Automatic Generation of Decomposition based Matrix Inversion Architectures

Abstract— Matrix inversion is an essential computation for various algorithms which are employed in multi-antenna wireless communication systems. FPGAs are ideal platforms for wireless communication; however, the need for vast amounts of customization throughout the design process of a matrix inversion core can overwhelm the designer. Decomposition methods provide the analytic simplicity and computational convenience necessary for computationally intensive matrix inversion. This paper presents automatic generation and optimization of different decomposition based matrix inversion architectures using a matrix inversion core generator tool, GUSTO, that is developed to enable easy design space exploration with different parameterization options. We present automatic generation of a variety of general purpose matrix inversion architectures and optimized application specific architectures which have comparable results to published matrix inversion architecture implementations, but offer the advantage of providing the designer the ability to study the tradeoffs between architectures with different design parameters.

I. INTRODUCTION

Matrix inversion is a common function found in many algorithms used in wireless communication systems. For example MIMO-OFDM systems use matrix inversion in equalization algorithms to remove the effect of the channel on the signal [1][2][3]. minimum mean square error algorithms for pre-coding in spatial multiplexing [4] and detection-estimation algorithms in space-time coding [5]. These systems often use a small number of antennas (2 to 8) which results in small matrices to be inverted and/or decomposed. For example the 802.11n standard [6] specifies a maximum of 4 antennas on the transmit/receive sides and the 802.16 [7] standard specifies a maximum of 16 antennas at a base station and 2 antennas at a remote station.

Matrix inversion is a computationally intensive calculation. Decomposition methods provide a means to simplify this computation. There are different decomposition methods, such as QR, LU and Cholesky, that solve matrix inversion. The selection of the decomposition method depends on the characteristics of the given matrix. For non-square matrices or when simple inversion to recover the data performs poorly, the QR decomposition is used to generate an equivalent upper triangular system, allowing for detection using the sphere decomposition or M-algorithm. For simpler detection via inversion of square channel matrices, the LU and Cholesky decompositions are compatible with positive definite and nonsingular diagonally dominant square matrices, respectively.

FPGAs are an ideal platform for wireless communication [8][9][10] due to their high processing power, flexibility and non recurring engineering (NRE) cost. However, FPGAs require vast amounts of customization throughout the design process and few tools exist which can aid the designer with the many system, architectural and logic design choices. Designing a high level tool for fast prototyping matrix inversion architectures is crucial.

For automatic generation and optimization of different matrix inversion architectures, we designed an easy to use tool, GUSTO (“General architecture design Utility and Synthesis Tool for Optimization”). GUSTO is the first tool of its kind to provide automatic generation and optimization of a variety of general purpose matrix inversion architectures with different parameterization options. It also optimizes the general purpose architecture to improve its area results and design quality which results in a scheduled, static, application specific architecture. GUSTO allows the user to select the matrix inversion method, the matrix dimension, the type and number of arithmetic resources, the data representation (the integer and fractional bit width), and the different modes of operation for general purpose or application specific architectures.

Our major contributions are:
- Automatic generation and optimization of decomposition based matrix inversion architectures with parameterized matrix dimensions, bit widths, resource allocation, modes and methods.
- Comparison of different decomposition based matrix inversion methods, QR, LU and Cholesky, in terms of different matrix dimensions, bit widths and parallelism.
- Detailed study of area, timing and throughput tradeoffs using different parameterizations.

The rest of the paper is organized as follows. In section II, we introduce MIMO systems, matrix inversion and its different matrix decomposition based solution methods: QR, LU and Cholesky. In section III, we introduce our tool and describe the optimizations performed: static architecture generation and trimming for optimization. Section IV presents our implementation results in terms of area and throughput and compares our results with previously published work. We conclude in Section V.
II. MIMO SYSTEMS, MATRIX INVERSION AND ITS METHODS

Orthogonal Frequency Division Multiplexing (OFDM) is a promising technology for high data rate wireless communications due to its robustness to frequency selective fading, high spectral efficiency, and low computational complexity. Multiple Input Multiple Output (MIMO) systems, which improve the capacity and performance of wireless communication by using multiple transmit and receive antennas, are often used in conjunction with OFDM to improve the channel capacity and mitigate intersymbol interference (ISI) [11].

Therefore, the detection problem becomes a Least Squares (LS) solution to a system of linear equations. Several different MIMO receive algorithms are employed for optimal detection of the transmitted signal [12]. The sphere decoding algorithm offers an exact method. However, tight timing constraints often make it infeasible to wait for the exact solution, and therefore heuristic algorithms are often used. Many heuristic algorithms employ matrix inversion, and therefore, matrix inversion is an essential computation for MIMO systems.

The inverse of a square matrix $A$ is shown as $A^{-1}$ such that

$$A \times A^{-1} = I$$

where $I$ is the identity matrix. Matrix inversion for $4 \times 4$ matrices can be seen as:

$$\begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{pmatrix}^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} \times \begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{pmatrix}^{-1}$$

Explicit matrix inversion of a full matrix is a computationally intensive method. If the inversion is encountered, one should consider converting this problem into an easy decomposition problem which will result in analytic simplicity and computational convenience. Below we describe three known and widely used decomposition methods and their properties to perform matrix inversion: QR, LU and Cholesky decomposition methods [13]. For square matrices, $n$ denotes the matrix size of the matrix such that $n = 4$ for $4 \times 4$ matrices. For rectangular matrices, $m$ and $n$ denote the number of rows and columns in the matrix respectively such that $m = 3, n = 4$ for $3 \times 4$ matrices.

- **QR**: Given $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = n$, QR factorization exists as $A = Q \times R$ where $Q \in \mathbb{R}^{m \times m}$ has orthonormal columns and $R \in \mathbb{R}^{m \times n}$ is upper triangular.

- **LU**: Given $A \in \mathbb{R}^{m \times n}$ with $\det(A(1:k, 1:k)) \neq 0$ for $k = 1 : n-1$, LU decomposition exists as $A = LU$. If LU decomposition exists and the given matrix, $A$, is nonsingular, then the decomposition is unique and $\det(A) = u_{11} \ldots u_{nn}$.

- **Cholesky**: Given a symmetric positive definite matrix, $A \in \mathbb{R}^{n \times n}$. Cholesky decomposition exists as $A = G \times G^T$ where $G \in \mathbb{R}^{n \times n}$ is a unique lower triangular matrix with positive diagonal entries.

where a matrix $A \in \mathbb{R}^{m \times n}$ is positive definite if $x^T A x > 0$ for $x \in \mathbb{R}^n$ and $x \neq 0$ where if $A$ is symmetric positive definite matrix then $A^T = A$. A positive definite matrix is always nonsingular and its determinant is always positive.

Note that Cholesky and LU decompositions work only with positive definite and nonsingular diagonally dominant square matrices, respectively. On the other hand, QR decomposition is more general and can be applied to any matrix. We further explain these decomposition methods, their characteristics and
algorithms, the resulting matrices and the solution steps for matrix inversion in the next subsections.

A. Matrix Inversion of Triangular Matrices

Triangular matrix inversion is used in all of the decomposition based (QR, LU and Cholesky) matrix inversion architectures described above and we use this subsection to describe why this inversion is relatively simple and therefore not a dominant calculation in any of these methods. Primarily, triangular matrix inversion requires fewer calculations compared to full matrix inversion because of its zero entries. The algorithm for triangular matrix inversion is shown in Figure 2 and described below.

Upper triangular matrix inversion is performed column by column. Calculating the diagonal entries of the $R^l$ matrix consists of simply dividing 1 by the diagonal entry of the $R$ matrix (3) and the rest of the column entries introduce multiplication and addition iteratively (1) which is then divided by the diagonal $R$ matrix entry (2).

$$R^{-1} j = 0$$

for $j = 1 : n$
for $i = 1 : j-1$
for $k = 1 : j-1$
1. $R^{-1} ij = R^{-1} ij + R^{-1} ik R_{kj}$
for $k = 1 : j-1$
2. $R^{-1} kj = -R^{-1} kj / R_{jj}$
3. $R^{-1} jj = 1 / R_{jj}$

Fig. 2. Matrix Inversion of upper triangular matrices.

B. QR Decomposition Based Matrix Inversion

QR decomposition is an elementary operation, which decomposes a matrix into an orthogonal and a triangular matrix. QR decomposition of a matrix $A$ is shown as $A = Q \times R$, where $Q$ is an orthogonal matrix, $Q^T \times Q = I$, $Q^T = Q^{-T}$, and $R$ is an upper triangular matrix (Figure 3(b)). The solution for the inversion of a matrix, $A^{-1}$, using QR decomposition is shown as follows:

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

This solution consists of three different parts: QR decomposition, matrix inversion for the upper triangular matrix and matrix multiplication (Figure 3(c)). QR decomposition is the dominant calculation where the next two parts are relatively simple due to the upper triangular structure of $R$ (as described in section A above).

There are three different QR decomposition methods: Gram-Schmidt orthonormalization (Classical or Modified), Givens Rotations (GR) and Householder reflections. Applying slight modifications to the Classical Gram-Schmidt (CGS) algorithm gives the Modified Gram-Schmidt (MGS) algorithm [13].

QRD-MGS is numerically more accurate and stable than QRD-CGS and it is numerically equivalent to the Givens Rotations solution [14][15][16] (the solution that has been the focus of previously published hardware implementations because of its stability and accuracy). Also, if the input matrix, $A$, is well-conditioned and non-singular, the resulting matrices, $Q$ and $R$, satisfy their required matrix characteristics and QRD-MGS is accurate to floating-point machine precision [16]. We therefore present the QRD-MGS algorithm in Figure 3(a) and describe it below.

$$QRD-MGS(A)$$
1 for $i = 1 : n$
2 $X_i = A_i$
3 for $i = 1 : n$
4 $R_i = ||X_i||$
5 $Q_i = X_i / R_i$
6 for $j = i+1 : n$
7 $R_{ij} = <Q_i X_j>$
8 $X_j = X_j - R_{ij} Q_i$

Fig. 3. QR decomposition modified Gram Schmidt (QR-MGS) algorithm is presented in (a). The resulting matrices of the decomposition are shown in (b). The solution steps of the matrix inversion are presented in (c).

$A, Q, R$ and $X$ are the input, orthogonal, upper triangular and intermediate matrices, respectively. The intermediate matrix is the updated input matrix throughout the solution steps. Matrices with only one index as $A_i$ or $X_j$ represent the columns of the matrix and matrices with two indices like $R_{ij}$ represent the entry at the intersection of $ith$ row with $jth$ column of the matrix where $i \leq j \leq n$.

In Figure 3(a) we show that we start every decomposition by transferring the input, $4 \times 4$, matrix columns, $A_i$, into the memory elements (2). Diagonal entries of the $R$ matrix are the Euclidean norm of the intermediate matrix columns which is shown as (4). The $Q$ matrix columns are calculated by the division of the intermediate matrix columns by the Euclidean norm of the intermediate matrix column, which is the diagonal element of $R$ (5). Non-diagonal entries of the $R$ matrix are computed by projecting the $Q$ matrix columns onto the intermediate matrix columns one by one (7) such that after the solution of $Q_i$, it is projected onto $X_j$ and $X_k$ to compute $R_{23}$ and $R_{24}$. Lastly, the intermediate matrix columns are updated by (8).

C. LU Decomposition Based Matrix Inversion

If $A$ is a square matrix and its leading principal submatrices are all nonsingular, matrix $A$ can be decomposed into unique lower triangular and upper triangular matrices. LU decomposition of a matrix $A$ is shown as $A = L \times U$, where $L$ and $U$ are the lower and upper triangular matrices respectively.
The solution for the inversion of a matrix, \( A^{-1} \), using LU decomposition is shown as follows:

\[
A^{-1} = U^{-1} \times L^{-1}
\]  
\[\text{(6)}\]

This solution consists of four different parts: LU decomposition of the given matrix, matrix inversion for the lower triangular matrix, matrix inversion of the upper triangular matrix and matrix multiplication (Figure 5(c)). Cholesky decomposition is the dominant calculation where the next three parts are relatively simple due to the triangular structure of the matrices.

\[
\begin{align*}
\text{Cholesky(A)} & \\
1 & \text{for } k = 1 : n \\
2 & G_{kk} = \sqrt{A_{kk}} \\
3 & \text{for } i = k+1 : n \\
4 & G_{ik} = A_{ik} / A_{kk} \\
5 & \text{for } j = k+1 : n \\
6 & A_{ij} = A_{ij} - G_{ik}G_{jk} \\
\end{align*}
\]

\[
G = \begin{bmatrix}
G_{11} & 0 & 0 & 0 \\
G_{21} & G_{22} & 0 & 0 \\
G_{31} & G_{32} & G_{33} & 0 \\
G_{41} & G_{42} & G_{43} & G_{44}
\end{bmatrix}
\]

\[
(GT)^{-1}
\]

\[
\begin{align*}
\text{(a) Cholesky Algorithm} & \\
\text{(b) Resulting Matrices} & \\
\text{(c) Solution Steps} & \\
\end{align*}
\]

**Fig. 5.** Cholesky decomposition algorithm is presented in (a). The resulting matrices of the decomposition are shown in (b). The solution steps of the matrix inversion are presented in (c).

Figure 5(a) shows the Cholesky decomposition algorithm. We start decomposition by transferring the input matrix, \( A \), into the memory elements. The diagonal entries of lower triangular matrix, \( G \), are the square root of the diagonal entries of the given matrix (2). We calculate the entries below the diagonal entries by dividing the corresponding element of the given matrix by the belonging column diagonal element (4). The algorithm works column by column and after the computation of the first column of the diagonal matrix with the given matrix entries, the elements in the next columns are updated (7). For example after the computation of \( G_{11} \) by (2), \( G_{21}, G_{31}, G_{41} \) by (4), second column: \( A_{22}, A_{32}, A_{42} \), third column: \( A_{33}, A_{43} \), and fourth column: \( A_{44} \) are updated by (7).

### D. Cholesky Decomposition Based Matrix Inversion

Cholesky decomposition is another elementary operation, which decomposes a symmetric positive definite matrix into a unique lower triangular matrix with positive diagonal entries. Cholesky decomposition of a matrix \( A \) is shown as \( A = G \times G^T \), where \( G \) is a unique lower triangular matrix, Cholesky triangle, and \( G^T \) is the transpose of this lower triangular matrix (Figure 5(b)). The solution for the inversion of a matrix, \( A^{-1} \), using Cholesky decomposition is shown as follows:

\[
A^{-1} = (G^T)^{-1} \times G^{-1}
\]  
\[\text{(7)}\]

This solution consists of four different parts: Cholesky decomposition, matrix inversion for the transpose of the lower triangular matrix, inversion of the lower triangular matrix and matrix multiplication (Figure 5(c)). Cholesky decomposition is the dominant calculation where the next three parts are relatively simple due to the triangular structure of the matrices.
consuming processes. Therefore, a high level tool for design space exploration and fast prototyping is essential and required.

GUSTO, “General architecture design Utility and Synthesis Tool for Optimization,” is such a high level design tool, written in Matlab, that is the first of its kind to provide design space exploration across different matrix inversion architectures. GUSTO allows the user to select the matrix inversion method (QR, LU or Cholesky decompositions), the matrix dimension, the type and number of arithmetic resources, the data representation (the integer and fractional bit width), and two modes of operation (Mode 1 or Mode 2) as shown in Figure 6.

**Fig. 6. Different modes of GUSTO.**

GUSTO has two different modes of operation. Mode 1 provides a general purpose architecture while Mode 2 provides an application specific architecture. The general purpose architecture is used for area and timing analysis of a general non-optimized solution, and the advantage of this architecture is that it is capable of solving any of the decomposition methods with a selection input. Unfortunately, Mode 1’s general purpose architectures generally do not lead to high-performance results. When the user knows the environmental requirements and matrix characteristics which will be encountered, choosing a specific method and creating an application specific architecture by optimizing/customizing these architectures to improve their area results is another essential step to enhance design quality.

In Mode 2, GUSTO creates a scheduled, static, application specific architecture while ensuring the correctness of the solution is maintained. We divided these optimizations into two sections: static architecture generation and trimming for optimization.

**Static architecture generation:** Mode 1 of GUSTO generates a general purpose architecture and its datapath by using resource constrained list scheduling after the required inputs are given. Simulating this architecture in Mode 2 helps us to reveal the assignments done to the arithmetic units and the memory elements during the scheduling process. Gathering this information and using it to cancel the scheduling process and dynamic memory assignments results in a static architecture with significant area and timing savings.

**Trimming for optimization:** GUSTO performs trimming/removing the unused resources from the general purpose architecture while ensuring that correctness of the solution is maintained. GUSTO simulates the architecture to define the usage of arithmetic units, multiplexers, register entries and input/output ports and trims away the unused components with their interconnects. A trimming example is shown in Figure 7. Suppose there are 2 arithmetic units with 2 inputs/1 output each and one memory with 1 input/2 output (a). Input / output port relationships between arithmetic unit A and the other units are shown in a block diagram in (b). Although Out_A, Out_B, Out_mem1, and Out_mem2 are all inputs to In_A1 and In_A2, not all the inputs may be used during computation. We can represent whether an input/output port is used or not during simulation in a matrix such as the one shown in (c). As the simulation runs, the matrix is filled with 1s and 0s representing the used and unused ports respectively. GUSTO uses these matrices to remove the unused resources (d). In this example, two inputs, Out_A, Out_mem1 to In_A1 and another two inputs, Out_B, Out_mem2 to In_A2 are removed.

**Fig. 7. Flow of GUSTO’s trimming feature.**

IV. RESULTS

In this section, we present different design space exploration examples using different inputs of GUSTO and compare our results with previously published FPGA implementations. Design space exploration can be divided into two parts, inflection point analysis and architectural design alternatives analysis.
Inflection Point Analysis: We partitioned inflection point analysis into another two parts. First, we present the results for decomposition methods which help us to quantify their effects on the computation of matrix inversion and then we present matrix inversion results. The total number of operations used in different decomposition methods is shown in Figure 8 in log domain. It is important to notice that there is an inflection point between LU and Cholesky decompositions at 4 × 4 matrices with a significant difference from QR decomposition. Furthermore, this inflection point is shifted to 5 × 5 matrices for matrix inversion implementations where LU and Cholesky have more significant differences in terms of total number of operations; besides the difference between QR and the other decomposition methods increases. To further investigate, we present different execution results, serial and parallel, for different bit widths and matrix dimensions to answer: at what matrix size does an inflection point occur and how does varying bit width and degree of parallelism change the inflection point? The comparisons for sequential and parallel executions of QR, LU and Cholesky methods are shown in Figure 9 (a, b, c and d) with different bit widths: 16, 32 and 64. Square, spade and triangle represent QR, LU and Cholesky methods respectively. solid, dashed and smaller dashed lines represent 64, 32 and 16 bits of bit widths respectively. The balloons denote the inflection points between these methods for the different bit widths.

![Graphs showing inflection points for different bit widths and matrix sizes for QR, LU, and Cholesky methods.](image)

**Fig. 8.** Total number of operations in log domain for decomposition based matrix inversion (light) and decompositions only (dark). Note that the dark bars overlap the light bars.

The sequential execution results of decomposition methods (a) show that the QR decomposition method executes a significantly higher number of clock cycles than the other methods. The 16 bit QR decomposition implementation...
requires the same number of clock cycles with the 64 bit LU decomposition implementation. Cholesky decomposition takes more clock cycles than LU decomposition where this difference becomes smaller for smaller number of bit widths. The sequential execution results of decomposition methods based matrix inversion (c) show that QR takes more clock cycles than Cholesky and LU again where Cholesky takes more cycles than LU. As the bit widths get smaller, the difference between QR and the other methods does not change, however the difference between Cholesky and LU decomposition becomes smaller.

The parallel execution results of decomposition methods (b) show that QR decomposition and Cholesky decomposition get closer to each other where LU decomposition performs better than the others. It is important to see that the 64 bit implementation of LU decomposition performs almost the same as the 32 bit Cholesky implementation and also the 32 bit LU implementation performs almost the same as the 16 bit implementation of Cholesky. The parallel execution results of decomposition methods based matrix inversion (d) show that QR decomposition based matrix inversion architectures have the highest number of clock cycles for all bit widths where Cholesky and LU decomposition based matrix inversion architectures have a similar number of clock cycles for small bit widths. However, LU decomposition uses increasingly fewer clock cycles than Cholesky decomposition with increasing bit widths and matrix dimensions. LU decomposition with 32 bits performs almost the same as QR decomposition with 16 bits. Also, the 64 bits LU decomposition performs almost the same as the 32 bits QR decomposition in terms of total number of clock cycles.

Architectural Design Alternatives: These analyses are shown for all decomposition based matrix inversion for different bit widths and matrix sizes. We present area results in terms of slices and performance results in terms of throughput. Throughput is calculated by dividing the maximum clock frequency (MHz) by the number of clock cycles to perform matrix inversion. Both mode 1 (non-optimized) and mode 2 (optimized) results are shown for QR decomposition based matrix inversion in Figure 10 (a) to show the improvement in the results with the optimization feature. It is shown that area and throughput increase up to the optimal number of resources as the number of resources increase. However, adding more than the optimal number of resources decreases throughput while still increasing area. Mode 2 of GUSTO finds the optimal number of resources which maximizes the throughput while minimizing area where the application specific architecture provides an average of 59% decrease in area and 3X increase in throughput over Mode 1’s general purpose (non optimized) design.

Bit width of the data is another important input for the matrix inversion. The precision of the results is directly dependent on the number of bits used. The usage of a high number of bits results in high precision at a cost of higher area and lower throughput. We present 3 different bit widths, 19, 26 and 32 bits in (b) for these three different decomposition based matrix inversion architectures. Usage of LU decomposition for matrix inversion results in smallest area and highest throughput compared to the other methods. Cholesky decomposition offers higher throughput at a cost of larger area compared to QR decomposition.

We also present three different matrix dimension, $4 \times 4$, $6 \times 6$ and $8 \times 8$, implementation results in (c) showing how the area and performance results scale with matrix dimension. We again observe that LU decomposition based matrix inversion architectures offer better area and throughput results compared to other methods for all matrix sizes.

Comparison: A comparison between our results and previously published implementations for a $4 \times 4$ matrix is presented in Table 1. For ease of comparison we present all of our implementations with bit width 20 as this is the largest bit width value used in the related works. Though it is difficult to make direct comparisons between our designs and those of the related works (because we used fixed point arithmetic instead of floating point arithmetic and fully used FPGA resources (like DSP48s) instead of LUTs), we observe that our results are comparable. The main advantages of our implementation are that it provides the designer the ability to study the tradeoffs between architectures with different design parameters and provides a means to find an optimal design.
V. CONCLUSION

This paper presents different decomposition based matrix inversion architectures using a matrix inversion core generator tool, GUSTO, that is developed to enable easy design space exploration for various matrix inversion architectures which targets reconfigurable hardware designs. GUSTO provides different parameterization options including matrix dimensions, bit width and resource allocations which enables us to study area and performance tradeoffs over a large number of different architectures. In this paper, we especially concentrate on QR, Cholesky and LU decomposition methods for matrix inversion, to observe the advantages and disadvantages of these methods in response to varying parameters. GUSTO is the only tool that allows design space exploration across different matrix inversion architectures. Its ability to provide design space exploration, which leads to an optimized architecture, makes GUSTO an extremely useful tool for applications requiring matrix inversion.

REFERENCES


