of the wavelet coefficients and thus it stores the corresponding "*time information*" of each wavelet coefficient. The output of the first level has eight wavelets which needs three bits to identify each of these wavelets and so *Token1* has been assigned with three bits. As for example, $Token1$ = "010"

TABLE B.2: Pseudocode for Token 1, 2 and 3 computation

***Token1* Computation : Wavelet Resolution Level - 1** (Fig. B.4)
block 0:
$w'_{1,1} <= max(w_{1,0}; w_{1,1})$;
$if(w_{1,0} > w_{1,1}) : Token_{1,1} <= 000$; $otherwise : Token_{1,1} <= 001$;
block 1:
$w'_{1,2} <= max(w'_{1,1}; w_{1,2})$;
$if(w'_{1,1} > w_{1,2}) : Token_{1,2} <= Token_{1,1}$; $otherwise : Token_{1,2} <= 010$;
block 2:
$w'_{1,3} <= max(w'_{1,2}; w_{1,3})$;
$if(w'_{1,2} > w_{1,3}) : Token_{1,3} <= Token_{1,2}$; $otherwise : Token_{1,3} <= 011$;
block 3:
$w'_{1,4} <= max(w'_{1,3}; w_{1,4})$;
$if(w'_{1,3} > w_{1,4}) : Token_{1,4} <= Token_{1,3}$; $otherwise : Token_{1,4} <= 100$;
block 4:
$w'_{1,5} <= max(w'_{1,4}; w_{1,5})$;
$if(w'_{1,4} > w_{1,5}) : Token_{1,5} <= Token_{1,4}$; $otherwise : Token_{1,5} <= 101$;
block 5:
$w'_{1,6} <= max(w'_{1,5}; w_{1,6})$;
$if(w'_{1,5} > w_{1,6}) : Token_{1,6} <= Token_{1,5}$; $otherwise : Token_{1,6} <= 110$;
block 6:
$if(w'_{1,6} > w_{1,7}) : Token1 <= Token_{1,6}$; $otherwise : Token1 <= 111$;
***Token2* Computation : Wavelet Resolution Level - 2** (Fig. B.5(a))
block 0:
$w'_{2,1} <= max(w_{2,0}; w_{2,1})$;
$if(w_{2,0} > w_{2,1}) : Token_{2,1} <= 00$; $otherwise : Token_{2,1} <= 01$;
block 1:
$w'_{2,2} <= max(w'_{2,1}; w_{2,2})$;
$if(w'_{2,1} > w_{2,2}) : Token_{2,2} <= Token_{2,1}$; $otherwise : Token_{2,2} <= 10$;
block 2:
$if(w'_{2,2} > w_{2,3}) : Token2 <= Token_{2,2}$; $otherwise : Token2 <= 11$;
***Token3* Computation : Wavelet Resolution Level - 3** (Fig. B.5(b))
block 0:
$if(w_{3,0} > w_{3,1}) : Token3 <= 0$; $otherwise : Token3 <= 1$;

denotes the third wavelet coefficient in the first resolution level. Similarly *Token2* and *Token3* are assigned two bits and one bit respectively. The token assignment corresponding to each wavelet per resolution level is shown in Table B.1. As at the highest level (here it is fourth level) the signal gets contaminated with low frequency noise [69], the wavelet output at this level is zeroed out and thus no term such as *Token4* appears in Table B.1. The internal generic architectures for *Token1*, *Token2* and *Token3* computation within the *Input Unit* of the *Decision Making Block* are shown in Fig. B.4 and Fig. B.5(a) and (b) respectively and the associated computations are given in Table B.2 in pseudo-code format.

As shown in Fig. B.4 and Table B.2, *Token1* computation units comprise of seven design blocks (Block0 - Block6). In the first block (shown as Block-0 in Fig. B.4),

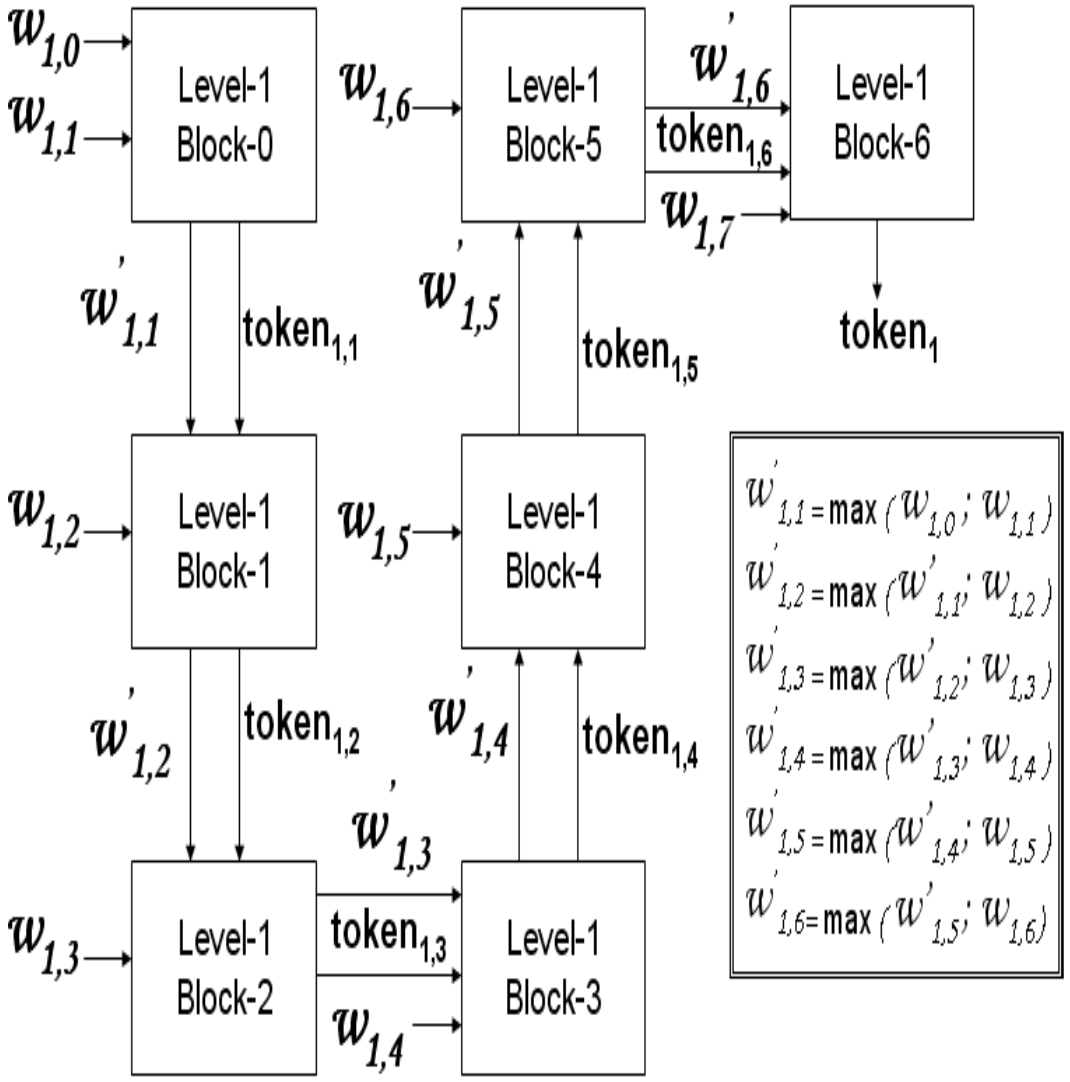


FIGURE B.4: Token 1 Generation using wavelets of level 1 of the analysis bank.

$w'_{1,1} = max(w_{1,0}; w_{1,1})$ are computed (see Table B.2). If level-1 wavelets in the analysis bank $w_{1,0} > w_{1,1}$, then *Token1* is assigned with 000, otherwise with 001. Following the same method intermediate tokens are computed within rest of the design blocks using these Level-1 wavelets and then finally in *Block-6*, if $w'_{1,6} > w_{1,7}$, then *Token1* is assigned with the three-digit value that $Token_{1,6}$ holds, otherwise *Token1* is assigned with 111. In the same way, as shown in Table B.2, *Token2* and *Token3* are computed using Level-2 and Level-3 wavelets. The tokens corresponding to each maternal thoracic channel are passed onto the *Range Calculate* block and the tokens associated with the abdominal channel are fed to the *Range Decide* block.

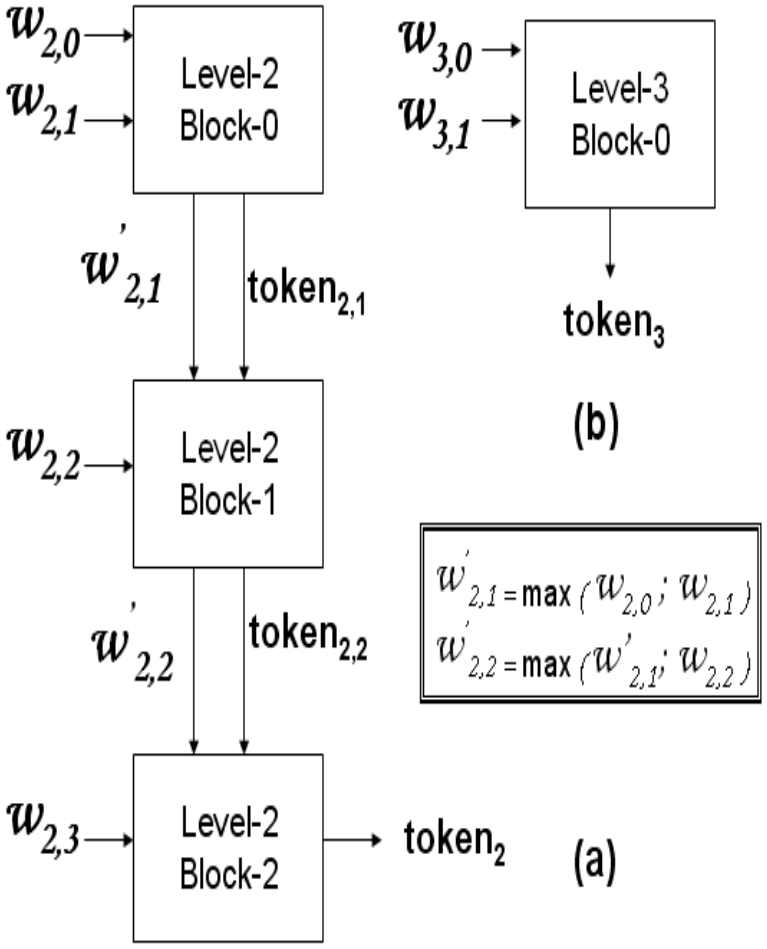


FIGURE B.5: (a) Token 2 Generation using wavelets of level 2 of the analysis bank, (b) Token 3 Generation using wavelets of level 3 of the analysis bank.

B.2.2 Range Calculate Block: Token Min-Max Computation

Amongst the received tokens from all thoracic channels, this block computes the maximum and minimum tokens for each resolution level. The internal architecture of this design block is shown in Fig. B.6. In this block *Token1 m1* and *Token1 m2* (*m1* and *m2* denote maternal thoracic channel 1 and channel 2) are compared within a sub-block (denoted by "Min-Max Comp" in Fig. B.6) which outputs the corresponding maxima and minima. In the same way *Token1 m3* and *Token1 m4* are compared to get their corresponding minima and maxima. Then in the next sub-blocks (denoted by "Token1 Max", "Token1 Min" etc. in Fig. B.6) these two minima/ maxima values are compared to get the ultimate minima (*Token1_min*)/ maxima (*Token1_max*). These values correspond to the modulus maxima range. In the same way *Token2_max*, *Token2_min*, *Token3_max* and *Token3_min* are calculated inside this block. These tokens are passed on to the *Range Decide* block. The computations involved within this block is provided in Table B.3 in the form of pseudo-code.

TABLE B.3: Computations within the "Range Calculate" block in Fig. B.6

Maternal Channel 1 and 2 (denoted by *m1* and *m2*)
Token1
$Token1\_max\_m1m2 <= max(Token1\_m1; Token1\_m2);$
$Token1\_min\_m1m2 <= min(Token1\_m1; Token1\_m2);$
Token2
$Token2\_max\_m1m2 <= max(Token2\_m1; Token2\_m2);$
$Token2\_min\_m1m2 <= min(Token2\_m1; Token2\_m2);$
Token3
$Token3\_max\_m1m2 <= max(Token3\_m1; Token3\_m2);$
$Token3\_min\_m1m2 <= min(Token3\_m1; Token3\_m2);$
Maternal Channel 3 and 4 (denoted by *m3* and *m4*)
Token1
$Token1\_max\_m3m4 <= max(Token1\_m3; Token1\_m4);$
$Token1\_min\_m3m4 <= min(Token1\_m3; Token1\_m4);$
Token2
$Token2\_max\_m3m4 <= max(Token2\_m3; Token2\_m4);$
$Token2\_min\_m3m4 <= min(Token2\_m3; Token2\_m4);$
Token3
$Token3\_max\_m3m4 <= max(Token3\_m3; Token3\_m4);$
$Token3\_min\_m3m4 <= min(Token3\_m3; Token3\_m4);$
Final *Token* Min-Max output from the Range Calculate block
$Token1\_max <= max(Token1\_max\_m1m2; Token1\_max\_m3m4);$
$Token1\_min <= min(Token1\_min\_m1m2; Token1\_min\_m3m4);$
$Token2\_max <= max(Token2\_max\_m1m2; Token2\_max\_m3m4);$
$Token2\_min <= min(Token2\_min\_m1m2; Token2\_min\_m3m4);$
$Token3\_max <= max(Token3\_max\_m1m2; Token3\_max\_m3m4);$
$Token3\_min <= min(Token3\_min\_m1m2; Token3\_min\_m3m4);$

TABLE B.4: Computations within the "Range Decide" block in Fig. B.3

Token1:
$Token1\_ab\_out <= Token1\_ab;$
$if(Token1\_min \leq Token1\_ab \leq Token1\_max)$:
$decision\_1 <= 1;$ // modulus maxima belongs to maternal signal
otherwise :
$decision\_1 <= 0;$ // modulus maxima belongs to fetal signal
Token2:
$Token2\_ab\_out <= Token2\_ab;$
$if(Token2\_min \leq Token2\_ab \leq Token2\_max)$:
$decision\_2 <= 1;$ // modulus maxima belongs to maternal signal
otherwise :
$decision\_2 <= 0;$ // modulus maxima belongs to fetal signal
Token3:
$Token3\_ab\_out <= Token3\_ab;$
$if(Token3\_min \leq Token3\_ab \leq Token3\_max)$:
$decision\_3 <= 1;$ // modulus maxima belongs to maternal signal
otherwise :
$decision\_3 <= 0;$ // modulus maxima belongs to fetal signal

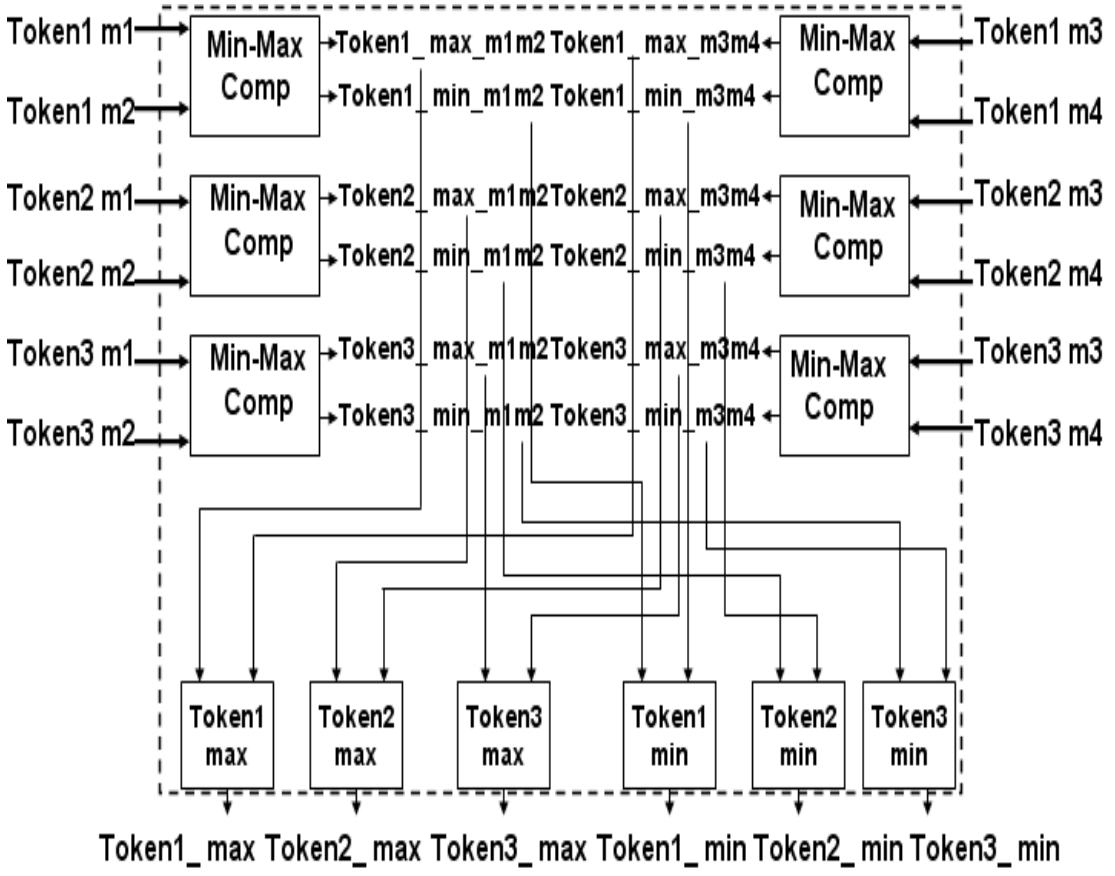


FIGURE B.6: Internal architecture of "Range Calculate" block.

B.2.3 Range Decide Block: Token Comparison

Range Decide unit receives the maximum and minimum thoracic tokens from the *Range Calculate* block as well as the abdominal tokens from the *Input Unit*. The abdominal tokens are compared with the maternal tokens maxima and minima. Then it decides either in favour of MECG if the abdominal token (*Token* ab*) lies within the range (between the minimum and maximum value of the thoracic tokens) otherwise it decides in favour of FECG and sets the active low "*Decision_**" signal to zero, where "*" indicates the resolution levels ($= 1, 2, 3$). As for example, if $Token1\_min \leq Token1\_ab \leq Token1\_max$, the corresponding modulus maxima in the first wavelet resolution level is adjudged to be a part of MECG and thus the respective $Decision\_1$ is assigned to one, but if this condition was not met, then it would have been decided in favour of FECG. The complete pseudo-code of the computations involved within this block is provided in Table B.4.

B.2.4 Update Block: MECG Removal

Update Block receives the tokens for different resolution levels and checks the respective "*Decision_**" signals. If it is in favour of MECG, then it will simply fill up the corresponding wavelet output register with zeroes (denoted by $w_{*,*}^{(ab,out)}$ in Table B.5), otherwise it will not update the registers and simply assign the incoming abdominal wavelets (denoted by $w_{*,*}^{(ab,in)}$ in Table B.5) from the Analysis Bank to the output registers. When the update phase is over, it starts sending these processed wavelets serially to the next block with "*maxima over*" via its Output Unit (see Fig. B.3). Detailed pseudo-code is provided in Table B.5.

TABLE B.5: Computations within the "Update Block" in Fig. B.3

Abdominal Wavelets (denoted by "ab"): Resolution Level-1
$if decision\_1 = 1$: // maternal modulus maxima detected case $Token1\_ab\_out$: // zero-out the corresponding wavelets $000: w_{1,0}^{(ab,out)} <= 0;\ 001: w_{1,1}^{(ab,out)} <= 0;$ $010: w_{1,2}^{(ab,out)} <= 0;\ 011: w_{1,3}^{(ab,out)} <= 0;$ $100: w_{1,4}^{(ab,out)} <= 0;\ 101: w_{1,5}^{(ab,out)} <= 0;$ $110: w_{1,6}^{(ab,out)} <= 0;\ 111: w_{1,7}^{(ab,out)} <= 0;$ $if decision\_1 = 0$: // modulus maxima belongs to fetal signal // Input Abdominal wavelets belong to fetal signal $w_{1,0}^{(ab,out)} <= w_{1,0}^{(ab,in)};\ w_{1,1}^{(ab,out)} <= w_{1,1}^{(ab,in)};$ $w_{1,2}^{(ab,out)} <= w_{1,2}^{(ab,in)};\ w_{1,3}^{(ab,out)} <= w_{1,3}^{(ab,in)};$ $w_{1,4}^{(ab,out)} <= w_{1,4}^{(ab,in)};\ w_{1,5}^{(ab,out)} <= w_{1,5}^{(ab,in)};$ $w_{1,6}^{(ab,out)} <= w_{1,6}^{(ab,in)};\ w_{1,7}^{(ab,out)} <= w_{1,7}^{(ab,in)};$
Abdominal Wavelets (denoted by "ab"): Resolution Level-2
$if decision\_2 = 1$: // maternal modulus maxima detected case $Token2\_ab\_out$: // zero-out the corresponding wavelets $00: w_{2,0}^{(ab,out)} <= 0;\ 01: w_{2,1}^{(ab,out)} <= 0;$ $10: w_{2,2}^{(ab,out)} <= 0;\ 11: w_{2,3}^{(ab,out)} <= 0;$ $if decision\_2 = 0$: // modulus maxima belongs to fetal signal // Input Abdominal wavelets belong to fetal signal $w_{2,0}^{(ab,out)} <= w_{2,0}^{(ab,in)};\ w_{2,1}^{(ab,out)} <= w_{2,1}^{(ab,in)};$ $w_{2,2}^{(ab,out)} <= w_{2,2}^{(ab,in)};\ w_{2,3}^{(ab,out)} <= w_{2,3}^{(ab,in)};$
Abdominal Wavelets (denoted by "ab"): Resolution Level-3
$if decision\_3 = 1$: // maternal modulus maxima detected case $Token3\_ab\_out$: // zero-out the corresponding wavelets $0: w_{3,0}^{(ab,out)} <= 0;\ 1: w_{3,1}^{(ab,out)} <= 0;$ $if decision\_3 = 0$: // modulus maxima belongs to fetal signal // Input Abdominal wavelets belong to fetal signal $w_{3,0}^{(ab,out)} <= w_{3,0}^{(ab,in)};\ w_{3,1}^{(ab,out)} <= w_{3,1}^{(ab,in)};$

B.3 Proposed Low Complexity VLSI Architecture for FECG Extraction

We have proposed a memory reduction methodology of classical DWT/IDWT architecture in Chapter 2 and it has been reported in [86]. This DWT/IDWT is one of the fundamental building blocks of the FECG architecture to be introduced now.

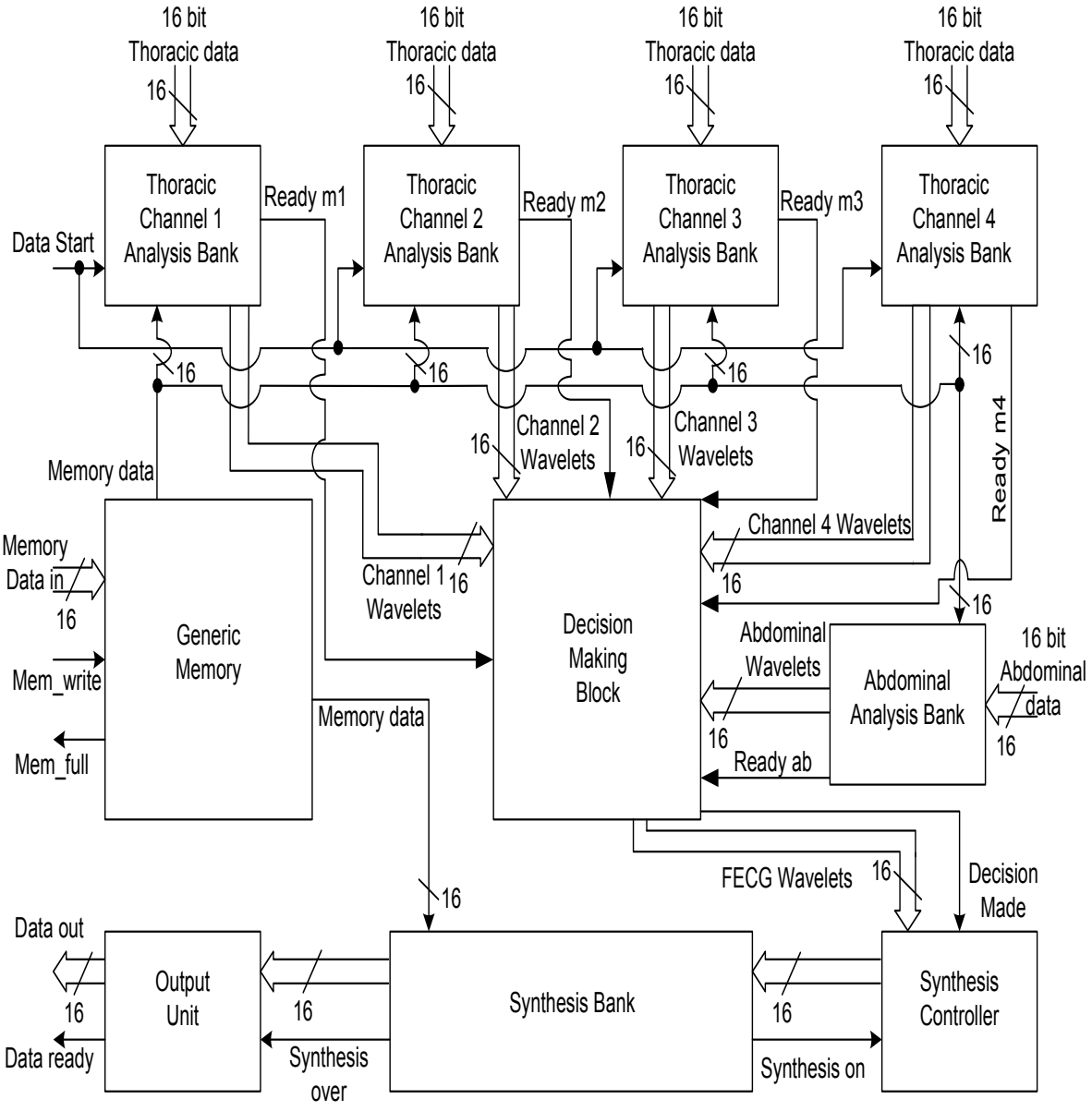


FIGURE B.7: Proposed VLSI architecture for FECG extraction.

The architecture is designed assuming each incoming frame consists of 16 ECG samples each having 16 bits word-length because normally the number of bits per ECG sample varies from 12 to 36 [19]. The filter length is assumed to be 16 and each coefficient consists of 16 bits. Fig. B.7 shows the architecture of the complete design. As number of lead used of FECG extraction may vary [69], we have considered 5-channel input for our architecture where four channels get thoracic ECG and one is used to receive

composite data. The main design units are Analysis Bank, Decision Making Block, Generic Memory and Synthesis Bank along with Controller and Output unit. The Analysis and Synthesis Bank are used for DWT and IDWT respectively and the reduced complexity DWT/IDWT technique discussed in Chapter-2 is used to design these blocks. This model is generic so that it can be configured with any kind of orthonormal wavelets. The concept of complex fuzzification [69] is substituted by using simple Decision Making Block which has already explained in detail in Section B.2.

As outlined in the last subsection, in the memory unit of this architecture only the non-repetitive combinations of filter coefficients are stored from the set of data. The strategy for address generation has been explained in detail in [86]. Since this memory is a generic one, it can be configured with any type of orthonormal mother wavelet depending upon the morphology of the signal and thereby gives users freedom to choose their own wavelet functions.

There are five Analysis Banks corresponding to five input channels in the overall architecture (Fig. B.7). Each of these consists of three units - Input Unit, Analysis Block and Analysis Controller. Analysis Block is further divided into Input Bank, Processing Unit and Output Bank. The Analysis and Synthesis banks are designed following the memory reduction methodology discussed in the last subsection. DA based filter design technique has been applied to realize the FIR filters of these blocks which replaces multiplication operations by additions only and thereby making the entire architecture multiplierless. In addition; there are two control units responsible for controlling the analysis and synthesis banks. Two active-low single bit signals enable the Analysis Bank and Synthesis Bank respectively and these signals may be enabled by the user or the previous processing unit in case of a complete system. When the output of these banks are ready, the respective controllers produce two active-low single bit signals - “analysis over” and “synthesis over” which indicates the completion of the process.

B.4 Discussion and Concluding Remarks

A simplified logic design methodology for the complex fuzzification algorithm for detecting the maternal modulus maxima locations from the abdominal composite has been proposed and a memory-efficient, multiplierless architecture for FECG Extraction based on a recently proposed reduced complexity DWT/ IDWT method has been presented. The proposed simplified logic design methodology has been explained in detail with the help of pseudo-codes and it can be seen that the implementation of this unit needs only simple comparison logic. The proposed FECG extraction architecture uses the reduced complexity DWT/IDWT method proposed recently in [86] and thus it requires significantly less memory than the conventional methods. This memory model is generic which is suitable for any kind of orthonormal wavelet implementation. For that

one only needs to pre-compute the parametric values of the combination of the filter coefficients depending upon selection of mother wavelets for different applications and load them into the memory before the processing begins. Moreover the overall architecture is completely multiplierless.

The proposed architecture has been coded in VHDL, synthesized using Synopsys Design Compiler with 0.13μm standard technology for a target frequency 1 MHz and power has been obtained using Synopsys Prime Time. Total synthesized cell area of the whole design is found out to be 14.2mm^2 and power consumption is 101.5μW at 1.2 V @ 1 MHz frequency. Our current work involves the gate level implementation of the proposed VLSI architecture and its characterization in terms of power consumption and silicon area overhead. It is hoped that the proposed architecture makes a useful contribution to the emerging area of pervasive healthcare especially in personalized health monitoring in a non-clinical home environment where light-weight, yet effective low-power signal processing is necessary.

Appendix C

Automated and Robust Channel Identification Scheme: Solving Permutation Indeterminacy of ICA for Artifacts Removal from ECG

Electrocardiography (ECG) has been used very effectively for decades to measure the physiological well being of human heart. A typical ECG measurement is generally contaminated with power-line interference, abrupt potential change due to loose electrode contact, motion artifacts, several non-cardiac biomedical artifacts, effect of respiration and also a combination of more than one of these sources [155]. This necessitates the use of some post processing techniques which can remove the hidden effects of these mixed artifacts and noise from the captured ECG signals.[1]

At this point, ICA has drawn tremendous attention in research community for last few years as an effective way to decompose a set of mixed recorded biomedical signals into separate sets of constituent independent latent variables [117]. This poses ICA as an appropriate post-processing tool for the removal of the unwanted contaminations from the recorded signals [156]-[162]. But ICA has an inherent problem called permutation indeterminacy which means loosing the order information of the separated signals [163]. Thus although an ECG can be separated from its artifacts, physicians depend on the

[1]The contents of this Appendix have partly appeared as "Automated and Robust Channel Identification Algorithm and Architecture to Solve Permutation Problem of ICA for Artifacts Removal from ECG in Remote Health Monitoring Environment" by Acharyya *et. al.* in *Proceedings of the UK Electronics Forum - 2010* and as "Robust Channel Identification Scheme: Solving Permutation Indeterminacy of ICA for Artifacts Removal from ECG", *32nd Annual IEEE EMBS Conference*. Please see Appendix-A for further detail.

visual inspection to decide which one is ECG and which one is an artifact [6]. In a typical hospital-based approach this permutation indeterminacy does not pose much problem since the separated signals are displayed on the screen and the physician can immediately identify which is the wanted ECG signal amongst the artifacts.

However for remote healthcare application the situation is different where the ECG characteristics need to be identified by an intelligent system in an automated fashion and not by human-being in the first instance. Thus, after post-processing the contaminated ECG signal using ICA, it is extremely important to find out the order of the signal i.e. clearly identify the channel containing the wanted ECG signal and the rest of the channels containing the unwanted artifacts in a distinct way without any human intervention. Without loss of generality, the same problem can be translated for other important biomedical signals like Electroencephalogram (EEG) and Electromyogram (EMG) as well.

In this context, to the best of our knowledge, only one research attempt has been reported so far [6] (see Section C.1 for detail) to analyze the fore-mentioned problem with an innovative algorithm which has the capability to identify the signal of interest out of all other contaminations after ICA post-processing rather than depending only on the visual observations. This motivated us to envisage a remote health monitoring system comprising of ICA as the post-processing unit followed by an automated channel identification module which can effectively solve the permutation indeterminacy in this context and outputs only the signal which is of interest to the physicians. This appendix presents our initial attempt to make this vision a reality.

The existing approach, as discussed in [6], is based on linear statistics to solve the permutation problem. Although it is the first of its kind, it considers abrupt change in signal potential as the only artifact present in the recorded ECG signals. This definition of artifact is not unique because several literatures [155]-[162] describe other artifacts which may also have severe effects on the recorded ECG signals and are quite capable of invalidating all the measurements. Therefore it is important to devise a robust algorithm which does not depend on the definition of any specific artifact.

In this appendix, we propose one such channel identification algorithm which can detect the signal of interest out of many widely known artifacts. The proposed algorithm has also been validated with nine practical case studies.

The rest of this appendix is organized as follows. Section C.1 discusses the cause of permutation ambiguity and describes the existing method. Section C.2 describes the proposed algorithm, Section C.3 provides the architecture of the proposed algorithm along with a performance analysis in terms of hardware complexity and computational delay, Section C.4 presents the experimental results and validates the robustness of the proposed algorithm through comparison with the existing approach and Section C.5 concludes this appendix.

C.1 Background and Related Work

C.1.1 ICA and Permutation Indeterminacy

Considering number of independent sources (n) is equal to the number of sensors, mixed signal set ($\mathbf{X}$) can be defined as [163]: $\mathbf{X} = \mathbf{AS}$, where $\mathbf{S} \in \mathbb{R}^n$ is the independent sources and $\mathbf{A} \in \mathbb{R}^{n \times n}$ represents the unknown mixing matrix. ICA retrieves $\mathbf{S}$ from $\mathbf{X}$ by formulating an unmixing matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$. It is evident that with no *apriori* knowledge of $\mathbf{S}$ and its ordering, it is impossible to estimate $\mathbf{A}$ accurately, therefore $\mathbf{B}$ is an approximate measure of $\mathbf{A}$ up to a permutation and scaling as given by [163]: $\mathbf{G} = \mathbf{BA} = \mathbf{PD}$, where $\mathbf{G}$ is the global matrix and $\mathbf{D}$ is the full rank diagonal matrix and $\mathbf{P}$ is the permutation matrix. The presence of the effect of $\mathbf{P}$ in the estimated components causes permutation indeterminacy.

C.1.2 Existing Approach

He, Clifford and Tarassenko proposed a method in [6] to tackle this ambiguity problem in the case of artifact removal from ECG using 3-D ICA. The fundamental considerations are either of the three channels comprises of noise or artifact or ECG. Type of artifact considered is transient i.e. the abrupt change in potential. The method comprises of following two stages.

Firstly, *Kurtosis* is computed on ICA processed data of three channels using [6]: $Kurt(x) = \mathcal{E}(x^4) - 3[\mathcal{E}(x^2)]^2$, where x is any signal in generalized sense and $\mathcal{E}[\cdot]$ represents statistical expectation operation. The modulus of *Kurtosis* is then compared with a precomputed *Threshold* value. Since continuous noise is more gaussian than ECG or artifacts, the computed modulus for this is less than the rest and thus such comparison removes the noise from the measurements.

Second stage is to identify the channel consisting of ECG and remove the artifact. This is done by deriving a metric called *Variance Index*. A complete frame within a channel is divided further into N sub-frames and variances are computed for each of these sub-frames following [6]: $Var_x = \sum_{n=1}^{N} [x(n) - \mathcal{E}[x(n)]]^2$. The difference of variances are computed for consecutive sub-frames within a same channel. The same procedure is repeated for other channels as well. Therefore, like kurtosis, these differences are compared with a precomputed threshold. This difference will become very high and lie above the threshold value for the sub-frame containing the transient. Thus this process removes the channel contaminated with artifact and identifies the channel containing ECG.

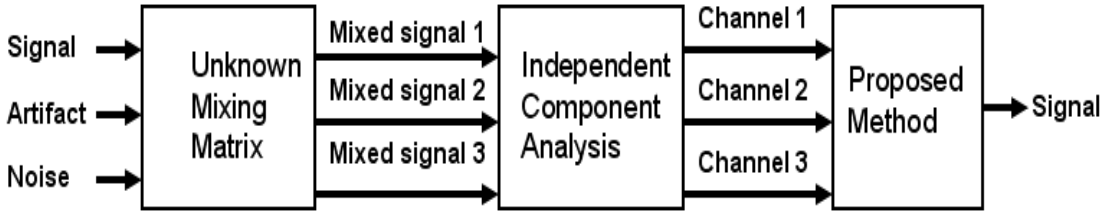


FIGURE C.1: Envisaged system with the proposed channel identification algorithm.

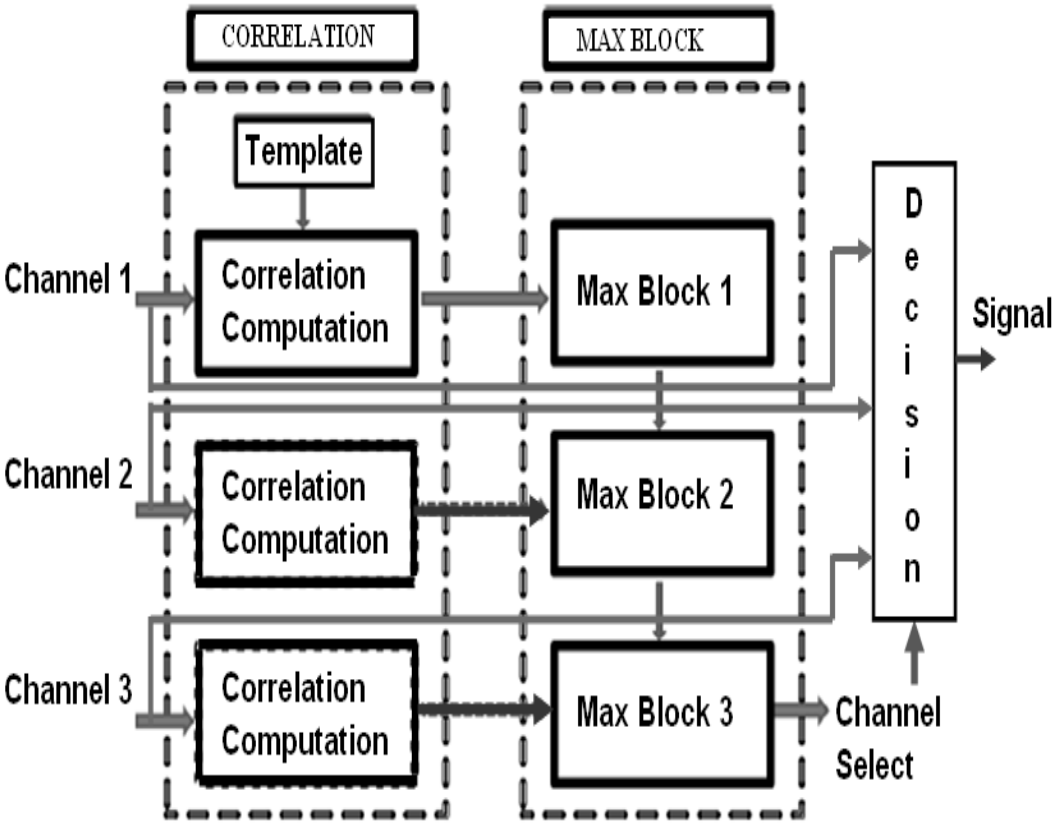


FIGURE C.2: Proposed architecture.

C.2 Proposed Algorithm

Fig. C.1 shows the overview of the whole system where it can be seen that the signal, artifact and noise are mixed by an unknown mixing matrix, mixed signals are obtained at the sensors and these are post-processed using ICA to produce three channel outputs. As mentioned in previous sections, these channels suffer from permutation ambiguity.

Therefore a channel identification algorithm is necessary which can identify the channel containing only the signal of interest out of these three as shown in Fig. C.1. Since [6], as mentioned in the beginning, is specific to one category of artifact, therefore it is necessary to formulate an algorithm which is independent of the definition of any artifact and thereby making it more robust.

Here, we propose one such algorithm which can address this problem successfully and thereby solving permutation indeterminacy in this context. This proposed algorithm is based on the classical *template matching* approach which is very well known concept and has been used for years in pattern recognition techniques. But, to the best of our knowledge, its potential has not been utilised in solving permutation indeterminacy from biomedical signal processing perspective. Since the physicians already know which particular biomedical signal they are interested to look at, therefore from practical point of view, firstly a template for that particular signal corresponding to any specific patient can easily be framed from the initial measurements. Subsequently based only on this practical assumption, each frame within the incoming channels can be cross correlated with the template which will yield a number of correlation coefficients. Next step is to find the *maximum* among all these computed correlation coefficients for all three channels.

The maximum among these three computed maximum correlation coefficients points to the channel containing the signal of interest. In this appendix ECG is considered as the signal of interest and nine case studies have been performed based on different nature of noise and artifacts as obtained from several published literatures [155]-[162]. These are explained in detail in the next section. The proposed algorithm is described in pseudo-code format as follows:

1. Choose c, the number of input channels ($1 \leq c \leq 3$) and initially set $c = 1$.
2. Choose time shift as d ($0 \leq d \leq K-1$) where K is the frame-length of each channel.
3. Choose sample number as i where $0 \leq i \leq K-1$.
4. If $(i-d) < 0$ or $(i-d) \geq K$, go to *Step* 6 else continue.
5. Compute cross correlation coefficient $r(d)$ as follows:

$$r(d) = \frac{\sum (X_i - \mathcal{E}[X])(Y_{i-d} - \mathcal{E}[Y])}{\sqrt{\sum (X_i - \mathcal{E}[X])^2}\sqrt{\sum (Y_{i-d} - \mathcal{E}[Y])^2}} \tag{C.1}$$

   where, X, Y and $\mathcal{E}[\cdot]$ denote the selected channel and template and mean respectively.
6. Assign $i \leftarrow i+1$. If $i \leq K-1$ go to *Step* 3, else continue.
7. Store correlation coefficients in $r_c(d)$ as: $r_c(d) = r(d)$.
8. Assign $d \leftarrow d+1$. If $d \leq K-1$, go to *Step* 4, else continue.
9. Store the maximum value of r_c in M_{r_c} as: $M_{r_c} = Max(r_c)$.
10. Assign $c \leftarrow c+1$. If $c \leq 3$, go to *Step* 1 else continue to next step.

11. If $M_{r_1} > (M_{r_2}$ and $M_{r_3})$, decide in favor of Channel 1 and end, else continue.
12. If $M_{r_2} > (M_{r_1}$ and $M_{r_3})$, decide in favor of Channel 2 and end, else continue.
13. If $M_{r_3} > (M_{r_1}$ and $M_{r_2})$, decide in favor of Channel 3 and end.

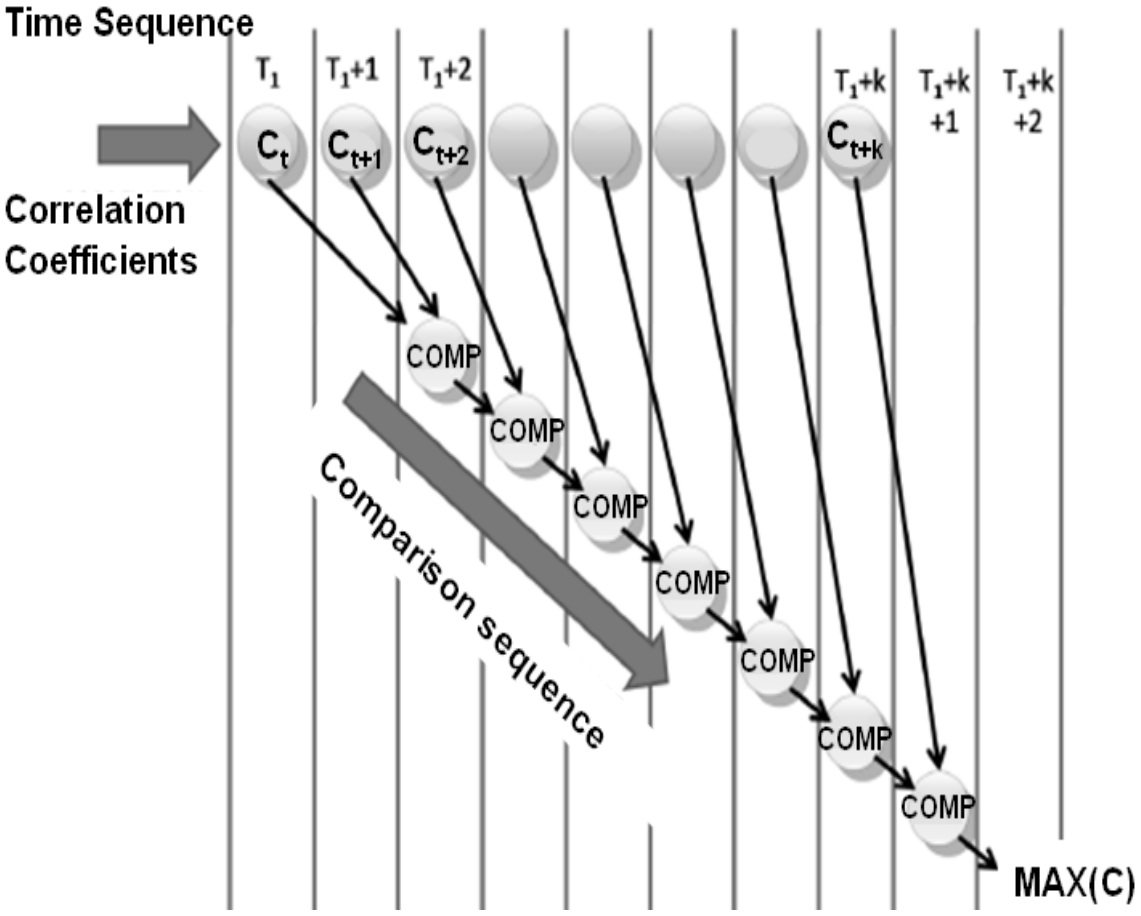


FIGURE C.3: Sequential search in each max block of the proposed architecture.

C.3 Architecture and Performance Analysis

C.3.1 Proposed Architecture

Fig. C.2 provides the architecture corresponding to the proposed algorithm as discussed in Section C.2. It consists of two main blocks - *Correlation* and *Max* block. Correlation block consists of three correlation computation modules which compute the coefficients with the precomputed template corresponding to each channel (see *step* 1-10 in the pseudo-code in Section C.2). Max block consists of three *maximum* coefficient computation modules which compute the maximum among all coefficients (see *step* 11-13 in the pseudo-code in Section C.2). As an example, output of "Max Block 1" is the maximum

correlation coefficient corresponding to channel 1. Point to be noted from Fig. C.2 that the output of Max Block 1 goes to Max Block 2 and thus the output of Max Block 2 contains the maximum coefficient between channel 1 and 2. In the same way Max Block 2 and 3 are connected to each other and the final output of Max Block 3 indicates the maximum coefficient among all three channels. Based on this decision, ultimately the channel containing the signal of interest is identified. Each module in the Max Block consists of simple comparator arrays. Since correlation module computes correlation coefficients sequentially, therefore it is preferable to follow sequential search procedure within these arrays.

Fig. C.3 shows graphically this technique along with the comparison sequence. Considering k-number of correlation coefficients $(C_t, ..C_{t+k})$, the latency of one max computation module is $(k+2)$ time unit.

In subsequent sub-sections we analyze the architectural performance of the proposed algorithm in terms of hardware complexity and delay. Although accurate estimation of these two parameters are difficult because of the basic differences of adopted technology, assumptions made for the derivations and their implementation dependencies; nevertheless our attempt here is to perform the approximate analysis. To compute overall hardware complexity and delay, two metrics - transistor count and delay of a basic two input NAND gate are used.

However, since implementation of different arithmetic unit can be done in different ways, first we derive the generic hardware complexity and delay expression based on the number of arithmetic operation considering a flat unfolded architecture without resource sharing (used for hardware complexity computation) or parallel computing capability (used for delay analysis) following the approach presented in [122] irrespective of any specific implementation. Then to give an insight into hardware, we consider only Ripple Carry Adder (RPA) and Subtractor (RPS), Conventional Array Multiplier (CAM), Non-restoring Array Divider (NAD) and Square Rooter (SQRT) structures as the means of implementing the arithmetic operations. Additionally it has been considered that the hardware complexity and delay of a squaring unit is 50% and 80% respectively of a conventional two-input multiplier [82].

C.3.2 Hardware Complexity

Considering frame-length K and word-length n, arithmetic operations involved in correlation block (see Fig. C.2) as given in (C.1) are as follows - $4K^2$ number of n-bit subtractions, $(2K^2 + K)$ number of $n \times n$-bit multiplications (considering $2K^2$ number of squaring units as well), K number of n by n-bit divisions and $2K$ number of square rooting. The max block (see Fig. C.2), as discussed in the last subsection, consists of comparators which have total $n(K-1)$ number of XOR and OR gates. Consider a n-bit

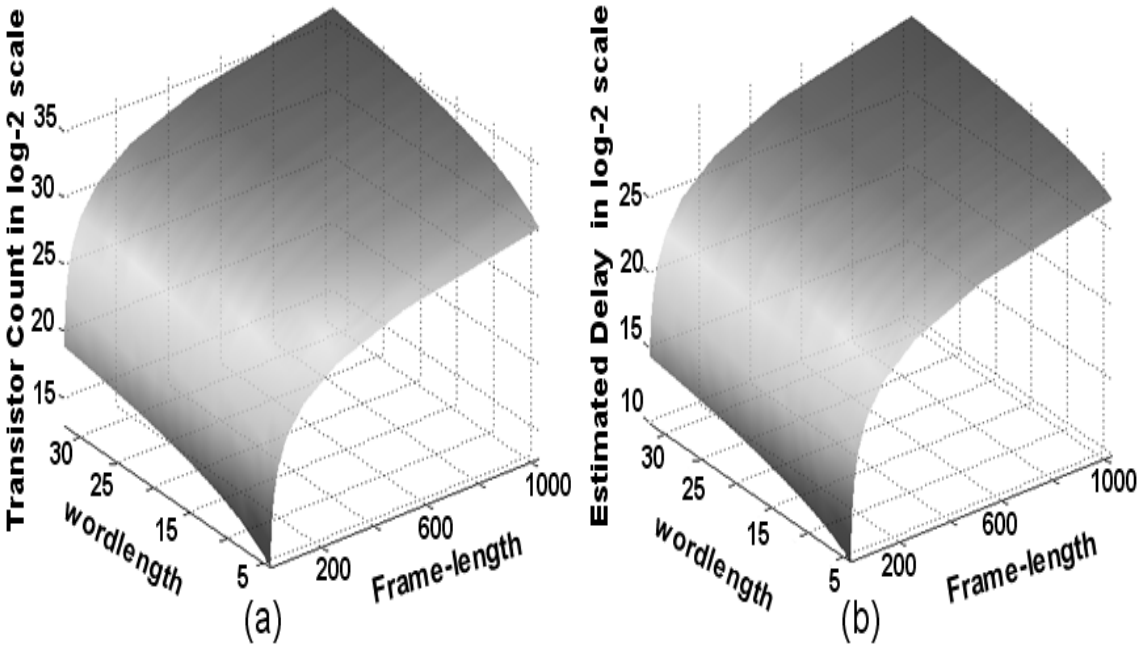


FIGURE C.4: (a) Variation of transistor count with framelength and wordlength, (b) variation of delay with framelength and wordlength.

RCA/RPS requires n Full Adders (FA) (in a simplified view) [122], $n \times n$ CAM requires $n(n-2)$ FA plus n Half Adders (HA) and n^2 AND gates [122], one n by n NAD consists of $0.5 \times n(3n-1)$ FA and $0.5 \times n(3n-1)$ XOR gates [122] and one n-bit SQRT needs $0.125 \times (n+6)n$ FA and XOR gates [138]. In addition, considering one FA cell requires 24 transistors, one HA cell and a two input XOR gates consists of 12 transistors and a two input AND and OR gate consist of 6 transistors [122], TC_A, TC_S, TC_M, TC_D and TC_{SQ} can be calculated where TC_* are the transistor counts for RCA, RPS, CAM, NAD and SQRT respectively following [122].

Using these TC_* in the fore-mentioned total number of operations, overall Transistor Count (TC) are computed as: $TC = 24K^2n + 93Kn^2 + 18Kn + 60K^2n^2 + 18nK - 18n$. Fig. C.4(a) plots the variation of TC with the frame-length and word-length where it is evident that TC increases with the increase in frame-length and word-length.

TABLE C.1: Cases considered for experiments

Channel	Cases Considered	Case No.
Signal	Noise free ECG	1
	Noisy ECG	7
Artifact	Respiration Artifact	2
	Low Power Artifact	3
	Non-cardiac Artifact (e.g. EEG)	4
	Muscle Movement	5
	High Power Transient	6
Noise	Low Power Random Noise	8
	High Power Random Noise	9

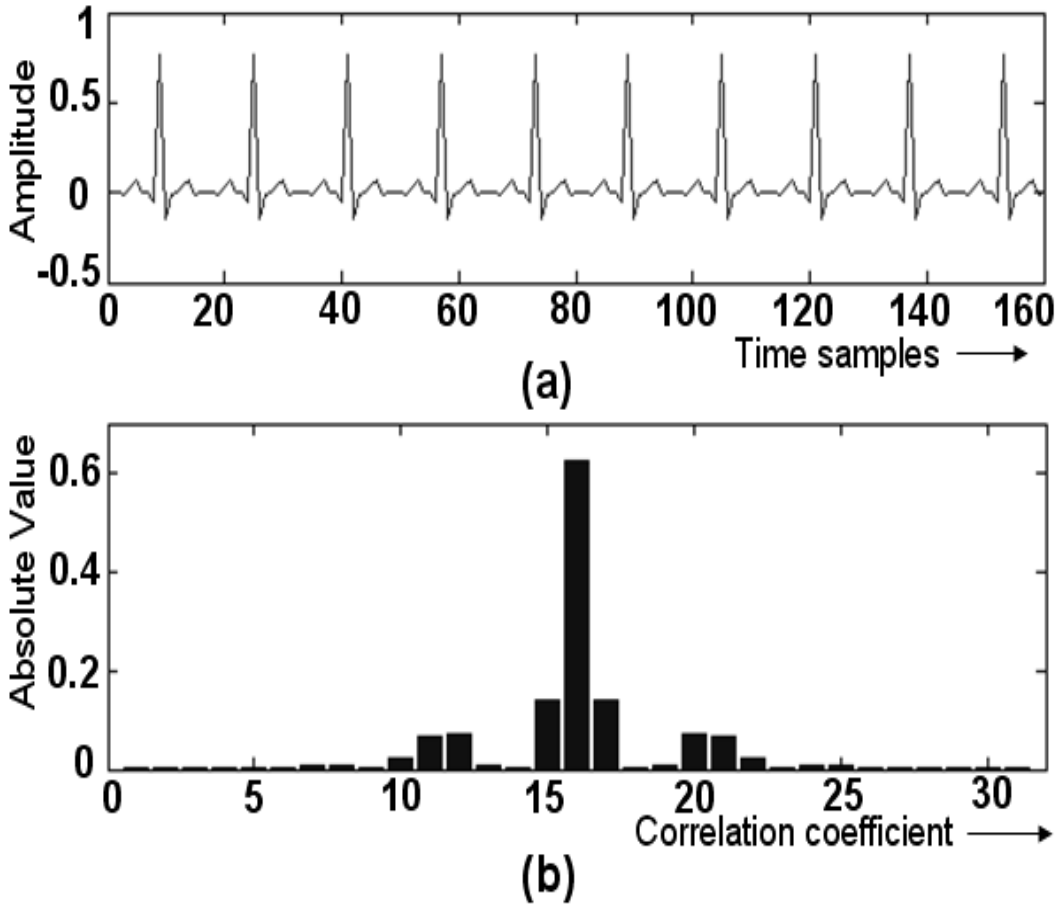


FIGURE C.5: (a) Case 1 - Noise free ECG, (b) correlation coefficients.

C.3.3 Delay Analysis

Denoting two input NAND gate delay by δ, the delay of n-bit RCA, RPS, CAM, NAD and SQRT can be given as $\tau_A = \tau_S = 2n\delta$, $\tau_M = 8n\delta$, $\tau_D = (3n+2)n\delta$ [122] and $\tau_{sq} = (n+1)(2n+3)\delta$ [138]. Delay of one two input XOR and OR gate can be represented as 3δ and 2δ respectively. Following the approach presented in [122], considering the delay of a squaring unit is 80% of a full multiplier [82] and combining τ_* along with the total arithmetic computations as derived in the last sub-section, overall delay (D) can be estimated as: $D = 28.8nK^2 + 7Kn^2 + 25Kn - 5n$. Fig. C.4(b) plots the variation of estimated delay with frame-length and word-length where it is evident that with the increase in word-length and frame-length, delay will also increase.

C.4 Experimental Results

Due to lack of ECG data recording facilities, we used the MIT-BIH online ECG database and also took help from [164] and simulated the ECG and artifact patterns in MATLAB.

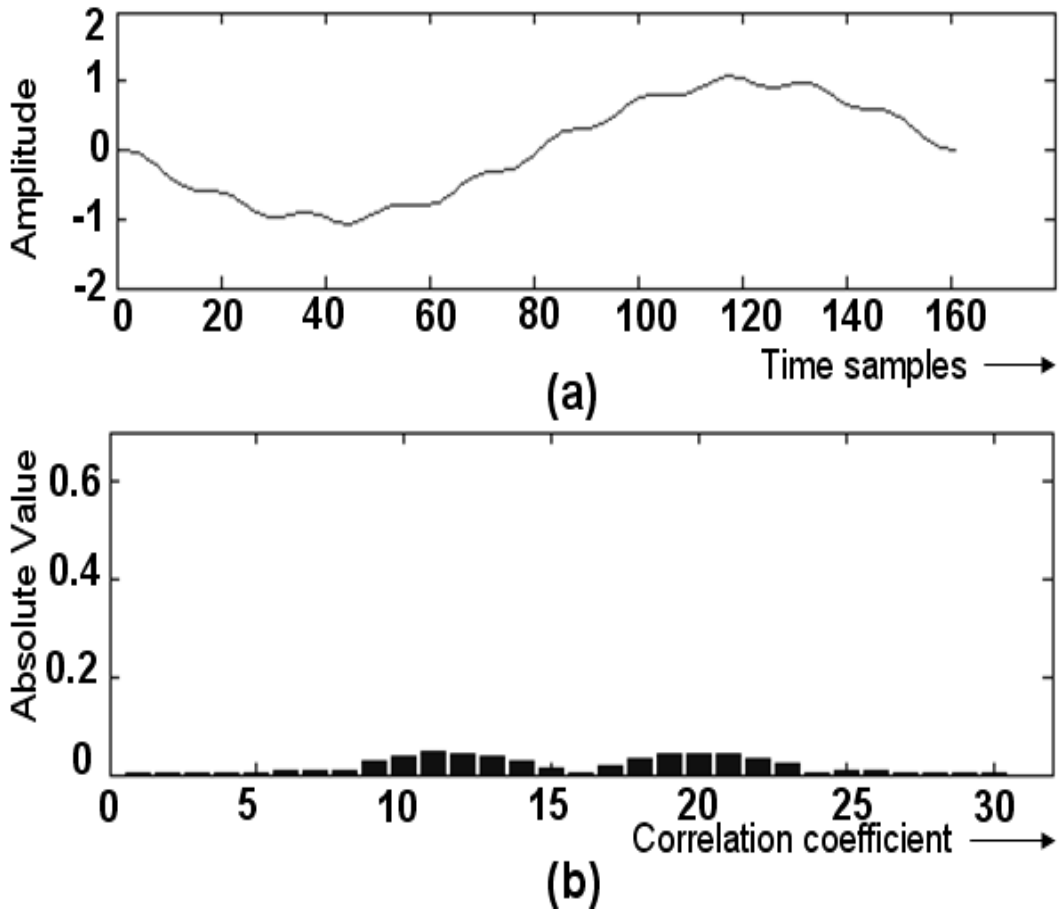


FIGURE C.6: (a) Case 2 - Respiration artifact, (b) correlation coefficients.

The proposed algorithm as well as the existing method discussed in [6] are also modeled in MATLAB. Due to better convergence speed and higher accuracy [75], FastICA has been chosen as the preferred one among different existing ICA algorithms for post-processing. The template of ECG waveform is generated by using Symlet Wavelet in MATLAB.

To validate the proposed algorithm we perform simulation based experiments considering total 9 cases in which two different types of ECG and noise are considered and 5 different types of artifacts have been chosen as used in literatures [155]-[162]. It is to be noted that among all the cases, only case 6 i.e. high power transient is considered in [6]. We computed the maximum of *correlation coefficients* for each case as detailed in Section C.2 according to the proposed algorithm. The first seven cases along with the distribution of the correlation coefficients are shown in Fig. C.5 - Fig. C.11. Low power and high power random noise sources are considered as case 8 and 9 as clatters occur from different haphazard sources inside our body. All these cases are shown in Table C.1.

To do the performance comparison, we computed also the *variance index* (see Section C.1) as mentioned in [6] for each case. Variance indices based on the existing approach

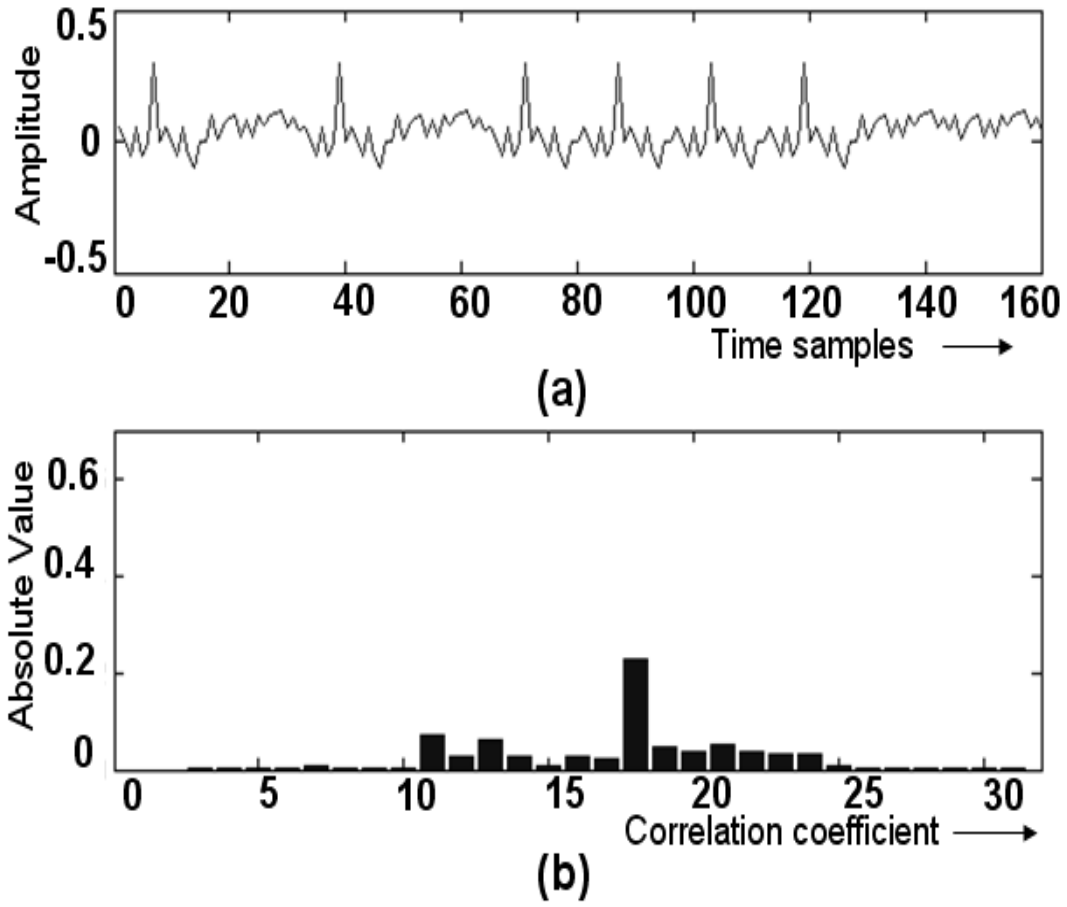


FIGURE C.7: (a) Case 3 - Low Power artifact, (b) correlation coefficients.

and *maximum correlation coefficient* values based on the proposed algorithm are shown in Table C.2 against each of the fore-mentioned cases. It is to be noted from Table C.2 that variance indices are not computed for case 8 and 9 because these cases correspond to different types of noise and [6] (see Section C.1) identifies noise by kurtosis, not by variance based method. It can be seen from Table C.2 that the variance index corresponding to case 6 is the highest among all other cases. Now considering [6] if a *threshold* is chosen just above the variance index corresponding to case 1 (i.e. 0.640054), high power transient type artifact (case 6) can easily be identified, whereas being the variance indices less than the selected threshold, other artifact types (case 2-5) can not be separated.

If the value of the selected threshold is lowered down, then there is a possibility that ECG signals (case 1 or 7) may wrongly be identified as artifact. *Therefore variance index can not be chosen as unique metric for channel identification and existing method [6] can not be generalized for all practical cases.*

Table C.2 also shows the values of *maximum* correlation coefficients according to the proposed algorithm and it can easily be seen that this value corresponding to case 1 is

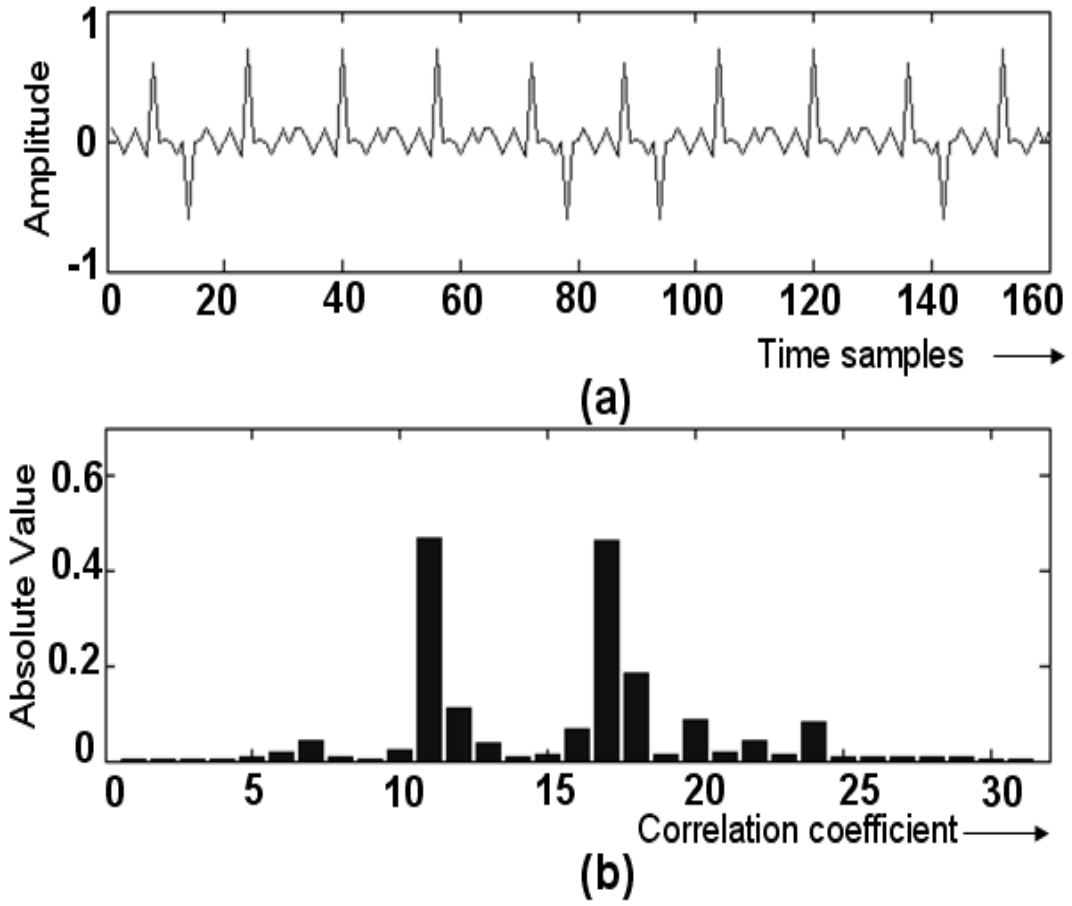


FIGURE C.8: (a) Case 4 - Non-cardiac artifact, (b) correlation coefficients.

TABLE C.2: Comparative results for different cases

Case No.	Existing Method (*Variance Index*)	Proposed method (*Max Correlation Coeff*)
1	0.640054	0.6237
2	0.116465	0.0443
3	0.072250	0.2328
4	0.000048	0.4664
5	0.081000	0.4035
6	6.560990	0.5422
7	0.506870	0.5787
8	N.A.	0.0792
9	N.A.	0.4530

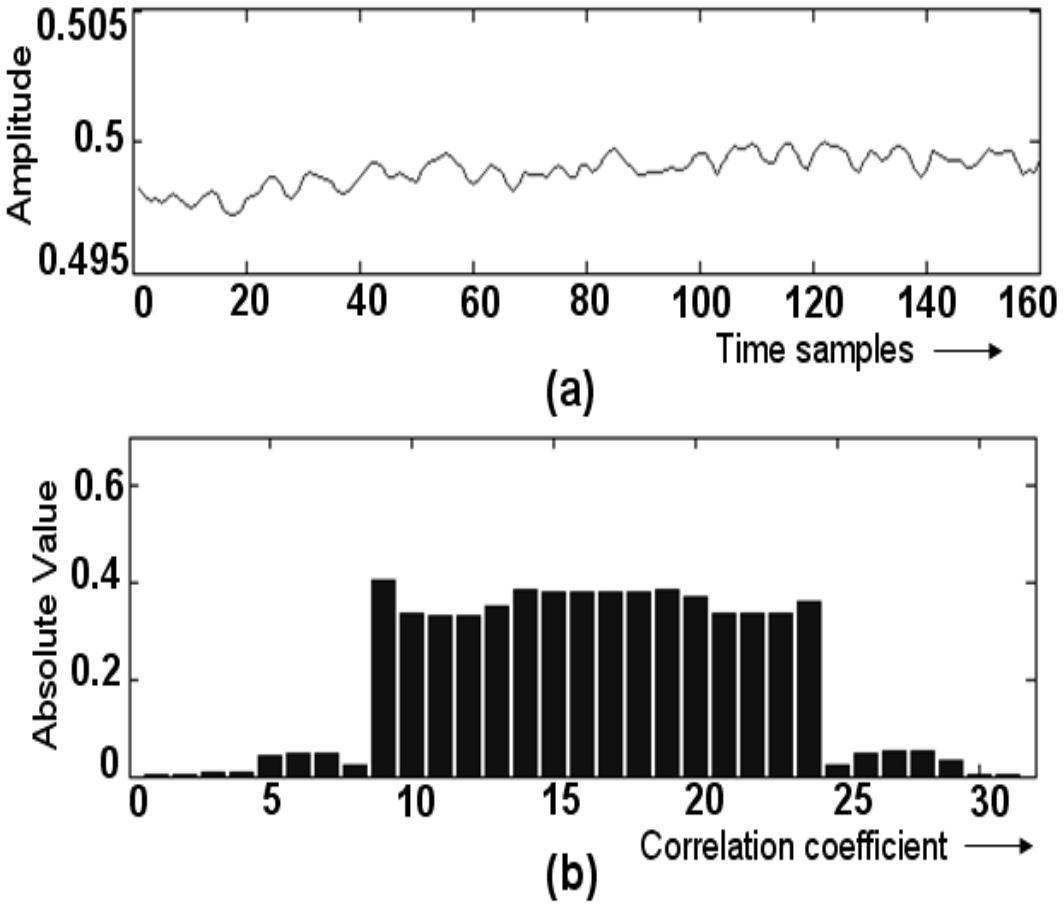


FIGURE C.9: (a) Case 5 - Muscle movement, (b) correlation coefficients.

the maximum among all other values and therefore this method can identify the channel containing the ECG signal. Even if the ECG signal is noisy (case 7), still its maximum correlation coefficient's value is the highest among all other artifacts as well as noise types. This comparative study justifies the robustness of the proposed algorithm over the existing approach. It can also be verified from Table C.2 that except for case 6, variance-indices computed for all other cases (case 2-5) based on existing method [6] are less than the precomputed threshold and therefore fails to detect the presence of other type of artifacts. On the other hand, as can be seen from Fig. C.5 and C.11 the proposed algorithm is able to detect the ECG signal (case 1 and 7) successfully since the maximum correlation coefficients corresponding to these two cases are higher than the rest.

C.5 Concluding Remarks

In this appendix a novel channel identification algorithm has been proposed which can successfully identify the channel containing the signal of interest from artifacts and noise.

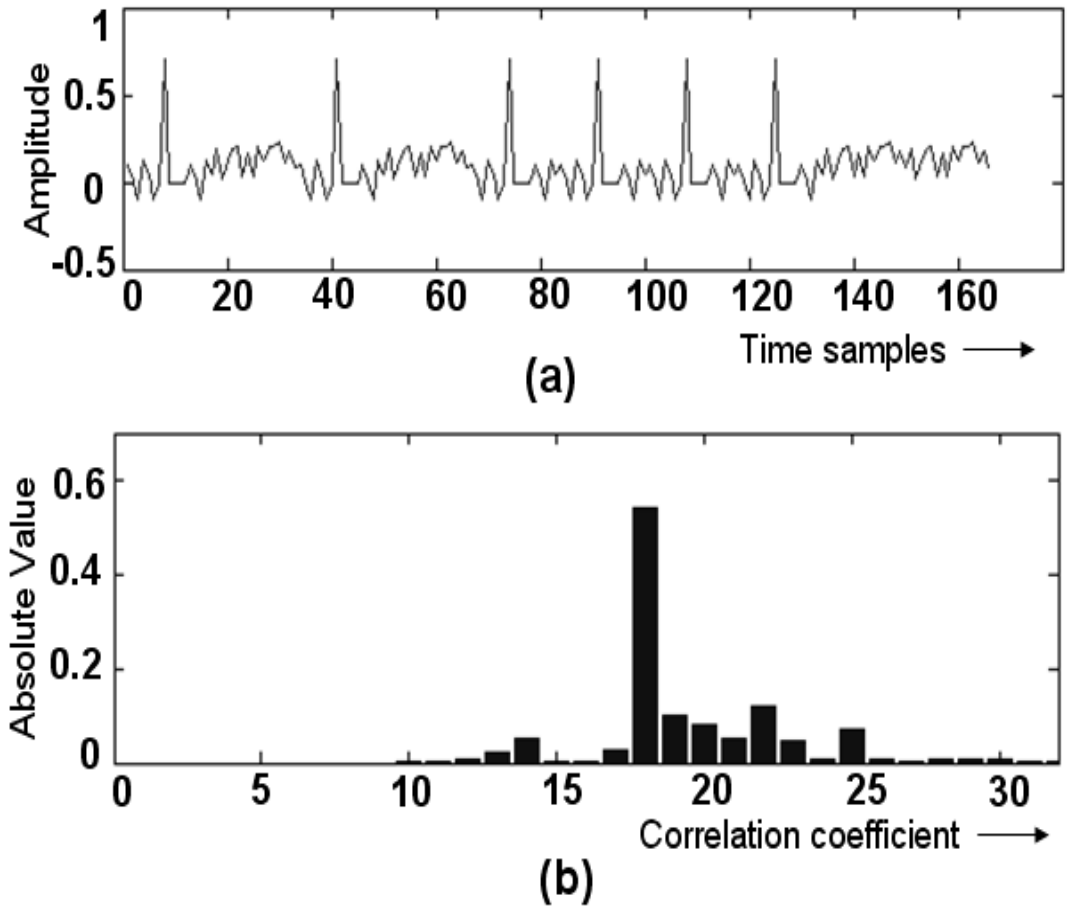


FIGURE C.10: (a) Case 6 - High Power Transients, (b) correlation coefficients.

The proposed algorithm does not depend on the definition of any specific artifact and therefore can be generalized unlike the existing method. This has also been validated with nine practical case studies. Since this is an initial attempt to solve the permutation indeterminacy from the artifact removal problem from ECG using ICA without human intervention, only simulation results are provided based on synthetic data. We do appreciate the fact that system design and deployment based on any particular algorithm for the sensitive fields such as medical science need significant amount of patient trials and huge amount of real data handling. Apart from the technical challenges, it involves waiting for the consent from the patients, nod from the hospital authorities as well as approval from the ethical committees which are bound to cause delay for the complete system validation in the real remote health monitoring environment. Nevertheless this forms part of our future research.

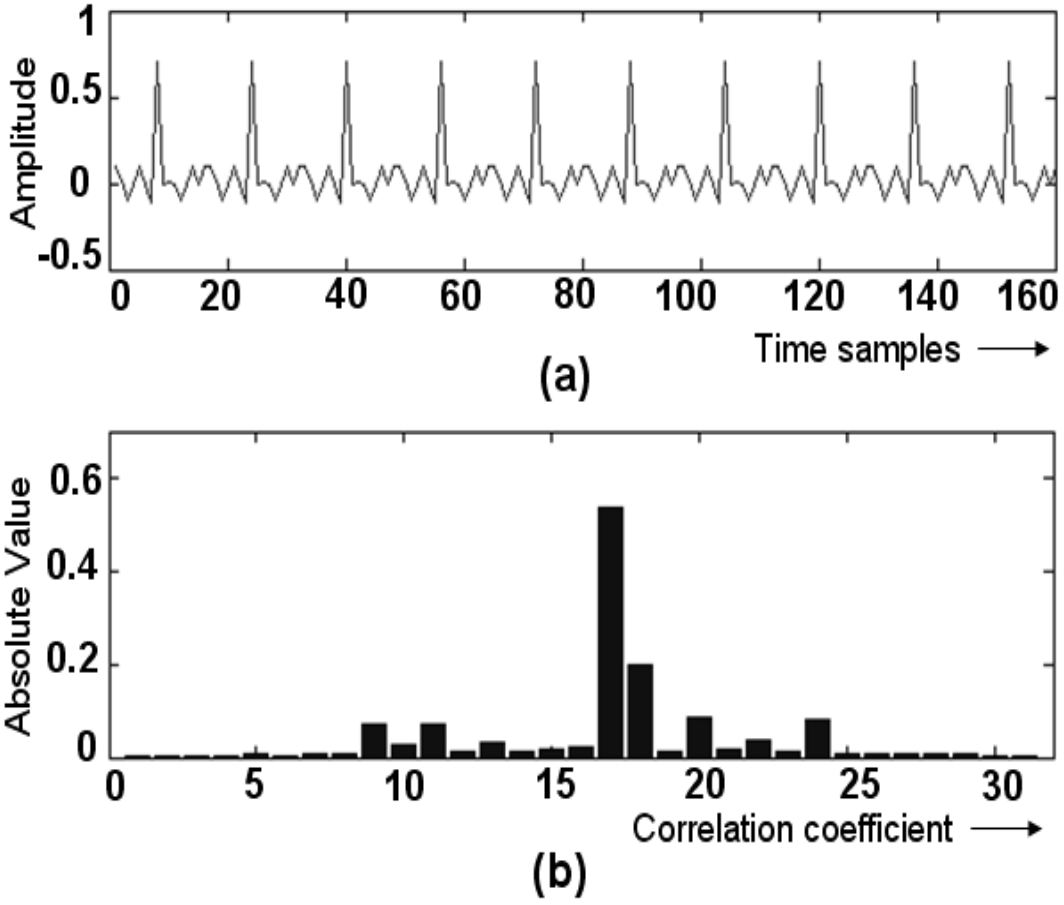


FIGURE C.11: (a) Case 7 - Noisy ECG, (b) correlation coefficients.

Bibliography

[1] L. Senhadji, G. Carrault, J. J. Bellanger, and G. Passariello, "Comparing wavelet transforms for recognizing cardiac patterns", *IEEE Engg. in Medicine and Biology Magazine*, vol. 14, pp. 167 - 173, 1995.

[2] C. Li, C. Zheng, and C. Tai, "Detection of ECG characteristic points using wavelet transforms", *IEEE Trans. Biomedical Engineering*, vol. 42, pp. 21 - 28, 1995.

[3] E. C. Karvounis, M. G. Tsipouras, D. I. Fotiadis, and K. K. Naka, "An automated methodology for fetal heart rate extraction from the abdominal electrocardiogram", *IEEE Trans. Information Technology in Biomedicine*, vol. 11, pp. 628 - 638, 2007.

[4] P. Addison, "The little wave with the big future", *Physics world*, vol. 17, pp. 35 - 39, 2004.

[5] Q. Pan, L. Zhang, G. Dai, and H. Zhang, "Two denoising methods by wavelet transform", *IEEE Transactions on Signal Processing*, vol. 47, pp. 3401 - 3406, 1999.

[6] T. He, G. Clifford, and L. Tarassenko, "Application of independent component analysis in removing artefacts from the electrocardiogram", *Neural Computing and Applications*, vol. 15, pp. 105 - 116, 2006.

[7] J. C. Chien, M. C. Huang, Y. D. Lin, and F. Chong, "A study of heart sound and lung sound separation by independent component analysis technique," 28^{th} *Annual Intl Conference of the IEEE Engg in Medicine and Biology Society*, pp. 5708-5711, September, 2006.

[8] Y. Liu and F. Li, "PCA: A Reference Architecture for Pervasive Computing", 1^{st} *International Symposium on Pervasive Computing and Applications*, pp. 99-103, 2006.

[9] I. Korhonen, J. Parkka and M. Gils, "Health Monitoring in the Home of the Future", *IEEE Engineering in Medicine and Biology Magazine*, vol. 22, no. 3, pp. 66-73, May-June, 2003.

[10] G. Borriello, V. Stanford, C. Narayanaswami and W. Menning, "Pervasive Computing in Healthcare", *Pervasive Computing, IEEE Computer Society*, pp. 17-19, 2007.

[11] T. L. Hayes, M. Pavel, N. Larimer, I. A. Tsay and J. Nutt, "Distributed Healthcare: Simultaneous Assessment of Multiple Individuals", *Pervasive Computing, IEEE Computer Society*, pp. 36-43, January- March, 2007.

[12] F. Wang, L. S. Docherty, K. J. Turner, M. Kolberg and E. H. Magill, "Services and Policies for Care At Home", *IEEE Pervasive Health Conference and Workshops*, pp. 1-10, Nov. 29- Dec. 1, 2006.

[13] N. Doulamis, "An Adaptable Emotionally Rich Pervasive Computing System", *European Signal Processing Conference (EUSIPCO)*, September 4-8, 2006, Italy.

[14] R. Chakravorty, "A Programmable Service Architecture for Mobile Medical Care", 4^{th} *Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, March, 2006.

[15] Y. V. Reddy, "Pervasive Computing: Implications, Opportunities and Challenges for the Society", 1^{st} *International Symposium on Pervasive Computing and Applications*, pp. 5-5, 2006.

[16] E. Jovanov, A. Lords, D. Raskovic, P. Cox, R. Adhami and F. Andrasik,"Stress Monitoring Using a Distributed Wireless Intelligent Sensor System", *IEEE Engineering in Medicine and Biology Magazine*, vol. 22, no. 3, pp. 49-55, May-June, 2003.

[17] K. J. Liszka, M. A. Mackin, M. J. Lichter, D. W. York, D. Pillai and D. S. Rosenbaum, "Keeping a Beat on the Heart", *IEEE Pervasive Computing*, pp. 42-49, October-December, 2004.

[18] U. Varshney, "Managing Comprehensive Wireless Patient Monitoring", *IEEE Pervasive Health Conference and Workshops*, pp. 1-4, Nov. 29, 2006.

[19] U. Varshney, "Pervasive Healthcare and Wireless Health Monitoring", *Journal of Mobile Networks and Applications, Springer Netherlands*, vol. 12, Nov. 2-3, June, 2007.

[20] J. A. Muras, V. Cahill and E. K. Stokes, "A Taxonomy of Pervasive Healthcare Systems", *IEEE Pervasive Health Conference and Workshops*, pp. 1-10, Nov. 29-dec. 1, 2006.

[21] S. Tilak, N. B. Abu-Ghazaleh and W. Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models", *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28-36, 2002.

[22] S. Cheekiralla and D. W. Engels, "A Functional Taxonomy of Wireless Sensor Network Devices", 2^{nd} *IEEE/ CreateNet Workshop on Broadband Advanced Sensor Networks (BaseNetS)*, October, 2005.

[23] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks", *IEEE Communications Magazine*, no. 2, pp. 102-114, August 2002.

[24] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors", *Communications of the ACM*, vol. 43, no. 5, pp. 51-58, May, 2000.

[25] D. Estrin, L. Girod, G. Pottie and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, pp. 2033-2036, 2001.

[26] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: A Survey", *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.

[27] A. Wang and A. Chandrakasan, "Energy-Efficient DSPs for Wireless Sensor Networks", *IEEE Signal Processing Magazine*, pp. 68-78, July, 2002.

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

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

[30] F. Truchetet and O. Laligant, "Wavelets in industrial applications: a review", *Wavelet Application in Industrial Processing II, Proceedings of SPIE*, vol. 5607, pp. 1-14, November, 2004.

[31] P. M. Bentley and J. T. E. McDonnell, "Wavelet transforms: an introduction", *Electronics and Communication Engineering Journal*, pp. 175-186, August, 1994.

[32] G. Strang, "Wavelets and Dilation Equations: A Brief Introduction", *SIAM Review, Society for Industrial and Applied Mathematics*, vol. 31, No. 4, pp. 614-6127, December, 1989.

[33] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, No. 7, July 1999.

[34] O. Rioul and M. Vetterli, "Wavelets and Signal Processing", *IEEE Signal Processing Magazine*, pp. 14-38, October, 1991.

[35] M. Vetterli and C. Herley, "Wavelets and Filter Banks: Theory and Design", *IEEE Trans. Signal Processing*, vol. 40, no. 9, pp. 2207-2232, September, 1992.

[36] O. Rioul and P. Duhamel, "Fast Algorithms for Discrete and Continuous Wavelet Transforms", *IEEE Trans. Information Theory*, vol. 38, no. 2, pp. 569-586, March, 1992.

[37] O. Rioul, "A Discrete-Time Multiresolution Theory", *IEEE Trans. Signal Processing*, vol. 41, no. 8, pp. 2591-2606, August, 1993.

[38] A. Graps, "An Introduction to Wavelets", *IEEE Computational Science and Engineering*, vol. 2, no. 2, pp. 50-61, Summer, 1995.

[39] A. Cohen and J. Kovacevic, "Wavelets: The mathematical Background", *Proceedings of IEEE*, vol. 84, no. 4, pp. 514-522, April, 1996.

[40] J. S. Walker, "Fourier Analysis and Wavelet Analysis", *Notices of the AMS*, vol. 44, no. 6, pp. 658-670, June-July, 1997.

[41] I. Daubechis, "Ten Lectures on Wavelets", *Published by SIAM*, 1992.

[42] C. K. Chui, "An Introduction to Wavelets", *Published by Academic Press*, 1992.

[43] S. G. Mallat, "A Wavelet Tour of Signal Processing", *Academic Press*, 2^{nd} Edition, 1999.

[44] B. B. Hubbard, "The World According To Wavelets: The Story of a Mathematical Technique in The Making", *published by A. K. Peters*, 1998.

[45] T. C. Denk and K. K. Parhi, "VLSI Architectures for Lattice Structure Based Orthonormal Discrete Wavelet Transforms", *IEEE Trans. Circuits and Systems-II : Analog and Digital Signal Processing*, vol. 44, no. 2, pp. 129-132, February, 1997.

[46] J. Fridman and E. S. Manolakos, "Discrete Wavelet Transform: Data Dependence Analysis and Synthesis of Distributed Memory and Control Array Architectures", *IEEE Trans. Signal Processing*, vol. 45, no. 5, pp. 1291-1308, May, 1997.

[47] G. Knowels, "VLSI Architecture For The Discrete Wavelet Transform", *Electronic Letters*, vol. 26, no. 15, pp. 1184-1185, 19 July, 1990.

[48] K. K. Parhi and T. Nishitani, "VLSI Architectures for Discrete Wavelet Transforms", *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, pp. 191-202, June, 1993.

[49] A. Grzeszczak, T. H. Yeap and S. Panchanathan, "VLSI Architecture For Discrete Wavelet Transform", *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 461-464, September, 1994.

[50] J. Fridman and E. S. Manolakos, "Distributed Memory and Control VLSI Architectures for the 1-D Discrete Wavelet Transform", *IEEE Workshop on VLSI Signal Processing*, vol. VII, pp. 388-397, October, 1994.

[51] M. Vishwanath, R. M. Owens and M. J. Irwin, "VLSI Architectures For The Discrete Wavelet Transform", *IEEE Trans. Circuits and System-II: Analog And Digital Signal Processing*, vol. 42, no. 5, pp. 305-316, May, 1995.

[52] X. Chen, T. Zhou, Q. Zhang, W. Li and H. Min, "A VLSI Architecture for Discrete Wavelet Transform", *IEEE International Conference on Image Processing*, vol. 1, pp. 1003-1006, September, 1996.

[53] A. Grzeszczak, M. K. Mandal, S. Panchanathan and T. Yeap, "VLSI Implementation of Discrete Wavelet Transform", *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 4, no. 4, pp. 421-433, December, 1996.

[54] C. Yu, C. A. Hsieh and S. J. Chen, "Design and Implementation of a Highly efficient VLSI Architecture for Discrete Wavelet Transform", *IEEE Custom Integrated Circuit Conference*, pp. 237-240, 1997.

[55] J. M. Jou, Y. H. Shiau and C. C. Liu, "Efficient VLSI Architectures for the Biorthogonal Wavelet Transform by Filter Bank and Lifting Scheme", *IEEE International Symp. Circuits and Systems*, pp. 529-532, 2001.

[56] M. Alam, C. A. Rahman, W. Badawy and G. Jullien, "Efficient Distributed Arithmetic Based DWT Architecture for Multimedia Applications ", *IEEE International Workshop on System-on-Chip for Real-Time Applications*, pp. 333-336, 2003.

[57] X. Cao, Q. Xie, C. Peng, Q. Wang and D. Yu, "An Efficient VLSI Implementation of Distributed Architecture for DWT", *IEEE* 8^{th} *Workshop on Multimedia Signal Processing*, pp. 364-367, 2006.

[58] M. Martina and G. Masera, "Low-Complexity, Efficient 9/7 Wavelet Filters VLSI Implementation", *IEEE Trans. Circuits and Systems-II: Express Briefs*, vol. 53, no. 11, pp. 1289-1293, November, 2006.

[59] K. G. Oweiss, A. Mason, Y. Suhail, A. M. Kamboh and K. E. Thomson, "A Scalable Wavelet Transform VLSI Architecture for Real-Time Signal Processing in High-density Intra-Cortical Implants", *IEEE Trans. Circuits and Systems-I: Regular Papers*, vol. 54, no. 6, pp. 1266-1278, June, 2007.

[60] M. Martina and G. Masera, "Multiplierless, Folded 9/7 - 5/3 Wavelet VLSI Architecture", *IEEE Trans. Circuits and Systems-II: Express Briefs*, vol. 54, no. 9, pp. 770-774, September, 2007.

[61] F. J. Taylor, "An Analysis of the Distributed Arithmetic Digital Filter", *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 34, no. 5, pp. 1165-1170, October, 1986.

[62] A. Bellaouar and M. I. Elmasry, "Low-power Digital VLSI Design: Circuits and Systems", *Springer Publication*, 1995.

[63] W. P. Burleson, "Polynomial Evaluation in VLSI Using Distributed Arithmetic", *IEEE Trans. Circuits and Systems*, vol. 37, no. 10, pp. 1299-1304, October, 1990.

[64] M. Unser and A. Aldroubi, "A Review of Wavelets in Biomedical Applications", *Proceedings of IEEE*, vol. 84, no. 4, pp. 626-638, April, 1996.

[65] A. Aldroubi and M. Unser, "Wavelets in Medicine and Biology", *Published by CRC Press*, 1996.

[66] M. Akay, "Wavelet Applications in Medicine", *IEEE Spectrum*, pp. 50-56, 1997.

[67] M. L. Jacobson, "Analysis and Classification of Physiological Signals Using Wavelet Transforms", 10^{th} *IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 906-909, December, 2003.

[68] H. Hassanpour and A. Parsaei, "Fetal ECG Extraction Using Wavelet Transform", *IEEE Computer Society CIMCA and IAWTIC*, 2006.

[69] A. Khamene and S. Negahdaripour, "A New Method for the Extraction of Fetal ECG from the Composite Abdominal Signal", *IEEE Trans. Biomedical Engineering*, vol. 47, no. 4, pp. 507-516, April, 2000.

[70] P.de Chazal, B.G. celler and R.B. Reilly, "Using Wavelet Coefficients for the Classification of the Electrocardiogram", 22^{nd} *Annual EMBS Conference*, pp. 64-67, July, 2000.

[71] P. S. Addison, "Wavelet Transforms and the ECG: A Review", *Physiol. Meas.*, vol-26, 2006.

[72] R. Polikar, "The Wavelet Tutorial", available online (http://users.rowan.edu/ polikar/wavelets/wtpart1.html).

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

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

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

[76] K. K. Shyu, M. H. Lee, Y. T. Wu and P. L. Lee, "Implementation of Pipelined FastICA on FPGA for Real-Time Blind Source Separation", *IEEE Trans. Neural Networks*, vol. 19, no. 6, pp. 958-970, June, 2008.

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

[78] T. A. Newton, "A Simple Algorithm for Finding Eigenvalues and Eigenvectors for 2x2 Matrices", *The American Mathematical Monthly*, Vol. 97, no. 1, pp. 57-60, January, 1990.

[79] G. James, "Modern Engineering Mathematics", *Pearson-Prentice Hall*, 4^{th} edition, 2007.

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

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

[82] A. A. Liddicoat and M. J. Flynn, "Parallel Square and Cube Computation", 34^{th} *Asilomar Conference on Signals, Systems and Computers*, pp. 1325-1329, 2000.

[83] K. Hwang, "Computer Arithmetic: Principles, Architecture and Design", *Wiley Publishing*, 1979.

[84] M. J. Schulte and K. E. Wires, "High-Speed Inverse Square Roots", 14^{th} *IEEE Symposium on Computer Arithmetic*, pp. 124-131, 1999.

[85] J. C. Majithia, "Non Restoring Binary Division Using A Cellular Array", *Electronics Letters*, Vol. 6, no. 10, pp. 303-304, 14^{th} May, 1970.

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

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

[88] C. Jutten and J. Herault, "Blind Separation of Sources, Part I: An Adaptive Algorithm Based on Neuromimetic Architecture", *Journal of Signal Processing, Elsevier Science Publishers*, vol. 24, pp. 1-10, 1991.

[89] J.F. Cardoso, "Blind Signal Separation: Statistical Principles", *Proceedings of The IEEE*, vol. 86, no. 10, pp. 2009-2025, October, 1998.

[90] A. Cichocki and S. Amari, "Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications", *John Wiley and Sons Ltd*, 2002.

[91] J. V. Stone, "Independent Component Analysis: A Tutorial Introduction", *MIT Press*, 2004.

[92] A. Cichocki, "What Is Intelligent Blind Signal Processing and Why It Is Important in Brain Science?", *RIKEN Brain Science Institute News, Japan*, no. 4, June, 1999.

[93] M. T. Pourazad, Z. Moussavi, F. Farahmand and R. K. Ward, "Heart Sounds Separation From Lung Sounds Using Independent Component Analysis", *IEEE Annual Conference on Engineering in Medicine and Biology*, pp. 2736-2739, September, 2005.

[94] R. Mutihac and R. C. Mutihac, "A Comparative Study of Independent Component Analysis Algorithms For Electroencephalography", *Romanian Reports in Physics*, vol. 59, no. 3, pp. 827-853, 2007.

[95] V. D. Calhoun and T. Adali, "Unmixing fMRI with Independent Component Analysis", *IEEE Engineering in Medicine and Biology Magazine*, pp. 79-90, March/ April, 2006.

[96] T. Aoyagi, H. Tokutaka, F. Fujimura and T. Maniwa, "Application of FastICA to Pulse Wave", *Proceedings of the* 9^{th} *International Conference on Neural Information Processing (ICONIP)*, vol. 2, pp. 769-772, 2002.

[97] Y. Tie and M. Sahin, "Separation of Spinal Cord Motor Signals Using the FastICA Method", *Journal of Neural Engineering*, vol. 2, pp. 90-96, 2005.

[98] S. Comani, M. Liberati, D. Mantini, E. Gabriele, D. Brisinda, S. D. Luzio, R. Fenici and G. Luca, "Characterization of Fetal Arrhythmias by Means of Fetal Magnetocardiography in Three Cases of Difficult Ultrasonographic Imaging", *Pacing and Clinical Physiology (PACE)*, vol. 27, no. 12, pp. 1647-1655, December, 2004.

[99] A. Hyvärinen and E. Oja, "Independent Component Analysis: Algorithms and Applications", *Neural Networks*, vol. 13, no. 4-5, pp. 411-430, June 2000.

[100] A. Hyvärinen, J. Karhunen and E. Oja, "Independent Component Analysis", *A Wiley Inter-Science Publication*, 2001.

[101] S. Roberts and R. Everson, "Independent Component Analysis: Principles and Practice", *Cambridge University Press*, 2001.

[102] S. Bermejo, "Finite Sample Effects of the FastICA Algorithm", *Neurocomputing, Elsevier*, vol. 71, pp. 392-399, 2007.

[103] N. N. Madras, "Lectures on Monte carlo Methods"', *American Mathematical Society*, 2002.

[104] J. E. Gentle, "Random Number Generation and Monte Carlo Methods", *New York: Springer*, 1998.

[105] M. Sharaf, H. A. K. Mansour, H. H. Zayed and M. L. Shore, "A Complex Linear Feedback Shift Register Design for the A5 Keystream Generator", *National Radio Science Conference*, Cairo, Egypt, March 15-17, 2005.

[106] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3-30, January, 1998.

[107] X. Li, J. Le and L. T. Pileggi, "Statistical Performance Modeling and Optimization", *Foundations and Trends in Electronic Design Automation*, vol. 1, no.4, pp. 331-480, 2006.

[108] V. J. Easton and J. H. McColl, "Statistics Glossary" (available online: *http://www.stats.gla.ac.uk/steps/glossary/*)

[109] T. W. Korner.,"Fourier Analysis ", *Cambridge University Press*, Cambridge,1996.

[110] S. C. Pei and M. H. Yeh, "Time and frequency split Zak transform for finite Gabor expansion", Signal Processing, vol. 52, no. 3, pp. 323-341, August, 1996.

[111] V. Venkatachalam and L. A. Jorge, "Nonstationary Signal Enhancement Using the Wavelet Transform ", Proceedings of the Twenty-Eighth Southeastern Symposium on System Theory, pp. 98-102, 1996.

[112] M. K. Tsatsanis and G. B. Giannakis, "Time-Varying System Identification and Model Validation Using Wavelets", *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3512-3523, Dec. 1993.

[113] Y. Zheng, Z. Lin and D. B. H. Tai, "Time-varying parametric system multiresolution identification by wavelets", *Int. Journal of System Science*, vol. 32, no. 6, pp. 775-793, 2001.

[114] Y. Dorfan, A. Feuer and B. Porat, "Modeling and Identification of LPTV Systems by Wavelets", *Signal Processing*, vol. 84, pp. 1285-1297, 2004.

[115] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review", *IEEE ASSP Magazine*, pp. 4-19, July, 1989.

[116] D. Benyamin, W. Luk and J. Villasenor, "Optimizing FPGA-based Vector Product Designs", 7^{th} *Annual IEEE Symposium on Field-programmable Custom Computing Machines*, pp. 188-197, 1999.

[117] 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.

[118] A. Hyvärinen, "Survey on Independent Component Analysis", *Neural Computing Surveys*, vol. 2, pp. 94-128, 1999.

[119] Z. K. Silagadze, "Multi-dimensional Vector Product", *A Mathematical and General Journal of Physics*, vol. 35, no. 23, pp. 4949-4953, 2002.

[120] D. C. Lay, "Linear Algebra And Its Applications", *Addison-Wesley*, Second Ed., Chapter 3, December, 1999.

[121] S. I. Grossman, "Elementary Linear Algebra", *Saunders College Publishing*, Fifth Ed., Chapter 2, 1994.

[122] A. Acharyya, K. Maharatna and B. M. Al-Hashimi, "Hardware Reduction Methodology for 2-Dimensional Kurtotic Fast ICA Based on Algorithmic Analysis and Architectural Symmetry", *IEEE Workshop on Signal Processing Systems*, October, 2009.

[123] F. L. Bauer, "Computational Graphs And Rounding Error", *SIAM J. Numer. Anal.*, vol. 11, no. 1, pp. 87-96, March 1974.

[124] J. Stoer and R. Bulirsch, "Introduction To Numerical Analysis", *Springer-Verlag*, Chapter 1, 1983.

[125] V. B. Eckmann, "Stetige Lösungen Linearer Gleichungssysteme", *Comm. Math. Helv.*, vol. 15, pp. 318-339, 1943.

[126] J. F. Adams, "Vector Fields of Spheres", *The Ann. of Math.*, vol. 75, no. 3, pp. 603-632, May, 1962.

[127] G. W. Whitehead, "Note on Cross-sections in STIEFEL Manifolds", *Comm. Math. Helv.*, vol. 37, pp. 239-240, 1963.

[128] R. B. Brown and A. Gray, "Vector Cross Products", *Comm. Math. Helv.*, vol. 42, pp. 222-236, 1967.

[129] A. Gray, "Vector Cross Products on Manifolds", *Trans. Amer. Math. Soc.*, vol. 141, pp. 469-504, Jul., 1969.

[130] W. S. Massey, "Cross Products of Vectors in Higher Dimensional Euclidean Spaces", *The Amer. Math. Monthly*, vol. 90, no. 10, pp. 697-701, Dec., 1983

[131] R. Shaw, "Vector Cross Products in n Dimensions", *Int. J. Math. Educ. Sci. Technol.*, vol. 18, no. 6, pp. 803-816, 1987.

[132] J. F. Blinn, "A Homogeneous Formulation for Lines in 3 Space", *Computer Graphics (Proc. Siggraph)*, vol. 11, no. 2, pp. 237-241, 1977.

[133] J. F. Blinn, "Lines in Space - Part 1: The 4D Cross Product", *IEEE Computer Graphics and Applications*, vol. 23, no. 2, pp. 84-91, March/April, 2003.

[134] B. Hoffmann, "About Vectors", *Dover Publications Inc*, Chapter 2, March 2003.

[135] R. Shaw and F. J. Yeadon, "On (a x b) x c", *The Amer. Math. Monthly*, vol. 96, no. 7, pp. 623-629, Aug - Sep., 1989.

[136] A. Dittmer, "Cross Product Identities in Arbitrary Dimension", *The Amer. Math. Monthly*, vol. 101, no. 9, pp. 887-891, Nov., 1994.

[137] A. Acharyya, K. Maharatna, J. Sun, B. M. Al-Hashimi and S. R. Gunn, "Hardware Efficient Fixed-Point VLSI Architecture for 2D Kurtotic FastICA", 19^{th} *European Conference on Circuit Theory and Design*, pp. 165-168, August, 2009.

[138] J. C. Majithia, "Pipeline Array For Square-Root Extraction", *Electronics Letters*, vol. 9, no. 1, pp. 4-5, 1973.

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

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

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

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

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

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

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

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

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

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

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

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

[151] 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, 1993.

[152] X. Hu and S. C. Bass, "A Neglected Error Source in the CORDIC Algorithm", *IEEE International Symposium on Circuits and Systems*, pp. 766-769, 1993.

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

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

[155] B. Boucheham, Y. Ferdi and M. C. Batouche, "Piecewise Linear Correction of ECG Baseline Wander: A Curve Simplification Approach", *Comp. Methods and Prog. in Biomed.*, vol. 78, pp. 1-10, 2005.

[156] A. K. Barros, A. Mansour and N. Ohnishi, "Removing Artifacts from Electrocardiographic Signals Using Independent Component Analysis", *Neurocomputing*, vol. 22, pp. 173-186, 1998.

[157] W. Zhou and J. Gotman, "Removal of EMG and ECG Artifacts from EEG based on Wavelet Transform and ICA", *IEEE EMBS Conference*, pp. 392-395, September, 2004.

[158] J. Yao and S. Warren, "A Short Study to Assess the Potential of Independent Component Analysis for Motion Artifact Separation in Wearable Pulse Oximeter Signals", *IEEE EMBS Conference*, pp. 3585-3588, September, 2005.

[159] E. Briselli *et. al*, "An Independent Component Analysis based Approach on Ballistocardiogram Artifact Removing", *Magnetic Resonance Imaging*, vol. 24, pp. 393-400, 2006.

[160] J. Escudero, R. Hornero, D. Abásolo, A. Fernández and M. L. Coronado, "Artifact Removal in Magnetoencephalogram Background Activity with Independent Component Analysis", *IEEE Trans. Biomed. Engg.*, vol. 54, no. 11, November, 2007.

[161] J. Taelman, S. V. Huffel and A. Spaepen, "Wavelet-Independent Component Analysis to Remove Electrocardiography Contamination in Surface Electromyography", *IEEE EMBS Conference*, pp. 682-685, 2007.

[162] M. Milanesi *et. al.*, "Independent Component Analysis Applied to the Removal of Motion Artifacts from Electrocardiographic Signals", *Med. Biol. Eng. Comput.*, vol. 46, pp. 251-261, 2008.

[163] A. Prieto, C. G. Puntonet and B. Prieto, "A Neural Learning Algorithm for Blind Separation of Sources based on Geometric Properties", *Signal Processing*, vol. 64, pp. 315-331, 1998.

[164] J. T. Catalano, "Guide to ECG Analysis", *Lippincott*, chapter-2, second edition, 2002.

[165] D. L. Jones *et. al.*, "Heart Disease and Stroke Statistics 2010 Update: A Report From the American heart Association", Circulation, vol. 121, pp. e46-e215, 2010.

[166] S. Allender *et. al.*, "European Cardiovascular Disease Satistics: 2008 Edition".

[167] "International Cardiovascular Disease Statistics", Statistical Fact Sheet - Populations, American Heart Association, 2004.

[168] http://www.who.int/cardiovascular_diseases/en/

[169] Population Projection EUROPOP 2008 by European Commission - http://epp.eurostat.ec.europa.eu

[170] American Heart Association statistical fact sheet Population 2007 update: International cardiovascular disease statistics.

[171] Euro Heart Survey: Cardiovascular diseases in Europe 2006, *Report of European Society of Cardiology.*

[172] "Preparing for an Aging Society: Challengers Faced by Healthcare System in European Union, Japan and United States", The Economic Research and Analytic Group at Frost and Sullivan, 2008.

[173] Jose Leal, Ramon Luengo-Fernandez, Alastair Gray, Sophie Petersen and Mike Rayner, "Economic burden of cardiovascular diseases in the enlarged European Union", *European Heart Journal*, vol. 27, pp. 1610 - 1619, 2006.

[174] A Report by Institute of Healthcare Improvement, 2007. (http://www.ihi.org/ihi)

[175] ESC press release, "Treatment of European coronary patients fails to meet standards of international guidelines", Results of third EUROASPIRE survey - Embargoed for release 13 March 2009

[176] Advanced Research and Technology for EMbedded Intelligence and Systems (ARTEMIS) Joint Undertaking (ARTEMIS JU) Sub-programmes-II (ASP-2): Healthcare Systems, 2010. (https://www.artemisia-association.org/attachments/933/ARTEMIS-IA-rolling_presentation.pdf)

[177] I. Graham et al., "European guidelines on cardiovascular disease prevention in clinical practice: fourth joint task force of the European Society of Cardiology and other societies", *Eur Journal of Cardiovasc Prev Rehabil*, vol. 14, suppl. 2, pp. S1 - S113, 2007.

[178] "e-health INTEROP Report- Phase 1 and 2", issued by The European Committee for Standardization (CEN)/the European Committee for Electrotechnical Standardization (CENELEC)/ the European Telecommunications Standards Institute (ETSI) in February, 2009.

[179] PHS 2020, support action project co-financed by the European Commission FP7-IST-2007 "The role of ICT in healthcare" http://www.ehealthnews.eu/phs2020

[180] E-Health and Personalized Health Care, The European Heart Rhythm Association (EHRA) Summit 2010, European Heart House, European Society of Cardiology, Sophia Antiopolis, March 22-23, 2010.

[181] The European Commission Recommendation *on cross-border interoperability of electronic health record systems*, July 2, 2008, Brussels, Belgium.

[182] W. Jin, "ECG Signals Analysis using Wavelets in Frequency Domain", *MSc. Dissertation in Systems-on-Chip*, School of Electronics and Computer Science, University of Southampton, September, 2010. (Supervised by A. Acharyya and K. Maharatna).

[183] T. Chen, "Computationally Efficient Generalized Algorithm for N-D determinant Computation", *MSc. Dissertation in System-on-Chip*, School of Electronics and Computer Science, University of Southampton, September, 2010. (Supervised by A. Acharyya and K. Maharatna)