Implementation of Complex Matrix Inversion using Gauss-Jordan Elimination Method in Verilog

P.Venkata Rao
MTech Student
Department of ECE G.V.P.College of Engg (A)
Visakhapatnam, A.P, India

K.R.K.Sastry
Associate Professor
Department of ECE G.V.P.College of Engg (A)
Visakhapatnam, A.P, India

ABSTRACT

It gives the architecture of an optimized complex matrix inversion using GAUSS-JORDAN (GJ) elimination in Verilog with single precision floating-point representation. The GJ-elimination algorithm uses a single precision floating point arithmetic components and control unit for performing necessary arithmetic operations. The proposed architecture implements the GJ-elimination algorithm for complex matrix element sequentially. Matrix inversion using GJ-elimination improves the frequency when compared with QR Decomposition algorithm. The design is targeted on XC5VLX50T Xilinx FPGA.

Key words

Matrix inversion, Gauss-Jordan Elimination, Floating Point and True Dual Port RAM

1. INTRODUCTION

For high data rate wireless communications we use Orthogonal Frequency Division Multiplexing (OFDM) due to its high spectral efficiency and low computational complexity. Multiple Input Multiple Output (MIMO) systems with OFDM used to improve the channel capacity. In order to remove the effect of the channel on the received signal in MIMO-OFDM systems, matrix inversion is an essential computational algorithm.

In order to implement matrix inversion in Verilog we use architectural features like Block RAM, DSP slices, and floating IP cores from Xilinx which facilitates the design process easy and reduce the design time.

Matrix inversion is complex algorithm that involves many mathematical operations to be done like addition/subtraction, multiplication and divisions. There are many methods available to calculate the matrix inversion. Matrix inversion can be calculated analytically [4] by using the adjoint matrix adj (A), and determinant, det (A) to solve the following equation:

$$A^{-1} = (1/\det A) \times \operatorname{adj}(A) \qquad (1)$$

Adjoint of the square matrix can be calculated by using the transpose of the matrix formed by the cofactors of elements in the determinant. By dividing the adjoint matrix with the determinant we get the required matrix inversion. Calculation of matrix inversion using analytical method is efficient for smaller matrices like 2 x 2, 3 x 3. But for 4 x 4 or large order matrices analytical method is inefficient, because it requires large resources and more number of iterations to be done which reduces the overall performance of the system. As a result we go for other matrix inversion algorithms based on decomposition such as LU, Cholesky and QR methods. In LU decomposition method the input matrix is decomposed into lower triangular and upper triangular matrices. The input matrix is decomposed into the following equation:

$$A = LU \qquad (2)$$

Here L is the lower triangular matrix, U is the upper triangular matrix. From the above equation inverse of the matrix can be calculated by using the following equation:

$$A^{-1} = U^{-1} L^{-1} \qquad (3)$$

QR and Cholesky decomposition methods are similar to the LU decomposition method. Calculation of matrix inversion using these algorithms also presents high complexity for its hardware implementation, as these algorithms use at least two matrix multiplications, in addition to the decomposition. In QR decomposition, the QRD Gram-Schmidt Ortho-normalization method [2] uses square root operations, whereas the QRD Givens-Rotations [3] uses sine and cosine operations.

Gauss-Jordan elimination algorithm is a direct method that requires only three basic arithmetic operations like addition/subtraction, multiplication and division. In this method no matrix multiplication is required and no square root operations are used as in the case of decomposition methods, which in turn significantly reduce the hardware complexity. The Gauss-Jordan algorithm is performed on the floating point numbers [4] , complex floating point numbers. We have implemented complex floating point numbers in Verilog HDL. The input is represented with single precision floating point representation in IEEE 754 format.

The organization of the thesis is as follows. Section 2 discuss the over view of MIMO-OFDM system. In section 3, we give a brief explanation of the Gauss-Jordan elimination algorithm, in section 4 we present the proposed architecture for Gauss-Jordan elimination algorithm. Section 5 gives the implementation results and Conclusions are shown in section 6.

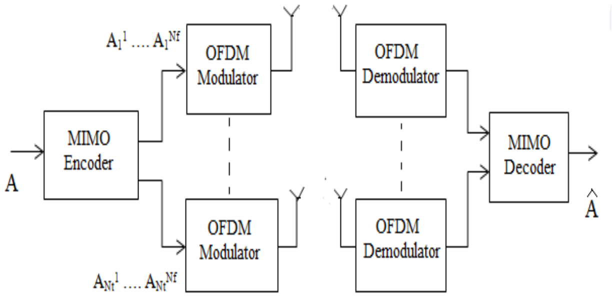

Figure 1: Block diagram for MIMO-OFDM system

2. MIMO-OFDM SYSTEM

A general block diagram for MIMO-OFDM system is shown in Figure 1. The main aim of using OFDM in a MIMO channel is fact that OFDM modulator turns MIMO channel into a set of parallel Frequency-Flat MIMO channels. In an MIMO-OFDM system the individual signals corresponding to the N transmit antennae are OFDM modulated before the transmission. Then a block of data symbols Ns are encoded into a code word matrix A of the order T x N which will then sent through N transmit antennas in T OFDM frames. The transmitted symbols from all the transmitting antennas are represented by A. The matrix B is formed by the subset of S, hence

$$B = \begin{bmatrix} A_1^k \\ \vdots \\ A_{N_t}^k \end{bmatrix}$$

The input data stream is passed through the MIMO channels, the received signal will be first sent to the OFDM demodulator. Without loss of generality, the received vector could be represented as

$$Y = HB + V \qquad (4)$$

Where Y is the received vector with N_r dimension, H is the channel estimation matrix with the order of N_t X N_r and V represents the zero mean complex Additive White Gaussian Noise. In order to get the input data at the output side exactly without any noise, the channel estimation matrix must be inverted. Many equalization algorithms used in MIMO-OFDM system uses the matrix inversion algorithm for acquiring exact input data at the receiver end.

The data symbols $\hat{A}$ is then estimated by linear detection algorithm such as Zero Forcing and is given by

$$B = H^{-1}Y \qquad (5)$$

3. GAUSS- JORDAN METHOD

In order to calculate the inverse for square matrix we use Gauss-Jordan elimination method. In this method the input matrix is to be augmented with the identity matrix in order to perform the matrix inversion. The order of the input matrix and identity matrix must be the same. Gauss-Jordan elimination method is also used to solve system of linear equations. The solution of the linear system A x X=I gives as result $X = A^{-1}$, where X is an n x n matrix of unknowns. In Gauss- Jordan elimination method only row operations are performed to get the required inverse for the given input matrix. Suppose the input matrix is A and the identity matrix I, then the matrix in the augmented form is

$$B = [A\ I] \qquad (6)$$

Then performing row operations to convert the above matrix into the form

$$B = [I\ A^{-1}] \qquad (7)$$

Here A^{-1} is the resultant inverse matrix. The total number iterations required to get the resultant inverse matrix is equal to the dimension n of the square matrix, suppose for 4 x 4 matrix the number of iterations required are 4. In each iteration j operations are done on the augmented matrix [A I] are:

1. For iteration j divide entire row j of the augmented matrix [A I] with the pivot element to get the value of pivot element position as one.

2. Now taking the j row as reference and making the first column as zero by multiplying the row j by each of these elements and subtracting it from their rows respectively until all the elements below the pivot element are zero.

3. Increment j value and the two steps until j=n

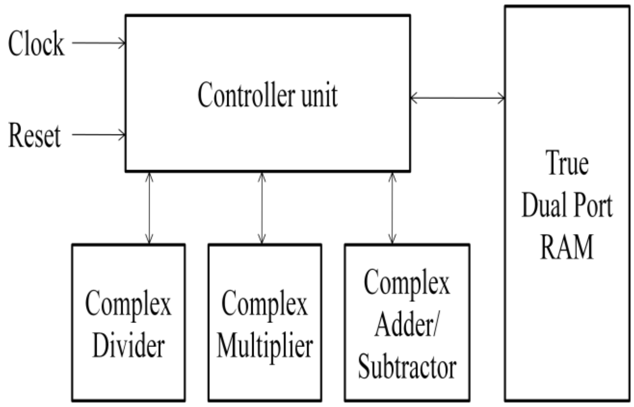

Figure 2: Proposed architecture for optimized GJ-elimination

4. THE PROPOSED ARCHITECTURE FOR GJ-ELIMINATION METHOD

The GJ-elimination method is performed on 4 x 4 input complex matrix it needs 8 complex division operations, 24 complex multiplication operations, and 24 complex addition/subtraction operations. These operations are repeated 4 times to produce resultant inverse matrix. The proposed architecture for GJ-elimination method is shown in Figure 2 consists of:

a. Complex multiplier

b. Complex divider

c. True Dual port Ram

d. Controller unit

In this architecture by using single precision floating point multiplier, floating point adder/subtractor and floating divider Xilinx IP cores we have the complex multiplier and complex divider modules.

a. Complex Multiplier

Complex multiplier in hardware is implemented by using four single-precision floating point multipliers and two single-precision floating point adders. In Verilog HDL complex multiplier is implemented by using floating multiplier and floating point adder IP cores. Consider the multiplication between two complex numbers a1+jb1 and a2+jb2 is shown in the Figure 3.

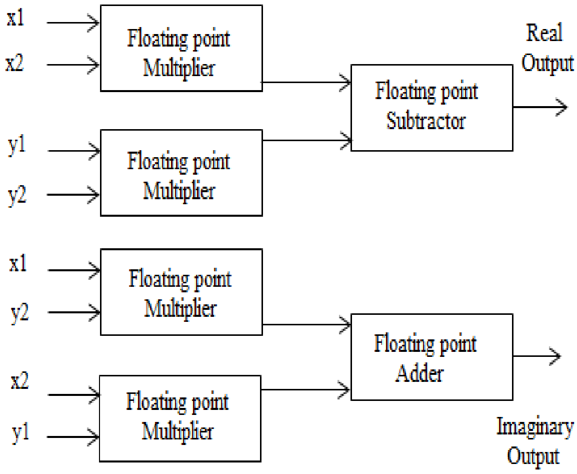
x1
x2
Floating point Multiplier
Real Output
Floating point Subtractor
y1
y2
Floating point Multiplier
x1
y2
Floating point Multiplier
Floating point Adder
x2
y1
Floating point Multiplier
Imaginary Output

Figure 3: Architecture for complex multiplier

Multiplication between the two complex numbers is solve by the equation

$$O = (a1+jb1)*(a2+jb2) \qquad (8)$$

Here a1, b1, a2 and b2 are fractional numbers which are represented in single-precision floating point format.

b. Complex Divider

Complex divider in hardware is implemented by using six single-precision floating point multipliers, three single-precision floating point adders and two single-precision floating point dividers. In Verilog HDL complex divider is implemented by using floating point multiplier, floating point adder and floating point divider IP cores. Consider the division between two complex numbers a1+jb1 and a2+jb2 is shown in the Figure 4.

Division between the two complex numbers is solve by the following equation

$$O =((a2+jb2)/(a1+jb1))*((a1-jb1)/(a1-jb1)) \quad (9)$$

c. True Dual Port Ram:

In order to store the real and imaginary values of the input matrix and the identity matrix during the initialization phase and intermediate phase we use two Block Rams of size 32X32 used for storing the 4X4 matrix.

d. Controller Unit:

Controller unit consists of a finite state machine (FSM) that point to the address of all the elements. The total number of arithmetic operations for each iteration is 8 complex division operations, 24 complex multiplication operations and 24 complex addition operations. Consider the input matrix A, where

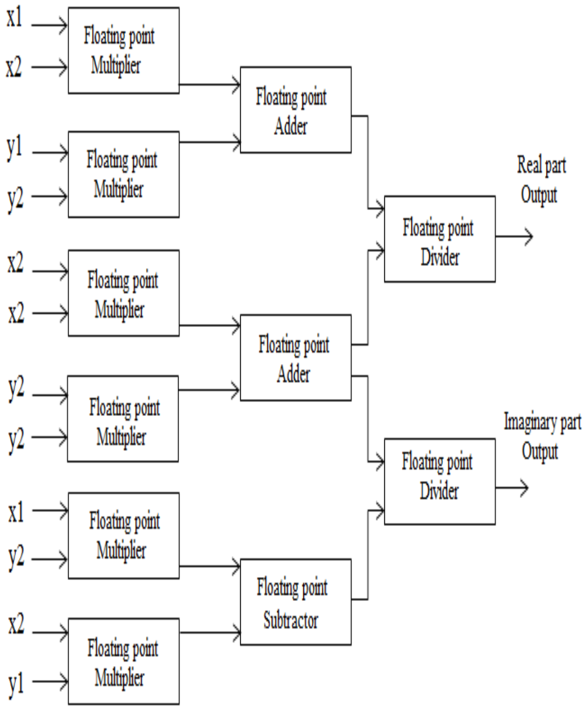
x1
x2
y1
y2
x2
x2
y2
y2
x1
y2
x2
y1
Floating point Multiplier
Floating point Multiplier
Floating point Multiplier
Floating point Multiplier
Floating point Multiplier
Floating point Multiplier
Floating point Adder
Floating point Adder
Floating point Subtractor
Floating point Divider
Floating point Divider
Real part Output
Imaginary part Output

Figure 4: Architecture for complex divider

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

In which a_{ij} is the complex number in row i and column j. the control unit stores the input matrix and identity matrix during the initialization phase in the Block RAM so that the data stored in Block RAM are

$$[AI] = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & 1 & 0 & 0 & 0 \\ a_{10} & a_{11} & a_{12} & a_{13} & 0 & 1 & 0 & 0 \\ a_{20} & a_{21} & a_{22} & a_{23} & 0 & 0 & 1 & 0 \\ a_{30} & a_{31} & a_{32} & a_{33} & 0 & 0 & 0 & 1 \end{bmatrix}$$

5. IMPLEMENTATION RESULTS:

The Gauss-Jordan elimination method is implemented using Xilinx Virtex 5 LX50T device. The architecture was used to implement the complex matrix inverse for 4 x 4 matrix. This implementation is used for any square matrix size, by changing the Block RAM size. The implementation is operating at frequency as the synthesis report shows Clock period: 7.949ns (frequency:125.804MHz). The architecture operates at the higher frequency when compared with the QR decomposition for 32 bit floating point number.

The following table shows the required resources for implementing the architecture for matrix inversion using QR decomposition method and Gauss-Jordan elimination method

Table 1 Resources used for QR and GJ-elimination method for 4 x 4 matrix

Resource	QR decomposition method	GJ-elimination method
Slice LUTs	9117	7993
DSP48E	22	16
IOBs	309	67
Frequency	115 MHz	125.804 MHz

6. CONCLUSION

The implemented architecture can efficiently perform matrix inversion for complex single precision floating point numbers with improved frequency and met the operating frequency 125.804 MHz. Xilinx Virtex 5 LX50T and Xilinx ISE software and XST synthesis tools are used for implementation and resource calculation.

Future scope includes frequency can be further improved by performing GJ-elimination method on particular elements.

7. REFERENCES

[1] Marjan Karkooti, Joseph R. Cavallaro,”FPGA Implementation Of Matrix Inversion Using QRD-RLS algorithm”, IEEE, 2005.

[2] Davide Cescato, Moritz Borgmann, Helmut Bölcskei, Jan Hansen, and Andreas Burg,” Interpolation-based QR decomposition in MIMO-OFDM systems,” IEEE Workshop on Signal Processing Advances in Wireless Communications- SPAWC, 2005.

[3] Zheng-Yu Huang and pei-Yun Tsai,” Efficient Implementation of QR Decomposition for Gigabit

MIMO-OFDM systems," IEEE Trans. on circuits and systems, October 2011.

[4] Ali Irturk, Bridget Benson, Shahnam Mirzaei, Ryan Kastner," An FPGA Design Space Exploration Tool for Matrix Inversion Architectures," SASP 2008.

[5] Janier Arias-Garc´ ia, Ricardo Pezzuol Jacobi, Carlos H. Llanos, Mauricio Ayala-Rinc´ on," A suitable FPGA implementation of floating point matrix inversion based on Gauss-Jordan Elimination," 2011 vii southern conference on programmable logic (SPL), April 2011.

[6] A. Burian, J. Takala and M. Ylinen, "A fixed point implementation of matrix inversion using Cholesky decomposition," Micro-NanoMechatronics and Human Science, 2003 IEEE International Symposium on,vol.3,pp. 1431-1434 Vol. 3, 27-30 Dec.2003.

[7] A. Happonen, O. Piirainen and A. Burian, "GSM channel estimator using a fixed point matrix inversion algorithm," Signals, Circuits and Systems, 2005. ISSCS 2005. International Symposium on , vol.1, no., pp. 119-122 Vol. 1, 14-15 July 2005.