

Fixed-Point Arithmetic for Implementing Massive MIMO Systems

Mi Tian, Mihai Sima, and Michael McGuire

Department of Electrical and Computer Engineering, University of Victoria

P.O. Box 1700 Stn CSC, Victoria, B.C. V8W 2Y2, Canada

tianmi66@uvic.ca, msima@ece.uvic.ca, mmcguire@uvic.ca

Abstract—Massive MIMO base stations are expensive to build due to the requirement for a large number of RF transceivers and high-resolution analog-to-digital converters. A way to reduce the implementation cost is to build the base stations with inexpensive hardware, resulting in the received signals to be coarsely quantized. First, the required signal power needed to achieve different receiver Bit-Error Rate (BER) levels is determined, as well as the extra signal power needed due to the quantization for given BER levels. To implement the data detection and decoding process in real time, fixed-point arithmetic with reduced precision is used. This article also reports the minimum wordlength needed to maintain the BER at acceptable levels. Specifically, the eigenvalue decomposition, which is the most computationally demanding portion of the receiver algorithm, can be calculated with wordlengths of 7 and 10 bits for eigenvectors and eigenvalues, respectively.

Index Terms—Massive MIMO, fixed-point arithmetic

I. INTRODUCTION

The number of Receive Antennas (Receivers), R , deployed in a Massive Multiple-Input Multiple-Output (MMIMO) base station is much larger than the number of Transmit Antennas (Transmitters), T , where only one transmit antenna per mobile user is considered in this article. Since large-scale computation is needed for data detection and decoding based on measurements from high-resolution Analog-to-Digital Converters (ADC) connected to highly-linear RF front-end amplifiers for each antenna, the cost of a MMIMO base station is very high. A way to reduce the cost of the implementation is to build the base stations with inexpensive low-end hardware [1], such as simple RF receivers feeding the received signals to coarse (1-, 2-, or 3-bit) ADCs [2], [3].

The conceptual equations of the detection process are given below, where $\mathbf{X}$ is the $T \times 1$ transmitted complex signal vector, $\mathbf{Y}$ is the $R \times 1$ received complex signal vector, $\mathbf{H}$ is the

$R \times T$ channel transfer matrix, where each entry is modeled as an independent identically distributed as complex circularly symmetric Gaussian random value, $\mathbf{N}$ is an $R \times 1$ complex noise vector, and $\mathbf{N}'$ is a $T \times 1$ complex noise vector.

$$\mathbf{Y} = \mathbf{H} \cdot \mathbf{X} + \mathbf{N} \Rightarrow \mathbf{H}^H \cdot \mathbf{Y} = \underbrace{(\mathbf{H}^H \cdot \mathbf{H})}_{\Theta} \cdot \mathbf{X} + \underbrace{\mathbf{H}^H \cdot \mathbf{N}}_{\mathbf{N}'} \quad (1)$$

The multiplication of both sides by $\mathbf{H}^H$ converts the $R \times T$ system to a $T \times T$ system, which is less expensive to solve. It is known that the Minimum Mean Square Error (MMSE) estimate of $\mathbf{X}$ from the reduced-order system has the same mean squared error as the solution of the original system [18]. The estimation of $\mathbf{X}$ through the solution of this linear system is the most computationally demanding portion of the receiver algorithm for reasonably high values of T . In this process, two major operations need to be performed: (i) calculation of the Hermitian matrix $\Theta = \mathbf{H}^H \cdot \mathbf{H}$, and (ii) EigenValue Decomposition (EVD) [4] of matrix $\Theta = \mathbf{Q} \cdot \Lambda \cdot \mathbf{Q}^H$ followed by solving the linear system by matrix multiplications.

The challenge is to perform these linear-algebra operations in real time on inexpensive hardware, such as reasonably sized Field-Programmable Gate Arrays (FPGA). To achieve such performance, only fixed-point arithmetic [5], [6] will be used. The channel matrix, $\mathbf{H}$, and the received signal vector, $\mathbf{Y}$, will be coarsely quantized with $B = 1$, $B = 2$, or $B = 3$ bits. This article expands our previous work [31] and reports the FPGA fixed-point arithmetic implementation details together with the minimum wordlength needed in the data detection and decoding process to maintain the BER at acceptable levels. Our contributions are as follows.

- 1) Strategy to efficiently load the estimated channel matrix, $\mathbf{H}$, into FPGA and efficiently calculate the matrix Θ .
- 2) FPGA implementation of the squared Euclidean norm and the square root operations with reduced precision used in converting Θ first into a Hessenberg form and then into a diagonal form.
- 3) Assessment of the minimum wordlength needed to complete the eigenvalue decomposition of matrix Θ through the Francis-Kublanovskaya algorithm.
- 4) Assessment of the BER and the extra signal power which is needed to compensate for the information loss due to quantization for different coarse quantization levels.

The article is organized as follows. Background information is presented in Section II. Coarse quantization in Massive

Manuscript received January 23, 2021. This work was supported in part by the University of Victoria, British Columbia, Canada, under a PhD fellowship, and a follow-up of the invited journal to the accepted and presented paper entitled "Massive MIMO in Fixed-Point Arithmetic" of the 23rd International Conference on Advanced Communication Technology (ICACT2021).

Mi Tian is a postdoctoral fellow in the Department of Electrical and Computer Engineering at University of Victoria, British Columbia, Canada (e-mail: tianmi66@uvic.ca).

Mihai Sima is an associate professor in the Department of Electrical and Computer Engineering at University of Victoria, British Columbia, Canada (corresponding author, phone: +1-250-721-8680, e-mail: msima@ece.uvic.ca).

Michael McGuire is an associate professor in the Department of Electrical and Computer Engineering at University of Victoria, British Columbia, Canada (e-mail: mmcguire@uvic.ca).

MIMO uplink communications is analyzed in Section III. The detection process implemented in fixed-point arithmetic is presented in Section IV. Section V concludes the article.

II. BACKGROUND

This section provides the fundamentals of MMIMO iterative channel estimation and simulation software for coarsely quantized MMIMO communications used in our previous work. It also outlines the basic FPGA architectural features.

A. Iterative Channel Estimation for MMIMO

In MMIMO a large number of sufficiently-spaced receive antennas results in a large probability of a high-capacity communications channel from each mobile user's transmitter to the base station [7]. This allows MMIMO systems to communicate over random multipath propagation channels with transmission power nearly as low as ideal Additive White Gaussian Noise (AWGN) channels. To take advantage of this situation, the receiver requires accurate estimates of radio Channel State Information (CSI), which in turn leads to the requirement for a large number of pilot symbols to be used when standard channel estimation techniques are employed [8].

Recently we have shown that the channel state information can be accurately estimated from coarsely quantized measurements using an iterative procedure, where data symbols detected and decoded in previous iterations are used as additional (virtual) pilot symbols [9]. For the first iteration, only the pilot symbols are used to estimate the channel, since no prior knowledge of the transmitted data is available. For the second and following iterations, channel estimation is enhanced by using the estimated data symbols from the previous iterations as virtual pilot symbols, to get a lower error estimate of the channel parameters. Our simulation results indicate that an accurate estimation is achieved in three iterations or less.

B. Simulation Software for Coarsely Quantized MMIMO

In our previous work [9] the linear system shown in Equation (1) was solved in 64-bit floating-point arithmetic [10] using the standard MATLAB[®] linear solution mechanism, where $\tilde{X} = H^+ \cdot Y$, where H^+ is the pseudo-inverse, is coded as $\tilde{X} = H \setminus Y$, as shown in Equations (2) and (3). While these standard solution computations give accurate results, direct implementation of these solutions for use in MMIMO receivers would require computational hardware with prohibitive cost in base stations. The backslash operator method was also applied for estimating the channel matrix, H .

$$Y = H \cdot X + N \Rightarrow \tilde{X} = H \setminus Y \quad (2)$$

$$H^H \cdot Y = \Theta \cdot X + N' \Rightarrow \tilde{X} = \Theta \setminus (H^H \cdot Y) \quad (3)$$

To reduce the latency of estimating X , we have recently proposed to solve the linear system in fixed-point arithmetic with reduced precision [31] through an eigenvalue decomposition of the Hermitian matrix Θ , as shown in Equation (4). It should be observed that the eigenvalue decomposition is intrinsically stable, since unitary matrix Q is a rotation matrix, which means that all its elements range from -1.0 to $+1.0$.

$$\Theta = Q \cdot \Lambda \cdot Q^H \Rightarrow \tilde{X} = Q \cdot \Lambda^{-1} \cdot Q^H \cdot H^H \cdot Y \quad (4)$$

C. Coarse Quantization

When signal x is in the range of the quantization range i , $q_{min}^i \leq x < q_{max}^i$, its reported quantized value is $q(x) = q_i$. The values of q_i for all the intervals i , which minimize the mean square error, $E[|q(x) - x|^2]$, where $E[\cdot]$ is the expectation operator, are easily shown to be the mean value of the signal in each interval $q_i = E[x | q_{min}^i \leq x < q_{max}^i]$ [11].

An independent Additive White Gaussian Noise (AWGN) channel with mean $\mu = 0$ and standard deviation, $\sigma = 1$, is assumed for each receive antenna in this article. Table I presents the probabilities, quantization thresholds, and the corresponding bin-average values for $B = 1$ -bit, $B = 2$ -bit, and $B = 3$ -bit quantization processes for a Gaussian random variable with zero mean and unity standard deviation.

TABLE I
PROBABILITIES, QUANTIZATION THRESHOLDS, AND THE
CORRESPONDING BIN-AVERAGE VALUES.

B	Probabilities	Quantization thresholds	Bin-average values
1	50%	0	± 0.797884561
2	25%	0 ± 0.674489572	± 0.324662678 ± 1.271106444
3	12.5%	0 ± 0.318634923 ± 0.674489572 ± 1.150349346	± 0.157972012 ± 0.491353344 ± 0.895384581 ± 1.646828306

In fixed-point representation, it is beneficial to scale up the smallest magnitude of the bin-average values, since this will simplify the implementation and slightly increase the precision. For example, in a $B = 2$ quantization process, 0.324662678 is promoted to 1.000000000 , and 1.271106444 is promoted to $1.271106444/0.324662678 = 3.915160350$. The other quantization processes will follow a similar pattern. The updated values are presented in Table II.

TABLE II
QUANTIZATION THRESHOLDS, SCALE FACTORS, AND THE
CORRESPONDING BIN-AVERAGE VALUES.

B	Quantization thresholds	Scale factor	Bin-average values
1	0	0.797884561	± 1.000000000
2	0 ± 0.674489572	0.324662678	± 1.000000000 ± 3.915160350
3	0 ± 0.318634923 ± 0.674489572 ± 1.150349346	0.157972012	± 1.000000000 ± 3.110382264 ± 5.667995047 ± 10.424810617

It is apparent that the measurements of the channel output can take only a small number of different values (for example, ± 1.0000 and ± 3.9152 in a process with $B = 2$ quantization bits). The estimated channel coefficients, the entries of H , are also coarsely quantized to reduce the computation cost in the receiver. Since the quantized values have equal probability, the entropy of the quantized signal is maximized, which in turn maximizes the upper bound of the mutual information between the received signal and the transmitted signal when their joint distribution is not known [12].

D. FPGA Architecture

Modern FPGA architectures consist of five types of modules: I/O Blocks (IOB), Configurable Logic Blocks (CLB), Digital Signal Processing (DSP) slices, Block Random Access Memories (BRAM), and a configurable Interconnection Network. The CLBs belonging to the fine-grained fabric are organized as a 2D array, where each CLB includes a configurable Look-Up Table (LUT) to implement bit-level logic functions, dedicated carry logic to support arithmetic operations such as binary and ternary adders [27], dedicated multiplexors, and Flip-Flops (FF). Software support for the fine-grained fabric includes macros and primitives, such as *CARRY4* which concatenates four LUTs to build a 4-bit binary/ternary adder or subtractor, *MUXF7* and *MUXF8* which instantiate 2-to-1 multiplexors, *BRLSHFT4* and *BRLSHFT8* which instantiate 4-bit and 8-bit barrel shifters, etc. The DSP slices belong to the coarse-grained fabric and support the implementation of large units such as 25×18 two's-complement multipliers, 48-bit accumulators, dual 24-bit or quad 12-bit adders, subtractors, and accumulators. The configurable interconnection network connects these the modules together to complete the implementation of digital circuits.

Modern FPGAs also integrate Block Random Access Memories (BRAM) on chip. The BRAM in the Virtex-7 family [29] can operate as either *one* 36 Kb dual-port RAM, which can be configured as $32K \times 1$, $16K \times 2$, $8K \times 4$, $4K \times 9$, $2K \times 18$, $1K \times 36$, or 512×72 , or *two independent* 18 Kb dual-port RAMs, where each RAM can be configured as $16K \times 1$, $8K \times 2$, $4K \times 4$, $2K \times 9$, $1K \times 18$, or 512×36 . These BRAMs can be used as large storage area or as large LUTs with multiple outputs to implement logic functions.

III. MASSIVE MIMO UPLINK COMMUNICATIONS WITH COARSE QUANTIZATION

In this article, we consider the receivers in the base stations of massive MIMO communications systems. The number of transmitters (or users) is denoted as T , each of which has one antenna. The base station is equipped with R antennas. The transmit and receiver antennas are spaced so that the channel propagation gains from each pair of transmit and receive antennas are independent. A spacing greater than one half of a wavelength between antennas can accomplish this independence. For a communication frequency of 1 GHz, which corresponds to a wavelength of $\lambda = 30$ cm, an array of 256 antennas can be organized as a square with the side length equal to $\sqrt{256} \times \frac{30}{2} = 240$ cm = 2.4 m. Such an array can be mounted, for example, on the roof of a city building.

If the measured signal on the receiver antennas for sample time n is held in a length R vector denoted as $\mathbf{y}[n]$, with the full received signal being denoted as

$$\mathbf{y}[n] = \sum_{l=0}^{L-1} \mathbf{h}[l] \mathbf{x}[n-l] + \mathbf{v}[n], \quad (5)$$

where $\mathbf{h}[l]$ is the R -by- T matrix holding channel gains from the transmitters to receivers with the entry in column t on row r specifying the propagation gain from transmitter t to receive antenna r for the delay of l sample periods, then $\mathbf{x}[n]$ is a

length T vector specifying the transmitted signals at sample time n , $\mathbf{v}[n]$ is a length R vector giving the measurement noise at sample n , and L is the length of the channel impulse responses in sample periods. The problem of the receiver is to estimate the vectors $\mathbf{x}[n]$ from the received signals $\mathbf{y}[n]$. A method of accomplishing this in modern wireless systems, such as advanced Wireless Local Area Networks (WLANs) or 5G advanced radio networks, is the use of Frequency Domain Equalization (FDE) [17]. For FDE, the received signals for all receiver antennas are first split into blocks of N samples. The key to OFDM is to compute the Discrete Fourier Transform (DFT) of the received signal $\mathbf{y}[n]$ as

$$\mathbf{Y}[k] = \sum_{n=0}^{N-1} \mathbf{y}[n] \exp\left(-j \frac{2\pi kn}{N}\right) \text{ for } k = 0 \dots N-1 \quad (6)$$

where N is the length of the DFT where a value of $N = 256$ is often used. If a cyclic prefix of $CP > L$ samples from the end of each block of N samples is copied and prepended to the start by the transmitter, as is performed in the OFDM signalling method used in most modern wireless standards, then a convolution in time is converted into a multiplication in the DFT domain. Thus, the relation from (5) becomes:

$$\mathbf{Y}[k] = \mathbf{H}[k] \mathbf{X}[k] + \mathbf{V}[k] \text{ for } k = 0, \dots, N-1 \quad (7)$$

where $\mathbf{H}[k]$, $\mathbf{X}[k]$, and $\mathbf{V}[k]$ are the DFTs of time-domain signals $\mathbf{h}[l]$, $\mathbf{x}[n]$, and $\mathbf{v}[n]$, respectively (Equation 1 is one out of N instances of Equation 7 for an unspecified frequency index k). It is apparent that the MIMO signalling over multipath radio channels becomes a simple matrix-vector multiplication in the frequency domain. The advantage of MIMO-OFDM signalling with FDE is that the equalization of the multiple antenna, multipath radio propagation channel can be reduced to linear algebra [18] where the methods described above can be applied.

For MIMO systems, each antenna $r = 1, \dots, R$ requires a radio receiver. In Massive MIMO (MMIMO) systems, where R is very large, this can lead to a prohibitive cost. If high resolution Analog-to-Digital Converters (ADCs) and highly-linear RF amplifiers are used, they can make up a large proportion of this cost. However, we can show that only inexpensive devices are needed for many MMIMO systems. This article considers an implementation of MMIMO in terms of Bit-Error Rate (BER) using both demodulation linked with an error correction code as opposed to [1]–[3] which only present capacity calculations. Here we present a demonstration of a MMIMO system performance showing that highly quantized measurements with lower precision calculations can give acceptable BER performance.

For the sake of presentation, the numbers of transmit and receive antennas will be powers of 2. A MMIMO system in a Standard Configuration (SC), as we propose to name it, has $T = 16$, $R = 128$. Other configurations can be of practical interest, for example Extended Users (EU) with $T = 32$, $R = 128$, Extended Base Station (EBS) with $T = 16$, $R = 256$, Extended Configuration (EC) with $T = 32$, $R = 256$, and Maximum Configuration (MC) with $T = 64$, $R = 256$, but they will not be considered in this article.

The number of taps L required to model the radio channel for acceptable receiver performance is determined by the difference in propagation distance between the shortest radio signal path and the longest radio signal path with significant received power. This distance is directly related to the so-called delay spread of the radio propagation channel after division by the radio signal propagation speed. The tolerated propagation distance difference is given by $L \times T_s \times c$ where T_s is the sampling time and the c is radio signal propagation speed which is approximately 3×10^8 m/s. For the radio systems of greatest interest, such as 5G cellular and advanced wireless local area networks, $L = 9$ and $T_s \approx 10^{-7}$ seconds [19]. This indicates that the system described in this article tolerates a difference of propagation distance up to $9 \times 10^{-7} \times 3 \times 10^8 = 270$ meters, which is sufficient for most urban microcells or indoor networking applications.

An MMIMO system in Standard Configuration was simulated using 4-QPSK modulation having a standard rate 1/2 convolutional error correction code with a constraint length of 7. We have assumed that the receiver has access to ideal channel state information, since it has been shown that with pilot-signals and iterative receiver algorithms, the channel estimation error can be made smaller than the measurement noise [9]. In practice, several methods exist for MIMO channel estimation [20]. The radio channel from each transmitter to receive antenna has a random propagation path with a length of $L = 9$ sample periods. In the Standard Configuration of $R = 128$ and $T = 16$ for a total received energy over all receivers per data bit over measurement noise density at each receiver, E_b/N_0 , of 5.15 dB an acceptable Bit Error Rate (BER) after coding of about 10^{-6} is achieved using a simulated ideal ADC with zero quantization error [9]. This provides an ideal system figure of merit to compare our systems with quantized measurements with. Given that $R/T = 8$, the signal-to-noise ratio (SNR) on each receive antenna is equal to $5.15 \text{ dB} - 10 \log 8 = 5.15 \text{ dB} - 9 \text{ dB} = -3.85 \text{ dB}$. This indicates that the noise power is significantly greater than the information signal power on each individual antenna, so that if a high resolution ADC is employed most of its output bits are nearly independent of the information signal. This provides the motivation for the use of coarse quantization, as for high-resolution quantizers the bits of lower significance are mostly independent of the data signal.

The system simulations are run with lower resolution ADCs to see the effects on the BER. For 4-bit quantization ($B = 4$) at 5 dB of E_b/N_0 , the BER is increased to 1.2×10^{-6} with an additional 0.05 dB of power being required to match the BER performance of 10^{-6} for ideal non-quantized measurements. For 3-bit quantization ($B = 3$), the BER is increased to 2.2×10^{-6} with an additional 0.22 dB of power required to match the BER performance of ideal non-quantized measurements. For 2-bit quantization ($B = 2$), the BER is increased to about 2×10^{-5} with an additional 0.85 dB of power required to match the performance of ideal non-quantized measurements. Finally, for 1-bit quantization ($B = 1$), the BER is increased to 4×10^{-3} with an additional 3 dB of power required to match the performance of ideal non-quantized measurements. It is apparent that there is not much gain in the BER to operate

above a 4-bit quantization. The additional 3 dB of power needed by a 1-bit quantization can be compensated out by doubling the number of receive antennas (which is mechanically feasible). A good trade-off is the 2-bit quantization ($B = 2$), which is used in this article as a case study to discuss the implementation of the data detection and decoding process.

IV. DETECTION IN FIXED-POINT ARITHMETIC

In this section the major operations of the receiver algorithm implemented in fixed-point arithmetic (calculating the Hermitian matrix Θ , its Hessenberg decomposition, and the eigenvalue decomposition through a numerically stable algorithm) are discussed, and numerical results for Standard Configuration ($T = 16$, $R = 128$) are presented.

A. Calculating $\Theta = H^H \cdot H$ in Fixed-Point Arithmetic

The computational pattern for calculating the matrix $\Theta = H^H \cdot H$ is sum-of-products, and the computation budget consists of T^2 inner products of R -element complex vectors. The inner product of two R -element complex-valued column vectors $\mathbf{X}$ and $\mathbf{Y}$

$$\begin{aligned} \mathbf{X} &= [x'_1 + jx''_1, \dots, x'_R + jx''_R]^T \\ \mathbf{Y} &= [y'_1 + jy''_1, \dots, y'_R + jy''_R]^T \end{aligned} \quad (8)$$

is given by

$$\begin{aligned} \mathbf{X}^H \cdot \mathbf{Y} &= [(x'_1 y'_1 + x''_1 y''_1) + \dots + (x'_R y'_R + x''_R y''_R)] \\ &\quad - j[(x''_1 y'_1 - x'_1 y''_1) + \dots + (x''_R y'_R - x'_R y''_R)] \end{aligned} \quad (9)$$

In coarse quantization with $B = 2$ -bit precision, only the values ± 1.0000 and ± 3.9152 can occur, as shown in Subsection II-C. Their encoding can be, for example, **10**, **11**, **00**, and **01**. It is apparent that these codes do not represent numerical values; they are merely labels. As a result, arithmetic cannot be directly implemented, and parallel counters [21]–[26] cannot be directly used to perform the multioperand additions required for the calculation of inner products. This problem is addressed below. In any case, the FPGA I/O data transfer time is significantly reduced due to this encoding style.

To calculate $\mathbf{X}^H \cdot \mathbf{Y}$, the products of two B -bit quantized values (such as $x'_1 y'_1$) are first calculated. The number of product values is also small, reaching 6 for $B = 2$ (± 1.0000 , ± 3.9152 , and ± 15.3288 , which can be encoded with 3 bits). Each product is a 4-bit logic function (2 bits for each of the multiplicands and multipliers). Its three bits can be calculated in parallel with three LUTs (one LUT per bit). It should be observed that these products are also labels corresponding to values, not the standard numerical values.

Collapsing these products is performed in two stages. In the first stage, BRAMs, which are each configured as a $4K \times 9$ memory element, are used as look-up tables. Four 3-bit products are concatenated to form a 12-bit address into a BRAM, which in turn returns their sum. The maximum 4-product sum is 4×15.3288 , which is a 7-bit quantity (sign bit included), if only the whole part is retained. However, since a BRAM provides a 9-bit output, two additional bits of precision

are available at no additional cost, making the sum a 9-bit quantity (sign bit included).

The advantage of using BRAMs is that the collapsing of four products into one larger sum is performed simultaneously with their conversion from labels to numerical values. In Equation (9) it is apparent that $2R = 256$ 3-bit products per real component (and $2R = 256$ 3-bit products per imaginary component) need to be collapsed. This means that $2 \times \frac{2R}{4} = 128$ BRAM accesses per complex-valued inner product are needed. In the second stage, the remaining $\frac{2R}{4} = 64$ sums in each of the real and imaginary components will be added together by a tree of ternary ripple-carry adders [27]. The final sum is a $9 + \log_2 64 = 15$ -bit quantity (2 fractional bits and 1 sign bit are assigned to the value) in the extreme case that all values have maximum magnitude and add up constructively. This translates into a tree of $22 + 8 + 3 + 1 = 34$ ternary adders with the wordlength of 15 bits for each real and imaginary component, which requires an estimate of $2 \times 15 \times 34 = 1020$ LUTs. As a result, the latency of an inner product is the sum of a LUT delay (for calculating the initial 3-bit products), a BRAM delay (for calculating the sum of four products), and the delays of four 15-bit ripple-carry adders (for collapsing the remaining sums). The two fractional bits of the final sum are removed through right shift and rounding; thus, the final sum is a 13-bit integer. Since Θ is Hermitian, its diagonal components are real, so the diagonal imaginary components do not need to be calculated. In addition, the entries above the main diagonal are the complex conjugates of the entries below the main diagonal, which means that the number of operations can be further reduced.

Due to their length and random nature, data signals of different users are likely to have very low correlation with each other. For this reason, Θ is diagonal dominant. A sample of the reduced-wordlength matrix Θ in a MMIMO with $R = 128$ and $T = 16$ is shown below. It is observed that its diagonal elements are real-valued and positive; in comparison, the real and imaginary components of off-diagonal elements are significantly smaller in magnitude. Statistics collected over one million samples of matrix Θ indicate that the largest off-diagonal magnitude has an average of 49 and a standard deviation of 6. This means that most off-diagonal values can be represented with 7 bits (sign bit included). The overflowing off-diagonal values will be saturated to ± 63 .

$$\Theta = \begin{pmatrix} 289 & 5+j & 4 & -11-j & 2 & 7+j & 6 & \dots \\ 5-j & 4 & 244 & 3+j & 1 & 6-j & 8 & \dots \\ -11+j & 2 & 3-j & 1 & 211 & -9-j & 13 & \dots \\ 7-j & 6 & 6+j & 8 & -9+j & 13 & 208 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad (10)$$

It is apparent that a wordlength of 9 bits provides sufficient precision to store each element of the matrix Θ . For higher quantization levels a longer wordlength may be required. However, if needed, a reduction of the wordlength could be performed in such cases by right shifting followed by truncation or rounding.

The channel matrix, H , has $R \times T$ complex-valued elements. This translates into a storage requirement of $2 \times$

$R \times T \times B$ bits. In standard configuration with $B = 16$ -bit precision, a number of $2 \times 128 \times 16 \times 16 = 65536$ bits will have to be downloaded and stored into FPGA. Given that the XC7VX1140T, which is the largest FPGA in the top of the line family of Virtex-7, has *only* 1100 I/O pins, all those bits cannot be downloaded in parallel. The benefits of the coarse quantization are apparent, as both the channel state transfer time and storage capacity requirement are highly reduced.

A channel matrix H in the standard configuration has $R \times T = 128 \times 16 = 2048$ complex-valued elements. Even with coarse quantization, the number of FPGA I/O pins is too small to accommodate the transfer of all these elements in parallel. A single column, which includes $R = 128$ complex-valued elements (or $2R = 256$ real-valued elements), would require $2 \times 256 = 512$ I/O pins with a coarse quantization with $B = 2$ bits. Since the XC7VX1140T FPGA, which is the top of the line in the Virtex-7 family, has 1100 I/O pins, it is difficult to download the entire matrix H in parallel, since not many pins would remain available for other tasks. As a result, multiplexing is needed. It is mentioned that Serializer/Deserializer (SERDES) techniques [28] are intentionally not used in order to keep the digital hardware of each receive antenna as simple and inexpensive as possible.

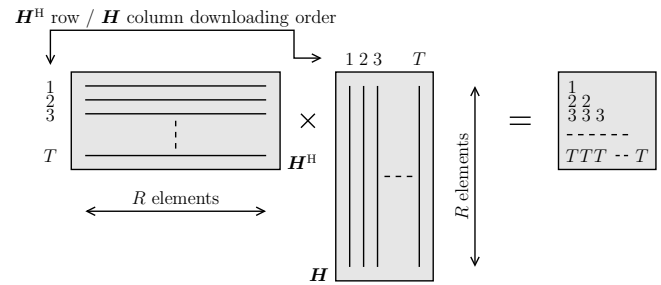


Fig. 1. Calculating $\Theta = H^H H$.

Figure 1 shows the strategy of calculating $\Theta = H^H H$. The elements of each column of H are loaded into the FPGA in parallel through separate I/O pins, and the columns are loaded sequentially. Column 1 of H (Row 1 of H^H) is first downloaded, allowing the inner product producing the top-left element, which is labeled 1, to be calculated. Column 2 of H (Row 2 of H^H) is downloaded next, allowing the inner products producing elements labeled 2 to be calculated. After the last column of H (last row of H^H) is downloaded, the inner products producing elements labeled T in the output matrix can be calculated. It is observed that each of the elements in a row of Θ can be calculated in parallel. The ability of exploiting this data-level parallelism depends on the FPGA logic capacity.

The BRAM is a true dual-port memory, which means that only $\frac{128}{2} = 64$ BRAM units are needed for each inner product. This means that $T \cdot 64 = 16 \cdot 64 = 1024$ BRAMs are needed for calculation of the entire matrix Θ . This hardware cost is significant. The number of BRAMs can be reduced by calculating the first eight columns of Θ in a first phase, and the last eight columns of Θ in a second phase. This technique will reduce the number of BRAM units to $\frac{1024}{2} = 512$, but will

require iterating through the last eight columns of $\mathbf{H}$ again. The matrix's upper triangular portion is the complex conjugate of the lower triangular portion, and thus computing its values does not require additional operations.

Each inner product is implemented with a pipeline of three stages: (i) LUT plus BRAM, (ii) two 15-bit ripple-carry adders, and (iii) two 15-bit ripple-carry adders. There are $T = 16$ rows, out of which 8 will be accessed twice, as mentioned in the previous paragraph. This means that $16 + 8 + (3 - 1) = 26$ pipeline stages are needed to complete the calculation of Θ . Additional stages will be needed by housekeeping routines. The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [30] indicate that matrix Θ can be calculated with a clock frequency of at least 100 MHz and with a latency of 50 cycles. The hardware utilization is 512 BRAMs, 23190 LUTs, and 8464 flip-flops.

B. Hessenberg Decomposition of Θ in Fixed-Point Arithmetic

It is well known that a Hessenberg form is preserved during a similarity transformation [4]. For this reason, a conversion from a Hermitian form, Θ , to a Hessenberg form, $\mathbf{A}_{\text{hess}} = \mathbf{Q}_{\text{hess}} \cdot \Theta \cdot \mathbf{Q}_{\text{hess}}^H$, is beneficial to be performed prior to applying the Francis-Kublanovskaya recursion, since this will significantly reduce the operation count [4]. It is observed that the reduced off-diagonal wordlength is highly beneficial for the fixed-point arithmetic since only off-diagonal elements are rotated in the Hessenberg conversion. For example, the first-column vector $\mathbf{v}$ (diagonal element is excluded), where $\mathbf{v}^T = [5 - j4, -11 + j2, 7 - j6, \dots]$, is converted into a vector $\mathbf{w}$, where $\mathbf{w}^T = [59 - j62, 0, 0, \dots]$. This vector still has a small magnitude compared to diagonal elements, which means that the Hessenberg form is also diagonal dominant. Moreover, since the off-diagonal elements of a Hessenberg form decrease in magnitude over Francis-Kublanovskaya iterations [13]–[15], there is no danger of overflow in any of those elements. According to our simulations, a wordlength of 8 bits can represent with sufficient precision the unitary matrices, $\mathbf{Q}_{\text{hess}}$, leading to Hessenberg and diagonal forms.

Calculating the Hessenberg/triagonal form of the Hermitian matrix Θ is performed with Householder reflectors. A Householder reflector of a complex-valued column vector $\mathbf{z} = [z_1, z_2, \dots]^T$, where $z_i \in \mathbb{C}, \forall i$, is expressed in terms of a Hermitian unitary matrix, $\mathbf{F}$, which is constructed as:

$$\mathbf{F} = \mathbf{I} - 2 \frac{\mathbf{v} \cdot \mathbf{v}^H}{\mathbf{v}^H \cdot \mathbf{v}}, \quad (11)$$

where:

$$\begin{aligned} \mathbf{v} &= \text{sgn}(z_1) \|z\| \mathbf{e}_1 - \mathbf{z} \\ \mathbf{e}_1 &= [1, 0, \dots]^T \\ \text{sgn}(z_1) &= \frac{z_1}{|z_1|} \quad (\text{complex signum}) \end{aligned} \quad (12)$$

A Householder reflector forces all vector elements with the exception of the first one to zero, as shown below.

$$\mathbf{F} \cdot \mathbf{z} = \text{sgn}(z_1) \|z\| \mathbf{e}_1 \quad (13)$$

In the above equations it is apparent that a Householder reflector relies on the squared Euclidean norm and the square

root operation. Since these two operations are on the critical path of the computation, it is critical to minimize their latencies. This will be addressed in the next two subsections.

C. Squared Euclidean Norm

To simplify the presentation, the Squared Euclidean Norm of a K -element complex-valued vector $\mathbf{X}$ with elements $X_{\text{re},k} + j X_{\text{im},k}$ is written as the Squared Euclidean Norm of a $2K$ -element real-valued vector with elements X_i :

$$\|\mathbf{X}\|^2 = \sum_{k=1}^K X_{\text{re},k}^2 + X_{\text{im},k}^2 = \sum_{i=1}^{2K} X_i^2 \quad (14)$$

The two's complement representation of each element X_i is given by:

$$\begin{aligned} X_i &= \overline{s_i x_{i,N-1} \dots x_{i,1} x_{i,0}} = \\ &= -2^{Nr} s_i + 2^{(N-1)r} x_{i,N-1} \dots + 2^r x_{i,1} + x_{i,0} \end{aligned} \quad (15)$$

where $x_{i,N-1}, \dots, x_{i,1}, x_{i,0}$ are digits in a numeral system of radix r and s_i is the sign bit. In coarsely quantized MMIMO systems, the precision of the off-diagonal elements of matrix Θ is on the order of a few bits. It is apparent that a two's complement representation with $r = 3$ (octal digits) and a precision of $N = 2$ digits plus a sign bit (which corresponds to a 7 bit two's complement integer) is sufficient to encode those off-diagonal elements shown in Equation 10. Therefore, the square of an element X_i is given by:

$$\begin{aligned} X_i^2 &= (-2^{2 \cdot 3} s_i + 2^3 x_{i,1} + x_{i,0})^2 = 2^{12} s_i^2 + 2^6 x_{i,1}^2 \\ &+ x_{i,0}^2 - 2^{10} s_i x_{i,1} - 2^7 s_i x_{i,0} + 2^4 x_{i,1} x_{i,0} \end{aligned} \quad (16)$$

where, obviously, $s_i^2 = s_i$. The last product, $x_{i,1} x_{i,0}$, can be decomposed in a similar fashion. Assume that $b_{i,02}, b_{i,01}$, and $b_{i,00}$ are the three bits of the octal digit $x_{i,0}$:

$$x_{i,0} = \overline{b_{i,02} b_{i,01} b_{i,00}} \quad (17)$$

Then, one can write:

$$x_{i,1} x_{i,0} = 2^2 b_{i,02} x_{i,1} + 2 b_{i,01} x_{i,1} + b_{i,00} x_{i,1} \quad (18)$$

It is observed that the negative terms in Equation (16) and all the terms in Equation (18) are products of an octal digit ($x_{i,1}$ and $x_{i,0}$) by a binary digit ($s_i, b_{i,02}, b_{i,01}$, and $b_{i,00}$). The importance of this representation will become apparent below. The sum of squares in Equation (14) can be written as

$$\begin{aligned} \sum_{i=1}^{2K} X_i^2 &= 2^{12} \underbrace{\sum_{i=1}^{2K} s_i}_{\Sigma_1} + 2^6 \underbrace{\sum_{i=1}^{2K} x_{i,1}^2}_{\Sigma_2} + \underbrace{\sum_{i=1}^{2K} x_{i,0}^2}_{\Sigma_3} \\ &- 2^{10} \underbrace{\sum_{i=1}^{2K} s_i x_{i,1}}_{\Sigma_4} - 2^7 \underbrace{\sum_{i=1}^{2K} s_i x_{i,0}}_{\Sigma_5} + 2^6 \underbrace{\sum_{i=1}^{2K} b_{i,02} x_{i,1}}_{\Sigma_6} \\ &+ 2^5 \underbrace{\sum_{i=1}^{2K} b_{i,01} x_{i,1}}_{\Sigma_7} + 2^4 \underbrace{\sum_{i=1}^{2K} b_{i,00} x_{i,1}}_{\Sigma_8} \end{aligned} \quad (19)$$

In a Householder reflector for matrix Θ , $K \leq T - 1$. In MMIMO standard configuration $T = 16$; therefore, $K \leq 15$. The sum Σ_1 (which is a 5-bit quantity) is calculated with a parallel counter in two stages. First, sums of six s_i bits are calculated with three 6-input LUTs per sum, since each such a sum is a 3-bit quantity; thus, no carry propagation occurs and the hardware requirement is $3 \times \lceil \frac{2K}{6} \rceil = 15$ LUTs. In the second stage the $\lceil \frac{2K}{6} \rceil = 5$ resulting 3-bit sums can be collapsed with two ternary ripple-carry adders [27] requiring 4 and 5 LUTs, respectively. According to our VHDL code synthesized and simulated with Vivado [30], ternary adders are instantiated out of behavioral code only if the bitwidth ≥ 7 . For bitwidth ≤ 6 , ternary adders are instantiated by coding with Xilinx primitives.

Sums Σ_2 and Σ_3 (11-bit quantities) have the same computational pattern. Σ_2 , for example, can be expanded as

$$\Sigma_2 = \sum_{j=1}^K \underbrace{x_{(2j-1),1}^2 + x_{(2j),1}^2}_{\Sigma'_{2,j}} \quad (20)$$

Since $x_{(2j-1),1}$ and $x_{(2j),1}$ are octal digits, each sum $\Sigma'_{2,j}$ (a 7-bit quantity) is a 6-input logic function and can be implemented with seven 6-input LUTs (no carry propagation occurs). $7 \times K = 105$ LUTs are needed in this stage. In the second stage $K = 15$ 7-bit arguments are to be collapsed. This is achieved by a tree of $5 + 1 + 1 = 7$ ternary ripple-carry adders, which require up to $7 \times 11 = 77$ LUTs.

The sums $\Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7$, and Σ_8 are 8-bit quantities and have the same computational pattern, which is sum of products of an octal digit by a binary digit. Σ_4 , for example, can be expanded as

$$\Sigma_4 = \sum_{j=1}^{\lceil \frac{2K}{3} \rceil} \underbrace{s_{3j-2} x_{(3j-2),1} + s_{3j-1} x_{(3j-1),1} + s_{3j} x_{(3j),1}}_{\Sigma'_{4,j}} \quad (21)$$

It is apparent that each sum $\Sigma'_{4,j}$ is a 5-bit value, and can be regarded as having three octal arguments ($x_{(3j-2),1}, x_{(3j-1),1}, x_{(3j),1}$) and three mask bits ($s_{3j-2}, s_{3j-1}, s_{3j}$). It can be implemented with a 4-bit ternary ripple-carry adder, that is, with a CARRY4 primitive and four 6-input LUTs in which the mask bits will drive three out of those six inputs. $4 \times \lceil \frac{2K}{3} \rceil = 40$ LUTs are needed in this stage. In the second stage $\lceil \frac{2K}{3} \rceil = 10$ four-bit arguments are to be collapsed. This is achieved by a tree of $3 + 1 + 1 = 5$ ternary ripple-carry adders, which requires up to $5 \times 8 = 40$ LUTs.

In the final stage, sums $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7$, and Σ_8 are collapsed to give the squared Euclidean norm, $\|X\|^2$ (a 16-bit quantity). This is achieved by a tree of $3 + 1 = 4$ ternary ripple-carry adders, which requires up to $4 \times 16 = 64$ LUTs. Long carry propagation over 12 bits, as it is apparent in Equation (19), occurs only in this final stage.

The source code was implemented in VHDL and Xilinx Design Constraints (XDC) were carefully specified. Simulations carried out with Vivado [30] indicate that the squared Euclidean norm can be calculated with a clock frequency of 100 MHz and with a latency of 5 cycles with timing slack

to spare. The hardware utilization includes 1160 LUTs and 235 flip-flops. This is a good result compared to the behavioral solution in implementing Equation (14) by collapsing $2K = 30$ squared 6-bit operands, whose latency would be approximately 50% larger than the proposed solution and would require over 2000 LUTs.

D. Square Root Operation

A number of $T - 2 = 14$ Householder reflectors need to be calculated for each matrix Θ . Since the square root operation is on the critical path of the Householder reflector, it is imperative to reduce its latency. Thus, recursive bit-level solutions, such as the Convergence Computing Method [32], would be too slow for a MMIMO application. A parallel implementation in fixed-point/integer arithmetic is sought.

Statistics collected over one million samples of matrix Θ and the resulting Hessenberg form indicate that the sum-of-squares of lower off-diagonal elements in a column can be represented with a bitwidth of 16 bits or less. In integer arithmetic, when only the integer bits are retained, the square root of 16-bit argument has a wordlength of 8 bits. In the most general case, the square root of an integer X has an integer part, $\lfloor Y \rfloor$, and a fractional part, $\{Y\}$:

$$\sqrt{X} = Y = \lfloor Y \rfloor + \{Y\}, \quad \text{where } \{Y\} \in [0, 1) \quad (22)$$

In order to keep the truncation error in the range $(-0.5, 0.5)$, the square root implemented in integer arithmetic is approximated as:

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } \{Y\} < 0.5 \\ \lfloor Y \rfloor + 1 & \text{if } \{Y\} > 0.5 \end{cases} \quad (23)$$

It is apparent that the fractional part, $\{Y\}$, can never be equal to 0.5, since in that case the argument X , which is an integer, would have a fractional part $\{X\} = 0.25$:

$$\begin{aligned} \sqrt{X} = Y = \lfloor Y \rfloor + 0.5 &\Rightarrow \\ \Rightarrow X = (\lfloor Y \rfloor + 0.5)^2 = \lfloor Y \rfloor^2 + \lfloor Y \rfloor + \underbrace{0.25}_{\{X\}} &\quad (24) \end{aligned}$$

This fractional part is small. Therefore, with a very good approximation:

$$\sqrt{\lfloor Y \rfloor^2 + \lfloor Y \rfloor} \approx \lfloor Y \rfloor + 0.5 \quad (25)$$

As a result, Equation (23) can be replaced by Equation (26), which in turn can be more easily converted into LUT configuration information. For example, consider the threshold in the square root domain $Y = 251.5 \Rightarrow \lfloor Y \rfloor = 251$. This translates into a threshold in the argument domain of $\lfloor Y \rfloor^2 + \lfloor Y \rfloor = 63252$. Then, the argument values $X \geq 63252$ will map into $\sqrt{X} = 252$, and the argument values $X < 63252$ will map into $\sqrt{X} = 251$. In a second example, the threshold in the square root domain $Y = 131.5 \Rightarrow \lfloor Y \rfloor = 131$ translates into a threshold in the argument domain of $\lfloor Y \rfloor^2 + \lfloor Y \rfloor = 17292$. Then, the argument values $X \geq 17292$ will map into $\sqrt{X} = 132$, and the argument values $X < 17292$

will map into $\sqrt{X} = 131$. All the other thresholds will follow a similar pattern.

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } X < \lfloor Y \rfloor^2 + \lfloor Y \rfloor \\ \lfloor Y \rfloor + 1 & \text{if } X \geq \lfloor Y \rfloor^2 + \lfloor Y \rfloor \end{cases} \quad (26)$$

In a brute force implementation, the integer part of the square root is retrieved from a large lookup table built with BRAM units. Since the argument and square root are 16-bit and 8-bit integers, respectively, the LUT size needs to be 2^{16} bytes or 64KB. Such a large LUT is implemented with sixteen $32K \times 1$ BRAMs (two BRAMs per output bit). The square root latency is one BRAM delay, thus it is very short, but the hardware cost is large. A technique to reduce the LUT size is presented next.

It is observed that $\lfloor Y \rfloor^2 + \lfloor Y \rfloor$ is an even number. This means that only 15 bits are in fact needed to form the LUT address, and the square root approximation can be written as:

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } (X \gg 1) < [(\lfloor Y \rfloor^2 + \lfloor Y \rfloor) \gg 1] \\ \lfloor Y \rfloor + 1 & \text{if } (X \gg 1) \geq [(\lfloor Y \rfloor^2 + \lfloor Y \rfloor) \gg 1] \end{cases} \quad (27)$$

Consider the same example of the threshold in the square root domain $Y = 251.5 \Rightarrow \lfloor Y \rfloor = 251$. By removing the least significant bit, argument values $(X \gg 1) \geq (63252 \gg 1) = 31626$ will map into $\sqrt{X} = 252$, and argument values $(X \gg 1) < 31626$ will map into $\sqrt{X} = 251$. This strategy reduces the number of BRAMs from sixteen to eight, thus halving the cost of the implementation.

In the described implementation, it is apparent that the maximum rounding error is slightly above 0.5, since that small fraction of 0.25 in Equation (24) is lost in approximating $X \approx \lfloor Y \rfloor^2 + \lfloor Y \rfloor$. In order to limit the maximum rounding error to exactly 0.5, the approximation in Equation (26) needs to be rewritten as:

$$\sqrt{X} \approx \begin{cases} \lfloor Y \rfloor & \text{if } X \leq \lfloor Y \rfloor^2 + \lfloor Y \rfloor \\ \lfloor Y \rfloor + 1 & \text{if } X > \lfloor Y \rfloor^2 + \lfloor Y \rfloor \end{cases} \quad (28)$$

Considering the same numerical example, that means that the argument values $X > 63252$ will have to map into $\sqrt{X} = 252$, and the argument values $X \leq 63252$ will have to map into $\sqrt{X} = 251$. In this case, the right shift operation needs to be accompanied with rounding, such that $63253 \gg 1 = 31627$. The implementation is shown in Figure 2. The latency of this implementation is given by the delay of a 15-bit adder and a BRAM delay, which is a very low. The hardware cost is eight BRAMs plus a 15-bit adder.

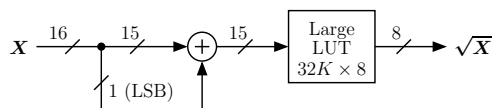


Fig. 2. Square root implemented with a large LUT.

The squared Euclidean norm and the square root, which are on the critical path of the Householder reflector implementation, as it is apparent in Equations (11) and (12), are implemented with LUTs and BRAMs. The application of a

Householder reflector, $F_i \cdot \Theta$, where $i = 1, 2, \dots, T-2$, and the accumulation of the successive Householder reflectors into a global unitary matrix, Q_{hess} , is implemented with DSP slices.

E. Francis-Kublanovskaya Algorithm in Fixed-Point Arithmetic

The eigenvalue decomposition of matrix Θ is performed in two steps: (i) initial transformation of Θ into a Hessenberg matrix A_{hess} , and (ii) diagonalization of A_{hess} through the Francis-Kublanovskaya (FK) recursion [13]–[15], in which each iteration consists of a QR decomposition followed by a reverse product. The QR decomposition is performed through rotating the 2-element vertical vector in each column such that the off-diagonal element in that column is forced to zero. This rotation can also be implemented with a Householder reflector.

Since the FK algorithm can diagonalize the matrix Θ directly, although with a much larger number of operations, the matrices Θ and A_{hess} are used interchangeably to simplify the remaining part of the presentation. At iteration k , the matrix $\Theta^{(k)}$ (where $\Theta^{(0)} = \Theta$) is decomposed into the product of a unitary matrix $Q^{(k)}$ and an upper triangular matrix $R^{(k)}$. By multiplying $R^{(k)}$ by $Q^{(k)}$, a similarity transformation of $\Theta^{(k)}$ is obtained, as shown in Equation 29.

$$\Theta^{(k+1)} = R^{(k)} \cdot Q^{(k)} = Q^{(k)H} \cdot \Theta^{(k)} \cdot Q^{(k)} \quad (29)$$

If $k \rightarrow \infty$, then the matrix $\Theta^{(k)}$ tends to a diagonal matrix, Λ , whose elements are the eigenvalues of the initial Θ :

$$\Theta = \underbrace{Q^{(n)H} \cdot \dots \cdot Q^{(k)H} \cdot \dots \cdot Q^{(0)H}}_{Q^H} \cdot \Lambda \cdot \underbrace{Q^{(0)} \cdot \dots \cdot Q^{(k)} \cdot \dots \cdot Q^{(n)}}_Q \quad (30)$$

The off-diagonal elements of matrix Θ (or the Hessenberg form A_{hess}) decrease in magnitude over FK iterations. As a result, there is no danger of overflow in any of those elements, and the wordlength needed to represent diagonal elements would suffice for representing the off-diagonal elements, too.

The unitary matrix Q of the eigenvalue decomposition of the Hermitian matrix Θ , and all unitary matrices $Q^{(k)}$ of the QR decompositions, are essentially rotation matrices. As a result, all their elements range between -1.0 and $+1.0$ (thus, there is no danger of encountering an overflow). In our fixed-point arithmetic implementation, a scale factor equal to 2^{qwl-1} is embedded into the matrix Q , so that its wordlength is $(qwl - 1)$ magnitude bits + 1 sign bit = qwl bits. The matrices Θ and Λ are already in fixed-point representation, their wordlengths being equal, $\theta wl = \lambda wl$. As a result, the eigenvalue decomposition can be written as:

$$\begin{aligned} \Theta &= Q \cdot \Lambda \cdot Q^H \Rightarrow \Theta = \frac{2^{qwl-1} Q}{2^{qwl-1}} \cdot \Lambda \cdot \frac{2^{qwl-1} Q^H}{2^{qwl-1}} \Rightarrow \\ &\Rightarrow 2^{2 \cdot (qwl-1)} \Theta = \underbrace{(2^{qwl-1} Q)}_{\underline{Q}} \cdot \Lambda \cdot \underbrace{(2^{qwl-1} Q^H)}_{\underline{Q}^H} \end{aligned} \quad (31)$$

where $\underline{Q}$ and $\underline{Q}^H$ is the unitary matrix and its Hermitian transpose in fixed-point representation (that is, with the scale factor

embedded). Similarly, $\underline{\Theta}^{(k)}$ is the fixed-point representation of $\Theta^{(k)}$, and an FK iteration in the fixed-point domain can be written as:

$$\left. \begin{aligned} \Theta^{(k)} &= \mathbf{Q}^{(k)} \cdot \mathbf{R}^{(k)} \Rightarrow \underline{\Theta}^{(k)} = \underline{\mathbf{Q}}^{(k)} \cdot \mathbf{R}^{(k)} \\ \underline{\Theta}^{(k+1)} &= \mathbf{R}^{(k)} \cdot \underline{\mathbf{Q}}^{(k)} \end{aligned} \right\} \Rightarrow \Rightarrow \underline{\Theta}^{(k+1)} = \underline{\Theta}^{(k+1)} \gg (qwl - 1) \quad (32)$$

It is mentioned that the last right shift operation may be performed with or without rounding.

The convergence of the Francis-Kublanovskaya algorithm can be significantly improved by introducing shifts in the diagonal values of matrices $\Theta^{(k)}$ [4]. There are three common shifts used in FK algorithms: (i) the Rayleigh quotient shift, which is equal to the last diagonal element; (ii) the Wilkinson shift, which is equal to the eigenvalue of the lower rightmost 2-by-2 sub-matrix of $\Theta^{(k)}$; and (iii) the Francis double shift, which applies a complex conjugate pair of shifts. In our simulations, the Rayleigh quotient shift has been used due to its implementation simplicity.

A challenge of designing a MMIMO base station is to minimize the wordlengths θwl and qwl , while the bit-error rate is maintained at acceptable levels. As mentioned in previous paragraphs, our simulation software using floating-point arithmetic used in prior work [9] has been converted to use fixed-point arithmetic. To estimate the transmitted vector $\mathbf{X}$, the linear system shown in Equation (1) is solved through an eigenvalue decomposition rather than the MATLAB® backslash operator. The wordlength of matrix $\mathbf{Q}$ was set to $qwl = 6$ and $qwl = 7$ bits, and the wordlength of matrix Θ was set to $\theta wl = 9$, $\theta wl = 10$, and $\theta wl = 11$. The number of quantization bits were set to $B = 1$, $B = 2$, and then $B = 3$. Extensive simulation runs were attempted, and BER performance figures were recorded.

In the system we have evaluated, each user independently transmits a single-carrier signal with a modulation of order 4 QPSK using 256-symbol blocks with a cyclic prefix of 16 samples. An error correction code consisting of a rate $\frac{1}{2}$ convolutional code applied over 32 single-carrier blocks is used. The simulated random multipath fading channel had a maximum delay of 9 symbol periods. The channel state is assumed to be stable over each set of 32 single-carrier blocks, where for each set the user transmits 2 pilots per block with content known a priori at the receiver which is used for initial channel estimation. A joint channel estimation/data detection/data decoding algorithm as described in [9] is used to provide the final result.

The BER figures versus the wordlength for implementing the eigenvalue decomposition in fixed-point arithmetic with reduced precision are presented in Table III. For reference, the BER figures corresponding to a double-precision floating-point implementation are also provided. It is apparent that for a coarse quantization with $B = 1$ bit, neither the floating-point nor the fixed-point implementations achieve a satisfactory BER in a small number of iterations. For a coarse quantization with $B = 2$ or $B = 3$, three to four iterations are sufficient to achieve a good BER. It is observed that an additional

iteration is generally needed in fixed-point implementation over the floating-point implementation to match their BERs; for example, $\text{BER}(B = 2, qwl = 6, \theta wl = 9, i = 3) = 0.0259$ (fixed-point), whereas $\text{BER}(B = 2, qwl = 64, \theta wl = 64, i = 2) = 0.0236$ (floating-point).

TABLE III
BER VERSUS WORDLENGTH FOR $R = 128$, $T = 16$, $E_b/N_0 = 5$ dB
(ONLY THE FIRST THREE ITERATIONS ARE SHOWN).

B	Wordlength		Iteration		
	qwl	θwl	$i = 1$	$i = 2$	$i = 3$
1	Double-Precision Floating-Point Arithmetic				
	64	64	0.4581	0.3235	0.2116
	Fixed-Point Arithmetic				
	6	9	0.4717	0.3990	0.2670
	6	10	0.4713	0.3443	0.2503
	6	11	0.4706	0.3461	0.2482
	7	9	0.4705	0.3930	0.2666
	7	10	0.4741	0.3442	0.2516
	7	11	0.4723	0.3520	0.2539
	Double-Precision Floating-Point Arithmetic				
2	64	64	0.2854	0.0236	0.0078
	Fixed-Point Arithmetic				
	6	9	0.3284	0.0823	0.0259
	6	10	0.3225	0.0395	0.0112
	6	11	0.3203	0.0311	0.0125
	7	9	0.3277	0.0794	0.0303
	7	10	0.3245	0.0399	0.0107
	7	11	0.3206	0.0315	0.0106
	Double-Precision Floating-Point Arithmetic				
	64	64	0.1849	0.0396	0.0066
3	Fixed-Point Arithmetic				
	6	9	0.2380	0.0379	0.0091
	6	10	0.2265	0.0389	0.0089
	6	11	0.2226	0.0453	0.0104
	7	9	0.2342	0.0354	0.0083
	7	10	0.2255	0.0371	0.0090
	7	11	0.2201	0.0436	0.0119

It is apparent that for a quantization with $B = 2$ bits, the matrix $\mathbf{Q}$'s wordlength, qwl , has little impact on BER. As an example, $\text{BER}(B = 2, qwl = 6, \theta wl = 9, i = 2) = 0.0823$, and $\text{BER}(B = 2, qwl = 7, \theta wl = 9, i = 2) = 0.0794$. On the other hand, the matrix Θ 's wordlength, θwl , has a much stronger impact on BER. For example, $\text{BER}(B = 2, qwl = 6, \theta wl = 10, i = 2) = 0.0395$, which a significant decrease from 0.0823. It should be observed that the BER does not improve significantly for $\theta wl \geq 11$. A MMIMO system with $B = 3$ is robust enough to operate with $\theta wl = 9$, but the cost of the implementation will be higher due to the increased complexity of the implementation in fixed-point arithmetic.

Based on these considerations, it is apparent that a good trade-off in implementing a MMIMO base station is a coarse quantization with $B = 2$ bits, a $\mathbf{Q}$'s wordlength $qwl = 7$ bits, and a Θ 's wordlength $\theta wl = 10$ bits. With such a reduced qwl , vector rotations can be implemented in inexpensive hardware, such as an array of reasonably sized FPGAs. A reduced θwl allows the extensive use of FPGA Digital-Signal Processing slices to perform the reverse products in the FK recursion, as well as complete the calculation of the solution of the linear system shown in Equation (1) through matrix multiplication, as outlined in Equation (4). Previous works are in line with these claims [16].

V. CONCLUSIONS

For 2-bit quantization ($B = 2$), an additional 0.85 dB of signal power is required to match the performance of ideal non-quantized measurements. The eigenvalue decomposition, the most computationally demanding portion of a MMIMO receiver algorithm, can be implemented in fixed-point arithmetic with wordlengths of 7 and 10 bits for eigenvectors and eigenvalues, respectively, at the expense of a small degradation of the Bit-Error Rate achieved in a double-precision floating-point implementation. An inexpensive set of reasonably sized FPGAs further reduce the cost of implementing massive MIMO base stations, which is a key impediment to their widespread deployment. In future work, the wordlength of the fixed-point implementation will be assessed in the presence of signal contamination from adjacent networks.

REFERENCES

- [1] C. Mollén, "High-End Performance with Low-End Hardware. Analysis of Massive MIMO Base Station Transceivers," Ph.D. dissertation, Linköping University, Linköping, Sweden, 2018.
- [2] C. Mollén, J. Choi, E. G. Larsson, and J. Robert W. Heath, "Uplink Performance of Wideband Massive MIMO with One-Bit ADCs," *IEEE Transactions on Wireless Communications*, vol. 16, no. 1, pp. 87–100, January 2017.
- [3] —, "Achievable Uplink Rates for Massive MIMO with Coarse Quantization," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*. New Orleans, Louisiana: IEEE, March 2017, pp. 6488–6492.
- [4] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, 1996.
- [5] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 2000.
- [6] M. Ercegovic and T. Lang, *Digital Arithmetic*, Morgan Kaufmann, 2003.
- [7] A. Chockalingam and B. S. Rajan, *Large MIMO Systems*. Cambridge University Press, 2014.
- [8] M. K. Ozdemir and H. Arslan, "Channel Estimation for Wireless OFDM systems," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 2, pp. 18–48, July 2007.
- [9] Z. Zhang, M. McGuire, and M. Sima, "Iterative Channel Estimation for Large Scale MIMO with Highly Quantized Measurements in 5G," in *The 28th European Signal Processing Conference (EUSIPCO 2020)*. Amsterdam, The Netherlands: European Association for Signal Processing (EURASIP), January 2021, pp. 1643–1647.
- [10] *IEEE Standard for Floating-Point Arithmetic. IEEE STD 754-2019*, IEEE Computer Society, July 2019.
- [11] J. M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, 2nd ed. Prentice-Hall, 1995.
- [12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006.
- [13] J. Francis, "The QR transformation – Part 1," *The Computer Journal*, vol. 4, no. 3, pp. 265–272, October 1961.
- [14] —, "The QR transformation – Part 2," *The Computer Journal*, vol. 4, no. 4, pp. 332–345, January 1962.
- [15] V. N. Kublanovskaya, "On some algorithms for the solution of the complete eigenvalue problem," *USSR Comput. Math. and Math. Physics*, no. 3, pp. 637–657, March 1961.
- [16] G. Xu, Y. Li, J. Yuan, R. Monroe, S. Rajagopal, S. Ramakrishna, Y. H. Nam, J.-Y. Seol, J. Kim, M. M. U. Gul, A. Aziz, and J. Zhang, "Full Dimension MIMO (FD-MIMO): Demonstrating Commercial Feasibility," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 8, pp. 1876–1886, August 2017.
- [17] A. Molisch, *Wireless Communications*, 2nd ed. West Sussex, United Kingdom: John Wiley & Sons, 2011.
- [18] A. Chockalingam and B. Sundar Rajan, *Large MIMO Systems*. Cambridge, United Kingdom: Cambridge University Press, 2014.
- [19] A. Zaidi, F. Athley, J. Medbo, U. Gustavsson, G. Durisi, and X. Chen, *5G Physical Layer: Principles, Models and Technology Components*. Academic Press, 2018.
- [20] M. Ozdemir and H. Arslan, "Channel estimation for wireless OFDM systems," *IEEE Communications Surveys Tutorials*, vol. 9, no. 2, pp. 18–48, 2007.
- [21] J. Robert F. Jones and J. Earl E. Swartzlander, "Parallel Counter Implementation," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 7, no. 3, pp. 223–232, October 1994.
- [22] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Efficient Synthesis of Compressor Trees on FPGAs," in *Proceedings of the 13th Asia and South Pacific Design Automation Conference (ASP-DAC 2008)*, Seoul, Korea, March 2008, pp. 138–143.
- [23] —, "Exploiting Fast Carry-Chains of FPGAs for Designing Compressor Trees," in *Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL'09)*, Prague, Czech Republic, August–September 2009, pp. 242–249.
- [24] T. Matsunaga, S. Kimura, and Y. Matsunaga, "Multi-Operand Adder Synthesis on FPGAs using Generalized Parallel Counters," in *Proceedings of the 15th Asia and South Pacific Design Automation Conference (ASP-DAC 2010)*, Taiwan, January 2010, pp. 337–342.
- [25] H. Parandeh-Afshar, A. Neogy, P. Brisk, and P. Ienne, "Compressor Tree Synthesis on Commercial High-Performance FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, no. 4, pp. 39:1–39:19, December 2011.
- [26] B. Khurshid and R. N. Mir, "High Efficiency Generalized Parallel Counters for Look-Up Table Based FPGAs," *International Journal of Reconfigurable Computing*, vol. 2015, September 2015.
- [27] J. M. Simkins and B. D. Philofsky, "Structures and Methods for Implementing Ternary Adders/Subtractors in Programmable Logic Devices," U.S. Patent 7,274,211 B1, September 2007.
- [28] Xilinx, Inc., "All Programmable 7 Series – Product Selection Guide," User Guide XMP101, v1.7, February 2018.
- [29] Xilinx, Inc., "7 Series FPGAs Memory Resources: User Guide," User Guide UG473, v1.13, February 2019.
- [30] Xilinx, Inc., "Vivado Design Suite – HLx Editions," ver. 2018.3, 2018.
- [31] M. Tian, M. Sima, and M. Michael McGuire, "Massive MIMO in Fixed-Point Arithmetic," in *Proceedings of the 23rd International Conference on Advanced Communications Technology (ICACT 2021)*, PyeongChang, Korea, February 2021, pp. 91–95.
- [32] Chen TC (1972) Automatic Computation of Exponentials, Logarithms, Ratios, and Square Roots. IBM Journal of Research and Development 16(4):380–388



Mi Tian was born in the People's Republic of China in 1988 and received her B.S. degree in optical communications engineering from Jilin University, Changchun, Jilin, China, in 2010, her M.S. degree in analogue and digital integrated circuit design from Imperial College London, U.K., in 2011, and her Ph.D. degree from the University of Victoria, British Columbia, Canada, in 2021. She is currently a postdoctoral fellow in the Department of Electrical and Computer Engineering at the same institution. Her research interests include wireless

communication systems, hardware implementation on FPGAs of massive MIMO communication systems.



Mihai Sima (S'00–M'04) was born in Romania in 1964 and received his B.Eng. degree in electronics engineering from Polytechnic Institute of Bucharest, Romania in 1989, and his Ph.D. degree in computer engineering from Delft University of Technology, The Netherlands, in 2004. Since 2003, he has been a faculty member in the Department of Electrical and Computer Engineering at the University of Victoria, British Columbia, Canada. His research interests include computer architecture and engineering, reconfigurable computing, embedded systems,

circuit design, and hardware security.



Michael McGuire (S'95–M'97) was born in Canada in 1970 and received his B.Eng. in computer engineering from the University of Victoria, British Columbia, Canada, in 1995, his M.A.Sc. in electrical engineering from the same institution in 1997, and his Ph.D. from the University of Toronto in 2003. Since 2003, he has been a faculty member in the Department of Electrical and Computer Engineering at the University of Victoria. Dr. McGuire's research area is in the application of signal processing to communications, developing new techniques for

Faster-than-Nyquist signaling, iterative channel estimation/data detection, and wireless radio localization suitable for advanced wireless systems.