Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 267460, 14 pages
doi:10.1155/2008/267460

Research Article

A Practical, Hardware Friendly MMSE Detector for MIMO-OFDM-Based Systems

Hun Seok Kim,[1] Weijun Zhu,[2] Jatin Bhatia,[2] Karim Mohammed,[1] Anish Shah,[1] and Babak Daneshrad[1]

[1] *Wireless Integrated Systems Research (WISR) Group, Electrical Engineering Department, University of California, Los Angeles, CA 90095, USA*
[2] *Silvus Communication Systems Inc., 10990 Wilshire Blvd, Suite 440, Los Angeles, CA 90064, USA*

Correspondence should be addressed to Hun Seok Kim, kimhs@ucla.edu

Received 27 July 2007; Revised 7 December 2007; Accepted 19 February 2008

Recommended by Huaiyu Dai

Design and implementation of a highly optimized MIMO (multiple-input multiple-output) detector requires cooptimization of the algorithm with the underlying hardware architecture. Special attention must be paid to application requirements such as throughput, latency, and resource constraints. In this work, we focus on a highly optimized matrix inversion free 4×4 MMSE (minimum mean square error) MIMO detector implementation. The work has resulted in a real-time field-programmable gate array-based implementation (FPGA-) on a Xilinx Virtex-2 6000 using only 9003 logic slices, 66 multipliers, and 24 Block RAMs (less than 33% of the overall resources of this part). The design delivers over 420 Mbps sustained throughput with a small 2.77-microsecond latency. The designed 4×4 linear MMSE MIMO detector is capable of complying with the proposed IEEE 802.11n standard.

1. INTRODUCTION

Since the early work of Foschini, Gans, Teletar, and Paulraj [1–4] almost a decade ago, thousands of papers have been published in the area of MIMO-based information theory, algorithms, codes, medium access control (MAC), and so on. By and large these works have been theoretical/simulation-based and have focused on the algorithms and protocols that deliver superior bit error rate (BER) for a given signal-to-noise power ratio (SNR). Little attention has been paid to the actual implementation of such algorithms in real-time systems that look to deliver 100's of million bits per second (Mbps) and possibly Gbps (Giga-bps) sustained throughput to an end user or application. To better focus our efforts and make our research results more relevant with mainstream MIMO systems, we decided to set the following specifications for our MIMO detector.

(1) Throughput: ability to process a minimum of 14.4 M (million) 4×4 channel instances per second. It is equivalent to 345.6 Mbps when using 64QAM (quadrature-amplitude-modulation).

(2) Latency: the entire detector latency should be below $4\,\mu$s. This is an important consideration in systems that require fast physical layer turn around time in order to maintain overall system efficiency at the MAC.

(3) Hardware complexity: the design should be such that it could easily fit onto a low-end FPGA (i.e., Xilinx Virtex-2 3000) or occupy no more than 40% of the resources of a high-end FPGA.

To put the above requirements into perspective, consider the needs of an IEEE 802.11n system [5]. The $4\,\mu$s of latency corresponds to 1/4 of the short interframe spacing (SIFS) time ($16\,\mu$s for 802.11n [5]). The SIFS time is the maximum latency allowed for the decoding of a packet and the generation of the corresponding ACK (acknowledgement)/NACK (no ACK). The throughput of 14.4 M channel instances per second is required to complete the MIMO detection for 52 data subcarriers in a single OFDM (orthogonal frequency division multiplexing) symbol interval ($3.6\,\mu$s with short guard interval) [5]. The throughput requirement is also necessary to meet the strict SIFS requirement in 802.11n and to guarantee timely completion of the MMSE solution.

The hardware complexity needs to be bounded so that the MIMO detector could be integrated with the rest of the system. In our design, we made the decision that the MIMO detector complexity should not be greater than the rest of the system and that the entire 802.11n compliant 4×4 MIMO transceiver must fit onto a single Virtex-2 8000 FPGA [6]. This translates to an upper bound of 40% resource utilization for the MIMO detector in a single high-end FPGA. With the above requirements in mind, our literature search revealed 4 classes of solutions. These included stand alone matrix inversion ASICs (application-specific integrated circuit), maximum likelihood- (ML-) based detectors, V-BLAST- (vertical Bell laboratories layered space-time architecture-) based detectors, and linear MMSE detectors.

A number of ASIC-based matrix inversion ICs were reported in the past decades [7, 8], when stand alone signal processing ASICs were common place. In today's world of SoC's (system of a chip), the solutions exemplified by these references are no longer relevant, as these solutions invariably miss the latency, throughput, and size requirements of our desired solution. A more recent body of work is more explicitly focused on the implementation of the recently developed MIMO detector algorithms. These can be classified as ML-based detectors [9–12], V-BLAST-type detectors [13–16], and MMSE-based detectors [17–20]. These solutions, although interesting in concept, still fail to meet the stringent latency and throughput requirements of a practical system such as 802.11n.

The class of FPGA- or ASIC-based ML detectors for MIMO systems is exemplified in the works reported in [9–12]. Whereas [9] focuses on an exhaustive search optimal ML algorithm, the work in [10–12] focuses on the implementation of suboptimal ML solutions. The work reported in [9, 11, 12] achieves throughputs that are lower than 60 Mbps (equivalently 3.75 M channel instances per second with 16QAM). The implementation results for [10] show that the throughput of the design is not guaranteed to be constant since the design is based on a nondeterministic tree search. Although this chip delivers 170 Mbps average throughput at an SNR of 20 dB, its throughput is highly dependent on the channel condition and the minimum required throughput is not guaranteed. In addition, the design in [10] is incapable of supporting 64QAM. Finally, with the exception of [9], the ML MIMO detectors reported in [10–12] require extra hardware resources for QR decomposition as a preprocessing step. The QR decomposition block has comparable algorithmic complexity to an entire linear MMSE MIMO detector.

V-BLAST MIMO detection algorithms had been believed to be promising solutions due to their lower complexity compared to ML-based algorithms and their higher performance relative to linear MMSE algorithms in hard detection [13]. A novel Square-Root algorithm was introduced in [14] for reduced complexity V-BLAST detection and was later implemented on FPGA [15] and ASIC [16] platforms. However, the implementation results in [15, 16] show that throughputs of the designed V-BLAST detectors are 0.125 M and 1.56 M channel instances per second, respectively, which are much lower than our requirement of 14.4 M channel instances per second. Furthermore, recent studies on soft-output detectors revealed that the performance of the soft-output V-BLAST detector is inferior to soft-output linear MMSE detectors in systems that employ bit interleaved coded modulation (BICM) [21].

Prior work on linear MMSE MIMO detectors [17–20] has shown that these algorithms have significantly lower complexity than ML algorithms and their performance in MIMO-BICM systems is quite comparable to ML algorithms especially when the number of antennas or the constellation size is large [21].

The most computationally intensive part of a linear MMSE MIMO detector is the matrix inversion operation. Hence, the majority of the previous work had approached the linear MMSE detection problem by focusing on efficient matrix inversion techniques. An MMSE detector based on QR decomposition via CORDIC- (coordinate rotation digital computer-) based Givens rotations is studied and implemented in [17]. Similarly, square-root-free SGR (squared Givens rotation) algorithm-based MMSE detectors are reported in the literature [17, 19]. A linear MMSE detector using the Sherman-Morrison formula, a special case of the matrix inversion lemma, is given in [18]. In [20], an FPGA implementation of a QR-RLS- (Recursive Least Square) based linear MMSE MIMO detector is reported. However, every linear MMSE detector designed in [17–20] fails to satisfy the design requirements outlined above. Each design suffers from either excessive hardware resource usage [17, 18] or exorbitant latency [20] to invert multiple channel matrices. Moreover, none of these implementations is able to provide a matrix inversion throughput higher than 7 M channel instances per second.

Based on the results of the literature search and our early work, it was clear that the MMSE-based solutions were good candidates for achieving the target requirements. At the conclusion of our work, a real-time FPGA implementation of the MIMO detector was realized on a Xilinx Virtex-2 FPGA and was integrated into an end-to-end MIMO-OFDM testbed [6]. The resulting 4×4 MIMO detector uses 9003 logic slices, 66 multipliers, and 24 Block RAMs (less than 33% of the overall resources of this part). The design delivers over 420 Mbps sustained throughput, with a small 2.77 μs latency.

This paper is organized as follows. In Section 2, we describe an 802.11n compatible MIMO-OFDM transceiver and the linear MMSE MIMO detection problem for the system. In Section 3, we propose a realistic algorithm complexity measure which considers both the number of operations and their input operand bit precisions. In Section 4, we compare several types of linear MMSE MIMO detection algorithms such as QR decomposition-based Squared MMSE algorithms and Square-Root algorithms along with complexity analysis and numerical stability simulations. The purpose of this comparison is to identify the best algorithm for actual implementation on FPGAs. In order to enhance the numerical stability of the algorithm, we propose a dynamic scaling technique and show its impact on fixed point algorithm performances in Section 4. The modified and scaled Gram-Schmidt QR decomposition algorithm

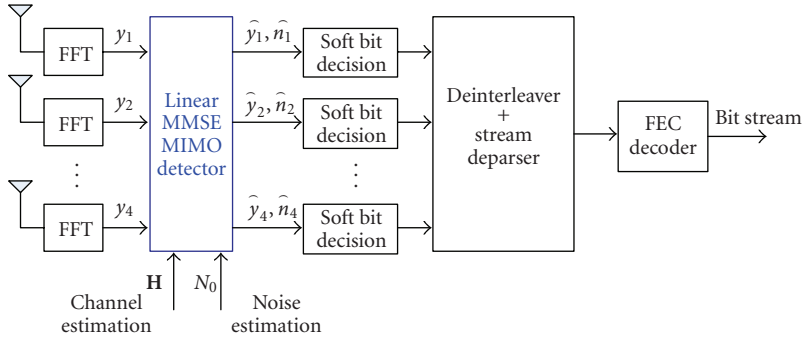


Figure 1: Receiver block diagram.

combined with Square-Root linear MMSE detection was selected and its hardware architecture is described in Section 5. The implementation results on Xilinx Virtex-2 and Virtex-4 FPGAs are presented in Sections 6 and 7 concludes the paper.

2. SYSTEM DESCRIPTION AND LINEAR MMSE MIMO DETECTION

We consider a linear MMSE MIMO detector a part of an entire 802.11n compliant 4×4 MIMO OFDM transceiver [6]. Figure 1 corresponds to the receiver block diagram.

We denote N as the number of transmit and receive antennas. For each subcarrier, we denote the $N \times 1$ received vector as $\mathbf{y}$ which is given in (1). Where $\mathbf{s}$ is the $N \times 1$ transmitted symbol vector, $\mathbf{H}$ is the $N \times N$ channel matrix, and $\mathbf{n}$ is the $N \times 1$ additive white Gaussian noise vector with covariance matrix $N_0 \cdot \mathbf{I}$;

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}. \tag{1}$$

In this paper, we focus on the MIMO detector block in Figure 1 which produces the linear MMSE solution $\widehat{\mathbf{y}}$, the estimate of the transmitted symbol vector $\mathbf{s}$, and the effective, post detection, noise variance vector $\widehat{\mathbf{n}}$. $\widehat{\mathbf{y}}$ and $\widehat{\mathbf{n}}$ are given by (2) and (3) [4, 22]:

$$\widehat{\mathbf{y}} = \left(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I}\right)^{-1} \cdot \mathbf{H}^*\mathbf{y} = \mathbf{W}_{\text{MMSE}} \cdot \mathbf{y}, \tag{2}$$

$$\begin{aligned} \widehat{\mathbf{n}} &= \operatorname{diag}\left(E\left[(\widehat{\mathbf{y}} - \mathbf{s})(\widehat{\mathbf{y}} - \mathbf{s})^*\right]\right) \\ &= \operatorname{diag}\left(N_0 \cdot \left(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I}\right)^{-1}\right), \end{aligned} \tag{3}$$

where $(\cdot)^*$ is a conjugate-transpose operation and $\operatorname{diag}(\cdot)$ represents the mapping of the diagonal components of a matrix to a column vector.

It is worth noting that as part of an entire MIMO system, the MIMO detector output will feed into a soft decision FEC (forward error correction) decoder, which in turn needs to calculate the log likelihood ratios (LLRs). The LLR calculations [21, 23] need the $\widehat{\mathbf{n}}$ estimates which will be provided by the proposed MIMO detector block. Note that $\widehat{\mathbf{n}}$ output from the MIMO detector is required only when the soft decision metric is used for FEC decoding. In our system, the linear MMSE detector provides $\widehat{y}_k$ and $\widehat{n}_k$ to the kth soft bit decision computation block (see Figure 1), where $\widehat{\mathbf{y}} = \left[\widehat{y}_1 \ \cdots \ \widehat{y}_N\right]^T$ and $\widehat{\mathbf{n}} = \left[\widehat{n}_1 \ \cdots \ \widehat{n}_N\right]^T$.

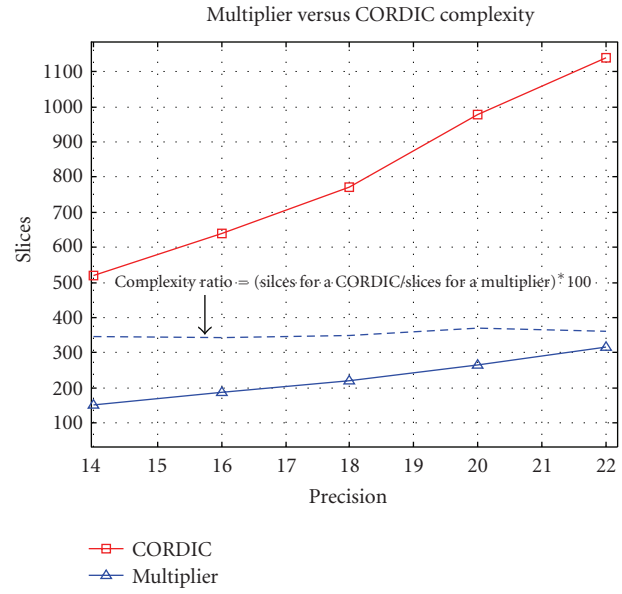


Figure 2: Hardware Complexity of a CORDIC or Multiplier.

3. A COMPREHENSIVE MEASURE OF ALGORITHMIC COMPLEXITY

Before we start to analyze the complexity of alternative MMSE MIMO detection algorithms, it is necessary to define a realistic and comprehensive measure of algorithm complexity. The traditional technique for estimating algorithm complexity is to simply count the number of operations. However, the operation count alone is not a sufficient measure to estimate realistic algorithm complexity especially when we consider fixed point precision issues. To illustrate this, consider Figure 2 which shows the FPGA slice count for a CORDIC operator and a lookup table-based multiplier synthesized on a Xilinx Virtex-2 FPGA. We have chosen these operators since all the candidate MMSE detection algorithms being considered here are either CORDIC or multiplier intensive.

Figure 2 clearly shows that the hardware complexity of a CORDIC operator or a multiplier is linearly proportional to its bit precision and the slice count ratio between two operators can be approximated as a constant within a wide precision range (14 ∼ 22 bits). We propose a more comprehensive metric for measuring the complexity of an algorithm. The new metric is defined in (4) and is termed the "*adjusted operation counts*". It takes into account the bit precision of the operator, the relative complexity of the operator, and naturally the number of operations. We will adopt this metric throughout the paper;

$$\text{Adjusted Operation Counts} = \sum_{m=1}^{M} \alpha_m \left(\frac{b_m^1 + b_m^2}{b_0} \right), \quad (4)$$

where M is the total number of operations, b_0 is a normalization factor, b_m^i is the bit precision of the ith input operand of the mth operation and α_m is the relative complexity coefficient of the mth operation.

Normalizing our operations to 16-bit precision multiplications, we let $b_0 = 32$ (corresponding to two 16-bit precision operands) and set $\alpha_m = 1$ when the mth operation is a multiplication. When comparing multiplier and CORDIC operations in FPGA implementation, we will assume that for the same precision, the CORDIC operation has 3.5 times higher hardware complexity than the multiplication as Figure 2 indicates ($\alpha_m = 3.5$ for a CORDIC). In this vein, a 24-bit precision multiplication is regarded as 1.5 effective operations (1.5 in *adjusted operation counts*) while a 24-bit precision CORDIC operation is counted as 5.25 in *adjusted operation counts*.

4. LINEAR MMSE DETECTION ALGORITHM COMPARISON

All the MMSE detector implementations reported in the literature [17–20] use a Squared MMSE formulation of the MIMO detector problem with an explicit matrix inversion of $(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$. Of these, [17, 19, 20] use QR decomposition to solve the matrix inversion problem.

A QR decomposition-based Squared MMSE formulation is given in (5)–(8):

$$\mathbf{A}_S = \mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I}, \quad (5)$$

$$\text{QR decomposition: } \mathbf{A}_S = \mathbf{Q}_S\mathbf{R}_S \Longrightarrow \mathbf{A}_S^{-1} = \mathbf{R}_S^{-1}\mathbf{Q}_S^*, \quad (6)$$

$$\widehat{\mathbf{y}} = \mathbf{R}_S^{-1}\mathbf{Q}_S^*\mathbf{H}^*\mathbf{y} = \mathbf{W}_{\text{MMSE}} \cdot \mathbf{y}, \quad (7)$$

$$\widehat{\mathbf{n}} = \text{diag}\left(N_0 \cdot \mathbf{R}_S^{-1}\mathbf{Q}_S^*\right). \quad (8)$$

The Square-Root MMSE formulation (9)–(11) [14, 22] exploits the structure of the compound matrix $\begin{bmatrix} \mathbf{H} \\ \sqrt{N_0} \cdot \mathbf{I} \end{bmatrix}$ in order to eliminate the need for matrix inversion. It also significantly reduces the precision requirements of the system, as will be seen in later sections. The Square-Root MMSE formulation was first introduced in [14] where it was used in the implementation of V-BLAST-type detectors [16, 22]. Its application to the linear MMSE MIMO detector has hitherto been unexplored and is one of the contributions of the present work;

$$\mathbf{A}_{SQ}^{1/2} = \begin{bmatrix} \mathbf{H} \\ \sqrt{N_0} \cdot \mathbf{I} \end{bmatrix} = \mathbf{Q}_{SQ}\mathbf{R}_{SQ} = \begin{bmatrix} \mathbf{Q_1} \\ \mathbf{Q_2} \end{bmatrix} \mathbf{R}_{SQ}, \quad (9)$$

$$\sqrt{N_0} \cdot \mathbf{I} = \mathbf{Q}_2\mathbf{R}_{SQ}, \qquad \mathbf{R}_{SQ}^{-1} = \frac{\mathbf{Q}_2}{\sqrt{N_0}}, \quad (10)$$

$$\widehat{\mathbf{y}} = \frac{\mathbf{Q}_2}{\sqrt{N_0}}\mathbf{Q}_1^*\mathbf{y} = \mathbf{W}_{\text{MMSE}} \cdot \mathbf{y},$$

$$\widehat{\mathbf{n}} = \text{diag}\left(\mathbf{Q}_2 \cdot \mathbf{Q}_2^*\right). \quad (11)$$

One interesting fact about the Square-Root formulation is that both $\mathbf{R}_{SQ}$ and $\mathbf{Q}_2$ are upper triangular matrices. In later section, we will exploit this property to help us reduce the number of hardware multipliers.

In order to come up with the best implementation, we carried out a side by side comparison of four alternative linear MMSE detection algorithms and chose the one with the lowest *adjusted operations count* metric. These four alternatives are

(1) Squared MMSE formulation with QR decomposition using Givens rotations,

(2) Squared MMSE with QR decomposition using modified Gram-Schmidt orthogonalization,

(3) Square-Root MMSE with QR decomposition using Givens rotations,

(4) Square-Root MMSE with QR decomposition using modified Gram-Schmidt orthogonalization.

A Givens rotation [24] can be efficiently implemented in hardware by using a CORDIC operator. Meanwhile, the Gram-Schmidt orthogonalization approach for QR decomposition was motivated by the presence of dedicated multipliers on the target FPGA, which could provide for a better balance in the utilization of the part. An overview of these techniques can be found in [17, 24].

4.1. Numerical stability analysis and algorithm complexity assessment

The *adjusted operation count* metric requires the minimum acceptable signal precision for each of the four alternatives listed above. In order to achieve this, fixed point simulations were performed operating over the IEEE 802.11n channel model D [25]. Our simulation setup includes a complete IEEE 802.11n reference system including all the transmitter and receiver elements shown in Figure 1. The simulation parameters such as the number of subcarriers, OFDM symbol duration, guard interval, and position of pilot subcarriers and the others are determined according to the IEEE 802.11n draft standard [5]. The 802.11n convolutional encoder along with soft-decision input Viterbi decoder was applied to the simulation. Packet size was set to 1000 bytes. The number of antennas at the transmitter and receiver was set to 4 and 64QAM constellation with FEC coding rate of 2/3 was used. This configuration corresponds to

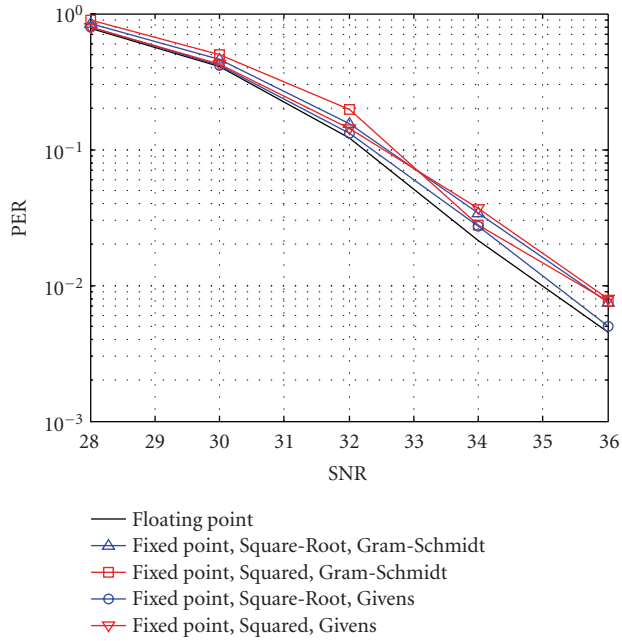


FIGURE 3: PER Performance of fixed point algorithms, 4×4 64 QAM.

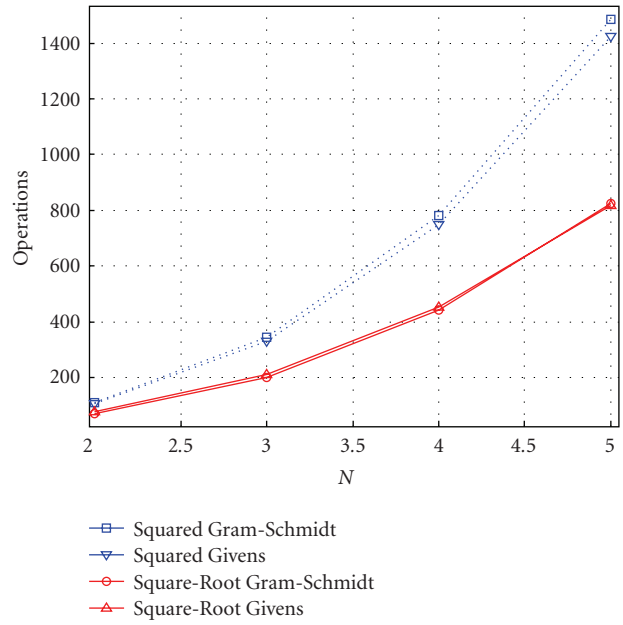


FIGURE 4: Operation count for each algorithm.

MCS (Modulation and Coding Scheme) 29 in the 802.11n specification. This particular configuration requires higher SNR than most other modulation and coding schemes in the 802.11n standard and as such will be more sensitive to the quantization noise and stability issues that plague finite precision systems.

The required bit precisions for the four alternative detector algorithms are presented in Table 1. The bit precisions were determined through a Monte-Carlo-based study that plotted the packet error rate (PER) for the end to end system with the aim of finding the required signal precision that resulted in a *precision loss* of less than 0.5 dB. We define *precision loss* as the difference in SNR required to achieve 1% PER when using floating point precision and when using fixed point precision. The required bit precision for each intermediate matrix (e.g., $\mathbf{R}_S^{-1}$ in Table 1) was obtained in isolation assuming that all other matrices were represented with floating point. After we obtained the required bit precisions for all intermediate matrices, they were combined and fine-tuned together via incremental precision modifications until we achieve the target *precision loss*. The PER curves in Figure 3 show the fixed point design performance of our system where all matrices and corresponding arithmetic operations are represented with finite bit precisions specified in Table 1. In general, more bit precisions are required to operate at higher SNRs where numerical stability issues become critical due to the higher condition number of the matrix $\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I}$.

It is worth noting that the required bit precisions for modified Gram-Schmidt QR decomposition are higher than those for the Givens rotation-based QR in both the Squared and the Square-Root MMSE detection cases. The better numerical stability of Givens rotation comes from its unitary transformation property which preserves the magnitude. However, in the Square-Root MMSE formulation, the difference of the required bit precision between Gram-Schmidt and Givens rotation-based methods becomes smaller. This is because of the structure of the Square-Root algorithm. That is, the lower half rows of the matrix $\mathbf{A}_{SQ}^{1/2}$ in the Square-Root algorithm have very small values at high SNR and become the main impediments for the Givens rotation method in computing accurate rotation angles. On the contrary, the same problem is not critical in the Gram-Schmidt method since its $\mathbf{Q}_{SQ}$ computation is based on an entire column vector rather than only two components of the column vector. Moreover, we observe that the bit precision requirement of the modified Gram-Schmidt QR decomposition method is significantly relaxed in Square-Root detection since neither $\mathbf{R}_{SQ}$ nor $\mathbf{R}_{SQ}^{-1}$ is involved in the detection process. In the following section, we will introduce a dynamic scaling technique which will further improve the numerical stability of modified Gram-Schmidt QR decomposition in Square-Root MMSE detections.

In order to use the *adjusted operation counts* as a realistic measure of algorithm complexity, the number of operations for each algorithm needs to be specified. This is shown in Table 2. Most of the arithmetic operations involved in MMSE algorithms take complex numbers as input. When counting the number of operations in Table 2, we equate a complex multiplication to 3 real multiplications [26] while vectoring and rotating CORDIC operations on complex numbers are counted as 2 and 3 real CORDIC operations, respectively [16, 17]. Figure 4 shows the operation counts (sum of the number of multiplication, division, square-root and CORDIC operations) for computing $\mathbf{W}_{\text{MMSE}}$ and $\hat{\mathbf{n}}$ as a function of the number of antennas N.

Table 1: Required bit precisions.

Fixed point computation	Squared MMSE algorithm		Square-Root MMSE algorithm	
	Gram-Schmidt QR-based	Givens QR-based	Gram-Schmidt QR-based	Givens QR- based
$\mathbf{H}$, N_0 $\mathbf{I}$ **(input)**	14		14	
$\mathbf{A}_s$ or $\mathbf{A}_{SQ}^{1/2}$	14		19	16
$\mathbf{Q}_S$ or $\mathbf{Q}_{SQ}$	25	15	19	16
$\mathbf{R}_S$ or $\mathbf{R}_{SQ}$	27	17	21	18
$\mathbf{R}_S^{-1}$	20		Not required	
$\mathbf{A}_S^{-1} = \mathbf{R}_S^{-1}\mathbf{Q}_S^*$	20		Not required	
$\frac{1}{\sqrt{N_0}}\mathbf{Q}_2$	Not required		14	
$\mathbf{W}_{\text{MMSE}}$	16		14	
$\hat{\mathbf{n}} = \text{diag}\,(N_0 \cdot \mathbf{A}_S^{-1}) = \text{diag}\,(\mathbf{Q}_2\mathbf{Q}_2^*)$	16		14	

Table 2: Operation counts for computing $\mathbf{W}_{\text{MMSE}}$ and $\hat{\mathbf{n}}$.

Real value input operations	Squared MMSE		Square-Root MMSE	
	Gram-Schmidt QR	Givens rotation QR	Gram-Schmidt QR	Givens rotation QR
Multiplications	$\frac{21}{2}N^3 + 7N^2 - \frac{1}{2}N$	$\frac{15}{2}N^3 + 6N^2 - \frac{1}{2}N$	$\frac{11}{2}N^3 + 5N^2 + \frac{5}{2}N$	$\frac{3}{2}N^3 + \frac{7}{2}N^2$
Divisions	N	N	$N+1$	1
Square roots	N	0	$N+1$	1
CORDIC operations	0	$\frac{5}{2}N^3 + \frac{3}{2}N^2 - 2N$	0	$3N^3 + 7N^2 - N - 1$

Combining the bit precisions in Table 1 and the operation counts for each algorithm in Table 2, we can compute the *adjusted operation counts* for all algorithms. In *adjusted operation counts* computation (4), we set $\alpha_m = 0$ for additions (the relative hardware complexity of an addition is 0) because they take much lower resources in FPGAs than multiplications or CORDIC operations. Furthermore, since all algorithms require a small number (less than $N + 2$) of divisions and square-root operations, a single time-shared divider and square-root operator will be sufficient for each algorithm when N is reasonably small (i.e., less than 6). Consequently, the hardware complexity involved in division and square-root operations will be assumed to be the same for all MMSE detection algorithms. Therefore, in this work, we compute *adjusted operation counts* by only considering multiplications ($\alpha_m = 1$) and CORDIC operations ($\alpha_m = 3.5$). The *adjusted operation counts* for a 4×4 MIMO detector are shown in Table 3. It is seen that the Square-Root MMSE detection algorithms require significantly less hardware resources than their Squared MMSE counterparts.

4.2. Algorithm enhancement: dynamic scaling technique

It is well known that the modified Gram-Schmidt QR decomposition has an advantage in numerical stability when compared to the original Gram-Schmidt algorithm [24]. In addition, we have found that this algorithm when used in a Square-Root MMSE detector can be made even more efficient by exploiting the fact that the $\mathbf{R}_{SQ}$ matrix, which results from the QR decomposition, does not contribute to the MMSE solution. Essentially, we can apply any processing to $\mathbf{A}_{SQ}^{1/2}$ as long as $\mathbf{Q}_{SQ}$ remains unchanged, even if it does not preserve $\mathbf{R}_{SQ}$. As we can see from (12), dynamic scaling of the ith column, $\mathbf{v}_i$, with an arbitrary constant c_i has the property of preserving the $\mathbf{Q}_{SQ}$ matrix but not necessarily $\mathbf{R}_{SQ}$:

$$\begin{aligned}\mathbf{A}_{SQ}^{1/2} &= [\mathbf{v}_1 \;\; \mathbf{v}_2 \;\; \cdots \;\; \mathbf{v}_N] = \mathbf{Q}_{SQ} \cdot \mathbf{R}_{SQ} = [\mathbf{u}_1 \;\; \mathbf{u}_2 \;\; \cdots \;\; \mathbf{u}_N] \cdot \mathbf{R}_{SQ} \\ &\Longrightarrow \tilde{\mathbf{A}}_{SQ}^{1/2} = [c_1\mathbf{v}_1 \;\; c_2\mathbf{v}_2 \;\; \cdots \;\; c_N\mathbf{v}_N] = \mathbf{Q}_{SQ} \cdot \tilde{\mathbf{R}}_{SQ} \\ &= [\mathbf{u}_1 \;\; \mathbf{u}_2 \;\; \cdots \;\; \mathbf{u}_N] \cdot \tilde{\mathbf{R}}_{SQ}.\end{aligned} \tag{12}$$

The modified and scaled Gram-Schmidt QR decomposition for the Square-Root MMSE solution with the recursive dynamic scaling step is shown in Algorithm 1. The dynamic scaling steps correspond to steps (d)–(g) in Algorithm 1. By exploiting this recursive scaling, we can guarantee that the maximum absolute value of the real or imaginary components of the vector $\mathbf{v}_j$ is always within a predefined range. The significance of the dynamic scaling technique on Gram-Schmidt QR decomposition comes from the fact that each column orthogonalization makes the magnitude of the projection columns ($\mathbf{v}_j := \mathbf{v}_j - r_{ij} \cdot \mathbf{u}_i$) become smaller as the recursive orthogonalization step continues. In order to resolve this problem, we introduced steps (d)–(e) in Algorithm 1. It makes the magnitude of the projection vector $\mathbf{v}_j$ always greater than a certain threshold so that we

Table 3: Adjusted operation counts for computing $\mathbf{W}_{\text{MMSE}}$ and $\widehat{\mathbf{n}}$ (4×4 detector).

	Squared MMSE		Square-Root MMSE	
	Givens rotation QR-based	Gram-Schmidt QR-based	Givens rotation QR-based	Gram-Schmidt QR-based
Adjusted operation counts	1150.50	872.50	780.84	488.44

(a) $\mathbf{A}_{SQ}^{1/2} = \begin{bmatrix} \mathbf{H} \\ \sqrt{N_0} \cdot \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \end{bmatrix}$
(b) for $i = 1$ to N
(c) for $j = i$ to N
(d) while $(\max\{|\Re(\nu_{1,j})|, |\Im(\nu_{1,j})|, \ldots, |\Im(\nu_{2N,j})|\} < 2^L)$
(e) $\mathbf{v}_j := 2\mathbf{v}_j$
(f) while $(\max\{|\Re(\nu_{1,j})|, |\Im(\nu_{1,j})|, \ldots, |\Im(\nu_{2N,j})|\} > 2^U)$
(g) $\mathbf{v}_j := \mathbf{v}_j/2$
(h) end
(i) $r_{ii} := \|\mathbf{v}_i\|$, $\mathbf{u}_i := \dfrac{\mathbf{v}_i}{\|\mathbf{v}_i\|}$
(j) for $j = i + 1$ to N
(k) $r_{ij} := \mathbf{u}_i^* \cdot \mathbf{v}_j$
(l) $\mathbf{v}_j := \mathbf{v}_j - r_{ij} \cdot \mathbf{u}_i$
(m) end
(n) end

Algorithm 1: The modified and scaled Gram-Schmidt QR decomposition ($\mathbf{v}_i = [\nu_{1,i} \cdots \nu_{2N,i}]^T$, $\Re(\cdot)$ and $\Im(\cdot)$ stand for real and imaginary parts of a complex number, respectively, $\mathbf{Q}_{SQ} = [\mathbf{u}_1 \cdots \mathbf{u}_N]$, L and U are predefined lower and upper bounds).

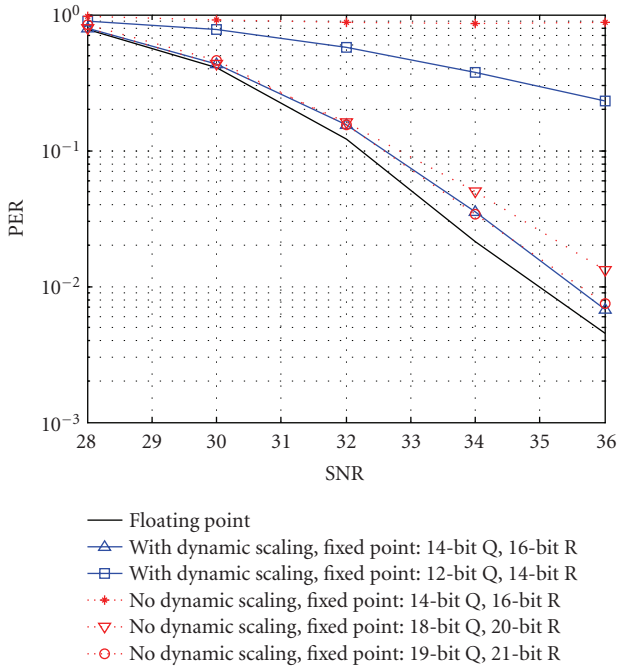


Figure 5: Impact of dynamic scaling on Gram-Schmidt QR.

can activate the full dynamic range all the time. Dynamic scaling also prevents r_{ii} (namely, $\|\mathbf{v}_j\|$) from becoming a very large number and consequently the dynamic range of $1/\|\mathbf{v}_j\|$ can be controlled such that it does not exceed the desired precision. This improves the numerical stability of $\mathbf{v}_i/\|\mathbf{v}_i\|$, while maintaining low bit precision.

The dynamic scaling technique is unique to Square-Root MMSE detection. The preprocessing such as (12) cannot be applied to Squared MMSE detection since its solution depends on both $\mathbf{R}_S$ and $\mathbf{Q}_S$ of the QR decomposition process. Hence, this type of dynamic scaling technique had not been exploited in previous works [17–20], which were based on the Squared MMSE formulation.

The impact of recursive dynamic scaling on a modified Gram-Schmidt QR-based Square-Root MIMO detection algorithm is shown in Figure 5 and Table 4. On average, 5 bits of precision is saved in the fixed point QR decomposition which makes the modified Gram-Schmidt QR-based Square-Root algorithm more hardware friendly. Note that a similar technique can be applied to Givens rotation-based QR decomposition in Square-Root MMSE detection. However, the impact of the dynamic scaling on a Givens rotation QR-based Square-Root algorithm is not as significant (see Table 4) due to the already well-behaved numerical properties of that algorithm. As Table 4 shows, the proposed dynamic scaling technique enhances the numerical stability of the modified Gram-Schmidt QR decomposition to a level that is comparable to the unitary transform-based QR.

The *adjusted operation counts* in Table 5 show the algorithm complexity both with and without dynamic scaling. As shown there, Square-Root MMSE detections (even without dynamic scaling) are approximately 40% less complex

Table 4: Required bit precisions of QR decomposition for Square-Root detection.

	Givens rotation-based QR		Gram-Schmidt-based QR	
	Without dynamic scaling	With dynamic scaling	Without dynamic scaling	With dynamic scaling
$\mathbf{A}_{SQ}^{1/2}$, $\mathbf{Q}_{SQ}$	16	14	19	14
$\mathbf{R}_{SQ}$	18	16	21	16
Precisions for all other matrices	Same as Table 1			

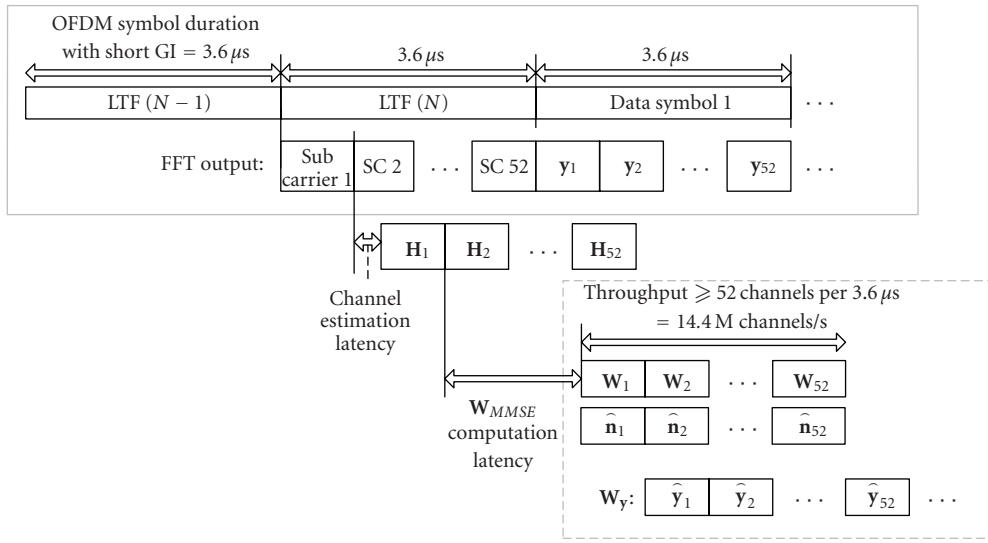


Figure 6: MIMO detection interface timing.

compared to Squared MMSE detection. In addition, the proposed dynamic scaling technique provides nearly 20% additional saving in hardware complexity for the Gram-Schmidt QR-based Square-Root MIMO detector.

Remark that when one considers hardware implementation on an FPGA, multiplication-intensive methods such as the modified Gram-Schmidt QR decomposition are usually more desirable than a CORDIC-intensive Givens rotation QR algorithm because (a) dedicated multipliers are available in FPGAs without extra cost, whereas CORDIC operators would consume significant number of FPGA slices (see Figure 2); (b) the latency of a pipelined CORDIC operator is linearly proportional to its bit precision while a dedicated multiplier on an FPGA has a single-clock latency. Based on the complexity assessment in this section and the fact that we target an FPGA implementation where a number of dedicated multipliers are available, we select the modified Gram-Schmidt QR decomposition combined with Square-Root MMSE MIMO detection as the algorithm for our hardware implementation. Dynamic scaling technique is also applied to the hardware design in order to reduce its bit precision requirement of the design.

5. HARDWARE IMPLEMENTATION ON FPGAs

The exploration of the algorithmic space in the prior sections led us to an algorithmic solution with the smallest *adjusted operation count* for a given performance. In this section, we continue to optimize the design by making tradeoffs and enhancements at the hardware architecture level. The primary tradeoffs made at this level are (i) time-sharing of multiplier resources; (ii) maximizing hardware utilization by exploiting the sparsity of some matrices; and (iii) an efficient implementation of the dynamic scaling procedure. These three techniques are elaborated in this section where the performance gain for each is clearly discussed. The result is an FPGA-based implementation that not only meets the requirements for this work, but is also quite superior to other detectors appearing in the recent literature.

5.1. MIMO detector overview and interface

For each subcarrier, the inputs to the MIMO detector are the $N \times N$ channel matrix $\mathbf{H}$ and the $N \times 1$ receive vector $\mathbf{y}$. Figure 6 shows the interface timing diagram of the MIMO detector for our IEEE 802.11n test case. This test case corresponds to a 4×4 MIMO OFDM system with 52 data subcarriers and an OFDM symbol duration of 3.6 μs.

The 802.11n packet structure includes several training symbols referred to as LTFs (long training fields) for the purpose of estimating the channel matrices for each of the 52 data subcarriers. After the last LTF symbol is processed, channel estimate matrices are fed into the MIMO detector. Upon the delivery of the first channel estimation matrix (corresponding to the first subcarrier) to the MIMO decoder, the decoder must produce the MMSE weight matrix $\mathbf{W}_{\text{MMSE}}$

Table 5: Adjusted operation counts for computing $\mathbf{W}_{\text{MMSE}}$ and $\widehat{\mathbf{n}}$ (4×4 MMSE detection).

	Squared MMSE		Square-Root MMSE			
	Givens rotation QR-based	Gram-Schmidt QR-based	Givens rotation QR-based		Gram-Schmidt QR-based	
	Dynamic scaling not available		Without dynamic scaling	With dynamic scaling	Without dynamic scaling	With dynamic scaling
Adjusted operation counts	1150.50	872.50	780.84	743.46	488.44	397.81

Table 6: Place and route report.

Target FPGA	Slices	Number of real multipliers	BRAMs
xc2v6000 (speed grade-6)	9,003 out of 33792	66	24
xc4vlx160 (speed grade-12)	7,932 out of 67854		
Target FPGA	Supportable operating clock frequency (f_{clk})	Latency (clocks)	Data throughput $\mathbf{W}_{\text{MMSE}}$, $\widehat{\mathbf{n}}$ and $\widehat{\mathbf{y}}$,
xc2v6000 (speed grade-6)	140 MHz	388	$f_{clk}/8$ (instances per second)
xc4vlx160 (speed grade-12)	160 MHz		

and the effective noise power vector $\widehat{\mathbf{n}}$ within $4\,\mu$s. This is per our design requirements. Figure 6 shows the timing diagram for this sequence of events. As soon as the $\mathbf{W}_{\text{MMSE}}$ matrix and the received vector $\mathbf{y}$ for the first subcarrier become available, the MMSE detection output vector $\widehat{\mathbf{y}}$ will be generated by the MIMO detector. The $\mathbf{W}_{\text{MMSE}}$, $\widehat{\mathbf{n}}$, and $\widehat{\mathbf{y}}$ computation throughput of the detector must be greater than or equal to 14.4 M instances per second which is the rate at which the $\mathbf{y}$ vectors and the channel estimates are presented to the MIMO detector. Otherwise, the detector will incur additional latency.

5.2. Multiplier sharing architecture

In our test case, a new 4×4 channel estimation matrix $\mathbf{H}$ is presented to the detector at the maximum rate of $\phi = 14.4$ M instances per second. A fully pipelined detector must provide $\mathbf{W}_{\text{MMSE}}$, $\widehat{\mathbf{n}}$, and $\widehat{\mathbf{y}}$ every 69.4 ns ($= 1/\phi$) without the need for a FIFO and unnecessary latency. Generally, the input/output rate ϕ is much lower than the maximum operating clock frequency of the hardware. We define the *multiplier time sharing order* (γ) in (13):

$$\text{multiplier time sharing order}\,(\gamma) = \frac{\text{Operating Clock Freq. in MHz}}{\phi}. \tag{13}$$

For our FGPA implementation, the multiplier time sharing order is 8 implying that the minimum operating clock frequency is 115.2 ($= 8\times\phi$) MHz and a single multiplier processes 8 sets of inputs within a 14.4 MHz cycle. This specific multiplier time sharing order is naturally coupled with the size of the compound matrix $\mathbf{A}_{SQ}^{1/2}$ for the Square-Root MIMO detector.

For the 4×4 Square-Root MIMO detection, $\mathbf{A}_{SQ}^{1/2}$ is an 8×4 matrix and each step in the modified Gram-Schmidt QR decomposition takes an 8×1 column vector of $\mathbf{A}_{SQ}^{1/2}$ as the input. Assuming that the matrix $\mathbf{A}_{SQ}^{1/2}$ is dense, the norm square computation steps ($\|\mathbf{v}\|^2$) and projection vector computation steps ($\mathbf{u}^* \cdot \mathbf{v}$ or $r_{ii} \cdot \mathbf{u}$) each require 8 complex multiplications. As a result, if the multiplier can run at a clock frequency of 115.2 MHz, the same multiplier can be shared within a single operation step ($\|\mathbf{v}\|^2$, $\mathbf{u}^* \cdot \mathbf{v}$, or $r_{ii} \cdot \mathbf{u}$) producing the output at the rate of 14.4 M instances per second. With this multiplier sharing architecture, the squared Euclidean norm ($\|\mathbf{v}\|^2$) and the $\mathbf{v}$ vector update ($\mathbf{v} := \mathbf{v} - (\mathbf{u}^*\mathbf{v}) \cdot \mathbf{u}$) operations require only 2 and 6 real multipliers, respectively.

Figures 7 and 8 show the overall architecture of the fully pipelined 4×4 MIMO detector with the proposed multiplier sharing architecture. The square-root and division operator in Figure 7 are instantiated by using Xilinx Coregen blocks [26].

5.3. Multiplier saving techniques

The modified and scaled Gram-Schmidt QR decomposition circuit in Figure 7 does not make use of the sparsity of the $\mathbf{A}_{SQ}^{1/2}$ matrix. Since the lower half of $\mathbf{A}_{SQ}^{1/2}$ is sparse ($\mathbf{Q}_2$ is upper triangular), the multipliers in the $\|\mathbf{v}\|^2$ and the $\mathbf{v} - (\mathbf{u}^*\mathbf{v}) \cdot \mathbf{u}$ computation are not active all the time. This can be exploited to save multipliers when the orthogonalization is performed on the columns of $\mathbf{A}_{SQ}^{1/2}$. In Figure 9, real multipliers in the $\|\mathbf{v}_1\|^2$ computation are *active* (shaded rectangles) during only 5 out of 8 clock cycles, and complex multipliers in the $\mathbf{u}_1^*\mathbf{v}_j$ computation have 4 *inactive* slots (unshaded rectangles) out of 8. Meanwhile, only 5 complex multiplications are required to compute $(\mathbf{u}_1^*\mathbf{v}_j) \cdot \mathbf{u}_1$ and the 5th component of $\mathbf{u}_1$ is a real number. Therefore, $(\mathbf{u}_1^*\mathbf{v}_j) \cdot u_{1,1} \sim (\mathbf{u}_1^*\mathbf{v}_j) \cdot u_{1,4}$ can be computed using the *inactive* cycles of the multipliers for $\mathbf{u}_1^*\mathbf{v}_j$ computation, while $(\mathbf{u}_1^*\mathbf{v}_j) \cdot u_{1,5}$ can use the *inactive* slots in the $\|\mathbf{v}_1\|^2$ computation. This technique provides a saving of 17% in the required multiplier resources for the QR decomposition circuit.

We can save additional multiplier resources in the scalar-matrix or matrix-matrix multiplication by exploiting the fact that some elements of the upper triangular matrix $\mathbf{Q}_2$ are real numbers. Among the 10 nonzero components in

Table 7: Resource usage comparison.

	Computation	Slices	Multiplier	BRAM	Note
This work	$\frac{\mathbf{Q}_2}{\sqrt{N_0}}$	6487 (Virtex2)	45	15	Corresponds to complexity of $(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$
	$\mathbf{W}_{\text{MMSE}}$	7679(Virtex2)	58	19	Includes $1/\sqrt{N_0} \cdot \mathbf{Q}_2$ computation
[17]	$\mathbf{W}_{\text{MMSE}}$	16865(Virtex2)	44	101	
[18]	$\alpha \cdot (\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$	4446(Virtex2)	101	N/A	α scaling will require additional multipliers and dividers.
[19]	$(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$	86% of Virtex2[(1)]	N/A	N/A	
[20]	$(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$	9117 (Virtex4)	22	9	

[(1)] The exact slice count is not available since its FPGA part name is not given.

Table 8: Throughput and latency comparison.

	Output	Max. throughput (million instances per second)	f_{clk}	Latency
This work	$\mathbf{W}_{\text{MMSE}}, \hat{\mathbf{y}}, \hat{\mathbf{n}}$	17.50	140 MHz	388 clks
		20.00	160 MHz	
[17]	$\mathbf{W}_{\text{MMSE}}$	N/A	N/A	3000 clks
[18]	$\alpha \cdot (\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$	6.75[(2)]	108 MHz	64 clks
[19]	$(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$	6.25[(2)]	100 MHz	350 clks
[20]	$(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$	0.13[(3)]	115 MHz	933 clks

[(2)] The throughput is not specified in the reference. However, it can be computed from its architecture.

[(3)] This is a floating point design.

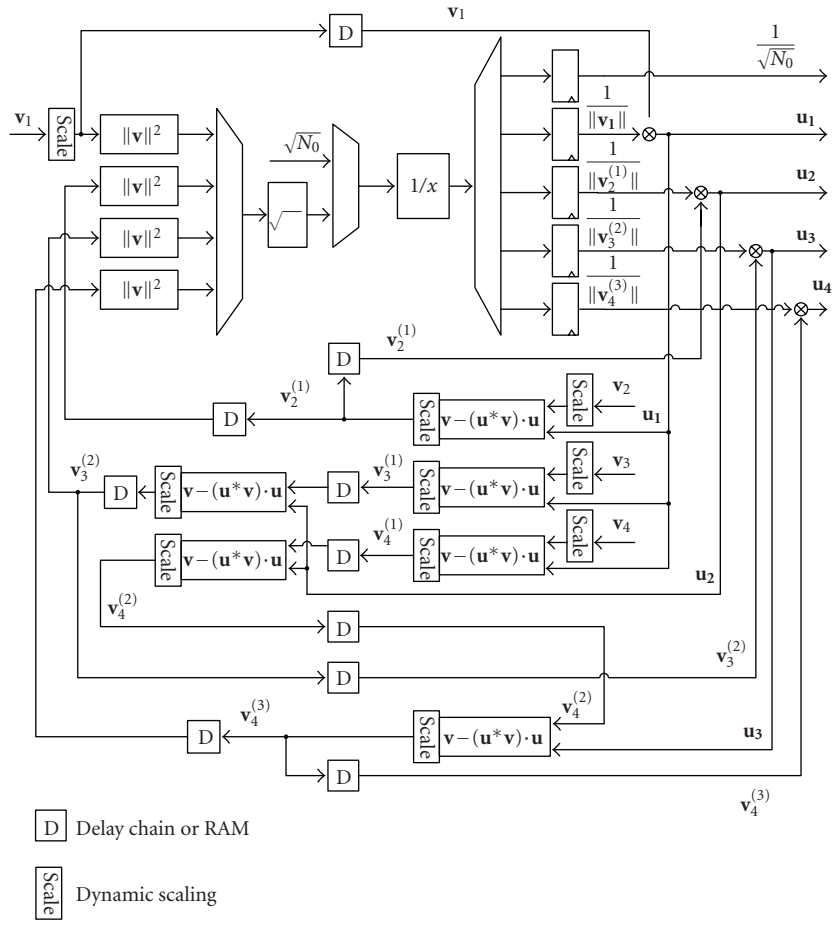


Figure 7: QR decomposition circuit.

Table 9: QR decomposition engine comparison.

	QR engine output	Throughput (million inst/s)	Gate count	Gate count per throughput	Note
This work	$\mathbf{Q}_1, \mathbf{Q}_2, \widetilde{\mathbf{R}}_{SQ}$	17.50 (Virtex2 FPGA, 0.120 μm)	157k[(4)]	8.97 k per M inst/s	$\mathbf{R}_{SQ}$ can be obtained by scaling each row of $\widetilde{\mathbf{R}}_{SQ}$.
[16]	$\mathbf{Q}_1, \mathbf{R}_{SQ}$	1.56 (0.25 μm ASIC)	54k	34.62 k per M inst/s	$\mathbf{R}_{SQ}$ is sorted based on column norms

[(4)] Estimated gate count from Xilinx ISE9.1 report. The QR engine was mapped, placed, and routed in isolation.

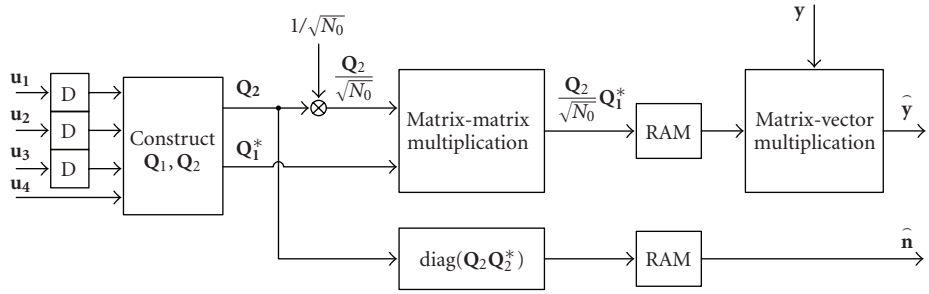


Figure 8: MMSE solution computation.

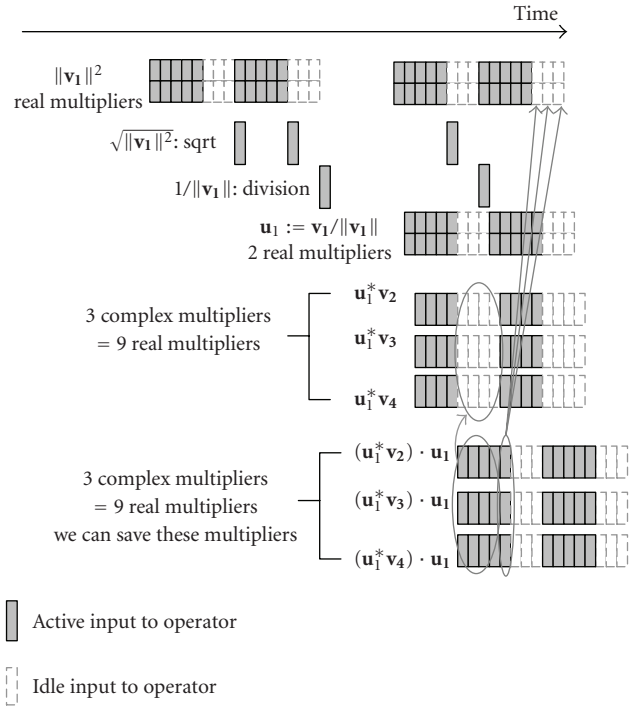


Figure 9: Vector orthogonalization schedule (ν_1).

$\mathbf{Q}_2$, 4 diagonal components are real and the remaining 6 are complex. Hence, when we compute diag ($\mathbf{Q}_2\mathbf{Q}_2^*$), 2 real multipliers are sufficient with $\gamma = 8$ time sharing. Similarly, $1/\sqrt{N_0}\cdot\mathbf{Q}_2$ (scalar-matrix) multiplication and $1/\sqrt{N_0}\mathbf{Q}_2\cdot\mathbf{Q}_1^*$ (matrix-matrix) multiplication can be implemented with 2 and 13 real multipliers, respectively. It is worth noting that on average, our design with multiplier sharing/saving achieves 93% multiplier utilization (active time over total time), which is 30% higher than the hardware utilization reported in [19].

5.4. Implementation of dynamic scaling

As shown in Figure 7, the QR decomposition procedure requires 10 identical dynamic scaling units. Thus, it is important to implement this function in a hardware optimized fashion. The input of the dynamic scaling circuit is an 8×1 vector $\mathbf{v}$ which corresponds to a column vector of the compound matrix $\mathbf{A}_{SQ}^{1/2}$. We implemented a pipelined dynamic scaling circuit where each component of the column vector $\mathbf{v}$ is fed sequentially as input. Figure 10 depicts the overall structure of the dynamic scaling circuit. The scaling is performed in two steps. First, absolute values of the real and imaginary parts of the input vector (all 8 elements of the vector) are *bitwise OR*ed and stored in a register called *accumulated_OR*. By looking at the most significant nonzero bit position of the *accumulated_OR* register, one can verify whether the largest absolute value of $\mathbf{v}$ is within the predefined range (14). Second, if the most significant nonzero bit position of the *accumulated_OR* register is out of bound, the input signals are shifted based on the position of the most significant nonzero bit of the *accumulated_OR register*. A brute-force approach for performing this most significant nonzero bit search and barrel shift is given in the pseudocode of Algorithm 2 where the bit precision for $\mathbf{v}$ is 14, the lower bound L is 11, and the upper bound is inactive (i.e., $U = 14$);

$$2^L \le \max\{|\text{real}(\nu_1)|, \ldots, |\text{real}(\nu_{2N})|, |\text{imag}(\nu_1)|, \ldots, |\text{imag}(\nu_{2N})|\} \le 2^U. \quad (14)$$

The deeply nested if-else statements shown in Algorithm 2 combined with the barrel shift operation are highly resource demanding when synthesized to an FPGA.

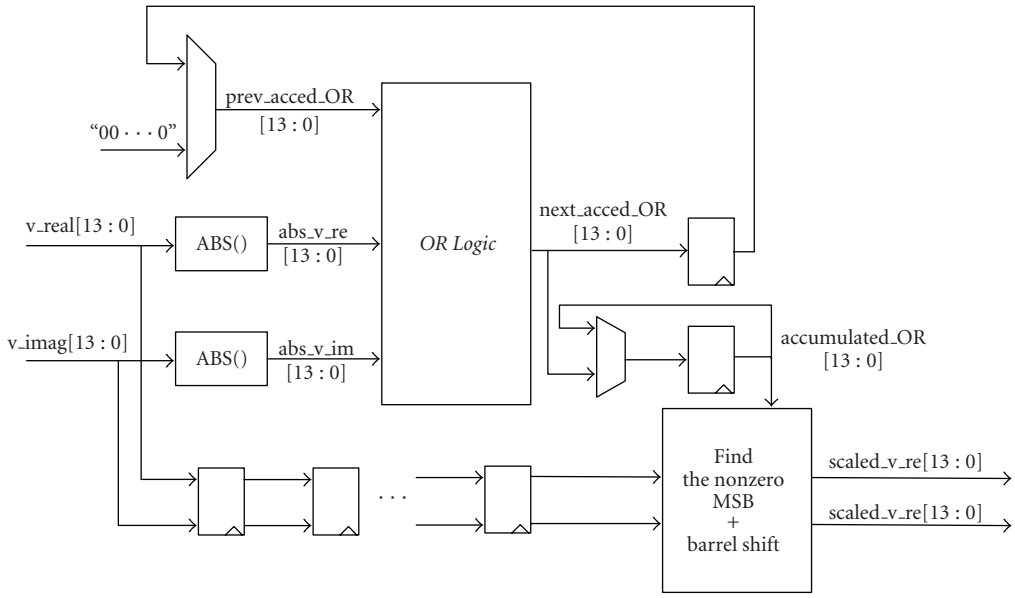
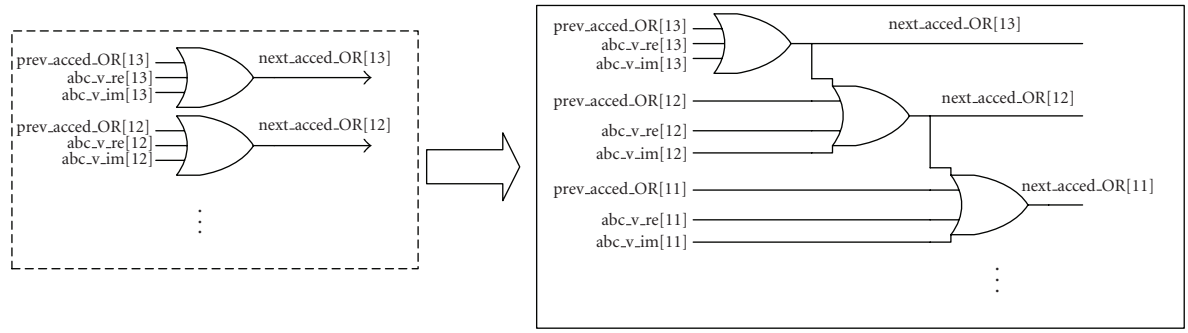


Figure 10: Dynamic scaling circuit.



(a) OR Logic for brute-force search

(b) OR Logic for binary search

Figure 11: OR Logic modification.

```
if (accumulated_OR (13 : 11) > 0) then
    0 bit shift;
elsif (accumulated_OR (10) = "1") then
    1 bit shift;
. . . . . .
. . . . . .
elsif (accumulated_OR(5) = "1") then
    6 bit shift;
else
    7 bit shift;
end if;
```

Algorithm 2: Brute-force search and barrel shift.

The deeply nested if-else statements can be avoided by using a binary search procedure. For the binary search, the *OR logic* in Figure 10 needs to be slightly modified from the simple bitwise OR shown in Figure 11(a) to the modified *OR logic* shown in Figure 11(b). The modified OR logic performs a *nonzero-bit propagating, bitwise OR* operation: once the first nonzero most significant bit is found, the remaining less significant bits from that position all become "1" regardless of other inputs. The modified OR logic allows us to use binary search, which is shown in the pseudocode of Algorithm 3.

The synthesis result for a binary search-based dynamic scaling circuit on a Virtex2 FPGA reveals that it consumes 145 slices, whereas the brute-force approach requires 241 slices. Hence, we achieve a saving of almost 1000 slices for the 10 dynamic scaling units needed throughout the entire QR decomposition process. It is worth noting that without dynamic scaling, each multiplier in the QR decomposition would have required larger bit precision which would have resulted in even more multiplier resources (refer to Table 1). Given that on an FPGA (such as the Xilinx Virtex-2 or Virtex-4 family) only 18×18 bit multipliers are available, each multiplication with more than 18-bit precision will require

```
if (accumulated_OR(8) = "1") then
    if (accumulated_OR(10) = "1") then
        if (accumulated_OR(11) = "1") then 0-bit shift;
        else 1-bit shift;
        end if;
    else
        if (accumulated_OR(9) = "1") then 2-bit shift;
        else 3-bit shift;
        end if;
    end if;
else
    if (accumulated_OR(6) = "1") then
        if (accumulated_OR(7) = "1") then 4-bit shift;
        else 5-bit shift;
        end if;
    else
        if (accumulated_OR(5) = "1") then 6-bit shift;
        else 7-bit shift;
        end if;
    end if;
end if;
```

Algorithm 3: Binary search and barrel shift.

2 dedicated multipliers. In other words, the number of hardware multipliers required for the Gram-Schmidt-based QR decomposition would have doubled if the proposed dynamic scaling technique was not applied. Hence, we can see that the dynamic scaling results in saving of 40 dedicated multipliers in the FPGA at the cost of 1450 additional logic slices.

6. FPGA IMPLEMENTATION RESULTS

The 4×4 MMSE MIMO detector design was successfully synthesized, placed, routed, and verified on both a Xilinx Virtex-2 and a Virtex-4 series part. These chips have a number of dedicated hardware multipliers and two-port block RAMs. Table 6 shows the implementation result of the IEEE 802.11n compatible MIMO detector including resource utilization from the place and route report.

The latency of the implemented detector is 2.77 μs when the operating clock frequency (f_{clk}) is 140 MHz, which is shorter than a single OFDM symbol in the IEEE 802.11n draft proposal [5]. The computation throughput for $\mathbf{W}_{\text{MMSE}}$, $\hat{\mathbf{n}}$, and $\hat{\mathbf{y}}$ is $f_{clk}/8$ instances per second, which implies that our target throughput of 14.4 M instances per second can be met by using any f_{clk} higher than 115.2 MHz. Assuming a 140 MHz clock, the implemented detector is able to compute 52 (the number of data subcarriers) $\mathbf{W}_{\text{MMSE}}$ matrices within 2.77 μs and provide a data throughput of 420 Mbps for a 64QAM 4×4 MIMO system.

Tables 7 and 8 show the resource usage, throughput, and latency results for this work and some prior work in the area. It is worth noting that none of [17–20] in Tables 7 and 8 is designed to produce a complete MMSE solution consisting of both $\hat{\mathbf{y}}$ and $\hat{\mathbf{n}}$ needed for soft-decision decoding. Additionally, [18–20] only compute $(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$ rather than $\mathbf{W}_{\text{MMSE}} = (\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1} \cdot \mathbf{H}^* = 1/\sqrt{N_0} \cdot \mathbf{Q}_2 \mathbf{Q}_1^*$. For comparison purposes, we specified the resource usage of our design for computing $1/\sqrt{N_0} \cdot \mathbf{Q}_2$, which corresponds to $(\mathbf{H}^*\mathbf{H} + N_0 \cdot \mathbf{I})^{-1}$ in Squared MMSE detection (compare (2) and (10)). Table 7 shows that our design utilizes 50% and 29% fewer slices than [17, 20], respectively, while the number of multipliers used in our design is only 45% of those in [18]. Meanwhile, the throughput of our design is at least 3 times higher than [18–20] as shown in Table 8. The throughput of [17] is not given but its latency is too high to meet the latency requirements of a commercial system such as 802.11n.

Finally, we compare our QR decomposition engine (on a Xilinx Virtex2) with an ASIC design in [16] which implements complex CORDIC-based QR decomposition on a 0.25 μm technology. The QR engine in [16] is designed for a Square-Root MMSE V-BLAST-type detector and is similar to the CORDIC-based QR decomposition for the Square-Root MMSE algorithm discussed in this work. However, it is worth noting that [16] does not compute $\mathbf{Q}_2$ while it computes $\mathbf{R}_{SQ}$ explicitly as only $\mathbf{R}_{SQ}$ is required in a successive canceling-based MIMO detector. Table 9 shows the throughput and (estimated) gate counts for QR decomposition engines in this work and the ASIC design reported in [16]. For comparison purposes, we assume that for a given design, an FPGA fabricated in 0.12 μm technology (Virtex2) provides comparable speed (throughput) as an ASIC implemented in 0.25 μm. By using *gate count per throughput* in Table 9 as the comparison metric, we observe that our design achieves significant gain over [16].

In summary, four main techniques which together form the major contribution of this work are responsible for this performance gain. They are (a) definition and adoption of a unified metric for simultaneous comparison of algorithmic complexity and numerical stability; (b) the combination of a modified Gram-Schmidt QR decomposition algorithm with Square-Root linear MMSE detection resulting in a matrix inversion free implementation; (c) a dynamic scaling algorithm that enhances numerical stability; and (d) an aggressive time-shared VLSI architecture. The above techniques are quite general and are readily applicable to any MMSE-based MIMO detector implementation.

7. CONCLUSION

In this paper, we studied hardware friendly algorithms that avoid matrix inversion for linear MMSE MIMO detection. We assessed algorithm complexity in terms of number of operations and bit precisions in fixed point designs, while considering FPGA implementation where a fixed number of dedicated hardware multipliers are available. We suggested a dynamic scaling technique for modified Gram-Schmidt QR decomposition that increases the numerical stability of the fixed point design. The resulting MIMO detector was successfully implemented and demonstrated on an FPGA. The designed 4×4 linear MMSE MIMO detector is capable of complying with the proposed IEEE 802.11n standard.

REFERENCES

[1] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–335, 1998.

[2] I. Telatar, "Capacity of multi-antenna Gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, 1999.

[3] A. J. Paulraj and C. B. Papadias, "Space-time processing for wireless communications," *IEEE Signal Processing Magazine*, vol. 14, no. 6, pp. 49–83, 1997.

[4] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bölcskei, "An overview of MIMO communications—a key to gigabit wireless," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–217, 2004.

[5] IEEE TGn Working Group, "Joint Proposal: High throughput extension to the 802.11 Standard," http://www.ieee802.org/11/.

[6] J. Chen, W. Zhu, B. Daneshrad, et al., "A real time 4×4 MIMO-OFDM SDR for wireless networking research," in *Proceedings of the 15th European Signal Processing Conference (EUSIPCO '07)*, Poznań, Poland, September 2007.

[7] C. M. Rader, "MUSE-a systolic array for adaptive nulling with 64 degrees of freedom, using Givens transformations and wafer scale integration," in *Proceedings of the International Conference on Application Specific Array Processors*, pp. 277–291, Berkeley, Calif, USA, August 1992.

[8] K. Raghunath and K. Parhi, "A 100 MHz pipelined RLS adaptive filter," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, vol. 5, pp. 3187–3190, Detroit, Mich, USA, May 1995.

[9] A. Burg, N. Felber, and W. Fichtner, "A 50 Mbps 4×4 maximum likelihood decoder for multiple-input multiple-output systems with QPSK modulation," in *Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS '03)*, vol. 1, pp. 332–335, Sharjah, UAE, December 2003.

[10] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1576, 2005.

[11] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proceedings of the 6th IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, pp. 65–68, Shanghai, China, May-June 2004.

[12] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, pp. 273–276, Phoenix, Ariz, USA, May 2002.

[13] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proceedings of the URSI International Symposium on Signals, Systems, and Electronics (ISSSE '98)*, pp. 295–300, Pisa, Italy, September-October 1998.

[14] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '00)*, vol. 2, pp. 737–740, Istanbul, Turkey, June 2000.

[15] Z. Guo and P. Nilsson, "A low-complexity VLSI architecture for square root MIMO detection," in *Proceedings of the IASTED International Conference on Circuits, Signals, and Systems (CSS '03)*, pp. 304–309, Cancun, Mexico, May 2003.

[16] P. Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner, "VLSI implementation of a high-speed iterative sorted MMSE QR decomposition," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '07)*, pp. 1421–1424, New Orleans, La, USA, May 2007.

[17] M. Myllylä, J.-M. Hintikka, J. R. Cavallaro, M. Juntti, M. Limingoja, and A. Byman, "Complexity analysis of MMSE detector architectures for MIMO OFDM systems," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 75–81, Pacific Grove, Calif, USA, October-November 2005.

[18] I. LaRoche and S. Roy, "An efficient regular matrix inversion circuit architecture for MIMO processing," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 4819–4822, Island of Kos, Greece, May 2006.

[19] F. Edman and V. Öwall, "A scalable pipelined complex valued matrix inversion architecture," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 5, pp. 4489–4492, Kobe, Japan, May 2005.

[20] M. Karkooti, J. R. Cavallaro, and C. Dick, "FPGA implementation of matrix inversion using QRD-RLS algorithm," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 1625–1629, Pacific Grove, Calif, USA, October-November 2005.

[21] I. B. Collings, M. R. G. Butler, and M. R. McKay, "Low complexity receiver design for MIMO bit-interleaved coded modulation," in *Proceedings of the 8th IEEE International Symposium on Spread Spectrum Techniques and Applications (ISSSTA '04)*, pp. 12–16, Sydney, Australia, August-September 2004.

[22] R. Böhnke, D. Wübben, V. Kühn, and K.-D. Kammeyer, "Reduced complexity MMSE detection for BLAST architectures," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 4, pp. 2258–2262, San Francisco, Calif, USA, December 2003.

[23] F. Tosato and P. Bisaglia, "Simplified soft-output demapper for binary interleaved COFDM with application to HIPER-LAN/2," in *Proceedings of IEEE International Conference on Communications (ICC '02)*, vol. 2, pp. 664–668, New York, NY, USA, April-May 2002.

[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.

[25] IEEE TGn Working Group, "TGn Channel Models," http://www.ieee802.org/11/.

[26] http://www.xilinx.com/ipcenter/index.htm.