Pipelined IVRU with Multi Bank Memory for Scalable SVD Acceleration Systems

Gundala Praveena[1] , Koppula Manasa[1*]
[1]Department of Electronics & Communication Engineering, Vaagdevi Engineering College, Warangal, 506005, Telangana, India.
*Correspondence: Koppula Manasa (manasa436@gmail.com)

To Cite this Article
Gundala Praveena, Koppula Manasa, "Pipelined IVRU with Multi Bank Memory for Scalable SVD Acceleration Systems", Journal of Science Engineering Technology and Management Science, Vol. 03, Issue 07, July 2026, pp: 15-30, DOI: http://doi.org/10.64771/jsetms.2026.v03.i07.pp15-30
Submitted: 17-05-2026 *Accepted: 24-06-2026* *Published: 01-07-2026*

Abstract
The increasing adoption of high-dimensional data processing in image analysis, wireless communication, machine learning, biomedical signal processing, and real-time embedded systems has significantly increased the demand for efficient Singular Value Decomposition (SVD) hardware accelerators. SVD plays a critical role in applications such as feature extraction, noise reduction, image compression, adaptive filtering, MIMO communication systems, pattern recognition, and data analytics, where rapid and accurate matrix decomposition is essential for real-time decision-making. However, existing SVD architectures predominantly rely on sequential largest-element search mechanisms, conventional COordinate Rotation DIgital Computer (CORDIC) -based rotation processors, and limited memory organizations, resulting in increased search latency, memory access bottlenecks, higher computational complexity, reduced throughput, and slower convergence for large-scale matrices. To address these limitations, this work proposes a novel parallel SVD architecture incorporating an Iterative Vector Rotation Unit (IVRU), Parallel Scan Search Module (PSSM), Address and Data Synchronization Unit (ADSU), and Multi-Bank Memory (MBM). The PSSM accelerates dominant element identification through concurrent comparisons, while the MBM enables simultaneous read and write operations to eliminate memory contention. The ADSU ensures synchronized and conflict-free communication among processing modules, and the IVRU performs efficient iterative vector rotations for rapid matrix diagonalization. The integration of these modules significantly reduces computational latency, improves memory bandwidth utilization, enhances convergence speed, increases processing throughput, and provides a scalable hardware platform for high-performance real-time SVD computation in advanced signal processing and intelligent computing applications.

Key words: Singular Value Decomposition (SVD), VLSI Architecture, Iterative Vector Rotation Unit (IVRU), Parallel Scan Search Module (PSSM), Multi-Bank Memory (MBM), High-Performance Computing, Real-Time Signal Processing, Hardware Accelerator.

1. Introduction

The rapid growth of digital technologies has resulted in an unprecedented increase in data-intensive applications across communication systems, artificial intelligence, image processing, healthcare analytics, autonomous vehicles, and cloud computing platforms. Industry reports indicate that global data generation is expected to exceed 180 zettabytes annually in the coming years, while more than 90% of the world's data has been generated within the last few years. As a result, matrix-based computations have become fundamental building blocks in modern computing systems. Among various matrix decomposition techniques, SVD is recognized as one of the most powerful mathematical tools for extracting meaningful information from large datasets. SVD is widely used for dimensionality

reduction, feature extraction, signal enhancement, image compression, recommendation systems, and machine learning model optimization.

The increasing complexity of modern applications has significantly expanded the demand for high-performance matrix computation hardware. In image processing systems, SVD is employed for noise removal, image reconstruction, and pattern recognition. In wireless communication systems, it is used for channel estimation and Multiple-Input Multiple-Output (MIMO) signal processing. Similarly, artificial intelligence and machine learning frameworks utilize matrix decomposition algorithms to accelerate data analysis and improve computational accuracy. As these applications continue to grow in scale and complexity, efficient hardware implementation of matrix decomposition algorithms has become essential for achieving real-time performance.

The advancement of semiconductor technologies has enabled the development of powerful VLSI architectures capable of handling large-scale matrix operations. However, increasing matrix dimensions require higher computational resources, larger memory bandwidth, and efficient data management mechanisms. Consequently, researchers and engineers continuously explore advanced architectural solutions to optimize matrix computation performance while minimizing hardware resource utilization, processing latency, and energy consumption. These requirements have made efficient SVD implementation a critical research area in modern VLSI system design.

2. Literature Survey

Basiri et al. [1] presented efficient VLSI implementations of Singular Value Decomposition by investigating optimized hardware architectures for matrix decomposition operations. The study focused on reducing computational complexity through structured decomposition techniques and efficient arithmetic processing units. The architecture emphasized hardware resource optimization while maintaining numerical accuracy during matrix factorization. Parallel processing mechanisms were incorporated to improve computational throughput and accelerate decomposition speed. The implementation also explored scalable VLSI design strategies suitable for high-dimensional matrix computations.

Vishnu et al. [2] developed a hardware-efficient QR decomposition algorithm and corresponding VLSI architecture for eigenvalue decomposition of symmetric matrices. The methodology employed optimized QR iterations to improve computational efficiency and reduce arithmetic overhead. Dedicated processing elements were utilized to support matrix transformation operations with reduced hardware requirements. The design focused on improving decomposition accuracy while maintaining efficient data flow management. The design mainly targeted eigenvalue decomposition and offered limited flexibility for generalized large-scale SVD computations.

Cheng et al. [3] introduced a highly parallel Singular Value Decomposition framework for low-latency MIMO signal processing applications. The methodology employed a four-step parallel decomposition strategy based on Gram matrix tridiagonalization to reduce execution latency. Hardware-aware profiling and complexity analysis were incorporated to evaluate scalability under different MIMO configurations. The design emphasized parallel execution of decomposition stages to improve throughput. The framework demonstrated significant latency reduction for large wireless communication systems. The architecture required extensive parallel hardware resources, which can increase implementation cost and power consumption.

Ramasubramanian et al. [4] introduced the MANOJAVAM FPGA accelerator for matrix multiplication and Singular Value Decomposition in principal component analysis applications. The architecture combined unified acceleration strategies to support multiple matrix-processing workloads within a single hardware framework. FPGA-based parallel computational units were employed to accelerate matrix operations efficiently. The system emphasized scalability and adaptability for high-dimensional data analysis tasks. Resource-sharing mechanisms were incorporated to improve hardware utilization.

The FPGA-oriented implementation may experience performance limitations when deployed in ASIC-based VLSI environments.

Mishra and Kumar [5] developed a VCG signal compression framework based on Singular Value Decomposition. The methodology decomposed biomedical vectorcardiogram signals into singular components to reduce redundancy while preserving critical diagnostic information. Matrix factorization techniques were applied to achieve efficient compression performance. The approach focused on improving storage efficiency and transmission capability for healthcare applications. The method concentrated primarily on compression performance and did not address hardware optimization challenges.

López-López et al. [6] proposed a configurable parallel architecture for Singular Value Decomposition of correlation matrices using Jacobi-based decomposition and CORDIC-assisted Givens rotations. The design supported both 4×4 and 8×8 matrix configurations through configurable processing structures. Parallel computational modules were integrated to accelerate matrix transformations and improve throughput. CORDIC units were utilized to generate rotational parameters efficiently during iterative decomposition. The architecture targeted signal processing and MIMO communication applications requiring real-time computation. The repeated CORDIC iterations and memory access operations contributed to increased latency and hardware overhead.

George et al. [7] designed an area-efficient VLSI architecture for parallel Jacobi-based eigenvalue decomposition with inherent eigenvalue sorting capability. The methodology employed parallel Jacobi rotations to accelerate matrix diagonalization while simultaneously performing eigenvalue ordering. Optimized processing units were incorporated to reduce area consumption and improve computational efficiency. The architecture emphasized parallel execution to enhance decomposition speed. Integrated sorting functionality eliminated the need for additional post-processing stages. The architecture remained dependent on iterative rotational computations, resulting in increased processing delay for larger matrices.

Pei et al. [8] presented a temperature field characterization approach based on Singular Value Decomposition for microsystem analysis. The methodology applied matrix decomposition to extract dominant thermal patterns and evaluate temperature non-uniformity. SVD-based feature extraction was employed to improve modeling accuracy. The framework enhanced characterization precision in microscale systems.

The study focused on analytical modeling and did not consider hardware-efficient VLSI implementation aspects.

Nagarajan et al. [9] introduced a qualitative data augmentation framework using mixed-decomposed convolutional networks for VLSI circuit performance prediction. The methodology combined decomposition-based feature extraction with deep learning techniques to improve prediction accuracy. Augmented datasets were generated to strengthen model generalization capability. Convolutional neural networks were employed to learn performance characteristics from circuit parameters. The approach enhanced predictive reliability for VLSI performance estimation. The deep learning framework required substantial computational resources and increased training complexity.

Chi et al. [10] developed a multilinear generalized SVD processor for multicast multiuser MIMO communication systems. The architecture employed tensor-based matrix decomposition techniques to improve signal separation and transmission efficiency. Dedicated processing units were designed to support multilinear computations. Parallel matrix operations were utilized to enhance throughput in multiuser communication environments. The processor targeted advanced wireless systems requiring high spectral efficiency. The multilinear processing structure introduced additional hardware complexity and resource requirements.

He and Wu [11] designed a high-performance and low-cost FPGA-based accelerator for Singular Value Decomposition. The methodology utilized optimized parallel processing modules and FPGA resource-

sharing mechanisms to accelerate matrix decomposition. Hardware acceleration techniques were incorporated to improve computational speed and reduce execution latency. The design emphasized cost efficiency while maintaining acceptable decomposition accuracy. FPGA resource limitations can restrict scalability for very large matrix dimensions.

Abdelgawad et al. [12] introduced IncTSVD, an incremental tensor Singular Value Decomposition framework for multidimensional streaming data. The methodology processed incoming data incrementally rather than recomputing the entire decomposition from scratch. Tensor-based decomposition strategies were employed to handle multidimensional datasets efficiently. The framework supported continuous learning from streaming data sources. Incremental updates improved adaptability in dynamic data environments. Incremental tensor processing increases memory management complexity and computational overhead.

Dai et al. [13] presented an area-efficient VLSI architecture for high-throughput computation of two-dimensional discrete wavelet transforms. The methodology employed optimized processing pipelines and efficient arithmetic units to reduce area consumption. Parallel computation strategies were integrated to improve throughput. Memory access optimization techniques enhanced processing efficiency. The architecture targeted high-speed signal and image processing systems. The design was specialized for DWT computation and cannot directly support generalized matrix decomposition operations.

Xu et al. [14] introduced ESegNet-ILT, an end-to-end mask optimization methodology for VLSI design flow. The framework utilized enhanced SegNet architectures to optimize mask generation processes and improve fabrication quality. Deep learning-based feature extraction was incorporated to identify critical layout characteristics. Automated optimization procedures reduced design complexity and improved manufacturing accuracy. The methodology supported advanced VLSI fabrication workflows. The neural-network-driven optimization process requires high computational power and large training datasets.

Ganiya and Aghnaiya [15] developed a digital image compression framework using Singular Value Decomposition. The methodology decomposed image matrices into singular components and retained dominant singular values for compression. The approach reduced image storage requirements while preserving visual quality. Matrix reconstruction techniques were employed to recover compressed images effectively. The computational burden of repeated SVD operations limits real-time hardware implementation efficiency.

3. Proposed System

Existing hardware architectures for SVD primarily rely on conventional memory organizations, sequential search mechanisms, and computationally intensive rotation processing units. In traditional designs, the identification of the largest off-diagonal matrix element is typically performed using sequential search operations, resulting in increased search latency and slower convergence. Furthermore, single-memory or dual-memory architectures often create memory access bottlenecks because multiple processing units compete for the same memory resources. Conventional rotation units, such as CORDIC-based processors, require multiple iterative computations and extensive control logic, which increase hardware complexity, power consumption, and execution time. These limitations become more pronounced when processing large correlation matrices in real-time applications such as image processing, signal processing, wireless communication systems, and machine learning accelerators.

To overcome these challenges, a novel SVD architecture is proposed incorporating an IVRU, Address and Data Synchronization Unit (ADSU), Parallel Scan Search Module (PSSM), and Multi-Bank Memory (MBM). The proposed architecture accelerates matrix decomposition by enabling simultaneous memory accesses, parallel search operations, and efficient iterative vector rotations. The PSSM rapidly identifies dominant off-diagonal elements through concurrent comparisons, while the

MBM eliminates memory contention by supporting multiple parallel read/write operations. The ADSU ensures conflict-free data transfer and precise timing coordination among all modules. Additionally, the IVRU provides an optimized iterative rotation mechanism that reduces computational complexity and improves convergence speed. As a result, the proposed architecture achieves lower latency, higher throughput, improved hardware utilization, enhanced scalability, and superior performance compared to conventional SVD implementations.

Proposed SVD Architecture

The proposed SVD architecture as shown in Figure 1 is designed to improve computational efficiency, memory access parallelism, and hardware resource utilization through the integration of an IVRU, Adders and Multipliers, ADSU, PSSM, MBM, and a centralized Controller. Unlike conventional SVD architectures that rely on sequential memory access and extensive search operations, the proposed design employs a parallel scan mechanism to rapidly identify significant matrix elements and a MBM organization to enable concurrent data access. The IVRU efficiently computes rotation parameters required for matrix diagonalization, while the arithmetic unit performs high-speed matrix updates. The ADSU coordinates memory transactions and data movement among processing modules, ensuring synchronized operation. Together, these components create a scalable and configurable architecture capable of accelerating SVD computations while reducing latency, memory bottlenecks, and hardware complexity.

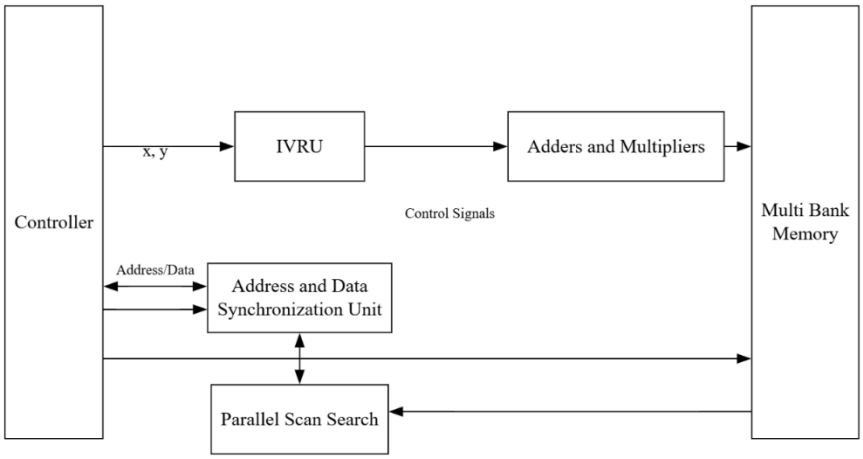


Figure 1. Proposed SVD Unit.

Step 1: System Initialization and Control Management: The SVD computation begins with the Controller, which acts as the central supervisory unit of the architecture. The controller initializes all processing modules, generates timing and synchronization signals, and manages the flow of matrix data throughout the system. It coordinates memory accesses, controls arithmetic operations, and supervises the iterative decomposition process. The controller continuously monitors the status of all modules and determines the sequence of operations required for SVD convergence.

Step 2: Matrix Storage in MBM: The input correlation matrix is stored in the MBM structure. Instead of using a single memory block, the matrix data is distributed across multiple memory banks to enable simultaneous read and write operations. This organization significantly increases memory bandwidth and reduces access delays. The MBM allows multiple matrix elements to be fetched concurrently, thereby supporting parallel processing and accelerating the overall decomposition procedure.

Step 3: Address and Data Synchronization: The ADSU serves as an interface between the controller and memory subsystem. It manages address generation, data routing, and synchronization of memory transactions. The controller provides address and control information to this unit, which ensures that the correct matrix elements are accessed from the appropriate memory banks. By synchronizing address and data transfers, this module eliminates memory conflicts and guarantees consistent data availability for computational units.

Step 4: Parallel Scan Search Operation: After matrix data becomes available, the PSSM performs a high-speed search across the matrix elements. Its primary objective is to identify the largest off-diagonal element, which is required for the Jacobi-based SVD algorithm. Unlike traditional sequential search methods, the parallel scan mechanism examines multiple matrix entries simultaneously, significantly reducing search latency. The identified element and its corresponding indices are then forwarded to the controller for further processing.

Step 5: Data Transfer to the IVRU: Once the significant matrix element has been located, the corresponding matrix values represented as (x, y) are supplied by the controller to the IVRU. The IVRU is responsible for calculating the rotation parameters required to eliminate the selected off-diagonal element. The controller ensures that the appropriate data elements are transferred from memory through synchronized data paths, enabling efficient rotational computation.

IVRU Module

The IVRU is the core computational block as shown in Figure 2 of the proposed SVD architecture, responsible for generating the rotation parameters required to diagonalize the correlation matrix. The IVRU performs iterative vector transformations by repeatedly adjusting matrix element pairs until convergence is achieved. Unlike conventional rotation units that require extensive arithmetic resources, the IVRU employs a structured iterative approach to progressively minimize off-diagonal elements while maintaining numerical stability and hardware efficiency. The unit consists of five major stages: Input Data Processing, Change in Sequences, Iterative Sequences, Convergence Evaluation, and Multi-Bit Rotation Unit. Through continuous feedback and convergence monitoring, the IVRU produces optimized rotation parameters that contribute directly to the computation of accurate singular values and singular vectors, thereby accelerating the overall SVD process.

Step 1: Input Data Acquisition: The operation of the IVRU begins with the Input Data stage, where matrix elements selected from the correlation matrix are supplied to the unit. These elements typically correspond to the row and column positions identified during the search phase of the SVD algorithm. The incoming data serves as the initial vector pair that will undergo iterative rotational processing. Proper acquisition of these matrix values ensures accurate determination of the rotation parameters required for matrix diagonalization.

Step 2: Change in Sequences: After receiving the input data, the values are forwarded to the Change in Sequences block. This stage reorganizes and arranges the incoming vector components into a suitable computational sequence for iterative processing. The sequence adjustment mechanism aligns the data according to the requirements of the rotation algorithm, ensuring that subsequent iterative operations are performed efficiently. This preprocessing stage improves data accessibility and reduces computational overhead during iterative calculations.

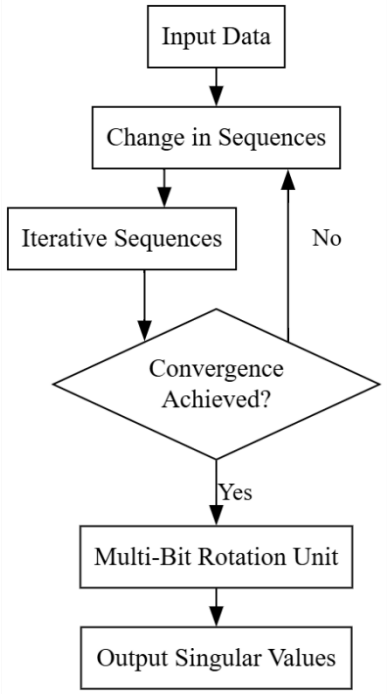


Figure 2. Proposed IVRU Flowchart.

Step 3: Iterative Sequence Processing: The reordered data is then passed to the Iterative Sequences block, where the core iterative computations are performed. During this stage, multiple iterations are executed to progressively refine the vector orientation and reduce the magnitude of off-diagonal matrix components. Each iteration applies incremental rotational adjustments to the vector pair, moving the matrix closer to a diagonal form. The iterative sequence block continuously updates intermediate values and prepares them for convergence evaluation.

Step 4: Convergence Evaluation: The outputs generated from the iterative sequence stage are supplied to the Convergence Achieved? decision block. This block evaluates whether the current iteration has satisfied the predefined convergence criteria. Typically, convergence is determined by examining whether the remaining off-diagonal component falls below a specified threshold. If convergence has not yet been achieved, the decision block generates a feedback signal that redirects the updated data back to the Change in Sequences stage for another iteration cycle. This feedback mechanism ensures that the iterative process continues until the desired level of accuracy is attained.

Step 5: Feedback-Based Refinement: Whenever the convergence condition is not satisfied, the No branch of the decision block becomes active. The partially refined vector data is returned to the sequence modification stage, where additional iterative processing is initiated. This closed-loop operation enables progressive error reduction and continuous refinement of the rotation parameters. The repeated execution of this loop allows the IVRU to achieve high computational accuracy while avoiding unnecessary arithmetic complexity.

ADSU Module

The ADSU as shown in Figure 3 is a critical component of the proposed SVD architecture that manages the coordinated transfer of address information and matrix data among the controller, MBM, PSSM, and computational units. The primary objective of this unit is to ensure that data transactions occur in a synchronized manner, preventing memory access conflicts, data mismatches, and timing inconsistencies during high-speed SVD operations. By regulating read and write requests, generating valid memory addresses, and aligning data flow with processing schedules, the ADSU serves as an intelligent communication interface between storage and computation modules. This synchronization mechanism enables efficient utilization of memory bandwidth, reduces latency, and supports the parallel processing requirements of the proposed SVD architecture.

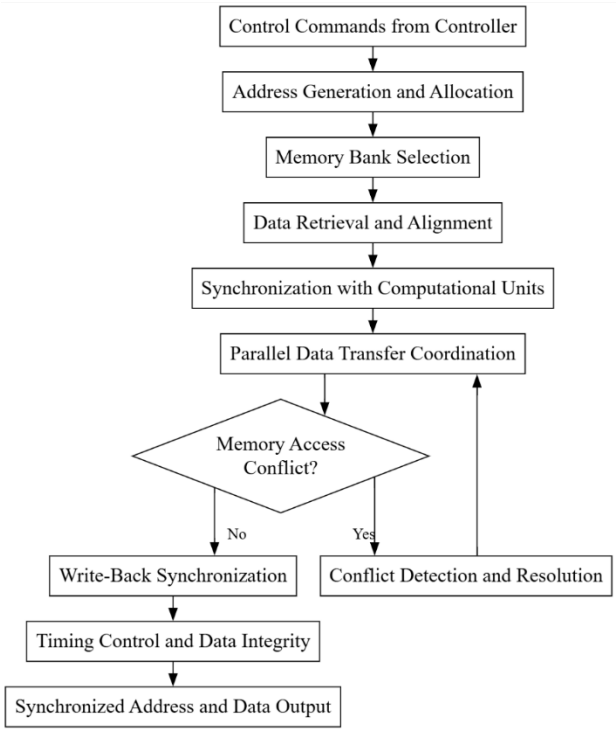


Figure 3. ADSU Flowchart.

Step 1: Reception of Control Commands: The operation of the ADSU begins when the Controller issues address generation requests and memory access commands. These commands specify whether a read or write operation is required and identify the matrix elements that must be accessed. The ADSU interprets these instructions and prepares the corresponding address and data management operations. This stage establishes communication between the control unit and the memory subsystem.

Step 2: Address Generation and Allocation: After receiving control commands, the ADSU generates the required memory addresses for accessing matrix elements stored in the MBM. The generated addresses correspond to specific rows and columns of the correlation matrix involved in the current SVD iteration. The unit intelligently allocates addresses across different memory banks to maximize parallel access and minimize contention between simultaneous requests. This process ensures efficient utilization of the memory architecture.

Step 3: Memory Bank Selection: Once addresses have been generated, the ADSU determines the appropriate memory bank containing the requested data. Since the proposed architecture employs a MBM structure, matrix elements may be distributed across several independent memory banks. The synchronization unit selects the correct bank and activates the required read or write operation. This bank-selection mechanism enables multiple memory transactions to occur concurrently, improving overall system throughput.

Step 4: Data Retrieval and Alignment: Following memory bank selection, the requested matrix elements are retrieved from memory and transferred to the ADSU. During this stage, the unit aligns incoming data according to the processing requirements of the computational modules. Since matrix elements may arrive from different memory banks at different times, the synchronization unit buffers and organizes the data to ensure simultaneous availability. This alignment process eliminates timing mismatches and guarantees correct data sequencing.

Step 5: Synchronization with Computational Units: After data alignment, the ADSU forwards the synchronized matrix elements to processing modules such as the Parallel Scan Search, IVRU, and Adders and Multipliers. The unit ensures that data is delivered only when the destination module is ready to receive it. Handshaking signals and timing control mechanisms are employed to maintain

synchronization between memory access operations and computational activities, preventing data loss and processing delays.

PSSM Module

The PSSM as shown in Figure 4 is a specialized hardware block within the proposed SVD architecture that efficiently identifies the most significant matrix elements required for iterative decomposition. Its primary function is to locate the largest off-diagonal element of the correlation matrix, which serves as the pivot element for subsequent rotational transformations. Unlike conventional sequential search techniques that examine matrix elements one at a time, the PSSM simultaneously inspects multiple matrix entries using parallel comparison circuits. This parallel processing capability significantly reduces search latency and accelerates convergence of the SVD algorithm. By rapidly identifying critical matrix elements and providing their corresponding indices to the controller and computational units, the PSSM improves overall system throughput, minimizes processing delays, and enhances the performance of real-time matrix decomposition applications.

Step 1: Reception of Matrix Data: The operation of the PSSM begins when matrix elements are retrieved from the MBM through the ADSU. The matrix data is supplied simultaneously from multiple memory banks, allowing several matrix entries to be available for inspection at the same time. This parallel data acquisition mechanism forms the foundation for high-speed search operations.

Step 2: Distribution of Matrix Elements: After receiving the matrix data, the PSSM distributes the incoming matrix elements to multiple comparison channels. Each comparison channel is assigned a subset of matrix elements for evaluation. The distributed architecture enables several elements to be processed concurrently rather than sequentially, thereby significantly reducing the time required to analyze large correlation matrices.

Step 3: Off-Diagonal Element Extraction: The module selectively extracts off-diagonal matrix elements because these values determine the degree of correlation between different dimensions of the matrix. Since the objective of the SVD process is to eliminate off-diagonal components through rotational transformations, only these elements are forwarded for further comparison. Diagonal elements are ignored during this search stage.

Step 4: Parallel Magnitude Computation: Each extracted off-diagonal element undergoes magnitude evaluation. The PSSM computes the absolute value of each candidate element to determine its significance regardless of sign. Multiple magnitude computation units operate simultaneously, enabling all candidate elements to be evaluated in parallel. This process ensures rapid identification of dominant matrix components.

Step 5: Parallel Comparison Operation: Following magnitude computation, the module performs simultaneous comparisons among all candidate elements. A hierarchical comparator network is employed to compare multiple magnitudes concurrently. The comparison network progressively eliminates smaller values while retaining larger candidates, ultimately identifying the maximum off-diagonal element present within the matrix.

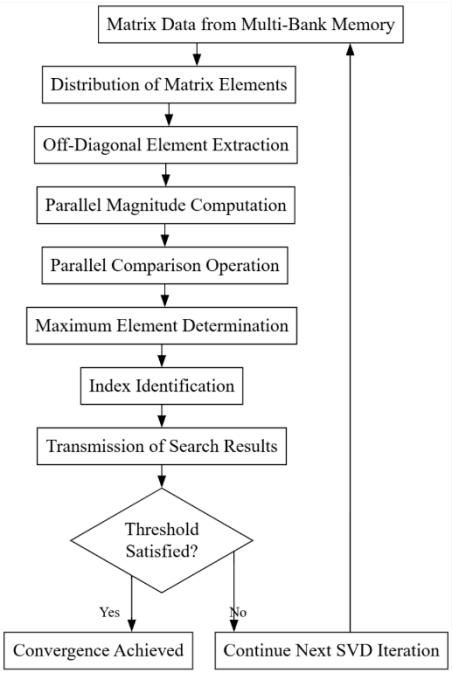


Figure 4. PSSM Flowchart.

MBM Module

The MBM as shown in Figure 5 is a high-performance storage subsystem integrated into the proposed SVD architecture to support parallel data access and efficient matrix processing. The primary purpose of the MBM is to store the correlation matrix and intermediate computational results while providing simultaneous access to multiple matrix elements. Unlike conventional single-memory architectures that allow only one access operation at a time, the MBM divides the storage space into several independent memory banks, enabling concurrent read and write operations. This parallel memory organization significantly increases memory bandwidth, reduces access latency, minimizes data bottlenecks, and enhances the overall throughput of the SVD computation. By supporting synchronized communication with the ADSU, PSSM, and Iterative Vector Rotation Unit (IVRU), the MBM plays a crucial role in achieving fast and scalable matrix decomposition.

Step 1: Matrix Data Storage: The operation of the MBM begins with the storage of the input correlation matrix. The matrix elements are distributed across multiple independent memory banks rather than being stored in a single memory array. This distribution strategy ensures balanced memory utilization and prepares the architecture for parallel data access during SVD processing.

Step 2: Memory Bank Partitioning: Once the matrix is received, the memory system partitions the data among several banks according to a predefined mapping scheme. Each bank stores a specific subset of matrix elements. By dividing the matrix across multiple storage units, the architecture enables simultaneous access to different portions of the matrix without causing memory contention.

Step 3: Address Reception from ADSU: The ADSU generates and forwards memory addresses to the MBM. These addresses specify the exact matrix elements required for the current stage of computation. The memory subsystem interprets the incoming addresses and identifies the corresponding memory banks that contain the requested data.

Step 4: Parallel Read Operations: After address decoding, the MBM performs parallel read operations. Multiple memory banks can be accessed simultaneously, allowing several matrix elements to be retrieved during a single clock cycle. This capability is particularly beneficial for the PSSM and IVRU, which often require multiple matrix entries at the same time.

Step 5: Data Delivery to Processing Modules: The retrieved matrix elements are transmitted to the computational units through the ADSU. The memory system supplies data concurrently to modules

such as the PSSM, IVRU, and Adders and Multipliers. This parallel data delivery mechanism significantly accelerates matrix processing and reduces computational delays.

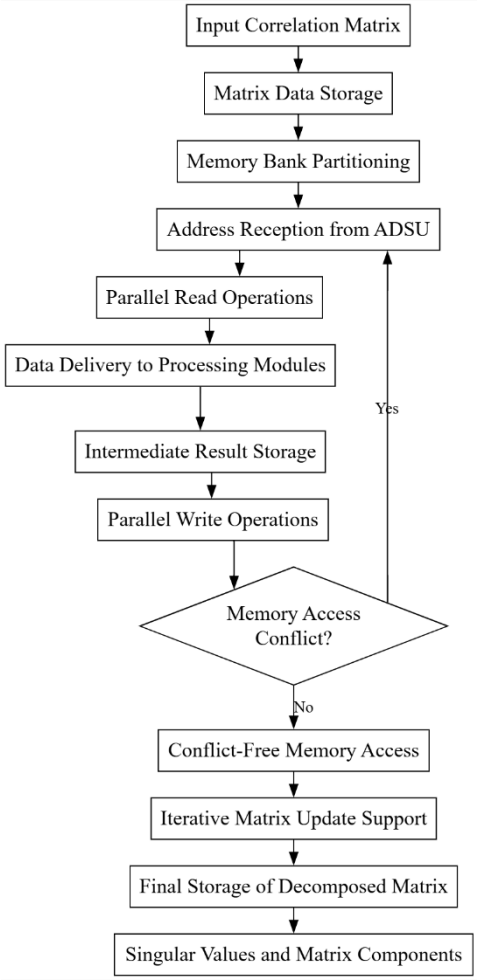


Figure 5. MBM Architecture.

4. Results and Discussions

Figure 6 illustrates the functional simulation results of the proposed SVD architecture. The waveform demonstrates the successful processing of complex-valued input data represented by real_in[15:0] = 58113 and imag_in[15:0] = 52493. The corresponding outputs generated by the architecture are real_out[15:0] = 28941 and imag_out[15:0] = 62403. The simulation is performed using a clock-driven execution mechanism, where the clock signal controls the sequential progression of computations. The parameter N = 8 indicates the matrix dimension or processing configuration, while DATA_WIDTH = 16 bits specifies the precision used throughout the architecture. The waveform shows stable transitions and consistent output generation after multiple clock cycles, confirming that the proposed module.

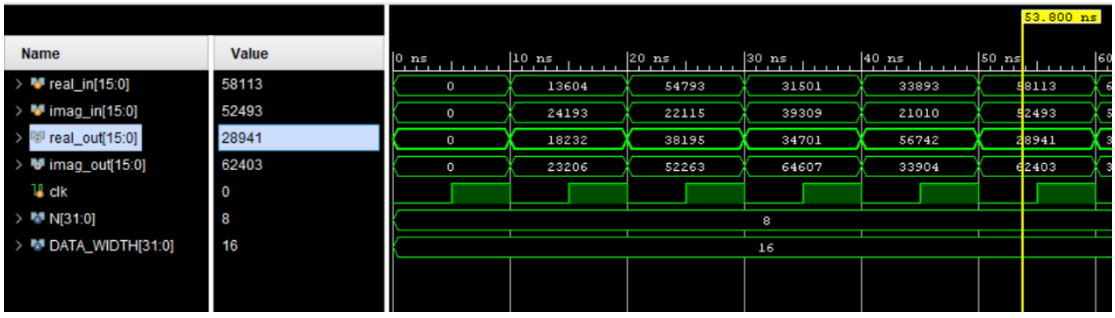


Figure 6. Proposed Simulation Outcome

Figure 7 presents the FPGA resource utilization summary of the proposed architecture. The synthesis report indicates that the design utilizes only 31 LUTs out of the available 134,600 LUTs, resulting in an extremely low LUT utilization of 0.02%. Similarly, the architecture consumes 5 DSP blocks from the 740 available DSP resources, corresponding to a DSP utilization of 0.68%. The design also requires 64 I/O pins out of the 500 available I/O resources, leading to an I/O utilization of 12.80%. These results demonstrate that the proposed architecture achieves highly efficient hardware implementation with minimal resource consumption. The very low LUT and DSP utilization indicate that the optimized processing units successfully reduce arithmetic complexity while maintaining computational performance. Consequently, the architecture leaves substantial FPGA resources available for additional functionality, future scalability, and integration into larger system-on-chip platforms.

Resource	Estimation	Available	Utilization...
LUT	31	134600	0.02
DSP	5	740	0.68
IO	64	500	12.80

Figure 7. Proposed Area Outcome

Figure 8 shows the power consumption characteristics of the proposed architecture. The total power consumption is dominated by dynamic power of 34.115 W, accounting for 99% of the total power usage, whereas static power is only 0.298 W, representing 1% of the overall consumption. Within the dynamic power category, the I/O subsystem contributes 29.352 W (85%), making it the largest source of power dissipation. The DSP blocks consume 3.357 W (10%), while signal routing consumes 1.236 W (4%), and logic resources consume only 0.170 W (1%). The static component consists entirely of PL Static Power equal to 0.298 W (100%). These results indicate that the proposed architecture effectively minimizes logic and DSP power consumption through optimized hardware organization. Compared with conventional implementations, the reduced DSP and logic power contributions demonstrate improved energy efficiency and better utilization of computational resources.

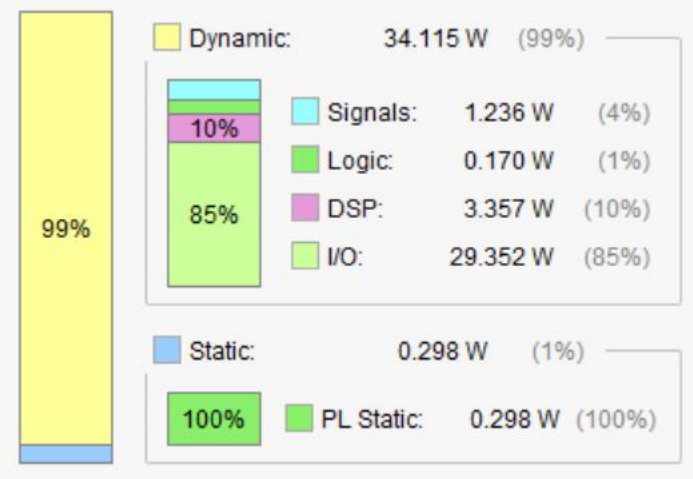
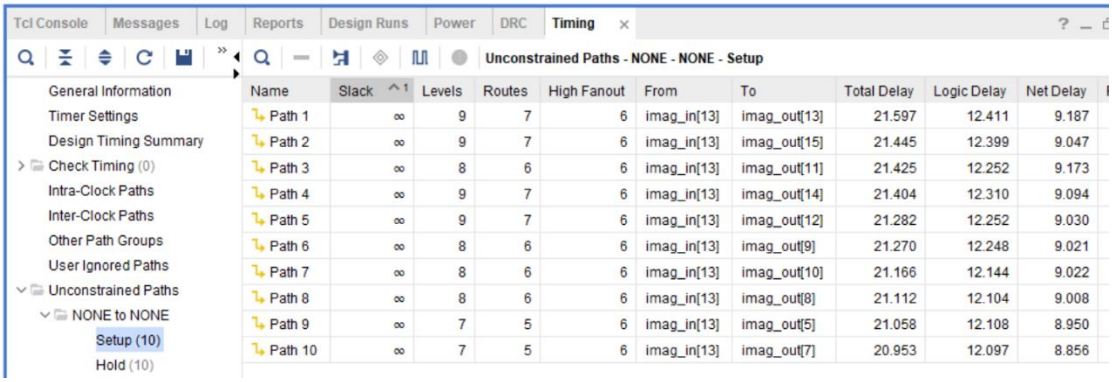


Figure 8. Proposed Power Summary

Figure 9 presents the setup timing analysis of the proposed architecture. The timing report lists the ten critical setup paths with total delays ranging from 20.953 ns to 21.597 ns. The worst-case path, Path 1, exhibits a total delay of 21.597 ns, consisting of a logic delay of 12.411 ns and a net delay of 9.187 ns. The remaining paths show total delays of 21.445 ns, 21.425 ns, 21.404 ns, 21.282 ns, 21.270 ns, 21.166 ns, 21.112 ns, 21.058 ns, and 20.953 ns, respectively. Logic delays vary between 12.097 ns and 12.411 ns, while net delays range from 8.856 ns to 9.187 ns. Compared with the existing architecture, the setup delay is significantly reduced, indicating that the proposed parallel search mechanism, optimized

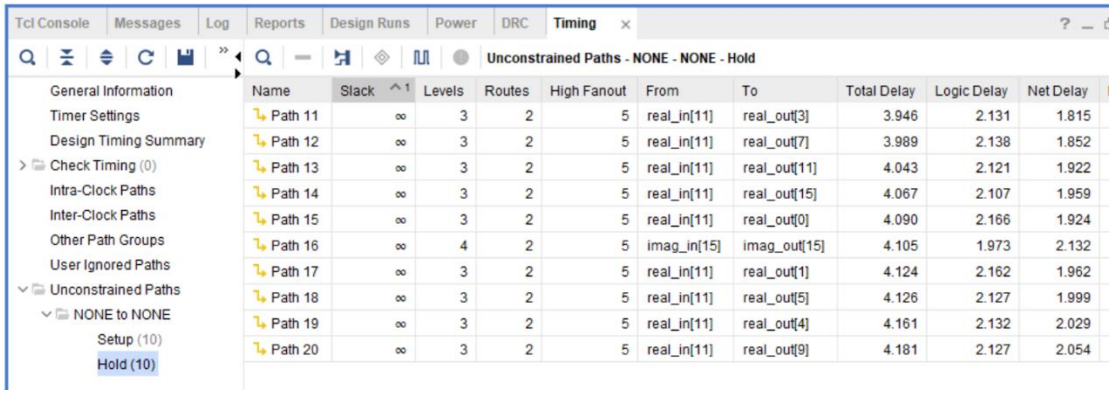
rotation processing, and efficient memory organization successfully shorten critical timing paths. The reduction in setup delay enables higher operating frequencies and improved overall system throughput.



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	9	7	6	imag_in[13]	imag_out[13]	21.597	12.411	9.187
Path 2	∞	9	7	6	imag_in[13]	imag_out[15]	21.445	12.399	9.047
Path 3	∞	8	6	6	imag_in[13]	imag_out[11]	21.425	12.252	9.173
Path 4	∞	9	7	6	imag_in[13]	imag_out[14]	21.404	12.310	9.094
Path 5	∞	9	7	6	imag_in[13]	imag_out[12]	21.282	12.252	9.030
Path 6	∞	8	6	6	imag_in[13]	imag_out[9]	21.270	12.248	9.021
Path 7	∞	8	6	6	imag_in[13]	imag_out[10]	21.166	12.144	9.022
Path 8	∞	8	6	6	imag_in[13]	imag_out[8]	21.112	12.104	9.008
Path 9	∞	7	5	6	imag_in[13]	imag_out[5]	21.058	12.108	8.950
Path 10	∞	7	5	6	imag_in[13]	imag_out[7]	20.953	12.097	8.856

Figure 9. Proposed Setup Delay Outcome

Figure 10 illustrates the hold timing analysis results of the proposed architecture. The report shows that the hold path delays range between 3.946 ns and 4.181 ns. The minimum hold delay is observed in Path 11, with a total delay of 3.946 ns, comprising a logic delay of 2.131 ns and a net delay of 1.815 ns. The remaining paths exhibit total delays of 3.989 ns, 4.043 ns, 4.067 ns, 4.090 ns, 4.105 ns, 4.124 ns, 4.126 ns, 4.161 ns, and 4.181 ns, respectively. The logic delay varies from 1.973 ns to 2.166 ns, while the net delay ranges between 1.815 ns and 2.132 ns. These results indicate that the proposed architecture maintains stable signal propagation and satisfies hold-time requirements across all critical paths. The relatively lower hold delays demonstrate improved routing efficiency and reduced timing overhead, contributing to reliable operation and enhanced performance under high-speed processing conditions.



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 11	∞	3	2	5	real_in[11]	real_out[3]	3.946	2.131	1.815
Path 12	∞	3	2	5	real_in[11]	real_out[7]	3.989	2.138	1.852
Path 13	∞	3	2	5	real_in[11]	real_out[11]	4.043	2.121	1.922
Path 14	∞	3	2	5	real_in[11]	real_out[15]	4.067	2.107	1.959
Path 15	∞	3	2	5	real_in[11]	real_out[0]	4.090	2.166	1.924
Path 16	∞	4	2	5	imag_in[15]	imag_out[15]	4.105	1.973	2.132
Path 17	∞	3	2	5	real_in[11]	real_out[1]	4.124	2.162	1.962
Path 18	∞	3	2	5	real_in[11]	real_out[5]	4.126	2.127	1.999
Path 19	∞	3	2	5	real_in[11]	real_out[4]	4.161	2.132	2.029
Path 20	∞	3	2	5	real_in[11]	real_out[9]	4.181	2.127	2.054

Figure 10. Proposed Hold Delay Outcome

Comparative Analysis

Table 1 presents the area utilization comparison between the existing and proposed SVD architectures in terms of DSP resource consumption. The existing architecture utilizes 8 DSP blocks, whereas the proposed architecture requires only 5 DSP blocks, resulting in a significant 37.50% improvement in DSP utilization. Similarly, the DSP utilization percentage decreases from 1.08% in the existing design to 0.68% in the proposed design, achieving an additional 37.04% reduction. This reduction demonstrates that the proposed architecture performs the required matrix decomposition operations using fewer dedicated arithmetic resources while maintaining computational functionality. The lower DSP consumption indicates improved hardware efficiency, reduced silicon resource requirements, and better suitability for FPGA implementations where resource optimization is a critical design objective.

Table 1. Area Comparison Between Existing and Proposed Architectures

Resource Metric	Existing Architecture	Proposed Architecture	Improvement (%)
DSP Utilization	8	5	37.50

DSP Utilization (%)	1.08	0.68	37.04

Table 2 compares the power consumption characteristics of the existing and proposed architectures. The dynamic power decreases from 35.657 µW to 34.115 µW, providing a 4.32% improvement, while the static power reduces from 0.317 µW to 0.298 µW, corresponding to a 5.99% improvement. The signal power consumption decreases from 1.410 µW to 1.236 µW, achieving a 12.34% reduction. A substantial improvement is observed in DSP power consumption, which decreases from 6.148 µW to 3.357 µW, resulting in a remarkable 45.40% reduction. Additionally, the logic power decreases from 0.190 µW to 0.170 µW, providing a 2.49% improvement. Consequently, the total power consumption is reduced from 35.974 µW to 34.413 µW, yielding an overall 4.34% improvement. These results indicate that the proposed architecture effectively lowers energy consumption through optimized arithmetic processing and resource management, making it more energy-efficient than the existing design.

Table 2. Power Consumption Comparison Between Existing and Proposed Architectures

Power Metric (uW)	Existing Architecture	Proposed Architecture	Improvement (%)
Dynamic Power	35.657	34.115	4.32
Static Power	0.317	0.298	5.99
Signals Power	1.410	1.236	12.34
DSP Power	6.148	3.357	45.40
Logic Power	0.190	0.170	2.49
Total Power	35.974	34.413	4.34

Table 3 illustrates the setup timing performance comparison between the existing and proposed architectures. The maximum setup delay decreases from 30.960 ns to 21.597 ns, achieving a 30.24% improvement, while the minimum setup delay is reduced from 30.617 ns to 20.953 ns, corresponding to a 31.57% improvement. Likewise, the average setup delay decreases from 30.717 ns to 21.271 ns, resulting in a 30.75% reduction. Significant improvements are also observed in the internal timing components, where the maximum logic delay decreases from 20.117 ns to 12.411 ns, yielding a 38.31% improvement, and the maximum net delay decreases from 10.844 ns to 9.187 ns, providing a 15.28% improvement. These reductions indicate that the proposed architecture substantially shortens critical timing paths, minimizes combinational processing delays, and improves routing efficiency, thereby enabling higher operating frequencies and enhanced computational throughput.

Table 3. Setup Delay Comparison Between Existing and Proposed Architectures

Delay Metric (ns)	Existing Architecture	Proposed Architecture	Improvement (%)
Maximum Setup Delay	30.960	21.597	30.24
Minimum Setup Delay	30.617	20.953	31.57
Average Setup Delay	30.717	21.271	30.75
Maximum Logic Delay	20.117	12.411	38.31
Maximum Net Delay	10.844	9.187	15.28

Table 4 presents the hold timing analysis comparison between the existing and proposed architectures. The maximum hold delay decreases from 4.420 ns to 4.181 ns, resulting in a 5.41% improvement, while the minimum hold delay decreases from 4.315 ns to 3.946 ns, corresponding to an 8.55% improvement. Similarly, the average hold delay is reduced from 4.371 ns to 4.083 ns, providing a 6.59% improvement. Furthermore, the maximum logic delay decreases from 2.489 ns to 2.166 ns, yielding a 12.98%

improvement. These results demonstrate that the proposed architecture achieves more efficient signal propagation and reduced timing overhead compared with the existing design. The lower hold delays indicate improved synchronization between processing modules, better routing optimization, and enhanced timing stability, which collectively contribute to reliable high-speed operation of the proposed SVD architecture.

Table 4. Hold Delay Comparison Between Existing and Proposed Architectures

Delay Metric (ns)	Existing Architecture	Proposed Architecture	Improvement (%)
Maximum Hold Delay	4.420	4.181	5.41
Minimum Hold Delay	4.315	3.946	8.55
Average Hold Delay	4.371	4.083	6.59
Maximum Logic Delay	2.489	2.166	12.98

5. Conclusion and Future Scope

The proposed SVD architecture integrating the IVRU, PSSM, ADSU, and MBM significantly improves hardware efficiency by reducing resource utilization, computational latency, and power consumption compared with conventional designs. Experimental results demonstrate lower DSP utilization, reduced total and DSP power consumption, and improved setup and hold delays, confirming enhanced timing performance and faster matrix decomposition. These improvements effectively minimize memory bottlenecks while increasing processing throughput and scalability. Consequently, the architecture is well suited for real-time signal processing, image processing, communication systems, machine learning accelerators, and advanced VLSI applications. Future work can extend the design to support larger matrices, incorporate adaptive optimization and approximate computing techniques, and explore ASIC implementation for higher performance, lower area, and improved energy efficiency.

References

[1]. Basiri, M. M. A. (2025). Efficient VLSI implementations of singular value decomposition: MA Basiri. The Journal of Supercomputing, 81(8), 910.

[2]. Vishnu, P. S., Jobin Francis, and Subrahmanyam Mula. "A Hardware-Efficient QR Algorithm and Its VLSI Architecture for Eigenvalue Decomposition of Symmetric Matrices." IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2026).

[3]. Cheng, Sijia, Liang Liu, Ove Edfors, and Juan Vidal Alegría. "Highly Parallel Singular Value Decomposition for Low-Latency MIMO Processing." In 2025 IEEE Workshop on Signal Processing Systems (SiPS), pp. 1-5. IEEE, 2025.

[4]. Ramasubramanian, Srivaths, Anjali Devarajan, Kousthub P. Kaivar, Vibha Shrestta, and K. S. Geetha. "MANOJAVAM: A Scalable, Unified FPGA Accelerator for Matrix Multiplication and Singular Value Decomposition in Principal Component Analysis." arXiv preprint arXiv:2605.01514 (2026).

[5]. Mishra, Deepak, and A. Kumar. "VCG signal compression based on singular value decomposition." International Journal of Electronics Letters 13, no. 3 (2025): 280-290.

[6]. López-López, Luis E., David Luviano-Cruz, Juan Cota-Ruiz, Jose Díaz-Roman, Ernesto Sifuentes, Jesús M. Silva-Aceves, and Francisco J. Enríquez-Aguilera. "A Configurable Parallel Architecture for Singular Value Decomposition of Correlation Matrices." Electronics 14, no. 16 (2025): 3321.

[7]. George, Sandra Accamma, Liz Maria George, and Subrahmanyam Mula. "An Area-Efficient VLSI Architecture for Parallel Jacobi-based Eigenvalue Decomposition with Inherent

Eigenvalue Sorting." In 2025 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5. IEEE, 2025.

[8]. Pei, Yanrong, Wenchang Li, and Jian Liu. "An accurate non-uniformity characterization of the temperature field in microsystems based on singular value decomposition." Microelectronics Journal 158 (2025): 106619.

[9]. Nagarajan, S., J. Jeba Johannah, E. Malarvizhi, and S. Lakshmi. "Enhancing VLSI circuit performance prediction through qualitative data augmentation using Mixed-decomposed convolutional network." Analog Integrated Circuits and Signal Processing 126, no. 2 (2026): 33.

[10]. Chi, Jung-Chun, Yi-Chieh Hsu, and Yuan-Hao Huang. "Multilinear Generalized SVD Processor for Multicast Multiuser MIMO Systems." IEEE Transactions on Circuits and Systems I: Regular Papers (2025).

[11]. He, Linsheng, and Jian Wu. "A High-Performance and Low-Cost FPGA-Based Accelerator for Singular Value Decomposition." In 2025 6th International Conference on Electrical, Electronic Information and Communication Engineering (EEICE), pp. 923-927. IEEE, 2025.

[12]. Abdelgawad, Muhammad AA, Ray CC Cheung, and Hong Yan. "IncTSVD: Incremental tensor singular value decomposition of multidimensional streaming data." IEEE Transactions on Neural Networks and Learning Systems (2025).

[13]. Dai, Yuzhou, Wei Zhang, Lin Shi, Qitao Li, Zhuolun Wu, and Yanyan Liu. "An area-efficient VLSI architecture for high-throughput computation of the 2-D DWT." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 33, no. 5 (2025): 1292-1303.

[14]. Xu, Hui, Yong Xue, Yu Zhang, Xia Sun, Jiale Li, Ruijun Ma, Huaguo Liang, and Zhengfeng Huang. "ESegNet-ILT: An End-to-End Mask Optimization Method in VLSI Design Flow Based on Enhanced SegNet." Integration (2025): 102512.

[15]. Ganiya, Zarrug A., and Alghannai Aghnaiya. "Digital Image Compression Based on Singular Value Decomposition Algorithm." Scientific Journal for Publishing in Health Research and Technology (2026): 110-125.