# HIGH PERFORMANCE, LOW COST SUBSPACE DECOMPOSITION AND POLYNOMIAL ROOTING FOR REAL TIME DIRECTION OF ARRIVAL ESTIMATION: ANALYSIS AND IMPLEMENTATION

Mrudula V. Athi
*Michigan Technological University*

Follow this and additional works at: https://digitalcommons.mtu.edu/etds
Part of the Electrical and Computer Engineering Commons, and the Mathematics Commons

# HIGH PERFORMANCE, LOW COST SUBSPACE DECOMPOSITION AND POLYNOMIAL ROOTING FOR REAL TIME DIRECTION OF ARRIVAL ESTIMATION: ANALYSIS AND IMPLEMENTATION

By

Mrudula V. Athi

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2014

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE In Electrical Engineering.

Department of Electrical and Computer Engineering

| | |
|---|---|
| Thesis Advisor: | *Dr. Seyed A. (Reza) Zekavat* |
| Committee Member: | *Dr. Timothy C. Havens* |
| Committee Member: | *Dr. Allan Struthers* |
| Department Chair: | *Dr. Daniel R. Fuhrmann* |

To amma appa

for unwavering belief in my capabilities that has been my biggest source of inspiration and for their constant support.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

I would like to thank Dr. Seyed A. Zekavat, who guided from the very first day of my Masters and gave valuable inputs for the progress of my thesis. I thank him for his constant attention to detail, technical guidance in preparing papers and helping me find a direction for my resaerch.

I thank my committee - Dr. Dr. Timothy C. Havens and Dr. Allan Struthers, who took time out of their busy schedule to evaluate and make the thesis better.

I would like to further thank Dr.Allan Struthers for his valuable inputs to mathematical aspects of my thesis. His class, which I took in Fall 2013 was helpful in developing portion of my thesis.

I'm particularly thankful to Shankar Giri Venkata Giri, fellow student who helped me understand various aspects of FPGA implementation and hardware design. Many discussions that I had with him over the course of two years have helped me scrutinize my ideas. I am thankful for his patience and constant support as a friend.

I would like to thank Andrew Boettcher, fellow classmate with whom I presented a poster for his help in understanding few complex mathematical concepts.

Asif Al-Rasheed ,Amir Torabi and Mohsen Jamalabdollahi fellow researchers and friends provided moral support and encouragement.

Special thanks go out to Chuck and Mark, for instant and proactive help with any issues in the lab. They were awesome and immensely approachable.

Finally, I would like to thank my parents, my sister and Swaraj, who were an endless source of encouragement and support.

# Abstract

This thesis develops high performance real-time signal processing modules for direction of arrival (DOA) estimation for localization systems. It proposes highly parallel algorithms for performing subspace decomposition and polynomial rooting, which are otherwise traditionally implemented using sequential algorithms. The proposed algorithms address the emerging need for real-time localization for a wide range of applications. As the antenna array size increases, the complexity of signal processing algorithms increases, making it increasingly difficult to satisfy the real-time constraints. This thesis addresses real-time implementation by proposing parallel algorithms, that maintain considerable improvement over traditional algorithms, especially for systems with larger number of antenna array elements. Singular value decomposition (SVD) and polynomial rooting are two computationally complex steps and act as the bottleneck to achieving real-time performance. The proposed algorithms are suitable for implementation on field programmable gated arrays (FPGAs), single instruction multiple data (SIMD) hardware or application specific integrated chips (ASICs), which offer large number of processing elements that can be exploited for parallel processing. The designs proposed in this thesis are modular, easily expandable and easy to implement.

Firstly, this thesis proposes a fast converging SVD algorithm. The proposed method reduces the number of iterations it takes to converge to correct singular values, thus achieving closer to real-time performance. A general algorithm and a modular system design are provided making it easy for designers to replicate and extend the design to larger matrix sizes. Moreover, the method is highly parallel, which can be exploited in various hardware platforms mentioned earlier. A fixed point implementation of proposed SVD algorithm is presented. The FPGA design is pipelined to the maximum extent to increase the maximum achievable frequency of operation. The system was developed with the objective of achieving high throughput. Various modern cores available in FPGAs were used to maximize the performance and details of these modules are presented in detail.

Finally, a parallel polynomial rooting technique based on Newton's method applicable exclusively to root-MUSIC polynomials is proposed. Unique characteristics of root-MUSIC polynomial's complex dynamics were exploited to derive this polynomial rooting method. The technique exhibits parallelism and converges to the desired root within fixed number of iterations, making this suitable for polynomial rooting of large degree polynomials. We believe this is the first time that complex dynamics of root-MUSIC polynomial were analyzed to propose an algorithm.

In all, the thesis addresses two major bottlenecks in a direction of arrival estimation system, by providing simple, high throughput, parallel algorithms .

# Chapter 1

# Introduction

Localization is a key component of various modern day applications such as wireless sensor networks (WSNs) [1], wireless body area networks (WBANs) [2], localization in acoustic arrays [3], network based localization [4, 5], localization in space based solar power generation [6] etc. These applications have a wide range of impact ranging from user experience improvement to effective safety and monitoring and exploring new frontiers of science and technology. Implementation of localization techniques require fairly complex signal processing algorithms. The complexity of these signal processing algorithms is magnified due to the fact that most of its applications demand real-time output. In addition, many localization applications are in mobile systems which may have to limited processing power and battery life. Thus, development of low complexity, and high performance localization algorithms are vital to its emerging applications. New algorithms amenable to parallelization, choice of platform suitable for such algorithms, efficient implementation and design on selected hardware are the key to developing a good real-time signal processing module.

This thesis was motivated by the need for real-time signal processing algorithms for wireless local positioning system (WLPS) [7] being developed at Michigan Technological University. WLPS is an active positioning system based on direct sequence-code division multiple access (DS-CDMA) scheme for indoor and urban areas. More specifically, this thesis addresses the need for real-time signal processing modules for direction of arrival(DOA) techniques.

This chapter offers an overview of the WLPS and introduces various components of the system. Different direction of arrival (DOA) techniques are discussed and root-MUSIC system is described in detail. Design and implementation aspects of a generic real-time signal processing system are introduced. Finally, we introduce the contributions of this thesis for the DOA root-MUSIC algorithm for implementation on WLPS signal processing platforms.

## 1.1 Wireless Local Positioning System (WLPS)

### 1.1.1 Localization

Positioning systems can be categorized as global positioning system (GPS) and local positioning system (LPS). GPS is a widely known precise satellite based positioning system. However, it does not perform well in indoor areas or urban areas due to reflections. Therefore, it is usually integrated with other local positioning systems such as LPS. LPS can be broadly categorized into two types: self positioning, where a mobile device finds its own instantaneous location with resepct to a fixed point, and remote positioning, where a mobile device finds the instantaneous position of other objects with respect to its own position. WLPS that is a system that is being developed at the Wireless Lab of Michigan Tech, is an active remote LPS that allows relative positioning of mobiles via mobiles.

### 1.1.2 WLPS - working and design

WLPS proposed in [8], is a remote active localization system that incorporates DS-CDMA signaling to localize mobile nodes. WLPS uses direction of arrival (DOA) and round trip time of arrival (TOA) of the signal from an active node to estimate the position of the node. Round trip TOA avoids a need to time synchronization that is a key component

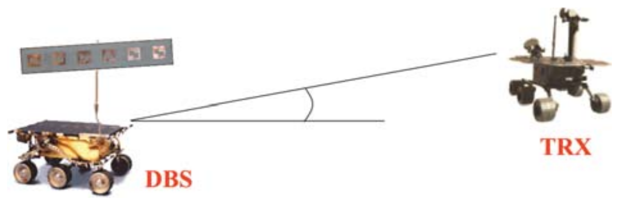


**Figure 1.1:** Wireless Local Positioning System

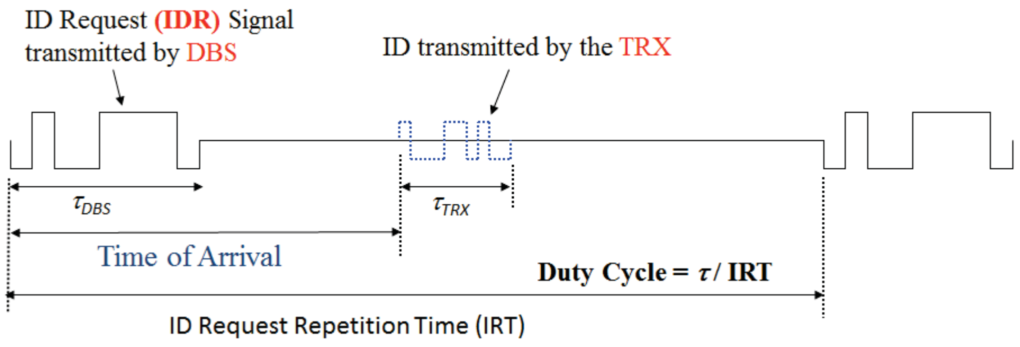

ID Request (IDR) Signal transmitted by DBS
ID transmitted by the TRX
$\tau_{DBS}$
$\tau_{TRX}$
Time of Arrival
Duty Cycle = $\tau$ / IRT
ID Request Repetition Time (IRT)

**Figure 1.2:** WLPS signaling scheme

of many TOA estimation algorithms. This reduces the complexity of this system that is vital for mobile applications. WLPS consists of a dynamic base station (DBS) in each of the monitoring mobile and a transponder (TRX) in target mobile as show in Figure 1.1. Each DBS is assigned a unique identification (ID). When a DBS trasnmits an ID request signal (IDR) to all the targets in its neighbourhood, the TRX responds with their IDs. On recognition of the target by DBS, it positions and tracks the target. Figure 1.2 represents the communication scheme described above.

The performance of WLPS system can be degraded due to inter-TRX-interference (IXI) at the DBS as well as inter-DBS-interference (IDI) at TRX. Multiple access (MA) schemes such as spatial division MA (SDMA) [9] and DS-CDMA [10] are employed to reduce

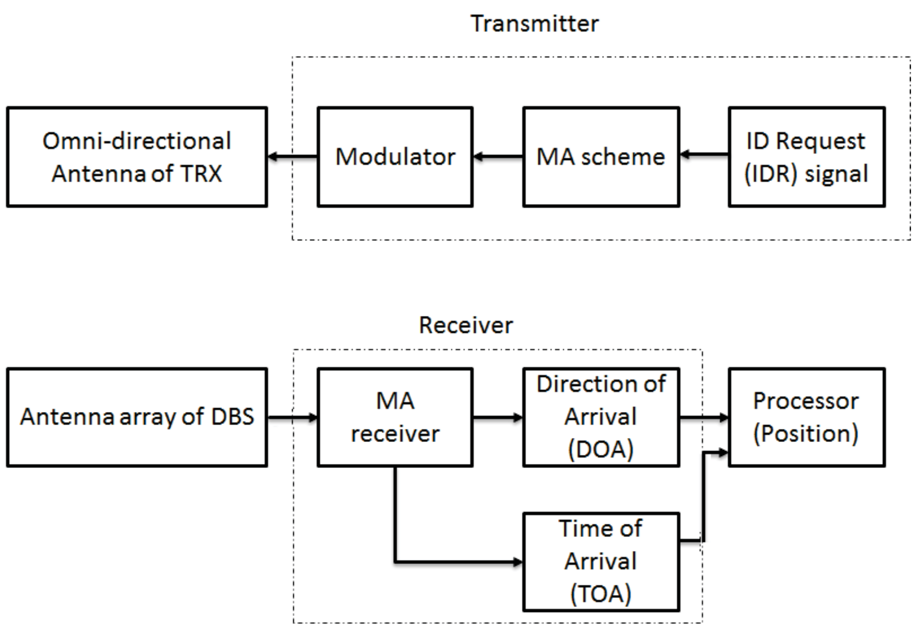


**Figure 1.3:** WLPS Structure

the interference effects. SDMA via beamforming (BF) and DS-CDMA are employed at the DBA. Therefore, TRX is simple and needs an omnidirectional antenna, a simple demodulator and a DS-CDMA transmitter. DBS as shown in Figure 1.3 has a complex structure. The receiver consists of an antenna array for SDMA, a MA receiver and a diversity combiner. The baseband signal is used for TOA and DOA estimation, which in turn is used for localization and tracking. Figure 1.4 represents the DBS structure in detail. RF front-end, analog to digital converter (ADC), digital down converter (DDC), phase-amplitude compensation with synchronisation and frequency tracking form the MA receiver. The baseband signal undergoes beamforming and is then used for channel estimation and TOA computation. DOA is obtained from individual channels before beamforming. The theoretical analysis of the system was carried in [7, 8] and various

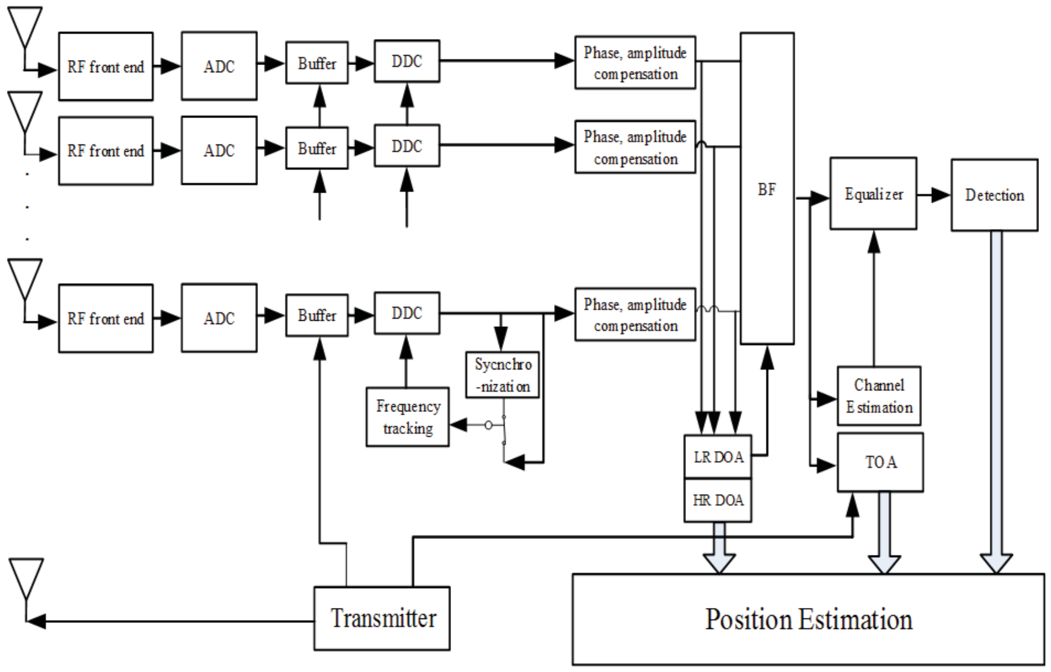


**Figure 1.4:** DBS Structure

aspects of receiver implementation were dealt in [11, 12]. The optimum beamforming for WLPS were explored in [13, 14] and suitable TOA estimation was proposed in [15]. A novel DOA estimation technique was proposed in [16] and ways to fuse DOA and TOA information was proposed in [17]. In this thesis, we dwell further into DOA estimation techniques by proposing high performance, low cost real-time signal processing algorithms with emphasis on implementation.

## 1.2 Antenna Array

An antenna array defined as a set of two or more antennas, is an important component of DBS receiver. Antenna arrays serve various purposes such as increasing overall gain, providing spatial diversity, optimizing signal to interference plus noise ratio (SINR) and DOA estimation [9]. Antenna arrays of different geometries, spacing and dimensions are used in various applications. For the purpose of WLPS, a linear array of $N$ equally spaced antenna elements is used. In WLPS, antenna array is used for beamforming and DOA estimation. Figure 1.5 shows an antenna array of patch antennas developed for WLPS at the Wireless Positioning Lab of Michigan Tech.

Let $x_1,\ldots,x_N$ be the signals received by antenna elements 1 to $N$. The output from antenna elements are multiplied in weights $w_1,\ldots,w_N$ and summed together resulting in a signal

$$y = \sum_{n=1}^{N} w_n x_n = W^H X \tag{1.1}$$

Let $\lambda$ be the wavelength, d the distance of separation and $\theta_{DOA}$ be the DOA, then array vector is given as

$$V = [1, e^{\frac{-j2\pi d\sin(\theta_{DOA})}{\lambda}}, \ldots, e^{\frac{-j2\pi(N-1)d\sin(\theta_{DOA})}{\lambda}}] \tag{1.2}$$

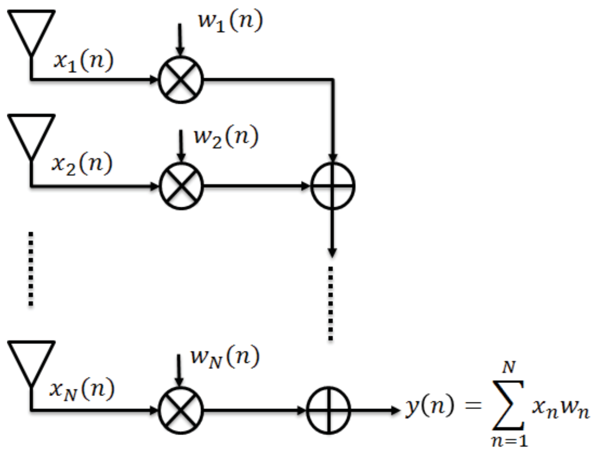


**Figure 1.5:** Antenna Array

and received signal is

$$AF = \sum_{n=1}^{N} w_n e^{\frac{-j2\pi(n-1)d\sin(\theta_{DOA})}{\lambda}} \tag{1.3}$$

Array antennas as stated before are required for DOA estimation and beamforming. The antenna array beam width decreases and beamforming resolution increases with number of antenna array elements. With increase in applications using higher frequencies and increasing popularity of patch antennas, the antenna size is decreasing. This has lead to usage of larger number of antenna arrays for mobile applications, which in turn increase the complexity of signal processing algorithms being used.

## 1.3 Direction of arrival (DOA) estimation methods

Various DOA estimation techniques have been proposed and implemented [9]. Here, we introduce only those that are important for the thesis namely : Delay and sum (DAS), Multiple signal classification (MUSIC) and root-MUSIC. Each of these methods will be introduced here and a comparison of these will be provided.

### 1.3.1 Delay and sum (DAS)

DAS is the simplest of all DOA estimation methods in terms of complexity and implementation. The derivation and signal model of DAS is briefed here [18]. It involves applying a set of complex weights $W = [w_1, \ldots, w_N]$ consistent with the array vector in (1.2) on $N$ incoming signals, summing them and measuring the output power. The weights delay the signal by changing its phase. Weight applied to the $i^{th}$ antenna corresponds to

$$w_i(\theta) = e^{j\phi_i(\theta)} \text{ where } \phi_i(\theta) = \frac{2\pi(i-1)d\sin(\theta)}{\lambda} \tag{1.4}$$

In (1.4) $\theta$ is the phase at which the weight is evaluated, $d$ is the separation between elements and $\lambda$ is wavelength as stated before. The input signal at each antenna $i$ is given as

$$r_i = Re[Ae^{j(\omega t+\theta_0+\phi_i(\theta_{DOA}))}] \tag{1.5}$$

where $A$ is the amplitude of received signal, $\omega = 2\pi f_0$ is the angular frequency for center frequency $f_0$ and $\phi_i(\theta_{DOA})$ is the relative phase delay of the signal due to $\theta_{DOA}$ which is given as

$$\phi_i(\theta_{DOA}) = \frac{-2\pi(i-1)d\sin(\theta_{DOA})}{\lambda} \tag{1.6}$$

When the weights are applied to received signal and summed

$$\begin{aligned} R(\theta,\theta_{DOA}) &= \sum_{i=1}^{N} r_i w_i(\theta) \\ &\sum_{i=1}^{N} Ae^{j(\omega t+\theta_0)} e^{j\phi_i(\theta_{DOA})} e^{j\phi_i(\theta)} \\ &\sum_{i=1}^{N} Ae^{j(\omega t+\theta_0)} e^{j\phi_i(\theta_{DOA})+\phi_i(\theta)} \end{aligned} \tag{1.7}$$

In (1.7) $r_i$ is the received signal as given in (1.5), $\phi_i(\theta)$ is given in (1.6), $\theta_{DOA}$ is the angle of DOA, $\theta_0$ is the phase introduced due to distance between transmitter and receiver and all other parameters are as defined previously.

The angle $\theta$ is varied from $\theta \in [-90^o, 90^o]$ and the received power is measured in steps of

$\Delta\theta$ at $\theta_l$ where $l = 1,\ldots,L$ and $L = 1 + \frac{180}{\Delta\theta}$. The received power corresponds to

$$P(\theta,\theta_{DOA}) = \frac{1}{A^2}|R(\theta_l,\theta_{DOA})|^2 \tag{1.8}$$

When $\theta_l = \theta_{DOA}$, power maximizes. Thus estimated DOA $\hat{\theta}_{DOA}$ is given as

$$\hat{\theta}_{DOA} = \text{ arg max} P(\theta,\theta_{DOA}) \tag{1.9}$$

DAS performs well only at high SNR conditions and is very sensitive to calibration and multipath. The resolution depends on $L$ (introduced in (1.8)), therefore increasing the computational cost for very high degree accuracy.

### 1.3.2 MUSIC

Given the condition that number of antennas are more than the number of sources and an estimate of number of sources is available, one can use a more complex DOA estimation technique known as multiple signal classification (MUSIC) [19]. For an array of $N$ sensors and $M$ sources $(M < N)$, the received sensor signal is

$$y(t) = V * x(t) + n(t) \tag{1.10}$$

where $V = [V(\theta_1),....,V(\theta_M)]$ is the steering matrix, $x(t) = [x_1(t),....,x_M(t)]^T$ is the transmitted signal and $n(t) = [n_1(t),....,n_N(t)]^T$ is the additive noise. The covariance matrix and its eigen decomposition is given as

$$R = E(xx^H) = E_s\Lambda_s E_s^H + E_n\Lambda_n E_n^H \tag{1.11}$$

$E_s$ and $E_n$ are the signal and noise subspaces respectively. $E(.)$ is the expectation operation.

$$\Lambda_s = diag(\lambda_1,....,\lambda_M) \text{ and } \Lambda_n = \sigma^2 I_{(N-M)\times(N-M)} \tag{1.12}$$

are the signal and noise eigenvalue vectors respectively, where $\sigma^2$ is the noise variance and $I$ represents an identity matrix.

The MUSIC spectrum corresponds to

$$S(\theta) = \frac{1}{V^H(\theta)AV(\theta)}, \qquad A = E_n E_n^* \tag{1.13}$$

$\theta$ is the angle at which the spectrum is evaluated and the range and resolution of the MUSIC depends on desired precision of DOA estimation. $V$ is the steering vector as given in (1.10) and $E_n$ is the noise subpaces as given in (1.12). Figure 1.6 depicts the MUSIC spectrum for a $\theta_{DOA} = 30^o$. The DOA is found by searching the spectrum for the peak. The peak corresponds to the $\theta_{DOA}$. Figure 1.7 represents the flow of operations done for MUSIC. MUSIC is known to offer good results at even low SNR and multipath conditions, but is

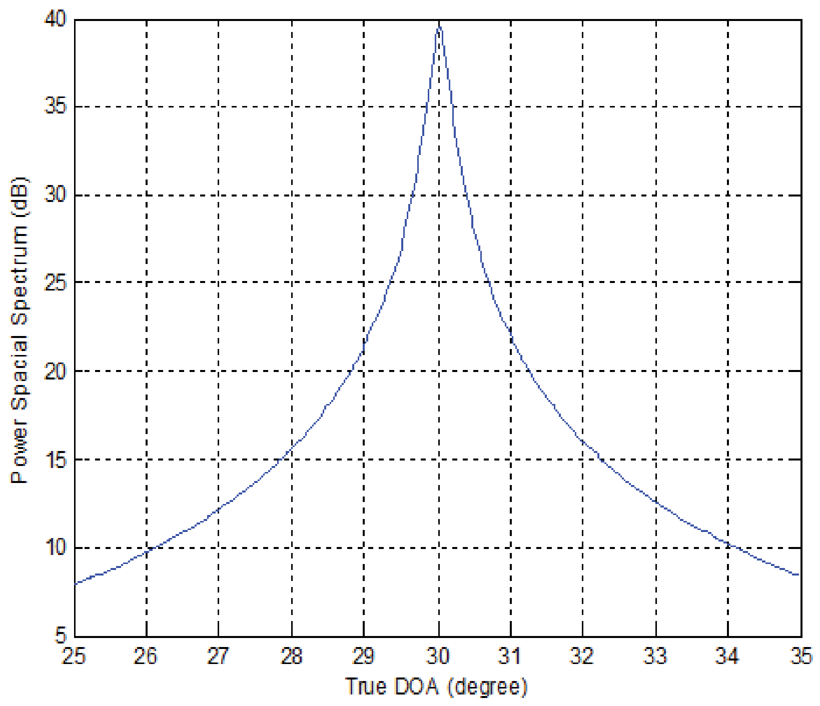


**Figure 1.6:** MUSIC Spectrum

highly sensitive to calibration. As the number of snapshots increases the performance of MUSIC asymptotically approaches the lower Cramer-Rao bouns (CRLB) [20]. Variations of MUSIC are available in huge numbers. [21, 22] describe few of these variations.

### 1.3.3 Root-MUSIC

Spectrum search step in MUSIC is an expensive operation. [23] proposes a new method for finding the DOA, applicable exclusively to uniform linear array (ULA). For a ULA, the

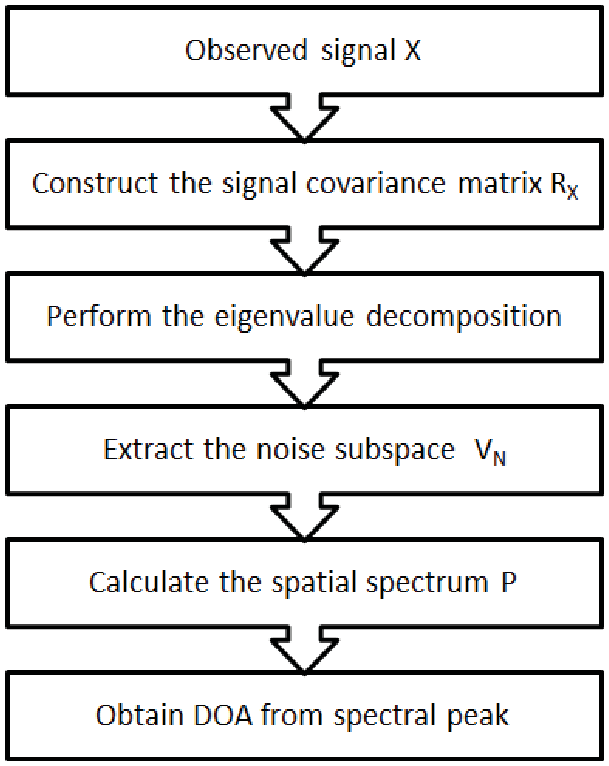

Observed signal X
Construct the signal covariance matrix $R_X$
Perform the eigenvalue decomposition
Extract the noise subspace $V_N$
Calculate the spatial spectrum P
Obtain DOA from spectral peak

**Figure 1.7:** MUSIC

$n^{th}$ element of steering vector, $V(\theta)$ corresponds to

$$V_n(\theta) = e^{-j2\pi n(\frac{d}{\lambda})\sin(\theta)}, \quad n = 1 \ldots N \tag{1.14}$$

where $d$ is the separation between the elements of antenna array and $\lambda$ is the wavelength of the signal. The inverse of MUSIC spectrum in (1.13) can be simplified using (1.14) as

$$\begin{aligned} S^{-1}(\theta) &= \sum_{n=1}^{N}\sum_{m=1}^{N} e^{-j2\pi m(\frac{d}{\lambda})\sin(\theta)} A_{mn} e^{j2\pi n(\frac{d}{\lambda})\sin(\theta)} \\ &= \sum_{l=-N+1}^{N-1} a_l e^{-j2\pi l(\frac{d}{\lambda})\sin(\theta)} \end{aligned} \tag{1.15}$$

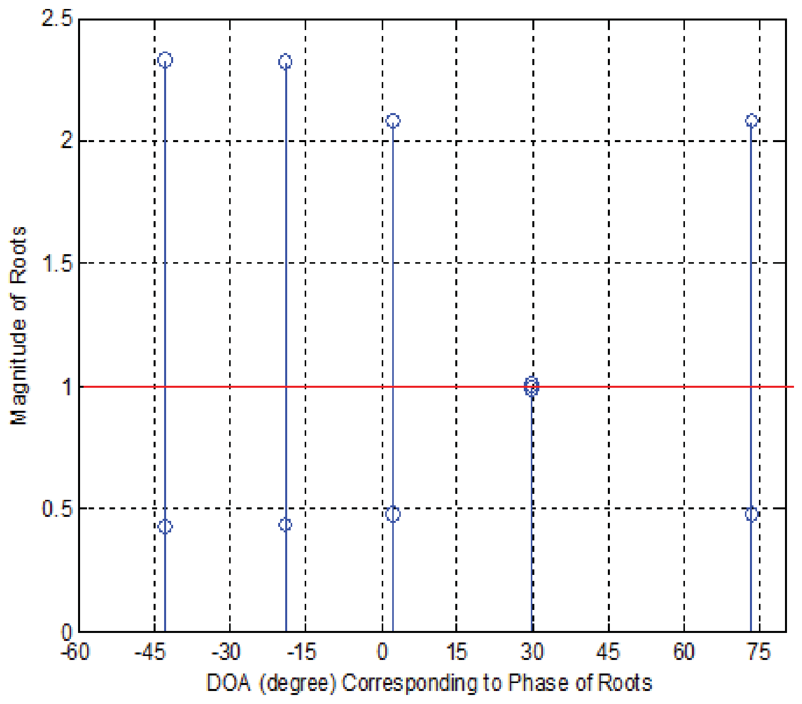


**Figure 1.8:** Roots of root-MUSIC polynomial

where $d$, $\lambda$ and $\theta$ are parameters as explained in (1.14) and (1.13) respectively. In addition, $a_l = \sum_{m-n=l} A_{mn}$ is the sum of entries of A along the $l^{th}$ diagonal. A polynomial can be constructed as

$$D(z) = \sum_{l=-N+1}^{N-1} a_l z^{-l} \tag{1.16}$$

where $a_l$ is as explained in (1.15). It is well known that roots on the unit circle of a polynomial $D(z)$ are used to extract the DOA of the signal. Figure 1.8 depicts the roots of $D(z)$ with $\theta_{DOA} = 30^o$. As it is observed, the root on unit circle is the one corresponding to $\theta_{DOA}$. Figure 1.9 depicts the flow of operations for obtaining DOA using root-MUSIC. It is

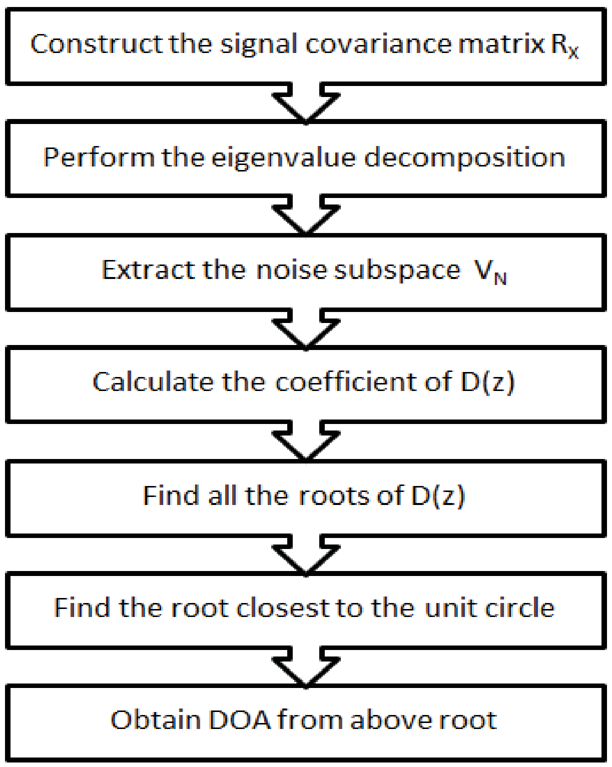

Construct the signal covariance matrix R<sub>x</sub>
Perform the eigenvalue decomposition
Extract the noise subspace V<sub>N</sub>
Calculate the coefficient of D(z)
Find all the roots of D(z)
Find the root closest to the unit circle
Obtain DOA from above root

**Figure 1.9:** Root-MUSIC

known that root-MUSIC performs better than MUSIC [24]. The limitation of root-MUSIC is the fact that it can be applied in its original form only for ULA. This issue has been addressed in various works such as [25]-[26].

Table 1.1 summarizes the advantages and shortcomings of each of the methods described and few other methods popularly know. As observed from this table, root-MUSIC is a very high resolution technique, however its complexity is high as well. High complexity may avoid real time implementation. Moreover, complexity leads to high power consumption. This thesis proposes algorithms that address the complexity issue of root-MUSIC.

**Table 1.1**
Comparison of DOA estimation techniques

| | Sensitivity to SNR | Sensitivity to calibration | Sensitivity to multipath | Resolution | Complexity |
|---|---|---|---|---|---|
| DAS[9] | High | Moderate | High | Low | Low |
| Maximum[27] Entropy | Moderate | Moderate | Moderate | Moderate | Moderate |
| MUSIC[19] | Low | High | Low | High | High |
| root-MUSIS[23] | Lower | High | Lower | Very high | High |
| ESPRIT[28] | Low | Low | Low | High | Very High |
| FUSION[16] | Moderate | Moderate | Moderate | Very High | Moderate |

## 1.4 Implementation of real-time signal processing algorithms

### 1.4.1 Real-time digital signal processing (DSP) system

A generic real-time DSP system can be represented as shown in Figure 1.10 [29]. The output of a sensor $x'(t)$ is amplified by the amplifier and the amplified signal $x(t)$ is passed through an anti-aliasing filter to limit the bandwidth of the signal, so that it satisfies the sampling theorem. The analog-to-digital converter (ADC) converts the analog signal $x(t)$ into digital signal $x(n)$, which is then ready to be processed by a DSP hardware. The reverse operations namely digital-to-analog (DAC), reconstruction filtering and amplification are applied once the DSP hardware produces the processed output signals. There are various choices for DSP hardware and the choice depends on various factors such as required

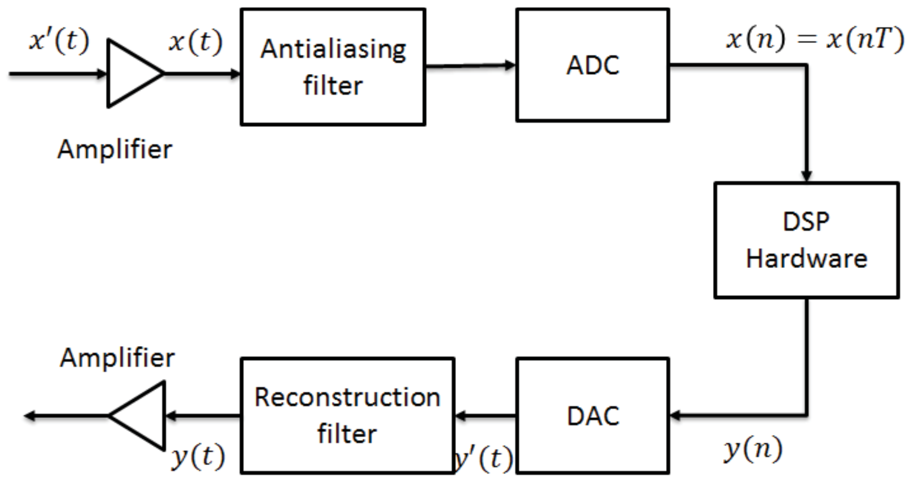


**Figure 1.10:** Real-time DSP system

**Table 1.2**
Comparison of various DSP hardwares

| | ASIC | FPGA | Microprocessors Microcontrollers | DSP Processors Processors | DSP processors with HW accelerators |
|---|---|---|---|---|---|
| Flexibility | None | Limited | High | High | Medium |
| Design time | Long | Medium | Short | Short | Short |
| Power Consumption | Low | Low -medium | Medium -high | Low -medium | Low -medium |
| Performance | High | High | Low- -medium | Medium- -high | High |
| Development cost | High | Medium | Low | Low | Low |
| Production cost | Low | Low -medium | Medium -high | Low -medium | Medium |

performance, cost, development time etc. Various DSP hardwares and their comparison is replicated here from [29] in Table 1.2 for the convenience of the reader.

### 1.4.2 Real-time constraints

Real-time computing is a concept applicable to any hardware or software system. A system is called "real-time" if it satisfies certain "real-time constraint". A real-time constraint implies that the system guarantees a response within a time frame. A real-time system can be classified as hard if missing a deadline implies total failure of the system and soft if missing a deadline merely causes the degradation of performance and not a complete failure.

A DSP system is said to be real-time if the signal processing time $t_p$ is less than the sampling period $T$. This implies that a processing task needs to be completed before a new sample comes in. Considering $t_{IO}$ as the overhead time for I/O operations then

$$t_p + t_{IO} < T \tag{1.17}$$

Therefore, the processing speed determines the maximum rate at which signal can be sampled. On the other hand, we can use faster DSP hardware to keep up with the desired sampling rate. Faster DSP hardware alone may not be enough to keep up with the sampling speed. Combination of simplified DSP algorithms, optimized system design or program and parallel processing needs to be adopted to achieve a given performance requirement. It is the duty of the system designer/architect to maintain a balance between cost and performance.

### 1.4.3 DSP hardware for WLPS

Field programmable gated arrays (FPGA) in the past were used as co-processor to digital signal processors in DSP systems. With recent improvements in FPGA capabilities, FPGAs are being considered as the main processor rather than a co-processor. We choose to implement the DSP algorithms of WLPS system on FPGA due to following reasons :

1. **Flexibility of design** : WLPS is still in its development stages and we expect various changes to the individual modules of the system. Usually, the DSP algorithms go through multiple revisions. Programmability of FPGAs enables revisions with minimal cost. Once the system design is finalized and the migration of design to application specific integrated circuits (ASICs) will be required, FPGA design can be adopted with little change.

2. **Parallel architecture** : Real-time constraints along with the requirement for high throughput transmission in WLPS necessitates highly parallel DSP algorithms. FPGA is a logical choice of hardware when planning to implement a highly parallel design. Other hardware such as digital signal processors and microprocessors have very little or no parallel processing elements that can be used for implementing our algorithms.

3. **Accommodating other functionalities**: WLPS hosts control logic apart from signal processing logic. Traditionally, the control task is delegated to a co-processor or external controller. But with large amount of processing elements available on FPGA, we can accommodate both control and DSP logic. Moreover, in critical applications such as security, surveillance, rescue etc, a considerable portion of processing elements needs to be dedicated to avoiding failures of system. Accommodating all the processing on single chip leads to compactness and faster inter-process communication, both of which are required in mobile applications where WLPS will be used. Moreover, FPGAs have wide variety of high speed I/Os which can be used for connecting to the RF frontend and other units on the system.

## 1.5 Thesis Contribution

Root-MUSIC was chosen as DOA estimation technique because it offers a better performance compared to MUSIC itself with reduced complexity. Figure 1.11 represents a conventional DOA system. Computationally intensive modules in a root-MUSIC system are subspace decomposition and polynomial rooting. These modules could possibly be the bottlenecks of the system in meeting real-time constraints. The subspace decomposition can be done either by performing Eigen value decomposition (EVD) of covariance matrix or Singular value decomposition (SVD) of the data matrix. Let

$$A = U\Sigma V^T \tag{1.18}$$

be the SVD of A . Here, $U$ and $V$ are the sigular vectors and $\Sigma$ is a diagonal matrix of singular values . Then

$$A^T A = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T \tag{1.19}$$

$\Sigma^T\Sigma$ is a diagonal matrix with $\sigma_i^2$ entries. Therefore, $\sigma_i$ are the eigenvalues of $A^T A$ and $V$ are the eigenvectors. Thus, we can safely say that either of the decompositions are the same. We choose to use SVD because matrix multiplication for obtaining covariance matrix can be avoided. Moreover, noise eigenvectors loose precision when matrix multiplication is accomplished in fixed point designs. The first contribution of this paper is the development of a SVD algorithm suitable for real-time signal processing. The second contribution is that we propose a highly parallel polynomial rooting scheme. These contributions together leas to a low cost, high performance system for DOA estimation.

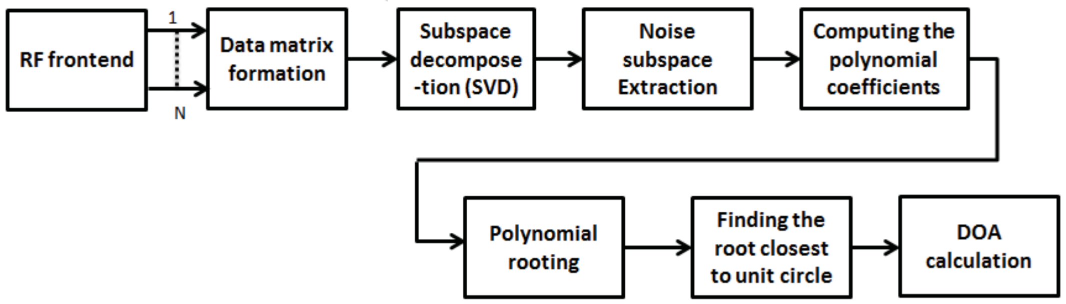

RF frontend
1
N
Data matrix formation
Subspace decompose -tion (SVD)
Noise subspace Extraction
Computing the polynomial coefficients
Polynomial rooting
Finding the root closest to unit circle
DOA calculation

**Figure 1.11:** Root-MUSIC system

### 1.5.1 Fast converging SVD for real-time signal processing and its FPGA implementation

Traditionally, SVD is sequential process as implemented in LAPACK and sequential machines [30]. Sequential algorithms have been driving designers away from implementing real-time DOA systems. When digital signal processors are the chosen DSP hardware, these sequential SVD algorithms can be used. For small matrix sizes, digital signal processors meet real-time constraints, but for larger matrix sizes meeting real-time constraints without any parallel processing is not possible. This has lead us to look into parallel algorithms for SVD.

Parallel algorithms for SVD have already been proposed in the literature [31]. These algorithms are suitable for implementation in platforms such as FPGA and ASICs, where there is enough parallelism to exploit. FPGA implementations of these SVD methods have been presented [32] in the past. All of these implementations haven't addressed one crucial issue of number of iterations it takes for the matrix to converge to correct singular values. Reducing the number of iterations to converge, can help the DSP system to meet the real-time constraints for larger matrices as well. A new approach for reducing the number of iterations for larger matrix size is proposed in this thesis. Although a large number of applications require SVD, there are no commercial off-the-shelf (COTS) intellectual property (IP) cores. Moreover, the research articles present SVD techniques which are difficult to replicate and often not well documented. This thesis provides a detailed description of SVD design with improved performance, which makes it easier for designers to replicate and make further changes. Modularity and extendibility can easily lead to development of IP cores. Although the proposed algorithm can be adopted for any DSP hardware, this thesis proposes path ways to exploit special functional units in FPGA to maximize performance.

### 1.5.2 Real-time root-MUSIC DOA estimation via a parallel polynomial rooting method

Polynomial rooting has been a subject of study for mathematicians for over centuries now. Finding all roots of a complex polynomial with low complexity, especially of degree greater than four is a subject of ongoing research. Many methods of varying orders of complexity have been proposed in the literature [33]-[34] but implementation of polynomial rooting on FPGA is rare if ever discussed. Polynomial rooting used in

LAPACK and implementations on sequential machines are based on companion matrix techniques [35]. Polynomial rooting using eigenvalue decomposition of companion matrix is inherently sequential and so are most of its variations. Implementation of these sequential algorithms on even high speed DSP hardwares cannot guarantee that the system is capable of meeting real-time constraints. Meeting the real-time constraints becomes increasingly difficult with increasing polynomial degree and for systems working at very high sampling rates. Few proposed polynomial rooting techniques [36, 37] are capable of parallelization, but these methods are either too complex to implement or are computationally intensive.

Motivated by these shortcoming in existing techniques and also propelled by findings of some unique geometry in Newton map of root-MUSIC polynomial, a new method of polynomial rooting specific to root-MUSIC was proposed in this thesis. There is a need for IP cores for generic polynomial rooting. Although the proposed method applies uniquely to root-MUSIC polynomial, it can be developed into an IP core because many existing and emerging applciaitons require DOA estimation for localization. Modularity, ease of implementation, extremely simple algorithm, extendability are main advantages of the proposed method.

## 1.6 Organization

This thesis is organized as follows. Chapter 2 describes the fast converging SVD method, its algorithm and a system level design. In addition, simulation results depicting the improvement in convergence are presented. Chapter 3 describes the implementation of the proposed SVD method on FPGAs. Details of each module is given, along with resource consumption and timing diagrams. The overall resource consumption and maximum achievable frequency of the SVD system are presented along with latency and throughput calculations. Chapter 4 presents the proposed polynomial rooting technique for root-MUSIC. The complex dynamics of the Newton map of the root-MUSIC is presented and findings regarding the unique characteristics of root-MUSIC polynomials are presented and proved. The proposed method is compared with existing polynomial rooting techniques. Finally, Chapter 5 concludes and lists possible directions for future work.

# Chapter 2

# Fast converging SVD for real-time signal processing and its FPGA implementation

This chapter introduces a novel fast converging Jacobi like SVD algorithm applicable to real-time signal processing of massive sensor arrays. The proposed algorithm highly increases the SVD convergence rate for larger matrices when compared to traditional Jacobi based methods. The parallel nature of the Jacobi methods is key to real time implementation intended for FPGAs. A highly modular system design which retains the inherent parallelism of the Jacobi based systolic arrays is proposed. The system was implemented for $4 \times 4$ and $8 \times 8$ matrix sizes on Virtex-6 FPGA. The proposed design was compared with the traditional design in terms of FPGA resource consumption, maximum achievable frequency and latency throughput tradeoff.

## 2.1 Introduction

Singular Value Decomposition (SVD) is an important component of many signal processing algorithms. Many applications such as image processing [38, 39, 40], channel estimation in multiple input multiple output -orthogonal frequency division multiple access (MIMO-OFDM) systems [41, 42, 43], biomedical applications [44] and direction-of-arrival (DOA) estimation for source Localization [18, 7] require real-time SVD. These applications demand fast convergence and high accuracy. For small sized arrays, it is easy to meet the above requirements given the high density of logic and high clock rates available on present day hardware. On the other hand in applications dealing with large matrices [45, 46, 47] achieving fast convergence is a difficult task. There is an increasing need for real time computation of SVD for large matrices because emerging applications using large arrays are proposed. Since there is an upper limit to the logic density and clocking rates in hardware, there is a need for fast converging SVD algorithms. Specifically, this work was motivated by the implementation of fast converging SVD algorithm for DOA estimation in new wireless local positioning system (WLPS) [7]. A fast converging SVD in WLPS would ultimately lead to lower power consumption in mobile applications. Moreover, it supports real time SVD computations for large matrices associated with large aperture antenna arrays. Configurable hardware platforms such as field programmable gated array (FPGA) or application specific integrated circuits (ASICs) are ideal for implementing WLPS. These platforms have huge number of logic at our disposition and are suitable for implementing parallel algorithms. As we will be implementing our parallel algorithms on these platforms, from hereon we refer to these as "parallel hardware".

SVD procedure prescribed by Golub-Kahan-Reinsch [48] is the standard method on sequential processors and is not suitable for parallel hardware. On the other hand, widely known Jacobi method [48] has inherent parallelism that has been exploited in many variations of this algorithm. This method can only be used for symmetric matrices and it has the advantage of quadratic convergence [49]. Moreover, it is proven to be more accurate than QR based methods [50]. Forsythe and Henrici [51] extended the Jacobi method to general matrix and a cyclic version of their proposed approach was later implemented by Brent-Luk-Van (BLV)on a Systolic Array[31]. BLV method uses a two-sided transformation and is proven to retain the quadratic convergence of Jacobi method [52]. Hestens proposed a one-sided variation of Jacobi's method but Hestens method does not directly produce singular vectors like the two-sided methods[53].

Special purpose CORDIC algorithm [54] and reduction in computations of the BLV array [55] have motivated implementations of SVD on Field Programmable Gated Arrays (FPGA). An improved SVD systolic array was proposed in [32] and is known to perform three times more efficient and faster than BLV. Although this method is faster than the original BLV, the convergence behaviour in terms of number of sweeps remains unchanged. For larger matrix sizes the number of sweeps in BLV is prohibitively large[56] and hence unsuitable for applications requiring real-time computation of SVD. Another FPGA based implementation was proposed in [57] which folds a 4x4 SVD problem to obtain the SVD of a larger matrix. Folding is a natural choice of implementation given limited hardware resources but this increases the computation time. We note that most efforts in improving SVD arrays are targeted towards decreasing the computation time of an iteration or reducing the hardware resources consumed. Since the above methods have already lead to maximum efficiency of a processing element, we delve into speeding up the convergence by decreasing the number of sweeps.

Unlike traditional methods which follow a fixed ordering of subproblems ,we propose a dynamic ordering where large element in each iteration is targeted. This highly increases the SVD convergence rate in larger matrices compared to traditional Jacobi based methods. The performance is compared in terms of number of sweeps. Monte Carlo simulations of various matrix sizes were carried and results are reported. The proposed method reduces the number of iterations by half for large matrix sizes. A fixed point streaming architecture was also proposed and implemented in Xilinx FPGA. Both the traditional and proposed methods are implemented and compared in terms of resources consumed, throughput and latency. The details of implementation are provied in future chapter.

Section 2.2 describes the traditional fixed order Jacobi method for reference. It is followed by the description of proposed algorithm. Section 2.3 outlines the architecture and provides details of hardware implementation. Section 2.4 describes the simulation result and compares the various aspects of implementation for the proposed and traditional algorithm. Section 2.5 concludes the chapter.

## 2.2 Proposed Method

In this section, first the traditional Jacobi based SVD method and the BLV array are introduced. Next, the proposed method and it's associated algorithm are introduced.

### 2.2.1 Traditional Method- Jacobi SVD algorithm and BLV Array

Jacobi methods use a sequence of plane rotations to diagonalize a matrix $A$. For SVD, two-sided plane rotations as shown below are used.

$$A_{i+1} = J_l^T A_i J_r \tag{2.1}$$

are used. Here $J_l$ and $J_r$ are Jacobi rotations of $\theta_l$ and $\theta_r$ in the $(p,q)$ plane $(p<q)$.A Jacobi matrix $J$ is an identity matrix where four elements with indices $(p,p)$, $(p,q)$, $(q,p)$

and $(q,q)$ are replaced by following values

$$J_{pp} = cos(\theta), J_{pq} = sin(\theta), J_{qp} = -sin(\theta), J_{qq} = cos(\theta) \tag{2.2}$$

are replaced with *cos* and *sin* of a rotation parameter $\theta$. $\theta$ is $\theta_l$ and $\theta_r$ for left and right sided rotations respectively. In each iteration $\theta_l$ and $\theta_r$ pairs are calculated to annihilate the off-diagonal elements of a $2 \times 2$ submatrix. The two sided rotation applied to each $2 \times 2$ submatrix corresponds to:

$$\begin{bmatrix} a'_{pp} & 0 \\ 0 & a'_{qq} \end{bmatrix} = \begin{bmatrix} c_l & s_l \\ -s_l & c_l \end{bmatrix}_p^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} \begin{bmatrix} c_r & s_r \\ -s_r & c_r \end{bmatrix}_q^T \tag{2.3}$$

$$\begin{bmatrix} a'_{ij} & a'_{i(j+1)} \\ a'_{(i+1)j} & a'_{(i+1)j} \end{bmatrix} = \begin{bmatrix} c_l & s_l \\ -s_l & c_l \end{bmatrix}_i^T \begin{bmatrix} a_{ij} & a_{i(j+1)} \\ a_{(i+1)j} & a_{(i+1)j} \end{bmatrix} \begin{bmatrix} c_r & s_r \\ -s_r & c_r \end{bmatrix}_j^T \tag{2.4}$$

where $c_l = cos(\theta_l)$, $c_r = cos(\theta_r)$, $s_l = sin(\theta_l)$, $s_r = sin(\theta_r)$ and $a'_{pp}$, $a'_{qq}$ are the diagonal elements obtained after applying the two sided digonalization process. It is observed that a left sided Jacobi rotation affects only the elements in columns $p$ and $q$ and right sided Jacobi rotation affects only elements in rows $p$ and $q$. This exposes inherit parallelism in Jacobi method and its evident that in a matrix of size $N$ ($N$ even) $N/2$ subproblems can be solved in parallel. Therefore an iteration consists of solving $N/2$ subproblems and it is a convention to call $N$ such iterations as a sweep. The iterations are carried on till the off-diagonal norm of the matrix $A$ given as

$$off(A) = sum_{i \neq j} A_{ij}^2, for A \in \mathbf{R} \tag{2.5}$$

is within a specified tolerance.

$$sweeps = \frac{\text{Number of iterations}}{\text{Matrix size}} \tag{2.6}$$

is within a specified tolerance.

Numerous parallel ordering schemes have been proposed in literature[31]. The basic building blocks of the BLV array consists of a Processing Element (PE) which is allocated to solve a $2 \times 2$ submatrix. There are functionally two types of PEs. A diagonal PE solves for $\theta_r$ and $\theta_l$ and applies the rotation specified in (2.3) to the diagonal submatrix. It also transmits the rotation parameters to the neighbouring processor. An off-diagonal PE applies left sided rotation $\theta_l$ to $2 \times 2$ submatrices on the rows $p$ and $q$ and right sided rotation $\theta_r$ to $2 \times 2$ submatrices on the columns $p$ and $q$. The array therefore consists of $N/2$ diagonal PEs and $(N/2)^2 - (N/2)$ off-diagonal PEs. Fig 2.1 depicts the structure of the array and communication between the processing elements. After each iteration, data is swapped between PEs consistent with the ordering scheme. The BLV array is step synchronised due to which PEs are ideal for almost about 66% of the time. Efficiency was improved by a factor of 3 by operating on data as and when it is available, rather than synchronizing the

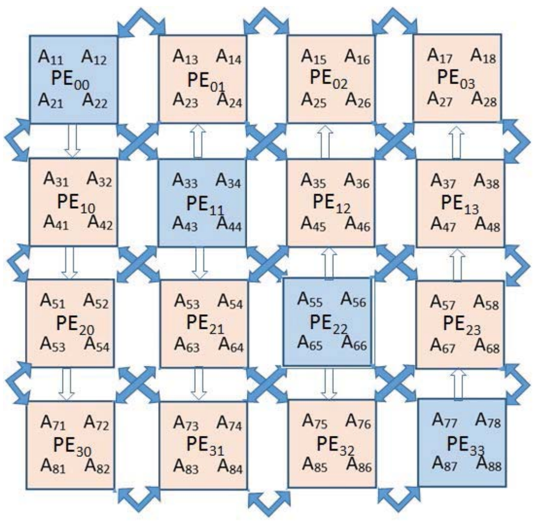


**Figure 2.1:** BLV Array for N=8. Data transmission is represented as solid arrows and rotation parameters transmission with unfilled arrows

steps [32]. Although this method increase the efficiency compared to the BLV array, the number of iterations still remain unchanged. We aim to address this issue and reduce the number of iterations required.

### 2.2.2 Proposed algorithm for faster convergence

The fixed ordering scheme used in BLV array has the advantages of simple design and low resource consumption, but for larger matrix sizes it take formidable number of iterations to converge. In this new algorithm, we address this issue of slow convergence by deviating from the traditional fixed ordering. We propose that in each iteration big off diagonal elements to be targeted. That is instead of forming a $2 \times 2$ submatrix using block-diagonal elements, the submatrix is formed by big off-diagonal elements. Since $N/2$ subproblems are solved in parallel, we can target $N/2$ big elements which satisfy the row column exclusivity. Row column exclusivity refers to the fact that $N/2$ big elements cannot have same row or column numbers as any of the other $N/2-1$ elements row and column numbers. Using this kind of dynamic ordering, we guarantee that in each iteration, the biggest element and few other large elements are targeted. Unlike fixed order schemes, this guarantees that the big off-diagonal elements are annihilated within the first few iterations and remaining iterations are allocated to annihilating the already small elements to the desired accuracy. The proposed algorithms is as follows.

```
repeat
  1. Find $N/2$ big elements from off diagonal entries.
       $(x_i, y_i)$ for $i = 1, \dots, N/2$
               where $x_{i+1} \neq x_{1,\dots,i}$, $x_{i+1} \neq y_{1,\dots,i}$,
                     $y_{i+1} \neq x_{1,\dots,i}$, $y_{i+1} \neq y_{1,\dots,i}$
               and $x_i \neq y_i$
  2.
       for $i = 1, \dots, N/2$ do
         2.1.Select $2 \times 2$ submatrix $A_{diag}$ to be processed by diagonal PE
         $A_{diag} = \begin{bmatrix} A_{x_i x_i} & A_{x_i y_i} \\ A_{y_i x_i} & A_{y_i y_i} \end{bmatrix}$
         2.2.Diagonal Processor
         → Solve the $2 \times 2$ SVD subproblem to obtain $\theta_r$ and $\theta_l$
         → Calculate and output left and right rotation parameters
→ Apply two sided Jacobi rotation on $2 \times 2$ submatrix

            $A'_{diag} = J'_l A_{diag} J_r$

→ Output data and wait for new input data
```

2.3.Column Processors

```
for $i = 1, \ldots, (N/2-1)$ do

    → Select$2 \times 2$ submatrix

    if $x_i < y_i$
    $A_{col}= \begin{bmatrix} A_{c_{2j-1}x_i} & A_{c_{2j}y_i} \\ A_{c_{2j}x_i} & A_{c_{2i}y_i} \end{bmatrix}$
    else
    $A_{col}= \begin{bmatrix} A_{c_{2j-1}y_i} & A_{c_{2j}x_i} \\ A_{c_{2j}y_i} & A_{c_{2i}x_i} \end{bmatrix}$
    where $c_k \neq x_i, y_i fork = 1, \ldots, (N-2)$

    →Apply right-sided rotation to the column submatrix

            $A'_{col} = A_{col} J_r$

    →Output data and wait for new input data

    →Replace the submatrix back into the matrix

end for
```

2.4.Row Processors

```
for $i = 1, \ldots, (N/2-1)$ do

    →Select$2 \times 2$ submatrix

    if $x_i < y_i$
```

```
                $A_{row} = \begin{bmatrix} A_{x_i r_{2j-1}} & A_{x_i r_{2j}} \\ A_{y_i r_{2j-1}} & A_{y_i r_{2j}} \end{bmatrix}$
                else
                $A_{row} = \begin{bmatrix} A_{y_i r_{2j-1}} & A_{y_i r_{2j}} \\ A_{x_i r_{2j-1}} & A_{x_i r_{2j}} \end{bmatrix}$
                where $r_k \neq x_i, y_i for k = 1, \ldots, (N-2)$

                →Apply left-sided rotation to column submatrix

                        $A'_{row} = J'_l A_{row}$

                →Output data and wait for new input data

                →Replace the submatrix back into the matrix

            **end for**

        **end for**

**until** off diagonal norm ,$off(A) < \xi(tolerance)$
```

## 2.3 Design and Implementation

### 2.3.1 Proposed System Design

#### 2.3.1.1 Controller and Big element finder

Similar to other Jacobi methods, the proposed method is suitable for parallel implementation. With an increase in the available processing units on parallel hardware, resources are hardly a limitation. Accordingly, we propose a highly parallel system shown in Figure 2.2. The solid arrows depict the communication of rotation parameters and plane arrows depict the communication of matrix elements. Because the proposed method is iterative, the stopping criterion is calculated and further iterations are started if a predefined tolerance is not met. This functionality is included in the controller. Controller also transfers data between the output and input memory banks if the iterations are to be continued. The proposed design is also intended to be pipelined to increase the throughput and controller needs to be designed accordingly. The big element finder module finds $N/2$ big elements as specified in the step 1 of the algorithm. This module produces $N/2$ pairs of row and column to be transmitted to the $N/2$ core modules.

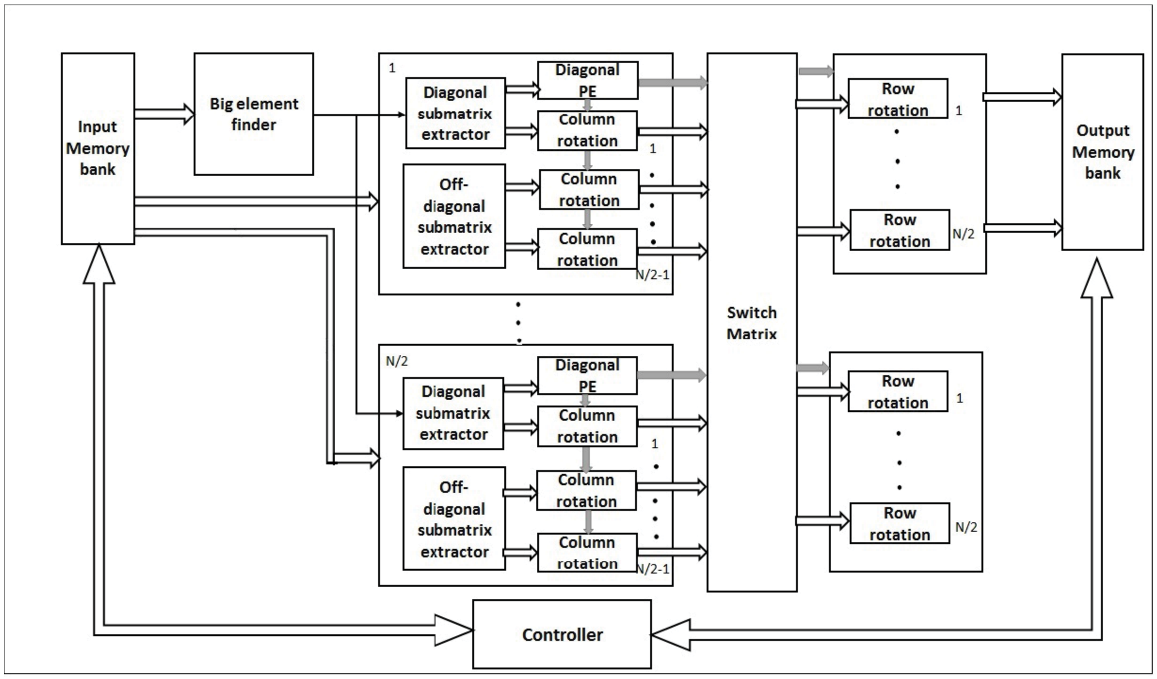


**Figure 2.2:** System level design for the proposed algorithm

#### 2.3.1.2 Core modules

The term core module refers to a set of modules which consists of a diagonal submatrix extractor, an off-diagonal submatrix extractor, a diagonal PE and $N/2$ pairs of column and row rotations. The diagonal submatrix extractor extracts a submatrix as described in step 2.1 of the algorithm. Off-diagonal submatrix extractor extracts $N/2-1$ submatrices as described in step 2.3 and 2.4 of the algorithm. These modules are necessitated due to dynamic ordering followed in the prescribed algorithm. The diagonal PE calculates the rotation parameters and transmits them to the column and row rotation modules. Column and row rotation applies the right and left sided rotations to the selected submatrices. A switch matrix is needed between column and row rotation modules to pass the appropriate matrices to the row rotation modules. It should be noted that a set of diagonal PE, column and row rotations are called a diagonal PE and a pair of column and row rotations are called off-diagonal PE in the BLV array. Hence, the number of parameter generators ($N/2$) and two sided rotation modules($(N/2)^2$) remain unchanged for the proposed approach and BLV array. The proposed high level design can be adopted for any parallel hardware.

### 2.3.2 Implementation on FPGA

As a proof of design and performance analysis, we chose to implement the proposed algorithm on Virtex-6 XCVLX365T using high-level design tool called System Generator for DSP by Xilinx. As the design was inherently parallel, we tried to further improve the system by pipelining to the maximum possible limit. Both the BLV array and the proposed algorithm were implemented to ensure a fair comparison. The modularity of design and implementation makes it easy for extending the system to bigger matrix sizes and word length if desired. BLV array consists of modules: diagonal PE, column and row rotation and data switching modules. The proposed array has additional modules: big element finder, submatrix extractor and switch matrix. Submatrix extractor and switch matrix are simple multiplexer based implementations with fine grain pipelining .

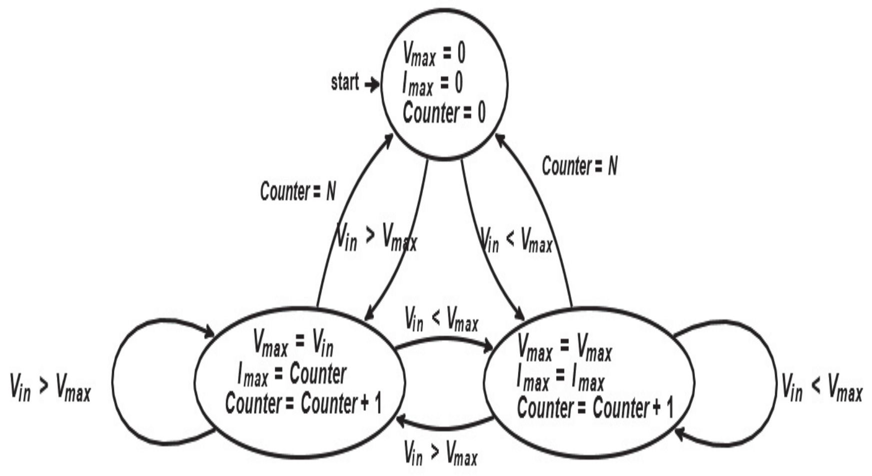


**Figure 2.3:** Finite State Machine for finding the biggest element in each row in a matrix of size N

### 2.3.2.1 Big element finder

This module finds the $N/2$ big elements which are row and column exclusive. It was assumed that the data of each row of matrix is received in streaming fashion. State machine in Figure 2.3 is used to find the biggest element in each row of a matrix of size $N$. The largest of these $N$ biggest elements gives the row and column number of the biggest element. The next biggest element is found among rows and columns excluding the rows and columns from which the previous biggest elements were found.

### 2.3.2.2 Diagonal PE

The Diagonal PE calculates the rotation parameters via Two-Plane-Rotation (TPR) method[55]. The inverse tan, sin and cos functions are implemented using CORDIC 4.0

macro provided by System Generator. Since hardware resources is not a constraint for smaller matrix sizes we choose not to fold the operations. For bigger matrix sizes the operations can be folded to save hardware resources. Each Diagonal PE uses 4 CORDIC modules.

#### 2.3.2.3 Column and Row rotation

Column and row rotations are each a $2 \times 2$ matrix multiplication, which translates to 8 multiplications and 4 additions. These are implemented using DSP48Es available in abundance in recent FPGAs. Multiplier design based on DSP slices are reported to consume lesser power [58]. For larger size matrices, multiplication operation can be folded into a single DSP48E by trading latency for higher frequency of operation.

## 2.4 Simulations and Discussions

### 2.4.1 Simulation Results

The performance of the proposed algorithm was compared with the BLV in terms of number of sweeps required to reduce the off diagonal norm (sum of the square of all the off diagonal elements) to a specified tolerance to a specified tolerance (in our simulations it is $10^{-15}$). It is conventional to measure convergence behaviour in terms of sweeps, where the sweeps represent number of iterations divide by $N$, for a matrix of size $N \times N$. The simulations were conducted for matrix sizes ranging from $4 \times 4$ to $128 \times 128$. Monte-Carlo simulations with uniformly distributed matrix elements were conducted. Figure 2.4 depicts the number of sweeps for various matrix sizes for the BLV array and the proposed array. It was found that the number of sweeps for proposed array plateaus for large matrix size. This means that for matrix sizes greater than 32 the number of iterations is nearly proportional to the matrix size. The factor by which the number of sweeps reduces increases with matrix size, hence the advantage of the method is highlighted for larger matrices.

In order to explain the observed effect we plot the off diagonal norm vs the iteration number in Figure 2.5. For $N = 128$, it is observed that the proposed algorithm's off diagonal norm reduces to set tolerance in a lower number of iterations as compared to the BLV array. The convergence of the proposed array for $N = 128$ is similar to that of BLV array for size $N = 64$. This difference in convergence behaviour between the proposed and BLV array is not very evident for $N = 8$.Note that the tolerance used for simulations($10^{-15}$) is an overkill for envisioned applications. Targeting the big elements in the initial iterations accelerates the annihilation of the off diagonal elements, resulting in the improved performance of the proposed method. This can be observed in Figure 2.5 where the off diagonal norm drops rapidly in the initial iterations for proposed method as against the BLV method.

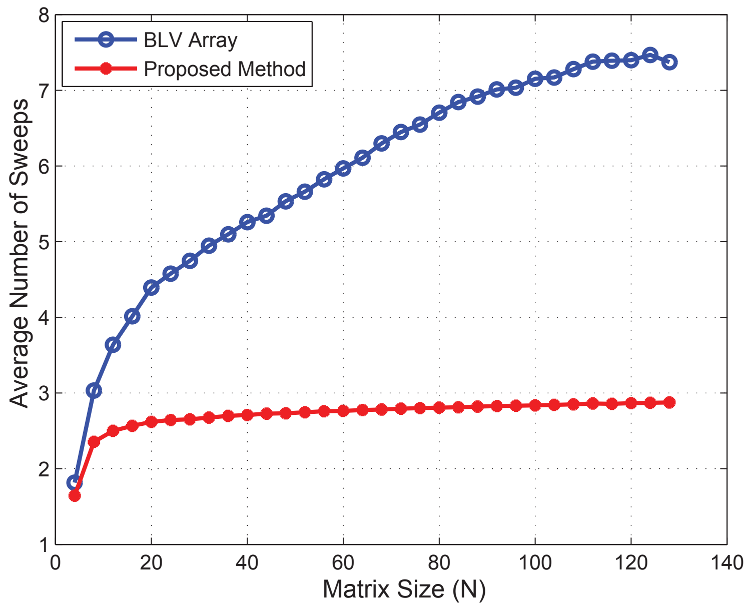


**Figure 2.4:** Number of sweeps vs. matrix size

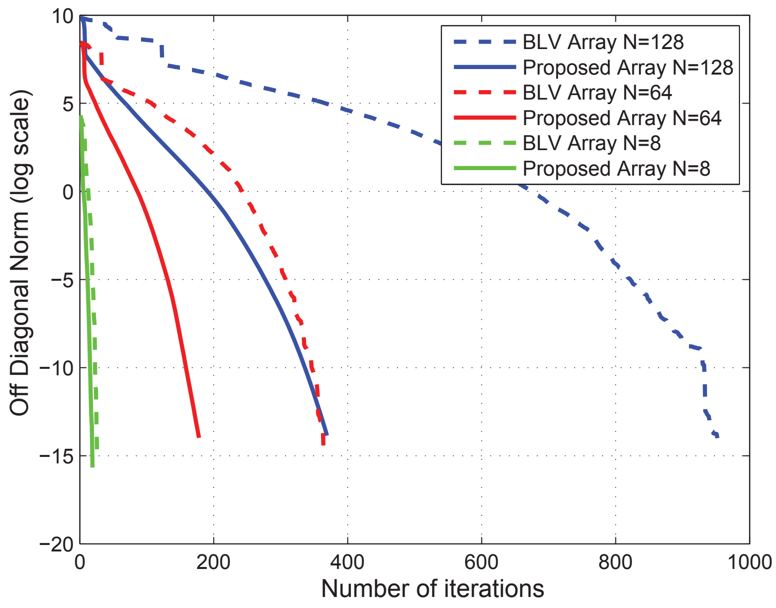


**Figure 2.5:** Convergence of matrix depicted as reduction in off diagonal norm vs iteration number

### 2.4.2 Implementation Results

For the purpose of fair comparison, FPGA implementation of the proposed and traditional BLV was done. Both the systems were designed with maximum possible pipelining resulting in comparable maximum achievable frequency of operation. Table 2.1 and 2.2 compare the resource consumption and maximum achievable frequency of the $4 \times 4$ and $8 \times 8$ proposed array with BLV arrays of equivalent size. Efforts were made to pipeline both the arrays to maximum possible extent to achieve the maximum frequency of operation. Given the iterative nature of the algorithm, similar frequency of operation with reduced number of sweeps amounts to faster overall convergence of the method. Although, we observe a marginal increase in resource consumption for the proposed design, the total number of slices consumed is less than 25% leaving area for other signal processing modules in the original applications. The DSP48Es have been pipelined in three stages which is the minimum for multiplier-based design. For bigger arrays, when the available number of DSP8Es fall short, we can fold the multiplication operation and pipeline the DSP slices up-to six stages to operate the DSP slices upto a maximum of 600MHz [58].

**Table 2.1**
FPGA Implementation of BLV and Proposed array for $4 \times 4$ matrix

| | Available on Virtex 6 | BLV Array ($4 \times 4$) | Proposed Array ($4 \times 4$) |
|---|---|---|---|
| Maximum Achievable Frequency | | 180.408 MHz | 180.018 MHz |
| Slice Registers | 455,040 | 12,386 (2%) | 14,208(3%) |
| Slice LUTs | 227,520 | 10,987 (4%) | 12,641(5%) |
| Occupied Slice | 56,880 | 3569 (6%) | 4094(7%) |
| DSP48E1s | 576 | 64 (11%) | 64(11%) |

**Table 2.2**
FPGA Implementation of BLV and Proposed array for $8 \times 8$ matrix

| | BLV Array($8 \times 8$) | Proposed Array($8 \times 8$) |
|---|---|---|
| Maximum Achievable Frequency | 107.001 MHz | 106.633 MHz |
| Slice Registers | 45,343(10%) | 51,254(11%) |
| Slice LUTs | 42,322 (18%) | 47,749(20%) |
| Occupied Slice | 14,300 (25%) | 16,371(28%) |
| DSP48E1s | 160 (27%) | 160(27%) |

### 2.4.3 Latency and Throughput

Latency and throughput are two key parameters for choosing a parallel and pipelined design for an application. Figure 2.6 sketches the timing of our design to investigate the throughput and latency tradeoffs of our system. This diagram does not depict the fine level pipelining and is being used to quantify the latency and throughput in general. Only modules which contribute to the latency (the ones whose latency cannot be hidden in other modules) are depicted. The throughput is equal to the inverse of the latency of the processing element with the highest latency, also called the critical element. In this case, part of diagonal processor which calculates the rotations parameters (referred to as parameter generator) is the critical element and is common to both proposed and BLV array. Hence, throughput for the proposed and BLV array is equivalent and corresponds to

$$T_{proposed} = T_{BLV} = \frac{1}{L(S_c)} = \frac{1}{T_{pg}} \tag{2.7}$$

. Here $L(.)$ is the latency of any given processing element and $T_{pg}$ is the latency of parameter generator. Latency can be calculated in way described via [59],

$$L = (2 \times i_c - 1) \times L(S_c) + \sum_{i=i_c+1}^{M} L(S_i) \tag{2.8}$$

where $i_c$ is the index number of the critical stage $S_c$, $S_i$ is $i^{th}$ stage of the pipeline and $M$ is the total number of pipeline stages. For our design, we consider the parameter generator as the critical stage. In this case, the number of stages prior to parameter generator is $N/2$ and accordingly index number if the critical stage is

$$i_c = \frac{N}{2} + 1 \tag{2.9}$$

and latency is given as

$$L_{proposed} = (2 \times (\frac{N}{2} + 1) - 1) \times T_{pg} + T_{cm} + T_{sm} + T_{rm} \tag{2.10}$$

where $T_{cm}$, $T_{sm}$ and $T_{rm}$ are latencies of column multiplication, switch matrix and row multiplication, respectively.$T_{pg}$ was introduced in (2.7). Except the processing element, all other modules have an idle time. This can be avoided by either folding the operations for saving area or by balancing the pipeline in a better way. Also, the critical element experiences an idle time before starting the next iteration on data set 1. This can be avoided by trying to keep the overall latency as an integral multiple of the latency of critical stage .

$$L = I \times T_{pg} \tag{2.11}$$

Where $L$ in the overall latency and integer $I$ is the number of data sets that can be processed before the next iteration on first data set starts. $T_{pg}$ has been introduced in (2.7). It is to be noted that for larger $N$ ,latency can be very large and we might have to go for different scheme of finding the large elements, where we may not be able to target best $N/2$ large elements always. Not being able to target the best large elements might impact the convergence minimally but the convergence behaviour will be still better than BLV. Latency and throughput tradeoff is a decision to be made based on the requirements of an application.

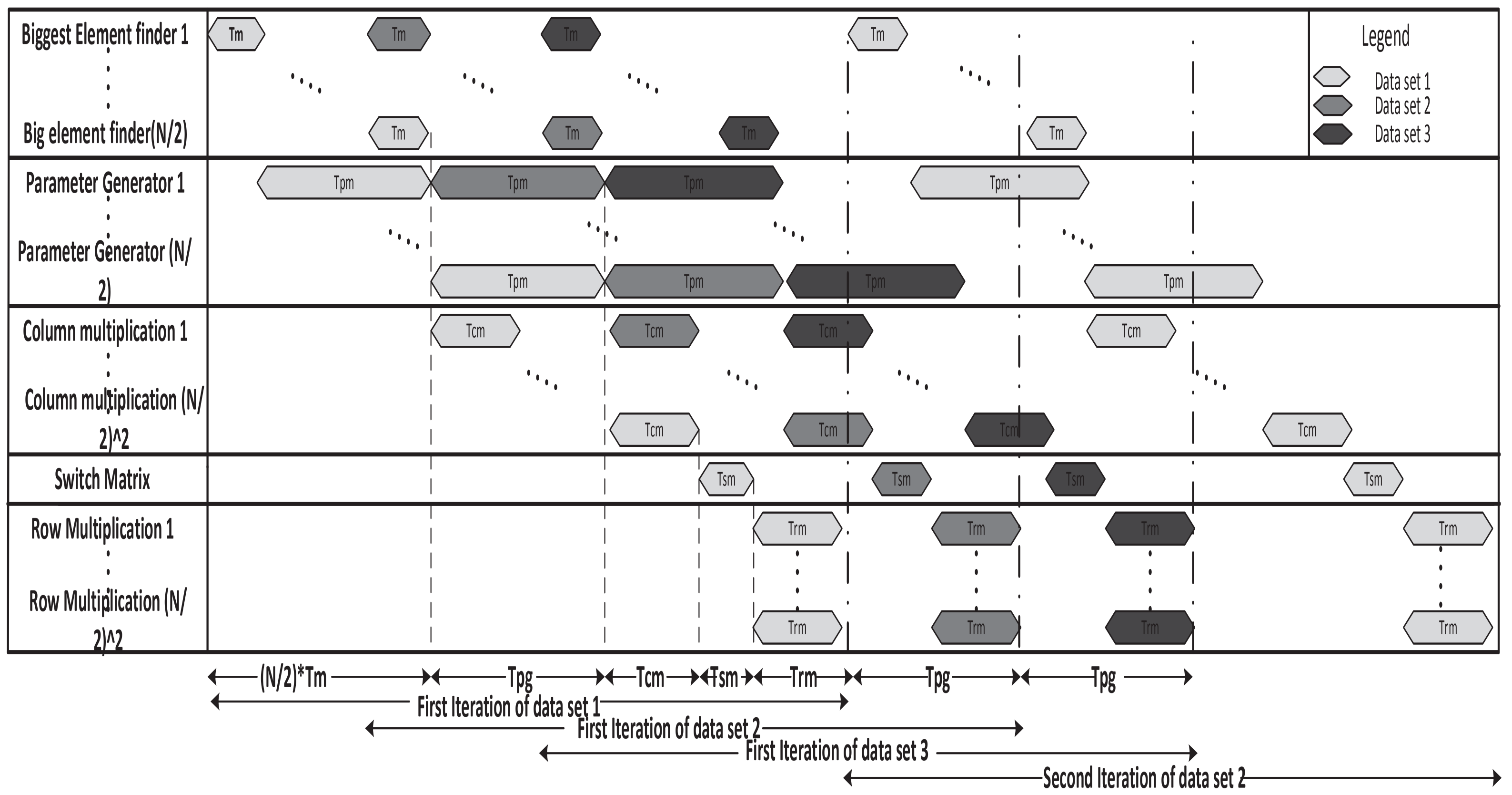


**Figure 2.6:** Timing diagram for the proposed SVD method

## 2.5 Conclusion

A new and fast converging algorithm for SVD suitable for real-time signal processing applications was proposed. A highly parallel and pipelined design for implementing this algorithm was also proposed. The higher convergence rate of the proposed technique was depicted by the reduction of the number of sweeps as the matrix size increases. Thus, the number of iterations needed by the proposed technique is significantly reduced (by about 50%) compared to the traditional BLV as the matrix size increases. The FPGA implementation proved that the system has the same throughput as the traditional system. Reduced number of sweeps with the same throughput implies faster convergence. Only a marginal increase in hardware resources was observed. The system can be clocked at frequency as high as 180 MHz for $4 \times 4$ matrix while occupying less than 10% of slices in Virtex-6 FPGA. We also noted that better convergence and comparable throughput is obtained as a tradeoff of latency. Hence, this proposed method is suitable for signal processing applications which operate on streaming data rather than on burst mode data.

# Chapter 3

# Details of FPGA implementation for proposed SVD

This chapter details the various aspects of FPGA implementation of SVD method proposed in previous chapter. The modules are explained in greater detail with help of finite state machine (FSM) diagrams and implementation diagrams. The overall system performance and comparative analysis has already been discussed in chapter 2, hence we limit the discussions in this chapter to implementation details of individual modules. The discussions in this chapter facilitates easy replication of the FPGA implementation. The modules are designed to eases the process of extending the proposed design to larger matrix sizes in the event of increase in number of antenna array elements. The design was pipelined to maximum possible extent to maximize the frequency of operation. As

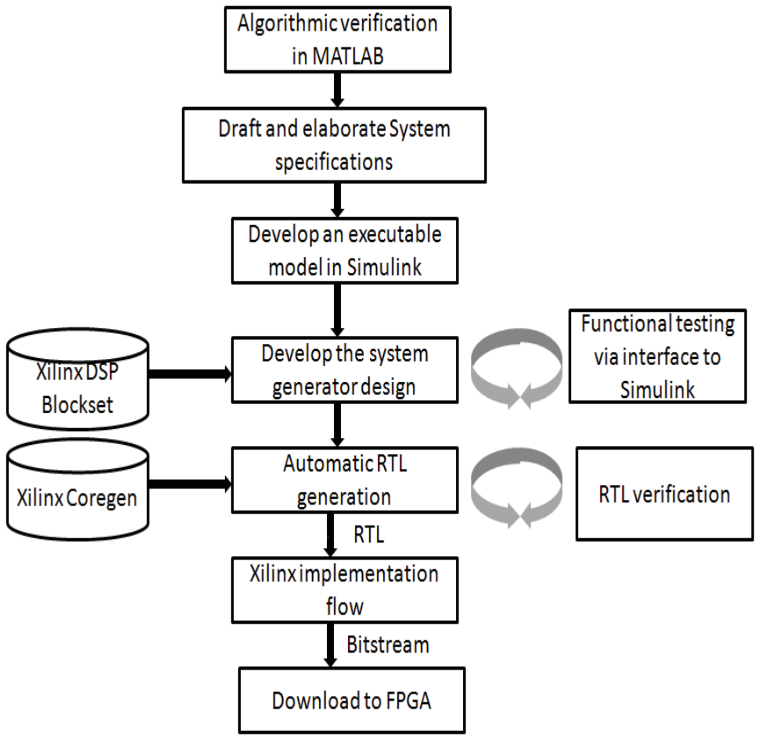

Algorithmic verification in MATLAB
Draft and elaborate System specifications
Develop an executable model in Simulink
Xilinx DSP Blockset
Develop the system generator design
Functional testing via interface to Simulink
Xilinx Coregen
Automatic RTL generation
RTL verification
RTL
Xilinx implementation flow
Bitstream
Download to FPGA

**Figure 3.1:** Design flow for System Generator development

mentioned before, the design was implemented on Virtex-6 XCVLX365T using high-level design tool called System Generator for DSP by Xilinx.

## 3.1 System Generator design flow

Figure 3.1 gives an overview of the various steps involved in System Generator design flow [60, 61]. A robust design needs to go through all of these steps and many times through multiple iterations of these. First step is to verify and analyze the proposed algorithms in MATLAB. A well written and detailed system specification document is a good starting point for FPGA design. Better the understanding of the various aspects of system design and features, lesser the number of iterations it takes through the design flow to arrive at the final implementation. Both the High level description (HLD) and low level description (LLD) need to be described in sufficient detail. Use of FSM diagrams, flowcharts and expected timing diagrams is encouraged. Since the proposed design is a fixed point design, world length and effects of word length need too be calculated and be specified as part of specifications document. Next, an executable Simulink model needs to be developed. This provides us with the understanding of timing of the system. This step is optional but is highly encourages as it may ease the process of System Generator design and come on handy when debugging and functional testing. Xilinx provides DSP blockset and other reference blockset for System Generator design. These blocksets need to be used for developing a System Generator design. It is important to follow the system specifications. System Generator can be easily interfaced to Simulink. While developing the individual modules and also while integrating different modules, one can use the Simulink interface to perform functional testing. Waveform viewer tool in System Generator is a convenient way to check the latencies. Often, functional verification and going through the iterations of design and testing is the most time consuming part of the design flow. System Generator uses the Xilinx Coregen functionality to automatically generate the RTL. Once the RTL is generated, RTL test bench needs to be created and RTL verification needs to take place. From here on, the steps are similar to a regular FPGA design flow of synthesis, translate, map, place and route and programming the FPGA with bitstream.

## 3.2 FPGA Modules

Various FPGA modules are described in details before the overall FPGA design is presented. A subset of these modules were used for implementing the BLV array as well. Throughout this discussion the notation of ($worlength\_fractional$) bits is used to represent the fixed point notation. For example, (16_14) implies the signal is of 16 bit wordlength with 14 fractional bits. Also the term "streaming matrix" implies that each row of the matrix is received in streaming fashion. Hence, we have $N$ channels corresponding to each row of a $N \times N$ matrix. Each element of the column is received in sequential manner. Therefore it takes $N$ clock periods to receive the entire matrix over $N$ channels. This signal flow is assumed to improve latency and also to keep the number of I/O pins to an optimum number. In case of very large $N$, the channels might need to be time multiplexed as we may not have enough I/O pins.The latency calculations were already discussed in sufficient detail in Section2.4.3.

### 3.2.1 Finding the $N/2$ big elements

This module finds the $N/2$ big elements which are row and column exclusive. Figure 3.2 represents the state machine for this module. First the diagonal elements are zeroed out as the big elements need to be from the off-diagonal. Zeroing out the diagonal elements is a simple task achieved with $N$ of these state machines depicted in Figure 3.3. The index $i$ in each of these state machines represents the channel number. For example $2^{nd}$ state machine with index $i = 2$ will be repeatedly pounding the $2^{nd}$ element of the streaming row data. In Figure 3.2 this operation is represented as $zerodiag()$. Next, the biggest element of the matrix is found as depicted in Figure 3.4. While the data (matrix entries) is streaming, the biggest element of each row can be obtained at the end of $N$ clock periods using the state machine depicted in Figure 3.5. Again, $N$ of these state machine are required to find the biggest element from $N$ channels. The $N$ biggest elements are then searched for biggest element. This can be done using a combinatorial circuit. We implement a $log_2(N)$ stage search for finding the biggest element out of $N$ elements. This can be done using $N-1$ comparators. The comparators should also output the index of the biggest element. After the biggest element is obtained, the entries corresponding to row and column of the biggest element are zeroed out using the state machine in Figure 3.6. Finding the biggest element of a streaming matrix and then zeroing out the row and column corresponding to the biggest element is referred to as $findmax()$ in Figure 3.2. And the biggest element of the resultant streaming matrix (after zeroing out the row and column entries to maintain row and column exclusivity) is then found by Figure 3.4. This operation is represented as $zerorowcol()$. This step is repeated till $N/2$ such big elements are found. We need not zero out the row and column entries after finding the $N/2^{nd}$ big element.

Figure 3.7 is the System Generator module for finding 4 big elements of a $8 \times 8$ streaming matrix.

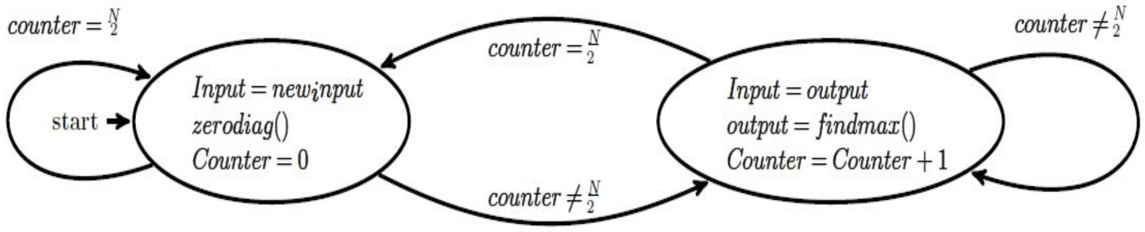


**Figure 3.2:** State machine for finding the $N/2$ big entries of $N \times N$ streaming matrix under row column exclusivity condition

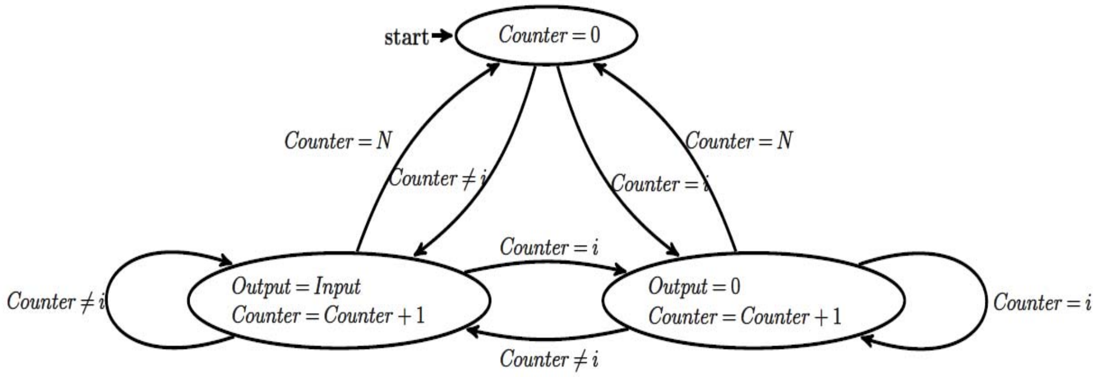


**Figure 3.3:** State machine for zeroing out the diagonal entries of a $N \times N$ streaming matrix

Figure 3.7 is the System Generator module for finding 4 big elements of a $8 \times 8$ streaming matrix. From the description above it is understood that first big element is found at the end of $N + log_2(N)$ clock cycles from the time of first column streams in. The indices of second element at $log_2(N) + L(zerorowcol)$ ,where $L(.)$ is the latency of function. The indices of subsequent big elements is obtained at subsequent $log_2(N) + L(zerorowcol)$ clock cycles. Figure 3.8 depicts the timing diagram for the modules in Figure 3.7. The clock cycles at which the first, second, third and fourth big element indices are found

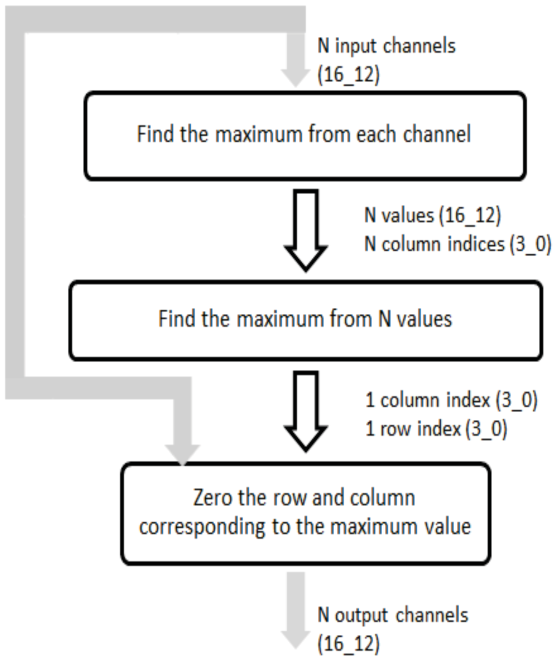

N input channels (16_12)
Find the maximum from each channel
N values (16_12)
N column indices (3_0)
Find the maximum from N values
1 column index (3_0)
1 row index (3_0)
Zero the row and column corresponding to the maximum value
N output channels (16_12)

**Figure 3.4:** Flow chart for finding the big element and its row column indices

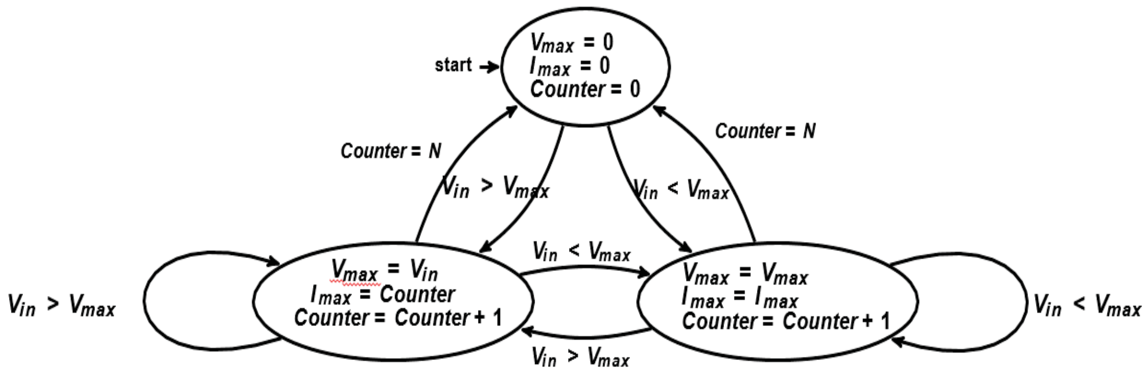


**Figure 3.5:** State machine for finding the biggest element from the channel of a streaming matrix

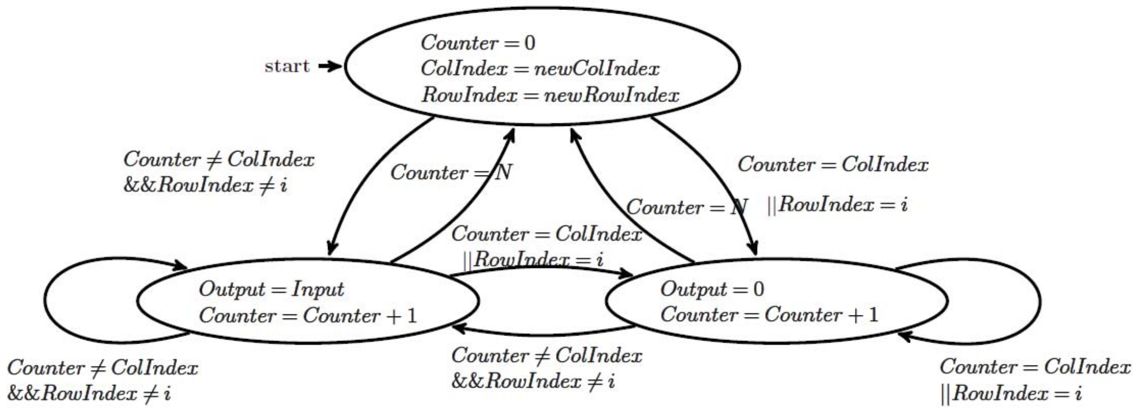


**Figure 3.6:** State machine for zeroing out the row and column of a $N \times N$ streaming matrix corresponding to the big element

include the latencies mentioned above and latencies due to pipelining. The figure serves the purpose of understanding the latencies of the module.

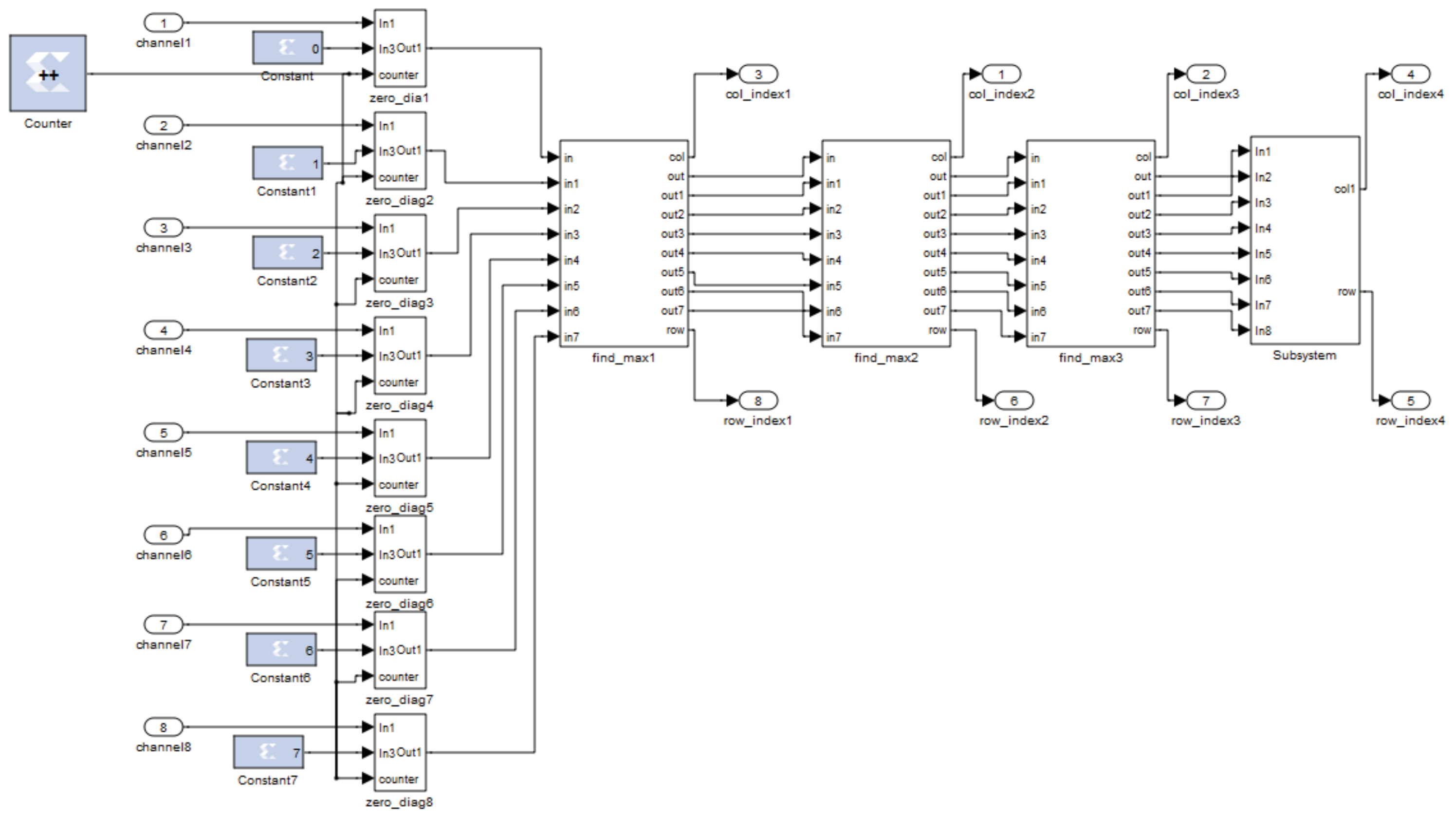


**Figure 3.7:** HLD diagram for the FPGA module to find 4 big elements from a $8 \times 8$ streaming matrix

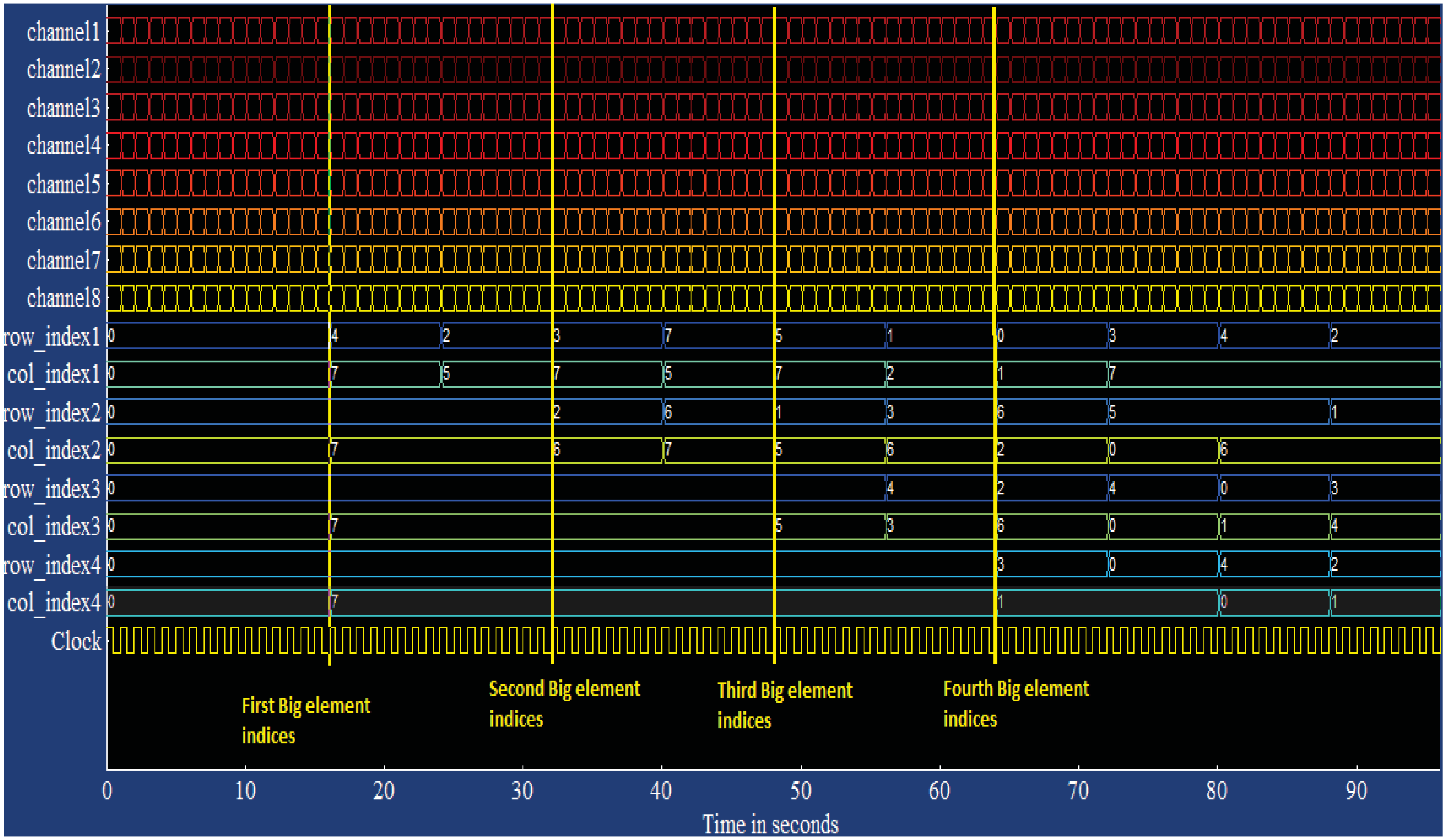


**Figure 3.8:** Timing diagram for the FPGA module for finding 4 big elements from a $8 \times 8$ streaming matrix

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} & a_{68} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} \end{bmatrix} \xrightarrow{\text{Extract submatrix}} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{ii} & b_{ij} \\ b_{ji} & b_{jj} \end{bmatrix}$$

**Figure 3.9:** An example of extracting a $2 \times 2$ submatrix corresponding to indices i and j from a $8 \times 8$ matrix

### 3.2.2 Submatrix selector

Once the indices of the big elements are available, we need to extract the submatrix with the biggest element as one of the off-diagonal element and symmetrically opposite entry of the $N \times N$ matrix as the other off-diagonal element. Figure 3.9 shows an example of submatrix extraction for a row column indices $(i, j)$. Either $a_{ij}$ or $a_{ji}$ is the big element found previously. Selecting submatrix is a simple logic based on multiplexers as depicted in Figure 3.10. The input to the module is streaming data of $N$ channels and the output is the submatrix which is produced at a downsampled rate of $f_s/N$ where $f_s$ is the frequency of operation. Figure 3.11 shows the timing diagram of extracting a $2 \times 2$ matrix from a $8 \times 8$ matrix given the row and column index.

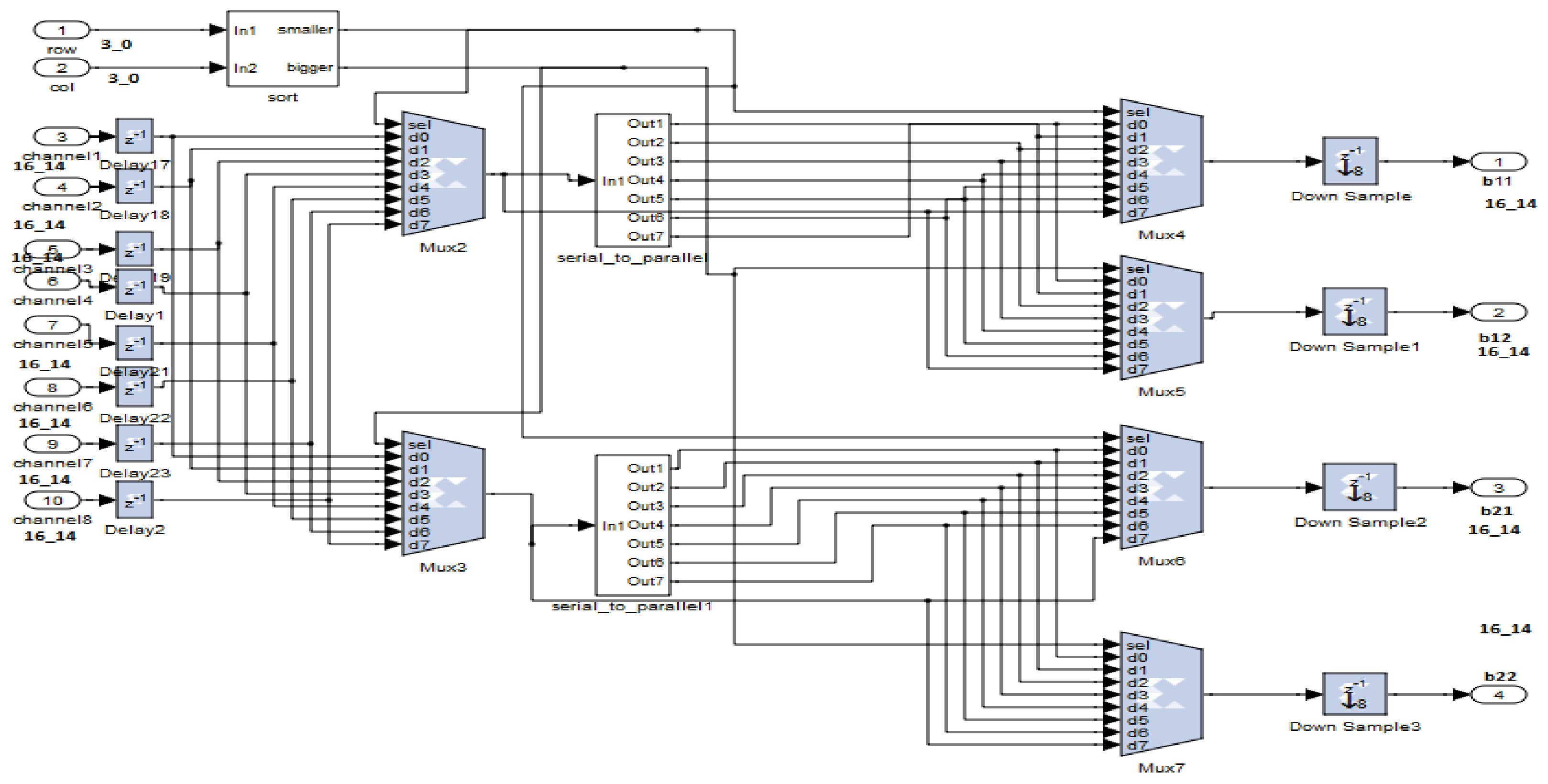


**Figure 3.10:** FPGA module for selecting a submatrix

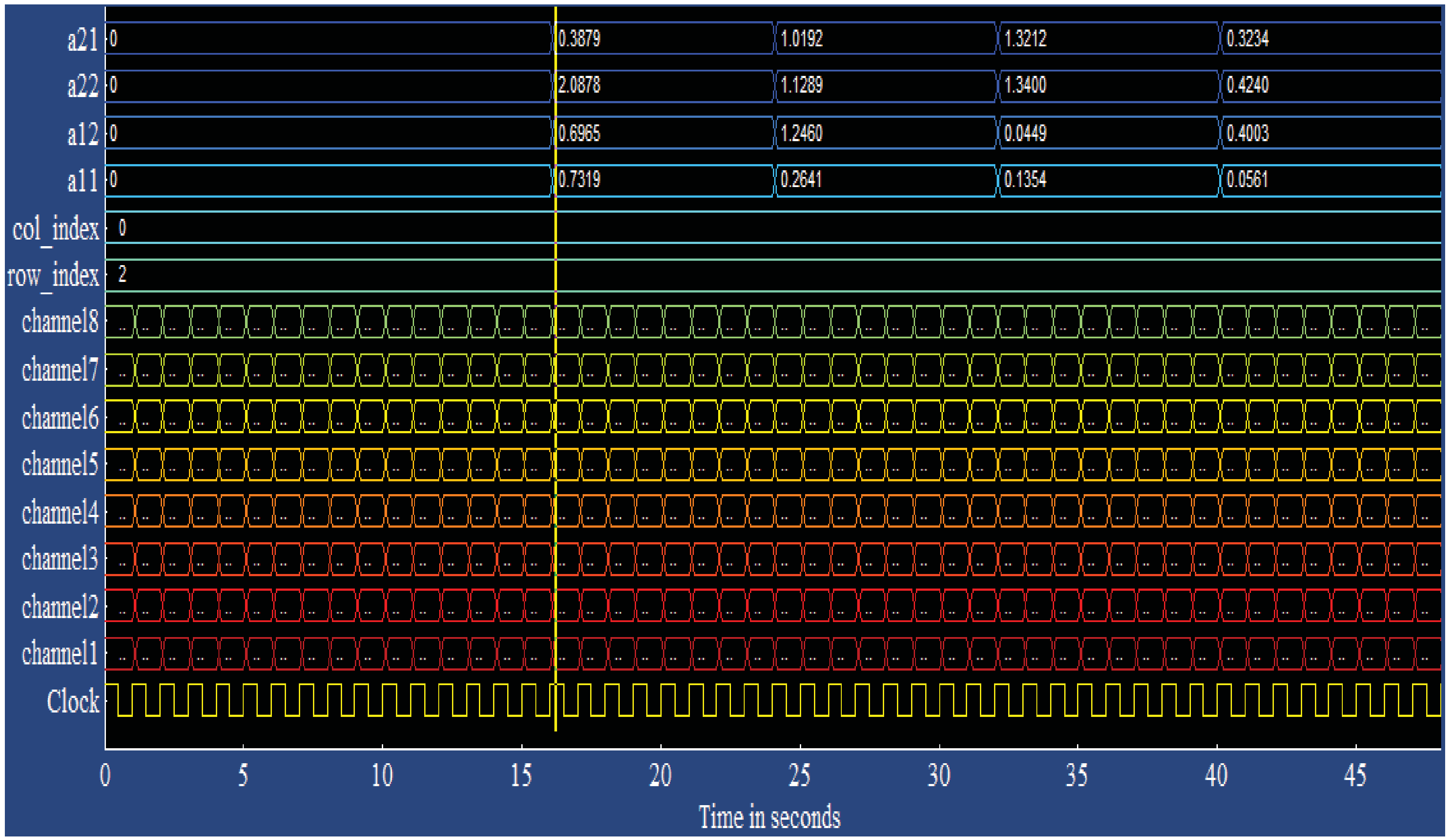


**Figure 3.11:** Timing diagram for the FPGA module for extracting a $2 \times 2$ submatrix from a $8 \times 8$ streaming matrix

### 3.2.3 Parameter generator

Given a submatrix, we apply the method given in [55] to obtain the rotation parameters that will annihilate the offdiagonal elements. Parameter generator only generates the parameter for left and right rotation. It does not apply them to the matrix. Let $A$ be the submatrix for which the rotation parameters need to be generated.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{3.1}$$

Then

$$\begin{aligned} p_1 = (a_{22}+a_{11})/2 \text{ , } p_2 = (a_{22}-a_{11})/2 \\ q_1 = (a_{21}-a_{12})/2 \text{ , } q_2 = (a_{21}+a_{12})/2 \end{aligned} \tag{3.2}$$

These parameters are used for calculating the left and right rotation parameters, $\theta_2$ and $\theta 2$

$$\begin{aligned} \theta_- = tan^{-1}(q_1/p_1) \text{ , } \theta_+ = tan^{-1}(q_2/p_2) \\ \theta_1 = (\theta_+ - \theta_-)/2 \text{ , } \theta_2 = (\theta_+ + \theta_-)/2 \end{aligned} \tag{3.3}$$

We then calculate the $cos(\theta_1)$ , $sin(\theta_1)$ , $cos(\theta_2)$ and $sin(\theta_2)$of these rotations parameters and transmit them. The arctan ,sin and cos operations are performed using CORDIC cores.

**Table 3.1**
Comparision of CORDIC in word serial and parallel configurations

| | *Maximum pipelining configuration* | *Maximum pipelining configuration* |
|---|---|---|
| Maximum frequency of operation | 139.237 MHz | 18.469MHz |
| Slice flip flops | 16% | 16% |
| 4 input LUTs | 15% | 2% |
| DSP48Es | 8% | 8% |

For further details of CORDIC configuration and architecture refer to [62].

The same operations are performed in Figure 3.12. Since physical space on FPGA was not a concern we chose to implement the CORDIC 4.0 with parallel architecture. This helped us achieve high frequency of operations. System generator core provides us with options to specify the level of pipelining. We can get maximum performance by pipelining to maximum extent which will cause more flip-flops to be used. Using minimal pipelining on the other hand reduces the number of flip flops used but cause reduction in maximum achievable frequency of operation. For the sake of comparison we synthesized a parameter generator one with CORDIC configured with maximum pipelining and another with minimal pipelining. This was implemented in Virtex 4 FPGA. The occupancy and frequency of operation can be compared in Table 3.1. What we can observe is only the number of flip flops that are occupied has increased in parallel configurations with huge increase in frequency of operation

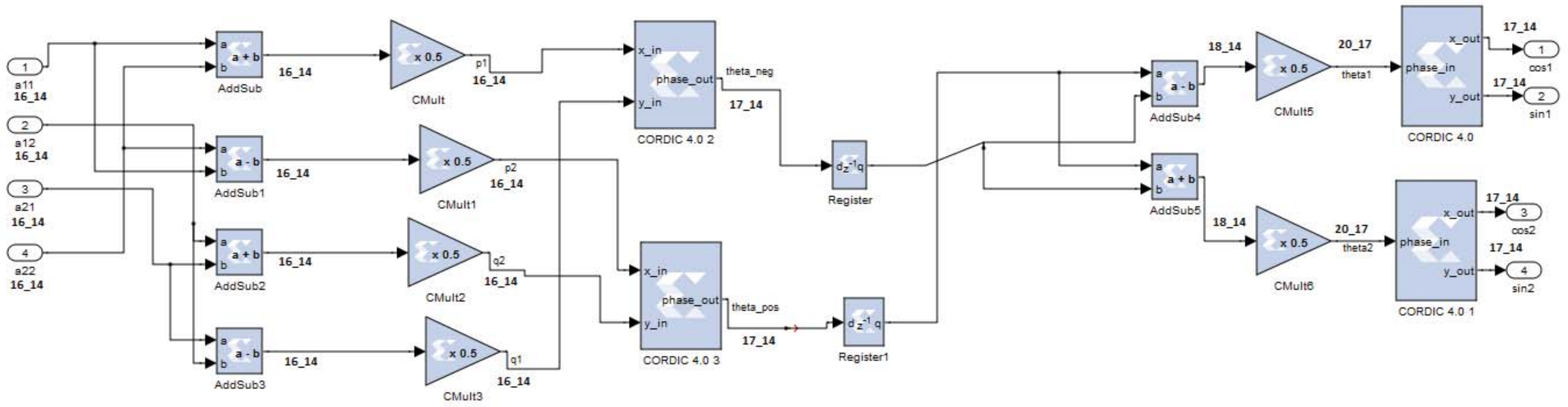


**Figure 3.12:** FPGA module for parameter generation

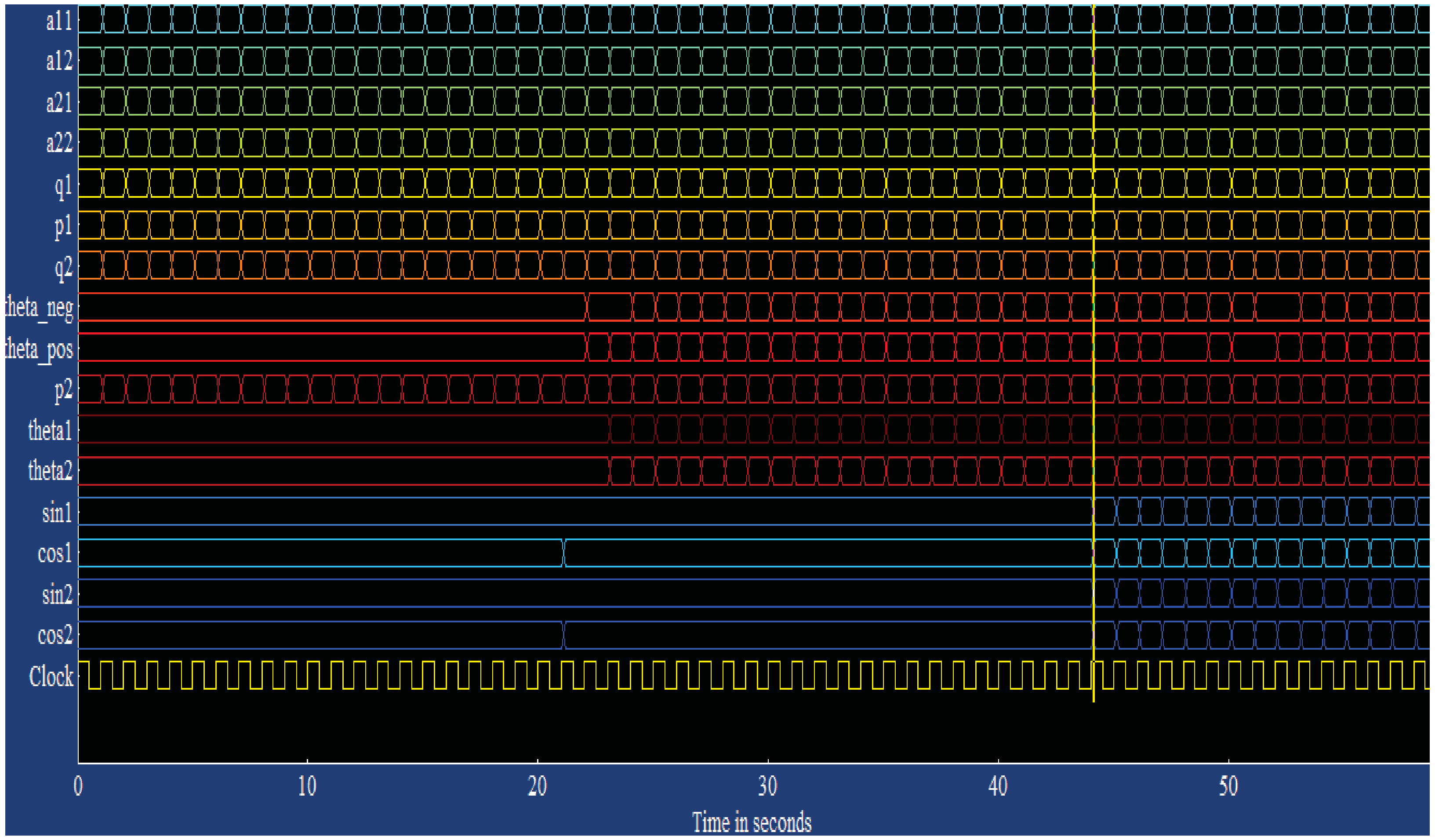


**Figure 3.13:** Timing diagram for the FPGA module for generating left and right rotation parameters

**Table 3.2**
Occupancy and frequency of operation for row multiplication

| | *Used* | *Available* | *Utilization* |
|---|---|---|---|
| Maximum frequency of operation | 260.756 MHz | NA | NA |
| Slice flip flops | 128 | 30,720 | v1% |
| 4 input LUTs | 152 | 30,720 | 1% |
| DSP48Es | 8 | 192 | 4% |

### 3.2.4 $2 \times 2$ matrix multiplication for row and column submatrix multiplication

Once the parameters are generated we need to apply the left and right sided rotation (column and row rotations) to the submatrices. Applying the rotation is 2 $2 \times 2$ matrix multiplication . A $2 \times 2$ matrix multiplication is 8 multiplications and 4 additions. We implement these using DSP48Es available in abundance in recent FPGAs. A $2 \times 2$ multiplication corresponding to left sided rotation is given as

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{3.4}$$

For column multiplication only the parameters change. The row multiplication module was synthesized on Virtex 4 and the occupancy and maximum achievable frequency achieved are as show in Table 3.2.This is depicted in Figure 3.15 and the corresponding timing diagram in Figure 3.14.

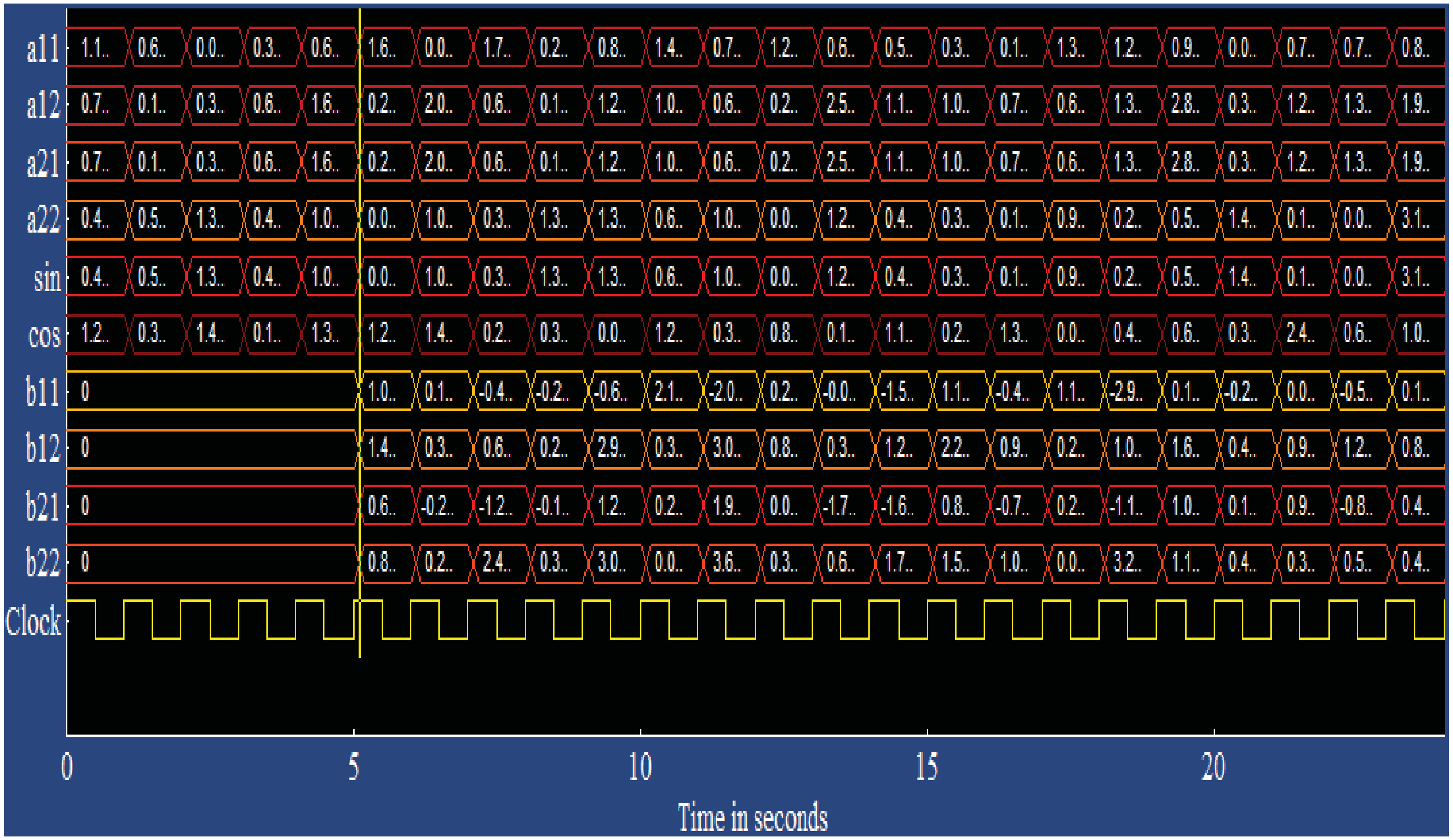


**Figure 3.14:** Timing diagram for the FPGA module for $2 \times 2$ submatrix multiplication

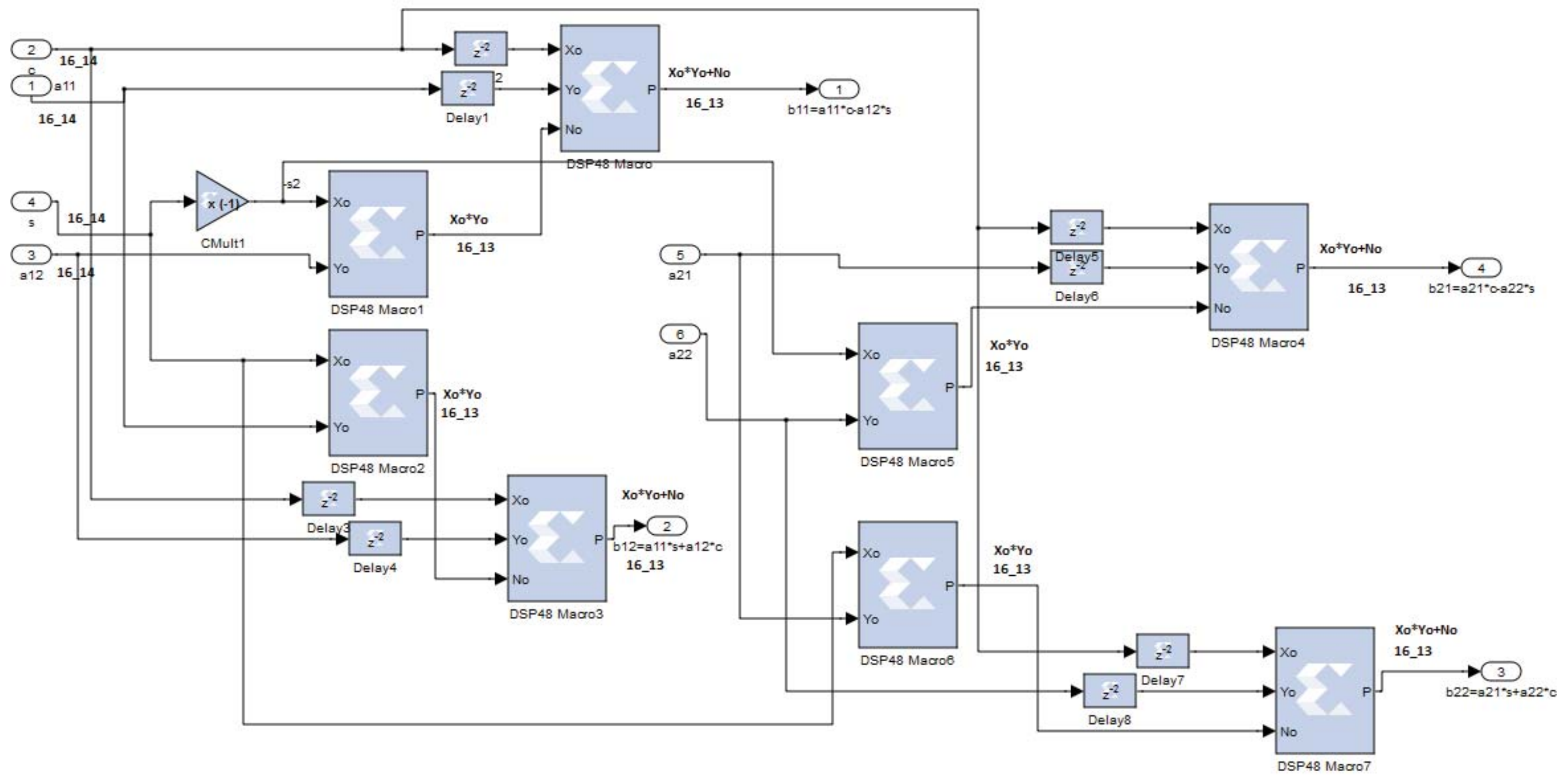


**Figure 3.15:** FPGA module for $2 \times 2$ matrix multiplication used for row and column submatrix multiplications

## 3.3 Conclusion

Modules which have major impact on the performance of the system such as parameter generator , row column multiplication (described in Section 2.3.2) were detailed in this chapter. We also detail the modules specific to proposed method namely big element finder and submatrix extractor. Other modules such as switch matrix and controller are simple to design and are open to interpretation by the designer. Controller can be implemented either as an FSM or on a soft core processors. Implementing on soft core processor makes the design easier and faster. The same soft core processors can be used for other control operations as well. We have tried to optimize the modules for performance. Special fucntional units such as DSP48E and IP cores such as CORDIC have been used to maximize performance. Although using these modules makes our design vendor specific, we believe these functional units and cores are available in most modern day FPGAs and can be easily replaced. This chapter describes the important modules in a detailed fashion and serves as a refernce for designers intending to replicate or extend the proposed design.

# Chapter 4

# Real-time root-MUSIC DOA estimation via a parallel polynomial rooting method

This chapter describes a new parallel polynomial rooting technique for real-time signal processing implementation of root-MUSIC suitable. Complex dynamics of root-MUSIC polynomial's Newton map were exploited to prescribe a minimal set of initial points for Newton's method. The proposed method is based on adaption of Newton's method as a global polynomial rooting technique. A set of initial points which guarantee that at least one of them would converge to the root closest to the unit circle (that corresponds to the direction of arrival (DOA)) were proposed. A comparison of the proposed method with existing general polynomial rooting technique was done to assess the computational complexity and possibility of parallelization. In addition, the performance of proposed system when incorporated into root-MUSIC was analyzed in terms of computations required to achieve a given accuracy of DOA.

## 4.1 Introduction

Direction of arrival (DOA) is a problem of interest in various applications such as wireless sensor networks (WSN), body area networks, local positioning systems etc. We are motivated to implement DOA estimation for a remote positioning system know as wireless local positioning system (WLPS)[7]. A WLPS system is intended for mobile devices, hence is limited in processing capabilities and requires real-time output. Over the past decades, various DOA estimation techniques have been developed. Subspace based techniques such as MUSIC[19], ESPRIT[28], maximum likelihood [63] and developments thereafter have gained significant attention. In particular, MUSIC and its variations have become popular due to their robustness, ease of implementation and because they are independent of array configuration. The spectrum search in MUSIC is a high complexity step and leads to the development of a search-free variation known as root-MUSIC[23]. Root-MUSIC uses polynomial rooting to find roots closest to the unit circle and in its original form is applicable only to uniform linear arrays (ULA). It is proven to have performance asymptotically similar to the MUSIC[24]. In WLPS, we deal with uniform linear array, hence root-MUSIC is a good choice for our DOA estimation.

As the number of sensors in the array increase, polynomial rooting becomes increasingly computation intensive. Researchers have proposed methods to reduce the complexity of polynomial rooting for DOA using Householder transformation[64, 65], which is a sequential process unsuitable for parallel implementations. Root-MUSIC based on unitary transformation resulting in real coefficient polynomials[66] were proposed as well. In this chapter, we aim to address polynomial rooting for complex coefficient polynomials, as the real polynomial rooting is a subset problem.

In mathematics, finding all roots of a complex polynomial of large degrees is an ongoing research [67].The most common method for finding the roots of a polynomial is via eigenvalue decomposition of companion matrix[35] and its faster variations [68, 69]. These methods are used in LAPACK and implementations on sequential machines. Their sequential nature make them unsuitable for real-time signal processing applications implemented on configurable platforms such as field programmable gated arrays (FPGA) and and single instruction multiple data (SIMD) platforms such as graphical processing units (GPU). Methods based on polynomial factorization [36, 70], though amenable to parallelization, are extremely complex to implement. Newton's method is a popular local rooting technique which can be adapted as a global rooting technique if a set of initial seed points which converge to each root can be obtained [37]. Newton's method has the advantages of fast convergence, simplicity of implementation and numerical stability. An attempt at using Newton's method was done in [71] by choosing an initial point using delay and sum (DAS) and iterating using root-MUSIC polynomial. This is effectively a local polynomial rooting technique and it fails to converge to the correct root in low SNR and multipath conditions.

This chapter depicts that root-MUSIC polynomial has a well defined Newton map geometry. This geometry is exploited to obtain a small set of well defined points which can be used as initial points for Newton's method. These set of points are selected such that there exists initial points for each root. By doing so, we are able to overcome the problem faced by [71] and effectively convert the Newton's method into a global polynomial rooting. Moreover, these set of points guarantee the convergence to the root closest to the polynomial irrespective of the nature of the channel. All the points in the prescribed set can be iterated in parallel, thus reducing overall processing time. Quadratic convergence of Newton's method allows us to fix the number of iterations based on the desired precisions of the root. The proposed method is highly parallel, terminates within fixed number of iterations resulting in a system with fixed latency, is easy to implement and can be easily extended to large sensor arrays. These properties make it highly suitable for implementation on FPGAs and SIMD hardware.

Section 4.2 highlights Newton map and signal model of the root-MUSIC crucial in developing the proposed method. Section 4.3 explores the complex dynamics of the Newton Map of root-MUSIC polynomial and emphasizes on certain well-defined characteristics that lead to the proposed method of polynomial rooting. Section 4.4 compares the proposed method to traditional methods in terms of computation complexity and it is found to be simpler than other methods and costs a degree less than other methods when implemented in parallel fashion. Section 4.5 concludes the chapter.

## 4.2 Background

This section explains the global geometry of Newton Map of a generic complex univariate polynomial [37]. We than briefly review the signal model of root-MUSIC.

### 4.2.1 Global geometry of Newton map

Throughout this chapter we deal with a complex univariate polynomial $p(z) : \mathbb{C} \to \mathbb{C}$ of a degree $d$. Newton's root finding method iterates on the associated Newton map

$$N_p : \mathbb{C} \to \mathbb{C}, z \to z - \frac{p(z)}{p'(z)} \tag{4.1}$$

to arrive at the root $\xi$ such that $p(\xi) = 0$. Newton map is a conformal map as the analytic function has derivative at all points of the complex plane. Given a starting point $z_0$, if the sequence $(z_0, z_1 = N_p(z_0), z_2 = N_p(z_1))$ converges to $\xi$, then $z_0$ is said to be in the basin of $\xi$. Collection of all such starting points which converges to $\xi$ is called the basin of root $\xi$. In addition, the connected components of the basin containing root $\xi$ is called the immediate basin $U_\xi$ . Connected spaces imply that any two points in space $D$ can be connected by

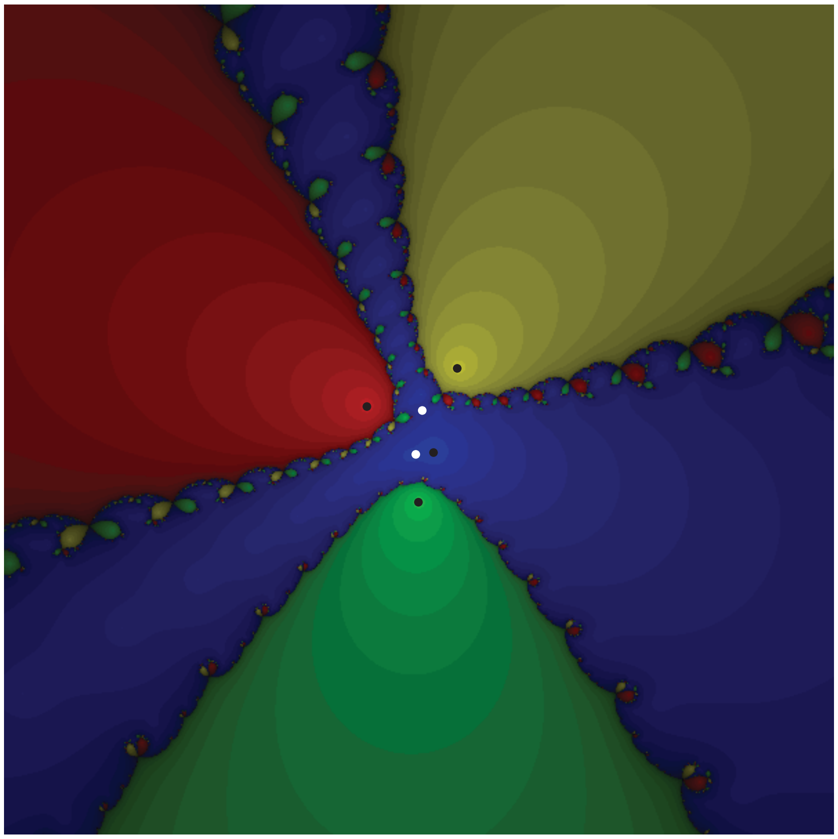


**Figure 4.1:** An example Newton Map

a curve lying wholly within $D$. In Figure 4.1, a Newton map of a fourth degree complex univariate polynomial $(4-3i)z^4+(2-3i)z^3+(1-2i)z^2+(1-3i)z+(2+1i)$ was sketched to illustrate different aspects of the geometry. The blue, green, red and yellow regions are the basins of the four roots of this polynomial, which are in turn indicated by black points in the complex plane. The basins are shaded in order to indicate how fast they converge to the root [72, 73].

Critical points of a conformal map forms the complex dynamics of the map. Hence we study the critical points of $N_p$, which are the solutions of

$$N_p'(z) = \frac{p(z)p''(z)}{p'(z)^2} = 0 \tag{4.2}$$

The roots of the polynomial and those of the second order derivative including multiplicities form the total number of critical points of a Newton map. The points in white in Figure 4.1 represent solutions of $p''(z) = 0$. Hence, both black and white points together form the complete set of the critical points. Let $m_\xi$ be the number of critical points in the immediate basin $U_\xi$ of a root $\xi$. From proposition 6 of [37], we know that an immediate basin has $m_\xi$ access to infinity. Accesses to infinity are the channels of basin that extend to infinity and each basin has atleast one such channel. This can be verified in Figure 4.1, where red, green and yellow basins have one access to infinity as there is only one critical point in the basin. Blue basin on the other hand has three separate accesses to infinity.

A set of at most $1.11d\log^2 d$ points (where $d$ is the degree of the polynomial) in $\mathbb{C}$ can be constructed for every polynomial $p(z)$, such that there is at least one point in the set that belongs to the basin of each root[37]. Polynomial $p$ needs to be normalized such that all its roots stay within the unit disk $D$. A specific set of $ns$ starting points $r_\nu \exp(i\vartheta_j)$ were suggested, where

$$s = \lceil 0.26632\log d \rceil, n = \lceil (8.32547d\log d) \rceil \tag{4.3}$$

$$r_\nu = (1+\sqrt{2})(\frac{d-1}{d})^{\frac{2\nu-1}{4s}}, \vartheta_j = \frac{2\pi j}{n} \tag{4.4}$$

for $1 \leq \nu \leq s$ and $0 \leq j \leq n-1$. It is to be noted that this applies to any polynomial and results in large number of starting points. In this chapter, we aim to reduce the number of starting points by exploiting the specific geometry of the root-MUSIC polynomial.

For an array of $M$ sensors and $L$ sources $(L < M)$, the received sensor signal is

$$y(t) = V * x(t) + n(t) \tag{4.5}$$

where $V = [V(\theta_1), ...., V(\theta_L)]$ is the steering matrix, $x(t) = [x_1(t), ...., x_L(t)]^T$ is the transmitted signal and $n(t) = [n_1(t), ...., n_M(t)]^T$ is the additive noise. The covariance matrix and it's eigen decomposition corresponds to

$$R = E(xx^H) = E_s\Lambda_s E_s^H + E_n\Lambda_n E_n^H \tag{4.6}$$

$E_s$ and $E_n$ are the signal and noise subspace respectively. $E(.)$ is the expectation opertaion.

$$\Lambda_s = diag(\lambda_1, ...., \lambda_L) \text{ and } \Lambda_n = \sigma^2 I_{(M-L)\times(M-L)} \tag{4.7}$$

are the signal and noise eiegenvalue vectors repectively, where $\sigma^2$ is the noise variance and $I$ represents an identitiy matrix. The MUSIC spectrum corresponds to

$$S(\theta) = \frac{1}{V^H(\theta)AV(\theta)}, \quad A = E_n E_n^* \tag{4.8}$$

Here $\theta$ represents the angle at which the spectrum is evaluated and the range and resolution depends on desired precision of DOA estimation. $V$ is the steering vector as given in (4.5) and $E_n$ is the noise subpaces as given in (4.7). For a uniform linear array the $m^{th}$ element of steering vector, $V(\theta)$ is

$$V_m(\theta) = e^{-j2\pi m(\frac{s}{\lambda})\sin(\theta)}, \quad m = 1...M \tag{4.9}$$

,where $s$ is the seperation between elements of the antenna array and $\lambda$ is the wavelength of the signal. Inverse of MUSIC spectrum in (4.8) can be simplified using(4.9) as

$$\begin{aligned} S^{-1}(\theta) &= \sum_{m=1}^{M}\sum_{n=1}^{M} e^{-j2\pi m(\frac{s}{\lambda})\sin(\theta)} A_{mn} e^{j2\pi m(\frac{s}{\lambda})\sin(\theta)} \\ &= \sum_{l=-M+1}^{M-1} a_l e^{-j2\pi l(\frac{s}{\lambda})\sin(\theta)} \end{aligned} \tag{4.10}$$

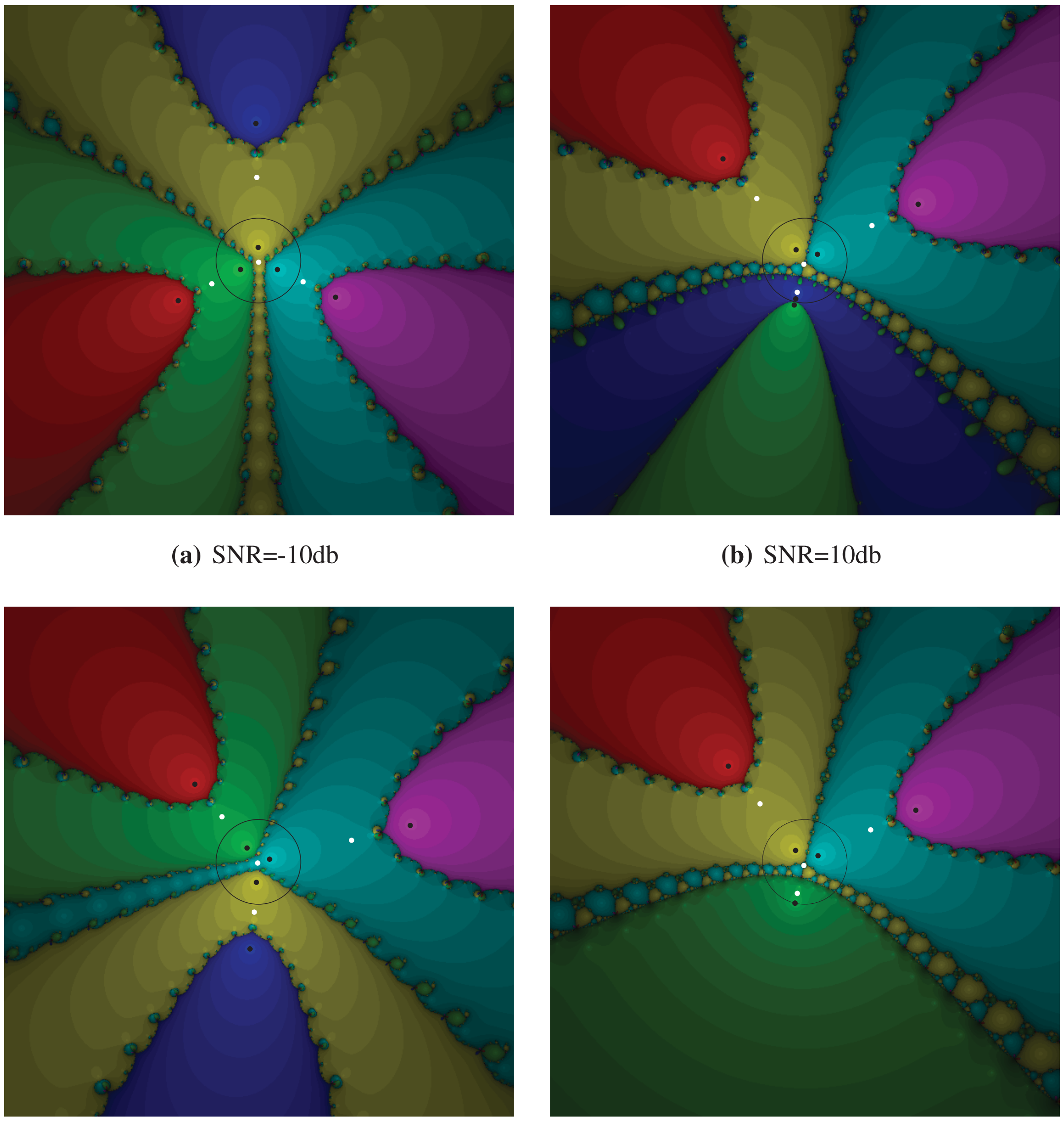
where $s$, $\lambda$ and $\theta$ are parameters as explained in (4.9) and (4.8) respectively. In addition, $a_l = \sum_{m-n=l} A_{mn}$ is the sum of entries of A along the $l^{th}$ diagonal. A polynomial can be constructed as

$$D(z) = \sum_{l=-M+1}^{M-1} a_l z^{-l} \tag{4.11}$$

where $a_l$ is as explained in (4.10). It is well known that roots on the unit circle of a polynomial $D(z)$ give the direction of arrival of the signal. From now on we refer to $D(z)$ as root-MUSIC polynomial (RM polynomial).

## 4.3 Complex dynamics of root-MUSIC polynomial and proposed polynomial rooting technique

Exploring the dynamics of Newton map of the RM polynomial was the key to arriving at our polynomial rooting method. In this section, we make a few observations regarding Newton map of RM polynomials and try to prove that these observations are valid for any RM. Figure 4.2 depicts the Newton map for RM polynomial of degree $d = 6$ i.e., for a sensor array of size $M = 4$. The received signal was measured at different channel conditions namely AWGN channel with $SNR = -10db$ and $SNR = 10db$, Rician channel with $NLOS/LOS = -5db$ and additive noise of $SNR = 0$ and ideal channel condition of $SNR = inf$. Figure 4.3 depcits the Newton map for various sensor array size ($M$) ie. RM polynomials of various degrees ($d = 2M-2$) at AWGN with $SNR = 0db$. A unit circle is also marked to offer an idea regarding relative positioning of the roots.



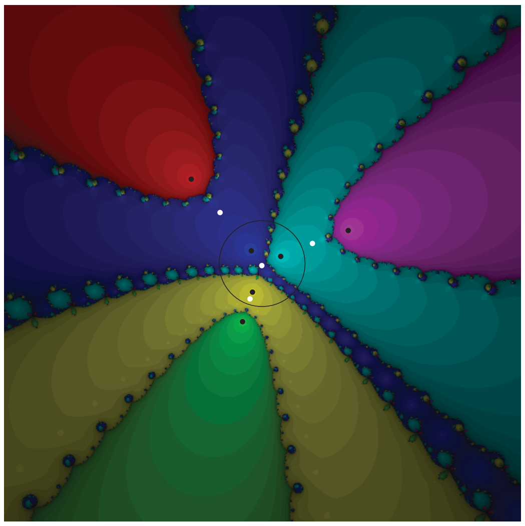
**(a)** SNR=-10db

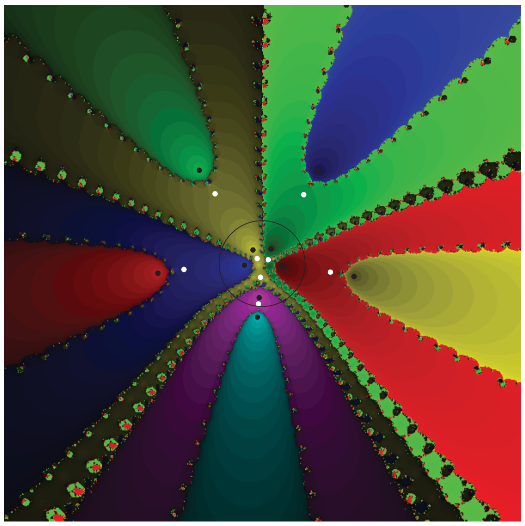
**(b)** SNR=10db

**(c)** SNR=0db and pdb=-5db

**(d)** SNR=inf db

**Figure 4.2:** Newton map of RM polynomial of degree 6 ie. a sensor array of 4 elements at various channel conditions


**(a)** Array size=4


**(b)** Array size=6

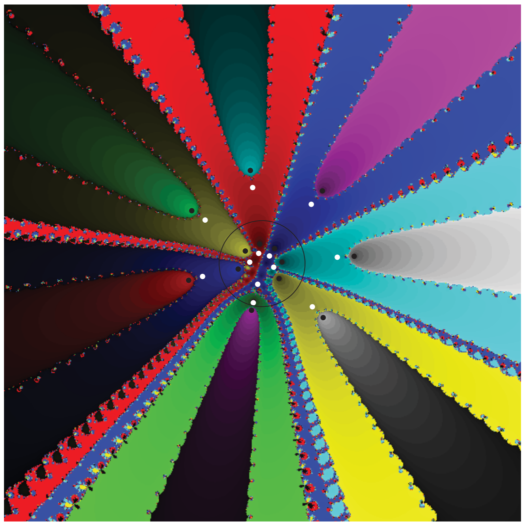

**(c)** Array size=8

**Figure 4.3:** Newton map of RM polynomial of various degrees at SNR=0db

### 4.3.1 Symmetry of polynomial roots across unit circle

The hermitian nature of matrix $A$ in (4.8) enables the coefficients of polynomial $D(z)$ to be complex conjugate pairs $a_l = a_l^*$. Accordingly,

$$D(z) = D^*(z) \tag{4.12}$$

Let the factorization of $D(z)$ and $D^*(z)$ be

$$D(z) = \prod_{k=0}^{d-1}(z - z_k), D^*(z) = \prod_{k=0}^{d-1}(1 - \bar{z}_k z) \tag{4.13}$$

where $d$ is the degree of polynomial. $z_k$ and $\bar{z}_k$ are the $d$ roots of the polynomial. Since $D(z) = D^*(z)$ we have $\bar{z}_k{}^{-1} = z_k$. This confirms that roots of a RM polynomial exist in pairs which are symmetric across the unit circle. Let the $d$ roots of $D(z)$ be represented as

$$z_0 = |z_0|e^{j\arg|z_0|} \ldots . z_{d-1} = |z_{d-1}|e^{j\arg|z_{d-1}|} \tag{4.14}$$

where $d = 2M - 2$. Due to symmetric nature the roots, the DOA represented by that roots are equal.

$$\frac{\lambda}{2\pi d}\arg z_k = \frac{\lambda}{2\pi d}\arg z_{d-k-1}, \quad k = 0 \ldots d/2 - 1 \tag{4.15}$$

This can be validated via Newton maps depicted in Figure 4.2 and Figure 4.3, where roots exist in symmetric pairs independent of the channel conditions and degree of polynomial be. Moreover, as SNR increases the roots corresponding to DOA move closer to the unit circle.

### 4.3.2 Accesses to infinity

As stated before, the critical points of the polynomial, hence the accesses to infinity are important in understanding the complex dynamics. For a polynomial $D(z)$ of degree $d$, total number of critical points are $(d+(d-2))$ corresponding to $d$ roots of $D(z)$ and $(d-2)$ roots of $D''(z)$. From Figure 4.3 we can see that roots of $D''(z)$ (depicted in white points) always lie in the basin of roots of $D(z)$ found within unit disk. We do not know of any case where this does not hold true for RM polynomials. This implies only roots within the unit disk have multiple access to infinity. Also we observe that basin of the roots fork into two channels at the apex of convex hull of roots outside the unit disk. Hence the total number of channels that intersect with the unit disk are

$$N_u = \frac{d}{2} + (d - 2 - \frac{d}{2}) = d - 2 \tag{4.16}$$

### 4.3.3 Proposed set of initial points

(4.15) confirms that DOA corresponding to the symmetric pair of roots are equal and it suffices to find one of the two. Thus, we need to find only half of the roots of a polynomial which reduces the number of initial points required for the Newton's method by two fold. We choose to find the set of roots inside the unit circle for two reasons. Firstly, the intial points can be placed on the unit circle for we know that unit circle lies entirely in the basins of the inner roots. Secondly, the distance between the initial point and the root it converges to is less, thus reducing the number of Newton iterations it takes to converge to the roots. From (4.16) we know that number of channel on the unit circle is $(d-2)$. Ignoring the fact that each channel can be of varying thickness on the unit circle, we place the $(d-2)$ equi-spaced initial points on the unit circle. This can lead to some of the roots having more than one initial points in its basin. One might argue that it is possible that no initial point lies in the basin of one or more roots. We observed that thickness of the channels are almost similar, reducing the chances of missing a basin while placing an initial point. We confirm with the help of extensive simulations of Newton maps of RM polynomial of degree 6 to 510 ($M = 6$ to 256) that none of the basins are missed. The proposed set of initial points are

$$z_{0k} = 1 * e^{1i*\frac{2\pi k}{N_u}}, \quad k = 1 \dots N_u \tag{4.17}$$

,where $N_u$ is the number of channels on unit disk as given in (4.16). In order to give a perspective on the reduction in size of initial set of points, we compute the number of initial points prescribed in (4.17) and (4.3). For polynomial of degree 14 ($M = 8$) the size of initial set is 12 points for the proposed method as compared to 308 points for the later. The huge size of initial set is justified for a generic polynomial but would be an overkill for structured simple polynomial like ours.

### 4.3.4 Proposed method

Now that the initial set of points are available to us, we can apply Newton's method to each of these initial points simultaneously. The required number of iterations $(N_i)$ is decided by the desired precision of roots. The correct root (corresponding to the DOA) from final set of points can be identified by two conditions. Firstly, the difference between consequent Newton iterations should converge to predefined tolerance $(\varepsilon)$ within $N_i$ iterations. That is

$$|z_{i+1} - zi| \rightarrow \varepsilon \text{ as } i \rightarrow N_i \tag{4.18}$$

This test eliminates all the initial points which lead to orbits attracted to stable periodic orbits. The second and more obvious condition being the correct root is the root closest to the unit circle.

The algorithm for proposed method is as below.

```
Choose $\varepsilon, N_i$

Determine $N_u = d - 2$

Initialize final set of points $N_f = [\,]$

**for** $i = 1, \ldots, N_u$ **do**

    initial point, $z_{i0} = 1 * e^{1i * \frac{2\pi k}{N_u}}$

    **for** $j = 1, \ldots, N_i$ **do**

        $z_{ij} = z_{i(j-1)} - \frac{p(z_{i(j-1)})}{p'(z_{i(j-1)})}$

    **end for**

    **if** $z_{j=1} - z_j \to \varepsilon$ as $j \to N_i$ **then**

        $N_f = [N_f, z_{iN_i}]$

    **else**

        discard $z_{iN_i}$

    **end if**

**end for**

Find root closest to the unit circle $z_{final}$

$\theta_{DOA} = \sin^{-1}(\frac{\lambda}{2\pi d}\arg(z_{final}))$
```

## 4.4 Simulation and Complexity analysis

It is clear that we need to perform $N_u \times N_i$ polynomial evaluations of $p(z)$ and $p'(z)$ each. Without using any special polynomial evaluation methods, each of these polynomial evaluations $p(z)$ of degree $d$ requires $2d$ multiplications and $d$ additions and polynomial evaluations $p'(z)$ requires $d$ multiplications and $d$ additions. It also needs $N_i \times N_u$ divisions and subtractions. Thus total number of floating point operations (assigning same weights to +,-,$\times$,/) required for proposed method is

$$\text{Total flops (proposed)} = N_u \times N_i \times (5d + 2) \tag{4.19}$$

Since $N_u = O(d)$, the overall computations is in the order of $O(d^2)$. Using $(d-2)$ parallel processors, the cost of computations in each processor is $O(d)$. Table 4.1 compares the various existing polynomial rooting with proposed method in terms of computation complexity and possibility of parallel implementation. Lower number of initial points, sufficient exploitable parallelism and $O(d)$ cost for each processor justifies the superiority of the proposed method over others. Monte-Carlo simulations were conducted to evaluate the performance of the proposed system in various channel conditions. We compare the performance in terms of root mean square error (RMSE) in DOA estimation at varying number of computations. The computations were expressed as total number of flops required for proposed method given in (4.19) and compared with the method proposed in

**Table 4.1**
Comparison of computational complexity of various polynomial rooting method

| Method | Computational Complexity | Parallel Implementations | Comments |
|---|---|---|---|
| Eigenvalue of companion matrix using QR decomp.[35] | $O(d^3)$ | None,QR decomp. is inherently sequential | Each QR step is $O(d^2)$ |
| Fast QR based on $n^{th}$ roots of unity of $p(z)$[68] | $O(d^2)$ | None,QR decomp. is inherently sequential | Each QR step is $O(d)$ |
| Roots by bala- nced factorization of polynomial[70] | $O(dlog^5(d)logB)$ | $O(log^6 d)logb$ using $O(dlog^{\omega} d)$ processors | Factorization introduces errors |
| Newton's method for general polynomial[37] | $O(d^2 log^2(d))$ | $O(d)$ using $O(dlog^2 d)$ processors | Large set of initial points |
| Proposed method for RM polynomial | $O(d^2)$ | $O(d)$ using $O(d)$ processors | Method specific to RM polynomials |

[68], as this is an improved version of popular "eigenvalue of companion matrix" methods. The total number of flops required for this method is :

$$\text{Total flops (existing method)} = 123 * d^2 + 32 * d \tag{4.20}$$

Figure 4.4 and Figure 4.5 evaluates the performance of proposed method for a single source at $DOA = 35^o$ where 32 samples were used for constructing covariance matrix for pure AWGN channel and Rician channel with one non-line of sight (NLOS) component respectively. It can be observed that the proposed method can achieve same the RMSE as existing method, with much lesser number of computations. Usually under good channel

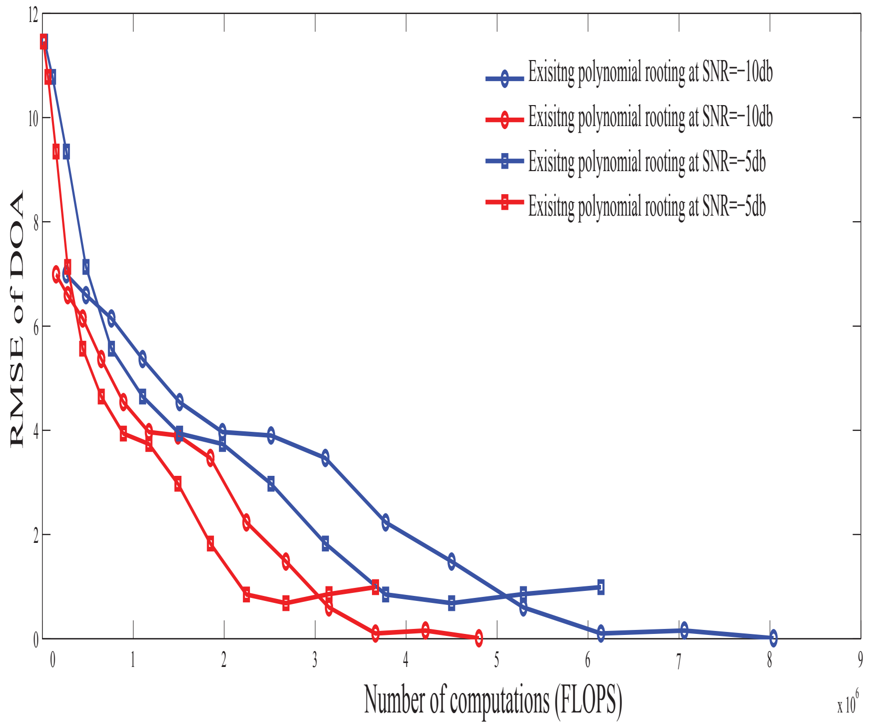


**Figure 4.4:** Computations vs RMSE in DOA for AWGN channel

conditions computationally cheap methods such as Delay and Sum (DAS) and its variations are used and systems resort to more complex methods like root-MUSIC only when channel conditions worse. Therefore, we have chosen to evaluate the performance only at low SNR and multipath conditions. It is to be noted that the proposed algorithm only replaces the root finding step in root-MUSIC algorithm therefore the overall performance of root-MUSIC remains unchanged. The advantage of proposed method lies in reduced complexity and inherent parallelism over traditional root finding methods.

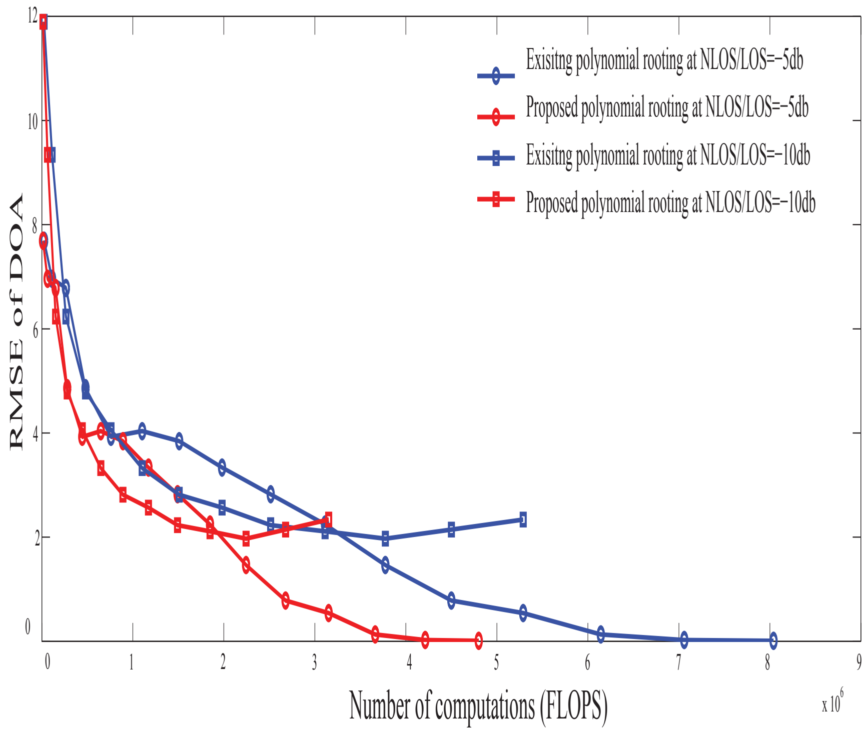


**Figure 4.5:** Computations vs RMSE in DOA for Rician channel

## 4.5 Conclusion

In this chapter, we propose a new parallel polynomial rooting technique for root-MUSIC. Complex dynamics of root-MUSIC polynomial's Newton maps were studied and few well defined characteristics were described and proved. By providing a well defined set of initial points for Newton's method that guarantee convergence to the desired root, we eliminated the need for other complex polynomial rooting techniques. The proposed method has an overall computational cost of $O(d^2)$ and $O(d)$ when implemented on $O(d)$ parallel processors. It is seen that comparable accuracy in DOA can be achieved via much lower computations by using the proposed technique. Inherent parallelism, fixed latency, ease of implementation and ease of extension to large sensor array (hence higher degree polynomial rooting) characterize this method and makes it suitable for real-time signal processing. Based on the results of this chapter is it anticipated that better understanding of complex dynamics of polynomials in other DOA techniques and signal processing schemes can lead to better polynomial rooting techniques. Future directions involve implementing the proposed method on a FPGA platform and exploring other polynomials in DOA estimation techniques.

# Chapter 5

# Conclusion

## 5.1 Conclusion

There is a growing need for real-time localization systems with various civilian and military applications. DOA estimation is considered a key component in many emerging localization systems such as WLPS. Subspace based DOA estimation for ULA, known as root-MUSIC was chosen for WLPS due to its superior performance with lower computational complexity compared to other subspace methods. Although root-MUSIC is computationally less complex than other DOA estimation techniques, it is still not simplified enough to meet the real-time constraints. This thesis was motivated by the need for real-time signal processing for root-MUSIC DOA implementation. Figure 5.1 depicts the root-MUSIC system with proposed algorithms replacing the traditional algorithms. The contributions of this thesis are mainly focused on: Fast converging SVD (subspace decomposition) and low cost, parallel polynomial rooting. The proposed modules are underlined. Both these algortihms were proposed with the objective of meeting real-time constraints for the WLPS system. FPGA was chosen as the hardware platform for reasons discussed before in section **??**. Both the proposed algorithms were analyzed and compared with existing algorithms. SVD algorithm was implemented on FPGA.

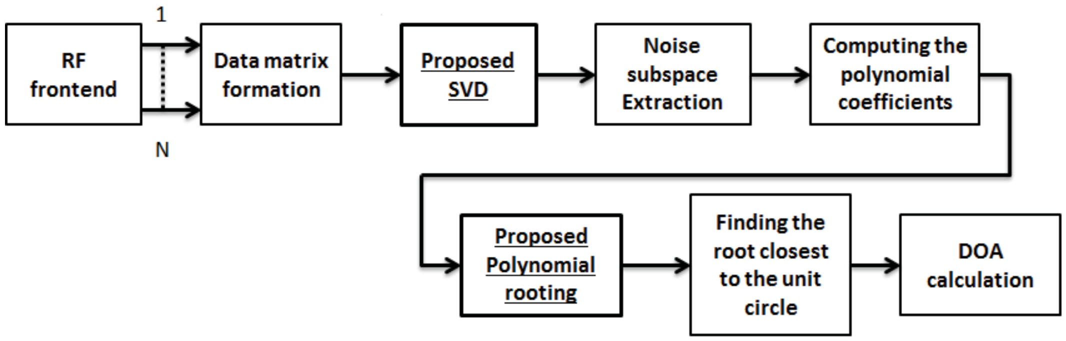

RF frontend
1
N
Data matrix formation
Proposed SVD
Noise subspace Extraction
Computing the polynomial coefficients
Proposed Polynomial rooting
Finding the root closest to the unit circle
DOA calculation

**Figure 5.1:** Root-MUSIC system with proposed algorithms

A new fast converging SVD was proposed, which depicted reduction in the number of iterations by almost half for large matrix sizes. This algorithm is suitable for real-time signal processing applications such as localization, channel estimation in MIMO systems, image processing etc. The fast convergence rate was due to the proposed dynamic ordering where $\frac{N}{2}$ big elements were annihilated in each iteration, where $N$ is the size of matrix. The proposed algorithm retained the parallelism proposed in [31]. The performance was analyzed against BLV array in terms of number sweeps (iteration/$N$) at various matrix sizes. The results were promising and motivated us to implement the proposed algorithm on FPGA.

A system design for the proposed algorithm was presented so that it can be utilized by designers who might want to adopt the proposed algorithm for other hardware platforms. The proposed design can be extended to bigger matrix sizes with minimal changes. The traditional BLV array and the proposed design were implemented on FPGA to compare their convergence rate and throughput. Highly parallel and pipelined architectures were developed for both designs. Comparative throughput for both algorithms coupled with faster convergence for the proposed algorithm represents the achievable improvement in the proposed method. Upto $180MHz$ of maximum frequency ($f_{max}$) can be achieved for a $4 \times 4$ matrix while it occupies less than 10% of FPGA capacity . $8 \times 8$ SVD can be clocked at upto $107MHz$ with less than 30% occupancy. The tradeoff of proposed method is increase in initial latency. For streaming architecture such as the one proposed it this thesis, the latency only impacts the time of first output, which can be treated as warming up period.

With improvements in SVD, the complexity bottleneck was shifted to the computationally complex task of polynomial rooting. As mentioned before in section 4.1 many polynomial rooting techniques are sequential and unsuitable for real-time polynomial rooting implementation. This thesis proposes a set of initial points such that all roots can be found by applying Newton iterations to these points. This initial set was developed vis analyzing complex dynamics of Newton Map of the root-MUSIC polynomial. The proposed method was compared with many existing methods, both sequential and parallel. When implemented in sequential manner the proposed method has computations cost of $O(d^2)$ and when implemented on $O(d)$ parallel processors its cost is $O(d)$, where $d$ is the degree of polynomial. The proposed method operates based on unique characteristics of root-MUSIC.

## 5.2 Future Work

### 5.2.1 In the direction of SVD

#### 5.2.1.1 Improvements to the proposed design

1. **Continued design effort**: Although maximum amount of parallelism has been exploited and the system has been pipelined to a large extent, continued design efforts can lead to slight improvements in the performance. The question remains if the improvement achieved is worth the design effort.

2. **Improvement in latency**: As already pointed out, the tradeoff of improvement in performance is an increase in initial latency. Although for streaming data processing, this latency translates to warm-up period, we believe this can be reduced using better

schemes to find $\frac{N}{2}$ maximum elements. Another question that remains unanswered is "How much degradation in performance will be seen if we do not target the best $\frac{N}{2}$ maximum elements?" . The present scheme finds the best possible $\frac{N}{2}$ maximum elements following row column exclusivity. We can improve the latency if we deviate from the proposed scheme but at the cost of performance.

#### 5.2.1.2 Developing an IP

It was emphasised throughout the thesis that there are no COTS IP cores for SVD. The design can easily lead to a versatile IP by working on the robustness of the design and by introducing more configurability. Following configurations can be introduced :

1. Word length and accuracy of singular values.

2. Matrix size as dynamic reconfiguration of FPGA [74]

3. Implementation of multiplication can be configured to either use DSP48Es or LUT base multipliers.

4. Implementation of CORDIC: Individual CORDIC modules can be configured but a high level configuration for all CORDICs will be convenient for users.This can be incorporated to either maximize the performance or minimize occupied slices.

#### 5.2.1.3 SVD using 4-Dimensional Given Rotation

Hmailtonian Quaternion algebra has been known for quite some time now and in [75], it was used for replacing $2 \times 2$ plane rotation with more powerful $4 \times 4$ rotation. This leads to Quaternion Jacobi like method for eigen value decomposition (EVD), which guarantees at least one sweep less than $2 \times 2$ Jacobi. It is possible to derive an SVD using $4 \times 4$ Quaternion Jacobi rotation (J4).

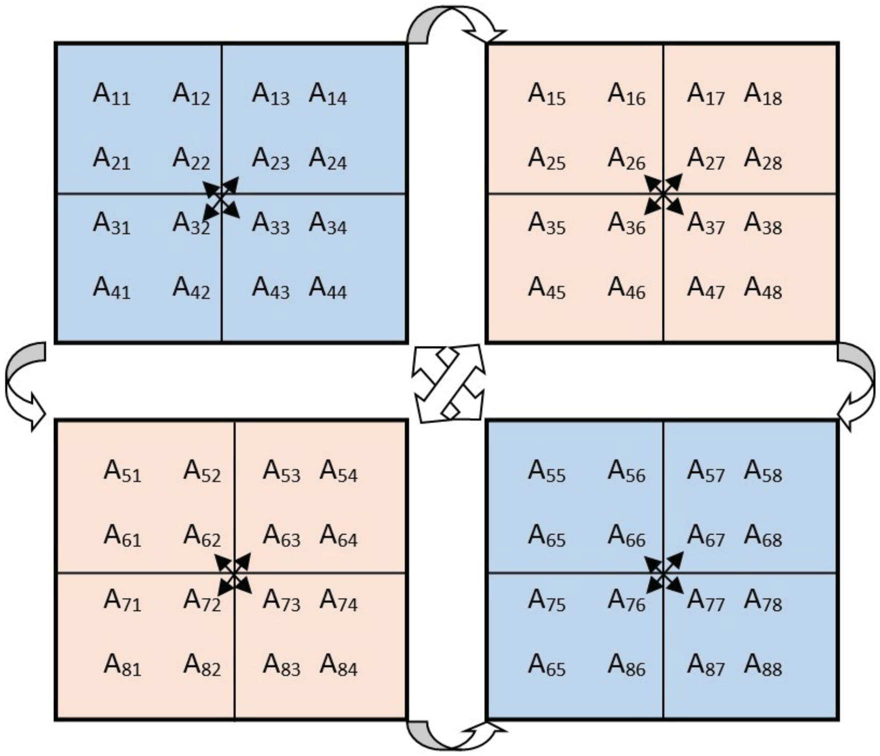


**Figure 5.2:** SVD using 4 dimensional Given's rotation

Figure 5.2 shows $2 \times 2$ array of J4 processors used for digitalizing a $8 \times 8$ matrix. Similar to original BLV a static ordering scheme can be developed for J4 based systolic array. This will lead to a reduction in the number of sweeps. But the question remains " Is this reduction in the number of sweeps justified by the increase in complexity?" J4 algebra for symmetric matrices has already been successfully derived for, but J4 algebra for general matrix needs attention. In a fashion similar to proposed method, we can target more than one big element in $4 \times 4$ submatrix.

### 5.2.2 In the direction of polynomial rooting

#### 5.2.2.1 Complex dynamics of Newton map of root-MUSIC polynomial

Although many of the observations of complex dynamics of Newton map of root-MUSIC polynomial have been proven in section 4.3, the access to infinity and related observations haven't been thoroughly treated. Study in this direction might expose further reduction of set of initial points or might expose shortcomings of the method that haven't been discovered yet.

#### 5.2.2.2 Implementation and others

1. FPGA implementation of the proposed Polynomial rooting needs to be completed to evaluate the occupancy and maximum achievable frequency of operation.
2. For larger degree of polynomials, it will be an interesting observation to see if implementing the proposed algorithm on graphical programming unit (GPU) offers any advantage over FPGA implementation.
3. To explore if other polynomial rooting based subspace techniques exhibit regular complex dynamics like root-MUSIC does.

# References

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 54–69, 2005.

[2] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung, "Body area networks: A survey," *Mobile Networks and Applications*, vol. 16, no. 2, pp. 171–193, 2011.

[3] A. Nehorai and E. Paldi, "Acoustic vector-sensor array processing," *Signal Processing, IEEE Transactions on*, vol. 42, no. 9, pp. 2481–2491, 1994.

[4] S. Guolin, C. Jie, G. Wei, and K. J. R. Liu, "Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 12–23, 2005.

[5] S. Gezici, T. Zhi, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects

for future sensor networks," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 70–84, 2005.

[6] S. A. Zekavat, O. Abdelkhalik, S. T. Goh, and D. R. Fuhrmann, "A novel space-based solar power collection via LEO satellite networks: Orbital management via wireless local positioning systems," in *Aerospace Conference, 2010 IEEE*, pp. 1–9.

[7] T. Hui and S. A. Zekavat, "A novel wireless local positioning system via a merger of DS-CDMA and beamforming: Probability-of-detection performance analysis under array perturbations," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 3, pp. 1307–1320, 2007. (Reza).

[8] S. A. Zekavat, H. Tong, and J. Tan, "A novel wireless local positioning system for airport (indoor) security," in *Defense and Security*, pp. 522–533, International Society for Optics and Photonics.

[9] L. C. Godara, "Application of antenna arrays to mobile communications. ii. beam-forming and direction-of-arrival considerations," *Proceedings of the IEEE*, vol. 85, no. 8, pp. 1195–1245, 1997.

[10] A. J. Viterbi, *CDMA: principles of spread spectrum communication.* Addison Wesley Longman Publishing Co., Inc., 1995.

[11] S. G. V. Giri, G. A. Price, and S. R. Zekavat, "A novel synchronization method for active positioning via DSSS: Achieving low resource usage and latency," in *Wireless*

*for Space and Extreme Environments (WiSEE), 2013 IEEE International Conference on*, pp. 1–6.

[12] M. Roddewig, S. A. Zekavat, and S. Nooshabadi, “Design of a costas loop down converter,” in *Circuits and Systems, 2009. MWSCAS ’09. 52nd IEEE International Midwest Symposium on*, pp. 244–247. (Reza).

[13] H. Tong and S. A. Zekavat, “LCMV beamforming for a novel wireless local positioning system: a stationarity analysis,” in *Defense and Security*, pp. 851–862, International Society for Optics and Photonics.

[14] H. Tong, J. Pourrostam, and S. Zekavat, “Optimum beam-forming for a novel wireless local positioning system: a stationarity analysis and solution,” *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 2007.

[15] M. Pourkhaatoun and S. Zekavat, “A novel ICA-based TOA estimation technique: achieving high resolution, high reliability, and, low cost,” in *proceedings IEEE International Workshop on Signal Processing and its Applications, WOSPA*, vol. 8, pp. 18–20.

[16] S. Zekavat, A. Kolbus, X. Yang, Z. Wang, J. Pourrostam, and M. Pourkhaatoun, “A novel implementation of DOA estimation for node localization on software defined radios: achieving high performance with low complexity,” in *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, pp. 983–986, IEEE.

[17] Z. Wang, “A novel semidistributed localization via multinode TOAâĂŞDOA fusion,” *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 7, pp. 3426–3435, 2009.

[18] S. A. R. Zekavat, *An Introduction to Direction-of-Arrival Estimation Techniques via Antenna Arrays*, pp. 279–317. John Wiley and Sons, Inc., 2011.

[19] R. O. Schmidt, “Multiple emitter location and signal parameter estimation,” *Antennas and Propagation, IEEE Transactions on*, vol. 34, no. 3, pp. 276–280, 1986.

[20] P. Stoica and A. Nehorai, “MUSIC, maximum likelihood, and cramer-rao bound: further results and comparisons,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 12, pp. 2140–2150, 1990.

[21] D. A. Linebarger, R. D. DeGroat, and E. M. Dowling, “Efficient direction-finding methods employing forward/backward averaging,” *Signal Processing, IEEE Transactions on*, vol. 42, no. 8, pp. 2136–2145, 1994.

[22] J. T. Mayhan and L. Niro, “Spatial spectral estimation using multiple beam antennas,” *Antennas and Propagation, IEEE Transactions on*, vol. 35, no. 8, pp. 897–906, 1987.

[23] A. Barabell, “Improving the resolution performance of eigenstructure-based direction-finding algorithms,” in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP ’83.*, vol. 8, pp. 336–339.

[24] B. D. Rao and K. V. S. Hari, “Performance analysis of root-music,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 12, pp. 1939–1949, 1989.

[25] C. P. Mathews and M. D. Zoltowski, “Eigenstructure techniques for 2-D angle estimation with uniform circular arrays,” *Signal Processing, IEEE Transactions on*, vol. 42, no. 9, pp. 2395–2407, 1994.

[26] M. Costa, A. Richter, F. Belloni, and V. Koivunen, “Polynomial rooting-based direction finding for arbitrary array configurations,” in *Sensor Array and Multichannel Signal Processing Workshop, 2008. SAM 2008. 5th IEEE*, pp. 58–62.

[27] J. P. Burg, “Maximum entropy spectral analysis,” in *37th Annual International Meeting.*, Society of Exploration Geophysics.

[28] R. Roy and T. Kailath, “ESPRIT-estimation of signal parameters via rotational invariance techniques,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 7, pp. 984–995, 1989.

[29] S. M. Kuo, B. H. Lee, and W. Tian, *Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications*. John Wiley and Sons, 2013.

[30] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, and A. McKenney, “LAPACK user’s guide SIAM,” 1999.

[31] F. T. L. R. P. Brent and C. V. Loan, "Computation of the singular value decomposition using mesh-connected processors," 1985.

[32] A. Ahmedsaid, A. Amira, and A. Bouridane, "Improved SVD systolic array and implementation on FPGA," in *Field-Programmable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on*, pp. 35–42.

[33] J. M. McNamee, "A bibliography on roots of polynomials," *Journal of Computational and Applied Mathematics*, vol. 47, no. 3, pp. 391–394, 1993.

[34] J. M. McNamee and V. Pan, *Numerical methods for roots of polynomials*, vol. 16. Newnes, 2013.

[35] A. Edelman and H. Murakami, "Polynomial roots from companion matrix eigenvalues," *Mathematics of Computation*, vol. 64, no. 210, pp. 763–776, 1995.

[36] V. Y. Pan, "Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding," *Journal of Symbolic Computation*, vol. 33, no. 5, pp. 701–733, 2002.

[37] J. Hubbard, D. Schleicher, and S. Sutherland, "How to find all roots of complex polynomials by newton's method," *Inventiones mathematicae*, vol. 146, no. 1, pp. 1–33, 2001.

[38] H. Andrews and C. Patterson, “Singular value decompositions and digital image processing,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 24, no. 1, pp. 26–53, 1976.

[39] A. Ahmedsaid and A. Amira, “Accelerating SVD on reconfigurable hardware for image denoising,” in *Image Processing, 2004. ICIP ’04. 2004 International Conference on*, vol. 1, pp. 259–262 Vol. 1.

[40] M. Rahmati, M. S. Sadri, and M. A. Naeini, “FPGA based singular value decomposition for image processing applications,” in *Application-Specific Systems, Architectures and Processors, 2008. ASAP 2008. International Conference on*, pp. 185–190.

[41] Y. L. Chen, C. Z. Zhan, T. J. Jheng, and A. Y. Wu, “Reconfigurable adaptive singular value decomposition engine design for high-throughput MIMO-OFDM systems,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 747–760, 2013.

[42] J. Lofgren, S. Mehmood, N. Khan, B. Masood, M. Awan, I. Khan, N. A. Chisty, and P. Nilsson, “Hardware implementation of an SVD based MIMO OFDM channel estimator,” in *NORCHIP, 2009*, pp. 1–4.

[43] W. Yue, K. Cunningham, P. Nagvajara, and J. Johnson, “Singular value decomposition hardware for MIMO: State of the art and custom design,” in

*Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, pp. 400–405.

[44] S. K. Jha and R. D. S. Yadava, "Denoising by singular value decomposition and its application to electronic nose data processing," *Sensors Journal, IEEE*, vol. 11, no. 1, pp. 35–44, 2011.

[45] A. Said, T. Kalker, L. Bowon, and M. Fozunbal, "Massively parallel processing of signals in dense microphone arrays," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 3080–3083.

[46] T. L. Marzetta, "How much training is required for multiuser MIMO?," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pp. 359–363.

[47] H. A. Suraweera, N. Hien Quoc, T. Q. Duong, Y. Chau, and E. G. Larsson, "Multi-pair amplify-and-forward relaying with very large antenna arrays," in *Communications (ICC), 2013 IEEE International Conference on*, pp. 4635–4640.

[48] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1996.

[49] J. H. Wilkinson, "Note on the quadratic convergence of the cyclic jacobi process," *Numerische Mathematik*, vol. 4, no. 1, pp. 296–300, 1962.

[50] J. Demmel and V. K., “Jacobi’s method is more accurate than QR,” *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1204–1245, 1992.

[51] G. E. F. Henrici and P., “The cyclic jacobi method for computing the principal values of a complex matrix,” *Trans. Amer. Math. Soc*, 1958.

[52] F. T. Luk and P. Haesun, “A proof of convergence for two parallel Jacobi SVD algorithms,” *Computers, IEEE Transactions on*, vol. 38, no. 6, pp. 806–811, 1989.

[53] M. R. Hestenes, “Inversion of matrices by biorthogonalization and related results,” vol. 6, pp. 51–90, Mar. 1958.

[54] J. R. Cavallaro and F. T. Luk, “CORDIC arithmetic for an SVD processor,” in *Computer Arithmetic (ARITH), 1987 IEEE 8th Symposium on*, pp. 113–120.

[55] B. Yang and J. BÃűhme, “Reducing the computations of the singular value decomposition array given by Brent and Luk,” *SIAM Journal on Matrix Analysis and Applications*, vol. 12, no. 4, pp. 713–725, 1991.

[56] N. D. Hemkumar and J. R. Cavallaro, “A systolic VLSI architecture for complex SVD,” in *Circuits and Systems, 1992. ISCAS ’92. Proceedings., 1992 IEEE International Symposium on*, vol. 3, pp. 1061–1064 vol.3.

[57] M. Weiwei, M. E. Kaye, D. M. Luke, and R. Doraiswami, “An FPGA-based singular value decomposition processor,” in *Electrical and Computer Engineering, 2006. CCECE ’06. Canadian Conference on*, pp. 1047–1050.

[58] Xilinx, *Virtex-6 FPGA DSP48E1 Slice-User Guide*, 2011.

[59] H. Javaid, X. He, A. Ignjatovic, and S. Parameswaran, "Optimal synthesis of latency and throughput constrained pipelined MPSoCs targeting streaming applications," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*, pp. 75–84, Oct 2010.

[60] Xilinx, *System Generator for DSP- User Guide*, 2012.

[61] Xilinx, *System Generator for DSP-Getting Started Guide*, 2011.

[62] Xilinx, *LogiCORE IP CORDIC v 4.0 -Product Specification*, 2011.

[63] Y. Bresler and A. Macovski, "Exact maximum likelihood parameter estimation of superimposed exponential signals in noise," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 5, pp. 1081–1089, 1986.

[64] L. C. Zhao, P. R. Krishnaiah, and Z. D. Bai, "On detection of the number of signals in presence of white noise," *Journal of Multivariate Analysis*, vol. 20, no. 1, pp. 1–25, 1986.

[65] Z. Bai, P. R. Krishnaiah, and L. Zhao, "On the direction of arrival estimation," report, DTIC Document, 1987.

[66] J. Selva, "Computation of spectral and root MUSIC through real polynomial rooting," *Signal Processing, IEEE Transactions on*, vol. 53, no. 5, pp. 1923–1927, 2005.

[67] J. M. McNamee and V. Pan, *Numerical methods for roots of polynomials*, vol. 16. Newnes, 2013.

[68] D. A. Bini, L. Gemignani, and V. Y. Pan, "Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations," *Numerische Mathematik*, vol. 100, no. 3, pp. 373–408, 2005.

[69] D. A. Bini, L. Gemignani, and V. Y. Pan, "Improved initialization of the accelerated and robust QR-like polynomial root-finding," *Electronic Transactions on Numerical Analysis*, vol. 17, pp. 195–205, 2004.

[70] C. A. Neff and J. H. Reif, "An efficient algorithm for the complex roots problem," *Journal of Complexity*, vol. 12, no. 2, pp. 81–115, 1996.

[71] S. A. Zekavat, A. Kolbus, Y. Xiaofeng, W. Zhonghai, J. Pourrostam, and M. Pourkhaatoun, "A novel implementation of DOA estimation for node localization on software defined radios: Achieving high performance with low complexity," in *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, pp. 983–986.

[72] W. T. Shaw, *Complex analysis with Mathematica*, vol. 1. Cambridge University Press, 2006.

[73] M. McClure, "Newton's method for complex polynomials," *A preprint version of a,"Mathematical Graphics" column from Mathematica in Education and Research*, pp. 1–15, 2006.

[74] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, “Invited paper: Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of xilinx FPGAs,” in *Field Programmable Logic and Applications, 2006. FPL’06. International Conference on*, pp. 1–6, IEEE.

[75] N. Mackey, “Hamilton and jacobi meet again: Quaternions and the eigenvalue problem,” *SIAM journal on matrix analysis and applications*, vol. 16, no. 2, pp. 421–435, 1995.

[76] F. Belloni, A. Richter, and V. Koivunen, “Doa estimation via manifold separation for arbitrary array structures,” *Signal Processing, IEEE Transactions on*, vol. 55, no. 10, pp. 4800–4810, 2007.

[77] L. Hong-bing, G. Yi-duo, and G. Jian, “Computational efficient DOA estimation algorithm based on MSWF and polynomial rooting,” in *Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on*, pp. 672–675.

[78] J. M. McNamee, “An updated supplementary bibliography on roots of polynomials,” *Journal of Computational and Applied Mathematics*, vol. 110, no. 2, pp. 305–306, 1999.

[79] K. V. Rangarao and S. Venkatanarasimhan, “gold-MUSIC: A variation on MUSIC to accurately determine peaks of the spectrum,” *Antennas and Propagation, IEEE Transactions on*, vol. 61, no. 4, pp. 2263–2268, 2013.

[80] H. Rohling, M.-M. Meinecke, K. Mott, and L. Urs, “Research activities in automotive radar,” in *Physics and Engineering of Millimeter and Sub-Millimeter Waves, 2001. The Fourth International Kharkov Symposium on*, vol. 1, pp. 48–51, IEEE.

[81] K. Saneyoshi, “Drive assist system using stereo image recognition,” in *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, pp. 230–235, IEEE.

[82] A. Ahmedsaid, A. Amira, and A. Bouridane, “Accelerating MUSIC method on reconfigurable hardware for source localisation,” in *Circuits and Systems, 2004. ISCAS ’04. Proceedings of the 2004 International Symposium on*, vol. 3, pp. III–369–72 Vol.3.

[83] C. Bobda and N. Steenbock, “Singular value decomposition on distributed reconfigurable systems,” in *Rapid System Prototyping, 12th International Workshop on, 2001.*, pp. 38–43.

[84] I. Bravo, P. Jimenez, M. Mazo, J. L. Lazaro, and A. Gardel, “Implementation in FPGA of Jacobi method to solve the eigenvalue and eigenvector problem,” in *Field Programmable Logic and Applications, 2006. FPL ’06. International Conference on*, pp. 1–4.

[85] S. Chi-Chia and J. Gotze, “VLSI circuit design concept for parallel iterative algorithms in nanoscale,” in *Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on*, pp. 688–692.

[86] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, “Wordlength optimization for linear digital signal processing,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 10, pp. 1432–1442, 2003.

[87] J. Coyne, D. Cyganski, and R. J. Duckworth, “FPGA-based co-processor for singular value array reconciliation tomography,” in *Field-Programmable Custom Computing Machines, 2008. FCCM ’08. 16th International Symposium on*, pp. 163–172.

[88] Z. Drmac, “Implementation of jacobi rotations for accurate singular value computation in floating point arithmetic,” *SIAM Journal on Scientific Computing*, vol. 18, no. 4, pp. 1200–1222, 1997.

[89] Z. Drmac and K. Veselic, “New fast and accurate jacobi SVD algorithm. i,” *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1322–1342, 2008.

[90] Z. Drmac and K. Veselic, “New fast and accurate jacobi SVD algorithm. II,” *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1343–1362, 2008.

[91] P. M. Gatis Valters1, “Automation of FPGA implementation of unitary transforms based on elementary generalized unitary rotation,” 2011.

[92] C. Kotas and J. Barhen, “Singular value decomposition utilizing parallel algorithms on graphical processors,” in *OCEANS 2011*, pp. 1–7.

[93] H. Kuan-Ju, S. Wei-Yeh, C. Jui Chung, F. Chih Wei, and F. Wai-Chi, “A pipeline VLSI design of fast singular value decomposition processor for real-time EEG

system based on on-line recursive independent component analysis," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pp. 1944–1947.

[94] H. Kyungtae and B. L. Evans, "Wordlength optimization with complexity-and-distortion measure and its application to broadband wireless demodulator design," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 5, pp. V–37–40 vol.5.

[95] H. Kyungtae, A. G. Olson, and B. L. Evans, "Automatic floating-point to fixed-point transformations," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pp. 79–83.

[96] N. Le Bihan and S. J. Sangwine, "Jacobi method for quaternion matrix singular value decomposition," *Applied Mathematics and Computation*, vol. 187, no. 2, pp. 1265–1271, 2007.

[97] L. M. Ledesma-Carrillo, E. Cabal-Yepez, R. de J Romero-Troncoso, A. Garcia-Perez, R. A. Osornio-Rios, and T. D. Carozzi, "Reconfigurable FPGA-based unit for singular value decomposition of large m x n matrices," in *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pp. 345–350.

[98] N. Mackey, “Hamilton and Jacobi meet again: Quaternions and the eigenvalue problem,” *SIAM J. Matrix Anal. Appl.*, vol. 16, no. 2, pp. 421–435, 1995.

[99] K. Natarajan, S. Arun, K. Murugaraj, and M. John, “An application specific matrix processor for signal subspace based speech enhancement in noise robust speech recognition applications,” in *ASIC, 2007. ASICON ’07. 7th International Conference on*, pp. 766–769.

[100] P. Soo-Chang, C. Ja-Han, and D. Jian-Jiun, “Quaternion matrix singular value decomposition and its applications for color image processing,” in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 1, pp. I–805–8 vol.1.

[101] H. H. Volker Strumpen and A. Agarwal, “A stream algorithm for the SVD,” 2003.

[102] B. B. Zhou and R. P. Brent, “On parallel implementation of the one-sided Jacobi algorithm for singular value decompositions,” in *Parallel and Distributed Processing, 1995. Proceedings. Euromicro Workshop on*, pp. 401–408.

[103] D. A. Bini, F. Daddi, and L. Gemignani, “On the shifted QR iteration applied to companion matrices,” *Electronic Transactions on Numerical Analysis*, vol. 18, pp. 137–152, 2004.

[104] D. A. Bini, V. Mehrmann, V. Olshevsky, E. E. Tyrtyshnikov, and M. van Barel, *Numerical methods for structured matrices and applications*, vol. 199. Springer, 2010.

[105] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, *A Fast QR Algorithm for Companion Matrices*, vol. 179 of *Operator Theory: Advances and Applications*, book section 7, pp. 111–143. BirkhÃd'user Basel, 2008.

[106] N. Dowlut and A. Manikas, "A polynomial rooting approach to super-resolution array design," *Signal Processing, IEEE Transactions on*, vol. 48, no. 6, pp. 1559–1569, 2000.

[107] A. Edelman and H. Murakami, "Polynomial roots from companion matrix eigenvalues," *Mathematics of Computation*, vol. 64, no. 210, pp. 763–776, 1995.

[108] B. Friedlander, "The root-MUSIC algorithm for direction finding with interpolated arrays," *Signal Processing*, vol. 30, no. 1, pp. 15–29, 1993.

[109] G. F. Hatke and K. W. Forsythe, "A class of polynomial rooting algorithms for joint azimuth/elevation estimation using multidimensional arrays," in *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, vol. 1, pp. 694–699 vol.1.

[110] B. Kalantari, "Voronoi diagrams and polynomial root-finding," in *Voronoi Diagrams, 2009. ISVD '09. Sixth International Symposium on*, pp. 31–40.

[111] S. G. Krantz, S. Kress, and R. Kress, *Handbook of complex variables*. Springer, 1999.

[112] M. Lang and B. C. Frenzel, “Polynomial root finding,” *Signal Processing Letters, IEEE*, vol. 1, no. 10, pp. 141–143, 1994.

[113] E. Lin, L. Bai, and M. Kam, “Efficient DOA estimation method employing unitary improved polynomial rooting,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP ’04). IEEE International Conference on*, vol. 2, pp. ii–257–60 vol.2.

[114] J. McKee, J. F. McKee, and C. Smyth, *Number theory and polynomials*. Cambridge University Press, 2008.

[115] J. M. McNamee, “An updated supplementary bibliography on roots of polynomials,” *J. Comput. Appl. Math.*, vol. 110, no. 2, pp. 305–306, 1999.

[116] J. M. McNamee and V. Pan, *Numerical methods for roots of polynomials*, vol. 16. Newnes, 2013.

[117] Z. Ming, Y. Li, and Z. Yongquan, “Hybrid evolution strategies for simultaneous solving all real roots of polynomial,” in *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, vol. 2, pp. 283–286.

[118] M. Rubsamen and A. B. Gershman, “Direction-of-arrival estimation for nonuniform sensor arrays: From manifold separation to fourier domain MUSIC methods,” *Signal Processing, IEEE Transactions on*, vol. 57, no. 2, pp. 588–599, 2009.

[119] D. Schleicher, “On the number of iterations of Newton’s method for complex polynomials,” *Ergodic Theory and Dynamical Systems*, vol. 22, no. 3, pp. 935–945, 2002.

[120] A. J. Weiss and B. Friedlander, “Direction finding for diversely polarized signals using polynomial rooting,” in *Signals, Systems and Computers, 1991. 1991 Conference Record of the Twenty-Fifth Asilomar Conference on*, pp. 287–289 vol.1.

[121] C. Xiang, X. Jingmin, and Y. Nishio, “Real polynomial form of music for uniform linear array,” in *Signal Processing Systems (SiPS), 2013 IEEE Workshop on*, pp. 366–370.

[122] F. G. Yan, M. Jin, and X. Qiao, “Low-complexity DOA estimation based on compressed MUSIC and its performance analysis,” *Signal Processing, IEEE Transactions on*, vol. 61, no. 8, pp. 1915–1930, 2013.

[123] R. Zekavat and R. M. Buehrer, *Handbook of position location: theory, practice and advances*, vol. 27. John Wiley and Sons, 2011.

[124] Y. Zhang and Z.-z. Zeng, “A new method for simultaneous extraction of all roots of algebraic polynomial,” in *Computational Intelligence and Security, 2009. CIS ’09. International Conference on*, vol. 1, pp. 197–200.

[125] F. Uhlig, “The  cal dqr algorithm, basic theory, convergence, and conditional stability,” *Numerische Mathematik*, vol. 76, no. 4, pp. 515–553, 1997.

# Appendix A

# Timing diagrams for proposed design and BLV array

This appendix contains the detailed timing diagrams for BLV array and proposed array. The latency calculations and a brief timing diagram were presented in Section 2.4.3. The diagrams here are only for reference for designer who want to get a better idea of how proposed array differs from the traditional BLV array in terms of timing.

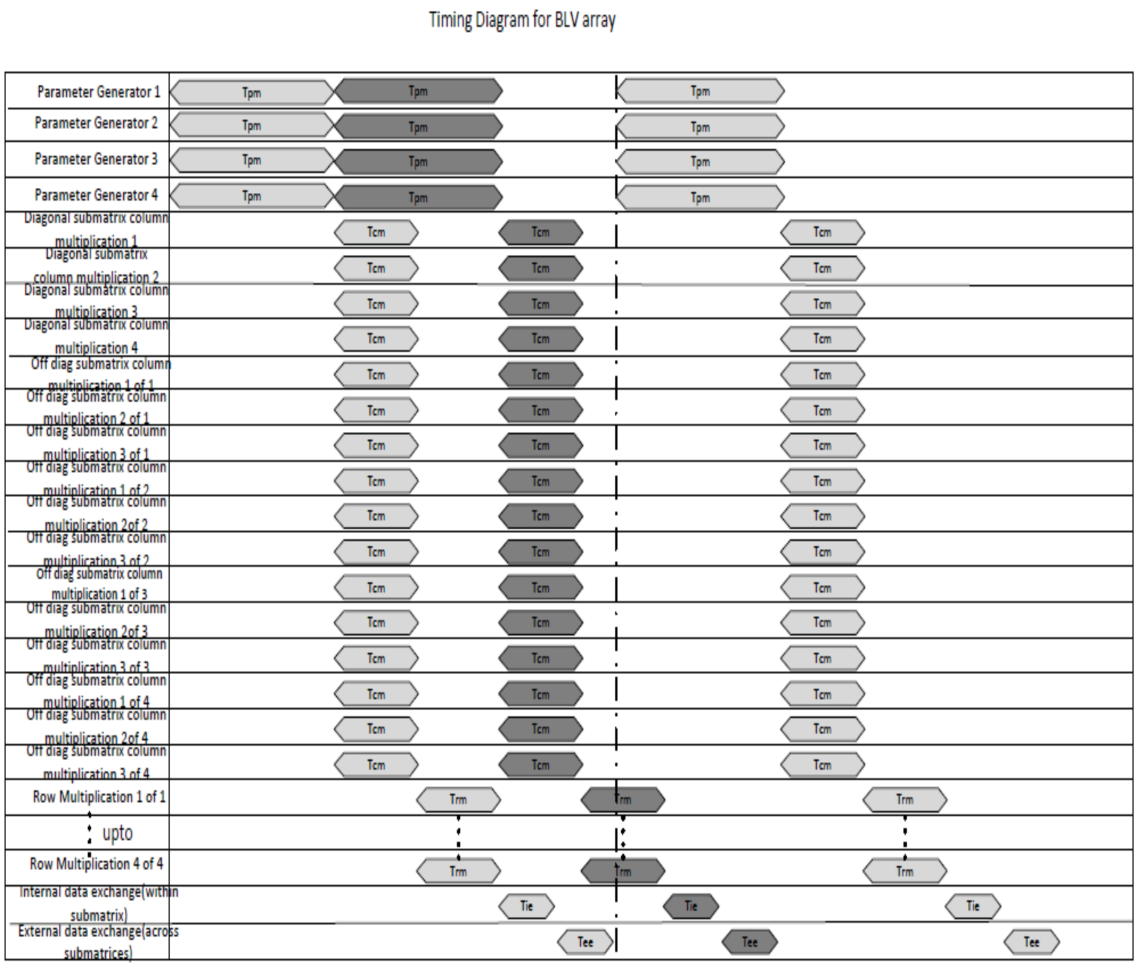

Timing Diagram for BLV array
Parameter Generator 1
Parameter Generator 2
Parameter Generator 3
Parameter Generator 4
Diagonal submatrix column multiplication 1
Diagonal submatrix column multiplication 2
Diagonal submatrix column multiplication 3
Diagonal submatrix column multiplication 4
Off diag submatrix column multiplication 1 of 1
Off diag submatrix column multiplication 2 of 1
Off diag submatrix column multiplication 3 of 1
Off diag submatrix column multiplication 1 of 2
Off diag submatrix column multiplication 2of 2
Off diag submatrix column multiplication 3 of 2
Off diag submatrix column multiplication 1 of 3
Off diag submatrix column multiplication 2of 3
Off diag submatrix column multiplication 3 of 3
Off diag submatrix column multiplication 1 of 4
Off diag submatrix column multiplication 2of 4
Off diag submatrix column multiplication 3 of 4
Row Multiplication 1 of 1
upto
Row Multiplication 4 of 4
Internal data exchange(within submatrix)
External data exchange(across submatrices)
Tpm
Tcm
Trm
Tie
Tee

**Figure A.1:** Detailed timing diagrams for BLV array

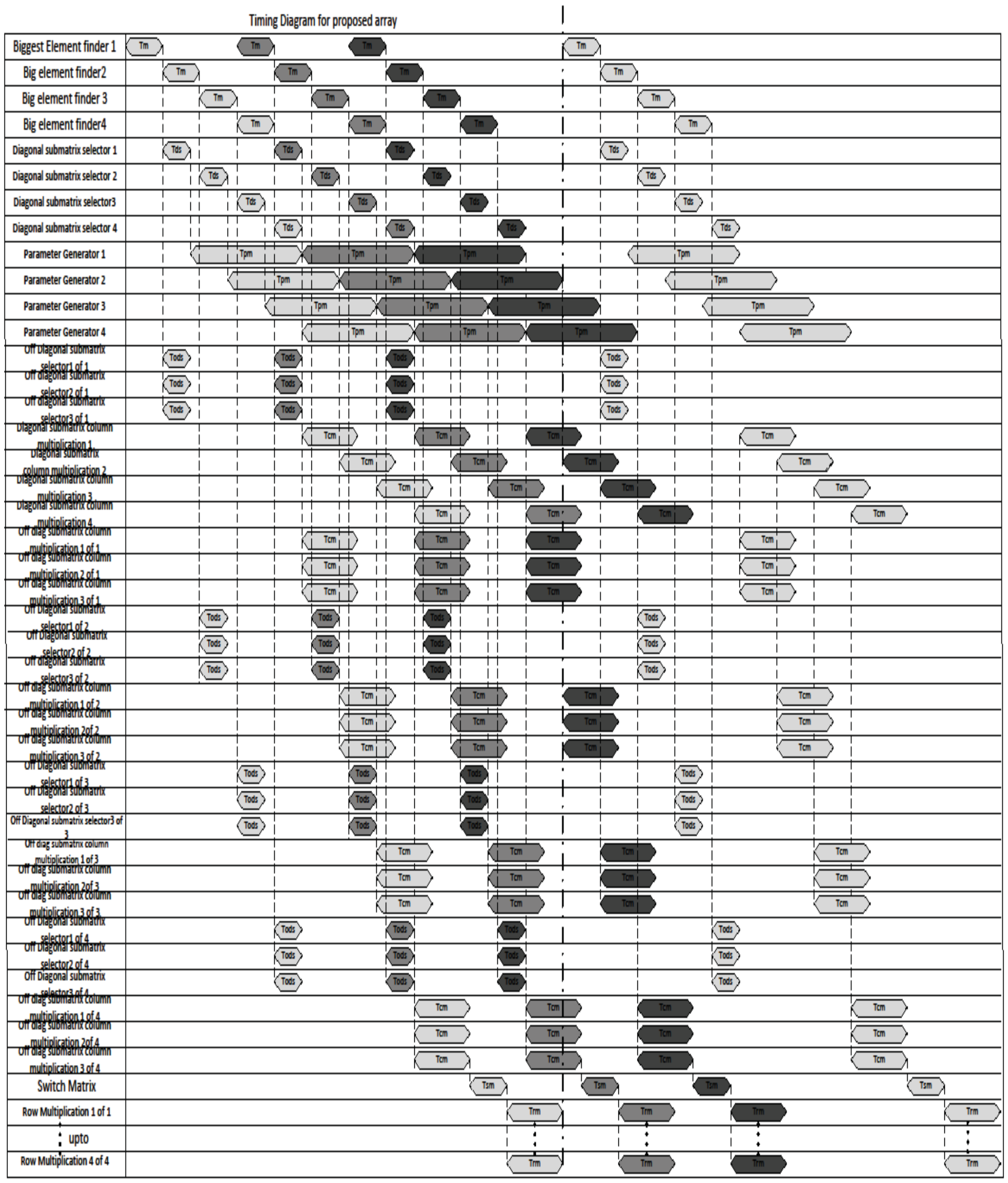

Timing Diagram for proposed array
Biggest Element finder 1
Big element finder2
Big element finder 3
Big element finder4
Diagonal submatrix selector 1
Diagonal submatrix selector 2
Diagonal submatrix selector3
Diagonal submatrix selector 4
Parameter Generator 1
Parameter Generator 2
Parameter Generator 3
Parameter Generator 4
Off Diagonal submatrix selector1 of 1
Off diagonal submatrix selector2 of 1
Off diagonal submatrix selector3 of 1
Diagonal submatrix column multiplication 1
Diagonal submatrix column multiplication 2
Diagonal submatrix column multiplication 3
Diagonal submatrix column multiplication 4
Off diag submatrix column multiplication 1 of 1
Off diag submatrix column multiplication 2 of 1
Off diag submatrix column multiplication 3 of 1
Off Diagonal submatrix selector1 of 2
Off Diagonal submatrix selector2 of 2
Off diagonal submatrix selector3 of 2
Off diag submatrix column multiplication 1 of 2
Off diag submatrix column multiplication 2of 2
Off diag submatrix column multiplication 3 of 2
Off Diagonal submatrix selector1 of 3
Off Diagonal submatrix selector2 of 3
Off Diagonal submatrix selector3 of 3
Off diag submatrix column multiplication 1 of 3
Off diag submatrix column multiplication 2of 3
Off diag submatrix column multiplication 3 of 3
Off Diagonal submatrix selector1 of 4
Off Diagonal submatrix selector2 of 4
Off Diagonal submatrix selector3 of 4
Off diag submatrix column multiplication 1 of 4
Off diag submatrix column multiplication 2of 4
Off diag submatrix column multiplication 3 of 4
Switch Matrix
Row Multiplication 1 of 1
upto
Row Multiplication 4 of 4
Tm
Tds
Tpm
Tods
Tcm
Tsm
Trm

**Figure A.2:** Detailed timing diagrams for proposed design and BLV array