



A 1.2mm² 416mW 1.44Mmat/s 64x16 Matrix Preprocessing ASIC for Massive MIMO in 22FDX

Conference Paper**Author(s):**

Nonaca, Darja ; Studer, Christoph 

Publication date:

2024-09-09

Permanent link:

<https://doi.org/10.3929/ethz-b-000699709>

Rights / license:

In Copyright - Non-Commercial Use Permitted

A 1.2 mm² 416 mW 1.44 M mat/s 64×16 Matrix Preprocessing ASIC for Massive MIMO in 22FDX

Darja Nonaca and Christoph Studer

Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland

Abstract—Massive multiuser (MU) multiple-input multiple-output (MIMO) enables concurrent transmission of multiple users to a multi-antenna basestation (BS). To detect the users' data using linear equalization, the BS must perform preprocessing, which requires, among other tasks, the inversion of a matrix whose dimension equals the number of user data streams. Explicit inversion of large matrices is notoriously difficult to implement due to high complexity, stringent data dependencies that lead to high latency, and high numerical precision requirements. We propose a novel preprocessing architecture based on the block-LDL matrix factorization, which improves parallelism and, hence, reduces latency. We demonstrate the effectiveness of our architecture through (i) massive MU-MIMO system simulations with mmWave channel vectors and (ii) measurements of a 22FDX ASIC, which is, to our knowledge, the first fabricated preprocessing engine for massive MU-MIMO with 64 BS antennas and 16 single-antenna users. Our ASIC reaches a clock frequency of 870 MHz while consuming 416 mW. At its peak throughput, the ASIC preprocesses 1.44 M 64 × 16 matrices per second at a latency of only 0.7 μs.

I. INTRODUCTION

Modern wireless communication systems leverage massive multiple-input multiple-output (MIMO) to enable multiuser (MU) communication at high data rates [1]. To enable efficient hardware implementation of data detection at the basestation (BS), one typically resorts to linear methods, such as linear minimum mean square error (LMMSE)-based equalization [2]. The complexity of such approaches, however, grows quickly for systems that must support a large number of simultaneously-transmitting users. In particular, the complexity of preprocessing, which computes the LMMSE filter matrix every time the channel changes, grows cubically in the number of users for all methods that have been implemented in hardware. Besides minimizing complexity, the preprocessing latency must be kept at a minimum to adhere to the stringent latency constraints of modern wireless systems. While approximate preprocessing methods that scale only quadratically in the number of users have been proposed [3], they only perform well (i) if the number of BS antennas is substantially larger than the number of users and (ii) the users' channels are sufficiently distinct. Therefore, efficient and also exact matrix preprocessing algorithms and hardware implementations are crucial to meeting the latency and quality constraints of massive MU-MIMO systems.

The work of DN and CS was supported in part by an ETH grant. Contact author: D. Nonaca (e-mail: dnonaca@iis.ee.ethz.ch).

The authors would like to thank GlobalFoundries for providing silicon fabrication through the 22FDX University Program. The authors also thank Seyed Hadi Mirfarshbafan for his assistance with the ASIC design flow and Gian Marti for comments on this paper.

A. Contributions

We propose the first fabricated ASIC of an exact matrix preprocessing engine for LMMSE-based data detection in massive MU-MIMO systems with 64 BS antennas and 16 single-antenna users. Our architecture carries out the following steps: (i) Gram-matrix computation, (ii) block-LDL (BLDL) matrix factorization, and (iii) backward substitution. To reduce complexity, we utilize the method from [4] to skip the otherwise necessary forward-substitution step. In contrast to other matrix-factorization methods, our BLDL-based architecture processes more data items in parallel, which reduces latency. To reduce silicon area, steps (i) and (ii) share the same hardware resources. A comparison with existing designs reveals that our fabricated and measured ASIC outperforms other designs in terms of throughput, area, latency, and/or error-rate performance.

B. Relevant Prior Work

A variety of algorithms for explicit matrix inversion exist, such as methods based on the Cholesky, LU, LDL, and QR matrix factorizations [5]. Several hardware architectures for preprocessing and LMMSE-based data detection in small-scale MIMO systems have been proposed: Reference [6] implements matrix inversion using rank-1 updates; references [7] and [8] perform an LU and LDL factorization, respectively, followed by forward and backward substitution; and references [9], [10] perform a QR factorization followed by inversion of the triangular matrix. For massive MU-MIMO systems, reference [11] provides synthesis results of a 128 × 16 preprocessing engine that uses the Cholesky decomposition followed by forward and backward substitution. References [12], [13] implement an approximate matrix inversion based on the Neumann series, which reduces complexity but sacrifices error-rate performance. In contrast, we propose an *exact* 64 × 16 BLDL-based matrix-preprocessing engine that avoids backward substitution, which reduces latency and complexity. Furthermore, we provide measurement results of a fabricated ASIC in 22FDX.

C. Notation

Boldface lowercase and uppercase letters represent column vectors and matrices, respectively. For a matrix \mathbf{G} partitioned into 2×2 blocks, $\mathbf{G}_{ij} \in \mathbb{C}^{2 \times 2}$ is the submatrix formed by the elements of \mathbf{G} consisting of the rows $(2(i-1)+1 : 2(i-1)+2)$ and columns $(2(j-1)+1 : 2(j-1)+2)$. The Hermitian transpose of \mathbf{G} is \mathbf{G}^H , and the entry on the m th row and

Algorithm 1 Block-LDL (BLDL) factorization [14]

input: $\mathbf{A} \in \mathbb{C}^{U \times U}$ partitioned into 2×2 blocks; $N = U/2$
for $j = 1$ to N **do**
 $\mathbf{D}_{jj} = \mathbf{A}_{jj} - \sum_{k=1}^{j-1} \mathbf{L}_{jk} \mathbf{D}_{kk} \mathbf{L}_{jk}^H$
for $i = j + 1$ to N **do**
 $\mathbf{L}_{ij} = (\mathbf{A}_{ij} - \sum_{k=1}^{j-1} \mathbf{L}_{ik} \mathbf{D}_{kk} \mathbf{L}_{jk}^H) \mathbf{D}_{jj}^{-1}$
end for
end for
output: $\mathbf{L}, \mathbf{D}^{-1}$

n th column is g_{mn} . Complex conjugation is indicated by the superscript $*$. The $N \times N$ identity matrix is \mathbf{I}_N .

II. SYSTEM MODEL AND BLDL-BASED PREPROCESSING

A. System Model and LMMSE-based Data Detection

We focus on the massive MU-MIMO uplink, in which U single-antenna UEs transmit data to a B -antenna BS. We model the frequency-flat input-output relation as $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$, where $\mathbf{y} \in \mathbb{C}^B$ is the received vector at the BS, $\mathbf{H} \in \mathbb{C}^{B \times U}$ is the channel matrix, $\mathbf{s} \in \mathbb{C}^U$ is the transmit symbol vector with entries taken from a constellation \mathcal{X} whose energy is normalized to E_s , and $\mathbf{n} \in \mathbb{C}^B$ is i.i.d. circularly-symmetric complex Gaussian noise with variance N_0 per entry.

Data detection deals with recovering the transmit vector \mathbf{s} from \mathbf{y} and (an estimate of) \mathbf{H} . LMMSE-based methods perform data detection in two phases: (i) *preprocessing* first calculates and then inverts the matrix

$$\mathbf{A} = \mathbf{H}^H \mathbf{H} + \frac{N_0}{E_s} \mathbf{I}_U, \quad (1)$$

whenever the channel matrix \mathbf{H} changes; and (ii) *equalization* is carried out for every transmit vector \mathbf{s} according to $\hat{\mathbf{s}}_{\text{LMMSE}} = \mathbf{A}^{-1} \mathbf{H}^H \mathbf{y}$. In what follows, we focus on the preprocessing phase as it dominates complexity and latency.

B. BLDL-based Matrix Preprocessing

After computing \mathbf{A} as in (1), we factorize $\mathbf{A} = \mathbf{L} \mathbf{D} \mathbf{L}^H$, where \mathbf{L} is a lower-triangular with 2×2 identity matrices on the diagonal and \mathbf{D} is a block diagonal matrix also consisting of 2×2 blocks. To improve parallelism, we utilize the BLDL factorization from [14], which is summarized in Alg. 1. In our architecture, we partition \mathbf{A} into 2×2 submatrices and the 2×2 submatrix inversions \mathbf{D}_{jj}^{-1} , $j = 1, \dots, \frac{U}{2}$ in Alg. 1 are calculated efficiently via direct inversion [5]

$$\mathbf{D}_{jj}^{-1} = \begin{bmatrix} a & b \\ b^* & d \end{bmatrix}^{-1} = \frac{1}{\Delta} \begin{bmatrix} d & -b \\ -b^* & a \end{bmatrix}, \quad (2)$$

where $\Delta = ad - bb^*$ is the determinant of \mathbf{D}_{jj} . After the BLDL factorization, we can rewrite \mathbf{A}^{-1} as

$$\mathbf{A}^{-1} = (\mathbf{L} \mathbf{D} \mathbf{L}^H)^{-1} = (\mathbf{L}^H)^{-1} \mathbf{D}^{-1} \mathbf{L}^{-1}. \quad (3)$$

By multiplying (3) from the left by \mathbf{L}^H , we now solve

$$\mathbf{L}^H \mathbf{X} = \mathbf{D}^{-1} \mathbf{L}^{-1} \quad (4)$$

for \mathbf{X} , where the solution $\hat{\mathbf{X}} = \mathbf{A}^{-1}$ will be the desired inverse. To reduce complexity, we follow the idea of [4] to solve for \mathbf{X}

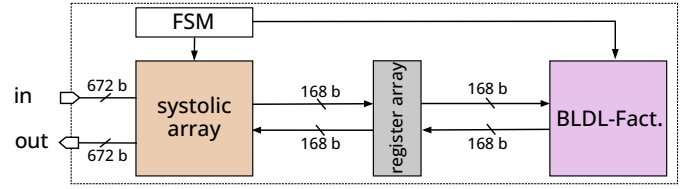


Fig. 1. Top-level architecture of the implemented preprocessing engine. The bus width of the input and output data accommodates the size of a row of the channel matrix \mathbf{H} (16 complex values of 21 bits per part). The bus at the interface with the register array fits a 2×2 matrix with 4 complex values.

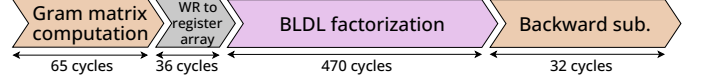


Fig. 2. Schedule (in clock cycles) of each preprocessing step.

without computing the inverse \mathbf{L}^{-1} . To illustrate this idea, consider the following simplified 3×3 example of (4)

$$\begin{bmatrix} 1 & l_{21}^* & l_{31}^* \\ 0 & 1 & l_{32}^* \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} d_1^{-1} & 0 & 0 \\ \alpha d_2^{-1} & d_2^{-1} & 0 \\ \beta d_3^{-1} & \gamma d_3^{-1} & d_3^{-1} \end{bmatrix}, \quad (5)$$

where (i) we use the fact that the inverse of a lower-triangular matrix is lower triangular and (ii) α , β , and γ are the off-diagonal elements of \mathbf{L}^{-1} . The process starts by solving for the third column of \mathbf{X} from bottom to top using backward substitution. This is equivalent to solving the following equations one after the other:

$$\begin{aligned} x_{33} &= d_3^{-1} \\ x_{23} + l_{32}^* x_{33} &= 0 \\ x_{13} + l_{21}^* x_{23} + l_{31}^* x_{33} &= 0. \end{aligned} \quad (6)$$

Once x_{33} , x_{23} , and x_{13} have been computed, one proceeds analogously by computing the second column of \mathbf{X} by solving

$$\begin{aligned} x_{22} + l_{32}^* x_{23} &= d_2^{-1} \\ x_{12} + l_{21}^* x_{22} + l_{31}^* x_{23} &= 0. \end{aligned} \quad (7)$$

Finally, one proceeds with the first column to solve for x_{11} :

$$x_{11} + l_{21}^* x_{12} + l_{31}^* x_{13} = d_1^{-1}. \quad (8)$$

We reiterate that this process for solving for $\mathbf{X} = \mathbf{A}^{-1}$ avoids inverting \mathbf{L} as the off-diagonal elements α , β , and γ are unused.

III. VLSI ARCHITECTURE

A. Architecture Overview

Fig. 1 depicts the top-level architecture of our preprocessing engine for LMMSE-based data detection that implements the procedure detailed in Sec. II-B. Specifically, our architecture performs: (i) Gram-matrix computation as in (1) using a systolic array followed by buffering the result in a flip-flop-based register array; (ii) matrix factorization as in Alg. 1 using a specialized BLDL factorization engine; and (ii) backward substitution as in Sec. II-B by reusing the systolic array. The complete schedule (in clock cycles) for a 64×16 channel matrix \mathbf{H} is depicted in Fig. 2. Since the matrix factorization step (ii) dominates the preprocessing latency, we utilize a BLDL-based approach, which enables higher parallelism than a Cholesky-, LDL-, LU-, or QR-based matrix-inversion approach.

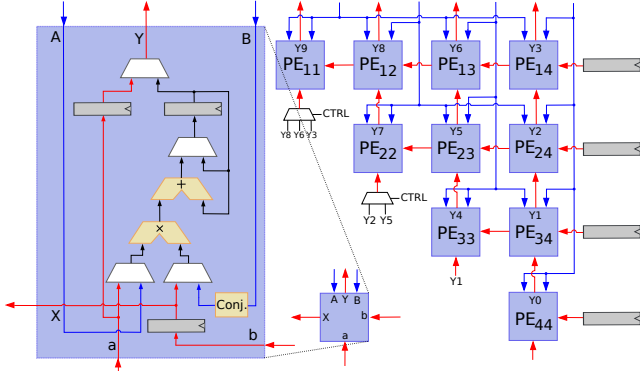


Fig. 3. Architecture details of the systolic array that supports two modes: Gram-matrix computation (blue datapath) and backward substitution (red datapath). We illustrate an architecture for $U = 4$ users.

B. Systolic Array

As illustrated in Fig. 3, we use a systolic array that supports two different modes. In the first mode, the matrix \mathbf{A} from (1) is computed. The systolic array consists of an upper-triangular array of processing elements (PEs), each containing a complex-valued multiplier (built from four real-valued multipliers) and an accumulator. Since \mathbf{A} is Hermitian, we only need to compute the upper-triangular part. Since our ASIC (see Sec. IV) is designed for a channel matrix \mathbf{H} of dimension $B = 64$ times $U = 16$, the systolic array consists of $(U^2 + U)/2$ PEs. In every clock cycle, the systolic array is fed with the i th row of \mathbf{H} , where each PE (m, n) (denoted by PE_{nm}) sequentially computes the entry $g_{mn} = \sum_{j=1}^B h_{mj} h_{nj}^*$ of $\mathbf{G} = \mathbf{H}^H \mathbf{H}$ in B clock cycles. Thus, our ASIC takes $B = 64$ clock cycles to calculate \mathbf{G} . One additional clock cycle is used to add the regularization term $\frac{N_0}{E_s}$ to the diagonal of \mathbf{G} to arrive at \mathbf{A} .

Subsequently, the entries of \mathbf{A} are buffered in the register array from (and to) which the BLDL factorization engine (see Sec. III-C) can read (and write). The register array has $(\frac{U^2}{2} + U)/4$ entries for the 2×2 submatrices of \mathbf{A} and extra $U/2$ entries to store the submatrices \mathbf{D}_{jj}^{-1} , $j = 1, \dots, \frac{U}{2}$.

In the second mode, the systolic array computes \mathbf{A}^{-1} using backward substitution. In the first clock cycle, the block diagonal PEs are loaded with \mathbf{D}_{jj}^{-1} , $j = 1, \dots, \frac{U}{2}$, submatrices computed during BLDL factorization. In the second clock cycle, the PE at the bottom passes the result to the PEs above, which multiply the received value with the appropriate $-l_{ij}^*$ value, accumulate the result in the internal register, and pass it to the PEs above and so on. For the U th column of the systolic array, this procedure is equivalent to solving the set of equations in (6). In the fourth clock cycle, the PE at the bottom of the $(U - 1)$ th column can start the same procedure described above, which corresponds to solving the set of equations in (7). The computations continue analogously for all the columns of the systolic array until the first column is reached. The backward-substitution step takes a total of $2U$ clock cycles.

C. Block-LDL Factorization Engine

Our BLDL-factorization engine implements a processor-like architecture, which is illustrated in Fig. 4. Specialized instructions, along with data fetch and result write addresses, are

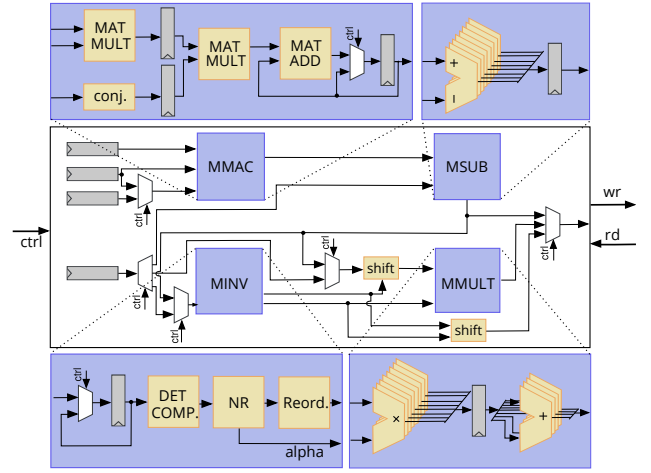


Fig. 4. BLDL-factorization engine. The architecture is composed of four arithmetic units: MMAC, MSUB, MINV and MMULT. The units operate on 2×2 matrices and are controlled by an FSM in a processor-like fashion.

encoded and stored in a look-up table (LUT). The instructions not bound to any data dependency are parallelized to minimize latency. In each clock cycle, one row of the LUT is read, triggering the state transitions of a finite-state machine (FSM), which provides the control signals to four main arithmetic units, each processing 2×2 submatrices: Matrix multiply-accumulate (MMAC), matrix subtraction unit (MSUB), a matrix inversion unit (MINV) composed of a complex-valued scalar inversion unit based on the Newton-Raphson iteration as in [7], and matrix-matrix multiplication (MMULT). The latencies of the MMAC, MSUB, MINV, and MMULT units are two, one, four, and one clock cycle(s), respectively.

D. Numerical Precision

To optimize efficiency, we exclusively utilize fixed-point arithmetic. Before feeding the rows of the channel matrix \mathbf{H} to the preprocessing engine, we assume that the entries in each row are normalized by the maximum absolute value in that row. This enables the use of 42 bit to represent a complex number (21 bit per part), respectively. To demonstrate the accuracy of our fixed-point design for a 64×16 massive MU-MIMO system with 16-QAM, we simulate the uncoded bit-error rate (BER). We use the QuaDRiGa mmMAGIC UMi [15] channel model (LoS and non-LoS) with 1° minimum user separation and perfect power control, and we perform least-squares channel estimation followed by LMMSE-based data detection.

Fig. 5 compares the uncoded BER between a floating-point reference and our fixed-point golden model. As another baseline, we also show the performance of the approximate inversion method from [12]. We also compare an alternative architecture in which Δ in (2) is forced to be real-valued (indicated by RD). We observe that the BER of our preprocessing engine follows closely that of the floating-point reference to an uncoded BER of about 10^{-3} under non-LoS conditions.

IV. IMPLEMENTATION RESULTS AND COMPARISON

Fig. 6(a) depicts the fabricated 5 mm^2 chip in GlobalFoundries' 22 FDXTM FD-SOI technology. The BLDL factoriza-

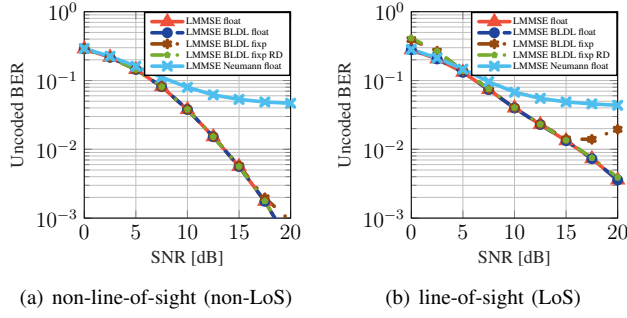


Fig. 5. Uncoded bit-error rate (BER) vs. signal-to-noise ratio (SNR) of LMMSE-based equalization for (a) non-LoS and (b) LoS mmWave channels.

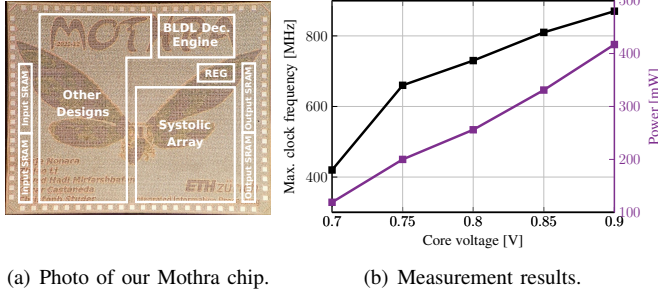


Fig. 6. (a) Die photo of our $2.5 \times 2 \text{ mm}^2$ Mothra chip containing the proposed preprocessing engine (right) along with other designs (left). (b) Clock frequency and power measurements of the preprocessing engine.

tion engine occupies 0.2 mm^2 , the systolic array 1 mm^2 , and the register file 0.04 mm^2 . The ASIC also includes input and output SRAMs to perform high-speed measurements. At nominal 0.9 V core supply and 25°C , the ASIC achieves a maximum clock frequency of 870 MHz with the critical path in the PE of the systolic array. To measure power, we create mmWave channel-matrix stimuli to loop the preprocessing engine for 7 ms . We do the same for accessing the input memory in order to isolate the dynamic power of the preprocessing engine by subtracting the power of such accesses from the total power. At the maximum clock frequency, the preprocessing engine consumes 416 mW . At 0.7 V core supply, our design achieves 420 MHz , which results in a power consumption of 120 mW and a throughput of 1.44 M mat/s and $0.7 \mu\text{s}$ latency; please refer to Fig. 6(b) for more details. As it can be seen from Tbl. I, our ASIC achieves significantly higher throughput and lower latency than the FPGA designs in [16], [17]. Our exact matrix inversion attains a comparable latency and throughput as [12], which achieves poor BER performance (cf. Fig. 5). It is challenging to compare our design to [11] as (i) they only provide synthesis results and (ii) the preprocessing latency was not reported. Furthermore, their reported cell area appears to be unusually compact, especially when considering that matrices of dimension 128×16 are processed.

V. CONCLUSIONS

We have proposed the first fabricated and measured preprocessing ASIC for LMMSE-based data detection in a 64 BS antenna, 16 user mmWave massive MU-MIMO system.

TABLE I
IMPLEMENTATION RESULTS AND COMPARISON WITH OTHER DESIGNS

	This work	Mahdavi [11]	Abbas [12]	Kumar [16]	Han [17]
Algorithm	BLDL	Cholesky	Neumann	QR	LDL
H dimension	64×16	128×16	80×16	25×25	32×32
Precision [bit]	21	20	16	32	—
Fabricated?	yes	no	no	no	no
Technology	22 nm	28 nm	65 nm	FPGA	FPGA
Core supply [V]	0.9	—	—	—	—
Active area [mm^2]	1.204	—	—	—	—
Cell area [kGE]	6.030	537	117	—	—
Max. clock freq. [MHz]	870	510	460	103.8	275
Max. throughput [M mat/s]	1.44	—	0.54	$5.6 \cdot 10^{-4}$	$3.2 \cdot 10^{-3}$
Latency [$\mu\text{s}/\text{mat}$]	0.7	—	1.85	1785	313.5
Power [mW]	416	210	—	1508	—
Norm. throughput ^a [M mat/s]	1.44	—	1.59	—	—
Norm. latency ^b [$\mu\text{s}/\text{mat}$]	0.69	—	0.63	—	—

^a Scaling by S and by S^{-1} where S is the relative dimension to 22 nm .

Unlike existing matrix-factorization approaches, our BLDL-based design improves parallel processing, thereby reducing preprocessing latency. Our ASIC achieves a throughput of 1.44 M mat/s and 416 mW at 870 MHz clock frequency at a latency of only $0.7 \mu\text{s}$. When compared to existing preprocessing engines, our implementation outperforms other designs in terms of throughput, area, latency, and/or error-rate performance.

REFERENCES

- [1] S. A. Busari *et al.*, “Millimeter-wave massive MIMO communication for future wireless systems: A survey,” *IEEE Comm. Surveys & Tutorials*, 2018.
- [2] M. A. Albreem *et al.*, “Massive MIMO detection techniques: A survey,” *IEEE Comm. Surveys & Tutorials*, 2019.
- [3] M. Wu *et al.*, “Implicit vs. explicit approximate matrix inversion for wideband massive MU-MIMO data detection,” in *IEEE JSPS*, 2018.
- [4] R. Clasen, *Numerical methods for inverting positive definite matrices*. RAND Corporate Tech. Rep., 1966.
- [5] G. H. Golub *et al.*, *Matrix Computations*, 3rd ed. The Johns Hopkins Univ. Press, 1996.
- [6] A. Burg *et al.*, “Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems,” in *IEEE ISCAS*, 2006.
- [7] C. Studer *et al.*, “ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation,” *IEEE JSSC*, 2011.
- [8] D. Auras *et al.*, “Efficient VLSI architectures for matrix inversion in soft-input soft-output MMSE MIMO detectors,” in *IEEE ISCAS*, 2014.
- [9] S. S. Omran *et al.*, “Fast QR decomposition based on FPGA,” in *IEEE ICOASE*, 2018.
- [10] C. K. Singh *et al.*, “VLSI architecture for matrix inversion using modified Gram-Schmidt based QR decomposition,” in *VLSID*, 2007.
- [11] M. Mahdavi *et al.*, “A VLSI implementation of angular-domain massive MIMO detection,” in *IEEE ISCAS*, 2019.
- [12] S. Abbas *et al.*, “Low-latency approximate matrix inversion for high-throughput linear pre-coders in massive MIMO,” in *IFIP/IEEE VLSI-SoC*, 2016.
- [13] H. Prabhu *et al.*, “Hardware efficient approximative matrix inversion for linear pre-coding in massive MIMO,” in *IEEE ISCAS*, 2014.
- [14] I. Stanimirovic, “Full-rank block LDL^* decomposition and the inverses of $n \times n$ block matrices,” in *JAMC*, 2012.
- [15] S. Jaekel *et al.*, “QuaDRiGa - quasi deterministic radio channel generator user manual and documentation,” Fraunhofer Heinrich Hertz Institute, Tech. Rep. v2.0.0, Aug. 2017.
- [16] K. V. S. Kumar *et al.*, “System on chip implementation of floating point matrix inversion using modified Gram-Schmidt based QR decomposition on PYNQ FPGA,” in *IEEE iSES*, 2021.
- [17] K. Han *et al.*, “Low-latency FPGA design and implementation of Hermitian matrix inversion based on partitioned systolic array for massive MIMO,” in *IEEE ICTA*, 2022.