International Journal of Advanced Research and Multidisciplinary Trends (IJARMT)
An International Open Access, Peer-Reviewed Refereed Journal
Impact Factor: 6.4 Website: https://ijarmt.com ISSN No.: 3048-9458

# VLSI Architecture for Pipeline and Parallel Array based Matrix Multiplication using Deep Learning Technique

**Dheeraj Kumar**
M. Tech. Scholar, Department of Electronics and Communication
Bhabha Engineering Research Institute, Bhopal
**Prof. Suresh S. Gawande**
Guide, Department of Electronics and Communication
Bhabha Engineering Research Institute, Bhopal

**Abstract**
Matrix multiplication is a fundamental operation in various scientific computations, image processing, and particularly in deep learning applications where it forms the backbone of convolutional and fully connected layers. The growing demand for real-time processing in artificial intelligence (AI) and machine learning systems necessitates highly efficient hardware implementations. This paper presents VLSI architecture for pipelined and parallel array-based matrix multiplication optimized for deep learning techniques. The proposed design leverages pipelining to enhance throughput and reduce latency, while parallel array structures ensure efficient handling of large-scale matrix operations with minimized computational delay. By adopting deep learning-driven optimization strategies, the architecture achieves improvements in area utilization, delay, and power efficiency compared to conventional multiplier-based designs. Simulation and synthesis results validate the effectiveness of the proposed approach, demonstrating its suitability for high-performance computing platforms, neural network accelerators, and embedded AI systems.
In this paper, we have proposed MM using deep learning approach. This design reduced hardware complexity, delay and input/output data format to match different application needs. The PPI-MO based MM is design Xilinx software and simulated number of slice, look up table and delay.

Keywords: - Deep Learning, Matrix Multiplication, Parallel Technique, Pipeline Technique

## 1. INTRODUCTION

Matrix multiplication is a fundamental mathematical operation extensively used in scientific computing, digital signal processing (DSP), image processing, and most prominently, in deep learning applications. In neural networks, especially in convolutional and fully connected layers, a significant portion of the computational workload arises from large-scale matrix multiplications. With the growing adoption of artificial intelligence (AI) in domains such as computer vision, natural language processing, and healthcare, the need for efficient hardware implementations of matrix multiplication has become increasingly critical. Traditional software-based implementations running on general-purpose processors are often inadequate due to high computational complexity, latency, and power consumption. Hence, hardware accelerators using Very Large Scale Integration (VLSI) architectures are being explored as an effective solution to address these challenges.

The demand for real-time processing in deep learning tasks requires architectures that can balance high throughput, reduced latency, low area, and power efficiency. VLSI architectures provide an opportunity to design specialized hardware for matrix multiplication that can outperform conventional processors and graphics processing units (GPUs) in terms of performance-per-watt and scalability. Among the existing approaches, pipelined and parallel array architectures have emerged as promising solutions. Pipelining enhances throughput by overlapping computations, thereby ensuring continuous data flow across stages, while parallel array structures allow multiple operations to be carried out simultaneously. The combination of these two design strategies enables highly efficient computation of large matrices, which is essential for deep learning applications.

Conventional multiplier-based matrix multipliers suffer from high area consumption and increased propagation delays as the matrix size grows. To overcome these limitations, optimized VLSI architectures focus on reducing the number of arithmetic units, sharing computations, and efficiently utilizing interconnections. The pipelined approach ensures minimal idle cycles during computation, while parallel array multipliers are designed to exploit data-level parallelism inherent in deep learning workloads. These optimizations make the architecture suitable for applications ranging from edge AI devices to large-scale cloud-based accelerators.

Furthermore, deep learning techniques themselves can be leveraged to guide hardware optimization. For instance, approximate computing, pruning, and quantization strategies in deep learning models reduce the precision requirements of matrix multiplication, thereby allowing simplified hardware implementations. By integrating such techniques, the VLSI architecture can be tailored to trade off between accuracy and efficiency depending on the application requirements. This is particularly relevant in embedded and mobile AI systems where power consumption is a critical constraint.

Recent advancements in VLSI design methodologies, such as the use of parallel systolic arrays, reconfigurable architectures, and multiplier-less techniques, have further strengthened the scope of matrix multiplication hardware accelerators. Systolic arrays, in particular, provide a scalable and regular structure that supports efficient mapping of deep learning algorithms, making them highly attractive for large neural network computations. The proposed work builds on these advancements by designing a pipelined and parallel array-based VLSI architecture specifically optimized for deep learning workloads, ensuring improvements in computation speed, area efficiency, and energy consumption.

In summary, the motivation behind this work lies in addressing the growing computational demands of deep learning through efficient hardware design. By combining pipelining, parallel array processing, and VLSI design techniques, the proposed architecture aims to deliver high-performance, low-power, and scalable matrix multiplication suitable for next-generation AI accelerators.

## 2. MATRIX MULTIPLICATION

The matrix multiplication can be represented as

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_m \end{bmatrix}_{M\mathbf{x}1} = \begin{bmatrix} a_{11} & a_{12} & . & . & a_{1n} \\ a_{21} & a_{22} & . & . & a_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{m1} & a_{m2} & . & . & a_{mn} \end{bmatrix}_{M\mathbf{x}N} \mathbf{x} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}_{N\mathbf{x}1}$$

**Figure 1: The Matrix Multiplication Operation**

(1)

Where ***Y*** and ***x*** are column vectors of size M and N respectively and ***A*** is an M×N matrix.

Inner-Product (IP): It is defined as the scalar multiplication of two vectors

$$y(\mathrm{i}) = \mathrm{A_i x} \qquad (2)$$

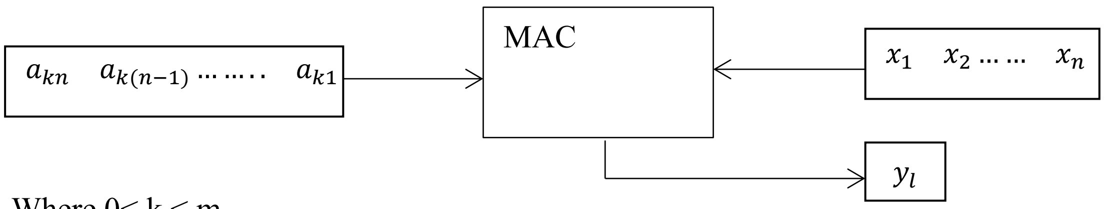
Where $A_i$ is the ith row of matrix A

So a Matrix-vector multiplication can be performed through M inner-product computation for M rows in A. Each Inner product computation (IPC) involves N multiply and add operations. Inner product computation can performed using two methods. These are:

From figure 3.2 it is observed that N cycles are required to perform 1 IPC of size N where one clock cycle equals $T_M + T_A$.

So, MN cycles are required to complete matrix-vector multiplication of size M×N with N× 1 where M numbers of IPC are performed each requiring N cycles.

Using 1 MAC unit



Where $0 < k < m$

$0 < l < n$

**Figure 2: IPC Using Single MAC**

So, M cycles are required to complete matrix-vector multiplication of size M×N with N× 1 where M IPC are performed each requiring 1 cycle for computation.

The conditions presented above show the extreme computing complexities. The computation time could be set in between M and MN cycles by suitably choosing the number of MAC units.

Using N MAC units

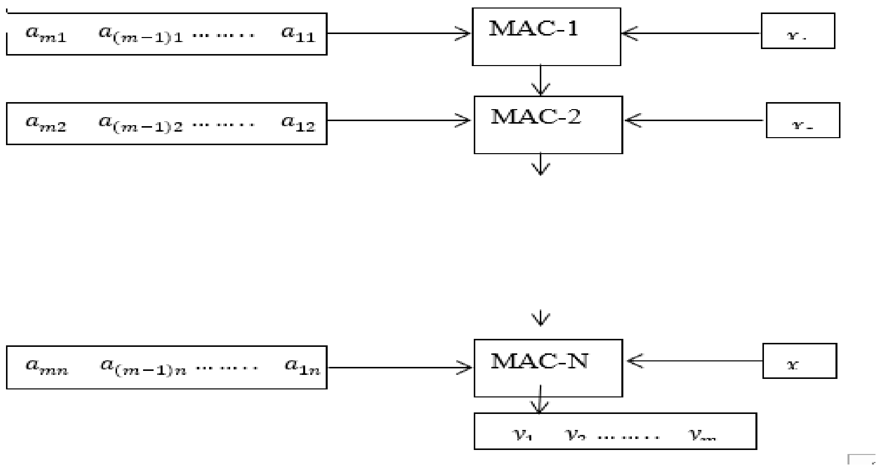

$a_{m1}$ $a_{(m-1)1}$ ⋯ $a_{11}$
$a_{m2}$ $a_{(m-1)2}$ ⋯ $a_{12}$
$a_{mn}$ $a_{(m-1)n}$ ⋯ $a_{1n}$
MAC-1
MAC-2
MAC-N
$y_1$ $y_2$ ⋯ $y_m$

**Figure 3: IPC using multiple MACs**

## 3. METHODOLOGY

### Proposed Parallel-Parallel Input and Multi Output (PPI-MO)

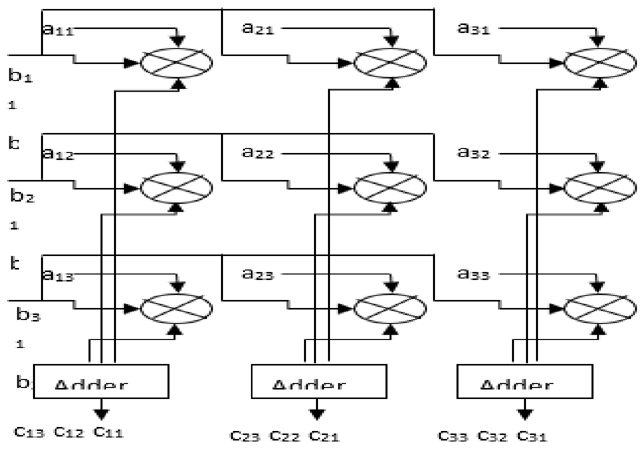


**Figure 4: Proposed PPI – MO Design for n = 3**

In this design, we opted for faster operating speed by increasing the number of multipliers and registers performing the matrix multiplication operation.

We have derived for parallel computation of 3 × 3 matrix-matrix multiplication and the structure is shown in figure 4.

For an n×n matrix – matrix multiplication, the operation is performed using $n^2$ number of multipliers, $n^2$ number of registers and $n^2 - n$ number of adders. The registers are used to store the partial product results. Each of the $n^2$ number of multipliers has one input from matrix B and the other input is obtained from a particular element of matrix A.

The dataflow for matrix B is in row major order and is fed simultaneously to the particular row of multipliers such that the $i^{th}$ row of matrix B is simultaneously input to the $i^{th}$ row of multipliers, where $1 < i < n$ . The elements of matrix are input to the multipliers such that, $(j,i)^{th}$ element of matrix A.

The $(i,j)^{th}$ multiplier, where$1 < i,j < n$. The resultant products from each column of multipliers are then added to give the elements of output matrix C. In one cycle, n elements of matrix C are calculated, so the entire matrix the elements of matrix C are obtained in column major order with n elements multiplication operation requires n cycles to complete.

Let us consider the example of a 3×3 matrix – matrix multiplication operation, for a better analysis of the design (as shown in figure 1). The hardware complexities involved for this design are 9 multipliers, 9 registers and 6 adders. Elements from the first row of matrix B ($b_{11}$ $b_{12}$ $b_{13}$) are input simultaneously to the first row of multipliers ($M_{11}$ $M_{12}$ $M_{13}$) in 3 cycles. Similarly, elements from other two rows of matrix B are input to the rest two rows of multipliers. A single element from matrix A is input to each of the multipliers such that, $(j,i)^{th}$ element of matrix A is input to the multiplier $M_{ij}$, where $1 < i,j < 3$. The resultant partial products from each column of multipliers ($M_{1k}$ $M_{2k}$ $M_{3k}$ where $1 < k$ 3) are added up in the adder to output the elements of matrix C. In each cycle, one column of elements from matrix C is obtained ($C_{1k}$ $C_{2k}$ $C_{3k}$ where$1 < k < 3$) and so the entire matrix multiplication operation is completed in 3 cycles.

## 4. SIMULATION RESULT

A FPGA (Field Programmable Gate Array) is an incorporated circuit comprising of an assortment of rationale squares, I/O cells and interconnection assets and this permits the chip to be reconfigured to associate the sources of info and yields (I/O) and rationale squares together from various perspectives. The representations of the permanent points and floating points are generally used in numerous applications. It is mainly applicable for designing process of the DSP applications. Also, floating point is demonstrated as very small to large numbers, which employed with the improved range. Every rationale square has customarily the capacity to do a basic rationale activity, for example, AND or XOR, and for the most part contains some level of memory, it be a straightforward flip-flop or a progressively intricate square of memory. The rationale squares have developed to be more rationale work squares utilizing query tables inside the squares to switch the current capacity; to perform assignments such math tasks.

For parallel in multiple out shift registers, all data bits appear on the parallel input immediately following the simultaneous entry of the date bits. Four-bit parallel in multiple out shift register is constructed by four D flip-flops.

In fig. 5 and fig. 6 have shown the resistor transistor logic (RTL) using 3×3 PPI-MO matrix multiplication and output waveform of 3×3 PPI-MO matrix multiplication respectively.

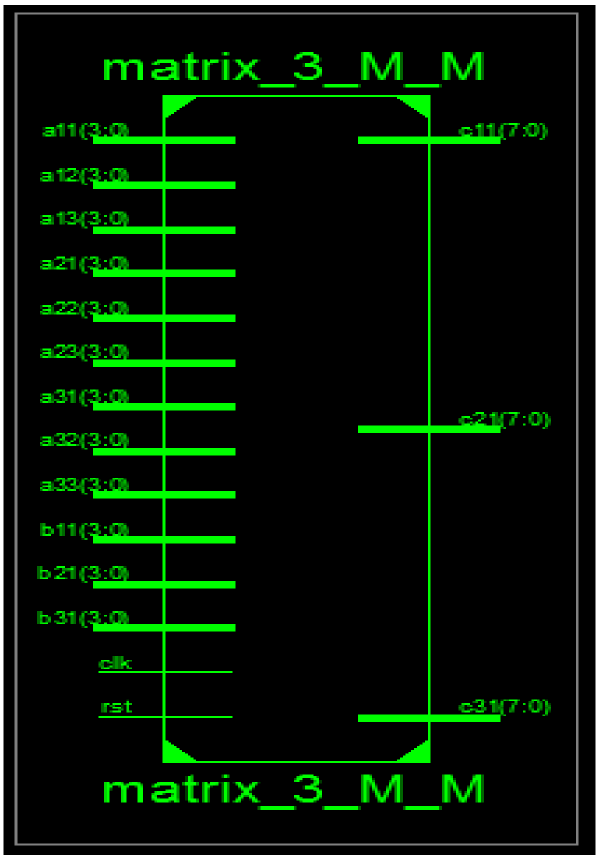


**Figure 5: View Technology Schematic of 3×3 Matrix Multiplications using PPI-MO**

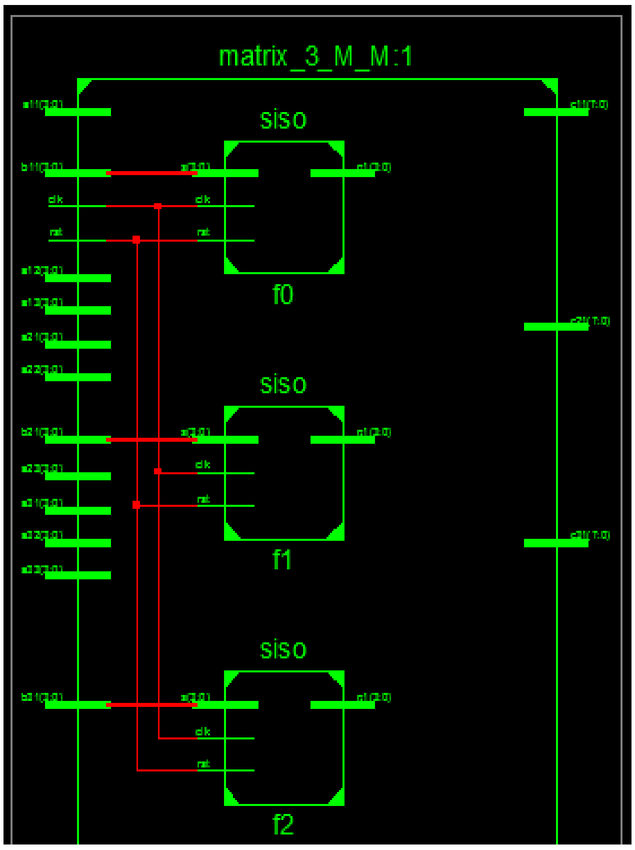
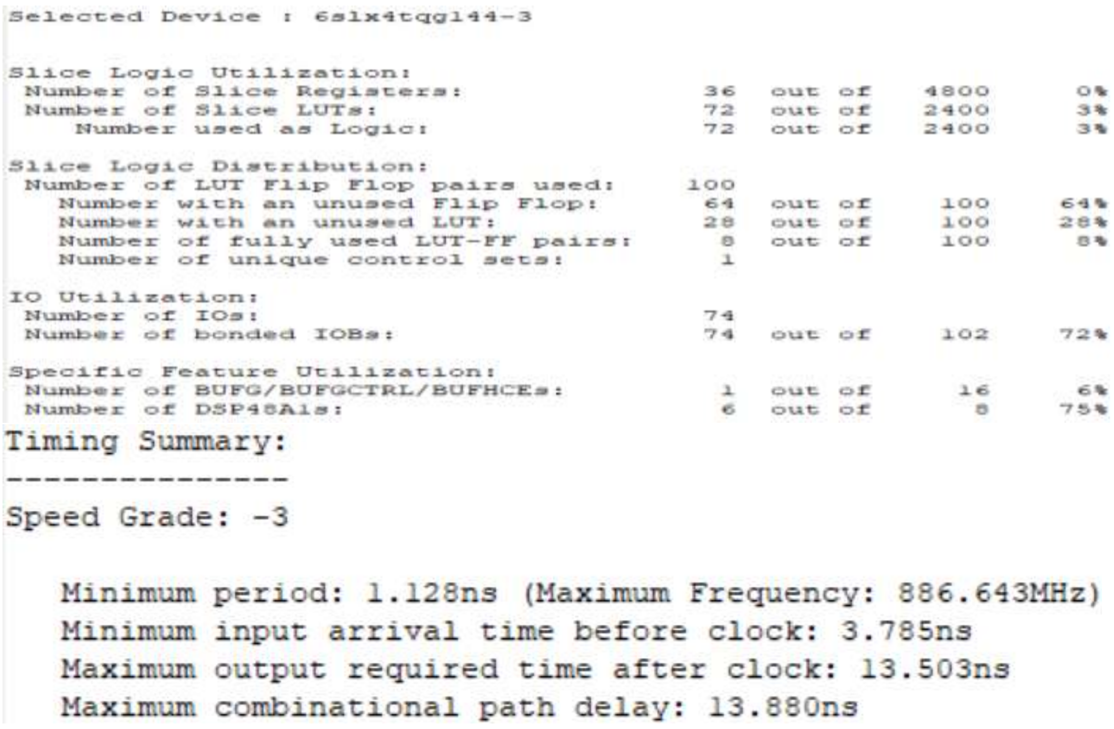


**Figure 6: View Technology Schematic of 3×3 Matrix Multiplications using PPI-MO**



```
Selected Device : 6slx4tqg144-3

Slice Logic Utilization:
 Number of Slice Registers:                36  out of   4800     0%
 Number of Slice LUTs:                     72  out of   2400     3%
    Number used as Logic:                  72  out of   2400     3%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:      100
   Number with an unused Flip Flop:        64  out of    100    64%
   Number with an unused LUT:              28  out of    100    28%
   Number of fully used LUT-FF pairs:       8  out of    100     8%
   Number of unique control sets:           1

IO Utilization:
 Number of IOs:                            74
 Number of bonded IOBs:                    74  out of    102    72%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRL/BUFHCEs:           1  out of     16     6%
 Number of DSP48A1s:                        6  out of      8    75%
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 1.128ns (Maximum Frequency: 886.643MHz)
   Minimum input arrival time before clock: 3.785ns
   Maximum output required time after clock: 13.503ns
   Maximum combinational path delay: 13.880ns
```

**Figure 7: Summary of 3×3 Matrix Multiplications using PPI-MO**

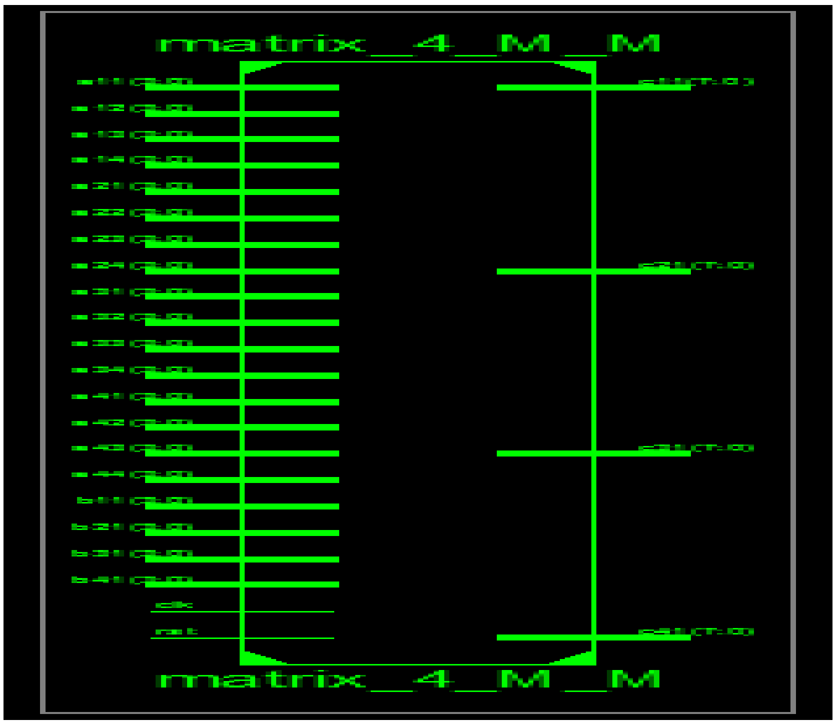


Figure 8: View Technology Schematic of 4×4 Matrix Multiplications using PPI-MO

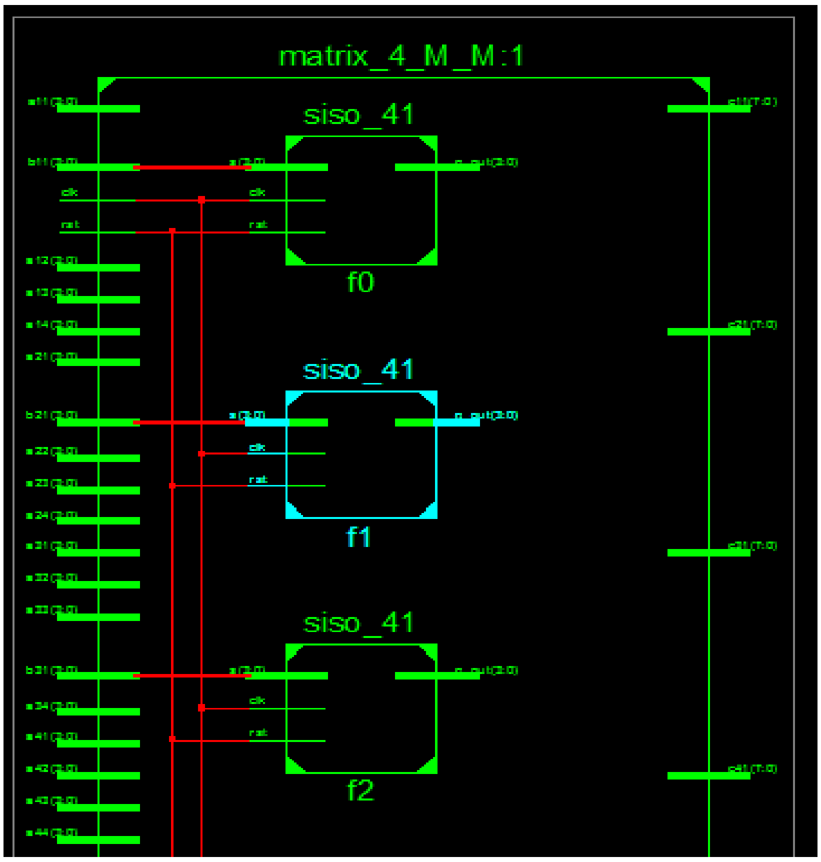
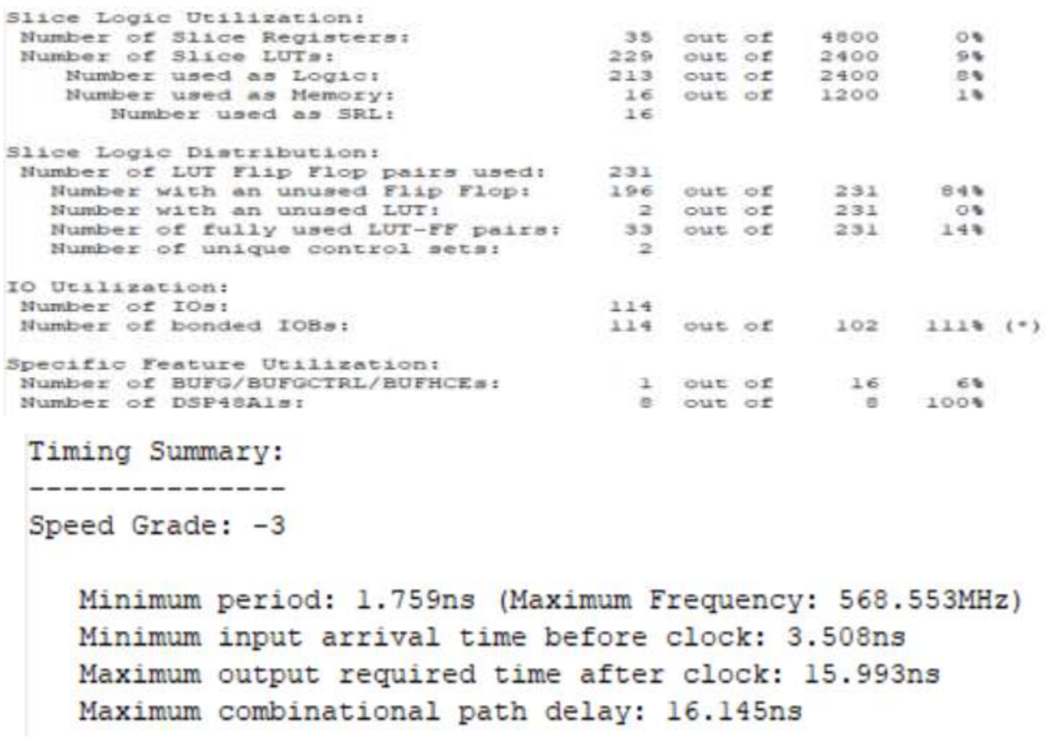


Figure 9: View Technology Schematic of 4×4 Matrix Multiplications using PPI-MO



```
Slice Logic Utilization:
  Number of Slice Registers:                 35  out of   4800     0%
  Number of Slice LUTs:                     229  out of   2400     9%
     Number used as Logic:                  213  out of   2400     8%
     Number used as Memory:                  16  out of   1200     1%
        Number used as SRL:                  16

Slice Logic Distribution:
  Number of LUT Flip Flop pairs used:       231
    Number with an unused Flip Flop:        196  out of    231    84%
    Number with an unused LUT:                2  out of    231     0%
    Number of fully used LUT-FF pairs:       33  out of    231    14%
    Number of unique control sets:            2

IO Utilization:
  Number of IOs:                            114
  Number of bonded IOBs:                    114  out of    102   111% (*)

Specific Feature Utilization:
  Number of BUFG/BUFGCTRL/BUFHCEs:            1  out of     16     6%
  Number of DSP48A1s:                         8  out of      8   100%
```

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 1.759ns (Maximum Frequency: 568.553MHz)
   Minimum input arrival time before clock: 3.508ns
   Maximum output required time after clock: 15.993ns
   Maximum combinational path delay: 16.145ns
```

**Figure 10: Summary of 4×4 Matrix Multiplications using PPI-MO**

## 5. CONCLUSION

Matrix multiplication plays a vital role in deep learning and high-performance computing, making its efficient hardware implementation essential for real-time applications. This work presented VLSI architecture for pipelined and parallel array-based matrix multiplication aimed at optimizing performance for deep learning workloads. By combining pipelining techniques with parallel array structures, the proposed architecture achieves enhanced throughput, reduced latency, and efficient resource utilization. Compared to conventional multiplier-based designs, it offers significant improvements in area efficiency, power consumption, and scalability, which are crucial for implementing large-scale neural networks.

The integration of deep learning-driven optimization strategies further strengthens the architecture's applicability, enabling it to handle approximate computing, quantization, and pruning-based workloads with minimal loss in accuracy. As a result, the design is well-suited for AI accelerators in domains such as image processing, natural language processing, healthcare, and embedded intelligence.

Overall, the proposed VLSI architecture demonstrates that pipeline and parallel array-based design methodologies can effectively address the challenges of speed, efficiency, and scalability in matrix multiplication. Future work may focus on extending the architecture for reconfigurable designs, multiplier-less techniques, and energy-aware computation, thereby broadening its scope for next-generation edge AI and cloud-based deep learning systems.

## REFERENCES

[1] E. S, S. S. A, S. D and N. M, "VLSI Implementation of Pipelined PE Systolic Array-Based 3x3 Matrix Multiplication for Deep Neural Network Accelerator," *2025 Fourth International Conference on Smart Technologies, Communication and Robotics (STCR)*, Sathyamangalam, India, 2025, pp. 1-4.

[2] Alaejos, G., Castelló, A., Martínez, H., Alonso-Jordá, P., Igual, F.D., Quintana-Ortí, E.S.: Micro-kernels for portable and efficient matrix multiplication in deep learning. J. Supercomput. **79**(7), 8124–8147, 2023.

[3] Kim, D., Kim, J.: Analysis of several sparse formats for matrices used in sparse-matrix dense-matrix multiplication for machine learning on GPUs. In: 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), pp. 629–631. IEEE, 2022

[4] Kim, S., Kim, J., Kim, N., Kang, M., Seo, J.: Improving inference time of deep learning model with partial skip of ReLU-fused matrix multiplication operations. In: 2022 International Conference on Electronics, Information, and Communication (ICEIC), pp. 1–4. IEEE, 2022.

[5] Rizwan, M., Jung, E., Park, Y., Choi, J., Kim, Y.: Optimization of matrix-matrix multiplication algorithm for matrix-panel multiplication on intel KNL. In: 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA), pp. 1–7. IEEE, 2022.

[6] Chen Yang, Siwei Xiang, Jiaxing Wang, Liyan Liang, "A High Performance and Full Utilization Hardware Implementation of Floating Point Arithmetic Units", 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), IEEE 2021.

[7] Wei Mao, Kai Li, Xinang Xie, Shirui Zhao, He Li;Hao Yu, "A Reconfigurable Multiple-Precision Floating-Point Dot Product Unit for High-Performance Computing", Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE 2021.

[8] Di Yan, Wei-Xing Wang, Lei Zuo and Xiao-Wei Zhang, "Revisiting the Adjoint Matrix for FPGA Calculating the Triangular Matrix Inversion", IEEE Transactions on Circuits and Systems II: Express Briefs, 2020.

[9] X.-W. Zhang, L. Zuo, M. Li and J.-X. Guo, "High-throughput FPGA implementation of Matrix inversion for control systems," Accepted by IEEE Transactions Ind. Electron., 2020.

[10] S. Ross Thompson and James E. Stine, "A Novel Rounding Algorithm for a High Performance IEEE 754 Double-Precision Floating-Point Multiplier", 38th International Conference on Computer Design (ICCD), IEEE 2020.

[11] P.L. Lahari, M. Bharathi, Yasha Jyothi and M Shirur, "High Speed Floating Point Multiply Accumulate Unit using Offset Binary Coding", 7th International Conference on Smart Structures and Systems (ICSSS), IEEE 2020.

[12] C. Zhang, et al, "On the low-complexity, hardware-friendly tridiagonal matrix inversion for correlated massive MIMO systems," IEEE Trans. Vehic. Tech., vol. 68, no. 7, pp. 6272-6285, Jul. 2019.

[13] S. Venkatachalam E. Adams H. J. Lee and S.-B. Ko "Design and analysis of area and power efficient approximate booth multipliers" IEEE Trans. Comput. vol. 68 no. 11 pp. 1697-1703 Nov. 2019.

[14] Y.-W. Xu, Y. Xi, J. Lan and T.-F. Jiang, "An improved predictive controller on the FPGA by hardware matrix inversion," IEEE Trans. Ind. Electron., vol. 65, no. 9, pp. 7395–7405, Sep. 2018.

[15] Lakshmi kiran Mukkara and K.Venkata Ramanaiah, "A Simple Novel Floating Point Matrix Multiplier VLSI Architecture for Digital Image Compression Applications", 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018) IEEE.

[16] D. Kalaiyarasi and M. Saraswathi, "Design of an Efficient High Speed Radix-4 Booth Multiplier for both Signed and Unsigned Numbers", 4th International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), IEEE 2018.