A SYSTOLIC ARRAY FOR LINEAR MIMO DETECTION BASED ON AN ALL-SWAP LATTICE REDUCTION ALGORITHM

Ni-Chun Wang[1], Ezio Biglieri[2] and Kung Yao[1]

[1]Department of Electrical Engineering, University of California Los Angeles, Los Angeles CA,USA
[2]Departament de Tecnologies de la Informació i les Comunicacions, Universitat Pompeu Fabra, Barcelona, Spain

ABSTRACT

A systolic array to implement lattice-reduction-aided linear detection is proposed for a MIMO receiver. The lattice reduction algorithm and the ensuing linear detections are operated in the same array, which can be hardware-efficient. All-swap lattice reduction algorithm (ASLR) is considered for the systolic design. ASLR is a variant of the LLL algorithm, which processes all lattice basis vectors within one iteration. Lattice-reduction-aided linear detection based on ASLR and LLL algorithms have very similar bit-error-rate performance, while ASLR is more time efficient in the systolic array, especially for systems with a large number of antennas.

Index Terms— MIMO receivers, systolic array, lattice reduction, wireless communications

1. INTRODUCTION

Lattice-reduction-aided detection (LRAD), which combines lattice reduction techniques with linear detections or successive spatial-interference cancellation, has been shown to yield some improvement of error-rate performance [1][2]. In LRAD, the lattice reduction algorithm need be performed when the channel changes. If the channel changing rate is high, or a large number of channel matrices need be processed such as in a MIMO-OFDM system, a fast-throughput hardware structure is needed for real-time applications. To this end, we propose a systolic array to implement the linear LRAD. Systolic array, allowing simple parallel processing, can achieve higher data rates without the demand on faster hardware capabilities. Hence, systolic array may be one of the best solutions for the practical implementation of a MIMO detector.

In this paper, we consider the LRAD based on all-swap lattice reduction (ASLR) instead of the most widely used LLL algorithm [3]. ASLR is a variant of LLL and was first proposed in [5] for real lattices. A complex-number version ASLR is presented in this paper. A crucial difference between ASLR and LLL algorithm is that all lattice basis vectors are simultaneously processed during a single iteration. Since ASLR was originally designed for parallel processing, a systolic array running ASLR is on average more efficient than one running LLL. After lattice reduction, linear detectors, such as zero-forcing (ZF) and minimum mean-square error (MMSE), can also be implemented by the same systolic array without any extra hardware cost.

The following notations are used throughout the remaining sections. Capital bold letters denote matrices, and lower case bold letters denote column vectors. $x_{i,j}$ denotes the *(i,j)*-entry of the matrix $\mathbf{X}$. Submatrix (subvector) formed from the a^{th} to b^{th} rows and m^{th} to n^{th} columns of $\mathbf{X}$ is denoted $\mathbf{X}_{a:b,m:n}$. $(\cdot)^T$, $(\cdot)^H$ and $(\cdot)^\dagger$ denote transpose, Hermitian transpose, and Moore-Penrose pseudoinverse of a matrix, respectively. $\|\mathbf{x}\|$ is the Euclidean norm of the vector $\mathbf{x}$. $[\![x]\!]$ indicates the closest integer to x. $\mathbf{I}_m$ and $\mathbf{0}_m$ are $m\times m$ identity and null matrices, respectively.

2. LATTICE-REDUCTION-AIDED LINEAR DETECTION

We consider a MIMO system with m transmit and n receive antennas in a rich-scattering flat-fading channel. Let $\mathbf{x}$ be the transmitted M-QAM signal vector, $\mathbf{y}$ the received signal vector and $\mathbf{H}$ the $n\times m$ channel matrix where the entries are uncorrelated, zero-mean, unit-variance complex Gaussian fading gains. The baseband model for this MIMO system is

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{n}$ is the white Gaussian noise vector. Additionally, we assume the channel matrix entries are fixed during each frame interval, and the receiver has perfect knowledge of the realization of $\mathbf{H}$.

In MIMO detection, the objective of the lattice reduction algorithm is equivalent to derive a better-conditioned matrix $\tilde{\mathbf{H}}$ along with a unimodular matrix $\mathbf{T}$ from the original channel matrix $\mathbf{H}$ under a given criterion such that $\tilde{\mathbf{H}} = \mathbf{H}\mathbf{T}$ [1]. Linear LRAD is to combine the lattice reduction algorithm with the linear detection, such as ZF and MMSE. Consider ZF first, and the estimated signal $\hat{\mathbf{x}}$ can be written as

$$\hat{\mathbf{x}} = \tilde{\mathbf{H}}^\dagger\mathbf{y} = \tilde{\mathbf{H}}^\dagger\left((\mathbf{H}\mathbf{T})(\mathbf{T}^{-1}\mathbf{x}) + \mathbf{n}\right) = \mathbf{T}^{-1}\mathbf{x} + \tilde{\mathbf{H}}^\dagger\mathbf{n}. \tag{2}$$

Let $\hat{\mathbf{x}}_q$ be a version of $\hat{\mathbf{x}}$ quantized elementwise. From (2), it is clear that $\hat{\mathbf{x}}_q$ is an estimate of $\mathbf{T}^{-1}\mathbf{x}$, rather than of $\mathbf{x}$. Hence, the last step is to transform $\hat{\mathbf{x}}_q$ back into an estimate of $\mathbf{x}$, i.e., $\hat{\mathbf{x}}_{LR} = \mathbf{T}\cdot\hat{\mathbf{x}}_q$. (2) also applies to MMSE detection if the extended system model in [1] is considered. Simply substitute, for $\mathbf{H}$ and $\mathbf{y}$, the extended channel matrix and the extended received vector, respectively. The remaining operations are the same as in ZF.

3. ALL-SWAP LATTICE REDUCTION ALGORITHM

Since ASLR is a variant of the LLL algorithm, we first summarize the *LLL-reduced* lattice. Let the $n\times m$ matrix $\mathbf{H}$ be a set of

The work of N.C. Wang was partially supported by National Science Council, Taiwan. (TMS-094-2-A-002).

lattice basis vectors, with QR decomposition (QRD) $\mathbf{H}=\mathbf{QR}$. $\mathbf{H}$ is called *complex LLL-reduced* if the following two conditions are satisfied [4]:

(a) $\Re(|r_{i,j}/r_{i,i}|)\le 1/2$ and $\Im(|r_{i,j}/r_{i,i}|)\le 1/2,\ 1\le i<j\le m$, (3)

(b) $\delta-|r_{i-1,i}/r_{i-1,i-1}|^2\le |r_{i,i}|^2/|r_{i-1,i-1}|^2,\ 2\le i\le m$. (4)

δ is a constant chosen between 1/2 to 1. The process to make the basis set satisfy (3) is called *size reduction* (SR).

Table I describes the complex ASLR algorithm. In the following discussion, we refer to the lines in Table I. One significant difference between LLL and ASLR is that the pair of columns k and $k-1$ with all even (or odd) indices k could be swapped simultaneously (lines 10 and 13). For systolic arrays, all these column swaps within one iteration can be done in parallel. Additionally, unlike the LLL algorithm considered in the literature [1][4], size reduction process in ASLR applies to all the columns of $\mathbf{H}$ during one iteration (lines 3~8), and we called it "full size reduction (FSR)." The advantage of FSR over SR in our proposed systolic array will be shown in Section 4.

Two minor modifications of the original ASLR algorithm are made to accommodate the systolic array design. First, the Givens rotation (lines 17~21) is executed before the column swap (line 22). This is because the Givens rotation process can work in parallel with FSR, whereas the columns swap cannot. This point will be made clear in Section 4. Second, the QR decomposition $\mathbf{Q}^H\mathbf{H}=\mathbf{R}$ is considered as the input of the algorithm, instead of $\mathbf{H}=\mathbf{QR}$. From lines 19 and 21, the Givens rotation matrix applies to the same two rows of $\mathbf{Q}^H$ and $\mathbf{R}$, which simplifies the design of the systolic array. Additionally, after LLL, $\tilde{\mathbf{Q}}^H$ is ready for calculating $\tilde{\mathbf{H}}^{\dagger}$ in the linear detection step.

TABLE I ALL SWAP LATTICE REDUCTION ALGORITHM

INPUT $\mathbf{Q}^H,\mathbf{R}$; *OUTPUT* $\tilde{\mathbf{Q}}^H=\mathbf{Q}^H,\tilde{\mathbf{R}}=\mathbf{R},\ \mathbf{T}$

(1) Initialization $\mathbf{T}=\mathbf{I}_m$; *order*=EVEN
(2) While (any swap is possible in lines (10) or (13))
(3) for $j=m,\cdots,2$ **full size reduction**
(4) for $i=j$-1,$\cdots$,1
(5) $\mu_{i,j}=\lceil r_{i,j}/r_{i,i}\rfloor$
(6) $\mathbf{R}_{1:i,j}:=\mathbf{R}_{1:i,j}-\mu_{i,j}\mathbf{R}_{1:i,i}$; $\mathbf{T}_{1:m,j}:=\mathbf{T}_{1:m,j}-\mu_{i,j}\mathbf{T}_{1:m,i}$
(7) end
(8) end
(9) If *order*=EVEN
(10) Execute (#) for all even k between 2~m such that $\delta-|r_{k-1,k}/r_{k-1,k-1}|^2>|r_{k,k}|^2/|r_{k-1,k-1}|^2$
(11) *order*=ODD
(12) else
(13) Execute (#) for all odd k between 2~m such that $\delta-|r_{k-1,k}/r_{k-1,k-1}|^2>|r_{k,k}|^2/|r_{k-1,k-1}|^2$
(14) *order*=EVEN
(15) end
(16) end

(#)
(17) $\phi=\tan^{-1}(\Im(r_{k-1,k})/\Re(r_{k-1,k}))$
(18) $\mathbf{G}_1=\begin{bmatrix} e^{-j\phi} & 0\\ 0 & 1\end{bmatrix}$ **Givens Rotation**
(19) $\mathbf{R}_{k-1:k,k-1:m}:=\mathbf{G}_1\cdot\mathbf{R}_{k-1:k,k-1:m}$, $\mathbf{Q}^H{}_{k-1:k,1:n}:=\mathbf{G}_1\cdot\mathbf{Q}^H{}_{k-1:k,1:n}$
(20) $\theta=\tan^{-1}(r_{k,k}/r_{k-1,k})$, $\mathbf{G}_2=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}$
(21) $\mathbf{R}_{k-1:k,k-1:m}:=\mathbf{G}_2\cdot\mathbf{R}_{k-1:k,k-1:m}$, $\mathbf{Q}^H{}_{k-1:k,1:n}:=\mathbf{G}_2\cdot\mathbf{Q}^H{}_{k-1:k,1:n}$
(22) Swap columns k-1 and k in $\mathbf{R}$ and $\mathbf{T}$ **Column Swap**

4. SYSTOLIC ARRAY FOR ASLR ALGORITHM

4.1. Systolic Array for FSR-LLL

In the following, we assume a 4×4 MIMO system and illustrate the proposed systolic array in three parts: full size reduction, Givens rotation, and column swap. Prior to the ASLR, QRD of the channel matrix is needed. In this paper, we assume that the matrices $\mathbf{Q}^H$ and $\mathbf{R}$ are computed by the systolic array proposed in [6].

a) *Full size reduction*: The systolic array for the linear LRAD is shown in Fig. 1(a). Four different kinds of processing elements (PE) are used, which are diagonal cells, off-diagonal cells, vectoring cells, and rotation cells. The operations of these PEs are shown in Fig. 1(b)(c)(d). The dotted lines represent the logic control signals transmission between cells, and the solid lines represent the data transmission. To initialize the processing, each element of the matrix $\mathbf{R}$, $\mathbf{Q}^H$ and $\mathbf{T}$ (denoted as r, q and t in Fig. 1(b)(d), respectively) are stored in the PE at the corresponding position. For example, the off-diagonal elements $q_{i,j}$ $r_{i,j}$ and $t_{i,j}$ are stored in the cell O_{ij}.

Fig. 2 shows the overall processes of full size reduction during one iteration. Only diagonal and off-diagonal cells are needed at this stage. Suppose the cells execute all operations in *data mode* or *size reduction mode* at each time instant. At $T=0$, the external controller sends in the logic control signal "*data*" to cell D_{33} through cell D_{44}. At $T=1$, cell D_{33} enters *data mode* and spreads out the "*data*" signal to the neighboring three cells. Meanwhile, D_{33} sends out the data $(r_{33},t_{33})^{(*)}$ to cell O_{34}. The star (*) indicates that the data are sent out by a diagonal cell and it drives the off-diagonal cell to compute μ. As a result, at $T=2$ cell O_{34} computes μ, and sends it out to the two neighboring cells O_{24} and D_{44} for the updating operations at the next time instant. At $T=3$, cell O_{23} is driven into *size reduction mode* by the incoming data $(r_{22},t_{22})^{(*)}$. A crucial point here is that cell O_{23} also propagates the data $(r_{22},t_{22})^{(*)}$ to cell O_{24}, and thus starts the column operations between column 2 and 4 at $T=4$. Essentially, full size reduction is a series of column operations between column j and all the columns prior to j. In general, all column operations on column j in $m\times m$ MIMO system end at $T=2m+j-3$ in cell O_{mj}. The full size reduction stops at $T=3m-3$, when all updates on column m are done.

When using systolic array, the advantage of FSR over SR can be shown by the following example. Suppose no column swap is necessary after $\mathbf{H}$ is size-reduced. In ASLR, no further processing is needed after FSR. Hence, the systolic array takes a total of $3m-3$ cycles to end the all processes. However, with SR the process will end until columns 2 to m are sequentially size-reduced and it takes $\sum_{j=2}^{m}(2m+j-3)=2.5m^2-4.5m+2$ cycles to end the LLL algorithm. As m increases, the advantage of FSR over SR becomes significant in this case.

b) *Givens rotation*: In line 10 or 13 of Table I, if there exists any k such that $\delta-|r_{k-1,k}/r_{k-1,k-1}|^2>|r_{k,k}|^2/|r_{k-1,k-1}|^2$, then ASLR proceeds to the Givens-rotation step. To simplify this condition check in the systolic array, we use a variant of (4) for a reduced lattice, $1/2\le|r_{k,k}|^2/|r_{k-1,k-1}|^2$ [8]. Since the condition check now only relates two r elements in the neighboring diagonal cells, it can be checked in parallel with the FSR step. For example, in Fig. 2 at $T=1$, cell D_{33} sends data $r_{3,3}$ to cell D_{22} along with the "*data*" signal and cell D_{22} will compute $|r_{3,3}|^2/|r_{2,2}|^2$ at $T=2$. If $1/2>|r_{k,k}|^2/|r_{k-1,k-1}|^2$, then the logic control signal "*swap*" is set to "*true*", and thus drives the vectoring cell to work. The vectoring cells and rotation cells perform the Givens rotation to the same two rows of $\mathbf{R}$ and $\mathbf{Q}^H$. In order to make $\mathbf{R}$ still an upper

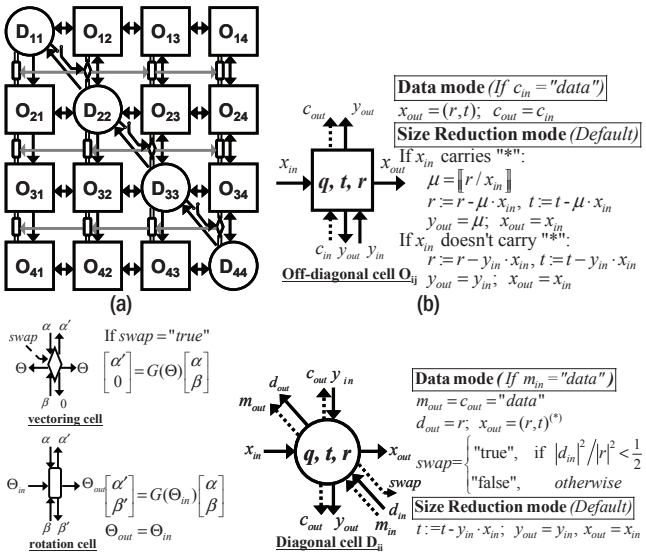


Fig. 1. (a) The systolic array for the linear LRAD of 4×4 MIMO system. (b)(c)(d) The operations of all processing elements.

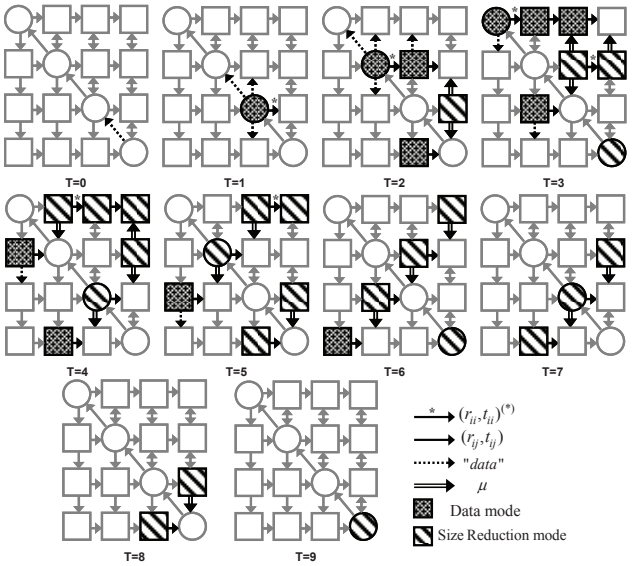


Fig. 2. Flow chart of the FSR operations in the systolic array.

triangular matrix after column swap, the vectoring cell annihilates the data $r_{k,k}$ by the Givens rotation matrix $G(\Theta)$ with the rotation angle $\Theta = (\phi, \theta)$ (lines 17 and 20 in Table I). The rotation cell simply rotates the input data with the angle given by the neighboring cell. Hence, the vectoring and rotation cells also work in a systolic way, with the rotation angle Θ propagating between cells. Note all diagonal cells could generate the "*swap*" signal during the FSR step. Therefore, there is a "*switch*", which is managed by the external controller, between each pair of the diagonal cell and the vectoring cell. If the current value of "*order*" is even (odd), then the "switch" between each cell $D_{k-1,k-1}$ with even (odd) index k and the vectoring cell is turned on by the external controller. Consequently, for every even (odd) index k, Givens rotation between rows $k-1$ and k could be executed if needed.

Additionally, a Givens rotation on rows k and $k-1$ can begin right after $r_{k-1,k}$ is updated during FSR without any interference to the remaining operations of FSR. This way, the time necessary to perform Givens rotations can be hidden by the FSR and this is the reason why we want the Givens rotation to occur prior to column swap in our design.

c) *Column swap*: If columns k and $k-1$ of $\mathbf{R}$ (and $\mathbf{T}$) should be swapped, the external controller will send command signals from the top cells of columns k and $k-1$ in order to force the swapping data. The command signals propagate vertically downward along these columns. More than one pair of columns could be swapped during one iteration, but all these pairs are swapped in parallel. Hence, the time spent on columns swap is the same as on swapping a single pair of columns. The external controller can send in the command signals after full size reduction and Givens rotation are ended. However, it is still possible that the column swap be partially overlapped in time with size reduction and Givens rotation.

Note that in our description we limit the applications of this systolic array only to $m\times m$ MIMO systems. For $m\times m$ MMSE-LRAD, although $\mathbf{Q}^H$ for the extended channel matrix is an $m\times 2m$ matrix, we can store each element of the two squared submatrices $\mathbf{Q}^H_{1:m,1:m}$ and $\mathbf{Q}^H_{1:m,(m+1):2m}$ in the PE at the corresponding position. Namely, $q_{i,j}$ and $q_{i,j+m}$ should be stored in the same PE, which still keeps the systolic array squared.

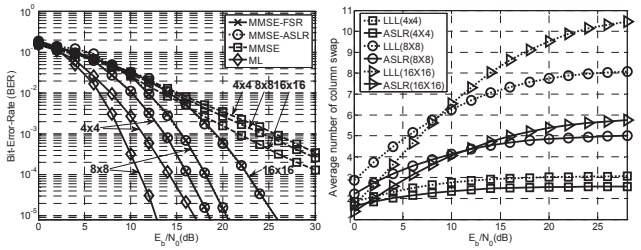
4.2. Comparison between LLL and ASLR

First, we compare the two algorithms in terms of bit-error-rate (BER) performance. In our simulation, 4-QAM is assumed for the transmitted symbols. Fig. 3(a) shows the BER results of MMSE-LRAD based on LLL and ASLR algorithm (denoted as MMSE-LLL and MMSE-ASLR, respectively). The two algorithms lead to almost the same results in all three MIMO systems. Hence, we can conclude that despite LLL and ASLR give different lattice reduced matrices, the linear LRAD based on these two algorithms have similar BER performance.

Next, we compare the efficiency of the systolic array for both algorithms in terms of the average number of column swaps in the overall process. Less column swapping implies less iterations, and thus less cycles in the systolic array. Fig. 3(b) shows the average number of column swaps in LLL and ASLR algorithms of the MMSE-LRAD. For ASLR, we count all the columns swaps during one iteration as only one swap since they are executed in parallel. As the number of antennas grows, the advantage of ASLR becomes significant. In 4×4 MIMO, the difference between two algorithms is no more than 0.5. However, in a 16×16 MIMO system, MMSE-ASLR has less than 62% of the column swaps comparing to MMSE-LLL when E_b / N_0 is above 10dB. Based on BER performance and time-efficiency comparisons, ASLR should be a better algorithm to be applied on our systolic array, especially with a large number of antennas.

5. SYSTOLIC ARRAY FOR LINEAR DETECTION

The linear-detection processes described in Section 2 can also be operated on the systolic array in Fig. 1(a). Consider the ZF detection first. To execute $\hat{\mathbf{x}} = \tilde{\mathbf{H}}^{\dagger}\mathbf{y} = \tilde{\mathbf{R}}^{-1}\tilde{\mathbf{Q}}^H\mathbf{y}$ in the systolic array, we separate it into two matrix--vector multiplications $\mathbf{v} = \tilde{\mathbf{Q}}^H\mathbf{y}$



(a) (b)

Fig. 3. (a) BER performance of LLL- and ASLR- based MMSE-LRAD. (b) The average number of column swaps in LLL and ASLR algorithms when performing MMSE LRAD.

and then $\hat{\mathbf{x}} = \tilde{\mathbf{R}}^{-1}\mathbf{v}$. Since $\tilde{\mathbf{Q}}^H$ stays in the systolic arrays after the lattice reduction ends, the received signal vector $\mathbf{y}$ can be fed to the systolic arrays from the top in a skewed manner as shown in Fig. 4(a). The vector $\tilde{\mathbf{Q}}^H\mathbf{y}$ is pumped out from the rightmost column of the array. The operations of the cells are shown in Fig. 4(d). Every cell performs the multiply-and-add operation. If MMSE is chosen, the input vector should be changed to an $2m\times 1$ extended received vector $\bar{\mathbf{y}}$. Let $\bar{\mathbf{y}} = [\mathbf{y}_1^T \ \mathbf{y}_2^T]^T$ and $\tilde{\mathbf{Q}}^H = [\mathbf{Q}_1 \ \mathbf{Q}_2]$, where $\mathbf{y}_1$, $\mathbf{y}_2$ are $m\times 1$ vectors and $\mathbf{Q}_1$, $\mathbf{Q}_2$ are $m\times m$ matrices. As mentioned in Section 4.1, the elements of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are stored in the same PEs. To compute $\mathbf{v}=\tilde{\mathbf{Q}}^H\bar{\mathbf{y}}$ using the systolic array, first we let $\mathbf{y}_1$ enter the array from the top and multiply it by $\mathbf{Q}_1$, which is the same as shown in Fig. 4(a). Then $\mathbf{y}_2$ enters the array right after $\mathbf{y}_1$ also in a skewed manner, and is multiplied by $\mathbf{Q}_2$. Hence, for MMSE we need an extra operation at the output of the array, which is $\mathbf{v} = \mathbf{Q}_1\mathbf{y}_1 + \mathbf{Q}_2\mathbf{y}_2$. For the remaining operations in the systolic array, there is no difference between ZF and MMSE detection.

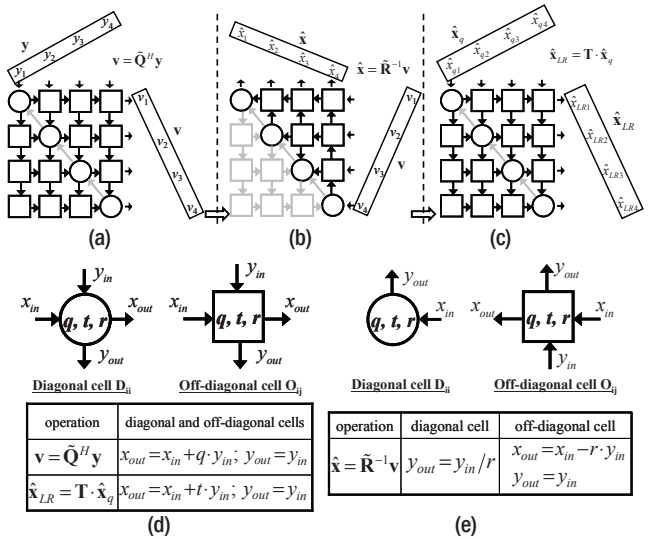
The second stage consists of computing $\hat{\mathbf{x}} = \tilde{\mathbf{R}}^{-1}\mathbf{v}$. Instead of computing $\tilde{\mathbf{R}}^{-1}$ directly, the following recurrence equation [7] is considered for the systolic design

$$\hat{x}_j = \frac{1}{\tilde{r}_{jj}}\left(v_j - \sum_{i=j+1}^{m} \tilde{r}_{ji}\hat{x}_i\right), \quad j : \text{from } m \text{ to } 1. \tag{5}$$

According to (5), it is clear that $\tilde{\mathbf{R}}^{-1}\mathbf{v}$ can be computed directly from the components of $\tilde{\mathbf{R}}$. As shown in Fig. 4(b), the vector $\mathbf{v}$ enters the array from the right, and $\hat{\mathbf{x}} = \tilde{\mathbf{R}}^{-1}\mathbf{v}$ is computed by the upper-triangular array with cell operations shown in Fig. 4(e). The output vector $\hat{\mathbf{x}}$ is then quantized elementwise outside the systolic array. The final step consists of multiplying the quantized vector $\hat{\mathbf{x}}_q$ by the unimodular matrix $\mathbf{T}$, which is very similar to the operations of $\mathbf{v} = \tilde{\mathbf{Q}}^H\mathbf{y}$. Hence, the data flow in Fig. 4(c) is the same as Fig. 4(a). The cell operations for $\hat{\mathbf{x}}_{LR} = \mathbf{T}\cdot\hat{\mathbf{x}}_q$ are shown in Fig. 4(d), and $\hat{\mathbf{x}}_{LR}$ is the final result of the linear LRAD.

In sum, there are one addition, one multiplication, and one division in each diagonal cell, and one addition and one multiplication in each off-diagonal cell for linear detection, be it ZF or MMSE. These operations are also contained in each cell at the LLL lattice reduction stage. Hence, there can be no extra hardware cost (adders or multipliers) in each cell for linear detection. Only extra control logic to the array is needed in order to have each PE work correctly in different modes.

6. CONCLUSION



operation	diagonal and off-diagonal cells
$\mathbf{v} = \tilde{\mathbf{Q}}^H\mathbf{y}$	$x_{out} = x_{in} + q\cdot y_{in};\ y_{out} = y_{in}$
$\hat{\mathbf{x}}_{LR} = \mathbf{T}\cdot\hat{\mathbf{x}}_q$	$x_{out} = x_{in} + t\cdot y_{in};\ y_{out} = y_{in}$

(d)

operation	diagonal cell	off-diagonal cell
$\hat{\mathbf{x}} = \tilde{\mathbf{R}}^{-1}\mathbf{v}$	$y_{out} = y_{in}/r$	$x_{out} = x_{in} - r\cdot y_{in}$ $y_{out} = y_{in}$

(e)

Fig. 4. The linear detection operations for (a) $\mathbf{v} = \tilde{\mathbf{Q}}^H\mathbf{y}$ (b) $\hat{\mathbf{x}} = \tilde{\mathbf{R}}^{-1}\mathbf{v}$ (c) $\hat{\mathbf{x}}_{LR} = \mathbf{T}\cdot\hat{\mathbf{x}}_q$ in the systolic array. (d)(e) The operations of the diagonal and off-diagonal cells in the systolic array at different stages.

In this paper, we proposed a systolic array to perform lattice-reduction-aided linear detection for MIMO receivers. The design is based on all-swap complex lattice-reduction algorithm, which generalizes the one originally proposed in [5] for real lattices. Compared to LLL algorithm, ASLR operates on a whole matrix, rather than on its single columns, during the column-swap and Givens-rotation steps. The linear detection can also be implemented on the same systolic array for the ASLR. Due to the high-throughput property of systolic arrays, our design appears very promising for high-data-rate systems, such as in a MIMO-OFDM system.

7. REFERENCES

[1] D. Wübben, R. Böhnke, V. Kühn and K.-D. Kammeyer, "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice reduction", *Proc. ICC*, pp. 798-802, 2004.

[2] H. Yao and G.W. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems", *Proc. GLOBECOM*, pp. 424-428, 2002.

[3] A. K. Lenstra, H.W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Annalen*, vol. 261, pp. 515-534, 1982.

[4] Y. H. Gan and H. W. Mow, "Complex lattice reduction algorithms for low-complexity MIMO detection," *Proc. GLOBECOM*, pp. 2953-2957, 2005.

[5] C. Heckler and L. Thiele, "A parallel lattice basis reduction for mesh-connected processor arrays and parallel complexity," *Proc. IPDPS*, pp.400-407, 1993.

[6] A. El-Amawy amd K.R. Dharmarajan, "Parallel VLSI algorithm for stable inversion of dense matrices," in *Proc. Computers and Digital Techniques*, vol. 136, pp. 575-580, 1989.

[7] F. Lorenzelli, P.C. Hansen, T.F. Chan, and K. Yao, "A systolic implementation of the Chan/Foster RRQR algorithm," *IEEE Trans. on Signal Processing*, pp. 2205-2208, 1994.

[8] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," Combinatorica, vol. 6, no. 1, pp. 1-13, 1986.