2010 International Conference on Communications and Mobile Computing

An Improved Implementation for the Recursive V-BLAST Algorithms to Save Memories and Permutation Operations

Hufei Zhu (Member, IEEE), Bin Li (Member, IEEE)
Communication Technology Research Dept.
Huawei Technologies Co., Ltd.
Shenzhen 518129, P.R. China
Email: zhuhufei, binli@huawei.com

Wen Chen (Member, IEEE)
Dept. of Electronic Engineering
Shanghai Jiao Tong Univ.
Shanghai 200240, P.R. China
Email: wenchen@sjtu.edu.cn

***Abstract*—In this paper, we propose an improved implementation for the recursive V-BLAST algorithms, which saves memories and permutation operations, as the existing V-BLAST algorithm with memory saving. When the same number of transmit and receive antennas are assumed, the proposed implementation requires only half memories and saves more than twice permutation operations, with respect to the existing V-BLAST algorithm with memory saving. Moreover, as the number of receive antennas increases relative to the number of transmit antennas, the ratio between the memories required by the proposed implementation and those required by the existing V-BLAST algorithm with memory saving becomes even smaller. On the other hand, when the same number of transmit and receive antennas are assumed, the speedups of the proposed implementation over the existing V-BLAST algorithm with memory saving in the number of multiplications, additions and flops (floating-point operations) are 2.28, 1.86 and 2.18, respectively.**

***Keywords*-V-BLAST; the recursive algorithms; memory saving; permutation operations;**

I. Introduction

Bell Laboratories Layered Space-Time architecture (BLAST), including the most practical version- vertical BLAST (V-BLAST) can achieve very high data rate and spectral efficiency in rich multi-path environments through exploiting the extra spatial dimension [1]. However, the required computational complexity is quite high. Recently several fast algorithms have been proposed to reduce the computational complexity of V-BLAST [2]–[7]. The square-root algorithm in [2] can reduce the computational complexity of the conventional V-BLAST by $0.7M$, where M is the number of transmit antennas. The speedups of the fast recursive algorithm in [3] over the algorithm in [2] in the number of multiplications and additions are 1.54 and 1.89 respectively [4]. Improvements for different parts of the fast recursive algorithm [3] were proposed in [5] and [4], respectively, which were then incorporated in [6] to give the "fastest known algorithm". The contributions of [6] also include a further improvement for the "fastest known algorithm". Base on the recursive algorithm for V-BLAST in [6], recently an improved recursive algorithm for V-BLAST was proposed in [7].

On the other hand, in [2]–[7], only the computational complexities of the V-BLAST algorithms were considered, while the required memories and permutation operations were not counted in [8], which are also important in practical implementations. Thus a recursive V-BLAST algorithm with memory saving was proposed in [8], which can save memories and permutation operations, with respect to the V-BLAST algorithm in [6].

In this paper, we propose an improved implementation for the recursive V-BLAST algorithms in [6], [7], which save memories and permutation operations, and do not increase the computational complexity. As a comparison, the recursive V-BLAST algorithm with memory saving in [8] saves less memories and permutation operations, and increases much computational complexity.

This paper is organized as follows. In Section 2, the system model for V-BLAST is overviewed. In Section 3, we describe the recursive V-BLAST algorithms in [6], [7] and the V-BLAST algorithm with memory saving in [8]. Then we propose the improved implementation for the recursive V-BLAST algorithms in [6], [7] in Section 4. Finally, we make conclusions in Section 5.

In the following sections, $(\bullet)^T$, $(\bullet)^*$, and $(\bullet)^H$ denote matrix transposition, matrix conjugate, and matrix conjugate transposition, respectively. I_M is the identity matrix with size M.

II. System Model

The considered V-BLAST system consists of M transmit antennas and $N(\geq M)$ receive antennas in a rich-scattering and flat-fading wireless channel. At the transmitter, the data stream is de-multiplexed into M sub-streams, and each sub-stream is encoded and fed to its respective transmit antenna. Each receive antenna receives the signals from all M transmit antennas. Let $\mathbf{s} = [s_1, s_2, \cdots, s_M]^T$ denote the vector of transmit symbols from M antennas, and assume $E(\mathbf{s}\mathbf{s}^H) = \sigma_{\mathbf{s}}^2 \mathrm{I}_M$. Then the received signal is given by

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

where $\mathbf{n}$ is the $N \times 1$ complex Gaussian noise vector with zero mean and covariance $\sigma_n^2 \mathbf{I}_N$, and $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_M]$ is the $N \times M$ complex channel matrix with statistically independent entries. The vector $\mathbf{h}_m$ represents the m^{th} column of $\mathbf{H}$.

The linear MMSE detection of $\mathbf{s}$ is

$$\mathbf{y} = \left(\mathbf{H}^H \mathbf{H} + \alpha \mathbf{I}_M\right)^{-1} \mathbf{H}^H \mathbf{x}, \tag{2}$$

where $\alpha = \sigma_n^2/\sigma_s^2$. Let $\mathbf{R} = \mathbf{H}^H \cdot \mathbf{H} + \alpha \mathbf{I}_M$. Then

$$\mathbf{Q} = \mathbf{R}^{-1} = \left(\mathbf{H}^H \mathbf{H} + \alpha \mathbf{I}_M\right)^{-1} \tag{3}$$

is the covariance matrix for the detection error $\mathbf{e} = \mathbf{s} - \mathbf{y}$ [2], [3], i.e., $E\{\mathbf{e}\mathbf{e}^H\} = \sigma_n^2 \mathbf{Q}$. So the best estimate, i.e., the component of $\mathbf{y}$ with the highest SNR, is corresponding to the minimal diagonal element in $\mathbf{Q}$.

The conventional V-BLAST scheme detects M components of $\mathbf{s}$ by M iterations with the optimal ordering. In each iteration, the component with the highest post detection SNR among all the undetected components is detected by an MMSE filter and then its effect is substracted from the received signal vector [1]–[3].

III. The Recursive V-BLAST Algorithms in [6], [7] and the V-BLAST Algorithm with Memory Saving in [8]

Represent the first m columns of $\mathbf{H}$ as $\mathbf{H}_m = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_m]$, where $\mathbf{h}_m$ is the m^{th} column of $\mathbf{H}$. Then write [6]

$$\begin{cases} \mathbf{R}_m = \mathbf{H}_m^H \mathbf{H}_m + \alpha \mathbf{I}_m = \begin{bmatrix} \mathbf{R}_{m-1} & \mathbf{v}_{m-1} \\ \mathbf{v}_{m-1}^H & \upsilon_m \end{bmatrix}, & \text{(4a)} \\ \mathbf{Q}_m = \mathbf{R}_m^{-1} = \begin{bmatrix} \mathbf{T}_{m-1} & \mathbf{w}_{m-1} \\ \mathbf{w}_{m-1}^H & \omega_m \end{bmatrix}, & \text{(4b)} \end{cases}$$

where $\mathbf{R}_{m-1}$ and $\mathbf{T}_{m-1}$ are the $(m-1) \times (m-1)$ leading principal sub-matrices of $\mathbf{R}$ and $\mathbf{Q}_m$, respectively.

In the *initialization* phase of the recursive V-BLAST algorithm [6], the initial $\mathbf{z}_M$ is computed by

$$\mathbf{z}_M = \mathbf{H}^H \mathbf{x}. \tag{5}$$

To obtain the initial $\mathbf{Q}_M$ from $\mathbf{Q}_1$, $\mathbf{Q}_m$ is computed from $\mathbf{Q}_{m-1}$ recursively (for $m = 2, 3, ..., M$) by

$$\begin{cases} \mathbf{T}_{m-1} = \mathbf{Q}_{m-1} + \dfrac{\mathbf{Q}_{m-1}\mathbf{v}_{m-1}\mathbf{v}_{m-1}^H \mathbf{Q}_{m-1}}{\upsilon_m - \mathbf{v}_{m-1}^H \mathbf{Q}_{m-1} \mathbf{v}_{m-1}}, & \text{(6a)} \\ \mathbf{Q}_m = \\ \begin{bmatrix} \mathbf{T}_{m-1} & -\upsilon_m^{-1}\mathbf{T}_{m-1}\mathbf{v}_{m-1} \\ -\upsilon_m^{-1}\mathbf{v}_{m-1}^H \mathbf{T}_{m-1} & \upsilon_m^{-1} + \upsilon_m^{-2}\mathbf{v}_{m-1}^H \mathbf{T}_{m-1}\mathbf{v}_{m-1} \end{bmatrix}. & \text{(6b)} \end{cases}$$

In the *recursion* phase of the V-BLAST algorithm [6], M entries of $\mathbf{s}$ are detected recursively with the optimal ordering. In each recursion, estimate the entry s_{p_m} in $\mathbf{s}$, i.e. the symbol with the highest signal-to-noise ratio (SNR) among all the undetected symbols, by

$$y_{p_m} = \mathbf{q}_{\hat{m}}^H \mathbf{z}_m, \tag{7}$$

where $\mathbf{q}_{\hat{m}}$ is the last m^{th} column of the permuted $\mathbf{Q}_{\hat{m}}$. Notice that in some variables for the *recursion* phase, we rewrite the subscript m into $\hat{m}$, to distinguish these variables from those in the *initialization* phase, e.g., usually $\mathbf{Q}_{\hat{m}} \neq \mathbf{Q}_m$. From y_{p_m}, obtain the hard decision $\hat{s}_{p_m} = Q[y_{p_m}]$, where $Q[\cdot]$ indicates the quantizing procedure according to the constellation in use. Then cancel the effect of s_{p_m} in $\mathbf{z}_m$ by

$$\mathbf{z}_{m-1} = \mathbf{z}_m^{[-1]} - \hat{s}_{p_m} \mathbf{v}_{\hat{m}-1}, \tag{8}$$

where $\mathbf{z}_m^{[-1]}$ is obtained by removing the last entry in $\mathbf{z}_m$, and $\mathbf{v}_{\hat{m}-1}$ is in the last column of the permuted $\mathbf{R}_{\hat{m}}$, as shown in (4a). Correspondingly deflate $\mathbf{Q}_{\hat{m}}$ by

$$\mathbf{Q}_{\hat{m}-1} = \mathbf{T}_{\hat{m}-1} - \omega_{\hat{m}}^{-1} \mathbf{w}_{\hat{m}-1} \mathbf{w}_{\hat{m}-1}^H, \tag{9}$$

where $\mathbf{T}_{\hat{m}-1}$, $\omega_{\hat{m}}$ and $\mathbf{w}_{\hat{m}-1}$ are in $\mathbf{Q}_{\hat{m}}$, as show in (4b).

The recursive V-BLAST algorithm in [6] is summarized in Table I, where $q_{i,k}^{\hat{m}}$ and $r_{i,k}^{\hat{m}}$ denote the entries in the i^{th} row and k^{th} column of $\mathbf{Q}_{\hat{m}}$ and $\mathbf{R}_{\hat{m}}$, respectively.

In [7], a fast algorithm to compute the inverse matrix $\mathbf{Q} = \mathbf{R}^{-1}$ is applied to improve the corresponding step of the recursive V-BLAST algorithm in [6]. The algorithm to compute the inverse matrix is

$$\begin{cases} \omega_m = \left(\upsilon_m - \mathbf{v}_{m-1}^H \mathbf{Q}_{m-1} \mathbf{v}_{m-1}\right)^{-1}, & \text{(10a)} \\ \mathbf{w}_{m-1} = -\omega_m \mathbf{Q}_{m-1} \mathbf{v}_{m-1}, & \text{(10b)} \\ \mathbf{T}_{m-1} = \mathbf{Q}_{m-1} + \omega_m^{-1} \mathbf{w}_{m-1} \mathbf{w}_{m-1}^H. & \text{(10c)} \end{cases}$$

Then in the recursive V-BLAST algorithm in [6], we can use (10) and (4b) instead of (6) to compute the initial $\mathbf{Q}_M$ recursively.

On the other hand, in [8] the V-BLAST algorithm with memory saving was proposed, where $\mathbf{R}$ is not calculated at all. Thus the V-BLAST algorithm with memory saving in [8] can save the memories for storing $\mathbf{R}$ and the associated permutation operations with $\mathbf{R}$. At the same time, it requires much more computational complexity than the V-BLAST algorithm in Table I. The V-BLAST algorithm in Table I only requires $\frac{1}{2}M^2N + \frac{2}{3}M^3 + O(MN + M^2)$ multiplications and additions [7], while the V-BLAST algorithm with memory saving in [8] requires $\frac{5}{2}M^2N + \frac{1}{6}M^3 + O(MN + M^2)$ multiplications and $2M^2N + \frac{1}{6}M^3 + O(MN + M^2)$ additions [8]. When $M = N$, the V-BLAST algorithm with memory saving in [8] requires $(\frac{5}{2} + \frac{1}{6})/(\frac{1}{2} + \frac{2}{3}) = 2.28$ times of multiplications and $(2 + \frac{1}{6})/(\frac{1}{2} + \frac{2}{3}) = 1.86$ times of additions, with respect to the V-BLAST algorithm in Table I. Then it can be seen that the floating-point operations (flops) [3] required by the V-BLAST algorithm with memory saving in [8] are $\left((\frac{5}{2} + \frac{1}{6}) \times 6 + (2 + \frac{1}{6}) \times 2\right) / \left((\frac{1}{2} + \frac{2}{3}) \times 8\right) =$ 2.18 times those required by the V-BLAST algorithm in Table I. Though the V-BLAST algorithm with memory saving in [8] requires much more computational complexity, it is still beneficial in some cases since it saves memories and permutation operations [8].

Table I
THE RECURSIVE ALGORITHM FOR V-BLAST IN [6]

Initialization:	Compute $\mathbf{z}_M$ by (5). Compute the initial $\mathbf{R} = \mathbf{R}_M = \mathbf{R}_{\hat{M}}$. Compute (6) recursively to get the initial $\mathbf{Q} = \mathbf{Q}_M = \mathbf{Q}_{\hat{M}}$.
Recursion:	Set $\mathbf{p} = [1, 2, \cdots, M]^T$. For $m = M, M-1, \cdots, 2$: (a) Find $l_m = \arg\min_{i=1}^{m}(q_{i,i}^{\hat{m}})$. Exchange rows and columns l_m and m in $\mathbf{Q}_{\hat{m}}$ and $\mathbf{R}_{\hat{m}}$. Exchange entries l_m and m in $\mathbf{p}$ and $\mathbf{z}_m$. (b) Use (7) to compute y_{p_m} (i.e. the estimate of the $p_m{}^{th}$ signal s_{p_m}). Then quantize y_{p_m} to get $\hat{s}_{p_m} = Q[y_{p_m}]$. (c) Cancel the effect of s_{p_m} in $\mathbf{z}_m$ by (8), to obtain $\mathbf{z}_{m-1}$. (d) Deflate $\mathbf{Q}_{\hat{m}}$ to obtain $\mathbf{Q}_{\hat{m}-1}$ according to (9). Remove the last column and row of $\mathbf{R}_{\hat{m}}$ to obtain $\mathbf{R}_{\hat{m}-1}$.
Solution:	When $m = 1$, execute the above-described step (c) to obtain $\hat{s}_{p_1}$. The hard decision of the symbol s_{p_m} is $\hat{s}_{p_m}$, $m = 1, 2, \cdots, M$.

Table II
TO AVOID PERMUTATION OPERATIONS IN THE V-BLAST ALGORITHM [6]

Recursion:	(a') Find $l_m = \arg\min_{i=1}^{m}(q_{p_i,p_i})$. Exchange entries l_m and m in $\mathbf{p}$. (b') Follow step (c). In (7), $\mathbf{q}_{\hat{m}} = [q_{p_1,p_m}, q_{p_2,p_m}, \cdots, q_{p_m,p_m}]^T$. (c') Follow step (d). In (8), $\mathbf{z}_m^{[-1]}$ is obtained by removing the $l_m{}^{th}$ entry in $\mathbf{z}_m$, while $\mathbf{v}_{\hat{m}-1} = [r_{p_1,p_m}, r_{p_2,p_m}, \cdots, r_{p_{(m-1)},p_m}]^T$. (d') In $\mathbf{Q}_M$, let $q_{p_i,p_k} = q_{p_i,p_k} - q_{p_m,p_m}^{-1} q_{p_i,p_m} q_{p_k,p_m}^*$, where $1 \le i, k \le m-1$ and $p_i \le p_k$. Release the memories that store the entries in the $p_m{}^{th}$ row or column of $\mathbf{R}_M$ and $\mathbf{Q}_M$.

IV. THE IMPROVED IMPLEMENTATION FOR THE RECURSIVE V-BLAST ALGORITHMS IN [6], [7] TO SAVE MEMORIES AND PERMUTATION OPERATIONS

In this section, we propose the improved implementation for the Recursive V-BLAST Algorithms in [6], [7]. The improved implementation does not increase the computational complexity. At the same time, it saves more memories and permutation operations than the V-BLAST algorithm with memory saving in [8]. In the following, we introduce the improved implementation for the recursive V-BLAST algorithm in Table I.

Firstly, in Table I, $\mathbf{H}$ is not required when computing $\mathbf{Q}_M$, since $\mathbf{Q}_M$ is computed from $\mathbf{R}_M$, while $\mathbf{H}$ is not required in the *recursion* phase, either. Thus after $\mathbf{R}_M$ is computed in the *initialization* phase, we can reuse the memories for $\mathbf{H}_{N\times M}$ to store $\mathbf{Q}_M$. Contrarily, the algorithm with memory saving in [8] can not reuse the memories for $\mathbf{H}$, since it needs $\mathbf{H}$ to compute $\mathbf{Q}_M$, and $\mathbf{H}$ is also required in the *recursion* phase [6, Table IV]. Notice that in the *recursion* phase, the algorithm with memory saving in [8] utilizes the columns of $\mathbf{H}$ for interference cancellation, while the recursive V-BLAST algorithm in Table I utilizes the entries of $\mathbf{R}_M$ for interference cancellation.

Secondly, in the improved implementation proposed by us, we only store the upper (or lower) triangular part of the Hermitian $\mathbf{R}_M$ and $\mathbf{Q}_M$, since the entries in the not-stored triangular part can be obtained by $r_{i,k} = r_{k,i}^*$ and $q_{i,k} = q_{k,i}^*$, where $r_{i,k}$ and $q_{i,k}$ are in the i^{th} row and k^{th} column of $\mathbf{R}_M$ and $\mathbf{Q}_M$, respectively. The algorithm with memory saving in [8] requires $N\times M + M\times M$ memories to store $\mathbf{H}_{N\times M}$ and $\mathbf{Q}_M$, while the proposed implementation only requires $2 \times \frac{(M+1)\times M}{2} = M\times M + M$ memories to store the triangular part of the Hermitian $\mathbf{R}_M$ and $\mathbf{Q}_M$. The ratio between the memories required by the algorithm with memory saving in [8] and those required by the proposed implementation is $(N\times M + M\times M)/(M\times M + M) \approx \frac{N}{M} + 1$. Then it can be seen that the ratio is nearly 2 when $N = M$, and increases as N increases relative to M.

Lastly, the steps of the *recursion* phase in Table I can be modified into those in Table II, to avoid the permutation operations with $\mathbf{z}_m$, $\mathbf{R}_{\hat{m}}$ and $\mathbf{Q}_{\hat{m}}$ in step (a). In Table II, we obtain the entries in the permuted $\mathbf{R}_{\hat{m}}$ and $\mathbf{Q}_{\hat{m}}$ from $\mathbf{R}_M$ and $\mathbf{Q}_M$, by $r_{i,k}^{\hat{m}} = r_{p_i,p_k}$ and $q_{i,k}^{\hat{m}} = q_{p_i,p_k}$. Moreover, in step (d'), some stored entries of $\mathbf{Q}_M$, i.e. those in $\mathbf{T}_{\hat{m}-1}$, are updated to the corresponding entries in $\mathbf{Q}_{\hat{m}-1}$ by (9). The algorithm with memory saving in [8] only saves the permutation operations with $\mathbf{R}_{\hat{m}}$, while the proposed implementation saves the permutation operations with $\mathbf{z}_m$, $\mathbf{R}_{\hat{m}}$ and $\mathbf{Q}_{\hat{m}}$. So it can be seen that the proposed implementation saves more than twice permutation operations, with respect to the algorithm with memory saving in [8].

It can be easily seen that the improved implementation for the recursive V-BLAST Algorithms in [6], [7] does not increase the required computational complexity. Then the improved implementation still requires $\frac{1}{2}M^2N + \frac{2}{3}M^3 + O(MN + M^2)$ multiplications and additions [6], [7]. So when $M = N$, the speedups of the proposed implementation over the V-BLAST algorithm with memory saving [8] in the number of multiplications, additions and flops are 2.28, 1.86 and 2.18, respectively.

V. Conclusion

In this paper, we propose the improved implementation for the recursive V-BLAST Algorithms in [6], [7], which saves memories and permutation operations, as the V-BLAST algorithm with memory saving in [8]. When $M = N$, the proposed implementation requires only half memories and saves more than twice permutation operations, with respect to the V-BLAST algorithm with memory saving in [8]. Moreover, as N increases relative to M, the ratio between the memories required by the proposed implementation and those required by the V-BLAST algorithm with memory saving in [8] becomes even smaller. On the other hand, when $M = N$, the speedups of the proposed implementation over the V-BLAST algorithm with memory saving [8] in the number of multiplications, additions and flops are 2.28, 1.86 and 2.18, respectively.

Acknowledgment

This work is supported by Major National S&T Program 2009ZX03003-002 of China.

References

[1] P. W. Wolniansky, G. J. Foschini, G. D. Golden and R. A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel", *Proc. URSI International Symposium on Signals, Systems, and Electronics (ISSSE98)*, pp. 295-300, 1998.

[2] B. Hassibi, "An efficient square-root algorithm for BLAST", *Proc. of IEEE International Conf. Acoustics, Speech, and Signal Processing, (ICASSP '00)*, pp. 737-740, June 2000.

[3] J. Benesty, Y. Huang and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system", *IEEE Trans. on Signal Processing*, pp. 1722-1730, July 2003.

[4] H. Zhu, Z. Lei, F.P.S. Chin, "An improved recursive algorithm for BLAST", *Signal Process.*, vol. 87, no. 6, pp. 1408-1411, Jun. 2007.

[5] L. Szczecinski and D. Massicotte, "Low complexity adaptation of MIMO MMSE receivers, Implementation aspects", *Proc. Global Commun. Conf. (Globecom05)*, St. Louis, MO, USA, Nov. 28 - Dec. 2, 2005.

[6] Y. Shang and X. G. Xia, "An Improved Fast Recursive Algorithm for V-BLAST With Optimal Ordered Detections", *IEEE International Conference on Communications 2008 (ICC 2008)*, May 2008, pp. 756 - 760.

[7] H. Zhu, W. Chen and F. She, "Improved Fast Recursive Algorithms for V-BLAST and G-STBC with Novel Efficient Matrix Inversion", IEEE International Conference on Communications (ICC), 2009, Dresden, Germany.

[8] Y. Shang and X. G. Xia, "On Fast Recursive Algorithms For V-BLAST With Optimal Ordered SIC Detection", *IEEE Transactions on Wireless Communications*, vol. 8, pp.2860-2865, June 2009.