VOL. 24, NO. 1, 2025, 53-56
elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Heterogeneous Hardware Architecture with Linear Algebra Acceleration

Chun F. Ong* and **M. N. Marsono**

Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.

*Corresponding author: cfong2@graduate.utm.my

Abstract: Linear algebra is essential in machine learning for dealing with large datasets. Linear algebra acceleration is directly related to the hardware used. Many works have proposed linear algebra accelerator architectures with the goal of improving energy efficiency and speed. The characterization of trade-offs in balancing acceleration and programmability of software routines is still insufficiently explored, particularly for edge analytics. Therefore, this paper proposes a heterogeneous hardware architecture consisting of a RISC-V system-on-chip and a linear algebra accelerator. Tested on Efinix Trion T120BGA324, the new architecture incorporates software routines and is clocked at 50 MHz. The improved design provides better timing closure and lower logic element use, with the lowest slack being 2.102 ns and the highest logic element use being 66.40%. The design incorporates a software routine for improved data management, reduced hardware resource utilization, and some computational load. The results show that the heterogeneous architecture outperforms the RISC-V System-on-Chip standalone by 156× in General Matrix Multiplication without accuracy loss.

Keywords: General Matrix Multiplication (GEMM), Linear algebra accelerator, Efinix Trion T120BGA324, RISC-V SoC

Article History: received 3 November 2024; accepted 19 February 2025; published 30 April 2025

1. INTRODUCTION

Machine learning has grown in popularity and application over the last decade, and many researchers are working to improve the machine learning algorithm and improve linear algebra acceleration. As machine learning is to process large data, it is better to process them in vectors or matrices, and linear algebra defines the study of vectors and matrices, which provides the operations of vectors and matrices on the data. Therefore, it is a fundamental and primary step in machine learning.

The majority of linear algebra accelerations are built on a single architecture. A central processing unit has the worst performance in linear algebra operation compared to graphical processing unit (GPU) and field programmable gate array (FPGA) [1, 2]. Because CPU is emphasized on its single-thread performance. Even though state-or-the-art CPUs were designed with parallelism techniques such as multi-threading, it is not guaranteed that the CPU will improve but most probably degrade the performance as it would magnify the non-deterministic impact on deep learning system that inconsistent models and results would be produced even though with the same settings, parameters, and datasets, on the identical hardware architecture [3]. Modern GPUs perform well in parallel computing on iterative tasks that demand intensive computation [4]. Thus, GPU has been the top choice in research for high-performance computing, especially machine learning for the last decade [5, 6], but GPU is not energy efficient, which is critical for edge processing. The application-specific integrated circuit (ASIC) is often the choice for the deployment of linear algebra acceleration [7]. However, an ASIC has a high development cost and low flexibility, as no optimization or reconfiguration is possible once the design is fabricated.

FPGA provides high flexibility because they are reconfigurable and reprogrammable. Heterogeneous hardware architecture with a parameterizable scalar core and a linear algebra accelerator is possible with the recently emerging instruction set architecture (ISA) RISC-V. Furthermore, because RISC-V is open-source, there are several free-to-use RISC-V cores available to any developer to modify and redesign the core to suit their desired specification. FPGA based RISC-V Sapphire SoC provided by Efinity is such a platform, which allows users to reconfigure the core at both a high level and detail reconfiguration via Verilog.

A linear algebra accelerator that is implemented in the form of extended instruction was proposed [7]. Both the accelerator and main core shared the same caches and memory which limits the data access and deteriorates the General Matrix Multiplication (GEMM) and Deep Neural Network (DNN) performance as they demand huge memory. Also, one of the linear algebra operations, such as the GEMM has a complexity of O(n3). More effort is needed in hardware design to obtain good timing closure with low resource usage.

In scientific research, especially in computer engineering, linear algebra is standardized, namely linear algebra subprograms (BLAS) [2]. BLAS are routines that provide standard building blocks for basic vector and matrix operations, which are widely used in machine learning [8]. It is divided into three levels: BLAS level 1

contains vector-vector operations, BLAS level 2 is for vector-matrix operations, and BLAS level 3 is for matrix-matrix operations [1, 9]. As linear algebra operations are fundamental in machine learning, the invention of the BLAS library makes easy the development of high-performance computing (HPC). Also, it ensures the performance portability of linear algebra routines, making it a standard building block in HPC [5].

This paper presents a heterogeneous architecture with linear algebra acceleration at level 3 Basic Linear Algebra Subprogram (BLAS). This is achieved by integrating an open-source linear algebra accelerator and parameterizable RISC-V core on the Efinix platform. We proposed an optimized design that requires low frequency at 50 MHz with reduced resource usage at 66.40% of logic elements used and achieves 2.102 ns positive slack. Results show a 100% accuracy and correctness of output matrix × data with 156 performance improvement. The heterogeneous hardware architecture provides a practical solution for improved linear algebra acceleration.

2. PRELIMINARY

We are targeting level 3 BLAS when designing the linear algebra accelerator. In this section, some basic notations, terms, and definitions related to GEMM are presented.

Definition 1. GEMM [10]
The GEMM is a dot product operation on two 2-D matrices as described in the notation:

$$C = \alpha A \times B + \beta C \quad (1)$$

where A, B, and C are matrices, α and β are scalar constants, where α to 1 and β to 0. The linear algebra accelerator is designed to calculate independent output without considering the previous result. To perform matrix multiplication, it is necessary to make sure the column of matrix A is equal to the row of matrix B. For instance, matrix A of m × k multiplied with matrix B of k × n will get matrix C of m × n. In this project, we are setting m, n, and k to have the same value.

Definition 2. GEMM Performance in Operations per Second [11]
The performance of GEMM is calculated in terms of the number of operations per second (OP/s). The number of operations of matrix multiplication can be calculated by using the notation:

$$OP = 2mnk \quad (2)$$

where m is the row of matrix A and matrix C, n is the column of matrix B and matrix C, and k is the row and column of matrix A and matrix B respectively.

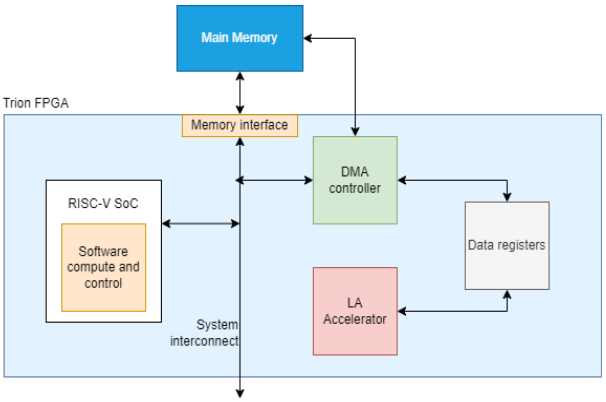

Figure 1. Proposed heterogeneous hardware architecture.

3. HETEROGENEOUS HARDWARE ARCHITECTURE

This section presents the proposed hardware architecture with linear algebra acceleration. As shown in Figure 1, the design consists of a RISC-V SoC, direct memory access (DMA) controller, registers, and the linear algebra accelerator. Figure 2 shows the algorithm for the linear algebra accelerator.

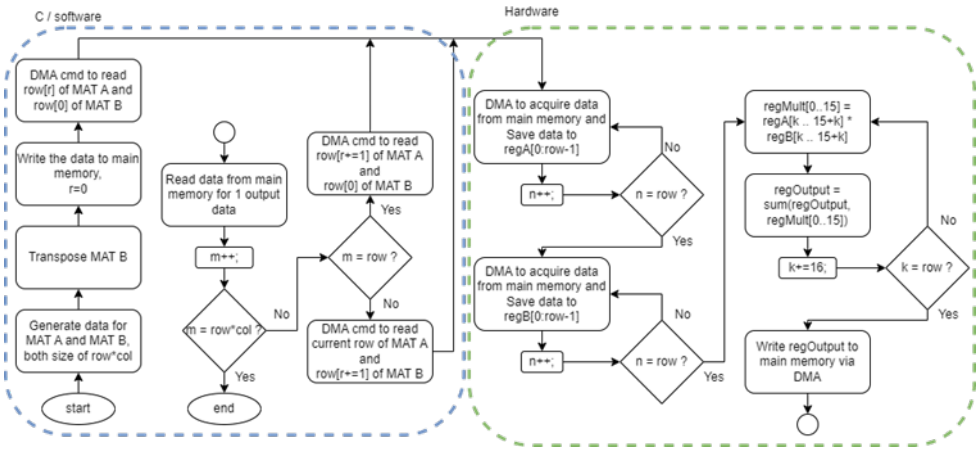

Figure 2. Algorithm of Linear Algebra Accelerator

To make sure the timing closure is met, and to ensure that fewer resources are used, a software routine is

included that works with the hardware design. As shown in Figure 3 and Figure 4, 16 data from each matrix A and matrix B will be stored in the registers to perform the dot product. The data is accumulated until one output matrix C element is computed. In this design, if the matrix size of A and B is larger than 16 for both columns and rows, the k counter is incremented by 16 in each loop until it reaches the k dimension of the matrix. This vastly reduces resource usage and improves timing closure.

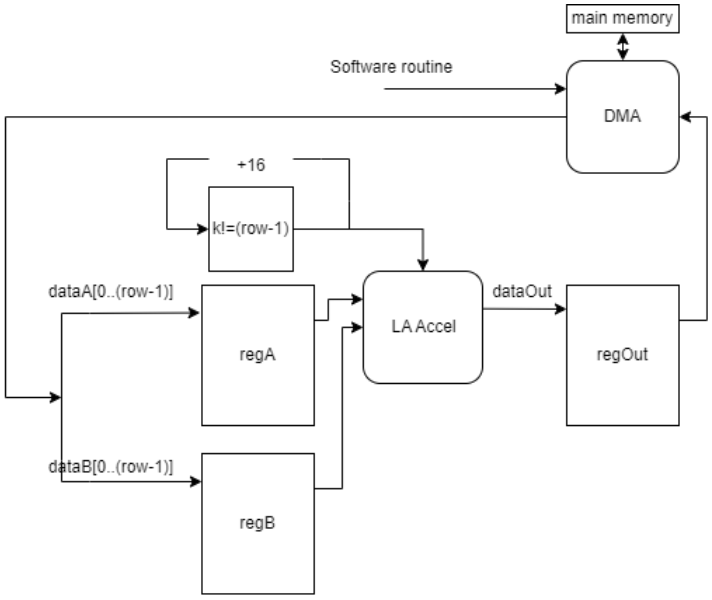

Figure 3. Functional block diagram of hardware architecture.

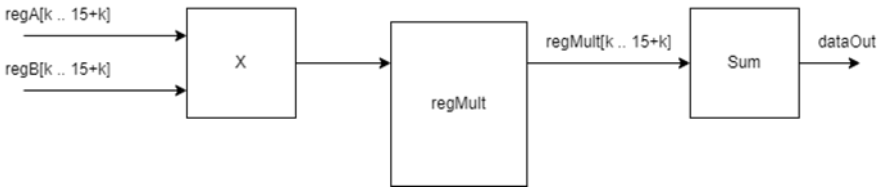

Figure 4. Linear algebra accelerator.

This is accomplished by including a software routine that manages the data by transposing matrix B and controlling memory accesses. This is made easier by using Efinix design platforms to implement the design. The Efinity IDE and Eclipse are two tools provided by Efinix. The heterogeneous architecture hardware design is created in Verilog on Efinity IDE and programmed to Trion FPGA on the Efinix development board T120BGA324. The software written in C on Eclipse will start the GEMM by sending commands to the hardware architecture, which will read data from the main memory via DMA.

The program is designed to read one row of data from matrix A and one row of data from transposed matrix B. Following the completion of the data read, the linear algebra accelerator will perform the dot product until one element of matrix C is calculated. The data is then written to main memory using DMA and as directed by the software. To compute the second element of matrix C, the software will instruct the hardware to read the second row of transposed matrix B, and the data of matrix A will remain unchanged until all rows of matrix B are read, indicating the elements in matrix C's first row are computed. The software will then notify the hardware to read the second row of matrix A and the first row of transposed matrix B in order to compute the first element in matrix C's second row. This procedure is repeated until all of the elements of matrix C have been calculated.

4. PERFORMANCE ASSESSMENT

It has an average performance of 1.7×10^{-1} MOP/s for all the matrix sizes with the highest performance of 1.9×10^{-1} MOP/s for matrix size of 256 × 256. This outcome clearly shows that a single-core hardware architecture is insufficient in performing the GEMM.

However, the GEMM performance improves significantly when the RISC-V SoC is paired with a linear algebra accelerator. The results show an increased performance when the matrix sizes increase. The improvement hit the highest with a performance of 285×10^{-1} MOP/s which is an improvement factor of 156 for matrix size of 512 × 512, and the lowest with a performance of 48.4×10^{-1} MOP/s which is an improvement factor of 29.9 for matrix size of 16 × 16.

Table 1. Performance and accuracy comparison between RISC-V standalone and RISC-V with LA Accelerator

Matrix Size (n*n)	RISC-V Standalone (10^{-1}MOP/s)	RISC-V with LA Accelerator (10^{-1}MOP/s)	Improvement In Performance	RISC-V Standalone Accuracy (%)	RISC-V with LA Accelerator Accuracy (%)
16	1.6	48.4	29.9	100	100
32	1.5	88.1	55.2	100	100
64	1.6	138.2	85.3	100	100
128	1.6	198.8	121.9	100	100
256	1.9	246.4	128.4	100	100
512	1.8	285.0	156.0	100	100

From the results, the resources used hit the minimum of 14.61% of total logic elements used, and a maximum of 66.40% of total logic elements used. This outcome shows that the design has space to configure to accommodate bigger matrix sizes. Also, timing closure is met such that the design can achieve the highest positive slack of 4.274 ns at the matrix size of 32 × 32 and the smallest positive slack of 2.102 ns at the matrix size of 512 × 512.

As the processing paradigm shifts to small-sized matrices, The design can be restructured as the processing paradigm shifts to small-sized matrices as well. It is referred to as batched matrix multiplication. Machine learning begins to utilize more multiple small matrix operations and gradually becomes the core operation of the machine learning framework.

Table 2. Resource usage and worst setup time

Matrix Size (n*n)	Resource Usage (%)	Worst Setup time (ns)
16	14.61	2.664
32	17.41	4.274
64	20.07	3.814
128	26.67	3.659
256	40.82	3.602
512	66.40	2.102

5. CONCLUSION

A heterogeneous hardware architecture with linear algebra acceleration was presented in this paper. We demonstrated that the GEMM performance on a platform consisting of a RISC-V SoC and a dedicated linear algebra accelerator is improved by 156× when compared to a RISC-V SoC alone with the same output correctness of 100%. We manage to keep the design resource usage at 66.40% of total logic elements used with the worst positive setup time of 2.102 ns using software. The design can support up to 512 × 512 matrix multiplication.

This project's design can be improved in the future so that it can accommodate larger matrices. The software routine can still be included in the design to balance the work between software and hardware while also reducing hardware resources for better design synthesis. Also, this architecture could be developed to heterogeneous manycore hardware architecture to further improve the performance.

REFERENCES

[1] Xiong, C. and Xu, N. Performance comparison of BLAS on CPU, GPU and FPGA. 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). IEEE. 2020, vol. 9. 193–197.

[2] Yin, L., Zhang, Y., Zhang, Z., Peng, Y. and Zhao, P. ParaX: boosting deep learning for big data analytics on many-core CPUs. Proceedings of the VLDB Endowment, 2021. 14(6): 864–877.

[3] Xiao, G., Liu, J., Zheng, Z. and Sui, Y. Nondeterministic Impact of CPU Multithreading on Training Deep Learning Systems.ISSRE. 2021. 557–568.

[4] Zhang, X., Tang, Z., Du, L. and Yang, L. An incremental iterative acceleration architecture in distributed heterogeneous environments with GPUs for deep learning. IEEE Transactions on Parallel and Distributed Systems, 2021. 32(11): 2823–2837.

[5] Gautier, T. and Lima, J. V. Xkblas: a high-performance implementation of blas- 3 kernels on multi-gpu server. 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE. 2020. 1–8.

[6] Dongarra, J., Gates, M., Kurzak, J., Luszczek, P. and Tsai, Y. M. Autotuning numerical dense linear algebra for batched computation with GPU hardware accelerators. Proceedings of the IEEE, 2018. 106(11): 2040–2055.

[7] Steffl, S. and Reda, S. Lacore: A supercomputing-like linear algebra accelerator for soc-based designs. 2017 IEEE International Conference on Computer Design (ICCD). IEEE. 2017. 137–144.

[8] Browne, S., Dongarra, J., Grosse, E. and Rowan, T. The Netlib mathematical software repository. D-lib Magazine, 1995. 1(9).

[9] Fibich, C., Tauner, S., Rossler, P. and Horauer, M. Evaluation of Open-Source Linear Algebra Libraries targeting ARM and RISC-V Architectures. 2020 15th Conference on Computer Science and Information Systems (FedCSIS). IEEE. 2020. 663–672.

[10] Wang, R., Yang, Z., Xu, H. and Lu, L. A high-performance batched matrix multiplication framework for GPUs under unbalanced input distribution. The Journal of Supercomputing, 2022. 78(2): 1741–1758.

[11] Abdelfattah, A., Tomov, S. and Dongarra, J. Fast batched matrix multiplication for small sizes using half-precision arithmetic on GPUs. 2019 IEEE international parallel and distributed processing symposium (IPDPS). IEEE. 2019. 111–122.