HOKKAIDO UNIVERSITY

Title	VLSI Implementation of a Scalable Pipeline MMSE MIMO Detector for a 4 x 4 MIMO-OFDM Receiver
Author(s)	Yoshizawa, Shingo; Ikeuchi, Hirokazu; Miyanaga, Yoshikazu
Citation	IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E94-A(1), 324-331 https://doi.org/10.1587/transfun.E94.A.324
Issue Date	2011-01-01
Doc URL	https://hdl.handle.net/2115/47066
Rights	copyright © 2011 IEICE
Type	journal article
File Information	ToFECCS94A-1_324-331.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

IEICE TRANS. FUNDAMENTALS, VOL.E94–A, NO.1 JANUARY 2011

PAPER

VLSI Implementation of a Scalable Pipeline MMSE MIMO Detector for a 4 × 4 MIMO-OFDM Receiver

Shingo YOSHIZAWA[†a)], *Member*, **Hirokazu IKEUCHI**[†], *Nonmember*, *and* **Yoshikazu MIYANAGA**[†], *Member*

SUMMARY MIMO-OFDM performs signal detection on a subcarrier basis which requires high speed computation in MIMO detection due to its large computational cost. Conventional designs in a MIMO detector increase processing time in proportion to the number of subcarriers and have difficulty in real-time processing for large numbers of subcarriers. A complete pipeline MMSE MIMO detector presented in our previous work can provide high speed computation. However, it tends to be excessive in a circuit scale for small numbers of subcarriers. We propose a new scalable architecture to reduce circuit scale by adjusting the number of iterative operations according to various types of OFDM system. The proposed detector has reduced circuit area to about 1/2 to 1/7 in the previous design with providing acceptable latency time.

key words: *wireless communications, MIMO-OFDM, MIMO detection, MMSE*

1. Introduction

Multiple-input multiple-output orthogonal frequency multiplexing (MIMO-OFDM) is effective in enhancing communication capacity or reliability and has become a key technology in current wireless communications. MIMO-OFDM is adopted in IEEE802.11n [1] and IEEE802.11ac [2] (known as standardization of next wireless LAN system). In MIMO technology, spatial division multiplexing (SDM) can achieve high data rates by use of multiple transmit and receive antennas. MIMO transmission of four streams will become practical use as IEEE802.11n option supports a 4×4 MIMO transmit mode which provides a maximum 600-Mbps data rate.

Signal detection in a MIMO receiver, called MIMO detection, needs high speed computation due to its large computational cost. Moreover, packet MIMO-OFDM estimates channel matrices at a preamble and computes inverted matrices in a short period before receiving data symbols. Otherwise, the receiver is forced to have long processing latency with large memory buffers or give up real-time processing. Hardware implementation of MIMO detection is one important issue in current wireless communication systems. Algorithms for MIMO detection are classified into linear detection [3]–[6] ordered successive interference cancellation (OSIC) [7], [8], and maximum-likelihood (ML) detection [9], [10]. They trade off between complexity and MIMO detection performance.

Since OFDM performs MIMO detection on a subcarrier basis, its computational cost is proportional to the number of subcarriers. A MIMO-OFDM receiver requires considerable throughput performance even for linear detection. We focus on hardware implementation of linear detection. Recent research has tackled linear detectors in zero-forcing (ZF) [3], [4] or minimum mean squared error (MMSE) [5], [6] for 4×4 MIMO-OFDM. For the MMSE criterion, QR decomposition and Sherman-Morrison formula algorithms have been adopted in a MMSE MIMO detector in [5], [6]. Their detectors suffer from real-time processing for large numbers of subcarriers because their architectures rely on a large number of iterative operations. On the other hand, we have presented a complete pipeline MMSE MIMO detector based on Strassen's matrix inversion in [11], [12]. This detector can make use of concurrent and pipeline processing and has systematic matrix computation suitable for hardware implementation. The processing time of the complete pipeline detector is 150 times faster than the other detectors on the condition of 512 subcarriers [12]. However, it has the drawback in circuit area. It tends to be excessive in a circuit scale for small numbers of subcarriers.

Note that the number of OFDM subcarriers widely ranges from 64 to 2048 such as IEEE802.11n/ac and IEEE802.16e/m wireless systems. The detectors based on iterative architecture [5], [6] cause the shortage of throughput performance for large numbers of subcarriers. The detector based on complete pipeline architecture [11] tends to be excessive in a circuit scale for small numbers of subcarriers. We present a new scalable pipeline MMSE MIMO detector which attains both circuit reduction and real-time processing and is effective for various types of OFDM system. The key point is to adjust the number of iterations according to OFDM parameters. The acceptable latency time is determined by OFDM parameters such as a signal sampling rate, a guard interval (GI) length and a subcarrier count. Proposed scalable pipeline architecture optimizes hardware structure according to the number of iterative steps. The scalable pipeline detector changes a matrix operation flow for each step and realizes circuit reduction, which is based on dynamically reconfigurable architecture. The VLSI implementation of the scalable pipeline MMSE MIMO detector and its circuit performance comparison with the conventional detectors are also reported in this paper.

Manuscript received December 4, 2009.
Manuscript revised July 2, 2010.

[†]The authors are with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo-shi, 060-0814 Japan.
a) E-mail: yosizawa@csm.ist.hokudai.ac.jp

DOI: 10.1587/transfun.E94.A.324

Copyright © 2011 The Institute of Electronics, Information and Communication Engineers

2. MIMO Detection and Timing Requirement

For a MIMO-OFDM system with M_T transmit antennas, M_R receive antennas, and N data subcarriers, a MIMO channel for k-th subcarrier is given by $\boldsymbol{H}_{[k]}$ with a $M_T \times M_R$ matrix. In MIMO detection, a received signal vector $\boldsymbol{y}_{[k,t]}$ for t-th data symbol is described by

$$\boldsymbol{y}_{[k,t]} = \boldsymbol{H}_{[k]}\boldsymbol{s}_{[k,t]} + \boldsymbol{n}_{[k,t]}, \tag{1}$$

where $\boldsymbol{s}_{[k,t]}$ and $\boldsymbol{n}_{[k,t]}$ represent a transmitted signal vector and white Gaussian noise, respectively. $\boldsymbol{H}_{[k]}$ is estimated from training symbols. The purpose of MIMO detection is to extract $\boldsymbol{s}_{[k,t]}$ from the received vector. The linear detection inverts the channel matrix using ZF or MMSE criterion. The ZF computes the inverse matrix of $\boldsymbol{H}^{-1}_{[k]}$. The matrix inversion of the MMSE is computed as

$$\boldsymbol{G}_{[k]} = (\boldsymbol{H}^H_{[k]}\boldsymbol{H}_{[k]} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{H}^H_{[k]}, \tag{2}$$

where $\boldsymbol{\sigma}^2$ indicates a noise variance and $(\cdot)^H$ denotes a function of Hermitian transpose. The matrix inversion of (2) is called preprocessing. The MIMO decoding extracts the approximate transmit vectors as

$$\hat{\boldsymbol{s}}_{[k,t]} = \boldsymbol{G}_{[k]}\boldsymbol{y}_{[k,t]}. \tag{3}$$

The timing chart of MIMO detection in 4×4 MIMO-OFDM (i.e., M_T=4, and M_R=4) is illustrated in Fig. 1. On receiving the last training symbol, a receiver starts computing MIMO channel matrices for all subcarriers. The preprocessing can be executed after computing each MIMO channel matrix. Since the data symbols follow the training symbols in packet mode OFDM, the receiver should complete the preprocessing before receiving the data symbols. Otherwise, the receiver has long processing delay with large memory buffers or gives up real-time processing. We define acceptable latency time as the sum of FFT duration and GI length, e.g., $T_{FFT}+T_{GI}$, which is used as a measure of real-time processing in this paper. For instance, the IEEE802.11n PHY frame has parameters of N=108, T_{FFT}=3.2 μs, and T_{GI}=0.8 μs. The acceptable latency time of the preprocessing is 4.0 μs in this case.

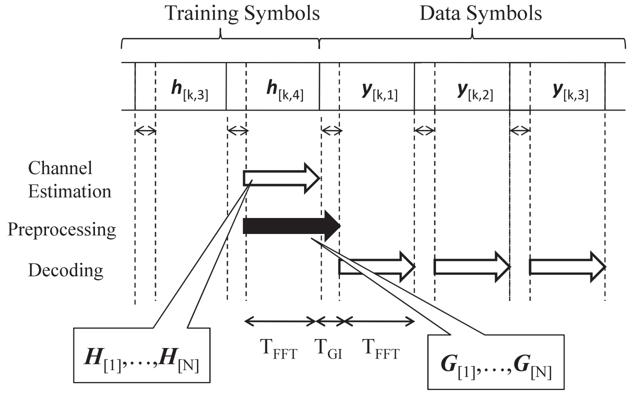


Fig. 1 Timing chart of MIMO detection.

3. Algorithms and Architectures

3.1 Algorithms

The hardware implementation of the preprocessing in (2) has been discussed in recent research due to its considerable calculation cost. QR decomposition and Sherman-Morrison formula as matrix inversion lemma are used for complexity reduction of matrix inversion [5], [6]. In our previous work, we have adopted Strassen's algorithm [11]. We briefly describe algorithms of QR decomposition by modified Gram-Schmidt and Strassen's matrix inversion for discussing VLSI architectures for a MMSE MIMO detector. In QR decomposition, a matrix $\boldsymbol{A}$ is divided into an orthogonal $\boldsymbol{Q}$ and a right triangular matrix $\boldsymbol{R}$ as $\boldsymbol{A}=\boldsymbol{QR}$. The procedure of the modified Gram-Schmidt for a 4 × 4 square matrix is summarized below:
- Definition

$$\boldsymbol{A} = [\boldsymbol{a}_1\ \ \boldsymbol{a}_2\ \ \boldsymbol{a}_3\ \ \boldsymbol{a}_4] \tag{4}$$
$$\boldsymbol{Q} = [\boldsymbol{q}_1\ \ \boldsymbol{q}_2\ \ \boldsymbol{q}_3\ \ \boldsymbol{q}_4] \tag{5}$$
$$\boldsymbol{R} = [R_{ij}]\ \ (1 \le i \le 4,\ \ 1 \le j \le 4) \tag{6}$$

- Iteration

```
for $i$=1 to 4
    $R_{ii} = \|\boldsymbol{a}_i^{(i)}\|^2$                          (7)
    $\boldsymbol{q}_i = \boldsymbol{a}_i^{(i)}/R_{ii}$                          (8)
    for $j$=$i$+1 to 4
        $R_{ij} = \boldsymbol{q}_i^T\boldsymbol{a}_j^{(i)}$                      (9)
        $\boldsymbol{a}_j^{(i+1)} = \boldsymbol{a}_j^{(i)} - \boldsymbol{q}_i R_{ij}$           (10)
    end
end.
```

To compute (2), matrix multiplication of $\boldsymbol{H}$ and inversion of triangular matrix $\boldsymbol{R}$ are additionally required (See Ref. [14] for the details). Strassen's algorithm for matrix inversion divides a square matrix into equal small matrices [13] and applies analysis solution. For a 4 × 4 matrix $\boldsymbol{\Omega}$, it is divided into 2 × 2 four submatrices and inversed, which can be expressed as

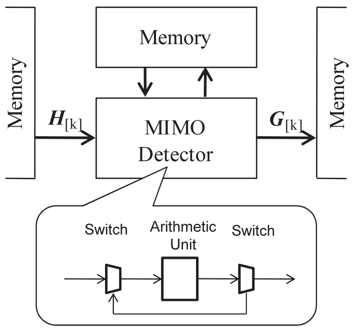
$$\boldsymbol{\Omega} = \begin{pmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{pmatrix} \tag{11}$$

$$\begin{aligned}\boldsymbol{\Omega}^{-1} &= \begin{pmatrix} \mu & -\omega_{11}^{-1}\omega_{12}\lambda^{-1} \\ -\lambda^{-1}\omega_{21}\omega_{11}^{-1} & \lambda^{-1} \end{pmatrix} \\ &= \begin{pmatrix} \omega'_{11} & \omega'_{12} \\ \omega'_{12} & \omega'_{22} \end{pmatrix}\end{aligned} \tag{12}$$

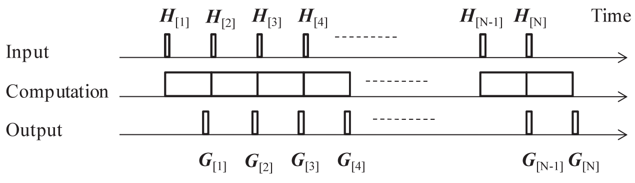
$$\lambda = \omega_{22} - \omega_{21}\omega_{11}^{-1}\omega_{12} \tag{13}$$

$$\mu = \omega_{11}^{-1} + \omega_{11}^{-1}\omega_{12}\lambda^{-1}\omega_{21}\omega_{11}^{-1}. \tag{14}$$

The common parts of ω_{11}^{-1}, $\omega_{11}^{-1}\omega_{12}$, $\omega_{21}\omega_{11}^{-1}$, and λ^{-1} contribute to complexity reduction. The MMSE criterion in (2)

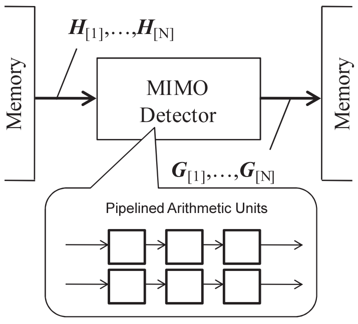


(a) Circuit structure

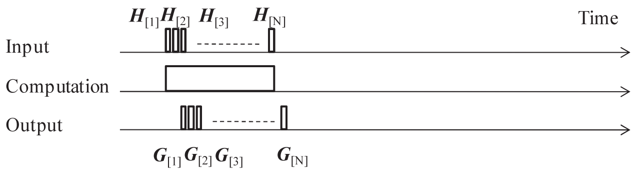

Time
Input
Computation
Output
H[1] H[2] H[3] H[4] H[N-1] H[N]
G[1] G[2] G[3] G[4] G[N-1] G[N]

(b) Timing chart

Fig. 2 Iterative architecture.



(a) Circuit structure



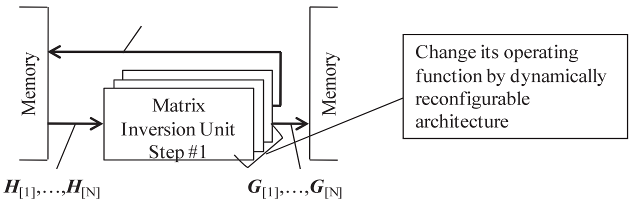
(b) Timing chart

Fig. 3 Complete pipeline architecture.

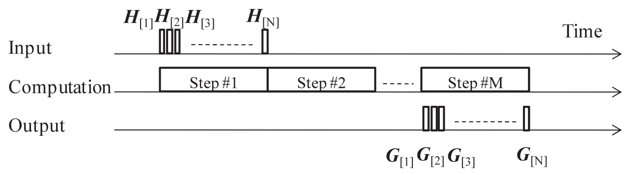
gives a complex conjugate symmetric at non-diagonal components for matrix inversion, i.e., $\omega_{12}=\omega_{21}^H$. It enables further complexity reduction by $\omega'_{12}=\omega'^H_{21}$. The whole procedure for computing (2) including matrix multiplications is explained in Sect. 4.

3.2 Architectures

Figures 2 and 3 show iterative and complete pipeline architectures for implementing the preprocessing in (2). The preprocessing should be computed for all N subcarriers. Iterative architecture repeats its computation by the number of subcarriers. Complete pipeline architecture operates a block


Memory
Memory
Matrix Inversion Unit Step #1
Change its operating function by dynamically reconfigurable architecture
H[1],…,H[N]
G[1],…,G[N]

(a) Circuit structure



(b) Timing chart

Fig. 4 Proposed scalable pipeline architecture.

of subcarriers which goes through all pipeline stages. Iterative architecture shares arithmetic units in most of the computations and realizes by a small circuit scale, which is shown in Fig. 2(a). Complete pipeline architecture has high throughput performance by increasing the number of simultaneously operating arithmetic units. It provides high speed computation and requires a large circuit scale, which is shown in Fig. 3(a). The throughput performance of complete pipeline architecture is hihger than that of iterative architecture. It causes a shorter processing time indicated by the timing chart in Figs. 2(b) and 3(b). QR decomposition is implemented by iterative architecture [6]. The modified Gram-Schmidt updates vectors of $\boldsymbol{q}_i$ and $\boldsymbol{a}_j^{(i+1)}$ in (8) and (10), respectively. Their updating operations can be implemented into shared arithmetic units in iterative architecture. Conversely, they are not suitable for complete pipeline architecture which makes use of concurrent and pipeline processing. The modified Gram-Schmidt updates and computes a vector or an element within a matrix. During this operation, a detector sets the other vectors and elements to idle states. Current and pipeline processing induces large memory buffers for these idle states in this situation. Strassen's algorithm is implemented by complete pipeline architecture [11], which enables concurrent and pipeline processing in the divided 2×2 submatrices shown in (12) and is suitable for its hardware implementation because of the systematic 2×2 matrix operations in (12)–(14).

Iterative architecture and complete pipeline architecture benefit in circuit scale and throughput performance, respectively. However, the following issues should be noticed:

- Processing time of iterative architecture is proportional to the number of subcarriers. It is difficult to keep the processing time which is less than the acceptable latency for large numbers of subcarriers.
- Circuit scale of complete pipeline architecture tends to be excessive for small numbers of subcarriers. If a detector has a long idle time waiting for data symbols after the preprocessing, it has room to reduce circuit scale by decreasing throughput performance.

Table 1 Properties of conventional and proposed architectures.

Architecture	Speed	Circuit Scale
Iterative [5], [6]	Low	Small
Complete Pipeline [11]	High	Large
Scalable Pipeline (Proposed)	Mid - High	Small - Mid

The number of OFDM subcarriers widely ranges from 64 to 2048 such as IEEE802.11n/ac and IEEE802.16e/m wireless systems. A MIMO detector should be optimized so as to attain both circuit reduction and real-time processing. We propose a new scalable pipeline architecture illustrated in Fig. 4. This architecture divides the preprocessing by multiple steps, where the block data of intermediate values at s-th step is indicated by the feedback of $\boldsymbol{P}^s_{[1]}$, $\boldsymbol{P}^s_{[2]}$,..., $\boldsymbol{P}^s_{[N]}$. The number of steps is determined so that processing time is less than acceptable latency time. The processing time can be estimated from OFDM parameters. A MIMO detector changes matrix operations by switching data paths for the steps, which is achieved by dynamically reconfigurable architecture. For small numbers of subcarriers, the scalable architecture increases the number of steps and provides small circuit scale. For large numbers of subcarriers, it increases throughput performance by connecting pipelined matrix operation units and performs the preprocessing in a few steps. The properties of conventional and proposed architectures are summarized in Table 1. Their comparison with the circuit implementations is discussed in Sect. 5. The design of a MMSE MIMO detector based on the scalable pipeline architecture is described in the next section.

4. Scalable Pipeline MMSE MIMO Detector

4.1 Design Procedure

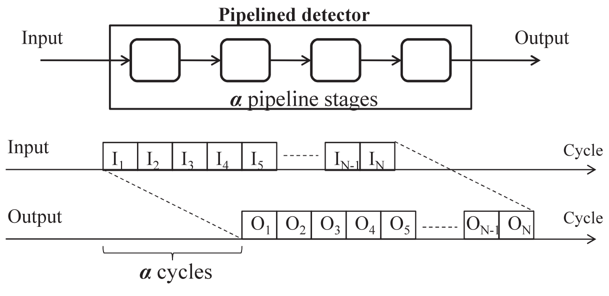
The design procedure of a scalable pipeline MMSE MIMO detector is described as follows:

(a) Determine the number of computation steps M according to OFDM parameters so that the processing time is approximately equal to the acceptable latency time.
(b) Divide 2 × 2 submatrix operations in Strassen's algorithm into M steps. The minimum numbers of 2 × 2 matrix operation units (e.g., matrix multiplier and matrix adder) can be determined by this work.
(c) Design a MMSE MIMO detector according to the numbers of 2×2 matrix operation units. Dynamically reconfigurable architecture is introduced to switch different matrix operation flows.

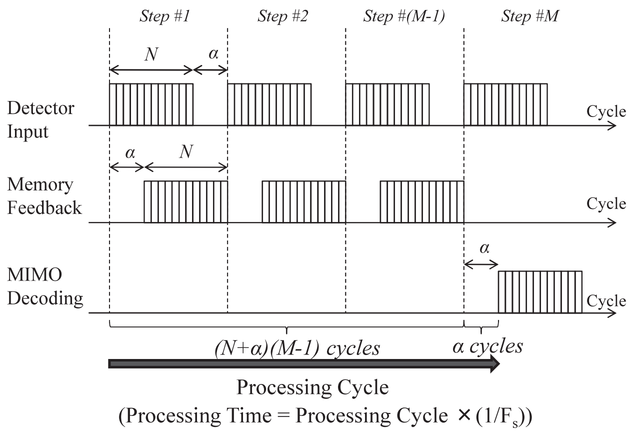
These steps are explained in Sects. 4.2, 4.3, and 4.4, respectively.

4.2 Number of Steps

Processing time of a scalable pipeline MMSE MIMO detector is explained by the timing chart in clock cycle unit shown in Fig. 5. When a MIMO detector has α pipeline stages, a block of data is delayed for α cycles, which is illustrated in



(a) Delay in a pipelined detector (α=4).



(b) Timing chart of a scalable pipeline MMSE MIMO detector.

Fig. 5 Processing time in a scalable MMSE MIMO detector.

Fig. 5(a). Next, we consider processing cycle in preprocessing from the timing chart in Fig. 5(b). From first to $(M-1)$-th steps, output data for all N subcarriers are fed into memory in $N+\alpha$ cycles. At the last step, the output data are directly used for the MIMO decoding shown in Fig. 1 without memory feedback. The processing time is given by the product of the processing cycle and a clock period, which is computed as

$$T_c = ((N + \alpha)(M - 1) + \alpha) \cdot (1/F_s), \tag{15}$$

where F_s donotes operating clock frequency. The acceptable latency is given by the sum of T_{FFT} and T_{GI} as shown in Fig. 1. The condition of T_C is expressed as

$$T_c \leq T_{FFT} + T_{GI}. \tag{16}$$

For instance, N=108, T_{FFT}=3.2 μs, and T_{GI}=0.8 μs assumes the OFDM parameters of IEEE802.11n standard with a 40-MHz channel. When a detector operates at 250-MHz clock frequency and has 12 pipeline stages,

$$((108 + 12) \cdot (M - 1) + 12) \cdot 1/(250 \times 10^6) \leq 3.2 \times 10^{-6} + 0.8 \times 10^{-6} \tag{17}$$

$$M \leq 9.23 \tag{18}$$

From the above calculation, 9-step computation is appropriate in this case. We treat the cases of 9-step and 2-step computations in the circuit, which are assumed in IEEE802.11n with a 40-MHz channel and IEEE802.11ac with an 80-MHz channel, respectively.

4.3 Strassen's Algorithm

As explained in Sect. 3, we use Strassen's algorithm in implementation of a MIMO detector. It divides a 4×4 matrix into four 2×2 submatrices and performs matrix inversion by analytic solution. The preprocessing in (2) can be expressed by the following equations:

$$\boldsymbol{b}_{11} = \boldsymbol{h}_{11}\boldsymbol{h}_{11} + \boldsymbol{h}_{12}\boldsymbol{h}_{12}^{*} + \sigma^2\boldsymbol{I} \tag{19}$$
$$\boldsymbol{b}_{12} = \boldsymbol{h}_{11}\boldsymbol{h}_{21}^{*} + \boldsymbol{h}_{12}\boldsymbol{h}_{22} \tag{20}$$
$$\boldsymbol{b}_{22} = \boldsymbol{h}_{21}\boldsymbol{h}_{21}^{*} + \boldsymbol{h}_{22}\boldsymbol{h}_{22} + \sigma^2\boldsymbol{I} \tag{21}$$
$$\boldsymbol{c}_1 = \boldsymbol{b}_{11}^{-1} \tag{22}$$
$$\boldsymbol{c}_2 = \boldsymbol{b}_{12}^{H}\boldsymbol{c}_1 \tag{23}$$
$$\boldsymbol{c}_3 = \boldsymbol{c}_2\boldsymbol{b}_{12} \tag{24}$$
$$\boldsymbol{c}_4 = \boldsymbol{b}_{22} - \boldsymbol{c}_3 \tag{25}$$
$$\boldsymbol{c}_5 = \boldsymbol{c}_4^{-1} \tag{26}$$
$$\boldsymbol{c}_6 = \boldsymbol{c}_2^{H}\boldsymbol{c}_5 \tag{27}$$
$$\boldsymbol{c}_7 = \boldsymbol{c}_6\boldsymbol{c}_2 \tag{28}$$
$$\boldsymbol{c}_8 = \boldsymbol{c}_1 + \boldsymbol{c}_7 \tag{29}$$
$$\boldsymbol{g}_{11} = \boldsymbol{c}_8\boldsymbol{h}_{11} - \boldsymbol{c}_6\boldsymbol{h}_{21} \tag{30}$$
$$\boldsymbol{g}_{12} = \boldsymbol{c}_8\boldsymbol{h}_{12} - \boldsymbol{c}_6\boldsymbol{h}_{22} \tag{31}$$
$$\boldsymbol{g}_{21} = -\boldsymbol{c}_6^{H}\boldsymbol{h}_{11} + \boldsymbol{c}_5\boldsymbol{h}_{21} \tag{32}$$
$$\boldsymbol{g}_{22} = -\boldsymbol{c}_6^{H}\boldsymbol{h}_{12} + \boldsymbol{c}_5\boldsymbol{h}_{22}, \tag{33}$$

where $\boldsymbol{h}_{ij}$ and $\boldsymbol{g}_{ij}$ (i={1,2}, j={1,2}) indicate a 2×2 submatrix of $\boldsymbol{H}_{[k]}$ and $\boldsymbol{G}_{[k]}$ with omitting a subcarrier index k. These computations consist of 2×2 matrix operations and have similarities among the above equations. The divisions of 9-step and 2-step computations in Strassen's algorithm are indicated in Table 2. Equations (19)–(33) are divided so as to equalize the numbers of matrix operations (multiplication, addition/subtraction, and inversion) at each step. The minimum requirement of 2×2 matrix arithmetic units in circuit design is given by this division.

4.4 Dynamically Reconfigurable Architecture

Figure 6 shows a flowchart of 2×2 matrix operations for the 9-step computation, which is given by two types of operation flow, i.e., "Type A" and "Type B." These matrix operation flows are used in steps in Table 2. The addition of elements by σ^2 and the Hermitian transpose are omitted because of their few complexity. The operation flows in "Type A" and "Type B" are different, however they can share the same operation units in matrix multiplication and addition. Dynamically reconfigurable architecture is a reasonable idea to realize this hardware resource sharing, which is achieved by changing data paths among matrix operation units. Dynamically reconfigurable architecture changes data path patterns among circuit units and their functions by the circuit units themselves. It provides a high degree of flexibility in changing circuit specifications and applications [15], [16] and hardware cost reduction by sharing common functional resources [17]. For digital signal processors, the flexibility enables implementation of various kinds of hardware processing in multimedia and wireless applications. This case reconfigures interconnections and logic functions in processor and memory units [15], [17]. For dedicated circuits, the reconfigurability is used for dynamically changing circuit specifications (e.g., varying FFT size in [16]) by reconfiguring interconnections and logic functions in arithmetic units. We make use of dynamically reconfigurable architecture for switching the matrix operation flows in a MIMO detector. It can minimize the used number of matrix arithmetic units by resource sharing even for every types of step computations, which are indicated by the items of "Minimum Requirements" in Table 2.

The circuit structure of a MIMO detector in the 9-step

Table 2 Division of Strassen's algorithm in (19) to (33).

(a) 9-Step Computation

Step	Equations	No. of 2x2 Matrix Arithmetic Units		
		2x2 MUL	2x2 ADD	2x2 INV
#1	(19)	2	1	0
#2	(20)	2	1	0
#3	(21)	2	1	0
#4	(22) - (25)	2	1	1
#5	(26) - (29)	2	1	1
#6	(30)	2	1	0
#7	(31)	2	1	0
#8	(32)	2	1	0
#9	(33)	2	1	0
Minimum Requirements in a MMSE MIMO Detector		2	1	1

(b) 2-Step Computation

Step	Equations	No. of 2x2 Matrix Arithmetic Units		
		2x2 MUL	2x2 ADD	2x2 INV
#1	(19) - (25)	8	4	1
#2	(26) - (33)	10	5	1
Minimum Requirements in a MMSE MIMO Detector		10	5	1

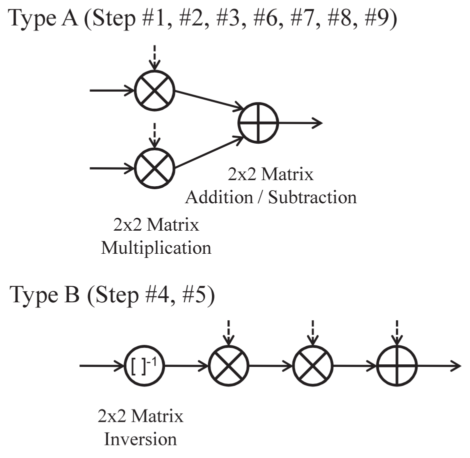

Type A (Step #1, #2, #3, #6, #7, #8, #9)
2x2 Matrix Addition / Subtraction
2x2 Matrix Multiplication
Type B (Step #4, #5)
[ ]-1
2x2 Matrix Inversion

Fig. 6 Flowchart of operations in the 9-step computation.

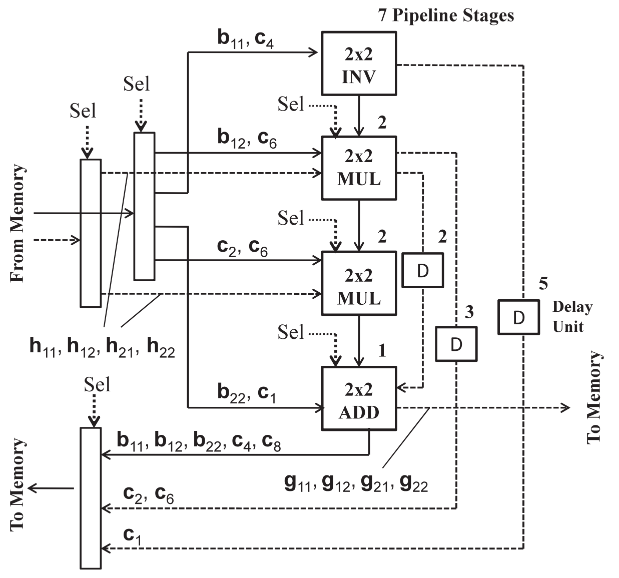


Fig. 7 Circuit structure in the 9-step computation.

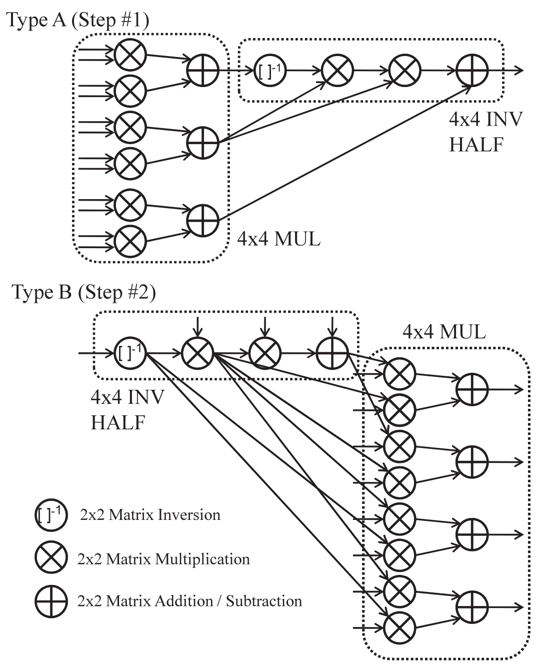


Fig. 8 Flowchart of operations in the 2-step computation.

computation is illustrated in Fig. 7. The signal "Sel" is used for switching the operation flows of "Type A" and "Type B." The output data moves to external memory and is reused as the intermediate value at the next step. For instance, the output data of $\boldsymbol{b}_{11}$ at the "Step #1" is utilized for the data input at the "Step #4." The external memory is assumed to be shared by the other processing, e.g., MIMO channel estimation and MIMO decoding. The detector has the total 12 pipeline stages when data goes through the matrix inversion to the matrix addition.

The flowchart of the 2-step computation is illustrated in Fig. 8. The "Type A" and "Type B" have the same matrix

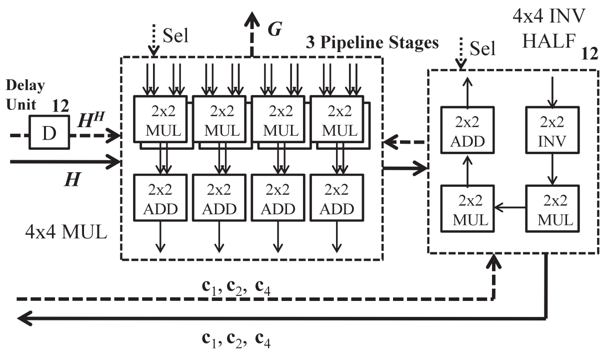


Fig. 9 Circuit structure in the 2-step computation.

operations marked as "4 × 4 MUL" and "4 × 4 INV HALF." It indicates that the 2-step computation can be achieved by changing data paths between "4 × 4 MUL" and "4 × 4 INV HALF." The circuit structure of a MIMO detector in the 2-step computation is shown in Fig. 9. The detector has maximum 15 pipeline stages. The order of operation flows in "4×4 MUL" and "4×4 INV HALF" can be switched by the control signal "Sel."

For 2×2 matrix operation units, we use the same circuit structure of pipelined arithmetic units presented in our previous work [11]. In 2 × 2 matrix inversion, we apply direct computation using

$$\begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix}^{-1} = \frac{1}{\lambda_{11}\lambda_{22} - \lambda_{12}\lambda_{21}} \begin{pmatrix} \lambda_{22} & -\lambda_{12} \\ -\lambda_{21} & \lambda_{11} \end{pmatrix}. \tag{34}$$

5. Implementation Results

The implementation results of the proposed detectors designed by 9-step and 2-step computations and the comparison with the conventional detectors are summarized in Table 3. The 9-step and 2-step computations operate at clock frequency of 250 MHz and 160 MHz, respectively. The 24-bit wordlength in a fixed-point format with dynamic floating scaling is adopted in both detectors. The data where the processing time is less than the acceptable time are meshed in the table. The detectors based on iterative architecture [5], [6] satisfy up to the condition of 52 subcarriers assuming IEEE802.11n standard at a 20-MHz channel. The complete pipeline detector [11] can provide real-time processing for all the conditions in subcarriers, however it requires two millions logic gates. The conditions of 216 and 472 subcarriers would be suitable for an 80-MHz channel OFDM discussed in the standardization of IEEE802.11ac, which have been evaluated in [12]. The 9-step computation is optimized for the condition of 108 subcarriers (explained in Sect. 4.2) and has reduced circuit scale by 86% compared with the previous detector [11]. The 2-step computation can satisfy all the conditions of subcarriers and has reduced circuit scale by 60%. The proposed scalable pipeline architecture achieves

Table 3 Circuit performance of MMSE MIMO detectors.

Reference	[5]	[6]	[11]	Proposed (9-Step)	Proposed (2-Step)
Operating Frequency (MHz)	167	140	160	250	160
Hardware Configuration (No. of Logic Gates)	ASIC 0.25μm	FPGA 0.12μm	ASIC 90nm	ASIC 90nm	ASIC 90nm
No. of Logic Gates	89 K	157 K	2.2 M	303 K	885 K
Power Consumption	N/A	N/A	700 mW	66 mW	167 mW
Supply Voltage	N/A	N/A	1.0 V	1.0 V	1.0 V
Processing Time for 52 Subcarriers in 20-MHz Channel (μs) (Acceptable latency time is 4 μs.)	31.2	2.77	0.19	2.10	0.51
Processing Time for 108 Subcarriers in 40-MHz Channel (μs) (Acceptable latency time is 4 μs.)	64.8	5.72		3.88	0.86
Processing Time for 216 Subcarriers in 80-MHz Channel (μs) (Acceptable latency time is 4 μs.)	129.6	10.8		7.34	1.54
Processing Time for 472 Subcarriers in 80-MHz Channel (μs) (Acceptable latency time is 7.2 μs.)	283.2	25.2		15.53	3.14

power reduction of 1/11 and 1/4 in the 9-step and 2-step computation, respectively.

6. Conclusion

We have presented a scalable pipeline MMSE MIMO detector for a 4×4 MIMO-OFDM receiver. The new concept is to optimize a circuit structure of a detector by adjusting the number of iteration steps according to various types of OFDM system. We have proposed scalable pipeline architecture based on this concept and presented the design of a MMSE MIMO detector. In the VLSI implementation, the designed detectors of 9-step and 2-step computations have attained both circuit reduction and real-time processing on the conditions of subcarriers in IEEE802.11n and IEEE802.11ac. Other scalable pipeline detectors (e.g., 4- and 5- step computations) could be designed according to the proposed architecture, whose implementations will be presented in our future works.

References

[1] "IEEE P802.11n/D4.00: Draft amendment to wireless LAN media access control (MAC) and physical layer (PHY) specifications: Enhancements for higher throughput," March 2008.

[2] Rolf de V, "802.11ac Usage Models Document," doc.:IEEE802.11-09/0161r2, Jan. 2009.

[3] V. Jungnickel, A. Forck, T. Haustein, et al., "1 Gbit/s MIMO-OFDM transmission experiments," IEEE Vehicular Technology Conference (VTC), vol.2, pp.25–28, Sept. 2005.

[4] J. Eilert, D. Wu, and D. Liu, "Efficient complex matrix inversion for MIMO software defined radio," IEEE International Symposium on Circuits and Systems (ISCAS), pp.2610–2613, May 2007.

[5] A. Burg, S. Haene, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," IEEE International Symposium on Circuits and Systems (ISCAS), pp.4102–4105, May 2006.

[6] H.S. Kim, W. Zhu, J. Bhatia, K. Mohammed, A. Shah, and B. Daneshrad, "A practical, hardware friendly MMSE detector for MIMO-OFDM-based systems," EURASIP Journal on Advances in Signal Processing, vol.2008, Article ID 267460, 2008.

[7] Z. Khan, T. Arslan, J.S. Thompson, and A.T. Erdogan, "Area & power efficient VLSI architecture for computing pseudo inverse of channel matrix in a MIMO wireless system," 19th International Conference on VLSI Design (VLSID), Jan. 2006.

[8] D. Perels, S. Haene, P. Luethi, A. Burg, N. Felber, W. Fichtner, and H. Bolcskei, "ASIC implementation of a MIMO-OFDM transceiver for 192 Mbps WLANs," 31st European Solid-State Circuits Conference (ESSCIRC), pp.215–218, Sept. 2005.

[9] S. Chen, T. Zhang, and M. Goel, "Relaxed tree search MIMO signal detection algorithm design and VLSI implementation," IEEE International Symposium on Circuits and Systems (ISCAS), pp.1147–1150, May 2006.

[10] B. Mennenga, E. Matus, and G. Fettweis, "Vectorization of the sphere detection algorithm," IEEE International Symposium on Circuits and Systems (ISCAS), pp.2806–2809, May 2009.

[11] S. Yoshizawa, Y. Yamauchi, and Y. Miyanaga, "VLSI implementation of a complete pipeline MMSE detector for a 4×4 MIMO-OFDM receiver," IEICE Trans. Fundamentals, vol.E91-A, no.7, pp.1757–1762, July 2008.

[12] S. Yoshizawa and Y. Miyanaga, "VLSI implementation of a 4×4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth," IEEE International Symposium on Circuits and Systems (ISCAS), pp.1743–1746, May 2009.

[13] V. Strassen, "Gaussian elimination is not optimal," Numer. Math., vol.13, no.3, pp.354–356, 1969.

[14] M. Myllyla, J.-M. Hintikka, J.R. Cavallaro, and M. Juntti, "Complexity analysis of MMSE detector architectures for MIMO OFDM systems," IEEE International Symposium on Spread Spectrum Techniques and Applications (ISSSTA), pp.12–16, Sydney, Aug. 2004.

[15] J. Becker, T. Pionteck, and M. Glesner, "DReAM: A dynamically reconfigurable architecture for future mobile communication applications," Proc. The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications, pp.312–321, Aug. 2000.

[16] G. Zhong, F. Xu, and A.N. Willson, "A power-scalable reconfigurable FFT/IFFT IC based on a multi-processor ring," IEEE J. Solid-

State Circuits, vol.41, no.2, pp.483–495, Feb. 2006.

[17] Y. Kim, M. Kiemb, C. Park, J. Jung, and K. Choi, "Resource sharing and pipelining in coarse-grained reconfigurable architecture for domain-specific optimization," Proc. Design, Automation and Test in Europe (DATE), vol.1, pp.12–17, March 2005.

Shingo Yoshizawa received the B.E., M.E., and Ph.D. degrees from Hokkaido University, Japan in 2001, 2003 and 2005, respectively. He is an Assistant Processor and currently working at the Graduate School of Information Science and Technology, Hokkaido University. His research interests are speech processing, wireless communication, and VLSI architecture.

Hirokazu Ikeuchi received the B.S. degree from Hokkaido University, Japan in 2009. He is currently studying at the Graduate School of Information Science and Technology, Hokkaido University. His research interests are wireless communication and VLSI design.

Yoshikazu Miyanaga received the B.S., M.S., and D.Eng. degrees from Hokkaido University, Japan in 1979, 1981, and 1986, respectively. From 1983 to 1987, he was a Research Associate at the Institute for Electronic Science, Hokkaido University. From 1987 to 1988, he was a Lecturer at the Faculty of Engineering of Hokkaido University. From 1988 to 1997, he was an Associate Professor there. He is currently a Professor at the Graduate School of Information Science and Technology, Hokkaido University. His current research interests are adaptive signal processing, non-linear signal processing, and parallel-pipelined VLSI systems.