Fast Complex Valued Matrix Inversion for Multi-User STBC-MIMO Decoding

Di Wu, Johan Eilert and Dake Liu
Linköping University
Department of Electrical Engineering
Linköping, 581 83, Sweden
{diwu,je,dake}@isy.liu.se

Dandan Wang, Naofal Al-Dhahir and Hlaing Minn
The University of Texas at Dallas
Department of Electrical Engineering
Richardson, TX 75083-0688, USA
{dxw053000, aldhahir, Hlaing.Minn}@utdallas.edu

Abstract

This paper studies the efficient complex matrix inversion for multi-user STBC-MIMO decoding. A novel method called Alamouti blockwise analytical matrix inversion (ABAMI) and its programmable VLSI implementation are proposed for the inversion of (in this context) large complex matrices with Alamouti sub-blocks. Our solution significantly reduces the number of operations which makes it more than 4 times faster than several other solutions in the literature. Furthermore, compared to these fixed function VLSI implementations, our solution is more flexible and consumes less silicon area because the hardware can be reused for many other operations. In addition to the routine analysis of the general computational complexity based on the number of basic operations, the computational latency is also measured in clock cycles based on the conceptual hardware for real-time matrix inversion.

1. Introduction

Multi-antenna or multi-in and multi-out (MIMO) schemes have been adopted by many standards as an advanced technology to greatly enhance the performance of wireless communications by utilizing various degrees of freedom. For example, for laptops equipped with MIMO technology, more than 8 antennas might be installed to achieve either diversity gain or increasing the data rate or both. Meanwhile, the increased complexity of MIMO system brings great challenge to its VLSI implementation, thus making it the focus of both academia and industry. Especially programmable solutions targeting Software-Defined-Radio (SDR) is among the hottest research directions within this area.

Since complex valued matrix manipulations such as matrix inversion and multiplication are common operations in the receiver of MIMO systems, the complexity of MIMO receiver is several magnitudes higher than that in SISO systems. Especially as the complexity of channel coding (e.g. Viterbi) increases only linearly, the complexity of the decoder becomes dominant. Therefore, efficient matrix inversion is a critical issue to be carefully addressed in MIMO systems.

In this paper, we present several novel methods of scalable matrix manipulations for multi-user STBC decoding, which form the key part of a MIMO receiver. Alamouti blockwise analytical matrix inversion (ABAMI) is proposed for the inverse of large complex matrices that are based on Alamouti sub-blocks.

The remainder of the paper is organized as follows. In Section 2, the system model is presented. Section 3 describes two traditional ways of matrix inversion. Our methods namly blockwise analytic matrix inversion (BAMI) and Alamouti blockwise analytic matrix inversion (ABAMI) are presented in Section 4. Section 5 presents the architecture of the application specific instruction set processor (ASIP) based SDR platform. The numerical representation of the system is discussed in Section 6. Section 7 presents the hardware cost using FPGA and ASIC technologies together with the computational latency. Finally, Section 8 concludes the paper.

2. System Model

Diversity in MIMO leads to improved reliability of information transmission for a given data rate, which can be achieved in multiple dimensions such as space, time and frequency. Among the diversity enhancing schemes, space-time block coding (STBC) is one of the most widely used transmission method. The basic principle of STBC is to transmit multiple copies of the information symbols over multiple independent channels in time and space. Alamouti STBC is a basic STBC scheme proposed in [1]. The basic 2×2 Alamouti matrix is defined in [1] as

$$A = \begin{bmatrix} a_1 & a_2 \\ -a_2^* & a_1^* \end{bmatrix}$$

Because of the simplicity of the Alamouti structure, based on Alamouti sub-blocks, multi-user STBC schemes can be implemented to enhance the data rate by utilizing a greater number of antenna pairs. As illustrated in Fig. 1, multi-user STBC is a scheme that uses multiple antenna arrays (two elements in each) to transmit information symbols using Alamouti STBC. The data rate enhancement can be achieved in both the uplink and downlink. Fig. 1 (a) illustrates a scenario where the base station (BS) receives data from two mobile stations (MS). And Fig. 1 (b) depicts the case where a BS is using two antenna arrays (two elements in each) to transmit data to a single MS.

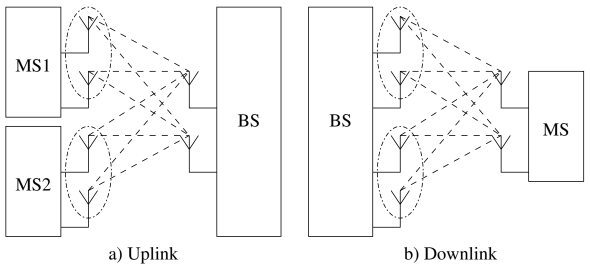

Figure 1. Alamouti Multi-User Transmission Scheme

In MIMO systems, the general transmission model is

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$$

where the channel matrix $\mathbf{H}$ is obtained with channel estimation, and the received vectors $\mathbf{y}$ are transformed using a linear or nonlinear matrix equalizer to estimate the transmitted vector $\mathbf{s}$. Vector $\mathbf{n}$ is the additive noise at the receiver side.

3. Matrix Inversion in MIMO

Matrix inversion, which is usually involved in the linear matrix equalizer, is a traditional topic in the area of numerical analysis and it has been well elaborated in many publications, e.g. in the book by Golub and Van Loan [2]. In the following two subsections, we give two classical approaches for matrix inversion and their corresponding hardware implementation.

3.1. Direct Analytic Matrix Inversion

A straightforward way to compute matrix inversion is the analytic approach. For example, the inversion of a 2×2 matrix is computed as follows:

$$\mathbf{H^{-1}} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Unfortunately, the analytic approach lacks scalability because the computation complexity increases very quickly as the size of the matrix increases and the numerical stability does not exist any more for the inversion of large matrices.

3.2. Matrix Inversion based on QR Decomposition

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. As mentioned in [2], there are several ways to compute QR decomposition, such as Gram-Schmidt transform, Householder matrices, and Givens rotations. Recently, Squared Givens rotations (SGR) [3] has received attention for hardware implementation for QR decomposition since it avoids the square-root operation and reduces the number of multiplications by half ([5], [6]). Also, parallelism exists in SGR so that it can be mapped to parallel processing hardware to achieve higher performance.

3.3. HW Implementation

Systolic array is a classical architecture to implement QR decomposition for high performance solutions. There have been numerous contributions in this area such as the systolic array for SGR in [4]. However, traditional systolic arrays usually consume large silicon area and do not scale very well as the size of the matrix increases. Edman proposed a linear array that is more scalable than the traditional array [5]. In [6], Karkooti combined similar nodes and added a scheduler to control the movement of data between nodes so as to avoid using a quadratic array. However, both of these solutions introduce heavy latency.

4. Blockwise Matrix Inversion

In [7], a blockwise analytic matrix inversion is proposed which significantly reduces the amount of operations needed compared to several other latest solutions such as [5] and [6]. In this paper, we extended the blockwise analytic matrix inversion (BAMI) proposed in [7] to larger matrices. Also a new method namely Alamouti blockwise analytic matrix inversion (ABAMI) is proposed to further reduce the computational complexity by utilizing the special structure of Alamouti sub-block based matrices.

4.1. Blockwise Analytic Matrix Inversion

A fast way to compute matrix inversion involves partitioning the matrix into four smaller matrices. For a matrix

M divided into sub-matrices A, B, C and D,

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

the inversion of M is

$$M^{-1} = \begin{bmatrix} W + ZX & -Z \\ -YX & Y \end{bmatrix},$$

where

$$\begin{aligned} W &= A^{-1} \\ X &= CW \\ Y &= (D - XB)^{-1} \\ Z &= WBY \end{aligned}$$

The blockwise analytic matrix inversion (BAMI) subdivides larger matrices using the blockwise matrix inversion and uses direct analytic matrix inversion to solve the base case. As shown in [7], compared to the direct analytic approach that inverts the whole matrix in one step, BAMI is more robust against the finite-length error which makes it more stable for finite-length implementations. Furthermore, BAMI requires fewer operations for larger matrices compared to the direct analytic approach which makes it more scalable to accommodate larger matrices.

4.2. Alamouti Blockwise Analytic Matrix Inversion

As proven in [8] and [9], 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, a new method namely Alamouti blockwise analytic matrix inversion (ABAMI) is proposed 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-flipping 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.

The same idea can be applied to blockwise matrix inversion of any matrix with Alamouti sub-blocks. In principle, since only half of the values need to be computed, both the number of operations and the number of registers needed can be reduced by half compared to the BAMI approach. Table 1 shows the number of fundamental operations (*real valued* multiplication, addition/subtraction and division, here one complex multiplication equals to four real multiplications and two real additions) needed by the BAMI and ABAMI to compute the inverse of complex valued matrices in different sizes.

Size of Matrix		2×2	4×4	8×8	16×16
BAMI	Mul	28	248	2032	16352
	Add/Sub	15	190	1788	15608
	Div	1	2	4	8
ABAMI	Mul	8	112	1120	8384
	Add/Sub	3	86	1004	7896
	Div	1	2	4	8

Table 1. Computational Complexity Measured by Number of Operations (theoretical "FLOPS")

5. Software-Defined-Radio Platform

5.1. HW Architecture

In order to meet the demand of baseband processing of various emerging wireless technologies that utilize MIMO, a baseband ASIP for software-defined-radio (SDR) is being developed in the division of computer engineering, Linköping University. Similar to its predecessor [10], it can support heterogeneous MIMO standards by only loading different programs instead of redesigning the hardware. The architecture of the processor is illustrated by Fig. 2.

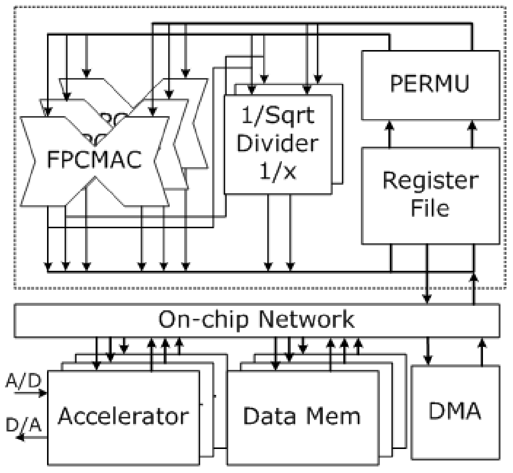

Figure 2. Floating Point Baseband SDR Processor

As depicted in Fig. 2, the datapath of this processor will include several Floating-Point Complex Multiply-ACcumulate (FPCMAC) units and several floating-point units that calculate $\frac{1}{x}, \frac{1}{\sqrt{x}}$ and real valued division. Fig. 3 illustrates the schematic of the FPCMAC which also can be used for FFT/IFFT. In this paper, the implementation of matrix inversion is based on the FPCMAC and the real divider. The instructions that involve the FPCMAC are shown in Table 2.

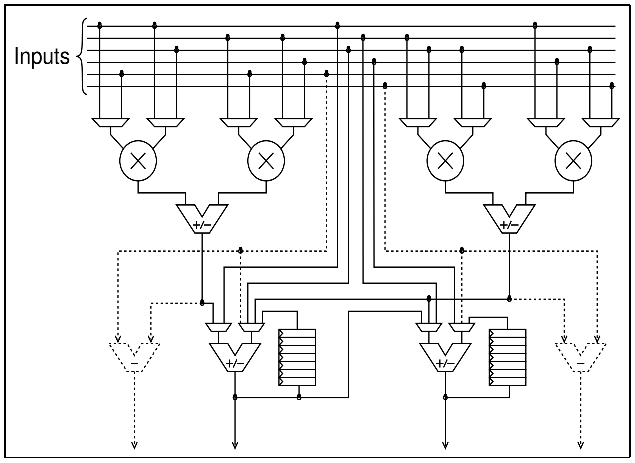

Figure 3. Schematic of the FPCMAC. The divider unit is not shown. The dotted adders are used during FFT/IFFT, their connections are not shown.

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

Table 2. Complex Floating-Point Instructions

5.2. Programming and Scheduling

Instructions such as CMUL, CMAC, SABS, SSA and RMUL are used to compute the matrix inversion as shown in Table 3. It is very important to schedule the algorithm so that the pipeline stalls caused by data dependency can be kept the minimum. Still it is rather straightforward to implement the blockwise analytic matrix inversion. For example, the assembly code in Table 3 (a) shows how to compute the inverse of a 2×2 matrix using BAMI method. And the inversion of a 2×2 Alamouti matrix using ABAMI is depicted in Table 3 (b).

(a) General 2×2

```
cmul      Aa,Ad,AC0
cmac      -Ab,Ac,AC0,t0
(stall for 2 cycles)
sabs      t0,div
(stall for 4 cycles
including the division)
rmul      ~t0,div,AC0
cmul      AC0,Ad,Xa
cmul      AC0,-Ab,Xb
cmul      AC0,-Ac,Xc
cmul      AC0,Aa,Xd
```

(b) Alamouti 2×2

```
ssa       Aa,Ab,div
(stall for 5 cycles
including the division)
rmul      ~Aa,div,Xa
rmul      -Ab,div,Xb
```

sabs	Complex squared abs
ssa	Sum of squared abs
cmul	Complex multiply
cmac	Complex multiply-add
rmul	Real multiply

Table 3. Code Examples for 2×2 Matrix Inversion

6. Numerical Representation

In real-world communications systems, narrow finite-length datatypes (e.g. 16-bit floating-point) are used for cost and speed reasons. Therefore, the error generated by the finite-length effect usually affects the performance of the system. Matrix inversion algorithms are sensitive to finite-length effects, which means that as the SNR improves, the induced finite-length error will become significant compared to the noise, which in the end dominates the BER.

6.1. Simulation Setting

In order to evaluate the numerical stability of different matrix inversion methods againt the finite-length effect, and to select the shortest wordlength which still supplies sufficient precision, a finite-length floating-point simulator is developed. Transmission schemes consisting of different number of users (in other words, different size of channel matrices) are simulated with their performance compared. For example, for a two-user case the simulated system has four transmitting antennas and two receiving antennas using STBC with two time slots, the resulting matrix is 4×4. The transmitter modulates random binary data with QPSK modulation. The decoder uses the MMSE algorithm to recover the information symbols.

Monte Carlo simulation is used to compute the BER/SNR curves. By default, the randomly generated channel matrices have unity power on average, and noise is added on the receiver side according to the SNR.

6.2. Wordlength Selection

In this paper, 16-bit (10-bit mantissa, 6-bit exponent) and 20-bit (14-bit mantissa, 6-bit exponent) floating-point representations are evaluated for the inverse of large matrices. Fig. 4 shows the BER/SNR curves of MMSE decoders using the blockwise analytic matrix inversion (BAMI)[7]. Although the 16-bit floating-point datatype supplies enough

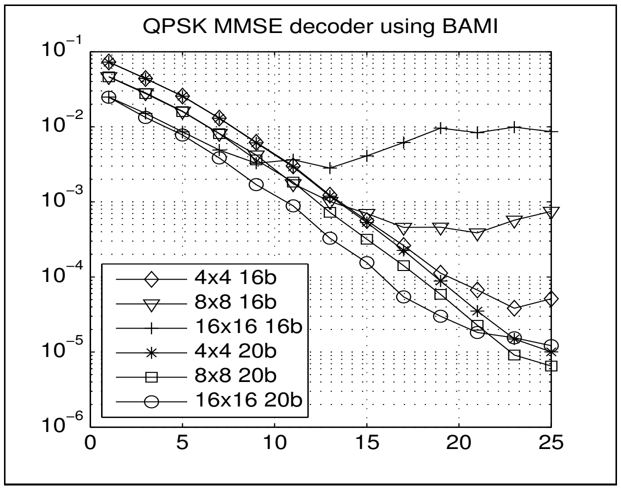

Figure 4. 16 bit vs 20 bit Floating-Point (BAMI)

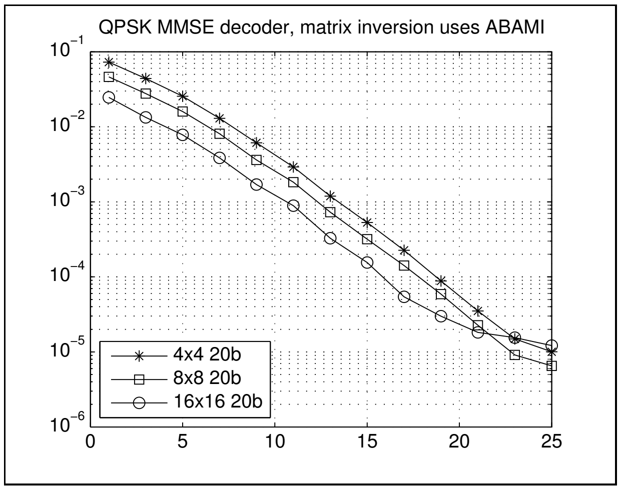

Figure 5. ABAMI using 20bit Floating-Point

precision for the inverse of small matrices, the finite-length error becomes significant when the size of matrix gets larger. For example, as illustrated in Fig. 4, for 16×16 matrix inversion, the BER performance will saturate even if the SNR increases further. In comparison, the 20-bit representation brings sufficient precision for larger matrices. Therefore, it is selected for our implementation.

6.3. Evaluation of Finite-length Error

Fig. 5 shows the MMSE decoding performance (SNR/BER) curves achieved by ABAMI method using 20-bit floating-point datatype. The number of users ranges from 2 (4×4 matrix) to 8 (16×16 matrix). Compared to those in Fig. 4, the curves in Fig. 5 clearly show that ABAMI has the same numerical stability as BAMI, which means 20-bit floating-point can also supply sufficient precision to ABAMI.

7. Hardware Cost

In order to compare our proposal with other solutions, two implementations are carried out. One of them is called SOLO which includes a single FPCMAC, a real divider, and the register file (32 general complex registers and 7 complex accumulation registers). The other is called QUARTET which contains four FPCMAC, one real divider and one separate register file for each FPCMAC. The implementations are carried out in FPGA and ASIC technologies.

7.1. FPGA

SOLO is implemented in Xilinx Virtex4-xc4vlx200 FPGA to be compared with other solutions. The input is a 4×4 matrix of complex floating-point values and output is the inverted matrix. All the basic units such as the floating-point real adder, subtracter, multiplier and divider are generated using Xilinx Core Generator which enables the optimized design based on soft IP Core V.3 from Xilinx. The implementation of the basic units set has 9 pipeline stages in the FPCMAC and takes 8 cycles to finish the real value division.

Compared to the synthesis results of other solutions shown in Table 4, for general complex valued matrices, the BAMI based matrix inversion using SOLO is 3 times faster than the linear array proposed by [5] and around 8 times faster than the solution proposed by [6]. For Alamouti sub-block based matrices, in case ABAMI is used, the performance is almost 4 times faster than [5] and 11 times faster than [6]. Meanwhile, SOLO consumes less than half of the hardware compared to [5] and less than 1/4 of that of [6]. Note that by utilizing the DSP48 hard macros in the FPGA, the number of slices consumed is reduced.

	SOLO		Ref [5]	Ref [6]
FPGA Type	Virtex4		Virtex2	Virtex4
Datatype	floating		fixed	floating
Wordlength (bits)	20		12	20
Num of Slices	1716	2094	4400	9117
Num of DSP48	8	0	N/A	22
Frequency (MHz)	100		100	115
Latency of 4×4 Matrix Inversion (cycles)	BAMI	ABAMI		
	120	85	350	933

Table 4. Comparison of FPGA Implementations for 4×4 Matrix Inversion

		Latency (cycles)		
Size of Matrix		4×4	8×8	16×16
SOLO	BAMI	96	604	4356
	ABAMI	54	320	2220
QUARTET	BAMI	49	210	1216
	ABAMI	45	152	708

Table 5. Latency (cycles) of Matrix Inversion

7.2. ASIC

The ASIC synthesis is done using the Synopsys design flow and 90 nm process from ST Microelectronics. In order to achieve low power, a low leakage library with a supply voltage of 1.08V is used instead of more aggressive libraries.

SOLO consumes 25k gates while QUARTET consumes 90k gates. Both SOLO and QUARTET can run at 500MHz with 3 pipeline stages in the FPCMAC. It takes 3 cycles for the non-pipelined real valued divider to finish its computation. Based on SOLO and QUARTET, the latency of both BAMI and ABAMI for matrices of different sizes is depicted in Table 5. The table clearly shows that the number of cycles saved by exploiting the Alamouti sub-block structure by ABAMI provides a 50% reduction compared to the BAMI case. Although the speed-up achieved by QUARTET against SOLO is trivial for small matrices (e.g. 4×4) because of the low number of operations available for parallel processing, the speed-up grows towards 400% as the size of the matrix increases. Note the overhead of data load/store is implicit here because the large register file and the permutation unit allow data shuffling in parallel to the computation.

8. Conclusion

In this paper, we extended the BAMI (blockwise analytical matrix inversion) method to larger matrices and proposed a specialized and more efficient method (ABAMI) to compute matrix inversion for Alamouti sub-block based STBC matrices. Furthermore, this method is mapped on to a programmable ASIP targeting Software-Defined-Radio system. According to the comparison with other solutions made in Sec. 7, our solution not only achieves higher performance but also consumes less silicon area. In addition, the SDR processor is a purely programmable device driven by instructions, which has much higher flexibility than systolic arrays thus making it suitable for multi-standard baseband processing.

9. Acknowledgment

The work of J. Eilert, D. Wu and D. Liu is supported partly by the Swedish National Foundation of Strategic Research SSF. The work of D. Wang and N. Al-Dhahir is supported in part by NSF contracts CCF04-30654 and DMS05-28010.

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

[3] R. Döhler, *"Squared Givens Rotation"*, IMA Journal of Numerical Analysis, no. 11, pp. 1–5, 1991.

[4] J. Gotze and U. Schwiegelshohn, *"A Square Root and Division Free Givens Rotation for Solving Least Square Problems on Systolic Arrays"*, J. SCI. STAT. COMPUT., vol. 12, pp. 800–807, July 1991.

[5] F. Edman, V. Öwall, *"Implementation of A Full Matrix Inversion Architecture for Adaptive Antenna Algorithms"*, Proc. WPMC'04, 2004.

[6] M. Karkooti, J. Cavallaro, C. Dick, *"FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm"*, Proc. 39th Asilomar Conference on Signals, Systems, and Computers, 2005.

[7] J. Eilert, D. Wu, D. Liu, *Efficient Complex Matrix Inversion for MIMO Software Defined Radio*, submitted to ISCAS2007.

[8] A. H. Sayed, W. M. Younis, A. Tarighat, *"An Invariant Matrix Structure in Multi-Antenna Communications"*, IEEE Signal Processing Letters, 2005

[9] W. M. Younis, A. H. Sayed, and N. Al-Dhahir, *"Efficient adaptive receivers for joint equalization and interference cancellation in multi-user space-time block-coded systems"*, IEEE Transactions on Signal Processing, vol. 51, no. 11, pp. 2849–2862, Nov. 2003

[10] A. Nilsson, E. Tell, D. Liu, *"Simultaneous Multi-standard Support in Programmable Baseband Processors"*, Proc. IEEE PRIME, 2006.