A reduced complexity parallel least mean square algorithm for adaptive beamforming

Ghattas Akkad

DOCTORAT BRETAGNE LOIRE / MATHSTIC

ENSTA BRETAGNE

THÈSE DE DOCTORAT DE

ENSTA Bretagne
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *(voir liste des spécialités)*

Par
Ghattas AKKAD

A Reduced Complexity Parallel Least Mean Square Algorithm for Adaptive Beamforming

Thèse présentée et soutenue à Brest, le 7 Decembre 2020
Unité de recherche : Lab-STICC, UMR 6285
Thèse N° :

Rapporteurs avant soutenance :

Karim ABED-MERAIM — Professor à l'Université d'Orléans
Nadège THIRION-MOREAU — Professor à l'Université de Toulon

Composition du Jury :

Président :	Christian JUTTEN	Professor à l'Université Grenoble Alpés
Examinateur :	Emanuel RADOI	Professor à l'Université Bretagne Occidentale
Dir. de thèse :	Ali MANSOUR	Professor à l'ENSTA Bretagne
Co-dir. de thèse :	Bachar ELHASSAN	Professor à l'Univérsité Libanaise

Invité(s) :

Stephane AZZOU — Professor à l'École Nationale d'Ingénieurs de Brest
Elie INATY — Associate Professor à l'Université du Balamand
Benoit ZERR — Professor à l'ENSTA Bretagne

ACKNOWLEDGEMENT

First and foremost I would like to express my deepest gratitude for my thesis supervisor Professor Ali MANSOUR, for without his unending guidance, mentoring, support, and warm hospitality this work would have never come to light. I would like to also thank my thesis co-supervisor Professor Bachar ELHASSAN for the similar and for his continuous encouragment.

Second, I would like to thank the jury committee president Professor Christian JUTTEN and the committee members: Professor Karim ABED-MERIAM, Professor Stephane AZZOU, Associate Professor Elie INATY, Professor Emanuel RADOI, Professor Nadège THIRION-MOREAU and Professor Benoit ZERR. It is an honor to have my work examined and assessed by professional experts such as yourselves.

Third, I am grateful to AID - DGA (l'Agence de l'Innovation de Defense - Direction Générale de l'Armement – Ministére des Armées) & ANR (Agence Nationale de la Recherche en France) for supporting our ANR-ASTRID – Project (ANR-19-ASTR-0005-03).

Moreover, I would like to heartily thank my mentors, colleagues and best friends at the University of Balamand (UOB) and in Lebanon: Associate Professor Rafic AYOUBI, Engineer Nayla Greige BALESH, Engineer Nassif DAOUD, Associate Professor Nabil KARAMI and Assistant Professor Mohamad NAJEM. A special thank you for UOB's Vice President and Dean of the Faculty of Engineering (FOE) Professor Rami ABBOUD for his encouragement and support. Indeed your hard work and dedication toward the advancement of the UOB and the FOE, despite all the unaccounted for challenges, is inspiring and motivational. Eventhought I did not get the chance to work with you, I am, without a single doubt, definite that the FOE will thrive under your guidance and supervision.

Additionally, I would like to thank the previous Vice President and previous Dean of the UOB Faculty of Engineering Professor Michel NAJJAR for his limitless support. It

has been an honour and a privilege to have studied and worked in a university and a faculty that you have devoted your life to build. It is with your effort and dedication that it has reached its international exposure and distinguished reputation. It has become a second home for the many and has taught us that home is not made of bricks and stones but where the heart and mind belongs.

I would like to, also, express my unending and immeasurable appreciation and gratitude for Mrs Mimo NAKAD, my eternal best friend and my source of boundless motivation. She is an enthusiastic, dedicated and wholehearted person, a true star. From her I learned that, quoting Gibran Khalil Gibran, "You give but little when you give of your possessions. It is when you give of yourself that you truly give".

Finally, I would like to thank my mother Mirana Abed AKKAD, my father Michel AKKAD, my aunt Kety Akkad SAWAYA, my sister Mariam AKKAD and my brothers Charbel and Roufaeil AKKAD, for without them I am nothing.

Table of Contents

List of Figures

List of Tables

NOMENCLATURE

List of Abbreviations

AD	Analog to Digital
AOA	Angle of Arrival
AP	Array Processing
ARM	Advanced RISC Machines
BRAM	Block Random Access Memory
CAN	Controller Area Network
CAWGN	Complex Additive White Gaussian Noise
CLB	Configurable Logic Block
CM	Constant Modulus
CORDIC	Coordinate Rotation Digital Computer
DAC	Digital to Analog Converter
DFT	Discrete Fourier Transform
DIF	Decimation in Frequency
DILFAST	Decoupled Iterative Least Squares Finite Alphabet Space-Time
DIT	Decimation in Time
DMA	Direct Memory Access
DOA	Direction of Arrival
DoF	Degrees of Freedom

DRC-pLMS	Delayed Reduced Complexity Parallel Least Mean Square
EF-LSL	Error-Feedback LSL
ESPRIT	Estimation of Signal Parameter Via Rotational Invariance Techniques
EVD	Eigenvalue Decomposition
FA	Finite Alphabet
FIR	Finite Input Response
FM	Fourrier Method
FNBW	First Null Beam width
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GPP	General Purpose Processor
GS-FAP	Gauss-Seidel Fast Affine Projection
HLL	High Level Language
HLS	High Level Synthesis
HoCA	Higher Order Cumulant
HPBW	Half Power Beam width
I2C	Inter-Integrated Circuit
IF	Intermediate Frequency
IIR	Infinite Input Response
IOB	Input/Output Block
IP-core	Intellectual Property
LC	Linear Combiner
LLMS	Least Mean Square - Least Mean Square

LMS	Least Mean Square
LS	Least Squares
LUT	Look Up Table
MAC	Multiply Accumulate
ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
MRVSS	Modified Robust Variable Step Size
MSE	Mean Square Error
MSINR	Maximum Signal to Interference Plus Noise Ratio
MUSIC	Multiple Signal Classification
MVDR	Minimum Variance Distortionless Response
NLMS	Normalised Least Mean Square
PDSP	Programmable Digital Signal Processing Processor
PLL	Phased Locked Loop
pLMS	Parallel Least Mean Square
QoS	Quality of Service
RAM	Random Access Memory
RC-pLMS	Reduced Complexity Parallel Least Mean Square
RF	Radio Frequency
RISC	Reduced Instruction Set Computer
RLMS	Recursive Least Mean Square
RLS	Recursive Least Squares
RTL	Register Transfer Level

SGD	Stochastic Gradient Descent
SLL	Side Lobe Level
SNR	Signal to Noise Ratio
SoC	System on Chip
SOCP	Second Order Cone Programming
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Reciever-Transmitter
USB	Universal Serial Bus
VSSLMS	Variable Step Size Least Mean Square
WSS	Wide Sense Stationary

List of Variables

$(AF(\theta))_n$	Array Factor Normalized at Unity
α	RLS Forgetting Factor
η_{Sy}	Multi-Stage Overall Desired and Output Signals Quantization Error
η_S	Multi-Stage Overall Desired Signals Quantization Error
η_u	Multi-Stage Overall Input Signals Quantization Error
η_w	Adaptive Filter Weight Signals Quantization Error
η_y	Multi-Stage Output Signals Quantization Error
$\hat{a}_{d,m}$	m^{th} Element of the Desired Signals Steering Vector Estimate
λ	Carrier Signals Wavelength
$\lambda_{A,max}$	Maximum Eigenvalue in Matrix $\mathbf{A}$
λ_i	i^{th} Eigenvalues
$\lambda_{R,max}$	Maximum Eigenvalue in Matrix $\mathbf{R}$
$\bar{\mathbf{v}}(k)$	Mean Coefficient Error Vector
$\bar{\mathbf{w}}(k)$	Adaptive Filters Mean Weight Vector
$\hat{\mathbf{a}}_d$	Desired Signals Steering Vector Estimate
$\widehat{\mathbf{R}}_{i+n}(\tau)$	Interference Plus Noise Correlation Matrix Estimate
$\mathbf{A}$	Multi Stage Adaptive Algorithm Overall Input Auto-Correlation Matrix
$\mathbf{a}_d$	Desired Signals Steering Vector
$\mathbf{a}_{i,l}$	l^{th} Interference Signals Steering Vector
$\mathbf{A}_i$	Matrix of Steering Vectors
$\mathbf{c}_i$	Beamformer Constraint Vector
$\mathbf{d}$	Desired Signal Vector

$\mathbf{H}_b$	Communication Channel Transfer Function
$\mathbf{I}$	Identity Matrix
$\mathbf{i}_l$	l^{th} Interference Signal
$\mathbf{L}(k)$	RLS Gain Matrix
$\mathbf{m}(k)$	Mean Coefficient Error Vector in a Rotated Coordinate System
$\mathbf{n}(k)$	Complex Additive White Gaussian Noise Vector Signal
$\mathbf{O}$	Unitary Matrix of Eigenvectors Rows
$\mathbf{p}(\tau)$	Adaptive Filters Input and Desired Signals Cross Correlation Vector
$\mathbf{p}_t$	FFT Rotation, Twiddle, Factor Exponent Input Vector
$\mathbf{Q}(\tau)$	RLS Input Signals Auto-Correlation Matrix
$\mathbf{Q}_r(\tau)$	RC-pLMS Input Signals Cross-Correlation Matrix
$\mathbf{R}(\tau)$	Adaptive Filters Input Signals Auto-Correlation Matrix
$\mathbf{R}_{i+n}(\tau)$	Interference Plus Noise Correlation Matrix
$\mathbf{S}_b$	Unknown Data Signal
$\mathbf{u}(k)$	RC-pLMS Total Input Vector
$\mathbf{u}_q(k)$	RC-pLMS Total Input Vector Subject to Quantization Error
$\mathbf{w}(k)$	Adaptive Filters Weight Vector
$\mathbf{w}_{1..4}(k)$	Adaptive Filters Weight Vector With Respect to First Four Input Samples
$\mathbf{w}_1(k)$	First Stage Adaptive Filters Weight Vector
$\mathbf{w}_2(k)$	Second Stage Adaptive Filters Weight Vector
$\mathbf{w}_{5..8}(k)$	Adaptive Filters Weight Vector With Respect to Last Four Input Samples
$\mathbf{w}_i(k)$	i^{th} Stage Adaptive Filters Weight Vector
$\mathbf{w}_{MVDR}$	MVDR Adaptive Beamformers Optimal Weight Vector

$\mathbf{w}_{oplms}$	LMS Adaptive Beamformers Optimal Weight Vector
$\mathbf{w}_{opr}$	RC-pLMS Adaptive Beamformers Optimal Weight Vector
$\mathbf{w}_{op}$	pLMS Adaptive Beamformers Optimal Weight Vector
$\mathbf{w}_q$	RC-pLMS Weight Vector Subject to Quantization Error
$\mathbf{x}(k)$	Adaptive Filters Input Signal Vector
$\mathbf{x}_{1..4}(k)$	First Four Samples of the Adaptive Filters Input Signal Vector
$\mathbf{x}_1(k)$	First Stage Adaptive Filters Input Signal Vector
$\mathbf{x}_2(k)$	Second Stage Adaptive Filters Input Signal Vector
$\mathbf{x}_{5..8}(k)$	Last Four Samples of the Adaptive Filters Input Signal Vector
$\mathbf{X}_b$	Input Signals Matrix
$\mathbf{x}_d$	Discrete Time FFT Input Signal Vector
$\mathbf{x}_f$	Adaptive Filters Input Signal Vector Subject to Fractional Delay
$\mathbf{z}(\tau)$	RLS Input and Desired Signals Cross Correlation Vector
$\mathbf{z}_r(\tau)$	RC-pLMS Input and Desired Signals Cross Correlation Vector at Different τ
μ	RC-pLMS Optimization Step Size
μ_1	First LMS Stage Optimization Step Size
μ_2	Second LMS Stage Optimization Step Size
μ_{LMS}	LMS Optimization Step Size
μ_{pLMS}	pLMS Optimization Step Size
∇_{LMS}	LMS Cost Functions Gradient
∇_{pLMS}	pLMS Cost Functions Gradient
$\nabla_{RC-pLMS}$	RC-pLMS Cost Functions Gradient

ϕ	Antenna Arrays Azimuth Angle of Arrival
ψ	Phase Shift
$\Psi(\varepsilon)$	Set of Erroneous Steering Vectors With Respect to ε
ψ_x	Phase Shift for Elements Placed in the x Plane
ψ_y	Phase Shift for Elements Placed in the y Plane
σ_a^2	Variance of the Constant a
σ_d^2	Desired Signals Variance
σ_p^2	In Phase Message Signals Variance
σ_q^2	Out Of Phase Message Signals Variance
τ	Signal Lag in a Wide Sense Stationary Process
τ_d	Time Delay Between Two Consecutive Antenna Elements
θ	Antenna Arrays Elevation Angle of Arrival
ε	Maximum Mismatch Error
Λ	Diagonal Matrix of Eigenvalues λ_i
ϑ	Very Small User Introduced Constant
ξ_{LMS}	LMS Cost Function
ξ_{pLMS}	LMS Cost Function
$\xi_{RC-pLMS}$	RC-pLMS Cost Function
ξ_{RLS}	RLS Cost Function
$AF(\theta)$	Array Factor With Respect to Angle of Arrival θ
BW_{3dB}	3 dB Beamwidth
c	Celerity of Electromagnetic Waves
$C(k)$	Chebyshev Polynomial Coefficients

$c_g(k)$	Chebyshev Polynomial General Expanded Form Coefficients
c_m	Erroneous Steering Vector
$d(k)$	Desired Signal Instantaneous Sample
$D(z)$	z Transform Sequence of the Desired Signal $d(k)$
D_1	Delay in the Adaptive Filters Instantaneous Error Signal
D_2	Delay in the Adaptive Filters Weight Update
D_3	Sum Relaxation Delay in the Adaptive Filters Weight Update
D_{ax}	Distance Between Two Consecutive Array Elements in the x Plane
D_{ay}	Distance Between Two Consecutive Array Elements in the y Plane
D_a	Distance Between Two Consecutive Array Elements
$e(k)$	Adaptive Filters Instantaneous Error Signal
$e_1(k)$	First Stage Adaptive Filters Instantaneous Error Signal
$e_2(k)$	Second Stage Adaptive Filters Instantaneous Error Signal
$e_{DLMS}(k)$	Delay LMS Instantaneous Error Signal
$e_i(k)$	i^{th} Stage Adaptive Filters Instantaneous Error Signal
$e_{LLMS}(k)$	LLMS Instantaneous Error Signal
$e_{LMS}(k)$	LMS Instantaneous Error Signal
e_m	Mismatch Error in the Desired Signals Steering Vector
$e_{pLMS}(k)$	pLMS Instantaneous Error Signal
$e_q(k)$	Adaptive Filters Instantaneous Error Signal Subject to QuantizationError
$e_{RC-pLMS}(k)$	RC-pLMS Instantaneous Error Signal
$e_{RLMSp}(k)$	RLMSp Instantaneous Error Signal
$e_{RLMS}(k)$	RLMS Instantaneous Error Signal

$e_{RLS}(k)$	RLS Instantaneous Error Signal
$e_t(k)$	Multi-Stage Adaptive Filters Total Instantaneous Error Signal
f_c	Carriers Signals Frequency
g	Antenna Directivity Gain
$H(z)$	LMS z Transfer Function
$H_{pLMS}(z)$	pLMS z Transfer Function
$H_{RC-pLMS}(z)$	RC-pLMS z Transfer Function
j	Imaginary Complex Number
$J(z)$	z Transform Sequence of the Instantaneous Error Signal $e(k)$
N	Number of Antenna Elements
N_c	FFT Sequence Length
N_x	Number of Antenna Elements in the x Plane
N_y	Number of Antenna Elements in the y Plane
nu	Step Size Exponent
P	Power Radiated by the Antenna
P_c	Projection to the $x-y$ Plane
p_t	FFT Rotation, Twiddle, Factor Exponent
q	Taylor Series Polynomial Expansion Point
r	Base Radix - r Number Representation
$R(z)$	z Transform Sequence of the Input Signals Auto-Correlation Matrix $\mathbf{R}(\tau)$
R_c	Circular Antenna Arrays Radius
r_{k-i}	Fractional Delay Filters Input Signal Auto-Correlation Estimate Time Average

r_{ki}	Fractional Delay Filters Input Signal Auto-Correlation Estimate
RD	Antenna Radiation Directivity
$S(k)$	RC-pLMS Total Desired Signal
$s(k)$	Message Signal
$S_q(k)$	RC-pLMS Total Desired Signal Subject to Quantization Error
$S_{RLMSp}(k)$	RLMSp Total Desired Signal
$S_{RLMS}(k)$	RLMS Total Desired Signal
$T_k(u_c)$	Chebyshev Polynomial Representation With Respect to u_c
U	Antenna Radiation Sensitivity (Practical)
U_0	Antenna Radiation Sensitivity (Theoretical)
u_c	Chebyshev Polynomials Change of Variable
U_{max}	Maximum Antenna Radiation Sensitivity
v_1	Random Gaussian Real Sequence
v_2	Random Gaussian Real Sequence
v_m	Random Complex Gaussian Sequence
$w_{1,m}$	m^{th} Element, First Stage, Instantaneous Weight
$W_{\frac{N_c}{2}}^{kn}$	kn^{th} Rotation, Twiddle, Factor for an FFT Sequence of Length $\frac{N_c}{2}$
$W_{N_c}^{k}$	k^{th} Rotation, Twiddle, Factor for an FFT Sequence of Length N_c
$X[k]$	FFT Sequence
$x_{1,m}$	m^{th} Element, First Stage, Instantaneous Input Signal
$x_{2,m}$	m^{th} Element, Second Stage, Instantaneous Input Signal
$x_c(t)$	Analog Input Signal
$x_d[k]$	Discrete Time Input Signals Sample

$y(k)$	Adaptive Filters Output Signal
$y_1(k)$	First $\frac{N}{2}$ Elements Adaptive Filters Output Signal
$y_2(k)$	Second $\frac{N}{2}$ Elements Adaptive Filters Output Signal
$y_d(k)$	Adaptive Filters Message Signal Subject to Fractional Delay
$y_{i,j}(k)$	Adaptive Filters j^{th} Interference Signal Subject to Fractional Delay
$y_{LMS1}(k)$	First LMS Stage Output Signal
$y_{LMS2}(k)$	Second LMS Stage Output Signal
$y_q(k)$	RC-pLMS Output Signal Subject to Quantization Error
$y_{RLS1}(k)$	First RLS Stage Output Signal

Chapter 1

Adaptive Beamforming: A History

Since the dawn of digital signal processing, researchers have thrived to develop real-time adaptive systems that can independently self-adjust to estimate/filter the desired data from an incoming noisy signal. The ability of an adaptive system/filter to autonomously operate in an unknown noisy environment, makes it an inevitable feature in numerous applications, such as: wireless communication, digital communication, biomedical engineering, control systems, geology and so on. While these adaptive filters still share a common mode of operation, the manner in which they extract the signal of interest changes for each class of applications [1, 2, 3]. Thus, the operation of a linear adaptive filter at a discrete time instance k can be summarized as follows: for an input vector signal $\mathbf{x}(k)$, a desired reference signal $d(k)$ and an output signal $y(k)$, the adaptive algorithm computes the error signal $e(k) = d(k) - y(k)$ to estimate the variable filter weight vector $\mathbf{w}(k+1)$ [2].

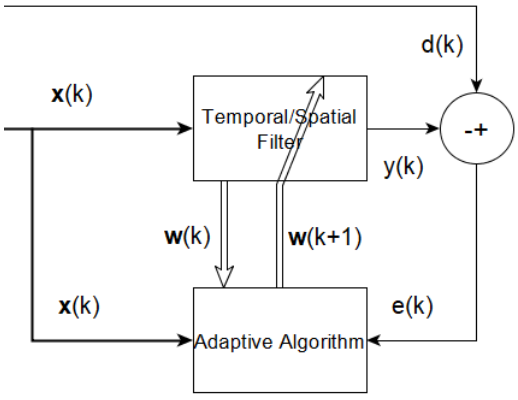

Figure 1.1 – Classical Adaptive Filter

As shown in Figure 1.1, the initial building blocks of an adaptive system are the linear filter and the adaptive algorithm. Moreover, the linear filter block can be modeled with a finite impulse response (FIR) or an infinite impulse response (IIR) for a one dimensional input vector (one sensor) or a spatial filter, i.e. linear combiner (LC) for a N dimensional input (sensor array). For spatial filters, the filtered output signal, $y(k)$, is obtained as a linear combination of the input signal $\mathbf{x}(k) = [x_1(k), x_2(k), ..., x_N(k)]^T$ and the filter weights $\mathbf{w}(k) = [w_1(k), w_2(k), ..., w_N(k)]^T$ and is given as $y(k) = \mathbf{w}^H(k)\mathbf{x}(k)$, where the superscripts ^{T} and $^\mathbf{H}$ denote respectively the vector transpose and the Hermitian operator, i.e. the conjugate transpose. Furthermore, the adaptive algorithm block is used to estimate the variable filter weights based on certain optimization algorithms. Two classical methods exist, in wide sense stationary (WSS) environments, for deriving recursive algorithms [2]:

1. Stochastic gradient descent (SGD): In SGD, the cost function is defined as the mean squared value of the error, and it is represented by a 2^{nd} order function of the filter weights. For recursive applications, the instantaneous squared error is supplied to update the variable weights. As such, the SGD technique refers to the least mean square (LMS) algorithm [2].
2. Least squares (LS): In this method, the cost function is defined as the sum of weighted error squares. In contrast to the SGD technique, LS utilizes matrix operations to compute the gain matrix and updates the variable weights. Such technique is referred to as the recursive least squares (RLS) [2].

With the recent, exponential, spread of wireless connected devices and their requirements in data rate and accuracy, the complexity of array processing algorithms have drastically increased [3, 4]. Such unprecedented growth resulted in a highly congested frequency spectrum [5]. Therefore, many researchers have coupled adaptive signal processing algorithms with antenna arrays to implement adaptive beamforming methods and further increase the spectral efficiency while simultaneously providing a high quality of service (QoS), higher data rates and a wider coverage at a reduced cost. However, the introduction of complex adaptive algorithms, in wireless communication, enforced new limitations on their hardware architecture, such as the need of reduced arithmetic complexity and that of pipeline and parallelism while preserving accelerated convergence and a low residual error in finite precision considerations [4]. Thus, in this chapter, we introduce the recent challenges in

signal processing for wireless applications and one of the most adopted array processing technique, i.e. adaptive beamforming. Additionally, we present our research objectives and motivation towards the imposed challenges.

1.1 Signal Processing in Wireless Communication

In practice, the majority of wireless communication applications, i.e. mobile communication, radar or sonar, involves spatial filtering techniques [1]. Such techniques are achieved by employing a sensor array architecture, generally, with equally spaced homogeneous elements, as shown in Figure 1.2 [4]. With the first antenna element acting as

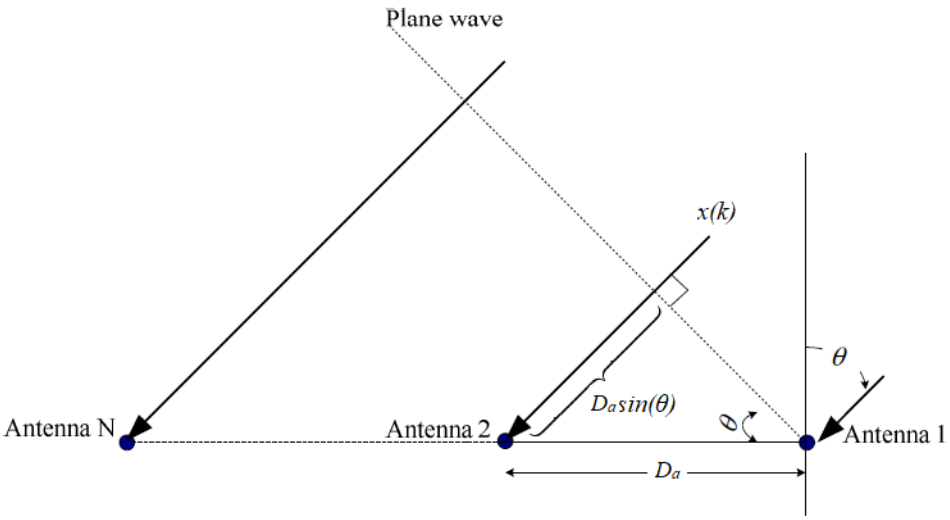

Figure 1.2 – Simple Antenna Array

a reference, θ is the angle of arrival (AOA), D_a is the distance between two consecutive antenna elements and $\mathbf{x}(k)$ is the input signal. In effect, the acquired signal is sampled in space to exploit the spatial properties of signals and noise through array processing (AP) methods, i.e. beamforming [1].

1.1.1 Adaptive Beamforming

Adaptive beamforming is a spatial multiplexing technique, for antenna arrays, used in performing directional signal reception and transmission. Its is achieved by generating a main, pencil, beam in the direction of the desired signal component, while steering nulls in the direction of interferences [6]. As shown in Figure 1.3 [4] a linear antenna with N sensors points its highest gain beam, main beam, towards the desired user while directing nulls to unwanted inputs.

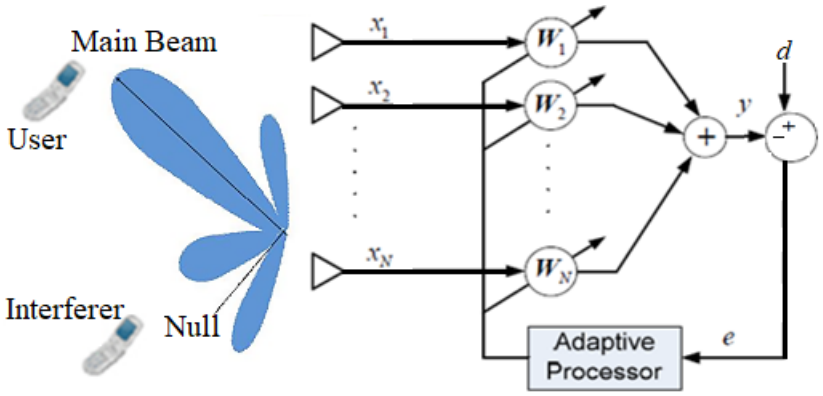

Figure 1.3 – Adaptive Beamforming System

The first beamforming system, i.e. the intermediate frequency (IF) side-lobe canceler was introduced by Howells in 1957 [2, 7]. Howells demonstrated the use of one degree of freedom, two sensor antenna array in amplifying a desired signal while attenuating interference. In his antenna architecture, Howells implemented a high gain antenna element with respect to a low gain, reference, omnidirectional, antenna forming a configurable array with a steerable main beam and null [2, 7]. Subsequently, in 1966, Applebaum derived the control law implementing a control loop for each antenna element [8, 9]. Applebaum's technique resulted in a generalization of Howells side lobe canceler and was based on maximizing the signal to noise ratio (SNR) of the antenna arrays output for random noise environments [2, 8, 9]. Another classical, fully adaptive beamformer, was developed by Widrow et al in 1967 [10]. In [10], the authors demonstrated the effective use of the LMS algorithm in providing a low complexity adaptive beamformer based on the steepest descent optimization technique and the use of the stochastic gradient [2].

The LMS algorithm, presented by Widrow et al, iteratively computes the filter weights by minimizing a pre-defined cost function based on the auto-correlation of the input signal and the cross-correlation of the input and the reference signal [10]. Moreover, a minimum variance distortionless response (MVDR) [11, 12] adaptive beamformer was later introduced by Capon in 1969 [2]. In this algorithm, Capon proposed minimizing the output average power (variance) subject to a pre-set constraint maximizing that of the desired signal [2]. Unlike the LMS, Capcon's MVDR requires previous knowledge on the incoming signals direction, i.e. the AOA [13]. Numerous modifications and algorithms were later on derived based on the classical approaches to accurately achieve optimal behavior, through the use of multi stage beamformers, in recovering the signal of interest and attenuating interference and noise [4, 5, 6, 13, 14, 15]. Some multi stage beamformers are the least mean square - least mean square (LLMS) algorithm [5], the recursive least mean square (RLMS) algorithm [15], the reduced complexity parallel RLMS (RLMSp) [6] and the parallel LMS (pLMS) [14].

1.2 Our Research Project

While the currently employed adaptive algorithms demonstrate satisfactory behavior and convergence in infinite precision mode, in practical applications, i.e within a finite precision mode, their performance is known to degrade. Such degradation is governed by the resource limitations and analog to digital converters of the hosting processor leading to the accumulation of round off and quantization errors [4, 16, 17, 18]. Additionally, the increase in complex computations, i.e. matrix operations, divisions and matrix inversions, due to the nature of the operations, severely degrades the speed of operations which in turn affects the beamformers reliability. Several variants of the classical beamformers have been proposed to present a parallel and easy to pipeline architecture while preserving convergence and acceptable performance. Such techniques are and not limited to: the Gauss-Seidel Fast Affine Projection (GS-FAP) algorithm [19], the a Priori Error-Feedback LSL (EF-LSL) algorithm using a logarithmic arithmetic [20], the relaxed look ahead pipelined LMS [21], the time sharedlook up table (LUT)-less LMS architecture [22] and the division free and variable regularized LMS [23], and the relaxed look ahead par-

allel LMS [24]. The GS-FAP demonstrates better performance over the LMS and some of its variants; However, at the cost of an increase in complexity and processing stages [19]. The EF-LSL structure presents a considerable reduction in the processing time and look up table usage; However, it is based on the least square algorithm, of $O(N^2)$ complexity, and requires the use of a logarithmic number system with a dedicated arithmetic unit [20]. The LLMS [5] and RLMS [15] are multi-stage LMS/LMS and RLS/LMS seperated by an estimate of the steering vector and connected by a delayed error feedback. These techniques show upper level performance over other LMS and RLS variants at the cost of doubling the computational requirements. Nevertheless, presenting a pipeline hardware architecture for the multi stage algorithms is difficult given the design error feedback path and its computational complexity. The previous cascade RLMS algorithm is thus simplified to present a parallel input structure as shown in [6] by eliminating the need for a cascading stage. However, the suggested improvement does not reduce the $O(N^2)$ complexity and does not provide an easy to pipeline architecture. On the other hand, the relaxed look-ahead pipeline LMS, and the time shared LUT-less LMS discussed in [21, 22], present a pipeline architecture for the classical LMS with no noticeable enhancement in the convergence speed nor in the error floor. Furthermore, the pipeline division free variable regularized LMS architecture presented in [23] still presents considerable complexity and requires an on-the-fly computation of its step size compared to the classical LMS. Therefore, it is of utmost importance to achieve a parallel, reduced complexity and easy to pipeline beamformer with improved convergence and low residual error.

1.2.1 Motivation

In this context, the main motivation of our research is to eliminate the trade off between the computational complexity and the performance of the multi-stage algorithms while presenting a suitable hardware architecture for limited resource devices. Through the use of the delay feedback technique [5, 6, 14, 15], we propose a two stages, parallel input LMS algorithm with an accelerated convergence and a minimal residual error for adaptive beamforming (pLMS). pLMS is formed of two LMS stages operating in parallel, where the final error signal is derived as a combination of individual stage errors. The error signal

of the second LMS stage (LMS_2) is multiplied by the imaginary number $j = \sqrt{-1}$ to combine with that of the first LMS stage (LMS_1). Additionally, we further simplify the pLMS to achieve a reduced complexity parallel LMS design (RC-pLMS). The RC-pLMS is obtained by adding a phased sample delayed version of the inputs to the LMS_1 to eliminate the need for a second independent set of weights. In order to present a pipeline, parallel, hardware architecture for the RC-pLMS, we propose the application of the delay and sum relaxed look-ahead technique (DRC-pLMS). Convergence and stability analysis are performed to determine the upper bound of the step size. The quantization effect analysis is conducted to assess the system performance within finite precision arithmetic. Finally, a hardware implementation of the DRC-pLMS design is done in order to study its resource consumption and behavior in finite precision arithmetic. The architecture is implemented using $Q2.15$ [1] format [4, 25].

1.2.2 Outlines

In chapter 1, we first present an overview on the concept of adaptive filtering and the advantages it provides for different digital signal processing (DSP) applications. Second, we discuss the benefits of applying adaptive filtering techniques in wireless communication in order to ease spectral congestion and improve overall performance through beamforming. Finally, we perform a comparative study on different, previous and recent, adaptive beamforming algorithms and hardware implementation while commenting on their advantages and disadvantages.

In chapter 2, we detail the basics of antenna arrays, the popular architectures currently adopted and their essential role in exploiting the spatial domain through beamforming. Concurrently, we explain the basic concepts in spatial filtering and beamforming such as: beam steering, null placement and array ambiguity.

In chapter 3, we highlight the different types and sub-types of adaptive beamforming techniques, i.e. blind, semi-blind, non-blind, temporal based and spatial based. Additionally, we provide an extensive mathematical overview for different, popular, temporal and spatial referenced adaptive beamforming algorithms and their variants. We then present

1. One signed bit, two integer bits and fifteen precision bits

a comparative study on the performance of each, with respect to the convergence profile, through the use of the mean square error (MSE) criteria. Additionally, we list some of the multi stage, cascade, adaptive beamforming algorithms, i.e. LLMS and RLMS, and we propose a reduced complexity parallel input RLMS structure.

Our contributions are detailed in chapters 4, 5 and 6 and are summarized as follows:

— A comparative experimental study is conducted in order to assess the performance of different processors and tools in implementing DSP related routines. The fast Fourier transform (FFT) is the core of numerous DSP applications and is implemented, in parallel and sequential forms, on different field programmable gate array (FPGA) and system on chip (SoC) families. Moreover, the FFT architecture is modeled through the use of various development techniques and tools such as: traditional hardware description language (HDL), high level language (HLL) and high level synthesis (HLS) tools. Synthesis results have shown that an HDL like, high speed design, can be obtained through the effective use of HLL design techniques and proper HLS compiler directives. Moreover, HLL and HLS tools provides a simple and easy method to target multi processor architectures and heterogeneous systems with a shorter design and testing time. However, the concluded disadvantage of using HLS tools is that optimization related tasks is offloaded to the compiler and is dependent on the proper use of compiler directives.

— A high accuracy, low complexity dynamic twiddle function generator using Chebyshev polynomial approximation. While the FFT contributes in accelerating numerous DSP applications, when implemented in finite precision, on limited resource devices, its performance tend to degrade due to the resulting loss of accuracy. As such, we propose a low complexity, high accuracy, dynamic twiddle factor computation method based on Chebyshev polynomial approximation. Moreover, we present its low complexity, low latency, high throughput architecture in finite precision mode. Simulation and synthesis results highlight the superior performance of the adopted approximation method compared to the classical Taylor approximation. In contrast, to the Taylor approximation method, the Chebyshev approximation achieved an accuracy of three decimal digits and smaller resource usage for a fifth order polynomial.

— A two stages parallel LMS (pLMS), its transfer function approximation and pipeline

hardware implementation for adaptive beamforming. In order to eliminate the LMS convergence speed and error floor trade off, while preserving a parallel and low latency architecture, we propose a multi stage parallel LMS architecture connected by error feedback. Where the pLMS overall error is formed as a combination of individual stage errors. In order to numerically compute the maximum step size, the pLMS transfer function approximation derived by modeling the input antenna as a finite input response (FIR) fractional delay filter using Lagrange interpolation. While the proposed pLMS structure is formed of two LMS stages in parallel, presenting a pipeline design is not straight forward given the dependency on the error feedback paths. As such, we propose the application of the delay and sum relaxed look ahead technique, independently, for each of the LMS stages. Thus, the resulting delay pLMS (DpLMS) is obtained and implemented in high throughput, low latency, parallel and pipeline architecture with finite precision. Software simulation results, reflected by the MSE and output beam pattern validated the superior performance of the pLMS with respect to other variants in different signal to noise ratio (SNR) environments. Moreover, with respect to the output beam pattern, the DpLMS implemented in finite precision, showed similar accuracy to that of the infinite precision one. Finally, synthesis results shows that the DpLMS achieved a maximum operating frequency of 208.33 MHz in a low complexity, high throughput architecture.

— A reduced complexity parallel LMS (RC-pLMS), its transfer function approximation and pipeline hardware implementation for adaptive beamforming. While the pLMS eliminated the LMS trade off while maintaining a parallel easy to pipeline design, it is formed of two LMS stages and requires twice the LMS resources. As such, in order to maintain an LMS like complexity while preserving pLMS convergence profile, we propose a single stage RC-pLMS design. RC-pLMS is obtained by modeling the pLMS as a single stage LMS with additional modified inputs, i.e. original input and desired signal subject to a one sample delay. Thus eliminating the need for a second LMS filter and reducing the complexity by half. Similarly, the RC-pLMS transfer function approximation is derived to numerically determine the maximum step size by modeling the input linear combiner as a FIR fractional

delay filter. The RC-pLMS pipeline hardware architecture is obtained through the application of the delay and sum relaxed look ahead techniques (DRC-pLMS) and implemented in finite precision mode. Simulation results, reflected by the MSE and output beam pattern highlight the superior performance of the RC-pLMS, compared to the pLMS and other variants in different SNR environments. Regardless of the adopted approximation, the RC-pLMS presented accelerated convergence, i.e. first 3 iterations, and low steady error while maintaining a low complexity LMS like design. Additionally, hardware simulation show that the finite precision DRC-pLMS achieved similar beam pointing accuracy to the, theoretical, infinite precision. In contrast to the pLMS, synthesis results shows that the DRC-pLMS is obtained at the cost of a marginal, negligible, increase in resource utilization compared to the classical LMS.

Chapter 2

Fundamentals of Antenna Array Beamforming

2.1 Introduction

In order to better illustrate the advantages of antenna arrays, their geometry and radiation pattern, it is important to first present an overview of the basic concepts of a simple antenna element, i.e. an Hertzian Dipole [26]. The Hertzian Dipole or short dipole, first introduced by Heinrich Hertz in 1886, and is formed of two conductor wires of equal length oriented end-to-end with a center-feeding source for transmitting or receiving RF energy [27, 28]. In order to achieve a greater communication range, in 1895 Macroni introduced a special case of the dipole structure, the vertical antenna, by grounding one end of the conductor wires, hence mono-pole, i.e. half dipole [27]. In theory, simple antenna elements are assumed isotropic radiators, i.e. an element which dissipates equal amount of power P in all directions with a radiation intensity $U_0 = \frac{P}{4\pi}$ [26]. As such, for an antenna radiating the same amount of power P, the directive gain can be defined as:

$$g = \frac{U}{U_0} = 4\pi\frac{U}{P} \tag{2.1}$$

where U is the practical antenna radiation intensity. From (2.1) we can define the radiation directivity (RD) as a function of the maximum radiation intensity U_{max} [26], such as:

$$RD = \frac{U_{max}}{U_0} = 4\pi\frac{U_{max}}{P} \tag{2.2}$$

However, for the Hertzian Dipole [26], we have:

$$RD = 1.5 \tag{2.3}$$

It is clear from (2.3) that the radiation directivity for a Hertzian Dipole is a constant and thus uncontrollable. However, to exploit the spatial domain and infer frequency reuse, it is mandatory to have an antenna architecture with a configurable and a controllable beam radiation pattern. Given their proven benefit in providing an electronically steerable beam radiation pattern through the application of adaptive beamforming techniques, antenna arrays can be reliably employed to perform spatial multiplexing and frequency reuse. Thus, in this chapter, we present the popular antenna array geometries as well as the basic concepts of the spatial filtering and beamforming.

2.2 Antenna Array Architecture

Antenna arrays were introduced to perform beamforming techniques for directional signal transmission and reception. Therefore, a beamforming is achieved by forming a main beam towards the direction of a signal of interest and nulls in the direction of interfering signals [5]. Moreover, the desired radiation pattern is achieved automatically by computing the convenient feeding currents phase and amplitude for each antenna element through the use of an adaptive algorithm. From Figure 2.1, the antenna arrays radiation pattern can be defined as follows [26]:

— Main beam: The main lobe which holds the highest power, i.e. the strongest radiation intensity U_{max}.

— Side lobes: The side beams acting as a local maxima with a radiation intensity $U < U_{max}$.

— Nulls: The angle at which no power or radio waves is radiated.

— Half Power Beam Width (HPBW): The HPBW characterizes the ability of an antenna to direct a beam and represented the 3 dB beamwidth, i.e. $BW_{3\text{dB}}$. The HPBW angle occupies the intensity region satisfying the condition $\frac{U_{max}}{2} \leq U \leq U_{max}$.

— First Null Beam width (FNBW): The FNBW describes the ability of the antenna to attenuate and reject interfering signals. The FNBW angle is formed by the main lobe.

— Side Lobe Level (SLL): The radiation intensity of the highest side lobe with respect to the peak of the main beam.

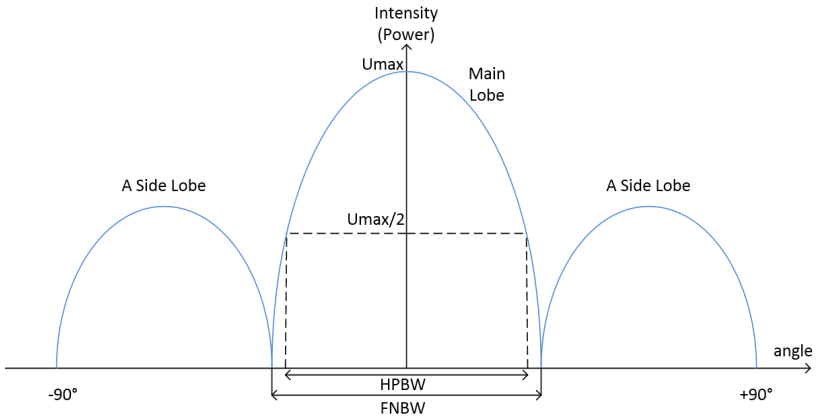

Figure 2.1 – Antenna Array Radiation Pattern (Cartesian Plot)

For focused transmission, with minimal power loss, it is desirable to decrease the HPBW, i.e. 5° for space communication [26]. With dipole antennas, the decrease in HPBW can only be achieved by increasing the length of the antenna; However, that may generate additional multi-lobes [26]. Consequently, the generated multi-lobes severally degrade the performance for long distance transmission by diminishing the power radiating in the direction of interest [26]. In contrast to dipole antennas, antenna arrays can be electronically configured to construct radiation patterns with specific beamwidth and orientation [26]. As such, different antenna array geometries and structures, i.e. linear, planar and circular, can be built in order to achieve the desired radiation properties [26].

2.2.1 Linear Array

The simplest and most popular type of antenna arrays is the linear array. The linear array consists of a number of antenna elements mounted on a straight line and separated

by a spatial distance. For an equally spaced element configuration, the linear array is said to be uniformly spaced and thus the notation uniform linear antenna (ULA). Figure 2.2 presents a ULA of N equally spaced antenna elements [29].

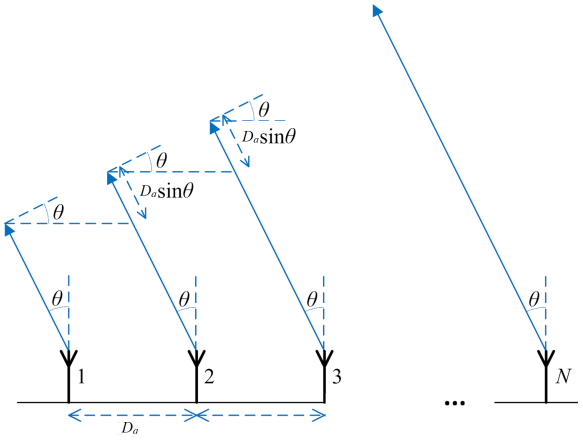

Figure 2.2 – Uniform Linear Antenna Array Structure

For narrow-band complex signals incoming from the far field [30] and with the first antenna element acting as a reference, θ is the angle of arrival, D_a and τ_d are the distance and time delay between consecutive antenna elements, respectively. The time delay τ_d is given by:

$$\tau_d = \frac{D_a \sin(\theta)}{c} \tag{2.4}$$

where c is the celerity of electromagnetic waves.

Let the input vector, $\mathbf{x}(k) = [x_1(k),\ x_2(k),\ .....,\ x_N(k)]^T$, where N is the number of antenna elements, at the discrete time instant, k, to the narrow-band beamformer be defined by:

$$\mathbf{x}(k) = \mathbf{a}_d s_d(k) + \sum_{l=0}^{N-1} \mathbf{a}_{i,l}\mathbf{i}_l(k) + \mathbf{n}(k) \tag{2.5}$$

with $[.]^T$ denotes the matrix transpose, $s_d(k)$ and $\mathbf{i}_l(k)$ are, respectively, the desired and interfering signals with $l < N$, $\mathbf{a}_d$ and $\mathbf{a}_{i,l}$ are the $N \times 1$ complex array steering vector for

the desired signal and for l^{th} interference, respectively, and $\mathbf{n}(k)$ stands for the complex additive white Gaussian noise (CAWGN) vector. A general form of $\mathbf{a}_d$ and $\mathbf{a}_{i,l}$ is given by:

$$\mathbf{a} = [1, e^{-j\psi}, e^{-j2\psi}, ...e^{-j(N-1)\psi}]^T \tag{2.6}$$

where the imaginary number $j = \sqrt{-1}$ and ψ is the phase shift of the received signal corresponding to the time delay τ_d such as:

$$\psi = 2\pi f_c \tau_d = \frac{2\pi c \tau_d}{\lambda} = 2\pi \frac{D_a \sin(\theta)}{\lambda} \tag{2.7}$$

where λ is the carrier signal wavelength, of frequency f_c defined as:

$$\lambda = \frac{c}{f_c} \tag{2.8}$$

Assuming the antenna array is of unity amplitude and zero phase weighting, from (2.6), the array sensitivity response, i.e. array factor (AF), for an angle of arrival (AOA), θ, to the acquired signal and its normalized form at unity, $(AF(\theta))_n$, can be represented as:

$$AF(\theta) = \sum_{i=1}^{N} e^{-j(i-1)\psi} = \frac{e^{-jN\psi} - 1}{e^{-j\psi} - 1} = \frac{\sin(\frac{N\psi}{2})}{\sin(\frac{\psi}{2})} e^{-\frac{j(N-1)\psi}{2}} \tag{2.9}$$

$$(AF(\theta))_n = \frac{\sin(\frac{N\psi}{2})}{N \sin(\frac{\psi}{2})} e^{-\frac{j(N-1)\psi}{2}} \tag{2.10}$$

From (2.10) and [26, 31] the 3 dB beamwidth, can be expressed as:

$$BW_{3\text{dB}} = 0.866 \frac{\lambda}{N D_a} \tag{2.11}$$

it is clear from (2.11) that the main beams 3 dB beamwidth, i.e. HPBW, is inversely proportional to the number of antenna elements [26]. As shown in Figure 2.3 for $N = 8$, $N = 32$ and $N = 64$ antenna elements, a more accurate and precise pointing beam can be achieved by increasing the number of antenna elements [26]. While (2.11) demonstrated the effect of array elements on the HPBW, its denominator exhibits additional variations dependent on the inter element spacing D_a.

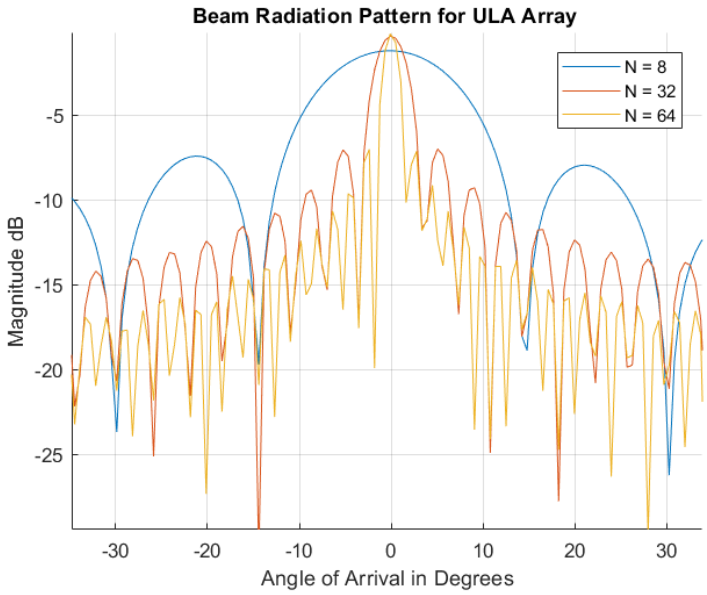

Figure 2.3 – Beam Radiation Pattern Vs Number of Antenna Elements with an Angle of Arrival of 0° and an Inter Element Spacing of $D_a = \frac{\lambda}{2}$

For arrays following custom configuration or design constraints, a maladjustment in inter element spacing can make rise to grating lobes, i.e. array ambiguity and aliasing issues [26]. An example of a grating lobe is shown in Figure 2.4 for a ULA array of $N = 8$ elements with inter element spacing of $D_a = \frac{\lambda}{2}$. From figure 2.4, it is clear that the main beam is directed towards the pre-set AOA of 0°; Moreover, the radiation pattern plot shows additional peaks at ±180°, respectively. The additional peaks presented are referred to grating lobes and results in array ambiguity and uncertainty towards the true direction of interest [26]. To mathematically illustrate the grating lobes effect, the modulus, $|.|$, of the normalized array factor is obtained from (2.10) as:

$$|(AF(\theta))_n| \quad = \frac{1}{N} \quad \frac{|\sin(\frac{\pi D_a N \sin(\theta)}{\lambda})|}{|\sin(\frac{\pi D_a N \sin(\theta)}{\lambda})|} \tag{2.12}$$

thus, for the 90°, Cartesian coordinate, plane, the position of the grating lobes is obtained at $\max(|(AF(\theta))_n|)$, i.e. when $\theta = \arcsin(\frac{n_i \lambda}{D_a})$ where $n_i \in \{1, 2, ..., N\}$ [26].

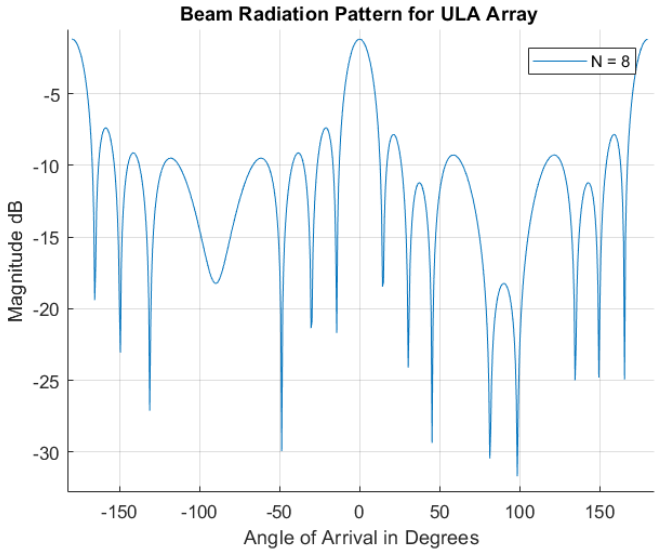

Figure 2.4 – ULA array Beam Radiation Pattern for Angle of Arrival 0° and an Inter Element spacing of $D_a = \frac{\lambda}{2}$

An example of grating lobes for different inter element spacing is shown in Figure 2.5.

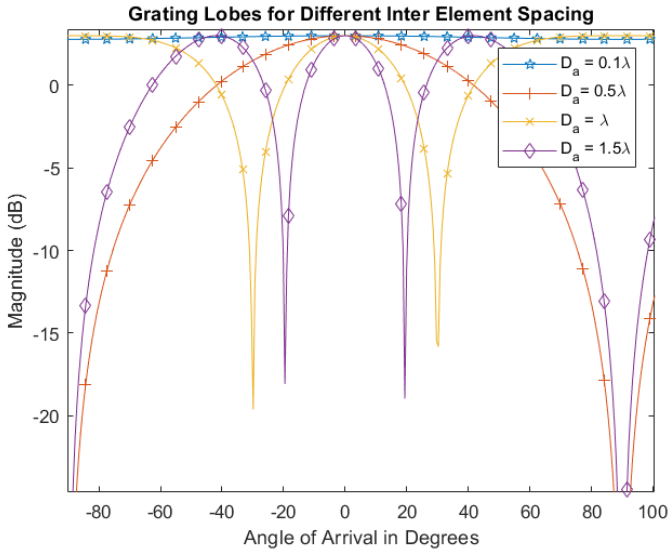

Figure 2.5 – Grating Lobes for Different Inter Element Spacing D_a

From the radiation obtained in Figure 2.5, it is clear that the main beam HPBW becomes narrower as the element spacing becomes wider, however for an element spacing $D_a > 0.5\lambda$ additional lobes appear with and incremental energy as D_a becomes larger [26].

2.2.2 Circular Array

In a circular antenna array, the elements are arranged in a circular shape with a radius R_c. The circular array does not suffer from array ambiguity, produces wider beams and provides full coverage on the azimuth plane [26, 29, 32]. An N elements circular antenna array with radius R_c is shown in Figure 2.6.

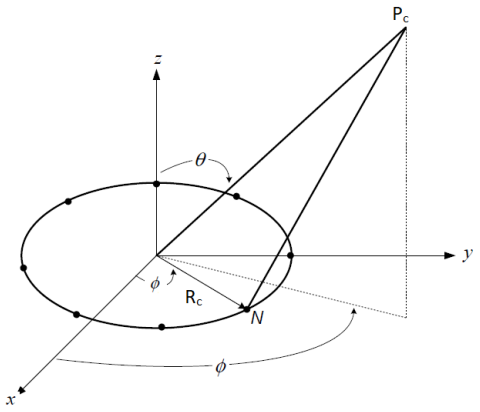

Figure 2.6 – Uniform Circular Array [33]

where ϕ is the azimuth angle and θ is the elevation angle and P_c is a projection point. As such, the new array factor for a circular geometry becomes:

$$AF(\theta, \phi) = \sum_{i=1}^{N_x} e^{-j2\pi \frac{r}{\lambda} sin(\theta) cos(\phi - \frac{2i\pi}{N})} \tag{2.13}$$

Given its previously stated advantages the circular array is a popular antenna geometry employed in application where the signal of interests is known to arrive from an azimuth angle. However, this configuration is at the cost of higher side lobe levels [29, 32].

2.2.3 Planar Array

A planar array is a 2D extension of the linear array, i.e. array elements arranged in the x, y plane. A popular planar array configuration is the rectangular array as shown in Figure 2.7.

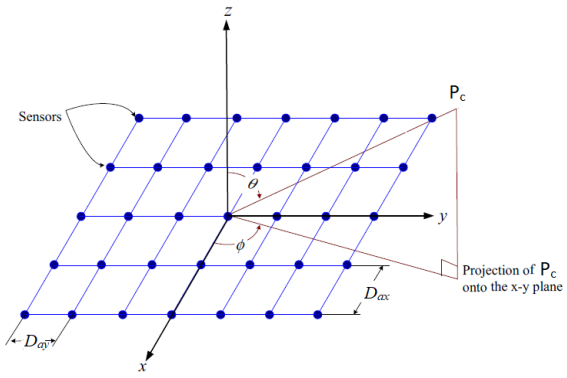

Figure 2.7 – Uniform Rectangular Array [33]

As shown in Figure 2.7, the planar array is a uniform rectangular array with inter elements spacing of D_{ax} and D_{ay}, respectively. The rectangular array is formed of a total of $N_x \times N_y$ elements, where N_x is the number of antennas along the x-plane and N_y is the number of antennas along the y-plane, respectively. In this example, the total number of elements is 24 with $N_x = 4$ and $N_y = 6$. Thus, the overall array factor becomes [26]:

$$AF(\theta, \phi) = \sum_{i=1}^{N_x} \sum_{k=1}^{N_y} e^{-j(i-1)\psi_x} e^{-j(k-1)\psi_y} \tag{2.14}$$

$$\psi_x = 2\pi \frac{D_{ax} \sin(\theta) \cos(\phi)}{\lambda} \tag{2.15}$$

$$\psi_y = 2\pi \frac{D_{ay} \sin(\theta) \cos(\phi)}{\lambda} \tag{2.16}$$

In contrast to the ULA and the circular array, the planar array allows the production of pencil beams by steering the main beam in the elevation plane as well [26].

2.3 Spatial Filtering and Beamforming

Spatial filtering is an inevitable feature of modern wireless communication systems and is used for inferring frequency reuse, limiting interference and providing higher data rates and signal to noise ratio (SNR). An antenna array geometry allows the creation of electronically steerable beams and nulls by specific current feeding and element configuration, i.e. beamforming [26]. The use of a beamforming technique steers the antennas main beam, with constructive amplitude, towards a signal coming from a desired location while directing nulls towards interference, i.e. spatial filtering [2, 26, 30].

2.3.1 Beam Steering

An efficient form of adaptive beamforming is achieved by connecting the antenna array to a beamformer processor through the use of a signal conditioning circuit and high speed analog to digital (AD) converters [2, 26, 34, 35]. The received signals are spatially sampled and collected by each antenna element, converted to their digital form and fed to the beamformer. Through the use of an adaptive algorithm, with respect to a desired reference signal, the beamformer appropriately weights the input samples to automatically steer the main beam towards the direction of the desired AOA while placing nulls in the direction of interference [2, 4, 26]. The adaptive weighting correctly filters the interfering signals irrespective of their characteristics and AOA [26].

Adaptive arrays controls the main beams and null steering towards specific direction with respect to a reference signal. Beam steering is achieved through appropriate complex weighting, i.e. appropriate setting of the complex amplitude and phase of the feeding currents [26]. The complex weights, for steering control, are of the form $w_n = \rho_n e^{j\omega_n}$ where w_n is the complex weight of the n^{th} antenna element, ρ_n and ω_n are the corresponding amplitude and phase, controlling the main beam and nulls angles, their beamwidth and their side lobe levels, respectively [13, 26].

The main beam steering equation, assuming no noise nor interference, can be written as [26]:

$$\mathbf{w} = \frac{1}{N}\mathbf{a}_d(\theta) \tag{2.17}$$

where $\mathbf{w}$ is the linear combiner filter weights, the added factor $\frac{1}{N}$ is a normalization factor to obtain a unity response in the direction of interest. Thus, for a 4 element ULA array, i.e. $N = 4$ with inter element spacing of $D_a = \frac{\lambda}{2}$ the required weights to steer the main beam towards a desired AOA of $\theta = 30°$ can be computed from (2.9) and (2.17), as follows:

$$\mathbf{w} = \frac{1}{4}[1,\ e^{-j1.5708},\ e^{-j3.1416},\ e^{-j4.9348}] \tag{2.18}$$

While the application of (2.17) successfully steers the main beam towards the desired AOA, it does not null any interfering signals.

In order to allow automatic null steering and attenuate unwanted signals additional constraints are required on the weight steering equation. In case of interference, directing a main towards the desired direction only results in a 3 dB rejection [26], which is a relatively small improvement compared to the cost and complexity of the beamformer. However, given the nature of the array configuration, it is possible to steer nulls in the direction of interference by computing the complex weights following a set constraint [26].

Let $\mathbf{a}_d(\theta_0)$ be the array steering vector for the desired signal and $\mathbf{a}_{i,1}(\theta_1), ..., \mathbf{a}_{i,k}(\theta_k)$ are the k, $N \times 1$ nulls steering vectors. The overall weight can be obtained as a solution to the following equations:

$$\mathbf{w^H a}_d(\theta_0) = 1 \tag{2.19}$$

$$\mathbf{w^H a}_d(\theta_k) = 0 \qquad \forall k \in [1, ..., L] \tag{2.20}$$

where the superscript $^{\mathbf{H}}$ is the Hermitian transpose, $L = N - 1$. Let $\mathbf{A}_i$ be a $N \times N$ square matrix whose columns are formed by the $k + 1$ steering vectors and $\mathbf{c}$ is a $N \times 1$ constraint vector of the form:

$$\mathbf{c}_i = [1, 0, 0, ..., 0]^T \tag{2.21}$$

assuming all steering vectors are linearly independent, i.e. $\mathbf{A}_i$ is non singular, the weight vector can be computed as [26]:

$$\mathbf{w^H} = \mathbf{c}_i^T \mathbf{A}_i^{-1} \tag{2.22}$$

where $\mathbf{A}_i^{-1}$ represents the inverse matrix of $\mathbf{A}_i$. Thus, for a 2 element ULA array, i.e. $N = 2$ with inter element spacing of $D_a = \frac{\lambda}{2}$, the required weights to steer the main beam towards a desired AOA of $\theta_0 = 50°$ while steering a null toward an interference signal at $\theta_1 = -15°$ can be obtained as follows:

$$\mathbf{a}_d(\theta_0) = [1, e^{-j2.4066}] \tag{2.23}$$

$$\mathbf{a}_{i,1}(\theta_1) = [1, e^{j0.8131}] \tag{2.24}$$

using (2.22) with:

$$\mathbf{A}_i = [\mathbf{a}_d(\theta_0), \ \mathbf{a}_{i,1}(\theta_1)] \tag{2.25}$$

$$\mathbf{c}_i = [1, 0]^T \tag{2.26}$$

we can compute $\mathbf{w}$ to obtain:

$$\mathbf{w} = [0.5 + j0.0195, -0.35708 + j0.3498]^T \tag{2.27}$$

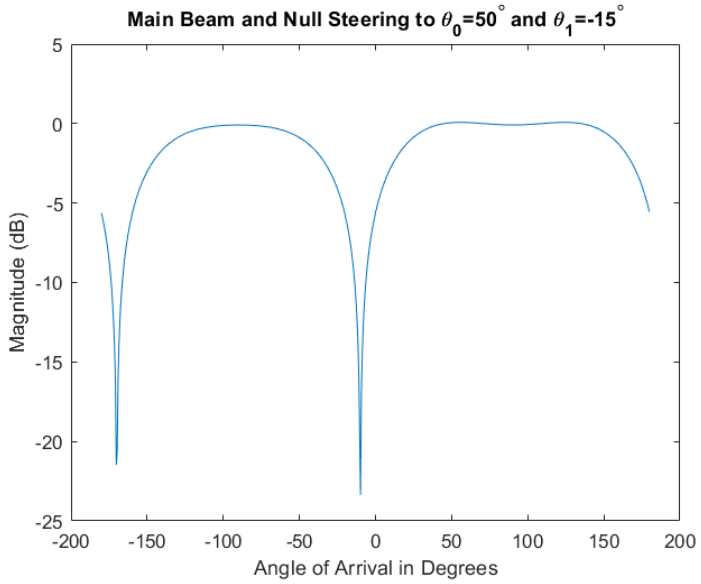

Figure 2.8 – Main Beam and Null Steering to $\theta_0 = 50°$ and $\theta_1 = -15°$ Respectively

From Figure 2.8, it is clear that the beamformer was able to steer a null toward the direction of the interfering signal $\theta_1 = -15°$, while keeping its main beam directed at the desired AOA $\theta_0 = 50°$.

2.4 Conclusion

In this chapter, we presented the benefits of using antenna arrays with a summary of the popular array geometries employed in communication systems, i.e. Linear, planar and circular arrays. Moreover, through the use of antenna arrays, we presented the concept of spatial filtering through the application of beamforming. While the provided beam and steering technique is effective in attenuating interfering signals, however it does not filter uncorrelated signal noise. Additionally, due to their narrow width, it is difficult to accurately place nulls in the desired direction. Thus, it is of a main interest to design a high performance beamformer with accurate beam and null pointing while maintaining a low complexity architecture.

Chapter 3

Overview of Adaptive Beamforming Algorithms

3.1 Introduction

While an antenna array configuration is capable of performing spatial filtering through beamforming, it is only a physical arrangement of independent antenna elements and cannot perform any processing task. Thus, a dedicated beamforming processor, i.e. beamformer, is required to collect incoming signal samples and electronically steer the main beam and nulls towards the desired directions [4, 5, 13, 26]. To obtain an automatically controllable beam, the beamformer implements an adaptive system that continuously adjusts the array parameters with respect to an adaptive algorithm. As such, the choice of the beamforming algorithm is generally governed by the imposed application constraints, i.e. robustness, accuracy and accelerated convergence, and the limitations of the hosting processor, i.e. limited resources and finite precision [4, 25, 36].

3.2 Adaptive Beamforming

Most adaptive algorithms are classified into three main categories: non-blind, semi-blind or blind [26], as shown in Figure 3.1. In non-blind beamforming the system relies on the temporal or spatial characteristics, i.e. the direction of arrival (DOA), of a reference signal to iteratively compute the array weights with respect to a set cost function or constraints [3, 26]. Thus, in non-blind beamforming, a clean copy of the reference signal is needed to achieve satisfactory behavior [2]. In contrast, the blind beamforming system

extracts some unknown characteristics of the incoming signal, related to the channel impulse response, the spectrum, the modulus or the envelop deviation, without any prior knowledge of the array geometry in order to correctly steer the main beam and nulls [2, 26].

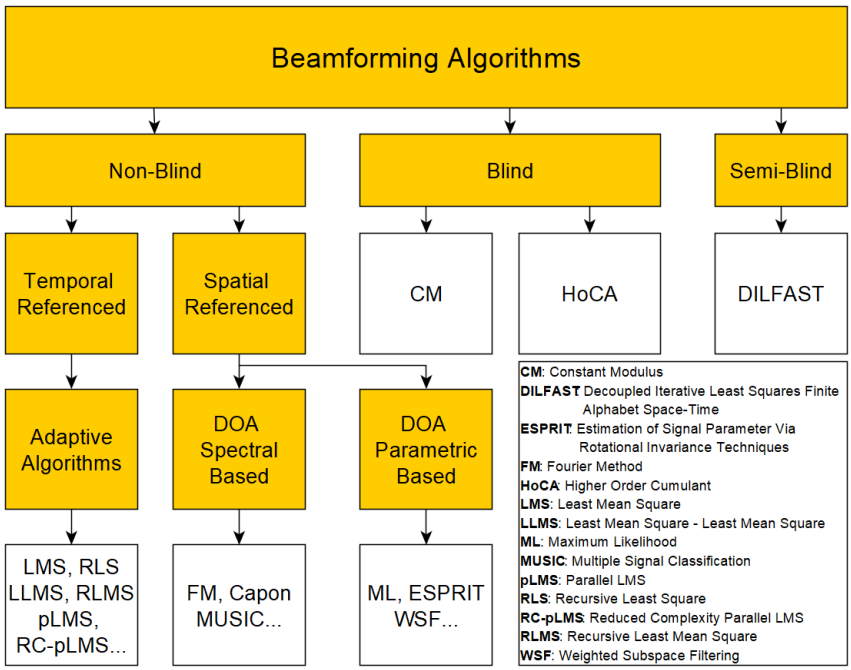

Figure 3.1 – Beamforming Algorithms

While most of the blind algorithms requires complex mathematical routines to achieve their goals, the inclusion of a training sequence is shown greatly accelerate the computation, hence the name semi-blind or hybrid [26].

3.2.1 Non-Blind based Beamforming

In non-blind beamforming, algorithms rely on a pre-set reference signal or parameters in order to effectively steer the main beam and nulls towards a desired look direction [2, 3]. The adaptive algorithm is said to be temporal referenced if it relies on a training sequence to form its reference signal and compute the array weights, or spatial referenced if it relies on the array characteristics and DOA [26].

For temporal referenced beamforming, all adaptive algorithms continuously update the filter weights with respect to a pre-set cost function, such as:

— Minimum mean square error (MMSE) [26]

— Maximum signal to interference plus noise ratio (MSINR) [26]

— Minimum variance distortionless response (MVDR) [26]

Identically, at convergence, all the listed algorithms compute the optimal weight given by the Weiner-Hopf equation [2, 10, 26] and defined as [4]:

$$\mathbf{w}_o \;=\; \mathbf{R}^{-1}\mathbf{p} \tag{3.1}$$

where $\mathbf{R} = \mathbf{R}(0)$ and $\mathbf{R}^{-1}$ are the input signal auto-correlation matrix and its inverse, respectively, and $\mathbf{p} = \mathbf{p}(0)$ is the cross correlation vector of the input $\mathbf{x}(k)$ and desired signal $d(k)$. $\mathbf{R}(0)$ and $\mathbf{p}(0)$ are defined at lag $\tau = 0$ as:

$$\mathbf{R}(\tau) \;=\; \mathrm{E}[\mathbf{x}(k-\tau)\mathbf{x}^{\mathbf{H}}(k)] \tag{3.2}$$

$$\mathbf{p}(\tau) \;=\; \mathrm{E}[d^{*}(k-\tau)\mathbf{x}(k)] \tag{3.3}$$

with E[.] being the expectation operator, the superscripts * and $^{\mathbf{H}}$ denotes the complex conjugation and the Hermitian transpose, and the lag $\tau = k_1 - k_2$. Where, k_1 and k_2 are different time instances from which an observation of the random process is taken. The performance of temporal adaptive algorithms depends on the selected algorithm and the availability of a noise free copy of the reference signal [26]. Some of the popular non-blind temporal referenced adaptive beamforming algorithms are the recursive least square (RLS), a simpler form of the Kalman filter, and the least mean square (LMS) [13].

Spatial referenced beamformers make use of the DOA of the incoming signal in order to correctly direct its main beam and nulls, as to amplify the desired signal while suppressing co-channel interferences [2, 26]. The performance of the spatial reference DOA based methods is mainly correlated with the performance of the DOA algorithm itself and its robustness against array calibration and inter element spacing errors [26]. Some of the popular direction finding techniques are spectral based, i.e. the Fourier method (FM) and the multiple signal classification (MUSIC) algorithm, and parametric based such as: the

maximum likelihood (ML) and the estimation of signal parameter via rotational invariance techniques (ESPRIT).

Althought fast adaptive algorithms exists, such as: the Fast adaptive ESPRIT algorithm [37, 38], the major disadvantage experienced is their high computational complexity [26]. The increase in complexity results in an intolerable delay, as the system scales in size, making these algorithms undesirable when targeting a hardware implementation on limited resource processors. In contrast, temporal referenced beamformers, benefits from self-correction against array calibration and pointing errors giving their dependency on a per-determined training sequence. For an array of N elements the beamformer makes use of N degrees of freedom (DoF), as such with one element directed to the desired source $N-1$ remains for nulling up to $N-1$ interferences. However, an attractive feature of temporal referenced algorithms is their ability to maximize the systems output signal to interference plus noise ratio (SINR) for a number of sources exceeding the number of antenna elements, i.e greater than N [2, 26]. Additionally, some of the temporal referenced beamformers, i.e. LMS, implements a low complexity architecture making it the most desirable for hardware implementations [4, 21, 22, 23, 24, 34].

3.2.2 Blind Based Beamforming

Since non-blind algorithms rely on a training sequence, during the initialization phase no data can be transmitted over the communication channel thus reducing link efficiency [26]. However, blind adaptive algorithms exploit the statistical and structural properties of the incoming signal without any knowledge on the arrays geometry or signal characteristics [26]. As such, blind adaptive algorithms do not need a training sequence, thus, implicitly upgrading the performance of the system and its spectral efficiency [2, 26].

In the case of blind beamforming, the algorithm samples the input signals in time domain forming the input signals matrix $\mathbf{X}_b$ to extract the data signal $\mathbf{S}_b$ and the systems transfer function $\mathbf{H}_b$ satisfying $\mathbf{X}_b = \mathbf{H}_b\mathbf{S}_b$ [26]. Most blind algorithms rely on the statistical knowledge of some parameters, i.e. envelop deviation, spectral self-coherence, cyclo-stationary and finite alphabet of symbols, in order to estimate of $\mathbf{S}_b$ or $\mathbf{H}_b$ through matrix decomposition. Some of the popular blind algorithms are: the constant modulus

(CM) algorithm and the higher order cumulant algorithm (HoCA) [2, 3, 5, 26]. However, while fast adaptive algorithms exists [39, 40, 41], they are of high complexity, such issue pressures blind algorithms and limits their performance when implemented on dedicated, limited resources, hardware, such as field programmable gate array (FPGA) [25].

3.2.3 Semi-Blind based Beamforming

The inclusion of a prior knowledge, i.e. a training sequence, in blind algorithms significantly improves their computational performance and real-time adaptation profile. The use of a training sequence introduces a new class of algorithms identified as hybrid or semi-blind. An example of a semi-blind algorithm with real time beamforming capabilities is the decoupled iterative least squares finite alphabet space-time (DILFAST) [26, 42]. The DILFAST algorithm uses a 'bit field' training sequence [26, 42] at its initialization phase and performs detection and estimation techniques based on the structural properties of the input signal. The algorithm makes use of the finite alphabet (FA) constellation to map the input samples without any need of subspace estimation [42]. While the DILFAST algorithm can be implemented in real-time it relies on the use of FA and is not robust against any change of the FA as a consequence of some operating conditions [26].

3.3 Adaptive Algorithms

Non-blind adaptive algorithms provide an attractive set of features making them suitable candidates for hardware implementation [4]. Moreover, an overview on different multi stage adaptive beamformer is presented. The multi stage beamformer is formed by two adaptive algorithms, either in cascade or in parallel, sharing a similar training sequence, in which the error signal, $e_2(k)$, of the second stage is delayed and fed back to combine with that of the first stage, $e_1(k)$, to form the total system error $e_t(k)$ [4, 13] as follows:

$$\begin{aligned} e_1(k) &= d(k) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}_1(k) \\ e_2(k) &= d(k) - \mathbf{w}_2^{\mathbf{H}}(k)\mathbf{x}_2(k) \\ e_t(k) &= e_1(k) - e_2(k-1) \end{aligned} \tag{3.4}$$

where $\mathbf{w}_1(k)$, $\mathbf{x}_1(k)$ and $\mathbf{w}_2(k)$, $\mathbf{x}_2(k)$ are the first and second stage filter weight and input vector, respectively. As such, in this section, we select some of the popular algorithms and provide an overview of their mathematical structure, their performance and their computational complexity while listing their advantages and drawbacks.

3.3.1 MVDR Algorithm for Beamforming

The minimum variance distortionless response (MVDR) beamformer maximizes the output SINR by minimizing the interference and noise power (variance), while preserving a distortionless response in the direction of the signal of interest [11, 13]. The output SINR is given by:

$$SINR \;=\; \frac{\mathrm{E}[|\mathbf{w^H s}(k)|^2]}{\mathrm{E}[|\mathbf{w^H}(\mathbf{i}(k)+\mathbf{n}(k))|^2]} = \frac{\sigma_d^2|\mathbf{w^H a}(\theta)|^2}{\mathbf{w^H R}_{i+n}\mathbf{w}} \tag{3.5}$$

where $|.|$ is the complex modulus, $\mathbf{a}(\theta)$ is the array steering vector in function a known reference angle of arrival (AOA), σ_d^2 is the desired signal variance and $\mathbf{R}_{i+n}$ is the $N \times N$ interference plus noise co-variance matrix defined as $\mathbf{R}_{i+n} = \mathrm{E}[(\mathbf{i}(k)+\mathbf{n}(k))(\mathbf{i}(k)+\mathbf{n}(k))^{\mathbf{H}}]$ [11, 13, 43]. Thus, the MVDR problem can be formulated, with respect to (3.5), as:

$$\min_{\mathbf{w}} \mathbf{w^H R}_{i+n}\mathbf{w} \quad st \quad \mathbf{w^H a}(\theta) = 1 \tag{3.6}$$

From (3.6), the optimal weight vector $\mathbf{w}_{MVDR}$ can be computed as:

$$\mathbf{w}_{MVDR} \;=\; \frac{\mathbf{R}_{i+n}^{-1}\mathbf{a}(\theta)}{\mathbf{a^H}(\theta)\mathbf{R}_{i+n}^{-1}\mathbf{a}(\theta)} \tag{3.7}$$

From (3.7), it is clear that the MVDR beamformer steers its main beam and nulls with respect to a known reference AOA, i.e. spatial referencing. Additionally, the unknown interference plus noise co-variance matrix $\mathbf{R}_{i+n}$ can be replaced by its estimate $\widehat{\mathbf{R}}_{i+n}$. However, the accuracy of the estimation depends on the length of the training sequence available [11, 12, 43]; Therefore, the data sample co-variance matrix estimate is computed

using the following equation:

$$\widehat{\mathbf{R}}_{i+n} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k) \tag{3.8}$$

where K is the number of training data samples. However, the MVDR does not provide any robustness against array calibration errors or look direction mismatch [2, 11, 13, 26]. Moreover, the resulting estimate, $\widehat{\mathbf{R}}_{i+n}$, also includes the desired signal component and presents additional ambiguity [2, 4, 13]. In order to provide additional robustness against random mismatch errors, a robust MVDR beamformer is proposed in [11, 43]. The robust MVDR algorithm update the minimization problem described by (3.6) in order to include additional constraints not only for the preset steering vector and look direction but for all other vectors belonging to the set of $\Psi(\varepsilon) = \{\mathbf{c}_m|\mathbf{c}_m = \mathbf{a} + \mathbf{e}_m, ||\mathbf{e}_m|| \leq \varepsilon\}$ to become [43]:

$$\min_{\mathbf{w}} \mathbf{w}^{\mathbf{H}}\widehat{\mathbf{R}}_{i+n}\mathbf{w} \quad s.t. \quad |\mathbf{w}^{\mathbf{H}}\mathbf{c}_{\mathrm{m}}| \geq 1 \qquad \forall \mathbf{c}_m \in \Psi(\varepsilon) \tag{3.9}$$

where $||.||$ denotes the vector Frobenius norm, $\mathbf{e}_m$ is the mismatch error in the steering vector and $\mathbf{c}_m$ is the erroneous steering vector. However, the optimization problem in (3.9) describes an infinite number of non convex constraints [43] and is reformulated to a single constraint corresponding to the worst case mismatch as a second order cone programming (SOCP) problem [43], such as:

$$\min_{\mathbf{w}} \mathbf{w}^{\mathbf{H}}\widehat{\mathbf{R}}_{i+n}\mathbf{w} \quad st \quad \mathbf{w}^{\mathbf{H}}\mathbf{a} \geq \varepsilon||\mathbf{w}|| + 1 \tag{3.10}$$

The SOCP based robust MVDR described in (3.10) is computationally intensive and of order $O(N^3)$ [43].

3.3.2 Least-Mean Square (LMS) Algorithm

The least mean square (LMS) algorithm, was first introduced by Widrow and Hoff in [10] and is the realization of the steepest descent optimization method by means of a stochastic gradient [2, 3, 4]. the LMS minimizes the mean square error (MSE) as a cost

function [10, 44, 45], to estimate the optimal filter weight, i.e. the Wiener solution. The MSE cost function, ξ_{LMS}, is defined by:

$$\xi_{LMS}(k) = \mathrm{E}[|e_{LMS}(k)|^2] = \mathrm{E}[e_{LMS}(k)e^*_{LMS}(k)] \tag{3.11}$$

where $e_{LMS}(k)$ is the error signal and $d(k)$ is the reference signal. Moreover, equation (3.11) can be expanded as:

$$\mathrm{E}[|e_{LMS}(k)|^2] = \mathrm{E}[|d(k)|^2] - \mathbf{p^H w}(k) - \mathbf{w^H}(k)\mathbf{p} + \mathbf{w^H}(k)\mathbf{R}\mathbf{w}(k) \tag{3.12}$$

Equation (3.12) is a quadratic equation in function of the array weight vector $\mathbf{w}(k)$, the input signal auto correlation matrix $\mathbf{R}$ and the input signal and desired signal cross correlation vector $\mathbf{p}$. Therefore, the optimal weight vector, $\mathbf{w}_{oplms}$, of $\mathbf{w}(k)$, assuming a wide sense stationary (WSS) process, can be obtained by differentiating (3.12) with respect to $\mathbf{w^H}(k)$ [5, 15, 46], and setting the resulting LMS gradient, ∇_{LMS}, to zero, such as:

$$\nabla_{LMS} = \frac{\partial \xi_{LMS}(k)}{\partial \mathbf{w^H}(k)} = -\mathbf{p} + \mathbf{R}\mathbf{w}(k) \tag{3.13}$$

Hence $\mathbf{w}_{oplms}$ becomes:

$$\mathbf{w}_{op} = \mathbf{R}^{-1}\mathbf{p} \tag{3.14}$$

The LMS error and weight update equations are obtained as follows:

$$e_{LMS}(k) = d(k) - y(k) \tag{3.15}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu_{LMS} e^*_{LMS}(k)\mathbf{x}(k) \tag{3.16}$$

where μ_{LMS} stands for the gradient descent step or the step size [4, 14, 44, 45]. To ensure stable performance and convergence, the upper bound of the step size, μ_{LMS}, is given

with respect to the stability analysis conducted in [2] as:

$$\mu_{LMS} \quad < \quad \frac{1}{\lambda_{R,max}} \tag{3.17}$$

where $\lambda_{R,max}$ is the maximum eigenvalue of $\mathbf{R}$. Thus, to ensure the convergence and the stability of LMS the step size μ_{LMS} must satisfy (3.17). While LMS offers minimal computational complexity, of order $O(N)$, it suffers from a trade off between its convergence speed and its error floor [4].

Several variants of the classical LMS have been proposed to improve its performance by eliminating the trade off between its convergence speed and steady state error [4]. This includes, and is not limited to, the normalized LMS (NLMS) [47, 48], variable step size LMS (VSSLMS) [49, 50, 51, 52, 53], the modified robust variable step size LMS (MRVSS) algorithm [54] and the least mean square - least mean square algorithm (LLMS) [5]. In NLMS, the authors in [47, 48] proposed adjusting the step size, μ, in accordance with the input signal power through auto-correlation, thus allowing faster convergence. However, NLMS, suffers degraded performance in low SINR environments due to the reduction in the step size [5], additionally, the systems performance is highly dependent on the choice of its initial parameters. In contrast, the VSSLMS variant described in [50] begins the adaptation process with respect to a large step size decremented as the algorithm approaches its steady state. Such technique allows accelerated convergence and a lower steady state error, however at the cost of a high increase in computational complexity [55]. Despite the suggested modifications, the NLMS and VSSLMS algorithms still suffer from degraded performance in low SINR conditions [55]. Thus, to further increase their robustness, the authors in [54] proposed the MRVSS LMS algorithm. In comparison, the MRVSS has shown satisfactory robustness against added noisy and when operated in non stationary environments. However, the MVRSS technique results in a considerable increase in the computational complexity and its performance also depends on particular system environments [5].

In LLMS a multi stage structure is proposed for improving the convergence rate while simultaneously reducing the residual error of a classical LMS [5]. LLMS is achieved by cascading two LMS stages by the use of an estimate of the adaptive array vector and error

feedback. This technique shows superior performance over previously discussed LMS variants, however at the cost of doubling the computational complexity and the introduction of a division operation with respect to the classical LMS [4, 5].

3.3.3 Recursive Least-Square (RLS) Algorithm

The RLS algorithm updates the arrays weight vector based on the minimization of a cost function, i.e. the sum of squared errors, ξ_{RLS}, for a known sampling window [6, 13, 15] and is given by:

$$\xi_{RLS} = \sum_{i=1}^{k} \alpha^{k-1} |e_{RLS}(k-1)|^2 \tag{3.18}$$

where the error signal, $e_{RLS}(k)$, is defined as $e_{RLS}(k) = d(k) - y(k)$ and $\alpha \in [0, 1]$ is the exponentially weighted forgetting factor [6]. Hence, the weight vector update formula becomes:

$$e_{RLS}(k) = d(k) - \mathbf{w}^{\mathbf{H}}(k)\mathbf{x}(k) \tag{3.19}$$

$$\mathbf{L}(k) = \frac{\alpha^{-1}\mathbf{Q}^{-1}(k-1)}{1 + \alpha^{-1}\mathbf{x}^{\mathbf{H}}(k)\mathbf{Q}^{-1}(k-1)\mathbf{x}(k)} \tag{3.20}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{L}(\mathrm{k})\mathbf{x}(k)e^{*}_{RLS}(k) \tag{3.21}$$

where $\mathbf{L}(k)$ is the gain matrix. The inverse signal auto-correlation matrix $\mathbf{Q}^{-1}(k)$ and the weight vector $\mathbf{w}(k-1)$ are defined as follows [15]:

$$\mathbf{Q}^{-1}(k) = \alpha^{-1}\mathbf{Q}^{-1}(k-1) - \frac{\alpha^{-2}\mathbf{Q}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k)\mathbf{Q}^{-1}(k-1)}{1 + \alpha^{-1}\mathbf{x}^{\mathbf{H}}(k)\mathbf{Q}^{-1}(k-1)\mathbf{x}(k)} \tag{3.22}$$

$$\mathbf{z}(k) = \sum_{i=1}^{k} \alpha^{k-i}\mathbf{p}(k) \tag{3.23}$$

$$\mathbf{Q}(k) = \sum_{i=1}^{k} \alpha^{k-i}\mathbf{R}(k) \tag{3.24}$$

While the RLS provides greater convergence, compared to the LMS it suffers from an increased computational complexity, of $O(N^2)$, lacks robustness against fixed point arith-

metic, i.e. underflow and divide by zero and does not provide user tracking capabilities [2, 6, 13, 15, 25]. Several variants have been proposed for maintaining an accelerated convergence profile and providing a tracking ability in time varying environments for the classical RLS [15]. These techniques include, and are not limited to, the adaptive forgetting factor RLS algorithm (AFF-RLS) [56], the variable forgetting factor RLS (VFFRLS), the extended recursive least square (EX-KRLS) algorithm [5], the recursive least mean square (RLMS) [15] and the parallel RLMS [6]. However, the improvement in the tracking ability of the RLS when implementing the AFF-RLS, the VFFRLS and the EX-KRLS algorithms is achieved at the cost of a considerably large increase in computational complexity [6, 15]. In contrast, the RLMS structure achieves an accelerated convergence with superior tracking capabilities compared to other modifications, by means of a multi-stage algorithm [13]. RLMS employs a RLS stage followed by a LMS stage separated by an estimate of the array image vector and connected by an error feedback. This technique shows superior performance over previously discussed LMS and RLS variants however at the cost of an increase in the computational complexity.

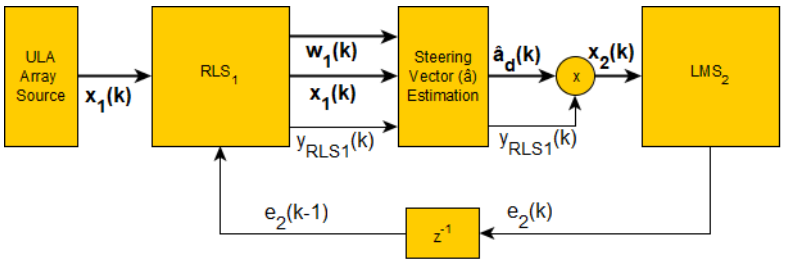

Figure 3.2 – Cascaded RLMS

3.3.4 RLMS Adaptive Beamformer

The Recursive-Least mean square (RLMS) algorithm is formed of a RLS stage followed by a LMS stage separated by an estimate of the array image vector. The two stages algorithms are connected by a delayed error feedback [15] as shown in Figure 3.2 where the block z^{-1} represents a one sample delay. In RLMS, the output of the first stage RLS_1, $y_{RLS1}(k)$, is multiplied by the estimate of the desired signal steering vector $\hat{\mathbf{a}}_d(k)$, forming the input, $\mathbf{x}_2(k)$, to the second LMS_2 stage. Moreover, $\hat{\mathbf{a}}_d(k)$, is given by its stochastic

approximation near convergence in its instantaneous form, detailed in Appendix A, [13, 15] as:

$$\hat{a}_{d,m} \simeq \frac{w_{1,m}(k)x_{1,m}(k)}{w_{1,m}(k)y_{RLS1} + \vartheta} \tag{3.25}$$

where $\hat{a}_{d,m}$ is the m^{th} elements of the complex steering vector approximate $\hat{\mathbf{a}}_d(k)$ and ϑ is a small constant introduced to mitigate a division by zero [5, 13] such that:

$$0 < \vartheta \ll \frac{|y_{RLS1}|}{N} \sum_{m=1}^{N} |w_{1,m}| \tag{3.26}$$

where $w_{1,m}(k)$, $x_{1,m}(k)$ and $\hat{a}_{d,m}$ are the RLS_1 tap weights, input signal and the estimate of the m^{th} antenna element of the complex steering vector $\tilde{\mathbf{a}}_d$ at the time instant k with $m \in \{1, 2, 3.....N\}$. Hence, the input to the m^{th} antenna element for the LMS_2 stage, $x_{2,m}(k)$, is derived in [5, 15] as:

$$x_{2,m}(k) = \frac{w_{1,m}(k)x_{1,m}(k)}{w_{1,m}(k)y_{RLS1}(k) + \vartheta} y_{RLS1}(k) \tag{3.27}$$

Moreover, a delayed version of the error signal $e_2(k)$ of the LMS_2 stage is fed-back to combine with that of the RLS_1 to form the overall error signal $e_{RLMS}(k)$ used to update the main tap weights of the RLS_1 stage. The overall error signal and the RLMS weight update equation, with respect to (3.21), becomes [15]:

$$e_{RLMS}(k) = e_1(k) - e_2(k-1) \tag{3.28}$$

$$\mathbf{w}_1(k) = \mathbf{w}_1(k-1) + \mathbf{L}(\mathrm{k})\mathbf{x}_1(k)e^*_{RLMS}(k) \tag{3.29}$$

While the RLMS presents an accelerated convergence and superior tracking abilities, its cascaded form introduces high latency when targeting a hardware implementation with a division operation such as in (3.25). Furthermore, the system complexity is of order $O(N^2)$ and is increased by that of the LMS stage and an additional $20N$ multiplications, $6N$ additions and $2N$ divisions for the steering vector estimate in (3.25). Consequently, the cascading nature of the RLMS introduces high latency, given the second stage dependency

on the steering vector estimate block, making it difficult to be implemented in a parallel and pipeline architecture [4, 21, 24].

3.3.5 Parallel RLMS Adaptive Beamformer

In order to present a low latency, division free architecture, suitable for a parallel implementation, we propose in [6] a two-stage parallel input RLMS (RLMSp) structure, as shown in Figure 3.3, where jz^{-1} denotes a multiplication by the imaginary number j with respect to a one sample delay.

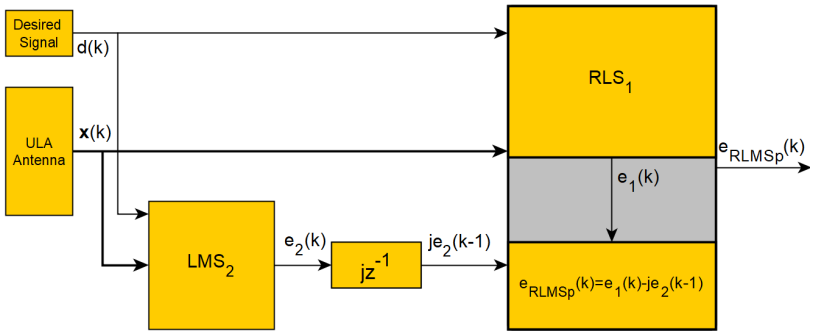

Figure 3.3 – Parallel Input RLMS (RLMSp)

From Figure 3.3, it is clear that the LMS_2 input is the signal samples and no longer a reconstructed input dependent on the array steering vector estimation. Moreover, the LMS_2 error signal $e_2(k)$ is now multiplied by the complex number j to protect against nulls and recurring samples [4]. The overall RLMSp error signal, $e_{RLMSp}(k)$, and weight update equations are [15]:

$$e_{RLMSp}(k) = e_1(k) - je_2(k-1) \tag{3.30}$$

$$\mathbf{w}_1(k) = \mathbf{w}_1(k-1) + \mathbf{L}(\mathrm{k})\mathbf{x}(k)e^*_{RLMSp}(k) \tag{3.31}$$

Recurring samples in the input and/or desired signals are usually formed as a consequence of repeated samples in the original message signal, analog to digital converter (ADC) impurities [57], quantization errors [34, 35, 36], low resolution resulting in receivers saturation [4, 17] and of symbol detection errors originated from the digital receivers in low signal to

noise ratio (SNR) environments [4, 58]. As such, in the adopted delay feedback method described by (3.28), recurring samples can result in error nulls and severely degrade the convergence performance of the RLMSp [4, 6, 14, 24]. As an example, we consider a sample of the desired signal vector, $\mathbf{d}$, taken at instances k and $k-1$, respectively and represented in its complex form such as:

$$d(k) = a_1 + jb_1 \tag{3.32}$$

$$d(k-1) = a_2 + jb_2 \tag{3.33}$$

Moreover, the modified RLMS and RLMSp desired signals, $S_{RLMS}(k)$ and $S_{RLMSp}(k)$, with respect to (3.28) and (3.30), are:

$$S_{RLMS}(k) = d(k) - d(k-1) \tag{3.34}$$

$$S_{RLMSp}(k) = d(k) - jd(k-1) \tag{3.35}$$

For recurring samples, i.e. $d(k) \approx d(k-1)$ we get $a_1 \approx a_2$ and $b_1 \approx b_2$, thus $S_{RLMS} \approx 0$. However, the RLMSp overall reference signal, $S_{RLMSp}(k)$, becomes:

$$S_{RLMSp}(k) = a_1 + jb_1 - ja_2 + b_2 \neq 0 \tag{3.36}$$

Therefore, from (3.36), the multiplication by the imaginary number j improves the systems robustness against repeated samples [4]. By eliminating the need for a steering vector estimate stage, the RLMSp complexity, compared to the RLMS, is reduced by $20N$ multiplications, $6N$ additions and $2N$ divisions. However, the RLMSp computational complexity is still of order $O(N^2)$ and its pipeline remains difficult due to the presence of error feedback paths [4, 21, 24].

3.3.6 LLMS Adaptive Beamformer

The least mean square-least mean square (LLMS) algorithm is a multi-stage LMS separated by an estimate of the array steering vector [5, 55] as shown in Figure 3.4. In LLMS, the output of the first stage LMS_1, $y_{LMS1}(k)$, is multiplied by the estimate of the

desired signals steering vector $\hat{\mathbf{a}}_d(k)$, forming the input, $\mathbf{x}_2(k)$, to the LMS_2 stage similar to (3.25) and (3.27).

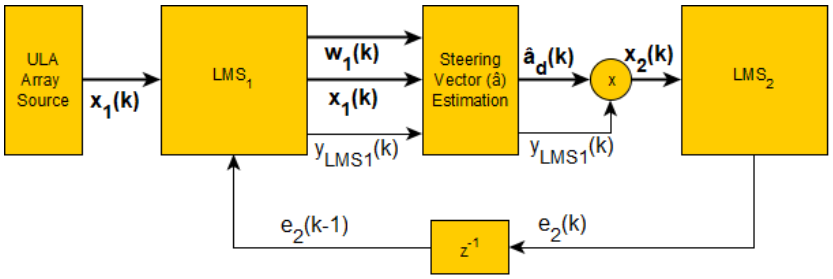

Figure 3.4 – Cascaded LLMS

Thus, the input to the LMS_2 with respect to $y_{LMS1}(k)$ becomes:

$$x_{2,m}(k) = \frac{w_{1,m}(k)x_{1,m}(k)}{w_{1,m}(k)y_{LMS1}(k) + \vartheta} y_{LMS1}(k) \tag{3.37}$$

Moreover, a delayed version of the error signal $e_2(k)$ of the LMS_2 stage is fed-back to combine with that of the LMS_1 and form the overall error signal $e_{LLMS}(k)$ used to update the main tap weights of the LMS_1 stage. As such, the LLMS overall error and weight update equations are presented as [5, 55]:

$$e_{LLMS}(k) = e_1(k) - e_2(k-1) \tag{3.38}$$

$$\mathbf{w}_1(k+1) = \mathbf{w}_1(k) + \mu_1 e^*_{LLMS}(k)\mathbf{x}_1(k) \tag{3.39}$$

The input and desired signals are assumed independent and identically distributed (i.i.d) with zero mean. The process is assumed WSS with no temporal correlation between samples [5].

While the LLMS presents an accelerated convergence with a low steady state error, it does not ensure the same convergence properties against recurring/repeated samples, i.e. $\mathbf{x}(k) \approx \mathbf{x}(k-1)$ and $d(k) \approx d(k-1)$. Recurring samples are formed as a consequence of a high sampling frequency or re-transmission for robustness. Also, its cascading structure introduces a large increase in computational complexity and latency [4]. Furthermore,

the system complexity is increased by an additional $20N$ real multiplications, $6N$ real additions and $2N$ real divisions for the steering vector estimate in (3.25). Consequently, the division operator affects the systems stability when targeting a hardware implementation [59], i.e. possibility of division by zero in finite precision arithmetic. Similar to the RLMS, the cascading nature of the LLMS introduces a high latency, given the dependency on two error feedback paths, making it difficult to be implemented in a pipeline architecture [4, 21, 24].

Algorithm	Initial Parameters
LMS	$\mu_{LMS} = 0.04$
RLS	$\alpha = 0.98$, $\mathbf{L}(0) = 0.5\mathbf{I}$, $\mathbf{Q}(0) = 0.025\mathbf{I}$
VFFRLS	$\lambda_{max} = 1$, $\gamma_{VFF} = 1.5$, $k_\alpha = k_\beta = 6$ $\varepsilon_{VFF} = 1$, $\delta = 1$
MRVSS	$\tilde{e}_{max} = 1$, $\tilde{e}_{min} = 0$, $\nu = 5 \times 10^{-4}$ $\mu_{max} = 0.2$, $\mu_{min} = 10^{-4}$ Initially: $\mu_{MRVSS} = \mu_{max}$, $\alpha = \eta = 0.97$ $\gamma = 4.8 \times 10^{-4}$
RLMS	$\alpha = 0.98$, $\mathbf{L}(0) = 0.5\mathbf{I}$, $\mathbf{Q}(0) = 0.025\mathbf{I}$ $\mu_{LMS2} = 0.2$, $\vartheta = 0.0004$
LLMS	$\mu_{LMS1} = 0.04$, $\mu_{LMS2} = 0.04$, $\vartheta = 0.0004$
RLMSp	$\alpha = 0.98$, $\mathbf{L}(0) = 0.5\mathbf{I}$, $\mathbf{Q}(0) = 0.025\mathbf{I}$ $\mu_{LMS2} = 0.2$

Table 3.1 – Simulation Initial Parameters

3.3.7 Comparison and Discussion

The simulation setups for evaluating the performance of the previously listed algorithms are as follows:

— ULA array with $N = 8$ antenna elements.

— A message signal and two interferes arriving at AOA of 30°, 45° and 80°, respectively. The inputs are generated as independent random complex Gaussian sequences [4, 60]

of the form $v_m = v_1 + jv_2$, where v_1 and v_2 are taken from a normal (Gaussian) distribution with zero mean and variance σ^2 of the form $\mathcal{N}(0, \sigma_p^2)$ and $\mathcal{N}(0, \sigma_q^2)$, respectively, where, $\sigma_p = 0.1$ and $\sigma_q = 0.2$. The subscripts p and q denote in-phase and quadrature, out of phase, respectively [4].

— With respect to the simulation criteria in [5], and since the pilot, training, signal is known, the desired signal $d(k)$, is considered as a copy of the message signal corrupted by CAWGN noise with a signal to noise ratio of SNR = 5 dB and SNR = 10 dB, respectively.

The initial parameters were chosen similar to that selected in [5, 33] and are presented in Table 3.1 where $\mathbf{I}$ is a $N \times N$ identity matrix.

The simulation is conducted following a Monte Carlo approach [4] with 150 realizations of 150 samples each and the performance of the adaptive algorithms is evaluated with respect to the resulting MSE.

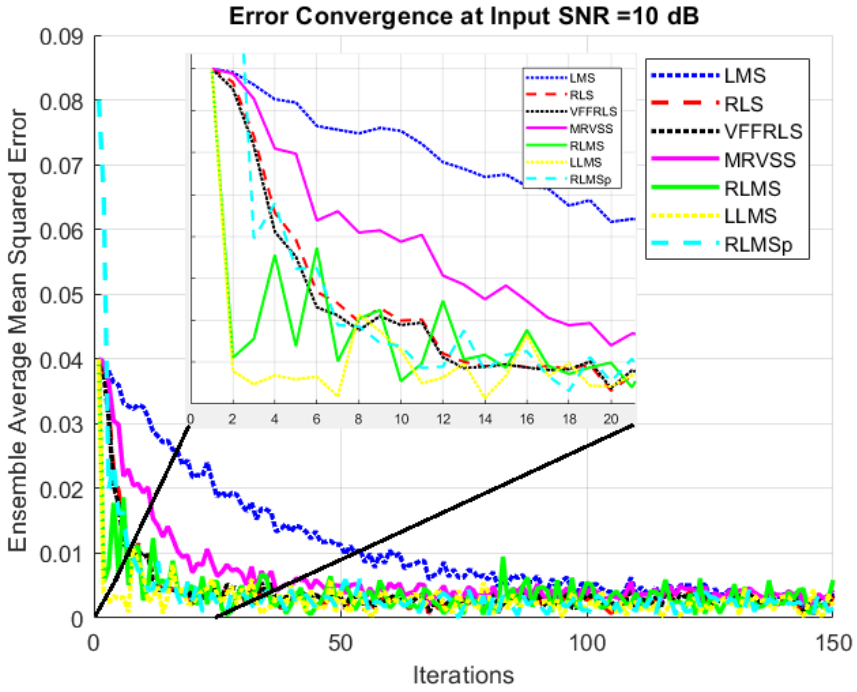

Figure 3.5 – MSE Convergence Behavior For The LMS and RLS Adaptive Beamformers and Their Variants For SNR = 10 dB

From Figure 3.5, all algorithms achieved convergence for a SNR = 10 dB. In contrast to the other variants, it can be seen, that only the LLMS and RLMS achieved their convergence from the 2^{nd} iteration with a minimal steady state error, i.e. MSE < 0.01.

To better assess their behavior in noisy environments, the simulations were repeated for a SNR = 5 dB.

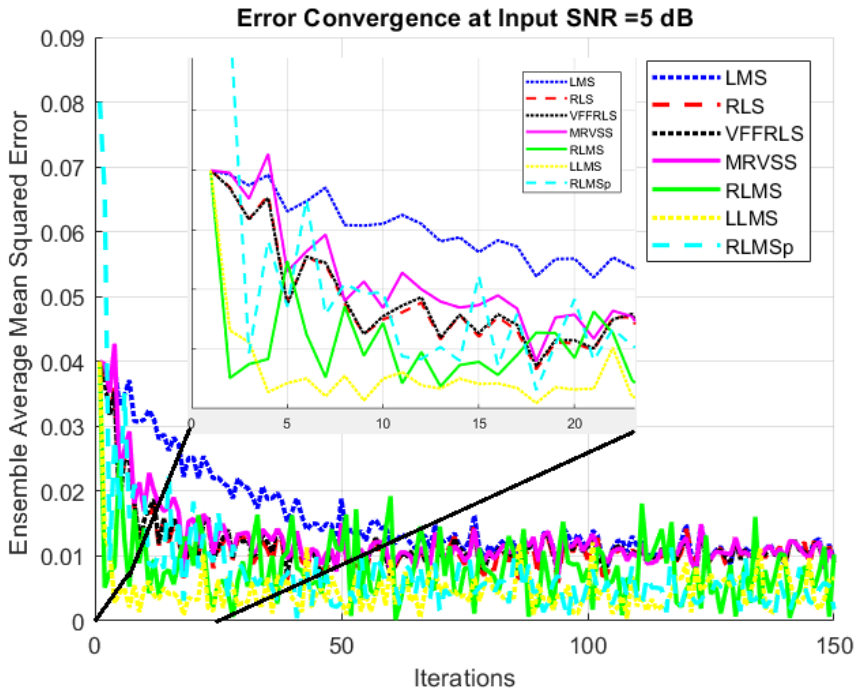

Figure 3.6 – MSE Convergence Behavior For The LMS and RLS Adaptive Beamformers and Their Variants For SNR = 5 dB

As shown in Figure 3.6, only the LLMS preserved an accelerated convergence and low steady state error profile in low SNR environments. In contrast to the RLMS, the RLMSp achieved a lower steady state error however with a slightly delayed convergence.

Table 3.2, presents a comparison of resource complexity where cMultiply, cAdd, cDivide define as complex multiplication, addition or division, respectively. From Table 3.2, it is clear that the RLMS, RLMSp and RLS require an undesirable complexity of order $O(N^2)$ and the need of a division operation. On the other hand, only the LLMS achieved

accelerated convergence and low steady error in low SNR comparison with a complexity of order $O(N)$. However, compared to the LMS, the LLMS doubles its resource requirements in addition to N complex divisions [4].

Algorithm	cMultiply	cAdd	cDivide
RLMS[15]	$3N^2 + 11N + 2$	$2N^2 + 9N + 6$	$N + 1$
RLMSp[6]	$3N^2 + 7N + 1$	$2N^2 + 6N + 3$	1
RLS	$3N^2 + 5N$	$2N^2 + 4N + 2$	1
LLMS[5]	$6N + 2$	$5N + 4$	N
LMS	$2N + 1$	$2N + 1$	0

Table 3.2 – Theoretical Complexity and Resource Usage

3.4 Conclusion

In this chapter, we introduced the different classes of beamforming algorithms, i.e. non-blind, semi-blind and blind. Additionally, we presented an overview of the most popular non-blind temporal and spatial referenced adaptive algorithms and some of their variants, while listing the advantages and disadvantages of each. To further illustrate their behavior and convergence properties, many simulations were conducted with respect to random complex Gaussian input sequences. The output MSE plot was used to validate and compare the algorithms performance, in terms of convergence speed and steady state error, for different SNR environments. Thus, in contrast to the LMS, RLS and their variants, only the LLMS achieved superior performance through an accelerated convergence and low steady state error profile. However, while the LLMS preserved an order $O(N)$ complexity, it still requires N complex divisions for the steering vector estimate. Additionally, the cascading nature of the LLMS and the error feedback paths makes it difficult to be implemented in a parallel and pipeline architecture. As such, our main motivation is to present a reduced complexity multi stage LMS variant suitable for a parallel pipeline hardware implementation while maintaining accelerated convergence, low steady state error and robustness against low SNR environments. The work conducted for the RLMSp has been published in the European Signal Processing Conference (EUSIPCO).

Chapter 4

Overview of Digital Signal Processing Implementation Techniques on Embedded Systems

4.1 Introduction

Signal processing is a multidisciplinary field of study and have become a vital technology in many modern applications [25, 61]. Applications embodying signal processing tasks and routines are endless, and rely on both analog and digital processes, however by the end of the 20^{th} century this discipline have become increasingly dominated by the digital field [61, 62]. Some examples of common digital signal processing (DSP) dependent areas are but not limited to: communication [4], machine learning [63], biomedical and healthcare [64], radar or sonar imaging [65, 66], instrumentation [57], information technology [67, 68, 69, 70] and seismology [2].

Historically, electronic signals and radio waves were processed and exchanged in their analog form, with the use of analog chips, filters and amplifiers, and with minimal control over information quality and reliability [61, 62]. However, the emergence of new technologies in the $1960s$ and $1970s$, i.e. satellite transmission, pushed the adopted analog signal processing design methodology and its components to their limits [61]. An example of a satellite transmission system is the broadcasting station: Pleumeur-Bodou Ground Station in north-west France.

Figure 4.1 – Pleumeur-Bodou Ground Station For Satellite Broadcasting [71]

As shown in Figure 4.1 [71, 72], the ground station and its antenna occupied a tremendous amount of space and resources to build [72]. Moreover, in order to detect and estimate the signal of interest with respect to the incoming sub-GHz high frequency noisy signal, a data acquisition system was implemented by means of multi-stage analog signal amplifiers and filters with a dedicated cooling unit [72].

As such, the rapid advancements in signal processing [61, 62] exhausted the use of analog circuitry designs by imposing tighter constraints reflected by the need of a low cost, real-time processing system [62]. In compliance with these advancements, and with the introduction of the programmable, finite precision, DSP processor (PDSP), the field of DSP have emerged to become the newly adopted signal processing design methodology [25, 62].

In this chapter, we present an overview of the popular hardware used for implementing signal processing routines, i.e. PDSP, field programmable gate array (FPGA) and system on a chip (SoC). Additionally, we discuss the different DSP implementation techniques and tools used, i.e. hardware description language (HDL), high level synthesis tools (HLS) and open computing language (openCL). Furthermore, we propose a high precision, low complexity, pipeline, dynamic fast Fourier transform (FFT) twiddle factor generation architecture using Chebyshev polynomials [59]. Finally, we compare the performance of the different design techniques and utilized hardware through the implementation of popular signal processing routines, i.e. the FFT and the least mean square (LMS) algorithm.

4.2 Embedded Systems for Digital Signal Processing (DSP) Applications

One of the most trivial signal processing tasks is filtering. Filters are used to mathematically modify the content of a signal, through the use of multiplication and addition, in order to either extract or dispose of information or noise [25]. A major contributor to the acceleration of DSP development in the 1970s was the introduction of the PDSP capable of performing a single, fixed point, multiply accumulate (MAC) routine in one clock cycle [25]. As a result, a broad range of dedicated signal processing hardware and design tools have been put forth to pave the way for the development of atomic, fine grained [62, 73], signal processing routines and hardware architectures [25, 62, 73]. A general DSP architecture is presented in Figure 4.2.

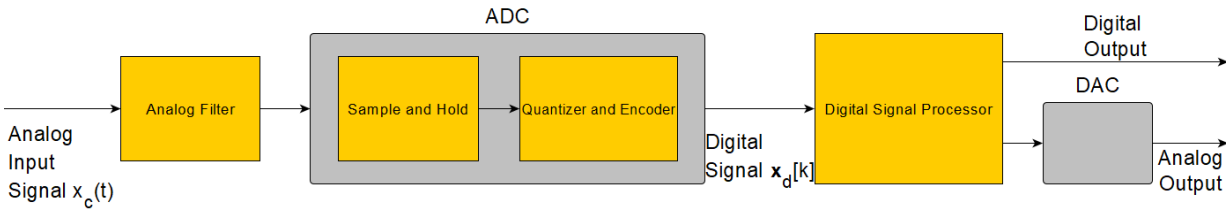

Figure 4.2 – General DSP System Structure

As shown in Figure 4.2, the DSP system is generally formed of an input analog filter usually acting as an anti-aliasing filter with respect to the sampling frequency f_s and the Nyquist theorem [25]. The anti aliasing filter removes unwanted mirror frequencies to avoid data ambiguity. The resulting signal is then fed to an analog to digital converter (ADC) which in turn samples and converts the continuous analog signal, $x_c(t)$, where t is a time instance, into its equivalent discrete digital form [17, 25, 35, 36]. The resulting digital output vector $\mathbf{x}_d[k]$, where k is a discrete sample time index, is then collected by the DSP processor, i.e. PDSP, FPGA, SoC..., to perform the desired steps and mathematical procedures that would have been previously handled by an analog circuit [25]. The final stage of the data flow is the processed signals output, hence as shown, the desired output can be obtained in its digital form for later storage and processing or converted to its

original analog form through the use of a digital to analog converter (DAC), i.e. audio output [25].

Parallel to these developments, programmable processors and VLSI technology are quickly evolving towards the design of heterogeneous systems with high performance, application specific multi core architecture [73]. Thus, in this section, we present a summary of the popular and currently adopted processors, i.e. PDSP, FPGA and SoC.

4.2.1 Programmable DSP Processors (PDSP)

In the start of the 1980s, Texas Instruments introduced the first reduced instruction set computer (RISC) PDSP microprocessor that revolutionized the era of DSP [25, 62]. In contrast to the first generation PDSP, based on the Von Neumann architecture, the second generation PDSP, based on the Harvard architecture, effectively separated the data memory from the program memory. Consequently, the RISC architecture allows independent and uninterrupted data communication between the processor pipeline and data memory bus [25, 62]. Additionally, in contrast to general purpose processors, the PDSP are application specific and can operate in real time with a set of optimized DSP instruction set, i.e. fixed point computation, saturation and overflow control, overlapped data fetch and advanced addressing modes [62, 73]. As such, the PDSP was capable of performing a fixed point MAC operation in a single clock cycle with a minimal overhead, considered the basis of digital filters and most DSP operations [25].

Over the past three decades, DSP applications and requirements changed drastically and imposed challenging constraints on the limited resources PDSP [62]. These constraints mainly focused on the need of providing a high speed parallel implementation given the parallel nature of the DSP algorithms. However, since PDSP are single core application specific processors with a pre determined instruction set and limited arithmetic units, it is impossible to reprogram their structure and redistribute resource usage. Moreover, a parallel implementation would require the use of many PDSP modules resulting in a dramatic increase in complexity and latency [62, 73]. Thus, it has become of utmost importance to migrate for field programmable semiconductor chips, i.e. FPGA, that offer resource manipulation for implementing a parallel architecture [62, 73].

4.2.2 Field Programmable Gate Arrays (FPGA) Processors

Field Programmable Gate Array or FPGA is a re-programmable semiconductor device formed of a 2D array of repeated digital configurable logic circuit blocks (CLB) connected by means of an interconnect fabric. Each CLB consists of four logic slices each containing a small number of look up tables, registers and a programmable switch box (SW) connecting the CLB to the internal fabric [25, 73].

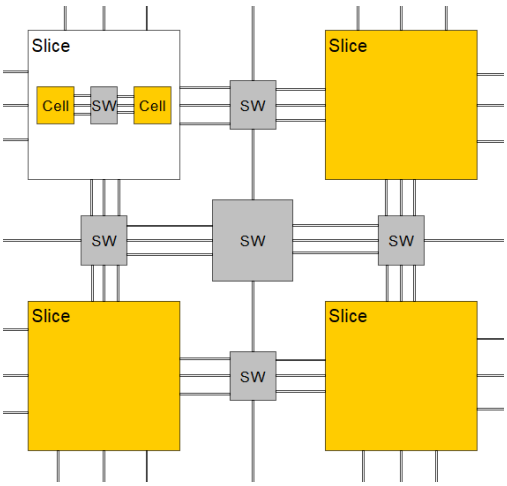

Figure 4.3 – Xilinx Spartan 3 Simplified FPGA CLB Fabric

As shown in Figure 4.3, in Xilinx Spartan 3 FPGAs, one CLB is formed of four interconnected slices of two logic cells each [74]. Moreover, a Xilinx Spartan 3 FPGA logic cell is formed of a four input look up table, a register element, a carry circuit for arithmetic operations and a multiplexing circuit [74].

The re-configurable nature of the FPGA logic cells allows implementing generic arbitrary and parallel, application specific architectures. A FPGA logic design is defined as a soft IP-core and can be modeled using a wide range of hardware and software design techniques [73]. These techniques are and not limited to: circuit schematics, hardware description languages, i.e. VHDL, Verilog, SystemVerilog and traditional software languages, i.e. C, synthesized with the use of high level synthesis (HLS) tools, such as the Xilinx Vivado HLS compiler, Intel HLS tools and Matlab HDL toolbox [62, 75]. In addition to the traditional logic blocks, some FPGA families include a wide range of hard-IP, macro,

blocks for frequently used components and functionalities [73, 74]. Some of the included hard-IP blocks are and not limited to: fast storage block random access memory (BRAM), input/output blocks (IOB), high speed dedicated multipliers, phase locked loop (PLL) blocks, embedded microprocessor, ADC and recently DSP specific blocks [25, 62, 73]. As such, an important FPGA benchmark parameter is the repetition rate and represents the number of realization of the same block in a device [25].

In modern FPGAs, manufacturers provide specific FPGA families and architectures for DSP related operation through the inclusion of a number of dedicated, transistor level, re-configurable DSP blocks [25, 62, 73]. An example of a DSP block is shown in Figure 4.4 from [76], for the Altera Stratix V FPGA family.

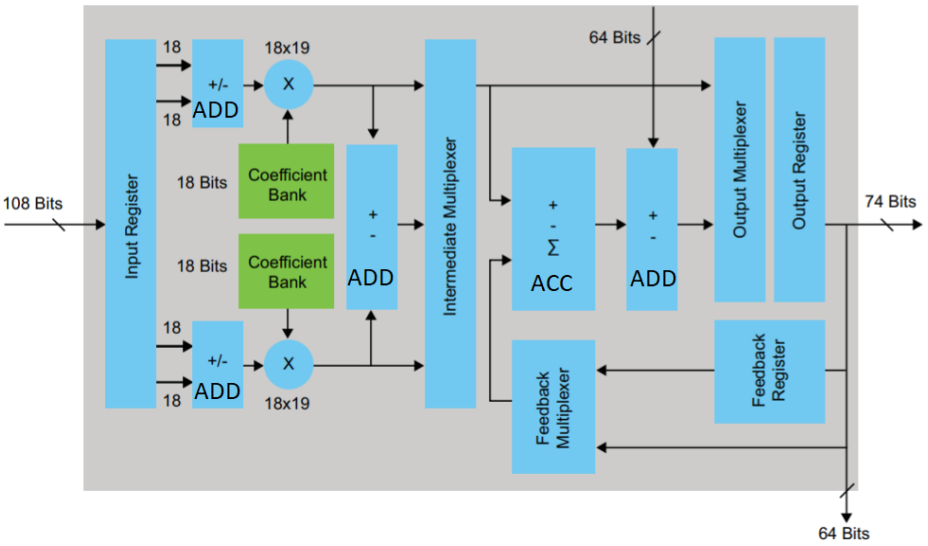

Figure 4.4 – Altera Stratix V Variable Precision DSP Block in Standard Precision Mode [76]

As shown in Figure 4.4, the Stratix V DSP block is configured in standard, 18 bits, precision mode while providing dedicated, high speed, arithmetic hardware and storage elements for computing popular DSP operations, i.e. MAC, pre-addition and complex multiplication [62, 73, 76]. The Stratix V, standard mode, DSP block allows the following multiplier configurations:

— 9×9 three independent multipliers.

— Two 18×18 independent multipliers in sum mode.

— Two 16×16 independent multipliers in sum mode.

— One 18×36 independent multiplier.

In addition to the presented, the Stratix V DSP block can be configured in high precision mode.

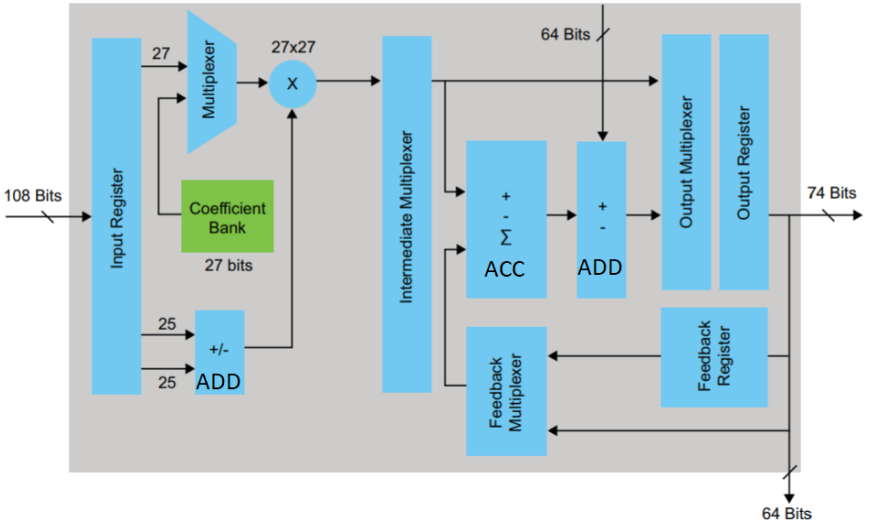

Figure 4.5 – Altera Stratix V Variable Precision DSP Block in High Precision Mode [76]

As shown in Figure 4.5, the high precision mode configuration implements one 27×27 independent multiplier. As such, the use of re-configurable devices in the deployment of DSP algorithms promises immeasurable aid in the advancement of the applications they embody [61, 62]. Thus, the main benefits adopting re-configurable computing techniques and devices for DSP can be summarized as:

— Implementing a dedicated and specialized architecture.

— Allowing low cost on field full or partial re-configuration.

— Implementing fine-grained [25] parallel pipeline architecture.

However, recent trends in DSP applications require the use of complex algorithms, i.e. adaptive, statistical, in parallel with real time data acquisition, processing and decision making routines. As such, modern FPGA families are coupled with a high performance

multi-core hard intellectual property (IP) microprocessor, i.e. ARM Cortex A9, with a ready to use real time operating system. The FPGA and hard embedded processor architecture is referred to as SoC or system on a chip, and is used to off load certain, non critical and data processing tasks, from the FPGA for efficient resource usage.

4.2.3 System on Chip (SoC) Processors

Resource abundant FPGA families allow an efficient implementation of a soft-IP microprocessor in order to off load non critical data handling tasks from the FPGA pipeline [25]. However, implementing a soft-IP microprocessor infer severe performance limitations on the host FPGA mainly by restricting its maximum operating frequency. Additionally, implementing a full pipeline microprocessor would require a larger FPGA thus greatly increasing the cost for a marginal increase in performance [25, 62, 73]. As such, and with the exponential growth in demand on a small, general purpose, low power and high performing embedded system, computer architectures are rapidly evolving toward heterogeneous multi-core systems, i.e SoC [25, 62, 73, 76]. While it is required to have a general purpose embedded system without a loss of performance, SoC models are equipped with many-cores and macro blocks each tailored for a specific application [25]. A popular, low to mid end, SoC model is the Altera Cyclone V SoC and incorporates a dual core, general purpose ARM cortex A9 microprocessor, a Cyclone V FPGA with multiple variable precision, programmable DSP cores. The ARM cortex A9 architecture is shown in Figure 4.6.

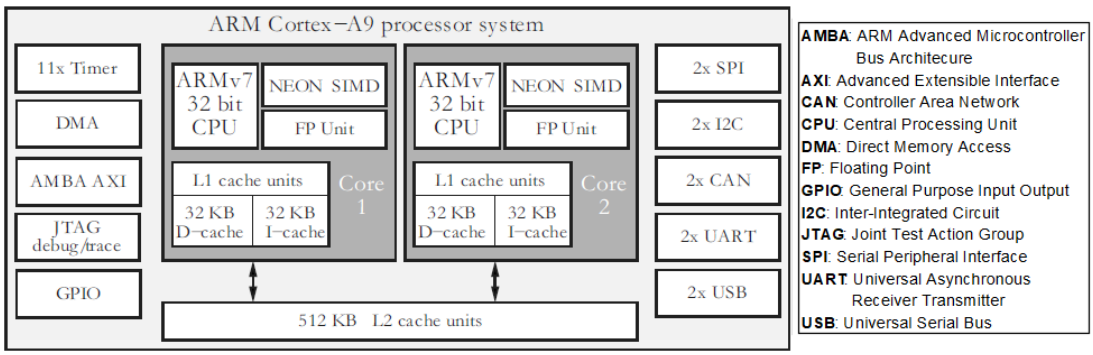

Figure 4.6 – ARM Cortex A9 Architecture [25]

Unlike a soft-IP microprocessor, the advanced RISC machines (ARM) cortex A9 is equipped with two 32bits. 800 MHz, hard processors each with a dedicated floating point unit (FPU) [25]. Moreover, the processor system provides a direct memory access (DMA) block, multiple timers and supports a variety of communication protocols with ready to use hardware, i.e. universal serial bus (USB), universal asynchronous receiver transmitter (UART), inter-integrated circuit (I2C), controller area network (CAN) and serial peripheral interface (SPI), with high speed memory units. Hence, the use of a dedicated microprocessor with programmable resources, frees critical FPGA resources, contributes to implementing a parallel, uninterrupted pipeline, architecture and accelerates the overall systems performance. However, programming a SoC is a multidisciplinary task and requires excessive software and hardware engineering skills [25, 75]. A comparative study based on various cost metrics is performed on the PDSP, the FPGA, the SoC and the general purpose processor (GPP).

	Performance	Flexibility	Power	Cost	Design Effort
PDSP	Medium	Medium	Medium	Medium	Medium
FPGA	Medium	High	High	Medium	Medium
SoC	High	High	Medium	Medium	High
GPP	Low	High	Medium	Low	Low

Table 4.1 – Implementation Comparison for DSP Applications [62]

As shown in Table 4.1, eventhought the SoC platform is the highest performing, it requires an extensive design effort and a variety of skills in order to present an optimized and efficient design. As SoC development requires the use of a set of unified programming tools and languages, i.e. HLS, to efficiently target heterogeneous systems for both hardware and software design and off load performance optimization to the compiler [75].

4.3 Heterogeneous Systems Design Techniques

While specialized hardware and application specific circuits provide considerable performance boost, the main drawback resides in their design and the limitations of their

synthesize tools [75]. For over a decade, researchers and engineers unified their efforts in optimizing embedded system design techniques and compilers for heterogeneous systems with multi core processors. A popular and currently adopted solution for the processor diversity problem is the use of higher levels of abstraction with the help of an optimized software compiler to describe hardware functionality [75].

Traditional HDL, i.e. VHDL and Verilog, allows the designer to define a cycle by cycle behavioral and structural description of a custom digital circuit. The resulting design is then synthesized to obtain its register transfer level (RTL) description with the use of a logic synthesis tool [75]. However, the use of HDL mandates the user to specify circuits functionality in a low level of abstraction which requires advanced hardware development expertise [75]. As such, hardware developers are migrating from the use of classical HDL design techniques to higher level, object oriented, languages, i.e. SystemVerilog, and synthesis tools, i.e. VivadoHLS [75]. In this section, we provide a simple overview on the most popular adopted HLS design techniques for DSP.

4.3.1 High Level Synthesis (HLS) Design

The increased dependency on heterogeneous systems, its multidisciplinary nature and the need of unified programming techniques paved the way towards accelerating HLS design tools and compilers with the aim of achieving a high performance, energy efficient design [75]. HLS tools implement hardware circuits by automatically converting a software programmable high level language (HLL), i.e. C, python, C++, SystemC, to an equivalent HDL format [75]. HLS design techniques benefit both software and hardware engineers by off loading the problem of optimization to the compiler through the use of compiler specific directives [75, 77]. Thus, allowing them to effectively design complex multi core systems at a high level of abstraction [75]. Some of the popular HLS tools are the VivadoHLS from Xilinx, Matlab HDL coder toolbox from Mathworks and the Intel Quartus Prime suite [73, 75, 77]. An example of some of the frequently used VivadoHLS compiler directives are [77]:

— Unroll: Used in order to infer a parallel architecture for iterative procedures while optimizing execution time and resource usage.

— Pipeline: Used to infer a pipeline architecture for a component or architecture, through the use of additional registers, to increase the maximum operating frequency while decreasing latency and input delay.

— Partition: Used to partition large memory blocks for parallel access.

— Map: Allows the compiler to optimize usage and resource allocation for memory elements.

While the use of HLS tools allows an automatic HDL generation, it will always be dependent on the users experience in adapting the HLL to a specific application structure [75]. As such, the software programmer is required to understand the basics of digital HDL design and properly adapt his code to reflect his circuit structure [75], which is considered as a time-consuming and error prone process. Therefore, recent HLS tools offers complete flexibility and support for the HLL with complete freedom for the developer [75].

4.4 FFT and The Dynamic Twiddle Factor Generator

The Fast Fourier Transform or FFT, initially introduced by Cooley and Tukey (1965), is an efficient implementation of the discrete Fourier transform (DFT) one of the most widely used transforms in DSP. The DFT computes a signals frequency spectrum by converting its finite length time domain sequence of equally spaced samples to its equivalent frequency domain representation with respect to its amplitude, frequency and phase [25, 59]. The DFT equation is given as follows:

$$X[k] = \sum_{n=0}^{N_c-1} x_d[n] e^{-j2\pi nk/N_c} \tag{4.1}$$

where $x_d[n]$ is a sample of the input signal $\mathbf{x}_d$ at time index n and $X[k]$ is its Fourier transform at index k. The popular Cooley Tukey FFT algorithms are those with a sequence of length N_c, where N_c is a power of a basis or a radix r, i.e. $N_c = r^v$ with v representing the number of stages [25, 77]. The FFT algorithm iteratively divides the input sequence into groups of odd and even samples [25, 77] by the use of decimation in time (DIT) and

decimation in frequency (DIF) techniques. Thus, with respect to (4.1), the N_c points FFT achieved through DIF can be computed as follows:

$$\begin{aligned} X[2k] &= \sum_{n=0}^{\frac{N_c}{2}-1} \left\{ x_d[n] + x_d\left[n + \frac{N_c}{2}\right] \right\} W_{N_c/2}^{kn} \\ X[2k+1] &= \sum_{n=0}^{\frac{N_c}{2}-1} W_{N_c}^{K} \left\{ x_d[n] - x_d\left[n + \frac{N_c}{2}\right] \right\} W_{N_c/2}^{-kn} \end{aligned} \tag{4.2}$$

where $W_{N_c}^{K}$ is the rotation or twiddle factor [59, 77] given as:

$$W_{N_c}^{k} = e^{-j2\pi k/N_c} = \cos\left(\frac{2\pi k}{N_c}\right) - j\sin\left(\frac{2\pi k}{N_c}\right) \tag{4.3}$$

Through successive decomposition for (4.2), i.e. divide and conquer, it is shown in [25], that the FFT technique reduces the DFT complexity from $O(N_c^2)$ to $O(N_c \log(N_c))$. A widely used and yet simple example is the FFT radix-2 DIF algorithm for a length $N_c = 8$ samples.

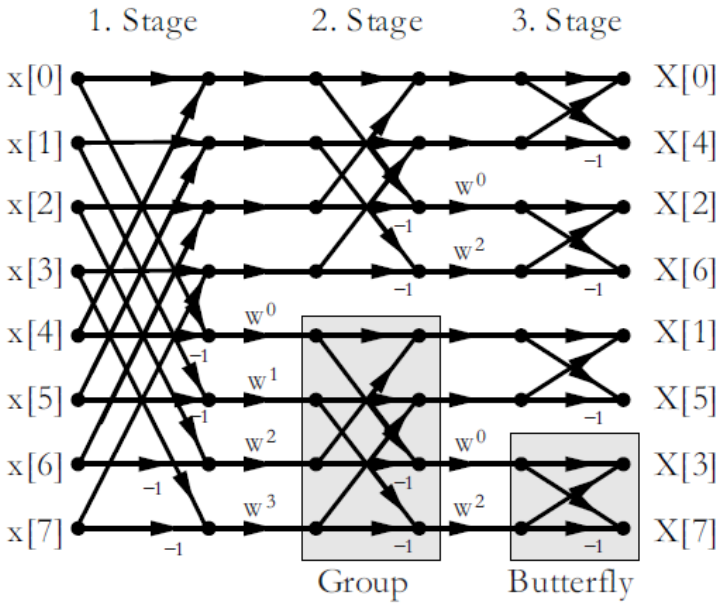

Figure 4.7 – FFT Radix-2 DIF For an Input Sequence of Length $N = 8$ [25]

As shown in Figure 4.7, through continuous decomposition of (4.2), it is clear that the heart of the FFT operation and its building block is the butterfly processor [25]. The butterfly processor is presented in details in Figure 4.8 and performs complex addition, complex multiplication and complex subtraction operations.

Figure 4.8 – FFT Radix-2 Butterfly Processor

From Figure 4.8, the butterfly processor is a two input two output architecture whose output is computed by mean of a twiddle factor multiplication [25, 59, 77].

4.4.1 Twiddle Factor Generator Using Chebyshev Polynomials

When implemented on hardware, in application specific architectures, the FFT butterfly processor rely on pre-computed twiddle factors initially stored in block memory [25, 59]. However, the use of internal memory imposes strict and unavoidable timing and precision constraints on the DSP design, where an increase in precision results in an increase in the memory usage [59]. Additionally, in some DSP applications, generic length FFT is often required and it is of crucial importance to dynamically generate high precision twiddle factors with respect to a low latency and low resource requirements architecture [59].

Popular, dynamic, twiddle factor generators rely on the use of the coordinate rotation digital computer (CORDIC) algorithm [25, 59] or Taylor function approximation [25, 59] to compute the cos and sin trigonometric functions as in (4.3). However, CORDIC approximation requires a large number of iterations in order to achieve higher precision, consequently resulting in a considerably large latency and a non optimal pipeline architecture [59]. In contrast, Taylor polynomial approximation allows the design of a fully pipeline low latency architecture; However, at the cost of a reduced precision the farther the input is from the Taylor expansion point [59].

Several variants and improvements were suggested for the CORDIC and Taylor approximation in order to present a high precision low complexity, easy to pipeline, design

[59, 78, 79, 80, 81, 82, 83]. However, the adopted design techniques and improvements still heavily rely on the use of internal memory elements and could not eliminate the dependency between CORDIC precision and required number of iterations [59]. As such, and since twiddle factors precision greatly affects the overall resolution of the FFT processor, we propose a high speed, high precision, low latency and low complexity architecture using Chebyshev polynomial. Chebyshev polynomials are orthogonal polynomials defined over the interval $[-1, 1]$ and can be calculated in a recursive form [59], such as:

$$\begin{aligned} T_0(u_c) &= 1 \\ T_1(u_c) &= u_c \\ T_{n+1}(u_c) &= 2u_c T_n(u_c) - T_{n-1}(u_c) \end{aligned} \tag{4.4}$$

where u_c is the change of variable mapping an input parameter, p_t, from the interval $[a_c, b_c]$ to $[-1, 1]$ defined as [59]:

$$u_c = \frac{2p_t - b_c - a_c}{b_c - a_c} \tag{4.5}$$

As such, a desired function can be approximated in polynomial form using Chebychev coefficients by expanding (4.4) [59], such as:

$$f(u_c) \cong \sum_{k=0}^{L-1} c_g(k) u_c^k \tag{4.6}$$

where $c_g(k)$ is the k^{th} order Chebyshev polynomial coefficient. Using (4.3), (4.4) and (4.6) the twiddle factors trigonometric terms are approximated, as follows:

$$\begin{aligned} Re\{f(u_c)\} &= \cos(\pi u_c) \\ Im\{f(u_c)\} &= \sin(-\pi u_c) \\ f(u_c) &= \cos(\pi u_c) + j\sin(-\pi u_c) \end{aligned} \tag{4.7}$$

Let the twiddle factor exponent be $p_t = \frac{2k}{N_c}$, thus for $a_c = 0$ and $b_c = 1$ the change of

variable u_c defined by (4.5) becomes:

$$u_c = 2p_t - 1 \tag{4.8}$$

Moreover, for a power of two length FFT input sequence (4.8) can be written as:

$$u_c = \frac{2^2 k}{2^v} - 1 = k2^{2-v} - 1 \tag{4.9}$$

thus eliminating the requirements for a multiply-divide procedure by inferring a right shift operation. The Chebyshev coefficients are computed for a 5^{th} order polynomial approximation are given in Table 4.2.

$f(u_c)$	5^{th} order Chebyshev Polynomial Coefficients					
	$c_g(0)$	$c_g(1)$	$c_g(2)$	$c_g(3)$	$c_g(4)$	$c_g(5)$
sin	-0.9994	0	1.2227	0	-0.2239	0
cos	0	-1.5707	0	0.6435	0	-0.0729

Table 4.2 – 5^{th} order Chebyshev Polynomial Coefficients For sine and cosine Approximations

4.4.2 Computer Simulations

Computer simulations were conducted to study the accuracy of the Chebyshev approximation compared to the traditional Taylor approximation for an input vector, such as $\mathbf{p}_t = [0, 0.125, 0.1667, 0.25, 0.5, 1]$ [59]. For comparison purposes a direct computation of the twiddle factor is performed, in infinite precision without considering any approximation and presented in Table 4.3.

$f(p_t)$	Twiddle Factor Direct Computation					
	$p_t(0)$	$p_t(1)$	$p_t(2)$	$p_t(3)$	$p_t(4)$	$p_t(5)$
sin	0	-0.3827	-0.5000	-0.7071	-1	0
cos	1	0.9239	0.8660	0.7071	0	-1

Table 4.3 – Twiddle Factor Direct Computation

The input vector *pc* was re-evaluated for a 5^{th} order Taylor approximation, over an expansion point $q = 0$, and a 5^{th} order Chebyshev approximation and the simulation result is presented in Tables 4.4 and 4.5, respectively.

$f(p_t)$	Twiddle Factor Computation Using Taylor Approximation					
	$p_t(0)$	$p_t(1)$	$p_t(2)$	$p_t(3)$	$p_t(4)$	$p_t(5)$
sin	0	-0.3827	-0.5000	-0.7071	-1.0045	-0.5240
cos	1	0.9239	0.8661	0.7074	0.2000	0.1239

Table 4.4 – Twiddle Factor Computation Using 5^{th} Order Taylor Approximation

From Table 4.4, it is clear that, compared to the direct computation, the Taylor approximation presents satisfactory results for the first three input terms, i.e. $p_t(0)$, $p_t(1)$, $p_t(2)$. However, as the input deviates further from the expansion point, i.e. $q = 0$, the approximation is no longer credible [59].

$f(u_c)$	Twiddle Factor Computation Using Chebyshev Approximation					
	$u_c(0)$	$u_c(1)$	$u_c(2)$	$u_c(3)$	$u_c(4)$	$u_c(5)$
sin	0	-0.3825	-0.5002	-0.70717	-0.9994	0
cos	1.0001	0.9238	0.8661	0.7072	0	-1.0001

Table 4.5 – Twiddle Factor Computation Using 5^{th} Order Chebyshev Approximation

In contrast to the Taylor approximation in Table 4.4, the Chebyshev approximation, presented in Table 4.5, guarantees a 3 decimal digit accuracy for all input values p. To further validate the Chebyshev approximations performance, the MSE is computed with reference to the result of Table 4.3 and is presented in Table 4.6.

f	Taylor and Chebyshev Infinite Precision Approximation MSE	
	Taylor MSE	Chebyshev MSE
sin	0.0458	8.33×10^{-9}
cos	0.2172	7.415×10^{-8}

Table 4.6 – Taylor and Chebyshev Infinite Precision Approximation MSE

As shown in Table 4.6, the Taylor approximation MSE for both the sine and cosine functions is much greater than that of the Chebyshev, with respect to the direct computation results. Thus, validating the superior accuracy of the Chebyshev approximation.

4.5 Hardware Implementation and Comparison

Given its inherent parallelism and popularity in the majority of the DSP operations, in this section, we implement the FFT radix-2 processor for a length $N_c = 8$ input sequence [77]. The FFT implementation is conducted using HDL and HLS design techniques and for different processors, i.e. FPGA and SoC [77]. Additionally, we present a hardware implementation for the low complexity, high accuracy dynamic twiddle factor generator using Chebyshev polynomials [59]. Finally, we implement a pipeline LMS adaptive beamformer using the relaxed look ahead technique as presented in [21] and we evaluate its performance under a finite precision arithmetic.

4.5.1 Fast Fourier Transform (FFT) Design Using HDL

The FFT radix-2 is first implemented in VHDL for a sequence of length $N_c = 8$ using DIF and under finite precision wordlength, i.e. 16 bits signed representation [77]. The implementation is conducted for different architectures, i.e. sequential and fully parallel, and for different processors, such as: FPGA and SoC. The adopted processors are the "Xilinx ZynQ 7Z020CLG484" SoC and the "Intel/Altera Cyclone IV EP4CE115F29C7" FPGA.

Name	Type	Logic Units	DSP	Multipliers	Registers
ZynQ	SoC	85000	220 18×25 bits	From DSP	106400
Cyclone IV	FPGA	114480	-	532 9×9 bits	NA

Table 4.7 – ZynQ and Cyclone IV Resource Comparison

As shown in Table 4.7, the "Cyclone IV" FPGA includes 532, 9×9 bits, ready to use,

high speed, transistor level multipliers. On the contrast, the "ZynQ" includes 220 DSP blocks expanding the processors capabilities to perform vital DSP operations, i.e. MAC, complex operations and pre-additions, in high speed, dedicated, transistor level circuitry [77]. The experiment was first conducted on the "ZynQ" SoC processor and synthesis results are presented in Table 4.8.

The implementation is studied for two different architectures, i.e. parallel and sequential. The "parallel" architecture is a full parallel, pipeline, implementation of a length $N_c = 8$ radix-2 DIF FFT [77]. The architecture is formed of $v = 3$ stages and a total of five butterfly processors [59, 77]. The parallel architecture achieved a maximum operating frequency of 196.539 MHz while using 20 DSP blocks, 3504 registers and 1680 logic units. DSP block allocation adheres with the presented structure in Figure 4.7 and the adopted wordlength since a full three stages radix-2 FFT requires five complex multipliers of four real multipliers each [77].

Architecture	ZynQ Synthesis Results			
	DSP	Registers	Logic Units	Clock (MHz)
Parallel	20	3504	1680	196.539
Sequential	16	1264	560	198.087

Table 4.8 – FFT Radix-2 DIF ZynQ Implementation and Synthesis Results

In contrast to the parallel architecture, a sequential structure was considered and implemented with similar simulation conditions [77]. The sequential architecture is formed only of one processing stage with four pipelined butterflies each [77]. As shown in Table 4.8, the sequential architecture operates at a maximum frequency of 198.087 MHz while using 16 DSP blocks, 1264 registers and 560 logic elements. Compared to the parallel implementation, the sequential design minimizes resource utilization, however at the cost of an increase in the latency and the input delay [77]. The implementation was then repeated for the "Cyclone IV" FPGA, synthesis results are displayed in Table 4.9.

The "Cyclone IV" allocates 40 and 32 multipliers for a parallel and sequential architecture, respectively. Compared to the "ZynQ", the "Cyclone IV" doubles the required multipliers due to the fact that the embedded multipliers support 9×9 bits multiplications

while the "ZynQ" DSP blocks are of 18×25 bits, as shown in Table 4.7.

Architecture	Cyclone IV Synthesis Results			
	Multipliers	Registers	Logic Units	Clock (MHz)
Parallel	40	1296	2113	251.760
Sequential	32	432	705	286.53

Table 4.9 – FFT Radix-2 DIF Cyclone IV Implementation and Synthesis Results

Moreover, the difference in the maximum operating frequency is caused by the processor pre-set timing constraints, compiler efficiency and transistor switching time [25, 77].

4.5.2 Fast Fourier Transform (FFT) Design Using HLS

The VHDL FFT implementation experiment was repeated, using HLS, with respect to the same design parameters and for the ZynQ SoC processor. The HLS design is conducted using a hardware optimized C programming language and different compiler directives [77]. The adopted pre-processing directives are: pipeline and unroll and are used to instruct the compiler to generate specific architectures, i.e. parallel, sequential and pipeline [77]. The FFT radix-2 DIF was first implemented without the use of "For loops" and evaluated for two different cases: without any directive and with the pipeline directive. Implementation and synthesis results are presented in Table 4.10.

Directives	FFT HLS Synthesis Results on ZynQ (Without For)				
	DSP	Registers	Logic Units	Latency	Input Delay
NA	16	1407	578	32	33
Pipeline	12	1283	453	18	4

Table 4.10 – ZynQ FFT Radix-2 DIF Implementation Using HLS (Without For Loop)

The design utilizes 16 DSP blocks and 12 DSP for a standard and pipeline architecture, respectively. Moreover, in contrast to the standard implementation with no directives, the pipeline directive optimize resource utilization while inferring a reduction in latency and input delay [77]. As such, HLS designs are highly correlated with the compilers efficiency

and with the use of the proper directives [77]. To further illustrate the effects of the compiler directives and their crucial role in HLS design, the FFT DIF radix-2 design is re-evaluated using "For loops" and with the pipeline and unroll directives [77]. Synthesis and implementation results are presented in Table 4.11.

Directives	FFT HLS Synthesis Results on ZynQ (With For)				
	DSP	Registers	Logic Units	Latency	Input Delay
NA	4	382	541	45	46
Pipeline FFT	12	1283	453	18	4
Pipeline For	4	404	471	23	24
Unroll For	16	1407	578	23	33
Unroll FFT	4	382	541	45	46

Table 4.11 – ZynQ FFT Radix-2 DIF Implementation Using HLS (With For Loop)

The "For loop" implementation without the use of compiler directives and the unroll FFT both results in a fully sequential scheme with only 1 butterfly, i.e. 4 DSP blocks. The sequential architecture is further validated by the large latency and input delay of 45 and 46, respectively. The resulting sequential architecture is a consequence of the use of a non optimized "For loop" implementation [77]. In contrast, it can be seen that the use of the unroll directive for the for loop design reduces the input delay, from 46 to 33 clock cycles and increases the DSP block usage to 16. As such, it can be concluded the unroll directed the HLS compiler to implement the for loop in parallel [77]. Therefore, it can be concluded that the HLS design technique is severely affected by the users experience and the proper choice of directive to obtain an efficient architecture [77].

4.5.3 Twiddle Factor Generation Using Chebyshev Polynomials in HDL

To further assess the performance of the twiddle factor generator using Chebyshev polynomials, a low complexity, pipeline, hardware architecture is presented. The 5^{th} order Chebyshev approximation is implemented using 18 bits signed wordlength and its finite

precision output is compared to that obtained by the direct and the Taylor schemes [59]. The implementation is performed for the "Xilinx ZynQ" and "Intel/Altera Cyclone V 5CSEMA5F31C6" SoC. Where, the "Cyclone V" SoC equips 85000 logic units, 128300 registers and 87 variable precision DSP blocks [59]. Using (4.6) and (4.7), we can write:

$$\begin{aligned} \cos(\pi u_c) &\cong u_c(c_g(0) + u_c^2(c_g(3) + c_g(5)u_c^2)) \\ \sin(-\pi u_c) &\cong c_g(0) + u_c^2(c_g(2) + c_g(4)u_c^2) \end{aligned} \tag{4.10}$$

Thus, with respect to (4.10), the resulting hardware architecture is shown in Figure 4.9, where $\gg$ denotes a shift right operation.

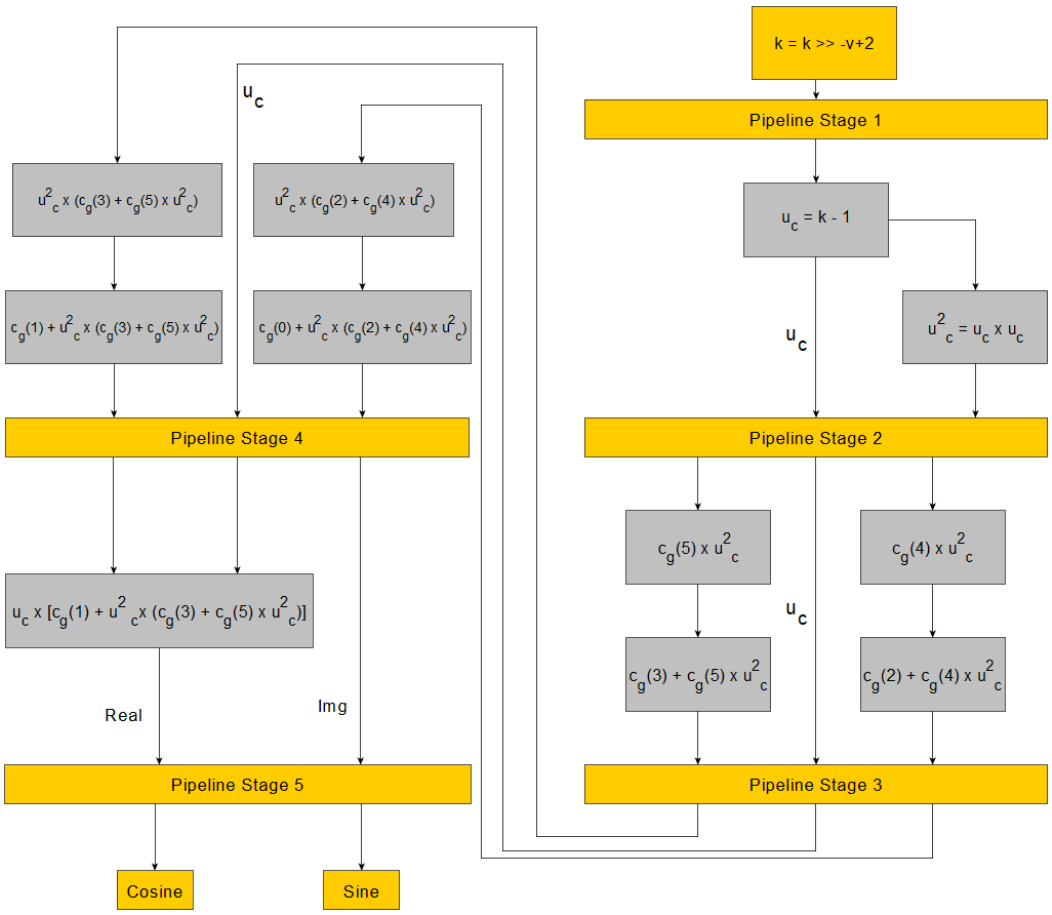

Figure 4.9 – 5^{th} Order Polynomial Hardware Architecture

As shown in Figure 4.9, the polynomial approximation is implemented in a pipeline parallel structure and can be used for Taylor and Chebyshev polynomials alike. Additionally, $\sin(-\pi u_c)$ is computed instead of $-\sin(\pi u_c)$, thus, eliminating the need for a two's complement adder at the output. The architecture is formed of five pipeline stages, i.e. the first output is obtained after 5 clock cycles [59], synthesis and implementation results are given in Table 4.12. The suggested implementation is resource friendly and only requires 2.72% and 6.89% DSP blocks for the "ZynQ" and "Cyclone V" SoC, respectively. Moreover, the maximum clock frequency is 174 MHz, as such the presented architecture is a high speed and low complexity structure [59].

SoC	Polynomial Approximation Parallel Pipeline Architecture			
	DSP	Registers	Logic Units	Frequency (MHz)
ZynQ	2.72%	246	0.076%	174.917
Cyclone V	6.89%	138	0.143%	174.64

Table 4.12 – 5^{th} Order Polynomial SoC Implementation

The finite precision simulation is conducted for an input vector the same input vector $\mathbf{p}_t = [0, 0.125, 0.1667, 0.25, 0.5, 1]$ in a 18 bits signed format. In contrast to the Chebyshev coefficients with a $Q2.15$ format [1], the Taylor coefficients required the use of a $Q3.14$ [2], resulting in a further decrease in precision [59]. The Taylor and Chebyshev output is presented in Tables 4.13 and 4.14, respectively.

$f(p_t)$	Finite Precision Taylor Approximation					
	$p_t(0)$	$p_t(1)$	$p_t(2)$	$p_t(3)$	$p_t(4)$	$p_t(5)$
sin	0	-0.3826	-0.5000	-0.7048	-0.9447	-0.5240
cos	1	0.9238	0.8661	0.7073	0.0198	0.1218

Table 4.13 – 5^{th} Order Taylor Approximation In $Q3.14$ Finite Precision Format

In contrast to the Taylor approximation in Table 4.13, the Chebyshev approximation, presented in Table 4.14, guarantees a 3 decimal digit accuracy for all input values p.

1. One sign bit, two integer bits and fifteen decimal bits
2. One sign bit, three integer bits and fourteen decimal bits

$f(u_c)$	Finite Precision Chebyshev Approximation					
	$u_c(0)$	$u_c(1)$	$u_c(2)$	$u_c(3)$	$u_c(4)$	$u_c(5)$
sin	0	-0.3825	-0.5001	-0.7077	-0.9993	-0.0006
cos	1	0.9231	0.8610	0.7060	0	-1.0002

Table 4.14 – 5^{th} Order Chebyshev Approximation In $Q2.15$ Finite Precision Format

To further validate the Chebyshev approximation performance, the finite precision approximation MSE is computed with reference to the infinite precision result of Table 4.3 and is presented in Table 4.15.

f	Taylor and Chebyshev Finite Precision Approximation MSE	
	Taylor MSE	Chebyshev MSE
sin	0.0463	4.4817×10^{-6}
cos	0.2098	2.1×10^{-7}

Table 4.15 – Taylor and Chebyshev Finite Precision Approximation MSE

As shown in Table 4.15, the Taylor approximation MSE for both the sine and cosine functions is much greater than that of the Chebyshev, with respect to the direct computation results. Thus, similar to the computer simulations, the hardware simulations result, validates the superior accuracy of the Chebyshev approximation in finite precision for a smaller order polynomial. Where smaller order polynomials requires less DSP resources, i.e. complex multipliers and adders.

4.5.4 Delay Relaxed Look-Ahead LMS

Presenting a pipe-lined architecture for a high throughput parallel implementation is difficult. Such difficulty is caused by the dependency of the coefficient update loop on the feedback error and filter output [21, 22, 23] since we cannot predict what will be the next error in order to determine the weights. Additionally, computing the error in only one clock cycle severely degrades the maximum operating frequency. To overcome the previous problem a delay and sum relaxed look-ahead approximation technique is presented in

[21] for slowly varying signals, i.e. assuming a wide sense stationary (WSS) process, and tested for a tapped delay line filter. Such technique is achieved by applying a delay of D_2 samples for the input and coefficient terms and an additional delay relaxation of D_1 samples for the error path, with $D_1 \leq N$, where N is the number of antenna elements. However, for larger filters the resulting hardware overhead becomes unacceptable, thus a sum relaxation is employed. The applied sum relaxation involves averaging only D_3 terms where $1 \leq D_3 \leq D_2$ [21]. Hence, the delay relaxed look ahead LMS weight update equation and error term can be expressed as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k-D_2) + \mu \sum_{i=0}^{D_3-1} e^*_{DLMS}(k-D_1-i)\mathbf{x}(k-D_1-i) \quad (4.11)$$

$$e_{DLMS}(k) = d(k) - \mathbf{w}^{\mathbf{H}}(k-D_2)\mathbf{x}(k) \quad (4.12)$$

Thus, using (4.11) and (4.12) a pipeline parallel implementation is presented for the delay relaxed look ahead LMS, subject to a linear combiner, in Figure 4.10 using finite precision $Q2.15$ format.

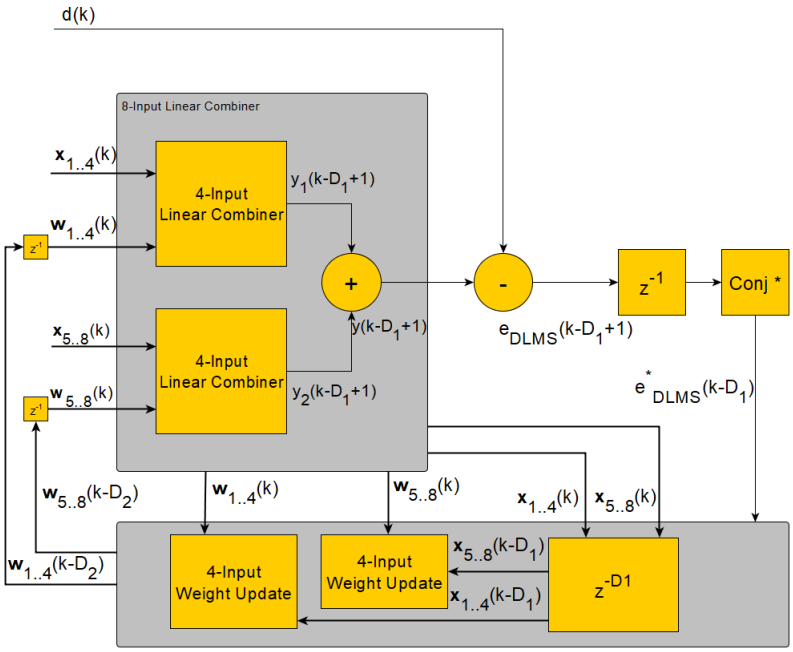

Figure 4.10 – Delay Relaxed Look Ahead LMS Hardware Architecture

Where $\mathbf{x}_{1..4}(k)$, $\mathbf{x}_{5..8}(k)$, $\mathbf{w}_{1..4}(k)$ and $\mathbf{w}_{5..8}(k)$ are the LMS input and weight vectors formed of first and last 4 elements, respectively. z^{-1}, z^{-D_1} represent the digital delay, i.e. registers of 1 and D_1 samples, respectively. The Conj block denotes complex conjugation. In addition, y_1 and y_2 form the intermediate outputs and they are defined as follows:

$$y_1(k) = \mathbf{w}^{\mathbf{H}}_{1..4}(k)\mathbf{x}_{1..4}(k) \tag{4.13}$$

$$y_2(k) = \mathbf{w}^{\mathbf{H}}_{5..8}(k)\mathbf{x}_{5..8}(k) \tag{4.14}$$

From Figure 4.10, a 8 input adaptive beamformer is formed by two 4 input linear combiner blocks and two 4 input weight update blocks. The system default external inputs are the input and desired signals $\mathbf{x}(k)$ and $d(k)$, respectively. The resulting stage outputs, y_1 and y_2, of each linear combiner are then combined to form the final output y and the total error e_{DLMS}. The resulting error and the external input signals $\mathbf{x}_{1..4}(k)$ and $\mathbf{x}_{5..8}(k)$ are then used to update the previous filter coefficients.

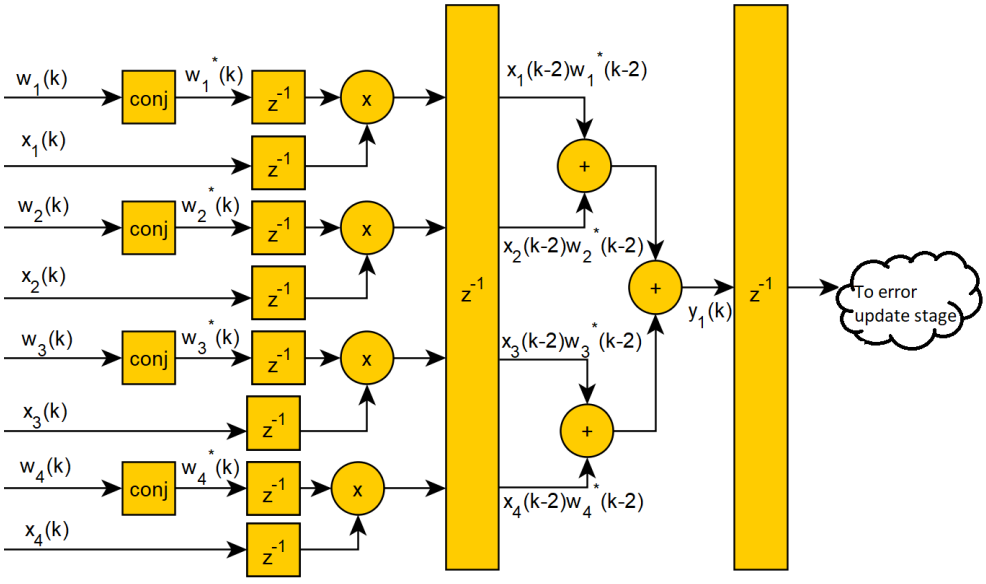

Figure 4.11 – Delay Relaxed Look Ahead LMS 4-Input Linear Combiner Architecture

Figure 4.11 shows that the multiplication and addition stages requires only one clock cycle, each, for all parallel inputs. Each complex multiplier is formed of four real multipliers

and one complex adder, i.e. it is equivalent two real adders. Moreover, from Figure 4.12, the update term is obtained by a right shift of 6 bits i.e. $\mu = 2^{-6} = 0.0156$ hence, omitting the need for an additional multiplier. All pipeline stages perform parallel operations and have a computational complexity of order $O(1)$.

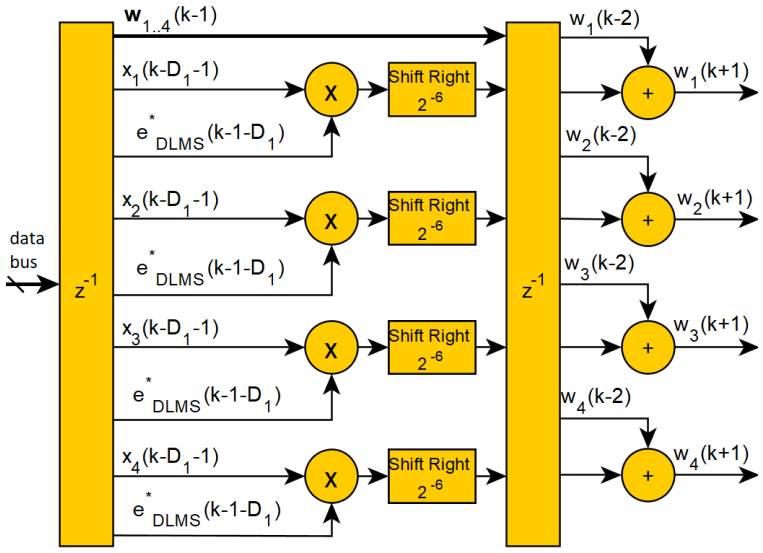

Figure 4.12 – Delay Relaxed Look Ahead LMS 4-Input Weight Update Architecture

Synthesis and implementation results are shown in Table 4.16 for the Intel "Stratix V 5SGXMABN3F45I4" model.

Design	DSP	Registers	Logic Units	Frequency(MHz)
DLMS	32	1746	773	208.33

Table 4.16 – 8-Input Delay Relaxed Look Ahead LMS Synthesis Results

From Table 4.16 the presented design achieves a maximum operating frequency of 208.33 MHz, while only using 32 DSP blocks, 1746 registers and 773 logic units. Hardware simulation is performed for the delay relaxed look ahead LMS in infinite precision (DLMS) and $Q2.15$ finite precision mode and is presented with the use of the beam radiation pattern. The simulation parameters are similar to that adopted, i.e. angle of arrival (AOA)

of 30°, first interference at 45° and second interference at 80°, the resulting beam radiation pattern is shown in Figure 4.13.

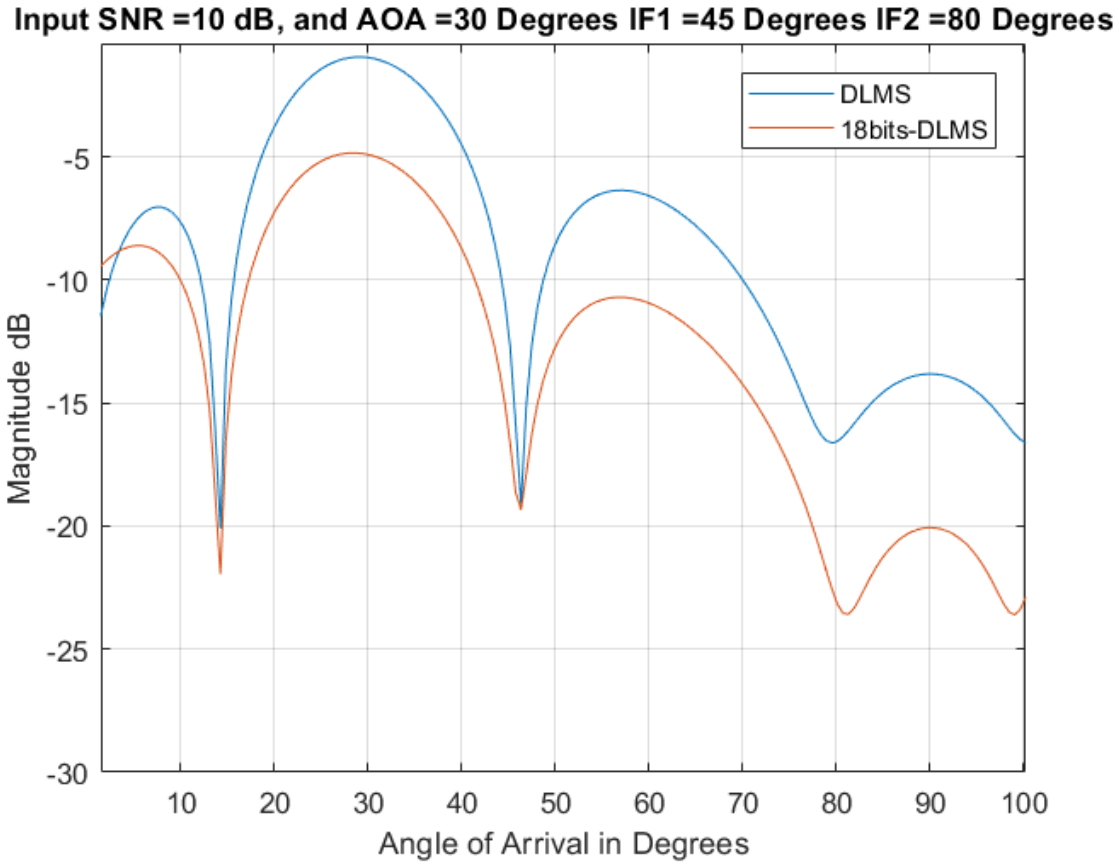

Figure 4.13 – Infinite and Finite Precision DLMS Beam Radiation Pattern

As shown in Figure 4.13, for a signal to noise ratio (SNR) of 10 dB both systems converged with near similar beam radiation pattern. As such, the delay relaxed look ahead LMS in $Q2.15$ precision shows equivalent infinite, theoretical, performance. While the suggested modification realizes a full pipeline architecture, but it doesn't eliminate the convergence speed and error floor trade off, it may deteriorate it in some cases [4, 21]. Additionally, the reduction in the averaging terms might introduce convergence issues [21].

4.6 Conclusion

In this chapter, we discussed some issues related to the hardware implementation of digital signal processing along with its advantages towards optimizing the performance and the cost of its host application. Moreover, we presented a comparative overview on several DSP optimized embedded processors, i.e. PDSP, FPGA and SoC, and different hardware design techniques, i.e. HDL and HLS. While general purpose processors presented degraded computational performance, when targeting application specific algorithms, it has been shown that this can be mitigated through the use of heterogeneous systems, i.e. SoC. However, the use of multi core SoC is a multidisciplinary problem and requires the use of HLL programming and HLS design tools. As such, we further proceeded by implementing an FFT radix-2 DIF processor using HDL, i.e. VHDL, and HLS, C on VivadoHLS, for different processor architecture. It has been shown that eventhought HLS design techniques accelerates development by automatically generating RTL equivalent code, it does require a proper use of compiler directives and coding standards. As most DSP routines implement a FFT processor, we proposed a high precision, low complexity, dynamic FFT twiddle factor generator using a 5^{th} order Chebyshev polynomial approximation and its pipeline hardware architecture. Through hardware and software simulations, we demonstrated the superior accuracy and performance of the proposed approximation scheme with respect to the traditional Taylor approximation. In contrast to the Taylor approximation, the proposed Chebyshev approximation method secured a precision up to three decimal digits for all input test elements in both finite and infinite precision modes. Finally, in this chapter we propose a pipeline implementation for the LMS adaptive beamformer based on the delay and sum relaxed look ahead technique. Hardware simulation is performed in finite, 18 bits, precision mode and demonstrated equivalent performance and accuracy compared to that of infinite precision. The work conducted has been published in the IEEE International Multidisciplinary Conference on Engineering Technology (IMCET) and in the Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES) conference.

Chapter 5

The Parallel LMS and it's Pipeline Hardware Implementation

5.1 Introduction

As previously stated, compared to the LMS, the RLS algorithm offers faster convergence and improved robustness against the input signals eigenvalue spread variation [6]. However, the RLS algorithm does not offer reliable tracking capabilities and requires extensive computations, i.e. computational complexity of order $O(N^2)$ [4], where N is the number of antenna elements. Several variants of the LMS algorithm have been put to light with the aim of eliminating the trade off between convergence speed and the achievable steady state error for a given adaptation step size [4, 14]. Some of the modified LMS algorithms include the normalized LMS (NLMS) [48], the constrained stability LMS (CSLMS) [49], the variable step size LMS (VSSLMS) [51, 52], the modified robust variable step size LMS (MRVSSLMS) [50] and the LLMS [5]. These algorithms use an additional computation stage to dynamically control the step size, i.e NLMS, CSLMS, VSSLMS and MRVSSLMS, or accurately tune the weights with respect to a multi stage, error averaging, structure, i.e. LLMS [5]. The LLMS adaptive algorithm is formed by two LMS stages connected by an estimate of the steering vector, where the overall total error is formed as a linear combination of each individual stages [4, 14]. Thus, for a uniform linear antenna (ULA) array of N elements, the LLMS doubles the computational requirements of a classical LMS and requires the use of a division operator for the cascading stage [24]. Therefore, the implementation of the LLMS requires a total of $6N+2$ complex multiplications, $5N+4$ complex additions and N complex divisions [4]. Given the cas-

cading nature of the LLMS, presenting a pipeline hardware implementation is extremely difficult [24]. Additionally, the introduction of a division operator exposes the system to underflow and to divide by zero errors in finite precision mode [4].

Thus, inspired by the multi stage error feedback technique [5, 6, 15], we propose a two-stage parallel, least mean square structure (pLMS) for adaptive beamforming and its pipeline hardware implementation. In contrast to LLMS, the pLMS is formed of two parallel LMS connected by error feedback, where the overall error signal is derived as a linear combination of individual stage errors, omitting the need for a cascading stage [4].

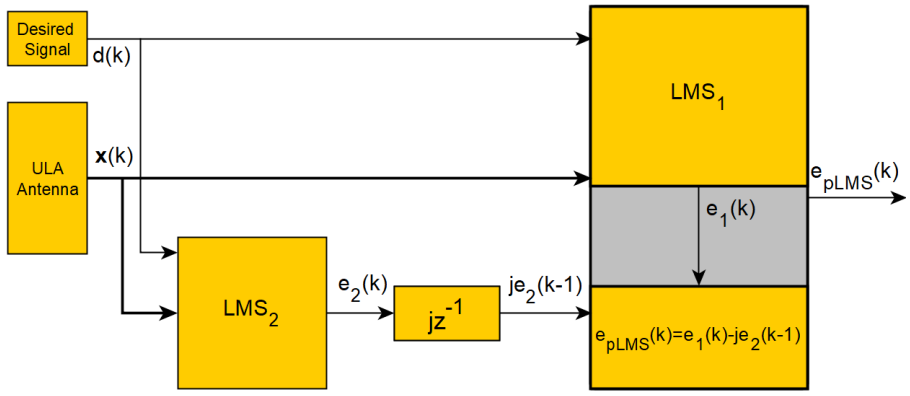

Figure 5.1 – pLMS Architecture

5.2 Multi Stage Parallel LMS (pLMS) Algorithm

By definition the pLMS beamformer is a multi stage LMS connected by an error feedback. As such, the error signal of the second LMS stage (LMS_2), $e_2(k)$, is delayed by one sample, multiplied by the imaginary number $j = \sqrt{-1}$, i.e. phase shift, and combined with the error of the first LMS stage (LMS_1), $e_1(k)$, to form the overall pLMS error e_{pLMS}. We should recall that the multiplication by j, as presented in section 3.3.5 and in (3.36), is introduced as an added robustness against recurring samples [24]. Recurring samples are formed as a consequence of consecutive data repetition in the original message signal, analog to digital converters (ADC) impurities [4, 17, 57], quantization errors, low resolution [4, 17, 34, 35, 36] and symbol detection errors originated from the digital

receivers in low SNR environments [4, 58]. As such, a multiplication by j is proposed for robustness against resulting error nulls [4, 24]. The pLMS structure is shown in Figure 5.1, where the block jz^{-1} represents one sample delay and a multiplication by j [4, 14]. As shown in Figure 5.1, the LMS_2 filter is no longer dependent on the steering vector estimate's output and operates in parallel with the LMS_1. Moreover, the total pLMS is formed as a linear combination of independent stage errors, such that:

$$\begin{aligned} e_{pLMS}(k) &= e_1(k) - je_2(k-1) \\ &= d(k) - jd(k-1) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}(k) + j\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1) \end{aligned} \tag{5.1}$$

where the superscript $^{\mathbf{H}}$ is the Hermitian transpose, $\mathbf{x}(k)$ is the input signal at the discrete time instance k, $e_i(k) = d(k) - \mathbf{w}_i^{\mathbf{H}}(k)\mathbf{x}(k)$, $i = 1, 2$ represents the stage identifier, $y_{LMS1}(k) = \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}(k)$ and $y_{LMS2}(k) = \mathbf{w}_2^{\mathbf{H}}(k)\mathbf{x}(k)$ are the first and second stage outputs, and $\mathbf{w}_1(k)$, $\mathbf{w}_2(k-1)$ are the current and delayed weight vectors of LMS_1 and LMS_2, respectively. Thus, from (5.1), the pLMS MSE cost function, $\xi_{pLMS}(k)$, can be computed as follows [4]:

$$\begin{aligned} \xi_{pLMS}(k) &= \mathrm{E}[|e_{pLMS}(k)|^2] \\ &= \mathrm{E}[e_1(k)e_1^*(k) + je_1(k)e_2^*(k-1) \\ &\quad -je_1^*(k)e_2(k-1) + e_2(k-1)e_2^*(k-1)] \end{aligned} \tag{5.2}$$

where E[.] is the expectation operation, |.| signifies the complex modulus and the superscript * denotes complex conjugation. Moreover, (5.2) can be expanded to obtain (5.3), further details can be found in Appendix B [4]:

$$\begin{aligned} \xi_{pLMS}(k) &= \mathrm{E}[|d(k)|^2] - \mathbf{p}^{\mathbf{H}}\mathbf{w}_1(k) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{p} + \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{R}\mathbf{w}_1(k) + \mathrm{E}[|d(k-1)|^2] \\ &\quad -\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) - \mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d^*(k)\mathbf{x}(k-1)] + j\mathrm{E}[d^*(k)d(k-1)] \\ &\quad +\mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[\mathbf{x}(k)\mathbf{x}(k-1)]\mathbf{w}_2(k-1) + j\mathrm{E}[d(k)d^*(k-1)] \\ &\quad -j\mathbf{w}_1^{\mathbf{H}}(k)\mathrm{E}[d^*(k-1)\mathbf{x}(k)] + j\mathbf{w}_1^{\mathbf{H}}(k)\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\ &\quad +j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_1(k) + j\mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d(k)\mathbf{x}(k-1)] \\ &\quad +j\mathrm{E}[\mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k)\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1)] - j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \end{aligned} \tag{5.3}$$

where $\mathbf{R} = \mathbf{R}(0)$ is the input signal auto-correlation matrix and $\mathbf{p} = \mathbf{p}(0)$ is the cross correlation vector of the input $\mathbf{x}(k)$ and desired signal $d(k)$. $\mathbf{R}(0)$ and $\mathbf{p}(0)$ are defined at lag $\tau = 0$ as:

$$\begin{aligned} \mathbf{R}(\tau) &= \mathrm{E}[\mathbf{x}(k-\tau)\mathbf{x}^{\mathbf{H}}(k)] && (5.4) \\ \mathbf{p}(\tau) &= \mathrm{E}[d^*(k-\tau)\mathbf{x}(k)] && (5.5) \end{aligned}$$

Where, assuming a wide sense stationary (WSS) process, the lag $\tau = k_1 - k_2$ and k_1 and k_2 are different time instances from which an observation of the random process is taken. With $\mathbf{w}_1(k)$ being the tap weights of interest [4, 5, 6, 13, 15]. The optimal weight vector, $\mathbf{w}_{op}$, of $\mathbf{w}_1(k)$ can be obtained by differentiating (5.3) with respect to $\mathbf{w^H}_1(k)$ [5, 15, 46], and setting the resulting pLMS gradient, $\nabla_{pLMS} = 0$, to obtain [4]:

$$\begin{aligned} \nabla_{pLMS} &= \frac{\partial \xi_{pLMS}(k)}{\partial \mathbf{w^H}_1(k)} \\ &= -\mathbf{p} + \mathbf{R}\mathbf{w}_1(k) - j\mathrm{E}[d^*(k-1)\mathbf{x}(k)] + j\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \quad (5.6) \end{aligned}$$

The optimal weight vector, $\mathbf{w}_{op}$, becomes [4]:

$$\mathbf{w}_{op} = \mathbf{R}^{-1}\mathbf{p} + j\mathbf{R}^{-1}\mathrm{E}[d^*(k-1)\mathbf{x}(k)] - j\mathbf{R}^{-1}\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \quad (5.7)$$

where $\mathbf{R}^{-1}$ is the input signals auto-correlation matrix inverse, assuming $\mathbf{R}$ is invertible [4]. The resulting gradient can be validated from (5.6) such as [4]:

$$\begin{aligned} \mathbf{w}_1(k+1) &= \mathbf{w}_1(k) - \mu_1 \nabla_{pLMS} \\ &= \mathbf{w}_1(k) - \mu_1[-\mathbf{p} + \mathbf{R}\mathbf{w}_1(k) - j\mathrm{E}[d^*(k-1)\mathbf{x}(k) \\ &\quad + j\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1)] \\ &= \mathbf{w}_1(k) + \mu_1\mathbf{x}(k)[d^*(k) - \mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k) + jd^*(k-1) \\ &\quad - j\mathbf{x}^{\mathbf{H}}(k-1)\mathbf{w}_2(k-1)] \quad (5.8) \end{aligned}$$

Thus, the pLMS adaptive algorithm is presented in (5.9) [4] as follow:

$$\begin{aligned}
&\mathbf{LMS}_1: \\
y_{LMS1}(k) &= \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}(k) \\
e_1(k) &= d(k) - y_{LMS1}(k) \\
e_{pLMS}(k) &= e_1(k) - je_2(k-1) \\
\mathbf{w}_1(k+1) &= \mathbf{w}_1(k) + \mu_1 e^*_{pLMS}(k)\mathbf{x}(k) \\
&\mathbf{LMS}_2: \\
y_{LMS2}(k) &= \mathbf{w}_2^{\mathbf{H}}(k)\mathbf{x}(k) \\
e_2(k) &= d(k) - y_{LMS2}(k) \\
\mathbf{w}_2(k+1) &= \mathbf{w}_2(k) + \mu_2 e_2^*(k)\mathbf{x}(k)
\end{aligned} \tag{5.9}$$

where μ_1 and μ_2 are the LMS_1 and LMS_2 step sizes, respectively.

5.2.1 Theoretical Stability Analysis

In order to determine under which conditions the pLMS is stable and converges to the optimal weight, a first order convergence and stability analysis is performed for LMS_1 and LMS_2.

5.2.2 First LMS Stage

Let the mean coefficient error vector, $\overline{\mathbf{v}}(k)$ [2] be defined as:

$$\overline{\mathbf{v}}(k) = \overline{\mathbf{w}}(k) - \mathbf{w}_{op} \tag{5.10}$$

where $\overline{\mathbf{w}}(k)$ is the mean weight vector. At steady-state and assuming both stages convergence, i.e. as $k \to \infty$, and $d(k-1) \approx y_{LMS1}(k-1) \approx y_{LMS2}(k-1)$, (5.7) becomes:

$$\begin{aligned}
\mathbf{w}_{op} &\approx \mathbf{w}_{oplms} + j\mathbf{R}^{-1}E[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) - j\mathbf{R}^{-1}E[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\
&\approx \mathbf{w}_{oplms}
\end{aligned} \tag{5.11}$$

Using the equality $\mathbf{w}_{oplms} = \mathbf{R}^{-1}\mathbf{p}$ in (5.11), we assume $\mathbf{w}_{op} \approx \mathbf{w}_{oplms}$. Thus, (5.11) becomes:

$$\mathbf{w}_{op} = \mathbf{A}^{-1}\left(\mathbf{p} + j\mathbf{p}(-1)\right) \tag{5.12}$$

Where, $\mathbf{A}$ is the final correlation matrix formed as a linear combination of the auto-correlation and cross-correlation matrices at lag 0 and 1, respectively; $\mathbf{A}$ is assumed invertible as a result of the random input noise [2] and is defined as:

$$\mathbf{A} = \mathbf{R} + j\mathbf{R}(1) \tag{5.13}$$

Thus, from (5.11), we can proceed as follows:

$$\begin{aligned}\overline{\mathbf{w}}(k+1) - \mathbf{w}_{op} &= \overline{\mathbf{w}}(k) - \mathbf{w}_{op} + \mu_1\mathbf{p} + j\mu_1\mathbf{R}(1)\mathbf{w}_{op} - \mu_1\mathbf{R}\overline{\mathbf{w}}(k) - \mu_1\mathbf{R}\mathbf{w}_{op} \\ &\quad +\mu_1\mathbf{R}\mathbf{w}_{op} + j\mu_1\mathbf{p}(-1) - j\mu_1\mathbf{R}(1)\mathbf{w}_{op} + \mu_1\mathbf{R}(1)\overline{\mathbf{w}}(k)\end{aligned} \tag{5.14}$$

Using the mean coefficient error vector notation and (6.17), we can write:

$$\begin{aligned}\overline{\mathbf{v}}(k+1) &= (\mathbf{I} - \mu_1\mathbf{R} - j\mu\mathbf{R}(1))\overline{\mathbf{v}}(k) + \mu_1\mathbf{p} - \mu_1(\mathbf{R} + j\mathbf{R}(1))\mathbf{A}^{-1}\mathbf{p} \\ &\quad +j\mu_1\mathbf{p}(-1) - \mu_1(j\mathbf{R} - \mathbf{R}(1))\mathbf{A}^{-1}\mathbf{p}(-1)\end{aligned} \tag{5.15}$$

The above equation can be simplified as follows:

$$\begin{aligned}\overline{\mathbf{v}}(k+1) &= (\mathbf{I} - \mu_1\mathbf{R} - j\mu_1\mathbf{R}(1))\overline{\mathbf{v}}(k) \\ &= (\mathbf{I} - \mu_1\mathbf{A})\overline{\mathbf{v}}(k)\end{aligned} \tag{5.16}$$

Using the eigenvalue decomposition (EVD), where Λ is a diagonal matrix with diagonal entries (λ_i) equal to the eigenvalues of $\mathbf{A}$, and $\mathbf{O}$ is a unitary matrix whose rows represent the eigenvectors of $\mathbf{A}$. We can now write $\mathbf{A} = \mathbf{O}^{-1}\Lambda\mathbf{O}$ [2]. We can rewrite (6.20) as:

$$\overline{\mathbf{v}}(k+1) = (\mathbf{I} - \mu_1\mathbf{O}^{-\mathbf{1}}\Lambda\mathbf{O})\overline{\mathbf{v}}(k) \tag{5.17}$$

Multiplying both sides of (6.21) by $\mathbf{O}$, we get:

$$\mathbf{O}\overline{\mathbf{v}}(k+1) \;=\; (\mathbf{I}-\mu_1\Lambda)\mathbf{O}\overline{\mathbf{v}}(k) \tag{5.18}$$

Let $\mathbf{m}(k) = \mathbf{O}\overline{\mathbf{v}}(k)$, where $\mathbf{m}(k)$ is $\overline{\mathbf{v}}(k)$ in a rotated coordinate, defined by the eigenvectors in $\mathbf{O}$, thus a convergence in $\mathbf{m}(k)$ means a convergence in $\mathbf{v}(k)$ [2], then:

$$\mathbf{m}(k+1) \;=\; (\mathbf{I}-\mu_1\Lambda)\mathbf{m}(k) \tag{5.19}$$

Since $(\mathbf{I}-\mu_1\Lambda)$ is a diagonal matrix, the stability and convergence is achieved with respect to the convergence of different first order difference equations formed by all N eigenvalues λ_i, $\forall i, i \in \{1, 2, .., N\}$ [2]. Thus, we define a set of N difference equations as follows:

$$\mathrm{m_i}(k+1) \;=\; (1-\mu_1\lambda_i)\mathrm{u_i}(k) \tag{5.20}$$

The convergence of the set of N difference equations is achieved if $|1-\mu_1\lambda_i| < 1$, [2]. Thus for the convergence in the mean sense, we require:

$$\mu_1 \;<\; \frac{1}{|\lambda_{A,max}|} \tag{5.21}$$

where the norm, $|\lambda_{A,max}|$, is the maximum eigenvalue in $\mathbf{A}$ and $|.|$ is the complex modulus, i.e. $\sqrt{Re\{\lambda\}^2 + Im\{\lambda\}^2}$. Thus, to ensure the convergence and the stability for LMS_1 the step size μ_1 must satisfy (6.25).

As the pLMS total error is formed with respect to LMS_2, the stability and convergence of LMS_2 is crucial. Since LMS_2 is the classical LMS algorithm with no additional modifications, the upper bound of the step size, μ_2, is given with respect to [2] as

$$\mu_2 \;<\; \frac{1}{\lambda_{R,max}} \tag{5.22}$$

where $\lambda_{R,max}$ is the maximum eigenvalue in $\mathbf{R}$. Thus, to ensure the convergence and the stability for LMS_2 the step size μ_2 must satisfy (5.22).

5.2.3 Transfer Function Approximation

A simple approximation to the behavior of the LMS adaptive algorithm, for temporal sampled signals, has been developed in [60]. The proposed approximation, models the system in terms of a discrete linear transfer function applicable for both deterministic and random inputs and is given by (5.23), such as:

$$H(z) = \frac{E[J(z)]}{E[D(z)]} = \frac{1}{1+\mu_{LMS}LR(z)} \tag{5.23}$$

μ_{LMS} is the LMS step size, L is the filter length, $J(z)$, $D(z)$ and $R(z)$, are the z transform polynomial whose coefficients are the instantaneous error signal $e(k)$, desired signal $d(k)$ and the input signals, $\mathbf{x}(k)$, auto-correlation estimates respectively [60, 84]. As such, In order to numerically assess the stability and performance of the proposed system, this section presents a discrete time transfer function approximation of the pLMS. Moreover, the input system described by the N equally spaced, identical antenna elements is modeled as a N^{th} order fractional delay filter employing a Farrow structure and a Lagrange interpolation [85]. Hence, the new pLMS input signal $\mathbf{x}_f(k)$ can now be defined as:

$$\mathbf{x}_f(k) = \mathbf{y}_{\mathrm{d}}(k) + \sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathbf{y}_{\mathrm{i,j}}(k) + \mathbf{n}(k) \tag{5.24}$$

where $\mathbf{y}_d(k)$, $\mathbf{y}_{\mathrm{i,j}}(k)$ are the message and interfering signals subject to a fractional delay filter and $\mathbf{n}(k)$ is a complex additive white Gaussian noise (CAWGN). The pLMS transfer function approximation can now be derived from (5.1) such that:

$$e_{pLMS}(k) = d(k) - jd(k-1) - y_1(k) + jy_2(k-1) \tag{5.25}$$

Assuming a WSS process and through the application of the Z transform for both sides of (5.25), the pLMS transfer function approximate $H_{pLMS}(z)$, as derived in Appendix D, becomes:

$$H_{pLMS}(z) = \frac{1+\mu_2 NR(z) - jz^{-1}}{(1+\mu_1 NR(z))(1+\mu_2 NR(z))} \tag{5.26}$$

Thus, for identical step sizes i.e. $\mu_{pLMS} = \mu_1 = \mu_2$ we obtain:

$$H_{pLMS}(z) = \frac{1 + \mu_{pLMS} NR(z) - jz^{-1}}{(1 + \mu_{pLMS} NR(z))^2} \tag{5.27}$$

The presented relationship in (5.26), starting at $k = 0$, includes both convergence and steady-state results [14, 60, 84]. While (5.26) and (5.27) present a simple approximation for the behavior of the pLMS adaptive beamformer; This approximation doesn't represent the optimal least-square solutions for the steady-state behavior [14].

5.3 pLMS Pipeline Hardware Implementation

While the pLMS offers an implicitly parallel structure most suitable for hardware implementation [24]. However, presenting a pipeline architecture is deemed difficult due to the presence of an error feedback loop in the LMS weight update algorithm [21, 22, 23, 24, 25]. As such, we propose the application of the delay and sum relaxed look ahead technique to present a parallel, pipeline pLMS hardware architecture (DpLMS) [21, 24].

5.3.1 Delay and Sum Relaxed Look Ahead pLMS

The delay and sum relaxation technique is applied for each of the LMS stages i.e., LMS_1 and LMS_2 separately [4, 21, 24]. As such, with respect to the weight update equation from (5.9), we start with a D_2 averaging look ahead step [21, 24] in the weight update path and a delay relaxation of D_1 samples in the error path. Hence, we obtain:

$$\mathbf{w}(k+1) = \mathbf{w}(k - D_2) + \mu \sum_{i=0}^{D_2 - 1} e^*(k - D_1 - i)\mathbf{x}(k - D_1 - i) \tag{5.28}$$

The modification presented in (5.28) is possible assuming the process is WSS and the gradient estimate undergoes only marginal changes over D_1 samples [21, 24]. In contrast to its set advantages in presenting an easy to pipeline form, the hardware overhead imposed by (5.28) is of $N(D_2 - 1)$ adders and becomes unacceptable for larger values of N and D_2 [21, 24]. Thus, to minimize the resulting overhead complications, an additional sum

relaxation of D_3 terms is applied, such that: $1 \leq D_3 \leq D_2$. Therefore, (5.28) is modified to become [24]:

$$\mathbf{w}(k+1) \quad = \quad \mathbf{w}(k-D_2) + \mu \sum_{i=0}^{D_3-1} e^*(k-D_1-i)\mathbf{x}(k-D_1-i) \tag{5.29}$$

Moreover, assuming μ is small enough [2], and $\mathbf{w}(k-D_2-1)$ can be approximated by $\mathbf{w}(k-D_2)$; In this case, the error update equation, with respect to D_2, becomes [24]:

$$e(k) \quad = \quad d(k) - \mathbf{w}^{\mathbf{H}}(k-D_2)\mathbf{x}(k) \tag{5.30}$$

Hence, a delayed pLMS (DpLMS) structure is now obtained by applying (5.29) and (6.35) to LMS_1 and LMS_2 [24].

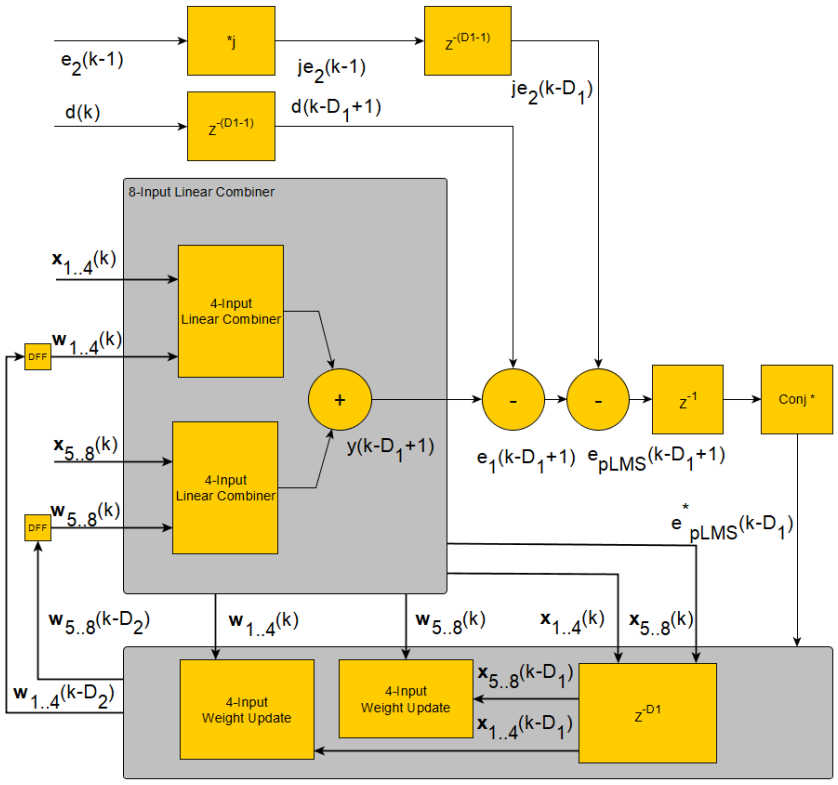

Figure 5.2 – 8-Elements DpLMS Hardware Architecture [24]

5.3.2 DpLMS Hardware Architecture

The DpLMS is implemented in finite precision mode with respect to the pLMS top level architecture presented in Figure 5.1 for an input source of $N = 8$ ULA arrays structure [24]. The adopted finite precision numbering format is the $Q2.15$ [1] format. The delay relaxation parameters are initialized, such as: $D_1 = 4$, $D_2 = 2$ and $D_3 = 1$ resulting in a total of six pipeline stages [24]. The proposed DpLMS top level architecture is shown in Figure 5.2, where $\mathbf{x}_{1..4}(k)$, $\mathbf{x}_{5..8}(k)$, $\mathbf{w}_{1..4}(k)$ and $\mathbf{w}_{5..8}(k)$ are the DpLMS input and weight vectors formed of the first and last 4 elements of $\mathbf{x}(k)$ and $\mathbf{w}(k)$, respectively [24]. The delay z^{-1}, z^{-D_1-1} the $*j$ and the "Conj" blocks denotes a digital delay, i.e. registers of one and $D_1 - 1$ samples, multiplication by j and complex conjugation, respectively.

The linear combiner and weight update blocks are defined in Figures 5.3 and 5.4, respectively.

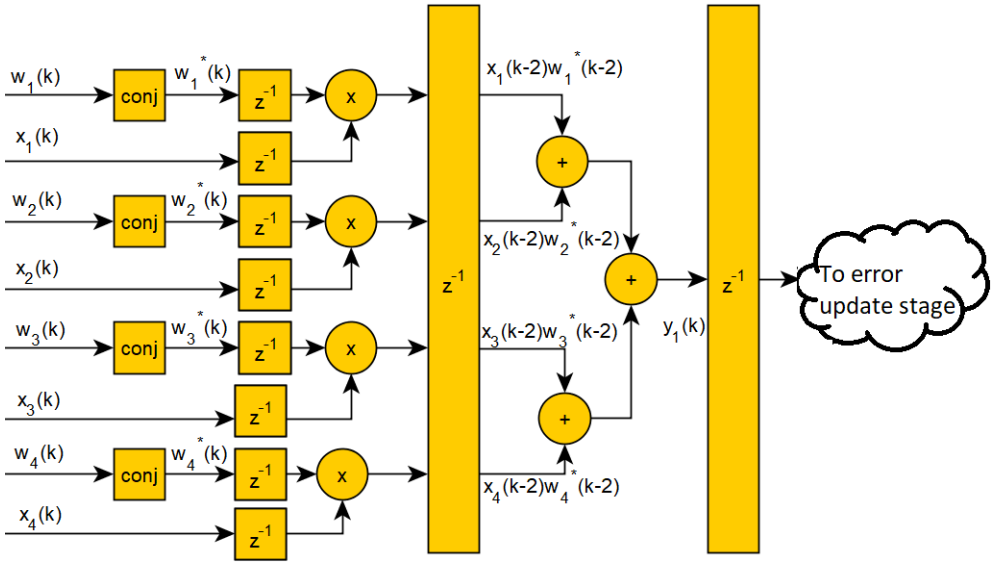

Figure 5.3 – 4-Input Linear Combiner Block

From Figure 5.3, it is clear that all multiplication and addition operation are performed in separate pipeline stages and requires one clock cycle each [24]. Thus, with an all parallel input architecture, the computational complexity of each stage is of order $O(1)$ [24]. The

1. 1 signed bit, 2 integer bits and 15 precision bits

inferred multiplier is a complex multiplier and is formed of four real multipliers and one complex adders, i.e. two real adders.

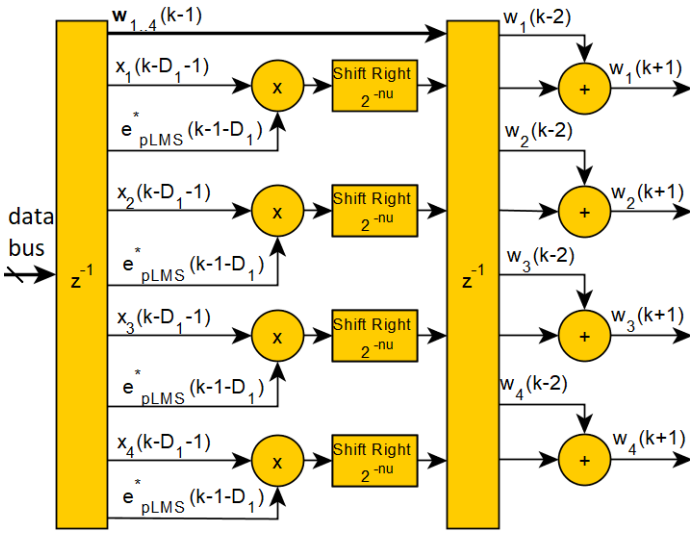

Figure 5.4 – 4-Input Weight Update Block

Moreover, the update term as shown in Figure 5.4, is obtained by an arithmetic shift right operation of nu bits, i.e. $\mu = 2^{-nu}$. The shift right operation is performed at wire speed hence omitting the requirement of an additional multiplier.

Design	DSP	Registers	Logic Elements	Frequency (MHz)
DLMS	32	1746	773	208.33
pLMS	64	3488	1546	208.33
DpLMS	64	3636	1567	208.33

Table 5.1 – DpLMS Beamformer Synthesis Results

5.3.3 Implementation and Synthesis Results

Synthesis and implementation results for the delayed LMS (DLMS), pLMS and DpLMS are obtained for the Intel "Stratix V 5SGXMABN3F45I4" model are presented in Table

5.1 [76]. In contrast to the pLMS, the DpLMS presents a pipeline high throughput structure at the cost of a negligible increase in the resource usage, i.e. logic elements and registers. Furthermore, in contrast to the original pLMS that can process one sample each $(D_1 + D_3) \times 4.8$ ns, i.e. input latency of 28.8 ns, the proposed DpLMS architecture can process one sample each 4.8 ns with an initial, setup, input latency of 6 cycles, i.e. 6 pipeline stages [24].

Algorithm	Initial Parameters
LMS	$\mu_{LMS} = 0.03125$
RLS	$\alpha = 0.98$, $\mathbf{L}(0) = 0.5\mathbf{I}$, $\mathbf{Q}(0) = 0.025\mathbf{I}$
LLMS	$\mu_1 = 0.03125$ $\mu_2 = 0.03125, \vartheta = 0.0004$
pLMS	$\mu_1 = 0.03125$, $\mu_2 = 0.03125$
DpLMS	$\mu_1 = 0.03125$, $\mu_2 = 0.03125$, $nu = 5$

Table 5.2 – Simulation Initial Parameters

5.4 Hardware and Software Simulations

A Monte Carlo type simulation is conducted to assess the behavior of the pLMS, the pLMS transfer function subject to fractional delay (pLMS-FD) and the DpLMS with respect to the LLMS, LMS and RLS adaptive algorithms. The simulation is performed for 500 realizations of 500 samples each where the input source is modeled as a ULA array with $N = 8$ elements. The input signal is formed by a combination of a message signal and two interferes impinging the array from broadside at an angle of 20°, 5° and 65°, respectively. The inputs are generated as independent random complex Gaussian sequences of the form $v_m = \mathcal{N}(0, \sigma_p^2) + j\mathcal{N}(0, \sigma_q^2)$ where, $\mathcal{N}$ denotes normal, zero mean, (Gaussian) distribution. σ_p^2 and σ_q^2 are the real and complex sequence variances, respectively. The final input signal is corrupted by CAWGN for a signal to noise ratio (SNR) of 10 dB. The parameters and initial conditions at $k = 0$ are initialized for the LMS and its variants with respect to Table 5.2, where $\mathbf{I}$ is the identity matrix.

5.4.1 Mean Square Error Convergence Analysis

The MSE simulation is used to study the convergence behavior of the LLMS, the pLMS, the DpLMS, the RLS and the classical LMS, and is presented in Figure 5.5. As shown in that Figure, the pLMS and the DpLMS convergence profile reflects that of the LLMS however with a lower steady state error, i.e. a better accuracy. The increase in accuracy is a result of eliminating the requirements for a steering vector estimation stage and the division operation. Additionally, it is clear that the DpLMS experienced a small delay in convergence with respect to the pLMs. The resulting delay is a consequence of the adopted delay and averaging scheme. Moreover, while the RLS achieved the best convergence profile, it requires an undesirable computational complexity of order $O(N^2)$ and lack tracking ability. In contrast, the pLMS and DpLMS achieved identical accelerated and accurate behavior while maintaining a linear computational complexity of order $O(N)$ [14, 24].

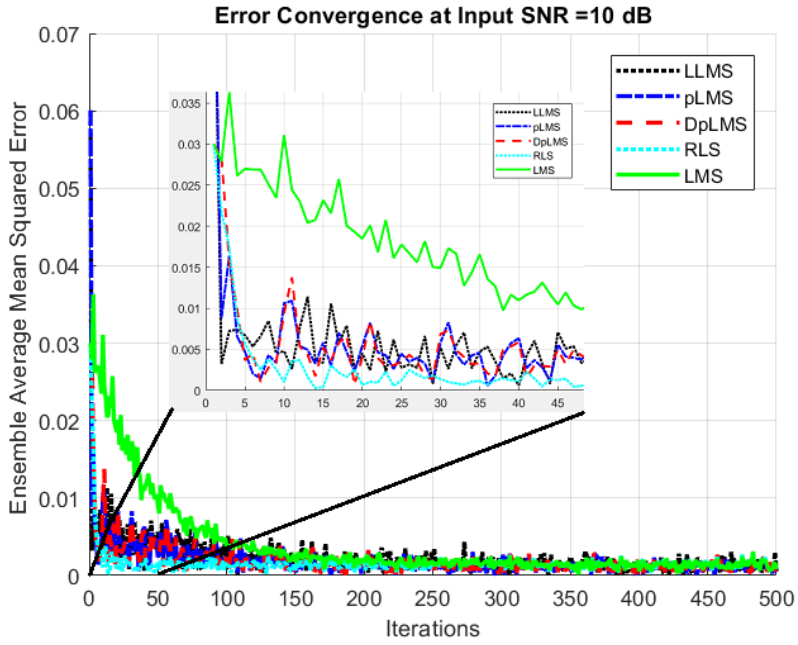

Figure 5.5 – pLMS MSE Convergence Behavior

As shown in Figure 5.6, the pLMS and pLMS-FD converged at the same iteration,

however the latter experienced a higher residual steady state error. The pLMS-FD steady state error profile is a consequence of the fractional delay filter approximation adopted previously [14, 24]. In order to validate the accuracy of the transfer function approximation with respect to the fractional delay filter, the pLMS MSE convergence behavior is evaluated against that of the pLMS-FD.

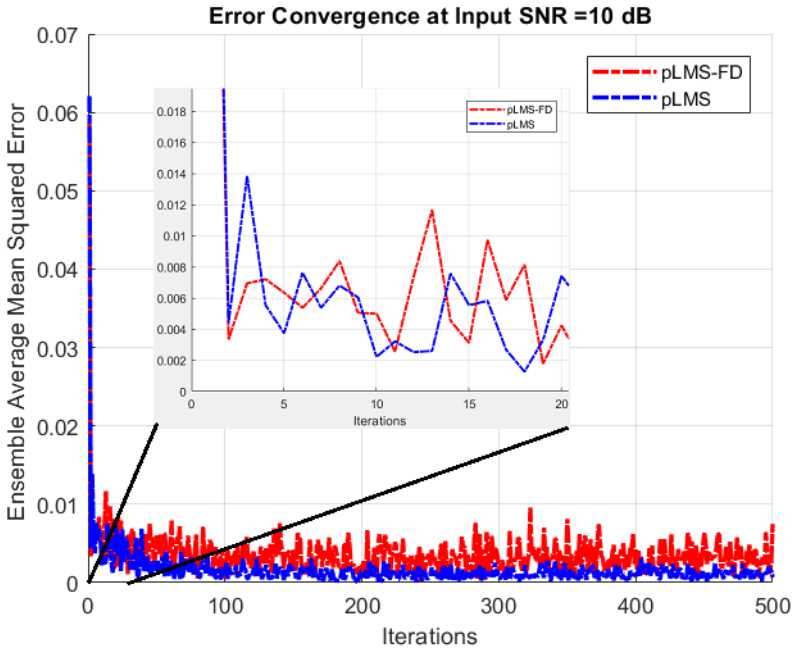

Figure 5.6 – pLMS and pLMS-FD MSE Convergence Behavior For Fractional Delay Filter

The pLMS was further compared against the LLMS for a sequence with repeated samples, the MSE convergence plot is shown in Figure 5.7. For a sequence with repeated samples, the LLMS failed to preserve its accelerated convergence profile. In contrast, the pLMS provided visible robustness in maintaining its accelerated convergence and low residual error characteristics and a result of the introduced error phase shift, i.e. multiplication by j [4, 14, 24].

Finally, the pLMS and DpLMS convergence behavior is analyzed for different SNR environments, i.e. from -5 dB to 7 dB with a step of 6 dB, the MSE output is presented in Figures 5.8 and 5.9.

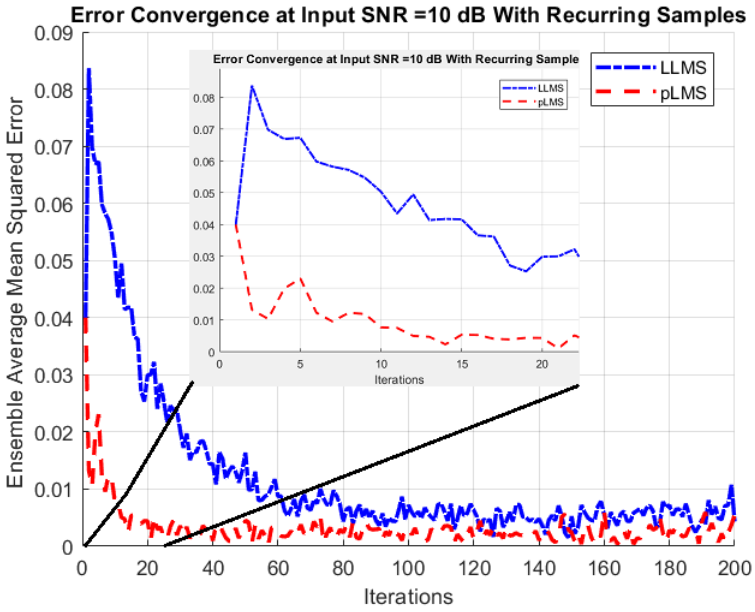

Figure 5.7 – LLMS and pLMS MSE Convergence Behavior for Recurring Samples

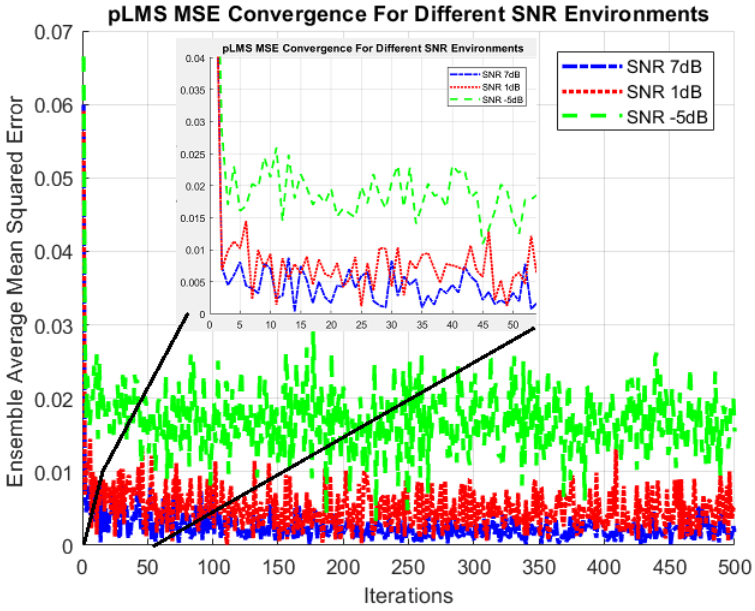

Figure 5.8 – pLMS MSE Convergence Behavior for Different SNR Environments

From Figure 5.8, the pLMS secured a stable convergence for an SNR as low as -5 dB for a small step size $\mu = 0.01$ and an average MSE of 0.02.

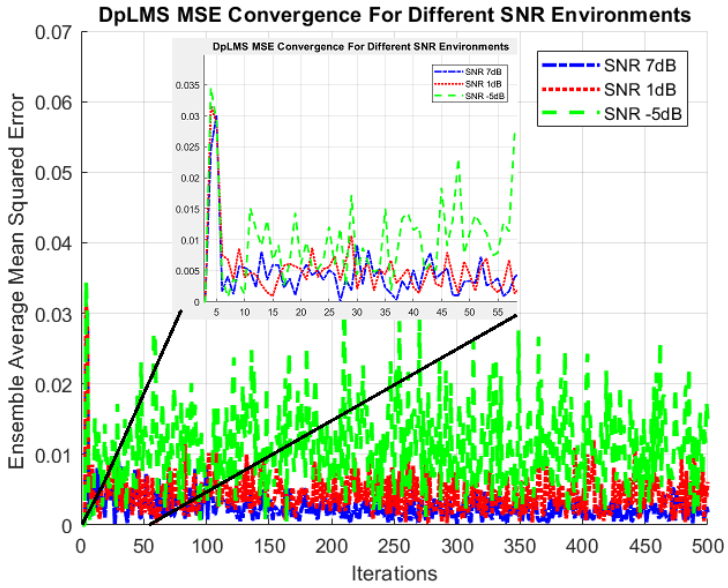

Figure 5.9 – DpLMS MSE Convergence Behavior for Different SNR Environments

In contrast, as shown in Figure 5.9, the DpLMS achieved an acceptable MSE convergence profile for only up to 1 dB. The degraded DpLMS behavior is a consequence of the adopting delay and averaging technique adopted in (5.29), i.e. the use of D_3 such that $1 \leq D_3 \leq D_2$ [14, 24]. Additional simulations cases, i.e. MSE beam localization, are shown in Appendix C.

5.4.2 Beam Radiation Pattern

Similar to the MSE convergence plot the beam radiation pattern is also evaluated in order to better assess the resulting beam pointing accuracy and the effect of the sum relaxation. The beam radiation pattern for the pLMS, the infinite precision DpLMS and the finite precision DpLMS is shown in Figure 5.10. The 18 bits finite precision DpLMS, simulated on the "Intel Stratix V FPGA", presented similar beam pattern and pointing accuracy compared to the infinite precision pLMS and DpLMS for an SNR of 10 dB [14].

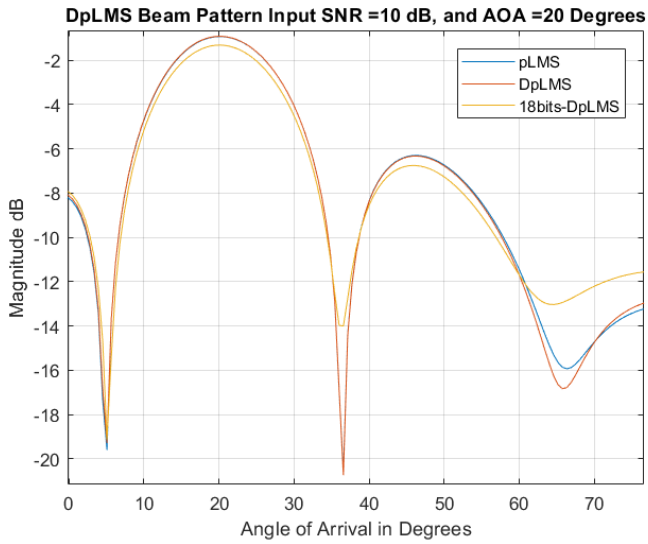

Figure 5.10 – DpLMS Finite and Infinite Precision Beam Radiation Pattern

Moreover, The beam pattern is further simulated for the pLMS and the finite precision DpLMS for an SNR environment ranging from -5 dB to 7 dB with a step of 6 dB and is shown in Figures 5.11 and 5.12, respectively.

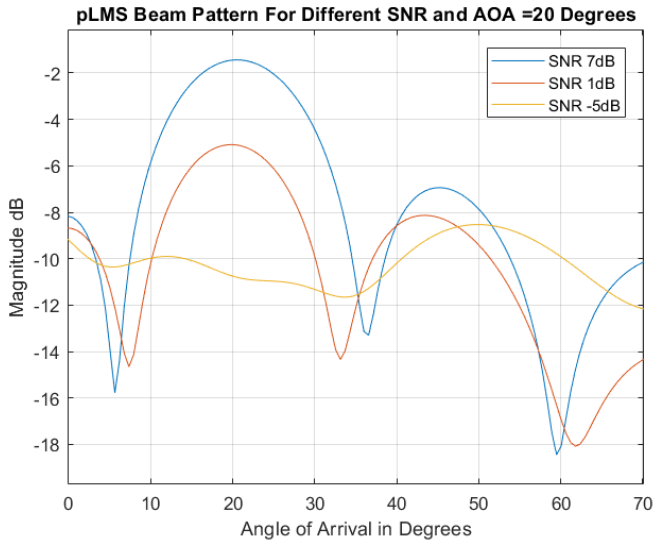

Figure 5.11 – pLMS Beam Pattern for Different SNR

The pLMS beam pattern shown in Figure 5.11 shows that the pLMS performance severely degraded in a -5 dB SNR environment. While the pLMS MSE successfully converged, the residual average error is increased.

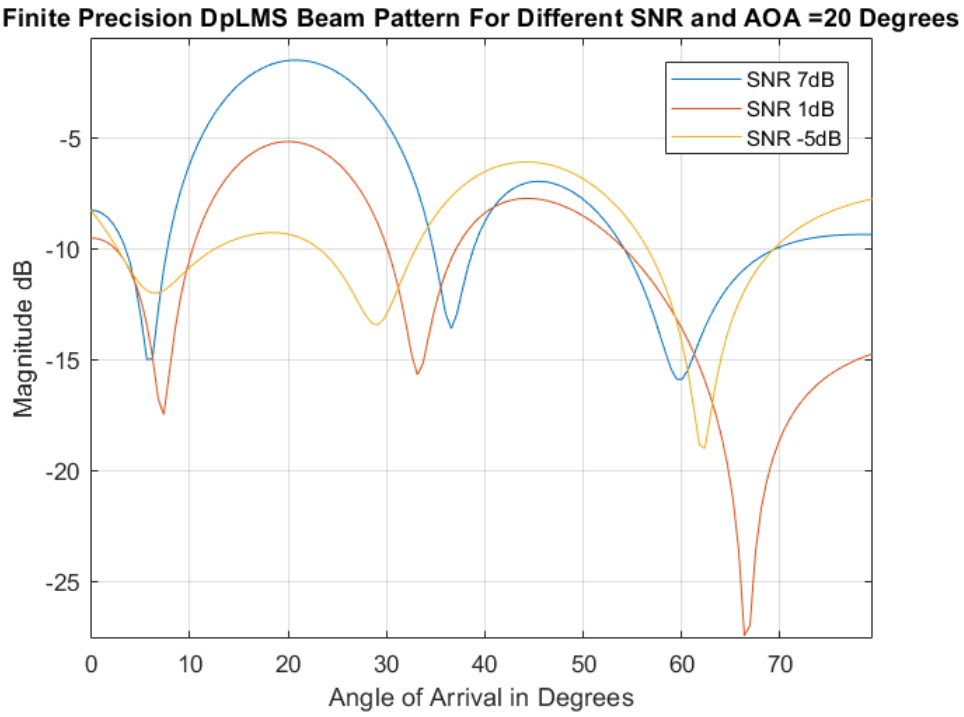

Figure 5.12 – Finite Precision DpLMS Beam Pattern for Different SNR

Similarly, from Figure 5.12 the DpLMS achieves a satisfactory behavior for SNR environments up to only 1 dB, however with a weaker main beam compared to that of the pLMS [14]. Such degradation validates the previously stated and can be omitted for large values of D_3, i.e. a high order moving average filter. However, as D_3 increases the hardware overhead increases [24].

5.4.3 Pole Zero Map Stability Plot

In order to numerically determine the maximum step size allowed for the pLMS a pole zero plot is performed for the transfer function approximation in (5.27). The pLMS

stability analysis is studied for different μ values and the pole zero plot is shown in Figure 5.13 [14]. As the step size μ_{pLMS} increases the resulting poles and zeros move further towards the outside of the unit circle. Thus, it can be concluded from the conducted study that the maximum step size to ensure convergence and stability falls in the range of $0.03 < \mu_{pLMS} < 0.04$ [14].

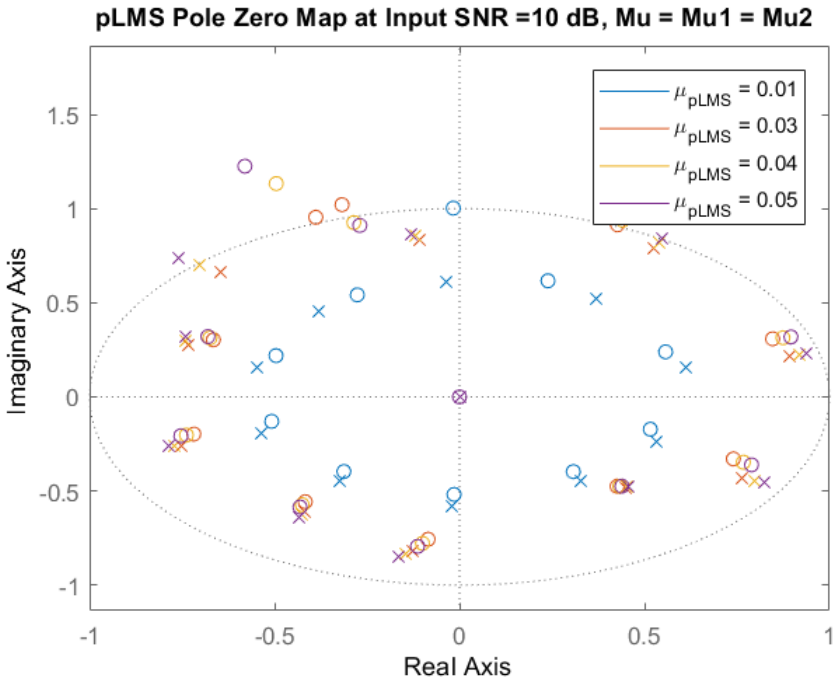

Figure 5.13 – pLMS Pole Zero Map for Different μ [14]

5.4.4 Computational Complexity Comparison

A resource complexity comparison for the pLMS against different adaptive algorithms recalled during this study is presented in Table 5.3 [14]. It is clear that the RLS adaptive algorithm and its variants require an undesirable complexity of order $O(N^2)$ [6, 14]. Furthermore, the RLMS, the LLMS and the RLMSp require $N + 1$, N and 1 complex divisions, respectively [14]. In contrast, the proposed pLMS structure presents superior convergence speed, low residual error and a parallel easy to pipeline architecture while

maintaining a computational complexity of order $O(N)$. Additionally, the pLMS omits the need of a division operator and provides a considerable reduction in resource usage, i.e. $2N$ complex multiplications, N complex additions and N complex divisions [4, 14]. However, compared to the LMS, the pLMS achieves its superior performance at the cost of a considerably large increase in resource requirements, i.e. double. As such, it is of crucial importance to present a reduced complexity structure of the pLMS while maintaining its accelerated convergence profile [4].

Algorithm	cMultiply	cAdd	cDivide
RLMS[15]	$3N^2 + 11N + 2$	$2N^2 + 9N + 6$	$N + 1$
RLMSp[6]	$3N^2 + 7N + 1$	$2N^2 + 6N + 3$	1
RLS	$3N^2 + 5N$	$2N^2 + 4N + 2$	1
LLMS[5]	$6N + 2$	$5N + 4$	N
pLMS	$4N + 2$	$4N + 4$	0
LMS	$2N + 1$	$2N + 1$	0

Table 5.3 – Theoretical Complexity and Resource Usage [14]

5.5 Conclusion

In this chapter, we proposed a multi stage parallel LMS structure (pLMS) for adaptive beamforming, we evaluated its transfer function approximation and we introduced its pipeline hardware implementation. pLMS is formed of two LMS stages operating in parallel and connected by an error feedback. The pLMS transfer function approximation is obtained by modeling the input spatial linear combiner as a fractional delay temporal filter with a Lagrange interpolation. Moreover, we propose a pipeline hardware implementation through the application of the delay and sum relaxed look ahead technique (DpLMS). Stability analysis is performed to determine the theoretical upper bound of the step size required to ensure convergence. The maximum allowable step size is determined numerically with respect to the transfer function approximation and the pole zero plot study. Simulation results demonstrated the superior performance of the pLMS and DpLMS in providing accelerated convergence and maintaining a low steady state error

while preserving a complexity of order $O(N)$. Both algorithms presented a satisfactory convergence profile and beam pattern for SNR up to 1 dB. Through pole zero plot analysis the maximum step size, allowed to maintain a proper pLMS convergence, is in the range of $0.03 < \mu < 0.04$. Synthesis and implementation results show that, in contrast to other RLS and LMS variants, the DpLMS provides a parallel, pipeline, low complexity and resource friendly architecture suitable for low end processors. Finally, the fixed point DpLMS beam radiation pattern validated the DpLMS accuracy in finite precision mode by providing a similar infinite precision beam directivity. The work conducted for the pLMS derivation and its hardware implementation has been separately published in the European Signal Processing Conference (EUSIPCO).

Chapter 6

The Reduced Complexity Parallel LMS and its Pipelined Hardware Implementation

6.1 Introduction

While the parallel LMS (pLMS) presented a parallel, easy to pipeline structure with accelerated convergence and low residual error, it still requires to instantiate two least mean square (LMS) stages, thus doubling the complexity of that of the classical LMS [4, 14]. In this chapter, we propose a reduced complexity pLMS structure for adaptive beamforming (RC-pLMS), its transfer function approximation and its pipeline hardware implementation [4].

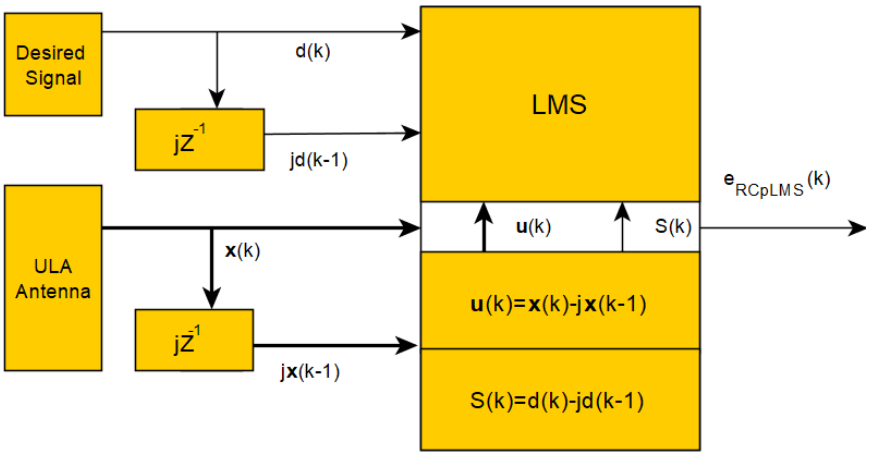

Figure 6.1 – Reduced Complexity pLMS (RC-pLMS)

RC-pLMS is formed from a single LMS stage with additional inputs, obtained by the applying a sample delay to the input and desired signal, as shown in Figure 6.1 [4]. Where, ULA is a uniform linear antenna, $e_{RC-pLMS}(k)$ is the RC-pLMS overall error, $\mathbf{u}(k)$ and $S(k)$ are the new system input and reference signals, respectively [4], the block jz^{-1} denotes multiplication by j with a one sample delay [4]. The new system inputs are defined, such as:

$$\begin{aligned} S(k) &= d(k) - jd(k-1) \\ \mathbf{u}(k) &= \mathbf{x}(k) - j\mathbf{x}(k-1) \end{aligned} \tag{6.1}$$

where $j = \sqrt{-1}$ is the imaginary number, $d(k)$ is the desired signal at the discrete time instance k and $\mathbf{x}(k)$ is the input signal. Moreover, The RC-pLMS transfer function approximation is obtained by modeling the input linear combiner filter by a temporal fractional delay finite impulse response (FIR) filter with a Lagrange interpolation [14]. Similarly, a parallel pipeline hardware implementation is presented through the application of the delay and sum relaxed look ahead technique (DRC-pLMS) [4]. Stability and quantization effect analyses are performed to determine the upper bound of the step size and assess the behavior of the system in a finite precision mode [4].

6.2 Reduced Complexity Multi Stage Parallel LMS (RC-pLMS) Algorithm

As previously stated, the pLMS is a multi stage LMS structure connected by an error feedback and its total error, $e_{pLMS}(k)$, is formed as a linear combination of individual stage errors, given as:

$$\begin{aligned} e_{pLMS}(k) &= e_1(k) - je_2(k-1) \\ &= d(k) - jd(k-1) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}(k) + j\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1) \end{aligned} \tag{6.2}$$

where the superscript $^{\mathbf{H}}$ is the Hermitian transpose, $e_i(k) = d(k) - \mathbf{w}_i^{\mathbf{H}}(k)\mathbf{x}(k)$, $i = 1,\ 2$ represents the stage identifier are the first and second stage outputs, and $\mathbf{w}_1(k)$, $\mathbf{w}_2(k-1)$

are the current and delayed weight vectors of the first and second LMS stages, LMS_1 and LMS_2, respectively. and its weight vector update equation is as follows:

$$\begin{aligned}\mathbf{w}_1(k+1) &= \mathbf{w}_1(k) + \mu_{pLMS}\mathbf{x}(k)[d^*(k) - \mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k) \\ &\quad + jd^*(k-1) - j\mathbf{x}^{\mathbf{H}}(k-1)\mathbf{w}_2(k)] \end{aligned} \tag{6.3}$$

where μ_{pLMS} is the pLMS step size, the superscript * represents complex conjugation. Moreover, the pLMS mean square error (MSE) cost function, ξ_{pLMS}, is thus defined from (6.2):

$$\begin{aligned}\xi_{pLMS}(k) &= \mathrm{E}[|d(k)|^2] - \mathbf{p}^{\mathbf{H}}\mathbf{w}_1(k) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{p} + \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{R}\mathbf{w}_1(k) + \mathrm{E}[|d(k-1)|^2] \\ &\quad -\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) - \mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d^*(k)\mathbf{x}(k-1)] \\ &\quad +\mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[\mathbf{x}(k)\mathbf{x}(k-1)]\mathbf{w}_2(k-1) + j\mathrm{E}[d(k)d^*(k-1)] \\ &\quad +j\mathrm{E}[d^*(k)d(k-1)] - j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\ &\quad -j\mathbf{w}_1^{\mathbf{H}}(k)\mathrm{E}[d^*(k-1)\mathbf{x}(k)] + j\mathbf{w}_1^{\mathbf{H}}(k)\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\ &\quad +j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_1(k) + j\mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d(k)\mathbf{x}(k-1)] \\ &\quad +j\mathrm{E}[\mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k)\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1)] \end{aligned} \tag{6.4}$$

where E[.] is the expectation operation, |.| signifies the complex modulus, $\mathbf{R} = \mathbf{R}(0)$ is the input signal auto-correlation matrix and $\mathbf{p} = \mathbf{p}(0)$ is the cross correlation vector of the input $\mathbf{x}(k)$ and desired signal $d(k)$. $\mathbf{R}(0)$ and $\mathbf{p}(0)$ are defined at lag $\tau = 0$ as:

$$\mathbf{R}(\tau) = \mathrm{E}[\mathbf{x}(k-\tau)\mathbf{x}^{\mathbf{H}}(k)] \tag{6.5}$$

$$\mathbf{p}(\tau) = \mathrm{E}[d^*(k-\tau)\mathbf{x}(k)] \tag{6.6}$$

Where, assuming a wide sense stationary (WSS) process, the lag $\tau = k_1 - k_2$ and k_1 and k_2 are different time instances from which an observation of the random process is taken. With respect to (6.4), the pLMS optimal weight vector, $\mathbf{w}_{op}$, is derived as follows:

$$\mathbf{w}_{op} = \mathbf{R}^{-1}\mathbf{p} + j\mathbf{R}^{-1}\mathrm{E}[d^*(k-1)\mathbf{x}(k)] - j\mathbf{R}^{-1}\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \tag{6.7}$$

where $\mathbf{R}^{-1}$ is the input signals auto-correlation matrix inverse, assuming $\mathbf{R}$ is invertible [4]. Assuming both system converges to their respective optimal weight, thus, as $k \to \infty$, we can approximate that $\mathbf{w}_1(k) \approx \mathbf{w}_1(k-1) \approx \mathbf{w}_{op}$ and that $\mathbf{w}_2(k) \approx \mathbf{w}_2(k-1) \approx \mathbf{w}_{oplms}$ [4, 5]. Therefore, the first and second LMS stages outputs y_{LMS1} and y_{LMS2}, having both interference and noise signals being suppressed, tend to approach the desired signal $d(k)$ [4, 5, 15]. As such, we assume that the filter output and the desired signal can satisfy the approximation $d(k-1) \approx y_{LMS1}(k-1) \approx y_{LMS2}(k-1)$ [4, 5, 15]. Thus, from (6.7) we can write:

$$\begin{aligned}
\mathbf{w}_{op} &= \mathbf{w}_{oplms} + j\mathbf{R}^{-1}E[d^*(k-1)\mathbf{x}(k)] - j\mathbf{R}^{-1}E[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\
&\approx \mathbf{w}_{oplms} + j\mathbf{R}^{-1}E[y^*_{LMS2}(k-1)\mathbf{x}(k)] - j\mathbf{R}^{-1}E[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\
&\approx \mathbf{w}_{oplms} + j\mathbf{R}^{-1}E[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) - j\mathbf{R}^{-1}E[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\
&\approx \mathbf{w}_{oplms} \qquad (6.8)
\end{aligned}$$

where $\mathbf{w}_{oplms} = \mathbf{R}^{-1}\mathbf{p}$ is the LMS optimal weight vector. As such, by simplifying (6.8), we reach the following approximation $\mathbf{w}_{op} \approx \mathbf{w}_{oplms}$ and hence we can assume that $\mathbf{w}_1(k) \approx \mathbf{w}_2(k-1)$, i.e. the use of a single set of weights, $\mathbf{w}(k)$ [4]. We now simplify the pLMS weight update equation given in (6.3) and remove any requirement for computing an independent set of filter coefficients, i.e. additional filter [4]. As such, (6.3) becomes:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{x}(k)[d^*(k) - \mathbf{x}^{\mathbf{H}}(k)\mathbf{w}(k) + jd^*(k-1) - j\mathbf{x}^{\mathbf{H}}(k-1)\mathbf{w}(k)] \quad (6.9)$$

where μ is the RC-pLMS step size, $\mathbf{w}(k)$ is the RC-pLMS filter weight vector. Thus, the RC-pLMS is presented as follows :

$$\begin{aligned}
S(k) &= d(k) - jd(k-1) \\
\mathbf{u}(k) &= \mathbf{x}(k) - j\mathbf{x}(k-1) \\
e_{RC-pLMS}(k) &= S(k) - \mathbf{w}^{\mathbf{H}}(k)\mathbf{u}(k) \\
\mathbf{w}(k+1) &= \mathbf{w}(k) + \mu\mathbf{x}(k)e^*_{RC-pLMS}(k) \qquad (6.10)
\end{aligned}$$

The RC-pLMS adaptive beamformer is formed of only one LMS filter with respect to two input adjustment blocks. Therefore, omitting the need for a second independent filter and reducing the original pLMS complexity by $2N+1$ complex multiplications and $N+2$ complex additions [4]. The RC-pLMS MSE cost function, defined as $\xi_{RCpLMS}(k) = E[|e_{RCpLMS}(k)|^2]$, is re-evaluated to obtain [4]:

$$\begin{aligned}\nabla_{RCpLMS} &= \frac{\partial \xi_{RCpLMS}(k)}{\partial \mathbf{w}^{\mathbf{H}}(k)} \\ &= -\mathbf{p} + \mathbf{R}\mathbf{w}(k) - \mathrm{E}[jd^*(k-1)\mathbf{x}(k)] + \mathrm{E}[j\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}(k) \\ &\quad +\mathrm{E}[jd^*(k)\mathbf{x}(k-1)] - \mathrm{E}[j\mathbf{x}(k-1)\mathbf{x}^{\mathbf{H}}(k)]\mathbf{w}(k) \\ &\quad -\mathrm{E}[d^*(k-1)\mathbf{x}(k-1)] + \mathrm{E}[\mathbf{x}(k-1)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}(k) \end{aligned} \tag{6.11}$$

where, ∇_{RCpLMS} is the RC-pLMS gradient. Assuming a WSS process, and using the equality $\mathbf{Q}_{\mathrm{r}}(\tau) = \mathbf{Q}_{\mathrm{r}}^*(-\tau)$ and $\mathbf{z}_{\mathrm{r}}(\tau) = \mathbf{z}_{\mathrm{r}}^*(-\tau)$, the RC-pLMS gradient defined in (6.11) can be re-written as [4]:

$$\begin{aligned}\nabla_{RCpLMS} &= -\mathbf{p} + \mathbf{R}\mathbf{w}(k) - \mathbf{z}_{\mathrm{r}}^*(-1) + \mathbf{Q}_{\mathrm{r}}(1)\mathbf{w}(k) \\ &\quad +\mathbf{z}_{\mathrm{r}}(1) - \mathbf{Q}_{\mathrm{r}}^*(-1)\mathbf{w}(k) - \mathbf{p} + \mathbf{R}\mathbf{w}(k) \\ &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(k) - 2\mathbf{z}_{\mathrm{r}}(-1) + 2\mathbf{Q}_{\mathrm{r}}\mathbf{w}(k) \end{aligned} \tag{6.12}$$

where the auto-correlation matrix and the cross-correlation vector are defined such as: $\mathbf{Q}_{\mathrm{r}}(1) = \mathrm{j}\mathbf{R}(1)$ and $\mathbf{z}_{\mathrm{r}}(-1) = \mathrm{j}\mathbf{p}(-1)$, respectively and represent an observation with a lag $\tau = 1$. Thus, by equating (6.12) to zero the RC-pLMS optimal weight, $\mathbf{w}_{opr}$, is obtained as [4, 46]:

$$\mathbf{w}_{opr} = \mathbf{A}^{-1}\left(\mathbf{p} + j\mathbf{p}(-1)\right) \tag{6.13}$$

where $\mathbf{A}$ is the RC-pLMS overall correlation matrix and it is formed as a linear combination of the auto-correlation matrices at lags 0 and 1 and is defined as:

$$\mathbf{A} = \mathbf{R} + j\mathbf{R}(1) \tag{6.14}$$

Assuming $\mathbf{A}$ is a full rank matrix; Thus its inverse $\mathbf{A}^{-1}$ exists as a consequence of the random added noise [2, 4].

6.2.1 Stability Analysis

In order to determine the upper bound of the step size and the conditions where the RC-pLMS converges to its optimal weight, we perform a first order stability analysis. The analysis studied with respect to the RC-pLMS learning rate, i.e. μ and the mean coefficient error vector, $\overline{\mathbf{v}}(k)$ [4, 10, 86] which is given by:

$$\overline{\mathbf{v}}(k) = \overline{\mathbf{w}}(k) - \mathbf{w}_{opr} \tag{6.15}$$

where $\overline{\mathbf{w}}(k)$ is the mean weight vector. From (6.9), we can proceed as follows [4]:

$$\overline{\mathbf{w}}(k+1) = \overline{\mathbf{w}}_1(k) + \mu\mathbf{p} - \mu\mathbf{R}\overline{\mathbf{w}}(k) + j\mu\mathbf{p}(-1) - j\mu\mathbf{R}(1)\overline{\mathbf{w}}(k) \tag{6.16}$$

At steady state and assuming convergence, we can approximate $\overline{\mathbf{w}}(k+1) \approx \overline{\mathbf{w}}(k)$ and $\overline{\mathbf{w}}(k-1) \approx \overline{\mathbf{w}}(k)$. As such, (6.16) becomes [4]:

$$\begin{aligned}\overline{\mathbf{w}}(k+1) - \mathbf{w}_{opr} &= \overline{\mathbf{w}}(k) - \mathbf{w}_{opr} + \mu\mathbf{p} + j\mu\mathbf{R}(1)\mathbf{w}_{opr} - \mu\mathbf{R}\overline{\mathbf{w}}(k) - \mu\mathbf{R}\mathbf{w}_{opr} \\ &\quad +\mu\mathbf{R}\mathbf{w}_{opr} + j\mu\mathbf{p}(-1) - j\mu\mathbf{R}(1)\mathbf{w}_{opr} + \mu\mathbf{R}(1)\overline{\mathbf{w}}(k)\end{aligned} \tag{6.17}$$

where $\mathbf{w}_{opr}$ is the RC-pLMS optimal weight vector. Using the mean coefficient error vector notation and equations (6.13), (6.17), we can write [4]:

$$\begin{aligned}\overline{\mathbf{v}}(k+1) &= (\mathbf{I} - \mu\mathbf{R} - j\mu\mathbf{R}(1))\overline{\mathbf{v}}(k) - \mu\mathbf{R}\mathbf{A}^{-1}\mathbf{p} - j\mu\mathbf{R}\mathbf{A}^{-1}\mathbf{p}(-1) \\ &\quad +\mu\mathbf{p} + j\mu\mathbf{p}(-1) - \mu\mathbf{R}(1)\mathbf{A}^{-1}\mathbf{p} + \mu\mathbf{R}(1)\mathbf{A}^{-1}\mathbf{p}(-1)\end{aligned} \tag{6.18}$$

Hence, (6.18) can be simplified to obtain [4]:

$$\begin{aligned}\overline{\mathbf{v}}(k+1) &= (\mathbf{I} - \mu\mathbf{R} - j\mu\mathbf{R}(1))\overline{\mathbf{v}}(k) + \mu\mathbf{p} - \mu(\mathbf{R} + j\mathbf{R}(1))\mathbf{A}^{-1}\mathbf{p} \\ &\quad +j\mu\mathbf{p}(-1) - \mu(j\mathbf{R} - \mathbf{R}(1))\mathbf{A}^{-1}\mathbf{p}(-1)\end{aligned} \tag{6.19}$$

(6.19) can be simplified as follows [4]:

$$\overline{\mathbf{v}}(k+1) \;=\; (\mathbf{I}-\mu\mathbf{R}-j\mu\mathbf{R}(1))\overline{\mathbf{v}}(k) = (\mathbf{I}-\mu\mathbf{A})\overline{\mathbf{v}}(k) \tag{6.20}$$

Applying the eigenvalue decomposition (EVD), with Λ being a diagonal matrix whose diagonal entries (λ_i) represent to the eigenvalues of $\mathbf{A}$, and $\mathbf{O}$ is a unitary matrix whose rows represent the eigenvectors of $\mathbf{A}$, it is possible to write $\mathbf{A} = \mathbf{O}^{-1}\Lambda\mathbf{O}$ [2, 4]. Therefore, we can rewrite (6.20) as follows:

$$\overline{\mathbf{v}}(k+1) \;=\; (\mathbf{I}-\mu\mathbf{O}^{-1}\Lambda\mathbf{O})\overline{\mathbf{v}}(k) \tag{6.21}$$

by multiplying both sides of (6.21) by the unitary matrix $\mathbf{O}$, we get [4]:

$$\mathbf{O}\overline{\mathbf{v}}(k+1) \;=\; (\mathbf{I}-\mu\Lambda)\mathbf{O}\overline{\mathbf{v}}(k) \tag{6.22}$$

Let $\mathbf{m}(k) = \mathbf{O}\overline{\mathbf{v}}(k)$, which represents $\overline{\mathbf{v}}(k)$ in a rotated coordinate system and it is defined by the eigenvectors in $\mathbf{O}$. Therefore, a convergence in $\mathbf{m}(k)$ implies a convergence in $\overline{\mathbf{v}}(k)$ [2, 4, 10], and we obtain:

$$\mathbf{m}(k+1) \;=\; (\mathbf{I}-\mu\Lambda)\mathbf{m}(k) \tag{6.23}$$

Since all elements in $(\mathbf{I}-\mu\Lambda)$ are diagonal, the stability is achieved with respect to the convergence of the first order difference equation formed by all N eigenvalues λ_i, $\forall i, i \in \{1, 2, .., N\}$ [2, 4]. Thus, with respect to (6.23), we can define a set of N difference equations as follows:

$$\mathrm{m_i}(k+1) \;=\; (1-\mu\lambda_i)\mathrm{m_i}(k) \tag{6.24}$$

The convergence of (6.24) is obtained by satisfying the condition $|1-\mu\lambda_i| < 1$ [2, 4, 86]. Moreover, for the convergence in the mean sense the step size should satisfy the following condition:

$$\mu \;<\; \frac{1}{|\lambda_{A,max}|} \tag{6.25}$$

where the norm, $|\lambda_{A,max}|$, is the maximum eigenvalue in $\mathbf{A}$ and $|.|$ is the complex modulus i.e. $\sqrt{Re\{\lambda\}^2 + Im\{\lambda\}^2}$ [4].

6.2.2 Transfer Function Approximation

Similar to the pLMS, the RC-pLMS transfer function approximation is obtained by modeling the input source as a temporal fractional delay FIR filter with a Lagrange interpolation [4]. As such, with respect to Appendix D, the pLMS transfer function is modified to reflect (6.10) and thus the RC-pLMS transfer function becomes as follows:

$$H_{RC-pLMS}(z) = \frac{1 - jz^{-1}}{1 + \mu N(1 - jz^{-1})R(z)} \tag{6.26}$$

The RC-pLMS transfer function approximation, given in (6.26), modifies that of the LMS to include the additional delay and phase shift term $1 - jz^{-1}$.

6.2.3 Quantization Effect Analysis

Adaptive algorithms, when implemented in finite precision and fixed point format can suffer a sever degradation in performance and diverge from their theoretical values [2, 4, 16, 25, 87]. Such performance decay is a result of quantization and round off errors due to the limitations imposed by the finite precision wordlength. Therefore, such errors, if not accounted in the design process, can cause catastrophic outcomes [2, 4, 16, 25, 87]. Thus, the analysis of the effects of quantization and round off errors becomes mandatory when targeting digital systems with a finite precision [2, 4, 16, 18].

For simplicity, we assume that the input and reference signals are Gaussian, zero-mean, uncorrelated and generated from an independent identically distributed (i.i.d) sequences. Additionally, the RC-pLMS analysis parameters $\mathbf{u}_q(k)$, $\mathbf{w}_q(k)$, $S_q(k)$, $e_q(k)$ and $y_q(k)$,

where the subscript q denotes the quantization process, are defined such as [4]:

$$\begin{aligned} \mathbf{u}_q(k) &= \mathbf{u}(k) + \eta_u(k) \\ \mathbf{w}_q(k) &= \mathbf{w}(k) + \eta_w(k) \\ S_q(k) &= S(k) + \eta_S(k) \\ y_q(k) &= y(k) + \eta_y(k) \\ e_q(k) &= S_q(k) - y_q(k) \end{aligned} \tag{6.27}$$

where $\eta_u(k)$, $\eta_w(k)$, $\eta_S(k)$ and $\eta_y(k)$ are the input signal, weight vector, reference signal and output quantization error respectively [4]. The quantization errors, $\eta_u(k)$ and $\eta_w(k)$, are assumed to be zero mean white sequences with variance σ_d^2, mutually independent and are also independent of $\mathbf{u}(k)$ and $\mathbf{w}(k)$, respectively. Hence, from (6.27) and the error update term in (6.10) we can write [4]:

$$\begin{aligned} e_q(k) &= S_q(k) - y_q(k) \\ &= S(k) + \eta_S(k) - \mathbf{w^H}_q(k)\mathbf{u}_q(k) - \eta_y(k) \\ &= S(k) + \eta_S(k) - \mathbf{w^H}(k)\mathbf{u}(k) - \mathbf{w^H}(k)\eta_u(k) \\ &\quad -\eta_w(k)\mathbf{u}(k) - \eta_w^*(k)\eta_u(k) - \eta_y(k) \end{aligned} \tag{6.28}$$

By eliminating higher order quantization error terms, i.e. $\eta_w^*(k)\eta_u(k) = 0$, (6.28) simplifies to [4]:

$$e_q(k) = e_{RCpLMS}(k) + \eta_{Sy}(k) - \mathbf{w^H}(k)\eta_u(k) - \eta_w(k)\mathbf{u}(k) \tag{6.29}$$

with $\eta_{Sy}(k) = \eta_S(k) - \eta_y(k)$. The RC-pLMS MSE subject to a quantization error, RC-pLMSq, can now be defined as $\xi_{RCpLMSq}(k) = [|e_q(k)|^2]$ and evaluated to obtain [4]:

$$\begin{aligned} \xi_{RCpLMSq}(k) &= \mathrm{E}[|e_q(k)|^2] \\ &= \xi_{RCpLMS}(k) + \mathrm{E}[e_{RCpLMS}(k)\eta_{Sy}^*(k) + e_{RCpLMS}^*(k)\eta_{Sy}(k) \\ &\quad -e_{RCpLMS}(k)\mathbf{w}(k)\eta_u^*(k) - e_{RCpLMS}(k)\mathbf{u^H}(k)\eta_w^*(k) \\ &\quad -e_{RCpLMS}^*(k)\mathbf{w^H}(k)\eta_u(k) - e_{RCpLMS}^*(k)\mathbf{u}(k)\eta_u(k)] \end{aligned} \tag{6.30}$$

Assuming the step size μ is small enough, the quantization error terms $\eta_w(k)\mathbf{u}(k)$ and $\mathbf{w}(k)\eta_u(k)$ becomes uncorrelated with each other and with the RC-pLMS error $e_{RCpLMS}(k)$ [2, 4]. Therefore, from [2, 4], (6.30) is simplified to become:

$$\begin{aligned} \xi_{RCpLMSq}(k) &= \mathrm{E}[|e_q(k)|^2] \\ &= J_{min}(1+\rho) + \xi_1(\sigma_w^2, \mu) + \xi_2(\sigma_d^2) \end{aligned} \tag{6.31}$$

where $J_{min}(1+\rho)$ is the MSE of the infinite precision algorithm, ρ is the misadjustment, $\xi_1(\sigma_w^2, \mu)$ is the quantization error resulting from $\eta_w(k)$ and $\xi_2(\sigma_d^2)$ is the quantization error resulting from $\eta_u(k)$ and $\eta_y(k)$ [2, 4]. With respect to [4, 18], (6.31) can now be written, at convergence, as follows:

$$\begin{aligned} \xi_{RCpLMSq}(k) &= \mathrm{E}[|e_q(k)|^2] \\ &= \xi_{min} + \frac{1}{2}\mu\xi_{min}tr(A) + \frac{N\sigma_a^2}{2\eta_u(k)\mu} + \frac{1}{\eta_u^2(k)}(|\mathbf{w}_{op}|^2 + a)\sigma_d^2 \end{aligned} \tag{6.32}$$

where ξ_{min} is the infinite precision LMS minimum MSE, $tr(\mathbf{A})$ is the trace of the correlation matrix $\mathbf{A}$, i.e. sum of diagonal elements, a is a random variable dependent on the inner product of $\mathbf{w^H}_q(k)$ and $\mathbf{u}_q(k)$ and σ_a^2 is the variance of a [4, 18]. From (6.31) and (6.32), it is clear that the step size directly contributes to the performance of the systems. Where, a decrease in the step size μ leads to a decrease in the misadjustment ρ and thus an improved performance [2, 4]. Moreover, by evaluating the third term in (6.32), a decrease in the step size μ increases the quantization error effect and causes the system to deviate from the infinite precision, theoretical, performance. Finally, the final term in (6.32) is only a function of the quantization errors $\eta_u(k)$ and $\eta_y(k)$ [2, 4, 18]. Therefore, it can be concluded that, in finite precision mode, the correct selection of the step size μ directly contributes to the stability and convergence of the systems. As such, μ may be decreased to a certain level, where the degradation effects of the quantization error become significant [2, 4].

6.3 DRC-pLMS Pipelined Hardware Implementation

Similar to the delayed LMS (DLMS) and the delayed parallel LMS (DpLMS) the RC-pLMS pipeline architecture is obtained through the application of the delay and sum relaxed look ahead technique (DRC-pLMS) [4, 21, 24]. As such, assuming a WSS process, (6.9) is modified by adding a delay relaxation of D_1 samples in the error path and D_2 delay in the weight update path, as follows [4]:

$$\mathbf{w}(k+1) \;=\; \mathbf{w}(k-D_2) + \mu \sum_{i=0}^{D_2-1} e^*_{RCpLMS}(k-D_1-i)\mathbf{x}(k-D_1-i) \qquad (6.33)$$

From (6.33), it is clear that the hardware overhead is extensive, i.e. $N(D_2-1)$. Additionally, for larger values of N and D_2, the hardware overhead becomes unacceptably high resulting in a complex, undesirable structure [4, 21]. To reduce the occurring overhead, (6.33) is modified by applying a sum relaxation of D_3 samples with $1 \leq D_3 \leq D_2$. As such, (6.33) becomes [4]:

$$\mathbf{w}(k+1) \;=\; \mathbf{w}(k-D_2) + \mu \sum_{i=0}^{D_3-1} e^*_{RCpLMS}(k-D_1-i)\mathbf{x}(k-D_1-i) \qquad (6.34)$$

Assuming μ is small enough, the DRC-pLMS error is computed as:

$$e_{RCpLMS}(k) \;=\; S(k) - \mathbf{w}^{\mathbf{H}}(k-D_2)\mathbf{u}(k) \qquad (6.35)$$

In contrast to the traditional look ahead technique, the delay and sum relaxed look-ahead technique does not result in a unique architecture. However, it may be considered as a transformation in the stochastic sense since the average output profile is conserved [4, 21].

6.3.1 DRC-pLMS Hardware Architecture

DRC-pLMS is implemented in a finite precision mode for a $Q2.15$ [1] format. The delay relaxation is initialized such as $D_1 = 4$, $D_2 = 2$ and $D_3 = 1$, i.e. six pipeline stages. The DRC-pLMS architecture is shown in Figure 6.2, where $\mathbf{u}_{1..4}(k)$, $\mathbf{u}_{5..8}(k)$, $\mathbf{w}_{1..4}(k)$

1. 1 signed bit, 2 integer bits and 15 precision bits

and $\mathbf{w}_{5..8}(k)$ are the DRC-pLMS input and weight vectors formed by the first and last 4 elements, respectively. z^{-1}, z^{-D_1} and z^{-D_1-1} represent a digital delay element, i.e. register, of 1, D_1, and $D_1 - 1$ samples, respectively and the Conj block denotes the complex conjugation [4].

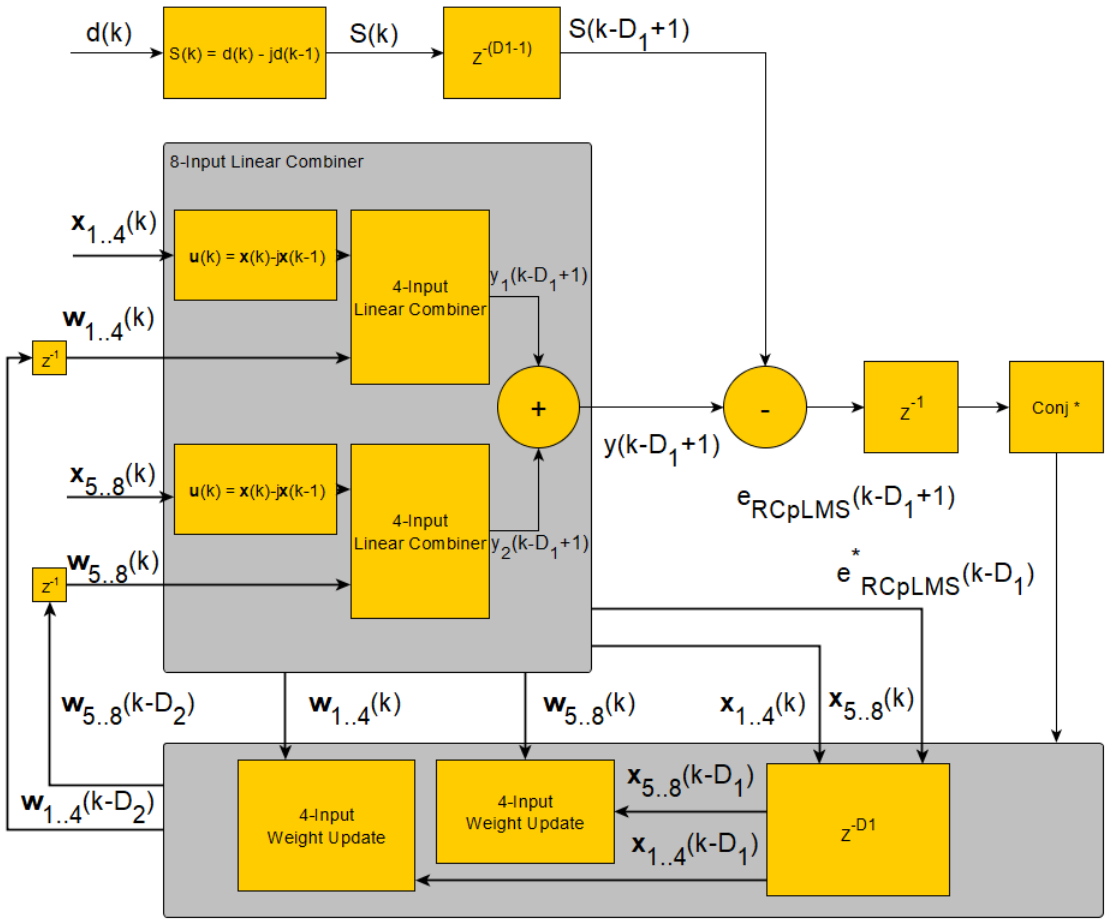

Figure 6.2 – 8-Input DRC-pLMS Beamformer Architecture [4]

In addition, y_1 and y_2 form the intermediate output and they are defined such as:

$$y_1(k) = \mathbf{w}_{1..4}^{\mathbf{H}}(k)\mathbf{u}_{1..4}(k) \tag{6.36}$$

$$y_2(k) = \mathbf{w}_{5..8}^{\mathbf{H}}(k)\mathbf{u}_{5..8}(k) \tag{6.37}$$

The 8 input adaptive beamformer is formed by a grouping of two 4 input linear combiner blocks and two 4 input weight update blocks, as shown in Figure 6.3. The LMS default input signals $\mathbf{x}(k)$ and $d(k)$ remain unchanged, however they are internally modified to form the new DRC-pLMS input signals $\mathbf{u}(k)$ and $S(k)$ [4]. The final output y is then obtained as a combination of the individual linear combiner outputs, y_1 and y_2 and is used to compute the total error e_{RCpLMS}. Finally, the total error, e_{RCpLMS}, is used to compute the new filter weights with respect to the input signals $\mathbf{x}_{1..4}(k)$ and $\mathbf{x}_{5..8}(k)$. It should be highlighted, with respect to Figure 6.2, that the DLMS architecture can be obtained by eliminating the input modification blocks forming $\mathbf{u}(k)$ and $S(k)$ [4].

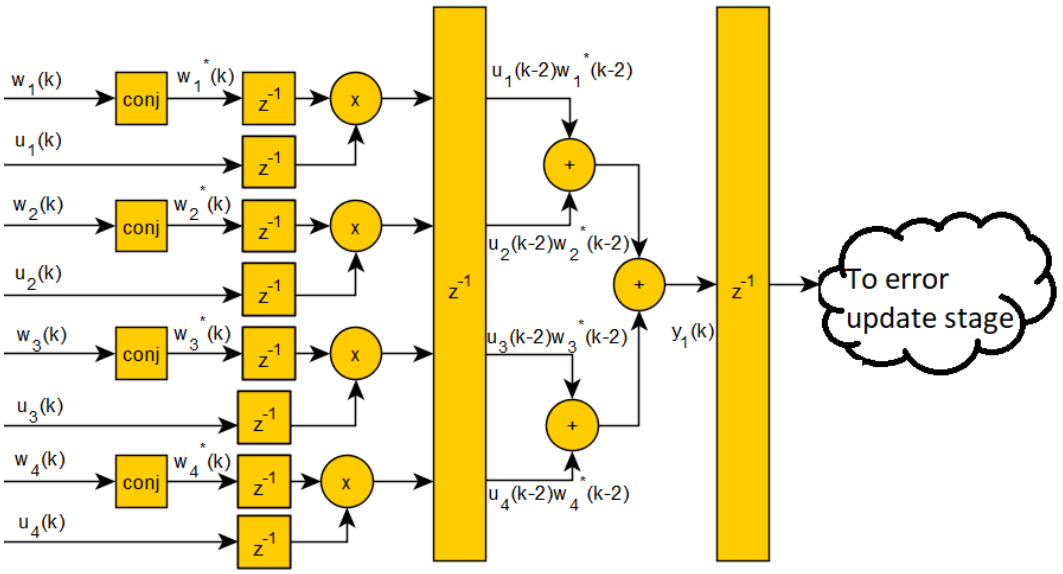

Figure 6.3 – DRC-pLMS 4-Input Linear Combiner Architecture [4]

The 4-input linear combiner and weight update blocks are presented in Figures 6.3 and 6.4, respectively. From Figure 6.3, we observe that the pipeline stages are formed of independent parallel multiplication and addition operations and thus requires a computational complexity of $O(1)$ [4]. Where each complex multiplier is formed of four real multipliers and one complex adder, i.e. two real adders [4]. In addition, the update term shown in Figure 6.4, is computed with respect to an arithmetic shift right of nu bits. The shifting operation is performed at the wire speed and thus eliminates the need of an

additional multiplier [4].

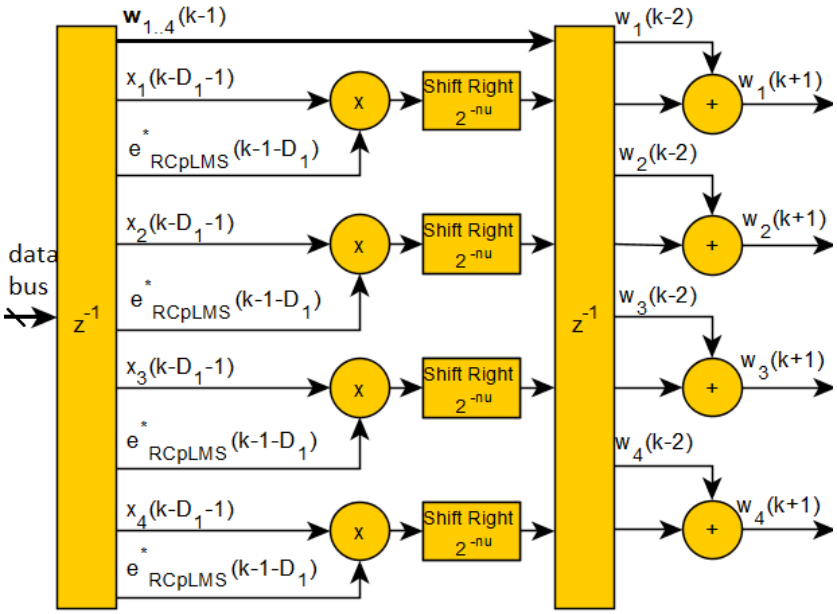

Figure 6.4 – DRC-pLMS 4-Input Weight Update Architecture

6.3.2 Implementation and Synthesis Results

The DRC-pLMS is implemented on the "Intel Stratix V 5SGXMABN3F45I4" model and compared to the VR-SNC-TDNLMS [23] and the LUT-Less Pipelined LMS [22] as shown in Table 6.1 [4]. Since the VR-SNC-TDNLMS [4, 23] and the LUT-Less Pipelined LMS [4, 22] implementation is conducted for different antenna count, different processors and different finite precision configuration, only the maximum operating frequency is compared.

As shown in Table 6.1, compared to the VR-SNC-TDNLMS [4, 23] and the LUT-Less Pipelined LMS [4, 22] variants, the DRC-pLMS achieved a pipeline structure operating at a maximum frequency of 208.33 MHz. Moreover, in contrast to the DpLMS, the DRC-pLMS only requires the use of 32 DSP blocks, i.e single LMS stage. However, compared to the LMS and DLMS [21], the proposed RC-DpLMS provides a superior convergence behavior and a lower steady state error at the cost of a negligible increase in resource

utilization, i.e. logic elements and registers [4].

Design	DSP	Registers	Logic Elements	Frequency (MHz)
DRC-pLMS	32	2065	905	208.33
DpLMS	64	3636	1567	208.33
DLMS	32	1746	773	208.33
VR-SNC TDNLMS [23]	276	10616	19374	124.144
LUT-Less Pipelined LMS [22]	-	1623	5811	84.24

Table 6.1 – 8-Input RC-pLMS Beamformer Synthesis Results

6.4 Hardware and Software Simulations

A Monte Carlo type simulation is conducted to assess the behavior of the RC-pLMS, the RC-pLMS transfer function subject to fractional delay (RC-pLMS-FD) and the DRC-pLMS with respect to the LLMS, pLMS and RLS adaptive algorithms.

Algorithm	Initial Parameters
LMS	$\mu_{LMS} = 0.5$
RLS	$\alpha = 0.98$, $\mathbf{L}(0) = 0.5\mathbf{I}$, $\mathbf{Q}(0) = 0.025\mathbf{I}$
LLMS	$\mu_{LMS1} = 0.0156$ $\mu_{LMS2} = 0.0156$,$\vartheta = 0.0004$
pLMS	$\mu_{LMS1} = 0.0156$, $\mu_{LMS2} = 0.0156$
RC-pLMS	$\mu = 0.0156$, $nu = 6$
DRC-pLMS	

Table 6.2 – Simulation Initial Parameters

The simulation is performed for 500 realizations of 500 samples each where the input

source is modeled as a ULA array with $N = 8$ elements. The input signal is formed by a combination of a message signal and two interferes impinging the array from broadside at different angles of arrival (AOA) of 30°, 45° and 80°, respectively [4]. The inputs are generated as independent random complex Gaussian sequences of the form $v_m = \mathcal{N}(0, \sigma_p^2) + j\mathcal{N}(0, \sigma_q^2)$ where, $\mathcal{N}$ denotes normal, zero mean, (Gaussian) distribution. σ_p^2 and σ_q^2 are the real and complex sequence variances, respectively. The final input signal is corrupted by CAWGN for a signal to noise ratio (SNR) of 10 dB and $\sigma_p^2 = 0.01$ and $\sigma_q^2 = 0.04$. The parameters and initial conditions at $k = 0$ are initialized for the LMS and its variants with respect to Table 6.2, where $\mathbf{I}$ is the identity matrix.

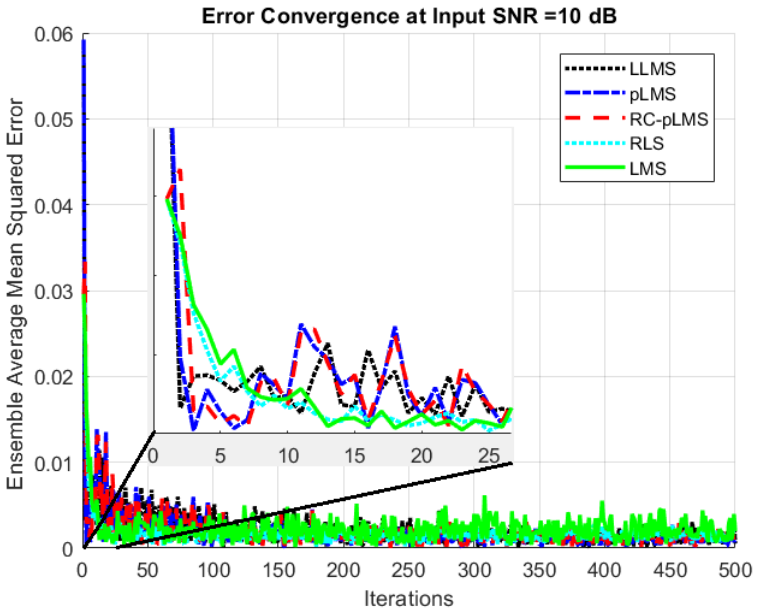

Figure 6.5 – RC-pLMS MSE Convergence Comparison [4]

6.4.1 Mean Square Error Analysis

The MSE simulation is used to study the convergence behavior of the least mean square - least mean square (LLMS), the pLMS, the RC-pLMS, the LMS and the recursive least square (RLS) [4]. From Figure 6.5, the RC-pLMS presented identical pLMS and LLMS convergence behavior. Moreover, for an SNR of 10 dB and in contrast to the LMS

with a step size of 0.5, i.e. maximum suggested step size [2, 4], the RC-pLMS preserved a superior convergence profile with the use of a much smaller step size $\mu_{RCpLMS} = 0.0156$ [4]. Additionally, RC-pLMS achieved its convergence in 3 iterations, and a low steady state error < 0.005 [4]. The use of a small step size allows the RC-pLMS to maintain its convergence in low SNR environments [4, 14]. However, compared to the RLS with a computational complexity of order $O(N^2)$, the suggested RC-pLMS presented faster convergence while maintaining a LMS like computational complexity of order $O(N)$ [4]. Moreover, the MSE behavior describes the systems convergence behavior with respect to the transient (pLMS) and the steady-state (RC-pLMS) approximation, thus validating the use of single set of weights as presented in (6.9).

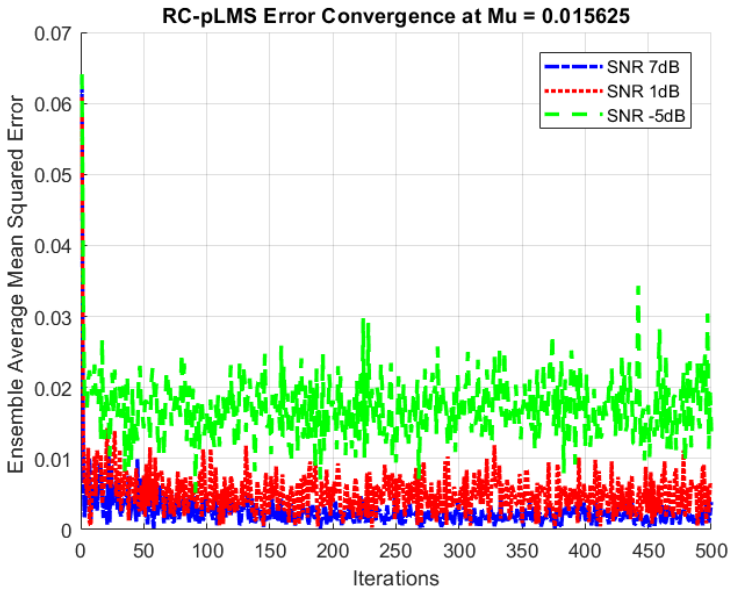

Figure 6.6 – RC-pLMS Convergence Behavior [4]

To further assess the performance and the stability of the RC-pLMS and the DRC-pLMS for different SNR environments, the MSE simulation was re-evaluated for multiple SNR, i.e. -5 dB to 7 dB with a step of 6 dB [4]. From Figure 6.6, it is clear that the RC-pLMS maintained its accelerated convergence and low residual error profile for a SNR = 1 dB. In contrast, for an SNR = -5 dB, RC-pLMS only maintain an accelerated convergence

however with a larger residual error [4].

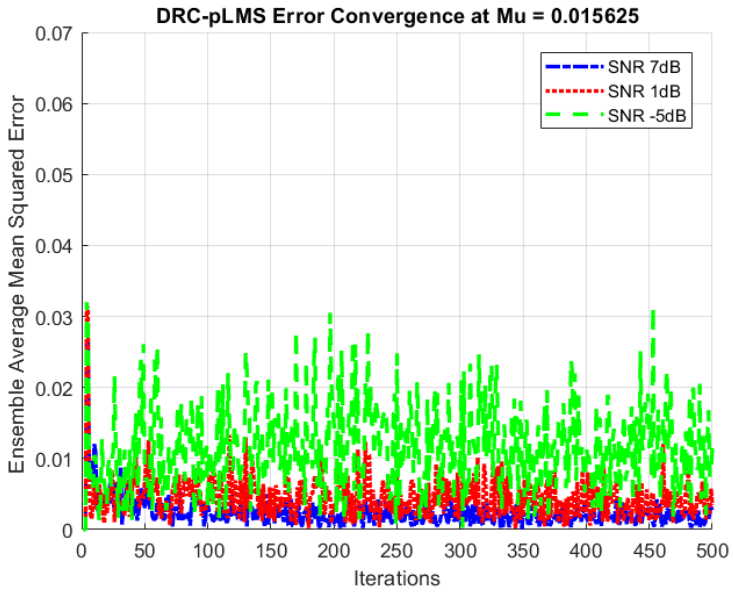

Figure 6.7 – DRC-pLMS Convergence Behavior [4]

In contrast, as shown in Figure 6.7, the DRC-pLMS was able to maintain accelerated convergence at SNR = 1 dB, however with a larger residual error. Such degradation is a consequence of the sum relaxation adopted in (6.34) [4].

6.4.2 Beam Radiation Pattern

The beam radiation pattern simulation output is presented, in Figure 6.8, to illustrate and compare the beam directivity of the RC-pLMS, the DRC-pLMS and the finite precision DRC-pLMS to that of the finite precision DLMS. From Figure 6.8, and for a SNR environment of 10 dB, similar to the DLMS, the proposed finite precision DRC-pLMS is able to accurately steer the output main beam towards the desired AOA, i.e. 30° while nulling interfering signals at 45° and 80° [4]. Moreover, the finite precision, DRC-pLMS shows near identical results to the theoretical models denoted by RC-pLMS and DRC-pLMS [4]. The RC-pLMS and DRC-pLMS beam radiation pattern is re-evaluated for different SNR environments, i.e.−5 dB to 7 dB with a step of 6 dB [4].

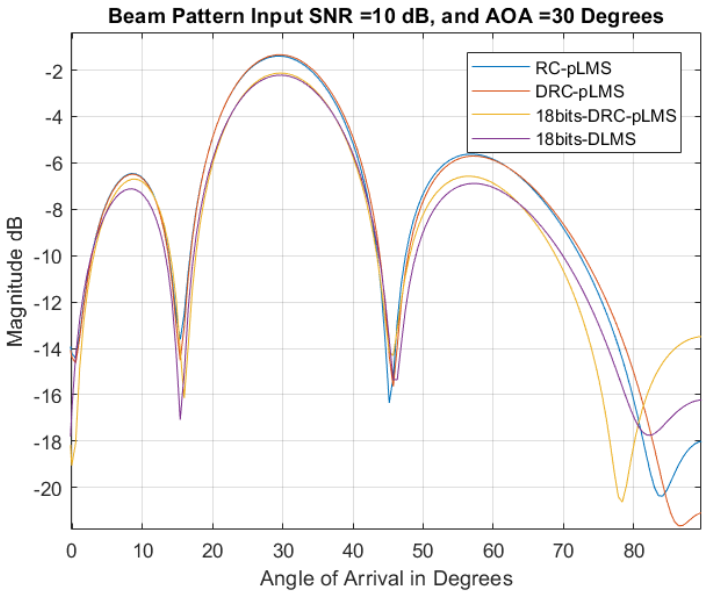

Figure 6.8 – Infinite and Finite Precision Beam Radiation Pattern for an Angle of Arrival 30° [4]

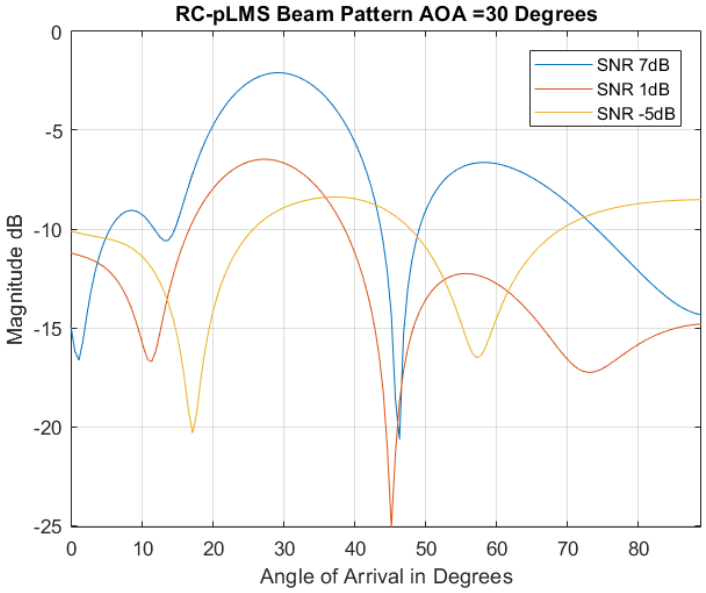

Figure 6.9 – RC-pLMS Beam Radiation Pattern for an Angle of Arrival 30° [4]

From Figure 6.9, the RC-pLMS successfully steered its main beam and nulls towards the AOA of interest, with an acceptable accuracy, for a SNR environment as low as 1 dB [4]. In contrast, as shown in Figure 6.10, the DRC-pLMS has managed to only steer its main beam towards the AOA of the desired signal for an SNR environment of up to 1 dB. The resulting degradation in beam accuracy perfectly align with the MSE results presented in Figures 6.6 and 6.7 where the average MSE becomes unacceptable for SNR $<$ 1 dB [4]. Additionally, the difference in accuracy between the RC–pLMS and DRC-pLMS is caused by the sum approximation implied in (6.34) [4]. Such difference can be omitted for increased accuracy by setting $D_3 = D_2$, however at the cost of a considerable increase in resource usage [4].

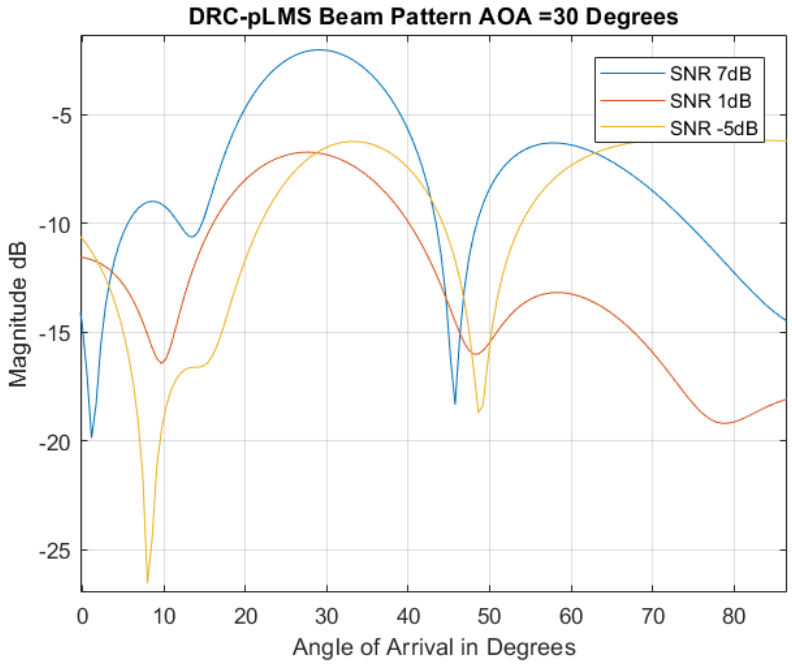

Figure 6.10 – DRC-pLMS Beam Radiation Pattern for an Angle of Arrival 30° [4]

6.4.3 Pole Zero Map Stability Plot

In order to numerically determine the maximum step size allowed for the RC-pLMS a pole zero plot is performed for the transfer function approximation in (6.26). The RC-

pLMS stability analysis is studied for different μ values and the pole zero plot is shown in Figure 6.11. As the step size μ increases the resulting poles and zeros move further towards the outside of the unit circle.

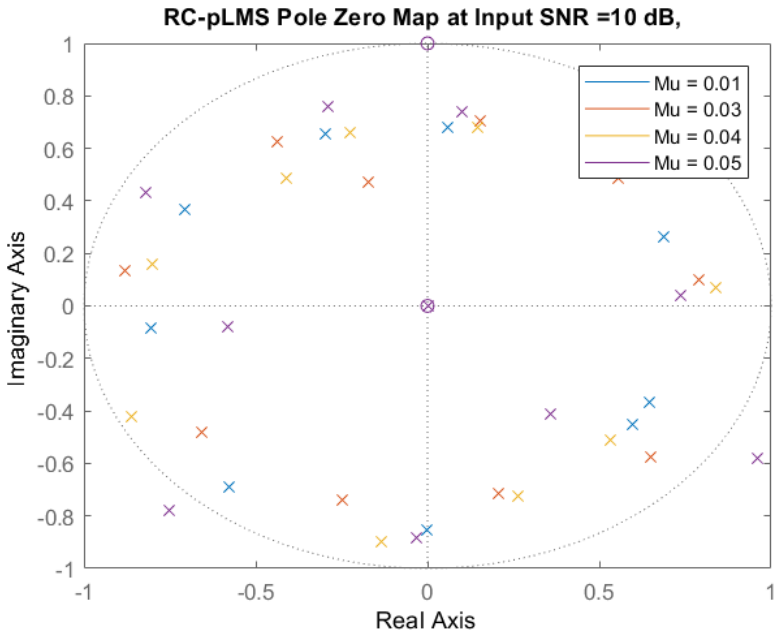

Figure 6.11 – RC-pLMS Pole Zero Map for Different μ

Thus, it can be concluded from the conducted study that the maximum step size to ensure convergence and stability falls in the range of $0.03 < \mu < 0.04$.

6.4.4 Computational Complexity Comparison

A resource complexity comparison for the RC-pLMS, the RLS, the LMS and their variants is presented in Table 6.3, where cMultiply, cAdd, cDivide and RLMSp denote the complex multiplication, the complex addition, the complex division and the parallel RLMS, respectively [4].

Algorithm	cMultiply	cAdd	cDivide
RLMS[15]	$3N^2 + 11N + 2$	$2N^2 + 9N + 6$	$N + 1$
RLMSp[6]	$3N^2 + 7N + 1$	$2N^2 + 6N + 3$	1
RLS	$3N^2 + 5N$	$2N^2 + 4N + 2$	1
LLMS[5]	$6N + 2$	$5N + 4$	N
pLMS	$4N + 2$	$4N + 4$	0
RC-pLMS [4]	$2N + 1$	$3N + 2$	0
LMS	$2N + 1$	$2N + 1$	0

Table 6.3 – Theoretical Complexity and Resource Usage [4]

From Table 6.3, the RLS algorithm and its variants require an higher computational complexity of order $O(N^2)$. Moreover, both the RLMS and the RLMSp mandate the use of $N + 1$ and 1 complex dividers, respectively [4]. In contrast, compared to the LLMS and the pLMS, the RC-pLMS provides near identical LMS resource usage and linear computational complexity of order $O(N)$ while providing a superior convergence profile and a lower residual error floor [4]. Thus, with respect to a marginal increase in resource requirements, i.e. additional adders and subtractors, the RC-pLMS eliminated the LMS convergence speed and error floor trade off, while preserving a LMS like low complexity structure.

6.5 Conclusion

In this chapter, we proposed a reduced complexity parallel LMS structure (RC-pLMS) for adaptive beamforming, we evaluated its transfer function approximation and we implemented its pipeline hardware implementation. RC-pLMS is formed of a simple LMS stage with additional modified inputs. The modified inputs are formed by applying a sample delay to the input and desired signals and multiplied by a phase shift, i.e. imaginary number j. The RC-pLMS transfer function approximation is obtained by modeling the input spatial linear combiner as a finite impulse response (FIR) fractional delay temporal filter with a Lagrange interpolation. Moreover, we propose a pipeline hardware imple-

mentation through the application of the delay and sum relaxed look ahead technique (DRC-pLMS). Stability and quantization effect analysis are performed to determine the upper bound of the step size required to ensure convergence, as well as the behavior of the system with finite precisions. The maximum allowable step size is determined numerically with respect to the transfer function approximation and the pole zero plot study. Simulation results demonstrate the superior performance of the RC-pLMS and DRC-pLMS in providing accelerated convergence and maintaining a low steady state error while preserving a complexity of order $O(N)$. The RC-pLMS beam pointing accuracy was evaluated to better highlight its precision and was represented with the beam radiation pattern. Compared to the LLMS and pLMS the RC-pLMS presented a high accuracy beam pattern by accurately steering its main beam towards the desired signal better nulls towards the two interreferences. Furthermore, the RC-pLMS superior performance was achieved by a marginal increase in resource usage, compared to the LMS. Also, compared to the Recursive Least Square (RLS) with quadratic complexity, the RC-pLMS presented near identical behaivor in a linear complexity form. With respect to the pLMS, the RC-pLMS presented similar behavior at half the resource requirements. RC-pLMS and DRC-pLMS algorithms presented a satisfactory convergence profile and beam patterns for a SNR environment as low as 1 dB. Through a pole zero plot analysis the maximum step size allowed to maintain proper RC-pLMS convergence is found to be in the range of $0.03 < \mu < 0.04$. Synthesis and implementation results show that, in contrast to other RLS and LMS variants, the RC-pLMS provides a parallel, pipeline, low complexity and resource friendly architecture suitable for low end processors. Finally, the fixed point DRC-pLMS beam radiation pattern validated the DRC-pLMS accuracy in a finite precision mode by providing similar infinite precision beam steering. The resulting work has been published in the IEEE Transaction of Circuits and Systems.

Conclusion and Future Work

Conclusion

Adaptive beamforming has become an inevitable feature in smart antenna systems to infer frequency reuse and ease spectral congestion by directional signal reception and transmission. However, the recent, exponential, growth of internet connected devices has exhausted spectral resources and imposed challenging constraints on adaptive algorithms when implemented on dedicated, limited resource, devices, i.e. field programmable gate array (FPGA). Such constraints are reflected by the requirements of the system to deliver accelerated convergence and high accuracy while maintaining a low complexity and a high throughput architecture.

The least mean square (LMS) is a popular and widely used adaptive algorithm in beamforming given its attractive, low complexity, $O(N)$ structure, where N is the number of antenna elements, most suitable for a hardware implementation. However, in contrast to its desirable features, the LMS still suffers from a trade off between its convergence speed and steady state error, i.e. accuracy. Moreover, presenting a pipeline LMS architecture is difficult due to the presence of the error feedback path in the weight update equation.

Several variants of the popular least mean square (LMS) have been proposed to present an improved convergence profile with minimal residual error. Similarly, several work has been conducted to implement high throughput, low complexity, pipeline parallel LMS architecture on FPGA. Such modifications are, and not limited to, the use of either of the following:

— The use of a normalized or variable step size to eliminate the LMS error floor and convergence speed trade off, such as: the normalized LMS (NLMS) [48], the variable step size LMS (VSSLMS) [49, 52] and the modified robust VSSLMS (MRVSSLMS) [50].

— The use of a multi stage adaptive algorithm to accelerate LMS convergence while

maintaining minimal steady state error, such as: the least mean square - least mean square (LLMS) [5], the recursive least mean square (RLMS) [15], the Kalman-LMS and Kalman-RLS [13], the parallel RLMS (RLMSp) [6] and the parallel LMS (pLMS) [14].

— The use of different delay and approximation techniques to present a pipeline LMS architecture, such as: the delay and sum relaxed look-ahead technique [21, 24], the time shared look up table (LUT)-less LMS [22] and the division free and variable regularized LMS [23].

Despite the listed modifications, the resulting LMS variants either presented an improved convergence profile, at the cost of an increase of complexity or presented a pipeline hardware architecture without any significant improvement on the theoretical performance, i.e. convergence speed and accuracy.

To present a high accuracy and fast converging adaptive beamforming structure, while maintaining an easy to pipeline low complexity LMS like design, we proposed the reduced complexity parallel LMS (RC-pLMS) algorithm. The RC-pLMS is formed of a single LMS stage with additional modified inputs and a linear computational complexity of order $O(N)$. Moreover, we implemented the RC-pLMS design in a parallel pipeline high throughput architecture through the application of the delay and sum relaxed look ahead technique (DRC-pLMS). The superior performance of the RC-pLMS and the finite precision DRC-pLMS has been demonstrated through extensive software and hardware simulations. Our contributions throughout this thesis are detailed as follows:

— We performed an experimental study in order to evaluate the behavior of popular signal processing algorithms, i.e. fast Fourier transform (FFT), used in frequency domain beamforming when implemented on limited resource devices. The FFT was implemented considering different architectures, i.e. sequential or parallel, different processors, i.e. FPGA and system on chip (SoC) and different design tools, traditional hardware description language (HDL), high level language (HLL) and high level synthesis (HLS) tools. Through simulation and synthesis results, we demonstrated the efficiency of using HLL and HLS design tools in targeting heterogeneous multi core architectures with embedded digital signal processing (DSP) processors. Additionally, the use of HLS tools resulted in improved design optimization, com-

pared to the classical HDL tools, through the use of specific compiler directives, thus reducing design and testing time by two thirds. However, since the HLL optimization efficiency is highly correlated with the HLS compiler, it becomes mandatory to use simple and atomic design techniques. We published this contribution in the IEEE International Multidisciplinary Conference on Engineering Technology (IMCET) [77].

— While the FFT processor is one of the core arithmetic operations in frequency domain beamforming, its performance significantly contributes towards the algorithms convergence speed and beam pointing accuracy. Thus, in this thesis, we proposed a low complexity, high accuracy, dynamic twiddle factor generator based on the Chebyshev polynomial approximation for the sine and cosine functions. Additionally, we proposed its finite precision and high throughput parallel pipeline, hardware implementation. Simulation and synthesis results illustrated the upper level accuracy of the proposed Chebyshev polynomial approximation compared to the traditional Taylor and coordinate rotation digital computer (CORDIC) techniques. We should highlight that only the Chebyshev polynomial approximation achieved a high precision, three decimal digits, low complexity high throughput design. This work has been published in the Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES) conference [59].

— Inspired by the enhanced performance of the multi stage adaptive beamforming algorithms, i.e. LLMS and RLMS. We conducted an experimental study for different multi stage designs and studied the effect of the steering vector estimation, i.e the cascading stage, on the behavior of the system. Experimental results concluded that the use of the steering vector cascading stage can severally degrade the accuracy of the system and might cause it to diverge from its theoretical value as a consequence of the division operation, i.e. underflow. Moreover, the estimate cascading stage introduces additional computational complexity and thus a larger hardware overhead when implemented on FPGA. Additionally, the cascading nature of the system makes it extremely difficult to present a pipeline and parallel hardware architecture. This contribution has been published in the KES International Conference on Intelligent Decision Technologies (KES-IDT) [13].

— In order to eliminate the requirements of using a steering vector estimate block, we proposed the RLMSp multi stage adaptive beamformer design with parallel inputs. Additionally, the second stage, LMS feedback error was multiplied by a phase shift, i.e. the imaginary number j, for robustness against repeated samples. Simulation results demonstrated the superior performance of the proposed RLMSp with respect to the cascade multi stage structures, i.e. LLMS and RLMS. It has been shown that the proposed RLMSp maintained an accelerated convergence and high accuracy profile under different SNR environments, however with a reduced complexity and a parallel ready architecture. Moreover, by omitting the use of the steering vector estimate block the complexity of the system was reduced by $20N$ real multiplications, $6N$ real additions and $2N$ real divisions. We published this part of our work in the European Signal Processing Conference (EUSIPCO) [6].

— As, the RLMSp still exhibits $2N$ real divisions and an undesirable computational complexity of order $O(N^2)$ making it a unfavorable choice for a pipeline hardware implementation. Thus, we proposed a two stages parallel input LMS structure, i.e. pLMS, for adaptive beamforming and its transfer function approximation. Additionally, in order to present a high throughput pipeline architecture we propose the application of the delay and sum relaxed look ahead technique (DpLMS) for each of the LMS stages. Simulation results demonstrated the superior performance of the pLMS in maintaining an accelerated convergence and a low steady state error profile in an SNR environment as low as 1 dB, compared to other variants. In contrast to the RLMSp, the pLMS exhibits an attractive, linear, computational complexity of order $O(N)$ and does not necessitate the use of extensive division routines. The theoretical stability analysis is conducted to determine the upper bound of the step size. Additionally, the maximum pLMS step size was numerically determined with respect to the transfer function approximation. Moreover, the pipeline DpLMS design was implemented on FPGA and showed identical, theoretical, beam pointing accuracy, thus further validating its efficiency in finite precision. Synthesis results showed that the pLMS achieved a maximum operating frequency of 208.33 MHz in a low complexity, low latency high throughput design. The work conducted for the pLMS derivation and its hardware implementation has been separately published

in the European Signal Processing Conference (EUSIPCO) [14, 24].

— While the pLMS presented highest performing and the easiest to pipeline structure, it still doubles the complexity of the classical LMS and requires the use of two independent LMS filters. As such, we proposed The RC-pLMS makes use of a single LMS stage in a similar low complexity design. Similar to the pLMS, in order to implement the RC-pLMS in a low latency, high throughput architecture on FPGA, we proposed the application of the delay and sum relaxed look ahead technique (DRC-pLMS). Simulation results demonstrated the outstanding performance of the proposed RC-pLMS compared to different adaptive beamforming algorithms. In contrast to the LLMS and pLMS variants, the RC-pLMS reduced the systems resource requirements by half, i.e single LMS stage, while maintaining accelerated convergence, high accuracy and robustness against repeated samples for a signal to noise ratio (SNR) environment as low as 1 dB. Stability analysis and quantization effect have been performed to study the behavior of the system and its robustness. Additionally, the maximum RC-pLMS step size was numerically determined with respect to the transfer function approximation. To further assess the behavior of the system, and compare it to that of the infinite precision theoretical value, we implemented the DRC-pLMS in finite precision mode on FPGA. Hardware simulation results, for the finite precision DRC-pLMS, showed similar infinite precision theoretical convergence behavior and beam pointing accuracy. Synthesis results show that the DRC-pLMS reached a greater operating frequency and low resource usage compared to other LMS implementations. Moreover, compared to the classical LMS pipeline implementation, the DRC-pLMS is achieved at the cost of a marginal and negligible increase in resource usage, i.e. two adders. The resulting work has been published in the IEEE Transaction of Circuits and Systems [4].

Finally, we can state that the RC-pLMS and the finite precision DRC-pLMS satisfied the set constraints by presenting a high performance, high accuracy adaptive beamforming algorithm while maintaining a low complexity high throughput LMS like hardware architecture.

Future Work

Adaptive beamforming is still an attractive area of research and with the recent achievement in communications and processor architectures, the benefits become countless.

I - In the short to medium term, we should try to enhance the results obtained through this thesis. Indeed, the RC-pLMS developed in chapter 6 of this thesis, is designed for a single linear antenna system and narrow band signals. To further explore its potential, we are planning to extend the actual version by investigating the following improvements:

a) In the proposed design, the RC-pLMS implements at its core a classical LMS algorithm, as such it would be worthwhile to study the performance of the system with respect to other variants, i.e. genetic LMS algorithm.

b) 3G, 4G, 5G and further generations dedicated to multimedia and fast internet access should consider wide band signals. Therefore, the RC-pLMS algorithm should be evolved to deal with such signals.

c) To conduct real world experiments and assess the systems behavior with respect to the fading and scattering effects.

d) With respect to the unprecedented popularity of the nanosatellite "CubeSat" communication systems and the spacing constraints they propose, it becomes interesting to investigate the RC-pLMS adaptive algorithms behavior and resources utilization for different antenna array structures, i.e. rectangular and circular, and study its robustness. Due to the strict CubeSat geometry constraints imposed, the proposed algorithm should be evaluated for robustness against the presence of element gain and different inter-element spacing.

II - Light fidelity (Li-Fi) communication systems [88] have gained popularity in indoor usage for communication and localization given their high speed transmission rate and free license. Thus, it would be challenging to implement a high accuracy beamformer system with a low complexity hardware architecture.

III - 5G+ and 6G networks are key enablers of the future intelligent, communication, networks and machines and are expected to provide multi-user multi-terabyte per

second (Tb/s) data rates [89]. 5G+ and 6G networks will probably use dedicated artificial intelligence processors and will require real time inference. As such, it will be extremely challenging to develop adaptive beamforming algorithms, for intelligent networks, through the use of deep learning systems and dedicated artificial intelligent processors that meets the networks requirements. We are willing to extend our research axes to include these new challenges.

APPENDIX A

Steering Vector Estimate

The output of the individual taps of the beamformer are given by:

$$x_m^{'}(k) \quad = \quad w_{1,m}x_{1,m}(k) \tag{6.38}$$

When the first stage, RLS, adaptive algorithm converges the output y_{RLS} tends to approach the desired signal $s_d(k)$ by suppressing the interference and noise signals. Therefore, let the beamformer output be defined by:

$$y_{RLS}(k) \quad \simeq \quad s_d(k) \tag{6.39}$$

At convergence and by applying the expectation operator to both sides of equation (2.5), we get:

$$E[x_{1,m}(k)] \quad \simeq \quad \hat{a}_{d,m}E[s_d(k)] \simeq \hat{a}_{d,m}E[y_{RLS}(k)] \tag{6.40}$$

Assuming that after convergence, we can approximate:

$$E[y_{RLS}(k)] \quad \simeq \quad y_{RLS}(k) \tag{6.41}$$

thus equation (6.40) can be rewritten as:

$$E[x_{1,m}(k)] \quad \simeq \quad \hat{a}_{d,m}y_{RLS}(k) \tag{6.42}$$

assuming that the input signal and the available tap weights are uncorrelated the expectation of equation (6.38) can be written as:

$$E[x^{'}_{1,m}(k)] \simeq E[w_{1,m}]E[x_{1,m}(k)] \tag{6.43}$$

therefore, from equations (6.42) and (6.43), we can estimate the array steering vector in expectation and instantaneous form as:

$$\hat{a}_{d,m}(k) \simeq \frac{E[x^{'}_{1,m}(k)]}{E[w_{1,m}]y_{RLS}(k) + \vartheta} \simeq \frac{x^{'}_{1,m}(k)}{w_{1,m}(k)y_{RLS}(k) + \vartheta} \tag{6.44}$$

APPENDIX B

pLMS Mean Square Error

The parallel LMS (pLMS) mean square error (MSE) ξ_{pLMS} is defined as:

$$\begin{aligned}\xi_{pLMS}(k) &= \mathrm{E}[|e_{pLMS}(k)|^2] \\ &= \mathrm{E}[e_1(k)e_1^*(k) + je_1(k)e_2^*(k-1) \\ &\quad -je_1^*(k)e_2(k-1) + e_2(k-1)e_2^*(k-1)] \end{aligned} \tag{6.45}$$

with $|.|$ signifies the complex modulus. Moreover, the first term of (6.45) can be expressed as:

$$\mathrm{E}[|e_1(k)|^2] = \mathrm{E}[|d(k)|^2] - \mathbf{p}^{\mathbf{H}}\mathbf{w}_1(k) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{p} + \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{R}\mathbf{w}_1(k) \tag{6.46}$$

the last term of (6.45) can be developed as follow:

$$\begin{aligned}\mathrm{E}[|e_2(k-1)|^2] &= -\mathrm{E}[d_1(k)\mathbf{x}_1^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) - \mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d_1^*(k)\mathbf{x}(k-1)] \\ &\quad +\mathrm{E}[|d(k-1)|^2] + \mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[\mathbf{x}_1(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \end{aligned} \tag{6.47}$$

In addition, the second term of (6.45) can be detailed as:

$$\begin{aligned}\mathrm{E}[e_1(k)e_2^*(k-1)] &= \mathrm{E}[-d(k)\mathbf{x}^{\mathbf{H}}(k-1)\mathbf{w}_2(k-1) - d^*(k-1)\mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}(k) \\ &\quad +d(k)d^*(k-1) + \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)\mathbf{w}_2(k-1)] \end{aligned} \tag{6.48}$$

and the third term of (6.45) becomes:

$$\begin{aligned}\mathrm{E}[e_1^*(k)e_2(k-1)] &= \mathrm{E}[-d^*(k)\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1) - d(k-1)\mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k) \\ &\quad +d^*(k)d(k-1) + \mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k)\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1)] \end{aligned} \quad (6.49)$$

substituting (6.46), (6.47), (6.48) and (6.49), in (6.45) the MSE ξ_{pLMS} becomes:

$$\begin{aligned}\xi_{pLMS}(k) &= \mathrm{E}[|d(k)|^2] - \mathbf{p}^{\mathbf{H}}\mathbf{w}_1(k) - \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{p} + \mathbf{w}_1^{\mathbf{H}}(k)\mathbf{R}\mathbf{w}_1(k) + \mathrm{E}[|d(k-1)|^2] \\ &\quad -\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) - \mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d^*(k)\mathbf{x}(k-1)] + j\mathrm{E}[d^*(k)d(k-1)] \\ &\quad +\mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[\mathbf{x}(k)\mathbf{x}(k-1)]\mathbf{w}_2(k-1) + j\mathrm{E}[d(k)d^*(k-1)] \\ &\quad -j\mathbf{w}_1^{\mathbf{H}}(k)\mathrm{E}[d^*(k-1)\mathbf{x}(k)] + j\mathbf{w}_1^{\mathbf{H}}(k)\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \\ &\quad +j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_1(k) + j\mathbf{w}_2^{\mathbf{H}}(k-1)\mathrm{E}[d(k)\mathbf{x}(k-1)] \\ &\quad +j\mathrm{E}[\mathbf{x}^{\mathbf{H}}(k)\mathbf{w}_1(k)\mathbf{w}_2^{\mathbf{H}}(k-1)\mathbf{x}(k-1)] - j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_2(k-1) \end{aligned} \quad (6.50)$$

where $\mathbf{w}_1(k)$ is the tap weight of interest.

RC-pLMS Mean Square Error

The RC-pLMS MSE, $\xi_{RC-pLMS}$, can be obtained from (6.50) by setting $\mathbf{w}(k) = \mathbf{w}_1(k) = \mathbf{w}_2(k)$. As such, we obtain:

$$\begin{aligned}\xi_{RC-pLMS}(k) &= \mathrm{E}[|d(k)|^2] - \mathbf{p}^{\mathbf{H}}\mathbf{w}(k) - \mathbf{w}^{\mathbf{H}}(k)\mathbf{p} + \mathbf{w}^{\mathbf{H}}(k)\mathbf{R}\mathbf{w}(k) + \mathrm{E}[|d(k-1)|^2] \\ &\quad -\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}(k-1) - \mathbf{w}^{\mathbf{H}}(k-1)\mathrm{E}[d^*(k)\mathbf{x}(k-1)] + j\mathrm{E}[d^*(k)d(k-1)] \\ &\quad +\mathbf{w}^{\mathbf{H}}(k-1)\mathrm{E}[\mathbf{x}(k)\mathbf{x}(k-1)]\mathbf{w}(k-1) + j\mathrm{E}[d(k)d^*(k-1)] \\ &\quad -j\mathbf{w}^{\mathbf{H}}(k)\mathrm{E}[d^*(k-1)\mathbf{x}(k)] + j\mathbf{w}^{\mathbf{H}}(k)\mathrm{E}[\mathbf{x}(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}(k-1) \\ &\quad +j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}_1(k) + j\mathbf{w}^{\mathbf{H}}(k-1)\mathrm{E}[d(k)\mathbf{x}(k-1)] \\ &\quad +j\mathrm{E}[\mathbf{x}^{\mathbf{H}}(k)\mathbf{w}(k)\mathbf{w}^{\mathbf{H}}(k-1)\mathbf{x}(k-1)] - j\mathrm{E}[d(k)\mathbf{x}^{\mathbf{H}}(k-1)]\mathbf{w}(k-1) \end{aligned} \quad (6.51)$$

where $\mathbf{w}(k)$ is the tap weight of interest.

APPENDIX C

RC-pLMS Performance Evaluation With Modulated Signals

In non-blind, temporal based, adaptive beamforming algorithms, the system undergoes a training phase before transmission in order to receive the training signal, compute its weights and steer its beam pattern accordingly. During the training phase no data is transmitted from the user to the station, and the reference signal is simple and well known. As such, in this section, we evaluate the performance of the RC-pLMS with respect to a simple binary message signal with 2 Phase Shift Keying (PSK) modulation and an amplitude of 0.2. Additionally, we include two interfering signals modulated in 4-PSK and 8 Qadrature Amplitude Modulation (QAM), with an amplitude of 0.1, respectively. The MSE convergence behavior is shown in Figure 6.12.

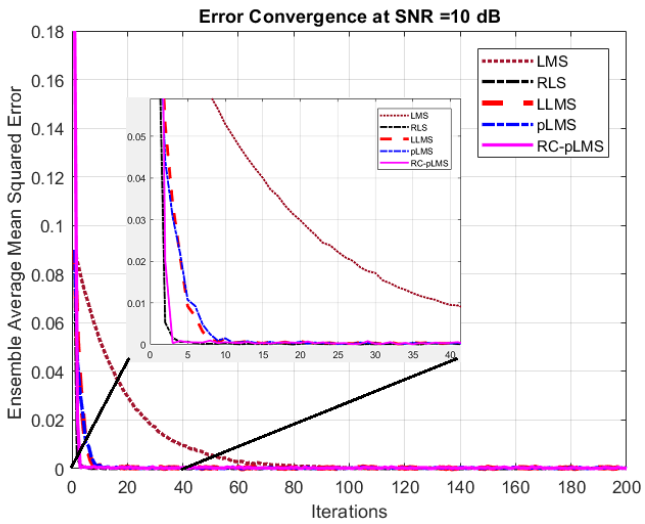

Figure 6.12 – RC-pLMS MSE Convergence Behavior for a Simple 2-PSK Message Signal

As shown in Figure 6.12, compared to the RLS with a quadratic complexity, the linear complexity RC-pLMS presented a fast and smooth convergence behavior where it achieved its convergence in the first 5 iterations.

The RC-pLMS is now re-evaluated, with respect to its beam pattern plot, for different SNR environments. The desired message is incoming at an angle of arrival of 30° and corrupted by two interfering signals incoming at 45° and 80°, respectively.

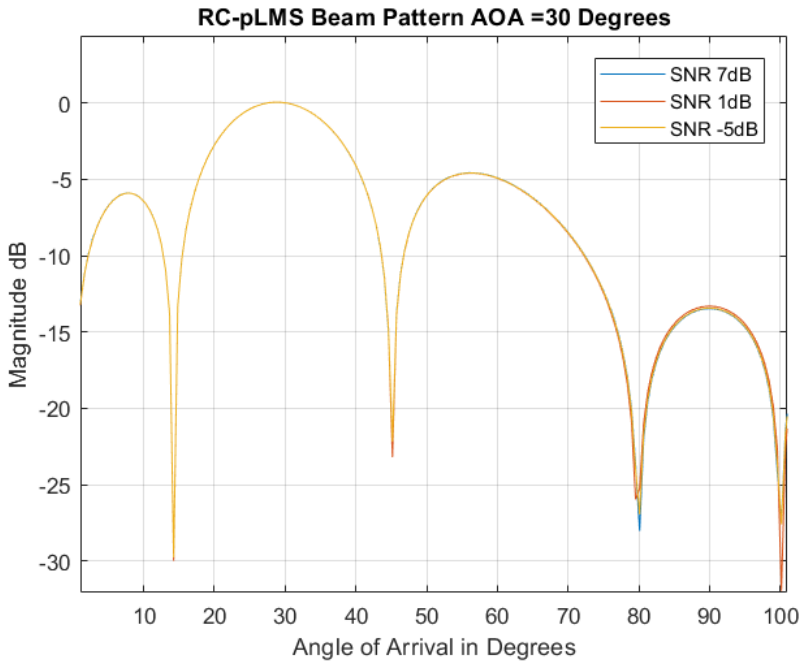

Figure 6.13 – RC-pLMS Beam Radiation Pattern for a Simple 2-PSK Message Signal and Different SNR

As shown from Figure 6.13, the RC-pLMS, for simple signals, maintained an accurate beam pattern for an SNR as low as -5 dB, where it successfully steered its main beam towards the direction of the desired user at 30° and its nulls towards the interfering signals at 45° and 80°, respectively.

In order to further evaluate the performance of the RC-pLMS we plot its MSE beam localization behavior with respect to the angle of arrival.

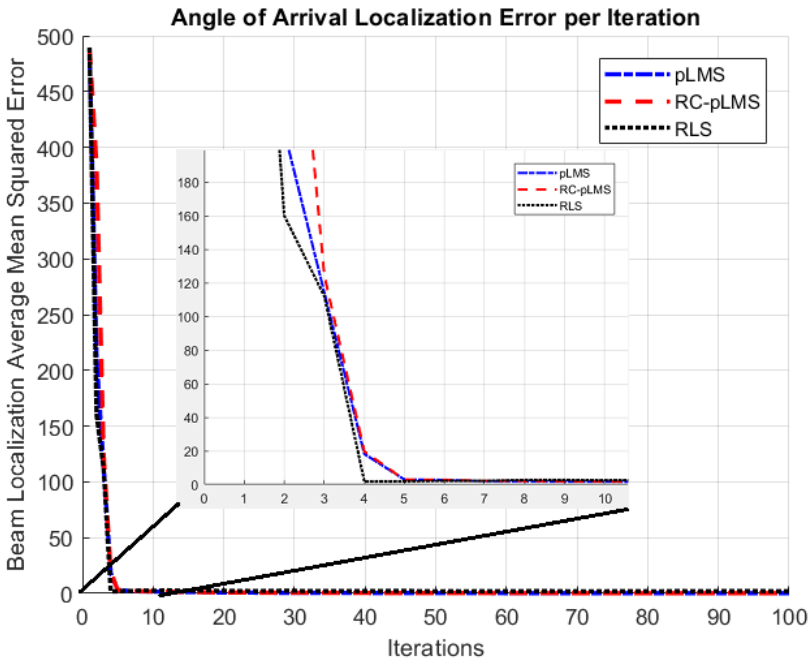

Figure 6.14 – RC-pLMS Beam MSE Localization With Respect to the Angle of Arrival

As shown in Figure 6.14, compared to the RLS with a quadratic computational complexity the linear pLMS presented similar accelerated convergence behavior where it achieved beam convergence in the first few iterations. Also, compared to the pLMS the RC-pLMS presented near identical convergence behavior and accuracy with only half of the resource requirements, thus futrher validating its reliability.

APPENDIX D

pLMS Transfer Function Approximation

The input system described by the N equally spaced, identical antenna elements is modeled as a N^{th} order fractional delay filter employing a Farrow structure and Lagrange interpolation. The new pLMS input signal $\mathbf{x}_f(k)$ can now be defined as:

$$\mathbf{x}_f(k) = \mathbf{y}_{\mathrm{d}}(k) + \sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathbf{y}_{\mathrm{i,j}}(k) + \mathbf{n}(k) \tag{6.52}$$

where $\mathbf{y}_d(k)$, $\mathbf{y}_{\mathrm{i,j}}(k)$ are the message and interfering signals subject to a fractional delay filter and $\mathbf{n}(k)$ is a complex additive white Gaussian noise (CAWGN). The pLMS transfer function approximation can now be derived such that:

$$e_{pLMS}(k) = d(k) - jd(k-1) - y_1(k) + jy_2(k-1) \tag{6.53}$$

Applying the z transform for both sides of (6.53), we obtain:

$$\mathrm{E}[D(z)] - jz^{-1}\mathrm{E}[D(z)] = \mathrm{E}[J_{pLMS}(z)] + \mathrm{E}[Y_1(z)] - jz^{-1}\mathrm{E}[Y_2(z)] \tag{6.54}$$

with

$$\mathrm{E}[J_{pLMS}(z)] = \mathrm{E}[e_{pLMS}(0)] + \mathrm{E}[e_{pLMS}(1)]z^{-1} + \mathrm{E}[e_{pLMS}(2)]z^{-2} + ... \tag{6.55}$$

$$\mathrm{E}[D(z)] = \mathrm{E}[d(0)] + \mathrm{E}[d(1)]z^{-1} + \mathrm{E}[d(2)]z^{-2} + ... \tag{6.56}$$

and

$$y_i(k) = \mathbf{w}_i^{\mathbf{H}}(k)\mathbf{x}_f(k) \tag{6.57}$$

putting $\mathbf{w}_1(0) = \mathbf{w}_2(0) = 0$, we obtain:

$$y_1(k) = \mu_1 \sum_{i=0}^{k-1} \beta_i(k) e_{pLMS}(i) \tag{6.58}$$

$$y_2(k) = \mu_2 \sum_{i=0}^{k-1} \beta_i(k) e_2(i) \tag{6.59}$$

where $\beta_i(k) = \mathbf{x}_f^{\mathbf{H}}(i)\mathbf{x}_f(k)$. Moreover, since $\beta_i(k)$ is time varying, it is difficult to achieve a solvable difference equation. However, we know that:

$$\beta_i(k) = \sum_{j=0}^{N-1} x_{f(i-j)} x_{f(k-j)} = N r_{ki} \tag{6.60}$$

$$r_{ki} = \frac{1}{N} \sum_{j=0}^{N-1} x_{f(i-j)} x_{f(k-j)} \tag{6.61}$$

and r_{ki} is the input signals auto-correlation estimate. Considering the input signal is WSS and its properties can be estimated by a time average to obtain $r_{ki} \approx r_{k-i}$. Hence, (6.58) and (6.59) can be approximated as having constant coefficients:

$$y_1(k) = N\mu_1 \sum_{i=0}^{k-1} r_{k-i} e_{pLMS}(i) \tag{6.62}$$

$$y_2(k) = N\mu_2 \sum_{i=0}^{k-1} r_{k-i} e_2(i) \tag{6.63}$$

Furthermore, applying the Z transform to both sides of (6.62) and (6.63), we get:

$$\mathrm{E}[Y_1(z)] = \mu_1 N \mathrm{E}[J_{pLMS}(z)] R(z) \tag{6.64}$$

$$\mathrm{E}[Y_2(z)] = \mu_2 N \mathrm{E}[J_2(z)] R(z) \tag{6.65}$$

where

$$\mathrm{E}[J_2(z)] = \mathrm{E}[e_2(0)] + \mathrm{E}[e_2(1)] z^{-1} + \mathrm{E}[e_2(2)] z^{-2} + ... \tag{6.66}$$

Additionally, by taking the expectation, it is assumed that r_{k-i} is the auto-correlation coefficient instead of its estimate and $R(z) = r_1 z^{-1} + r_2 z^{-2} + ...$ is a polynomial in the Z field. Furthermore, from (6.54), (6.64) and (6.65), we get:

$$\begin{aligned} \mathrm{E}[D(z)] - jz^{-1}\mathrm{E}[D(z)] &= \mathrm{E}[J_{pLMS}(z)] + \mu_1 N \mathrm{E}[J_{pLMS}(z)]R(z) \\ &\quad -jz^{-1}\mu_2 N \mathrm{E}[J_2(z)]R(z) \end{aligned} \tag{6.67}$$

Using (6.67), and the LMS transfer function approximation given as:

$$H(z) = \frac{\mathrm{E}[J(z)]}{\mathrm{E}[D(z)]} = \frac{1}{1 + \mu L R(z)} \tag{6.68}$$

we can write

$$\mathrm{E}[J_2(z)] = \frac{\mathrm{E}[D(z)]}{1 + \mu_2 N R(z)} \tag{6.69}$$

The pLMS transfer function approximate becomes:

$$H_{pLMS}(z) = \frac{1 + \mu_2 N R(z) - jz^{-1}}{(1 + \mu_1 N R(z))(1 + \mu_2 N R(z))} \tag{6.70}$$

RC-pLMS Transfer Function Approximation

The RC-pLMS is computed following the same procedure of the pLMS and with respect to the LMS transfer function approximation in (6.68). Thus, the RC-pLMS transfer function approximation becomes as follows:

$$H_{RC-pLMS}(z) = \frac{1 - jz^{-1}}{1 + \mu N (1 - jz^{-1}) R(z)} \tag{6.71}$$

APPENDIX E

Arithmetic Complexity

In order to better highlight the performance of each algorithm, in this section we present the MSE convergence behavior with respect to time. This simulation is run on a machine with an Intel Core i5-6500HQ processor, 8 gigabytes (GB) of memory, Windows 10, 64-bits operating system and Matlab version 2017. The MSE convergence plot is shown in Figures 6.15, 6.16 and 6.17.

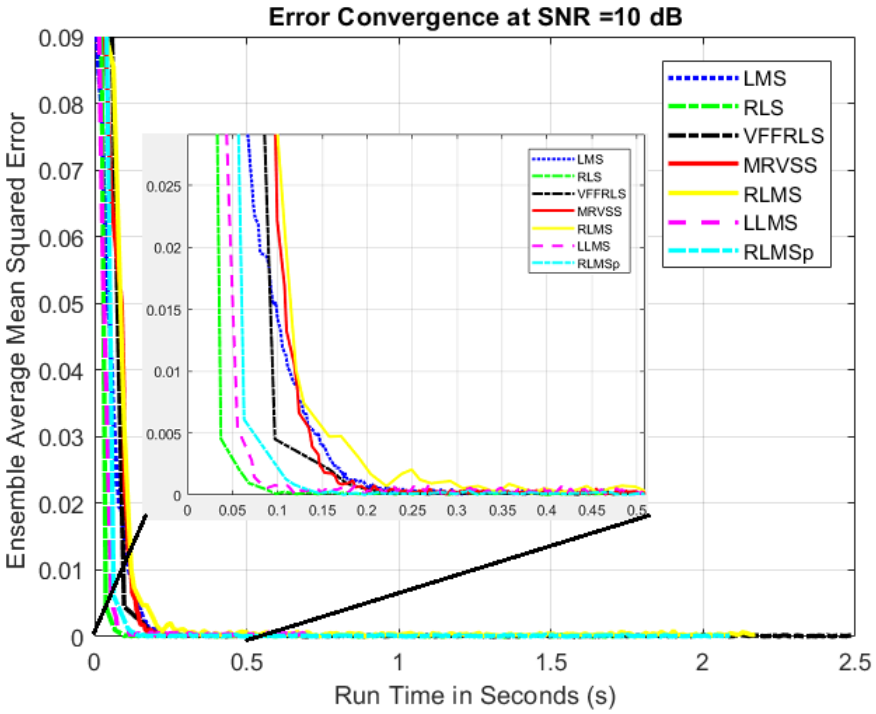

Figure 6.15 – LMS, RLS and Other Variants MSE Convergence vs Time Plot

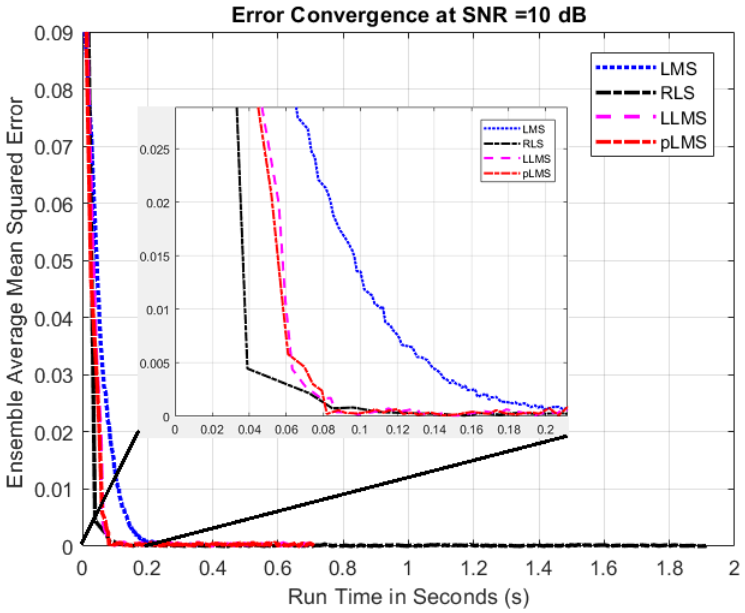

Figure 6.16 – pLMS MSE Convergence Behavior MSE vs Time Plot

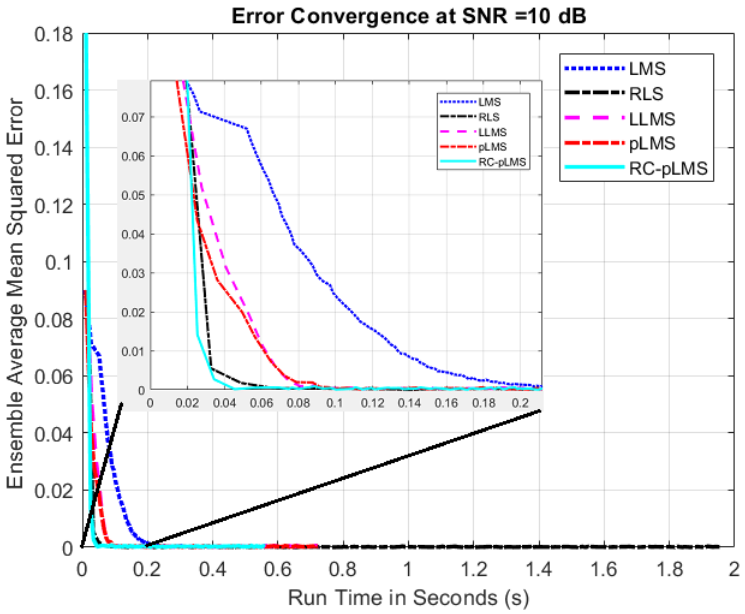

Figure 6.17 – RC-pLMS MSE Convergence Behavior vs Time Plot

AUTHOR'S PUBLICATIONS

Book Chapter:

1. **G. Akkad**, R. Ayoubi, A. Mansour, and B. ElHassan, "Hardware Architecture For a Bit-Serial Odd-Even Transposition Sort Network," *Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES)*, (Pisa, Italy) vol. 627, p. 145, Sept. 2020.

Journal Paper:

1. **G. Akkad**, A. Mansour, B. A. ElHassan, E. Inaty, R. Ayoubi, and J. A. Srar, "A Pipelined Reduced Complexity Two-Stages Parallel LMS Structure for Adaptive Beamforming," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2020.

International Conferences:

1. **G. Akkad**, A. Mansour, B. ElHassan, and E. Inaty, "A Multi-Stage Parallel LMS Structure and its Stability Analysis Using Transfer Function Approximation," in *28th European Signal Processing Conference (EUSIPCO)*, (Amsterdam, Netherlands), August 2020.
2. **G. Akkad**, A. Mansour, B. ElHassan, E. Inaty, and R. Ayoubi, "Two Stages Parallel LMS Structure A Pipelined Hardware Architecture," in *28th European Signal Processing Conference (EUSIPCO)*, (Amsterdam, Netherlands), August 2020.

3. **G. Akkad**, A. Mansour, B. Elhassan, J. Srar, M. Najem, and F. Leroy, "An Efficient Non-Blind Steering Vector Estimation Technique For Robust Adaptive Beamforming With Multistage Error Feedback," in *Intelligent Decision Technologies 2019*, (Malta), June 2019.
4. **G. Akkad**, A. Mansour, B. ElHassan, J. Srar, M. Najem, and F. LeRoy, "Low Complexity Robust Adaptive Beamformer Based On Parallel RLMS and Kalman RLMS," in *27th European Signal Processing Conference (EUSIPCO)*, (A Coruna, Spain), Sep. 2019.
5. **G. Akkad**, R. Ayoubi, and A. Abche, "Constant Time Hardware Architecture for a Gaussian Smoothing Filter," in *2018 International Conference on Signal Processing and Information Security (ICSPIS)*, (Dubai, United Arab Emirates), Nov. 2018.
6. **G. Akkad**, A. Mansour, B. ElHassan, F. L. Roy, and M. Najem, "Twiddle Factor Generation Using Chebyshev Polynomials and HDL for Frequency Domain Beamforming," in *Applications in Electronics Pervading Industry, Environment and Society: APPLEPIES*, (Pisa, Italy), Sep. 2018.
7. **G. Akkad**, A. Mansour, B. ElHassan, F. L. Roy, and M. Najem, "FFT Radix-2 and Radix-4 FPGA Acceleration Techniques Using HLS and HDL for Digital Communication Systems," in *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, (Beirut, Lebanon), Nov. 2018.
8. R. Ayoubi, S. Istambouli, A. Abbas, and **G. Akkad**, "Hardware Architecture For A Shift-Based Parallel Odd-Even Transposition Sorting Network," in *2019 Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, (Beirut, Lebanon), July 2019.

BIBLIOGRAPHY

[1] A. Uncini, *Fundamentals of adaptive signal processing.* Springer, 2015.

[2] S. Haykin, *Adaptive Filter Theory (5th Ed.).* McMaster University, Ontario Canada: Pearson, 2013.

[3] A. Mansour, R. Mesleh, and M. Abaza, "New Challenges In Wireless And Free Space Optical Communications," *Optics and Lasers in Engineering*, vol. 89, pp. 95 – 108, 2017.

[4] G. Akkad, A. Mansour, B. A. ElHassan, E. Inaty, R. Ayoubi, and J. A. Srar, "A Pipelined Reduced Complexity Two-Stages Parallel LMS Structure for Adaptive Beamforming," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2020.

[5] J. A. Srar, K. Chung, and A. Mansour, "Adaptive Array Beamforming Using a Combined LMS-LMS Algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 11, pp. 3545–3557, Nov. 2010.

[6] G. Akkad, A. Mansour, B. ElHassan, J. Srar, M. Najem, and F. LeRoy, "Low Complexity Robust Adaptive Beamformer Based On Parallel RLMS and Kalman RLMS," in *27th European Signal Processing Conference (EUSIPCO)*, (A Coruna, Spain), pp. 1–5, Sep. 2019.

[7] P. W. Howells, "Intermediate Frequency Side-Lobe Canceller," Aug. 24 1965. US Patent 3,202,990.

[8] S. Applebaum, "Adaptive Arrays," *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 585–598, 1976.

[9] S. Applebaum and D. Chapman, "Adaptive Arrays with Main Beam Constraints," *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 650–662, 1976.

[10] B. Widrow, J. McCool, and M. Ball, "The Complex LMS Algorithm," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 719–720, April 1975.

[11] S. A. Vorobyov, "Principles of Minimum Variance Robust Adaptive Beamforming Design," *Signal Processing*, vol. 93, no. 12, pp. 3264–3277, 2013.

[12] S. A. Vorobyov, A. B. Gershman, Z.-Q. Luo, and N. Ma, "Adaptive Beamforming with Joint Robustness Against Mismatched Signal Steering Vector and Interference Nonstationarity," *IEEE Signal Processing Letters*, vol. 11, pp. 108–111, Feb. 2004.

[13] G. Akkad, A. Mansour, B. Elhassan, J. Srar, M. Najem, and F. Leroy, "An Efficient Non-Blind Steering Vector Estimation Technique For Robust Adaptive Beamforming With Multistage Error Feedback," in *Intelligent Decision Technologies 2019*, (Malta), pp. 13–23, June 2019.

[14] G. Akkad, A. Mansour, B. ElHassan, and E. Inaty, "A Multi-Stage Parallel LMS Structure and its Stability Analysis Using Transfer Function Approximation," in *28th European Signal Processing Conference (EUSIPCO)*, (Amsterdam, Netherlands), pp. 1–5, August 2020.

[15] J. A. Srar and K. Chung, "Adaptive Array Beam Forming Using A Combined RLS-LMS Algorithm," in *14th Asia-Pacific Conference on Communications*, (Tokyo, Japan), pp. 1–5, Oct. 2008.

[16] J. A. Srar, K. Chung, and A. Mansour, "LLMS Adaptive Beamforming Algorithm Implemented With Finite Precision," in *20th Telecommunications Forum (TELFOR)*, (Belgrade, Serbia), pp. 303–306, Nov. 2012.

[17] A. Farina and L. Ortenzi, "Effect of ADC and Receiver Saturation On Adaptive Spatial Filtering Of Directional Interference," *Signal processing*, vol. 83, no. 5, pp. 1065–1078, 2003.

[18] C. Caraiscos and Bede Liu, "A Roundoff Error Analysis Of The LMS Adaptive Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 1, pp. 34–41, Feb. 1984.

[19] F. Albu, J. Kadlec, N. Coleman, and A. Fagan, "The Gauss-Seidel Fast Affine Projection Algorithm," in *IEEE Workshop on Signal Processing Systems*, (San Diego, CA, USA), pp. 109–114, Oct. 2002.

[20] F. Albu, J. Kadlec, N. Coleman, and A. Fagan, "Pipelined Implementations Of The a Priori Error-Feedback LSL Algorithm Using Logarithmic Arithmetic," in *IEEE*

International Conference on Acoustics, Speech, and Signal Processing, (Orlando, FL, USA), pp. III–2681–III–2684, May 2002.

[21] N. R. Shanbhag and K. K. Parhi, "Relaxed Look-Ahead Pipelined LMS Adaptive Filters And Their Application To ADPCM Coder," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 12, pp. 753–766, Dec. 1993.

[22] R. K. Sarma, M. T. Khan, R. A. Shaik, and J. Hazarika, "A Novel Time-Shared and LUT-Less Pipelined Architecture for LMS Adaptive Filter," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 1–10, 2019.

[23] W. Zhao, J. Q. Lin, S. C. Chan, and H. K. . So, "A Division-Free and Variable-Regularized LMS-Based Generalized Sidelobe Canceller for Adaptive Beamforming and Its Efficient Hardware Realization," *IEEE Access*, vol. 6, pp. 64470–64485, 2018.

[24] G. Akkad, A. Mansour, B. ElHassan, E. Inaty, and R. Ayoubi, "Two Stages Parallel LMS Structure A Pipelined Hardware Architecture," in *28th European Signal Processing Conference (EUSIPCO)*, (Amesterdame, Netherlands), pp. 1–5, August 2020.

[25] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays (Signals and Communication Technology)*. (Berlin, Heidelberg), Springer-Verlag, 2004.

[26] B. Allen and M. Ghavami, *Adaptive Array Systems: Fundamentals and Applications*. John Wiley & Sons, 2005.

[27] L. W. Couch, H. Shao, X. Li, and L. Liu, *Digital and Analog Communication Systems*, vol. 5. Citeseer, 1997.

[28] M. J. Alexander and M. J. Salter, "The Design of Dipole and Monopole Antennas with Low Uncertainties," *IEEE Transactions on Instrumentation and Measurement*, vol. 46, no. 2, pp. 539–543, 1997.

[29] L. Xing, J. Zhu, Q. Xu, D. Yan, and Y. Zhao, "A Circular Beam-Steering Antenna With Parasitic Water Reflectors," *IEEE Antennas and Wireless Propagation Letters*, vol. 18, no. 10, pp. 2140–2144, Oct. 2019.

[30] P. S. Yedavalli, T. Riihonen, X. Wang, and J. M. Rabaey, "Far-Field RF Wireless Power Transfer With Blind Adaptive Beamforming For Internet Of Things Devices," *IEEE Access*, vol. 5, pp. 1743–1752, 2017.

[31] M. Skolnik, "Resolution of Angular Ambiguities in Radar Array Antennas with Widely-Spaced Elements and Grating Lobes," *IRE Transactions on Antennas and Propagation*, vol. 10, no. 3, pp. 351–352, 1962.

[32] P. Ioannides and C. A. Balanis, "Uniform Circular Arrays For Smart Antennas," *IEEE Antennas and propagation magazine*, vol. 47, no. 4, pp. 192–206, 2005.

[33] Srar, Jalal Abdulsayed, "Adaptive Antenna Array Beamforming Using a Concatenation of Recursive Least Square and Least Mean Square Algorithms," 2011.

[34] W. Shang, Z. Dou, W. Xue, and Y. Li, "Digital Beamforming Based On FPGA For Phased Array Radar," in *Progress In Electromagnetics Research Symposium-Spring (PIERS)*, (St. Petersburg, Russia), pp. 437–440, May 2017.

[35] V. Seneviratne, A. Madanayake, and L. T. Bruton, "Multidimensional-DSP Beamformers Using the ROACH-2 FPGA Platform," *Electronics*, vol. 6, no. 3, p. 49, 2017.

[36] Q. Jing, Y. Li, and J. Tong, "Performance Analysis Of Multi-Rate Signal Processing Digital Filters On FPGA," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 31, 2019.

[37] R. Badau, G. Richard and B. David, "Fast adaptive esprit algorithm," *IEEE/SP 13th Workshop on Statistical Signal Processing*, (Bordeaux, France), pp. 289–294, July. 2005.

[38] R. Badau, G. Richard and B. David, "Adaptive ESPRIT algorithm based on the PAST subspace tracker," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Hong Kong, China), pp. VI-229, April. 2003.

[39] I. Kacha, K. Abed-Meraim and A.. Belouchrani, "A fast adaptive blind equalization algorithm robust to channel order over-estimation errors," *Sensor Array and Multichannel Signal*, (Barcelona, Spain), pp. 148-152, July. 2004.

[40] Y. Zhang, J. Zhao, J. Li and Y. Sun, "A fast convergent algorithm for joint blind equalization and carrier recovery," *9th International Conference on Signal Processing)*, (Beijing, China), pp. 1784-1787, Oct. 2008.

[41] N. Xie, Y. Zhou, M. Xia and W. Tang, "Fast Blind Adaptive Beamforming Algorithm With Interference Suppression," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 3, pp. 1985-1988, May. 2008.

[42] J. Laurila, K. Kopsa, R. Schurhuber, and E. Bonek, "Semi-Blind Separation and Detection of Co-Channel Signals," in *1999 IEEE International Conference on Communications*, (Vancouver, BC, Canada), pp. 17–22, June 1999.

[43] A. El-Keyi, T. Kirubarajan, and A. B. Gershman, "Robust Adaptive Beamforming Based on the Kalman Filter," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 3032–3041, Aug. 2005.

[44] D. P. Mandic, S. Kanna, and A. G. Constantinides, "On The Intrinsic Relationship Between The Least Mean Square And Kalman Filters [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 117–122, Nov. 2015.

[45] J. Kim and A. D. Poularikas, "Performance Analysis Of The Adjusted Step Size NLMS Algorithm," in *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*, (Atlanta, GA, USA), pp. 467–471, Mar. 2004.

[46] P. Bouboulis and S. Theodoridis, "Extension of Wirtinger's Calculus To Reproducing Kernel Hilbert Spaces And The Complex Kernel LMS," *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 964–978, March 2011.

[47] B. D. Van Veen and K. M. Buckley, "Beamforming: A Versatile Approach To Spatial Filtering," *IEEE Acoustics, Speech, and Signal Processing (ASSP) magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[48] V. H. Nascimento, "The Normalized LMS Algorithm With Dependent Noise," *Proc. Anais do 19 Simpòsio Brasileiro de Telecomunicações*, (Fortaleza, Brazil), Sep. 2001.

[49] J. Gorriz, J. Ramirez, S. Cruces-Alvarez, D. Erdogmus, C. Puntonet, and E. Lang, "Speech Enhancement in Discontinuous Transmission Systems Using the Constrained-Stability Least Mean Squares Algorithm," *The Journal of the Acoustical Society of America*, vol. 124, no. 6, pp. 3669–3683, 2008.

[50] Z. Xiubing *et al.*, "A New Modified Robust Variable Step Size LMS Algorithm," in *4th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, (Xian, China), pp. 2699–2703, May 2009.

[51] Z. Shengkui, M. Zhihong, and K. Suiyang, "A Fast Variable Step-Size LMS Algorithm with System Identification," in *2nd IEEE Conference on Industrial Electronics and Applications (ICIEA)*, (Harbin, China), pp. 2340–2345, May 2007.

[52] R. H. Kwong and E. W. Johnston, "A Variable Step Size LMS Algorithm," *IEEE Transactions on signal processing*, vol. 40, no. 7, pp. 1633–1642, 1992.

[53] T. Aboulnasr and K. Mayyas, "A Robust Variable Step-Size LMS-Type Algorithm: Analysis and Simulations," *IEEE Transactions on signal processing*, vol. 45, no. 3, pp. 631–639, 1997.

[54] E. M. Lobato, O. J. Tobias, and R. Seara, "Stochastic Modeling of the Transform-Domain εLMS Algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1840–1852, May 2008.

[55] J. A. Srar, K. Chung, and A. Mansour, "A New LLMS Algorithm for Antenna Array Beamforming," in *2010 IEEE Wireless Comm and Networking Conference*, (Sydney, NSW, Australia), pp. 1–5, April 2010.

[56] S. Lee, J.-s. Lim, and K. Sung, "A Low-Complexity AFF-RLS Algorithm Using a Normalization Technique," *IEICE Electronic Express*, vol. 6, pp. 1774–1780, 12 2009.

[57] A. Zoubir, M. Viberg, R. Chellappa, and S. Theodoridis, *Academic Press Library in Signal Processing Volume 3: Array and Statistical Signal Processing*. Elsevier, 2013.

[58] S. R. Theodore *et al.*, "Wireless Communications: Principles and Practice," in *Upper Saddle River*, p. 69, Prentice Hall, 2002.

[59] G. Akkad *et al.*, "Twiddle Factor Generation Using Chebyshev Polynomials and HDL for Frequency Domain Beamforming," in *Applications in Electronics Pervading Industry, Environment and Society, Springer Lecture Notes in Electrical Engineering (APPLEPIES)*, (Pisa, Italy), Sept. 2018.

[60] P. Clarkson and P. White, "Simplified Analysis Of The LMS Adaptive Filter Using A Transfer Function Approximation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 7, pp. 987–993, July 1987.

[61] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, *FPGA-Based Implementation of Signal Processing Systems*. Wiley Online Library, 2017.

[62] R. Tessier and W. Burleson, "Reconfigurable Computing for Digital Signal Processing: A Survey," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 28, no. 1-2, pp. 7–27, 2001.

[63] D. Burgos, J. Kunzler, R. Lemos, and H. Silva, "Adaptive Beamforming For Moving Targets Using Genetic Algorithms," in *2015 Workshop on Engineering Applications - International Congress on Engineering (WEA)*, (Bogota, Colombia), pp. 1–5, Oct. 2015.

[64] A. Boschmann, A. Agne, L. Witschen, G. Thombansen, F. Kraus, and M. Platzner, "Fpga-Based Acceleration of High Density Myoelectric Signal Processing," in *International Conference on ReConFigurable Computing and FPGAs*, (Mexico City, Mexico), pp. 1–8, Dec. 2015.

[65] G. Akkad, R. Ayoubi, and A. Abche, "Constant Time Hardware Architecture for a Gaussian Smoothing Filter," in *International Conference on Signal Processing and Information Security (ICSPIS)*, (Dubai, United Arab Emirates), pp. 1–4, Nov. 2018.

[66] G. Akkad, M. ElHassan, and R. Ayoubi, "FPGA Hardware Architecture for Stereoscopic Image Compression Based on Block Matching, Watermarking and Hamming Code," in *2016 International Image Processing, Applications and Systems (IPAS)*, (Hammamet, Tunisia), pp. 1–5, Nov. 2016.

[67] A. Mkhinini, P. Maistri, R. Leveugle, and R. Tourki, "HLS Design of a Hardware Accelerator for Homomorphic Encryption," in *IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, (Dresden, Germany), pp. 178–183, April 2017.

[68] V. Migliore, M. M. Real, V. Lapotre, A. Tisserand, C. Fontaine, and G. Gogniat, "Fast Polynomial Arithmetic for Somewhat Homomorphic Encryption Operations in Hardware with Karatsuba Algorithm," in *International Conference on Field-Programmable Technology (FPT)*, (Xi'an, China), pp. 209–212, Dec. 2016.

[69] G. Akkad, R. Ayoubi, A. Mansour, and B. ElHassan, "Hardware Architecture for a Bit-Serial Odd-Even Transposition Sort Network," *Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES)*, (Pisa, Italy) vol. 627, p. 145, Sept. 2020.

[70] R. Ayoubi, S. Istambouli, A. Abbas, and G. Akkad, "Hardware Architecture For A Shift-Based Parallel Odd-Even Transposition Sorting Network," in *Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, (Beirut, Lebanon), pp. 1–6, July 2019.

[71] L. C. des telecoms, "Antenne PB1 Sous RADOME." France, Accessed on 8/10/2020 From https://www.cite-telecoms.com/antenne-pb1-sous-radome/

[72] L. de Gouyon Matignon, "Pleumeur-Bodou and the French CNET." Accessed on 8/10/2020 From https://www.spacelegalissues.com/pleumeur-bodou-and-the-french-cnet/

[73] R. Mueller, J. Teubner, and G. Alonso, "Data Processing on FPGAs," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 910–921, 2009.

[74] Chu, Pong P, *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version.* John Wiley & Sons, 2011.

[75] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, *et al.*, "A Survey and Evaluation of FPGA High-Level Synthesis Tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, 2015.

[76] Altera, "Enabling High-Performance DSP Applications with Arria V or Cyclone V Variable-Precision DSP Blocks," Accessed on 4/19/2020, From Intel FPGA https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01159-arriav-cyclonev-dsp.pdf

[77] G. Akkad, A. Mansour, B. ElHassan, F. L. Roy, and M. Najem, "FFT Radix-2 and Radix-4 FPGA Acceleration Techniques Using HLS and HDL for Digital Communication Systems," in *IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, (Beirut, Lebanon), pp. 1–5, Nov. 2018.

[78] M. Garrido, "A New Representation of FFT Algorithms Using Triangular Matrices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 10, pp. 1737–1745, 2016.

[79] C.-P. Hung, S.-G. Chen, and K.-L. Chen, "Design of an Efficient Variable-Length FFT Processor," in *IEEE International Symposium on Circuits and Systems*, (Vancouver, BC, Canada),vol. 2, pp. II–833, May 2004.

[80] R. H. Neuenfeld, M. B. Fonseca, E. A. da Costa, and J. P. Oses, "Exploiting Addition Schemes for the Improvement of Optimized Radix-2 and Radix-4 FFT Butterflies," in *IEEE 8th Latin American Symposium on Circuits & Systems (LASCAS)*, (Bariloche, Argentina), pp. 1–4, Feb. 2017.

[81] C. Yu, K.-H. Lee, and C.-F. Kuo, "Low-Complexity Twiddle Factor Generator for FFT Processors," in *IEEE International Conference on Consumer Electronics (ICCE)*, (Las Vegas, NV, USA), pp. 350–351, Jan. 2017.

[82] J. Zhou, "A New Method to Generate Twiddle Factor Using CORDIC Based Radix-4 FFT Butterfly," in *International Conference on Communications, Circuits and Systems (ICCCAS)*, (Chengdu, China), vol. 2, pp. 505–508, Nov. 2013.

[83] S. N. Shinde, "Twiddle Factor Generation Using CORDIC Processor for Fingerprint Application," in *International Conference on Computer, Communication and Control (IC4)*, (Indore, India), pp. 1–7, Sep. 2015.

[84] J. Han, J. R. Zeidler, and W. H. Ku, "Quality of Approximation in Error Transfer Function Analysis of the LMS Adaptive Filters," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers*, (Pacific Grove, CA, USA), vol. 1, pp. 586–590 Vol.1, Nov. 2003.

[85] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the Unit Delay FIR all Pass Filters Design," *IEEE Signal Processing Magazine*, vol. 13, pp. 30–60, Jan. 1996.

[86] O. Macchi and E. Eweda, "Second-Order Convergence Analysis Of Stochastic Adaptive Linear Filtering," *IEEE Transactions on Automatic Control*, vol. 28, no. 1, pp. 76–85, January 1983.

[87] J. A. Srar, K. Chung, and A. Mansour, "Performance of an LLMS Beamformer in the Presence of Element Gain and Spacing Variations," in *The 17th Asia Pacific Conference on Communications*, (Sabah, Malaysia), pp. 593–598, Oct. 2011.

[88] A. M. Nor and E. M. Mohamed, "Millimeter Wave Beamforming Training Based on Li-Fi Localization in Indoor Environment," in *IEEE Global Communications Conference*, (Singapore, Singapore), pp. 1–6, Dec. 2017.

[89] Z. Zhang *et al.*, "6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.

Mot clés : LMS, formation de faisceaux adaptative, FPGA, réseau de capteurs, RC-pLMS.

Résumé : Pour améliorer l'efficacité du spectre, la formation adaptative de faisceaux devient une caractéristique inévitable pour les antennes intelligentes. Les systèmes embarqués de communication sans fil imposent des contraintes difficiles liées à l'implémentation en parallèle et en pipeline avec des ressources limitées. Certaines variantes des algorithmes accélérent la convergence tout en maintenant une faible erreur résiduelle. D'autres présentent des architectures de pipeline parallèle. Donc, les algorithmes actuels profitent d'une convergence améliorée, au prix d'une augmentation de la complexité, ou bien une architecture matérielle de pipeline sans aucune amélioration significative. Pour présenter une solution unifiée, nous proposons un algorithme en deux étapes, appelée structure parallèle des moindres carrés moyens (pLMS). Une conception de pLMS à complexité réduite (RCpLMS) a été aussi développée. Afin de concevoir une architecture matérielle en pipeline, nous avons appliqué la technique de relaxation de de somme en retard (DRC-pLMS). Une étudedes performances sur différents plateformes et architectures a été menée. Les simulations démontrent les performances exceptionnelles du RC-pLMS. DRC-pLMS fonctionne à une fréquence maximale de 208,33 MHz avec une légère augmentation des ressources par rapport à LMS.

Keywords: LMS, Adaptive Beamforming, FPGA, Sensor Array, RC-pLMS.

Abstract: Ever since its inception, adaptive beamforming has become an inevitable feature in smart antenna array to improve the spectrum efficiency. However, modern embedded wireless communication systems have imposed challenging constraints on adaptive algorithms when targeting a parallel and pipelined implementation on limited resource devices, like Field Programmable Gate Array (FPGA). Such constraints include reduced complexity, parallelism, accelerated convergence and low residual error. Several variants of classical adaptive beamformers were proposed to accelerate the convergence while maintaining a low error floor. Other suggestions focused on a parallel, pipeline architecture. The resulting beamforming algorithms either presented an improved convergence profile, at the cost of an increase of complexity or presented a pipeline hardware architecture without any significant improvement. To present a unified solution with superior convergence profile while maintaining a low complexity parallel pipeline architecture, we propose a two-stages algorithm, called parallel least mean square structure (pLMS). pLMS is further simplified to obtain the reduced complexity pLMS design (RC-pLMS). In order to design a pipelined hardware architecture, we applied the delay and sum relaxation technique (DRC-pLMS). A study on the behavior and the performance of different hardware design tools and processor architectures is conducted. Computer simulations demonstrated the outstanding performance of RC-pLMS. The DRC-pLMS can operate at a maximum frequency of 208.33 MHz with a minor increase in resource usage compared to LMS.