A Reconfigurable Architecture for QR Decomposition Using A Hybrid Approach

Xinying Wang, Phillip Jones and Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University, Ames, Iowa, USA
Email: {xinying, phjones, zambreno}@iastate.edu

***Abstract*—QR decomposition has been widely used in many signal processing applications to solve linear inverse problems. However, QR decomposition is considered a computationally expensive process, and its sequential implementations fail to meet the requirements of many time-sensitive applications. The Householder transformation and the Givens rotation are the most popular techniques to conduct QR decomposition. Each of these approaches have their own strengths and weakness. The Householder transformation lends itself to efficient sequential implementation, however its inherent data dependencies complicate parallelization. On the other hand, the structure of Givens rotation provides many opportunities for concurrency, but is typically limited by the availability of computing resources. We propose a deeply pipelined reconfigurable architecture that can be dynamically configured to perform either approach in a manner that takes advantage of the strengths of each. At runtime, the input matrix is first partitioned into numerous sub-matrices. Our architecture then performs parallel Householder transformations on the sub-matrices in the same column block, which is followed by parallel Givens rotations to annihilate the remaining unneeded individual off-diagonals. Analysis of our design indicates the potential to achieve a performance of 10.5 GFLOPS with speedups of up to $1.46\times$, $1.15\times$ and $13.75\times$ compared to the MKL implementation, a recent FPGA design and a Matlab solution, respectively.**

***Index Terms*—Architecture, FPGA, QR decomposition, Householder transformation, Givens rotation.**

I. Introduction

QR decomposition has been widely used in many signal processing applications such as MIMO systems [1], beamforming [2] and image recovery [3] to calculate the inverse of matrices or solve linear systems. However, its inherent computational complexity makes it unlikely to satisfy the requirements of many time-sensitive designs, especially when the system operates on a large-scale dataset. QR decomposition is generally considered as an $O(n^3)$ operation, and previous research has shown that more than 10 minutes could be taken to perform QR decomposition-based robust Principal Component Analysis on a $110,592 \times 100$ matrix, which is far beyond the requirements of potential real-time applications such as video surveillance or traffic monitoring [4].

The Gram-Schmidt process, Householder transformation and Givens rotation are known as the most popular algorithms for QR decomposition [5], among which, the Householder transformation and the Givens rotation are considered numerical stable algorithms, while the Gram-Schmidt process provides an opportunity to perform successive orthogonalizations. Parallel designs have been previously investigated to accelerate QR decomposition on traditional multi-core systems [6], [7], GPUs [8] and reconfigurable computing platforms [9]–[12].

The Householder transformation is efficient in its vectorized operations. However, parallelization of the Householder transformation is challenged by the data dependencies among vectors [13]. To help mitigate the issue of data dependency, a tiled QR decomposition (also known as the blocked Householder transformation) was proposed [14], and has been demonstrated to better exploit the parallelism available on multi-core CPUs [15], GPU [8] and FPGAs [10], [11].

The Givens rotation provides better opportunities for highly parallel designs. However, the scalability of Givens rotation-based QR decomposition is typically limited by the $O(n^2)$ processing elements (PEs) needed to fully parallelize those rotations for an $n \times n$ matrix [16]. A two-dimensional systolic array was devised for fast parallel Givens rotations on a single FPGA [9]. However, the scalability was severely restricted due to the large amounts of resources required.

In this paper, we present a hybrid approach that leverages the strengths of both Householder transformation and Givens rotation by applying the most appropriate of the two at each stage of the QR decomposition process. We propose a reconfigurable architecture for QR decomposition, which can be dynamically configured to perform either Householder transformation or Givens rotation, both of which are deeply pipelined. To process large data sets, the input matrix is partitioned into multiple columns of sub-matrices. The sub-matrix columns are processed successively, while the sub-matrices in the same column are applied with parallel independent Householder transformations. Then, the dense sub-matrix column is transformed into numerous upper triangular sub-matrices, on which highly parallel Givens rotations are performed to annihilate the remaining non-zero elements. Our experimental results show our design can achieve 10.5 GFLOPS with speedups of up to $1.46\times$, $1.15\times$ and $13.75\times$ compared to the Intel Math Kernel Library (MKL) implementation on a single CPU core [17], an FPGA-based tiled matrix decomposition [10], and a single threaded Matlab routine, respectively.

II. Theoretical Background

A. QR Decomposition

QR decompostion of an $m \times n$ matrix A has a form given by eq. (1)

978-1-4799-3765-3/14 $31.00 © 2014 IEEE
DOI 10.1109/ISVLSI.2014.92

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

where Q is an $m \times m$ matrix, which is an orthogonal matrix such that $Q^T \cdot Q = I$, and R is an $m \times n$ upper triangular matrix [5].

B. Householder Transformation

The Householder transformation [5] is a linear process that reflects a vector through a plane containing the origin. The transformed vector is orthogonal to and has the same norm as the original vector. To perform the linear reflection of a vector x, the Householder matrix as shown in eq. (2) is employed, in which v is a unit vector orthogonal to the plane.

$$H= I - 2vv^T \tag{2}$$

To perform the transformation so that all the elements in the transformed vector below the first entry are zero, the unit vector v can be constructed as shown in eq. (3) and eq. (4), where e is a unit vector $(1, 0, 0, \cdots, 0)^T$.

$$u= x+\|x\|e \tag{3}$$

$$v= \frac{u}{\|u\|} \tag{4}$$

By applying the Householder matrix, the householder transformation is performed, eq. (5), where c is $\pm\|x\|$.

$$H \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} c \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

C. Givens Rotation

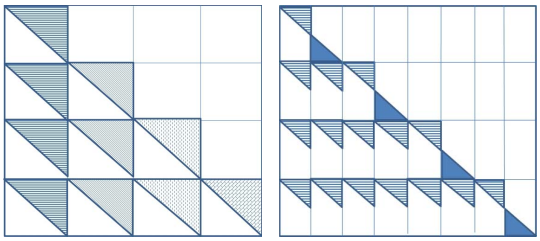
The Givens rotation introduces zeros to matrices through plane rotations. After determining the plane rotation angle (θ) for paired elements, as shown in the eqs. (7,8,9), zero elements can be introduced by conducting rotations in the form of eq. (6) [5]. Since only two elements are operated on in a single rotation, the Givens rotation provides better opportunities to process individual components in parallel.

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix} \tag{6}$$

$$x = \sqrt{a^2 + b^2} \tag{7}$$

$$\cos\theta = \frac{a}{x} \tag{8}$$

$$\sin\theta = \frac{-b}{x} \tag{9}$$



(a) $p \times p$ sub-matrices (b) $p \times q$ sub-matrices

Fig. 1: An example sub-matrix partition of an $m \times m$ matrix for our Hybrid QR decomposition algorithm

III. Related Work

The numerical stability and sequential implementation efficiency of the Householder transformation has lead it to be employed in many standard software packages (e.g. Matlab, LAPACK) [18]. Parallel implementations of the Householder transformation have been investigated on multi-core systems, GPUs and reconfigurable computing platforms [7], [8], [11]. However, the performance improvement is challenged by its inherent data dependencies. Although the tiled matrix [19] was introduced to efficiently partition data sets, near ideal speedups on multi-core platform-based or GPU-based designs are achieved only for matrices with large dimensions due to heavy inter-core communication [7], [8]. In [11], an efficient FPGA-based QR decomposer for tall-skinny matrices was presented. An additional concern with this decomposer is that during the merge stage parallelism decreases as the quantity of intermediate results reduce.

The Givens rotation has been proven to be the most accurate and stable approach for QR decomposition [5], [20]. Compared to the Householder transformation, the Givens rotation provides more opportunities for parallelism, especially when annihilating individual isolated elements. In [9], [16], [21], a 2-dimensional systolic array was employed to demonstrate the parallel implementation of Givens rotation in hardware. However, their scalability was constrained due to the limited resources on a single chip. As demonstrated in [21], 86% of a Virtex-II FPGA's resources were consumed to factorize a 4×4 matrix. Our proposed architecture looks to leverage the benefits of both the sequential efficiency of the Householder transformation and parallelizability of Givens rotation by using a hybrid approach.

IV. Hybrid QR Algorithm

As previously mentioned, the Householder transformation is able to efficiently zero out all the components of a vector below the first entry. However, its inherent data dependency makes parallelization challenging. The Givens rotation provides better opportunities to pursue parallelism and more flexibility for processing individual isolated elements. However, decomposing a large dense matrix by Givens rotation requires a large number of rotation operations. To improve the performance of QR decomposition by combining the advantages of both algorithms, our approach divides the input matrix

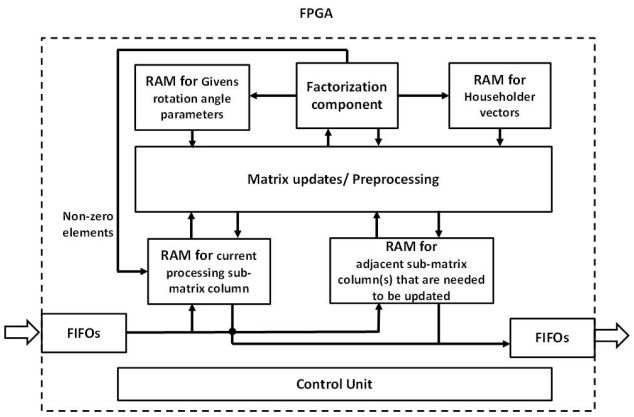


Fig. 2: Block diagram of our Hybrid QR decomposition architecture.

into a number of sub-matrices, on which local Householder transformations are performed in parallel on sub-matrices in the same column block. Then, parallel Givens rotations are employed to annihilate the remaining isolated non-zero lower triangular elements. Compared to [11], which targeted tall-skinny matrices and employed the Householder transformation both to factorize sub-matrices and merge the transformed sub-matrices, Givens rotations are better for conducting parallel processing at the merge stage, especially when floating-point arithmetic is used, whose computations have relatively long latencies. In addition, the Givens rotation can potentially achieve additional acceleration when factorizing partially sparse matrices.

In our hybrid QR decomposition algorithm, the input matrix is divided into $m \times n$ sub-matrices. The sub-matrices from the same columns can be processed in parallel both for factorization and updates, while sub-matrices having the same row indices are processed successively from left to right. The factorization process and update operation can be performed simultaneously if no data dependencies exist between them. The sub-matrices can either be square as shown in Fig. 1a, or rectangular, as shown in Fig. 1b. As demonstrated in Fig. 1, the lower triangular part of the matrix has been transformed into a number of upper triangular sub-matrices (as depicted by the shaded area) by parallel Householder transformations, which then will be annihilated by parallel Givens rotations.

V. Our Architecture for QR Decomposition

A reconfigurable computing platform provides a flexible medium for dynamically configuring our architecture to perform either the Householder transformation or the Givens rotation algorithm. The detailed calculations of the Householder transformation algorithm and the Givens rotation approach are described in Table I, both of which can be primarily summarized as three steps: a) preprocessing, b) factorization, and c) matrix updates. In the preprocessing stage, the Householder transformation employs vector-vector multiplication to compute the squared vector norms, while the Givens rotation performs multiplication-addition operations,

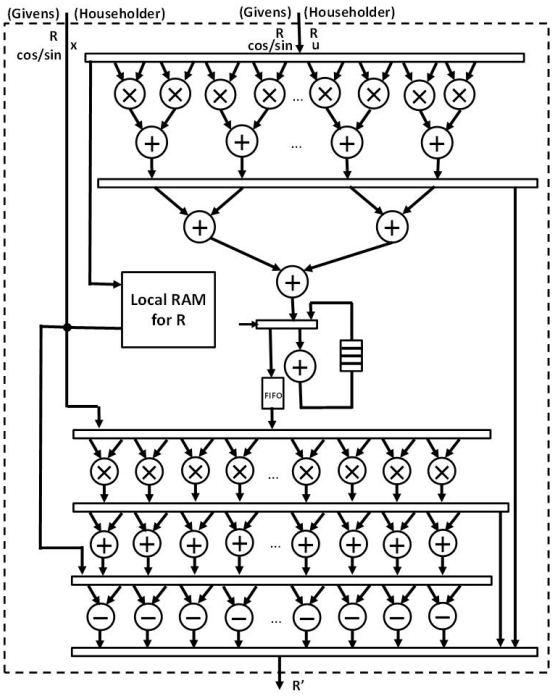


Fig. 3: Matrix updates/preprocessing component architecture.

both computations are required during the matrix update phase. To help optimize hardware resource usage, a deeply pipelined component is devised that performs both preprocessing and matrix update operations. Fig. 2 provides a high-level block diagram of our hybrid QR decomposition architecture. The matrix updates/preprocessing component and the factorization component are the pipelined computational engines. RAMs are employed to temporarily hold the generated Householder vectors and Givens rotation parameters. Additionally to help reduce communication overheads, the current processing sub-matrix column and the adjacent sub-matrix columns are kept in local memory. The adjacent sub-matrix column is moved to the local storage of current processing sub-matrix column when all its respective updates have completed. The number of sub-matrix columns that can be held on chip is determined by the amount of on-chip resources as well as the matrix dimensions.

A. Preprocessing Component

The preprocessing component is responsible for producing the squared vector norms for the Householder transformation and square sums for the Givens rotations. To calculate the squared vector norms, a reverse binary tree structure was implemented as shown in Fig. 3. The top level is equipped with n multipliers, under which there are $\lceil \log(n) \rceil$ levels of adders. An additional adder is employed as an accumulator when the vector length is greater than the number of multipliers. Only the multipliers and top level adders are used to perform preprocessing for Givens rotation. To save hardware resources, the preprocessing component is implemented as part of the matrix update component.

TABLE I: Hybrid QR decomposition approach computations.

Phases	Householder	Givens Rotation
Preprocessing	$x \leftarrow \begin{bmatrix} R_{i,i} \\ R_{i+1,i} \\ R_{i+2,i} \\ \cdots \\ R_{n,i} \end{bmatrix}$ $v_1 \leftarrow R_{i,i}^2 + R_{i,i+1}^2 \cdots + R_{n,i}^2$	$t_1 \leftarrow R_{i,i}$ $t_2 \leftarrow R_{j,i}$ $t \leftarrow t_1^2 + t_2^2$
Factorization	$e_1 \leftarrow \sqrt{v_1}$ $v_2 \leftarrow (R_{i,i} - e_1)^2 + R_{i,i+1}^2 \cdots + R_{n,i}^2$ $e_2 \leftarrow \sqrt{v_2}$ $x_1 \leftarrow R_{i,i} - e_1$ $u \leftarrow \frac{x}{e_2}$	$r \leftarrow \sqrt{t}$ $cos\theta = \frac{t_1}{r}$ $sin\theta = -\frac{t_2}{r}$ $R_{i,i} \leftarrow r$ $R_{j,i} \leftarrow 0$
Matrix Updates	$R \leftarrow R - 2 * u * u^T * R$	$k \leftarrow 1$ **to** $NumofColumn$ $\begin{bmatrix} Q_{k,i} \\ Q_{k,j} \end{bmatrix} \leftarrow \begin{bmatrix} cos\theta & sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} Q_{k,i} \\ Q_{k,j} \end{bmatrix}$

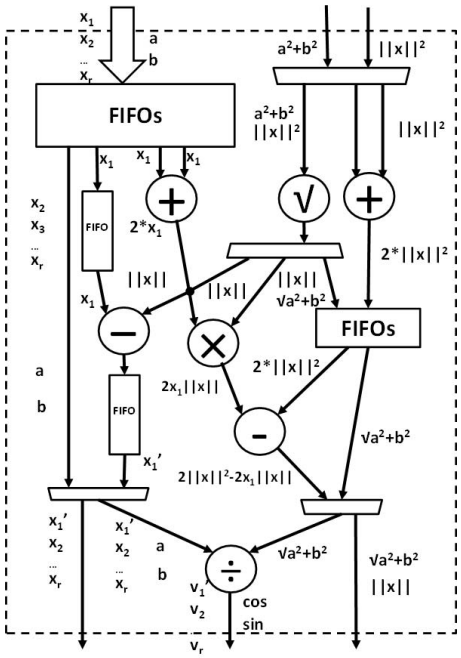


Fig. 4: Dataflow view of the factorization component.

B. Factorization Component

$$e_1 = \sqrt{R_{i,i}^2 + R_{i+1,i}^2 \cdots + R_{n,i}^2} \quad (10)$$

$$e_2 = \sqrt{(R_{i,i} - e_1)^2 + R_{i+1,i}^2 \cdots + R_{n,i}^2}$$
$$= \sqrt{R_{i,i}^2 - 2 * R_{i,i} * e_1 + e_1^2 + R_{i+1,i}^2 \cdots + R_{n,i}^2}$$
$$= \sqrt{2 * (R_{i,i}^2 + R_{i+1,i}^2 \cdots + R_{n,i}^2 - R_{i,i} * e_1)} \quad (11)$$

The factorization component uses a unified architecture to perform both the Householder transformation and the Givens rotation. Fig. 4 illustrates this reconfigurable architecture as a data flow graph. All of the computational cores are deeply pipelined, and factorization can be switched between the Householder transformation and the Givens rotation seamlessly. FIFOs are employed to synchronize the data streams, while FSM-based control units are used to manage the runtime configuration of the architecture.

For the Householder transformation, whose input is the original untransformed vector and the sum of its squared vector elements, the process is started by computing the L2-norm e_1 with the square root operation as shown in eq. (10), where i is the index of the column vector within a sub-matrix and n is its row dimension. In parallel, additions are performed by a pair of adders to calculate $2 * R_{i,i}$ and $2 * (R_{i,i}^2 + R_{i+1,i}^2 \cdots + R_{n,i}^2)$ respectively, in which $R_{i,i}$ is the first entry of the input vector. Then, a multiplier is used to multiply $2 * R_{i,i}$ by e_1, which is followed by the subtraction, $2 * (R_{i,i}^2 + R_{i+1,i}^2 \cdots + R_{n,i}^2) - 2 * R_{i,i} * e_1$, whose result is the square of e_2 (eq. 11). To obtain the Householder vector, a division calculates $\frac{x}{e_2^2}$, in which e_2^2 is the result of the subtractor and x is identical to the input vector except the first entry has been transformed to $x_1 - e_1$. Since the use of a Householder vector is in the form of vector-vector multiplication ($\frac{x}{e_2} * \frac{x'}{e_2}$) in the subsequent matrix updates, its computation can be replaced by $x * \frac{x'}{e_2^2}$. This makes the square root operation of e_2^2 unnecessary. The final output of the Householder transformation is the Householder vector $\frac{x}{e_2^2}$ and e_1, the first and only non-zero entry of the transformed vector.

To perform the Givens rotation, paired matrix elements and the sum of their squares are entered. The square root operation is employed to compute $\sqrt{a^2 + b^2}$, after which $\frac{a}{\sqrt{a^2+b^2}}$ and $\frac{b}{\sqrt{a^2+b^2}}$ are calculated by the divider to produce the Givens rotation parameters $\cos\theta$ and $\sin\theta$ respectively.

C. Matrix Update Component

As described in Table I, the update process of the Householder transformation is conducted through matrix-vector operations, while the Givens rotation uses simple element-wise multiplications and additions/subtractions to update the affected matrix entries. The brute-force way to compute $R \leftarrow R - 2 * u * u^T * R$ is started by the vector-vector multiplication $u * u^T$, whose result is a matrix. Then, the result matrix is multiplied by matrix R, which makes the time complexity of the update process $O(n^3)$. To help optimize the computation,

$u^T * R$ is calculated first, whose product is a row vector. Next, vector-vector multiplication is performed. This reduces the time complexity of the update process to $O(n^2)$, thus largely reducing computational workload, especially for large-scale data sets.

Fig. 3 illustrates the update component architecture. The reverse binary tree structure is responsible for vector-matrix multiplication, while the multipliers in the lower half of the architecture are employed to perform vector-vector multiplication, which is followed by adders to double the results from the multipliers. The Householder transformation update process ends with subtractions. The Givens rotation updates only use the multipliers and the first level of adders below those multipliers. Just as for the factorization component, the architecture is deeply pipelined and can be configured to perform updates for either the Householder transformation or the Givens rotation.

D. I/O considerations

In our design, the input matrix is assumed to be stored in an off-chip memory. Only the column of sub-matrices under processing and their adjacent sub-matrix columns are temporally held on-chip. The number of sub-matrices that can be kept on-chip is determined by the sub-matrix dimension and the capacity of on-chip memory. Our analysis indicates that on average only 4 double-precision floating-point operands need to be communicated between the architecture and the off-chip memory each clock cycle to factorize a 1024×1024 matrix, which consists of 16×16 sub-matrices based on the available on-chip memory.

VI. Implementation and Evaluation

A. Implementation and experimental setup

Our design is implemented in VHDL on a single Xilinx Virtex-5 XC5VLX330 FPGA of the Convey HC-2 platform [22], for which eight memory controllers with a total of 16 DDR2 channels are available. Our architecture uses double-precision floating-point IP cores [23] that are configured to use 9, 14, 57 and 57 pipeline stages for multipliers, adders/subtractors, dividers and square roots respectively. All multipliers, dividers and square roots are implemented as LUT logic only, while adders/subtractors are configured to use LUT logic or dedicated multiplier circuitry (DSPs). To avoid bottlenecks incurred by the bandwidth of on-chip memories, high-bandwith BRAMs are needed. To improve the flexibility of data usage, instead of using single ported high-bandwith BRAMs, simple dual-ported BRAMs were employed, for which each BRAM was configured with a width of 64-bits for input and output ports. The Xilinx design tools reported that our placed and routed design consumes 85.7% of the slice LUTs, 84.4% of DSP48Es and 79.5% of BRAMs. Re-synthesis is required if the input matrix or sub-matrix dimensions are changed.

B. Performance Analysis

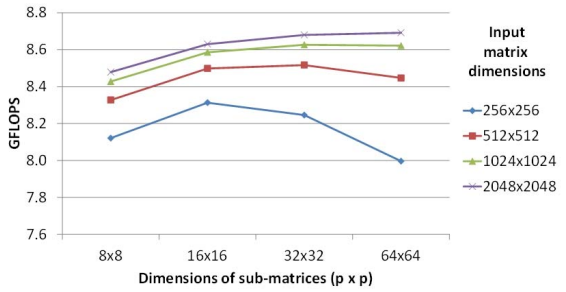
We use GFLOPS (Giga FLoating-point Operations Per Second) as our metric to compare dimensional and partition-dependent performance. In Fig. 5a, the performance of our QR decomposition architecture running at a frequency of 150 MHz is demonstrated, in which different dimensional matrices are applied and various partition strategies are evaluated. Dimensional peak performance is achieved with partitioned sub-matrices of different sizes. Parallelism is able to reduce the idle time of floating-point cores caused by the latencies of accumulation due to vector lengths greater than the number of multipliers used for calculating the norms (16 in our case). Thus, the number of partitions in one sub-matrix column dictates efficiency. In our implementation, at least 16 sub-matrices are needed in a sub-matrix column for efficient computation. This number is the same as the latency of the floating-point accumulator. Processing a sub-matrix column may introduce idle time for computational cores as the number of elements needing processing decreases. Therefore, to obtain the best dimensional performance for square matrices, the number of sub-matrix columns has to be minimized, while sufficient parallelism is maintained.

In Fig. 5b, rectangular matrices are evaluated, in which the number of sub-matrices in a sub-matrix column is equal to that in a sub-matrix row. As row dimension increases, the rectangular sub-matrix becomes more skinny, thus performance improves as more parallelism is achieved, while the remaining elements after the Householder transformation is reduced.

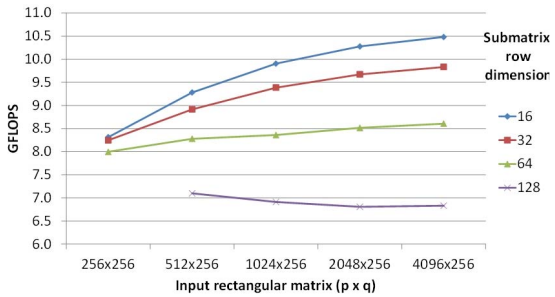
We compare the performance of our design with the results of CPU based MKL implementations [17], a GPU implementation [24], a Matlab routine and a recent FPGA work [10] in Fig. 6. Our design shows speedups of up to $1.46\times$, $1.15\times$ and $13.75\times$ compared to the MKL implementation on a single core [17], FPGA-based tiled matrix decomposition [10] and Matlab routine respectively. We are able to achieve 10.5 GFLOPS, which is 80.5% of the theoretical maximum computation throughput of our design (i.e. 87 floating-point cores $\times$ 150 MHz = 13.05 GFLOPS). In [11], they have demonstrated the performance of FPGA, GPU and CPU (MKL) implementations, although their implementation shows better performance for their FPGA-based tall-skinny matrix QR decomposer, a Virtex-6 SX475T is employed which provides a much higher clock frequency and more computing resources than our platform.

VII. Conclusion and Future Work

A reconfigurable architecture is proposed and implemented on an FPGA-based platform to perform QR decomposition, which exploits both the advantages of the Householder transformation in its efficient vectorized computation and the Givens rotation in its flexible and highly parallel operations. The architecture can be configured to perform either the Householder transformation or the Givens rotation at runtime, in which the Householder transformations are employed to transform the sub-matrices from dense to triangular, while the Givens rotation is used to zero out the remaining unneeded non-zero elements. Our experimental results show our design can achieve a performance of 10.5 GFLOPS with speedups of up to $1.46\times$, $1.15\times$ and $13.75\times$ compared to a MKL implementation, a recent FPGA design and a Matlab solution respectively. For future work, we plan to explore potential



(a) square matrices with square sub-matrices



(b) rectangular matrices with rectangular sub-matrices

Fig. 5: The performance of our architecture for performing QR decomposition on square and rectangular matrices.

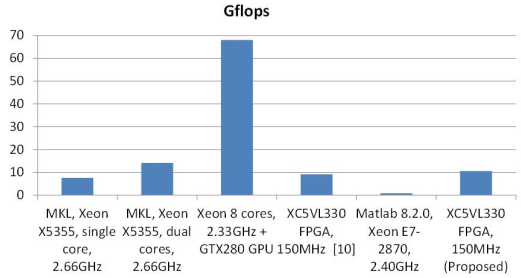


Fig. 6: Performance comparison with single core, multi-core, GPU and recent FPGA work.

applications of our architecture (e.g. beamforming, image recovery).

ACKNOWLEDGEMENTS

This work is supported in part by the National Science Foundation (NSF) under awards CNS-1116810 and CCF-1149539.

REFERENCES

[1] P. Xue, K. Bae, K. Kim, and H. Yang, "Progressive equalizer matrix calculation using QR decomposition in MIMO-OFDM systems," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC)*, 2013, pp. 593–596.

[2] Z. Liu and J. McCanny, "Implementation of adaptive beamforming based on QR decomposition for CDMA," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[3] C. Qiu and N. Vaswani, "ReProCS: A missing link between recursive robust PCA and recursive sparse recovery in large but correlated noise," *The Computing Research Repository*, vol. abs/1106.3286, 2011.

[4] M. J. Anderson, G. Ballard, J. Demmel, and K. Keutzer, "Communication-avoiding QR decomposition for GPUs," in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, May 2011, pp. 48–58.

[5] G. Golub and C. Van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[6] J. Dongarra, M. Faverge, T. Herault, J. Langou, and Y. Robert, "Hierarchical QR factorization algorithms for multi-core cluster systems," in *Proceedings of the IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, May 2012.

[7] M. Soliman, "Efficient implementation of QR decomposition on Intel multi-core processors," in *Prceedings of the Seventh International Computer Engineering Conference (ICENCO)*, 2011, pp. 25–30.

[8] A. Kerr, D. Campbell, and M. Richards, "QR decomposition on GPUs," in *Proceedings of Workshop on General Purpose Processing on Graphics Processing Units (GPGPU)*, Mar 2009.

[9] X. Wang and M. Leeser, "A truly two-dimensional systolic array FPGA implementation of QR decomposition," *ACM Transaction on Embedded Computing System*, vol. 9, no. 1, pp. 1–17, Oct. 2009.

[10] Y.-G. Tai, K. Psarris, and C.-T. D. Lo, "Synthesizing tiled matrix decomposition on FPGAs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, 2011, pp. 464–469.

[11] A. Rafique, N. Kapre, and G. Constantinides, "Enhancing performance of tall-skinny QR factorization using FPGAs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 443–450.

[12] S. Aslan, S. Niu, and J. Saniie, "FPGA implementation of fast QR decomposition based on Givens rotation," in *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2012, pp. 470–473.

[13] M. Leoncini, G. Manzini, and L. Margara, "Parallel complexity of Householder QR factorization," in *Proceedings of the Fourth Annual European Symposium on Algorithms (ESA)*, Sep 1996.

[14] H. Bouwmeester, M. Jacquelin, J. Langou, and Y. Robert, "Tiled QR factorization algorithms," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011.

[15] J. Dongarra, M. Faverge, T. Hérault, J. Langou, and Y. Robert, "Hierarchical QR factorization algorithms for multi-core cluster systems," *The Computing Research Repository*, vol. abs/1110.1553, 2011.

[16] A. El-Amawy and K. R. Dharmarajan, "Parallel VLSI algorithm for stable inversion of dense matrices," *Journal of Computers and Digital Techniques*, vol. 136, no. 6, pp. 575–580, 1989.

[17] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra, "A class of parallel tiled linear algebra algorithms for multicore architectures," *Journal of Parallel Computing*, vol. 35, no. 1, pp. 38–53, Jan 2009.

[18] H. F. Walker, "Implementation of the GMRES method using Householder transformations," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 1, pp. 152–163, Jan. 1988.

[19] H. Bouwmeester, M. Jacquelin, J. Langou, and Y. Robert, "Tiled QR factorization algorithms," *The Computing Research Repository*, vol. abs/1104.4475, 2011.

[20] W. M. Gentleman, "Error analysis of QR decompositions by Givens transformations," *Journal of Linear Algebra and its Applications*, vol. 10, pp. 189–197, 1975.

[21] F. Echman and V. Owall, "A scalable pipelined complex valued matrix inversion architecture," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, 2005, pp. 4489–4492.

[22] "The Convey HC-2 computer architecture overview." [Online]. Available: http://www.conveycomputer.com/

[23] "Logicore IP Floating-point operator data sheet," March 2011. [Online]. Available: http://www.xilinx.com/

[24] S. Tomov, R. Nath, H. Ltaief, and J. Dongarra, "Dense linear algebra solvers for multicore with GPU accelerators," in *Proceedings of the IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, April 2010, pp. 1–8.