IMPLEMENTATION OF A PROGRAMMABLE LINEAR MMSE DETECTOR FOR MIMO-OFDM

Johan Eilert, Di Wu and Dake Liu

Department of Electrical Engineering, Linköping University, 58183 Linköping, Sweden

ABSTRACT

This paper presents a linear minimum mean square error (LMMSE) symbol detector for MIMO-OFDM enabled mobile terminals. The detector is implemented using a programmable baseband processor aimed for software-defined radio (SDR). Owing to the dynamic range supplied by the floating-point SIMD datapath, special algorithms can be adopted to reduce the computational latency of detection. The programmable solution not only supports different transmit/receive antenna configurations, but also allows hardware multiplexing to obtain silicon and power efficiency. Compared to several existing fixed-functional solutions, the one proposed in this paper is smaller, more flexible and faster.

Index Terms— MIMO systems, OFDM, Programmable circuits, Very-large-scale integration

1. INTRODUCTION

Multi-antenna or multi-in and multi-out (MIMO) is a technology to greatly enhance the performance of wireless communications by utilizing various degrees of freedom. Orthogonal frequency division multiplexing (OFDM) is a promising technology for its capability to convert a frequency-selective channel into a number of flat fading subchannels. Due to the property that subcarriers are orthogonal to each other, the guard bands are no longer necessary, which greatly increases the spectrum efficiency. Although signals in different subcarriers are overlapped in frequency, it is possible to recover at the receiver side as long as the orthogonality is maintained. MIMO and multi-carrier (e.g. OFDM and OFDMA) technologies have been adopted by most emerging wireless broadband standards (e.g. WiMAX and 3GPP LTE) to increase the spectrum efficiency.

Since complex valued matrix manipulations such as matrix inversion and QR decomposition are common operations in the receiver of MIMO-OFDM systems, the receiver complexity is much higher than that in SISO and single-carrier systems. Therefore, with the limited amount of computing power available to mobile terminals, trade-off between the performance and power consumption is a critical issue to be carefully addressed in MIMO-OFDM systems.

2. SYSTEM MODEL

A general MIMO enhanced multi-carrier system model is depicted as in Fig. 2 which might be either OFDM or OFDMA based system. Taking a MIMO-OFDM system as an example, it has N_T TX antennas, N_R antennas and K sub-carriers in one OFDM block. It is assumed that the time-variant wireless channel is static or quasi-static during P consecutive OFDM blocks. Channels between each pair of TX-RX antennas are uncorrelated from each other.

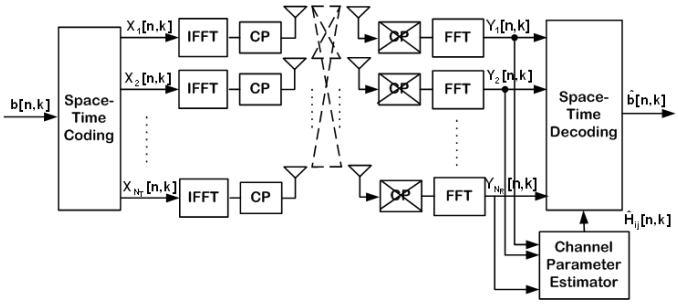

Fig. 1. A MIMO Enhanced Multi-Carrier System

During a transmission interval n, a stream of binary bits b is coded into N_T symbol blocks. The signal on the kth subcarrier at the ith TX antenna is denoted by $X_i[n,k]$, where $i = 1,\ldots,N_T, k = 0,\ldots,K-1, n = 0,\ldots,P-1$. The received signal at RX antenna j is

$$Y_j[n,k] = \sum_{i=1}^{N_T} X_i[n,k]H_{ij}[n,k] + N_j[n,k] \qquad (1)$$

where $H_{ij}[n,k]$ is the frequency response between antennas i and j, $N_i[n,k]$ is additive Gaussian noise with zero mean and variance σ_n^2. Eq. 1 can also be written as

$$\mathbf{Y}(n) = \mathbf{X}(n)\mathbf{H} + \mathbf{N}(n) \qquad (2)$$

3. LINEAR MMSE DETECTION

LMMSE is one of the most straightforward detection schemes which outperforms Zero-Forcing detection by taking the noise into consideration. Meanwhile, it has still a reasonable implementation complexity. According to the Eq. 2, LMMSE detection can be described in the following

$$\mathbf{X} = \left(\mathbf{H^H H} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{H^H Y} \qquad (3)$$

which involves the precalculation of a coefficient matrix

$$\mathbf{W} = \left(\mathbf{H^H H} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{H^H} \qquad (4)$$

The precalculation involves quite a few matrix manipulations such as matrix inversion and multiplication. The frequency of these matrix manipulations depends on the variation of the wireless channel. Taking a 4×4 spatial multiplexing or 2×2 space-time block coding (STBC) MIMO-OFDM based WiMAX system as an example, we assume 512 subcarriers are used and the working frequency f is 2.5 GHz. If the mobile handset is moving at speed v = 100km/h, then based on the following formula

$$f_m = \frac{vf}{c}$$

the maximum Doppler shift f_m is 231.5 Hz. The following formula given in [2],

$$T_c \approx \frac{1}{f_m}$$

estimates the channel coherence time T_c to be $4\ ms$. In this case, excluding the pilot and null subcarriers, the number of channel matrices to be inverted is 420 before the detection can start. Although the channel matrix will not change drastically within the coherence time, the detection can not start before the estimation and preprocessing of the channel matrices is finished. Thus the inversion of a 4×4 matrix as well as all other channel estimation computation needs to be finished as soon as possible. The goal of this paper is to find a practical solution that meets this real-time constraint.

4. MATRIX INVERSION

$\mathbf{H}$ can be any matrix of arbitrary size. As mentioned in [5], the size of $\mathbf{H}$ considered is typically between 2×2 and 4×4. Although larger matrices (e.g. 8×8) can still be managed [5], the cost of real-time implementation will be much higher.

4.1. The General Case

As mentioned in [3], for large matrices, matrix inversion is traditionally implemented by applying QR factorization to the original matrix to generate an upper triangular matrix R, then the result can be computed using back substitution. However, for small matrices, as shown by [4], there are other alternatives which are faster, more silicon efficient while still providing sufficient numerical stability. In [4], the use of a method called blockwise analytic matrix inversion (BAMI) is proposed to compute the inversion of complex-valued matrices by partitioning the matrix into four smaller matrices, and then compute the inverse based on computations on these smaller parts. For example, to compute the inverse of a 4×4 matrix M, it is first divided into four submatrices

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

The inverse of M can be computed as:

$$\begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}$$

4.2. STBC: A Special Case

Alamouti matrix [1] based orthogonal space-time-block-coding (STBC) has been widely used to exploit the diversity in space and time domain. The basic 2×2 Alamouti matrix is defined in [1] as

$$A = \begin{bmatrix} a_1 & a_2 \\ -a_2^* & a_1^* \end{bmatrix} \quad (5)$$

the structure of matrices based on 2×2 Alamouti sub-blocks remains invariant under several nontrivial matrix operations including matrix inversion and QR decomposition. Therefore, based on BAMI and the special structure of Alamouti matrix, in Ref. [5] a new method namely Alamouti blockwise analytic matrix inversion (ABAMI) is proposed by us which significantly reduces the amount of computation needed to invert large Alamouti sub-block based matrices. For example, the inversion of a 2×2 Alamouti matrix can be computed as follows:

$$\mathbf{H^{-1}} = \begin{bmatrix} a_1 & a_2 \\ -a_2^* & a_1^* \end{bmatrix}^{-1} = \frac{1}{a_1 {a_1}^* + a_2 {a_2}^*} \begin{bmatrix} a_1^* & -a_2 \\ a_2^* & a_1 \end{bmatrix}$$

where $a_1 {a_1}^* + a_2 {a_2}^* = |a_1|^2 + |a_2|^2 = \alpha_A$ is a real valued result computed from the two complex values a_1 and a_2. In order to calculate the inverse, we need the following several operations: one dot operation to generate α_A, one operation to calculate the real valued $1/\alpha_A$, two complex-with-real multiplications and a few sign-flip operations. Compared to the BAMI method, the number of operations is reduced by almost half, which makes ABAMI by far the simplest method for matrix inversion in literature. The LMMSE detecter presented in this paper is using BAMI and ABAMI based matrix inversion to compute the coefficient matrix in Eq. (4).

5. ARCHITECTURE OF PROGRAMMABLE HARDWARE

As mentioned in [6], systolic array is a classical architecture to implement QR decomposition for high performance solutions. However, traditional systolic arrays usually consume large silicon area and do not scale very well as the size of the matrix changes. Since it is already shown by [5] that BAMI and ABAMI can efficiently handle matrix inversion using programmable HW, in order to fully prove the concept, a complete LMMSE detector is designed and implemented. The detector has limited programmability due to design simplications, nevertheless it supplies enough flexibility to support most linear detectors (e.g. ZF and LMMSE).

As depicted in Fig. 2, the baseband processor presented in this paper contains a 4-way SIMD Floating-Point Complex Multiply and ACcumulation (FPCMAC) datapath. Fig. 3 shows the schematic of one FPCMAC. With 32 complex-valued general registers and four accumulation registers, the processor is enough to compute the inverse of matrices not larger than 4×4 with little memory overhead. For larger matrices (e.g. 8×8), data need to be moved in between the register file and memory thus introducing some overhead. For-

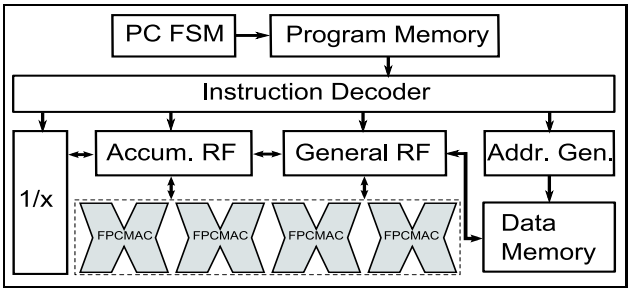

Fig. 2. Architecture of programmable hardware used for MMSE detection

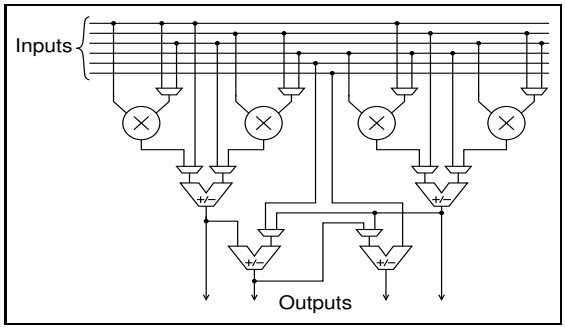

Fig. 3. Schematic of the FPCMAC (the divider unit is not shown)

tunately, in most standards, only channel matrices not larger than 4×4 are involved.

6. SIMULATION

The performance (SNR/BER) curves of MMSE detection are depicted in Fig. 4 and Fig. 5 using 16-bit, 20-bit and 64-bit (IEEE double precision) floating-point datatypes. The size of the matrices involved ranges from 4×4 to 8×8. As illustrated in Fig. 5, for 8×8 matrix inversion, the BER/SNR performance curve will saturate even if the SNR increases further. In comparison, the 20-bit representation brings sufficient precision for larger matrices. It has also been shown in [5] that ABAMI has the same numerical stability as BAMI.

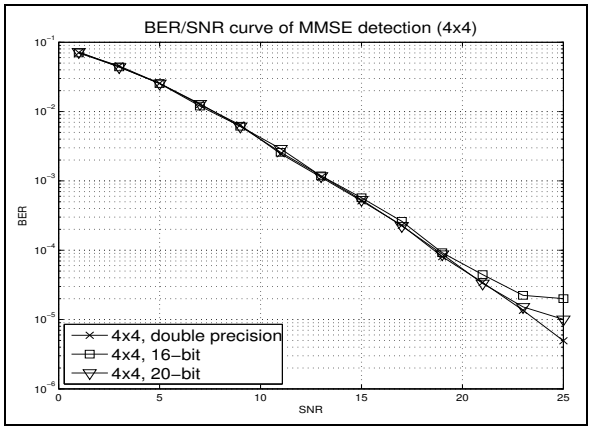

Fig. 4. Performance of LMMSE Detection (4×4)

Mnem	Name	Description	cycles
sabs	Cplx squared abs	$c = a.r^2 + a.i^2$	2
ssa	Sum squared abs	$c = a.r^2 + a.i^2 + b.r^2 + b.i^2$	3
cip	Cplx inner product	$c = \sum (a_i.r^2 + a_i.i^2)$	4
cmul	Cplx multiply	$c.r = a.r * b.r - a.i * b.i$ $c.i = a.r * b.i + a.i * b.r$	2
cmac	Cplx multiply-add	$c.r = c.r + a.r * b.r - a.i * b.i$ $c.i = c.i + a.r * b.i + a.i * b.r$	3
rmul	Real-Cplx multiply	$c.r = a.r * b; c.i = a.i * b$	1
inv	Real $1/x$	$b = 1/a$	3

Table 1. Complex Floating-Point Instructions

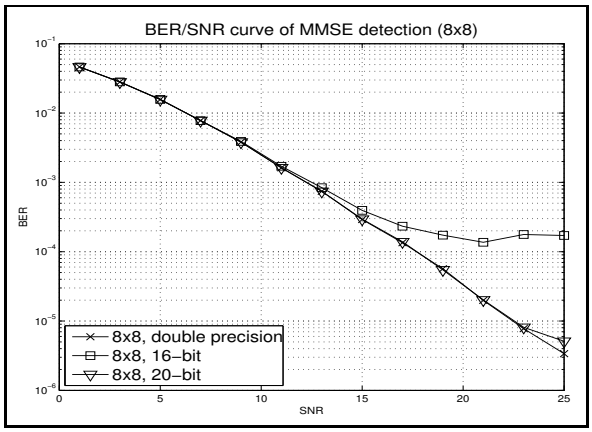

Fig. 5. Performance of LMMSE Detection (8×8)

7. IMPLEMENTATION

In order to evaluation the performance of the LMMSE detector presented in this paper with several existing solutions, it is synthesized using both the Xilinx FPGA and ST 65 nm CMOS ASIC technologies.

7.1. FPGA Prototype

For the FPGA implementation, Xilinx ISE and Core Generator were used to synthesize the design based on the Virtex4-xc4vlx200 FPGA. The input is a 4×4 matrix of complex floating-point values and output is the inverse matrix. All the basic units are generated using Xilinx Core Generator. Both the 16-bit and 20-bit implementations have 9 pipeline stages in the FPCMAC and take 8 cycles to finish the real value division (1/X). Our design has also been synthesized using Virtex2 and the main difference from Virtex4 is the clock frequency. As depicted in Table 2, compared to the latest synthesis result presented in [7] and [6], our 20-bit implementation is much faster and occupies smaller area.

	Impl_16b	Impl_20b	Ref [7]	Ref [6][1]
FPGA Type	Virtex4	Virtex4	Virtex2	Virtex2
Num of Parallel Streams	4	4	1	1
Datatype	floating	floating	fixed	fixed
Wordlength (bits)	16	20	16	12
Num of Slices	7312	9474	16805	4400[1]
Num of DSP48/MULT	0	0	44	0
Frequency (MHz)	120	110	66	100
Cycles to compute W	270	270	3000	350[1]
Latency/subcarrier (μs)	0.563	0.614	45	N/A[1]

Table 2. FPGA Implementation Result Comparison ([1]the implementation in [6] only performs matrix inversion)

7.2. ASIC Implementation

Table 3 depicts the synthesis result including the gate count, working frequencies and pipeline stages for various floating-point wordlength. The 16-bit implementation can easily run at 400 MHz and takes 88 cycles to compute the inverse of a 4×4 matrix and in total 202 cycles to finish the preprocessing including matrix inversion and multiplications. The 20-bit implementation can run at 400 MHz by having 3 pipeline stage in the 1/X unit. In this case, it requires 90 cycles and 204 cycles to finish the same tasks accordingly. In other words, the preprocessing of 420 matrices can be finished within $53\ \mu s$ which is only 1.23% of the channel coherence time.

	Impl_16b	Impl_20b
Wordlength (bits)	16	20
Num of Parallel Streams	4	4
Area (kgate)	90	120
Num of Pipeline Stages in FPCMAC	3	3
Cycles for Division	1	3
Working Frequency (MHz)	400	400
Cycles to Compute Matrix Inversion	85	90
Cycles to Compute W	202	204
Latency/subcarrier (μs)	0.126	0.128
Latency for 420 subcarriers (μs)	53	54

Table 3. ASIC Implementation of the Baseband Processor

8. DESIGN EVALUATION AND TRADE-OFF

8.1. Long vs. Short Wordlength

For any embedded system, there exists a design trade-off which depends on the target of the system. From a cost efficency perspective, the shorter the wordlength, the smaller the silicon area and the higher the clock frequency can be. On the other hand, in order to accommodate various antenna configurations (in other words, channel matrices in different sizes), longer wordlength is expected to supply enough precision for the matrix inversion. Facing such a dilemma, a trade-off must be made according to the product specification.

For example, in case the baseband processor is designed for standards that are relatively fixed, only the set of antenna configurations specified in the standard needs to be covered. As specified in WiMAX (IEEE-802.16-2005), only matrices not larger than 4×4 are involved, which means that the 16-bit implementation in this paper is sufficient for the preprocessing of channel matrices. Meanwhile, since silicon cost is important for a volume product, the 16-bit implementation is more attractive due to the smaller area.

In case the processor is designed for SDR systems, the antenna configuration and other system parameters are usually unknown and dynamic, so that longer wordlength is preferred to maintain the precision during the computation. Therefore, for SDR systems aimed for algorithm research or defence applications which require the quick deployment of a new system, longer wordlength (e.g. 20-bit) is expected.

8.2. Fixed-Point vs. Floating-Point

Although it is possible to use fixed-point hardware for the matrix inversion in MIMO systems, it usually requires very careful scaling (especially under badly conditioned channel) which is more effort demanding for hardware and software development. In comparison, floating-point hardware has the advantage of higher productivity. Furthermore, as the semiconductor process scales to 45 nm, tapeout cost is more vital compared to silicon efficiency, thus more weight should to be put on the flexibility of hardware in the design phase.

8.3. Fixed-Functional vs. Programmable

The last but not the least, aside from the LMMSE detector, the programmable hardware can be reused for other kernel operations. For example, with a minimum of extra hardware (two adders), it can execute a 512-point FFT in around 2300 clock cycles. By time-division hardware multiplexing (reuse), the programmable hardware occupies smaller silicon area, which reduces the leakage-power. This is especially important because leakage power increases significantly when the semiconductor process reaches nanoscale.

9. CONCLUSION

In this paper, a programmable LMMSE detector for MIMO-OFDM is presented. By comparing it with other existing solutions in Sec. 7, our implementation is the fastest with the lowest silicon cost. The result shows that application specific programmable hardware allows us to achieve both efficiency and flexibility. On the other hand, this can only be achieved when a good trade-off is made based on algorithm/hardware codesign.

10. ACKNOWLEDGEMENT

The work of J. Eilert, D. Wu and D. Liu is supported partly by the STRINGENT program from SSF. The authors would like to thank ST Microelectronics for supplying 65nm process and D. Wang at UT Dallas for discussion on STBC algorithms.

11. REFERENCES

[1] S. M. Alamouti, "A Simple Transmit Diversity Technique for Wireless Communications", *IEEE J. Select. Areas Commun*, vol. 16, no. 8, pp. 1451-1458, 1998

[2] J. Andrews, A. Ghosh, R. Muhamed, "Fundamentals of WiMAX: Understanding Broadband Wireless Networking", Prentice Hall, Mar 2007

[3] G. H. Golub, C. F. Van Loan, "Matrix Computations, Third Edition", The Johns Hopkins University Press, 1996.

[4] J. Eilert, D. Wu, D. Liu, "Efficient Complex Matrix Inversion for MIMO Software Defined Radio", *Proc. IEEE ISCAS*, 2007.

[5] D. Wu, J. Eilert, D. Liu, D. Wang, N. Al-Dhahir and H. Minn, "Fast Complex Valued Matrix Inversion for Multi-User STBC-MIMO Decoding", *Proc. IEEE ISVLSI*, 2007.

[6] F. Edman, V. Öwall, "A Scalable Pipelined Complex Valued Matrix Inversion Architecture", *Proc. IEEE ISCAS*, 2005.

[7] M. Myllylä, J. Hintikka, J. R. Cavallaro and M. Juntti, M. Limingoja, A. Byman, "Complexity Analysis of MMSE Detector Architectures for MIMO OFDM Systems", *Proc. 39th Asilomar Conference on Signals, Systems and Computers*, 2005.