VLSI Architecture for Matrix Inversion using Modified Gram-Schmidt based QR Decomposition

Chitranjan K. Singh, Sushma Honnavara Prasad, and Poras T. Balsara
Center for Integrated Circuits and Systems
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
PO Box 830688, Richardson, TX 75083, USA
Email:{cks022000,shp052000,poras}@utdallas.edu

***Abstract*—Matrix inversion and triangularization problems are common to a wide variety of communication systems, signal processing applications and solution of a set of linear equations. Matrix inversion is a computationally intensive process and its hardware implementation based on fixed-point (FP) arithmetic is a challenging problem. This paper proposes a fully parallel VLSI architecture under fixed-precision for the inverse computation of a real square matrix using QR decomposition with Modified Gram-Schmidt (MGS) orthogonalization. The MGS algorithm is stable and accurate to the integral multiples of machine precision under fixed-precision for a well-conditioned non-singular matrix. For typical matrices (4×4) found in MIMO communication systems, the proposed architecture was able to achieve a clock rate of 277 MHz with a latency of 18 time units and area of 72K gates using 0.18-μm CMOS technology. For a generic square matrix of order n, the latency required is $5n - 2$ which is better than all previously known architectures. With the use of LUTs and log-domain computations, the total area has been reduced compared to architectures based on linear-domain computations.**

I. Introduction

In the current decade Multi-Input Multi-Output (MIMO) systems have generated tremendous research interest as they offer high reliability and high data rate [1]. Due to significant performance gains provided by MIMO, it is being widely adopted in most of the current and next generation wireless communication systems. To exploit the full potential of gains offered by MIMO, computationally efficient design of a wireless baseband communication receiver has become difficult and challenging. Signal processing circuits involved in a MIMO receiver have to be designed for high data throughput and low latency owing to their application in real-time wireless systems. The computational accuracy of signal detection has direct consequence on the throughput and reliability achieved in the receiver.

Matrix inversion and triangularization (QR/LU decomposition) are essential components of all MIMO receivers [2]. Besides MIMO receiver, most of the sub-optimal receiver techniques such as ZF/MMSE channel equalizer, a wide variety of signal processing applications, least square problems and the solution of algebraic linear systems of equations typically involve matrix inversion or matrix triangularization. This also includes inter-carrier interference cancellation in mobile OFDM receivers.

The accuracy and throughput of matrix inversion directly affects performance of such systems. In addition, latency is also a critical design issue. VLSI architectures for matrix inversion are computationally demanding owing to low latency, high accuracy and high throughput requirements. All such earlier VLSI architectures are based on Gauss-Jordan Elimination, LU factorization, Cholesky Factorization [3], Matrix Inversion Lemma [4], or QR decomposition based on Givens Rotations (GR) [5]. QR decomposition using Orthogonalization of vectors based on Classical Gram-Schmidt (CGS) algorithm is numerically unstable [6] for arbitrary matrices.

QR decomposition can also be done using a slightly modified version of CGS, referred to as Modified Gram-Schmidt (MGS) [6], [7] algorithm. This algorithm was proven to be numerically equivalent to Givens Rotations based QR factorization [8]–[10]. The number of operations required in QR decomposition using GR, $2mn^2(1 - n/3m)$, and MGS, mn^2, are almost the same for an $(m \times n)$ matrix A with $m \geq n$. For $m \geq \frac{2n}{3}$, MGS even requires lesser operations than GR. For a nonsingular and not too ill-conditioned matrix, QR decomposition based on MGS algorithm is accurate [9], [10] to the floating-point machine precision. However, to achieve lower power and complexity, most VLSI ASICs are implemented using fixed-point (FP) arithmetic. Due to the quantization and round-off errors in fixed-point architecture, there is loss in accuracy and hence in turn the orthogonality of Q. The orthogonalization error in fixed-point version of MGS algorithm is bounded by the product of condition number $\kappa(A)$ of $(m \times n)$ matrix *A* and machine precision ϵ [9]. This implies that for a well-conditioned matrix, FP architecture for MGS is still accurate to the integer multiples of the machine precision ϵ. This motivated us to work on fixed-point VLSI architecture for matrix inversion based on QR decomposition using MGS algorithm. The discussions in this paper are limited to square matrices of size $(n \times n)$ with real elements, as the focus here is on matrix inversion. For QR decomposition of a non-square matrix of size $(m \times n)$, one can refer to [11].

This paper proposes a fully parallel VLSI architecture for QR decomposition using MGS algorithm and matrix inversion of a $(n \times n)$ square matrix. Our architecture uses the column-oriented version of MGS algorithm involving inner product and norm computation of n-dimensional vectors. The proposed architecture makes use of LUTs for computing multiplication, square and square root involved in inner product and norm computations. The division involved in normalization of column vectors is also done using LUTs. Our method computes matrix inverse in less than $5n$ units of time which is better than all earlier results reported for architectures [5].

The rest of this paper is organized as follows: Section II offers a brief overview of the channel model for MIMO systems and the need for matrix inversion. Section III explains QR decomposition based on CGS and MGS algorithms and describes the matrix inversion procedure using QR decomposition. Section IV presents the proposed architecture for QR decomposition and matrix inversion. Section V discusses the latency, operations involved and the throughput achieved. Section VI concludes with the summary of key results of this paper.

II. MIMO Systems

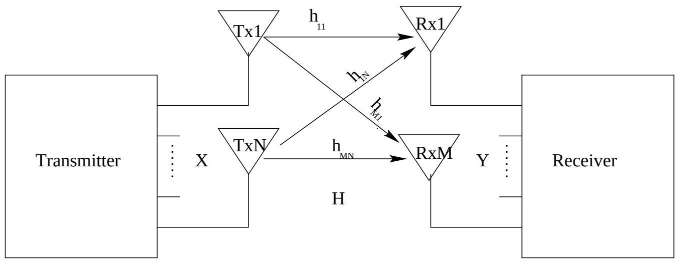

Fig. 1. Block Diagram of a N Transmit and M Receive MIMO System.

The block diagram of a N transmit and M receive antennae MIMO system is shown in Fig.1. The channel model under a flat fading channel condition is given by Eq.1.

$$Y = HX + Z, \tag{1}$$

where, Y is a $(M \times T)$ complex received matrix, H is a $(M \times N)$ complex channel matrix, X is a $(N \times T)$ transmitted matrix whose elements are taken from a complex modulation constellation, and Z is a $(M \times T)$ complex additive white Gaussian noise matrix. Here, T is the number of symbol periods over which data is being transmitted. The complex number format represented by Eq.1 can be decomposed into a real number format as given in Eq.2.

$$y = As + n \tag{2}$$

In Eq.2, $y = [\Re(Y)^T \; \Im(Y)^T]^T$, $s = [\Re(X)^T \; \Im(X)^T]^T$, $n = [\Re(Z)^T \; \Im(Z)^T]^T$ and A is an $(m \times n)$ real matrix,

$$A = \begin{bmatrix} \Re(H) & -\Im(H) \\ \Im(H) & \Re(H) \end{bmatrix}; m = 2M, n = 2N. \tag{3}$$

To estimate the transmitted complex data symbols, Eq.2 can be formulated as Least Squares (LS) problem. The solution to the LS problem can be obtained through suboptimal techniques such as MMSE or ZF equalization or maximum likelihood sphere decoder technique. A good summary for these techniques can be found in [2]. The suboptimal techniques MMSE and ZF require matrix inversion, while the near-optimal technique sphere decoder requires only QR decomposition of *A*. As *M*,*N* increases beyond four, a MIMO system gains very little in performance for a disproportionate increase in receiver complexity. A typical MIMO system has $M, N \leq 4$ as performance gain achieved beyond four antennae is insignificant compared to the increased receiver complexity. The following section discusses the CGS and MGS algorithms.

III. Matrix Inversion Using QR Decomposition

QR decomposition is one of the popular matrix factorization methods. It factorizes a matrix $A_{m\times n}$=$Q_{m\times m}.R_{m\times n}$, where R is an upper triangular matrix and Q is orthogonal in that $Q^H Q = I$, where Q^H is the Hermitian (transpose and conjugate) of Q. This amounts to finding orthonormal basis for ran(A) [6], [7]. QR decomposition can be used to solve full rank least squares problem. The original matrix *A* and decomposed matrices *Q* and *R* for $m = n = 4$ are represented by Eqs.4-6. Throughout this paper a_j, q_j and r_j with $j = 1$ to n denote the column vectors of matrices *A*, *Q* and *R*, respectively. w_j with $j = 1$ to n denote the intermediate column vectors of *Q* used in the CGS and MGS algorithms.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, \tag{4}$$

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}, \tag{5}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{bmatrix}. \tag{6}$$

Gram-Schmidt orthogonalization [6], [7] is a direct method to compute *Q* and *R*. The Gram-Schmidt process for an $m \times n$ matrix *A* proceeds as:

$$q_j = \left(a_j - \sum_{i=1}^{j-1} r_{ij} q_i \right) / r_{jj}, \tag{7}$$

where, q_j and a_j represent column vectors and r_{ij}=$\langle a_j, q_i \rangle$, $i = 1$ to $j - 1$. The next subsections explain in detail the Classical and Modified versions of the Gram-Schmidt Algorithm.

A. Classical Gram-Schmidt Algorithm

The Classical Gram-Schmidt (CGS) algorithm depicted in Fig.2 allows a memory efficient implementation due to its inherent parallelism and the fact that Q can rewrite the original columns of *A*. However, in the presence of round off errors due to fixed-precision computation, orthogonality of Q cannot be guaranteed and may be lost completely too. Due to the loss in orthogonality and accuracy, CGS is not good for practical applications involving finite-precision arithmetic.

```
for $j = 1 : n$;
    $w_j = a_j$
    for $i = 1 : (j-1)$,
        $r_{ij} = \langle a_j, q_i \rangle$
        $w_j = w_j - r_{ij} q_i$
    end
    $q_j = w_j / ||w_j||_2$
    $r_{jj} = ||w_j||_2$
end
```

Fig. 2. CGS Algorithm.

```
for $j = 1 : n$;
    $w_j = a_j$
    for $i = 1 : (j-1)$,
        $r_{ij} = \langle w_j, q_i \rangle$
        $w_j = w_j - r_{ij} q_i$
    end
    $q_j = w_j / ||w_j||_2$
    $r_{jj} = ||w_j||_2$
end
```

Fig. 3. MGS Algorithm.

B. Modified Gram-Schmidt Algorithm

The Modified Gram-Schmidt Algorithm given in Fig.3 overcomes the limitation of CGS by subtracting linear combinations of q_j not directly from *A* but from intermediate w_j before constructing the orthonormal vectors. It was shown in [8] that MGS is numerically superior to CGS. The bounds for loss of orthogonality of vectors $\bar{Q}$ computed in the MGS process as well as the error in factorization $\bar{E}$ is given by,

$$A + \bar{E} = \bar{Q}\bar{R} \text{ and } ||\bar{E}||_2 \leq \zeta_1 \varepsilon ||A||_2 \tag{8}$$

$$||I - \bar{Q}^H \bar{Q}|| \leq \zeta_2 \varepsilon \kappa(A) \tag{9}$$

If $\zeta\varepsilon\kappa < 1$ then $A + \hat{E} = \hat{Q}\bar{R}$, $\hat{Q}^H\hat{Q} = I$ and $||\hat{E}||_2 \leq \zeta\varepsilon||A||_2$, (10)

where, $\zeta_1(m,n)$, $\zeta_2(m,n)$ and $\zeta(m,n)$ are low degree polynomials in m and n depending only on details of computer, ε is the machine precision and $\kappa(A)$ is the condition number of the matrix $A = (a_1, ...a_n)$. Eq.8, derived in [9], indicates that $\bar{Q}\bar{R}$ is a backward-stable factorization of *A*, that is, the product of $\bar{Q}\bar{R}$ represents *A* up to machine precision. The level of orthogonality in $\bar{Q}$ is dependent on $\kappa(A)$, as indicated by Eq.9. If *A* is not too ill-conditioned, then $\bar{Q}$ is orthogonal to machine precision. Eq.10 tells that there exists an exact orthonormal matrix $\hat{Q}$ so that $\hat{Q}\bar{R}$ is a QR factorization of a slightly perturbed *A* under the condition $\zeta\kappa(A)\varepsilon < 1$. If $\zeta\kappa(A)\varepsilon \geq 1$, the matrix *A* is too ill-conditioned and nearly singular. Thus, it can be concluded that MGS based QR factorization gives machine level precision for all well-conditioned matrices.

C. A-inverse computation using QR

When *Q* is orthogonal, that is $Q^H Q = QQ^H = I$, $Q^{-1} = Q^H$. Thus, once matrix *A* has been decomposed to *Q* and *R*, its inverse is computed as

$$A^{-1} = R^{-1} Q^H, \tag{11}$$

```
for $j = 1 : n$;
    for $i = 1 : (j-1)$,
        $ir_{ij} = ir(i, (1:j-1)) * r((1:j-1), j)$
    end
    $ir_{(1:j-1),j} = -ir_{(1:j-1),j} / r_{jj}$
    $ir_{jj} = 1/r_{jj}$
end
```

Fig. 4. Computation of R-inverse.

where, R^{-1} and Q^H are the inverses of *R* and *Q*, respectively. Fig.4 summarizes the algorithm for R^{-1} computation.

The matrix multiplication process is known to be stable. The stability of R^{-1} has been discussed in [5]–[7], where the authors have proved that computation of R^{-1} through the process given in Fig.4 is stable. Thus, A^{-1} computation based on MGS algorithm is stable.

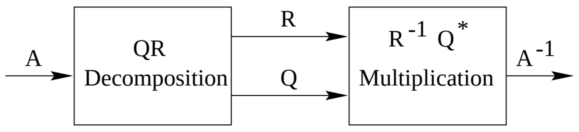

Fig. 5. Top Level Block Diagram of a Matrix Inverse.

IV. Architecture for Matrix Inversion Using MGS

Fig.5 gives a top level overview of the proposed architecture for a square matrix $A_{n \times n}$ with real elements. There are three major steps in the matrix inversion process: QR decomposition, R^{-1} computation and multiplication of matrix R^{-1} with Q^T. QR decomposition in turn involves computation of norms and inner products of n-dimensional vectors, division of n-dimensional vectors by norm (a scalar) and scaling of normalized columns of *Q* by the inner product (a scalar). R^{-1} computation using the procedure in Fig.4 involves multiplications and additions of scalars. Since *Q* is an orthogonal matrix, its inverse is just Q^T. The multiplication of R^{-1} and Q^T as given in Eq.13 (example with $n = 4$ has been chosen) is done using a systolic array to achieve maximum parallelism. Furthermore, in order to minimize latency, computation of R^{-1} and the $R^{-1}Q^T$ is started as soon as the intermediate results are ready.

$$R = \begin{bmatrix} ir_{11} & ir_{12} & ir_{13} & ir_{14} \\ 0 & ir_{22} & ir_{23} & ir_{24} \\ 0 & 0 & ir_{33} & ir_{34} \\ 0 & 0 & 0 & ir_{44} \end{bmatrix} \tag{12}$$

$$A^{-1} = R^{-1}Q^T$$

$$= \begin{bmatrix} ir_{11} & ir_{12} & ir_{13} & ir_{14} \\ 0 & ir_{22} & ir_{23} & ir_{24} \\ 0 & 0 & ir_{33} & ir_{34} \\ 0 & 0 & 0 & ir_{44} \end{bmatrix} \begin{bmatrix} q_{11} & q_{21} & q_{31} & q_{41} \\ q_{12} & q_{22} & q_{32} & q_{42} \\ q_{13} & q_{23} & q_{33} & q_{43} \\ q_{14} & q_{24} & q_{34} & q_{44} \end{bmatrix} \tag{13}$$

The proposed architecture is composed of processing elements to do norm, inner product, scaling of vector by inner product, division of vector by scalar and w_j (intermediate

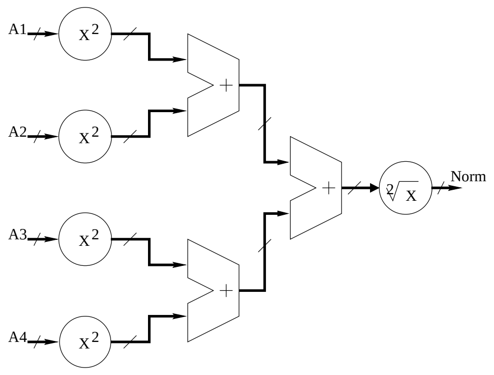

Fig. 6. Norm computation of a 4-dimensional vector in linear-domain.

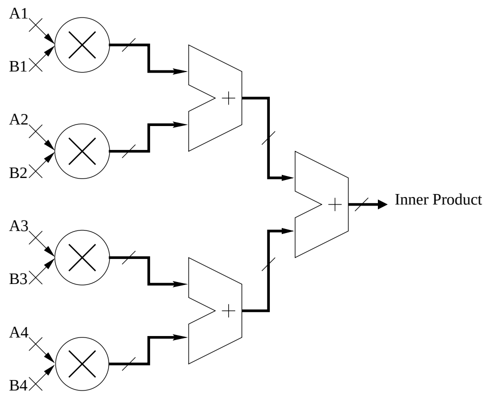

Fig. 7. Inner Product of two 4-dimensional vectors in linear-domain.

column vector of Q) update. Figs.6,7 illustrate the basic architectures for norm and inner product computation, respectively for $n = 4$. These computations involve tedious arithmetic operations like multiplication and square root. Direct implementation of these operations is not only hardware intensive but also increases the latency of the system. To remove this bottleneck, the proposed architecture uses an LUT based approach throughout the design to compute $log_2(x)$ and 2^x. That is, multiplication, division, square and square root in linear-domain are translated to addition, subtraction, shift left and shift right, respectively in log-domain. Table I summarizes translation from linear operations to log-domain operations using the LUT based approach.

After the end of log-domain calculations, the results are translated into linear domain using 2^x LUTs. LUTs have

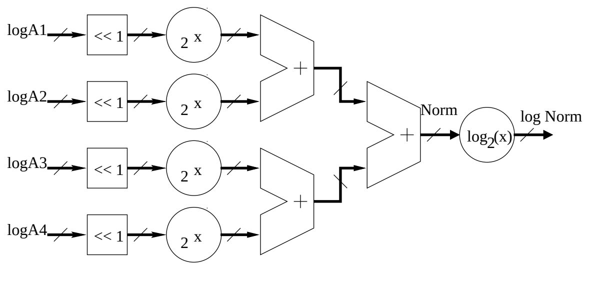

Fig. 8. Norm computation of a 4-dimensional vector in log-domain.

TABLE I
ARITHMETIC OPERATIONS USING LUTS

$x \times y$	$2^{(log_2(x)+log_2(y))}$
x/y	$2^{(log_2(x)-log_2(y))}$
x^2	$2^{(2log_2(x))}$
$\sqrt{x}$	$2^{(log_2(x)/2)}$

limited area overhead due the fact that LUTs can be stored in ROMs. This small overhead in area is largely compensated by avoiding the use of multipliers, dividers and square root. In this design, a word length of 16 bits was chosen with a fractional precision of 10 bits for the data path. The log LUT size was chosen to be 64×15 and the exp LUT size was 256×15. The discussion on optimum LUT size and optimum word length and fractional precision can be found in [11]. To restrict the size of log LUTs, only six fractional bits are taken as address lines to the LUT. This is achieved by normalizing the log argument in the range [0,1) and re-normalizing the log value after accessing the LUT. In order to perform norm and inner product computations in a single time unit, *n* log and *n* exp LUTs have been used. All the adders and subtractors in the design were implemented using $2's$ complement arithmetic. The architectures for norm and inner product computation in log domain are given in Figs.8 and 9, respectively.

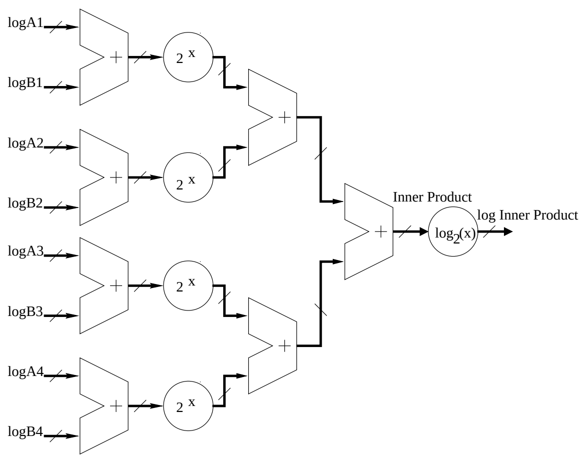

Fig. 9. Inner Product of two 4-dimensional vectors in log-domain.

The Up/Down scaling of a vector by a scalar requires only addition or subtraction in log-domain. For each QR decomposition, there are in total *n* norms, *n* divisions of vector by norm, $\frac{n(n-1)}{2}$ inner products and $\frac{n(n-1)}{2}$ scaling operations of a vector by inner product. Inverse computation of the upper triangular matrix *R* requires sign negation of *n* scalars and $\frac{n(n-1)}{2}$ ir_{ij} computations involving only additions/subtractions in log-domain. The last stage of matrix multiplication requires scaling of $\frac{n(n-1)}{2}$ vectors by scalars. The number of operations involved in each of the processing elements as well as the QR decomposition and matrix inversion using the LUT based approach are summarized in Tables

TABLE II

NUMBER OF OPERATIONS INVOLVED IN LOG-DOMAIN PE

Processing Element	ADD/SUB	LOG LUT Access	EXP LUT Access
Inner Product	$n + (n-1)$	1	n
Norm	$0 + (n-1)$	1	n
Scaling by Inner Product	$n + 0$	0	0
Division by Norm	$n + 0$	0	0
w_j Update	$0 + n$	n	n

TABLE III

NUMBER OF OPERATIONS FOR MATRIX INVERSION

Operation	QR Decomposition	R^{-1}	$R^{-1}Q^*$ Multiplication
ADD/SUB	$(4n-1)[\frac{n(n-1)}{2}] + (2n-1)n$	$\frac{n(n-1)(n+1)}{3}$	$2n[\frac{n(n-1)}{2}] + n^2$
LOG LUT Access	$(n+1)[\frac{n(n-1)}{2}]+n$	$\frac{(n-1)(n-2)}{2}$	0
EXP LUT Access	$2n[\frac{n(n-1)}{2}] + n^2$	$\frac{(n-1)(n-2)(n+3)}{6}$	$n[\frac{n(n-1)}{2}]+ n^2$

II and III for an $(n \times n)$ matrix. Note that in column two of Table II, the first term indicates the number of log-domain ADD/SUBs and the second term the number of linear-domain ADD/SUBs.

V. ANALYSIS AND RESULTS

A. Comparative Analysis of Linear and Log Domain Computations

Norm computation requires n square computations followed by $log_2 n$ stages of adders. The LUT based approach requires $n-1$ additions and includes a single log LUT access and n exp LUT accesses. Inner product of real vectors requires n multiplications and $log_2 n$ stages of adders. Again, the LUT based approach reduces the computation to $2n-1$ additions, a single log LUT and n exp LUT accesses, respectively. Scaling a vector with the inner product works out to be n multiplications which correspondingly translates to n additions using the LUT approach. In order to compute the unit vector, vector division by norm is required. This amounts to n additions in the described architecture. The intermediate w_j-update in the QR decomposition process requires n subtractions. Using LUTs at this stage works out to be more expensive owing to the fact that there is an overhead involved in conversions between log and linear domains.

B. Implementation Results for a 4×4 MIMO Channel Matrix

The number of operations involved in the entire matrix inversion process starting from QR to final matrix multiplication is 202 additions/subtractions, 111 exp LUT accesses and 29 log LUT accesses. It can be observed from Table III that the matrix inversion process requires computations of cubic order, $O(n^3)$. The data flow for the matrix inversion process for a 4×4 matrix is illustrated in Fig.10. By analyzing the data flow, it is found that the delay between two successive inner product computations is 3 time units. Since the last column of *R* has (*n*-1) inner products, which is the maximum for any column, the total time required for computing all the inner products for the last column will be $3(n-1)$ time units. In addition, the final norm and unit vector computation for the last column requires two additional time units. Each preceding (*n*-1) columns of *R* require 2 more time units giving rise to $2(n-1)$ extra time units. Thus, the total latency of matrix inversion is $3(n-1)+2(n-1)+2+1 = 5n-2$ time units, where the last time unit comes from the final computation of $ir_{j4} \times q_4$ products. The latency of the proposed architecture, $5n-2$ is the best among all previously known architectures. The ASIC implementation of the proposed architecture using 0.18-μm CMOS technology required an area of 72K gates and was able to run at a clock speed of 277 MHz with a latency of 67 clock cycles. For a wireless system with block fading channel, channel estimation and pre-processing such as QR-decomposition and matrix inversion of the channel matrix are done only once every block-length of data. The latency of $0.24us$ from the proposed architecture is much within the time allocated for channel processing which is in ms for a typical block fading channel in current wireless standards. For such applications, throughput of matrix-inversion is irrelevant owing to the fact that this operation needs to be performed only once per block-length of data. However, for the record, the throughput achieved by the proposed design is $1.2Gbps$.

VI. CONCLUSION

A fully parallel VLSI architecture for matrix inversion using QR decomposition based on Modified Gram-Schmidt algorithm has been proposed. This matrix inversion process is stable for a well-conditioned non-singular matrix and is accurate to the machine precision in floating-point arithmetic and to the integral multiples of machine precision under fixed-precision arithmetic. The proposed architecture was able to achieve a clock rate of 277 MHz with a latency of 18 time units and an area of 72K gates for a 4×4 matrix using 0.18-μm CMOS technology. For a generic square matrix of order *n*, the latency required is $5n-2$ which is better than all previously known architectures. With the use of $log_2(x)$ and 2^x LUTs and log-domain computations, the total area has been reduced compared to architectures based on linear-domain computations. Although the architecture for matrix inversion proposed in this paper is intended for MIMO systems, it can be used in applications covering communication systems, signal processing and least-square problems. In future work, we propose to compare the inversion of a complex matrix, of order N with a real matrix of order $n = 2N$ for area and latency.

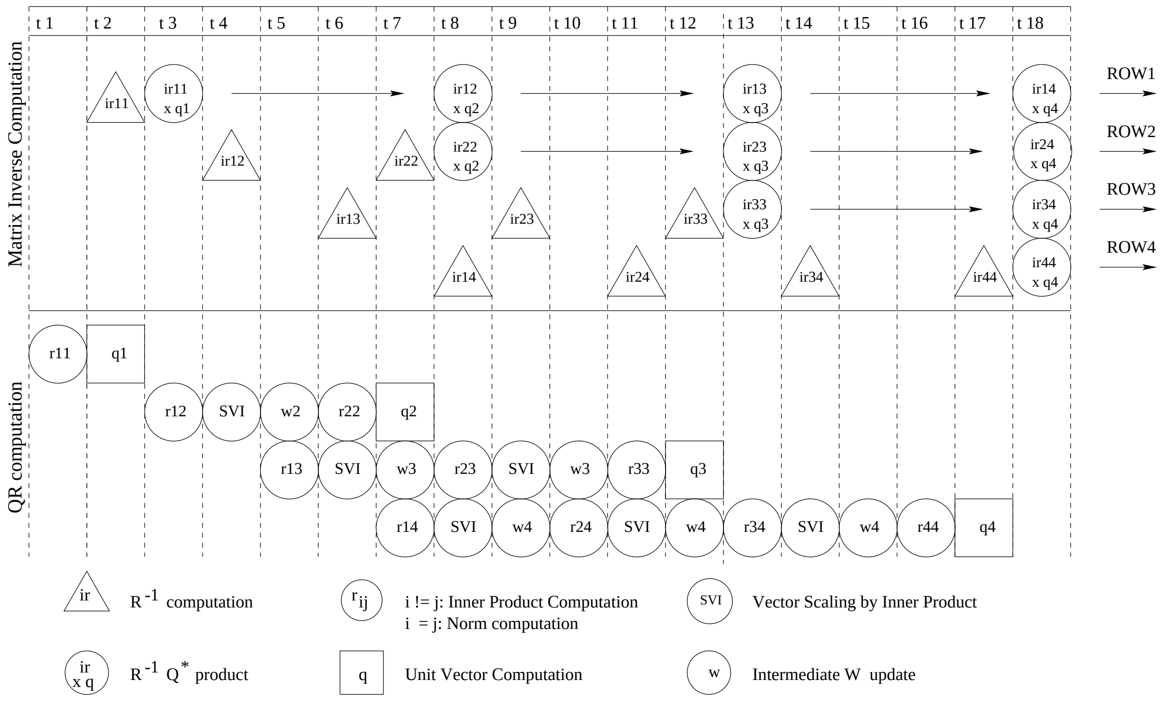

Fig. 10. Data Flow Diagram for Matrix Inversion.

References

[1] I. Telatar, "Capacity of multi-antenna gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov./Dec. 1999.

[2] T. Kailath, H. Vikalo, and B. Hassibi, "Mimo receive algorithms," in *in Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Cambridge University Press, 2005.

[3] A. Burian, J. Takala, and M. Ylinen, "A fixed-point implementation of matrix inversion using cholesky decomposition," in *Proc. IEEE Int. Midwest Symp. Circuits Systems*, Midwest, Dec. 2003, pp. 1431–1434.

[4] I. LaRoche and S. Roy, "An efficient regular matrix inversion circuit architecture for mimo processing," in *Proc. IEEE Int. Symp. Circuits Systems*, Greece, May 2006, pp. 4819–4822.

[5] A. El-Amawy and K. R. Dharmarajan, "Parallel vlsi algorithm for stable inversion of dense matrices," *IEE Proceedings on Computers and Digital Techniques*, vol. 136, no. 6, pp. 575–580, Nov. 1989.

[6] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: John Hopkins University Press, 1996.

[7] A. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, Pa: SIAM, 1996.

[8] ——, "Solving linear least squares problems by gram-schmidt orthogonalization," *BIT 7*, pp. 1–21, 1967.

[9] A. Björck and C. Paige, "Loss and recapture of orthogonality in the modified gram-schmidt algorithm," *SIAM J. Matrix Anal. Appl.*, vol. 13(1), pp. 176–190, 1992.

[10] A. Björck, "Numerics of gram-schmidt orthogonalization," *Linear Algebra and Its Applications*, vol. 198, pp. 297–316, 1994.

[11] C. K. Singh, S. H. Prasad, and P. T. Balsara, "A fixed-point implementation of qr decomposition," in *Proc. IEEE Dallas Workshop Circuits Systems*, Dallas, TX, Oct. 2006, to appear.