VLSI architecture of a *K*-best detector for MIMO–OFDM wireless communication systems

Jian Haifang(鉴海防)[1, 2, †] and Shi Yin(石寅)[1]

(1 Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China)

(2 Graduate University of the Chinese Academy of Sciences, Beijing 100049, China)

Abstract: The *K*-best detector is considered as a promising technique in the MIMO–OFDM detection because of its good performance and low complexity. In this paper, a new *K*-best VLSI architecture is presented. In the proposed architecture, the metric computation units (MCUs) expand each surviving path only to its partial branches, based on the novel expansion scheme, which can predetermine the branches' ascending order by their local distances. Then a distributed sorter sorts out the new *K* surviving paths from the expanded branches in pipelines. Compared to the conventional *K*-best scheme, the proposed architecture can approximately reduce fundamental operations by 50% and 75% for the 16-QAM and the 64-QAM cases, respectively, and, consequently, lower the demand on the hardware resource significantly. Simulation results prove that the proposed architecture can achieve a performance very similar to conventional *K*-best detectors. Hence, it is an efficient solution to the *K*-best detector's VLSI implementation for high-throughput MIMO–OFDM systems.

Key words: MIMO–OFDM; VBLAST; *K*-best detector; VLSI

DOI: 10.1088/1674-4926/30/7/075004 **EEACC:** 1265B

1. Introduction

Nowadays, the need for high-throughput wireless broadband communication systems is growing. MIMO–OFDM combines multiple-input multiple-output (MIMO) and orthogonal frequency division multiplexing (OFDM) technologies. As a promising approach to realize high data-rate wireless communications with larger spectral efficiency, MIMO–OFDM has been adopted in many standards, such as IEEE802.11n[1]. But a challenge for the MIMO–OFDM implementation is to design a low-complexity detector suitable for the efficient very large scale integration (VLSI) realization. Maximum likelihood detection (MLD) is optimal in theory, but it is not feasible in practical systems, because the computational complexity increases exponentially with the transmission rate[2]. *K*-best algorithms can achieve near-ML performance but greatly reduces the complexity compared to the MLD, so they are especially attractive for VLSI implementation and have been extensively studied recently[3–6].

In this paper, we propose a new *K*-best VLSI architecture for high-throughput MIMO–OFDM systems, which use a large number of antennas with a high-order quadrature amplitude modulation (QAM) scheme. Using novel expansion and sorting methods, the proposed architecture can efficiently sort out the surviving paths with a lower demand on hardware resources.

2. System model

We consider a MIMO–OFDM system with $N_{\rm tx}$ transmission and $N_{\rm rx}$ reception antennas. The information bits are mapped into M-QAM symbols, where M is the constellation order. A vertical Bell laboratories layered space-time (V-BLAST) scheme is applied to transmit independent data streams simultaneously. On each subcarrier, the complex baseband equivalent model can be expressed as

$$\boldsymbol{Y} = \boldsymbol{HS} + \boldsymbol{V}, \tag{1}$$

where $\boldsymbol{S}$ is the $N_{\rm tx}$-dimensional transmit data symbol vector, $\boldsymbol{H}$ is the $N_{\rm rx}N_{\rm tx}$ channel matrix, and $\boldsymbol{V}$ is the $N_{\rm rx}$-dimensional complex AWGN vector with zero mean and a variance of $\sigma^2 = N_0/2$ ($N_0/2$ is the background noise power spectral density)[4]. It is assumed that the channel matrix $\boldsymbol{H}$ is known by the receiver and remains constant throughout the data symbol duration.

3. *K*-best detection scheme

3.1. Conventional *K*-best detection algorithm

The optimal MLD algorithm can be expressed as[3]

$$\hat{\boldsymbol{s}} = \arg\min_{s\in\Omega^{N_{\rm tx}}} \|\boldsymbol{y} - \boldsymbol{Hs}\|^2 . \tag{2}$$

MLD exhaustively searches through all the candidate symbols to find the vector $\hat{\boldsymbol{s}}$, which can minimize the Euclidian distance between the received symbol vector $\boldsymbol{y}$ and the symbol vectors $\boldsymbol{Hs}$. K-best is a breadth-first approach with a performance close to the MLD, and only considers K paths with the shortest partial Euclidian distances (PEDs) when proceeding to the next layer. These paths are called surviving paths. The value of K is a trade-off between detection performance and the computational complexity.

† Corresponding author. Email: jhf@semi.ac.cn

Received 12 December 2008, revised manuscript received 9 March 2009

It is assumed that $N_{\text{rx}} \geqslant N_{\text{tx}}$, which is a necessary condition for the QR decomposition to yield an upper triangular matrix, and hence for the implementation of the K-best detection. We denote the QR decomposition of the channel matrix as $\boldsymbol{H} = \boldsymbol{QR}$, where $\boldsymbol{Q}$ is a $N_{\text{rx}}N_{\text{tx}}$ unitary matrix, $\boldsymbol{Q}^{\boldsymbol{H}}\boldsymbol{Q} = \boldsymbol{I}_{\text{tx}}$, and $\boldsymbol{R}$ is an upper triangular $N_{\text{tx}}N_{\text{tx}}$ matrix. Defining $\boldsymbol{Z} = \boldsymbol{Q}^{\boldsymbol{H}}\boldsymbol{Y}$, we can rewrite Eq. (1) as $\boldsymbol{Z} = \boldsymbol{RS} + \boldsymbol{W}$, where $\boldsymbol{W} = \boldsymbol{Q}^{\boldsymbol{H}}\boldsymbol{V}$. Since $\boldsymbol{Q}$ is unitary, the noise $\boldsymbol{W}$ remains white. Then Equation (1) can be expanded as[4]

$$\hat{s} = \arg \min_{s \in \Omega^{N_{\text{tx}}}} \sum_{i=1}^{N_{\text{tx}}} \left| z_i - \sum_{j=i}^{N_{\text{tx}}} \boldsymbol{R}_{ij} s_j \right|^2 . \tag{3}$$

The above problem can be considered as a tree search problem with N_{tx} layers, starting from the N_{tx} layer due to the upper triangular structure of the matrix $\boldsymbol{R}$. The symbol in each layer is detected based on the previously detected symbols. In the above process, the K-best algorithm searches through all the branches of each surviving path. Therefore, there are KM total candidates to be visited and sorted in every layer. This will incur heavy computational complexity for system using a large number of antennas together with a high order constellation.

3.2. Proposed K-best detection scheme

In the M-QAM case, the modulated symbols are complex-valued. Real valued decomposition (RVD) can be used to facilitate K-best detection[5]. By this means, Equation (1) can be expanded as

$$\begin{bmatrix} \text{Re}(\boldsymbol{Y}) \\ \text{Im}(\boldsymbol{Y}) \end{bmatrix} = \begin{bmatrix} \text{Re}(\boldsymbol{H}) & -\text{Im}(\boldsymbol{H}) \\ \text{Im}(\boldsymbol{H}) & \text{Re}(\boldsymbol{H}) \end{bmatrix} \begin{bmatrix} \text{Re}(\boldsymbol{S}) \\ \text{Im}(\boldsymbol{S}) \end{bmatrix} + \begin{bmatrix} \text{Re}(\boldsymbol{V}) \\ \text{Im}(\boldsymbol{V}) \end{bmatrix}, \tag{4}$$

where $\text{Re}(\boldsymbol{S}), \text{Im}(\boldsymbol{S}) \in \{-\sqrt{M}+1, \cdots, -1, 1, \cdots, \sqrt{M}-1\}$, which denote the real and imaginary parts of the variables, respectively. Using SVD, the N_{tx}-dimensional complex signal can be decomposed into a $2N_{\text{tx}}$-dimensional real-valued signal. Thus, in the i-th layer of the tree, the accumulated PEDs of the surviving paths can be evaluated as[4]

$$T_i(\boldsymbol{S}^i) = T_{i+1}(\boldsymbol{S}^{i+1}) + \left| e_i(\boldsymbol{S}^i) \right|^2 , \tag{5}$$

$$e_i(\boldsymbol{S}^i) = z_i - \sum_{j=i}^{2N_{\text{tx}}} \boldsymbol{R}_{ij} s_j, \tag{6}$$

where $\boldsymbol{S}^i = [s_i s_{i+1} \ldots s_{2N_{\text{tx}}}]^T$, $T_{2N_{\text{tx}}+1}(\boldsymbol{S}^{2N_{\text{tx}}+1}) = 0$, $\left| e_i(\boldsymbol{S}^i) \right|^2$ is called the local distance, which denotes the distance increment between two successive nodes in the tree. According to Eq. (6), for s, any candidate of s_{i} is:

$$e_i(s) = (z_{\text{i}} - \sum_{j=i+1}^{2N_{\text{tx}}} \boldsymbol{R}_{ij} s_j) - \boldsymbol{R}_{ii} s. \tag{7}$$

Let $e = e_i(s)$, $u = (z_i - \sum_{j=i+1}^{2N_{\text{tx}}} \boldsymbol{R}_{ij} s_j)$, for each surviving path, Equation (7) can then be denoted as

$$e = u - \boldsymbol{R}_{ii} s, \tag{8}$$

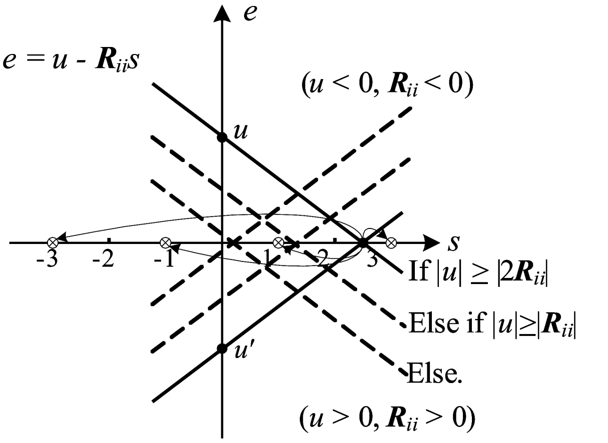
e
$e = u$ - $\boldsymbol{R}_{ii}s$
($u < 0$, $\boldsymbol{R}_{ii} < 0$)
u
s
-3
-2
-1
1
2
3
If $|u| \geq |2\boldsymbol{R}_{ii}|$
Else if $|u| \geq |\boldsymbol{R}_{ii}|$
Else.
u′
($u > 0$, $\boldsymbol{R}_{ii} > 0$)

Fig. 1. Predetermining the ascending order of s_i in the 16-QAM .

Table 1. The ascending order of s_i in the 64-QAM case.

If	Sign (u) = Sign ($\boldsymbol{R}_{\text{ii}}$)	Sign (u) ≠ Sign ($\boldsymbol{R}_{ii}$)
$\lvert u\rvert \geq 6\lvert\boldsymbol{R}_{ii}\rvert$	[7 5 3 1 –1 –3 –5 –7]	[–7 –5 –3 –1 1 3 5 7]
$6\lvert\boldsymbol{R}_{ii}\rvert > \lvert u\rvert \geq 5\lvert\boldsymbol{R}_{ii}\rvert$	[5 7 3 1 –1 –3 –5 –7]	[–5 –7 –3 –1 1 3 5 7]
$5\lvert\boldsymbol{R}_{ii}\rvert > \lvert u\rvert \geq 4\lvert\boldsymbol{R}_{ii}\rvert$	[5 3 7 1 –1 –3 –5 –7]	[–5 –3 –7 –1 1 3 5 7]
$4\lvert\boldsymbol{R}_{ii}\rvert > \lvert u\rvert \geq 3\lvert\boldsymbol{R}_{ii}\rvert$	[3 5 1 7 –1 –3 –5 –7]	[–3 –5 –1 –7 1 3 5 7]
$3\lvert\boldsymbol{R}_{ii}\rvert > \lvert u\rvert \geq 2\lvert\boldsymbol{R}_{ii}\rvert$	[3 1 5 –1 7 –3 –5 –7]	[–3 –1 –5 1 –7 3 5 7]
$2\lvert\boldsymbol{R}_{ii}\rvert > \lvert u\rvert \geq \lvert\boldsymbol{R}_{ii}\rvert$	[1 3 –1 5 –3 7 –5 –7]	[–1 –3 1 –5 3 –7 5 7]
Else	[1 –1 3 –3 5 –5 7 –7]	[–1 1 –3 3 –5 5 –7 7]

where u and $\boldsymbol{R}_{ii}$ are constants. So, Equation (8) is a linear equation with a slope coefficient equal to $-\boldsymbol{R}_{ii}$. Let $e = 0$, then the root of Eq. (8) is

$$\hat{s} = \frac{u}{\boldsymbol{R}_{ii}}. \tag{9}$$

So, we can predetermine the ascending order of s_i by comparing $|u|$ with the products of $|\boldsymbol{R}_{ii}|$ and several certain numbers. In the 16-QAM case, we can compare $|u|$ with $|2\boldsymbol{R}_{ii}|$ and $|\boldsymbol{R}_{ii}|$, respectively, to determine the ascending order of s_i. As illustrated in Fig. 1, when sign (u) = sign ($\boldsymbol{R}_{ii}$) and if $|u| \geqslant |2\boldsymbol{R}_{ii}|$, the ascending order of s_i is [3 1 –1 –3], which means that the local distance $e_i(3) < e_i(1) < e_i(-1) < e_i(-3)$. Otherwise, if $|2\boldsymbol{R}_{ii}| \geqslant |u| \geqslant |\boldsymbol{R}_{ii}|$, the ascending order of s_i is [1 3 –1 –3]. In all other cases, the ascending order of s_i is [1 –1 3 –3]. When sign (u) ≠ sign ($\boldsymbol{R}_{ii}$), the ascending order of s_i can also be predetermined by the same means.

In the 64-QAM case, we can compare $|u|$ with $\{6|\boldsymbol{R}_{ii}|, 5|\boldsymbol{R}_{ii}|, 4|\boldsymbol{R}_{ii}|, 3|\boldsymbol{R}_{ii}|, 2|\boldsymbol{R}_{ii}|, |\boldsymbol{R}_{ii}|\}$, respectively, to predetermine the ascending order of s_i, which is listed in Table 1.

Based on the ascending order of s_i, we can expand each surviving path to its partial branches and update the accumulated PEDs. In order to carry out a limited tree search, we expand the surviving paths on demand. For example, in the case of 16-QAM (K= 4), we divide the K surviving paths into K/2 = 2 groups (G1, G2), where the 1st and 2nd surviving paths are in G1, while the 3rd and 4th are in G2. Then we expand each path in G1 into three branches, while we expand each path in G2 into two branches. The accumulated PEDs of the

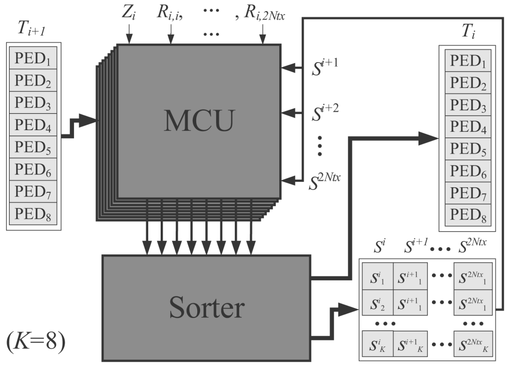

Fig. 2. Proposed *K*-best VLSI architecture.

expanded branches are updated according to Eq. (5). Based on the above expansion results, we can sort out the new *K* surviving paths of the *i*-th layer in a distributed manner. We consider the above example of 16-QAM (K = 4). Firstly, we sort out three candidates from G1 (6 to 3), and two candidates from G2 (4 to 2) by the intercomparison of the branches in each group. Then we can sort out the new K = 4 surviving paths from five good candidates (5 to 4).

By the above novel expansion and sorting methods, the proposed *K*-best scheme can greatly reduce the computational complexity of the tree search. For a 2 × 2 16-QAM case, the proposed scheme only needs to be expanded to 28 branches to sort out the final surviving path; this is nearly half the number needed in the conventional *K*-best algorithm, which needs to be expanded to a total of 52 branches. During the sorting process, the proposed scheme performs 29 comparison operations at most, while the conventional *K*-best algorithm needs to perform 129 comparison operations.

4. Proposed *K*-best VLSI architecture

Based on our proposed *K*-best scheme, we present a new VLSI *K*-best architecture for 4 × 4 MIMO-OFDM systems modulated by 64-QAM. As shown in Fig. 2, the architecture consists of a K = 8 metric computation unit (MCUs) and a *K*-best sorter. Each MCU expands a surviving path to its partial branches and updates the accumulated PEDs. Based on the MCUs' outputs, the distributed sorter sorts out new *K*-best surviving paths. In each stage, the surviving paths and their PEDs are stored in two groups of register lists. The key modules of the proposed architecture are described as follows.

4.1. Metric computation unit

In our proposed architecture, the K = 8 surviving paths are simultaneously processed by eight parallel MCUs, which are divided into $K/2 = 4$ groups (G1–G4). Each group includes two MCUs. Each path in G1–G4 is expanded to 6, 4, 3, and 2 branches, respectively, in order to reduce the complexity. The MCU's structure in G1 is shown in Fig. 3, which expands a surviving path to six branches. In Fig. 3, a group of multipliers M1 and an adder are used to compute u. Another six fixed multipliers M2 and six comparers are used to predetermine the ascending order of the candidates of s_i. Another group of multipliers M3, which are followed by six subtracters and six square absolute value calculators, are used to compute the local distances of the expanded branches. Finally, six adders are used to update the accumulated PEDs. The MCUs export the expanded branches and their corresponding accumulated PEDs to the sorter.

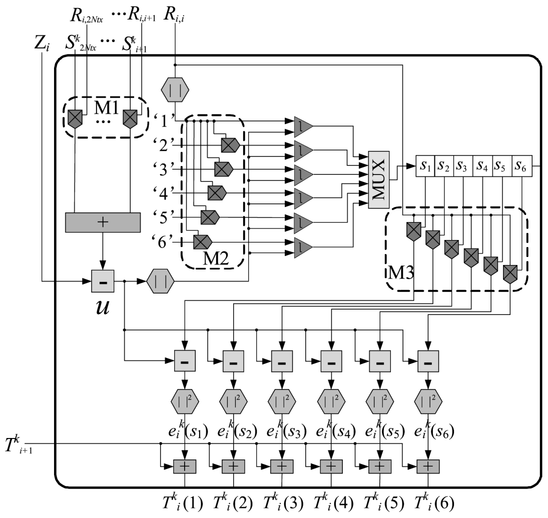
$R_{i,2Ntx}$ ⋯$R_{i,i+1}$ $R_{i,i}$
Z_i S^k_{2Ntx} ⋯ S^k_{i+1}
M1
M2
M3
'1'
'2'
'3'
'4'
'5'
'6'
MUX
s_1 s_2 s_3 s_4 s_5 s_6
+
-
u
| |
| |^2
$e_i^k(s_1)$ $e_i^k(s_2)$ $e_i^k(s_3)$ $e_i^k(s_4)$ $e_i^k(s_5)$ $e_i^k(s_6)$
T^k_{i+1}
$T^k_i(1)$ $T^k_i(2)$ $T^k_i(3)$ $T^k_i(4)$ $T^k_i(5)$ $T^k_i(6)$

Fig. 3. Proposed MCU structure.

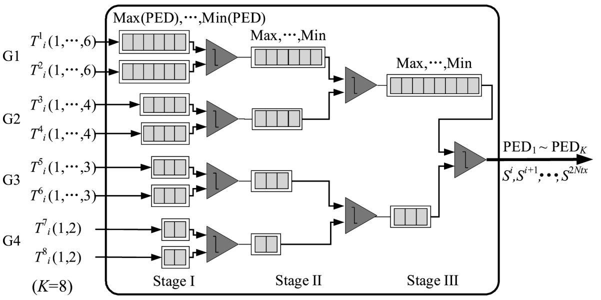

Fig. 4. Proposed sorter structure for 64-QAM (K = 8).

4.2. Sorter

The proposed distributed sorter is illustrated in Fig. 4, which can be divided into three stages. Each stage includes a group of register lists and some comparers. The sorter works in pipelines. Stage I sorts out 6, 4, 3, 2 candidates from G1 to G4 (12 to 6, 8 to 4, 6 to 3, 4 to 2). Based on the outputs of stage I, stage II sorts out eight (10 to 8) and three (5 to 3) shorter paths. Finally, stage III sorts out the new K = 8 surviving paths from all the eleven candidates yielded by stage II (11 to 8). For each comparison operation, the candidate with shorter accumulated PEDs is chosen as the output and replaced by its nearest sibling, while the longer one stays for the next comparison.

5. Complexity analysis and simulation

In the *K*-best scheme, the addition, the multiplication, and the comparison play critical roles in the computational complexity. Table 2 shows the comparison of the fundamental operations between the conventional *K*-best scheme and the proposed architecture. According to Table 2, the proposed architecture can approximately reduce the fundamental opera-

Table 2. Comparison of number of fundamental operations between different *K*-best implementations.

N_{tx}	Operations	16-QAM, K = 4		64-QAM, K = 8	
		This work	Conventional *K*-best	This work	Conventional *K*-best
2	Addition/Subtraction	76	124	192	504
	Multiplication	52	72	124	312
	Comparison	⩽ 29	129	⩽ 99	1043
3	Addition/Subtraction	152	224	384	768
	Multiplication	108	124	256	448
	Comparison	⩽ 51	246	⩽ 191	1995
4	Addition/Subtraction	244	428	608	1128
	Multiplication	180	228	420	680
	Comparison	⩽ 73	⩽345	260	2947

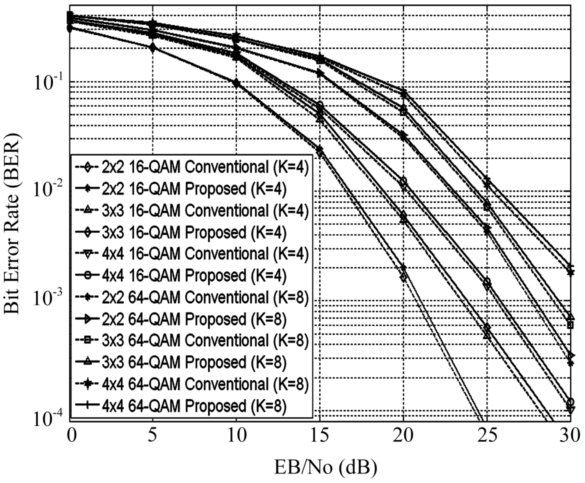

Fig. 5. BER of different *K*-best implementations.

tions by 50% and 75% for the 16-QAM and the 64-QAM, respectively, consequently significantly saving hardware resources. So, the proposed architecture is especially suitable for high-throughput MIMO-OFDM systems.

The performance of the proposed *K*-best VLSI architecture is simulated using a computer and the Rayleigh fading channel model. The simulation results for both 16-QAM (*K*= 4) and 64-QAM (*K*= 8) schemes are presented in Fig. 5, from which we can see that the bit error rate (BER) performance of the proposed architecture is very close to that of the conventional *K*-best scheme.

6. Conclusions

In this paper, a novel *K*-best VLSI architecture for the high-throughput MIMO-OFDM detector is presented. The proposed MCU predetermines the ascending order of each surviving path's branches by their local distances and expands the surviving paths only to partial branches. Then the distributed sorter sorts out the new surviving paths in pipelines. Complexity analysis and simulation results prove that the proposed architecture can significantly reduce the computational complexity while maintaining the similar performance of the conventional *K*-best scheme, yielding an efficient solution to the VLSI implementation of high-throughput MIMO-OFDM systems.

References

[1] Bölcskei H. MIMO-OFDM wireless systems: basics, perspectives, and challenges. IEEE Wireless Commun, 2006, 13(4): 31

[2] Burg A, Borgmanr M, Wenk M, et al. Advanced receiver algorithms for MIMO wireless communications. Design, Automation and Test in Europe, Munich, Germany, 2006, 1: 593

[3] Fujita T, Onizawa T, Jiang W, et al. A new signal detection scheme combining ZF and *K*-best algorithms for OFDM/SDM. 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2004, 4: 2387

[4] Shabany M, Su K, Gulak P G, et al. A pipelined scalable high-throughput implementation of a near-ML *K*-best complex lattice decoder. ICASSP, 2008: 3173

[5] Wenk M, Zellweger M, Burg A, et al. *K*-best MIMO detection VLSI architectures achieving up to 424 Mbps. ISCAS Circuits and Systems, 2006: 4

[6] Li Q, Wang Z. An improved *K*-best sphere decoding architecture for MIMO systems. Fortieth Asilomar Conference on Signals, Systems and Computers, 2006: 2190