Reconfigurable Adaptive Singular Value Decomposition Engine Design for High-Throughput MIMO-OFDM Systems

Yen-Liang Chen, Cheng-Zhou Zhan, Ting-Jyun Hseng, and An-You (Andy) Wu, Member, IEEE

Abstract—Singular value decomposition (SVD) is an optimal method to obtain spatial multiplexing gain in MIMO channels. However, the high implementation cost and high decomposing latency of the SVD restricts its usage in current wireless communication applications. In this paper, we present a complete adaptive SVD algorithm and a reconfigurable architecture for high-throughput MIMO-OFDM systems. There are several proposed architectural design techniques: reconfigurable scheme, division-free adaptive step size scheme, early termination scheme, and data interleaving scheme. The reconfigurable scheme can support all antenna configurations in a MIMO system. The division-free adaptive step size and early termination schemes are used to effectively reduce the decomposing latency and improve hardware utilization. The data interleaving scheme helps to deal with several channel matrices concurrently. Besides, we propose an orthogonal reconstruction scheme to obtain more accurate SVD outputs, and then the system performance will be greatly enhanced. We apply our SVD design to the IEEE 802.11n applications. This design is implemented and fabricated in UMC 90 nm 1P9M CMOS technology. The maximum operating frequency is measured 101.2 MHz, and the corresponding power dissipation is 125 mW. The core size is 2.17 mm$^2$ and the die size occupies 4.93 mm$^2$. The chip result shows that the average latency is only 0.33% of the wireless local area network (WLAN) coherence time. Hence, the proposed reconfigurable adaptive SVD engine design is very suitable for high-throughput wireless communication applications.

Index Terms—SVD, MIMO, OFDM, reconfigurable architecture, adaptive array processing.

I. INTRODUCTION

Due to the rapid evolution of wireless communication and the demand of high data rate for multi-media information access in recent years, the single-input single-output (SISO) transmission has become insufficient [1], [2]. Therefore, the research about multi-input multi-output (MIMO) technology becomes an important topic in many advanced wireless communication standards [3]–[5]. The advantage of a MIMO system is that it exploits the space dimension to improve the system capacity and reliability. However, in a MIMO system, one receive antenna may suffer from the interference of other transmit antennas [6], [7]. This leads the receiver hard to obtain correct data. By applying the singular value decomposition (SVD) technique [8]–[10], the interference can be totally eliminated. Hence, the throughput and coverage of a MIMO system can be greatly enhanced. From an information-theoretical viewpoint, the use of SVD can be claimed as an optimal solution [11]–[13]. Besides, the advanced WLAN standard, IEEE 802.11n [14]–[16], has treated the SVD technique as an optional MIMO signal processing technique to enhance system performance. [17] also shows that the application of the SVD technique has the highest throughput compared with other MIMO signal processing techniques in the IEEE 802.11n systems. This indicates that the SVD technique is very important for the MIMO wireless communication systems.

Nowadays, there are several issues in applying the SVD technique to the wireless communication systems. These issues are discussed in detail as follows.

- In many wireless communication standards, a MIMO system is usually combined with orthogonal frequency division multiplexing (OFDM) technology. The SVD engine needs to deal with hundreds of channel matrices of almost all subcarriers before data transmission. Hence, it is important to effectively reduce the total computational complexity.
- In the WLAN environment, the coherence time over which the channel is considered essentially time-invariant is about 0.07 sec [17], [18]. This indicates that we should complete the SVD operations of all channel matrices as soon as possible. Otherwise, the SVD results cannot be used for the present channel condition.
- Assume that a MIMO system consists of up to $M_T$ transmit antennas and $M_R$ receive antennas. There are possibly $M_R \cdot M_T$ antenna configurations as well as channel matrix sizes. Hence, it is necessary to design a reconfigurable SVD engine for all antenna configurations. For example, in an 802.11n system, the number of transmit antennas or receive antennas is defined from 1 to 4. The SVD engine should be capable of dealing with 16 antenna configurations.

Y.-L. Chen is with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: ben@access.ee.ntu.edu.tw).

C.-Z. Zhan is with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: chenjo@access.ee.ntu.edu.tw).

T.-J. Hseng is with MediaTec Inc., Hsinchu 31040, Taiwan (e-mail: xtra@access.ee.ntu.edu.tw).

A.-Y. Wu is with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: andywu@access.ee.ntu.edu.tw).

The material in this paper was presented in part at ESSCIRC 2010, Seville, Spain.
In recent years, [19] proposed one ASIC realization of the SVD without the need of CSI for WLAN applications. However, the chip implements an adaptive blind-tracking \( \Sigma \) algorithm [20] which is not complete for SVD outputs, and long convergence time is another disadvantage for the high-throughput MIMO-OFDM applications. [27] proposed a matrix decomposition architecture according to the Golub-Kahan SVD (GK-SVD) algorithm [29]. It achieves higher processing throughput than [19] with lower hardware cost. Based on the matrix decomposition architecture in [27], [28] proposed a hardware-efficient VLSI architecture by modifying the GK-SVD algorithm and using a high-speed Givens rotation design. To increase the processing speed, it only computes \( \mathbf{V} \) and \( \Sigma \) which are partial of SVD outputs. Nevertheless, the above-mentioned SVD designs only support 4x4 (4 transmit and 4 receive antennas) antenna configuration which is not enough for dealing with different antenna configurations.

In this paper, we propose a complete adaptive SVD algorithm, as well as a reconfigurable architecture design, for the high-throughput MIMO-OFDM systems. Some key features are listed as follows: 1) Adaptive step size scheme, partial update scheme, and subcarrier inherit scheme to effectively reduce the decomposing latency, and to increase the processing throughput. 2) Reconfigurable architecture for all antenna configurations in a MIMO system. 3) Early termination scheme to improve hardware utilization without losing system performance. 4) Data interleaving scheme to deal with several channel matrices simultaneously. 5) Orthogonal reconstruction scheme to enhance the system performance. We implement the proposed reconfigurable SVD engine for the application of the IEEE 802.11n systems with up to four transmit antennas and four receive antennas. This chip is implemented using 90nm CMOS technology with core area of 2.17 mm\(^2\). It can be measured at 101.2 MHz with 125 mW power consumption. As compared with other related works, this chip achieves the highest throughput and power efficiency in the 4x4 SVD operations. In addition, the chip result shows that for an 802.11n system, the average latency of our SVD engine is only 0.33\% of the WLAN coherence time. Therefore, the proposed SVD engine is very suitable for the high-throughput MIMO-OFDM systems.

The remainder of this paper is organized as follows. In Section II, we introduce the SVD technique in a MIMO system and review the adaptive blind-tracking \( \Sigma \) algorithm. In Section III, the proposed complete adaptive SVD algorithm is presented. The proposed architectural design techniques for reconfigurable SVD engine are demonstrated in Section IV. The orthogonal reconstruction scheme is described in Section V. Section VI demonstrates the simulation and implementation results of the proposed SVD engine. Finally, we conclude this paper in Section VII.

In this paper, the following notation will be adopted. We use boldfaced capital letters to indicate matrices and boldfaced lowercase letters to indicate vectors. \( \mathbf{I} \) denotes the identity matrix. \( \cdot \)\(^H \) denotes the complex transpose of a vector or matrix. \( \operatorname{tr}(\cdot) \) denotes the trace of a matrix. \( \| \cdot \| \) denotes the two-norm of a vector. \( \mathbf{R}(\cdot, k) \) denotes the \( k \)th column of the matrix \( \mathbf{R} \). \( \langle \mathbf{a}, \mathbf{b} \rangle \) is the Euclidean inner product as \( \mathbf{b}^H \mathbf{a} \). \( \mathbf{C}^{p \times 1} \) denotes the set of \( p \times 1 \) complex vectors, and \( \mathbf{C}^{p \times q} \) denotes the set of \( p \times q \) complex matrices.

### II. INTRODUCTION TO SVD TECHNIQUE

#### A. MIMO System Model and SVD

Consider a MIMO system with \( N_T \) transmit and \( N_R \) receive antennas. The baseband, discrete-time equivalent model is written by \( \mathbf{y} = \mathbf{Hx} + \mathbf{z} \), where \( \mathbf{H} \subseteq \mathbf{C}^{N_R \times N_T} \) is the complex channel matrix. \( \mathbf{z} \subseteq \mathbf{C}^{N_R} \) is the additive white complex Gaussian noise vector. \( \mathbf{x} \subseteq \mathbf{C}^{N_T} \) is the transmitted data vector, and \( \mathbf{y} \subseteq \mathbf{C}^{N_R} \) is the received data vector. If we decompose the channel matrix \( \mathbf{H} \) by the SVD technique, we have

\[
\mathbf{H} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^H, \quad (1)
\]

where \( \mathbf{U} \) and \( \mathbf{V} \) are an \( N_R \times N_R \) left singular matrix and an \( N_T \times N_T \) right singular matrix, respectively. Both \( \mathbf{U} \) and \( \mathbf{V} \) are unitary matrices (i.e., \( \mathbf{U}^H \mathbf{U} = \mathbf{I} \) and \( \mathbf{V} \mathbf{V}^H = \mathbf{I} \)). \( \mathbf{\Sigma} \) is an \( N_R \times N_T \) matrix with only real and non-negative main diagonal entries. The entry \( (i, i) \) of \( \mathbf{\Sigma} \) denotes the \( i \)th largest value \( \sigma_i \), with \( 1 \leq i \leq \min(N_R, N_T) \).

Let \( \mathbf{x}' \subseteq \mathbf{C}^{N_T} \) be the symbol vector such that \( \mathbf{x} = \mathbf{V} \mathbf{x}' \) and the received signal \( \mathbf{y} \) is multiplied by \( \mathbf{U}^H \) as shown in Fig. 1. The channel between \( \mathbf{x}' \) and \( \mathbf{y}' \) can be written as

\[
\mathbf{y}' = \mathbf{U}^H \mathbf{y} = \mathbf{U}^H (\mathbf{Hx} + \mathbf{z}) = \mathbf{U}^H \mathbf{HVx}' + \mathbf{z}' = \mathbf{\Sigma} \mathbf{x}' + \mathbf{z}'. \quad (2)
\]

Note that the distribution of \( \mathbf{z}' \) is invariant since \( \mathbf{U} \) is unitary. The MIMO channel can be treated as \( d = \min(N_R, N_T) \) independent parallel Gaussian subchannels. The \( i \)th subchannel has the gain being \( \sigma_i \). Hence, the transmitter can send independent data streams across these parallel subchannels without any antenna interference. Note that the values \( \sigma_1, \sigma_2, \ldots, \sigma_d \) are called the singular values of \( \mathbf{H} \). The column vectors of \( \mathbf{V} \) (i.e., \( \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_T} \)) are the right singular vectors of \( \mathbf{H} \), and the column vectors of \( \mathbf{U} \) (i.e., \( \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N_R} \)) are the left singular vectors of \( \mathbf{H} \).

#### B. Review of Adaptive Blind-Tracking \( \Sigma \) Algorithm

In [20], the authors proposed an adaptive blind-tracking \( \Sigma \) algorithm for \( \mathbf{U} \) and \( \mathbf{\Sigma} \) as shown in Table I. \( n \) denotes the discrete time index. Without loss of generality, we omit the time index \( n \) in this subsection for simplicity. The received signal \( \mathbf{y} \) is used to estimate the autocorrelation matrix \( \mathbf{K}_y = E[\mathbf{yy}^H] \). Hence \( \mathbf{K}_y \) is the estimated autocorrelation matrix of \( \mathbf{K}_y \). \( \beta \) is the forgetting factor and its choice depends on the
stationary degree of the channel. $d$ is the number of useful subchannels. The algorithm is to perform LMS-based estimation to find the pair $(\mathbf{w}_i, \lambda_i)$. The step size $\mu_i$ controls the convergence speed and accuracy. The deflation process cancels the information of the pair $(\mathbf{w}_i, \lambda_i)$ for the estimation of next pair $(\mathbf{w}_{i+1}, \lambda_{i+1})$. The blind-tracking and deflation process continues until all pairs are estimated. The singular pairs $(\mathbf{u}_i, \sigma_i)$ of channel matrix $\mathbf{H}$ can be derived by the use of the pairs $(\mathbf{w}_i, \lambda_i)$ as follows:

$$\sigma_i = \sqrt{\lambda_i}, \quad \mathbf{u}_i = \frac{\mathbf{w}_i}{\sqrt{\lambda_i}}, \quad i = 1, 2, \ldots, d. \quad (3)$$

[19] implemented the adaptive blind-tracking UΣ algorithm for 4x4 antenna system as shown in Fig. 2. The forgetting factor $\beta$ is set to one. Hence, the autocorrelation matrix is estimated by using the instantaneous received signals only. This reduces the computational complexity at the expense of additional square root and division. The step size $\mu_i$ is adaptively adjusted as $0.05/\lambda_i$.

### III. PROPOSED ADAPTIVE SVD ALGORITHM

In many MIMO OFDM-based communication standards, the channel matrix $\mathbf{H}$ can be obtained through channel estimation [21]-[23]. With this additional information, we propose a complete adaptive SVD algorithm for high-throughput MIMO OFDM-based applications. The BER performance may be affected by imperfect channel estimation, $\mathbf{H}$, and the degradation discussed in referenced works [21]-[23] about the channel estimation which is beyond the scope of this work.

#### A. The Derivation of Matrix $\mathbf{R}_i$

In Table I, the positive semidefinite matrix $\mathbf{R}_i$ is estimated by a moving average of the recent received signal vectors. In many MIMO OFDM-based standards, the channel matrix $\mathbf{H}$ is already known by channel estimation. Therefore, we can utilize the information to evaluate accurate $\mathbf{R}_i$:

$$\mathbf{R}_i = \begin{cases} \mathbf{H}^H \mathbf{H}, & N_R \geq N_T, \\ \mathbf{H}^H, & N_R < N_T. \end{cases} \quad (4)$$

With this definition of $\mathbf{R}_i$, we can still use the same update and deflation process to find the pairs $(\mathbf{w}_i, \lambda_i)$ sequentially. In the $i$th update process, we have

$$\mathbf{w}_{i}(n+1) = \mathbf{w}_i(n) + \mu_i (\mathbf{R}_i - \lambda_i(n) \mathbf{I}) \mathbf{w}_i(n),$$

$$\lambda_i(n+1) = \lambda_i(n) + \mu_i (\mathbf{R}_i - \lambda_i(n) \mathbf{I}) \mathbf{w}_i(n),$$

$$\mu_i(n+1) = \frac{0.05}{\lambda_i(n+1)} \quad (i=1,2,3,4)$$

where $d$ is $\min(N_R, N_T)$. And the $i$th deflation process is given by

$$\mathbf{R}_{i+1} = \mathbf{R}_i - \mathbf{w}_i(n+1) \mathbf{w}_i(n+1)^H, \quad i = 1, 2, \ldots, (d-1). \quad (6)$$

After convergence, we have $\mathbf{w}_i$ and $\lambda_i$ with $1 \leq i \leq (d-1)$. We can derive the singular values and the corresponding singular vectors of $\mathbf{H}$ by using the pairs $(\mathbf{w}_i, \lambda_i)$. Since it is possible to have $H_R \geq N_T$ or $N_R < N_T$, there are two cases to be considered. For the case when $N_R \geq N_T$, we have

$$\sigma_i = \sqrt{\lambda_i}, \quad \mathbf{v}_i = \frac{\mathbf{w}_i}{\sqrt{\lambda_i}}, \quad \mathbf{u}_i = \frac{\mathbf{H} \mathbf{v}_i}{\sigma_i}, \quad i = 1, 2, \ldots, (d-1). \quad (7)$$

On the other hand, when $N_R < N_T$, we only need to interchange $\mathbf{v}_i$ with $\mathbf{u}_i$, and $\mathbf{H}$ is changed to $\mathbf{H}^H$ in (7).

#### B. Partial Update Scheme

In Table I, $\mathbf{w}_d$ and $\lambda_d$ are derived by applying the update operation. From our observation, after the $(d-1)$-time deflation, the positive semidefinite matrix $\mathbf{R}_d$ can be expressed as

$$\mathbf{R}_d = \mathbf{w}_d \mathbf{w}_d^H. \quad (8)$$

Hence, the update operation for $\mathbf{w}_d$ and $\lambda_d$ is unnecessary. We can directly find the $d$th singular value and the corresponding singular vectors by some simple operations. For the case of $N_R \geq N_T$, we get

$$\sigma_d = \sqrt{\text{tr}(\mathbf{R}_d)}, \quad \mathbf{v}_d = \frac{\mathbf{R}_d^{(:,1)}}{\|\mathbf{R}_d^{(:,1)}\|}, \quad \mathbf{u}_d = \frac{\mathbf{H} \mathbf{v}_d}{\sigma_d}. \quad (9)$$

On the other hand, when $N_R < N_T$, we only need to interchange $\mathbf{u}_d$ with $\mathbf{v}_d$, and $\mathbf{H}$ is changed to $\mathbf{H}^H$ in (9). The advantage of applying partial update is to effectively reduce the decomposing latency.

#### C. Adaptive Step Size Scheme

The step size $\mu_i$ is an important parameter for the convergence speed and stability of the algorithms. As mentioned in Appendix, the objective function is a quartic function which is complicated (also mentioned in [19]) to derive the exact bound of the step size. We derive a loose bound by approximating the objective function from a quartic function to quadratic function in Appendix. We have derived a convergence region and a near-optimal step size as follows

$$0 < \mu_i < \frac{1}{\lambda_i}, \quad (10)$$

and
\[
\mu_i = \frac{2}{3\lambda_i} > \frac{2}{3\lambda_{i+1}}
\]  

(11)

Hence, fixed step size is inefficient and not robust for all kinds of channel matrices. In [19], the step size is adaptively adjusted as \(0.05/\langle\lambda(n)\rangle\) which is too small for fast convergent purpose. Therefore, for the goal of fast and stable convergence, the proposed adaptive step size is given by

\begin{table}
\centering
\caption{Pseudo-code of the Proposed Adaptive SVD Algorithm}
\begin{tabular}{l}
\hline
Given \(H, N_x, N_y\) \\
\(R_i = H^i H^i, N_x \geq N_y\) \\
d = \min(N_x, N_y) \\
1) Update and Deflation \\
for \(i = 1: (d-1)\), \\
\begin{align*}
\text{Initial setting} \\
& w(0) = \text{adjacent subcarrier’s } w_i(\infty) \\
& \lambda(0) = \langle w(0) \rangle^2/\langle w(0) \rangle
\end{align*}
\end{tabular}
\end{table}

\begin{tabular}{l}
\hline
Update the \(i\)-th pair \\
end \\
2) Derivation of \(u, v, \sigma_j, \) \\
if \(N_x \geq N_y\) \\
\begin{align*}
\sigma_j &= \frac{\langle u_j \rangle}{\lambda_j}, v_j = \frac{w_j}{\langle u_j \rangle}, u_j = H v_j / \sigma_j \\
\text{end}
\end{align*}
else \\
\begin{align*}
\sigma_j &= \frac{1}{\sigma_j}, u_j = H v_j / \sigma_j \\
\text{end}
\end{align*}
end \\
3) Partial Update for \(u, v, \sigma\) \\
if \(N_x \geq N_y\) \\
\begin{align*}
\sigma_j &= \frac{\langle u_j \rangle}{\lambda_j}, v_j = \frac{R_j}{\langle R_j \rangle}, u_j = H v_j / \sigma_j \\
\text{end}
\end{align*}
else \\
\begin{align*}
\sigma_j &= \frac{1}{\sigma_j}, u_j = H v_j / \sigma_j \\
\text{end}
\end{align*}
end \\
4) Gram-Schmidt for remaining singular vectors \\
if \(N_x \geq N_y\) \\
for \(k = 1: (N_x - N_y)\) \\
\begin{align*}
& w_{x,i} = e_i - \sum_{i=1}^{k-1} (e_i, u_i) \cdot u_i \\
& w_{y,i} = \left[ w_{x,i} \right]
\end{align*}
end \\
else \\
for \(k = 1: (N_y - N_x)\) \\
\begin{align*}
& w_{x,i} = e_i - \sum_{i=1}^{k-1} (e_i, v_i) \cdot v_i \\
& w_{y,i} = \left[ w_{x,i} \right]
\end{align*}
end \\
end
\end{table}

\[
\mu_i(n) = \frac{a}{\lambda_i(n)},
\]  

(12)

where \(a\) is a scaling factor. From (11), we suggest that the value of \(a\) could be 0.75 or 0.5 for hardware-friendly implementation.

D. Subcarrier Inherit Scheme

In (5), we have to give the initial values of \(\{w_i(n)\}_{n=1}^{\infty}\) for each update process. Although the update equation in (5) surely converges with arbitrary initial values, choosing good initial values can help to speed up the update processes. We denote \(w_i(0)\) and \(w_i(\infty)\) as the initial and converged values of \(w_i(n)\). In a wireless MIMO-OFDM system, since two adjacent subcarriers often have similar channel matrices, one subcarrier’s converged information is useful to its adjacent subcarrier. Therefore, if one subcarrier’s \(\{w_i(\infty)\}_{n=1}^{\infty}\) is obtained, we can take the converged values as its adjacent subcarrier’s initial values of \(\{w_i(n)\}_{n=1}^{\infty}\). It should be noted that pilot and null subcarriers will be skipped since they do not need SVD operations.

E. Gram-Schmidt Scheme for Non-square Matrix

Generally speaking, for an \(N_y \times N_x\) channel matrix, we need to find \(d\) singular values, \(N_x\) left singular vectors, and \(N_y\) right singular vectors, where \(d = \min(N_x, N_y)\). After applying the above schemes, we can find \(d\) singular values, \(d\) left singular vectors, and \(d\) right singular vectors. If the channel matrix is square, it means that \(d = N_x = N_y\). Therefore, we can find all singular values and singular vectors. But for the case of non-square channel matrix, assume that \(N_x > N_y\), we have \(d = N_y\); there are still \((N_x - N_y)\) unsolved left singular vectors (i.e., \(u_{N_y+1}, u_{N_y+2}, \ldots, u_{N_x}\)) after applying the above schemes. On the other hand, when \(N_x < N_y\), there are \((N_y - N_x)\) right singular vectors (i.e., \(v_{N_y+1}, v_{N_y+2}, \ldots, v_{N_y}\)). Notice that both the cases are similar. To find these remaining vectors, recall that \(U\) and \(V\) are the unitary matrices, the column vectors in \(U\) or \(V\) are orthonormal to each other. That is

\[
\langle u_i, u_j \rangle = 0, \quad \forall i \neq j,
\]  

(13)

and

\[
\langle v_i, v_j \rangle = 0, \quad \forall i \neq j.
\]  

(14)

Therefore, the remaining vectors can be obtained by applying the Gram-Schmidt technique [8]. First, we consider the case of \(N_x > N_y\). After applying the above schemes, we already have \(u_1, u_2, \ldots, u_{N_y}\). Then the remaining left singular vectors can be obtained by

\[
w_{d+i} = e_k - \sum_{i=1}^{d+i-1} \langle e_k, u_i \rangle \cdot u_i,
\]  

\[
u_{d+i} = \frac{w_{d+i}}{\|w_{d+i}\|}, \quad k = 1, 2, \ldots, (N_x - N_y),
\]  

(15)

where \(e_k\) is orthonormal to \(e_i\) with \(k \neq i\), and \(e_k\) is unequal to \(u_i\) with \(1 \leq i \leq (d + k - 1)\). Note that for the case of \(N_x < N_y\), we only need to replace \(u_i\) with \(v_i\) and to interchange \(N_x\) with \(N_y\) in (15). The proposed adaptive SVD algorithm is summarized in Table II.

IV. ARCHITECTURE DESIGN OF PROPOSED SVD ENGINE

The block diagram of the proposed reconfigurable adaptive SVD engine is depicted in Fig. 3. There are 2 single-port SRAM banks in the memory module, and 416 entries \(\times 80\) bits memory banks in the \(H\) buffers. The detailed wordlength consideration of the architecture and memory banks will be discussed in Section VI.D. It consists of six functional units which are zero padding unit, deflation unit, update unit, singular calculation unit, partial update unit, and simplified Gram-Schmidt unit. We could implement deflation unit directly and the block diagram of deflation unit derived from (6) is shown in Fig. 4. The register REG is used to store all entries of the positive semidefinite matrix. In the first update process, \(R_i = R_i\). After the first update process, \(R_{i+1}\) is derived from \(R_i\). In the remainder of this section, each unit will be described in more detail.
depends on the size of the original channel matrix. Therefore, \( d \) is still equal to \( \min(N_R, N_T) \). Fig. 5 shows the block diagram of zero padding unit. A given channel matrix \( \mathbf{H}_{N_R \times N_T} \) is extended to \( \mathbf{H}_{M_R \times M_T} \) by inserting zeros, and the multiplexer is used to construct the positive semidefinite matrix \( \mathbf{R}_1 \), based on (4). We also apply the zero padding scheme to singular calculation unit and partial update unit. According to (7), Fig. 6 illustrates the architecture of singular calculation unit. Three multiplexers are used to consider two cases of \( N_R \geq N_T \) and \( N_R < N_T \). We employ (9) to realize partial update unit as shown in Fig. 7.

2) Simplified Gram-Schmidt Scheme for Non-Square Channel Matrix

In (15), we apply Gram-Schmidt technique to find the remaining vectors for the case of \( N_R > N_T \). Due to the fact that the entries of a channel matrix, as well as the entries of its singular vectors, are always complex-valued, we can define \( \mathbf{e}_k \) as a unit vector with the \( k \)-th entry being 1. With this setting, we can rewrite (15) into a more simplified form:
\[ w_{d+k} = e_k - \sum_{i=1}^{d+k-1} u_{i,k}^{*} \cdot u_i \]
\[ = e_k - \left[ u_{1,k}, u_{2,k}, \ldots, u_{d+k-1,k} \right] \left[ u_{1,k}, u_{2,k}, \ldots, u_{d+k-1,k} \right]^H \]
\[ = e_k - G_k g_k, \quad (17) \]

where \( u_{i,k} \) means the \( k \)-th element of \( u_i \). Note that for the case of \( N_R < N_T \), we only need to replace \( u_i \) with \( v_i \) and to interchange \( N_R \) with \( N_T \) in (17). After this simplification, it is easier to implement a reconfigurable Gram-Schmidt design for different sizes of channel matrices. We can choose the maximum size of \( e_k, G_k, \) and \( g_k \) in advance. In an \( M_R \times M_T \) MIMO system, the maximum size of \( e_k, G_k, \) and \( g_k \) is \( L \times 1, L \times (L-1), \) and \( (L-1) \times 1 \) respectively, where \( L = \max(M_R, M_T) \). For smaller-size antenna configurations, we just need to insert zeros in \( e_k, G_k, \) and \( g_k \). The block diagram of Gram-Schmidt unit is shown in Fig. 8 for the case of \( N_R > N_T \).

---

**B. Architectural Design of Update Unit**

The main computational time of our SVD architecture is in the update unit. Fig. 9 shows the block diagram of the original update unit based on (5) and (12). For the architectural design of the update unit, we propose three schemes to reduce the decomposing latency and enhance the hardware utilization.

1) **Division-Free Adaptive Step Size Scheme**

In order to achieve fast convergent purpose, the step size \( \mu(n) \) is adaptively adjusted with \( \lambda(n) \). Obviously, in Fig. 9, there is a division at every iteration in the update unit. This will slow down the operating speed. For this reason, we propose a division-free adaptive step size scheme to avoid the division in the update unit. Due to the property of the step size [9], we don’t need to calculate the exact value of \( \mu(n) \). From (12), the step size is in inverse proportion to \( \lambda(n) \). If we transform \( \lambda(n) \) into a number of powers of two which is the nearest to and greater than \( \lambda(n) \). Hence, the new step size can be expressed as

\[ \mu'(n) = \frac{a}{2^t}, \quad (18) \]

where \( t \) is an integer, and its value depends on the word-length of \( \lambda(n) \). Since the new step size is a number of powers of two, a shift operation can be substituted for a division at every iteration in the update operation. Fig. 10 shows the mapping circuit that transforms \( \lambda(n) \) into a number of powers of two, and the block diagram of the update unit with division-free adaptive step size scheme is shown in Fig. 11. Also notice that

\[ 0 < \mu'(n) \leq \mu(n). \quad (19) \]

The stability of convergence is still guaranteed. Although the number of converged iterations increases, the required time at every iteration can be effectively reduced. Hence, the overall latency is reduced.

2) **Early Termination Scheme**

In (5), the correction vector for \( w_i(n+1) \) is given by

\[ \Delta w_i(n) = \mu(n) (R - \lambda(n)I) w_i(n). \quad (20) \]

For a floating-point view, \( \Delta w_i(n) \) is always non-zero. However,
for a fixed-point implementation, if every entry of $\Delta w_i(n)$ satisfies the following condition:

$$\Delta w_i(n) < 2^{-\text{Fractional Length of } w_i(n)},$$

where $\Delta w_i(n)$ is the $k$-th element of $\Delta w_i(n)$. Then $\Delta w_i(n)$ can be considered as a vector with all elements being zeros if $w_i(n)$ is converged. Clearly, after $w_i(n)$ is converged, the remaining iteration operation is redundant. In order to further reduce decomposing latency and enhance hardware utilization, we propose an early termination scheme as follows:

If $\mu_i (R - \hat{\lambda}_i(n) I) w_i(n) = 0$

Terminate and go to the deflation operation

else

Keep iterative operation

$$\Delta w_i(n) \rightarrow \text{flag}$$

Fig. 12. Hardware design of the early termination scheme.

$$w_{i,j} \rightarrow \text{MUX} \rightarrow R_{i,1}, R_{i,2}, \ldots, R_{i,16} \rightarrow \text{To memory unit}$$

From memory unit

$$\text{Zero Padding} \rightarrow \text{Zero Padding} \rightarrow \text{Zero Padding} \rightarrow \text{MUX} \rightarrow \text{Normalization} \rightarrow \text{To update unit}$$

To update unit

$$w_{i,j} \rightarrow \text{MUX} \rightarrow R_{i,1}, R_{i,2}, \ldots, R_{i,16} \rightarrow \text{16}$$

where $0$ is an all-zero vector. The hardware design of early termination scheme is illustrated in Fig. 12. The “flag” signal is used to check that the terminated condition is met or not. If “flag” equals bit 0, the entries of the correction vector are all zeros and the update operation will be terminated. The block diagram of the update unit with early termination scheme is shown in Fig. 11. Notice that the overall performance with early termination is the same as that without early termination.

3) Data Interleaving Scheme

For the MIMO OFDM-based communication standards, there are tens or hundreds of subcarriers, and each subcarrier has its own channel matrix. Hence, the SVD engine needs to deal with these channel matrices before data transmission. Motivated from [19], [24], we apply the concept of data-interleaving to our SVD engine to deal with 16 channel matrices at the same time. The main architectural change is in the update unit as shown in Fig. 11, where $R_{i,j}$ means the $j$-th positive semidefinite matrix in the $i$-th update process, and $(w_{i,j}, \hat{\lambda}_{i,j})$ is the $i$-th update pair for $R_{i,j}$. The critical path is in the update unit, therefore we use data-interleaving scheme to insert 16 memory units (registers) in each loop of the update unit to store $w_{i,j}$ and $\hat{\lambda}_{i,j}$ of each channel matrix. Notice that the data interleaving scheme must be applied to deflation unit to store 16 positive semidefinite matrices as shown in Fig. 13.

V. ORTHOGONAL RECONSTRUCTION FOR FIXED-POINT IMPLEMENTATION

In (13) and (14), the orthogonal property among the singular vectors is preserved in floating-point representation. However, since all the elements are expressed in finite precision in fixed-point implementation, the orthogonal property will be destroyed. Applying the SVD operation to the channel matrix $H$, we have

$$\Sigma = U^H H.$$  (23)

The destruction of the orthogonal property will cause non-zero values of the off-diagonal entries of the diagonal matrix $\Sigma$. Such the non-zero off-diagonal values will result in the interferences among all antennas, and then the system performance will be degraded. Hence, the destruction of the orthogonal property should be carefully handled. In our SVD design, this property is destroyed by quantization error and the inaccurate deflation processes with finite precision. Especially, error propagation induced by the deflation processes may cause a fatal error to the orthogonal property. Take two left singular vectors as an example,

$$\langle u_i, u_j \rangle = \varepsilon, \quad \forall i \neq j.$$  (24)

If $u_i$ and $u_j$ have perfect orthogonal property, $\varepsilon$ should be equal to zero. If the orthogonal property of $u_i$ and $u_j$ is destroyed by quantization error, the value of $\varepsilon$ is close to the accuracy which fixed-point implementation can represent. Nevertheless, error propagation induced by the deflation processes may lead $\varepsilon$ to be hundred times of the system accuracy. The destruction of the orthogonal property caused by quantization error cannot be prevented. Therefore, we propose an operation called orthogonal reconstruction (OR) to eliminate the destruction caused by the deflation processes and improve the system performance.

Assume that we already have the $d$ left singular vectors $u_1, u_2, \ldots, u_d$ after the update and deflation processes. Note that the first left singular vector $u_1$ does not suffer from the errors caused by the deflation process. For other left singular vectors $u_i$ with $i > 1$, we eliminate the inaccurate remaining part from $u_i$ to $u_{i-1}$ by applying Gram-Schmidt technique as follows

$$u_{\text{OR},i} = u_i,$$

$$\hat{u}_i = u_i - \sum_{j=1}^{i-1} \langle u_j, u_{\text{OR},j} \rangle \cdot u_{\text{OR},j},$$

$$u_{\text{OR},i} = \hat{u}_i \| \hat{u}_i \|.$$  (25)

where $i = 2, 3, \ldots, d$, and $u_{\text{OR},i}$ is the $i$th left singular vector after the orthogonal reconstruction process. Note that for right
singular vectors, we only need to replace $\mathbf{u}_i$ with $\mathbf{v}_i$ in (25). After applying orthogonal reconstruction to all singular vectors, the most interferences caused by the inaccurate deflation processes can be eliminated.

For the architecture of orthogonal reconstruction, we have to modify Gram-Schmidt unit in Fig. 8. We rewrite the second equation in (25) into a more compacted form:

$$
\hat{\mathbf{u}}_i = \left[ \mathbf{u}_i, \mathbf{u}_{OR,i}, \ldots, \mathbf{u}_{OR,i-1} \right] \\
\times \left[ \begin{array}{c}
\mathbf{u}_i^H \\
\mathbf{u}_{OR,i}^H \\
\vdots \\
\mathbf{u}_{OR,i-1}^H
\end{array} \right]^{-1}
$$

(26)

The operation in (26) can be executed by two successive matrix-vector multipliers. Based on (17), (25), and (26), Fig. 14 shows the block diagram of Gram-Schmidt unit with some modification. The multiplexer is used for considering structures with data interleaving scheme for throughput enhancement in hardware consideration.

Before the system simulation, we have to determine the maximum iteration number in (10) and (11) according to the Appendix. The proposed adaptive step size makes the iterative updating have both fast and stable convergence. We compare four step sizes: 1) $\mu_i(n) = 0.05/\lambda_i(n)$ in [19], 2) the proposed $\mu_i(n) = 0.5/\lambda_i(n)$, 3) the proposed $\mu_i(n) = 0.75/\lambda_i(n)$, and 4) the near-optimal step size in (11). Assume that the entries of a channel matrix $\mathbf{H}$ are independent and identically distributed (i.i.d.) according to $\mathcal{CN}(0, 1)$, where $\mathcal{CN}(0, 1)$ denotes the complex Gaussian distribution with independent real and imaginary parts distributed according to $\mathcal{N}(0, 1)$. We define the instantaneous error $e(n)$ as

$$
e(n) = \left\| \mathbf{w}_i(n) - \mathbf{w}_{opt} \right\|_2
$$

(27)

where $\mathbf{w}_{opt}$ is the optimal vector of $\mathbf{H}$ in the first update process. We only consider the first update process since the subsequence update processes have similar results. Fig. 15 compares the convergence rate of different step sizes over 1000 independent channel realizations. The proposed adaptive step size is not only guaranteed to have stable convergence rate but also much faster than the step size $0.05/\lambda_i(n)$ in [19]. Notice that at the early stage of total iterations, the proposed step size has the faster convergence rate than the near-optimal step size. It is reasonable since the near-optimal step size has the optimal convergence speed only when the current vector is close to the optimal vector.

**B. Effect of the Subcarrier Inherit Scheme**

We apply the proposed reconfigurable adaptive SVD engine to the IEEE 802.11n applications. To determine the word-lengths in our design, we performed extensive floating point simulation and dynamic range analysis. We list the word-lengths of some key signals used in the fixed-point simulation and chip implementation are shown in the form (integer, fractional). The word-length of real part or imaginary part of each entry of $\mathbf{H}$, $\mathbf{R}$, $\mathbf{w}_i$, $\lambda_i$, $\mathbf{u}_i$, $\mathbf{v}_i$, and $\sigma_i$ is (3, 7), (6, 14), (4, 12), (6, 26), (1, 7), (1, 7), and (4, 13), respectively.

To observe the effect of the subcarrier inherit scheme, we consider the channel model E [17], [25] in a 128-subcarrier 4x4 system. Assume that the division-free adaptive step size with the early termination scheme is applied. When the subcarrier inherit scheme (SIS) has been included and excluded, Table III and Table IV show the averaged required iteration number in updating each pair $(\mathbf{w}_i, \lambda_i)$ for $\mu_i(n) = 0.5/\lambda_i(n)$ and $\mu_i(n) = 0.75/\lambda_i(n)$, respectively. Note that with the partial update scheme, updating the last pair $(\mathbf{w}_4, \lambda_4)$ is unnecessary. As this shows, utilizing the subcarrier inherit scheme has the significant effect of reducing the total iterations by 19.1% and 21.5% for $\mu_i(n) = 0.5/\lambda_i(n)$ and $\mu_i(n) = 0.75/\lambda_i(n)$, respectively.

**C. System Simulation**

Before the system simulation, we have to determine the maximum iteration number in the update process. In Table III and Table IV, the first update process requires more iterations.

![Fig. 15. The convergence rate of different step sizes in the first update process.](image-url)
If $\mu(n) = 0.5/\tilde{\lambda}(n)$, the mean and the standard deviation of the required iteration numbers in the first update process are 26.5 and 9.5. Therefore, in order to guarantee that almost all pairs $(\mathbf{w}_i, \tilde{\lambda}_i)$ are converged, we choose the maximum iteration number in each update process as 64 which is roughly equal to the sum of the mean and the four-times standard deviation. Then, the proposed SVD engine is applied to the IEEE 802.11n PHY system [17]. The performance metric is bit error rate (BER). The simulation environment settings are listed below. 1) AWGN, Ch E (nLOS) channels [25], 4 spatial streams. 2) Assume perfect channel state information is obtained. 3) MIMO Technique: SVD. 4) Signal constellation: 4-QAM, 16-QAM, and 64-QAM. 5) FFT (IFFT) size: 128. 6) Code rate 1/2 convolutional code with constraint length 7, generator polynomials [133 171] [26]. 7) Block interleaving is used.

The simulation result is shown in Fig. 16, Fig. 17, and Fig. 18. Our target BER is $10^{-5}$. In the floating-point view, the proposed SVD design has no performance loss compared with the ideal SVD. Without orthogonal compensation, the proposed SVD fixed-point design only works well at 4-QAM with a performance loss of 0.4 dB compared with the ideal SVD. If orthogonal reconstruction is applied to our SVD fixed-point design, there is no performance loss at 4-QAM and 16-QAM. Besides, in the signal constellation of 64-QAM, our SVD fixed-point design has little performance loss of 0.6 dB compared with the ideal SVD.

**D. Chip Implementation**

For a baseline design, we adopt $\mu(n) = 0.5/\tilde{\lambda}(n)$ and the subcarrier inherit scheme is not applied. The memory banks of the channel matrices and SVD results are describe as follow. Assume 10-bit precision of each real or imaginary number in the channel matrix $\mathbf{H}$ is given, 320 bits are required for storing one 4x4 complex matrix. To avoid memory access collision, total storages of 16 channel matrices are divided into 4 single-port memory banks. The columns of one channel matrix are stored in 4 different memory banks so that we are able to access one complete channel matrix per cycle. In summary, 4 16 entries $\times$ 80 bits memory banks are required as channel matrix storage in our design.

There are two memory banks in the memory unit in Fig. 3 to store the elements of $\mathbf{U}$, $\mathbf{V}$, and $\Sigma$. Bank 1 is designed for $\mathbf{U}$ and $\mathbf{V}$. We use 16-bit precision for each element in $\mathbf{U}$ and $\mathbf{V}$. Two elements are stored in each entry of memory bank 1. Total entries required for bank 1 is 256, 16 elements $\times$ 16 matrices, and the overall size is 256 entries $\times$ 32 bits. Bank 2 is designed for storing $\Sigma$, the singular values, and the wordlength of each singular value is 17 bits. Total entries required for bank 2 is 64,
4 elements × 16 matrices, and the overall size is 64 entries × 17
bits. In addition, the matrix-to-matrix multiplication is
performed by the matrix-to-vector multiplier in 4 cycles.
The chip is fabricated in UMC 90 nm 1P9M Low-K CMOS
technology and measured

Table V
CHIP SUMMARY

<table>
<thead>
<tr>
<th>Technology</th>
<th>UMC 90nm 1P9M Low-K Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO/core VDD</td>
<td>3.3V / 1.0V</td>
</tr>
<tr>
<td>Core Area</td>
<td>1.475 mm × 1.475 mm</td>
</tr>
<tr>
<td>Die Area</td>
<td>2.22 mm × 2.22 mm</td>
</tr>
<tr>
<td>Gate Count</td>
<td>543.9k</td>
</tr>
<tr>
<td>Frequency</td>
<td>101.2 MHz (max)</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>125mW @101.2 MHz</td>
</tr>
</tbody>
</table>

Fig. 19. Die Photo of the proposed reconfigurable adaptive SVD engine design.

with Tektronix pattern generator TLA 715 and logic analyzer
TLA 5203. Fig. 19 shows the die photo of the fabricated chip
design. The chip feature is summarized in Table V. The core
size is 1.475 mm × 1.475 mm. The number of total gate counts
is 543.9k. The die size is 2.22 mm × 2.22 mm giving a total area
of 4.93 mm². The maximum operating frequency is measured
101.2 MHz and the total power consumption is measured 125
mW for the 4×4 SVD operations. In order to consider reduction
of power consumption, we can reduce the core supply voltage
to 0.65V as shown in Fig. 20. The corresponding maximum
operating frequency and power consumption are 43.48 MHz
and 22.1 mW, respectively.

![Graph of measured frequency and power vs. core supply voltage](image)

Fig. 20. Measured frequency and power of the chip design.

For comparison, we use two performance indices. First, the
throughput is defined by the number of channel matrices that
the SVD engine can deal with per second

\[
\text{Throughput} = \frac{\text{Number of processed channel matrices}}{\text{Time (sec)}}. \quad (28)
\]

In the worst updating cases of proposed SVD operation,
there are 64 iterations for each singular pair updating without
early termination scheme and do not have to update the last
singular pair. We need 64 iteration per singular pair × (4-1)
singular pairs × 16 matrices = 3072 cycles, and extra 308 cycles
for other operations. The equivalent throughput is derived as
16/(3380 cycles × (1/101.2MHz)) = 479.05k-matrices/sec. For
16 mm channel matrices, the total cycles required are (16 × 64
× (m-1) + 308) cycles. There are 308, 1332, 2356, and 3380
cycles required when processing 16 1×1, 2×2, 3×3, and 4×4
matrices respectively. In other words, the equivalent
throughputs are 5.3M, 1.2M, 687k, and 479k-matrices/sec for
1×1, 2×2, 3×3, and 4×4 matrices respectively.

Then the power efficiency can be expressed as

\[
\text{Power Efficiency} = \frac{\text{Throughput (k)}}{\text{Power consumption (mW)}}. \quad (29)
\]

The technology scaling of power from 180nm@1.8V to
90nm@1.0V is given by \( P_{90} = P_{180} \times \frac{(C_{90}/C_{180})^2}{(V_{90}/V_{180})^2} = P_{180} \times 0.5 \times (1.0/1.8)^2 = P_{180} \times 0.1543 \). The proposed
reconfigurable adaptive SVD engine design is compared with
other designs as shown in Table VI. [19] proposed an SVD chip
without the need of CSI. The block-type pilots are utilized in
the IEEE 802.11n systems for training symbol-based channel
estimation of each subcarrier. The least-square (LS) and
minimum-mean-square-error (MMSE) techniques [30] are
widely used for channel estimation when training symbols are
available. The complexity is fairly low owing to no matrix
inversion required in channel estimation with pre-defined

Table VI
COMPARISON TABLE

<table>
<thead>
<tr>
<th>Support Antenna Configurations</th>
<th>JSSC’07 [19]</th>
<th>ACSSC’07 [27]</th>
<th>ISCAS’08 [28]</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>U and Σ</td>
<td>U, Σ and V</td>
<td>V and Σ</td>
<td>U, Σ and V</td>
</tr>
<tr>
<td>Technology</td>
<td>90 nm</td>
<td>180 nm</td>
<td>180 nm</td>
<td>90 nm</td>
</tr>
<tr>
<td>Core Size</td>
<td>3.63 mm²</td>
<td>0.41 mm²</td>
<td>0.41 mm²</td>
<td>2.17 mm²</td>
</tr>
<tr>
<td>Gate Count</td>
<td>980k</td>
<td>42.3k</td>
<td>42.3k</td>
<td>549.9k</td>
</tr>
<tr>
<td>Frequency</td>
<td>100 MHz</td>
<td>133 MHz</td>
<td>149 MHz</td>
<td>101.2 MHz</td>
</tr>
<tr>
<td>Power</td>
<td>34mW@0.4 V</td>
<td>160mW@1.8 V</td>
<td>N/A</td>
<td>125mW@1V</td>
</tr>
<tr>
<td>Throughput</td>
<td>50k</td>
<td>86.4k</td>
<td>303k</td>
<td>479k*</td>
</tr>
<tr>
<td>Power Efficiency</td>
<td>1.47</td>
<td>3.50</td>
<td>N/A</td>
<td>3.83</td>
</tr>
</tbody>
</table>

* Consider 4×4 SVD operation.

The power consumptions of the proposed SVD with 1×1, 2×2 and 3×3 matrices are about
12mW, 33mW and 74mW, respectively.
orthogonal training sets [17]. SVD in [19] only supports the 4x4 antenna system and implements the $U\Sigma$ algorithm which is not complete for SVD. The overall computational complexity of the SVD in [19] is proportional to the iteration number required which is about 500. By applying the proposed adaptive step size and partial update schemes in our proposed design, the iteration number required per matrix in our design is $3380/16 \approx 212$ at most. In addition, the average iteration number can be further reduced by 20% with the proposed subcarrier inherit scheme as shown in Table III and IV. [28] can be considered as an improved design of [27], but it only computes $V$ and $\Sigma$ which are partial of SVD outputs. Our design is able to handle 16 4x4 channel matrices at the same time.

Compared with other related works, only our work can support all antenna configurations in a MIMO system. Among all designs, our SVD chip has the highest throughput and power efficiency in the 4x4 SVD operations. In addition, the chip result shows that in an 802.11n system with 128 subcarriers, the average latency of our SVD chip is only 0.33% of the WLAN coherence time. Therefore, our SVD engine design is very suitable for high-throughput wireless communication applications.

In order to effectively enhance the throughput, we can use larger adaptive step size and apply the subcarrier inherit scheme to our SVD engine. First, we replace $\mu_i(n) = 0.5\lambda_i(n)$ with $\mu_i(n) = 0.75\lambda_i(n)$. This costs four additional complex adders in hardware implementation. Second, by applying the subcarrier inherit scheme, the registers in the update state can hold the converged values of the previous subcarriers until their adjacent subcarriers’ channel information comes. Hence, additional multiplexers are required in hardware implementation. If $\mu_i(n) = 0.75\lambda_i(n)$ and the subcarrier inherit scheme is applied, the mean and the standard deviation of the required iteration numbers in the first update process are 14.4 and 5.6, respectively. Therefore, we can choose the maximum iteration number in each update process as 36 which is roughly equal to the sum of the mean and the four-times standard deviation. Notice that for the first 16 subcarriers’ SVD operations, the maximum iteration number in each update process should be bigger since no additional information could speed up the convergence time. With this scenario, the throughput of our SVD engine can be enhanced to 850k with little extra hardware cost.

We used the clock gating scheme to turn off the unused multipliers with smaller channel matrices. The power consumption is not directly related to the operating cycles, but related to the executed operation per cycle in average. The main operation in the proposed SVD algorithm is matrix-to-vector multiplication whose complexity is proportional to $N^2$, where $N$ is the length of the vector. Owing to the leakage power and other common operations in different matrix sizes, the power consumption of 1x1–4x4 matrices are 12, 33, 74, and 125mW, respectively. The corresponding power consumptions of processing non-square matrices are close to that of square matrices with size of min(row, col.), where row and col. are the numbers of rows and columns of the channel matrices.

In summary, a reconfigurable SVD for different antenna sets and deriving all singular vectors are required for the application to IEEE 802.11n systems. The throughput requirement is also high. Compared with the referenced work in [19], our SVD engine is able to achieve the goals mentioned above. For the throughput consideration, we proposed the adaptive step size, partial update scheme and subcarrier inherit scheme to accelerate the overall processing. The throughput and power efficiency is about 9 times and 2.6 times than that in [19], respectively. The throughput improvement with subcarrier inherit scheme is about 20% as shown in Table III and IV. The proposed design with OR scheme is able to be 4dB better at least compared with the design without OR scheme as shown in Fig.17 and 18.

VII. CONCLUSION

This paper presents a reconfigurable adaptive SVD engine design for MIMO-OFDM systems. The proposed architectural design techniques can lower the computational complexity, effectively reduce the decomposing latency, and support all antenna configurations in a MIMO system. These design strategies enable the usage of SVD to be effectively applied to the high-throughput wireless communication applications. Our SVD engine is implemented in UMC 90-nm CMOS technology for the application of IEEE 802.11n systems with sixteen antenna configurations. The proposed SVD engine achieves a higher throughput rate than other related works. Moreover, the chip result shows that for an 802.11n system, the average latency of our SVD engine is only 0.33% of the WLAN coherence time. Therefore, the proposed SVD engine is very suitable for the high-throughput MIMO-OFDM applications.

APPENDIX

We show the detailed derivations of (10) and (11). Assume that the matrix $R \in \mathbb{C}^{d \times d}$ is a positive semidefinite matrix. The eigenvalue decomposition of $R$ can be expressed as

$$R = U\Lambda U^H,$$

(A.1)

where $\Lambda$ is a $d \times d$ matrix with only real and non-negative main diagonal entries. The entry $(i, i)$ of $\Lambda$ denotes the $i$th largest eigenvalue $\lambda_i$, with $i = 1, 2, \ldots, d$. The $i$th column vector $u_i$ in $U$ is called the $i$th eigenvector corresponding to the $i$th largest eigenvalue $\lambda_i$.

Consider the objective function $J(w)$:

$$J(w) = \frac{1}{2} w^H R w - \frac{1}{4} (w^H w)^2.$$  

(A.2)

In [20], the authors have proved that all the stationary point of $J(w)$ are eigenvectors of $R$ with magnitude being the square root of the corresponding eigenvalue of $R$. Besides, if the dominant eigen pair $\sqrt{\lambda_1} u_1$ is of multiplicity one, the dominant eigen pair is the global maximum point of $J(w)$.

Take the gradient of $J(w)$, we have

$$\nabla_w J(w) = R w - (w^H w) w.$$  

(A.3)
To maximize the objective function $J(w)$, it is straightforward to apply the steepest-descent techniques [9]. The update formula is given by

$$w(n+1) = w(n) + \mu \cdot \nabla_w J(w),$$

where $\mu$ is the step size. Generally speaking, the value of the step size directly impacts the convergence speed, stability, and accuracy of the adaptive algorithms. Since the objective function $J(w)$ is a fourth-order function in $w$, the analysis of the step size is complicated. Hence, we will give a loose bound by approximating $J(w)$ to a quadratic function around the optimal point $\sqrt{\lambda_1}u_1$. By invoking the second-order Taylor series expansion of $J(w)$ around the optimal point $\sqrt{\lambda_1}u_1$, $J(w)$ can be approximated by

$$J(w) = J(\sqrt{\lambda_1}u_1) + \frac{1}{2}(w - \sqrt{\lambda_1}u_1)^T \nabla^2_w J(\sqrt{\lambda_1}u_1)(w - \sqrt{\lambda_1}u_1),$$

where $\nabla^2_w J(w)$ is the Hessian of $J(w)$ which can be expressed as

$$\nabla^2_w J(w) = R - \left( w^H w \right) I - 2w w^H.$$

By substituting the optimal point $\sqrt{\lambda_1}u_1$ into (A.6), we obtain

$$\nabla^2_w J(\sqrt{\lambda_1}u_1) = U \Lambda U^H - \lambda_1 U U^H - 2 \lambda_1 u_1 u_1^H,$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$.

Hence, (A.20) provides a useful bound for the stability or convergence of the adaptive algorithm. To analyze the convergence speed of the adaptive algorithm, we define a time constant $\tau_k$ as the number of iterations required for $c_k(n)$ to decay to $1/e$ of its initial value $c_k(0)$, that is,

$$c_k(n) = e^{-\frac{n}{\tau_k}} c_k(0), \quad k = 1,2,\ldots,d.$$

From (A.16) and (A.21), the time constant $\tau_k$ is expressed as

$$\tau_k = \frac{1}{\ln |1 - \mu \lambda_k|}, \quad k = 1,2,\ldots,d.$$
and

\[ \tau_k \geq \tau_{k+1}, \quad (A.25) \]

where \( k = 2, 3, \ldots, d \). According to (A.24), \( \{\tau_k\}_{k=2}^{d} \) are the decreasing curves in the convergence region. From (A.25), given a value of \( \mu^* \), the maximal time constant is either \( \tau_1 \) or \( \tau_2 \).

Therefore, we have to find a good step size to minimize the maximal time constant. This condition is occurred at \( \tau_1 = \tau_2 \), that is,

\[ \frac{-1}{\ln|\|\mu^*\| - \mu_0|} = \frac{-1}{\ln|\|\mu^*\| - \mu_1|}. \quad (A.26) \]

By employing (A.7) and (A.26) and solving for \( \mu \), we have

\[ \mu_{opt} = \frac{2}{3\lambda_1 - \lambda_2}. \quad (A.27) \]

It should be noted that \( \mu_{opt} \) is a near-optimal step size since \( \tilde{J}(\mathbf{w}) \) in (A.5) is an approximate function to describe \( J(\mathbf{w}) \) around the optimal point. As a result, (A.27) provides a good guideline in choosing a proper step size of the adaptive algorithm.

REFERENCES


Ting-Jyun Jheng received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 2007, and the M.S. degree from National Taiwan University, Taipei, in 2009, both in electronic engineering. He is currently an engineer with MediaTek Inc., Hsinchu, Taiwan.

His research interests include the design of very large-scale integration architectures and circuits for digital signal processing and communication systems.

An-You (Andy) Wu (S’91–M’96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1987, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1992 and 1995, respectively, all in electrical engineering.

From August 1995 to July 1996, he was a Member of Technical Staff (MTS) at AT&T Bell Laboratories, Murray Hill, NJ, working on high-speed transmission IC designs. From 1996 to July 2000, he was with the Electrical Engineering Department, National Central University, Taiwan. In August 2000, he joined the faculty of the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, where he is currently a Professor. His research interests include low-power/high-performance VLSI architectures for DSP and communication applications, adaptive/multirate signal processing, reconfigurable broadband access systems and architectures, and SoC platform for software/hardware co-design.

Dr. Wu was a recipient of the A-class Research Award from National Science Council for four times. He has served on many technical program committees of IEEE international conferences, and was an Associate Editor of IEEE TRANSACTIONS.

Conference paper information:

A list of differences:
◆ In Section III, the subcarrier inherit scheme is proposed to speed up the convergence rate.
◆ We add a convergence rate analysis of the proposed adaptive SVD algorithm in Appendix.
◆ Detailed architectural designs are illustrated in Section IV.
◆ For fixed-point implementation, we propose an orthogonal reconstruction scheme to enhance error rate performance in Section V.
◆ The convergence rate comparison of different adaptive step sizes and the effect of the subcarrier inherit scheme are shown in Section VI-A and VI-B, respectively.
◆ In the system simulation of Section VI-C, we also consider the constellations of 16-QAM and 64-QAM.
◆ In Section VI-D, we also add two related works for chip comparisons.