# Low Latency and Hardware-Efficient Singular Value Decomposition Algorithm for Real-Time Applications

Pujari Lokesh
*Dept. of Electrical Electronics and Communication Engineering*
*Sharda University*
*Greater Noida, India*
email: pujarilokesh0009@gmail.com

Ruqaiya Khanam
*Dept. of Electrical Electronics and Communication Engineering,*
*Centre for Artificial Intelligence in Medicine, Imaging & Forensic,*
*Sharda University*
*Greater Noida, India*
email: dr.kruqaiya@gmail.com

Debarghya Paul
*Dept. of Electrical Electronics and Communication Engineering*
*Sharda University*
*Greater Noida, India*
email: debarghyapaul06@gmail.com

Sachin Kumar
*Dept. of Electrical Electronics and Communication Engineering*
*Sharda University*
*Greater Noida, India*
email: sachinmizo1223@gmail.com

***Abstract*— In linear algebra, the Singular Value Decomposition (SVD) method divides a matrix into the product of three matrices that has many real time applications which includes Artificial Intelligence, Machine Learning, Image Compression, Bioinformatics, Signal Processing and so on. Even nowadays engineers facing problems on calculating SVD for different matrix by using the different algorithms due to its more computation - time & more power - consumption in real - time applications. Previously, algorithms like Golub- Kahan(GK) Algorithm, Divide & Conquer(D&K) Algorithm, and Jacobi Algorithm were implemented on Field Programmable Gate Array(FPGA) and analyzed computation-time & power-consumption. In this paper, we selected the Jacobi algorithm for implementing SVD on FPGA because of its more accuracy and high parallel processing capacity than other algorithms, so it can give less computation time, the utilization of hardware resources and power consumption was improved over earlier efforts by taking into account the CORDIC algorithm for computing trigonometric functions and writing code in HLS.**

***Keywords*— *Singular Value Decomposition, Two-sided Jacobi Algorithm, CORDIC Algorithm, FPGA, HLS***

## I. Introduction And Related works

SVD breaks the any complex matrix into the product of three matrices in linear algebra, this process is very important in many real time applications like control systems[1], image compression[2], radar signals compression[3], image denoising[4], bioinformatics[5], machine learning, data science, artificial intelligence and so on. Obtaining high speed computation process, low power utilization, and less resources utilization on hardware are main issues facing while implementing SVD in real world applications. Computation speed and power utilization is very important. For achieving the high computation speed we used, two-sided Jacobi algorithm which have more accuracy and high parallel processing capacity than other algorithms and CORDIC algorithm which computes trigonometric functions in our proposed work. CORDIC algorithm not only increases the computation speed but also decreases power utilization and resources utilization on hardware. We implemented SVD on the ZCU104 FPGA, and we analyzed the computation speed, power consumption, and resources utilization on FPGA by writing algorithm in C programming language by using HLS tool.

SVD mathematical theory was developed by Eugenio Beltrami and Camille Jordon independently in 1870's [6]. Golub developed an more computationally stable algorithm in 1960[6], after this many algorithms were developed for implementing SVD. Some algorithms provides the serial implementation and some provides the parallel implementation. Normally serial algorithms can be used where the time limitation is not a major issue while implementing SVD[7]. SVD was computed in a few articles, however the hardware utilization increased exponentially as the input matrix's dimension increased.Reference[8] showed Jacobi algorithm is most accurate algorithm among all algorithms. SVD was implemented in many FPGAs platforms like Virtex 2 Pro[9], XC 3S 500E[10] , XC 6S 100t[11] and so on, analyzed the power consumption, calculation time and resources utilization. Mesh connected architecture of Jacobi algorithm was implemented on FPGA[9]. Fast and generic architecture of SVD was implemented on FPGA[12]. Some were also tried to implement SVD on GPU[12].

## II. Background

### A. Singular Value Decomposition(SVD)

As mentioned earlier SVD of matrix has a great number of applications in real world and it is useful for researchers in engineering field. Many algorithms like Golub-Kahan (GK) Algorithm, Divide & Conquer(D&K) Algorithm, Jacobi algorithm were developed for calculating SVD. Among all algorithms Jacobi algorithm shows greater

accuracy and parallel processing capacity, it is more suitable approach for hardware implementation [13,14].

*B. Jacobi Algorithm*

Jacobi algorithm has two methods: one-sided Jacobi method and two-sided Jacobi method. In one-sided Jacobi method, for making an matrix diagonal only one-sided matrix multiplication will happen either with left-sided one or right-sided one, but in two-sided Jacobi method, both left-sided and right-sided matrix are responsible for making the matrix diagonal also known as Singular values matrix. Because of its high precision, direct computation of singular vectors, and high parallel processing capability, the two-sided Jacobi technique is more observable in real-time applications, despite being more expensive to implement in hardware than the one-sided Jacobi method.

*C. Two-sided Jacobi Algorithm*

Singular Value Decomposition of a any real m×n matrix is decomposition of that matrix into product of the three matrices, consider 'S' be the real m×n matrix:

$$S = U \sum V^T \quad (1)$$

In Equation (1), U is the left sided singular vector of dimension m×m matrix which has orthonormal columns, ∑ is also called as singular values matrix with m×n dimension which has positive elements arranged in descending order and $V^T$ is right sided singular vector of dimension n×n matrix which has orthonormal rows.

Matrix S is diagonalized using a sequence of plane rotations in the two-sided Jacobi technique for SVD computation. Two Jacobi rotational matrices are multiplied in each rotation. To create the singular values matrix ∑, the left and right Jacobi rotational matrices are multiplied in each rotation, setting all of matrix A's non-diagonal components to zero.

$$S_0 = S$$

$$S_{p+1} = U_p^T S_p V_p = J_p^L(l,k,\Theta)^T S_p J_p^R(l,k,\Phi), (0 \le p \le p_r) \quad (2)$$

In Equation(2), $J^L(l,k,\Theta)^T$ is multiplied with matrix S, which is called left-sided Jacobi rotation matrix, and it only affects rows l and m, $J^R(l,k,\Phi)$ is called as right sided Jacobi rotation matrix. These two rotational matrices are independent of each other so they can multiply once at a time without any difficulty for getting the results. By determining the actual values of the Θ and Φ, which are also known as rotational angle for two sides, forming the $J^L(l,k,\Theta)^T$ and $J^R(l,k,\Phi)$ matrices, and its multiplication with S matrix for multiple rotations produces the required singular values matrix∑. These rotations/iterations goes till the entire non- diagonal elements of matrix S becomes zero or nearly zero.

$$S_{p+1} = J_p^L J_{p-1}^L J_{p-2}^L \cdots J_1^L J_0^L S J_0^R J_1^R \cdots J_{p-1}^R J_p^R \quad (3)$$

$$\sum = S_{p+1}, \quad V = \prod_{j=0}^{p} J_j^L, \quad U = \prod_{j=0}^{p} J_j^R$$

Equation(3), represents the $S_{p+1}$ is the required singular value matrix ∑ after p sweeps/iterations, after sequential multiplications of right and left Jacobi rotation matrices left-sided and right-sided singular vectors are obtained.

At every iteration, l and k values are chosen such that they can make non-diagonal elements of matrix S to zero for producing singular values matrix. Each column is orthogonal to the others following a sweep.

For finding the values of left and right rotational angles of left-sided & right-sided Jacobi rotational matrices, following formulas can be used:

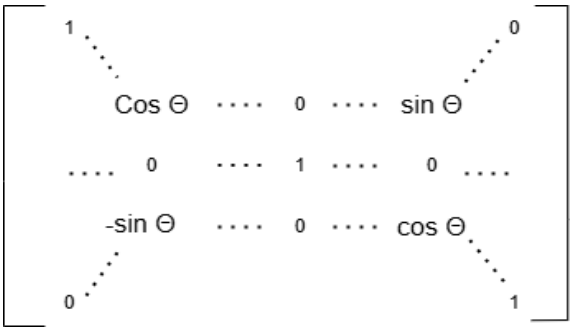
$$\Phi = \frac{1}{2}\left(tan^{-1}\frac{a_{kl}+a_{lk}}{a_{kk}-a_{ll}} + tan^{-1}\frac{a_{kl}-a_{lk}}{a_{kk}+a_{ll}}\right) \quad (4)$$

$$\Theta = \frac{1}{2}\left(tan^{-1}\frac{a_{kl}+a_{lk}}{a_{kk}-a_{ll}} - tan^{-1}\frac{a_{kl}-a_{lk}}{a_{kk}+a_{ll}}\right) \quad (5)$$

Equations(4,5) represents the formulas for left and right rotational angles, $a_{ml}$ , $a_{lm}$ , $a_{mm}$ , $a_{ll}$represents the elements in input matrix S.

For finding the Jacobi rotation matrix J(l, k, Θ), elements in the identity matrix which are in place l row ,l column and k row, k column has to replaced with cos Θ terms, l row and k column element has to be replaced by sin Θ and l column and k column element has to be replaced by -sin Θ terms.

Desired Jacobi rotation matrix:




(6)

Matrix(6) represents the Jacobi rotation matrix for the n×m input matrix, which means this can be used for any dimension input matrix. Based on dimension of input matrix, number of sweeps are decided, during every sweep Jacobi rotation matrices has to be determined, after the completion of sweeps, the series multiplication of rotation matrices gives the left-sided, right-sided singular vectors and singularvalue matrix.

Two-sided Jacobi algorithm calculates the both left-sided and right-sided rotation matrices at the same time independently, in this way algorithm does the parallel processing.

Jacobi Rotational matrices has to be find at every sweep to get the accurate singular values and singular vectors matrices at end.

By increasing number of sweeps/iterations while calculating SVD, accuracy of singular values & singular vectors can be increased. But for having the resources constraints on the FPGA, certain number of sweeps has to be decided for getting the required accuracy. So, taking the certain number of sweeps for getting required accuracy is very important and very crucial part while calculating the

SVD. The detailed explanation of algorithm is explained in Pseudo_code1.

Pseudo_code1. SVD using two-sided Jacobi algorithm

**input:** matrix S
**output:** singular vectors and singular values

1. **for** each sweep (sweeps number based on input matrix(S) dimension)
2. **for** l=1,2,…..,n-1
3. **for** k=l+1,……,n
4. Find values of Θ and Φ using CORDIC algorithm
5. Update the left and right singular vectors
6. $S_{p+1}=J^{L}(l,k,\Theta)^{T} S_p J^{R}(l,k,\Phi)$
7. Sort Singular values($S_{p+1}$)in descending order
8. Rearrange the Singular vectors(U and V) based on sorted singular values
9. **end**

Determine the l and k values based on dimension of the input matrix, if matrix dimension is m×m then l value goes up to m-1 and k value goes up to m. If it goes then only desired and accurate singular values and singular vectors can be obtained at the end.

*D. Algorithm Execution halt*

Sweeps number plays a important role in deciding accuracy of singular values matrix by making non- diagonal elements zero. Based on deciding the actual number of sweeps one can reduce the execution time of the algorithm. Many experiments had been performed by [16] to determine the average number of sweeps that are required for getting accurate singular values based on size of input matrix. Brief about sweeps required has been discussed in the below table I.

TABLE I. REQUIRED SWEEPS[15]

| S.No. | Matrix Size | Required Sweeps |
|---|---|---|
| 1. | 4×4 | 2.96 |
| 2. | 6×6 | 3.63 |
| 3. | 8×8 | 4.07 |
| 4. | 10×10 | 4.39 |
| 5. | 20×20 | 5.23 |
| 6. | 30×30 | 5.67 |
| 7. | 40×40 | 5.92 |
| 8. | 50×50 | 6.17 |
| 9. | 100×100 | 6.81 |

*E. CORDIC Algorithm*

CORDIC algorithm is very useful for calculating the trigonometric functions like sine and cosine values. It calculates the trigonometric values by vector rotations on unit circle. The point of intersection of vector and circle gives the sine and cosine values. Based on the input angle, vector moves on the circle from x=1,y=0 by using certain angles finally the vector reaches to the certain location by making some angle with positive X-axis, that angle is approximately equal to the input angle. The point of intersection of vector and circle at final angle is (x,y), here x value gives the cosine value of input angle & y value gives the sine value of input angle.

Equations which represents CORDIC algorithm are:

$$x_{t+1} = x_t - d_t y_t 2^{-n} \quad (7)$$

$$y_{t+1} = y_t + d_t x_t 2^{-n} \quad (8)$$

$$A_{t+1} = A_t - d_t \arctan(2^{-n}) \quad (9)$$

Where,

$d_t$ = -1 , if $A_t \geq 0$

= 1 , if $A_t < 0$

Equations(7,8,9) have to use at every iteration/rotation up to t rotations, if t value is more sine and cosine values are more accurate, x, y and A represents the x-coordinate, y-coordinate, and angles respectively. The value d has to select based on the based angle A.

The advantage of CORDIC algorithm is more in hardware implementation, because it reduces the computation time, because here instead of multiplication of bit shifting will happens. This bit shifting process reduces the computation work which is happening for multiplication process.

## III. PROPOSED MODEL

We implemented SVD by using two-sided Jacobi algorithm, in process of calculating SVD, we calculated singular values, left-sided & right-sided singular vectors. For calculating the singular values and singular vectors, Jacobi rotation matrix is required, for finding the Jacobi rotation matrix, trigonometric sine and cosine are required. For reducing the computation work while calculating trigonometric functions we used CORDIC algorithm. This CORDIC algorithm find the approximate values of sine and cosine values, supplies them to the Jacobirotation matrix.

.

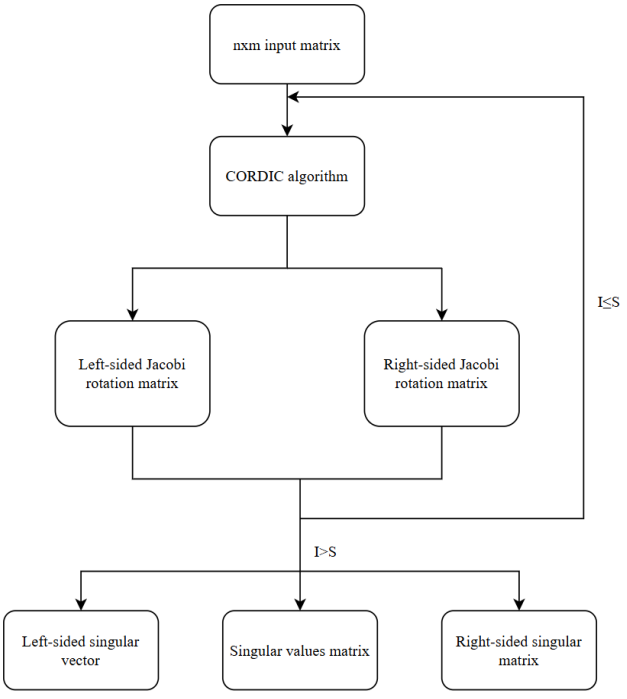



Fig.1. Block diagram of implemented SVD

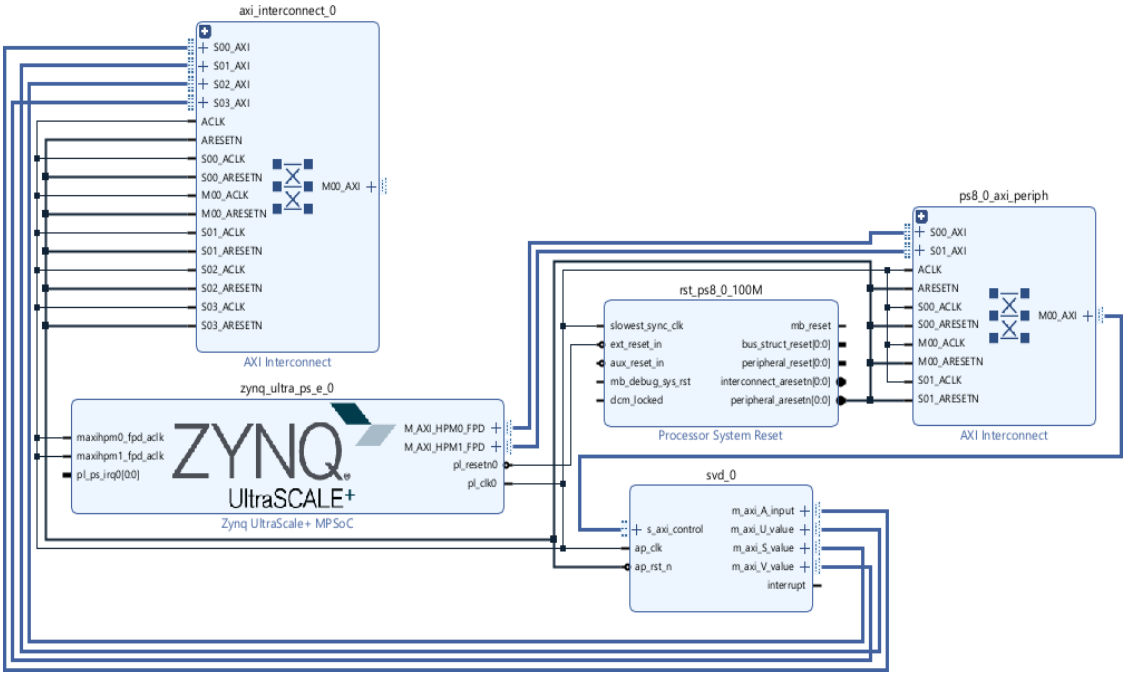



Fig.2. Block design for SVD IP integration

Fig.2. shows the block design from the tool Vivado 2023.2 version. This block diagram shows integration of proposed SVD IP with ZYNQ processor and AXI interconnector bus.
Instead of using traditional ways for computing sine and cosine values, using CORDIC algorithm decrease computation time, power utilization, resources utilization on FPGA of SVD using two-sided CORDIC algorithm
We implemented SVD using Jacobi algorithm on ZCU104 FPGA, analyzed power consumption, computation time, resources utilization for different dimensions of input matrix and compared with previous works on SVD.
Fig(1) represents the block diagram of implemented SVD, in which left-sided and right-sided Jacobi rotation matrices are calculated by using CORDIC algorithm for calculating trigonometric functions. S represents required sweeps and I represents the iteration number, if I is less than or equal to required sweeps, then left and right sided Jacobi matrices has to be calculated that much of time. Left-sided singular vector, singular values matrix and right-sided singular vector are obtained in SVD after the completion of number of sweeps.

## IV. Performance Evaluation

The discussed algorithm was written in C programming language using the Vivado(Vitis HLS) tool. By simulating that algorithm in Vitis HLS 2023.2, we exacted the results of calculation time, resources utilization of hardware, maximum frequency by selecting the ZCU104 FPGA board for different dimensional matrices. We calculated the calculation time and presented in table II.

Table II. Calculation time on FPGA

| S.No. | Size of matrix | FPGA(ZCU104) latency | XC 6S 100t Latency[11] |
|---|---|---|---|
| 1. | 2x2 | 3.160us | 0.3us |
| 2. | 4x4 | 18.780us | 12.1us |
| 3. | 8x8 | 0.189ms | 0.536ms |
| 4. | 10x10 | 0.336ms | 1.001ms |
| 5. | 20x20 | 2.707ms | 12.1ms |
| 6. | 50x50 | 45.499ms | 69.5ms |

Dimensions 2x2, 4x4, 8x8, 10x10, 20x20, 50x50 matrices has been considered in our work, and calculation time has been calculated. In proposed design it is shown that utilized hardware has been reduced by 7% in BRAMS, 22% in LUT compared to previous designs. Even our proposed design shows calculation time decrement for high dimensional matrices. So, calculation speed is more. More calculation speed is very useful in real time applications for faster implementation and working.

TABLE III. COMPARISON WITH PREVIOUS WORKS

| Reference Number | FPGA | Resources Utilization | Frequency | Latency |
|---|---|---|---|---|
| [10] | XC 3S 500E | 2576+12 DSP+ 8BRAMs | 50MHz | _ |
| [9] | Virtex 2 Pro | 8130 | 12MHz | _ |
| [16] | XC 3S 500E | 1140+2 BRAMs | 90MHz | _ |
| [17] | XC 3S 1000 | 6140+ 24 DSP+ 120BRAMs | 58MHz | _ |
| [11] | XC 6S 100t | 1504+ 16 DSP+ 3 BRAMs | 127.5MHz | 12.1us |
| [18] | XC7VX485T | 10% BRAM, 37% LUT | 48MHz | 25us |
| Proposed design | ZCU104 | 3%BRAM, 3% DSP, 5% FF, 15% LUT | 100MHz | 18.07us |

Table III shows the comparison of our proposed work with previous works, this result shows the improvement in frequency with compared previous works. This shows pipelining stages were executed properly and loop unrolling increases the parallel processing in our work.

## V. CONCLUSION

Implementing SVD using Two-Sided Jacobi algorithm, with CORDIC algorithm for calculating Jacobi rotation matrices decreases the computation time. Our work was implemented by using HLS software, it provides #pragmas which are special directives and are useful for unrolling the loop, dividing the pipelining stages properly, this increases parallel processing, computation speed and improvement in frequency has been achieved. So, by using our implemented method in real applications decreases the computation time. While implementing SVD for higher dimensional matrices take the required number of sweeps, so that computation time can be decreased. Above simulations results shown the improvement in utilization of hardware resources. SVD using Two-sided Jacobi algorithm may be optimized more in future for implementing it in real time applications.

## REFERENCES


[1] F. Fairman, "Jordan form realization via singular value decomposition", IEEE transactions on circuits and systems, vol. 35, no. 11, p. 1431-1434, 1988.

[2] H. Prasantha, H. Shashidhara, K.B. Murthy, "Image compression using SVD", IEEE International Conference on Computational Intelligence and Multimedia Applications, 2007.

[3] A. Kwarteng, and Y.M. Missah, "Radar Signals Compression using Singular Value Decomposition (SVD) Approach", International Journal of Computer Applications, vol. 150, no. 12, 2016.

[4] J. Orchard, M. Ebrahimi and A. Wong, "Efficient nonlocal-means denoising using the SVD," *2008 15th IEEE International Conference on Image Processing*, San Diego, CA, USA, pp. 1732-1735, doi: 10.1109/ICIP.2008.4712109, 2008.

[5] Z. . -H. Duan, L. S. Liou, T. Shi and J. A. DiDonato, "Application of singular value decomposition and functional clustering to analyzing gene expression profiles of renal cell carcinoma," *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, Stanford, CA, USA, pp. 392-393, doi: 10.1109/CSB.2003.1227341,2003.

[6] Stewart, G.W., On the early history of the singular value decomposition. SIAM review, vol. 35, no. 4, pp. 551-566, 1993.

[7] Dongarra, J.J. and D.W. Walker, Software libraries for linear algebra computations on high performance computers. SIAM review, vol. 37, no. 2, pp. 151-180, 1995.

[8] Demmel, J. and K. Veseliü, Jacobi's method is more accurate than QR. SIAM Journal on Matrix Analysis and Applications, vol. 13, no. 4, pp. 1204-1245, 1992.

[9] W. Ma, M. E. Kaye, D. M. Luke and R. Doraiswami, "An FPGA-Based Singular Value Decomposition Processor," *2006 Canadian Conference on Electrical and Computer Engineering*, Ottawa, ON, Canada, pp. 1047-1050, doi: 10.1109/CCECE.2006.277355,2006.

[10] Mohanty.R, Anirudh.G, T. Pradhan, B. Kabi, A. Routray, "Design and performance analysis of fixed-point Jacobi SVD algorithm on reconfigurable system". IERI Procedia vol. 7, pp. 21-27, 2014.

[11] A. Shiri and G. K. Khosroshahi, "An FPGA Implementation of Singular Value Decomposition," *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, Yazd, Iran,pp. 416-422, doi: 10.1109/IranianCEE.2019.8786719,2019.

[12] Y. Ma and D. Wang, "Accelerating SVD computation on FPGAs for DSP systems," *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, Chengdu, China, pp. 487-490, doi: 10.1109/ICSP.2016.7877882, 2016.

[13] Golub, G. and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix. Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis,. vol. 2, no. 2p. 205-224.

[14] Demmel, Applied numerical linear algebra,. 1997.

[15] Brent, R.P. and F.T. Luk, A systolic architecture for almost linear time solution of the symmetric eigenvalue problem., Cornell University, 1982.

[16] Rahmati, M., M.S. Sadri, and M.A. Naeini. FPGA based singular value decomposition for image processing applications. in International Conference on Application-Specific Systems, Architectures and Processors, ASAP. 2008.

[17] L. M. Ledesma-Carrillo, E. Cabal-Yepez, R. d. J. Romero-Troncoso, A. Garcia-Perez, R. A. Osornio-Rios and T. D. Carozzi, "Reconfigurable FPGA-Based Unit for Singular Value Decomposition of Large m x n Matrices," *2011 International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, pp. 345-350, doi: 10.1109/ReConFig.2011.77, 2011.

[18] M. TIAN, M. SIMA and M. McGUIRE, "Behavioral Implementation of SVD on FPGA," *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Louisville, KY, USA, pp. 495-500, doi:10.1109/ISSPIT.2018.8642667,2018.