VLSI Implementation of Digital Signal Processing Algorithms for MIMO Detection and Channel Pre-Processing

by

Dimpesh Patel

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

VLSI Implementation of Digital Signal Processing Algorithms for MIMO Detection and Channel Pre-Processing

Dimpesh Patel

Master of Applied Science, 2010

Graduate Department of Electrical and Computer Engineering

University of Toronto

Abstract

The efficient high-throughput VLSI implementation of Soft-output MIMO detectors for high-order constellations and large antenna configurations has been a major challenge in the literature. This thesis introduces a novel Soft-output K-Best scheme that improves BER performance and reduces the computational complexity significantly by using three major improvement ideas. It also presents an area and power efficient VLSI implementation of a 4×4 64-QAM Soft K-Best MIMO detector that attains the highest detection throughput of 2 Gbps and second lowest energy/bit reported in the literature, fulfilling the aggressive requirements of emerging 4G standards such as IEEE 802.16m and LTE-Advanced. A low-complexity and highly parallel algorithm for QR Decomposition, an essential channel pre-processing task, is also developed that uses 2D, Householder 3D and 4D Givens Rotations. Test results for the QRD chip, fabricated in 0.13μm CMOS, show that it attains the lowest reported latency of 144ns and highest QR Processing Efficiency.

Acknowledgments

This thesis bears my name as the sole author, yet as any endeavor that spans the course of several years, it would have been impossible for me to complete without the help and encouragement of numerous people. First and foremost, I would like to express my most sincere gratitude towards my supervisor Professor P. G. Gulak, for being a role model through his relentless work ethic, skillful administration, insightful teaching methods, intelligent approach to research and boundless enthusiasm.

I thank the members of my M.A.Sc defense committee, Prof. Paul Chow, Prof. T. J. Lim and Prof. L. Qian for their time and insightful suggestions.

I would also like to gratefully acknowledge the financial support provided by University of Toronto, Natural Sciences and Engineering Research Council of Canada (NSERC) and CMC Microsystems.

I thank Jaro Pristupa for solving CAD-related problems with speed and skill. Also, special thanks to Jeetendar Narsinghani (CMC), Kentaro Yamamoto and Mahdi Shabany for their help in ASIC design and testing processes.

I feel blessed for getting to know so many good friends during my studies at the University of Toronto. I have learned a lot from them and I am grateful to all of them. Special thanks to Krunal Patel and his wife for being intimate, supportive and wonderful friends. Many thanks to friends from BA5000, BA5158, Glenn's group and those from outside the department. In particular, I would like to thank Ameer Youssef, Soroush Tabatabaei, Nasim Nikkhoo, Meysam Zargham, Kevin Banovic, Saeed Moradi, Vadim Smolyakov, Mayukh Roy, Safeen Huda, Alireza Nilchi, Amir Parayandeh, Mike Bichan, Dustin Dunwell, David Halupka, Oleksiy Tyshchenko, Siamak Sarvari, William Song, Hemesh Yasotharan, Shariar Shahramian, Behrooz Abiri, Farsheed Mahmoudi and Roya Doostnejad.

I am grateful beyond measure to my parents, my grandma and my brother, Nishant, for their love, encouragement and continuous support. Without their support and sacrifices my dreams would have remained dreams.

Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{y}$	Real received symbol vector
$\mathbf{s}$	Real transmitted symbol vector
$\widetilde{\mathbf{s}}$	Complex transmitted symbol vector
$\mathbf{H}$	Real MIMO channel matrix
$\mathbf{v}$	Real noise vector
$\mathbf{Q}$	Unitary matrix
$\mathbf{R}$	Upper triangular matrix with real entries
$\mathbf{z}$	Post-processed real received symbol vector
$\widetilde{\mathbf{y}}$	Complex received symbol vector
$\widetilde{\mathbf{s}}$	Complex transmitted symbol vector
$\widetilde{\mathbf{H}}$	Complex MIMO channel matrix
$\widetilde{\mathbf{v}}$	Complex noise vector
N_R	Number of receive antenna
N_T	Number of transmit antenna
$\mathbf{x}(n)$	Transmitted bit vector at time n
$\widehat{\mathbf{x}}$	Estimated version of the transmitted vector
x_k	The k-th bit of $\mathbf{x}$
$S_k^{(1)}$	Set of all vectors $\mathbf{x}$ with bit position x_k being "1"
$S_k^{(0)}$	Set of all vectors $\mathbf{x}$ with bit position x_k being "0"
$\mathcal{O}$	Complex constellation
Q	Constellation size/ordinality
M_c	Number of bits per constellation point
R	Number of bits per channel use
σ^2	Noise variance
$\mathcal{N}_c$	Complex Gaussian distribution
$\Re\{\cdot\}$	Real part of a complex number
$\Im\{\cdot\}$	Imaginary part of a complex number

Ω	Set of possible real entries in $\mathcal{O}$
K	Number of K-Best candidates in each level of the tree
$T_l(s^{(l)})$	Accumulated partial Euclidean distance in level l
$e_l(s^{(l)})$	Distance increment between two successive nodes in level l
$\mathcal{K}_l$	List of K-Best children in level l
$\mathcal{C}_l$	The set of all the current best child of all parents
$\mathcal{D}_l$	PED values of the elements of $\mathcal{C}_l$
h_l	l-th column of channel matrix $\mathbf{H}$
$\widehat{s}_i$	i-th estimated symbol at the receiver
r	Sphere constraint in SD
r_{lj}	An entry of matrix $\mathbf{R}$
$\bar{r}_{lj}$	The scaled version of r_{lj} by r_{ll}
$s_l^{[k]}$	k-th best child of a parent in level l
$\mathcal{L}$	All visited points, which have not been announced as the next best sibling

List of Acronyms

A/D Analog-to-Digital

ASIC Application-Specific Integrated Circuit

AWGN Additive White Gaussian Noise

BER Bit-Error-Rate

BLAST Bell Labs Layered Space-Time

bpcu Bits per Channel Use

CDMA Code Division Multiple Access

CMOS Complementary Metal Oxide Semiconductor

CTC Convolutional Turbo Coding

D/A Digital-to-Analog

DSP Digital Signal Processor

ECC Error Correcting Codes

FC First Child

FEC Forward Error Correction

FFT Fast Fourier Transform

HSDPA High-Speed Downlink Packet Access

HVT High Vt (transistor threshold voltage)

LLR Log-Likelihood Ratio

LMMSE Least Minimum Mean Squared Error

LR Lattice Reduction

LTE Long Term Evolution

LVT Low Vt (transistor threshold voltage)

Mbps Mega bits per second

MIMO Multiple-Input Multiple-Output

MKSE Modified K-Best Schnorr-Euchner

ML Maximum-Likelihood

MUX Multiplexer

NC Next Child

OFDM Orthogonal Frequency-Division Multiplexing

PAPR Peak-to-Average-Power-Ratio

PE Processing Element

PED Partial Euclidean Distance

PSK Phase Shift Keying

PVT Process Voltage Temperature

QAM Quadrature Amplitude Modulation

QRD QR Decomposition

QoS Quality-of-Service

RVD Real-Valued Decomposition

S/P Serial to Parallel Conversion (Demux)

SD Sphere Decoding

SE Schnorr-Euchner

SER Symbol Error Rate

SINR Signal-to-Interference-and-Noise Ratio

SISO Single-Input Single-Output

SM Spatial Multiplexing

SNR Signal-to-Noise-Ratio

SPE Soft-Output Processing Element

SVT Standard Vt (transistor threshold voltage)

VLSI Very Large Scale Integration

WLAN Wireless Local Area Network

WMAN Wireless Metropolitan Area Network

WiMAX Worldwide Interoperability for Microwave Access

ZF Zero-Forcing

1 Introduction

1.1 Introduction to MIMO Systems

The use of multiple transmit and receive antennas in wireless systems, known as Multiple-Input Multiple-Output (MIMO) systems [2], offers powerful performance enhancing capabilities and is one of the most interesting and promising areas of recent innovations in wireless communications. MIMO technology offers a number of benefits that help tackle the challenges posed by both the impairments in the wireless channel, as well as the resource constraints [3]. MIMO systems also exploit the spatial dimension, in addition to the time and frequency dimensions, by using multiple antennas at the transmitter and the receiver, which leads to high spectral efficiency [4].

This high spectral efficiency offered by MIMO technology has made it the technology of choice in many wireless standards. For example, in the Wireless Local Area Network (WLAN) IEEE 802.11n standard, MIMO systems are used to achieve data rates of up to 480 Mbps. MIMO systems are also used for the IEEE 802.16e Wireless Metropolitan Area Network (WMAN) system, also known as Worldwide Interoperability for Microwave Access (WiMAX) ([5], [6], [7]). They are also envisioned to be used in the the next generation WiMAX systems, the IEEE 802.16m standard, that require high-mobility MIMO systems with peak data rates of up to 1 Gbps. Furthermore, the advancement of the Long Term Evolution (LTE) project, the Evolved Universal Terrestrial Radio Access (E-UTRA) standard, also known as the LTE-Advanced standard, plans to utilize MIMO systems with large antenna configurations (4×4 and 8×8) and high data rates (up to 1 Gbps for downlink and 500 Mbps for uplink).

MIMO systems employ multiple antennas at both the transmitter and receiver sides to increase spectral efficiency and to meet the aggressive requirements of these standards. The number of transmit and receive antennas will be denoted by N_T and N_R, respectively. This high spectral efficiency leads to an increase in the overall system reliability, achievable data rate, system capacity, coverage area and a decrease in the required transmit power [3]. However, these desirable attributes compete with one another and maximizing each of

them requires different transmission schemes. The transmission schemes exploit the high spectral efficiency, offered by MIMO systems, by leveraging three types of gains [8]:

- **Array gain** refers to picking up a larger share of the transmitted power at the receiver, which also increases the receive SNR. Array gain improves resistance to noise and helps to suppress interference, thereby improving the coverage and extending the range of wireless communication systems [3]. Beamforming [9], that uses antenna arrays to focus energy, can be used to maximize array gain.

- **Spatial Diversity gain** mitigates the effects of fading and is realized by providing the receiver with multiple (ideally independent) copies of the transmitted signal in space, frequency or time [3]. The diversity order corresponds to the slope of the bit-error-rate (BER) curve and is directly related to the number of independent received signal copies. Space-time coding [10] is used to exploit diversity gain, which increases link-reliability and Quality-of-Service (QoS).

- **Spatial Multiplexing gain** allows for a linear increase in spectral efficiency and peak data rates by transmitting multiple, independent data streams concurrently within the bandwidth of operation. The Spatial Multiplexing (SM) transmission scheme uses the N_T transmit antennas to achieve high peak data rates and hence to increase the capacity of the wireless network.

In this thesis, we focus on MIMO systems that employ the Spatial Multiplexing (SM) transmission scheme. As mentioned, a Spatial Multiplexing gain can be realized by transmitting N_T independent data streams. Thus, Spatial Multiplexing can be used to increase transmission data rates by up to a factor of N_T, compared to single antenna communication systems. However, the multiple transmitted data streams interfere with each other at the receiver, and in order to reliably separate these received data streams, we require MIMO receivers with $N_R \geq N_T$ [3].

However, these significant performance improvements offered by MIMO systems come at the expense of considerably more complex signal processing at the receiver. Hence, one of the major challenges in MIMO systems is to design low-complexity receiver algorithms and develop their efficient Very Large Scale Integration (VLSI) implementations. In terms of complexity, one of the most challenging implementation tasks in MIMO receivers is the realization of the MIMO detectors. In the Spatial Multiplexing mode, the task of a MIMO detector is to separate the spatially multiplexed data streams at the receiver and

provide estimates for transmitted symbols within each data stream. Also, in practice, most wireless communication systems use Error Correcting Codes (ECC), demanding a posteriori probability (APP) information about each bit for iterative Forward Error Correction (FEC) purposes. The ECC coded MIMO systems with Soft-Output MIMO detectors generally offer a much superior bit error rate (BER) performance compared to the uncoded MIMO systems with Hard-Output MIMO detectors. Therefore, Soft-Output MIMO signal detection is highly desirable.

1.2 Motivations and Thesis Objectives

The emerging 4G wireless standards, such as IEEE 802.16e/m (WiMAX), IEEE 802.11n and LTE-Advanced, require MIMO systems with high data rates (up to 1Gbps), large constellation orders (64-QAM and 256-QAM) and large antenna configurations (4×4 and 8×8). Hence, for an ECC coded MIMO system with worst case rate of 1/2, these standards require Soft-Output MIMO detectors with detection throughputs of up to 2Gbps. However, for large constellations and large antenna configurations, existing Soft-Output MIMO detection schemes lead to a very large computational complexity, for the purpose of computing APP values for the transmitted bits. As a result, the VLSI implementation of these MIMO detectors consumes large silicon area and power, as well as achieve only medium-rate data throughput rates that falls short of most aggressive next-generation 4G applications. Hence, an area and power efficient high-throughput VLSI implementation of a Soft-Output MIMO detector, with close to ML performance, for high-order quadrature amplitude modulation (QAM) schemes still poses many challenges.

One of the objectives of this thesis is to develop a low-complexity Soft-Output MIMO detection scheme that allows efficient computation of APP values for the transmitted bits for high-order constellations and large antenna configurations. The proposed scheme should reduce the total number of operations required and should not cause significant degradation in the BER performance compared to the optimal ML detection. Another objective of this thesis is to develop an efficient VLSI implementation of a Soft-Output MIMO detector that offers high detection throughput and minimizes gate count and power consumption, such that it can fulfill the aggressive requirements imposed by the emerging 4G wireless standards. In particular, this thesis will focus on designing a Soft-Output 4×4 64-QAM MIMO detector, with sustained detection throughput of up to 2 Gbps.

Several types of channel processing operations run in parallel with MIMO detection. One

of these is the QR Decomposition (QRD) of the estimated channel characteristic matrix. QRD is required by most MIMO detection schemes to decompose the channel matrix **H** into a unitary matrix **Q** and an upper triangular matrix **R**, providing a suitable framework for sequential detection schemes. However, the implementation details for QRD are often overlooked. As mentioned earlier, the emerging 4G wireless standards require MIMO systems with high data rates, high mobility and large antenna configurations. Furthermore, the high-mobility applications involve dynamic and fast-varying channel environments, which require QR Decomposition to be performed very frequently. However, for decomposition of large complex channel matrices, state-of-the-art QRD implementations have high computational complexity and present a throughput bottleneck due to the large number of sequential operations required. This in turn leads to either large QRD Processing Latency (defined as the number of clock cycles required to produce a new set of output **Q** and **R** matrices) or to large area and power requirements. Hence, QRD implementations are required for decomposing large complex channel matrices, while minimizing QRD processing latency, silicon area and power consumption requirements.

Hence, the second objective of this thesis is to develop a QR Decomposition scheme that allows low-complexity decomposition of large complex matrices, by reducing the number of computations required and by increasing their execution parallelism, to resolve the throughput bottleneck. Another objective of this thesis is to utilize the proposed QRD scheme to develop an efficient architecture for the QR Decomposition core that offers low hardware complexity and power consumption requirements. The proposed QRD architecture should also minimize the QRD Processing Latency, such that it can be used in high-mobility applications, envisioned in the new 4G wireless standards, that involve dynamic and fast-varying channel environments. In particular, this thesis will focus on designing a QRD core for 4×4 MIMO systems with quasi-static channel model that is updated every four consecutive channel uses.

1.3 Thesis Outline

The outline of the thesis is as follows. Chapter 2 discusses the overall MIMO system model and the simulation framework used for all functional simulations within the thesis. It also provides background information on the various MIMO detection schemes and the basic QR Decomposition algorithms, along with their performance and complexity characteristics. Chapter 3 describes the proposed Soft-Output K-Best MIMO detection scheme and

compares its complexity and performance with the existing Soft-Output K-Best detection schemes. Chapter 4 presents the architecture for VLSI implementation of the Soft K-Best MIMO detector and provides details about the functionality and architecture of each of the sub-blocks. Chapter 5 describes the proposed hybrid QR Decomposition scheme, as well as describes the multi-dimensional CORDIC algorithms that form the basis for the proposed QRD scheme. Chapter 5 also addresses the VLSI implementation aspects of the proposed QRD scheme and reports the ASIC implementation and the test results for the fabricated design, as well as compares them with the state-of-the-art QRD implementations. Finally, Chapter 6 concludes the thesis and provides potential directions for future work.

2 Fundamentals of MIMO Detection and Pre-Processing

This chapter is divided in two major parts. The first part, Sections 2.1 and 2.2, describe the MIMO system under consideration, the simulation framework used for all MATLAB simulations, as well as discusses the notation and terminology that will be used throughout the thesis. The second part, Sections 2.3 and 2.4, provides background information on various MIMO detection schemes and the basic QR Decomposition algorithms, along with their performance and complexity characteristics.

2.1 MIMO System Model

Let us consider a MIMO system shown in Fig. 2.1, where the number of transmit antennas is denoted by N_T and the number of receive antennas is denoted by N_R. In this thesis, we always assume that $N_R \geq N_T$. At time n, the bit sequence $\mathbf{x}(n) = \left[x_1(n), \ldots, x_{M_c N_T}(n)\right]^T$ is sent to N_T parallel streams using a serial-to-parallel (S/P) block, which are mapped into a complex vector $\widetilde{\mathbf{s}}(n) = \left[\widetilde{s}_1(n), \ldots, \widetilde{s}_{N_T}(n)\right]^T$ by N_T linear modulators at the transmitter front end[1]. Each element $\widetilde{s}_i(n)$ is taken from a complex constellation $\mathcal{O}$ (such as rectangular Quadrature Amplitude Modulation (QAM)) composed of $Q = |\mathcal{O}| = 2^{M_c}$ distinct points. This means that every set of M_c consecutive bits is mapped to a complex QAM constellation point. In fact, this implies that $\widetilde{\mathbf{s}} \in \mathcal{O}^{N_T}$, where the index n is removed hereafter for brevity. The transmission rate of the corresponding MIMO system, with N_T transmit antennas in spatial multiplexing (SM) mode is then given by $R = N_T \log_2 Q = N_T M_c$ bits per channel use (bpcu). For a fair comparison, which is independent of the number of transmit antennas and of the modulation scheme, the signal vector $\widetilde{\mathbf{s}}$ is normalized before transmission in such a way that the average transmitted power is one (i.e., $E\{\| \widetilde{\mathbf{s}} \|^2\}$=1).

[1] In this thesis, complex variables are distinguished from real variables by a "$\sim$" sign. Moreover, matrices and vectors are distinguished from scalars by using a bold font. For instance, the complex channel matrix is referred to by $\widetilde{\mathbf{H}}$ whereas the real channel matrix is denoted by $\mathbf{H}$.

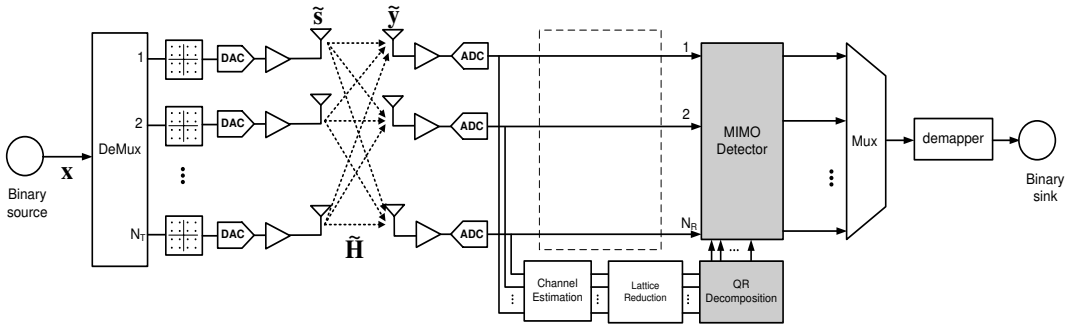

Figure 2.1: The MIMO system model under consideration.

The complex baseband equivalent model of the MIMO wireless channel, that yields the N_R-dimensional received vector $\widetilde{\mathbf{y}} = \left[\widetilde{y}_1, \ldots, \widetilde{y}_{N_R}\right]^T$ is given by the following equation:

$$\widetilde{\mathbf{y}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{s}} + \widetilde{\mathbf{v}}, \tag{2.1}$$

where $\widetilde{\mathbf{H}} = \{H_{ij}\}_{i=1}^{N_R}{}_{j=1}^{N_T}$ denotes a $N_R \times N_T$ dimensional channel characteristic matrix, representing the complex-valued channel gains between each transmit and each receive antenna. Also, $\widetilde{\mathbf{v}} = \left[\widetilde{v}_1, \ldots, \widetilde{v}_{N_R}\right]^T$ represents the N_R dimensional independent identically distributed (i.i.d) circularly symmetric complex zero-mean Additive White Gaussian Noise (AWGN) thermal noise vector with variance σ^2 per complex dimension, i.e., $\widetilde{v}_i \sim \mathcal{N}_c(0, \sigma^2)$. Furthermore, the entries of $\widetilde{\mathbf{H}}$ are chosen independently as zero-mean complex Gaussian random variables with variance one per complex dimension. The signal-to-noise-ratio (SNR) is defined as the ratio between the total transmitted power, which is normalized to one, and the variance of the thermal noise, i.e., SNR$= 1/\sigma^2$.

The task of the MIMO detector at the MIMO receiver is to obtain the best possible estimate of the transmitted signal vector $\widetilde{\mathbf{s}}$ in the Euclidean sense based on the received vector $\widetilde{\mathbf{y}}$. i.e.,

$$\widehat{\widetilde{\mathbf{s}}} = \underset{\widetilde{\mathbf{s}} \in \mathcal{O}^{N_T}}{\arg\min} \parallel \widetilde{\mathbf{y}} - \widetilde{\mathbf{H}}\widetilde{\mathbf{s}} \parallel^2 . \tag{2.2}$$

Note that, it is assumed that the MIMO receiver is provided with an accurate estimate of the channel $\widetilde{\mathbf{H}}$, which can be obtained during a separate training phase with the aid of pilot symbols.

As shown in Fig. 2.1, after being detected by the MIMO detector, the estimated symbols are transformed back into their corresponding bit representations using the demapper block. Digital-to-Analog (D/A) and Analog-to-Digital (A/D) converters are used at the MIMO transmitter and receiver, respectively, to convert the signals from digital to analog

domain and vice versa. Note that some other blocks such as the channel estimator block and the lattice reduction block are also shown in Fig. 2.1 at the receiver. The channel estimator provides an estimate of the current channel status based on the pre-known transmitted pilot symbols. However, in this thesis we assume that the channel is perfectly known to the receiver.

In addition to the complex channel model described above, the equivalent real model can also be derived using a Real Value Decomposition (RVD) scheme [11]. However, in this thesis, in order to simplify the hardware implementation of the MIMO detector and the QRD blocks, a slightly different approach is used for the RVD scheme, which is more suitable for concurrent computations and the VLSI implementation. The real model of (2.1) can be written as:

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{v}, \tag{2.3}$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_{2N_R-1}, y_{2N_R}]^T$, $\mathbf{s} = [s_1, s_2, \cdots, s_{2N_T-1}, s_{2N_T}]^T$ and $\mathbf{H}$ are the equivalent real-valued vectors with the following mappings:

$$\begin{aligned} y_{2k-1} &= \Re\{\tilde{y}_k\}, \qquad y_{2k} = \Im\{\tilde{y}_k\} \\ s_{2k-1} &= \Re\{\tilde{s}_k\}, \qquad s_{2k} = \Im\{\tilde{s}_k\} \\ v_{2k-1} &= \Re\{\tilde{v}_k\}, \qquad v_{2k} = \Im\{\tilde{v}_k\}, \end{aligned}$$

Also, real-valued channel matrix $\mathbf{H}$ is derived from complex channel matrix $\widetilde{\mathbf{H}}$ based on the following mapping:

$$\mathbf{H} = \begin{bmatrix} \Re(\widetilde{H}_{11}) & -\Im(\widetilde{H}_{11}) & \cdots & \Re(\widetilde{H}_{1N_T}) & -\Im(\widetilde{H}_{1N_T}) \\ \Im(\widetilde{H}_{11}) & \Re(\widetilde{H}_{11}) & \cdots & \Im(\widetilde{H}_{1N_T}) & \Re(\widetilde{H}_{1N_T}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Re(\widetilde{H}_{N_R1}) & -\Im(\widetilde{H}_{N_R1}) & \cdots & \Re(\widetilde{H}_{N_RN_T}) & -\Im(\widetilde{H}_{N_RN_T}) \\ \Im(\widetilde{H}_{N_R1}) & \Re(\widetilde{H}_{N_R1}) & \cdots & \Im(\widetilde{H}_{N_RN_T}) & \Re(\widetilde{H}_{N_RN_T}) \end{bmatrix}_{2N_R \times 2N_T}, \tag{2.4}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary parts of a complex variable, respectively. Note that

$$s_i \in \Omega = \left\{(-\sqrt{Q}+1), \cdots, -1, +1, \cdots, (+\sqrt{Q}-1)\right\}, \tag{2.5}$$

where Ω is the set of possible real entries in the constellation for in-phase and quadrature parts with $|\Omega| = \sqrt{Q}$. Another way to describe (2.2) is to say the objective of the MIMO

Maximum-Likelihood (ML) detection method is to find the closest transmitted vector $\hat{\mathbf{s}}$ based on the observation $\mathbf{y}$, i.e.,

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\Omega^{2N_T}}{\arg\min} \parallel \mathbf{y} - \mathbf{H}\mathbf{s} \parallel^2 . \tag{2.6}$$

The above definitions, imply that $|\Omega|^{2N_T} = |\mathcal{O}|^{N_T}$, meaning that a complex $N_R \times N_T$ MIMO system can be modeled as a real $2N_R \times 2N_T$ MIMO system.

Now, let us denote the QR Decomposition of the channel matrix as $\mathbf{H} = \mathbf{Q}\mathbf{R}$, where $\mathbf{Q}$ is a unitary matrix of size $2N_R \times 2N_T$ and $\mathbf{R}$ is an upper triangular $2N_T \times 2N_T$ matrix. Performing the nulling operation by $\mathbf{Q}^H$ results in the updated system equation:

$$\mathbf{z} = \mathbf{Q}^H\mathbf{y} = \mathbf{R}\mathbf{s} + \mathbf{w}, \tag{2.7}$$

where $\mathbf{w} = \mathbf{Q}^H\mathbf{v}$. Note that here $\mathbf{Q}^H$ represents conjugate transpose of the matrix $\mathbf{Q}$, i.e. $\mathbf{Q}^H = (\mathbf{Q}^T)^*$). Since the nulling matrix is unitary, the noise, $\mathbf{w}$, remains spatially white and the norm vector in (2.6), which represents the ML detection rule, can be rewritten as [12]:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\Omega^{2N_T}}{\arg\min} \parallel \mathbf{z} - \mathbf{R}\mathbf{s} \parallel^2 . \tag{2.8}$$

Now, for the Soft-Output MIMO detection purposes, the MIMO detector needs to generate a posteriori probability (APP) about the encoded bits, $\mathbf{x}$, which is usually expressed as a Log-Likelihood Ratio (LLR) value. The LLR of bit x_k, the k-th bit of $\mathbf{x}$, is defined as:

$$\begin{aligned} \mathrm{LLR}(x_k|\mathbf{z}) &= \ln\frac{P[x_k = 1|\mathbf{z}]}{P[x_k = 0|\mathbf{z}]} && (2.9)\\ &\approx \min_{\mathbf{x}\in S_k^{(0)}} \parallel \mathbf{z} - \mathbf{R}\mathbf{x} \parallel^2 - \min_{\mathbf{x}\in S_k^{(1)}} \parallel \mathbf{z} - \mathbf{R}\mathbf{x} \parallel^2 && (2.10) \end{aligned}$$

where (2.10) is derived from (2.9) based on standard simplifications [13]. Also, $\mathrm{S}_k^{(1)}$ and $\mathrm{S}_k^{(0)}$ represent all vectors $\mathbf{x}$ with bit position x_k being "1" and "0", respectively.

2.2 Simulation Framework for MIMO Detection and Pre-Processing Schemes

The bit-error-rate (BER) results in this thesis have been obtained from MATLAB/Verilog simulations and/or tested chip measurements, based on the i.i.d. channel model assumption. This model is valid in rich-scattering environments with sufficient spacing between the antennas (on the order of one wavelength) unless explicitly mentioned otherwise. It is further noted that all presented simulation results assume perfect channel knowledge at the receiver so that the channel estimation and detection can be separated. Also, for the purpose of MIMO detection and pre-processing, it is assumed that the channel is quasi-static and is updated after every four consecutive channel use. Note also that the Error Probability simulations for the proposed Soft-Output MIMO detection scheme, presented in Sections 3.3.2 and 3.3.4, also use the same simulation framework as the BER simulations.

Furthermore, for the BER simulations for the Soft-Output MIMO detector, it is assumed that the MIMO system uses Error Correcting Codes (ECC). The ECC coded MIMO system model utilizes Convolutional Turbo Codes (CTC) with rate = 1/2 and 600 bytes/block. Also, the CTC Decoder uses 8 decoder iterations and the interleaver coefficients and other simulation parameters are set according to the IEEE 802.16e WiMAX standard [6].

In terms of the modulation selection, the simulation results for 16-QAM and 64-QAM modulation schemes are presented. However, for implementation purposes, 64-QAM was chosen for two reasons. First, most of the hardware implementations for MIMO detectors reported in the literature to-date focus on the 16-QAM scheme, due to the higher complexity of the designs for 64-QAM constellation, which motivates us to fill this gap. Secondly, 64-QAM has been chosen to be one of the mandatory supported constellations in several standards including IEEE 802.16e (WiMAX 2×2), IEEE 802.16m (WiMAX 4×4), IEEE 802.11n WLAN (2×2 MIMO), 3GPP LTE and LTE-Advanced, which practically justifies its implementation. Both floating-point and fixed-point simulation results are presented and discussed throughout the thesis.

2.3 MIMO Detection Schemes

For Spatial Multiplexing (SM) schemes, we assume that the channel matrix $\widetilde{\mathbf{H}}$ is perfectly known at the receiver. Therefore, the task of a MIMO detector is to provide the decision on the transmitted symbol vector $\widetilde{\mathbf{s}}$, given the received symbol vector $\widetilde{\mathbf{y}}$, where $\widetilde{\mathbf{y}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{s}} + \widetilde{\mathbf{v}}$. There are two classes of MIMO detectors: *Hard-Decision* detectors and *Soft-Decision* detectors. The Hard-Decision detectors are useful for detecting uncoded or coded transmissions, where the decision of MIMO detectors (the estimated symbol vector) will be used as the final decision. The Soft-Decision detectors are used in ECC coded MIMO systems, where an iterative Receiver needs soft information (a posteriori probability values) being exchanged between detection and decoding modules following the "turbo principle" [14]. In this thesis, we focus on the Soft-Output MIMO detection problem as most practical wireless communication systems employ iterative Forward Error Correction (FEC) using ECC codes such as Convolutional Turbo Codes (CTC) and Low-Density Parity-Check (LDPC) codes.

As shown in Fig. 2.2, current MIMO detection schemes can be listed within the context of the following main categories:

- Exhaustive search Maximum-Likelihood (ML) detection.
- Sub-optimal linear receivers (ZF, MMSE).
- Sub-optimal non-linear receivers (V-BLAST, SIC).
- Near-optimal non-linear receivers (Sphere Decoder (SD), K-Best).

In ideal situations, it is desired to implement low-power MIMO detectors with near Maximum-Likelihood (ML) performance. However, the complexity of the optimal ML detection scheme grows exponentially with the number of transmit antennas and the constellation order. Therefore, suboptimal lower-complexity approaches need to be developed for practical commercial applications. Sub-optimal Linear detectors are one of the alternatives for the lower complexity detectors, which are based on the principles of Zero-Forcing (ZF) or Minimum Mean-Square Error (MMSE). Although they can greatly reduce the

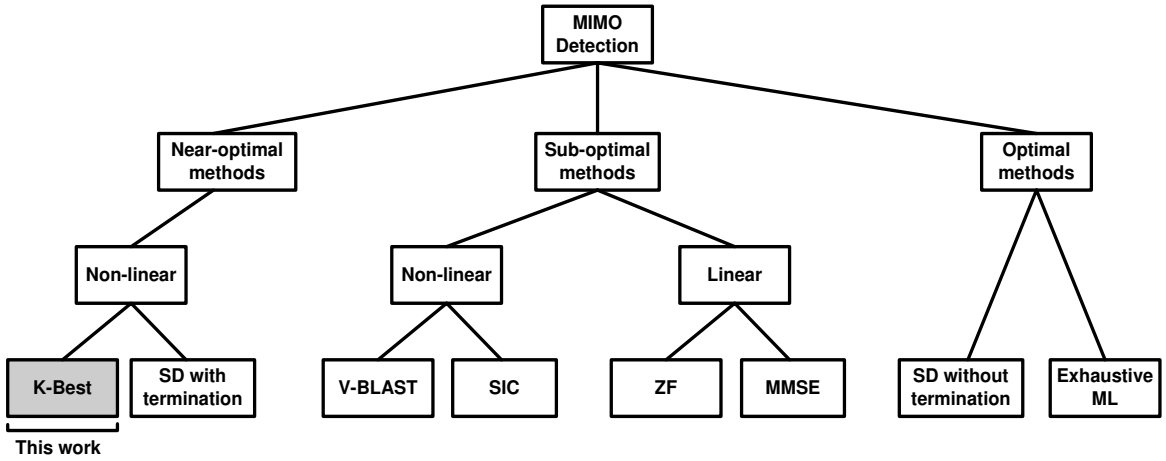

Figure 2.2: Taxonomy of MIMO detection algorithms [1]. The focus of this thesis is highlighted.

computational complexity, they suffer from significant performance loss. Successive Interference Cancelation (SIC) detectors and V-BLAST detectors [15] are also prone to decision error propagation and can only provide minor performance improvement.

The other alternative is to use Near-optimal Non-linear detectors, [16], [17]. Depending on how they carry out the non-exhaustive search, the Near-optimal Non-linear detector methods generally fall into two main categories, namely *depth*-first search, and *breadth*-first search. Sphere decoding (SD) [18] is the most attractive depth-first approach whose performance is ML under the assumption of unlimited execution time, [16]. However, the actual runtime of the algorithm (search time for estimated symbol vector) is dependent not only on the channel realization, but also on the operating signal-to-noise-ratio (SNR) [19]. Thus leading to a variable throughput rate, which results in extra overhead in the hardware due to the extra required FIFO buffers and lower hardware utilization.

Among the breadth-first search methods, the most well-known approach is the K-Best algorithm [20]. The K-Best detector guarantees an SNR-independent fixed-throughput with a performance close to ML. Being fixed-throughput in nature along with the fact that the breadth-first approaches are feed-forward detection schemes, makes them especially attractive for VLSI implementation. In this thesis, we focus on the design of a K-Best detector, which has been highlighted with a gray box in Fig. 2.2. The following presents details about breadth-first search methods and the conventional K-Best algorithm.

2.3.1 Conventional K-Best MIMO Detection Algorithm

Breadth-first tree traversal is a nonrecursive scheme, which starts from the root and traverses the tree in the forward direction only. On each level, the algorithm visits all admissible nodes and considers their associated children to construct a new set of admissible nodes on the next level before it proceeds [1]. In each level, a subset of all visited nodes are chosen as the surviving admissible nodes based on a criterion (e.g., their Partial Euclidean Distance (PED) from the received symbol). For the final level, the examined children, corresponding to the admissible leaves, comprises a set from which the decoder finally searches for the solution of (2.6).

Among the breadth-first search methods, the most well-known approach is the K-Best algorithm [20]. To understand the K-Best algorithm, let us first consider an $N_R \times N_T$ Q-QAM MIMO system. The detection problem of such a system can be formulated as a tree-search problem with N_T levels in the complex domain and $2N_T$ levels in the real domain through the RVD scheme [1]. Therefore, given an implementation in the real-domain, the problem in (2.6) can be considered as a tree-search problem with $2N_T$ levels.

The K-Best algorithm explores this tree from the root to the leaves by expanding each level and selecting the K best candidates in each level, which are called the surviving nodes of that level based on a criterion [21]. To make this clearer, let us consider K surviving nodes in level i. Each of these nodes has $\sqrt{Q}$ possible children in level $i+1$, from the symmetry in the Q-QAM constellation. The K-Best algorithm visits all these children and calculates their Partial Euclidean Distances (PEDs) resulting in $K\sqrt{Q}$ children at level $i+1$. Once the PED values are calculated, the K-Best algorithm sorts all these $K\sqrt{Q}$ children and selects the K best children as the surviving nodes in level $i+1$. The K-Best algorithm is a feed-forward detection method proceeding in the forward direction only. This method offers a trade-off between optimality and complexity with respect to the value of K [22], [23]. Thus an appropriate value of K should be determined using extensive simulations for each system framework and/or application.

Now, as discussed in Section 2.1, the updated ML detection rule (with QR Decomposition) can be rewritten as [12]:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\Omega^{2N_T}}{\arg\min} \parallel \mathbf{z} - \mathbf{R}\mathbf{s} \parallel^2 . \tag{2.11}$$

Exploiting the upper triangular nature of $\mathbf{R}$, this norm vector can be further expanded as:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\Omega^{2N_T}} \sum_{l=1}^{2N_T} \left| z_l - \sum_{j=l}^{2N_T} r_{lj} s_j \right|^2, \tag{2.12}$$

which can be thought of as a tree-search problem with $2N_T$ levels. Due to the upper-triangular structure of matrix $\mathbf{R}$, the K-Best algorithm starts from the last row of the matrix (detection tree), i.e., row(level) $l = 2N_T$. Therefore, equation (2.12) can be evaluated in a recursive manner as follows.

$$T_l\big(\mathbf{s}^{(l)}\big) = T_{l+1}\big(\mathbf{s}^{(l+1)}\big) + \left| e_l\big(\mathbf{s}^{(l)}\big) \right|^2 \tag{2.13}$$

$$e_l\big(\mathbf{s}^{(l)}\big) = z_l - \sum_{j=l}^{2N_T} r_{lj} s_j = L_l\big(\mathbf{s}^{(l)}\big) - r_{ll} s_l, \tag{2.14}$$

where $\mathbf{s}^{(l)} = [s_l\ s_{l+1} \cdots s_{2N_T}]^T$, $T_l\big(\mathbf{s}^{(l)}\big)$ is the accumulated partial Euclidean distance (PED) with $T_{2N_T+1}\big(\mathbf{s}^{(2N_T+1)}\big) = 0$, $\left| e_l\big(\mathbf{s}^{(l)}\big) \right|^2$ denotes the distance increment between two successive nodes/levels in the tree, and

$$L_l\big(\mathbf{s}^{(l)}\big) = z_l - \sum_{j=l+1}^{2N_T} r_{lj} s_j = r_{ll}(\bar{z}_l - \sum_{j=l+1}^{2N_T} \bar{r}_{lj} s_j) = r_{ll} \bar{L}_l\big(\mathbf{s}^{(l)}\big), \tag{2.15}$$

where $\bar{z}_l$, and $\bar{r}_{lj}$ denote the scaled z_l and r_{lj} by r_{ll}, respectively, (i.e., $z_l = \bar{z}_l r_{ll}$, and $r_{lj} = \bar{r}_{lj} r_{ll}$).

Finally, the path with the lowest PED at the last level of the tree is the hard-decision output of the detector, whereas, for a soft-decision output, all of the existing paths at the last level are considered to calculate the Log-Likelihood Ratio (LLR) values. The details about extension of the K-Best algorithm to produce soft outputs will be presented in Chapter 3.

2.4 Matrix QR Decomposition Algorithms

Matrix triangularization and orthonormalization (QR Decomposition) is an essential signal processing task for MIMO receivers. The QR Decomposition (QRD) operation decomposes an $n \times m$ matrix $\mathbf{H}$ into an $n \times m$ unitary and orthonormal matrix $\mathbf{Q}$ (such that $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$, where $\mathbf{Q}^H = (\mathbf{Q}^T)^*$) and an $m \times m$ upper triangular matrix $\mathbf{R}$. The 3 common algorithms

to perform matrix QR Decomposition include: the Modified Gram-Schmidt Orthonormalization (MGS) Algorithm, Householder transformations and Givens rotations.

2.4.1 The Modified Gram-Schmidt Orthonormalization Algorithm

The MGS algorithm is based on the Gram-Schmidt linear algebra technique for orthogonalizing and normalizing a set of basis vectors in an inner product space. It uses vector projections and normalizations to generate an orthonormal set of column vectors for **Q** and the upper triangular entries for **R**. The columns of **Q** represent the set of basis vectors, and the entries for **R** represent the inner-product of the original columns of **H** with those basis vectors (i.e. the components along the orthonormal basis vectors). The following are the primary equations for the Gram-Schmidt algorithm for QR Decomposition:

For a given matrix **H** with n columns:

$$\mathbf{H} = [h_1 \ h_2 \ \cdots \ h_n] \tag{2.16}$$

For the i^{th} column, let $e_i = \frac{h_i}{\|h_i\|}$. Now, define the projection of vector h_j on vector e_i to be:

$$proj_{e_i} h_j = \frac{< e_i \ , \ h_j >}{< e_i \ , \ e_i >} \ e_i \tag{2.17}$$

where, $< e_i \ , \ h_j >$ represents inner product of $n \times 1$ vectors e_i and h_j. This inner product can be computed using the following equation:

$$< e_i \ , \ h_j >= \sum_{k=1}^{n} e_i(k) \times h_j(k) \tag{2.18}$$

where, $e_i(k)$ and $h_j(k)$ represent kth elements of the vectors e_i and h_j, respectively. Now, define vectors u_1, u_2, $\cdots$, u_1 such that:

$$u_j = h_j \ - \ \sum_{i=1}^{j-1} proj_{e_i} h_j \tag{2.19}$$

Therefore the matrix **Q** will be:

$$\mathbf{Q} = [e_1 \ e_2 \ \cdots \ e_n] = \left[\frac{u_1}{\| \ u_1 \ \|} \ \frac{u_2}{\| \ u_2 \ \|} \ \cdots \ \frac{u_n}{\| \ u_n \ \|} \right] \tag{2.20}$$

And the matrix $\mathbf{R}$ can be obtained as:

$$\mathbf{R} = \mathbf{Q}^H\mathbf{H} = \begin{bmatrix} \| u_1 \| & < e_1 \ , \ h_2 > & < e_1 \ , \ h_3 > & \cdots & < e_1 \ , \ h_n > \\ 0 & \| u_2 \| & < e_2 \ , \ h_3 > & \cdots & < e_2 \ , \ h_n > \\ 0 & 0 & \| u_3 \| & \cdots & < e_3 \ , \ h_n > \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \| u_n \| \end{bmatrix} \tag{2.21}$$

Note that $< e_i \ , \ h_i >= \| u_i \|$ and $< e_i \ , \ h_j >= 0$ for $i \ > \ j$.

2.4.2 Givens Rotations

The QR Decomposition algorithm with Givens rotations uses a series of two-dimensional vector rotations to nullify the lower diagonal elements of the $\mathbf{R}$ matrix one by one. Each of these acts of making the elements zero is referred to as a Vectoring operation. For each Vectoring operation, there are corresponding Rotation operations to rotate the other column vectors in the matrix by the same angle used for the Vectoring operation.

To achieve the combination of Vectoring and Rotation, the entire matrix is multiplied by a Givens rotation matrix, $\mathbf{G}$. The product of all the Givens Rotation matrices G_1 to G_n (where n is the number of lower diagonal elements in the matrix $\mathbf{H}$ that need to be nullified) gives the $\mathbf{Q}^H$ matrix.

The Givens rotation matrix to nullify the element $H_{i,j}$ can be represented as follows:

$$\mathbf{G}(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{2.22}$$

where $c = \cos\theta$ and $s = \sin\theta$ appear at the intersection of the i^{th} row and j^{th} column. In

other words, a Givens rotation matrix is an identity matrix with the following substitutions:

$$\begin{aligned} G_{i,i} &= \cos\theta \\ G_{j,j} &= \cos\theta \\ G_{i,j} &= \sin\theta \\ G_{j,i} &= -\sin\theta \end{aligned}$$

Using Givens rotations, a single two-dimensional vector rotation can be illustrated as follows:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{2.23}$$

where, $r = \cos\theta \cdot a + \sin\theta \cdot b = \sqrt{a^2 + b^2}$.

Now, as mentioned earlier, the final output matrix $\mathbf{R}$ can be obtained by pre-multiplying the $\mathbf{H}$ matrix by all the $\mathbf{G}$ matrices, as shown below:

$$\mathbf{G}_n\mathbf{G}_{n-1}\cdots\mathbf{G}_3\mathbf{G}_2\mathbf{G}_1\mathbf{H} = \mathbf{R} \tag{2.24}$$

In addition, the combination of $\mathbf{G}$ matrices can be used to find $\mathbf{Q}$, as follows:

$$\mathbf{G}_n\mathbf{G}_{n-1}\cdots\mathbf{G}_3\mathbf{G}_2\mathbf{G}_1\mathbf{H} = \mathbf{R} = \mathbf{Q}^H\mathbf{Q}\mathbf{R} = \mathbf{Q}^H\mathbf{H}$$

$$\mathbf{Q} = (\mathbf{G}_n\mathbf{G}_{n-1}\cdots\mathbf{G}_3\mathbf{G}_2\mathbf{G}_1)^H \tag{2.25}$$

2.4.3 Householder Transformations

A Householder transformation is a reflection that takes a vector and reflects it about some plane. This reflection operation can be used to rotate a multi-dimensional vector in such a way that all coordinates, except the first one, are nullified. Thus, Householder transformations can be used to perform QR Decomposition using a very small number of computational steps. However, each of these steps are computationally intensive, since a large number of multiplication, division and addition operations need to be performed in each step [24]. Also, since a Householder transformation affects multiple rows simultaneously, it is not straightforward to perform the multiple transformations in parallel [24].

The Householder transformations can be used to compute QR Decomposition of a matrix

as follows. Consider an input matrix $\mathbf{H}$ with n columns:

$$\mathbf{H} = [h_1 \; h_2 \; \cdots \; h_n]$$

The QR Decomposition using Householder transformations starts by nullifying the elements in the first column of the $\mathbf{H}$ matrix, by pre-multiplying it with the transformation matrix T_1. The matrix T_1 can be calculated as:

$$\mathbf{T}_1 = I_1 - 2\frac{u_1 u_1^H}{u_1^H u_1} \tag{2.26}$$

where, I_1 is an $n \times n$ identity matrix. Also, $u_1 = h_1$ - $\| \; h_1 \; \| e_1$, where $e_1 = [1 \; 0 \; 0 \; \cdots \; 0]^T$. Thus, pre-multiplication of the $\mathbf{H}$ matrix by T_1 zeroes all entries of the first column below the diagonal and updates all other columns appropriately. The Householder transformation matrix T_i, to zero elements below diagonal in h_i, can be obtained using the following generalized equation:

$$\mathbf{T}_i = I_i - 2\frac{u_i u_i^H}{u_i^H u_i} \tag{2.27}$$

where, I_i is an identity matrix with $n - i + 1$ rows and columns. Hence, pre-multiplication of T_1, T_2, ..., T_n yields the upper-triangular $\mathbf{R}$ matrix, as follows:

$$\mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1 \mathbf{H} = \mathbf{R} \tag{2.28}$$

Also, the unitary matrix $\mathbf{Q}$ can be attained as:

$$\mathbf{Q} = (\mathbf{T}_n \mathbf{T}_{n-1} \cdots \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1)^H \tag{2.29}$$

The advantages and disadvantages of these 3 QRD algorithms: Modified Gram-Schmidt algorithm, Givens rotations and Householder transformations will be investigated further in Chapter 5 from an implementation point of view. This analysis will then be used to develop a novel low-complexity and highly parallel QRD scheme for decomposing complex channel matrix $\mathbf{H}$.

3 The Soft-Output K-Best MIMO Detection Algorithm

3.1 Introduction

As outlined in Chapter 2, the K-Best algorithm offers SNR-independent fixed throughput and performance close to optimal ML detection. Its feed-forward nature and fixed-throughput characteristic make the K-Best algorithm the method of choice for VLSI implementation of the proposed MIMO detector in this thesis. However, an area and power efficient high-throughput VLSI implementation of a soft-output K-Best MIMO detector for high-order quadrature amplitude modulation (QAM) schemes still poses many challenges.

To address these challenges, we are motivated by [1] that proposes a scalable pipelined architecture for 4×4 64-QAM hard-output MIMO detector that offers a fixed critical-path, independent of the constellation order. The architecture in [1] uses an on-demand expansion scheme for visiting the intermediate nodes of the search tree, efficient distributed sorters and is scalable to larger number of antennas and constellation orders. In 0.13μm CMOS technology, this hard-output MIMO detector achieves a detection throughput of up to 675 Mbps, while occupying 0.9 mm^2 core area with 135 mW power consumption at 282 MHz clock frequency [25].

However, in practice, most wireless communication systems use Error Correcting Codes (ECC), demanding a posteriori probability (APP) information about each bit for ECC decoding purposes. Coded MIMO systems with Soft-Output MIMO detectors generally offer a much superior bit error rate (BER) performance compared to uncoded MIMO systems. Therefore, Soft-Output MIMO signal detection is highly desirable. In this thesis, we aim to extend the Hard K-Best detector presented in [1] to produce log-likelihood ratio (LLR) value for each transmitted bit. The Soft-Output K-Best MIMO detection scheme proposed in this thesis offers large BER performance improvement with marginal increase in computational complexity, while still preserving the high detection throughput offered by the Hard K-Best detector.

3.2 Existing Soft K-Best Detection Schemes and Challenges

Let us consider an $N_R \times N_T$ MIMO system with Q-QAM modulation scheme. Assuming an implementation in the real-domain (using Real Value Decomposition), the detection problem can be considered as a tree-search problem with $2N_T$ levels. A Soft-Output K-Best MIMO detection scheme needs to compute the LLR value for each of the $N_T log_2(Q)$ transmitted bits in $\mathbf{x}$ using equation (2.10), also shown below. The LLR of bit x_k, the k-th bit of $\mathbf{x}$, is defined as:

$$\mathrm{LLR}(x_k|\mathbf{z}) = \ln\frac{P[x_k = 1|\mathbf{z}]}{P[x_k = 0|\mathbf{z}]} \tag{3.1}$$

$$\approx \min_{\mathbf{x}\in S_k^{(0)}} \| \mathbf{z} - \mathbf{Rx} \|^2 - \min_{\mathbf{x}\in S_k^{(1)}} \| \mathbf{z} - \mathbf{Rx} \|^2 \tag{3.2}$$

where, $\mathrm{S}_k^{(1)}$ and $\mathrm{S}_k^{(0)}$ represent all vectors $\mathbf{x}$ with bit position x_k being "1" and "0", respectively. The following describes the existing Soft K-Best detection schemes and the challenges they pose.

3.2.1 Conventional Soft K-Best Detection Scheme

The conventional method for computing LLR values for Soft K-Best detection scheme simply extends the K best paths at the level $2\mathrm{N}_T$ - 1 to all of the possible $\sqrt{Q}$ symbols at the level $2\mathrm{N}_T$. The resulting $\mathrm{K}\sqrt{Q}$ vectors of dimensions $2\mathrm{N}_T \times 1$ are then categorized to be either in $\mathrm{S}_k^{(1)}$ or $\mathrm{S}_k^{(0)}$ for each bit k, and are used to compute LLR value for each of the $\mathrm{N}_T\mathrm{log}_2(\mathrm{Q})$ transmitted bits. However, this conventional Soft K-Best detection scheme leads to the following challenges:

Challenge 1: For large QAM constellations, this method for computing LLR values leads to high computational complexity. Moreover, the K-Best algorithm for large constellations tends to use larger K values to attain acceptable BER performance, which increases the computational complexity even further. This is because of the fact that the conventional Soft K-Best scheme extends the K paths at level $2\mathrm{N}_T$ - 1 to all possible $\mathrm{K}\sqrt{Q}$ vectors exhaustively. The scheme then needs to compute the Partial Euclidean Distance (PED) values for each of these $\mathrm{K}\sqrt{Q}$ paths and compare them to produce the LLR value for each

of the $N_T\log_2(Q)$ transmitted bits. For example, for a Soft K-Best detector used in 4×4 64-QAM MIMO systems with $K = 10$, this scheme requires computation and comparison of 80 ($K\sqrt{Q} = 10*\sqrt{64} = 80$) PED values to produce LLR values of 24 ($N_T\log_2(Q) = 4*\log_2(64) = 24$) bits.

From a VLSI implementation perspective, this large computational complexity leads to either low throughput or to large area and power requirements. For example, the Soft K-Best detector proposed in [26] for 4×4 64-QAM MIMO system with $K = 32$ uses this scheme to generate soft outputs. However, due to the large computational complexity, this Soft K-Best detector can only achieve a peak throughput of 115 Mbps, while consuming 1200 mW power and a silicon area of 31 mm^2 in 0.13μm CMOS.

Challenge 2: The K-Best algorithm offers a trade-off between optimality and complexity with respect to the value of K [19] [23]. This implies that a possible way to decrease computational complexity of the Soft K-Best scheme is to avoid use of very large K values. However, for nominal K values, the conventional Soft K-Best detection scheme provides only a marginal improvement in the BER performance compared to a Hard K-Best detector. This can be noticed from the BER curves provided in Fig. 3.1 and Fig. 3.2 for 16-QAM, K=10 and for 64-QAM, K=10, respectively.

This phenomenon can be explained by examining the quality of the output LLR values from the conventional Soft K-Best detection scheme. Note that LLR for a particular bit will be assigned either $+\infty$ or $-\infty$, when all of the vectors used for LLR computation have a '0' or a '1' for the given bit. Let us define the *Invalid LLR Percentage* as a metric for quantizing LLR quality. The metric *Invalid LLR Percentage* computes the percentage of bits in the $2N_T \times 1$ vector that has an LLR of $+\infty$ or $-\infty$. Thus a lower *Invalid LLR Percentage* would imply better LLR quality, and hence superior BER performance. Table 3.1 shows the minimum, average and maximum *Invalid LLR Percentage* values for a conventional Soft K-Best detector in a 4×4 MIMO system for 64-QAM and $K = 10$.

From Table 3.1, note that for the complete $2N_T \times 1$ output vector, the *Invalid LLR Percentage* is very high. This in turn implies poor LLR quality and hence explains the inferior BER performance. From Table 3.1, it can also be noticed that the LLR quality for output complex symbol 1 is much superior compared to the LLR quality for the rest of the N_T - 1 complex symbols. Note also that here the complex output symbol 1 corresponds to levels $2N_T$ - 1 and $2N_T$ of the detection tree, while the rest of the N_T - 1 complex symbols

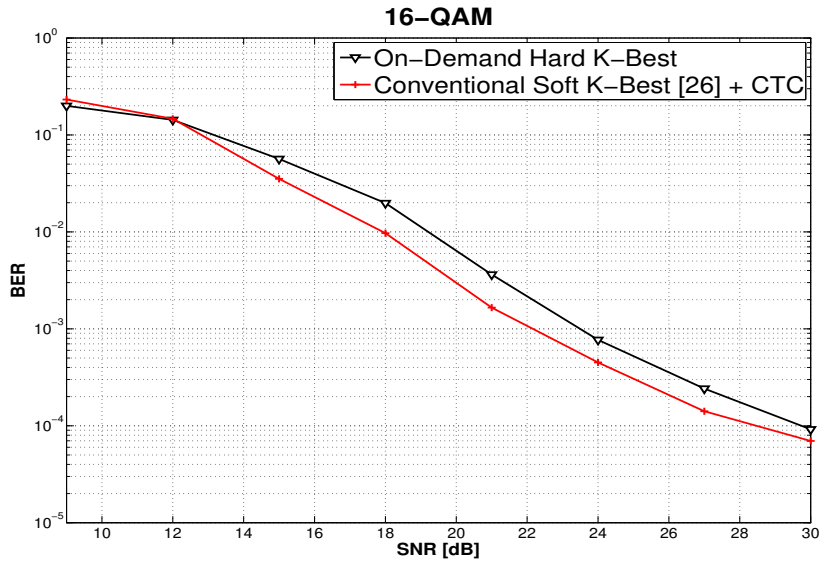

Figure 3.1: BER Performance of Hard K-Best and conventional Soft K-Best Detection scheme - for 4×4 MIMO system with 16-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations

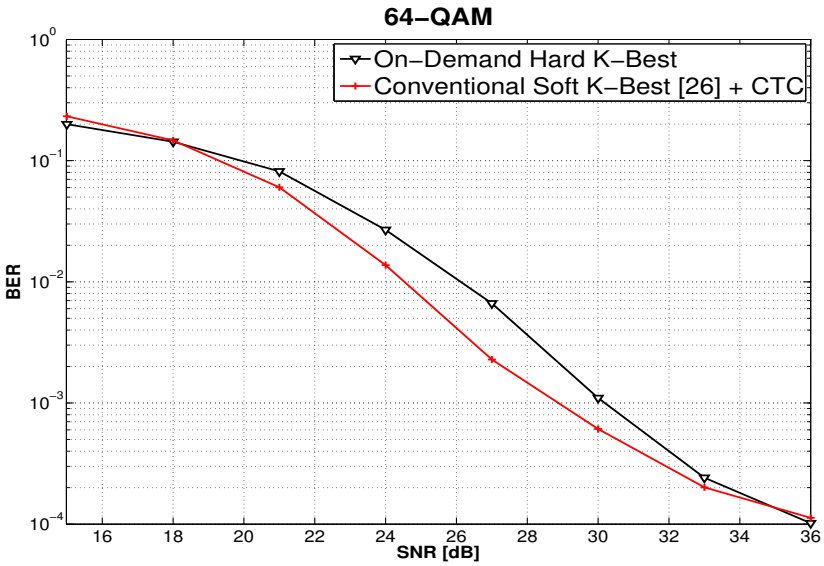

Figure 3.2: BER Performance of Hard K-Best and conventional Soft K-Best Detection scheme - for 4×4 MIMO system with 64-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations

Table 3.1: *Invalid LLR Percentage* values for conventional Soft K-Best detection in 4×4 64-QAM MIMO system with K=10

Metric:	Invalid LLR Percentage - Conventional Soft K-Best:
Complete Output Vector - Min	45.05%
Complete Output Vector - Max	65.37%
Complete Output Vector - Avg	57.23%
Output Complex Symbol 1 - Min	12.15%
Output Complex Symbol 1 - Max	46.73%
Output Complex Symbol 2 - Min	26.89%
Output Complex Symbol 2 - Max	94.37%
Output Complex Symbol 3 - Min	32.35%
Output Complex Symbol 3 - Max	96.57%
Output Complex Symbol 4 - Min	36.94%
Output Complex Symbol 4 - Max	97.70%

correspond to the first $2N_T$ - 2 levels of the detection tree.

This difference in the LLR quality can be explained using the following observation. For the conventional Soft K-Best scheme, there are only K distinct $(2N_T - 1) \times 1$ real symbol vectors, corresponding to the first $2N_T$ - 1 levels, in the complete set of $K\sqrt{Q}$ vectors. Hence, for the bits corresponding to the first $2N_T$ - 1 levels, only K vectors are used to compute their LLR values, leading to poor LLR quality. However, for bits corresponding level $2N_T$, all of the $K\sqrt{Q}$ vectors are utilized to compute LLR values, thus leading to a much superior LLR quality.

To summarize, since the conventional Soft K-Best detection scheme expands the paths at the last tree level exhaustively, it leads to a high computational complexity. This complexity increases dramatically with an increase in constellation size and value of K. Furthermore, for the smaller K values, the performance gain of the conventional Soft K-Best scheme, compared to the Hard K-Best detection, is negligible. This is due to the inferior LLR quality for the bits corresponding to the last N_T - 1 complex symbols, since only K paths are used to compute their LLR values.

3.2.2 The MKSE Soft K-Best Detection Scheme

The major issue with the conventional Soft K-Best detection scheme is the lack of significant improvement compared to Hard K-Best detection scheme, mainly due to the insufficient number of paths used to compute LLR values for the last N_T - 1 complex symbols. To resolve this issue, the authors in [11] propose to use the information contained in the discarded paths (DP), that are terminated pre-maturely and are not extended to the end sub-lattice. The resulting Modified K-Best Schnorr-Euchner (MKSE) detection scheme [11] will be discussed briefly below.

For the K-Best algorithm, only K smallest PED paths are chosen at each tree level, and the rest of the K($\sqrt{Q}$-1) paths are discarded. The MKSE detection scheme retains these K($\sqrt{Q}$-1) discarded paths, virtually augments them to full length by making assumptions on the remaining undetected symbols and then uses their final PED values to compute LLR for the $N_T\log_2(Q)$ transmitted bits. The ZF estimation[1] method, presented in [16], is the simplest method for augmenting discarded paths. The number of tree levels from which the discarded paths should be retained and augmented can be decided through simulations. The MKSE detector design presented in [11] for 4×4 16-QAM MIMO system utilizes the discarded paths from the last 4 tree levels.

Let us denote the number of tree levels from which the discarded paths are utilized as 'L'. Hence, this MKSE scheme uses L*[K($\sqrt{Q}$ - 1)] + K vectors to compute LLR for the bits corresponding to the first $2N_T$ - 1 levels, and L*[K($\sqrt{Q}$ - 1)] + K$\sqrt{Q}$ vectors to compute LLR for the last $\log_2(Q)/2$ bits. This increase in the number of vectors used for LLR computation reduces the invalid LLR count, and hence improves the LLR quality for the bits corresponding to the first $2N_T$ - 1 tree levels. Table 3.2 below shows the minimum, average and maximum *Invalid LLR Percentage* values for the MKSE detector in 4×4 MIMO system with 64-QAM modulation scheme, K=10 and L=6. Note that this table also repeats the *Invalid LLR Percentage* values from Table 3.1, for the conventional Soft K-Best detector, for comparison purposes.

From Table 3.2, it can be noticed that the *Invalid LLR Percentage* values are considerably smaller for the MKSE scheme, compared to those for the conventional approach. This improvement in the LLR quality is mainly due to the much larger number of paths used for LLR computation for the first $(2N_T$ -1)($\log_2(Q)/2$) bits. For example, for 4×4 64-QAM MIMO detector with K=10 and L=6, a total of 430 vectors are used to compute

[1]The ZF estimation method estimates an undetected symbol by simply rounding the received point to the nearest lattice point.

Table 3.2: *Invalid LLR Percentage* values for conventional and MKSE Soft K-Best detection in 4×4 64-QAM MIMO system with K=10

Metric:	Invalid LLR Percentage - Conv. Soft K-Best:	Invalid LLR Percentage - MKSE Soft K-Best:
Complete Output Vector - Min	45.05%	5.19%
Complete Output Vector - Max	65.37%	10.15%
Complete Output Vector - Avg	57.23%	7.17%
Output Complex Symbol 1 - Min	12.15%	0.00%
Output Complex Symbol 1 - Max	46.73%	2.21%
Output Complex Symbol 2 - Min	26.89%	0.00%
Output Complex Symbol 2 - Max	94.37%	6.31%
Output Complex Symbol 3 - Min	32.35%	0.00%
Output Complex Symbol 3 - Max	96.57%	16.30%
Output Complex Symbol 4 - Min	36.94%	20.57%
Output Complex Symbol 4 - Max	97.70%	24.67%

LLR for the first 21 bits. Note that for the same scenario, the conventional Soft K-Best scheme only used 10 vectors for LLR computation for these first 21 bits. This results in a substantial reduction in the *Invalid LLR Percentage* values for the first 3 symbols and it also improves the BER performance of the Soft K-Best detector. As can be seen from the BER curves shown in Fig. 3.3 for 4×4 64-QAM MIMO system, the MKSE detection scheme yields significant performance gain compared to the conventional Soft K-Best approach and Hard K-Best detection.

Challenges: The MKSE scheme improves the LLR quality and hence improves the performance. However, since it has to process a very large number of vectors, it still leads to very high computational complexity for large constellations, large antenna configurations and large value of K.

The MKSE detection scheme retains K($\sqrt{Q}$-1) discarded paths from each tree level and performs ZF augmentation [16] on them to extend them to the final tree level. Hence, assuming that the discarded paths are utilized from L tree levels, the MKSE scheme needs to process a total of LK($\sqrt{Q}$-1) discarded paths. For example, for 4×4 64-QAM MIMO system with K=10, this scheme needs to ZF augment and compute the updated PED for 70 discarded paths from each tree level. Furthermore, the number of paths to be processed at each level increases linearly as the algorithm goes further down in the detection tree. For the same example shown above and assuming L=6, the MKSE algorithm needs to ZF

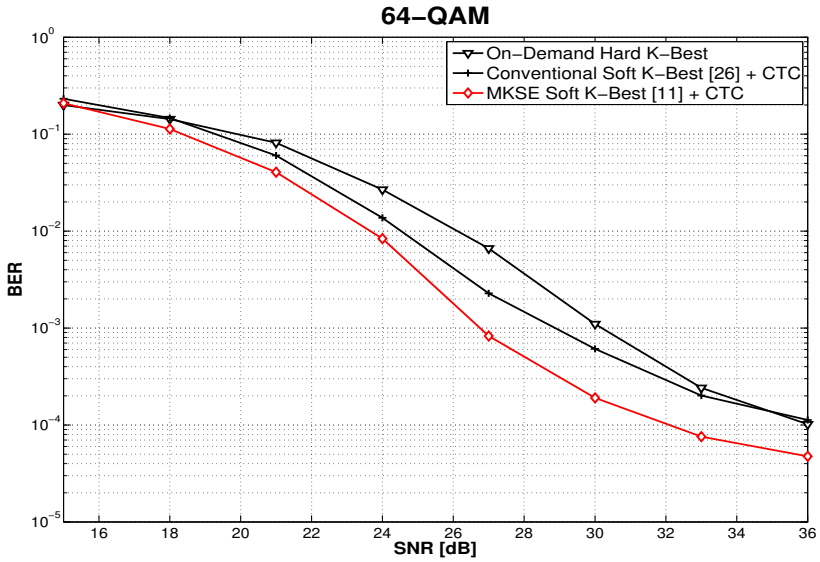

Figure 3.3: BER Performance of Hard K-Best, conventional Soft K-Best and MKSE Detection schemes - for 4×4 MIMO system with 64-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations:

augment and compute the updated PED for 70, 140, 210, 280, 350 and 420 discarded paths for tree levels 2, 3, 4, 5, 6 and 7, respectively. Furthermore, at the end of the detection tree, the MKSE scheme needs to sort and compare PED values for a total of 420 paths. Also, since the MKSE scheme extends the K best paths at level $2N_T$ - 1 to all $K\sqrt{Q}$ paths at level $2N_T$ exhaustively, these 80 exhaustively extended paths need to be compared to the 420 discarded paths for LLR computation purposes.

From a hardware perspective, this large number of discarded path ZF augmentation, PED computation and comparison leads to either a throughput bottleneck or large hardware complexity. In other words, for area and power constrained MIMO detectors, this leads to low throughput due to the large processing latency. On the other hand, for MIMO detectors that require high throughput, the large amount of computations to be performed in the given small number of cycles leads to large area and power requirements. Thus, the MKSE scheme yields significant improvement in the BER performance, however it leads to prohibitive computational complexity for large constellations and large antenna configurations. Hence, it is not immediately feasible to use the MKSE scheme to realize detectors to be used in MIMO systems with large antenna configurations, such as 4×4 and 8×8, and with large constellation size, such as 64-QAM and 256-QAM constellations.

3.3 Proposed MIMO Detection scheme

Many new 4G wireless standards require MIMO systems that support high data rates, high mobility and large antenna configurations. For example, the IEEE 802.16m and LTE-Advanced standards include applications with mobile speeds up to 350 km/h, maximum antenna configuration of 8×8 and downlink peak data rates of up to 1Gbps. Hence, these applications require MIMO detectors for large constellations and large antenna configurations that offer large enough throughput to meet the aggressive peak data rate requirements. Also, since these MIMO detectors are required for mobile applications, it is desirable to minimize silicon area and power consumption requirements.

The conventional Soft K-Best detection scheme, which just uses the paths extended exhaustively at the last tree level for LLR computation, poses 2 major issues: high computational complexity and lack of significant performance improvement compared to the Hard K-Best detection scheme. To improve the BER performance, the MKSE detection scheme is presented in [11], that retains and utilizes the discarded paths from selected levels of the detection tree, along with the exhaustively extended paths at the last level, to compute LLR values. However, the scheme increases computational complexity even further since it has to perform ZF augmentation, PED computation and PED comparison for the discarded paths. Hence, for large constellations and large antenna configurations, designing high throughput, area and power efficient MIMO detectors become challenging.

To resolve these issues, this thesis proposes a novel Soft K-Best detection scheme to compute LLR values for the transmitted bits efficiently. The proposed Soft K-Best detection scheme uses 3 major improvement ideas to reduce computational complexity, and hence to make hardware implementation of high-throughput low-complexity MIMO detectors possible, while not sacrificing any major BER performance gain. In other words, the proposed Soft K-Best detection scheme uses the ideas from the MKSE scheme [11], namely utilization of discarded paths and exhaustive path expansion at the last tree level, and improves upon them to reduce computational complexity. Also, note that the proposed Soft K-Best scheme uses the novel On-Demand Hard K-Best scheme presented in [1] and [27] as its basis, and extends it to allow LLR computation. The following provides a brief description of the On-Demand Hard K-Best scheme and details about the improvement ideas used by the proposed Soft K-Best scheme to reduce computational complexity.

3.3.1 On-Demand Hard K-Best Scheme [1]

Let us consider a $2N_R \times 2N_T$ real-valued MIMO system with channel matrix $\mathbf{H}$. As described earlier, this can be thought of as a detection problem in a tree with $2N_T$ levels, K nodes per level and $\sqrt{Q}$ children per node. The task of a Hard K-Best detection scheme is to find the lowest PED $2N_T \times 1$ vector at the last level of the tree. Because of the upper triangular structure of matrix $\mathbf{R}$, the algorithm starts from the last row of the matrix ($2N_T$-th row, which is the first level of the detection tree) and goes all the way up to the first row of the matrix, which is the last ($2N_T$-th) level of the detection tree. Note that in conventional Hard K-Best scheme, all the possible children of a level are expanded exhaustively. This exhaustive expansion grows significantly with the constellation size. The On-Demand Hard K-Best scheme [1], described below, offers a clever way to calculate the K best candidates at each level without performing an exhaustive search.

Let's consider level l of the tree and assume that the set of K-Best candidates in level $l-1$ (denoted by $\mathcal{K}_{l-1}$) is known. Each node in level $l-1$ has $\sqrt{Q}$ possible children, so there are $K\sqrt{Q}$ possible children in level l. One of the main elements of the On-Demand Hard K-Best scheme [1] is to find the children of each node on-demand and in the order of increasing PED rather than calculating the PED of all the children exhaustively. In other words, the key idea of the proposed distributed K-Best scheme is to find the First Child (FC) of each parent node in $\mathcal{K}_{l-1}$. Note that the first child refers to the child with the lowest *local* PED among all children of a parent. Finding the first child does not require any sorting and can be realized using a simple rounding operation. Among these first children, the one with the lowest PED is definitely one of the K-Best candidates in $\mathcal{K}_l$. That child is selected and is replaced by its Next Child (NC). Note that next child refers to the child with the next lowest *local* PED. This process is repeated K times to find the K-Best candidates in level $l(\mathcal{K}_l)$. The same procedure is performed for each level of the tree, resulting in the algorithm in Table 3.3 for the detection problem throughout the whole tree. Note that details about First Child (FC) and Next Child (NC) calculations will be presented in Appendix A.

The proposed scheme in Table 3.3 is pictorially depicted in Fig. 3.4 for level l where $\sqrt{Q} = 4$ and $K = 3$. It shows the way that $\mathcal{K}_l$ is derived from $\mathcal{K}_{l-1}$. The input to the algorithm is the K best selected nodes of level $l-1$, which are the current parents with corresponding PEDs of 0.1, 0.4, and 0.6. Each parent can be further expanded to four offsprings resulting in 12 children whose PEDs are shown in Fig. 3.4. Let $\mathcal{C}_l$ represent

Table 3.3: The On-Demand Hard K-Best Algorithm [1].

1) Find the K-Best children of level 1 ($\mathcal{K}_1$).
For $l = 2 : 1 : 2N_T$
2) Find the *first* child of all candidates in $\mathcal{K}_{l-1}$. Call this set $\mathcal{C}_l$.
3) For $k = 1 : K$
3.1) Select the child in $\mathcal{C}_l$ with the lowest PED.
3.2) Announce this child as the next K-Best candidate in level l (i.e., add it to $\mathcal{K}_l$).
3.3) Replace it with its *next* best sibling.
End
End

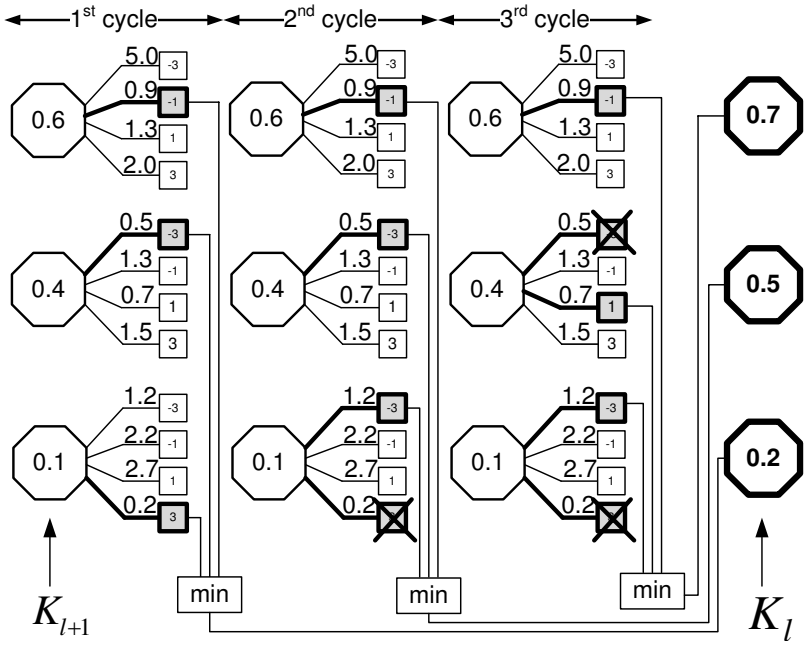

Figure 3.4: The On-Demand Hard K-Best algorithm for $\sqrt{M} = 4$ and $K = 3$ and example PED values [1].

the set consisting of all the current best children of all parents, and $\mathcal{D}_l$ represent their corresponding PEDs (in Fig. 3.4, $\mathcal{C}_l = \{c_{12}^l = -1, c_{21}^l = -3, c_{34}^l = 3\}$ with $\mathcal{D}_l = \{d_{12}^l = 0.9, d_{21}^l = 0.5, d_{34}^l = 0.2\}$, where c_{ij}^l represents the j-th child of the i-th parent in level l). It is easy to see that the child with the lowest PED in $\mathcal{C}_l$ is definitely one of the K-Best candidates in $\mathcal{K}_l$, e.g., c_{34}^l in Fig. 3.4 (1^{st} cycle). This is because the child with the lowest PED in $\mathcal{C}_l$ has globally the lowest PED. This child can be mathematically represented as $c_{\bar{k}\bar{m}}^l$ where

$$\bar{k}\bar{m} = \arg\min_{k,m}(d_{km}^l \in \mathcal{D}_l). \tag{3.3}$$

Thus this child should be added to $\mathcal{K}_l$. To find the next best child in the K-Best list, $c_{\bar{k}\bar{m}}^l$ and its corresponding PED ($d_{\bar{k}\bar{m}}^l$) are removed from $\mathcal{C}_l$, and $\mathcal{D}_l$, respectively (2^{nd} cycle in Fig. 3.4). Following this removal, the next best sibling of this child is added to $\mathcal{C}_l$ ($c_{31}^l = -3$ with $d_{31}^l = 1.2$ in Fig. 3.4). Taking the same approach, the child in $\mathcal{C}_l$ with the lowest PED is definitely among the final K-Best list ($c_{21}^l = -3$ with $d_{21}^l = 0.5$). This child should be added to $\mathcal{K}_l$, be removed from $\mathcal{C}_l$ and be replaced by its next best sibling (3^{rd} cycle). This procedure repeats $K = 3$ times to find all the K-Best candidates (see Fig. 3.4). The final children in the K-Best list are c_{34}^l, c_{21}^l, and c_{23}^l with PEDs 0.2, 0.5, and 0.7, respectively. Note that using the On-Demand Hard K-Best scheme, only 5 children of 12 possible children are visited in Fig. 3.4. This amount of savings becomes increasingly significant for large K and/or M values.

3.3.2 Improvement 1: Relevant Discarded Paths Selection

As discussed earlier, the MKSE scheme retains and utilizes K($\sqrt{Q}$-1) discarded paths (DPs) from each tree level. However, due to the On-Demand nature of child expansion, the Hard K-Best scheme presented in [1] and [27] only produces K-1 discarded paths at each tree level. The current Soft K-Best scheme would accumulate these K-1 discarded paths at each tree level. Hence, assuming that the discarded paths are utilized from L tree levels, a straightforward extension of the Hard K-Best scheme from [1] and [27] will still need to process a total of L(K-1) discarded paths to produce soft outputs.

Let us denote the partial paths at each tree level (paths from root node to an intermediate tree level) or the complete paths (paths from root node to the last tree level) by $\mathbf{x}$, and a bit position within the path by j. Also denote the Minimum PED (MinPED) for the j^{th} bit in $\mathbf{x}$ being '0' as MinPED_j^0 and the Minimum PED for the j^{th} bit being '1' as MinPED_j^1. The following 3 observations lead to the derivation of the first improvement

idea, namely: selection and utilization of only relevant discarded paths.

1. For the Hard K-Best scheme, the K-1 discarded paths at each tree level are already sorted according their PED values.

2. Among the paths at a particular tree level and for jth bit in these paths, a K Best path will definitely yield smaller MinPED$_j^0$ and MinPED$_j^1$, compared to a discarded path at that tree level.

3. If a particular discarded path does not provide any extra information (i.e. MinPED for one or more bits), then there is no advantage in storing and ZF augmenting that discarded path.

Based on these observations, the current improvement idea proposes to analyze the K best paths and the rest of the discarded paths at each tree level, and only select those discarded paths for further processing (ZF augmentation and LLR computation) that yield MinPED for at least one of the bits. The rest of the unselected discarded paths at that tree level should not be stored or processed any further, and hence they should be just abandoned. The relevant discarded paths are selected using the following process at each tree level:

Step 1: First, populate a Bit Occurrences table for the symbols corresponding to the current tree level and all previous levels, using the K best paths at the current tree level.

Step 2: Then, attempt to fill in the remaining entries in the Bit Occurrences table using the discarded paths accumulated and forwarded from the previous levels.

Step 3: At the end, examine each of the sorted discarded paths at the current level one by one (in the order of ascending PEDs), and select the current discarded path for further processing only if it fills at least one of the void entries in the Bit Occurrences table for the current tree level.

Note that for Level k, a bit occurrence table is simply a table of dimensions $2 \times (N_T - k - 1) * (log_2(Q)/2)$, that keeps track of occurrences of "0" and "1" values for $(N_T - k - 1) * (log_2(Q)/2)$ bits (an $(N_T - k - 1) \times 1$ real-valued vector) in the K-Best paths and accumulated chosen discarded paths. Thus, this improvement idea proposes to select

and forward only the relevant discarded paths that contain useful information for LLR computation, out of the total L(K-1) discarded paths. In other words, this improvement idea recognizes and eliminates the irrelevant discarded paths to reduce the overall computational complexity significantly. Through mathematical analysis, the largest number of selected discarded paths can be derived to be ($(2N_T - 1)(\log_2(Q)/2)$) - (K-1), for the given constellation size, antenna configuration and the value of K. Note that the number of selected discarded paths is the largest in the worst case scenario, when discarded paths are utilized from all of the intermediate tree levels and when all of the K-1 best paths at the last level only yield one different bit from all of the previous K best paths. Thus, this improvement idea results in large savings in computational complexity, since the Soft K-Best detection scheme now only needs to ZF augment, compute and compare the PED for a maximum of ($(2N_T - 1)(\log_2(Q)/2)$) - (K-1) paths. For example, for 4×4 64-QAM MIMO detector with K=10 and L=6, a maximum of 12 paths need to be processed using this improvement idea, as opposed to the 54 discarded paths processed earlier.

However, this improvement idea only uses the discarded paths that have minimum PED for a particular bit at the current tree level and ignores the PED for the unselected paths. Hence, even though this improvement idea reduces the computational complexity by a large amount, it might lead to some BER performance degradation. This might happen in the following case: if for a particular bit, ZF augmentation of an unselected discarded path yields smaller final Euclidean distance compared to the final Euclidean distance for ZF extension of a chosen discarded path. This case is classified as an error case in the LLR computation, since it eventually causes LLR quality degradation compared to the case where all K-1 discarded paths at each tree level are forwarded for LLR computation. In order to quantize the error probability, a 4×4 64-QAM MIMO detector was simulated for large number of transmitted bits and the occurrences of the error case were monitored. Figure 3.5 shows the error percentage plot obtained using these simulations.

From the error percentage plot, it can be observed that the error probability is uniformly distributed and is not a function of SNR value. Furthermore, the maximum value of error percentage is approximately 7.93% and the average is 7.76%. Thus, the probability that ZF augmentation of an unselected discarded path yields smaller PED compared to a selected discarded path is relatively small. The details of BER performance loss and reduction in computational complexity due to this improvement idea will be discussed in Section 3.4.

Thus, to summarize, the selection of relevant discarded paths using the method shown

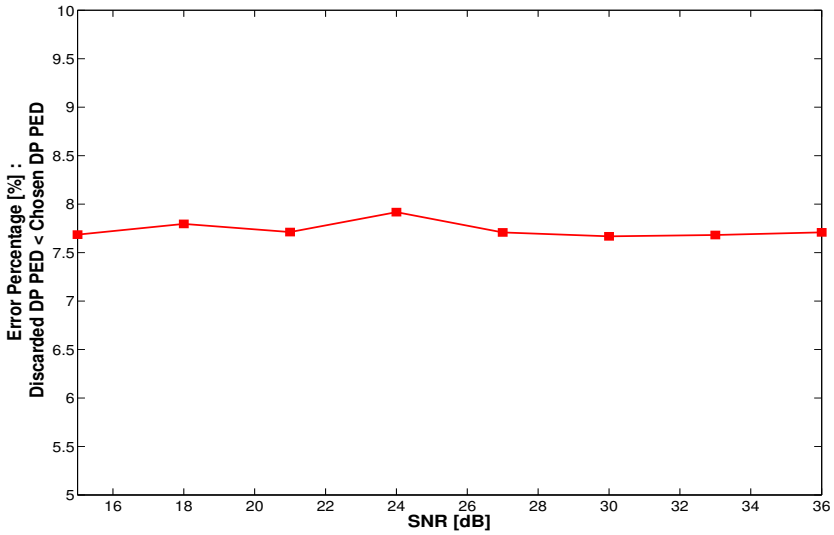

Figure 3.5: Error Percentage Plot for improvement idea 1: Relevant Discarded Paths (DP) Selection.

above only chooses the discarded paths that contain useful information for LLR computation, and abandons the irrelevant discarded paths from further processing. This helps to avoid redundant ZF augmentation and PED computations, and hence reduces the overall computational complexity significantly, at the cost of a marginal loss in the BER performance.

3.3.3 Improvement 2: Last Stage On-Demand Expansion

As described in Section 3.2 above, another major issue with the existing Soft K-Best schemes, that leads to high computational complexity, is the exhaustive expansion of the paths at the last tree level. In other words, the existing Soft K-Best detection schemes exhaustively expands the K best paths at the level $2N_T$ -1 to all possible $K\sqrt{Q}$ paths at level $2N_T$. These Soft K-Best schemes then compute PED for these exhaustively expanded $K\sqrt{Q}$ paths and compare these PEDs to compute the LLR values. Hence, for large constellations and for large values of K, this approach leads to a very large number of PED computations and comparisons. For example, for a 4×4 64-QAM MIMO detector with K=10, this will require computation and comparison of 80 PED values at the last tree level, to compute LLR values for 24 transmitted bits. Hence, this causes either

large processing latency (defined as: the number of cycles required to compute each new set of LLR values) for hardware constrained detectors or large hardware requirements for high-throughput detectors.

However, through careful analysis, it can be observed that within the complete set of $K\sqrt{Q}$ vectors, there are only K distinct $2N_T$ - 1 $\times$ 1 symbol vectors, corresponding to the first $2N_T$ - 1 levels of the detection tree. Hence, the exhaustively extended $K\sqrt{Q}$ paths only improve the quality of LLR for the last $\log_2(Q)/2$ transmitted bits, that correspond to the last level of the detection tree. The exhaustive expansion of the K best paths at the level $2N_T$ - 1 to $K\sqrt{Q}$ paths at the level $2N_T$ does not yield any LLR quality improvement for the first $(2N_T - 1)(\log_2(Q)/2)$ transmitted bits. Thus, the existing approach of exhaustively extending paths at the last level provides minimal BER performance gain, while requiring a large amount of extra resources for implementation.

To resolve these issues, this thesis proposes to use the Last Stage On-Demand Expansion scheme, described below:

Step 1: First, extend the K best paths at tree level $2N_T$ - 1 to exactly K paths at level $2N_T$ using ZF augmentation.

Step 2: Use these K ZF augmented paths at tree level $2N_T$ to fill the MinPED table for the first $(2N_T - 1)(\log_2(Q)/2)$ bits, because for these bits, the K ZF augmented paths at the level $2N_T$ yield the smallest PED values.

Step 3: For the last $(\log_2(Q)/2)$ bits:

- **3.1:** First use the lowest PED ZF augmented path, from the K ZF augmented paths at level $2N_T$, to fill exactly half of the MinPED table for the last $(\log_2(Q)/2)$ bits.
- **3.2:** Then, perform on-demand extension [1] and use at most 2K-1 paths, in the order of ascending PEDs, to fill the remaining half of the MinPED table.

The Last Stage On-Demand Expansion scheme expands the K paths at the tree level $2N_T$-1 on-demand to only the 2K-1 relevant paths at level $2N_T$ for LLR computation purposes. The on-demand nature of path extension ensures that these 2K-1 paths have the lowest PED among the $K\sqrt{Q}$ paths. It also ensures that these paths are expanded and utilized to fill the MinPED table in the order of ascending PED. In other words, if a particular entry in the MinPED table is already filled, there is no need to compare

the current MinPED with the PED for the paths extended afterward. Hence, this avoids PED comparisons at the last level for the purpose of filling the MinPED table and LLR computation.

Thus, to summarize, the Last Stage On-Demand Expansion scheme reduces the number of path extensions from $K\sqrt{Q}$ to only 2K-1 and reduces the number of PED comparisons from $K\sqrt{Q}$ to 0. The details of BER performance and reduction in computational complexity due to this improvement idea will be discussed in Section 3.4.

3.3.4 Improvement 3: Relaxed LLR Computation Scheme

The MKSE detection scheme, presented in [11], performs LLR computation using the 3 step process shown below:

Step 1: First, fill the MinPED table for all $N_T\log_2(Q)$ bits using the last level extension of the K-Best paths.

Step 2: Then, examine each discarded path and compare its PED with the existing MinPEDs to attempt to fill or update the MinPED table.

Step 3: Use the minimum PED data in the MinPED table to compute LLR values (by simply subtracting the MinPEDs) for each of the $N_T\log_2(Q)$ transmitted bits.

As can be observed, the second step in this process requires comparison of each discarded path PED to the current $2N_T\log_2(Q)$ MinPEDs. Hence, this leads to a total of $(2N_T\log_2(Q))(L(K\text{-}1))$ comparisons, which in turn leads to a large computational complexity. However, if an assumption is made that the MinPED values attained by extending the K-Best paths are always smaller compared to the discarded path PEDs, then a large amount of computations can be avoided. This assumption is the basic idea behind the Relaxed LLR Computation Scheme. Hence, the Relaxed LLR Computation Scheme modifies Step 2 in the LLR computation process to be the following:

Step 2 - Modified: For each discarded path, in the ascending order of PEDs, fill an entry in the MinPED table only if it is still empty.

Thus, the Relaxed LLR Computation Scheme does not perform any PED comparisons and fills an entry in the MinPED table, only if that bit was not covered by either the

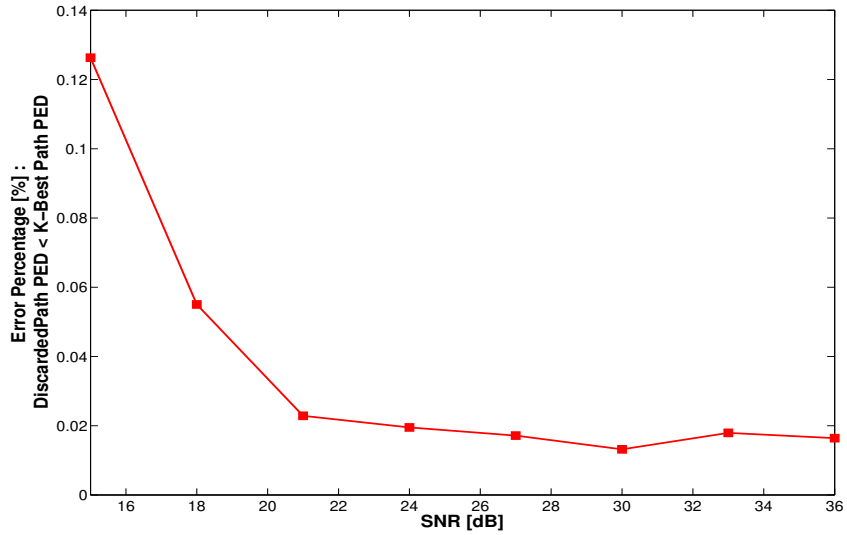

Figure 3.6: Error Percentage Plot for improvement idea 3: Relaxed LLR Computation.

extended K-Best paths or the preceding discarded paths. Hence, this scheme essentially approximates output LLR values, rather than performing exact computation.

However, this assumption is not always valid, and hence might lead to degradation in LLR quality and hence in the resulting detector BER performance. This might happen in the following case: if for a particular bit, a discarded path has a smaller PED compared to the current MinPED value for that bit. This case is classified as an error case in the LLR computation and its corresponding error probability is quantified by simulating a 4×4 64-QAM MIMO detector for a large number of transmitted bits. Fig. 3.6 shows the error percentage plot obtained using these simulations.

From the error percentage plot, it can be observed that the probability of error is higher at low SNR values and reduces significantly for higher SNR values. This can be explained by noticing that at low SNR values, the larger noise might cause good vectors to be discarded on the intermediate stages, which after ZF augmentation to the last level end up having smaller PED values than the K-Best vectors. Note that the maximum value of error percentage is approximately 0.125% and the average is 0.03%.

Thus, it can be concluded that the Relaxed LLR Computation Scheme causes only minor LLR quality degradation, while reducing computational complexity by a large amount. The details of BER performance loss and reduction in computational complexity due to this improvement idea will be discussed in Section 3.4.

3.3.5 Proposed Soft-Output K-Best Detection Scheme

Let us consider a $2N_R \times 2N_T$ real-valued MIMO system with channel characteristic matrix **H**. As described earlier, this can be thought of as a detection problem in a tree with $2N_T$ levels, K lowest PED paths per level and $\sqrt{Q}$ children per path. The task of a Hard K-Best detector is to find the lowest PED $2N_T \times 1$ vector at the last level of the tree. On the other hand, the task of a Soft K-Best detector is to compute LLR values for $(2N_T)(\log_2(Q)/2)$ transmitted bits using all of the existing paths at the last level and the discarded paths from the intermediate levels. Table 3.4 shows the proposed Soft K-Best detection algorithm that computes the LLR values for $(2N_T)(\log_2(Q)/2)$ bits efficiently. Note that the On-Demand Hard K-Best scheme, used as the starting point for the proposed Soft K-Best scheme, was described in detail in [1]. Hence, in this thesis, we just utilize this scheme and do not provide details about its individual steps.

Due to the upper triangular structure of matrix **R**, the Soft K-Best algorithm starts from the last row of the matrix (corresponding to the first level of the detection tree) and goes all the way up to the first row of the matrix, which is the last level of the detection tree. At each level in the detection tree, the algorithm first uses the On-Demand child expansion scheme to find the K-Best paths at that level. The algorithm then uses a three-step process to choose only relevant discarded paths for further processing (ZF augmentation, PED computation and LLR computation) out of the K-1 discarded paths at that level. Disposal of the redundant discarded paths and use of only relevant discarded paths, using this three-step process, reduces the overall computational complexity significantly, at the cost of marginal loss in the BER performance.

The proposed Soft K-Best scheme then uses the Last Stage On-Demand Expansion scheme, presented in Section 3.3.3, to expand the K best paths at level $2N_T$ - 1 to only 2K-1 lowest PED paths at level $2N_T$, rather than expanding them exhaustively to $K\sqrt{Q}$ paths at this level. This reduces the number of path extensions from $K\sqrt{Q}$ to only 2K-1 and reduces the number of PED comparisons from $K\sqrt{Q}$ to 0. The Soft K-Best scheme then uses the Relaxed LLR Computation scheme, presented in Section 3.3.4, to update the MinPED table without performing any PED comparisons. As the very last step, the proposed Soft K-Best detection scheme computes LLR values for the $(2N_T)(\log_2(Q)/2)$ bits, by simply subtracting the MinPED values for each bit. Thus, using the 3 improvement ideas and the On-Demand Hard K-Best scheme, the proposed Soft K-Best detection scheme reduces the overall computational complexity significantly.

Table 3.4: Proposed Soft K-Best Detection Scheme

1) Find the K-Best children of level 1. ($\mathcal{K}_1$).

2) For $l = 2 : 1 : 2N_T - L - 1$
- **2.1)** Find the K-Best paths at level l using the On-Demand child expansion scheme ($\mathcal{K}_l$).

End

3) For $l = 2N_T - L : 1 : 2N_T - 1$
- **3.1)** Find the K-Best paths at level l using the On-Demand child expansion scheme ($\mathcal{K}_l$).
- **3.2)** Populate a Bit Occurrences table for symbols corresponding to levels l -> 1 (2* l *($\log_2$(Q)/2) entries) using the K-Best paths $\mathcal{K}_l$.
- **3.3)** Update the Bit Occurrences table using the discarded paths accumulated from all previous levels ($\mathcal{D}_{l-1 \rightarrow 1}$). Also, copy ($\mathcal{D}_{l-1 \rightarrow 1}$) to ($\mathcal{D}_{l \rightarrow 1}$).
- **3.4)** Examine each of the K-1 discarded paths at the current level ($\mathcal{D}_l$) and select it for further processing (add to $\mathcal{D}_{l \rightarrow 1}$) only if it fills at least one void entry in the Bit Occurrences table.
- **3.5)** Perform ZF augmentation and PED update for each discarded path in $\mathcal{D}_{l \rightarrow 1}$ to level $l + 1$.

End

4) At level $l = 2N_T$
- **4.1)** Sort the discarded paths from ($\mathcal{D}_{2N_T-1 \rightarrow 1}$) in the ascending order of PED.
- **4.2)** Extend the K-Best paths at level 2N_T-1 ($\mathcal{K}_{2N_T-1}$) to exactly K paths at level 2N_T using ZF augmentation.
- **4.3)** Use these K paths at level 2N_T to fill the MinPED table for the first (2N_T -1)($\log_2$(Q)/2) bits.
- **4.4)** To compute LLR for the last ($\log_2$(Q)/2) bits:
  - **4.4.1)** Use the lowest PED ZF augmented path at level 2N_T to fill exactly half of the MinPED table for these bits.
  - **4.4.2)** Perform On-Demand extension of ($\mathcal{K}_{2N_T-1}$) and use at most 2K-1 paths, in the order of ascending PEDs, to fill the rest half of the MinPED table for these bits.

5) Use the sorted 2N_T $\times$ 1 discarded paths from ($\mathcal{D}_{2N_T-1 \rightarrow 1}$) to update the MinPED table using the Relaxed LLR Computation scheme.

6) Compute LLR values using the minimum PED data in the MinPED table for each of the 2N_T($\log_2$(Q)/2) transmitted bits.

3.4 Complexity Analysis and Performance Comparison

In this section, we study the computational complexity of the conventional [26] and the MKSE [11] Soft K-Best schemes, as well as the proposed Soft K-Best detection scheme. The focus of this complexity analysis would be on the parts of the detector that perform soft output (LLR) computation, as we assume that all of these schemes use the same Hard K-Best detection algorithm as their basis. For the complexity analysis of the proposed Soft K-best scheme, we will first start with the computational complexity of the MKSE scheme, and then analyze the incremental complexity reduction due to each of the 3 improvement ideas discussed in Section 3.3 above.

Let us consider a $2N_R \times 2N_T$ real-valued MIMO system with Q-QAM constellation. Also assume that all of the Soft K-Best detection schemes under consideration use the same values for K and L. Table 3.5 shows the computational complexity comparison between different schemes. For comparison metrics, the number of PED updates, PED comparisons and ZF augmentations are considered. However, assessment and comparison of basic computational operations, such as number of additions, multiplications, shifts and comparisons, might assist in a more thorough complexity analysis. Hence, Table 3.6 below shows the complexity comparison between different schemes in terms of these basic computational metrics.

It can be observed that the conventional Soft K-Best scheme requires a large number of PED updates and PED comparisons, but no ZF augmentations. The number of PED updates, PED comparisons and ZF augmentations are increased even further while using the MKSE scheme. This in turn causes a large increase in the number of shift, addition and comparison operations required. The first improvement idea, Relevant Discarded Paths Selection, reduces the number of FC augmentations, PED updates and PED comparisons required, since it only selects and processes relevant discarded paths and abandons further processing of the redundant discarded paths. The Last Stage On-Demand Expansion scheme, presented in Section 3.3.3, expands the K best paths at level $2N_T$-1 to only 2K-1 lowest PED paths at level $2N_T$, rather than expanding them exhaustively to all $K\sqrt{Q}$ possible paths at this level. Due to this reduction in path extension, this scheme further reduces the number of PED updates and comparisons required, and hence causes a reduction in the number of shift, addition and comparison operations required for LLR computation. Finally, the Relaxed LLR Computation scheme eliminates the need for PED comparison between the discarded path PED and the extended K-Best path PED, and

Table 3.5: Computational Complexity of various Soft K-Best schemes

Metric/Scheme	**Conventional** [26]	**MKSE** [11]	**This Work**
PED Computations	$K\sqrt{Q}$	$K\sqrt{Q}$ + $LK(\sqrt{Q}\text{-}1)$	(2K-1) + $((2N_T\text{-}1)*(\log_2(Q)/2))$-(K-1)
PED Comparisons	$K\sqrt{Q}$	$K\sqrt{Q}$ + $((LK(\sqrt{Q}\text{-}1)*(2N_T\log_2(Q))$	(2K-1)
ZF Estimations	0	$LK(\sqrt{Q}\text{-}1)$	$((2N_T\text{-}1)*(\log_2(Q)/2))$-(K-1)

hence reduces the number of PED comparisons required significantly.

Note that, since the proposed Soft K-Best detection scheme uses all of these 3 improvement ideas, it integrates the complexity reduction effect due to each of these ideas. Hence, the proposed Soft K-Best scheme offers a significantly lower computational complexity compared to the MKSE scheme [11]. To demonstrate effectiveness of the proposed Soft K-Best detection scheme, Table 3.7 lists the number of shifts, additions, multiplications and comparisons required by each Soft K-Best scheme for two case scenarios: 4×4 64-QAM MIMO detector with K=10, L=6 and 4×4 16-QAM MIMO detector with K=10, L=6. From these tables, it can be observed and concluded that the proposed Soft K-Best scheme reduces the number of addition, multiplication, shift and comparison operations required to compute LLR values for $N_T\log_2(Q)$ bits, approximately by a factor of 5, compared to the MKSE scheme.

For the purpose of comparing BER performance, the various Soft K-Best MIMO detectors were combined with Convolutional Turbo Coding (CTC) Encoder and Decoder with rate = 1/2, 600 bytes/block and 8 decoder iterations. These combined models were customized for use with a 4×4 MIMO system with 64-QAM and for a 4×4 MIMO system with 16-QAM modulation scheme. Figures 3.7 and 3.8 show the BER curves obtained by simulating these combined models for 16-QAM and 64-QAM case scenarios, respectively.

From the BER curves, it can be noticed that the first improvement idea, Relevant Discarded Paths Selection, causes the largest BER performance degradation compared to the other two improvement ideas. For example, for the 64-QAM case (Fig. 3.8), this

Table 3.6: Computational Complexity (in terms of basic fixed-point mathematical operations) of various Soft K-Best schemes

Metric/ Scheme	**Conventional** [26]	**MKSE** [11]	**This Work**
Shift	$3K\sqrt{Q}$	$17L(K-1) + 3K\sqrt{Q}$	$17[((2N_T-1)*(\log_2(Q)/2))-(K-1)] + 3(2K-1)$
Additions	$6K\sqrt{Q}$	$26L(K-1) + 6K\sqrt{Q}$	$26[((2N_T-1)*(\log_2(Q)/2))-(K-1)] + 6(2K-1)$
Multiplications	$K\sqrt{Q}$	$L(K-1) + K\sqrt{Q}$	$[((2N_T-1)*(\log_2(Q)/2))-(K-1)] + (2K-1)$
Comparisons	$K\sqrt{Q}$	$L(K-1)(2N_T\log_2(Q)) + K\sqrt{Q}$	$(2K-1)$

Table 3.7: Basic Operations Count for various Soft K-Best schemes

Scheme	**Metric**	**Conventional** [26]	**MKSE** [11]	**This Work**
4×4 16-QAM (K=10)	Shift	120	1038	**142**
	Additions	240	1644	**244**
	Multiplications	40	94	**24**
	Comparisons	40	1768	**19**
4×4 64-QAM (K=10)	Shift	240	1158	**261**
	Additions	480	1884	**426**
	Multiplications	80	134	**31**
	Comparisons	80	2672	**19**

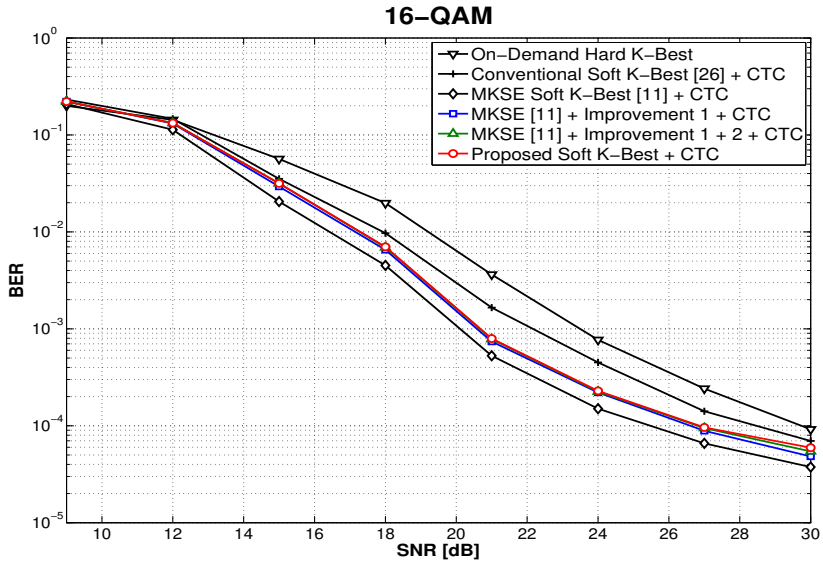

Figure 3.7: BER Performance of Hard K-Best and various Soft K-Best Detection schemes - for 4x4 MIMO system with 16-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations:

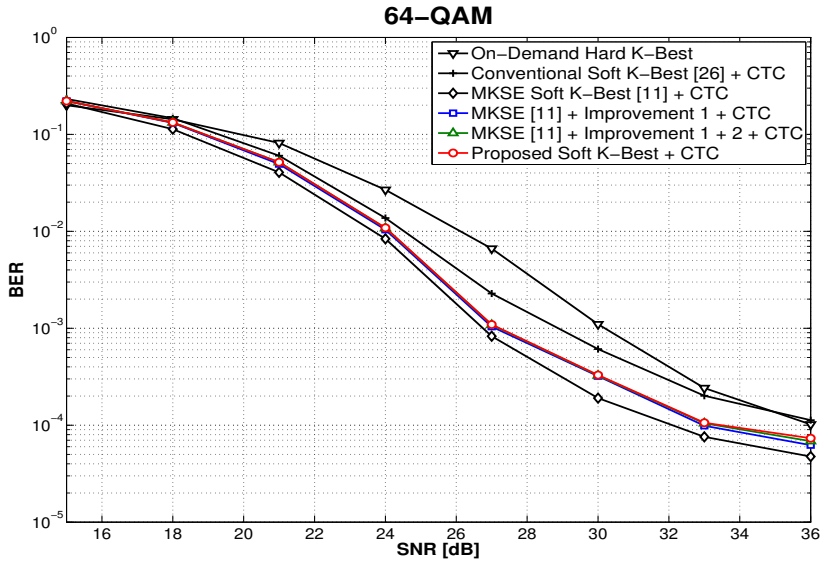

Figure 3.8: BER Performance of Hard K-Best and various Soft K-Best Detection schemes - for 4x4 MIMO system with 64-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations:

improvement idea results in approximately 0.4 dB loss at BER = 10^{-3}, compared to the MKSE detection scheme. As already discussed in detail in Section 3.3.2, this BER degradation is due to the following error case: For a particular bit, ZF augmentation of an unselected discarded path yields smaller final Euclidean distance compared to the final Euclidean distance for ZF extension of a chosen discarded path. This error case causes LLR quality degradation, and hence BER performance degradation, compared to the case where all K-1 discarded paths at each tree level are forwarded for LLR computation.

However, the other two improvement ideas, Last Stage On-Demand Expansion and Relaxed LLR Computation, only cause approximately 0.1 dB and 0.04 dB loss at BER = 10^{-3}, respectively. This is consistent with the observation made from the Error Percentage plot (Fig. 3.6) for the Relaxed LLR Computation scheme, which demonstrates that the probability of error is around 0.03% on average. Thus, in total, for the 64-QAM case, the proposed Soft K-Best detection scheme results in 0.54 dB loss compared to the MKSE scheme presented in [11], while reducing the number of computations required for LLR calculation by a factor of 5. However, it should also be noted that the proposed Soft K-Best scheme improves the BER performance by around 1.7 dB and 2.9 dB compared to the conventional Soft K-Best scheme and the Hard K-Best scheme at BER = 10^{-3}, respectively. Similar observations can also be made from the BER curves for the 16-QAM case (Fig. 3.7) as well.

To summarize, the proposed Soft K-Best detection scheme reduces the computational complexity significantly, while only causing a relatively small BER performance loss compared to the MKSE scheme. Also, the proposed Soft K-Best scheme offers a large performance improvement compared to the conventional Soft K-Best scheme and the Hard K-Best scheme, with comparable complexity. For an example case of 4×4 64-QAM Soft K-Best MIMO detector with K=10, the proposed Soft K-Best scheme reduces the computational complexity at least by a factor of 5, while only causing 0.54 dB loss in BER performance compared to the MKSE scheme. For this example case, the proposed K-Best scheme also offers a large BER performance improvement of 2.9 dB in compared to the Hard K-Best detection scheme.

3.5 Summary

The conventional Soft K-Best detection scheme, which just uses the paths extended exhaustively at the last tree level for LLR computation, poses the major issues of high

computational complexity and lack of significant performance improvement compared to the Hard K-Best detection scheme. To improve the BER performance, a modified MKSE detection scheme is presented in [11] that utilizes the discarded paths from intermediate tree levels, in addition to the exhaustively extended paths at the last level, to compute LLR values. However, this scheme increases computational complexity even further. Thus, using the existing Soft K-Best detection schemes, it is not feasible to design high throughput, area and power efficient MIMO detectors for large constellations and large antenna configurations, envisioned in future WiMAX and LTE systems.

To resolve these issues, this thesis proposed a novel Soft K-Best detection scheme that reduces computational complexity significantly. The key contributions are the three major improvement ideas: Relevant Discarded Paths Selection, Last Stage On-Demand Expansion and Relaxed LLR Computation Scheme. The proposed Soft K-Best detection scheme offers a large improvement in BER performance compared to the conventional Soft K-Best scheme and the Hard K-Best scheme. Moreover, the proposed Soft K-Best scheme offers significant reduction in the computational complexity compared to the MKSE scheme, while not sacrificing any major BER performance gain.

4 VLSI Implementation of a Soft-Output K-Best MIMO Detector

4.1 Introduction

The K-Best algorithm is a well-known approach and an alternative to Sphere Decoding for MIMO detection offering two important advantages: SNR-independent fixed throughput and the ability to define a pipelined architecture due to its feed-forward nature. Spatial Multiplexing (SM) hard and soft-output MIMO detectors based on the K-Best algorithm for 16-QAM systems with 4-transmit and 4-receive antennas (4×4) have been realized for low-order constellations [11, 28]. The 4×4 16-QAM soft-output MIMO detector presented in [11] offers a peak throughput of 107Mbps and requires 97KGates core area in 0.13μm CMOS technology. However, emerging 4G wireless standards, such as IEEE 802.16m and LTE-Advanced, require MIMO systems with high-order constellation schemes (e.g. 64-QAM), large number of antennas (e.g. 4×4) and high data rates (up to 1Gbps).

State-of-the-art 4×4 64-QAM MIMO detectors in the literature consume large silicon area [26, 29], achieve only medium-rate (115Mbps) data throughput rates [26, 29, 30] that falls short of the most aggressive next-generation 4G applications, exhibit variable data throughput that is a function of signal-to-noise-ratio (SNR) and, all implementations to-date reveal some performance loss, [29, 30], relative to a perfect K-Best implementation. One of the major issues that most of these detector designs face is the large number of operations required to compute the Log-Likelihood Ratio (LLR) values for the transmitted bits. Furthermore, as discussed in Chapter 3, for nominal values of K, these designs offer only a marginal performance gain compared to the hard-output K-Best detectors. This issue can be resolved by utilizing discarded paths from intermediate tree levels [11] for the purpose of computing LLR values. However, for large constellations and large antenna configurations (4×4, 64-QAM), a straightforward implementation of a MIMO detector using this idea leads to prohibitive hardware complexity, which leads to either extremely low throughput or large area and power requirements. Hence, an area and

power efficient scalable high-throughput VLSI implementation for a 4×4 64-QAM MIMO detector, envisioned in LTE and WiMAX systems, remains a significant technical challenge especially for high multimedia data rates that are envisaged to approach 1Gbps in future systems.

Note that, in this chapter, we solely focus on the implementation of the K-Best algorithm. Thus it is assumed that the sorted QR-decomposition algorithm according to [31] is applied to the channel matrix before our implemented ASIC to generate the $\mathbf{R}$ matrix using the scaled and decoupled architectures [32], which also applies the generated $\mathbf{Q}$ matrix to the received vector to generate $\mathbf{z}$. Without loss of generality, it is also assumed that i-th row of matrix $\mathbf{R}$ and z_i are normalized by r_{ii}. Therefore, $L_i(\mathbf{s}^{(i)})$ in equation (2.15) can be written as:

$$L_i(\mathbf{s}^{(i)}) = r_{ii}(\bar{z}_i - \sum_{j=i+1}^{2N_T} \bar{r}_{ij} s_j), \tag{4.1}$$

where $\bar{z}_i$, and $\bar{r}_{ij}$ denote the scaled z_i and r_{ij} by r_{ii}, respectively, (i.e. $z_i = \bar{z}_i r_{ii}$, and $r_{ij} = \bar{r}_{ij} r_{ii}$).

4.2 Proposed Soft K-Best Detector - General Architecture Description

As discussed in Chapter 3, the issues mentioned above can be resolved using the novel Soft K-Best detection scheme presented in Table 3.4. This proposed Soft K-Best detection scheme reduces computational complexity significantly, and hence enables hardware implementation of high-throughput low-complexity MIMO detectors. The proposed scheme first uses the basic idea from the MKSE scheme [11], namely utilization of intermediate discarded paths, to achieve a substantially superior BER performance compared to the Hard K-Best detection scheme. It then uses three innovative improvement ideas to reduce the overall computational complexity significantly, with only minimal BER loss. The proposed Soft K-Best detection scheme is shown in Table 3.4 for $2\mathrm{N}_R \times 2\mathrm{N}_T$ Q-QAM real-valued MIMO systems. In this chapter, a customized version of this scheme is used to develop a VLSI architecture of a 4×4 64-QAM Soft K-Best detector.

The proposed architecture with all intermediate parameters for a 4×4 64-QAM Soft-output K-Best MIMO detector with $K = 10$ and Ω={-7,-5,-3,-1,+1,+3,+5,+7} is shown in Fig. 4.1. There are $2N_T = 8$ levels in the tree. The first level of the tree, corresponding to the last row of equation (2.7), opens up all the possible values in Ω, and calculates their

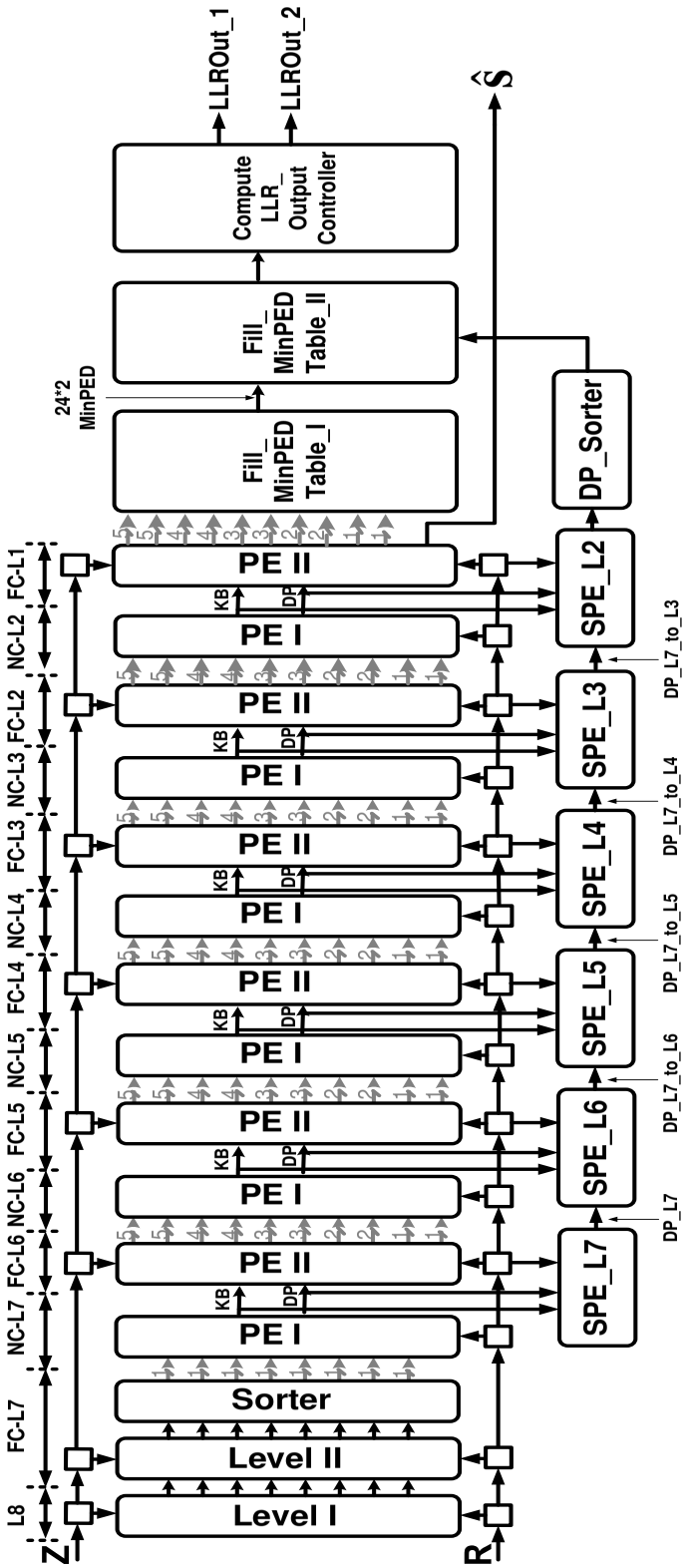

Figure 4.1: The Proposed VLSI architecture of a 4×4 64-QAM Soft-output K-Best MIMO detector with $K = 10$.

corresponding PEDs. The output of this stage is $\mathcal{K}_1$ resulting in $|\Omega| = 8$ PED values, which is done by `Level I` in Fig. 4.1. For each of the nodes in $\mathcal{K}_1$, the first child is found and its PED is updated using the `FC-Block` in `Level II`. The `Sorter` block, shown in Fig. 4.1, then sorts all eight resulting PED values in four cycles, to determine the lowest PED.

The output of the `Sorter` block is the sorted FCs of L7, i.e. $\mathcal{C}_2$, which are all loaded simultaneously to the next stage (i.e., `PE I`) to form $\mathcal{K}_2$. Note that the dashed gray arrows in Fig. 4.1 imply that the data is loaded only once after the completion of the previous stage, and the number on them shows how many cycles after the completion of the previous stage data is loaded. Generally speaking, in each level, one `PE II` block is used to generate and sort the list of all FCs of the current level and one `PE I` block is used to generate the K-Best list of the current level. This fact is denoted in Fig. 4.1 by FC-Li and NC-Li labels under each level's block. The `PE I` block takes the FCs of each level and uses a PED sorter and a core called `NC-Block` in the feedback loop to generate the K-Best paths of that level one-by-one. The `PE II` receives the K-Best candidates of the previous level, one after the other, and generates the FC of each received K-Best candidate one-by-one, and sorts them as they arrive. It finally transfers them to its following `PE I` block for computation of the K-Best paths. Since at the last level only the FC with the lowest PED is of concern, only one `PE II` block is used for the first level (FC-L1), whose output is the solution to the hard detection symbol $\widehat{\mathbf{s}}$. Note that the architecture presented for Hard detection is based on the Hard-output K-Best MIMO detector architecture presented in [1].

The `SPE` and `DP_Sorter` blocks together create the discarded path (DP) datapath that retains selected discarded paths from each tree level, performs ZF augmentation and PED computation for them and sorts them. The `SPE` block samples in the K-Best paths and K-1 discarded paths from the `PE I` block at each level, as well as the accumulated selected discarded paths from all of the previous levels. It then observes the bit occurrences in the K-Best paths and accumulated selected discarded paths, and uses this observation to select and tag only useful discarded paths at the current level. The `SPE` block then computes ZF augmentation for all of the selected discarded paths using the `FC-Block` and updates their PED values. At the end of the DP datapth, the `DP_Sorter` block sorts all of the accumulated selected discarded paths in the order of ascending PEDs to prepare them to be used for LLR computation.

The `Fill_MinPEDTable_I`, `Fill_MinPEDTable_II` and `ComputeLLR_OutputController` blocks perform the task of computing LLR values using the chosen discarded paths and

extended K-Best paths at the last tree level. From equuation 3.1, the process of LLR computation can be performed in two steps. In the first step, the selected discarded paths and extended K-Best paths can be observed to fill the minimum PED (MinPED) table for each transmitted bit. In the second step, this MinPED table can be used to compute LLR values for each bit, by simply subtracting the MinPED values for each bit. The `Fill_MinPEDTable_I` block performs the task of initializing and filling the MinPED table using the $2N_T \times 1$ paths generated through the Last Stage On-Demand Expansion scheme, described in Section 3.3.3. The `Fill_MinPEDTable_II` block then updates this MinPED table using the accumulated selected discarded paths. The final stage of the Soft-output K-Best detector, the `ComputeLLR_OutputController` block, computes LLR values for the 24 transmitted bits and outputs them in parallel using the `LLROut_1` and `LLROut_2` ports. Note that two output ports are required because 24 13-bit LLR values need to be sampled out every 10 clock cycles. In other words, for this particular case, use of two 13-bit output ports help minimize the total number of output pins.

The proposed VLSI architecture for Soft K-Best detector was modeled in Verilog HDL, synthesized using Synopsys Design Compiler and placed and routed using Cadence SoC Encounter/Silicon Ensemble. The RTL and gate level netlists were verified with the golden model generated from the fixed-point MATLAB model. The final Soft K-Best ASIC was fabricated in 0.13 μm IBM 1P8M CMOS technology using ARM standard library cells.

4.3 Proposed Soft K-Best Detector - Input/Output Schedule

4.3.1 Input Schedule

The inputs to the Soft K-Best architecture, shown in Fig. 4.1, are the entries of the $\mathbf{R}$ matrix as well as the $\mathbf{z}$ vector in equation (2.7). Using the RVD scheme, proposed in equation (2.4), the resulting $\mathbf{R}$ matrix after the QR decomposition has symmetry features that can be exploited to advantage, which are explained by the following example [1]. Note that this RVD scheme is a slightly modified version of the conventional RVD scheme, and is more suitable for concurrent computations in the hardware implementation of the Soft K-Best detector. Consider a 4×4, 64-QAM MIMO system (8×8 real domain). The $\mathbf{R}$ matrix using the modified RVD scheme is as follows:

$$\mathbf{R} = \begin{bmatrix} r_{22} & 0 & r_{24} & -r_{23} & r_{26} & -r_{25} & r_{28} & -r_{27} \\ 0 & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} & r_{27} & r_{28} \\ 0 & 0 & r_{44} & 0 & r_{46} & -r_{45} & r_{48} & -r_{47} \\ 0 & 0 & 0 & r_{44} & r_{45} & r_{46} & r_{47} & r_{48} \\ 0 & 0 & 0 & 0 & r_{66} & 0 & r_{68} & -r_{67} \\ 0 & 0 & 0 & 0 & 0 & r_{66} & r_{67} & r_{68} \\ 0 & 0 & 0 & 0 & 0 & 0 & -r_{88} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{88} \end{bmatrix}. \tag{4.2}$$

From this $\mathbf{R}$ matrix, it can be noticed that each pair of consecutive rows share the same entries with a possible sign flip. Therefore, in the VLSI architecture, the input values of two consecutive levels share the r_{ij} values, and hence this RVD scheme reduces both the number of input pads and the required memory to buffer the r_{ij} values. Another implication of this structure is that the first children of the odd rows do not depend on the K-Best list of the preceding even row [1]. For instance, the first child of the fifth row is independent of s_6 and that is due to the fact that $r_{56} = 0$.

Note that in the proposed Soft K-Best detector architecture, it is assumed that the channel is quasi-static and is updated every four channel uses. This implies that the QR decomposition and the input $\mathbf{R}$ entries need to be updated with a proper frequency. In total, there are 16 distinct entries in the $\mathbf{R}$ matrix as well as 32 entries corresponding to the four input $\mathbf{z}$ vectors [1]. Hence, if the input $\mathbf{R}$ matrix and the four input $\mathbf{z}$ vectors are to be sampled in simultaneously and assuming that each entry requires 16 bits on average, the number of input pads for the chip will be 768 pads, which is not feasible for a cost-efficient implementation.

In order to avoid this large number of pads in our ASIC, input $\mathbf{R}$ and $\mathbf{z}$ entries are received one-by-one at the inputs, are buffered and consumed later at the proper time [1]. Moreover, for the proposed Soft K-Best detector, we use a multiplexed scheme where each 16-bit r_{lj} is received in two consecutive clock cycles, i.e., 8 bits per clock cycle. The received entries are buffered at the input and are used at the proper time in the architecture. Using this scheme the number of pads is significantly reduced [1], with the only drawback being a longer initial latency. Fig. 4.2 shows such a scheduling at the input for two consecutive iterations. Note that since we assumed that the channel is updated every four channel

[1] There are four received $\mathbf{z}$ vectors per channel matrix and each $\mathbf{z}$ vector has eight entries corresponding to eight levels of the tree, which results in a total of 32 entries.

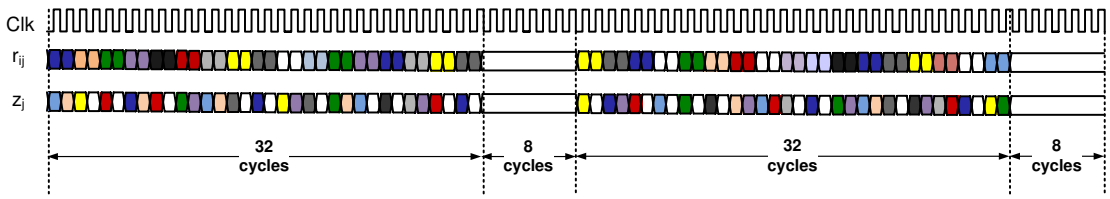

Figure 4.2: Proposed Soft K-Best Input Scheduling - for reading **R** matrix and **z** vectors.

use and since 10 cycles are required for each 4×1 received vector detection, each iteration takes 40 cycles to be processed. In the first 32 cycles the values of r_{ij} and z_j are buffered at the input, and at the end of the 40 cycles the buffered samples are used by the detection core. As shown in Fig. 4.2, each r_{ij} value is read in two cycles (different cycle colors in the figure), while each z_j requires only one clock cycle. This is due to the fact that there are 16 r_{ij} entries per channel matrix and 32 z_j values for 4 consecutive **z** vectors. Note that all of these fixed-point numbers are implemented using the two's complement representation.

4.3.2 Output Schedule

The major outputs of the proposed Soft K-Best detector are the Log-Likelihood Ratio (LLR) values for each transmitted bit, computed using equation (3.1), and the hard detection output symbol $\widehat{\mathbf{s}}$. As shown in Fig. 4.1, the output ports `LLROut_1` and `LLROut_2` are used to output 24 13-bit LLR values for 24 bits ($N_T log_2(Q) = 4 \times log_2(64) = 24$ bits) every $K = 10$ clock cycles. Note again that this is due to the fact that the proposed Soft K-Best detector computes LLR values for a 4×1 complex **z** vector every 10 cycles. Also, in order to attain a cost and area-efficient implementation and reduce the required pad count, only two 13-bit ports, `LLROut_1` and `LLROut_2`, are used to output the 13-bit LLR values.

However, since the Soft K-Best detector needs to output 24 LLR values within 10 clock cycles using only two 13-bit ports, they need to be multiplexed such that 2 LLR values can be sampled out in a single clock cycle, i.e. at the positive and negative clock edges. Fig. 4.3 shows the output scheduling for the LLR output ports. As shown in this figure, each of the `LLROut_1` and `LLROut_2` ports output 12 LLR values within the first 6 clock cycles, with a new 13-bit LLR value sampled out every half cycle (indicated by toggling and using different colors every half cycle in the figure). Table 2 shows the exact schedule of the output LLR values for a 4×1 transmitted symbol vector $\mathbf{s} = [s_1\ s_2\ s_3\ s_4]^T$. Note

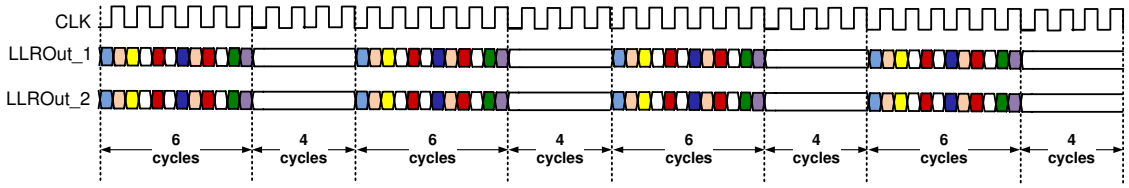

Figure 4.3: Proposed Soft K-Best Output Scheduling - for writing out LLR values.

Table 4.1: Output Scheduling for LLR values for transmitted symbol vector ($\mathbf{s}$).

Clock Cycle/Edge:	LLROut_1:	LLROut_2:
Cycle 1 - posedge (↑)	$s_1[5]$	$s_1[4]$
Cycle 1 - negedge (↓)	$s_1[3]$	$s_1[2]$
Cycle 2 - posedge (↑)	$s_1[1]$	$s_1[0]$
Cycle 2 - negedge (↓)	$s_2[5]$	$s_2[4]$
Cycle 3 - posedge (↑)	$s_2[3]$	$s_2[2]$
Cycle 3 - negedge (↓)	$s_2[1]$	$s_2[0]$
Cycle 4 - posedge (↑)	$s_3[5]$	$s_3[4]$
Cycle 4 - negedge (↓)	$s_3[3]$	$s_3[2]$
Cycle 5 - posedge (↑)	$s_3[1]$	$s_3[0]$
Cycle 5 - negedge (↓)	$s_4[5]$	$s_4[4]$
Cycle 6 - posedge (↑)	$s_4[3]$	$s_4[2]$
Cycle 6 - negedge (↓)	$s_4[1]$	$s_4[0]$

that this pattern and schedule is repeated every 10 clock cycles.

4.4 Proposed Soft K-Best Detector - Detailed VLSI Architecture

This section provides details about the architecture and functionality of the blocks used in the proposed Soft K-Best MIMO detector. A detailed description of some common sub-blocks used throughout the architecture will be provided first, and will be followed by a description of the major functional blocks. Note that since the proposed Soft K-Best architecture uses a few sub-blocks from the Hard K-Best architecture presented in [1], the following subsections will provide detailed information for only the blocks used specifically for Soft K-Best detection. The details about blocks specific to the Hard K-Best detector have been included in Appendix B.

4.4.1 Common Sub-Block Description

Multiplication (MU)

The overall Soft K-Best detector architecture involves two types of multiplications. The multiplication of $\bar{z}_i \times r_{ii}$ and $s_i \times r_{ij}$ (see equation (4.1)). The first multiplication is realized using 13-bit×13-bit multipliers [1]. However, the second multiplication can be implemented using an alternative architecture, which takes less area and has a much smaller critical path. This architecture, called `MU`, is shown in Fig. 4.4 using a few multiplexers (MUX) and adders (SUM). In this figure, the numbers on the right represent the bit location in s_j (i.e. s_j can be represented with 4 bits), "$\ll n$" represents n shifts to the left and the tiny bubble "$-\circ$" denotes the negation operation. Note that all of the fixed-point numbers use the two's complement representation.

Note that this simpler implementation of the multiplication operation, $s_i \times r_{ij}$, is possible due to the fact that the values of s_i are drawn from a finite pre-determined odd-integer set $\Omega = \left\{(-\sqrt{Q}+1), \cdots, -1, +1, \cdots, (+\sqrt{Q}-1)\right\}$, where Q is the constellation size. The structure of the `MU` block is such that the adder always produces one of the odd multiples of r_{ij} (i.e., r_{ij}, $3r_{ij}$, $5r_{ij}$, $7r_{ij}$), depending on the value of s_i. The MUXs before the adder perform the function of selecting correct operands for the adder. The first operand of the adder can be either of r_{ij} or -r_{ij}, and the second operand of the adder can be any of 0, $2r_{ij}$, $4r_{ij}$ or $8r_{ij}$. The MUX in the last stage, after the adder, utilizes the Most Significant Bit (MSB) of s_j to make the decision on whether or not to negate the output $s_i \times r_{ij}$.

This way of implementation of the multiplication operation is much faster than a normal multiplier implementation. As will be discussed in Section B.5, the motivation for designing the `MU` block is due to the fact that this multiplication lies in the critical path of the architecture, which is on a feedback path [1]. Since the fine-grained pipelining technique cannot be used in the feedback path to improve the overall throughput, an efficient implementation of the multiplier using this scheme is critical to enhance the maximum operating frequency for the Soft K-Best detector.

MapBinaryToConstellation blocks

As discussed in Section 4.2, the proposed MIMO detector uses real-domain implementation of the K-Best detection algorithm. This in turn implies that the input and intermediate data are real signed numbers that are represented in binary format using two's complement representation. However, for LLR computation purposes, the intermediate data

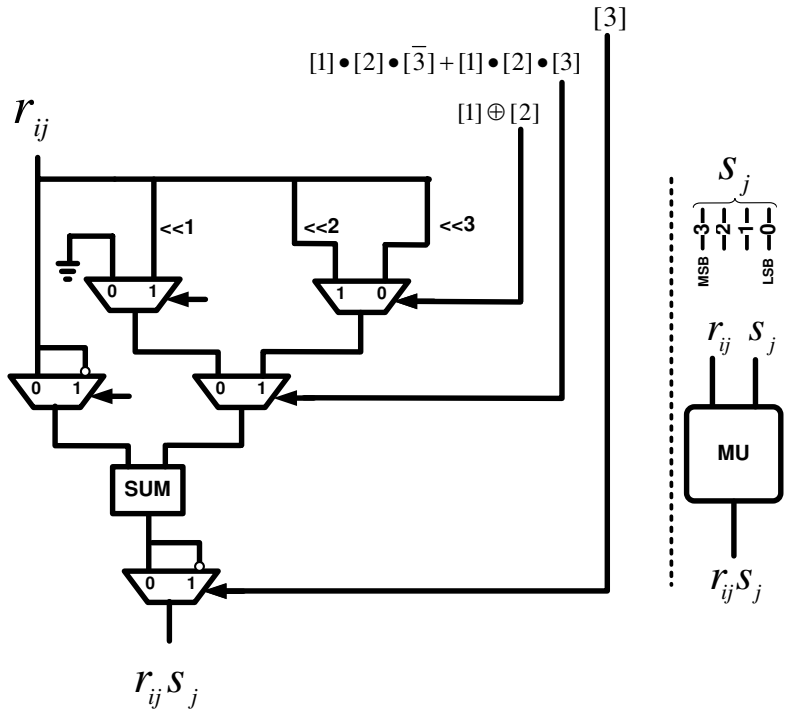

Figure 4.4: Alternative architecture for Multiplication - the MU block. Note that here [x] refers to the xth bit of S_j.

(specifically the partial vectors at the intermediate levels) need to be utilized as complex numbers, whose binary representation is attained by mapping them onto a gray coded 64-QAM constellation. This is accomplished in the proposed MIMO detector using the `MapBin2Constellation_Re` and `MapBin2Constellation_Im` blocks.

Fig. 4.5 shows a 64-QAM constellation, with a gray coded binary representation of each complex point in the constellation. From this constellation, the mapping from binary two's complement representation of real and imaginary symbols to the constellation representation can be derived as shown in Table 4.2. In the proposed Soft K-Best detector, the `MapBin2Constellation_Re` and `MapBin2Constellation_Im` blocks implement this mapping using combinational circuitry for real and imaginary elements, respectively. For example, the partial real-valued vector {1001 1101 0101 1111 0011 0101} in two's complement representation corresponds to partial vector {-7-3i, 5-1i, 3+5i}. Using the `MapBin2Constellation_Re` and `MapBin2Constellation_Im` blocks, this vector will be mapped to {000 111 101 110 111 001} in the constellation representation.

Mapper and Limiter

Once $s_i^{[0]}$ was calculated based on $\bar{L}_i(\mathbf{s}^{(i)}) = L_i(\mathbf{s}^{(i)})/r_{ii}$, the first child needs to be calculated by mapping the value $s_i^{[0]}$ to the nearest point in Ω (slicing). This is done in two consecutive

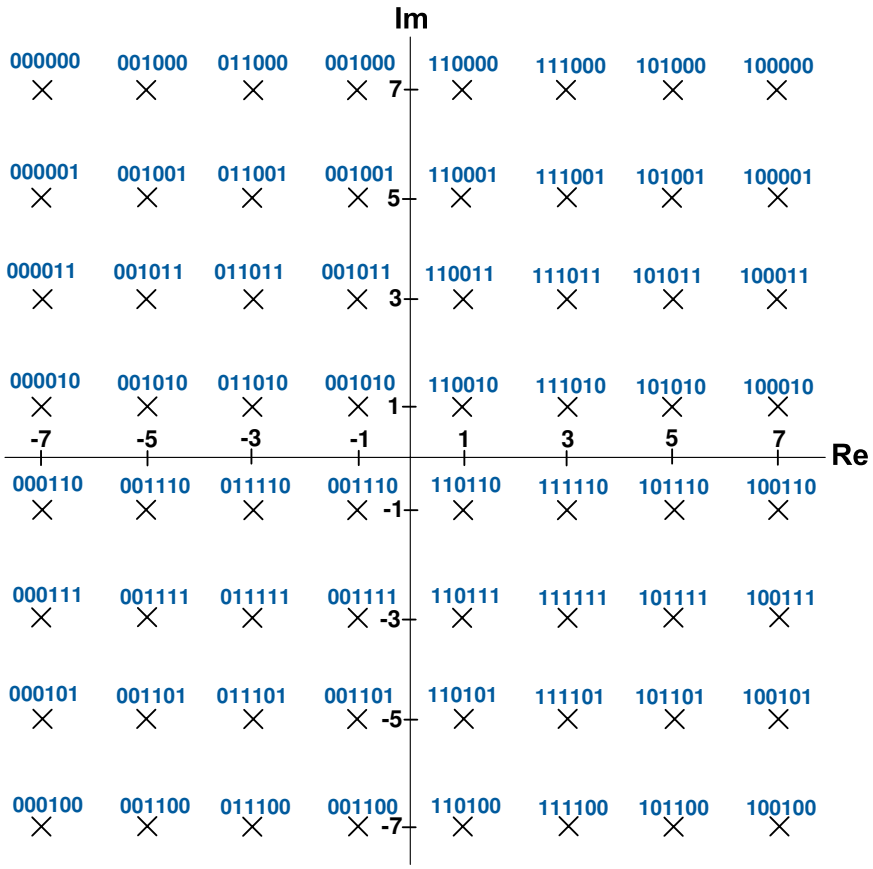

Figure 4.5: A 64-QAM Constellation, with Gray Coded binary representation of each Complex point.

stages. First $s_i^{[0]}$ is mapped to its nearest odd integer number ($\tilde{s}_i^{[0]}$) using the `Mapper` block and then if $\tilde{s}_i^{[0]}$ is outside the allowed boundary of Ω, it will be bounded by the `Limiter` block to generate $s_i^{[1]}$. The process of mapping to the nearest odd integer number is implemented by the `Mapper` block shown in Fig. 4.6 [1]. The reason behind this implementation is the fact that the nearest odd number to $s_i^{[0]}$ is $2\lfloor \frac{s_i^{[0]}}{2} + 0.5 + 0.5 \rfloor - 1 = 2\lfloor \frac{s_i^{[0]}+1}{2} + 0.5 \rfloor - 1$, where $\lfloor \cdot \rfloor$ represents the truncation operation.

The process of limiting the value in the predefined range is done through the `Limiter` block. In other words, if $\tilde{s}_i^{[0]}$ is outside the boundaries of Ω, the `Limiter` block guarantees that the upper/lower bounds (e.g., +7/-7 in 64-QAM) are chosen as the selected points in Ω. The `Limiter` block is shown in Fig. 4.7 with examples on the action taken on $\tilde{s}_i^{[0]}$ to determine $s_i^{[1]}$ for three different cases.

Table 4.2: Binary two's complement to Constellation representation Mapping table

	Symbol	Binary Two's Complement Representation	Constellation Representation
Re Symbols	-7	1001	000
	-5	1011	001
	-3	1101	011
	-1	1111	010
	1	0001	110
	3	0011	111
	5	0101	101
	7	0111	100
Im Symbols	-7	1001	100
	-5	1011	101
	-3	1101	111
	-1	1111	110
	1	0001	010
	3	0011	011
	5	0101	001
	7	0111	000

4.4.2 Soft PE (SPE)

`SPE` is a general block used for all levels from level 7 to level 2. In general, at each level, this block receives the list of K-Best paths (KB) and discarded paths (DP) at that level, in addition to the chosen discarded paths (ChosenDP) accumulated from all of the previous level `SPE` blocks. As its output, this block generates a list of ZF augmented ChosenDPs and the discarded paths selected from the current tree level. Thus, in general, an `SPE` block performs the tasks of selecting relevant discarded paths at the present level and computing their ZF augmentation. In other words, an `SPE` block simply implements steps `3.2` to `3.5` of the proposed Soft K-Best algorithm, shown in Table 3.4.

The overall architecture of the `SPE` block is shown in Fig. 4.8 for Level k (Lk). As shown, the overall architecture consists of three major sub-blocks: `Note Bit Occurrences (NBO)`, `Tag Discarded Paths (TDP)` and `FCBlock`. Fig. 4.9 shows the timing diagram for the input, output, as well as the internal signals of the `SPE` block. Note that this timing diagram is shown for only one set of data (a single $\mathbf{z}$ vector), and the portions of each signal corresponding to this data set are highlighted. Also, note that each of these signal

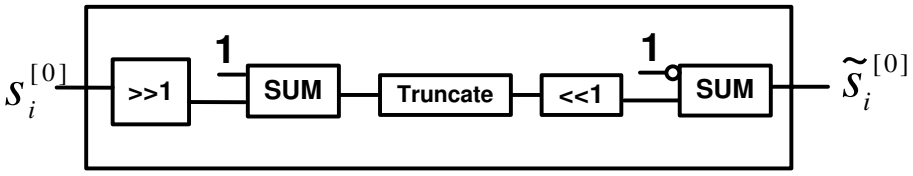

Figure 4.6: The architecture of the `Mapper`, where $\tilde{s}_i^{[0]} = 2\lfloor \frac{s_i^{[0]}+1}{2} + 0.5 \rfloor - 1$.

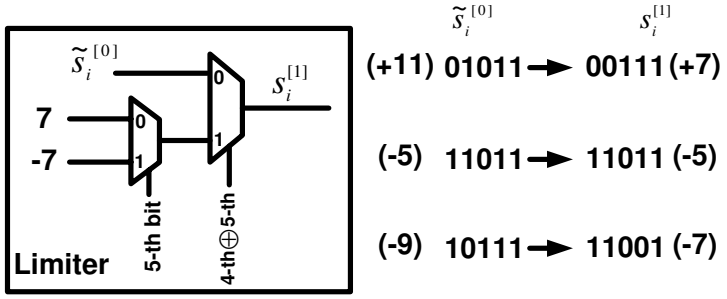

Figure 4.7: The architecture for the `Limiter` block.

patterns repeat every $K = 10$ clock cycles in the proposed Soft K-Best detector.

As shown in Fig. 4.9, the 10 K-Best paths are sampled out from a `PE I` block at the positive edges (posedge) of cycles 1 to 10. Also, the accumulated ChosenDPs are ready at the `SPE` block inputs at posedge cycle 9. The `NBO` block first uses these K-Best paths and ChosenDPs to fill a bit occurrence table at the current level. Since, this last K-Best path is sampled out at posedge cycle 10, the bit occurrence table is finalized and is available at the `NBO` block outputs at posedge cycle 11. Furthermore, due to the deeply pipelined architecture of the Soft K-Best detector, the discarded paths (DP) at the current level need to be sampled out from a `PE I` block in the same cycle as the last K-Best path. In other words, the `TDP` block samples in all $K - 1 = 9$ discarded paths at the current level at posedge cycle 10. The `TDP` block then uses the bit occurrence table to observe each of the 9 discarded paths and to tag a DP for further processing only if it fills at least one of the void entries in the bit occurrence table. The `FCBlock` then samples in only tagged discarded paths and accumulated ChosenDPs, from posedge cycle 13 to posedge cycle 22, and computes their ZF augmentation [16] and updated PED values. Since, the latency of the `FCBlock` is 4 clock cycles, the ZF augmented chosen discarded paths are sampled out from the `FCBlock` at posedge cycle 17 to 26. The following sections will provide detailed description of the `NBO`, `TDP` and `FCBlock` blocks.

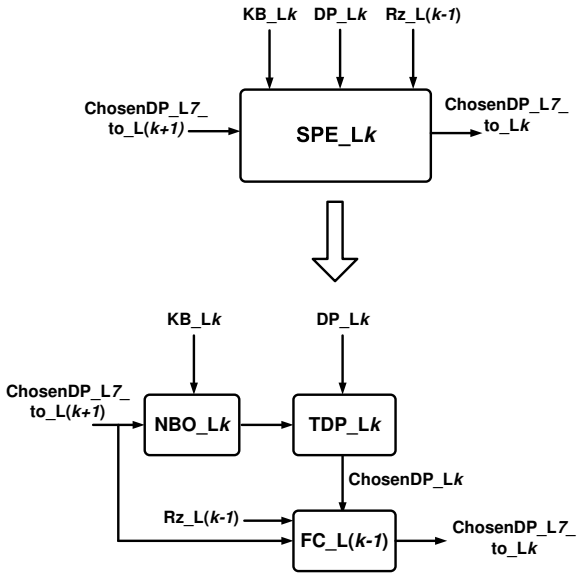

Figure 4.8: Generic overall architecture of an SPE block.

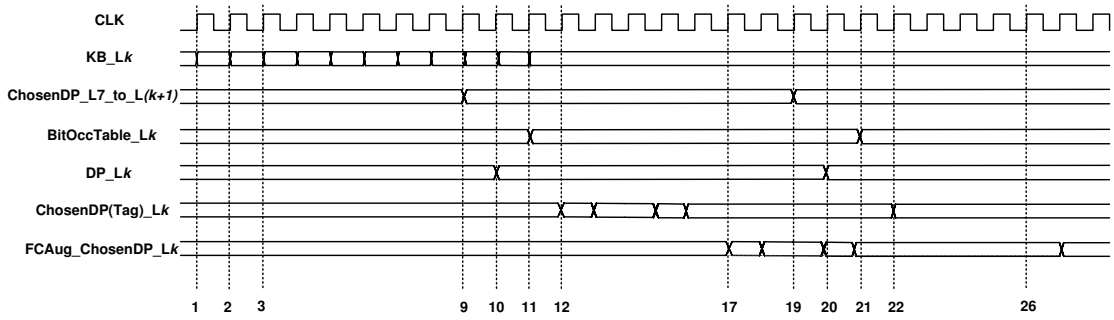

Figure 4.9: Timing Diagram for input, output and intermediate signals of the SPE block.

4.4.3 NBO and TDP Blocks

The NBO and TDP blocks together implement the improvement idea: Relevant Discarded Paths Selection, presented in Section 3.3.2, that has also been encapsulated in steps 3.2 to 3.4 of the proposed Soft K-Best algorithm. The NBO block samples in K-Best paths from the current level and accumulated ChosenDPs from the previous levels. These K-Best and discarded paths are then observed and utilized to fill the bit occurrence table at the current level. Note again that for Level k, a bit occurrence table is simply a table of dimensions $2\times(N_T-k-1)(log_2(Q)/2)$, that keeps track of occurrences of "0" and "1" values for $(N_T-k-1)(log_2(Q)/2)$ bits ($(N_T-k-1)\times 1$ real-valued vector) in the K-Best paths and accumulated ChosenDPs.

Fig. 4.10 shows the overall architecture of the Note Bit Occurrences (NBO) block for Level 6 (NBO_L6). As shown, the NBO block consists of two major parts: KB_Datapath

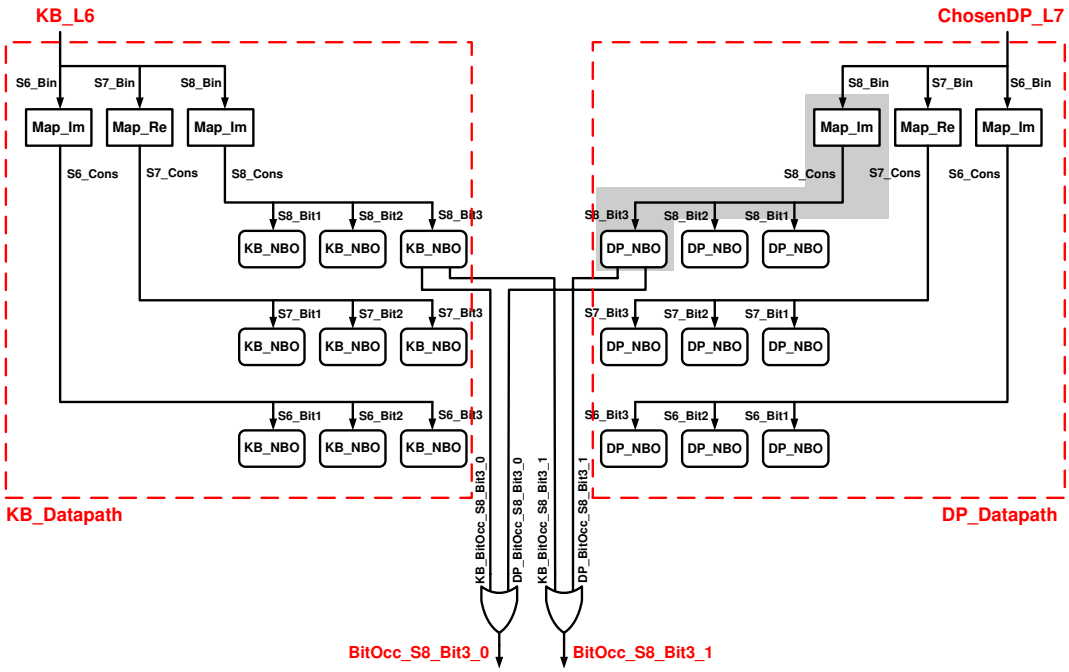

Figure 4.10: Overall architecture of the `Note Bit Occurrences (NBO)` block - with critical path highlighted.

and `DP_Datapath`. These `KB_Datapath` and `DP_Datapath` parts independently fill the bit occurrence tables for K-Best paths and accumulated ChosenDPs, respectively, in a completely parallel manner. The logical `OR` operation at the `NBO` block output then merges these two independent bit occurrence tables. In each of the datapaths, the process of filling the bit occurrence table is carried out in two steps. In the first step, the input symbol in binary two's complement representation is converted to its constellation representation. The second step then uses this constellation representation to update the bit occurrence table using the blocks `KB_NBO` and `DP_NBO`.

Figures 4.11 and 4.12 show the architecture of the `KB_NBO` and `DP_NBO` sub-blocks, respectively. In both of these blocks, logical `XNOR` operation with "0" or "1" is used to determine whether the current bit is "0" or "1". The `KB_NBO` sub-block uses a special mechanism to reset the KB bit occurrence table, using the logical `OR` and `AND` operations. This special reset mechanism is required since the `KB_NBO` sub-block needs to be active for all $K = 10$ clock cycles and there are no idle cycles are available for register reset. Hence, `KB_NBO` needs to be reset in the same clock cycle when the first K-Best path is processed. Contrary to this, the `DP_NBO` sub-block does not need this special reset mechanism and can be reset by simply resetting the registers. However, the `DP_NBO` does require a tagging

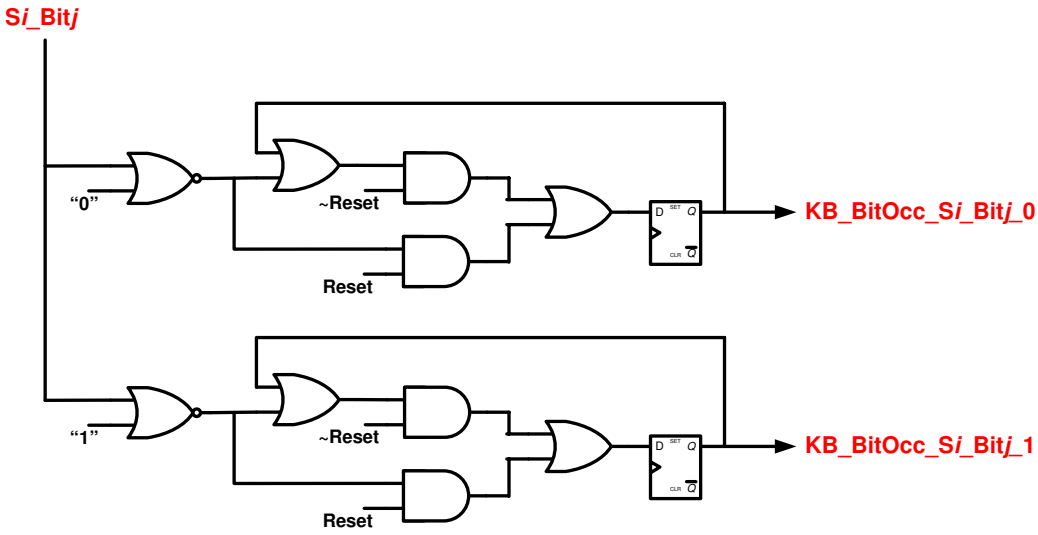

Figure 4.11: Architecture of the KB_NBO sub-block.

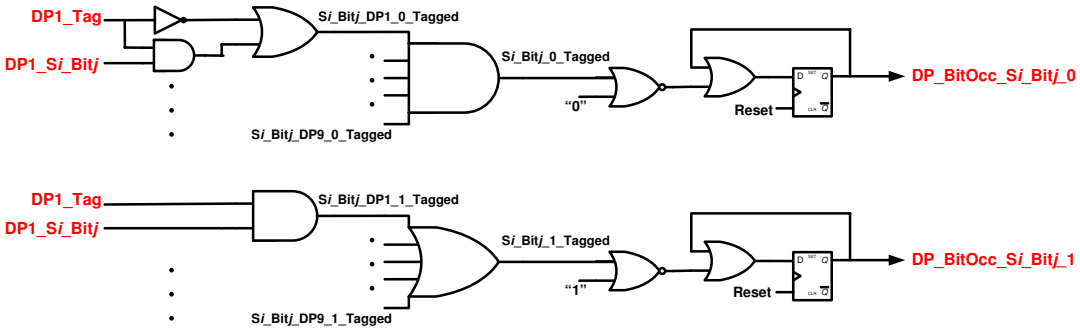

Figure 4.12: Architecture of the DP_NBO sub-block.

mechanism at its inputs. This mechanism is required to recognize and tag the bits that belong to the selected relevant discarded paths. The overall critical path of the NBO block is also shown in Fig. 4.10, which consists of MapBin2Constellation_Re/Im and DP_NBO sub-blocks.

The TDP block utilizes the bit occurrence table prepared by the NBO block to observe and tag each of the 9 discarded paths at the current level. In other words, a discarded path will be selected for further processing and will be tagged "1" only if it fills a void entry in the present bit occurrence table. The overall architecture of the TDP block is shown in Fig. 4.13. The "Load_BOTable" signal is used to load the bit occurrence table into the appropriate DP_TDP sub-blocks, from the NBO block outputs. Once the bit occurrence table is loaded, the large MUX at the TDP block input is used to choose the current discarded path to be processed. The combination of MapBin2Constellation_Re/Im and DP_TDP sub-blocks are then used compute the tags for individual bits. The overall tag for the

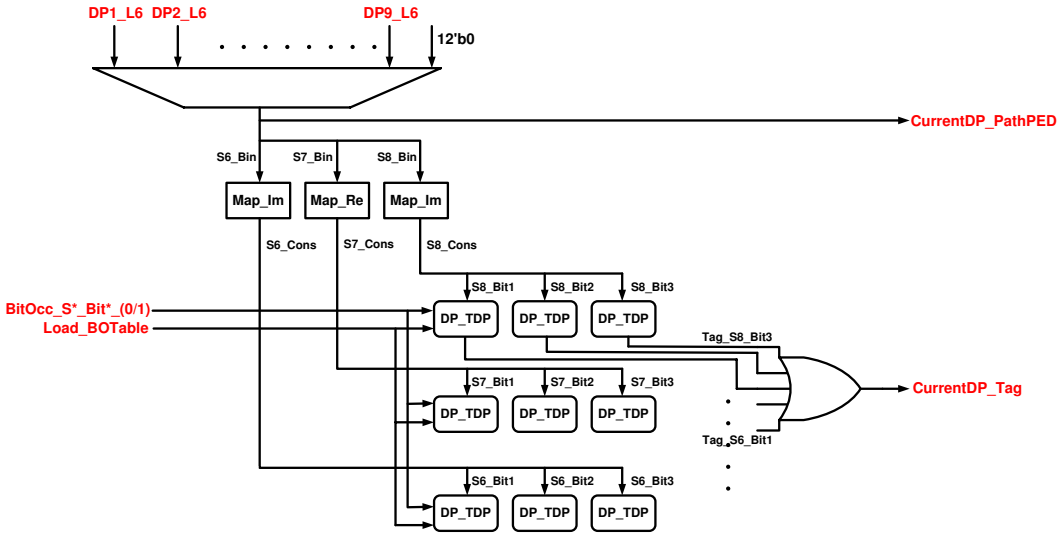

Figure 4.13: Overall architecture of the `Tag Discarded Paths (TDP)` block.

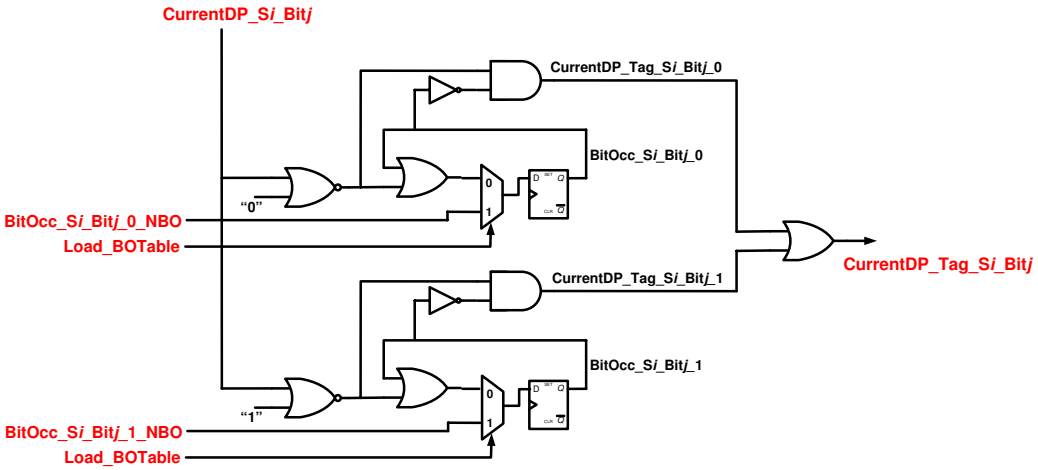

Figure 4.14: Architecture of the `DP_TDP` sub-block.

current discarded path, "CurrentDP_Tag" is computed by using the logical `OR` operation, as shown in Fig. 4.13. Fig. 4.14 shows the architecture of the `DP_TDP` sub-block for a single bit. It uses similar circuitry to determine whether the current bit is "0" or "1" and to update the bit occurrence table. Additional circuitry has been added to load the input bit occurrence table and to compute the tag for the current bit. Note that the complete process of selecting and tagging a discarded path is performed in a single clock cycle, and hence the critical path for the overall `TDP` block contains all of the sub-blocks from the DP inputs to the "CurrentDP_Tag" output signal.

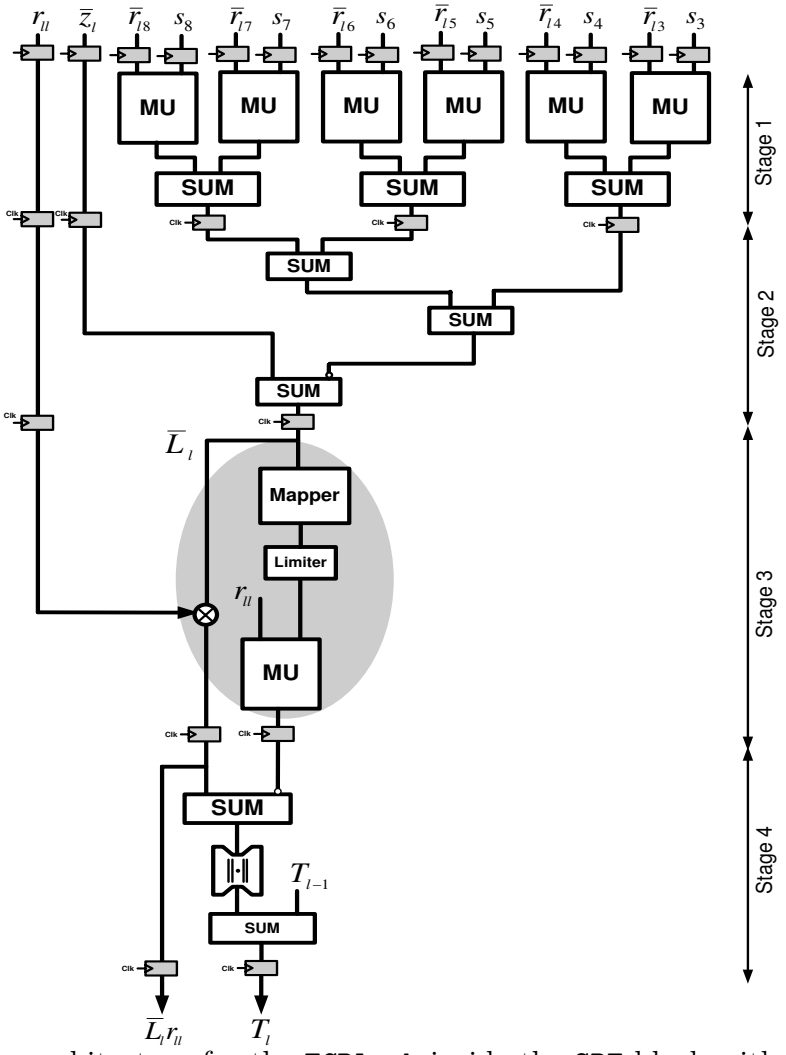

Figure 4.15: The architecture for the `FCBlock` inside the `SPE` block with the critical path highlighted.

4.4.4 FCBlock

The main tasks of the `FCBlock` are to calculate the $\bar{L}_i$ value, to find the first child of the current parent based on the calculated $\bar{L}_i$, and to update its PED value. The proposed architecture for the `FCBlock` is shown in Fig. 4.15. In order to increase the total throughput in the architecture, pipelining has been used by the introduction of the registers on all the forward paths. It increases the throughput at the cost of additional latency in the circuit. The complete `FCBlock` block incurs a latency of 4 clock cycles.

The proposed architecture for the `FCBlock` consists of four pipeline levels. In the first pipeline level (shown in Fig. 4.15), there are six `MU` blocks. However, depending on the `SPE` block in which it is used, only part of these `MU` blocks are used. For instance, for the

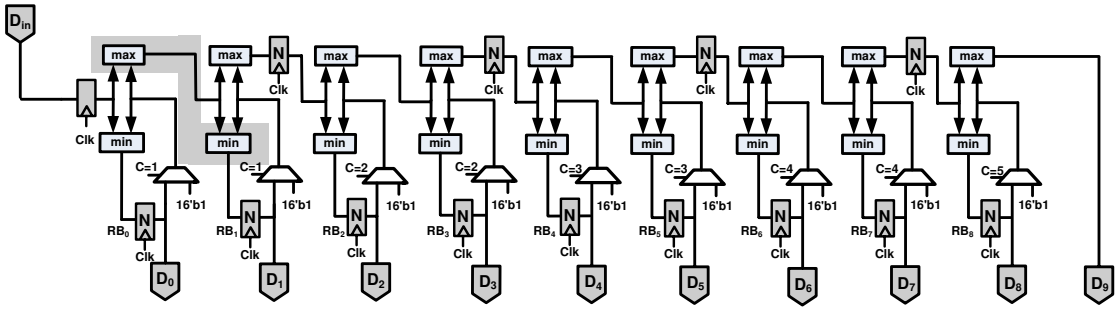

Figure 4.16: The architecture for the DP_Sorter block with the critical path highlighted.

SPE blocks of stage SPE_L6 and SPE_L5, only the first two MUs are implemented. For SPE blocks of stage SPE_L4 and SPE_L3, the first four MUs and finally for SPE blocks of stage SPE_L2 and SPE_L1, all of the six MUs are implemented [1].

The first two levels of the architecture calculates the $\sum_{j=i+1}^{2N_T} \bar{r}_{ij} s_j$ in equation (2.15) and the value of $\bar{L}_i$. Using the $\bar{L}_i$, in Level 3, the first child is calculated using the Mapper and Limiter sub-blocks in the FCBlock. Finally the blocks in Level 4 calculate the PED value of the announced first child. Thus, the FCBlock computes ZF augmentation for the accumulated ChosenDPs and the selected discarded paths at the current level. The output ZF augmented discarded paths and their tags are transferred to the NBO block of the next level.

4.4.5 DP_Sorter

As discussed in Section 4.4.2, the SPE blocks retain selected discarded paths from each tree level, as well as perform ZF augmentation and PED computation for them. The output of the SPE_L2 block consists of only selected discarded paths of dimensions $2N_T \times 1$, that have been fully ZF augmented to the last tree level. However, the Fill_MinPEDTable_II block requires the input discarded paths to be sorted and to be supplied in pairs (input two discarded paths per clock cycle). Hence, the purpose of the DP_Sorter block is to sort all of the accumulated selected discarded paths in the order of ascending PEDs.

The architecture of the DP_Sorter block is shown in Fig. 4.16, where D_{in} is the input port and D_0-D_9 are the output ports. The FCBlock at the output of SPE_L2 computes the first child and updated PED values for the selected discarded paths. These discarded paths and their corresponding PED values are transferred to DP_Sorter block sequentially, which then sorts these paths and calculated PED values as they arrive. In the proposed architecture for DP_Sorter, the sorted PEDs are stored in the register banks, depicted

by N-bit registers in Fig. 4.16 and denoted by $\texttt{RB}_0 - \texttt{RB}_8$. At every clock cycle, two register banks are updated at the same time. This is because of the fact that the registers on the upper part of the sorter are located in every other stage. This is required to guarantee the correct functionality of the `Fill_MinPEDTable_II` block. The functionality of the `DP_Sorter` is such that the larger values are shifted to the right while the smaller values are shifted to the left. Once the last element (10-th element in 64-QAM) enters the sorter, it updates the first two register banks, thus the first two are guaranteed to have the two smallest PED values. Therefore, at the next clock cycle, they can be transferred to the following `Fill_MinPEDTable_II` block. After the second clock cycle, the next two register banks are updated and they are also ready to be transmitted. Therefore, the sorted discarded paths and their PED values are transferred to the next block on a pair-by-pair basis.

Note also that once the last element comes in and the first two register banks are sent to the next stage, the internal min/max functions should be initialized to the highest positive number to avoid the comparison between the first element of the next iteration and the last element of the current iteration. This is implemented through the introduction of a MUX before the min/max functions and is controlled by a control signal (signal `C` in Fig. 4.16). This control signal is incremented every clock cycle and is initialized to zero at the end of each iteration. For instance, when `C=1`, the input to the first two min/max functions are initialized to the largest 16-bit number (i.e., `16'b1`), thus once the first discarded path of the next iteration comes in, it would not be compared with the previous stored values in the register banks from the previous iteration.

4.4.6 Fill_MinPEDTable_I

The `Fill_MinPEDTable_I` is the first block, in the set of 3 serial blocks that perform LLR computation. This block performs the task of initializing and filling the minimum PED (MinPED) table using the last level extension of K-Best paths. In other words, the `Fill_MinPEDTable_I` block samples in ZF augmented 10 $2N_T \times 1$ K-Best paths, processes them further and utilizes them to generate the MinPED table and the corresponding MinPED tag values. Recall that a MinPED table is simply a table of dimensions $2 \times N_T log_2(Q)$, that stores minimum path PED values for $N_T log_2(Q)$ transmitted bits. The detailed definition and description of the MinPED table was provided in Section 3.3.2. The `Fill_MinPEDTable_I` block populates the minimum PED table using the improvement

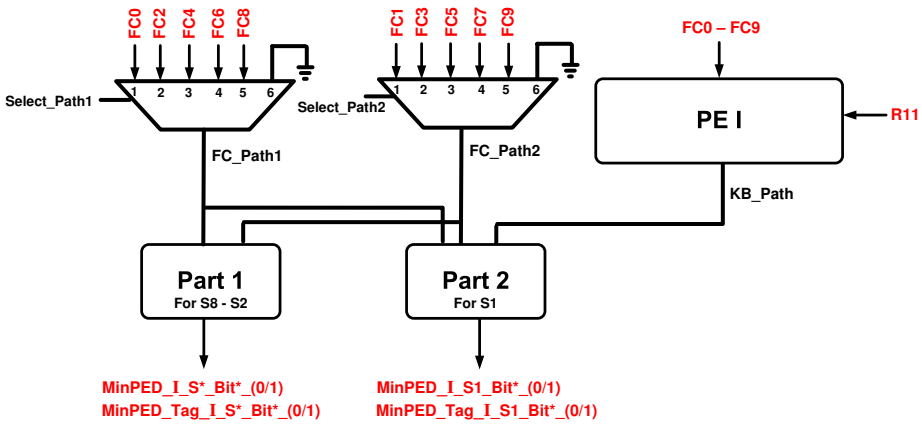

Figure 4.17: Overall Architecture of the `Fill_MinPEDTable_I` block.

idea: Last Stage On-Demand Expansion, presented in Section 3.3.3. In other words, this block implements steps `4.2` to `4.4`, also shown below, of the proposed Soft K-Best algorithm.

Step 4.2: Extend the K best paths at tree level $2N_T$ - 1 to exactly K paths at level $2N_T$ using ZF augmentation.

Step 4.3: Use these K ZF augmented paths at tree level $2N_T$ to fill the MinPED table for the first $(2N_T - 1)(\log_2(Q)/2)$ bits, because for these bits, the K ZF augmented paths at the level $2N_T$ yield the smallest PED values.

Step 4.4: For the last $(\log_2(Q)/2)$ bits:

4.4.1: First use the lowest PED ZF augmented path, from the K ZF augmented paths at level $2N_T$, to fill exactly half of the MinPED table for the last $(\log_2(Q)/2)$ bits.

4.4.2: Then, perform on-demand extension [1] and use at most 2K-1 paths, in the order of ascending PEDs, to fill the remaining half of the MinPED table.

Fig. 4.17 shows the overall architecture of the `Fill_MinPEDTable_I` block. As shown, this block consists of 3 major sub-blocks. The `PE I` sub-block used in this block is identical to the one used in the hard K-Best datapath, as shown in Fig. 4.1. It receives the sorted list of ZF augmented 10 K-Best paths at level $2N_T - 1$ and uses an On-Demand path extension scheme [27] to generate a list of K-Best paths at Level $2N_T$. Since the On-Demand expansion scheme uses Schnorr-Euchner (SE) enumeration, the 10 K-Best paths

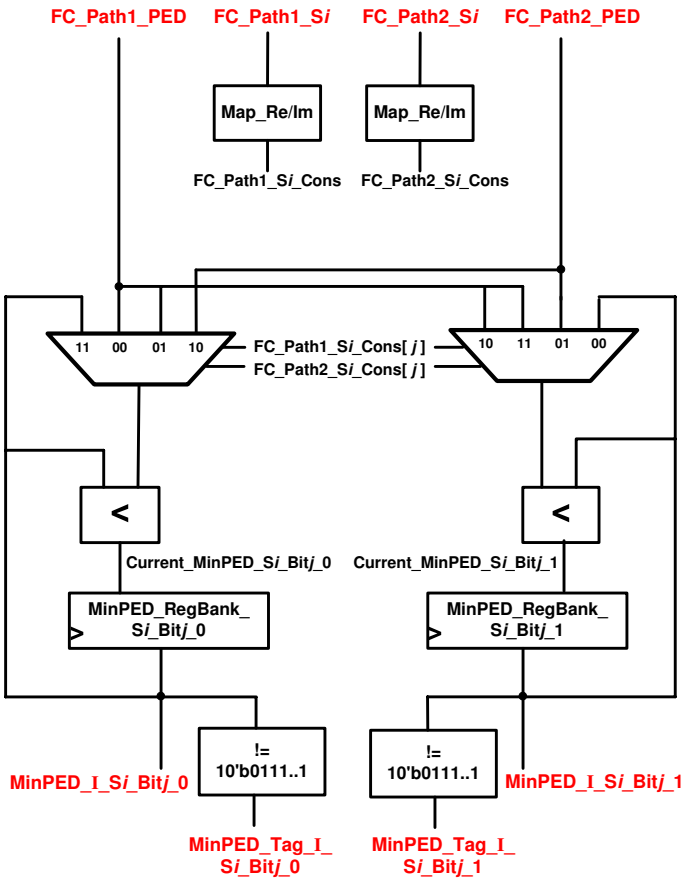

Figure 4.18: Architecture of a single functional block (for j^{th} bit of i^{th} transmitted symbol) for the `Fill_MinPEDTable_I_Part1` sub-block.

are generated one-by-one, in the order of ascending PEDs. These K-Best paths are then utilized in the `Fill_MinPEDTable_I_Part2` sub-block to fill the MinPED table. More details about the architecture and functionality of the `PE I` sub-block are included in Section B.4.

As shown in Fig. 4.17, the multiplexors at the top select the current ZF augmented path and transfer them to the "FC_Path1" and "FC_Path2" ports each clock cycle. The `Fill_MinPEDTable_I_Part1` and `Fill_MinPEDTable_I_Part2` sub-blocks then utilize these ZF augmented K-Best paths ("FC_Path1" and "FC_Path2") and the K-Best paths from the `PE I` sub-block ("KB_Path") to populate the MinPED table for $N_T log_2(Q)$ bits and compute their tags. The `Fill_MinPEDTable_I_Part1` sub-block samples in "FC_Path1" and "FC_Path2", and fills the MinPED table for real-valued symbols "S8" to "S2", corresponding to the the first $(2N_T$ -1)$(\log_2(Q)/2)$ transmitted bits, using steps `4.2` and `4.3` shown above. Fig. 4.18 shows the architecture of the `Fill_MinPEDTable_I_Part1` sub-block, for j^{th} bit of i^{th} real-valued transmitted symbol. As shown, the `MapBin2Constellation_Re/Im`

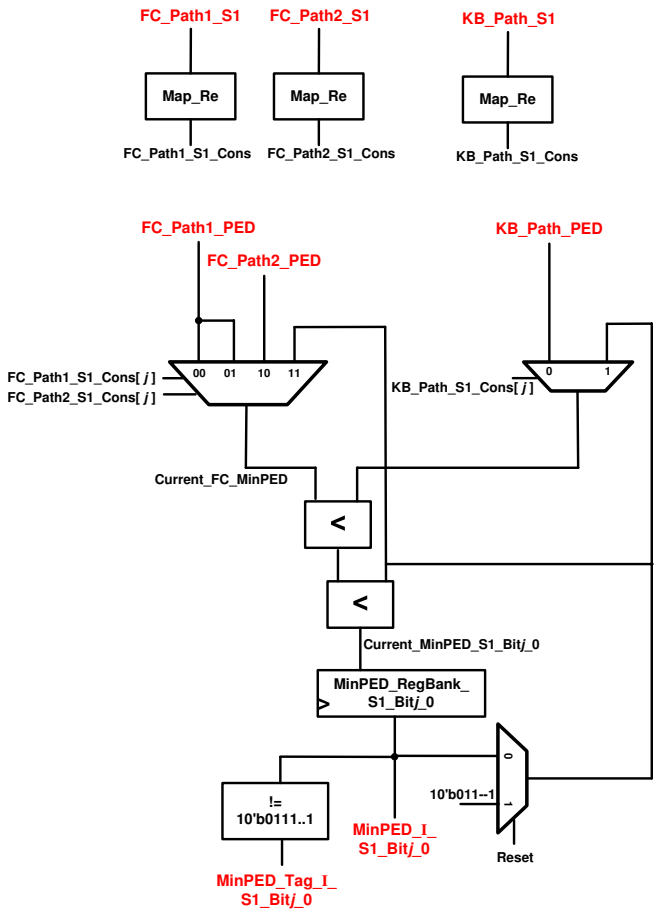

Figure 4.19: Architecture of a single functional block (for j^{th} bit of transmitted symbol S1) for the `Fill_MinPEDTable_I_Part2` block.

sub-blocks are first used to attain the constellation representation of the "Si". This constellation representation of "Si" is then used as select signals for the MUXs that select the minimum PED among "FC_Path1_PED" and "FC_Path2_PED", which is then compared with the present stored MinPED value to find the "Current_MinPED_Si_Bit$j$_(0/1)" values. As the final step, this current MinPED value is sampled by the MinPED register bank and is utilized to find the MinPED tag value. Note that, at the beginning of each set of paths, the MinPED register bank is initialized with 10'b0111111111, which is the largest positive value that can be represented by using 10 digits. Hence, a MinPED tag of "1" (MinPED != 10'b0111111111) would imply that MinPED value has been found for the current bit.

The `Fill_MinPEDTable_I_Part2` sub-block samples in "FC_Path1", "FC_Path2" and "KB_Path", and fills the MinPED table for real-valued symbol "S1", corresponding to the the last $\log_2(Q)/2$ transmitted bits, using steps `4.4.1` and `4.4.2` shown above. Fig.

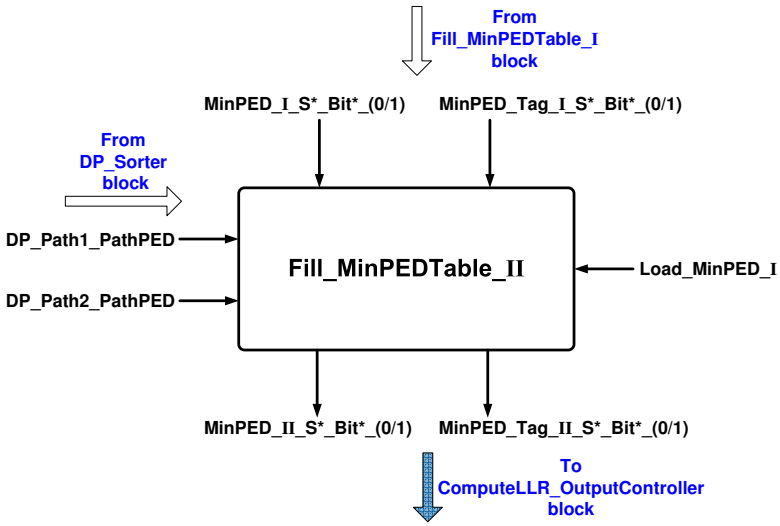

Figure 4.20: Overall Architecture of the `Fill_MinPEDTable_II` block.

4.19 shows the architecture of the `Fill_MinPEDTable_I_Part2` sub-block, for j^{th} bit of "S1". As shown, the `MapBin2Constellation_Re` sub-blocks are first used to attain the constellation representation of the "S1" symbol for the ZF augmented and K-Best paths, which are then used as select signals for MUXs that generate "Current_FC_MinPED" and "Current_KB_MinPED" values. The two serial comparators are then used to find "Current_MinPED_S1_Bit$j$_(0/1)" values. Note that the `Fill_MinPEDTable_I_Part2` sub-block uses a special mechanism to reset the MinPED register bank, shown in Fig. 4.19. This special reset mechanism is required since this block needs to be active for all $K = 10$ clock cycles (processing 1 K-Best path per cycle) and there are no idle cycles are available for register reset.

4.4.7 Fill_MinPEDTable_II

The `Fill_MinPEDTable_II` block performs the task of updating the MinPED table using the PED values of the chosen discarded paths that have been ZF augmented to the last tree level. Fig. 4.20 shows the input and output signals for the `Fill_MinPEDTable_II` block, along with information about which blocks they come from or go to. As shown, the `Fill_MinPEDTable_II` block samples in the present MinPED table and their tags from the `Fill_MinPEDTable_I` block and outputs an updated MinPED table and tags to the `ComputeLLR_OutputController` block. It also receives two discarded paths per clock cycle, in the order of ascending PED values, from the `DP_Sorter` block.

Fig. 4.21 shows the architecture of a single functional block, used to store and update

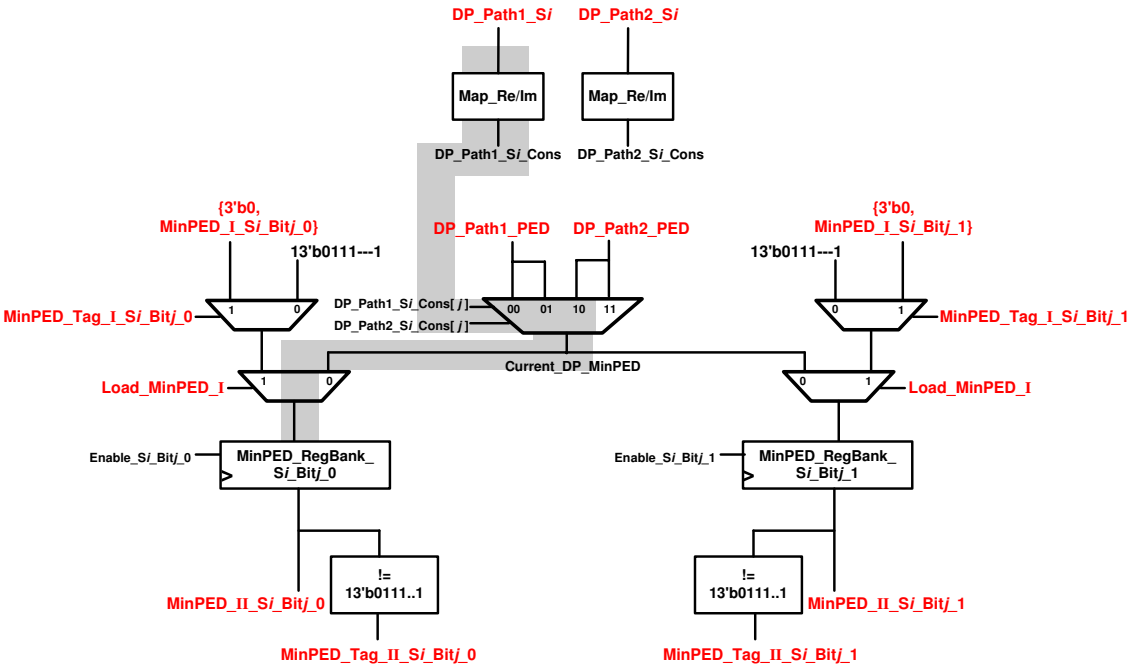

Figure 4.21: Architecture of a single functional block (for j^{th} bit of i^{th} transmitted symbol) for the `Fill_MinPEDTable_II` block - with the critical path highlighted.

MinPED value for j^{th} bit of i^{th} transmitted symbol. The overall `Fill_MinPEDTable_II` block consists a total of 24 instances of this single functional block. As shown in Fig. 4.21, the `MapBin2Constellation_Re/Im` sub-blocks are first used to attain constellation representation of the "Si" symbols for both "DP_Path1" and "DP_Path2". These constellation representations are then used as select signals for the MUX that selects the minimum PED among "DP_Path1_PED" and "DP_Path2_PED", to generate the "Current_DP_MinPED" value. The two branches on left and right sides of this MUX perform the task of storing and updating the MinPED for "0" and "1" values of the current bit, respectively. Note that since the PED word-length for K-Best and discarded paths are 10 bits and 13 bits, respectively, the input 10-bit MinPED values from the `Fill_MinPEDTable_I` block needs to be converted to 13-bit fixed-point format. Note that Fig. 4.21 also shows the critical path of the `Fill_MinPEDTable_II` block.

The `Fill_MinPEDTable_II` block implements the improvement idea: Relaxed LLR Computation Scheme, presented in Section 3.3.4. This improvement idea approximates the actual LLR computation by making the assumption that the MinPED values attained by extending the K-Best paths are always smaller compared to the discarded path PEDs.

Hence, for the proposed `Fill_MinPEDTable_II` block, there is no need to compare the "Current_DP_MinPED" value with the current MinPED value stored in the MinPED register bank. This results in significant hardware savings, since there are no comparators required. At the final outputs of the `Fill_MinPEDTable_II` block, the updated MinPED tags are computed by comparing the current stored MinPED with 13'b0111111111111. These computed MinPED tags are then utilized by the `ComputeLLR_OutputController` block, to ease the process of LLR computation significantly.

4.4.8 ComputeLLR_OutputController

The `ComputeLLR_OutputController` is the last block in the pipelined architecture of the proposed Soft K-Best MIMO detector. This block receives the table of MinPED values, that has been populated using the last level extension of K-Best paths and the FC augmented selected discarded paths. It uses these MinPED values and their corresponding tags to compute the Log-likelihood Ratio (LLR) values for 24 ($N_T * log_2(Q) = 4 * log_2(64) = 24$) transmitted bits. The `ComputeLLR_OutputController` block then outputs these LLR values using the scheduling described in Section 4.3.

Fig. 4.22 shows the overall architecture of the `ComputeLLR_OutputController` block, with critical path highlighted. As shown, the overall architecture can be divided into two major parts. Part1 consists of circuitry to compute LLR values using the MinPED table and Part2 operates as an Output Controller, to schedule and output the LLR values appropriately. As shown in Fig. 4.22, the `ComputeLLR_OutputController` block computes LLR values for 8 bits every 2 clock cycles. The interface between the `Fill_MinPEDTable_II` and the `ComputeLLR_OutputController` blocks contains multiplexors that generate appropriate values for the "MinPED_II_Bit$j$_(0/1)" signals, for $j = 1$ to 8, by simply selecting them from the MinPED values for all 24 bits, using the table 4.3.

The first part of the `ComputeLLR_OutputController` block selects MinPED values for one of the bits each cycle, using the control signal "Sel_MinPED_LivevsStored", which is set to "0" in cycle 1 and to "1" in cycle 2, in each set of 2 cycles. Note that the MinPED transfer scheduling between the `Fill_MinPEDTable_II` and the `ComputeLLR_OutputController` blocks creates the need for register banks to sample and store the MinPED values for the second bit. The chosen bit MinPED values for "0" and "1" are then subtracted to compute LLR values using equation (3.2). The `ComputeLLR_OutputController` block then utilizes the MinPED tag values to check validity of the computed LLRs and decide on the correct

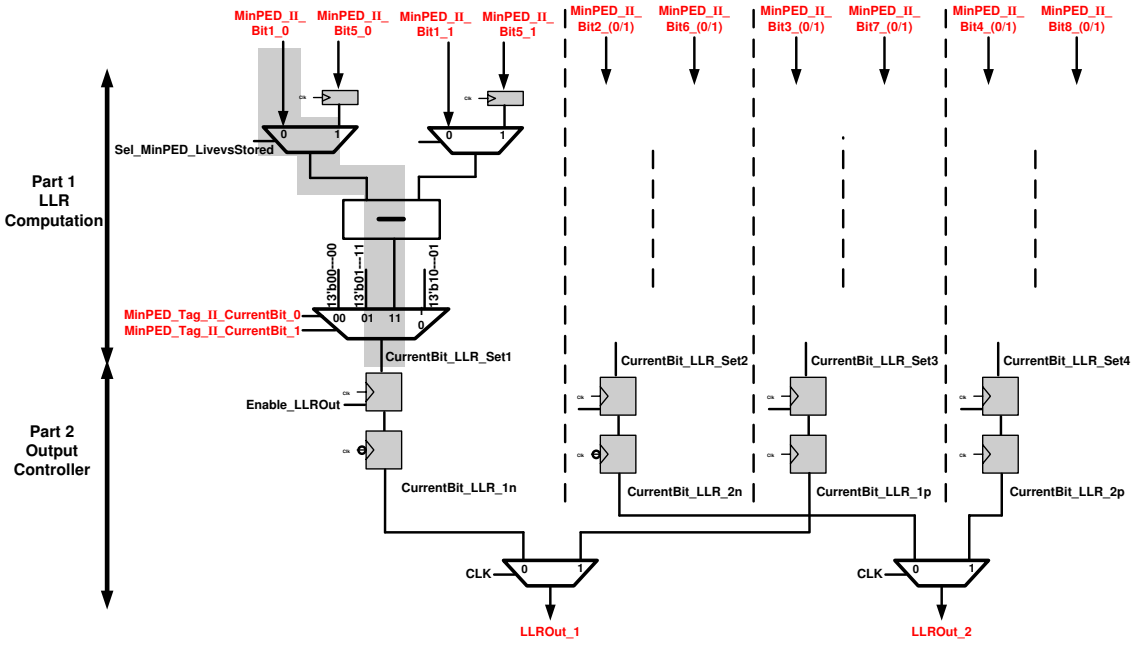

Figure 4.22: Overall Architecture of the `ComputeLLR_OutputController` block - with the critical path highlighted.

output LLR values.

The second part of the block acts as an Output Controller to output computed LLR values for 24 transmitted bits according the scheduling shown in table 4.1. As shown in Fig. 4.22, the two sets of register banks in series are used to reduce the critical path length. The clock driven MUXs then output two LLR values per clock cycle (one at each of the positive and negative edges of clock) at the ports "LLROut_1" and "LLROut_2". Thus, each port in the second part of the `ComputeLLR_OutputController` block outputs 12 LLR values within 6 clock cycles, in each set of 10 cycles.

4.5 Latency and Bit-True Simulations

The fine-grained pipelining used by the proposed Soft K-Best detector improves the throughput at the cost of the larger latency. Starting from the first block, the latency of `Level I` is 2 cycles, `Level II` has a 3-cycle latency and the `Sorter` block's latency is 4 cycles. Furthermore, the `PE I` block's latency is $K = 10$ cycles, and finally `PE II`'s latency is 10 cycles plus an additional 6 cycles for the pipelined `FC-Block`. Since the

Table 4.3: MinPED Transfer Schedule at the interface between Fill_MinPEDTable_II and ComputeLLR_OutputController blocks.

MinPED_II_Bit$j$_(0/1)/ Clock Cycle Set:	Cycles 1 and 2:	Cycles 3 and 4:	Cycles 5 and 6:
MinPED_II_Bit1_(0/1)	MinPED_II_ S1_Bit5_(0/1)	MinPED_II_ S2_Bit3_(0/1)	MinPED_II_ S3_Bit1_(0/1)
MinPED_II_Bit2_(0/1)	MinPED_II_ S1_Bit4_(0/1)	MinPED_II_ S2_Bit2_(0/1)	MinPED_II_ S3_Bit0_(0/1)
MinPED_II_Bit3_(0/1)	MinPED_II_ S1_Bit3_(0/1)	MinPED_II_ S2_Bit1_(0/1)	MinPED_II_ S4_Bit5_(0/1)
MinPED_II_Bit4_(0/1)	MinPED_II_ S1_Bit2_(0/1)	MinPED_II_ S2_Bit0_(0/1)	MinPED_II_ S4_Bit4_(0/1)
MinPED_II_Bit5_(0/1)	MinPED_II_ S1_Bit1_(0/1)	MinPED_II_ S3_Bit5_(0/1)	MinPED_II_ S4_Bit3_(0/1)
MinPED_II_Bit6_(0/1)	MinPED_II_ S1_Bit0_(0/1)	MinPED_II_ S3_Bit4_(0/1)	MinPED_II_ S4_Bit2_(0/1)
MinPED_II_Bit7_(0/1)	MinPED_II_ S2_Bit5_(0/1)	MinPED_II_ S3_Bit3_(0/1)	MinPED_II_ S4_Bit1_(0/1)
MinPED_II_Bit8_(0/1)	MinPED_II_ S2_Bit4_(0/1)	MinPED_II_ S3_Bit2_(0/1)	MinPED_II_ S4_Bit0_(0/1)

`Fill_MinPEDTable_I` block contains an internal `PE I` block, its total latency is 11 cycles, while the `Fill_MinPEDTable_II` and `ComputeLLR_OutputController` blocks have a latency of 7 clock cycles each. Therefore, according to the overall architecture shown in Fig. 4.1, the total latency of the architecture is $2+3+4+6\times10+6\times16+11+7+7=190$ cycles. However, note that the architecture outputs LLR values for 24 transmitted bits (corresponding to a single 4×1 complex transmitted vector) every 10 clock cycles.

Table 4.4 shows the number of bits associated with different variables in the algorithm for the bit-true simulation[2] as well as the hardware implementation for the case of a 4×4, 64-QAM constellation in the form of $[n : m]$, where n and m denote the total number of bits (word-length) and the number of bits in the fractional part (fraction-length), respectively. The fixed-point simulations are performed using the 2's complement number representation. Note that the word lengths in Table 4.4 have been derived based

[2]Bit-true simulation refers to the simulation results with finite word-length effect, which is also equivalently called the fixed-point simulation.

Table 4.4: Fixed-point Word-Length (bits) of Parameters.

Parameter	r_{ii}	r_{ij}	z_i	s_i	KB_PED	DP_PED	LLR
Word Length†	[13:12]	[16:11]	[13:4]	[3:0]	[10:7]	[13:7]	[13:7]

† $[n : m]$ an n-bit number with m bits for the fractional part.

on extensive bit-true simulation results to minimize the BER loss relative to the floating-point result (i.e., less than 0.5 dB at BER = 10^{-3}) for the Soft K-Best MIMO detector. This means that based on the extensive floating-point simulations, the dynamic range of all the variables were determined, based on which the required number of bits for integer and fractional parts were calculated.

4.6 BER Simulations and Design Comparison

4.6.1 BER Simulation Results

The K-Best MIMO detection algorithm is not an ML-optimal detector, and hence it might miss the hard-ML point, resulting in performance loss. However, for a proper choice of K value, its bit-error-rate (BER) performance approaches the optimal case over a reasonable range of SNR values. The floating-point and fixed-point MATLAB models for the proposed Soft K-Best MIMO detector were combined with Convolutional Turbo Coding (CTC) Encoder and Decoder with rate = 1/2, 600 bytes/block and 8 decoder iterations. These combined models were customized for use in single-carrier 4×4 MIMO system with 64-QAM and 16-QAM modulation schemes. In both of these simulations, the K-Best approach has been tested for 2000 blocks, where each block consists of 4800 bits (600 bytes/block * 8 bits/byte). Hence, the K-Best detectors were simulated for 9.6Mbits in total. Test vectors are created using: (i) pseudo-random data, (ii) complex-valued random Gaussian channel matrix $\widetilde{\mathbf{H}}$ with statistically independent elements updated per four channel use, and (iii) additive white Gaussian (circularly symmetric) complex random noise.

Figures 4.23 and 4.24 show the BER curves obtained by simulating these combined models for 16-QAM and 64-QAM case scenarios, respectively. From these BER curves, it can be noticed that the fixed-point results for all cases are within a reasonable distance from the floating-point results (i.e., less than 0.5 dB at BER = 10^{-3}). Note that the number of bits for various parameters in the fixed-point simulation is based on Table 4.4. Also, it can be noticed that the proposed Soft K-Best scheme results in considerable BER performance improvement compared to the conventional Soft K-Best detection and Hard

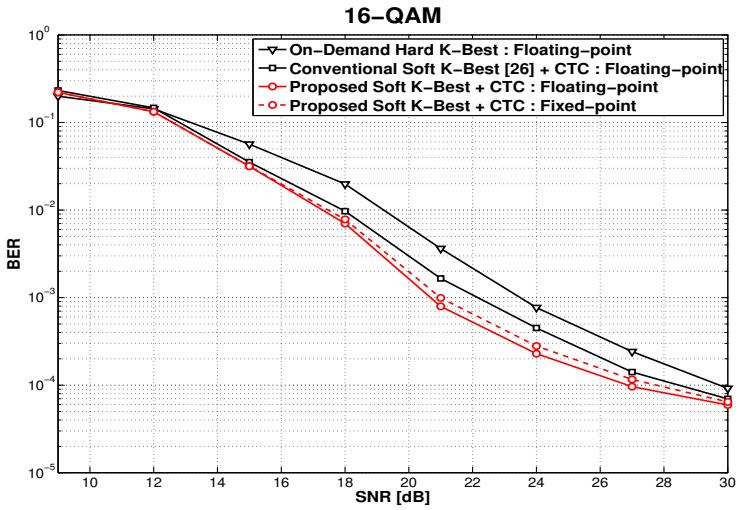

Figure 4.23: BER Performance of Hard K-Best and the proposed Soft K-Best Detection scheme (Floating-point and Fixed-point) - for 4x4 MIMO system with 16-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations:

K-Best detection schemes. For the 64-QAM case (Fig 4.24), fixed-point model of the proposed Soft K-Best detector improves the BER performance by approximately 1.8 dB and 2.9 dB at BER = 10^{-3} compared to fixed-point models of conventional Soft K-Best and Hard K-Best, respectively.

4.6.2 Design Characteristic Comparison

For the purpose of design comparison, the proposed VLSI architecture for Soft K-Best detector was modeled in Verilog HDL, synthesized using Synopsys Design Compiler and placed and routed using Cadence SoC Encounter/Silicon Ensemble. The RTL and gate level netlists were verified with the golden model generated from the fixed-point MATLAB model [1]. The final Soft K-Best ASIC was fabricated in 0.13 μm IBM 1P8M CMOS technology using ARM standard library cells. Timing analysis on the RC extracted netlist shows that the proposed Soft K-Best MIMO detector attains peak data throughput of 655 Mbps, while running at the highest clock frequency of 270 MHz. The proposed MIMO detector design consumes 1.45 mm^2 (174 KG)[3] silicon area and 195 mW power.

[3]The average gate density of the ARM 0.13 μm standard cell library is 120 KG/mm^2.

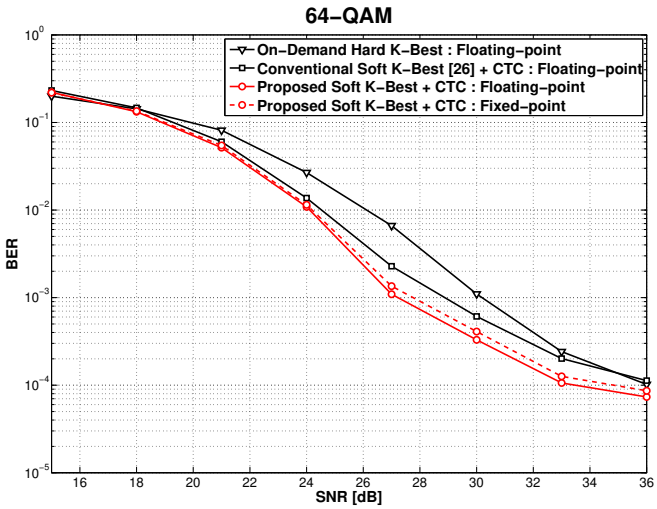

Figure 4.24: BER Performance of Hard K-Best and the proposed Soft K-Best Detection scheme (Floating-point and Fixed-point) - for 4x4 MIMO system with 64-QAM, K=10 - using Convolutional Turbo Coding with rate = 1/2, 600 bytes/block and 8 decoder iterations:

The proposed Soft K-Best MIMO detector was also synthesized and characterized in TSMC 65nm CMOS technology with ARM standard library cells. Synthesis results in the 65nm CMOS technology with Low Vt (LVT) transistors and `tt1p3V25C` PVT (typical process, 1.3V VDD and 25^{o}C temperature) demonstrate that the proposed Soft K-Best detector attains the peak data throughput of 2Gbps, while operating at 833MHz clock frequency and consuming 0.57 mm^2 silicon area and 280 mW of power. In order to reduce power consumption and minimize decoding energy per bit, the proposed MIMO detector was also synthesized with Standard Vt (SVT) transistors and `ss1p1V105C` PVT, as well as with High Vt (HVT) transistors and `tt1p1V25C` PVT. The peak throughput, area and power consumption results, as well as decoding energy per bit, have been summarized in Table 4.5.

Table 4.5 provides an overview and comparison of all the reported ASIC implementations in the literature for 4×4 16-QAM and 64-QAM MIMO detectors, respectively, including this work. The applied algorithm and the value of the K parameter in the case of the K-Best algorithm, the fabricated technology, as well as some detailed specifications of each ASIC are listed. If information on other designs is not provided by the authors, this is indicated by the entry `N/A`. From this table, the following points can be inferred:

Table 4.5: Comparison of the Current ASIC Implementations of 4×4 MIMO Detectors.

Reference	[11]	[26]	[30]	[29]	[1]	This work 0.13μm	This work 65nm LVT tt1p3V25C	This work 65nm SVT ss1p1V105C	This work 65nm HVT tt1p1V25C
Modulation	16-QAM	64-QAM	64-QAM	64-QAM	64-QAM	64-QAM	64-QAM	64-QAM	64-QAM
Method	K-Best	K-Best	K-Best	SD/SIC	K-Best	K-Best	K-Best	K-Best	K-Best
K-value	5	64	64	5	10	10	10	10	10
Process	0.13 μm	0.13 μm	65 nm	90 nm	0.13 μm	0.13 μm	65 nm	65 nm	65 nm
Core Area	97 KG	5270 KG	174 KG	294 KG	114 KG	174 KG	298 KG	302 KG	293 KG
Max Freq.	200 MHz	270 MHz	200 MHz	166 MHz	270 MHz	270 MHz	833 MHz	335 MHz	266 MHz
Throughput	107 Mbps	100 Mbps	115 Mbps	95 Mbps	655 Mbps	655 Mbps	2000 Mbps	800 Mbps	640 Mbps
Latency	1.2 μs	N/A	N/A	0.25 μs	0.6 μs	0.7 μs	0.230 μs	0.566 μs	0.712 μs
Power	N/A	847 mW	11 mW	N/A	131 mW	195 mW	280 mW	84 mW	65 mW
Energy/bit	N/A	8470pJ/b	96pJ/b	N/A	200pJ/b	298pJ/b	140pJ/b	105pJ/b	101.5pJ/b
Soft/Hard	Soft	Soft	Soft	Hard	Hard	Soft	Soft	Soft	Soft
SNR Dep.	No	Yes	Yes	Yes	No	No	No	No	No
Domain	Real	Complex	Complex	Complex	Real	Real	Real	Real	Real

1. Comparing this thesis to [26], which are both fabricated in 0.13 μm CMOS technology, reveals a significant reduction in the area achieved and significant increase in the peak data throughput using our proposed Soft K-Best scheme. Also, our proposed detector attains a much lower power consumption and decoding energy per bit requirements. These are because of the fact that only a tiny fraction of all the children are expanded in our architecture. Furthermore, the proposed Soft K-best scheme uses a Last Stage On-Demand Expansion strategy, that avoids exhaustive expansion, and hence eliminates a large amount of computations required for LLR calculation at the last tree level. Also, $K = 64$ in [26] is in the complex domain as opposed to $K = 10$ in our case in the real domain with a finer granularity.

2. The Soft K-Best design presented in [30] achieves smaller power consumption and decoding energy requirements compared to the proposed Soft K-Best design. However, the design in [30], fabricated in the state-of-the-art 65nm technology, attains a decoding throughput of only 115Mbps. Furthermore, this K-Best detection algorithm [30] uses major approximations, such as approximate sorting with coarse granularity, bidirectional partial tree search and ECC feedback-aided detection bypassing, that results in significant performance degradation compared to exact K-Best BER results. Hence, to summarize, compared to [30], the proposed Soft K-Best detector attains a much higher decoding throughput and larger gain in BER performance, while requiring marginally larger energy per bit.

3. As mentioned earlier, the proposed Soft K-Best detection scheme builds up on the Hard K-Best detection scheme presented in [1]. Comparing our Soft K-Best design with the Hard K-Best design presented in [1], it can be noticed that the Soft K-Best design attains the same decoding throughput of 655Mbps, in spite of the much larger number of extra calculations required to compute LLR values. Furthermore, the Soft-output MIMO detection in this thesis with ECC provides a much superior bit error rate (BER) performance compared to the uncoded hard-output MIMO detection presented in [1]. The proposed Soft-output MIMO detector achieves BER performance improvement of 2.9 dB at BER = 10^{-3}, and still offers a throughput of 655Mbps.

4. The emerging 4G wireless standards, IEEE 802.16m (WiMAX) and LTE-Advanced, pose aggressive decoding throughput requirements for MIMO detectors. Note that

these standards require peak uncoded data rates of up to 1Gbps, which translates to Downlink coded data rates of up to 2Gbps. As can be noticed from Table 4.5, none of the previously published MIMO detectors fulfill these demanding throughput requirements. However, synthesis results for the proposed Soft K-Best detector in 65nm CMOS technology with LVT transistors and `tt1p3V25C` PVT show that it attains a peak data throughput of 2Gbps, while requiring 0.57 mm^2 silicon area and 280 mW of power. Thus, to the best of our knowledge, the proposed Soft K-Best detector in this thesis is the only design to-date that fulfills the aggressive data rate requirements imposed by the emerging IEEE 802.16m and LTE-Advanced wireless standards.

5. The mobile applications, envisioned in the IEEE 802.16m and LTE-Advanced standards, require MIMO detectors with low power consumption and minimum possible decoding energy per bit. The synthesis of the proposed MIMO detector in TSMC 65nm CMOS technology with SVT transistors and `ss1p1V105C` PVT shows power consumption requirement only 84 mW, while still attaining a throughput of 800Mbps. Furthermore, the proposed design with HVT transistors and `tt1p1V25C` PVT dissipates only 65mW power, with throughput of 640 Mbps, hence achieving a low decoding energy per bit of 101.5 pJ/bit.

Fig. 4.25 compares the Soft K-Best detector presented in this thesis to previously published works. It compares the total achieved throughput as a function of the number of kilo-gates (KG) used in each design. Designs for 16-QAM (▲) and 64-QAM (■) have been distinguished with different icons. The highest achieved throughput is reported for SNR-dependent schemes. It can be seen that our implementation has the highest throughput ever reported for the 4×4, 64-QAM designs. As was expected, in general, the designs in 16-QAM take lower area as the value of K, the number of possible children per parents, and the sorting cores are almost 4× smaller compared to that of 64-QAM.

4.7 Summary

This chapter presented an area and power efficient novel high-throughput VLSI implementation of a 4×4 64-QAM MIMO detector, that is suitable for high-order constellation

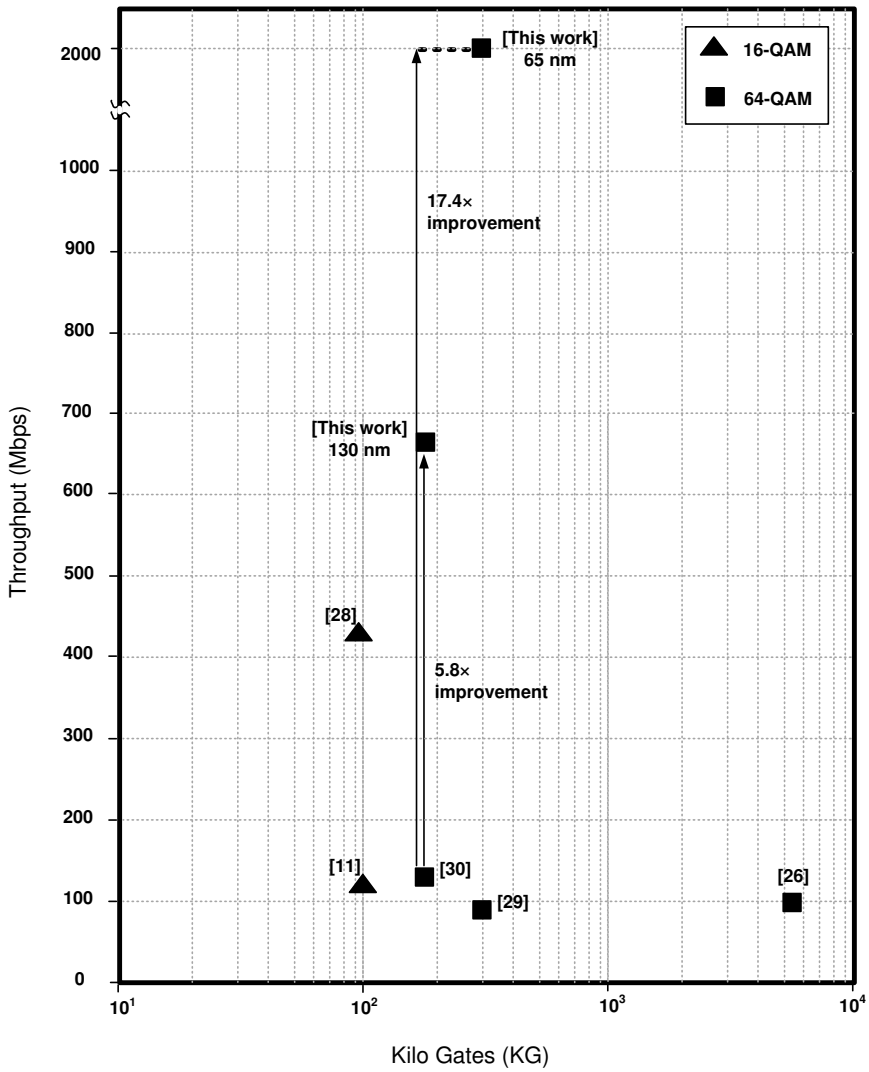

Figure 4.25: Throughput vs. Gate Count Comparison with Previously Published Works.

schemes. The proposed MIMO detector is designed based on a low-complexity high-performance Soft K-Best detection scheme presented in section 3.3.5. This scheme utilizes information contained in the discarded paths to improve BER performance, and then reduces computational complexity using three innovative improvement ideas. The proposed Soft K-Best detector uses a deeply pipelined architecture to maximize throughput and various strategies to minimize hardware and power requirements.

The proposed design attains a 5.8× improvement in throughput in 0.13 μm technology and a 14.7× improvement in 65 nm technology, compared to the highest throughput published previously for Soft-output MIMO detectors. Synthesis results in 65nm CMOS technology shows that the proposed Soft-output MIMO detector attains a peak coded

data throughput of 2Gbps, which makes it the only design to-date that fulfills the aggressive data rate requirements (1Gbps uncoded) imposed by the emerging IEEE 802.16m and LTE-Advanced 4G wireless standards. Furthermore, this detector is also suitable for low-power mobile applications that require high data rates, since it achieves a low power consumption of 65mW at 1.1V supply and low decoding energy per bit requirement of 101.5 pJ/bit, while still providing a data throughput of 640 Mbps.

5 QR Decomposition - Algorithm and VLSI Implementation

5.1 Introduction

Several types of channel pre-processing operations run in parallel with MIMO detection, one of which is the QR Decomposition (QRD) of the estimated channel characteristic matrix. QRD is required by many types of MIMO detection schemes, such as Successive Interference Cancellation (SIC), V-BLAST, K-Best, Sphere Decoding and many other schemes. It is used in MIMO receivers to transform the $N_R \times N_T$ complex channel matrix $\mathbf{H}$ into a $N_R \times N_T$ unitary and orthonormal matrix $\mathbf{Q}$ (such that $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$, where $\mathbf{Q}^H = (\mathbf{Q}^T)^*$) and a $N_T \times N_T$ upper triangular matrix $\mathbf{R}$. As discussed in Section 2.1, these complex matrices $\mathbf{Q}$ and $\mathbf{R}$, along with the received symbol vector, $\mathbf{y}$, are then used by the MIMO detector to estimate the transmitted symbol vector, $\mathbf{s}$. The major reason why QR Decomposition is preferred for MIMO receivers, compared other matrix triangularization schemes, is because of the fact that the matrix $\mathbf{Q}$ generated by QR Decomposition is a unitary matrix, which helps to avoid the noise enhancement problem and keeps noise spatially white.

Many new 4G wireless standards require MIMO systems with high data rates, high mobility and large antenna configurations. For example, the IEEE 802.16m and LTE-Advanced standards include applications with mobile speeds up to 350 km/h, maximum antenna configuration of 8×8 and Downlink peak data rates of up to 1Gbps. Furthermore, the high-mobility applications involve dynamic and fast-varying channel environments, which require channel estimation and QR Decomposition to be performed very frequently, for every few channel uses. Thus, it is desired to minimize the QRD Processing Latency, which is formally defined as the number of cycles after which a new set of QRD outputs

is ready. Also, for MIMO receivers to be used in battery-operated mobile devices, it is desired to minimize QRD power dissipation and silicon area as much as possible. Hence, to summarize, the applications in the emerging 4G wireless standards require QRD implementations for decomposing large complex channel matrices, while minimizing QRD processing latency, silicon area and power consumption requirements.

The outline of this chapter is as follows. The chapter first lists the existing QR Decomposition schemes and describes the implementation challenges they face. It then describes various types of CORDIC algorithms that provide a low complexity method to implement vector rotations that are executed during QR Decomposition. The proposed QRD scheme is then described and its computational complexity is analyzed. The chapter then describes the overall VLSI architecture for the proposed QRD core and provides details about architecture and functionality for each of the individual sub-blocks. The last part of this chapter provides BER results and test results for the fabricated QRD chip. It also presents the design characteristics of the fabricated QRD chip and compares it with other state-of-the-art QRD designs.

5.2 Existing QR Decomposition Schemes and Implementation Challenges

The 3 basic methods for computing matrix QR Decomposition include: the Modified Gram-Schmidt Orthonormalization (MGS) algorithm, Householder transformations and Givens rotations. The MGS algorithm computes **Q** and **R** matrices, column by column, by using vector projection, norm and other computations [33]. However, for fixed-precision arithmetic, it offers lesser accuracy and numerical stability due to round-off errors and loss of orthogonality introduced during the vector projections onto planes [33]. Also, a straightforward implementation of this algorithm requires multiplication, division and square-root operations, which lead to high implementation complexity and high computation latency.

The authors of [34] propose an idea of using log-domain computations to implement multiplication, division and square-root operations using low-complexity adders, subtractors and shifters. However, this scheme requires frequent conversions between log and linear

domains using LOG ($log_2(x)$) and EXP (2^x) Look-Up Tables (LUT). Hence, a VLSI implementation for this scheme requires a large storage space to hold these Look-Up Tables and results in large gate count. In a 0.18 μm CMOS technology, this design requires a core area of 72KG and attains a QRD processing latency of 67 cycles at a clock frequency of 277 MHz. To reduce the gate count, the authors of [35] propose a modified MGS scheme that reduces circuit complexity and power consumption by using an approximation in the MGS algorithm step that requires division by a real-valued norm. Since multiplication is a simpler operation to implement than division, [35] substitutes the division and square-root computations with multiplication and inverse square-root ($1/\sqrt{x}$) calculations. The computation of the inverse square-root is further approximated by using the following function, that is attained by manual curve fitting within the desired range of x:

$$\frac{1}{\sqrt{x}} \approx 0.965820 - (\frac{1}{4})x - (\frac{1}{32})x$$

However, since this QRD scheme uses an approximation to an actual function, for fixed-precision arithmetic, it might lead to a degradation in the bit error rate (BER) performance of the MIMO detector. Furthermore, since the QRD core presented in [35] uses an iterative architecture, it incurs a very large processing latency. In 0.13 μm CMOS, this QRD design requires a QRD processing latency of 139 clock cycles at 269 MHz and requires only 23.3KG silicon area.

As another way of computing **Q** and **R** matrices, Householder transformations can be used to transform the input channel matrix **H** to the final upper-triangular **R** matrix, by eliminating all of the elements below the diagonal in a column simultaneously [33]. However, a major disadvantage of Householder transformations, when used for QR Decomposition, is that since a Householder reflection operates on all of the matrix rows simultaneously, it is not straightforward to carry out multiple reflections in parallel, which could have helped to speed up the QR Decomposition process [24]. Also, a straightforward VLSI implementation of the Householder algorithm requires multiplication, division and square-root operations, and hence leads to very high hardware complexity.

As an alternative, Givens rotations have the capability of selectively annihilating individual matrix elements by rotating two-dimensional real or complex column vectors to align them with the pivot axis. Since Givens Rotations work on two matrix rows at a time, they can be more easily parallelized, to reduce the QR Decomposition processing

latency [24]. Application of Givens rotations to two-dimensional column vectors within the input **H** matrix can be implemented using either multiply-and-add operations or more commonly using the COordinate Rotation DIgital Computer (CORDIC) algorithm [36]. The Vectoring and Rotation modes of the CORDIC algorithm can be used to approximate vector rotation and hence perform Givens rotations using low-complexity shift and add operations. Hence, the capability of performing multiple Givens rotations in parallel, which in turn leads to higher throughput, as well as the lower hardware complexity of the CORDIC modules makes Givens rotations the method of choice for implementing QR Decomposition in this thesis.

However, for matrices with large dimensions (e.g. 4×4 complex), performing QRD using the conventional sequence of Givens rotations might lead to high computational complexity, due to the large number of Vectoring and Rotation operations required. For an example of MIMO systems with 4 transmit and 4 receive antennas, the process of decomposing a 4×4 complex channel characteristic matrix **H** into 4×4 complex matrices **Q** and **R** using the conventional sequence of Givens rotations will require a total of 26 real Vectoring and 200 real Rotation operations. In [37], QRD for a 4×4 complex matrix **H** is implemented using the conventional sequence of Givens rotations, which attains the processing latency of 67 cycles at 125MHz clock frequency, and requires 54KG in 0.25 μm CMOS technology. Furthermore, as will be discussed in detail in Section 5.4.1, the sequential nature of annihilations for the $H_{i,j}^{Re}$ matrix elements and the large number of Rotation operations required for each element annihilation causes a throughput bottleneck. These factors will lead to high computational complexity, larger hardware requirements and high power dissipation for throughput-constrained systems. Hence, a QR Decomposition architecture designed using these schemes will not be suitable for use in MIMO receivers embedded within mobile devices, that essentially require signal processing blocks with low power dissipation and low silicon area.

5.3 Conventional, Multi-Dimensional and Householder CORDIC Algorithms

5.3.1 Conventional 2D CORDIC Algorithm

The COordinate Rotation DIgital Computer (CORDIC) algorithms, introduced by Volder [38] and extended by Walther [39], provide the mechanism to perform vector rotations in hardware using low-complexity adders and shifters. Thus, the CORDIC algorithms, in their Vectoring and Rotation modes, can be used to approximate 2D Givens rotations. In the Vectoring mode, the CORDIC algorithm rotates the input vector by a necessary angle to align the resulting vector with the X axis. In the Rotation mode, the input vector is rotated by the specified angle to attain updated co-ordinates of the vector after rotation.

In the Vectoring mode, the output is a rotation angle and the norm of the original vector. The CORDIC algorithm in effect attempts to minimize the Y component of the updated vector at each incremental rotation, and hence it uses the sign of the residual Y component to determine the direction for the next rotation. Since the angle accumulator is initialized with zero and is updated at each incremental rotation, it will contain the traversed angle, i.e. the angle between the vector and the x-axis, at the end of the Vectoring operation. To summarize, in Vectoring mode, the CORDIC elementary rotation equations are [36]:

$$\begin{aligned} X^{i+1} &= X^i - 2^{-i} D^i Y^i \\ Y^{i+1} &= Y^i + 2^{-i} D^i X^i \\ \theta^{i+1} &= \theta^i - D^i tan^{-1}(2^{-i}) \end{aligned} \tag{5.1}$$

where, $D^i = -sign(Y^i)$. Thus, if the Vectoring operation is completed so that the residual Y component is zero, we have [36]:

$$\begin{aligned} X^n &= A^n\sqrt{(X^0)^2+(Y^0)^2} \\ Y^n &= 0 \\ \theta^n &= tan^{-1}(Y^0/X^0) \\ A^n &= \prod\sqrt{1+2^{-2i}} \end{aligned} \tag{5.2}$$

Note, that here $[X^0 \;\; Y^0]^T$ and $[X^n \;\; Y^n]^T$ represent the input and output vectors to the Vectoring process, respectively. Also, A^n represents the processing gain of the CORDIC algorithm, where n represents the number of CORDIC algorithm iterations.

In the Rotation mode, the angle accumulator is first initialized with the desired rotation angle. The direction of elementary rotations is determined so that the magnitude of the residual angle, in the angle accumulator, is diminished. In this mode, the CORDIC algorithms use the same elementary rotation equations as shown in 5.1, however, at each iteration, D^i is determined as: $D^i = sign(\theta^i)$. Also, once the Rotation operation is completed, the final outputs can be written as [36]:

$$\begin{aligned} X^n &= A^n[X^0\cos\theta^0 - Y^0\sin\theta^0] \\ Y^n &= A^n[Y^0\cos\theta^0 + X^0\sin\theta^0] \\ \theta^n &= 0 \\ A^n &= \prod\sqrt{1+2^{-2i}} \end{aligned} \tag{5.3}$$

Note, that the CORDIC algorithms approximate the actual vector rotations by using a series of successively smaller elementary rotations by angles $tan^{-1}2^{-i}$. Hence, there is a direct trade-off between n, the number of CORDIC algorithm iterations, the accuracy of the vector rotations and computational complexity of the rotation operation. In other words, an increase in the value of n improves the vector rotation accuracy, however, it leads to larger computational complexity, and hence larger resource requirements.

5.3.2 Multi-Dimensional CORDIC Algorithm

Multi-dimensional Givens rotations operate on column vectors of dimensions larger than 2, to align them with the first axis (Vectoring operation) and then to apply the same rotation to rotate other vectors (Rotation operation). This approach increases the parallelism in the vector rotation operation by processing all components of the vector simultaneously. However, a generic way to implement multi-dimensional Givens rotations, for column vector dimensions 3 or larger, is to use high-complexity multiply-and-accumulate based algorithms. From the VLSI implementation perspective, this results in reduced latency, however, leads to much larger hardware requirements.

To resolve this issue, [40] presents Multi-dimensional CORDIC algorithms that extend the conventional two-dimensional CORDIC algorithms to 3D and 4D. In other words, the 3D and 4D CORDIC algorithms, presented in [40], approximate 3D and 4D Givens rotations using low-complexity shift and addition operations. Note that, the 3D and 4D vector rotation refer to rotations of 3×1 and 4×1 real-valued vectors. The CORDIC elementary rotation equations for 3D Givens rotations is shown below [40]:

$$\begin{aligned} X_1^{i+1} &= X_1^i(1-2^{-2i}) + X_2^i(D_1^i 2^{-i+1} + D_1^i D_2^i 2^{-2i+1}) + X_3^i(2D_2^i 2^{-i} + 2^{-2i+1}) \\ X_2^{i+1} &= X_1^i(-D_1^i 2^{-i+1} + D_1^i D_2^i 2^{-2i+1}) + X_2^i(1-2^{-2i}) + X_3^i(D_1^i 2^{-i+1} + D_1^i D_2^i 2^{-2i+1}) \\ X_3^{i+1} &= X_1^i(-2D_2^i 2^{-i} + 2^{-2i+1}) + X_2^i(-D_1^i 2^{-i+1} + D_1^i D_2^i 2^{-2i+1}) + X_3^i(1-2^{-2i}) \end{aligned} \tag{5.4}$$

where, the rotation directions are calculated as: $D_1^i = sign(X_1^i \cdot X_2^i)$ and $D_2^i = -sign(X_1^i \cdot X_3^i)$. Also, the CORDIC processing gain for the 3D CORDIC algorithm can be calculated using the following equation, for n iterations of this 3D CORDIC algorithm:

$$A^n = \prod 1 + 3 * 2^{-2i} \tag{5.5}$$

The CORDIC elementary rotation equations for 4D Givens rotations are shown in 5.6 below [40]:

$$\begin{aligned} X_1^{i+1} &= X_1^i - 2^{-i}D_1^i X_2^i - 2^{-i}D_2^i X_3^i - 2^{-i}D_3^i X_4^i \\ X_2^{i+1} &= 2^{-i}D_1^i X_1^i + X_2^i + 2^{-i}D_3^i X_3^i - 2^{-i}D_2^i X_4^i \\ X_3^{i+1} &= 2^{-i}D_2^i X_1^i - 2^{-i}D_3^i X_2^i + X_3^i + 2^{-i}D_1^i X_4^i \\ X_4^{i+1} &= 2^{-i}D_3^i X_1^i + 2^{-i}D_2^i X_2^i - 2^{-i}D_1^i X_3^i + X_4^i \end{aligned} \tag{5.6}$$

where, the rotation directions are calculated as: $D_1^i = -sign(X_1^i \cdot X_2^i)$, $D_2^i = -sign(X_1^i \cdot X_3^i)$ and $D_3^i = -sign(X_1^i \cdot X_4^i)$. Also, the CORDIC processing gain can be given as:

$$A^n = \prod \sqrt{1 + 3 * 2^{-2i}} \tag{5.7}$$

Since these 3D and 4D CORDIC algorithms can annihilate multiple elements simultaneously (2 elements for the 3D case and 3 elements for the 4D case) using only shift and addition operations, they offer a significant reduction in hardware complexity, as well as reduction in the overall processing latency. The details about hardware implementation of these equations to develop 3D and 4D CORDIC processors will be presented in Section 5.6.

5.3.3 Householder CORDIC Algorithm

Householder transformations also provide the capability of annihilating multiple elements simultaneously by reflecting a multi-dimensional input vector onto a plane. The details about how the Householder reflections can be used to rotate multi-dimensional vectors, and hence reduce QR Decomposition processing latency, were presented in Section 2.4.3. However, a straightforward VLSI implementation of the Householder algorithm requires multiplication, division and square-root operations, and hence it leads to very high hardware complexity [33]. To resolve this issue, [41] presents novel Householder CORDIC algorithms that use sequences of simple Householder reflections, which can be easily implemented using shift, carry-save-addition (CSA) and simple addition operations. In [41], the authors derive the elementary rotation matrix for generic nD Householder CORDIC algorithms, as products of two simple Householder reflections. The details about the

derivation of these elementary rotation matrices and their corresponding control sign selection laws, generalization of these algorithms in Euclidean and pseudo-Euclidean spaces and the algorithm convergence proofs are also presented in [41].

The elementary rotation matrix for nD Householder CORDIC algorithm has been customized for 3D and 4D cases using appropriate parameters. Equation 5.8 below shows the rotation equations for a single (i^{th}) iteration of the 3D Householder CORDIC algorithm.

$$\begin{aligned} X_1^{i+1} &= X_1^i - 2^{-2i+1}X_1^i + 2^{-i+1}D_1^iX_2^i + 2^{-i+1}D_2^iX_3^i \\ X_2^{i+1} &= -2^{-i+1}D_1^iX_1^i + X_2^i - 2^{-2i+1}D_1^iD_2^iX_3^i \\ X_3^{i+1} &= -2^{-i+1}D_2^iX_1^i - 2^{-2i+1}D_1^iD_2^iX_2^i + X_3^i \end{aligned} \tag{5.8}$$

where, the rotation directions can be obtained from the input operands as: $D_1^i = sign(X_1^i \cdot X_2^i)$ and $D_2^i = sign(X_1^i \cdot X_3^i)$. Also, the CORDIC processing gain for the 3D Householder CORDIC algorithm can be calculated using the following equation:

$$A^n = \prod 1 + 2^{-2i+1} \tag{5.9}$$

Similarly, the elementary rotation equations for i^{th} iteration of the 4D CORDIC algorithm can be derived as follows:

$$\begin{aligned} X_1^{i+1} &= X_1^i(1 - 3 * 2^{-2i}) + X_2^i(2^{-i+1}D_1^i) + X_3^i(2^{-i+1}D_2^i) + X_4^i(2^{-i+1}D_3^i) \\ X_2^{i+1} &= X_1^i(-2^{-i+1}D_1^i) + X_2^i(1 + 2^{-2i}) + X_3^i(-2^{-2i+1}D_1^iD_2^i) + X_4^i(-2^{-2i+1}D_1^iD_3^i) \\ X_3^{i+1} &= X_1^i(-2^{-i+1}D_2^i) + X_2^i(-2^{-2i+1}D_1^iD_2^i) + X_3^i(1 + 2^{-2i}) + X_4^i(-2^{-2i+1}D_2^iD_3^i) \\ X_4^{i+1} &= X_1^i(-2^{-i+1}D_3^i) + X_2^i(-2^{-2i+1}D_1^iD_3^i) + X_3^i(-2^{-2i+1}D_2^iD_3^i) + X_4^i(1 + 2^{-2i}) \end{aligned} \tag{5.10}$$

where, the rotation directions can be obtained from the input operands as: $D_1^i = sign(X_1^i \cdot X_2^i)$, $D_2^i = sign(X_1^i \cdot X_3^i)$ and $D_3^i = sign(X_1^i \cdot X_4^i)$. Also, the CORDIC processing gain for the 4D Householder CORDIC algorithm can be calculated using the following equation, for n iterations of this 4D Householder CORDIC algorithm:

$$A^n = \prod 1 + 3^{-2i+1} \tag{5.11}$$

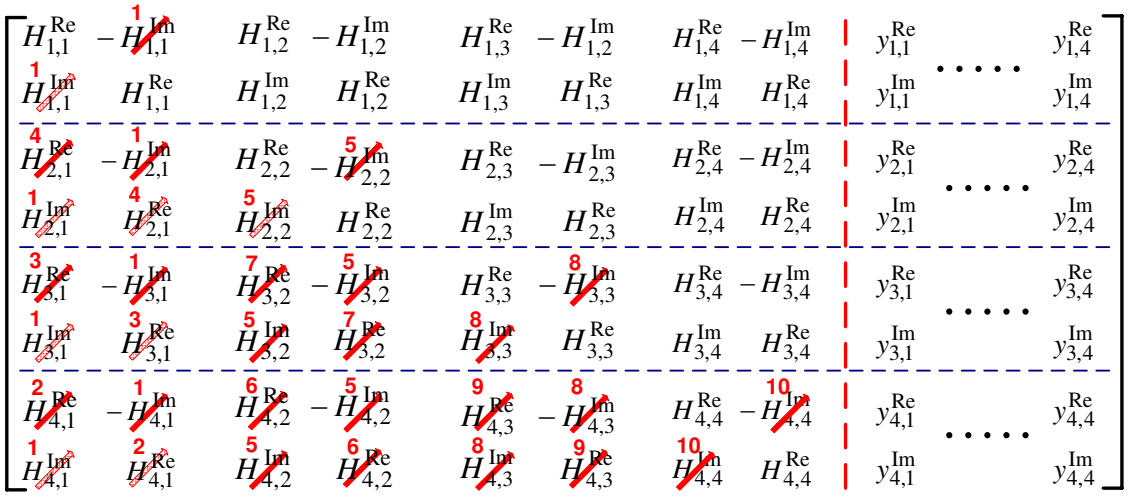

Figure 5.1: Element Annihilation Sequence for the Conventional Givens Rotations QRD Scheme.

5.4 Proposed QR Decomposition scheme

5.4.1 Proposed QRD scheme - for 4×4 MIMO systems

As discussed in Section 5.2, the three popular methods for computing QR Decomposition include the Modified Gram-Schmidt Orthonormalization (MGS) algorithm, Householder transformations and Givens rotations. Among these, the Givens rotations method is superior in terms of BER performance and hardware complexity, compared to the other two methods. However, QR Decomposition of the channel characteristic matrix $\mathbf{H}$ using the conventional sequence of Givens rotations [42] leads to an excessive amount of computations, since it does not exploit the symmetry between the adjacent columns of the $\mathbf{H}$ matrix. For example, QRD of a 4×4 complex channel matrix $\mathbf{H}$ and computation of $\mathbf{z} = \mathbf{Q}^H * \mathbf{y}$ for 4 input 4×1 complex $\mathbf{y}$ vectors using the conventional sequence of Givens rotations requires a total of 26 Vectoring and 200 Rotation operations.

The authors of [43] present a modified sequence of Givens rotations that helps to keep the symmetry of the $\mathbf{H}$ matrix intact during the triangularization process, and hence reduces the number of element annihilations and corresponding Rotation operations required. Note that use of this scheme produces 4 upper-triangular sub-matrices, as shown in [43]. However, if a modified Real Value Decomposition (RVD), as shown in Fig. 5.1, is used to convert the complex 4×4 $\tilde{H}$ matrix to its real counterpart ($\mathbf{H}$), then we can attain a strictly upper-triangular 8×8 real-valued $\mathbf{R}$ matrix using this scheme [44].

Application of the modified sequence of Givens rotations on the **H** matrix created using the modified RVD scheme will keep the symmetry between the adjacent columns intact during the triangularization process. Hence, this will reduce the total number of Vectoring operations required to 16, however, the number of Rotation operations required is 136, which is still very large. This will lead to high computational complexity and hence larger hardware requirements for throughput-constrained systems. For example, in order to attain a QR Decomposition processing latency of 40 clock cycles and assuming that each Vectoring or Rotation operation requires 8 clock cycles (assume 8 iterations of the CORDIC algorithm), the complete QR Decomposition core will require a total of $(16 + 136)/5 = 31$ iterative CORDIC processors. On the other hand, for hardware constrained systems, this will increase the QRD processing latency, and hence reduce throughput by a considerable amount.

By taking a closer look at the triangularization process, it can be noticed that the Vectoring and Rotation operations corresponding to the annihilation of the $H_{i,j}^{Im}$ elements can be performed in a completely parallel manner, since they operate on independent set of rows of the **H** matrix. For example, for the **H** matrix shown in Fig. 5.1, Givens rotations to annihilate the $H_{3,1}^{Im}$ and $H_{4,1}^{Im}$ elements will operate on row pairs 5,6 and 7,8 of the **H** matrix, respectively, and hence they can be executed in parallel. However, annihilation of the $H_{i,j}^{Re}$ elements and their corresponding Rotation operations have to be performed sequentially. For example, the Givens rotations to annihilate the $H_{3,1}^{Re}$ and $H_{4,1}^{Re}$ elements will operate on row pairs 3,5 and 5,7 of the **H** matrix, respectively, and hence they can not be performed in parallel. We must first perform Givens rotations to annihilate the $H_{4,1}^{Re}$ element using the $H_{3,1}^{Re}$ element as a pivot element and then annihilate the $H_{3,1}^{Re}$ element by using $H_{2,1}^{Re}$ as the pivot element. Another issue with the annihilation of $H_{i,j}^{Re}$ elements is that the number of Rotation operations required corresponding to the annihilation of each $H_{i,j}^{Re}$ elements is very large. For example, for annihilation of the $H_{4,1}^{Re}$ element, according to [43], Givens rotations need to be performed on all columns of rows 5,7 and 6,8. Hence, annihilation of each $H_{i,j}^{Re}$ element requires twice the number of Rotation operations, compared to those for $H_{i,j}^{Im}$. Thus, to summarize, the Givens rotations corresponding to the annihilation of the $H_{i,j}^{Re}$ elements contribute the most to the total number of Rotations and they have to be performed sequentially, and hence they cause a throughput bottleneck and increased hardware complexity.

To resolve these issues, this thesis proposes a *hybrid* QR Decomposition scheme that uses a combination of Multi-dimensional Givens rotations, Householder transformations

and the conventional two-dimensional Givens rotations to compute the QR Decomposition of a 4×4 complex channel matrix $\tilde{H}$ and to compute the $\mathbf{z} = \mathbf{Q}^H * \mathbf{y}$ for 4 input 4×1 complex $\mathbf{y}$ vectors. The proposed scheme relieves the throughput bottleneck and reduces the hardware complexity by first decreasing the number of Rotation operations required and then by enabling their parallel execution. The basic idea is to annihilate multiple $H_{i,j}^{Re}$ elements in parallel, by using Multi-dimensional Givens rotations and Householder transformations, and to reduce the circuit complexity by implementing these multi-dimensional vector rotations using CORDIC algorithms that only utilize low-complexity shift and addition operations. Also, for the $H_{i,j}^{Im}$ elements, that do allow parallel Vectoring and Rotation operations, the 2D Givens rotations can be used to perform annihilation with maximum parallelism and minimal complexity.

It should be noted that the proposed scheme also uses the special sequence of element annihilations, presented in [43], that keeps the symmetry between the adjacent columns of $\mathbf{H}$ intact. Hence, the proposed scheme will only need to perform Vectoring and Rotation operations on odd numbered columns of $\mathbf{H}$, and the values for the elements in the even numbered columns can be derived directly, without any computations. Also, the proposed scheme uses the Multi-dimensional CORDIC and Householder CORDIC algorithms, described in Section 5.3 above, to implement Multi-dimensional Givens rotations and Householder transformations for 3D and 4D vectors. The elementary rotation equations for 3D CORDIC, 4D CORDIC, Householder 3D CORDIC and Householder 4D CORDIC algorithms, shown in (5.4), (5.6), (5.8) and (5.10) respectively, were compared for their implementation complexity. The comparison results were then used to make the decision about which algorithms to use for 3D and for 4D vector rotations. It was decided to use Householder CORDIC algorithms for 3D vector rotations and the Multi-dimensional CORDIC algorithms for 4D vector rotations.

The proposed QR Decomposition scheme for 4×4 complex matrix is shown in Table 5.1. The algorithm begins with annihilating the $H_{i,1}^{Im}$ elements in the first column of the $\mathbf{H}$ matrix. As mentioned above, the Vectoring and Rotation operations corresponding to the annihilation of the $H_{i,1}^{Im}$ elements can be performed in a completely parallel manner, and hence the conventional 2D Givens rotations are used for these element annihilations. After the nullification of the $H_{i,1}^{Im}$ elements in the first column of $\mathbf{H}$, the algorithm uses 4D Givens rotations [40] to annihilate the elements $H_{4,1}^{Re}$, $H_{3,1}^{Re}$ and $H_{2,1}^{Re}$ simultaneously. As mentioned above, using the conventional 2D Givens rotations, the annihilation of these elements had to be performed sequentially, which led to very large number of sequential

Table 5.1: The proposed *hybrid* QR Decomposition Scheme for 4×4 complex matrix.

1) Annihilate $H_{1,1}^{Im}$, $H_{2,1}^{Im}$, $H_{3,1}^{Im}$ and $H_{4,1}^{Im}$ using 2D CORDIC algorithm.
2) Annihilate ($H_{2,1}^{Re}$, $H_{3,1}^{Re}$, $H_{4,1}^{Re}$) using 4D CORDIC algorithm.
3) Annihilate $H_{2,2}^{Im}$, $H_{3,2}^{Im}$ and $H_{4,2}^{Im}$ in parallel using 2D CORDIC algorithm.
4) Annihilate ($H_{3,2}^{Re}$, $H_{4,2}^{Re}$) using Householder 3D CORDIC algorithm.
5) Annihilate $H_{3,3}^{Im}$ and $H_{4,3}^{Im}$ using 2D CORDIC algorithm.
6) Annihilate $H_{4,3}^{Re}$ using 2D CORDIC algorithm.
7) Annihilate $H_{4,4}^{Im}$ using 2D CORDIC algorithm.

Rotation operations and hence a throughput bottleneck. However, using the 4D Givens rotations, the annihilation is performed in parallel and the corresponding number of Rotation operations has been reduced by a factor of 3. Specifically, the 4D Givens rotations propagate the effect of $H_{i,1}^{Re}$ element annihilation to rows 1,2,3,4 and 5,6,7,8 simultaneously, and hence reduces the number of Rotation operations required from 42 to 14.

As the next step, the conventional 2D Givens rotations are used once again to perform parallel annihilation of the $H_{i,2}^{Im}$ elements in the third column of the H matrix. The scheme then uses the 3D Householder CORDIC algorithm [41] to annihilate $H_{4,2}^{Re}$ and $H_{3,2}^{Re}$ simultaneously. The effect of element annihilation is propagated to non-zero elements in rows 2,3,4 and 6,7,8 in parallel, and this further reduces the number of corresponding rotation operations by a factor of 2. As the last step, the algorithm annihilates the $H_{3,5}^{Im}$, $H_{4,3}^{Im}$, $H_{4,3}^{Re}$ and $H_{4,4}^{Re}$ elements, in the order given, using the conventional 2D Givens rotations.

Fig. 5.2 demonstrates the annihilation order used in the proposed scheme, where the number on top of each arrow shows the sequential step number in the annihilation process. For example, the number "1" on top of arrows for $H_{1,1}^{Im}$, $H_{2,1}^{Im}$, $H_{3,1}^{Im}$ and $H_{4,1}^{Im}$ demonstrate that all of these four elements are annihilated in step 1, in a completely parallel manner. Also, from Fig. 5.1 and Fig. 5.2, it can be noticed that the proposed scheme annihilates the $H_{2,1}^{Re}$, $H_{3,1}^{Re}$ and $H_{4,1}^{Re}$ elements simultaneously in step 2, as opposed to the conventional

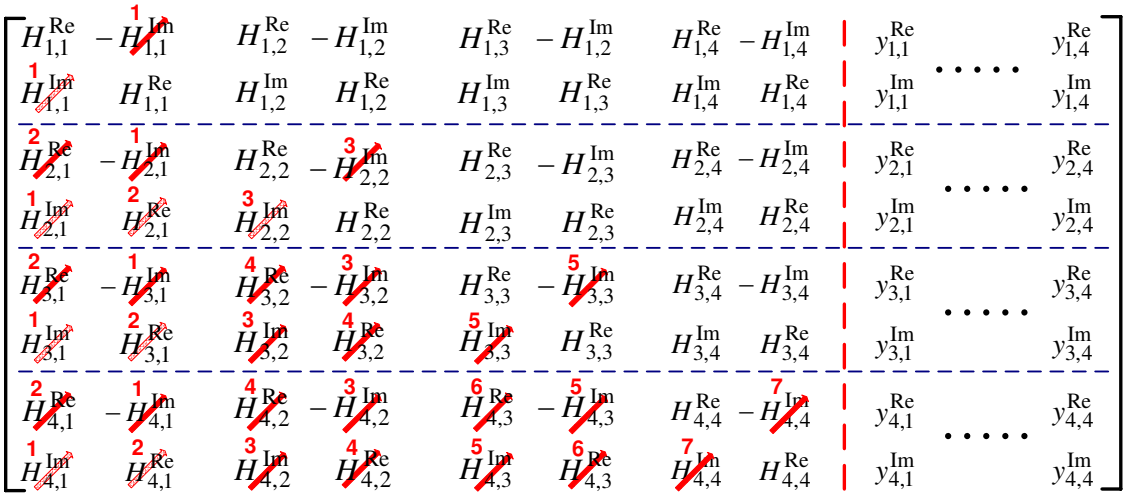

Figure 5.2: Element Annihilation Sequence for the Proposed QRD Scheme.

Givens rotation scheme that annihilates these elements in 3 sequential steps (steps 2, 3 and 4 in Fig. 5.1). The similar observation can be made for annihilation of elements $H^{Re}_{3,2}$ and $H^{Re}_{4,2}$. Thus, the proposed scheme reduces the number of sequential steps required, and hence reduces the overall QRD processing latency.

5.4.2 Proposed QRD scheme - Generalization

The proposed QRD scheme is presented in Section 5.4.1 for decomposition of a 4×4 complex channel characteristic matrix **H**. However, it can be generalized to perform QR Decomposition of matrices of any size, by appropriately using the 2D and 4D CORDIC algorithms and the Householder 3D CORDIC algorithm. The generalization of the proposed scheme for QR Decomposition of an $n \times n$ complex matrix can be performed using the following ideas:

1. For element annihilations that operate on independent sets of rows of **H** and can be parallelized, use the conventional 2D CORDIC algorithm to attain maximum parallelization and minimal computational complexity.

2. For sequential element annihilations that use common pivot rows, use Householder 3D CORDIC and 4D CORDIC algorithms to nullify these elements simultaneously, and hence reduce latency by a factor of 2 and 3, respectively, and also remove the throughput bottleneck.

For example, an extended QRD scheme for 6×6 complex channel matrix, derived using the ideas listed above, is shown in Table 5.2. As shown, this scheme requires a total of 11 sequential steps that use either of 2D, Householder 3D or 4D CORDIC algorithms to nullify the elements of the input 6×6 complex **H** matrix, to convert it to an upper-triangular **R** matrix. Note that, according to [40], for vector rotations with dimensions larger than 4, the multiply-and-accumulate based algorithms offer lower computational complexity compared to the Multi-dimensional and Householder CORDIC algorithms. Hence, the extensions of the proposed QRD scheme to $n \times n$ matrices should avoid using CORDIC algorithms of dimensions larger than 4.

The proposed scheme shown in Table 5.2 for QRD of 6×6 complex matrix uses the low-complexity 2D CORDIC algorithm [36] to annihilate the necessary $H_{i,j}^{Im}$ elements in parallel. For annihilation of the $H_{i,j}^{Re}$ elements, the proposed scheme uses the optimal number of 4D, Householder 3D and 2D CORDIC vector rotations, such that the number of sequential steps required to annihilate the necessary $H_{i,j}^{Re}$ elements in each column is minimized. For example, for column 1 of 6×6 **H** matrix, the proposed QRD scheme needs to annihilate a total of 5 $H_{i,1}^{Re}$ elements. Hence, the proposed scheme first uses the 4D CORDIC algorithm to annihilate $H_{6,1}^{Re}$, $H_{5,1}^{Re}$ and $H_{4,1}^{Re}$ elements simultaneously, and then uses the Householder 3D CORDIC algorithm to nullify the remaining 2 elements, $H_{3,1}^{Re}$ and $H_{2,1}^{Re}$. Thus, all 5 elements in column 1 are annihilated using only 2 sequential steps, as opposed to 5 sequential steps required using the 2D CORDIC algorithm.

As can be derived from Table 5.2, the proposed scheme requires 3 4D Vectoring, 24 4D Rotation, 2 Householder 3D Vectoring, 14 Householder 3D Rotation, 23 2D Vectoring and 80 2D Rotation operations to perform QRD of a 6×6 complex matrix. Through mathematical analysis, the number of 2D, Householder 3D and 4D Vectoring and Rotation operations required for QR Decomposition of an $n \times n$ complex matrix have been derived as shown in Table 5.3. Thus, from these equations, QRD of 4×4 complex matrix requires 1 4D Vectoring, 6 4D Rotation, 1 Householder 3D Vectoring, 4 Householder 3D Rotation, 11 2D Vectoring and 21 2D Rotation operations. Furthermore, QRD of an 8×8 complex matrix requires 7 4D Vectoring, 76 4D Rotation, 2 Householder 3D Vectoring, 14 Householder 3D Rotation, 39 2D Vectoring and 191 2D Rotation operations.

To summarize, the generalization of the proposed QRD scheme for processing $n \times n$ complex matrices can be performed by appropriately utilizing the 2D, Householder 3D and 4D CORDIC algorithms, according to the rules mentioned above, depending on the type and number of element annihilations to be performed. As discussed above, the proposed

Table 5.2: The proposed *hybrid* QR Decomposition Scheme for 6×6 complex matrix.

1) Annihilate $H_{1,1}^{Im}$, $H_{2,1}^{Im}$, $H_{3,1}^{Im}$, $H_{4,1}^{Im}$, $H_{5,1}^{Im}$ and $H_{6,1}^{Im}$ in parallel using 2D CORDIC algorithm.

2) Annihilate ($H_{6,1}^{Re}$, $H_{5,1}^{Re}$, $H_{4,1}^{Re}$) using 4D CORDIC algorithm.

3) Annihilate ($H_{3,1}^{Re}$, $H_{2,1}^{Re}$) using Householder 3D CORDIC algorithm.

4) Annihilate $H_{2,2}^{Im}$, $H_{3,2}^{Im}$, $H_{4,2}^{Im}$, $H_{5,2}^{Im}$ and $H_{6,2}^{Im}$ in parallel using 2D CORDIC algorithm.

5) Annihilate ($H_{6,2}^{Re}$, $H_{5,2}^{Re}$, $H_{4,2}^{Re}$) using 4D CORDIC algorithm.

6) Annihilate $H_{3,2}^{Re}$ using 2D CORDIC algorithm.

7) Annihilate $H_{3,3}^{Im}$, $H_{4,3}^{Im}$, $H_{5,3}^{Im}$ and $H_{6,3}^{Im}$ in parallel using 2D CORDIC algorithm.

8) Annihilate ($H_{6,3}^{Re}$, $H_{5,3}^{Re}$, $H_{4,3}^{Re}$) using 4D CORDIC algorithm.

9) Annihilate $H_{4,4}^{Im}$, $H_{5,4}^{Im}$ and $H_{6,4}^{Im}$ in parallel using 2D CORDIC algorithm.

10) Annihilate ($H_{6,4}^{Re}$, $H_{5,4}^{Re}$) using Householder 3D CORDIC algorithm.

11) Annihilate $H_{5,5}^{Im}$ and $H_{6,5}^{Im}$ in parallel using 2D CORDIC algorithm.

12) Annihilate $H_{6,5}^{Re}$ using 2D CORDIC algorithm.

13) Annihilate $H_{6,6}^{Im}$ using 2D CORDIC algorithm.

Table 5.3: Equations for number of 2D, Householder 3D and 4D Vectoring and Rotation Operations Required for QRD of an $n \times n$ Complex Matrix

Operation	**Number of Operations Required for $n \times n$ Complex Matrix**
4D Vectoring	$\sum_{i=1}^{n-2}(\lfloor \frac{n-i}{3} \rfloor)$
4D Rotation	$\sum_{i=1}^{n-2}[(\lfloor \frac{n-i}{3} \rfloor) \times 2(n-i)]$
Householder 3D Vectoring	$\sum_{i=1}^{n-2}(\lfloor \frac{(n-i) - (3\lfloor \frac{n-i}{3} \rfloor)}{2} \rfloor)$
Householder 3D Rotation	$\sum_{i=1}^{n-2}[(\lfloor \frac{(n-i) - (3\lfloor \frac{n-i}{3} \rfloor)}{2} \rfloor) \times 2(n-i)]$
2D Vectoring	$\sum_{i=1}^{n-2}[(n-i) - (3\lfloor \frac{n-i}{3} \rfloor) - (2\lfloor \frac{n-i}{2} \rfloor)] + \sum_{i=1}^{n}(n-i+1) + 1$
2D Rotation	$\sum_{i=1}^{n-2}[(n-i) - (3\lfloor \frac{n-i}{3} \rfloor) - (2\lfloor \frac{n-i}{2} \rfloor)] \times (n-i) + \sum_{i=1}^{n}(n-i+1) \times (n-i) + 1$

QRD scheme reduces the number of sequential annihilation steps required significantly, and hence reduces the QRD processing latency and removes the throughput bottleneck, compared to the existing Givens rotations based QRD schemes. The proposed scheme also utilizes the low-complexity CORDIC algorithms for two-dimensional and multi-dimensional vector rotations, that results in a substantial reduction in the computational complexity required for QRD calculation.

5.5 Proposed QR Decomposition - Overall Architecture Description

Emerging 4G wireless standards require QR Decomposition implementations for processing large complex channel matrices, while minimizing QRD processing latency, silicon area and power consumption requirements. For decomposition of large complex matrices, the existing QRD schemes lead to high computational complexity, sequential throughput bottleneck and lack of parallelism. Hence, the published QRD VLSI implementations, using these existing schemes, either lead to large QRD processing latency or to large silicon area

and power requirements. In Section 5.4, we proposed a *hybrid* QR Decomposition scheme that uses a unique combination of Multi-dimensional Givens rotations, Householder transformations and the conventional 2D Givens rotations to both reduce the computational complexity and achieve higher execution parallelism. This hybrid QRD scheme is utilized in this section to develop a VLSI architecture for a QRD core to decompose a 4×4 complex channel matrix $\mathbf{H}$, and compute updated symbol vectors $\mathbf{z} = \mathbf{Q}^H * \mathbf{y}$ for four received 4×1 complex $\mathbf{y}$ vectors.

As mentioned earlier, it is desired to develop a QRD architecture that decomposes large channel matrices with minimal QRD processing latency, and also minimizes gate count and power consumption requirements as much as possible. Considering the large number of Vectoring and Rotation operations that need to be performed to output 4×4 complex $\mathbf{R}$ matrix and four 4×1 complex $\mathbf{z}$ vectors, an architecture with linear or triangular systolic arrays will require an extremely large amount of hardware resources [45]. On the other hand, an iterative architecture where one or more CORDIC processors are used repeatedly to perform the complete QR Decomposition, will have much smaller silicon area and power requirements, however, they will lead to very large QRD processing latency, due to the large number of Vectoring and Rotation operations that need to be done iteratively [46]. In order to perform the large number of Vectoring and Rotation operations required to compute a new $\mathbf{R}$ matrix and 4 complex $\mathbf{z}$ vectors in the smallest possible number of cycles, while also minimizing the area and power requirements, this thesis proposes an efficient semi-pipelined semi-iterative architecture that uses un-rolled CORDIC processors iteratively, along with complex controllers, to maximize throughput and resource utilization, while minimizing the area and power requirements. The proposed QRD architecture attains a processing latency of merely 40 clock cycles at 278 MHz, while occupying 0.3mm^2 core area (36KG) and dissipating 48.2mW at 1.32V supply.

Fig. 5.3 shows the overall architecture of the proposed QR Decomposition core. The overall architecture consists of a total of 6 pipelined stages, each with latency less than or equal to 40 cycles. The first stage is an `Input Controller` stage, that provides the interface with the preceding stage in the MIMO Receiver. This stage serves the purpose of reading in one 4×4 complex $\mathbf{H}$ matrix and four 4×1 complex $\mathbf{y}$ vectors every 40 clock cycles from the preceding stage. The `Input Controller` stage then stores the read data and uses them to supply appropriate input operands to the CORDIC processors. The last stage in the QRD architecture is an `Output Controller` stage that serves the purpose of transferring the output 4×4 complex $\mathbf{R}$ matrix and four output 4×1 complex $\mathbf{z}$ vectors

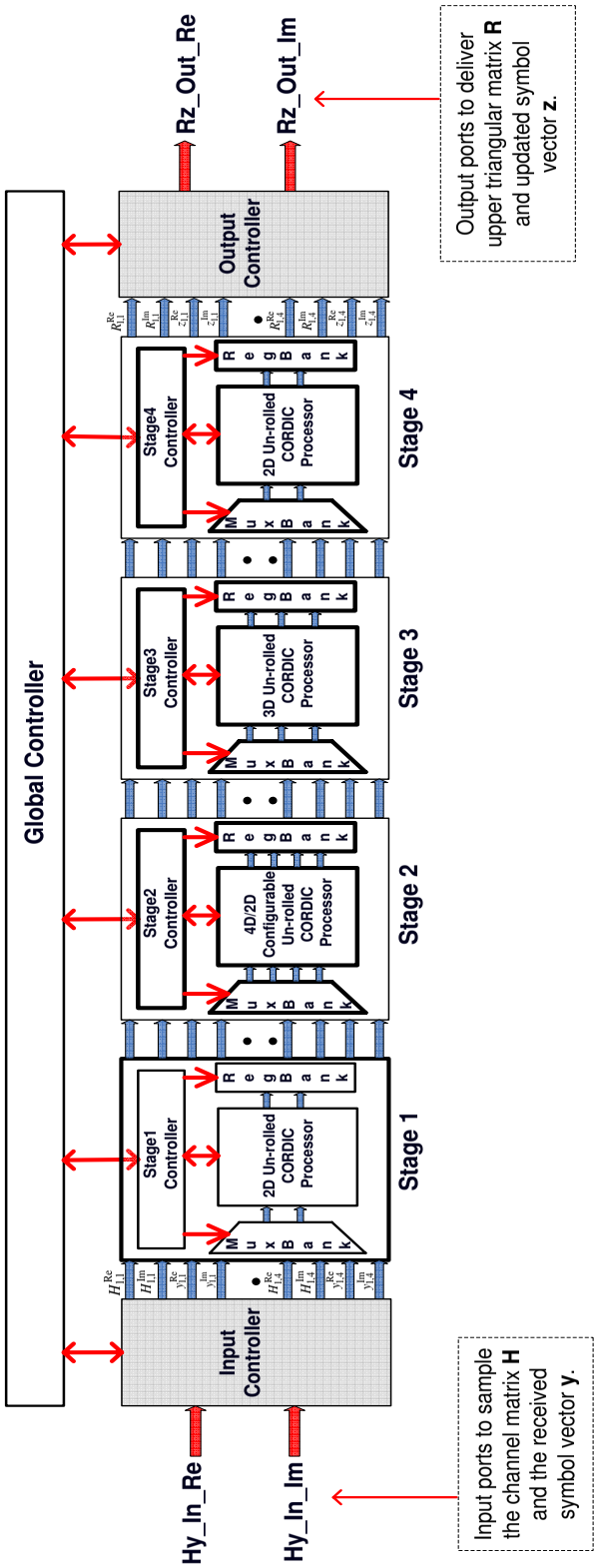

Figure 5.3: Overall Architecture of the Proposed QR Decomposition Core.

($\mathbf{z} = \mathbf{Q}^H * \mathbf{y}$) to the succeeding K-Best MIMO Detector block every 40 clock cycles. Both the `Input Controller` and the `Output Controller` blocks read in or write out 1 complex number (2 16-bit Real numbers) each cycle, by following the scheduling shown in Table C.1 in Appendix C.

The four central stages, `Stage1-4`, compute the QR Decomposition of input $\mathbf{H}$ matrix, as well as 4 $\mathbf{z}$ vectors, simultaneously, using un-rolled pipelined 2D, Householder 3D and 4D/2D Configurable CORDIC processors. The details about functionality and architecture of these un-rolled CORDIC processors will be provided in Section 5.6. As shown in Fig. 5.3, each of these four central stages also contains a multiplexor (MUX) bank (`MuxBank`) and a register bank (`RegBank`) in the datapath, in addition to the un-rolled CORDIC processors. In each stage, the `MuxBank` serves the purpose of selecting the input operands for the CORDIC processor in that stage every clock cycle. The `RegBank` at the output of each stage is used to re-direct the CORDIC outputs to appropriate registers and to hold them until the current stage completes its desired computations and all outputs are ready to be passed to the next stage as inputs.

In terms of the control path, each of these stages contains an independent `Stage Controller` that controls the operation of the datapath modules, to enable them to perform the required operations within the given number of clock cycles. Specifically, the `Stage Controller` provides the select and other control signals to direct appropriate data in and out of the CORDIC processor every cycle. The `Stage Controller` also provides the required control signals to the CORDIC processors to control their mode of operation (Vectoring or Rotation), rotation direction transfers and re-use of the pipelined CORDIC stages to maximize resource utilization. In addition to the individual stage controllers, the QRD architecture also contains a `Global Controller` that controls the overall operation of the complete QRD core. As mentioned, all 6 pipelined stages perform a certain fixed set of tasks every 40 clock cycles, independently of each other. Hence, the `Global Controller` contains a counter that provides a global count (from 1 to 40), in order to synchronize the operation of each of the 6 stages. The `Global Controller` also provides the required control signals to ensure correct functionality of the `Input Controller` and `Output Controller` blocks, as well as `Stage Controller` blocks within each central stage.

Note that the data format for input, output and internal data for the QR Decompostion core, is signed numbers in two's complement format with a word-length of 16 bits and 11 bits for the fractional part. Note that due to their larger dynamic range, the Householder

3D CORDIC processor uses an extra bit for the integer part, and hence requires a word-length of 17 bits. These word-length and number of bits for fractional part were derived using extensive bit-true simulations, such that the BER loss relative to the floating-point result is minimized. Also, the total latency of the architecture from input **H** and **y** matrices to output **R** and **z** matrices is 160 clock cycles. However, due to its deeply pipelined nature, the QRD architecture processes a new set of **H** and **y** matrices , and produces a new set of **R** and **z** output matrices every 40 clock cycles.

5.6 Proposed QR Decomposition - Detailed VLSI Architecture

5.6.1 CORDIC Processors - Proposed General Architecture

The CORDIC algorithm uses a series of shift and addition operations to evaluate many basic arithmetic and mathematical functions [47]. It is also very suitable for implementing Givens rotations, using its Vectoring and Rotation modes [36]. There are a number of ways to design the CORDIC processors, that implement the CORDIC algorithms. Hence, the architecture of the CORDIC processor, for the given application, depends on the latency and hardware resource constraints.

For the QR Decomposition architecture under consideration, a large number of Vectoring and Rotation operations need to be performed at each pipelined stage within 40 clock cycles, while trying to achieve the smallest gate count possible. In other words, the architectures for the CORDIC processors need to be designed with the primary aim of achieving high throughput, possibly performing 1 Vectoring or Rotation operation every cycle. And then, as the secondary aim, the area of the CORDIC processors should be reduced as much as possible, using various strategies. Iterative CORDIC processors provide a minimum hardware solution, however, they have a considerably large processing latency [36]. On the other hand, fully un-rolled Pipelined CORDIC processors offer very high throughput, however, their straightforward implementation poses very large resource requirements [36]. Hence for the novel QR Decomposition architecture in this thesis, we propose to use an un-rolled, deeply pipelined architecture with iterative stages to design

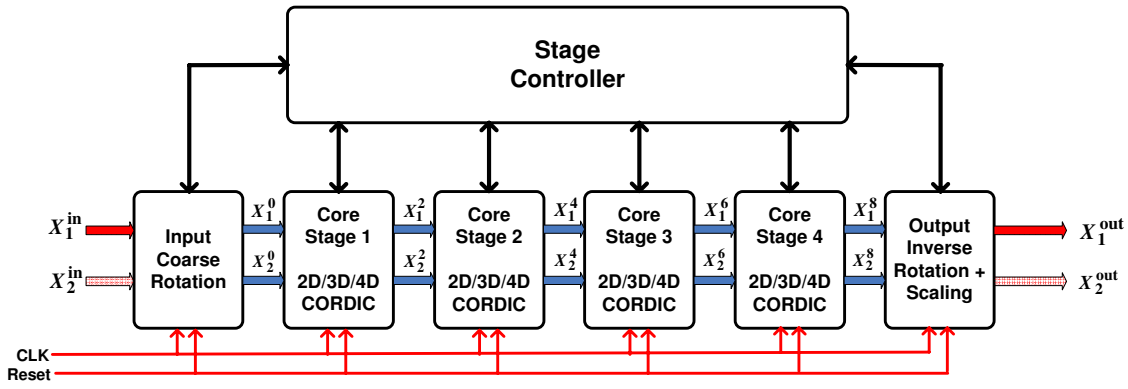

Figure 5.4: General Architecture of CORDIC Processors Used in the Proposed QRD Core.

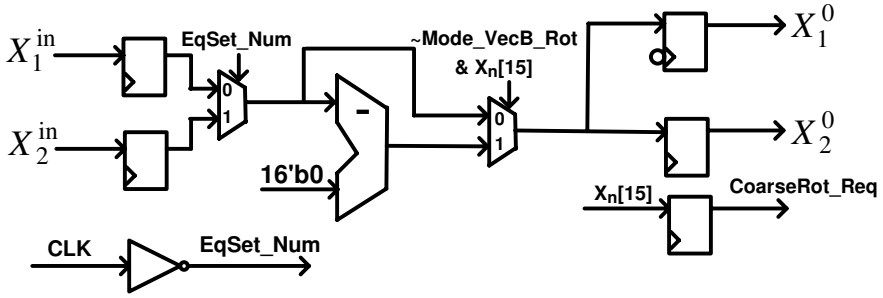

Figure 5.5: Architecture of the Input Coarse Rotation stage for 2D CORDIC Processors.

the 2D, Householder 3D and 4D/2D Configurable CORDIC processors, with major modifications to reduce the gate count and the number of cycles required for complete Vectoring and Rotation operations.

Fig. 5.4 shows the general architecture of the CORDIC processors used in the proposed QRD core in this thesis. In general, the 2D, Householder 3D and 4D/2D Configurable CORDIC processors consist of multiple pipelined core stages, where each stage implements one or more of the CORDIC elementary rotation equations. Each CORDIC Core stage is designed to work in either Vectoring or Rotation mode, which in turn is controlled by the `Stage Controller` block. In addition to the Core stages that implement the elementary rotation equations, the CORDIC processors also include an input coarse rotation stage and an output stage that performs both inverse coarse rotation and output scaling. The architectures for these stages are shown in Fig. 5.5 and Fig. 5.6, respectively. The CORDIC Vectoring and Rotation algorithms are limited to rotation angles between -$\pi/2$ and +$\pi/2$, and hence for composite rotation angles larger than $\pi/2$, the input and output coarse rotation stages rotate the input and output vectors by -π and +π, respectively. From a hardware perspective, this is implemented in the input coarse rotation stage by

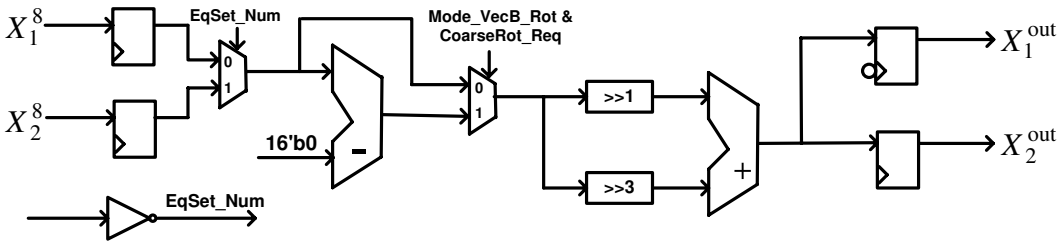

Figure 5.6: Architecture of the Output Coarse Rotation + Scaling stage for 2D CORDIC Processors.

simply monitoring the sign of the input operands and negating (using the two's complement scheme) them, if required. Based on the input operands, the input coarse rotation stage generates the control signal "CoarseRot_Req", which is then used by the output inverse coarse rotation stage to decide whether to perform inverse rotation or not.

The output scaling stage scales the CORDIC outputs by a constant factor, in order to compensate for the CORDIC processing gain, described in Sections 5.3.1, 5.3.3 and 5.3.2. The proposed QRD architecture approximates the scaling operation to reduce the circuit complexity. For example, for the 2D CORDIC case, implementation of the exact scaling by factor 0.6097 requires signed multipliers. However, approximation of this scaling by $2^{-1} + 2^{-3}$ (0.6250) will allow its implementation with considerably lower circuit complexity, by only using hardwired shifts and signed addition, as shown in Fig. 5.6. Similarly, the Householder 3D and 4D CORDIC processors use approximate scale factors of 0.1875 ($2^{-3} + 2^{-4}$) and 0.3125 ($2^{-2} + 2^{-4}$), respectively. The impact of these approximations on the BER performance is very minor, which will be discussed in detail in Section 5.7. Also, note that based on MATLAB simulations, architectural decisions were made to use 8 CORDIC iterations.

5.6.2 2D CORDIC Processor

The 2D CORDIC algorithm was described in Section 5.3.1, as a method to implement the Vectoring and Rotation operations for Givens rotations. The CORDIC elementary rotation equations, shown in (5.1), are used to implement both Vectoring and Rotation operations, where the elementary rotation direction is calculated using the input vector co-ordinates for the Vectoring case, and using the residual angle for the Rotation case. Hence, a single stage architecture for the 2D CORDIC processor can be designed to be programmable to

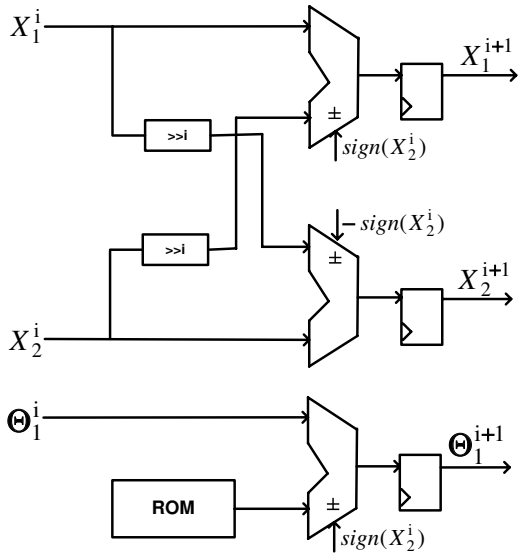

Figure 5.7: Conventional Architecture of a single pipeline stage for 2D CORDIC Processor.

execute appropriate CORDIC equations, depending on its mode of operation (Vectoring or Rotation). Fig. 5.7 shows the generic single stage architecture for 2D un-rolled CORDIC processor, derived using equation (5.1). As shown, the architecture uses a ROM module and a large amount of hardware resources for the angle datapath. Also, since it only implements one iteration of the CORDIC elementary rotation equation, it leads to low resource utilization and large hardware requirements. For example, for a 2D un-rolled CORDIC processor with 8 CORDIC iterations, a total of 8 instances of this single stage architecture will be required. The resulting CORDIC processor will take 8 clock cycles to complete Vectoring or Rotation operation for a single 2D vector and will require a total of 24 adder and 8 ROM modules.

The proposed single stage architecture, shown in Fig. 5.8, for the 2D un-rolled CORDIC processor resolves these issues using two major improvement strategies, namely: implicit angle transfer and re-use of hardware resources for execution of multiple CORDIC iterations in a single clock cycle using the same single stage. The idea of implicit angle transfer simply computes the elementary rotation directions in the Vectoring mode, stores them in the `Stage Controller` registers, and utilizes them directly in the Rotation mode of operation [41]. Thus, the CORDIC processor does not need to explicitly compute the rotation angle in the Vectoring mode and does not need to utilize and keep track of it to derive the elementary rotation directions. This results in hardware savings of around 30%, since the adders, registers, MUXs and the ROM that make up the angle datapath for each

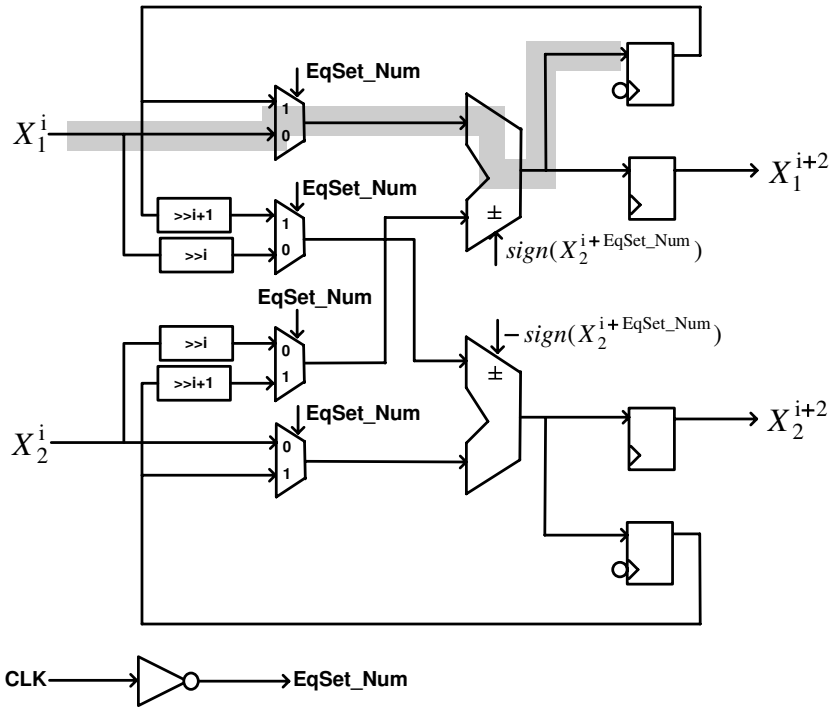

Figure 5.8: Proposed Architecture of a single pipeline stage for 2D CORDIC Processor - with the critical path highlighted.

CORDIC stage can be removed.

The second improvement strategy proposes to implement 2 sets of 2D CORDIC elementary rotation equations using the same single stage. The idea is to use the same set of 16-bit signed adders twice and use MUXs to select inputs to these adders, with the clock signal acting as the MUX select signal. In the first half of the clock cycle, the inputs corresponding to the first set of elementary equations are passed to the adders. The adder outputs, from the first half of the clock cycle processing, are used as the adder inputs for the second half of the clock cycle. For each elementary equation set implementation, the elementary rotation directions are computed from the input operands, and are used to operate the signed adders in either addition or subtraction mode. Thus, using this strategy, only 4 instances of the single stage architecture shown in Fig. 5.8 will be required for the 2D un-rolled CORDIC processor with 8 CORDIC iterations. The resulting CORDIC processor will take 4 clock cycles to complete Vectoring or Rotation operation for a single 2D vector and will require a total of 8 adders and 0 ROM modules. Thus, this reduces the number of cycles required for Vectoring and Rotation operations required by a factor of 2, the amount of hardware required by a factor of 3 and increases the datapath hardware utilization to approximately 100%.

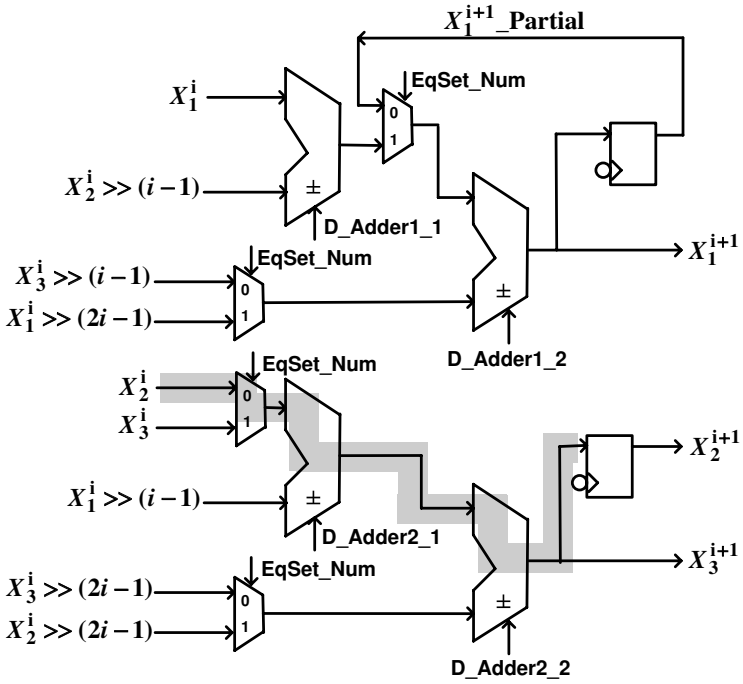

Figure 5.9: Proposed Architecture of a single pipeline stage for 3D CORDIC Processor - with the critical path highlighted.

Also, since each CORDIC single stage needs to perform fixed shift, it can be performed using re-wiring of the input operands and hence the area intensive Barrel Shifters can be removed [36]. Note that identical architectures of the 2D CORDIC processors are used in Stage 1 and Stage4 of the proposed QR Decomposition core. In Stage 1, it performs 4 2D Vectoring and 24 2D Rotation operations, and in Stage 3, it performs 3 2D Vectoring and 24 2D Rotation operations within 40 clock cycles.

5.6.3 Householder 3D CORDIC Processor

The 3D un-rolled CORDIC processor performs Vectoring and Rotation operations on three-dimensional column vectors (3×1 real-valued vectors) using the Householder 3D CORDIC elementary rotation equations, shown in equation (5.8). The 3D un-rolled CORDIC processor consists of 4 pipelined single stages, each of which implements 2 sets of Householder 3D elementary rotation equations, within 2 clock cycles. Fig. 5.9 shows the architecture of a single stage of the Householder 3D un-rolled CORDIC processor, with critical path highlighted. Note that this architecture also uses the same area saving strategies that were

used for the 2D CORDIC processor, described in Section 5.6.2.

As shown in Fig. 5.9, the four signed adders are used a total of 4 times, within 2 clock cycles, to implement the 2 iterations of the Householder 3D CORDIC equations. Note again that the MUXs, controlled by the clock signal, are used to provide appropriate inputs to the adders. The top two adders compute X_1^{i+1} by adding the 4 terms in the first equation, shown in (5.8). The bottom two adders compute X_2^{i+1} and X_3^{i+1}, in each half of a single clock cycle. The outputs X_1^{i+1}, X_2^{i+1} and X_3^{i+1} are then fed back as inputs to the same single stage, and the same procedure is used to compute X_1^{i+2}, X_2^{i+2} and X_3^{i+1}, which serve as the final outputs of the single stage. Note that the Householder 3D CORDIC processor is used in the QR Decomposition Stage 3, and it performs 1 3D Vectoring and 12 3D Rotation operations within 34 clock cycles.

5.6.4 4D/2D Configurable CORDIC Processor

Stage 2 of the proposed QR Decomposition core contains a 4D/2D configurable un-rolled CORDIC processor. This CORDIC processor consists of 8 pipelined single stages, each of which is programmable to operate in either 4D or 2D mode. In the 2D mode of operation, each single stage of the 4D/2D configurable CORDIC processor can perform 4 2D Vectoring or Rotation operations in parallel, in a single clock cycle. In the 4D mode of operation, it can perform a single Vectoring or Rotation operation, by implementing the 4 elementary rotation equations shown in 5.6, within a single clock cycle.

Fig. 5.10 shows the architecture of a single stage of the 4D/2D configurable un-rolled CORDIC processor, with critical path highlighted. The MUXs select the input data and rotation directions for the adders according to the mode of operation (2D or 4D). Since, in the 4D mode of operation, each of these adders are used twice, the clock signal driven MUXs have also been cascaded to select the adder inputs. For the 4D mode of operation, the adders are used to compute X_1^{i+1} and X_2^{i+1} in the first half of the clock cycle, and to compute X_3^{i+1} and X_4^{i+1} in the second half of the clock cycle. In the 2D mode of operation, the CORDIC processor performs Vectoring and Rotation operations on two sets 2D vectors, $[X_1^i \ X_2^i]^T$ and $[X_3^i \ X_4^i]^T$, in parallel, in each half of the clock cycle. In other words, the adders compute $[X_1^{i+1} \ X_2^{i+1}]^T$ and $[X_3^{i+1} \ X_4^{i+1}]^T$ in the first half of the clock cycle, for the first set of two 2D input vectors. The same process is repeated in the second half of the clock cycle to compute updated vectors for the second set of two 2D input vectors.

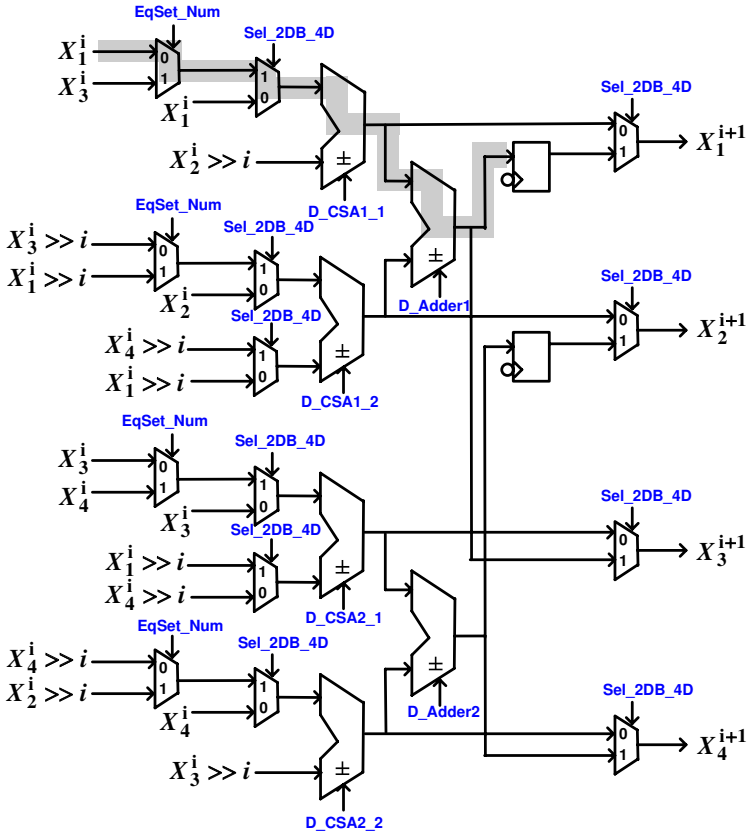

Figure 5.10: Proposed Architecture of a single pipeline stage for the 4D/2D Configurable CORDIC Processor - with the critical path highlighted.

The 4D/2D configurable un-rolled CORDIC processor performs a total of 1 4D Vectoring, 14 4D Rotation, 3 2D Vectoring and 18 2D Rotation operations within 36 clock cycles. This makes Stage 2 the most computation and hardware intensive stage within the complete QR Decomposition core. Also, due to the hardware intensive nature, the critical path of the single stage of the 4D/2D configurable CORDIC processor accounts for the critical path of the overall QR Decomposition core.

5.7 BER Simulation Results

The QR Decomposition operation does not directly estimate the transmitted vector, and hence it does not have a direct impact on the BER performance. However, the accuracy of the channel matrix QR Decomposition does have an effect on the MIMO detection

process. In other words, a QRD core decomposes the channel matrix **H** and processes received symbol vectors **y**, to produce an upper-triangular matrix **R** and updated symbol vectors **z** ($\mathbf{z} = \mathbf{Q}^H * \mathbf{y}$). The matrix **R** and symbol vectors **z** are then used by the MIMO detector to estimate the transmitted vectors $\hat{s}$. These estimated transmitted vectors, $\hat{s}$, are then compared to the actual transmitted vectors, **s**, to quantize the BER performance of the MIMO Receiver (QRD + MIMO Detector).

Thus, sources of inaccuracy in QR Decomposition will produce **R** matrix and **z** vectors that may not be exactly the same as the actual **R** and **z** attained mathematically (using ideal QRD) in floating-point format. Use of these perturbed **R** and **z** matrices for MIMO detection would lead to errors in estimating the transmitted vector $\hat{s}$, which then results in BER performance degradation compared to ideal QRD. In this thesis, the BER performance of the proposed QRD scheme and its VLSI implementation is quantified by comparing floating-point and fixed-point QRD models when combined with a 4×4 64-QAM Hard-output K-Best detector, with $K = 10$. The combined MATLAB models were simulated for 100,000 packets, where each packet consists of $4 \times log_2(Q) \times N_T = 4 \times 6 \times 4 = 96$ bits (9.6Mbits in total) for 4×4 MIMO system.

As mentioned in Section 5.6.1, the proposed QRD architecture uses approximations for the actual scale factors for compensating CORDIC processing gain, in order to simplify the VLSI implementation of the scaling operation. In other words, the scale factors 0.6097, 0.1896 and 0.3364 for 2D, Householder 3D and 4D CORDIC processors have been approximated with 0.6250 ($2^{-1} + 2^{-3}$), 0.1875 ($2^{-3} + 2^{-4}$) and 0.3125 ($2^{-2} + 2^{-4}$), respectively. However, as mentioned above, these approximations might lead to increased inaccuracy in the resulting **R** matrix and **z** vectors, and hence might cause BER performance degradation. Fig. 5.11 shows the BER performance of the QRD + K-Best MIMO detector integration with actual and approximated scale factors. As shown, the approximation of the actual scale factors only leads to a BER performance loss of 0.14 dB at BER = 10^{-3}. However, in terms of hardware implementation, this leads to significant hardware savings since the actual signed multiplication can be implemented using only hardwired shifts and signed addition.

In the proposed QRD scheme, the CORDIC algorithms are utilized to perform vector rotation. The accuracy of the vector rotations depends on the number of iterations used for the CORDIC algorithm. In other words, a larger number of CORDIC iterations will lead to better accuracy, however, will also lead to larger hardware complexity. Hence, in order to decide on the number of CORDIC algorithm iterations to be used, the MATLAB

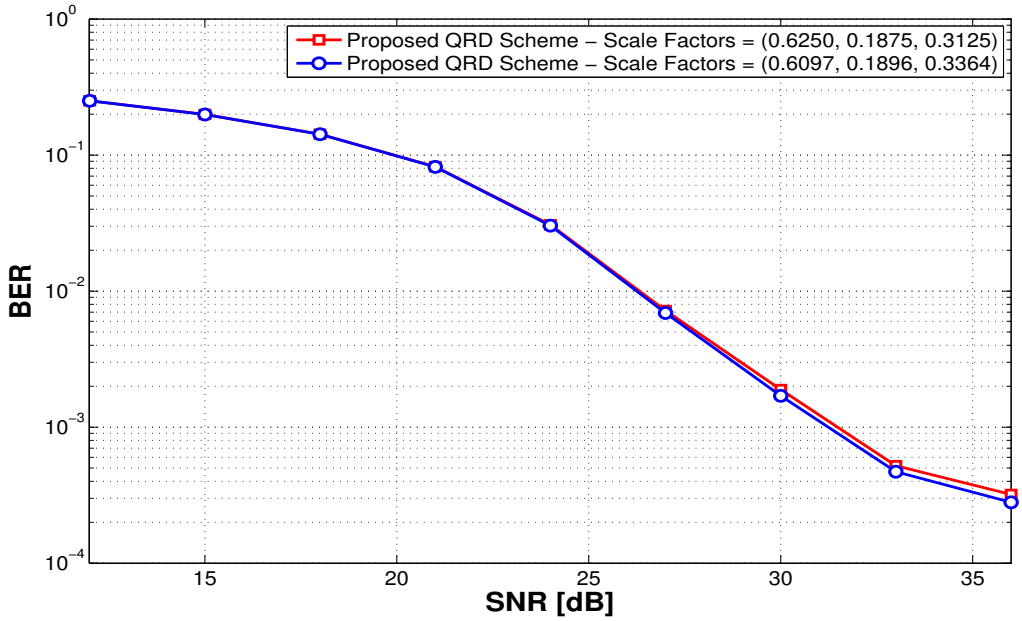

Figure 5.11: BER Performance of proposed QRD Scheme with different CORDIC Processing Gain Scale Factors

models for the proposed QRD were simulated with different number of CORDIC iterations. Fig. 5.12 shows the BER curves attained using these MATLAB simulations. From these BER curves, it can be noticed that QRD using 6 CORDIC iterations leads to a significant BER performance degradation, compared to QRD with 8 CORDIC iterations. On the other hand, QRD using 10 CORDIC iterations yields a BER performance improvement of approximately 0.28 dB at BER $= 10^{-3}$. However, from an implementation perspective, use of 10 CORDIC iterations leads to a much higher computational complexity, which results in either large QRD processing latency or large hardware and power requirements. This justifies our choice of using 8 iterations for 2D, Householder 3D and 4D/2D configurable CORDIC processors.

Fig. 5.13 shows the BER curves obtained by simulating the combination of QR Decomposition and K-Best MIMO Detector for different QRD schemes. The QR Decomposition MATLAB models use 8 CORDIC iterations and the scale factors of 0.6250, 0.1875 and 0.3125 for 2D, Householder 3D and 4D CORDIC processors, respectively. From Fig. 5.13, it can be noticed that the BER performance for the proposed QR Decomposition scheme is identical to that of the QRD scheme using the conventional sequence of Givens rotations, for both floating-point and fixed-point models. This can be justified by noticing that the average absolute difference between the $\mathbf{R}$ matrix and the $\mathbf{z}$ vectors produced by both of these schemes is on the order of 10^{-5} for floating-point models and on the order of 10^{-3}

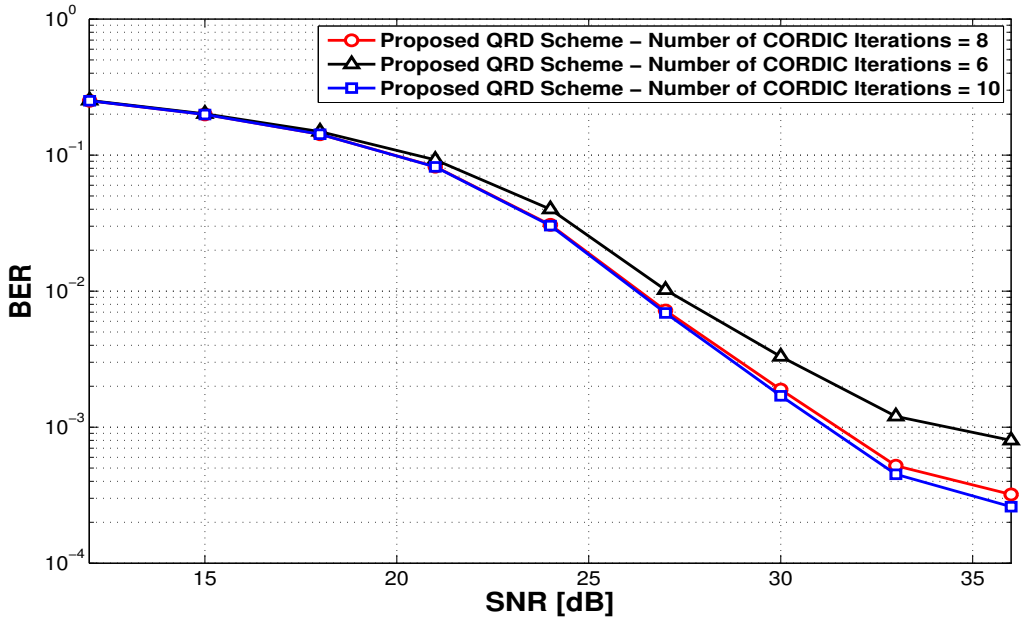

Figure 5.12: BER Performance of proposed QRD Scheme with different Number of CORDIC Algorithm Iterations

for fixed-point models. Note that these floating-point and fixed-point MATLAB models for QR Decomposition use the CORDIC algorithms for performing Givens rotations. Fig. 5.13 also shows the BER curve for QR Decomposition using ideal Givens rotations, implemented in floating-point arithmetic (as opposed to implementing them using the CORDIC algorithm). It can be noticed that the BER performance for QRD using ideal Givens rotations is marginally better compared to that when Givens rotations are implemented using the CORDIC algorithm. This can be explained by the fact that the CORDIC algorithm just approximates actual vector rotations, with the accuracy dependent on the number of CORDIC algorithm iterations used and the compensation scale factors used.

5.8 Test Results and Design Comparison

The proposed QR Decomposition core was fabricated in a 0.13 μm IBM 1P8M CMOS process and was tested using an Agilent(Verigy) 93000 SoC high-speed digital tester and a Temptronic TP04300 thermal forcing unit. The die micrograph for the QRD chip is shown in Fig. 5.14. The test setup consisting of the 93K SoC tester, Temptronic TP04300 thermal forcing unit, load board and the DUT is shown in Fig. 5.15. The nominal core supply voltage is 1.2 V, whereas the I/O voltage is 2.5 V. The functionality of the QRD

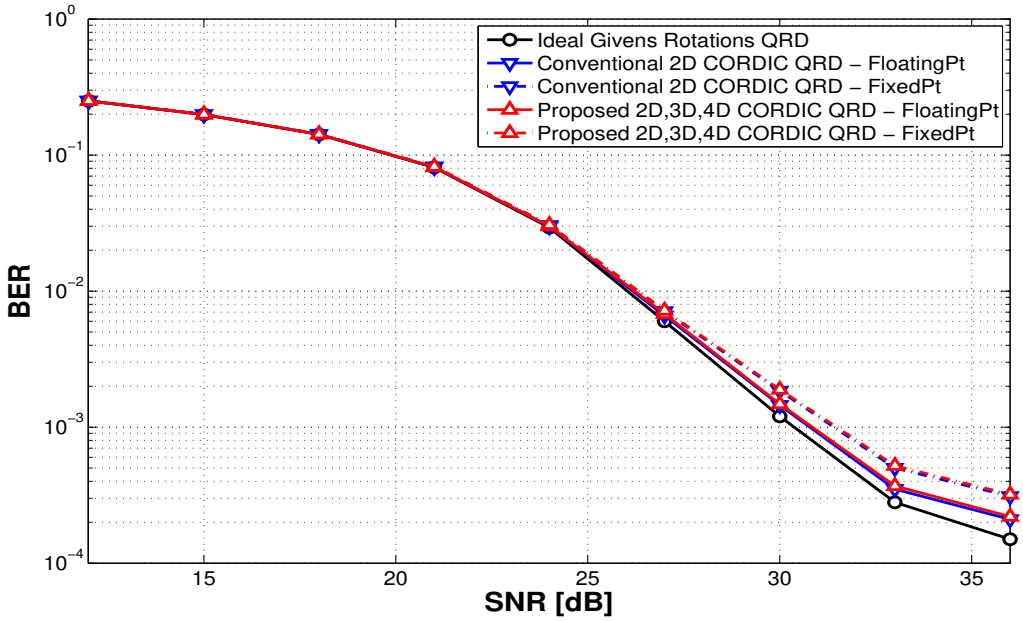

Figure 5.13: BER Performance of different QRD Schemes for 4×4 matrix decomposition - combined with 64-QAM K-Best MIMO Detector with K=10

core was verified by generating and passing channel matrices and received symbol vectors at different SNR values to the chip through the tester and comparing the QRD outputs with the expected values from the bit-true simulations both from MATLAB and Verilog HDL simulations. The BER performance of the QRD core was measured as follows:

1. Complex-valued random Gaussian channel characteristic matrix, updated every four channel uses, was generated and was used to transmit the symbol vectors.

2. For a given SNR value, additive white Gaussian noise with the desired variance was generated and was used along with the channel matrix to derive the received symbol vectors.

3. A test vector, including the input channel matrix and received symbols, as well as all the required control and enable signals, was generated using MATLAB.

4. This generated test vector was then converted to a VCD file using ModelSim, a Verilog HDL simulator.

5. The `V93K_TestGenerator` tool [48] was then used to convert the test vector VCD file to timing files (".tim"), configuration files (".pin"), and binary test vector files (".binl") required for testing.

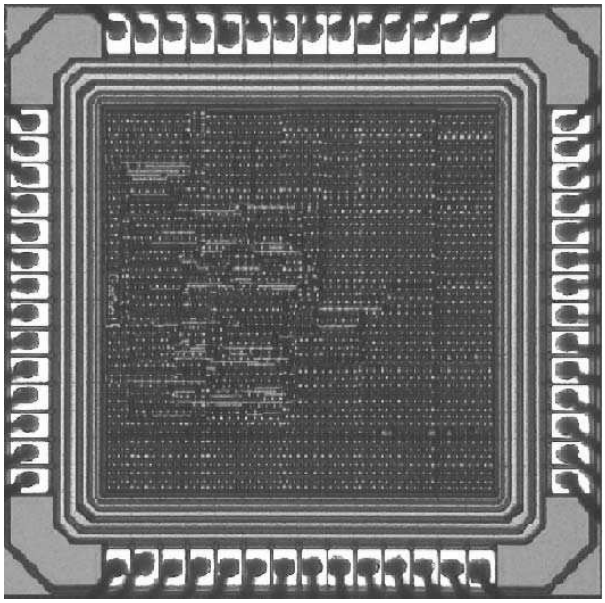

Figure 5.14: Die Micrograph for the QRD Chip.

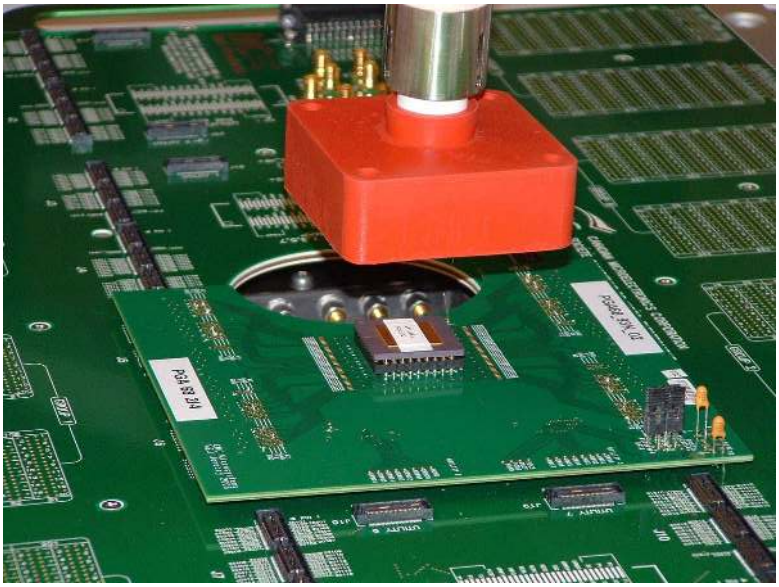

Figure 5.15: Test setup using Verigy 93K tester, Temptronic TP04300 thermal forcing unit head, and the DUT.

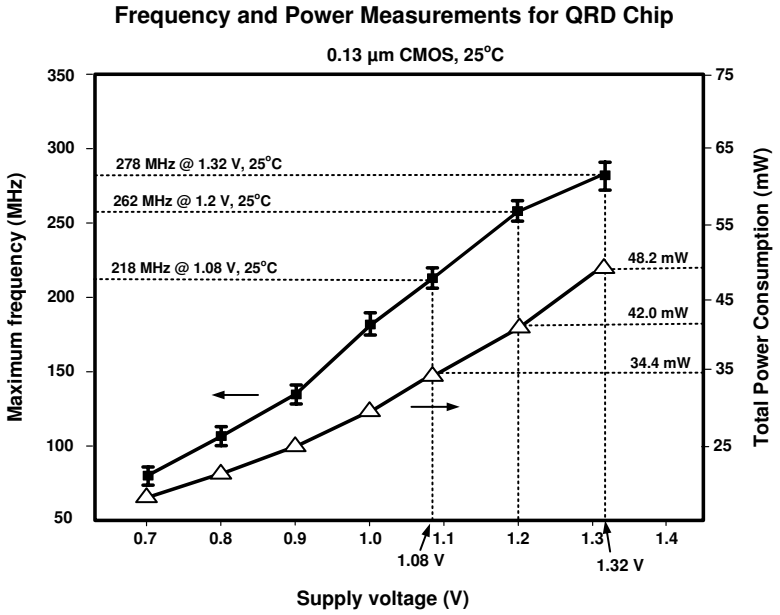

Figure 5.16: Measured Maximum Operating Frequency and Power Dissipation vs. Supply Voltage @ 25C.

6. The core supply voltage along with the I/O supply voltage are set appropriately.

7. These files were loaded onto the V93K SoC tester and were used to supply input test vectors to the QRD chip.

8. An at-speed test was run on the QRD chip and the outputs are compared against the desired bit stream generated by the MATLAB simulation.

Fig. 5.16 shows a Shmoo plot depicting the maximum operating frequency and the total power dissipation of the design versus the supply voltage at 25°C. A total of five chips were tested, where the average and the max/min values of the achieved frequency have been shown in Fig. 5.16. The detailed measurement results are presented in Appendix D in Table D.1 to Table D.15. At 25°C and 1.32V supply voltage, the QRD design operates at a clock rate up to 278 MHz and consumes 48.2 mW of power. The temperature was forced to be at 25°C using the Temptronic TP04300 thermal forcing unit. Also, using this Temptronic TP04300 thermal forcing unit, test results at 0°C and 85°C yield clock rates of 292 MHz and 254 MHz, while dissipating 51.5 mW and 43.7 mW, respectively, at 1.32V supply. The complete measurement results at these temperatures have been presented in Appendix D.

Table 5.4: Chip Characteristics and Comparison to Previous QR Decomposition Implementations.

Reference	[35]-2008	[34]-2007	[49]-2009	[37]-2007	This work
Process	0.13 μm	0.18 μm	0.18 μm	0.25 μm	0.13 μm
QRD Algorithm Used	MGS	MGS	MGS	Givens Rot	Hybrid
Input Matrix Size	4×4 Complex	4×4 Real	4×4 Real	4×4 Complex	4×4 Complex
QRD Processing Mode	Real	Real	Real	Complex	Real
QRD Processing Latency	139 cycles	67 cycles	44 cycles	67 cycles	40 cycles
Max Clock Frequency	269 MHz	277 MHz	270 MHz	125 MHz	278 MHz
QRD Processing Latency [ns]	516 ns	241 ns	162 ns	536 ns	144 ns
Core Area	23.2 KG	72 KG	51 KG	54 KG	36 KG
QRD Processing Efficiency (1/ns · KG) ($\times 10^3$)	5.346	0.923	1.936	2.212	12.352
Tested Chip	No	No	No	No	Yes
Power Consumption	N/A	N/A	N/A	N/A	48.2mW @ 1.32V

Table 5.4 shows the measured results for the designed QRD chip and compares it to other published state-of-the-art QR Decomposition implementations for decomposing 4×4 matrices. For the MMSE QR Decomposition design presented in [37], new **Q** and **R** matrices are produced every 67 cycles, running at 125 MHz, and the total core area required is 54 KG. The authors of [34] use log-domain computations to simplify implementation of multiplication, division and square-root operations in the Modified Gram-Schmidt (MGS) algorithm. However, this scheme requires considerable storage space to hold the look-up tables, and hence it requires large core area, as shown in Table 5.4. Note that the core area and processing latency numbers here are given for the complete matrix inversion operation, which requires an additional matrix multiplication stage after QR Decomposition. On the other hand, the authors of [35] present a low complexity approximation of the inverse square-root function to simplify the implementation of the division by norm operations in the MGS algorithm. This leads to a considerably lower gate count of 23.2 KG, however, the

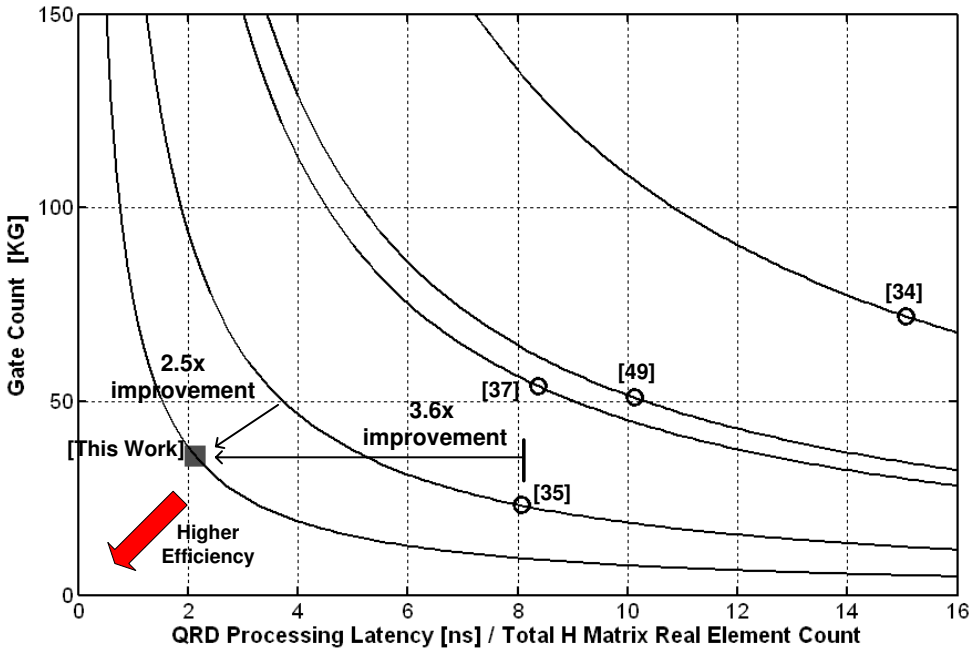

Figure 5.17: Comparison of QR Processing Efficiency between this work and previous QRD implementations.

QR Decomposition processing latency for this architecture is 139 cycles. In comparison, the novel QR Decomposition scheme and architecture proposed in this thesis outputs a new 4×4 complex $\mathbf{R}$ matrix and four 4×1 complex $\mathbf{z}$ vectors every 40 cycles, at a clock frequency of 278MHz, and requires a gate count of 36 KG. Thus, this architecture achieves the lowest QRD processing latency, while still achieving the second lowest core area.

As can be noticed from Table 5.4, some of the reference QRD chips have been designed for processing matrices with dimensions other than 4×4 complex. Hence, to allow fair comparison, we introduce a new figure of merit, `QR Processing Efficiency`, as follows:

$$QR\ Processing\ Efficiency \triangleq \frac{Total\ H\ Matrix\ Real\ Element\ Count}{Gate\ Count\ \times\ Processing\ Latency} \tag{5.12}$$

Note that for complex matrices, the `Total H Matrix Real Element Count` in equation 5.12 is attained by first using Real Value Decomposition (RVD) to convert the complex matrix to its real counterpart, and then by counting the total number of elements in the real-valued matrix. Fig. 5.17 shows the `QR Processing Efficiency` comparison between

the the reference QRD chips and our proposed design. Each hyperbola in Fig. 5.17 represents a constant value of `QR Processing Efficiency` (computed by taking the reciprocal of the product of the two axes) for each QRD design. Note that since `QR Processing Efficiency` is the reciprocal of the product of the two axes, the distance of the hyperbola to the origin is inversely proportional to the `QR Processing Efficiency` metric. In other words, hyperbolas that are relatively closer to the origin represent larger value of `QR Processing Efficiency`, and hence a better QRD design. Since the hyperbola for the presented QRD design is closest to the origin, the presented design attains the highest `QR Processing Efficiency`. From Fig. 5.17 and Table 5.4, the presented QRD core offers a 3.6× reduction in processing latency (for 4×4 complex case) and a 2.5× increase in QR Processing Efficiency, compared to the best reported design [35].

5.9 Summary

In order to fulfill the aggressive requirements of new 4G wireless standards, QR Decomposition implementations are required that decompose large complex channel matrices with minimum possible processing latency, silicon area and power consumption requirements. However, for decomposition of large channel matrices, the state-of-the-art QRD implementations cause high computational complexity and throughput bottlenecks, which leads to either large QRD Processing Latency or to large area and power requirements.

This chapter proposed a *hybrid* QR Decomposition scheme that reduces the number of computations required and increases their execution parallelism by using a unique combination of Multi-dimensional Givens rotations, Householder transformations and Conventional 2D Givens rotations. The computational complexity is further reduced by using the CORDIC algorithm to implement these multi-dimensional vector rotations. A semi-pipelined semi-iterative architecture is presented for the QRD core, that uses innovative design ideas to develop 2D, Householder 3D and 4D/2D Configurable CORDIC Processors, such that they can perform the maximum possible number of Vectoring and Rotation operations within the given number of cycles, while minimizing gate count and maximizing resource utilization. The test results for the QRD chip, fabricated in 0.13μm 1P8M CMOS technology, demonstrate that the QRD chip attains the lowest reported processing latency of 40 clock cycles (144 ns) at 278 MHz for 4×4 complex matrices at room temperature. It also outperforms all of the previously published QRD designs by offering the highest QR Processing Efficiency, while consuming only 0.3 mm^2 silicon area and 48.2 mW.

6 Conclusions and Future Work

6.1 Conclusions

Developing a high-throughput low-complexity Soft-Output MIMO detector for high-order constellation sizes and large antenna configurations has been a significant challenge in the literature. To address this issue, this thesis proposed a novel Soft-Output K-Best MIMO detection scheme that improved BER performance by utilizing information contained in the paths discarded at the intermediate tree levels by the K-Best algorithm. The proposed algorithm reduced the number of computations required significantly by detecting and processing only useful discarded and K-Best paths and by using approximations to actual calculations, where appropriate. For the case of 4×4 64-QAM MIMO detection with K=10, the proposed scheme reduced the number of computations required by a factor of 5 for LLR computation purposes, compared to the MKSE Soft-Output detection scheme (that uses all discarded and K-Best paths). Furthermore, for the same case, the proposed scheme leads to a BER performance improvement by 1.7 dB (at BER = 10^{-3}) compared to the conventional Soft K-Best detection scheme.

An area and power efficient high-throughput VLSI implementation of a 4×4 64-QAM K-Best MIMO detector was realized using the presented Soft-Output MIMO detection scheme. The proposed Soft K-Best detector used a deeply pipelined architecture to maximize throughput and various strategies to minimize hardware and power requirements. In a 0.13 μm CMOS technology node, the proposed design attained a detection throughput of up to 655 Mbps, which is 5.8× higher compared to the highest Soft-output detection throughput published previously. Synthesis results in 65nm CMOS demonstrated that the proposed MIMO detector attains a peak data throughput of 2 Gbps, which makes it the only design published to-date that fulfills the aggressive data rate requirements of the next-generation IEEE 802.16m and LTE-Advanced 4G wireless standards.

The thesis also presented an efficient low-complexity algorithm for channel matrix QR

Decomposition, an essential channel pre-processing task for MIMO detection. The proposed *hybrid* QRD algorithm reduces the number of computations required and increases their execution parallelism by using a unique combination of multi-dimensional and two-dimensional vector rotations. Further reduction in computational complexity is attained by implementing the vector rotations using low-complexity Multi-dimensional and Householder CORDIC algorithms. Moreover, the thesis proposed a semi-pipelined semi-iterative VLSI implementation for the QRD core, that used novel architectures for 2D, Householder 3D and 4D/2D Configurable CORDIC processors, to maximize throughput and resource utilization, while minimizing gate count. The QRD chip was fabricated in 0.13μm 1P8M CMOS technology and test results demonstrate that it attains the lowest reported processing latency of 40 clock cycles (144 ns) at 278 MHz for QRD of 4×4 complex matrices and offers the highest QR Processing Efficiency compared to all of the previously reported QRD chips.

6.2 Future Directions

Soft K-Best MIMO Detector - Complex Mode

The complex mode for MIMO detection offers the advantage of processing only half as many tree levels, compared to the real mode. However, the number of possible children to be expanded per parent is twice as large and the sorting per level is more complicated. Moreover, all the operations including the Euclidean distance calculation in all levels are in the complex domain. However, depending on the objectives and the specifications of the targeted MIMO detector core, sometimes it might be desirable implement the Soft K-Best MIMO detector in the complex mode. Hence, extension of the proposed low-complexity Soft-output K-Best scheme to the complex domain and an efficient hardware implementation of MIMO detector in the complex mode is an interesting topic worth investigating further.

Soft K-Best MIMO Detector - Extension to 256-QAM

Since 256-QAM is considered as a possible optional modulation for the emerging 4G standards, the ASIC implementation of a Soft-output K-Best MIMO detector in Chapter 4 for

the 4×4 256-QAM case is an interesting avenue for further study. There is no hardware implementation in the literature for the 256-QAM case, because of the significant increase in the complexity of a Soft K-Best detector for 256-QAM constellation. However, the On-Demand nature of K-Best path selection, as well as the improvement ideas to reduce computational complexity of the LLR Computation operation makes the proposed scheme a low-complexity Soft K-Best MIMO detection scheme, and hence makes the realization of a 256-QAM system feasible.

Soft K-Best MIMO Detector and QRD - Extension to 8×8 MIMO Systems

Emerging 4G standards also require MIMO systems with antenna configurations up to 8×8. The generalized version of the proposed QR Decomposition scheme (presented in Section 5.4.2) can be customized to process 8×8 complex channel matrix. This low-complexity customized QRD scheme for 8×8 MIMO can then be used, along with the low-complexity, high-throughput architectures of 2D, Householder 3D and 4D/2D Configurable CORDIC processors to develop QRD core architecture with low QRD processing latency, low area and power consumption requirements. Furthermore, the proposed Soft-output K-Best MIMO detection scheme presented in Chapter 3 is scalable to larger antenna configurations and larger constellation orders. Hence, this Soft-output K-Best detection scheme can be directly used to develop a VLSI implementation of a Soft K-Best detector for 8×8 64-QAM, as well as 8×8 256-QAM MIMO systems.

A The On-Demand Hard K-Best Detection Scheme - First/Next Child Calculation [1]

The On-Demand Hard K-Best scheme [1] was described briefly in Section 3.3.1. As mentioned, this scheme requires the computation of First Child (FC) and Next Child (NC) for parent nodes, in order to generate the K-Best path list at each tree level. Based on the system model in equation 2.13, the first child ($s_l^{[1]}$) of a node in $\mathcal{K}_{l-1}$ is the one minimizing $e_l\big(\mathbf{s}^{(l)}\big)$, i.e.,

$$s_l^{[1]} = \arg\min_{s_l \in \Omega} \left| e_l\big(\mathbf{s}^{(l)}\big) \right|^2 = \arg\min_{s_l \in \Omega} \left| L_l\big(\mathbf{s}^{(l)}\big) - r_{ll} s_l \right|^2. \tag{A.1}$$

This is because $T_{l+1}\big(\mathbf{s}^{(l+1)}\big)$ is in common between all children of a parent.

Therefore, $s_l^{[1]}$ can be found by rounding $s_l^{[0]} = L_l\big(\mathbf{s}^{(l)}\big)/r_{ll}$ to the nearest odd number in Ω (represented by $\lceil \cdot \rfloor$ in this dissertation). Note that rounding $s_l^{[0]}$ to the nearest odd number in Ω is equivalent to calculating $2\lfloor \frac{s_l^{[0]}+1}{2} + 0.5 \rfloor - 1$, where $\lfloor \cdot \rfloor$ represents the truncation operation. In order to find the next children (NC), the Schnorr-Euchner technique, [50], is employed, which implies a zig-zag movement around $s_l^{[0]}$ to select the

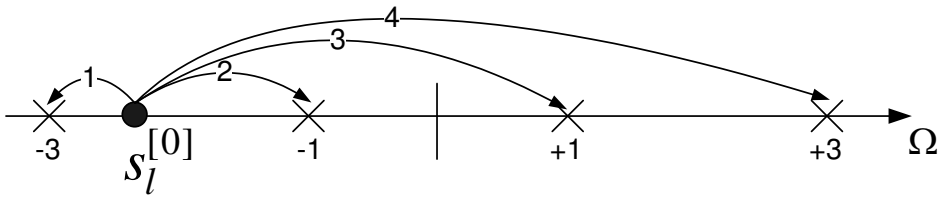

Figure A.1: The order of the SE row-enumeration for four consecutive enumerations in 16-QAM.

Table A.1: First/Next Child Selection Procedure for Node j [1].

1) First Child

1.1) $s_l^{[0]} = \ell_l$
1.2) $s_l^{[1]} \leftarrow \lceil s_l^{[0]} \rfloor$, $n_l^j \leftarrow 2$.

2) Next (k-th) Child

2.1) $\mathsf{SignBit} = \begin{cases} \mathsf{Sign}\left(s_l^{[k-1]} - s_l^{[k-2]}\right) & \texttt{if } s_l^{[k-1]} \neq \pm(\sqrt{M}-1) \\ -1 & \texttt{if } s_l^{[k-1]} = (\sqrt{M}-1) \\ +1 & \texttt{if } s_l^{[k-1]} = -(\sqrt{M}-1) \end{cases}$

2.2) $s_l^{[k]} \leftarrow s_l^{[k-1]} + n_l^j \times \mathsf{SignBit}$.

2.3) $n_l^j = \begin{cases} 2 & \texttt{if } s_l^{[k-1]} = \pm(\sqrt{M}-1) \\ n_l^j + 2 & \texttt{otherwise} \end{cases}$

consecutive elements in Ω. Fig. A.1 shows such an enumeration for $\sqrt{Q} = 4$. As shown, the SE enumeration finds the closest points in a real domain one-by-one by changing the search direction. In this example -3 is the closest point to $s_l^{[0]}$ (denoted by arrow 1), the next point is -1 (denoted by arrow 2) and so on. Note that the direction is negative and then positive, and normally alternates between the two. However, if the SE enumeration reaches the upper/lower bound of Ω (+3/-3 in this example), the direction of the search remains fixed, as is the case for 3-rd and 4-th children in Fig. A.1. Based on this strategy, the procedure of selecting the first/next child of node j in level l is described in Table A.1, where n_l^j denotes the number of moves, and SignBit represents the direction. In fact, SignBit alternates between positive and negative unless it reaches $\pm(\sqrt{M}-1)$. The number of moves also increases by 2 every time and is reset to 2 if boundaries of Ω are reached.

The detailed pseudo-code shown in Table A.2 describes the On-Demand Hard K-Best scheme. In the initialization step, the K best nodes of the first level are selected creating $\mathcal{K}_1$ (`Step I.1`, and `I.2`). For each of the elements in $\mathcal{K}_1$, the first child and its corresponding weight are found (`Step II.2`, and `II.3`). The child with the lowest weight is selected as one of the K best nodes (`Step II.4.1` and `II.4.2`) and is replaced by its next best sibling (`Step II.4.3 to II.4.5`). This process is repeated K times to find all the K-Best nodes

Table A.2: The Proposed Implementation for the On-Demand Hard-Output K-Best Algorithm [1].

I. INITIALIZATION

1) Find the K-Best children of level 1 of tree C_1.

2) Set $\mathcal{K}_1 = C_1$.

II. EXPANSION & SORT

For $l = 2 : 1 : 2N_T$

1) $\mathcal{K}_l = \emptyset$.

2) Find $\mathcal{C}_l$, the set of the first child of each node in $\mathcal{K}_{l-1}$.

3) Calculate $\mathcal{D}_l$, the weights of the elements in $\mathcal{C}_l$.

4) For $k = 1 : K$

4.1) $\bar{k}\bar{m} = \arg\min_{k,m} \left\{ d^l_{k,m} \in \mathcal{D}_l \right\}$

4.2) $\mathcal{K}_l \leftarrow \mathcal{K}_l + \left\{ c^l_{\bar{k}\bar{m}} \right\}$

4.3) $\mathcal{C}_l \leftarrow \mathcal{C}_l - \left\{ c^l_{\bar{k}\bar{m}} \right\}$, and $\mathcal{D}_l \leftarrow \mathcal{D}_l - \left\{ d^l_{\bar{k}\bar{m}} \right\}$.

4.4) Find the next best child of $\bar{k}$-th parent, $c^l_{\bar{k}\hat{m}}$.

4.5) $\mathcal{C}_l \leftarrow \mathcal{C}_l + \left\{ c^l_{\bar{k}\hat{m}} \right\}$, and $\mathcal{D}_l \leftarrow \mathcal{D}_l + \left\{ d^l_{\bar{k}\hat{m}} \right\}$.

End

End

III. DETECTION

1) $\bar{k}\bar{m} = \arg\min_{k,m} \left\{ d^1_{k,m} \in D_1 \right\}$

2) Announce $c^1_{\bar{k}\bar{m}}$ with all of its parents up to the first level of tree as the hard decision output $\widehat{\mathbf{s}}$.

---

of level l and is performed for all the levels down to level $2N_T$. Among the K-Best nodes of level one, the node that has the lowest PED $\left(\bar{k}\bar{m} = \arg\min_{k,m}\left\{d^1_{k,m} \in D_1\right\}\right)$ with all of its ancestors up to the first level are announced to be the output of the hard-decision detection problem (`Step III.1` and `III.2`).

B Soft K-Best Detector - Detailed VLSI Architecture for Hard Detection Specific blocks [1]

This section provides details about the architecture and functionality of the blocks used in the proposed Soft K-Best MIMO detector, that are specific to Hard K-Best detection. Note that the design ideas for these blocks were used from the Hard K-Best MIMO detector presented in [1]. Detailed description of the overall Soft K-Best detector architecture, some common sub-blocks used throughout the architecture, details about the major functional blocks used for Soft K-Best detection was provided in Section 4.4.

B.1 Level I

The input to `Level I` is r_{88} and $\bar{z}_8$ and its output is the PEDs of all the elements in Ω, which are the nodes in the 8-th level of the tree. The detail of the architecture is shown in Fig. B.1. It employs a 13-bit×13-bit multiplier, eleven adders and the absolute value block. Note that the absolute value block, representing the ℓ^1-norm, can be replaced by a squaring operation block, ℓ^2-norm, which can be easily implemented using a carry-save-adder technique [11]. However, simulation results show that the difference in the BER performance is negligible [28]. Fig. B.2 confirms this and shows the performance result for a 4×4, 64-QAM MIMO system with $K = 10$ for ℓ^1-norm and ℓ^2-norm cases. It shows that the performance result for both cases is almost the same but due to the low-complexity nature of the ℓ^1-norm, it is the preferred approach for the implementation.

Since `Level I` is on the feed-forward path of the architecture, a fine-grained pipelining technique can be employed inside the block in order to increase the system throughput. Two-stage pipelining is employed in `Level I`, which is shown by two and five positive-edge-triggered registers/flip-flops added in stage one and two, respectively (Fig. B.1). In fact, by using the registers the block is broken down to two consecutive stages, which avoids a long critical path and implies that the critical path of the architecture contains

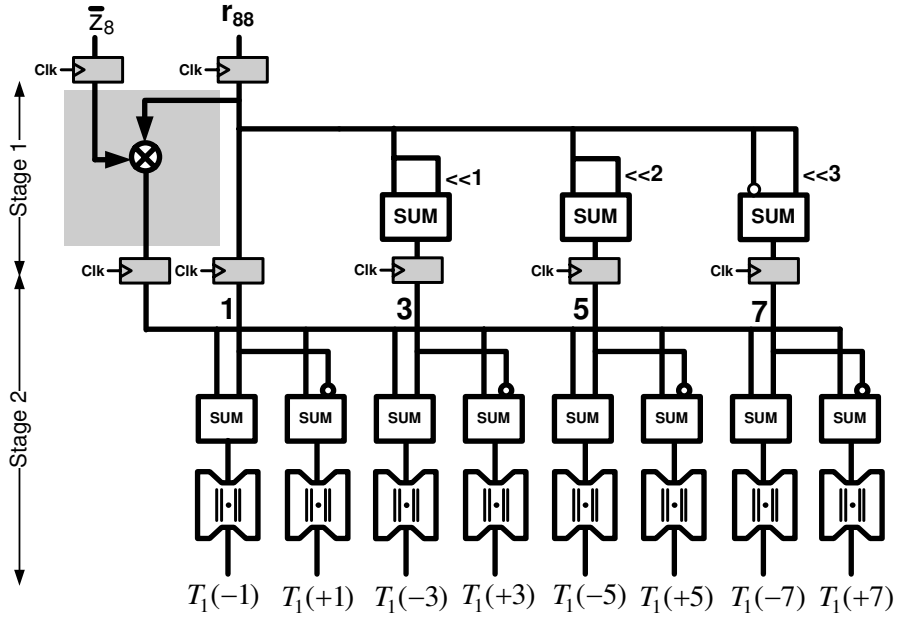

Figure B.1: The architecture for `Level I` with the critical path highlighted [1].

only one multiplication. It is assumed that all the registers used in this thesis are triggered by the positive-edge of the clock.

B.2 Level II

The input to `Level II` is the PED values of the 8-th level and its output is the PED values of the first children in the 7-th level of the tree. In fact, in the `Level II` block, the first children of the eight nodes in the 8-th level are determined. Note that due to the structure of the $\mathbf{R}$ matrix in (4.2), the first children in the 7-th level of the tree are all the same and independent of their parents in level 8 (because $r_{78} = 0$). This child is determined and is used to calculate the updated PED values of the nodes in the 7-th level. Since $r_{77} = -r_{88}$, no extra input is required for the calculations in `Level II`. The equation of the 7-th level can be written as

$$r_{77}\bar{z}_7 = r_{77}s_7 + v_7 \Rightarrow \bar{z}_7 = s_7 + \frac{v_7}{r_{77}}. \tag{B.1}$$

This implies that in order to find the first child in the 7-th level, $\bar{z}_7$ is applied to the input of the `Mapper/Limiter` block whose output is the first child. The architecture of the `Level II` block is shown in Fig. B.3. Once the first child was determined it is multiplied by r_{77}

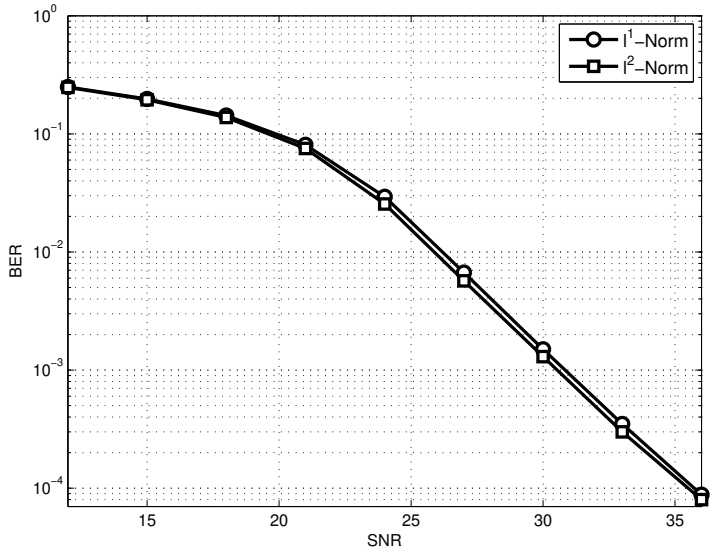

Figure B.2: The performance of a 4×4 64-QAM MIMO system with $K = 10$ for ℓ^1-norm and ℓ^2-norm case [1].

using the `MU` block. The input normalized $\bar{z}_7$ value is also multiplied by r_{77} after which the Euclidean distance between the first child and the received vector (i.e., $|r_{77}\bar{z}_7 - r_{77}s_7|$) is calculated and the result is added to the PED values of the 8-th level PEDs to derive the eight updated PEDs of the 7-th level. A fine-grained pipelining technique has also been employed in this block to break it into four stages in order to limit the length of the critical path. Note that in order to guarantee the correct functionality of the architecture after pipelining, three stages of registers are required to be inserted on all input PEDs coming from the `Level I` block (i.e., $T_1(-1), \cdots, T_1(+7)$).

B.3 Sorter Block

The input to the `Sorter` block is the set of eight PED values of the 7-th level FCs and the main task of this block is to generate the sorted list of these PED values. The architecture of the `Sorter` is shown in Fig. B.4. The eight inputs are denoted by D_0-D_7. The `Ctrl` signal is used to load the data in one clock cycle. Using this architecture, it

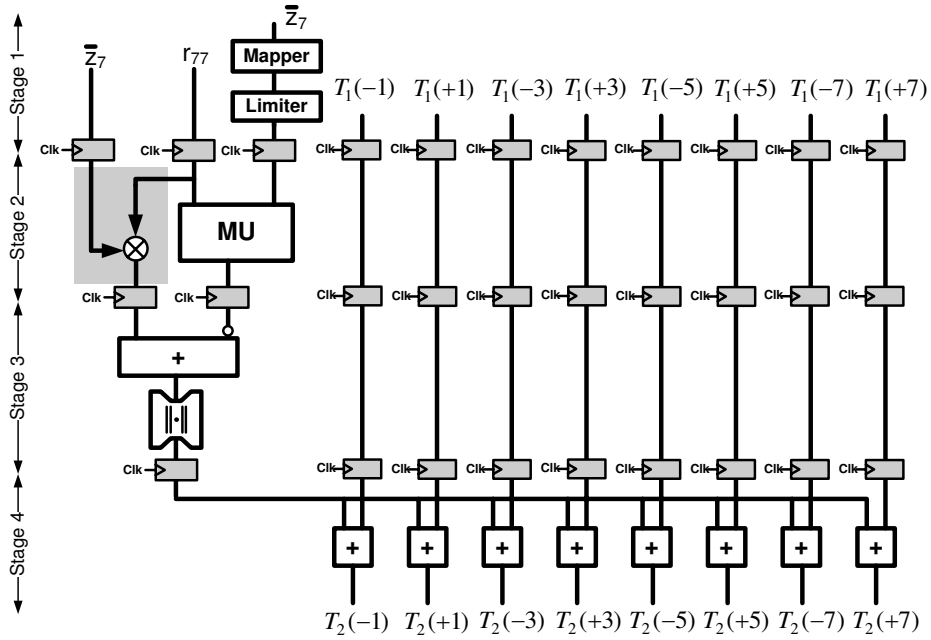

Figure B.3: The architecture for `Level II` with the critical path highlighted [1].

takes four clock cycles to sort all the eight PED values, which are stored in the positive-edge-triggered registers after the comparators[1]. This architecture can be used as a general sorter, which sorts K numbers in $K/2$ clock cycles. This efficient architecture performs the sorting operation twice as fast as a bubble sorter [22] because two consecutive minimizations/maximizations are implemented in one clock cycle between two registers. One such set of consecutive minimizations is highlighted in Fig. B.4, which is also the critical path of the `Sorter` block. Since the global critical path of the MIMO detector is larger than that of the `Sorter` block, the two consecutive minimizations do not limit the total throughput. Note that the factor N on the registers, shown in Fig. B.4, represents a register bank ($\texttt{RB}_i$) of length N bits, used to store the child list (path history) as well as the updated PED values[2].

B.4 PE I Block

`PE I` is a general block used for all the levels from level 7 to level 2. It receives the sorted list of the first children of each level and generates the K best candidates of that level. For instance the output of the `PE I` in level 7, called `NC-L7`, is the $K = 10$ consecutive best

[1]The comparators are realized using the blocks provided in the ARM standard cell library.
[2]The path list grows from one level to another so does the value of N. This means that the value of N is different for each `PE I/II` stage.

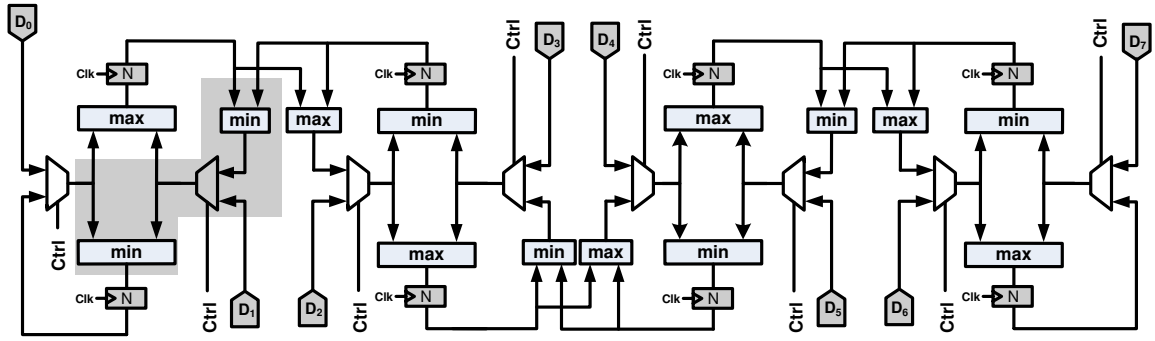

Figure B.4: The architecture for the `Sorter` block with the critical path highlighted [1].

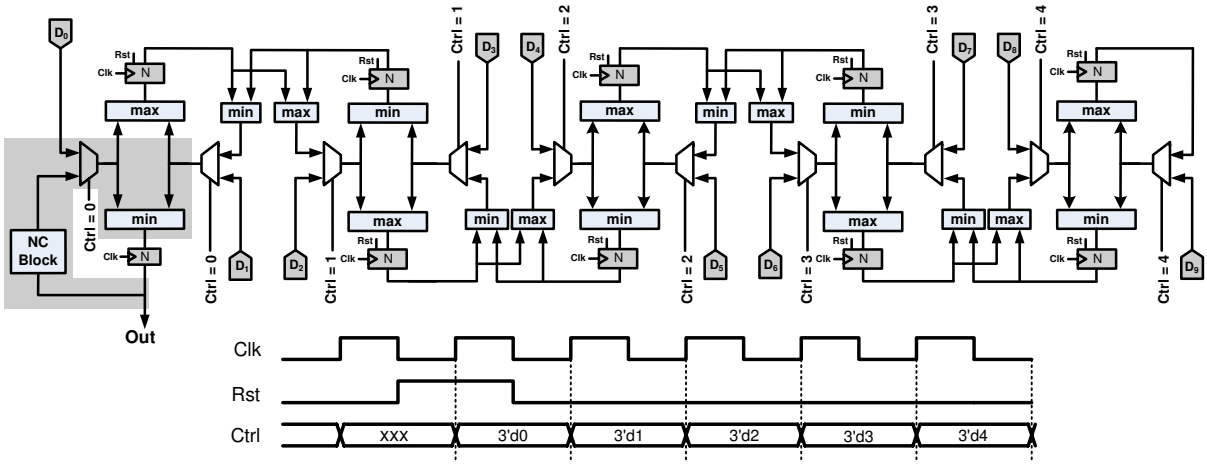

Figure B.5: The architecture for the `PE I` block with the critical path highlighted [1].

candidates with the lowest PED values in level 7, generated one-by-one in series at the output. In other words, `PE I` implements `Steps 4.1-4.5` of the Hard K-Best algorithm. The architecture of `PE I` is shown in Fig. B.5. It consists of a sorter, and a block called `NC-Block` on the feedback path. In fact, `PE I` receives the sorted list of the PEDs from the preceding stage. It finds the best one with the lowest PED and announces it as the next K-Best candidate at the output, and then calculates the next best sibling of the announced child through the `NC-Block` and feeds it back to the sorter to locate the correct location of the new sibling in the already sorted list in `PE I`. Finally it performs the comparison and announces the next K-Best candidate in the next clock cycle. The following points clarify the details of this architecture:

- The main task of the sorter in this block is to receive a sorted list and finds the right

position of a new entry in the sorted list, while announcing the entry with the lowest PED every clock cycle. In other words, as a new entry comes into the sorter, it walks through the sorter and finds its correct ordered position. This position is updated every clock cycle by the introduction of the new sibling from the feedback path.

- Before the sorted PED values of the preceding stage are loaded into the `PE I` block, there is a reset signal, `Rst` in Fig. B.5, that initializes all the register banks (except the one attached to the output) to the maximum possible number. This is necessary to avoid any interference from the previous values stored in them and makes them ready to process the new list. The `Rst` signal also initializes the control signal, `Ctrl` in Fig. B.5, to zero whose value increases every clock cycle. The `Ctrl` signal is used to load the sorted list from the preceding stage to the `PE I` block. Note that the data in the sorted list is loaded one pair at the time. For instance, when `Ctrl=0`, D_0 and D_1 are loaded and when `Ctrl=4`, D_8 and D_9 are loaded. The reason is to guarantee the proper functionality of the architecture when `PE I` and `PE II` are co-operating together. Additional details will be provided in Section B.6. A snapshot of the signal transitions and the relation between the `Clk`, `Rst`, and `Ctrl` signals are also shown by an example in Fig. B.5.

- The critical path of the `PE I` block is highlighted in Fig. B.5. It contains a MUX, a comparator and the `NC-Block`. The main task of the `NC-Block` is to determine the next best sibling of an already announced best child. It also finds the PED value of this sibling and sends the information to the sorter part of the `PE I` block. Since the `NC-Block` is on the feedback portion of the architecture, pipelining cannot help to increase the throughput of the architecture. In fact, this path is the critical path of the whole MIMO architecture. Therefore, an efficient architecture needs to be proposed for the `NC-Block` to ensure an overall high-throughput architecture.

B.5 NC-Block

The detail of the `NC-Block` architecture is shown in Fig. B.6. The main task of the `NC-Block` is to determine the next sibling of the currently announced best child using the SE enumeration technique. Thus, the `NC-Block` in the i-th level needs to calculate the number of jumps n_i , the direction of the next move, `SignBit`, and finally calculates the

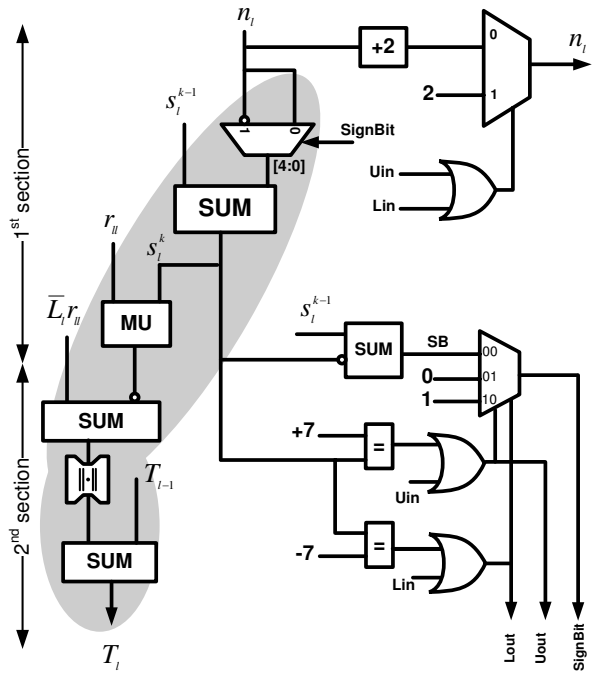

Figure B.6: The architecture for the `NC-Block` with the critical path highlighted [1].

PED value of the new sibling[3]. These three tasks are implemented by the architecture shown in Fig. B.6, where `SignBit` determines the direction of the SE enumeration for the next child and `Uout`(`Lout`) determines if the SE enumeration has reached the upper (lower) boundary of the Ω set. In fact, the PED value of the new sibling can be determined as follows:

$$T_i = T_{i+1} + |L_i - r_{ii}s_i| = T_{i+1} + r_{ii}|\bar{L}_i - s_i|, \tag{B.2}$$

where $(\mathbf{s}^{(l)})$ was omitted for brevity of discussion and $\bar{L}_i = \bar{z}_i - \sum_{j=i+1}^{2N_T} \bar{r}_{ij}s_j$. As mentioned, any effort to simplify this block and/or reduce its critical path, has a direct and significant effect on the total achievable data rate. In order to optimize its critical path, the following two efficient techniques were utilized in our VLSI architecture:

1. *Avoid multiplication*: Since the value of $\bar{L}_i$ depends only on the selected symbols up to level i and is independent of the current sibling (s_i), the values of $\bar{L}_i$ and $\bar{L}_i r_{ii}$ can be calculated using the `FC-Block` in the preceding block[4] and forwarded to the `NC-Block` as an input (see Fig. B.6). This is a preferred approach as the required multiplication to calculate $\bar{L}_i r_{ii}$ will be rescheduled and removed from the

[3]The signal `SB` in Fig. B.6 represents the sign bit of the result of the adder.

[4]For `PE I` of `NC-L7`, the preceding block is `Level II`. For all other `PE I` blocks, this block is the `PE II` block in the preceding stage. For instance for `PE I` in `NC-L3`, this is done in `PE II` in `FC-L3`.

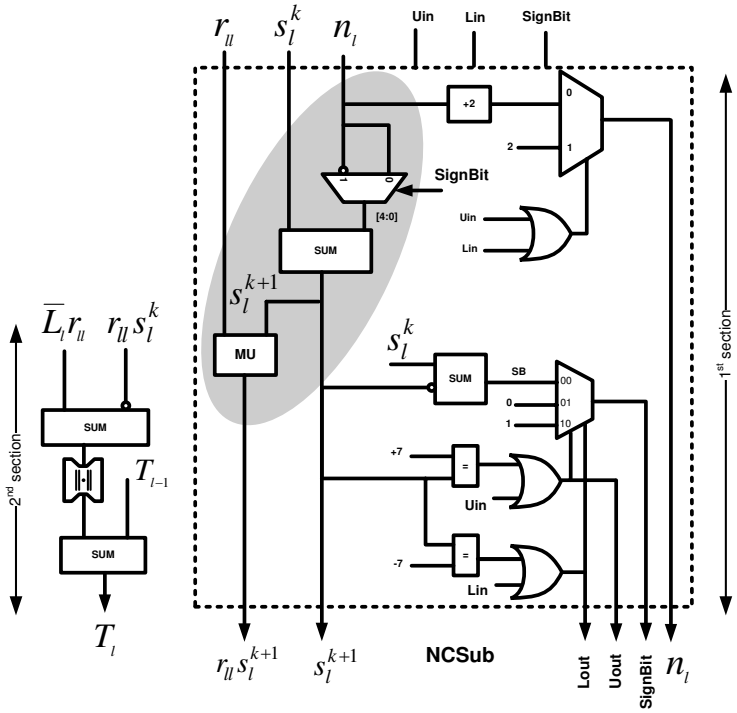

Figure B.7: The architecture for the `NC-Block` with improved critical path [1].

critical path and is shifted to a block that is pipelineable. Moreover, the second multiplication, i.e., $r_{ii}s_l$, is realized using the `MU` block.

2. *Broken critical path*: As can be seen from the `NC-Block` architecture (Fig. B.6), the critical path has three adders (one 4-bit and two 16-bit adders), as well as the `MU` block. The critical path associated with this architecture is 4.8ns in 0.13 μm CMOS technology using the ARM standard cell library. The first part of the critical path (specified by 1^{st} section in Fig. B.6) calculates the next sibling, which can be transferred to the `FC-Block` in the preceding block. This means that the `FC-Block` would calculate both the first and second best child of each parent and sends them to the `NC-Block`. The `NC-Block` calculates the PED value of the second best child while determining the third best child and so on. This implies that the `NC-Block` block always calculates one child ahead. Using this approach in our ASIC implementation yields a critical path of length 3.65ns, thus higher overall throughput. The use of this scheduling technique effectively breaks the critical path of the `NC-Block` down into two smaller parts (1^{st} and 2^{nd} section in Fig. B.6). This is shown in Fig. B.7 with the improved critical path. The first section of the `NC-Block` on the right hand

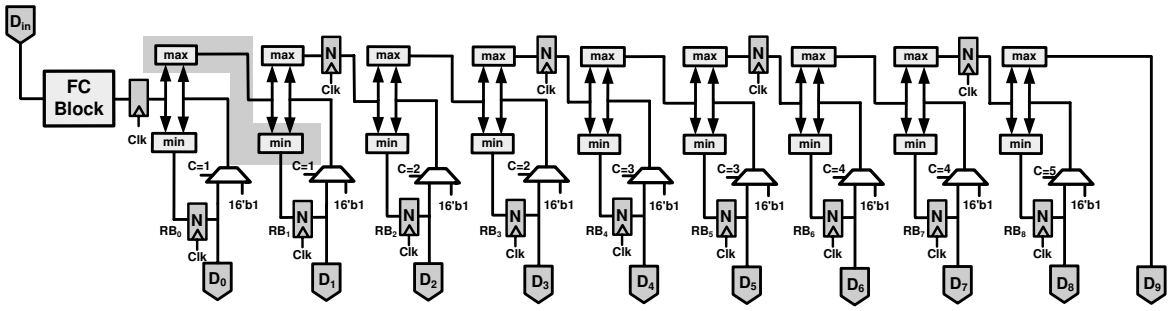

Figure B.8: The architecture for the PE II block with the critical path highlighted [1].

side of Fig. B.7 is denoted by NCSub, which is the block that will be added to the preceding FC-Block in order to calculate the second best child.

In brief, by forwarding all the processing required to start the second section in the NC-Block to the preceding FC-Block, the critical path is significantly improved.

B.6 PE II Block

The output of PE I is the serial list of K-Best candidates of the the current level, generated one-by-one at the output. As each of the K-Best candidates is generated, it is sent to the PE II block to calculate the first children of the next level and sort them as they arrive. The architecture of the PE II block is shown in Fig. B.8, where D_{in} is the input port and D_0-D_9 are the output ports. At the beginning, the first child of the K-Best candidate of the previous stage and its updated PED value are calculated by the FC-Block, and then using the sequential sorter, the calculated PED values are sorted as they arrive. Note that this process is performed on a cycle basis since the PE II block is connected to the output of the PE I block in a pipelined fashion. In the proposed architecture for PE II, the sorted PEDs are stored in register banks, depicted by N-bit registers in Fig. B.8 and denoted by $\texttt{RB}_0 - \texttt{RB}_8$. At every clock cycle, two register banks are updated at the same time. This is because of the fact that the registers on the upper part of the sorter are located in every other stage. This is required to guarantee the correct functionality of the PE I. The functionality of the sorter is such that the larger values are shifted to the right while the smaller values are shifted to the left. Once the last element (10-th element in 64-QAM) enters the sorter, it updates the first two register banks, thus the first two are guaranteed to have the two smallest PED values. Therefore, at the next clock cycle, they can be transferred to the following PE I block. After the second clock cycle, the next two register banks are updated and they are also ready to be transmitted. Therefore, the PED

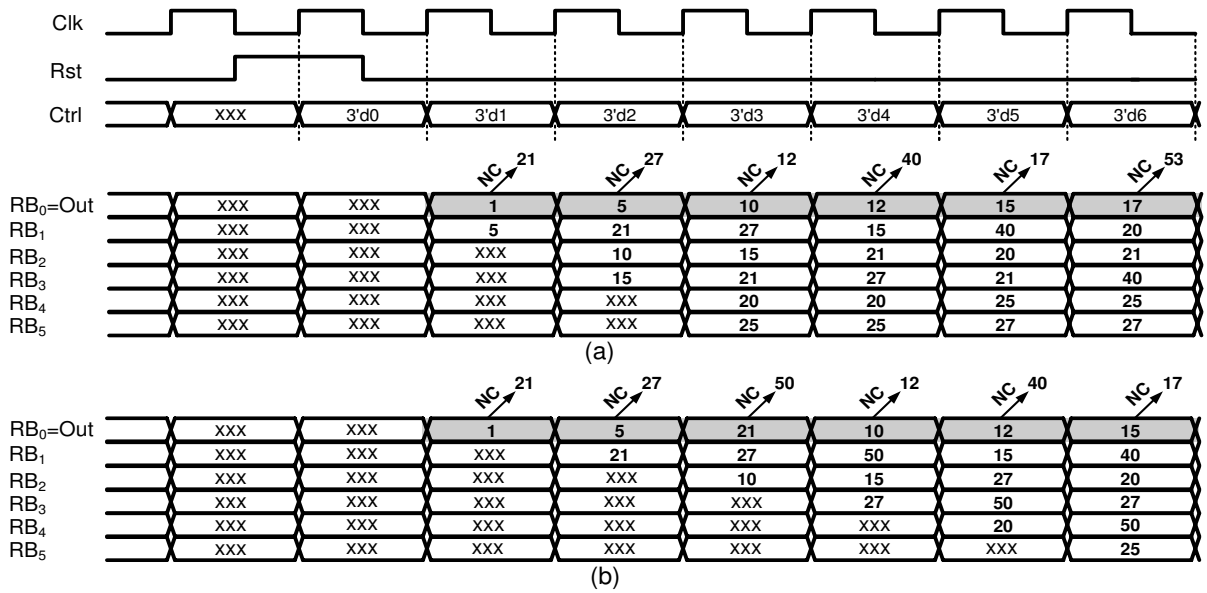

Figure B.9: The pairwise data transfer from PE II to PE I, (a) two entries at a time, (b) one entry at a time [1].

values are transferred to the next level on a pair-by-pair basis. This fact is shown in Fig. 4.1 with grey lines between the `PE II` block and the `PE I` block and the numbers on them represent the number of clock cycles after the arrival of the last K-Best candidate to the `PE II` in which they are transferred. This transfer is performed only once every K clock cycles.

The reason that the data is transferred one pair at a time can be understood in Fig. B.9. This example is for $K = 6$, and six register banks in `PE I`, which are represented by $\texttt{RB}_0$-$\texttt{RB}_5$. In fact, Fig. B.9 shows the internal register updates between different register banks in `PE I`. The first schedule (Fig. B.9.a) shows the transitions when the data is transferred in pairs, while the second schedule (Fig. B.9.b) shows the case where the data is pushed into `PE I` one entry at a time. The expected correct sorted list for this example at the output of the `PE II` block is $\{1, 5, 10, 12, 15, 17\}$. For example, in the first schedule, $\{1, 5\}$ are transferred to `PE I` in one clock cycle and in the next cycle, $\{10, 15\}$ are transferred. However, for the second schedule, each element in the FC list is fed one-by-one to the `PE I` block. Note that in the `PE I` block, each FC that is chosen as the best candidate is fed back to the `NC-Block` to calculate its next sibling. This is represented in Fig. B.9 by the arrows and the associated PED value of the next sibling next to it. For instance, the PED value of the next sibling of the candidate with PED=1 is 21. The first register bank $\texttt{RB}_0$ is connected to the output, which represents the list of the K-Best candidates of the current level. This output is highlighted by the grey line in the schedules (the first row of each

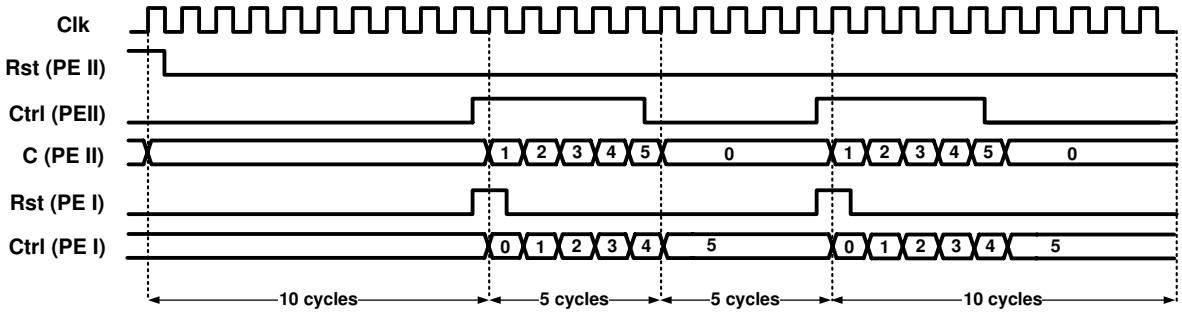

Figure B.10: The timing scheduling between a typical pair of `PE II` and `PE I` [1].

schedule). Note that the one-by-one strategy, i.e., Fig. B.9(b), results in the wrong result at the output (i.e., $\{1, 5, 21, 10, 12, 15\}$), while feeding the data one pair at a time (Fig. B.9(b)) results in the proper functionality (i.e., $\{1, 5, 10, 12, 15, 17\}$).

Note also that once the last element comes in and the first two register banks are sent to the next stage, the internal min/max functions should be initialized to the highest positive number to avoid the comparison between the first element of the next iteration and the last element of the current iteration. This is implemented through the introduction of a MUX before the min/max functions and is controlled by a control signal (signal `C` in Fig. B.8). This control signal is incremented every clock cycle and is initialized to zero at the end of each iteration. For instance, when `C=1`, the input to the first two min/max functions are initialized to the largest 16-bit number (i.e., `16'b1`), thus once the first K-Best candidate of the next iteration comes in, it would not be compared with the previous stored values in the register banks from the previous iteration. This makes the core utilization 100% as `PE I` and `PE II` are fully pipelined with zero latency with respect to one another.

The scheduling between `PE II` and `PE I` blocks along with their control signals such as signal `C`, `Ctrl(PE I)`, and `Ctrl(PE II)` is shown in Fig. B.10. Two `Rst` signals are two input signals whereas the other signals are internal signals that perform the scheduling of various data exchange operations occurring in the two blocks. For instance, one clock cycle after the `Ctrl` signal of `PE II` becomes high, the new input signals are inserted into `PE II` and the internal signal `C` is initialized, which guarantees the proper implementation of comparison and avoids the interference between two consecutive iterations. It takes 5 clock cycles to insert all the entries into `PE II` and takes 5 more clock cycles to generate the first two FC candidates in the output of `PE II`. This is exactly the time when the `Rst` signal of `PE I` is raised high to initiate reading the FCs two at a time from the preceding

`PE II` with the help of its `Ctrl` signal, which performs as an internal counter determining the correct scheduling.

All of the above blocks are interconnected together in a pipelined fashion and every clock cycle data exchange occurs between the adjacent blocks. This means all the data are calculated and transferred sequentially operand-by-operand between the blocks. A proper scheduling scheme at the input of the chip guarantees the delivery of the correct $\bar{r}_{ij}$ and $\bar{z}_i$ values to the blocks.

C Proposed QR Decomposition Core - Input/Output Data Schedule

As described in Section 5.5, the proposed QRD core processes a new 4×4 complex channel matrix **H** and four 4×1 complex received symbol vectors **y** every 40 cycles, and outputs the corresponding 4×4 complex upper-triangular **R** matrix and four 4×1 complex updated symbol vectors **z** every 40 clock cycles. However, due to the pipelined architecture of the QRD core, the total latency from input **H** and **y** matrices to output **R** and **z** matrices is 160 clock cycles. This is demonstrated at a high level in Fig. C.1. As shown in Fig. C.1, the channel matrix **H1** and symbol vectors $\mathbf{y1}_1$, $\mathbf{y1}_2$, $\mathbf{y1}_3$ and $\mathbf{y1}_4$, corresponding to the first set of QRD inputs, are sampled in by the QRD core between clock cycles 1 and 40. Their corresponding QRD outputs, the **R1** matrix and the updated symbol vectors $\mathbf{z1}_1$, $\mathbf{z1}_2$, $\mathbf{z1}_3$ and $\mathbf{z1}_4$, are sampled out within the 40 clock cycles, after a 160 cycle latency. In other words, the first set of outputs are sampled out from cycles 161 to 200. Note that as shown in Fig. C.1, this schedule for input and output data repeats every 40 clock cycles.

The proposed QRD core reads in or writes out one complex number (two 16-bit Real numbers) each cycle. Table C.1 shows the schedule used to read in a 4×4 complex channel characteristic matrix **H** and four 4×1 complex received symbol vectors **y**, every 40 clock cycles. The table also shows the output schedule used by the QRD core to output a

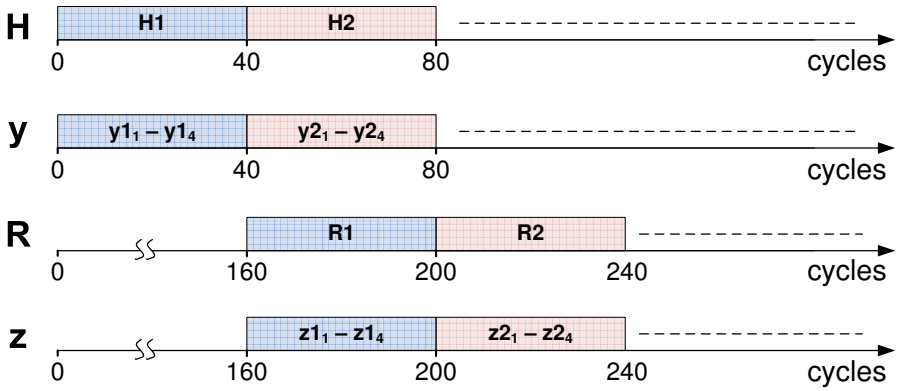

Figure C.1: Timing Schedule used for QR Decomposition Input and Output Data.

4×4 complex upper-triangular matrix **R** and four 4×1 complex updated symbol vectors **z**, every 40 cycles.

Table C.1: QRD Core - Input/Output Schedule

Input/Output Sampled at posedge and negedge of Cycle Number:	**Hy_In_Re**	**Hy_In_Im**	**Rz_Out_Re**	**Rz_Out_Im**
1	$H_{1,1}^{Re}$	$H_{1,1}^{Im}$	16'h0000	16'h0000
2	$H_{1,2}^{Re}$	$H_{1,2}^{Im}$	$R_{1,1}^{Re}$	16'h0000
3	$H_{1,3}^{Re}$	$H_{1,3}^{Im}$	$R_{1,2}^{Re}$	$R_{1,2}^{Im}$
4	$H_{1,4}^{Re}$	$H_{1,4}^{Im}$	$R_{2,2}^{Re}$	16'h0000
5	$Y_{1,1}^{Re}$	$Y_{1,1}^{Im}$	$R_{1,3}^{Re}$	$R_{1,3}^{Im}$
6	$Y_{1,2}^{Re}$	$Y_{1,2}^{Im}$	$R_{2,3}^{Re}$	$R_{2,3}^{Im}$
7	$Y_{1,3}^{Re}$	$Y_{1,3}^{Im}$	$R_{1,4}^{Re}$	$R_{1,4}^{Im}$
8	$Y_{1,4}^{Re}$	$Y_{1,4}^{Im}$	$Z_{1,1}^{Re}$	$Z_{1,1}^{Im}$
9	$H_{2,1}^{Re}$	$H_{2,1}^{Im}$	$Z_{1,2}^{Re}$	$Z_{1,2}^{Im}$
10	$H_{2,2}^{Re}$	$H_{2,2}^{Im}$	$Z_{1,3}^{Re}$	$Z_{1,3}^{Im}$
11	$H_{2,3}^{Re}$	$H_{2,3}^{Im}$	$Z_{1,4}^{Re}$	$Z_{1,4}^{Im}$
12	$H_{2,4}^{Re}$	$H_{2,4}^{Im}$	$R_{2,4}^{Re}$	$R_{2,4}^{Im}$
13	$Y_{2,1}^{Re}$	$Y_{2,1}^{Im}$	$Z_{2,1}^{Re}$	$Z_{2,1}^{Im}$
14	$Y_{2,2}^{Re}$	$Y_{2,2}^{Im}$	$Z_{2,2}^{Re}$	$Z_{2,2}^{Im}$
15	$Y_{2,3}^{Re}$	$Y_{2,3}^{Im}$	$Z_{2,3}^{Re}$	$Z_{2,3}^{Im}$
16	$Y_{2,4}^{Re}$	$Y_{2,4}^{Im}$	$Z_{2,4}^{Re}$	$Z_{2,4}^{Im}$
17	$H_{3,1}^{Re}$	$H_{3,1}^{Im}$	16'h0000	16'h0000
18	$H_{3,2}^{Re}$	$H_{3,2}^{Im}$	16'h0000	16'h0000
19	$H_{3,3}^{Re}$	$H_{3,3}^{Im}$	16'h0000	16'h0000
20	$H_{3,4}^{Re}$	$H_{3,4}^{Im}$	16'h0000	16'h0000

Input/Output Sampled at posedge and negedge of Cycle Number:	**Hy_In_Re**	**Hy_In_Im**	**Rz_Out_Re**	**Rz_Out_Im**
21	$Y_{3,1}^{Re}$	$Y_{3,1}^{Im}$	16'h0000	16'h0000
22	$Y_{3,2}^{Re}$	$Y_{3,2}^{Im}$	16'h0000	16'h0000
23	$Y_{3,3}^{Re}$	$Y_{3,3}^{Im}$	$R_{3,3}^{Re}$	16'h0000
24	$Y_{3,4}^{Re}$	$Y_{3,4}^{Im}$	16'h0000	16'h0000
25	$H_{4,1}^{Re}$	$H_{4,1}^{Im}$	$R_{3,4}^{Re}$	$R_{3,4}^{Im}$
26	$H_{4,2}^{Re}$	$H_{4,2}^{Im}$	16'h0000	16'h0000
27	$H_{4,3}^{Re}$	$H_{4,3}^{Im}$	16'h0000	16'h0000
28	$H_{4,4}^{Re}$	$H_{4,4}^{Im}$	16'h0000	16'h0000
29	$Y_{4,1}^{Re}$	$Y_{4,1}^{Im}$	16'h0000	16'h0000
30	$Y_{4,2}^{Re}$	$Y_{4,2}^{Im}$	$Z_{3,1}^{Re}$	$Z_{3,1}^{Im}$
31	$Y_{4,3}^{Re}$	$Y_{4,3}^{Im}$	$Z_{3,2}^{Re}$	$Z_{3,2}^{Im}$
32	$Y_{4,4}^{Re}$	$Y_{4,4}^{Im}$	$Z_{3,3}^{Re}$	$Z_{3,3}^{Im}$
33	16'h0000	16'h0000	$Z_{3,4}^{Re}$	$Z_{3,4}^{Im}$
34	16'h0000	16'h0000	16'h0000	16'h0000
35	16'h0000	16'h0000	16'h0000	16'h0000
36	16'h0000	16'h0000	$R_{4,4}^{Re}$	16'h0000
37	16'h0000	16'h0000	$Z_{4,1}^{Re}$	$Z_{4,1}^{Im}$
38	16'h0000	16'h0000	$Z_{4,2}^{Re}$	$Z_{4,2}^{Im}$
39	16'h0000	16'h0000	$Z_{4,3}^{Re}$	$Z_{4,3}^{Im}$
40	16'h0000	16'h0000	$Z_{4,4}^{Re}$	$Z_{4,4}^{Im}$

D QR Decomposition - Detailed Measurement Results

This Appendix presents the test results from testing five working QRD chips at 0°C, 25°C and 85°C. The detailed measurement results of all the five chips in terms of the maximum operating frequency and power consumption is documented in Table D.1 to Table D.15 for different supply voltages.

Table D.1: Measurement Results for Chip #1 @ 0^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)[1]	13.90	9.60	7.70	5.70	4.80	3.90	3.50
f (MHz)[2]	72	104	130	176	210	258	286
P (mW)[3]	19.45	22.59	27.85	32.41	39.08	45.52	51.62

[1] t: clock period. [2] f: clock frequency.
[3] P: core power @ supply voltage.

Table D.2: Measurement Results for Chip #1 @ 25^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	12.82	10.21	8.20	5.88	4.96	4.00	3.59
f (MHz)	78	98	122	170	202	250	278
P (mW)	19.63	23.57	26.25	32.56	39.98	43.36	48.20

Table D.3: Measurement Results for Chip #1 @ 85^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	14.70	11.12	9.10	6.17	5.20	4.14	3.74
f (MHz)	68	90	110	162	192	242	268
P (mW)	17.78	21.33	24.85	30.89	37.32	39.56	43.70

Table D.4: Measurement Results for Chip #2 @ 0^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	14.72	9.87	7.93	5.90	4.82	3.93	3.54
f (MHz)	68	102	126	170	208	254	282
P (mW)	15.78	19.33	24.25	30.46	37.52	43.86	49.22

Table D.5: Measurement Results for Chip #2 @ 25^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	14.28	10.86	8.62	6.02	5.05	4.10	3.67
f (MHz)	70	92	116	166	198	244	272
P (mW)	18.32	22.45	24.68	30.92	37.68	41.74	45.78

Table D.6: Measurement Results for Chip #2 @ 85^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	16.12	11.62	9.62	6.32	5.32	4.21	3.79
f (MHz)	62	86	104	158	188	238	264
P (mW)	18.83	19.29	22.84	28.45	35.33	37.26	42.52

Table D.7: Measurement Results for Chip #3 @ 0^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	12.82	9.10	7.35	5.49	4.55	3.78	3.38
f (MHz)	78	110	136	182	220	264	296
P (mW)	20.23	24.83	29.55	35.62	41.95	48.34	54.75

Table D.8: Measurement Results for Chip #3 @ 25^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	11.90	9.62	7.81	5.62	4.81	3.81	3.44
f (MHz)	84	104	128	178	208	262	290
P (mW)	19.95	23.68	27.44	33.56	40.38	45.24	50.47

Table D.9: Measurement Results for Chip #3 @ 85^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	13.51	10.20	8.77	5.88	4.96	4.00	3.59
f (MHz)	74	98	114	170	202	250	278
P (mW)	22.52	23.75	26.32	32.25	39.47	41.46	46.72

Table D.10: Measurement Results for Chip #4 @ 0^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	13.15	9.25	7.46	5.51	4.58	3.84	3.42
f (MHz)	76	108	134	182	218	260	292
P (mW)	19.45	22.59	27.32	33.79	39.25	46.42	53.32

Table D.11: Measurement Results for Chip #4 @ 25^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	13.88	9.62	7.69	5.62	4.67	3.91	3.52
f (MHz)	72	104	130	178	214	256	284
P (mW)	17.95	20.52	24.54	30.21	36.84	43.23	48.56

Table D.12: Measurement Results for Chip #4 @ 85^{o}C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	14.70	11.12	9.25	6.09	5.05	4.09	3.67
f (MHz)	68	90	108	164	198	244	272
P (mW)	20.73	21.74	24.36	30.92	37.45	39.42	44.52

Table D.13: Measurement Results for Chip #5 @ 0°C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	12.20	8.77	7.04	5.37	4.42	3.74	3.35
f (MHz)	82	114	142	186	226	268	298
P (mW)	21.45	26.38	29.55	36.34	42.74	50.84	56.55

Table D.14: Measurement Results for Chip #5 @ 25°C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	12.50	9.10	7.35	5.43	4.58	3.78	3.42
f (MHz)	80	110	136	184	218	264	292
P (mW)	17.95	20.52	24.54	30.21	36.84	43.23	48.56

Table D.15: Measurement Results for Chip #5 @ 85°C.

Supply voltage	0.7 V	0.8 V	0.9 V	1.0 V	1.08 V	1.2 V	1.32 V
t (ns)	13.88	10.63	9.10	5.81	5.00	3.96	3.59
f (MHz)	72	94	110	172	200	252	278
P (mW)	21.25	22.58	26.12	33.65	38.32	40.86	46.14

References

[1] M. Shabany, "VLSI Implementation of Digital Signal Processing Algorithms for MIMO/SISO Systems," *Ph.D. Thesis, University of Toronto*, 2009.

[2] G. Foschini and M. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–334, 1998.

[3] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO Wireless Communications*. Cambridge Univ. Press, 2007.

[4] J. Andrews, A. Ghosh, and R. Muhamed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. Pearson Education Inc., 2007.

[5] A. Salvekar, S. Sandhu, Q. Li, M.-A. Vuong, and X. Qian, "Multiple-antenna technology in WiMAX systems," *Intel Technology Journal*, vol. 8, no. 3, Aug. 2004.

[6] W. Standard, "IEEE 802.16 standard for local and metropolitan area networks, part 16: Air interface for fixed broadband wireless acess systems." *WiMAX Standard*, 2004.

[7] W. Forum, "WiMAX forum mobile system profile release 1.0 approved specification (revision 1.4.0:2007-05-02)," *WiMAX Forum*, May 2005.

[8] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge Univ. Press, 2003.

[9] B. D. V. Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Magazine*, pp. 4–24, 1998.

[10] E. G. Larsson and P. Stoica, *Space-Time Block Coding for Wireless Communications*. Cambridge Univ. Press, 2003.

[11] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-Best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.

[12] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[13] B. M. Hochwald and S. ten Brink, "Achieving Near-Capacity on a Multiple-Antenna Channel," *IEEE Trans. Commun.*, vol. 51, pp. 389–399, Mar. 2003.

[14] S. Haykin, M. Sellathurai, Y. de Jong, and T. Willink, "Turbo-MIMO for wireless communications," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 48–53, Oct. 2004.

[15] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," *in Proc. URSI ISSSE*, pp. 295–300, 1998.

[16] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest Point Search in Lattices," *IEEE Trans. on Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[17] M. O. Damen, A. Chkeif, and J. C. Belfiore, "Lattice code decoder for space-time codes," *IEEE Communications Letters*, vol. 4, no. 5, pp. 161–163, May 2000.

[18] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, pp. 463–471, Apr. 1985.

[19] J. Jalden and B. Ottersten, "On the Complexity of Sphere Decoding in Digital Communications," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[20] T. M. Aulin, "Breadth-First Maximum Likelihood Sequence Detection: Basics," *IEEE Trans. on Comm.*, vol. 47, no. 2, pp. 208–216, Feb. 1999.

[21] J. B. Anderson, "Limited search trellis decoding of convolutional codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 944–955, Sep. 1989.

[22] K. W. Wong, C. Y. Tsui, R. S. K. Cheng, and W. H. Mow, "A VLSI architecture of a K-Best lattice decoding algorithm for MIMO channels," *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, pp. 273–276, May 2002.

[23] Y. L. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *Proc. IEEE 56th Veh. Technol. Conf.*, pp. 1041–1045, 2002.

[24] J. Boleng and M. Misra, "Load balanced Parallel QR decomposition on Shared Memory Multiprocessors," *Parallel Computing*, vol. 27, pp. 1321–1345, Sep. 2001.

[25] M. Shabany and P. G. Gulak, "A 0.13μm CMOS 655Mb/s 4x4 64-QAM K-Best MIMO detector," *Proc. IEEE Int. Solid-State Circuits Conf.*, pp. 256–257, 2009.

[26] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO Signal Detector Design and VLSI Implementation," *IEEE Trans. on Very Large Scale Integration VLSI Systems*, vol. 15, no. 3, pp. 328–337, Mar. 2007.

[27] M. Shabany and P. G. Gulak, "Scalable VLSI Architecture for K-Best Lattice Decoders," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 940–943, 2008.

[28] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-Best MIMO detection VLSI architectures achieving up to 424 Mbps," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1151–1154, 2006.

[29] H.-L. Lin, R. C. Chang, and H. Chan, "A high-speed SDM-MIMO decoder using efficient candidate searching for wireless communication," *IEEE Trans. on Circuits, Syst. II*, vol. 55, no. 3, pp. 289–293, Mar. 2008.

[30] S. Chen and T. Zhang, "Low power Soft-output Signal Detector design for Wireless MIMO Communication Systems," *in Proc. International Symp. on Low Power Electronics and Design*, pp. 232–237, 2007.

[31] D. Wubben, R. Bohnke, V. Kuhn, and K. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

[32] L. Davis, "Scaled and decoupled Cholesky and QR decompositions with application to spherical MIMO detection," *in Proc. of WCNC*, vol. 1, pp. 326–331, 2003.

[33] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: John Hopkins University Press, 1996.

[34] C. Singh, S. Prasad, and P. Balsara, "VLSI Architecture for Matrix Inversion using Modified Gram-Schmidt based QR Decomposition," *International Conference on VLSI Design*, pp. 836–841, Jan. 2007.

[35] P. Salmela, A. Burian, H. Sorokin, and J. Takala, "Complex-valued QR decomposition implementation for MIMO receivers," *in Proc. IEEE ICASSP 2008*, pp. 1433–1436, Apr. 2008.

[36] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," *Proc. of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, pp. 191–200, Feb 1998.

[37] P.Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner, "VLSI Implementation of a High-Speed Iterative Sorted MMSE QR Decomposition," *in Proc. IEEE ISCAS 2007*, pp. 1421–1424, May 2007.

[38] J. Volder, "The CORDIC Trigonometric Computing Technique," *in IRE Trans. Electronic Computers*, vol. 8, no. 3, pp. 330–334, Sep 1959.

[39] ——, "A Unified Algorithm for Elementary Functions," *in Proc. AFIPS Spring Joint Computing Conf.*, vol. 38, pp. 379–385, Nov 1971.

[40] J. Delosme and S. Hsiao, "CORDIC algorithms in Four Dimensions," *Advanced Signal Processing Algorithms, Architectures, and Implementations, Proc. SPIE*, vol. 1348, no. 1, pp. 349–360, July 1990.

[41] ——, "Householder CORDIC Algorithms," *IEEE Transactions on Computers*, vol. 44, no. 8, pp. 990–1001, Aug. 1995.

[42] A. El-Amawy and K. R. Dharmarajan, "Parallel VLSI algorithm for Stable Inversion of Dense Matrices," *Computers and Digital Techniques, IEEE Proceedings*, vol. 136, no. 6, pp. 575–580, Nov. 1989.

[43] Y. T. Hwang and W. D. Chen, "A Low Complexity Complex QR Factorization Design for Signal Detection in MIMO OFDM Systems," *in Proc. IEEE ISCAS 2008*, pp. 932–935, May 2008.

[44] D. Patel, M. Shabany, and P. G. Gulak, "A Low-Complexity High-Speed QR Decomposition Implementation for MIMO Receivers," *in Proc. IEEE ISCAS 2009*, pp. 33–36, May 2009.

[45] A.Maltsev, V. Pestretsov, R. Maslennikov, and A. Khoryaev, "Triangular Systolic Array with Reduced Latency for QR-decomposition of Complex Matrices," *in Proc. IEEE ISCAS 2006*, pp. 1421–1424, May 2006.

[46] F. Sobhanmanesh and S. Nooshabadi, "Parametric minimum hardware QR-factoriser Architecture for V-BLAST Detection," *in IEEE Proceedings on Circuits, Devices and Systems*, vol. 153, no. 5, pp. 433–441, Oct 2006.

[47] S. Wang, V. Piuri, and E. Swartzlander, "A Unified View of CORDIC Processor Design," *in Proc. of IEEE 39th Midwest symposium on Circuits and Systems*, vol. 2, pp. 852–855, Aug 1996.

[48] D. Patel, "How to migrate HDL Design to ATE Test Plan Quickly and Efficiently the V93K-TestGenerator Tool," *CMC application note*, Oct. 2008.

[49] R. H. Lai, C. M. Chen, P. Ting, and Y. H. Huang, "A Modified Sorted-QR Decomposition Algorithm for Parallel Processing in MIMO Detection," *Proc. IEEE ISCAS 2009*, pp. 1405–1408, May 2009.

[50] C. P. Schnorr and M. Euchner, "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems," *Math. Programming*, vol. 66, pp. 181–191, 1994.