OmniMemory-Centric Organization & Operation Kernel (OOOK), A Near-memory in-bank Compute Architecture within DRAM

Barak Binyamin, *Undergrad, Rochester Institute Of Technology*

[1]

Abstract— **Common computing systems are based on the Von Neumann Architecture, placing the main processing unit a distance away from the memory. As a side effect of this decision, throughput is limited, the effect of which is known as the von Neumann Bottleneck. The gains outway the alternatives for general applications, but fall short during memory or computationally intensive tasks. To handle more intensive tasks, additional hardware called accelerators, also known as co-processors, can be added to accelerate particular workloads. A commonly used coprocessor is a Graphics Processing Unit (GPU), which assists with graphics processing applications. The accelerator discussed in this paper, OmniMemory-Centric Organization & Operation Kernel (OOOK), is a Near Memory Processor (NMP), which assists in memory constrained workloads. NMP's take advantage of throughput gains when computing is placed closer to the memory hardware, typically at the loss of some memory capacity. Applications of NMP's usually include AI, in memory databases, & real-time analytics. Components of OOOK were simulated or analytically represented and compared to a Core i5 Processor and Intel Iris Plus Graphics 650 GPU. The results show that OOOK performs better than a GPU, but does not perform better than a CPU in the categories analyzed.**

I. INTRODUCTION

Near-memory computing offers a paradigm shift in computing architecture by integrating computational resources directly into or in close proximity to memory units. This proximity reduces data movement overhead and latency, enabling improvements in performance and energy efficiency. By processing data where it resides, near-memory computing minimizes the need for frequent data transfers between memory and processing units, which are common bottlenecks in traditional von Neumann architectures. This approach is particularly beneficial for data intensive workloads such as machine learning, analytics, and graph processing, where large datasets often exceed the capacity of on-chip caches and require frequent accesses to external memory. [1] This paper considers one possible near-memory implementation by branching of existing memory hardware known as SDRAM, and taking inspiration from Array Processors.

II. BACKGROUND

A. *Array vs Vector Processing*

Single Instruction, Multiple Data (SIMD) and vector processors represent two distinct approaches to parallel processing in computing architectures. SIMD processors execute the same instruction on multiple data elements simultaneously, while vector processors specialize in efficiently processing arrays or vectors of data. SIMD processors achieve parallelism by executing a single instruction across multiple Processing Elements (PE's), each operating on different data elements. This approach is beneficial for tasks that exhibit data-level parallelism, such as multimedia processing, signal processing, and scientific simulations. By performing identical operations on multiple data elements in parallel, SIMD processors can achieve significant speedups compared to scalar processors. SIMD processors are often referred to as array processors as well. In current general computing architectures like x86, SIMD capabilities are available, but somewhat limited, typically supporting operations on up to 4 values simultaneously. Conversely, Graphics Processing Units (GPUs) provide similar capabilities but may exhibit reduced performance over shorter ranges due to their vector processor architecture. [2] OOOK adopts a SIMD approach.

B. *Considering Floor Planning*

When designing a processor, numerous factors must be weighed, including the complexity & placement of computational logic. Proximity to memory resources enhances data throughput, leading to higher data rates. However, this proximity also incurs increased costs due to limitations in current fabrication processes, especially as logic complexity increases. Footprint size is another critical consideration. Larger chip designs are costlier due to fabrication constraints and electrical properties. Moreover, as data transport across each millimeter becomes less reliable, achieving reliability

[1]This work was supported in part by the Rochester Institute of Technology as a final project for Purab Sutradhar & Dr. Amlan Ganguly in CMPE-789 Memory Centric Computing, May 2024.

First: Barak Binyamin is with the Rochester Institute of Technology, Rochester, NY 14624 USA (e-mail: bb5369@g.rit.edu).

Color versions of one or more of the figures in this article are not currently available at any known location

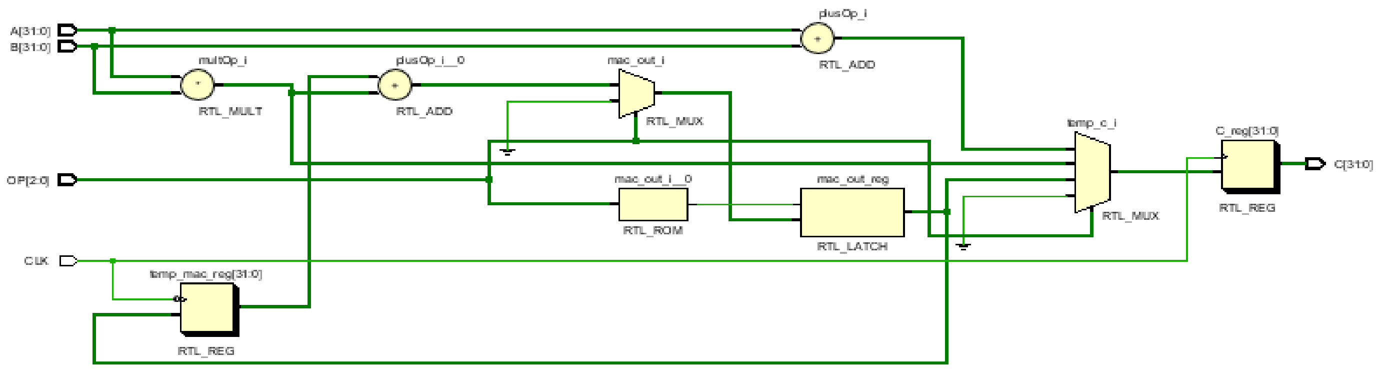


Fig. 2. PE Architecture Schematic

requires additional circuitry or compromises on transfer speeds. The tradeoff in floorspace involves balancing the complexity of logic, the number of PE's, memory, and input/output (IO) hardware. Traditionally, IO accounts for approximately half of the chip's footprint. [3]

III. PROPOSED METHOD

A. *Design Overview*

The architecture of OOOK utilizes DDR4 x4 (4GB 16 bank SDRAM) as its foundation. Please refer to Fig. 1 for visual reference.

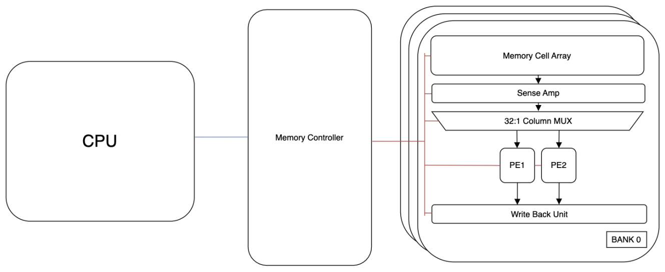


Fig. 1. Architecture of OOOK

Each of the 16 banks in the standard DDR4 memory is altered to contain 2 PE's, denoted as PE1 and PE2, as well as read, write, control and logic conversion circuitry. These PE's are equipped to perform addition, multiplication, and multiply-accumulate operations. Controlling the accelerator can be accomplished by enhancing the capabilities of a conventional memory controller. As depicted in the diagram, the memory controller takes charge of managing memory requests and instructing operations. In addition it is assumed a custom driver would be developed to utilize the extended functionalities of this custom memory controller. This approach mirrors the management of other accelerators.

The architecture of each PE can be seen in Fig 2. The inputs to each PE include two 32-bit inputs, a 3 bit opcode, and a clock, while the output of each PE is a 32-bit result. Simulation of this design can be seen in the results section.

Each of the processors respond to memory controller issued microinstructions, of which can be seen in Table I.

Table I

PE Operation Codes & Instructions

Code	Instruction
000	ADD, out=A+B
001	MUL, out=AxB
010	MAC, out=AxB+REG
011	CLR, out=REG
100-111	NOP, out=0x00000000

The instructions listed in Table I dictate the operation and status of each PE. Here, inputs denoted as A and B both originate from DRAM and are routed through read and conversion circuitry, both under the control of the memory controller. This approach differs from Newton in the lack of an external buffer for half of the inputs, as well as the implementation of multiple instructions per bank.[4] Notably absent from the table is an instruction for data pass-through, which is currently not implemented. However, it is anticipated that in the future, this functionality will be designated by operation code 111.

As mentioned previously, the architecture of this design adopts a SIMD approach, as such, commands from the CPU would be designated to conduct the same operation, on multiple data elements. One possible issue this design might face is data locality. Typically data belonging to some data structures can be scattered across different banks, but this concern could be handled or flagged by a dedicated compiler, or by a deliberate design decision in construction programs. Runtime locality issues could be detrimental to functionality and performance, as such, the memory controller could be equipped with safety checks to reorganize data before issuing commands. These corrections would likely result in a significant deterioration in performance.

B. *Design Decisions*

Each PE is implemented using Lookup Tables (LUTs), simplifying floor plan estimations and ensuring design extensibility. Simulation and synthesis with Xilinx Vivado, as can bee seen in Appendix 1 & 2, indicated that each PE requires 128 LUTs, 96 Flip Flops, and 3 DSPs. Converting Flip Flops to LUTs at a ratio of 1 to 1.5 yields an equivalent of 144 LUTs. Similarly, DSPs, which handle the multiplication, addition, and multiply-accumulate operations, translate to a total of 2802 LUTs. Thus, the total number of LUTs per PE is approximately 3074.

Estimating the floor plan involves assessing a 6-input LUT, which typically utilizes around 384 transistors. With DDR4 employing 24nm technology, each transistor can be approximated as a 24nm x 24nm square. Consequently, a 384-transistor 6-input LUT occupies roughly 924nm^2. Multiplying this by the total number of LUTs per PE yields an area of 280,376nm^2, or 0.280376mm^2.

Given that the area of a bank is assumed to be 1.1mm^2, it's estimated that two PE's would occupy approximately half the memory in a bank. Therefore, the decision was made to incorporate 2 PE's per bank. It's assumed that additional components such as multiplexers, write-back mechanisms, and control hardware are factored into these estimates.

Consequently, the resulting memory capacity is reduced from 4GB to 2GB, while enabling the enhanced feature of having 16 banks with 2 reconfigurable PE's each, totaling 32 PE's adjacent to the memory.

IV. RESULTS

A benchmark is required to compare the performance of OOOK, an Intel Core i5 processor, and Intel Iris Plus Graphics 650 GPU. The benchmark, developed through analytical methods, focuses on evaluating a scenario where a large number of sensors transmit extensive data to a server. This choice was made because server-side database operations represent a typical use case for near-memory devices. In this scenario, the server is tasked with performing minor operations on numerous collected readings. Given that the operations in this scenario involve multiplication, addition, and multiply-accumulate, an effective benchmark involves assessing the impact of adding a constant value (such as correcting skewed readings due to configuration errors) to each sensor value against both a CPU and GPU. The benchmark evaluates scenarios involving 1 million, 10 million, and 100 million 32-bit unsigned sensor values, assuming they are already stored in DRAM.

A. *Timing Parameters for OOOK*

The PE implemented in OOOK was simulated in Xilinx Vivado to determine the minimum delay of the longest logic path. The delay was determined to be 4.6ns as can be seen in Appendix 3. The design's additional features were subjected to analytical modeling. Some noteworthy values, extracted from Micron's datasheet on DDR4 x4 hardware [5], include CL (Column Strobe Latency) at 14.16ns, tRCD (Row-to-Column Delay) at 14.16ns, and tRP (Row Precharge Delay) at 14.16ns. CL represents the latency of column strobes, tRCD denotes the duration for transferring data from DRAM cells to sense amplifiers during row activation, and tRP signifies the precharge delay involved in accessing a memory row, deactivating it, activating the next row, and initiating access to the subsequent row. These values can be summed to calculate the memory access time. So it can be estimated that memory access time is about 42.48ns.

Together a read, an operation and a write would likely take a maximum of 42.48ns + 4.6ns + 42.48ns = 89.56ns. If parallelism is taken advantage of, this time can be cut by a factor of 32 due to each of the PE's. This results in an average of 89.56ns/32=2.799ns per full operation.

B. *Timing Parameters for Intel I5 Core*

An Intel I5 Core is typically clocked at 1.2GHz and pipelined as well. With the introduction of SIMD, it's now possible to perform up to 8 arithmetic operations per chip, 4 per core. A read to a register from ram typically takes 3 clock cycles, an addition operation takes 1 clock cycle, and a write to ram takes 3 clock cycles [6], resulting in a total time 7 clock cycles per 8 full operations. Given that one clock cycle takes about 0.83ns (1/1.2GHz), the total time for 8 operations would be about 7*.83ns = 5.81ns per 8 operations. This results in an average of 5.81ns/8 = 0.726ns per full operation.

C. *Timing Parameters for Intel Iris Plus Graphics 650 GPU*

The Intel Iris Plus Graphics 650 has 8 ALU units per of its 48 execution units, So 48x8 = 384 ALU's that can all run in parallel. The intel iris typically runs at 300 Mhz when it's not overclocked [7]. Assuming a read to a register takes 3 cycles, an operation takes 1 cycle, and a write to ram takes 3 cycles, it would take 7 cycles per 384 operations (if alu's used simultaneously). Given that a clock cycle would be 333.33ns (1/300MHz), the total time for 384 operations would be about 7*333.33ns = 2333.31ns. This results in an average of 2333.31ns/384 = 6.076ns per full operation.

D. *Final Benchmark*

Given the calculations made, a table was deduced to show the analytical results of 1,10,100 million operations for all three devices. The results can be seen in Table II.

Table II

Benchmark Results

Arch.	Est. 1 million Additions	Est. 10 million Additions	Est. 100 million Additions
OOOK	2.799 ms	27.99 ms	279.9 ms
CPU	0.726 ms	7.26 ms	72.6 ms
GPU	6.076 ms	60.76 ms	607.6 ms

Table II shows that OOOK is estimated to perform better than a GPU but worse than a CPU for the task of running a simple operation on a large quantity of in ram values.

V. CONCLUSION

In conclusion, while common computing systems based on the Von Neumann Architecture have historically provided sufficient performance for general applications, they often face limitations, particularly during memory or computationally intensive tasks, due to the von Neumann Bottleneck. To address these challenges, additional hardware accelerators or co-processors, such as Graphics Processing Units (GPUs), have been utilized to offload specific workloads and enhance performance. However, this paper introduces a novel accelerator, the OmniMemory-Centric Organization & Operation Kernel (OOOK), designed as a Near Memory Processor (NMP) to cater to memory-constrained workloads. NMPs leverage the proximity to memory hardware to achieve throughput gains, albeit with some trade-offs in memory capacity. Through simulation and analytical comparison against a Core i5 Processor and Intel Iris Plus Graphics 650 GPU, the results indicate that while OOOK outperforms the GPU, it does not surpass the CPU in the categories analyzed. These findings suggest the potential of NMPs like OOOK to address specific computing challenges, particularly in memory-constrained environments, but underscore the continued importance of traditional CPU architectures for certain tasks.

VI. REFERENCES

[1] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, "*Near-memory computing: Past, present, and future,Microprocessors and Microsystems*", Volume 71, 2019, ISSN 0141-9331, https://doi.org/10.1016/j.micpro.2019.102868. (https://www.sciencedirect.com/science/article/pii/S0141933119300389)

[2] Dr. Onur Mutlu*, "Computer Architecture: SIMD and GPUs", 2013*

[3] Dr. Amlan Ganguly, Purab Sutradhar, "*CMPE- 789, Memory-Centric ArchitecturesLecture 5: Memory Organization*", 2024

[4] M. He et al., "*Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning*," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Athens, Greece, 2020, pp. 372-385, doi: 10.1109/MICRO50266.2020.00040 https://ieeexplore.ieee.org/abstract/document/9251855

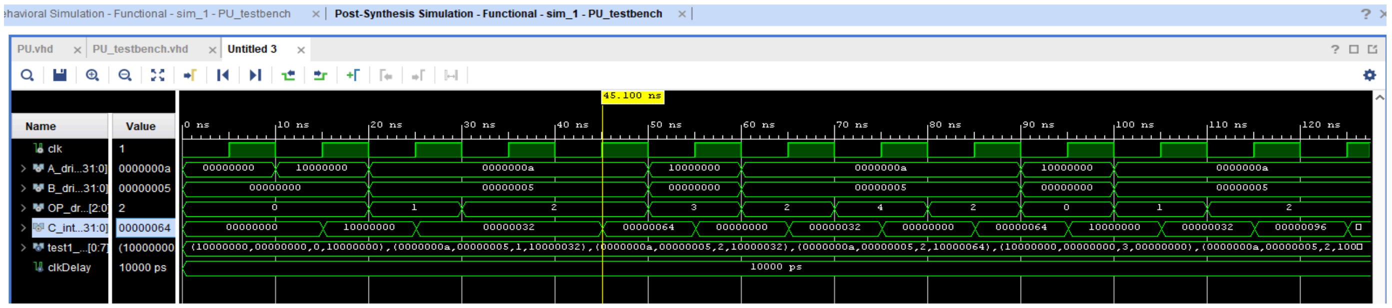
[5] Micron, "*DDR4 SDRAM - MT40A2G4, MT40A1G8, MT40A512M16*", 2023

[6] Notebookcheck, "*Intel-Iris-Plus-Graphics-650 statistics*" www.notebookcheck.net, 2024

[7] Intel, *intel-core-i5-750-processor statistics*, ark.intel.com

[8] Barak Binyamin, "*OOOK, A Near-memory in-bank Compute Architecture within DRAM*", https://github.com/BarakBinyamin/oook, 2024

APPENDIX I

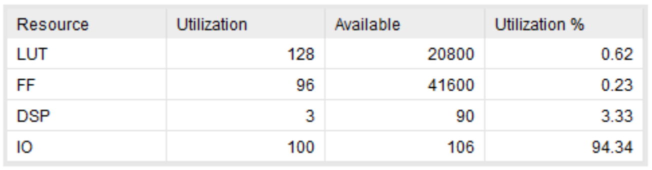

ehavioral Simulation - Functional - sim_1 - PU_testbench
Post-Synthesis Simulation - Functional - sim_1 - PU_testbench
PU.vhd
PU_testbench.vhd
Untitled 3
45.100 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns

Name	Value
clk	1
A_dri...31:0]	0000000a
B_dri...31:0]	00000005
OP_dr...[2:0]	2
C_int...31:0]	00000064
test1_...[0:7]	(10000000
clkDelay	10000 ps

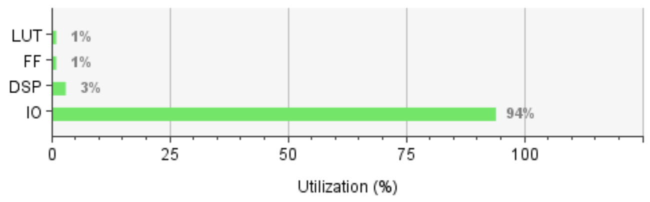
```
A:    00000000 | 10000000 | 0000000a | 10000000 | 0000000a | 10000000 | 0000000a
B:    00000000 | 00000005 | 00000000 | 00000005 | 00000000 | 00000005
OP:   0 | 1 | 2 | 3 | 2 | 4 | 2 | 0 | 1 | 2
C:    00000000 | 10000000 | 00000032 | 00000064 | 00000000 | 00000032 | 00000000 | 00000064 | 10000000 | 00000032 | 00000096
test1: (10000000,00000000,0,10000000),(0000000a,00000005,1,10000032),(0000000a,00000005,2,10000032),(0000000a,00000005,2,10000064),(10000000,00000000,3,00000000),(0000000a,00000005,2,100
clkDelay: 10000 ps
```

A Waveform showing functionality of PE

APPENDIX II

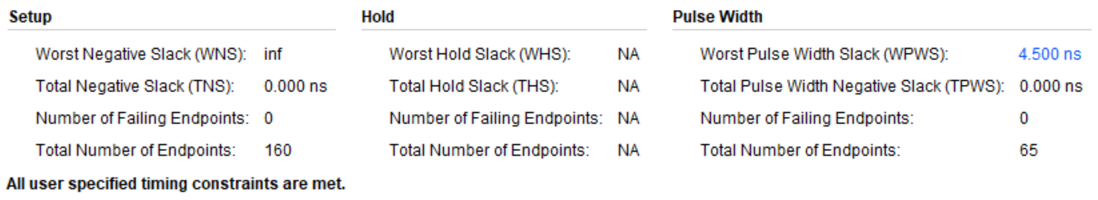


Resource	Utilization	Available	Utilization %
LUT	128	20800	0.62
FF	96	41600	0.23
DSP	3	90	3.33
IO	100	106	94.34



Resource Utilization of Synthesized PE

APPENDIX III



Setup

Worst Negative Slack (WNS):	inf
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	160

Hold

Worst Hold Slack (WHS):	NA
Total Hold Slack (THS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	65

All user specified timing constraints are met.

Timing Summary of PE