Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A High throughput Optimized SVD Processor for 8x8 MIMO Precoding with Adaptable Precision Controller

Sudhakar K

sudhakar.cse@kongu.edu

Kongu Engineering College

Jeyakumar P

M.Kumarasamy College of Engineering

Research Article

Keywords: Precoding, SVD, MIMO communications, eigen values,matrix and diagonalization

Posted Date: March 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8479771/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A High throughput Optimized SVD Processor for 8x8 MIMO Precoding with Adaptable Precision Controller

Sudhakar K[1] and Jeyakumar P[2]

[1*]Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode, 638060, Tamilnadu, India.

[2]Department of Electronics and Communication Engineering, M.Kumarasamy College of Engineering,, Thalavapalayam, 639113, Tamilnadu, India.

Contributing authors: sudhakardhivya18@gmail.com; jeyakumarpsg@gmail.com;

Abstract

The Singular Value Decomposition (SVD) Processor for 8X8 MIMO precoding systems is designed. The 2-sided Cyclic Jacobi algorithm is used in our work. Here, the whole matrix is split into 2x2 matrix and are processed separately, which at each iteration exchange values between them .Through the use of an adaptive precision controller, the processing matrix size has been efficiently reduced, and the diagonalization process has been accelerated to boost throughput. 32 bit Fixed the data path format is utilised to accommodate the increased accuracy of Eigen vector elements and Eigen values from the 8x8 MIMO channel matrix. The thresholds for precision can be modified and thus the accuracy of the SVD processor is changeable according to the user requirements. The SVD processor provides throughput of 675k matrix/s in 40nm CMOS technology.

Keywords: Precoding, SVD, MIMO communications, eigen values,matrix and diagonalization

1 Introduction

Millimeter wave (mmWave) Multiple Input Multiple Output (MIMO) system is one of the most essential techniques to increase throughput rate for 5G based communications.The hybrid analog/digital precoding technique together forms radio frequency(RF) precoding vectors and a precoding matrix to control the high pathloss in the mmWave channel, and to get the acceptable spectral efficiency with reduction in power of RF chain and complexity[1]. The important issue is of generation precoding matrices based on the "channel matrix decomposition result" for the RF precoder and the digital baseband, respectively. A new algorithm has been proposed that calculates precoding matrices in different phases to attenuate the calculation overheads of matrix updates in each iteration[2]. The system performance can also be enhanced by using precoding technology but the complexity is increased in mmWave systems when working on antenna arrays. To reduce the hardware cost, a new method of creating the joint Radio frequency and baseband precoder reduces the complexity of the computation of the precoder restoration algorithm and allows a hardware architecture that is highly parallel[3]. For a optimum balance between the design architecture complexity and it's throughput rate, an SVD processor for MIMO precoding systems is implemented which is a architecture built upon memory with various processing basic elements[4]. A hardware friendly implementation scheme for calculating SVD by 2 by 2 Givens rotations as fundamental operation is devised . The scheme is also implemented as a pipelined architecture of 2 levels to use the computing parallelism efficiently for high throughput process[5].

The MIMO precoding[6] based on a high number of antennas has become important.Numerous matrix fragmenting schemes are used for MIMO precoding. The SVD de-links the spatially multiplex communication channel and therefore it makes possible to transmit the eigen value-sub channels independently[7].

The two sided Jacobi algorithm uses right Jacobi rotations and left jacobi rotations to annihilate the off diagonal elements of the matrix [8],[9] while in [10], Hestenes-Jacobi method is implemented. A modified two sided Jacobi process is implemented in [11]. The super linear convergence rate method i.e. self-power method, , is adopted in [12].Golub-Kahan (GK)-SVD, the two-phase algorithm, containing bi-diagonalization and QR reduction method is used in [13] and [14].For diagonalization. iterative processes are generally performed Thus, to enhance the throughput, the methods to expedite convergence are essential. designing a hardware well planned architecture for computing steering matrices in MIMO (Multiple Input Multiple Output) communication systems. Steering matrices are essential for various MIMO algorithms, including beamforming and spatial multiplexing, which are used to optimize signal transmission and reception.

The architecture presented is aimed at reducing the computational complexity and hardware resources required for matrix operations. It employs

optimization techniques to streamline the computation process, such as efficient algorithms for matrix multiplication and reduction, thereby minimizing the number of operations and enhancing throughput

The following can be used to summarise this work's main contributions: First, thresholding methods that quicken the diagonalisation process are incorporated into the implementation, and precision is supplied with adjustable precision from 10^{-5} to 10^{-2}, also Dynamic assignment of the No. of cycles for Sine and cosine value generator are being done, according to the weights of the input matrix, while we can also control the number of cycles or iterations for the rotation of the matrix.

The CORDIC algorithm is presented as a solution due to its suitability for hardware implementations. This algorithm is favored for its simplicity and efficiency, utilizing basic shift-and-add operations, which are well-suited for FPGA hardware. Unlike traditional methods that may involve complex arithmetic operations, CORDIC provides a way to compute trigonometric functions iteratively through rotation, which aligns well with the parallel processing capabilities of FPGAs. The introduction further discusses the advantages of FPGA over other hardware platforms for implementing the CORDIC algorithm. FPGAs offer parallelism and configurability, which can lead to significant improvements in computation speed and efficiency. By employing the CORDIC algorithm on FPGA, the paper aims to achieve a solution that meets the demands of real-time applications where both performance and accuracy are crucial. Overall, the paper sets out to demonstrate how the CORDIC algorithm can be effectively utilized on FPGA to compute trigonometric functions, addressing the need for fast and accurate computational methods in various technological applications[15].

In the following, Sec. II. describes the Multi Input Multi Output precoding system and Singular Value Decomposition algorithm. The proposed adaptable precision controller is illustrated in Sec. III, while In Sec. IV, Hardware architecture is presented The Sec. V, contains the working flow of the system and the complete SVD architecture. Implementation results and the comparisons are given in Sec. VI. Sec. VII is the conclusion.

2 MIMO Coding and the SVD Technique

MIMO (Multiple Input Multiple Output) precoding and the SVD (Singular Value Decomposition)[16] algorithm are crucial in optimizing wireless communication systems. MIMO technology increases capacity and reliability by utilising many antennas at the transmitter and receiver. Precoding involves applying a transformation to transmitted signals based on the channel state information(CSI) to enhance signal quality and reduce interference.

Let us assume a MIMO precoding [17] system withN^T Ttransmitter antennas and N^R receiving antennas.Assuming that S streams of data are being transmitted the received signal "y"can be expressed in

$$y = HFx + n \tag{1}$$

where $H \in C^{N_R \times N_T}$ is the complex matrix of the channel; $F \in C^{N_T \times S}$ is our matrix for precoding; $x = [x_1 x_2 \cdots x_S]^T$ of length S is the user data size; $n \in C^{N_R \times 1}$ is the AWGN . Assume that $E\{x^H x\} = E_x$ and $E\{nn^H\} = . The$ $\gamma^2 I N_R$ signal-to-noise ratio can be defined as $\alpha = E_x/\gamma^2$. If the matrix is breakdown by SVD, then

$$H = U\Sigma V^H \tag{2}$$

where $U \in C^{M \times M}$ and $V \in C^{N \times N}$ represent unitary matrices that express the left eigen matrix and the right eigen matrix, respectively; $\Sigma \in R^{M \times N}$ is a diagonal matrix containing the real eigen or singular values.

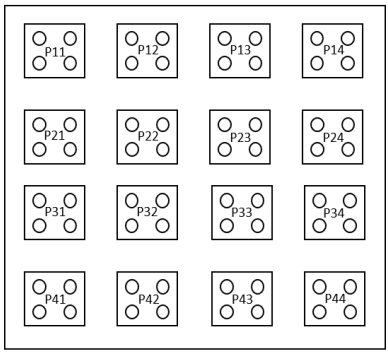


Fig. 1. 8 X 8 Matrix into 16 autonomous processors

For a matrix of dimension 2 x 2, the eigen values and vectors calculation is achieved using cyclics Jacobi's technique, which denotes that, if matrix A is a real matrix which is also symmetric and if U is a matrix which is orthogonal then

$$S = UAU' \tag{3}$$

will be diagonal, and the right diagonal of S is made up of eigen values of A in jumbled order. For larger matrices of size more than 4x4 , the matrix splited into 2 x 2 matrices[18] and as shown in Fig. 1, the processors perform autonomously and trade elements between them at each iteration. The algorithm is iterative in which at each iteration, the processed matrix is Jacobi-rotated by the angle which comes from the diagonal processors after working on the diagonal matrices. In Figure 1, P11, P22, P33 and P44 are the 2x2 diagonal matrices and remaining matrices are off-diagonal matrices, that are sent to the diagonal processors in next iteration. For example,for a symmetric matrix of size M x M. The 2 sided cyclic Jacobi's algorithm used for the matrix having size larger than 2 x 2, to compute Eigen value and eigen vectors is explained below. Extract the elements of the 2x2 matrix A which is at the diagonal. Let m, n, o and p be the elements of A .if

$$2 * |n| < \epsilon * |m - p| \tag{4}$$

then make sine=0 and cosine = 1, where ϵ denotes a very small number. If the condition is false, then we are required to perform the following computations

for the particular diagonal matrix elements. A 2x2 SVD can be illustrated as

$$R(\theta_t)^T \begin{bmatrix} m & n \\ o & p \end{bmatrix} R(\theta_r) = \begin{bmatrix} \psi & 0 \\ 0 & \psi \end{bmatrix} \tag{5}$$

where θ_l and θ_r are the jacobi left rotation and jacobi right rotation angles, respectively. The jacobi rotation matrix generated is

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \tag{6}$$

and the matrix elements are M = $\begin{bmatrix} m & n \\ o & p \end{bmatrix}$ The rotation angles θ_l and θ_r essential to diagonalize the input matrix M can be expressed as the inverse tangents or arctan of the input matrix data values as:

$$\theta_r + \theta_l = \arctan\left(\frac{c+b}{d-a}\right) \tag{7}$$

$$\theta_r - \theta_l = \arctan\left(\frac{c-b}{d+a}\right) \tag{8}$$

The left rotation angle and right rotation angle can be computed using the equations (7) and (8)for a 2x2 matrix. Optimum choice of the double-sided rotations can lead the matrix A to concur to a diagonal matrix consisting of eigen values. At each iteration k, a left jacobi rotation operation of θ_l and a right jacobi rotation operation of θ_r in the plane i j are selected to minimize the off-diagonal elements a_{ij} and a_{ji} of A to a value approximately zero. All the operations that minimize the off-diagonal values in the diagonal processors are autonomous and hence can be processed in concurrent to each other. To update the 8x8 array, the left rotation angles are carried from the diagonal processor to every off diagonal processor in the similar row, and the right rotation angles are carried to each off diagonal processors in the same column. This transmission is called systolic array transmission. After the update of the matrix, a change in the order of the rows and the columns of the 8x8 matrix is carried out according to the ordering such that such that it is parallel. An 8x8 connected array is needed to carry out this trade of data. The permutation leads to the next set of matrices, which takes the process into next iteration and further annihilates the new diagonal elements,eventually resulting in the final singular values of the matrix.

3 Proposed Adaptable Precision Controller

An Adaptable Precision Controller dynamically adjusts the precision of computations or data representations based on specific requirements or conditions. It balances accuracy with resource usage, optimizing performance in various

contexts such as digital signal processing, machine learning, and embedded systems.

The controller assesses the current needs, adjusts precision levels accordingly, performs computations, and monitors the system to ensure performance and accuracy. It also includes error management strategies to handle inaccuracies resulting from precision changes. This approach enhances performance, resource efficiency, and flexibility by adapting precision to meet varying demands and operational constraints The main idea that reduces the number of cycles, is the redundancy based upon the type of the input given to the system.If the weight of the input matrix elements is small,then the computation of the eigen values through this algorithm converges with the same accuracy in less number of cycles.

The Sine and Cosine generator, which works upon the iterations to reach to the exact values iteration by iteration, required accuracy can be altered to reach to the result much faster because the error produced in case of the bits having less weights is not significant enough, since the matrix bits have to be multiplied by these values and if these matrix elements having small weights, it does not deviate the result significantly thus giving us the same result with the required accuracy in less number of cycles.

This is not true for the inputs for higher weights. So, the system needs to work on both the modes based upon the input matrix, which is first checked by the Adaptable Precision Controller. Precision of the overall system can also be altered according to the user requirements ranging from 10^{-2} to 10^{-5} Parameter "mode" is taken as input from the user to the Adaptable precision controller that defines the four modes our system can work in :

Mode = 4'b0001: Identifies the matrix elements weights and assigns the precision and cycles dynamically.

Mode = 4'b0010: User defined operation, set the precision to 10^{-2}.

Mode = 4'b0100: User defined operation, set the precision to 10^{-5}.

Mode = 4'b1000: Normal Operation, for the best accuracy but takes more computation time.

4 Hardware Architecture

4.1 Sine and Cosine(Jacobi Matrix Generator)

Coordinate Rotation Digital Computer,was formulated by Jack E. Volder in the year 1959, which is an Hardware efficient algorithm to compute sine and cosine, It is an iterative Algorithm for the purpose of Circular Rotation without the involvement of multiplication process and Delay/Hardware cost is near to division operation or process of finding square root[19]. The elementary functions evaluation is embedded as a generic rotation operation, which reduce the rotation operation into iterative elementary rotations, where each rotation is realised with basic shifters and adders.CORDIC algorithm's iterative method, which simplifies the computation of trigonometric functions through basic shift and add operations, making it well-suited for FPGA hardware. The paper

details the FPGA architecture designed to perform these computations, including precision adjustments and optimization techniques to balance accuracy and resource usage. The results demonstrate that FPGA-based implementations offer high-speed computation and efficient use of hardware resources, making them ideal for real-time applications in digital signal processing and communication systems.

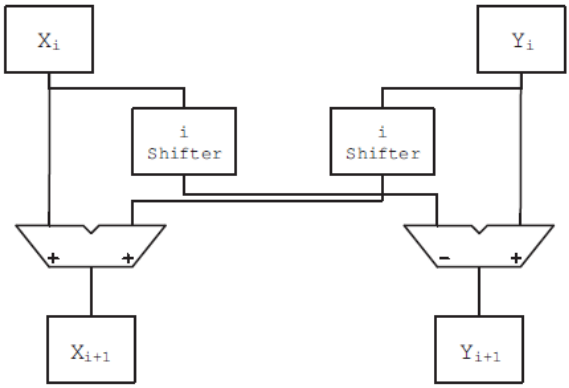


Fig. 2. Basic Rotational Unit

The angle input is given to the angle compiler. The X_{in} and Y_{in} values are programmed with 1 and 0 respectively. The CORDIC operation results the updated version of sine and cosine values in the X_{out} and Y_{out} after the required operation. The CORDIC algorithm makes the operation of multiplication a part of its rotation operation.Therefore, the design will not be having multiplier hardware[19]. Fig. 2, depicts the basic CORDIC hardware for the rotational operation. The Number of iterations in our module are flexible from 10 to 31 which provides the adaptable precision capability for the system.

4.2 Diagonal and Off-diagonal Processors

Fig. 3. depicts the architecture of the Diagonal processor, that takes the 2X2 matrix as input, then finds the required angle and the subsequent Jacobi Matrix which is then sent to Matrix multiplier, that generates the rotated matrix in which the off diagonal elements are annihilated.

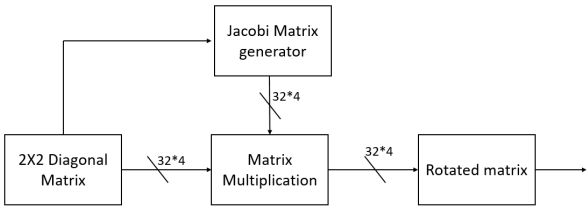


Fig. 3. Diagonal Processor architecture

Fig. 4. depicts the architecture of the Off- diagonal processor, that takes the off diagonal matrix from the 8X8 matrix and uses the Jacobi matrix generated by the Diagonal processor and then generates the required rotated off-diagonal matrix elements which are then sent to diagonal processor to be annihilated based on a fixed order of rotation.

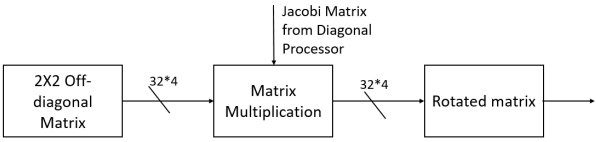


Fig. 4. Off-Diagonal Processor architecture

4.3 Order Generator

The rows and columns of the 8X8 matrix are switched in a predetermined order following the matrices' rotation. This process of diagonal matrices and off diagonal matrices is carried until the matrix will get con-cured to a matrix that has only diagonal elements. For completion of a step, matrices columns and rows are exchanged between adjoining processors, so that new elements are sent to the diagonal processors for the next iteration, that makes the diagonal elements zero[4]. The interchanging row and column order is such that any pair does not occur more than once and it takes "n-1" steps, means for our case such pairs are generated in 7 steps.

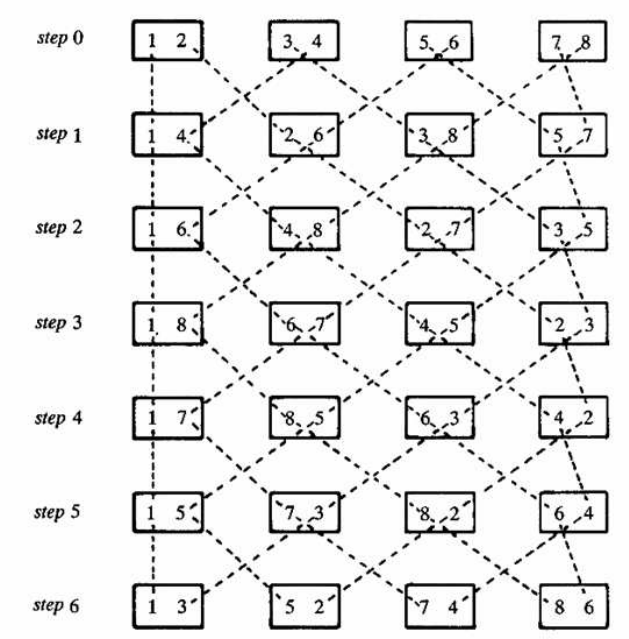


Fig. 5. Order of rotation of rows and column

The Fig. 5. gives the order of rotation of rows and columns used in subsequent iterations

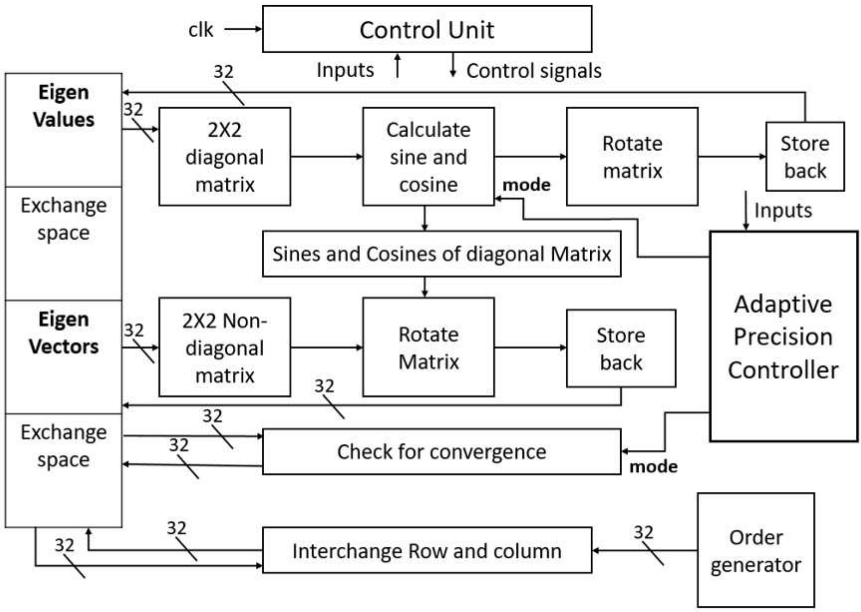


Fig. 6. Overall Block diagram of SVD processor

5 SVD Processor and Model workflow

The Fig. 6 is the overall architecture of our SVD processor, the data path for higher Precision is chosen to be 32 bit fixed point numbers, with 1 bit signed, 15 bit integer and 16 bit fraction part. For each iteration the 8x8 matrix is divided into 16 2x2 matrices which are then sent to the APC(Adaptable Precision Controller) for detection of the "mode" in which our processors will work, accordingly the number of cycles will be fixed for the diagonal and off-diagonal processors, In the diagonal processors, angles are generated by the eq. (7) and eq. (8) and rotation matrix in equation (6) is generated by the help of these angles. Then the processed matrix after matrix multiplication is generated by both diagonal and off diagonal processors as in eq. (5).

This completes one single iteration of the process. For the next iteration, the order generator will exchange the rows and columns according to the next step as shown in fig. 6, so that the new 2x2 matrices go to the diagonal processors and the off-diagonal elements can be eliminated to generate the final eigenvalues after all the iterations are complete.

Resource	Utilization	Available	Utilization %
LUT	9154	64000	14.30
LUTRAM	382	17600	2.17
FF	6647	128000	5.19
DSP	106	160	66.25
IO	324	400	81.00

Fig. 7. Utilisation after Synthesis

Resource	Utilization	Available	Utilization %
LUT	9116	64000	14.24
LUTRAM	352	17600	2.00
FF	6647	128000	5.19
DSP	106	160	66.25
IO	324	400	81.00

Fig. 8. Utilisation after Implementation

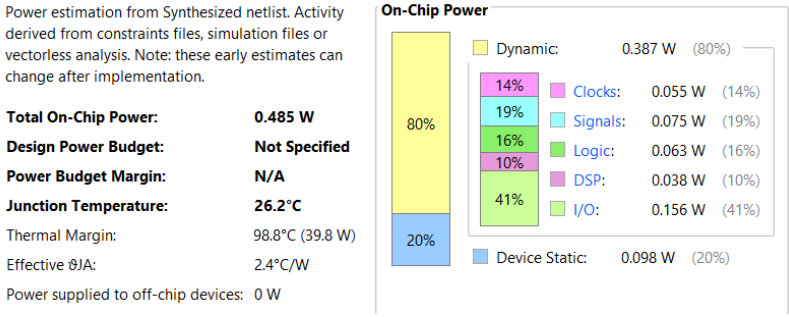


Fig. 9. Power report after Synthesis

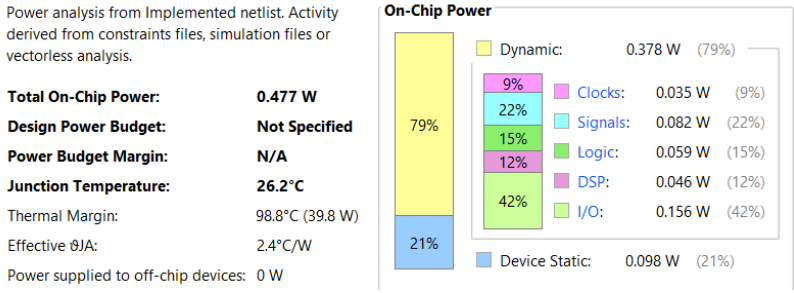


Fig. 10. Power report after implementation

6 Implementation results and comparison

The proposed SVD processor is realised using the two sided jacobi algorithm in which it takes 17 and 117 cycles respectively for off diagonal and diagonal

processes, while rest cycles for further matrix multiplication. Diagonal processors take more cycles to calculate the Jacobi matrix at every iteration, while the Off-diagonal processors use the Jacobi matrix generated by the diagonal processors. While we can see through the simulation results, that the computation time comes down when we use other modes of operation of our processor, which gives lesser precision and based upon the input elements weight, this brings down the computation cycles required.

The design was implemented on 40 nm CMOS Technology, Fig. 7. depicts the device utilisation summary results after the synthesis, where number of LUTs and LUTRAMs used are 9154 and 382 respectively, while after the optimisation and implementation the LUTs anf LUTRAMs used are reduced to 9116 and 352 respectively as given in Fig. 8. Power after synthesis is depicted in Fig. 9., where we can observe the system takes 485mW power, of which dynamic power share is at 80% as 387mW and rest 20% is static power of 98mW, while after the optimisation and implementation the power is reduced to 477mW as shown in Fig. 10., of which dynamic power share is 79% at 378mW and Static power 21% at 98mW.

The average number of cycles required are 293 and throughput with an increase of over 8% as compared to previous designs is 675k (matrix/second) at 198 MHz operating frequency.

Table 1: TABLE OF COMPARISON

Parameters	[20]	[4]	This work
Implementation	Synthesis	ASIC layout	Synthesis
Algorithm	GR-SVD	Golub-SVD	2-sided Jacobi
Matrix dimension	8x8	2x2-8x8	2x2-8x8
Datapath Representation	19 bit Fixed	19 bit Fixed	32 bit Fixed
Precision	adjustable	adjustable	adjustable + adaptable
No. of Cordics	16	16	16
Technology	45nm	40nm	40nm
Computation Cycles	313	368	293
Clock Freq.	199 MHz	185 MHz	198 MHz
Gate Count	526K	555K	605K
Core Voltage	0.9V	1V	1V
Max. Throughput(Matrix/sec)	636K	591K	675K
Power(mW)	266.7	303	478
Power efficiency	2.38^a	1.95^b	2.37^a

a: Synthesis result. b: Chip result.

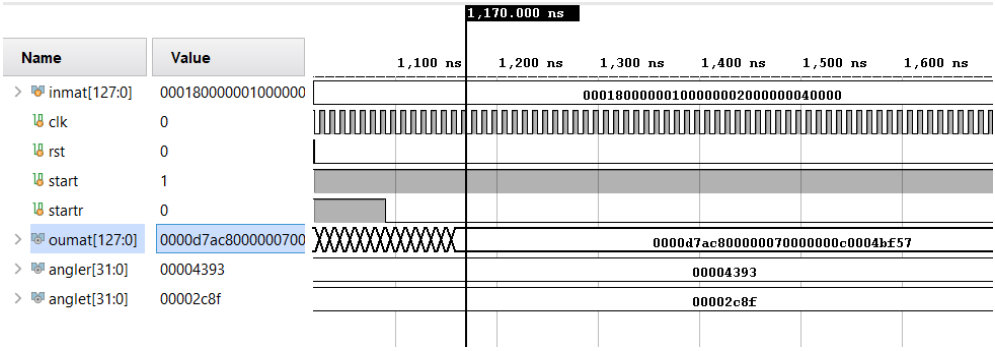


Fig. 11. Simulation results

7 Conclusion

In this paper, we have implemented the SVD processor, with adaptable Precision controller having the capability to dynamically reduce the number of cycles of the jacobi matrix generator with the range of 10 to 31 based upon the input elements weights, systolic array architecture is also used to help exchange data between the independent processors without any delay so that after the completion of one iteration next can start immediately without any delay, 16 CORDICs are used for the implementation of the Jacobi Matrix generators which helps the hardware to be free of the bulky multipliers and calculate the sine and cosine of the angles using adders and shifters. The 32-bit Fixed-Point representation is used in the data path, processor is implemented on Spartan 7 FPGA board, in which total LUTs used are 9116 which leads to 15% of the available LUTs in our device consuming 478mW of power, the SVD processor generates output at a throughput rate of 675K matrix/s at 198MHz operating frequency.

Declaration

There are no conflicts of interest, according to the authors.

Author's contributions

The implementation steps were carried out by Sudhakar K. and Jeyakumar P. for the publication "A High throughput Optimized SVD Processor for 8x8 MIMO Precoding with Adaptable Precision Controller" to which all authors participated. The authors of the first draft of the manuscript were Sudhakar

K., Jeyakumar P; they also offered feedback on earlier drafts of the work. The final manuscript has been read and approved by all writers.

Acknowledgments

Funding: This research article does not have any funding.

Availability of data and material

Since no data sets were created for this article's current investigation, data sharing is not applicable.

References

[1] Ho, C.-K., Cheng, H.-Y., Huang, Y.-H.: Hybrid precoding processor for millimeter wave mimo communications. IEEE Transactions on Circuits and Systems II: Express Briefs **66**(12), 1992–1996 (2019)

[2] Chen, K.-T., Hwang, Y.-T., Liao, Y.-C.: Vlsi design of a high throughput hybrid precoding processor for wireless mimo systems. IEEE Access **7**, 85925–85936 (2019)

[3] Lee, Y.-Y., Wang, C.-H., Huang, Y.-H.: A hybrid rf/baseband precoding processor based on parallel-index-selection matrix-inversion-bypass simultaneous orthogonal matching pursuit for millimeter wave mimo systems. IEEE Transactions on Signal Processing **63**(2), 305–317 (2014)

[4] Wu, C.-H., Tsai, P.-Y.: An svd processor based on golub–reinsch algorithm for mimo precoding with adjustable precision. IEEE Transactions on Circuits and Systems I: Regular Papers **66**(7), 2572–2583 (2019)

[5] Hwang, Y.-T., Chen, K.-T., Wu, C.-K.: A high throughput unified svd/qrd precoder design for mimo ofdm systems. In: 2015 IEEE International Conference on Digital Signal Processing (DSP), pp. 1148–1151 (2015). IEEE

[6] Zhao, L., Li, J., Huang, S., Wu, X., Jiang, M.: Low-complexity hybrid precoding for sub-connected millimeter-wave massive mimo systems. Signal Processing **219**, 109381 (2024)

[7] Perahia, E., Stacey, R.: Next Generation Wireless LANs: 802.11 N and 802.11 Ac. Cambridge university press, ??? (2013)

[8] Yang, C.-H., Chou, C.-W., Hsu, C.-S., Chen, C.-E.: A systolic array based gtd processor with a parallel algorithm. IEEE Transactions on Circuits and Systems I: Regular papers **62**(4), 1099–1108 (2015)

[9] Chiang, C.-H., Huang, S.-A., Chen, C.-E., Yang, C.-H.: A 2× 2–16× 16 reconfigurable ggmd processor for mimo communication systems. In: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2018). IEEE

[10] Guenther, D., Leupers, R., Ascheid, G.: A scalable, multimode svd precoding asic based on the cyclic jacobi method. IEEE Transactions on Circuits and Systems I: Regular Papers **63**(8), 1283–1294 (2016)

[11] Cuomo, S., Marcellino, L., Navarra, G.: A parallel implementation of the hestenes-jacobi-one-sides method using gpu-cuda. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 722–725 (2018). IEEE

[12] Athi, M.V., Zekavat, S.R., Struthers, A.A.: Real-time signal processing of massive sensor arrays via a parallel fast converging svd algorithm: Latency, throughput, and resource analysis. IEEE Sensors Journal **16**(8), 2519–2526 (2016)

[13] Senning, C., Studer, C., Luethi, P., Fichtner, W.: Hardware-efficient steering matrix computation architecture for mimo communication systems. In: 2008 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 304–307 (2008). IEEE

[14] Zhan, C.-Z., Chen, Y.-L., Wu, A.-Y.: Iterative superlinear-convergence svd beamforming algorithm and vlsi architecture for mimo-ofdm systems. IEEE Transactions on Signal Processing **60**(6), 3264–3277 (2012)

[15] Kumar, P.A.: Fpga implementation of the trigonometric functions using the cordic algorithm. In: 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 894–900 (2019). IEEE

[16] Chen, B., Yang, L., Wu, M.: Generalized singular value decomposition-based secure beam hybrid precoding for millimeter-wave massive mimo systems. Applied Sciences **15**(7) (2025)

[17] Tarboush, S., Ali, A., Al-Naffouri, T.Y.: Compressive estimation of near-field channels for ultra-massive mimo wideband thz systems. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5. IEEE, ??? (2023)

[18] Mahale, G.V., Bartakke, P.P.: Eigen values and vectors computations on virtex-5 fpga platform cyclic jacobi's algorithm using systolic array architecture (2011)

[19] Renardy, A.P., Ahmadi, N., Fadila, A.A., Shidqi, N., Adiono, T.: Fpga

implementation of cordic algorithms for sine and cosine generator. In: 2015 International Conference on Electrical Engineering and Informatics (ICEEI), pp. 1–6 (2015). IEEE

[20] Wu, C.-H., Liu, C.-Y., Tsai, P.-Y.: Design of an svd engine for 8× 8 mimo precoding systems. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–4 (2017). IEEE