VLSI Implementation of Lattice Reduction for MIMO Wireless Communication Systems

by

Ameer Youssef

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

© Copyright by Ameer Youssef 2010
VLSI Implementation of Lattice Reduction for MIMO Wireless Communication Systems

Ameer Youssef

Master of Applied Science, 2010
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Lattice-Reduction has become a popular way of improving the performance of MIMO detectors. However, developing an efficient high-throughput VLSI implementation of LR has been a major challenge in the literature. This thesis proposes a hardware-optimized version of the popular LLL algorithm that reduces its complexity by 70% and achieves a fixed runtime while maintaining ML diversity. The proposed algorithm is implemented for 4x4 MIMO systems and uses a novel pipelined architecture that achieves a fixed low processing latency of 40 cycles, resulting in a fixed throughput that is independent of the channel correlation. The proposed LR core, fabricated in 0.13μm CMOS, is the first fabricated and tested LR ASIC implementation in the literature. Test results show that the LR core achieves a maximum clock rate of 204 MHz, yielding a throughput of 510 Mbps, thus satisfying the aggressive throughput requirements of emerging 4G wireless standards, such as IEEE-802.16m and LTE-Advanced.
Acknowledgments

This thesis bears my name as the sole author, yet it is truly an endeavour that could not have been accomplished without the support and encouragement of numerous people. First and foremost, I would like express my sincere gratitude and thanks to God for giving me everything that I have and to Him I owe everything that I have achieved. I would like to thank God for giving me the opportunity to study and conduct research at such a prestigious university like the University of Toronto and with a truly wonderful teacher, guide and supervisor, Professor Glenn Gulak. His crystal clear research vision, amazing technical resourcefulness, exceptional work ethic and boundless enthusiasm have made this thesis possible. In addition, I would like to thank Professor Gulak for his amazingly friendly approach and simply being a true gentleman.

I thank the members of my M.A.Sc. defense committee, Professor Frank Kschischang, Professor Jason Anderson and Professor Bruce Francis for taking the time to review my thesis and for providing expert technical comments and insightful suggestions.

I would also like to gratefully acknowledge the financial support provided by each of the University of Toronto, the Ontario Graduate Scholarship (OGS) and CMC Microsystems.

I thank Jaro Pristupa for solving CAD-related problems with speed and skill. Also, special thanks to Jeetendar Narsinghani (CMC), Dimpesh Patel and Mahdi Shabany for their help in ASIC design and testing processes. Many thanks to Mariusz Jarosz for designing the custom high speed board used in this thesis for testing.

I feel blessed for getting to know so many good friends during my studies at the University of Toronto. I have learnt a lot from them and I am grateful to all of them. Many thanks to friends from BA4162, BA5000, Professor Gulak’s group and those from outside the department. In particular, I would like to thank Dimpesh Patel, Soroush Tabatabaei, Hayssam Dahrouj, Siyu Liu, Mahdi Shabany, Peyman Razaghi, Wong Ming Fai (Patrick), Weifei Zeng, Nasim Nikkhoo, Vadim Smolyakov, Mayukh Roy, Safeen Huda, Akram Nafee, Mohamed Youssef Abdollah, David Halupka, Oleksiy Tyshchenko, Siamak Sarvari, Alireza Nilchi and William Song.
I cannot begin to express my gratitude to my wife, life partner and sole mate Sara Saker. Her selfless giving and unwavering support, in the easy times and even more when times got tough, have made this journey all the more special.

Finally, I am grateful beyond measure to my parents Ashraf Youssef and Winnifred Youssef for their unconditional love, heartfelt encouragement and continuous support. They have sacrificed so much for me to reach this milestone in my life, and I remain forever indebted to them for everything they have given me.
# Contents

List of Figures viii

List of Tables xi

1 Introduction 1
   1.1 Introduction to MIMO Systems .......................... 1
   1.2 Motivation for Lattice Reduction .......................... 3
   1.3 Thesis Objectives ........................................ 4
   1.4 Thesis Outline ......................................... 5

2 Fundamentals of MIMO Detection and Lattice Reduction 6
   2.1 MIMO-OFDM System Overview .............................. 6
      2.1.1 MIMO System Model .................................. 7
      2.1.2 Lattice Representation ............................... 9
   2.2 Simulation Framework .................................... 10
   2.3 MIMO Detection Schemes .................................. 11
      2.3.1 Optimal MIMO Detection ............................. 11
      2.3.2 Suboptimal MIMO Detection ........................... 11
      2.3.3 Near-optimal MIMO Detection ......................... 12
      2.3.4 The K-Best Detector ................................ 13
   2.4 Correlation and MIMO Detection Challenges ............... 14
   2.5 Lattice Reduction Concept ................................ 15
   2.6 Lattice Reduction Algorithms .............................. 17
      2.6.1 Gaussian Reduction .................................. 17
      2.6.2 LLL Algorithm ...................................... 22
      2.6.3 Seysen’s Algorithm .................................. 25
      2.6.4 Brun’s Algorithm ................................... 26
2.7 Considerations for LR-aided detection ........................................... 27
  2.7.1 Shifting and Scaling ........................................................... 28
  2.7.2 Boundary Control ............................................................... 28
  2.7.3 Lattice Reduction with K-Best .............................................. 28

3 Hardware-Optimized Lattice Reduction Algorithm .................. 30
  3.1 Introduction ................................................................. 30
  3.2 Analysis of Existing Lattice Reduction Algorithms .............. 30
    3.2.1 Background .............................................................. 30
    3.2.2 Proposed Analysis Framework ..................................... 32
    3.2.3 Complexity Analysis ................................................... 33
    3.2.4 Correlation and BER Performance Analysis .................... 38
    3.2.5 Impact of LR on lowering MIMO detection complexity ....... 40
  3.3 Lattice Reduction Implementation Challenges ..................... 41
  3.4 Proposed Lattice Reduction Algorithm ................................ 43
    3.4.1 MU Quantization ....................................................... 43
    3.4.2 Replacing Lovász Condition with Siegel Condition .......... 45
    3.4.3 Replacing \( \tilde{Q} \) transformation with \( \tilde{Z} \) transformation 46
    3.4.4 Replacing Multiplications with 2D CORDICs .................... 46
    3.4.5 Applying Fixed Complexity Technique ......................... 48
    3.4.6 Parameter Selection for Proposed LR Algorithm ............. 49
  3.5 Complexity Analysis and Performance Comparison ............... 51
  3.6 Summary ................................................................................. 55

4 VLSI Implementation of Hardware-Optimized Lattice Reduction .... 56
  4.1 Introduction ................................................................. 56
  4.2 Proposed Lattice Reduction Implementation - General Architecture 59
  4.3 Proposed Lattice Reduction Implementation - Detailed VLSI Architecture 61
    4.3.1 MU Calculation ........................................................... 62
    4.3.2 Complex Size Reduction ............................................... 63
    4.3.3 Siegel Calculation ....................................................... 64
    4.3.4 Basis Update Implementation ....................................... 65
    4.3.5 Balanced Pipeline Design ............................................. 67
    4.3.6 Implementation of a Single Lattice Reduction Iteration .... 68
## Contents

4.4 Proposed Lattice Reduction Implementation - Task Scheduling 69  
4.4.1 Inputs 69  
4.4.2 Scheduling of Pipeline Stages 70  
4.4.3 Other Pipeline Choices 80  
4.4.4 Iteration Scheduling 80  
4.4.5 Outputs 81  
4.4.6 LR Core Usage and Adaptability Features 81  
4.5 Fixed-Point Simulations 83  
4.6 Test Results and Design Characteristics Comparison 84  
4.6.1 Test Results 86  
4.6.2 Design Comparison 88  
4.7 Summary 92  

5 Conclusions and Future Work 94  
5.1 Conclusions 94  
5.2 Future Work 95  
5.2.1 Data Storage Handling 95  
5.2.2 Design for Various Rates of Channel Variation 95  
5.2.3 Integration of LR with MIMO Receiver Blocks 96  
5.2.4 Exploitation of LR in MIMO System Design 97  

A Proposed LR Core - Input/Output Data Schedule 98  
B Proposed LR Core - Detailed Measurement Results 101  

References 107
List of Figures

2.1 MIMO-OFDM System Model ............................... 6
2.2 Generated Decision Regions a) before LR, b) after LR ........... 16
2.3 MIMO System Model a) without LR, b) with LR ................. 17
2.4 Complex Gaussian Reduction Example - (Basis Orthogonalization Progress) 21

3.1 CDF of iterations for R/CLLL ($\delta \in \{\frac{3}{4}, 1\}$) and SEY (Greedy/Lazy) for a 4x4 MIMO system .......................... 36
3.2 CDF of iterations for R/CLLL ($\delta \in \{\frac{3}{4}, 1\}$) and SEY (Greedy/Lazy) for a 8x8 MIMO system .......................... 36
3.3 CDF of basis updates for R/CLLL ($\delta \in \{\frac{3}{4}, 1\}$) and SEY (Greedy/Lazy) for a 4x4 MIMO system .......................... 37
3.4 CDF of basis updates for R/CLLL ($\delta \in \{\frac{3}{4}, 1\}$) and SEY (Greedy/Lazy) for a 8x8 MIMO system .......................... 37
3.5 BER Performance of LR-aided KBR (K-Best Real) using LLL and SEY - for 4 × 4 MIMO System with 64-QAM and K=10 ......................... 39
3.6 BER Performance of LR-aided KBR (K-Best Real) using LLL and SEY - for 4 × 4 MIMO System with 256-QAM and K=15 ......................... 39
3.7 PDF of $\Re(\mu)$ and $\Im(\mu)$ for CLLL with $\delta \in \{\frac{3}{4}, 1\}$ for 4x4 MIMO ................. 44
3.8 Real Valued $\tilde{R}$ Matrix augmented with four $\tilde{z}$ vectors - Shown in the case of columns 1 and 2 being swapped, with the vectoring operations numbered 47
3.9 BER Performance of LR-aided KBR (K-Best Real) with Complex LR vs ML - for a 4 × 4 MIMO System with 64-QAM .......................... 50
3.10 CDF of iterations for Complex LR algorithms (CLLL, SEY and this work (HOLLL)) for a 4 × 4 MIMO System .......................... 52
3.11 CDF of basis updates for Complex LR algorithms (CLLL, SEY and this work (HOLLL)) for a 4 × 4 MIMO System .......................... 52
3.12 BER Performance of LR-aided ZF & V-BLAST with Complex LR algorithms (CLLL and this work (HOLLL)) - for $4 \times 4$ MIMO with 64-QAM  

3.13 BER Performance of LR-aided KBR (K-Best Real, K=10) with Complex LR algorithms (CLLL and this work (HOLLL)) - for $4 \times 4$ MIMO with 64-QAM  

4.1 The Proposed Pipelined VLSI Architecture for the Hardware-Optimized LLL Algorithm for a $4 \times 4$ MIMO System  

4.2 Architecture for the Quantized $\mu_q$ Calculation Block (block I/O names are shown in parenthesis).  

4.3 Architecture for the Complex Size Reduction (CSR) Block (block I/O names are shown in parenthesis).  

4.4 Architecture for Siegel Condition with Dynamic $\delta$ Control (block I/O names are shown in parenthesis).  

4.5 Block Diagram of the Configurable 9-Stage Pipelined CORDIC.  

4.6 Block Diagram for one HOLLL iteration including clock cycles.  

4.7 Size reduction schedule k=2.  

4.8 Size reduction schedule k=3.  

4.9 Size reduction schedule k=4.  

4.10 Illustration of Size Reduction operations for k=4.  

4.11 CORDIC pipeline schedule for $k = 2$.  

4.12 CORDIC pipeline schedule for $k = 3$.  

4.13 CORDIC pipeline schedule for $k = 4$.  

4.14 CORDIC wrapper for efficient implementation of feedback paths.  

4.15 Input and Output Data Timing Schedule for LR Core.  

4.16 Fixed Point Simulation of Hardware-Optimized LLL (HOLLL) for $4 \times 4$ 64-QAM K-Best Real (KBR, K=10).  

4.17 Die Micrograph of HOLLL Lattice Reduction 0.13$\mu$m CMOS Core.  

4.18 Test Setup using Verigy 93000 SoC tester, Temptronic TP04300 thermal forcing unit and the DUT (LR Core). The chip package leads make connection to the circuit board pads via elastomers, which are used to accommodate small deviations in the pitch and height of circuit board pads. (An elastomer is a sheet of insulating rubber housing multiple rows of symmetrically embedded metal filaments which form electrical connections.)
List of Figures

4.19 Measured BER at 510Mb/s for Hardware-Optimized LLL (HOLLL) with 4 × 4 64-QAM K-Best Real (K=10). .................................................. 87
4.20 Measured Maximum Operating Frequency and Power Dissipation vs. Supply Voltage at 25°C. .................................................. 88
## List of Tables

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Complex Gaussian Reduction Algorithm</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>Complex LLL Algorithm (CLLL)</td>
<td>24</td>
</tr>
<tr>
<td>3.1</td>
<td>The complexity of LLL and SEY LR Algorithms for $4 \times 4$ and $8 \times 8$ MIMO systems.</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Post-LR Correlation Comparison for LLL and SEY LR Algorithms for $4 \times 4$ and $8 \times 8$ MIMO systems.</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison between different $4 \times 4$ K-Best Implementations</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>Hardware Optimized LLL Algorithm (HOLLL)</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Quantization of $\mu$</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparison between LR algorithms for $4 \times 4$ MIMO</td>
<td>51</td>
</tr>
<tr>
<td>3.7</td>
<td>Post-LR Correlation Comparison for Complex LR Algorithms (CLLL, Greedy SEY and This Work (HOLLL)) for $4 \times 4$ and $8 \times 8$ MIMO systems.</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>Fixed-point Word-Length (bits) of LR Data Variables</td>
<td>83</td>
</tr>
<tr>
<td>4.2</td>
<td>Chip Characteristics and Comparison to Current LR VLSI Implementations</td>
<td>89</td>
</tr>
<tr>
<td>4.3</td>
<td>Summary of LR Design Characteristics and Measured Results</td>
<td>93</td>
</tr>
<tr>
<td>A.1</td>
<td>LR Core - Input/Output Schedule</td>
<td>99</td>
</tr>
<tr>
<td>B.1</td>
<td>Measurement Results for Chip #1 @ $0^\circ$C.</td>
<td>102</td>
</tr>
<tr>
<td>B.2</td>
<td>Measurement Results for Chip #1 @ $25^\circ$C.</td>
<td>102</td>
</tr>
<tr>
<td>B.3</td>
<td>Measurement Results for Chip #1 @ $80^\circ$C.</td>
<td>102</td>
</tr>
<tr>
<td>B.4</td>
<td>Measurement Results for Chip #2 @ $0^\circ$C.</td>
<td>103</td>
</tr>
<tr>
<td>B.5</td>
<td>Measurement Results for Chip #2 @ $25^\circ$C.</td>
<td>103</td>
</tr>
<tr>
<td>B.6</td>
<td>Measurement Results for Chip #2 @ $80^\circ$C.</td>
<td>103</td>
</tr>
<tr>
<td>B.7</td>
<td>Measurement Results for Chip #3 @ $0^\circ$C.</td>
<td>104</td>
</tr>
<tr>
<td>B.8</td>
<td>Measurement Results for Chip #3 @ $25^\circ$C.</td>
<td>104</td>
</tr>
</tbody>
</table>
B.9 Measurement Results for Chip #3 @ 80°C. ............................ 104
B.10 Measurement Results for Chip #4 @ 0°C. ............................ 105
B.11 Measurement Results for Chip #4 @ 25°C. ............................ 105
B.12 Measurement Results for Chip #4 @ 80°C. ............................ 105
B.13 Measurement Results for Chip #5 @ 0°C. ............................ 106
B.14 Measurement Results for Chip #5 @ 25°C. ............................ 106
B.15 Measurement Results for Chip #5 @ 80°C. ............................ 106
**List of Symbols**

- $d$: Displacement vector for shifting and scaling in lattice reduction
- $h_k$: Updated $k$th column of $H$ in Seysen’s algorithm
- $s$: Complex transmitted symbol vector
- $\hat{s}$: Estimated complex transmitted symbol vector
- $u$: Selected column of $(HH^H)^{-1}$ in Brun’s algorithm
- $v$: Complex noise vector
- $x$: Post-LR complex transmitted symbol vector
- $x(n)$: Transmitted bit vector at time $n$
- $w$: Post-processed complex noise vector
- $y$: Complex received symbol vector
- $z$: Post-processed complex received symbol vector
- $z$: LR-reduced post-processed complex received symbol vector
- $C$: Inverse of LR complex transformation matrix $T$ in Brun’s algorithm
- $G$: Gram matrix of $H$, defined as $G = H^H H$
- $K$: Number of K-Best candidates in each level of the tree
- $H$: Complex MIMO channel matrix
- $\tilde{H}$: LR-reduced complex channel matrix
- $H^d$: Dual of complex matrix $H$, where $H^d \triangleq H(H^H H)^{-1}$
- $\mathcal{L}(H)$: Lattice of channel matrix $H$, defined as the span of the columns of $H$
- $|\mathcal{L}|$: Volume of a fundamental cell of the lattice $\mathcal{L}$
- $\mathcal{L}^d$: Dual of the lattice $\mathcal{L}$
- $M$: Constellation size/ordinality
- $M_c$: Number of bits per constellation point
- $N_c$: Complex Gaussian distribution
- $N_R$: Number of receive antennas
- $N_T$: Number of transmit antennas
- $\mathcal{O}$: Complex constellation
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>Unitary matrix with complex entries</td>
</tr>
<tr>
<td>$R$</td>
<td>Upper triangular matrix with complex entries</td>
</tr>
<tr>
<td>$R$</td>
<td>Number of bits per channel use</td>
</tr>
<tr>
<td>$S(H)$</td>
<td>Seysen orthogonality measure of $H$</td>
</tr>
<tr>
<td>$T$</td>
<td>LR complex transformation matrix</td>
</tr>
<tr>
<td>$Z$</td>
<td>The set of integers</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>Noise variance</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Subtraction factor in complex Gaussian reduction</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Rounding error in complex Gaussian reduction</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Size reduction coefficient in LLL</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Swapping control factor in LLL</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>Complex entries in Givens rotation matrix $\Theta$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Update value in Seysen’s algorithm</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Unrounded update value in Seysen’s algorithm</td>
</tr>
<tr>
<td>$\xi$</td>
<td>Size reduction factor in Brun’s algorithm</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>Set of possible real entries in $O$</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Complex Givens rotation matrix in LLL</td>
</tr>
<tr>
<td>$\kappa(H)$</td>
<td>Condition number of $H$</td>
</tr>
<tr>
<td>$\Lambda(H)$</td>
<td>Orthogonality defect of $H$</td>
</tr>
</tbody>
</table>
List of Notations

\( a \) Scalar value
\( |a| \) Absolute value of \( a \)
\( \mathbf{a} \) Complex-valued vector
\( \mathbf{\hat{a}} \) Real-valued vector
\( (\mathbf{a})^* \) Complex-conjugate of \( \mathbf{a} \)
\( \mathbf{a}^T \) Transpose of \( \mathbf{a} \)
\( \mathbf{\tilde{a}} \) Post-LR version of \( \mathbf{a} \)
\( \|\mathbf{a}\| \) Norm of \( \mathbf{a} \)
\( \langle \mathbf{a}, \mathbf{b} \rangle \) Inner product of \( \mathbf{a} \) and \( \mathbf{b} \)

\( \mathbf{A} \) Complex-valued matrix
\( \mathbf{\bar{A}} \) Real-valued matrix
\( \mathbf{\tilde{A}} \) LR-Reduced version of \( \mathbf{A} \)
\( \mathbf{A}^{-1} \) Inverse of \( \mathbf{A} \)
\( \mathbf{A}^H \) Conjugate transpose of \( \mathbf{A} \)
\( A_{ij} \) Entry in \( i \)-th row and \( j \)-th column of \( \mathbf{A} \)
\( \mathbf{a}_i \) \( i \)-th column of \( \mathbf{A} \)
\( \mathbf{A}(a:b,c:d) \) Submatrix of \( \mathbf{A} \), formed by rows \( a \) to \( b \) and columns \( c \) to \( d \).

\( \mathcal{R}\{\cdot\} \) Real part of a complex number
\( \mathcal{I}\{\cdot\} \) Imaginary part of a complex number
\( \lceil \cdot \rceil \) Rounding to the nearest integer
List of Acronyms

4G Fourth Generation
LR Lattice Reduction
ADC Analog-to-Digital Converter
AoA Angle of Arrival
AoD Angle of Departure
ASIC Application-Specific Integrated Circuit
AWGN Additive White Gaussian Noise
BER Bit-Error-Rate
bpcu bits per channel use
BS Base Station
BU Basis Update
CGR Complex Gaussian Reduction
CLLL Complex LLL
CMOS Complementary Metal Oxide Semiconductor
CORDIC COordinate Rotation DIgital Computer
CSR Complex Size Reduction
CU Channel Use
DAC Digital-to-Analog Converter
List of Acronyms

DUT Device Under Test
ELLL Effective LLL
FFT Fast Fourier Transform
FPGA Field-Programmable Gate Array
Gbps Giga bits per second \((10^9 \text{ bits/sec})\)
GDS Graphic Database System
HOLLL Hardware Optimized LLL
IFFT Inverse Fast Fourier Transform
ISI Inter-Symbol Interference
LLL Lenstra, Lenstra and Lovász
LR Lattice Reduction
LSB Least Significant Bit
LTE Long Term Evolution
Mbps Mega bits per second \((10^6 \text{ bits/sec})\)
MIMO Multiple-Input Multiple-Output
ML Maximum-Likelihood
MMSE Minimum Mean Squared Error
MS Mobile Station
MSB Most Significant Bit
MUX Multiplexer
OFDM Orthogonal Frequency-Division Multiplexing
PED Partial Euclidean Distance
**List of Acronyms**

- **PSK**  Phase Shift Keying
- **QAM**  Quadrature Amplitude Modulation
- **QoS**  Quality-of-Service
- **QR**  QR Decomposition
- **RLLL**  Real LLL
- **RS-LLL**  Reverse Siegel LLL
- **RVD**  Real-Valued Decomposition
- **SD**  Sphere Decoding
- **SISO**  Single-Input Single-Output
- **SM**  Spatial Multiplexing
- **SNR**  Signal-to-Noise-Ratio
- **SoC**  System on Chip
- **S/P**  Serial to Parallel Conversion (Demux)
- **SR**  Size Reduction
- **V-BLAST**  Vertical Bell Laboratories Layered Space-Time
- **VCD**  Value Change Dump
- **VLSI**  Very Large Scale Integration
- **XPD**  Cross-Polarization Discrimination
- **WiMAX**  Worldwide Interoperability for Microwave Access
- **WLAN**  Wireless Local Area Network
- **WMAN**  Wireless Metropolitan Area Network
- **ZF**  Zero-Forcing
1 Introduction

1.1 Introduction to MIMO Systems

The use of multiple transmit and receive antennas in wireless communication systems, known as multiple-input multiple-output (MIMO) systems [1], is one of the most significant advances in wireless technology in the past decade. MIMO offers tremendous gains in data rates, signal reliability and range, which has made it currently the technology of choice in many state-of-the-art wireless standards [2]. For example, in the Wireless Local Area Network (WLAN) IEEE 802.11n standard, MIMO is the key enabling technology for achieving target data rates of up to 480Mbps. MIMO is also used in achieving the high data rates of IEEE 802.16e Wireless Metropolitan Area Network (WMAN) system also known as Worldwide Interoperability for Microwave Access (WiMAX) [3], as well as the next generation WiMAX for high mobility systems, the IEEE 802.16m standard [4].

The 3rd Generation Partnership Project (3GPP), which is a global initiative for defining next generation mobile communication standards, uses MIMO technology as a basis for the Long Term Evolution (LTE) standard with data rates of 100Mbps for downlink and 50Mbps for uplink [5], as well as the upcoming 4G LTE-Advanced standard that aims at 1Gbps for downlink and 500Mbps for uplink [6].

These wireless standards take advantage of the high spectral efficiency of MIMO systems to achieve much higher data rates compared to traditional single-input single-output (SISO) systems. In addition, high spectral efficiency leads to higher system reliability, increased coverage area and lower transmission power requirements. The multitude of transmit and receive antennas enables this high spectral efficiency due to three types of performance gains [7]:

- **Array Gain**: Multiple receive antennas are able to pick up a larger share of the transmitted power, which naturally extends the communication range. Additionally, this increase in received signal power leads to a higher signal-to-noise ratio (SNR), thus enhancing the resistance to noise and suppressing interference.
• **Diversity Gain**: Transmitted signals experience non-deterministic fluctuations in the attenuation of their signal power, known as fading. This leads to a received signal with constantly, and randomly, varying amplitude. Fading, which greatly diminishes the quality of the communication link, can be mitigated through transmitting copies of the signal experiencing (ideally) independent fading paths. Multiple receive antennas combine the arriving signals and reconstruct the transmitted signal with much less amplitude variability compared the traditional SISO case. The diversity order is equal to the number of independently fading paths, which corresponds to the slope of the bit-error-rate (BER) curve. Without transmit channel knowledge and assuming an optimum receiver with $N_R$ antennas, the maximum diversity order is $N_R$.

• **Spatial Multiplexing Gain**: The increased number of transmit and receive antennas means that multiple data streams can now be transmitted in parallel through the same channel, thus “reusing” the spatial dimension (hence the term “spatial multiplexing” or SM). Through this parallelization technique, a linear increase in peak data rates can be achieved by transmitting $N_T$ multiple independent data streams of lower rates. This leads to substantially larger channel capacity rates for MIMO channels compared to the traditional SISO channel. The SM gain depends on the number of parallel streams, and thus is limited to the minimum of $N_T$ and $N_R$.

There are tradeoffs between the three aforementioned gains and, based on the application, MIMO systems use different transmission schemes to maximize one particular gain. For instance, beamforming techniques are used in conjunction with MIMO systems to maximize array gain [8], while the space-time coding scheme is used to exploit diversity and thus increases link reliability and Quality-of-Service (QoS) [9,10]. This thesis focuses on the MIMO systems which maximize the spatial multiplexing (SM) gain. In these systems, the MIMO detector is required to simultaneously separate and process the spatially multiplexed data streams received by the $N_R$ receive antennas. This greatly increases the complexity of detection algorithms. On the other hand, the high data rates and mobility requirements of emerging fourth generation (4G) standards, such as WiMAX and LTE-Advanced, necessitate Very Large Scale Integration (VLSI) implementations that are capable of very high throughput as well as being power and area efficient. As a result, designing low-complexity MIMO detectors that optimize this tradeoff between performance and complexity has become one of the most challenging aspects of MIMO systems.


1.2 Motivation for Lattice Reduction

One of the main factors limiting the detection performance of MIMO receivers is the correlation between the basis vectors which form the channel matrix. On one hand, the Maximum-Likelihood (ML) detector makes decisions based on optimal closest-point search [11]. Therefore, its results are independent of the basis used and consequently the ML detector is not strongly affected by the correlation of the basis chosen to represent the channel \(^1\). On the other hand, non-optimal detectors (both linear and non-linear) depend heavily on the choice of the basis and as a result these detectors cannot achieve full diversity unless the basis vectors are perfectly orthogonal (i.e. they have zero correlation) which is very rarely the case in a typical channel, due to multi-path effects and antenna design constraints [12, 13]. Moreover, the higher the correlation between the basis vectors, the lower the diversity achieved by non-optimal detectors. As a result, they perform very poorly in the presence of ill-conditioned channels (channels with highly correlated basis vectors). The conclusion is that in order to close the gap to ML performance and achieve higher diversity, the channel matrix must be transformed so that it is represented by an orthogonal (or nearly-orthogonal) basis, thereby minimizing the correlation and maximizing the achievable diversity.

Recently, the application of Lattice Reduction (LR) techniques to MIMO communication systems was proposed with the aim of closing the gap to the ML performance limit [14]. The term lattice reduction refers to the linear transformation of the basis vectors representing the channel matrix so that they become orthogonal or nearly-orthogonal without changing the actual constellation points. This in effect decreases the correlation between the basis vectors and results in a better-conditioned channel matrix. The LR technique used in [14] is the Gaussian reduction algorithm, which is the simplest form of lattice reduction and is only applicable to the 2x2 MIMO case. Using lattice reduction significantly improved the performance of linear detectors and the diversity achieved was shown to match that of ML. This was a major milestone since no previous linear MIMO detector could achieve ML diversity.

This compelling improvement in the performance of linear detectors as a result of using lattice reduction initiated a large number of research efforts directed at expanding on

---

\(^1\)Theoretically, the performance of the ML detector is independent of the basis, however, this is not true when practical limitations of hardware implementation are taken into account. Section 2.4 discusses these limitations and shows how lattice reduction can be used to greatly simplify the hardware complexity of the ML detector.
the work of [14]. This included: (a) expanding the application of lattice reduction to arbitrarily higher dimensions (4 × 4, 6 × 6, 8 × 8, …) by using LR algorithms other than Gaussian reduction, and (b) extending these new LR techniques to apply to non-linear detectors. However, for dimensions higher than $2 \times 2$, the lattice reduction technique becomes exponentially more complex. As a result, the associated LR algorithms exhibit high complexity as well as a non-deterministic runtime, rendering them unsuitable for a practical hardware implementation.

1.3 Thesis Objectives

LR techniques have the potential for not only significantly improving the performance of MIMO detectors, but also enabling the reduction of the detection complexity while maintaining low BER performance. This reduction in complexity of LR-aided MIMO detectors, which is made possible by the improved condition of the channel matrix, allows for higher throughput as well as more efficient and cost-effective VLSI implementations, thereby enabling MIMO systems to match the aggressive specifications of 4G wireless standards such as data rates of up to 1Gbps, large constellation orders (64-QAM and 256-QAM) and large antenna configurations ($4 \times 4$ and $8 \times 8$).

In order to realize the true potential of lattice reduction, there is a need to develop a near-optimal low-complexity LR algorithm suitable for high-throughput VLSI implementation. This has yet to be achieved in a practical hardware system. To date, there have only been a few hardware implementations of LR algorithms, namely [15–19]. Each of these implementations attempts to optimize the LR algorithm for hardware implementation. However, because of the inherent complexity and iterative nature of the existing LR algorithms, these implementations suffer from high computational complexity, low-throughput and non-deterministic runtime, and therefore cannot be used in a realtime MIMO receiver. In addition, most of these LR designs have been implemented on Field-Programmable Gate Array (FPGA) platforms, with none realized and silicon-proven in a practical high performance Application-Specific Integrated Circuit (ASIC) chip implementation.

The objective of this thesis is to design and implement the first LR ASIC core with silicon-verified results. The designed LR core is required to meet the high data rates envisioned for 4G standards such as WiMAX and LTE-Advanced. This objective can be broken down into several steps as follows:
1 Introduction

1. Development of an optimized LR algorithm: This entails examining the existing state-of-the-art LR algorithms, selecting the most efficient algorithm in the context of MIMO detection and then optimizing the selected algorithm for an efficient VLSI implementation. Optimization goals are to reduce complexity, achieve deterministic runtime and reduce the number of required iterations, all while maintaining high quality lattice reduction.

2. Efficient Hardware Design: Once an optimized algorithm is reached, an efficient architecture must be designed for it. The VLSI design is required to have high-throughput necessary for 4G data rates as well as low power required for implementation in 4G mobile systems with limited power resources. In addition, bit-true simulations must ensure that the lattice reduction quality remains high and is only marginally affected by the fixed-point implementation.

3. Silicon-proven implementation: Following hardware simulation and verification, the design is implemented on an ASIC platform. The fabricated chip should be tested and shown to meet 4G throughput specifications as well as low power requirements.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 introduces the MIMO system model and notation used along with a background on MIMO detection methods. This is followed by a detailed examination of the lattice reduction technique and some of the popular algorithms used to implement it. Chapter 3 presents a performance and complexity analysis of existing LR algorithms and introduces the proposed lattice reduction scheme which has been optimized for hardware implementation. The various improvements to the algorithm are discussed along with a complexity and performance comparison with traditional LR algorithms. Chapter 4 describes the VLSI implementation of the proposed LR algorithm. The architectures of both top-level and individual blocks are discussed along with the scheduling implemented for each. The chip specifications and measured performance from the ASIC LR core are presented and compared to current state-of-the-art LR VLSI implementations. Chapter 5 concludes this thesis by summarizing the benefits of the proposed LR implementation followed by a discussion of future work in lattice reduction within the MIMO detection context.
2 Fundamentals of MIMO Detection and Lattice Reduction

2.1 MIMO-OFDM System Overview

Orthogonal Frequency Division Multiplexing (OFDM) is a very popular wireless communication modulation scheme, which has been developed to efficiently handle communication over multiple transmitter and receiver antennas [20]. OFDM is characterized by the transmission of multiple parallel data streams, referred to as subcarriers, such that the subcarriers are orthogonal to each other in the frequency domain. Each subcarrier is modulated through traditional modulation schemes such as Quadrature Amplitude Modulation (QAM) or Phase Shift Keying (PSK). The orthogonality of the subcarriers makes OFDM very robust against InterSymbol Interference (ISI), and can be efficiently implemented through an Inverse Fast Fourier Transform (IFFT) operation on the transmitter side and a Fast Fourier Transform (FFT) operation on the receiver side.

A simplified architecture of a MIMO-OFDM system is shown in Fig. 2.1. Starting with the transmitter on the left side, a binary source produces a binary sequence which is demuxed into $N_T$ signals and then modulated to points on the constellation through the mapper. The modulated signals are passed through an IFFT operation, converted to their analog counterparts and transmitted via $N_T$ antennas. The transmitted signals pass

![Figure 2.1: MIMO-OFDM System Model](image-url)
through a channel modeled by the matrix $H$, and then they are received by $N_R$ antennas. After an analog to digital conversion and an FFT operation, the received signals are fed to two paths: one is the main detection path that includes the MIMO detector, while the other path is used for channel estimation and preprocessing. The estimated and preprocessed channel is forwarded to the MIMO Detector, which produces the estimated symbols. These symbols are then multiplexed, demapped and converted into the estimated binary sequence.

### 2.1.1 MIMO System Model

The system model shown in Fig. 2.1 is now explored in more detail with mathematical definitions and system equations. Let us consider a MIMO system with $N_T$ transmit and $N_R$ receive antennas. In this thesis, it is always assumed that $N_R \geq N_T$. At time $n$, the bit sequence $\mathbf{x}(n) = [x_1(n), x_2(n), \ldots, x_{M_cN_T}(n)]^T$ is sent to $N_T$ parallel streams using a serial-to-parallel (S/P) block, which are mapped into a complex vector $\mathbf{s}(n) = [s_1(n), s_2(n), \ldots, s_{N_T}(n)]^T$ by $N_T$ linear modulators at the transmitter front end$^1$. Each element $s_i(n)$ is taken from a complex constellation $\mathcal{O}$, such as rectangular Quadrature Amplitude Modulation (QAM), composed of $M = |\mathcal{O}| = 2^{M_c}$ distinct points, meaning that every $M_c$ consecutive bits are mapped to one complex constellation point. This implies that $\mathbf{s} \in \mathcal{O}^{N_T}$, 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 $R = N_T \log_2 M = N_T M_c$ 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 $\mathbf{s}$ is normalized before transmission in such a way that the average transmitted power is one (i.e. $E\{\|\mathbf{s}\|^2\} = 1$).

The equivalent baseband model of the MIMO wireless channel that yields the $N_R$-dimensional received vector $\mathbf{y} = [y_1, y_2, \ldots, y_{N_R}]^T$ is given by the following equation:

$$\mathbf{y} = \mathbf{Hs} + \mathbf{v}, \quad (2.1)$$

where $\mathbf{H} = \{H_{ij}\}_{i=1}^{N_R} \_{j=1}^{N_T}$ denotes the $N_R \times N_T$ dimensional channel matrix representing the complex-valued channel gains between each transmit and each receive antenna. In

---

$^1$In this thesis, vectors are distinguished from scalars by using a bold font, where $\mathbf{x}$ is a vector and $x$ is a scalar value. Furthermore, matrices are denoted in upper case with bold font such as $\mathbf{H}$.
addition, $\mathbf{v} = [v_1, v_2, \cdots, v_{N_R}]^T$ 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 $\sigma^2$ per complex dimension, i.e., $v_i \sim \mathcal{N}(0, \sigma^2)$. Furthermore, the entries of $\mathbf{H}$ 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., $\text{SNR} = 1/\sigma^2$.

The objective of the MIMO detector at the receiver is to obtain the best possible estimate of the transmitted signal vector $\mathbf{s}$ in the Euclidean sense based on the received vector $\mathbf{y}$. This is represented by the following equation:

$$\hat{s} = \arg \min_{s \in \mathbb{C}^{N_T}} \| \mathbf{y} - \mathbf{Hs} \|^2.$$  \hspace{1cm} (2.2)

In this thesis it is assumed that the channel estimator block in the MIMO receiver provides an accurate estimate of the channel $\mathbf{H}$, which can be obtained via a separate training phase with the aid of pilot symbols [21].

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

In addition to the complex channel model described above, the equivalent real model can also be derived using a real-valued decomposition (RVD) scheme [22]. However, in this thesis, a slightly different RVD scheme is used. This modified RVD scheme is more suitable for concurrent computations and VLSI implementation. The real model$^2$ of (2.1) can be written as:

$$\bar{\mathbf{y}} = \bar{\mathbf{H}} \bar{s} + \bar{\mathbf{v}},$$  \hspace{1cm} (2.3)

where $\bar{\mathbf{y}} = [\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_{2N_R-1}, \bar{y}_{2N_R}]^T$ and $\bar{s} = [\bar{s}_1, \bar{s}_2, \cdots, \bar{s}_{2N_T-1}, \bar{s}_{2N_T}]^T$ are the equivalent

$^2$Note that in this thesis, real-valued vectors and matrices are distinguished from their complex-valued counterparts by a bar on the top of the symbol. For example, $\bar{\mathbf{y}}$ denotes a real-valued vector while $\mathbf{y}$ indicates a complex-valued vector. For a comprehensive list of notation used, the reader is referred to the List of Notations section at the beginning of this thesis.
real-valued vectors with the following mappings:

\[
\begin{align*}
\bar{y}_{2k-1} &= \mathcal{R}(y_k), & \bar{y}_{2k} &= \mathcal{I}(y_k), \\
\bar{s}_{2k-1} &= \mathcal{R}(s_k), & \bar{s}_{2k} &= \mathcal{I}(s_k), \\
\bar{v}_{2k-1} &= \mathcal{R}(v_k), & \bar{v}_{2k} &= \mathcal{I}(v_k),
\end{align*}
\]  

(2.4)

In addition, the real-valued channel matrix \( \bar{H} \) is derived from the complex channel matrix \( H \) based on the following mapping:

\[
\bar{H} = \begin{bmatrix}
\mathcal{R}(H_{11}) & \mathcal{I}(H_{11}) & \cdots & \mathcal{R}(H_{1N_T}) & \mathcal{I}(H_{1N_T}) \\
\mathcal{I}(H_{11}) & \mathcal{R}(H_{11}) & \cdots & \mathcal{I}(H_{1N_T}) & \mathcal{R}(H_{1N_T}) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\mathcal{R}(H_{N_R1}) & \mathcal{I}(H_{N_R1}) & \cdots & \mathcal{R}(H_{N_RN_T}) & \mathcal{I}(H_{N_RN_T}) \\
\mathcal{I}(H_{N_R1}) & \mathcal{R}(H_{N_R1}) & \cdots & \mathcal{I}(H_{N_RN_T}) & \mathcal{R}(H_{N_RN_T})
\end{bmatrix}_{2N_R \times 2N_T}, \tag{2.5}
\]

where \( \mathcal{R}(\cdot) \) and \( \mathcal{I}(\cdot) \) denote the real and imaginary parts of a complex variable, respectively. Note that the resulting detection equation in the real domain will be as follows:

\[
\bar{s} = \arg \min_{\bar{s} \in \Omega} \| \bar{y} - \bar{H}\bar{s} \|^2, \tag{2.6}
\]

where \( \Omega \) is the set of possible real entries in the constellation for in-phase and quadrature parts as follows:

\[
\bar{s}_i \in \Omega = \left\{ \frac{(-\sqrt{M} + 1)}{E_s}, \ldots, \frac{-1}{E_s}, \frac{1}{E_s}, \ldots, \frac{(+\sqrt{M} - 1)}{E_s} \right\}, \tag{2.7}
\]

where \( E_s = 2(M - 1)/3 \) is the average symbol energy for an M-QAM constellation.

### 2.1.2 Lattice Representation

For a given matrix \( H \), the lattice \( \mathcal{L}(H) \) is defined to be the span of the columns of \( H \), which is the set of all integer linear combinations generated by the columns of \( H \). Based on this definition, the columns of \( H \) are also referred to as the basis vectors for \( \mathcal{L}(H) \). The set \( \{Hs\} \) represents a subset of \( \mathcal{L}(H) \) and the transmitted vector \( s \) can be considered as the coordinates of a lattice point. Therefore, the received signal vector \( y \) is an observation of lattice point \( Hs \) corrupted by complex AWGN \( v \). Based on this interpretation, another
2 Fundamentals of MIMO Detection and Lattice Reduction

way to describe (2.1) is to say that the objective of the MIMO detection method is to find the closest lattice point \( \hat{s} \) based on an observation \( y \) within a given lattice \( \mathcal{L}(H) \) and with the constraint that the found lattice point must be a point in the constellation \( \mathcal{O} \).

The QR-decomposition of the estimated channel matrix is an important preprocessing step required by many MIMO detectors. QR decomposition refers to decomposing the channel matrix as \( H = QR \), where \( Q \) is a unitary \( N_R \times N_T \) matrix and \( R \) is an upper triangular \( N_T \times N_T \) matrix. The upper triangular nature of \( R \) facilitates a suitable framework for sequential detection schemes. Performing the following nulling operation by \( Q^H \) yields the updated system equation as follows:

\[
z = Q^H y = Q^H H s + Q^H v = R s + w, \quad (2.8)
\]

where \( w = Q^H v \). Since the nulling matrix is unitary, the noise, \( w \), remains spatially white and thus there is no noise enhancement. The modified MIMO detection equation becomes:

\[
\hat{s} = \arg \min_{s \in \mathcal{O}^{N_T}} \| z - R s \|^2 = \arg \min_{s \in \mathcal{O}^{N_T}} \sum_{i=1}^{N_T} |z_i - \sum_{j=i}^{N_T} R_{ij} s_j|^2. \quad (2.9)
\]

2.2 Simulation Framework

The bit-error-rate (BER) results in this thesis have been obtained from computer simulations and/or tested chip measurements based on the i.i.d. channel model assumption. In addition, all presented simulation results assume perfect channel knowledge at the receiver so that the channel estimation and detection can be separated. Furthermore, in this thesis it is assumed, for the purpose of MIMO detection and preprocessing, that the channel is quasi-static and is updated every four consecutive channel uses (i.e. the channel is valid over four consecutive received symbol vectors).

With regards to the modulation scheme of choice, BER performance results in this thesis will focus on LR-aided detection using 64-QAM modulation\(^4\). This is due to the fact that 64-QAM is 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

\(^3\)Note that \( Q^H \) refers to the conjugate transpose of \( Q \), which is defined as taking the transpose of \( Q \) and then the complex conjugate of each entry, i.e. \( Q^H = (Q^T)^\ast \).

\(^4\)Although, this thesis shows that the complexity and performance of lattice reduction is independent of the chosen modulation, however the chosen modulation is of significance when evaluating the performance of LR-aided MIMO detectors.
802.11n WLAN (2 × 2 MIMO), 3GPP LTE and LTE-Advanced. Both floating-point and fixed-point simulation results are presented and discussed in this work.

2.3 MIMO Detection Schemes

As previously mentioned, the task of the MIMO detector is to provide a decision on the transmitted symbol vector \( \mathbf{s} \) given the received vector \( \mathbf{y} \). There are various MIMO detection algorithms that exist to perform this task. Based on their relative detection accuracy, these detection algorithms can be classified into three groups: optimal, suboptimal and near-optimal.

2.3.1 Optimal MIMO Detection

The most common optimal MIMO detector is the Maximum-Likelihood (ML) detector. The ML detector is optimal in the sense of achieving the lowest BER performance. In the presence of additive white Gaussian noise (AWGN) it can be shown that ML detection is equivalent to searching for the lattice point \( \mathbf{s} \) that lies on the constellation \( \mathcal{O} \) and is closest to the received point \( \mathbf{y} \) in the lattice \( \mathcal{L}(\mathbf{H}) \). This leads to optimal detection performance. However, the drawback is that the complexity of ML grows exponentially with both the number of transmit antennas and the constellation size \( M \). For this reason, ML is generally not practical for implementation in MIMO receivers, rather it is used as a reference point in simulations to judge the BER performance of other MIMO detectors.

2.3.2 Suboptimal MIMO Detection

Suboptimal detectors can be subdivided into two groups: a) linear detectors and b) non-linear suboptimal detectors. Linear detectors implement linear equalization algorithms with the aim of reversing the effect of the channel. This linear cancellation approach processes the parallel data streams all at once without accounting for order, hence leading to a low complexity detector. However these linear detectors can only achieve a maximum diversity of \( N_R - N_T + 1 \), resulting in a poor BER performance, especially in symmetric systems where \( N_R = N_T \). The two most common linear detectors are the Zero-Forcing (ZF) detector and the Minimum-Mean-Squared-Error (MMSE) detector [2]. On the other hand, non-linear suboptimal detectors rely on detecting the symbols in order, from strongest to
weakest, with the decisions made on earlier symbols used to decide on later symbols. This scheme, known as Successive Interference Cancellation (SIC), is used in the well-known Vertical Bell Labs Layered-Space-Time (V-BLAST) detector [23]. Compared to linear detection schemes, the SIC approach increases the diversity order with each iteration. The first detected symbol sees a diversity order of $N_R - N_T + 1$, the second sees a diversity order of $N_R - N_T + 2$ and so forth. The SIC approach does achieve a better BER performance compared to linear detection schemes, however, SIC suffers from error propagation, with the BER performance being dominated by the data stream detected first.

2.3.3 Near-optimal MIMO Detection

Near-optimal MIMO detectors implement non-linear algorithms and are capable of achieving near-ML performance with much less complexity than ML. Recall that MIMO detection can be interpreted as finding the closest lattice point $\hat{s}$ for a given lattice $L(H)$, with the condition that the found lattice point lies on the constellation. This problem can in fact be restated as a tree-search problem, with the leaves of the tree representing the set of potential solutions. The ML approach considers all the leaves when searching for a solution, and thus it is optimal but also suffers from exponential complexity. However, the complexity can be reduced significantly if the leaves that are unlikely to lead to the desired solution can be removed from the search. This can be achieved by eliminating entire subtrees, which lead to these unlikely solutions, based on a pre-defined performance metric, a process known as tree-pruning.

Tree-searching methods, which rely on tree-pruning, fall into two major categories, namely depth-first methods and breath-first methods. The Sphere-Decoding (SD) detector [24] is the most common depth-first method. SD, which constrains the tree-search to points lying inside a sphere with a pre-defined radius, has ML-performance under the assumption of unlimited execution time. The main disadvantage of SD is that its throughput is dependent on the SNR value, since the SNR is used to determine the sphere constraint. This leads to a variable throughput rate, which results in extra overhead in the hardware due to the extra required buffers and lower hardware utilization.

The most popular breadth-first method is the K-Best algorithm. K-Best 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 schemes with no feedback, makes the K-Best detection algorithm especially attractive.
for VLSI implementation. In addition, the tradeoff between performance and complexity can be controlled by adjusting the $K$ factor in the K-Best algorithm. For these reasons, this thesis focuses on the K-Best detector as the primary MIMO detection scheme for applying lattice reduction. The following section offers a brief overview of the K-Best algorithm along with its performance and complexity challenges, which this work proposes to solve through lattice reduction techniques.

### 2.3.4 The K-Best Detector

To understand the K-Best algorithm, consider an $N_R \times N_T$ $M$-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. Therefore, given an implementation in the real-domain, the equation 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 at that level based on a criterion [25]. To clarify this, consider $K$ surviving nodes in level $i$. Each of these nodes has $\sqrt{M}$ possible children in level $i + 1$, from the symmetry in the M-QAM constellation. The K-Best algorithm visits all these children and calculates their Partial Euclidean Distances (PEDs) resulting in $K\sqrt{M}$ children in level $i + 1$. Once the PED values are calculated, the K-Best algorithm sorts all these $K\sqrt{M}$ existing paths according to their accumulated PEDs and selects the $K$ best paths as the surviving nodes in level $i + 1$. Note that a larger value of $K$ leads to better performance but higher complexity, while a lower value of $K$ lowers the complexity of the detector but at the expense of degrading the BER performance. As a result, the K-Best algorithm offers a trade-off between optimality and complexity with respect to the value of $K$ [26].

Based on the above description, the steps in the K-Best algorithm (in the real domain) can be summarized as:

1. **Initialization**: Set one path at level $2N_T + 1$ with PED = 0.
2. **Expansion**: Expand each of the $K$ surviving paths from the previous step to $\sqrt{M}$ new possible children in $\Omega$ and calculate the updated PED for each resulting path (i.e. a total of $K\sqrt{M}$ children).
3. **Sorting**: Sort all $K\sqrt{M}$ existing paths according to their accumulated PEDs and select the best $K$ paths.
4. If not at the last level, go to step 2, otherwise stop.
2.4 Correlation and MIMO Detection Challenges

Tree-search detection schemes such as K-Best perform best, in the sense of near-ML performance, when the signal space is orthogonal, i.e., the basis vectors representing the lattice are orthogonal. However, in most practical scenarios, there exists correlation between the channel basis vectors leading to inferior detection performance. This correlation arises due to scattering effects [12] as well as physical geometry constraints of antenna arrays, both at the transmitter and receiver [13]. Scattering is heavily dependent on the nature of the environment in which the wireless communication is taking place, and hence it is generally outside of the control of the designer. As for antenna arrays, the designer does have much more control. However, it is difficult to design antenna arrays for MIMO systems in a way that completely eliminates antenna correlation. Furthermore, antenna correlation leads to other types of correlation in MIMO systems such as fading correlation [12].

To offer a brief insight into the antenna design tradeoffs, two important factors of antenna array design are considered, namely, 1) Antenna spacing and 2) Antenna polarization. In order to achieve low antenna correlation and increase spatial diversity, the antennas must be placed far apart, about 10 carrier wavelengths at the Base Station (BS) and at least 0.5 wavelengths at the Mobile Station (MS) [27]. The smaller spacing at the MS is possible due to the much wider Angle of Arrival (AoA) resulting from scattering compared to the narrow Angle of Departure (AoD) at the BS. Moreover, close spacing of antennas causes mutual coupling which leads to further increasing the correlation between the antenna signals [1]. On the other hand, there is a need to reduce antenna spacings due to the increased drive for reducing form factor and power consumption, which requires shrinking BS equipment and MS device sizes.

The second design consideration that also generates a similar tradeoff is the antenna polarization, which refers to the relative orientation of the antennas in the antenna array. Antenna correlation can be reduced by placing the antennas perpendicular to each other, more generally referred to as cross-polarized antennas. This concept can apply to both transmit and receive antennas. Another advantage of cross-polarization is that perpendicular antennas can be co-located and still have sufficiently low correlation, leading to a reduction of overall equipment size [28]. However, compared to linear polarization (i.e., parallel antennas), cross-polarization suffers from Cross-Polarization Discrimination (XPD), which refers to the imbalance of received signal power at each of the receive antennas [27]. In extreme cases, the XPD can be very large such that the diversity gain
is severely degraded, so much so that it begins to approach the case of a single receive antenna.

As mentioned in Section 1.2, LR has been introduced as channel preprocessing step which can significantly mitigate the effects of channel correlation by orthogonalizing the basis vectors of the estimated channel. Applying LR to near-optimal MIMO detectors has the potential to reduce the sensitivity of such detectors to channel correlation, thus making the detectors more robust and able to maintain near-ML performance over wide ranges of channel conditions. Moreover, LR can even be applied to ML detectors to significantly lower their complexity. Although the performance of ML detectors is based on a closest point search, making it independent of basis vector correlation, however, to maintain high-throughput in a practical hardware implementation, there is necessarily a limited precision for numbers, with fixed-point implementations preferred over floating-point ones. Moreover, the non-orthogonal decision regions caused by basis vector correlation, complicates the design of hardware comparators. By orthogonalizing decision regions, LR allows the use of simple hardware comparators that reduces the overall complexity of ML detectors.

2.5 Lattice Reduction Concept

For a given basis \( \mathbf{H} \), lattice reduction means to transform \( \mathbf{H} \), via a linear transformation matrix \( \mathbf{T} \), into a new basis \( \tilde{\mathbf{H}} = \mathbf{HT} \) which is more orthogonal. This transformation is guaranteed to produce the same lattice, i.e., \( \mathcal{L}(\tilde{\mathbf{H}}) = \mathcal{L}(\mathbf{H}) \), if and only if, the \( N_T \times N_T \) matrix \( \mathbf{T} \) is unimodular, i.e., \( \mathbf{T} \) contains only complex integer entries [29] with \( \det(\mathbf{T}) = \pm 1 \). Using the above transformation, (2.1) can be rewritten as:

\[
y = \mathbf{Hs} + \mathbf{v} = \mathbf{HTT}^{-1}s + \mathbf{v} = \tilde{\mathbf{H}}x + \mathbf{v} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}x + \mathbf{v}, \tag{2.10}
\]

where \( \tilde{\mathbf{Q}}\tilde{\mathbf{R}} \) is the QR-decomposition of \( \tilde{\mathbf{H}} \) and \( \mathbf{x} = \mathbf{T}^{-1}s \). Note that \( \mathbf{Hs} \) and \( \tilde{\mathbf{H}}x \) describe the same point in a lattice but the LR-reduced matrix \( \tilde{\mathbf{H}} \) is much better conditioned than the original channel matrix \( \mathbf{H} \) as a result of the more orthogonal basis vectors of \( \tilde{\mathbf{H}} \). The detector finds an estimated \( \hat{\mathbf{x}} \) in the lattice-reduced constellation and estimates the original symbol via \( \hat{\mathbf{s}} = \mathbf{T}\hat{\mathbf{x}} \). Such detection has better performance due to a better-conditioned channel matrix \( \tilde{\mathbf{H}} \). Note that by applying LR, the MIMO detection equation becomes,

\[
\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{C}^{N_T}} \| \mathbf{z} - \tilde{\mathbf{R}}\mathbf{x} \|^2. \tag{2.11}
\]
The idea behind orthogonalizing the basis vectors of the estimated channel matrix, is to reduce their correlation and make the decision regions closer to that of the ideal regions of the ML detector. This is illustrated in Fig. 2.2 below, where the basis vectors and decision regions before and after LR reduction are shown for a $2 \times 2$ case. Clearly, the decision regions generated by the orthogonal basis vectors are more noise-resistant than decision regions generated by non-orthogonal basis vectors.

![Figure 2.2: Generated Decision Regions a) before LR, b) after LR](image)

It is important to observe that the process of lattice reduction does not affect the transmitter side in any way. Lattice reduction is essentially a re-interpretation of the transmitted signal ($\mathbf{x}$ instead of $\mathbf{s}$) based on the new lattice-reduced channel matrix on the receiver side. The lattice-reduced matrix is used to improve detection quality through its more orthogonal basis vectors. However, the detected symbol is $\mathbf{x}$, and by re-transforming it into $\mathbf{s}$ (by multiplying it by $\mathbf{T}$), the re-interpretation made earlier is cancelled out. This process is illustrated in Fig. 2.3.
2 Fundamentals of MIMO Detection and Lattice Reduction

Detector

\[ v \]
\[ H \]
\[ s \]
\[ \hat{s} \]

Detector

\[ \hat{y} \]
\[ s \]
\[ \hat{s} \]

Detector

\[ \hat{h} = HT \]
\[ x \]
\[ \hat{x} \]
\[ T \]

Detector for \( s \)

\[ y \]

Figure 2.3: MIMO System Model a) without LR, b) with LR

2.6 Lattice Reduction Algorithms

There are many lattice reduction algorithms in the mathematical literature. However, in terms of its application to the MIMO detection context, there are three main algorithms which have been proposed: 1) the LLL algorithm [30], 2) Seyser’s Algorithm [31] and 3) Brun’s algorithm [32]. The fundamentals of these three algorithms, which apply to arbitrary matrix dimensions, are presented in this section\(^5\). To help illustrate the basic concept of an LR algorithm in action, we first present the foundational LR algorithm for 2 × 2 matrices that initiated the idea of using LR for improving MIMO detection, namely, the Gaussian reduction algorithm [14].

2.6.1 Gaussian Reduction

Gaussian reduction is the simplest form of lattice reduction. Within the context of MIMO detection, it was first proposed by [14] and it only applies to 2 × 2 case. The main idea of Gaussian reduction is to iteratively subtract integer copies from one vector of \( H \) out of the other to reduce their correlation. Before exploring the details of the algorithm, it is

\(^5\)In Chapter 3, it is shown that the complex version of LLL (CLLL) is the most efficient of these LR algorithms. Hence, CLLL is chosen for further optimizations to produce the proposed Hardware-Optimized LLL (HOLLL).
necessary to first introduce the following important theorem [14]:

**Theorem 2.1** If \( u \) and \( v \) are the two shortest non-zero vectors in the lattice, then \( u \) and \( v \) are the optimal basis for detection (optimal in the sense of achieving lowest BER performance), which is also the “most orthogonal” basis possible for the given lattice.

Based on the above theorem, an alternative to finding the two most orthogonal basis vectors in a lattice is to find the two shortest basis vectors. Consider the \( 2 \times 2 \) complex channel case with \( H = [h_1 \ h_2] \). Let \( \tilde{h}_1 \) denote the component of \( h_1 \) that is orthogonal to \( h_2 \) and denote \( \tilde{h}_2 \) as the component of \( h_2 \) that is orthogonal to \( h_1 \). Arbitrarily set \( h_1 \) to be the shorter vector, i.e. \( \|h_1\| < \|h_2\| \). Start by subtracting \( \eta \) complex integer copies of \( h_1 \) from \( h_2 \) to obtain the first instance of \( \tilde{h}_2 \). This can be represented as:

\[
\tilde{h}_2 = h_2 - \eta h_1,
\]

(2.12)

Where \( \eta \) is chosen such that it minimizes the inner product \( \langle h_1, \tilde{h}_2 \rangle \), which represents the correlation between \( \tilde{h}_2 \) and \( h_1 \).

\[
\eta = \arg \min_{n \in \mathbb{Z}+j\mathbb{Z}} |\langle h_1, \tilde{h}_2 \rangle| = \arg \min_{n \in \mathbb{Z}+j\mathbb{Z}} |\langle h_1, h_2 \rangle - n\|h_1\|^2| = \left\lfloor \frac{\langle h_1, h_2 \rangle}{\|h_1\|^2} \right\rfloor,
\]

(2.13)

Note that the \( \lfloor \cdot \rfloor \) operation refers to rounding to the nearest integer, with real and imaginary components rounded separately. The next step is to replace \( h_2 \) with \( \tilde{h}_2 \), and find the new correlation as follows:

\[
\langle h_1, \tilde{h}_2 \rangle = \left\langle h_1, \left( h_2 - \left\lfloor \frac{\langle h_1, h_2 \rangle}{\|h_1\|^2} \right\rfloor h_1 \right) \right\rangle = \left( \left\lfloor \frac{\langle h_1, h_2 \rangle}{\|h_1\|^2} \right\rfloor \|h_1\|^2 \right) \cdot \|h_1\|^2 = \rho \cdot \|h_1\|^2
\]

(2.14)

where \( \rho \) represents the rounding error. Since the real and imaginary rounding errors will be no more than \( \frac{1}{2} \), (i.e. \( R(\rho) < \frac{1}{2} \) and \( I(\rho) < \frac{1}{2} \)), therefore \( R(\langle h_1, \tilde{h}_2 \rangle) < \frac{1}{2} \) and \( I(\langle h_1, \tilde{h}_2 \rangle) < \frac{1}{2} \). Following this, check if the new vector \( \tilde{h}_2 \) is shorter than \( h_1 \), if so then swap them and check if further subtraction is possible (i.e. if more iterations are possible).

The complete Gaussian reduction algorithm [14] is summarized in Table 2.1.

---

6The term “most orthogonal basis vectors” is used to refer to the basis vectors with the lowest inner product (after taking the absolute value), or equivalently the angle between them being the closest to 90 degrees.
Table 2.1: Complex Gaussian Reduction Algorithm

1) Compute \( \langle h_1, h_2 \rangle \)
   if ( \( \Re((h_1, \tilde{h}_2)) < \frac{1}{2}\|h_1\|^2 \) \&\& \( \Im((h_1, \tilde{h}_2)) < \frac{1}{2}\|h_1\|^2 \) )
   stop;
   else
   replace \( h_2 \) with \( h_2 - \left( \frac{\langle h_1, h_2 \rangle}{\|h_1\|^2} \right) h_1 \), and go to Step 2;
   end

2) Compute new \( \|h_2\| \)
   if ( \( \|h_2\| > \|h_1\| \) )
   stop;
   else
   swap \( h_1 \) and \( h_2 \), and go to Step 1;
   end

To illustrate the Complex Gaussian Reduction (CGR) algorithm in action, a simple example is provided below. This example will also illustrate the general concept of size reduction of basis vectors and how it leads to a more orthogonal basis. The progression of the basis vectors throughout the algorithm is also illustrated in Fig. 2.4.

Example 2.1: Complex Gaussian Reduction Example

Consider a \( 2 \times 2 \) channel matrix \( H = \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix} \). Therefore the two basis vectors are

\( h_1 = \begin{bmatrix} 6 \\ 8 \end{bmatrix} \) and \( h_2 = \begin{bmatrix} 7 \\ 9 \end{bmatrix} \). The CGR algorithm proceeds as follows:

Iteration 1:

1) \( \|h_1\| = 10 < \|h_2\| = \sqrt{130} \approx 11.4 \)

2) Compute: \( \langle h_1, h_2 \rangle = h_2^T h_1 = \begin{bmatrix} 6 \\ 8 \end{bmatrix} \begin{bmatrix} 7 & 9 \end{bmatrix} = 114 \)

3) Check: \( \langle h_1, h_2 \rangle \leq \frac{1}{2}\|h_1\|^2 = 50 \) \( \Rightarrow \) No, then continue.
4) Compute: 
\[ \tilde{h}_2 = \left( h_2 - \left( \frac{\langle h_1, h_2 \rangle}{\| h_1 \|^2} \right) h_1 \right) = \left[ \begin{array}{c} 7 \\ 9 \end{array} \right] - \left[ \begin{array}{c} 114 \\ 100 \end{array} \right] \cdot \left[ \begin{array}{c} 6 \\ 8 \end{array} \right] = \left[ \begin{array}{c} 1 \\ 1 \end{array} \right] \]

\[ \Rightarrow \text{new } h_2 = \left[ \begin{array}{c} 1 \\ 1 \end{array} \right] \]

5) Check: \[ \| h_2 \| = \sqrt{2} > \? \| h_1 \| = 10 \Rightarrow \text{No, then swap and go to Iteration 2.} \]

**Iteration 2:**

New Basis: \[ h_1 = \left[ \begin{array}{c} 1 \\ 1 \end{array} \right] \text{ and } h_2 = \left[ \begin{array}{c} 6 \\ 8 \end{array} \right] \]

1) \[ \| h_1 \| = \sqrt{2} < \| h_2 \| = 10 \]

2) Compute: \( \langle h_1, h_2 \rangle = h_2^T h_1 = [6 \ 8] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 14 \)

3) Check: \( \langle h_1, h_2 \rangle \leq \frac{1}{2} \| h_1 \|^2 = 1 \Rightarrow \text{No, then continue.} \)

4) Compute: 
\[ \bar{h}_2 = \left( h_2 - \left( \frac{\langle h_1, h_2 \rangle}{\| h_1 \|^2} \right) h_1 \right) = \left[ \begin{array}{c} 6 \\ 8 \end{array} \right] - \left[ \begin{array}{c} 14 \\ \frac{1}{2} \end{array} \right] \cdot \left[ \begin{array}{c} 1 \\ 1 \end{array} \right] = \left[ \begin{array}{c} -1 \\ 1 \end{array} \right] \]

\[ \Rightarrow \text{new } h_2 = \left[ \begin{array}{c} -1 \\ 1 \end{array} \right] \]

5) Check: \[ \| h_2 \| = \sqrt{2} > \? \| h_1 \| = \sqrt{2} \Rightarrow \text{No, then swap and go to Iteration 3.} \]

**Iteration 3:**

New Basis: \[ h_1 = \left[ \begin{array}{c} -1 \\ 1 \end{array} \right] \text{ and } h_2 = \left[ \begin{array}{c} 1 \\ 1 \end{array} \right] \]

1) \[ \| h_1 \| = \| h_2 \| = \sqrt{2} \]

2) Compute: \( \langle h_1, h_2 \rangle = h_2^T h_1 = [1 \ 1] \begin{bmatrix} -1 \\ 1 \end{bmatrix} = 0 \Rightarrow \text{Stop. (note: } h_1 \perp h_2) \)
Figure 2.4: Complex Gaussian Reduction Example - (Basis Orthogonalization Progress)
2.6.2 LLL Algorithm

Extending Theorem 2.1 to the general case of $N_T$ basis vectors, the general lattice reduction problem is equivalent to finding a basis with short and nearly orthogonal vectors. Based on this, a popular criterion for lattice reduction is to find a basis $\tilde{H}$ such that the product $\|\tilde{h}_1\| \cdot \ldots \cdot \|\tilde{h}_{N_T}\|$ is minimized [33]. This is equivalent to minimizing the orthogonality defect defined as:

$$\Lambda(\tilde{H}) \triangleq \frac{1}{|L|} \prod_{n=1}^{N_T} \|\tilde{h}_n\|^2$$

(2.15)

where $|L| = \det(H^H H)$ is the volume of a fundamental cell of the lattice $L$ and is independent of the basis chosen for the lattice. Note that $\Lambda(\tilde{H}) \geq 1$ with equality, if and only if, the basis of $\tilde{H}$ is orthogonal. The problem of finding such an orthogonal basis is NP-hard [34]. Several suboptimal LR algorithms have been developed in the literature, of which LLL is the most popular LR algorithm used in the MIMO detection context. The LLL algorithm, due to Lenstra, Lenstra and Lovász [30], was first proposed in 1982, but has only recently risen to prominence within MIMO detection research, particularly after the exciting potential of lattice reduction was discovered. LLL is essentially the generalization of Gaussian reduction to arbitrarily higher dimensions, and has become very popular due to its polynomial-time complexity and guarantee of a bounded orthogonality defect.

The LLL algorithm was originally proposed for real-valued matrices but has also been extended in [35] to work with complex-valued matrices. The Real-valued LLL (RLLL) algorithm was used in [36] to expand the 2x2 lattice reduction work in [14] to higher dimensions. The application of the Complex-valued LLL (CLLL) to MIMO detection was later introduced in [29]. Being a generalization of CGR, LLL shares the same core idea of achieving a more orthogonal basis via iterative size reduction operations coupled with appropriate column swapping. In the size reduction operations, the lengths of the basis vectors are reduced by subtracting from each vector its integer components in each of the previous smaller vectors (i.e. those vectors which have already been processed). This contributes to increasing the orthogonality of the basis vectors by reducing their length in a pair-wise manner. Once all possible size reduction operations have been completed, the next step in LLL compares the length of the current basis vector to the previous one and swaps them if they are not in ascending order. This reordering is done with the aim of allowing further size reductions to take place. In this manner, LLL is considered a local
The CLLL algorithm (complex version of LLL) [24] will be now examined in detail and is shown in Table 2.2. The CLLL algorithm takes the QR-decomposition of the channel matrix as an input and iteratively reduces the correlation between the basis vectors of the channel matrix to produce a near-orthogonal basis for matrix $\tilde{H} = \tilde{Q}\tilde{R}$ that satisfies:

$$|\Re(\tilde{R}_{l,k})|, |\Im(\tilde{R}_{l,k})| \leq \frac{1}{2}|\tilde{R}_{l,l}| \quad \forall \quad 1 \leq l \leq k \leq N_T$$

$$\delta|\tilde{R}_{k-1,k-1}|^2 \leq |\tilde{R}_{k-1,k}|^2 + |\tilde{R}_{k,k}|^2 \quad \forall \quad 2 \leq k \leq N_T.$$  

(2.16)
(2.17)

These two conditions are known as the size reduction and the Lovász basis swapping condition, respectively. Moving from $k = 2$ to $N_T$, the algorithm performs basis reduction operations to size-reduce each $k$th column of $\tilde{R}$ against its previous $1:k-1$ columns (lines 4-8). As before, $\lfloor \cdot \rfloor$ operation in line (5) refers to rounding to the nearest integer. After the size reduction, the Lovász condition is checked for the $k$th and $(k-1)$th columns of $\tilde{R}$; if it passes, then the two columns are swapped and Givens Rotations are applied to maintain the upper-triangular nature of $\tilde{R}$ (lines 9-14), otherwise the algorithm proceeds to the next column pair. The $\delta \in [\frac{1}{4}, 1]$ factor controls the tradeoff between the speed of the algorithm and the quality of the reduced basis, e.g., $\delta = 1$ gives the highest quality but slowest execution, while $\delta = \frac{1}{4}$ gives the fastest execution but lowest quality. The common choice $\delta = \frac{3}{4}$ achieves a good balance between speed and quality\(^7\). The outputs of CLLL are the updated $\tilde{Q}$, $\tilde{R}$ and $T$ matrices.

\(^7\)For complex LLL and ($\delta = \frac{3}{4}$), the authors in [35] show that the orthogonality defect is bounded to the following: $\sqrt{\Lambda(\tilde{H})} \leq 2^{N_T(N_T-1)}$. 

---

[23] Fundamentals of MIMO Detection and Lattice Reduction
Table 2.2: Complex LLL Algorithm (CLLL)

\[(\tilde{Q}, \tilde{R}, T) = \text{CLLL}(Q, R, \delta)\]

1) \(\tilde{Q} = Q; \tilde{R} = R; \ T = I_{N_T \times N_T};\)

2) \(k = 2;\)

3) \(\text{while } k \leq N_T\)

4) \(\text{for } l = k - 1 : -1 : 1\)

5) \(\mu = \left[\tilde{R}_{l,k} / \tilde{R}_{l,l}\right];\)

6) \(\tilde{R}(1 : l, k) = \tilde{R}(1 : l, k) - \mu \cdot \tilde{R}(1 : l, l);\)

7) \(T(:, k) = T(:, k) - \mu \cdot T(:, l);\)

8) \(\text{end}\)

9) \(\text{if } \delta \cdot |\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2\)

10) \(\text{Swap (} k - 1 \text{)th and } k \text{th columns in } \tilde{R} \text{ and } T;\)

11) \(\Theta = \begin{bmatrix} \alpha^* & \beta \\ -\beta & \alpha \end{bmatrix}, \quad \alpha = \frac{||\tilde{R}(k - 1 : k, k - 1)||}{||\tilde{R}_{k,k-1}||}, \quad \beta = \frac{||\tilde{R}(k - 1 : k, k - 1)||}{||\tilde{R}_{k,k-1}||};\)

12) \(\tilde{R}(k - 1 : k, k - 1 : N_T) = \Theta \tilde{R}(k - 1 : k, k - 1 : N_T);\)

13) \(\tilde{Q}(; : k - 1 : k) = \tilde{Q}(; : k - 1 : k) \Theta^H;\)

14) \(k = \max(k - 1, 2);\)

15) \(\text{else}\)

16) \(k = k + 1;\)

17) \(\text{end}\)

18) \(\text{end}\)
2.6.3 Seysen’s Algorithm

Another important lattice reduction algorithm is Seysen’s algorithm (SEY) [31], which has been recently applied to MIMO detection in [37]. SEY performs global reduction compared to the local pair-wise reduction approach of LLL. In addition, SEY simultaneously reduces a basis $\mathbf{H}$ and its dual $\mathbf{H}^d$ of the dual lattice $\mathcal{L}^d$ (where $\mathbf{H}^d \triangleq \mathbf{H}(\mathbf{H}^H\mathbf{H})^{-1}$). The basis orthogonality is increased by iteratively reducing a measure known as Seysen’s orthogonality measure, defined as:

$$S(\tilde{\mathbf{H}}) \triangleq \sum_{n=1}^{N_T} ||\tilde{h}_n||^2||\tilde{h}_n^d||^2,$$

where $S(\tilde{\mathbf{H}}) \geq N_T$, and its minimum value $N_T$ is achieved if and only if $\tilde{\mathbf{H}}$ is orthogonal. A basis $\mathbf{H}$ is SEY-reduced if a (local) minimum of $S(\tilde{\mathbf{H}})$ is found. This minimum is found through an iterative basis reduction process, where in each iteration an index pair $(k, l)$ with $k, l \in \{1, \ldots, N_T\}$ is selected and a corresponding update value $\lambda_{kl}$ is used to update the basis as follows:

$$\tilde{\mathbf{H}} = [\tilde{\mathbf{h}}_1 \cdots \tilde{\mathbf{h}}_{k-1} \tilde{\mathbf{h}}_k' \tilde{\mathbf{h}}_{k+1} \cdots \tilde{\mathbf{h}}_{N_T}] \quad \text{with} \quad \tilde{\mathbf{h}}_k' = \tilde{\mathbf{h}}_k + \lambda_{kl} \tilde{\mathbf{h}}_l,$$

As shown in [37], the best update value $\lambda_{ij}$ such that $S(\tilde{\mathbf{H}})$ is minimized is:

$$\lambda_{ij} \overset{\Delta}{=} [\gamma_{ij}] = \left[ \frac{1}{2} \left( \frac{\tilde{G}_{ji}^d}{\tilde{G}_{ii}} - \tilde{G}_{jj}^d \right) \right] = 0 \quad \forall \ 1 \leq i \neq j \leq N_T,$$

where $\tilde{\mathbf{G}}$ is the Gram matrix of $\tilde{\mathbf{H}}$, i.e. $\tilde{\mathbf{G}} = \tilde{\mathbf{H}}^H \tilde{\mathbf{H}}$ and $\tilde{\mathbf{G}}^d = \tilde{\mathbf{H}}^d \tilde{\mathbf{H}}^d$ is $\tilde{\mathbf{H}}^d = (\tilde{\mathbf{H}}^H \tilde{\mathbf{H}})^{-1}$.

In each iteration, SEY starts by calculating the update values $\lambda_{ij}$. Next, $\tilde{\mathbf{H}}$, $\tilde{\mathbf{H}}^d$ and $\mathbf{T}$ are updated as follows:

$$\tilde{h}_k' = \tilde{h}_k + \lambda_{kl} \tilde{h}_l,$$
$$\tilde{h}_l^d = \tilde{h}_l^d - \lambda_{kl}^* \tilde{h}_k^d,$$
$$t_k' = t_k + \lambda_{kl} t_l,$$

where the indices $(k, l)$ are selected according to (2.22) so that the decrease in $S(\tilde{\mathbf{H}})$ is
maximized. Following a derivation in [37], the indices are selected as follows:

\[
(k, l) = \arg \max_{(i,j)} \left\{ 2 \tilde{G}_{jj} \tilde{G}^d_{ii} \left( 2 \Re \{ \lambda^* \gamma_{ij} \} - |\lambda_{ij}|^2 \right) \right\}.
\]

SEY combined with this optimal index selection is known as “Greedy SEY”, since in each iteration the highest possible reduction is achieved. However, it does have \(O(N_T^2)\) computational complexity since it requires \(N_T^2 - N_T \) index tests. An \(O(N_T)\) but suboptimal approach, known as “Lazy SEY” [38], selects any update index pair \((k, l)\) such that \(\lambda_{kl} \neq 0\), which is guaranteed to reduce \(S(\tilde{H})\) [37]. This makes the index selection process linear, but leads to a higher number of average iterations, as later shown in Chapter 3.

Following basis updates, the \(\lambda_{ij}\) values corresponding to updated entries of \(\tilde{G}\) and \(\tilde{G}^d\) are calculated. The new optimal \(\lambda_{kl}\) is selected and the reduction process described above repeats until \(\lambda_{ij} = 0\) \(\forall (ij)\), which represents a local minimum of \(S(\tilde{H})\).

### 2.6.4 Brun’s Algorithm

In addition to LLL and SEY, there is a third less popular lattice reduction technique which has been used in the MIMO context. This technique is based on Brun’s algorithm for finding approximate integer relations in [32]. This algorithm applies specifically to poorly conditioned channel matrices that only have a single small eigenvalue. This is relevant since i.i.d. Gaussian channels with large condition number usually have just a single eigenvalue which causes the occurrence of long basis vectors as shown in [39,40]. The application of the Brun’s algorithm minimizes the effect of this small eigenvalue resulting in a better conditioned channel matrix.

For a given \(N_R \times N_T\) channel matrix \(H\), consider the lattice \(\mathcal{L}\) to be the lattice generated by the span of the columns of \(H\), which can be considered basis vectors as discussed earlier. Brun’s algorithm increases the basis orthogonality by iteratively reducing the orthogonality defect \(\Lambda(\tilde{H})\). The algorithm proceeds as follows:

**Step 1:** Initialization: Select an arbitrary column of \((HH^H)^{-1}\), set a vector \(u^{(0)}\) equal to it. Initialize the following: \(H^{(0)} = H, T^{(0)} = I, C^{(0)} = I = T^{-1(0)}, i = 0\).

**Step 2:** Select the indices \(s\) and \(t\) of the two largest entries of \(u^{(i)}\)

\[
s = \arg \max_{n \in \{1, \ldots, N_T\}} |u^{(i)}_n| \quad \text{and} \quad t = \arg \max_{n \in \{1, \ldots, N_T\} / s} |u^{(i)}_n|
\]
Step 3: Compute the reduction factor $\xi$:

$$
\xi = \left\lfloor \frac{u_s^{(i)}}{u_t^{(i)}} \right\rfloor = \left\lfloor \frac{u_s^{(i)} \cdot u_t^{(i)} \ast}{|u_t^{(i)}|} \right\rfloor \quad (2.23)
$$

Step 4: Perform the following updates using the $\xi$ reduction factor:

$$
\begin{align*}
    u_s^{(i+1)} &= u_s^{(i)} - \xi u_t^{(i)} \\
    h_s^{(i+1)} &= h_s^{(i)} - \xi^* h_t^{(i)} \\
    t_s^{(i+1)} &= t_s^{(i)} - \xi^* t_t^{(i)} \\
    c_t^{(i+1)} &= c_t^{(i)} + \xi^* c_s^{(i)}
\end{align*}
$$

(2.24)

Step 5: Check termination condition: $\|h_s^{(i+1)}\| > ? \|h_s^{(i)}\|$ (i.e. check if $\Lambda(\tilde{H})$ has stopped decreasing). If so, then return $u^{(i)}$, $\tilde{H} = H^{(i)}$, $T = T^{(i)}$ and $T^{-1} = C^{(i)}$. Otherwise, increment $i$ and go back to Step 2.

As a result of the reduced scope of this algorithm, it achieves a much lower complexity than either LLL or SEY. However, this comes at the expense of inferior performance to both algorithms, especially if the channel matrix has more than one small eigenvalue. The result of this inferior performance is that although Brun’s algorithm increases the diversity, however, it does not achieve the full ML diversity, unlike LLL and SEY. Another drawback of this approach is the fact that since Brun’s algorithm is a precoding scheme, it requires the channel to be known at the transmitter, which is often difficult to achieve. These disadvantages are the main reasons for the lack of popularity of the Brun’s algorithm in the MIMO context.

### 2.7 Considerations for LR-aided detection

In order to use the LR technique, a few considerations must be taken into account. First, there is the need for shifting and scaling the received signal when using LR. Second, there is the requirement of boundary control on the detected symbols as a result of considering an infinite constellation in LR-aided detection. Finally, the complications of applying LR to the K-Best detector are discussed and a method, which has been developed in [41], to eliminate these complications is presented.
2.7.1 Shifting and Scaling

The original points in the constellation are required to consist of continuous integers in \( \mathbb{Z} + i\mathbb{Z} \) (the set of complex integers) before LR can be used. However, the real and imaginary parts of QAM constellations do not consist of continuous integers, thus, it is necessary to shift and scale the original constellation. Normally, a displacement vector \( \mathbf{d}_{N_T \times 1} = [1 + i, \cdots, 1 + i]^T \) is used to shift and scale the received signal \( \mathbf{y} \) as follows:

\[
\tilde{\mathbf{y}} = (\mathbf{y} + \mathbf{Hd})/2 = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}(\mathbf{x} + \mathbf{d})/2 + \mathbf{v}/2 = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}\tilde{\mathbf{x}} + \mathbf{v}/2, \quad (2.25)
\]

Hence, \( \tilde{\mathbf{y}} \) is hereafter used for detection purposes and the detected vector, i.e., \( \tilde{\mathbf{x}} \), is transformed to the original format via \( \mathbf{x} = 2\tilde{\mathbf{x}} - \mathbf{d} \).

2.7.2 Boundary Control

Lattice reduction treats the constellation as an infinite lattice which must be accounted for. Recall that under normal circumstances (i.e., without LR), the set \( \{\mathbf{Hs}\} \) represents a subset of \( \mathcal{L}(\mathbf{H}) \) and the transmitted vector \( \mathbf{s} \) can be considered as the coordinates of a lattice point. Since transmitted vector \( \mathbf{s} \) originates from a finite constellation, therefore for a given channel matrix \( \mathbf{H} \), the resulting lattice subset \( \{\mathbf{Hs}\} \) is finite. With LR, the transmitted vector is now \( \mathbf{x} = \mathbf{T}^{-1}\mathbf{s} \), which is not bound to a finite constellation. Therefore, LR-aided MIMO detection must consider the constellation as an infinite lattice. Note that this can lead to detecting symbols outside the constellation, which should be mapped back to the constellation after detection through boundary control. This mapping takes place by choosing the constellation point closest to the detected symbol.

2.7.3 Lattice Reduction with K-Best

The application of the LR technique to linear receivers as well as SIC receivers is straightforward, since for each of these methods, only the best candidate is selected at each level. Therefore once LR is applied, the LR-reduced version of the received vector is derived (i.e. \( \tilde{\mathbf{z}} = \tilde{\mathbf{Q}}^H\mathbf{y} \)) and the detection process proceeds as normal. On the other hand, applying LR to K-Best is not as simple. Recall that for the conventional K-Best algorithm in M-QAM systems, there are \( \sqrt{M} \) children per parent to be considered. Thus, the number of children per parent is known a priori. However, once the LR technique is applied, the new basis of the LR-reduced channel \( \tilde{\mathbf{H}} \) distorts the symmetry of the tree formed by the original
channel $\mathbf{H}$. The exact post-LR shape of the tree is unknown since it depends on the new basis of $\tilde{\mathbf{H}}$, which in turn depends on the LR parameters used as well as the correlation between the original basis of $\mathbf{H}$. Hence, the number of children as well as their values are no longer known in advance. As a result, the implementation of the conventional K-Best cannot be considered independent of the LR process. Moreover, this asymmetry necessitates modifying the K-Best algorithm and causes an increase in its complexity to such a degree that it nearly reaches the same complexity of the optimal but impractical ML detector.

Consequently, the application of LR to K-Best has, until recently, been considered impractical due to the distortion of the tree symmetry caused by LR. However, a novel K-Best algorithm developed in [41] was able to eliminate this joint LR/K-Best implementation challenge. This was achieved by introducing an efficient sorting and expansion scheme in which the best $K$ children are generated one-by-one and on-demand without visiting all the other nodes. Clearly, this on-demand version of K-Best is much more suitable for LR-aided detection, and hence it has been used in this thesis instead of the conventional K-Best algorithm. Note that the on-demand K-Best scheme produces the exact K-Best solution, therefore no performance loss is incurred as a result of adopting this LR-friendly scheme.
3 Hardware-Optimized Lattice Reduction Algorithm

3.1 Introduction

Emerging 4G wireless standards necessitate MIMO systems with high data rates, high mobility and large antenna configurations as discussed in Chapter 1. This leads to higher complexity MIMO detection schemes in order to accommodate the larger number of antennas and fast-varying channel environments without sacrificing the BER performance. Chapter 2 presented the lattice reduction technique, which has the potential of greatly increasing the performance and efficiency of MIMO systems through simultaneously lowering the complexity and increasing the BER performance of MIMO detectors (in high SNR regimes). Chapter 2 also presented the three most popular lattice reduction algorithms within the MIMO detection context, namely: the LLL algorithm, Seysen’s algorithm and Brun’s algorithm. However, in order to realize the potential of lattice reduction, none of these LR algorithms are suitable for hardware implementation in their original forms. There is a need to develop an LR algorithm optimized for hardware implementation while maintaining a high quality lattice reduction solution. This hardware-optimized algorithm should have low computational complexity, low number of iterations and be suitable for a high-throughput pipelined implementation.

3.2 Analysis of Existing Lattice Reduction Algorithms

3.2.1 Background

The first step before applying algorithm optimizations is to select the most computationally efficient LR algorithm out of LLL, SEY and Brun’s. Brun’s algorithm is not suitable for consideration for the demands of 4G wireless systems due to its many approximations that result in inferior BER performance as discussed in Chapter 2. Hence, this work
focuses its analysis on the LLL and SEY algorithms. The analysis consists of two parts, namely: complexity analysis and BER performance analysis. Below, a background of existing analysis work of the LR algorithms is presented, while highlighting areas which require further work (i.e. in order to make the correct LR algorithm choice for hardware implementation).

**Previous Complexity Analysis**

There is a debate in the LR literature as to which of LLL or SEY has lower complexity and which enables a better BER performance. The results of this debate have varied based on whether the real or complex version of LLL and SEY were used and which comparison criterion was chosen. Various comparison criteria have been chosen in the literature, ranging from the number of iterations, to the number of basis updates and even the number of column operations. In [37], Real-valued LLL (RLLL) was compared to complex-valued SEY and shown to have a higher average number of iterations but a lower per-iteration complexity. In [42], Complex-valued LLL (CLLL) is presented as having lower complexity than complex-valued SEY due to the higher per-iteration complexity of SEY as well as the processing overhead needed for calculating the pseudo inverse of $H$. On the other hand, [38] suggests that SEY has lower complexity than LLL since SEY is based on global-optimization\(^1\), while LLL focuses on local optimization\(^2\). The authors of [38] support their argument by showing that SEY has a lower average number of basis updates than LLL. However, the results in [38] are only presented for RLLL and real-valued SEY, with no mention made of how the complex-valued versions of the algorithms compare.

**Previous BER Performance Analysis**

LLL has been proven to achieve ML diversity in MIMO decoding [43]. Although this has yet to be proven mathematically for SEY, BER simulations in [37, 38, 42, 44] seem to indicate that SEY also achieves ML diversity. Despite displaying the same ML diversity, the gap of LLL and SEY to the ML BER curve differ based on the type of MIMO detector being employed for LR-aided detection. It has been shown that for linear detectors such as ZF and MMSE, SEY outperforms LLL. On the other hand, for SIC-based detectors such as V-BLAST, SEY and LLL have been shown to display similar BER performance. The

---

\(^1\)Since SEY considers all basis vectors in the lattice simultaneously and performs operations on those particular vectors that will reduce the lattice according to Seysen’s orthogonality measure.

\(^2\)Since in each LLL iteration only two adjacent basis vectors (i.e. matrix columns) may be exchanged.
comparison of LLL and SEY for near-optimal non-linear MIMO detectors such as K-Best has not yet, to the best of our knowledge, been published in the literature. This, in fact, is the most important BER performance comparison, since it is near-optimal detectors that are able to meet the stringent demands of emerging 4G wireless systems.

3.2.2 Proposed Analysis Framework

To determine which LR algorithm should be chosen for further optimizations and hardware implementation, there is a need for a comprehensive complexity and performance analysis of both LLL and SEY. To achieve this, the following analysis criteria are used:

- Definition of LR Iteration and Basis Update
- Number of Operations
- Algorithm Variations and Scaling

Note that this same analysis framework will also apply when analyzing the performance of our proposed hardware-optimized LR algorithm in Section 3.4.

Definition of LR Iteration and Basis Update

LLL and SEY are fundamentally different algorithms. Therefore, for a fair comparison, there is a need to clearly define what is considered an LR iteration and what is considered an LR basis update operation for each of LLL and SEY, independent of the underlying calculation method of each. In this regard, let’s consider the lattice reduction process, during which a series of partially reduced channel matrices $\tilde{H}_i$ are produced satisfying $\tilde{H}_i = H^T_i$, where $i$ represents the iteration number. Therefore, if the total number of iterations is $N_I$ then $\tilde{H}_0 = H$ and $\tilde{H}_{N_I} = \tilde{H}$. The LR algorithms are compared based on this definition of an iteration. As for the number of basis updates, $N_{BU}$, a basis update in LLL occurs only if the Lovász condition passes (i.e. $\{N_{BU}\}_{LLL} \leq \{N_I\}_{LLL}$), while for SEY, each iteration includes a basis update (i.e. $\{N_{BU}\}_{SEY} = \{N_I\}_{SEY}$).

Number of Operations

Previous research efforts in [38,42,44] comparing LLL and SEY have focused on the number of iterations and/or the number of basis updates. Although these criteria are very useful
Hardware-Optimized Lattice Reduction Algorithm

in judging the efficiency of each algorithm and estimating their respective run-times, they alone are not sufficient especially when targeting hardware implementation. For a more thorough analysis, it is necessary to compare LLL and SEY in terms of the number distinct real-valued operations, namely, addition, multiplication, division and square root. Where complex multiplication was taken to equal 2 real additions and 4 real multiplications while complex division (used in SEY) was taken equal to 3 real additions, 6 real multiplications and 2 real divisions. Moreover, these distinct real-valued operations are combined into a compact form of comparison through calculating the number of real Floating-Point Operations (FLOPS). Each addition and multiplication is considered equivalent to a single FLOP (i.e. add = mult = 1 FLOP), while division and square root operations are each considered equivalent to 8 FLOPS (i.e. div = sqrt = 8 FLOPS). The analysis of required number of FLOPS by an algorithm gives an indication of the total amount of computational work it requires, and hence it is an important measure in evaluating the computational efficiency of each LR algorithm.

Algorithm Variations and Scaling

To offer a balanced view of LLL and SEY, the complexity analysis is performed in several variations of each algorithm. In other words, this work analyzes both the real and complex version of each of LLL, as well as varying the values of \( \delta \) for LLL (specifically \( \delta \in \{ \frac{3}{4}, 1 \} \)) and employing both Greedy and Lazy versions of SEY. Moreover, to offer an indication of how each algorithm scales with the number of antennas, a complexity comparison for both 4 × 4 and 8 × 8 MIMO systems is performed.

3.2.3 Complexity Analysis

Table 3.1 compares the complexity of R/CLLL and Greedy/Lazy SEY. It should be noted that in this complexity analysis, the LR iterations were not limited for the purpose of accurately estimating the pure algorithmic complexity. In addition, for the LLL-based algorithms, a sorted QR decomposition was used, instead of traditional QR, as it has been shown to reduce the LLL complexity [45]. For the purposes of these simulations, pseudo-random \( H \) channel matrices were generated with entries drawn from the standard normal distribution (i.e. the entries were chosen independently as zero-mean complex Gaussian random variables with variance one per complex dimension). These generated channel matrices were then supplied as inputs to the LR algorithms (for LLL-based algorithms,
the QR decomposition of $\mathbf{H}$ was the input). It should also be noted that the complexity of the LR algorithms is independent of each of the QAM constellation size $M$, the SNR and the noise (e.g. AWGN), since the correlation between the columns of the input $\mathbf{H}$ does not depend on these system parameters\(^3\).

From the results in Table 3.1, CLLL with $\delta = \frac{3}{4}$ clearly offers the lowest complexity as it is significantly more efficient than RLLL and SEY. CLLL offers a 44% and 81% reduction in the FLOP count compared to RLLL and SEY, respectively. From Table 3.1, it is observed that although Lazy SEY has lower per-iteration complexity than Greedy SEY, it does require nearly double the number of iterations due to its suboptimal index selection. This leads to a higher FLOP count for Lazy SEY.

For VLSI implementation, it is also important to consider the distribution of $N_I$, since in practice the number of iterations cannot be unlimited. The cumulative density functions ($CDF$) of $N_I$ for the LR algorithms, for $4 \times 4$ and $8 \times 8$ MIMO, are shown in Fig. 3.1 and Fig. 3.2, respectively. The simulation results show that the $CDF(N_I)$s of SEY and CLLL are very similar and both execute in fewer iterations and with less variability than either RLLL or Lazy SEY. This will result in a better performance for CLLL and Greedy SEY in a practical implementation where the iterations are limited. In addition to analyzing the distribution of $N_I$, it is also important to examine the behaviour of the number of basis updates $N_{BU}$. The basis update process is one of the most computationally intensive operations in both LLL and SEY and therefore the hardware complexity is greatly affected by the required $N_{BU}$. In this regard, the ($CDF$) of $N_{BU}$, for both $4 \times 4$ and $8 \times 8$ MIMO, are shown in Fig. 3.3 and Fig. 3.4, respectively. Again, CLLL and Greedy SEY display the most desirable behaviour with a lower average and less variable $N_{BU}$ compared to the other LR algorithms.

\(^3\)Note that these system parameters become of large significance when the BER performance of LR-aided MIMO detection is considered.
Table 3.1: The complexity of LLL and SEY LR Algorithms for $4 \times 4$ and $8 \times 8$ MIMO systems.

<table>
<thead>
<tr>
<th>LR Algorithm</th>
<th>$4 \times 4$</th>
<th>$8 \times 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iters</td>
<td>BUs</td>
</tr>
<tr>
<td>RLLL $\delta = 1$</td>
<td>34.7</td>
<td>14.6</td>
</tr>
<tr>
<td>RLLL $\delta = 0.75$</td>
<td>18.6</td>
<td>6.1</td>
</tr>
<tr>
<td>CLLL $\delta = 1$</td>
<td>7.7</td>
<td>2.9</td>
</tr>
<tr>
<td>CLLL $\delta = 0.75$</td>
<td>5.6</td>
<td>1.5</td>
</tr>
<tr>
<td>SEY Greedy</td>
<td>5.5</td>
<td>5.5</td>
</tr>
<tr>
<td>SEY Lazy</td>
<td>9.3</td>
<td>9.3</td>
</tr>
</tbody>
</table>

$^1$ FLOPS = Add + Mult + $(8 \times \text{Div}) + (8 \times \text{Sqrt})$
Figure 3.1: CDF of iterations for R/CLLL (\( \delta \in \{\frac{3}{4}, 1\}\)) and SEY (Greedy/Lazy) for a 4x4 MIMO system

Figure 3.2: CDF of iterations for R/CLLL (\( \delta \in \{\frac{3}{4}, 1\}\)) and SEY (Greedy/Lazy) for a 8x8 MIMO system
Figure 3.3: CDF of basis updates for R/CLLL ($\delta \in \{\frac{3}{4}, 1\}$) and SEY (Greedy/Lazy) for a 4x4 MIMO system.

Figure 3.4: CDF of basis updates for R/CLLL ($\delta \in \{\frac{3}{4}, 1\}$) and SEY (Greedy/Lazy) for a 8x8 MIMO system.
3.2.4 Correlation and BER Performance Analysis

As discussed earlier, the purpose of lattice reduction is to lower the correlation between the basis vectors of the channel matrix, with the aim of improving the detection performance. To compare LLL and SEY in terms of their impact on lowering the correlation, the channel condition number \( \kappa(H) \) is used as a measure of correlation in the channel matrix \( H \). The measure \( \kappa(H) \) is defined as the ratio between the largest and smallest singular values of \( H \) as follows [46]:

\[
\kappa(H) = \frac{\sigma_1}{\sigma_{N_T}}, \tag{3.1}
\]

where \( \sigma_1 \geq \cdots \geq \sigma_{N_T} \) are the singular values of \( H \) [47]. Based on this measure of correlation, Table 3.2 below shows the impact of LLL (real and complex) and SEY (Greedy and Lazy) on reducing correlation compared to the initial pre-LR correlation in \( H \). The above results, which were calculated based on the average of 100,000 input channel matrices and with the LR iterations left unlimited, show that both LLL and SEY are able to significantly lower the channel condition number (which is used as a measure of correlation) by approximately 74\% and 72\% for 4 × 4 and 8 × 8 channel matrices, respectively. It was also observed that the best result is achieved by Greedy SEY, which corresponds with previous research results that show Greedy SEY outperforming LLL for linear detectors.

Table 3.2: Post-LR Correlation Comparison for LLL and SEY LR Algorithms for 4 × 4 and 8 × 8 MIMO systems.

<table>
<thead>
<tr>
<th>( \kappa(H_{4\times4}) )</th>
<th>Pre-LR</th>
<th>CLLL ( \delta = 1 )</th>
<th>CLLL ( \delta = 0.75 )</th>
<th>RLLL ( \delta = 1 )</th>
<th>RLLL ( \delta = 0.75 )</th>
<th>SEY GREEDY</th>
<th>SEY LAZY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-LR</td>
<td>10.85</td>
<td>2.77</td>
<td>2.82</td>
<td>2.94</td>
<td>2.98</td>
<td>2.70</td>
<td>2.71</td>
</tr>
<tr>
<td>( \kappa(H_{8\times8}) )</td>
<td>24.20</td>
<td>6.65</td>
<td>6.93</td>
<td>8.04</td>
<td>7.80</td>
<td>5.58</td>
<td>5.60</td>
</tr>
</tbody>
</table>

As mentioned earlier, the targeted MIMO detector type is the near-optimal detector, specifically the K-Best detector. Therefore, despite LLL showing slightly higher correlation measurements than SEY, higher precedence is given to the impact on improving the BER performance when applied to the K-Best detector, as this is the real objective of LR-aided detection. To this end, this work presents the BER performance analysis with the near-optimal K-Best detector, specifically with the real version of K-Best (denoted as KBR for K-Best Real). The simulation results are for a single-carrier 4 × 4 MIMO system and are shown for both 64-QAM and 256-QAM schemes in Fig. 3.5 and Fig. 3.6, respectively.
Figure 3.5: BER Performance of LR-aided KBR (K-Best Real) using LLL and SEY - for 4 × 4 MIMO System with 64-QAM and K=10

Figure 3.6: BER Performance of LR-aided KBR (K-Best Real) using LLL and SEY - for 4 × 4 MIMO System with 256-QAM and K=15
Fig. 3.5 shows the BER performance of a 64-QAM scheme for ML, K-Best, and LR-aided K-Best for $K = 10$. The LR-aided K-Best is shown with both LLL (real and complex with $\delta = \frac{3}{4}$) and SEY (Greedy and Lazy). Fig. 3.5 shows that for a fixed value of $K = 10$, the LR technique can greatly improve the BER performance of K-Best in high-SNR regimes and in fact, it achieves the same diversity as ML. In addition, there is no distinct difference in performance between LLL and SEY when used with K-Best. This is true provided that the stop condition (Equations 2.16, 2.17 and 2.20) for each LR algorithm has been reached (i.e. the iterations are not limited to a certain fixed number). Similar results are seen with a 256-QAM scheme as shown in Fig. 3.6.

One important observation is that LR-aided K-Best detection suffers from a BER performance loss at low-SNR values as seen in Fig. 3.5 and Fig. 3.6. This performance loss is due to the boundary control operation in the K-Best algorithm [41], however, this is not problematic since in a practical implementation LR would only apply to K-Best when high SNR ranges are under consideration (i.e. only for SNR values greater than a predetermined threshold value). For example, in 64-QAM and 256-QAM, LR will be applied for SNR values greater than 31 dB and 38 dB, respectively.

### 3.2.5 Impact of LR on lowering MIMO detection complexity

In addition to lowering the channel correlation and improving the BER performance, another important advantage of LR-aided K-Best is that, as shown in Fig. 3.5, the $K$ value can be decreased while maintaining ML diversity and without significantly degrading the near-optimal performance. For instance, $K = 2$ has equivalent performance as $K = 10$, and for $K = 1$ there is only a small performance loss. This is critical for VLSI implementations, since lower $K$ values result in a significantly lower-complexity implementation of K-Best.

Fig. 3.6 shows the BER performance of the K-Best/LR technique for a $4 \times 4$ 256-QAM system. Again, the LR-aided K-Best detection significantly outperforms the traditional K-Best algorithm and also allows for very low $K$ values (e.g. $K=1,2$) while maintaining near-ML performance and ML diversity.

Table 3.3 below illustrates the effect of LR on lowering the complexity of the K-Best algorithm. The lower $K$ value, made possible by LR, results in a large reduction in both the number of expanded children in the search process as well as the number of sorting cycles
3 Hardware-Optimized Lattice Reduction Algorithm

Table 3.3: Comparison between different $4 \times 4$ K-Best Implementations

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Metric</th>
<th>$[48] \ K={5,10,15}$</th>
<th>$[41] \ K={5,10,15}$</th>
<th>$[41] \ with \ LR \ K={2,3,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-QAM</td>
<td>expand</td>
<td>120</td>
<td>67</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>sort</td>
<td>110</td>
<td>35</td>
<td>14</td>
</tr>
<tr>
<td>64-QAM</td>
<td>expand</td>
<td>384</td>
<td>141</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>sort</td>
<td>372</td>
<td>70</td>
<td>21</td>
</tr>
<tr>
<td>256-QAM</td>
<td>expand</td>
<td>790</td>
<td>219</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>sort</td>
<td>773</td>
<td>105</td>
<td>28</td>
</tr>
</tbody>
</table>

1 Values of $K$ for 16-QAM, 64-QAM and 256-QAM, respectively.

required. The complexity savings are more noticeable for higher order modulation schemes with savings of 60%, 70% and 73% for 16-QAM, 64-QAM and 256-QAM, respectively.

3.3 Lattice Reduction Implementation Challenges

In the analysis above, it was shown that CLLL and Greedy SEY offer equivalent BER performance enhancements for K-Best and that both LR algorithms have better iteration-limited performance than RLLL and Lazy SEY. However, it was also shown that CLLL has lower complexity than Greedy SEY when compared in terms of the average number of operations. Based on these results, CLLL was chosen as the base algorithm that this work seeks to optimize for a hardware implementation and then use this optimized LR algorithm to design the LR ASIC core. Note that, despite being the most promising LR algorithm, CLLL is still a computationally intensive algorithm that is not directly suitable for hardware implementation, hence the need for optimizations. The challenges in implementing CLLL are briefly highlighted below, while the solutions for them are discussed in the following section where our proposed LR algorithm is presented.

1. One of the most computationally intensive parts of CLLL is the calculation of the $\mu$ factor used in the size reduction step (line 5 in Table 2.2). In both implementations in [16] and [18], a divider was used for the calculation.
2. Direct implementation of the Lovász condition used in CLLL (line 9 in Table 2.2) requires significant computation resources since it involves squaring, addition, multiplication and comparison. To reduce the complexity, the square root of both sides can be taken to remove the squaring operation on the left side. However, this introduces a square root operation on the right side, which can be solved by realizing that it is equivalent to a norm of a 3-D vector which can be calculated using 3-D HouseHolder Rotations [16]. However, this method is still computationally intensive and remains a bottleneck in the CLLL algorithm.

3. In the CLLL algorithm, if in a given iteration $k$ the Lovász condition holds true, then the $k$th and $(k-1)$th columns are swapped in $\tilde{R}$ and $T$. To restore the upper triangular nature of $\tilde{R}$, a Givens rotation operation must be performed (line 11-12 in Table 2.2). Consequently, a Givens rotation operation is also applied to $\tilde{Q}$ to maintain the $\tilde{H} = \tilde{Q}\tilde{R}$ relationship. In the VLSI implementation in [16], the Givens rotation operations were performed using an array of 8x8 real multipliers, while in [18] two complex multipliers combined with two real-complex multipliers were used. Another computationally intensive operation is the calculation of the $\alpha$ and $\beta$ factors needed for the Givens rotations. In [18], this was performed using a divider and a square root block. However, in [16], $\alpha$ and $\beta$ were calculated using 3D inverse HouseHolder CORDIC.

4. The CLLL algorithm suffers from variable execution time and complexity since the number and sequence of iterations depends on the input $R$ matrix, and thus depends on the varying correlation of the channel matrix $H$. This variability increases the complexity of any hardware implementation of CLLL and prevents a high-throughput pipelined design from being realized.
3.4 Proposed Lattice Reduction Algorithm

Our proposed algorithm, herein-after referred to as hardware-optimized LLL (HOLLL), introduces improvements to the CLLL algorithm, while maintaining its performance, to reduce its complexity, minimize its delay and make it more suitable for a VLSI implementation. Our proposed algorithm, HOLLL, for efficient lattice reduction is shown in Table 3.4 below, and the list of improvements follow afterwards:

Table 3.4: Hardware Optimized LLL Algorithm (HOLLL)

\[(\tilde{R}, \tilde{Z}, T) = \text{HOLLL}(R, Z, \delta)\]

1) \(\tilde{R} = R; \ T = I_{NT \times NT};\ stop = \text{FALSE};\)
2) while \(stop = \text{FALSE}\)
3) \(k = 2; \ stop = \text{TRUE};\)
4) while \(k \leq NT\)
5) for \(l = k - 1 : -1 : 1\)
6) \(\mu_q = \text{QUANTIZE}(\tilde{R}_{l,k}/\tilde{R}_{l,l}, [0, \pm 1, \pm 2]);\)
7) \(\tilde{R}(1:l,k) = \tilde{R}(1:l,k) - \mu_q \cdot \tilde{R}(1:l,l);\)
8) \(T(:,k) = T(:,k) - \mu_q \cdot T(:,l);\)
9) end
10) if \(\delta \cdot |\tilde{R}_{k-1,k-1}| > |\tilde{R}_{k,k}|\)
11) Swap \((k-1)\text{th and } k\text{th columns in } \tilde{R} \text{ and } T;\)
12) Update \(\tilde{R} \text{ and } \tilde{Z} \text{ using 2D CORDICs;}\)
13) \(stop = \text{FALSE};\)
14) end
15) \(k = k + 1;\)
16) end
17) end

3.4.1 MU Quantization

Fig. 3.7 below shows the dynamic range of \(\mu\), which clearly is very limited. In fact, over 99.9% of the values of \(\mu\) lie within \([0, \pm 1, \pm 2]\) for CLLL for both \(\delta = \frac{3}{4}\) and \(\delta = 1\). In the proposed algorithm \(\mu_q\) is quantized to these values, and hence eliminating the division and rounding operations. The procedure for calculating \(\mu_q\) is shown in Table 3.5.
Figure 3.7: PDF of $\Re(\mu)$ and $\Im(\mu)$ for CLLL with $\delta \in \{\frac{3}{4}, 1\}$ for 4x4 MIMO

Table 3.5: Quantization of $\mu$

\begin{itemize}
\item 1) $\mu_r = 0, \mu_i = 0$;
\item 2) if $\left(0.5 \cdot |\tilde{R}_{l,l}| \leq |\Re(\tilde{R}_{l,k})| \leq 1.5 \cdot |\tilde{R}_{l,l}|\right)$ \{$\mu_r = 1$\};
\item 3) else if $\left(|\Re(\tilde{R}_{l,k})| \geq 1.5 \cdot |\tilde{R}_{l,l}|\right)$ \{$\mu_r = 2$\};
\item 4) if $\left(0.5 \cdot |\tilde{R}_{l,l}| \leq |\Im(\tilde{R}_{l,k})| \leq 1.5 \cdot |\tilde{R}_{l,l}|\right)$ \{$\mu_i = 1$\};
\item 5) else if $\left(|\Im(\tilde{R}_{l,k})| \geq 1.5 \cdot |\tilde{R}_{l,l}|\right)$ \{$\mu_i = 2$\};
\item 6) $\mu_q = \text{sgn}\left(\frac{\Re(\tilde{R}_{l,k})}{\tilde{R}_{l,l}}\right) \cdot \mu_r + i \left(\text{sgn}\left(\frac{\Im(\tilde{R}_{l,k})}{\tilde{R}_{l,l}}\right) \cdot \mu_i\right)$;
\end{itemize}
As shown in Table 3.5, $\mu_q$ can be calculated by a series of simple comparisons. The multiplications by 0.5 and 1.5 can be easily implemented using a bit-shift operation and an adder. Furthermore, this quantization of $\mu$ greatly simplifies the subsequent size reduction operations on lines (6) and (7) of Table 2.2, which are expanded below into real and imaginary component calculations for clarification.

$$
\Re(\tilde{R}(1 : l, k)) = \Re(\tilde{R}(1 : l, k)) - \mu_r \cdot \Re(\tilde{R}(1 : l, l)) - \mu_i \cdot \Im(\tilde{R}(1 : l, l))
$$

$$
\Im(\tilde{R}(1 : l, k)) = \Im(\tilde{R}(1 : l, k)) - \mu_i \cdot \Re(\tilde{R}(1 : l, l)) + \mu_r \cdot \Im(\tilde{R}(1 : l, l))
$$

$$
\Re(T(:,k)) = \Re(T(:,k)) - \mu_r \cdot \Re(T(:,l)) - \mu_i \cdot \Im(T(:,l))
$$

$$
\Im(T(:,k)) = \Im(T(:,k)) - \mu_i \cdot \Re(T(:,l)) + \mu_r \cdot \Im(T(:,l))
$$

The multiplications in the above equations are now reduced to simple bit-shift operations. In [16], real multipliers were used to implement size reductions, while in [18], complex multipliers were used. On the other hand, a VLSI implementation of the algorithm presented in this work, only requires simple bit-shift operations and adders. Therefore, it is clear that by adopting the quantized $\mu$ approach, both the need for multipliers and dividers has been eliminated, and thus leading to a significant speed up and reduction in complexity.

It should be noted that [16] is in fact the first LR design to suggest quantizing $\mu$ instead of performing division and rounding operations. However, the authors proceed to state that the resulting collection of comparators and tree logic to perform the required comparison becomes prohibitive for large values of $\mu$, and as a result the authors continue with the division and rounding approach. In our simulations we have shown that it is sufficient to only consider the range of $\{0, \pm 1, \pm 2\}$ for $\mu$, resulting in simplified quantization logic.

### 3.4.2 Replacing Lovász Condition with Siegel Condition

Based on the LLL improvement idea in [49], this work proposes replacing the Lovász condition with the much simpler Siegel condition (line (10) in Table 3.4). The Siegel condition can be easily implemented in hardware using shift and comparison operations. The LR implementations in [18] and [19] also used the simpler Siegel condition, however, the squared version was used. In the proposed algorithm in this thesis, the square root of both sides is taken to avoid extra calculations. In addition, instead of setting $\delta$ to $(1/\sqrt{2})$, it is kept at $(1/2)$ to increase the LR-reduction quality.
3.4.3 Replacing $\tilde{Q}$ transformation with $\tilde{Z}$ transformation

The CLLL algorithm computes each of $\tilde{Q}$ and $\tilde{R}$. However, the MIMO detection schemes which require QR decomposition such as V-BLAST and K-BEST do not use the $\tilde{Q}$ matrix directly. Equation (2.11) shows that $\tilde{z} = \tilde{Q}^H y$ must be computed first before the detection can take place. Therefore, an algorithm which performs operations on $\tilde{z}$ directly would be more efficient. This is the approach taken in the proposed algorithm. Line (13) in Table (2.2) is replaced with $\tilde{Z}_{k-1:k,1:CU} = \Theta \tilde{Z}_{k-1:k,1:CU}$ where $CU$ is the channel use number (i.e. the number of received vectors for which the channel matrix $H$ is assumed to remain constant). In the targeted application it is assumed that the channel is valid over four consecutive received symbol vectors, therefore $CU = 4$ is used. The matrix $\tilde{Z}$ refers to the matrix which includes all four $\tilde{z}$ vectors, i.e. $\tilde{Z}_{N_r \times CU} = [\tilde{z}_1 \tilde{z}_2 \cdots \tilde{z}_{CU}]$. Note that the aforementioned equation is not present in Table 3.4 since the complexity of the $\tilde{Z}$ (and $\tilde{R}$) calculations has been further reduced as shown in the following section.

3.4.4 Replacing Multiplications with 2D CORDICs

The proposed algorithm resolves the complexity issue of both $(\alpha, \beta)$ calculations and Givens rotations multiplications (lines 11-13 in Table 2.2). The chosen approach relies entirely on 2D CORDIC vectoring and rotation operations, which greatly simplifies the calculations and allows for parallel operation. CORDIC (COordinate Rotation Digital Computer) algorithms are hardware efficient algorithms that can be used to calculate a variety of trigonometric functions while using only shift and add operations [50]. The use of CORDIC in the LR algorithm is for 2D vectoring (i.e. nullification) and rotation operations. Given a 2D vector with components $x$ and $y$ along the x-axis and y-axis, the CORDIC equations to align the vector with the x-axis (i.e. the vectoring mode) are

\[
\begin{align*}
x_{i+1} &= x_i - y_i \cdot d_i \cdot 2^{-i}, \\
y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i}, \\
\phi_{i+1} &= \phi_i - d_i \cdot \tan^{-1}(2^{-i}),
\end{align*}
\]

where the direction decision $d_i$ is based on the sign of the residual $y_i$ component, such that $d_i = +1$ if $y_i < 0$, otherwise $d_i = -1$. $\phi$ represents the accumulated angle through the vectoring process and is initialized to zero. Note that the CORDIC equations to rotate the 2D vector by a given angle (i.e. rotation mode) are the same exact equations used
for the vectoring mode except that goal is to nullify $\phi$ instead of $y$. This is achieved by having the direction decision in each iteration based on the residual $\phi$ component (i.e. the residual angle). In this case, $d_i = -1$ if $\phi_i < 0$, otherwise $d_i = +1$. Note that in rotation mode, the initial value $\phi_0$ is the desired rotation angle.

To illustrate the adopted basis update approach using 2D CORDIC operations, Fig. 3.8 above shows an example in which the 1st and 2nd columns of $\tilde{R}$ have been swapped but no Givens rotations have occurred yet. Note, that the $\tilde{R}$ matrix is shown in its real-valued decomposition (RVD) form based on the approach presented in Chapter 2. The $\tilde{R}$ matrix is also augmented with four $\tilde{z}$ vectors ($CU = 4$). The algorithm uses a special sequence of rotations adopted from [51]. This optimized sequence only requires operations on the odd columns of the RVD matrix, instead of the traditional operations on both even and odd columns. As a result, this sequence reduces the number of required vectoring and rotation operations by nearly half. The sequence is performed as follows:

1. $[\tilde{R}_{1,2}^{(R)} \cdot \tilde{R}_{1,2}^{(3)}]$ is vectored, thus zeroing $\tilde{R}_{1,2}^{(3)}$. The angle resulting from this nullification can be used to rotate the rest of rows 1 and 2 of the augmented RVD matrix. No computations are necessary for even columns as they follow directly from the rotations in the odd columns. This results in 7 rotation operations (3 for $\tilde{R}$ and 4 for $\tilde{Z}$) following the initial vectoring operation.

2. The updated $[\tilde{R}_{1,2}^{(R)} \cdot \tilde{R}_{2,2}^{(R)}]$ is vectored, and both row sets (1 and 3 for real-valued entries) and (2 and 4 for imaginary valued entries) are rotated. Instead of calculating
rotations on the even columns, the same values can be obtained by applying the resulting rotation angle to the odd columns of rows 2 and 4. Therefore, there are 3 rotations for real-valued entries of $\tilde{R}$ and 3 for the imaginary ones. Similarly, there are 4 rotations for real-valued entries of $\tilde{Z}$ and 4 for the imaginary ones. Hence, a total of 14 rotations operation follow the vectoring operation.

3. The updated $[\tilde{R}_{2,1}^{(R)} \tilde{R}_{2,1}^{(S)}]$ is vectored, and the rest of rows 3 and 4 are rotated with the resulting rotation angle. The number of rotations is 6 (2 for $\tilde{R}$ and 4 for $\tilde{Z}$).

By following steps 1-3 above, both the $\tilde{R}$ and $\tilde{Z}$ matrices are fully updated, while maintaining RVD symmetry. This results in a total of 3 vectoring operations and 27 (=7+14+6) rotations operations. Extending the example to swapping cases other than the case of $k = 2$ (i.e. swapping columns 1 and 2) is straightforward. Each of the cases $k = 3$ and $k = 4$ include three vectoring operations similar to the case of $k = 2$. The total number of rotations for $k = 3$ and $k = 4$ is easily found to be 23 (=6+12+5) and 19 (=5+10+4), respectively.

To reduce the complexity even further, the $\phi_i$ rotation angles are not computed but rather an implicit angle transfer scheme is adopted by passing the direction decision vector $[d_0 \cdots d_n]$ from the vectoring operation to the subsequent rotation operations as in [52]. In addition, all the required rotations, after each vectoring, can be performed in parallel leading to a high speed implementation. Finally, it should be noted that 2D CORDIC operations incur a gain that approaches 1.647 as the number of iterations goes to infinity. Ideally this should be reversed by applying a 0.6073 gain, however to simplify the design of this gain and avoid the need for a constant multiplier, this factor is approximated as (0.625 = 0.5 + 0.125), which can be implemented using two shift operations and an addition. BER simulations show that the performance loss due to this approximation is negligible.

3.4.5 Applying Fixed Complexity Technique

The problem of non-deterministic complexity of R/CLLL was solved in [53], in which a fixed complexity CLLL (fcCLLL) algorithm was developed. In this work, the same approach is adopted, as shown in Table 3.4, where a pre-determined sequence of $k$ values is traversed, as opposed to the original CLLL where the sequence of $k$ depends on the input $R$. Another advantage of this approach is that fcCLLL outperforms CLLL when the
number of iterations are limited as shown in [53]. We believe that this due to the averaging of the LR application over all the columns of $\mathbf{R}$ which is important when the iterations are limited. For example, if the iterations are limited to 9, then fcCLLL (and also the proposed algorithm) would apply LR operations to all the columns of $\mathbf{R}$ more than once. However in CLLL, it is possible that most of the iterations focus on the first columns and only reach the final column once or not at all. If the last column has high correlation with previous columns and thus requires reduction and swapping, then the LR reduction quality will be degraded. Another advantage of predetermining $k$ is that the LR iterations can be implemented independently and used in a pipelined high-throughput fashion, as opposed to the high-delay iterative approach in which a single architecture is used for all iterations, which has been the previous approach, e.g. [16] and [18]. Note that [53] shows that fcCLLL has a higher complexity than CLLL, however this is not true for the proposed HOLLL algorithm as a result of the many other improvements that have been included in HOLLL to reduce the complexity.

### 3.4.6 Parameter Selection for Proposed LR Algorithm

Based on extensive simulations, the following parameters were selected for the proposed algorithm (HOLLL) to optimize the tradeoff between performance and complexity. The complexity analysis and BER simulations in the next section were performed using these parameters. These same design parameters were also used for the hardware implementation of HOLLL presented in Chapter 4.

**LR Iteration Limit** As a result of implementing the fixed complexity method in [53], it is most efficient to implement the LR iterations as multiples of 3 such that complete passes through the $\tilde{\mathbf{R}}$ matrix are performed. Based on this, HOLLL was simulated with the iterations limited to each of $(3, 6, 9, 12)$, and a limit of 9 was selected since it achieved the best trade-off between performance and complexity. More iterations added very little to the performance, while less iterations significantly degraded the LR reduction quality.

**MU Quantization** In this algorithm, the quantization of $\mu$ has been limited to 0, $\pm 1, \pm 2$. Further quantization to $\pm 3$ and greater did not show a noticeable BER performance improvement.
**Siegel Factor** A value of $\delta = \frac{1}{2}$ for the Siegel swapping criterion was shown to produce high-quality LR-reduced results while maintaining fast execution.

**CORDIC Iterations** After selecting an LR iteration limit of 9, the HOLLL algorithm was simulated for various CORDIC iteration values to evaluate their effect on the BER performance. The simulation, shown in Fig. 3.9 below, indicates that a CORDIC iteration count of 9 has a minimal performance loss compared to performing the ideal case of matrix multiplication. The implementation cost of adding more CORDIC iterations outweighs any small additional performance gain, while having less CORDIC iterations degrades the diversity.

![Figure 3.9: BER Performance of LR-aided KBR (K-Best Real) with Complex LR vs ML - for a 4 × 4 MIMO System with 64-QAM](image)
3.5 Complexity Analysis and Performance Comparison

Table 3.6 below compares the complexity of the proposed HOLLL algorithm to CLLL and Greedy SEY for a $4 \times 4$ MIMO system. The algorithms are compared in terms of the average number of iterations (Iters) and basis updates (BUs) as well as the average number of real-valued operations (addition, multiplication, division and square root). Clearly, HOLLL has a much lower complexity than the other LR algorithms as a result of not requiring multiplication, division, or square root operations. Furthermore, HOLLL displays a much lower average number of iterations and basis updates compared to the other LR algorithms. By comparing the FLOP count, it can be seen that HOLLL results in nearly 70% fewer FLOPS than CLLL with $\delta = \frac{3}{4}$ and 94% less FLOPS than Greedy SEY. Moreover, since the FLOP count for HOLLL only consists of additions (i.e. no multiplications) then it is clear that this will lead to an even greater reduction in complexity of an implemented architecture compared to the CLLL and SEY which include multiplications in their FLOP count.

<table>
<thead>
<tr>
<th>LR Algorithm</th>
<th>Iters</th>
<th>BUs</th>
<th>Add</th>
<th>Mult</th>
<th>Div</th>
<th>Sqrt</th>
<th>FLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLLL ($\delta = 1$)</td>
<td>7.8</td>
<td>2.9</td>
<td>577.3</td>
<td>699.7</td>
<td>39.6</td>
<td>2.9</td>
<td>1617.0</td>
</tr>
<tr>
<td>CLLL ($\delta = 0.75$)</td>
<td>5.6</td>
<td>1.5</td>
<td>451.6</td>
<td>546.8</td>
<td>27.0</td>
<td>1.5</td>
<td>1227.8</td>
</tr>
<tr>
<td>SEY Greedy</td>
<td>5.5</td>
<td>5.5</td>
<td>1680.5</td>
<td>2400.3</td>
<td>284.0</td>
<td>0</td>
<td>6353.0</td>
</tr>
<tr>
<td>This Work (HOLLL)</td>
<td>3.8</td>
<td>0.3</td>
<td><strong>380.6</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>380.6</strong></td>
</tr>
</tbody>
</table>

$^1$FLOPS = Add + Mult + (8 x Div) + (8 x Sqrt)

The cumulative density functions (CDF) of the number of iterations for the LR algorithms are shown in Fig. 3.10 below. It is clear that HOLLL LR-reduces a larger percentage of input matrices at lower iterations than the other LR algorithms. This translates into a superior performance for the proposed algorithm in a practical VLSI implementation with limited iterations. Fig. 3.10 shows the CDF of the number of basis updates for the LR algorithms, again HOLLL displays the most desirable behaviour with a lower average and less variable $N_{BU}$ compared to the other LR algorithms.
3 Hardware-Optimized Lattice Reduction Algorithm

Figure 3.10: CDF of iterations for Complex LR algorithms (CLLL, SEY and this work (HOLLL)) for a $4 \times 4$ MIMO System

Figure 3.11: CDF of basis updates for Complex LR algorithms (CLLL, SEY and this work (HOLLL)) for a $4 \times 4$ MIMO System
The impact of the proposed HOLLL algorithm on reducing channel matrix correlation compared to other CLLL and SEY is shown in Table 3.7 below. It can be observed that given unlimited iterations, HOLLL does not achieve the low correlation offered by CLLL and SEY. However, HOLLL is specifically optimized for a practical iteration-limited scenario, and given an iteration limit of 9, HOLLL clearly produces a low correlation similar to that of CLLL and SEY, but at a much lower complexity as shown earlier.

Table 3.7: Post-LR Correlation Comparison for Complex LR Algorithms (CLLL, Greedy SEY and This Work (HOLLL)) for 4 × 4 and 8 × 8 MIMO systems.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Pre-LR</th>
<th>CLLL (δ = 0.75)</th>
<th>Greedy SEY</th>
<th>This Work (HOLLL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IterLim</td>
<td>N/A</td>
<td>∞</td>
<td>9</td>
<td>∞</td>
</tr>
<tr>
<td>κ((H_{4x4}))</td>
<td>10.85</td>
<td>2.82</td>
<td>3.06</td>
<td>2.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.10</td>
</tr>
<tr>
<td>κ((H_{8x8}))</td>
<td>24.18</td>
<td>6.92</td>
<td>11.85</td>
<td>5.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.53</td>
</tr>
</tbody>
</table>

In the following figures, the simulation results for a single-carrier 4 × 4 MIMO system are presented for a 64-QAM scheme. Fig. 3.12 shows the BER performance of the proposed HOLLL algorithm compared to CLLL for the low-complexity detectors ZF and V-BLAST. Note that since ZF does not involve a QR decomposition, HOLLL was modified to calculate \(\tilde{H}\) directly. This is done through the same exact operations performed on \(T\) in lines (8) and (11) of Table 3.4. As shown in the figure, despite the limited iterations of HOLLL (here it is set to 9), its performance is very close to the ideal CLLL algorithm with unlimited iterations. HOLLL clearly outperforms CLLL with an equal iteration limit of 9. Fig. 3.13 compares the BER performance of HOLLL, CLLL and SEY when applied to the near-ML K-Best detection scheme for \(K = 10\). With an iteration limit of 9, HOLLL outperforms CLLL despite the much lower complexity of our algorithm.
Figure 3.12: BER Performance of LR-aided ZF & V-BLAST with Complex LR algorithms (CLLL and this work (HOLLL)) - for 4 × 4 MIMO with 64-QAM

Figure 3.13: BER Performance of LR-aided KBR (K-Best Real, K=10) with Complex LR algorithms (CLLL and this work (HOLLL)) - for 4 × 4 MIMO with 64-QAM
3.6 Summary

In this chapter, a thorough analysis of existing LR algorithms was presented. Each of the LLL (Real and Complex) and SEY (Greedy and Lazy) algorithms were analyzed and compared. The complexity and performance results have shown that CLLL is the most computationally efficient algorithm with the lowest number of real-valued operations and a desirable iteration-limited performance. Based on this, CLLL was chosen as the algorithm to which optimizations were applied to achieve a low-complexity high-speed algorithm without sacrificing the lattice reduction quality. The implementation challenges associated with CLLL were then presented and a hardware-optimized LLL (HOLLL) algorithm was proposed to address these challenges. The proposed algorithm completely eliminates the need for multipliers and dividers, as it relies only on add, shift and comparison operations. As a result, HOLLL displays very low complexity, 70% lower than CLLL. Furthermore, HOLLL shows excellent iteration-limited performance, outperforming CLLL given an iteration-limit of 9. Based on these improvements, the proposed HOLLL algorithm is considered to be more suitable for hardware implementation than both CLLL and SEY. In the next chapter, the VLSI architecture and ASIC implementation of HOLLL is presented along with the measured chip results.
4 VLSI Implementation of Hardware-Optimized Lattice Reduction

4.1 Introduction

Lattice reduction (LR) has emerged recently as a technique that has the potential to both increase the performance of MIMO detectors to achieve ML diversity, while simultaneously lowering their complexity. Several LR algorithms have been proposed in the context of MIMO detection, such as Brun’s algorithm, Seysen’s algorithm and the LLL algorithm. In a thorough analysis presented in Chapter 3, it was shown that the complex version of LLL is the most efficient LR algorithm while maintaining a high-quality lattice reduction required for achieving ML diversity. However, despite this, CLLL in its original form still displays a high-computational and non-deterministic complexity. This makes the original form of CLLL unsuitable for a high-throughput, low-complexity VLSI hardware implementation, which is a crucial requirement for emerging 4G wireless systems.

There have been a few recent research efforts to realize a VLSI implementation of LR [15–18]. These efforts have targeted variants of the CLLL algorithm for implementation, while the LR design [15] implemented Brun’s algorithm. However, these implementations suffer from several deficiencies, which make them incapable of meeting the stringent demands of 4G systems such as LTE and WiMAX. These demands include data rates approaching 1Gb/s, mobile speeds up to 350 km/h and large antenna configurations up to $8 \times 8$.

Presented below are the deficiencies of the current published LR implementations:

- **Performance Loss:** Many of the existing LR designs suffer from a performance loss and are not able to achieve the desired ML-diversity for LR-aided MIMO detectors.

- **Variable Throughput:** Current state-of-the art LR implementations all suffer from non-deterministic processing latency leading to a variable throughput rate. A fixed throughput LR core has yet to be developed.
• **Low Throughput:** A high-throughput pipelined LR core, with fixed processing latency\(^1\), for a \(4 \times 4\) MIMO system has yet to be developed.

• **Large Area:** Implementations with minimal BER performance loss are area intensive despite having low-throughput.

• **Channel Dependent:** The number of iterations of most of the current designs depends on the initial correlation of the input estimated channel, leading to a channel correlation dependent throughput.

• **Non-Adaptive Design:** Existing LR implementations suffer from a lack of flexibility. It is desirable that as the conditions of the channel vary (e.g. SNR and correlation), that the LR reduction behaviour also varies, which is not possible in current proposed designs.

• **No Silicon-Proven Design:** There is currently no existing silicon-proven chip implementation of an LR core. Considering the usage of LR cores as an intellectual property (IP) block, it is critical to verify designs in silicon in the form of test chips. This is necessary to reduce the risk of integrating IP blocks into larger chips implementing complete wireless transceivers.

To the best of our knowledge, all the LR designs in the literature have at least two or more of the above deficiencies. For example, consider the proposed LR VLSI design in [15], which implements an optimized version of Brun's algorithm. Although the proposed ASIC design achieves low processing latency and small area as a result of the low complexity Brun's algorithm, the design suffers from significant performance loss as it is unable to achieve ML diversity. To avoid this performance loss, the majority of proposed LR implementations in the literature have focused on the CLLL algorithm. However, the tradeoff, as discussed in Chapter 3, is that the CLLL algorithm suffers from high-computational complexity and non-deterministic run-time. Current LR implementations have relied on optimizing the CLLL algorithm with the aim of lowering its complexity, however, these optimizations have not yet offered a complete solution as the designs have tended to heavily favour one performance metric at the expense of the other. In order to analyze these tradeoff choices made in previous LR designs, the key elements of the CLLL algorithm that affect the efficiency and optimality of the VLSI implementation are introduced first.

---

\(^1\)Processing latency refers to the number of clock cycles required between consecutive matrix outputs from the LR core. It does not refer to the initialization latency.
As presented in Chapter 3, the main computational procedures in the CLLL algorithm which impact a hardware implementation are:

1. The $\mu$ factor calculation.
2. The size-reduction procedure.
3. The Lovász swapping criteria.
4. The basis update procedure.

These four procedures can occur in each LR iteration and, because of this repetition, the method of implementing these procedures has a great impact on the resulting hardware complexity and overall LR reduction quality. Other critical LR design factors impacting the performance/complexity tradeoff include:

5. The method for limiting the maximum run-time (e.g., limiting the number of iterations or swaps).
6. The bit resolution of the fixed-point implementation.
7. The scheduling of operations, which affects the needed hardware resources as well as the maximum achievable throughput.

All of the above 7 factors need to be optimized in order to achieve the best tradeoff between the performance and complexity of the CLLL VLSI core. However, as mentioned earlier, previous CLLL implementations have tended to focus on some factors to the exclusion of the others. Consider the proposed CLLL implementation in [16] that is able to achieve ML diversity. The tradeoff for achieving this, is that the design relies on accurate representations of the ideal CLLL algorithm, which leads to a high complexity solution. For example, the $\mu$ factor calculation is performed using a divider, which, although accurate, results in a large amount of hardware resources. Similarly, the same design achieves high accuracy for the size reduction and basis update steps, but requires the use of an array of eight multipliers. Finally, in terms of bit resolution, the design uses 26 bits (13 integer and 13 fractional) for the entries in the $R$ matrix. This further increases the complexity of the implementation, especially the complexity of the aforementioned divider and multiplier array. A similar case exists in the proposed LR design in [17] and [18], in which a divider and complex multiplier arrays are used. Moreover, each of the previous three
designs require performing a square root operation for the calculation of the parameters of the Givens rotations matrix. In [16] and [17], this was performed using a 3D-Householder CORDIC module while in [18], the square root operation was performed via an off-the-shelf IP block.

On the other hand, LR implementations that have focused primarily on lowering complexity and increasing throughput, have suffered from a lower lattice reduction quality leading to a BER performance which falls short of ML diversity. For example, consider the proposed LR design in [19] that offers very high-throughput while occupying a small silicon area, which is made possible by limiting the number of column swaps in CLLL to a maximum of four swaps. The drawback is that this low limit for the number of swaps causes an inferior BER performance when combined with a MIMO detector.

In summary, it can be observed that LR implementations have generally fallen into two categories, one group which focuses on maintaining high BER performance (i.e. ML diversity) at the expense of complexity, while the other group focuses on lowering the complexity and increasing throughput, at the expense of BER performance. In this thesis, the objective is to design an LR VLSI core which achieves the best of both worlds, by lowering the complexity of CLLL but with only minimal impact on performance. In the remainder of this chapter, the proposed VLSI implementation of LR based on the hardware-optimized CLLL algorithm (introduced in Chapter 3) will be presented. In addition to addressing the performance/complexity tradeoff, this work also seeks to resolve the aforementioned deficiencies present in LR designs in the literature, such as variable throughput, channel dependence, non-adaptive designs and lack of a silicon-proven design.

4.2 Proposed Lattice Reduction Implementation - General Architecture

Chapter 3 introduced a hardware-optimized CLLL algorithm that greatly reduced the algorithm complexity by eliminating the need for multiplication, divisions and square root operations. The proposed algorithm, HOLLL (hardware-optimized LLL), relies only on additions, comparisons and bit-shift operations. Moreover, these optimizations made to CLLL do not come at the expense of the lattice reduction quality. In fact, this work has shown that in a practical scenario, in which the iterations are limited, HOLLL outperforms
CLLL. In terms of impact on the hardware implementation complexity, the hardware architecture required to implement HOLLL is significantly more efficient than previously reported LR implementations, which required multiplier arrays, dividers and square root modules. Furthermore, as a result of adopting the fixed complexity technique of [53], HOLLL can be implemented in a pipelined fashion with each LR iteration being implemented in parallel, thus allowing for a high-throughput implementation.

The proposed VLSI architecture for the LR core implementing HOLLL for a 4 × 4 MIMO system is shown in Fig. 4.1 below. The complete architecture consists of an input controller, followed by 7 HOLLL pipelined stages and ending with an output controller. The input and output controllers are necessary to accommodate the limited number of IO pads associated with a cost-effective silicon implementation. The 7 pipelined LR stages perform 9 LR iterations, with iterations (3,4) and (6,7) being performed in parallel\(^2\). As a result of adopting the fixed complexity method in [53], the 9 LR iterations follow the sequence \(k = \{2, 3, 4, 2, 3, 4, 2, 3, 4\}\), where \(k\) is the column of \(R\) operated on in each iteration, as shown in Table 3.4. The initial inputs to the LR core are a 4 × 4 complex \(R\) matrix, four 4 × 1 complex \(z\) vectors\(^3\) (represented by the matrix \(Z_{4 \times 4}\)) and the identity matrix \(I_{4 \times 4}\). The \(R\) matrix comes from a modified QR block that takes the estimated channel \(H\) and the received vector \(y\), performs the QR decomposition and outputs the \(R\) matrix (where \(H = QR\)) and the \(z\) vector (where \(z = Q^H y\)). A QR decomposition block of this type can be found in [54]. The input identity matrix \(I_{4 \times 4}\) represents the initial state of the complex unimodular transformation matrix \(T\) used to perform the lattice reduction process (i.e. \(\tilde{H} = HT\)). Through an optimized scheduling sequence, which will be discussed later in this chapter, the iterations for \(k = 2\) and \(k = 4\) column operations can be performed in parallel. This in turn reduces the total latency to 7 LR stages instead of 9.

As shown in Fig. 4.1, the intermediate \(\tilde{R}, \tilde{Z}\) and \(T\) values are stored in register banks between each stage. Control of the sequence of operations is implemented using a distributed approach, where each LR stage has its own dedicated controller for local operations. All 9 LR iterations are run in parallel through the 7 stage pipelined architecture which are synchronized such that every 40 clock cycles, on the positive edge of the clock, the updated \(\tilde{R}\), \(\tilde{Z}\) and \(T\) values in each register bank are transferred to the one in the next pipeline stage.

\(^2\)Later in this Chapter it is shown how iterations with \(k = 4\) and \(k = 2\) can be performed simultaneously leading to the ability to parallelize iteration (3,4) and (6,7).

\(^3\)Four detected vectors are needed due to the quasi-static channel being updated every four channel use.
After completing the 9 LR iterations, the output controller produces, every 40 cycles, the LR-reduced $4 \times 4 \tilde{\mathbf{R}}$ matrix, the $4 \times 4 \mathbf{T}$ matrix, and four $4 \times 1 \tilde{\mathbf{z}}$ vectors.

The detailed VLSI architecture of the LR stages and their internal submodules is presented next, including the scheduling used to achieve a low processing latency of 40 clock cycles (i.e. new LR-reduced $\tilde{\mathbf{R}}$, $\tilde{\mathbf{z}}$ and $\mathbf{T}$ outputs every 40 cycles).

### 4.3 Proposed Lattice Reduction Implementation - Detailed VLSI Architecture

Following a bottom-up approach, the next sections describe the architectural blocks used to implement the Size Reduction and Basis Update steps (respectively lines 5-9 and 10-14 in Table 3.4), followed by the architecture used for a single HOLLL iteration and finally the pipelined integration of all the LR stages and the overall task scheduling.
4 VLSI Implementation of Hardware-Optimized Lattice Reduction

4.3.1 MU Calculation

Referring to Table 3.5, which details the $\mu$ quantization procedure, it is observed that the calculation of the quantized complex $\mu_q$ value consists of separately calculating the real and imaginary components ($\mu_r$ and $\mu_i$) as well as their respective signs. However, given the knowledge of the future use of the $\mu_q$ factor (i.e. multiplications in the size reduction operations) and also considering that the quantized values are limited to $\{0, \pm 1, \pm 2\}$, it is possible to avoid explicit calculation of the $\mu_r$ and $\mu_i$ components. This is done by decomposing $\mu_r$ and $\mu_i$ into the individual results of the conditional statements in Table 3.5 and using these binary values as multiplexor controls in the size reduction operation, which is shown in the following section. The individual results of the conditional statements involving $\mathcal{R}\{ \tilde{R}_{l,k} \}$ (lines 3-4 in Table 3.5) are denoted $\mu_1^{Re}$ and $\mu_2^{Re}$, such that $\mu_r = \mu_1^{Re} + \mu_2^{Re}$. In a similar fashion, $\mu_i = \mu_1^{Im} + \mu_2^{Im}$.

![Figure 4.2: Architecture for the Quantized $\mu_q$ Calculation Block (block I/O names are shown in parenthesis).](image)

Based on the above optimization, the architecture for computing the quantized $\mu_q$ coefficient is shown in Fig. 4.2. As seen in the figure, the quantization results in elementary operations leading to a calculation delay of only one clock cycle. The comparators shown in the figure are multi-bit comparators to accommodate the word length of the size reduction inputs.\footnote{Fixed point parameters are discussed in Section 4.5} It should be noted that the conditional results ($\mu_1^{Re}$, $\mu_2^{Re}$, $\mu_1^{Im}$, $\mu_2^{Im}$) are...
4 VLSI Implementation of Hardware-Optimized Lattice Reduction

stored in local registers to be used by the size reduction operations.

### 4.3.2 Complex Size Reduction

The size reduction operations in the proposed HOLLL algorithm (lines 7-8 in Table 3.4, further expanded into real and imaginary operations in equation 3.2) are achieved via a Complex Size Reduction block (CSR) using the control outputs from the $\mu$ quantization block. The architecture for the CSR block is shown in Fig. 4.3. In total, for each iteration, three CSR blocks are used. One CSR block is used for calculating the real value of $\tilde{R}(1 : l, k)$, while another CSR block is used for the imaginary component calculation. A third is used for $T$ matrix size reductions. As will be seen in the upcoming discussion of the schedule, the size-reduced values of $R$ are time-critical, in the sense that they are required immediately by CORDIC rotation operations. Any delay in calculations would lead to delaying the start of the CORDIC operations, thus increasing the processing latency. Hence, the real and imaginary size reduction operations are parallelized by using two CSR blocks. On the other hand, the real and imaginary size-reductions of the $T$ matrix can be serialized, since they are not used in CORDIC operations and thus do not impact the processing latency. Note that as a result of the quantized $\mu_q$ factor, only elementary operations (add/sub/shift) are required in the CSR block to implement lines 7-8 in Table 3.4, thus each CSR block has an execution time of only one clock cycle.

![Figure 4.3: Architecture for the Complex Size Reduction (CSR) Block (block I/O names are shown in parenthesis).](image)

In the proposed implementation, the aim was to maximize hardware utilization and minimize area by requiring only three CSR blocks and reusing them, through controlled
multiplexing of inputs and outputs, to complete all the size reduction operations. Previous efforts to reduce the complexity of LLL were based on performing size reduction against \( \mathbf{R}(k - 1,:) \) only (i.e. only the first execution of the loop in lines 5-9 in Table 3.4); this is known as Effective LLL (ELLLL) [17]. However, in the proposed design, performing full size reduction with minimal hardware (using only three CSR blocks) is possible without affecting the latency of the overall HOLLL iteration. This is because, as shown later, the latency of Givens rotations via pipelined CORDIC operations (which is itself at maximum throughput) is longer than that of size reduction in all cases of \( k \) (i.e. \( k = \{2, 3, 4\} \)).

### 4.3.3 Siegel Calculation

The \( \delta \) factor in the swapping condition (using either Siegel or Lovász conditions) plays a key role in the performance of the LR algorithms. A larger \( \delta \) typically produces higher quality LR-reduced matrices leading to a better BER performance, but a smaller \( \delta \) results in a faster execution and a better iteration limited performance. With this in mind, the Siegel condition (line 10 in Table 3.4) was implemented to allow dynamic control of \( \delta \) as shown in Fig. 4.4, where the value of \( \delta \) can be controlled via primary inputs to the LR core. The allowable \( \delta \) values were selected from the set \( \{\frac{1}{8}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}\} \). This flexibility allows the LR algorithm to adapt to varying input conditions (e.g. SNR and correlation) as well as allow for dynamic control of \( \delta \) within a single LR reduction. This dynamic control is important since, following suggestions by [49] and [55], our simulations show that the lattice reduction quality for limited iterations can be further increased by applying a smaller \( \delta \) in the earlier LR iterations (to maximize speed) followed by a larger \( \delta \) in the latter LR iterations (to maximize quality), as opposed to applying a fixed value for all iterations.

![Figure 4.4: Architecture for Siegel Condition with Dynamic \( \delta \) Control (block I/O names are shown in parenthesis).](image)

\[ \delta_2 \delta_1 \]

<table>
<thead>
<tr>
<th>( \delta_2 )</th>
<th>( \delta_1 )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0.625</td>
<td>0.625</td>
</tr>
<tr>
<td>01</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>0.375</td>
<td>0.375</td>
</tr>
<tr>
<td>11</td>
<td>0.25</td>
<td>0.25</td>
</tr>
</tbody>
</table>

64
4 VLSI Implementation of Hardware-Optimized Lattice Reduction

4.3.4 Basis Update Implementation

The Basis Update step (lines 11-12 in Table 3.4) consists of column swapping followed by Givens Rotations which are implemented using 2D CORDIC vectoring and rotation operations. CORDIC processors typically fall into two categories in terms of their architecture, namely: 1) iterative architectures, and 2) pipelined architectures. The advantage of the iterative approach is that the hardware resources required are minimized to a great degree. However, this comes at the expense of parallelizing operations and thus limits the maximum throughput. With respect to the Givens rotations required in the Basis Update step in the proposed algorithm, there is a large number of 2D CORDIC rotations operations which must be performed following each of the required three vectoring operations shown in Fig. 3.8. Using only one CORDIC processor for all the iterations of each of the rotations operations will severely increase the latency of the Basis Update step and thus lead to a very low overall throughput for the LR ASIC.

To maximize the throughput, it was chosen to unroll the 2D CORDIC processor into 9 pipelined stages, each having a one cycle delay for a total delay of 9 clock cycles. Fig. 4.5 shows the architecture for the 9 CORDIC stages. Each CORDIC stage can be configured to be used in either vectoring mode or rotation mode thus achieving maximum utilization. Furthermore, instead of explicitly calculating the CORDIC rotation angles, direction signals were used to encode the rotation angle as shown in [52]. This implicit angle calculation results in eliminating the angle data path, hence achieving a 30% hardware savings.

The first and last CORDIC stages each include coarse rotation options for vectoring and rotation, respectively. The coarse rotation capability is necessary to extend the range of the CORDIC rotation angles from \([-\pi/2, \pi/2]\) to \([-\pi, \pi]\), and thus allow the zeroing of vectors \([x, y]\) with a negative \(x\) value. The CORDIC vectoring mode can only operate on \([x, y]\) with a positive \(x\) value, therefore, an input coarse rotation capability \((c_{in})\) was included in the first CORDIC stage. In vectoring mode, if the sign of \(x\) is negative, then \(c_{in}\) is asserted and the \([x, y]\) input is rotated. The value of \(c_{in}\) is stored locally and is only subject to changing during the next vectoring mode. The subsequent rotation operations proceed normally, however in the last CORDIC stage, an output coarse rotation is applied based on the value of \(c_{out}\), which takes on the value of \(c_{in}\) if in rotation mode. The last CORDIC stage also includes a scaling unit to apply the 2D CORDIC gain compensation factor of \((0.6073)\). The scaling unit is implemented using shift and add operations to apply an approximate factor of \((0.625)\).
Figure 4.5: Block Diagram of the Configurable 9-Stage Pipelined CORDIC.
Pipeline Operation

The pipeline operates as follows. A 2D vector \([x, y]\) is fed into the first stage of the pipeline. This initial vector is connected to the two 16-bit inputs \(x_0\) and \(y_0\). The mode required is also supplied as an input through the binary signal \(m_0\), where a value of 1 indicates vectoring mode, and a value of 0 indicates rotation mode. As discussed in Section 3.4.4, in every basis update operation (whether for column \(k = 2, 3, \) or 4), there exists three vectoring operations. Each of these is followed by a series of rotation operations which rotate using the angle produced in the vectoring operation (i.e. the angle needed for nulling the \(y\) component). As a result, in terms of the pipelined CORDIC implementation, it is safe to assume that there is always a vectoring operation occurring before starting rotation operations. This implies that the direction signals, which represent the rotation angle, do not need to be supplied as inputs to the pipeline. Rather, during vectoring mode, they are calculated based on the sign of the \(y\) component, and stored locally in each stage for later use in rotation operations during the subsequent rotation mode.

4.3.5 Balanced Pipeline Design

The main architectural blocks used to implement the Size Reduction and Basis Update steps were presented above, namely: the \(\mu\) calculation block, the CSR block, the Siegel calculation block and the 9-stage pipelined CORDIC. By examining the signal paths in the design blocks, it can be observed that the resulting critical path in the LR core consists of two adders and two multiplexors. This occurs in both the CSR and the last stage of the pipelined CORDIC. Meanwhile, the fastest data path consists of one adder and one multiplexor, which occurs in the intermediate stages of the pipelined CORDIC. As a result, it is clear that the path delays in the LR core are quite similar, with no one very slow path dominating the critical path. This leads to a balanced pipelined design, and hence no re-timing optimization was required at the synthesis stage\(^5\).

\(^5\)In designs where one slow path dominates the critical path and no further pipelining can be performed (e.g. maximum allowed latency has already been reached), then one possible technique is to resort to re-timing during the synthesis stage. Re-timing involves moving registers across cycle boundaries with the aim of achieving better balance between path delays.
4.3.6 Implementation of a Single Lattice Reduction Iteration

The size reduction and basis update blocks presented above are now combined to form one single HOLLL iteration. This can be observed in the block diagram for an individual HOLLL iteration shown in Fig. 4.6 below. The diagram also shows the clock cycles required for the three cases: \(k=2\), \(k=3\) and \(k=4\). The scheduling shown was optimized to minimize hardware resources while maximizing throughput, resulting in a low processing latency of 40 cycles. The specifics of the task scheduling will be discussed in detail later in this chapter. The general operation of the HOLLL iteration is discussed below.

Matrix data is read from the input register bank, followed by the \(\mu_q\) and Siegel calculation blocks which execute in parallel and in one clock cycle. The results of the \(\mu_q\) calculation are stored in local registers and used to control the addition/subtraction in the Size Reduction
step. If the Siegel condition passes, then the Basis Update Step (lines 11-12 in Table 3.4) starts in cycle 3 when the new size-reduced $\tilde{R}_{k,k}$ value is available. Through an efficient scheduling sequence, all CORDIC operations are complete by clock cycles 44, 42 and 40 for $k=2$, $k=3$ and $k=4$, respectively. The final values are stored in the output register bank to be used by the next HOLLL iteration. Note that each HOLLL iteration has a dedicated controller that manages dataflow and scheduling. The core of the controller consists of a modulo 40 counter. The value of this counter indicates which operations to perform during that specific clock cycle to achieve the required schedule.

4.4 Proposed Lattice Reduction Implementation - Task Scheduling

This section presents, in detail, the scheduling adopted to achieve a high throughput of one LR-reduced output matrix every 40 cycles. Presented first is the scheduling used for inputting matrix data, followed by the scheduling used within each LR iteration for different matrix columns (i.e. different $k$ values). Finally, the scheduling used for outputting the LR-reduced matrix data is shown, followed by a discussion of the pipelined fashion in which the proposed LR core can be used in a larger receiver system.

4.4.1 Inputs

The inputs to the architecture are the entries of the complex $R$ matrix and the complex $z$ vector in equation (2.8). For a $4 \times 4$ MIMO system, the $R$ matrix has the following structure:

$$
R = \begin{bmatrix}
\tilde{R}_{11} & R_{12} & R_{13} & R_{14} \\
0 & \tilde{R}_{22} & R_{23} & R_{24} \\
0 & 0 & \tilde{R}_{33} & R_{34} \\
0 & 0 & 0 & \tilde{R}_{44}
\end{bmatrix},
$$

where the diagonal entries are real-valued while the upper-triangular entries are complex-valued. For the $z$ vector, recall that in the channel model it is assumed that the channel is quasi-static and is updated every four channel uses. As a result, there are four $4 \times 1$ complex received vectors $y$ for a given channel $H$. This implies that the proposed LR ASIC
core is required to process four $4 \times 1$ vectors (represented by $\mathbb{Z}_{4 \times 4}$) for every $\mathbb{R}$ matrix. Therefore, decomposing the complex-valued entries into two separate entries, one real and one imaginary, yields a total of 16 distinct real-valued entries for the $\mathbb{R}$ matrix and 32 entries for the input $\mathbb{Z}$ matrix. Assuming each entry requires 16 bits\textsuperscript{6}, a straightforward parallel input approach will require 768 pads ($(32+16) \times 16$), which is not a practical cost-efficient solution. To avoid this large number of pads in the manufactured LR chip, the inputs are received serially rather than in parallel and stored for later use.

The optimized input approach which was adopted, consists of 8 pads for real entries and 8 pads for imaginary entries. Given that each entry is 16 bits, it is necessary to multiplex the inputs, such that each entry is received over two clock cycle edges. The lower byte, referring to the eight least significant bits (LSB), is entered on the positive edge of the clock, while the upper byte, referring to the eight most significant bits (MSB), is entered on the following negative edge of the clock. For a total of 26 real entries (10 for $\mathbb{R}$ and 16 for $\mathbb{Z}$), this translates to 26 clock cycles. The imaginary entries can similarly be entered in 22 cycles (since there are 6 imaginary entries for $\mathbb{R}$ and 16 for $\mathbb{Z}$). The drawback of this approach is the higher initial latency, however this is only upon start up and does not affect the throughput of the design. In addition, considering integrating the proposed ASIC LR into a MIMO receiver, the preceding QR decomposition block will not simultaneously produce all entries for the output $\mathbb{R}$ and $\mathbb{Z}$ matrices, but rather the outputs will be distributed over a number of clock cycles. Therefore, given a joint approach of QR and LR, the overall data transfer latency can be eliminated through optimal scheduling.

As the $\mathbb{R}$ and $\mathbb{Z}$ matrices are entered, they are stored in a register bank. This input register bank is also pre-loaded with the identity matrix $\mathbb{I}_{4 \times 4}$, which is the initial value for the $\mathbb{T}$ matrix. An input controller module controls the operation of reading the input values and storing them in the register bank.

4.4.2 Scheduling of Pipeline Stages

In order to achieve a low processing latency of 40 cycles, careful scheduling was needed to maximize the use of hardware components in each LR iteration. The LR iterations exist in three variations, one for each of the three cases of $k = \{2, 3, 4\}$, where $k$ indicates the column number for a given LR iteration. In addition, it was previously shown that each

\textsuperscript{6}The final fixed point implementation of the proposed LR core indeed utilizes a 16 bit resolution for $\mathbb{R}$ and $\mathbb{Z}$ entries, as will be discussed later in the chapter.
iteration consists of two main steps, namely, the Size Reduction step and the Basis Update step. Therefore, the scheduling for each of these steps for all three column variations is presented next.

**Size Reduction Scheduling**

As presented earlier, the Size Reduction step consists of four main hardware blocks, namely, one μ calculation block, two CSR blocks for the real and imaginary R reduction and finally one CSR block for T matrix reductions. The task schedule for each of these blocks for the case of $k=2$ is shown in Fig. 4.7. On the right hand side, the clock cycles with respect to the 40 cycle processing latency is shown. Starting at the first cycle (i.e. cycle #0), the μ calculation block executes by taking in the matrix values $\tilde{R}_{1,1}$, $\mathfrak{R}(\tilde{R}_{1,2})$ and $\mathfrak{I}(\tilde{R}_{1,2})$ which correspond to the inputs $x_{Re_i}$, $y_{Re_i}$ and $y_{Im_i}$. Note in the figure, a solid line around the hexagon indicates a real-valued entry while a dotted line is used to indicate the imaginary component. In addition, the number inside each hexagon represents the matrix column number. In the next cycle, the μ block produces $\mu_r$ and $\mu_i$ on the outputs $mu_{Re_o}$ and $mu_{Im_o}$\(^7\). These outputs are fed directly into the three CSR blocks to begin the size reduction. Note that since $\tilde{R}_{2:N_T,1}$ are zero by definition, therefore only one real size reduction and one imaginary size reduction are needed. For T however, all rows need to be size reduced, implying four real and four imaginary size reductions. As shown in Fig. 4.7, the implementation of the four real size reductions is followed by the four imaginary ones. Note that in the figure, the colour of the hexagon is used to indicate the row of operation in the matrix. For $R$ rows start from $k$-1 up to 1 (i.e. only one row for the case of $k=2$), while for $T$, the rows are operated on in order from 1 to 4.

For the case of $k=2$, the final size-reduced outputs for $\tilde{R}$ and $T$ are produced in cycles 2 and 9 respectively. The schedules for cases $k=3$ and $k=4$ are shown in figures 4.8 and 4.9. The same comments made above regarding the representation of matrix entries (i.e. solid/dotted lines, number and color of hexagons) apply. For $k=3$, the final size-reduced outputs for $\tilde{R}$ and $T$ are produced in cycles 10 and 17, and for $k=4$ they are in cycles 18 and 25. To help understand the scheduling figure, an illustration of the operations performed for $k=4$ is shown in Fig. 4.10.

It should be noted that the delays in calculating the second and third values of μ shown

\(^7\)Note that in the actual hardware implementation, the outputs of the comparisons (i.e. $\mu_{Re}^1$, $\mu_{Re}^2$, $\mu_{I_m}^1$ and $\mu_{I_m}^2$) are used instead of $\mu_r$ and $\mu_i$. However, for the sake of clarity, the $\mu_r$ and $\mu_i$ notation is kept here to help better understand the schedule.
in the schedules for \( k = 3 \) and \( k = 4 \) have been chosen in order to correspond with the end of size reduction operations in \( T \), which require the previous \( \mu \) value. This has no impact on the processing latency, since the only size reduction values that are critical in terms of processing latency are the \( \Re(\vec{R}_{k-1,k}) \) and \( \Re(\vec{R}_{k-1,k}) \) outputs, which are used in the CORDIC vectoring operations (if a swap is indicated). These two outputs are produced in the shortest amount of cycles (i.e. in cycle \#2), for all cases \( k = \{2, 3, 4\} \).
Figure 4.8: Size reduction schedule k=3.
Figure 4.9: Size reduction schedule k=4.
Figure 4.10: Illustration of Size Reduction operations for $k=4$. 

\[ \begin{pmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ 0 & R_{22} & R_{23} & R_{24} \\ 0 & 0 & R_{33} & R_{34} \\ 0 & 0 & 0 & R_{44} \end{pmatrix} \]
Basis Update Scheduling

As presented earlier, the Basis Update step in the proposed implementation consists of two main hardware blocks namely, the Siegel calculation block and the 2D CORDIC pipeline. The Siegel calculation block is identical for all column cases \( k = \{2, 3, 4\} \), since in all cases the same comparison \( \delta \cdot |\tilde{R}_{k-1,k-1}| > |\tilde{R}_{k,k}| \) applies. In addition, for all \( k \) cases, the Siegel calculation block executes during the first cycle (i.e. cycle \#0) and produces the column swap decision at the positive edge of cycle \#1, as shown in Fig. 4.6. If a swap is indicated, then this triggers the beginning of executing Givens rotation operations via the 2D CORDIC pipeline. As discussed in Chapter 3, there are three vectoring operations regardless of the value of \( k \). However, in terms of rotations, the cases \( k = \{2, 3, 4\} \) require 27, 23 and 19 rotations operations respectively. Given this large number of rotations and also the pipeline depth of 9 CORDIC iterations, it is obvious that the Givens rotation calculation dominates the processing latency of the LR iteration. As a result, it is crucial to implement an optimized schedule in order to meet the requirement of an overall processing latency of 40 cycles.

The task schedule for CORDIC operations for the case of \( k = 2 \) is shown in Fig. 4.11. The legend below the schedule indicates, as before, that a hexagon with a solid line represents a real-valued matrix entry while that with a dotted line represents the imaginary component. The colours of the hexagons are used to separate between \( \mathbf{R} \) and \( \mathbf{Z} \), and they also indicate the different rows of the two matrices. The Givens rotations procedure for the case of \( k \) involves rows \( k \) and \( k - 1 \) of matrices \( \mathbf{R} \) and \( \mathbf{Z} \), therefore, as shown in Fig. 4.11, the two rows involved in CORDIC operations are 1 and 2, since the case \( k = 2 \) is the one under consideration. Finally, the number inside the hexagon indicates the column number of the matrix entry. The arrows in the figure represent feedback paths in which the output of the CORDIC pipeline is directly fed back into its input.

The CORDIC operations proceed as follows. Once a column swap is indicated and performed (i.e. the situation of Fig. 3.8) then in the following cycle the mode input \( m_i \) is asserted to indicate vectoring mode and the inputs \( x_i \) and \( y_i \) are loaded with \( \mathbf{R}(\tilde{R}_{1,1}) \) and \( \mathbf{R}(\tilde{R}_{1,1}) \), respectively. This vector proceeds through the 9 stages of the CORDIC pipeline until the imaginary components are nulled. During this nulling procedure, the direction signals for each stage are stored. Following the first input vector, the mode input is deasserted and the remaining complex vectors of row 1 of \( \tilde{\mathbf{R}} \) and \( \tilde{\mathbf{Z}} \) are entered to be rotated using the stored direction signals. Following the same method, the second
vectoring operation starts when the first output, $\Re(\tilde{R}_{1,1})$, is ready (i.e. at the positive edge of the 10th cycle). This output along with $\Re(\tilde{R}_{2,1})$ serve as the input vector. As before, the remaining entries in the row are entered after this in a sequential manner, where real vectors and imaginary vectors are rotated separately. The third vectoring operation starts when the updated complex vector $[\Re(\tilde{R}_{2,2}), \Im(\tilde{R}_{2,2})]$ is ready. Again, the mode input is asserted to indicate a vectoring operation, and then deasserted to perform the remaining 6 rotations.

The final outputs from the CORDIC pipeline for the case of $k = 2$ is the updated $\Re(\tilde{Z}_{2,4})$ and $\Im(\tilde{Z}_{2,4})$ matrix entries, which are produced in the 42nd cycle. Since the CORDIC pipeline itself starts, as discussed earlier, at the positive edge of cycle #2, therefore the total latency of the LR iteration in the case of $k = 2$ is 44 cycles. The schedule for cases $k = 3$ and $k = 4$ follow the same logic as presented above for $k = 2$ and are shown in Fig. 4.12 and Fig. 4.13, respectively. Similarly, the resulting processing latency for cases $k = 3$ and $k = 4$, are found to be 42 and 40 cycles.

Note that the latencies for $k = 2$ and $k = 3$ are 44 and 42 cycles, which are both higher than the required processing latency of 40 cycles. One straightforward way to resolve this is to reduce the number of CORDIC stages, such that 40 cycles can be met. However, the drawback is the associated reduction in BER performance and moving further away from ML diversity as shown in Fig. 3.9. A better solution is to try to manipulate the schedule
Figure 4.12: CORDIC pipeline schedule for $k = 3$.

Figure 4.13: CORDIC pipeline schedule for $k = 4$. 
such that the overall LR core processing latency of 40 cycles can still be met without sacrificing CORDIC stages. The presented schedule was chosen to exactly achieve this. The key is always starting with $\mathbf{R}$ rotation operations before $\mathbf{Z}$ rotations\(^8\). Taking the example of $k = 2$, it is clear that this leads to the final four output vectors being $\mathbf{Z}$ vector outputs, as seen in Fig. 4.7. These matrix entries are not involved in any of $\mu$ calculation, Siegel calculation or size reductions of the next LR iterations, i.e. $k = 3$. Furthermore, for Givens rotations during the Basis Update step, these entries are only required when starting cycle #5. As a result, the LR iteration of $k = 3$ can in fact start exactly at the 40 cycle mark, thus hiding the higher 44 cycle processing latency of the previous $k = 2$ iteration. Similarly the 42 cycles of $k = 3$ can be hidden by starting the next $k = 4$ iteration at the 40 cycle mark. Finally, since the final (i.e. ninth) LR iteration implements a $k = 4$ case, therefore the entire LR core can sustain a 40 cycle processing latency.

![Figure 4.14: CORDIC wrapper for efficient implementation of feedback paths.](image)

To conclude this discussion on CORDIC pipeline implementation and scheduling, the implementation strategy used to efficiently perform the feedback paths (represented by arrows in the CORDIC scheduling figures) is presented. Looking at the schedules, it can be noted that the feedback inputs into $x_i$ are either $x_o$ or a delayed version of $y_o$. Carefully examining the schedule, it was observed that the number of delay cycles is consistent for a given $k$ value. For $k = \{2, 3, 4\}$ the delay cycles are 7, 6 and 5, respectively. As for the feedback inputs into $y_i$, then this only consists of $y_o$. To greatly simplify the required control logic and input multiplexing, it was decided to implement a CORDIC pipeline wrapper block which employs a feedback shift register, feedback paths and two

\(^8\)Note that since the direction signals are saved locally during vectoring mode, therefore the order of subsequent rotations does not affect the result. However, carefully chosen ordering is necessary to achieve the lowest possible processing latency.
input multiplexors. The wrapper, shown in Fig. 4.14, allows for simple switching between inputs and various feedback paths via control of the multiplexor taps.

### 4.4.3 Other Pipeline Choices

In the proposed LR core, a low processing latency of 40 cycles was targeted. The pipeline choices made were based on this target, which in turn dictated the critical path of the design as discussed in Section 4.3.5. The 40 cycle target meant that no reduction of the critical path (existing in the CSR block and the last stage of the CORDIC pipeline) was possible via further pipelining. For example, pipelining the CSR block to execute in two clock cycles instead of one, would result in an extra cycle delay to the start of the Givens rotations operation. Hence, the first vectoring operation would start in cycle #4 instead of cycle #3. This would result in the Givens rotations completing one cycle later, leading to a processing latency of 41 cycles instead of 40 cycles.

For LR designs targeting processing latencies higher than 40 cycles, other pipeline choices can be made and lower critical paths can be achieved. For example, the architectural design blocks of the LR core can be further pipelined such that the critical path consists of only one adder and one multiplexor (instead of the current critical path composed of two adders and two multiplexors). This would lead to a higher maximum clock frequency, but with the tradeoff of having a higher processing latency.

### 4.4.4 Iteration Scheduling

In the previous section, it was shown how each LR stage \((k = \{2, 3, 4\})\) can operate at a throughput of 1 LR iteration per 40 cycles. Implementing 9 LR iterations is possible by simply cascading 9 LR stages in order, resulting in a total latency of 360 cycles. However, through careful examination of the HOLLL algorithm, it can be seen that \(k = 4\) iterations operate primarily on \((\tilde{R}_{3:4,:}, \tilde{Z}_{3:4,:}, T_{3:4,:})\), while \(k = 2\) iterations operate on \((\tilde{R}_{1:2,:}, \tilde{Z}_{1:2,:}, T_{1:2,:})\). Hence, as shown in Fig. 4.1, the iterations corresponding to \(k = 4\) and \(k = 2\) can be parallelized (i.e. both execute during the same 40 cycle period). Note, that finely tuned scheduling is required to accommodate size reduction operations on matrix entries common to both \(k = 4\) and \(k = 2\) iterations (i.e. \(\tilde{R}_{1:2,3:4}\) and \(T_{1:2,3:4}\)). As a result of parallelizing \(k = 4\) and \(k = 2\) iterations, the total latency is reduced by 80 cycles, leading to a latency of 280 cycles.
4.4.5 Outputs

The outputs from the final LR stage are the LR-reduced complex \( \tilde{R} \), complex \( \tilde{Z} \) matrices in addition to the complex unimodular transformation matrix \( T \). These outputs are stored into the final register bank, referred to as the output register bank. These LR-reduced matrix values are then driven through output pads in a certain sequence and with a specific format via the output controller block. Similar to the discussion regarding the scheduling of the inputs to the LR core, the aim here is to increase efficiency and reduce costs of implementation by lowering the number of pads as much as possible while maintaining a processing latency of 40 cycles.

For a \( 4 \times 4 \) MIMO system, the individual matrix outputs consist of 16 distinct real-valued entries for the \( \tilde{R} \) matrix and 32 entries for the \( \tilde{Z} \) matrix. This is identical to the number of \( \tilde{R} \) and \( \tilde{Z} \) entries presented in the discussion of input scheduling. For the \( T \) output matrix, there are 16 complex entries, which correspond to 32 real-valued matrix entries. The word lengths for \( \tilde{R} \), \( \tilde{Z} \) and \( T \) are 16, 16 and 7, respectively. Each of these outputs will be provided to the MIMO detector which follows the LR block. Considering a MIMO detector chip with separate entries for \( \tilde{R} \), \( \tilde{Z} \) and \( T \), the output pads are divided into \( \tilde{R} \) pads, \( \tilde{Z} \) pads and \( T \) pads. Moreover, the outputs are constrained to only emerge on the positive edge of the clock to offer as much flexibility for the subsequent MIMO detector.

For the limit of 40 cycles and given the number of real-valued entries for each matrix, the chosen pad numbers are 8, 16 and 7 for \( \tilde{R} \), \( \tilde{Z} \) and \( T \), respectively. For \( \tilde{R} \), there are only 16 entries, so 8 pads were used with multiplexing, such that a single entry is output over 2 cycles; leading to 32 total output cycles. For \( \tilde{Z} \) and \( T \) this is not possible, since the number of entries is greater than 20 (i.e. \( 40/2 \)), therefore, one matrix entry is outputted per clock cycle. Hence the need for 16 pads for \( \tilde{Z} \) and 7 pads for \( T \), which leads to a total output latency of 32 cycles for each. The exact output sequence, which is performed by the output controller, is shown in Appendix A.

4.4.6 LR Core Usage and Adaptability Features

The final LR core consists of a total latency of 320 cycles (40 for input stage, 280 for the optimized 7 LR stages (9 LR iterations) and 40 for the output stage). The processing latency for the LR core is 40 cycles, which allows for accepting new \( R \) and \( Z \) inputs and producing LR-reduced \( R \), \( Z \) and \( T \) matrices every 40 cycles. This operation is illustrated
in Fig. 4.15. To make the proposed LR core adaptable to various conditions, it was chosen to implement two features which can be controlled during the LR operation, without a need for resetting the core. The first feature is the dynamic control of the $\delta$ swapping threshold factor for any given iteration of the 9 LR iterations. As shown earlier, this feature is advantageous in its ability to customize the LR reduction for various conditions of operations (e.g. SNR, modulation, detection method). To achieve this the $\delta$ factor is provided as a primary input through two pads to implement the coding shown in Fig. 4.4, namely: $\{00,01,10,11\} = \{0.25,0.375,0.5,0.625\}$.

The second feature implemented in the LR core is a dynamic ON/OFF control of applying LR. As shown in Fig. 3.13, LR techniques only become advantageous, in terms of increasing BER performance of the K-Best detector, in high SNR regimes. In low to medium SNR ranges, the K-Best detector actually outperforms LR-aided K-Best. This is due to the boundary control operations, as discussed in Section 3.2.4. Thus, it is desirable to have a pass-through mode for the LR core, in which no lattice reduction is applied to the matrix inputs. To achieve this, the dynamic ON/OFF LR control was implemented as a primary input which can be changed for each new input data set (i.e. for each new R and Z matrices). The dynamic capability allows for seamless transition in and out of LR mode, which is important to maintain a high throughput and avoid any undesirable pipeline bubbles.

An optimal use of this LR by-pass feature in a complete receive system would be to control the ON/OFF LR operation via an SNR estimation block. The system can be pre-characterized during a training period to determine at which SNR LR should start to apply. For example, in Fig. 3.13 LR should be applied for SNR values higher than 31dB.
4.5 Fixed-Point Simulations

The chosen bit resolution has a large impact on BER performance. This is due to the fact the input channel matrix (which in fact is the $R$ matrix after QR decomposition) passes through a series of LR iterations in which it is both the data that is iteratively modified (via fixed-point size reduction and basis update operations) but also the control criteria used to determine future operations and iteration sequences. Maintaining high accuracy in fixed-point operations is critical because of this dual use of the channel matrix as data and control, however, this obviously conflicts with the goal of reducing complexity.

To efficiently derive the fixed point parameters, extensive bit-true simulations of the floating point version of HOLLL were performed. Bit-true simulations refer to the detailed modeling of hardware operations but with floating point accuracy. Based on this bit-true representation, the dynamic range was analyzed, throughout all the LR iterations, of all variables storing matrix entries of $\tilde{R}$, $\tilde{Z}$ and $T$. This was used to establish the required number of bits for integer and fractional parts.

In the tradeoff between performance and implementation area, the word lengths presented in Table 4.1 were shown to achieve the best BER results. The word lengths consist of an integer part and a fractional part, where 4 integer bits and 12 fraction bits are used for both $\tilde{R}$ and $\tilde{Z}$ matrices, and 7 integer bits are used for the $T$ matrix. These fixed point parameters are collectively summarized as $[16,12,7]$.

Table 4.1: Fixed-point Word-Length (bits) of LR Data Variables

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$R_{ij}$</th>
<th>$Z_{ij}$</th>
<th>$T_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Length$^1$</td>
<td>[16:12]</td>
<td>[16:12]</td>
<td>[7:0]</td>
</tr>
</tbody>
</table>

$^1 [n : m]$ an $n$-bit number with $m$ bits for the fractional part.

Fig. 4.16 shows the BER performance of the fixed point HOLLL algorithm compared to its floating point counterpart, as well as the CLLL algorithm in a practical iteration-limited scenario. The detection algorithm used is the near-ML K-Best with $K=10$ and $4 \times 4$ 64-QAM MIMO. For high SNR values and an iteration limit of 9, the HOLLL algorithm outperforms CLLL (0.6dB at BER = $10^{-4}$ and 2.2dB at BER = $10^{-5}$). In addition, there is only a small performance loss due to the fixed point implementation of HOLLL, relative to the floating point version of the algorithm (only 0.3dB at BER = $10^{-5}$).
4.6 Test Results and Design Characteristics Comparison

The proposed VLSI architecture was modeled in Verilog HDL using ModelSim, synthesized using Synopsys Design Vision and placed and routed using Cadence SoC Encounter. The chip boundary and final GDS (Graphic Database System) stream out was performed using Cadence Virtuoso. The golden fixed-point MATLAB model was used to validate the RTL (Register Transfer Language) and gate-level netlists. The final verified ASIC LR core was fabricated in 0.13 \( \mu \)m IBM 1P/8M CMOS technology using Artisan standard library cells. The die micrograph for the LR core is shown in Fig. 4.17. To the best of our knowledge, this represents the world’s first fabricated LR core.
Figure 4.17: Die Micrograph of HOLLL Lattice Reduction 0.13µm CMOS Core.

Figure 4.18: Test Setup using Verigy 93000 SoC tester, Temptronic TP04300 thermal forcing unit and the DUT (LR Core). The chip package leads make connection to the circuit board pads via elastomers, which are used to accommodate small deviations in the pitch and height of circuit board pads. (An elastomer is a sheet of insulating rubber housing multiple rows of symmetrically embedded metal filaments which form electrical connections.)
4.6.1 Test Results

The fabricated LR core was tested using an Agilent (Verigy) 93000 System on Chip (SoC) high-speed digital tester and a Temptronic TP04300 thermal forcing unit. The test setup consisted of the 93K SoC tester, the thermal forcing unit, a load board holding the DUT (Device Under Test). This setup is shown in Fig. 4.18. The nominal supply voltage supplied to the LR core is 1.2V, while the I/O voltage is 2.7V. The operation of the LR core was verified by passing input vectors at different SNR values to the chip through the tester and comparing the detector outputs with the expected values from the bit-true simulations both from MATLAB and ModelSim simulations. The BER performance of the LR core was measured as follows:

1. A sequence of complex-valued random Gaussian channel matrices was generated in MATLAB. The channel matrix was updated every four channel uses and used to transmit the symbol vectors.

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

3. The input channel matrix and received symbols were converted to their fixed-point representations based on the desired bit resolution.

4. The fixed-point input channel matrix and received symbols were supplied to an accurate fixed-point model of the LR core, which then generated the expected fixed-point outputs represented by an output bitstream.

5. The fixed-point input channel matrix and received symbols in addition to all required control signals were also combined into a test vector using MATLAB.

6. The generated test vector was then converted to a VCD (Value Change Dump) file using ModelSim.

7. The V93K_TestGenerator tool [56] was then used to convert the test vector VCD file to pin configuration files (“.pin”), timing files (“.tim”), and binary test vector files (“.binl”) required for testing.

8. These files were loaded into the V93K SoC tester and were used to supply input test vector to the LR chip.
9. The core supply voltage and I/O supply voltage are set appropriately. This was performed by specifying a voltage level file (“.lev”) in the V93K SoC tester environment. This file is also used to control current limits.

10. An at-speed test was run on the LR chip and the outputs were compared against the expected bit stream generated by the MATLAB simulations.

Fig. 4.19 below shows the measured BER results (at a clock rate of 204MHz at 1.32V supply and 25°C) with respect to the ideal ML detector and the floating-point simulation of HOLLL. The detection algorithm used in conjunction with HOLLL is the near-ML K-Best with $K=10$ and $4 \times 4$ 64-QAM MIMO. For high SNR values, the LR core offers a dramatic performance enhancement when applied to K-Best. For a BER of $10^{-4}$, the proposed LR design achieves an SNR gain of 4.1dB compared to the K-Best detector. This large SNR gain signifies the importance of LR as part of future MIMO detection systems. It can also be observed that the LR core starts to improve the detection performance when the SNR is increased beyond 31dB. Thus, this SNR value can be used as the threshold for enabling/disabling the LR by-pass mode of our proposed LR core.

Figure 4.19: Measured BER at 510Mb/s for Hardware-Optimized LLL (HOLLL) with $4 \times 4$ 64-QAM K-Best Real ($K=10$).
Fig. 4.20 above shows a Shmoo plot depicting the measured maximum operating frequency and total power dissipation of our LR chip versus the supply voltage at 25°C. A total of five chips were tested, where the average, maximum and minimum frequency values are also shown in Fig. 4.20. At 25°C and 1.32V supply voltage, the proposed LR core operates at a clock rate of 204MHz while dissipating 59.4mW. The temperature was forced to 25°C using the Temptronic TP04300 thermal forcing unit. Using this thermal forcing unit, the temperature was also forced to 0°C and 80°C, which resulted in clock rates of 208MHz and 185MHz, while dissipating 58.7mW and 56.8mW at 1.32V supply, respectively. The complete chip measurement results are presented in Appendix B.

### 4.6.2 Design Comparison

Table 4.2 provides an overview and comparison of reported LR implementations for $4 \times 4$ MIMO systems, including this work. The applied LR algorithm, the implementation platform (e.g. FPGA or ASIC), as well as the performance specifications of each design are listed. If information on other designs is not provided by the authors or not applicable (e.g. number of FPGA slices for ASIC designs), this is indicated by the entry N/A.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LR Algorithm</td>
<td>Brun’s CLLL ELLL Clarkson RS-LLL HOLLL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BER Performance</td>
<td>Loss ML-diversity ML-diversity Loss Loss ML-diversity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td>0.25µm Virtex-5 Virtex-5 Virtex-II Pro 0.13µm 0.13µm 65nm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core Area [mm²]</td>
<td>1.27 N/A N/A N/A 0.925 1.04 0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gate Count [kGE]</td>
<td>53 78.7 64.7 N/A 107 125¹ 193</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slices</td>
<td>N/A 1,712 1,369 7,349 N/A N/A N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock Freq. [MHz]</td>
<td>44.6 163 205 100 333 204 833</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycles/Matrix</td>
<td>3.1avg 130avg 102avg 420avg 14avg 40/fixed 40/fixed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time/Matrix [ns]</td>
<td>69.5avg 797.5avg 497.6avg 4200avg 42.0avg 196.1/fixed 48.0/fixed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput²[Mbps]</td>
<td>111.5 120 193 23 832.5 510 2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Throughput</td>
<td>Yes Yes Yes Yes Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tested Chip</td>
<td>No No No No Yes No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹ One Gate Equivalent (GE) in 0.13µm and 65nm CMOS technology corresponds to the area of their respective 2-input drive-two NAND gates.

² Throughput results are based on LR-aided MIMO detection using 64-QAM, with the detector unrestricted in its ability to match the LR block in terms of its Cycles/Matrix throughput.
By examining the design comparison shown Table 4.2, the following points can be inferred:

1. The LR algorithm implemented in [15] is an optimized version of Brun’s algorithm. Other than this implementation, the LR designs in the literature have focused on implementing variants of the CLLL algorithm, due to its efficiency (unlike Seyesn’s algorithm) and its potential to achieve ML diversity (unlike Brun’s algorithm). The design in [16] implements CLLL close to its ideal form. The design in [16] implements Effective LLL (ELLL), which is CLLL with the exception of only performing size reduction against one matrix column (i.e. only size reduces against column $k-1$). This is also referred to as early-termination of size reduction. Clarkson’s algorithm, implemented in [18], is CLLL with the exception of replacing the Lovász swapping criterion with the simpler Siegel criterion. The design in [19] combines ELLL and a reverse-order Siegel criterion to produce an algorithm known as Reverse-Siegel LLL (RS-LLL).

2. “ML-Diversity” refers to an LR-aided BER performance result capable of achieving ML diversity (within the limits of fixed-point arithmetic and a finite number of iterations), which is the primary goal of lattice reduction. “Loss” refers to a significant deviation from ML diversity. The results shown are based on BER performance results provided by the authors of each implementation. Unlike the majority of LR implementations, the LR design in this thesis has a significant advantage of achieving high throughput while maintaining ML diversity. In fact, the proposed design represents the only ASIC implementation of LR capable of achieving ML diversity.

3. For the throughput calculation, it is assumed that a MIMO detection method in 64-QAM follows the LR block and is unrestricted in its ability to match the LR block in terms of its Cycles/Matrix throughput. Based on this, the resulting bit rates were calculated and included in Table 4.2.

4. The proposed LR design in this work has a significant advantage of being the only design with fixed throughput, independent of the correlation of the channel matrix. This is due to the design completing all computations in a fixed number of cycles (i.e. fixed processing latency). For the other reported LR implementations, the throughput and processing latency results represent an average, since their exact number of
required cycles depends on the correlation of the input matrix $R$. Clearly, this variability is highly undesirable for a practical implementation of LR as a preprocessing step in a larger MIMO detection framework.

5. A very recent LR implementation is the design presented in [19]. It offers the highest throughput (333 MHz) and the lowest number of average cycles per LR-reduced matrix (14). However, the tradeoff is that the design relies on limiting of the number of column swaps in the LLL reduction process to only 4, which leads to a significant deviation from ML diversity as can be observed in the BER performance results in [19]. Another way in which the authors are able to achieve a very low average of 14 cycles, is by immediately terminating an iteration if the swapping condition is false. This leads to a low average of cycles per iterations due to the frequent by-passing of the cycles required for basis updating. The disadvantage is that this approach leads to a variable number of cycles per iteration depending on the result of the swapping condition. However, in the LR design proposed in this thesis, if the swapping condition is false, then the stage controller continues to wait until the 40 cycles complete before transferring the matrix values to the next LR stage. This is necessary in order to maintain a fixed throughput.

6. Compared to other LR designs, the LR implementation in this work exhibits a larger gate count. However, it is important to note that all the other LR designs are based on a single LR stage which is reused to perform all the LR iterations. As a result, these designs cannot accept any new matrix inputs until the current matrix is fully reduced, which depends on the correlation of the matrix. On the other hand, the design in this work consists of a pipelined architecture in which each of the 9 iterations are performed independently, resulting in the ability to accept new matrix data every 40 cycles. In fact, if the per-iteration area of the proposed implementation is compared to other LR designs, it is found the design in this work only occupies 13.9 kGE (kilo Gate Equivalent), which is significantly lower than any other reported LR implementation. The reason behind the ability to achieve this small area is due to adopting the Hardware-Optimized LLL (HOLLL) algorithm which eliminates the need for multipliers, dividers and square root blocks, which are very area intensive and which other designs rely on for performing lattice reduction.

7. The proposed LR design in 0.13$\mu$m achieves a data rate of 510Mbps, which satisfies
the requirements for the majority of applications of next generation wireless systems. It is interesting to see how this throughput scales with more advanced CMOS technologies, such as 65nm and 45nm CMOS, and whether the milestone of 1Gbps can be achieved. Towards this goal, the design was synthesized in 65nm CMOS. Post-synthesis results indicate a very high throughput of 2Gbps, which is the highest reported throughput for any reported LR design. This high throughput guarantees compatibility with the peak data rates for the most demanding applications (e.g. high speed transmission of rich multimedia) of future 4G wireless standards, such as WiMaX (IEEE 802.16m) and LTE-Advanced\(^9\).

8. To the best of our knowledge, the proposed LR core in this work is the only LR ASIC implementation design in the literature that has been fabricated and tested. The results from previous designs are either from FPGA implementations ([16–18]) or post-synthesis ASIC designs ([15, 19]). Therefore, compared to other reported results, the chip characteristics of the proposed LR core represent silicon proven results which offer a high level of confidence towards including the design as an IP block in a larger MIMO receiver chip.

### 4.7 Summary

A novel high-throughput silicon implementation of the proposed hardware-optimized LLL algorithm was presented in this chapter. The design relies on a highly pipelined architecture and optimized scheduling to achieve a low processing latency of 40 cycles. Moreover, the 40 cycle processing latency is constant, thus leading to a fixed throughput that is independent of the channel correlation. Furthermore, when combined with the K-Best detector, the BER performance is greatly enhanced (a gain of 4.1dB for a BER of $10^{-4}$). In 0.13\(\mu m\) CMOS, the fabricated LR chip has a maximum clock rate of 204 MHz while consuming 59.4 mW at 1.32V supply voltage. Moreover, important controllable features were included in the design to allow adjusting the LR \(\delta\) factor as well as enabling/disabling the LR operation through an optional by-pass mode. Table 4.3 summarizes the design characteristics and measured results of the proposed LR core, which is the first LR ASIC chip in

---

\(^9\)The highest proposed peak data rate for IEEE 802.16m is 1Gbps which targets super high-speed multimedia applications [4]. Meanwhile, the LTE-Advanced standard aims at 1Gbps for downlink and 500Mbps for uplink [6].
the literature that has been fabricated and tested; thus making it the first silicon-proven LR design.

Table 4.3: Summary of LR Design Characteristics and Measured Results

<table>
<thead>
<tr>
<th>LR Algorithm</th>
<th>HOLLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMO Configuration</td>
<td>$4 \times 4$</td>
</tr>
<tr>
<td>Channel Dependent</td>
<td>No</td>
</tr>
<tr>
<td>Domain</td>
<td>Complex</td>
</tr>
<tr>
<td>Word length ($R, Z, T$)</td>
<td>16, 16, 7</td>
</tr>
<tr>
<td>CMOS Process</td>
<td>$0.13 \mu m$ 1P/8M</td>
</tr>
<tr>
<td>Core Area</td>
<td>1.04 mm$^2$</td>
</tr>
<tr>
<td>Total Area</td>
<td>2 mm$^2$</td>
</tr>
<tr>
<td>Gate Count</td>
<td>125 kGE</td>
</tr>
<tr>
<td>Package</td>
<td>CFP120</td>
</tr>
<tr>
<td>Core Supply</td>
<td>1.32 V</td>
</tr>
<tr>
<td>I/O Supply</td>
<td>2.7 V</td>
</tr>
<tr>
<td>Max. Clock Freq.</td>
<td>204 MHz</td>
</tr>
<tr>
<td>Temperature (@ max. clock freq.)</td>
<td>25°C</td>
</tr>
<tr>
<td>Power (@ max. clock freq.)</td>
<td>59.4 mW</td>
</tr>
<tr>
<td>Cycles/Matrix</td>
<td>40</td>
</tr>
<tr>
<td>Latency</td>
<td>320 Clock Cycles</td>
</tr>
<tr>
<td>Features</td>
<td>LR By-Pass Mode and Variable $\delta$ Factor</td>
</tr>
<tr>
<td>Fabricated and Tested Chip</td>
<td>Yes</td>
</tr>
</tbody>
</table>
5 Conclusions and Future Work

5.1 Conclusions

Developing a low complexity high-throughput lattice reduction block for large antenna configurations has been a significant challenge in the literature. To address this issue, this thesis proposed a novel hardware-optimized version of the popular LLL algorithm. Through an extensive study, this work shows that complex LLL (CLLL) is the most efficient algorithm compared to other LR algorithms such as Seysen and Brun’s algorithms. Following this result, optimizations were introduced to CLLL that eliminated the need for multipliers, dividers and square root blocks. The proposed algorithm, which relies only on adders, comparators and shifters, achieves a 70% complexity reduction compared to CLLL. Moreover, it was shown that for LR-aided MIMO detection, the proposed LR algorithm outperforms CLLL in the practical case where the number of LR iterations is limited.

Using the proposed Hardware-Optimized LLL algorithm (HOLLL), a high-throughput ASIC implementation of an LR core for $4 \times 4$ MIMO systems was realized. The proposed VLSI design uses a deeply pipelined architecture to maximize throughput while minimizing hardware requirements. In addition, an optimized schedule was employed to achieve a fixed low processing latency of 40 cycles, which results in a fixed throughput that is independent of the channel correlation. Furthermore, when combined with the K-Best detector the BER performance is greatly improved (a gain of 4.1dB for a BER of $10^{-4}$). The LR core was fabricated in 0.13$\mu$m CMOS and achieves a maximum clock rate of 204 MHz, which allows it to sustain a throughput of 510Mbps. The designed LR chip represents the first fabricated and tested LR chip in the literature. When ported to 65nm, the design achieves a throughput of 2Gbps, thus easily satisfying the most demanding throughput requirements of next-generation WiMAX and LTE-Advanced wireless systems.
5 Conclusions and Future Work

5.2 Future Work

5.2.1 Data Storage Handling

The pipelined design of LR has significant memory requirements, as a result of the multiple matrices involved in the reduction process (i.e. $\tilde{R}$, $\tilde{Z}$ and $T$) as well as the numerous LR stages needed to implement the LR iterations. In the proposed LR design, register banks were used to store the matrices operated on in each LR stage, which resulted in a per-iteration gate count of 13.9 kGE and a total of 125 kGE for the entire LR core. For MIMO systems with very limited area resources, it would be interesting to explore a different data storage strategy. For example, embedded SRAM memory offers the potential to significantly reduce the matrix storage area, however, the tradeoff being the more complex memory control in addition to data access speed constraints. Exploring tradeoffs of various memory strategies for pipelined LR implementation remains open for research.

5.2.2 Design for Various Rates of Channel Variation

In this thesis, it was assumed that the channel is quasi-static and updated every four channel uses (i.e. updated once for every four received symbol vectors). However, it is possible that the rate of channel variation be slower or faster than this rate. Strategies for adapting the proposed LR core for both situations are discussed below.

Fast Varying Channels

For channels with very rapid variation, a channel use of two might be required. As a result, the throughput requirement for LR will be doubled, since the channel needs to be LR-reduced in half the number of cycles. For the proposed LR implementation, this would result in a processing latency requirement of 20 cycles versus 40 cycles. The 40 cycles achieved by the proposed design comes primarily from the Basis Update step, in which the CORDIC pipeline is used to perform the three required vectoring operations and the corresponding rotation operations for each. Even with the use of multiple CORDIC pipelines to perform the rotations in parallel, the latency can only be reduced to 37 cycles, since the three vectoring operations must be performed serially due to data dependencies.

Despite the above CORDIC pipeline limitations, there are two potential strategies for achieving the lower 20 cycle processing latency. The first would be to realize that based on the much lower complexity of the proposed LR design (compared to a MIMO detector...
such as K-Best), the LR core can actually be run at twice the clock frequency of that used for the detector. This effectively allows the LR core to maintain a processing latency of 40 cycles, which translates to 20 cycles with respect to the MIMO detection clock rate. In case providing a clock with twice the detection frequency rate is not feasible in the MIMO receiver, then the other strategy would be to alter the CORDIC pipeline to operate on both edges of the clock. This would half the latency of the CORDIC operations, which dominate the overall LR processing latency. Using both clock edges for CORDIC operations has been employed in the QR decomposition in [54], and the same strategy can be applied to the proposed LR core.

**Slow Varying Channels**

For channels with slow rates of variation, the LR core would not require the high throughput of 510 Mbps achieved by the proposed design at a clock frequency of 204 MHz. In fact, a much lower throughput would be sufficient, thus leading to a significant reduction in the required clock frequency. Despite this, the proposed LR core, which was designed to maximize throughput, can in fact be very advantageous for such applications with slow rates of channel variations. This is because the lower clock frequencies can be sustained at reduced power supply voltages. This results in quadratic reduction of the power consumption with respect to supply voltage, which is especially critical for mobile applications. For example, a power supply of 0.8 V is sufficient for achieving a clock frequency of 68 MHz, and the resulting power consumption becomes only 8.4 mW. Therefore, for future application of the LR core to slowly varying channels, a study can be performed on the relationship between the rate of channel variation and the minimum required clock frequency (and hence the minimum power supply voltage that could be used). This study will help system designers understand the power requirements of the LR core for various channel conditions.

**5.2.3 Integration of LR with MIMO Receiver Blocks**

Chapter 3 showed how LR can be used to significantly reduce the complexity of the K-Best detector. For example, LR-aided K-Best with \( K = 2 \) has equivalent performance as with \( K = 10 \). As a result, by considering of the impact of LR, different design strategies can be explored for low complexity MIMO detection. Consequently, the joint implementation of LR and K-Best is an interesting area of investigation. Another possibility for integration is
investigating an efficient joint implementation of QR and LR as the preprocessing blocks for the K-Best detector. Note that within the MIMO receiver system, a separate SNR measurement block can be used to dynamically enable and disable the LR by-pass mode provided by the LR core proposed in this work.

5.2.4 Exploitation of LR in MIMO System Design

LR techniques such as LLL, offer a practical solution to achieve ML diversity when combined with MIMO detectors. This opens the door for exploring new ways of designing MIMO systems that exploit diversity, since any previous loss of diversity due to the correlation between diversity branches can now be recovered through LR. Therefore, with the realization of the impact of LR, other MIMO system design considerations can be given priority over reducing correlation.
A Proposed LR Core - Input/Output Data Schedule

The input and output schedules for the proposed LR core are shown in Table A.1. All input/output operations are completed within 40 cycles to sustain the 40 cycle processing latency requirement. On the input side, there are two sets of eight pads for the real and imaginary matrix entries. Each entry of \( R \) and \( Z \) is 16 bits, therefore they are entered via the eight pads on both edges of the clock. The 8 LSBs are entered first on the positive edge, followed by the 8 MSBs on the negative edge. The total number of real entries is 26 (10 real \( R \) entries and 16 real \( Z \) entries), thus requiring 26 cycles to complete. Meanwhile, the total number of imaginary entries is 22 (6 imaginary \( R \) entries and 16 imaginary \( Z \) entries), thus requiring 22 cycles to complete. The exact sequence of outputs was chosen to match the output sequence of the QR design in [54], since the LR operation follows the QR block in the MIMO receiver as shown in Fig. 2.1 in Chapter 2.

On the output side, there are three sets of pads for the LR-reduced matrices \( \tilde{R} \) and \( \tilde{Z} \) as well as the transformation matrix \( T \). The number of pads for each is 8, 16 and 7, respectively. Since the word length of \( \tilde{R} \) entries is 16 bits, therefore the outputs are divided into 8 LSBs (lower byte) and 8 MSBs (upper byte) and the two bytes are outputted on consecutive positive clock edges. There is a total of 26 \( \tilde{R} \) outputs (16 real and 10 imaginary), therefore all outputs can be performed within the 40 cycles and there is no need to use both edges of the clock. As for the other two output matrices, \( \tilde{Z} \) and \( T \), each has 32 entries. Since the number of pads allocated to them matches their word lengths, therefore all outputs are complete in 32 clock cycles, well within the 40 cycle requirement.
The inputs into Re_in[7:0] and Im_in[7:0] are shown as 16 bits in the table since they are entered over both edges of the clock. The eight least significant bits (LSB) are entered on the positive clock edges while the eight most significant bits (MSB) are entered on the negative edges.

Table A.1: LR Core - Input/Output Schedule

<table>
<thead>
<tr>
<th>Cycle Number</th>
<th>Re_in[7:0]</th>
<th>Im_in[7:0]</th>
<th>R_out[7:0]</th>
<th>Z_out[15:0]</th>
<th>T_out[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16'h0000</td>
<td>16'h0000</td>
<td>R_{Re}^{2,4}[15:8]</td>
<td>Z_{Re}^{4,1}</td>
<td>T_{Re}^{1,4}</td>
</tr>
<tr>
<td>2</td>
<td>R_{Re}^{1,1}</td>
<td>16'h0000</td>
<td>R_{Re}^{2,4}[7:0]</td>
<td>Z_{Im}^{4,1}</td>
<td>T_{Im}^{1,4}</td>
</tr>
<tr>
<td>3</td>
<td>R_{Re}^{1,2}</td>
<td>R_{Im}^{1,2}</td>
<td>R_{Im}^{2,4}[15:8]</td>
<td>Z_{Re}^{4,2}</td>
<td>T_{Re}^{2,4}</td>
</tr>
<tr>
<td>4</td>
<td>R_{Re}^{2,2}</td>
<td>16'h0000</td>
<td>R_{Im}^{2,4}[7:0]</td>
<td>Z_{Re}^{4,2}</td>
<td>T_{Im}^{2,4}</td>
</tr>
<tr>
<td>5</td>
<td>R_{Re}^{1,3}</td>
<td>R_{Im}^{1,3}</td>
<td>R_{Im}^{1,4}[15:8]</td>
<td>Z_{Re}^{4,3}</td>
<td>T_{Re}^{3,4}</td>
</tr>
<tr>
<td>6</td>
<td>R_{Re}^{2,3}</td>
<td>R_{Im}^{2,3}</td>
<td>R_{Im}^{1,4}[7:0]</td>
<td>Z_{Re}^{4,3}</td>
<td>T_{Im}^{3,4}</td>
</tr>
<tr>
<td>7</td>
<td>R_{Re}^{1,4}</td>
<td>R_{Im}^{1,4}</td>
<td>R_{Im}^{1,4}[15:8]</td>
<td>Z_{Re}^{4,4}</td>
<td>T_{Re}^{4,4}</td>
</tr>
<tr>
<td>8</td>
<td>Z_{Re}^{1,1}</td>
<td>Z_{Im}^{1,1}</td>
<td>R_{Im}^{1,4}[7:0]</td>
<td>Z_{Re}^{4,4}</td>
<td>T_{Im}^{4,4}</td>
</tr>
<tr>
<td>9</td>
<td>Z_{Re}^{1,2}</td>
<td>Z_{Im}^{1,2}</td>
<td>R_{Re}^{3,4}[15:8]</td>
<td>Z_{Re}^{5,1}</td>
<td>T_{Re}^{5,1}</td>
</tr>
<tr>
<td>10</td>
<td>Z_{Re}^{1,3}</td>
<td>Z_{Im}^{1,3}</td>
<td>R_{Im}^{3,4}[7:0]</td>
<td>Z_{Re}^{5,3}</td>
<td>T_{Re}^{5,3}</td>
</tr>
<tr>
<td>11</td>
<td>Z_{Re}^{1,4}</td>
<td>Z_{Im}^{1,4}</td>
<td>R_{Im}^{3,4}[15:8]</td>
<td>Z_{Re}^{5,2}</td>
<td>T_{Re}^{5,2}</td>
</tr>
<tr>
<td>12</td>
<td>R_{Re}^{2,4}</td>
<td>R_{Im}^{2,4}</td>
<td>R_{Im}^{3,4}[7:0]</td>
<td>Z_{Re}^{5,2}</td>
<td>T_{Re}^{5,2}</td>
</tr>
<tr>
<td>13</td>
<td>Z_{Re}^{2,1}</td>
<td>Z_{Im}^{2,1}</td>
<td>R_{Re}^{4,4}[15:8]</td>
<td>Z_{Re}^{5,3}</td>
<td>T_{Re}^{5,3}</td>
</tr>
<tr>
<td>14</td>
<td>Z_{Re}^{2,2}</td>
<td>Z_{Im}^{2,2}</td>
<td>R_{Re}^{4,4}[7:0]</td>
<td>Z_{Re}^{5,3}</td>
<td>T_{Re}^{5,3}</td>
</tr>
<tr>
<td>15</td>
<td>Z_{Re}^{2,3}</td>
<td>Z_{Im}^{2,3}</td>
<td>R_{Re}^{3,3}[15:8]</td>
<td>Z_{Re}^{5,4}</td>
<td>T_{Re}^{5,4}</td>
</tr>
<tr>
<td>16</td>
<td>Z_{Re}^{2,4}</td>
<td>Z_{Im}^{2,4}</td>
<td>R_{Re}^{3,3}[7:0]</td>
<td>Z_{Re}^{5,4}</td>
<td>T_{Re}^{5,4}</td>
</tr>
<tr>
<td>17</td>
<td>16'h0000</td>
<td>16'h0000</td>
<td>R_{Re}^{2,3}[15:8]</td>
<td>Z_{Re}^{2,1}</td>
<td>T_{Re}^{1,2}</td>
</tr>
<tr>
<td>18</td>
<td>16'h0000</td>
<td>16'h0000</td>
<td>R_{Re}^{2,3}[7:0]</td>
<td>Z_{Re}^{2,1}</td>
<td>T_{Re}^{1,2}</td>
</tr>
<tr>
<td>19</td>
<td>16'h0000</td>
<td>16'h0000</td>
<td>R_{Im}^{2,3}[15:8]</td>
<td>Z_{Re}^{2,2}</td>
<td>T_{Re}^{2,2}</td>
</tr>
<tr>
<td>20</td>
<td>16'h0000</td>
<td>16'h0000</td>
<td>R_{Im}^{2,3}[7:0]</td>
<td>Z_{Re}^{2,2}</td>
<td>T_{Re}^{2,2}</td>
</tr>
</tbody>
</table>

1 The cycle numbers presented are based on the number of positive edges of the clock. The inputs into Re_in[7:0] and Im_in[7:0] are entered over both edges of the clock. The eight least significant bits (LSB) are entered on the positive clock edges while the eight most significant bits (MSB) are entered on the negative edges.
<table>
<thead>
<tr>
<th>Cycle Number</th>
<th>Re_in[7:0]</th>
<th>Im_in[7:0]</th>
<th>R_out[7:0]</th>
<th>Z_out[15:0]</th>
<th>T_out[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{1,3}^{Re}[15 : 8]</td>
<td>Z_{2,3}^{Re}</td>
<td>T_{3,2}^{Re}</td>
</tr>
<tr>
<td>22</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{1,3}^{Re}[7 : 0]</td>
<td>Z_{2,3}^{Im}</td>
<td>T_{3,2}^{Im}</td>
</tr>
<tr>
<td>23</td>
<td>R_{3,3}^{Re}</td>
<td>16’h0000</td>
<td>R_{1,3}^{Im}[15 : 8]</td>
<td>Z_{2,3}^{Re}</td>
<td>T_{4,2}^{Re}</td>
</tr>
<tr>
<td>24</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{1,3}^{Im}[7 : 0]</td>
<td>Z_{2,4}^{Im}</td>
<td>T_{4,2}^{Im}</td>
</tr>
<tr>
<td>25</td>
<td>R_{3,4}^{Re}</td>
<td>R_{3,4}^{Im}</td>
<td>R_{2,2}^{Re}[15 : 8]</td>
<td>Z_{1,1}^{Re}</td>
<td>T_{1,1}^{Re}</td>
</tr>
<tr>
<td>26</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{2,2}^{Re}[7 : 0]</td>
<td>Z_{1,1}^{Im}</td>
<td>T_{1,1}^{Im}</td>
</tr>
<tr>
<td>27</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{1,2}^{Re}[15 : 8]</td>
<td>Z_{1,2}^{Re}</td>
<td>T_{1,2}^{Re}</td>
</tr>
<tr>
<td>28</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{1,2}^{Re}[7 : 0]</td>
<td>Z_{1,2}^{Im}</td>
<td>T_{1,2}^{Im}</td>
</tr>
<tr>
<td>29</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>R_{1,2}^{Im}[15 : 8]</td>
<td>Z_{1,3}^{Re}</td>
<td>T_{3,1}^{Re}</td>
</tr>
<tr>
<td>30</td>
<td>Z_{3,1}^{Re}</td>
<td>Z_{3,1}^{Im}</td>
<td>R_{1,2}^{Im}[7 : 0]</td>
<td>Z_{1,3}^{Im}</td>
<td>T_{3,1}^{Im}</td>
</tr>
<tr>
<td>31</td>
<td>Z_{3,2}^{Re}</td>
<td>Z_{3,2}^{Im}</td>
<td>R_{1,1}^{Re}[15 : 8]</td>
<td>Z_{1,4}^{Re}</td>
<td>T_{1,4}^{Re}</td>
</tr>
<tr>
<td>32</td>
<td>Z_{3,3}^{Re}</td>
<td>Z_{3,3}^{Im}</td>
<td>R_{1,1}^{Im}[7 : 0]</td>
<td>Z_{1,4}^{Im}</td>
<td>T_{1,4}^{Im}</td>
</tr>
<tr>
<td>33</td>
<td>Z_{3,4}^{Re}</td>
<td>Z_{3,4}^{Im}</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>34</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>35</td>
<td>16’h0000</td>
<td>16’h0000</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>36</td>
<td>R_{4,4}^{Re}</td>
<td>16’h0000</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>37</td>
<td>Z_{4,1}^{Re}</td>
<td>Z_{4,1}^{Im}</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>38</td>
<td>Z_{4,2}^{Re}</td>
<td>Z_{4,2}^{Im}</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>39</td>
<td>Z_{4,3}^{Re}</td>
<td>Z_{4,3}^{Im}</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
<tr>
<td>40</td>
<td>Z_{4,4}^{Re}</td>
<td>Z_{4,4}^{Im}</td>
<td>8’h00</td>
<td>16’h0000</td>
<td>7’h00</td>
</tr>
</tbody>
</table>
B Proposed LR Core - Detailed Measurement Results

This Appendix presents the test results from five working LR ASIC chips at 0°C, 25°C and 80°C. Tables B.1 to B.15 show the detailed measurement results of all five chips in terms of maximum operating frequency and power consumption, for various supply voltages.
### Table B.1: Measurement Results for Chip #1 @ 0°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)&lt;sup&gt;1&lt;/sup&gt;</td>
<td>14.8</td>
<td>10.6</td>
<td>8.4</td>
<td>7.2</td>
<td>5.7</td>
<td>4.8</td>
</tr>
<tr>
<td>f (MHz)&lt;sup&gt;2&lt;/sup&gt;</td>
<td>68</td>
<td>94</td>
<td>119</td>
<td>140</td>
<td>175</td>
<td>208</td>
</tr>
<tr>
<td>I (mA)&lt;sup&gt;3&lt;/sup&gt;</td>
<td>10.5</td>
<td>15.9</td>
<td>21.8</td>
<td>26.5</td>
<td>35.6</td>
<td>44.5</td>
</tr>
<tr>
<td>P (mW)&lt;sup&gt;4&lt;/sup&gt;</td>
<td>8.4</td>
<td>14.3</td>
<td>21.8</td>
<td>28.6</td>
<td>42.7</td>
<td>58.7</td>
</tr>
</tbody>
</table>

<sup>1</sup> t: clock period.  
<sup>2</sup> f: clock frequency.  
<sup>3</sup> I: core current.  
<sup>4</sup> P: core power @ supply voltage.

### Table B.2: Measurement Results for Chip #1 @ 25°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.8</td>
<td>10.7</td>
<td>8.5</td>
<td>7.2</td>
<td>5.8</td>
<td>4.9</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>68</td>
<td>93</td>
<td>118</td>
<td>139</td>
<td>172</td>
<td>204</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.5</td>
<td>16</td>
<td>21.7</td>
<td>26.9</td>
<td>35.3</td>
<td>45</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.4</td>
<td>14.4</td>
<td>21.7</td>
<td>29.1</td>
<td>42.4</td>
<td>59.4</td>
</tr>
</tbody>
</table>

### Table B.3: Measurement Results for Chip #1 @ 80°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.9</td>
<td>10.8</td>
<td>8.5</td>
<td>7.4</td>
<td>6.1</td>
<td>5.4</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>67</td>
<td>93</td>
<td>118</td>
<td>135</td>
<td>164</td>
<td>185</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.7</td>
<td>16.3</td>
<td>22.3</td>
<td>27</td>
<td>35.8</td>
<td>43</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.6</td>
<td>14.7</td>
<td>22.3</td>
<td>29.2</td>
<td>43</td>
<td>56.8</td>
</tr>
</tbody>
</table>
Table B.4: Measurement Results for Chip #2 @ 0°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.5</td>
<td>10.4</td>
<td>8.2</td>
<td>6.9</td>
<td>5.7</td>
<td>4.7</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>69</td>
<td>96</td>
<td>122</td>
<td>145</td>
<td>176</td>
<td>213</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.7</td>
<td>16.1</td>
<td>22.3</td>
<td>26.9</td>
<td>36.2</td>
<td>45.8</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.6</td>
<td>14.5</td>
<td>22.3</td>
<td>29.1</td>
<td>43.4</td>
<td>60.5</td>
</tr>
</tbody>
</table>

Table B.5: Measurement Results for Chip #2 @ 25°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.7</td>
<td>10.6</td>
<td>8.3</td>
<td>7.1</td>
<td>5.8</td>
<td>4.8</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>68</td>
<td>94</td>
<td>121</td>
<td>142</td>
<td>172</td>
<td>208</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.7</td>
<td>16.4</td>
<td>22.7</td>
<td>27.7</td>
<td>36.5</td>
<td>45.7</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.6</td>
<td>14.8</td>
<td>22.7</td>
<td>29.9</td>
<td>43.8</td>
<td>60.3</td>
</tr>
</tbody>
</table>

Table B.6: Measurement Results for Chip #2 @ 80°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.8</td>
<td>10.7</td>
<td>8.5</td>
<td>7.3</td>
<td>6.1</td>
<td>5.3</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>68</td>
<td>94</td>
<td>118</td>
<td>137</td>
<td>164</td>
<td>189</td>
</tr>
<tr>
<td>I (mA)</td>
<td>11</td>
<td>16.8</td>
<td>23</td>
<td>27.7</td>
<td>36.3</td>
<td>44.1</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.8</td>
<td>15.1</td>
<td>23</td>
<td>29.9</td>
<td>43.6</td>
<td>58.2</td>
</tr>
</tbody>
</table>
Table B.7: Measurement Results for Chip #3 @ 0°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.5</td>
<td>10.3</td>
<td>8.1</td>
<td>6.8</td>
<td>5.6</td>
<td>4.6</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>69</td>
<td>97</td>
<td>123</td>
<td>147</td>
<td>180</td>
<td>217</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.7</td>
<td>16.4</td>
<td>21.9</td>
<td>27.3</td>
<td>36.9</td>
<td>46.2</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.6</td>
<td>14.8</td>
<td>21.9</td>
<td>29.5</td>
<td>44.3</td>
<td>61.0</td>
</tr>
</tbody>
</table>

Table B.8: Measurement Results for Chip #3 @ 25°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.7</td>
<td>10.3</td>
<td>8.2</td>
<td>6.9</td>
<td>5.7</td>
<td>4.7</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>68</td>
<td>97</td>
<td>122</td>
<td>145</td>
<td>175</td>
<td>213</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.5</td>
<td>16.5</td>
<td>22.3</td>
<td>27.9</td>
<td>35.9</td>
<td>45.2</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.4</td>
<td>14.9</td>
<td>22.3</td>
<td>30.1</td>
<td>43.1</td>
<td>59.7</td>
</tr>
</tbody>
</table>

Table B.9: Measurement Results for Chip #3 @ 80°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.6</td>
<td>11</td>
<td>8.2</td>
<td>7.1</td>
<td>5.84</td>
<td>5.2</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>68</td>
<td>91</td>
<td>122</td>
<td>141</td>
<td>171</td>
<td>192</td>
</tr>
<tr>
<td>I (mA)</td>
<td>11</td>
<td>16</td>
<td>22.8</td>
<td>27.2</td>
<td>36.9</td>
<td>44</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.8</td>
<td>14.4</td>
<td>22.8</td>
<td>29.4</td>
<td>44.3</td>
<td>58.1</td>
</tr>
</tbody>
</table>
Table B.10: Measurement Results for Chip #4 @ 0°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>15.5</td>
<td>11.2</td>
<td>8.7</td>
<td>7.5</td>
<td>5.9</td>
<td>5.1</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>65</td>
<td>89</td>
<td>115</td>
<td>134</td>
<td>170</td>
<td>198</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.2</td>
<td>15.5</td>
<td>21.5</td>
<td>26.4</td>
<td>35.8</td>
<td>44.1</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.2</td>
<td>14</td>
<td>21.5</td>
<td>28.5</td>
<td>43</td>
<td>58.2</td>
</tr>
</tbody>
</table>

Table B.11: Measurement Results for Chip #4 @ 25°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>15.3</td>
<td>10.85</td>
<td>8.7</td>
<td>7.5</td>
<td>6</td>
<td>5.2</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>65</td>
<td>92</td>
<td>115</td>
<td>133</td>
<td>167</td>
<td>192</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.4</td>
<td>16.3</td>
<td>21.5</td>
<td>26.4</td>
<td>35.5</td>
<td>44.9</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.3</td>
<td>14.7</td>
<td>21.5</td>
<td>28.5</td>
<td>42.6</td>
<td>59.3</td>
</tr>
</tbody>
</table>

Table B.12: Measurement Results for Chip #4 @ 80°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.7</td>
<td>11.1</td>
<td>8.8</td>
<td>7.7</td>
<td>6.3</td>
<td>5.5</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>68</td>
<td>90</td>
<td>114</td>
<td>130</td>
<td>159</td>
<td>182</td>
</tr>
<tr>
<td>I (mA)</td>
<td>11.1</td>
<td>16.2</td>
<td>22</td>
<td>26.3</td>
<td>35</td>
<td>43.9</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.9</td>
<td>14.6</td>
<td>22</td>
<td>28.4</td>
<td>42</td>
<td>58</td>
</tr>
</tbody>
</table>
Table B.13: Measurement Results for Chip #5 @ 0°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>15.3</td>
<td>11.3</td>
<td>8.6</td>
<td>7.4</td>
<td>6.0</td>
<td>5.2</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>66</td>
<td>89</td>
<td>117</td>
<td>135</td>
<td>168</td>
<td>192</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.4</td>
<td>15.4</td>
<td>21.8</td>
<td>26.2</td>
<td>36.1</td>
<td>43.2</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.3</td>
<td>13.9</td>
<td>21.8</td>
<td>28.3</td>
<td>43.3</td>
<td>57</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>15.1</td>
<td>11.2</td>
<td>8.6</td>
<td>7.5</td>
<td>6.1</td>
<td>5.3</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>66</td>
<td>89</td>
<td>116</td>
<td>134</td>
<td>164</td>
<td>191</td>
</tr>
<tr>
<td>I (mA)</td>
<td>10.5</td>
<td>15.6</td>
<td>22.1</td>
<td>26.3</td>
<td>35</td>
<td>43.9</td>
</tr>
<tr>
<td>P (mW)</td>
<td>8.4</td>
<td>14</td>
<td>22.1</td>
<td>28.4</td>
<td>42</td>
<td>58</td>
</tr>
</tbody>
</table>

Table B.15: Measurement Results for Chip #5 @ 80°C.

<table>
<thead>
<tr>
<th>Supply voltage</th>
<th>0.8 V</th>
<th>0.9 V</th>
<th>1.0 V</th>
<th>1.08 V</th>
<th>1.2 V</th>
<th>1.32 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>t (ns)</td>
<td>14.6</td>
<td>11</td>
<td>8.8</td>
<td>7.6</td>
<td>6.3</td>
<td>5.5</td>
</tr>
<tr>
<td>f (MHz)</td>
<td>69</td>
<td>91</td>
<td>114</td>
<td>132</td>
<td>159</td>
<td>182</td>
</tr>
<tr>
<td>I (mA)</td>
<td>11.2</td>
<td>16.4</td>
<td>22.3</td>
<td>27.3</td>
<td>35.1</td>
<td>43.7</td>
</tr>
<tr>
<td>P (mW)</td>
<td>9.0</td>
<td>14.8</td>
<td>22.3</td>
<td>29.5</td>
<td>42.1</td>
<td>57.7</td>
</tr>
</tbody>
</table>
References


