Implementation and Benchmarking of a Near Memory Hardware Vector Adder for Embedded SoC Processing Acceleration

Eduardo Carvalho, *Student, UFMG,* Renan Silva, *Student, UFMG,*

***Abstract*—Processing In Memory (PIM) and Near Memory Processing (NMP) have been areas of research since the 1960s, driven by the need to overcome the limitations of traditional von Neumann architectures, particularly the bottleneck associated with data transfer between the CPU and memory. As technology has evolved, the tools and resources available for developing NMP-based solutions have become more sophisticated, making it feasible for vendors to implement NMP-centered Intellectual Property (IP) cores effectively.**

In this research, we focus on the practical implementation of a hardware vector adder using NMP principles, leveraging AMD's Vitis High-Level Synthesis (HLS) and Vivado tools. The primary goal is to assess the efficiency gains achievable by performing computations closer to memory, thus minimizing data movement and reducing latency. By comparing the performance and resource utilization of the vector adder implemented in Vitis HLS, Vivado (for RTL design), and a CPU-centered approach, we provide a comprehensive analysis of the trade-offs involved.

The results demonstrate significant improvements in both computation speed and energy consumption when utilizing NMP, highlighting its potential in embedded systems where efficiency and performance are critical. Specifically, the NMP implementation showed faster execution times compared to the centralized CPU approach. Additionally, the comparison between the HLS and RTL implementations revealed insights into the development time and resource allocation, with HLS offering a more streamlined design process at the cost of some performance optimizations that can be achieved with hand-tuned RTL design. These findings confirm that NMP, coupled with modern design tools, is a viable strategy for enhancing the performance of embedded systems in data-intensive applications.

***Index Terms*—PIM, NMP, Processing In Memory, Near Memory Processing, HLS, High Level Synthesis.**

I. Introduction

THIS paper aims to investigate and explore modern approaches to addressing the von Neumann Bottleneck, with a particular focus on in-memory and near-memory processing techniques. In today's digital age, the consumption and generation of data have been growing exponentially, driven by the rapid advancements in machine learning, artificial intelligence, and data analytics tasks. These data-intensive applications demand more efficient and scalable computing architectures to handle the increasing volume and complexity of data.

Historically, the exponential growth in processing power, as predicted by Moore's Law, has driven advancements in computer architecture. However, as the physical limitations of transistor scaling become more apparent, traditional processor-centered architectures are encountering significant challenges in managing large datasets. One of the most critical issues is the need to continuously transfer data between memory and the central processing unit (CPU) for computation. This back-and-forth movement of data introduces substantial overhead and latency, particularly in scenarios involving big data. The constant shuttling of data not only slows down processing but also leads to higher power consumption, making traditional architectures increasingly inefficient.

To mitigate these issues, emerging technologies such as in-memory processing (PIM) and near-memory processing (NMP) are being explored as potential solutions to break the von Neumann Bottleneck. These approaches aim to minimize the need for data transfer by bringing computation closer to the memory, thereby reducing latency and energy consumption. In PIM, even minimal operations are proposed to be performed closer to, or even within, the memory itself. For instance, as discussed in [1], simple operations such as data copying and bitwise operations could be executed directly within DRAM, bypassing the need for data to be transferred to the CPU. This shift in processing paradigm could lead to significant improvements in computational efficiency and energy usage, especially in data-intensive applications.

Building upon these insights, this study proposes a practical exploration of near-memory processing by implementing a vector adder within an embedded System on Chip (SoC). The objective is to demonstrate, at a beginner level, the potential efficiency gains of processing closer to memory. Furthermore, the study will examine the differences between traditional design methodologies and high-level synthesis (HLS), utilizing tools such as AMD Vitis HLS and Vivado. Through this implementation, the study seeks to provide a hands-on understanding of the advantages of near-memory processing, contributing to the broader discourse on overcoming the limitations of traditional computing architectures in the face of growing data demands. .

II. Related Works

Several studies have explored the concept of processing data closer to or within the memory, aiming to mitigate the challenges posed by the von Neumann bottleneck. Among these, the work by [1] stands out as a seminal contribution. The authors of "Processing Data Where It Makes Sense" provide an in-depth analysis of how data-centric processing can reduce

Universidade Federal de Minas Gerais

latency and energy consumption, particularly in data-intensive applications. This paper serves as a crucial reference point for our study, as it lays the foundation for understanding the advantages of near-memory processing (NMP) and in-memory processing (PIM) architectures. Our work builds upon these insights, extending the exploration by implementing a near-memory vector adder on an embedded SoC platform, thereby providing a practical demonstration of the concepts discussed in [1].

[2] dives into various PIM technologies at the architectural level, exploring existing implementations, challenges such as data coherence and compatibility, and the future prospects of PIM. The study emphasizes the importance of PIM in addressing the growing demands of memory-intensive applications driven by advances in artificial intelligence, big data, and cloud computing.

in the context of new hardware solutions for in-memory operations [3] demonstrate significant advancements in processing within memory. The Ambit architecture enables bulk bitwise operations directly within DRAM, leveraging the internal structure of DRAM arrays to perform operations such as AND, OR, and NOT without transferring data to the CPU. This approach drastically reduces data movement, improving both speed and energy efficiency. The experiments show that Ambit can achieve a performance improvement of up to 32 times compared to conventional CPU-based methods for bitwise operations. Moreover, the energy savings are substantial, making Ambit a highly efficient solution for applications requiring bulk bitwise operations.

Expanding the research, [4] presents an innovative solution for accelerating neural networks using a memory-centric reconfigurable accelerator called NeuRAM3. This approach leverages 3D stacking technology, integrating multiple layers of memory and logic into a compact unit. The architecture enables computational operations, such as multiplications and additions, to be performed directly within the memory, significantly reducing the need to transfer data to external processors.

Several studies have explored the trade-offs between High-Level Synthesis (HLS) and Register Transfer Level (RTL) design approaches in hardware development. For instance, [5] conducts a detailed comparison between HLS and manually written RTL, focusing on aspects such as productivity, performance, and area efficiency. Similarly, [6] investigates the differences in FPGA-based hardware acceleration when using HLS versus hand-crafted RTL designs, highlighting the advantages and disadvantages of each method.

This paper also seeks to explore the potential pros and cons of developing hardware using HLS compared to traditional RTL design. By implementing a vector adder using both methodologies, we aim to provide a practical analysis of how each approach impacts design time, resource utilization, and overall performance. This study contributes to the ongoing discourse by offering insights into the effectiveness of HLS in comparison to RTL, particularly in the context of near-memory processing architectures.

III. Methodology

The ZedBoard Zynq-7000 ARM/FPGA was the SoC used to implement the vector adder. The choice of this specific board, comes to the necessity of using both the silicon processor and the reconfigurable characteristic provided by the integrated FPGA. This combination enables the development of complex embedded systems where the ARM processor runs software while the FPGA can be configured for custom hardware acceleration, signal processing, or other specialized tasks.

For the vector adder design, the softwares used was the AMD's Vitis HLS and Vivado. The IP responsible for the near memory processing, capable of performing the operations and the interface with the processor and the memory, was implemented in Verilog .This Verilog module implements an AXI-lite interface for a near-memory processing (NMP) unit, allowing external control via a standard AXI-lite bus. It manages several internal registers to handle different operations, such as reading, writing, and vector addition, directly in memory. The module includes a state machine that governs the transitions between various operational states: idle (waiting), reading, writing, and performing vector operations. The BRAM (Block RAM) is used to store vectors and intermediate results, enabling efficient vector addition by accessing data directly in memory.

The module interacts with the BRAM through dual ports, allowing simultaneous read and write operations for vector processing. The vector addition is done by fetching vector elements from two source addresses, adding them, and storing the result at a destination address . This process is repeated for a specified vector length, with the state machine iterating over the vector elements.

The design focuses on minimizing latency and improving data processing efficiency by leveraging near-memory processing, where operations are performed closer to where the data resides, reducing the need for extensive data movement across the system. The use of AXI-lite provides a standardized interface for controlling this custom logic, making it easier to integrate with other system components. The module is particularly suited for accelerating memory-bound tasks, such as those involving large-scale vector computations, within an embedded system or FPGA-based architecture.

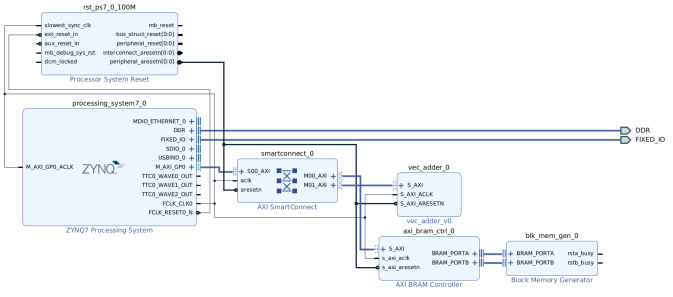

Fig. 1: Block diagram implementation in Vivado

The testing interface was designed in C, it interacts with the NMP unit using Xilinx AXI interface. The code contains various functions to perform different kinds of tests to analyze the performance and accuracy of the NMP system. It goes

from the first function that performs a simple read/write test on the AXI interface to a register and reading it back,a sanity test write two values in different memory addresses, verifies it and perform the addition of those two values before writing it into a third address, then the function checks if the result matches the expected sum.

In addition to checking the correct operation and functionality of the feature, this code also implements functions to collect metrics to base the performance of the application. The timing tests measures the time required to perform vector addition operations on pairs of vectors of increasing lengths, using the specific Xilinx function XTimeGetTime. The results are printed to the console, showing how long it took to add each vector pair. This output is the main metric used for evaluating the efficiency of the NMP system, particularly in terms of how well it scales with increasing vector sizes. The results are shown in the next section.

IV. Results and Discussion

Figure 2 shows the operation latency for software vector addition over the processing system memory on O0 and O3 optimization, software vector addition over the programmable logic memory and hardware vector addition over the programmable logic.

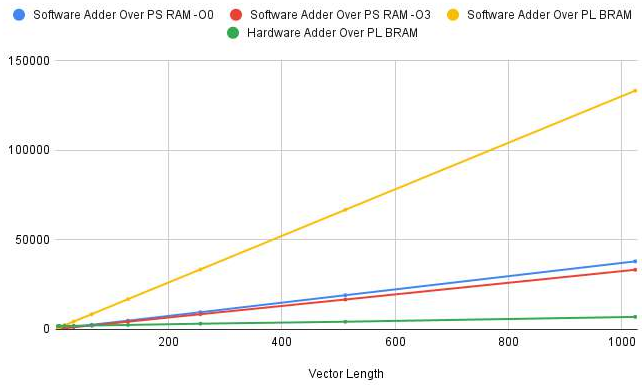

Fig. 2: Latency [ns] x Length [words] for the four measured operations

For each plot, the trend line $y = ax + b$ results in $R \approx 1$. Using this approximation, Table 1 show that the hardware adder is 26 times faster than the software adder over a BRAM, over 7 times faster than non-optimized software addition over the processing system RAM and over 6 times faster than optimized software addition over the processing system RAM.

Operation	1	2	3
a_i/a_0	26	7.4	6.4

TABLE I: Ratio between each operation and the hardware addition

Table 2 shows the percentage of the total resource utilization due to the adder IP.

It's worth noting that, as the current design demands highly timing driven synthesis and elaboration, area and utilization optimization are limited. Therefore, it's essential to carefully evaluate the trade-off between resource consumption and latency reduction based on the specific application, as this metric is neither negligible nor impeding. Nonetheless, the proposed design demonstrates superior scalability compared to traditional operations, with minor increase in latency for longer vectors. However, for short vectors, the accelerator's prepare and trigger operations result in a higher latency when compared to standard paradigms.

Resources *	Design Total	IP	% used for IP
Slice LUTs	980	318	32
Slice Registers	1192	430	36
Slice	442	147	33
LUT as Logic	974	318	33
LUT as Memory	6	0	0
Block RAM	6	4	67
BUFGCTRL	7	0	0

TABLE II: Design's resource utilization

V. Future Work

To conclude, this research underscores the potential of near and in-memory processing (NMP) solutions as a promising approach to overcoming the von Neumann bottleneck, despite the challenges associated with these technologies. The next phase of this work will focus on enhancing the transparency and accessibility of data read and write operations, which is crucial for fully harnessing the potential of NMP.

As highlighted in related studies, numerous operations have been successfully implemented within memory, each demonstrating the potential benefits of NMP. By expanding this project to include additional operations, we aim to further showcase the versatility and power of these technologies. This exploration will not only deepen our understanding of NMP but also provide valuable insights into its practical applications.

Power consumption remains a critical issue linked to the von Neumann bottleneck, with significant energy wasted due to the extensive data movement between memory and processors. This work proposes the development of a dedicated platform for experimentally tracking system power consumption. By quantifying the energy savings achieved through reduced data travel distances, the study aims to provide concrete evidence of the efficiency gains offered by near-memory processing.

In summary, the combination of enhancing data operations, expanding in-memory functionalities, and rigorously evaluating power consumption will provide a comprehensive understanding of the benefits and limitations of NMP technologies. This work not only addresses the immediate challenges posed by the von Neumann bottleneck but also paves the way for future advancements in energy-efficient computing, which is increasingly vital in the context of big data, AI, and other data-intensive applications.

VI. Conclusion

To conclude, this research highlights the effectiveness and potential of Near Memory Processing (NMP) as a solution to the long-standing von Neumann bottleneck, particularly in the

context of embedded systems. By implementing a hardware vector adder using both High-Level Synthesis (HLS) and Register Transfer Level (RTL) methodologies within the AMD Vitis and Vivado environments, we were able to demonstrate significant efficiency gains in terms of both computation speed and energy consumption. The comparison between the NMP-based approach and traditional CPU-centered processing further underscores the advantages of reducing data movement by processing closer to the memory.

The results of this study suggest that NMP not only accelerates processing tasks but also reduces the overall power consumption of the system, making it an attractive option for applications where performance and energy efficiency are critical. Additionally, the use of modern development tools such as Vitis HLS simplifies the design process, although there are trade-offs in terms of the fine-tuned performance that can be achieved with manual RTL design.

Future work should focus on expanding the range of operations that can be performed using NMP and exploring the integration of NMP with other emerging technologies in embedded systems. Moreover, continued efforts to quantify the benefits of NMP in real-world applications will be essential to driving broader adoption of this approach in the industry. Overall, this research contributes to the growing body of evidence supporting NMP as a viable and efficient strategy for modern computing challenges.

REFERENCES

[1] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors and Microsystems*, vol. 67, pp. 28–41, 6 2019.

[2] X. Zou, S. Xu, X. Chen, L. Yan, and Y. Han, "Breaking the von neumann bottleneck: architecture-level processing-in-memory technology," *Science China Information Sciences*, vol. 64, 6 2021.

[3] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," pp. 273–287, 2017.

[4] R. Karam, S. Paul, R. Puri, and S. Bhunia, "Memory-centric reconfigurable accelerator for classification and machine learning applications," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, may 2017. [Online]. Available: https://doi.org/10.1145/2997649

[5] Y. Liang, K. Rupnow, Y. Li, D. Min, M. N. Do, and D. Chen, "High-level synthesis: Productivity, performance, and software constraints," *Journal of Electrical and Computer Engineering*, 2012.

[6] M. Gurel, *A comparative study between rtl and hls for image processing applications with fpgas*. University of California, San Diego, 2016.

Eduardo Henrique

Renan Neves