Available online at www.sciencedirect.com

ScienceDirect

IERI Procedia 7 (2014) 21 – 27

Procedia IERI

www.elsevier.com/locate/procedia

2013 International Conference on Applied Computing, Computer Science, and Computer Engineering

Design and Performance Analysis of Fixed-Point Jacobi SVD Algorithm on Reconfigurable System

Ramanarayan Mohanty[a], Gonnabhaktula Anirudh[a], Tapan Pradhan[a], Bibek Kabi[a], Aurobinda Routray[a]*

[a]*Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, 721302, India*

Abstract

This paper presents design and performance analysis of fixed-point two sided Jacobi Singular Value Decomposition (SVD) algorithm on reconfigurable system using pipelined state-of-the-art CORDIC architecture. The algorithm has been implemented in reconfigurable hardware with the proposed architecture to achieve faster performance for matrices with larger dimensions. This design has not only reduced the computational complexity but also exploited parallelism in data transfer methods. Various quantization modes along with their range of relative errors are discussed. A comparative study of execution time shows that FPGA implementation with increasing dimension is found to be superior to its floating-point counterpart. Accuracy of FPGA and SystemC based fixed-point implementation is compared based on number of accurate fractional bits, signal-to-quantization-noise-ratio (SQNR), orthogonality and factorization errors with respect to double precision floating-point results.

Selection and peer review under responsibility of Information Engineering Research Institute

*Keywords:*CORDIC, fixed-point arithmetic, singular value decomposition (SVD), reconfigurable architecture, signal-to-quantization-noise-ratio (SQNR).

* Ramanarayan Mohanty. Tel.: +91-947-633-3547; *E-mail:* ramanarayan.mohanty@hotmail.com.

Selection and peer review under responsibility of Information Engineering Research Institute
doi:10.1016/j.ieri.2014.08.005

1. Introduction

With the wide spread application of embedded computing, FPGA based fixed-point architectures have gained considerable importance in the fields of communication [12], signal and image processing [13]. Concurrently with these rapid development over the past few decades, various bottlenecks like computational speed and accuracy have been a major concern. These affect real-time implementation of numerical algorithms like singular value decomposition (SVD). To achieve improved performance, SVD is implemented in reconfigurable system with parallel architecture. SVD of symmetric matrix is computed using established algorithms such as, Jacobi class and QR class [1] and [3]. Jacobi class is the oldest and the slowest one, but it is more accurate ensuring stability with higher degree of potential for parallelism. Since SVD algorithm consists of extensive computation like orthogonal transformation, getting high throughput depends on the design of arithmetic units. These units are the building blocks for high performance computation. COordinate Rotation DIgital Computer (CORDIC) architecture provides a better data independent model for arithmetic designs [6], [14] and [15]. It reduces the complexity of the designs with the help of plane rotation by using simplified hardware components, mainly adders and shifters [7] and [5]. Thus, considerable research has been devoted towards integrating SVD algorithms on specialized parallel architectures for pattern recognition, signal and image processing applications [4].

Most papers so far have only focused on parallel Jacobi SVD using classical CORDIC architecture to compute the singular values of a 2×2 matrix [2]. On the other hand, this paper contributes an architectural design for fixed-point Jacobi SVD algorithm using pipelined state-of-the-art CORDIC technique for larger matrices (40×40). A comparative study of execution time between double precision floating-point and fixed-point FPGA implementation is carried out. With increase in dimension FPGA implementation is found to be superior to its floating-point counterpart. It also analyses the performance of the architecture with respect to floating-point and SystemC fixed-point implementation based on number of accurate fractional bits, signal-to-quantization-noise-ratio (SQNR), orthogonality and factorization errors.

The rest of the paper is organized as follows. Section 2 discusses the CORDIC based Jacobi SVD algorithm and the proposed architecture. Results in terms of number of accurate fractional bits, SQNR, orthogonality and factorization errors for SystemC and FPGA based fixed-point implementation are discussed in section 3. Finally the paper is concluded in section 4.

2. Proposed pipelined CORDIC based Jacobi SVD algorithm

For any real matrix, there exist two orthogonal matrices, and a diagonal matrix such that SVD of 'M' is given as

$$M_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^{T}_{n\times n} \tag{1}$$

where the diagonal elements of Σ (σ_i) are known as singular values and U, V are called left and right singular vectors. If $m = n$ and the matrix is symmetric then U and V span the same vector space. Out of several SVD algorithms Jacobi algorithm works by performing a sequence of orthogonal similarity transformations. The transformation through rotation is done using $Z(p,q,\phi)$ and can be represented as

$$M_{i+1} = Z_i^T M_i Z_i \tag{2}$$

where $Z(p,q,\phi)$ matrix is obtained by changing p^{th} and q^{th} rows and columns of an identity matrix of the same size. With each iteration the updated matrix M_{i+1} is more diagonal than its predecessor. Computation of $M_i Z_i$ in (2) by right rotation takes n CORDIC operations and $Z_i^T(M_i Z_i)$ takes another n CORDIC operations resulting in a total of $2n$ number of CORDIC operations. Singular vectors are obtained

as

$$U^T = \prod_{k=0}^{p} Z_k^T \text{ and } V = \prod_{k=0}^{p} Z_k \tag{3}$$

The value of rotation angle ϕ for each iteration is obtained using (5) which involves elements of a 2×2 matrix selected sequentially from the symmetric matrix M . Thus

$$\begin{bmatrix} a'_{pp} & 0 \\ 0 & a'_{qq} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{4}$$

where a_{pp} , a'_{pp} , a_{qq} , a'_{qq} and are elements of the input matrix before and after rotation respectively.

$$\phi = \frac{1}{2} \tan^{-1} \frac{2a_{pq}}{a_{qq} - a_{pp}} \tag{5}$$

In this design initially ϕ is calculated using vector method and used in the transformation process mentioned in (2).

Since we intend to compute SVD for larger matrices and CORDIC is a recursive algorithm, pipelined architecture is considered to be more suitable. However, this architecture causes data hazards for matrices with dimension less than 21×21 because the latency of the CORDIC and gain compensation units is 20 clock cycles. In order to circumvent this problem we stall the pipeline for the second rotation of the iteration but this increases the execution time (small matrices $< 21 \times 21$). For larger matrices, a single CORDIC clock cycle should include two memory reads and two memory writes. This is not feasible using a dual port Block RAM. Therefore, the memory is made to run at twice the speed of the CORDIC unit to accommodate all the data transfers. The algorithm needs $log_2 n$ sweeps. After a sufficient number of sweeps quadratic convergence is observed while calculating norm of off-diagonal elements of M . Each sweep has $n(n-1)/2$ iterations. Each iteration has $2n$ CORDIC rotations and one CORDIC arctangent for computation of singular values. For finding singular vectors we need n cordic rotations per iteration. Therefore total CORDIC operations needed are: $C(n) = log_2 n \times (n(n-1)/2) \times (2n + n + 1)$.

2.1. Design of the Proposed architecture

The basic workflow diagram of the proposed design is shown in Fig. 1. In this implementation the input matrix used for computation is stored in the Dual-Port Block RAMs of the FPGA. The matrix elements required in each iteration are fetched from corresponding Block RAM using multiplexers and sent to the CORDIC modules which rotate them through the angle provided by the arctangent unit. The elements are then stored back into their respective addresses. The Address and Timing module controls the generation of read/write addresses to Block RAMs in each clock cycle. The module maintains synchronisation among all other blocks and controls the clock signals provided to each block. Arctangent module uses a vectoring mode CORDIC FSM (Finite State Machine) to compute the angle at the beginning of each iteration.

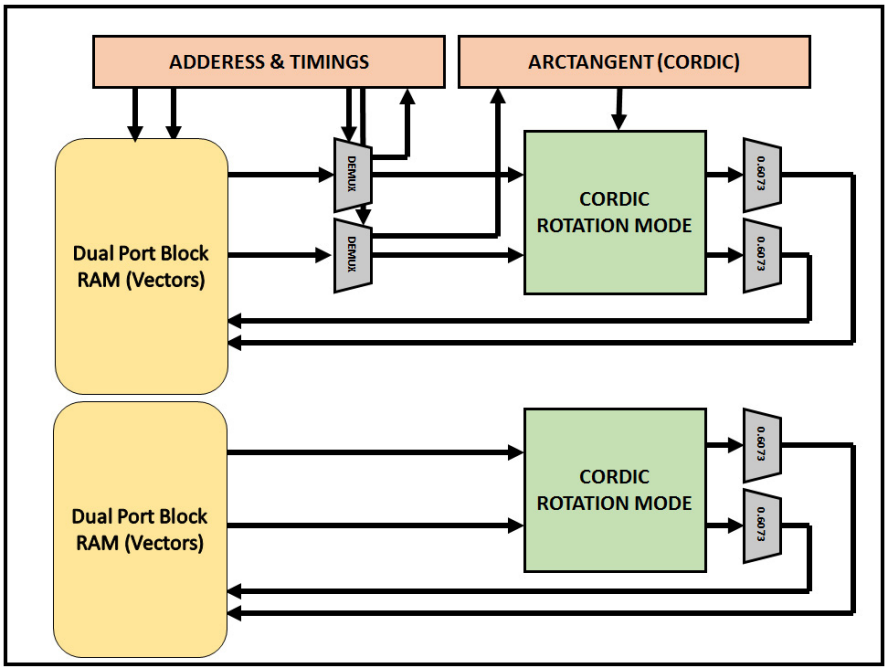


Fig. 1. Workflow diagram of pipelined Jacobi algorithm

The DEMUXs (De-Multiplexers) control the dataflow between rotation mode CORDIC (Singular Vectors) and Arctangent Blocks. At the beginning of each iteration, they route the input to arctangent module for angle calculation and later to CORDIC (rotation) module for rotation. Singular values and vectors are computed simultaneously using two pipelined CORDIC modules arranged in parallel. A multiplier is used after each CORDIC (rotation) module to make the system processing gain unity. The CORDIC used for rotation has 14 pipeline stages and an additional 6 clock cycles are required for CORDIC gain compensation. Gain compensation is done using four 16 -bit multipliers per CORDIC. 16 -bit multipliers are used instead of 32 -bit multipliers to avail the maximum clock speed. The resource utilization for the proposed pipelined CORDIC architecture is summarized in Table 1.

2.2. Implementation

Implementation is based on a Xilinx Spartan-3e development board with an XC3S500E (UG230 package, speed grade-5) FPGA [10]. We use Verilog to describe the design, which is synthesized and mapped to the target platform using Xilinx ISE 14.4 software package. The whole design operates from a single 50 MHz clock. Our FPGA design uses a constant fixed-point representation. In this format variables are assigned with a single Q point definition, which remains same throughout the program. 32-bit wordlength with Q25 format is used for our fixed-point implementation. Optimum Q point is chosen at a minimum error through an iterative process. Constant fixed-point representation is suitable when variables have close dynamic ranges.

We have simulated fixed-point Jacobi algorithm in SystemC using different Q formats obtained from range estimation process. In SystemC fixed-point data is represented using the following attributes [11].

$$< wordlength\,(WL), integer\ wordlength\,(IWL), quantization\ mode, overflow\ mode > \quad (6)$$

Wordlength, integer wordlength, quantization and overflow modes are the most important factors governing the accuracy of fixed-point algorithms [16]. Quantization and overflow are two serious limitations

while converting a floating-point program into its fixed-point counterpart due to insufficient fractional and integer wordlength respectively. There are various quantization modes to handle these errors. Quantization noise arises when data are represented or stored using finite precision arithmetic. During this process excess bits are eliminated and the data is approximated using finite number of bits or wordlength. In an algorithm these errors propagate through fixed-point arithmetic operations and are accumulated at the output. Quantization noise is assumed to be uncorrelated with the signals and also uncorrelated among themselves. There are mainly two types of quantization errors truncation and rounding [8] and [9]. In case of truncation all bits after b bits are discarded. Hence for a positive number the quantized value is always less than the actual value resulting in a negative error. Rounding is similar to truncation except the case that a '1' is added with the LSB to round up or down. Ranges of different relative errors during various quantization modes are shown in Table 2. q is quantization step and is equal to, $q=2^{-b}$, k is the number of bits eliminated from β-bits during quantization and b is the number of bits for fractional part. Overflow or underflow is protected by means of saturation or wraparound process.

Table 1. FPGA resources for pipelined CORDIC design of Jacobi algorithm

Resources	Used	Available	Utilization (%)
Number of Slices	2756	4656	59
Number of Slice Flip Flops	3924	9312	42
Number of 4 input LUTs	5283	9312	56
Number of Block RAMs	8	20	40
Number of MULT18×18SIOs	12	20	60
Number of GCLKs	2	24	8

Table 2. Range of relative errors of different quantization modes

Type of Quantization error	Number representation	Range of relative errors
Truncation	Sign magnitude	$-(2^{-b}-2^{-\beta}) \le \varepsilon_t \le (2^{-b}-2^{-\beta})$
Rounding	All numbers	$-\frac{1}{2}(2^{-b}-2^{-\beta}) \le \varepsilon_r \le \frac{1}{2}(2^{-b}-2^{-\beta})$

3. Results and Discussion

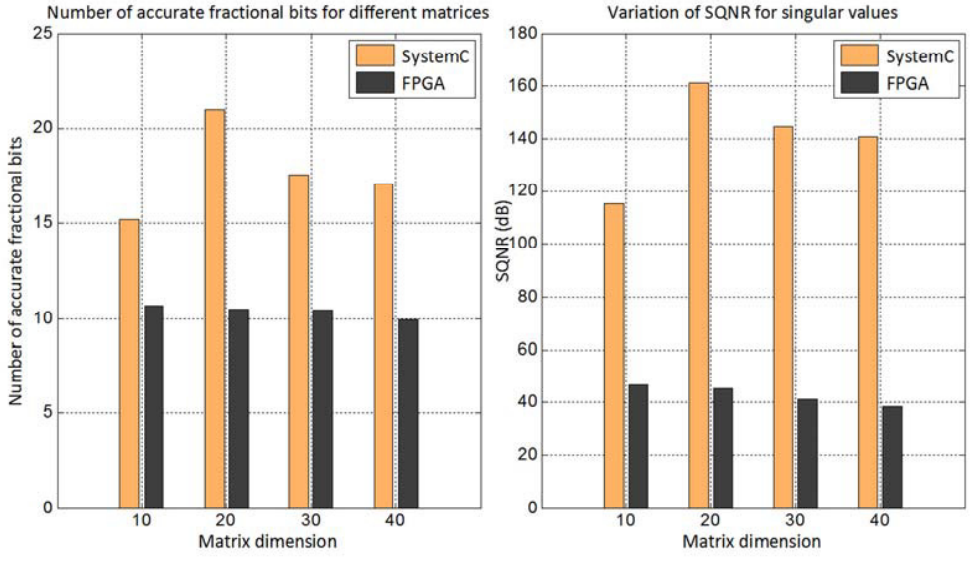
The performance of pipelined CORDIC architecture based Jacobi SVD implementation is shown in Table 3 and Fig. 2. Here the comparative study has been carried out with the double precision floating-point and SystemC based sequential fixed-point simulation in a work station with an Intel core(TM)2 Quad, 2.5 GHz processor and 4GB DDR2 memory. Accuracy is compared on the basis of number of accurate fractional bits, SQNR, orthogonality and factorization errors. Number of accurate fractional bits is computed using (7).

$$Number\ of\ accurate\ fractional\ bits = -\log_2(\max(abs(x_{float} - x_{fixed}))) \quad (7)$$

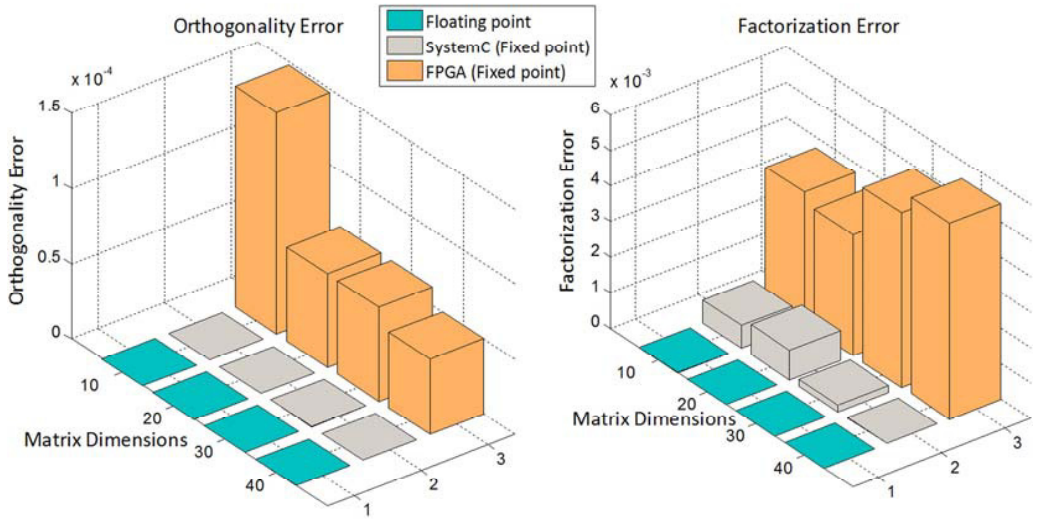
From Fig. 2. (a) we observe that, SQNR and the number of accurate fractional bits are higher in case of SystemC based fixed-point simulation because of variable Q formats. However the FPGA implementation is faster than double precision floating-point implementation (Table 3). This is because of the advantage of proposed architecture involving pipelining. Fig. 2. (a) shows the variation of SQNR for different matrices. Fig. 2. (b) show the accumulation of orthogonality and factorization errors in the output vectors. Primary reasons for the evolution and accumulation of these errors in FPGA implementation are – (*i*) Use of 16-bit multipliers as a correction factor and (*ii*) Insufficient integer wordlength and fractional wordlength.

Table 3. Comparison of Execution Time (Sec)

Matrix Dimension	Floating-Point	Fixed-Point (FPGA)
10	2.0e-4	3.57064e-3
20	3.0e-3	4.28464e-3
30	94.0e-3	12.54604e-3
40	484e-3	26.86384e-3



(a)



(b)

Fig. 2. (a) Number of accurate fractional bits and variation of SQNR for Jacobi SVD; (b) Orthogonality and Factorization error for Jacobi SVD

4. Conclusion

In this paper we propose the pipelined CORDIC architecture of Jacobi SVD algorithm for higher order matrices. The accuracy is compared in terms of number of accurate fractional bits, SQNR, orthogonality and factorization errors with SystemC based results. The results of execution time show that the pipelined CORDIC architecture performs (1000 times) faster than double precision floating-point implementation for larger matrices.

The accuracy can be increased by raising the number of pipeline stages as well as by assigning optimized integer and fractional wordlength for different variables. A faster Jacobi SVD algorithm can be realised through implementing it on GPU. An optimized SystemC based implementation can be realized through a high level synthesis on FPGA.

References

[1] G. Golub and W. Kahan, Calculating the Singular Values and Pseudo Inverse of a Matrix, *J. SIAM Nummer. Anal.*, 1965, Ser. B, 2, 2.
[2] J. R. Cavallaro, F. T. Luk, CORDIC arithmetic for an SVD processor, *J. of Parallel and Distributed Computing*, 1988 5, 3, pp. 271-290.
[3] T. Pradhan, A. Routray, B. Kabi, Comparative Evaluation of Symmetric SVD Algorithms for Real-Time Face and Eye Tracking, *Matrix information geometry*, 2012, pp. 323-340.
[4] S. F. Hsiao, J. M. Delosme, Parallel singular value decomposition of complex matrices using multidimensional CORDIC algorithms, *IEEE Trans. Signal processing*, 1996, 44, 3, pp. 685-697.
[5] M. D. Ercegovac, T. Lang, "Redundant and on-line CORDIC: application to matrix triangularization and SVD", IEEE Trans. on Computers, 1990, 39, 6, pp. 725-740.
[6] R. Andraka, "A survey of CORDIC algorithms for FPGA based Computers", *ACM/SIGDA sixth international symposium on Field Pro-grammbale Gate Arrays*, session, 1998 9, pp. 191-200.
[7] P. K. Meher, S. Y. Park, "CORDIC Designs for Fixed Angle of Rotation", *IEEE Trans. on VLSI sys.*, 2013, 21, 2, pp. 217-228.
[8] D. Menard, R. Rocher and O. Sentieys, Analytical fixed-point accuracy evaluation in linear time-invariant systems, *IEEE Trans. on circuits and sys.-1: Regular papers*, 2008, 55, 19.
[9] R. Rocher, D. Menard, P. Scalart and O. Sentieys, Analytical appraoch for numerical accuracy estimation of fixed-point systems based on smooth operations, *IEEE Trans. on circuits and sys.-I: Regular papers*, 2012, 59, 10.
[10] Spartan-3e starter kit FPGAs: Complete Data Sheet, Xilinx Datasheet DS312 (v4.1), Xilinx Inc., San Jose, CA, Jul. 2013 [Online].Available: http://www.xilinx.com/support/documentation/ data_sheets/ds312.pdf
[11] *IEEE STD 1666-2005 IEEE Standard SystemC Language Reference Manual*,http://standards.ieee.org/getieee/ 1666/download/1666-2005.pdf.
[12] J. Liu, Y. V. Zakharov and B. Weaver, Architecture and FPGA Design of Dichotomous Coordinate Descent Algorithms, *IEEE Trans. on circuits and sys.-1: Regular papers*, 56**,** (2009).
[13] H. Huang and L. Xiao, CORDIC Based Fast Radix-2 DCT Algorithm, *IEEE Signal Process. Lett.*, 20, (2013).
[14] P. M. Szecowka, P. Malinowski, CORDIC and SVD implementation in digital hardware, *17th Intnl. MIXDES Conf.*, 2010, pp. 237-242.
[15] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan, and K. Maharatna, 50 years of CORDIC: Algorithms, architectures, and applications, *IEEE Trans. on Circuits and Systems I: Regular Papers,* 2009, 56(9), 1893-1907.
[16] T. Pradhan, B. Kabi and A. Routray, Fixed-Point Hestenes Algorithm for Singular Value Decomposition of Symmetric Matrices. *Proceedings of the 2013 International Conference on Electronics, Signal Processing and Communication Systems*, 2013.