Efficient MIMO Preprocessor With Sorting-Relaxed QR Decomposition and Modified Greedy LLL Algorithm

LIRUI CHEN\textsuperscript{1}, ZUOCHENG XING\textsuperscript{1}, YONGZHONG LI\textsuperscript{2}, AND SHIKAI QIU\textsuperscript{1}

\textsuperscript{1}National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410003, China
\textsuperscript{2}College of Computer Engineering and Applied Mathematics, Changsha University, Changsha 410003, China

Corresponding author: Lirui Chen (chenlirui14@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61874140.

ABSTRACT This paper proposes a high-efficient preprocessing algorithm for $16 \times 16$ MIMO detections. The proposed algorithm combines a sorting-relaxed QR decomposition (SRQRD) and a modified greedy LLL (MGLLL) algorithm. First, SRQRD is conducted to decompose the channel matrices. This decomposition adopts a relaxed sorting strategy together with a paralleled Givens Rotation (GR) array scheme, which can reduce the processing latency by 60\% compared with conventional sorted QR decomposition (SQRD). Then, an MGLLL algorithm is conducted to improve detection performance further. The MGLLL algorithm adopts a paralleled selection criterion, and only process the most urgent iterations. Thus the processing latency and column swaps can be reduced by 50\% and 75\%, respectively, compared with the standard LLL algorithm. Finally, the bit-error-rate (BER) performance of this preprocessing algorithm is evaluated using two MIMO detectors. Results indicate that this preprocessor suffers a negligible performance degradation compared with the combination of the standard LLL algorithm and SQRD. Based on this preprocessing algorithm, a pipelined hardware architecture is also designed in this paper. A series of systolic coordinated-rotation-digital-computer (CORDIC) arrays are utilized, and highly-pipelined circuits are designed, helping this architecture achieve high frequency performance. This architecture is implemented using 65-nm CMOS technology, which can work at a maximum frequency of 625 MHz to process channel matrices every 16 clock cycles. The latency is 0.9 us. Comparisons indicate that this preprocessor outperforms other similar designs in terms of latency, throughput, and gate-efficiency.

INDEX TERMS MIMO, SQRD, lattice reduction, LLL, greedy LLL, VLSI, CORDIC.

I. INTRODUCTION

Multiple-input-multiple-output (MIMO) technique has been extensively utilized in wireless communications to increase spectrum efficiency [1]. In MIMO systems, signal detection remains to be a challenging task, especially for larger-scaled MIMO systems [2]. Sole MIMO detectors can not meet the application requirements, because the optimal maximum likelihood (ML) detector [3] is deemed impossible for hardware implementation, while other sub-optimal detectors, such as the minimum-mean-square-error (MMSE) detector [4] and K-best detector [5], suffer from nonnegligible diversity degradation, especially when the number of user antennas is comparable to that of base station antennas. Therefore, the preprocessing technique is typically utilized to help the sub-optimal detectors achieve near-ML performance. This technique can also decrease the detection complexity within the same performance constraint [6]. Therefore, the preprocessing technique has played a predominant role for MIMO detectors in terms of the accuracy, latency, and throughput, especially when the channel is not slowly varying. Among the existing preprocessors [6], [7], the combination of SQRD and lattice reduction (LR) is regarded as one of the most significant preprocessing schemes, which is also adopted by this work.

In MIMO systems, the QRD [8] is utilized to decompose a channel matrix $H$ into a unitary matrix $Q$ and an upper-triangular matrix $R$. Based on QRD, the SQRD [9] incorporates a sorting process into the QRD steps to generate matrices $R$ with larger diagonal elements $r_{i,i}$ in the rear. As a larger $r_{i,i}$ leads to a better immunity to the interference, SQRD can effectively reduce the error propagation...
for the interference-elimination-based MIMO detectors such as the spherical detector (SD), K-best detector, etc. [10]. When combined with the LR technique, SQRD can enhance the convergence rate of LR iterations, thus reduce the LR complexity compared with sole QRD [11]. The majority of existing QRD/SQRD algorithms are based on four basic methods: the Householder transformation (HT) method, Givens rotation (GR) method, Gram-Schmidt (GS) method, and Cholesky method. HT method [12], [13] has a good adaptability even for decomposing correlated matrices, but relatively high complexity makes HT method unfavorable for hardware implementation. GS method [14], [15] is notable for its low latency because it decomposes matrices at column-wise. However, as GS method entails plenty of multipliers and other complicated operations such as square root and division, its hardware overhead is relatively higher. GR algorithm [16], [17] has an excellent numerical stability, thus shorter resources can meet the precision requirements for hardware implementation. Moreover, GR method is easy to achieve high frequency performance and simultaneously low area overhead with the help of CORDIC algorithm. However, the GR method suffers from long latency because it nullifies the matrix entries one by one. Another drawback of GR method is that the norm value of each column is not explicitly presented during the QRD process, therefore, additional resource would be required to calculate the norm value if the SQRD is conducted. Cholesky method [7] gathers the two advantages of both low latency and relatively low complexity. But this method only calculates the matrix $R$, thus additional resource would be required to calculate the matrix $Q$ if necessary. As this calculation entails a matrix multiplication of $H^T$ and $R^{-1}$, the hardware overhead is not ignorable. In MIMO systems, where low-area design is imperative, the GR method is believed to have a bright application prospect. Hence, this paper is devoted to addressing the long-latency problem of the GR-based SQRD while maintaining its low complexity.

The LR technique aims at finding a more orthogonal base for the same lattice of channel matrix $R$. As better orthogonality leads to higher detection accuracy, LR technique can effectively bridge the performance gap between the optimal MIMO detector and sub-optimal detectors. Several types of LR algorithms have been proposed by researchers [18]–[21], among which the LLL algorithm has attracted considerable attention due to its near-optimal diversity gain and polynomial complexity. Conventional implementations [22], [23] about the LLL algorithm consider the worst case that each iteration of the LLL algorithm would perform the column swap procedure, which leads to low hardware utilization because actually no calculation is required for some iterations according to the condition-check results. To alleviate this problem, the greedy LLL algorithm is proposed in [24], [25], which only performs the LLL iterations with column swaps. Thus the problem is how to select the most essential iteration each time so as to maximize the convergence rate. The existing greedy LLL algorithms typically adopt two selection criteria: selecting the iteration that maximizes the degradation of LLL potential (defined below), e.g., [26], [27]; or selecting the backmost iteration that violates the condition check inequation, e.g., algorithm 2 of [25]. Unfortunately, these two criteria are adopted alternatively, yet no literature absorbs the benefits from them both. Moreover, existing greedy LLL algorithms are only proposed theoretically, without sufficient consideration for hardware implementation. For example, only one iteration is selected at one time, leading to long latency. Furthermore, the variable iteration time also brings some difficulty for hardware design.

This paper proposes an efficient preprocessing algorithm, together with the hardware architecture, for 16 × 16 MIMO systems. The proposed processor consists of an SQRD component and an MGLLL component. In the SQRD component, a relaxed sorting strategy is adopted, which selects four columns with the minimum norm values at one time and swap them to the front of the matrix. Thus the GR process for these four columns can be performed in parallel. Compared with the conventional GR-based SQRD algorithm, this strategy can reduce the latency by 60%. In addition, the $\ell^2$-norm is replaced by $\ell^1$-norm during the sorting procedure to reduce the hardware overhead. In the MGLLL component, a paralleled constant-throughput scheme is adopted. Moreover, a novel selection criterion is proposed for selecting the most urgent iterations. This criterion comprehensively absorbs the benefits from the two conventional criteria. In addition, two iterations can be concurrently selected at one time. Thereby, the convergence is notably enhanced, and merely 6 stages are sufficient to realize near-LLL performance. Compared with the non-greedy LLL algorithms, this algorithm only processes two iterations out from the 8 iterations at each stage, so the column swaps are reduced by 75%. Performance simulation indicates that this preprocessor suffers from negligible performance degradations compared with the combination of standard LLL algorithm and SQRD. Based on this preprocessing algorithm, the corresponding hardware architecture is also proposed in this paper. Highly-pipelined CORDIC scheme is employed, and hardware-reuse is adopted, helping this preprocessor achieve appropriate trade-off among the throughput, area and latency performance. This architecture is implemented using a 65-nm CMOS technology, which can work at a maximum frequency of 625 MHz. The matrices are processed every 16 clock cycles, and the latency is 0.9 us. The comparison indicates that this preprocessor is superior to other similar works in terms of latency, throughput, and gate-efficiency performance.

The rest of this paper is organized as follows: Section II briefly introduces the system model and related works. Section III specifies the proposed preprocessing algorithm and the performance evaluation using SIC and K-best detectors. The hardware architecture is demonstrated in Section IV. Section V presents the implementation results and the comparisons with the state-of-art designs. Finally, Section VI draws the conclusion.
II. SYSTEM MODEL AND RELATIVE WORKS

A. SYSTEM MODEL

For an uplink MIMO system with $N_r$ receiving antennas on the base station (BS) side and $N_t$ transmission antennas on the user side, the system model can be presented as

$$ y = Hs + n $$

(1)

where $s \in \mathbb{C}^{N_t \times 1}$ represents the transmitted symbol vectors, $\Omega$ is the constellation set from a specific modulation, $H \in \mathbb{C}^{N_r \times N_t}$ denotes the Rayleigh flat-fading channel matrix. All the column vectors $h_i \ (i = 1, 2, \ldots, N_t)$ are complex-valued random vectors drawn from independent and identical Gaussian distribution with zero mean and unitary variance. $n \in \mathbb{C}^{N_r \times 1}$ is a white Gaussian noise vector with zero mean and variance $\sigma^2$. $y \in \mathbb{C}^{N_r \times 1}$ denotes the received signal vector. In this paper, we only consider the case where the BS is equipped with an equal number of antennas to the users, namely, $N_r = N_t = N = 16$. In addition, QAM-64 is adopted as the modulation, and the channel matrix $H$ is assumed to have been properly estimated.

B. SORTED QR DECOMPOSITION

SQRD is typically employed to decompose the channel matrix $H$ as

$$ H = QRP^{-1} $$

(2)

where $Q$ is a unitary matrix, $R$ is an upper-triangular matrix, and $P$ is a column permutation matrix. Thus the model of (1) can be reformed as

$$ Q^H y = RP^{-1} s + Q^H n $$

(3)

Let $\tilde{y} = Q^H y$, $\tilde{s} = P^{-1} s$, and $\tilde{n} = Q^H n$, the SQRD procedure transforms the MIMO model as

$$ \tilde{y} = R \tilde{s} + \tilde{n} $$

(4)

Numerous methods can be applied to perform the decomposition of (2), of which the GR method is known for its low complexity and relatively higher stability. As presented in (5), the GR method is performed by applying a series of rotation matrix $G_{i,j}$ to the channel matrix $H$.

$$ (G_{N,N}) \ldots (G_{N,2} \ldots G_{3,2})(G_{N,1} \ldots G_{2,1})HP_1 \ldots P_{N-1} = R $$

(5)

where each $G_{i,j}$ nullifies one element ($h_{i,j}$) in the lower-triangular part of $H$. After the nullifying process for the $i$th column, a permutation matrix $P_i$ is utilized to swap the column with the minimum norm to the front for the subsequent nullifying process. Thereby, the matrices $Q$ and $P$ can be generated as

$$ Q^H = (G_{N,N}) \ldots (G_{N,2} \ldots G_{3,2})(G_{N,1} \ldots G_{2,1}) $$

$$ P = P_1 \ldots P_{N-1} $$

(6)

As the matrix $Q$ is utilized in the form of $Q^H y$, the calculation of $Q$ can be substituted by performing the same rotation on signal $y$ as that on matrix $H$. The elementary rotation in (5) can be performed by directly calculating the rotation matrix $G_{i,j}$, or alternatively, by the CORDIC algorithm. In the first method, matrix $G_{i,j}$ is constructed based on the function

$$ \begin{bmatrix} c \quad s \\ -s \quad c \end{bmatrix} \begin{bmatrix} h_{i,j} \\ h_{i,j+1:N} \end{bmatrix} = \begin{bmatrix} h_{i,j}^{\text{new}} \\ h_{i,j+1:N}^{\text{new}} \end{bmatrix} $$

(7)

where

$$ \begin{align*}
    c &= \frac{h_{i,j}}{\sqrt{h_{i,j}^2 + h_{i,j+1:N}^2}}, \\
    s &= \frac{h_{i,j+1:N}}{\sqrt{h_{i,j}^2 + h_{i,j+1:N}^2}}
\end{align*} $$

(8)

This method entails complicated matrix multiplication and square root operations, so the CORDIC algorithm is typically employed to perform the rotation of (7). To nullify a complex-valued entry $h_{i,j}$, two CORDIC steps are generally required. In the first step, two CORDIC operations are performed on $h_{i,j}$ and $h_{i,j+1}$ respectively to zero the corresponding imaginary parts. During the second step, one CORDIC is performed on the real parts of $h_{i,j}$ and $h_{i,j+1}$ to nullify $h_{i,j}$, while another CORDIC is utilized to update the imaginary parts of latter elements.

C. LR TECHNIQUE AND GREEDY LLL ALGORITHM

After SQRD, the LR technique is typically employed to transform the matrix $R$ as

$$ R = Q_L R_L T^{-1} $$

(9)

where $Q_L$ is a unitary matrix, $T$ is a unimodular matrix with integer entries and $\det(T) = 1$. Let $\tilde{y}_L = Q_L^H \tilde{y}$, $x = T^{-1} \tilde{s}$, and $n_L = Q_L^H \tilde{n}$, the LR technique transforms the function (4) as

$$ \tilde{y}_L = R_L x + n_L $$

(10)

As $R_L$ is better conditioned than $R$, the LR technique can remarkably improve the MIMO detection performance. Several LR algorithms have been designed by researchers, among which the LLL algorithm is notable for its near-optimal diversity gain with polynomial complexity. The LLL algorithm modifies the matrix $R$ to satisfy the size-reduction condition (11) and Lovász condition (12) inequations. In each iteration of the LLL algorithm, the size reduction is first performed, followed by a condition check procedure. If the Lovász condition is violated, the corresponding two columns will be swapped, and a givens rotation will be utilized to maintains the upper-triangular property.

$$ \frac{1}{2} |r_{i-1,i-1}| \geq |r_{i-1,j}|, \quad 2 \leq i \leq j \leq N $$

(11)

$$ \delta |r_{i-1,i-1}|^2 \leq |r_{i,i}|^2 + |r_{i-1,i-1}|^2, \quad 2 \leq i \leq N $$

(12)

However, the implementation of the LLL algorithm suffers from low hardware efficiency because the column swap may not happen in some LLL iterations. To address it, the greedy LLL algorithm is proposed by researchers [24]–[27], which only performs the iterations with column swaps. Algorithm 1 presents the basic scheme of existing greedy LLL algorithms.
Algorithm 1 Conventional Greedy LLL Algorithm

1: Input: $\mathbf{y}$, $R$, $P$, $\delta$; Output: $\mathbf{y}_L$, $R_L$, $T$
2: Initial: $R = [R|\mathbf{y}]$; $T = P$; stage = 0;
3: while {condition violation exists & stage < MaxStage} do
4: stage = stage + 1;
5: // size reduction or effective size reduction:
6: for $n=2:N$ do
7: for $(i=1:n-1)$ or $(i=n-1)$ do
8: $\mu = |\overline{R}_i|/|\overline{R}_i|$
9: $R_{i:N,n} = R_{i:N,n} - \mu R_{i:N,i}$;
10: $T_{i:N,n} = T_{i:N,n} - \mu T_{i:N,i}$;
11: end for
12: end for
13: select the $k$th iteration;
14: // LLL reduction:
15: swap columns $k-1$ and $k$ in $R$ and $T$;
16: update $R$ by Givens Rotation;
17: end while
18: $[R_L|\mathbf{y}_L] = \mathbf{R}$; $T_L = T$

In Algorithm 1, the size reduction is first performed. Then, the most urgent $k$th iteration is selected according to a particular criterion. Through this selection, the $k$th iteration is more inclined to perform the column swap operation. Thereby, the hardware utilization is improved, and the convergence rate is also enhanced.

Existing greedy LLL algorithms focus mainly on how to select the most urgent iteration on line 13 of Algorithm 1. Two criteria are commonly adopted by researchers, of which the first one aims at maximizing the degradation of the LLL potential $D$, which is defined as [25], [28]

$$D = \prod_{n=1}^{N-1} d_n = \prod_{n=1}^{N-1} |\overline{r}_{nn}|^{2(N-n)}$$

(13)

where $d_n = \det^2(L_i) = \prod_{i=1}^{n} |\overline{r}_{i,i}|^2$, and $L_i$ is a sublattice spanned by $q_1, \ldots, q_n$. After swapping the columns in the $k$th iteration, the new potential $D_k$ can be calculated as

$$D_k = \frac{|\overline{r}_{k-1,k}|^2 + |\overline{r}_{k,k}|^2}{|\overline{r}_{k-1,k-1}|^2}D = f(k)D$$

(14)

Hence, to maximize the degradation of the potential $D$, this criterion selects the iteration with the minimum $f(k)$. In [26], [27], the function $f(k)$ is directly utilized, whereas, in the first algorithm of [25], a relaxed version of $f(k)$ is adopted to simplify the selection process and to enhance the parallelism. Another criterion aims at decreasing the error propagation of MIMO detectors, as presented in the second algorithm of [25]. This criterion considers the facts that in MIMO detectors, e.g., SIC and K-best detectors, the signals are detected from back forward, and that the previously detected signals have significant effects on the correctness of the latter detection. Therefore, this criterion lies its priority on the backmost iteration that violates the condition inequality of (12). To sum up, the first criterion can realize higher performance, but it entails complicated division and comparisons, so its complexity is relatively higher. By contrast, the second criterion has low complexity, but its performance is also lower. In the existing greedy LLL algorithms, these two selection criteria are alternatively adopted, yet no design benefits from them both. Furthermore, only one iteration is selected at each stage, leading to severe latency for hardware implementation.

III. PROPOSED PREPROCESSING ALGORITHM

This section introduces the proposed preprocessing algorithm in Algorithm 2, including a sorting-relaxed QR decomposition algorithm and a modified greedy LLL algorithm, for $16 \times 16$ MIMO systems.

A. SORTING RELAXED QR DECOMPOSITION

In Algorithm 2, lines 4-20 demonstrate the sorting-relaxed QR decomposition algorithm for decomposing the channel matrix $H$ as (2). This algorithm employs the CORDIC algorithm to achieve low hardware overhead and high frequency performance. In addition, a relaxed sorting strategy is designed to alleviate the long-latency problem.

1) $\ell^1$-NORM

In conventional GR-based SQRD algorithms, $\ell^2$-norm entails complicated square operations during the initialization and updating processes. Compared with $\ell^2$-norm, the $\ell^1$-norm entails only simple adders, as presented on lines 5-9 of Algorithm 2. Therefore, $\ell^1$-norm is adopted in the SRQRD algorithm to reduce the hardware costs with negligible performance degradation. Notice that the norm values in SQRD are always positive. Therefore, unsigned comparators can meet the requirement for hardware design.

2) PREDICTIVE SORTING STRATEGY

Conventional GR-based SQRD algorithms swap only one column at each stage. This strategy leads to numerous idle clock cycles because the nullifying procedures for the next column can not start until the current column is completely processed. In the proposed SRQRD algorithm, as presented on lines 11-12 of Algorithm 2, $k$ columns with the minimum norm values are selected at one time, so the nullifying procedures for these columns can be performed in parallel (as detailed below). As the $(m_2 \sim m_k)$th columns are selected predictably, the false predictions would inevitably cause some side effects. According to the performance evaluation as presented below, the performance degradation caused by this strategy is negligible under a parameter $k = 4$.

3) PARALLEL CORDIC PROCESS

After the sorting procedure in each stage, the CORDIC operations (line 13 of Algorithm 2) can achieve the parallelism at two levels. Fig.1 takes a $4 \times 4$ matrix as an example to interpret this parallelism in detail. In Fig.1, the parameters
Algorithm 2 Proposed Preprocessing Algorithm

1: Input: $H \in \mathbb{C}^{N \times N}$, $y \in \mathbb{C}^{N \times 1}$, stage, $\delta_1$, $\delta_2$;
2: Output: $R$, $T$, $\tilde{y}_L$;
3: Initial: $R = |H[y]|$, $P = P^{N \times N}$, $k = 4$;
4: // SRQD:
5: for $j = 1 : N$ do
6: for $i = 1 : N$ do
7: $\text{norm}(j) = |\text{Re}((\tilde{r}_{i,j}))| + |\text{Im}((\tilde{r}_{i,j}))|$
8: end for
9: end for
10: for $s = 1; s \leq N; s = s + k$ do
11: $\{m_1, m_2, \ldots, m_k\} = \text{arg min}_{d=s,s+1-N} \{\text{norm}(d)\}$;
12: swap the $(m_1, m_2, \ldots, m_k)$ columns to the front of $(s-N)$th columns in $R$, $P$, $\text{norm}$;
13: parallel CORDIC for the $(m_1, m_2, \ldots, m_k)$th columns in $R$;
14: for $j = s : N$ do
15: for $i = s : s + k - 1$ do
16: $\text{norm}(j) = |\text{Re}((\tilde{r}_{i,j}))| + |\text{Im}((\tilde{r}_{i,j}))|$
17: end for
18: end for
19: end for
20: $T = P$;
21: // MGLLL:
22: for $s = 1; s \leq \text{stage}; s = s + 1$ do
23: if $s \in \text{odd}$ then
24: $n_{\text{list}} = [N/2 + 2, N/2 + 4, \ldots, N]$;
25: else
26: $n_{\text{list}} = [N/2 + 1, N/2 + 3, \ldots, N - 1]$;
27: end if
28: sel1 = $[0, 0]$; sel2 = $[0, 0]$;
29: for $n = n_{\text{list}}$ do
30: efficient size-reduction (lines 7-11 of Algorithm 1);
31: $\eta = |\tilde{r}_{n,n}|/|\tilde{r}_{n-1,n-1}|$
32: bit0 = $\eta \leq \delta_i$;
33: bit1 = $\eta \leq \delta_0$;
34: $\text{prio} = \{\text{bit1}, \text{bit0}\}$ // unsigned integer
35: if $\text{prio} \geq \text{sel}2(2)$ then
36: sel2 = $[n, \text{prio}]$;
37: end if
38: if $\text{sel}1(2) < \text{sel}2(2)$ then
39: Swap sel1 and sel2;
40: end if
41: end for
42: for $n = [\text{sel}1(1), \text{sel}2(1)]$ do
43: swap columns $n-1$ and $n$ in $R$ and $T$
44: update $R$ by Givens Rotation using CORDICs;
45: end for
46: end for
47: for $m = 1; m \leq N - 1; m = m + 1$ do
48: for $n = N; n \geq m + 1; n = n - 1$ do
49: $\mu = |\tilde{r}_{n-m,n}|/|\tilde{r}_{n-m,n-m}|$;
50: $R_{1:N,n} = R_{1:N,n} - \mu R_{1:N,n-m}$;
51: $T_{1:N,n} = T_{1:N,n} - \mu T_{1:N,n-m}$;
52: end for
53: end for
54: $[R, \tilde{y}_L] = \tilde{R}$; $T_L = T$.

FIGURE 1. An example of the parallel CORDIC operations presented on line 13 of Algorithm 2. (with parameters $N = 4$, $s = 1, k = 4$).

of Algorithm 2 are assumed as $N = 4$, $s = 1$, and $k = 4$. In the matrices, letters $R$ and $C$ represent the real-valued and complex-valued entries, respectively. Each arrow indicates a CORDIC operation, and the color is utilized to distinguish the scaling types after the CORDIC procedures. Notice that after each CORDIC operation, the elements are magnified by $K_n$ (generally $K_n = 1.647$). Therefore, a scaling operation is generally utilized to multiply these elements with $1/K_n$. In Fig.1, scale_x0 denotes a CORDIC operation without scaling, whereas scale_x1 and scale_x2 denote the CORDIC operations followed by scaling factors of $1/K_n$ and $1/K_n^2$, respectively. As two scale_x1 operations are replaced by a scale_x2 operation in some cases, the amount of scaling operations can be reduced in the SRQD algorithm. As shown in Fig.1, level-1 parallelism exists within the same column that the matrix entries are processed concurrently, whereas the conventional GR-based SQRD algorithm uses $r_{ii}$ to nullify $r_{i+i:N,i}$ one by one. Level-2 parallelism exists across the columns that the next column can start the CORDIC operations before the current column is completely processed. According to the schedule as presented below, merely 48 CORDIC cycles are enough for the SRQD algorithm to decompose $16 \times 16$ complex-valued matrices. Whereas in the conventional SQRD algorithm with a serial nullifying strategy, the number of CORDIC cycles can be evaluated as

$$ P_{\text{cordic}} = N + \sum (1 : N - 1) \quad (15) $$

where $N$ represents the matrix size. When $N = 16$, the conventional SQRD algorithm would take 136 CORDIC cycles. Therefore, the proposed SRQD algorithm is promising to reduce the latency by approximately 65% compared with conventional SQRD.

B. MODIFIED GREEDY LLL ALGORITHM

Lines 21-53 of Algorithm 2 present the modified greedy LLL Algorithm, called the MGLLL algorithm, for $16 \times 16$ MIMO systems. Compared with conventional greedy LLL algorithms, this algorithm is more friendly for hardware implementation because the complexity is fixed. Moreover, a novel selection criterion is proposed in the MGLLL algorithm, which comprehensively benefits from the two conventional criteria. In addition, two or more iterations can be concurrently selected, which can notably reduce the
processing latency. Fig.2 takes a $16 \times 16$ matrix as an example to describe the even and odd stages of the MGLLL algorithm.

1) PARALLEL SCHEME WITHIN STAGES

Conventional greedy LLL algorithms perform the iterations in the serial mode because of the data dependency. These algorithms suffer from severe latency for hardware implementation. To address it, the proposed MGLLL algorithm adopts a parallel scheme within the stages to reduce latency. As presented in Fig.2, each stage of the MGLLL is divided into numerous iterations, and each iteration corresponds to a pair of columns. In an iteration, the efficient size-reduction is first performed (line 30 of Algorithm 2). Then, the Siegel condition (line 31-33 of Algorithm 2) is checked, based on which a quantitative priority is calculated (line 34 of Algorithm 2). Finally, two iterations with the highest priorities are selected to perform the LLL reduction, including a column swap operation together with a GR updating. In Fig.2, the first parallelism exists within a stage that the iterations are independent of each other, so the for-loop on lines 29 and 42 can be performed in parallel. Another parallelism comes from the Siegel condition that the condition check procedure and the size-reduction procedure are free from data dependency. Therefore, the iteration selection based on the Siegel condition can also be performed in parallel with the size-reduction procedure.

2) ITERATION SELECTION CRITERION

Based on this parallel scheme, a novel selection criterion is designed in this paper to select the most urgent iterations to perform the LLL reduction process. Lines 31-40 of Algorithm 2 interprets this criterion, which comprehensively benefits from the two basic criteria (as stated in section II-C). This selection only considers the rear four iterations, as stated on line 23-27 of Algorithm 2, because the front columns are well-conditioned after the SRQRD process. In this selection criterion, the Siegel potential functions $\eta$ is first calculated for each iteration, just like the first conventional criterion. Then, the $\eta$ is compared with $\delta_l$ and $\delta_s$ to generate the metric variable $\text{prio}$. Finally, two iterations with the maximum $\text{prio}$ values are selected by this stage. For the iterations with the same $\text{prio}$ values, the backmost iteration has the highest priority in this selection, which is the similar to the second conventional criterion. Compared with the conventional criterion that directly uses the potential $\eta$ for comparisons, this method uses a 2-bits variable $\text{prio}$ instead, so the register length is remarkably reduced. In addition, the complicated division operation (line 31 of Algorithm 2) can be converted to multiplications of $\bar{r}_{n-1,n-1}$ with $\delta_l$ and $\delta_s$. Therefore, by properly setting the values of $\delta_l$ and $\delta_s$, such as $\delta_l = 0.75$ and $\delta_s = 0.5$, this operation can be achieved by simple shifters and adders. More importantly, two or more iterations can be concurrently selected at one time, so the convergence is notably enhanced, and merely 6 stages are sufficient to achieve a near-LLL performance, whereas in other greedy LLL algorithms, 6 stages can only meet the requirements of smaller $4 \times 4$ matrices [25], and tens of stages are required for larger sized matrices. Compared with the non-greedy LLL algorithms, the proposed MGLLL algorithm selects only two iterations out from the 8 iterations. Thereby, the column swaps can be reduced by approximately 75%. After performing all these stages, a full-size reduction (FSR) is utilized to improve the orthogonality of matrix $R$ further, as presented on lines 47-53 of Algorithm 2.

C. PERFORMANCE EVALUATION

To evaluate the performance of this preprocessing algorithm, a $16 \times 16$ MIMO simulation system is designed in this paper. K-best and SIC detectors are employed in the simulation to test the BER performance. In addition, 64-QAM is adopted as the modulation scheme, and a [133 171] convolutional code is employed together with an interleaver. The channel is assumed to exhibit Rayleigh fading, and the channel matrix $H$ is assumed to have been properly estimated. The elements of $H$ are complex-valued random numbers drawn from Gaussian distribution with zero mean and unitary variance. The transmitted data comes from a random bitstream, and a frame of convolutional code consists of 160 MIMO symbols. For each simulation, 100000 frames are transmitted for statistical analysis. During the simulation, the parameters of Algorithm 2 are configured as $\delta_l = 0.75$, $\delta_s = 0.5$, and $\text{stage} = 6$, according to the simulation results of Fig.3. In Fig.3, 10000 matrices $H$ are processed using the proposed preprocessing algorithm with different parameters $\text{stage}$ and $\delta_l$ ($\delta_s$ is set as $\delta_s = \delta_l - 0.25$). Then, the average condition numbers of the resulting matrices $R$ are calculated for different parameter configurations. As a lower condition number of matrix $R$ leads to higher BER performance for MIMO detectors, the parameters $\delta_l$, $\delta_s$, and $\text{stage}$ can be determined accordingly by Fig.3.

Fig.4 employs the K-best ($K=10$) detector to compares the BER performance of different preprocessing algorithms in a $16 \times 16$ MIMO system. As the majority of existing prepro-
cessors are modified from the $\ell^2$-norm SQRD algorithm and standard LLL algorithm, the combination of SQRD and LLL algorithms is demonstrated in Fig.4 as a reference processor. Compared with the reference preprocessor, the proposed design (SRQRD+MGLLL) suffers a performance degradation of merely 1dB at a BER target of $10^{-5}$. The Cholesky method is proposed in [7], which consists of an $\ell^2$-norm SQRD and a partial iterative LLL algorithm. The proposed preprocessor exhibits a similar performance to the Cholesky method, denoting that the $\ell^1$-norm and the predictive sorting strategy in the SRQRD algorithm cause negligible performance degradation while reducing latency and area. The combination of SRQRD and Full_PLLL in Fig.4 adopts a similar parallel LLL scheme to this paper, except that the Full_PLLL is a non-greedy LLL algorithm as presented in [22]. The comparison of this paper with the Full_PLLL case indicates that the selection criterion in the MGLLL algorithm causes negligible side effects for K-best detectors. The difference between the CGLLL_v2 and MGLLL algorithms is that the CGLLL_v2 algorithm utilizes the second conventional selection criterion. The comparison with the CGLLL_v2 curve indicates that the proposed selecting criterion in the MGLLL algorithm is superior to the second conventional method. Although the first conventional selection criterion (CGLLL_v1) achieves an identical performance to our criteria under the proposed algorithm scheme, however, its calculation is rather complicated, as stated above. Therefore, the proposed selection criterion can absorb the high performance and low complexity properties of the first and second conventional selection criteria, respectively. Other existing greedy LLL algorithms can also realize a near-LLL performance, but 6 stages are required for $4 \times 4$ matrices [25] and 9 stages are required for larger sized $8 \times 8$ or $16 \times 16$ matrices. Therefore, the proposed MGLLL algorithm has a faster convergence, denoting to lower latency.

The above simulations are conducted using the K-best MIMO detector, which indicates that the proposed preprocessing algorithm can significantly improve the detection performance; and that the proposed SRQRD and MGLLL algorithms suffer from negligible performance degradation while reducing latency and complexity compared with other conventional SQRD and non-greedy LLL algorithms. To confirm that the proposed preprocessor maintains its advantages for other MIMO detectors, Fig.5 shows the same simulations conducted for SIC detectors. Notice that in this figure, the CGLLL_v1 is omitted for a bright exhibition, because it is almost identical to the proposed design. According to Fig.5, similar conclusions can be drawn to that from the K-best detectors. Additionally, the proposed design exhibits a better superiority than the Cholesky method in high SNR scenarios.

### D. Complexity Analysis

This subsection analyzes the complexity of the proposed preprocessing algorithm. Table I illustrates the computational complexities of the proposed SRQRD algorithm together with other SQRD algorithms summarized in [7], [29]. Complex-valued matrices are considered in this table, and N represents the matrix size. A complex-valued multiplication (CM) is equivalent to four real-valued multiplications (RM) and two real-valued additions. A complex-valued addition (CA) is equivalent to two real-valued additions (RA). As suggested by [7], the real-valued square root and division...
calculations are each equivalent to an RM. Therefore, the computational complexities of the HT, GS, GR, and Cholesky algorithms can be evaluated by the number of RA and RM operations in Table 1. As the proposed SRQRD algorithm is realized by a CORDIC array, the complexity is evaluated with the number of 2-D CORDIC operations, which is presented as

\[
O = \sum_{c=1}^{N} c^2 + \sum_{c=2}^{N-1} 2c(c-1) = N^3 + \frac{1}{2}N^2 - \frac{1}{2}N \tag{16}
\]

where \(c\) indicates the column index from right to left of the matrix \(R\), and \(N\) is the matrix size. In (16), the left summation represents the CORDIC operations utilized to eliminate the imaginary parts of matrix entries and to update the following entries of the same row, while the right summation represents the CORDIC operations each utilized to zero a real-valued element or to update the following two elements. Assume a 2-D CORDIC operation is equivalent to \(\varphi\) real-valued multiplications.

\[\text{TABLE 1. Computational complexities of various SQRD algorithms.}\]

| Algorithm | Real-valued additions | real-valued multiplications | sorting |
|-----------|-----------------------|-----------------------------|---------|
| HT        | \(2N^4 + 5N^3 + \frac{21}{2}N^2\) | \(\frac{8}{3}N^4 + \frac{20}{3}N^3 + 14N^2\) | N-1     |
| GS        | \(4N^3 - 2N\) | \(4N^3 + 4N^2 + N\) | N-1     |
| GR*       | \(4N^3 + \frac{15}{2}N^2 - \frac{23}{4}N\) | \(\frac{16}{3}N^3 + \frac{23}{4}N^2 - \frac{107}{6}N\) | N-1     |
| Cholesky  | \(2N^3 - 3N^2 + 2N\) | \(3N^3 + 5N\) | N-1     |
| SRQRD\(^{b}\) | - | \(\varphi(N^3 + \frac{1}{2}N^2 - \frac{1}{2}N)\) | \(\frac{1}{2}N\) |

\(^{a}\) Realized by matrix multiplications.

\(^{b}\) One 2-D CORDIC operation is equivalent to \(\varphi\) real-valued multiplications.

where \(c\) indicates the column index from right to left of the matrix \(R\), and \(N\) is the matrix size. In (16), the left summation represents the CORDIC operations utilized to eliminate the imaginary parts of matrix entries and to update the following entries of the same row, while the right summation represents the CORDIC operations each utilized to zero a real-valued element or to update the following two elements. Assume a 2-D CORDIC operation is equivalent to \(\varphi\) RMIs, the complexity of the SRQRD algorithm can be quantified as \(\varphi(N^3 + \frac{1}{2}N^2 - \frac{1}{2}N)\) RMs. According to our hardware design experience, the factor \(\varphi\) is approximately 2.5 when the CORDIC is configured with 10 iterations as in this paper. Notice that the sorting operations are listed separately in Table 1, therefore, the \(\frac{1}{2}N^3\) RMs for sorting operations presented in [7] are not included in the RM item. As listed in Table 1, the complexity of the proposed SRQRD algorithm is notably lower than those of the HT, GS, and GR algorithms, and is slightly lower than that of the Cholesky algorithm. Assume \(N=16\) and \(\varphi = 2.5\), the SRQRD algorithm can reduce the number of RMs by 94.9%, 39.5%, 57.0%, and 14.8% compared with the HT, GS, GR, and Cholesky algorithms, respectively. Moreover, the number of sorting operations is also reduced remarkably as listed in Table 1, which helps the SRQRD algorithm achieves lower latency than other algorithms.

Table 2 compares the complexity of the proposed selection criterion with the two conventional criteria. For a fair comparison, these criteria are assumed to be utilized in the same algorithm scheme of Algorithm 2, and the corresponding complexities for selecting two iterations in a stage are listed in Table 2. In addition, the parameters \(\delta_i\) and \(\delta_1\) in the MGLLL criterion are respectively set as 0.5 and 0.75, according to the simulation below. The parameters \(\delta\) in the two conventional criteria are both set to be 0.75. In the first conventional criterion, \(\frac{1}{4}N\) division operations are performed to calculate the \(\eta_i\) for each iteration in a stage, and \(N - 3\) comparisons are utilized to select the minimum two \(\eta_i\). For the second conventional criterion, \(\frac{1}{4}N\) multiplications and comparisons are used to compare \(\delta_i[r_{n-1,n-1}]\) with \(|r_{n,n}|\) for each of the \(\frac{1}{4}N\) iterations. As \(\delta = 0.75\), the multiplication with \(\delta\) is equivalent to an addition in this table. For the MGLLL criterion, the multiplication with \(\delta_i\) is negligible in complexity, so \(\frac{1}{4}N\) additions and \(\frac{1}{2}N\) comparisons are utilized to compare \(|r_{n,n}|\) with \(\delta_i|r_{n-1,n-1}|\) and \(\delta_1|r_{n-1,n-1}|\) for each of the \(\frac{1}{4}N\) iterations. To select two iterations from the \(\frac{1}{4}N\) candidates, \(\frac{1}{2}N - 3\) additional comparisons are required in the MGLLL criterion. Compared with the first conventional criterion, the \(\frac{1}{2}N\) divisions are substituted by \(\frac{1}{2}N\) additions in the MGLLL criterion, therefore, the computational complexity can be remarkably reduced. Although the MGLLL criterion takes more comparisons than the second conventional criterion, considering that some of these comparisons in the MGLLL criterion are performed among shorter 2-bit signals (the prio signal in Algorithm 2), this complexity increase is ignorable. To sum up, the complexity of the second conventional criterion is notably lower than that of the first criterion, and the proposed MGLLL criterion benefits from the low complexity of the second conventional criterion.

\[\text{TABLE 2. Computational complexities of different selection criteria for greedy LLL algorithms.}\]

| Criterion      | divisions | comparisons | additions |
|----------------|-----------|-------------|-----------|
| Conventional-v1| \(\frac{1}{2}N\) | \(N - 3\) | 0         |
| Conventional-v2| 0         | \(\frac{1}{2}N\) | \(\frac{1}{2}N\) |
| MGLLL criterion| 0         | \(N - 3\) | \(\frac{1}{4}N\) |

IV. HARDWARE ARCHITECTURE

Based on this preprocessing algorithm, the corresponding hardware architecture is also proposed in this paper. This architecture is designed for \(16 \times 16\) MIMO systems, and 64-QAM is adopted as the modulation scheme. To save IO ports, the complex-valued matrix \(H (H = [h_1, h_2, \ldots, h_{16}])\) is transferred at column-wise, and the vector \(y\) follows behind \(H\). Highly-parallelized pipeline scheme is designed to improve throughput, and hardware reuse is adopted to save area.

Fig.6 presents the top block of this preprocessor, which consists of an SRQRD component (the upper part) and an MGLLL (the lower part) component. During the SRQRD component, a tree adder (TA) is first utilized to calculate the norm \(\|m\|\) for each column \(h_i\). Meanwhile, the ID unit gives out the index signal \((id)\) for each \(h_i\). Then, these columns are transferred through the 4 SRQRD-i modules, together with the \(nm\) and \(id\) signals, to generate the upper-triangular matrix \(R\) and the sorted \(id\) sequence.
is converted into a permutation matrix $\mathbf{P}$ by the id2P unit. After the SRQRD component, the columns of $\mathbf{R}$ and $\mathbf{P}$ are sent to the MGLLL component for further processing. In the MGLLL component, the 6 LR-i modules are first employed, correspond to the 6 stages as stated on lines 22-46 of Algorithm 2. After that, the FSR-R and FSR-T modules are utilized to perform the full-size reductions on matrices $\mathbf{R}$ and $\mathbf{T}$, respectively. Notice that the matrices $\mathbf{R}$ and $\mathbf{T}$ are transferred at column-wise before the FSR modules, whereas in the FSR-R and FSR-T, they are processed row by row from the bottom up.

A. SRQRD COMPONENT

The SRQRD component mainly consists of a TA unit, an ID unit, an id2P unit, and 4 SRQRD-i units. The TA unit takes two clock cycles to calculate the norm for each column. The ID unit is a 4-bit counter which gives out the indexes from 0 to 15. The Up-norm unit takes two clock cycles to update the norm values, as listed on lines 14-18 of Algorithm 2. As the TA, ID, Up-norm, and id2P units are structurally simple, they are omitted in the detailed exhibition. The architectures of the sorting unit and the GR array (GRA) unit are demonstrated as follows.

1) GRA UNIT

The GRA units are utilized to zero the lower-triangular part of the matrix $\mathbf{H}$, and each GRA unit corresponds to 4 columns of $\mathbf{H}$. Fig.7 demonstrates the block diagram of the 4 GRA units, which are composed of two kinds of basic GR modules, i.e. GR-e and GR-v modules, and a series of scaling modules and buffers. The GR-e module is utilized to zero the imaginary part of an element, while the GR-v module can nullify a real-valued element. The number i in a GR-v block indicates that this row is processed by a GR-v module together with the ith row, and the color represents the column index of the nullified element. The scale_x0, scale_x1, and scale_x2 represent the scaling modules behind the CORDIC cells with scaling factors of $1/K_s$, $1/K_{s2}$, respectively. Fig.7 indicates that the GR-e and GR-v modules can work in high parallel, due to the proposed sorting strategy. The four GRA units take 14, 14, 12, and 8 CORDIC cycles respectively to decompose a $16 \times 16$ complex-valued matrix $\mathbf{H}$.

The GR-e and GR-v modules are presented in Fig.8, which are utilized to process the element flow and vector flow, respectively. As the GR-v modules are always placed behind the GR-e modules, the first input vector in a GR-v module is filled with real-valued elements. For the first input signal, the CORDIC cells in the GR-e module and the upper part of the GR-v module work in vector mode to calculate the rotation angles. Based on these angles, the three CORDIC cells then work in rotation mode to update the following signals. In this paper, the CORDIC cell is designed with 5 pipeline stages, and each stage includes 2 CORDIC iterations, according to the simulation as stated below. Considering an additional clock for the scaling operation, a CORDIC cycle equals to 6 clock cycles in the hardware implementation.

2) SORTING UNIT

The sorting unit is utilized to select four columns with the minimum norm values and swap them to the front. As the swap operations for signals $\mathbf{H}$, norm, and index are the same, this paper takes the signal norm as an example to show the architecture in detail. Fig.9 demonstrates the sorting circuit for signal norm, which consists of three types of registers, four comparators, and a series of multiplexers and control signals. The Reg-A register chain is utilized for buffering the successive norm signals. This chain is long enough to store all the norm signals for the columns that have not been decomposed. The Reg-C registers are utilized as memories to store the currently smallest norm values, and these values increase from left to right. During each clock, the input norm signal is compared with the four Reg-C registers. If it is smaller than any of these registers, it will be inserted into the Reg-C chain, and the rightmost Reg-C will be shifted out. After generating all the norm signals, the signals in Reg-C will be sent out during the first four clock cycles. Meanwhile, the four data shifted out from Reg-A are temporarily stored in Reg-B. During the followed clock cycles, the addr2 is initialized as 0 to output other norm signals. If the current output signal has already been selected previously in Reg-C, the addr2 will be increased by 1 to output the next signal. As presented in Fig.9, this circuit takes totally $(N - s + 2)$ clock cycles to sort the columns from the $s$th to the $N$th.

3) SUMMARIZE OF THE SRQRD ARCHITECTURE

Similar architectures about the QRD/SQRD have previously been proposed by [7] and [6]. In [7], an SQRD architecture is proposed for $16 \times 16$ MIMO systems. This architecture is based on the Cholesky algorithm, and the sorting process is based on the diagonal elements of the Gram matrix $\mathbf{G}$. As the sorting operations are performed before each stage, and the sorting operations need to wait the updating of all diagonal elements of the Gram matrix $\mathbf{G}$, the architecture of [7] suffers from long latency. In [6], the QRD processor is designed for $8 \times 8$ MIMO systems. The CORDIC method and matrix multiplication are both employed in this processor. To keep with the pace of matrix multiplication, the CORDIC modules in [6]
FIGURE 7. Block diagram of the GRA units. The matrix R is input at column wise, and the colors distinguish the column index of the nullified element.

FIGURE 8. Block diagram of the GR-e and GR-v units.

FIGURE 9. Circuit of the sorting unit for signal norm. (The s-nm module of the top block.)

are each allocated with 1 clock, which has a severe impact on the frequency performance. Compared with [7], the proposed SQRD architecture only needs four sorting modules, thereby, the latency is remarkably reduced. In addition, the decompositions for four columns are performed in parallel, so the idle clock cycles are notably reduced and the latency can also be reduced further. Compared with [6], the CORDIC modules in this architecture are deeply pipelined, which helps the architecture achieve an excellent frequency performance to meet the high-throughput requirement of future communication.

B. MGLLL COMPONENT

As shown in Fig.7, the MGLLL component consists of 6 LR-i units and two full-size reduction units. In each LR-i unit, the columns of matrix R are sent to the SR-R module to perform size reduction. Meanwhile, the Sel module acquires the diagonal elements $r_{i,j}$ from these columns to select the most urgent two iterations. The selection results are transferred to the LLLR module via the sel signal, and the selected iterations are further processed by the PE modules in the LLLR unit. As the architecture for signal $T$ is the same as that for the upper rows of matrix $R$, it is omitted in the detailed exhibition. The architectures for the SR-R, Sel, LLLR, and FSR-R units are presented as follows.

1) SR-R UNIT

The SR-R unit is utilized to perform the size-reduction procedures on each column pair of the matrix $R$. Fig.10 demonstrates the architecture of this unit, which consists of a division and rounding circuit and 16 SR-cells. Notice that this architecture is designed for the odd stages. Therefore, the multiplexer chooses elements from the (9, 11, 13, 15)th rows. For the even stages, the architectures are similar, except that the elements are chosen from the (8, 10, 12, 14)th rows. In the lower part of this architecture, the signal $r_{i,j+1}$ is divided by $r_{i,j}$ in a pipelined divider, and the quotient is rounded by the following module to generate the signal $\mu$. The divider includes two pipeline stages, and each calculates two bits of the quotient. After the Round module, the signal $\mu$ is constructed with 1 bit for sign, 3 bits for integer, and an additional 1 bit for rounding result, in both of the real and complex parts. As the signal $\mu$ is valid every two clock cycles, therefore, an additional multiplexer is utilized to output zero during the invalid clock cycles. The signal $\mu$ is
2) SELECTION UNIT
The Sel unit is utilized to select the two iterations according to the proposed criterion. Fig.11 presents the architecture of this unit, which consists of three fragments, i.e., the Condition check, Sort & selection, and Reorder fragments. During the first fragment, the Siegel condition inequations are checked by two comparators. As $\delta_s$ and $\delta_l$ are respectively 0.5 and 0.75, the two multipliers can be simply realized by shifters and adders. During the second fragment, the checking results are combined with the addr signal to form the 4-bits prio signal. This signal is passed through the sorting circuit, and the largest two values are selected by the two registers. Notice that only $\text{prio}[3:2]$ is utilized for comparison in CMP3 and CMP4, and that only $\text{prio}[1:0]$ is used in the output signals. Finally, a reordering circuit is adopted to arrange the output port for the selected iteration addresses. This circuit is utilized to avoid the currently selected address being sent to another port.

3) LLL REDUCTION UNIT
The LLLR unit is utilized to perform LLL reduction for the two selected iterations. Fig.12-(a) demonstrates the architecture of this unit, which has two processing elements (PE), corresponding to the two iterations. In each PE module, two rows (denoted as the $i$th and $(i+1)$th rows) of matrix $R$ are selected according to the sel signal. Then, a Col-swap module is used to swap the $i$th and $(i+1)$th columns, after which the GR module is adopted to maintain the upper-triangular property. Finally, another Col-swap module is used to perform the same swap operation as in another PE unit. Outside the PE units, the columns are passed through the Col-swap module to keep pace with the swapping operations in PE-1 and PE-2 modules. Fig.12-(b) presents the architecture of the GR module, which consists of three CORDIC stages. The first stage converts $r_{i,i}$ to real, and the second stage nullifies the $r_{i+1,i}$ element. Finally, the third stage ensures the diagonal elements to be real.

4) FULL-SIZE REDUCTION UNIT
The FSR-R and FSR-T units are utilized to perform the full-size reduction on lines 47-53 of Algorithm 2. In this paper, only the FSR-R unit is presented graphically, and the architecture for the FSR-T unit is similar. As shown in Fig.13-(a), the FSR-R unit is composed of a Column2Row module and a series of Fcell modules and buffers. The Column2Row uses a register chain to accumulate the successive columns of matrix $R$. After that, the matrix $R$ is sent out row by row from the bottom up. The symbol $r_{-j}$ denotes an element flow of the $i$th column. The Fcell is a single size-reduction cell, which can work as a divider or as a subtractor. For the elements $r_{i,j}$, the Fcell module works as a divider to calculate the factor $\mu$ and the reduced $r_{i,j}$ element. After that, the factor $\mu$ is stored in a memory, and the Fcell works as a subtractor to subtracts the $r_{-j}$ with $\mu$ multiple of $r_{-j}$. As the Fcell is designed for complex-valued elements, the subtraction in the Fcell module is realized by 4 real-valued subtractors. Fig.13-(b) shows the architecture for these real-valued subtractors, which can work as a divider or a subtractor according to the selection of the green multiplexers. When the green multiplexers are set to ‘0’, the Div_cell works as a divider to calculate $q_0 = [a_2/a_1]$. 
and \( rem = a_2 - qa_1 \). Otherwise, it works as a subtractor to calculate \( rem = a_2 - qa_1 \). In Fig. 13-(b), both \( q_1 \) and \( q_0 \) are signed integers, and \( q[3:1] \) represents the absolute integral fragment while \( q[0] \) is a rounding bit.

5) SUMMARIZE OF THE MGLLL ARCHITECTURE

Literatures [22] also proposes an LR architecture for MIMO detections. In [22], the LR processor adopts a paralleled odd-even scheme, and the number of stages can be easily configured for achieving the optimal trade-off between BER performance and throughput. For each stage, the whole matrix is input concurrently, and the elements of the same row are processed by different CORDIC modules. Therefore, this architecture would require numerous IO ports and complicated wire connections if it is extended for larger-scale MIMO systems. In addition, the real-valued data format and the Lovasz condition in [22] also impede its parallelism. Compared with [22], the MGLLL architecture designed a CORDIC unit that can work in both rotation and vector modes. Thereby, the elements of the same row can be successively processed in the same CORDIC unit and the wire connection is simplified remarkably. Unlike in [22], the MGLLL architecture adopts a complex-valued data format together with the Siegel condition to improve the parallelism. Most importantly, only two LLL reduction units are required for each stage in the MGLLL architecture, whereas eight units would be required in [22] if it is extended for \( 16 \times 16 \) MIMO systems. Therefore, the area utilization of the MGLLL architecture is higher than the architectures based on the non-greedy LLL algorithms.

C. FIXED-POINT SIMULATION

To determine the iteration number of the CORDIC modules and the word length (WL) for each register, a fixed-point simulation is conducted based on the \( 16 \times 16 \) MIMO link. During this simulation, the proposed preprocessor is adopted, together with a K-best MIMO detector. The parameters are set as \( \delta_s = 0.5, \delta_l = 0.75, \text{stage} = 6 \) in the preprocessor, and \( K=10 \) in the K-best detector. In addition, the calculation for the preprocessing algorithm is performed using the fixed-point data models, whereas the double floating data format is adopted for calculating other blocks in the MIMO link.

The first simulation is conducted about the number of CORDIC iterations. In this simulation, the register length is set long enough to ensure the calculation accuracy, whereas the number of CORDIC iterations varies from 8 to 11. The performance for each configuration is listed in Fig. 14, where the case with \( \text{Ite} = \text{Inf} \) is also listed as a theoretically optimal accuracy model. Notice that the Inf model is actually configured with 20 CORDIC iterations. According to Fig. 14, 10 CORDIC iterations are chosen in this paper to realize a near-optimal accuracy. Considering the frequency property, these 10 iterations are divided into 5 pipeline stages for hardware implementation.

Another simulation is conducted to determine the register length of the matrix \( \mathbf{R} \). First, the fractional part is constructed according to the result of Fig. 15. In Fig. 15, the integer part is set long enough, whereas the number of fractional bits varies from 13 to 16. A double-floating case is also employed to represent the accurate model. As shown in Fig. 15, 15 bits are sufficient for performing a near-accurate calculation. Notice that the 15-bits fraction is only utilized to represent the matrix \( \mathbf{R} \), while other registers, e.g., the signal \( \mathbf{nm} \), may be shorter than 15 bits in the fractional parts. For each register, the number of integer bits is determined to avoid overflow. Table 3 lists the WL structures for some significant registers.

| component | SRQRD | MGLL |
|-----------|-------|------|
| Register  | R, nm, id | R, T, div, \( \mu \) |
| signed    | s, u, u | s, s, s, s |
| integer-bits | 5, 5, 4 | 7, 5, 7, 4 |
| fractional-bits | 15, 6, 0 | 0, 0, 6, 0 |
TABLE 4. Gates and latency distributions of the implemented architecture.

| Functional Blocks | Gate Count | Gate Percentage | Latency (clocks) |
|-------------------|------------|----------------|-----------------|
| SRQRD-1           | 1376k      | 23.4%          | 105             |
| SRQRD-2           | 1102k      | 18.7%          | 99              |
| SRQRD-3           | 756k       | 12.8%          | 83              |
| SRQRD-4           | 316k       | 6.1%           | 53              |
| LR-1              | 228×6      | 3.9%×6         | 25×6            |
| FSR               | 930k       | 15.8%          | 76              |
| Total             | 5891k      | 100%           | 566             |

TABLE 5. Implementation results and comparisons.

| Item                  | [6]     | [7]     | This     |
|-----------------------|---------|---------|----------|
| MIMO Size             | 8×8     | 16×16   | 16×16    |
| Modulation            | 64-QAM  | 64-QAM  | 64-QAM   |
| Operations            | QR=LR   | SQR     | SQR+LR   |
| Implementation        | Chip    | Post-layout | Synthesis |
| Technology            | 90nm    | 65nm    | 65nm     |
| Voltage               | 1.1V    | 1.2V    | 1.2V     |
| Frequency             | 65MHz   | 588MHz  | 625MHz   |
| Gate Count            | 978k    | 3720k   | 5681k    |
| Latency (clocks)      | 187     | 412     | 706      |
| Latency               | 2.9us   | 0.7us   | 1.2us    |
| Ave. cycle per matrix | 175     | 16      | 16       |
| Matrix Rate (Mat/s)^a | 0.4M    | 36.8M   | 39.1M    |
| Vector Rate (Vec/s)   | 65M     | 36.8M   | 625M     |
| Nor. MatRate          | 0.13M   | 36.8M   | 39.1M    |
| Nor. VecRate          | 22.5M   | 36.8M   | 625M     |
| Gate Efficiency       | 0.13    | 9.9     | 6.5      |
| (Mat/s/KG)            | 11.0    | 6.6     |          |

V. IMPLEMENTATION RESULTS AND COMPARISONS

This architecture is synthesized using 65-nm CMOS technology. The voltage is 1.2 V, and the gate count is 5891K, in terms of the two-input NAND gates. Simulation indicates that this preprocessor can work at a maximum frequency of 625 MHz to process the 16 × 16 complex-valued matrices in 566 clock cycles. The power is 3.4 W, and the latency is 0.9 us. Table 4 illustrates the gates and latency distribution of this architecture. Other implementation results are listed in Table 5 to be compared with other similar works. In Table 5, the latency is defined as the duration from the first input to the first output. The matrix rate is evaluated when the preprocessor only processes the channel matrices. In contrast, the vector rate is defined where the preprocessor only performs the QR^H y operations. For a fair comparison, the matrix rate and vector rate are normalized to 65-nm technology and 16 × 16 matrix size in Table 5 as

\[
\text{Nor. MatRate} = \frac{\text{Matrix Rate} \times \text{Tech} \times N^2}{65\text{nm} \times 16^2}
\]

\[
\text{Nor. VecRate} = \frac{\text{Vector Rate} \times \text{Tech} \times N^2}{65\text{nm} \times 16^2}
\]

where N indicates the MIMO dimension. To comprehensively compare the throughput and area overhead, the gate efficiency is also defined in Table 5 as

\[
\text{Gate Efficiency} = \frac{\text{Nor. MatRate}}{\text{Gate Count}}
\]

In Table 5, the proposed design is compared with other similar works presented in [6] and [7]. Literature [6] proposed an 8 × 8 MIMO preprocessor, including the QRD and LR techniques. The QRD is realized with the CORDIC-based GR method, and the LR technique employs a paralleled LLL scheme. As the matrix is not sorted during the QRD process, more stages are required in the LR block, leading to severe latency. Literature [7] introduces a Cholesky pre-processing algorithm together with the VLSI implementation for 16 × 16 MIMO systems. Sorted QRD is designed based on the Cholesky method, and a partial iterative LR scheme is adopted to realize near-optimal performance. Compared with [6] and [7], the proposed preprocessor achieves the highest frequency performance, which is due to the highly pipelined CORDIC modules. The gate count of our design is 5891K for the SQR+LR case, notably higher than that of [6]. For this phenomenon, two reasons can be considered that the matrix size in [6] is smaller than that in our design, and that the hardware blocks are reused in [6] to perform the QRD and LR operations whereas in our design the hardware is pipelined. The second reason above can indeed increase the gate count for our design, however, it also helps our design achieve better throughput and latency performance. The gate count of the SQR item in our design is relatively smaller than that in [7]. But the LR block of our design costs more gates than that of [7], which is because the LR design in [7] does not calculate the matrix T. In terms of the latency, our design can reduce the latency by 29% and 25% in the SQR and SQR+LR cases, compared with those of [7]. Furthermore, the latency reduction even reaches 69% when compared with the design in [6]. The latency superiority of our design is due to the relaxed sorting strategy of the SQRD component, together with the paralleled scheme and selection strategy of the LR component. In Table 5, the throughput is compared from two perspectives, i.e., the matrix rate and vector rate. The matrix rate is more significant for fast varying channel scenarios, while the vector rate is more crucial for slowly varying channel circumstances. As listed in Table 5, literature [7] achieves an outstanding matrix rate of 36.8 M. However, its vector rate is merely 36.8 M because it reuses the hardware to calculate QR^H y in 16 clock cycles. Literature [6] is an opposite case of [7], which achieves an excellent vector rate of 65 M (normalized to 22.5 M) and a relatively inferior matrix rate of 0.4 M (normalized to 0.13 M). Compared with [6] and [7],
our design achieves the most excellent performance in both matrix rate and vector rate. So, the proposed design is appropriate for applications of both fast and slow varying channel circumstances. As the major concern is focused on decomposing the channel matrices, the matrix rate is adopted in the definition of the gate efficiency. Table 5 indicates that the proposed preprocessor achieves a more excellent gate efficiency than other similar works.

VI. CONCLUSION

This paper proposes a pre-processing algorithm that combines the sorting-relaxed QR decomposition and the modified greedy LLL algorithm. In this preprocessor, a relaxed sorting strategy is utilized to reduce latency for QR decomposition, and a paralleled selection criterion is designed for the greedy LLL algorithm to achieve low complexity while maintaining the BER performance. Based on this algorithm, a highly pipelined hardware architecture is also designed in this paper. Comparisons indicate that this preprocessor is superior to the state-of-art designs in terms of the latency, matrix rate, and vector rate performance. This design is appropriate for both scenarios with slow and fast channel changes to achieve high gate efficiency. Future work will focus on the low-complexity MIMO detectors.

REFERENCES

[1] P. Varzakas, “Average channel capacity for Rayleigh fading spread spectrum MIMO systems,” Int. J. Commun. Syst., vol. 19, no. 10, pp. 1081–1087, Dec. 2006.
[2] S. Yang and L. Hanzo, “Fifty years of MIMO detection: The road to large-scale MIMOs,” IEEE Commun. Surveys Tuts., vol. 17, no. 4, pp. 1941–1988, 4th Quart., 2015.
[3] W. van Etten, “Maximum likelihood receiver for multiple channel transmission systems,” IEEE Trans. Commun., vol. 24, no. 2, pp. 276–283, Feb. 1976.
[4] A. Trimeche, N. Boukad, A. Sakly, and A. Mitbia, “Performance analysis of 2E and MMSE equalizers for MIMO systems,” in Proc. 7th Int. Conf. Design Technol. Integr. Syst. Nanos. Era, May 2012, pp. 1–6.
[5] B. Y. Kong and I. C. Park, “Improved sorting architecture for K-best MIMO detection,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 64, no. 9, pp. 1042–1046, Sep. 2017.
[6] C.-F. Liao, J.-Y. Wang, and Y.-H. Huang, “A 3.1 Gb/s 8×8 sorting reduced K-best detector with lattice reduction and QR decomposition,” IEEE Trans. Very Large Scale Integ. (VLSI Syst.), vol. 22, no. 12, pp. 2675–2688, Dec. 2014.
[7] G. Peng, L. Liu, S. Zhou, Y. Xue, S. Yin, and S. Wei, “Algorithm and architecture of a low-complexity and high-parallelism preprocessing-based k-best detector for large-scale MIMO systems,” IEEE Trans. Signal Process., vol. 66, no. 7, pp. 1860–1875, Apr. 2018.
[8] N. Peer, Y. Marin, and R. Dabora, “Improved QRD-QLD algorithm for low complexity MIMO decoding,” IEEE Commun. Lett., vol. 18, no. 10, pp. 1703–1706, Oct. 2014.
[9] C. Zhang, H. Prabhu, L. Liu, O. Edfors, and V. Owall, “Energy efficient SQRD processor for LTE–A using a group-sort update scheme,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Jun. 2014, pp. 193–196.
[10] A. Chauhan and R. Mehra, “Analysis of QR decomposition for MIMO systems,” in Proc. Int. Conf. Electron. Syst. Signal Process. Comput. Technol., Jan. 2014, pp. 69–73.
[11] D. Wu, J. Eilert, and D. Liu, “Lattice-reduction aided multi-user STBC decoding with resource constraints,” in Proc. IEEE 18th Int. Symp. Pers., Indoor Mobile Radio Commun., Sep. 2007, pp. 1–5.
[12] R. Gangarajaiah, L. Liu, M. Stala, P. Nilsson, and O. Edfors, “A high-speed QR decomposition processor for carrier- aggregated LTE–A downlink system,” in Proc. Eur. Conf. Circuit Theory Design (ECCTD), Sep. 2013, pp. 1–4.
[13] R. Gangarajaiah, O. Edfors, and L. Liu, “An adaptive QR decomposi- tion processor for carrier-aggregated LTE–A in 28-nm FD-SOI,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 64, no. 7, pp. 1914–1926, Jul. 2017.
[14] Y. Miyao, Y. Nagao, M. Kurokaki, and H. Ochi, “Sorted QR decom- position for high-speed MMSE MIMO detection based wireless communica- tion systems,” in Proc. IEEE Int. Symp. Circuits Syst., May 2012, pp. 2857–2860.
[15] D. Shin and J. Park, “A low-latency and area-efficient Gram–Schmidt- based QR architecture for MIMO receiver,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 8, pp. 2606–2616, Aug. 2018.
[16] C. Zhang, H. Prabhu, Y. Liu, L. Liu, O. Edfors, and V. Owall, “Energy efficient group-sort QRD processor with on-line update for MIMO channel pre-processing,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 5, pp. 1220–1229, May 2015.
[17] H. Lee, H. Kim, M. Cho, and J. Kim, “Low-latency implementation of CORDIC-based sorted QR decomposition for high-speed MIMO-OFDM system,” in Proc. 29th Int. Conf. Radioelektronika (RADIOELEKTRONIKA), Apr. 2018, pp. 1–4.
[18] S. Lyu and C. Ling, “Boosted KZ and LLL algorithms.” IEEE Trans. Signal Process., vol. 65, no. 18, pp. 4784–4796, Sep. 2017.
[19] W. Zhang, X. Ma, and A. Swami, “Designing low-complexity detectors based on Seyesen’s algorithm,” IEEE Trans. Wireless Commun., vol. 9, no. 10, pp. 3301–3311, Oct. 2010.
[20] Q. Zhou and X. Ma, “Improved element-based lattice reduction algorithms for wireless communications,” IEEE Trans. Wireless Commun., vol. 12, no. 9, pp. 4414–4421, Sep. 2013.
[21] M. Guenach, “Novel lattice reduction algorithms: Precoder reduction and vector perturbation tradeoffs,” IEEE Commun. Lett., vol. 22, no. 8, pp. 1628–1631, Aug. 2018.
[22] C.-F. Liao and Y.-H. Huang, “Power-saving 4×4 lattice-reduction proce- sor for MIMO detection with redundancy checking,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 2, pp. 95–99, Feb. 2011.
[23] B. Gestner, W. Zhang, X. Ma, and D. V. Anderson, “VLSI implementation of an effective lattice reduction algorithm with fixed-point considerations,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., Apr. 2009, pp. 577–580.
[24] J. Wen and X.-W. Chang, “GfcLLL: A greedy selection-based approach for fixed-complexity LLL reduction,” IEEE Commun. Lett., vol. 21, no. 9, pp. 1965–1968, Sep. 2017.
[25] Q. Wen and X. Ma, “Efficient greedy LLL algorithms for lattice decoding,” IEEE Trans. Wireless Commun., vol. 15, no. 5, pp. 3560–3572, May 2016.
[26] W. Zhang, S. Qiao, and Y. Wei, “A diagonal lattice reduction algorithm for MIMO detection,” IEEE Signal Process. Lett., vol. 19, no. 5, pp. 311–314, May 2012.
[27] K. Zhao, Y. Li, H. Jiang, and S. Du, “A low complexity fast lat- tice reduction algorithm for MIMO detection,” in Proc. IEEE 23rd Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC), Sep. 2012, pp. 1612–1616.
[28] C. Ling and N. Howgrave-Graham, “Effective LLL reduction for lattice decoding,” in Proc. IEEE Int. Symp. Inf. Theory, Jun. 2007, pp. 196–200.
[29] Z.-Y. Huang and P.-Y. Tsai, “Efficient implementation of QR decomposi- tion for gigabit MIMO-OFDM system,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 10, pp. 2531–2542, Oct. 2011.

LIRUI CHEN received the B.S. degree from the School of Microelectronics and Solid-State Electronics, University of Electronic Science and Technology of China, Chengdu, China, in 2014, and the M.S. degree in electrical engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2017, where he is currently pursuing the Ph.D. degree in electrical engineering. His current research interests include VLSI signal processing and wireless communication.
ZUOCHENG XING received the B.S. degree from the Guilin University of Electronic Technology, in 1987, and the M.S. and Ph.D. degrees from the National University of Defense Technology, in 1990 and 2001, respectively. He was a Professor with the School of Computer, National University of Defense Technology. His research interests include microprocessor architecture design, 5G wireless communications, and VLSI architecture design for communication.

YONGZHONG LI received the B.S. degree in radio technology from the Huazhong University of Science and Technology, Wuhan, China, and the M.S. degree in software engineering from the National University of Defense Technology, Changsha, China. He has been engaged in teaching and research on computer science and technology for over 20 years. His research interest includes network communication.

SHIKAI QIU received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2017. He is currently pursuing the master’s degree in electronic science and technology with the National University of Defense Technology, Hunan, China. His current research interests include 5G, microprocessor technology, and VLSI signal processing.

* * *