FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm

Marjan Karkooti, Joseph R. Cavallaro
Center for Multimedia Communication, Department of Electrical and Computer Engineering
MS-366, Rice University, 6100 Main St., Houston, TX 77005-1892.
{marjan, cavallar}@rice.edu

Chris Dick
Xilinx Inc., 2100 logic Dr., San Jose, CA, 95124
chris.dick@xilinx.com

Abstract— **This paper presents a novel architecture for matrix inversion by generalizing the QR decomposition-based recursive least square (RLS) algorithm. The use of Squared Givens rotations and a folded systolic array makes this architecture very suitable for FPGA implementation. Input is a 4×4 matrix of complex, floating point values. The matrix inversion design can achieve throughput of $0.13M$ updates per second on a state of the art Xilinx Virtex4 FPGA running at 115 MHz. Due to the modular partitioning and interfacing between multiple Boundary and Internal processing units, this architecture is easily extendable for other matrix sizes.**

I. Introduction

Orthogonal Frequency Division Multiplexing (OFDM) is a popular method for high-rate data transmission in wireless environments. In OFDM, the channel bandwidth is divided into several narrow subbands. The frequency response over each of these subbands is flat. Hence, a frequency-selective channel is transformed into several flat-fading subchannels. The time domain waveforms of the subcarriers are orthogonal, yet the signal spectra corresponding to different subcarriers overlap in frequency. Therefore, the available bandwidth is used very efficiently. The data rate of the system is aggregate of the data rate per subchannel. These features make OFDM suitable for high data rate applications. Another advantage of OFDM systems is that they are less susceptible to various kinds of impulse noise. These characteristics result in reduced receiver complexity.

MIMO (Multiple Input Multiple Output) systems use multiple antennas at both the transmitter and the receiver. Each antenna simultaneously transmits a small piece of data using the same frequency band to the receiver. By taking advantage of the spatial diversity resulting from spatially separated antennas, the receiver can process the data flows and put them back together. This technique utilizes the bandwidth very efficiently.

MIMO channels become frequency-selective during high data-rate transmission due to the multipath characteristics of the environment. By combining OFDM and MIMO, these frequency selective channels can be transformed to a set of frequency flat MIMO channels. Hence decreasing the receiver complexity. Therefore, MIMO-OFDM systems are very promising in broadband wireless systems [1] , [2].

Each receiver in MIMO-OFDM systems should equalize the received signal to remove the effect of channel on the signal. Most of equalization/ detection algorithms need to invert a matrix which is either the channel state information (H) or a nonlinear function of it ($f(H)$). Increasing the number of transmitter and receiver antennas in the system, results in a higher data rate. At the same time, dimensions of matrix $f(H)$ increase, requiring more computations to invert the matrix in less time. This makes the matrix inversion block a bottleneck in these systems.

In this work, we developed an architecture for matrix inversion by generalizing the QR decomposition-based Recursive Least Square algorithm (QRD-RLS) [3]. This algorithm has wide applications in wireless communications and signal processing such as beamforming, channel equalization and HDTV. QRD-RLS is numerically stable and has rapid convergence. It also involves local communication between nodes which is suitable for hardware implementation.

Inverting a matrix using QR decomposition requires a number of rotations to nullify the unwanted values. The standard rotation algorithm called Givens rotation(GR) requires square root operations and divisions that are expensive for hardware. Dohler [4] proposed a square-root-free version of the Givens rotations or "Squared Givens Rotation (SGR)" in 1991. This algorithm eliminates the need for square-root operations and also spares half of the multiplications.

Another algorithm, introduced by Gotze and Schwiegelshohn [5] is the square-root and division free (SDGR) version of the Givens rotations. This algorithm works in a different number realization system and instead of eliminating square roots and divisions, shifts these operations to the final stages of computation. Compared to the SGR, SDGR performs more operations to finish a similar task. Table I compares the number of different operations in each of these algorithms for inverting a 4×4 matrix of real

TABLE I

Number of Operations used by SGR and SDGR for inverting a 4x4 Matrix of real numbers

	Squared Givens rotation	Square root and division-free Givens rotation
Multiplication	27	60
Addition	16	16
Division	10	10
Square root	0	4

numbers.

Another rotation algorithm which is very suitable for fixed-point calculations, is CORDIC (COordinate Rotation DIgital Computer) [6]. It is an iterative algorithm for calculating trigonometric functions such as sine and cosine. This algorithm is a combination of shifts and adds and does not require any multiplications [7], [8], [9],.

The Squared Givens rotation algorithm is used in the proposed architecture design. It has several advantages over the classic Givens rotations and CORDIC. The SGR requires no square roots and half of the multiplications compared to the standard GR. It is much faster than CORDIC. It also can be used with floating point arithmetic which needs 20 percent less number of bits comparing to fixed point and CORDIC arithmetic for the same accuracy [10]. Reference [11] has a comparison between the MMSE detector structures including a matrix inversion core using CORDIC and SGR. The CORDIC-based architecture results in hardware with twice as much area and more than 50% more latency compared to the SGR-based architecture.

Floating point allows for rapid prototyping but has traditionally had higher hardware costs. New generations of FPGA provide faster hardware structures for supporting floating point operations. These pipelined units are very competitive with unrolled iterative algorithms like CORDIC for elementary function evaluation. For applications in which numerical properties of input matrices are general and possibly unknown, rapid prototyping with floating point hardware has great benefits.

The remainder of this paper is organized as follows. The next section review matrix inversion and the QRD-RLS algorithm. The Squared Givens rotation algorithm and systolic array are described in sections III and IV. Architecture design and FPGA implementation will be discussed in sections V and VI. Concluding remarks will be in section VII.

II. Matrix Inversion and QRD-RLS

Inverting a matrix A using Gaussian elimination has a complexity of $O(n^3)$. The complexity of matrix inversion in hardware becomes prohibitive for real time applications and large values of n. Our goal is to invert a matrix of size 12×12 in hardware. In this paper we present the results for inverting a matrix of size 4×4. The same idea and a slight modification in hardware can be used for larger matrix sizes. In the hardware design, we are using QR decomposition and systolic arrays.

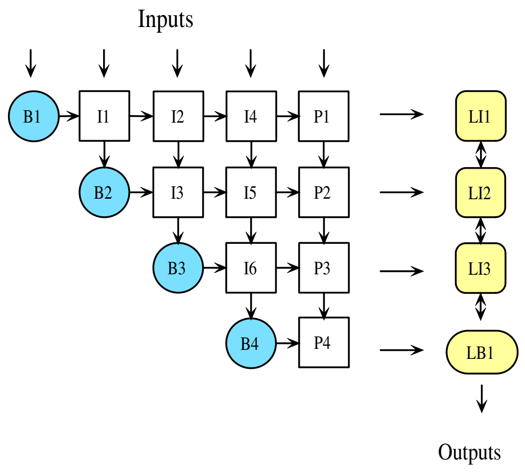

Fig. 1. Systolic Array for QR Decomposition.

Let A be $n\times p$ matrix of full rank p. The QR decomposition is decomposing matrix A to a triangular matrix $R_{p\times p}$ and an orthogonal matrix Q using plane rotations.

$$A = QR \tag{1}$$

Rotation algorithm can be Givens rotation or any of its variations such as SGR, SDGR or CORDIC. Then, finding the pseudo-inverse of matrix A, is equal to :

$$A^{-1} = (A^H A)^{-1} A^H = (R^H R)^{-1} R^H Q^H = R^{-1} Q^H$$

Recursive least square algorithm based on QR decomposition (QRD-RLS), can be used to find the inverse matrix. The main idea of QRD-RLS algorithm is to find a solution for the system of equations

$$Ax = b \tag{2}$$

by minimizing the least square error $min(|b - Ax|)$. This can be done by transforming A to an upper triangular matrix using QR decomposition and systolic arrays and substituting the elements backwards into the equations.

By generalizing the above procedure to p dimensions and solving the equation

$$AX = I \tag{3}$$

in which I is the identity matrix, we can find the inverse of a matrix A, $X = A^{-1}$. The next section presents the SGR, the rotation algorithm used in our architecture.

III. Squared Givens Rotations

Squared Givens rotation was introduced by Dohler. Let U be the upper triangular $p \times p$ matrix which results from triangularization of A by Gaussian elimination. Then $U = D_R R$ and $D_R = diag(R)$ then (1)can be rewritten as:

$$A = Q_A D_U^{-1} U \tag{4}$$

in which $Q_A = QD_R$ and $D_U = diag(U) = D_R^2$. In SGR, matrix U is calculated instead of R. This requires one half of the multiplications compared to the standard Givens rotation and do not require square roots.

Suppose A to be already partially reduced. Let $\mathbf{a} = (0, ..., 0, a_k, ..., a_p)$ be a row whose element $a_k \neq 0$ is to be annihilated by the next standard rotation. We suppose that there is another row $\mathbf{r} = (0, ..., 0, r_k, ..., r_p)$ with a sufficient number of leading zeros. The standard Givens rotation :

$$\begin{aligned} q &= ({r_k}^2 + {a_k}^2)^{1/2} \\ \bar{\mathbf{r}} &= q^{-1}(r_k\mathbf{r} + a_k\mathbf{a}) \\ \bar{\mathbf{a}} &= q^{-1}(-a_k\mathbf{r} + r_k\mathbf{a}) \end{aligned}$$

generates new rows with an additional zero in the kth position of $\bar{\mathbf{a}}$. If we assume to replace the following values in the above equations:

$$\begin{aligned} \mathbf{u} &= r_k\mathbf{r} \\ \mathbf{a} &= w^{1/2}\mathbf{v} \end{aligned}$$

with a given scalar $w > 0$, then the Squared Givens Rotation (SGR) will be:

$$\begin{aligned} \bar{\mathbf{u}} &= \mathbf{u} + wv'_k\mathbf{v} && (5) \\ \bar{\mathbf{v}} &= \mathbf{v} - v_k/u_k\mathbf{u} && (6) \\ \bar{w} &= wu_k/\bar{u}_k && (7) \end{aligned}$$

After completely annihilating A (or V), U will be the upper triangular matrix which is equal to $U = diag(R)R$. Later, this matrix can be used to find the inverse of A by using back substitution.

IV. Systolic Arrays

Use of systolic arrays for matrix triangularization is a well known concept [12]. We adopt the idea and extend it to SGR in our design. Fig. 1 shows a systolic array for the QRD-RLS algorithm. The triangular part consists of two different node types: Boundary cell and Internal cell. These nodes perform rotations on each element in the input matrix to zero out the unwanted elements of the matrix and transform it to an upper triangular form.

The circular *Boundary cell* performs the "vectoring" operation on complex-valued inputs which means that it nullifies the imaginary part of complex numbers and outputs the rotation angle to the internal cells.

The square *Internal cell* "rotates" the input values with the rotation-angle received from the boundary cell in each row. The values move top-down and from left to right in this system. The internal cells in the last column of the triangular part in Fig. 1 update the "b" values in (2).

The nodes in the linear part of the systolic array receive the upper triangular matrix from the triangular part and perform back-substitution to find elements of matrix X in (3). Final outputs are generated in the $LB1$ cell. The next section will describe the designed architecture in more detail.

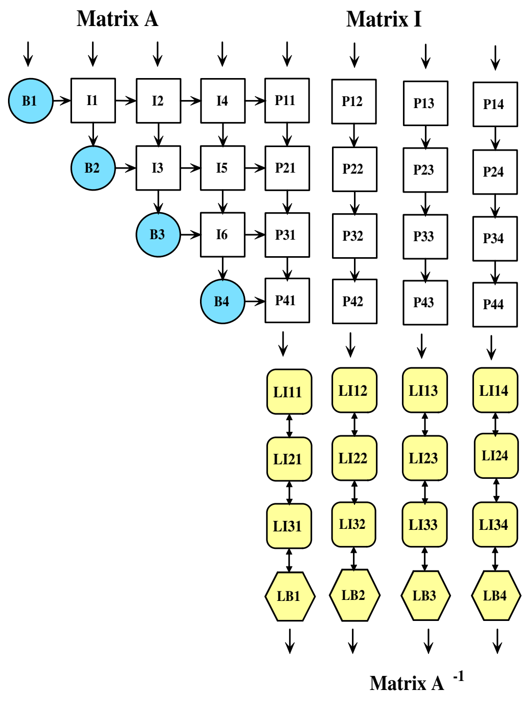

Fig. 2. Systolic Array for 4x4 Matrix Inversion.

V. Architecture Design

The systolic array for inverting a 4×4 matrix is shown in Fig. 2. If we implement every single node in this diagram, it requires a large area and has very high throughput. In our approach, we combined similar nodes, added memory blocks and a scheduler that controls movement of data between nodes (See Fig.3). In this figure, boundary and internal cells are same as the ones in Fig. 1. The back substitution cell is a combination of four cells $LI1_i, LI2_i, LI3_i, LB_i$.

During the initialization step, the input matrix is stored in A-Mem cells. After receiving the first value, the boundary cell starts processing it. Because there is no data dependency at the beginning of the process, boundary cell does not have to wait for the initialization step to finish and these two steps can be pipelined. The boundary cell performs vectorizing on the values and sends the rotation angle to the internal cell. It computes the equations:

$$\begin{aligned} \bar{u} &= u_k + wv'_kv, \\ \bar{w} &= wu_k/\bar{u}_k, \\ c &= v_k/u_k \end{aligned}$$

Fig. 4 shows a block diagram of this cell. Inputs are the values for u and v_k and w and outputs are the values for $\bar{u}, \bar{w}$ and c. The outputs of boundary cells enter the internal cells for more processing. They are also stored in the B-Mem memory cells. Boundary cell and internal cell will be active during the next step working on different sets of input data.

Internal cell runs on the elements 2 to p in each row of the input matrix A and the identity matrix I in (3). It computes

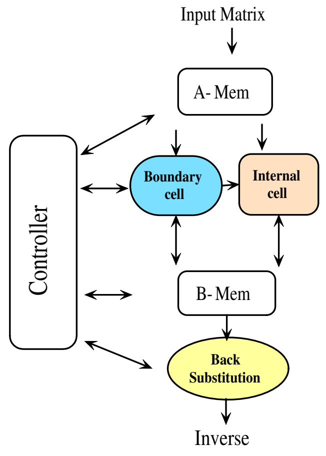

Fig. 3. QRD-RLS Block Diagram.

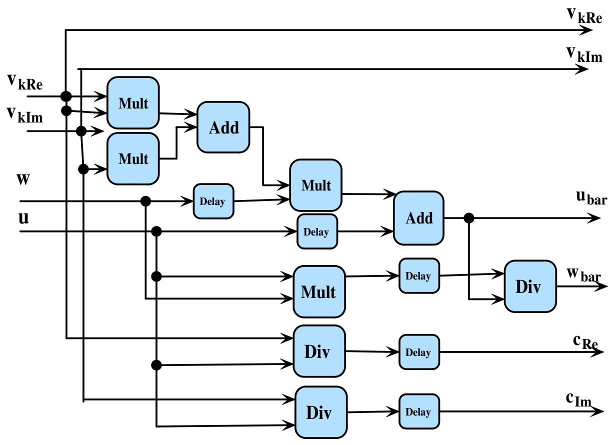

Fig. 4. Boundary Cell Block Diagram.

the following equations:

$$\begin{aligned}\bar{u} &= u_{old} + w v_k' v \\ \bar{v} &= v - c u_{old}\end{aligned}$$

The block diagram of an internal cell is shown in Fig. 5. Inputs are the values for u_{old},v,c,w and v_k and outputs are the values for $\bar{u}, \bar{v}$.

When all the values of triangular matrix U are computed, and the identity matrix is also updated with new rotated elements (P), then values enter the back substitution node. This node computes the values of inverse matrix $X = A^{-1}$:

$$x_{4j} = \frac{p_4}{u_{44}} \tag{8}$$

$$x_{ij} = \frac{(p_i - \sum_{k=i+1}^{4} u_{ik} x_k)}{u_{ii}} \tag{9}$$
$$i = 3..1, j, k = 1..4$$

The final output X or inverse of matrix A is stored in B-Mem memories. The control unit controls the flow of inputs and outputs to each of the nodes and memory blocks of the system.

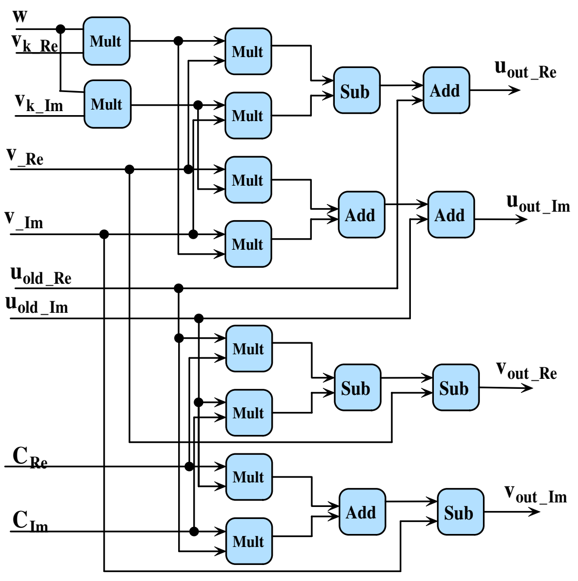

Fig. 5. Internal Cell Block Diagram.

VI. FPGA Implementation

The System Generator [13], a high level design tool from Xilinx is used to implement and test the matrix inversion design on the Virtex4-xc4vlx200 FPGA. Input is a 4×4 matrix of complex, floating point values and output is the inverse matrix. Table II shows the design statistics for a 4×4 matrix inversion core. We assumed 14 bits for mantissa, 6 bits for exponent of floating point numbers and one sign bit. For floating point operators (adder and multiplier) we have used the available operators from Xilinx. Floating point divider is designed in System Generator by using the division IP Core v.3. All these operators are compatible with IEEE754 standard. Table III shows the resources that each of the floating point operators use on FPGA.

On a state of the art Virtex4-xc4vlx200 FPGA running at 115 MHz, this matrix inversion architecture achieves a throughput of $8.1us$ or $0.13M$ updates per second. By pipelining the triangular section and the back substitution part of the design, throughput can increase to $0.15M$ updates per second. The latency for generating the upper triangular matrix is 777 cycles and back substitution has a latency of 156 cycles. These latencies can be decreased by adding more boundary or internal nodes to the design or decreasing the word length requirements. The design is easily extendable to other matrix sizes of $n \times p$ by changing the control unit. There is a tradeoff between number of cells (and hence area of the design) and throughput. For larger matrices, if throughput is less than required, we can increase number of cells and use a semi-parallel approach instead of the current folded model.

TABLE II

RESOURCES FOR 4 BY 4 MATRIX INVERSION CORE ON A VIRTEX4-FPGA

	4×4 Matrix Inversion Core
Slices	9117
DSP48	22
BRAM	9
IOB	309

TABLE III

RESOURCES USED FOR FLOATING POINT OPERATORS ON XILINX FPGA

	Adder	Multiplier	Divider
Slices	433	112	808
DSP48	0	1	0
FFs	303	141	1534
LUTs	503	159	503

VII. CONCLUSIONS

A matrix inversion core is designed and implemented on Xilinx Virtex4 FPGAs using QRD-RLS and Squared Givens Rotation algorithms. The design runs with a clock rate of 115 MHz and achieves a throughput of $0.13M$ updates per second. This design is easily extendable to other matrix sizes.

VIII. ACKNOWLEDGEMENT

This work was supported in part by Xilinx Inc., Nokia Inc., National Instruments, Inc., and NSF under grants EIA-0224458 and EIA-0321266.

REFERENCES

[1] H. Yang, "A road to future broadband wireless access: MIMO-OFDM-Based air interface," *IEEE Communications Magazine*, vol. 43, pp. 53 – 60, Jan 2005.

[2] J. Yue, K. J. Kim, J. Gibson, and R.A.Iltis, "Channel estimation and data detection for MIMO-OFDM systems," in *In Proceedings of IEEE Global Telecommunications Conference*, vol. 2, pp. 581 – 585, 1-5 Dec 2003.

[3] S. Haykin, *Adaptive Filter Theory*. Prentice Hall, third ed.

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

[5] 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.

[6] J. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, 1959.

[7] J. R. Cavallaro and F. Luk, "CORDIC Arithmetic for an SVD Processor," in *Journal of Parallel and Distributed Computing*, June 1988.

[8] N. Hemkumar and J. Cavallaro, "Redundant and Online CORDIC for Unitary Transformations," in *IEEE Transactions on Computers*, vol. 43, pp. 941–954, August 1994.

[9] K. Kota and J. Cavallaro, "Numerical Accuracy and Hardware Tradeoffs for CORDIC Arithmetic for Special-Purpose Processors," in *IEEE Transactions on Computers*, vol. 42, pp. 769–779, July 1993.

[10] R. Walke, R. Smith, and G.Lightbody, "Architectures for Adaptive Weight Calculation on ASIC and FPGA," in *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1375 – 1380, 24-27 Oct 1999.

[11] M. Myllyla, J. Hintikka, J. Cavallaro, M. Juntti, M. Limingoja, and A. Byman, "Complexity Analysis of MMSE Detector Architecture for MIMO OFDM Systems," in *Proceedings of the 2005 Asilomar conference,Pacific Grove, CA*, Oct 30 - Nov 2 2005.

[12] H. K. W.M. Gentleman, "Matrix Triangularization by Systolic Arrays," *Real-Time Signal Processing*, vol. 298, pp. 19–26, 1981.

[13] "www.xilinx.com."