ASIP-based Flexible MMSE-IC Linear Equalizer for MIMO Turbo-Equalization Applications

Atif Raza Jafri, Daoud Karakolah, Amer Baghdadi, Michel Jézéquel
Institut TELECOM; TELECOM Bretagne; UMR CNRS 3192 Lab-STICC
Electronics Department, Technopôle Brest Iroise CS 83818, 29238 Brest
Université Européenne de Bretagne, France
E-mail: {ar.jafri,daoud.karakolah,amer.baghdadi,michel.jezequel}@telecom-bretagne.eu

Abstract

A novel 16-bit flexible Application-Specific Instruction-set Processor for an MMSE-IC Linear Equalizer, used in iterative turbo receiver, is presented in this paper. The proposed ASIP has an SIMD architecture with a specialized instruction-set and 7-stage pipeline control. It supports diverse requirements of MIMO-OFDM wireless standards such as use of QPSK, 16-QAM and 64-QAM modulation in 2×2 and 4×4 spatially multiplexed MIMO-OFDM environment. For these various operational modes, analysis of MMSE-IC LE equations and corresponding complex data representations was conducted. Efficient computational and storage resource sharing is proposed through: (1) Matrix Register Banks (MRB) multiplexing, (2) 16-bit Complex Arithmetic Unit (CAU) comprised of 4 combined complex adder/subtractor/multiplier units, 2 real multipliers, 5 complex adders, and 2 complex subtractors, and (3) flexible 32-bit to 16-bit data conversion at multipliers' output.

With this architecture, the designed ASIP ensures, along with flexibility, high performance in terms of throughput and area. Logic synthesis results reveal a maximum clock frequency of 546 MHz and a total area of 0.37 mm^2 using 90 nm technology. For 2×2 spatially multiplexed MIMO system, the proposed ASIP achieves a throughput of 273 MSymbol/Sec.

1. Introduction

Turbo equalization is a well established concept to jointly solve the equalization and decoding tasks [1][6]. Initially this concept was used in the removal of Inter symbol Interference (ISI) caused by multipath fading using optimal maximum likelihood (ML) algorithm. However, with the use of higher data rates and bigger constellations of modulation, low-complexity suboptimal algorithms such as Zero Forcing (ZF) and Minimum-Mean-Square-Error Interference-Canceller (MMSE-IC) linear equalizers were developed [10][7]. Later on, with the introduction of OFDM and MIMO in new standards of mobile communication, OFDM provides robustness against the ISI caused by multipath while the concept of turbo equalization can be used to remove the interference caused by MIMO channel.

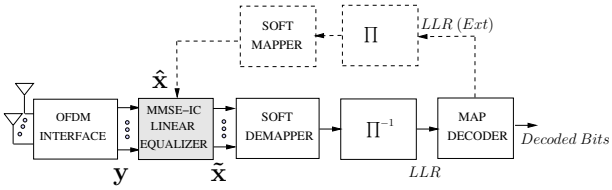


Figure 1. MIMO turbo receiver scheme

A MIMO receiver using turbo equalization is shown in Fig. 1. The symbols corrupted through noise and co-antenna interference are received in the MMSE-IC linear equalizer (LE). The interference is partially cancelled through MMSE-IC LE and the symbols are demapped into Log Likelihood Ratios (LLR) and deinterleaved. The MAP decoder receives these LLR and removes the effect of noise and outputs extrinsic information again in shape of LLR. These LLRs are interleaved and mapped to generate decoded symbols in the feedback path. Decoded symbols serve equalizer as a priori information to improve its output. This iterative process continues till required error rate performance is achieved at the output of the decoder.

Recent wireless standards support the MIMO use in 2×2 and 4×4 spatial multiplexing configuration along with QPSK, 16-QAM and 64-QAM as modulation types. To support such requirements, most of existing work has been focused on the inversion of variable-sized complex-numbered matrices. In [2] the authors focus is on transforming systolic array architecture for matrix inversion into linear array architecture and hence saving computational elements at the cost of control logic. Similar work has been carried out in [8] where authors have proposed a SGR and CORDIC based LMMSE detector. Matrix inversion through block-wise analytical method has been implemented in [3]. The presented processor-based work does not consider a turbo environ-

978-3-9810801-5-5/DATE09 © 2009 EDAA

ment and hence use floating-point arithmetic to attain the required precision. Using analytic method of matrix inversion, a fully dedicated architecture for MMSE-IC LE for 2×2 turbo MIMO system with precoding has been proposed in [5]. It achieves high throughput, however lacks flexibility and induces significant silicon area due to the large number of used computational elements.

To the best of our knowledge, none of the existing works has tackled the complete equalizer design in an turbo context considering flexibility and performance tradeoff. We propose in this paper the first flexible and high performance ASIP model for MMSE-IC LE supporting 2×2 and 4×4 spatially multiplexed turbo MIMO applications using QPSK, 16-QAM and 64-QAM modulations.

The rest of the paper is organized as follows. Next section presents the MMSE-IC LE algorithm. Section 3 details the proposed ASIP architecture model. Section 4 illustrates the designed instruction set while section 5 is dedicated for simulation and synthesis results. Finally section 6 concludes the paper.

2. MMSE-IC LE Algorithm

At the inputs of the equalizer, the received symbol vector $\mathbf{y}$ is given by the following expression.

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \eta \tag{1}$$

where $\mathbf{y}$ is a vector of size of number of receive antennas (N_r), $\mathbf{x}$ is a vector of size of number of transmit antennas (N_t), $\mathbf{H}$ is channel matrix of size N_rxN_t and the η is column vector of Additive White Gaussian Noise(AWGN) of size N_r. The output $\tilde{x}$ of the equalizer using time invariant approximation as proposed by [7] is given by:

$$\tilde{x}_k = \lambda_k \mathbf{p}_k^H (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}} + \hat{x}_k \mathbf{h}_k) \tag{2}$$

where $k = 1, 2, ...N_t$, $\hat{\mathbf{x}}$ is vector of decoded symbols of size N_t and $\hat{x}_k$ is k^{th} element of this vector, $\mathbf{h}_k$ is k^{th} column of $\mathbf{H}$ matrix and $(.)^H$ is Hermitian operator. The other parameters λ_k, and $\mathbf{p}_k$ are given by:

$$\mathbf{p}_k = \mathbf{E}^{-1}\mathbf{h}_k \tag{3}$$

$$\mathbf{E} = (\sigma_x^2 - \sigma_{\hat{x}}^2)\mathbf{H}\mathbf{H}^H + \sigma_w^2\mathbf{I} \tag{4}$$

where σ_x^2, $\sigma_{\hat{x}}^2$ and σ_w^2 are variances of transmitted symbols, decoded symbols and noise. I is identity matrix.

$$\lambda_k = \frac{\sigma_x^2}{1 + \sigma_{\hat{x}}^2 \beta_k} \; where \; \beta_k = \mathbf{p}_k^H \mathbf{h}_k \tag{5}$$

Eq. (2) can be rewritten in the form

$$\begin{aligned} \tilde{x}_k &= \lambda_k \mathbf{p}_k^H (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}) + \lambda_k \mathbf{p}_k^H \hat{x}_k \mathbf{h}_k \\ &= \lambda_k \mathbf{p}_k^H (\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}) + g_k \hat{x}_k \end{aligned} \tag{6}$$

where g_k is equivalent bias of AWGN noise whose real part is used in demapper. In the above mentioned equations, two types of computations can be identified, those which are performed once for whole block of symbols for which channel has been considered as constant (eq. (3) to eq. (5) along with computation of $\lambda_k \mathbf{p}_k^H$ and g_k) and those which are performed repeatedly to estimate each symbol of transmitted block (eq. (6)). In other words these two type of computations use similar resources but not concurrently. Thus, computational resource sharing can be efficiently exploited.

3. ASIP Architecture for MMSE-IC LE

Tradeoffs between ASIC high performance and programmable processor flexibility are achieved by the Application Specific Instruction Set Processor (ASIP) [9]. Besides the possibility of computational resource sharing presented in the previous section, selecting an ASIP approach provides the flexibility required to meet diverse demands of current and future wireless standards.

3.1. Context of Architectural Choices

Regarding architectural choices, an effort has been made to derive basic instructions which can be used to perform complex numbered matrix computations involved in the processing of target algorithm. The derived instructions are further decomposed into micro operations like addition, subtraction and multiplication which are performed in different stages of pipeline. As first step, while transforming the floating point representation of the reference application software model into fixed point, it was found that at maximum 16-bit signed representation with different bits for integer and fractional part is sufficient to represent all the parameters involved during different computational steps of MMSE-IC LE algorithm.

To ensure the reuse of resources for different computations, involving operands with different fixed point representations, certain rules have been set. First of all, while reading input data from memories, the data which is represented in less than 16-bits, is sign extended to 16-bit. Secondly, a programmable 32 to 16-bit conversion is performed at the outputs of multipliers. Last of all, to avoid the hazards caused by overflow/underflow during an arithmetic operation, mechanism is provided to fix the output at its maximum/minimum limit.

3.1.1 Complex Addition / Subtraction / Multiplication

The complex number addition and subtraction/conjugate uses two real adders and subtractors respectively. The implementation of multiplication of two complex numbers is achieved using following expression.

$$(a+bj)(c+dj) = a(c+d) - d(a+b) + \{a(c+d) + c(b-a)\}\, j$$

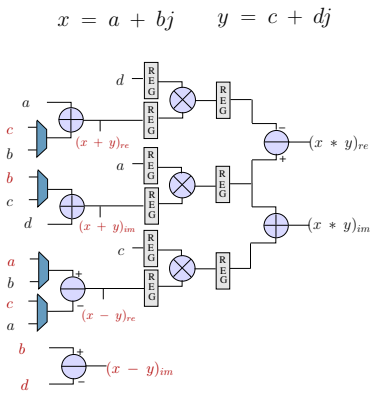


Figure 2. Combined complex adder/subtractor/ multiplier (CCASM)

This process of complex multiplication is spread over three stages. The combined architecture of complex adder, subtractor and multiplier (CCASM) is shown in Fig. 2.

3.1.2 Complex Number Inversion

The inverse of a complex number can be computed using following expression:

$$\frac{1}{a+bj} = \frac{a}{a^2+b^2} - \frac{b}{a^2+b^2}j$$

3.1.3 Matrix Inversion

The expression for the inversion of 2×2 matrix through analytical method is given by:

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

For a 4×4 matrix, the matrix is divided into four 2×2 matrix and inversion can be achieved block wise.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}$$

where

$$W = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}$$
$$X = -A^{-1}B(D - CA^{-1}B)^{-1}$$
$$Y = -(D - CA^{-1}B)^{-1}CA^{-1}$$
$$Z = (D - CA^{-1}B)^{-1}$$

3.2. ASIP Architecture

The ASIP is mainly composed of Matrix Register Banks (MRB), Complex Arithmetic Unit (CAU) and Control Unit

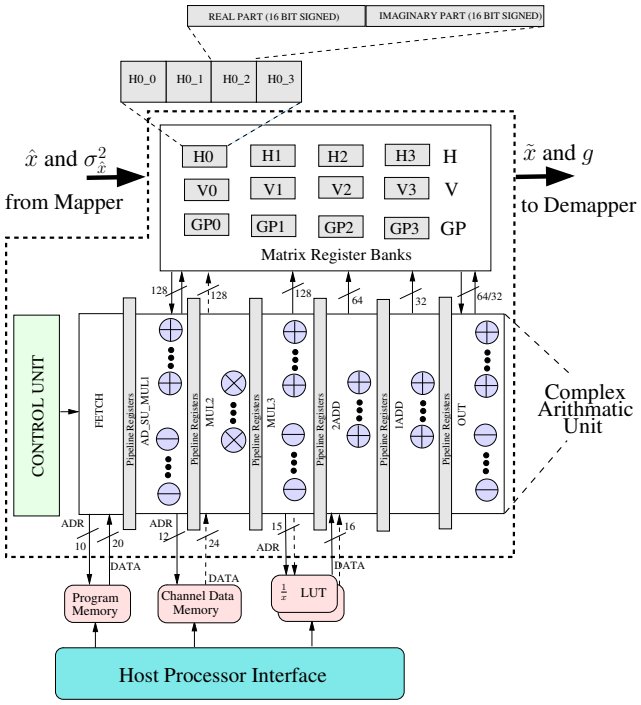
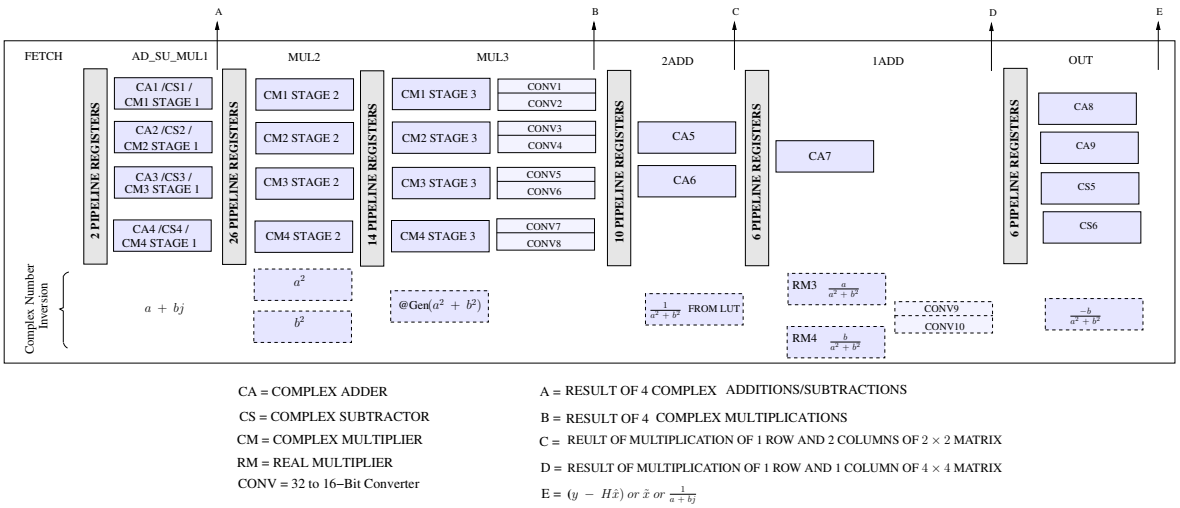


Figure 3. ASIP block diagram

(CU) beside its memory interface. The input to the ASIP are through "Channel Data Memory" and the soft mapper as shown in Fig. 3. The data bus of all inputs is set to 16 (32 bit for complex number). This provides flexibility to use upto 16 bit data representation and in case of smaller data widths, signed/unsigned extension can be done externally. The memories are connected to host processor which is responsible to assign tasks to the ASIP. The ASIP has 7 pipeline stages named as; FETCH, AD_SU_MUL1, MUL2, MUL3, 2ADD, 1ADD and OUT.

3.2.1 Matrix Register Banks

To store a complex number two separate 16-bit registers have been used, one storing the real and the other imaginary part. Based on the requirements of the eq. (6) for a 4×4 spatially multiplexed MIMO system, 12 MRBs have been proposed, where each MRB can store 4 complex numbers (Fig. 3). H-MRB (H0, H1, H2, and H3) which are connected to the memory, can store 4 rows or columns of Channel Matrix. Four V-MRB (V0, V1, V2, and V3) store 16 entries of $\lambda_k p_k$. GP0, GP1,GP2, and GP3 are assigned the storage of g_k , $\hat{x}_k$, y and the estimated symbols $\tilde{x}$ respectively. Other than this specific use, these registers save the intermediate results of computation performed in eq. (3) to eq. 5. Three other registers of 16 bits store the variances of Noise, modulation symbol and decoded symbols ; pipeline registers and registers for REPEAT instruction.



CA = COMPLEX ADDER
CS = COMPLEX SUBTRACTOR
CM = COMPLEX MULTIPLIER
RM = REAL MULTIPLIER
CONV = 32 to 16–Bit Converter

A = RESULT OF 4 COMPLEX ADDITIONS/SUBTRACTIONS
B = RESULT OF 4 COMPLEX MULTIPLICATIONS
C = REULT OF MULTIPLICATION OF 1 ROW AND 2 COLUMNS OF 2×2 MATRIX
D = RESULT OF MULTIPLICATION OF 1 ROW AND 1 COLUMN OF 4×4 MATRIX
E = $(y - H\hat{x})$ *or* $\hat{x}$ *or* $\frac{1}{a+bj}$

Figure 4. CAU and pipeline stages

3.2.2 Complex Arithmetic Unit

The CAU of the ASIP has the computational resources to perform 4 concurrent complex additions, subtractions, complex conjugation and multiplications; and inversion ($\frac{1}{x}$) of a complex number. The overall resources and their use in different pipeline stages is shown in Fig. 4. The resources in each stage of pipeline are arranged in order to be used maximally and efficiently. In MUL3 stage, 32-bit to 16-bit transformation is performed according to the information provided in multiply instruction. The inversion process of a complex number in different pipeline stages is shown as dotted area of Fig. 4. For this particular operation, additional resources are required as Look-Up Tables (LUT), two 32 to 16-bit converters, and two real multipliers.

3.2.3 Control Unit

The ASIP control unit is based on 7-stage pipeline as mentioned above. It controls the flow of the program over the designed datapath (MRBs, CAU) during the different stages of the pipeline.

4. ASIP Instruction Set

The instructions of ASIP is 20 bit long and are categorized as follows.

4.1 LOAD, MOVE, REPEAT, NOP

LOAD instruction is used to load Channel Matrix into H-MRB while MOVE instruction is used to transfer data between MRBs. REPEAT instruction repeats a block of code as many times as given in REPEAT_SIZE Register. NOP instruction is used to add some empty cycles during the execution of program.

4.2 NEGATION, CONJUGATE

In case of NEGATION all four complex numbers of any one of H-MRB are subtracted from zero and transferred to respective V-MRB. In case of CONJUGATE only imaginary part of complex numbers are subtracted from zero. The real parts of complex numbers are saved, as they are, in corresponding places in V-MRB.

4.3 ADD, SUBTRACT

In these instructions any one of H-MRB can be added or subtracted from any one of V-MRB and the result is stored in GP0-MRB.

4.4 MULTIPLY

This category is the most demanding one in the MMSE-IC LE. Different fields of the multiply instruction are detailed in Fig. 5(a). Eight different opcodes fall under this category to use complex multipliers for multiplication of 4×4 and 2×2 matrices, multiplication of 4 complex numbers, 3 different MAC instructions and two instructions to compute the output symbols $\tilde{x}$.

Different possible sources to complex multipliers are shown in the Fig. 5(b). Depending upon the fields "Source1" and "Source2" of the instruction, 4 operands are selected as source1 and 4 as source2 for 4 complex multipliers. To obtain different 16-bit fixed-point representations from 32-bit output of complex multipliers, 32 to 16-bit converters are designed. These converters (Fig. 5(c)) select one of 16 consecutive bits from 32-bit multiplication result depending upon the "16-bit Control" field of the instruction. A combinational logic has also been provided to detect overflow/underflow with each choice of bit selection and consequently saturate the output value to maximum/minimum

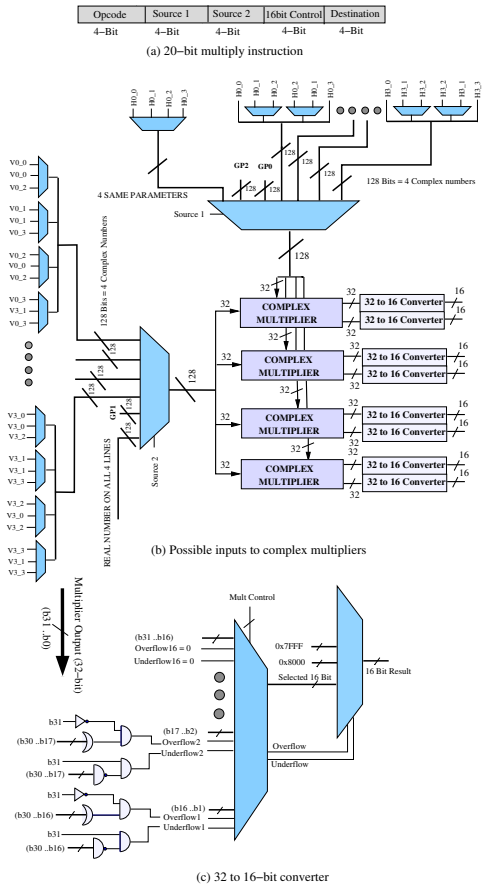


Figure 5. Complex multiplication datapath: (a) 20-bit Multiply Instruction, (b) Possible inputs to complex multipliers, (c) 32 to 16-bit converter

```
LOAD H0_0 0x001
LOAD H0_3 0x004
NOP
CONJ H0 V0 ; H^H in V Banks
;compute first column of HH^H
MULT_2x2 H0 V0_SU 2DOT14 GP0_01
;compute second column of HH^H
MULT_2x2 H0 V0_SL 2DOT14 GP0_23
NOP
NOP
NOP ; HH^H ready in GP0
MOVE GP0 H0
MAC2 H0 SIGMA1 13DOT3 GP0
NOP
NOP
NOP
; E Matrix ready in GP0
MOVE GP0 H0
;Computation of E^-1
;Computation of Modulous of E
MOVE_REV H0 V0
MULT_2x2 H0 V0 16DOT0 GP0_01
;Changing sign and places of E
MOVE_MOD1 H0 V0
```

(a) MIMO 2×2

```
LOAD H0_0 0x001
LOAD H3_2 0x00f
LOAD H3_3 0x010
CONJ H0 V0;H^H in V Banks
CONJ H1 V1
CONJ H2 V2
CONJ H3 V3
;compute first row of HH^H
MULT_4X4 H0 V0 2DOT14 GP0_0
MULT_4X4 H0 V1 2DOT14 GP0_1
MULT_4X4 H0 V2 2DOT14 GP0_2
MULT_4X4 H0 V3 2DOT14 GP0_3
;compute second row of HH^H
MULT_4X4 H1 V0 2DOT14 GP0_0
MULT_4X4 H1 V1 2DOT14 GP0_1
MULT_4X4 H1 V1 2DOT14 GP0_2
MULT_4X4 H1 V1 2DOT14 GP0_3
;first row of HH^H ready in GP0
MOVE GP0 H0
HH^H ready in H Banks
```

(b) MIMO 4×4

Figure 6. Sample programs for (a) 2×2 and (b) 4×4 MIMO equalization

FPGA Synthesis Results(Xilinx Virtex5 xc5vlx330)	
Slice Registers	3,174 out of 207,360 (1%)
Slice LUTs	11,299 out of 207,360 (5%)
DSP48Es	14 out of 192(7%)
Frequency	130 MHz
ASIC Synthesis Results (Synopsis Design Compiler)	
Technology	ST $90nm$
Conditions	Worst Case ($0.9V$; 105^oC)
Area	$0.37mm^2$
Frequency	546 MHz

Table 1. Synthesis Results

bounds. The "Destination" field of instruction selects the destination for the result.

4.5 DIVIDE

This instruction is used to invert a complex number. The first operation during execution of this instruction starts in the third stage of the pipeline to use the real multipliers. LUT have been used to store the inversion values. The overall operation is shown as dotted area of Fig. 4.

5. Simulation and Synthesis Results

In our work we have used the Processor Designer framework from CoWare Inc. which is based around LISA ADL[4] and allows the automatic generation of ASIP software development tools along with VHDL and Verilog codes for hardware synthesis and system integration. Thus, the proposed ASIP architecture was described in LISA ADL. Using this description, software development tools were generated through Processor Designer framework. The application program was written in assembly language for 2×2 and 4×4 MIMO system with QPSK modulation, which was compiled and linked. Later on the results were verified using Processor Debugger. Parts of assembly language programs for two use cases are shown in Fig. 6. Fig. 6(a) shows the computation of E (eq. (4)) and few steps of E^{-1} and Fig. 6(b) shows the steps for computation of HH^H. Flexibility for higher modulation types is achieved through the support of various fixed-point representations required by these modulations.

Using the Processor Generator tool, VHDL description of the ASIP architecture was generated which is synthesized for FPGA and ASIC implementation. The result of synthesis are arranged in Table. 1. Table. 2 shows the number of clock cycles required for the computation of different steps of target algorithm. With these results, a throughput of 273 MSymbol/sec for 2×2 and 145 MSymbol/sec for 4×4 MIMO system, can be achieved at a frequency of 546 MHz.

Most of the available existing solutions in literature

Expression	MIMO 2×2 (Cycles)	MIMO 4×4 (Cycles)
E (Ref. eq. 4)	13	39
E^{-1} (Ref. eq. 4)	18	68
$\mathbf{p}_k$ (Ref. eq. 3)	9	27
β_k (Ref. eq. 5)	7	16
λ_k (Ref. eq. 5)	16	20
$\lambda_k \mathbf{p}_k^H$, g_k (Ref. eq. 5)	7	15
Total (Ref. eq. 3 to 5)	**70**	**185**
Symbol $\tilde{x}$ Throughput (Ref. eq. 6)	4 symbols / 8 cycles	4 symbols / 15 cycles

Table 2. ASIP computation time for MMSE-IC LE equations

Matrix Inversion (Clock Cycles)		
	MIMO 2×2	MIMO 4×4
Ref. [2]	81	350
Ref. [3]	13	90
Our ASIP	18	68
Computation of $\mathbf{p}_k$ Eq. 3 (Clock Cycles)		
Ref. [8]	415	-
Ref. [5]	15	-
Our ASIP	40	144
Operational Frequency MHz (ST 90nm ASIC tech.)		
Ref. [3]	500	
Our ASIP	546	

Table 3. Performance Comparison

present limited contributions to specific parts of MIMO equalization like matrix inversion or parameters like matrix size. An effort has been made to compare certain parameters with available state of the art implementations as shown in Table. 3. Obtained results demonstrate that the proposed solution outperform exiting implementations in terms of flexibility and/or execution time.

6. Conclusion

In this paper, we have presented a novel high throughput ASIP implementing an MMSE-IC linear equalizer for turbo equalization application. Analysis and simulation of MMSE-IC LE equations allowed to identify potential complex-numbered computational resource sharing and different data representations.

Combination of MRB and CAU enables efficient resource sharing through specialized instruction set. Programmable data converters provides required flexibility for 16-bit fixed point arithmetic operations while handling overflow/underflow situations.

Flexibility of the presented ASIP architecture allows its reuse for any 2×2 or 4×4 spatially multiplexed turbo MIMO application with QPSK, 16-QAM, and 64-QAM. The proposed architectural optimizations enable a maximum throughput of 273 MSymbol/sec for 2×2 and 145 MSymbol/sec for 4×4 MIMO systems with an equivalent 85 KGates area. The presented original contribution demonstrates promising results using ASIP approach to implement MMSE-IC LE and promotes its reuse for other MMSE-based applications.

7. Acknowledgments

This work was supported in part by the AFANA project of the French Research Agency (ANR), the Interaccess project, and HEC Pakistan.

References

[1] C. Douillard, M. Jézéquel, C. Berrou, A. Picart, P. Didier, and A. Glavieux. "Iterative correction of inter symbol interference : Turbo equalization". *European Trans. on Telecom.*, vol. 6:pp. 507–511, Sept-Oct 1995.

[2] F. Edman and V. Owall. "A scalable pipelined complex valued matrix inversion architecture". *in Proc. IEEE International Symposium on Circuits and Systems.*, May 2005.

[3] J. Eilert, D. Wu, and D. Liu. "Efficient complex matrix inversion for MIMO Software Defined Radio". *in Proc. IEEE International Symposium on Circuits and Systems.*, 2007.

[4] A. Hoffmann, O. Schliebusch, A. Nohl, G. Braun, and H. Meyr. "A methodology for the design of Application Specific Instruction Set Processor ASIP using the Machine Description Language LISA". *"In proceedings of the ICCAD, San Jose, USA"*, Nov 2001.

[5] D. Karakolah, C. Jégo, C. Langlais, and M. Jézéquel. "Architecture dedicated to the MMSE equilizer of iterative reciver for linerly precoded MIMO systems". *International Conference on Information Communication Technologies: from Theory to Application*, April 2008.

[6] R. Koetter, A. C. Singer, and M. Tuchler. "Turbo Equalization". *IEEE Signal Processing Magazine.*, January 2004.

[7] C. Laot, R. L. Bidan, and D. Leroux. "Low complexity MMSE turbo equalization: A possible solution for EDGE". *IEEE Trans. on Wireless Communication.*, vol. 4(3):pp. 965–974, 2005.

[8] M. Myllyla, J. Hintikka, J. Cavallaro, and M. Juntti. "Complexity analysis of MMSE detector architecture for MIMO OFDM systems". *"in Proc. Asilomar Conf. on Signal, Systems and Computers."*, 2005.

[9] A. Oraioglu and A. Veidenbaum. "Application Specific Microprocessors"(Guest editors introduction). *IEEE Design and Test Magazine*, Jan-Feb 2003.

[10] X. Wang and H. Poor. "Iterative (turbo): Soft interference cancellation". *IEEE Trans. on Commun.*, vol. 47(7):pp. 1046–1061, July 1999.