An FPGA Design Space Exploration Tool for Matrix Inversion Architectures

Ali Irturk[†], Bridget Benson[†], Shahnam Mirzaei[‡], Ryan Kastner[†]

Abstract— **Matrix inversion is a common function found in many algorithms used in wireless communication systems. As FPGAs become an increasingly attractive platform for wireless communication, it is important to understand the tradeoffs in designing a matrix inversion core on an FPGA. This paper describes a matrix inversion core generator tool, GUSTO, that we developed to ease the design space exploration across different matrix inversion architectures. GUSTO is the first tool of its kind to provide automatic generation of a variety of general purpose matrix inversion architectures with different parameterization options. GUSTO also provides an optimized application specific architecture with an average of 59% area decrease and 3X throughput increase over its general purpose architecture. The optimized architectures generated by GUSTO provide 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

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) [1]. Matrix inversion is an essential computation for various algorithms which are employed in MIMO-OFDM systems, e.g. equalization algorithms to remove the effect of the channel on the signal [2][3][4], minimum mean square error algorithms for pre-coding in spatial multiplexing [5] and detection-estimation algorithms in space-time coding [6].

FPGAs are an increasingly common platform for wireless communication [7-9]. FPGAs are a perfect platform for computationally intensive arithmetic calculations like matrix inversion as they provide powerful computational architectural features: vast amounts of programmable logic elements, embedded multipliers, shift register LUTs (SRLs), Block RAMs (BRAMs), DSP blocks and Digital Clock Managers (DCMs). If used properly, these features enhance the performance and throughput significantly. However, the highly programmable nature of the FPGA can also be a curse. An FPGA offers vast amounts of customization which requires the designer to make a huge number of system, architectural and logic design choices. This includes decisions on resource allocation, bit widths of the data, number of functional units and the organization of controllers and interconnects. These choices can overwhelm the designer unless she is provided with design space exploration tools to help her prune the design space.

For more efficient design space exploration and development, we designed an easy-to-use tool, GUSTO (**G**eneral architecture design **U**tility and **S**ynthesis **T**ool for **O**ptimization), which allows us to select various parameters such as different matrix dimensions, integer and fractional bits of the data, resource allocation, modes for general purpose or application specific architectures, etc. GUSTO provides two modes of operation. In mode 1, it creates a general purpose architecture and its datapath for given inputs. In mode 2, it optimizes/customizes the general architecture to improve its area results and design quality. Mode 2 performs this improvement by trimming/removing the unused resources from the general purpose architecture and creating a scheduled, static, application specific architecture while ensuring that correctness of the solution is maintained. GUSTO also creates required HDL files which are ready to simulate, synthesize and map.

The main contributions of this paper are:

- an easy-to-use matrix inversion core generator for design space exploration with reconfigurable matrix dimensions, bit widths, resource allocation, modes and methods which can generate and/or optimize the design;
- a study of the area, timing and throughput tradeoffs using different design space decisions;
- the determination of an inflection point, in terms of matrix dimensions and bit widths, between QR decomposition method and analytic method.

The rest of this paper is organized as follows: Section II introduces MIMO systems, matrix inversion and two methods to solve matrix inversion: QR decomposition method and analytic method. Section III explains the architectural design of the core generator. Section IV introduces FPGA resources, discusses design decisions and challenges, presents implementation results in terms of area and performance and compares our results with other published FPGA implementations. We conclude in Section V.

II. MIMO Systems, Matrix Inversion and Its Methods

The received signal of a MIMO system with N transmit and M receive antennas is $Y = HX + w$, where X, Y and w are the complex transmitted signal, complex received signal and complex white Gaussian noise respectively. The wireless channel is described as an $M \times N$ deterministic matrix H. The received signal equation can be replaced by its real valued equivalent for computational convenience. 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 [10]. 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 in MIMO-OFDM systems.

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

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

where I is the identity matrix. Below we describe two known methods to perform matrix inversion: QR decomposition method and analytic method. QR decomposition method is generally viewed as the preferred method because it scales well for large matrix dimensions while the complexity of the analytic method increases dramatically as the matrix dimensions grow. However, for small matrices, the analytic method, which can exploit a significant amount of parallelism, outperforms the QR decomposition method.

A. Matrix Inversion using QR Decomposition

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, $Q^T \times Q = Q \times Q^T = I$, $Q^{-1} = Q^T$, and R is an upper triangular matrix. The solution for the inversion of matrix A, A^{-1}, using QR decomposition is shown as follows:

$$A^{-1} = R^{-1} \times Q^T \text{ [14]} \tag{2}$$

This solution consists of three different parts, QR decomposition, matrix inversion for the upper triangular matrix and matrix multiplication. QR decomposition is the most computationally intensive calculation where the next two parts are relatively simple due to the upper triangular structure of R.

There are three different QR decomposition methods: Gram-Schmidt orthogonormalization (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 [14]. QRD-MGS is numerically more accurate and stable than QRD-CGS and it is numerically equivalent to the Givens Rotations solution [11][12][13] (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 [13]. We therefore present the QRD-MGS algorithm in Figure 1 and describe it below.

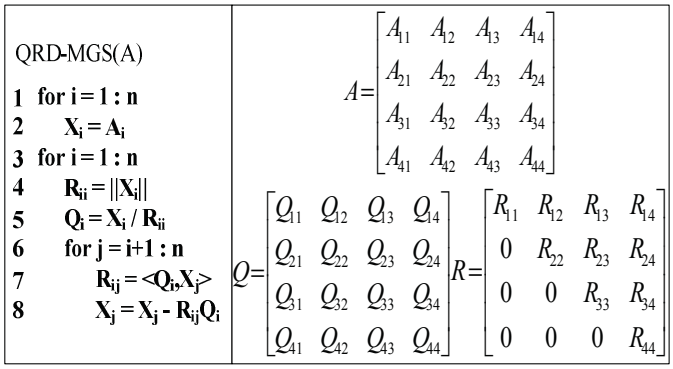

Fig. 1. QRD-MGS Algorithm and sample A, Q, R matrices (4 × 4).

A, Q, R and X are the input, orthogonal, upper triangular and intermediate matrices, respectively. The intermediate matrix, X, 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 $1 \le i,j \le n$.

We start every decomposition by transferring the input matrix columns, A_i, into the memory elements (2). Diagonal entries of the R matrix are the Euclidean norm of the X matrix columns which is shown as (4). The Q matrix columns are calculated by the division of the X matrix columns by the Euclidean norm of the X 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 X matrix columns one by one (7) such that after the solution of Q_2, it is projected onto X_3 and X_4 to compute R_{23} and R_{24}. Lastly, X matrix columns are updated by (8).

B. Matrix Inversion using Analytic Method

Another method for inverting an input matrix A, is the analytic method which uses the adjoint matrix, $Adj(A)$, and determinant, $det\ A$. This calculation is given by

$$A^{-1} = \frac{1}{\det A} \times Adj(A) \tag{3}$$

The adjoint matrix is the transpose of the cofactor matrix where the cofactor matrix is formed by using determinants of the input matrix with signs depending on its position. It is formed in three stages. First, we find the transpose of the input matrix *A* by interchanging the rows with the columns. Next, the matrix of minors is formed by covering up the elements in its row and column and finding the determinant of the remaining matrix. Finally, the cofactor of any element is found by placing a sign in front of the matrix of minors by calculating $(-1)^{(i+j)}$. These calculations are shown in Figure 2 (a) for the first entry in the cofactor matrix, C_{11}.

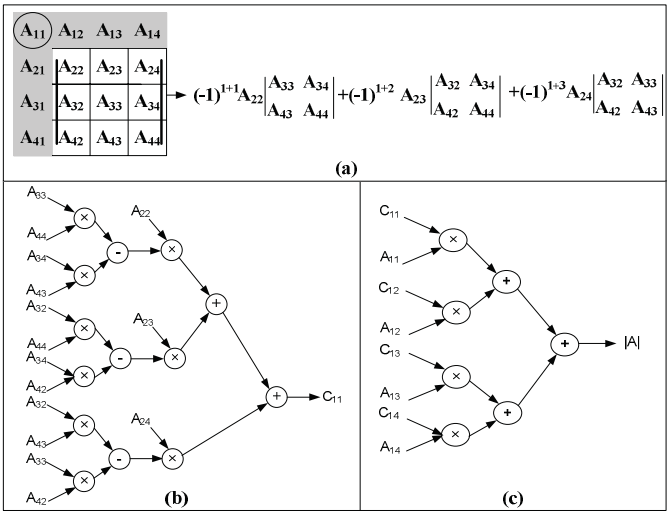

Fig. 2. Matrix Inversion with analytic approach. The first element of cofactor matrix, C_{11}, and determinant calculation for a 4 × 4 matrix is shown in (a), (b) and (c) respectively.

The calculation of the first entry in the cofactor matrix, C_{11} is also presented in (b). This stage is repeated 16 times for a 4 × 4 matrix to form the cofactor matrix. The adjoint matrix is the transpose of the cofactor matrix and formed using register renaming. After the calculation of the adjoint matrix, the determinant is calculated using a row or a column which is shown in (c). The last stage is the division between the adjoint matrix and the determinant which gives the inverted matrix.

For the analytic method, we present three different designs, *Implementation A, B,* and *C*, with varying levels of parallelism to form cofactor matrices. *Implementation A* uses one core, *Implementation B* uses two cores and *Implementation C* uses 4 cores for the cofactor formulation shown in (b). In the next section, we present our core generator GUSTO which is an infrastructure for fast prototyping of QR decomposition method and analytic method.

III. Matrix Inversion Core Generator Tool

There are several different architectural design alternatives for these solution methods of matrix inversion. Thus, it is important to study tradeoffs between these alternatives and find the most suitable solution for desired results such as most time efficient or most area efficient design. Performing design space exploration is a time consuming process where there is an increasing demand for higher productivity. High level design tools offer great convenience by easing this burden and giving us the opportunity to test different alternatives in a reasonable amount of time. Therefore, designing a high level tool for fast prototyping is essential.

GUSTO, "**G**eneral architecture design **U**tility and **S**ynthesis **T**ool for **O**ptimization," 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. As shown in Figure 3, GUSTO allows the user to select the matrix inversion method (QR decomposition or analytic), the matrix dimension, the type and number of arithmetic resources, the data representation (the integer and fractional bit width), and the mode of operation (Mode 1 or Mode 2).

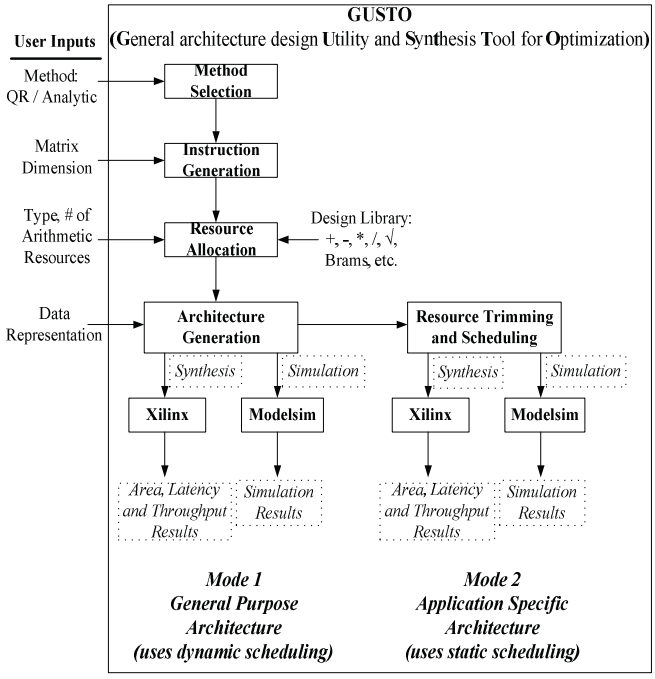

Fig. 3. Different modes of GUSTO.

Mode 1 of GUSTO generates a general purpose architecture and its datapath by using resource constrained list scheduling after the required inputs are given. The general purpose architecture is used for area and timing analysis for a general non-optimized solution. The advantage of generating a general purpose architecture is that it can be used to explore other algorithms, (other than analytic and QR) so long as these algorithms require the same resource library. However Mode

1's general purpose architectures generally do not lead to high-performance results. Therefore optimizing/customizing these architectures to improve their area results is another essential step to enhance design quality.

In Mode 2, GUSTO performs this improvement by trimming/removing the unused resources from the general purpose architecture and creating a scheduled, static, application specific 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 4. Suppose there are 3 arithmetic units and one memory with 2 inputs and 1 output each (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, Mem, and Out_C are all inputs to In_A1 and In_A2, not all the inputs may be used during simulation. 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).

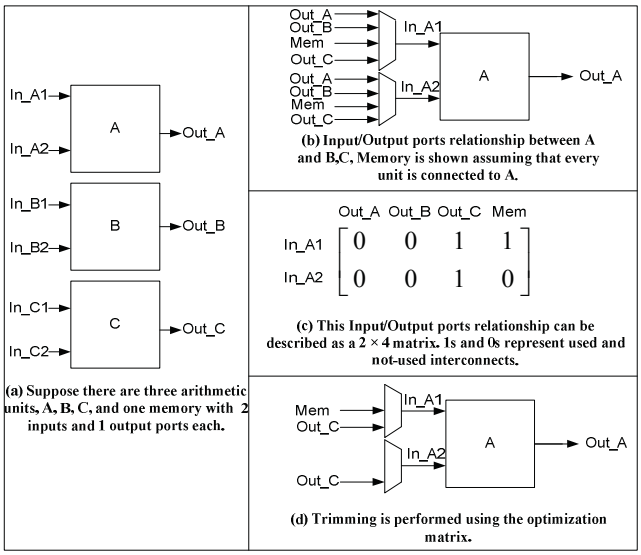

Fig. 4. Flow of GUSTO's trimming feature.

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 and architectural design alternatives analysis.

Inflection Point Analysis: The total number of operations used in both methods is shown in Figure 5 (a) in log domain. It is important to notice that the total number of operations increases by an order of magnitude for each increase in matrix dimension for the analytic method making the analytic solution unreasonable for large matrix dimensions. Since the analytic approach does not scale well, there will be an inflection point where the decomposition approach will provide better results. At what matrix size does this 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 and analytic methods are shown in (b and c) with different bit widths: 16, 32 and 64. We used implementation A for the parallel implementation of analytic method. Solid and dashed lines represent QR decomposition method and analytic method results respectively. The balloons denote the inflection points between the two methods for the different bit widths.

The sequential execution results (b) show that the analytic method offers a practical solution for matrix dimensions $\leq 4 \times 4$. It also gives the same performance as the QR decomposition method for 5×5 matrices using 64 bits. The analytic method result increases dramatically for 6×6 matrices (not shown) where it needs 12,251 clock cycles (for 16 bits) as opposed to 1,880 clock cycles for QR decomposition suggesting the analytic method is unsuitable for matrix dimensions $\geq 6 \times 6$.

The parallel execution results are shown in (c). Analytic method offers a practical solution for matrix dimensions $\leq 4 \times 4$ and it is preferred for 5×5 matrix dimension for 32 and 64 bits. The increase in the clock cycle is again dramatic for matrix dimensions $\geq 6 \times 6$ for the analytic method demanding to use the QR decomposition method for these larger matrix dimensions.

Architectural Design Alternatives: These analyses are shown for QR decomposition based matrix inversion for 4×4 matrices. 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. We present both mode 1 (non-optimized) and mode 2 (optimized) results in (d) to show the improvement in our results with the optimization feature, and present only mode 2 results in (e) and (f).

We investigate different resource allocations using both modes of GUSTO and present the results in Figure 5 (d). As expected from mode 1, (d) shows an increase in area and throughput as the number of resources increase up to optimal number of resources. Adding more than optimal number of resources decreases throughput while still increasing area. However, mode 2 of GUSTO finds the optimal number of resources which maximizes the throughput while minimizing area which is shown in (d). Mode 2's optimized application specific architecture can therefore provide an average of 59% decrease in area and 3X increase in throughput over Mode 1's general purpose (non optimized) design.

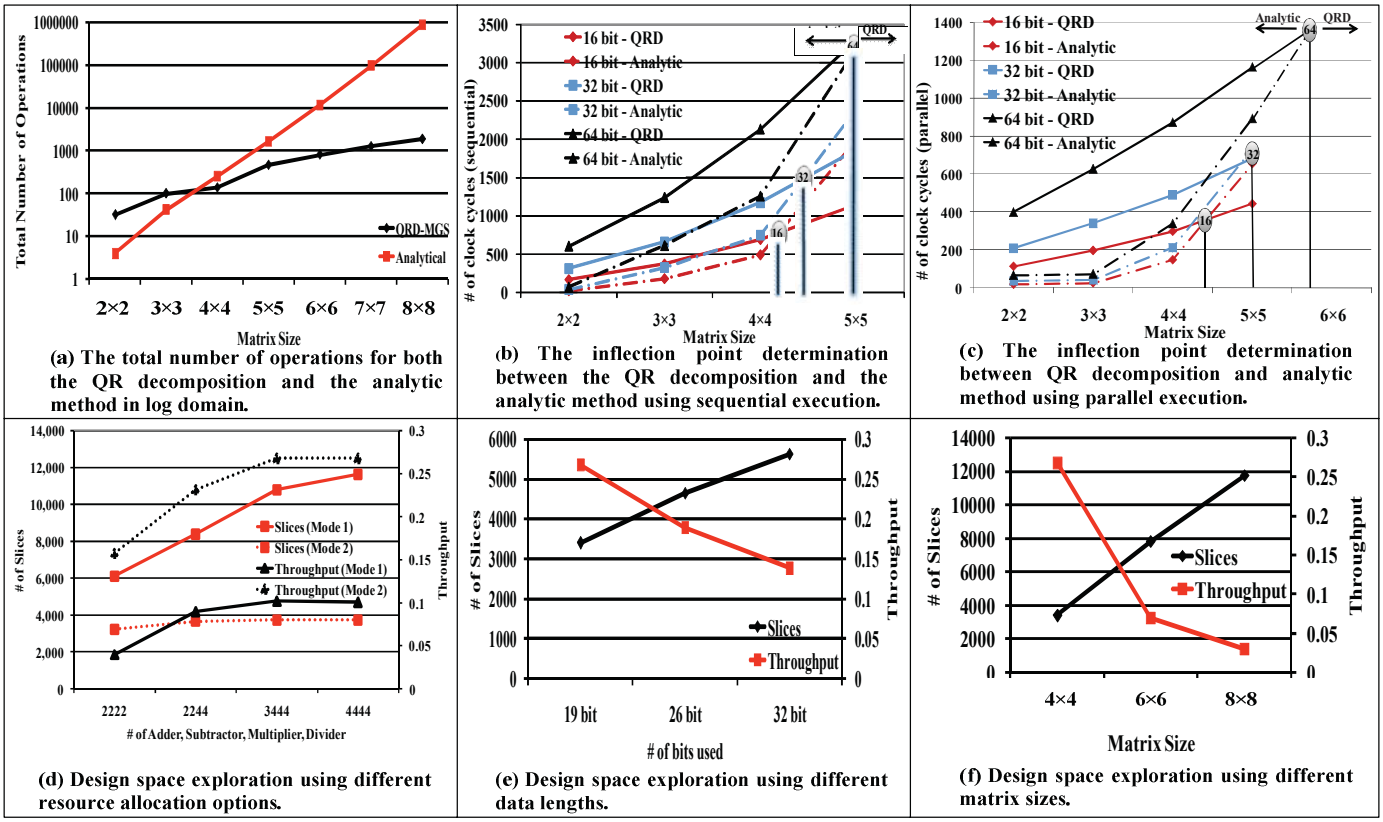

Fig. 5. Different design space exploration examples of our tool. This exploration divided into two parts: inflection point and architectural design alternatives analysis.

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 (e). We also present three different matrix dimensions, 4 × 4, 6 × 6 and 8 × 8, implementation results in (f) showing how the area and performance results scale with matrix dimension.

Comparison: A comparison between our results and previously published implementations for a 4 × 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 provides the designer the ability to study the tradeoffs between architectures with different design parameters and provides a means to find an optimal design.

TABLE I
COMPARISONS BETWEEN OUR RESULTS AND PREVIOUSLY PUBLISHED PAPERS. NR DENOTES NOT REPORTED.

	Ref[15]	*Ref[15]*	***Our ImplA***	***Our ImplB***	***Our ImplC***	*Ref[16]*	*Ref[17]*	***Our***
Method	Analytic	Analytic	**Analytic**	**Analytic**	**Analytic**	QR	QR	**QR**
Bit width	16	20	**20**	**20**	**20**	12	20	**20**
Data type	floating	floating	**fixed**	**fixed**	**fixed**	fixed	floating	**fixed**
Device type	Virtex 4	Virtex 4	**Virtex 4**	**Virtex 4**	**Virtex 4**	Virtex 2	Virtex 4	**Virtex 4**
Slices	1561	2094	**702**	**1400**	**2808**	4400	9117	**3584**
DSP48s	0	0	**4**	**8**	**16**	NR	22	**12**
BRAMs	NR	NR	**0**	**0**	**0**	NR	NR	**1**
Throughput *($10^6 \times s^{-1}$)*	1.04	0.83	**0.38**	**0.72**	**1.3**	0.28	0.12	**0.26**

V. CONCLUSION

This paper describes a matrix inversion core generator tool, GUSTO, that we 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. We especially concentrate on QR decomposition method and analytic method for matrix inversion, to observe the advantages and disadvantages of both 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 (i.e. MIMO systems).

REFERENCES

[1] L. Hanzo, T. Keller, "OFDM and MC-CDMA: A Primer," *Wiley-IEEE Press*, 2006.

[2] L. Zhou, L. Qiu, J. Zhu, "A novel adaptive equalization algorithm for MIMO communication system", *Vehicular Technology Conference*, 2005, Volume 4, 25-28 Sept., 2005 Page(s):2408 – 2412.

[3] T. Abe, S. Tomisato,T. Matsumoto, "A MIMO turbo equalizer for frequency-selective channels with unknown interference"**,** *IEEE Transactions on Vehicular Technology*, Volume 52, Issue 3, May 2003 Page(s):476 – 482.

[4] T. Abe and T. Matsumoto, "Space-time turbo equalization in frequency selective MIMO channels," *IEEE Trans. Vehicular Technology.*, pp. 469–475, May 2003.

[5] K. Kusume, M. Joham, W. Utschick, G. Bauch, **"**Efficient Tomlinson-Harashima precoding for spatial multiplexing on flat MIMO channel," *IEEE International Conference on Communications*, Volume 3, 16-20 May 2005 Page(s):2021 - 2025 Vol. 3.

[6] C. Hangjun, D. Xinmin, A. Haimovich, "Layered turbo space-time coded MIMO-OFDM systems for time varying channels", *Global Telecommunications Conference*, 2003. IEEE Volume 4, 1-5 Dec. 2003 Page(s):1831 - 1836 vol.4.

[7] Y. Meng, A.P. Brown, R. A. Iltis, T. Sherwood, H. Lee, R. Kastner, "MP core: algorithm and design techniques for efficient channel estimation in wireless applications," *Design Automation Conference, 2005. Proceedings. 42nd* , vol., no., pp. 297-302, 13-17 June 2005.

[8] R. A. Iltis, S. Mirzaei, R. Kastner, R. E. Cagley,B. T. Weals, "Carrier Offset and Channel Estimation for Cooperative MIMO Sensor Networks," *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE* , vol., no., pp.1-5, Nov. 2006.

[9] R. E. Cagley, B. T. Weals, S. A. McNally, R. A. Iltis, S. Mirzaei, R. Kastner, "Implementation of the Alamouti OSTBC to a Distributed Set of Single-Antenna Wireless Nodes," *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE* , vol., no., pp.577-581, 11-15 March 2007.

[10] T. Kailath, H. Vikalo, B. Hassibi, "MIMO Receive Algorithms." Space-Time Wireless Systems: From Array Processing to MIMO Communications. *Cambridge University Press*, 2005.

[11] A. Bjőrck, 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.

[12] A. Bjőrck, "Numerics of Gram-Schmidt orthogonalization," *Linear Algebra and Its Applications*, vol. 198, pp. 297-316, 1994.

[13] C. K. Singh, S.H. Prasad, P.T. Balsara, "VLSI Architecture for Matrix Inversion using Modified Gram-Schmidt based QR Decomposition", *20th International Conference on VLSI Design*. (2007) 836 – 841.

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

[15] J. Eilert, D. Wu, D. Liu, "Efficient Complex Matrix Inversion for MIMO Software Defined Radio", *IEEE International Symposium on Circuits and Systems*. (2007) 2610 – 2613.

[16] F. Edman, V. Öwall, "A Scalable Pipelined Complex Valued Matrix Inversion Architecture", *IEEE International Symposium on Circuits and Systems*. (2005) 4489 – 4492.

[17] M. Karkooti, J.R. Cavallaro, C. Dick, "FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm", *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers* (2005) 1625 – 162.