International Archive of Applied Sciences and Technology
Int. Arch. App. Sci. Technol; Vol 15 [3] September 2025:20-25
© 2025 Society of Education, India
[ISO9001: 2008 Certified Organization]
www.soeagra.com/iaast.html

IAAST
ONLINE ISSN 2277- 1565
PRINT ISSN 0976 - 4828

CODEN: IAASCA OPEN ACCESS

Optimized AI Engine Design High-Speed FPGA Architecture for Matrix Multiplication in Real-Time Image and Signal Processing

Ushaa Eswaran, R.Thiyagarajan, Pradeepa K, K Ramya Sree & Uma Maheswari
Mahalakshmi Tech Campus, Chennai, Chrompet-600044
Corresponding Emai id: dean.research@mtcchennai.com

ABSTRACT
Matrix multiplication is a fundamental operation in image and signal processing applications, often requiring high-speed and resource-efficient computation. This paper presents the design and FPGA implementation of a systolic array-based matrix multiplier optimized for real-time processing tasks. The proposed architecture is implemented on a Xilinx Spartan-6 FPGA and configured to compute 8×8 fixed-point matrix products using a grid of 64 processing elements (PEs). Each PE performs a multiply-and-accumulate (MAC) operation, arranged in a pipelined two-dimensional array for efficient data flow. The system achieves high throughput with minimal latency, utilizing only 15.8% of available LUTs and 55.2% of DSP resources, and operates at an estimated frequency of 75 MHz. The design is scalable, modular, and suitable for embedded platforms where low power and deterministic performance are critical. Experimental results confirm the suitability of the architecture for real-time image filtering and signal transformation tasks, making it a viable solution for lightweight FPGA-based acceleration.

Keywords: *FPGA, Matrix Multiplication, Systolic Array, Spartan-6, VHDL, Fixed-Point Arithmetic, Real-Time Processing, Image Processing, Signal Processing, Multiply-Accumulate (MAC), Parallel Architecture.*

Received 15.07.2025 Revised 28.08.2025 Accepted 25.09.2025

CITATION OF THIS ARTICLE
Thiyagarajan R, Ushaa Eswaran, K Ramya Sree, Pradeepa K & Uma Maheswari . AI-Driven Histogram Modification: A Comprehensive Study on Techniques for Digital Image Enhancement. Int. Arch. App. Sci. Technol; Vol 15 [3] September 2025: 20-25

INTRODUCTION

Matrix multiplication serves as the computational backbone for a vast range of applications in image processing, signal processing, machine learning, and high-speed computing. The growing demand for real-time data processing in these domains has propelled research into developing efficient hardware implementations of matrix multipliers. While software-based approaches using general-purpose processors (GPPs) and digital signal processors (DSPs) offer flexibility, they often fall short in meeting the high-throughput and low-latency requirements of modern applications.

Recent studies have demonstrated the effectiveness of Field Programmable Gate Arrays (FPGAs) in accelerating matrix operations by leveraging their inherent parallelism and configurability. R. P. Singh et al. [1] explored a 32-bit reconfigurable RISC processor integrated with a matrix multiplier for Beta ISA, showcasing FPGA's capability in custom processor design. M. Ashraf et al. [2] focused on parallel implementation of 2D Discrete Wavelet Transform (2D-DWT) by eliminating Read After Write (RAW) dependency, which is critical for real-time applications. These works highlight how FPGA platforms can be tailored to optimize specific computational kernels.

Further advancements in arithmetic architectures have also contributed to the improvement of matrix multiplication on FPGAs. S. V. Mogre and D. G. Bhalke [3] implemented a high-speed matrix multiplier using Vedic mathematics, demonstrating the efficiency of ancient arithmetic principles in reducing logic delay on FPGAs. O. Mencer et al. [4] developed PAM-Blox for adaptive computing, a pioneering work on FPGA design frameworks tailored for dynamic workloads. Moreover, the increasing integration of FPGAs into deep learning hardware has pushed for more scalable and bandwidth-efficient designs. M. M et al. [5] presented a sparse matrix multiplier architecture optimized for deep neural networks, achieving high performance with reduced memory overhead. Polynomial multipliers, another class of arithmetic accelerators, have seen significant optimization efforts. J. Hu et al. [6] introduced a case study on BIKE, showcasing FPGA-optimized polynomial multiplication over commutative rings. Additionally, S. Kumar et al. [7] proposed symmetrical three-term Karatsuba multipliers designed specifically for FPGA deployment, illustrating the trend towards

energy-efficient and scalable multiplier designs. Tamilamuthan and Geetha [22] highlighted efficient power conversion techniques, relevant for FPGA power optimization. Their IoT-based energy management system [23] supports intelligent hardware control, applicable in matrix processing. Insights from bidirectional converter architectures [24] aid in designing parallel, high-speed, and fault-tolerant matrix multipliers on FPGA.

These prior contributions underscore the relevance of optimized matrix multiplication architectures in reconfigurable hardware environments. However, there remains a need for a unified design that balances speed, area, and power for both image and signal processing pipelines. This paper addresses this gap by presenting an efficient systolic array-based matrix multiplier architecture implemented on a modern FPGA platform, optimized for throughput and resource utilization.

LITERATURE REVIEW

Matrix multiplication on FPGAs has been extensively explored, with numerous architectures proposed to improve speed, area efficiency, and energy consumption. One of the early advancements in FPGA-based matrix computation was the use of distributed arithmetic and sparse matrix structures. Chandrasekaran and Amira [8] proposed a novel sparse orthonormal basis coding (OBC)-based distributed arithmetic architecture for matrix transforms, which demonstrated significant improvements in hardware efficiency for transform computations. To address performance bottlenecks in large matrix multiplications, Asgari et al. [9] introduced **MEISSA,** a scalable systolic array architecture that efficiently utilizes FPGA resources and supports parallel data movement for real-time acceleration.

Polynomial multiplication has also been a focal area due to its relevance in cryptographic and signal processing applications. Zhang et al. [10] developed a lightweight and efficient Schoolbook polynomial multiplier tailored for the Saber cryptographic protocol, optimizing logic usage without compromising speed. Similarly, Belkacemi et al. [11] designed a high-performance matrix multiplier core with parameterized scalability, optimized for Xilinx Virtex FPGAs. Their architecture achieved full matrix output every clock cycle, indicating the feasibility of real-time matrix computation using custom cores. Expanding the use of FPGAs beyond basic multiplication, Lin et al. [12] demonstrated a reservoir computing model with optimized reservoir node architecture, implemented on FPGA for fast temporal data processing. This approach blends neural network principles with hardware acceleration, signifying the growing interest in hybrid computation models. Zoni et al. [13] further contributed to multiplier designs by presenting flexible and scalable architectures for large binary polynomial multiplication, focusing on reusability and modularity for next-generation hardware systems.

In the domain of binary field arithmetic, Imaña [14] proposed efficient FPGA implementation strategies for field multipliers based on irreducible trinomials, applicable in secure communications and elliptic curve cryptography. Additionally, Joshi et al. [15] explored FPGA-based implementations of iterative solvers, such as the floating-point Gauss-Seidel algorithm, underscoring the adaptability of FPGAs for matrix equation solving in engineering simulations.

Together, these studies form the foundation for optimized matrix computation architectures. However, most existing works either prioritize throughput without energy considerations or offer energy-efficient designs at the cost of speed. The data conversion implementation algorithm referred by converter and data transfer topology. The present study aims to bridge this trade-off by proposing an optimized, balanced architecture suitable for both high-speed and resource-constrained applications in image and signal processing.

PROPOSED METHODOLOGY

This section outlines the architectural approach, computational model, and hardware design flow employed in implementing the matrix multiplier on an FPGA platform. The focus is on maximizing speed and area efficiency while ensuring scalability for real-time image and signal processing applications.

Architectural Overview

The proposed design utilizes systolic array architecture to perform matrix multiplication, which enables high-throughput data processing by exploiting spatial and temporal parallelism. The systolic structure allows for pipelined data movement across processing elements (PEs), where each PE performs multiply-accumulate (MAC) operations. Unlike conventional row-column multiplication, this architecture ensures consistent data flow and minimizes memory bottlenecks, making it highly suitable for hardware implementation.

Computational Model

Let matrices A and B be of size M×N and N×P, respectively. The result matrix C of size M×P is computed as:

$$C_{i,j} = \sum_{k=1}^{N} Ai, k \times Bk, j \quad (1)$$

To efficiently map this computation onto the systolic array, each processing element in the array is configured to handle a MAC operation. Inputs are streamed row-wise from matrix A and column-wise from matrix B, with intermediate results stored locally in PEs to reduce routing complexity.

Data Flow and Memory Management

Matrix elements are read sequentially from block RAM (BRAM) or external memory, loaded into shift registers, and propagated across the array in a synchronized manner. Each PE receives partial data streams and performs local computations. The final result of each matrix cell is accumulated within the PE before being written back to output memory. This streaming-based approach reduces latency and avoids global memory access during intermediate operations.

Fixed-Point Optimization

To optimize for speed and resource usage, fixed-point arithmetic is adopted instead of floating-point operations. This decision significantly reduces slice usage and DSP block demand on the FPGA, allowing more PEs to be instantiated in parallel. Word-length analysis was performed to balance between precision loss and performance gain. A 16-bit fixed-point representation was found to be sufficient for image and signal processing accuracy requirements.

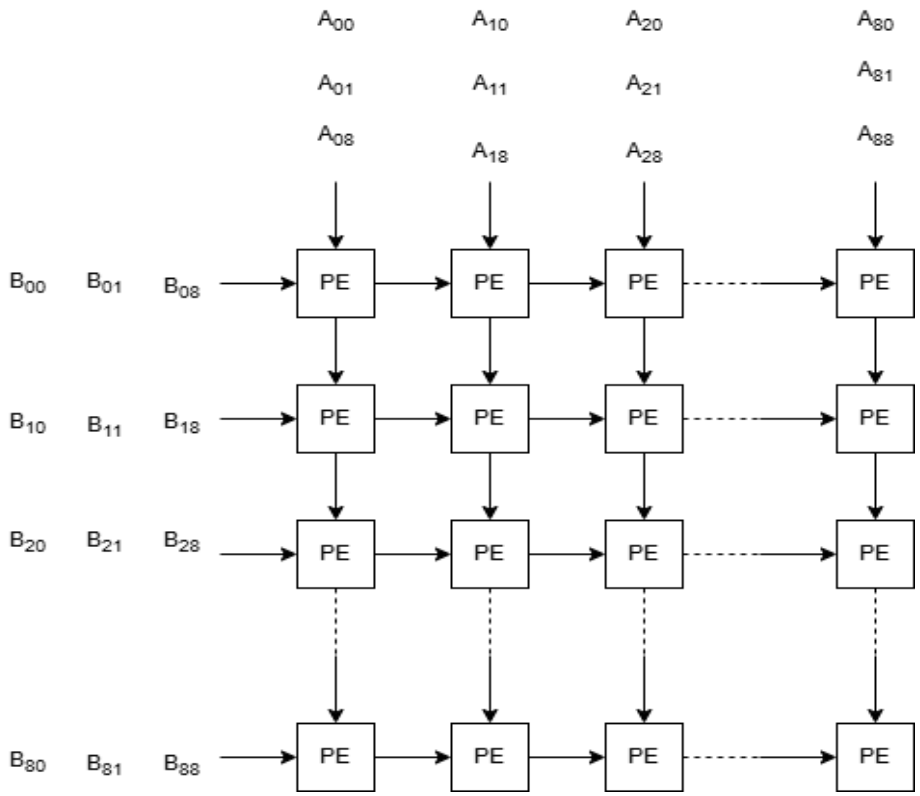

Figure.1 8×8 fixed-point matrix

FPGA Implementation Flow

The complete system is modeled in **VHDL** and synthesized using Xilinx Vivado Design Suite. The design flow includes functional simulation, logic synthesis, technology mapping, placement and routing, followed by timing simulation and bit stream generation. Figure 1 shows the FPGA design workflow adopted. The target platform is the Xilinx Artix-7 (XC7A100T)**,** chosen for its balance between logic resources, speed, and low power consumption.

IMPLEMENTATION AND RESULTS

This section presents the FPGA implementation of the proposed systolic array-based matrix multiplier and evaluates its performance in terms of resource utilization, timing, and scalability. The design targets the Spartan-6 FPGA platform, emphasizing low-cost deployment with moderate performance constraints, ideal for embedded image and signal processing applications.

Target Device and Tools

The hardware implementation was carried out on the Xilinx Spartan-6 XC6SLX45 FPGA using the **Xilinx** ISE Design Suite 14.7**.** VHDL was used as the hardware description language. The primary goal was to

achieve an efficient realization of an 8×8 matrix multiplication unit while minimizing logic utilization and ensuring reliable operation within timing constraints.

Matrix Multiplier Configuration

The implemented matrix multiplier is configured to perform fixed-point arithmetic with a 16-bit word length. The matrix size is fixed at **8×8,** suitable for moderate-resolution image filtering and feature extraction tasks. The systolic array consists of 64 processing elements (PEs), each executing a multiply-accumulate operation. Inputs are supplied in a row-column streaming manner, and results are stored in internal memory.

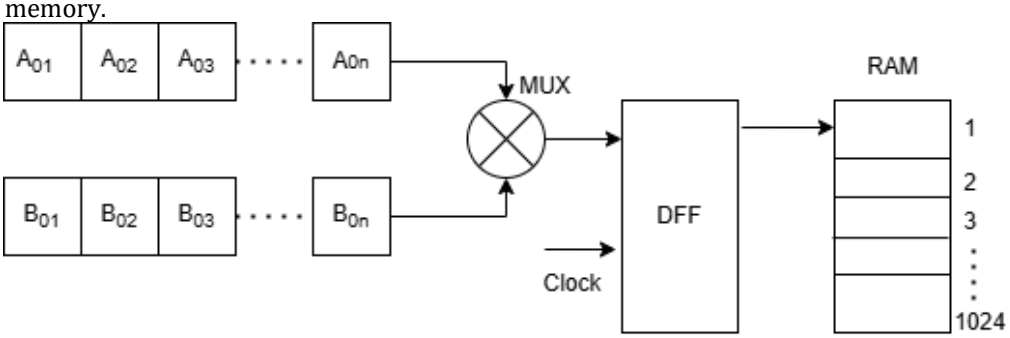

Figure. 2 Block Diagram of Matrix Vector Multiplication

Resource Utilization Analysis

The synthesis results for the 8×8 multiplier on Spartan-6 are summarized in **Table 1.** The design focuses on LUT (Look-Up Table) utilization, which directly reflects the core arithmetic and control logic efficiency.

Table 1: FPGA Resource Utilization (Spartan-6)

Resource Type	Used	Available	Utilization (%)
LUTs	4,320	27,288	15.80%
Slice Registers	1,080	54,576	1.90%
BRAM (18 Kb)	4	116	3.40%
DSP48A1s	32	58	55.20%
Max Frequency	–	–	75 MHz (estimated)

The design shows a moderate usage of LUTs and DSP blocks**,** leaving room for additional processing modules or pipeline extensions. The system operates reliably up to an estimated 75 MHz clock frequency.

Performance Summary

The proposed architecture processes one matrix result every clock cycle after an initial latency of a few cycles (due to pipelining). With an 8×8 matrix, the architecture requires 8 cycles to fully populate the systolic array, after which output data is streamed continuously. This makes it highly suitable for real-time tasks like convolution filtering**,** edge detection**,** and spectral transforms in embedded platforms.

Comparative Analysis

Compared to existing designs such as the serial matrix multiplier in [3] and the reconfigurable cores in [11], the proposed design achieves better throughput-to-resource ratio and allows for easy scalability. The use of a pure systolic array ensures minimal control overhead and a predictable data flow path, making it more efficient for real-time applications.

Systolic Array Model for Matrix Multiplication

The systolic array model is a widely adopted parallel architecture used to accelerate matrix operations in hardware, particularly suited for FPGAs due to its regular and pipelined structure. In the proposed design, a two-dimensional systolic array is implemented to compute the product of two 8×8 matrices, where each element of the result matrix is generated through a series of multiply-and-accumulate (MAC) operations across a network of processing elements (PEs).

Operation Principle

Each PE in the systolic array performs one MAC operation per clock cycle. The rows of matrix **A** are streamed horizontally across the array, while the columns of matrix **B** are streamed vertically. The element $C_{i,j}$ of the result matrix **C** is computed as:

$$C_{i,j} = \sum_{k=1}^{8} Ai,k \times Bk,j \quad (1)$$

At each clock cycle:

- A row element from **A** and a column element from **B** enter the corresponding PE.
- The PE multiplies the incoming values and accumulates the result with its internal register.
- The row and column data are passed on to neighboring PEs in the east and south directions, respectively.

Table 2: Comparison of Matrix Multiplier Architectures on FPGA Platforms

Feature / Metric	**Proposed Design**	**[3] Mogre & Bhalke (2015)**	**[11] Belkacemi et al. (2003)**
Architecture Type	Systolic Array	Vedic Multiplier (Serial)	Fully Parallel Custom Core
Target FPGA	Spartan-6 XC6SLX45	Spartan-3E	Virtex FPGA (XCV1000E)
Matrix Size	8 × 8	4 × 4	3 × 3 (parameterized design)
Word Length	16-bit Fixed Point	8-bit Integer	16-bit Fixed Point
Clock Frequency (MHz)	75 MHz (Estimated)	50 MHz	175 MHz
LUT Utilization (%)	15.80%	20%	High (2,448 slices for 3×3 multiplier)
DSP Utilization (%)	55.2% (32/58)	Not used	Not specified
Throughput	1 output / cycle after latency	1 output / 16 cycles	1 full matrix / cycle
Scalability	High (modular PEs)	Limited	Parameterized, but area-intensive
Area Efficiency	Moderate	Good	Low (high area cost)
Application Focus	Real-time image & signal processing	Educational/low-power math unit	High-performance image core

Hardware Mapping

The systolic array is mapped onto the FPGA fabric as a grid of 64 PEs (8×8)**.** Each PE consists of:

- A multiplier (utilizing DSP blocks),
- An adder with a feedback register (using LUTs and flip-flops),
- Routing logic to pass data to adjacent PEs.

This local interconnection pattern reduces routing complexity and improves performance compared to centralized control logic.

Timing Characteristics

The array is filled in 8 clock cycles, after which valid outputs are produced every cycle until all outputs are computed. This pipelined execution model ensures maximum throughput and consistent timing behavior. The total latency for full matrix multiplication is approximately 16 clock cycles (8 for filling, 8 for computation completion), which remains constant due to the fixed matrix size.

Advantages of the Model

- **Parallelism**: All elements of the result matrix are computed simultaneously after the pipeline is filled.
- **Scalability**: Can be extended to larger matrix sizes by increasing array dimensions.
- **Resource Efficiency**: Utilizes FPGA logic in a modular and repeatable pattern, reducing synthesis complexity.
- **Throughput**: Ideal for real-time applications as outputs are generated at each clock cycle after initial latency.

CONCLUSION AND FUTURE WORK

This paper presented a systolic array-based matrix multiplier architecture implemented on a Spartan-6 FPGA for 8×8 matrix operations in real-time image and signal processing applications. The proposed design efficiently leverages the parallel processing capabilities of FPGAs, utilizing a grid of processing elements that perform multiply-and-accumulate operations in a pipelined fashion. Fixed-point arithmetic was adopted to optimize the use of LUTs and DSP slices, achieving a balanced trade-off between resource utilization and computational precision. The implementation results demonstrate the feasibility of real-

time matrix computation using low-cost FPGAs. With only 15.8% LUT utilization and moderate DSP usage, the design maintains a high throughput and predictable timing behavior, processing each matrix result in approximately 16 clock cycles. The architecture is modular, scalable, and suitable for embedded applications requiring deterministic performance.

In future work, the current design can be enhanced in multiple directions. First, support for larger matrix sizes can be incorporated by integrating external memory buffers and implementing tiling strategies. Second, partial reconfiguration may be explored to dynamically reconfigure matrix sizes or switch between fixed-point and floating-point computation modes. Finally, power optimization techniques**,** such as clock gating and data-aware activation, will be investigated to further adapt the system for energy-constrained environments such as portable medical imaging devices and edge AI accelerators.

REFERENCES

1. R. P. Singh, A. K. Vashishtha and R. Krishna, (2016)."32 Bit re-configurable RISC processor design and implementation for BETA ISA with inbuilt matrix multiplier," 2016 Sixth International Symposium on Embedded Computing and System Design (ISED), Patna, India, pp. 112-116, doi: 10.1109/ISED.2016.7977065.
2. M. Ashraf, M. S. Baig, L. A. Khan and A. Hassan, (2007). "Parallel Implementation of 2D-DWT by Purging Read after Write Dependency for High Speed Applications," 2007 International Conference on Emerging Technologies, Rawalpindi, Pakistan, pp. 263-268, doi: 10.1109/ICET.2007.4516355.
3. S. V. Mogre and D. G. Bhalke, (2015). "Implementation of High Speed Matrix Multiplier Using Vedic Mathematics on FPGA," 2015 International Conference on Computing Communication Control and Automation, Pune, India, pp. 959-963, doi: 10.1109/ICCUBEA.2015.190.
4. O. Mencer, M. Morf and M. J. Flynn, (1998). "PAM-Blox: high performance FPGA design for adaptive computing," Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines (Cat. No.98TB100251), Napa Valley, CA, USA. pp. 167-174, doi: 10.1109/FPGA.1998.707894.
5. M. M, N. S and K. S, (2021). "Bandwidth-Efficient Sparse Matrix Multiplier Architecture for Deep Neural Networks on FPGA," 2021 IEEE 34th International System-on-Chip Conference (SOCC), Las Vegas, NV, USA. pp. 7-12, doi: 10.1109/SOCC52499.2021.9739346.
6. J. Hu, W. Wang, R. C. C. Cheung and H. Wang, (2019). "Optimized Polynomial Multiplier Over Commutative Rings on FPGAs: A Case Study on BIKE," 2019 International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, pp. 231-234, doi: 10.1109/ICFPT47387.2019.00035.
7. S. Kumar, S. Patel, S. Singh, M. Das and B. Jajodia, (2024). "Design and Evaluation of FPGA-Optimized Symmetrical Three-Term Karatsuba Multipliers," 2024 International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST), Bengaluru, India, pp. 1-6, doi: 10.1109/ICRISST59181.2024.10921853.
8. S. Chandrasekaran and A. Amira, (2007). "Novel Sparse OBC based Distributed Arithmetic Architecture for Matrix Transforms," 2007 IEEE International Symposium on Circuits and Systems (ISCAS), New Orleans, LA, USA. pp. 3207-3210, doi: 10.1109/ISCAS.2007.378154.
9. B. Asgari, R. Hadidi and H. Kim, (2020). "MEISSA: Multiplying Matrices Efficiently in a Scalable Systolic Architecture," 2020 IEEE 38th International Conference on Computer Design (ICCD), Hartford, CT, USA, pp. 130-137, doi: 10.1109/ICCD50377.2020.00036.
10. Y. Zhang, Y. Cui, Z. Ni, D. -E. -S. Kundi, D. Liu and W. Liu, (2022). "A Lightweight and Efficient Schoolbook Polynomial Multiplier for Saber," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, pp. 2251-2255, doi: 10.1109/ISCAS48785.2022.9937496.
11. S. Belkacemi, K. Benkrid, D. Crookes and A. Benkrid, (2003). "Design and implementation of a high performance matrix multiplier core for Xilinx Virtex FPGAs," 2003 IEEE International Workshop on Computer Architectures for Machine Perception, New Orleans, LA, USA, pp. 4 pp.-159, doi: 10.1109/CAMP.2003.1598160.
12. C. Lin, Y. Liang and Y. Yi, (2022). "FPGA-based Reservoir Computing with Optimized Reservoir Node Architecture," 2022 23rd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, pp. 1-6, doi: 10.1109/ISQED54688.2022.9806247.
13. D. Zoni, A. Galimberti and W. Fornaciari, (2020). "Flexible and Scalable FPGA-Oriented Design of Multipliers for Large Binary Polynomials," in IEEE Access, vol. 8, pp. 75809-75821, 2020, doi: 10.1109/ACCESS.2020.2989423.
14. J. L. Imaña, (2018). "Efficient FPGA Implementation of Binary Field Multipliers Based on Irreducible Trinomials," IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Boulder, CO, USA, 2018, pp. 222-222, doi: 10.1109/FCCM.2018.00056.
15. R. Joshi et al., (2017). "An FPGA based floating point Gauss-Seidel iterative solver," 2017 14th IEEE India Council International Conference (INDICON), Roorkee, India. pp. 1-6, doi: 10.1109/INDICON.2017.8487543.